Spicing it up with 🔥SpicyNeRF🌶️: Novel View Synthesis Using Thermal Images

Griffin Holt
Stanford University🌲
Department of Electrical Engineering
gholt@stanford.edu

Andrew Zhang
Stanford University🌲
Department of Electrical Engineering
azhang82@stanford.edu

Abstract

In this report, we present SpicyNeRF🌶️, a preliminary method of applying and adapting Neural Radiance Field (NeRF) methods to generate novel thermal views of a scene. As a baseline, we separately run a variation of the vanilla NeRF model Nerfacto on a paired set of both thermal and traditional RGB images of the same scene, where each thermal photo corresponds to an RGB photo taken simultaneously from the same device. By comparing the results, we found that the first major hurdle to the NeRF pipeline was an accurate estimation of the parameters of the thermal camera used. This paper explores different methods to overcome this. We first explored parameter estimation methods using classical computer vision techniques. Specifically, we extracted the extrinsics of the RGB photos and directly mapped them to the thermal photos since they are taken from the same view. The thermal camera intrinsics were then estimated separately using COLMAP; however this was shown to be untenable. We then instead adopt a neural solution which makes the thermal camera extrinsics and intrinsics parameters to be learned during NeRF training. We show that this is a massive improvement over classical techniques. However, there still exist many limitations as the neural net implementation struggles to train on 360-degree view datasets, causing errors in the rendering. Over this, we finally present SpicyNeRF🌶️, which leverages a presupposed corresponding RGB dataset to create better initializations of the extrinsics of the thermal images for better training.

1. Introduction

Novel view synthesis (NVS) is the task of generating a new view of a scene given a sparse set of input images of the scene. For example, given a number of images of an object from different vantage points, one would like to infer what the object would look like had an image been captured from a different vantage point.

1.1. Vanilla NeRF

The advent of NeRF [6] in recent years has presented a shift in focus toward neural based representations for novel view synthesis, where neural networks are now used to create neural implicit representations of the scene. In its original formulation, a set of camera poses are first given. Each pose is a five dimensional vector consisting of a 3D spatial vector $\mathbf{x}$ as well as a 2D look direction $\mathbf{d}$ parameterized by two angles. Ideally, these would represent a diverse set of different views of the same scene. This 5D vector defines the location of an image plane consisting of $H \times W$ pixels. It is the value of the color at each of these pixels that then need to be evaluated in order to represent the view of the scene. To do this, thinking along the lines of the classical ray tracing algorithm, we can imagine that from each camera location through each pixel there exists an inward coming ray coming from the scene. If we evaluate the color information this light ray carries as it propagates through space, we can compute what the color would be at each pixel. The next step then is to query 3D points in space along each of these outgoing rays given a ray direction and see what color information is carried at that point. Specifically, at each point, we will want to compute an RGB value as well as a volume density σ, which represents how "transparent" that point is and how likely a ray is to travel through it. In essence then, the neural network takes as input a spatial location $\mathbf{x}$ as well as the 2D ray direction $\mathbf{d}$, and returns a 4D $RGB\sigma$ vector. The intuition behind also inputting the ray direction is that the color of an object can change depending on what direction you view it from.

Despite the capacity of neural networks to universally approximate any function, it is known that they

are biased towards learning low frequency functions [8]. This can present problems in the case of NeRF when complex textures of a scene need to be represented by high frequency color variations. To tackle this, each of the 3D query points $\mathbf{x}$ and 2D ray directions $\mathbf{d}$ are mapped onto a higher dimensional input $\gamma(\mathbf{x})$, $\gamma(\mathbf{d})$ before finally being input into the network, similar to that of positional encoding. Doing so allows the neural net to learn higher frequency functions to accurately represent color variation.

By providing a set of images with known camera poses, the neural net is purposely overfit to the scene, learning the color information corresponding to each of the points in space to allow for a rendering that matches the training data. This is in contrast to most neural network applications, where overfitting is generally undesired. In the end, the weights of the network in a way "represent" that of the scene, and in a very meaningful way can be thought of as a compressed version of a 3D model of the scene.

More specifically, in order to finally render a new view of the scene, a new camera pose is given, and rays are again sent out. Along the ray are sampled positions, and at each position, the neural net evaluates the corresponding volume density σ and color information $\mathbf{c}$. The final color value of the ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ (where $\mathbf{o}$ and $\mathbf{d}$ represent the camera position and ray direction respectively) with near and far bounds t_n and t_f is as follows:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \tag{1}$$

where $T(t) = \exp\Big(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\Big)$.

Evaluation of the integral in equation 1 across all pixels in the image plane allows for the final rendering of a view of a scene. This can then be compared with the ground truth images to calculate an L2 loss as described by

$$\mathcal{L} = \sum_{r\in\mathcal{R}} \left[\left|\left| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right|\right|_2^2 + \left|\left| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right|\right|_2^2 \right] \tag{2}$$

where $\mathcal{R}$ represents the set of all rays used, $C(\mathbf{r})$ represent the ground truth color, and $\hat{C}_c$, $\hat{C}_f$ represent the color values of a ray calculated using a coarse and fine sampling technique respectively. Performing gradient descent over this loss to update the parameters of the network allow the model to then overfit to the scene and then represent it implicitly from newly queried viewpoints. Although different variations of NeRF may use different sampling techniques leading to different loss functions, the essence of each one is same which involves optimizing over a photometric-based loss.

