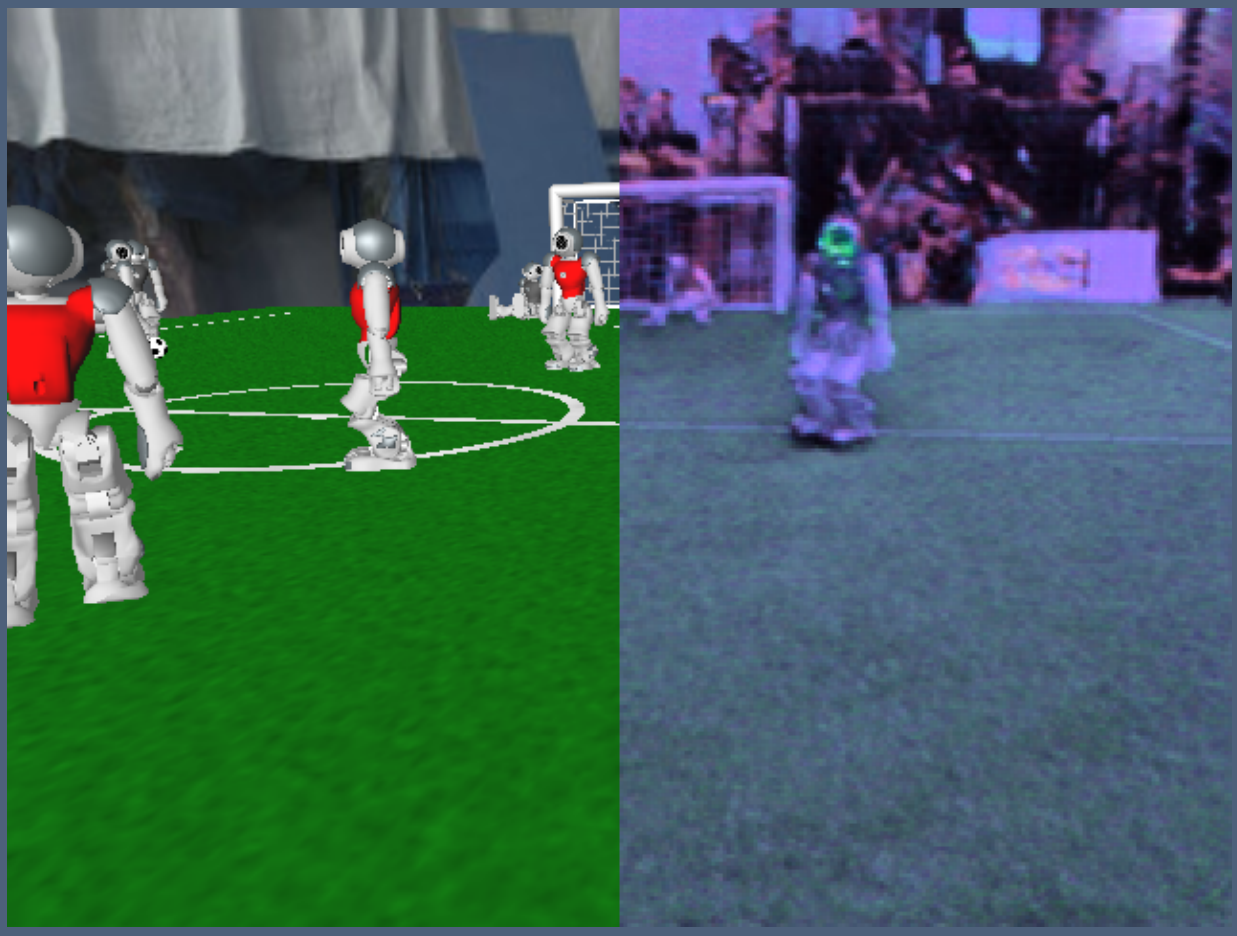
Enhancing simulation images with GANs



Layout: typeset by the author using LaTeX.
Cover illustration: Hidde Lekanne gezegd Deprez

Enhancing simulation images with GANs

in the context of the RoboCup Standard Platform League

Hidde G.J. Lekanne gezegd Deprez
11348062

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

July 3rd, 2020

Contents

Abstract

This thesis addresses the problem of generating realistic images from simulated environments within the Standard Platform League of the RoboCup. Automatic generation of realistic images would be beneficial in the process of developing machine learning algorithms. This is because when training those algorithms, large datasets are needed with annotations. In the case of automatic generation, this could be provided from the ground truth of the simulation model. For the Standard Platform League these developments are needed to increase the performance of the ball and/or opponent detection. The approach taken in this thesis, enables researchers to enhance the realism of an existing simulator simply by collecting images from the real world. Then use those images as training example to enhance simulated views. In order to organize such a process, a translation function between simulated images and real images needs to be explored and discovered. Such a function can be learned by generative adversarial networks.

In order to develop a workable approach, the objective function has to be clearly specified. Otherwise the translated images are not as diverse or accurate as real images. Various architectures such as CycleGan and MUNIT have been applied in previous work to create such transformations. Both architectures are adjusted in this thesis to the needs within the Standard Platform League. After tuning the loss functions and modifying the architectures, better results are obtained compared to the base implementations. Our approach significantly reduces the perceptual distance between a real image and a simulated image. Additionally, the impact of this approach is demonstrated by fact that a machine learning algorithm, such as an image-instance segmenter, performs notably better when trained on the translated images compared to being trained on the purely simulated ones.

This work was presented at the Virtual Humanoid RoboCup Open Workshops event on the 27th of June 2020[1].

[1] https://humanoid.robocup.org/virtual-rohow-2020/program/

1 Introduction

Simulation has been an important component in the development of artificial agents which act in the real world. According to Zlajpah [1] the ability of simulated environments to predict events faster than real-time without contracting damages makes the simulated environment faster and cheaper compared to real world testing. For this reason, Zlajpah claims simulations have been extensively used throughout last decades. In the field of Artificial Intelligence, simulations can be used to train a neural network while some hand-picked real situations can be used to fine-tune and/or evaluate the network. In the context of image-instance segmentation, simulations are desirable because the cost of a large manually annotated dataset could be significant: manual annotation of real images takes much longer than automatic annotation of synthetic ones. Furthermore, the automatic annotations are always accurate, unlike humans who occasionally misclassify or misannotate. Yet the simulation has to overcome a reality gap, specifically a loss in performance of applications developed in the simulation when they are applied in the real world. This thesis aims to reduce this reality gap with a method based on Generative Adversarial Networks, as explained in Section 3.

In recent years the use of simulation for computer vision has gained the interest of the scientific community in the Standard Platform League (SPL)[2]. This is a competition within the yearly RoboCup event[3]. Here, teams from different universities program the same type of robot, currently the Nao robot[4], to play several competitive football games autonomously.

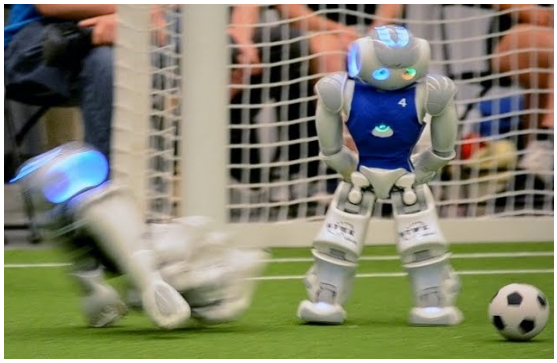


Figure 1: Screenshot from 2018 Final of the RoboCup Standard Platform League.

