Unpaired Learning of Dense Visual Depth Estimators for Urban Environments

Vitor Guizilini and Fabio Ramos
School of Information Technologies
University of Sydney, Australia
{vitor.guizilini;fabio.ramos}@sydney.edu.au

Abstract: This paper addresses the classical problem of learning-based monocular depth estimation in urban environments, in which a model is trained to directly map a single input image to its corresponding depth values. All currently available techniques treat monocular depth estimation as a regression problem, and thus require some sort of data pairing, either explicitly as input-output ground-truth pairs, using information from range sensors (i.e. laser), or as binocular stereo footage. We introduce a novel methodology that completely eliminates the need for data pairing, only requiring two unrelated datasets containing samples of input images and output depth values. A cycle-consistent generative adversarial network is used to learn a mapping between these two domains, based on a custom adversarial loss function specifically designed to improve performance on the task of monocular depth estimation, including local depth smoothness and boundary equilibrium. A wide range of experiments were conducted using a variety of well-known indoor and outdoor datasets, with depth estimates obtained from laser sensors, RGBD cameras and SLAM pointclouds. In all of them, the proposed CycleDepth framework reaches competitive results even under a more restricted training scenario.

Keywords: Depth Estimation, Generative Adversarial Networks, Unpaired Learning, Monocular Cameras

1 Introduction

Visual sensors have a series of properties that make them especially attractive for data collection in robotics applications: they are relatively cheap, compact, with low power consumption and produce dense representations that include color information. However, differently from traditional range-based sensors such as lasers or sonars, cameras are unable to directly provide depth estimates from a single image, which is necessary for tasks such as mapping and 3D reconstruction. Discerning the shape of unstructured scenes from a single image is a key problem in perception, and historically this limitation has been addressed by relying on multiple viewpoints of the same scene, either with stereo configurations [1], moving cameras [2] or changing light conditions [3].

Recently, there has been a surge in works that pose the task of monocular depth estimation as a learning problem [4, 5], trying to predict the depth of each pixel using models that have been trained using a large number of data samples. Furthermore, new developments in back-propagation algorithms and parallel computing have made multi-layered neural networks (i.e. Deep Learning) the *de facto* tool for such applications, achieving unprecedented results particularly in autonomous navigation tasks, for vehicle localization and mapping. These works can be broadly divided into two categories: supervised, in which ground-truth is available as data collected from a different range-based sensor and a direct mapping between input and output is learned [6, 7]; or unsupervised, in which multiple overlapping views are available, either from stereo or moving cameras, and the latent transformation into disparity values between frames is learned [8, 9].

In this paper we propose a third category: unpaired, which does not rely on data labeling or synchronization of any sort. Supervised methods require input-output pairs, with explicit ground-truth information, and unsupervised methods require input stereo pairs, which contain implicit latent ground-truth information. Our method, on the other hand, only requires two separate datasets containing samples from the input and output domains, and it uses a cycle-consistent generative adversarial network [10] to train a model capable of mapping between these domains. The result is an end-to-end monocular depth estimation pipeline that can be trained with input and output data from different sources (i.e. from different runs, environments or even vehicles), while still producing results that are competitive with current state-of-the-art supervised and unsupervised techniques. Interestingly, as a "side effect", input depth estimates can also be mapped directly into images, producing an artificial textured reconstruction of range information.

To the best of our knowledge, the only other work to use unpaired learning for depth estimation, without relying on RGB-D inputs [11] or intermediate steps [12], is from Fish et al. [13], based on Adversarial Inverse Graphics Networks, however they still require stereo pairs to estimate ego-motion and optical flow between frames. Our CycleDepth framework does not require pairings of any sort, and thus is more suitable for tasks in which input and output samples cannot be obtained simultaneously during the training process (i.e data from mobile devices, real and synthetic domain transfer, leveraging unlabeled data, and so forth). The main contributions of this paper are as follows:

- **A novel architecture for monocular depth estimation** that does not require input-output or stereo pairs, only unrelated samples of each domain.
- **The introduction of cycle-consistent boundary-equilibrium losses** and how they can be used to achieve better unpaired learning without mode collapse.
- **A novel technique for sparse pointcloud 2D projection** that uses the Hilbert Maps framework [14] to produce a denser depth representation of observed structures.

2 Preliminaries

We start this section by discussing the concept of unpaired learning, and how it relates to the more traditional supervised and unsupervised training methodologies. Afterwards, we briefly describe Generative Adversarial Networks and their use in domain transfer tasks, including the mapping between color and depth images. Finally, we provide an overview on Hilbert Maps for occupancy modeling, that are used to increase the density of depth estimates obtained from pointcloud data.

2.1 Unpaired Learning

Broadly speaking, learning tasks are traditionally split into two categories: *supervised*, in which for each input there is a corresponding output, used as ground-truth to guide model convergence; and *unsupervised*, in which there is no ground-truth, and the model has to learn an underlying model that produces the desired results. For the particular task of depth estimation, supervised learning assumes training pairs of color and depth images, that are used to produce a model that maps each pixel of a given test input image into its corresponding depth estimate [7]. This includes the use of Markov Random Fields (MRF), Conditional Random Fields (CRF), semantic information and scene priors [15]. Unsupervised learning, on the other hand, relies on secondary pairing, such as overlapping stereo information [9] or sequential frames [16], and depth estimation becomes a by-product of the reconstructive model trained between these two pieces of information.

Recently, a novel method for domain transfer that does not require paired samples (i.e. *unpaired* learning) has been proposed by Xie et al. [10]. Instead of exploiting learning on the level of paired samples, it works on the level of sets: given two domains X and Y, each defined by its own image set, it trains a mapping $F : X \rightarrow Y$ such that $\hat{\mathbf{y}} = F(\mathbf{x} \in X)$ is indistinguishable from $\mathbf{y} \in Y$, given a model trained to discriminate $\hat{\mathbf{y}}$ from $\mathbf{y}$. Because this is under-constrained (there are infinitely many mappings F that will produce the same $\hat{\mathbf{y}}$ distribution), cycle consistency is enforced by introducing another mapping $G : Y \rightarrow X$ which should be the inverse of F, so that $G(F(\mathbf{x})) \approx \mathbf{x}$ and $F(G(\mathbf{y})) \approx \mathbf{y}$. Although the concept of unpaired learning has already been explored [17, 18], the formulation in [10] is not task-specific, and thus can be applied to a wide variety of domain transfer scenarios with minimal modifications. A similar framework is proposed in [19], using two sets of encoder-decoder networks and enforcing that both domains share the same latent feature space for low-dimensional projection.

