Intelligent Deep Data Generation in 2D and 3D

by

Alexandra Carlson

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in The University of Michigan
2022

Doctoral Committee:

Associate Professor Matthew Johnson-Roberson, Co-Chair
Assistant Professor Ram Vasudevan, Co-Chair
Associate Professor Matias del Campo
Professor Jessy Grizzle
Assistant Professor Justin Johnson

Alexandra Carlson

askc@umich.edu

ORCID iD: 0000-0002-7628-7023

This thesis is dedicated to my family

ACKNOWLEDGEMENTS

I have many people to thank for their generous support and guidance throughout my graduate research career. First, a big thank you to my Ph.D. advisor, Professor Matt Johnson-Roberson, for providing me with many years of mentorship and support. I appreciate your constant encouragement and all of the guidance you have given me. Thank you to the members of my doctoral committee – Professor Jessy Grizzle, Professor Matias del Campo, Professor Justin Johnson, and Professor Ram Vasudevan – for your help and insightful comments. I greatly appreciate all of your time and effort throughout this process.

Thank you to my labmates in the Ford Center for Autonomous Vehicles (FCAV) lab. I am grateful for all of your help and support. I would also like to thank the members of the Architecture and Artificial Intelligence lab (AR2IL). I have really enjoyed my collaborations with AR2IL and have learned a lot from working with all of you. Beyond that, I am thankful to have colleagues that I can also call my friends.

I would like to give a big thank you to the Robotics faculty and staff. All of you have created an amazing program and environment for both learning and research. None of this would be possible without you.

Finally to my friends and family – thank you for everything that you do for me. I could not have gotten through this journey without your immense support and care.

This work was supported in part by a fellowship from the Robotics Institute at the University of Michigan, and by the Ford Motor Company via the Ford-UM Alliance.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Deep Convolutional Neural Networks, which are a family of biologically inspired machine vision algorithms, have become ubiquitous in perception systems for autonomous agents due to their ability to accurately perform complex tasks and learn internal models of the real visual world. The efficacy of these algorithms to emulate and even surpass the performance of biological vision systems on certain tasks makes them particularly unique computational tools. For example, neural network architectures can achieve higher-than-human performance on image classification tasks, suggesting that these algorithms extract visual representations of objects (both geometric and stylistic features) that are potentially better for classification in comparison to the ones extracted by their human counterparts. In fact, neural networks have been used with great success at generative modeling and synthesis of the visual world. Traditional computer graphics engines can generate photorealistic scenes, but can generate content that is only as detailed and accurate as their underlying physics/light transport functions. In contrast, generative networks, like GANs (generative adversarial networks), can create highly detailed scenes with little to no input information. However, there are so few constraints in the learning process of these methods that it is difficult to control which real world scene factors or physical models (e.g., low level pixel statistics) are incorporated into the generated objects during the synthesis process; there is no guarantee that these networks are comprehensive models of the visual manifold. Another consequence of this is that there is no way to navigate the network-generated scenes in the same kind of way

one can reliably navigate through a 3D game engine. To address this challenge, this thesis proposes an alternative solution: physically-based object manipulation and editing. To achieve this, generative frameworks are created by fusing physics-models, high level attribute-based models, and deep neural networks. The objective of these frameworks is to inject real world information into both 2D (image) and 3D (mesh) datasets. To validate that realistic, salient information is being modeled and transferred by the proposed frameworks, experiments are carried out using a variety of different perceptual quality and performance metrics as a measure of dataset realism and generalizability. The combination of physically-based constraints and deep features ultimately leads to hybrid model-based, data-driven solutions that achieve realistic image and object editing. This thesis ultimately demonstrates that the outputs of these proposed hybrid frameworks can be used to improve the performance of a wide variety of autonomous computer vision and design applications.

CHAPTER I

Introduction

1.1 Motivation

Society is currently under going a digital and software revolution. Software is pervasive, and has positively impacted a huge number of fields, particularly those that require complex solutions for vision-dominant perceptual tasks, like segmentation and detection, or tracking and modeling. These areas include AR/VR, gaming, perception systems for autonomous agents, and novel design tools. This revolution is driven by the recent and impressive advancements in machine learning and AI, as well as the increased availability of datasets on which to train these algorithms. The performance and generalizability of these algorithmic tools to the real world relies almost entirely on the quality of the dataset on which they are trained. The training dataset needs to capture enough variance in visual information so as to be a reasonable approximation of the real world, and it specifically needs to capture all information necessary to complete the desired task. These dataset requirements can hugely vary based upon the problem space; for example, for tasks like pedestrian and car detection in images, it is necessary for datasets to capture a large variety of pedestrians/cars to best represent all possible ways those objects can appear under different scene conditions, like weather and times of day. In contrast, for design

tools, datasets need to capture more abstracted concepts, like aesthetics, creativity and varying styles. Accounting for and capturing all different facets required for a task is highly non trivial. Due to to physical, financial, and time constraints, it is nearly impossible to create a truly representative dataset that unbiasedly samples all possible scene or object variation. Thus while in theory, datasets are supposed to be complete representations of the world, in practice, datasets are only able to capture a subset of possible real world scenarios, and thus, instead of helping us train algorithms that work in the real open world, they have become closed worlds unto themselves [137].

Ultimately, a key component of bridging this knowledge gap within datasets is to capture more authentic representations of the visual world within the training datasets for these algorithms. This has spurred a large body of research and development into solutions that address the issues of dataset bias and deficiency, which we discuss below.

1.2 Real Data Generation/Collection

As previously mentioned, real datasets can be inadequate for the training and testing of neural networks because they are too small to capture all the complexity of the real world. Naively one might think to simply collect more real data. However, there are significant acquisition, time, and financial costs that are huge barriers to this solution. To get a better understanding of these costs, consider the following example for autonomous vehicle datasets: the Oxford Robot Car dataset [90]. It took over a year to collect (driving the same 10km loop sometimes multiple times a day), captured in all weather conditions, including heavy rain, night, direct sunlight and snow. It amounts to approximately 20 million images. It is an ideal dataset. At

first glance, the cost to collect this dataset is the time taken to collect the dataset as well as the cost of acquiring and maintaining the car and sensor rig. However, there is another cost to factor: labeling the dataset. While the time cost of labeling depends upon several factors, including the desired quality of the labels and the particular crowdsourcing protocol used, it takes approximately 1 min per image for bounding box labels [104]. For more complex labeling, such as pixel level semantic segmentation, it can take a human 60-90 minutes to label a single image [29]. This means for labeling Oxford Robot dataset for bounding boxes would take just over 300k man hours (which is roughly 34 years). For semantic segmentation, it would take anywhere from 20mil to 30mil hours, which is several thousand years of labeling time. Then there is the monetary cost of labeling; a popular crowdsourcing tool for labeling is Amazon Turk. For labeling tasks rated 'tough', it pays users approximately $5 per hour [104], which yeilds around $1.5 million for bounding box labels and $100 million for segmentation labels. So, due to the cost of acquisition and the cost of labeling, simply 'collecting more data' can present an insurmountable barrier. Thus, an alternative to real data collection is highly desirable.

1.3 Synthetic Data Generation

An alternative solution is generating synthetic data. With the advancements in gaming and rendering engines, we have the capabilities of generating incredibly realistic datasets that capture significantly more visual information that real datasets with no labeling cost. Another desirable feature of rendering and gaming engines is that we can control what visual information is represented within the generated dataset.

In fact, for image-based tasks, a large number of photorealistic, simulated datasets

have been created in recent years [113, 147, 164, 41, 93, 115, 49, 99, 1, 59], each varying in quality. An example is the Virtual KITTI [49] and Virtual KITTI2 datasets [18], which are designed to add more variance to the KITTI world. Another more photorealistic example is the datasets generated from Grand Theft Auto [113], which capture significantly more complex visual information than the virtual KITTI datasets. Examples of these datasets are shown in Figure 1.1. However, there are

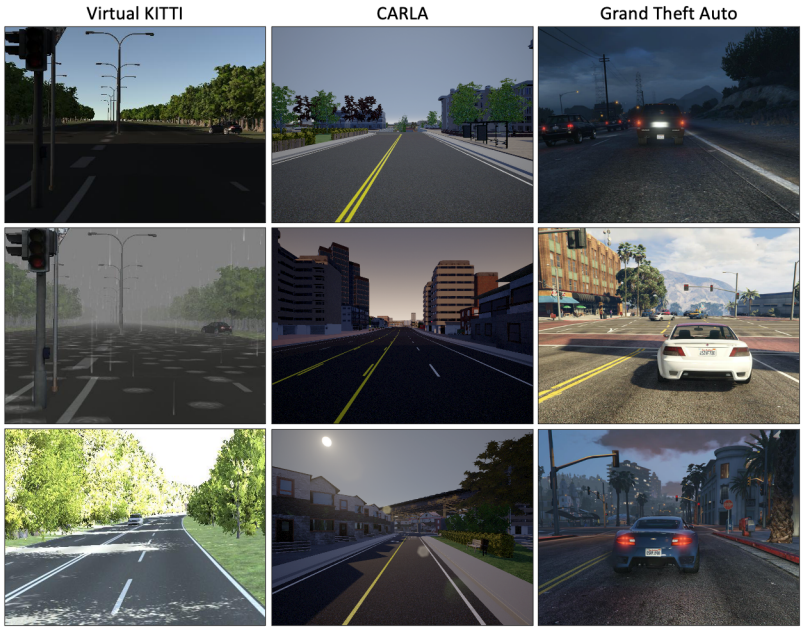

Figure 1.1: Examples of different synthetically generated datasets. Left hand column shows examples of Virtual KITTI [49], middle columns shows image examples of CARLA [41], and right hand row shows examples of GTA [113].

several drawbacks with synthetic datasets. First, these datasets are very expensive to generate. To resemble the real world, they require artists to carefully model specific environments and 3D objects in great detail, and two they require significant rendering/computational power to generate [135, 138]. For example , the hyper photorealistc movie, Jungle Book (released in 2016) took 30 million render hours, 19 hours per frame, and 800 artist-years of effort [45]. This leads to another significant drawback: the render quality of the output of these rendering engines. Due to the

computational intractability of highly photorealistic datasets, there is a clear, observable difference between most synthetic datasets and real data. These differences can be attributed to discrepancies between physics simulators and the real world, whether it is in a behavior mismatch from improperly tuned parameters of the simulator's physics engine, to unmodeled (or hard to model) physical effects, like shading and lighting, or surface texture. Low-fidelity simulated sensors like image renderers, or poorly modeled scene and object geometry can also lack the noise, detail and richness that exists in real world analogs. [135]. These drawbacks negate the primary selling point of synthetic data, namely, that arbitrarily large amounts of labeled data are available essentially for free.

1.4 Deep-learning and Neural Data Generation

Deep Convolutional Neural Networks, which are a family of biologically inspired machine vision algorithms, have become ubiquitous in visual perception systems for autonomous agents due to their ability to accurately perform complex tasks and learn internal models of the real visual world. The efficacy of these algorithms to emulate and even surpass the performance of biological vision systems on certain tasks makes them particularly unique computational tools. For example, neural network architectures can achieve higher-than-human performance on image classification tasks [72], suggesting that these algorithms extract visual representations of objects (both geometric and stylistic features) that are potentially better for classification in comparison to the ones extracted by their human counterparts. In fact, Neural networks have been used with great success at generative modeling and image synthesis because of their ability to extract and model stylistic, geometric, and semantic properties of the visual world. Generative networks, specifically generative

adversarial networks (GANs), can create highly detailed objects and scenes, both in 2D and 3D, with little to no input information. GANs are comprised of two neural networks that are trained adversarially: a generator network generates/renders candidate data points, and the discriminator network that evaluates them based upon real data points. This training regime allows the generator network to learn the underlying distribution of the real world scene elements and physical processes that are captured in its training dataset without them being explicitly defined, unlike standard renderers. The unconstrained nature of neural networks, coupled with their great ability to model high dimensional, complex functions, allows these networks to potentially learn better ways to model realistic scene components. In fact, these types of neural networks have been applied almost as learned rendering engines to transform input image datasets into new seasons, differing times of day, and artistic styles [61, 68, 82, 87, 9, 22, 169, 24, 50]. They have also been applied to generative, stylistic modeling in 3D mesh, pointcloud, and voxel datasets [146, 145, 51, 134].

However, their unconstrained nature is a double edged sword. Often what we see in practice is that there are so few constraints in the learning process of these methods that it is difficult to control which real world scene factors or physical models (e.g., low level pixel statistics) are incorporated into the generated objects during the synthesis process; there is no guarantee that these networks are comprehensive models of the visual manifold. Furthermore, without more constraints built into the synthesis process, these networks can negatively transfer features that are not realistic. An example of this in image datasets is shown in Figure 1.2. The virtual images from GTA are on the left; on the right are the corresponding images translated by CycleGAN [169, 50]. Furthermore, while it appears that global visual features like color cast are modeled by the network, there is no guarantee that realistic, local pixel

statistics are modeled. Another consequence of this is that there is no way to navigate the network-generated scenes the same kind of way you can reliably navigate through a 3D game engine or renderer. This means that we cannot use these networks to inject specific visual factors into data; its all and whatever the network has learned that gets injected. Ultimately, these neural rendering frameworks introduced artifacts into data that corrupt the realism of the output.

Figure 1.2: The virtual images from GTA are on the left; on the right are the corresponding images translated by CycleGAN [169] into the "real world", specifically the Cityscapes dataset [29]. The negatively transferred features are highlighted in the red boxes

1.5 Problem Statement

This thesis seeks solutions to the following challenges in intelligently modeling visual information in datasets:

1. Real world data is prohibitively expensive to collect and label
2. Synthetic/classically-rendered data isn't realistic enough to capture the detail of the real world and is also expensive to generate
3. Generative neural networks trained to as renderers introduce unrealistic artifacts due to their unconstrained nature

1.6 Contributions

We propose that we can solve the above issues by more efficiently capturing the salient factors in the distribution of visual information in datasets in a way that will allow for intelligent and controllable dataset generation. The main objective of this dissertation is to improve our ability to model and manipulate visual information captured in data by fusing neural networks with cross-disciplinary knowledge and physics-based models of environmental conditions in an effort to better capture and model real world information into both 2D (image) and 3D (mesh) datasets. The combination of physically and expert knowledge-based constraints and deep neural networks ultimately leads to hybrid model-based, data-driven solutions that achieve realistic image and object generation. This thesis ultimately demonstrates that the outputs of these proposed hybrid frameworks can be used to improve the performance of a wide variety of computer vision applications.

This thesis seeks to build upon existing methods with the following contributions that propose intelligent, controllable dataset generation frameworks as solutions to different problems in domain transfer, novel design frameworks, and scene modeling, respectively:

1. Novel optimization framework that mimics image formation in a camera and transfers realistic sensor effects from real data to synthetic image datasets in the effort to overcome the synthetic-reality gap. We demonstrate that this form of intelligent, physically-based image augmentation is an effective way to boost performance of models trained in the purely synthetic domain and evaluated in real world domains (Chapter II)

2. Novel optimization framework that leverages Graph Neural networks to transfer

the sensibilities and style of a designer onto user-input 3D meshes (Chapter III)

3. A novel neural network that couples multi-sensor outputs (RGB, lidar) to achieve relighting of complex, outdoor scenes (Chapter IV)

CHAPTER II

Sensor Transfer:

2.1 Introduction

In this chapter, the aim is to present a physically-based image generation optimization framework that offers a viable solution to the problem of the synthetic to real domain gap. Synthetic datasets are designed to contain numerous spatial and environmental features that are found in the real domain: images captured during different times of day, in various weather conditions, and in structured urban environments. However, in spite of these shared features and high levels of photorealism, images from synthetic datasets are noticeably stylistically distinct from real images. Figure 2.1 shows a side-by-side comparison of two of widely-used real benchmark vehicle datasets, KITTI [52, 48], Cityscapes [30], and a state-of-the-art synthetic dataset, GTA *Sim10k* [114, 66]. These differences can be quantified; a performance drop is observed between training and testing deep neural networks (DNNs) between the synthetic and real domains [66]. This suggests that real and synthetic datasets differ in their global pixel statistics. Domain adaptation methods attempt to minimize such dissimilarities between synthetic and real datasets that result from an uneven representation of visual information in one domain compared to the other. Recent domain adaptation research has focused on learning salient visual features

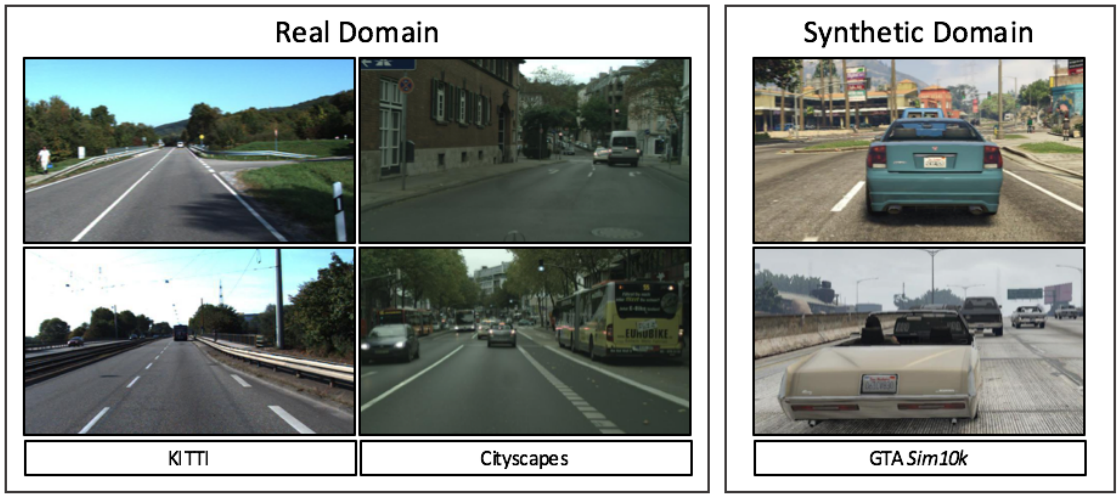

Figure 2.1: A comparison of images sampled from the real domain, KITTI Benchmark dataset (shown in the left hand column), images taken from the Cityscapes dataset (shown in the center column), and images from GTA *Sim10k* dataset (shown in the right hand column). Note that each dataset has a distinct visual style, specifically differing color cast, brightness, and blur.

from real data – specifically scene lighting, scene background, weather, and occlusions – using generative adversarial frameworks in an effort to better model the representation of these visual elements in synthetic training sets [162, 141, 118]. However, little work has focused on modelling realistic, physically-based augmentations of synthetic data. Carlson et al. [20] demonstrate that randomizing across the sensor domain significantly improves performance over standard augmentation techniques. The information loss that results from the interaction between the camera model and lighting in the environment is not generally modelled in rendering engines, despite the fact that it can greatly influence the pixel-level artifacts, distortions, and dynamic range, and thus the global visual style induced in each image [53, 31, 47, 7, 128, 38, 39].

