Adversarial Texture Optimization from RGB-D Scans

Jingwei Huang\textsuperscript{1,3}, Justus Thies\textsuperscript{2}, Angela Dai\textsuperscript{2}, Abhijit Kundu\textsuperscript{3}, Chiyu “Max” Jiang\textsuperscript{3,4}, Leonidas Guibas\textsuperscript{1}, Matthias Nießner\textsuperscript{2}, and Thomas Funkhouser\textsuperscript{3}

\textsuperscript{1}Stanford University \hspace{1cm} \textsuperscript{2}Technical University of Munich \hspace{1cm} \textsuperscript{3}Google Research \hspace{1cm} \textsuperscript{4}UC Berkeley

Abstract

Realistic color texture generation is an important step in RGB-D surface reconstruction, but remains challenging in practice due to inaccuracies in reconstructed geometry, misaligned camera poses, and view-dependent imaging artifacts. In this work, we present a novel approach for color texture generation using a conditional adversarial loss obtained from weakly-supervised views. Specifically, we propose an approach to produce photorealistic textures for approximate surfaces, even from misaligned images, by learning an objective function that is robust to these errors. The key idea of our approach is to learn a patch-based conditional discriminator which guides the texture optimization to be tolerant to misalignments. Our discriminator takes a synthesized view and a real image, and evaluates whether the synthesized one is realistic, under a broadened definition of realism. We train the discriminator by providing as ‘real’ examples pairs of input views and their misaligned versions – so that the learned adversarial loss will tolerate errors from the scans. Experiments on synthetic and real data under quantitative or qualitative evaluation demonstrate the advantage of our approach in comparison to state of the art (see Figure 1, right). Our code is publicly available\textsuperscript{1} with video demonstration\textsuperscript{2}.

1. Introduction

The wide availability of consumer range cameras has spurred extensive research in geometric reconstruction of real-world objects and scenes, with state-of-the-art 3D reconstruction approaches now providing robust camera tracking and 3D surface reconstruction\cite{22, 19, 33, 9}. However, producing photorealistic models of real-world environments requires not only geometric reconstruction but also high-quality color texturing. Unfortunately, due to noisy input data, poorly estimated surface geometry, misaligned camera poses, unmodeled optical distortions, and view-dependent lighting effects, aggregating multiple real-world images into high-quality, realistic surface textures is still a challenging problem. In order to overcome these problems, various approaches have been developed to optimize color textures using models to adjust camera poses\cite{36, 15}, distort images\cite{4, 15, 36}, and balance colors\cite{15, 36}. However, these prior approaches are not expressive enough and/or their optimization algorithms are not robust enough to handle the complex distortions and misalignments commonly found in scans with commodity cameras – and therefore they fail to produce high-quality results for typical scans, as shown in the results from Zhou and Koltun\cite{36} in Figure 1.

To address these issues, we propose a flexible texture optimization framework based on a learned metric that is robust to common scanning errors (right side of Figure 1).

\textsuperscript{1}https://github.com/hjwdzh/AdversarialTexture
\textsuperscript{2}https://youtu.be/SZx1RnOESeK

Figure 1. Our goal is to reconstruct high-quality textures from an RGB-D scan. Unlike traditional methods which optimize for a parametric color map to reduce misalignment error (Zhou and Koltun\cite{36}), we learn a misalignment-tolerant discriminator, producing sharper textures.
The key idea behind our approach is to account for misalignments in a learned objective function of the texture optimization. Rather than using a traditional object function, like L1 or L2, we learn a new objective function (adversarial loss) that is robust to the types of misalignment present in the input data. This novel approach eliminates the need for hand-crafted parametric models for fixing the camera parameters [36, 15], image mapping [36, 4] or color balance [15]. Instead, we jointly solve texture with an adversarial evaluation metric to tolerate the errors.

