Accepted for publication in IEEE-PAMI

Incorporating Season and Solar Specificity into Renderings made by a NeRF Architecture using Satellite Images

Michael Gableman and Avinash Kak

Abstract—As a result of Shadow NeRF and Sat-NeRF, it is possible to take the solar angle into account in a NeRF-based framework for rendering a scene from a novel viewpoint using satellite images for training. Our work extends those contributions and shows how one can make the renderings season-specific. Our main challenge was creating a Neural Radiance Field (NeRF) that could render seasonal features independently of viewing angle and solar angle while still being able to render shadows. We teach our network to render seasonal features by introducing one more input variable — time of the year. However, the small training datasets typical of satellite imagery can introduce ambiguities in cases where shadows are present in the same location for every image of a particular season. We add additional terms to the loss function to discourage the network from using seasonal features for accounting for shadows. We show the performance of our network on eight Areas of Interest containing images captured by the Maxar WorldView-3 satellite. This evaluation includes tests measuring the ability of our framework to accurately render novel views, generate height maps, predict shadows, and specify seasonal features independently from shadows. Our ablation studies justify the choices made for network design parameters.

Index Terms—NeRF, Satellite Imagery, Image-based rendering, Time-varying imagery

1 Introduction

Our goal in this paper is to extend the NeRF formalism to render season-specific images with correct shadows from multidate and multiview satellite images. What we seek to accomplish is exemplified by the results shown in Fig. 1 in which the columns represent different seasons, and the eight rows are the eight different Areas of Interest (AOIs) we tested from the 2019 IEEE GRSS dataset [1]. These rendered images are from viewpoints, solar angles, and times of the year not used for training the NeRF. The examples shown in Fig. 1 use the time of the year to alter the seasonal features while keeping the viewing and solar angles constant. In addition, we provide an animated example of changing viewpoints with fixed time of year and solar angle on our GitHub page[1], where our code is also available. Furthermore, the figure shows that the seasonal features in the rendered image correlate with the time of the year given as input.

The original NeRF formulation, presented by Mildenhall et al. [2], did not have mechanisms to account for changes in an image unrelated to changes in viewing angle. This restriction on NeRFs was relaxed by Martin-Brualla et al. [3] and by Derksen and Izzo [4]. Both works accounted for non-viewing angle image changes by introducing additional input into the NeRF. NeRF in the Wild (NeRF-W) [3] accomplished this by using two learnable embeddings for each training image. One embedding captured the *appearance* of the image, in essence, the information that could explain alterations in the manifestation of objects in the scene. The other embedding captured the *transient* portions of the image. These portions of the image contain moving objects that should not appear in a rendered image. To avoid a trivial solution where every point in every image is deemed transient, NeRF-W uses the Bayesian learning framework of Kendall et al. [5]. More details on NeRF-W are provided in Section 2.

Derksen and Izzo [4] noted that a large amount of image variation in satellite images is caused by changing light conditions and factored this observation directly into the NeRF model. They used the twin concepts of *solar visibility* and *sky color* as two additional outputs of their NeRF framework, known as Shadow NeRF (S-NeRF). The solar visibility computes the amount of direct sunlight at each point in the model, and the sky color accounts for global light conditions and models the background illumination. To compute these additional outputs, S-NeRF takes a solar angle as input to the network. S-NeRF limits the influence of the solar angle by only allowing it to influence the rendering in a manner consistent with how light interacts with surfaces. More details on how S-NeRF uses solar visibility and sky color are provided in Section 2. In addition to outputting solar visibility and sky color, S-NeRF replaces the computation of color with the computation of albedo color.

What has motivated the work reported in NeRF-W [3] and S-NeRF [4] is exactly our motivation as well. Satellite images of the same scene may be captured months apart. As a result, two satellite images may exhibit different seasonal features and shadows while containing transient objects. We seek to create a NeRF framework capable of accounting for the seasonal features and shadows while simultaneously discarding portions of the training images containing transient objects. Transient objects, such as cars moving along a road, appear in only a single image and should not be allowed to influence the final rendering. Seasonal features manifest themselves differently during different times of

1. https://github.com/EnterpriseCV-6/Season-NeRF.git

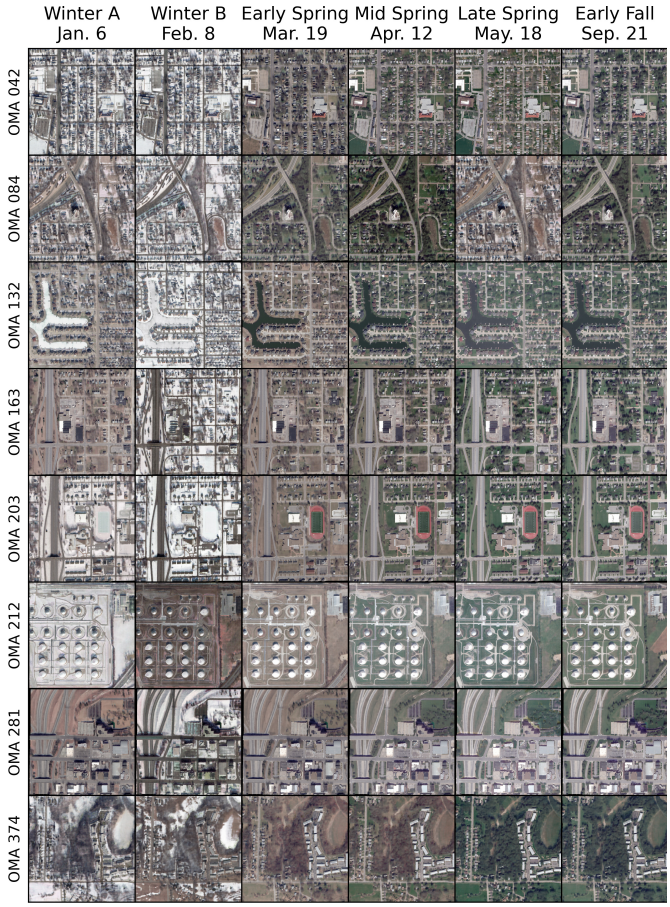

Fig. 1. Results of generating images on eight regions. Each view is generated at 26 degrees off-nadir with a 45-degree azimuth angle. The solar angle is at a 50 degrees elevation and 135-degree azimuth. Images in the same columns have the same time input.

the year. For example, images may contain large swathes of green foliage from late spring to early fall. However, it is common to see snow and brown foliage from late fall to early spring. Shadows are caused by objects occluding light rays from the sun to a world point.

The framework we present in this paper, which we call Season-NeRF, computes the seasonally adjusted albedo, density, sky color, and solar visibility as a function of position within the model, solar angle, and time of the year. The solar angle and the time of the year are provided in the metadata associated with the satellite image. We wish to ensure that any changes in the time of the year only change the seasonal features, and changes to the solar angle only change shadows. Without limitations, the time of the year and solar angle can manipulate the rendering in ways unassociated with their respective features. To constrain the use of the additional inputs, we only allow the time of the year to alter the value of the seasonally adjusted albedo, and we only allow the solar angle to alter the values of the sky color and solar visibility. We also limit how much the time of the year can alter the seasonally adjusted albedo and how solar visibility and sky color interact with seasonally adjusted albedo. More details on how the time of the year influences the albedo color are provided in Section 3.1, and we describe the computation of shadows in Section 3.2.

Accurate placement of shadows within an image and correct novel view rendering require an accurate density estimation. To encourage accurate density estimation, we provide a height map to Season-NeRF during the early stages of training. Our use of structural information is similar to the technique used by Deng et al. [6]. In a depth-supervised NeRF, traditional computer vision methods extract 3D world points. These points are used to provide depth supervision during training. We use this height map to guide the density computation in the early stages of training. *In the later stages, we stop using the height map* as the map is noisy, and the partially trained Season-NeRF can provide more accurate structural estimations. More details on our use of a height map are provided in Section 3.4.

In addition to solar and seasonal variation in images, we also account for transient objects. Transient objects are moving objects that appear in single images and should be excluded from the final model. NeRF-W uses learned image-specific embeddings to compute the network's uncertainty of the predicted color. The network learns to assign regions with transient objects a higher uncertainty. We propose an alternative that removes the need to learn image-specific embeddings. The loss proposed by Barron [7] allows the network to reduce the influence of transient objects automatically. This loss allows us to use a simpler architecture, removes the need to compute and account for an uncertainty field, and simplifies the rendering process during training.

When evaluating Season-NeRF's performance, we show that Season-NeRF can render high-quality images, stably render seasons, and correctly specify seasons and shadows. Due to the limited data, we can only withhold four images from each region for testing. Of the four images withheld for testing, we ensure that three of these images span unique seasons. The fourth image is selected to ensure that the set of testing images in each AOI spans a reasonable range of solar and viewing angles. The quality of the rendering is determined by comparing the testing images with rendered images at the same viewing and solar angle. In addition to image similarity, we generate a height map of the region and compare that to the lidar data associated with the AOI. The accuracy of our estimated shadow masks is computed by comparing the estimated mask to the exact mask Season-NeRF computes. Stably rendering a season means that the seasonal features should only change due to changing the time of the year, not because of changes in the solar or the viewing angles. To measure this, we render images across a wide range of viewing angles and solar angles. The extent of the change introduced by altering the viewing and solar angle is measured against the extent of the change introduced by altering viewing time. Correctly specify seasons (i.e., getting a winter image for a time input during the winter or a spring image for a time input during the spring) is confirmed by rendering images across the entire year and visually confirming the correct seasonal characteristics.

The remainder of this paper is organized as follows, Section 2 discusses works related to our approach. In Section 3, we provide details on the specifics of our approach. Section 4 contains the results of Season-NeRF when applied to eight different regions and an ablation study. We break our analysis into three subsections. In Section 4.1, we show how Season-NeRF can render novel views. Then, in Section

4.2, we show how Season-NeRF can generate a height map and predict shadows, and in Section 4.3, we show the seasonal stability and specificity of Season-NeRF. In addition, within each subsection, we show how different components contribute to the accuracy of the final model. Our results and conclusions are summarized in Sec. 5.

2 Related Works

Neural Radiance Fields (NeRFs) have become a popular tool for novel view reconstruction since they were used by Mildenhall et al. [2]. Since this work, NeRFs have been used with success in many situations. Work with NeRFs generally falls into three categories, improved novel view generation [3], [4], [8], [9], [10], [11], [12], [13], [14], reduced reliance on camera parameters [15], [16], and faster model generation [6], [17], [18], [19]. While all these areas are relevant to understanding NeRFs, we are primarily interested in improved novel view generation, specifically for satellite images.

A NeRF computes a density, ρ[2], and a color, C, for every point within a 3-dimensional world space. This computation is carried out by a neural network trained on pixel-ray pairs. Every pixel has a known color, and its corresponding ray is the preimage of the pixel in the world space. The NeRF uses a ray to render a pixel's color by considering the color and density of every point along the ray. This computation requires integration along the ray. However, such an approach is not practical, so the quadrature rule is used to approximate the integral along the ray. The rendered color of a pixel with a preimage ray χ containing points $\vec{X}_0, \vec{X}_1, \cdots, \vec{X}_n$ is

$$Col\left(\chi\right) = \sum_{\vec{X}_i \in \chi} C\left(\vec{X}_i\right) P_S\left(\chi, \vec{X}_i\right), \tag{1}$$

where P_S is the probability that $\vec{X}_i$ is a surface point for ray χ, C is the color output by the NeRF at $\vec{X}_i$, and Col is the rendered color of the pixel. A point on a ray is the surface point for the ray if the point exists (that is, the space is occupied) and the point is visible along the ray. Furthermore, a point on a ray is visible along the ray if no prior points along the ray exist. Thus,

$$P_S\left(\chi, \vec{X}_i\right) = P_E\left(\vec{X}_i\right) P_V\left(\vec{X}_i, \chi\right), \tag{2}$$

$$P_E\left(\vec{X}_i\right) = 1 - e^{-\rho\left(\vec{X}_i\right)\delta_{\vec{X}_i}}, \tag{3}$$

and

$$P_V\left(\vec{X}_i, \chi\right) = e^{-\sum_{j=0}^{i-1} \rho\left(\vec{X}_j\right)\delta_{\vec{X}_j}} \tag{4}$$

where $\delta_{\vec{X}_i}$ is the distance between the points $\vec{X}_i$ and $\vec{X}_{i+1}$ and ρ is the density at $\vec{X}_i$[3]. Initial work by Mildenhall et al. [2] suggested using a weighted stratified sampling scheme for determining points along a ray; however, Mari et al. [9] use a uniform stratified sampling scheme. We are not aware of any work comparing the two schemes and use uniform stratified sampling throughout our work.

