GANmouflage: 3D Object Nondetection with Texture Fields

Rui Guo[1,3*]   Jasmine Collins[2]   Oscar de Lima[1]   Andrew Owens[1]

University of Michigan[1]   UC Berkeley[2]   XMotors.ai[3]

(a) Camouflaged object	(b) Viewpoints of camouflaged object	(c) Object without camouflage

Figure 1. We learn to camouflage 3D objects within scenes. Given an object's shape, position, and a distribution of possible viewpoints it will be seen from, we estimate a texture field that will conceal it. We show example outputs from our model, with two viewpoints for each camouflaged object. Please see the videos on our webpage for more examples.

Abstract

We propose a method that learns to camouflage 3D objects within scenes. Given an object's shape and a distribution of viewpoints from which it will be seen, we estimate a texture that will make it difficult to detect. Successfully solving this task requires a model that can accurately reproduce textures from the scene, while simultaneously dealing with the highly conflicting constraints imposed by each viewpoint. We address these challenges with a model based on texture fields and adversarial learning. Our model learns to camouflage a variety of object shapes from randomly sampled locations and viewpoints within the input scene, and is the first to address the problem of hiding complex object shapes. Using a human visual search study, we find that our estimated textures conceal objects significantly better than previous methods.

1. Introduction

Using fur, feathers, spots, and stripes, camouflaged animals show a remarkable ability to stay hidden within their environment. These capabilities developed as part of an evolutionary arms race, with advances in camouflage leading to advances in visual perception, and vice versa.

Inspired by these challenges, previous work [34] proposed the *object nondetection* problem: to create an appearance for an object that makes it undetectable. Given an object's shape and a sample of photos from a scene, the goal is to produce a texture that hides the object from every viewpoint that it is likely to be observed from. This problem has applications in hiding unsightly objects, such as utility boxes [7], solar panels [30, 50], and radio towers, and in concealing objects from humans or animals, such as surveillance cameras and hunting platforms. Moreover, since camouflage models must ultimately thwart highly effective visual systems, they may provide a better scientific understanding of the cues that these systems use. Animal camouflage, for instance, has developed strategies for avoiding perceptual grouping and boundary detection cues [31, 53].

A successful learning-based camouflage system, likewise, must gain an understanding of these cues in order to thwart them.

Previous object nondetection methods are based on non-parametric texture synthesis. Although these methods have shown success in hiding cube-shaped objects, they can only directly "copy-and-paste" pixels that are directly occluded by the object, making it challenging to deal with complex backgrounds and non-planar geometry. While learning-based methods have the potential to address these shortcomings, they face a number of challenges. Since even tiny imperfections in synthesized textures can expose a hidden object, the method must also be capable of reproducing real-world textures with high fidelity. There is also no single texture that can perfectly conceal an object from all viewpoints at once. Choosing an effective camouflage requires 3D reasoning, and making trade-offs between different solutions. This is in contrast to the related problem of image inpainting, which can be posed straightforwardly as estimating masked image regions in large, unlabeled photo collections [35], and which lack the ability to deal with multi-view constraints.

We propose a model based on neural texture fields [23, 33, 36, 43] and adversarial training that addresses these challenges (Figure 2). The proposed architecture and learning procedure allow the model to exploit multi-view geometry, reproduce a scene's textures with high fidelity, and satisfy the highly conflicting constraints provided by the input images. During training, our model learns to conceal a variety of object shapes from randomly chosen 3D positions within a scene. It uses a conditional generative adversarial network (GAN) to learn to produce textures that are difficult to detect using pixel-aligned representations [57] with hypercolumns [20] to provide information from each view.

Through automated evaluation metrics and human perceptual studies, we find that our method significantly outperforms the previous state-of-the-art in hiding cuboid objects. We also demonstrate our method's flexibility by using it to camouflage a diverse set of complex shapes. These shapes introduce unique challenges, as each viewpoint observes a different set of points on the object surface. Finally, we show through ablations that the design of our texture model leads to significantly better results.

2. Related Work

Computational camouflage We take inspiration from early work by Reynolds [39] that formulated camouflage as part of an artificial life simulation, following Sims [46] and Dawkins [13]. In that work, a human "predator" interactively detects visual "prey" patterns that are generated using a genetic algorithm. While our model is also trained adversarially, we do so using a GAN, rather than with a human-in-the-loop. Later, Owens *et al.* [34] proposed the

problem of hiding a cuboid object at a specific location from multiple 3D viewpoints, and solved it using nonparametric texture synthesis. In contrast, our model learns through adversarial training to hide both cuboid and more complex objects. Bi *et al.* [5] proposed a patch-based synthesis method that they applied to the multi-view camouflage problem, and extended the method to spheres. However, this work was very preliminary: they only provide a qualitative result on a single scene (with no quantitative evaluation). Other work inserts difficult-to-see patterns into other images [10, 60].

Animal camouflage. Perhaps the most well-known camouflage strategy is *background matching*, whereby animals take on textures that blend into the background. However, animals also use a number of other strategies to conceal themselves, such as by masquerading as other objects [49], and using *disruptive coloration* to elude segmentation cues and to hide conspicuous body parts, such as eyes [12]. The object nondetection problem is motivated by animals that can dynamically change their appearance to match their surroundings, such as the octopus[1] [19]. Researchers have also begun using computational models to study animal camouflage. Troscianko *et al.* [54] used a genetic algorithm to camouflage synthetic bird eggs, and asked human subjects to detect them. Talas *et al.* [53] used a GAN to camouflage simple triangle-shaped representations of moths that were placed at random locations on synthetic tree bark. In both cases, the animal models are simplified and 2D, whereas our approach can handle complex 3D shapes.

Camouflaged object detection. Recent work has sought to detect camouflaged objects using object detectors [15, 29, 56] and motion cues [8, 28]. The focus of our work is generating camouflaged objects, rather than detecting them.

Adversarial examples. The object nondetection problem is related to adversarial examples [6, 18, 52], in that both problems involve deceiving a visual system (*e.g.*, by concealing an object or making it appear to be from a different class). Other work has generalized these examples to multiple viewpoints [2]. In contrast, the goal of the nondetection problem is to make objects that are concealed from a human visual system, rather than fool a classifier.

Texture fields. We take inspiration from recent work that uses implicit representations of functions to model the surface texture of objects [23, 33, 36, 43]. Oechsle *et al.* [33] learned to texture a given object using an implicit function, with image and shape encoders, and Saito *et al.* [43] learned a pixel-aligned implicit function for clothed humans. There are three key differences between our work and these methods. First, these methods aim to reconstruct textures from given images while our model predicts a texture that can conceal an object. Second, our model is conditioned on a 3D input scene with projective structure, rather than a set

---

[1]For a striking demonstration, see this video from Roger Hanlon: https://www.youtube.com/watch?v=JSq8nghQZqA

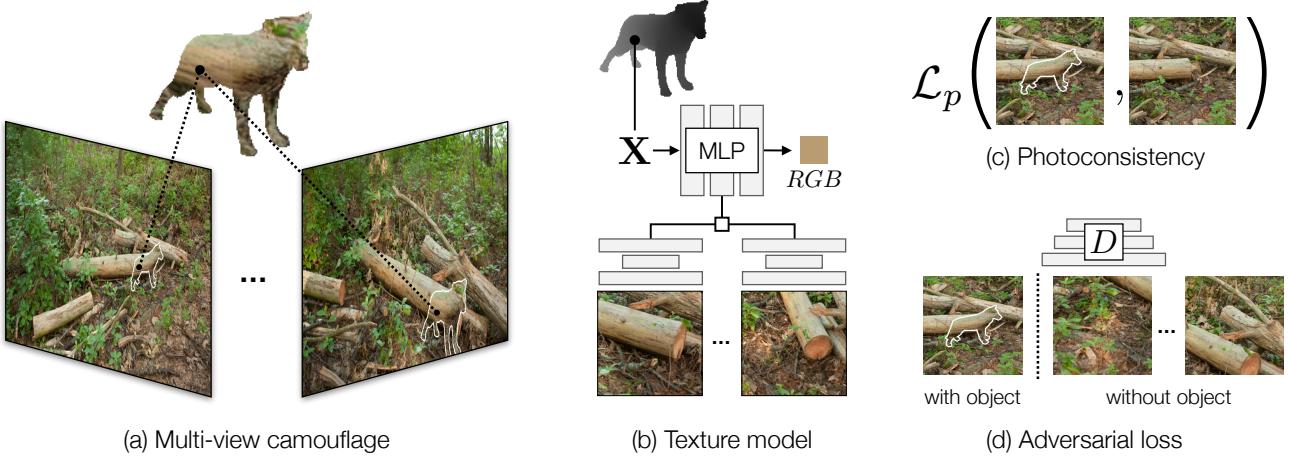(a) Multi-view camouflage   (b) Texture model   (c) Photoconsistency   (d) Adversarial loss

