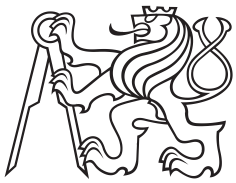
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Day to Night Image Style Transfer with Light Control

Bc. Josef Čech

Supervisor: Ing. David Hurych, Ph.D.
Supervisor–specialist: Ing. Tuan-Hung Vu, Ph.D.
Field of study: Cybernetics and Robotics
May 2021

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Ing. David Hurych, Ph.D., for the continuous support, patience, and motivation during the research and writing of this thesis. Special thanks go to Ing. Tuan-Hung Vu, Ph.D., who was always helpful and available for consultation in the field of domain adaptation and deep learning in general.

Last but not least, I would like to thank my parents and girlfriend, whose love and support made this work a pleasant undertaking (♡).

This work has been supported by CTU Valeo Scholarship and has also received funding from Valeo.ai.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Signature:

Prague, May 21, 2021

Abstract

Before a self-driving car becomes a matter of everyday life, it is essential to reliably solve safe driving in adverse conditions, such as night. Computer vision algorithms – that are vital for optimal control and decision-making systems – are typically data-driven.

Nevertheless, the acquisition of nighttime data is expensive, and hand-labeling is extremely laborious. Therefore, we address the challenging problem of data augmentation and propose a novel approach in day-to-night image translation with 3D-aware light control. We leverage classical rendering techniques as well as contemporary deep neural networks to produce photorealistic nighttime images.

With the illuminated nighttime images, we literally shed light on the real-world problem of object detection. Using a recent day-night driving dataset BDD100K [1], we train the object detector on several mixtures of real and fake (nighttime synthesized) images with the original scene annotations and evaluate its performance on real nighttime testing data. Experimental results show that our approach is on par or even outperforms competitive state-of-the-art methods for image translation. Furthermore, with a proper mixture of real and fake data, our method proposed boosts the detector performance.

Keywords: Artificial intelligence, computer vision, deep learning, generative adversarial networks, day-to-night image translation, data augmentation, object detection

Supervisor: Ing. David Hurych, Ph.D.
Valeo.ai, Valeo R&D Center
108 00 Prague 10, Sazečská 247/2
Czech Republic

Abstrakt

Předtím než se samořídící vozidla stanou věcí každodenního života, je nezbytné spolehlivě vyřešit bezpečné řízení v nepříznivých podmínkách, jako je noc. Algoritmy počítačového vidění – které jsou zásadní pro optimální řízení a rozhodovací systémy – typicky vyžadují mnoho dat.

Získávání nočních dat je však drahé a ruční anotování je extrémně pracné. Proto se zabýváme náročným problémem rozšiřování dat a navrhujeme nový přístup v překladu obrázku ze dne na noc s 3D kontrolou osvětlení. Pro vytvoření fotorealistických nočních obrázků využíváme jak klasické renderovací techniky, tak i moderní hluboké neuronové sítě.

S osvětlenými nočními obrázky doslova vrháme světlo na skutečný problém detekce objektů. Pomocí současného silničního datasetu denních a nočních obrázků BDD100K [1] trénujeme detektor objektů na několika směsích reálných a falešných (syntetizovaných nočních) obrázků s původními anotacemi scény a vyhodnocujeme jeho výkon na reálných nočních testovacích datech. Experimentální výsledky ukazují, že náš přístup je srovnatelný, nebo dokonce překonává konkurenční state-of-the-art metody pro překlad obrázků. Navíc se správnou směsí reálných a falešných dat naše navržená metoda posiluje výkon detektoru.

Klíčová slova: Umělá inteligence, počítačové vidění, hluboké učení, generativní adversariální sítě, překlad obrázku ze dne na noc, rozšíření dat, detekce objektů

Překlad názvu: Změna stylu obrazových dat z denních na noční s kontrolou osvětlení

Contents

Figures

Tables

Chapter 1

Introduction

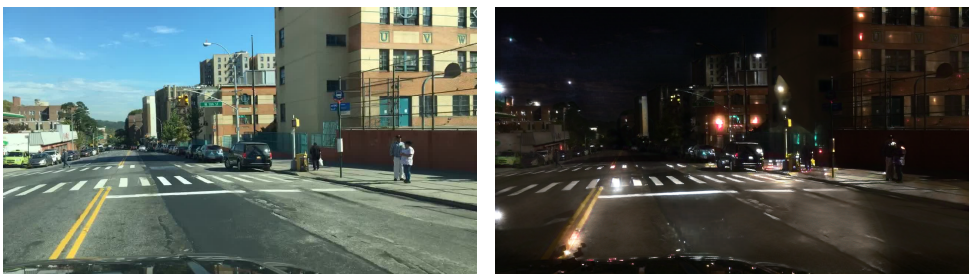
1.1 Motivation

Before a self-driving car becomes a matter of everyday life, it is essential to reliably solve safe driving in adverse conditions, e.g., rain, snowstorms, fog, dawn or dusk, and even complete darkness at night. The primary concern of this thesis is the challenging problem of driving at night.

To make the control of the autonomous vehicle possible, we must first perceive and understand the road context. That involves a range of high-level computer vision tasks, such as object detection, semantic segmentation, or object tracking. These techniques are usually data-driven, i.e., a large volume of diverse data is required.

Nevertheless, the acquisition of nighttime data is expensive. In contrast to clear daytime images, manual annotation of nighttime images is timely and far more accuracy-demanding. Given these circumstances, public road traffic datasets are usually lacking annotated nighttime data. Even though they are occasionally available, the number of pedestrians and instances of other classes is significantly reduced at night.

Common geometric augmentation techniques like cropping, flipping, or rotation can be applied with satisfactory results. Still, they can only produce limited variations of the same data. A tempting approach is to leverage affordable day images and synthesize their nighttime counterparts (see Figure 1.1). This is a notoriously difficult problem known as image style transfer.



(a) : Original daytime image. **(b) :** Artificial nighttime image.

Figure 1.1: Original daytime image and its nighttime counterpart by our method.

Imagine, there would be an even more advanced level of transformation that would allow us to manipulate the scene illumination.

1.2 Our Approach

In this thesis, we propose a novel approach in day-to-night image style transfer with 3D-aware light control. For the reader's convenience, we present our day-to-night translation pipeline in Figure 1.2. Let us now briefly describe these five steps. First, we estimate a depth map from a single image. Next, we reconstruct a 3D scene with the depth map guidance. Then, we illuminate the reconstructed surface. Classical rendering tools are employed to generate a 3D-aware lightmap. Based on the lightmap, we adjust the brightness of the original daytime image afterwards. As a final step, we use contemporary deep neural networks to turn the intermediate dimmed image into a photorealistic nighttime image. For further details, consult Chapter 3.

This day-to-night translation framework can easily simulate street lighting or turn on the ego-vehicle's headlights. In Chapter 5, we show our synthesized images along with translations by competitive state-of-the-art methods.

We anticipate that by training on synthesized images on its own or along with real data, the proposed augmentation strategy will lead towards a performance enhancement of computer vision algorithms. For testing purposes, we design an object detection benchmark in Chapter 6.

All of this raises two big questions.

- How realistic night images can we produce?
- Does it help to improve the detection performance on real testing data?

We discuss the visual image quality and experimental results in Chapter 7.

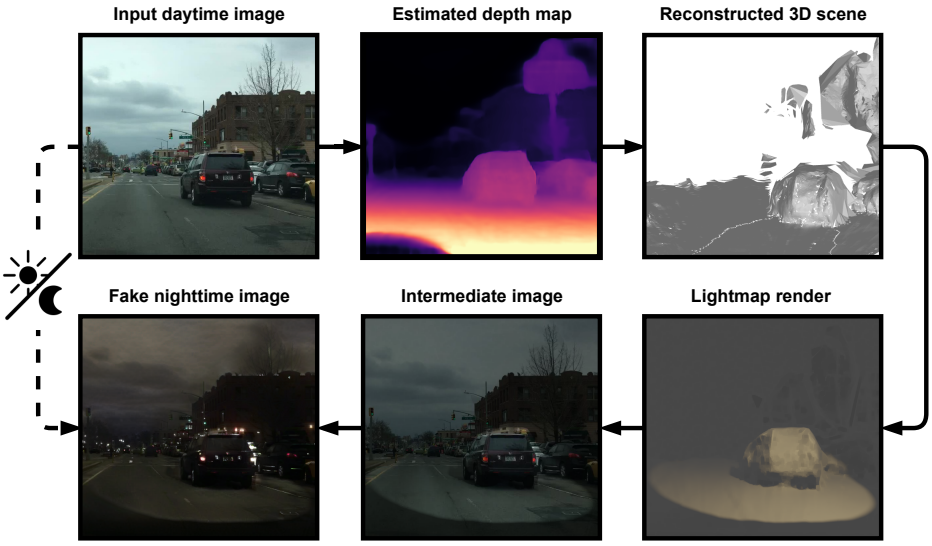


Figure 1.2: Day-to-night translation pipeline.

In summary, the contribution of this thesis is threefold.

1. Design of the day-to-night style transfer framework that allows us to control the amount, position, and direction of light sources.
2. Inspection of the performance of the object detector trained on various mixtures of real and fake (nighttime synthesized) images with the original scene annotations.
3. Correction of time-of-day misclassifications in BDD100K dataset [1].

Chapter 2

Related Work

Nowadays, learning-based methods are almost exclusively used for image-to-image style transfer. In particular, approaches leveraging generative adversarial networks are the most prominent.

Generative Adversarial Network. The generative adversarial network (GAN), introduced in [2], is a deep learning architecture based on a zero-sum game between a generator and a discriminator. The generator aims to create realistic fake samples to fool the discriminator, while the discriminator detects fake samples in a mixture of real and fake samples. See Section 3.7.1 for a mathematical formulation and further details.

The advent of this phenomenon has sparked a "GAN-rush" in the entire AI community and is receiving tremendous attention, with over 28 k papers published[1] since GAN introduction in 2014.

Let us mention at least a few remarkable studies that used GAN architecture in the past year. To enlarge a limited real dataset and enhance COVID-19 detection accuracy, CovidGAN [3] synthesizes chest X-ray images. Based on the appearance of two people, FamilyGAN [4] can predict what their child would look like. GAN helps to identify grape leaf disease in [5], segment skin lesions from dermoscopy images in [6], or visualize floods in [7]. Finally, a very recent House-GAN++ [8] automatically generates house floor plans.

Supervised Image-to-Image Translation. A popular state-of-the-art conditional GAN setting in [9], also known as a pix2pix framework, proved to be useful in many image-to-image translation tasks, including day-to-night. Nevertheless, a paired image dataset is required. That is too restrictive for us, and for the rest of this chapter, we dive into the area of unsupervised algorithms instead.

Unsupervised Image-to-Image Translation. A widely used utilization of GAN architecture for unpaired image-to-image translation is CycleGAN [10]. It basically consists of two generators and two discriminators. One generator performs image translation from domain X to domain Y. The other one translates in the opposite direction, i.e., from domain Y to domain X. To

[1] Papers from 2014 to 2021 related to *generative adversarial networks* according to Google Scholar.

determine the quality of the generated images, there is one discriminator corresponding to each translation direction. On top of the standard adversarial loss, a cycle consistency loss is defined to achieve higher authenticity and retain the structure between the source and target domain. We devote Section 3.7.2 to this attractive model design.

Many other unsupervised general-purpose, occasionally concurrent, methods like UNIT [11], DiscoGAN [12], DualGAN [13], CyCADA [14], ComboGAN [15], TransGaGa [16] are also built on the idea of the cycle consistency.

A more recent work on image synthesis from CycleGAN authors is CUT [17] that uses a multilayer, patchwise contrastive loss to maximize the mutual information between two patches, one of which is from the source domain and the other from the target domain. It should be mentioned that the cycle consistency loss is missing in this approach. For more information about the loss function, see Section 3.7.3. Compared to CycleGAN, faster training and memory efficiency are promised in CUT.

Day-to-Night Style Transfer. Let us now look at special methods that were designed to perform a day-to-night image translation.

Preserving objects after the day-to-night image translation is a central topic of [18] that introduces a structure-aware neural network AugGAN. As the name suggests, AugGAN is also based on GAN architecture. AugGAN incorporates a low-level semantic segmentation into the standard sum of adversarial and cycle consistency loss. Interestingly, in addition to the synthetic-to-synthetic and real-to-real, AugGAN can also perform synthetic-to-real transformations or vice versa.

ForkGAN [19] is another image synthesizing framework explicitly related to day-to-night image style transfer. Besides the adversarial loss, it also contains the cycle consistency loss and presents a fork-shaped learning approach. One of the branches provides a reconstruction loss, and the other is followed by a refinement stage for achieving more precise output images during inference.

As deep learning-based methods are laborious to tune, not predictable, produce lights at unnatural locations, cause deconvolution or checkerboard artifacts [20], and other random failures, classical approaches in day-to-night image translation are still emerging.

To the best of the authors' knowledge, [21] is the first work that uses classical techniques for day-to-night image translation with light control. This approach relies on the prior knowledge of semantic image segmentation. After converting the image to the HSV representation, the brightness is adjusted – guided by the semantic segmentation – independently for each class, e.g., the lane marking is set brighter. Random light splatters are added, and the final lightmap intensity is gradually decreased based on the coarse depth (see Figure 2.1 for a day-to-night translation example).

Night-to-Day Style Transfer. The opposite approach of image time-of-day attribute manipulation to improve visual tasks in adverse conditions is a night-to-day image translation.

(a) : Original daytime image. **(b) :** Artificial nighttime image.

Figure 2.1: Example of classical day-to-night translation [21].

In [18], the night-to-day image translation is viewed as a beneficial preprocessing step to ease the labeling of real nighttime images. In [19], they use the night-to-day transition direction to deal with the challenge of rainy nights. However, while performing the night-to-day translation, black moving objects, e.g., cars, may disappear. That is a safety shortcoming that can easily lead to an accident when used in real traffic. Other studies that address night-to-day image translation are [22], [23], [24], or [25].

Continuous Style Transfer. More advanced methods, such as DLOW [26], DNI [27], SMIT [28], StarGAN v2 [29], HiDT [30], SAVI2I [31] or very recent CoMoGAN [32], even provide a continuous transition between different domains, e.g., day and night or various seasons and weather conditions.

Using these methods, a time-lapse effect can be created because all intermediate stages[2] are available.

Style Transfer with Relighting. A difficult problem of light and shadow manipulation has also been the subject of some recent papers.

To ensure proper shading, a 3D scene approximation is generally needed. In [33], they use a geometry-aware neural network and synthetic stereo images from multiple views. On the contrary, in [34], the geometry is acquired by a monocular depth estimation and coarse semantic image segmentation. Unfortunately, both methods consider the sun as the only source of light.

Impressive results of the pix2pix framework [9], without explicit geometry information involved, are presented in [35]. However, they plan to use 3D information in future work since it has been shown beneficial in other studies.

An explicit focus on shadow casting is a great advantage of these light-oriented approaches since it is not normally treated within general-purpose methods. Emphasizing the importance of the residual shadow removal for the translated image appearance, specialized methods are arising, e.g., DeshadowNet [36], Mask-ShadowGAN [37], RIS-GAN [38], or the recent one from [39].

A classical image illumination technique is proposed in [40]. At first, a random circular binary lightmap is generated. In order to obtain a more realistic light attenuation, the initial binary lightmap is subjected to the

[2] E.g., [dawn → day → dusk → night] for time-of-day transitions.

Euclidean distance transform. The lightmap is then applied to the original image, simulating a light source effect (brightening) or shadow effect (darkening) by pixel-wise addition or subtraction, respectively. The illumination changes can be global by modifying all pixel values uniformly or local by randomly choosing small regions.

A more recent variant [41] extends the previous one by more sophisticated ellipse-shaped lightmaps to simulate the illumination at different angles. Furthermore, they study global changes in contrast, sharpness, and color saturation. Note that the illumination changes in [40] and [41] are completely stochastic and do not take the 3D geometry of the scene into account, thereby generating naive lighting effects lacking realism.

Day-to-Night Style Transfer for Data Augmentation. In the context of our work, it is worth mentioning that the resulting artificial images are abundantly used for data augmentation and improvement of high-level computer vision tasks, such as car detection [18], [19], [25], [42], pedestrian detection [25], lane detection [43], semantic segmentation [23], [24] or background subtraction [40], [41].

