Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische Friedrich-Wilhelms-Universität Bonn

Master Thesis

Synthetic-to-Real Domain Adaptation Using Contrastive Unpaired Translation

Author:
Benedikt Tobias Imbusch

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Jun.-Prof. Dr. Florian Bernard

Supervisor:
Max Schwarz

May 2022

Declaration

I hereby declare that I am the sole author of this thesis and that none other than the specified sources and aids have been used. Passages and figures quoted from other works have been marked with appropriate mention of the source.

Place, Date

Signature

Abstract

Deep learning models for vision have become a key component of many robotic systems over the last years. Their usefulness is largely dependent on the availability of training data. In practice, manual annotation of training data, e.g. for semantic segmentation, is often difficult and economically not feasible. Synthetic images are a viable alternative as the ground truth data for deep learning tasks can be retrieved directly from the renderer. However, synthetic data suffers from the domain gap: Visual differences between real and synthetic images lead to impaired generalization performance when a network is trained on synthetic data and inference is done on real images.

In this thesis, we follow a learning-based approach to perform synthetic-to-real domain adaptation and thus minimize the domain gap. In combination, we propose a multi-step method to obtain training data without manual annotation effort: From 3D object meshes, we generate images using the modern synthesis pipeline Stillleben. After a proof-of-concept for supervised domain adaptation, we focus on utilizing a state-of-the-art unsupervised image-to-image translation method for the subsequent adaptation step. The translation network is trained from unpaired images, i.e. just requires an un-annotated collection of real images. The generated and refined images can then be used to train deep learning models for a particular task. We also propose and evaluate extensions to the translation method that further increase performance, such as patch-based training, which shortens training time and increases global consistency.

We evaluate our method and demonstrate its effectiveness on two robotics datasets, YCB-Video and HomebrewedDB. Moreover, we give insight into the learned refinement operations.

Finally, we introduce and briefly evaluate other domain adaptation directions, which include a real-to-synthetic and a symmetric approach.

Contents

1 Introduction

Robotic systems need to address several key challenges in order to be able to autonomously act in a dynamic environment. Among these are computer vision tasks like semantic segmentation, object recognition, and 6D pose estimation. Nowadays, these tasks are most commonly solved using deep learning techniques. With increasing computation resources available, more complex network architectures are developed, raising the need for increasing amounts of training data. Acquiring training data, however, often involves tedious manual annotation of images with semantic labels or 6D poses. It is typically not feasible to create custom datasets for every specific setup.

To overcome this issue, previous approaches successfully relied on fine-tuning of networks pre-trained on generic datasets, reducing the required annotation effort (Morrison et al. 2018; Schwarz, Lenz, et al. 2018). Recently, approaches were introduced to generate synthetic training images, e.g. from 3D object meshes like in *Stillleben* (Schwarz and Behnke 2020). The benefit of such techniques is that ground truth data—like 6D object poses or semantic segmentation masks—is trivially available from the renderer, eliminating the need for manual annotation while providing highly accurate annotations. While Stillleben yields good generalization to real test images on the *YCB-Video* dataset (Xiang, Schmidt, et al. 2018) for semantic segmentation, the achieved results are still considerably inferior compared to training on real images (Schwarz and Behnke 2020). The reason for this difference is the so-called domain gap between synthetic and real data, i.e. the discrepancy between the synthetic data distribution and the real data distribution. Therefore, the model learned by a segmentation network trained on synthetic data is able to only partly capture the real data distribution from which the data is sampled at inference time.

We aim to obtain better results from purely synthetic data and therefore need to align the distributions more closely. We propose to tighten the domain gap by learning a mapping from the synthetic to the real image distribution.

First, as a proof-of-concept, we investigate the domain adaptation task in a supervised manner based on a *U-Net* network (Ronneberger, Fischer, and Brox 2015). As a supervised approach for this problem needs annotated real data and involves other limitations, we only demonstrate the general feasibility for our scenario and

do not evaluate it further.

Subsequently, we address the domain adaptation problem in an unsupervised manner, which forms the core of our thesis. Specifically, we only require synthetic data with ground truth and un-annotated real data, without direct correspondences between the images of both datasets. To learn the mapping from synthetic to real data, we apply the *GAN*-based *CUT* approach by Park et al. (2020) in a patch-based manner. Key challenges to be addressed here include the handling of backgrounds and ensuring shape consistency.

We evaluate the unsupervised method on a semantic segmentation task on both the YCB-Video dataset and the *HomebrewedDB* dataset (Kaskman et al. 2019). Our results show large performance improvements, nearly reaching what is possible with real annotated training data for YCB-Video. To understand how the observed performance improvements can be explained, we further examine deep image features of real, synthetic, and refined synthetic frames using the *LPIPS* distance (Zhang et al. 2018) and t-SNE embeddings.

Combining the creation of synthetic images using Stillleben, the unsupervised domain adaptation method and the following use of the refined images for seman-

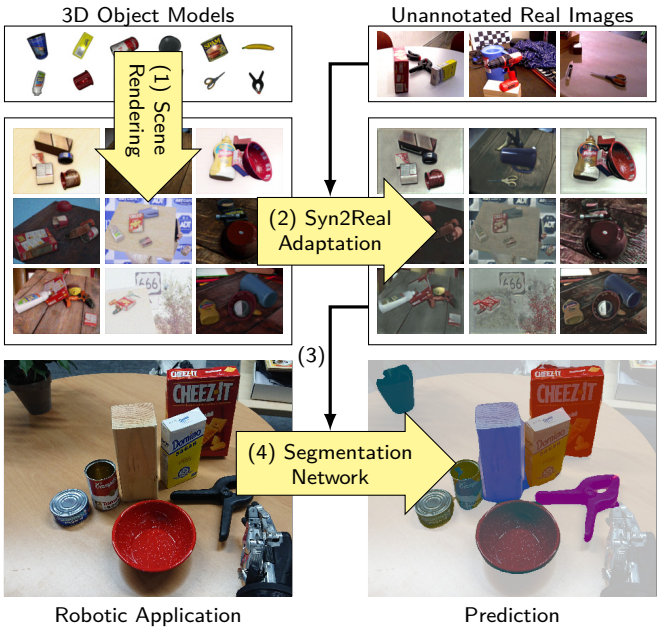


Figure 1.1: Our method yields robust task performance in real settings, just from 3D object models and unannotated real images (top). We simulate and render plausible scenes from the 3D meshes (1). Our adaptation model aligns the synthetic and real image distributions more closely (2). The refined image dataset is used to train a task-specific network (3), which is applied in the target domain (4). None of these steps requires annotations.

tic segmentation, we propose a complete pipeline, visualized in Figure 1.1: Our pipeline provides high-quality semantic segmentations—e.g. for robotic grasping applications—based on only 3D object models and un-annotated real images. We highlight that no step of this pipeline requires manual annotation effort.

This is complemented by a discussion of possible extensions of the presented approach and more conceptual changes with respect to the adaptation direction, supported by initial experimental results and founded on the insights gained in the evaluation.

This thesis is structured as follows: In Chapter 2, we introduce several key concepts relevant for the following chapters. After discussing related work in Chapter 3, this thesis contains the following contributions:

1. A multi-step method to obtain annotated training data from 3D object meshes and real images, first using a proof-of-concept supervised domain adaptation approach (with annotated real images) and afterwards using a patch-based application of the CUT approach for unsupervised domain adaptation without any annotated real data (Chapter 4),

2. an evaluation of the unsupervised method on a segmentation task on two robotics datasets, alongside an analysis of the learned adaptation operations (Chapter 5), and

3. a discussion of further domain adaptation strategies that go beyond the synthetic-to-real setting using a modified version of the unsupervised method (Chapter 6).

In Chapter 7, the thesis is concluded by a summary of our results and possible starting points for future work.

2 Fundamentals

2.1 Domain Adaptation For Visual Data

Consider two related but distinct data distributions $\mathcal{X}$ and $\mathcal{Y}$, for instance synthetic and similar real images. Due to camera properties and complex physical object properties, synthetic images cannot completely match the appearance of real images. The difference between both distributions is called the domain gap. In settings where we can sample from $\mathcal{X}$ but not from $\mathcal{Y}$—e.g. when we need annotated training data for some learning task but annotated real data is not available—and still need to learn as much as possible about $\mathcal{Y}$, we may want to reduce the domain gap. This process is called domain adaptation and essentially refers to approximating a mapping $f : \mathcal{X} \to \mathcal{Y}$. Solving this task for visual data using deep neural networks is an established area of research, as is apparent from the literature review by M. Wang and Deng (2018).

They group the approaches into two main categories: heterogeneous and homogeneous domain adaptation. The former refers to the case when the domain gap arises from the fact that source and target domain have different feature spaces. In the latter case, both domains share their feature space but still the respective distributions $\mathcal{X}$ and $\mathcal{Y}$ do not match. In this work, we address homogeneous domain adaptation, as do the related domain adaptation approaches presented in Section 3.3. Another distinction made by M. Wang and Deng (2018) is one-step vs. multi-step domain adaptation: There may exist situations, where the domain gap is too large to be effectively reduced by a single translation network. To address this issue, multi-step domain adaptation makes use of intermediate representations between the two relevant distributions. In our case, one-step domain adaptation is sufficient because the synthetic and the real image distribution are rather close and we only intend to learn minor optical changes while—most importantly—exactly preserving object shapes.

In this thesis, $\mathcal{X}$ denotes the synthetic images distribution and a set of samples x thereof is defined as $x \in X \subsetneq \mathcal{X}$. Likewise, $\mathcal{Y}$ denotes the real data distribution.

2.2 Generative Adversarial Networks (GANs)

First proposed by Goodfellow et al. (2014), *Generative Adversarial Networks (GANs)* denote a neural network paradigm used for unsupervised generative modeling, i.e. for approximating a data-generating distribution p_{data}. The approach is not restricted but commonly applied to image data. As this thesis focuses on image data, we explain GANs with respect to the example of images.

A GAN typically consists of two neural networks: The generator network and the discriminator network. In the original publication by Goodfellow et al., the generator $G(z, \theta_g)$ creates images based on its parameterization θ_g and a latent input vector z, sampled as $z \sim p_z$ from a prior distribution p_z. The goal is to train G such that the distribution p_g induced by the generator output for $z \sim p_z$ closely approximates p_{data}. To this end, a discriminator network $D(x, \theta_d)$ is used, where x is some image input and θ_d the discriminators parameterization. D is presented with real images from p_{data} and generated images from p_g and outputs a scalar value $d \in [0, 1]$, indicating whether x was a real or a generated ("fake") image.

Intuitively, both networks are competing to outperform each other: The generator is optimized to produce images that can "fool" its adversary, the discriminator, to classify them as real images. The discriminator, however, is trained towards better distinction of real and faked images. This setup induces a dynamic that results in generator output that looks similar to real data from p_{data}, i.e. turns the resulting p_g into a good approximation of p_{data}. In terms of game theory, this corresponds to a two-player minimax game between the generator G and the discriminator D. The described optimization is realized using the following shared objective function:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)} \left[\log D(x)\right] + \mathbb{E}_{z \sim p_z(z)} \left[\log\left(1 - D\left(G(z)\right)\right)\right], \tag{2.1}$$

where in practice the expected value is approximated using a mini-batch of real or generated samples. The first term drives the output of D towards 1 for real images and is irrelevant for the generator. The second term drives the output of D towards 0 for generated images when training the discriminator. By minimizing it in the optimization of G, G is encouraged to produce images that maximize the discriminator's output, i.e. let it classify the fake images as real. In the limit, the system should converge towards $D(x, \theta_d) = \frac{1}{2}$.

While in the original formulation by Goodfellow et al. (2014), G is provided with noise as input, it is also possible to condition the image generation on other images, resulting in an image-to-image translation setup. Park et al. (2020) make use of this modification in their approach which we use in this thesis, see Section 4.3.

This adaptation can—ignoring any possible skip connections—be interpreted as generating the latent vector z not randomly but directly from the source image distribution using an encoder network.

In practice, optimizing a GAN is a non-trivial task and might fail. Possible problems include vanishing gradients or mode collapse, denoting the case where the generator produces trivial output like e.g. black images, resulting in p_g largely deviating from p_{data}. A modification to the standard GAN definition—that Park et al. (2020) adopt as default in their implementation—has been proposed as *Least Squares Generative Adversarial Networks (LSGAN(s))* by Mao et al. (2017) to overcome the vanishing gradient problem and stabilize the training. The main difference is that the discriminator is trained using a least squares loss compared to the cross-entropy loss in the standard formulation. Intuitively, the benefit of the changed loss function is the following: Using cross-entropy, generated images that are on the correct side of the discriminator's decision boundary do not generate a training signal for the generator, although they might strongly deviate from p_{data}. With a least squares loss, there is still a significant training signal, i.e. gradient, in these cases. Mao et al. (2017) demonstrate that LSGANs generate images of higher quality and exhibit a more stable training behavior than the original GAN formulation.

2.3 Similarity Measures

To analyze the results of our unsupervised domain adaptation approach and for the training of the supervised method, we need similarity measures to compare images. In this thesis, we use two measures that go beyond classical pixel-wise distances like the L2 distance.

2.3.1 Learned Perceptual Image Patch Similarity (LPIPS)

The *Learned Perceptual Image Patch Similarity (LPIPS)* metric, proposed by Zhang et al. (2018), is a so-called perceptual metric. As a loss function, it can be used for training image-to-image tasks. It is calculated based on image features obtained by passing the images to be compared through a pre-trained *Convolutional Neural Network (CNN)*, e.g. *AlexNet* (Krizhevsky, Sutskever, and G. E. Hinton 2012) trained on *ImageNet*[1] as used in our work.

