Coupling Rendering and Generative Adversarial Networks for Artificial SAS Image Generation

Albert Reed*, Isaac D. Gerg†¶, John D. McKay†, Daniel C. Brown†‡, David P. Williams‖, Suren Jayasuriya*§

*School of Electrical, Computer & Energy Engineering, §School of Arts, Media & Engineering, Arizona State University Tempe, AZ USA

†Applied Research Laboratory, ‡Graduate Program in Acoustics, ¶School of Electrical Engineering and Computer Science, Pennsylvania State University State College, PA USA

‖NATO STO Centre for Maritime Research & Experimentation, La Spezia, Italy

I. INTRODUCTION

There is a growing demand for large-scale Synthetic Aperture Sonar (SAS) datasets. This demand stems from data-driven applications such as Automatic Target Recognition (ATR) [1]–[3], segmentation [4] and oceanographic research of the seafloor, simulation for sensor prototype development and calibration [5], and even potential higher level tasks such as motion estimation [6] and micronavigation [7]. Unfortunately, the acquisition of SAS data is bottlenecked by the costly deployment of SAS imaging systems, and even when data acquisition is possible, the data is often skewed towards containing barren seafloor rather than objects of interest. This skew introduces a data imbalance problem wherein a dataset can have as much as a 1000-to-1 ratio of seafloor background to object-of-interest SAS image chips.

An alternative to real-world SAS image capture is to generate artificial SAS images that can be used to construct large-scale datasets. This has been approached through either model-driven, physics-based approaches or more recently through data-driven, machine learning approaches such as generative adversarial networks (GANs). These two methods have seen relative levels of success in synthesizing SAS data. Physics-based models, such as scattering models [8], [9], allow for absolute user specification and control of environment and SAS physics interactions, and simulate physical realistic effects such as speckle and spatial coherence length. However, these models induce an intractable computational burden (i.e. hours for a single image), and are unable to produce SAS realistic images due to the complexity of modeling the entire scene and environment explicitly. On the other hand, data-driven models such as GANs can rapidly generate a large number of realistic SAS images that match input distribution statistics. However, the user has little control or specification over scene content in these images, and we will show that these models cannot generalize and generate truly novel images, especially when trained in the data-starved SAS regime.

To leverage the strengths of both model and data-driven approaches, we propose a hybrid pipeline combining the two.

In particular, we utilize an optical renderer coupled with a GAN. The optical renderer serves as a method for quickly rendering an image that approximates the interactions of SAS systems with objects on the seafloor, and the GAN ingests these images as input and colors them with SAS-realistic properties which are traditionally hard to model in closed form.

Our specific contributions are as follows:

1) Use of an open source physically-based optical ray tracer to render artificial SAS images.
2) A Wasserstein generative adversarial network to color these optically rendered SAS images with the visual and statistical qualities of real SAS images.
3) A hybrid pipeline from optical rendering to a GAN that allows for control over target and scene geometry, background, and sources while maintaining this SAS image realism.

These contributions are validated on a real SAS dataset [10], and we perform qualitative and quantitative analysis to study both the benefits and drawbacks of our approach. We hope this work spurs more synergistic combinations of physics-based modeling with data-driven machine learning approaches for SAS in the future.

II. RELATED WORK

A. Generative Adversarial Networks

We introduce the literature surrounding generative adversarial networks (GANs) for those unfamiliar with these machine learning models. Generative adversarial networks (GANs), invented by Goodfellow et al. [11], are generative models that learn to generate new data that follow the distribution statistics of a given training dataset. For example, a GAN trained on images of cats will produce images of cats not seen before in the training data. GANs have been used to generate photorealistic faces of fake celebrities [12], new audio waveforms [13], and even new clothing for fashion [14].

A GAN's fundamental form comprises two neural networks: a generator and a discriminator. The generator network samples random vectors from a high dimensional prior distribution

(e.g. Gaussian) as input, and transforms this input vector into a generated image through a series of non-linear upsampling operations. This prior distribution is typically called the latent space for the GAN, and encodes semantic information (e.g. type of object) once the GAN has been trained, although this information is not fully interpretable or explicit. The discriminator network is presented the generator's output image as well as a real image from the training dataset, and then tasked with labeling generated images and real dataset images as either fake or real. In summary, a GAN trains its generator to produce more plausible images similar to the training dataset, and the discriminator to be the critic and force the generator to improve its performance in order to fool the discriminator. This process is known as adversarial training. Formally, the GAN loss function defines a minimax game which encourages the discriminator to assign correct labels and the generator to trick the discriminator into assigning the incorrect labels, and the solution to this minimax game minimizes the Jensen-Shannon divergence between the generated and training image distributions [11].

This adversarial training can be difficult in practice due to convergence issues. There have been a myriad of suggested improvements for training GANs [12], [15], [16]. One of the most effective methods is the Wasserstein-GAN (WGAN), which replaces the original GAN loss function that minimized the Jensen-Shannon divergence between generated and training distributions, with one that minimizes the more stable Wasserstein-1 distance [17], [18]. We utilize this architecture extensively in our experiments.

Scene control. One major drawback of GANs is the inability to control the scene content in generated images, such as the position and appearance of objects. The typical GAN framework samples a random noise input vector and produces an image that appears to belong to the training set statistically, but the user has little to no control to edit the contents of this image. There have been some attempts to remedy this limitation in the literature, but it usually offers coarse control over the latent space [16], [19]–[21]. Image-to-image translation [22], [23] maps images from one domain to another, e.g. changing black-and-white images to color or daytime landscapes to nighttime landscapes. Style transfer uses the layers of convolutional neural networks to transfer the "style" of images from one domain to another, such as a regular image to be "painted" in the style of Monet or Van Gogh [24]. Our contributions adopt methods from image-to-image translation and style transfer literature as we learn the mapping from rendered optical images to a training dataset of SAS images.

GANs for Sonar. While GANs have been used to generate sonar images of bare seafloor, their potential for generating physically realistic targets on the seafloor is relatively unexplored. Chen et al. [25] train GANs on seafloor images and demonstrate that generated images appear both realistic and novel. Style transfer was used to place targets on the seafloor, however this method did not always infer shadow geometries or glint effects. Lee et al. [26] also use style

transfer to color acoustically rendered images with the global style statistics of real sonar images. Sung et al. [27] use a conditional GAN [23] for paired image-to-image translation of optically captured objects to insonified objects for forward-looking sonar applications. In a recent study that is closest to our approach [28], an unpaired image-to-image translation GAN generates targets on the seafloor by using a ray-tracer to insert rendered images onto real seafloors. Shadow geometry and glint lines are then calculated explicitly using elevation maps. In contrast, our approach refines the entire target and seabed jointly, and does not require prior information of the seabed, or the calculation of shadow geometries and glint lines since it infers these properties from the training data.