The work of Derksen and Izzo [4] provided a means for a NeRF to account for shadows in a scene. They accounted for shadows by expanding NeRF's outputs to include solar visibility (S_{vis}) and sky color (sky), which were computed by considering the solar angle. Their variant of NeRF is known as S-NeRF. Rather than outputting color, S-NeRF outputs albedo color (A). S-NeRF also modifies the rendering process to incorporate the new output terms. The color portion C from Equation 1 is replaced with

$$C\left(\vec{X}, \vec{w}\right) = \left(S_{vis}\left(\vec{X}, \vec{w}\right) + \left(1 - S_{vis}\left(\vec{X}, \vec{w}\right)\right) \text{sky}\left(\vec{w}\right)\right) * A\left(\vec{X}\right), \tag{5}$$

where $\vec{w}$ is the solar angle. For S-NeRF to correctly render shadows, the model trains with two types of rays, *image rays* and *solar rays*. Image rays are associated with a pixel from the training data and a solar angle. They are used to ensure the network accurately renders scenes. Solar rays are not associated with the training data, and the solar angle associated with a solar ray is negative one times the direction of the solar ray. Solar rays are used to ensure that the solar visibility output by the network is consistent with the density output of the network. By adding these modifications, S-NeRF can account for shadows and changing solar conditions during training and evaluation. Allowing the sky color to vary with the solar angle allows S-NeRF to account for changes in the image's appearance caused by different atmospheric effects. Furthermore, S-NeRF uses sinusoidal representation networks (SIRENs) as described in Sitzmann et al. [20].

Another method for accounting for changes in appearance was suggested in Martin-Brualla et al. [3]. They created a modification to NeRF known as NeRF-W. In NeRF-W, each training image is associated with a learned embedding. The network uses this embedding to output a density and color adjustment for each image, which is used during training to account for features unique to the training image. The density and color adjustments are ignored during the evaluation. Furthermore, NeRF-W outputs an uncertainty estimate β in addition to the color and density adjustments. Drawing from the work of [21], this uncertainty changes the loss from Mean Squared Error (MSE) loss to

$$L = \frac{||C_{GT} - C_{Est}||_2^2}{2\beta_\Sigma^2} + \frac{\log\left(\beta_\Sigma^2\right)}{2} \tag{6}$$

where

$$\beta_\Sigma = \sum_{\vec{X}_i \in \chi} P_{S,T}\left(\chi, \vec{X}_i\right) \beta_i \tag{7}$$

where C_{GT} is the ground truth color, C_{Est} is the estimated color, and $P_{S,T}$ is the transient probability that $\vec{X}_i$ is the surface for χ. The model learns to assign transient objects a higher uncertainty than non-transient objects. This results in the model lowering the contribution of the transient objects to training loss.

In addition to incorporating ideas from S-NeRF and NeRF-W, Mari et al. [9] used bundle adjustment to refine the parameters modeling the relationship between pixels and

2. In [2], σ is used for density instead of ρ, however in this paper σ is used to indicate the sigmoid non-linearity function.

3. The symbols P_S and P_V are analogous to w_i, T_i from [2], however as $\vec{w}$ is used for solar angle and T is associated with time in this work, we introduce new notation.

rays. Their model, known as Sat-NeRF, directly utilizes the Rational Polynomial Coefficient (RPC) approximation of the satellite camera model when determining rays associated with pixels. The RPC model is described in Grodecki, and Dial's work [22]. The RPC model for satellites is a popular model used to approximate the physical satellite camera model. Sat-NeRF also uses the 3D points extracted during bundle adjustment to apply the depth supervision technique described in Deng et al. [6]. Like NeRF-W, Sat-NeRF learns embeddings for each image and outputs uncertainty to account for transient objects. However, they forgo the image-specific density and color modifications in NeRF-W.

As an alternative to the method used in [3] and [9] to account for transient objects, we use the loss function described by Barron in [7]. This loss function is defined as

$$L_{\alpha,c}(x) = -\log\left(\frac{1}{cZ(\alpha)}\exp\left(-f(x,\alpha,c)\right)\right), \tag{8}$$

$$f(x,\alpha,c) = \frac{|\alpha-2|}{\alpha}\left(\left(\frac{(x/c)^2}{|\alpha-2|}+1\right)^{\alpha/2}-1\right), \tag{9}$$

and

$$Z(\alpha) = \int_{-\infty}^{\infty}\exp\left(-f(x,\alpha,1)\right)dx. \tag{10}$$

In the above equations, x refers to the difference between the predicted and actual values, and α and c are both learnable parameters. In Equation 9, if $|x| < c$, the function behaves like the MSE loss. For values of $|x| > c$, Equation 9's behavior is determined by α. As α approaches two, the loss behaves like the MSE loss, and as α approaches zero, the loss behaves like the Cauchy loss. Thus the larger the value of α, the larger the contribution to the gradient for errors when $|x| > c$. Theoretically, α can be any real number, and c can be any positive number; however, in practice, α is forced to remain in the range $(0, 2)$ and c remains in the range $(0, 1)$. Note that Equation 8 is the negative log-likelihood of a probability density function created from the normalized form of Equation 9. Using Equation 8 instead of Equation 9 prevents the network from converging to a trivial solution by minimizing α.

3 Method

Season-NeRF, like all variants of NeRF, computes the density and color of every point within the world. Season-NeRF uses the time of the year and the location within the world to determine the color of world points. These input terms allow our approach to account for seasonal variation within the image set. Season-NeRF uses solar angle and location within the model to compute solar visibility. Solar angle alone is used to compute the sky color. The solar visibility and sky color of points along a ray are used to compute a shadow adjustment term for rendered pixels. By using the loss function proposed in Barron [7], we attempt to reduce the influence that transient objects in the training images have on the model.

3.1 Seasonal Variation Adjustment

To account for the seasonal changes, we alter the computation of the albedo color by adding a seasonal adjustment term. We refer to the new term as the seasonally adjusted albedo. The seasonal adjustment term is a function of position within the model and the time of the year (t). The sigmoid non-linearity is not applied to the seasonally adjusted albedo until after the albedo color has been merged with the seasonal adjustment term. Applying the non-linearity after the combination of the albedo color and seasonal adjustment allows the combined term to take on any value before the non-linearity. After the non-linearity has been applied the color will fall in the valid color range of $[0, 1]$. Thus the seasonally adjusted albedo is expressed as

$$A_t\left(\vec{X},t\right) = \sigma\left(A^*\left(\vec{X}\right)+T_a\left(\vec{X},t\right)\right) \tag{11}$$

where A^* is the albedo color before the sigmoid non-linearity, σ, is applied and T_a is the seasonal adjustment term.

To compute the seasonal adjustment term, we use two intermediate terms, which we refer to as temporal class and temporal adjustment, $T_C(t)$ and $T_A\left(\vec{X}\right)$. Temporal class is dependent on the time of the year, and temporal adjustment is dependent on the position within the model. The temporal class is a $N \times 1$ matrix, where N is a hyper-parameter describing the number of different seasonal classes the model can render. We use softmax to ensure the values in T_C form a discrete probability distribution over the seasonal classes. The temporal adjustment is a $C \times N$ matrix, where C is the number of output channels in the rendered image. The seasonal adjustment is

$$T_a\left(\vec{X},t\right) = T_A\left(\vec{X}\right) \times T_C(t). \tag{12}$$

Intuitively, the ith position of $T_C(t)$ represents the probability that time t corresponds to season i, and the columns of $T_A\left(\vec{X}\right)$ represent the seasonal adjustments for each seasonal class. This results in T_a being the expected seasonal adjustment given the probability of each seasonal class T_C. If N is too large (that is, the model can generate too many possible seasons), shadow effects will be absorbed into the seasonal adjustment. If N is too small, the model lacks the descriptive capability to render every season. We let $N = 4$ for our tests, resulting in a good balance between solar and temporal changes.

Our model only allows t to change the value of the seasonally adjusted albedo, leaving density, solar visibility, and sky color constant with respect to t. The limitation that a region's density cannot change over time is an unrealistic assumption, as changes in vegetation can cause the density to change. However, we do not allow our model's density to vary over time for two reasons. First, limiting the computation in this way prevents the model from using t to change the density within the world or shadows within rendered images to account for purely visual changes of objects with fixed densities, such as buildings. Second, each region contains a single ground truth height map, making accurate evaluation of the quality of changing height maps caused by changing densities difficult. The seasonally adjusted rendered color is

$$Col_t(\chi,t) = \sum_{\vec{X}_i\in\chi} A_t\left(\vec{X}_i,t\right)P_S\left(\chi,\vec{X}_i\right) \tag{13}$$

and is used instead of Equation 1.

To compute the seasonally adjusted rendered color, we must input the time of the year into the network. Rather than directly sending the day and month to the network, we encode the time as

$$t = (\cos(2\pi t_p), \sin(2\pi t_p)) \tag{14}$$

where t_p is the fraction of the year completed. Given two times of the year, t_0 and $t_1 = t_0 + \delta$, the L2 distance between the two encodings is $2|\sin(\pi\delta)|$. The L2 distance achieves its maximum value at $\delta = 1/2$, corresponding to the day at the opposite end of the year. This approach also ensures that days at the beginning and end of the year have similar encodings.

3.2 Shadow Adjustment

Like S-NeRF, Season-NeRF outputs $S_{vis}\left(\vec{X}, \vec{w}\right)$ and sky $(\vec{w})$ however we do not apply these terms directly to the seasonally adjusted albedo. Instead, we use these terms to compute a shadow mask, which measures the probability that a rendered pixel is not in shadow. The equation that describes the shadow mask is

$$M(\chi, \vec{w}) = \sigma\left(\kappa\left(\mu + \sum_{\vec{X}_i \in \chi} P_S\left(\chi, \vec{X}_i\right) S_{vis}\left(\vec{X}_i, \vec{w}\right)\right)\right), \tag{15}$$

where $\vec{w}$ is the solar angle, S_{vis} is the solar visibility from the network, and σ is the sigmoid activation function. Note that $M(\chi, \vec{w})$ is the probability that the the surface of χ is visible from the sun. Therefore, $1 - M(\chi, \vec{w})$ is the probability that the surface of χ is in shadow. The hyperparameter κ controls the rapidity of the transition between non-shadow and shadow. The hyperparameter μ controls the threshold where this transition occurs. We let $\kappa = 30$ and $\mu = -.2$, as we found these perform well for shadow generation.

Given a shadow mask and a seasonally adjusted rendered color, the shadow-and-seasonally adjusted rendered color is

$$\begin{aligned} Col_{SA}(\chi, \vec{w}, t) = Col_t(\chi, t) * \\ (M(\chi, \vec{w}) + (1 - M(\chi, \vec{w}))\,\text{sky}(\vec{w})). \end{aligned} \tag{16}$$

Equation 16 assumes that $Col_t(\chi, t)$ is the color of the pixel in direct sunlight and sky $(\vec{w})$ $Col_t(\chi, t)$ is the color of the pixel when the pixel is in shadow. The value of $M(\chi, \vec{w})$ is the probability that the rendered pixel is in shadow. Other sources than direct light from the sun may still illuminate regions in shadow. As such, we allow $||\text{sky}(\vec{w})||_2$ to be greater than zero to ensure regions in shadow are partially lit.