Closely related to our work, in [42], they examine the influence of synthetic nighttime image augmentation on the car detection task. For day-to-night image translation, they use pure CycleGAN without further architecture modifications. To simplify the task, they exclude problematic situations from the data, i.e., only the images with clear or partly cloudy weather and at least one car present are chosen. The remaining images are cropped such that a car's lane is centered and resized to 256×256 resolution to overcome high computational demands. Thereafter, small and occluded bounding boxes are removed. Leaving the simplification of the input data aside, it is exciting to see that these artificial nighttime images help improve the detection performance on real testing data.

Chapter 3

Method Proposed

This chapter describes a theoretical basis of the novel approach in day-to-night image style transfer with 3D-aware light control. Along the way, we extensively illustrate individual steps.

3.1 Depth Estimation

Before we can build the 3D geometry model of the scene, we need to acquire depth information. In particular, a pixel-wise depth map associated with the input image is of our primary interest. Ordinarily, depth maps are not part of datasets[3]. Therefore, we are looking for a suitable monocular depth estimation technique.

The monocular depth estimation is an active and fast-growing area of research, and a lot of successful learning-based methods with amazing performance have been published in the past years, e.g., SfMLearner [44], DORN [45], Monodepth [46], its improved version Monodepth2 [47] or PackNet-SfM [48]. See Figure 3.1 for a side-by-side comparison.

As a depth estimation framework, we choose Monodepth2 (the official PyTorch implementation) since it is one of the most recent methods with

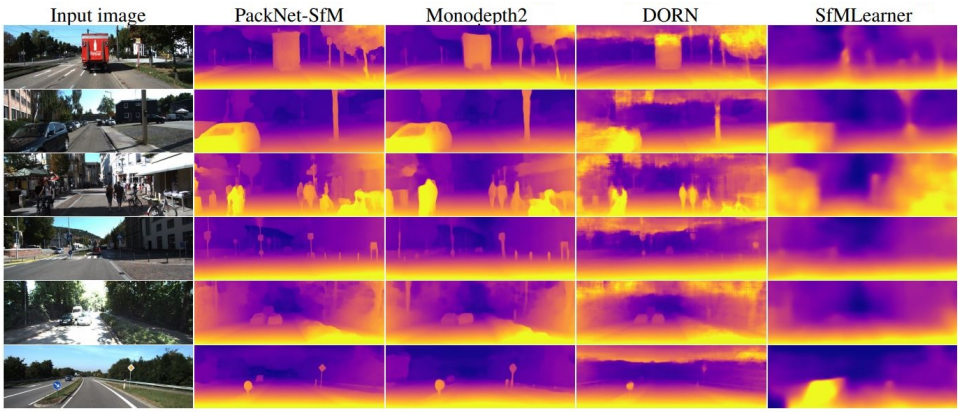


Figure 3.1: Comparison of recent depth estimation methods [48].

[3]Indeed, having a real dataset with exact per-pixel depth maps is not even possible.

promising results and a leading position among the unsupervised methods [49][4]. Compared to PackNet-SfM, it supports not only monocular but also stereo training modality. While processing stereo images is naturally convenient for this task, the stereo training modality becomes handy when the stereo images are present in the dataset. With respect to depth estimation of temporal objects (e.g., moving cars), Monodepth2 claims to be superior to other methods.

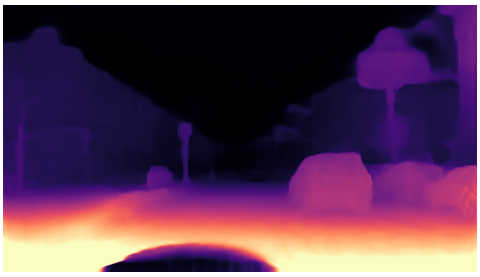
To learn more about the training process of Monodepth2, see Section 5.3.1.

3.2 3D Point Cloud Back-projection

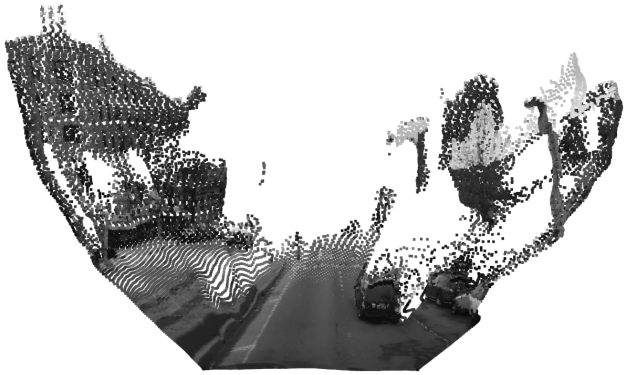
The input image extended by the estimated (or ideally, ground truth) depth map is used to recover the original 3D scene. In other words, our goal here is to back-project each image coordinate $\mathbf{x} \in \mathbb{P}^2$ of the input image to the corresponding 3D point $\mathbf{X} \in \mathbb{P}^3$, as shown in Figure 3.2.

Besides the precise depth information, the quality of the point cloud back-projection also relies on accurate knowledge of the intrinsic parameters, i.e., the focal length f and the coordinates of the principal point $\mathbf{p}$, of the camera used.

(a) : Input daytime image.



(b) : Estimated depth map.



(c) : Back-projected point cloud.

Figure 3.2: 3D scene recovery.

[4] An independent overview paper.

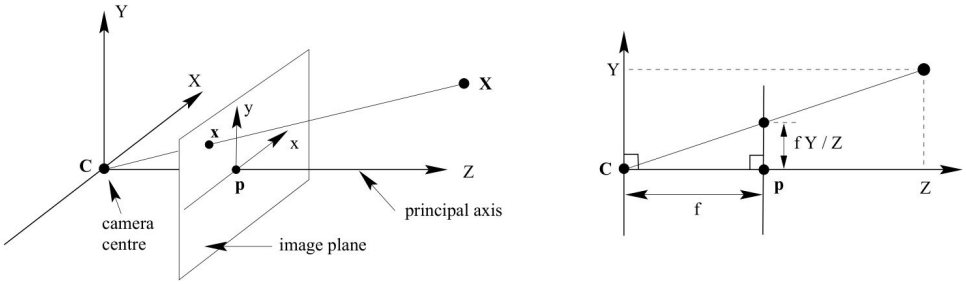


Figure 3.3: Pinhole camera projective geometry [50].

Following the core idea of the central projection mapping of the pinhole camera model from [50], we define an inverse mapping (see a geometric motivation in Figure 3.3).

Definition 3.1. *Let* $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}$ *be image coordinates and* $\mathbf{X} = \begin{bmatrix} X & Y & Z \end{bmatrix}^{\mathrm{T}}$ *world coordinates. Let* f *be the focal length and* $\mathbf{p} = \begin{bmatrix} p_x & p_y \end{bmatrix}^{\mathrm{T}}$ *be the coordinates of a principal point. Given a depth value* d_{xy} *at* $\begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}$*, an inverse central projection mapping of a pinhole camera model from the image coordinates* $\begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}$ *to the world coordinates* $\begin{bmatrix} X & Y & Z \end{bmatrix}^{\mathrm{T}}$ *is defined as*

$$X = \frac{d_{xy}(x - p_x)}{f}, \tag{3.1}$$

$$Y = \frac{d_{xy}(y - p_y)}{f}, \tag{3.2}$$

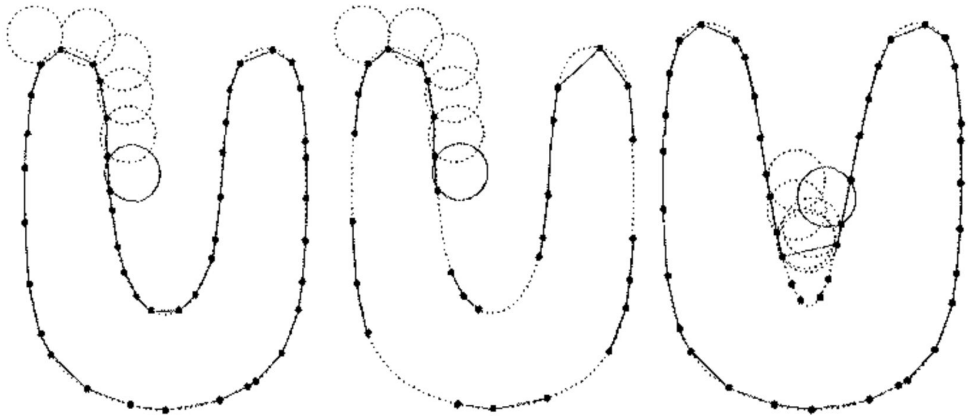
$$Z = d_{xy}. \tag{3.3}$$

An open-source 3D data processing library Open3D 0.9.0 [51] is used for the practical implementation of this procedure.

3.3 Surface Mesh Reconstruction

To illuminate the 3D scene, it is necessary to reconstruct a surface representation of the back-projected point cloud that will reflect light. In our pipeline, this is done by a memory and time-efficient ball-pivoting algorithm [52] that performs a triangular mesh interpolation.

The ball-pivoting algorithm is motivated by the physics of a rolling ball (see Figure 3.4a). As an initialization step, we drop the virtual ball onto the point cloud. If the ball is large enough and does not fall through, it will get caught between three points. These three points form a seed triangle. From this place, the ball is rolling along the edges of the seed triangle until it settles in a new place and forms another triangle from the previous two points and the new one. The ball keeps pivoting and adding more triangles in the same manner until a complete mesh is formed.

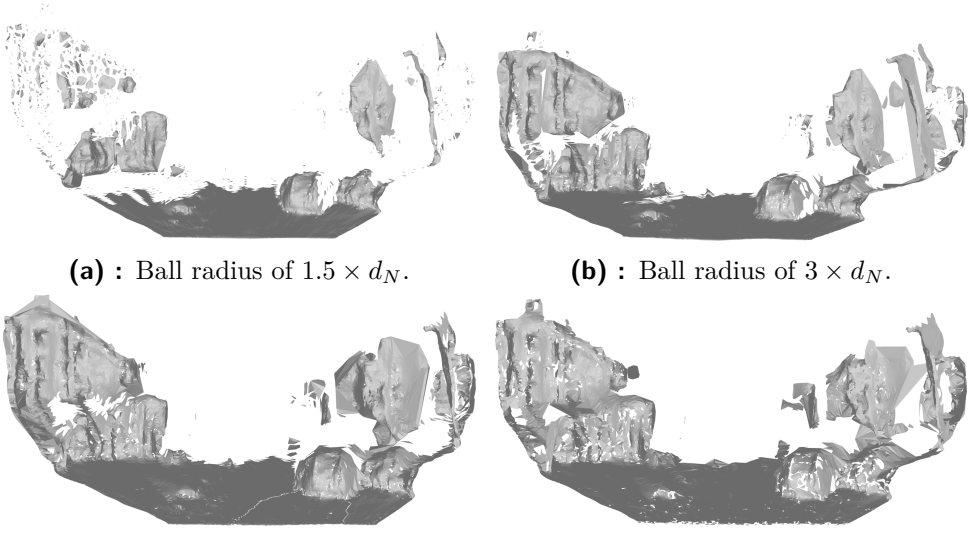


(a) : Optimal radius. **(b) :** Too small radius. **(c) :** Too large radius.

Figure 3.4: The ball-pivoting algorithm in 2D [52].

The radius of the ball is to be determined empirically based on the scale of the mesh, usually as a multiple of the mean distance between neighboring points d_N. If the ball is designed too small, some of the edges will be missed (cause of holes, see Figure 3.4b). On the other hand, if the ball is designed too large, some of the points will not be reached (loss of details, see Figure 3.4c). As shown in Figure 3.5, the reconstructed mesh provides the best visual experience with the ball radius five times larger than the mean distance between neighboring points. This radius is used in further steps.

To incorporate the ball-pivoting algorithm into our pipeline, we use the recently released PyMeshLab [53], a Python library interface to the 3D mesh processing open-source software MeshLab [54].



(a) : Ball radius of $1.5 \times d_N$. **(b) :** Ball radius of $3 \times d_N$.

(c) : Ball radius of $5 \times d_N$. **(d) :** Ball radius of $7 \times d_N$.

Figure 3.5: Reconstructed mesh surface for various ball radii in terms of the mean distance between neighboring points d_N.

3.4 Lightmapping

After obtaining the surface representation of the 3D scene geometry, it is time for enlightenment and shading.

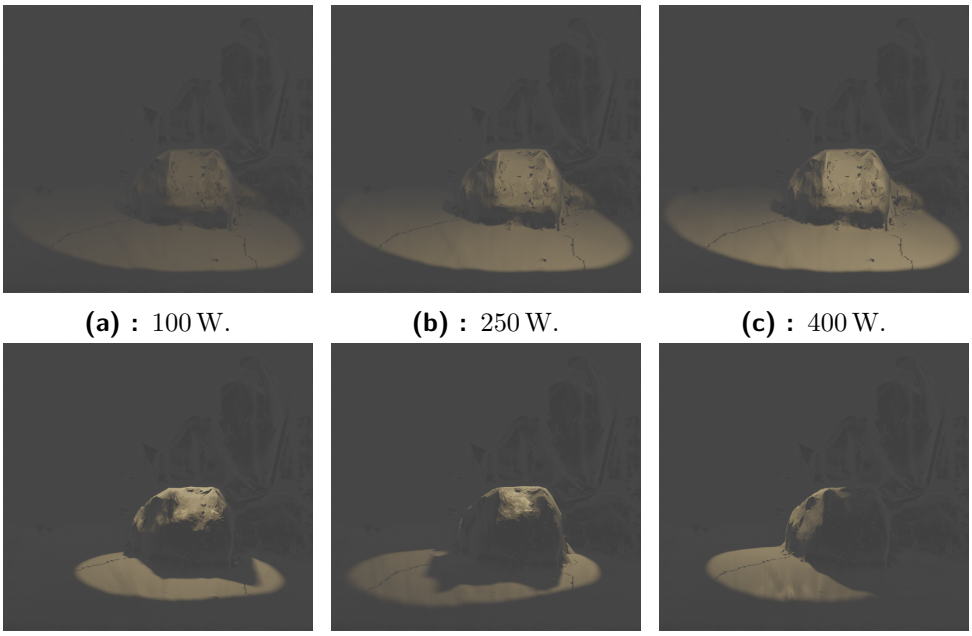
Without a doubt, street lamps are significant and important light sources at night. Another dominant source of illumination are cars' headlights. In particular, the headlights of the ego vehicle are inherently present in the picture. We can model both types of light sources as a spotlight that emits a cone-shaped beam of light from the tip of the cone, in a given direction [55].

Figures 3.6a-3.6c depict various power settings with simple lightmap renders. Along with the intensity, this approach elegantly allows us to fully control the position and orientation of the light source. Figures 3.6d-3.6f demonstrate the controllability and 3D awareness of the illumination.

The desired light setup is an essential input of the method. Note that the light sources are to be engineered manually[5]. This is no exception since the lights are normally hand-crafted by the user, even in some state-of-the-art methods that focus on light control, e.g., [33].

The illuminated mesh is rendered and projected into the camera plane.

For the light source rendering, we use a 3D computer graphics open-source software Blender 2.82 [55].



(a) : 100 W. **(b) :** 250 W. **(c) :** 400 W.

(d) : Front. **(e) :** Front-right. **(f) :** Front-left.

Figure 3.6: Lightmap renders for various power settings (a)-(c), positions and rotations (d)-(f) of the spotlight.

[5] See Appendix A for possible automation of this step.

3.5 Lightmap Filtering

Due to inaccurate mesh reconstruction, unenlightened black segments may appear in the lightmap render. To diminish the significance of these faults, we smooth the lightmap by implementing a closing morphological transformation, a useful technique for closing small holes in the picture [56].

Before we define the closing transformation, we need to introduce two fundamental operations, erosion and dilation. Quoting essentially verbatim from [57], we formally define erosion and dilation as follows.

Definition 3.2. *Let A and B be sets in $\mathbb{Z}^2$, assuming B to be a structuring element.*

The erosion of A by B is the set of all points z such that B, translated by z, is contained in A. In the equation form,

$$A \ominus B = \{z \,|\, (B)_z \subseteq A\}\,. \tag{3.4}$$

The dilation of A by B is the set of all displacements, z, such that $\hat{B}$ and A overlap by at least one element, where $\hat{B}$ denotes the reflection of B about its origin. In the equation form,

$$A \oplus B = \left\{z \,|\, (\hat{B})_z \cap A \neq \emptyset\right\}. \tag{3.5}$$

The definition above assumes a binary image. Let us extend this definition to a gray-scale image.

Definition 3.3. [57]
Let the function $f(x, y) : \mathbb{Z}^2 \mapsto \mathbb{Z}$ represent a gray-scale image and assume that $b(x, y) : \mathbb{Z}^2 \mapsto \mathbb{Z}$ is a structuring element.