Inspired by the success of adversarial networks in image synthesis [14], we propose to use a learned conditional discriminator to serve our objective function, and jointly optimize the color texture of a reconstructed surface with this discriminator. The condition is a captured image $I_A$ from the source view $V_A$, and the query is either (i) “real:” a second captured image $I_B$ (from an auxiliary view $V_B$) projected onto the surface and then rendered back to $V_A$, or (ii) “fake:” an image of the optimized synthetic texture rendered to view $V_A$. By optimizing the surface texture while jointly training this conditional discriminator, we aim to produce a texture that is indistinguishable from reprojections of captured images from all other views. During the optimization, the discriminator learns invariance to the misalignments and distortions present in the input dataset, while recognizing synthetic artifacts that do not appear in the real images (local blurs and seams). Therefore, the textures optimized to fool the discriminator (ours in Figure 1) appear more realistic than in previous approaches.

Our experiments show that this adversarial optimization framework produces notably improved performance compared to state-of-the-art methods, both quantitatively on synthetic data and qualitatively on real data. Moreover, since it tolerates gross misalignments, we are able to generate realistic textures on CAD models which have been only roughly aligned to 3D scans, in spite of large mismatches in surface geometry. This opens up the potential to produce CAD models with realistic textures for content creation.

2. Related Work

RGB-D scanning has a rich history, and many approaches have been proposed for color texture generation.

View aggregation. Common texture generation methods [19, 36] average projected input images to generate textures. To reduce blurriness artifacts, some approaches select a single or a few candidate views for each region [11]. Others formulate a multi-label selection energy minimization problem to minimize seam artifacts [21, 27, 30, 32, 15]. For instance, [15] aims at selecting the best view for each region to balance the visual sharpness and color consistency of boundaries between neighboring regions with different views selected, which is modeled as a multi-label graph-cut problem [5]. Our method does not explicitly define the aggregation method, but implicitly aggregates colors from different views based on a learned adversarial metric.

Parametric color optimization. Several approaches have been proposed to improve the mapping of input images to textures with parametric models, leveraging both human supervision [12, 23, 24, 34], as well as automatic optimization [3, 25]. Zhou et al. [36] propose to optimize a parametric model comprising camera poses and non-rigid grid deformations of input images to minimize an L2 color consistency metric. While these methods are able to fix small misalignments, their deformation models are often not expressive enough to handle many real-world distortions, particularly those due to largely approximate surface geometry. In contrast to a hand-crafted deformation model, we learn a distortion-tolerant adversarial loss.

Patch-based color optimization. Patch-based image synthesis strategies have been proposed for color texture optimization [4]. Rather than non-rigid image warping, they re-synthesize the input image with the nearest patch [26] to handle misalignments. However, general misalignment cannot be accurately modeled by translating patches, and the L2 loss is not robust to color, lighting or sharpness differences. Our method optimizes the discriminator to cover all these problems without requiring explicit re-synthesis.

Neural textures. Recently, neural rendering approaches have been proposed to synthesize a feature map on a surface that can be interpreted by a deep network to produce novel image views. For instance, [29] stores appearance information as high-dimensional features in a neural texture map associated with the coarse geometry proxy and decodes to color when projected to novel views. [28] stores the appearance information as high-dimensional features in volumes, and [2] uses features stored with points. These methods rely on the representation power of generative networks at rendering times to obtain novel viewpoints, which limits their applicability in standard graphics pipelines.
re-projection can be achieved by projecting the color image from any other view to this view, as shown in Figure 3. Such re-projection can be achieved by projecting the color image onto the surface and then rendering back to another view. Unlike the simple 2D example, it is highly possible that there is no texture solution so that each local patch perfectly matches the one view from the input images, given camera and geometry error. However, the proposed approach is expected to push those inconsistencies to the smooth textured regions to hide any artifacts that can be easily identified by the discriminator, and thereby producing locally consistent realistic texture solution.

