

# Touch-GS: Visual-Tactile Supervised 3D Gaussian Splatting

Aiden Swann<sup>\*1</sup>, Matthew Strong<sup>\*2</sup>, Won Kyung Do<sup>1</sup>, Gadiel Sznaiers Camps<sup>3</sup>,  
Mac Schwager<sup>3</sup>, Monroe Kennedy III<sup>1,2</sup>

**Abstract**—In this work, we propose a novel method to supervise 3D Gaussian Splatting (3DGS) scenes using optical tactile sensors. Optical tactile sensors have become widespread in their use in robotics for manipulation and object representation; however, raw optical tactile sensor data is unsuitable to directly supervise a 3DGS scene. Our representation leverages a Gaussian Process Implicit Surface to implicitly represent the object, combining many touches into a unified representation with uncertainty. We merge this model with a monocular depth estimation network, which is aligned in a two stage process, coarsely aligning with a depth camera and then finely adjusting to match our touch data. For every training image, our method produces a corresponding fused depth and uncertainty map. Utilizing this additional information, we propose a new loss function, variance weighted depth supervised loss, for training the 3DGS scene model. We leverage the DenseTact optical tactile sensor and RealSense RGB-D camera to show that combining touch and vision in this manner leads to quantitatively and qualitatively better results than vision or touch alone in a few-view scene syntheses on opaque as well as on reflective and transparent objects. Please see our project page at [armlabstanford.github.io/touch-gs](https://armlabstanford.github.io/touch-gs).

## I. INTRODUCTION

Accurate 3D scene and object representations are an essential aspect of robotic interactions with an environment. Neural Radiance Fields (NeRF) [1] have gained prominence as an effective 3D representation. NeRFs have been applied to a number of robotics challenges, including path planning [2] and manipulation [3]. The method of 3D Gaussian Splatting (3DGS) [4] has recently advanced the field by providing high-quality and high-speed training along with real-time rendering. Precise representations and real-time visual reconstruction are important aspects for robots to interact with their environments. However, in most cases, visual-only information is not sufficient to interact with complex objects. Situations as common as reaching into a cluttered drawer or grasping something in the dark necessitate a sense of touch.

Recent advances in gel-based optical tactile sensors like DenseTact and GelSight [5], [6] have enabled robots to have a sense of touch. Many works have addressed the use of tactile sensors in robotic applications including topics such as manipulation [7], [8] and shape reconstruction [9], [10]. In this work, we enhance 3DGS by using the fusion of visual-tactile data. Just as humans seamlessly integrate both

This research was supported by NSF Graduate Research Fellowship No. DGE-2146755 and NSF Grant No. 2142773, 2220867.

[<sup>1</sup>] are with the Department of Mechanical Engineering, [<sup>2</sup>] Department of Computer Science, [<sup>3</sup>] Department of Aeronautics and Astronautics, Stanford University, Stanford CA, USA. Emails: {swann, mastro1, wkdo, gsznaier, schwager, monroek}@stanford.edu

<sup>\*</sup>Both authors contributed equally to this work.

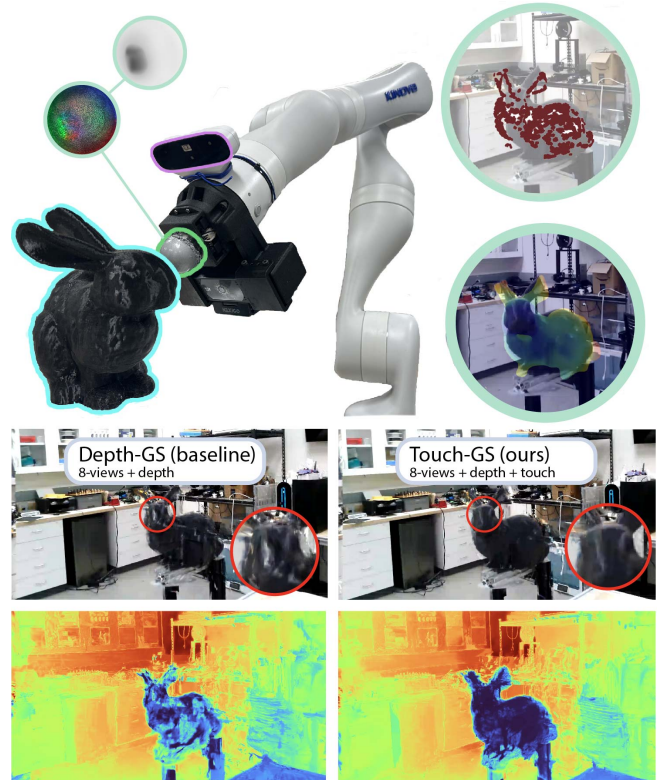


Fig. 1: Touch-GS combines monocular depth estimation priors with tactile data-informed implicit surfaces to generate high-quality 3DGS scenes from few training images. Adding touch data significantly enhances 3DGS quality (right) compared with RGB-D alone (left).

touch and vision to achieve a unified understanding of their environment, robots can leverage this essential integration between touch and vision for interaction with the world. In a robotic environment, it is not feasible to collect hundreds of images to train a 3DGS. In this approach, we utilize both monocular depth priors and tactile data to generate high-quality few-shot and challenging scenes, such as few-view object-centric scenes, mirrors, and transparent objects.

**Depth Supervision for NeRFs.** NeRF visual and geometric quality can be improved through depth supervision. Several works have utilized Structure from Motion (SfM) keypoints, and monocular depth estimation to train few-view NeRFs [11], [12], [13]. These prior works demonstrate enhanced scene quality, but still require many views to estimate camera poses and sparse keypoints. This was addressed in [14], which does not use COLMAP. However, none of these works utilize touch data, which can provide more accurate

depth information.

**Tactile Sensing for Robotic Tasks.** Recent research has shown that the high resolution of vision-based tactile sensors allows for rich tactile information and increased sensing capabilities [6], [5]. They can be used for object manipulation tasks [15], [7], localization [10], shape reconstruction, and even classification of unknown objects [16], [17].

