CHALMERS
UNIVERSITY OF TECHNOLOGY

Quality Inspection Using Machine Learning Techniques for Pose Estimation

Bridging the domain gap between synthetic and real images using CycleGAN

Master's thesis in Systems, Control and Mechatronics

FILIP FRANSON
MAX WEDENMARK
® Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Quality Inspection Using Machine Learning Techniques for Pose Estimation

Bridging the domain gap between synthetic and real images using CycleGAN

FILIP FRANSON
MAX WEDENMARK

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal processing and biomedical engineering
Computer vision and medical image analysis
Chalmers University of Technology
Gothenburg, Sweden 2020

Quality Inspection Using Machine Learning Techniques for Pose Estimation
Bridging the domain gap between synthetic and real images using CycleGAN
FILIP FRANSON & MAX WEDENMARK

Supervisor: Per-Lage Götvall, Volvo Trucks AB
Examiner: Torsten Sattler, Chalmers University of Technology

Master's Thesis 2020
Department of Electrical Engineering
Division of Signal processing and biomedical engineering
Computer vision and medical image analysis
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: In the left column are two synthetic images with rendered objects. In the middle column are the same objects on the same background after passing the object through a trained CycleGAN. In the right column are images of the real object from the T-LESS dataset.

Typeset in LaTeX

Quality Inspection Using Machine Learning Techniques for Pose Estimation
Bridging the domain gap between synthetic and real images using CycleGAN
FILIP FRANSON
MAX WEDENMARK
Department of Electrical Engineering
Chalmers University of Technology

Abstract

When training a neural network for pose estimation solely on synthetic images the network tends to overfit to certain features in the synthetic domain that are not present in the real world. As a result, the trained network has a decreased performance when applied to real images at test time. Previous research has concluded that in order to reduce overfitting and improve performance, the domain gap between the synthetic and real images needs to be reduced. This thesis has investigated how well the generative model CycleGAN can learn a mapping between the two domains in order to translate renderings into more realistic versions and as such reduce this domain gap. To do this CycleGAN was trained using real images from the T-LESS dataset and synthetic images used when training the pose estimation network PVNet. The effectiveness of reducing the domain gap was evaluated by comparing the performance of PVNet when trained on synthetic and CycleGAN generated data respectively. In addition to this the performance of CycleGAN was also measured by visual inspection in order to find if the transformed images were more realistic than the synthetic ones. In addition to testing on T-LESS objects, custom images were generated in order to find how well the translation task can be performed using less than perfect images. From the experiments it can be concluded that CycleGAN can reduce the domain gap and by doing so improve the performance of PVNet when applied in a real-world setting.

Keywords: **CycleGAN, PVNet, Pose estimation, Machine learning, Domain gap, Generative models, Synthetic training data**

Acknowledgements

First and foremost we would like to thank our examiner and supervisor Torsten Sattler, Associate Professor at Chalmers University of Technology for assistance with technical details and thorough feedback throughout the thesis. Secondly we would like to thank Per-Lage Götvall, Senior Research Engineer at Volvo AB for the opportunity to work on this interesting project and for continuous feedback. In addition to this we would also like to extend our thanks to Jun-Yan Zhu, first author of CycleGAN, and Sida Peng, first author of PVNet, for continued dialogues and assistance with technicalities related to their methods.

Filip Franson & Max Wedenmark, Gothenburg, June, 2020

Contents

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
1.1 Purpose . . . 2
1.2 Related work . . . 3
1.2.1 Image-to-image translation . . . 3
1.2.2 Generative Adversarial Networks (GANs) . . . 4
1.2.3 Domain adaptation . . . 5
1.2.4 6D Pose Estimation . . . 5
1.3 Data acquisition . . . 6

2 Theory **9**
2.1 GAN - Generative Adversarial Networks . . . 9
2.1.1 CycleGAN . . . 10
2.2 6D Pose estimation . . . 12
2.2.1 Pixel-wise Voting Network . . . 12

3 Method **15**
3.1 Implementation . . . 15
3.2 Description of the pipeline . . . 16
3.2.1 Training CycleGAN . . . 16
3.2.2 Using CycleGAN to train PVNet . . . 17
3.2.3 Generator/discriminator architecture . . . 18
3.3 Evaluation methods . . . 18

4 Results **19**
4.1 Experiments . . . 19
4.2 Data used for evaluation . . . 20
4.3 Evaluation metrics . . . 21
4.4 Implementation details . . . 23
4.5 Evaluation . . . 23
4.5.1 Translation task for T-LESS objects . . . 24
4.5.2 Pose estimation for T-LESS objects . . . 27
4.5.3 Translation task for Volvo object . . . 30

5 Conclusion and future work **33**
5.1 Contributions 33
5.2 Conclusion 33
5.3 Future work 34

Bibliography **37**

List of Figures

1.1 Figure taken from [1]. On the top row to the left is the setup used for data acquisition for T-LESS [1]: 1) Turntable with markers, 2) Black background used for taking training images, removed for test images, 3) The three different sensors on an arm with adjustable tilt in order to let the object be stationary between the images. The bottom left is an example image of the acquired data after processing taken from the T-LESS dataset. On the top right the setup used when taking images for the Top Hat Rail (THR) dataset [2] with an example image before processing in the bottom right taken from the THR dataset [2]. 7

2.1 Left: forward consistency. A sample x in source domain processed by G should end up close to x when processed in the reverse direction by F. Right: backward consistency. A sample y in target domain processed by F should end up close to y when processed by G. Figure taken from [3]. 11

2.2 Illustration of paired data (left) and unpaired data (right). In the two examples, the left column is rendered images of the real object in the right column. 12

2.3 Figure taken from [4] showing the overview of PVNet. (a) The input image is processed by (b) PVNet which outputs (c) unit vectors pointing towards keypoints and (d) semantic labels. These are used to generate (e) keypoint hypotheses by RANSAC-voting and (f) probability distributions of the keypoints. 14

3.1 An overview of the pipeline developed in this work. For each object, a CycleGAN model is trained using unlabeled real and labeled synthetic images which are later used to translate annotated renderings into more realistic versions. The new renderings are then used as training data when training PVNet for the pose estimation task. 16

3.2 Processing steps from existing T-LESS rendering to CycleGAN training image. The object in the original rendering (a) is cropped out and placed onto a black background (b). Based on the largest bounding box of all renderings in the dataset, the object is further isolated (c) to facilitate training and reduce training time. 17

4.1 The object used in the first experiment to tell if CycleGAN can translate an image between domains. Images taken from the T-LESS dataset [1]. 19
4.2 A selection of images of the custom object taken from the video sequence (upper row) used to generate the training images for CycleGAN (bottom row). The training images are rough around the edges due to reflections in the material as well as insufficient lighting of the background. 21
4.3 Results of the image translation task on object 15 in T-LESS dataset with real images of the object in the first row for comparison. The remaining images are CycleGAN input/output pairs. Note that this is the raw output from the generative network and that additional processing steps are done before any training of PVNet. 25
4.4 Input/output pairs from CycleGAN models trained on 100/500/1500/2500 images from each domain respectively. Object texture and color improves with the number of images while edges are smoother with fewer training samples. 25
4.5 Examples of some cases where CycleGAN fails to learn the mapping between the two domains. In the first row it fails to preserve the pose and instead translates the objects as if the hollow side of the object was visible. In the second row the learning has collapsed, making the output useless for further use. 27
4.6 Schematic view of how synthetic images used in PVNet are translated into a more realistic images using CycleGAN. The object in the synthetic image is cropped and placed on a black background which the CycleGAN model is trained on. The object in the output from CycleGAN is then cropped out using the mask acquired when creating the synthetic image. It is then placed on the same background as the original synthetic image. 28
4.7 Samples of objects 14, 15 and 16 illustrating the difference between the renderings, the CycleGAN generated images and the actual object. The CycleGAN generated image is composed as described in Section 3.2 i.e placing the object on the original background. Note that the real images are not correlated to the synthetic images and are only shown for visual reference. 28
4.8 Visualization of CycleGAN input/output pairs on the custom U-pipe object. The CycleGAN generated images maintain the original pose, have a similar metallic surface as the real images and manages to produce purple gaskets on each end of the pipe. The CycleGAN generated images have similar artifacts as the real images, such as jagged edges, which can be seen especially in the upper right and lower right pair. Note that the shown pairs are examples of the more successful translations while worse ones are show in Figure 4.9. 31
4.9 Input/output pairs from the same CycleGAN model shown in Figure 4.8. These are some examples of artifacts which distort the geometry or texture of the object making it unsuitable for use as training data. 32

List of Tables

4.1 The training parameters that were tuned in this thesis when training CycleGAN. The values shown are the ones used for CycleGAN when training on object 14, 15 and 16 in T-LESS. 23
4.2 Impact on pose estimation from PVNet trained on renderings processed by CycleGAN trained on 100/500/1500/2500 images from each domain. For this experiment, object 15 in T-LESS was used. 26
4.3 Pose estimation results from PVNet trained on {synthetic images} and {CycleGAN generated images} respectively. For all objects, the performance according to ADI is improved when using CycleGAN generated images to train PVNet. Performance according to 5cm 5° is not as consistent. 29
4.4 Result for PVNet trained on T-LESS objects with respect to ADI and 5cm 5°. One network is trained for each of the following datasets: {real images, synthetic images}, {real images} and {real images, CycleGAN generated images}. 30

1
Introduction

At Volvo, assembling a vehicle requires operators to follow detailed instructions involving multiple steps which include fitting different components in the right place and order. This means inevitably that a few products may have flaws which, if not discovered by the quality inspection team, will reach the customer. In order to increase the quality at the production line, Volvo investigates the possibility of training a neural network for quality inspection. This network should be able to tell whether or not parts are correctly assembled or if there are parts missing. With such a system, the risk of defect products leaving the manufacturing plants will hopefully decrease which would make Volvo an even more competitive company on the market.

To locate an object in space and to determine its orientation is referred to as the *pose estimation problem* [5, 6, 7, 8, 9, 4, 10], i.e. estimating the 6D-pose of an object given a 2D image. This can be solved using various methods but as of late, deep learning approaches have shown to be most prominent for computer vision tasks [11]. A traditional method to generate the data needed to train neural networks for e.g. object detection is to take real images and annotate them manually. Datasets annotated this way exist for simpler classification tasks such as binary classification of certain animals, image recognition of common objects and, although not simple, facial recognition. Datasets exist here because these are everyday objects of which there are a massive amount of annotated images available. The same is not the case for factory objects and especially not for pose-annotated images. Annotating images with object poses means specifying the rotation and translation of the object which is a challenging task for humans compared to identifying objects in an image. Because of this, companies that are interested in creating a network for quality inspection on their own objects will often times have to create a dataset on their own. This could prove to be a costly affair since each object usually requires thousands of annotated images which would be expensive to create mainly because of the large number of objects of interest. Because of this, new methods are being researched which instead rely on synthetic data [12, 7, 4, 9].

Most, if not all, components at Volvo each have their corresponding CAD-model which could be used to create synthetic training data. By using synthetic images in this way, the issue of having to create expensive datasets is reduced since the annotation can be automatically when creating the synthetic image. Doing this would therefore allow for more flexible training and creation of neural networks since the step of generating annotated real data is diminished.