[2] https://spl.robocup.org/
[3] https://www.robocup.org/
[4] https://www.softbankrobotics.com/emea/en/nao

This interest within the SPL community for the use of simulation for computer vision is due to advances in methods; such as the developments in Convolutional Neural Nets (CNNs). In addition advances in hardware with the introduction of the Nao V6 increased that interest too. Last year a CNN named JET-Net was released, it performs bounding-box object detection on a image of 80x60 [2]. When combined with the JIT-compiler; a neural network compiler specifically made for the Nao V6, this CNN can be run within 1.995 milliseconds on the Nao V6 [3]. This development has solidified the applicability of object detection with neural networks in the football competition, where each frame is generally processed within 16.67 milliseconds. Notably the authors of the CNN paper used a Generative Adversarial Network (GAN) to augment simulated images to be useful for training the object detector in estimating distances to detected objects. The authors applied both simulated and real images as input and translated the input to an imaginary space where the origin domain couldn't be detected. This method however, was not used to train the object detector from scratch. A dataset of real images was used to train the object detector while the imaginary space was only used to let the robot identify features useful for distance estimation.

Another team has developed a realistic stochastic scene-render in Unreal Engine 4 [4]. They provide ample evidence that the rendered images generalize well to the real world by running a detection benchmark trained with only the simulated data on real data. This application however, is not an simulation and can only render realistic looking static scenes. Thus current approaches either rely on static scene rendering or try to find a image space between real and simulated space. Thus a complete translation from simulation to real has not been attempted in this field. Within this thesis a method is proposed to make such a translation.

1.1 Image-to-Image Translation

Within the GAN community, Image-to-Image translation is a topic within the GAN community centered around translating an image into another domain with a restriction [5]. For example a restriction to change the underlying information of the image can be enforced. This will make the GAN translate the image to a different style. For instance, this happens within networks capable of photo to painting translations, which take a photo and stylize it to fit a painters profile [6]. The translation can also be semantic, for example translating a cat to a dog but with the same fur color and pose [7]. The problem posed in this thesis is somewhere in between these two kinds of translation tasks. Robots in the real world experience different lightning conditions than in the simulation, making a style translation necessary. Additionally detected objects such as other robots, goalposts and

balls should only change slightly and should not be transformed into other kinds of objects. However, the background, is semantically different between simulation images and real images as there are different objects represented. Furthermore, the grass of the field is arguably semantic, as grass can differ in height and types of blades.

1.2 Paired and Unpaired Training

Besides a difference in what an architecture can translate, the method of training can make another difference. An architecture can be trained either paired or unpaired. The paired training provides the exact translations the neural net should make. Training in paired form is easier as there is no ambiguity about what the output should be. There are numerous applications for this including sketch to handbag, topography to aerial picture and segmentation to image demonstrated by pix2pix [8]. Yet, paired data is not always readily available or easy to obtain. In the case of a simulation, it requires simulated images to align perfectly with the real images. Since robots are in motion during an actual game and the simulation isn't perfect, this alignment process is unfeasible with currently available simulations. These simulations are discussed in section 2.1. Hence, in this thesis unpaired training is used. Unlike paired training, unpaired training involves a significant challenge: somehow the neural net needs to find out the relation between different domains without being explicitly shown how they map on to each other.

1.3 Problem Definition

One of the benefits of using GANs to enhance simulation images is to enable researchers to use simulations with low detail renderers to develop competitive computer vision algorithms from simulation with a minimal reality gap. Additionally any current simulation can be extended by this approach after collecting unpaired image datasets. Therefore the research question is: *"How far can the gap between simulation and real images be bridged through the use of GANs in the context of Robocup SPL?"*. The aim for this thesis is to identify the challenges to overcome the reality gap with GANs and provide benchmarks for future work. For this goal a dataset is procured consisting of images obtained at RoboCup games taken from the Nao-v6 robots. Additionally a simulator is chosen and augmented in order to obtain simulation images. Thereafter, a number of GANs are trained and evaluated. The evaluation is done by three metrics; the Fréchet Inception Distance (FID); the performance of a state-of-the-art instance segmenter Mask-RCNN, when the translated

images are fed into this segmenter; and the performance of a state-of-the-art panoptic segmenter, when translated images are fed in this segmenter.

2 Theory

To understand the different aspects of the architectures proposed in this thesis, some concepts have to be explored. Firstly the simulator needs to be explained as the images gathered from this simulator will have an effect on every application using those images thereafter. Secondly two key components of the proposed architectures are explained. These components are the generative adversarial network architecture and the variational autoencoder. Afterwards the two proposed architectures CycleGan and MUNIT are explained in more detail. Additionally they are contrasted to each other. Finally the different evaluation methods are explained in detail. The Fréchet Inception Distance used to calculate distances between two datasets, the MASK-RCNN instance segmenter used to evaluate the machine learning potential of instance segmentation and the panoptic segmenter which is able to make full use of the annotations.

2.1 SPL RoboCup Simulations

Related to the SPL RoboCup are a number simulations made and maintained by either competing teams or researchers without a team. To the knowledge of the author simulators are available from the Berlin United SPL-Spark[5], the Nao-devils SimRobot version[6], the HULKS SimRobot version[7], the B-Human SimRobot version[8] and the ROS Gazebo plugin[9]. The simulator chosen, for reasons outlined in section 3.1, is the HULKS version of SimRobot. This simulator was originally developed by the B-Human team [9] and later adopted by the HULKs. The HULKs added motion blurring to the simulation as well as connecting it to their own robot behavior modules. The simulator aims to be a realistic simulator where every sensor has a realistic input. The rendering is done with the OpenGL library and only contains assets for the goalnet, ball, robot and grass. Other objects seen in the simulation are simple 3D shapes. The simulation has a rigid body physics engine where objects can collide with other objects in a realistic way. Due to these features realistic robot behavior is possible to simulate in this simulator.

This simulator is mainly used to test and develop new algorithms used in the SPL league. This particular version has only been used by the HULKs as a debug tool, but similar versions have been used in training object detectors [2].

[5] https://github.com/BerlinUnited/SimSpark-SPL
[6] https://github.com/NaoDevils/CodeRelease
[7] https://github.com/HULKs/HULKsCodeRelease
[8] https://github.com/bhuman/BHumanCodeRelease
[9] https://github.com/ros-naoqi/nao_virtual

2.2 Generation Adversarial Networks (GANs)

The first Generative Adversarial Network was an architecture with two neural networks, a generator which tries to construct an image from randomized noise and a discriminator which tries to determine whether an image is authentic or generated [10]. The discriminator activation can be used as a learned loss function for the generator. This results in a loss function of $max_G\ min_D(E_x[log(D(x))] + E_z[log(1 - D(G(z)))])$.

1. G is the generator.
2. $G(z)$ is the generated output given latent vector z.
3. D is the discriminator.
4. $D(x)$ is the prediction of the discriminator whether an image is real or not.
5. max_G and min_D means that the generator tries to minimize this, and the discriminator tries to maximize this function.
6. E_x is the expectation of real data sampling with x.
7. E_z is the expectation of latent data sampling with z.
8. $E_x[log(D(x))]$ is the expectation of the discriminator correctly identifying real images.
9. $E_z[log(1 - D(G(z)))]$ is the expectation of the discriminator correctly identifying the generated images as fake.

