AAS 20-610

GENERATIVE MODEL FOR SPACECRAFT IMAGE SYNTHESIS USING LIMITED DATASET

Tae Ha Park*, Simone D'Amico†

This work presents for the first time a conditional Generative Adversarial Network (GAN) to sample arbitrary high-fidelity spacecraft images from the learned distribution of spacecraft texture and illumination conditions (i.e., *styles*). The proposed SPEEDGAN utilizes a low-texture template of the spacecraft to empower SPEEDGAN with a priori knowledge of the spacecraft geometry and pose, allowing the model to focus on creating a spacecraft style. The SPEEDGAN also trains with a content loss from style transfer literature to improve the structural fidelity of the generated spacecraft images. Trained on a limited dataset containing images from the computer graphics renderer and the hardware-in-the-loop simulation facility, SPEEDGAN generates samples with remarkable visual qualities, measured by both human and a separate neural network for pose estimation. It is also capable of separated control over different style aspects, such as spacecraft texture and illumination effects.

INTRODUCTION

Some of the most exciting mission concepts in development are on-orbit servicing and active debris removal missions, such as the RemoveDEBRIS mission by Surrey Space Centre,[1] the Phoenix program by DARPA,[2] the Restore-L mission by NASA,[3] and GEO servicing programs proposed by startup companies such as Infinite Orbits* and Effective Space†. In these missions, the on-board capability of estimating and tracking the relative pose (i.e., position and orientation) of a non-cooperative target spacecraft is crucial to the autonomous generation of the servicer's trajectory and control sequences. Unlike stereovision or light detection and ranging (LIDAR), a monocular camera is a natural choice of sensor due to its low mass and power requirements suitable for small spacecraft such as CubeSats. Therefore, extensive body of works has been dedicated to monocular-based pose estimation architectures.[4–12]

Inspired by computer vision methods in terrestrial applications, traditional spacecraft pose estimation architectures rely on feature extraction and correspondence. Specifically, in the absence of a coarse *a priori* pose knowledge[4,5] or known fiduciary markers on the target,[6] robust and accurate detection of edge[13] or point[14] features is critical to an algorithm which iteratively finds the pose solution with the best feature correspondence.[15,16] Unfortunately, classical image processing-based feature extraction mechanisms are likely to detect partial or spurious features on spaceborne images due to a number of challenges unique to space environment, such as adverse illumination condition, high contrast, low signal-to-noise ratio, and the Earth background. Moreover, feature matching is

*Ph.D. Candidate, Department of Aeronautics & Astronautics, Stanford University, Stanford, CA 94305.
†Assistant Professor, Department of Aeronautics & Astronautics, Stanford University, Stanford, CA 94305.

*https://www.infiniteorbits.io/

†https://www.effective.space/

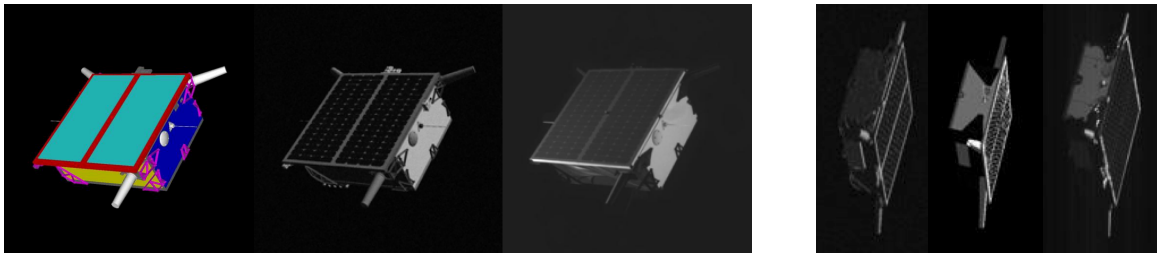

Figure 1. Various images of the Tango spacecraft from PRISMA mission. (*Left*) Low-texture template, SPEED synthetic, and SPEED real images. (*Right*) Images from SPEED synthetic, Imitation-25,[20] and PRISMA-25 datasets. PRISMA-25 contains flight images captured during the rendezvous phase of the PRISMA mission.

expensive due to a large search space of unknown correspondences and limited on-board processing power.[17]

Recently, Convolutional Neural Networks (CNN) have become an attractive alternative to replace a part of or the entire pose estimation architecture. Since CNN is simply a data-driven nonlinear function trained from a set of input-output pairs, its flexibility allows vastly different types of CNN architectures in both terrestrial and spaceborne applications. For instance, PoseCNN[18] maps an input image to a 4D unit quaternion and 3D translation vector via separate CNN branches in an end-to-end fashion, whereas Park et al.[11] maps an image cropped around the spacecraft to a vector containing 2D locations of a set of N pre-selected spacecraft corners. CNN predicts these corners in a deterministic order; therefore, the 2D-3D feature correspondence is automatically established, requiring only a single iteration of an existing feature matching algorithm such as EPnP[15] to compute the complete 6D pose solution. PVNet[19] also estimates object keypoints by predicting pixel-wise unit vectors towards each keypoints. Not only are keypoint locations selected via a RANSAC-like voting scheme of these vectors, they also admit uncertainty in prediction which can be accounted for in feature matching algorithm.

Due to a severe lack of a large-scale flight imagery of the target spacecraft with accurately annotated pose labels, any spaceborne CNN must be trained almost exclusively on images generated via computer graphics (i.e., *synthetic*) and/or the ground-based hardware-in-the-loop testbed (i.e., *real*). The Spacecraft Pose Estimation Dataset (SPEED)[21] is the first publicly available dataset comprising 15,300 synthetic and real images of the Tango spacecraft from the PRISMA mission.[7,22] It consists of 15,000 synthetic images from the OpenGL-based renderer and 300 real images of a 1:1 mockup spacecraft model captured from SLAB's Testbed for Rendezvous and Optical Simulation (TRON) facility. SPEED served as the dataset for the Satellite Pose Estimation Competition (SPEC) co-organized by the Space Rendezvous Laboratory (SLAB) of Stanford University and the Advanced Concepts Team of the European Space Agency.[12] The top-performing entries were able to achieve sub-degree orientation and sub-millimeter position accuracies on the synthetic test set.