B. Simulated SAS Images

Modeling the time series of the acoustic field scattered from a scene can be approached through a variety of techniques. The exact calculation of the field scattered from a boundary requires solving the Helmholtz-Kirchhoff integral equation over a set of specified boundary conditions. Analytic solutions to this integral equation exist for a small number of very simplistic target shapes. For more complex scenarios, the solution may be calculated to arbitrary accuracy through numerical techniques such as the boundary element or finite element methods. While these methods are exact, they come with a significant computational burden and are very difficult to scale up to large scenes and/or high frequencies. Frequently, researchers developing a high-frequency SAS model will resort to approximate methods to accelerate the model computation.

A common type of approximate model discretizes the surface into individual scattering elements, independently calculates the fields scattered from each discrete element, and integrates these individually scattered fields. Under this type of model, multiple scattering is ignored as well as elastic scattering mechanisms. There are several existing models that have approached this problem by discretizing the scene into a triangular mesh and applying the Kirchhoff approximation to these individual facets [29]–[34]. Point scattering and point-based scattering models have also proven capable of generating representative time series for a simulated scene [8], [35]. While these approximate techniques are relatively fast, they still have a high computational burden. For a simulation of a high-resolution SAS scene the simulation of a single image can require hours of computation.

The simulation community is attempting to address this drawback through several different methods. Direct generation of magnitude imagery is being approached through generative adversarial networks [25], [27], [28]. Additionally, recent research programs have investigated insertion of realistic targets into preexisting real datasets [36]. Each of these approaches attempts to generate larger quantities of either fully synthetic or hybrid data than are currently available from fielded sensors. These larger datasets are being developed with the goals of using them for machine learning applications.
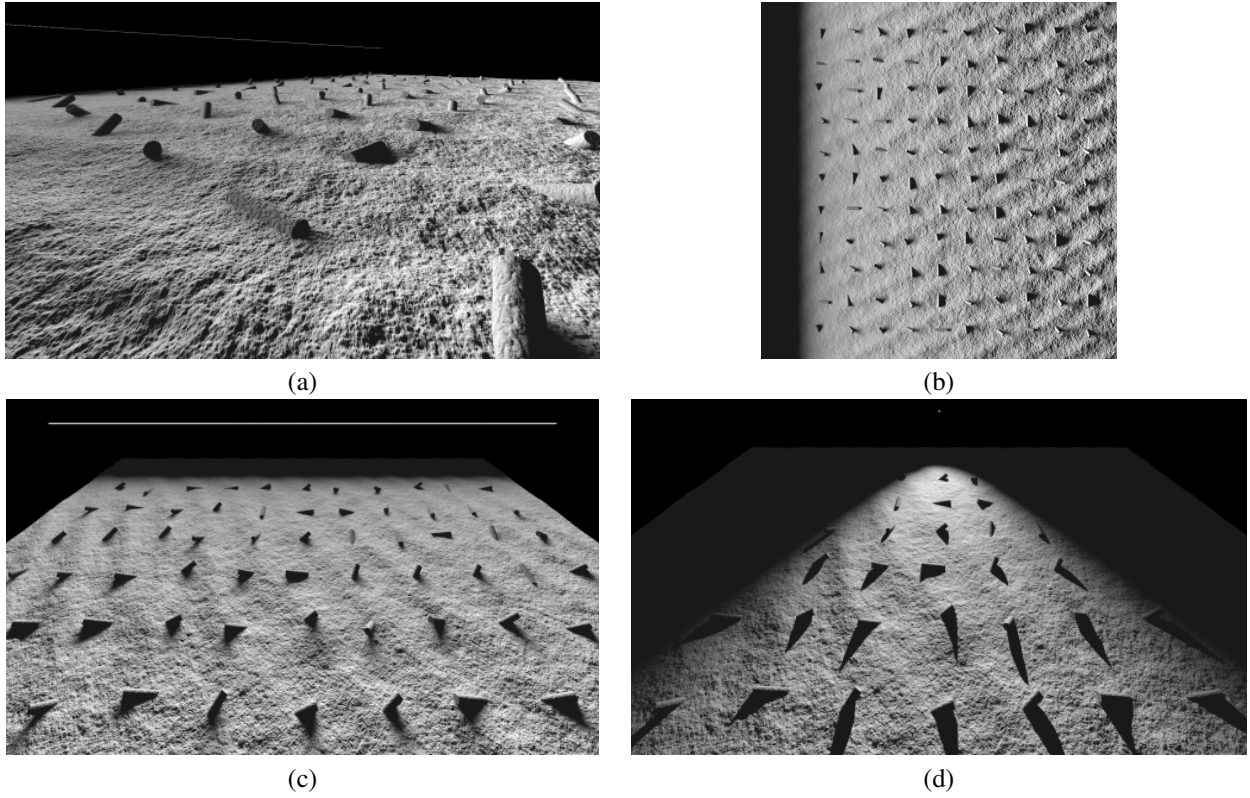
Fig. 1: Artificial target field rendered four different ways to demonstrate the scene geometry and light source beam patterns. The light sources are denoted as point spheres in the scenes. In (a) and (c), the high density of light sources presents as a solid line. (a) Viewpoint rendered from just above the seafloor at range. (b) Rendered scene using the SAS imaging geometry for viewpoint. The scene represents an area of 60m x 60m with cylinders of random orientation and burial depth spaced 5m apart. (c) Array of light sources viewed at max range. (d) Single light source viewed at max range where the conical shaped beam pattern used in the simulations is visible.

III. APPROACH

Our approach leverages both model-driven and data-driven methods to simulate SAS images. In this section, we discuss the components of our proposed pipeline including the choice of optical rendering over acoustic rendering and the use of generative adversarial networks. We find this combination to be complementary in their benefits and drawbacks, and validate this in our experimental results in Section IV.

We use the term *physical realism* for SAS data that satisfy the acoustic physics and have been beamformed from raw data. Note that in this paper, we do not claim physical realism as we are not modeling any acoustical properties explicitly in our machine learning. We use the term *SAS image realism* or *SAS realism* for short, for images that qualitatively look similar to real SAS images for the dynamic-range compressed magnitude image. The goal of this paper is to generate artificial images that have high SAS realism.

A. Optical Rendering of SAS Images using POV-Ray

As noted in Section II-B, there exist several simulators to model the acoustic field for SAS image formation, but these models are either computationally expensive (hours to render

an image) or provide limited control over the geometry of the models, seafloor, environment, and acoustic sources during rendering. For this reason, we decided to render SAS images using an optical renderer, modeling acoustic sources as optical sources and simulating ray tracing to form the image. Optical renderers solve the Rendering equation [37] using Monte Carlo integration, and have been developed to be reasonably fast (order of seconds to minutes for an image) with high control over scene content creation. We acknowledge that to acquire these benefits, we sacrifice both physical realism by using an optical renderer as opposed to an acoustic renderer, as well as SAS realism since our output rendered images do not look like real SAS images. While the first limitation cannot be overcome, achieving better SAS realism can be achieved by our data-driven methods explained in the next subsection.