As previously mentioned, producing annotated data for machine learning applications is laborious. Being able to use synthetic images instead would open up new possibilities as data would be much more accessible. A difficulty when training a neural network on synthetic data is the *domain gap* between synthetic and real images. The network may learn certain details in the synthetic data which are not present in its real counterpart [12]. Therefore reducing this domain gap is a major part of the pose estimation problem using synthetic data [12, 13], which is an active research field today, and is the main focus in this thesis.

There has been multiple previous theses at Volvo related to training neural networks on synthetic data. The most recent and relevant ones are [14, 5]. In [14] a pipeline for generating a synthetic dataset from CAD-models was created along with testing the dataset using neural networks for object detection. This work was extended on in [5] where the authors made use of this pipeline along with additional methods to train neural networks for pose estimation and object detection.

1.1 Purpose

The purpose of the thesis is to find how well the generative network CycleGAN [15] can reduce the differences, i.e the domain gap, between real images and synthetic images. The method will be evaluated both by visual inspection and by training PVNet [4], a neural network for 6D pose estimation, on synthetic images and CycleGAN generated images separately. The performance of these two models will then be compared to each other when applied on real images. The evaluation method is used because of the intended final implementation, which is to train neural networks for pose estimation using only synthetic data. These networks will later be used for quality inspection which means that they require high performance with minimal errors.

In [5] the authors concluded that the domain gap has to be reduced in order to improve performance of PVNet when trained using only synthetic images but applied in a real world setting. This conclusion is the main reason for investigating the effect of reducing the domain gap between real and synthetic images in this thesis. Since pose annotations are generated when creating the synthetic images, it is of great importance that the pose of the object is not altered when they are processed by CycleGAN. The original CycleGAN paper [15] show promising results for transforming images between two domains with little to no alteration of the image structure. In addition to this the method does not require paired training data, making it easier to generate training data for the model. These are the main reasons why CycleGAN will be investigated for reducing the domain gap between synthetic and real data.

1.2 Related work

The following section will cover previous works related to this thesis in order to allow for further reading as well as better motivating why CycleGAN was chosen as the method to reduce the domain gap between real and synthetic images. In addition to this it also covers the reasoning behind choosing PVNet as evaluation method.

1.2.1 Image-to-image translation

Image-to-image translation is most commonly used to change images such that they represent a different distribution all together. Examples could be changing a summer image to instead show the same scene but during winter time or taking a normal photograph and changing it to look like a painting made by a specific artist [16, 15].

In [17] new images are created by combining image *content* from one source and image *style* from another. In this case, image content refers to what is in the image, e.g. objects or environment, while image style refers to the visual appearance of the image. In the paper, a photograph of a row of houses is used and is translated into having similar style as e.g. The Starry Night by Van Gogh. The results are interesting since the method allows them to interpolate between different levels of structure and style allowing for high or low level alterations of the images. This method not only changes texture and color but also shapes. The shapes in the original image are not changed when using CycleGAN which makes it more suitable to use for creating pose-annotated training data. In addition to this, the model proposed in [17] only allows to transfer the style of one image rather than a domain of images. Because of these two things the method is not suitable for this thesis.

In [16] the authors make use of image-to-image translation to create Pix2Pix. In addition to the example in the beginning of this section, they can also colourise a black and white image and create new images based on labels. The labels in this case would look similar to the output you would get from performing semantic segmentation where each color represents a type of object. Training these models require paired training data [15] which means that the specific training image has to be represented in both domains. To use this method the images has to be annotated in both domains which defeats the purpose of reducing the domain gap as the annotated real images could be used in the first place. This is not the case when using CycleGAN and therefore it is more suitable for creating pose annotated training data.

CycleGAN is presented in [15] where the authors make use of what was discovered in [16] but instead used it to translate between domains rather than between specific image styles. This means that in addition to being able to make a photograph look like a painting made by Van Gogh, they can also do translations between domains such as the ever so popular example of making horses look like zebras and vice versa. What is interesting with this approach is that it does not require paired training data. This is of great importance for this thesis and one of the reasons

CycleGAN will be used. Something that separates CycleGAN from Pix2Pix, in addition to not requiring paired training data, is the overall network structure. Pix2Pix use a structure more similar to traditional GANs [3] with one generator and one discriminator while in CycleGAN there are instead two generators and two discriminators and as such is able to learn a mapping in both directions.

1.2.2 Generative Adversarial Networks (GANs)

In [12] the authors propose a method to train a model to improve the realism of a simulator using unlabeled real data. They call their method SimGAN and base it on traditional GANs [3] but make several changes to the model to be able to generate realistic images from the simulator's output. In the paper the method is used to improve the performance of an algorithm used for gaze estimation and show results that competes with state-of-the-art algorithms on the MPIIGaze dataset [18] without any labeled real data. The findings suggest that refining synthetic images this way can increase performance of the network once it is applied to real data compared to when the training images was not refined. The method shows great promise for the intended application in this thesis as it manages to efficiently lessen the domain gap between the two distributions of images. However in [15] it is stated that SimGAN enforces the output to be close to the input in a certain way which is something CycleGAN does not require. According to [15], this makes CycleGAN a more flexible approach for the translation task compared to SimGAN.

In [19] the authors train a GAN-based model using labeled synthetic data and unlabeled real data with the intention to train another neural network for image classification. The goal is for the image classification network to perform well on real world examples even though it was trained using unlabeled real images. They show promising results and state that the method is not specific to image classification which shows that the same principle can be used for other problems. One key assumption that is made in the method is that the differences between the two domains are of low-level character such as noise, resolution, color or illumination rather than of high-level character such as type of object or geometric differences. This would be a viable method to reduce the domain gap for training neural networks which further supports the idea of using a GAN to lessen the domain gap. However according to [15] this method would require the input and output to be similar in a predefined way as in [12]. The authors of [15] argue that their approach is a more flexible choice for the same task.

1.2.3 Domain adaptation

A different approach was taken by Ganin and Lempitsky [20] who instead use a technique called domain adaptation. In essence this means that the network is trained on one properly annotated distribution, the synthetic data, with the intention to use it on a non-annotated distribution, the real data. They do this by integrating domain adaptation in the training stage of deep neural networks such that the model focuses on learning features present in both the synthetic and the real domain. They optimize their feature extractor by learning features present in both domains. They do this by using a domain classifier which has the task of telling a real image from a synthetic one. However, the feature extractor is trained in order to make it harder for the domain classifier to tell the two types apart. The features passed to the domain classifier are also passed to a label classifier such that it is trained on features present in both domains. Their end goal, i.e. being able to train on synthetic data and apply it on real data, is similar to the purpose of this thesis however the way this is achieved differs significantly. The major difference is that our goal is to bridge the domain gap by learning a mapping from source data to target data while they instead alter how the network is trained.

In [21] an unsupervised domain adaptation method is proposed which focuses on making the covariance in source domain similar to the covariance in target domain. The proposed method is called correlation alignment (CORAL). Instead of, as in this thesis, having a machine learning based approach to reduce the domain gap they compute the covariance in source domain, apply a whitening transformation on the source data and then re-colors it with the covariance of the target domain. They evaluate their method on object recognition tasks where they show an increased performance over other methods.

1.2.4 6D Pose Estimation

A Pixel-wise Voting Network (PVNet) was created by Peng et al. [4] as a solution to the 6D pose estimation problem. They utilize a Pixel-wise voting network to estimate keypoints used for a Perspective-n-Point algorithm [22]. This approach is more robust to occlusion and objects only being partially in frame of the image. Peng et al. evaluated this approach on multiple datasets such as the YCB-dataset [9], LINEMOD [23] and OccludedLINEMOD [8] with great success. In addition to these they also performed experiments on the T-LESS dataset [1], although this was done after publishing their paper. The code being publicly available in addition to being successful on multiple dataset is two reasons for using PVNet in this thesis. Another reason is that they use both real and synthetic images to train the network on certain objects. This means that the code is structured in a way such that it is able to handle both real and synthetic images which is useful for this thesis.

The creators of PoseCNN [9] presents a convolutional neural network for solving the 6D pose estimation problem. First their method estimates the center of the object followed by the distance to it in order to estimate the position. By then using the predicted bounding boxes, and passing them through the final part of the network, they get a quaternion representation of the 3D rotation for each object. In addition to presenting this network they also propose a new loss function in order to allow PoseCNN to handle symmetric objects. They evaluate their method on the YCB-dataset [9] and OccludedLINEMOD [8]. Although having promising results it has not been tested on the T-LESS dataset like PVNet. The main difference between PoseCNN and PVNet is the way keypoint correspondences are found. The keypoint localization approach in PVNet makes it more robust against occluded objects compared to PoseCNN [4].

In [24] a new method for object detection and 6D pose estimation which is called Dense Pose Object Detector (DPOD) is presented. It uses an encoder-decoder neural network to estimate 2D-3D correspondences between the input image and 3D models. Given these correspondences the 6D pose is estimated using Perspective-n-Point (PnP) and RANSAC [22]. Following these steps, additional refinement of the estimated pose is made using a custom made deep learning scheme which is evaluated using real and synthetic data. The DPOD implementation was successful [7, 13, 25, 26, 27] both before and after the refinement step for methods using real and/or synthetic data [24]. While being successful it is not publicly available and as such using it in this thesis would be infeasible.

1.3 Data acquisition

One key part of any successful neural network is the underlying data. What kind of data this is depends on the implementation but something all neural networks share is the need for data. Most of the time very large amounts of data is needed, especially for neural networks which are related to images. In addition to simply having this data it also has to be properly annotated. This means that for every data point there has to be additional information stating what is needed for the neural network to learn. For pose estimation this means that in addition to having plenty of object images, the pose of the objects in each image has to be annotated in order for the neural network to be able to learn its pose. This annotation is most of the time what is costly to generate. These annotations can be acquired in different ways depending on the type of annotation but in many cases, data is being annotated manually. This is a common form of data generation for object detection, semantic segmentation and instance segmentation because humans are versatile and can do these annotations quickly and accurately.

Annotating datasets for pose estimation, where the differences between each image is relatively small, manual annotation is not used. This is because it is something humans are bad at, we can not tell the position and rotation of an object by looking at it. Instead setups similar to those in Figure 1.1 are used. This allows automatic annotation of the objects as the sensors are moved a specific number of degrees between each image. This yields accurate annotations with less human interaction although it still requires manual labour to a large extent in setting up the rig, changing objects and so on.