Figure 2. **Camouflage model.** (a) Our model creates a texture for a 3D object that conceals it from multiple viewpoints. (b) We generate a texture field that maps 3D points to colors. The network is conditioned on pixel-aligned features from training images. We train the model to create a texture that is (c) photoconsistent with the input views, as measured using a perceptual loss, and (d) difficult for a discriminator to distinguish from random background patches. For clarity, we show the camouflaged object's boundaries.

of images. Finally, the constraints provided by our images are mutually incompatible: there is no single way to texture a 3D object that satisfies all of the images. Other work has used implicit functions to represent 3D scenes for view synthesis [9, 32, 47, 57]. Sitzmann *et al.* [47] proposed an implicit 3D scene representation. Mildenhall *et al.* [32] proposed view-dependent neural radiance fields (NeRF). Recent work created image-conditional NeRFs [9, 57]. Like our method, they use networks with skip connections that exploit the projective geometry of the scene. However, their learned radiance field does not ensure multi-view consistency in color, since colors are conditioned on viewing directions of novel views.

Inpainting and texture synthesis. The camouflage problem is related to image inpainting [3, 4, 14, 21, 35, 59], in that both tasks involve creating a texture that matches a surrounding region. However, in contrast to the inpainting problem, there is no single solution that can completely satisfy the constraints provided by all of the images, and thus the task cannot be straightforwardly posed as a self-supervised data recovery problem [35]. Our work is also related to image-based texture synthesis [3, 14, 17] and 3D texture synthesis [23, 33, 36]. Since these techniques fill a hole in a single image, and cannot obtain geometrically-consistent constraints from multiple images, they *cannot be applied to our method without major modifications*. Nevertheless, we include an inpainting-based baseline in our evaluation by combining these methods with previous camouflage approaches.

3. Learning Multi-View Camouflage

Our goal is to create a texture for an object that camouflages it from all of the viewpoints that it is likely to be observed from. Following the formulation of Owens et

al. [34], our input is a 3D object mesh $\mathcal{S}$ at a fixed location in a scene, a sample of photos $I_1, I_2, ..., I_N$ from distribution $\mathcal{V}$, and their camera parameters $\mathbf{K}_j, \mathbf{R}_j, \mathbf{t}_j$. We desire a solution that camouflages the object from $\mathcal{V}$, using this sample. We are also provided with a ground plane $\mathbf{g}$, which the object has been placed on.

Also following [34], we consider the camouflage problem separately from the *display* problem of creating a real-world object. We assume that the object can be assigned arbitrary textures, and that there is only a single illumination condition. We note that shadows are independent of the object texture, and hence could be incorporated into this problem framework by inserting shadows into images (Sec. 4.5). Moreover, changes in the amount of lighting are likely to affect the object and background in a consistent way, producing a similar camouflage.

3.1. Texture Representation

We create a surface texture for the object that, on average, is difficult to detect when observed from viewpoints randomly sampled from $\mathcal{V}$. As in prior work [34], we render the object and synthetically insert it into the scene.

Similar to recent work on object texture synthesis [23, 33, 36], we represent our texture as continuous function in 3D space, using a *texture field*:

$$t_\theta : \mathbb{R}^3 \to \mathbb{R}^3. \tag{1}$$

This function maps a 3D point to an RGB color, and is parameterized using a multi-layer perceptron (MLP) with weights θ.

We condition our neural texture representation on input images, their projection matrices $\mathbf{P}_j = \mathbf{K}_j[\mathbf{R}_j|\mathbf{t}_j]$, and a 3D object shape $\mathcal{S}$. Our goal is to learn a *texturing function* that produces a texture field from an input scene:

$$G_\theta(\mathbf{x}; \{\mathbf{I}_j\}, \{\mathbf{P}_j\}, \mathcal{S}) \tag{2}$$

where $\mathbf{x}$ is a 3D *query point* on the object surface.

3.2. Camouflage Texture Model

To learn a camouflaged texture field (Eq. 2), we require a representation for the multi-view scene content, geometry, and texture field. We now describe these components in more detail. Our full model is shown in Figure 2.

Pixel-aligned image representation. In order to successfully hide an object, we need to reproduce the input image textures with high fidelity. For a given 3D point $\mathbf{x}_i$ on the object surface and an image $\mathbf{I}_j$, we compute an image feature $\mathbf{z}_i^{(j)}$ as follows.

We first compute convolutional features for $\mathbf{I}_j$ using a U-net [41] with a ResNet-18 [22] backbone at multiple resolutions. We extract image features $\mathbf{F}^{(j)} = E(\mathbf{I}_j)$ at full, $\frac{1}{4}$, and $\frac{1}{16}$ scales. At each pixel, we concatenate features for each scale together, producing a multiscale hypercolumn representation [20].

Instead of using a single feature vector to represent an entire input image, as is often done in neural texture models that create a texture from images [23, 33], we exploit the geometric structure of the multi-view camouflage problem. We extract *pixel-aligned* features $\mathbf{z}_i^{(j)}$ from each feature map $\mathbf{F}^{(j)}$, following work in neural radiance fields [57]. We compute the projection of a 3D point $\mathbf{x}_i$ in viewpoint $\mathbf{I}_j$:

$$\mathbf{u}_i^{(j)} = \pi^{(j)}(\mathbf{x}_i), \qquad (3)$$

where π is the projection function from object space to screen space of image $\mathbf{I}_j$. We then use bilinear interpolation to extract the feature vector $\mathbf{z}_i^{(j)} = \mathbf{F}^{(j)}(\mathbf{u}_i^{(j)})$ for each point i in each input image $\mathbf{I}_j$.

Perspective encoding. In addition to the image representation, we also condition our texture field on a *perspective encoding* that conveys the local geometry of the object surface and the multi-view setting. For each point $\mathbf{x}_i$ and image $\mathbf{I}_j$, we provide the network with the viewing direction $\mathbf{v}_i^{(j)}$ and surface normal $\mathbf{n}_i^{(j)}$. These can be computed as: $\mathbf{v}_i^{(j)} = \frac{\mathbf{K}_j^{-1}\mathbf{u}_i^{(j)}}{\|\mathbf{K}_j^{-1}\mathbf{u}_i^{(j)}\|_2}$ and $\mathbf{n}_i^{(j)} = \mathbf{R}_j\mathbf{n}_i$, where $\mathbf{u}_i^{(j)}$ is the point's projection (Eq. 3) in homogeneous coordinates, and $\mathbf{n}_i$ is the surface normal in object space. To obtain $\mathbf{n}_i$, we extract the normal of the face closet to $\mathbf{x}_i$.

We note that these perspective features come from the images that are used as *input* images to the texture field, rather than the camera viewing the texture, *i.e.* in contrast to neural scene representations [9, 32, 57], our textures are not viewpoint-dependent.

Texture field architecture. We use these features to define a texture field, an MLP that maps a 3D coordinate $\mathbf{x}_i$ to a color $\mathbf{c}_i$ (Eq. 1). It is conditioned on the set of image features for the N input images $\{\mathbf{z}_i^{(j)}\}$, as well as the sets

of perspective features $\{\mathbf{v}_i^{(j)}\}$ and $\{\mathbf{n}_i^{(j)}\}$:

$$\mathbf{c}_i = T(\gamma(\mathbf{x}_i); \{\mathbf{z}_i^{(j)}\}, \{\mathbf{v}_i^{(j)}\}, \{\mathbf{n}_i^{(j)}\}) \qquad (4)$$

where $\gamma(\cdot)$ is a positional encoding [32]. For this MLP, we use a similar architecture as Yu *et al.* [57]. The network is composed of several fully connected residual blocks and has two stages. In the first stage, which consists of 3 blocks, the vector from each input view is processed separately with shared weights. Mean pooling is then applied to create a unified representations from the views. In the second stage, another 3 residual blocks are used to predict the color for the input *query point*. Please see the supplementary material for more details.