The erosion of the image f by structuring element b at any location (x, y) is defined as the minimum value of the image in the region coincident with b when the origin of b is at (x, y). In the equation form,

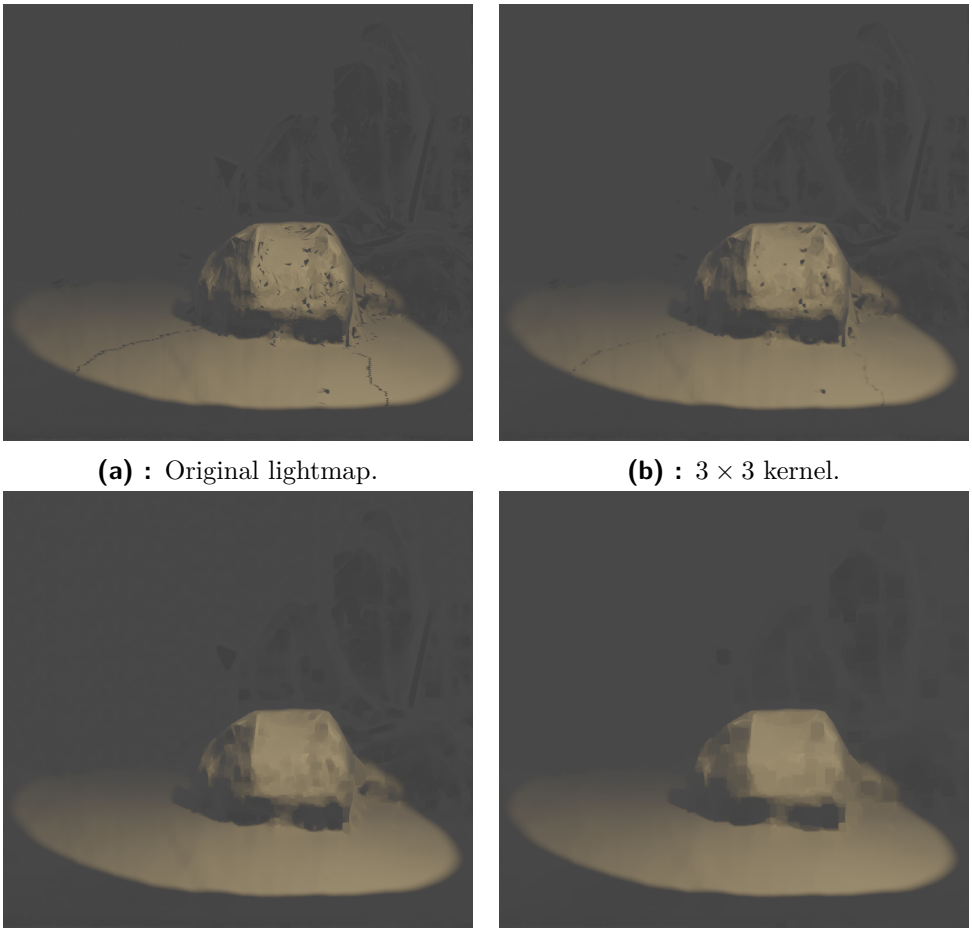
$$[f \ominus b]\,(x, y) = \min_{(s,t)\in b} \{f(x + s, y + t)\}\,, \tag{3.6}$$

where x and y are incremented through all values required so that the origin of b visits every pixel in f.

The dilation of the image f by structuring element b at any location (x, y) is defined as the maximum value of the image in the window outlined by $\hat{b}$ when the origin of $\hat{b}$ is at (x, y), where $\hat{b}$ denotes the reflection of b about its origin. In the equation form,

$$[f \oplus b]\,(x, y) = \max_{(s,t)\in b} \{f(x - s, y - t)\}\,, \tag{3.7}$$

where x and y are incremented through all values required so that the origin of b visits every pixel in f.



(a) : Original lightmap. **(b) :** 3×3 kernel.

(c) : 6×6 kernel. **(d) :** 9×9 kernel.

Figure 3.7: Closing the lightmap with various kernel dimensions.

Finally, we can define the closing transformation for gray-scale images.

Definition 3.4. [57]
Let the function $f(x, y) : \mathbb{Z}^2 \mapsto \mathbb{Z}$ *represent a gray-scale image and assume that* $b(x, y) : \mathbb{Z}^2 \mapsto \mathbb{Z}$ *is a structuring element.*

The closing of image f *by structuring element* b *is defined as a dilation followed by erosion. In the equation form,*

$$f \bullet b = (f \oplus b) \ominus b. \tag{3.8}$$

In the case of an RGB image, each channel is processed independently [58].

In Figure 3.7, we can see the effect of the smoothing operation for various dimensions of the kernel. The lightmaps filtered by the morphological closing transformation form a more coherent impression. As a tradeoff between filtering intensity and preserving sharp details, we choose the 6×6 kernel dimension.

In this step, we take advantage of a computer vision open-source library OpenCV 3.4.8 [59].

3.6 Intermediate Image

Once the processing of the lightmap is accomplished, we adjust the brightness of the input daytime image proportionally to the pixel values in the lightmap.

First of all, the RGB input image is converted to an alternative HSV[6] color representation [60], where the V-component allows us to directly access the brightness of the image. The lightmap is scaled so that the brightest pixel is at the maximum of the numerical representation range[7]. Then, we perform a pixel-wise multiplication of the scaled lightmap and the V-component of the input image to produce a dimmed variant, hereinafter referred to as an intermediate image. The intermediate image with adjusted brightness is shown in Figure 3.8c.

The OpenCV 3.4.8 [59] is employed for the RGB-to-HSV conversion.

3.7 Touch of Deep Learning

In this section, we introduce the concept of a generative adversarial network (Section 3.7.1) and its two adaptations for unsupervised image-to-image translation CycleGAN (Section 3.7.2) and CUT (Section 3.7.3).

There are two main purposes of the translation framework.

1. A direct translation of the daytime image to its nighttime counterpart, i.e., only day-to-night style translation without light control.

2. A finalizing step of our pipeline. The intermediate image is subjected to the deep learning-based generative model to obtain an authentic and photorealistic nighttime image (see Figure 3.8d). From now on, we will refer to this translation as intermediate-to-night.

We are going to learn more about the training of these methods in Chapter 5.

3.7.1 Generative Adversarial Network

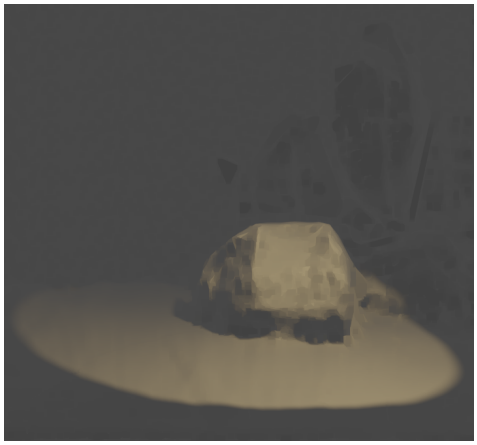
As we stated in Chapter 2, the generative adversarial network (GAN) [2] is a popular concept for generating new artificial samples from a given data distribution p_{data}. It consists of two differentiable functions, a generator G and a discriminator D. Both functions are typically represented by neural networks.

According to [61], the process of learning can be viewed as a competition between the generator and the discriminator. The generator produces fake samples $\hat{\boldsymbol{x}} = G(\boldsymbol{z})$ from a random noise $\boldsymbol{z}$ (e.g., uniform distribution). The discriminator adversarially tries to distinguish between real samples and samples produced by the generator. More closely, the discriminator estimates the probability that the input sample is from the real data distribution. The

[6] HSV stands for hue, saturation, value. It is also known as HSL (hue, saturation, lightness) or HSB (hue, saturation, brightness).

[7] E.g., 255 in the case of an 8-bit unsigned integer.

(a) : Input daytime image.



(b) : Filtered lightmap.

(c) : Intermediate image.

(d) : Fake nighttime image[8].

Figure 3.8: Key results of the image manipulation.

generator and discriminator alternate periodically in the training phase. One is always kept constant during the training phase of the other one.

Defining a loss function[9]

$$\begin{aligned}\mathcal{L}_{\mathrm{GAN}}(G,\,D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[\log D(\boldsymbol{x})\right] + \\ + \mathbb{E}_{\boldsymbol{z}\sim p_{\mathrm{noise}}(\boldsymbol{z})}\left[\log\left(1 - D(G(\boldsymbol{z}))\right)\right],\end{aligned} \tag{3.9}$$

this competition can be mathematically expressed as a zero-sum game

$$G^* = \arg\min_G \max_D \; \mathcal{L}_{\mathrm{GAN}}(G,\,D). \tag{3.10}$$

An optimum – called Nash equilibrium [62] – is reached when the generator produces samples so realistic that the generator cannot distinguish between real and fake samples, i.e., its decision for any input is equivalent to flipping a coin. At such a moment, the training is at the end, and the desired generator of plausible artificial data G^* is obtained.

[8] Translated from the intermediate image by CUT [17] (see details in Section 5.3.2).

[9] The symbol $\mathbb{E}$ denotes an expected value over all data instances of the corresponding distribution.

3.7.2 CycleGAN

Let $\{\boldsymbol{x}_i\}_i^N$, where $\boldsymbol{x}_i \in X$ for each i, and $\left\{\boldsymbol{y}_j\right\}_j^M$, where $\boldsymbol{y}_j \in Y$ for each j, be two unaligned image sets[10] from different domains X, Y and let $p_{\text{data}}(\boldsymbol{x})$, $p_{\text{data}}(\boldsymbol{y})$ be the respective data distributions. The main goal of CycleGAN [10] is to learn a transformation

$$G : X \mapsto Y, \tag{3.11}$$

i.e., a generator that translates the image from domain X to domain Y, such that the resulting artificial image $\hat{\boldsymbol{y}} = G(\boldsymbol{x})$, $\boldsymbol{x} \in X$ fits into the distribution of images from the domain Y.

The generator is trained in a similar way as the general GAN in Section 3.7.1. However, the source domain image is taken as input instead of the random noise. The adversarial loss of the generator G is then

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G,\, D_Y,\, X,\, Y) = \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}(\boldsymbol{y})}\left[\log D_Y(\boldsymbol{y})\right] + \\ + \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\log\left(1 - D_Y(G(\boldsymbol{x}))\right)\right].\end{aligned} \tag{3.12}$$

Simultaneously, an inverse transformation

$$F : Y \mapsto X, \tag{3.13}$$

is learned. The generator F yields a fake image $\hat{\boldsymbol{x}} = F(\boldsymbol{y})$, $\boldsymbol{y} \in Y$, imitating the style of domain X. Analogously to (3.12), we define the adversarial loss of the generator F as

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(F,\, D_X,\, Y,\, X) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\log D_X(\boldsymbol{x})\right] + \\ + \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}(\boldsymbol{y})}\left[\log\left(1 - D_X(F(\boldsymbol{y}))\right)\right].\end{aligned} \tag{3.14}$$

In effect, the CycleGAN architecture consists of two GANs (i.e., two generators G, F, and two associated discriminators D_X, D_Y), as shown in Figure 3.9a.

To avoid structural changes during translation, CycleGAN introduces a cycle consistency regularization based on the composition of the two generators. The forward cycle consistency, i.e., F composed with G, can be written as

$$F(G(\boldsymbol{x})) = \hat{\boldsymbol{x}} \approx \boldsymbol{x}, \quad \boldsymbol{x} \in X, \tag{3.15}$$

and likewise, the backward cycle consistency, i.e., G composed with F, as

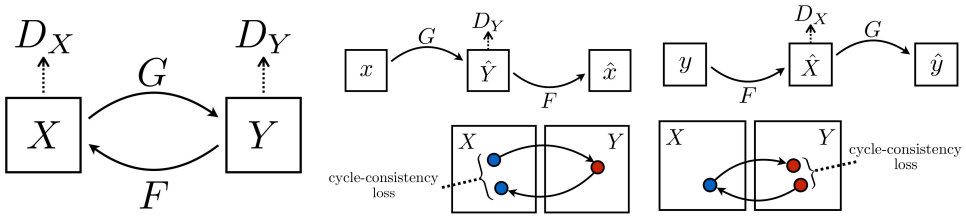
$$G(F(\boldsymbol{y})) = \hat{\boldsymbol{y}} \approx \boldsymbol{y}, \quad \boldsymbol{y} \in Y. \tag{3.16}$$

In other words, by transforming the image back and forth, we should get close to the input image. Consult Figures 3.9b and 3.9c for the block schemes of (3.15) and (3.16), respectively.

Next, we follow [10] by defining a cycle consistency loss

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G,\, F,\, X,\, Y) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[\|F\left(G(\boldsymbol{x})\right) - \boldsymbol{x}\|_1\right] + \\ + \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}(\boldsymbol{y})}\left[\|G\left(F(\boldsymbol{y})\right) - \boldsymbol{y}\|_1\right].\end{aligned} \tag{3.17}$$

[10] For simplicity, hereafter, we omit the subscript i and j as in the original paper [10].



(a) : Principal architecture. **(b) :** Forward cycle consistency. **(c) :** Backward cycle consistency.

Figure 3.9: Block diagrams of the CycleGAN model [10].

Both adversarial losses (3.12), (3.14) – the key expressions of GANs to generate realistic images – combined with the cycle consistency loss (3.17) form a full objective of CycleGAN

$$\begin{aligned}\mathcal{L}(G,\,F,\,D_X,\,D_Y,\,X,\,Y) = \mathcal{L}_{\text{GAN}}(G,\,D_Y,\,X,\,Y) + \\ + \mathcal{L}_{\text{GAN}}(F,\,D_X,\,Y,\,X) + \\ + \lambda\mathcal{L}_{\text{cyc}}(G,\,F,\,X,\,Y),\end{aligned} \tag{3.18}$$

where $\lambda \in \mathbb{R}$ is a scaling factor. The optimization problem is then

$$G^*,\,F^* = \arg\min_{G,\,F}\max_{D_X,\,D_Y} \mathcal{L}(G,\,F,\,D_X,\,D_Y,\,X,\,Y). \tag{3.19}$$

At the end of the training, we obtain a pair of plausible generators G^*, F^*. Assuming X and Y represent the day and night domains, respectively, we only need to keep the generator G^* for our purposes. The rest can be discarded.

3.7.3 Contrastive Unpaired Translation

Another approach to designing the generator G is proposed in [17]. The notation and task requirements in the following are adopted from Section 3.7.2.

The spirit of contrastive learning in CUT is a calculation of cross-entropy loss

$$\ell(\boldsymbol{v},\,\boldsymbol{v}^+,\,\boldsymbol{v}^-) = -\log\left[\frac{\exp\left(\boldsymbol{v}\cdot\boldsymbol{v}^+/\tau\right)}{\exp\left(\boldsymbol{v}\cdot\boldsymbol{v}^+/\tau\right) + \sum_{n=1}^{N}\exp\left(\boldsymbol{v}\cdot\boldsymbol{v}_n^-/\tau\right)}\right], \tag{3.20}$$

where $\boldsymbol{v} \in \mathbb{R}^K$, $\boldsymbol{v}^+ \in \mathbb{R}^K$ and $\boldsymbol{v}^- \in \mathbb{R}^{N\times K}$ are the output, corresponding, and N noncorresponding vectorized (K-dimensional) image patches, respectively, and $\tau \in \mathbb{R}$ is a scaling factor.

Equation (3.20) yields the probability that the corresponding patch $\boldsymbol{v}^+$ will be selected from all provided input patches $\boldsymbol{v}^+$ and $\boldsymbol{v}^-$.

Patches from multiple layers of the network are passed through a two-layer perceptron H and combined to form a loss called PatchNCE[11].

[11] Please refer to [17] and [63] for the exact $\mathcal{L}_{\text{PatchNCE}}$ formulation.

A final loss combines the adversarial loss (3.12) with the PatchNCE loss on images from domain X. To penalize structural changes, the PatchNCE loss on images from domain Y is involved. The full objective of CUT is then

$$\begin{aligned}\mathcal{L}(G,\, D_Y,\, H,\, X,\, Y) = \mathcal{L}_{\text{GAN}}(G,\, D_Y,\, X,\, Y) + \\ + \lambda_X \mathcal{L}_{\text{PatchNCE}}(G,\, H,\, X) + \\ + \lambda_Y \mathcal{L}_{\text{PatchNCE}}(G,\, H,\, Y),\end{aligned} \tag{3.21}$$

where $\lambda_X \in \mathbb{R}$ and $\lambda_Y \in \mathbb{R}$ are scaling factors.