We utilize the Persistence of Vision Ray Tracer, simply known as POV-Ray [38], a ray tracing software which renders a scene in the optical wavelength regime. It ingests as input a text file describing the scene (including light sources, camera position, and objects) and outputs an image as seen by the camera through its associated parameters. POV-Ray is capable of rendering non-trivial scattering physics such as
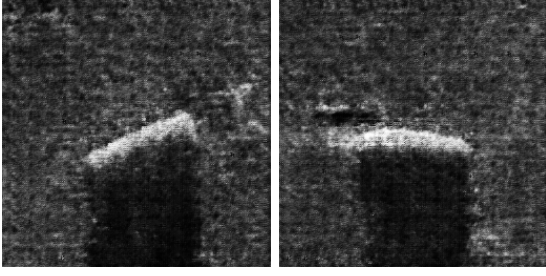
Fig. 2: Two generated examples from DCGAN. We noted that DCGAN mimicked the training dataset by only generating objects centered in the tile.

reflection, refraction, and texture. We use POV-Ray to generate a grayscale optical image with similar observation properties as a SAS image operating at high frequency (specifically $\frac{f_{max}}{f_{min}} \leq 1.5$) [39].

We accomplish this by creating a scene consisting of four items: 1) Array of light sources, 2) seafloor height map [40]–[42], 3) objects with associated location & orientation parameters, 4) a camera descriptor specifying an overhead view and orthographic projection. The items are placed in the scene in correspondence to a typical SAS geometry with the sonar altitude operating at one-tenth the maximum imaging range. The array of light sources is necessary to mimic the illumination given by the synthetic aperture geometry. The camera is positioned directly above the center of the seafloor scene consistent with the viewpoint of synthetic aperture imagery. Figure 1 shows several views of the simulated scene for reference, as well as the scene rendered from the viewpoint mimicking SAS imagery. Notice how the SAS realism is low for these images, including the absence of effects such as speckle. This motivated us to explore data-driven methods for SAS image generation.

B. SAS Image Rendering using GANs

To achieve SAS realism for our generated images, we turn to recent advances in machine learning, namely generative adversarial networks (GANs). As mentioned in Section II-A, GANs allow for high photorealism for optical images, although they lack control over scene content due to their probabilistic nature of sampling from the latent space. To test the effectiveness of GANs for SAS image generation, we use the common DCGAN architecture [43] trained on our SAS dataset using the Wasserstein with gradient penalty value function [18]. Our DCGAN learns to generate SAS images of objects on the seafloor with the visual qualities of our training dataset.

In Figure 2, we show the results of training DCGAN on real SAS images. We observe that generated images contain only objects with positions and rotations found in the training set. We note that more advanced GAN architectures that employ progressive growing strategies [12], [16] can further improve SAS realism, but there are no practical ways to control the scene content such as location of the objects in the scene or the shadows.

C. Hybrid POV-RAY to WGAN Architecture

Our main contribution in this paper is the proposed novel pipeline that synergistically combines POV-Ray and GANs in order to fabricate SAS images. In Figure 3, we detail our proposed pipeline where POV-Ray generates synthetic SAS images, and then a GAN improves these SAS images to reflect the statistics and visual quality of real SAS images. This solution allows for fast rendering of SAS images, with high levels of SAS realism, while allowing the user to control the geometry of the model, its location and orientation along the seafloor, and the placement of the sources in the scene. In other words, our GAN is conditioned (similar to a conditional GAN [21], [23]) on the POV-Ray image and preserves scene parameters such as target geometry while updating the SAS image realism of the background and scattering.

One of the challenges of our application domain as compared to rendering natural images is the lack of data to train our data-driven models. To overcome this, we introduce a feature extraction step using an autoencoder within our pipeline that effectively allows semantic features such as target geometry and orientation to flow from the POV-Ray rendered image to the final output.

Pipeline. We mathematically formulate our pipeline as follows. Let ρ_p be the distribution of POV-Ray rendered SAS images, and ρ_r the distribution of real SAS images. We seek a function $G : \rho_p \rightarrow \hat{\rho}_p$ that transforms the rendered image distribution such that the statistical distance between $\hat{\rho}_p$ and ρ_r is minimized. This in effect enforces SAS image realism in our pipeline. Since we do not know our image distributions ρ explicitly, we lack a way to directly find an optimal function for G. However, using only the rendered images and real image samples we have available, the Wasserstein GAN training routine allows us to indirectly minimize the distance between these two distributions by finding a G that satisfies the Kantorovich-Rubinstein duality [17].

To ensure our output images still preserve the target geometry and other scene parameters, we define ϕ to be a feature extractor that maintains $|\phi_p - \hat{\phi}_p| < \gamma$, where γ is a tuneable hyperparameter, and $\phi_p, \hat{\phi}_p$ represent the high-level scene descriptors (object position, seafloor style, etc.) of our rendered and transformed images respectively. To enforce the constraint for ϕ, we extract feature vectors from a convolutional autoencoder (AE) trained on ρ_r. Our convolutional autoencoder is a neural network tasked with reducing each high dimensional image in ρ_r into a smaller dimensional column vector. To most effectively compress the dimensionality of the ρ_r images, our convolutional AE learns a set of feature filters that capture the salient properties of these images. We constrain the AE to reducing images into a dimension that contains enough information to preserve the global properties of the image, such as scene descriptors like object placement and shadows, but allows higher order effects necessary for SAS realism to be determined by the GAN. Since our autoencoder learns features from ρ_r, we are biased toward preserving "realistic" scene parameters between $\hat{\rho}_p$ and ρ_p.
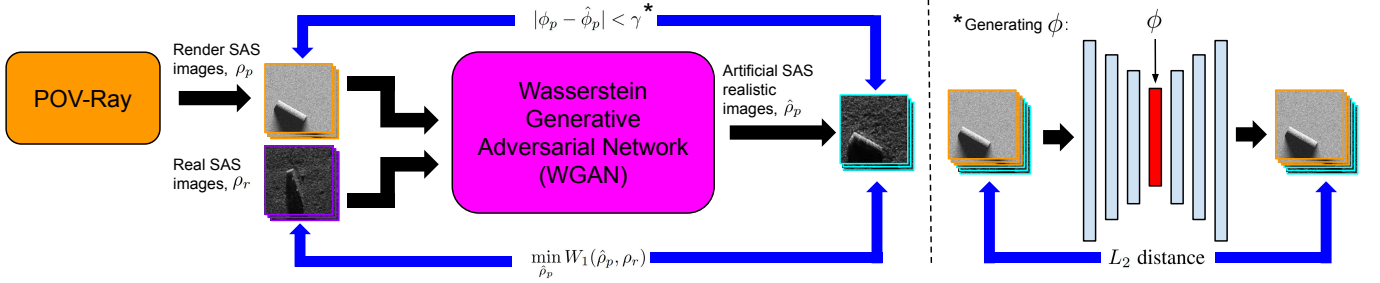
Fig. 3: Our proposed POV-Ray to GAN pipeline for generating artificial SAS images. The left side of the dashed line shows our image generation pipeline. The top equation shows our constraint for maintaining scene descriptors between rendered and transformed images. The bottom equation shows our minimization of the Wasserstein-1 distance between ρ_r and $\hat{\rho}_p$. The right side of the dashed line illustrates generating ϕ from our autoencoder that was trained on ρ_r.