Rendering. To render the object from a given viewpoint, following the strategy of Oechsle *et al.* [33], we determine which surface points are visible using the object's depth map, which we compute using PyTorch3D [38]. Given a pixel $\mathbf{u}_i$, we estimate a 3D surface point $\mathbf{x}_i$ in object space through inverse projection: $\mathbf{x}_i = d_i\mathbf{R}^T\mathbf{K}^{-1}\mathbf{u}_i - \mathbf{R}^T\mathbf{t}$, where d_i is the depth of pixel i, $\mathbf{K}, \mathbf{R}, \mathbf{t}$ are the view's camera parameters, and $\mathbf{u}_i$ is in homogeneous coordinates. We estimate the color for all visible points, and render the object by inserting the estimated pixel colors into a background image, $\mathbf{I}$. This results in a new image that contains the camouflaged object, $\hat{\mathbf{I}}$.

3.3. Learning to Camouflage

We require our camouflage model to generate textures that are photoconsistent with the input images, and that are not easily detectable by a learned discriminator. These two criteria lead us to define a loss function consisting of a photoconsistency term and adversarial loss term, which we optimize through a learning regime that learns to camouflage randomly augmented objects from random positions.

Photoconsistency. The photoconsistency loss measures how well the textured object, when projected into the input views, matches the background. We use a perceptual loss, $\mathcal{L}_{photo}$ [17, 26] that is computed as the normalized distance between activations for layers of a VGG-16 network [45] trained on ImageNet [42]:

$$\mathcal{L}_{photo} = \sum_{j \in J} \mathcal{L}_P(\hat{\mathbf{I}}_j, \mathbf{I}_j) = \sum_{j \in J, k \in L} \frac{1}{N_k} \|\phi_k(\hat{\mathbf{I}}_j) - \phi_k(\mathbf{I}_j)\|_1$$

$$(5)$$

where J is the set of view indices, L is the set of layers used in the loss, and ϕ_k are the activations of layer k, which has total dimension N_k. In practice, due to the large image size relative to the object, we use a crop centered around the object, rather than $\mathbf{I}_j$ itself (see Figure 2(c)).

Adversarial loss. To further improve the quality of generated textures, we also use an adversarial loss. Our model

Figure 3. **Multi-view results**. Multiple object views for selected scenes, camouflaged using our proposed model with four input views. The views shown here were held out and not provided to the network as input during training.

tries to hide the object, while a discriminator attempts to detect it from the scene. We randomly select *real* image crops y from each background image $\mathbf{I}_j$ and select *fake* crops $\hat{y}$ containing the camouflaged object from $\hat{\mathbf{I}}_j$. We use the standard GAN loss as our objective. To train the discriminator, D, we minimize:

$$\mathcal{L}_D = -\mathbb{E}_y[\log D(y)] - \mathbb{E}_{\hat{y}}[\log(1 - D(\hat{y}))] \qquad (6)$$

where the expectation is taken over patches randomly sampled from a training batch. We implement our discriminator using the fully convolutional architecture of Isola *et al.* [24]. Our texturing function, meanwhile, minimizes:

$$\mathcal{L}_{adv} = -\mathbb{E}_{\hat{y}}[\log D(\hat{y})] \qquad (7)$$

Self-supervised multi-view camouflage. We train our texturing function G (Eq. 2), which is fully defined by the image encoder E and the MLP T, by minimizing the combined losses:

$$\mathcal{L}_G = \mathcal{L}_{photo} + \lambda_{adv}\mathcal{L}_{adv} \qquad (8)$$

where λ_{adv} controls the importance of the two losses.

If we were to train the model with only the input object, the discriminator would easily overfit, and our model would fail to obtain a learning signal. Moreover, the resulting texturing model would only be specialized to a single input shape, and may not generalize to others. To address both of these issues, we provide additional supervision to the model by training it to camouflage randomly augmented shapes at random positions, and from random subsets of views.

We sample object positions on the ground plane $\mathbf{g}$, within a small radius proportional to the size of input object $\mathcal{S}$. We uniformly sample a position within the disk to determine the position for the object. In addition to randomly sampled locations, we also randomly scale the object within a range to add more diversity to training data. During training, we randomly select N_i input views and N_r rendering views without replacement from a pool of training images sampled from $\mathcal{V}$. We calculate $\mathcal{L}_{photo}$ on both N_i input views and N_r views while $\mathcal{L}_{adv}$ is calculated on N_r views.

4. Results

We compare our model to previous multi-view camouflage methods using cube shapes, as well as on complex animal and furniture shapes.
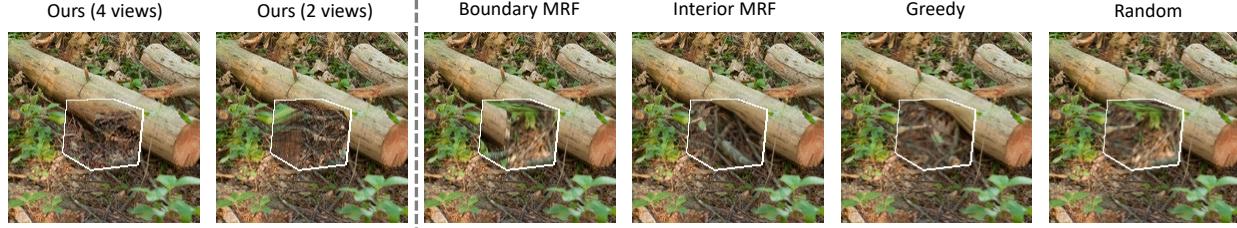
4.1. Dataset

We base our evaluation on the scene dataset of [34], placing objects at their predefined locations. Each scene contains 10-25 photos from different locations. During capturing, only background images are captured, with no actual object is placed in the scene. Camera parameters are estimated using structure from motion [48]. To support learning-based methods that take 4 input views, while still having a diverse evaluation set, we use 36 of the 37 scenes (removing one very small 6-view scene). In [34], their methods are only evaluated on cuboid shape, while our method can be adapted to arbitrary shape without any change to the model. To evaluate our method on complex shapes, we generate camouflage textures for a dataset of 49 animal meshes from [62]. We also provide a qualitative furniture shape from [11] (Fig. 1).
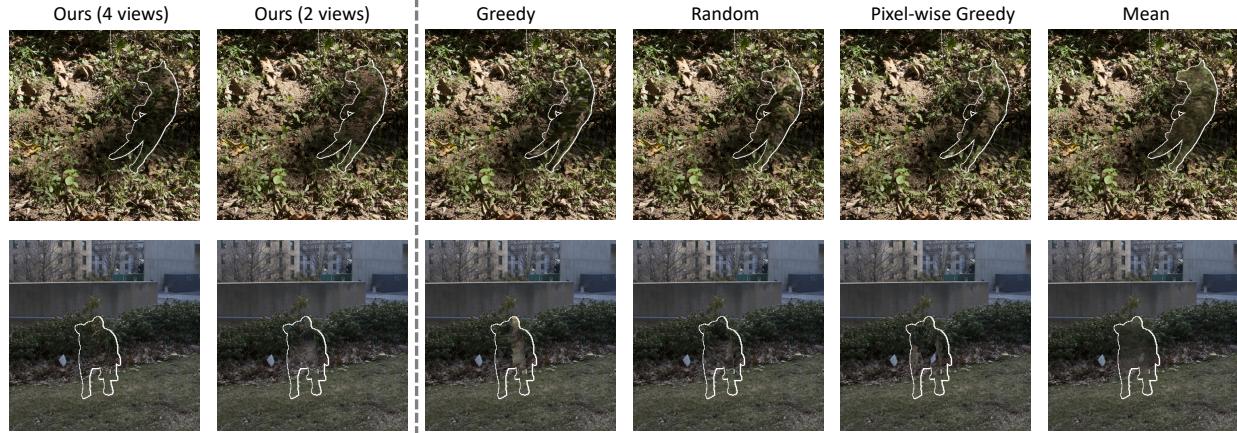
4.2. Implementation Details

For each scene, we reserve 1-3 images for testing (based on the total number of views in the scene). Following other work in neural textures [23], we train one network per scene. We train our models using the Adam optimizer [27] with a learning rate of 2×10^{-4} for the texturing function G and 10^{-4} for the discriminator D. We use $\lambda_{adv} = 0.5$ in Eq. 8. We resize all images to be 384×576 and use square crops of 128×128 to calculate losses.