3.3 The Loss Function and Network Architecture

We pass solar and image rays through the network during training. As with S-NeRF, our loss function behaves differently for solar and image rays. The loss function associated with image rays consists of three terms. The first term applies Barron's loss to the absolute difference in the RGB color of the rendered pixel and the ground truth pixel. We use Barron's loss to reduce the effect of transient objects on the training process. The second loss term encourages at least one channel in the seasonally adjusted rendered color to be above a user-defined threshold. By encouraging the model to have at least one large channel in $Col_t(\chi, t)$, we avoid using the seasonally adjusted albedo in place of shadows to describe dark regions. The second loss term is

$$L_A\left(\vec{C}\right) = \frac{1}{n}\sum_{i=1}^{3}\left(1 - \frac{\min\left(\vec{C}_i, \mathbb{A}\right)}{\mathbb{A}}\right)^2 \tag{17}$$

where $\mathbb{A}$ is a hyperparameter such that $\mathbb{A} \in (0, 1]$. We use $\mathbb{A} = 0.2$ as most non-shadow regions of our images have at least one channel above this value. Even with this loss term, our network tended to ignore shadows by setting the sky color to one. To avoid this, we added a third loss function

$$L_{sky}(\text{sky}) = \sum_{i=1}^{3}\begin{cases} 0 & \text{sky}_i \le \mathbb{S} \quad (18a) \\ \left(\frac{1}{\mathbb{S}} * \text{sky}_i - 1\right)^2 & \text{sky}_i > \mathbb{S}, \quad (18b) \end{cases}$$

where $\mathbb{S}$ is a hyperparmeter such that $\mathbb{S} \in (0, 1]$. We use $\mathbb{S} = 0.5$ because this will encourage areas in shadow to reduce the seasonally adjusted albedo color by at least 50% while still allowing some indirect light to illuminate the region. Given an image ray $\left(\chi, \vec{w}, t, \vec{C}_{GT}\right)$, the total loss associated with that ray is

$$\begin{aligned} L_{\text{IR}} = & L_{\alpha,c}\left(\vec{C}_{GT} - Col_{SA}(\chi, \vec{w}, t)\right) \\ & + L_A(Col_t(\chi, t)) + L_{sky}(\text{sky}(\vec{w})). \end{aligned} \tag{19}$$

We shall now describe the training loss associated with solar rays. The loss function associated with solar rays consists of two terms. The first term minimizes the MSE between the estimated solar visibility, $S_{vis}(\vec{w}, \chi)$ and the exact solar visibility computed by Equation 4. The second term is the loss described in Equation 18. Applying Equation 18 to solar rays ensures the sky color is appropriate, even if the input solar angle is far from the solar angles from the training set. The total loss for a solar ray is

$$\begin{aligned} L_{\text{SR}} = & L_{sky}(\text{sky}(\vec{w})) + \\ & \frac{1}{K}\sum_{j=1}^{K}\left(S_{vis}\left(\vec{X}_j, \vec{w}\right) - P_V\left(\vec{X}_j, \chi\right)\right)^2. \end{aligned} \tag{20}$$

Unlike S-NeRF, we do not use an absorption term for solar ray loss. The accuracy of Season-NeRF's shadow mask is limited by the accuracy of its density computation because the training process uses the density output of the network to determine the solar visibility.

We shall now describe the architecture of Season-NeRF. Season-NeRF consists of SIREN fully connected layers. Some of these layers include batch normalization, despite batch normalization being non-standard with NeRFs and SIRENs, as we found this gave improved results, as shown in Table 1[4]. Table 1 uses the Structural Similarity Index Measure (SSIM) to measure image quality and Mean Absolute Error (MAE) to measure the quality of the height map. Furthermore, the inputs of our network undergo positional

4. Our tables employ the symbol ↑ to represent metrics where larger numbers signify superior results and ↓ for metrics where smaller numbers signify superior results. In addition, we use ● to indicate the result with the best score and ● to indicate the worst score.

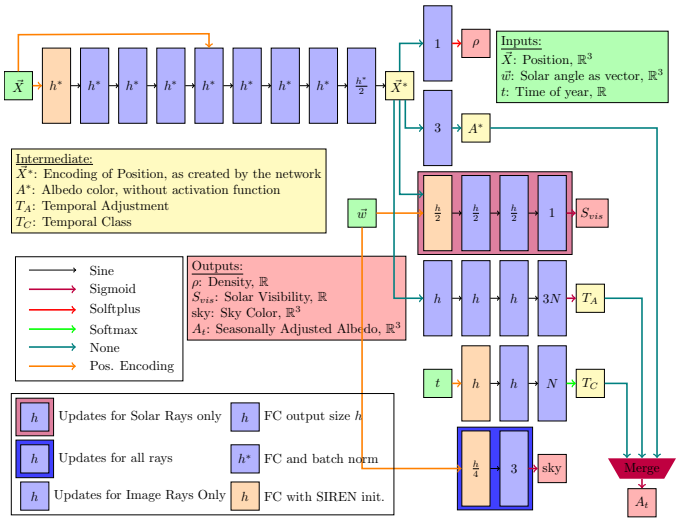

Fig. 2. Season-NeRF's network architecture where, $\vec{X}$ is the input's spatial coordinates, $\vec{w}$ is the solar angle, and t is the time of input. The model predicts density, seasonally adjusted albedo, solar visibility, and sky color. The Merge trapezoid indicates the effect of Equation 11 and does not contain any learnable parameters.

encoding, as described in [2], before being processed. We provide a network diagram of Season-NeRF in Fig. 2. We set the layer width of the network to 512. During training, we use 512 image rays and 1024 solar rays per batch, with all rays sampled at 96 points. When backpropagating the loss for image rays, we freeze weights within the model used only to compute solar visibility. Also, when backpropagating the loss for solar rays, we freeze weights within the model that are not used exclusively for the computation of sky color or solar visibility. We use the One Cycle Learning Rate Scheduler described by Smith and Topin [23] during training with a maximum learning rate of $1.5\text{e}{-4}$. We train our network for 50000 steps, which takes approximately 7 hours on an Nvidia GeForce GTX 1080 Ti GPU.

3.4 Multi-Phase Training

We train Season-NeRF in two phases. The first phase accounts for 20% of the training and uses a height map to jump-start the training process. As noted in Deng et al. [6], we can use the height information to speed up the training and accuracy of a neural radiance field. We acquire a height map via space carving [24] and use it to modify the computed density during the early stages of training. The height map used for depth supervision are created from the same images used to train the NeRF. The images used for evaluating the NeRF and the ground truth height map *are not used* during the construction of the height map used for training. In the first phase of training, we replace ρ with ρ_m, which is

$$\rho_m\left(\vec{X}\right) = \Gamma\rho\left(\vec{X}\right) + (1-\Gamma)\,\rho_H\left(\vec{X}, \delta_{\vec{X}}, H\right), \tag{21}$$

where Γ linearly increases from 0 to 1 during the first phase, and ρ_H is the density required for the neural radiance field to match the height map.

The value of ρ_H depends on the height map H, the location within the model $\vec{X}$, and the distance between $\vec{X}$ and the next point along the ray, $\delta_{\vec{X}}$. It is

$$\rho_H\left(\vec{X}, \delta_{\vec{X}}, H\right) = \begin{cases} \dfrac{10}{\delta_{\vec{X}}} & \vec{X} \text{ is at or below H} \quad (22a) \\ 0 & \vec{X} \text{ otherwise.} \quad (22b) \end{cases}$$

From Equation 3, it is evident that ρ_H encourages each point at or below a surface to have a $P_E \approx 1.0$. In addition to ρ_m, we add a prior approximation loss

$$L_p\left(\vec{X}\right) = L_{\alpha,c}\left(e^{-\rho\left(\vec{X}\right)\delta_{\vec{X}}} - e^{-\rho_h\left(\vec{X},\delta_{\vec{X}},H\right)\delta_{\vec{X}}}\right). \tag{23}$$

Equation 23 ensures that the ρ will be updated in the early stages of training despite having minimal impact on the rendered color when Γ is near zero.

The second phase accounts for the remaining 80% of training and ceases using ρ_m. Furthermore, we no longer use the prior loss defined in Equation 23 as the errors in our prior height map cause ρ_H to be less accurate than ρ. In addition, height maps cannot represent overhangs that may be present in the scene.

3.5 Hyperparamter Tuning

The method we employ for hyperparameter tuning involves two parts. In the first part, we manually adjust hyperparameters to create decent results. In the second part, we employ Bayesian optimization as described in [25] to refine the hyperparameters. This process was run on OMA 248, which is not included in the results sections. We excluded OMA 248 as the hyperparameter tuning process involves feedback from the ground truth lidar, and thus all of the data associated with OMA 248 becomes training data.

To evaluate the performance of a set of hyperparameters, we must combine the results of multiple metrics, measuring different qualities. To this end, we combine the SSIM, MAE, and seasonal stability into a single score, which is maximized during Bayesian optimization. This score is

$$Score = \frac{SSIM}{SSIM_B} - \frac{MAE}{MAE_B} + \begin{cases} 1 & EM < EM_B \\ 0 & else \end{cases} \tag{24}$$

where $SSIM_B$, MAE_B, and EM_B are the scores corresponding to the manually tuned parameters. In Equation 24, EM refers to the worst earth-movers distance found after applying the seasonal stability process described in Section 4.3. Creating and evaluating Season-NeRF for a set of hyperparameters takes approximately eight hours. To speed up this process, we reduce the number of training steps from 50000 to 5000 and down-sample the images to keep the number of epochs consistent with the number of epochs when using 50000 steps. As a result, the training and evaluation for a set of hyperparameters is reduced to about 30 minutes.

4 Tests and Results

We analyze the performance of Season-NeRF on areas of interest from the *2019 IEEE GRSS Data Fusion Contest* [1], which contains images captured by Maxar WorldView-3 between 2014 and 2016. WorldView-3 captures 8-band visible and near-infrared images; however, [1] provides provides pan-sharpened RGB images. Images used in Season-NeRF

TABLE 1
Comparison of results with and without batch normalization.

Region	SSIM, SA ↑		MAE ↓	
Case:	With Batch Norm	No Batch Norm	With Batch Norm	No Batch Norm
OMA 042	0.562 ●	0.562 ●	2.357 ●	2.496 ●
OMA 084	0.529 ●	0.514 ●	3.769 ●	4.684 ●
OMA 132	0.704 ●	0.709 ●	1.279 ●	1.524 ●
OMA 163	0.527 ●	0.518 ●	2.073 ●	2.311 ●
OMA 203	0.696 ●	0.691 ●	1.429 ●	2.586 ●
OMA 212	0.679 ●	0.670 ●	1.102 ●	1.212 ●
OMA 281	0.618 ●	0.617 ●	2.296 ●	3.476 ●
OMA 374	0.473 ●	0.474 ●	4.211 ●	5.095 ●
Average	0.599 ●	0.594 ●	2.314 ●	2.923 ●

undergo top-of-atmosphere correction to remove most of the variation caused by daily changes in atmospheric conditions. The images are of the city of Omaha, Nebraska, USA and are RGB images of size 2048 by 2048 pixels covering an area of approximately 580 by 580 meters. However, we downsample the images to be 512 pixels by 512 pixels for training. Table 2 contains a summary of each region.

TABLE 2
Overview of data. For each area, four images are reserved for testing.

Area Index	Num. Imgs.	Height Range (m)
OMA 042	39	38.3
OMA 084	26	61.1
OMA 132	41	31.9
OMA 163	40	40.1
OMA 203	43	51.7
OMA 212	38	34.6
OMA 281	41	67.3
OMA 374	30	59.4

Given a set of images, we withhold four for testing purposes and use the remaining for training. Three of these images represent prototypical seasons, and the fourth is selected to ensure a wide range of viewing and solar angles. The prototypical seasonal images are selected to ensure each testing set contains an image with large amounts of snow, green foliage, and brown foliage. The prototypical images are shown in Fig. 3 with an example distribution of the training data shown in Fig. 4. We use bundle-adjusted RPCs acquired by the method described in [26] with initial height maps acquired by the space carving process propounded in [24].

4.1 Novel View and Seasonal Variability Tests

To evaluate the quality of our model's novel view renderings, we use the Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index Measure (SSIM) [27] and compare generated images with images withheld from the training process. In addition to applying these metrics to the output using the exact times provided in the satellite image's metadata, we also perform seasonal alignment. These results are shown in Table 3[5].