2.1.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [20] are a class of unsupervised algorithms implemented by two neural networks competing against each other in a zero-sum game. One is the *generator*, that creates samples supposed to come from the same underlying distribution p_{data} as training data, while the *discriminator* examines samples to determine if they are real or fake. Both networks are continuously optimized, so as the generator learns how to create fake samples similar to real ones, the discriminator learns how to better distinguish between them. Formally speaking, GANs are a structured probabilistic model [21] containing observed variables $\mathbf{x} \sim p_{data}$ and latent variables $\mathbf{z}$. The discriminator is a function $y = D(\mathbf{x}, \boldsymbol{\theta}^D)$, and the generator is a function $\mathbf{x}' = G(\mathbf{z}, \boldsymbol{\theta}^G)$, where y is the probability of $\mathbf{x}$ being a real sample, $\mathbf{x}'$ is a fake generated sample and $(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G)$ are the parameter sets of each network. These parameter sets are optimized by minimizing two distinct loss functions, with the discriminator aiming to minimize $\mathcal{L}^D(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G)$ controlling only $\boldsymbol{\theta}^D$, while the generator aims to minimize $\mathcal{L}^G(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G)$ controlling only $\boldsymbol{\theta}^G$. Because each network's loss function depends on both parameter sets, the solution is a Nash equilibrium defined by a tuple $(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G)$ that is a local of $\mathcal{L}^D$ with respect to $\boldsymbol{\theta}^D$ and a local minimum of $\mathcal{L}^G$ with respect to $\boldsymbol{\theta}^G$.

2.1.2 Adversarial Loss Function

The choice of loss functions are key to a GAN's success, since they define how each network will be optimized during the training process to better achieve its goal. In the original implementation [20], the discriminator loss function is defined as (dependencies on $\boldsymbol{\theta}$ are removed for notation simplicity):

$$\mathcal{L}^D(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G) = -\frac{1}{2}\mathbb{E}_{\mathbf{x}} \log D(\mathbf{x}) - \frac{1}{2}\mathbb{E}_{\mathbf{z}} \log(1 - D(G(\mathbf{z}))), \tag{1}$$

which is the standard cross-entropy cost for a binary classified with sigmoid output, only trained on two data batches: one coming from the dataset (real, with label 1) and another coming from the generator (fake, with label 0). For the generator, a zero-sum solution would produce $\mathcal{L}^G = -\mathcal{L}^D$, which can be summarized as a value function $V(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G) = -\mathcal{L}^D(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G)$ specifying the discriminator's payoff. However, this approach does not work well in practice, since the discriminator aims to minimize the same cross-entropy that the generator is trying to maximize, which causes computational issues like vanishing gradients [22]. The solution is to, instead of flipping the discriminator sign, flip the target used to construct the cross-entropy cost, which then becomes:

$$\mathcal{L}^G(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G) = \frac{1}{2}\mathbb{E}_{\mathbf{z}} \log D(G(\mathbf{z})). \tag{2}$$

Training is performed in steps, by alternating between each loss function [23]. Since its introduction, a substantial amount of work has been done to improve GAN performance and address some of its original limitations [24, 25]. Several different convergence measures have also been proposed [25], to estimate the quality of generated samples during training.

2.2 Hilbert Maps for Occupancy Modeling

In [14] a novel framework for scene reconstruction was proposed, in which real-world complexity is represented linearly by projecting spatial coordinates into a high-dimensional feature vector. Under a particular set of assumptions [26], such as inner product representation, this structure is known as a *Hilbert space* and, furthermore, if point evaluation in this space is a continuous linear functional (i.e. if $||f - g||$ is small for functions f and g, then $|f(x) - g(x)|$ is also small for all x), then it becomes a *Reproducing Kernel Hilbert Space* (RKHS) [27].

We assume a training dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathcal{R}^D$ is a point in the D-dimensional space and $y_i = \{-1, +1\}$ is its corresponding occupancy state. This dataset is used to incrementally learn a discriminative model $p(y|\Phi(\mathbf{x}), \mathbf{w})$, parametrized by a weight vector $\mathbf{w}$ that predicts the occupancy values of new query points $\mathbf{x}_*$. It is known that, in high-dimensional spaces, linear separators are almost always adequate to separate classes [28]. Because of that, a simple Logistic Regression (LR) classifier [29] can be used, with the probability of non-occupancy for a query point given by:

$$p(y_* = -1|\Phi(\mathbf{x}_*), \mathbf{w}) = \frac{1}{1 + \exp\left(\mathbf{w}^T\Phi(\mathbf{x}_*)\right)}, \tag{3}$$

where $\Phi(\mathbf{x}_*)$ is a feature vector defined over $\mathbf{x}_*$, that projects input data into a RKHS for calculations. To optimize the weight parameters $\mathbf{w}$ based on information contained in $\mathcal{D}$, we minimize an objective function using the information contained in $\mathcal{D}$. This is usually done using stochastic gradient descent optimization [30], in which mini-batches of training data are used to take small steps towards a local minimum. This approach facilitates the use of large-scale datasets and lends itself naturally to online learning, where new data is constantly being produced and incorporated into the model.

3 Methodology

In this section the proposed CycleDepth framework is introduced and discussed in details. We start by describing how unpaired learning can be applied to the problem of monocular depth estimation, using cycle-consistent GANs to train mapping functions between color and depth domains. Afterwards, a novel adversarial loss function is proposed, that improves over [10] by including a boundary-equilibrium component and a series of constraints aimed specifically to improve performance for the task of monocular depth estimation. A novel methodology for pointcloud 2D projection is also proposed, to increase the density of depth image estimates during training.