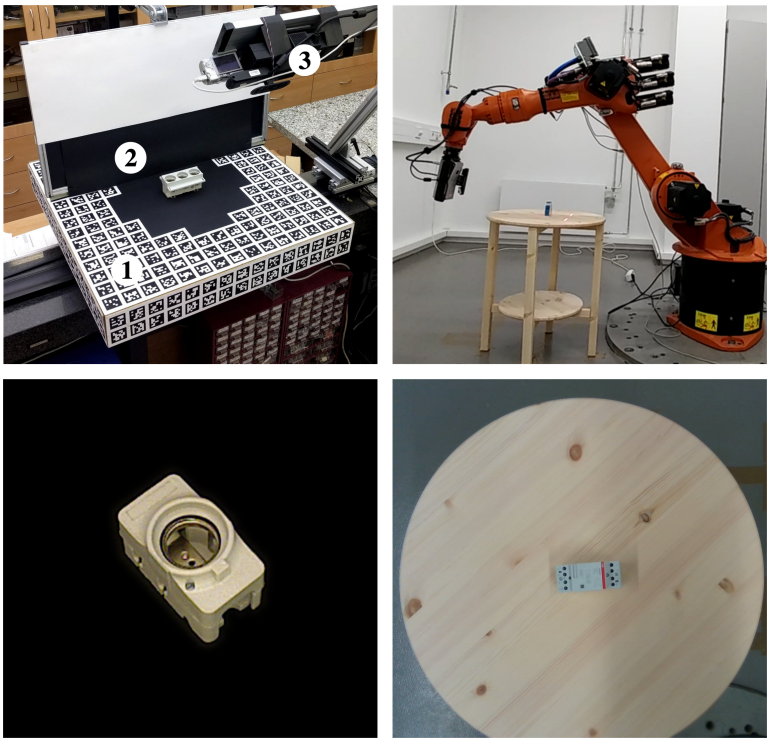

Figure 1.1: Figure taken from [1]. On the top row to the left is the setup used for data acquisition for T-LESS [1]: 1) Turntable with markers, 2) Black background used for taking training images, removed for test images, 3) The three different sensors on an arm with adjustable tilt in order to let the object be stationary between the images. The bottom left is an example image of the acquired data after processing taken from the T-LESS dataset. On the top right the setup used when taking images for the Top Hat Rail (THR) dataset [2] with an example image before processing in the bottom right taken from the THR dataset [2].

2
Theory

In this chapter, technical details are provided to give a better understanding of the core methods used in this thesis. More specifically, details will be given regarding how a generative adversarial network (GAN) works, how CycleGAN differs from a traditional GAN and how the pose of an object is estimated in PVNet.

2.1 GAN - Generative Adversarial Networks

A GAN [3] is a machine learning model that learns to generate new data with the same distribution as the data contained in the training set. It consists of two networks: a **generator** G and a **discriminator** D which are trained simultaneously in a two-player min-max game. The goal of the generator is to output a sample that the discriminator can not distinguish from the training set, i.e. maximize the probability that the discriminator is wrong. On the contrary, the goal of the discriminator is to be able to correctly tell whether or not the output sample from G is a sample from the training set or not, i.e. minimize its classification error. A well-trained GAN should ideally end up Nash equilibrium where neither the discriminator nor the generator can decrease their loss [28].

Discriminator
Let $p_{\text{data}}(x)$ be data in source domain and $p_{\text{data}}(y)$ be data in target domain. The discriminator outputs a value $D_Y \in [0, 1]$ which indicates its uncertainty regarding whether the sample is from source or target domain. To obtain a well-functioning discriminator it is essential to maximize the probability that it can distinguish between synthetic and real samples. Therefore, the objective for the discriminator [3] is

$$\mathcal{L}_{\text{GAN}}(D_Y) = \underbrace{\mathbb{E}_{y \sim p_{\text{data}(y)}}[\log D_Y(y)]}_{\text{real data}} + \underbrace{\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x)))]}_{\text{generated data}}, \tag{2.1}$$

which is to be maximized. $D_Y(y)$ is the probability of a sample in the target domain being real and $D_Y(G(x))$ is the probability that a generated sample from G is real.

Generator

The objective of the generator is to learn a mapping $p_{\text{data}}(x) \rightarrow p_{\text{data}}(y)$ such that $D_Y(G(x)) = 1$, i.e. the discriminator classifies the output from the generator as a real data sample. Mathematically, the objective for the generator is

$$\mathcal{L}_{\text{GAN}}(G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x)))], \tag{2.2}$$

which is to be minimized and therefore counteracts the second term in (2.1).

Combining D and G

By combining (2.1) and (2.2), the objective for GAN becomes

$$\mathcal{L}_{\text{GAN}}(D_Y, G) = \mathbb{E}_{y \sim p_{\text{data}(y)}}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x)))], \tag{2.3}$$

which the generator G wants to minimize while the discriminator D_Y aims to maximize it. Therefore the goal is to find optimal G^* and D_Y^* that minimizes (2.3) with respect to G and maximizes it w.r.t. D_Y. This is also often called *adversarial loss* which is the key component in successfully generating data using GANs.

2.1.1 CycleGAN

CycleGAN is a method for image translation tasks where the goal is to be able to transform images from one domain to another. An example of such task demonstrated by the authors of the first CycleGAN paper [15] is transforming images of horses to zebras and vice versa. What separates this method from ordinary GAN is: (i) the use of *two* generators, G and F, and *two* discriminators, D_X and D_Y, instead of one and (ii) an additional loss, the *cycle consistency loss*. The motivation behind this extension is that it helps the otherwise under-constrained mapping in image-to-image translation where only the adversarial loss is present. In such cases, a generator often ignores the input image and always generates the same image in the target domain which is called mode collapse. To counteract this, CycleGAN aims to learn two mappings: $G : X \rightarrow Y$ and $F : Y \rightarrow X$ and introduce the cycle consistency loss. Mapping G uses the adversarial loss to ensure that the distribution $p(G(X)) \sim p(Y)$ while mapping F uses the cycle consistency loss to impose that $p(F(G(X))) \sim p(X)$. In other words this means that if an image is translated from domain X to domain Y using generator G, it should end up as the original image when processed by the reverse translation using generator F. The cycle consistency loss contains both *forward* and *backward* cycle consistency which is illustrated in Figure 2.1.

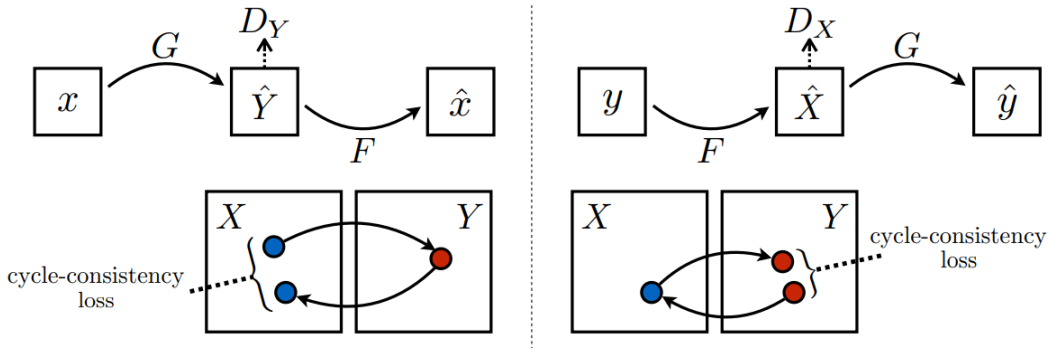

Figure 2.1: Left: forward consistency. A sample x in source domain processed by G should end up close to x when processed in the reverse direction by F. Right: backward consistency. A sample y in target domain processed by F should end up close to y when processed by G. Figure taken from [3].

Mathematically, the cycle consistency loss [15] is

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}(x)}}[||F(G(x)) - x||_1] + \mathbb{E}_{y \sim p_{\text{data}(y)}}[||G(F(y)) - y||_1], \tag{2.4}$$

which essentially is the reconstruction error in the cycle-consistency. The full objective for CycleGAN is obtained when combining the objective in (2.3) for the two generator discriminator pairs and the cycle consistency loss in (2.4) as

$$\mathcal{L}_{\text{tot}}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y) + \mathcal{L}_{\text{GAN}}(F, D_X) + \lambda \mathcal{L}_{\text{cyc}}(G, F), \tag{2.5}$$

where λ is a tuning parameter that determines the impact of the two types of losses. The goal is to find optimal generators G^* and F^* that minimize $\mathcal{L}_{\text{tot}}(G, F, D_X, D_Y)$ while maximizing it with respect to the two discriminators, i.e.

$$G^*, F^* = \min_{G,F} \max_{D_X, D_Y} \mathcal{L}_{\text{tot}}(G, F, D_X, D_Y). \tag{2.6}$$

As previously mentioned in this section, the additional features of CycleGAN not present in the original GAN opens up for the possibility of using unpaired data which makes the model more accessible to use in applications where generating a paired dataset is tedious. The difference between a paired and unpaired dataset is illustrated in Figure 2.2. Using a generative network to generate data for a 6D pose estimation neural network is one such task where an aligned dataset could be time consuming to generate. In order to be able to translate an arbitrary pose from one domain to another, most poses should be present in the two domains. A natural consequence of this is that the dataset needs to be relatively large. Therefore, being able to use unpaired data is a major incentive for using CycleGAN in data generation for pose estimation networks.

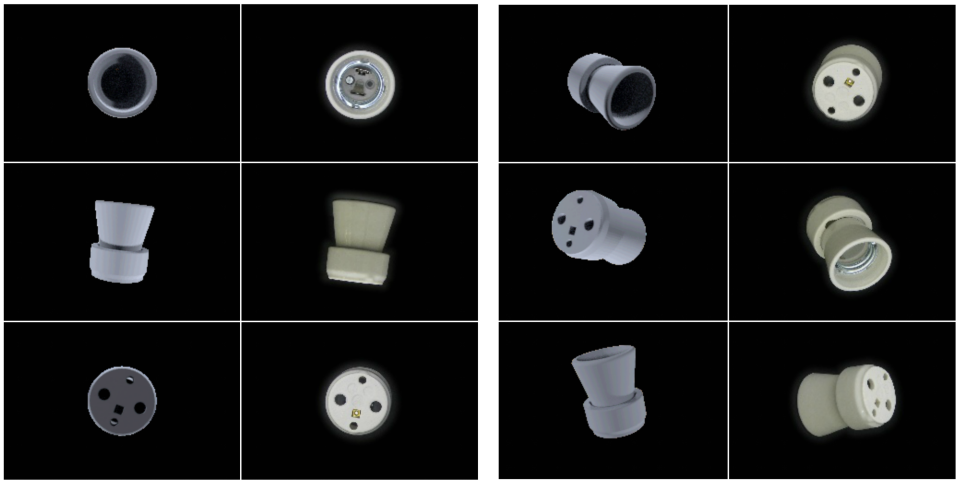

Figure 2.2: Illustration of paired data (left) and unpaired data (right). In the two examples, the left column is rendered images of the real object in the right column.

2.2 6D Pose estimation

In computer vision applications, such as autonomous driving and robotic manipulation, there is a need to read and understand the visual content of digital images. The idea is to make computers capable of visually interpreting the world much like humans do. 6D pose estimation is a task within the field of computer vision where the goal is to determine both the position and the orientation of an object relative to the camera that captured the image. Deep learning methods are nowadays the undisputed winner for these types of tasks [11] since they are able to better learn and extract higher level features from the raw digital input image compared to other methods. One high-performing deep learning method for pose estimation is PVNet [4] which will be used in this thesis as it was used in a previous thesis [5] and has open-source code available on GitHub [29]. In addition to this it is trained on synthetic and real images which means that it should benefit from a reduced domain gap as well as making an implementation more straightforward.