For the discriminator D the ideal case is to identify every real image as real and every generated image as fake, resulting in a score of two. For the generator G it is ideal to confuse the discriminator to such a degree that it misclassifies all samples z, though realistically the score will not be lower than real/fake ratio, since the discriminator can guess always real or always fake based on this ratio.

2.3 Variational Auto Encoders (VAEs)

Recent work has used VAEs in combination with Variational Auto Encoders [11]. Likewise, the image translation architectures used in this thesis use VAEs to encode and decode to latent spaces. A variational autoencoder is an extension of the autoencoder. This autoencoder is a neural network which aims to represent input data in a latent space. This

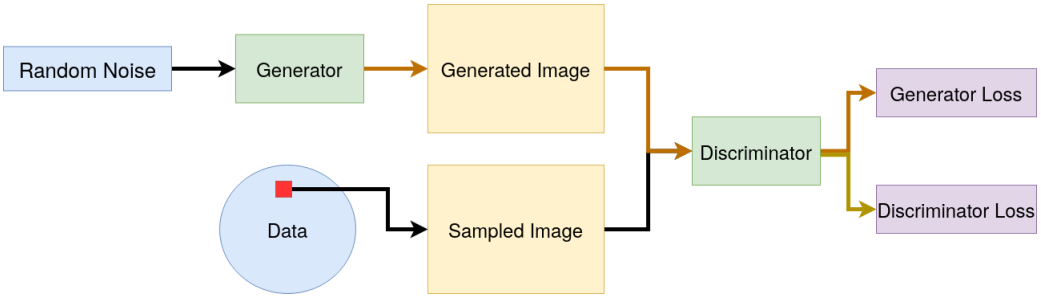


Figure 2: The GAN architecture [10], the brown arrows indicate the generator backpropagation and the yellow arrows indicate the discriminator backpropagation.

latent space usually has a lower number of dimensions than the input [12]. An autoencoder exists out of an encoder and a decoder; encoders transform the input to the latent variables and the decoder transforms the latent variables to a reconstruction of the input. The error is the difference between an input image and the reconstructed image, usually defined as $||A - A'||^2$ where A is the input and A' the reconstruction. This makes autoencoders self-supervised, as the input is also the desired output. These autoencoders are effective at compression and denoising. Generating samples however, presents an issue: the latent space isn't necessarily regular, and assuring regularity is not trivial [13]. A non-regular latent space is difficult to sample from as the mean of all objects within a class may not be of that same class. Variational Auto Encoders (VAEs) assure that an attempt is made by the autoencoder to have a regularized latent space [13]. This is achieved by defining the latent space in distribution parameters, usually normal distributions with a mean and variance. From this space, meaningful samples can be generated by sampling from neighborhoods of encoded images. This generation method can be enhanced by using a discriminator and treating the decoder as an generator.

2.4 CycleGan

The CycleGan [14] architecture exists out of two VAEs, noted as encoder-decoder pairs in figure 3, and two discriminators. In this architecture three different loss functions are optimized. Firstly the adversarial loss is defined the same as in section 2.2, with the VAE defined as the generator. Secondly there is the cycle consistency loss, which is the distance between a reconstructed image after being translated twice. In formal terms this is $G_A(G_B(A))$ where G_A and G_B are generators for domain A and B respectively. Thirdly there is the identity loss, which is the difference between the input and output of a generator given a image in the same domain as its output domain. Formally this can be written

down as $G_A(A) \approx A$. All formulas have the reverse version where domain A and B are switched.

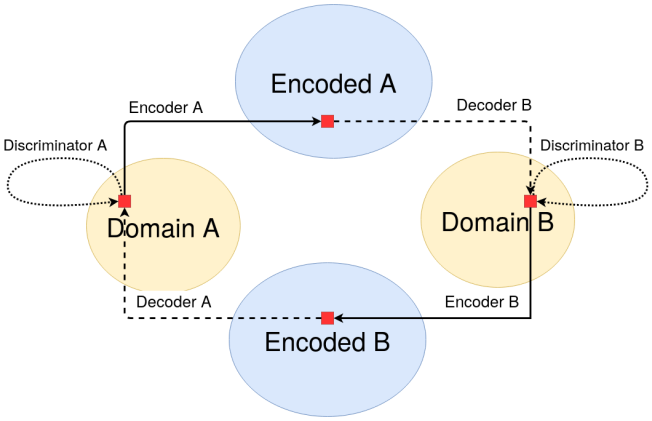


Figure 3: The CycleGan architecture [15]

2.5 Unsupervised Image-to-Image Translation (UNIT)

Similar to CycleGan, the UNIT [15] architecture exists out of two VAEs. In this architecture however, they share a latent space as well as share weights of some layers. The layers which share weights are the last few layers of the encoders and the first few layers of the decoders, how many may change depending on the task. The UNIT applies the same discriminator loss and same reconstruction loss as CycleGan. However the UNIT does not have a identity loss like CycleGan.

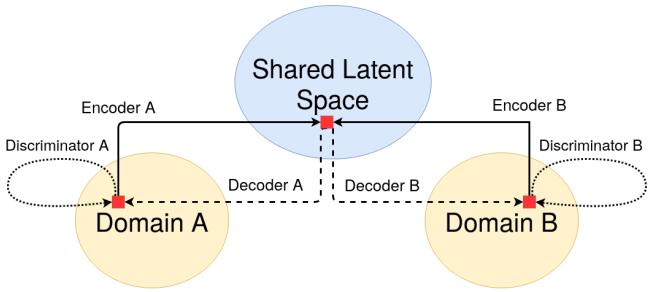


Figure 4: The UNIT architecture [14]

2.6 Multimodal Unsupervised Image-to-Image Translation (MUNIT)

MUNIT [16] is an extension of UNIT where domain specific style code is added. The idea of the authors is that with this extension semantic information and style information are disentangled. The underlying assumption is that a domain transfer can be formulated in the transfer of semantic information and an augmentation according to domain specific style related information. According to this statement, any information in the shared latent space should be semantic. Likewise, any information in the style space should only capture the style of the image. Because the style code is domain specific, it can be sampled unlimited times per translated image. This makes more diverse and multi-modal translation possible.