For each optimization iteration, we randomly select two input images, $I_A$ (source image) and $I_B$ (auxiliary image) with corresponding camera poses $V_A$ and $V_B$. The conditioning is $I_A$ from the viewpoint $V_A$, and the ‘real’ image is $I_B$ projected to the scan geometry and rendered from $V_A$, while the ‘fake’ image is the synthesized texture rendered from $V_A$. We alternating optimize the texture and discriminator. During texture optimization, we adjust the texture pixel colors to maximize the adversarial loss such that it looks more realistic under the discriminator scoring. During discriminator optimization, we minimize the adversarial loss such that it better classifies real and fake examples. We linearly combine adversarial loss with an L1 loss that decays exponentially as the optimization proceeds, which helps the optimizer find a good initial texture solution.

Network Architecture: Our framework is adopted from the PatchGAN discriminator architecture proposed by Isola et al. [18]. We choose that framework because it is designed to produce local details that look as realistic as a given set of input images. We use three convolutional layers, resulting in a patch size of $70 \times 70$, which we find suitable for our input images of resolution $640 \times 480$. We apply a PatchGAN to evaluate local $70 \times 70$ patches of images rather than the entire image. Patches are selected for discriminator training.
we additionally add an L1 loss to the texture optimization if more than half of the patch is not occluded. Thus, patches used for training have sufficient overlap. Unlike the original, we remove all batch normalization layers and feed a single view example for each optimization iteration, which we empirically found to improve performance. Conditioned on the input view, we ask the discriminator to evaluate the residual of the synthesized example subtracted by the condition input. Finally, since we focus on evaluating foreground regions (pixels corresponding to input geometry), we remove the loss terms for regions where background comprises more than 90% of the receptive field.

### 3.2. Texture Optimization

To retrieve a texture, we jointly optimize the texture and the misalignment-tolerant metric. Inspired by the adversarial loss used in Pix2Pix [18], we express our view-conditioned adversarial loss as:

\[
\mathcal{L}_v(T, D) = E_{x,y}(\log D(x, y)) + E_{x,M_t}(\log(1 - D(x, M_t(T)))).
\]

where \( T \) and \( D \) represent the target texture image and the discriminator parameters we are optimizing for. \( x \) is the condition, a reprojected color image from the input sequence of captured images. \( M_t \) is the fixed texture-to-image mapping given the camera pose associated with \( x \). Here, a real example is an image \( y \) re-projected to the view of \( x \). We optimize \( D \) with the objective to correctly identify real examples, misaligned real imagery, and fake examples rendered from the texture as \( M_t(T) \). Simultaneously, we optimize the texture \( T \) such that it is difficult to be identified as fake when mapped to view of \( x \).

Since the adversarial loss alone can be difficult to train, we additionally add an L1 loss to the texture optimization to provide initial guidance for the optimization:

\[
\mathcal{L}_{L1}(T) = E_{x,y,M_t}||y - M_t(T)||_1.
\]

Our objective texture solution is:

\[
T^* = \arg \min_T \max_D \mathcal{L}_v(T, D) + \lambda \mathcal{L}_{L1}(T).
\]

During training, we initialize all pixels in texture image to zero and \( \lambda = 10 \). The high \( \lambda \) allows the L1 loss to provide an initial texture, and for every 1000 steps we exponentially decay the lambda by a factor of 0.8. We optimize in alternating fashion for each optimization step, using the Adam optimizer for both the texture and discriminator with learning rates \( 10^{-3} \) and \( 10^{-4} \) respectively. For each object or scene, we optimize for 50000 steps to finalize our texture.

### 3.3. Differentiable Rendering and Projection

To enable the optimization of the RGB texture of a 3D model, we leverage a differentiable rendering to generate synthesized ‘fake’ views. We pre-compute a view-to-texture mapping using pyRender [17], and can then implement the rendering with a differentiable bilinear sampling.

