Improving Sim-to-Real Transfer in Vision-Based Robot Navigation via Instance-Level GAN-Based Data Augmentation [*]

Ignacio G. López-Francos, Rory Lipkis, Pavlo Vlastos, Adrian Agogino
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA, USA
ignacio.lopez-francos@nasa.gov

Achieving robust vision-based robotic tasks requires large amounts of data, which are often difficult to obtain in real-world scenarios. Simulators and synthetic data offer a cost-effective alternative, but the visual gap between simulation and reality hinders the performance of models when deployed in real-world environments. In this paper, we present a data augmentation pipeline that integrates a foundation model (Segment Anything Model) with an unsupervised image-to-image translation model (CycleGAN) for instance-level domain transfer from simulation to reality. This pipeline enables the generation of realistic labeled data from synthetic images for training supervised machine learning models for machine perception tasks. We evaluate our approach on real-world data for ego-vehicle pose estimation, a critical autonomous navigation task involving the prediction of cross-track position and heading angle relative to road center line markings. The results of our tests show that our GAN-based data augmentation pipeline significantly outperforms models trained solely on simulation data or on data processed with standard image augmentation methods for sim-to-real transfer, enhancing model robustness and generalizability in real-world scenarios. Our method provides a scalable and flexible data augmentation tool for leveraging large synthetic datasets to enhance vision-based robotic navigation tasks.

I. INTRODUCTION

Learning-based visual robotic navigation requires large amounts of labeled camera data to train a model to *perceive* the environment, estimate the robot's state within it, plan a trajectory, and physically execute it. Collecting enough labeled data for such task is expensive, time consuming, and hard to obtain in the real world, as it requires large and diverse episodes [1] (e.g., different environments, different times of the day, different weather conditions, etc.) to train a model for the robot to perform robustly in a variety of scenarios [2]. In aviation [3] and space [4] applications, this issue is exacerbated by rigorous regulatory constraints, the high costs of data collection, the rarity of opportunities for empirical flight testing, and the scarcity of labeled datasets. Designing vision-based space robotic systems for deployment in extraterrestrial environments such as Mars [5] or Enceladus [6] faces the additional challenge of operating in environments for which we have limited (if any) data, hindering our ability to adequately train models on scenarios that the systems will encounter off-world.

The use of simulators can help mitigate this data issue by cost-effectively generating a well-distributed synthetic dataset with ground-truth annotations across diverse scenarios representative of the variability in the real world. However, and despite recent advancements in the quality of the graphics in simulators, there always exists some visual and physical differences between data collected in simulation and the real word. Models trained in simulation may not fully capture the nuances of real-world physics, sensor noise, and environmental conditions, leading to a discrepancy known as the *reality gap*, or more specifically in the context of synthetic data, the *Sim-to-Real gap*. When training a model with simulated data and deploying in the real world, this discrepancy can hinder the model's ability to generalize, resulting in unsatisfactory performance.

To address the visual Sim-to-Real gap, a common strategy is the use of Domain Adaptation techniques to make synthetic images more realistic by transferring the knowledge learned from real-world images. These methods often rely

[*]GOVERNMENT RIGHTS NOTICE. This work was authored by employees of KBR Wyle Services, LLC under Contract No. 80ARC020D0010 with the National Aeronautics and Space Administration. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, or allow others to do so, for United States Government purposes. All other rights are reserved by the copyright owner.

on Generative Adversarial Networks (GANs) [7–9], a type of deep learning architecture that consist of two competing neural networks: a generator and a discriminator. The generator is trained to produce new “realistic-looking” synthetic data, while the discriminator attempts to detect whether or not the generator’s images are real. Through the use of an *adversarial loss*, this competition forces the generator to produce data that becomes, in principle, increasingly indistinguishable from the real data.

An extension of the original GAN is CycleGAN [10] which introduces a *cycle consistency loss* to enable domain translation without requiring paired examples. Specifically, CycleGAN ensures that when translating an image from the source domain to the target domain and then back to the source domain, you end up with an image that closely resembles to the original. In other words, it "cycles" the image back to its original domain without losing too much information or introducing too much distortion. As a result, CycleGAN learns the underlying relationships between domains without explicit one-to-one correspondences, facilitating effective domain adaptation in scenarios where obtaining paired training data is impractical or infeasible.

The idea of using CycleGAN for synthesizing large-scale, realistic images has been explored extensively in robotics and other fields. For example, it has been used to improve crop object detection [11], detect defects in petrochemical pipelines [12], segment organs or pathologies in CT scans [13], improve robotic visual grasping via reinforcement learning [14–16], perform audio super-resolution without the need of paired low- and high-resolution audio files [17], and enhance underwater scenes for vision-driven navigation of autonomous underwater vehicles (AUVs) [18]. In our work, we aim to improve vision-based robot navigation by using CycleGAN as the key component of a data augmentation pipeline. Our pipeline generates *realistically-looking* road surface images by transferring the visual style of real-world roads onto synthetic images collected from a front-facing camera mounted on a rover operating in a custom graphics-based simulator.

One caveat of CycleGAN is its tendency to adapt a global style to the entire input image, which can inadvertently alter or remove visual cues that are critical for specific tasks. For example, in our case, it may incorrectly transfer the learned texture of the asphalt from real images to the sidewalk regions in the synthetic images, potentially confusing a downstream algorithm designed to estimate the ego vehicle’s position on the road for safe navigation.

To address this issue, we propose an instance-level style transfer approach in our data augmentation pipeline, where the style transfer is restricted to specific pixels in the target image corresponding to road asphalt. Traditionally, achieving this level of precision requires training a segmentation model on manually labeled data—a laborious and time-consuming process. However, advancements in foundation models now offer scalable and generalizable alternatives. In this work, we leverage the Segment Anything Model (SAM) [19] to automatically identify road asphalt pixels in synthetic images. The combination of SAM with CycleGAN, ensure that our data augmentation pipeline applies style transfer exclusively to road regions.

Our main contributions in this work are as follows:

1) A data augmentation pipeline that enhances the quality of labeled synthetic images for zero-short Sim-to-Real transfer.
2) A procedure for instance-level real road style transfer that selectively transforms only the pixels corresponding to road asphalt, ensuring realism while preserving non-road elements.
3) An application of the proposed method to real-world vision-based robotic navigation tasks.

We evaluate our approach on a vision-based robot navigation task—specifically, ego-vehicle pose estimation for line following. Experimental results show that training a machine learning model for this task with the images generated by our GAN-based data augmentation pipeline leads to significant improvements in robustness and generalizability. Compared to models trained solely on simulation data or using standard data augmentation techniques, our method achieves superior performance. These results underscore the effectiveness of our data augmentation strategy in bridging the Sim-to-Real gap and enhancing the reliability of robotic perception systems in real-world scenarios.

II. RELATED WORK

The adoption of deep learning-based methods has created a paradigm shift in robotics, which traditionally relied on well-defined rule-based methods, and has significantly accelerated progress in the field. We now see robots exhibiting human-like behaviors capable of tackling complex, unpredictable, real-world tasks, such as grasping, manipulation, and navigation. Deep learning-based methods excel because they can learn from vast amounts of data, automatically extracting and generalizing complex patterns and features that are not easily programmable using traditional approaches. This allows robots to adapt to new and dynamic environments, significantly improving their decision-making and problem-solving capabilities. However, this reliance on large, annotated datasets is also a key disadvantage in robotics, as generating annotated data for training from real-world systems is both expensive and time-consuming. In this section, we review the use of data augmentation techniques to address this limitation, the role of simulations to generate cost-effective data, and the Sim-to-Real gap problem along with techniques to address it.

A. Data Augmentation Techniques

Data augmentation is a critical component in machine learning, particularly for deep learning models, where the availability and diversity of training data significantly influence their performance and generalizability. The seminal work by LeCun et al.[20] introduced one of the earliest forms of data augmentation, employing simple techniques such as image warping on handwritten digits to enhance neural network training for classification tasks. This foundational work highlighted the dual benefits of data augmentation: increasing dataset size and ensuring that the dataset accurately reflects the variability of real-world conditions.

To handle class data imbalances—common in datasets used for model training—techniques such as oversampling and undersampling are used. These methods adjust the class distribution to mitigate model bias, either by increasing the representation of underrepresented classes or reducing overrepresented ones. For computer vision applications, traditional data augmentation strategies include geometric transformations (e.g., scaling, rotation, translation, shearing, and flipping), color space modifications, kernel-based filtering for image sharpening or blurring, image mixing (e.g., CutMix), and random erasing [21]. These approaches enhance a model's generalization by exposing it to diverse transformations of the same input data.