However, the major drawback of relying on synthetic imagery to train a pose estimation CNN is that it has visual qualities (e.g., spacecraft texture and illumination effects) that are different from those of other image domains (see Figure 1), inevitably overfitting the model to the features unique to the synthetic images. This leads to worse performances on real images[12] and on PRISMA flight images.[10,11] Interestingly, Sharma[10] points out in Figure 2 that the Spacecraft Pose Network (SPN)[9] trained on synthetic images show comparable performance trends when tested on SPEED real images and PRISMA flight images, suggesting the real images from the TRON facility can be used as

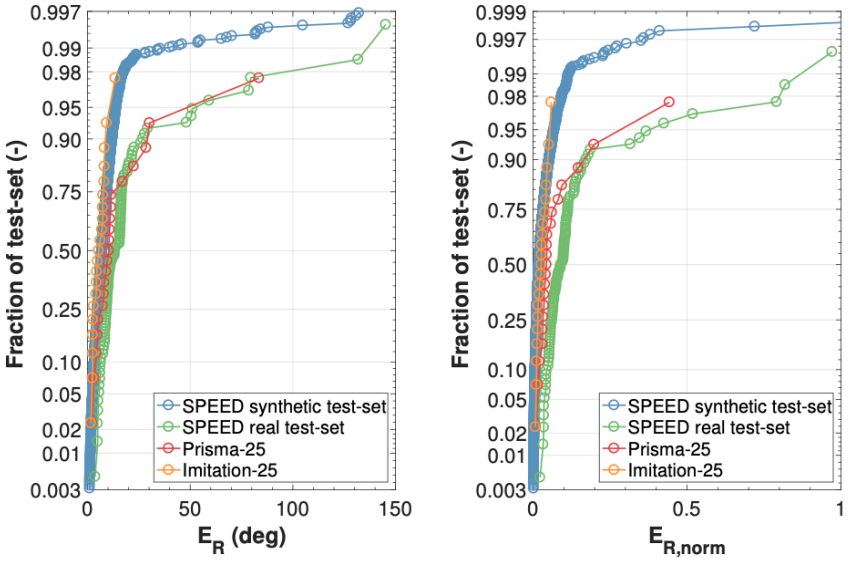

Figure 2. (*Left*) Cumulative distribution of rotation error of SPN model[9] when tested on SPEED synthetic test set, real test set, PRISMA-25, and Imitiation-25 datasets. (*Right*) Same distribution of the rotation error normalized by inter-spacecraft separation and camera field-of-view. Figures taken from Sharma.[10]

a surrogate of flight images when validating a pose estimation CNN's performance. This also implies that the real images can be used for training a CNN alongside the synthetic images in order to improve its performance on flight images. However, as evidenced by the proportion of the number of images in SPEED, currently available real images are severely limited in terms of quantity and the variability of spacecraft pose and illumination conditions. Moreover, a hardware-in-the-loop facility inherently requires much time and effort to capture and calibrate tens of thousands of real images. Therefore, a method to efficiently generate a large number of real images based on a limited amount is desirable.

In this paper, a deep generative model is proposed to respond to the aforementioned challenges. Specifically, this work introduces SPEEDGAN, a conditional Generative Adversarial Network (cGAN) to efficiently sample the high-fidelity spacecraft images with specified pose and arbitrary lighting and texture typical to both synthetic and real datasets. The architecture of SPEEDGAN is based on StyleGAN.[23] However, a number of modifications is made in order to ensure the structural fidelity and pose of the generated spacecraft images. First, the SPEEDGAN generator takes as an additional input a low-texture template image of the spacecraft (see Figure 1), effectively *conditioning* the generator with *a priori* knowledge of the spacecraft pose and geometry. This also allows the user to assign pose labels to the generated samples. Second, the SPEEDGAN generator training is regularized via the content loss[24] to ensure the generated image retains the same *content* as the template. It is shown that including content loss in the generator's training objective improves the training and the generated sample's visual quality as measured by the Fréchet Inception Distance[25] score. Owing to the innovation of the baseline StyleGAN's architecture and regularization technique, SPEEDGAN is also able to control spacecraft texture and illumination condition separately. Lastly, a pose estimation CNN by Park et al.[11] is shown to perform well on samples generated by SPEEDGAN's samples, suggesting they retain features useful for pose estimation.

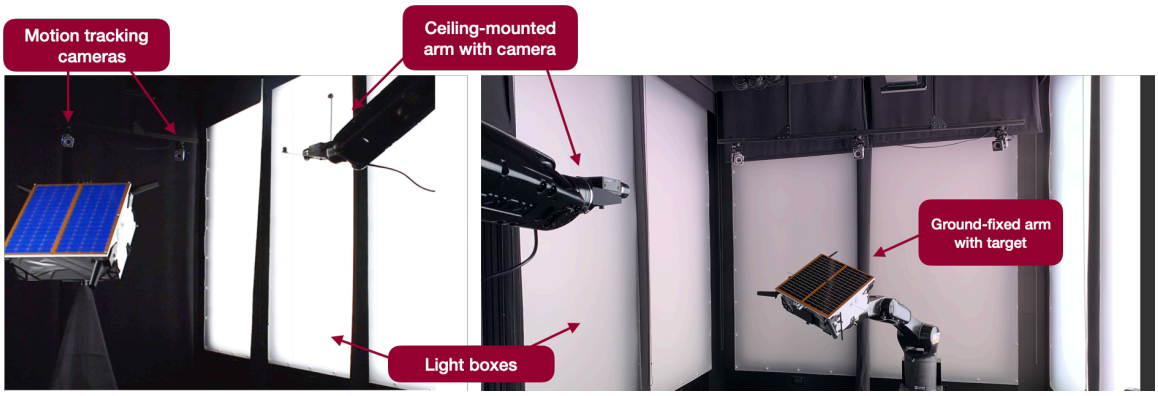

Figure 3. (*Left*) Early TRON facility with 1:1-scale mockup model of the Tango spacecraft. (*Right*) Updated TRON facility with a ground-fixed robot arm and a lightweight 1:2-scale mockup model.

This paper is organized as follows: the next section provides backgrounds on generative aversarial networks and style transfer, two instances of generative models that motivate the SPEEDGAN's basic architecture and implementation. The section afterwards elaborates on the SPEEDGAN's architecture and the training objective, which is followed by a section on the training dataset, implementation details, and a metric to evaluate the generator's outputs. The paper ends with some analyses of SPEEDGAN's generated samples and conclusion with future directions.

BACKGROUND

This section provides a brief background of the sources of SPEED dataset: the Optical Stimulator (OS)[20,26] camera emulator software and the TRON facility at SLAB. These toolsets are utilized to generate the dataset used in this paper. Then, a brief introduction of Generative Adversarial Networks (GAN) and style transfer are given, both of which form a foundation of the StyleGAN[23] architecture heavily used by SPEEDGAN.

Optical Stimulator (OS)

The OS software use MATLAB and C++ language bindings of OpenGL to render non-stellar objects (e.g., satellites, asteroids, planets, etc.) of arbitrary pose. The OpenGL's frament shader allows to render realistic illumination effect on the object's surface based on the object's pose with respect to multiple light sources with different radiometric intensity, such as sun and Earth albedo. SPEED's synthetic imagery[9] is created by selecting the direction of solar illumination that best matches the lighting in the Earth images captured by the Himawari-8 geostationary meteorological satellite*. The OS software is also capable of rendering stellar objects and non-stellar objects at far-range with accurate radiometric properties. Therefore, it has found applications beside close-range pose estimation, such as rendering images in hardware-in-the-loop fashion to validate far-range angles-only navigation algorithms.[26,27]

TRON

The early TRON space simulator room ($8 \times 3 \times 3$ m) at SLAB featured a 6DOF Kuka robotic arm, which is mounted to the ceiling and translates on a track at speeds up to 1.5m/s (see Figure

* https://himawari8.nict.go.jp/

3, left). The facility included a 1:1-scale mockup model of the Tango spacecraft fixed on a tripod. Special low reflectance materials, sun simulators, and custom light boxes allow the reproduction of illumination conditions encountered in space navigation. The ground-truth pose of the objects (camera and scene) is provided by a Vicon 10-units camera system which tracks infrared markers attached to both mockup and camera. This facility was used to create SPEED real images and also part of the dataset used in this work to train SPEEDGAN.