To create the misaligned ‘real’ images \( \{I_b \} \) seen from \( V_A \), we compute a reprojection; note that here we do not need to maintain gradient information. For each pixel \( p_A \) in the source image, we need to determine the corresponding pixel \( p_B \) in the auxiliary image, so that a bilinear sampling can be applied to warp image from the \( V_B \) to \( V_A \). Specifically, for \( p_A \) with depth value \( d_A \) from the source depth map, we can determine its 3D location in the source view’s space as \( p_A = d_A K^{-1} p_A \) where \( K \) is the intrinsic camera matrix. Suppose the transformations from the camera to the world space for the source and the auxiliary views are given as \( T_A \) and \( T_B \), the corresponding 3D and pixel location in the auxiliary view are \( p_B = T_B^T A p_A \) and \( p_B = K p_B \). The pixel is visible in the auxiliary view if \( p_B \) is in the scope of the image and the difference between z-dimension of \( p_B \) and \( d_A \) from the auxiliary depth map is \( < \theta_z \). We use \( \theta_z = 0.1 \) meters for scenes \( \theta_z = 0.03 \) for object level scanning.

### 4. Experiments

#### Evaluation Metric

For evaluation, we adopt several different metrics to measure the quality of the generated texture compared to the ground truth. First, we propose the nearest patch loss as an indicator of how close the patch appearance of the texture is to the ground truth. Specifically, for each pixel \( u \) we extract a \( 7 \times 7 \) patch centered around it in the generated texture and find the L2 distance \( d(u) \) between it and the nearest neighbor patch in the ground truth texture. We define the nearest patch loss as the average of all \( d(u) \). Second, we adopt the perceptual metric [35] to evaluate perceptual quality. Finally, we propose to measure the difference between generated textures and ground
Synthetic 2D Example  We first verify the effectiveness of our method with a synthesized 2D example. We aim to optimize for a 2D image, given input observations with 2D micro-translation errors. We use an image resolution of $512 \times 512$ and translation error $\in [-16, 16]^2$. During texture optimization, we randomly select one observation as the source and another observation as the auxiliary, and optimize the target image to be more realistic under the current discriminator. Figure 5 shows the resulting image optimized with our approach in comparison to a naive L1 loss.

Visually, our optimized image is sharper and perceptually closer to the ground truth while an L1 loss results in blurry effect from aggregating multiple misaligned observations. In this simple setting, we evaluate the exact patch loss for each pixel quantitatively as the L2 distance of patches centered at this pixel between the generated image and the same one in the ground truth. The exact overall exact patch loss is the L2 norm of exact patch losses for all pixels. We additionally evaluate the nearest patch loss. Optimization with the L1 loss achieves 10.7 exact patch loss while ours is 11.3. However, we achieve 1.53 nearest patch loss, which is smaller than L1 as 7.33. This suggests that our method prefers realistic misalignment to blur. We successfully derive an image where every local patch is nearly identical to a misaligned version of the patch in the ground truth image.

Synthetic 3D Example  In order to quantitatively evaluate our 3D texture generation, we create a synthetic dataset of 16 models randomly selected from ShapeNet [7] across different categories. These shapes typically contain sharp edges and self-occlusion boundaries, complexities reflecting those of real-world objects. Since we aim to address arbitrary texturing, we enrich the appearance of these shapes by using 16 random color images from the internet as texture images. To create virtual scans of the objects, we uniformly sample $> 900$ views on a unit hemisphere by subdividing an icosahedron, from which we render the textured geometry as observed color images. To simulate misalignment, we associate each rendered image with a slightly perturbed camera pose, and to simulate geometry errors, we apply random perturbations to the geometric model. We use a set of errors increasing from $n = 1$ to $n = 4.5$, and refer to the supplemental material for additional detail regarding generating camera and geometry perturbations.