Seasons can change rapidly, and weather events sometimes occur unusually early or late in a season. These sudden seasonal changes can cause the network to render an image containing seasonal characteristics that are valid for the time of the year but do not match the exact characteristics shown in the ground truth. Seasonal alignment is accomplished by finding the time of year and the sky color such that the MSE between the rendered image and an image containing the desired seasonal features is minimized. Examples of rendered images are shown in Fig. 3. OMA 084, OMA 212, and OMA 281 are times when seasonal alignment is very beneficial.

Examining the rendered images and image similarity scores provides evidence that Season-NeRF can construct images at novel view angles. Selecting a wide range of seasons for the testing images allows us to check that seasonally independent features persist in the rendered images. Objects, like buildings and roads, have their appearance modified by seasonal events throughout the year but are still visible and identifiable. Additional evidence of this capability is provided in Sec. 4.3.

We consider five cases to examine the effectiveness of Season-NeRF's components. These cases are the entire implementation of Season-NeRF (Case A), Season-NeRF using S-NeRF's form of shadow prediction (Case B), Season-NeRF replacing Barron's loss with MSE loss (Case C), Season-NeRF without using a height map to guide the early stages of training (Case D), and Season-NeRF with only a single temporal class available (Case E). An example of how the inclusion of multiple temporal classes alters the output is shown in Fig. 5. From visual similarity scores shown in Table 3, we conclude that allowing a model to account for multiple seasons drastically improves the quality of the rendered images.

4.2 Height Map and Shadow Mask Tests

To evaluate the height map, we compute the mean absolute height error (MAE), as described in [28], relative to lidar. Examples of the height maps rendered by Season-NeRF

5. For tests involving more than two cases, we include multicolored indicators. The amount of green shown in the indicator is proportional to the distance to the best result for the metric.

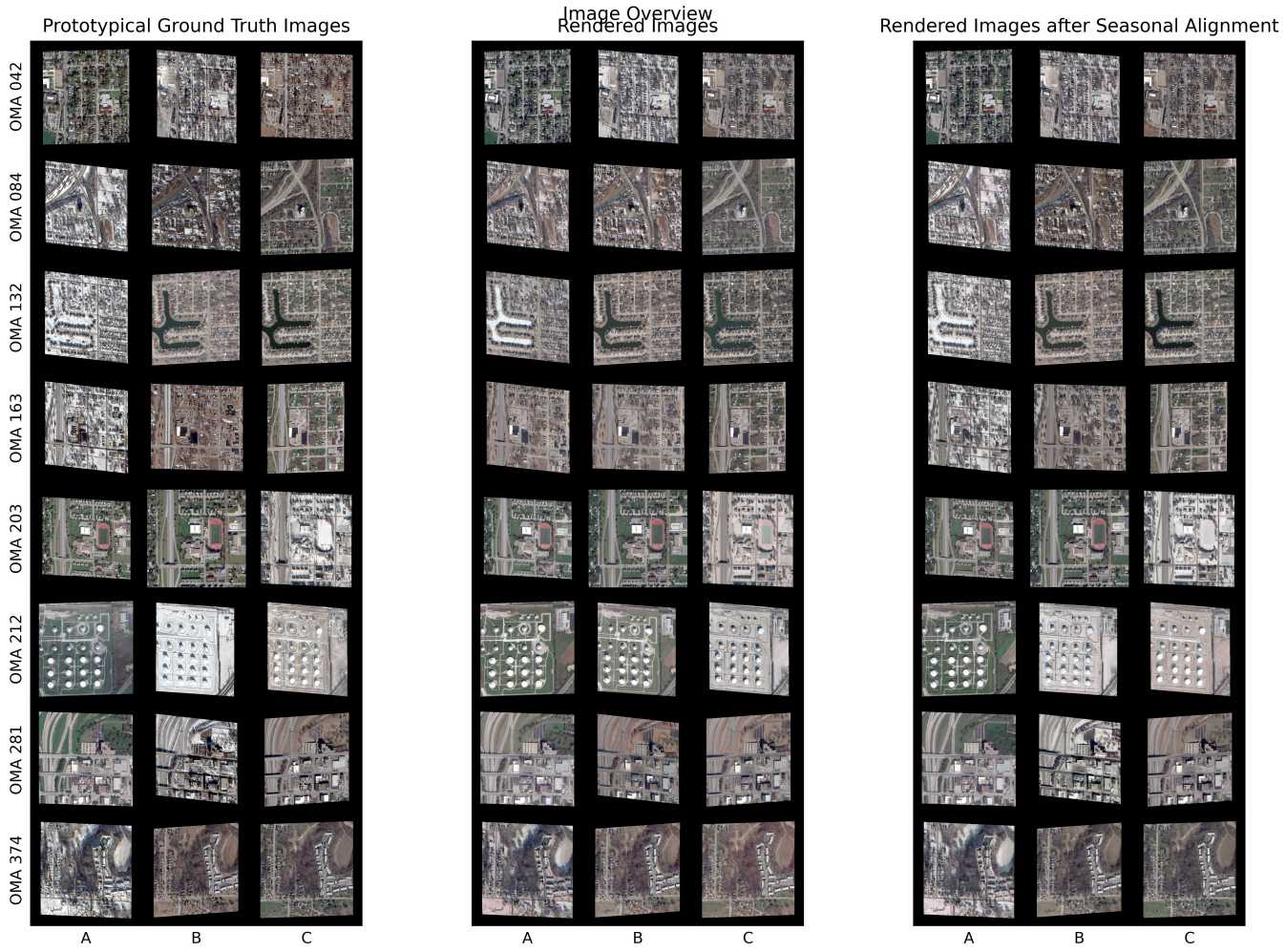

Fig. 3. The left hyper-column contains ground truth images, the center hyper-column contains rendered images with direct time input, and the right hyper-column contains rendered images after seasonal alignment.

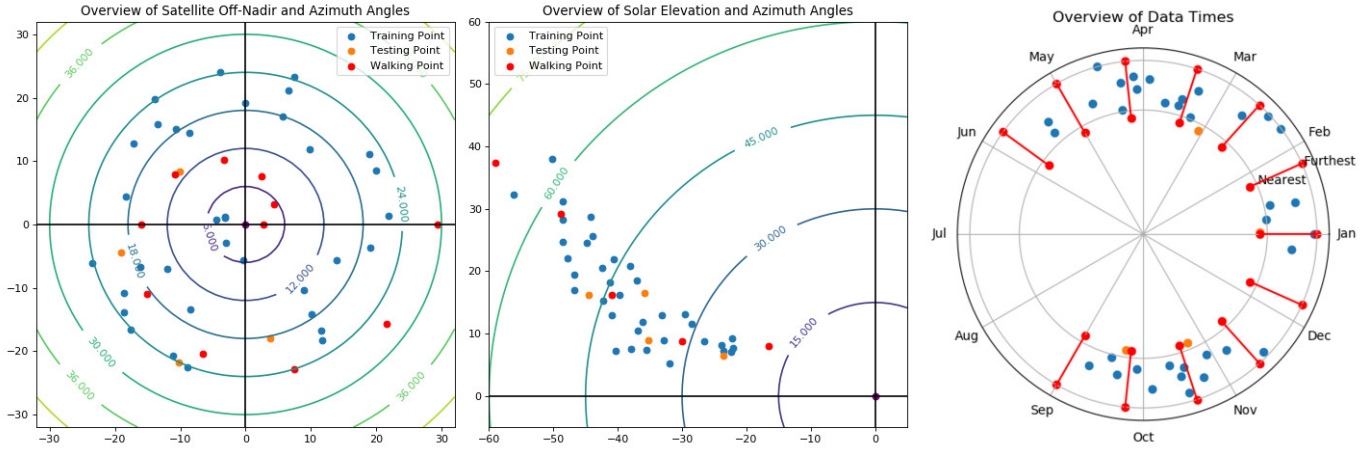

Fig. 4. Distribution of viewing angles, solar angles, and times for OMA 042. The satellite and solar angle graphs are on a polar scale, with distance from the origin measuring off-nadir and elevation angles. The rotation from the positive x-axis measures the azimuth angle. The time plot uses rotation to reflect the time of the year. Distance from the origin indicates the sum of the solar and viewing angle difference between the training point and the closest testing point in terms of days apart. The smallest summed rotational difference is 23.4 degrees, and the largest is 187.0 degrees. The walking points occur at multiple times and viewing angles and are represented by a line, indicating that views near and far from the training points are used.

process are shown in Fig. 6. Table 4 contains a quantitative comparison of our height maps with lidar data. We considered the same cases as in Section 4.1 for the evaluation of the height map. As with image quality, the inclusion of

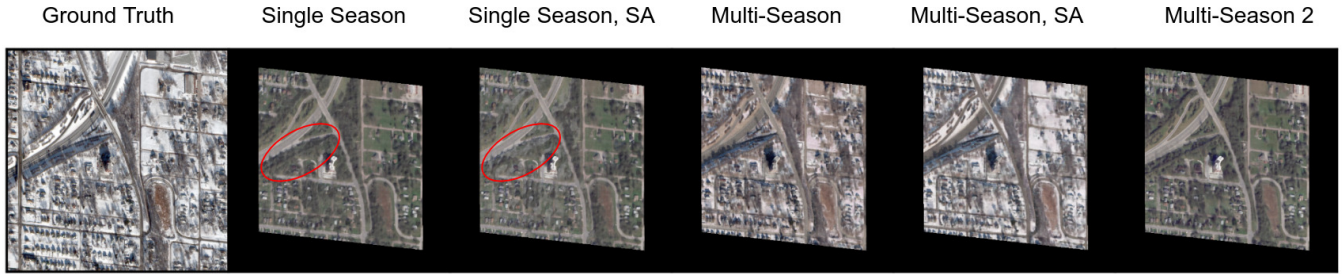

Fig. 5. This image is an example of how a single season limits the ability of NeRF to render novel views. Despite generating a high-quality rendering with accurate buildings and roads, the seasonal features in the single-season output are incorrect. Attempting seasonal alignment allows the sky color to change and the model to turn part of a forest into shadow (area circled in red) but does not allow the season to change due to the limitations of how shadows can affect the rendering. However, the multi-season approach generates an image with the correct buildings, roads, and seasonal features for a winter scene. Seasonal alignment improves these results by allowing the network to generate a winter scene with closer seasonal features to the ground truth. Also included on the far right is an image generated with seasonal features similar to the single-season model as evidence that the multi-season approach can generate multiple seasons with a single model.

TABLE 3
Results of comparing rendered images with testing images. Included are directly rendered images and scores after seasonal alignment (SA). Case A: Full Model. Case B: Full Model, S-NeRF Solar loss. Case C: MSE Loss. Case D: No Height Map. Case E: No Seasonal Adjustment.

Region	PSNR ↑					SSIM ↑				
Cases:	A	B	C	D	E	A	B	C	D	E
OMA 042	19.35	19.43	19.30	19.22	17.45	0.60	0.60	0.60	0.59	0.48
OMA 084	19.17	18.46	19.00	19.12	16.85	0.55	0.52	0.54	0.53	0.35
OMA 132	19.82	19.84	20.12	20.12	17.59	0.62	0.61	0.64	0.63	0.51
OMA 163	18.65	18.78	19.36	18.61	17.60	0.52	0.54	0.56	0.52	0.46
OMA 203	21.02	21.21	21.34	20.39	18.80	0.65	0.65	0.65	0.63	0.55
OMA 212	17.79	18.12	18.46	17.65	15.93	0.57	0.59	0.59	0.56	0.51
OMA 281	19.74	19.96	20.36	19.91	19.08	0.60	0.61	0.61	0.60	0.55
OMA 374	20.48	20.33	20.55	20.20	18.91	0.52	0.53	0.54	0.52	0.45
Average	19.50	19.51	19.81	19.40	17.78	0.58	0.58	0.59	0.57	0.48

Region	PSNR, SA ↑					SSIM, SA ↑				
Cases:	A	B	C	D	E	A	B	C	D	E
OMA 042	19.73	19.85	19.83	19.65	17.46	0.60	0.60	0.60	0.60	0.48
OMA 084	20.32	20.25	20.46	20.04	16.96	0.57	0.56	0.57	0.54	0.35
OMA 132	21.21	21.45	21.27	21.18	17.60	0.67	0.67	0.67	0.67	0.51
OMA 163	19.82	19.76	19.80	19.59	17.61	0.59	0.57	0.58	0.56	0.46
OMA 203	21.44	21.67	21.63	21.23	18.83	0.66	0.66	0.65	0.64	0.55
OMA 212	20.26	21.09	20.47	20.32	16.20	0.61	0.64	0.63	0.61	0.51
OMA 281	20.99	21.19	21.16	21.03	19.15	0.64	0.64	0.63	0.62	0.55
OMA 374	21.28	21.44	21.43	20.91	18.95	0.53	0.54	0.54	0.53	0.45
Average	20.63	20.84	20.75	20.49	17.85	0.61	0.61	0.61	0.60	0.48

seasonal variation resulted in a substantial improvement to the height map. In addition, the inclusion of a prior is a significant factor in the quality of the final height map generated by Season-NeRF.