The emergence of GANs introduced a transformative approach to data augmentation. GANs are capable of generating synthetic instances of data that preserve the essential statistical and visual characteristics of the original dataset. As noted by Bowles et al.[22], GANs can "unlock" additional information embedded within the dataset. Various GAN architectures have since been developed to address specific augmentation needs, including DCGANs[8], Progressively Growing GANs [23], CycleGANs [10], and Conditional GANs [24].

In this work, we build upon these advancements by designing a data augmentation pipeline that leverages CycleGAN to perform style transfer. Unlike traditional global style adaptation, our approach incorporates a foundation model to identify and focus exclusively on pixels corresponding to road asphalt, enabling instance-level style transfer. This method enhances the realism of synthetic road images, resulting in a dataset better suited for training vision-based models for robotic navigation.

B. Digital Twins for Synthetic Image Generation

Obtaining annotated data for robotic applications is both challenging and expensive. Simulators offer a cost-effective solution by enabling the generation of precisely labeled training data at low cost and scale. This capability is particularly advantageous for training machine learning models that require large, diverse datasets to perform reliably in the real world. The use of simulators for synthetic image generation is well-established in robotics. Digital twin technologies have become a cornerstone in this field, providing detailed and interactive virtual replicas of physical environments. Common simulation engines used to build digital twins include Gazebo [25], Unreal Engine [26], Unity [27], and, most recently, NVIDIA Omniverse Isaac Sim [28]. These platforms are not only used for generating synthetic data at scale but also for testing machine learning models across numerous scenarios, including leveraging techniques such as reinforcement learning.

To support our group's research, we developed a custom Digital Twin (DT), referred to as NASA Ames DT [29], which accurately models the NASA Ames Research Center campus and its installations. We built NASA Ames DT on Unreal Engine 4 and used modeling tools such as Blender [30] and FreeCAD [31] to generate high-fidelity virtual replicas of assets, including our physical rover, buildings, roads, and other environmental elements. This simulation environment serves as a comprehensive platform for generating synthetic data and testing machine learning models for autonomous navigation. Figure 1 shows a comparison between our NASA Ames DT and the real-world NASA Ames Research Center campus.

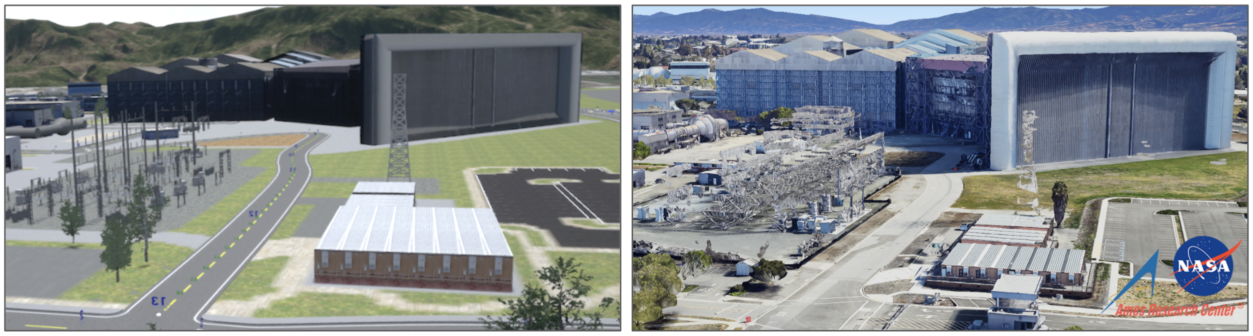


Fig. 1 An aerial perspective comparison of (left) the NASA Ames DT and (right) the real-world render from Google Earth. Both images show Parsons Ave, a road used for testing, and the wind tunnel within the NASA Ames Research Center campus, in Mountain View, CA.

To interface with NASA Ames DT, we integrated Microsoft AirSim [32], a versatile simulation framework that allows full control of the vehicle via its Python API. Using AirSim, we can collect annotated data at scale, including RGB images, depth maps, and segmentation masks from the simulated cameras onboard the vehicle, as well as telemetry data. For each data point, we can also collect the rover's position (i.e., lateral displacement) and orientation (i.e., heading angle) relative to a predefined center line, which can be used as labels for training machine learning models for vehicle pose estimation.

Through NASA Ames DT and the AirSim API, we are able to generate thousands of diverse labeled images and associated telemetry, significantly reducing reliance on expensive large real-world data collection.

C. Simulation-to-Reality

Training machine learning models purely in simulation or using only synthetic data often leads to sub-optimal performance when these models are deployed in the real world. This discrepancy, known as the *Sim-to-Real gap*, arises from the inability of simulators to fully replicate the complex visual and physical properties of real-world environments [9, 33, 34]. While modern simulators have become increasingly realistic, the gap remains a persistent challenge. In this work, we focus on image processing techniques to address the *visual* component of the Sim-to-Real gap, which stems from discrepancies in appearance between simulated and real-world imagery.

Two primary approaches for bridging the visual Sim-to-Real gap are **Domain Randomization** and **Domain Adaptation**, illustrated in Figure 2. These methods offer complementary strategies to enable models trained on synthetic data to perform effectively in real-world environments.

Domain Randomization creates diverse and randomized variations of the source domain ($\mathcal{S}$) by deliberately altering visual properties such as texture, color, and lighting. The goal is to expose the model to such a wide range of variations during training that, when deployed in the target domain ($\mathcal{T}$), the real-world environment appears as another variation. This forces the model to focus on invariant features that remain consistent across all scenarios, improving its generalization capability.

Domain Adaptation, on the other hand, aligns the source domain ($\mathcal{S}$) with the target domain ($\mathcal{T}$) by leveraging techniques that transfer knowledge learned in $\mathcal{S}$ to $\mathcal{T}$, often without requiring labeled data from the target domain. This approach seeks to reduce the domain discrepancy, enabling models trained in the source domain to generalize effectively in the target domain.

In our work, we integrate both approaches. First, we apply Domain Randomization by generating a diverse synthetic dataset through randomizing weather conditions and times of day in the NASA Ames DT simulator. Second, we leverage CycleGAN to develop a data augmentation pipeline as a Domain Adaptation method, allowing the model to learn the visual style of real-world roads and transfer it to synthetic images. The details of this Domain Adaptation pipeline are described in Section III.C.

III. METHODS

Given a labeled dataset in a source domain ($\mathcal{S}$)—simulation—and an unlabeled dataset in a target domain ($\mathcal{T}$)—real-world—we aim to train a model on $\mathcal{S}$ that generalizes effectively to $\mathcal{T}$. In this section, we detail our methodology, which