In Table 1, we study the effect of varying camera and geometry errors in this synthetic 3D setting. We report evaluation metrics for our approach as well as several state-of-the-art texture optimization methods, including methods based on an L1 loss and texturing using sharpest frame selection [31]. Our approach outperforms all other methods, as it avoids blurring effects often seen with L1 and ColorMap [36], and it avoids seams and over-sharpeness introduced by methods relying on sharpness selection (3DLite [15] and sharpest frame selection). VGG [20] aggregates views by blending deep features, which is insufficient for handling misalignment artifacts. Two example scenes with increasing errors in camera and geometry are shown in Figure 6.

We additionally study the behavior of all methods in this experiment using the perceptual metric [35] in Figure 8. Although the performance drops for all methods with the increase of camera/geometry errors, our approach maintains the best perceptual quality as the errors increase. Figure 7 shows a qualitative comparison; our approach maintains a sharp result while ColorMap produces increasingly blurriness as the error increases.

**Table 1.** Evaluation of different methods on our 3D synthetic dataset averaged across different levels of camera pose and geometry errors.

<table>
<thead>
<tr>
<th>Camera / Geometry</th>
<th>Patch</th>
<th>Perceptual</th>
<th>Gradient</th>
<th>Sharp</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>6.01 / 6.65</td>
<td>0.256 / 0.261</td>
<td>0.10 / 0.005</td>
<td>0.041 / 0.037</td>
</tr>
<tr>
<td>ColorMap</td>
<td>6.34 / 6.49</td>
<td>0.252 / 0.287</td>
<td>0.14 / 0.061</td>
<td>0.056 / 0.036</td>
</tr>
<tr>
<td>Sharpest</td>
<td>5.96 / 5.92</td>
<td>0.236 / 0.281</td>
<td>0.06 / 0.058</td>
<td>0.027 / 0.025</td>
</tr>
<tr>
<td>3DLite</td>
<td>6.06 / 6.06</td>
<td>0.385 / 0.407</td>
<td>0.06 / 0.036</td>
<td>0.087 / 0.069</td>
</tr>
<tr>
<td>VGG</td>
<td>6.97 / 7.05</td>
<td>0.390 / 0.388</td>
<td>0.15 / 0.084</td>
<td>0.061 / 0.096</td>
</tr>
<tr>
<td>Ours</td>
<td>5.81 / 5.28</td>
<td>0.193 / 0.181</td>
<td>0.03 / 0.028</td>
<td>0.022 / 0.018</td>
</tr>
</tbody>
</table>

**Alternative Discriminators?** We analyze the design choices for our misalignment-tolerant conditional discriminator in Figure 9. Removing the auxiliary view (b) and thus relying only on the source view to provide ‘real’ examples to the discriminator (similar to pix2pix [18]) renders the
metric unable to handle misalignments. We also evaluate a general discriminator that classifies whether a generated patch is real or fake among entire input view sets without any condition (c), resulting in ambiguity as to where real patches come from. Our conditional discriminator leverag-
Table 2. Mean perceptual loss comparing the input images and rendered textures from different methods. Our method achieves best performance in the real and CAD datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Object</th>
<th>ScanNet</th>
<th>CAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>0.197</td>
<td>0.470</td>
<td>0.199</td>
</tr>
<tr>
<td>ColorMap</td>
<td>0.186</td>
<td>0.461</td>
<td>0.234</td>
</tr>
<tr>
<td>Sharpest</td>
<td>0.223</td>
<td>0.510</td>
<td>0.260</td>
</tr>
<tr>
<td>3DLite</td>
<td>0.185</td>
<td>0.445</td>
<td>0.238</td>
</tr>
<tr>
<td>VGG</td>
<td>0.272</td>
<td>0.534</td>
<td>0.289</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.175</strong></td>
<td><strong>0.395</strong></td>
<td><strong>0.176</strong></td>
</tr>
</tbody>
</table>

**Real Scene Scans** To demonstrate the capability of our approach to optimize texture on a larger scale, we run our algorithm on the ScanNet dataset [8], which provides RGB-D sequences and reconstructed geometry of indoor scenes. We evaluate our approach on scenes with ID ≤ 20 (∼2000 – 3000 frames per scan) and compare it with the existing state of the arts. Figure 11 (rows 5-9) and Table 2 (middle column) show qualitative and quantitative comparisons. Our method produces texturing most perceptually similar to the observed images; our misalignment-tolerant metrics aids in avoiding blur, increased sharpness, or excess noise produced by other methods due to camera and geometry errors in real-world scans.