To measure the quality of our predicted shadow masks, we compute an exact shadow mask using Equation 15 by replacing $V\left(\vec{X}_i, \vec{w}\right)$ with the exact solar visibility computed along the ray by Equation 4. In theory, we could always use the exact value to compute the shadow mask instead of using a network to approximate it. However, this is not practical as using the exact computation to render a ray with n points is $O\left(n^2\right)$. Instead, if the approximate computation is used, rendering is a $O\left(n\right)$ operation. Examples of the predicted shadow mask and the exact shadow mask are shown in Fig. 10 with numerical results shown in Table 5. In addition Fig. 7 shows an example of the shadow mask from Equation 15 before applying the sigmoid function. In Fig. 9, we show an image before and after the shadow mask has been applied. We compare the estimated shadow mask to the exact shadow mask instead of the shadow mask from the lidar DSM as shadows are predicted entirely based on the model density. Errors in the model density will result in errors in the shadows and create ambiguity regarding the performance of the estimated shadow mask, as it is unclear if a loss in accuracy is due to poor approximation of the exact shadows or problems in the density model.

As with [6] and [9], the inclusion of prior height information drastically improves the quality of the model

generated by Season-NeRF. However, we also compare the quality of the model learned with the prior height map with the quality of the prior height map itself. We note that in most cases, Season-NeRF results in a height map superior to the one provided during training. As shown in Fig. 8, Season-NeRF initially learns height information from the prior height map, including some of the errors in the prior height map. However, in the later phase of the training, when the prior height map is no longer being used, the training process can correct the errors in the prior. Correcting these errors results in a superior height map from the NeRF compared to the prior provided to the NeRF for training.

4.3 Seasonal Specificity and Stability Tests

Seasonal specificity refers to the capability of Season-NeRF to render seasonal features that are expected based on provided time of the year. Seasonal stability refers to these features' invariance when changing viewing and solar angles. The ideal method for measuring a model's ability to be specific and stable is to generate images across all reasonable viewing angles, solar angles, and time combinations. These images could then be compared to ground truth images to show that the quality does not vary and the rendered images have the desired seasonal properties and shadows while still accurately capturing properties invariant to seasons and shadows. While it is possible to render images for all different input configurations, we are extremely limited in the available ground truth data. As such, we must determine an alternative method to test the specificity and stability of Season-NeRF approach.

To determine that Season-NeRF can render a specific season, we render 180 images, each approximately 6 days apart, spanning Spring (Fig. 11), Summer (Fig. 12), Fall (Fig. 13), and Winter (Fig. 14). We can visually confirm the ability of Season-NeRF to render images with the expected seasonal features and correct seasonally independent properties. The rendered images contain expected seasonal features for Spring, Fall, and Winter. In Winter (Fig. 14), snow melts and returns in several regions, and in late fall (last columns of Fig. 13), some images have snow, and others do not, as some regions had an early snowfall and others did not. In Spring (Fig. 11), we can see brown foliage becoming green; however, in the last column, certain regions seem to revert to winter. This pattern continues in Summer (Fig. 12), where we see snow appear. However, we attribute this strange behavior to the lack of data during the summer months, which explains why our model could not correctly predict the appearance of these seasons. Despite the lack of data from summer, giving the model a time of the year during summer does not result in a distorted image. Instead, it results in a valid image with incorrect seasonal features but correct seasonally independent features. Thus, Season-NeRF can accurately specify the season, assuming training data is available during the desired season.

To measure the stability of the images generated by Season-NeRF, we measure the Earth Mover's Distance (EMD) [29] between images rendered at the same time of the year but with different viewing and solar angles. Shifts in the location of structures within the image caused by changing the viewing angles should have minimal impact on the image's histogram. Therefore, shifts in viewing angles should not significantly influence the EMD since EMD is a histogram-based image similarity metric, and we are not using a variant of EMD that considers pixel position. As seasonal shifts generally affect the entire image, these changes should provide the bulk of the distance measured by EMD. However, viewing and solar angle changes would

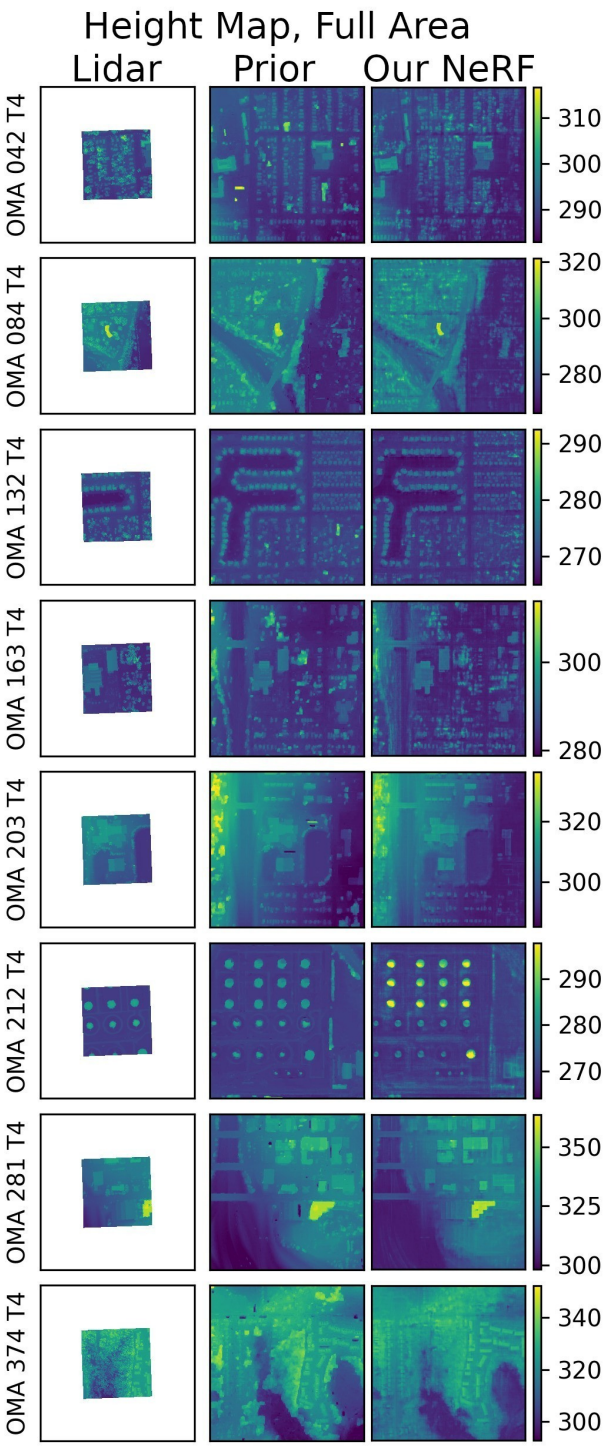

Fig. 6. Height maps of lidar data, where available (left), prior height map (center), and Season-NeRF's height map (right).

TABLE 4
Quality of height maps relative to lidar data. Highlighted in green is the best score per metric for each region. Case Prior: Space Carved Height Map. Case A: Full Model. Case B: Full Model, S-NeRF Solar loss. Case C: MSE Loss. Case D: No Height Map. Case E: No Seasonal Adjustment.

Reg	MAE ↓						RMSE ↓					
	Prior	A	B	C	D	E	Prior	A	B	C	D	E
042	2.92	2.47	2.37	2.41	2.43	2.79	4.08	3.56	3.44	3.47	3.42	3.79
084	3.34	2.82	3.11	2.90	3.94	6.22	5.06	4.13	4.60	4.39	5.14	7.95
132	1.20	1.22	1.26	1.23	1.46	1.67	2.13	2.02	2.10	2.07	2.06	2.44
163	1.80	1.43	1.68	1.88	2.04	2.42	2.75	2.31	2.69	2.61	2.77	3.10
203	1.37	1.17	1.44	1.07	2.02	1.87	2.61	1.92	2.45	1.81	2.82	2.82
212	0.64	0.94	0.84	0.91	1.11	1.13	1.44	1.76	1.59	1.69	1.99	1.96
281	1.77	1.46	1.51	1.61	2.79	2.65	3.32	2.46	2.52	2.67	3.86	3.66
374	5.27	4.44	4.29	4.52	5.22	5.91	6.75	5.88	5.77	5.90	6.49	7.32
Avg	2.29	1.99	2.06	2.07	2.62	3.08	3.52	3.01	3.14	3.08	3.57	4.13

Reg	Percent within 1 m ↑						Median Error ↓					
	Prior	A	B	C	D	E	Prior	A	B	C	D	E
042	0.17	0.26	0.30	0.31	0.31	0.25	2.22	1.61	1.55	1.63	1.71	2.16
084	0.34	0.30	0.28	0.32	0.16	0.13	2.27	1.82	2.06	1.68	3.14	4.90
132	0.67	0.61	0.60	0.63	0.47	0.47	0.63	0.73	0.75	0.69	1.08	1.10
163	0.41	0.57	0.48	0.37	0.35	0.28	1.17	0.84	1.04	1.41	1.55	2.07
203	0.56	0.65	0.56	0.68	0.38	0.46	0.86	0.64	0.89	0.61	1.44	1.12
212	0.85	0.73	0.76	0.76	0.67	0.65	0.25	0.45	0.41	0.49	0.68	0.70
281	0.54	0.54	0.54	0.50	0.27	0.26	0.90	0.89	0.91	0.99	2.02	2.04
374	0.12	0.18	0.21	0.14	0.13	0.09	4.21	3.23	3.11	3.31	4.39	5.06
Avg	0.46	0.48	0.47	0.46	0.34	0.32	1.56	1.28	1.34	1.35	2.00	2.39

TABLE 5
Quantitative overview of EM distance when time is constant. Case A: Full Model. Case B: Full Model, S-NeRF Solar loss. Case C: MSE Loss. Case D: No Height Map. Case E: No Seasonal Adjustment. While Case D contains the best shadow prediction, it also performs poorly for height map quality.

Region	Accuracy ↑					Sun F1 ↑				
Cases:	A	B	C	D	E	A	B	C	D	E
OMA 042	0.94	0.94	0.93	0.97	0.91	0.97	0.97	0.96	0.98	0.95
OMA 084	0.94	0.95	0.94	0.98	0.94	0.96	0.98	0.95	1.00	0.91
OMA 132	0.94	0.94	0.94	0.98	0.91	0.97	0.97	0.96	0.99	0.97
OMA 163	0.94	0.95	0.95	0.98	0.92	0.97	0.98	0.98	0.99	0.97
OMA 203	0.95	0.93	0.96	0.98	0.91	0.96	0.96	0.96	0.99	0.95
OMA 212	0.97	0.98	0.97	0.99	0.93	0.98	0.99	0.99	0.99	0.97
OMA 281	0.97	0.96	0.98	1.00	0.93	0.97	0.96	0.97	0.99	0.94
OMA 374	0.95	0.92	0.95	0.98	0.92	0.96	0.95	0.97	1.00	0.94
Average	0.95	0.95	0.95	0.98	0.92	0.97	0.97	0.97	0.99	0.95

Region	Shadow Precision ↑					Shadow Recall ↑				
Cases:	A	B	C	D	E	A	B	C	D	E
OMA 042	0.77	0.84	0.78	0.80	0.65	0.47	0.37	0.32	0.49	0.42
OMA 084	0.77	0.70	0.74	0.91	0.61	0.53	0.41	0.49	0.85	0.71
OMA 132	0.82	0.82	0.76	0.87	0.71	0.48	0.33	0.41	0.56	0.28
OMA 163	0.79	0.88	0.81	0.89	0.62	0.47	0.31	0.31	0.58	0.41
OMA 203	0.76	0.66	0.72	0.81	0.64	0.54	0.38	0.54	0.63	0.44
OMA 212	0.80	0.91	0.90	0.89	0.65	0.68	0.64	0.49	0.73	0.49
OMA 281	0.78	0.66	0.78	0.87	0.60	0.66	0.60	0.63	0.76	0.62
OMA 374	0.67	0.52	0.75	0.75	0.58	0.60	0.40	0.44	0.69	0.56
Average	0.77	0.75	0.78	0.85	0.63	0.55	0.43	0.45	0.66	0.49

still result in a non-zero EMD.