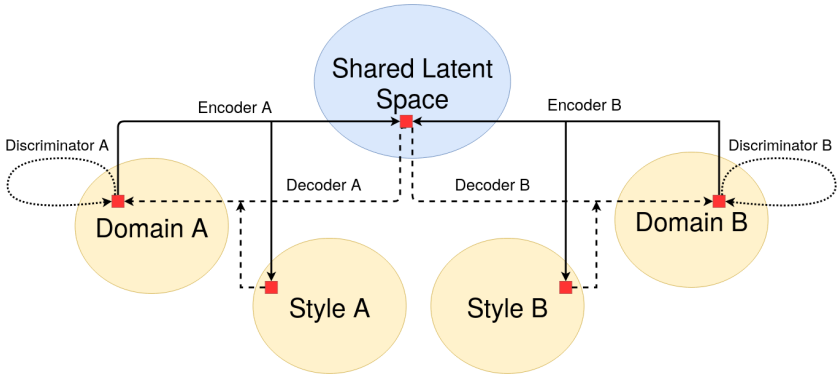


Figure 5: The MUNIT architecture [16]

2.7 Fréchet Inception Distance (FID)

FID is a perceptual distance measure, which can be used to evaluate the quality of a generated dataset. This measure is inspired by the Inception Score (IS) which is a score which increases with when meaningful features are present [17]. The authors of the IS define meaningful as low entropy in the label distribution. They also define diversity as high entropy in the chance of a label distribution given any random noise vector processed by the generator. The label distribution is generated by Inception Net [18]: an image classifier trained on the ILSVRC-2012 dataset [19]. Most importantly, this score correlates with human perception of image quality [17]. Thus this method can evaluate a translated dataset without the need of the original one. The authors of the FID however, noted several unintuitive results coming from this measure and therefore proposed a new measure [20]. The FID take the internal code of Inception Net and calculates the probability of that inter-

nal code. This probability is approximated by a multivariate Gaussian distribution. This distribution is calculated for two datasets, the translated one and the original one. Finally the Fréchet distance is used to calculate the distance between the two distributions.

$$d^2((m, C), (m_w, C_w)) = ||m - m_w||^2 + Tr(C + C_w - 2(CC_w)^{1/2})$$

1. m and m_w are the means of feature vectors of the real and translated datasets respectively.
2. C and C_w are the covariance matrices of the feature vectors.
3. $||m - m_w||^2$ is the mean distance between the two datasets.
4. $Tr(C + C_w - 2(CC_w)^{1/2})$ is the difference in covariance.

This measure can be used to estimate the distance between two datasets based on features used by Inception Net. This is unlike pixel based methods which can have numerous unintuitive results. For example, slight movement of objects can have significant impact on pixelwise distances while perceptually the difference is inconspicuous.

2.8 Mask-RCNN

Mask-RCNN is instance segmenter: a CNN trained to detect bounding boxes and segmentations of objects [21]. This architecture is build out of four main components, a feature extractor, a bounding box proposer, a bounding box classifier and a bounding box segmenter. The feature extractor can be any CNN, but usually architectures like VGG [22] or ResNet [23] are used, in the case of Mask-RCNN it is ResNet. ResNet is extended to a Feature Pyramid Network (FPN) which produces feature maps on different scales. Layers on top of the pyramid contain high level features, layers at the bottom contain low-level features. This is the backbone of Mask-RCNN. This FPN is then used to propose Regions of Interests (ROIs) by the Region Proposal Network (RPN). Afterwards these ROIs are classified and aligned onto the feature maps. The ROI proposal and classifications are part of the faster-RCNN network [24]. The authors of Mask-RCNN extended this architecture such that the ROI alignment with the feature layers is accurate enough for segmentation. The mask and bounding box prediction are the head of the Mask-RCNN network.

This network has been implemented in the detectron2 [25] repository of Facebook research. This repository hosts a number architectures capable of object detection, segmentation and pose estimation. It is also benchmarked as one of the faster frameworks[10].

[10] `https://detectron2.readthedocs.io/notes/benchmarks.html`

When evaluating the MASK-RCNN the main measure used is the Average Precision (AP) of both bounding boxes and segmentations. The AP is the area under the recall-precision curve found when plotting precision against recall. For both bounding boxes and segmentations the Intersection over Union (IoU) cutoff is an important factor. The IoU is the area of the intersection of two shapes divided by the union of both shapes. Generally a fixed IoU is chosen as the cutoff point for accurate detections. For the COCO annotation used in this thesis however, ten IoU cutoff points are used. They range from 0.5 till 0.95 with a step size of 0.05. For each step the AP is calculated with that step as a cutoff point, for each class these ten APs averaged to form a class AP.

2.9 Panoptic

The task of panoptic segmentation is the combination of previously mentioned instance segmentation and semantic segmentation. Where as instance segmentation identifies different instances of the same class, semantic segmentation does not. For classes such as "background" such semantic segmentation is a better fit than instance segmentation as there is only one background in any image. These semantic classes are commonly referred to as "stuff" whereas instance classes are commonly called "things". For panoptic segmentation, the previously mentioned FPN (Section 2.8) is extended with a semantic segmentation output layer which works in parallel with the instance segmenter [26]. This network is implemented in the Detectron2 framework similar to the Mask-RCNN instance segmenter [26].

3 Method

Now that the theoretical foundations are established, the precise methods used in this thesis can be explained in detail. For this thesis a dataset had to be gathered existing of simulation images and real images. Additionally to encourage reproducibility, code and data is released. Furthermore the exact training methods with their hyperparameters are disclosed for both the GANs and the segmenters. Finally the calculation of the FIDs are explained.

3.1 Dataset

For the purpose of this thesis, some adjustments have been made to the simulation[11]. A skydome texture was added to provide complex features which an unsupervised GAN can match with the complex background real robots observe. The previously single color background provided problems in two-fold. Firstly the mismatch between the simulated background and the real background makes the GAN rely on the perspective bias to guess where the background begins. Secondly, robot features represented in the latent space double as background features. This causes robots to turn into pillars or blurs when translating from simulated to real.

Another change was made to the native semantic segmentation camera, which produces segmentations of objects the robot can see. This module being rewritten to give each individual component an unique ID, resulting in over 700 different components. This enables more control compared to the 26 colors the default module works with. Finally the necessary code was inserted to save images from the camera feed and link them to their annotations.

The semantic segmentation was converted to two formats: instance segmentation and panoptic segmentation. Instance segmentation contains polygons for every object in the image, objects may overlap if necessary. Panoptic segmentation contains an image where each class or instance has a different id. In this type of annotations, no overlap is possible.

3.1.1 Accessibility

The simulated dataset is published together with all translation models and the panoptic annotations [27]. Furthermore the code used in this thesis is available at `https://`

[11] `https://github.com/HiddeLekanne/HULKsCodeRelease`

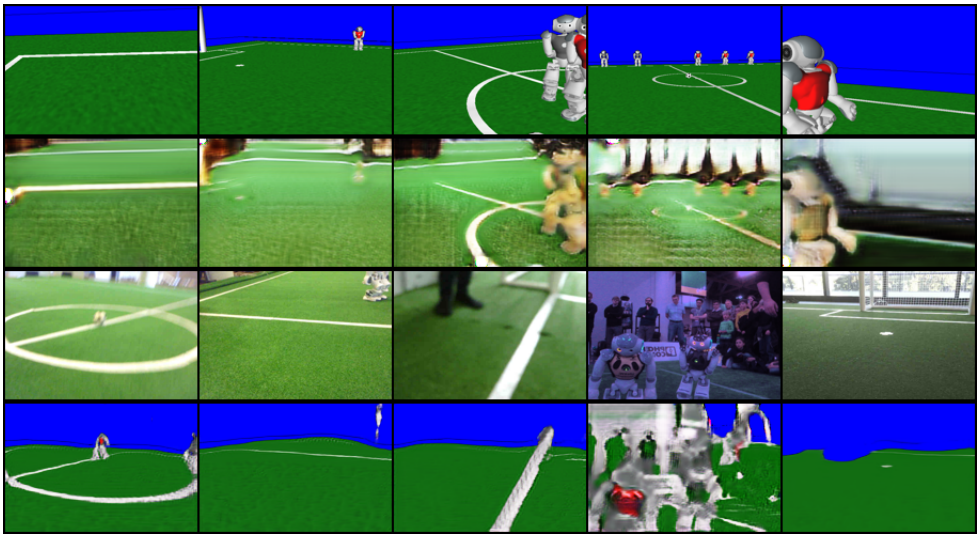


Figure 6: CycleGan early experiments. Due to the lack of features, a difference in perspective (camera angle) is contributing significantly to the translation errors

`github.com/HiddeLekanne/Robocup-SPL-Simulated2Real`.