3.1 CycleDepth Formulation

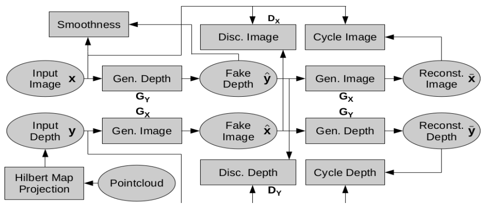
The objective is to learn mapping functions between two domains: X (image) and Y (depth), given unrelated training samples $\{\mathbf{x}_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in X$, and $\{\mathbf{y}_j\}_{j=1}^{M}$, where $\mathbf{y}_j \in Y$. Image samples are assumed to have dimensions $w \times h \times c$, with $c = 1$ if grayscale and $c = 3$ if colored, and depth samples are assumed to have dimensions $w \times h \times 1$. As depicted in Fig. 1a, this is achieved through the use of two generators $\hat{\mathbf{x}} = G_X(\mathbf{y})$ and $\hat{\mathbf{y}} = G_Y(\mathbf{x})$, each taking real inputs from one domain and mapping them into the other to produce fake estimates, and two discriminators D_X and D_Y, that aim to distinguish between $\{\mathbf{x}, \hat{\mathbf{x}}\}$ and $\{\mathbf{y}, \hat{\mathbf{y}}\}$ respectively (further details of each architecture are discussed in the supplementary material). Additionally, cycle consistency is achieved by applying each generator to its corresponding fake estimate, to create a reconstructed version of inputs $\bar{\mathbf{x}} = G_X(\hat{\mathbf{y}}) = G_X(G_Y(\mathbf{x}))$ and $\bar{\mathbf{y}} = G_Y(\hat{\mathbf{x}}) = G_Y(G_X(\mathbf{y}))$. During training, it is enforced that $\bar{\mathbf{x}}$ should approximate $\mathbf{x}$ and $\bar{\mathbf{y}}$ should approximate $\mathbf{y}$ according to a predefined distance metric, which prevents the learned mappings G_X and G_Y from contradicting each other.

3.2 Adversarial Depth Loss

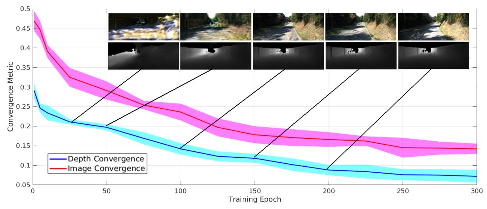
Adversarial losses [20] are used to train both sets of generators and discriminators, with G attempting to minimize an objective function, thus improving the quality of generated fake samples, while D tries to maximize it, to better distinguish between real and fake samples. In [10] the original adversarial loss formulation (Eqs. 1 and 2) is used, in conjunction with a cycle-consistent constraint to approximate inverse mappings. However, there are several well-documented shortcomings presented by this original formulation, such as mode collapse (only one convincing output is learned) [31] and balancing the convergence of generator and discriminator [22]. Because of that, here we explore an alternative technique for adversarial training, recently proposed in [25], and modify it to be used for domain transfer in a cycle-consistent scenario.

The Boundary-Equilibrium Generative Adversarial Network (BEGAN) technique uses an auto-encoder as discriminator [24], and instead of trying to match data distributions directly it attempts to match auto-encoder loss distributions, using a loss function derived from the Wasserstein distance [32]. It is shown that this can be achieved using a typical GAN objective function with an additional equilibrium term to balance generator and discriminator training. They are assumed to be at equilibrium when $\mathbb{E}\big[\mathcal{L}(\mathbf{x})\big] = \mathbb{E}\big[\mathcal{L}(\bar{\mathbf{x}})\big]$. This generator-discriminator equilibrium can be relaxed with the introduction of a new parameter $\gamma \in [0, 1]$, defined as $\gamma = \mathbb{E}[\mathcal{L}(\hat{\mathbf{x}}] \,/\, \mathbb{E}[\mathcal{L}(\mathbf{x})]$. This term lets us balance between different goals, with lower values of γ leading to lower image diversity, because the discriminator focuses more heavily on auto-encoding real images. The objective function for adversarial training is then defined as:

$$\mathcal{L}_{D_X} = \mathcal{L}(\mathbf{x}) - k_t \mathcal{L}(\hat{\mathbf{x}}) \;\text{ and }\; \mathcal{L}_{G_X} = \mathcal{L}(\hat{\mathbf{x}}) \;\text{, with }\; k_{t+1} = k_t + \lambda_k(\gamma \mathcal{L}(\mathbf{x}) - \mathcal{L}(\hat{\mathbf{x}})), \tag{4}$$



(a) Diagram of the proposed training pipeline for RGB and depth estimation.



(b) Convergence values $\mathcal{C}_X$ and $\mathcal{C}_Y$, including samples in different training stages.

Figure 1: The proposed CycleDepth framework.

where $k_t \in [0, 1]$ is a variable used to maintain equilibrium during training, and λ_k is the learning rate for k. To account for both sets of generators and discriminators in the proposed domain transfer framework, individual objective functions are summed up, so that $\mathcal{L}_D = \alpha\mathcal{L}_{D_X} + (1 - \alpha)\mathcal{L}_{D_Y}$ and $\mathcal{L}_G = \alpha\mathcal{L}_{G_X} + (1 - \alpha)\mathcal{L}_{G_Y}$. The parameter $\alpha \in [0, 1]$ defines how much emphasis will be put in each mapping, i.e. higher values of α indicate that transfers from $X \to Y$ are more relevant to the task at hand than $Y \to X$. In addition, cycle-consistency is enforced by applying a L1 penalty to the reconstructed versions of inputs:

$$\mathcal{L}_C = \mathbb{E}_{\mathbf{x}}\Big[||\mathbf{x} - \bar{\mathbf{x}}||_1\Big] + \mathbb{E}_{\mathbf{x}}\Big[||\mathbf{y} - \bar{\mathbf{y}}||_1\Big]. \tag{5}$$

For the task-specific purpose of depth estimation, we take inspiration from [9] and include a gradient smoothness term, to encourage depth values to be locally smooth with an L1 penalty on depth gradients ∂Y. As depth discontinuities often coincide with image gradients ∂X, this distance penalty is weighted with an edge-aware term, so that:

$$\mathcal{L}_{S_X} = \frac{1}{wh}\sum_{i,j}^{w,h}\Big(|\partial_x Y_{ij}|e^{-||\partial_x X_{ij}||} + |\partial_y Y_{ij}|e^{-||\partial_y X_{ij}||}\Big). \tag{6}$$