To estimate the expected EMD between two seasonally different images, we consider the EMD between the prototypical testing images. The EMD between prototypical testing images provides a value for the expected EMD between images with different season features. The thresh-

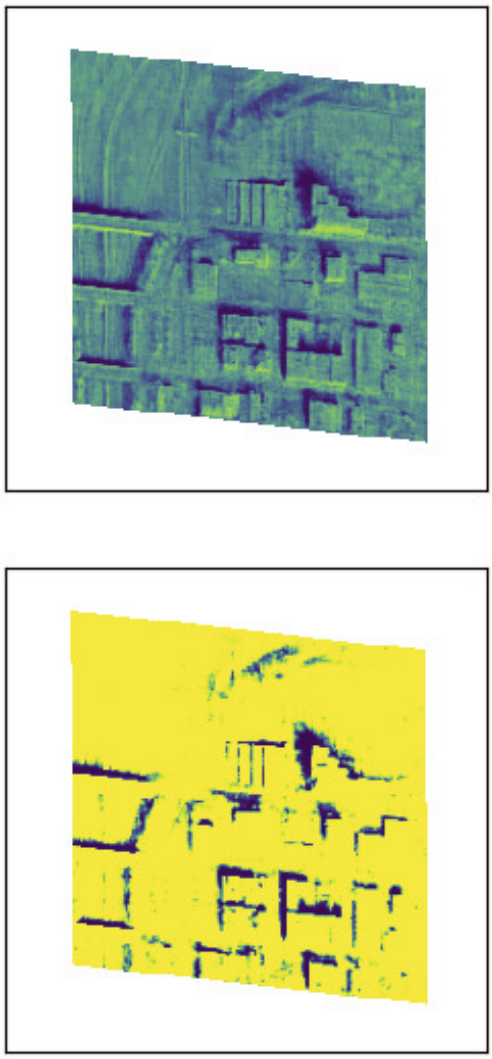

Fig. 7. Example of application of Equation 15 (bottom), as well as the image before sigmoid, scaling via κ, and shifting via μ are used (top). Both images are in the range of $[0, 1]$.

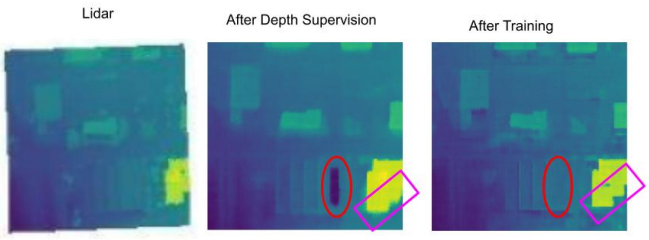

Fig. 8. Lidar data of OMA 281 (left). Intermediate height map generated by Season-NeRF at the end of depth supervision (center). Final height map generated by Season-NeRF (right). Note the gauge (circled in red) from the depth supervision and the fuzzy building border (in pink rectangle). These errors in the height map are from the prior, and the training process corrects them after depth supervision stops.

olds for seasonal changes between prototypical images are summarized in Table 6, and Table 7 provides an overview of the EMD across 660 different combinations of view angle, solar angle, and time. Each test case is computed at one of five solar angles, eleven viewing angles, and twelve viewing

Fig. 9. Example of shadow rendering on OMA 374. Results of rendered images without shadow mask (left) and with shadow mask (right). Regions circled in red contain shadows that are not rendered until after applying the shadow mask.

times. These points are shown in Fig. 4. We are primarily interested in Cases A and B, given that Cases C, D, and E have inferior performance in other areas. However, we include all cases in Table 7 for completeness.

We compare the results of our method for rendering shadows with those of S-NeRF's method of rendering shadows in Figs. 16 and 15. In Fig. 16, we show the image pairs from each region with the largest EMD when the time of the year is constant. We provide a histogram of the EMD of all the image pairs we tested in Fig. 15. Based on the histogram in Fig. 15 and rendered images in Fig. 16, we conclude our shadow rendering process results in a seasonally stable network, where solar and view angle do not alter seasonal features. Using the solar approach from S-NeRF does not ensure that the seasonal features are independent of the solar angle. We conclude this because a significant portion of the image pairs rendered with S-NeRF's shadow method had an EMD above the threshold set by the prototypical images, indicating a change in the seasonal features.

TABLE 6
Baseline EMD for seasonal effects

Baseline Scores	Min	Median	Max
OMA 042	13.23	16.30	19.18
OMA 084	8.76	23.07	23.25
OMA 132	9.18	18.83	23.95
OMA 163	9.65	16.64	19.76
OMA 203	6.01	23.43	24.47
OMA 212	17.88	25.50	33.95
OMA 281	11.19	11.39	12.05
OMA 374	5.48	17.83	18.01
Average	10.17	19.12	21.83

4.4 Analysis of Barron's Loss Compared to MSE Loss

A comparison of visual quality scores in Table 3 and height map quality in Table 4 models trained using Barron's loss perform almost identically to those trained with MSE loss. However, the models have different performances in shadow prediction, with Barron's loss improving shadow recall by an average of 13%. The change in performance is unexpected, as shadow prediction never uses Barron's loss.

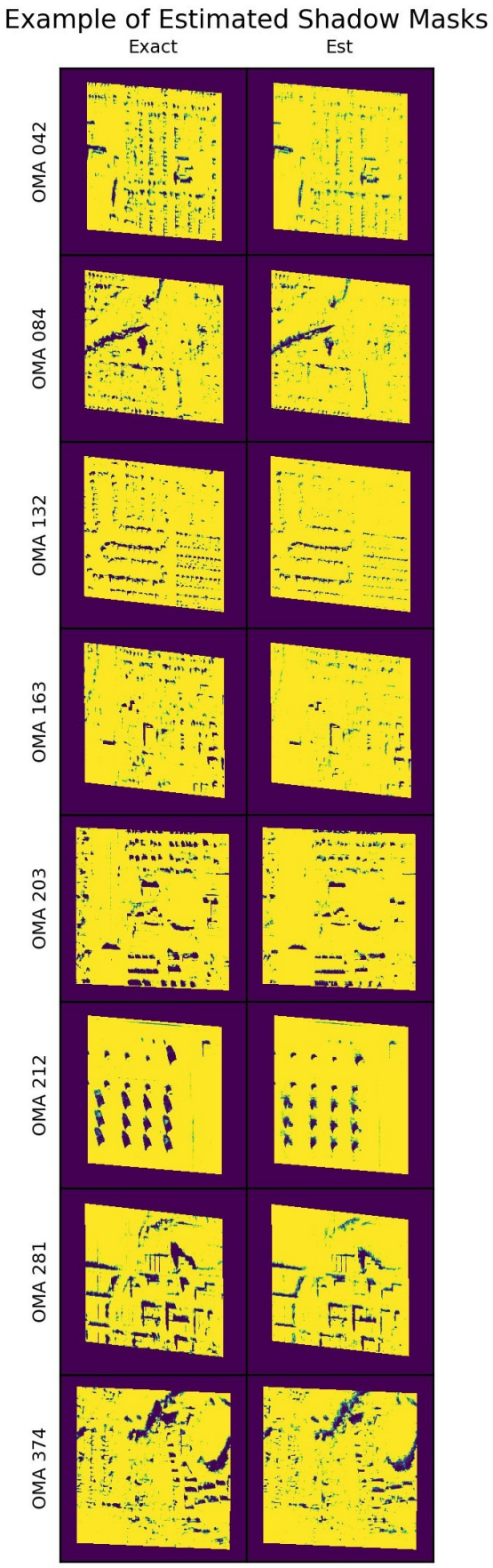

Fig. 10. Examples of estimated shadow masks (right) compared to exact shadow masks (left)

TABLE 7
Quantitative overview of EMD when time is constant. Case A: Full Model. Case B: Full Model, S-NeRF Solar loss. Case C: MSE Loss. Case D: No Height Map. Case E: No Seasonal Adjustment.

Reg	Median ↓				
	A	B	C	D	E
042	1.60	3.21	1.81	1.54	2.28
084	2.24	12.16	1.78	1.44	3.01
132	1.49	7.40	1.76	1.35	1.76
163	1.76	8.28	1.47	1.43	2.20
203	2.46	6.28	2.35	1.86	2.52
212	1.43	1.51	1.33	1.52	1.52
281	2.42	6.86	2.56	2.24	4.94
374	2.45	4.59	1.77	1.25	2.18
Avg	1.98	6.29	1.85	1.58	2.55

Reg	95% Quantile ↓				
	A	B	C	D	E
042	3.57	8.55	3.99	3.08	4.72
084	5.39	24.37	3.85	2.66	5.41
132	3.07	15.17	3.91	2.46	3.32
163	3.38	17.31	3.02	2.89	4.16
203	4.82	13.29	5.17	3.46	4.91
212	2.54	8.91	2.43	2.72	2.89
281	6.03	9.69	7.14	5.25	8.14
374	7.69	32.26	4.77	2.17	7.64
Avg	4.56	16.19	4.29	3.09	5.15

Reg	Max ↓				
	A	B	C	D	E
042	5.74	12.29	6.53	5.31	6.49
084	8.79	27.88	6.68	3.33	7.52
132	4.75	19.31	6.41	4.04	4.27
163	5.40	20.74	4.59	4.02	5.50
203	7.24	16.64	7.82	4.45	6.57
212	3.94	10.98	4.44	3.80	5.41
281	9.25	12.55	10.34	8.71	10.24
374	10.21	36.49	6.83	7.10	9.04
Avg	6.92	19.61	6.71	5.10	6.88

Determining an explanation for this phenomenon is an area for future work.

4.5 High Resolution Results

We also consider how increasing the resolution of the images used to create Season-NeRF affects the performance. To accomplish this, we do not down-sample the images; however, we must reduce the size of the regions we consider from 500 by 500 meters to 250 by 250 meters. We consider using parameters tuned for large area low-resolution images and parameters tuned for small area high resolution. The results are shown in Table 8, with example renderings in Fig. 17. Unsurprisingly, tuning the parameters results in superior performance. We show the changed parameters in Table 9.