Network Architecture. We adopt a similar generator architecture to [24] of a series of downsampling convolution layers, a series of residual layers, and a series of upsampling+convolutional layers. We use the discriminator network from DCGAN which consists of a series of downsampling convolutional layers. We train over a value function that minimizes the Wasserstein-1 distance between ρ_r and $\hat{\rho}_p$. The value function,

$$\min_G \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{x \sim \rho_r} [D(x)] - \mathop{\mathbb{E}}_{\hat{x} \sim \hat{\rho}_p} [D(\hat{x})] - \\ \lambda \mathop{\mathbb{E}}_{\hat{x} \sim \rho_{\hat{x}}} [(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2], \quad (1)$$

is created from the Kantorovich-Rubinstein duality as shown in [17], [18].

The Wasserstein-1 distance between ρ_r and $\hat{\rho}_p$ is minimized when we find a D network, which is encouraged to be 1-Lipschitz smooth by the gradient penalty term, that maximizes this value function. Here, $\hat{x} \sim \hat{\rho}_p$ are generated images from the distribution created through $G : \rho_p \rightarrow \hat{\rho}_p$, and $\hat{x} \sim \rho_{\hat{x}}$ are randomly sampled generated samples for calculating the gradient norm of D.

In order to obtain a set of filters for extracting ϕ, we train our autoencoder using a series of four downsampling covolutional layers, each followed by ReLU activations and max-pooling, to compress our 256×256 input images into a 1024-dimensional column vector, ϕ, and then four nearest-neighbor interpolated upsamplings followed by convolutions to reconstruct the image from the compressed vector. After training, we disregard the upsampling layers and use the downsampling layers to generate a 1024-dimensional ϕ vector for a given image. We train our autoencoder over an L_2 distance between input and reconstructed images.

IV. EXPERIMENTAL RESULTS

We present experimental work to illustrate the potential quality of SAS GAN with respect to genuine sonar image data. Our hybrid pipeline produces fabricated SAS images with realistic SAS-image characteristics. We also show control over the target and source geometry and location in the scene due to POV-Ray. To analyze our results, we employ quantitative

metrics to relate fabricated SAS image quality fidelity to genuine SAS data, and discuss challenges of evaluating results for data-driven methods for this application domain.

A. Implementation Details

SAS Data. Our genuine SAS data used in this section comes from the MUSCLE autonomous underwater vehicle courtesy of the NATO Centre for Marine Research & Experimentation. This system has a high-frequency SAS developed by Thales with a center frequency of 300 kHz and bandwidth of 60 kHz. The imagery from MUSCLE has approximate resolutions of 2.5 cm and 1.25 cm in the along-track and across-track directions, respectively, and can reach out to 150m in range [10]. We note that we use only 560 images containing cylindrical objects in this dataset, which is very data-starved for deep learning applications. The images contained a diversity in both background types and target orientation, though we note that all the targets were centered in their images as this will be important later.

POV-Ray specifications. We render cylindrically shaped targets with a size approximated to the targets in the dataset. These targets are rendered with a rough surface on a seabed of Gaussian distributed pixel values. The noise on the targets and backgrounds encourages stochasticity in generated samples.

Training details. We train our SAS GAN pipeline on 2 Titan-X GPUs for 24 hours with 560 real images and 850 POV-Ray images. All images are sized 256×256 pixels. We optimize the Wasserstein with gradient penalty value function shown in Equation 1 (with gradient penalty set to 10) using the Adam optimizer with a learning rate of 0.001 and batch size of 4.

B. Qualitative Results

Figure 4 displays several POV-Ray input images and the associated SAS GAN output images. SAS GAN captures the image resolution and blurring common to the SAS images by dramatically altering the given POV-Ray rendered inputs. This shows the network is performing a style transfer from POV-Ray images to SAS images while still retaining the target in the image. SAS GAN is acting as intended: it is given a
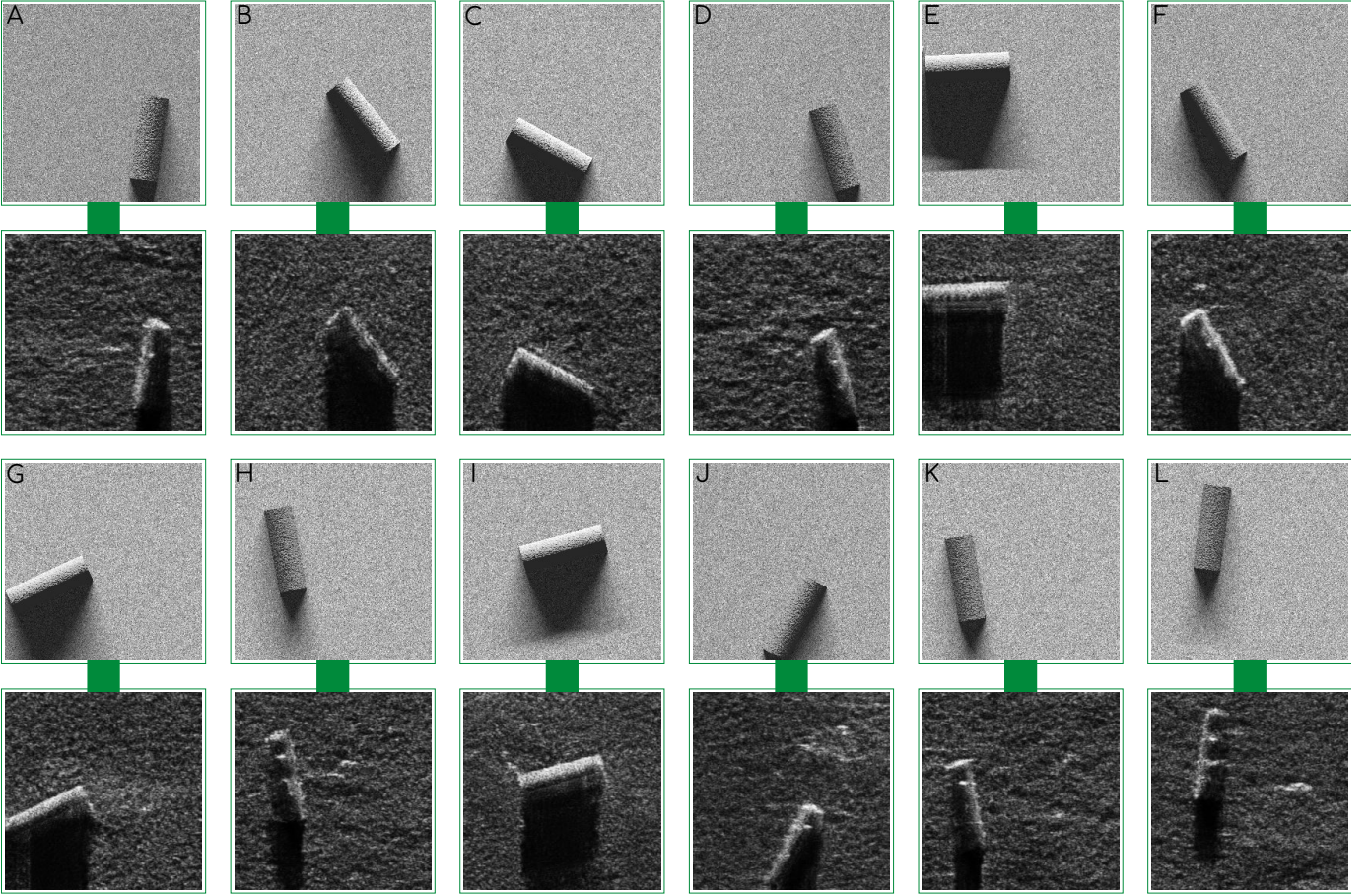
Fig. 4: Example SAS GAN images. The POV-Ray (lettered) inputs are connected by green lines to their associated SAS GAN images.