[1] https://www.image-net.org/index.php

Formally, the LPIPS is given by

$$\mathcal{L}_{\text{LPIPS}}(x, \hat{x}) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \left|\left| w_l \odot \left(y^l_{hw} - \hat{y}^l_{hw}\right)\right|\right|_2^2, \tag{2.2}$$

where x is the target image and $\hat{x}$ the compared image. y^l, $\hat{y}^l \in \mathbb{R}^{H_l \times W_l \times C_l}$ denote the corresponding normalized feature activations for layer l. $w_l \in \mathbb{R}^{C_l}$ scales the activations channel-wise per layer. Finally, $\odot$ denotes Hadamard's product.

Zhang et al. (2018) showed that this loss leads to visually superior results compared to using a per-pixel metric like the L2 norm. One of the reasons for this is that the L2 loss does not account for the dependencies between neighboring pixels which becomes problematic for tasks with image output (compared to e.g. classification output). Besides, pixel-wise losses like L2 often lead to blurred results, for optimization-theoretical reasons. Further details on perceptual losses as well as arguments for their superiority to per-pixel measures can be found in the work by J. Johnson, Alahi, and Fei-Fei (2016) and Zhao et al. (2017).

2.3.2 t-SNE embeddings

Image similarity for larger sets of images can also be compared by visualizing the underlying data distributions using *t-distributed stochastic neighbor embeddings (t-SNE embeddings)*. t-SNE embeddings have been introduced by Van der Maaten and G. Hinton (2008) as a technique for the visualization of high-dimensional data. It performs dimensionality reduction in such a way that meaningful two (or three) dimensional maps of the data are produced—aiming to preserve the essential structure of the manifold on which the high-dimensional data lies.

The key idea behind t-SNE embeddings is to align a low-dimensional distribution Q with the high-dimensional data distribution P. To achieve this, pairwise similarities are determined for the available samples from P and from this, conditional probabilities $p_{j|i}$ are calculated, where $p_{i|i} := 0$ and assuming a Gaussian around the datapoint with index i. In a similar manner, but assuming a *Student t-distribution*, conditional probabilities are also calculated for the same number of samples in the low-dimensional space as $q_{j|i}$. Based on this, P and Q are iteratively aligned by minimizing the *Kullback-Leibler divergence*

$$KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{2.3}$$

While experimental data by Van der Maaten and G. Hinton (2008) shows the superiority of their approach compared to related approaches in the context of

visualizing large high-dimensional datasets, some drawbacks are to be kept in mind: First, the Kullback-Leibler divergence is a non-convex function, meaning that the iterative optimization yields different results for each run, given a different initialization. Besides, visualizations of t-SNE embeddings have a tendency to suggest clusters in the data where there are none, as pointed out by Wattenberg, Viégas, and I. Johnson (2016).

To visualize images by calculating t-SNE embeddings, we could use raw pixel values. However, this results in a rather high input dimensionality and also produces embeddings that are not representative for what a neural network detects within the images. The latter is a particularly important point because we are interested in the benefit of refining synthetic images towards more realism for following deep learning tasks. It is of higher interest whether the domain adaptation process achieves a closer alignment of the feature activations of a pre-trained neural network than whether the raw pixel-value distributions are aligned closer. Therefore, meaningful t-SNE embeddings for our context can be obtained by calculating them on (pooled) feature activations of a pre-trained neural network, like for instance done by Schwarz, Schulz, and Behnke (2015).

3 Related Work

The need for large amounts of annotated data to train neural networks is a well known problem. Especially in scenarios where no existing dataset is suitable, there is a huge need to perform the training without these large amounts of manually labeled training data. To this end, techniques have been developed that follow different approaches to avoid the need for manually labeled training data. In the following, we present several of them, with a special focus on unsupervised domain adaptation for synthetic data as the main approach of this thesis belongs to this family of techniques. As we address semantic segmentation as the subsequent deep learning task after domain adaptation in this thesis, we focus on approaches that consider semantic segmentation.

3.1 Weakly-Supervised Learning

One group of techniques is largely self-supervised or only weakly supervised, but relies on some human interaction. Zhi et al. (2021) use RGB-D data from a handheld camera to learn a scene representation in a *Neural Radiance Fields (NeRF)*-based (see Mildenhall et al. (2020)) manner that—aside from the typical properties, namely RGB information and volume density—also models semantic information. They embed the used *Multi-Layer Perceptron (MLP)* in an interactive system in which the user can assign class labels to the determined semantic classes, starting from the beginning of the training. The authors compare their technique to state-of-the-art solutions that use large labeled datasets and find that their solution outperforms them given less than 100 interactions (i.e. clicks). Inherently, due to being NeRF-based, a shortcoming of the proposed solution is that the training is specific to a given scene. Thus, it can be used to more easily generate large labeled datasets or for robotic *SLAM* scenarios.

ScribbleSup by D. Lin et al. (2016) follows a different approach for semantic segmentation: They keep the reliance on large annotated datasets but instead of dense per-pixel annotations, only sparse scribble annotations like depicted in Figure 3.1 are required, drastically reducing the annotation effort. The authors use a graphical model to propagate the scribble annotations to the corresponding pixels

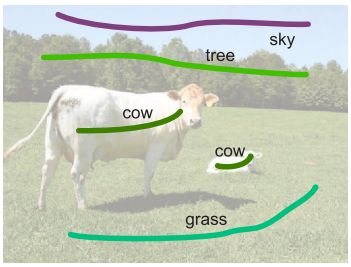


Figure 3.1: Scene with scribble annotations. Annotations of this form are needed by ScribbleSup for training. Taken from D. Lin et al. (2016).

and a CNN for the actual segmentation training. Both processes are included in a combined loss function and the optimization alternates between label propagation and network training. The network outputs are directly used to improve the labels. The authors report a segmentation performance slightly below what is achieved with full pixel-wise labels on the *PASCAL VOC 2012* dataset (Everingham et al. n.d.).

Ho et al. (2020) propose a slightly different approach: They developed a method for semantic segmentation that needs initial, partial labels using which a CNN is trained. During training, the network predictions are used to interactively improve the labels, e.g. by annotating complicated features not yet recognized by the network. This is done repeatedly as a loop of training, segmentation and correction. The application domain is medical image processing.

3.2 Feature Learning

Another way to overcome the lack of labeled training data is to use fully self-supervised methods. They are designed and trained to learn meaningful features or, in other words, embeddings of the input data. Based on these latent codes, subsequent tasks can be solved, like image classification.

VICReg by Bardes, Ponce, and Lecun (2022) is a representation learning technique for image processing. Their architecture consists of an encoder and an expander, where the first outputs the latent representation for subsequent deep learning tasks and the latter is used to calculate a loss that drives the training towards meaningful representations. The main contribution is the loss setup to—in parallel—balance variance, invariance and covariance. Most importantly, the latter is minimized over pairs of embedding components to ensure sufficient decorrelation and the first one is optimized to be above or at a given threshold for the individual embedding components to get meaningful latent variables. In combination, this prevents the well-known problem of mode collapse.

F. Wang et al. (2020) introduce a method for self-supervised feature learning called *Invariance Propagation*. Their technique is based on contrastive learning and aimed at learning features that are invariant to intra-class variance in order to learn more conceptual information, whereas many other techniques focus on intra-image variations. The authors evaluate their method on various deep learning tasks and report to outperform existing approaches on semi-supervised image classification, i.e. classification after fine-tuning a model using small parts of labeled datasets.

Baevski et al. (2022) propose *data2vec*, a multi-purpose approach for self-supervised learning, applicable to e.g. computer vision or language-based tasks. Their goal is to find latent representations of an input sample, like an image, based on masked versions of it. This is different from related approaches that are focused on one application domain and predict domain-specific representations. Technically, they use a *Transformer* network. The authors report performance on an image classification task that is comparable to superior to related approaches, depending on the data.

3.3 Unsupervised Domain Adaptation

The approach we follow in this thesis is unsupervised domain adaptation in a synthetic-to-real setup. The lack of annotated training data is addressed by generating automatically labeled synthetic data and refining these images towards more realism in an unsupervised way.

Stein and Roy (2018) apply domain adaptation to warehouse and outdoor scenes using the *CycleGAN* approach (Zhu et al. 2017). Similar to our method, they address a semantic segmentation task and use separate networks for domain adaptation and segmentation. However, the system as a whole is applied to robotic navigation and larger-scale scene understanding, in contrast to in our case robotic manipulation in small-scale scenes. This might explain why our initial experiments with CycleGAN did not yield satisfactory results. The CUT architecture employed by our approach is easier to train and generally yields better results (Park et al. 2020).

In similar manner, Mueller et al. (2018) successfully demonstrated the use of CycleGAN-based domain adaptation for synthetic training data. Their application domain is hand pose tracking. To ensure accurate preservation of the hand poses, they propose *GeoConGAN*, adding a geometric consistency loss to the CycleGAN objective. Probably resulting from our patch-based application of the newer CUT approach, we did not experience geometric inconsistencies. Therefore, adding

complexity in the form of another loss component calculated using an additional CNN appears not justified to us.

Shrivastava et al. (2017) use a GAN-based approach with a patch-based discriminator for synthetic-to-real domain adaptation of hands and eyes with the goal of pose estimation. They use L1 regularization on (identity or more complex) transformed image features to constrain the GAN towards preservation of the image content. The GAN's discriminator is trained on batches of refined images accumulated over time to stabilize the adversarial learning. CUT's contrastive learning-based approach for content consistency appears far more flexible and data-adapting to us, compared to the proposed L1 regularization.

Bousmalis et al. (2017) propose *PixelDA*, another GAN-based approach for domain adaptation. Like in our scenario, they apply it to small objects but focus more on classification and pose estimation while highlighting broader applicability. In addition to the standard setup consisting of a generator and a discriminator, they add a task-specific classifier to their model, trained on both synthetic and generator-refined synthetic images to support the domain adaptation. To maintain correspondences between the synthetic images and their refined versions, they propose to penalize content dissimilarities using a masked pairwise mean squared error, given depth data is available from the renderer. The resulting generated backgrounds, mainly replacing black backgrounds, appear rather noisy to us. While this might even benefit generalization for classification, we expect that more consistent backgrounds are needed in our case. Additional experiments that we performed at full resolution with the CUT architecture have shown a detrimental effect of masking out the backgrounds in our application domain.

CyCADA by Hoffman et al. (2018) is another domain adaptation approach derived from the idea of CycleGAN. This technique guides the adaptation process in two ways: The authors propose loss components for aligning the distributions both in the pixel space and the feature space. Besides, the authors suggest to use loss components specific to the subsequent deep learning task to enforce semantic consistency. Hoffman et al. (2018) report better performance on a semantic segmentation task after domain adaptation than for the existing unsupervised adaptation approaches. However, the training is computationally costly and complex—compared to the far simpler but still very effective objective of CUT.

The *DLOW* technique proposed by Gong et al. (2019) is based on the CycleGAN concept as well. It generalizes the idea of domain adaptation beyond mapping a source domain $\mathcal{S}$ to a target domain $\mathcal{T}$: The authors introduce a model for "domain flow generation". Intuitively, a parameter $z \in [0, 1]$ is introduced to control how far an image from $\mathcal{S}$ should be adapted towards $\mathcal{T}$. A mentioned key benefit of this technique is that learning the intermediate steps supports the domain adaptation

process. In their experiments, Gong et al. (2019) show improved results on a semantic segmentation task compared to plain CycleGAN domain adaptation. However, the improvement is not very substantial. The previously mentioned shortcomings of CycleGAN compared to CUT apply for this approach as well.

4 Methodology

To perform synthetic-to-real domain adaptation for use on subsequent deep learning tasks, our main goal is to find a mapping $f : \mathcal{X} \to \mathcal{Y}$ in a learning-based manner, where $\mathcal{X}$ denotes the synthetic and $\mathcal{Y}$ the real image distribution. As the real image distribution we assume the underlying distribution of YCB-Video (Xiang, Schmidt, et al. 2018) or related datasets containing images of several small objects. Figure 4.1 shows exemplary samples from $\mathcal{X}$ and $\mathcal{Y}$ in which the visual differences that form the domain gap can be seen.

Figure 4.1: Real YCB-Video frames (bottom) alongside corresponding synthetic images (top). It can be seen that, besides the backgrounds, there are visual differences in the object appearance that form the domain gap.

In this chapter, we present our methodology to find a suiting f in two main parts: In Section 4.2, we describe a supervised domain adaptation approach alongside its limitations. Thereafter, an unsupervised approach is presented in Section 4.3.

We remark that our primary goal for using domain adaptation is to tighten the domain gap. In a synthetic-to-real scenario, this essentially means to refine images to look more natural. However, we are not really interested in the appearance to the human eye but in getting better results for subsequent deep learning tasks based on refined synthetic images. Thus, "looking more natural" is not to be understood in the strict perceptual sense—although there are relations between deep images features and human perception, like demonstrated for instance by

Zhang et al. (2018) with regards to the LPIPS metric. In this thesis, we choose semantic segmentation as the subsequent task on which we evaluate our approach in Chapter 5.