In this chapter, we present work from [21] and [20] to work towards closing the gap between real and synthetic data domains by transferring the style of real images that results from the presence of various sensor effects to synthetic images. We propose a novel learning framework that performs *sensor transfer* on synthetic data. That

is, the network learns to transfer the real sensor effect domain – blur, exposure, noise, color cast, and chromatic aberration – to synthetic images via a generative augmentation network.

Our main contributions include the following:

1. a physically-based image augmentation framework that transfers visual sensor effects between datasets,

2. we present experiments that demonstrate that augmenting relatively small labeled datasets using the proposed *sensor transfer* generates more robust and generalizable training datasets that improve the performance of DNNs for object detection and semantic segmentation tasks in urban driving scenes for both real and synthetic visual domains.

The chapter is organized as follows. Section 2.2 describes related work in synthetic to real domain adaptation as well as data augmentation. Section 2.3 describes our proposed camera model and sensor transfer learning framework. Section 2.4 presents experiments and results on benchmark datasets. Section 2.5 concludes this chapter.

2.2 Related Work

The work presented in this chapter focuses on augmenting the training data directly so that it can be applied to any task or input into any deep neural network regardless of the architecture. Zhang et al. 2017 [165] demonstrate that the level of photorealism of the synthetic training data directly impacts the robustness and performance of the deep learning algorithm when tested on real data across a variety of computer vision tasks. However, it remains unclear what features of real data are necessary for this performance gain, or what parts of rendering pipelines should be modified to bridge the synthetic to real domain gap. Much work in the fields

of data augmentation and learned rendering pipelines have proposed methods that shed light on this topic, and are summarized below.

2.2.1 Domain Randomization

Recent work on domain randomization seeks to bridge the sim-to-real domain gap by generating synthetic data that has sufficient random variation over scene factors and rendering parameters such that the real data falls into this range of variation, even if the rendered data does not appear photorealistic. Such scene factors include such as textures, occlusion levels, scene lighting, camera field of view, and uniform noise, and have been applied to vision tasks in complex indoor and outdoor environments [135, 138]. The drawback of these techniques is that they only work if they sample the visual parameters spaces finely enough, and create a large enough dataset from a broad enough range of visual distortions to encompass the variation observed in real data. This can result in intractably large datasets that require significant training time for a deep learning algorithm. While we also aim to achieve robustness via an augmentation framework, we can use smaller datasets to achieve state-of-the-art performance because our method is learning how to augment synthetic data with salient visual information that exists in real data. Note that, because our work focuses on image augmentation outside of the rendering pipeline, it could be used in addition to domain randomization techniques.

2.2.2 Optimizing Augmentation

In contrast to domain randomization, task-dependent techniques have been proposed to achieve more efficient data augmentation by learning the type and number of image augmentations that are important for algorithm performance. State-of-the-art methods [108, 32, 76] in this area treat data augmentation as a form of

network regularization, selecting a subset of augmentations that optimize algorithm performance for a given dataset and task as the algorithm is being trained. Unlike these methods, we propose that data augmentation can function as a domain adaptation method. Our learning framework is task-independent, and uses physically based augmentation techniques to investigate the visual degrees of freedom (defined by physically-based models) necessary for optimizing network performance from the synthetic to real domain.

2.2.3 Image-to-Image Translation for Domain Adaptation

Impressive advances have been made in both paired and unpaired image-to-image translation [169, 65, 168, 63, 61, 86] to bridge a variety of domain gaps, including season-to-season, night-to-day, and sim-to-real. However, image-to-image translation performed between image sets with complex, varied environments often introduces unrealistic distortion artifacts into the underlying structure of the scene. This can yield poor performance for visual tasks such as object detection and semantic segmentation [42]. In contrast, the proposed method does not alter the spatial information in the scene, and instead translates images from one domain to another constrained by physically-based image augmentation.

2.2.4 Learned Rendering Pipelines for Domain Adaptation

Several studies have proposed unsupervised, generative learning frameworks that either take the place of a standard rendering engine [141] or complement the rendering engine via post-processing [127, 124, 60] in order to model relevant visual information directly from real images with no dependency on a specific task framework. Both [141] and [60] are applied to complex outdoor image datasets, but are designed to learn distributions over simpler spatial features in real images, specifi-

cally scene geometry. Other methods, such as [127, 124], attempt to learn low-level pixel features. However, they are only applied to image sets that are homogeneously structured and low resolution. This may be due to the sensitivity of training adversarial frameworks. Our work focuses specifically on modeling the camera and image processing pipeline rather than scene elements or environmental factors that are specific to a given task. Our method can be applied to high resolution images of complex scenes.

2.2.5 Impact of Sensor Effects on Deep Learning

Recent work has demonstrated that elements of the image formation and processing pipeline can have a large impact upon learned representation for deep neural networks across a variety of vision tasks [67, 36, 38, 39]. The majority of methods propose learning techniques that remove these effects from images [36]. As many of these sensor effects can lead to loss of information, correcting for them is non-trivial, potentially unstable, and may result in the hallucination of visual structure in the restored image. In contrast, Carlson et al. [20] demonstrate that significant performance boosts can be achieved by augmenting images using physically-based, sensor effect domain randomization. However, their method requires hand-tuning/evaluation of the visual quality of image augmentation. This human-in-the-loop dependence is inefficient and difficult to scale for large synthetic datasets, and the evaluated visual image quality is subjective. Rather than removing these effects, randomly adding them in, or manually adding them in via human-in-the-loop, our method learns the the style of sensor effects from real data and transfers this *sensor style* to synthetic images to bridge the synthetic-to-real domain gap.

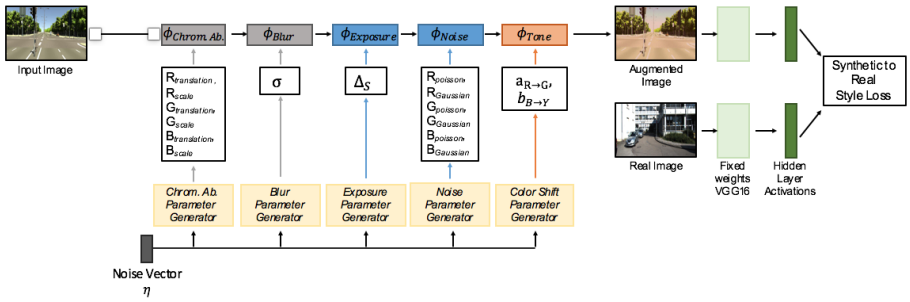

Figure 2.2: The schematic of the proposed sensor transfer network structure. The style loss trains the sensor effect parameter generators (represented as the yellow boxes) to select parameters that transform the input synthetic images based upon how the sensor effect augmentation functions alter the style of the real data domain. This effectively transfers 'sensor style' of the target dataset to the source dataset.

2.3 Methods and Proposed Approach

The objective of the sensor transfer network is to learn the the optimal set of augmentations that transfer sensor effects from a real dataset to a synthetic dataset. Our complete Sensor Transfer Network is shown in Figure 2.2.

2.3.1 Sensor Effect Augmentation Pipeline

We adopt the sensor effect augmentation pipeline from [20]. This is the backbone of the Sensor Transfer Network. Refer to [20] for a detailed discussion of each function and its relationship to the image formation process in a camera. We briefly describe each sensor effect augmentation function below for completeness. The sensor effect augmentation pipeline is a composition of chromatic aberration, Gaussian blur, exposure, pixel-sensor noise, and post-processing color balance augmentation functions:

$$I_{aug.} = f_{color}(f_{noise}(f_{exposure}(f_{blur}(f_{chrom.ab.}(I))))) \tag{2.1}$$

Chromatic Aberration

To model lateral chromatic aberration, we apply translations (t_x, t_y) in 2D pixel

space to each of the color channels of an image. To model longitudinal chromatic aberration, we scale the green color channel relative to the red and blue channels of an image by a value S. We combine these parameters into an affine transformation on each pixel in color channel of the image. The augmentation parameters learned for this augmentation function are S, the red channel translations R_x and R_y, the green channel translations G_x and G_y, and the blue channel translations B_x and B_y.

Blur

We implement out-of-focus blur, which is modeled by convolving the image with a Gaussian filter [27].We fix the window size of the kernel to 9.0. The augmentation parameter learned for this augmentation function is the standard deviation σ of the kernel.

Exposure

We implement the exposure density function developed in [15, 95]:

$$I = f(S) = \frac{255}{1 + e^{-A \times S}} \tag{2.2}$$

where I is image intensity, S models the incoming light intensity, and A is a constant value that describes image contrast. We set A to 0.85. This model is used to re-expose an image as follows:

$$S' = f^{-1}(I) + \Delta S \tag{2.3}$$

$$I_{exp} = f(S') \tag{2.4}$$

The augmentation parameters learned for this augmentation function are ΔS to model changing exposure, where a positive ΔS relates to increasing the exposure,

and a negative value indicates decreasing exposure.

Noise

We use the Poisson-Gaussian noise model proposed in [47]:

$$I_{noise}(x, y) = I(x, y) + \eta_{poiss}(I(x, y)) + \eta_{gauss} \tag{2.5}$$

where $I(x, y)$ is the ground truth image at pixel location (x, y), η_{poiss} is the signal-dependent poisson noise, and η_{gauss} is the signal-independent gaussian noise. The augmentation parameters learned for this augmentation function are the η_{poiss} and η_{gauss} for each color channel, for a total of six parameters.

Post-processing

We model post-processing techniques done by cameras, such as white balancing or gamma transformation, by performing linear translations in LAB color space [62, 8]. The augmentation parameters learned for this augmentation function the are translations in the a (red-green)and b (blue-yellow) channels in normalized LAB space.

2.3.2 Training the Sensor Transfer Network

A high-level overview of a single training iteration for a single sensor effect is given in Figure 2.3. Each sensor effect augmentation function has its own parameter generator network. The training objective for each of these networks is to learn the distribution over its respective augmentation parameter(s) based upon real data. Each generator network is a two-layer, fully connected neural network. The following steps are required to perform a single training iteration of the Sensor Transfer Network using a single synthetic image. First, a 200 dimensional uniform noise vector, η,

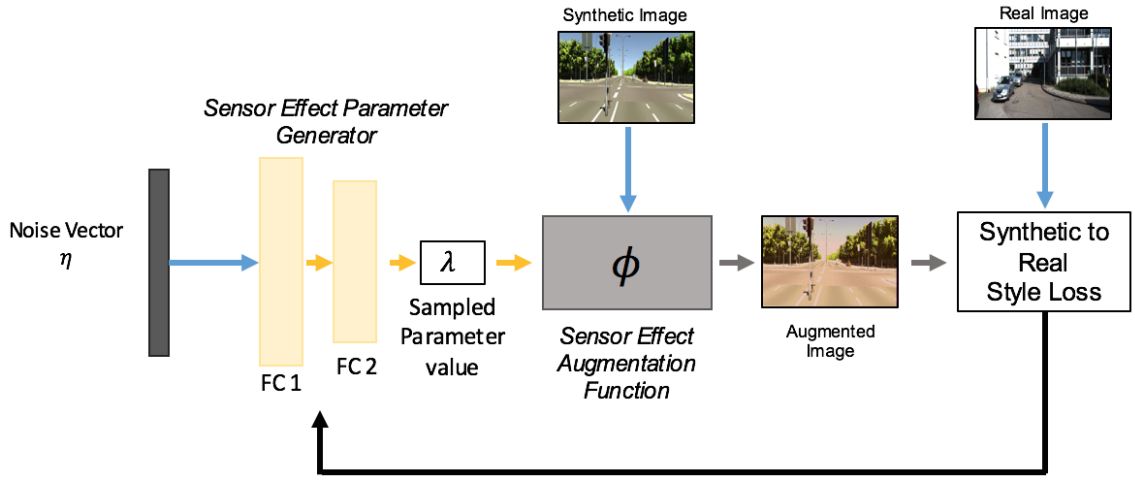

Figure 2.3: A detailed schematic of how the training process occurs for a single sensor effect function. A 200 dimensional uniform noise vector (sampled from the range -1 to 1) is generated for a given input synthetic image. The uniform noise vector is input into the fully connected neural network that constitutes the parameter generator, which outputs sampled value(s) for the respective sensor effect augmentation function. The sampled parameter value(s) and the input synthetic image are fed into the sensor effect augmentation function, which outputs an augmented synthetic image. The style loss is calculated between the augmented synthetic image and a real image. This style loss is then backpropagated through the augmentation functions to train the parameter generator to select parameters that reduce the style differences between the real and augmented synthetic images.

is generated and paired with the input synthetic image. The noise vector η is input into each separate generator network. Each generator network consists of two fully connected layers that together project η into its respective sensor effect parameter space. For example, the blur parameter generator will map the η to a value in the σ parameter space. The output sampled parameters, paired with the input synthetic image are then input into the augmentation pipeline, which outputs an augmented synthetic image. This augmented synthetic image is then paired with a real image, both of which are input to the loss function.

We employ a loss function similar to the one used in Johnson et. al [65]. We assume that the layers of the VGG-16 network [126] trained on ImageNet [34] encode relevant style information for salient objects We fix the weights of the pretrained VGG-16 network, and use it to project real and augmented synthetic images into the

hidden layer feature spaces. We calculate the style loss, given in Eqn. 2.6, and use this as the training signal for the parameter generators.

$$L_{style}(y, \hat{y}) = \Sigma_j \|G_j^\theta(y) - G_j^\theta(\hat{y})\|^2 \tag{2.6}$$

In the above equation, y is a real image batch, $\hat{y}$ is an augmented synthetic image batch, $G_j^\theta(y)$ is the Gram matrix of the feature map $\theta(y)$ of hidden layer j of the pretrained VGG-16 network, and $G_j^\theta(\hat{y})$ is the corresponding quantity for augmented synthetic images. Through performance-based ablation studies, we found that $j = 10$ gives the best performance, so the style loss is calculated for the first ten layers of VGG-16. Once calculated, the style loss is backpropagated through the sensor effect augmentation functions to train the sensor effect parameter generators. The above process is repeated with images from the synthetic and real datasets until the style loss has converged.

We train the sensor effects generators concurrently to learn the joint probability distribution over the sensor effect parameters. This is done to capture the dependencies that exist between these effects in a real camera. Once training is complete, we can fix the weights of the parameter generators, and use them to sample learned parameters to augment synthetic images. Table 2.1 shows the statistics of the learned distributions for sensor effect parameters of different real datasets. See Section **??** for analysis and discussion of the learned parameters. Note that style loss was chosen because it is independent of spatial structure of an image. In effect, the augmentation parameter generators learn to sample the distributions of sensor effects in real data as constrained by the style of the real image domain.

Figure 2.4: Qualitative comparison of unaugmented GTA*Sim10k* in the first column, Sensor Transfer augmented GTA*Sim10k* images in the second column, MUNIT augmented GTA*Sim10k* in the third colum, UNIT-augmented GTA*Sim10k* in the fourth column, and CycleGAN-augmented GTA*Sim10k* in the last column. The first two rows are GTA*Sim10k* translated to the KITTI domain, and the second two rows are GTA*Sim10k* translated into the Cityscapes domain. Note that, for the Sensor Transfer augmented images, the primary sensor effect transferred in GTA*Sim10k*→Cityscapes augmentation is decreased exposure, whereas the primary sensor effects transferred in GTA*Sim10k*→KITTI augmentation is a blueish hue and increased exposure. In comparison, images augmented using the image-to-image translation networks lose a significant amount of spatial information. These methods also cannot handle night time images as well as the proposed method.

2.4 Experiments

2.4.1 Experimental Setup

To verify that the proposed method can transfer the sensor effects of different datasets, we train Sensor Transfer Networks using the following synthetic and real benchmark datasets: GTA*Sim10k* [66] is comprised of 10,000 highly photorealistic synthetic images collected from the Grand Theft Auto (GTA) rendering engine. It captures different weather conditions and time of day. The Cityscapes [30] training image set is comprised of 2975 real images collected in over 50 cities across Germany. The KITTI training set [52] is comprised of 7481 real images collected in Karlshue, Germany. We train a Sensor Transfer Network to transfer the sensor style of the KITTI training set to GTA*Sim10k*, which is referred to as GTA*Sim10k*→KITTI. We also train a Sensor Transfer Network to transfer the sensor style of the Cityscapes training set to GTA*Sim10k*, which is referred to as GTA*Sim10k*→Cityscapes. To train each Sensor Transfer Network, we use a batch size of 1 and learning rate of $2e^{-5}$. We trained each network for 4 epochs, until the sensor effect parameters converged. For all experiments, we compare our results to the Sensor Effect Domain Randomization [20] of GTA*Sim10k* as a baseline measure to ensure that the transfer of effects is viable over sampling. To generate the Sensor Effect Domain Randomization augmentations, we used the same human-selected parameter ranges as in [20]. To benchmark our method against other, image-based domain adaptation methods, we use the state-of-the-art image-to-image translation methods CycleGAN [169], UNIT [86], and MUNIT [61] as additional baseline measures. Each of the CycleGAN, UNIT, and MUNIT image-to-image translation networks were trained to transfer GTA*Sim10k* to Cityscapes, and separately to transfer GTA*Sim10k* to KITTI. Each network was trained using either the default hyperparameters provided in the respective paper(s)

Table 2.1: Learned sensor effect parameters for GTA*Sim10k*→Cityscapes and GTA*Sim10k*→KITTI, and the Sensor Effect Domain Randomization parameters from Carlson et al. [20]. Note that for the Sensor transferred parameters in the first two rows, the mean and standard deviation of each sensor effect parameter value is given in the convention $\mu \pm \sigma$. For the Carlson et al. [20] Sensor Effect Domain Randomization parameters, given in the final row, the minimum and maximum of the human selected range is provided. Quantitatively, the GTA*Sim10k*→KITTI increases image exposure, adds chromatic aberration, noise, and adds a blue color cast. For GTA*Sim10k*→Cityscapes, image exposure is decreased, adds chromatic aberration, a higher level of noise is added, and slight yellow-blue color cast is applied.