1.2. SpicyNeRF🌶️

NeRF allows for extremely inexpensive 3D renderings of objects, which can lead to a wide range of applications from AR to scene representation for autonomous driving. It is of interest then of many to extend the capabilities of NeRF to non visible parts of the spectrum.

One possibility is adapting NeRF methods to thermal images. Thermal images have a wide array of use cases, such as reconnaissance and medical imaging. Its appearance, however, presents many challenges for NeRF. The vanilla NeRF model requires images to have known camera extrinsics and intrinsics, which represent the camera poses and various coefficients such as focal length, image size, and distortion coefficients. These parameters, particularly the extrinsics, are unavailable in most application settings, and so are instead estimated using classical structure from motion techniques like COLMAP [9] before being input to NeRF. These work by matching different features across a set of images of a view of an object, but thermal images often are unable to provide that information due to their relative lack of textural information, and so fail to produce proper camera extrinsics and intrinsics.

We propose instead SpicyNeRF🌶️, an alternative NVS pipeline that adapts NeRF to thermal images. Rather than having the NeRF model require the extrinsics and intrinsics to be attached to each image, we instead make them parameters to be learned during training time, allowing us to wholly bypass the need for COLMAP or other parameter estimation methods. This has been described in many works [14]; however, there have been many difficulties in optimizing these parameters. We presuppose then the existence of a set of corresponding RGB photos of each thermal image, and further refine the learning procedure by initializing the learnable extrinsics to the ones extracted from COLMAP on the RGB photos. We find that these neural based methods begin to give a much more realistic rendering of a thermal scene.

2. Related Work

We separately survey two categories of related work in the NVS literature: one that assumes camera parameters are known, and one that assumes them to be unknown.

2.1. Parameters Known

For many NVS pipelines, input images are required to have the appropriate camera extrinsics and intrinsics. This is the case in vanilla NeRF, but also in many other methods, such as Mip-NeRF [1], Scene Representation Networks [10], Occupancy Networks [5], and Multiplane Imaging (MPIs) [2]. These methods are all unified in that a volumetric function of a scene is learned by using photometry-based optimization, and each has its own benefits by leveraging different image features and varying sampling techniques. However they also all require camera poses to be known, which in many applications such as ours is often unavailable.

2.2. Parameters Unknown

So we instead turn to works where the intrinsics and extrinsics need not be provided. These parameters are instead to be jointly optimized along with the scene geometry. Traditional work in this field include the aforementioned COLMAP, which works by matching features across a set of images. However, such a system is expected to fail on thermal images which lack distinct features.

More recent work has looked to using neural based methods to also estimate camera parameters. In this line of work, we have the state of the art BARF [4], however this implementation is only capable of learning the extrinsics, and still requires the intrinsics to be provided. Over this there exist works such as NeRF– [6] which makes both the intrinsics and extrinsics learnable parameters of the NeRF model. Limitations however include the assumption that the training data represents only forward facing scenes, meaning all photos of the scene are taken from one plane. Optimization over the special orthogonal group $SO(3)$, which represent the set of possible rotation matrices that the camera extrinsics can take on, has been known to be difficult due to properties of its algebra group [13], and so learning the poses for a full 360 degree scene remains an outstanding challenge.

3. Methodology

3.1. Classical CV Methods

As a baseline, we first run the thermal images through a variation of the vanilla NeRF model Nerfacto [12], which combines components of different variations of NeRF to achieve a balance between speed of training and quality of rendering. We use the popular NeRF API, Nerfstudio [11], which modularizes the pipeline and allows for easy viewing of the rendering during training time and with that more qualitative analyses of the final scene.

Nerfstudio works by first taking the input images and estimating each one's extrinsics and intrinsics via COLMAP. The batch of processed images is then fed to the flavor of NeRF chosen. The training process, as well as the estimated camera extrinsics, can then be visualized in a viewer.

To overcome the poor camera pose estimation expected of COLMAP for thermal parameter estimation, we leverage the fact that we also have a paired set of RGB photos taken simultaneously with the thermal images. The camera parameters estimated using COLMAP [9] for the RGB photos can then be naively transferred to each corresponding thermal image. We expect the extrinsic mapping to be justified, since the camera lens for both cameras were very proximal on the device we used. The intrinsic mapping, which involves a transfer of the distortion coefficients, the image size parameters, as well as focal lengths, is less than justified, as each camera should have extremely different intrinsics. We instead attempt to estimate these intrinsics separately apart from the extrinsics using COLMAP again. After performing this mapping, we can then run the pose-estimated thermal images through Nerfstudio.