Recently, the TRON facility has undergone a significant upgrade to include a second 6DOF Kuka robotic arm fixed on the ground (see Figure 3, right). This robot holds a lightweight 1:2-scale mockup model, thus allowing the simualtion of full relative orientation distribution and much larger inter-spacecraft separation. With the upgraded software, the more accurate ground-truth pose of the objects will be acquired by data fusion of measurements from the Kuka joints and the Vicon tracking system. While the images from the current facility is not used or presented in this paper, future works on spaceborne CNNs will be trained and validated using those images.

Generative Adversarial Networks (GAN)

A GAN originally proposed by Goodfellow et al.[28] consists of two competing neural networks: a generator (G) which learns to map a low-dimensional latent space to an image, and a discriminator (D) which learns to discern if an image is from the data distributon or the generated distribution. These networks are modeled as CNNs for image inputs.[29] The training of GAN naturally leads to the latent space learning the underlying distribution of the dataset. This is done by training the GAN over the following adversarial loss,

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))], \tag{1}$$

where $\boldsymbol{x}$ is a *real* image drawn from the data distribution p_{data}, and $\boldsymbol{z}$ is a random vector drawn from the known prior p_z such as a standard normal distribution. This is equivalent to minimizing the Jensen-Shannon divergence between the data and the generated distributions. While early GANs suffered from mode collapse and unstable training, novel losses such as minizing Wasserstein distance[30,31] between two distributions have been proposed to stabilize the GAN training. Recently, ProGAN[32] was proposed to efficiently train a GAN to generate high-resolution images (e.g., 1024 $\times$ 1024 pixels) by progressively growing the GAN layers and the sample resolution.

Style Transfer

Style transfer studies how one can alter an image (i.e., content) in the style of another image (e.g., painting). The seminal work of Gatys et al.[33] demonstrates the image style and content can be decomposed and extracted from the features of a pretrained CNN. The extracted content and style are formulated as loss functions to train a feed-forward style transfer network.[34] However, these works are limited to a single style or 32 styles[35] per network. In order to render a style transfer network generalizable to arbitrary style images, Huang & Belongie[36] proposes adaptive instance normalization (AdaIN) given a content input $\boldsymbol{x}$ and a style input $\boldsymbol{y}$, defined as

$$\text{AdaIN}(\boldsymbol{x},\boldsymbol{y}) = \sigma(\boldsymbol{y})\frac{\boldsymbol{x} - \mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})} + \mu(\boldsymbol{y}). \tag{2}$$

Here, μ and σ refer to the mean and standard deviation of a given style input, which adaptively re-scale the content input by the statistics of the style. Similarly, Ghiasi et al.[37] uses InceptionV3 network[38] to extract the style embedding $\boldsymbol{y}$ from a style image.

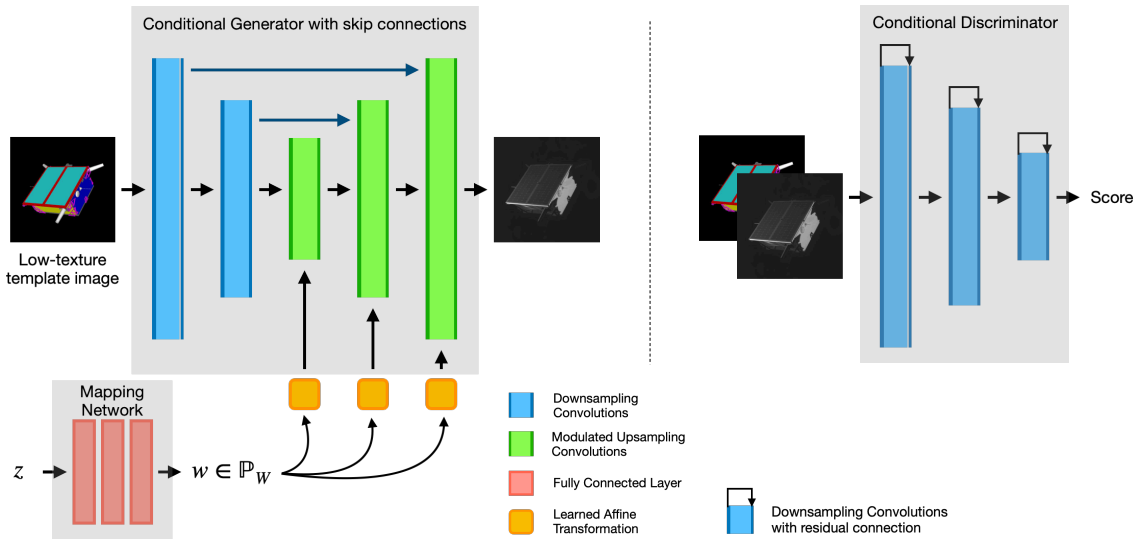

Figure 4. The SPEEDGAN generator (*left*) and discriminator (*right*) architectures. The generator receives low-texture template image and a random noise z drawn from the standard normal distribution.

SPEEDGAN

In this section, the SPEEDGAN architecture illustrated in Figure 4 and its training objectives are explained. The readers interested in further details are encouraged to refer to seminal works by Karras et al.,[23,32,39] from which much of the specific implementation details are borrowed.

Architecture

The SPEEDGAN generator architecture is based on that of StyleGAN,[23] which deviates from the traditional GAN architecture which simply maps a latent vector to an image. Instead, the StyleGAN generator maps a learned constant to an image via a feed-forward convolutional network, while the latent vectors are injected similar to style transfer networks.[37] Specifically, the StyleGAN generator first maps the latent vector $\boldsymbol{z}$ via a mapping network to an intermediate latent vector $\boldsymbol{w}$. Then, the learned affine transformation of $\boldsymbol{w}$, denoted $\boldsymbol{y} = [\boldsymbol{y}_\mu, \boldsymbol{y}_\sigma]$, is injected to each convolutional layer via AdaIN operation (see Eq. 2), scaling and biasing the individual normalized feature maps $\boldsymbol{x}_i$ via $\boldsymbol{y}_\mu$ and $\boldsymbol{y}_\sigma$ respectively. Figure 4 shows that SPEEDGAN generator adopts the same strategy of injecting the latent vector. The original StyleGAN also injects stochastic noise scaled by learned factors after each layer to promote stochacisity in the generated samples (e.g., human hair, freckles, etc.); however, SPEEDGAN forgoes such detail as the dataset lacks such stochastic variation to begin with.

A StyleGAN-based generator alone would have to learn to create high-fidelity images of a spacecraft with arbitrary pose and illumination by itself. This creates two problems. One is that outputing correctly structured spacecraft is an unnecessarily difficult job for a deep neural network to learn. Two is that the user has no control over the pose of the generated images, making it impossible to assign labels. Therefore, similar to Sharma et al.,[40] the generator is conditioned with low-texture templates generated using the same OpenGL-based renderer of SPEED by assigning different colors to each parts of the spacecraft 3D model (see Figure 1). With a priori knowledge of the spacecraft geometry and pose, the generator can instead focus on learning the distribution of the external

effects, or *styles* of the spacecraft, such as illumination, texture, etc. Moreover, the pose of the template images can be automatically assigned to the generated samples.