Proposed Method GTA*Sim10k*→Cityscapes Sensor Effect Parameters

Chrom. Ab.	Blur	Exposure	Noise	Post-processing
G_{scale}: $0.999 \pm 2.398e^{-5}$	σ: $0.718 \pm 1.34e^{-13}$	ΔS: -0.273 ± 0.0249	$R_{gauss.}$: $1.0e^{-6} \pm 1.382e^{-18}$	a:$-0.002 \pm 5.239e^{-4}$
R_{tx}: $0.004 \pm 6.221e^{-5}$			$R_{poiss.}$: $1.0e^{-6} \pm 1.382e^{-18}$	b: $-0.0116 \pm 4.727e^{-4}$
R_{ty}: $0.007 \pm 5.511e^{-5}$			$G_{gauss.}$: $5.41 \pm 4.249e^{-4}$	
G_{tx}: $0.005 \pm 1.111e^{-5}$			$G_{poiss.}$: $1.15e^{-2} \pm 7.913e^{-5}$	
G_{ty}: $0.006 \pm 4.718e^{-5}$			$B_{gauss.}$: $1.0e^{-6} \pm 1.382e^{-18}$	
B_{tx}: $0.006 \pm 5.793e^{-5}$			$B_{poiss.}$: $6.8e^{-4} \pm 4.608e^{-6}$	
B_{ty}: $-5.052 \pm 1.16e^{-4}$				

Proposed Method GTA*Sim10k*→KITTI Sensor Effect Parameters

Chrom. Ab.	Blur	Exposure	Noise	Post-processing
G_{scale}: $1.001 \pm 6.425e^{-5}$	σ: $0.941 \pm 5.173e^{-7}$	ΔS: 0.0823 ± 0.003	$R_{gauss.}$: $9.5e^{-3} \pm 3.713e^{-4}$	a:$-0.0131 \pm 5.426e^{-4}$
R_{tx}: $1.134e^{-4} \pm 9.416e^{-5}$			$R_{poiss.}$: $3.07e^{-2} \pm 1.295e^{-3}$	b: $-0.0882 \pm 3.25e^{-3}$
R_{ty}: $-0.0013 \pm 6.874e^{-5}$			$G_{gauss.}$: $4.5e^{-3} \pm 2.005e^{-4}$	
G_{tx}: $-4.67e^{-4} \pm 5.65e^{-5}$			$G_{poiss.}$: $2.62e^{-2} \pm 1.111e^{-3}$	
G_{ty}: $-0.0014 \pm 7.228e^{-5}$			$B_{gauss.}$: $2.65e^{-2} \pm 1.111e^{-3}$	
B_{tx}: $-0.003 \pm 1.245e^{-4}$			$B_{poiss.}$: $4.47e^{-2} \pm 1.187e^{-3}$	
B_{ty}: $-5.16e^{-5} \pm 1.096e^{-4}$				

Carlson et al. [20] Sensor Effect Domain Randomization Parameters

G_{scale}: 0.998-1.002	κ_{size}: 3-11	ΔS: -0.6-1.2	$R_{gauss.}$: 0.00-0.05	a: -10.0-10.0
R_{tx}: -0.003-0.003	σ: 0.0-3.0		$G_{gauss.}$: 0.00-0.05	b: -10.0-10.0
R_{ty}: -0.003-0.003			$B_{gauss.}$: 0.00-0.05	
G_{tx}: -0.003-0.003			$R_{poiss.}$: 0.00-0.05	
G_{ty}: -0.003-0.003			$G_{poiss.}$: 0.00-0.05	
B_{tx}: -0.003-0.003			$B_{poiss.}$: 0.00-0.05	
B_{ty}: -0.003-0.003				

or until the networks converged.

2.4.2 Evaluation of Learned Sensor Effect Augmentations

Qualitatively, from observing Figure 2.1, KITTI images feature more pronounced visual distortions due to blur, over-exposure, and a blue color tone. Cityscapes, on the other hand, has a more under-exposed, darker visual style.

Figure 2.4 shows examples of unaugmented GTA*Sim10k* images in comparison to those same images augmented by the proposed Sensor Transfer network and baseline image-to-image translation networks. When compared to Figure 2.1, it does appear that, for both the sensor transfer of GTA*Sim10k*→KITTI and GTA*Sim10k*→Cityscapes,

realistic aspects of exposure, noise, and color cast are transferred to GTA*Sim10k*. The statistics of the learned parameter values are given in Table 2.1. In general the selected parameter values generate augmented synthetic images with style that matches the real datasets. We hypothesize that the color shift for GTA*Sim10k*→Cityscapes is not as strong as GTA*Sim10k*→KITTI because there is a more even distribution of sky and buildings in Cityscapes, where as KITTI has a significant number of instances of sky. Interestingly, the blur parameter, σ, did not converge and was pushed towards zero for both GTA*Sim10k*→Cityscapes and GTA*Sim10k*→KITTI. This suggests that Gaussian blur does not match the blur captured by style of real images. Further research could consider more accurate models of blur, such as motion blur.

2.4.3 Impact of Learned Sensor Transformation on Object Detection for Benchmark datasets

Table 2.2: Results of the sensor effects augmentations on Faster R-CNN object detection performance. The percent change for CycleGAN [169], UNIT [86], MUNIT [61], the Carlson et al. [20] and proposed method are calculated relative to the full, unaugmented baseline datasets.

Training Dataset	Tested on KITTI	
Augmentation Method	AP_{Car}	Gain
Baseline	51.01	—
CycleGAN [169]	48.75	↓-2.25
UNIT [86]	51.21	↑0.21
MUNIT [61]	45.50	↓-5.51
Carlson et al. [20]	48.94	↓-2.07
Proposed Method	**52.67**	↑**+1.66**

Training Dataset	Tested on Cityscapes	
Augmentation Method	AP_{Car}	Gain
Baseline	30.13	—
CycleGAN [169]	29.30	↓-0.83
UNIT [86]	28.05	↓-2.08
MUNIT [61]	26.20	↓-3.93
Carlson et al. [20]	34.89	↑+4.76
Proposed Method	**35.48**	↑**+5.35**

To evaluate if the Sensor Transfer Network is adding in salient visual information for vision tasks in the real image domain, we train an object detection neural

network on the unaugmented and augmented synthetic data and evaluate the performance of the object detection network on the real data domains, KITTI and Cityscapes. We chose to use Faster R-CNN as our base network for 2D object detection [112]. Faster R-CNN achieves relatively high performance on the KITTI benchmark dataset. Many state-of-the-art object detection networks that improve upon these results still use Faster R-CNN as their base architecture.

We compare Faster R-CNN networks trained on the proposed method to Faster R-CNN networks trained on unaugmented GTA*Sim10k*, GTA*Sim10k* augmented using the Sensor Transfer Domain Randomization from Carlson et al., GTA*Sim10k* augmented using CycleGAN, GTA*Sim10k* augmented using UNIT, and GTA*Sim10k* augmented using MUNIT. To create augmented training datasets, we combine the unaugmented GTA*Sim10k* with varying amounts of augmented GTA*Sim10k* data. For all datasets, both augmented and unaugmented, we trained each Faster R-CNN network for 10 epochs using two Titan X Pascal GPUs in order to control for potential confounds between performance and training time. We evaluate the Faster R-CNN networks on either the KITTI training dataset or the Cityscapes training dataset depending on the Sensor Transfer Network used for training dataset augmentation. Each dataset is converted into Pascal VOC 2012 format to standardize training and evaluation, and performance values are the VOC AP50 reported for the car class [44].

Table 2.2 shows the object detection results for the proposed method in comparison to the image-to-image translation and domain randomization baselines. In general, the addition of sensor effect augmentations has a positive boost on Faster R-CNN performance for training on GTA*Sim10k* and testing on Cityscapes. Our proposed method, for both GTA*Sim10k*→Cityscapes and GTA*Sim10k*→KITTI, achieves

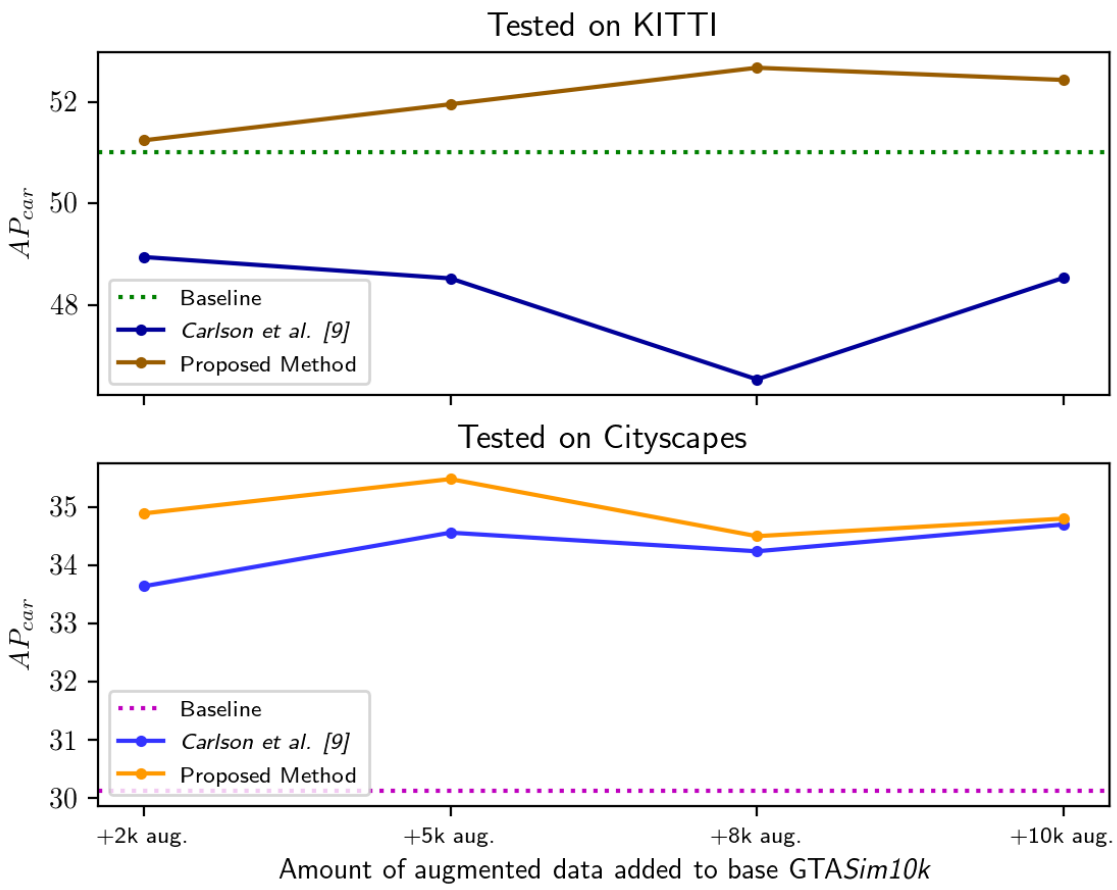

Figure 2.5: Results of the learned sensor effect augmentations on Faster R-CNN object detection performance. Note that higher performance can be achieved using smaller synthetic datasets augmented with the proposed method for both KITTI and Cityscapes.

the best performance over both the baseline and Sensor Effect Domain Randomization.

To evaluate the impact of Sensor Transfer on the number of synthetic training images required for maximal object detection performance, we trained Faster R-CNNs on datasets comprised of the 10k unagumented GTA*Sim10k* images combined with either 2k augmented images, 5k augmented images, 8k augmented images, or 10k augmented images. Figure 2.5 captures the effect of increasing number of augmentations on Faster R-CNN performance. We see that, when compared to the Sensor Transfer domain randomization method, fewer training images are required when using Sensor

Transfer augmentation for both GTA*Sim10k*→KITTI and GTA*Sim10k*→Cityscapes. Our results indicate that learning the augmentation parameters allows us to train on significantly smaller datasets without compromising performance. This demonstrates that we are more efficiently modeling salient visual information than domain randomization. Interestingly, the Sensor Effect Domain Randomization method does worse than baseline across all levels of augmentation when tested on KITTI. We expect that this is because human-chosen set of parameter ranges, which are shown in the bottom row of Table 2.1, do not generalize well when adapting GTA *Sim10k* to KITTI even though they may generate visually realistic images. One reason for this is that the visually realistic parameter ranges selected in [20] where chosen using a GTA dataset of all daytime images, whereas GTA*Sim10k* contains an even representation of daytime and nighttime images. This further demonstrates the importance of learning the sensor effect parameter distributions constrained by how they affect the styles of both the real and synthetic image datasets.

2.5 Conclusion

In this chapter, we presented a novel deep learning framework that transfers sensor artifacts from one dataset to another in a physically-based way. In general, the results presented in Section 2.4 show that the proposed Sensor Transfer Network reduces the synthetic to real domain gap more effectively and more efficiently than domain randomization. Future work includes increasingly the complexity and realism of the Sensor Transfer augmentation pipeline by modeling other, different sensor effects, as well as implementing models that better capture the pixel statistics of real images, such as motion or defocus blur. Other avenues include investigating the impact of task performance and problem space on the sensor effect parameter selection, and

evaluating how the proposed method impacts performance for training synthetic datasets rendered with various levels of photorealism.

CHAPTER III

Sensibility Transfer:

3.1 Introduction

Architecture can be defined as the organization of spatial components and voids through the assemblage of matter, which results in objects embedded not only in the three dimensional space of the physical environment, but also embedded within a symbolic and artistic culture defined by the designer [33]. The ability to generate novel and diverse 3D objects is essential to the architectural design process. The synthesis of a 3D object most often begins in early concept design (sketching on paper), to building, testing, and refining a 3D mesh model using standard 3D modeling softwares, such as Grasshopper, Maya, or Zbrush. However, this is not an easy task. 3D modeling softwares are very difficult to master because they require expertise with the software-specific operations that are required to generate the desired object manually. However, irrespective of the modeling software used, in order for the design process to be successful, the user must also have reasonable aesthetic and artistic sensibility. In basic terms, design sensibility can be defined as the ability to perceive and create the visual and perceptual appeal (beauty) of an object. There are a variety of different high level, abstracted concepts that together form the notion of sensibility: aesthetics, style, functionality, and the semantic nature of the object.

Each of these concepts describes different ways in which the geometric and material components of an object relate and interact with one another. Thus, each of these elements needs to be accounted for during the model generation process to produce a desirable outcome. As a result, honing this skill to a point where a user can successfully generalize and apply it to all of his/her generated objects can take years of training in formal art education, as well as years of practice. As a consequence, the development of intuitive and automatic 3D editing methods that allow users of all skill levels to explore novel design aesthetics while maintaining real-world physical properties is highly desirable.

Neural Networks are attractive candidates for high-level design editing frameworks because of their ability to extract and model stylistic, geometric, as well as attribute and semantic properties of the visual world. There has been incredible development in user controlled GAN renderers [13, 122, 155, 14], as well as methods for Style transfer and Image to Image translation [86, 61, 150, 68, 82, 107, 87, 106, 92, 149, 3], to name a few. Due to the power of their learned features and feature extraction capabilities, neural networks have also been applied to analyzing various works of art [156, 151, 11, 70]. However, an inherent limitation of these frameworks is that they are restricted to manipulations on an image grid in 2D; while these generative processes produce very visually stunning and potentially creative/aesthetic results, they can only get designers so far in the realm of architecture, whose design process is defined by and embodies the nature of 3D objects and structures. Thanks to recent advances in 3D deep learning the convolution kernel has been extended to non-euclidean spaces with graph convolutional neural networks (GCNNs). These breakthroughs have allowed for significant performance increases on 3D visual tasks, from classification to the reconstruction and generation of point clouds, voxels, and

meshes [146, 54, 145, 54, 51, 107, 10, 57].

Thus, the goal of the work presented in this chapter is to leverage the impressive advances in 3D deep neural networks to learn high level design concepts, specifically design sensibility, in a data driven manner.

This chapter presents a design technique that is capable of examining the sensibility and creation of architectural objects. Key to this technique is to train a GCNN to capture the sensibilities of a specific designer, namely Professor Matias del Campo of the University of Michigan's Taubman School of Architecture and Design, based upon 3D object's abstract features such as visual style, perceived aesthetic quality, and perceived functional quality. Through this work, we aim to give an answer to the question: How can a Neural Network interrogate and model the inherent sensibility of a specific designer?

The main contributions of this chapter are:

1. a novel dataset comprised of 3D models (meshes) that have been labeled according to design sensibility;

2. a novel optimization framework that leverages GraphCNNs to transfer sensibility features onto input meshes.

This chapter lays out a method that collects 3D models from the hand of one designer, creating a large database of models in two distinct categories: houses and columns – in order to train a Neural Network to come up with additional model solutions that are generated using the learned features of the trained network conditioned on the sensibilities captured within the training dataset. As far as we are aware, this is the first attempt to use classification graph convolutional neural networks to perform focused, artistic manipulations to meshes.