**Gaussian Process Implicit Surface (GPIS) Usage for Robot Manipulation.** Signed Distance Field (SDF)s are a common implicit approach to represent a 3D object as a 0-level-set of an SDF function. Typically, SDFs are represented as learned functions [18], [9]. GPISs, use a Gaussian Process (GP) to represent an SDF [19] with applications to both robotic manipulation and object representations [20], [21]. The advantage of a GPIS is the ability to leverage a continuous object representation derived from discrete touches along with the variance associated with the GP model.

**Fusing Vision and Touch for Robotics.** In addition to the GPIS capability with tactile information from the sensor, using multi-modal input including vision, depth, or touch has been addressed widely in multiple tasks including creating a shared multimodal space [22], [23]. Among the current works, it is important to reconstruct an accurate real-world object mesh to solve the manipulation task [24], [3]. Unlike tactile-only or many-view vision-based shape reconstruction [25], [26], this work aims to reconstruct the object from a small number of RGB images with touch input to estimate an accurate object mesh out of RGB, depth, and touch.

### A. Key Contributions

We propose Touch-GS, the first approach that combines both tactile and visual data to train 3DGS. Our main contributions in Touch-GS are as follows:

- We introduce a Gaussian Process Implicit Surface (GPIS) to synthesize tactile data in a representation suitable for supervising 3DGS training.
- We optimally fuse the touch GPIS and monocular depth estimation through Bayesian Inference to create depth and uncertainty images for additional training supervision for the 3DGS beyond the RGB images.
- We show qualitative and quantitative improvements across a variety of scenes in few-view scene synthesis, including scenes with mirrored and transparent objects.
- Our method can supply touch supervision to improve any other Neural Radiance Field representation beyond 3DGS, e.g., Nerfacto [27] or the original NeRF [1].

The remainder of the paper is outlined as follows. Section II first covers mathematical preliminaries and Section III introduces our method. Section IV showcases our results, and finally, Section V discusses these results and future avenues of research.

## II. PRELIMINARIES

### A. Point-based NeRF and 3DGS

3DGS and other point-based NeRF methods utilize a slightly different representation than standard NeRFs. Rather than represent space as a continuous function where empty

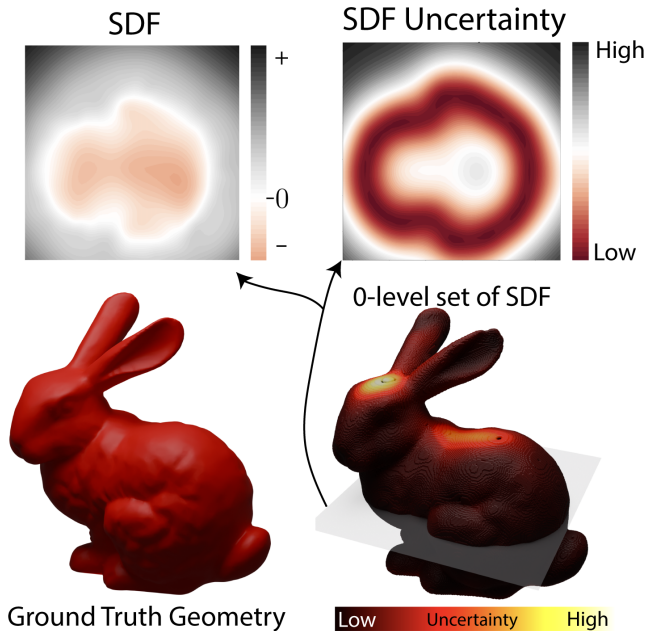


Fig. 2: The GPIS is created by finding the 0-level-set of our GP-based SDF. We utilize the uncertainty at the 0-level-set to enhance the accuracy of our model. A z-axis slice of both SDF and uncertainty is shown above the bunnies.

space is defined implicitly, in these point based methods, free space as truly empty and image formation is done by blending ordered points with overlapping pixels. We define a ray with camera origin  $\mathbf{o}$  and orientation  $\mathbf{d}$  as follows:

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}, \quad t \geq 0, \quad (1)$$

The color  $C(\mathbf{r})$  and depth  $D(\mathbf{r})$  along the ray are computed by blending the set of  $\mathcal{N}$  ordered points intersecting the ray:

$$\hat{C} = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad \hat{D} = \sum_{i \in \mathcal{N}} d_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (2)$$

where  $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$ , and  $d$  is the distance of a particular point along a ray. The following L2 loss between the ground truth (GT) image and volumetric rendered image can be used to optimize the NeRF.

$$\mathcal{L}_c = \sum_{r \in \mathcal{R}} \left\| \hat{C}(r) - C(r) \right\|_2^2 \quad (3)$$

Regardless of the representation, either volume-based or point-based, these loss functions will work with our method. We can then perform depth supervision in a similar manner to color supervision.

### B. Gaussian Process Implicit Surface

An implicit surface is a surface defined by the 0-level set of a function  $f(x)$ .

$$\mathcal{S}_0 \triangleq \{x \in \mathbb{R}^d \mid f(x) = 0\}. \quad (4)$$

In our case,  $f(x)$  is a SDF, which maps every point in 3D space to the minimum distance between that point and a given surface. We opt for a GPIS because it not only encodes distance information but also captures variance