Equation (3.21), as usual, is to be optimized

$$G^* = \arg \min_G \max_{D_Y} \mathcal{L}(G,\, D_Y,\, H,\, X,\, Y), \tag{3.22}$$

to get a plausible generator G^*.

Chapter 4

Datasets

A vast amount of diverse datasets exist within the autonomous driving community. What is more, new datasets are constantly being released. In the last two years, these are, for example, nuScenes [64], Honda 3D Dataset (H3D) [65], PreSIL [66], Argoverse [67], A*3D [68], Dark Zurich [69], Waymo Open Dataset [70], Lyft Level 5 [71], Audi Autonomous Driving Dataset (A2D2) [72] or Aachen Day-Night Dataset [73]. Most notably, with an emphasis on 3D scene understanding.

There are various specializations of datasets, and it can be difficult to find the right one. As we may have noticed, some datasets, such as Cityscapes [74] or KITTI [75], became more popular over time and were frequently used in various benchmarks due to their versatility.

Nevertheless, a lack of night images is a common drawback of the majority of publicly available datasets. On the contrary, the NightOwls dataset [76] – a rare exception – contains only night images. Fortunately, there are a few datasets that involve both daytime and nighttime images, and we focus on them in Section 4.1.

4.1 Day-Night Datasets

Unlabeled Day-Night Datasets. Although they contain day and night data, some datasets do not provide any form of content annotation.

The Alderley Day/Night Dataset [77] offers one lap (22 km) of driving at the Nurburgring racing circuit in Germany and a short journey (8 km) through the suburb of Alderley in Australia. The same routes were traveled and recorded at both daytime and nighttime. Pairs of images – capturing the same scene under different lighting conditions – were manually aligned but not annotated.

The Oxford RobotCar Dataset [78] consists of over 100 traversals of the same route (10 km) through Oxford in England and addresses a long-term road vehicle autonomy in different weather conditions and times of day (6 % of traversals are at night). Like in the Alderley Day-Night Dataset, neither bounding boxes nor semantic segmentations are provided in the Oxford RobotCar Dataset. For some tasks, this is enough, but we essentially rely on object annotations.

(Partially) Labeled Day-Night Datasets with Limitations. In the Raincouver dataset [79], there are 326 semantically segmented images with 95 nighttime frames. The Dark Zurich dataset [69] comprises 8377 images with a great night ratio (36 %). However, only 151 nighttime images are annotated with pixel-wise semantic segmentation. The Nighttime Driving dataset [80] contains 35000 images. In addition to daytime (18 %) and nighttime (22 %), they also distinguish civil (20 %), nautical (20 %), and astronomical (20 %) twilight. For testing purposes, a pixel-wise semantic segmentation is provided for 50 manually chosen highly diverse night frames. The testing data are the only available at the moment[12].

We are concerned that these datasets contain too few annotated images to work well with deep learning approaches. Therefore, we are more interested in bigger day-night datasets.

The Mapillary Vistas Dataset [81] promises a large dataset with day and night images and a comprehensive pixel-wise instance-level semantic segmentation. Since the night ratio is not specified and time-of-day labels are not provided, we decided to assign the labels manually on our own. Several tens of thousands of images later, we found 28 and a single night image in the whole training and validation set, respectively.

Similar to Mapillary Vistas, night images – without the actual ratio being specified – are also announced in other huge state-of-the-art datasets, such as ApolloScape [82] or Argoverse [67]. Further analysis is needed for these datasets. Nevertheless, the manual revision is extremely time-consuming, and the utility is apparently not guaranteed. In the following, let us look at large datasets explicitly stating the portion of night images.

Suitable Day-Night Datasets. Table 4.1 summarizes in more detail the rest of the day-night datasets that suit our needs.

Because no dataset in Table 4.1 contains semantic image segmentation, we are limited to object detection experiments in Chapter 6. We also list the capturing frequency since it is essential for the depth estimator to have training videos or consecutive temporal frames[13].

Dataset	Annotated Images	Annotation Type[14]	Night Images [%]	Capturing Frequency [Hz]
A*3D[15] [68]	39 k	2D/3D-BB	30.0	55
BDD100K [1]	80 k	2D-BB	39.9	30
KAIST [83]	95 k	2D-BB	34.4	20
nuScenes [64]	40 k	3D-BB	11.6	12
Waymo OD [70]	230 k	2D/3D-BB	11.2	10

Table 4.1: Suitable day-night datasets.

[12] The training data are still being prepared for public release.
[13] In this case, daytime records are sufficient.
[14] Available for all images.
[15] Not yet publicly available. The download link is still marked as in progress.

In relation to night traffic, it is noteworthy that the KAIST dataset [83] also provides FIR (far-infrared) images aligned with RGB images. The KAIST community is working on a newer version of the dataset [84]. It should include stereo images, LiDAR point clouds, and dense depth maps. We tried to obtain these nonpublic data directly from the authors but met with little success.

According to our analysis, a total of four publicly available datasets, i.e., BDD100K [1], KAIST [83], nuScenes [64], and Waymo OD [70], are suitable for our purposes. Reasonable size, the uppermost ratio of night images, and high frame rate are the clear motivation for choosing BDD100K dataset. Moreover, it is widely used in related studies [19], [24], [25], [42].

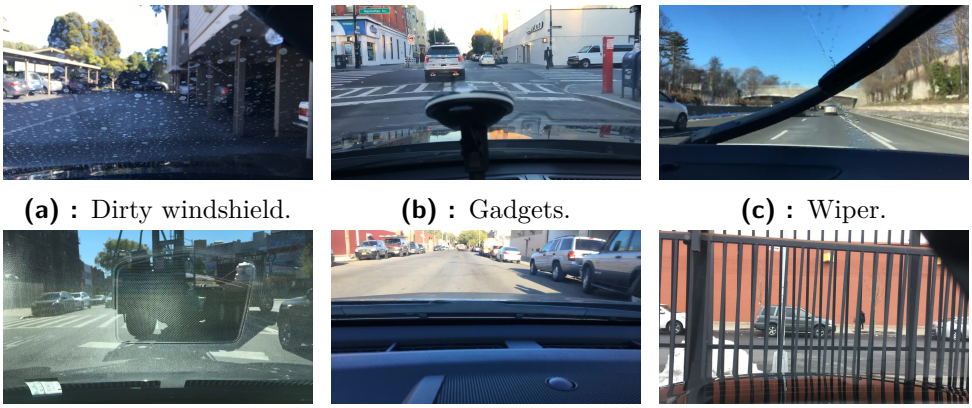
4.2 Berkeley DeepDrive Dataset

4.2.1 Dataset Overview

The BDD100K [1] is a large-scale driving dataset. It is one of the richest, if not the richest, publicly available day-night road traffic datasets regarding night images. Nearly the same number of day and night images brings a great opportunity to study day-night image style transfer.

Adverse weather conditions, including snow, rain, or fog, are naturally present in the dataset. The crowdsourcing model with tens of thousands of drivers ensures diverse geographical distribution and variable camera positions. Challenging but realistic scenarios (e.g., a wiper in motion, hood or dashboard in camera view, reflections of the interior on the windshield, or drops on the windshield) are very frequent in BDD100K (for better insight, see Figure 4.1).

The dataset is based on 100 k video clips (40 seconds each). One image frame is taken for annotation from each video clip at the tenth second, yielding a set of 100 k different images. Ten classes are extensively annotated with bounding boxes. We choose a subset of six common moving objects, i.e., car, person, bus, truck, bicycle, and motorcycle, for the evaluation of our models.



(a) : Dirty windshield. **(b) :** Gadgets. **(c) :** Wiper.

(d) : Reflections. **(e) :** Hood, dashboard. **(f) :** Fence in view.

Figure 4.1: Challenging scenarios in BDD100K [1].

A pixel-wise instance-level[16] semantic image segmentation is available for a 10 k sample – also referred to as BDD10K – of the dataset. Indirectly proving the difficulty of such a task, only 230 (about 2.9 %) public night images are segmented. Beyond the bounding box labels and semantic segmentation, the BDD100K also covers lane marking detection, drivable area segmentation, or multiple object tracking.

4.2.2 Custom Data Split

The data are split into training (70 k), validation (10 k), and testing (20 k) sets. Ground truth labels are hidden within the testing set, which is intended to be evaluated via a submission server. However, due to ongoing maintenance, it is not available[17]. To overcome this inconvenience and evaluate the models properly, we make use of the abundance of data in the training set by separating a subset (20 k) for testing.

As a result, a new data split arises, i.e., training (50 k), validation (10 k), and testing (20 k), where the validation set is left as is and the original unlabeled test set is not considered at all. Keep in mind that individual images were acquired from different video clips and are sufficiently independent. The custom split does not change the time-of-day distribution.

4.2.3 Correction of Time-of-Day Misclassifications

To easily distinguish between day and night domains, images are provided with time-of-day labels. To bridge the gap between day and night, a conjoint dawn/dusk label is introduced. Sometimes, e.g., in the tunnel, it is hard to decide, and the time-of-day label is set as undefined.

Nevertheless, the time-of-day labels are relatively often wrong. This has already been noticed in [42]. Some errors are at a day-night boundary (soft error). More seriously, some dark night images are marked as daytime and vice versa (hard error). Figure 4.2 shows several misclassified examples.

(a) : Nighttime images labeled as daytime.

(b) : Daytime images labeled as nighttime.

Figure 4.2: Examples of time-of-day misclassifications in BDD100K [1].

[16] The instance IDs were not released until April 2021.
[17] The submission server is expected to be launched by summer 2021.

Nighttime Label (31890 images)		
Error Type	Absolute Error	Relative Error [%]
Daytime	129	0.40
Dawn/Dusk	207	0.65
Daytime Label (41986 images)		
Error Type	Absolute Error	Relative Error [%]
Nighttime	237	0.56
Dawn/Dusk	214	0.51
Dawn/Dusk Label (5805 images)		
Error Type	Absolute Error	Relative Error [%]
Daytime	139	2.39
Nighttime	25	0.43

Table 4.2: Incorrect time-of-day labels in BDD100K [1][18,19].

Driven by suspiciousness, we checked all images in the dataset and found a total of 951 (about 1.2 %) incorrect time-of-day labels (see Table 4.2). In our opinion, the amount of errors is far from being negligible.

Having a list of misclassifications, the correction of time-of-day labels is an obvious and straightforward step. The correction is important so that our results (e.g., a performance of an object detector[20]) are not distorted. Table 6.1 analyzes the discrepancy in terms of object detection performance.

We shared the list of misclassifications – an outcome of this laborious task – with the BDD100K community. Much to our delight, it will be incorporated into the future updates of the BDD100K dataset. The list of misclassifications is also available in the attachment of this thesis.

[18] The numbers correspond to the sum of errors in the training and validation sets.

[19] The remaining 319 images belong to the undefined class.

[20] If trained on daytime and tested on nighttime images, the error could severely affect both the training phase (challenging nighttime images are seen) and the testing phase (much easier daytime images improve the performance).

Chapter 5

Artificial Images Generation

5.1 Quality Measures

5.1.1 Fréchet Inception Distance

To assess the quality of the generated images quantitatively, we compute a conventional and widely accepted metric, Fréchet Inception Distance (FID) [85], defined as

$$\mathrm{FID} = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{1/2}\right), \tag{5.1}$$

where (μ_r, Σ_r) and (μ_g, Σ_g) are the mean and covariance of the real and generated data, respectively. Since (5.1) measures the distance between the real and generated data distributions, the lower FID score indicates higher quality images.

It is able to detect intraclass mode dropping or various types of artifacts [86]. Furthermore, the FID score was shown to be a suitable quality measure even for high-resolution images [87], [88].

We use the official PyTorch implementation [89] for practical calculation.

5.1.2 User Study

As no metric is perfect, a user study might be helpful to assess the realism of the generated nighttime images. Using a randomly chosen batch of 30 real daytime images, three non-expert participants were shown an epoch-wise sequence of generated nighttime images[21] and asked to choose three best and three worst epochs for each base image from the batch[22].

Besides the overall authenticity of the synthesized images, the participants considered the number and severity of various artifacts, the amount of night style characteristics (e.g., some epochs generated non-night images with bright sky), and other common failures (see Figure 5.8). For the images processed by our pipeline, they also took into account the preservation of added lights and shadows since some epochs tend to remove them.

[21] One day-to-night translation of the base image for each epoch.

[22] Totaling 600 images for the entire batch and 20 epochs of training.

The results are presented in the form of a histogram, where the best and the worst epochs are shown. In addition, an unweighted combination (difference) is computed to get a tradeoff. Alongside the FID score, this metric serves as a simple detector of undesirable visual anomalies and should reveal epochs with persistently ill-suited images.

The evaluation by people is expensive. Consequently, the metric is biased as it only considers 30 images. Yet it gives us some basic understanding.

Are aesthetic images really better? By intuition, we would expect the visually plausible data to be the most beneficial for boosting the performance of vision tasks. However, despite common sense, this may not be true for a computer algorithm, i.e., solely the criterion of aesthetic images may not be enough. Conversely, even imperfect or unrealistic images can be helpful. Although beyond our computational capabilities, this is a fascinating aspect that needs to be tested experimentally, ideally, for every possible model.

5.2 Direct Day-to-Night Translation

5.2.1 CycleGAN and CUT

In order to have competitive artificial nighttime images, we train two existing state-of-the-art methods, i.e., CycleGAN [10] – the most widespread baseline in image-to-image translation – and its indirect successor CUT [17], to perform day-to-night image translation directly.

Translation models are intended for offline data augmentation. We wish to make the most of the available data, and we do not need the models to generalize well to unseen images, i.e., there is no need to worry about overfitting on the training set. However, we aim to use the same model to transform the validation set. Therefore, images from the validation set are incorporated among the training samples, yielding 31179 daytime and 23915 nighttime images. This is not ideal, as sharing the style distribution between training and validation data slightly distorts the independence of the validation set. It must be mentioned that the testing set – as designed in Section 4.2.2 – is always kept separate.

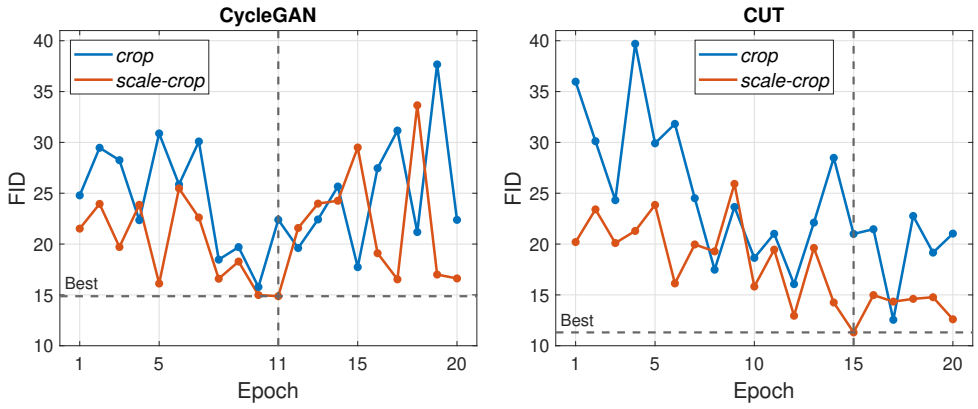
Note that it is recommended to use a cropped image in the training phase. Then, as the networks are fully convolutional, the trained models can be applied for image inference of arbitrary resolution. Figure 5.1 shows an evaluation of two training setups, i.e., *crop* and *scale-crop*.

In the *crop* setting, a 540×540 square is cut from the original image[23]. This crop size optimizes the memory usage of the GPU[24]. Other parameters are left default, e.g., learning rate 0.0002 or a single image batch size.

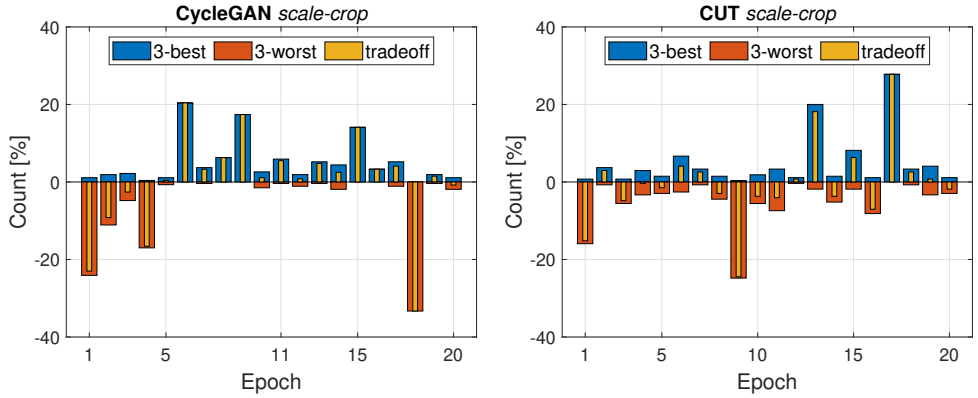
The *scale-crop* setting is the same as the *crop* setting, except the image is scaled to a resolution of 960×540 at first. Although at the lower resolution,

[23] The original image size in the BDD100K is 1280×720.