The chapter is organized as follows. Section 3.2 describes related works in mesh editing and deformation and 3D deep learning. Section 3.3.1 describes the sensibility dataset we created for this problem, as well as the proposed optimization framework for sensibility-driven mesh deformation. We present qualitative results and demonstrations of the optimization framework in Section 3.4 before concluding in Section 3.5.

3.2 Related Work

In this section we briefly review the related work that served as foundation for the proposed approach.

3.2.1 3D Data Representation

There are several standard ways of representing 3D data for training neural networks to perform 3D computer vision tasks. These representations are categorized into rasterized forms and geometric forms. Examples of rasterized, gridded forms include multi view RGB data, RGB-D data, and voxels. Examples of geometric, non-gridded forms include point clouds and meshes. The most popular rasterized form, voxels, is an attractive representation for 3D learning because convolutional neural networks can easily be extended to perform convolutions over 3D pixels. However, high quality local details and complex structures, both of which are key in defining a sense of shape, geometry, and thus style in 3D, are not easily achievable with the voxel representation due to the poor memory efficiency of the representation: memory and computational requirements of deep learning approaches based on the voxel grid representation scale cubically with the output size. Furthermore, as the resolution of voxel representations increases, there is an additional, and significant, computation cost. This often results in low resolution voxel models that appear

blocky and unrealistic. On the other hand, meshes are comprised of vertices and edges that define faces of polygonal shapes. This representation is memory efficient, compact, and allows for high fidelity representation of complex surface geometry. For example, realistic voxel models typically require hundreds of times the number of voxels than a polygonal model would need in vertexes, putting a significantly larger strain on the memory and CPU. This motivates our choice of using meshes as our input model as well as our choice of deep learning algorithm. Thus, for the remainder of this discussion, we focus on works that also perform on meshes.

3.2.2 Traditional Mesh Deformation

Significant work exists within the Computer Graphics field that focuses on developing algorithms that analyze solely the geometry or leverage shape learning to enact edits on 3D objects, specifically by deforming and fitting a user-specified source shape to a user-specified target shape. They rely on a sparse set of user-provided control/guide points (and typically mathematical priors). The transformations of these points are interpolated/propagated to all vertices in the mesh by casting deformation as an optimization problem to achieve interactive free-form deformations, while preserving local details and characteristics of the shape [129]. These methods are not data-driven, meaning that the only shape information they can access is contained in the two user input mesh objects or the control points. Some well-known examples are Laplacian editing [130] and as-rigid-as-possible manipulation [129] that regularize user-guided deformations to maintain local curvature based on mesh Laplacians and local rigidity. The biggest drawback of these approaches is that the non convex optimization problem that gives a solution for mesh deformation is highly intricate, and large deformations can yield implausible results with unnatural volume changes [167]. This necessitates expertise with the tools on the part of the user, which negates the

purpose of automated deformation. Additionally, these methods cannot capture or learn the shape variability from the given dataset, and thus cannot verify the semantic or perceptual plausibility of the deformed shapes; this is left up to the user to select appropriate guide points that will ensure realism of the output deformed object. This suggests that data driven approaches, such as the proposed method, could yield easier to use mesh editing/deformation techniques that can leverage learned shape distributions to ensure the realism of the output mesh.

3.2.3 2D to 3D Deep Learning of Mesh Deformation

As mentioned in 3.1, Convolutional neural networks have been quite successful as image generators. There has been significant development of design techniques that hack these 2D rendering and editing methods to be used in the design process of 3D models. Examples include 2D to 3D Neural Style Transfer, 2D silhouette-based vertex optimization, and 3D Deep Dreaming proposed in Neural 3D Mesh Renderer [**?**]. Other impressive differentiable renderers that propose image-based mesh deformation frameworks include Pytorch3D [111], SoftRas [88], and DIB-R [26]. There has also been impressive work in jointly training 2D and 3D networks for image-guided mesh deformation and reconstruction [146, 54, 145]. However, these techniques are inherently limited by the loss of rich information that occurs when compressing the 3D world into a 2D representation. Furthermore, it is difficult to control the object properties that are transferred in these processes, requiring significant post-processing on the part of the user to make the output realistic. The proposed mesh deformation technique builds upon work in 2D neural network feature transfer methods for images, but instead of using 2D trained networks to transfer learned features, we transfer features directly learned in 3D.

3.2.4 3D Deep Learning of Mesh Deformation

The recent success of deep learning has inspired alternative methods for data-driven mesh deformation, synthesis, and reconstruction. The focus of these methods has typically revolved around simultaneously solving two subproblems. First is part or substructure deformation or transfer, and second, is the preservation of fine geometric surface detail. These methods typically leverage a generative, Variational Autoencoder framework to learn a latent space that disentangles varying desired properties of meshes [50, 139, 51, 97]. Many of these also leverage graph convolutions [139, 51, 97] to construct these frameworks. The objective of these frameworks are to learn shape variation manifolds that can be used to sample latent representations that are used to generate 3D shapes. The drawbacks of some of these methods is that they require large datasets with semantic and part annotations to aid in the disentangling process, which are time consuming and difficult to collect [51, 97]. A separate body of works forgo purely generative models and instead use deep neural networks to learn continuous extensions of traditional deformation methods, most notably [105, 158]. The proposed method differs from the above deep 3D generative models because it does not depend upon learning latent, compressed representations of meshes. Instead, it takes an existing mesh as input and generates a variant by deforming it directly in 3D. As a result, the proposed method enables reusing existing 3D shape data with their associated meta-information, e.g., mesh color and texture, which can be carried along in the deformation. Note also that this work does not focus on part or substructure transfer, or transforming mesh pose, it focuses on modeling and transferring abstract artistic concepts and qualities that define an artists' sensibility.

3.3 Methods and Proposed Approach

The proposed approach can be broken down into several steps: we start first by generating and labeling a large databases of obj mesh models of specific architectural classes: houses and columns, described in Section 3.3.1. The next step is to train a graph convolutional neural network on this dataset to learn a mapping function between the cartesian coordinate space of 3D models to our defined label space of sensibility features, described in Section 3.3.2. The final step placing the trained GCNN into an optimization framework that takes in a user-specified 3d models as input, and then deforms the vertices of this model to optimize the shape for a user-specified visual aesthetic or functionality label, described in Section 3.3.3.

3.3.1 Sensibility Dataset Generation

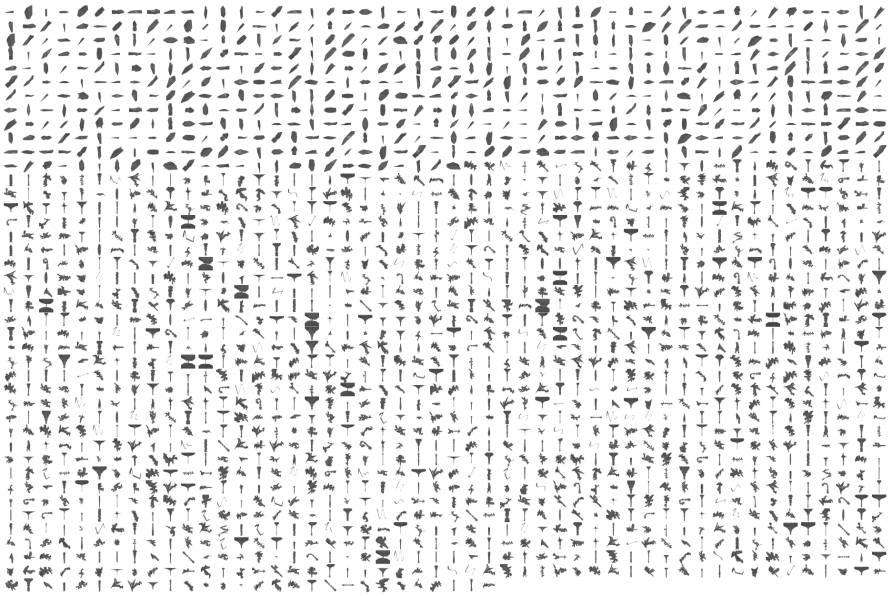

Figure 3.1: This is a rendering of all the models contained within the Sensibility dataset. It is a mix of abstract column and house models that capture different facets of design sensibility.

The database of models created for this work is visualized 3.1. They are all genus-0 meshes, and are comprised of only surfaces/exteriors, i.e., they are hollow volumes. The dataset has a total of 2178 models: 1552 columns and 626 houses. The resolution of the meshes ranges from the smallest mesh at 44 vertices to the largest mesh at 92140 vertices. The house models have an average of a 100 vertices, min of 44 max of 320, where as column models have an average of 3400 vertices, with min of 120 and max of 90000.

The models were generated using a series of the following techniques. The models started as low poly models generated with the standalone software TopMod - a small topological mesh modeling software. This allows the generation of lowpoly models that were designed to become either entire buildings (house) or parts of the architecture (columns). These first generation models were saved as OBJ files and imported into Autodesk Maya in order to create enough variation to constitute the database. Imported models were distorted in shape, randomly rotated, and mirror-cut in order to come to a number of around 1500 models. Using blend shaping in Maya (this procedure allows the user to select new locations for specific model vertices, and then the program generates intermediate meshes by interpolating between the old and new vertex locations), the designer, Matias del Campo, increased the number of varying models to the current total.

The dataset was manually labeled, by Matias del Campo, using the following procedure. The label space of the dataset decomposes the definition of a 'designer's sensibility' into the following properties: semantic, style, functionality, and aesthetics. The semantic property of a given data object defines the object's concept/meaning. Within our dataset, it can take on the value of house or column. The style property of an object refers to the substructures of the object that are distinctive and define

its appearance; they are determined by the theoretical/artistic principles that influenced the object's end design. In this Sensibility Dataset, a given object can take on one of three styles: Style A, which is inspired by structures/features typically associated with baroque; Style B, which is defined by substructures that exist within classic architecture; and Style C, which is defined by features associated with cubism. The functionality property is defined as the practicality of the object and its ability to serve a purpose well. Within our dataset, an object's functionality property is a score of 1-5, where 1 is 'not functional' and 5 is 'fully functional'. The aesthetics property is defined as the non-utilitarian pleasure the object evokes; subjective and sensori-emotional values, or sometimes called judgments of sentiment and taste. Within our dataset, an object's aesthetic property is a score of 1-5, where 1 is 'not pleasing' and 5 is 'very pleasing'. Note that both ugly, and nonfunctioning models are intentionally included in order to provide a complete example space that captures the total differences between functional, non-functional and aesthetically pleasing or not.

Note that the dataset was not designed with these labels in mind; it was modeled by Matias del Campo so as to capture his own unique and subjective design 'sensibility'. He (and he alone), then labeled this dataset based upon his subjective reactions to the models, specifically to how their low and high level structural patterns that manifested in style, aesthetics, etc. The dataset statistics are provided in Figure 3.4. Note that the dataset is biased towards specific a specific style. This is intentional and reflects the biases in the design sensibility of the author of the models in the dataset.

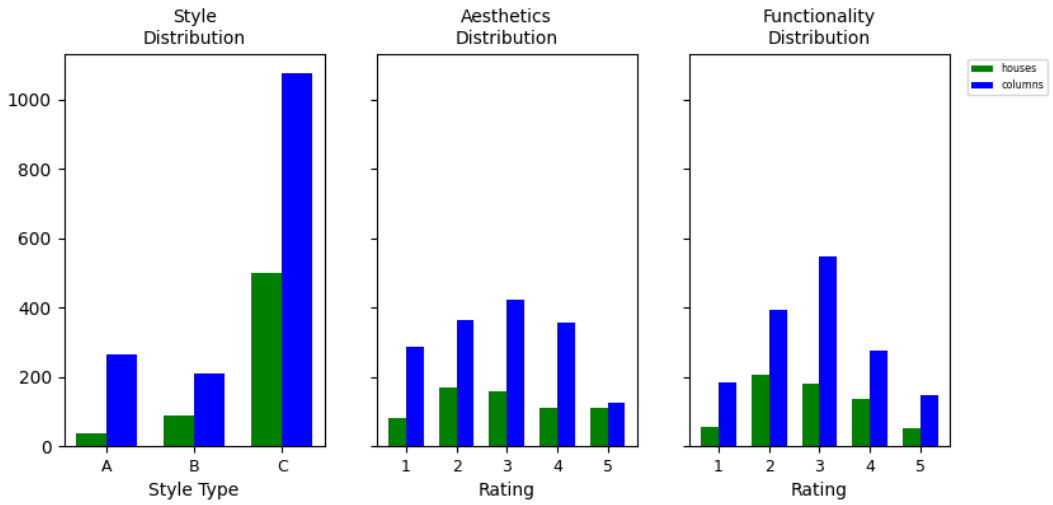

Figure 3.2: The distribution of labels for the Sensibility Dataset.

3.3.2 Sensibility Transfer GCNN Optimization Framework

GCNN Preliminaries and Notations

A 3D mesh is a collection of vertices, edges and faces; it can be defined as a graph $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ the where $\mathcal{V} = \{v_i\}_{i=1}^{N}$ is the set of N vertices in the mesh, $\mathcal{E} = \{e_i\}_{i=1}^{E}$ is the set of E edges with each connecting two vertices, and $\mathbf{F} = \{f_i\}_{i=1}^{N}$ are the feature vectors attached to the vertices. Graph Convolution Kipf2017SemiSupervisedCW for a layer l is defined as:

$$f_p^{l+1} = ReLU(w_0 f_p^l + \sum_{q \in \mathcal{N}(p)} w_1 f_q^l) \tag{3.1}$$

where $f_p^l \in \mathbb{R}^{d_l}, f_p^{l+1} \in \mathbb{R}^{d_{l+1}}$ are the feature vectors on vertex p before and after the convolution, and $\mathcal{N}(p)$ is the neighboring vertices of $p; w_0$ and w_1 are the learnable parameter matrices of $d_l \times d_{l+1}$ that are applied to all vertices. Note that w_1 is shared for all edges, and thus (3.1) works on nodes with different vertex degrees.

Implementation Details

The goal of this neural network is to learn a reasonable approximation of the function that describes a given architect's design sensibility, i.e., a function that

maps from 3D metric space to our semantic, style, aesthetic, and functionality label spaces. Due to the extreme differences in resolutions for columns and houses, we split the Sensibility Dataset into these two semantic subsets and trained two separate semantic networks, one on house models, and one on column models.

For each of these networks, we implemented a multi-task classification GCNN architecture, shown in Figure 3.3. It has four graph convolution layers, with feature dimensions of 128, 256, 256, 512. We then implemented a global average pooling layer that operates on the vertex dimension of the graph convolutional features. This output was fed into a shared fully connected layer, and this representation was input into three separate linear branches, one for functionality prediction, one for aesthetic prediction, and one for style prediction. Each branch is trained using standard cross entropy loss. The total loss is the sum over the loss from each task branch. Both networks were trained with a batch size of 32, a learning rate of 2e-4, with Adam optimization. They were trained until the loss converged. The network was implemented using the Pytorch3D library [111].

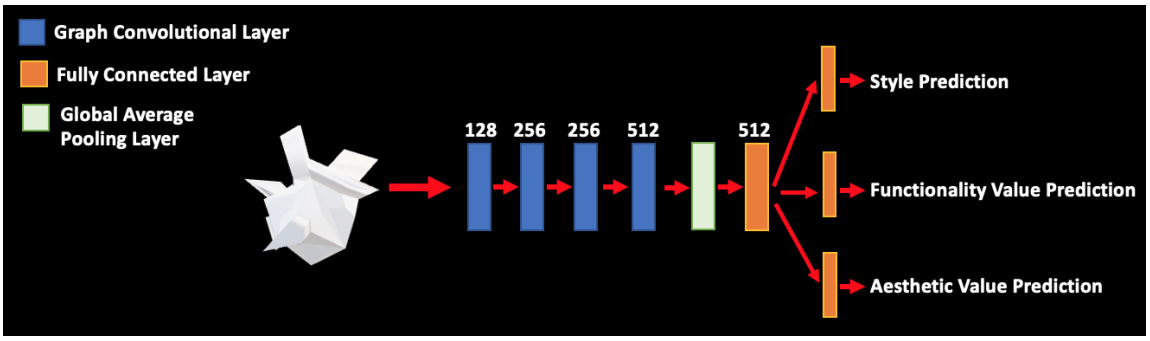

Figure 3.3: Sensibility prediction graphCNN architecture.

3.3.3 Sensibility Transfer Optimization Framework

The trained Sensibility prediction GCNNs will now act as our approximation of the dataset's 'designer', namely, Matias del Campo. They have learned the 3D mesh

features associated with our sensibility label space. We now can fix the parameters of these networks and invert them to transfer user input labels (and the corresponding sensibility features) to input meshes. We set the input mesh vertices as variables, and specify a set of desired style, aesthetic, and functionality labels. Using gradient-based optimization, we can iteratively change the locations of the vertices of the input mesh (i.e., deform the input mesh to produce an output mesh) such that the final deformed mesh would produce the specified, desired output labels. Note that the mesh vertices are now being learned, and the network parameters and labels are fixed. This process is similar to class-level deep dreaming [98]; we are directly manipulating the vertex locations of the input mesh in order to minimize the differences between the predicted sensibility labels and the user-input/target sensibility labels.

For each iteration of the optimization, we use one of the fixed GCNNs to project the current mesh into label space. We compare the mesh label predictions to the user-specified/desired mesh labels by calculating the cross entropy loss between them. This error is then backpropagated through the network into 3D space, where we now have an error value for each vertex, which represents how much each vertex contributed to the mesh prediction. These values are used to deform the vertex locations of the mesh in 3D space. We perform iterations until the loss value has converged.