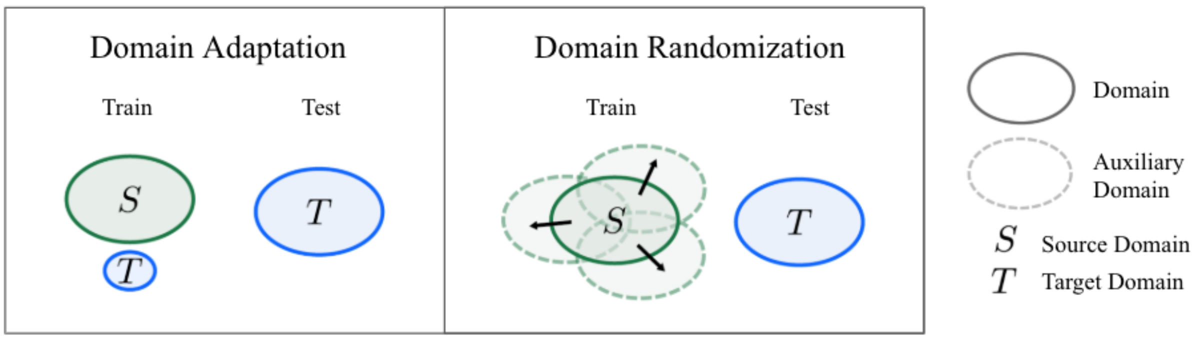


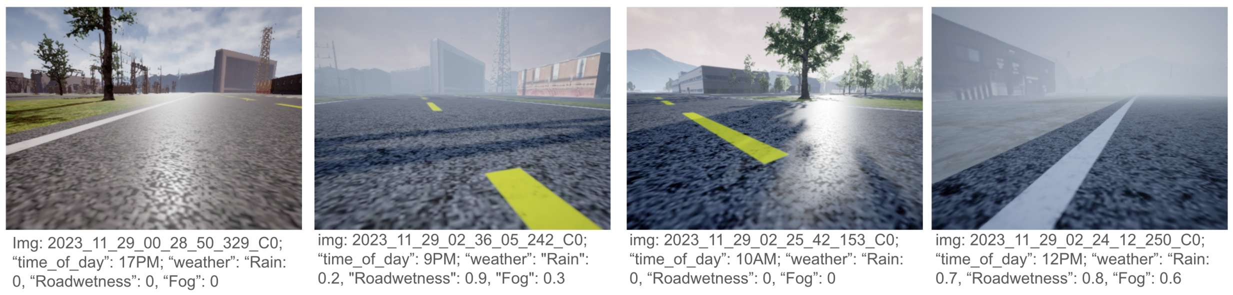
Fig. 2 Comparison of Domain Adaptation and Domain Randomization, the two primary approaches for Sim-to-Real transfer. In Domain Adaptation, the source domain ($\mathcal{S}$) is aligned with the target domain ($\mathcal{T}$) using auxiliary data, while in Domain Randomization, diverse variations of the source domain are created to enable generalization to the target domain. Figure adapted from [35].

involves collecting annotated data from NASA Ames DT ($\mathcal{S}$), collecting data from our physical rover in the real world ($\mathcal{T}$), training CycleGAN on these datasets, and developing a custom data augmentation pipeline to generate visually realistic and labeled data for training machine learning models for vision-based autonomous navigation.

A. Synthetic Data Collection

We first collected a diverse synthetic dataset of images and annotations using the NASA Ames DT simulator and AirSim API (see Section II.B). The rover in the virtual environment is equipped with an adjustable camera system that can be fixed to various positions and orientations on its chassis. For our experiments, we set a single forward-facing camera mounted at the center of the rover's front section. The simulation environment was carefully designed to mimic the target domain, as well as other physical test locations within the NASA Ames Research Center campus [36].

Using AirSim API, we built custom Python scripts to fully automate the data collection process. These scripts allowed us to control the rover's movement and systematically vary parameters to generate datasets that reflect real-world variability. The parameters included the vehicle's pose (position and orientation) within the simulation, different times of day, and various weather conditions. For each parameter, we specified randomized ranges to replicate environmental conditions likely to be encountered during real-world tests at NASA Ames. For instance, weather conditions were set to sunny, cloudy, and rainy-we could include snow, but it is excluded since it does not occur in our local testing environment in Mountain View. Figure 3 showcases the diversity of data generated in the simulator, with examples under varying weather conditions and times of day.


Img: 2023_11_29_00_28_50_329_C0; "time_of_day": 17PM; "weather": "Rain: 0, "Roadwetness": 0, "Fog": 0

img: 2023_11_29_02_36_05_242_C0; "time_of_day": 9PM; "weather": "Rain": 0.2, "Roadwetness": 0.9, "Fog": 0.3

img: 2023_11_29_02_25_42_153_C0; "time_of_day": 10AM; "weather": "Rain: 0, "Roadwetness": 0, "Fog": 0

img: 2023_11_29_02_24_12_250_C0; "time_of_day": 12PM; "weather": "Rain: 0.7, "Roadwetness": 0.8, "Fog": 0.6

Fig. 3 Examples of images generated in the NASA Ames DT simulator under various simulated weather conditions and times of day. The annotations display parameters such as time of day, weather, road wetness, and fog level, demonstrating the simulator's capability to generate diverse datasets.

The scripts also allowed us to automate the control of the virtual rover. This process began by spawning the vehicle at a random pose within a predefined bounded area in the simulator (e.g., within specific test roads). Next, the vehicle was programmed to follow predefined trajectories, collecting images and telemetry data—including lateral displacement and heading angle relative to the road center line—at a rate of 1 Hz.

By following this systematic approach, we generated thousands of synthetic images from the virtual rover onboard camera, paired with corresponding telemetry and labels, providing a diverse synthetic dataset for this study.

B. Real Data Collection

We collected real-world image data using the NASA R-RAV rover [36], which serves as the physical counterpart to our virtual rover modeled in simulation. The NASA R-RAV rover, shown in Figure 4, is a custom-built six-wheeled platform equipped with a front-mounted camera, an NVIDIA Jetson Xavier AGX as its primary computing platform—which enables deployment of larger models by leveraging its onboard GPU for parallel computations—a Pixhawk PX4 flight controller board, a telemetry radio, a motor controller board, and additional electronics for power distribution and signal control. The chassis incorporates a differential drive system, allowing for sharp turns and in-place rotations. The rover was designed with three different control modes: remote control, wireless control via a laptop ground control station (GCS), and autonomous control. Sets of captured images could be saved to onboard memory at user-defined intervals and uploaded to backup servers.

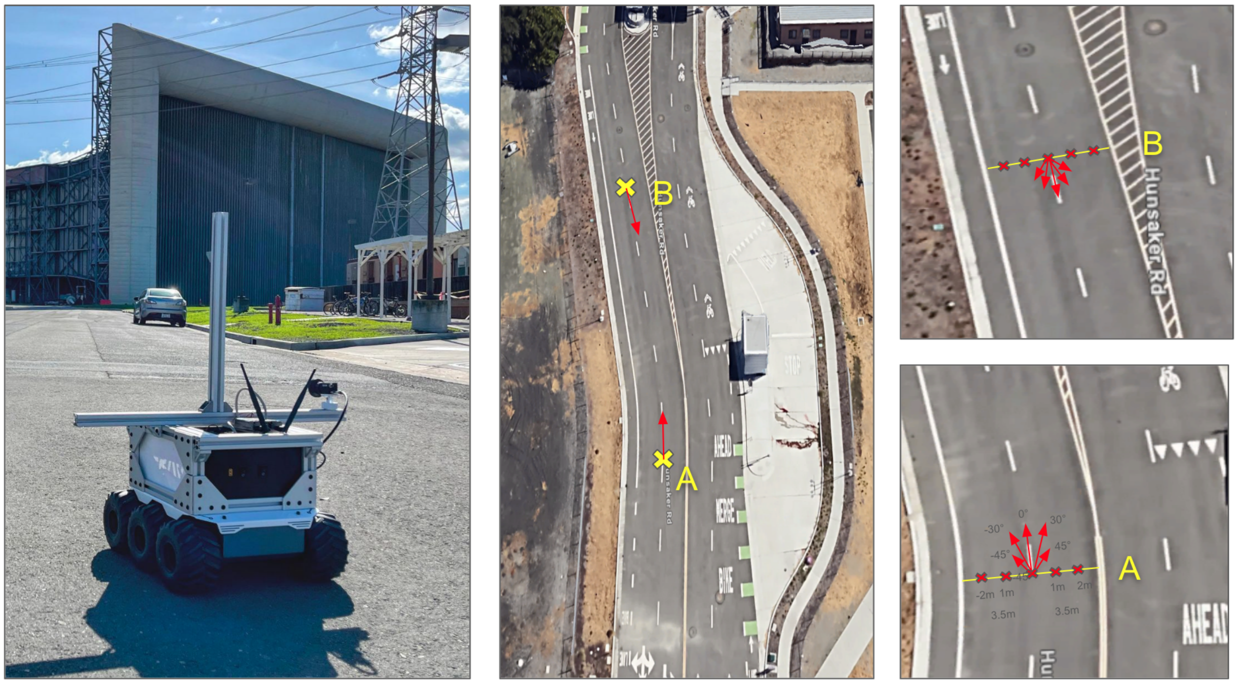


Fig. 4 Left: The NASA R-RAV rover in Parsons Ave., in front of NASA Ames Research Center wind tunnel. Right: Aerial images of Hunsaker Road, showing the specific area for the data collection test. Starting points A and B are marked, along with the predefined positions and orientations for the test runs.

Figure 5 shows the high-level NASA R-RAV rover hardware system block diagram. The system allows for both manual and autonomous control. Using manual control, the user can move and position the rover through a remote controller. Since some operations, such as image data collection and executing specific programs, is controlled from the GCS, the system uses radios for a wireless interface to the rover. This also allows for features such as video telemetry to the GCS.