The same intuition can be applied to the inverse mapping, in the sense that image discontinuities ∂X often coincide with depth gradients ∂Y, to produce the gradient smoothness term $\mathcal{L}_{S_Y}$. The full gradient smoothness objective function, that is applicable to both generators, is the sum of each individual component weighted by α, so that $\mathcal{L}_S = \alpha\mathcal{L}_{S_X} + (1 - \alpha)\mathcal{L}_{S_Y}$. Now we can define the full objective function to be minimized during training, which is of the form:

$$\mathcal{L} = \mathcal{L}_G + \mathcal{L}_D + \beta_C\mathcal{L}_C + \beta_S\mathcal{L}_S, \tag{7}$$

where β_C and β_S control the relative importance of the cycle and smoothness components, respectively, in relation to the adversarial loss for generative and discriminative components. Furthermore, a global measure of convergence can be derived by finding the closest reconstruction $\mathcal{L}(\mathbf{x})$ with the lowest absolute value of the instantaneous process error, which is formulated as:

$$\mathcal{C} = \mathcal{C}_X + \mathcal{C}_Y = \mathcal{L}(\mathbf{x}) + |\gamma\mathcal{L}(\mathbf{x}) - \mathcal{L}(\hat{\mathbf{x}})| + \mathcal{L}(\mathbf{y}) + |\gamma\mathcal{L}(\mathbf{y}) - \mathcal{L}(\hat{\mathbf{y}})|, \tag{8}$$

which is used to determine when the network has reached its final state or if the model has collapsed (i.e. every input produces the same output), as shown in Figure 1b.

3.3 Reconstructed Depth Images

A common way to store range-based sensor data is through the use of pointclouds, which can be easily projected back into a 2D plane to produce depth images, containing distance estimates for all pixels that have a corresponding world point. These projections constitute the domain Y for function mapping within the proposed CycleDepth framework, and conversely reconstructed depth images can be projected back to recover their corresponding pointclouds. Assuming a rectified camera projection matrix $\mathbf{P}_{rect} \in \mathbb{R}^{3\times4}$, a rectifying rotation matrix $\mathbf{R}_{rect} \in \mathbb{R}^{3\times3}$ and a rigid body transformation matrix from camera to range-based sensor $\mathbf{T}^{cam}_{range} \in \mathbb{R}^{4\times4}$, a 3D point $\mathbf{p}$ can be projected into pixel $\mathbf{u}$ as such:

$$\mathbf{u} = \mathbf{P}_{rect}\mathbf{R}_{rect}\mathbf{T}^{cam}_{range}\mathbf{p}\,. \tag{9}$$

An example of this projection can be seen in Fig. 2. The input image is shown in (a), in (b) we can see the input pointcloud, and (d) depicts the projected 3D points on the 2D image plane, colored by depth. The corresponding depth image is shown in (c), where we can notice its sparsity, mostly due to low pointcloud density in areas further away from the sensor. Spatial dependency modeling is a crucial aspect in computer vision, and the introduction of such irregular gaps can severely impact performance. Because of that, here we propose projecting not the pointcloud itself, but rather its occupancy model, as generated by the Hilbert Maps (HM) framework. This methodology has recently been successfully applied to the modeling of large-scale 3D environments [33], producing a continuous occupancy function that can be queried at arbitrary resolutions. We employ the same feature vector from [33], defined by a series of squared exponential kernel evaluations against an inducing point set $\mathcal{M} = \{\mathcal{M}_i\}_{i=1}^M = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^M$, obtained by clustering the pointcloud and calculating mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ estimates for each subset of points:

$$\Phi(\mathbf{x}, \mathcal{M}) = \Big[k(\mathbf{x}, \mathcal{M}_1), \ldots, k(\mathbf{x}, \mathcal{M}_M)\Big]\ ,\ k(\mathbf{x}, \mathcal{M}_i) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)\right). \tag{10}$$

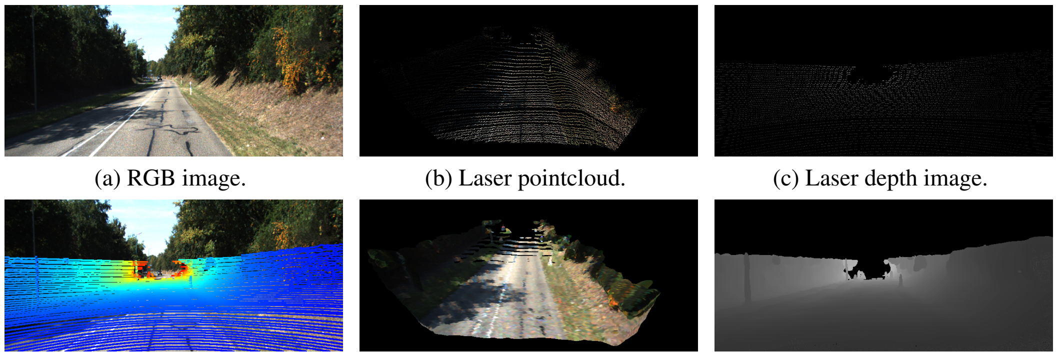

(a) RGB image. (b) Laser pointcloud. (c) Laser depth image.
(d) RGB image + Laser. (e) Reconstruction (prob. 80%). (f) Reconstructed depth image.

Figure 2: Example of reconstructed pointcloud projection using the Hilbert Maps framework.

Clustering is performed using the Quick-Means algorithm proposed in [34], due to its computational efficiency and ability to produce consistent cluster densities. However, this algorithm is modified to account for variable cluster densities within a function, in this case the distance d from origin. This is achieved by setting $r_i = r_o = \tau \cdot f(d)$, where r_i and r_o are the inner and outer radii used to define cluster size and τ is a scaling constant. The intuition is that areas further from the center will have fewer points, and therefore larger clusters are necessary to properly interpolate over such sparse structures. The trade-off for this increase interpolative power is loss in structure details, since a larger volume will be modeled by the same cluster.

Once the occupancy model has been trained, it can be queried (Eq. 3) to produce a reconstruction of the environment given a certain occupancy probability, as shown in Fig. 2e. Afterwards, each pixel can be checked for collision in the 3D space, producing depth estimates. An example of reconstructed depth image is depicted in Fig. 2f, where we can see that virtually all previously empty areas in Fig. 2c were filled by the occupancy model, while maintaining spatial dependencies intact (up to the reconstructive capabilities of the Hilbert Maps framework).