In order to condition the generator with an image, SPEEDGAN adopts the U-Net architecture[41] similar to pix2pix model[42] for image style translation. While using an encoder/decoder-like architecture is common in many studies of style transfer, SPEEDGAN uses adversarial learning with the discriminator to learn the underlying distribution of such styles present in the dataset. The skip connections are used to concatenate the activations of the early layers to those of the later layers, transferring the knowledge about the template input. Similarly, the discriminator is conditioned on the templates by simply concatenaing them to the input. The discriminator employs residual connections after each downsampling operations, and it outputs a score or a probability that the input image is drawn from the training or generated distribution.

Training Objectives

Similar to StyleGAN, SPEEDGAN use the original loss in Eq. 1 with R1 regularization[43] for the discriminator. The R1 regularization imposes a penalty on the discriminator gradients on real data to promote local convergence of the generative models. Then, from the discriminator loss of Eq. 1, complete discriminator loss to be minimized is given as

$$\mathcal{L}(D) = -\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] + R_1(\theta), \tag{3}$$

where

$$R_1(\theta) = \frac{\gamma}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\|\nabla D_\theta(\boldsymbol{x})\|^2]. \tag{4}$$

Here, θ includes all learnable parameters of the discriminator D, and γ controls the scale of the regularization term.

For generator, the objective is to minimize $\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$ as shown in according to Eq. 1. Instead, as adopted in StyleGAN, the generator of SPEED maximizes a non-saturating loss defined as $\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log D(G(\boldsymbol{z}))]$. Additionally, in order to increase the structural fidelity of the generated spacecraft images, the generator loss is augmented with the content loss used in training a style transfer network.[34,37] Specifically, content loss between the generated sample ($G(\boldsymbol{z})$) and the template ($\boldsymbol{c}$) is defined as the squared Euclidean distance of the respective activations of a classification CNN. The complete generator loss to be minimized is then given as

$$\mathcal{L}(G) = -\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log D(G(\boldsymbol{z})) + \lambda\|f(G(\boldsymbol{z})) - f(\boldsymbol{c})\|^2] \tag{5}$$

where λ is the penalty factor, and $f(\cdot)$ is the activation at the `conv_4` layer of the VGG-19 model[44] pre-trained on ImageNet.[45]

EXPERIMENT

This section explains the dataset used to train SPEEDGAN and some implementation details regarding the GAN architecture, training, and data augmentation. It also introduces Fréchet Inception Distance (FID)[25] which is used to quantify the visual quality of the generated samples.

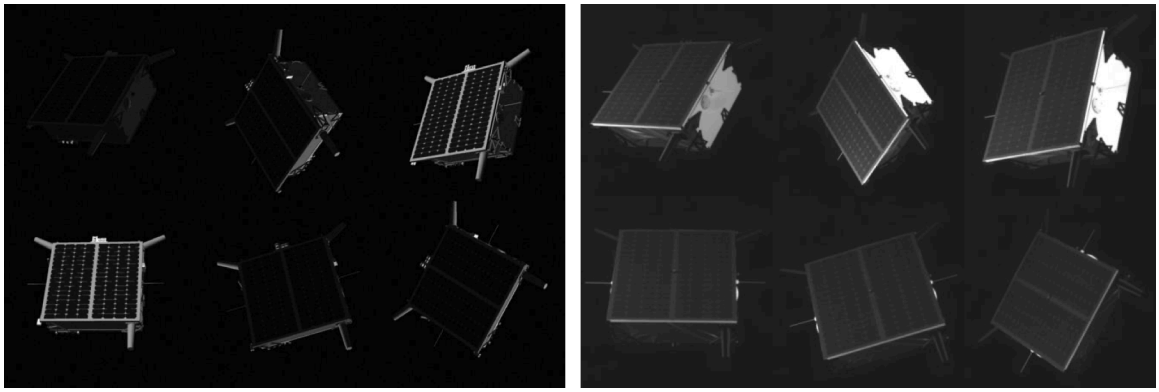

Figure 5. 6 synthetic (*left*) and 6 real (*right*) images representative of 6 different poses present in the training dataset.

Dataset

The dataset for training SPEEDGAN includes 1,200 images composed of 600 synthetic and 600 real images. The real images consist of 300 images that have been published under SPEED and 300 unpublished images. These images were captured using the early TRON testbed; however, due to its limitation on the robot maneuverability and the size of the target mockup spacecraft, the 600 real images have very restricted pose and lighting variability. Specifically, they consist of 6 sets of 100 images sharing similar pose and virtually identical illumination conditions. The 600 synthetic images are generated using the OS camera-emulating software. The synthetic images share the pose labels identical to the real images with random sunlight directions, having more diverse illumination condition compared to the real imagery (see Figure 5). Finally, a set of 600 low-texture templates are created with the same pose set by removing any illumination effect and instead assigning colors to different parts of the spacecraft.

In order to prevent the discriminator overfitting, the dataset is randomly augmented during training. First, the dataset is pre-processed by cropping the original 1920×1200 pixels images into 1200×1200 pixels square images around the spacecraft. Then, during training, a square image is first randomly rotated about the center and then cropped into a random patch of 1024×1024 pixels. These two geometric augmentations respectively simulate in-plane rotation and translation of the target. The same augmentation is performed for the template images to maintain consistent pose. The final image is resized into 256×256 pixels.

Since the 1,200 training images consists of very limited pose distribution, the generator may overfit to the training templates with 600 unique relative orientations. Therefore, in order to gauge the generator's ability to create high-fidelity samples on *unseen* templates, a separate 1,200 test *synthetic* images are created by randomly perturbing the ground-truth orientation of the training images by an angle drawn from the uniform distribution of $[-10, 10]$ degrees. Since these perturbations are not restricted to in-plane rotation, the SPEEDGAN never sees most of these templates during training.

Implementation Details

SPEEDGAN retains much of the implementation details of StyleGAN in our experiment. Specifically, all trainable parameters employ equalized learning rate[32] which dynamically scales the weights

Generator

Block name	Operations	Activation	Output Shape
Input	-	-	3 x 256 x 256
fromRGB	Conv2d (1 x 1)	LReLU	16 x 256 x 256
ConvDown256	Conv2d (3 x 3)	LReLU	16 x 256 x 256
	Conv2d (3 x 3)	LReLU	16 x 256 x 256
ConvDown128	Conv2d (3 x 3 / 2)	LReLU	32 x 128 x 128
	Conv2d (3 x 3)	LReLU	32 x 128 x 128
ConvDown64	Conv2d (3 x 3 / 2)	LReLU	64 x 64 x 64
	Conv2d (3 x 3)	LReLU	64 x 64 x 64
ConvDown32	Conv2d (3 x 3 / 2)	LReLU	128 x 32 x 32
Conv32	ModConv2d (3 x 3)	LReLU	128 x 32 x 32
	ModConv2d (3 x 3)	LReLU	128 x 32 x 32
ConvUp64	ModConv2d (3 x 3)	LReLU	128 x 32 x 32
	ModConv2d (3 x 3 * 2)	LReLU	64 x 64 x 64
ConvUp128	ModConv2d (3 x 3)	LReLU	128 x 64 x 64
	ModConv2d (3 x 3 * 2)	LReLU	32 x 128 x 128
ConvUp256	ModConv2d (3 x 3)	LReLU	64 x 128 x 128
	ModConv2d (3 x 3 * 2)	LReLU	16 x 256 x 256
Conv256	ModConv2d (3 x 3)	LReLU	32 x 256 x 256
	ModConv2d (3 x 3)	LReLU	16 x 256 x 256
toRGB	Conv2d (1 x 1)	linear	1 x 256 x 256