3.2 GAN Training

The different architectures are all trained with the same data, 9040 pictures of synthetic data and a random subsample of 10.000 of the real data. The validation set is 10% of the total amount of data in both datasets. All translation examples shown in this thesis are from the evaluation sets. Training time for each architecture was 50 epochs of full dataset with batch size 2. This amounted to 200.000 iterations which is comparable with the original iteration numbers of CycleGan [14] and MUNIT [16]. The GPU used was a Geforce RTX 2018 Ti running on CUDA version 10.1 and Torch version 1.4. For both architectures the Pytorch implementations were used[12]. The image size was 320x240 which is a downscale of a factor of two from 640x480. Parameters not addressed in this section remained the same as in the original implementations.

[12] `https://github.com/eriklindernoren/PyTorch-GAN`

3.2.1 Adjusted Loss

For both architectures an adjustment for the loss function has been implemented. For CycleGan the cycle consistency loss has been amplified to five times the normal rate. Additionally, the loss was only applied to the simulation-real-simulation case, not the real-simulation-real case. The amplification without this modification causes in-field distortions, like referees walking on the field, to be represented as special cases of robots. This made some robots in simulation turn into black blurs when the GAN tries to translate them. The MUNIT architecture got the same adjustment.

3.3 Instance Segmenter

The instance segmenter is trained on 8136 images. These are the same images used for training the GANs. The instance segmenter is trained five times; four times on translated simulation images by four different GAN configurations and once on the original simulation images. It is evaluated twice per dataset; once on a size 904 evaluation dataset originating from the simulation and once on a size 500 dataset of manually annotated real images. Similarly to the training sets, the evaluation sets are part of the evaluation sets used for the GANs. The framework used is the Detectron2 [25] as described in section 2.8. The specific architecture chosen was the Mask-RCNN with a FPN backbone and ResNet with 51 layers. According to the authors of Mask-RCNN the FPN is to be the best method to structure the feature layers from ResNet [21]. For the size of the backbone, the large ResNet backbone of 101 layers overfits slightly more compared to the 51 layers ResNet for the datasets used in this thesis, see appendix A. The shorthand for this configuration is mask_rcnn_R_50_FPN_1x. A segmenter is trained in the Detectron2 framework [25], it is trained for 80.000 iterations. The instance segmenter is tasked to predict the fields, goalframe, goalnets, robots, balls, lines and backgrounds. However only balls and robots are evaluated in the manually labelled datasets. To evaluate the instance segmenter both the bounding box and the segmentations are used.

3.4 Panoptic Segmenter

The panoptic segmenter is trained the same way as the Mask-RCNN with some notable differences. The same backbone was used except that it was extended with semantic segmentation as described in section 2.9. A different training-validation split on the same dataset was used. This different split was due to the different kind of annotation needed for this task making the previously dataset incompatible. Furthermore the categories goalframes, goalnets, robots and balls were designated as instance segmentations ('things') while background, field and lines were designated as semantic segmentations ('stuff').

3.5 FID

The Fréchet Inception Distance (FID) is calculated on the training sets and the validation sets as presented in Table 1 and 2 of Section **??**. Not all possible combinations of datasets are calculated, only combinations with an simulation or real domain and a translated domain are calculated. The distances to the simulated images are calculated for measuring feature retention. At the same time it measures style difference. The distances to the real images are about feature approximation, which is the reality gap defined by the FID. For the CycleGan architecture, only one image is generated per simulation image as CycleGan doesn't have diversity in its output even with random noise [16]. The MUNIT architecture is configured to generate five images per simulation image with random uniform style parameters.

4 Results

For the results. Firstly a qualitative evaluation of the different GANs is done. This gives an intuitive way to see the results of the different architectures. Secondly the FIDs between translated domains and simulated or real domains are compared for more robust evaluation. Thirdly both instance and panoptic segmentation architectures trained on the different translated datasets are evaluated on the real dataset in order to investigate the remaining reality gap. Finally some qualitative evaluation is done on the panoptic segmentation results to further identify some effects different GAN architectures can have.

4.1 Samples

The samples shown here are a randomized sample from the validation set for qualitative evaluation of the different GAN approaches described in Section 3.2.

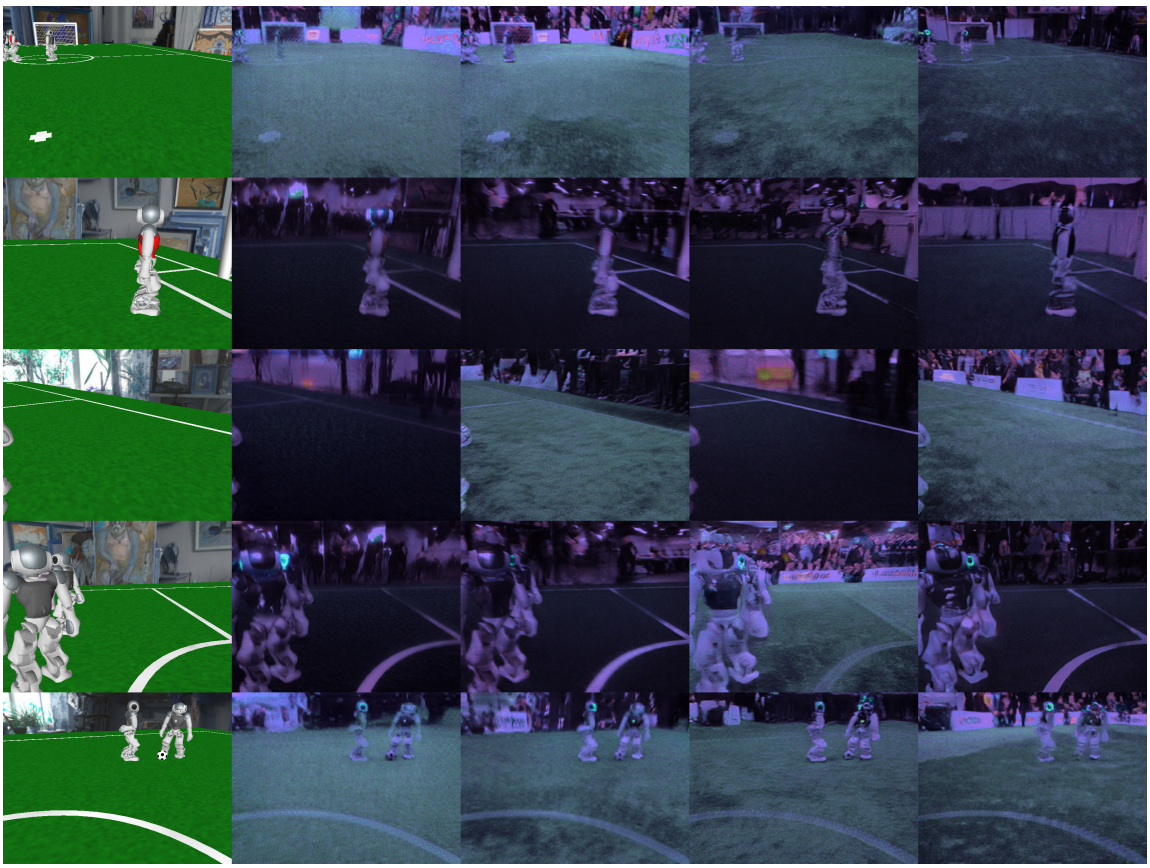


Figure 7: Random sample translated by different generators, from left to right: Original image, CycleGan, CycleGan-adjusted, MUNIT and MINUT adjusted loss.