5 Conclusion

Given multidate and multiview satellite images of a scene, we can render novel views with specific solar and seasonal

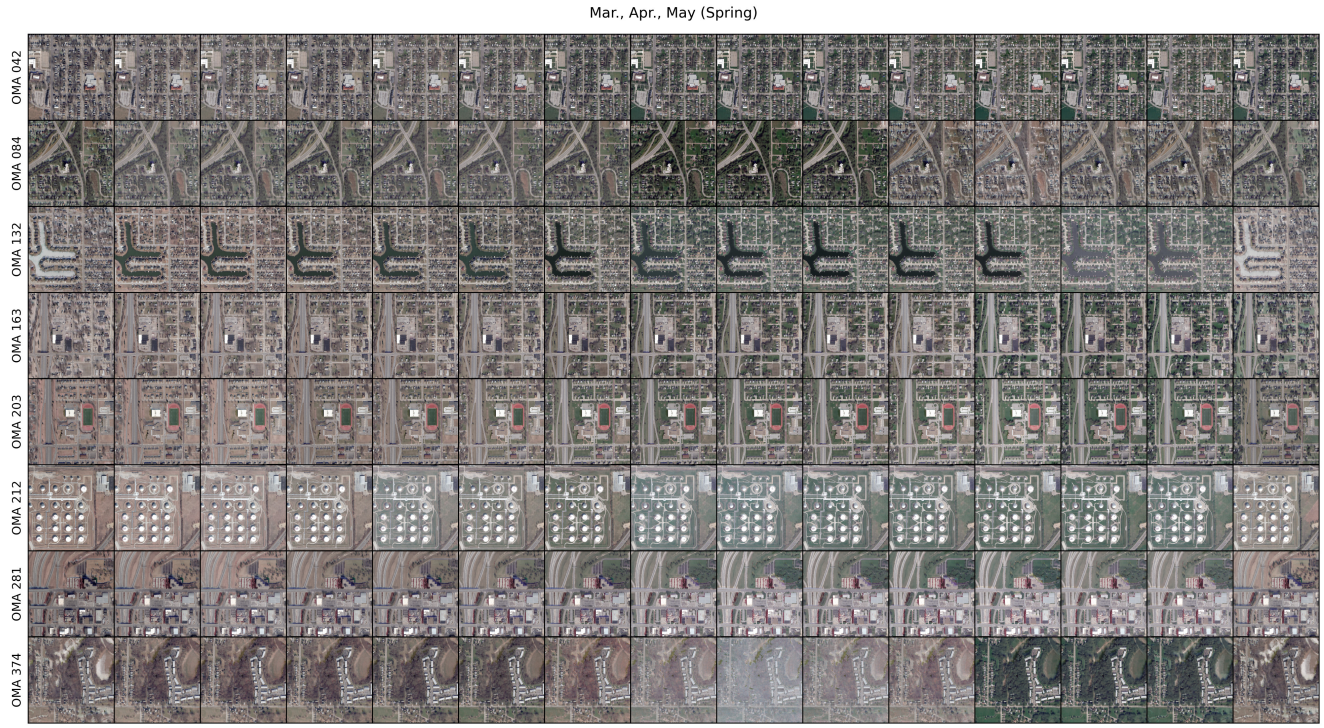

Fig. 11. Images rendered approximately six days apart during March, April, and May. They approximate the spring season.

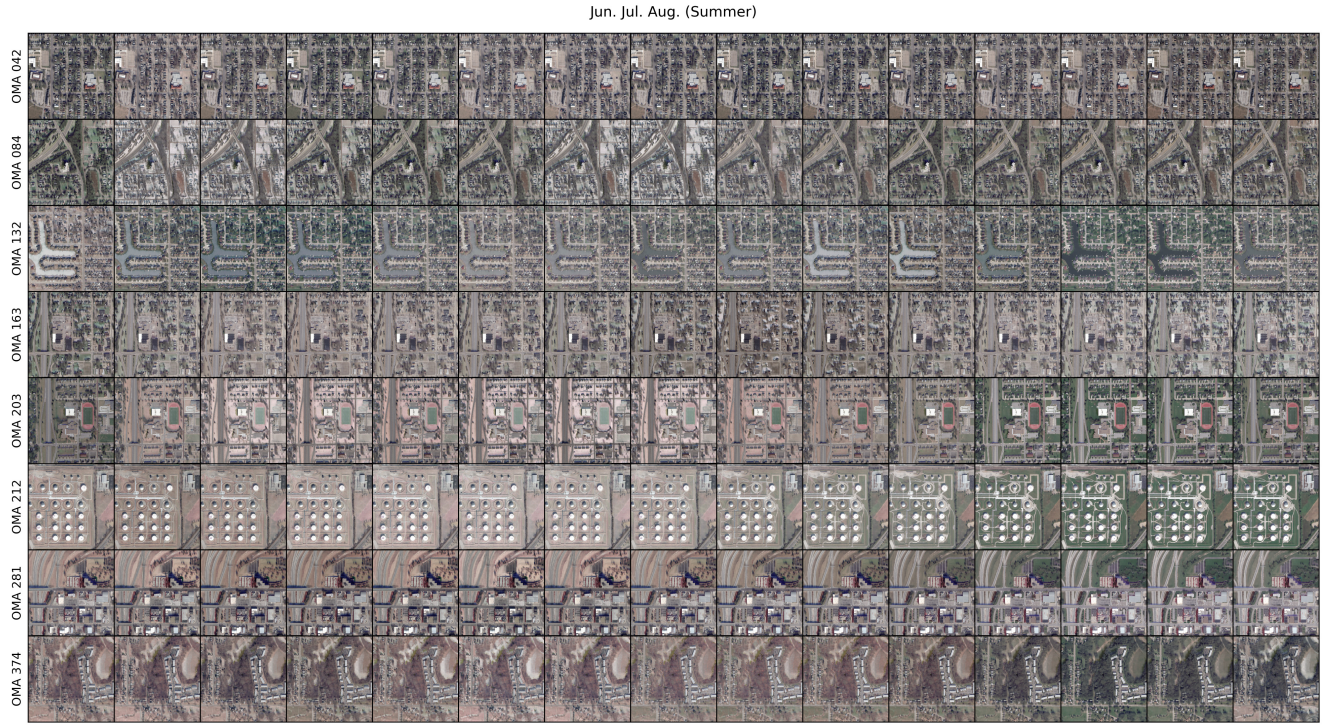

Fig. 12. Images rendered approximately six days apart during June, July, and August. They approximate the summer season.

Fig. 13. Images rendered approximately six days apart during September, October, and November. They approximate the Fall season.

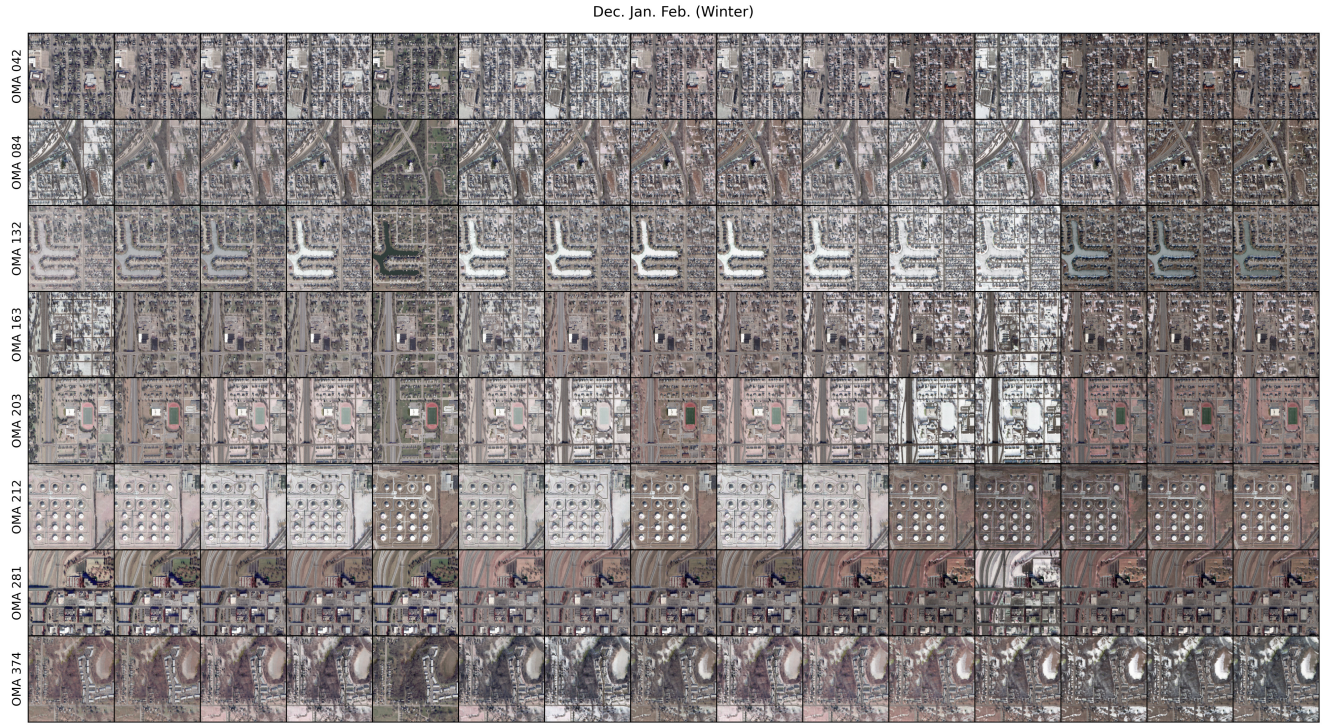

Fig. 14. Images rendered approximately six days apart during December, January, and February. They approximate the Winter season.

TABLE 8
Results on small regions. Case A: Full Model on Full Region. Case F: Full Model without downsizing and parameters tuned for small regions. Case G: Full model without down-sampling default parameters.

Region	SSIM, SA ↑			MAE ↓		
Case:	A	F	G	A	F	G
OMA 042	0.602	0.586	0.562	2.473	1.957	2.357
OMA 084	0.571	0.566	0.529	2.823	2.442	3.769
OMA 132	0.674	0.702	0.704	1.219	1.040	1.279
OMA 163	0.585	0.532	0.527	1.431	1.466	2.073
OMA 203	0.656	0.705	0.696	1.171	0.987	1.429
OMA 212	0.610	0.711	0.679	0.939	0.889	1.102
OMA 281	0.636	0.634	0.618	1.461	1.312	2.296
OMA 374	0.533	0.464	0.473	4.436	4.216	4.211
Average	0.608	0.613	0.599	1.994	1.789	2.314

TABLE 9
A summary of the initial and best hyperparameters found as a result of the hyperparamter tuning process. LR: learning rate, λ_{SC} weight for solar correction terms, λ_{DS} weight for depth supervision terms, n: number of seasonal classes κ: scale for shadow mask μ: Shift for shadow mask $\mathbb{S}$: Maximum sky color $\mathbb{A}$: Minimum color without shadows

Region	$\log_{10}(LR)$	λ_{SC}	λ_{DS}	n	κ	μ	$\mathbb{S}$	$\mathbb{A}$	SSIM	MAE
Low Res	-4.84	0.03	1.00	4	30.00	0.20	0.50	0.20	0.53	2.52
High Res	-4.37	0.46	2.63	8	37.91	0.10	0.73	0.15	0.57	1.70

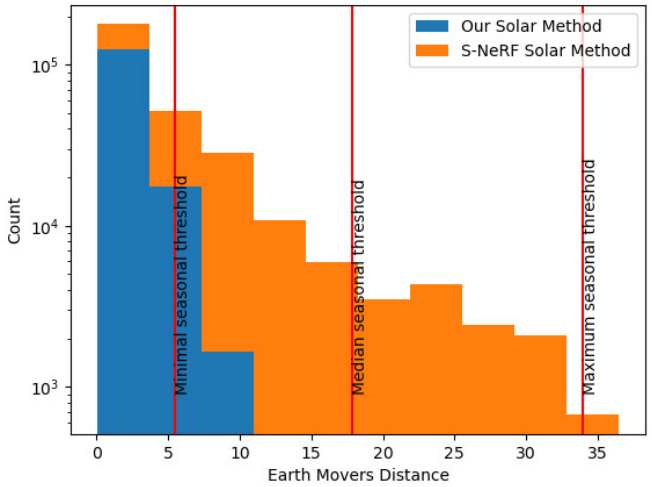

Fig. 15. A histogram of the EMD amongst images with varying views and solar angles but a fixed time of the year.

TABLE 10
Summary of the performances for each of the variations of Season-NeRF. Case A: Full Model. Case B: Full Model, S-NeRF Solar loss. Case C: MSE Loss. Case D: No Height Map. Case E: No Seasonal Adjustment.

Cases:	A	B	C	D	E
Seasonal Variation	X	X	X	X	
Seasonal Stability	X		X	X	X
Accurate DSM	X	X	X		
Good Shadow Recall	X			X	

features using a NeRF. Since seasonal features are obviously linked to the time of the year, Season-NeRF uses the time of the year to compute seasonally adjusted albedo colors. In addition, Season-NeRF uses the viewing and solar angles to determine which regions of the rendered image contain shadows. The network architecture of Season-NeRF ensures the time of the year can only alter the seasonal features by limiting the time of the year to change the seasonally adjusted albedo and leaving other outputs independent of the time of the year. While the method for rendering shadows described in S-NeRF [4] works well, it allows the solar angle to influence the seasonal features unduly. To discourage the network from using the seasonally adjusted albedo to explain shadows, we alter how shadows are computed and modify the loss function. We expand the loss function to punish the network for outputting dark colors via the seasonally adjusted albedo. Furthermore, we punish the network for using trivial sky colors. Finally, we freeze parameters not used in the computation of the solar visibility or sky color when training the network with solar rays and freeze parameters used only in the computation of solar visibility when training with image rays. The ability to render seasonal features can also cause the network to learn less accurate densities. To balance this tendency, we provide a height map to our model as a guide for the early stages of training.