4 Experimental Results

Four different datasets were used to validate the proposed CycleDepth framework: *KITTI* [35] and *CaRINA* [36], containing monocular images and Velodyne pointclodus from vehicles driving in urban environments; *DSO* [37], containing monocular images and SLAM pointclouds from a hand-held camera in outdoor environments; and *NYU* [38], containing sequences from a variety of indoor scenes recorded with a RGBD camera. On the first three datasets, pointclouds were first transformed into depth images (see Section 3.3), based on the intrinsic parameters of the RGB camera. Note that, even though these datasets contain paired input-outputs, different sets were used to obtain RGB and depth training data, thus simulating an unpaired learning scenario. The same on-the-fly data augmentation scheme from [9] was used, with a 50% chance of horizontal flip and a 50% chance of random gamma, brightness and color shifts. Further implementation details are provided in the supplementary material, to facilitate reproduction.

4.1 Individual Datasets

The first experiment involved the *KITTI* dataset, using 12 different sequences for a total of 11957 RGB and depth images. From these, cross-validation was performed by rotatively selecting 5 runs for RGB training, another 5 for depth training and the remaining 2 for testing. Some results are depicted in Fig. 3, including comparisons with MonoDepth [9] and the standard CycleGAN [10]. Note that all other considered techniques require paired data for training, either as ground-truth depth estimates or stereo images, while the proposed CycleDepth framework was trained solely on unpaired data and was still able to achieve visually compelling results. In Fig. 4a we see some examples of pointcloud reconstruction from recovered depth images, which can be applied to further tasks such as dense monocular SLAM [39]. Additionally, Fig. 4b depicts some examples of RGB image reconstruction from depth information, which is learned as a by-product of the proposed cycle-consistent methodology and can be used to hallucinate textures into range data. The same process was repeated for the *CaRINA* and *DSO* datasets, with results depicted in Fig. 5.

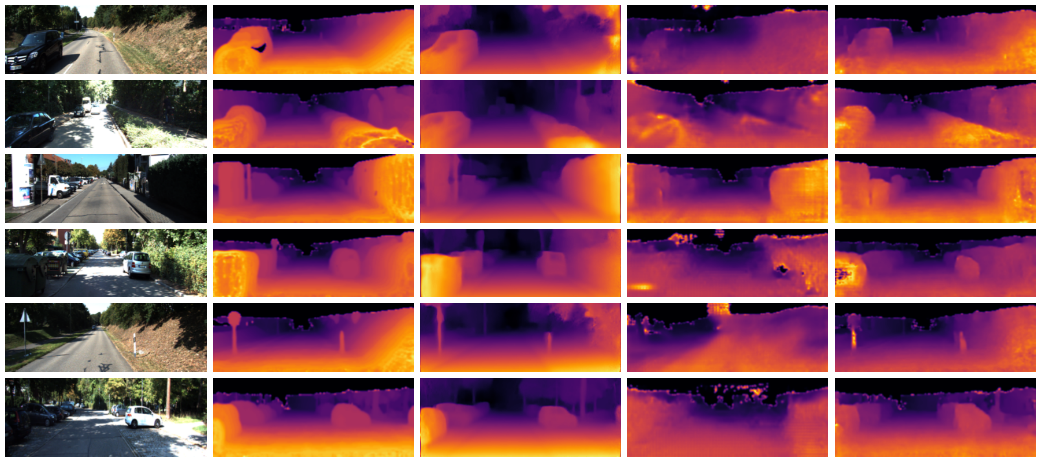

(a) Input image (b) Ground-Truth (c) MonoDepth (d) CycleGAN (e) CycleDepth

Figure 3: Monocular depth estimation results in the *KITTI* dataset using different techniques. A full evaluation video can be found in `https://www.youtube.com/watch?v=aGWhH8aY6FY&feature=youtu.be`.

For completeness, experiments were also conducted with the *NYU* dataset, by randomly selecting 1200 RGB and depth images for training and using the remaining 249 for testing. While outdoor datasets produced more consistently accurate depth estimates (see Table 1), indoor results were much more varied, with some testing samples producing surprisingly accurate estimates while others failed to provide meaningful measures. We attribute this larger variance to the higher diversity present in indoor environments, while outdoor datasets are more structured. The relatively small number of training samples could also be a factor, however a deeper analysis is left for future work. Note that, while cycle-consistent training requires four different networks, inference can be performed using only G_X for RGB or G_Y for depth image generation. Because of that, we were able to achieve end-to-end speeds of up to 40 Hz with 256×128 input images, using a NVIDIA Titan XP GPU card), which makes the proposed approach suitable for real-time applications using high-end processors.

4.2 Cross-Training

To further test the unpaired learning properties of CycleDepth, we also performed experiments using RGB and depth images from different sources. Note that this includes projecting the input pointclouds from one dataset as depth images using the intrinsic parameters from the RGB camera on the other dataset. We noticed this approach produces substantially better results than using depth images projected within the same dataset, while not violating the proposed unpaired learning assumption,

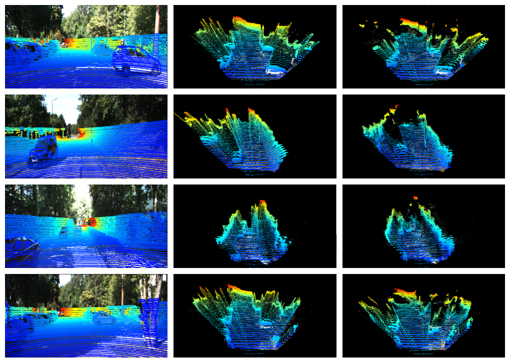
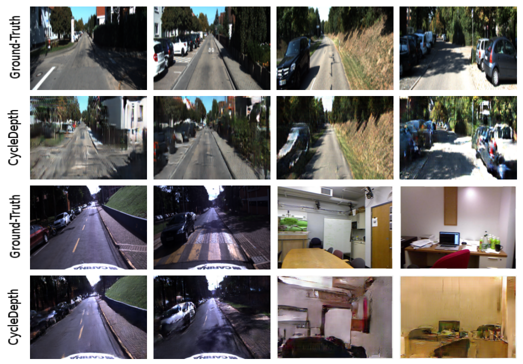