2.2.1 Pixel-wise Voting Network

Pixel-wise Voting Network (PVNet) [4] was introduced in 2019 and is able to reach state-of-the-art results on LINEMOD [23], Occlusion LINEMOD [8] and YCB-video dataset [9]. The authors have also conducted experiments on T-LESS which is the dataset used in this thesis. These experiments can be found on their GitHub repository [29]. PVNet estimates the pose of the object in two steps: first voting-based 2D object keypoint detection based on convolutional neural networks and then computation of the camera matrix using a modified Perspective-n-Point [22] (PnP) algorithm.

2D-3D keypoint correspondences

To solve the PnP-problem one first has to find 2D-3D keypoint correspondences. The keypoints are constructed from the 3D model for each object. A common way of selecting such keypoints is to use the corners of the 3D bounding box [30, 31] for the object but in PVNet they use keypoints that instead are spread out on the object surface. In their paper they show empirically that this method yields better results. They generate a set of K keypoints by defining the first keypoint as the object center and iteratively create new keypoints on the surface that is farthest away from the current set of keypoints until the size of the keypoint set is K. To do this, they use the farthest point sampling algorithm [32]. The projection of the created 3D keypoints are then the ground truth 2D keypoints used during training.

PVNet is able to generate hypotheses of 2D keypoints that are occluded or outside of the image by predicting semantic labels and unit vectors directed towards every 2D keypoint on the object. With the directions from each pixel in the object to every keypoint, they use RANSAC [22] to generate hypotheses of 2D locations for each keypoint. Since these hypotheses are generated from voting, each keypoint has its own voting score indicating its uncertainty of where it is located in the image. The process is illustrated in Figure 2.3. This way, the authors in [4] argue that PVNet handles object occlusion and cases where the object is partly outside of the image. This is a good property in an industrial setting where it can not be guaranteed that the object is fully visible at all times. It also enforces the network to put more emphasis on local object features and also becomes more robust against irrelevant background clutter.

Uncertainty-driven PnP

Based on 3D object points and 2D image projections, PnP is the problem of estimating the 6D camera pose which can be done with standard PnP solvers such as EPnP [33]. Unlike traditional solvers, PVNet also uses information regarding uncertainty from the keypoint hypotheses step to address the fact that different keypoints may have different confidences that, according to the authors, should be considered when solving the PnP problem. With the estimated mean position $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$ for $k = 1, ..., K$ where K is the number of keypoints, the 6D pose is computed by minimizing the Mahalanobis distance [4]

$$\min_{R,\boldsymbol{t}} \sum_{k=1}^{K} \Big(\tilde{\boldsymbol{x}}_k - \boldsymbol{\mu}_k\Big) \boldsymbol{\Sigma}_{\boldsymbol{k}}^{-\mathbf{1}} \Big(\tilde{\boldsymbol{x}}_k - \boldsymbol{\mu}_k\Big) \tag{2.7}$$

with $\tilde{\boldsymbol{x}}_k = \pi(R\boldsymbol{X}_k + \boldsymbol{t})$ where $\boldsymbol{X}_k$ is the 3D coordinate of the keypoint, $\tilde{\boldsymbol{x}}$ its 2D projection and π is the perspective projection function transforming from camera to image coordinates. R and $\boldsymbol{t}$ is the 3D rotation and translation respectively and are initialized by the EPnP solver. Finally, (2.7) is solved using the Levenberg-Marquardt [34] algorithm.

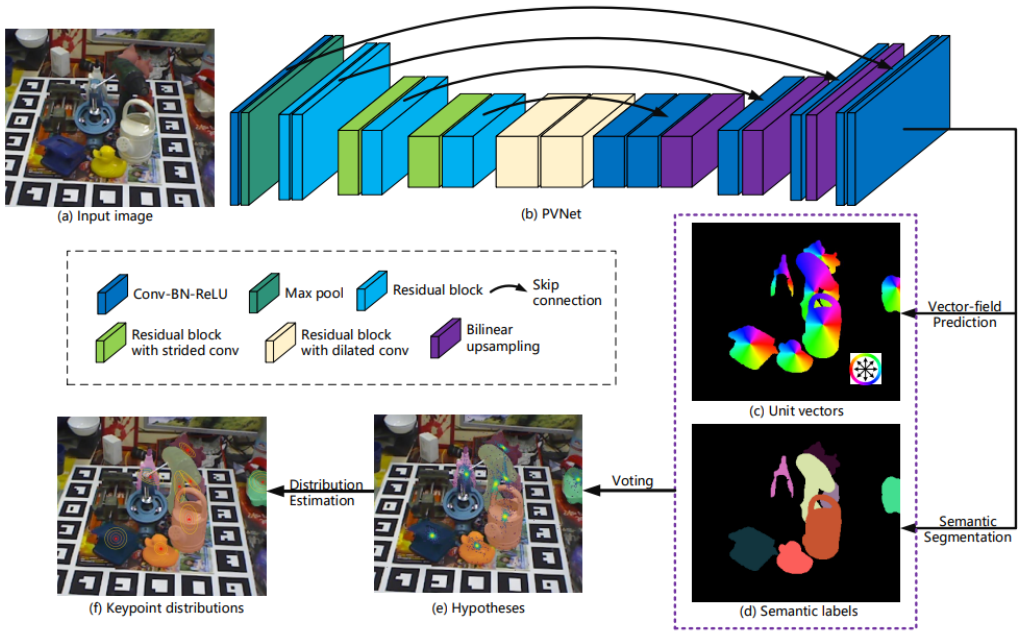

Figure 2.3: Figure taken from [4] showing the overview of PVNet. (a) The input image is processed by (b) PVNet which outputs (c) unit vectors pointing towards keypoints and (d) semantic labels. These are used to generate (e) keypoint hypotheses by RANSAC-voting and (f) probability distributions of the keypoints.

3
Method

To improve performance when the pose estimation network PVNet is trained on synthetic images and applied on real objects, CycleGAN was used to increase the realism of such images. As described in Section 2.1.1, CycleGAN does this by learning a mapping between the two domains in an unsupervised manner where synthetic and real images must be fed to the model during training.

In this chapter, an overview of how the problem was approached is given in Section 3.1 followed by a more concrete description of the developed pipeline in Section 3.2 and how the method was evaluated is presented in Section 3.3.

3.1 Implementation

Initially, CycleGAN was setup using code from the authors of the CycleGAN paper [15]. In addition to this PVNet was setup using an installation guide provided by the authors of PVNet available on their GitHub [29]. Following the initial setups the hyperparameters for CycleGAN was investigated in order to find what changes can be made to allow for better and more efficient training.

The first step regarding PVNet was to investigate its performance on three different types of data based on objects in the T-LESS dataset. The first was the recreation of their pre-trained models available on their GitHub [29] where both real and synthetic images were used. This was in order to have a point of comparison. Secondly it was examined how PVNet performs when trained only on their synthetic training data to see how it performs without processing the synthetic images using CycleGAN. Lastly, PVNet was trained on only real images in order to find how well CycleGAN generated images can compete with real data.

Following this, work on the pipeline described in Section 3.2 began. This was done in an agile manner where each part of the pipeline was evaluated before continuing with the next step. The first step of this development process is also what could be considered the first experiment. In essence, this was ensuring that CycleGAN actually can learn a mapping from renderings to real images while preserving the original pose. The experiments will be described in more detail in chapter 4.1. Once a CycleGAN model has been successfully trained it can be properly evaluated on PVNet. This means that PVNet will be trained using only the generated renderings from CycleGAN and be compared to the the result obtained from training PVNet on the three datasets described in the previous paragraph.

3.2 Description of the pipeline

In this section, a description of the entire pipeline developed in this thesis will be presented. An overview of the method is illustrated in Figure 3.1. The pipeline can be divided into two parts: (i) train CycleGAN on real and synthetic images for each object and (ii) generate more realistic versions of the original pose-annotated renderings using the trained CycleGAN and train PVNet on the CycleGAN generated images. In this way, the idea is to reduce the domain gap between real and synthetic data to address problem of generalization occurring when training a pose estimation network on renderings and applying it on real objects.

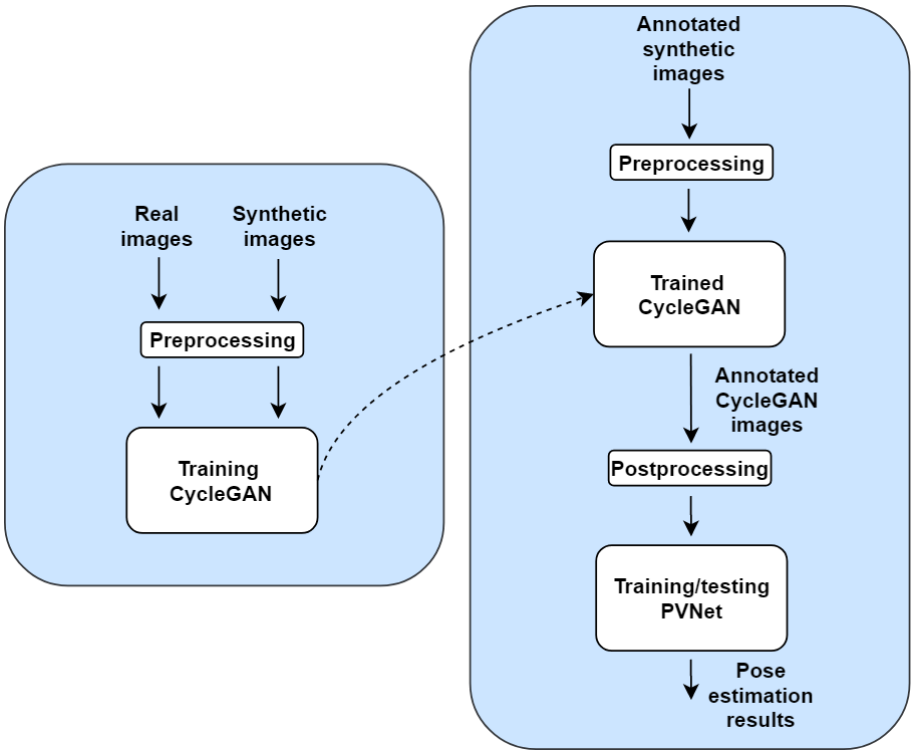

Figure 3.1: An overview of the pipeline developed in this work. For each object, a CycleGAN model is trained using unlabeled real and labeled synthetic images which are later used to translate annotated renderings into more realistic versions. The new renderings are then used as training data when training PVNet for the pose estimation task.

3.2.1 Training CycleGAN

To use the first part of the pipeline, i.e. training CycleGAN, unlabeled real and labeled synthetic images of an object are required. The images in the two domains can be unpaired however it is assumed that there are some underlying relationships between them [15]. One such relationship could be that the real and rendered objects are placed in a similar scene. The T-LESS dataset contains CAD models, real images with black background and test images for 30 different industry-relevant objects. These objects are chosen as they are textureless and have a high degree of

symmetry to make the pose estimation task more challenging. With this dataset it is possible to render images from the 3D models and use them together with the real images to train CycleGAN. In this thesis, synthetic images of T-LESS objects are generated in Blender using a script provided by the authors of PVNet available on their GitHub [29]. These images are used to better be able to determine the impact of the translation task when training PVNet on the CycleGAN generated images.