[24] Considering a single NVIDIA GeForce GTX 1080 Ti GPU with 11 GB memory size.



(a) : FID evaluation.



(b) : User study.

Figure 5.1: Evaluation of fake nighttime images generated from daytime by CycleGAN [10] and CUT [17].

the crop captures a wider field of view. Hence, the model is effectively provided with more contextual information about the scene.

Finally, we let both models train for 20 epochs. This could equivalently be expressed as an approximately two-week computation task on a single NVIDIA GeForce GTX 1080 Ti GPU.

Given the heavy memory demands of high-resolution image synthesis, the FID score in Figure 5.1a is generally calculated between a batch of 3 k artificial night images[25] and all training and validation real night images[26]. The minimum number of images in each batch required by [89] is 2048. However, is our 3 k subset variable enough? Out of curiosity, we translated a larger sample of 10 k artificial nighttime images per epoch with the models based on the *crop* setting to see how it affects the FID score. In Figure 5.2, we can see that the expansion of the batch of artificial night images does not bring new insight. In other words, the distribution of artificial night images is sufficiently described by the 3 k subset.

Looking at Figure 5.1a, we quickly found out that the *scale-crop* setting is

[25] Sampled randomly but then kept fixed for all calculations.

[26] The number of images does not have to be equal. Data distribution is what matters.

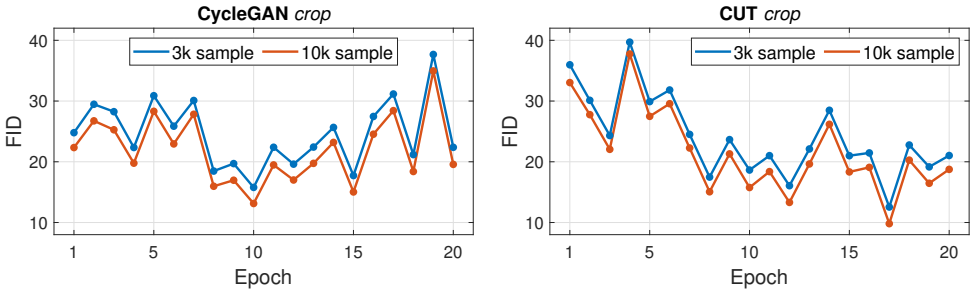


Figure 5.2: FID dependence on sample size for the *crop* setting.

better for CycleGAN as well as for CUT. Results of the inspection by people in Figure 5.1b show the outcomes of the corresponding best epochs to be visually plausible. The trained models associated with the best epochs of the *scale-crop* setting (according to the FID score) are used to generate two complete datasets (one for each method) of artificial night images. These data augmentations are then used to train the object detector in Chapter 6.

5.2.2 ForkGAN

The ForkGAN framework [19] complements the general-purpose methods CycleGAN and CUT with an approach that explicitly focuses on day-to-night image style transfer.

Thanks to the courtesy of Ziqiang Zheng[27], who shared the original non-public data with us, we have the opportunity to use the same images as they did in [19] for our data augmentation experiments. Due to the computational relief, we were able to use limited computing power for other experiments.

5.3 Day-to-Night Translation with Our Method

Recall the description of our method proposed in Chapter 3. In this section, we broaden the explanation of both learning-based steps of the pipeline, i.e., the depth estimation (Section 5.3.1) and GAN-based intermediate-to-night image translation (Section 5.3.2).

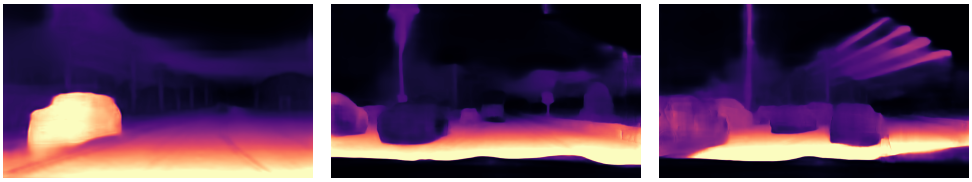
5.3.1 Monodepth2

Having the importance of the accurate depth estimation in mind (the rest of the pipeline strongly relies on the depth map quality), we make an appropriate effort to extensively test and analyze the Monodepth2 framework [47] – selected in Section 3.1 – and choose the best model possible.

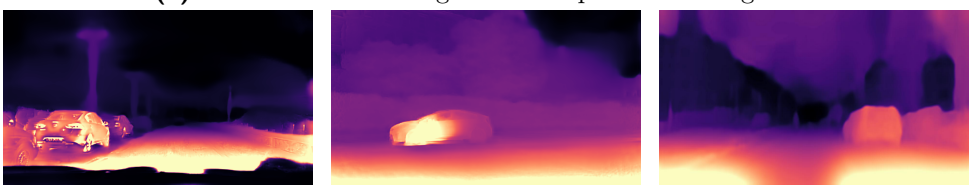
The Monodepth2 community offers several prediction models pretrained[28] on KITTI [75]. Namely, monocular (M), stereo (S), and mono plus stereo (MS) training modalities, each with resolutions of 640×192 and 1024×320.

[27] A leading author of ForkGAN from UISEE Technology (Beijing) Co., Ltd.

[28] The pretrained models are available from https://github.com/nianticlabs/monodepth2.



(a) : Too close oncoming or too far parallel-driving cars.



(b) : Overtrained. **(c) :** Lateral car. **(d) :** Sky.

Figure 5.3: Common depth estimation failures.

We compare the quality of the depth estimation for models pretrained on KITTI, fine-tuned[29] on BDD100K, and trained on BDD100K from scratch.

Monodepth2 needs the camera to move sufficiently between subsequent frames. That is why we carefully select 200 training video clips[30] in which the ego vehicle is still in motion. As the camera captures images at a high rate, even driving without stopping sometimes provides too little relative movement. With a small motion between frames, it is very hard to train the model. Therefore, we investigate the effect of frame skipping. Gradually increasing the number of skipped frames, we take every second, fourth, sixth, eighth, and tenth frame of the sequence.

We train with four different image resolutions[31], i.e., scaled to 1280×704, 1024×576, 512×288 (for faster training) and cropped to 640×192 (to maintain the KITTI resolution). According to the resolution, we adjust a batch size to fit into GPU memory (i.e., 12, 10, 4, and 1 for resolutions 640×192, 512×288, 1024×576, and 1280×704, respectively). Other parameters are left default, e.g., learning rate 0.0001, and each model is trained for 20 epochs.

To have a rough idea of time complexity, the training on a single NVIDIA GeForce GTX 1080 Ti GPU takes about five hours per epoch for the lower resolutions (640×192, 512×288) and about ten hours per epoch for the higher resolutions (1024×576 and 1280×704).

Our requirements for depth maps are hard to express by any metric. Instead, a subjective visual evaluation is used to have a better insight and control over specific objectives (which are indeed also visual). To assess the quality of the predicted depth maps, we focus on sharp estimation of spatial objects, correct sky prediction, continuous depth transition on the road, temporal object deletion, and other common failures (see Figure 5.3). The quality of the model is hard to be judged from only one image frame. Thus, we calculate the depth prediction for entire video clips.

[29] Initialized with KITTI pretrained (MS, 640×192).
[30] The limit was set because of the high computational demands of Monodepth2 training.
[31] Note that the height and width have to be divisible by 32.

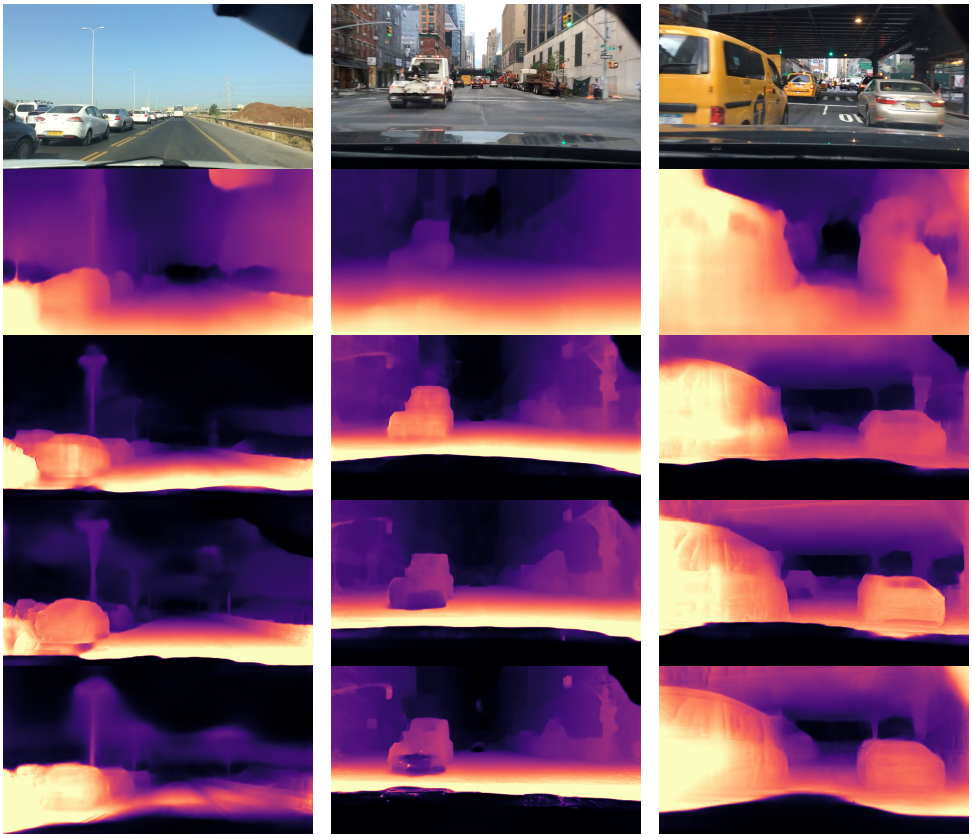
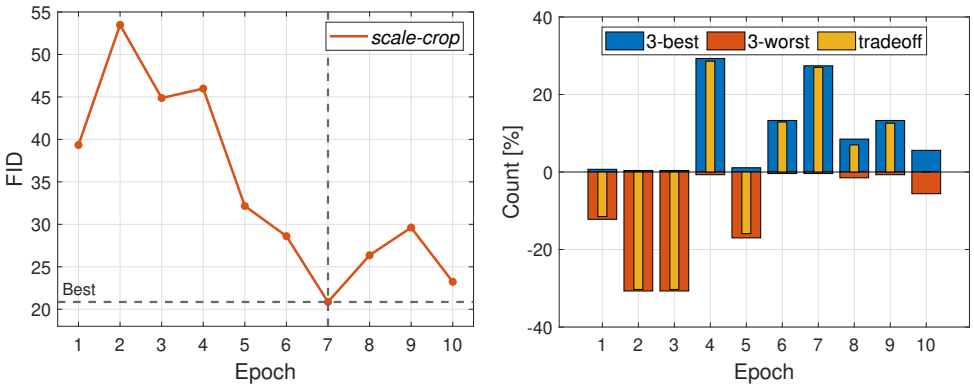


Figure 5.4: The best-trained Monodepth2 models. Row-wisely, (i) original image, (ii) KITTI pretrained (MS, 640 × 192), (iii) fine-tuned (1024 × 576, frame step 2, epoch 5), (iv) fine-tuned (512 × 288, frame step 4, epoch 6), and (v) trained from scratch (512 × 288, frame step 6, epoch 14).

In Figure 5.4, we can see a preview of the best performing models in various complex scenes. The depth maps in the second row – corresponding to the KITTI pretrained model – are stable but blurry. Presumably, the consistency is due to the training with stereo images, while the blurriness is probably caused by the different aspect ratio and resolution of the images in BDD100K. On the contrary, the fine-tuned or trained-from-scratch models produce sharper but locally inconsistent predictions. From the video perspective, in some cases, the quality of depth estimation strongly oscillates.

A noticeable black area at the bottom of the images is an incorrectly estimated hood of the car. The network associates objects which move at the same velocity as the camera (e.g., the hood or parallel-driving car) with being infinitely far [47]. Nowadays, the temporal object deletion is the main problem of contemporary monocular depth estimation frameworks, and finding a method to fix it is a research topic on its own. To remark, this failure does not occur with the KITTI pretrained model due to the stereo training.

From the visual point of view, the fine-tuned (1024 × 576, frame step 2, epoch 5) model appears to be the best of all models (consult the images in the third row of Figure 5.4) and is used in our pipeline.



(a) : FID evaluation. **(b) :** User study.

Figure 5.5: Evaluation of fake nighttime images generated from intermediate images by CUT [17].

5.3.2 CUT

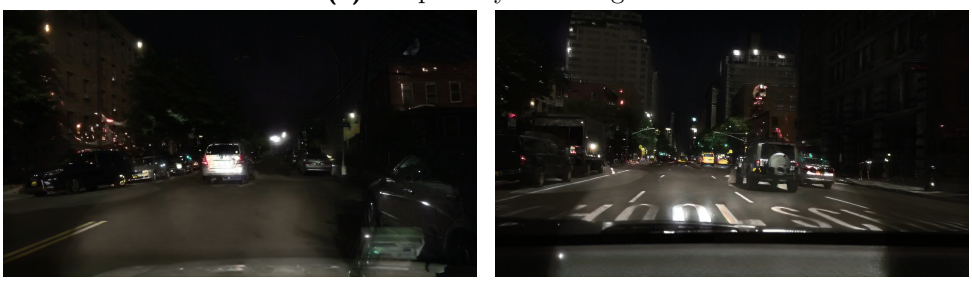
Inspired by the advantages and drawbacks of learning-based image translation techniques, we leverage the observations from Section 5.2.1 for the final step of our day-to-night translation pipeline. However, instead of day-to-night, the translation model is trained to perform the intermediate-to-night style transfer with a single fixed light setup – four random street lamps and ego-vehicle's headlights on (best viewed in Figure 5.7b). Considering the high computational demands of the training, we adopt only the highest-rated model, i.e., CUT [17] with the *scale-crop* setting. Compared to daytime, we assume that the distribution of intermediate images is closer to the target night domain. Hence, we find 10 epochs of training to be enough.

In Figure 5.5, there are results of both quality measures. The epoch with the lowest FID score, i.e., the seventh epoch, is also one of the highest positive peaks of the user study. Therefore, we take the model from the seventh epoch as the final one. Analogous to day-to-night models, we generate a complete dataset of artificial night images and use it for the object detector training in Chapter 6.

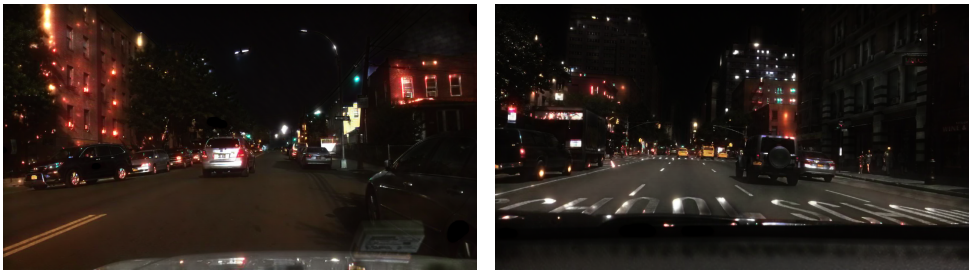
5.4 Qualitative Comparison and Failure Cases

At the end of this chapter, we present a comparison of the final nighttime images. City street and highway environments are shown in Figure 5.6 and Figure 5.7, respectively. Common translation artifacts and failures follow in Figure 5.8. The results are discussed in Chapter 7.

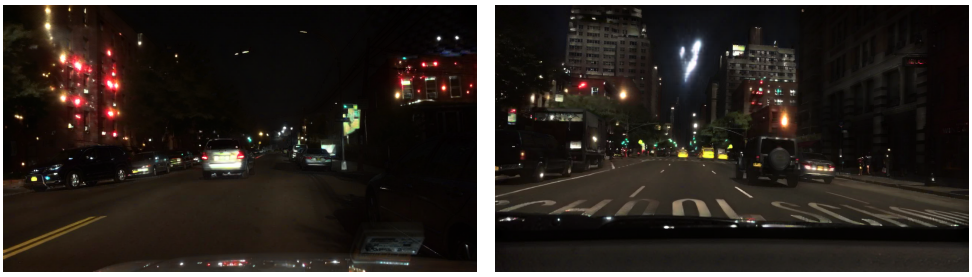
(a) : Input daytime image.