The primary purpose of Season-NeRF is to account for seasonal changes within satellite images. To this end, we introduced temporal adjustment capability into our NeRF process. In addition to this, Season-NeRF uses a novel approach for depth supervision that utilizes a height map rather than a set of 3D points as is used on [6]. In the future, we would like to compare the performance of NeRFs using different depth supervision techniques with varying quality of prior information. In addition, we would like to

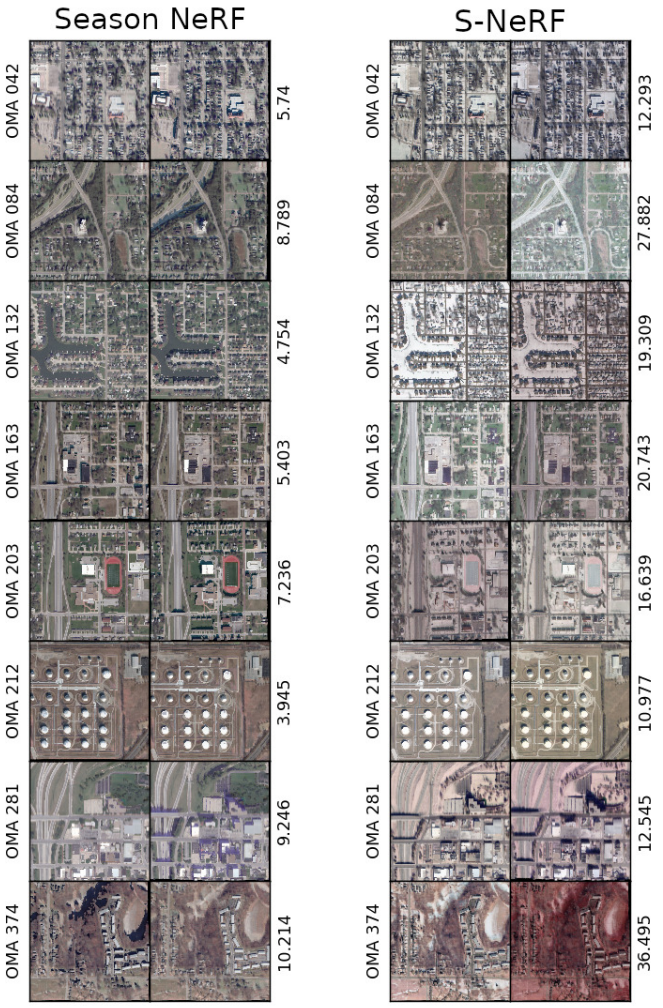

Fig. 16. Examples of the image pairs with the largest EMD for each region. The EM distance between the two images is on the right side of each column.

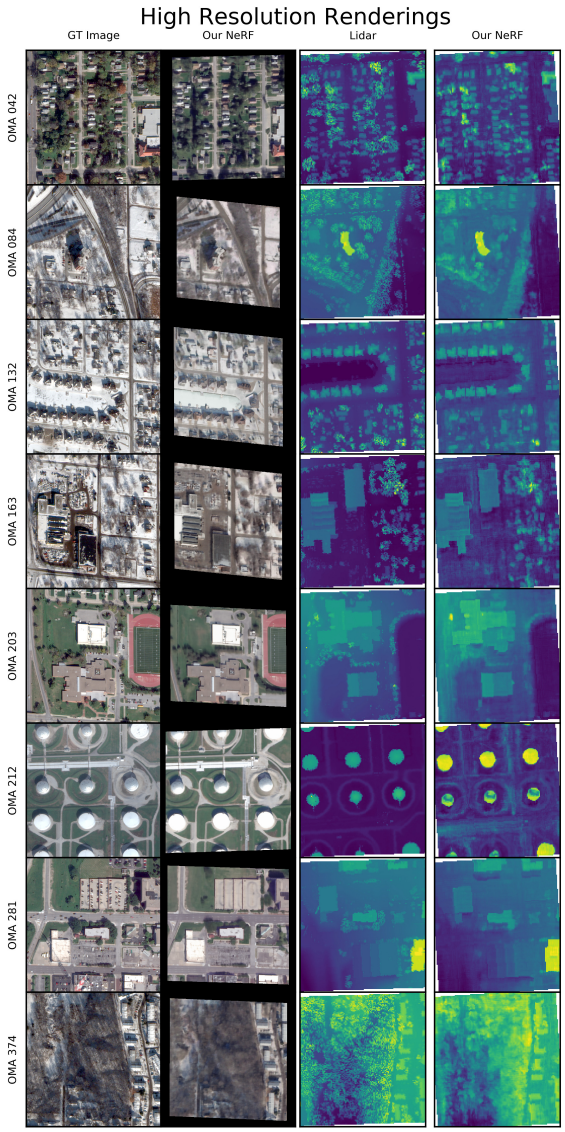

Fig. 17. Results with high resolution images on small regions.

incorporate the findings of [30] to improve the performance of transient object removal for Season-NeRF.

We show the performance of Season-NeRF in eight AOIs from the *2019 IEEE GRSS Data Fusion Contest* [1], which contains images captured by the Maxar WorldView-3 satellite between 2014 and 2016. We show how, by including seasonal variation, we can improve the quality of the rendered image as measured by PSNR and SSIM. By including a height map, we can improve the mean altitude error over every region by an average of .6 meters. Using Barron's loss function, we can improve the average shadow recall by 13%. However, Barron's loss does not have a discernible impact on the quality of the height map or rendered image. As summarized in Table 10, each aspect of the Season-NeRF framework contributes to the capabilities of the radiance field. Our method for computing shadows allows solar features to remain independent from seasonal features despite a limited amount of training data. As a result, we can specify solar and seasonal features for novel view renderings of scenes captured by satellite images.

REFERENCES

[1] B. Le Saux, N. Yokoya, R. Hänsch, and M. Brown, "Data fusion contest 2019 (dfc2019)," 2019. [Online]. Available: https://dx.doi.org/10.21227/c6tm-vw12 1, 6, 17

[2] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *CoRR*, vol. abs/2003.08934, 2020. [Online]. Available: https://arxiv.org/abs/2003.08934 1, 3, 6

[3] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," *CoRR*, vol. abs/2008.02268, 2020. [Online]. Available: https://arxiv.org/abs/2008.02268 1, 3, 4

[4] D. Derksen and D. Izzo, "Shadow neural radiance fields for multi-view satellite photogrammetry," *CoRR*, vol. abs/2104.09877, 2021. [Online]. Available: https://arxiv.org/abs/2104.09877 1, 3, 16

[5] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" 2017. [Online]. Available: https://arxiv.org/abs/1703.04977 1

[6] K. Deng, A. Liu, J. Zhu, and D. Ramanan, "Depth-supervised nerf: Fewer views and faster training for free," *CoRR*, vol. abs/2107.02791, 2021. [Online]. Available: https://arxiv.org/abs/2107.02791 2, 3, 4, 6, 9, 16
[7] J. T. Barron, "A more general robust loss function," *CoRR*, vol. abs/1701.03077, 2017. [Online]. Available: http://arxiv.org/abs/1701.03077 2, 4
[8] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron, "Nerv: Neural reflectance and visibility fields for relighting and view synthesis," *CoRR*, vol. abs/2012.03927, 2020. [Online]. Available: https://arxiv.org/abs/2012.03927 3
[9] R. Marí, G. Facciolo, and T. Ehret, "Sat-nerf: Learning multi-view satellite photogrammetry with transient objects and shadow modeling using rpc cameras," 2022. [Online]. Available: https://arxiv.org/abs/2203.08896 3, 4, 9
[10] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," 2021. 3
[11] Y. Wei, S. Liu, Y. Rao, W. Zhao, J. Lu, and J. Zhou, "Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo," 2021. 3
[12] G. Yang, W. Zhou, H. Peng, D. Liang, T. Mu, and S. Hu, "Recursive-nerf: An efficient and dynamically growing nerf," *CoRR*, vol. abs/2105.09103, 2021. [Online]. Available: https://arxiv.org/abs/2105.09103 3
[13] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "Nerf++: Analyzing and improving neural radiance fields," *CoRR*, vol. abs/2010.07492, 2020. [Online]. Available: https://arxiv.org/abs/2010.07492 3
[14] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Deformable neural radiance fields," *CoRR*, vol. abs/2011.12948, 2020. [Online]. Available: https://arxiv.org/abs/2011.12948 3
[15] Q. Meng, A. Chen, H. Luo, M. Wu, H. Su, L. Xu, X. He, and J. Yu, "Gnerf: Gan-based neural radiance field without posed camera," 2021. 3
[16] C. Lin, W. Ma, A. Torralba, and S. Lucey, "BARF: bundle-adjusting neural radiance fields," *CoRR*, vol. abs/2104.06405, 2021. [Online]. Available: https://arxiv.org/abs/2104.06405 3
[17] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo," 2021. 3
[18] A. Jain, M. Tancik, and P. Abbeel, "Putting nerf on a diet: Semantically consistent few-shot view synthesis," 2021. 3
[19] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images," 2021. 3
[20] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *CoRR*, vol. abs/2006.09661, 2020. [Online]. Available: https://arxiv.org/abs/2006.09661 3
[21] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491. 3
[22] J. Grodecki and G. Dial, "Block adjustment of high-resolution satellite images described by rational polynomials," *Photogrammetric Engineering and Remote Sensing*, vol. 69, pp. 59–68, 01 2003. 4
[23] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," 2017. [Online]. Available: https://arxiv.org/abs/1708.07120 6
[24] K. Kutulakos and S. Seitz, "A theory of shape by space carving," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, 1999, pp. 307–314 vol.1. 6, 7
[25] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," 2012. 6
[26] S. Patil, B. Comandur, T. Prakash, and A. C. Kak, "A new stereo benchmarking dataset for satellite images," *CoRR*, vol. abs/1907.04404, 2019. [Online]. Available: http://arxiv.org/abs/1907.04404 7
[27] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. 7
[28] M. Bosch, Z. Kurtz, S. Hagstrom, and M. Brown, "A multiple view stereo benchmark for satellite imagery," 10 2016, pp. 1–9. 7
[29] Y. Rubner, C. Tomasi, and L. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, pp. 99–121, 11 2000. 10
[30] S. Sabour, S. Vora, D. Duckworth, I. Krasin, D. J. Fleet, and A. Tagliasacchi, "Robustnerf: Ignoring distractors with robust losses," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 20 626–20 636. 17

Michael Gableman is a graduate student in the School of Electrical and Computer Engineering School at Purdue University, West Lafayette, IN, USA. He received a bachelor's degree in mathematics and computer science from Ripon College, Ripon, WI, USA, in 2016 and a master's degree in computer and engineering from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA in 2019. His research interests include computer vision, machine learning, neural radiance fields, and satellite imagery.

Avinash C. Kak received the Ph.D. degree in electrical engineering from the Indian Institute of Technology, Delhi, India, in 1970. He is currently a Professor of electrical and computer engineering with Purdue University, West Lafayette, IN, USA. His coauthored book Principles of Computerized Tomographic Imaging (SIAM, 2001) was republished as a classic in applied mathematics. His other coauthored book, Digital Picture Processing, also considered by many to be a classic in computer vision and image processing, was published by Academic Press in 1982. His more recent books were written for his "Objects Trilogy" project. All three books of the trilogy have been published by John Wiley & Sons. The first, Programming with Objects, came out in 2003, the second, Scripting with Objects, in 2008, and the last, Designing with Objects, in 2015. His research interests include algorithms, languages, and systems related to wired and wireless camera networks, robotics, computer vision, etc.