controlled target input and produces a realistic-looking output with a target in the same position.

Note that SAS GAN learns to alter the shadow pattern of the input render. In each case of Figure 4, the given POV-Ray shadow is altered to match shadows more characteristic of SAS images, namely by becoming longer and darker. This better aligns with shadows that would be formed from targets at these type of distances away from the vehicle.

Along with adjusting the shadow, SAS GAN has learned to generate fine-grained details visible on targets. Several targets exhibit strong circular patterns on top of the cylinder. These details are learned from the MUSCLE system and relate to realistic target shapes. SAS GAN produces these realistic targets *without instruction* (i.e. a regularization technique that encourages such behavior). SAS GAN is not just learning a simple style transfer of where to place shadows or background texture; it is actively figuring out what a target is. We argue that our qualitative results are compelling and have high SAS realism; this was informally confirmed by several SAS researchers and engineers who viewed the images.

Novelty of generated images. One of the difficulties of training in a data-starved regime is that the GAN can learn to identically copy the training set and reproduce those images

at test time. This means that the network has not learned to generalize across the image dataset, or not effectively learned the input distribution to sample novel points (images) from that distribution. To ensure that our network is indeed learning the input distribution, we checked the nearest neighbor in both image space (ℓ_2 loss) as well as feature space ϕ to ensure no image was copied.

Figure 5 shows the results of our nearest neighbor experiment. We confirm that SAS GAN is not simply copying the MUSCLE data as the closest real images are markedly different. The nearest neighbor MUSCLE images returned are also cylinder images, however they do not always align with the target geometry present in SAS GAN. Sometimes the nearest neighbor images had unusual artifacts (C) or repetitive images (D-F), potentially caused by measuring distance in feature space ϕ which can overlook such visual inconsistencies.

C. Translation, Rotation, and Background Diversity

One of our stated goals was to allow target control over geometry and location in the image. This is important since the MUSCLE training data did not have all possible orientations of the cylinder and all targets were explicitly centered in the images. To generate novel and diverse images, our hybrid
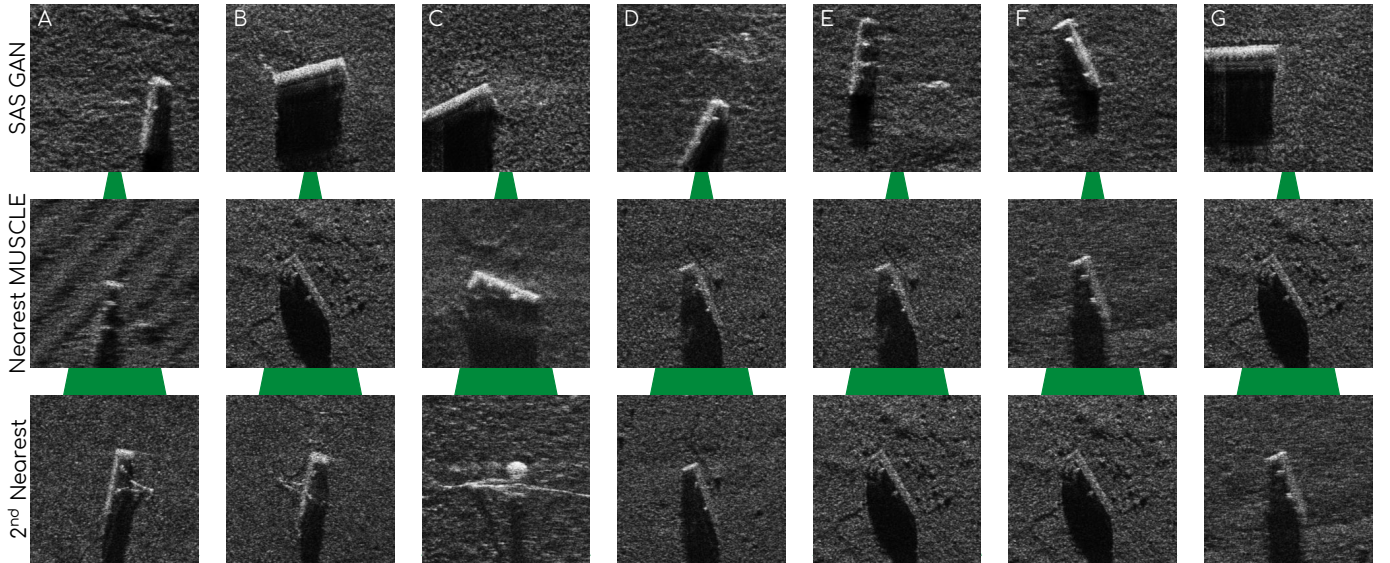
Fig. 5: Nearest neighbor results for a selection of SAS GAN fabrications (top row) and genuine target examples from MUSCLE (bottom three rows) using the autoencoder distance.

pipeline must preserve a POV-Ray input target's location and orientation while increasing the SAS realism of the output.

Figure 6(a-b) shows the results of such translation and rotation experiments respectively, in which the target location in the POV-Ray images was varied. SAS GAN is able to sustain realistic target scenarios throughout these transformations. The translation experiment shows that SAS GAN preserves the target's diagonal movement with little change in the target's appearance. Similarly, the rotational effects on the shadows appear in line with what we would expect. It is interesting to note the case where the cylinder is viewed at endfire; the face nearest the vehicle scatters strongly while the rest of the target blends into the background. This suggests that, while SAS GAN is able to impressively model many different views of the cylinder, this exact case lies on the edge of its capabilities. Either more training is required to solve this or a more nuanced approach is needed. Regardless, the overall performance in both translation and rotation is encouraging.