3.2. Neural Based Methods

Beyond classical CV methods, we also utilize a slightly modified version of the implementation as described in NeRF– [14] to instead make the camera parameters jointly learnable with the scene rendering during training. The neural net, which as described above takes as input a high dimensional projected spatial coordinate $\gamma(\mathbf{x})$, $\gamma(\mathbf{d})$ and outputs the 4D color information, is tiny by deep learning conventions, but can still perform extremely well. Specifically, the projected spatial coordinate $\gamma(\mathbf{x})$ is first passed through 4 Linear-ReLU layers. The output from this is then separately used in two ways, one to calculate the scalar density σ using a linear layer, and the other to be passed through another linear layer before being concatenated with the projected look angle $\gamma(\mathbf{d})$ of the image. By calculating the density before concatenation with the look angle, we ensure that the density is independent of the vantage point. After concatenation, the vector is then passed through one final linear-ReLU layer before finally being passed through a last linear layer to produce an RGB 3-vector. A diagram of this network architecture is shown in Figure 1.

In addition to the traditional learnable weights and biases of the MLP layers specified above we make two additional categories of parameters learnable as described in the NeRF– paper: the camera intrinsics and extrinsics.

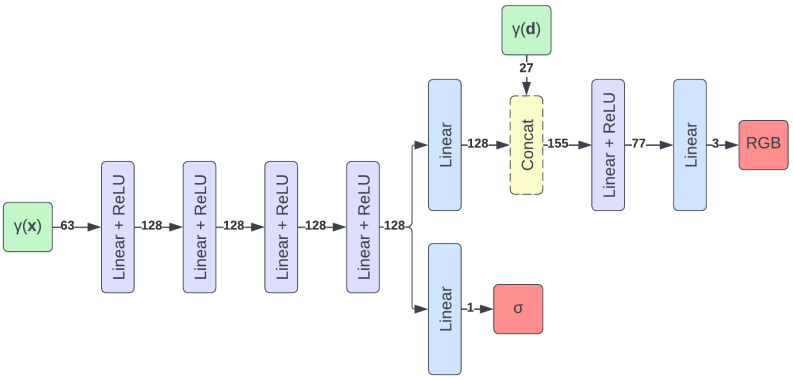


Figure 1. Architecture of the neural network used to compute the density σ and RGB values at a given spatial coordinate and look angle by taking as input the projected spatial coordinate $\gamma(\mathbf{x})$ and the projected look angle $\gamma(\mathbf{d})$. Numbers in between layers represent the dimension of the output.

For a pinhole model of a camera, the intrinsics involve the focal length f as well as the principle points c_x and c_y, which in our case can be considered to be the center of the image so that $c_x \approx W/2$ and $c_y \approx H/2$, where H and W are the dimensions of the input image in pixels. The only learnable intrinsic then is the focal length.

The extrinsics of the camera involve knowing the camera's position in space as well as its orientation. This can all be summarized by the camera-to-world transformation matrix $\mathbf{T}_{wc} = [\mathbf{R}|\mathbf{t}]$, where $\mathbf{R} \in SO(3)$ and $t \in \mathbb{R}^3$. $\mathbf{R}$ is any matrix that represents a rotation in Euclidean space about the origin. This means that it must preserve the origin, Euclidean distance, as well as the orientation. The t vector can straightforwardly be thought of as a translation vector that positions the camera in $\mathbb{R}^3$.

From these parameters, we can then begin the ray-tracing algorithm used in the NeRF pipeline. To render the color value of a pixel p of a given image I_i, we send out the ray $\hat{\mathbf{r}}_{i,p}(h) = \hat{\mathbf{o}}_i + h\hat{\mathbf{d}}_{i,p}$, from the ray origin $\hat{\mathbf{o}}_i = \hat{\mathbf{t}}_i$ in the ray direction

$$\hat{\mathbf{d}}_{i,p} = \hat{\mathbf{R}}_i \begin{pmatrix} (u - W/2)/\hat{f} \\ (v - H/2)/\hat{f} \\ -1 \end{pmatrix}, \tag{3}$$

where u and v are the pixel location in the image grid. We sample a number of 3D points along this ray and evaluate the color information using the NeRF model.

We can see that the above equation is fully differentiable in the camera parameters when finally calculating the L2 loss of the final rendered color with respect to some training set, so one may naively make the parameters $\hat{\pi}_i = (\hat{f}, \hat{\mathbf{R}}_i, \hat{\mathbf{t}}_i)$ all learnable parameters. However this does not work in the case of the rotation matrix $\hat{\mathbf{R}}_i$. Because $\hat{\mathbf{R}}_i$ must lie in $SO(3)$ space, it can be problematic to carry out a straightforward gradient-descent based backpropagation update on $\hat{\mathbf{R}}_i$. We instead learn an alternative parameter $\phi = \alpha\omega$, where ω is a normalized rotation axis, and α is the rotation angle from said axis and varies from 0 to π. Each ϕ can be mapped to a valid rotation matrix using Rodrigues' formula

$$\mathbf{R} = \mathbf{I} + \frac{\sin(\alpha)}{\alpha}\phi\hat{} + \frac{1 - \cos(\alpha)}{\alpha^2}(\phi\hat{})^2, \tag{4}$$

where $(\cdot)\hat{}$ represents the skew operator and converts a vector to a skew-symmetric matrix.