3.4 Experiments

To evaluate the proposed method's ability to transfer Matias del Campo's 'design sensibility', we present the following three qualitative experiments. We first present extensive examples of the sensibility-optimized meshes using the proposed framework. We then present qualitative ablation experiments manipulating the dif-

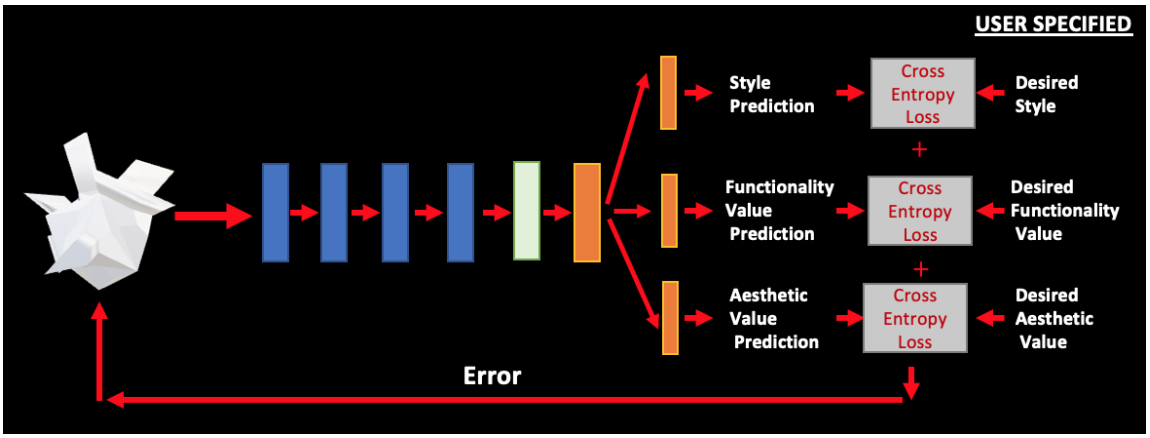

Figure 3.4: Shown above is a pictoral summary of the Sensibility Transfer Optimization Framework.

ferent sensibility axes of our label space to investigate the quality of the transfer. We then present an experiment where we use a network interpretability technique, visual saliency, to elucidate potential mesh sub structures that could uniquely contribute to aesthetics, functionality, or style. Note that our final two analyses focus on houses for brevity.

3.4.1 Mesh Sensibility Optimization Results

In Figure 3.5, we show examples of deformations produced by the proposed framework. On the left hand side are examples of various simple geometries deformed to be columns of various styles using the proposed framework. The first image panel on the left hand side is a icosahedron deformed to a column with Style B, the second/middle image panel on the left hand side is a cube deformed into a column with Style C, and the bottom image panel on the left hand side is a cylinder deformed to a column with StyleB . On the right hand side are examples of various input shapes deformed into houses, with the the top image panel showing a dodecahdron deformed to a Style C house, the middle image panel showing an octahedron deformed to a Style A house, and the bottom panel showing an icosahedron deformed to a style C house. All deformed models used a high aesthetic and medium functionality value.

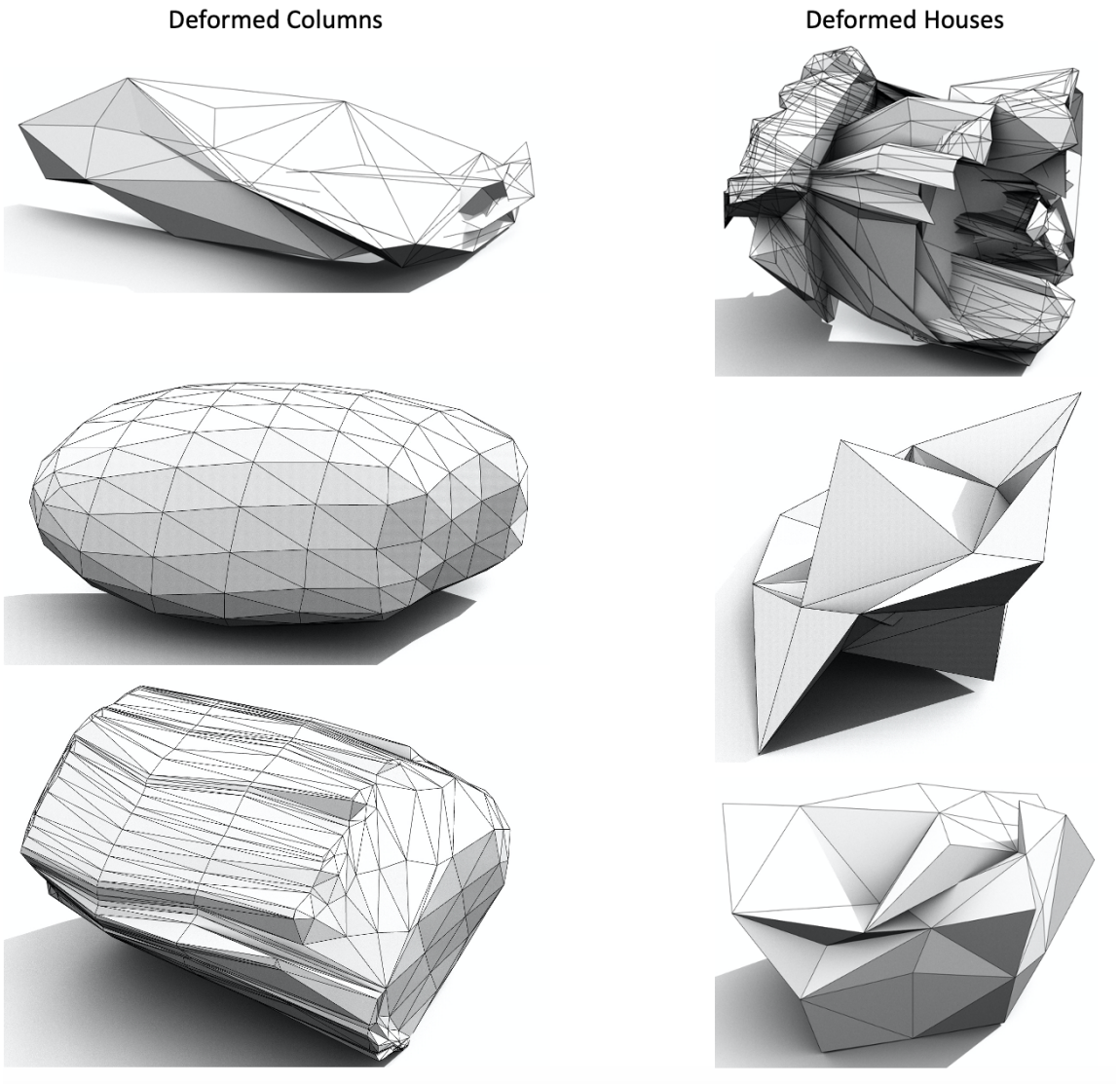

Figure 3.5: Examples of deformations produced by the proposed framework. In the left hand column are examples of various shapes deformed into columns, and in the right hand column are examples of deforming various shapes into houses.

For the columns, we see that the network has been able to learn to elongate and thin out the input shapes to capture the global structure that is typically associated with columns. Similarly, with the houses, we see that they retain volumes similar to their original shapes, and have surfaces that could be considered 'floor-like'. In terms of style, we observe that each shape does have a distinct difference in appearance. Recall that the classification of Style A, B and C in terms of its Style relied

on a set of specific rules derived from Baroque, Classic and Cubist architecture. For example, Baroque can be classified through features such as symmetry, curvature and concave/convex spiel. Classic can be read here as Classicist or Modern, where Modern relies on the formal and Stylistic qualities of high Modern architecture from the 1920's to the 1950's (proportionality, orthogonal, asymmetrical). Cubist as pertaining to the triangulation of polygonal bodies, akin to features found in Czech Cubist architecture. This of course is blatant simplification of each of these styles, and rather an innate response from the creator of the database - producing a less scientific but rather spontaneous response to visual stimuli. We see that the generated shapes do captured these features for each of the desired styles.

3.4.2 Ablation Experiments

This ablation experiment investigated the manipulation of each of the different label spaces while holding the others constant. This was done in an effort to examine the quality of the internal models learned by the GCNN for each of these feature axes. For this experiment, we focused on the generation of different houses from an octahedron mesh. For each of the different labels, style, aesthetics, and functionality, we fixed all other parameters and varied one. The outputs are presented in Fig. 6. For varying style, which is in the top row of the figure, we set aesthetics and functionality to have ratings of 4 and held those values constant while altering the style. We see that the final results have distinctly different forms. For varying functionality, shown in the middle row of the figure, we set the style to be Style C, and fixed aesthetics to have a rating of 4 while varying the functionality rating. For varying aesthetics, which is shown in the bottom row, we set the style to be Style C, and fixed the functionality to be 3 while varying the aesthetic rating. As in the previous experiment, we observe that semantic and stylistic shapes are the

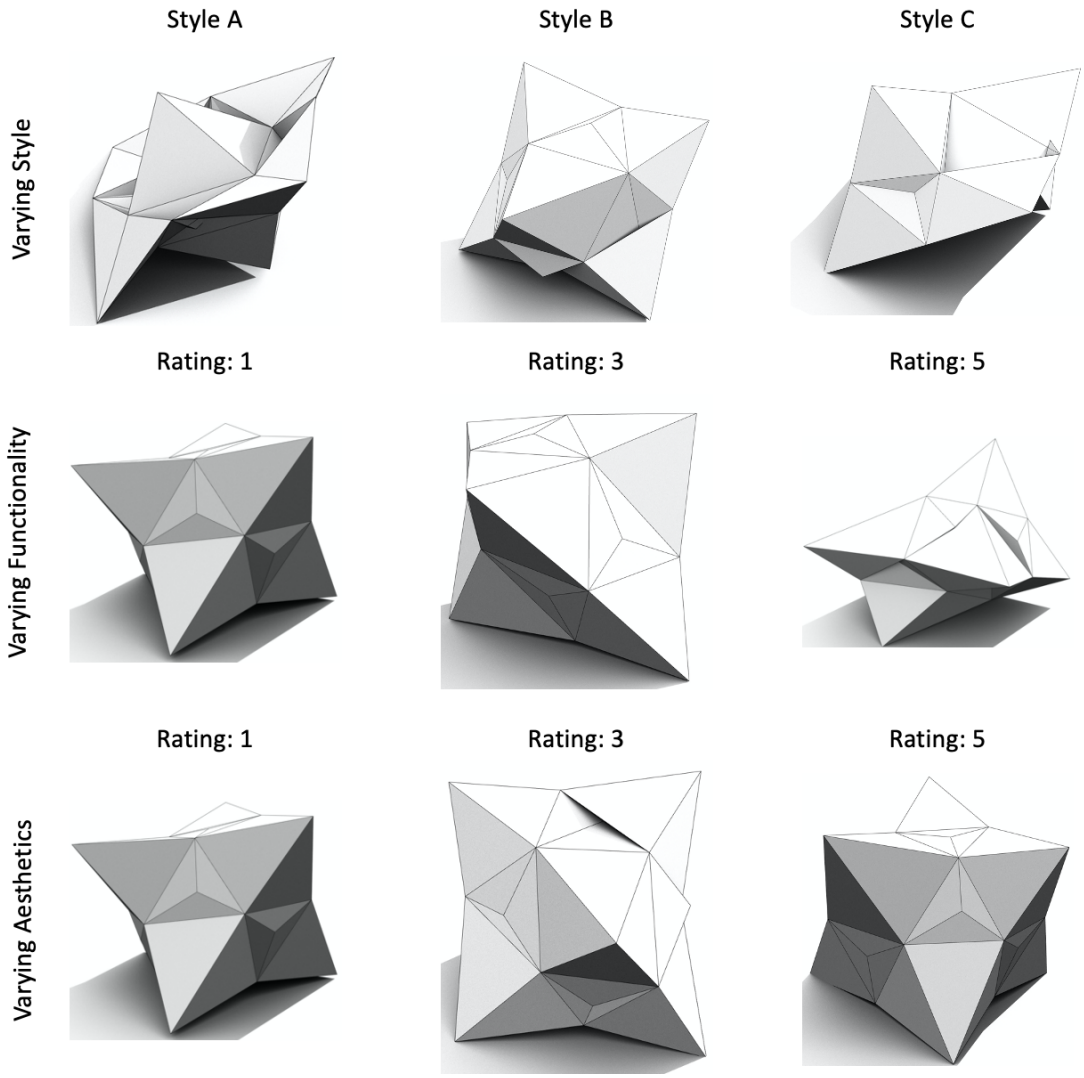

Figure 3.6: The outputs. Style, functionality and aesthetics do not necessarily follow objective criteria, but the criteria put into place by Matias del Campo. For example the result of varying functionality of a rating of 5 produces a volume that can be converted into a low slung house on a hillside.

dominant features transferred with reasonable fidelity to how they manifest in the Sensibility Dataset. The fidelity of the aesthetics and functionality transfer is much more nuanced due to their significantly more subjective nature, particularly with regards to the qualities that make a different styles of houses more or less functional or aesthetic to the designer and labeler of the dataset. The gauge of functionality has to do with the resulting model proportions; if the resulting figure has the wrong

proportions in terms of designing a house it is "less functional" (the figure is too low, too high, too narrow in order to accommodate the program of a house). We can see this manifest in the models as the functionality increases; the models go from 'too high' walls to spread out into flattened roof and ceiling features as the functionality score is increased. Aesthetics is far more difficult to capture objectively because it really relies on the labeler's unique and subjective sensibility. For the shape generated with high aesthetics (a rating of 5), it has well proportioned length to width to height. This means that the particular model can be scaled proportionally and could fulfill various programs, from House to Concert hall. The relationship between concave and convex parts is also nicely balanced, giving an allover even figure. The model silhouette is exciting, without being overly aggressive, despite lots of pointy elements.

3.4.3 Visual Saliency of Sensibility

From the previous two results, we have demonstrated that this model has effectively learned 'architectural taste/aesthetics' as defined by the architect and labeler of the dataset. However, what is unclear is if the neural network will learn to associate the same visual features with 'pleasing' as an architect does. To further our investigation into GCNNs capabilities of Sensibility transfer, we performed the following network interpretability experiment to shed light upon what features of the input meshes the neural network learns to associate with the different abstract concepts of style, aesthetics, and functionality. There has been incredible advances in interpretability and explainability of learned features for conventional convolutional neural networks [102, 101]. These advancements have led to design/editing techniques such as Deep Dream [98]. Inspired by this body of work, we extend a particular interpretability technique, Visual Saliency [125] to GCNNs. This will allow us to qualitatively/visually examine the structural features that our Sensibility network

associates with different style, aesthetic, and functionality classes. Visual saliency is the distinct, task-dependent perceptual quality which makes some items in the world 'more useful' than their neighbors. In an image, a saliency map shows each pixel's unique feature in the context of visual information and visual tasks. For example, in image classification, a pixel that is part of a specific semantic class, e.g., dog, could have a high saliency/intensity value. Similarly, in the context of graphs/meshes, saliency at a vertex can be defined in terms of how much that particular vertex contributes to specific visual or perceptual features.

We can use gradient-based visual saliency mapping to assign an intensity value (or color) to each vertex in our input mesh. This intensity will correspond to how much that given vertex contributes to a particular class prediction. To calculate the visual saliency for a given input mesh, we first perform a forward pass of the input mesh using the trained Sensibility prediction network. Then, choosing one of the task predictions, we calculate the backward pass to get the gradient of that prediction with respect to the input mesh vertices. The normalized gradients serve as our vertex colors.

More formally, consider our neural network as a function, f, with three task branches, f_{style}, f_{aesth}, and f_{func} for the tasks of style classification, aesthetic rating classification, and functionality rating classification, respectively. For each task branch, we can solve the following to get the saliency $S*$, for all vertices v in our input mesh M:

$$S*_{style} = \max_{M}(f_{style}(M))$$

$$S*_{aesth} = \max_{M}(f_{aesth}(M))$$

$$S*_{func} = \max_{M} x(f_{func}(M))$$

Using this technique, we can shed light upon what features of the input meshes the neural network learns to associate with the different abstract concepts of style, aesthetics, and functionality. Examples for three house models are shown in Figure 3.7

We see complex, shape-dependent relationships between the features for style, aesthetics, and functionality. For example, in the top row, it appears that the Sensibility prediction network is using different and unique vertex features for each task. However, in the middle row it appears that it leverages very similarly vertex clusters for predicting style, functionality, and aesthetics. Finally, in the last layer, we see that the same vertices have positive impact upon the style and aesthetics predictions, but not the functionality. This highlights the difficulty of trying to quantify and encode highly intertwined design concepts. Further work will be needed to see if it is possible to disentangle these abstract properties from one another.