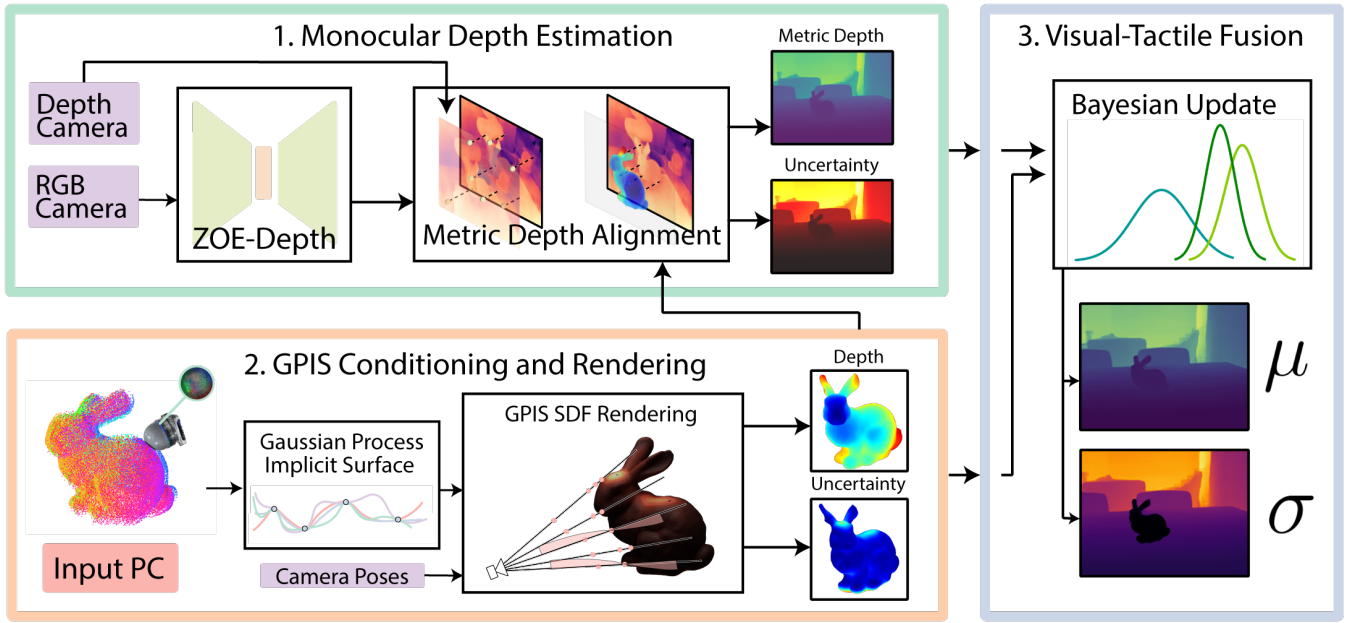


Fig. 3: Overview of our method, Touch-GS: 1. We utilize a monocular depth estimation algorithm, which is metrically aligned in a two phase process with RealSense depth and the GPIS output. 2. We condition a GP on the point cloud generated by DenseTact, rendering this into a series of depth and uncertainty images. 3. Monocular depth and tactile information are combined to produce a single set of training images that combine touch and vision.

information. GPs are an extremely versatile non-parametric machine learning method. We postulate that our SDF  $f(x)$  follows a GP with a given mean and covariance [19]

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \quad (5)$$

The SDF, GP uncertainty, and corresponding 0-level set are shown in Fig. 2. After making a series of observations, we condition and inference on the GP using a CUDA accelerated library called GPtorch [28].

### III. METHOD

Our core method is divided into 3 main modules as shown in Fig. 3: 1. Monocular depth estimation on the few-input view RGB data, 2. GPIS conditioning and rendering on the tactile sensor data, and 3. Visual-tactile fusion. These three steps output a single set of depth and variance images. These fused images are then used to train a 3DGS model with a slightly modified training and initialization procedure. It is important to note that our approach is modular between different methods of NeRF, point-based or volumetric.

#### A. GPIS Conditioning and Rendering

We combine an array of optical tactile sensor measurements, in this case DenseTact, into a single GPIS representation of the object. This fusion of points into a surface is essential for performing supervision on a NeRF. When we touch an object with DenseTact, many areas will be left uncovered, while others will produce conflicting information due primarily to sub-millimeter noise in the manipulator’s estimated pose. The GPIS seamlessly combines this information into a 3D estimate of the surface along with uncertainty at every point.

1) *GPIS Conditioning*: In order to produce an accurate SDF, we need to define both the 0-level set and the gradient with respect to it. We leverage the orientation of the touch to compute the gradient of the surface at each of these points. Following [19], we can enforce this during conditioning by generating artificial points both inside and outside the object with negative and positive values respectively:

$$\begin{aligned} \mathbf{x}_{in} &= \mathbf{x}_0 - \delta \hat{\mathbf{n}}, & f(\mathbf{x}_{in}) &= -\delta \\ \mathbf{x}_{out} &= \mathbf{x}_0 + \delta \hat{\mathbf{n}}, & f(\mathbf{x}_{out}) &= +\delta, \end{aligned} \quad (6)$$

where  $\hat{\mathbf{n}}$  is the normal vector of a given tactile reading which we use as an estimate of the true normal vector  $\mathbf{n}$ , and  $\mathbf{x}_0$  is the set of contact points on the surface of the target object measured by DenseTact, and  $\delta$  is a small positive scalar value representing the offset from the surface. In addition, we add several points near the center of the object with negative values. These are computed by taking the average position along successive Z-slices of the model.

$$\mathbf{c}_i = \frac{1}{|\mathbf{X}_i|} \sum_{\mathbf{x} \in \mathbf{X}_i} \mathbf{x}, \quad \mathbf{x}_{int,i} = \mathbf{c}_i, \quad f(\mathbf{x}_{int,i}) = -\epsilon \quad (7)$$

Where  $\mathbf{X}_i$  is the set of points in the  $i$ -th Z-slice,  $\mathbf{c}_i$  is the centroid of the  $i$ -th Z-slice, and  $\epsilon$  is a small positive value used for the SDF value of the interior. We now condition our GPIS on the following set of points.

$$\mathcal{X} = \{\mathbf{x}_0, \mathbf{x}_{in}, \mathbf{x}_{out}, \mathbf{x}_{int,i}\}, \quad \mathcal{Y} = \{0, -\delta, +\delta, -\epsilon\} \quad (8)$$

While this approximation is not a perfect representation of the true value of the SDF, we find it works well for reasonably shaped objects as shown in Sec. IV. For our GP