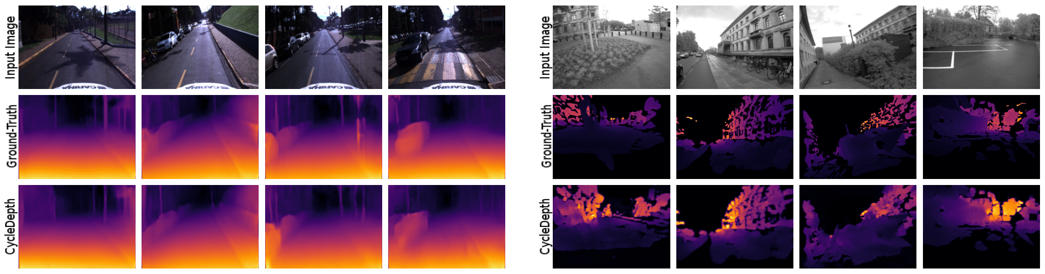
Image + laser Ground-truth Reconstruction

(a) Pointcloud reconstruction from RGB images.



(b) RGB image reconstruction from pointclouds.

Figure 4: Examples of end-to-end reconstruction (RGB $\longleftrightarrow$ Pointcloud) using CycleDepth.



(a) *CaRINA* dataset. (b) *DSO* dataset.

Figure 5: Examples of monocular depth results using CycleDepth in different datasets.

since it allows the use of datasets without information from both sensors. This virtual camera was positioned facing forward at the center of the pointcloud ($\mathbf{T}^{cam}_{range} = I$), however this is not strictly necessary, and different points of views can be used in a data augmentation scenario to generate multiple depth images from the same pointcloud.

A quantitative comparison between different learning-based monocular depth estimation techniques can be found in Table 1, both for individual and cross-training. The CycleDepth and CycleGAN algorithms were trained as described previously, and pre-trained models freely available online were used for the others. Interestingly, CycleDepth achieved better results than some of the other techniques, despite its more restrictive training methodology. We attribute this behavior to a higher similarity between training and testing data, which is possible due to the unpaired properties within the proposed framework. Additionally, cross-training between the *KITTI* and *CaRINA* datasets did not significantly affect performance, indicating that CycleDepth is indeed capable to generalize over different datasets to produce accurate mappings between domains. However, it still relies on structure similarity, as shown by a performance decrease when the *DSO* dataset, containing data collected from a hand-held camera, is introduced, even though it relatively useful estimates are still produced. This behavior is further amplified by the cross-training between indoor and outdoor datasets, that failed to provide meaningful results due to radical differences in data collection methods.

Table 1: Quantitative results using different learning-based monocular depth estimation techniques. The same scale-invariant log-depth error metric introduced in [40] is used for comparison.

Method		Root Mean Squared Error (log-depth, scale-invariant)			
		KITTI	*CaRINA*	*DSO*	*NYU*
MonoDepth [9]		0.181 ± -0.710	0.205 ± -0.659	0.342 ± -0.499	$---$
DepthMap [40]		0.214 ± -0.768	0.252 ± -0.710	0.435 ± -0.437	0.151 ± -1.011
CNN-Depth [41]		0.199 ± -0.758	0.257 ± -0.659	0.408 ± -0.534	0.494 ± -1.312
CycleGAN [10]		0.511 ± -0.057	0.543 ± -0.094	$0.662 \pm +0.020$	0.220 ± -0.572
CycleDepth	*KITTI*	0.220 ± -0.499	0.247 ± -0.358	0.404 ± -0.069	$---$
	CaRINA	0.260 ± -0.383	0.212 ± -0.467	0.373 ± -0.122	$---$
	DSO	0.379 ± -0.251	0.347 ± -0.271	0.252 ± -0.291	$---$
	NYU	$---$	$---$	$---$	0.265 ± -0.409

5 Conclusion

This paper proposes a novel learning-based methodology for monocular depth estimation that does not require paired data during training. Differently from other techniques, it uses a cycle-consistent generative adversarial network to directly learn mappings between RGB and depth images, without the need for data synchronization of any kind. Training is performed using a novel adversarial loss function with boundary-equilibrium properties, to improve stability while avoiding mode collapse, and introduces a cycle-consistent gradient smoothness component to improve performance on monocular depth estimation tasks. Experiments were conducted using a variety of well-known indoor and outdoor datasets, with the proposed CycleDepth framework achieving competitive results when compared to state-of-the-art pairing-based techniques. Additionally, we introduce a novel technique for depth image generation from sparse pointclouds, that produces denser representations by projecting Hilbert Maps occupancy models. Future work will focus on improving the various aspects of cycle-consistent training and inference, aiming for better generalization properties, and leveraging the use of simulated data, for which the proposed technique is particularly well suited.

Acknowledgments

This research was supported by funding from the Faculty of Engineering & Information Technologies, The University of Sydney, under the Faculty Research Cluster Program.

References

[1] Y. Furukawa and C. Hernndez. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2015.

[2] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[3] A. Abrams, C. Hawley, and R. Pless. Heliometric stereo: Shape from sun position. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[4] R. Roberts, C. Potthast, and F. Dellaert. Learning general optical flow subspaces for egomotion estimation and detection of motion anomalies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[5] V. Guizilini and F. Ramos. Semi-parametric learning for visual odometry. *International Journal of Robotics Research (IJRR)*, 2013.

[6] K. Karsch, C. Liu, and S. Kang. Depth transfer: Depth extraction from video using non-parametric sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2014.

[7] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2014.

[8] R. Garg, V. Kumar, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[9] C. Godard, O. Mac-Aodha, and G. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

[11] Y. Li, K. Qian, T. Huang, and J. Zhou. Depth estimation from monocular image and coarse depth points based on conditional GAN. In *MATEC Web of Conferences*, volume 175, 2018.

[12] A. Atapour-Abarghouei and T. Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[13] H.-Y. Fish, A. Herley, W. Seto, and K. Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[14] F. Ramos and L. Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015.

[15] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[16] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[18] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[20] I. Goodfellow, J. Pouget-Abadie, X. Mehdi, X. Bing, D. Warde-Farley, S. Ozair, A. Courville, and J. Bengio. Generative adversarial networks. In *arXiv:1406.2661*, volume 2014, Advances in Neural Information Processing Systems (NIPS).