3.5 Conclusion

There are two main paths of inquiry to be considered when assessing an algorithm's ability for design processes: first, the interrogation of the technical expertise necessary to train neural networks to generate successful solutions for pragmatic problems. This can be plan optimization, structural optimization and the analysis of the consumption of material. The second path to be explored is the aspects of

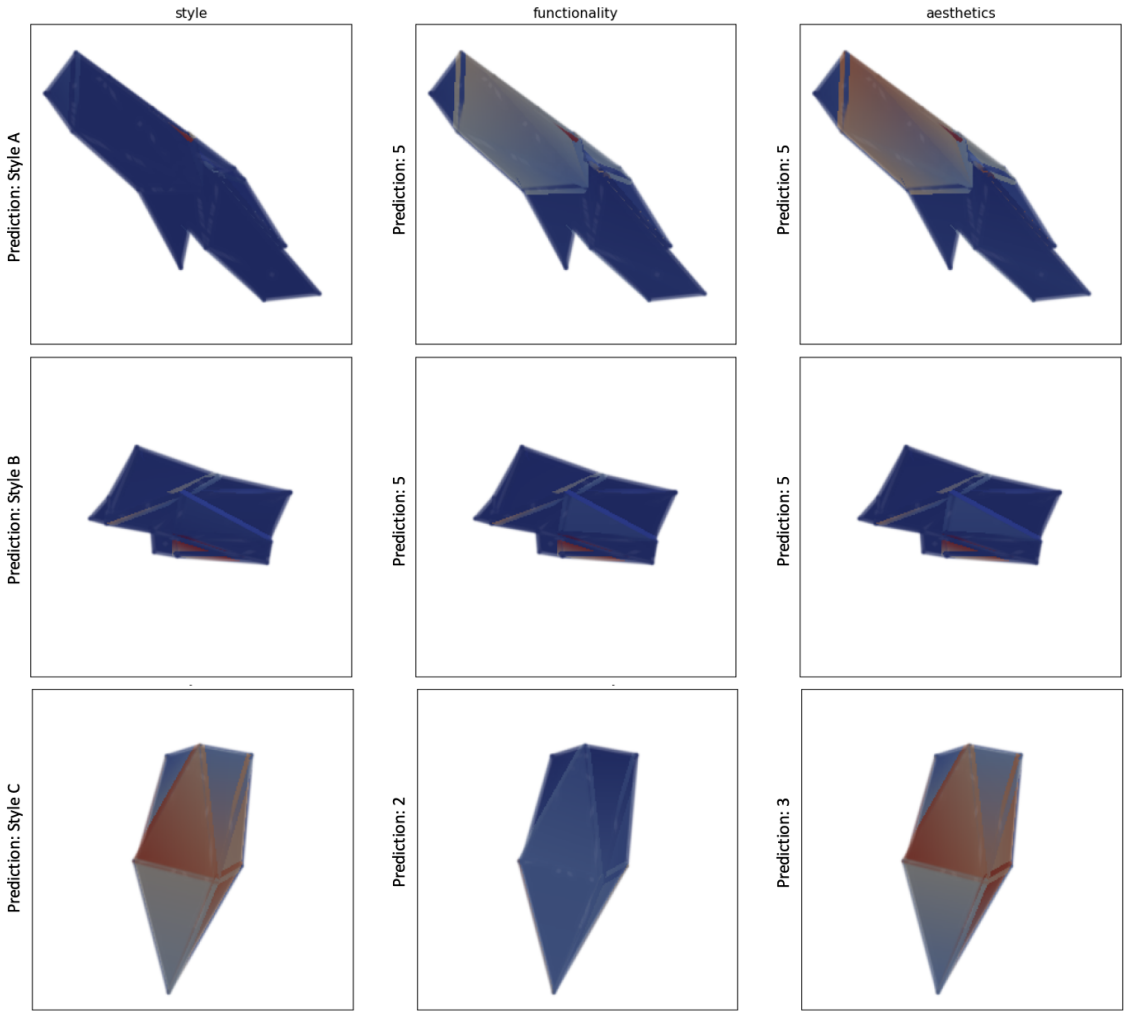

Figure 3.7: Examples of visual saliency for the three sensibility prediction tasks for three house models. The high positive values are near-red, the high negative values are near-blue, and the vertices that have a derivative near zero (thus not making much of an impact on the output class prediction when the vertex location is changed) are gray/near-white.

architectural design pertaining to studies of morphology, style, mood and creativity. In this thesis chapter, our goal was to test the capabilities of neural networks to model and transfer the highly abstract and complex concept of a designer's sensibility. Leveraging the powerful ability of neural networks to ingest and learn from large databases of images that can span cultural and historical dimensions that a human or humans would take a lifetime to synthesize and learn, we present a solution that allows for the transfer of aesthetic, style, and functionality features to meshes to

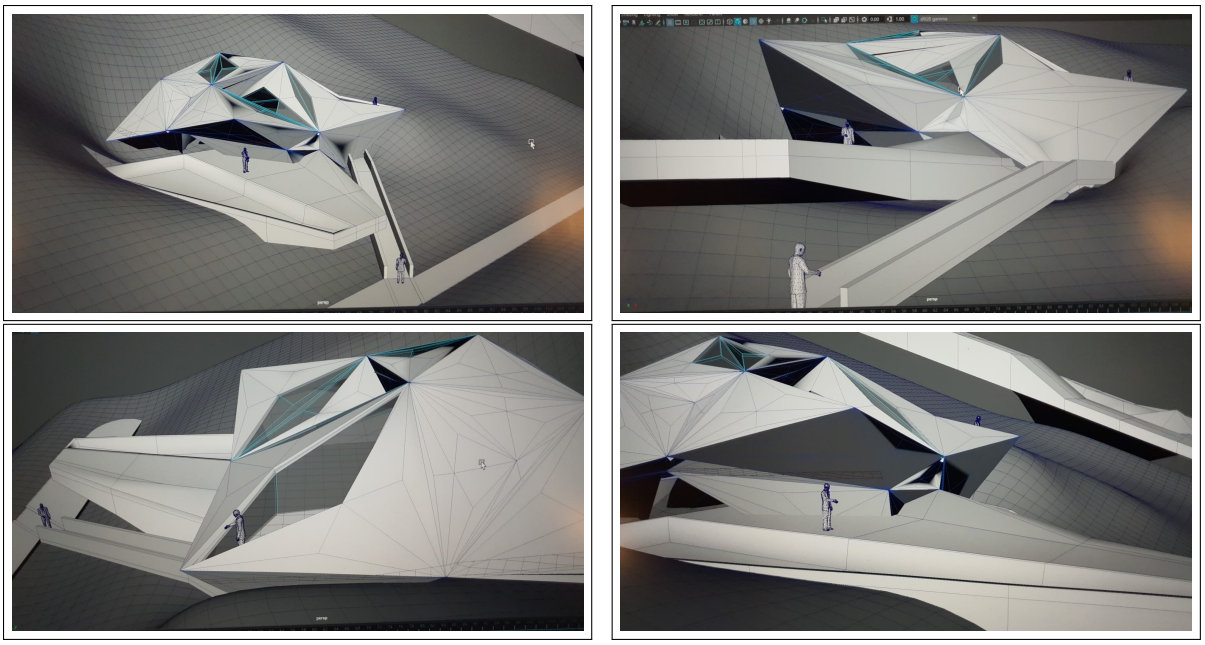

Figure 3.8: Examples of one of the manipulated, cubist houses as a full building, from different viewing angles.

generate new objects that capture the design sensibility of our co-author, Matias del Campo. We show a finalized rendering of one of the Sensibility transferred cubist houses in Figure 3.8. Future work would be to collect larger, more comprehensive datasets (possibly from different designers/architects), which would allow for a more thorough investigation into how neural networks represent these abstract art concepts. Another avenue of future work is to explore the feature spaces of generative networks on these kinds of datasets.

CHAPTER IV

Illumination Transfer

4.1 Introduction

In this chapter, we propose an image synthesis framework that leverages geometry and reflectance information from a Lidar scanner to control the illumination in an output image, which can be applied as a solution to image relighting.

In outdoor road environments, the appearance of a salient object, such as a car or pedestrian, is highly dependent upon the illumination of the scene. Illumination effects have a particularly strong impact on a scene's representation in camera's RGB space, which is the most common sensor space for driving datasets. Adjusting the time of day for this sensor space can induce significant changes to the scene's appearance despite the fact that its underlying structure and material properties remain the same. In fact, illumination changes have been shown to significantly decrease performance of state of the art segmentation, detection, and tracking algorithms in outdoor driving scenarios [4, 12, 110]. Thus, an important task for a car's visual system is to accurately capture a scene's illumination model from the given sensor data. This requires separating the visual changes in the image pixel intensity that are due to variance in the underlying surface material properties from the changes that are exclusively due to the effects of the scene illumination. Illumination model-

ing techniques can be data-driven methods [160, 84, 116, 22, 28], or purely based on physics and vision principles, or some combination of the two [79, 100, 64, 89, 83, 77]. The majority of these illumination modeling methods focus on learning or modeling illumination exclusively using RGB images. There has been impressive advances in more data-driven illumination modeling and image relighting using deep learning [43, 109, 157, 116, 153]. These methods rely on having multiple views of the scene under varying illumination conditions, which are difficult to obtain for complex, outdoor driving datasets.

On the other hand, physics-based techniques utilize rigid, hand crafted priors that prevent them from generalizing to large, complex noisy scenes composed of many material types. However, one such physics-based method, intrinsic images, is particularly attractive for outdoor driving scenes. Intrinsic images are the decomposition of an RGB image into its intrinsic properties (e.g., reflectance, albedo, normals), and these intermediate representations can be used to manipulate illumination in RGB space in a deterministic manner [142, 25, 120, 121, 74, 166]. In fact, several research efforts have explored a variation of intrinsic images called illumination invariant color spaces [91, 159, 71, 46] that derive a scene representation directly from a single RGB image that is solely based upon underlying surface reflectance. Unfortunately, the derivation of illumination invariant color spaces are based only upon RGB cameras, and as a result can still contain illumination artifacts.

In this chapter, we instead consider how we might model illumination using an alternative sensor space: Lidar. Laser scanner return intensity, which accompanies 3D point-cloud coordinates, provides information about the power of the backscattered laser signal and it is dependent upon the surface properties [69]. This information is useful to determine the reflectance properties of encountered surfaces and is has

been utilized in a variety of applications, particularly in urban material classification [161, 154, 136, 37, 56].

The method proposed in this chapter uses the measured Lidar intensity as an estimate of the scene surface reflectance, which when combined with the point cloud geometry information and RGB chromaticity, can be used to light the captured scene. In an effort to overcome the sparsity of the Lidar signal, we also present two methods to generate a dense reflectance image. The first uses standard projection and 2d interpolation of Lidar into image space, and the second uses a deep volumetric learning method based upon Neural Radiance Fields (abbreviated NeRF) [96] to generate a dense Lidar reflectance and geometric model of the scene.

To our knowledge, this is the first attempt at specifically lighting outdoor driving datasets by transferring lighting features onto Lidar intensity. We anticipate that the proposed model, by adding in realistic illumination artifacts into images, is a first step towards understanding how RGB-Lidar sensor fusion can be used to represent and learn accurate outdoor scene illumination models.

The chapter is organized as follows. Section 4.2 describes related works in deterministic and learning-based illumination modeling frameworks. Section 4.3 describes the proposed deterministic lighting framework that fuses Lidar and RGB information, as well as the two methods to generate dense reflectance maps. We present qualitative results and demonstrations of the proposed frameworks in Section 4.4 before concluding in Section 4.5.

4.2 Related Work

In this work, the objective is to transfer realistic lighting effects onto an image given a point cloud and a single RGB image. This process requires both removing

and re-synthesizing illumination features within the image. The proposed method is based upon previous work in illumination invariant color spaces, intrinsic image decomposition, image relighting, shadow modeling, and neural rendering. We briefly review the literature in these areas below.

4.2.1 Illumination Invariant Color spaces

Most similar in objective and method to the proposed framework are Illumination Invariant color spaces. These comprise a family of color representations, computed from RGB, that removes (or minimizes) scene color variations due to varying scene lighting conditions based upon the sensor model and physical properties of the scene [91, 119, 71, 46, 6, 143]. The methods all use varying sensor and physics models as priors, specifically lambertian reflectance, approximately Planckian lighting, and fairly narrowband camera sensors, to derive an intrinsic reflectivity image from a single RGB image. The output of these methods is a grayscale image that can be interpreted as an intrinsic image that portrays only the inherent reflectance properties in the scene without harsh illumination effects, specifically cast shadows, self-shading, and over/under exposure. These color spaces have been applied to a variety of vision tasks in an attempt to improve accuracy and robustness to illumination. For example, there has been great success in applying illumination invariant color spaces to improving visual odometry accuracy [91, 94]. There has also been some work to indicate that combing illumination invariant representation of an image with its chromaticity channels is an effective preprocessing step for semantic segmentation [4, 5], but that the performance benefits appear to be dependent on both the segmentation network architecture and training dataset, which in general is not useful. This chapter is the first work to explore if these representations can be used to relight scenes.

4.2.2 Inverse graphics and Intrinsic Images for Reflectance and Shading Estimation

Intrinsic image decomposition and inverse graphics rendering are methods that decompose an image into a set of intermediate representations or features that correspond to different physical process in the real world, e.g, normal maps, albedo, shading, and reflectance [142, 25, 120, 121, 74, 166]. These intermediate representations can be sampled to synthesize new unseen images, which allows for image relighting. However, there are drawbacks to these techniques that prevent them from being applied to large scale outdoor scenes. These methods make simplifying assumptions about the scene structure to make the reconstruction tractable, and thus are usually applied to scenes that contain similar spatial information, such as faces [24, 148, 144]. These methods also either require multiple images of the same scene [109] or require complicated training regimes corresponding to complex synthetic datasets [40, 73], both of which are difficult to generalize to driving datasets.

4.2.3 Lidar Intensity for estimating intrinsic scene properties

Material characteristics, specifically surface reflectance and roughness have a have a large impact on the measured intensity values [69]. A number of studies in the archaeological and aerial image fields have studied Lidar intensity as a strong source of information for tasks such as material detection and classification [161, 154, 136, 37, 56]. In fact, this work has been extended to model a surface's BRDF using Lidar intensity [80, 23, 81, 58]. Despite this work, there are very few instances in which Lidar intensity has been used in relighting. While there is work in colorizing point clouds via intensity [85, 19, 123] as well as work in point cloud relighting [117, 152, 55], these methods mainly use the geometry given by the point cloud, and forgo the Lidar intensity as a source of reflectance information. In contrast, in this chapter we explore

the use of Lidar intensity as a direct measure of surface reflectance that can be used in illumination modeling and relighting frameworks.

4.2.4 Learning reflectance models from images data

The goal of image relighting is to change the existing illumination condition captured in an image or set of images to a target illumination condition. At their core, image relighting techniques attempt to model the light transport function, which is composed of a model of scene lighting and the BRDF (bidirectional reflectance density function) that describes the material properties throughout the scene and how they interact with scene illumination [132, 153]. However, to model this function the majority of deep relighting methods require multiple images (anywhere between 5-1000 depending on the image relighting method) of the same scene under different lighting conditions to generate an accurate estimate of the lighting function [153, 109], and/or material labels of objects in the scene [2], neither of which is easy to obtain for outdoor driving datasets. Recently, in the past year, there has been a deep learning/graphics revolution thanks to a family of models called Neural radiance Fields, abbreviated as NeRF [96]. These models pair small multilayer perceptrons with traditional ray tracing techniques to learn a volumetric, emissive model of the scene. They have been extended to learn reflectance models of scenes [131, 35, 16, 17, 163]. The majority of these reflectance NeRF models rely on dense, 360 degree capture multiview datasets of a single scene under varying illumination conditions, which are not obtainable for driving datasets. However, recent work [133, 103, 78] has demonstrated the applicability of NeRF to driving data. In this chapter, we present one of the first attempts to fuse an additional sensor, Lidar, into a NeRF framework with the goal of using dense reflectances maps for illumination modeling in driving datasets.

4.3 Proposed Approach

The objective of our approach is to alter the illumination conditions of a given input image and its accompanying point cloud such that it appears to have been taken at a specific time of day. This approach is inspired by Illumination invariant color spaces. We propose to light images by modeling how illumination effects manifest in the three channels of CEILAB color space: the L (luminance) channel, the a chrominance channel (red-green opponent color axis) and the b chrominance channel (blue-yellow opponent color axis). We use the Lidar intensity as a noisy estimate of scene surface reflectance that can be used to estimate the luminance channel. We describe this process in the sections below.

For autonomous vehicle driving datasets, we define the scene lighting conditions as the light source location in the scene, which for outdoor images would be the sun position. We choose this parameterization because it is easy to extract from GPS/timestamp information, and also because altering the light source location in an image can simultaneously changes the perceived illumination effects in the scene, specifically shadow distributions, color temperature and brightness. Thus, the proposed approach takes in a single input image from a driving trajectory, its corresponding Lidar scan and a sun position vector derived from the GPS/timestamp information.

Modeling the Luminance channel using Lidar Intensity

The luminance L of a point lying on an outdoor surface can be defined as [55]:

$$L = V * R * E_{Sun} * (N \cdot D)$$

where R is the surface reflectance, V is the visibility of the point to the light source, E_{Sun} is the luminance of the sun that hits the point, N is the surface nor-

mal at the point, and D is the position vector of the light source. For simplicity and computational tractability, we forgo modeling indirect illumination and skylight illumination that would normally be present in an outdoor scene, and only consider the sun as a directional light source. To produce the luminance channel map, we generate a reflectance map R by aggregating Lidar scans about the chosen RGB image and project them into the camera view. The visibility V is generated using standard ray tracing paired with the the sun position, the normals and depth at each point in the velo scan, and then projected into the 2D image space of the camera. It can take on a binary value of 0 or 1 indicating if the point is in light or shadow. We also threshold the L and b channels to capture any shadows that are cast by geometry not captured in the Lidar scan. These two maps are added together to create the final visibility/shadow map V. We upsample each of these outputs using nearest neighbor interpolation on the images to get dense maps.

Modeling Chromasticity

While Lidar intensity can provide information about scene surface properties like roughness and reflectance, it cannot give us an estimate of the base color of objects within the scene. Thus, we directly use the a and b CEILAB channels extracted from the selected RGB image as a source of chromaticity information. This ultimately allows us to take advantage of the joint nature of the image structure and Lidar data reflectance properties. However, the CEILAB chromaticity channels still have illumination artifacts in them that are unique to the source sun position, specifically, shadows. Areas in shadow tend to be more blue (due to lack of yellow sunlight) and areas in light tend to be more yellow [75]. We correct these artifacts using as simple chrominance ratio between the sunlit points and shadowed points in the chrominance channels. This method was first proposed in [55]. For the b channel, we

Figure 4.1: Example of removing a hard cast shadow in the chrominance channels. The first column contains an example of an uncorrected image patch (where the dark pixels indicate the presence of a hard cast shadow), and the second column contains the corrected image patch. Note that there are still artifacts that remain in the shadow penumbra, i.e, the shadow edges.

observe that shadows are more blue (lower value), and sunlight regions more yellow (higher value). In contrast, we observe for the a channel that shadows are more red (higher value), sunlight regions more green (lower value). We therefore find that the following simple multiplicative updates for adjusting the chrominance values of shadow pixels suffices:

$$a(p_{Shadow}) = a(p_{Shadow}) * \frac{a(\bar{p_{Sunlit}})}{a(\bar{p_{Shadow}})}$$
$$b(p_{Shadow}) = b(p_{Shadow}) * \frac{b(\bar{p_{Sunlit}})}{b(\bar{p_{Shadow}})}$$

where p_{Shadow} are the pixels in shadow and p_{Sunlit} are the pixels lit by the sun (derived from the shadow map), and $a(\bar{p_{Sunlit}})$ is the average chrominance value of the sunlit pixels in the a channel, where as $a(\bar{p_{Shadow}})$ is the average or the shadow pixels. The same convention is used for the b channel. An example is shown in Figure 4.1.