To ensure that our randomly chosen object locations are likely to be clearly visible from the cameras, we randomly sample object positions on the ground plane (the base of the cube in [34]). We allow these objects to be shifted at most $3\times$ the cube's length. During training, for each sample, we randomly select $N_i = 4$ views as input views and render the object on another $N_r = 2$ novel views. The model is trained with batch size of 8 for approximately 12k iterations. For evaluation, we place the object at the predefined position from [34] and render it in the reserved test views.

(a) Qualitative results on cubes

(b) Qualitative results on animal shapes

Figure 4. **Comparison between methods for cuboids and complex shapes.** We compare our method with previous approaches for the task of concealing (a) cuboids and (b) animal shapes. Our method produces objects with more coherent texture, with the 4-view model filling in textures that tend to be occluded.

4.3. Experimental Settings

4.3.1 Cuboid shapes

We first evaluate our method using only cuboid shapes to compare with the state-of-the-art methods proposed in Owens et al. [34]. We compare our proposed 2-view and 4-view models with the following approaches:

Mean. The color for each 3D point is obtained by projecting it into all the views that observe it and taking the mean color at each pixel.

Iterative projection. These methods exploit the fact that an object can (trivially) be completely hidden from a single given viewpoint by back-projecting the image onto the object. When this is done, the object is also generally difficult to see from *nearby* viewpoints as well. In the *Random* method, the input images are selected in a random order, and each one is projected onto the object, coloring any surface point that has not yet been filled. In *Greedy*, the model samples the photos according to a heuristic that prioritizes viewpoints that observe the object head-on (instead of random sampling). Specifically, the photos are sorted based on the number of object faces that are observed from a direct angle ($> 70°$ with the viewing angle).

Example-based texture synthesis. These methods use Markov Random Fields (MRFs) [1, 16, 37] to perform example-based texture synthesis. These methods simultaneously minimize photoconsistency, as well as smoothness cost that penalizes unusual textures. The Boundary MRF model requires nodes within a face to have same labels, while Interior MRF does not.

4.3.2 Complex shapes

We also evaluated our model on a dataset containing 49 animal meshes [62]. Camouflaging these shapes presents unique challenges. In cuboids, the set of object points that each camera observes is often precisely the same, since each viewpoint sees at most 3 adjacent cube faces (out of 6 total). Therefore, it often suffices for a model to camouflage the most commonly-viewed object points with a single, coherent texture taken from one of the images, putting any conspicuous seams elsewhere on the object. In contrast, when the meshes have more complex geometry, each viewpoint sees a very different set of object points.

Since our model operates on arbitrary shapes, using these shapes requires no changes to the model. We trained our method with the animal shapes and placed the animal object at the same position as in the cube experiments. We adapt the simpler baseline methods of [34] to these shapes, however we note that the MRF-based synthesis methods assume a grid graph structure on each cube face, and hence cannot be adapted to complex shapes without significant changes.

Method	Confusion rate	Avg. time (s)	Med. time (s)	n
Mean	$16.09\% \pm 2.29$	4.82 ± 0.37	2.95 ± 0.14	988
Random	$39.66\% \pm 3.02$	7.63 ± 0.50	4.68 ± 0.35	1011
Greedy	$40.32\% \pm 2.96$	7.94 ± 0.52	4.72 ± 0.36	1054
Boundary MRF [34]	$41.29\% \pm 2.95$	8.50 ± 0.51	5.39 ± 0.40	1068
Interior MRF [34]	$44.66\% \pm 3.01$	8.19 ± 0.51	5.19 ± 0.42	1048
Ours (2 views)	$\mathbf{51.58}\% \pm \mathbf{2.99}$	$\mathbf{9.19} \pm \mathbf{0.51}$	$\mathbf{6.46} \pm \mathbf{0.42}$	1074
Ours (4 views)	$\mathbf{53.95}\% \pm \mathbf{3.05}$	$\mathbf{9.29} \pm \mathbf{0.57}$	$\mathbf{6.11} \pm \mathbf{0.50}$	1025

Table 1. **Perceptual study results with cubes.** Higher numbers represent a better performance. We report the 95% confidence interval of these metrics.

Method	Confusion rate	Avg. time (s)	Med. time (s)	n
Mean	$36.46\% \pm 2.17$	6.39 ± 0.30	4.04 ± 0.17	1898
Pixel-wise greedy	$50.43\% \pm 2.20$	7.25 ± 0.32	4.73 ± 0.20	1987
Random	$51.61\% \pm 2.29$	7.81 ± 0.36	5.25 ± 0.36	1831
Greedy	$52.50\% \pm 2.18$	7.69 ± 0.34	5.13 ± 0.25	2017
Ours (4 views)	$\mathbf{61.93}\% \pm \mathbf{2.14}$	$\mathbf{8.06} \pm \mathbf{0.33}$	$\mathbf{5.66} \pm \mathbf{0.27}$	1970

Table 2. **Perceptual study results on animal shapes.** Higher numbers represent a better performance. We report the 95% confidence interval of these metrics.

Mean. As with cube experiment, we take the mean color from multiple input views as the simplest baseline.

Iterative projection. We use the same projection order selection strategy as in cube experiment. We determine whether a pixel is visible in the input views by using a ray-triangle intersection test.

Pixel-wise greedy. Instead of projecting each input in sequential order, we choose the color for each pixel by selecting color from the input views that has largest view angle.

4.4. Perceptual Study

To evaluate the effectiveness of our method, we conduct a perceptual study. We generally follow the setup of [34], however we ask users to directly click on the camouflaged object [54], without presenting them with a second step to confirm that the object (or isn't) present. This simplified the number of camouflaged objects that subjects see by a factor of two. We recruited 267 and 375 participants from Amazon Mechanical Turk for the perceptual study on cuboid and complex shapes, respectively, and ensured no participant attended both of the perceptual studies.

Each participant was shown one random image from the reserved images of each scene in a random order. The first 5 images that they were shown were part of a training exercise, and are not included in the final evaluation. We asked participants to search for the camouflaged object in the scene, and to click on it as soon as they found it. The object in the scene was camouflaged by a randomly chosen algorithm, and placed at the predefined position. After clicking on the image, the object outline was shown to the participant. We recorded whether the participant correctly clicked on the camouflaged object, and how long it took them to click. Each participant had one trial for each image and a maximum of $60s$ to find the camouflaged object.

Results on cuboid shapes. The perceptual study results on cuboid shapes are shown in Table 1. We report the con-

(a) Real cube   (b) With shadow   (c) No shadow

Figure 5. **Effect of shadow on generated textures.** We simulate the effect of shadows of the object in an indoor scene, using the reference object (a). Our model generates a texture with a shadow (b) by conditioning on composite images that contain the real shadow (but no real cube). (c) Result without shadow modeling.

fusion rate, average time, and median time measured over different methods. We found that our models significantly outperform the previous approaches on all metrics. To test for significance, we followed [34] and used a two-sided t-test for the confusion rate and a two-sided Mann-Whitney U test (with a 0.05 threshold for significance testing). We found that our method outperforms all the baseline methods significantly in the confusion rate metric. Both of our model variations outperform Interior MRF ($p < 2 \times 10^{-3}$ and $p < 3 \times 10^{-5}$). There was no significant difference between 2 and 4 views ($p = 0.28$). In terms of time-to-click, our method also beats the two MRF-based methods. Compared with Boundary MRF, our method requires more time for participants to click the camouflaged object ($p = 0.0024$ for 2 views and $p = 0.039$ for 4 views).

Results on complex shapes. The perceptual study results on complex shapes are shown in Table 2. We found that our model obtained *significantly better* results than previous work on confusion rate. Our model also obtained significantly better results on the time-to-find metric. We found that in terms of confusion rate, our method with 4 input views is significantly better than the baseline methods, 9.42% better than Greedy method and 10.32% better than Random method. For time-to-click, our method also performs better than baseline methods compared with Greedy and Random.