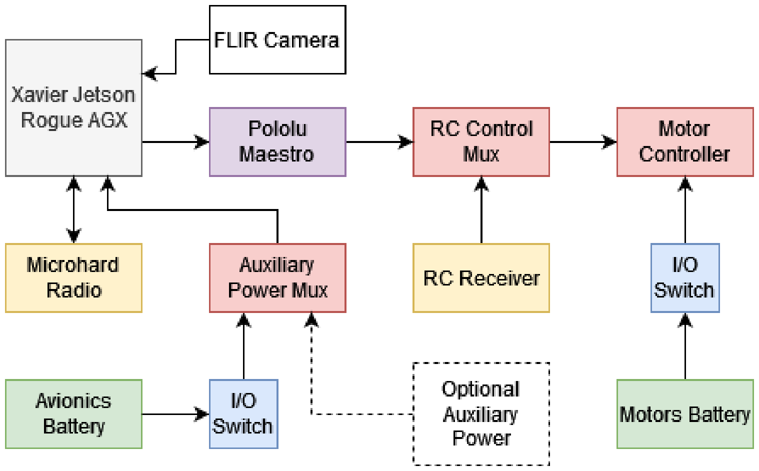


Fig. 5 Block diagram of NASA R-RAV hardware system.

For the data collection test, we selected a gated section of Hunsaker Road within NASA Ames Research Center. This location, closed to regular traffic, provided a controlled environment for data collection. The test section of the road was bounded by two predefined starting points, labeled as A and B, as shown in the aerial imagery in Figure 4. These points served as the initial positions for the rover during data collection.

To ensure diversity in the dataset collected, we systematically modified the rover's initial position and orientation for each test run. The following parameters were defined:

- **Starting points:** A and B, along the road.
- **Position offsets** (x, y in meters): $-2, -1, 0, 1, 2$.
- **Orientation angles** (θ in degrees): $-45, -30, 0, 30, 45$.

Prior to data collection, the rover's camera was calibrated using a black-and-white checkerboard with known dimensions placed at a fixed distance. A total of 50 test runs were conducted, one for every combination of the three parameters. During each data collection run, the rover was teleoperated via remote control to navigate along the dashed centerline, moving from A to B while following oscillating trajectories from one side of the road to the other. This oscillation ensured that the centerline remained visible within the camera's field of view, which is necessary for predicting the rover's pose relative to it. Using the GCS, the rover was signaled over radio to initiate image capture at a predefined rate. Additional asynchronous image captures could also be manually triggered from the GCS, if necessary.

Image labeling was performed using two configurations. In the first configuration, images were manually labeled in real-time from the GCS as either "left" or "right" relative to the centerline. In the second configuration, we estimated the vehicle's position and orientation. This process involved manually marking the centerline in each image, followed by running a custom script that used the calibrated camera parameters to compute the cross-track error (lateral displacement from the centerline) and heading angle (angular deviation from the centerline).

A total of 1,892 images were collected during the data collection process. Figure 6 shows a selection of these images. After filtering out images with excessive distortion or an obscured centerline and preprocessing the remainder (cropping the road area and resizing), we finalized a dataset of 1,109 images. This curated dataset (`real_data`) was used as the real-world test set for this study.

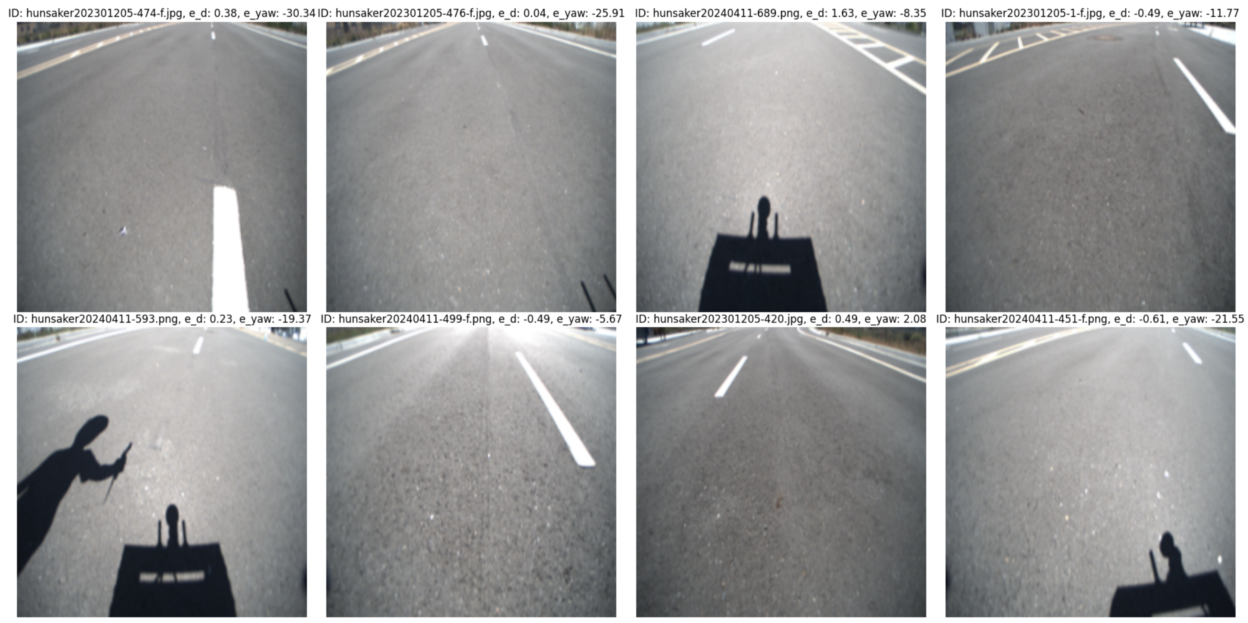

ID: hunsaker202301205-474-f.jpg, e_d: 0.38, e_yaw: -30.34
ID: hunsaker202301205-476-f.jpg, e_d: 0.04, e_yaw: -25.91
ID: hunsaker20240411-689.png, e_d: 1.63, e_yaw: -8.35
ID: hunsaker202301205-1-f.jpg, e_d: -0.49, e_yaw: -11.77
ID: hunsaker20240411-593.png, e_d: 0.23, e_yaw: -19.37
ID: hunsaker20240411-499-f.png, e_d: -0.49, e_yaw: -5.67
ID: hunsaker202301205-420.jpg, e_d: 0.49, e_yaw: 2.08
ID: hunsaker20240411-451-f.png, e_d: -0.61, e_yaw: -21.55

Fig. 6 Examples of images collected during the data collection test on Hunsaker Road with the physical R-RAV rover. Each image is annotated with its corresponding file ID, lateral displacement (e_d), and heading angle ($e_{\mathbf{yaw}}$) relative to the centerline. These annotations were computed through a semi-automated process involving manual marking of the centerline and camera-based pose estimation.

C. Data Augmentation Pipeline

The primary goal of our data augmentation pipeline is to adapt images from the simulator to make them visually closer in appearance to real-world data.

At a high level, our pipeline comprises two main components: (1) learning the real-world road asphalt visual style using CycleGAN and (2) transferring this style to specific road regions in synthetic images using a foundation model.

1. Learning Real Road Asphalt Style

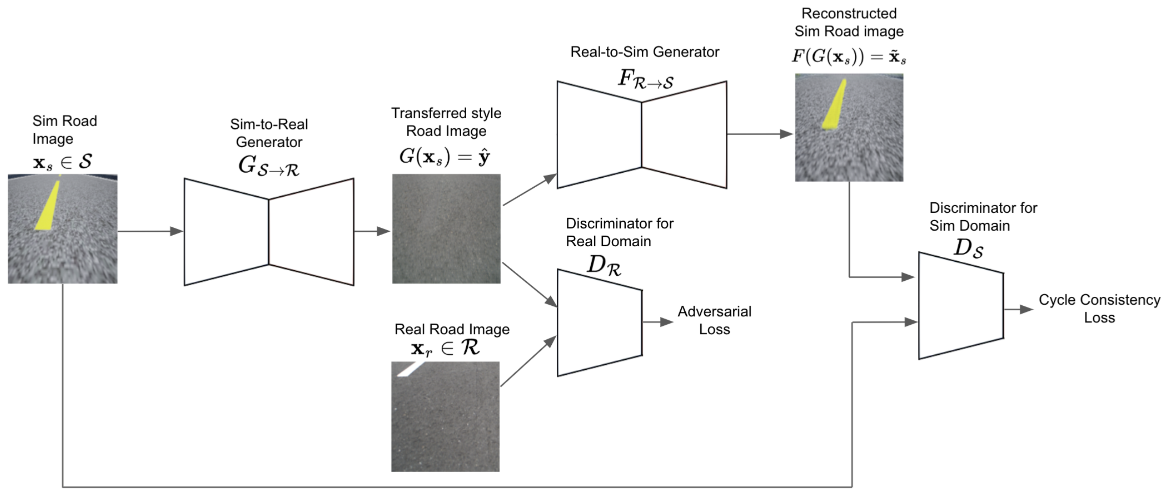
We leverage CycleGAN [10], an unpaired image-to-image translation model, to learn the visual style of real-world road asphalt and address the sim-to-real gap. For training CycleGAN, we used two datasets: $\mathcal{R}$, consisting of 120 unlabeled real-world road images collected by our physical rover, and $\mathcal{S}$, comprising 120 labeled synthetic road images generated through the NASA Ames DT. These datasets represent the real-world and synthetic domains, respectively.