Because the Rodrigues' Formula translates a vector in $\mathbb{R}^3$ to a matrix in $SO(3)$ and is differentiable, we can instead make ϕ a learnable parameter in place of the rotation matrix and more easily carry out gradient descnet updates.

Mathematically, gradient descent based optimization should allow for a full recovery of the correct parameters. However, without good initializations, it can be extremely difficult for the network to fully learn the correct ϕ for each image again due to properties of $SO(3)$ space. As a result, the authors of NeRF– run their implementation exclusively on forward facing datasets. This means that all images were taken from a single plane facing forward, with only small perturbations in rotation. By initializing ϕ to zero for each image, the network only needs to learn a small deviation from the identity rotation to match the ground truth camera rotation matrix. For datasets that involve full 360 degree views, we expect the model to perform much worse.

As an improvement over running our surround-view dataset naively through NeRF–, we leverage the availability of a corresponding RGB dataset which can be run through COLMAP to give camera extrsinics.

Similar to our baseline method, we can then map these extrinsics to the thermal ones by appropriately initializing the corresponding ϕ network parameters. COLMAP however outputs the estimated rotation matrices, and so the ϕ for each image must be back calculated by deriving the appropriate α and ω. This can be done by observing that ω is the eigenvector of rotation matrix $\mathbf{R}$ with associated eigenvalue $\lambda = 1$, i.e. $\mathbf{R}\omega = \omega$. α can be derived by observing that

$$\mathbf{Tr}(\mathbf{R}) = 1 + 2\cos(\alpha), \tag{5}$$

which takes advantage of the trace properties of the skew matrix.

Rearranging for α we obtain

$$\alpha = \pm \arccos\left(\frac{1}{2}\mathbf{Tr}(\mathbf{R}) - 1\right), \tag{6}$$

where the sign of α is determined by testing whether $+\alpha$ or $-\alpha$ correctly reconstructs $\mathbf{R}$.

By extracting $\mathbf{R}$ for each RGB image, we can then derive the corresponding $\phi = \alpha\omega$ and appropriately initialize each ϕ parameter for every thermal image. COLMAP also provides the $\mathbf{t}$ vectors, which can be directly used as an initialization in the network. We believe that providing a better initialization close to the ground truth, backpropagation can allow the network to quickly converge to the ground truth extrsinics.

An obvious limitation to this final proposed method is the necessity of having available corresponding RGB photos. In many applications, such as night time reconnaissance, it may be difficult to obtain corresponding visible light photos. Fortunately, for our specific application, which was exploring thermal images as used in detecting shorts in electronic circuits, RGB photos are readily available. In fact, many cameras provide the option of simultaneously taking both an RGB and thermal image, which justify our use of extrinsic mapping.

4. Dataset and Features

The novelty of using thermal images for NeRF required us to make our own multi-view thermal image dataset. We are grateful to Dr. Rivez who allowed us to use the thermal cameras used in their power electronics lab SUPER-lab.

We specifically used SUPER-lab's FLIR 363900 thermal camera, which can simultaneously capture both thermal and RGB photos. The RGB and thermal photos are 640 x 480 and 320 x 240 pixels large respectively.

In total we captured 181 pairs of thermal-RGB photos. To allow for maximal thermal and textural variety, we chose to image an active hotplate with a heat sink placed on top. Two pairs of the raw images are shown in Figure 2.

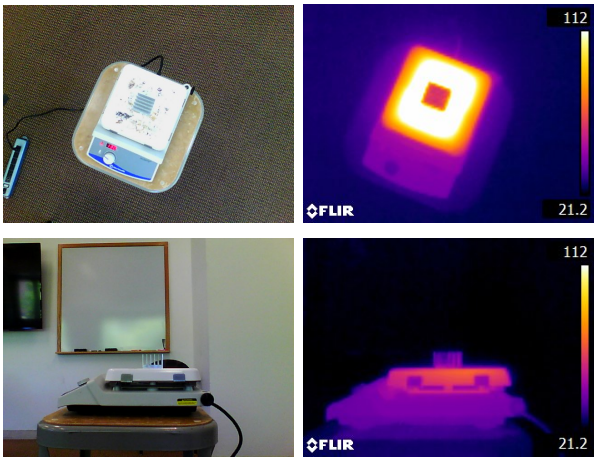


Figure 2. Two pairs of RGB and thermal photos of an active hot plate with a heat sink placed above

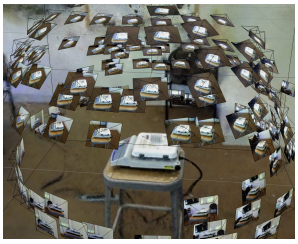
As one can see, the FLIR camera software exports photos with an irremovable watermark and temperature scale. Cropping was thus needed, and photos were taken to account for this. In order to preserve the original principle points, the photos had to be center cropped, resulting in a final resolution of 280 x 186.

Most machine learning applications require the use of a separate training and test dataset to quantify the generality of the final trained model. In NeRF contexts, a test dataset is often unable to be obtained because we have no ground truth labels for the camera parameters, a necessary prerequisite. In all of our methods, the ground truth camera parameters are never known, and so it does not make sense to make a training/evaluation dataset split. Most of the time it makes much more sense to rely on qualitative analyses of the success of the final rendering, which can be easily done by creating a video or using a viewer.