4.5. Qualitative Results

We visualize our generated textures in several selected scenes for both cube shapes and animal shapes in Figure 3. We compare our method qualitatively with baseline methods from [34] in Figure 4. We found that our model obtained significantly more coherent textures than other approaches. The 2-view model has a failure case when none of the input views cover an occluded face, while the 4-view model is able to generally avoid this situation. We provide additional results in the supplement.

Effects of shadows. Placing an object in a real scene may create shadows. We ask how these shadows effect

| Ours (4 views) | No $\mathcal{L}_{adv}$ | No $\mathcal{L}_{photo}$ on input views | pixelNeRF encoder |

Figure 6. **Ablations.** We show how the choice of different components changes the quality of the camouflage texture.

Model	LPIPS↓	SIFID↓
Boundary MRF [34]	0.1228	0.0867
Interior MRF [34]	0.1185	0.0782
DeepFill v2 [58] + Projection [34]	0.1469	0.1245
LaMa [51] + Projection [34]	0.1263	0.1006
LDM [40] + Projection [34]	0.1305	0.0976
No $\mathcal{L}_{adv}$	0.1064	0.0720
No $\mathcal{L}_{photo}$ on input views	0.1131	0.0856
With pixelNeRF encoder [57]	0.1047	**0.0712**
Ours (2 views)	0.1079	0.0754
Ours (4 views)	**0.1034**	0.0714

Table 3. **Evaluation with automated metrics.** We compare our method to other approaches, and perform ablations.

our model's solution (Figure 5), exploiting the fact that these shadows are independent of the object's texture and hence function similarly to other image content. In [34], photos with (and without) a real cube are taken from the same pose. We manually composite these paired images to produce an image without the real cube but with its real shadow. We then provide these images as conditioning input to our model, such that it incorporates the presence of the shadow into its camouflage solution. While our solution incorporates some of the shadowed region, the result is similar. Note that other lighting effects can be modeled as well (*e.g.*, by compensating for known shading on the surface).

4.6. Automated evaluation metrics

To help understand our proposed model, we perform an automated evaluation and compare with ablations:

- **Adversarial loss:** To evaluate the importance of $\mathcal{L}_{adv}$, we set λ_{adv} to 0 in Eq. 8. We evaluate the model performance with only $\mathcal{L}_{photo}$ used during training.
- **Photoconsistency:** We evaluate the importance of using all N_i input views in Eq. 5. The ablated model has $\mathcal{L}_{photo}$ only calculated on N_r rendering views during training.
- **Architecture:** We evaluate the importance of our pixel-aligned feature representation. In lieu of this network, we use the feature encoder from pixelNeRF [57].
- **Inpainting:** Since inpainting methods cannot be directly applied to our task without substantial modifications, we combind several inpainting methods with the Greedy model. We selected several recent inpainting methods DeepFillv2 [58], LaMa [51], LDM [40] to inpaint the object shape in each view, then backproject this texture onto the 3D surface, using the geometry-based ordering from [34].

Evaluation metrics. To evaluate the ablated models, we use LPIPS [61] and SIFID metrics [44]. Since the background portion of the image remains unmodified, we use crops centered at the rendered camouflaged objects.

Results. Quantitative results are shown in Table 3 and qualitative results are in Figure 6. We found that our full 4-view model is the overall best-performing method. In particular, it significantly outperforms the 2-view model, which struggles when the viewpoints do not provide strong coverage from all angles (Fig. 6). We also found that the adversarial loss significantly improves performance. As can be seen in Fig. 6, the model without an adversarial loss fails to choose a coherent solution and instead appears to average all of the input views. The model that uses all views to compute photoconsistency tends to generate more realistic textures, perhaps due to the larger availability of samples. Compared with the pixelNeRF encoder, our model generates textures with higher fidelity, since it receives more detailed feature maps from encoder. We obtain better performance on LPIPS but find that this variation of the model achieves slightly better SIFID. This suggests that the architecture of our pixel-aligned features provides a modest improvement. Finally, we found that we significantly outperformed the inpainting and MRF-based methods.

5. Discussion

We proposed a method for generating textures to conceal a 3D object within a scene. Our method can handle diverse and complex 3D shapes and significantly outperforms previous work in a perceptual study. We see our work as a step toward developing learning-based camouflage models. Additionally, the animal kingdom has a range of powerful camouflage strategies, such as disruptive coloration and mimicry, that cleverly fool the visual system and may require new learning methods to capture.

Limitations. As in other camouflage work [34], we do not address the problem of physically creating the camouflaged object, and therefore do not systematically address practicalities like lighting and occlusion.

Ethics. The research presented in this paper has the potential to contribute to useful applications, particularly to hiding unsightly objects, such as solar panels and utility boxes. However, it also has the potential to be used for negative applications, such as hiding nefarious military equipment and intrusive surveillance cameras.

References

[1] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. In *ACM SIGGRAPH 2004 Papers*, pages 294–302. 2004. 6

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018. 2

[3] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 3

[4] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, 2000. 3

[5] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Patch-based optimization for image-based texture mapping. *ACM Trans. Graph.*, 36(4):106–1, 2017. 2

[6] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017. 2

[7] Joshua Callaghan. Public art projects, 2016. 1

[8] Hala Lamdouar Charig Yang, Erika Lu, Andrew Zisserman, and Weidi Xie. Self-supervised video object segmentation by motion grouping. 2021. 2

[9] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. *arXiv preprint arXiv:2103.15595*, 2021. 3, 4

[10] Hung-Kuo Chu, Wei-Hsin Hsu, Niloy J Mitra, Daniel Cohen-Or, Tien-Tsin Wong, and Tong-Yee Lee. Camouflage images. *ACM Trans. Graph.*, 29(4):51–1, 2010. 2

[11] Jasmine Collins, Shubham Goel, Achleshwar Luthra, Leon Xu, Kenan Deng, Xi Zhang, Tomas F Yago Vicente, Himanshu Arora, Thomas Dideriksen, Matthieu Guillaumin, and Jitendra Malik. Abo: Dataset and benchmarks for real-world 3d object understanding. *arXiv preprint arXiv:2110.06199*, 2021. 5

[12] Hugh Bamford Cott. Adaptive coloration in animals. 1940. 2

[13] Richard Dawkins et al. *The blind watchmaker: Why the evidence of evolution reveals a universe without design*. WW Norton & Company, 1996. 2

[14] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999. 3

[15] Deng-Ping Fan, Ge-Peng Ji, Guolei Sun, Ming-Ming Cheng, Jianbing Shen, and Ling Shao. Camouflaged object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2777–2787, 2020. 2

[16] William T Freeman, Thouis R Jones, and Egon C Pasztor. Example-based super-resolution. *IEEE Computer graphics and Applications*, 22(2):56–65, 2002. 6

[17] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 3, 4

[18] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2

[19] Roger Hanlon. Cephalopod dynamic camouflage. *Current Biology*, 17(11):R400–R404, 2007. 2

[20] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015. 2, 4, 11

[21] James Hays and Alexei A Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26(3):4–es, 2007. 3

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 11

[23] Philipp Henzler, Niloy J Mitra, , and Tobias Ritschel. Learning a neural 3d texture space from 2d exemplars. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 3, 4, 5

[24] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 5

[25] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 11

[26] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016. 4

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[28] Hala Lamdouar, Charig Yang, Weidi Xie, and Andrew Zisserman. Betrayed by motion: Camouflaged object discovery via motion segmentation. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 2

[29] Trung-Nghia Le, Yubo Cao, Tan-Cong Nguyen, Minh-Quan Le, Khanh-Duy Nguyen, Thanh-Toan Do, Minh-Triet Tran, and Tam V Nguyen. Camouflaged instance segmentation in-the-wild: Dataset and benchmark suite. *arXiv preprint arXiv:2103.17123*, 2, 2021. 2

[30] Rob Matheson. Solar panels get a face-lift with custom designs, 2017. 1

[31] Sami Merilaita, Nicholas E Scott-Samuel, and Innes C Cuthill. How camouflage works. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 372(1724):20160341, 2017. 1

[32] 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. In *European conference on computer vision*, pages 405–421. Springer, 2020. 3, 4, 11