While the real images in the T-LESS dataset all have black backgrounds, the synthetic images of T-LESS objects are created with backgrounds from the SUN2012 [35] dataset. Since CycleGAN assumes some underlying relationship between the two domains, the data is preprocessed before training CycleGAN. The steps are shown in Figure 3.2. The rendered object is cropped out and placed on a black background using the provided mask for each rendering. To increase the size of the object in relation to the image size, both the rendered and real objects are cropped out based on the largest object bounding box. This facilitates training and reduces the overall image size which speeds up training significantly.

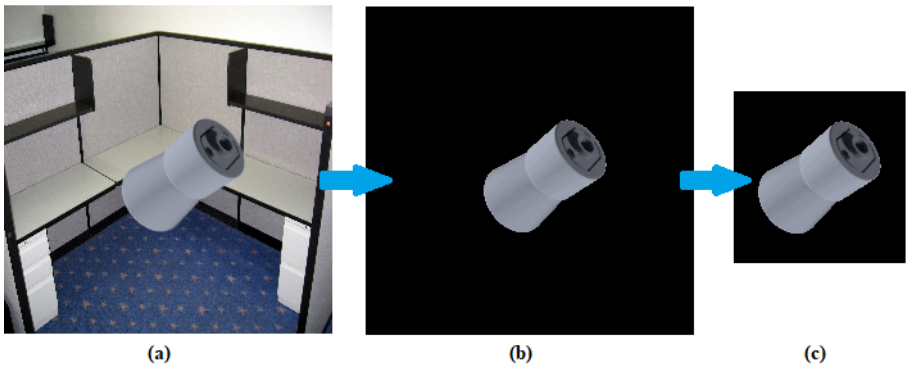

Figure 3.2: Processing steps from existing T-LESS rendering to CycleGAN training image. The object in the original rendering (a) is cropped out and placed onto a black background (b). Based on the largest bounding box of all renderings in the dataset, the object is further isolated (c) to facilitate training and reduce training time.

3.2.2 Using CycleGAN to train PVNet

After a CycleGAN model has been successfully trained, it can be used to translate arbitrary renderings of the same object in the original dataset into more realistic versions. When generating images using a trained CycleGAN model, the images are preprocessed in the same way as when training CycleGAN, see Figure 3.2. The output of the generative network will have the same image size and background as in Figure 3.2(c) but with a new version of the object. To be able to train PVNet on the CycleGAN generated image, the reverse steps in Figure 3.2 (c) $\rightarrow$ (b) $\rightarrow$ (a) are taken in the postprocessing steps of CycleGAN. This way the only difference between the original training image and the new one is the object itself which makes it possible to more accurately determine the impact of the translation task.

3.2.3 Generator/discriminator architecture

In this thesis no additional changes are done to the core of the discriminator and generator architecture used in the original CycleGAN paper [15]. The generator used in CycleGAN is a modification of the generative networks used in [36] as it has proven to be successful on style transfer tasks. Details regarding the network architecture can be found in the supplementary material of [36]. The modifications made by CycleGAN to the generator architecture in [36] is that adding additional residual blocks [37] depending on the resolution of the training images. For images of size 128×128 they use 6 blocks while for images of size 256×256 or larger, they use 9 blocks. As for the discriminator they use a PatchGAN [38] which aims to determine which domain $N \times N$ patches of the image belongs to. For the translation task in CycleGAN, 70×70 patches are used. The discriminator is passed across the image using convolutions and the result is averaged to obtain the final output from the discriminator.

3.3 Evaluation methods

The evaluation for CycleGAN is inherently tricky. The only way to really know whether or not CycleGAN itself is performing well is to look at the output. This is unfortunate since it is subjective what one would consider a good result. What was looked at in this thesis to tell whether a CycleGAN was performing well were object texture, distortion, color and pose preservation. It is of great importance that CycleGAN does not alter the pose of the object since this would make the existing annotation from the rendering useless. This way of visually evaluating the method will be used for the objects from T-LESS and from Volvo Trucks AB.

To evaluate how suitable CycleGAN is for reducing the domain gap, the metrics from PVNet trained on CycleGAN images are used. These can then be compared to results where PVNet is trained on real images, synthetic images or a combination of the two. A benefit of evaluating the methods this way is that it is objective and measurable. In addition to this it also better shows how well the domain gap is closed according to the pose estimation network instead of how we subjectively interpret it as being closed. This evaluation method will be used for objects in the T-LESS dataset but not for objects from Volvo Trucks AB. This is due to the time it takes to create an annotated dataset without having a complete system for it at hand.

4
Results

In the following chapter the experiments will be covered along with their respective results. Before the results, specifics related to data acquisition, evaluation metrics and implementation details will be covered in order to better understand how the results has been produced.

4.1 Experiments

Following the setup of the CycleGAN network, a series of experiments were conducted in order to establish how relevant CycleGAN is for translating synthetic images into more realistic images.

Experiment 1:
The first experiment is essential to the thesis and is conducted in order to tell whether or not CycleGAN can generalize a translation for an object from the T-LESS [1] dataset. The first object of interest was object 15 shown in Figure 4.1. The hyperparameters of the network were tuned to allow the network to create an image which better resembles a real object than the synthetic images. These hyperparameters are then used for the remaining experiments.

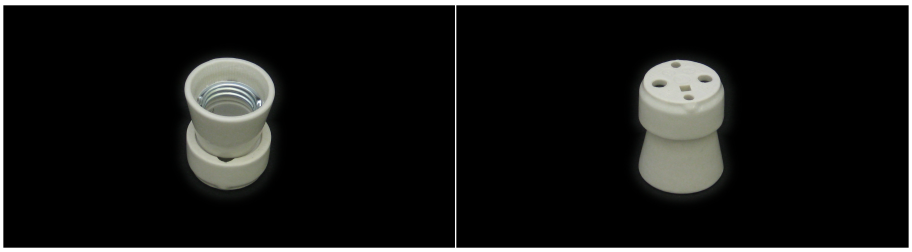

Figure 4.1: The object used in the first experiment to tell if CycleGAN can translate an image between domains. Images taken from the T-LESS dataset [1].

Experiment 2:
The second experiment aims to find how many images are needed in order for CycleGAN to produce good results. The number of real images is most interesting as these are most laborious to generate. In addition to this, using less images will result in shorter training times of CycleGAN. Both of these factors will help determine if the method is suitable for large-scale applications. The experiment will be evaluated partly by visually inspecting the translated images to find if they are satisfactory and partly by training PVNet in order to tell whether or not additional images yield a better CycleGAN model.

Experiment 3:
This experiment consists of training CycleGAN for different objects from the T-LESS dataset and using the new CycleGAN generated images to train and evaluate PVNet. To do this we create five types of datasets containing: {real images, synthetic images}, {real images}, {synthetic images}, {CycleGAN generated images} and {real images, CycleGAN generated images}. Comparing the performance of PVNet when trained on the different datasets will give insight into how well CycleGAN is able to close the domain gap.

Experiment 4:
The final experiment investigates if it is possible to train CycleGAN on custom objects where the training data is generated using a simple setup. To establish this, a U-pipe provided by Volvo Trucks AB is used. The synthetic images are generated using Blender and the real images are obtained by extracting frames from video sequences. Following the creation of this dataset the images were used to train a CycleGAN model and, as in experiment 1, the results were evaluated using visual inspection.

4.2 Data used for evaluation

The data used to evaluate the pipeline described in Section 3.2 can be divided into two parts. The first part is related to the T-LESS dataset and most experiments were conducted on this dataset. The objects in this dataset are industry relevant and texture-less objects such as power sockets, lamp switches, light bulb connectors and similar. The second part is related to the U-pipe object provided by Volvo Trucks AB. In this part the real and synthetic images were produced by us.

For experiments 1-3 the real images needed were taken from the T-LESS dataset. The images used was of the highest resolution available and later cropped and scaled to 128×128 pixels in order to allow for training CycleGAN. The resolution of the images had to be reduced since training on high resolution images takes a massive amount of memory due to the network structure. Training CycleGAN also requires synthetic images which are the same size as the real images and are created as described in Section 3.2.1. The generated synthetic images have resolution 256×256 and the object is, as described in Section 3.2.1, cropped out and scaled to 128×128 when training CycleGAN. The output object from CycleGAN is placed

on the original background, as described in Section 3.2.2, in order to create the training data for PVNet which is of size 256×256.

When working on experiment 4, synthetic images were rendered in Blender and the real images were taken and prepared by us. The object used was a U-pipe made out of plain metal with a purple gasket at each end. Using a simple setup with a green screen, which the object was placed at, several short videos were captured while rotating the camera around the object. Every third frame was then extracted from these video sequences and the background was removed. In addition to this, the images were made square to allow for easier training of CycleGAN. The resulting images isolate the object however it is not a flawless crop. Since there exists no object mask it has to be isolated based on colors alone which in turn leave some jagged edges, residual colors or other artifacts which can be seen in Figure 4.2. Since the U-pipe is metallic, it will naturally reflect the surrounding colors giving the object a green hue as well. The images do not perfectly represent the real objects but they do so sufficiently well in order to determine whether or not the translation task can be done on other custom objects.

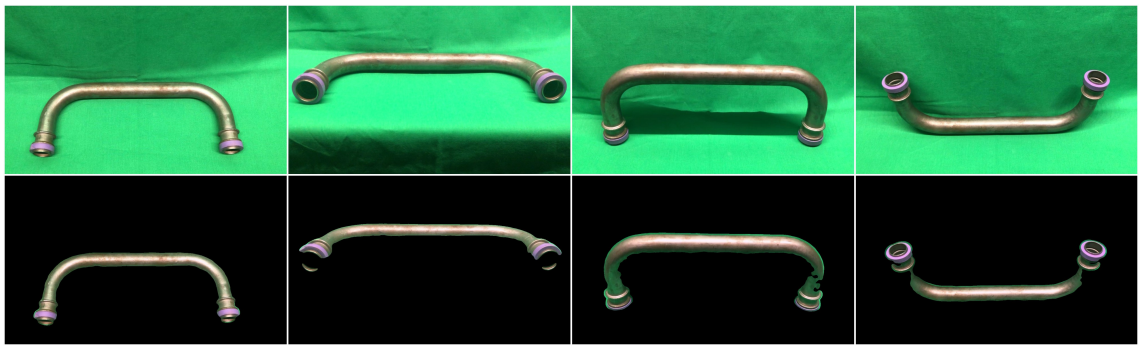

Figure 4.2: A selection of images of the custom object taken from the video sequence (upper row) used to generate the training images for CycleGAN (bottom row). The training images are rough around the edges due to reflections in the material as well as insufficient lighting of the background.

4.3 Evaluation metrics

Given an image, one aims to estimate the 6D pose of the object and evaluating the accuracy of this estimation is not always straightforward. For example, if the object of interest is a texture-less mug, there exists multiple poses that are indistinguishable from one another which makes the object pose ambiguous. There are also instances where the pose ambiguity is caused by self-occlusion, e.g. the handle of the mug is hidden behind the mug, occlusion by other objects or where the object is only partially captured in the image.