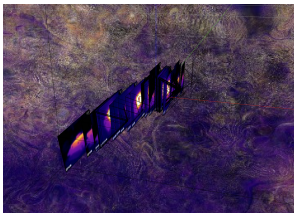
5. Experiments, Results, and Discussion

5.1. Classical Techniques

We first ran our collected RGB and thermal data through the Nerfacto model using Nerfstudio. All parameters were set to their default values, and we did a qualitative analysis using the viewer. As expected, as a preprocessing step into the neural model, Nerfstudio uses COLMAP to extract the camera parameters. As expected, the algorithm completely fails on the thermal images, being unable to even given rough estimates. The API itself complains that there were not enough features detected in the images. As a result, we



(a) RGB



(b) Thermal (Uncropped)

Figure 3. Pose estimations visualized as given by COLMAP. We can see that the algorithm struggles with thermal images. Note that the thermal images used are the uncropped ones. COLMAP completely failed to give pose estimations for the cropped thermal images

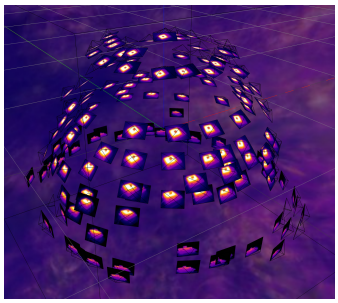
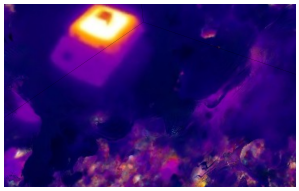


Figure 4. Thermal poses after extrinsic and intrinsic transfer.

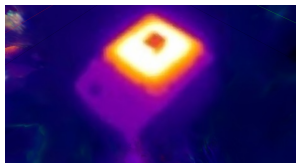
were not even able to begin training on the data. For contrast, we ran the same pipeline on the RGB photos, which worked as expected and successfully generated high quality novel views. The estimated poses are shown in Figure 3. We additionally show the esimated poses used when we use the uncropped thermal images with the watermark and temperature scale attached. COLMAP successfully runs but estimates them to all face the same direction, which we believe occurs because all of the detected features are in the constant watermark. This illustrates nonetheless the inability to detect features on the thermal image itself.

To properly run the model, we then naively transferred the estimated camera parameters for the RGB photos to the thermal ones. This result is shown in Figure 4. Although the extrinsic mapping is well justified, the intrinsics are less so since we expect the focal length and image size to be completely different for the thermal camera. This can be seen when we view the rendering from the same angle as a given training example. We expect these to be at least as good since the neural net should easily overfit to the training data.

We noticed that the "correct" image is actually contained within that rendering, however it is off-centered. Indeed, by adjusting the intrinsics to account for the



(a) Before



(b) After

Figure 5. Rendering viewed from the vantage point of a training example's before and after partial intrinsic correction.

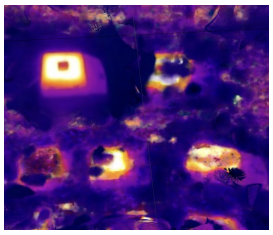


Figure 6. Rendering after partial intrinsic correction from a vantage point not contained in the training data.

correct image dimensions, we see that the rendering produces a much better view, as shown in Figure 5.

Despite the success of the neural net to fit to the training data, the rendering completely failed when a novel view was queried. This is illustrated in Figure 6.

The only intrinsic left to estimate then was the focal length. We tried to extract the intrinsics separately from the extrinsics using COLMAP again, however it once again completely failed, and so a full correction was unable to be made.

For each of these methods we record the peak signal to noise ratio (PSNR) in Table 1. PSNR is one of the most widespread quantiative metrics used to measure NeRF performance, and is described by:

$$PSNR = 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \quad (7)$$

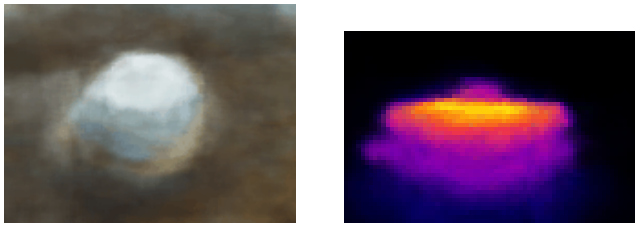
where MAX_I is the maximum possible value of the image, and MSE is defined below and represents the average pixel by pixel square difference across the entire training set.

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (8)$$

PSNR acts as a proxy for the L2 distance, but is flipped in that a higher value is desired. PSNR is usually expressed in units dB. Despite the utility of quantitative metrics, we hope that our figures serve as a much better metric in gauging the success (or lack thereof) of our renderings.

Case	PSNR (dB)
RGB	20.73
Thermal (Uncropped)	12.17
Thermal (Naive Parameter Transfer)	9.97
Thermal (Partial Intrinsic Correction)	10.64

Table 1. PSNR Results



(a) RGB (b) Thermal

Figure 8. Novel rendering with 0 initialization

5.2. Neural Techniques

Over the sub-optimal results obtained using traditional computer vision techniques, we next turned to neural based solutions in making the camera parameters learnable as part of network training. Our implementation was done in PyTorch [7] following the architecture described in Section 3.2 and is based off the Colab implementation given in the NeRF– GitHub repo [14].