CycleGAN enables a mapping $G : \mathcal{S} \longrightarrow \mathcal{R}$, where a synthetic road image $\mathbf{x}_s \in \mathcal{S}$ is transformed into a realistic image $\hat{\mathbf{y}} = G(\mathbf{x}_s)$, closely resembling a real-world road image $\mathbf{x}_r \in \mathcal{R}$. Similarly, CycleGAN also learns the inverse mapping $F : \mathcal{R} \longrightarrow \mathcal{S}$, which maps real-world images back to the synthetic domain. These mappings are trained jointly using Adversarial and Cycle Consistency Losses, which are described below. Figure 7 illustrates this training process.

1) **Adversarial Loss:** This ensures that the transformed image $\hat{\mathbf{y}} = G(\mathbf{x}_s)$ visually resembles real-world road images $\mathbf{x}_r$ as closely as possible. A discriminator $D_{\mathcal{R}}$ is trained to distinguish between real and generated images, while the generator G is optimized to produce images that "fool" the discriminator. A similar adversarial loss is applied to the inverse mapping F, with discriminator $D_{\mathcal{S}}$ ensuring that real-world images transformed back to the synthetic domain $\hat{\mathbf{z}} = F(\mathbf{x}_r)$ resemble the synthetic road images $\mathbf{x}_s$.
2) **Cycle Consistency Loss:** This loss ensures that translating an image from one domain to another and back again reconstructs the original image, enforcing consistency in the learned mappings. Specifically, for a synthetic road image $\mathbf{x}_s \in \mathcal{S}$, the reconstructed image $\tilde{\mathbf{x}}_s = F(G(\mathbf{x}_s))$ should be close to the original $\mathbf{x}_s$. Similarly, for a real-world road image $\mathbf{x}_r \in \mathcal{R}$, the reconstruction $\tilde{\mathbf{x}}_r = G(F(\mathbf{x}_r))$ should closely resemble the original real-world image $\mathbf{x}_r$. This loss ensures that the transformations preserve the structural integrity of the input images.

After training CycleGAN on our datasets, we obtained two generators, G and F, as well as their corresponding discriminators $D_{\mathcal{R}}$ and $D_{\mathcal{S}}$. For our specific use case, we focus on G, which translates synthetic road images into realistic-looking images. Once trained, we save the generator G to integrate it into our data augmentation pipeline, for transforming synthetic images into visually realistic data that better resemble the real-world domain.

Details on how G is integrated into our data augmentation pipeline are explained in the next section.



Sim Road Image
$\mathbf{x}_s \in \mathcal{S}$
Sim-to-Real Generator
$G_{\mathcal{S}\to\mathcal{R}}$
Transferred style Road Image
$G(\mathbf{x}_s) = \hat{\mathbf{y}}$
Real-to-Sim Generator
$F_{\mathcal{R}\to\mathcal{S}}$
Reconstructed Sim Road image
$F(G(\mathbf{x}_s)) = \tilde{\mathbf{x}}_s$
Discriminator for Real Domain
$D_{\mathcal{R}}$
Adversarial Loss
Real Road Image
$\mathbf{x}_r \in \mathcal{R}$
Discriminator for Sim Domain
$D_{\mathcal{S}}$
Cycle Consistency Loss

Fig. 7 Overview of CycleGAN training for transferring the style of a real-world road to a synthetic one. The image shows the mappings $G : \mathcal{S} \rightarrow \mathcal{R}$ and $F : \mathcal{R} \rightarrow \mathcal{S}$, along with their corresponding discriminators $D_{\mathcal{R}}$ and $D_{\mathcal{S}}$. Adversarial loss ensures that transformed images $\hat{\mathbf{y}} = G(\mathbf{x}_s)$ and $\hat{\mathbf{z}} = F(\mathbf{x}_r)$ are indistinguishable from their target domains, while cycle consistency loss enforces that reconstructed images $\tilde{\mathbf{x}}_s = F(G(\mathbf{x}_s))$ and $\tilde{\mathbf{x}}_r = G(F(\mathbf{x}_r))$ closely match the originals.

2. Instance-Level Real Road Style Transfer

One limitation of CycleGAN is its global style adaptation, which can inadvertently alter or remove critical visual elements necessary for downstream navigation tasks. To address this, we rely on a semantic segmentation model to identify and isolate specific regions in the image for targeted style application. In our scenario, we apply the learned real-world style from CycleGAN's generator G exclusively to the road asphalt regions in the synthetic images.

To create a data augmentation pipeline that is both generalizable and scalable, we make use of Segment Anything Model (SAM) [19], a foundation model developed by Meta. SAM is trained on the extensive SA-1B dataset, comprising over 1 billion segmentation masks across 11 million images, and exhibits robust zero-shot performance in various segmentation tasks. Its architecture consists of three key components:

1) **Image Encoder:** A Vision Transformer (ViT) that extracts comprehensive feature representations from input images.
2) **Prompt Encoder:** Processes user-defined inputs—such as points, boxes, or text prompts—to guide the segmentation process.
3) **Mask Decoder:** Combines information from both the image and prompt encoders to generate precise segmentation masks.

This architecture allows SAM to generalize effectively to new image distributions and tasks without additional training, facilitating the zero-shot segmentation capabilities that we leverage in this work.

Our data augmentation pipeline, illustrated in Figure 8 and detailed in Algorithm 1, processes synthetic images S collected from NASA Ames DT using the following steps:

1) **Road Mask Extraction:** Each synthetic image $\mathbf{x}_s \in \mathcal{S}$ is passed through SAM to extract a segmentation mask corresponding to road asphalt regions.
2) **Style Transfer:** CycleGAN's generator G, pre-trained to learn the visual style of real-world asphalt, is applied to the extracted road mask, transferring the real-world road style while leaving road markings (e.g., dashed lines) unaltered.

3) **Overlay:** The transformed road mask is overlaid onto the original synthetic image $\mathbf{x}_s$ to create a realistic-looking road scene.
4) **Line Markings Conversion:** To ensure consistency with the real-world test dataset, a post-processing script converts yellow line markings to white by modifying the RGB pixel values within the corresponding line marking masks.

This instance-level approach ensures that the style transfer is confined to road regions and preserves the visual consistency of non-road elements in the image.

After running the data augmentation pipeline and processing all the synthetic images from NASA Ames DT, we end up with a new dataset of realistic-looking road images paired with precise labels, ready for training machine learning models for downstream vision-based autonomous applications.