Discriminator

Block name	Operations	Activation	Output Shape
Input	-	-	4 x 256 x 256
fromRGB	Conv2d (1 x 1)	LReLU	8 x 256 x 256
Conv256	Conv2d (3 x 3)	LReLU	8 x 256 x 256
	Conv2d (3 x 3 / 2)	LReLU	16 x 128 x 128
Conv128	Conv2d (3 x 3)	LReLU	16 x 128 x 128
	Conv2d (3 x 3 / 2)	LReLU	32 x 64 x 64
Conv64	Conv2d (3 x 3)	LReLU	32 x 64 x 64
	Conv2d (3 x 3 / 2)	LReLU	64 x 32 x 32
Conv32	Conv2d (3 x 3)	LReLU	64 x 32 x 32
	Conv2d (3 x 3 / 2)	LReLU	128 x 16 x 16
Conv16	Conv2d (3 x 3)	LReLU	128 x 16 x 16
	Conv2d (3 x 3 / 2)	LReLU	128 x 8 x 8
Conv8	Conv2d (3 x 3)	LReLU	128 x 8 x 8
	Conv2d (3 x 3 / 2)	LReLU	128 x 4 x 4
Conv4	Minibatch Stdev	-	129 x 4 x 4
	Conv2d (3 x 3)	LReLU	128 x 4 x 4
	Conv2d (4 x 4)	LReLU	128 x 1 x 1
	Linear	Linear	1 x 1 x 1

Figure 6. SPEEDGAN generator and discriminator architectures. Arrows in generator indicate skip connections via concatenation. Arrows in discriminator indicate residual connections via element-wise addition.

by the per-layer normalization constant of He's initializer.[46] Except the last layer of both networks, leaky ReLU activation is used with $\alpha = 0.2$. All up/downsampling layers also include bilinear filtering to prevent the output signal aliasing.[47] The layers prior to style vector injection in the generator normalize the feature vectors in each pixel to unit length after each convolution operation.[32] The discriminator does not employ any normalization, however the minibatch standard deviation layer is added towards the end of the discriminator.[32] The overall architecture is laid out in Figure 6. Given the size of the dataset, both networks are kept very light in order to minimize overfitting, as the generator only has about 635,000 learnable parameters.

In the subsequent StyleGAN2,[39] the authors note that AdaIN operations often fail to suppress the amplification of some feature maps, creating blob-like artifacts in the generated samples. In order to prevent this issue, SPEEDGAN generator replaces the AdaIN operation with equivalent modulation/demodulation of convolution weights as suggested in StyleGAN2.[39]

In all experiments, SPEEDGAN is trained on 256×256 pixels images with latent vector drawn from 16-dimensional standard normal distribution. The mapping network consists of 8 layers of size 16. The training is done with the AdamW[48] optimizer with learning rate 0.001, $\beta_1 = 0$, $\beta_2 = 0.99$ on minibatches of size 20. The loss regularization penalty factors are set as $\gamma = 0.5$ and $\lambda = 0.1$. Moreover, given the additional computation required to evaluate the discriminator's gradients in R1 regularization in Eq. 4 and the content loss regularization in Eq. 5, these terms are only added to the respective loss once every k minibatches in order to reduce training time[39] ($k = 8$ for G, $k = 16$ for D). Moreover, unless specified otherwise, we employ style mixing regularization as done in StyleGAN,[23] where for 90% of the training images, two random latent vectors are used in the generator instead of one to generate samples. Specifically, they are applied before and after the randomly selected cut-off layer to localize the effect of the style vector at each layer on the final generated samples.

SPEEDGAN is implemented with PyTorch v1.6.0 and trained on an NVIDIA GeForce RTX 2080 Ti 12GB GPU. The architecture in Figure 6 generally takes about 16 hours in this setup to show SPEEDGAN discriminator 3 million training images of size 256×256 pixels. This amounts

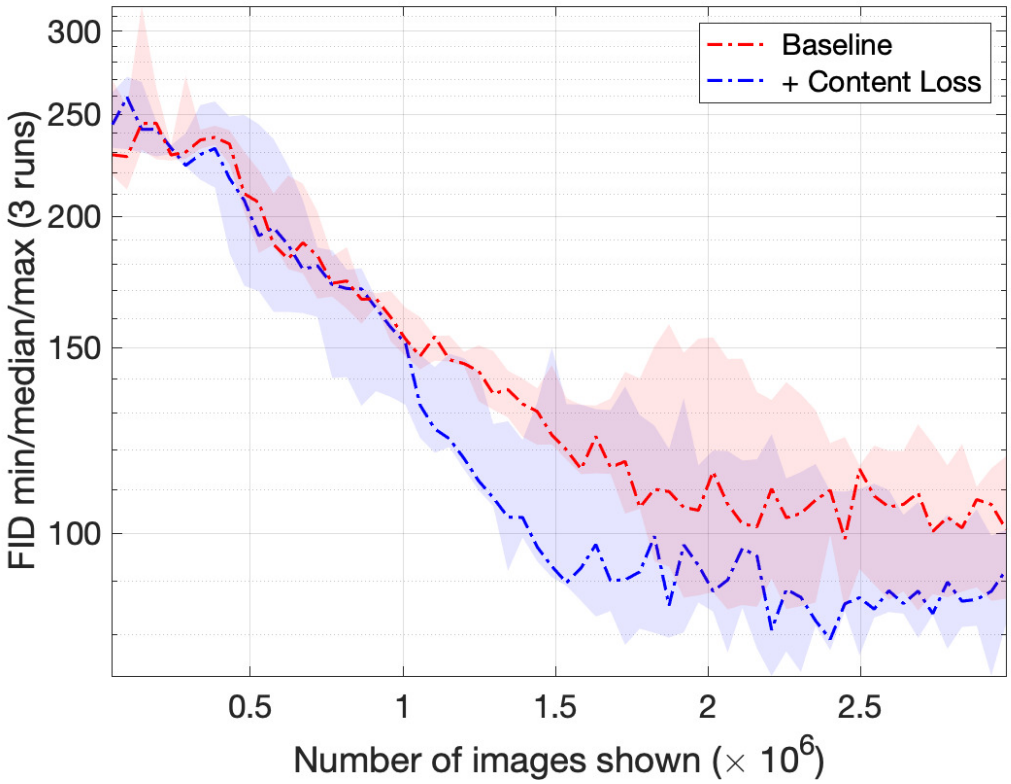

Figure 7. Training curve of FID score versus number of training images shown to the SPEEDGAN discriminator. Each configuration is run 3 times, and minimum, median, and maximum of FID performance at each training step are reported.

to training with the given dataset of 1,200 images for 2,500 epochs.