4.3.1 Extending NeRF for densifying Lidar reflectance maps

Densifying our lidar reflectance maps using 2D nearest neighbor interpolation can only get us so far; we cannot use this method to hallucinate the parts of the image where we have no Lidar measurements. In particular, for driving datasets, this means we cannot model the upper areas of the scene, including the sky. We propose

to leap frog off the incredible success of NeRF models to densify our sparse Lidar reflectance maps. We specifically choose the MINE NeRF model [78] as a starting point. MINE was designed to perform novel view synthesis and depth estimation from a single image. The network design fuses Multiplane Images (MPI) [140] with NeRF to learn a continuous 3D model of a scene captured by a single camera image. The objective of MINE is to model the 3D scene within the camera viewing frustrum. Given a single image and target disparity as input, MINE uses an encoder-decoder network to predict a 4-channel plane/image (RGB and volume density, or sigma) at the injected depth within the frustrum. This is done at many sampled depth values to reconstruct the 3D space within the camera frustum. The reconstructed frustum can then be easily rendered into novel views using differentiable rendering. See Figure 4.2 for a pictorial overview of the network. To train MINE on a driving dataset, each scene is defined as a stereo image pair. Thus, one 'source' stereo view is injected into the MINE framework and its goal is to reconstruct the other, held-out stereo image.

The MINE model is attractive for our problem because it allows us to learn across scenes, and thus across sparse lidar signals. This is in contrast to original NeRF models, which can only be trained on single scenes, which we found in practice struggles to learn a dense model from a sparse reflectance signal. To generate dense Lidar reflectance maps, we modify the MINE decoding network to output a Lidar reflectance value in addition to an RGB and sigma value. The predicted Lidar intensity is alpha-composited to generate a final, dense reflectance map. To train the network, we use the extrinsic and intrinsic camera models to project the sparse lidar measurements into the camera view frustrum, and apply a mean absolute error loss between these ground truth points and the predicted lidar intensity values.

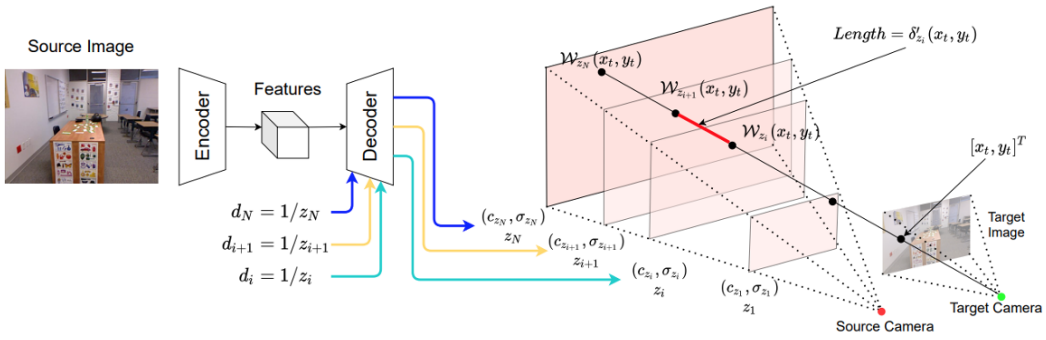

Figure 4.2: Pictorial overview of the MINE NeRF framework. Figure is taken from the original paper [78]. To generate dense Lidar reflectance maps, we modify the decoding network to output Lidar reflectance value in addition to an RGB and sigma value. The predicted Lidar intensity is alpha-composited to generate a final, dense reflectance map.

4.4 Results and Experiments

All experiments are performed with the KITTI driving dataset [48]. To generate a lit RGB image, we take an RGB frame at some chosen time t. To produce an accompanying point cloud, we aggregate the velo scans at time $t-1$, t, and $t+1$.

4.4.1 Comparison to Illumination Invariant color spaces for relighting

We first present results that compare the predicted Lidar-based luminance channel from the proposed method against the estimated reflectance/intensity maps generated from other methods. We specifically evaluate against those generated by illumination invariant color spaces. We use each of these estimated reflectance channels in the proposed relighting framework, and evaluate the visual quality of the produced RGB image. Because we only have access to a single RGB image taken at a single light source, in order to quantitatively evaluate the proposed relighting framework we use the proposed relighting method by reconstructing KITTI images using the ground truth/source sun position. Qualitative results of this on single images are

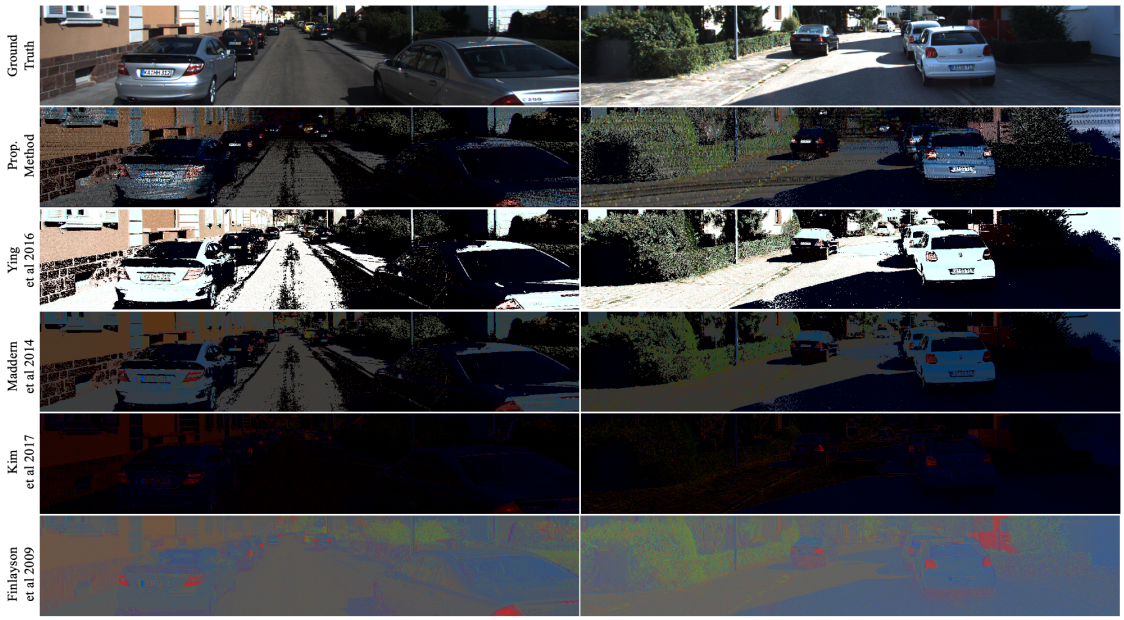

Figure 4.3: Examples of KITTI images lit to their ground truth lighting

provided in Figure 4.3. We see that the proposed method produces lit images that significantly more consistent, specifically in terms of color temperature, brightness, and specular effects. To quantitatively verify that the proposed lighting method is more accurate across images, we randomly sampled 1000 images from the KITTI raw dataset and calculated the average PSNR and SSIM for the proposed method vs. the illumination invariant image baselines. The results are given in Table 4.1. Note that these metrics are calculated only for pixels where we have Lidar reflectance data. We see that the proposed method produces lit RGB values that have significantly higher quality than the baselines.

An interesting behavior that was uncovered during this experiment is that the number of Lidar scans that are aggregated over for generating a reflectance image for a particular RGB is a tunable hyperparameter. Aggregating over too many velo scans corrupts the local geometry due to Lidar sensor noise and the dynamic nature of the data collection. This noise will overwhelm the metrics relying on absolute

Table 4.1: Average PSNR and SSIM for 1000 lit images sampled randomly from the KITTI drive.

Reflectance Est. Method	Reconstructed KITTI RGB	
	PSNR↑	SSIM↑
Maddern et al 2014 [91]	17.98	0.69
Ying et al 2016 [159]	17.94	0.73
Kim et al 2017 [71]	14.39	0.50
Finlayson et al 2009 [46]	19.72	0.61
Proposed Method	**21.40**	**0.75**
Proposed Method + dense intensity	9.2	0.28

error and small local areas of the output image. This observation further motivates the need to generate smooth lidar reflectance maps.

However, the proposed method has several failure modes. In the first panel in Figure 4.4, we observe that one of the failure modes of the proposed method is that it cannot accurately model illumination effects that result from the imaging sensor, specifically saturation/overexposure. This is reasonable considering that the proposed image formation pipeline does not include a model of the camera sensor. Future work could investigate modeling a camera sensor to capture these effects as well. In the second column of Figure 4.4, we see another failure mode: Lidar beam artifacts. The Lidar sensor sweeps are replicated in the predicted RGB image, which is highly undesirable. We also see that a lot of high frequency detail in the bush/leaves are lost also as a result of the Lidar sampling. Furthermore, we cannot light or reconstruct the parts of the scene that have no Lidar returns, which is also highlighted in the first column of Figure 4.4 (red square bottom image). This means that we cannot model the upper parts of KITTI, including the sky, which carries a significant impact upon the perceived illumination and time of day in the scene. This motivates our exploration in the next section into modeling a dense Lidar reflectance signal, which will allow us to apply the proposed method over the entire scene.

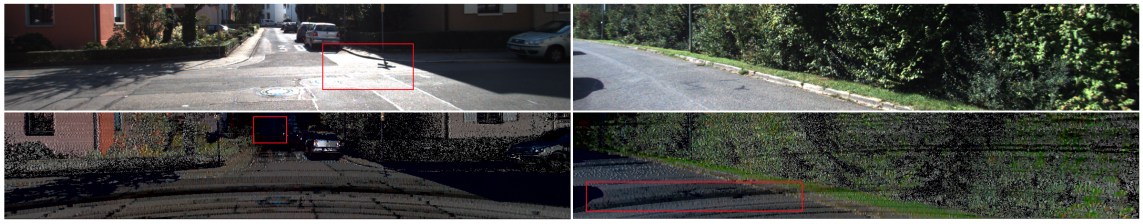

Figure 4.4: Examples of Failure modes for the proposed method. Top row are ground truth RGB images, bottom row are the corresponding lit images using the proposed method. Red squares are used to highlight specific failure modes of the proposed method. In the first column, the proposed method is unable to replicate the overexposed road in the target RGB image (highlighted by the red square in the top image; this saturation effect is absent in the predicted image below). We also see that there are artifacts introduced by the scanning geometry of the Lidar sensor. For example, we cannot light/reconstruct the parts of the scene that have no Lidar data, which is also highlighted in the first column (red square bottom image). In the second column, the beam artifacts of the Lidar scan are highlighted in the lit image.

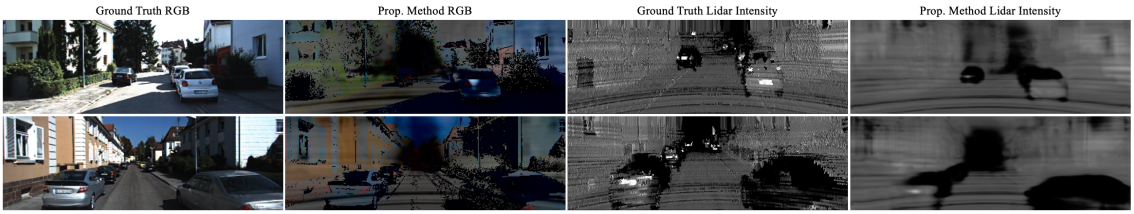

Figure 4.5: Examples of KITTI images lit to their ground truth lighting (second image column) using the proposed densified Lidar model.

4.4.2 Evaluation of Relighting via densified Lidar reflectance

Figure 4.5 shows several examples of lit images using the proposed relighting method and dense reflectance maps output from the modified MINE framework. We see that, while global structures are present in the dense liar intensity maps, the model has learned significant Lidar beam artifacts. It also has learned that the sky is a surface in the scene that has reflectance properties, which is not physically correct. Furthermore, we see that the quality of the generated dense reflectance maps performs the worse when used for relighting in comparison to both the reflectance maps from original proposed method as well as those from illumination invariant color spaces, see final row of Table 4.1.

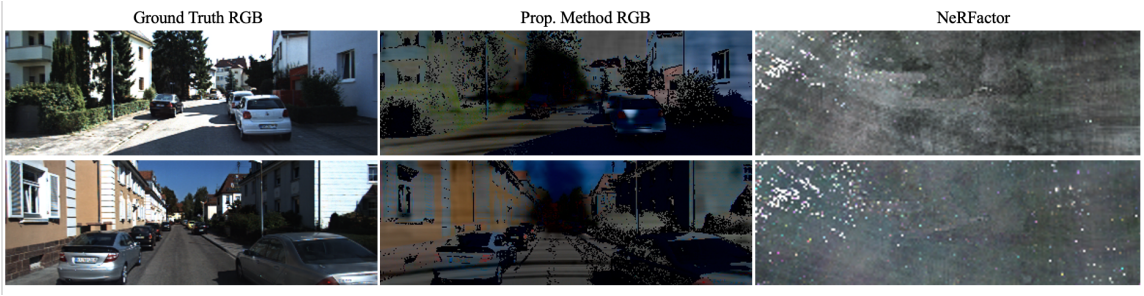

Figure 4.6: Examples of KITTI images lit to their ground truth lighting (second image column) using the proposed densified Lidar model, and examples of NeRFactor relighting the same KITTI scene are shown in the final column. Note that NeRFactor struggles to disentangle scene properties, ultimately yielding images with no clear geometry or lighting.

In general, it appears that NeRF models struggle learning dense information from sparse ground truth, and future work will need to investigate how to better model Lidar information within a ray tracing framework.

However, in comparison to the state-of-the-art NeRF relighting methods, modeling lighting changes using the proposed dense reflectance maps produces significantly better lit images. We compare relighting KITTI scenes to their ground truth lighting conditions using the proposed dense method and NeRFactor [163]. Qualitative results are shown in Figure 4.6. We see that our proposed method using modified MINE to create a dense scene reflectance map significantly outperforms NeRFactor, which cannot even learn accurate scene geometry for the outdoor scenes, let alone disambiguate the intrinsic scene factors needed to successfully model light transport.

4.5 Discussion and Conclusion

In this chapter, we propose a deterministic, single image relighting framework that leverages 3D information from point clouds to transfer illumination effects for real scenes. Our results indicate that Lidar intensity can be a significantly useful representation for modeling/capturing the underlying reflectance of a scene, and that this

information can be used in simple image formation models to relight images. Other avenues and potential extensions of this work would be to incorporate a weather and imaging pipeline models into this framework to better capture all of the myriad influences of illumination that impact image appearance.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Georgios Albanis, Nikolaos Zioulis, Anastasios Dimou, Dimitrios Zarpalas, and Petrod Daras. Dronepose: Photorealistic uav-assistant dataset synthesis for 3d pose estimation via a smooth silhouette loss. *arXiv preprint arXiv:2008.08823*, 2020.

[2] Hassan Abu Alhaija, Siva Karthik Mustikovela, Andreas Geiger, and Carsten Rother. Geometric image synthesis. In *Asian Conference on Computer Vision*, pages 85–100. Springer, 2018.

[3] Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. *arXiv preprint arXiv:1802.10151*, 2018.

[4] Naif Alshammari, Samet Akcay, and Toby P Breckon. On the impact of illumination-invariant image pre-transformation for contemporary automotive semantic scene understanding. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1027–1032. IEEE, 2018.

[5] Naif Alshammari, Samet Akçay, and Toby P Breckon. Multi-task learning for automotive foggy scene understanding via domain adaptation to an illumination-invariant representation. *arXiv preprint arXiv:1909.07697*, 2019.

[6] José M Álvarez Alvarez and Antonio M Ĺopez. Road detection based on illuminant invariance. *IEEE transactions on intelligent transportation systems*, 12(1):184–193, 2010.

[7] Alexander Andreopoulos and John K Tsotsos. On sensor bias in experimental methods for comparing interest-point, saliency, and recognition algorithms. volume 34, pages 110–126. IEEE, 2012.

[8] S Annadurai. Fundamentals of digital image processing. Pearson Education India, 2007.

[9] Asha Anoosheh, Eirikur Agustsson, Radu Timofte, and Luc Van Gool. Combogan: Unrestrained scalability for image domain translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 783–790, 2018.

[10] Tristan Aumentado-Armstrong, Alex Levinshtein, Stavros Tsogkas, Konstantinos G Derpanis, and Allan D Jepson. Cycle-consistent generative rendering for 2d-3d modality translation. *arXiv preprint arXiv:2011.08026*, 2020.

[11] Mihai Badea, Corneliu Florea, Laura Florea, and Constantin Vertan. Can we teach computers to understand art? domain adaptation for enhancing deep networks capacity to de-abstract art. *Image and Vision Computing*, 77:21–32, 2018.

[12] Fateme Bahri, Moein Shakeri, and Nilanjan Ray. Online illumination invariant moving object detection by generative neural network. *arXiv preprint arXiv:1808.01066*, 2018.

[13] David Bau, Hendrik Strobelt, William Peebles, Bolei Zhou, Jun-Yan Zhu, Antonio Torralba, et al. Semantic photo manipulation with a generative image prior. *arXiv preprint arXiv:2005.07727*, 2020.