[33] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4531–4540, 2019. 2, 3, 4

[34] Andrew Owens, Connelly Barnes, Alex Flint, Hanumant Singh, and William Freeman. Camouflaging an object from many viewpoints. In *CVPR*, 2014. 1, 2, 3, 5, 6, 7, 8, 12

[35] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016. 2, 3

[36] Tiziano Portenier, Siavash Arjomand Bigdeli, and Orcun Goksel. Gramgan: Deep 3d texture synthesis from 2d exemplars. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6994–7004. Curran Associates, Inc., 2020. 2, 3

[37] Yael Pritch, Eitam Kav-Venaki, and Shmuel Peleg. Shift-map image editing. In *2009 IEEE 12th international conference on computer vision*, pages 151–158. IEEE, 2009. 6

[38] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 4

[39] Craig Reynolds. Interactive evolution of camouflage. *Artificial life*, 17(2):123–136, 2011. 2

[40] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 8

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. 4, 11

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 4

[43] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019. 2

[44] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4570–4580, 2019. 8, 12

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 4

[46] Karl Sims. Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372, 1994. 2

[47] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019. 3

[48] Noah Snavely, Steven Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. acm trans graph 25(3):835-846. *ACM Trans. Graph.*, 25:835–846, 07 2006. 5

[49] Martin Stevens and Sami Merilaita. *Animal camouflage: mechanisms and function*. Cambridge University Press, 2011. 2

[50] Jack Stewart. Tesla unveils its new line of camouflaged solar panels, 2016. 1

[51] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. *arXiv preprint arXiv:2109.07161*, 2021. 8

[52] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 2

[53] Laszlo Talas, John G Fennell, Karin Kjernsmo, Innes C Cuthill, Nicholas E Scott-Samuel, and Roland J Baddeley. Camogan: Evolving optimum camouflage with generative adversarial networks. *Methods in Ecology and Evolution*, 11(2):240–247, 2019. 1, 2

[54] Jolyon Troscianko, Jared Wilson-Aggarwal, David Griffiths, Claire N Spottiswoode, and Martin Stevens. Relative advantages of dichromatic and trichromatic color vision in camouflage breaking. *Behavioral Ecology*, 28(2):556–564, 2017. 2, 7

[55] Naoto Usuyama and Karanbir Chahal. Unet/fcn pytorch, Accessed: 2021. 11

[56] Jinnan Yan, Trung-Nghia Le, Khanh-Duy Nguyen, Minh-Triet Tran, Thanh-Toan Do, and Tam V Nguyen. Mirrornet: Bio-inspired camouflaged object segmentation. *IEEE Access*, 9:43290–43300, 2021. 2

[57] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 2, 3, 4, 8, 11

[58] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. *arXiv preprint arXiv:1806.03589*, 2018. 8

[59] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018. 3

[60] Qing Zhang, Gelin Yin, Yongwei Nie, and Wei-Shi Zheng. Deep camouflage images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12845–12852, 2020. 2

[61] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 8, 12

[62] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 5, 6

Appendix

A. Implementation details

Encoder E**.** In our model, we use a U-net [41] with ResNet-18 [22] as our feature encoder to extract multi-scale image feature maps from multi-view input images. The U-net has feature maps in 5 different scales in the decoder with shapes:

1. $32 \times H \times W$
2. $64 \times \frac{H}{2} \times \frac{W}{2}$
3. $64 \times \frac{H}{4} \times \frac{W}{4}$
4. $128 \times \frac{H}{8} \times \frac{W}{8}$
5. $128 \times \frac{H}{16} \times \frac{W}{16}$

We select 3 level of feature maps($1, \frac{1}{4}, \frac{1}{16}$ scales) as the hypercolumn [20] features to the lateral multi-layer perceptron T, resulting in a total of 224 channels. We use the implementation of [55].

Texture field architecture. We base our texture field MLP on [57]. The detailed structure is shown in Figure 7. The conditional vectors $\{\mathbf{z}_i^{(j)}\}, \{\mathbf{v}_i^{(j)}\}, \{\mathbf{n}_i^{(j)}\}$ have 230 channels (224 from $\mathbf{z}_i^{(j)}$, 3 from $\mathbf{v}_i^{(j)}$ and 3 from $\mathbf{n}_i^{(j)}$) from each input view. The positional encoding [32] is computed as:

$$\gamma(\mathbf{x}_i) = \begin{bmatrix} \mathbf{x}_i \\ \cos(2^0 \mathbf{x}_i) \\ \sin(2^0 \mathbf{x}_i) \\ \vdots \\ \cos(2^{L-1} \mathbf{x}_i) \\ \sin(2^{L-1} \mathbf{x}_i) \end{bmatrix}. \quad (9)$$

We set $L = 10$, which results in a size of 63 for $\gamma(\mathbf{x}_i)$.

The network is composed of 2 stages. In the first stage, the hidden size is 256 dimensions. We have N separate branches with shared weights for the N input views. After a unified feature representation is generated by mean pooling, the second stage predicts an RGB color for the query point.

Discriminator D**.** For the discriminator, we use the model of [25], after replacing the batch normalization layers with instance normalization layers. The network is

Method	Confusion rate↑	Avg. time↑	Med. time↑	n	LPIPS [61]↓	SIFID [44]↓
Mean	16.09% ± 2.29	4.82 ± 0.37	2.95 ± 0.14	988	0.1609	0.1637
Random	39.66% ± 3.02	7.63 ± 0.50	4.68 ± 0.35	1011	0.1365	0.0966
Greedy	40.32% ± 2.96	7.94 ± 0.52	4.72 ± 0.36	1054	0.1312	0.0914
Boundary MRF [34]	41.29% ± 2.95	8.50 ± 0.51	5.39 ± 0.40	1068	0.1228	0.0867
Interior MRF [34]	44.66% ± 3.01	8.19 ± 0.51	5.19 ± 0.42	1048	0.1185	0.0782
Ours (2 views)	**51.58% ± 2.99**	**9.19 ± 0.51**	**6.46 ± 0.42**	1074	0.1079	0.0754
Ours (4 views)	**53.95% ± 3.05**	**9.29 ± 0.57**	**6.11 ± 0.50**	1025	**0.1034**	**0.0714**

Table 4. **Quantitative results with cubes.**

Method	Confusion rate↑	Avg. time↑	Med. time↑	n	LPIPS [61]↓	SIFID [44]↓
Mean	36.46% ± 2.17	6.39 ± 0.30	4.04 ± 0.17	1898	0.0883	0.0441
Pixel-wise greedy	50.43% ± 2.20	7.25 ± 0.32	4.73 ± 0.20	1987	0.0976	0.0590
Random	51.61% ± 2.29	7.81 ± 0.36	5.25 ± 0.36	1831	0.0888	0.0418
Greedy	52.50% ± 2.18	7.69 ± 0.34	5.13 ± 0.25	2017	0.0881	0.0419
Ours (4 views)	**61.93% ± 2.14**	**8.06 ± 0.33**	**5.66 ± 0.27**	1970	**0.0798**	**0.0350**

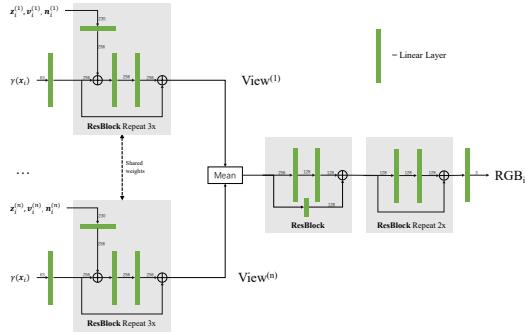Table 5. **Quantitative results with animal shapes.**

Figure 7. **Architecture of Multi-Layer Perceptron.** Our model applies a sequence of residual blocks with shared weights to the embedding provided by each viewpoint. We average pool across all views, then predict a color.

composed of a sequence of Convolution-InstanceNorm-LeakyReLu blocks. In particular, it has structures of channels of $64, 128, 256, 512$ in each of its blocks. Instance normalization is not applied to the first block.