Another key aspect to SAS GAN's output is the background. The POV-Ray render has the target with shadow encompassed by speckle noise so we do not impart some control over the seafloor like we do the cylinder. This results in the types of textures illustrated in Figure 7. Here we see a few different seafloor types. While the differences can be subtle, like in examples C, D, and E, they are distinct from one another. SAS GAN also learns to put an occasional clutter item, like in G, despite no instruction.

D. Quantitative Results

One of the challenges of evaluating data-driven methods for image generation is the lack of suitable quantitative metrics to evaluate image quality. This difficulty has been noted in multiple domains, spurring work on reference and reference-free visual quality assessment [44]. However, we still evaluate

MUSCLE→MUSCLE	$2.324 \pm .038$
SAS GAN→MUSCLE	$2.934 \pm .046$
DCGAN→MUSCLE	$2.289 \pm .035$
POV-Ray→MUSCLE	$14.472 \pm .065$

TABLE I: FID Scores. The MUSCLE to MUSCLE score serves as a reference score to give context to the DCGAN, POV-Ray, and SAS GAN scores.

some common metrics for GAN performance to elucidate some insights into the network's behavior.

FID score. To quantitatively evaluate GAN output quality, Heusel et al. proposed the Fréchet inception distance (FID) in [45], also known as the 2-Wasserstein distance. As this metric has been widely adopted in the literature [12], [16], [46], we use a variant on this score for providing a quantitative metric comparing the similarity of the DCGAN, SAS GAN, and POV-Ray distributions. Given two datasets of images A and B, the proposed FID metric uses an intermediate layer from Inception network [47] pretrained on ImageNet [48] images to capture the average feature vectors, ϕ_A, ϕ_B, and covariance matrices C_A, C_B over all images $i \in A, B$. Then if we use these mean vectors and covariance matrices to define $\rho_A = \mathcal{N}(\phi_A, C_A)$ and $\rho_B = \mathcal{N}(\phi_B, C_B)$ as multivariate Gaussian distributions, the 2-Wasserstein distance between ρ_A and ρ_B is defined as

$$W_2(\rho_A, \rho_B) = ||\phi_A - \phi_B||_2^2 + \text{Tr}(C_A + C_B - 2(C_B^{\frac{1}{2}} C_A C_B^{\frac{1}{2}})^{\frac{1}{2}}), \quad (2)$$

where Tr is the trace.

Since the ImageNet feature space is not optimal for capturing SAS image features, we instead utilize the features captured from our autoencoder trained on the dataset of MUSCLE images. The results of these experiments are displayed in Table I.

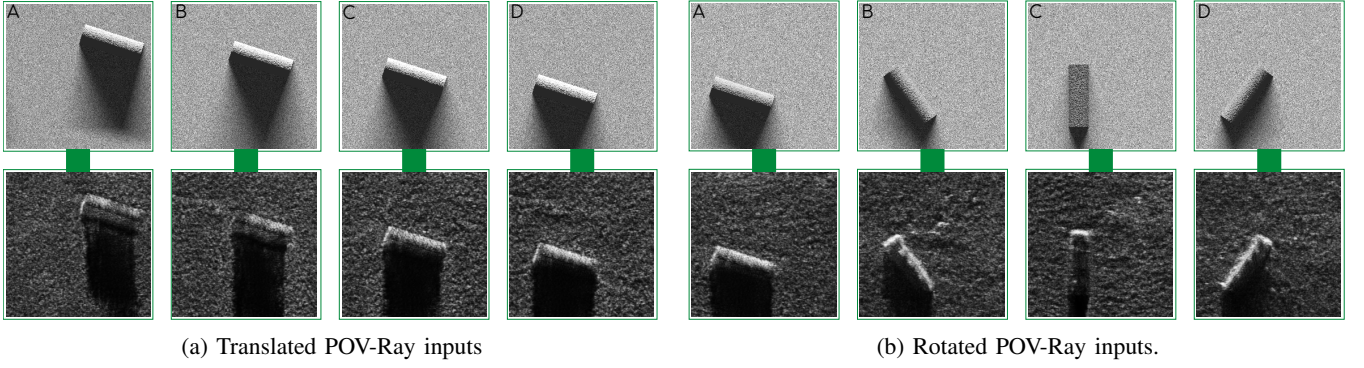(a) Translated POV-Ray inputs (b) Rotated POV-Ray inputs.

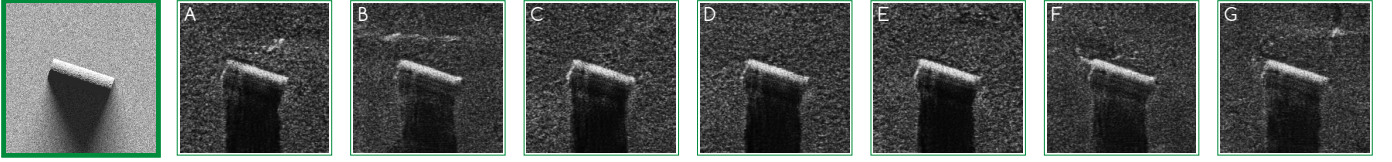Fig. 6: Example SAS GAN images for translated (a) and rotated (b) POV-Ray inputs.

Fig. 7: Example seafloor generations for a given target. The far left image is an example POV-Ray input and the rest are SAS GAN fabrications. Note that the only difference to the SAS GAN's input to generate these different backgrounds was the noise outlining the POV-Ray target.

We first note that the DCGAN to MUSCLE score is actually lower than the baseline reference score of MUSCLE to MUSCLE. This phenomena hints at a limitation of using FID score; it cannot filter out GANs which replicate the training dataset since two identically copied datasets will achieve the best possible FID score of zero. The FID from DCGAN to MUSCLE is slightly lower than the FID from SAS GAN to MUSCLE, and both distances are significantly lower than POV-Ray to MUSCLE. We suspect SAS GAN has a higher FID than DCGAN because SAS GAN has the ability to generate objects in positions and orientations not found in the MUSCLE dataset. In contrast, DCGAN generates objects only in positions and orientations that can be interpolated from images found in the MUSCLE dataset. Thus we hypothesize SAS GAN is actually augmenting the MUSCLE distribution, rather than simply replicating it like DCGAN, and therefore scoring technically worse when its FID score is measured. To obtain more tangible evidence for this hypothesis, we utilized a non-linear dimensionality reduction technique called t-SNE [49] to visualize image distribution structure.

t-SNE. t-distributed Stochastic Neighbor Embedding (t-SNE) is a well-established dimensionality reduction technique that summarizes high dimensional data in a digestible Cartesian grid [49]. t-SNE offers insight into the geometries of globally non-linear data, but since its Cartesian mappings are obtained through the non-deterministic minimization of a non-convex function, it is important to consider its results carefully. In Figure 8, we plot the t-Distributed Stochastic Neighbor Embedding (t-SNE) feature space of both the GAN models, MUSCLE images, and POV-Ray renders when they are passed through the autoencoder features ϕ.