Algorithm 1 Instance-Level Real Road Style Transfer Image Generation

```
Require: $\mathcal{S}$                                  ▷ Dataset containing synthetic images from NASA Ames DT
Require: $G$                                            ▷ Pre-trained CycleGAN generator for real road style
1: for $\mathbf{x}_s \in \mathcal{S}$ do
2:     semantic_masks ← SAM($\mathbf{x}_s$)                    ▷ Apply SAM to extract semantic segments
3:     road_mask ← semantic_masks[“road”]                ▷ Select mask corresponding to the road
4:     realistic_road_mask ← $G$(road_mask)               ▷ Transfer real road style to road mask using generator $G$
5:     $\mathbf{x}_s$ ← Overlay($\mathbf{x}_s$, realistic_road_mask)      ▷ Overlay transformed mask onto original image
6:     store $\mathbf{x}_s$                                   ▷ Save processed image
7: end for
```

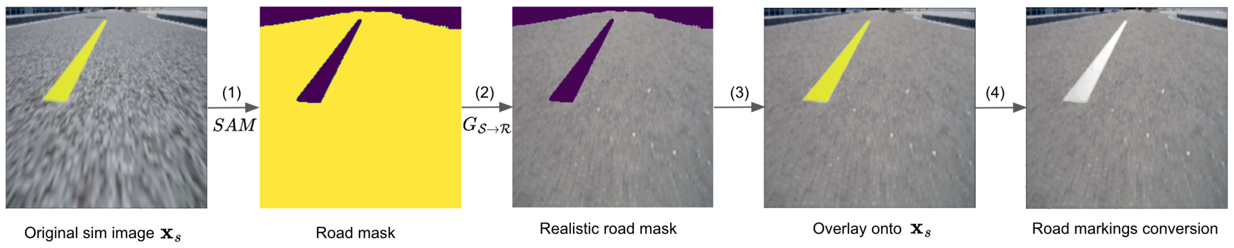

(1)
SAM
(2)
$G_{\mathcal{S}\to\mathcal{R}}$
(3)
(4)
Original sim image $\mathbf{x}_s$
Road mask
Realistic road mask
Overlay onto $\mathbf{x}_s$
Road markings conversion

Fig. 8 Instance-Level Real Road Style Transfer pipeline. Steps include: (1) road mask extraction using SAM, (2) real-world road style transfer using CycleGAN's generator G, (3) overlaying the transformed road mask onto the original image, and (4) converting yellow line markings to white.

3. Vehicle Pose Estimation

Vehicle pose estimation is a crucial task in localization, allowing a vehicle to determine its position and orientation relative to a reference, such as the center line of a road. By using onboard camera images, we can predict two key metrics: the vehicle's cross-track error distance (e_d) and the heading angle error (e_{yaw}), which indicate the lateral displacement and orientation error (angular deviation) relative to the road center line, respectively.

To achieve accurate pose estimation, we train a deep neural network, named **TaxiNet**, which is based on the **MobileNetV2** architecture. TaxiNet takes a single image from the vehicle's front-facing camera as input and outputs predictions for e_d and e_{yaw}. This approach enables real-time inference for autonomous center line following tasks.

We trained TaxiNet using an NVIDIA Adam Quadro RTX 5000 GPU. We used the Adam optimizer with a learning rate of 1e-3 and a learning rate decay every 50 epochs, a custom smooth L1 loss function, and a batch size of 32 over 200 epochs. We include an Early Stopping callback that stops training after 5 consecutive epochs with no improvement over validation loss. Additionally, we ran several training cycles with additional batches of data to further enhance model performance. The same training regime was applied to every dataset and model trained.

In the next section IV.B, we will present the results of our approach, showcasing improvements in pose estimation accuracy compared to baseline methods.

IV. EXPERIMENTS

A. Vehicle Pose Estimation for Line-Following Task

We evaluated our GAN-based data augmentation pipeline on a vision-based robot navigation task: vehicle pose estimation for line following. Accurate ego-vehicle pose estimation is crucial for autonomous vehicle (AV) systems, as it allows the vehicle to determine its position relative to a reference, such as the road centerline. This capability is essential for safe and efficient navigation. Using onboard camera images, we predict the vehicle's cross-track error distance (e_d) and cross-track error heading angle (e_{yaw}), which represent the lateral displacement and orientation error relative to the centerline, respectively.

1. Datasets

For our experimental setup we used four datasets:

- **Simulation Dataset (`sim_data`):** This synthetic dataset, collected from NASA Ames DT (see Section III.A), contains 7,840 front-camera images of the rover. Images were cropped and resized to 224×224 pixels, with corresponding telemetry data and ground truth annotations for e_d and e_{yaw}.
- **Standard Augmented Dataset (`std_aug_data`):** We generated a new data set from the simulation dataset using standard augmentation techniques. This approach focuses on color and visual perturbation, maintaining the original properties and labels of the images. For each image in the simulation dataset we randomly applied one of the following augmentations: posterization (2-bit color reduction), color jittering (brightness, contrast, saturation, and hue adjustments), Gaussian blur (kernel size: 5–9, σ: 0.1–5), AugMix (a combination of multiple augmentations), and automatic contrast optimization. We generated a total of 7,840 transformed images, keeping the original labels from the simulation dataset.
- **GAN-Augmented Dataset (`GAN_aug_data`):** This is the dataset created using the instance-level real road style data augmentation pipeline (see Section III.C.2), and it includes 7,480 images with the same annotations as the simulation dataset.
- **Real Dataset (`real_data`):** Collected using our physical rover during tests at NASA Ames Research Center (see Section III.B). It contains 1,109 front-camera images cropped and resized to 224×224 pixels, with corresponding e_d and e_{yaw} annotations. This dataset was primarily used for testing and benchmarking Sim-To-Real transfer strategies in a zero-shot fashion.

2. Sim-To-Real Transfer Strategies

We evaluated our GAN-based data augmentation pipeline by training a machine learning model on each of the synthetic datasets. In particular, we trained TaxiNet, a lightweight convolutional neural network based on MobileNetV2 architecture [37], optimized for embedded vision applications. TaxiNet predicts e_d and e_{yaw} from a single image captured by the vehicle's front camera.

The same training regime was used for all three datasets. Training was performed on an NVIDIA Quadro RTX 5000 GPU using Adam optimizer with a learning rate of 1e−3 and a learning rate decay every 50 epochs, a custom smooth L1 loss function, and a batch size of 32 over 200 epochs. Early stopping was applied to halt training after five consecutive epochs with no improvement in validation loss.

B. Evaluation

We designed our experiment setup to test the following hypotheses:

- **Hypothesis 1:** The performance of TaxiNet on real-world data improves when trained on our GAN-augmented dataset compared to training on the simulation dataset alone.
- **Hypothesis 2:** Our GAN-based data augmentation pipeline provides better generalization compared to standard data augmentation techniques.

We tested each of our three TaxiNet models on the real dataset (`real_data`). Table 1 summarizes the results. TaxiNet trained on the GAN-augmented dataset outperformed the model trained on the simulation dataset across both mean absolute error (MAE) and mean squared error (MSE) metrics for e_d and e_{yaw}. Specifically, we observed improvements of 26.19% and 23.10% in MAE, and 45.00% and 28.44% in MSE, respectively. These results validate Hypothesis 1, demonstrating that GAN-based data augmentation enhances model robustness and performance in real-world scenarios.

Table 1. Vehicle Pose Estimation Results with Different Sim-To-Real Transfer Methods on Real Data

Method	MAE		MSE	
	e_d (m)	e_{yaw} (deg)	e_d (m)	e_{yaw} (deg)
Sim Only	0.84	8.40	1.20	161.94
Std Aug	0.68	9.40	0.78	163.89
GAN Aug	**0.62**	**6.46**	**0.66**	**115.88**

Additionally, TaxiNet trained with the GAN-augmented dataset outperformed the model trained with the Standard Augmented Dataset (`std_aug_data`), with improvements of 8.82% and 31.28% in MAE for e_d and e_{yaw}, respectively, and 15.38% and 29.29% in MSE for e_d and e_{yaw}. These results validate Hypothesis 2, emphasizing the effectiveness of our GAN-based data augmentation pipeline for Sim-To-Real transfer.

Figure 9 illustrates the benchmark results between models trained on simulation-only data (`sim100`) and GAN-augmented data (`ganaug100`). The scatter plots on the left compare the predicted versus actual values of e_d (top) and e_{yaw} (bottom). The points for `ganaug100` align more closely with the identity line, indicating better prediction accuracy. On the right, the histograms of prediction errors for both e_d and e_{yaw} show narrower distributions for `ganaug100`, further demonstrating its improved performance and generalization.

Figure 10 presents a selection of best and worst predictions for e_d and e_{yaw} from the GAN-augmented model. The top row showcases examples where the model closely matches the ground truth, both for e_d (left) and e_{yaw} (right). In contrast, the bottom row shows some of the worst predictions, where discrepancies between the predicted and actual values are more pronounced, often occurring under challenging scenarios with sharp angles or poorly visible center lines.