4.1 Stillleben

The Stillleben library (Schwarz and Behnke 2020) is a framework for generation and rendering of cluttered tabletop scenes. Stillleben operates on arbitrary input meshes and generates random arrangements through the use of a physics engine. Besides, it is able to render arrangements with given object poses or to use different surfaces beyond the standard tabletop setup. The arranged scenes are then rendered with a modern *physics-based-rendering (PBR)* pipeline. This pipeline can put the images into complex, real lighting situations through support for *Image-based Lighting (IBL)* data. To obtain even more realistic images, a post-processing step adds effects simulating the behavior of a real camera, such as noise, chromatic aberration, white balancing errors, and over-/underexposure. Depending on the objects to be rendered, it might be that object surface features are missing in the meshes, like e.g. stickers for the objects from YCB-Video. Stillleben provides a means to add such surface features randomly. Alongside the RGB image data, ground truth data for downstream machine learning tasks like semantic segmentation or 6D pose estimation is provided, which is available directly from the renderer. Figure 4.2 depicts a generated image alongside more data on the visible objects from the renderer. A segmentation model trained with purely Stillleben-generated synthetic data has been shown to reach respectable performance on the YCB-Video dataset (Schwarz and Behnke 2020).

We use Stillleben as the rendering engine to create synthetic images for our domain adaptation approaches and as a part of the meshes-to-segmentation pipeline. For the supervised approach, we make use of Stillleben's capability to render images with given object poses that are provided with the YCB-Video dataset. Note that it is not straightforward to render a table surface in this case: First, we cannot directly ensure that all objects lie on the simulated table and second, we cannot guarantee the physical plausibility of the resulting images because the object poses are fixed and the real supporting surface might not be planar. Therefore, we restrict ourselves to rendering the fixed arrangements embedded in an IBL environment that provides lighting and a background. For the unsupervised approach, we have more degrees of freedom with respect to the arrangements. Therefore, we make use of Stillleben's PBR capabilities and render tabletop scenes with different table surfaces and random, yet physically consistent object arrangements

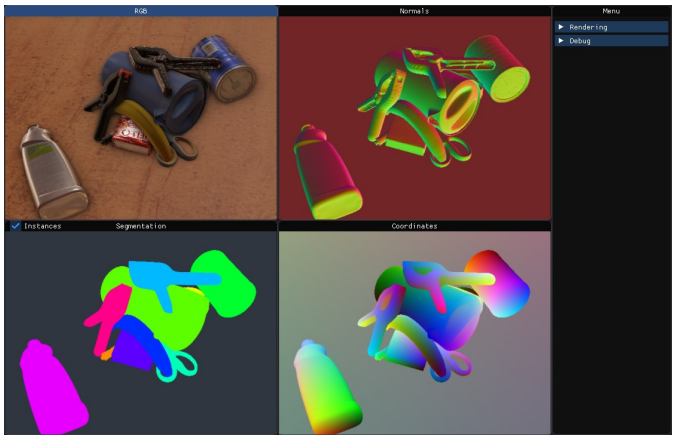


Figure 4.2: Stillleben user interface. On the upper left, the synthetic image can be seen. The other three visualizations depict more data form the renderer, like the corresponding semantic segmentation mask (bottom left). Taken from `https://ais-bonn.github.io/stillleben/`.

that are similar to those in YCB-Video. As for the supervised approach, we use IBL data for the lighting. However, the objects are not embedded into the IBL scenes but backgrounds are randomly selected from the *ObjectNet3D* database (Xiang, Kim, et al. 2016) to obtain more diverse data. While we use the feature to add stickers for the YCB-Video dataset, we do not use it for the *HomebrewedDB* dataset (Kaskman et al. 2019), see Section 5.1.3.

4.2 Supervised Approach

As a proof-of-concept for synthetic-to-real domain adaptation in our setup, we first use a supervised approach. The key idea here is to train a CNN on data pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. x is generated such that it contains the same objects as y with matching poses—raising the need for pose annotations for all used images from the real distribution. Figure 4.1 depicts exemplary image pairs with matching poses. Challenges include the handling of image backgrounds and the simulation of sensor noise. The image backgrounds are particularly challenging, because we do not have the original background data available separately from the images.

4.2.1 Network Architecture and Training Details

For our domain adaptation network, we use the *U-Net* architecture (Ronneberger, Fischer, and Brox 2015), depicted in Figure 4.3. It consists of an encoder part and a decoder part which are nearly symmetric. The corresponding layers are

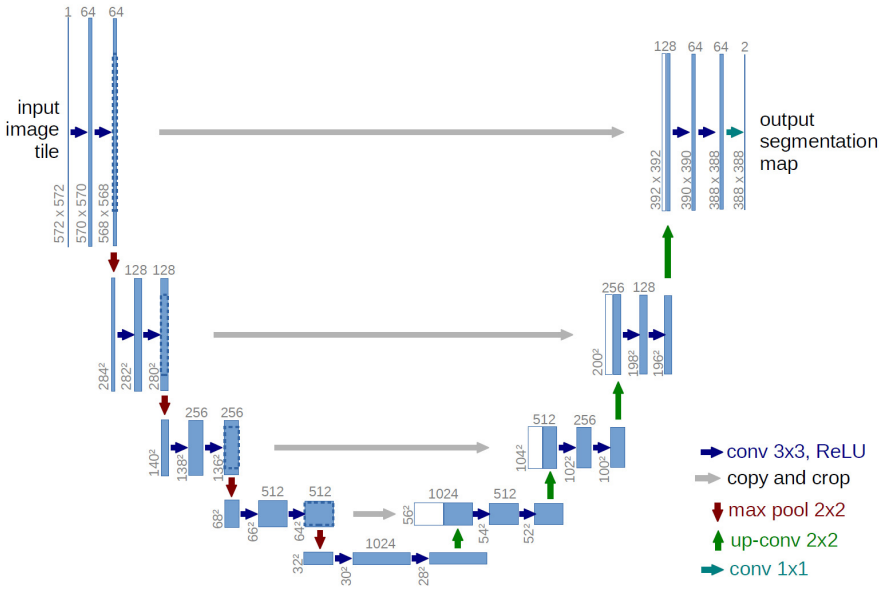
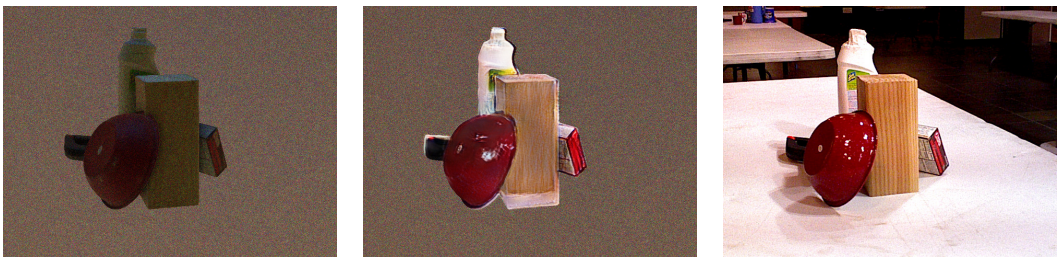


Figure 4.3: The U-Net architecture. Taken from Ronneberger, Fischer, and Brox (2015).

connected by skip connections. Originally, U-Net was developed for semantic segmentation, but its structure does also allow to use it for image translation. U-Net and modified versions of it have already been successfully applied to various image-to-image translation scenarios by for instance Isola et al. (2017) or Kandel et al. (2020). Particularly the skip connections make this architecture suitable for our use case: The refinements we need to apply to tighten the domain gap are on a rather small scale and the overall structure of the images needs to be preserved. The skip connections provide the network with a means to easily learn the identity mapping, complemented with the small-scale refinements.

To train the network, we use the LPIPS distance (Zhang et al. 2018) as our loss function, introduced in Section 2.3.1.

To decide how to apply this loss function, however, is not trivial: While the poses from annotated real images can be exactly transferred to the synthetic images, the backgrounds cannot. Solutions include using no or randomly chosen backgrounds. Here, we embed the rendered objects in a digital environment map (IBL map) to also capture the lighting influence of the environment on the objects. In contrast to for instance object surface properties, the resulting differences in the backgrounds are substantial and there is no inherent correspondence between the background of a synthetic and the respective real image. Thus, it is not feasible to train the network to replace a synthetic background with a consistent real one. For the same reason, using synthetic images with no backgrounds would not be feasible either. This is supported by experiments which yielded severe artifacts in these



(a) (b) (c)

Figure 4.4: Refinement results from the supervised approach. a) shows the synthetic image, b) the version refined using our trained U-Net adaptation network and c) the baseline real image from YCB-Video. The synthetic and the real image are taken from a distinct test set and have not been presented to the network during training.

cases. We therefore train the network to preserve the synthetic background while adapting the foreground. Formally, this translates to

$$\mathcal{L} = \mathcal{L}_{\text{LPIPS}} \left(y \odot \text{sgn} \left(l_{\text{gt}} \right) + x \odot \left(1 - \text{sgn} \left(l_{\text{gt}} \right) \right), \hat{y} \right), \tag{4.1}$$

as our loss function where l_{gt} denotes the ground truth segmentation label for both the synthetic and the real image (x and y, respectively, as before), $\hat{y}$ the network output and where $\odot$ is Hadamard's product.

We train the network for 24 epochs of 10k fixed image pairs using the Adam optimizer with a batch size of 8 and using batch normalization after the convolution blocks. To stabilize the training, we employ gradient clipping. Figure 4.4 shows exemplary results for a synthetic-real image pair taken from a distinct test set. It can be seen that the image's objects visually appear more natural while the background mostly stays the same. Also, the object shapes are accurately preserved—which is important for re-using ground truth annotations from the synthetic input data.

4.2.2 Limitations

In this thesis, we do not further evaluate the presented supervised approach in a quantitative manner or further optimize our method but only briefly demonstrate it as a proof-of-concept. The reason are the strong methodical limitations of this approach: First, the presented technique needs real images with 6D pose annotations. This contradicts our goal of avoiding manual annotation efforts to obtain training data. One possible application of this approach might be as some form of data augmentation or in combination with highly accurate pose estimation

techniques. The second major limitation arises from the kind of adaptation the network has to do: It is trained to separate the foreground objects from the background, to apply an identity mapping to the background and to let the foreground objects appear like in the corresponding real image. This impairs the generalization capabilities with respect to unseen scenes or object arrangements compared to unpaired, unsupervised techniques that have to learn more fundamental concepts from the training data in order to perform domain adaptation.

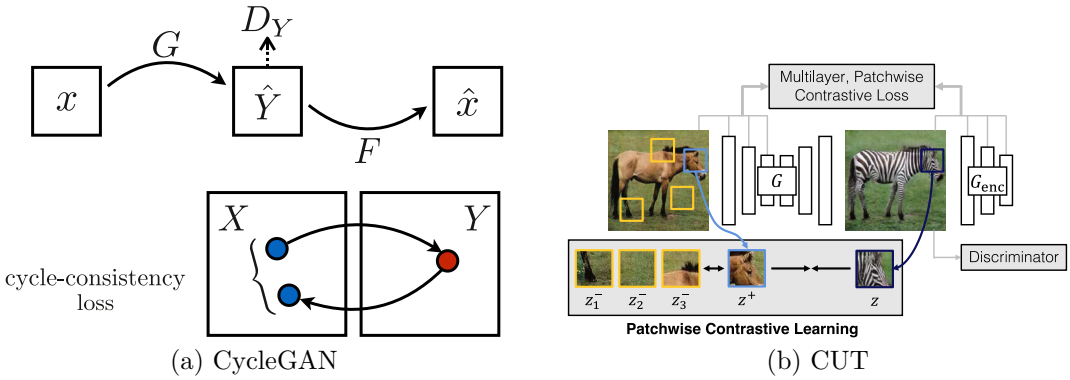
4.3 Unsupervised Approach

To overcome the previously mentioned limitations of the supervised proof-of-concept technique, we propose the unsupervised approach explained in the following. After explaining *Contrastive Unpaired Translation (CUT)* by Park et al. (2020), on which we base our work, we explain our modifications to this technique and how we apply it.

In addition to the challenges that had to be addressed in the supervised setting, ensuring object shape consistency is now a non-trivial but highly important requirement. While the supervised setting strongly guides the network towards preserving object shapes, unpaired translation in general does not constrain the network (enough) towards this. However, strong shape consistency is important for subsequent tasks where we need the annotation data provided by the renderer to still be valid for the refined synthetic images.

4.3.1 Contrastive Unpaired Translation (CUT)

Our domain adaptation approach is largely based on Contrastive Unpaired Translation (CUT) as introduced by Park et al. (2020). It is an image-to-image translation technique, aimed at preserving the image content while adapting the appearance to the target domain, i.e. it performs conditional image generation. CUT is related to the well-known CycleGAN approach by Zhu et al. (2017) which pursues the same objective. Both are GAN-based, can be used for unpaired image sets and have to address the same key issue: Training a GAN for unpaired image-to-image translation is in general an under-constrained task. CycleGAN employs a second GAN for a reverse mapping from the target to the source domain. The method enforces correspondences between input and output image by a cycle-consistency loss that penalizes differences resulting from passing an image through both the forward and the reverse GAN subsequently (and vice versa), see Figure 4.5a). This avoids a collapse of the generator, i.e. mapping all inputs to a single output in the target domain.



(a) CycleGAN (b) CUT

Figure 4.5: CycleGAN and CUT key ideas. a) CycleGAN uses two GANs G and F for a cyclic mapping of input images. The depicted cycle-consistency loss is also applied vice versa, i.e. for images from the target distribution Y. b) CUT uses a single GAN with generator G and ensures content preservation using contrastive learning. Taken from Zhu et al. (2017) and Park et al. (2020), respectively.