[21] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

[22] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. In *arXiv:1701.00160*, 2016.

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[24] J. Z. M. Mathieu and Y. LeCun. Energy-based generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[25] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. In *arXiv:1702.08431*, 2017.

[26] G. Sansone. *Orthogonal Functions: Revised English Version*. Dover Books on Mathematics, 2012.

[27] B. Schölkopf, K. Muandet, K. Fukumizu, S. Harmeling, and J. Peters. Computing functions of random variables via reproducing kernel Hilbert space representations. *Statistics and Computing*, 25(4):755–766, 2015.

[28] P. Komarek. Logistic regression for data mining and high-dimensional classification. Technical report, Carnegie Mellon University, 2004.

[29] D. Freedman. *Statistical Models: Theory and Practice*. Cambridge University Press, 2005.

[30] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the International Conference on Computational Statistics (COMPSTAT)*, pages 177–186, 2010.

[31] V. Dumoulin, I.Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[32] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. In *arXiv:1701.07875*, 2017.

[33] V. Guizilini and F. Ramos. Large-scale 3d scene reconstruction with Hilbert maps. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[34] V. Guizilini and F. Ramos. Learning to reconstruct 3d structures for occupancy mapping. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

[35] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[36] P. Shinzato, T. Santos, L. Rosero, D. Ridel, C. Massera, F. Alencar, M. Batista, A. Hata, F. Osorio, and D. Wolf. Carina dataset: An emerging-country urban scenario benchmark for road detection systems. In *Proceedings of the International Conference on Intelligent Transportation Systems (ITSC)*, 2016.

[37] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. In *arXiv:1607.02555*, 2016.

[38] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[39] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[40] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[41] I.Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *Proceedings of the International Conference on 3D Vision*, 2016.

Supplementary Material for: Unpaired Learning of Dense Visual Depth Estimators for Urban Environments

Vitor Guizilini and Fabio Ramos
School of Information Technologies
University of Sydney, Australia
{vitor.guizilini;fabio.ramos}@sydney.edu.au

1 Architectures

Here we provide implementation details that facilitate the reproduction of results available in the main paper, more specifically related to the generators and discriminators used in the proposed cycle-consistent monocular depth estimation algorithm. Although not necessary, for simplicity both G_X G_Y generators have the same architecture, and are used to map inputs from one domain into the other, so that $\hat{\mathbf{x}} = G_X(\mathbf{y})$ and $\hat{\mathbf{y}} = G_Y(\mathbf{x})$, where $\mathbf{x}$ and $\mathbf{y}$ are real RGB and depth images respectively. Similarly, both D_X and D_Y discriminators also have the same architecture, and are used to separate between real and fake samples from each domain, i.e. between $\{\mathbf{x}, \hat{\mathbf{x}}\}$ and $\{\mathbf{y}, \hat{\mathbf{y}}\}$ respectively. The choice of parameters that compose the proposed boundary-equilibrium, cycle-consistent and gradient-smooth loss function is also presented and discussed, including how variations of these parameters might affect results in different ways.

1.1 Generator

The generative architecture is a fully convolutional auto-encoder inspired by [1], in which input images are first projected into low-dimensional, multiple-channel feature maps and then reconstructed to produce the corresponding output in a different domain (i.e. color to depth or vice-versa). A diagram of the proposed generative architecture is depicted in Figure 1, including its various modules. The input image is first padded with reflected information and goes through a series of convolutions, first with kernel size $k = 7 \times 7$ and stride $s = 1$ (no image dimension change) and then with kernel size $k = 3 \times 3$ and $s = 2$ (image dimension is decreased by a factor of 2). At the same time, the number c of channels in the resulting feature maps increase, so more patterns can be simultaneously captured. The Leaky Rectified Linear Unit (LReLU) activation function [2], with $\alpha = 0.05$, is used to introduce non-linearities between layers, in conjunction with instance normalization [3], that has been shown to substantially improve image stylization results. Afterwards, a series of 9 residual blocks [4], commonly used to improve optimization of deeper networks, serve to further process these feature maps, before two sequential transposed convolutions (i.e. deconvolutions), to recover the original input dimensions, and a final padded convolution that brings the number of channels to the required output dimensionality (1 for depth and 3 for color images). For the final layer, a hyperbolic tangent $tanh$ activation function is used, to produce values between $[-1, 1]$ that can be directly mapped as pixel intensities.

1.2 Discriminator

The discriminator architecture is inspired by [5], but without fully connected layers for hidden state generation. Each convolutional block is composed of two convolution operations with kernel sizes $k = 3 \times 3$ and Exponential Linear Unit (ELU) activation functions [6]. This is followed by a resize operation that modifies the spatial dimensions of current feature maps ($m = 1/2$ is a downsample to half the original dimensions, $m = 2$ is an upsample to double the original dimensions, using nearest neighbors, and $m = 1$ leaves the dimensions untouched). These convolutional blocks are repeated four times, with increasing number c of channels as spatial dimensions decrease. Afterwards, this