V. CONCLUSION

We introduced a novel Sim-to-Real data augmentation pipeline to improve the performance of machine learning models for vision-based navigation tasks. Our pipeline generates realistic, labeled images—which are typically difficult and expensive to obtain in the real world—by transforming synthetic data from the NASA Ames DT simulator using a small set of unlabeled real-world images collected at our testing site at NASA Ames Research Center. We trained CycleGAN to capture the visual style of real-world road surfaces and integrated Segment Anything Model (SAM) to perform instance-level style transfer, precisely targeting road regions in synthetic images. To validate our approach, we trained TaxiNet model, a lightweight convolutional neural network for vehicle pose estimation, and demonstrated that our GAN-based data augmentation pipeline significantly improves model robustness and generalizability in real-world scenarios. Specifically, models trained using our GAN-augmented data outperformed those trained solely on simulation data or with standard augmentation techniques.

Our experiments showed that this pipeline enables effective zero-shot transfer of vision-based navigation models from simulation environments to physical environments, substantially reducing the need for extensive real-world data collection. Nevertheless, a key limitation of our approach is CycleGAN's tendency to learn an average style or texture of the scene, which reduces its effectiveness under diverse environmental conditions or roads with varying appearances.

Future work will focus on improving the adaptability of the style transfer component to accommodate a broader range of environmental conditions. This includes expanding our method to additional semantic categories beyond road asphalt, such as sidewalks, vehicles, trees, and buildings. While CycleGAN could technically be trained to learn the textures of these additional categories individually, we plan to explore more scalable generative approaches, such as diffusion models, to improve flexibility and performance. Additionally, integrating our pipeline with reinforcement learning frameworks could further enhance the training and deployment of autonomous navigation systems in simulated environments. By addressing these limitations and exploring new avenues, we aim to develop a more versatile and robust data augmentation tool for advancing machine learning models for vision-based robotic navigation tasks.

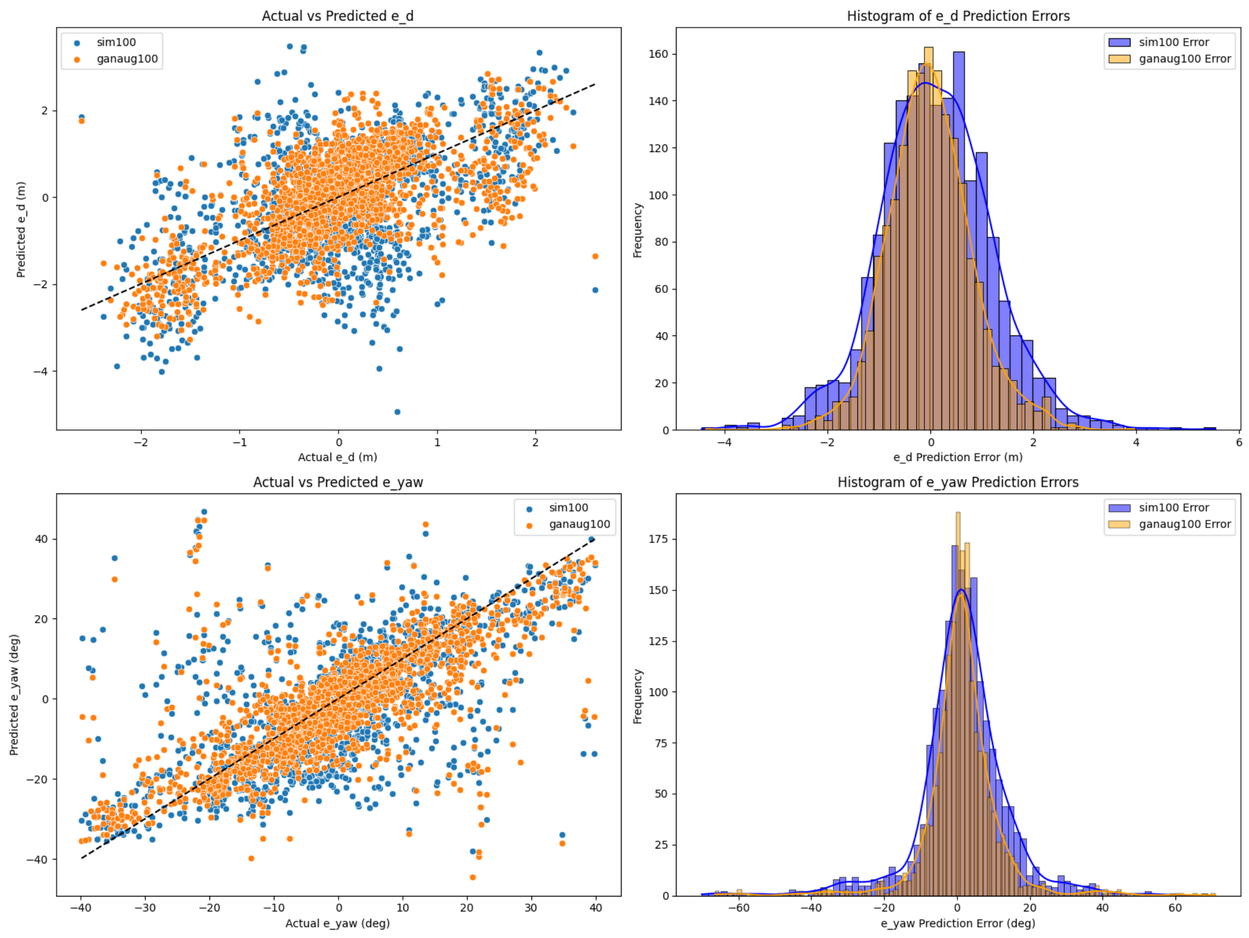


Fig. 9 Comparison of prediction performance between models trained on simulated-only data (`sim100`) and GAN-augmented data (`ganaug100`). Left plots show the relationship between actual and predicted e_d (top) and e_{yaw} (bottom), with models trained using GAN-augmented data showing closer alignment to the identity line. Right plots present histograms of prediction errors, where GAN-augmented models achieve narrower error distributions, demonstrating improved accuracy and generalization.

Acknowledgments

This work was supported by the System-Wide Safety (SWS) Project under the NASA Aeronautics Research Mission Directorate (ARMD) Airspace Operations and Safety Program (AOSP).

References

[1] Padalkar, A., Pooley, A., Jain, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Brohan, A., et al., "Open x-embodiment: Robotic learning datasets and rt-x models," *arXiv preprint arXiv:2310.08864*, 2023.

[2] Coicheci, S., and Filip, I., "Self-driving vehicles: current status of development and technical challenges to overcome," *2020 IEEE 14th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2020, pp. 000255–000260. https://doi.org/10.1109/SACI49304.2020.9118809.

[3] Shin, S., and Hwang, I., "Data-mining-based computer vision analytics for automated helicopter flight state inference," *Journal of Aerospace Information Systems*, Vol. 14, No. 12, 2017, pp. 652–662.

[4] Schmitt, M., Prexl, J., Ebel, P., Liebel, L., and Zhu, X. X., "Weakly supervised semantic segmentation of satellite images for land cover mapping–challenges and opportunities," *arXiv preprint arXiv:2002.08254*, 2020.