CUT uses an alternative way to ensure content preservation and removes the second GAN. Advantages of this include reduced complexity and training time. Besides, the inherent assumption that $f : \mathcal{X} \to \mathcal{Y}$ is a bijection is removed which is helpful in our case with respect to the various and differing synthetic and real lighting settings. The authors of CUT directly compare its performance with CycleGAN and show its superiority, see (Park et al. 2020). Figure 4.5 shows key design ideas of a) CycleGAN and b) CUT. The mentioned alternative way to retain the image content is contrastive learning. The main idea behind this is to learn an embedding that maps associated data points at close distance, in contrast to other, not associated data points and to subsequently use this embedding to guide the training of the main network. Park et al. (2020) apply this idea to image-to-image translation by proposing the *PatchNCE* loss which is based on the *InfoNCE* by Oord, Li, and Vinyals (2018). The intuition here is that a patch of the translated image should have more information in common with the same patch in the source image (positive) than with N other patches from the source image (negatives). More specifically, this is done as follows (a visualization of the calculation can be found in Figure 4.6): A patch from the synthetic source image x is sampled and features extracted from it are passed through a simple MLP. The same is done for the corresponding patch from the translated image and the other N patches from the source image. Based on this, the MLP is trained as a classifier to select the positive sample from the $N + 1$ source patches, given the translated output patch. Park et al. (2020) use cross-entropy as their loss function.

The GAN's generator has an encoder-decoder architecture and thus provides a feature extractor, the encoder part. The aforementioned features are taken from the encoder. To account for different levels of visual complexity, features from specific encoder layers[1] are taken and used for training a distinct MLP per layer. Together, the MLPs' cross-entropy losses form the PatchNCE loss. Technically, the patch sampling works as follows: First, single pixels are chosen as "patches". Deeper in the encoder, the corresponding receptive fields grow, thus increasing the patch size considered for the respective features.

The PatchNCE calculation is visualized in Figure 4.6. Note that the figure depicts our patch-based application of the CUT approach, so the aforementioned patches become subpatches of the patches used for the training of CUT as a whole.

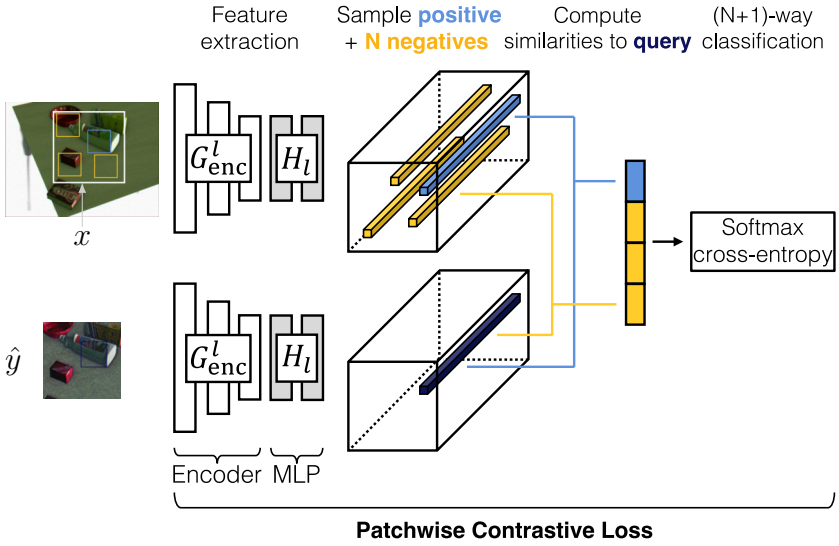


Figure 4.6: The PatchNCE is calculated based on the selected patch x from the synthetic image and the corresponding generated refined image $\hat{y}$. From these smaller images, subpatches are selected for the calculation. Adapted from Park et al. (2020).

Training the MLPs towards embedding source and target patch features close to each other, with patches at different levels of complexity, strongly constrains the GAN towards content preservation. In CUT, the PatchNCE is also calculated for the target image to stabilize the training by hinting the network to keep images from the target domain identical.

Park et al. (2020) combine the PatchNCE with a traditional GAN loss that trains the GAN generator to apply the changes in appearance for—in our case—

[1]These layers can be manually chosen.

domain adaptation. In combination, the following loss function is proposed:

$$\mathcal{L} = \mathcal{L}_{\text{GAN}}(G, D, X, Y) + \lambda_{\text{NCE}} \cdot (\mathcal{L}_{\text{PatchNCE}}(G, H, X) + \mathcal{L}_{\text{PatchNCE}}(G, H, Y)), \tag{4.2}$$

where G denotes the GAN's generator, D the discriminator, X the source image and Y the target image. H denotes the MLPs used for the PatchNCE. Park et al. (2020) suggest to choose $\lambda_{\text{NCE}} = 1$.

4.3.2 Enhancements to Contrastive Unpaired Translation

While CUT has been proposed for translating real images to results that are realistic to the human eye, with input backgrounds more consistent with the foreground than in our case[2] and with more diverse objects than our images have, our setting is different: We translate the images primarily to be used as input to a neural network, use randomly chosen backgrounds and have a highly restricted number of objects and scenes. Therefore, we apply the CUT approach with modifications:

Patch-based Application

Its authors propose to apply CUT to images at full resolution. However, we decided to train it in a patch-based way. There are several reasons for this: First, we have substantial variability in the images produced by Stillleben, especially with respect to the backgrounds, and a large set of real images. It seems sensible to use many images from both distributions for the training in order to achieve good generalization to unseen images. Working at full resolution (640×480 pixels for YCB-Video), however, induces long training times when following the learning duration of 400 epochs proposed by Park et al. (2020) and using a large training dataset. Second, the changes to the source image we aim to achieve are at small scale. Ideally, our domain-adapted images reflect the visual properties induced by the camera used for the real images but are content-wise very close to the source data. Therefore, we do not need to consider full images and—also apart from the higher training cost—doing so might even have a detrimental effect. Experimental results support this motivation. CUT models trained at full resolution often deform relevant objects or hallucinate parts of them in new places, as can be seen in Figure 4.7. While the training is performed on patches, inference is still possible at full resolution, due to the GAN generator's architecture. We thus argue that it is sufficient and even beneficial to work on image patches.

[2]We remind the reader that the background is randomly chosen from ObjectNet3D for each generated image.

(a) synthetic (b) refined

Figure 4.7: A synthetic image and a CUT-refined version of it with CUT trained on full-resolution images. Note how training at full resolution leads to deformations and hallucinations of objects in the background image.

The selected patch size has to be small enough to notably reduce the computation effort and prevent global effects like in Figure 4.7. At the same time, it has to be large enough to still contain sufficient information and to ensure that sampling sub-patches for the PatchNCE is still possible in a meaningful way. We propose and evaluate patch sizes between 60^2 and 160^2 pixels. The patch selection is done by random cropping. Given the generator's architecture and the PatchNCE calculation, patch sizes below $< 100^2$ pixels can be thought to be too small from a theoretical perspective. Experimental evidence, however, shows that other, beneficial effects of small patch sizes can outweigh this, see Chapter 5.

Noise Injection

Synthetic images can imitate image noise, for instance as proposed by Foi et al. (2008). Noise from real cameras, however, exposes properties that are hard to model and can exhibit high variance. It could be the case that at the synthetic-to-real domain adaptation task, it is hard for a GAN to add such noise to an image. We hypothesize that injecting noise within the translation process can be helpful to resolve this. Given the GAN's encoder-decoder architecture, we decided to inject noise directly at the input of the decoder by adding N normally-distributed random feature maps to the M feature maps from the encoder, where $N \ll M$. To ease the noise integration and return to the previous number of feature maps, we add three convolution layers before the actual decoder part. We remark that noise is injected both during training and inference and that the noise is randomly generated for each image.

4.3.3 Training Details

To obtain the refined images, we pass synthetic images through the CUT generator. Before, we train CUT on unpaired sets of synthetic and real images for 400 epochs, following the curriculum suggested by Park et al. (2020). For the training of CUT, we choose a batch size of 40 irrespective of the patch size. This is to keep the sources of variation between the results for different patch sizes limited. The deviation from the standard batch size of 1 for CUT has two main reasons: First, the training time can be substantially reduced while improving memory usage. Second, we argue that for the patch-based application of CUT it is beneficial to use more averaged gradients, especially as some of the randomly sampled patches may consist entirely of irrelevant background information. In order to expose the network only to images that could occur in reality, we do not horizontally flip images for data augmentation as proposed by Park et al. (2020). The network weights are initialized normally. Further deviations from the defaults of the standard implementation are stated in the respective experiment descriptions in Chapter 5. All training of CUT has been performed using NVIDIA A100-SXM4-40GB accelerator cards.

5 Evaluation

In this chapter, we evaluate the unsupervised approach for domain adaptation and the proposed enhancements we presented in Chapter 4.

First, we evaluate it on a semantic segmentation task using two robotics datasets. Subsequently, we give insight into the learned refinement operations.

5.1 Semantic Segmentation

In many applications, the goal of visual domain adaptation is to create images that look appealing to the human eye. As mentioned before, in our robotics use case, we are not actually interested in well-looking images but in images that yield better results on subsequent data processing tasks than the original synthetic images.

Therefore, we evaluate our method on a semantic segmentation task. This evaluation is performed similarly to how it is done by Schwarz and Behnke (2020) for reasons of comparability as we use Stillleben to generate our synthetic images. We choose the YCB-Video dataset (Xiang, Schmidt, et al. 2018) as our main dataset for the evaluation—because Stillleben has been optimized for and evaluated on this dataset, see (Schwarz and Behnke 2020). Besides, YCB-Video provides the necessary ground truth segmentation data as well as pose data for the supervised approach. To demonstrate the broader applicability of our approach we also evaluate the performance on the HomebrewedDB dataset (Kaskman et al. 2019). This dataset is sufficiently distinct from YCB-Video but still related enough to consist of similar arrangements, i.e. of tabletop scenes with small objects.

5.1.1 Evaluation Metrics

For the semantic segmentation task, we use *RefineNet* (G. Lin et al. 2017), an established network architecture for semantic segmentation, and train it from scratch on 450k images, subdivided into 300 epochs of 1500 images. The segmentation performance is evaluated on a test set of annotated real images by calculating the mean *Intersection over Union (IoU)* over all classes present in the dataset. All IoU values in the following are calculated on the respective test sets.

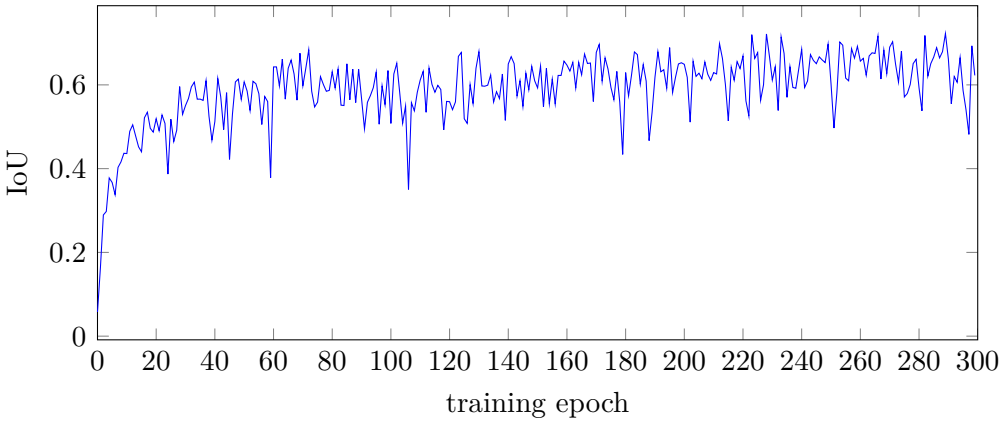


Figure 5.1: The Test IoU on YCB-Video for training on synthetic images over 300 epochs shows significant variance between epochs.

We are interested in the performance on test data after training on three image sets: synthetic images from Stillleben, CUT-refined synthetic images from Stillleben, and real images (disjoint from the test set) for comparison. For the refined images, we use the ground-truth labels provided by the renderer for the corresponding unrefined images. Ground-truth labels for both the training and the test real images are provided as part of the datasets used for the evaluation.

As can be seen in Figure 5.1, the performance on the test data differs significantly between epochs during the training of RefineNet. In the following, we therefore always visualize the distribution of IoU values over the 50 last training epochs instead of just indicating the IoU value for the final epoch.

5.1.2 Results on YCB-Video

For the prior training of CUT, we use 10k images generated using Stillleben and 10k images from the training set of YCB-Video.