Two common evaluation metrics in 6D pose estimation are Average Distance of Model Points (ADD) introduced in [10] and 5cm 5° introduced in [39] and are used in PVNet. 5cm 5° emphasizes the rotational part of the pose over the translational part as it is harder to stay within 5 deg than within 5cm. For ADD on the other hand the translational part is more important than the rotational part according to [40]. However, for the objects in T-LESS the authors of PVNet use a slightly modified version of ADD which is often called ADI [41] and was proposed in [10]. Therefore the ADI metric will be used instead of ADD for T-LESS in this thesis as well.

In 5cm 5°, the estimated pose must be within 5 cm translational error and 5° angular error when compared to the ground truth for it to be considered correct. In ADD, known model points $\boldsymbol{x}$ are projected by the estimated pose $\hat{\boldsymbol{P}}$ and the ground truth pose $\boldsymbol{P}$ respectively. The metric is then the average distance between these projections. If the model $\mathcal{M}$ has no indistinguishable views, the error is calculated as

$$e_{\text{ADD}}(\hat{\boldsymbol{P}}, \boldsymbol{P}; \mathcal{M}) = \frac{1}{N} \sum_{\boldsymbol{x} \in \mathcal{M}} ||\boldsymbol{P}\boldsymbol{x} - \hat{\boldsymbol{P}}\boldsymbol{x}||_2 \tag{4.1}$$

where N is the total number of points on the object model. On the other hand, if the model *has* indistinguishable views like many objects in T-LESS, Hinterstoisser et al. in [10] propose to instead use ADI where the error is calculated as

$$e_{\text{ADI}}(\hat{\boldsymbol{P}}, \boldsymbol{P}; \mathcal{M}) = \frac{1}{N} \sum_{\boldsymbol{x_1} \in \mathcal{M}} \min_{\boldsymbol{x_2} \in \mathcal{M}} ||\boldsymbol{P}\boldsymbol{x_1} - \hat{\boldsymbol{P}}\boldsymbol{x_2}||_2 \tag{4.2}$$

where $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are the two closest model points.

The ADI error in (4.2) is therefore a relaxation of the ADD error in (4.1) since it does not require the projection of the same model point from $\boldsymbol{P}$ and $\hat{\boldsymbol{P}}$ to be close. Rather, there should be some other model point $\boldsymbol{x}'$ that, when projected with the ground truth, is close to the projection of $\boldsymbol{x}$ with the estimated pose $\hat{\boldsymbol{P}}$. The estimated pose is considered correct if the error is below a certain threshold, i.e.

$$\text{score} = \begin{cases} 1 & \text{if } e \leq k \cdot m \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where k is a scaling factor and m is the distance between the two furthest points in $\mathcal{M}$. In all experiments, $k = 0.1$ which is the default value in PVNet. The range for both 5cm 5° and ADI are between 0 and 1 where 0 means that no estimations are considered correct and 1 means that all estimations are considered correct.

4.4 Implementation details

The computer used for all experiments is a Dell precision 7740 running Ubuntu 18.04 LTS with a 16GB Quadro RTX 5000/PCLe/SSE-2 graphics card and an Intel core i7-9850H CPU @ 2.60 GHz x 12 processor. The video sequences were taken with an iPhone X at 720p 30 fps and frames were extracted using VLC media player.

To successfully train a CycleGAN model, the hyperparameters needs to be tuned based on the data used. Table 4.1 highlights the parameters that were mostly tested in this thesis and are the parameters used to obtain most of the result shown in Table 4.3 and 4.4. With the computer described above, training CycleGAN for 200 epochs on 128×128 images using batch size 8 takes approximately 5 hours when 1000 training images from each domain are used.

During the training of PVNet all hyperparameters were set to default values. The default values in this case are those active when you first set it up following their installation guide. With the computer used in this thesis, training PVNet takes approximately 20 hours when using the prepared data for T-LESS that the authors of PVNet provide on their GitHub [29].

Option	**Our value**	**Description**
`--netG`	`resnet_6blocks`	Architecture of the generator
`--netD`	`3`	Number of layers in the 70×70 PatchGAN discriminator
`--batch_size`	`8`	Batch size used for training
`--load_size`	`128`	Load images to this size
`--preprocess`	none	No preprocessing of images
`--lr`	`0.0002`	Initial learning rate for the Adam optimizer
`--n_epochs`	`100`	Number of epochs with initial learning rate
`--n_epochs_decay`	`100`	Number of epochs to decay learning rate to zero

Table 4.1: The training parameters that were tuned in this thesis when training CycleGAN. The values shown are the ones used for CycleGAN when training on object 14, 15 and 16 in T-LESS.

4.5 Evaluation

To better explain and present the results from the different experiments they are divided into subsections. These subsections are related to the translation task of CycleGAN, the results from PVNet using objects from T-LESS as well as the translation task on a custom object from Volvo. All results related to PVNet are obtained from a separate test set on real images provided in the T-LESS dataset. Each experiment related to PVNet was run only once due to the time it takes to train PVNet, see Section 4.4. Due to this, the values for the performance metrics in each experiment may vary if averaged over multiple runs.

4.5.1 Translation task for T-LESS objects

Experiment 1 is conducted in order to tell how well CycleGAN can translate renderings into more realistic looking images. The real objects in the T-LESS dataset are captured with a black background. The renderings used when training CycleGAN are the same as used by the authors of PVNet on the T-LESS dataset as this gives a good ground for comparison when evaluating the effect of the translation task. To better isolate the differences in the two domains, and therefore facilitate the training, all rendered objects are processed to also have a black background. In Figure 4.3, CycleGAN input/output pairs are shown together with real images of object 15 in the T-LESS dataset. As mentioned in Section 3.3, it is important that the translation yields realistic object texture and color, preserves the pose of the input and does not distort the object. By visual inspection, the translation task fulfills these requirements to a large extent but has some flaws, especially on the hollow parts of the object.

Experiment 2 addresses the number of real images needed to produce good results from CycleGAN. This is of interest since it will ease the process of generating real images on custom objects and reduce the training time of the networks. To facilitate the evaluation of the experiment, only the dataset size was varied while network architecture and hyperparameters were kept constant even though the optimal settings can change with the dataset size. This way, the impact of varying the dataset size is isolated such that conclusions can be drawn from the changes. The real images were selected uniformly from the T-LESS dataset to ensure that most object poses are present in the training set. Four different CycleGAN models were trained on 100, 500, 1500 and 2500 images from each domain respectively. The networks are evaluated both visually and against PVNet where the original renderings are processed by the different CycleGAN models. In Figure 4.4, four input/output pairs for the different models are shown.

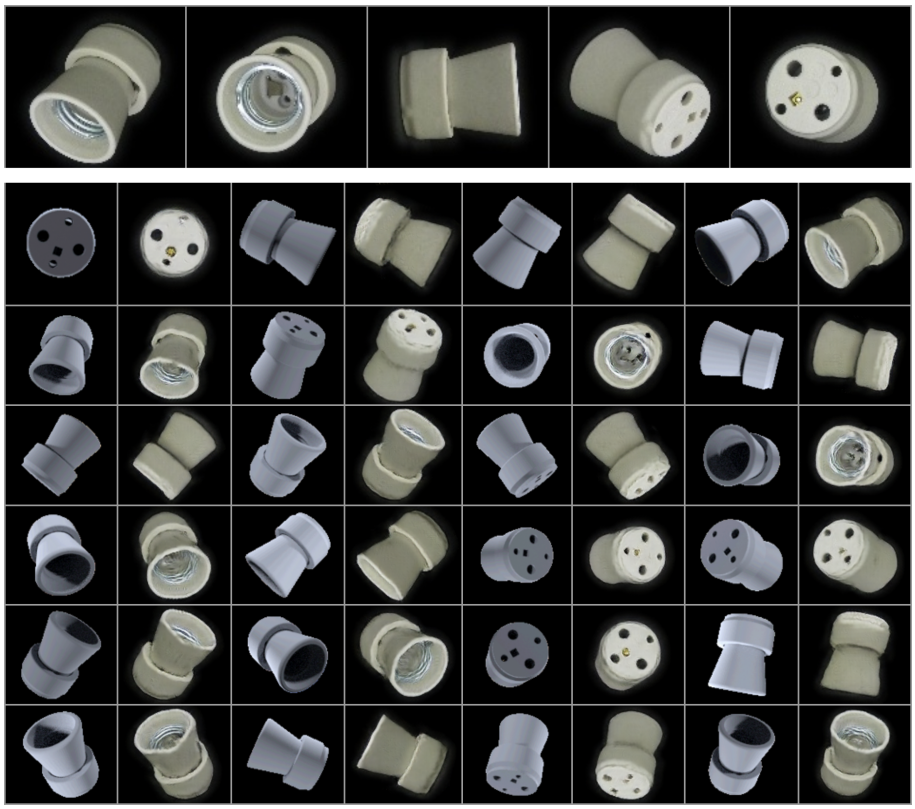

Figure 4.3: Results of the image translation task on object 15 in T-LESS dataset with real images of the object in the first row for comparison. The remaining images are CycleGAN input/output pairs. Note that this is the raw output from the generative network and that additional processing steps are done before any training of PVNet.

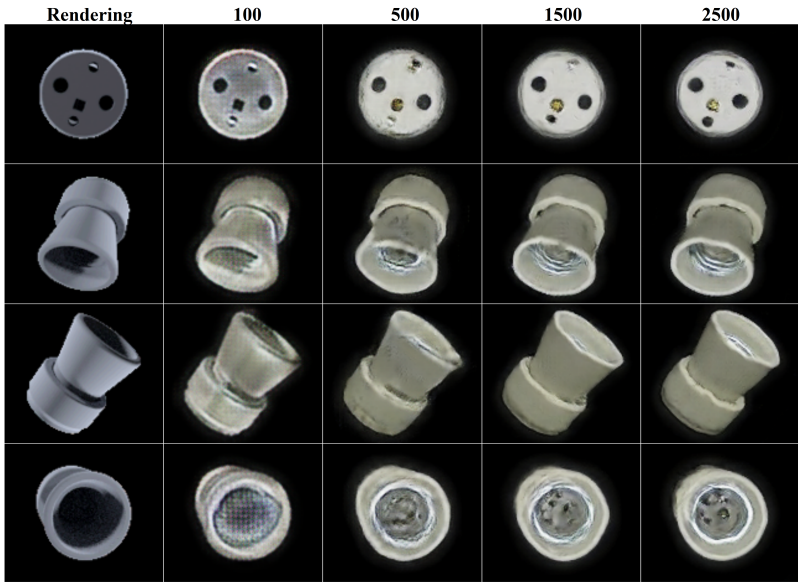

Figure 4.4: Input/output pairs from CycleGAN models trained on 100/500/1500/2500 images from each domain respectively. Object texture and color improves with the number of images while edges are smoother with fewer training samples.