B. Additional Results

Quantitative metrics. We show the calculated LPIPS [61] and SIFID [44] scores on crops of rendered camouflaged objects in Table 4 (cube) and Table 5 (animal shapes). Objects are placed at the exactly same place in each scene for each method. We use a square crop centered at the object that has size of $32\lceil d/32 \rceil + 32$, where d is the maximum dimension of a foreground camouflaged object. We calculate these two metrics on all test views and report the average scores on all scenes.

Qualitative Results. We show the 36 scenes used in our evaluation, along with the different methods. Cube results are shown in Figure 8, and animal results are shown in Figure 9-11. The viewpoint and animal shapes are **randomly selected** for these visualizations. We crop the images around the objects to show them more clearly.

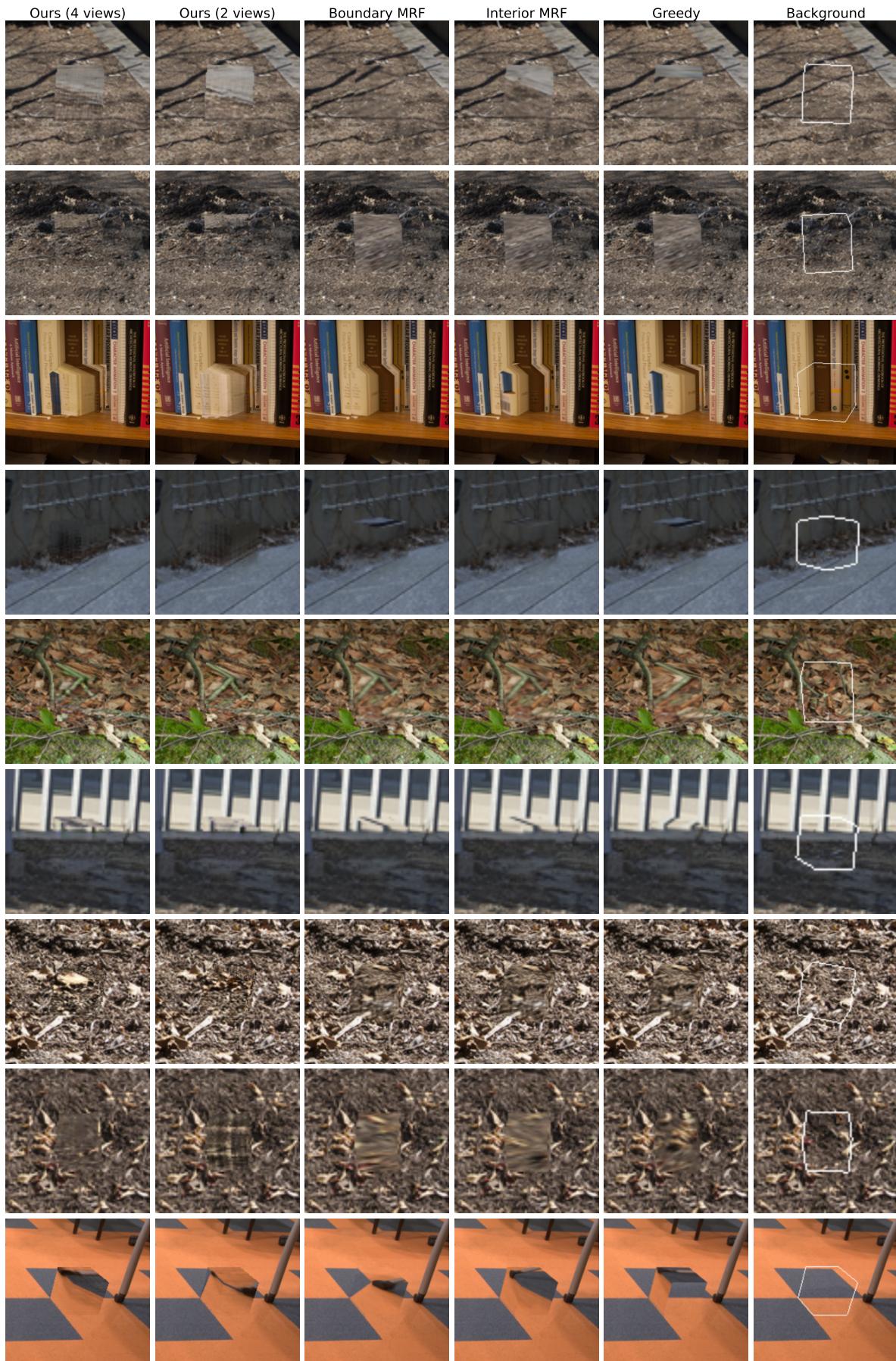| Ours (4 views) | Ours (2 views) | Boundary MRF | Interior MRF | Greedy | Background |

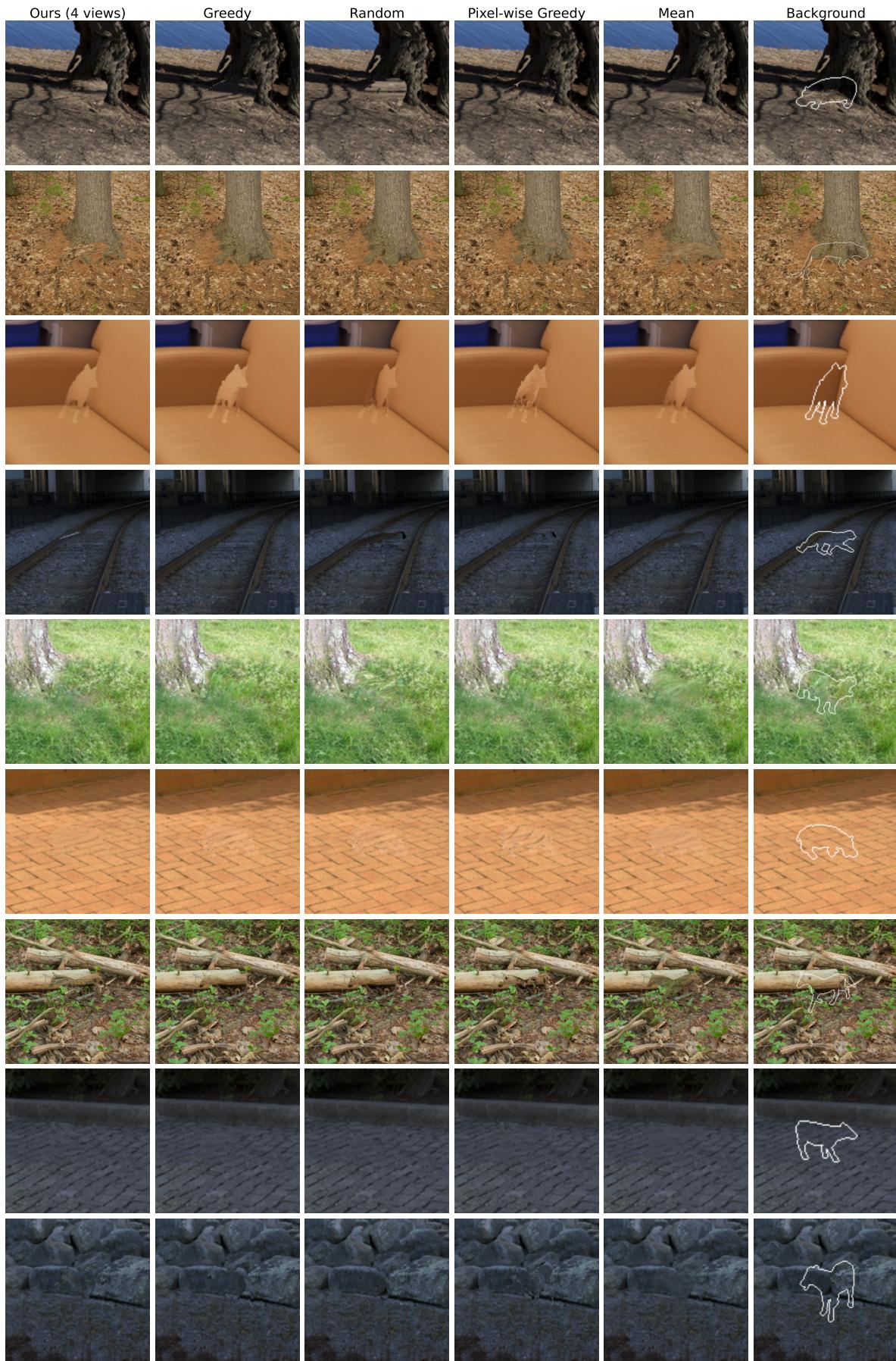Figure 8. Comparison between methods for camouflaging cubes.