Patch Size

The first investigated aspect is the patch size, which we choose between 60^2 and 160^2 pixels as mentioned in Section 4.3. It is worth noting that internally CUT works with patch sizes that are multiples of 4. In other cases, patches are rescaled to have such size using bicubic interpolation. While it appears possible that this interpolation would introduce some beneficial or detrimental smoothing to the input images, we saw no consistent effect on the results. We evaluate the performance for the following patch sizes: 60^2, 70^2, 90^2, 100^2, 120^2, and 160^2 pixels. The results are depicted in Figure 5.2, alongside the results for RefineNet trained

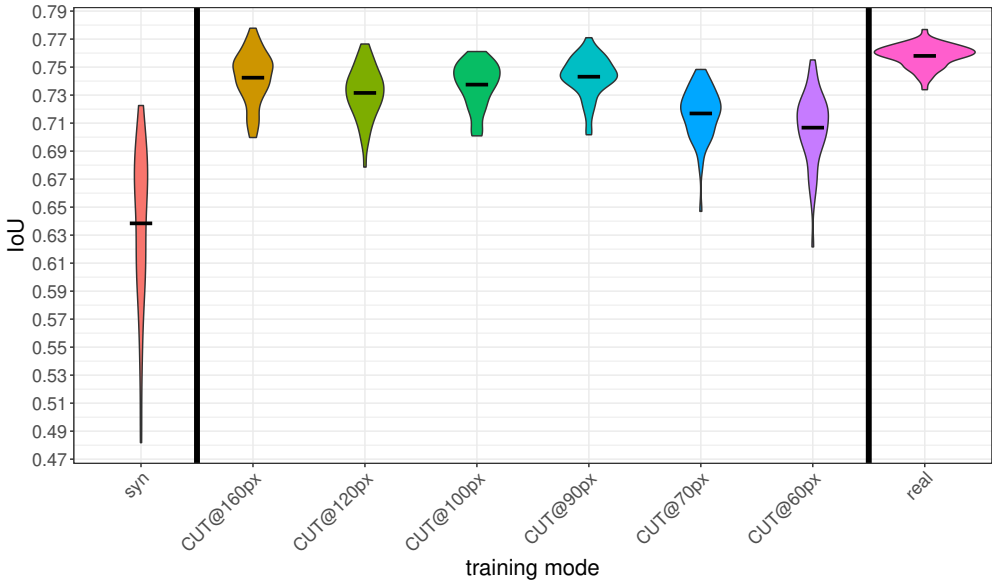


Figure 5.2: Results on YCB-Video. The test IoU distribution over the last 50 training epochs for CUT-refined images for CUT patch sizes 160^2, 120^2, 100^2, 90^2, 70^2, and 60^2. The leftmost plot depicts the test results for training with synthetic images, the rightmost plot for training with real images. Training with CUT-refined synthetic images not only yields higher IoU values than pure synthetic images but also narrower distributions.

on purely synthetic and real YCB-Video images. It can be seen that especially at patch size 60^2, but to some degree still for 70^2, only insufficient information is conveyed for this dataset compared to the larger patch sizes. 160^2 and especially 90^2 yield the best results, as can also be seen in the numerical results in Table 5.1. Given the fact that the training time for CUT largely depends on the patch size (see Table 5.2), 90^2 seems to be the best trade-off. Using CUT-refined synthetic images offers a significant benefit over using pure Stillleben images and shows segmentation performance that is close to the level yielded by training on real data.

Apart from the general performance increase, CUT-refining the synthetic images offers another benefit: Irrespective of the patch size, refining the images leads to a significantly narrower distribution of the IoU values, closer to real data, which yields the narrowest distribution. In contrast, the variance is rather high for training on synthetic images. For practical use, a narrower distribution is helpful because the performance after the training can be estimated more reliably and does not depend too much on the exact epoch at which we stop the training. As for the mean IoU value over the last 50 epochs, a patch size of 90^2 pixels also yields the narrowest distribution in terms of the standard deviation, see Table 5.1.

Table 5.1: Results on YCB-Video.

Training mode	Mean IoU (↑)	SD[1] (↓)	Rel. to Real (↑)	Rel. to Syn. (↑)
synthetic	0.638	0.0560	0.842	—
CUT_{160px}	0.742	0.0187	0.979	+16.3%
CUT_{120px}	0.732	0.0187	0.966	+14.7%
CUT_{100px}	0.737	0.0161	0.972	+15.5%
CUT_{90px}	**0.743**	**0.0145**	**0.980**	**+16.5%**
CUT_{70px}	0.717	0.0186	0.946	+12.4%
CUT_{60px}	0.707	0.0257	0.933	+10.8%
real	0.758	0.0081	1.000	+18.8%

[1] SD denotes the standard deviation.

Table 5.2: Patch Size & Training Epoch Time

Patch size [px]	Time [s]
160×160	181
120×120	110
100×100	82
90×90	66
70×70	54
60×60	43

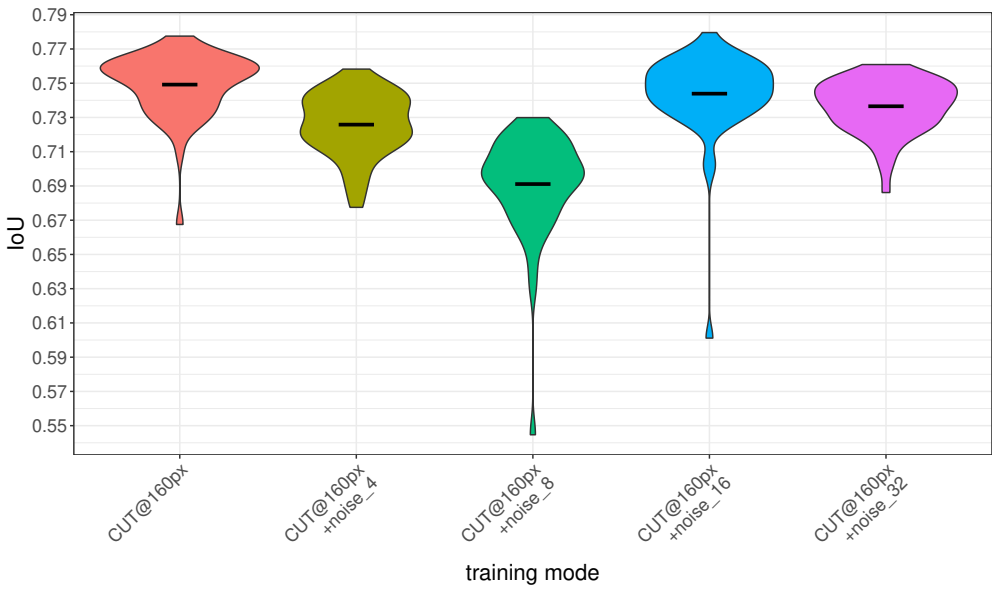


Figure 5.3: Noise injection. The test set IoU distributions for 0, 4, 8, 16, and 32 injected random feature maps show no beneficial effect of injecting noise.

Noise Injection

For the noise injection described in Section 4.3.2, we tested injecting 0, 4, 8, 16 and 32 random feature maps at a patch size of 160^2 pixels. The results can be seen in Figure 5.3. While 8 feature maps appear to even have a detrimental effect across multiple runs of this experiment, the general impression is that injecting noise is not beneficial. This contradicts the hypothesis that additional randomness apart from the noise added by Stillleben's camera model is helpful for the GAN to closer match the real image distribution. We hypothesize that the artificial image noise from Stillleben is sufficient to produce images that cannot too easily be distinguished from real images by the GAN discriminator based on the kind of present noise.

Exponential Moving Average for RefineNet

Another change, however, is beneficial but not directly related to CUT: Using an *exponential moving average (EMA)* of the RefineNet model parameters for the evaluation on the test set significantly improves the performance and reduces the variability of the IoU (decay factor: 0.995). This is consistent over all patch sizes for CUT-refined images as well as synthetic and real images, as can be seen in Figure 5.4. We therefore use this modification in the following for the experiments with HomebrewedDB and for the further experiments in Chapter 6. For later

comparison with HomebrewedDB, we give the corresponding numerical results in Table 5.3.

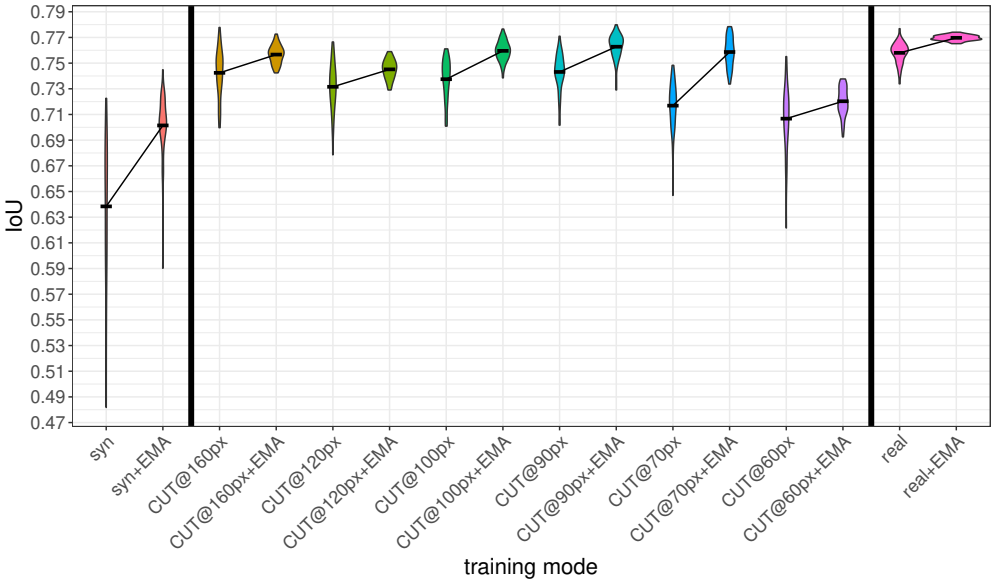


Figure 5.4: Using an exponential moving average (EMA) for the RefineNet parameters improves the performance and reduces the IoU variability.

Table 5.3: Results on YCB-Video (with EMA).

Training mode	Mean IoU (↑)	SD[1] (↓)	Rel. to Real (↑)	Rel. to Syn. (↑)
synthetic+EMA	0.701	0.0247	0.910	—
CUT_{160px}+EMA	0.757	**0.0073**	0.983	+8.0%
CUT_{120px}+EMA	0.745	0.0077	0.968	+6.3%
CUT_{100px}+EMA	0.760	0.0078	0.987	+8.4%
CUT_{90px}+EMA	**0.763**	0.0097	**0.991**	**+8.8%**
CUT_{70px}+EMA	0.759	0.0117	0.986	+8.3%
CUT_{60px}+EMA	0.720	0.0110	0.935	+2.7%
real+EMA	0.770	0.0021	1.000	+9.8%

[1] SD denotes the standard deviation.

5.1.3 Results on HomebrewedDB

We also evaluate our approach on the HomebrewedDB dataset (Kaskman et al. 2019) to demonstrate the broader applicability of the presented approach. Both datasets consist of small objects placed on a table. However, the overall image appearance and the presented scenes differ significantly. We do not use the plain HomebrewedDB dataset but the data offered in the *BOP* challenge[1]. For both the offered test images (*BOP'19/20 test images (Primesense)*) and the validation images (*Validation images (Primesense)*), ground truth semantic annotations are included. No real training images are offered. However, 3D meshes for all objects are available, which we need for Stillleben.

We restrict our evaluation to the subset S of HomebrewedDB objects which are present in the test images. We generate synthetic scenes with objects from S and train CUT on validation images containing objects from S. The segmentation model trained on real data is trained analogously. In all cases, the official test set is used to evaluate the performance of the trained RefineNet models.

For generating the synthetic images using Stillleben, we make two slight modifications compared to the process proposed for YCB-Video by Schwarz and Behnke (2020), based on the appearance of the real images: We remove the stickers randomly added to the objects and instead of rendering the objects on a textured table, a white table is used. Without any further adaptation to HomebrewedDB, we reach IoU values around 0.48 which are inferior to those achieved for YCB-Video. The achieved IoU for real images is slightly lower than for YCB-Video, see Figure 5.5 and Table 5.4.

For the training of CUT, we use 10k synthetic images and all 1020 real validation images. Obviously, most real images are presented multiple times during each epoch. Due to our patch-based training, the shown part of the image, however, varies. We restrict ourselves to patch sizes of 70^2 and 90^2 pixels. The general impression when looking at the images from both datasets is that the objects typically take up less room in the images in HomebrewedDB compared to YCB-Video.

With the same hyperparameter choices as for YCB-Video, we saw segmentation performance inferior to that of pure Stillleben. Looking at the produced images, the reason is a mode collapse: Most of the images are grey-textured, with the object shapes barely visible. This is consistent over multiple runs and patch sizes. A reason might be that for this dataset and the respective synthetic images produced by Stillleben, the loss weighting insufficiently ensures content preservation. Therefore, we propose to increase the weight of the PatchNCE while keeping it low enough to allow for meaningful changes to the appearance. The results for

[1] https://bop.felk.cvut.cz/datasets/

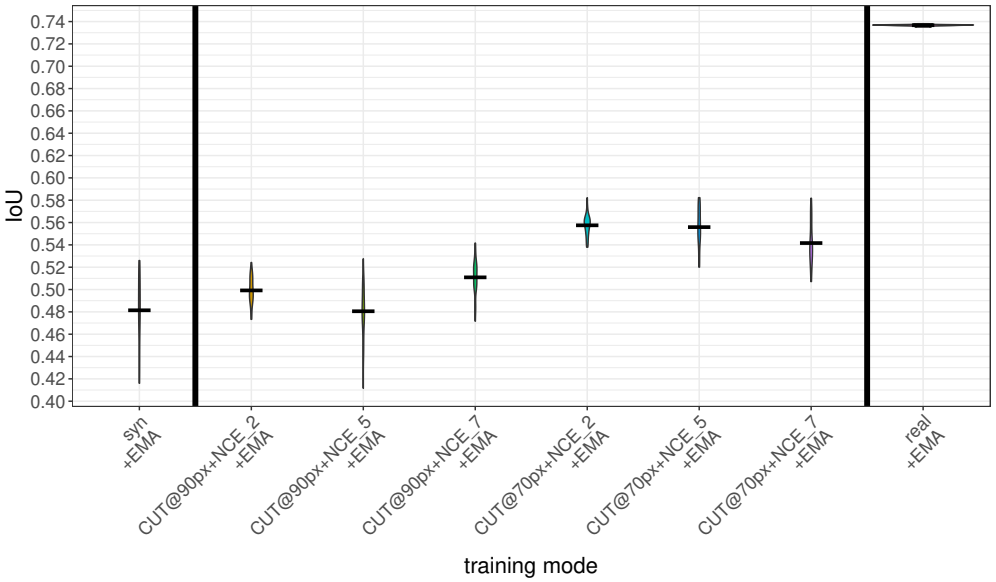


Figure 5.5: Results on HomebrewedDB. We show the test IoU distribution over the last 50 training epochs for CUT-refined images for CUT patch size 90^2 and $\lambda_{\text{NCE}} = 2,\ 5,\ 7$ as well as for patch size 70^2 with the same values for λ_{NCE} (from left to right), complemented by the results for training on synthetic (left) and real images (right).