We will walk the reader through Figure 8, point out relevant features, and explain how they lend insight into the performance of the different generated images. Looking at the MUSCLE (purple) dataset, we see they form a ring structure. Notice as one goes from MUSCLE images 1-6 clockwise, the t-SNE position of the image correlates with the target's orientation at the center of a tile. This shows that the autoencoder features semantically disentangled the latent space and learned target orientation as a scene parameter. This is later exploited by our novel ϕ loss. Thus, our t-SNE analysis shows that the MUSCLE dataset (1) has a rotational structure in the latent space of the ϕ and (2) has no off-center targets.

Looking at the POV-Ray images in yellow, we notice they cluster in t-SNE away from the MUSCLE dataset, and qualitatively the images do not have the same distributional statistics as the more realistic SAS images of SAS GAN and DC GAN (evidenced by the large FID score). However, POV-Ray images do include both centered and off-centered targets.

For DCGAN (red), we show that they also cluster near the MUSCLE dataset and exhibit the same rotation in their latent space. Note however that DCGAN images sampled near the center (e.g. DCGAN 6) are of lower quality because the network is trying to generate novel images in a region for which there are no nearby MUSCLE images.

Finally, we analyze the results of our SAS GAN (light blue). Note SAS GAN images that cluster near the MUSCLE data are qualitatively of the same SAS realism. But, SAS GAN also generates a new cluster of images on the right (e.g. SAS GAN 7-12), showing that the network can generalize and extrapolate beyond the training set to create novel images. However, we do observe a limitation: not all of these sampled images are
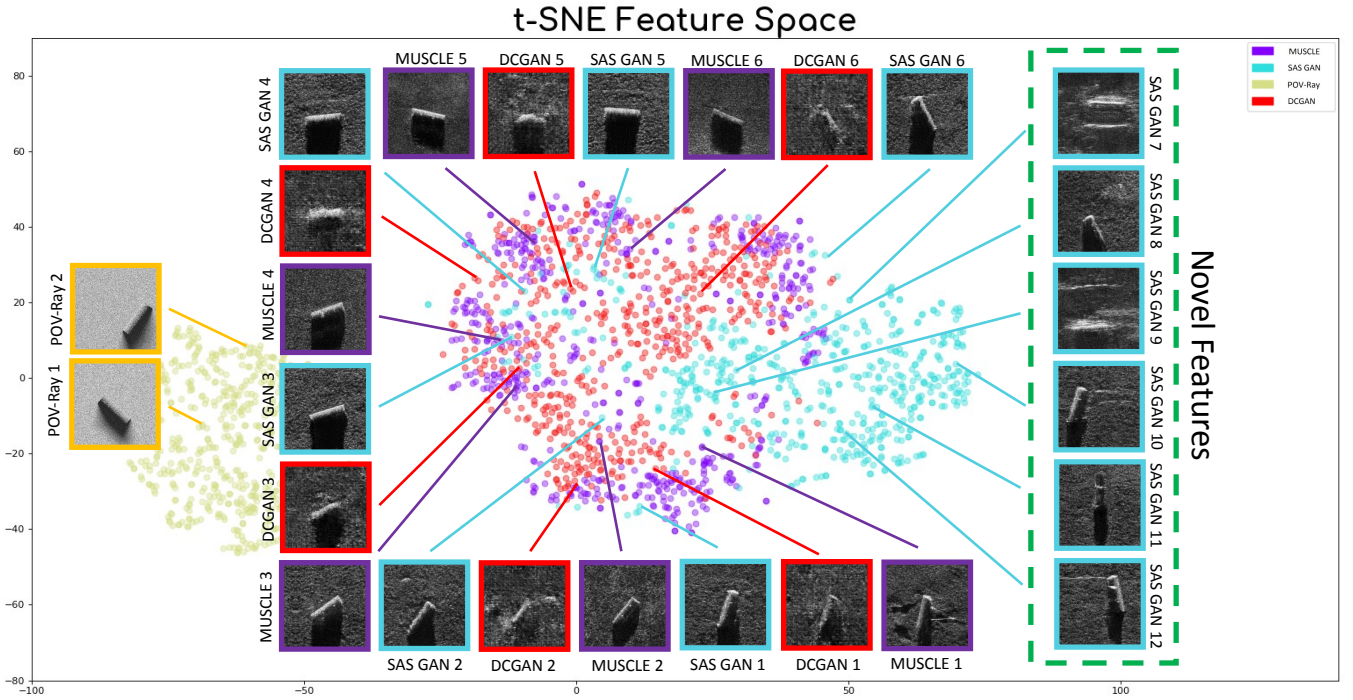
Fig. 8: t-SNE visualization of the feature space spanned by real MUSCLE images, POV-Ray renders, and fabricated imagery generated by both DCGAN and our SAS GAN. Images are numbered for easy reference.

of high SAS realism. But the new cluster does exhibit off-centered images and varying target orientations, which are not possible for DCGAN and not present in the real MUSCLE dataset. We argue that this new cluster augments the MUSCLE distribution with novel images and explains the discrepancy in FID scores between DCGAN and SAS GAN.

V. DISCUSSION

Our hybrid approach coupling optical rendering with GANs yields several interesting discussion points and avenues for future investigation. As validated in our experimental results, we achieve high levels of SAS image realism while enabling control over scene geometry and parameters. Further, our pipeline achieves fast rendering times at test time: we can render a single SAS image in 250 milliseconds, and thus approximately 900 images in one hour. This is useful for dataset augmentation for data-starved tasks such as ATR.

There are several limitations to our approach. In Figure 9, we show some failure cases of our GAN output. These include effects unrealistic shadow geometries and orientations, and unrealistic visual artifacts on the targets. Some of these errors could be mitigated by a larger, well-curated SAS image dataset, which would benefit machine learning approaches.

There are several directions for future reasearch. To achieve better physical realism (as opposed to SAS image realism), we can condition the GAN on input seeded from an acoustic simulator such as point-based scattering models [8]. Further work is needed to quantify the performance of GAN outputs for SAS images. Finally, fully completing the pipeline and
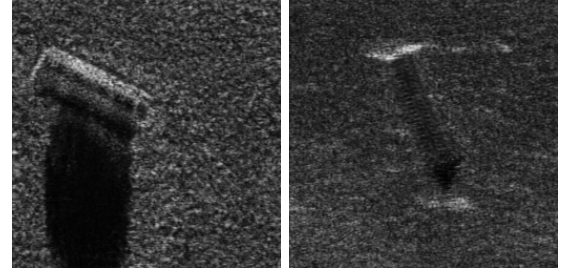

Fig. 9: Two examples of failure cases: The left image shows a cylindrical object with an unrealistic shadow. The right image shows a poorly generated object with repeating artifacts and unrealistic glint lines.

evaluating the level of physical realism and control of scene parameters/content generation needed for SAS ATR (similar to [28]) would help show the advantages of hybrid pipelines.