[14] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1901.09887*, 2019.

[15] S. A. Bhukhanwala and T. V. Ramabadran. Automated global enhancement of digitized photographs. *IEEE Transactions on Consumer Electronics*, 40(1):1–10, Feb 1994.

[16] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021.

[17] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan Barron, and Hendrik Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. *Advances in Neural Information Processing Systems*, 34, 2021.

[18] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2. *arXiv preprint arXiv:2001.10773*, 2020.

[19] Xu Cao and Katashi Nagao. Point cloud colorization based on densely annotated 3d shape dataset. In *International Conference on Multimedia Modeling*, pages 436–446. Springer, 2019.

[20] Alexandra Carlson, Katherine A Skinner, and Matthew Johnson-Roberson. Modeling camera effects to improve deep vision for real and synthetic data. *arXiv preprint arXiv:1803.07721*, 2018.

[21] Alexandra Carlson, Katherine A Skinner, Ram Vasudevan, and Matthew Johnson-Roberson. Sensor transfer: Learning optimal sensor effect image augmentation for sim-to-real domain adaptation. *IEEE Robotics and Automation Letters*, 4(3):2431–2438, 2019.

[22] Alexandra Carlson, Ram Vasudevan, and Matthew Johnson-Roberson. Shadow transfer: Single image relighting for urban road scenes. *arXiv preprint arXiv:1909.10363*, 2019.

[23] Dario Carrea, Antonio Abellan, Florian Humair, Battista Matasci, Marc-Henri Derron, and Michel Jaboyedoff. Correction of terrestrial lidar intensity channel using oren–nayar reflectance model: An application to lithological differentiation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 113:17–29, 2016.

[24] Jiawei Chen, Janusz Konrad, and Prakash Ishwar. A cyclically-trained adversarial network for invariant representation learning. *arXiv preprint arXiv:1906.09313*, 2019.

[25] Qifeng Chen and Vladlen Koltun. A simple model for intrinsic image decomposition with depth cues. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 241–248, 2013.

[26] Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances In Neural Information Processing Systems*, 2019.

[27] Hejin Cheong, Eunjung Chae, Eunsung Lee, Gwanghyun Jo, and Joonki Paik. Fast image restoration for spatially varying defocus blur of imaging sensor. *Sensors*, 15(1):880–898, 2015.

[28] George Chogovadze, Rémi Pautrat, and Marc Pollefeys. Controllable data augmentation through deep relighting. *arXiv preprint arXiv:2110.13996*, 2021.

[29] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[30] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[31] Florent Couzinie-Devy, Jian Sun, Karteek Alahari, and Jean Ponce. Learning to estimate and remove non-uniform image blur. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1075–1082, 2013.

[32] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[33] Manuel DeLanda. *Assemblage theory*. Edinburgh University Press, 2016.

[34] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[35] Dawa Derksen and Dario Izzo. Shadow neural radiance fields for multi-view satellite photogrammetry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1152–1161, 2021.

[36] Steven Diamond, Vincent Sitzmann, Stephen Boyd, Gordon Wetzstein, and Felix Heide. Dirty pixels: Optimizing image classification architectures for raw sensor data. *arXiv preprint arXiv:1701.06487*, 2017.

[37] L Díaz-Vilariño, H González-Jorge, M Bueno, P Arias, and I Puente. Automatic classification of urban pavements using mobile lidar data and roughness descriptors. *Construction and Building Materials*, 102:208–215, 2016.

[38] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In *Quality of Multimedia Experience (QoMEX), 2016 Eighth International Conference on*, pages 1–6. IEEE, 2016.

[39] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[40] Chris Donahue, Zachary C Lipton, Akshay Balsubramani, and Julian McAuley. Semantically decomposing the latent spaces of generative adversarial networks. *arXiv preprint arXiv:1705.07904*, 2017.

[41] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

[42] Aysegul Dundar, Ming-Yu Liu, Ting-Chun Wang, John Zedlewski, and Jan Kautz. Domain stylization: A strong, simple baseline for synthetic to real image domain adaptation. *arXiv preprint arXiv:1807.09384*, 2018.

[43] Majed El Helou, Ruofan Zhou, Sabine Süsstrunk, Radu Timofte, Mahmoud Afifi, Michael S Brown, Kele Xu, Hengxing Cai, Yuzhong Liu, Li-Wen Wang, et al. Aim 2020: Scene relighting and illumination estimation challenge. In *European Conference on Computer Vision*, pages 499–518. Springer, 2020.

[44] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[45] Sanja Fidler. talk title. Neural Information Processing Systems, 2016.

[46] Graham D Finlayson, Mark S Drew, and Cheng Lu. Entropy minimization for shadow removal. *International Journal of Computer Vision*, 85(1):35–57, 2009.

[47] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. volume 17, pages 1737–1754. IEEE, 2008.

[48] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[49] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016.

[50] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2018)*, 37(6):To appear, 2018.

[51] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao(Richard) Zhang. Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019)*, To appear(To appear):To appear, 2019.

[52] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[53] Michael D Grossberg and Shree K Nayar. Modeling the space of camera response functions. volume 26, pages 1272–1282. IEEE, 2004.

[54] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[55] Maximilien Guislain, Julie Digne, Raphaëlle Chaine, Dimitri Kudelski, and Pascal Lefebvre-Albaret. Detecting and correcting shadows in urban point clouds and image collections. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 537–545. IEEE, 2016.

[56] Xiaofeng Han, Huan Wang, Jianfeng Lu, and Chunxia Zhao. Road detection based on the fusion of lidar and image data. *International Journal of Advanced Robotic Systems*, 14(6):1729881417738102, 2017.

[57] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[58] Sharif Hasan, Josef Jansa, and Norbert Pfeifer. Brdf-based correction of colorized aerial lidar point clouds. In *2015 Joint Urban Remote Sensing Event (JURSE)*, pages 1–4. IEEE, 2015.

[59] Tomáš Hodaň, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta N Sinha, and Brian Guenter. Photorealistic image synthesis for object instance detection. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 66–70. IEEE, 2019.

[60] Shiyu Huang, Deva Ramanan, undefined, undefined, undefined, and undefined. Expecting the unexpected: Training detectors for unusual pedestrians with adversarial imposters. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 00:4664–4673, 2017.

[61] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[62] Richard Sewall Hunter. Accuracy, precision, and stability of new photoelectric color-difference meter. In *Journal of the Optical Society of America*, volume 38, pages 1094–1094, 1948.

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

[64] Michael Janner, Jiajun Wu, Tejas D Kulkarni, Ilker Yildirim, and Josh Tenenbaum. Self-supervised intrinsic image decomposition. *Advances in Neural Information Processing Systems*, 30, 2017.

[65] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[66] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 746–753. IEEE, 2017.

[67] Christopher Kanan and Garrison W Cottrell. Color-to-grayscale: does the method matter in image recognition? *PloS one*, 7(1):e29740, 2012.

[68] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[69] Alireza G Kashani, Michael J Olsen, Christopher E Parrish, and Nicholas Wilson. A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, 15(11):28099–28128, 2015.

[70] Diana Kim, Bingchen Liu, Ahmed Elgammal, and Marian Mazzone. Finding principal semantics of style in art. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 156–163. IEEE, 2018.

[71] Taeyoung Kim, Yu-Wing Tai, and Sung-Eui Yoon. Pca based computation of illumination-invariant space for road detection. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 632–640. IEEE, 2017.

[72] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[73] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pages 2539–2547, 2015.

[74] Pierre-Yves Laffont, Adrien Bousseau, and George Drettakis. Rich intrinsic image decomposition of outdoor scenes from multiple views. *IEEE transactions on visualization and computer graphics*, 19(2):210–224, 2012.

[75] Jean-Francois Lalonde, Alexei A Efros, and Srinivasa G Narasimhan. Estimating the natural illumination conditions from a single outdoor image. *International Journal of Computer Vision*, 98(2):123–145, 2012.

[76] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869, 2017.

[77] Louis Lettry, Kenneth Vanhoey, and Luc Van Gool. Darn: a deep adversarial residual network for intrinsic image decomposition. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1359–1367. IEEE, 2018.

[78] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. Mine: Towards continuous depth mpi with nerf for novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12578–12588, 2021.

[79] Junxuan Li, Hongdong Li, and Yasuyuki Matsushita. Lighting, reflectance and geometry estimation from 360∘ panoramic stereo. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10586–10595. IEEE Computer Society, 2021.

[80] Xiaolu Li and Yu Liang. Surface characteristics modeling and performance evaluation of urban building materials using lidar data. *Applied Optics*, 54(15):4750–4759, 2015.

[81] Xiaolu Li, Yu Liang, and Lijun Xu. Bidirectional reflectance distribution function based surface modeling of non-lambertian using intensity data of light detection and ranging. *JOSA A*, 31(9):2055–2063, 2014.

[82] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 453–468, 2018.

[83] Zhengqi Li and Noah Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–387, 2018.

[84] Andrew Liu, Shiry Ginosar, Tinghui Zhou, Alexei A Efros, and Noah Snavely. Learning to factorize and relight a city. *arXiv e-prints*, pages arXiv–2008, 2020.

[85] Jitao Liu, Songmin Dai, and Xiaoqiang Li. Pccn: Point cloud colorization network. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3716–3720. IEEE, 2019.

[86] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708, 2017.

[87] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10551–10560, 2019.

[88] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[89] Yunfei Liu, Yu Li, Shaodi You, and Feng Lu. Unsupervised learning for intrinsic image decomposition from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3248–3257, 2020.

[90] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[91] Will Maddern, Alex Stewart, Colin McManus, Ben Upcroft, Winston Churchill, and Paul Newman. Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles. In *Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China*, volume 2, page 3, 2014.

[92] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1429–1437, 2019.

[93] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth. *arXiv preprint arXiv:1612.05079*, 2016.

[94] Colin McManus, Winston Churchill, Will Maddern, Alexander D Stewart, and Paul Newman. Shady dealings: Robust, long-term visual localisation using illumination invariance. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 901–906. IEEE, 2014.

[95] G. Messina, A. Castorina, S. Battiato, and A. Bosco. Image quality improvement by adaptive exposure correction techniques. In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 1, pages I–549–52 vol.1, July 2003.

[96] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[97] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*, 2019.

[98] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.

[99] Matthias Müller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, 126(9):902–919, 2018.

[100] Thomas Nestmeyer, Jean-François Lalonde, Iain Matthews, and Andreas Lehrmann. Learning physics-guided face relighting under directional light. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5124–5133, 2020.

[101] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[102] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

[103] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2856–2865, 2021.

[104] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Training object class detectors with click supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6374–6383, 2017.

[105] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[106] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, pages 319–345. Springer, 2020.

[107] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[108] Mattis Paulin, Jérôme Revaud, Zaid Harchaoui, Florent Perronnin, and Cordelia Schmid. Transformation pursuit for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3646–3653. IEEE, 2014.

[109] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. *ACM Transactions on Graphics (TOG)*, 38(4):78, 2019.

[110] Manikandasriram Srinivasan Ramanagopal, Cyrus Anderson, Ram Vasudevan, and Matthew Johnson-Roberson. Failing to learn: Autonomously identifying perception failures for self-driving cars. *CoRR*, abs/1707.00051, 2017.

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

[112] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[113] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proc. Eur. Conf. Comput. Vis.*, pages 102–118. Springer, 2016.

[114] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[115] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.

[116] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Neural radiance fields for outdoor scene relighting. *arXiv preprint arXiv:2112.05140*, 2021.

[117] Manuele Sabbadin, Gianpaolo Palma, Francesco Banterle, Tamy Boubekeur, and Paolo Cignoni. High dynamic range point clouds for real-time relighting. In *Computer Graphics Forum*, volume 38, pages 513–525. Wiley Online Library, 2019.

[118] Christos Sakaridis, Dengxin Dai, Simon Hecker, and Luc Van Gool. Model adaptation with synthetic and real data for semantic dense foggy scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 687–704, 2018.

[119] Moein Shakeri and Hong Zhang. Illumination invariant representation of natural images for visual place recognition. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 466–472. IEEE, 2016.

[120] Li Shen, Ping Tan, and Stephen Lin. Intrinsic image decomposition with non-local texture cues. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2008.

[121] Li Shen, Chuohao Yeo, and Binh-Son Hua. Intrinsic image decomposition using a sparse representation of reflectance. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2904–2915, 2013.

[122] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2020.

[123] Takayuki Shinohara, Haoyi Xiu, and Masashi Matsuoka. Point2color: 3d point cloud colorization using a conditional generative network and differentiable rendering for airborne lidar. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1062–1071, 2021.

[124] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2107–2116, 2017.

[125] K. Simonyan, A. Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.

[126] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[127] Leon Sixt, Benjamin Wild, and Tim Landgraf. Rendergan: Generating realistic labeled data. *Frontiers in Robotics and AI*, 5:66, 2018.

[128] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[129] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[130] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, 2004.

[131] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021.

[132] Xin Sun, Kun Zhou, Yanyun Chen, Stephen Lin, Jiaoying Shi, and Baining Guo. Interactive relighting with dynamic brdfs. *ACM Transactions on Graphics (TOG)*, 26(3):27, 2007.

[133] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. *arXiv preprint arXiv:2202.05263*, 2022.

[134] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[135] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

[136] Thoreau Rory Tooke, Nicholas C Coops, and Jessica Webster. Predicting building ages from lidar data with random forests for building energy modeling. *Energy and Buildings*, 68:603–610, 2014.

[137] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[138] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 969–977, 2018.

[139] Edgar Tretschk, Ayush Tewari, Michael Zollhöfer, Vladislav Golyanik, and Christian Theobalt. Demea: Deep mesh autoencoders for non-rigidly deforming objects. *arXiv preprint arXiv:1905.10290*, 2019.

[140] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 551–560, 2020.

[141] VSR Veeravasarapu, Constantin Rothkopf, and Ramesh Visvanathan. Adversarially tuned scene generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2587–2595, 2017.

[142] Chunxue Wang, Huayan Zhang, and Ligang Liu. Total generalized variation-based retinex image decomposition. *The Visual Computer*, 37(1):77–93, 2021.

[143] Ende Wang, Yong Li, Anan Sun, Huashuai Gao, Jingchao Yang, and Zheng Fang. Road detection based on illuminant invariance and quadratic estimation. *Optik*, 185:672–684, 2019.

[144] Mengjiao Wang, Zhixin Shu, Shiyang Cheng, Yannis Panagakis, Dimitris Samaras, and Stefanos Zafeiriou. An adversarial neuro-tensorial approach for learning disentangled representations. *International Journal of Computer Vision*, 127(6-7):743–762, 2019.

[145] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018.

[146] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3dn: 3d deformation network. In *CVPR*, 2019.

[147] Yi Zhang Siyuan Qiao Zihao Xiao Tae Soo Kim Yizhou Wang Alan Yuille Weichao Qiu, Fangwei Zhong. Unrealcv: Virtual worlds for computer vision. *ACM Multimedia Open Source Software Competition*, 2017.

[148] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Interpretable transformations with encoder-decoder networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5726–5735, 2017.

[149] Po-Wei Wu, Yu-Jing Lin, Che-Han Chang, Edward Y Chang, and Shih-Wei Liao. Relgan: Multi-domain image-to-image translation via relative attributes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5914–5922, 2019.

[150] Wayne Wu, Kaidi Cao, Cheng Li, Chen Qian, and Chen Change Loy. Transgaga: Geometry-aware unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[151] Daan Wynen, Cordelia Schmid, and Julien Mairal. Unsupervised learning of artistic styles with archetypal style analysis. *Advances in Neural Information Processing Systems*, 31:6584–6593, 2018.

[152] Xiaoyan Xing, Yanlin Qian, Sibo Feng, Yuhan Dong, and Jiri Matas. Point cloud color constancy. *arXiv preprint arXiv:2111.11280*, 2021.

[153] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics (TOG)*, 37(4):126, 2018.

[154] Wai Yeung Yan, Ahmed Shaker, and Nagwa El-Ashmawy. Urban land cover classification using airborne lidar data: A review. *Remote Sensing of Environment*, 158:295–310, 2015.

[155] Ceyuan Yang, Yujun Shen, and Bolei Zhou. Semantic hierarchy emerges in deep generative representations for scene synthesis. *arXiv preprint arXiv:1911.09267*, 2019.

[156] Jordan Yaniv, Yael Newman, and Ariel Shamir. The face of art: landmark detection and geometric style in portraits. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019.

[157] Amirsaeed Yazdani, Tiantong Guo, and Vishal Monga. Physically inspired dense fusion networks for relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 497–506, 2021.

[158] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 75–83, 2020.

[159] Zhenqiang Ying, Ge Li, Xianghao Zang, Ronggang Wang, and Wenmin Wang. A novel shadow-free feature extractor for real-time road detection. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 611–615, 2016.

[160] Ye Yu, Abhimitra Meka, Mohamed Elgharib, Hans-Peter Seidel, Christian Theobalt, and William AP Smith. Self-supervised outdoor scene relighting. In *European Conference on Computer Vision*, pages 84–101. Springer, 2020.

[161] Zohreh Zahiri, Debra F Laefer, and Aoife Gowen. Characterizing building materials using multispectral imagery and lidar intensity data. *Journal of Building Engineering*, 44:102603, 2021.

[162] He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *arXiv preprint arXiv:1701.05957*, 2017.

[163] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021.

[164] Yi Zhang, Weichao Qiu, Qi Chen, Xiaolin Hu, and Alan Yuille. Unrealstereo: Controlling hazardous factors to analyze stereo vision. In *2018 International Conference on 3D Vision (3DV)*, pages 228–237. IEEE, 2018.

[165] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5057–5065. IEEE, 2017.

[166] Hao Zhou, Xiang Yu, and David W Jacobs. Glosh: Global-local spherical harmonics for intrinsic image decomposition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7820–7829, 2019.

[167] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph laplacian. In *ACM SIGGRAPH 2005 Papers*, pages 496–503. 2005.

[168] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[169] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.