$\lambda_{\text{NCE}} = 2,\ 5,\ 7$ can be seen in Figure 5.5 and Table 5.4 for both patch sizes considered. A modest increase of $\lambda = 2$ appears to be favorable, as does a patch size of 70^2 pixels—with regards to the IoU and its variability as well. Thus, combined with the theoretical reasoning from above, we do not further evaluate larger patch sizes. With these changes we see a significant improvement compared to the results using raw Stillleben images, quantitatively larger than for YCB-Video with EMA by a margin in relative terms, see Table 5.4 and Table 5.3. Not only the IoU is increased by CUT-refining the synthetic images, but also the IoU variance over the training epochs of RefineNet is reduced—consistent with what we see for YCB-Video.

Based on these results, we conclude that our approach is applicable also beyond YCB-Video with only minor changes needed for a related dataset. Further optimization of Stillleben for HomebrewedDB has the potential to achieve segmentation performance on refined synthetic data comparable to the performance on real data for this dataset as well.

Table 5.4: Results on HomebrewedDB (with EMA).

Training mode	Mean IoU (↑)	SD[1] (↓)	Rel. to Real (↑)	Rel. to Syn. (↑)
synthetic	0.481	0.0304		—
$\text{CUT}_{90\text{px},\lambda_{\text{NCE}}=2}$	0.499	0.0123	0.677	+3.7%
$\text{CUT}_{90\text{px},\lambda_{\text{NCE}}=5}$	0.481	0.0223	0.653	+0.0%
$\text{CUT}_{90\text{px},\lambda_{\text{NCE}}=7}$	0.511	0.0150	0.693	+6.2%
$\text{CUT}_{70\text{px},\lambda_{\text{NCE}}=2}$	**0.558**	**0.0094**	**0.757**	**+16.0%**
$\text{CUT}_{70\text{px},\lambda_{\text{NCE}}=5}$	0.556	0.0173	0.754	+15.6%
$\text{CUT}_{70\text{px},\lambda_{\text{NCE}}=7}$	0.542	0.0193	0.735	+12.7%
real	0.737	0.0006	1.000	+53.2%

[1] SD denotes the standard deviation.

5.1.4 Combination with Real Training Data

Until now, we considered the case where we have no real training data and aim to enhance the usefulness of synthetic data. However, there also might be cases where we have real training data available but the achieved performance is not good enough. In their experimental setup, Schwarz and Behnke (2020) have shown that it is beneficial to train RefineNet on synthetic and real data at the same time by randomly choosing the mini-batches from both datasets. One might wonder whether the use of CUT-refined images enhances this effect. The results for both YCB-Video and HomebrewedDB are depicted in Figure 5.6. Consistent across both datasets, the achieved IoU on the respective test sets is higher with refined images than without. However, the effect is less pronounced than for combining real with purely synthetic data. From this, we hypothesize that using synthetic data has a regularizing influence on the training with real data, namely that the learned features are more domain-invariant. This effect is smaller for synthetic images refined towards more realism.

5.2 Learned Refinement Operations

As pointed out before, we are mainly interested in achieving good segmentation performance in order to improve the value of synthetic training data for robotic applications. Still, it is worth analyzing what CUT is actually doing to the synthetic images. Figs. 5.7a) and b) show six synthetic images based on the YCB-Video objects and their CUT-refined versions, respectively.

While some refined images look more realistic, some do actually not seem re-

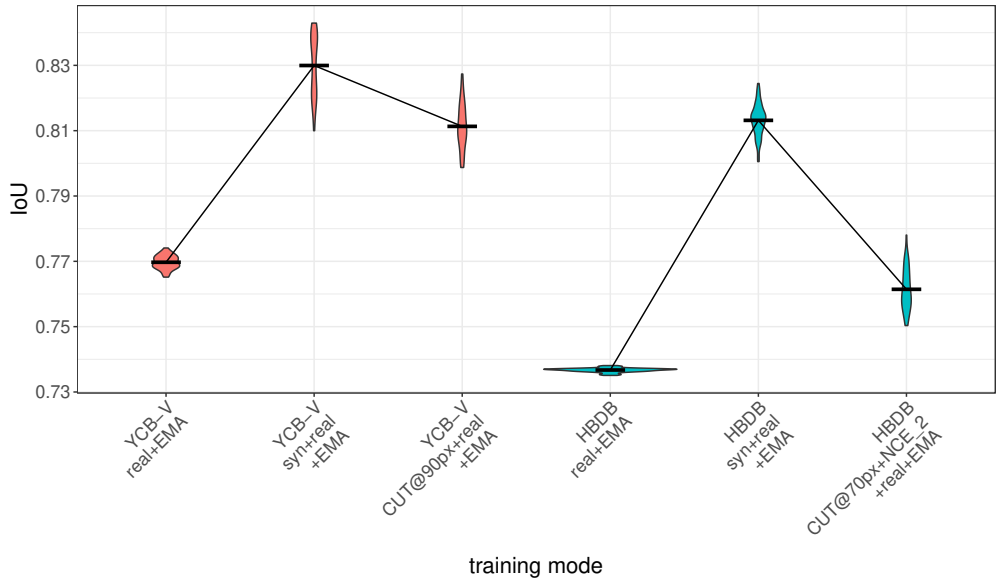


Figure 5.6: Mixing synthetic and real data. We show the test IoU distribution over the last 50 training epochs for real training images, real and synthetic images mixed, as well as real and refined synthetic images mixed, for YCB-Video and HomebrewedDB, respectively.

alistic to the human eye. Still, we achieve generalization to real data on the YCB-Video segmentation task that is close to what we get when training on real data. Hence, we hypothesize that—even if not for the human eye—refining the images aligns the synthetic and real image distributions more closely in the feature space of a CNN like the CUT generator.

5.2.1 Analysis using the LPIPS distance

In a first step, we use the LPIPS distance metric already introduced in Section 2.3.1 and used in Section 4.2 to compare the image distributions quantitatively. Specifically, we use the YCB-Video keyframes, render corresponding Stillleben images using the pose annotations and refine them using a trained CUT generator. Like before, we calculate the LPIPS distance using features extracted from AlexNet (Krizhevsky, Sutskever, and G. E. Hinton 2012) pre-trained on ImageNet[2]. To study the effect of our image refinements, we compare the mean LPIPS distance (in the following: mLPIPS) of synthetic and real images with the mean distance of CUT-refined synthetic and real images, where the mean is calculated over the YCB-Video keyframes.

[2]https://www.image-net.org/index.php

(a) synthetic

(b) refined

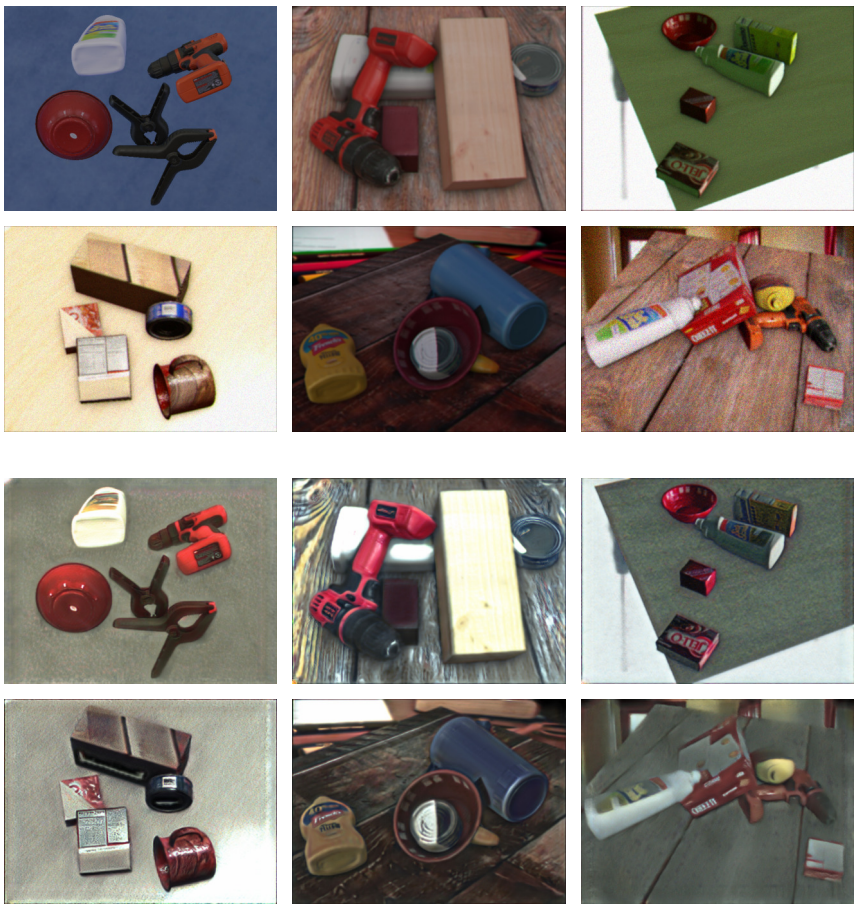


Figure 5.7: Synthetic images and their CUT-refined versions.

Using a CUT network trained at a patch size of 90^2 pixels, we obtain the following results (lower values are better): The mLPIPS for synthetic and real data is 0.72833961 compared to 0.70913235 for CUT-refined and real data (improvement: +2.7%). Thus, the perceptual distance is slightly decreased by refining the images. The distribution around the mean value is visualized in the left half of Figure 5.8. It can be seen that it is narrower when the refinements are applied and more pronounced below the mean value, again compared to pure synthetic data.

As described before, we cannot fully replace the synthetic backgrounds with real ones in our domain adaptation setup. Thus, we also compare the images with the backgrounds masked out, using the ground truth segmentation label from the Stillleben renderer[3]. The resulting mLPIPS values are 0.1859759 and 0.18570992 for synthetic and CUT-refined images, respectively, showing only a very slight

[3]We use this segmentation mask for all three images, as by definition the segmentation masks provided for the YCB-Video keyframes are identical.

improvement by the applied refinements (+0.1%). Correspondingly, the LPIPS distributions as depicted in the right half of Figure 5.8 are rather similar, with a slightly vertically larger broad region below the mean value for the refined images.

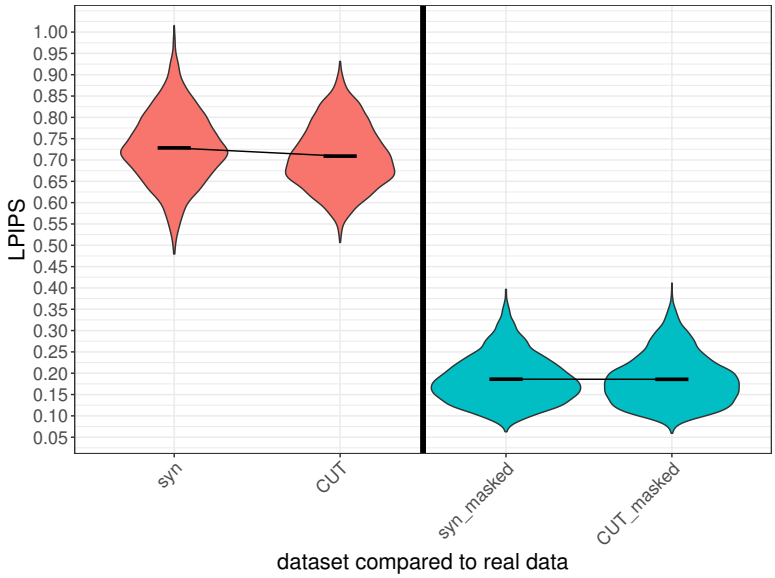


Figure 5.8: LPIPS distance distribution between synthetic and real images alongside the corresponding distribution for CUT-refined and real images. The distributions are calculated for the YCB-Video keyframes with synthetic images with corresponding poses. Lower values are better. For the full images (left), we see an improvement after applying the refinements. This improvement is almost not present when masking out the backgrounds (right).

Summing up, the analysis using LPIPS distance values points towards a reduced domain gap after applying refinements using CUT. However, the seen improvements are not particularly large which does not correspond to the drastically improved results in the semantic segmentation task described before. We therefore hypothesize that a single scalar value like the LPIPS, calculated using a fixed network trained on a dataset with strongly differing content might not be sufficient to capture the actual changes by the refinements. To overcome this, we analyze the image distributions more closely in the following part.