The first sample (Figures 7) consists of different translations made by the different GAN architectures. The different Generation Adversarial Networks broadly produce the same results. The color composition of the real images is reproduced very well in the translation process. Additionally the background of the images look representative of the real dataset to such a degree that a segmenter could learn to recognize it as such. Furthermore field features are mostly accurately represented. Some exceptions to this are in dark images cases where some line segments seem to blur black. These can be found in Figure 7 column 2, 3 and 5 of row 4. Futhermore the MUNIT light images have the problem that lines and balls sometimes blend with the field. This can be found in Figure 7 column 5 row 2 and 5. Note however, that the MUNIT architecture is difficult to compare since the style code is different between the two versions. The same style effect can be expressed by different style parameters when different versions of the same architecture are used, thus a one on one comparison is impossible. For CycleGan such a comparison is possible in Figure 8.

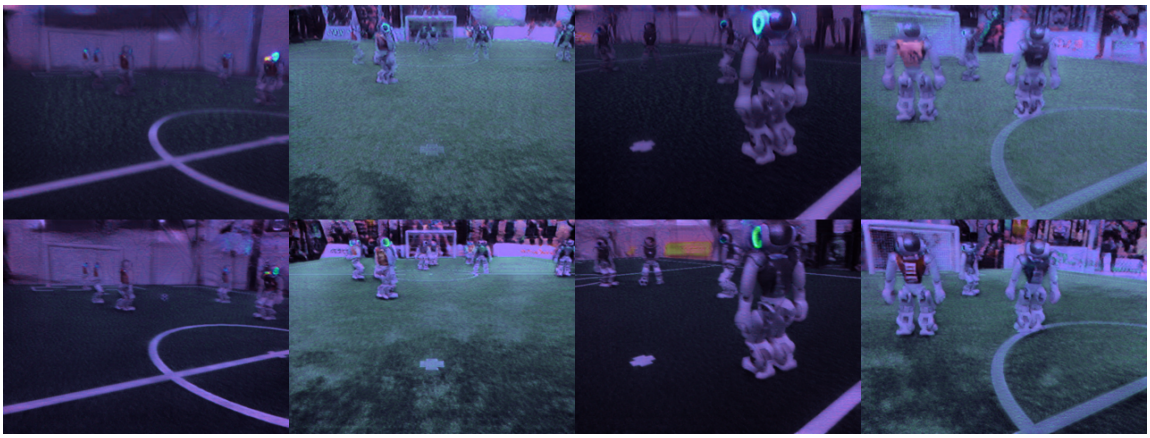


Figure 8: CycleGan normal at the top, adjusted loss at the bottom. Objects are more clearly defined in the adjusted loss results.

The second sample (Figure 8) showcases the difference of the adjusted loss makes on the CycleGan architecture. The top row showcases generally less clear images and sometimes disappearing features such as the ball in the first image or robot legs in the second image. The question remains if this adjusted loss has the same effect in a multimodal approach.

In the multimodal approach (Figure 9), comparison is difficult, as the same style parameters have different effects in different network instances. Therefore it is difficult to qualitatively evaluate the difference. It may be noted that in general more robot specific features

Figure 9: MUNIT normal at the top, adjusted loss at the bottom.

are visible in the far away robots when looking at the adjusted loss generated images. Examples of this are column 2, 5 and 6. An exception to this is column 3 has a particularly powerful style augmentation, where everything changed to field.

4.2 Fréchet Inception Distance (FID)

The FID score as described in Section 2.7 measures the distance between two datasets. Therefore it is a useful metric to see how close the translated images are to the target dataset, and how different they are from the original dataset.

Table 1: FID of train dataset and translation GAN combinations (lower is better).

	Simulation	Real
CycleGan	207.270	52.773
CycleGan-adjusted	**176.545**	38.085
MUNIT	212.587	38.982
MUNIT-adjusted	202.637	**28.096**

The FID scores seem to follow a trend where CycleGan has the highest distance to the real dataset and MUNIT-adjusted the lowest. Interesting result is that the adjusted loss not only lowers the distance to the simulation dataset but also to the real dataset. But this result might not be generalizable to the whole dataset. Therefore the validation sets are also compared.

Table 2: FID scores of validation dataset and translation GAN combinations (lower is better).

	Simulation	Real
CycleGan	233.825	75.017
CycleGan-adjusted	**203.249**	61.012
MUNIT	234.606	49.497
MUNIT-adjusted	224.423	**39.271**

For the validation sets the same trend shows as seen with the training sets. This provides evidence that the this trend holds true for the whole dataset too. Also note the absolute difference in the results between the train and validation datasets; this is due the statistical smoothing effect of increasing sample sizes, lowering absolute scores for smaller datasets.

4.3 Instance Segmentation Results

The FID scores do not evaluate the results a machine learning algorithm might have when trained on the translated set. To evaluate this a segmenter was trained on the four different

translated datasets and the original one. Both the bounding box and the segmentations were evaluated on Average Precision as described in Section 2.8. The reality gap is defined as the difference between the best performing simulated validation APs and the best real validation APs. The reality gaps of both bounding boxes and instance segmentation are averaged for the summarized reality gap.

Table 3: Bounding Box AP on translated data set (higher is better).

	lines	robots	goalframe	goalnet	field	ball	background
Pure Simulation	**73.613**	**83.434**	**87.585**	**86.707**	96.216	**78.754**	**93.675**
CycleGan	69.330	77.351	83.326	81.659	**96.588**	69.965	92.647
CycleGan-adjusted	71.165	79.587	84.474	84.124	96.177	72.626	93.241
MUNIT	63.616	69.561	76.675	75.303	95.538	51.413	90.075
MUNIT-adjusted	65.472	72.382	79.545	76.755	95.835	61.883	90.774

The bounding box APs in Table 3 show the highest results for instance segmenters trained on purely simulation, evaluated on pure simulation. The easier categories seem to be the field and background, but this is also due to their relative size on the images. Whereas balls occupy relatively small bounding boxes, small bounding box errors result in larger AP loss compared to the same absolute errors occurring with field and background categories. To evaluate the reality gap on a deeper level, where individual pixels matter, segmentation APs are used.