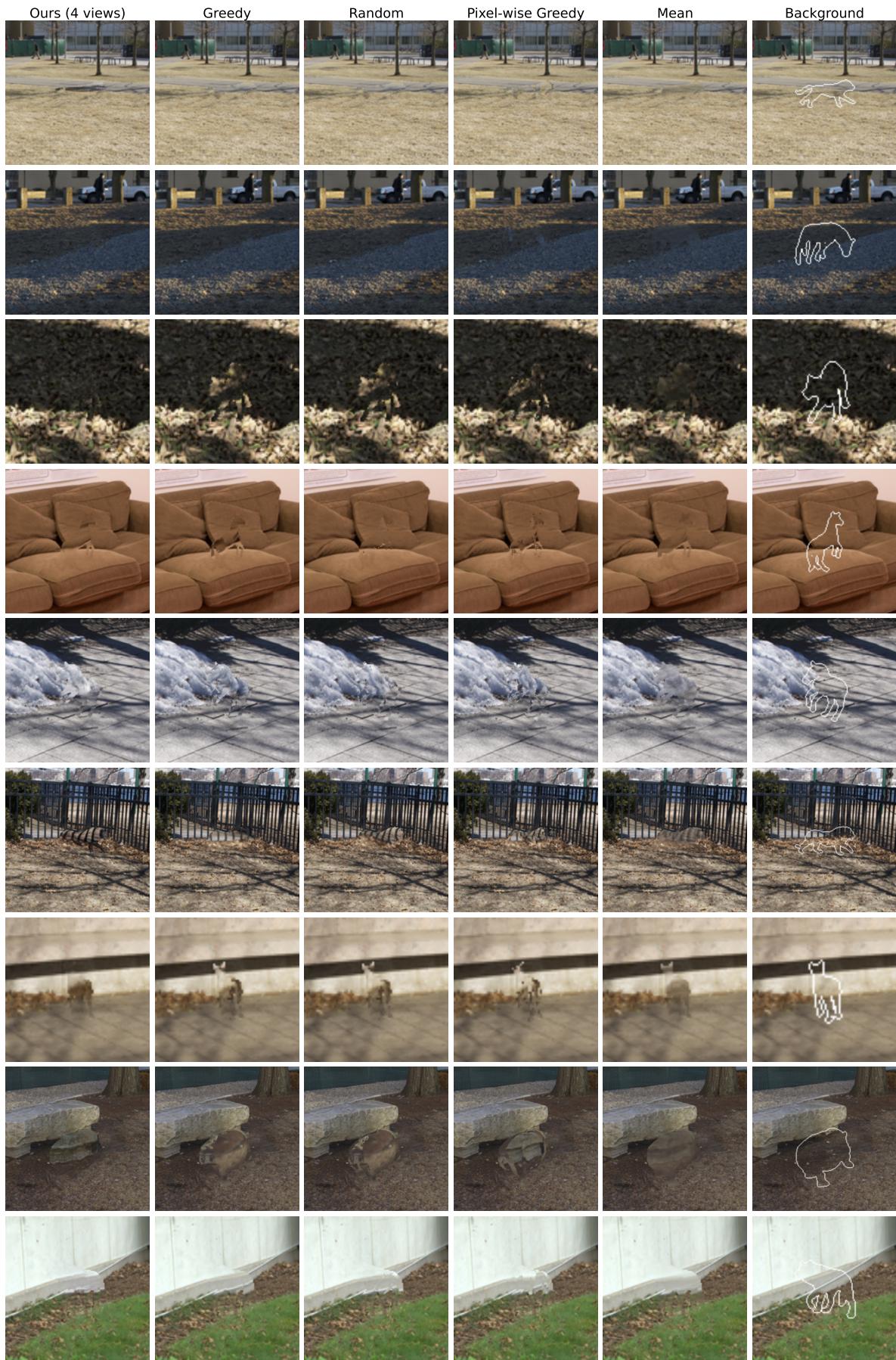Figure 9. Comparison between methods for camouflaging animals (page 1 of 3).

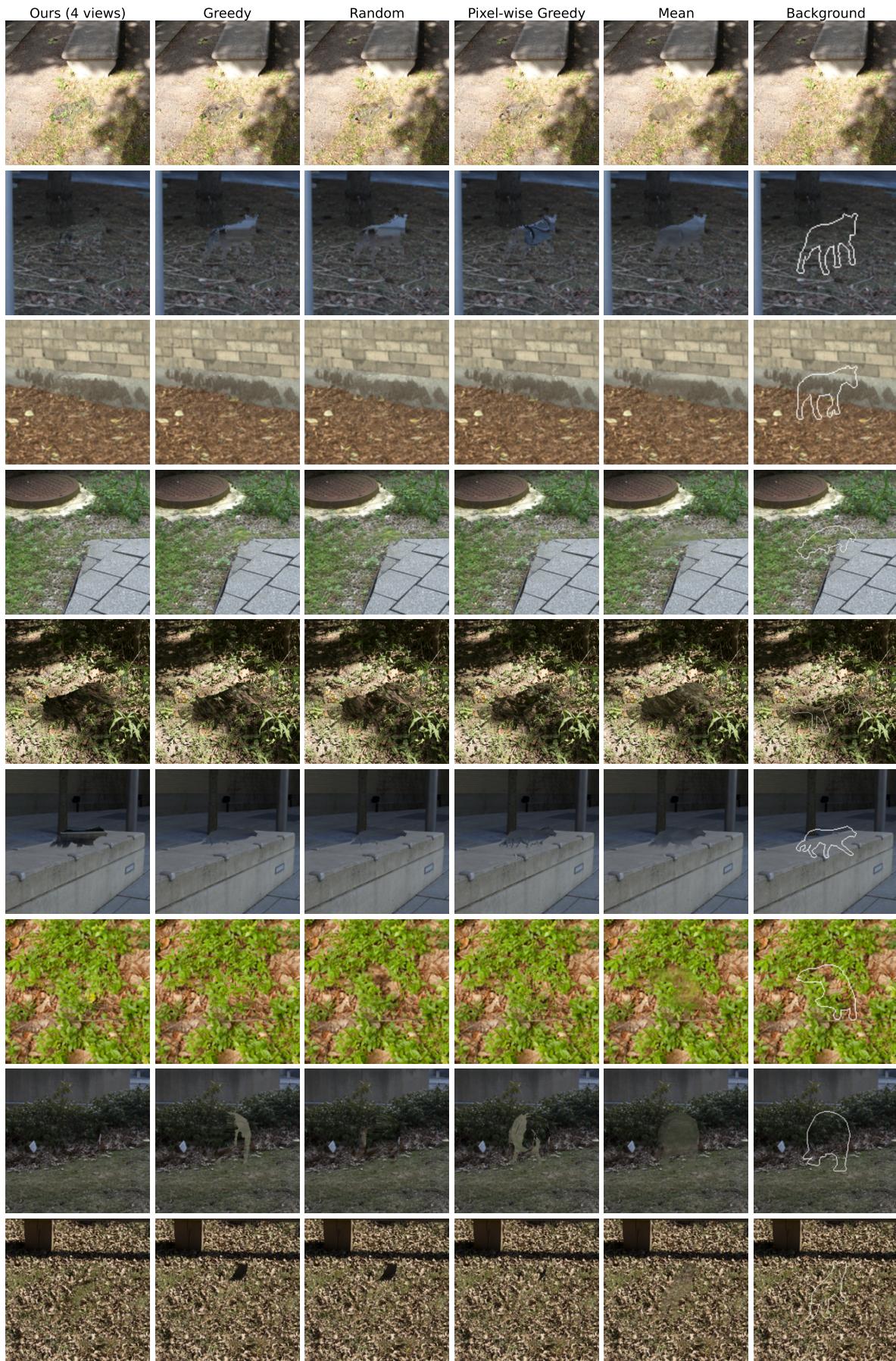Figure 10. Comparison between methods for camouflaging animals (page 2 of 3)

Figure 11. Comparison between methods for camouflaging animals (page 3 of 3)