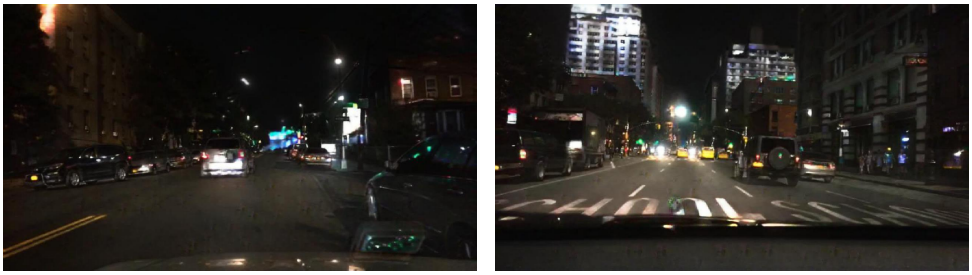
(b) : Our method (headlights on).



(c) : CycleGAN [10].



(d) : CUT [17].



(e) : ForkGAN [19].

Figure 5.6: Visual comparison of fake nighttime images (city street).

(a) : Input daytime image.

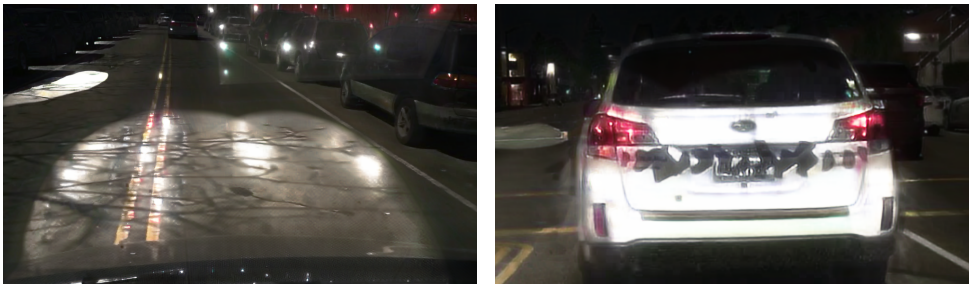
(b) : Our method (headlights on).

(c) : CycleGAN [10].

(d) : CUT [17].

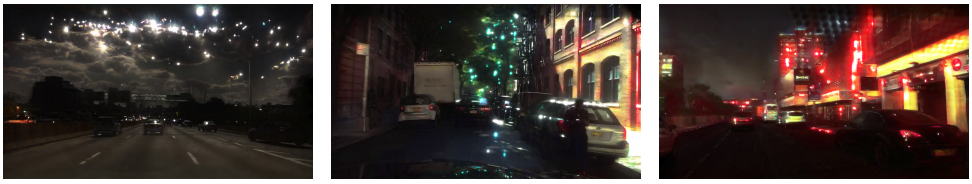
(e) : ForkGAN [19].

Figure 5.7: Visual comparison of fake nighttime images (highway).

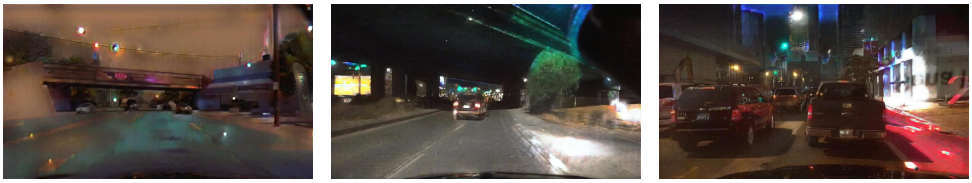


(a) : Our method.

(b) : CycleGAN [10].



(c) : CUT [17].



(d) : ForkGAN [19].

Figure 5.8: Common translation artifacts and failures.

Chapter 6

Experimental Results

Object detection is a thoroughly researched area of computer vision and is extensively used for benchmarking purposes. In this chapter, we train the object detector on several mixtures of real and fake (nighttime synthesized) images generated in Chapter 5.

The performance of trained models tested on real nighttime data is used as a quantitative quality measure for the generated nighttime images. Specifically, a mean average precision with 50 % threshold (mAP50) and COCO mean average precision (mAP@[50:5:95]) are computed. If you are aware of the concept of the mean average precision score, you can skip the following paragraph and continue with the description of the data flow diagram.

Performance Measures. Let us denote the ground truth bounding box by b_{gt} and the predicted bounding box by b_{pred}. To measure the prediction accuracy of b_{pred}, an intersection over union is computed

$$\text{IoU} = \frac{\text{area of } b_{\text{pred}} \cap b_{\text{gt}}}{\text{area of } b_{\text{pred}} \cup b_{\text{gt}}}. \tag{6.1}$$

We emphasize that the calculation of the bounding boxes overlap only makes sense if they belong to the same class.

If the intersection over union exceeds the specified threshold, we count b_{pred} as true positive (TP). Predicted bounding boxes that do not exceed the IoU threshold are counted as false positives (FP). Similarly, ground truth bounding boxes without an associated correct prediction are counted as false negatives (FN). With TP, FP, and FN, we can now express

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{6.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{6.3}$$

A standard requirement for the correct detection is a single IoU threshold of 50 %. Given the detection confidence, we follow [90] by constructing a precision-recall curve. The AP50 score is then computed as an area under this precision-recall curve[32]. An analogous but stricter variant (in terms of

[32] For further clarification, please refer to the paper [90].

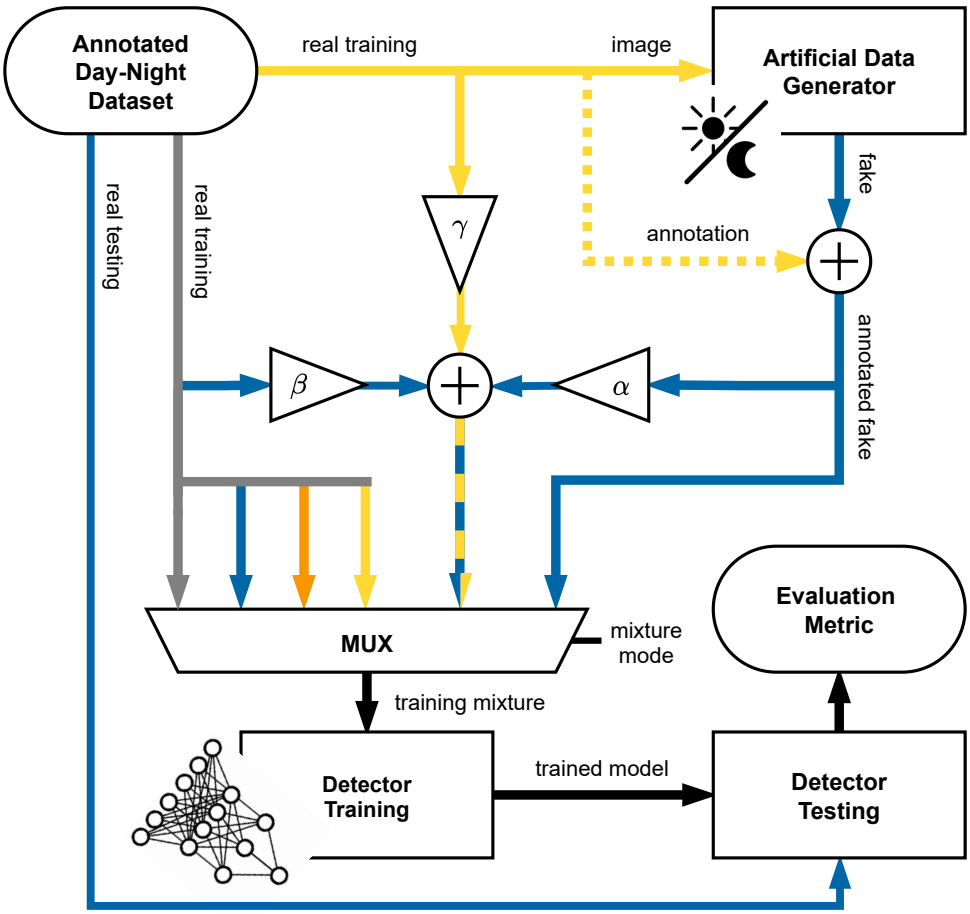


Figure 6.1: Data flow diagram of the object detection benchmark. The color of arrows in the upper part of the diagram encodes the time-of-day of data, i.e., daytime, daytime + dawn/dusk, nighttime, and all together.

b_{pred} localization) is the AP@[50:5:95] score[33] that is averaged across ten IoU thresholds between 50 % and 95 % (i.e., 5% step).

Finally, the average precision with 50 % threshold (AP50) and average precision (AP@[50:5:95]) are averaged over all classes, yielding the mean average precision with 50 % threshold (mAP50) and mean average precision (mAP@[50:5:95]), respectively. Conventionally, when there is no possibility of confusion, we shorten AP@[50:5:95] and mAP@[50:5:95] as AP and mAP, respectively.

Description of the Data Flow Diagram. For clarity, we summarize our experimental setup in an easy-to-follow diagram (see Figure 6.1).

In the upper left corner, there is a source of annotated day-night images. In our case, this is the BDD100K [1] dataset. The upper right *Artificial Data Generator* block – the core of our research – performs day-to-night image

[33] Motivated by the COCO competition at https://cocodataset.org.

style transfer. It can be either our method proposed, CycleGAN [10], CUT [17], or ForkGAN [19].

Image annotations are not required by any translation method listed above and are illustratively detached from the data stream. Since we keep the semantic segmentation of the scene valid after the style transformation, we can reuse the original daytime annotations and assign them to the corresponding fake nighttime counterparts.

Besides the real-only and fake-only branches, we also mix some real and fake data with each other. The contribution ratio of individual branches is controlled by the blocks α, β, and γ (the respective subsets of images are chosen randomly). The multiplexer component serves as a data selector. One branch of data is selected for training the detector.

Lastly, we evaluate the trained model using the annotated real nighttime testing data, i.e., we calculate the mAP50 and mAP metrics.

6.1 Object Detection Framework

Despite more recent – usually speed-oriented – promising approaches, such as SSD [91], RetinaNet [92], CSP [93], or the later models of YOLO series (e.g., YOLOv3 [94], YOLOv4 [95]), the Faster R-CNN [101] framework (as a speed-accuracy tradeoff) still remains the gold standard in object detection benchmarking. Of the related works listed in Chapter 2, it has been used, for instance, in AugGAN [18], ForkGAN [19], or Arruda [42]. That encourages us to train the Faster R-CNN detector for our purposes too.

The Faster R-CNN implementation is available within most contemporary codebases, e.g., Detectron2 [96], MMDetection [97], or TensorFlow Object Detection API [98]. Let us choose the lightweight and high-efficient MMDetection platform.

We initialize the network with the ResNet-50 [99] backbone pretrained[34] on the MS COCO dataset [100]. We adopt the SGD optimizer with momentum 0.9. Further, we set the batch size to 6, learning rate to 0.0025, and weight decay to 0.0001. Other parameters are left default. Empirically, we found that the optimum is reached within 16 epochs[35].

We choose the best epoch according to the mean average precision (mAP) of the validation set. The same time-of-day is chosen for validation as for training. In other words, we optimize the detector for the same time-of-day domain from which the training data comes (e.g., when training on daytime, the validation is performed on the daytime data as well). If not explicitly stated otherwise, the best-selected models are tested exclusively on night testing data.

Although every training requires plenty of computing resources and time, we repeat each training procedure five times to ensure proper evaluation and lessen the effect of randomness and "lucky situations". Thereby, we get five

[34] The weights are available from https://github.com/open-mmlab/mmdetection.

[35] We trained several models for 30 epochs but experienced overfitting to training data.

different scores for each data mixture. Next, we compute the average and standard deviation for each five-tuple. This good practice can also be seen in the related study [42].

A great amount of time and effort was spent on detector training. The training on a single NVIDIA GeForce GTX 1080 Ti GPU takes, on average, about half an hour per 10 k images per epoch. For example, a complete (5×16)-epochs training with 26 k daytime images takes over 100 hours. The following sections comprise a total of 130 trained models. The overall GPU-time consumption exceeded 4200 hours[36].

6.2 Object Detection Baselines

In order to have a baseline, we train the detector on the original real data for various times of day, i.e., daytime, daytime + dawn/dusk (non-night), night, day + night, and all together. By the original real data, we mean the split after the correction of time-of-day misclassifications in Section 4.2.3 (for the comparison with the uncorrected data, see Table 6.1). In the diagram in Figure 6.1, these baseline experiments are depicted by the first four branches from the left. The results are presented in Table 6.2.

In general, the models are evaluated on night testing data. We make one exception to this rule. The model corresponding to the last row of Table 6.2 is trained on all training data and tested on all testing data. If we look at the same model a line above, there is a large performance downgrade when the testing set is reduced to the challenging nighttime.

Mixture Mode	mAP50 [%]	mAP [%]	Car [%]	Person [%]	Bus [%]	Truck [%]	Bicycle [%]	M-cycle [%]
Day (errors)	47.3 ± .8	25.6 ± .2	37.0 ± .4	26.8 ± .4	29.5 ± 2.0	29.6 ± .6	19.0 ± .0	11.3 ± .6
Day (fixed)	44.5 ± 1.0	23.7 ± .6	35.1 ± .3	25.8 ± .4	27.3 ± 2.0	28.7 ± .9	16.5 ± .6	9.2 ± .6
Non-night (errors)	49.1 ± 1.3	26.5 ± .6	38.3 ± .1	27.4 ± .3	32.3 ± .6	31.9 ± .9	17.7 ± .7	11.4 ± 1.1
Non-night (fixed)	48.1 ± .6	26.1 ± .3	37.6 ± .3	27.0 ± .7	31.8 ± .8	31.4 ± .9	18.7 ± .5	9.9 ± .5
Night (errors)	52.7 ± .6	28.5 ± .1	41.9 ± .1	28.2 ± .1	37.6 ± .3	34.0 ± .1	18.3 ± .4	10.9 ± .4
Night (fixed)	53.0 ± .7	28.5 ± .4	42.2 ± .7	28.0 ± .6	36.3 ± 1.6	34.6 ± .6	18.0 ± 1.0	12.0 ± .7

Table 6.1: Performance impact of the correction of time-of-day misclassifications in BDD100K [1].

Mixture Mode	# Images	mAP50 [%]	mAP [%]	Car [%]	Person [%]	Bus [%]	Truck [%]	Bicycle [%]	M-cycle [%]
Day	26 k	44.5 ± 1.0	23.7 ± .6	35.1 ± .3	25.8 ± .4	27.3 ± 2.0	28.7 ± .9	16.5 ± .6	9.2 ± .6
Non-night	30 k	48.1 ± .6	26.1 ± .3	37.6 ± .3	27.0 ± .7	31.8 ± .8	31.4 ± .9	18.7 ± .5	9.9 ± .5
Night	20 k	53.0 ± .7	28.5 ± .4	42.2 ± .7	28.0 ± .6	36.3 ± 1.6	34.6 ± .6	18.0 ± 1.0	12.0 ± .7
Day + Night	46 k	56.1 ± .2	30.9 ± .2	42.5 ± .2	29.1 ± .4	39.9 ± 1.7	39.0 ± .4	20.9 ± 1.0	13.8 ± 1.1
All	50 k	56.1 ± .5	31.1 ± .6	43.1 ± .4	29.7 ± .6	39.9 ± .7	39.3 ± 1.0	21.4 ± .9	13.4 ± 1.5
All (tested all)	50 k	61.9 ± .3	35.8 ± .3	48.1 ± .2	33.7 ± .7	43.6 ± .4	41.1 ± .7	26.6 ± .5	22.0 ± .6

Table 6.2: Object detection baseline (trained on real data only).

[36] Counting only the training phases of the presented experiments.

6.3 Object Detection with Data Augmentation

In Table 6.3, there are the results of the object detector trained on several mixtures of real and fake images. For the reader's convenience, we reiterate the relevant baseline in the first row of each subtable. The fake nighttime images are always translated from the real daytime images. With this in mind, the corresponding baseline for the fake night is the real day. The training mixtures are illustrated in Figure 6.1 by the two lines on the right of the multiplexer input. The rightmost blue line, i.e., fake night images only, corresponds to the upper subtable. The training mixtures with a nontrivial real data component in the other three subtables share the yellow-blue dashed line and are specified by the associated triplet (α, β, γ).