Table 4: Instance segmentation AP on translated data set (higher is better).

	lines	robots	goalframe	goalnet	field	ball	background
Pure Simulation	4.603	**72.623**	**50.758**	**81.292**	**92.158**	**80.130**	**92.364**
CycleGan	4.76	64.572	41.892	73.729	92.424	70.392	90.300
CycleGan-adjusted	**4.88**	67.298	44.736	76.739	92.539	72.374	90.936
MUNIT	3.803	54.248	29.565	63.974	90.921	49.814	86.036
MUNIT-adjusted	4.201	58.058	33.557	66.484	91.227	61.317	87.351

The segmentation APs in Table 4 are generally lower, as the task is more difficult compared to predicting bounding boxes. Notably the lines have terrible segmentations compared to their relatively good bounding box APs. Segmentation seems especially hard for this category. Furthermore there is a downward trends in both the bounding boxes and the segmentation APs where multimodal worsens the APs and adjusted loss betters the APs. This holds true for the robots, goalframe, goalnet, ball and background categories. These

scores give a mixed evaluation of the difficulty of the dataset and the accuracy of the annotations after translation. To evaluate the reality gap, additional evaluation needed on an manually annotated real dataset.

Table 5: Bounding Box AP on real data set (higher is better).

	robots	ball
Pure Simulation	0	0
CycleGan	62.322	44.689
CycleGan-adjusted	63.075	39.479
MUNIT	67.020	**48.522**
MUNIT-adjusted	**67.946**	47.251

The reality gap has been evaluated for the two categories that were manually annotated: the robots and ball as seen in Table 5 and 6. When evaluating on the real dataset which is described in Section 3.1, the naive approach of training a segmenter in simulation without translating proves disastrous. The segmenter classifies everything as background, this issue could be solved by removing the background category, but that kind of accidental generalizability runs contrary to the aims of this thesis. The APs in Table 5 are notably better when trained on the multimodal approaches, differing an average of 4%. The gap between these scores and the best case shown in Table 3 is 15% and 30% for robots and balls respectively. This indicates a significant reality gap which is discussed in Section 5.2.

Table 6: Instance segmentation AP on real data set (higher is better).

	robots	ball
Pure Simulation	0	0
CycleGan	47.479	49.497
CycleGan-adjusted	49.232	42.327
MUNIT	55.926	50.835
MUNIT-adjusted	**56.705**	**50.910**

For the segmentation APs in Table 6 much of the same holds true as the bounding box APs. The reality gap is slightly worse with 16% for robots and almost 31%

4.4 Panoptic Segmentation

The panoptic segmentation is the most complete way to frame the problem of segmentation where instance segmentation and semantic segmentation are combined. For this evaluation only the instance segmentations are used to identify the reality gap. This allows for easy comparison to the instance segmenter of the previous section. The full results are listed in Appendix B. The results of the panoptic segmenter are similar to the results obtained from the instance segmenter with a few notable exceptions. Dissimilarities can be explained by the use of a different train evaluation split as described in Section 3.4. There is only one Real dataset.

Table 7: Bounding Box AP on translated data set (higher is better).

	robot	goalframe	goalnet	ball
Pure Simulation	**82.422**	**87.098**	**88.393**	**75.917**
CycleGan	76.421	82.340	81.521	68.883
CycleGan-adjusted	78.404	83.696	83.983	69.983
MUNIT	67.294	75.049	74.127	48.619
MUNIT-adjusted	70.961	76.294	74.589	58.648

For the bounding boxes in Table 7 the best scores are slightly lower for the categories robot, ball and goalframe compared to the instance segmenter in Table 3. Additionally the panoptic approach displays roughly the same AP scores for other architectures.

Table 8: Instance segmentation AP on translated data set (higher is better).

	robot	goalframe	goalnet	ball
Pure Simulation	**73.368**	**41.292**	**82.269**	**81.845**
CycleGan	63.752	34.464	73.279	68.470
CycleGan-adjusted	65.656	36.492	76.429	71.295
MUNIT	50.551	24.168	61.659	47.626
MUNIT-adjusted	54.945	26.941	64.667	59.525

The segmentation APs in Table 8 are equally similar to the segmentation APs from the instance segmenter in Table 4. The scores are slightly lower which can be explained by the use of a different evaluation set.

Table 9: Bounding Box AP on real data set (higher is better).

	robots	ball
Pure Simulation	0	0
CycleGan	63.373	45.063
CycleGan-adjusted	62.010	38.440
MUNIT	**66.052**	**45.943**
MUNIT-adjusted	65.623	43.253

The bounding box APs on the real dataset of the panoptic in Table 9 are slightly worse compared to the APs of the instance segmenter 5. The results for the panoptic segmenter trained on the pure simulator are equally bad caused by the same problem of classifying everything as background. The reality gap is 15% and 30% for robots and balls respectively.

Table 10: Instance segmentation AP on real data set (higher is better).

	robots	ball
Pure Simulation	0	0
CycleGan	51.681	**56.037**
CycleGan-adjusted	49.885	50.039
MUNIT	54.401	51.895
MUNIT-adjusted	**55.522**	52.602

The panoptic evaluation produces similar results as the instance segmentation in Table 10 with a notable exception. The CycleGan performs significantly better on both robot and ball segmentations compared to the MASK-RCNN CycleGan results in Table 6. This suggests the panoptic approach has slightly different requirements in order to be perform well. The reality gap is 19% and 25% for robots and balls respectively.

4.5 Segmentation Samples

Some pictures are taken out of the real evaluation set in order to highlight some differences in the panoptic segmenters trained on different translation approaches. In these pictures the semantic ('stuff') classes are annotated with a singular color while instance ('thing') classes have a different color per instance, both semantic and instance types are described in Section 3.4.

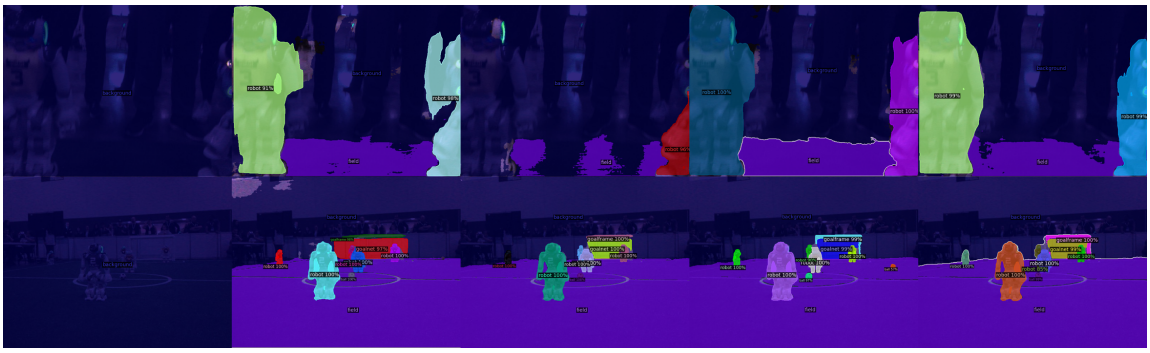


Figure 10: Results of panoptic segmentation on real images trained on pure simulation, CycleGan, CycleGan adjusted, MUNIT and MUNIT adjusted respectively.

In Figure 10 the effects of different GAN architectures are visible. For the first row all except CycleGan-Adjusted segment the two robots correctly. The CycleGan appears to generalize poorly with respect to robot segmentation in some cases, explaining its relatively worse APs on the real dataset compared to all other methods. The second row showcases segmentation of a piece of field as a ball, this occasional imagining of objects is the result of the style augmentation happening in with standard MUNIT architecture. The MUNIT-adjusted does not have the same problem. Finally notice that even though lines are now correctly treated as stuff the class is not correctly predicted. The lines in the picture are annotated "negatively"; meaning there is no annotation at all where the lines annotation should be.