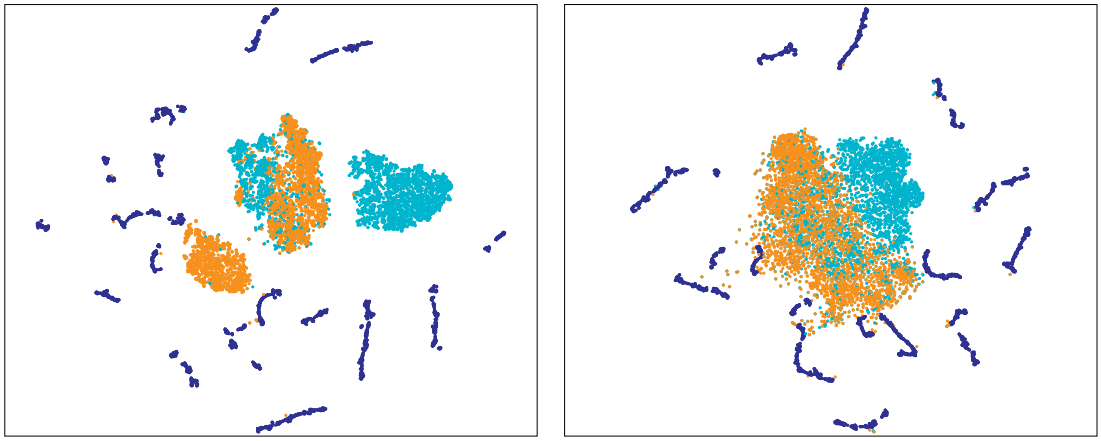
5.2.2 Analysis using t-SNE embeddings

To analyze the synthetic, CUT-refined and real image distributions, we employ t-SNE embeddings (see Section 2.3.2) to project CNN features of the actual images into two dimensions. This is done, because we cannot investigate the image distri-

butions in the high-dimensional feature space of a CNN directly. For the reason to use CNN features and not image data directly, see Section 2.3.2. Like before, we create triples of real YCB-Video keyframes, Stillleben images and CUT-refined versions thereof. Also as done previously, we employ a CUT generator trained at a patch size of 90^2 pixels. For the resulting set of triples, we calculate the feature maps of one extraction layer of RefineNet trained on real YCB-Video images. We apply adaptive average pooling with output size 1×1 to each feature map to obtain a vector of scalar values. Based on these vectors for all images, we calculate the t-SNE embeddings.

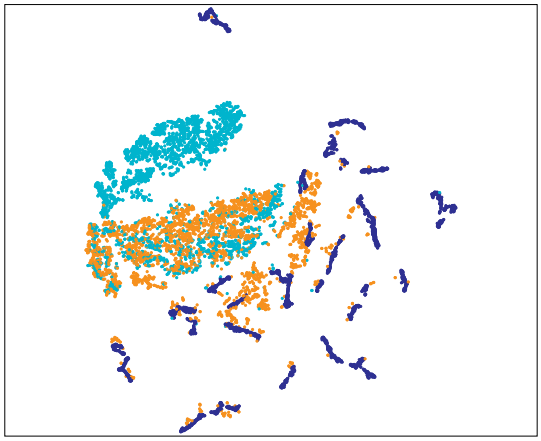
Figs. 5.9a) and b) depict embeddings based on features from an early extraction layer and a late extraction layer, respectively. It can be seen that subsequent keyframes of the real YCB-Video video sequences are closely aligned. Besides, the synthetic images appear to form two clusters—especially for the early extraction layer—for which it was not possible to reliably determine their origin. We hypothesize that this split might be an artifact introduced by the tendency of t-SNE embeddings to form clusters, see (Wattenberg, Viégas, and I. Johnson 2016). We see a slight shift into the direction of the real samples introduced by the CUT refinements in a) and b), which is more apparent when inspecting which synthetic data point corresponds to which refined and real data point. Some samples are even shifted more or less exactly to the corresponding real samples in the embeddings.

As mentioned for the LPIPS, we are more interested in what happens to the appearance of the objects in the images rather than in the backgrounds. Therefore, we also calculate t-SNE embeddings for all image triples with the backgrounds masked out, again using the segmentation labels provided by the Stillleben renderer. The results are depicted in Figs. 5.9c) and d) for an early and a late extraction layer, respectively. The general impression is that CUT-refining the images both spreads the distribution and also aligns the distribution closer to the real image distribution for the early extraction layer (see c)). For several images, the refined synthetic images are embedded quite close to the corresponding real images—an effect that has been seen in a) but is much more pronounced for the masked images. Besides, the two clusters visible in the synthetic data are dispersed to some degree. In later extraction layers of RefineNet, the effect of better alignment with real data is less clearly visible but still present, see d). We saw similar effects to the ones described here for the early layers of AlexNet trained on ImageNet, supporting the hypothesis that the domain adaptation actually does align the distributions more closely.

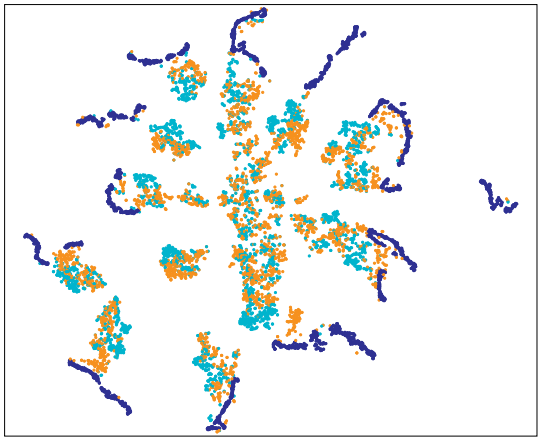


(a) early unmasked

(b) late unmasked



(c) early masked



(d) late masked

Figure 5.9: t-SNE embeddings for different extraction layers of RefineNet, based on images without (a, b) and with (c, d) masked backgrounds (turquoise: synthetic, orange: refined synthetic, blue: real images).

6 Domain Adaptation Beyond Synthetic-To-Real

The main topic of this thesis is synthetic-to-real domain adaptation. However, our primary goal is to tighten the domain gap between synthetic and real images to improve the generalization to real data in subsequent deep learning tasks. In the following, we outline different further ways to apply the approach presented in Section 4.3 and modifications to this approach that provide alternative means to reduce the domain gap. A detailed analysis and optimization of said ideas would go beyond the scope of this thesis. Still, our ideas and discussions are backed by some initial experiments.

6.1 Real-To-Synthetic Domain Adaptation

Performing synthetic-to-real domain adaptation raises the question whether real-to-synthetic domain adaptation might be a task that is easier to learn for a GAN. The reason for this is the fact that real images have more complex properties than synthetic images. Thus, it can be hypothesized that it would be easier for a network to "unlearn" these properties compared to adding them to an image.

For subsequent deep learning tasks (i.e. semantic segmentation in this thesis), the workflow has to be changed slightly: At training time, we can use the unmodified synthetic images instead of refined images. At inference time, the real images need to be passed through the CUT generator before the actual inference on the trained task network is performed. This implies faster training of the task network but at the same time also increased runtime and latency for the inference. In real-time applications like robotic manipulation, this has to be carefully considered and balanced with the achieved network performance.

Experiments with this inverse setting, however, yield results on the subsequent semantic segmentation task inferior to the original approach, consistently across patch sizes and independent of adapted hyperparameters for the training of CUT. Unpublished work on a similar field of application by a research group known to us also points into the direction that—in spite of the intuition outlined above—real-

to-synthetic domain adaptation might be a harder task to learn. We hypothesize that a part of the explanation in our setting is that training the segmentation network on refined images encourages better generalizing features to be learned, due to more variability in the data compared to pure synthetic images. This would be consistent with the wider spread distributions of refined images seen in Section 5.2.2. Besides, any variability introduced by the refinements is only added to the test set in the real-to-synthetic setting, which are few images compared to the 300k frames of synthetic data used for the training that are refined in the synthetic-to-real setting.

In combination with semantic segmentation, it is even possible to go a step further than pure domain adaptation (see Figure 6.1 for a visual intuition): In the real-to-synthetic case, we have the segmentation labels for the target image distribution available or alternatively, we can generate synthetic images without backgrounds. So one can even train CUT to refine the relevant objects to look more like in synthetic images and to remove the real backgrounds. Corresponding training data for the subsequent segmentation network is available as synthetic images without backgrounds can be used. As described for the supervised approach in Section 4.2, the real and synthetic image backgrounds largely differ and also using CUT, it is not possible to actually fully replace the synthetic with real backgrounds or vice versa in a learning-based manner. Removing backgrounds, however, appears easier. The main (theoretical) benefit of this modification therefore, is the fact that the semantic segmentation network does not need to generalize from synthetic to real backgrounds.

Initial experiments with this setting were performed at full resolution as the patch-based approach would be presented with too much content-less (i.e. black) area otherwise, destabilizing the training. The obtained results are promising, but still expose strong artifacts, as can be seen in Figure 6.2: While some objects are preserved and cut out accurately as intended, some objects have holes (a)) or are only partly retained (b)). Besides, parts of the backgrounds are still in the images after CUT is applied. Further research would be required to obtain better quality results.

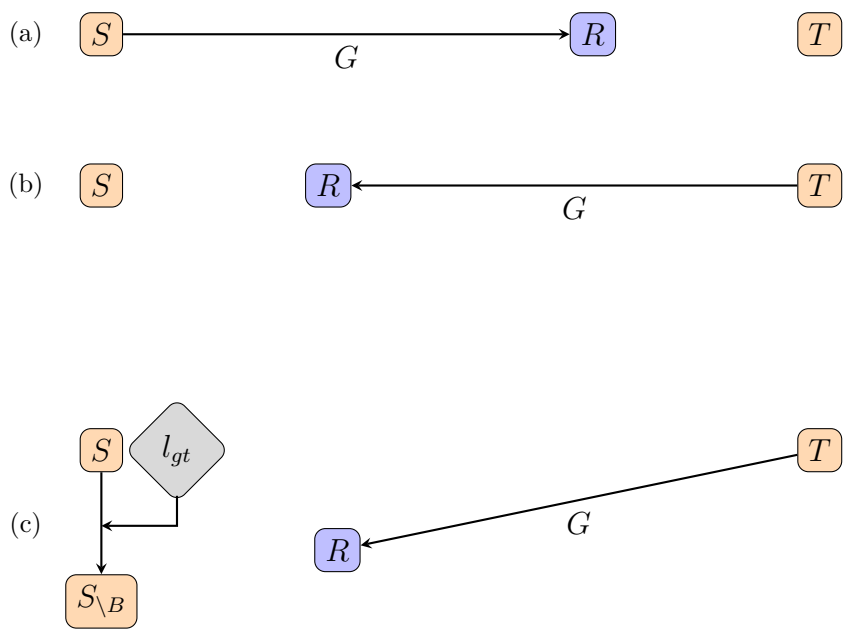
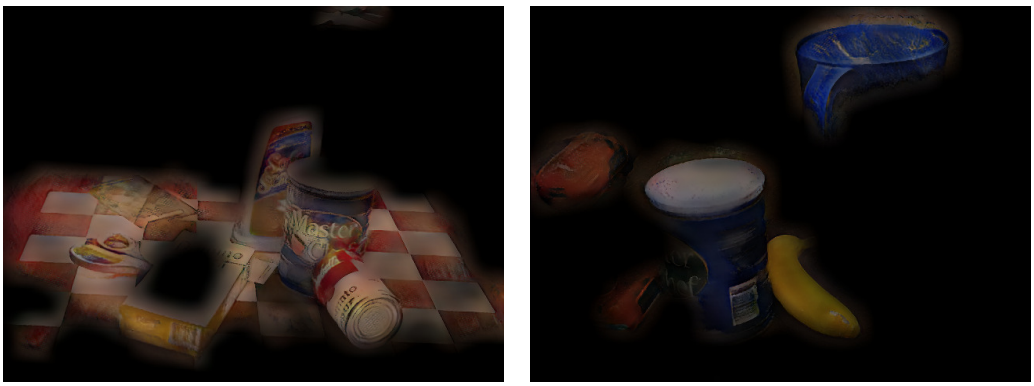


Figure 6.1: Domain Adaptation Schemes. S denotes the source (synthetic) image distribution, T the target (real) and R the refined image distribution and G is the generator. The distance between S and T represents the domain gap. a) In synthetic-to-real domain adaptation, the synthetic images are refined to look more like the images from the target distribution. b) In the inverse real-to-synthetic setting, the images from the real distribution are refined, i.e. mapped towards S. c) In the extended real-to-synthetic case, the synthetic ground truth labels l_{gt} are used to get the source distribution without background $S_{\setminus B}$. Now, the images from T are mapped towards $S_{\setminus B}$.



(a) (b)

Figure 6.2: Real YCB-Video frames after being passed through a CUT generator trained to perform real-to-synthetic domain adaptation, with synthetic images without backgrounds. The general idea of removing the backgrounds works, but in some locations too much or too few is removed, resulting in strong artifacts.

6.2 Real and Synthetic Domain Adaptation—Towards a Common Embedding

As mentioned earlier, we aim for good performance on subsequent deep learning tasks, i.e. in this thesis semantic segmentation. This allows us to relax the idea of synthetic-to-real or real-to-synthetic adaptation: Instead of narrowing the domain gap from one side, we can also train CUT to learn a common embedding for images from both the synthetic and the real image distribution. This would reduce the generalization effort for the segmentation network.

We remind the reader of the optimization objective for CUT: A GAN loss is combined with the PatchNCE applied to refined images from the synthetic image distribution (source distribution) and to real images (target distribution), where the latter is done to stabilize the training by encouraging an identity mapping for real images, see (Park et al. 2020). For the GAN loss, the discriminator is trained on refined synthetic and unchanged real images.