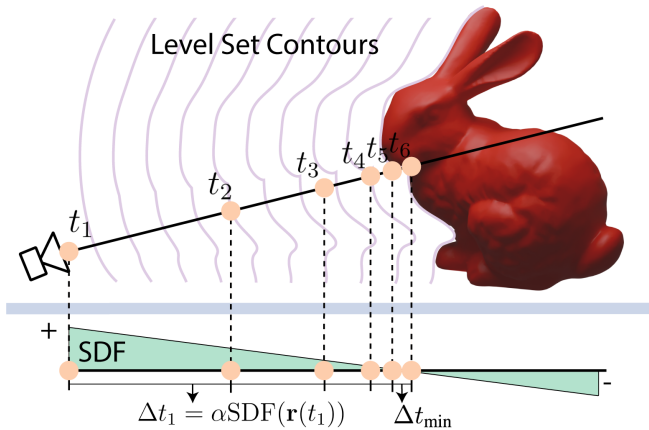


Fig. 4: We show our optimized SDF rendering process.  $\alpha = .5$ , thus each step halves the distance to the surface.

covariance function, we use the Matérn kernel with  $\nu = 3/2$ .

$$C_{3/2}(d) = \sigma^2 \left( 1 + \frac{\sqrt{3}d}{\rho} \right) \exp \left( -\frac{\sqrt{3}d}{\rho} \right) \quad (9)$$

Here,  $d$  is the distance between two points and  $\rho$  and  $\sigma$  are parameters optimizing over during training.

2) *GPIS Rendering*: We leverage the SDF nature of our conditioned GP to render depth and variance images in the poses of each RGB image. The SDF provides a crucial advantage by indicating the shortest distance to the surface, allowing for an optimized ray-marching that skips empty space. Consider the ray defined in (1). The ray advances by increments of  $\Delta t$  determined by the SDF at the current ray position, until the surface is reached or a maximum number of steps is exceeded, as shown in

$$\Delta t = \max(\alpha \text{SDF}(\mathbf{r}(t)), \Delta t_{\min}) \quad (10)$$

Where  $\Delta t_{\min}$  is the minimum allowable step size and  $\alpha$  is a tunable parameter. This variable step size marching is shown in Fig. 4.

3) *Depth and Variance Image Generation*: To render the GPIS from various viewpoints, we incorporate a standard pinhole camera model which maps from 3D world coordinates into pixel coordinates. For each pixel in the RGB image, we compute a corresponding depth value by determining the intersection point of the ray with the object’s surface. This ray marching is done by stepping along each  $r_{ij}$  in a given camera image following (10) until the SDF value is within a specified tolerance of 0. Additional optimizations are done to accelerate the rendering process. Prior to ray marching, we intersect every ray analytically with the minimum volume sphere, which covers a coarse 0-level set. This eliminates rays which will not intersect the surface. The variance image is generated by evaluating the uncertainty in the GPIS at these intersection points. GPIS surfaces along with uncertainties are shown in Fig. 5 and Fig. 6.

### B. Monocular Depth Estimation and Alignment

While the GPIS representation produces high-quality depth information around the target object, it lacks coverage

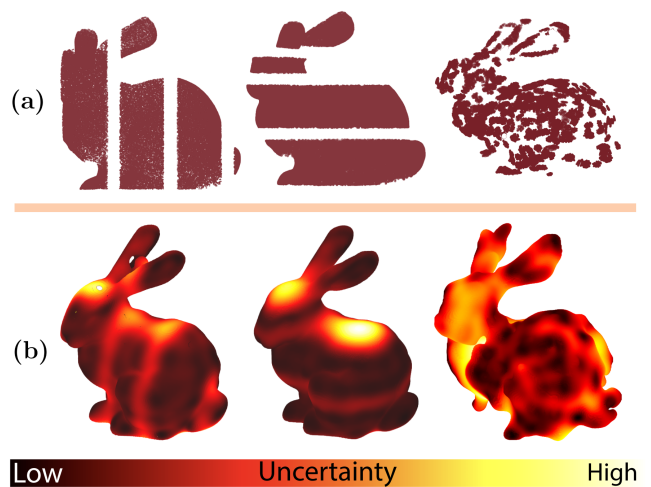


Fig. 5: (a) shows the input point clouds and (b) shows the rendered 0-level-set colored by uncertainty. The method is able to fill in the gaps in the point cloud while showing more uncertainty in the interpolated areas. The last column is a real-world dataset.

on the background of the scene. We posit that improving full scene coverage will improve the overall quality of a NeRF reconstruction and geometry of a scene, and even localize the target object in the scene. To supplement this background coverage, we utilize a monocular depth estimation and alignment procedure, as shown in Fig. 3, which outputs an estimated absolute scale depth map of the scene. While this closely follows the primary method in [11], our method produces absolute world frame estimates that can be deployed onto a real-world robotic system, where we rely on a robot’s kinematics and off-the-shelf depth sensors to align model depth estimations.

1) *ZoeDepth Estimation and Alignment*: We utilize the state-of-the-art monocular depth estimation network ZoeDepth [29], but our approach can be applied to any metric or relative monocular depth estimator. Using the raw output of ZoeDepth is not enough; in order to train a NeRF, it is necessary to align the output of ZoeDepth with real-world sparse depth data and touch depth data. To best align the output of ZoeDepth to the scene, we perform a straightforward two-step alignment procedure to construct the most accurate depth map to train a NeRF.

a) *Sparse Depth Scene Alignment*. With an off-the-shelf depth sensor (e.g., the Intel RealSense), we can perform a depth alignment to jointly learn a scale factor and offset to compute an aligned depth map  $D_{\text{ZOE}}^*$ . Concretely, for each image, we formulate the first alignment stage as a least squares problem shown below:

$$s^*, t^* = \arg \min_{s, t} \sum_{p \in D_{\text{sparse}}} \|D_{\text{sparse}}(p) - D_{\text{ZOE}}(p; s, t)\|^2 \quad (11)$$

where  $s^*$  is the scene scale factor,  $t^*$  is the offset, and  $p$  is a sparse depth keypoint from sparse depth image  $D_{\text{sparse}}$ .