Evaluation Metric

Apart from the qualitative analysis of the sample quality, it is beneficial to quantify it with respect to the training dataset. Here we use the popular Fréchet Inception Distance (FID)[25] which compares the statistics of the InceptionV3[38] activations of real ($\boldsymbol{x}_r$) and generated ($\boldsymbol{x}_g$) images. Specifically, FID assumes these activations are drawn from multivariate Gaussian distributions with respective mean (m_r, m_g) and covariance (C_r, C_g). After computing these statistics from both sample batches, FID computes the Wasserstein-2 distance between the two Gaussian distributions as

$$d^2(m_r, C_r, m_g, C_g) = \|m_r - m_g\|^2 + \mathrm{Tr}(C_r + C_g - 2\sqrt{C_r C_g}). \tag{6}$$

In all experiments, 12,000 random samples are generated with 10 samples per each training template. These samples are compared to all available 1,200 training images to compute the FID score.

RESULTS

First, Figure 7 shows the training curve of SPEEDGAN trained without style mixing regularization, where the performance is measured in terms of FID score of the samples generated with

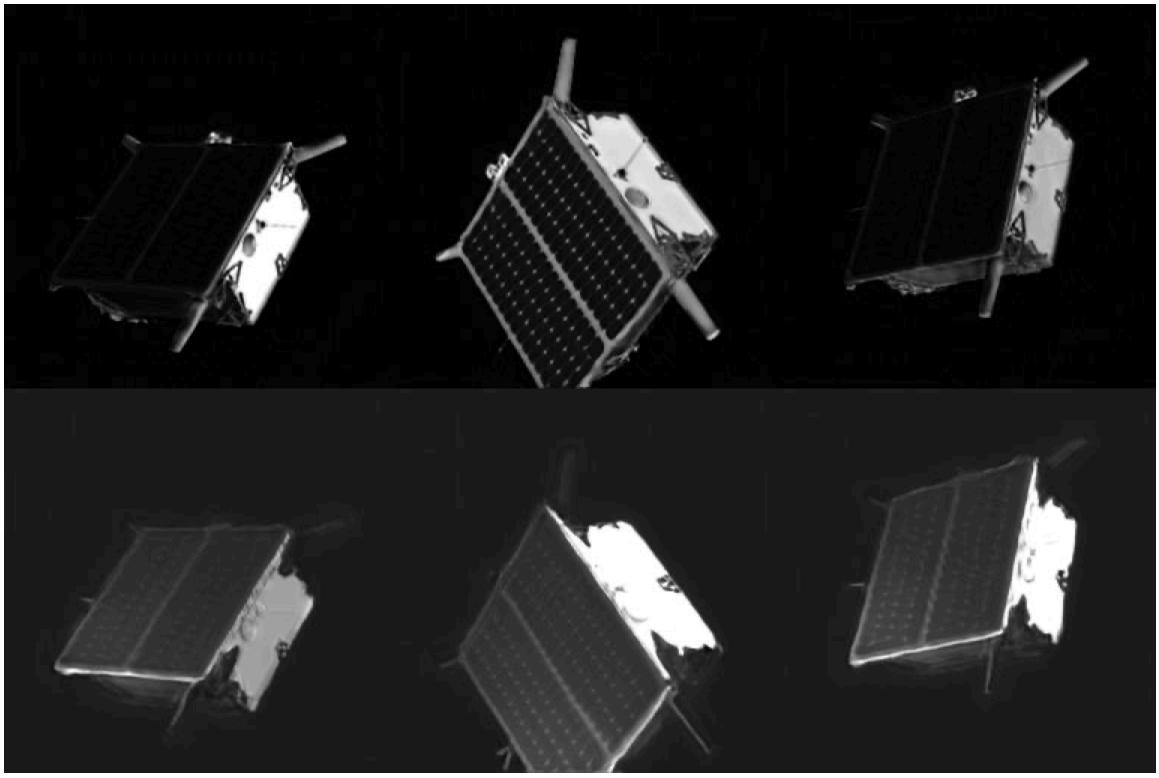

Figure 8. (*Top*) Samples resembling synthetic images. (*Bottom*) Samples resembling TRON real images.

the training templates. The training curve shows some divergence in training pattern due to lack of training images; however, it also exhibits better and faster convergence when the content loss is used to augment the generator loss. Qualitatively, the final samples are visualized in Figure 8 after training SPEEDGAN generator with style mixing regularization for 10 million images. Here, the samples from the generator trained with content loss are shown for three distinct poses. The samples carry detailed patterns of the solar panel, side brackets, and the spacecraft edges.

The advantage of using a StyleGAN-based generator is that the style-mixing regularization during training promotes localization of styles at different layers of the generator, i.e., modifying a subset of style vectors applied to different layers in the generator will only affect certain aspects of the image. For example, for a human face dataset, Karras et al.[23] demonstrates that injecting different style vectors to early layers at low resolution modifies high-level features such as pose, face shape, etc., whereas injecting at later layers at high resolution modifies small details such as hair style, color, etc.

Figure 9 shows that similar trend can be seen on a synthetic-like sample by injecting the style of a real-like sample to different blocks of SPEEDGAN. It is evident that such style replacement in the early layers do not make noticeable change to the sample properties. The effect is the largest when the style replacement is made in the middle layers. Specifically, styles injected to `ConvUp256` (see Figure 6) layers control spacecraft texture, whereas styles injected to `ConvUp128` layers control the effect of illumination, such as overall brightness and the visibility of the solar panel. Such disentanglement, or decoupling, of different styles enables a guided generation of the spacecraft