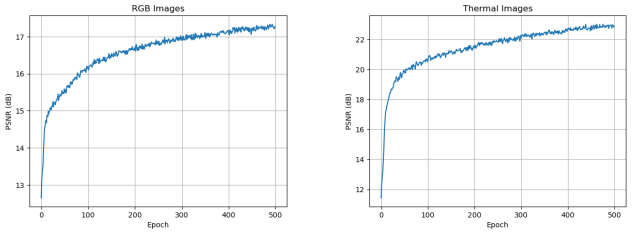
Specifically, we used Kaiming initialization [3] for the network weights, and first assume completely unknown camera parameters. The extrinsics are thus initialized at $\mathbf{t} = 0$ and $\phi = 0$, which means that each camera starts at the origin facing the $-z$ direction. The focal lengths f_x and f_y are respectively initialized to H and W. The focal lengths, camera poses, and network weights are each separately trained using a different Adam optimiser, but they all start with a learning rate of 0.001. For the network weights, the learning rate decays every 10 epochs by a factor of 0.9954. The focal length and pose learning rates decay every 10 epochs by a factor of 0.9. Everything is jointly trained for 500 epochs, unless otherwise mentioned.

The PSNR of the model during training is graphed in Figure 7 for the case of completely unknown camera parameters for both RGB and thermal photos. We also display a novel view generated for both modalities in Figure 8. We attach all further renders made in the Appendix. We urge the reader to additionally inspect the GIFs attached to this report to form a better qualitative judgement of the results.

From the figures, we can see that PSNR achieves good convergence. We also see that for the thermal case, the results are both quantitatively and qualitatively a massive improvement over all previous methods discussed. The rendering actually produces a rigid body that much resembles the original hot plate system. However, we can see that from certain angles, the bottom of the hotplate appears much less well-rendered, and overall the body seemed "amorphous", meaning the effective point cloud was not well-formed. We confirmed this by analyzing the RGB case, where we see the model struggled much more in comparison to the traditional Nerfacto method. Indeed, the renderings show something more of a "cloud" of a hotplate, rather than a solid body. The better performance of the thermal images over the RGB ones in this case may be due to the lack of texture in the first place of the thermal images, a case where the texture-less feature of thermal images act instead as a performance enhancer. Nevertheless, much of these difficulties were anticipated as mentioned in the previous discussion on the difficulty of adapting this method to full view datasets, and the massive improvement over previous failures to even begin to estimate the parameters cannot be understated.

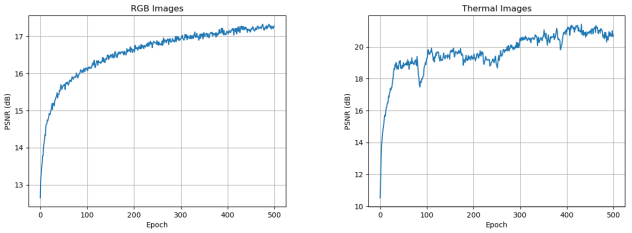
5.3. SpicyNeRF 🌶️

Finally, to adapt NeRF– to our 360 degree dataset, we leveraged the availability of our RGB dataset and initialized the t and ϕ vectors of the thermal images according to the procedure outlined in Section 3.2. We



(a) RGB Images (b) Thermal Images

Figure 7. PSNR vs. Epoch using 0 initialization



(a) RGB Images (b) Thermal Images

Figure 9. PSNR vs. Epoch using extrinsics initialization

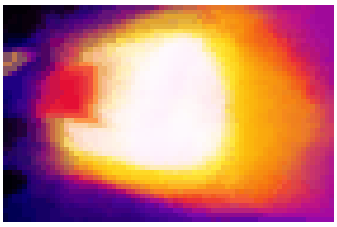


Figure 10. Novel thermal render using extrinsic initialization.

ran our entire RGB dataset through COLMAP, and then transformed the extracted extrinsics to the appropriate ϕ and t values. The network parameters were then initialized accordingly. The PSNR of the model during training for both RGB and thermal images is shown in Figure 9, and can be further visualized by looking a novel rendering shown in Figure 10. Again, additional renderings are attached in the Appendix, and a GIF is attached to this report.

In this case, we notice again good convergence of PSNR for both modalities, however there was a noticeable downgrade in performance with regard to both quantitative and qualitative metrics for the thermal case (however it still represents a great improvement over non-neural based parameter estimation methods). The RGB case performed almost exactly the same, which we were especially surprised to see given that this method has been shown to be robust for 360 degree RGB datasets when an approximate parameter prior is supplied. Our initial guess as to why this failed was because COLMAP uses a different coordinate system from that of NeRF–. This is a known problem, as some models, such as the original NeRF, use a coordinate system known as Normalized Device Coordinates (NDC), which normalizes every point in the scene to some range between -1 and 1. As to why the thermal images performed better with 0 initialization, we presuppose that our coordinate systems were incompatible, meaning the initialized R matrices could have been facing completely away from the scene for some images. This may lead to worse loss compared to a 0 initialization where at least every image is facing the scene.