Our approach also works for SfM algorithms such as COLMAP, which estimate camera poses and outputs a set

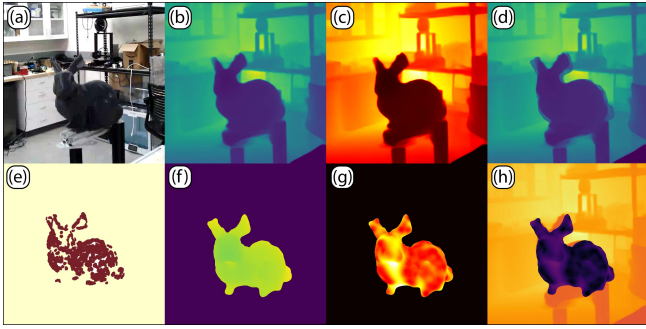


Fig. 6: The snapshots of various parts of the method on the real dataset: (a) RGB image, (b) Monocular depth, (c) Monocular uncertainty, (d) Fused touch and monocular depth, (e) DenseTact point cloud, (f) GPIS depth, (g) GPIS uncertainty, (h) Fused touch and monocular uncertainty.

of sparse keypoints per image. These sparse points are traditionally used as direct supervision or aligning a monocular depth map; however, a successful COLMAP run requires at least 30 images with meaningful visual features to output a list of sparse keypoints. Instead, our method relies on the estimation of camera poses via a robot’s kinematics.

*b). Object Alignment.* While alignment from sparse depth data grounds the overall scale and offset of ZoeDepth, depth data from the GPIS is still useful to further align our vision depth maps. Because the touch data is from one object, we only learn an offset  $t_{GPIS}$  for each depth image. This is implemented as simply reapplying the same linear alignment but constraining the scale to be 1, and only updating the depth of the object in the ZoeDepth output. This step reshifts the object but does not update meaningful background depths.

*2) Vision Uncertainty Estimation:* We compute a simple visual uncertainty map as the original depth multiplied by a simple scalar value. Contrary to [13], our monocular depth estimator does not provide us with a corresponding estimation of depth uncertainty. To this end, we rely on a simple heuristic for uncertainty: farther away depth values should be given more uncertainty, which is reasonable given that the ZoeDepth model is less accurate as distance increases. We also add a simple constant to the result to give touch more priority when fusing the two. Along with a depth map  $\mu_{ZOE}$ , we now have uncertainty map  $\sigma_{ZOE}^2$ .

### C. Depth and Touch Fusion

The output of the parallel pathways of our method are two sets of depth images, both in the same frame, along with corresponding variance information. We combine these two data sources into a single depth and variance supervision pair. This ensures that during training, we weight the touch and monocular depth data relative to their respective uncertainty.

*1) Bayesian Update:* We treat the problem of fusing the two depth images as a Bayesian update with the dense depth as our prior and the tactile data as our measurement. We assume that the distributions are normally distributed and

that each measurement is independent.

$$\begin{aligned}\sigma_{fuse}^2 &= \left( \frac{1}{\sigma_{ZOE}^2} + \frac{1}{\sigma_{GPIS}^2} \right)^{-1} \\ \mu_{fuse} &= \sigma_{fuse}^2 \left( \frac{\mu_{ZOE}}{\sigma_{ZOE}^2} + \frac{\mu_{GPIS}}{\sigma_{GPIS}^2} \right)\end{aligned}\quad (12)$$

This update rule is applied pixelwise to the outputs of monocular depth estimation and GPIS conditioning / rendering. An example set of input and fused images is shown in Fig. 6.

### D. Model Training

In training our NeRFs, certain depth values should be valued more than others; a depth from touching an object should present a tighter constraint on depth supervision than the estimated depth of an object a few meters away. Additionally, few-input view NeRFs require a good initialization to avoid local minima. Accordingly, we propose a new loss function and novel improvement to the 3DGS initialization.

*1) Uncertainty Weighted Depth Supervision:* With our new variance and depth images, we not only have mean information but also uncertainty; we propose a new cost function, called *uncertainty weighted depth supervision*,

$$\mathcal{L}_d^{(i)} = \alpha^{(i)} \left\| \tilde{d}^{(i)} - d^{(i)} \right\|_2^2, \text{ where } \alpha^{(i)} \propto e^{-w\sigma^{(i)}}, \quad (13)$$

where  $\sigma$  is the uncertainty given by the variance image and  $w$  is a tunable weight parameter. The total combined loss can be expressed as  $\mathcal{L} = \mathcal{L}_{color} + \lambda \mathcal{L}_d$ . This loss function is weighted based on the uncertainty of our GPIS mean. We also add an optional depth loss weight decay shown as  $\lambda_{i+1} = \beta \lambda_i$ , where  $\beta$  is a value between 0 and 1. We recommend this to get a NeRF out of local minima if the depth weight is too high.

*2) Model Initialization:* We propose a modified point initialization performed in 3DGS. 3D point initialization becomes a much higher influence on training in the few-input view case, and as such, requires an accurate initialization to guide the GS into a geometrically precise and photorealistic scene. We do not have COLMAP data to initialize the 3DGS and cannot rely on a potentially noisy depth sensor for initialization, especially in environments where vision is not sufficient, such as reflective surfaces. Thus, for each image  $I_k$  in our few input view list, we backproject the depth of each GPIS depth image  $\mu_{GPIS,k}$  into the world frame with the robot camera poses, and use the combined point cloud to initialize a 3DGS. We also initialize on only touch to highlight its usefulness for an entire scene. Finally, we observe that in the case of 3DGS, convergence is reached quickly, and thus train all 3DGS models in our ablations and real-world experiments for 15000 steps for a fair comparison.