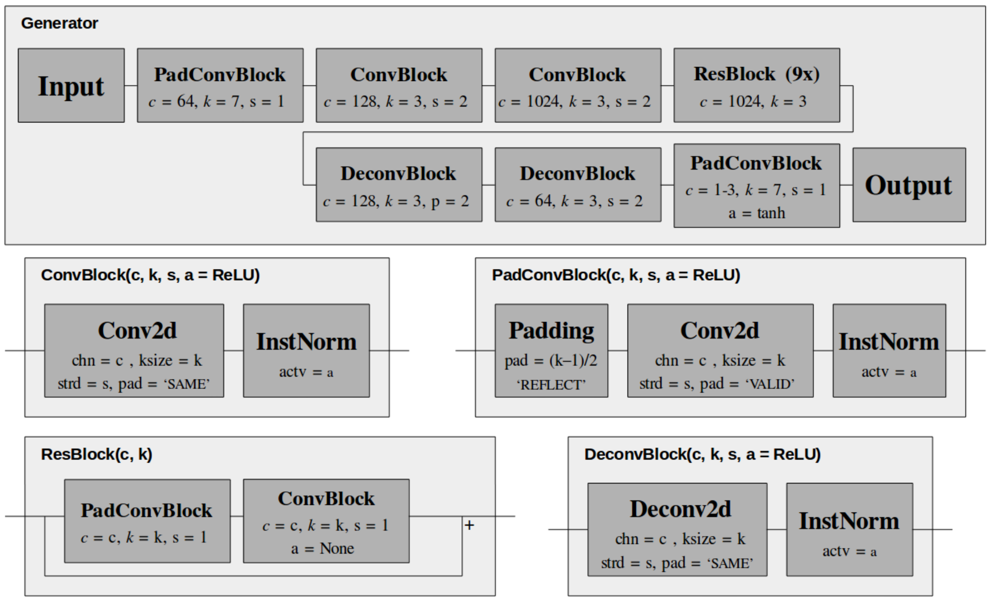

Generator
Input
PadConvBlock c = 64, k = 7, s = 1
ConvBlock c = 128, k = 3, s = 2
ConvBlock c = 1024, k = 3, s = 2
ResBlock (9x) c = 1024, k = 3
DeconvBlock c = 128, k = 3, p = 2
DeconvBlock c = 64, k = 3, s = 2
PadConvBlock c = 1-3, k = 7, s = 1 a = tanh
Output
ConvBlock(c, k, s, a = ReLU)
Conv2d chn = c , ksize = k strd = s, pad = 'SAME'
InstNorm actv = a
PadConvBlock(c, k, s, a = ReLU)
Padding pad = (k–1)/2 'REFLECT'
Conv2d chn = c , ksize = k strd = s, pad = 'VALID'
InstNorm actv = a
ResBlock(c, k)
PadConvBlock c = c, k = k, s = 1
ConvBlock c = c, k = k, s = 1 a = None
+
DeconvBlock(c, k, s, a = ReLU)
Deconv2d chn = c , ksize = k strd = s, pad = 'SAME'
InstNorm actv = a

Figure 1: Proposed CycleDepth generative architecture, used in experiments.

process is reversed and four other convolutional blocks are used, now with increasing spatial dimensions and decreasing number of channels. Skip-connections [7] are also used to facilitate gradient propagation between layers, as the concatenation of hidden states directly up-sampled to match the output dimensions of each convolutional block. The final convolutional block brings the number of channels to the required output dimensionality (1 for depth and 3 for color images) and uses a hyperbolic tangent $tanh$ activation function to produce values between $[-1, 1]$ that can be directly mapped as pixel intensities.

1.3 Parameters

In all experiments, we set the cycle-consistency and gradient smoothness ratios $\beta_C = 10$ and $\beta_S = 25$, as shown in Equation 7 of the main paper. A mapping ratio $\alpha = 0.6$ was used, to determine the weight of individual domain transfers, and a diversity parameter $\gamma = 0.3$ was used to balance between generator/discriminator optimization, as shown in Equation 4 of the main paper. Different

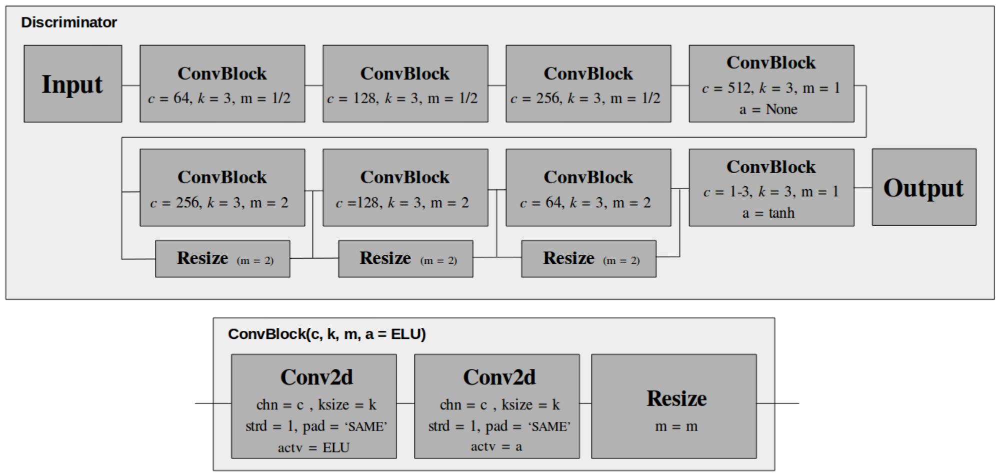

Discriminator
Input
ConvBlock c = 64, k = 3, m = 1/2
ConvBlock c = 128, k = 3, m = 1/2
ConvBlock c = 256, k = 3, m = 1/2
ConvBlock c = 512, k = 3, m = 1 a = None
ConvBlock c = 256, k = 3, m = 2
ConvBlock c =128, k = 3, m = 2
ConvBlock c = 64, k = 3, m = 2
ConvBlock c = 1-3, k = 3, m = 1 a = tanh
Output
Resize (m = 2)
Resize (m = 2)
Resize (m = 2)
ConvBlock(c, k, m, a = ELU)
Conv2d chn = c , ksize = k strd = 1, pad = 'SAME' actv = ELU
Conv2d chn = c , ksize = k strd = 1, pad = 'SAME' actv = a
Resize m = m

Figure 2: Proposed CycleDepth discriminative architecture, used in experiments.

Adam optimizers [8] were used for generators and discriminators, with learning rate 0.0001 for the first 50 epochs and decreasing linearly by a factor of 100 over the next 250 epochs, for a total of 300 epochs. We started the equilibrium parameter $k_0 = 0$ with a constant learning rate of $\lambda = 0.001$, as shown in Eq. 4. At each epoch, X and Y were independently randomized and processed sequentially with a batch size of 2. Wherever reconstructed depth images were used, as introduced in Section 3.3 of the main paper, the scaling constant for average cluster distance was set to $\tau = 0.01$, and probabilities $p(y_* = 1|\Phi(\mathbf{x}_*), \mathbf{w}) > 0.8$ were considered as occupied for surface generation.

References

[1] J.-Y. Zhu, T. Park, P. Isola, and A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[2] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. In *http://arxiv.org/abs/1505.00853*, 2015.

[3] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient to fast stylization. In *arXiv:1607.08022*, 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. In *arXiv:1702.08431*, 2017.

[6] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[7] G. Huang, Z. Liu, K. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.