As can be seen in Figure 4.4, texture and color of the output improves with the number of images used in training. This is expected as more training images yields more object views in the training set which is advantageous when arbitrary poses are to be translated. On the other hand, training on fewer images tend to keep edges more smooth compared to when trained on more images. In Table 4.2, results from PVNet trained on renderings processed by the four different CycleGAN models are shown. It can be observed that performance, according to ADI, improves using the CycleGAN model trained on 100 images. ADI increases further when CycleGAN is trained on additional images up to 1500 images. The change in performance are more distinct for ADI than for 5cm 5° where there is little to no change between the different CycleGAN models. According to the metrics, the translational part improves while the rotational part remains similar with number of training images for CycleGAN. This means that the general performance seems to improve, both visually and according to the metrics, when using more images. However, the performance does not seem to improve with additional images after using 1500 although more than 2500 images has not been tested.

	Rendering	100	500	1500	2500
ADI	0.29	0.40	0.54	**0.57**	0.52
5cm 5°	0.09	**0.10**	0.08	0.08	0.07

Table 4.2: Impact on pose estimation from PVNet trained on renderings processed by CycleGAN trained on 100/500/1500/2500 images from each domain. For this experiment, object 15 in T-LESS was used.

While the translation task is successful for some objects in the T-LESS dataset, it was found during experiment 3 that CycleGAN has greater difficulties in learning the mapping between the two domains for some other objects in the dataset. The translation can have artifacts such as loss of texture, pose or geometry distortion which can be seen in Figure 4.5. Since CycleGAN is an interplay between four separate neural networks, the exact causes for these artifacts can be hard to identify. It is however reasonable to say that the artifacts are caused by either generator/discriminator imbalance, choice of hyperparameters, the underlying data, the object itself or a combination of these. The hyperparameters described in Section 4.4 resulted in successful translations for the objects in Section 4.5.2. Even though the hyperparameters worked for these objects they are not general for all objects in T-LESS meaning that additional tuning might yield successful translation for the failed cases as well.

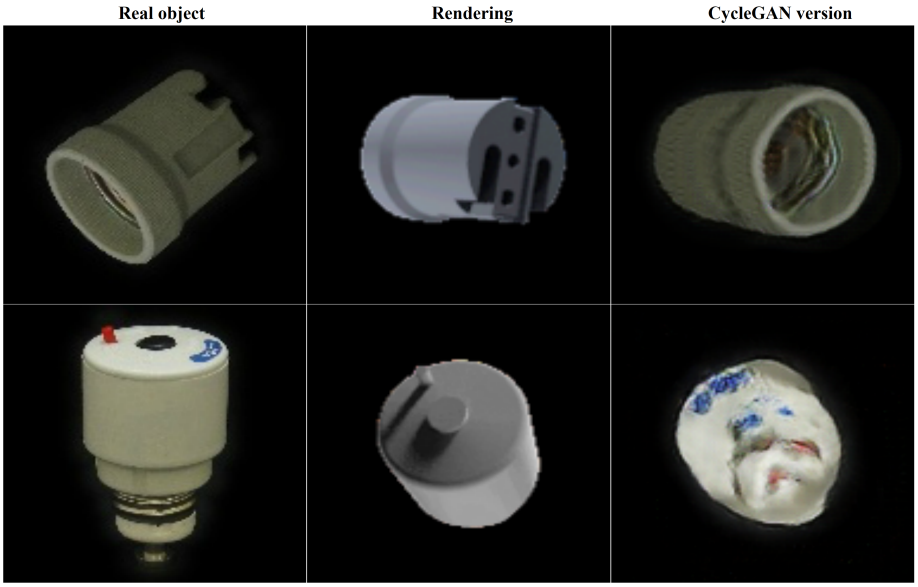

Figure 4.5: Examples of some cases where CycleGAN fails to learn the mapping between the two domains. In the first row it fails to preserve the pose and instead translates the objects as if the hollow side of the object was visible. In the second row the learning has collapsed, making the output useless for further use.

4.5.2 Pose estimation for T-LESS objects

In experiment 3, in order to directly measure the impact of the translation task, the original T-LESS renderings used by the authors of PVNet was processed by CycleGAN trained on a subset of these renderings. The steps from synthetic images to CycleGAN generated images are shown in Figure 4.6. The objects chosen for evaluation are object 14, 15 and 16 in the T-LESS dataset which can be seen in Figure 4.7. As the pose is preserved it is possible to keep the pose-annotation generated with the original rendering when training PVNet with the CycleGAN generated image. To evaluate the impact of the translated images, PVNet is trained on five different datasets: {real images, synthetic images}, {real images}, {synthetic images}, {CycleGAN generated images} and {real images, CycleGAN generated images}. Using these datasets it is possible to determine how well CycleGAN manages to close the domain gap. Based on the results from experiment 2, shown in Table 4.2, 1500 images from each domain was used when training the CycleGAN models used.

Figure 4.6: Schematic view of how synthetic images used in PVNet are translated into a more realistic images using CycleGAN. The object in the synthetic image is cropped and placed on a black background which the CycleGAN model is trained on. The object in the output from CycleGAN is then cropped out using the mask acquired when creating the synthetic image. It is then placed on the same background as the original synthetic image.

Figure 4.7: Samples of objects 14, 15 and 16 illustrating the difference between the renderings, the CycleGAN generated images and the actual object. The CycleGAN generated image is composed as described in Section 3.2 i.e placing the object on the original background. Note that the real images are not correlated to the synthetic images and are only shown for visual reference.

The results in Table 4.3 shows an increased performance for CycleGAN generated images when compared to synthetic images according to ADI for all tested objects. The most noteworthy improvement is for object 15 where the performance is almost doubled. As stated in Section 4.3, the ADI metric puts emphasis on the translational part of the pose and as such CycleGAN helps improve the estimation of translation for the objects compared to when only using synthetic images. The 5cm 5°, which puts an emphasis on the rotational part, is not as consistent and performance according to this metric may improve or remain according to the results in Table 4.3.

Object	**Metric**	**Synthetic images**	**CycleGAN images**
14	ADI	0.03	**0.15**
	5cm 5°	0.00	0.00
15	ADI	0.29	**0.57**
	5cm 5°	**0.09**	0.08
16	ADI	0.24	**0.32**
	5cm 5°	0.06	**0.16**

Table 4.3: Pose estimation results from PVNet trained on {synthetic images} and {CycleGAN generated images} respectively. For all objects, the performance according to ADI is improved when using CycleGAN generated images to train PVNet. Performance according to 5cm 5° is not as consistent.

Table 4.4 shows additional results acquired when training PVNet on datasets containing real images. These results are used to further show how CycleGAN affects the training of PVNet when using real images as well as CycleGAN generated images. To make for a more reasonable comparison the results are compared to two other datasets using real images. The results here are not as conclusive as in Table 4.3 but it still seems like including CycleGAN images can improve performance. For object 14, best performance is obtained by not including the synthetic images in the training set. For object 15 the performance is rather even between the three datasets. It is marginally better using both real and synthetic images according to ADI while 5cm 5° is best for real images only. For object 16, training PVNet on real images and CycleGAN images yields best result.

Object	Metric	Real images + synthetic images	Real images	Real images + CycleGAN images
14	ADI	0.17	**0.23**	**0.23**
	5cm 5°	**0.01**	**0.01**	0.00
15	ADI	**0.52**	0.50	0.49
	5cm 5°	0.13	**0.14**	0.12
16	ADI	0.63	0.61	**0.64**
	5cm 5°	0.20	0.16	**0.26**

Table 4.4: Result for PVNet trained on T-LESS objects with respect to ADI and 5cm 5°. One network is trained for each of the following datasets: {real images, synthetic images}, {real images} and {real images, CycleGAN generated images}.

The results in Table 4.3 shows that it is evident that translating the synthetic images using CycleGAN can reduce the domain gap and as such improve the performance on PVNet. This means that using CycleGAN generated images rather than synthetic images is completely feasible and would most often yield an improved performance. The most notable increase in performance is object 15 where ADI reaches 0.57. This is twice as accurate compared to when using only synthetic images and is also better than when it was trained on both real and synthetic images as can be seen in Table 4.4. We believe this is mainly due to two things. Firstly, as mentioned in Section 4.1, the hyperparameters were tuned for object 15 and as such this translation can be considered the most successful one. Secondly, the CycleGAN generated images are not completely flawless, as can be seen in Figure 4.3. These artifacts may act as a form of data augmentation which creates more variation in the dataset and by doing so helps PVNet learn.

While PVNet trained on CycleGAN images yield similar result as to when it is trained on datasets including real images for object 14 and 15, this is not the case for object 16. This means that, in general, one cannot expect to achieve similar results when training solely on CycleGAN generated images as to when using real images in the training data. Although not guaranteed, it is possible to slightly improve performance further by combining real images and CycleGAN images.

4.5.3 Translation task for Volvo object

The result in Table 4.3 shows that when PVNet is trained solely on synthetic images it can better generalize to real images if the synthetic images are first processed by CycleGAN. In experiment 4, which is elaborated on below, the applicability of the CycleGAN approach is investigated. To apply this method in practice, it is of great importance to be able to train a CycleGAN model on custom objects and not solely rely on existing datasets. As described in Section 4.2 a dataset containing both synthetic and real images of a metallic U-pipe from Volvo was created. For this experiment, we use the same hyperparameters as in Table 4.1 except for load and crop settings where we load the images at 400×400 resolution and train them on randomly cropped 256×256 patches. We also augment the data by rotating it randomly to increase the number of viewpoints of the object. Based on the result

in Table 4.2 we use roughly 1200 images from each domain to train CycleGAN for this object.

In Figure 4.8 it can be seen that CycleGAN successfully translates the rendering into having similar characteristics as the real dataset, see Figure 4.2. As brought up in Section 4.2 the real training images have jagged edges due to reflecting the green color in the background and poor lightning. These artifacts are also present in the translated images. In Figure 4.9 less successful outputs from the same CycleGAN model are shown where the contour of the pipe is distorted or texture is lost. Since CycleGAN learns a mapping between the synthetic and the real domain based on the images we feed to the network, we believe it is essential to have a consistent representation of the object in each of the two domains. In our dataset the real images did not consistently yield similar object appearances due to the imperfect cropping which we believe is the reason that some angles of the object are better translated than others.