## IV. RESULTS

### A. Experimental Setup

Touch-GS is implemented in both simulated and real-world experimental setups. The simulated evaluation assesses the performance of various ablations in Blender. We obtain our camera poses from Blender and use a point cloud

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	D-MSE $\downarrow$	D-MSE-O $\downarrow$
3DGS w/o Depth	15.78	0.53	0.52	23.04	8.40
<b>Sparse-Depth [12]</b>	<b>18.00</b>	<b>0.54</b>	<b>0.47</b>	<b>13.79</b>	<b>0.92</b>
<b>Dense-Depth [11]</b>	<b>18.21</b>	<b>0.55</b>	<b>0.46</b>	<b>5.77</b>	<b>0.82</b>
Raw Dense-Depth	17.37	0.55	0.45	9.38	7.48
Touch-Aligned Vision	17.99	0.56	0.46	6.39	0.20
Ours Touch Only	14.32	0.54	0.57	31.07	0.46
Touch w/o Initialization	17.02	0.54	0.48	21.70	0.16
Ours w/o Initialization, Uncertainty	18.11	0.56	0.46	6.57	0.14
<b>Ours w/o Uncertainty</b>	<b>19.19</b>	<b>0.60</b>	<b>0.42</b>	<b>5.25</b>	<b>0.016</b>
<b>Ours</b>	<b>19.20</b>	<b>0.60</b>	<b>0.42</b>	<b>5.29</b>	<b>0.016</b>
<i>3DGS w/ GT Depth</i>	<i>20.57</i>	<i>0.63</i>	<i>0.37</i>	<i>3.29</i>	<i>0.008</i>

TABLE I: Blender simulated scene ablations with 3DGS

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	D-MSE-O $\downarrow$
Nerfacto	12.57	0.310	0.77	0.85
DS-NeRF	17.16	0.58	0.46	99.17
Dense Depth	18.06	0.56	0.46	1.68
<b>Our Method</b>	<b>18.37</b>	<b>0.56</b>	<b>0.44</b>	<b>0.60</b>

TABLE II: Nerfacto Results

Object	Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
Real-world	3DGS	10.30	0.41	0.61
	Dense-Depth	11.40	0.45	0.55
	No Uncertainty	11.71	0.47	0.53
Bunny	3DGS	15.39	0.65	0.42
	Dense-Depth	<b>15.75</b>	<b>0.66</b>	<b>0.42</b>
	<b>Our Method</b>	15.51	<b>0.66</b>	<b>0.42</b>
Mirror	3DGS	14.43	0.55	0.42
	Dense-Depth	<b>14.85</b>	<b>0.57</b>	<b>0.41</b>
	<b>Our Method</b>	14.71	<b>0.57</b>	<b>0.41</b>
Prism	3DGS	14.43	0.55	0.42
	Dense-Depth	<b>14.85</b>	<b>0.57</b>	<b>0.41</b>
	<b>Our Method</b>	14.71	<b>0.57</b>	<b>0.41</b>

TABLE III: Real-world experiment with multiple objects

sampled from the GT geometry. To generate sparse points for alignment, we take a small percent (under 1%) of the GT depth data and perturb it with noise that quadratically increases with distance. We train our scene on 5 equally spaced views around the bunny.

The real-world evaluation assesses the use of Touch-GS and uncertainty in real-world conditions. Touch-GS requires accurate poses for the camera, depth camera, and touch sensor. We utilized the RGB and depth cameras installed in the Kinova™ Gen 3 and attached the DenseTact 2.0 [5] at the end-effector of the robot arm as shown in Fig. 1. We collect between 8-151 RGB-D images and between 150-500 touches per object. From the reconstructed depth image of the sensor, we determined the contact point by thresholding the estimated point cloud. The final touch data includes between 5,000 to 20,000 points per touch.

### B. Simulation Evaluation

We perform an extensive ablation of our method, which includes ablations with GS and a method called Nerfacto, which is the default NeRF method used in Nerfstudio [27]. Our first set of ablations builds up our method from 3DGS. Our second set of ablations performs an analysis of our method on Nerfacto, which is Nerfstudio’s default, volumetric method. Our ablations in GS are listed in Table I for brevity. We reimplement [12] and [11] to the best of our ability; the former method’s code was not written for

3DGS, and the latter method’s code has not been released. We also tune both methods’ depth loss weight to best assess their performance. Finally, we train 3DGS with ground truth depth provided by Blender (and initializing from ground truth bunny points), providing a strong upper bound for the quality of a model trained with perfect depth. Each method is run 10 times and we report the mean of the standard NeRF metrics (PSNR, SSIM, LPIPS) and ground truth depth and object depth mean square error (MSE), in meters, which we call D-MSE and D-MSE-O respectively.

In Table I, 3DGS performs the worst (besides D-MSE), as a lack of depth leads to unrecognizable depths and the inability to localize the bunny. Using sparse depth alone significantly improves visual and depth metrics; indicating how even some form of depth priors improves NeRF generalization. [11] outperforms sparse depth in all metrics, most notably the scene ground truth depth MSE, shown in Fig. 7. Aligning vision to touch leads to slightly worse depth loss but better object depth loss.

The need for alignment is shown in Raw Dense-Depth, where the NeRF performs better than pure GS, but the lack of scale and offset is most noticeable in the poor object ground truth depth MSE. Using only touch improves visual quality without initialization and has an even higher D-MSE-O than when GPIS initialization is performed for just touch. This is due to the model correctly learning the 3D geometry of the bunny but being unable to localize it without background depth. When we fuse both vision and touch without initialization, the visual quality is comparable to [11] but with a higher depth loss and lower object ground truth depth loss. Once the GPIS point initialization is included, the visual and geometric quality improves dramatically; there is  $< 2\text{cm}$  depth object MSE. This is seen clearly in the first column of Fig. 7 in our method, where the bunny geometry is extremely sharp. It is apparent that touch and vision complement each other: vision improves the background which localizes a touched object, and touch improves the background rendering. In Table II, our method is the best in all metrics for Nerfacto, demonstrating the value of touch.