REFERENCES

[1] J. Stack, "Automation for underwater mine recognition: current trends and future strategy," in *SPIE Defense, Security, and Sensing*, vol. 8017, International Society for Optics and Photonics, 2011.

[2] D. P. Williams, M. Couillard, and S. Dugelay, "On human perception and automatic target recognition: Strategies for human-computer cooperation," in *2014 22nd International Conference on Pattern Recognition*, pp. 4690–4695, Aug 2014.

[3] D. P. Williams, "Underwater target classification in synthetic aperture sonar imagery using deep convolutional neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2497–2502, Dec 2016.

[4] M. Mignotte, C. Collet, P. Perez, and P. Bouthemy, "Sonar image segmentation using an unsupervised hierarchical mrf model," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1216–1231, 2000.

[5] T. E. Blanford, D. C. Brown, and R. J. Meyer Jr, "Design considerations for a compact correlation velocity log," in *Proceedings of Meetings on Acoustics 175ASA*, vol. 33.

[6] Y. Doisy, "General motion estimation from correlation sonar," *IEEE Journal of Oceanic Engineering*, vol. 23, no. 2, pp. 127–140, 1998.

[7] A. Bellettini and M. A. Pinto, "Theoretical accuracy of synthetic aperture sonar micronavigation using a displaced phase-center antenna," *IEEE Journal of Oceanic Engineering*, vol. 27, no. 4, pp. 780–789, 2002.

[8] D. C. Brown, S. F. Johnson, and D. R. Olson, "A point-based scattering model for the incoherent component of the scattered field," *J. Acoust. Soc. Am.*, vol. 141, no. 3, pp. EL210–EL215, 2017.

[9] G. S. Sammelmann, "Propagation and scattering in very shallow water," in *MTS/IEEE Oceans Conference*, vol. 1, pp. 337–344 vol.1, Nov 2001.

[10] D. P. Williams, "A novel framework for evaluating performance-estimation models," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, pp. 5285–5302, Aug 2019.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[12] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[13] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[14] N. Kato, H. Osone, K. Oomori, C. W. Ooi, and Y. Ochiai, "GANs-based Clothes Design: Pattern Maker Is All You Need to Design Clothing," in *Proceedings of the 10th Augmented Human International Conference 2019*, AH2019, ACM, 2019.

[15] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in neural information processing systems*, pp. 2234–2242, 2016.

[16] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

[17] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv e-prints*, p. arXiv:1701.07875, Jan 2017.

[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, pp. 5767–5777, 2017.

[19] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, pp. 2172–2180, 2016.

[20] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, "Gan dissection: Visualizing and understanding generative adversarial networks," *arXiv preprint arXiv:1811.10597*, 2018.

[21] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[22] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[23] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[24] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," *European Conference on Computer Vision (ECCV)*, pp. 694–711, 2016.

[25] J. L. Chen and J. E. Summers, "Deep neural networks for learning classification features and generative models from synthetic aperture sonar big data," in *Proceedings of Meetings on Acoustics 172ASA*, vol. 29, p. 032001, ASA, 2016.

[26] S. Lee, B. Park, and A. Kim, "Deep Learning from Shallow Dives: Sonar Image Generation and Training for Underwater Object Detection," *arXiv e-prints*, p. arXiv:1810.07990, Oct 2018.

[27] M. Sung, H. Cho, J. Kim, and S. Yu, "Sonar image translation using generative adversarial network for underwater object recognition," in *2019 IEEE Underwater Technology (UT)*, pp. 1–6, April 2019.

[28] A. I. Karjalainen, R. Mitchell, and J. Vazquez, "Training and validation of automatic target recognition systems using generative adversarial networks," in *2019 Sensor Signal Processing for Defence Conference (SSPD)*, pp. 1–5, May 2019.

[29] G. S. Sammelmann, "Computer-readable software and computer-implemented method for performing an integrated sonar simulation," Aug. 1 2000. US Patent 6,096,085.

[30] G. S. Sammelmann, "Propagation and scattering in very shallow water," in *MTS/IEEE OCEANS Conf.*, (Honolulu, HI), pp. 337–344, Nov. 2001.

[31] G. S. Sammelmann, "High frequency images of proud and buried 3D-targets," in *MTS/IEEE OCEANS Conf.*, vol. 1, (San Diego, CA), pp. 266–272, Sep. 2003.

[32] P. A. M. De Theije and H. Groen, "Multistatic sonar simulations with SIMONA," in *International Conference on Information Fusion*, pp. 1–6, July 2006.

[33] A. J. Hunter, *Underwater Acoustic Modeling for Synthetic Aperture Sonar*. PhD thesis, University of Canterbury, Christchurch, NZ, Jun. 2006.

[34] A. T. Abawi, "Kirchhoff scattering from non-penetrable targets modeled as an assembly of triangular facets," *J. Acoust. Soc. Am.*, vol. 140, no. 3, pp. 1878–1886, 2016.

[35] M. Chen, S. Zhang, and J. Tang, "An interferometric synthetic aperture sonar raw signal simulation based on points-scatterer model," in *Joint Conference on Computational Sciences and Optimization*, vol. 1, pp. 367–369, April 2009.

[36] P. Y. Mignotte, J. Vazquez, J. Wood, and S. Reed, "PATT: A performance analysis and training tool for the assessment and adaptive planning of Mine Counter Measure (MCM) operations," in *MTS/IEEE OCEANS*, pp. 1–10, Oct 2009.

[37] J. T. Kajiya, "The rendering equation," *SIGGRAPH Comput. Graph.*, vol. 20, pp. 143–150, Aug. 1986.

[38] Persistence of Vision Pty. Ltd., "Persistence of vision (tm) raytracer." https://www.povray.org/download.

[39] D. Köhntopp, *Shape-based Machine Perception of Man-Made Objects on Underwater Sensor Data*. PhD thesis, Jacobs University Bremen, 2018.

[40] S. F. Johnson, *Synthetic aperture sonar image statistics*. PhD thesis, The Pennsylvania State University, 2009.

[41] D. Tang, F. S. Henyey, B. T. Hefner, and P. A. Traykovski, "Simulating realistic-looking sediment ripple fields," *IEEE Journal of Oceanic Engineering*, vol. 34, no. 4, pp. 444–450, 2009.

[42] S. F. Johnson and D. C. Brown, "SAS simulations with procedural texture and the point-based sonar scattering model," in *MTS/IEEE OCEANS Conf.*, pp. 1–7, Oct 2018.

[43] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[44] W. Lin and C.-C. J. Kuo, "Perceptual visual quality metrics: A survey," *Journal of Visual Communication and Image Representation*, vol. 22, no. 4, pp. 297–312, 2011.

[45] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

[46] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[49] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.