**Real to CAD Models** Since our method can better handle errors from approximate surface geometry, it is possible to consider texturing CAD models using real-world images to attain realistic appearances. While large datasets of 3D CAD models are now available [7], they are often untextured or textured simplistically, resulting in notably different appearance from real-world objects. To test whether our method can be applied in this challenging scenario, we use our collected dataset of real object scans, retrieve similar CAD models from ShapeNet manifold [16], and rigidly align them to the scanned objects. We then replace the scanned geometry with the CAD model and then use the captured color images and estimated poses from the scan to optimize the CAD texture. Qualitative and quantitative evaluation of our approach in comparison to existing state-of-the-art methods are show in Figure 11 (rows 10-13) and Table 2 (right column), respectively. Our approach is able to handle both camera poses errors as well as the synthetic-real geometry differences to produce texturing perceptually very similar to observed imagery, whereas other methods suffer strong blur, noise, and seam artifacts under these errors.

**Perceptual Quality** Although we lack ground truth texturing for objects in real environments, we can compare the perceptual loss [35] of the rendered textured geometry from the corresponding viewpoint. We select 10 views uniformly distributed from the scanning video and render the textured model to compute the mean of the perceptual loss. Table 2 shows the performance of different methods on the object scans, scene scans and the CAD models; our method achieves the best performance in these three scenarios.

Additionally, we perform a user study to evaluate the quality of the texture, shown in Figure 10. Our user study comprised 63 participants who were asked to vote for the texture which produced a rendering closest to the input image. For some views, it can sometimes be difficult for users to differentiate between different methods when regions are largely uniform in color. Nevertheless, our method is still notably preferred over other texturing approaches. We provide additional comparisons with [32] and [13] in supplemental C and describe the influence of sparse views for training discriminators in supplemental D.

**Runtime.** On average, our released implementation takes 7.3 minutes per object and 33.4 minutes per scene on a single TITAN X GPU.

**5. Conclusion**

We have proposed a misalignment-tolerant metric for texture optimization of RGB-D scans, introducing a learned texturing objective function for maintaining robustness to misalignment errors in camera poses and geometry. We represent the learned function as a conditional discriminator trained with an adversarial loss where ‘real’ examples characterize various misalignment errors seen in the input data. This avoids explicit parametric modeling of scanning errors, and enables our optimization to produce texturing reflective of the realism. Our approach opens up the potential for texturing synthetic CAD models with real-world imagery. It also makes an important step towards creating digital content from real-world scans, towards democratized use, for instance in the context of AR and VR applications.

**Acknowledgements**

This work was supported by the ZD.B and ERC Starting Grant Scan2CAD (804724), NSF grants CHS-1528025 and IIS-1763268, a Vannevar Bush Faculty Fellowship, and grants from the Samsung GRO program and the Stanford SAIL Toyota Research Center.
Figure 11. Visual comparison on object scans, ScanNet [8] scans of scenes, and CAD models aligned with object scans. Due to misalignment errors in both camera pose and geometry, both L1 loss and ColorMap [36] produce blurry artifacts, sharpest selection and 3DLite [15] result in inconsistent regions or breaks in texture structure, and VGG [20] blends learned features resulting in structural artifacts and noise. Our misalignment-tolerant approach produces sharp and consistent textures.
References

[27] Sudipta N Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. Interactive 3d architectural...