### C. Real world Evaluation

For the real-world evaluation, we leave all tunable parameters of the GPIS fixed with the exception of  $\rho$  in (9), which is adjusted due to the differing length scales of our four scenes. In the vision depth and uncertainty maps, we only update the

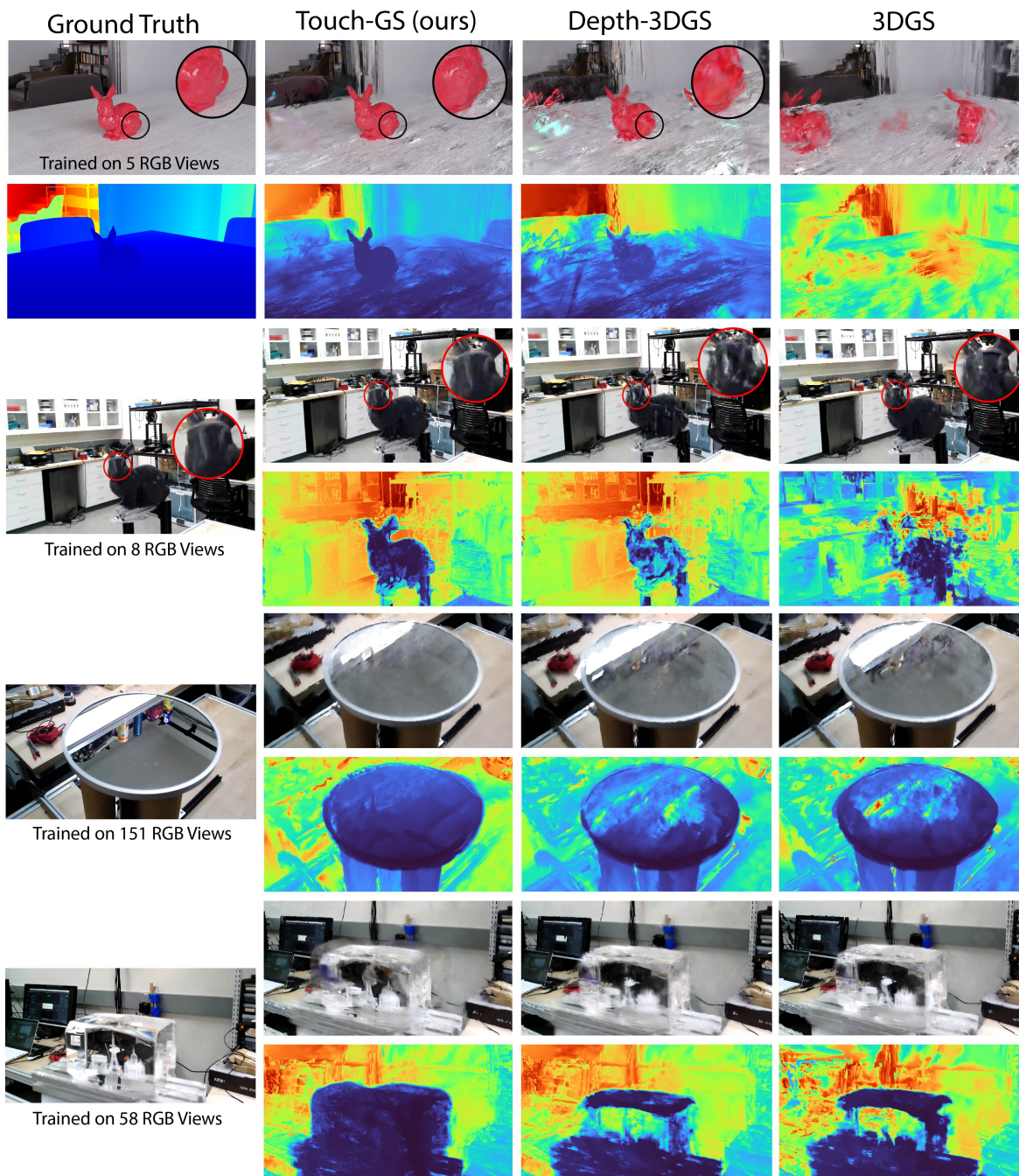


Fig. 7: Here we show comparisons of our method to the baselines and ground truth across 1 synthetic and 3 real scenes.

constant added to vision uncertainty, and we also filter out differences between ZoeDepth and the GPIS larger than 3 meters when we compute the touch-aligned vision map. In training the NeRF, we update the depth and/or uncertainty weight for each scene. We report mean PSNR, SSIM, and LPIPs with 3 trials for each method.

In the real-world bunny example, we train a NeRF on 8 input views and test on 40. Without any depth priors, 3DGS struggles, failing to properly render the ears of the bunny and background. The baseline from [11] outperforms 3DGS in all visual qualities with noticeably better geometries; however, it fails to completely render the bunny. Our method with

uncertainty is able to both render the background and bunny with clear geometries; rendering the tail of the bunny which was seen in touch and never predicted by vision.

Method	Chamfer ↓	Hausdorff ↓
3DGS	0.037	0.12
Dense-Depth	0.027	0.14
<b>Our Method</b>	<b>0.023</b>	<b>0.094</b>

TABLE IV: Real-world Bunny Object Geometric Reconstruction Quality.

We report the average shape dissimilarity (Chamfer distance) and maximal shape dissimilarity (Hausdorff distance)

between the sensed real-world bunny and a ground truth bunny mesh in Table IV. We compute a transformation, which we refine with Chamfer distance, to best align each method’s point cloud to ground truth for a most fair comparison. While Depth-3DGS has better overall shape reconstruction than 3DGS, it suffers from larger outliers, which is due to ZoeDepth incorrectly localizing the bunny. Across all metrics, our approach significantly outperforms 3DGS and Depth-3DGS, highlighting the importance of touch.

The mirror and prism examples serve as a challenging scene for vision. Even with many input views (151 for the mirror and 58 for the prism), the rendered depths from both 3DGS and vision-depth-supervised 3DGS are full of holes. The features shown in the mirror present a challenge for 3DGS and depth GS, which view the reflected features as objects with depth. In the other methods, the prism is transparent. Our method renders a flat, geometry respecting, surfaces on the mirror and prism, and while the visual metrics of [11] are better than our method, it is because the high depth weights given to touch make it harder to render the reflections and see-through objects, which we plan to address in future work. But the accurate geometries of these objects in our method present an important first step for robotic manipulators to operate in these environments.