Including the PatchNCE on real images in the objective allows the generator to also meaningfully process real images. Thus, we can change the discriminator training such that it learns to distinguish the generator output for synthetic images from the output for real images. Figure 6.3 depicts the modified setup. We hypothesize that introducing this symmetry into the generator training helps to find a common embedding for both image distributions while the PatchNCE still sufficiently constrains the generator's output content-wise.

First experiments with this change yield results on the subsequent semantic segmentation task that overall do not improve on what is obtained with the synthetic-to-real approach as evaluated in Chapter 5, the effect is rather detrimental. However, for patch sizes that previously yielded particularly weak results, these are improved, as can be seen in Figure 6.4. We conclude, that this modification to the training for itself is not beneficial yet but with further optimization might be able to provide convincing results, possibly even more consistently over various patch sizes.

A possible improvement in this context would be an additional loss component derived from a semantic segmentation network that is trained in parallel on the results of CUT after a first, initializing training phase, inspired by the work of Bousmalis et al. (2017). This would ensure that we obtain results that are feasible for a CNN to be segmented. For different subsequent tasks, corresponding loss networks could be used.

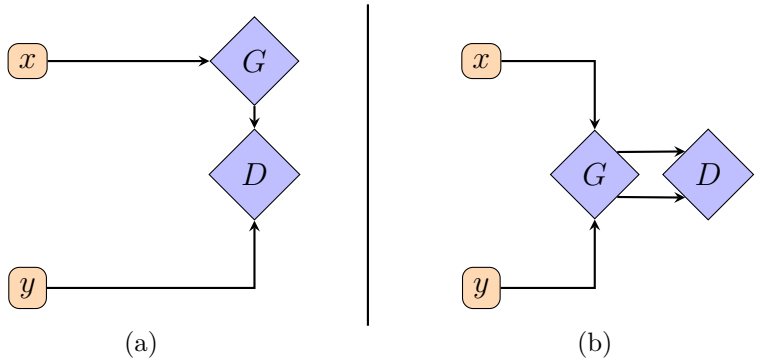


Figure 6.3: Towards a common embedding. a) In the basic setup, the discriminator D is trained to distinguish synthetic images x that have been refined using the generator G from real images y. b) We propose to train D symmetrically on images, with both x and y being passed through the generator G.

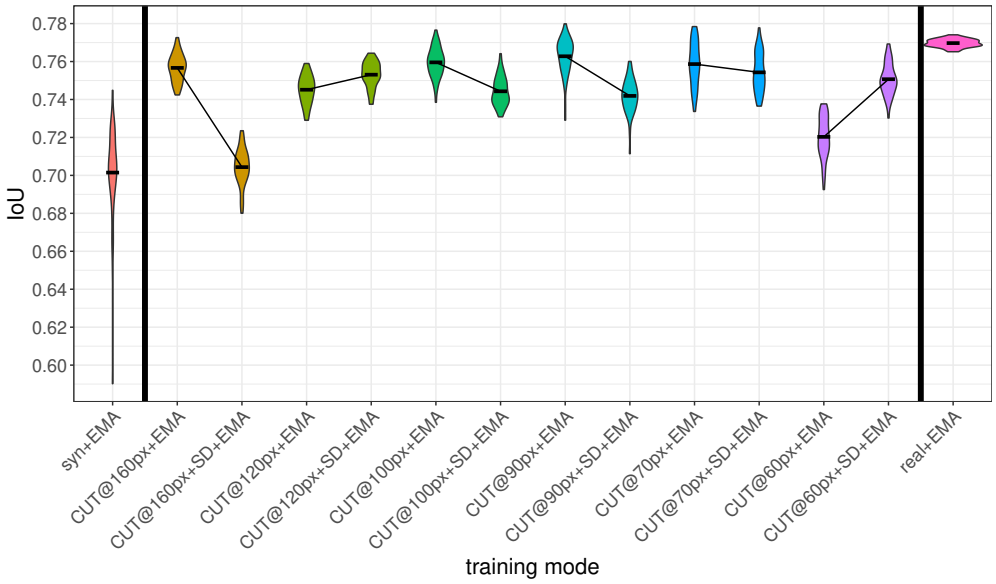


Figure 6.4: Training the CUT discriminator symmetrically (in the plot: *+SD*) yields mixed results. For some patch sizes the IoU is improved while it is lower for others. Overall, the best results are still achieved with a non-symmetrically trained discriminator. Depicted is again the test IoU distribution over the last 50 training epochs, as in Chapter 5.

7 Conclusion and Future Work

In this chapter, we summarize the results of our work. Finally, we point out possible future work, based on where the scope of this thesis ends.

This thesis has—in combination—presented a pipeline for semantic segmentation of RGB images based on 3D object models and un-annotated real images that performs nearly as good on semantic segmentation as a model trained on annotated real images. This pipeline consists of the Stillleben library for generating synthetic training images, an unsupervised domain adaptation technique and a subsequent segmentation network trained using only the refined, domain-adapted synthetic images.

The core scientific contributions of this thesis have been the patch-based application of CUT to domain adaptation and its structured evaluation, complemented by the proposed extensions and modified application scenarios. Besides, we have presented a proof-of-concept supervised domain adaptation approach.

We have shown that the proposed unsupervised technique is able to greatly improve the segmentation results obtained from synthetic data for two robotics datasets. While for YCB-Video, this is sufficient to nearly reach the results achieved with real annotated training data, the results for the HomebrewedDB dataset are comparable in relative terms but not yet as good in absolute terms.

In conclusion, we have demonstrated for our application domain that, given good synthetic data, unsupervised domain adaptation is a viable method to remove the need for real annotated training data.

The main reason for the shortcomings in the results on HomebrewedDB is the lower achieved baseline segmentation performance for the Stillleben-generated synthetic images containing objects from this dataset. A possible extension point would thus be to optimize the Stillleben library for this and possibly more datasets and investigate how well the results seen for YCB-Video can be replicated.

Particularly interesting for future work are the modifications to our approach explained in the previous chapter. Based on our preliminary results, the most promising approach would be to find a common embedding for synthetic and real images and train subsequent task networks on this representation. Key challenges in here include preserving the synthetic ground truth labels and constraining the optimization into a beneficial direction.

To address the mentioned challenges, but also separately in the synthetic-to-real context, another possible point to extend the proposed method would be to include a semantic segmentation loss component in the domain adaptation training. More generally, it appears interesting to investigate task-specific loss components.

Our evaluation on semantic segmentation has been focused on the RefineNet architecture, in line with the evaluation of Stillleben by Schwarz and Behnke (2020). It might be of interest how our results translate to training of other, state-of-the-art segmentation networks. In a broader sense, also the effect of the learned refinement operations on the training of different deep learning tasks like classification or 6D pose estimation could be investigated.

Based on our results, we believe that visual domain adaptation is an interesting and promising research area for robotic applications, with the potential to bridge the domain gap between synthetic and real data and thus to remove the need for large annotated datasets in perspective.

List of Figures

List of Tables

Bibliography

Baevski, Alexei, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli (2022). "data2vec: a general framework for self-supervised learning in speech, vision and language." In: *arXiv preprint arXiv:2202.03555*.

Bardes, Adrien, Jean Ponce, and Yann Lecun (2022). "VICReg: variance-invariance-covariance regularization for self-supervised learning." In: *ICLR 2022-10th International Conference on Learning Representations*.

Bousmalis, Konstantinos, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan (2017). "Unsupervised pixel-level domain adaptation with generative adversarial networks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 95–104.

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

Foi, Alessandro, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian (2008). "Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data." In: *IEEE Transactions on Image Processing* 17.10, pp. 1737–1754.

Gong, Rui, Wen Li, Yuhua Chen, and Luc Van Gool (2019). "DLOW: domain flow for adaptation and generalization." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial nets." In: *Advances in Neural Information Processing Systems* 27.

Ho, David Joon, Narasimhan P Agaram, Peter J Schüffler, Chad M Vanderbilt, Marc-Henri Jean, Meera R Hameed, and Thomas J Fuchs (2020). "Deep interactive learning: an efficient labeling approach for deep learning-based osteosarcoma treatment response assessment." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 540–549.

Hoffman, Judy, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell (2018). "CyCADA: cycle-consistent adversarial domain adaptation." In: *International Conference on Machine Learning*. PMLR, pp. 1989–1998.

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros (2017). "Image-to-image translation with conditional adversarial networks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1125–1134.

Johnson, Justin, Alexandre Alahi, and Li Fei-Fei (2016). "Perceptual losses for real-time style transfer and super-resolution." In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 694–711.

Kandel, Mikhail E, Yuchen R He, Young Jae Lee, Taylor Hsuan-Yu Chen, Kathryn Michele Sullivan, Onur Aydin, M Taher A Saif, Hyunjoon Kong, Nahil Sobh, and Gabriel Popescu (2020). "Phase imaging with computational specificity (PICS) for measuring dry mass changes in sub-cellular compartments." In: *Nature Communications* 11.1, pp. 1–10.

Kaskman, Roman, Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic (2019). "HomebrewedDB: RGB-D dataset for 6D pose estimation of 3D objects." In: *International Conference on Computer Vision (ICCV) Workshops*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet classification with deep convolutional neural networks." In: *Advances in Neural Information Processing Systems* 25.

Lin, Di, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun (2016). "ScribbleSup: scribble-supervised convolutional networks for semantic segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3159–3167.

Lin, Guosheng, Anton Milan, Chunhua Shen, and Ian Reid (2017). "RefineNet: multi-path refinement networks for high-resolution semantic segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mao, Xudong, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley (2017). "Least squares generative adversarial networks." In: *International Conference on Computer Vision (ICCV)*. IEEE, pp. 2813–2821.

Mildenhall, Ben, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng (2020). "NeRF: representing scenes as neural radiance fields for view synthesis." In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 405–421.

Morrison, Douglas, Adam W Tow, Matt Mctaggart, R Smith, Norton Kelly-Boxall, Sean Wade-Mccue, Jordan Erskine, Riccardo Grinover, Alec Gurman, T Hunn, et al. (2018). "Cartman: the low-cost cartesian manipulator that won the amazon robotics challenge." In: *International Conference on Robotics and Automation (ICRA)*, pp. 7757–7764.

Mueller, Franziska, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt (2018). "GANerated hands for real-time 3D hand tracking from monocular RGB." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 49–59.

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals (2018). "Representation learning with contrastive predictive coding." In: *arXiv preprint arXiv:1807.03748*.

Park, Taesung, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu (2020). "Contrastive learning for unpaired image-to-image translation." In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 319–345.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: convolutional networks for biomedical image segmentation." In: *Medical Image Comput-*

ing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III. Ed. by Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi. Vol. 9351. Lecture Notes in Computer Science. Springer, pp. 234–241.

Schwarz, Max and Sven Behnke (2020). "Stillleben: realistic scene synthesis for deep learning in robotics." In: *International Conference on Robotics and Automation (ICRA)*, pp. 10502–10508.

Schwarz, Max, Christian Lenz, Germán Martín García, Seongyong Koo, Arul Selvam Periyasamy, Michael Schreiber, and Sven Behnke (2018). "Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing." In: *International Conference on Robotics and Automation (ICRA)*, pp. 3347–3354.

Schwarz, Max, Hannes Schulz, and Sven Behnke (2015). "RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features." In: *International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1329–1335.

Shrivastava, Ashish, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb (2017). "Learning from simulated and unsupervised images through adversarial training." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2242–2251.

Stein, Gregory J and Nicholas Roy (2018). "GeneSIS-RT: generating synthetic images for training secondary real-world tasks." In: *International Conference on Robotics and Automation (ICRA)*, pp. 7151–7158.

Van der Maaten, Laurens and Geoffrey Hinton (2008). "Visualizing data using t-SNE." In: *Journal of Machine Learning Research* 9.11.

Wang, Feng, Huaping Liu, Di Guo, and Sun Fuchun (2020). "Unsupervised representation learning by invariance propagation." In: *Advances in Neural Information Processing Systems* 33, pp. 3510–3520.

Wang, Mei and Weihong Deng (2018). "Deep visual domain adaptation: a survey." In: *Neurocomputing* 312, pp. 135–153.

Wattenberg, Martin, Fernanda Viégas, and Ian Johnson (2016). "How to use t-SNE effectively." In: *Distill*.

Xiang, Yu, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese (2016). "ObjectNet3D: a large scale database for 3D object recognition." In: *European Conference on Computer Vision (ECCV)*. Springer.

Xiang, Yu, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox (2018). "PoseCNN: a convolutional neural network for 6D object pose estimation in cluttered scenes." In: *Robotics: Science and Systems (RSS)*.

Zhang, Richard, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang (2018). "The unreasonable effectiveness of deep features as a perceptual metric." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595.

Zhao, Hang, Orazio Gallo, Iuri Frosio, and Jan Kautz (2017). "Loss functions for image restoration with neural networks." In: *IEEE Transactions on Computational Imaging* 3.1, pp. 47–57.

Zhi, Shuaifeng, Edgar Sucar, Andre Mouton, Iain Haughton, Tristan Laidlow, and Andrew J Davison (2021). "iLabel: interactive neural scene labelling." In: *arXiv preprint arXiv:2111.14637*.

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A Efros (2017). "Unpaired image-to-image translation using cycle-consistent adversarial networks." In: *International Conference on Computer Vision (ICCV)*.