6. Conclusion and Future Work

Overall, we have seen that neural-based parameter estimation methods wholly overcome the challenges presented when using classical computer vision methods in applying NeRF to thermal images. Renderings are overall much better well-formed on both the qualitative and quantitative front, as can be seen in our provided renderings.

In the end, SpicyNeRF🌶️ failed to generalize to 360 degree scenes, and performed worse than when 0 initialization was carried out. We are confident however that this can be easily overcome through a simple coordinate conversion between NDC and world coordinates, assuming our guess that the coordinate system mismatch is the cause of our poorer results. Immediate work in the then future can look into making that conversion, or even adapting the existing model to world coordinates.

Even if it worked, a major limitation of SpicyNeRF🌶️ is the requirement of a corresponding RGB dataset, which in many applications is unavailable. If 360 degree view thermal datasets are to be used for NeRF, more research must be done in seeing how ground truth camera parameters can be extracted with aribtrary initialization. We have shown in this report that without a corresponding RGB dataset, one is limited to forward facing scenes only.

Looking further, we believe there to be many additional hurdles in applying NeRF properly to thermal images beyond just parameter estimation. The rendering equation used in all of these NeRF variations was designed to account for how visible light travels through a medium. Thermal imaging, which is carried out in the long-wave infrared (LWIR) part of the spectrum, may behave quite differently when interacting with matter. If we are to accurately reconstruct the surface temperature of the scene rather than matching the arbitrary color scale used to represent LWIR in our images, then a more sophisticated rendering premise and equation may be needed.

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021. 3

[2] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent, 2019. 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. 7

[4] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields, 2021. 3

[5] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space, 2019. 3

[6] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 1, 3

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. 7

[8] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019. 2

[9] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3

[10] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations, 2020. 3

[11] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023. 3

[12] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, et al. Nerfstudio: A modular framework for neural radiance field development. *arXiv preprint arXiv:2302.04264*, 2023. 3

[13] Camillo J Taylor and David J Kriegman. Minimization on the lie group so (3) and related manifolds. 1994. 3