## V. CONCLUSION

In this work, we have shown how Touch-GS fuses visual-tactile data to produce 3DGS scenes. We demonstrate qualitative and quantitative improvements over our baselines of standard and depth-supervised 3DGS. Robotic systems of the future need to be able to fuse touch and vision to interact with their environment. Our work addresses the fundamental balance between coarse RGB data fused with fine tactile data. In the future, we hope to expand this representation to be dynamic, including both deformability and frictional properties of the object in question to create a true digital twin.

## REFERENCES

- [1] 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,” *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.
- [2] T. Chen, P. Culbertson, and M. Schwager, “Catnips: Collision avoidance through neural implicit probabilistic scenes,” 2023.
- [3] S. Suresh, H. Qi, T. Wu, T. Fan, L. Pineda, M. Lambeta, J. Malik, M. Kalakrishnan, R. Calandra, M. Kaess, J. Ortiz, and M. Mukadam, “Neural feels with neural fields: Visuo-tactile perception for in-hand manipulation,” 2023.
- [4] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” *ACM Transactions on Graphics (ToG)*, vol. 42, no. 4, pp. 1–14, 2023.
- [5] W. K. Do, B. Jurewicz, and M. Kennedy, “Densetact 2.0: Optical tactile sensor for shape and force reconstruction,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 12 549–12 555.
- [6] E. Donlon, S. Dong, M. Liu, J. Li, E. Adelson, and A. Rodriguez, “Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger,” 2018.
- [7] W. K. Do, B. Aumann, C. Chungyoun, and M. Kennedy, “Inter-finger small object manipulation with densetact optical tactile sensor,” *IEEE Robotics and Automation Letters*, 2023.
- [8] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik, “General in-hand object rotation with vision and touch,” 2023.
- [9] M. Comi, Y. Lin, A. Church, A. Tonioni, L. Aitchison, and N. F. Lepora, “Touchsd: A deepsd approach for 3d shape reconstruction using vision-based tactile sensing,” 2023.
- [10] J. Zhao, M. Bauza, and E. H. Adelson, “Fingerslam: Closed-loop unknown object localization and reconstruction from visuo-tactile feedback,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8033–8039.
- [11] J. Chung, J. Oh, and K. M. Lee, “Depth-regularized optimization for 3d gaussian splatting in few-shot images,” 2024.
- [12] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, “Depth-supervised nerf: Fewer views and faster training for free,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 882–12 891.
- [13] B. Roessle, J. T. Barron, B. Mildenhall, P. P. Srinivasan, and M. Nießner, “Dense depth priors for neural radiance fields from sparse input views,” 2022.
- [14] Y. Fu, S. Liu, A. Kulkarni, J. Kautz, A. A. Efros, and X. Wang, “Colmap-free 3d gaussian splatting,” 2023.
- [15] W. Xu, Z. Yu, H. Xue, R. Ye, S. Yao, and C. Lu, “Visual-tactile sensing for in-hand object reconstruction,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8803–8812.
- [16] C. Pan, M. Lepert, S. Yuan, R. Antonova, and J. Bohg, “In-hand manipulation of unknown objects with tactile sensing for insertion,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 8765–8771.
- [17] J. A. Solano-Castellanos, W. K. Do, and M. Kennedy III, “Embedded object detection and mapping in soft materials using optical tactile sensing,” *arXiv preprint arXiv:2308.11087*, 2023.
- [18] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “Deepsdf: Learning continuous signed distance functions for shape representation,” 2019.
- [19] O. Williams and A. Fitzgibbon, “Gaussian process implicit surfaces,” in *Gaussian Processes in Practice*, April 2007. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/gaussian-process-implicit-surfaces-2/>
- [20] S. Dragiev, M. Toussaint, and M. Gienger, “Gaussian process implicit surfaces for shape estimation and grasping,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2845–2850.
- [21] Y. Chen, A. E. Tekden, M. P. Deisenroth, and Y. Bekiroglu, “Sliding touch-based exploration for modeling unknown object shape with multi-fingered hands,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 8943–8950.
- [22] F. Yang, C. Feng, Z. Chen, H. Park, D. Wang, Y. Dou, Z. Zeng, X. Chen, R. Gangopadhyay, A. Owens *et al.*, “Binding touch to everything: Learning unified multimodal tactile representations,” *arXiv preprint arXiv:2401.18084*, 2024.
- [23] L. Fu, G. Datta, H. Huang, W. C.-H. Panitch, J. Drake, J. Ortiz, M. Mukadam, M. Lambeta, R. Calandra, and K. Goldberg, “A touch, vision, and language dataset for multimodal alignment,” *arXiv preprint arXiv:2402.13232*, 2024.
- [24] E. Smith, D. Meger, L. Pineda, R. Calandra, J. Malik, A. Romero Soriano, and M. Drodzdzal, “Active 3d shape reconstruction from vision and touch,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 064–16 078, 2021.
- [25] D. Watkins-Valls, J. Varley, and P. Allen, “Multi-modal geometric learning for grasping and manipulation,” in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 7339–7345.
- [26] E. Smith, R. Calandra, A. Romero, G. Gkioxari, D. Meger, J. Malik, and M. Drodzdzal, “3d shape reconstruction from vision and touch,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 193–14 206, 2020.
- [27] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa, “Nerfstudio: A modular framework for neural radiance field development,” in *ACM SIGGRAPH 2023 Conference Proceedings*, ser. SIGGRAPH ’23, 2023.
- [28] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, “Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration,” in *Advances in Neural Information Processing Systems*, 2018.
- [29] S. F. Bhat, R. Birkel, D. Wofk, P. Wonka, and M. Müller, “Zoedepth: Zero-shot transfer by combining relative and metric depth,” 2023.