Fake Night (26 k images)								$(\alpha = 1, \beta = 0, \gamma = 0)$[37]
Training Data	mAP50 [%]	mAP [%]	Car [%]	Person [%]	Bus [%]	Truck [%]	Bicycle [%]	M-cycle [%]
Day (real)	44.5 ± 1.0	$23.7 \pm .6$	$35.1 \pm .3$	$25.8 \pm .4$	27.3 ± 2.0	$28.7 \pm .9$	$16.5 \pm .6$	$9.2 \pm .6$
Our Method	40.2 ± 1.3	$21.3 \pm .6$	$32.2 \pm .7$	$23.2 \pm .5$	26.0 ± 1.4	25.1 ± 1.3	$16.7 \pm .8$	$4.7 \pm .9$
CycleGAN [10]	$39.1 \pm .5$	$21.1 \pm .3$	$31.7 \pm .7$	$24.3 \pm .5$	$26.7 \pm .9$	24.0 ± 1.3	$16.9 \pm .3$	$2.8 \pm .6$
CUT [17]	39.6 ± 1.2	$21.2 \pm .6$	$31.7 \pm .7$	$23.0 \pm .4$	27.3 ± 2.0	23.8 ± 1.9	$16.4 \pm .7$	5.0 ± 1.4
ForkGAN [19]	$47.5 \pm .6$	$25.5 \pm .4$	$37.3 \pm .4$	$25.5 \pm .9$	34.3 ± 1.3	30.2 ± 1.8	17.8 ± 1.1	$7.5 \pm .6$
Real Day + Fake Night (26 k images)								$(\alpha, \beta = 0, \gamma = 1 - \alpha)$
α/γ Ratio [%]	mAP50 [%]	mAP [%]	Car [%]	Person [%]	Bus [%]	Truck [%]	Bicycle [%]	M-cycle [%]
Day (real)	44.5 ± 1.0	$23.7 \pm .6$	$35.1 \pm .3$	$25.8 \pm .4$	27.3 ± 2.0	$28.7 \pm .9$	$16.5 \pm .6$	$9.2 \pm .6$
25/75	41.5 ± 1.1	$22.3 \pm .4$	$33.2 \pm .5$	$24.8 \pm .4$	$26.8 \pm .8$	27.2 ± 2.1	$16.1 \pm .3$	5.6 ± 1.0
50/50	42.3 ± 1.2	$22.7 \pm .6$	$32.7 \pm .7$	$24.4 \pm .4$	$28.7 \pm .8$	27.9 ± 1.2	$17.1 \pm .6$	$5.6 \pm .7$
75/25	41.6 ± 1.3	$22.4 \pm .6$	$33.2 \pm .5$	$24.1 \pm .7$	28.2 ± 1.2	27.5 ± 1.4	16.3 ± 1.6	$5.3 \pm .6$
Night (fake)	40.2 ± 1.3	$21.3 \pm .6$	$32.2 \pm .7$	$23.2 \pm .5$	26.0 ± 1.4	25.1 ± 1.3	$16.7 \pm .8$	$4.7 \pm .9$
Real Day + Fake Night (52 k images[38])								$(\alpha = 1, \beta = 0, \gamma = 1)$
Training Data	mAP50 [%]	mAP [%]	Car [%]	Person [%]	Bus [%]	Truck [%]	Bicycle [%]	M-cycle [%]
Day (real)	44.5 ± 1.0	$23.7 \pm .6$	$35.1 \pm .3$	$25.8 \pm .4$	27.3 ± 2.0	$28.7 \pm .9$	$16.5 \pm .6$	$9.2 \pm .6$
Our Method	$44.8 \pm .8$	$24.2 \pm .3$	$34.7 \pm .7$	$24.9 \pm .6$	$30.4 \pm .9$	29.8 ± 1.0	$18.1 \pm .8$	7.2 ± 1.9
CycleGAN [10]	$43.4 \pm .8$	$23.5 \pm .4$	$34.2 \pm .7$	$25.5 \pm .3$	$28.4 \pm .9$	$27.7 \pm .4$	$18.4 \pm .9$	6.6 ± 1.1
CUT [17]	$42.6 \pm .9$	$23.2 \pm .4$	$33.4 \pm .5$	$24.5 \pm .2$	29.1 ± 1.4	$28.4 \pm .7$	$18.2 \pm .6$	5.5 ± 1.0
ForkGAN [19]	$49.3 \pm .6$	$26.7 \pm .4$	$38.5 \pm .3$	$26.5 \pm .5$	$35.6 \pm .6$	32.5 ± 1.5	18.8 ± 1.1	8.6 ± 1.6
Real Night + Fake Night (46 k images)								$(\alpha = 1, \beta = 1, \gamma = 0)$
Training Data	mAP50 [%]	mAP [%]	Car [%]	Person [%]	Bus [%]	Truck [%]	Bicycle [%]	M-cycle [%]
Day + Night (real)	$56.1 \pm .2$	$30.9 \pm .2$	$42.5 \pm .2$	$29.1 \pm .4$	39.9 ± 1.7	$39.0 \pm .4$	20.9 ± 1.0	13.8 ± 1.1
Our Method	$56.2 \pm .2$	$31.0 \pm .3$	$43.0 \pm .3$	$28.8 \pm .4$	$41.2 \pm .4$	$39.3 \pm .5$	$21.2 \pm .6$	12.8 ± 1.1
CycleGAN [10]	$56.1 \pm .3$	$30.9 \pm .3$	$42.6 \pm .2$	$29.0 \pm .6$	$40.2 \pm .9$	$38.6 \pm .5$	$21.7 \pm .7$	$13.3 \pm .3$
CUT [17]	$56.0 \pm .3$	$31.0 \pm .3$	$42.8 \pm .3$	$29.1 \pm .2$	40.3 ± 1.0	$38.9 \pm .4$	21.4 ± 1.0	$13.6 \pm .9$
ForkGAN [19]	55.1 ± 1.0	$30.3 \pm .5$	$42.8 \pm .1$	$28.6 \pm .5$	40.1 ± 1.3	37.9 ± 1.0	20.2 ± 1.0	$12.4 \pm .8$

Table 6.3: Object detection performance for various training mixtures.

[37] An alternative way of specifying the fake nighttime stream.
[38] To remark, there are just 26 k unique base images.

Chapter 7

Discussion and Conclusions

7.1 Discussion

Quality of Synthesized Images. As shown in Figures 5.6b and 5.7b, our method can synthesize convincing nighttime images with high visual fidelity[39].

The designed lights are sharp and bright. That was our intention, and it demonstrates how the light reacts to the 3D geometry of the scene. However, in spite of the overwhelming visual impression, we probably generated images that have a different distribution than the majority of images in the nighttime testing set. According to the FID score, our generated images are about 84 % and 40 % farther from the target distribution than the images generated by CycleGAN [10] and CUT [17], respectively. We suppose that it is caused by the aforementioned sharp and bright lights. On the contrary, CycleGAN and CUT produce a large number of small lights (see respective images in Figures 5.6 and 5.7).

The most significant visual shortcomings of our method lie in two aspects.

- First, the shadow residuals – most notably on the road (see the left image in Figure 5.8a). We use the original daylight to create the intermediate image, and this approach does not take shadows into account.
- Second, the unenlightened black segments as a consequence of the inaccurate mesh reconstruction (see the right image in Figure 5.8a). This is a common artifact also in similar geometry-aware studies, e.g., [33].

In exceptional cases, the lights generated by CycleGAN or CUT are unnaturally green or red (see the middle and right images in Figures 5.8b and 5.8c). One can conjecture that it is due to traffic lights in night images. An extreme case of this issue is a "polar light" effect (see the middle image in Figure 5.8b). There is a sunny translation and a spectacular anomaly – sparkling sky – in the left pictures in Figure 5.8c and Figure 5.8b, respectively.

Unlike our method, CycleGAN or CUT, the ForkGAN [19] network often generates images with strong artifacts (see Figure 5.8d).

[39] In the attachment, you can watch a video teaser of our method.

Experimental Results. In Chapter 6, we used the synthesized images to stimulate the object detector performance. It turned out that a proper data augmentation that helps improve the detector performance is quite a tricky task (even for state-of-the-art methods). Unfortunately, in most cases, it is still better to train the detector merely on the original images than with the contribution of synthesized images. Especially, the training on artificial night data on its own is detrimental.

In the case of ForkGAN, we can see vastly higher scores. Nevertheless, we have some doubts about the correctness of the experiments. The ForkGAN model was most likely trained on our testing data since we are using the split designed in Section 4.2.2, i.e., the network could have seen the style of our testing data during training. Therefore, we must take the ForkGAN experiments with caution and exclude ForkGAN from competitive methods.

Experimental results in Table 6.3 show that the advantage of data augmentation heavily depends on the specific mixture of real and fake data. Note, however, that the performance of our method equals or surpasses the state-of-the-art in all experiments. Although CycleGAN and CUT are based on different optimization strategies, the qualitative and quantitative results are consistently comparable.

In the first subtable of Table 6.3, there are results of the training on fake night data only, and the drop in performance of all methods compared to the baseline is over 4 % in mAP50 and over 2 % in mAP@[50:5:95].

In the second subtable, we can see that it is not that straightforward to mix the training data. While we got the best results for real-only and the worst for fake-only, the transition is not monotonous. Indeed, we got a higher precision for 50 % contribution of fake night images than for 25 % or 75 % contribution.

In the case of the original daytime and synthesized nighttime mixture of images in the third subtable, our method outperforms the daytime baseline (i.e., it is better to train on this mixture than on the daytime only) and both competitive state-of-the-art methods. In particular, the mAP50 score is higher by 0.3 %, 1.4 %, and 2.2 % and the mAP@[50:5:95] score by 0.5 %, 0.7 %, and 1.0 % compared to the daytime baseline, CycleGAN, and CUT, respectively. Moreover, the standard deviation decreased by 20 % in mAP50 and by 50 % in mAP@[50:5:95] with respect to the daytime baseline, implying a more robust model. These results reveal the potential power of our method.

The last subtable – corresponding to the mixture of original and synthesized nighttime images – shows an almost balanced performance of all training modes of the detector. Notwithstanding the favorable scores, we call the results into question since we can interpret it as data saturation with real nighttime images. An ablation study might be helpful. Surprisingly, in this experiment, ForkGAN is the worst of all methods.

The caveat here is the current lack of diverse illumination conditions from which the detector could profit. An investigation of the detector performance with various light setups is the subject of future work. In Appendix A, we propose a possible solution to reach this goal in a smart and automated way.

Correction of Time-of-Day Misclassifications. The question to be answered is how does the correction of time-of-day misclassifications (see Section 4.2.3) influence the object detection performance. Table 6.1 shows the results of both splits side-by-side (as usual, tested on nighttime testing data). We can observe that after the correction, the daytime performance decreased by 2.8 % in mAP50 and by 1.9 % in mAP@[50:5:95]. It seems reasonable since the testing data (i.e., nighttime images) are missing from the daytime training set. In the same spirit, we can see that the non-night (daytime + dawn/dusk) performance decreased by 1.0 % in mAP50 and by 0.4 % in mAP@[50:5:95]. On the other hand, the nighttime performance slightly increased by 0.3 % in mAP50 (the mAP@[50:5:95] score remained the same). This makes sense as well since there are only relevant data present in the nighttime training set (i.e., nighttime images only). A complementary interpretation of both performance differences is missing daytime data (much easier for the object detection task) in the nighttime testing set.

7.2 Conclusions

In this thesis, we introduced a novel day-to-night image translation method with 3D-aware light control. The method leverages classical rendering techniques as well as contemporary deep neural networks and produces compelling and photorealistic results.

This transformation technique allowed us to literally shed light on the real-world problem of object detection. We trained the detector (Faster R-CNN [101]) on several mixtures of real and fake (nighttime synthesized) images. In terms of mAP50 and mAP@[50:5:95], our method is on par or even outperforms the competitive state-of-the-art methods CycleGAN [10] and CUT [17] trained to perform day-to-night image style transfer. Furthermore, with a proper mixture of real and fake data, our method proposed boosts the detector performance.

Even though it is not yet uniformly beneficial, the proposed innovative solution has substantial potential. Our advantage over most previous works is that one given image can generate a plethora of images by changing the lighting conditions and thereby introducing a greater diversity in data.

A side contribution of this work is the correction of time-of-day misclassifications in the BDD100K dataset [1]. The dataset is still under development, and our observations were warmly appreciated by the community[40] and it will be incorporated into the future updates of the dataset.

To conclude, the objectives of this thesis were met in general, with minor concerns regarding the enhancement of the object detector performance that is to be investigated in future work.

[40] Special thanks go to Asst. Prof. Fisher Yu, Ph.D. from ETH Zürich (Switzerland) for his interest.

Bibliography

[1] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

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

[3] A. Waheed, M. Goyal, D. Gupta, A. Khanna, F. Al-Turjman, and P. R. Pinheiro, "CovidGAN: Data augmentation using auxiliary classifier GAN for improved Covid-19 detection," *IEEE Access*, vol. 8, p. 91916–91923, 2020.

[4] R. Sinha, M. Vatsa, and R. Singh, "FamilyGAN: Generating kin face images using generative adversarial networks," in *European Conference on Computer Vision (ECCV) Workshops*, 2020.

[5] B. Liu, C. Tan, S. Li, J. He, and H. Wang, "A data augmentation method based on generative adversarial networks for grape leaf disease identification," *IEEE Access*, vol. 8, pp. 102 188–102 198, 2020.

[6] B. Lei, Z. Xia, F. Jiang, X. Jiang, Z. Ge, Y. Xu, J. Qin, S. Chen, T. Wang, and S. Wang, "Skin lesion segmentation via generative adversarial networks with dual discriminators," *Medical Image Analysis*, vol. 64, p. 101716, 2020.

[7] B. Lütjens, B. Leshchinskiy, C. Requena-Mesa, F. Chishtie, N. D. Rodríguez, O. Boulais, A. Piña, D. Newman, A. Lavin, Y. Gal, and C. Raïssi, "Physics-informed GANs for coastal flood visualization," *Computing Research Repository (CoRR)*, vol. abs/2010.08103, 2020.

[8] N. Nauata, S. Hosseini, K. Chang, H. Chu, C. Cheng, and Y. Furukawa, "House-GAN++: Generative adversarial layout refinement networks," *Computing Research Repository (CoRR)*, vol. abs/2103.02574, 2021.

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

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

[11] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," *Computing Research Repository (CoRR)*, vol. abs/1703.00848, 2017.

[12] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," *Computing Research Repository (CoRR)*, vol. abs/1703.05192, 2017.

[13] Z. Yi, H. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised dual learning for image-to-image translation," *Computing Research Repository (CoRR)*, vol. abs/1704.02510, 2017.

[14] J. Hoffman, E. Tzeng, T. Park, J. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "CyCADA: Cycle-consistent adversarial domain adaptation," *Computing Research Repository (CoRR)*, vol. abs/1711.03213, 2017.

[15] A. Anoosheh, E. Agustsson, R. Timofte, and L. V. Gool, "ComboGAN: Unrestrained scalability for image domain translation," *Computing Research Repository (CoRR)*, vol. abs/1712.06909, 2017.

[16] W. Wu, K. Cao, C. Li, C. Qian, and C. C. Loy, "TransGaGa: Geometry-aware unsupervised image-to-image translation," *Computing Research Repository (CoRR)*, vol. abs/1904.09571, 2019.

[17] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *European Conference on Computer Vision (ECCV)*, 2020.

[18] C. T. Lin, S. W. Huang, Y. Y. Wu, and S. H. Lai, "GAN-based day-to-night image style transfer for nighttime vehicle detection," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 22, no. 2, pp. 951–963, 2021.

[19] Z. Zheng, Y. Wu, X. Han, and J. Shi, "ForkGAN: Seeing into the rainy night," in *European Conference on Computer Vision (ECCV)*, 2020.

[20] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[21] S. Y. Baek and S. Lee, "Day-to-night road scene image translation using semantic segmentation," in *Pacific Graphics Short Papers, Posters, and Work-in-Progress Papers*. The Eurographics Association, 2020.

[22] A. Anoosheh, T. Sattler, R. Timofte, M. Pollefeys, and L. V. Gool, "Night-to-day image translation for retrieval-based localization," *Computing Research Repository (CoRR)*, vol. abs/1809.09767, 2018.