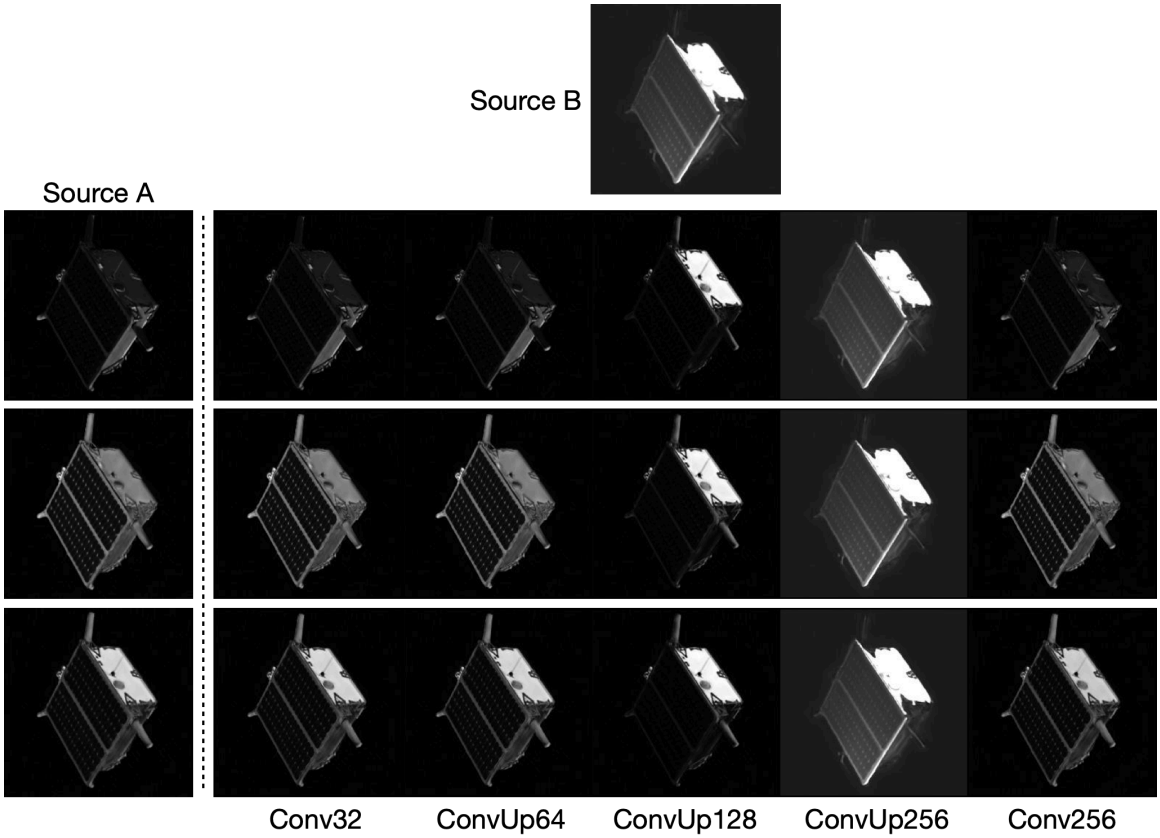

Figure 9. Three samples from Source A (*leftmost column*) and one from Source B (*top*) are generated from respective latent style vectors. The samples in each row are generated from Source A by copying the style vectors of Source B to the specified block of the generator (see Figure 6)

Table 1. Performance of a pose estimation network (KRN)[11] on training vs. generated samples.

	$E_{\mathrm{R}}[^\circ]$	E_{T} [m]	SLAB/ESA Score
Training	23.35	[0.085, 0.067, 1.014]	0.699
Generated	25.15	[0.083, 0.082, 0.921]	0.704

samples. Figure 9 also suggests that given the limited variability of styles (e.g., two spacecraft textures, constant illumination in real images) in the current dataset, the style injection in the early layers is unnecessary. Indeed, while styles corresponding to coarse spatial resolution have shown to have more control over the global features (e.g., pose shape) for the human face dataset,[32] for SPEEDGAN such information is already provided by the template image.

Finally, in order to diagnose the quality of generated samples from the perspective of a pose estimation CNN, the Keypoint Regression Network (KRN) by Park et al.[11] is used to regress 11 keypoint locations from the generated samples. The original KRN is used alongside the Object Detection Network (ODN) which first detects and isolates the region-of-interest (RoI) around the spacecraft, such that KRN performs keypoint regression on images cropped around the detected RoI. In this work, KRN pre-trained on SPEED synthetic training images is applied directly on the

generated 256 $\times$ 256 pixels samples. Then, the predicted keypoint locations are translated to the *true* locations in the original 1920 $\times$ 1200 pixels image, which are used to compute the quaternion ($\tilde{\boldsymbol{q}}$) and the translation vector ($\tilde{\boldsymbol{t}}$) using EPnP algorithm.[15] These are compared against the true pose ($\boldsymbol{q}, \boldsymbol{t}$). Specifically, the same metrics as in Park et al.[11] are used, where

$$E_{\mathrm{R}} = 2 \arccos |\tilde{\boldsymbol{q}} \cdot \boldsymbol{q}| \tag{7}$$

$$E_{\mathrm{T}} = |\tilde{\boldsymbol{t}} - \boldsymbol{t}| \tag{8}$$

$$\text{SLAB/ESA Score} = \frac{1}{N} \sum_{i=1}^{N} \frac{\|\tilde{\boldsymbol{t}}^{(i)} - \boldsymbol{t}^{(i)}\|_2}{\|\boldsymbol{t}^{(i)}\|_2} + E_{\mathrm{R}}^{(i)}. \tag{9}$$

The combined ODN/KRN architecture has scored 0.0626 on SPEED synthetic test set and 0.3951 on SPEED real test set.[11]

Table 1 shows the performance of the KRN-only architecture. The KRN's performance on the training dataset, which consists of 600 synthetic and 600 real images, is worse than reported by Park et al.[11] The discrepancy is due to significantly different image pre-processing procedures employed in this work, especially in terms of image cropping. However, most importantly, Table 1 demonstrates the KRN's performance on the generated samples is similar to its performance on the training images in terms of both rotation and translation errors, indicating that SPEEDGAN is able to produce samples with features that can be recognized by another CNN trained on the true synthetic images.

CONCLUSION

This works presents SPEEDGAN, the very first generative model to synthesize the spacecraft images. Specifically, by combining the state-of-the-art GAN generator architecture and the style transfer network with its training objective, SPEEDGAN is able to generate from the low-texture templates the high-quality images of the Tango spacecraft with characteristics of both SPEED synthetic and real imageries. Specifically, it was shown that adding content loss to the generator's training objective significantly accelerates the GAN training as represented by the FID score. Moreover, the SPEEDGAN's generator can localize the effects of style vectors in its architecture, such that modifying a style vector in one block affects the sample spacecraft's texture and illumination when modified in another block. The visual quality of the generated samples are also assessed in terms of their viability as a resource to train a pose estimation CNN. Specifically, it is shown that SPEEDGAN can generate samples with features that can be recognized by the pose estimation CNN trained on SPEED synthetic images.

Immediate future works include further improving the visual quality of the generated samples and devising a quantitative metric similar to FID which captures both sample quality and variety. Moreover, SPEEDGAN will be trained on SPEED+, an upgraded dataset which will contain a large quantity of real images with much more diverse pose distribution and illumination conditions. Specifically, the updated TRON facility will be capable of (1) generating a large-scale dataset of calibrated spacecraft images with uniformly distributed poses and (2) capturing a spacecraft image corresponding to a commanded pose. These new capabilities will help answer a number of long-term goals, one of which is to further diversify the training dataset for pose estimation with new realistic samples based on synthetic and real images, but having visual properties that are unlike those of sythetic or real images.

ACKNOWLEDGEMENT

The authors would like to thank the King Abdulaziz City for Science and Technology (KACST) Center of Excellence for research in Aeronautics & Astronautics (CEAA) at Stanford University for sponsoring this work. The authors would like to thank OHB Sweden, the German Aerospace Center (DLR), and the Technical University of Denmark (DTU) for the 3D model of the Tango spacecraft used to create the images used in this work. The authors would also like to thank Dr. Sumant Sharma for generating the low-texture template images of the spacecraft.

REFERENCES

[1] J. L. Forshaw, G. S. Aglietti, N. Navarathinam, H. Kadhem, T. Salmon, A. Pisseloup, E. Joffre, T. Chabot, I. Retat, R. Axthelm, S. Barraclough, A. Ratcliffe, C. Bernal, F. Chaumette, A. Pollini, and W. H. Steyn, "RemoveDEBRIS: An in-orbit active debris removal demonstration mission," *Acta Astronautica*, Vol. 127, 2016, p. 448–463, 10.1016/j.actaastro.2016.06.018.