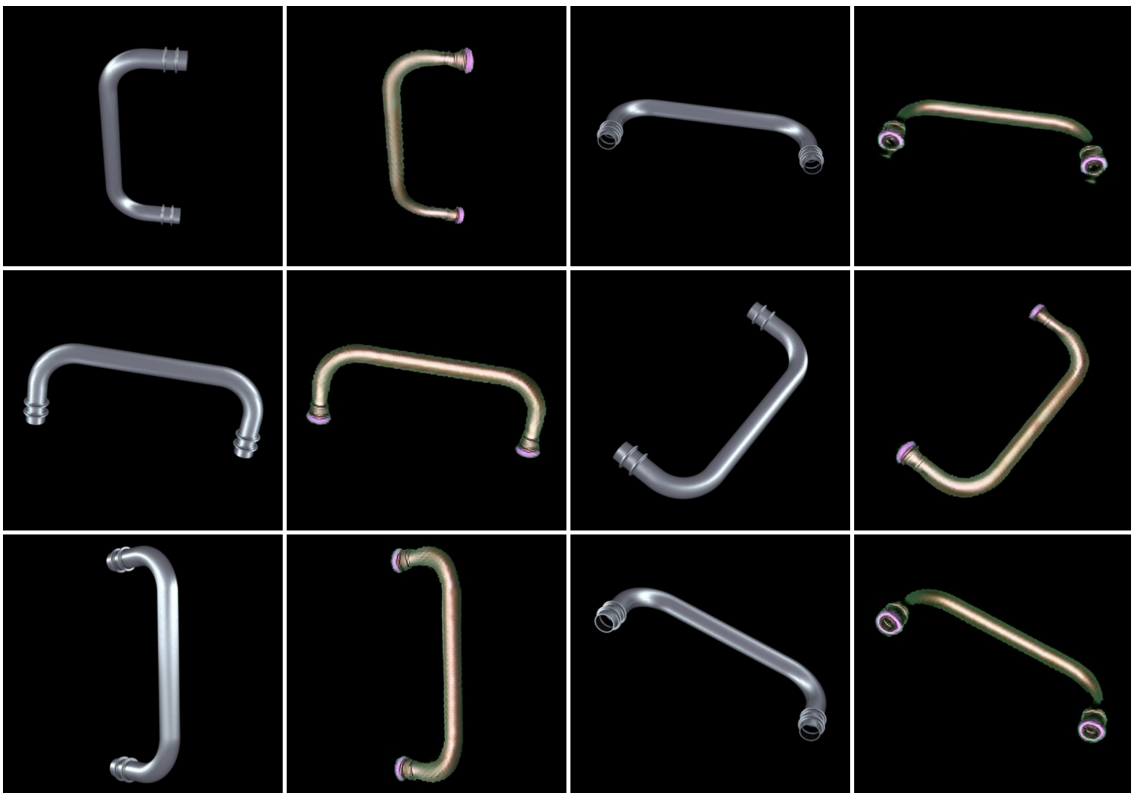

Figure 4.8: Visualization of CycleGAN input/output pairs on the custom U-pipe object. The CycleGAN generated images maintain the original pose, have a similar metallic surface as the real images and manages to produce purple gaskets on each end of the pipe. The CycleGAN generated images have similar artifacts as the real images, such as jagged edges, which can be seen especially in the upper right and lower right pair. Note that the shown pairs are examples of the more successful translations while worse ones are show in Figure 4.9.

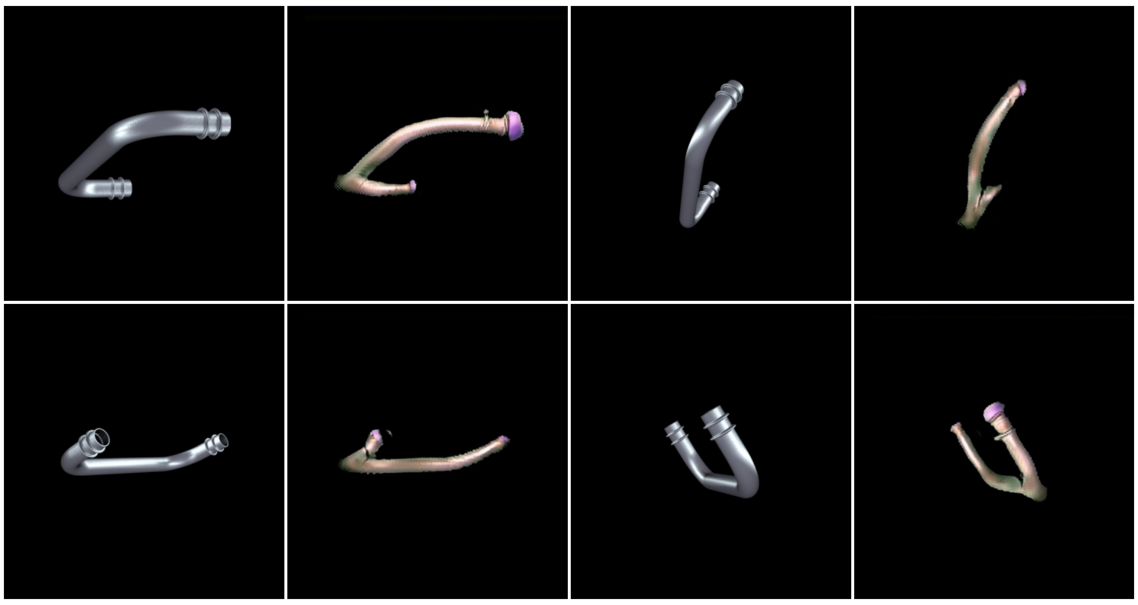

Figure 4.9: Input/output pairs from the same CycleGAN model shown in Figure 4.8. These are some examples of artifacts which distort the geometry or texture of the object making it unsuitable for use as training data.

It is clear from this experiment that CycleGAN manages to generate images of an object which are more representative of the real domain than the synthetic images. In addition to this it can also be concluded that translating a custom object is rather straightforward and can easily be done for additional objects. The results from experiment 3, shown in in Table 4.3, indicates that if the object can be successfully translated the performance of PVNet should be improved. Based on this, if PVNet is to be trained on custom objects without using real images, it should be beneficial to first process the synthetic images through a trained CycleGAN model.

5
Conclusion and future work

In this chapter the conclusions drawn in chapter 4 will be summarized. In addition we discuss contributions of the work, point out shortcomings and provide guidance on how these can be addressed in future work.

5.1 Contributions

In this thesis we show that CycleGAN can learn to translate renderings into more realistic versions while preserving the pose. This means that the original pose-annotations generated from the rendering step is still accurate for the new images generated by CycleGAN. By doing this, we also show that the domain gap between real and synthetic images can be reduced. This was shown by training PVNet solely on synthetic and CycleGAN images respectively and comparing the performance. The comparison showed that PVNet trained using CycleGAN generated images performed better than when it was trained only on the original synthetic images. With these findings we hope our work opens up for new opportunities within research fields or applications where the domain gap is an issue.

5.2 Conclusion

In this thesis the effects of using the generative model CycleGAN to bridge the domain gap between synthetic and real images has been investigated. The intended use for this is to improve the performance of a neural network for pose estimation, in this case PVNet, when it is trained on synthetic images and applied on real objects. We show that CycleGAN can successfully generate images with similar characteristics as the real object while preserving object pose such that the pose-annotations from the renderings can be used. We however fail to create well-functioning CycleGAN models for some of the objects in the T-LESS dataset. We still believe that CycleGAN can learn the mapping for these objects as well by using other hyperparameters, generator/discriminator architecture or by processing the data differently.

For the successfully trained CycleGAN models, we use pose-annotated renderings of three objects in the T-LESS dataset and train PVNet on these renderings. When comparing performance between networks trained on renderings and CycleGAN generated images respectively we show significant improvement according to the ADI metric while only slightly improving performance according to the 5cm 5° metric. We can therefore conclude that at least the translational part of the pose can be improved using CycleGAN images over the renderings. Overall we believe that it would be beneficial to train PVNet on CycleGAN generated images rather than synthetic images as this would yield a better performance. We also show the flexibility of the CycleGAN approach for data generation by using our own renderings and real images of a U-pipe to train a CycleGAN model.

5.3 Future work

In order to continue on this topic in the future the next step would be to to take more professional images of custom objects of interest in order for CycleGAN to better generalize the transformation of the object. Following this would be training PVNet on the custom dataset consisting of the object of interest in order to tell how well it works on a completely custom dataset. In addition to this creating renderings to use for training both CycleGAN and PVNet would be necessary in order to train on a completely custom dataset.

To further investigate the consistency of the approach additional iterations of experiments must be conducted in order to find out how consistently CycleGAN can learn the translation and perform well. Before this, in order to reduce training time, a more through investigation into the number of images required would be necessary. It would be of great value to find a general threshold for how many images, real and synthetic, are needed in order to successfully train CycleGAN. This means that it would be necessary to train and evaluate multiple iterations of the same network in order to find out two things. Firstly the consistency of the approach, how often it performs well, what its peak performance is and what its worst performance is with the same parameters and data. Secondly it would also allow us to find how many images are necessary in order to achieve the desired performance versus the time spent training.

Currently there exists no proper metric to measure the performance of CycleGAN and the best way to know if it performs well or not is by visual inspection. This is inherently a problem when using this method. Creating a form of metric to measure the performance would allow for more automatic training which would be incredibly useful in large-scale applications such as in an industry. In the current state plenty of human interaction would be required during training to ensure that the model learns the desired mapping between the two domains.

As shown in Figure 4.5, not all translations were successful. It would be valuable to know if there are any correlations between successful and failed CycleGAN models. This would provide insight into whether or not this is a viable method for arbitrary objects or if there are certain objects which are not suited for this approach or if we were simply unable to find the correct hyperparameters to train the network.

It would also be of interest to see how the original resolution of the real images impacts the learning of CycleGAN. The images from T-LESS were captured at a higher resolution than the ones taken by us. Before training the resolution of all images were reduced to the same resolution but it would be interesting to investigate if the original resolution has any effect on the training of CycleGAN.

Bibliography

[1] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-less: An rgb-d dataset for 6d pose estimation of texture-less objects," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 880–888.

[2] "Top hat rail dataset." [Online]. Available: https://www.dlr.de/rm/en/desktopdefault.aspx/tabid-11999/

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[4] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570.

[5] T. Opheim and J. Lecerof, "Instance segmentation and pose estimation of assembly parts using deep learning and synthetic data generation," Master's thesis, NTNU, 2019.

[6] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 306–316. [Online]. Available: http://proceedings.mlr.press/v87/tremblay18a.html

[7] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[8] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European conference on computer vision*. Springer, 2014, pp. 536–551.

[9] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[10] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision*. Springer, 2012, pp. 548–562.

[11] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and*

neuroscience, vol. 2018, 2018.

[12] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2107–2116.

[13] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 699–715.

[14] J. Li, P.-L. Götvall, J. Provost, and K. Åkesson, "Training convolutional neural networks with synthesized data for object recognition in industrial manufacturing," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 1544–1547.

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

[16] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[17] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.

[18] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Mpiigaze: Real-world dataset and deep appearance-based gaze estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 1, pp. 162–175, 2017.

[19] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3722–3731.

[20] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," *arXiv preprint arXiv:1409.7495*, 2014.

[21] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[22] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[23] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *2011 international conference on computer vision*. IEEE, 2011, pp. 858–865.

[24] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1941–1950.

[25] F. Manhardt, W. Kehl, N. Navab, and F. Tombari, "Deep model-based 6d pose refinement in rgb," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 800–815.

[26] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 683–698.

[27] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.

[28] K. Kurach, M. Lucic, X. Zhai, M. Michalski, and S. Gelly, "The gan landscape: Losses, architectures, regularization, and normalization," 2018.

[29] "Pvnet github repository," https://github.com/zju3dv/clean-pvnet, accessed: 2020-02-25.

[30] M. Oberweger, M. Rad, and V. Lepetit, "Making deep heatmaps robust to partial occlusions for 3d object pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 119–134.

[31] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3828–3836.

[32] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[33] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[34] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[35] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 3485–3492.

[36] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[38] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[39] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2930–2937.

[40] Z. Li, G. Wang, and X. Ji, "Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[41] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis *et al.*, "Bop: Benchmark for 6d object pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.