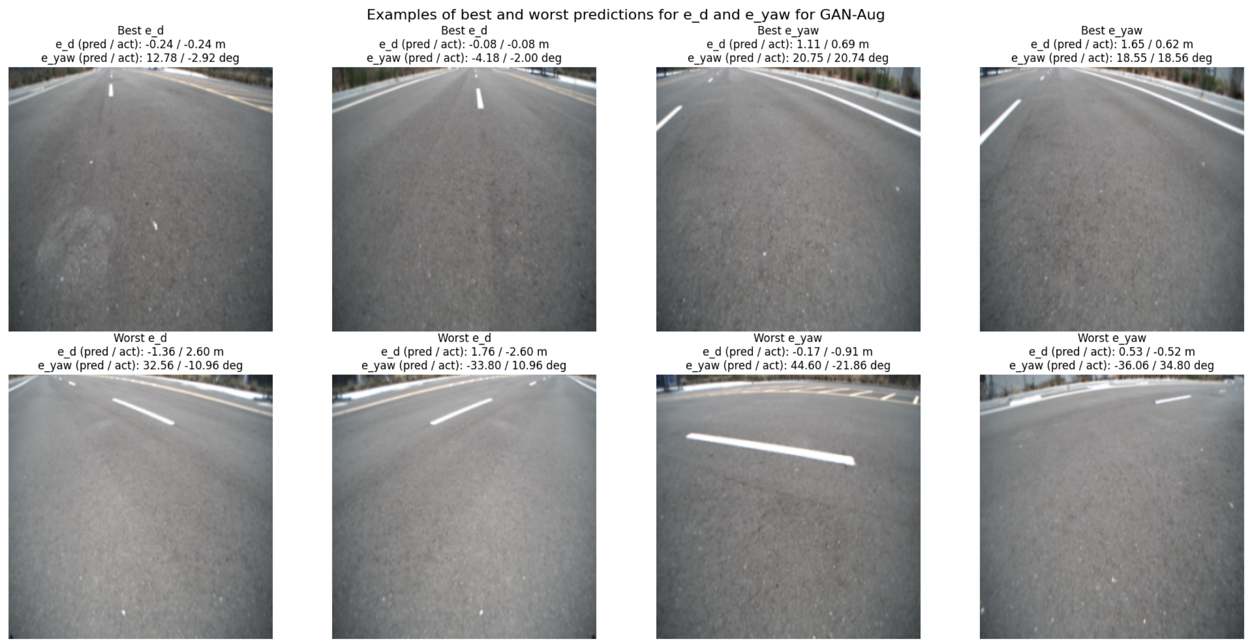

Examples of best and worst predictions for e_d and e_yaw for GAN-Aug
Best e_d
e_d (pred / act): -0.24 / -0.24 m
e_yaw (pred / act): 12.78 / -2.92 deg
Best e_d
e_d (pred / act): -0.08 / -0.08 m
e_yaw (pred / act): -4.18 / -2.00 deg
Best e_yaw
e_d (pred / act): 1.11 / 0.69 m
e_yaw (pred / act): 20.75 / 20.74 deg
Best e_yaw
e_d (pred / act): 1.65 / 0.62 m
e_yaw (pred / act): 18.55 / 18.56 deg
Worst e_d
e_d (pred / act): -1.36 / 2.60 m
e_yaw (pred / act): 32.56 / -10.96 deg
Worst e_d
e_d (pred / act): 1.76 / -2.60 m
e_yaw (pred / act): -33.80 / 10.96 deg
Worst e_yaw
e_d (pred / act): -0.17 / -0.91 m
e_yaw (pred / act): 44.60 / -21.86 deg
Worst e_yaw
e_d (pred / act): 0.53 / -0.52 m
e_yaw (pred / act): -36.06 / 34.80 deg

Fig. 10 Examples of best and worst predictions for e_d and e_{yaw} from the GAN-augmented model. The top row displays the best predictions for each metric, while the bottom row highlights the worst, illustrating cases where the model struggles with accuracy.

[5] Farley, K. A., Williford, K. H., Stack, K. M., et al., "Mars 2020 Mission Overview," *Space Science Reviews*, Vol. 216, No. 8, 2020, p. 142. https://doi.org/10.1007/s11214-020-00762-y.

[6] Carpenter, K., Thoesen, A., Mick, D., Martia, J., Cable, M., Mitchell, K., Hovsepian, S., Jasper, J., Georgiev, N., Thakker, R., et al., "Exobiology Extant Life Surveyor (EELS)," *Earth and Space 2021*, American Society of Civil Engineers, 2021, pp. 328–338. https://doi.org/10.1061/9780784483374.033, URL https://ascelibrary.org/doi/abs/10.1061/9780784483374.033.

[7] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative Adversarial Networks," , 2014.

[8] Radford, A., Metz, L., and Chintala, S., "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," , 2016.

[9] Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Krishnan, D., "Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks," , 2017.

[10] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A., "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," , 2020.

[11] Liu, D., Li, Z., Wu, Z., and Li, C., "DT/MARS-CycleGAN: Improved Object Detection for MARS Phenotyping Robot," , 2023.

[12] Chen, K., Li, H., Li, C., Zhao, X., Wu, S., Duan, Y., and Wang, J., "An automatic defect detection system for petrochemical pipeline based on cycle-gan and yolo v5," *Sensors*, Vol. 22, No. 20, 2022, p. 7907.

[13] Rashid, Z. U., Stringer, J., Butta, N., Sharma, N., Khan, M., Khan, I., and Khan, S. U., "Data augmentation using generative adversarial networks (GANs) to synthesize images of blood cells," *Scientific Reports*, Vol. 9, No. 1, 2019, p. 16884. https://doi.org/10.1038/s41598-019-52737-x, URL https://www.nature.com/articles/s41598-019-52737-x.

[14] Ho, D., Rao, K., Xu, Z., Jang, E., Khansari, M., and Bai, Y., "RetinaGAN: An Object-aware Approach to Sim-to-Real Transfer," , 2021.

[15] Liu, D., Chen, Y., and Wu, Z., "Digital twin (dt)-cyclegan: Enabling zero-shot sim-to-real transfer of visual grasping models," *IEEE Robotics and Automation Letters*, Vol. 8, No. 5, 2023, pp. 2421–2428.

[16] Rojtberg, P., Pollabauer, T., and Kuijper, A., “Style-transfer GANs for bridging the domain gap in synthetic pose estimator training,” *2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, IEEE, 2020, pp. –. https://doi.org/10.1109/aivr50618.2020.00039, URL http://dx.doi.org/10.1109/AIVR50618.2020.00039.

[17] Yoneyama, R., Yamamoto, R., and Tachibana, K., “Nonparallel High-Quality Audio Super Resolution with Domain Adaptation and Resampling CycleGANs,” , 2023. URL https://arxiv.org/abs/2210.15887.

[18] Fabbri, C., Islam, M. J., and Sattar, J., “Enhancing underwater imagery using generative adversarial networks,” *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 7159–7165.

[19] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R., “Segment Anything,” , 2023.

[20] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P., “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, Vol. 86, No. 11, 1998, pp. 2278–2324. https://doi.org/10.1109/5.726791.

[21] Shorten, C., and Khoshgoftaar, T. M., “A survey on image data augmentation for deep learning,” *Journal of big data*, Vol. 6, No. 1, 2019, pp. 1–48.

[22] Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., Dickie, D. A., Hernández, M. V., Wardlaw, J., and Rueckert, D., “Gan augmentation: Augmenting training data using generative adversarial networks,” *arXiv preprint arXiv:1810.10863*, 2018.

[23] Karras, T., Aila, T., Laine, S., and Lehtinen, J., “Progressive Growing of GANs for Improved Quality, Stability, and Variation,” , 2018. URL https://arxiv.org/abs/1710.10196.

[24] Mirza, M., and Osindero, S., “Conditional Generative Adversarial Nets,” , 2014. URL https://arxiv.org/abs/1411.1784.

[25] Koenig, N., and Howard, A., “Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004, pp. 2149–2154.

[26] Epic Games, “Unreal Engine,” , 2019. URL https://www.unrealengine.com.

[27] Unity Technologies, “Unity,” , 2023. URL https://unity.com/, game development platform.

[28] Corporation, N., “NVIDIA Omniverse Isaac Sim,” NVIDIA Developer Website, 2023.

[29] Lipkis, R., Lopez-Francos, I. G., Vlastos, P. G., and Agogino, A., “AmesDT: Digital Twin and Autonomy Validation Environment,” *Proceedings of the AIAA SciTech Forum*, American Institute of Aeronautics and Astronautics (AIAA), 2025, pp. –.

[30] Community, B. O., *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org.

[31] Riegel, J., Mayer, W., and van Havre, Y., “FreeCAD,” , 2017. URL http://www.freecadweb.org, available from http://www.freecadweb.org.

[32] Shah, S., Dey, D., Lovett, C., and Kapoor, A., “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,” *Field and Service Robotics*, 2017, pp. –. URL https://arxiv.org/abs/1705.05065.

[33] Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P., “Sim-to-Real Transfer of Robotic Control with Dynamics Randomization,” *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. –. https://doi.org/10.1109/icra.2018.8460528, URL http://dx.doi.org/10.1109/ICRA.2018.8460528.

[34] Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., Levine, S., and Vanhoucke, V., “Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping,” , 2017.

[35] Volpi, R., Larlus, D., and Rogez, G., “Continual Adaptation of Visual Representations via Domain Randomization and Meta-learning,” , 2021. URL https://arxiv.org/abs/2012.04324.

[36] Lopez-Francos, I. G., Mitchell, S. C., Lipkis, R., Vlastos, P. G., Mbaye, S., and Infeld, S. I., “A Model-Based Systems Engineering Approach for Developing an Autonomous Rover Testbed,” *AIAA 2023-1894*, AIAA, 2023, pp. –. https://doi.org/10.2514/6.2023-1894, URL https://doi.org/10.2514/6.2023-1894.

[37] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C., “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” , 2019. URL https://arxiv.org/abs/1801.04381.