[2] B. Sullivan, D. Barnhart, L. Hill, P. Oppenheimer, B. L. Benedict, G. V. Ommering, L. Chappell, J. Ratti, and P. Will, "DARPA Phoenix Payload Orbital Delivery system (PODs): "FedEx to GEO"," *AIAA SPACE 2013 Conference and Exposition*, 2013, 10.2514/6.2013-5484.

[3] B. B. Reed, R. C. Smith, B. J. Naasz, J. F. Pellegrino, and C. E. Bacon, "The Restore-L Servicing Mission," *AIAA Space 2016*, 2016, 10.2514/6.2016-5478.

[4] A. Cropp and P. Palmer, "Pose Estimation and Relative Orbit Determination of a Nearby Target Microsatellite using Passive Imager," *5th Cranfield Conference on Dynamics and Control of Systems and Structures in Space 2002*, 2002, pp. 389 – 395.

[5] A. Petit, E. Marchand, and K. Kanani, "Vision-based space autonomous rendezvous: A case study," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 619–624, 10.1109/IROS.2011.6094568.

[6] M. R. Leinz, C.-T. Chen, M. W. Beaven, T. P. Weismuller, D. L. Caballero, W. B. Gaumer, P. W. Sabasteanski, P. A. Scott, and M. A. Lundgren, "Orbital Express Autonomous Rendezvous and Capture Sensor System (ARCSS) flight test results," *Sensors and Systems for Space Applications II* (R. T. Howard and P. Motaghedi, eds.), Vol. 6958, International Society for Optics and Photonics, SPIE, 2008, pp. 62 – 74, 10.1117/12.779595.

[7] S. D'Amico, M. Benn, and J. L. Jørgensen, "Pose estimation of an uncooperative spacecraft from actual space imagery," *International Journal of Space Science and Engineering*, Vol. 2, No. 2, 2014, p. 171, 10.1504/ijspacese.2014.060600.

[8] S. Sharma, J. Ventura, and S. D'Amico, "Robust Model-Based Monocular Pose Initialization for Noncooperative Spacecraft Rendezvous," *Journal of Spacecraft and Rockets*, 2018, p. 1–16, 10.2514/1.a34124.

[9] S. Sharma and S. D'Amico, "Pose Estimation for Non-Cooperative Spacecraft Rendezvous Using Neural Networks," *2019 AAS/AIAA Space Flight Mechanics Meeting, Ka'anapali, Maui, HI*, January 13-17 2019.

[10] S. Sharma, *Pose Estimation of Uncooperative Spacecraft using Monocular Vision and Deep Learning*. PhD thesis, Stanford University, Department of Aeronautics & Astronautics, Aug 2019.

[11] T. H. Park, S. Sharma, and S. D'Amico, "Towards Robust Learning-Based Pose Estimation of Noncooperative Spacecraft," *2019 AAS/AIAA Astrodynamics Specialist Conference, Portland, Maine*, August 11-15 2019.

[12] M. Kisantal, S. Sharma, T. H. Park, D. Izzo, M. Märtens, and S. D'Amico, "Satellite Pose Estimation Challenge: Dataset, Competition Design and Results," *IEEE Transactions on Aerospace and Electronic Systems*, 2020, pp. 1–1.

[13] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 8, June 1986, pp. 679–698, 10.1109/TPAMI.1986.4767851.

[14] C. Harris and M. Stephens, "A combined corner and edge detector," *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[15] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An Accurate O(n) Solution to the PnP Problem," *International Journal of Computer Vision*, Vol. 81, No. 2, 2008, p. 155–166, 10.1007/s11263-008-0152-6.

[16] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Readings in Computer Vision*, 1987, p. 726–740, 10.1016/b978-0-08-051581-6.50070-2.

[17] S. Sharma and S. D'Amico, "Comparative Assessment of Techniques for Initial Pose Estimation using Monocular Vision," Vol. 123, 2016, pp. 435–445.

[18] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," *Robotics: Science and Systems XIV*, 2018, 10.15607/rss.2018.xiv.019.

[19] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Oral*, 2019.

[20] S. Sharma, C. Beierle, and S. D'Amico, "Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks," *2018 IEEE Aerospace Conference*, March 2018, pp. 1–12.

[21] S. Sharma, T. H. Park, and S. D'Amico, "Spacecraft Pose Estimation Dataset (SPEED)," Stanford Digital Repository. Available at: `https://doi.org/10.25740/dz692fn7184`, 2019.

[22] S. D'Amico, P. Bodin, M. Delpech, and R. Noteborn, "PRISMA," *Distributed Space Missions for Earth System Monitoring Space Technology Library* (M. D'Errico, ed.), Vol. 31, ch. 21, pp. 599–637, 2013, 10.1007/978-1-4614-4541-8_21.

[23] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4396–4405.

[24] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2414–2423.

[25] 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," *Advances in Neural Information Processing Systems 30*, 2017.

[26] C. Beierle and S. D'Amico, "Variable-Magnification Optical Stimulator for Training and Validation of Spaceborne Vision-Based Navigation," *Journal of Spacecraft and Rockets*, Vol. 56, Feb 2019, pp. 1–13, 10.2514/1.A34337.

[27] J. Sullivan, T. A. Lovell, and S. D'Amico, "Angles-Only Navigation for Autonomous On-Orbit Space Situational Awareness Applications," *2018 AAS/AIAA Astrodynamics Specialist Conference, Snotbird, Utah*, August 19-23 2018.

[28] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, Cambridge, MA, USA, MIT Press, 2014, p. 2672–2680.

[29] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *International Conference on Learning Representations*, 2016.

[30] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks," *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[31] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved Training of Wasserstein GANs," *Advances in Neural Information Processing Systems 30*, 2017.

[32] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," *International Conference on Learning Representations*, 2018.

[33] L. A. Gatys, A. S. Ecker, and M. Bethge, "A Neural Algorithm of Artistic Style," *arXiv*, Aug 2015.

[34] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," *European Conference on Computer Vision*, 2016.

[35] V. Dumoulin, J. Shlens, and M. Kudlur, "A Learned Representation For Artistic Style," *International Conference on Learning Representation*, 2017.

[36] X. Huang and S. Belongie, "Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1510–1519.

[37] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, "Exploring the structure of a real-time, arbitrary neural artistic stylization network," *British Machine Vision Conference*, 2017.

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[39] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[40] S. Sharma, C. Beierle, and S. D'Amico, "Generative Adversarial Networks for High-Fidelity Simulation of Spacecraft Proximity Operations," tech. rep., Stanford Space Rendezvous Laboratory (SLAB), April 23 2018.

[41] O. Ronneberger, P.Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Vol. 9351 of *LNCS*, Springer, 2015, pp. 234–241. (available on arXiv:1505.04597 [cs.CV]).

[42] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," *CVPR*, 2017.

[43] L. Mescheder, A. Geiger, and S. Nowozin, "Which Training Methods for GANs do actually Converge?," *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 3481–3490.

[44] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, 2014.

[45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, Vol. 115, No. 3, 2015, pp. 211–252, 10.1007/s11263-015-0816-y.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[47] R. Zhang, "Making Convolutional Networks Shift-Invariant Again," *ICML*, 2019.

[48] I. Loshchilov and F. Hutter, "Fixing Weight Decay Regularization in Adam," *CoRR*, Vol. abs/1711.05101, 2017.