[14] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021. 2, 3, 7

Appendix: Rendered Novel Views

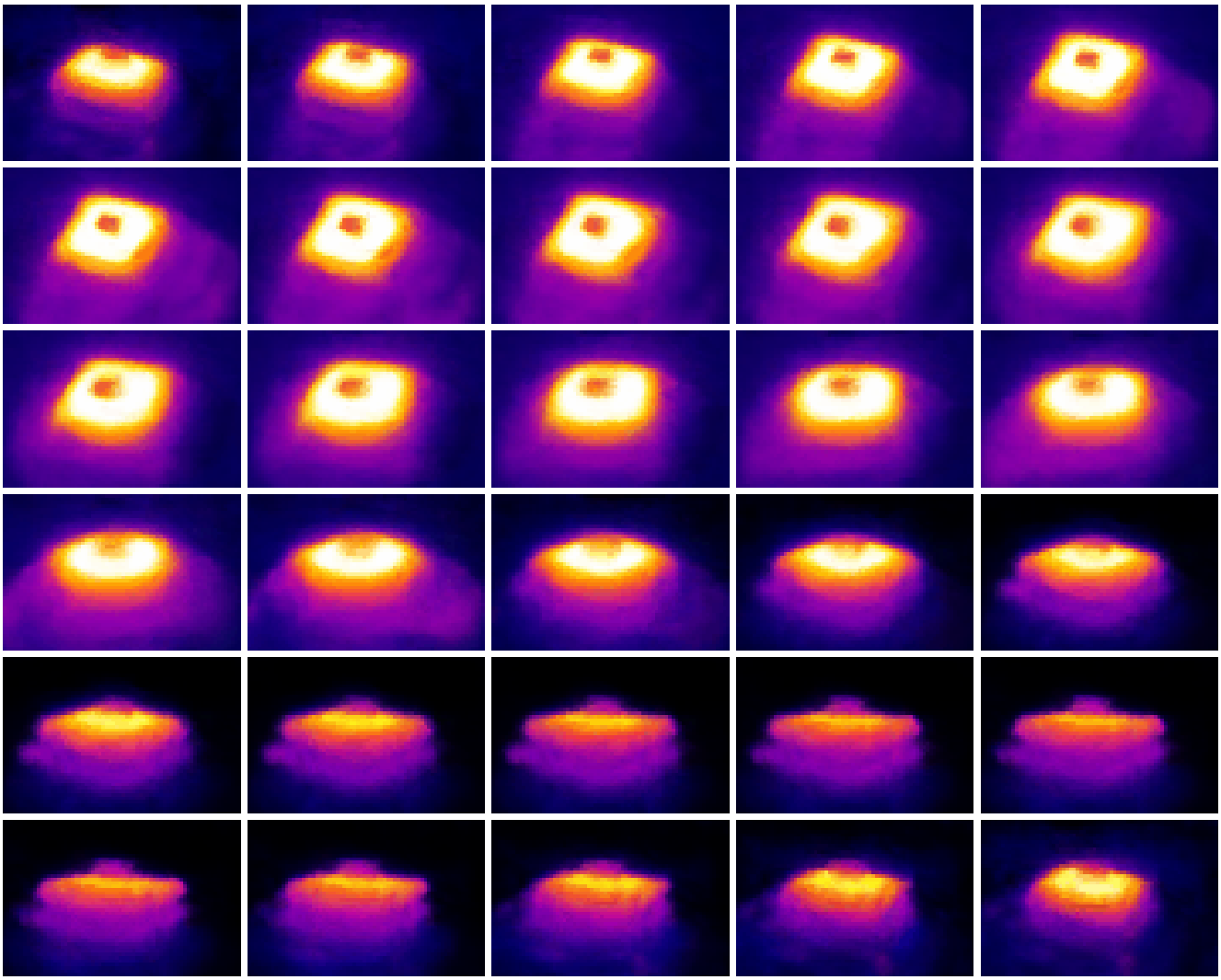


Figure 11. Rendered novel views using 0 initialization for thermal images.

Figure 12. Rendered novel views using 0 initialization for RGB images.

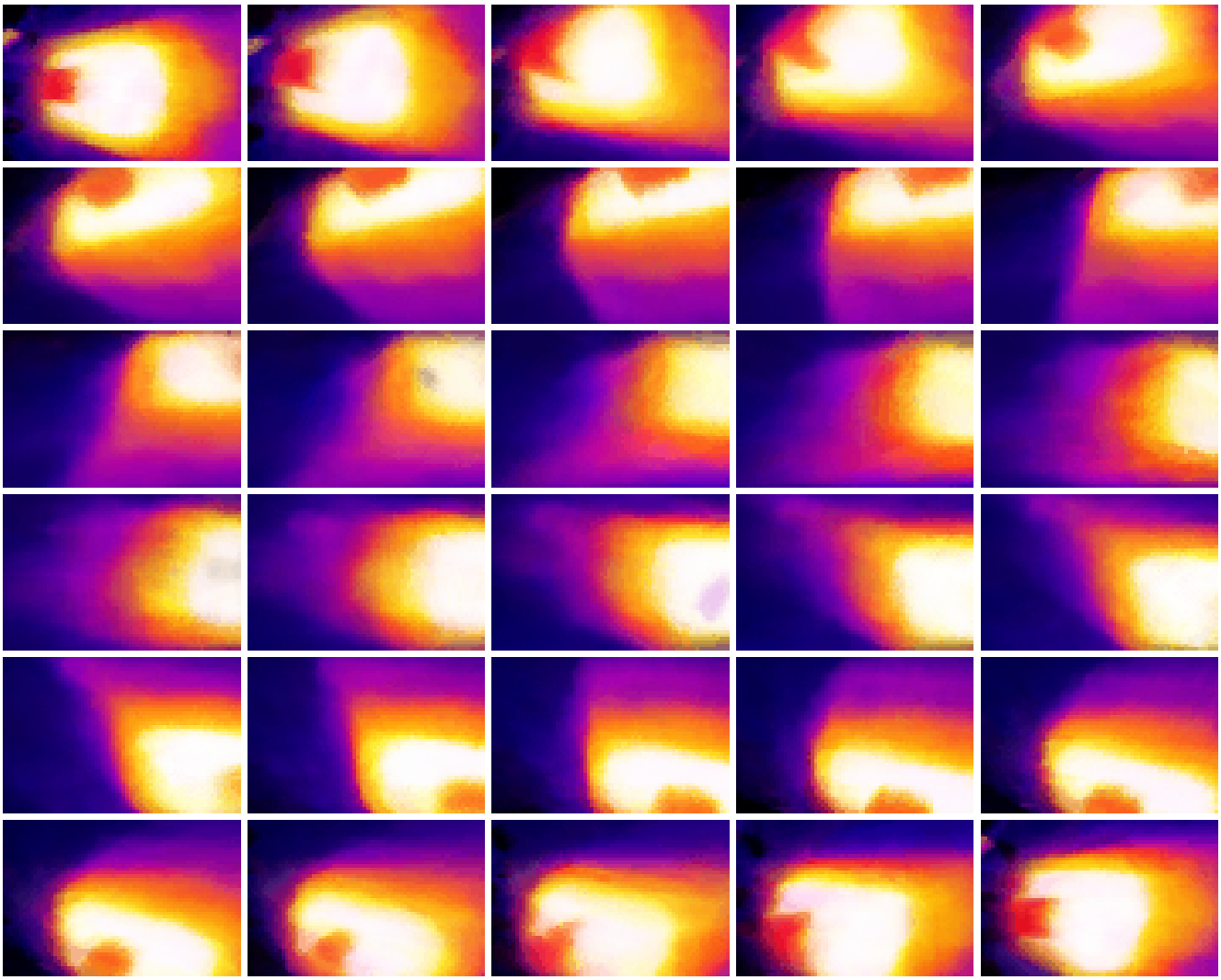


Figure 13. Rendered novel views using extrinsic initialization for thermal images.