[23] E. Romera, L. M. Bergasa, K. Yang, J. M. Alvarez, and R. Barea, "Bridging the day and night domain gap for semantic segmentation," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1312–1318.

[24] L. Sun, K. Wang, K. Yang, and K. Xiang, "See clearer at night: Towards robust nighttime semantic segmentation through day-night image conversion," *Computing Research Repository (CoRR)*, vol. abs/1908.05868, 2019.

[25] M. Schutera, M. Hussein, J. Abhau, R. Mikut, and M. Reischl, "Night-to-day: Online image-to-image translation for object detection within autonomous driving by night," *IEEE Transactions on Intelligent Vehicles (T-IV)*, pp. 1–1, 2020.

[26] R. Gong, W. Li, Y. Chen, and L. V. Gool, "DLOW: Domain flow for adaptation and generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[27] X. Wang, K. Yu, C. Dong, X. Tang, and C. C. Loy, "Deep network interpolation for continuous imagery effect transition," *Computing Research Repository (CoRR)*, vol. abs/1811.10515, 2018.

[28] A. Romero, P. Arbelaez, L. Van Gool, and R. Timofte, "SMIT: Stochastic multi-label image-to-image translation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2019.

[29] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "StarGAN v2: Diverse image synthesis for multiple domains," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[30] I. Anokhin, P. Solovev, D. Korzhenkov, A. Kharlamov, T. Khakhulin, A. Silvestrov, S. I. Nikolenko, V. Lempitsky, and G. Sterkin, "High-resolution daytime translation without domain labels," *Computing Research Repository (CoRR)*, vol. abs/2003.08791, 2020.

[31] Q. Mao, H. Lee, H. Tseng, J. Huang, S. Ma, and M. Yang, "Continuous and diverse image-to-image translation via signed attribute vectors," *Computing Research Repository (CoRR)*, vol. abs/2011.01215, 2020.

[32] F. Pizzati, P. Cerri, and R. de Charette, "CoMoGAN: Continuous model-guided image-to-image translation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[33] J. Philip, M. Gharbi, T. Zhou, A. Efros, and G. Drettakis, "Multi-view relighting using a geometry-aware network," *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, vol. 38, no. 4, 2019.

[34] A. Carlson, R. Vasudevan, and M. Johnson-Roberson, "Shadow transfer: Single image relighting for urban road scenes," *Computing Research Repository (CoRR)*, vol. abs/1909.10363, 2019.

[35] P. Gafton and E. Maraz, "2D image relighting with image-to-image translation," *Computing Research Repository (CoRR)*, vol. abs/2006.07816, 2020.

[36] L. Qu, J. Tian, S. He, Y. Tang, and R. W. H. Lau, "DeshadowNet: A multi-context embedding deep network for shadow removal," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[37] X. Hu, Y. Jiang, C.-W. Fu, and P.-A. Heng, "Mask-ShadowGAN: Learning to remove shadows from unpaired data," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[38] L. Zhang, C. Long, X. Zhang, and C. Xiao, "RIS-GAN: Explore residual and illumination with generative adversarial networks for shadow removal," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[39] F. Vasluianu, A. Romero, L. V. Gool, and R. Timofte, "Self-supervised shadow removal," *Computing Research Repository (CoRR)*, vol. abs/2010.11619, 2020.

[40] D. Sakkos, H. P. H. Shum, and E. S. L. Ho, "Illumination-based data augmentation for robust background subtraction," *13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, 2019.

[41] D. Sakkos, E. S. L. Ho, H. P. H. Shum, and G. Elvin, "Image editing-based data augmentation for illumination-insensitive background subtraction," *Journal of Enterprise Information Management (JEIM)*, 2020.

[42] V. F. Arruda, T. M. Paixao, R. F. Berriel, A. F. De Souza, C. Badue, N. Sebe, and T. Oliveira-Santos, "Cross-domain car detection using unsupervised image-to-image translation: From day to night," *International Joint Conference on Neural Networks (IJCNN)*, 2019.

[43] T. Liu, Z. Chen, Y. Yang, Z. Wu, and H. Li, "Lane detection in low-light conditions using an efficient data enhancement : Light conditions style transfer," in *IEEE Intelligent Vehicles Symposium (IV)*, 2020.

[44] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[45] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2002–2011.

[46] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[47] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth prediction," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[48] S. Pillai, R. Ambrus, and A. Gaidon, "SuperDepth: Self-supervised, super-resolved monocular depth estimation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[49] C. Zhao, Q. Sun, C. Zhang, Y. Tang, and F. Qian, "Monocular depth estimation based on deep learning: An overview," *Science China Technological Sciences*, vol. 63, no. 9, p. 1612–1627, 2020.

[50] R. I. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge University Press, 2004.

[51] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[52] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 5, no. 4, pp. 349–359, 1999.

[53] A. Muntoni and P. Cignoni, "PyMeshLab," Jan. 2021.

[54] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: An open-source mesh processing tool," in *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.

[55] Blender Online Community, *Blender - A 3D modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, 2021.

[56] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. PAMI-9, no. 4, pp. 532–550, 1987.

[57] R. C. Gonzalez and R. E. Woods, *Digital image processing*, 4th ed. Prentice Hall, 2018.

[58] *The OpenCV reference manual*, Itseez, April 2021.

[59] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[60] A. R. Smith, "Color gamut transform pairs," in *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '78. Association for Computing Machinery, 1978, p. 12–19.

[61] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[62] J. F. Nash, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[63] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," *Computing Research Repository (CoRR)*, vol. abs/2002.05709, 2020.

[64] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," *Computing Research Repository (CoRR)*, vol. abs/1903.11027, 2019.

[65] A. Patil, S. Malla, H. Gang, and Y.-T. Chen, "The H3D dataset for full-surround 3D multi-object detection and tracking in crowded urban scenes," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[66] B. Hurl, K. Czarnecki, and S. Waslander, "Precise synthetic image and LiDAR (PreSIL) dataset for autonomous vehicle perception," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2522–2529.

[67] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3D tracking and forecasting with rich maps," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8740–8749.

[68] Q.-H. Pham, P. Sevestre, R. S. Pahwa, H. Zhan, C. H. Pang, Y. Chen, A. Mustafa, V. Chandrasekhar, and J. Lin, "A*3D dataset: Towards autonomous driving in challenging environments," in *Proceedings of the IEEE International Conference in Robotics and Automation (ICRA)*, 2020.

[69] C. Sakaridis, D. Dai, and L. Van Gool, "Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[70] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo Open Dataset," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2443–2451.

[71] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft Level 5 perception dataset 2020," https://level5.lyft.com/dataset/, 2019.

[72] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth, "A2D2: Audi autonomous driving dataset," *Computing Research Repository (CoRR)*, vol. abs/2004.06320, 2020.

[73] Z. Zhang, T. Sattler, and D. Scaramuzza, "Reference pose generation for visual localization via learned features and view synthesis," *Computing Research Repository (CoRR)*, vol. abs/2005.05179, 2020.

[74] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[75] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[76] L. Neumann, M. Karg, S. Zhang, C. Scharfenberger, E. Piegert, S. Mistr, O. Prokofyeva, R. Thiel, A. Vedaldi, A. Zisserman, and B. Schiele, "NightOwls: A pedestrians at night dataset," in *Asian Conference on Computer Vision (ACCV)*. Springer, 2018, pp. 691–705.

[77] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1643–1649.

[78] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017.

[79] F. Tung, J. Chen, L. Meng, and J. J. Little, "The Raincouver scene parsing benchmark for self-driving in adverse weather and at night,"

IEEE Robotics and Automation Letters (RA-L), vol. 2, no. 4, pp. 2188–2193, 2017.

[80] D. Dai and L. Van Gool, "Dark model adaptation: Semantic image segmentation from daytime to nighttime," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[81] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The Mapillary Vistas Dataset for semantic understanding of street scenes," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[82] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The ApolloScape open dataset for autonomous driving and its application," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[83] S. Hwang, J. Park, N. Kim, Y. Choi, and I. S. Kweon, "Multispectral pedestrian detection: Benchmark dataset and baseline," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1037–1045.

[84] Y. Choi, N. Kim, S. Hwang, K. Park, J. S. Yoon, K. An, and I. S. Kweon, "KAIST multi-spectral day/night data set for autonomous and assisted driving," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 19, no. 3, pp. 934–948, 2018.

[85] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, ser. NIPS'17. Curran Associates Inc., 2017, p. 6629–6640.

[86] A. Borji, "Pros and cons of GAN evaluation measures," *Computing Research Repository (CoRR)*, vol. abs/1802.03446, 2018.

[87] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *Computing Research Repository (CoRR)*, vol. abs/1812.04948, 2018.

[88] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[89] M. Seitzer, "pytorch-fid: FID score for PyTorch," https://github.com/mseitzer/pytorch-fid, August 2020, version 0.1.1.

[90] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *International Journal of Computer Vision (IJCV)*, vol. 88, no. 2, pp. 303–338, 2010.

[91] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*. Springer International Publishing, 2016, pp. 21–37.

[92] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 42, no. 02, pp. 318–327, 2020.

[93] W. Liu, S. Liao, W. Ren, W. Hu, and Y. Yu, "High-level semantic feature detection: A new perspective for pedestrian detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5182–5191.

[94] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *Computing Research Repository (CoRR)*, vol. abs/1804.02767, 2018.

[95] A. Bochkovskiy, C. Wang, and H. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *Computing Research Repository (CoRR)*, vol. abs/2004.10934, 2020.

[96] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[97] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: OpenMMLab detection toolbox and benchmark," *Computing Research Repository (CoRR)*, vol. abs/1906.07155, 2019.

[98] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *Computing Research Repository (CoRR)*, vol. abs/1611.10012, 2016.

[99] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[100] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *Computing Research Repository (CoRR)*, vol. abs/1405.0312, 2014.

[101] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 28. Curran Associates, Inc., 2015.

[102] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European Conference on Computer Vision (ECCV)*. Springer International Publishing, 2018, pp. 833–851.

[103] MMSegmentation Contributors, "MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark," https://github.com/open-mmlab/mmsegmentation, 2020.

[104] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *Computing Research Repository (CoRR)*, vol. abs/1703.06870, 2017.

Appendix A

Detection of Light Sources for Smarter Light Control

The light setup is a crucial prerequisite for our method. Thus far, we have only relied on the ad hoc hand-operated design. Nevertheless, that is time-consuming, and it is not reliably applicable to each and every image of the entire dataset. For instance, almost every image in the BDD100K dataset [1] is captured from a different camera position, which involves plenty of nonstandard situations (see Figure 4.1).

It would be perfect if we could autonomously position the lights in a semantically oriented manner and turn on the relevant light sources in each image independently (regardless of the camera position). We shortly experimented in this direction and achieved some satisfactory results.

A.1 Headlights and Taillights Detection

Leveraging a recently published annotation tool Hasty.ai[41], we manually assigned more than 4200 headlights and taillights bounding boxes to 1000 training and 100 validation images. With the guidance of these annotations, we trained the Faster R-CNN [101] object detector with the same configuration as described in Section 6.1 and reached 69.2 % mAP50 and 35.0 % mAP@[50:5:95] on the validation images. By adding more annotated images, we would probably improve the detection performance even further.

In Figure A.1, we can see a few validation images with predicted bounding boxes of headlights and taillights with associated detection confidence. High detection performance and Figure A.1 show the feasibility of this approach.

The headlights and taillights annotations in the standard COCO [100] file format are available in the attachment of this thesis.

A.2 Semantic Segmentation of Street Lamps

We also took advantage of the semantically segmented subset (10 k) of the BDD100K dataset that contains annotations of street poles. We trained

[41] For more information, please visit https://hasty.ai.

Figure A.1: Validation images with predicted headlights and taillights.

the DeepLabv3+ [102] segmentation model for the purpose of street pole inference in the unannotated part of the dataset. For efficient implementation, we used the MMSegmentation platform [103]. We initialized the network with the ResNet-50 [99] backbone pretrained[42] on the Cityscapes dataset [74]. We adopted the SGD optimizer with momentum 0.9. Further, we set the batch size to 2, learning rate to 0.0025, and weight decay to 0.0005. Other parameters were left default. We let the network train for 40 epochs. Measured on the validation set, in epoch 24, we achieved the highest performance of 60.6 % mAP50 and 51.9 % mAP@[50:5:95].

Since April 2021, semantic instance segmentation is also available for BDD10K. In addition to semantic segmentation, it includes detailed street lamp annotations. This suits our intention (not every pole has to be a lamp), and we are interested in training the instance segmentation model in a similar way as for semantic segmentation of the poles. For instance segmentation, we suggest using the Mask R-CNN framework [104] – available under the MMDetection codebase [97].

[42] The weights are available from https://github.com/open-mmlab/mmsegmentation.

A.3 The Most Exciting Future Avenue

In future work, we would like to incorporate the headlights and taillights detection and street lamp segmentation outcomes into our day-to-night translation pipeline for smarter light control. We believe that it will conveniently complement manually defined lighting, provide a fantastic visual experience, and allow for a greater variety of training data.

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Čech Josef** Personal ID number: **466176**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Day to night image style transfer with light control

Master's thesis title in Czech:

Změna stylu obrazových dat z denních na noční s kontrolou osvětlení

Guidelines:

Deep learning techniques have enabled the emergence of several state-of-the-art models to address problems in different domains, such as image classification, regression, object detection and semantic segmentation [6, 7]. However, these techniques are data-driven. Capturing enough variety in night scenes is very challenging because the amount of pedestrians and other classes instances is significantly reduced. In this work the student shall focus on translating images from day as the source domain into night as the target domain in driving scenarios with light control.
For the purposes of this work the following requirements need to be kept:
a) Keep the semantic segmentation of the scene valid after the domain transfer.
b) Do not significantly change the texture of the objects, ideally change just the scene lighting.
Keeping in mind the conditions a) and b), the thesis goals are:
1. Study the state of the art in image-to-image transfer, especially in the day-to-night domain.
2. Choose a suitable semantic segmentation and/or object detection model with training codes available. This model will be trained on several mixtures of real and fake (night time synthesized) images with the original scene annotations. The performance of trained models tested on real night time data will be used as the quality measure for generated training images.
3. Design an image-to-image (day-to-night) transfer method that would allow one to control the amount, position and direction of light sources which will influence the appearance of the synthesized night scene.
4. Take at least two state-of-the-art methods that perform day-to-night image transfer and perform the day-to-night transfer on training data. Compare the quality of images generated by these methods with the new design from point 3 using the metric suggested in point 2.
5. Find suitable datasets that contain both day-time and night-time images in driving scenarios with annotated objects for training of the image-to-image transfer model and the semantic segmentation and/or object detection model for evaluation.
In order to defend this thesis, the novel image-to-image transfer method does not necessarily have to beat the state-of-the-art methods. The proper experiments design and comparison methodology is what matters.

Bibliography / sources:

[1] GAN-Based Day-to-Night Image Style Transfer for Nighttime Vehicle Detection
Che-Tsung Lin, Sheng-Wei Huang, Yen-Yi Wu, Shang-Hong Lai
IEEE Transactions on Intelligent Transportation Systems, 2020
[2] Cross-Domain Car Detection Using Unsupervised Image-to-Image Translation: From Day to Night
Vinicius F. Arruda, Thiago M. Paixao, Rodrigo Ferreira Berriel, Alberto F. De Souza, Claudine Badue, Nicu Sebe, Thiago Oliveira-Santos
International Joint Conference on Neural Networks, 2019
[3] Lane Detection in Low-light Conditions Using an Efficient Data Enhancement: Light Conditions Style Transfer
Tong Liu, Zhaowei Chen, Yi Yang, Zehao Wu, Haowei Li
Computing Research Repository, 2020
[4] ForkGAN: Seeing into the Rainy Night
Ziqiang Zheng, Yang Wu,, Xinran Han and Jianbo Shi
IEEE European Conference on Computer Vision, 2020
[5] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros
IEEE International Conference on Computer Vision, 2017
[6] Mask R-CNN
Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross B. Girshick
IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020
[7] High-Level Semantic Feature Detection: New Perspective for Pedestrian Detection
Wei Liu,Shengcai Liao, Weiqiang Ren, Weidong Hu, Yinan Yu
IEEE Computer Vision and Pattern Recognition, 2019
[8] Multi-view Relighting using a Geometry-Aware Network

Name and workplace of master's thesis supervisor:

Ing. David Hurych, Ph.D., VALEO, Sazečská 247/2, 108 00 Praha

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **28.01.2021** Deadline for master's thesis submission: **21.05.2021**

Assignment valid until:
by the end of summer semester 2021/2022

Ing. David Hurych, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