5 Discussion

5.1 Adjusted loss effect

The idea behind the adjusted loss is to ground the generated images more in the simulated domain and by proxy force the generator to focus more on semantic differences. In theory this would improve the performance of any object detector as wrong segmentations are minimized. Wrong segmentations meaning segmentations which are correct in simulation, but having their features removed or obfuscated after domain translation through a GAN. This would mean a segmenter learns to detect the wrong features. In practice the segmenter used for evaluation is not significantly confused by those wrong segmentations.

Instead, the added diversity of the multi-modal approach of MUNIT has a much greater positive effect on the results. It could be the case that the adjusted loss reduces diversity for the ball appearances.The CycleGan-adjusted having the worst overall AP scores, could be the result of a compounding lack of diversity in both the architecture and the adjusted loss function.

5.2 Reality Gap Metrics

The reality gap measured by the FID conflicts slightly with the gap measured by the instance segmenter. The FIDs show that the MUNIT-adjusted architecture produces the best dataset in order to close the reality gap. The instance segmenter AP scores, seem unaffected by the adjusted loss function, even though the FIDs predict an even better score. This difference in metrics can be explained by numerous factors, firstly the underlying neural net used for the FIDs could simply not process certain features important to ball and robot detection. Inversely it could also be the case that certain features in the background are important to the FID metric but not for detection. This would mean that the difference between the ILSVRC-2012 dataset [19] and the dataset used in this thesis skews this metric. Secondly the segmenter may be unable to make use of the better features as the result of a ill-posed problem definition. Using an instance segmenter for both things and stuff is a naive formulation. This may affect the average precision scores for the balls and goals which are manually labelled. The panoptic segmenter however, produces similar or slightly worse results as the instance segmenter. This reaffirms that the problem definition is not ill-posed for the evaluation used in this thesis.

The Panoptic segmenter raises another issue, namely that the ball segmentations are the best when trained on CycleGan. This would mean that the panoptic problem has different

requirements than the instance segmentation problem. A likely explanation would be that the segmentation of the field influences the detection of the ball such that the ball area is overestimated less in the panoptic case.

5.3 Real Evaluation

Because of . This leaves only qualitative evaluation to estimate the gap for the other classes. As can be seen in Section 4.5, the gap appears to be similar for all classes except lines. The lines perform badly even when designated as semantic segmentations ('stuff').

Compared an alternative method of generating synthetic data, such as the stochastic scene-render [4] the APs reported in this thesis are significantly lower. The AP reported in that paper is 94.40% for the instance bounding box task. These results are achieved with a specifically build architecture, which was both smaller and trained specifically for the task on numerous configurations. In that same paper additional model size show a reduction in performance when trying to detect instances on the field. This trend could also explain why the large size of the MASK-RCNN model affects the performance negatively for the task posed in this thesis.

6 Conclusion

This thesis aimed to identify the challenges and possibilities of image-to-image translation between simulation images and real images for the SPL. As stated in the introduction no other literature has addressed the reality gap caused by image translation GANs within context. The gap was identified as 28 FID for the best performing architecture: MUNIT-adjusted. However, additional evaluation with instance segmenters produced results which indicated that this estimate was optimistic. With the evaluation of the instance segmenter the reality gap was estimated at 15.5% AP loss for robots and 30.5% AP loss for balls with MUNIT-adjusted still being the best architecture. After evaluating the panoptic segmenter this gap calculated to be a very similar loss of 17.5% and 27.5%. In the panoptic evaluation, the MUNIT-adjusted was not the best architecture. However, MUNIT based architectures still produced the best results on average. Therefore it can be concluded that the MUNIT architecture provides better datasets than the CycleGan architecture. This is mainly because of its style augmentation, which may cause erroneous annotations, but provides the diversity needed for better results.

7 Future Work

Although many questions about enhancing simulated images were covered, some questions will be left unanswered. First of all, only one simulation was used to generate synthetic data. The generalizability of the architectures used in this thesis to other simulations, although likely, still needs to be demonstrated. Furthermore, the most general case of multiple simulations to multiple kinds of camera settings of robots also needs to be explored. Finally the results of in this thesis highlight the potential of the unsupervised translation with state of the art detectors, but this promise also needs to be tested on detection algorithms used on real robots.

8 Acknowledgements

The SPL team named HULKs provided both the real image dataset and the simulator used for this thesis. They were a great help in making this thesis possible.

References

[1] Leon Žlajpah. Simulation in robotics. *Mathematics and Computers in Simulation*, 79(4):879–897, 2008.

[2] Bernd Poppinga and Tim Laue. JET-Net: Real-time object detection for mobile robots. In Stephan Chalup, Tim Niemueller, Jackrit Suthakorn, and Mary-Anne Williams, editors, *RoboCup 2019: Robot World Cup XXIII*, volume 11531 of *Lecture Notes in Artificial Intelligence*, pages 227–240. Springer, 2019.

[3] Bernd Poppinga and Tim Laue. Jet-net: real-time object detection for mobile robots. In *Robot World Cup*, pages 227–240. Springer, 2019.

[4] Timm Hess, Martin Mundt, Tobias Weis, and Visvanathan Ramesh. Large-scale stochastic scene generation and semantic annotation for deep convolutional neural network training in the robocup spl. In *Robot World Cup*, pages 33–44. Springer, 2017.

[5] Shizuo Kaji and Satoshi Kida. Overview of image-to-image translation by use of deep neural networks: denoising, super-resolution, modality conversion, and reconstruction in medical imaging. *Radiological physics and technology*, 12(3):235–248, 2019.

[6] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[7] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018.

[8] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] Thomas Röfer, Tim Laue, Judith Müller, Michel Bartsch, Arne Böckmann, Florian Maaß;, Thomas Münder, Marcel Steinbeck, Simon Taddiken, Alexis Tsogias, and Felix Wenk. B-human team description for robocup 2013. In Sven Behnke, Manuela Veloso, Arnoud Visser, and Rong Xiong, editors, *RoboCup 2013: Robot Soccer World Cup XVII Preproceedings*. RoboCup Federation; http://www.robocup.org/, 2013.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[11] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566, 2016.

[12] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49, 2012.

[13] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[14] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[15] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 700–708. Curran Associates Inc., 2017.

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

[17] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[21] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[25] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[26] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

[27] Lekanne gezegd Deprez Hidde G.J. and A. Visser. The Dutch Nao Team synthetic dataset, 6 2020.

A Mask-RCNN large nets

On the particular dataset used in this paper, changing the size of the MASK-RCNN net performs the same or worse. The APs for the best performing MASK-RCNN with ResNet size 101 according to the paper[21] trained with the cycleGan with adjusted loss performs as follows:

Segmentation AP							
Data	lines	robots	goalframe	goalnet	field	ball	background
Converted	5.098	67.913	45.429	78.595	92.800	71.234	91.061
Real	-	47.578	-	-	-	42.633	-
Bounding Boxes AP							
Converted	68.672	77.94	82.741	83.030	96.200	70.570	92.210
Real	-	59.667	-	-	-	40.203	-

B Panoptic Evaluation

The panoptic method has a specific method of evaluation. This method of evaluating is difficult to compare with evaluation methods used for the instance segmenter. Additionally this method cannot be used to evaluate the real dataset as it requires full semantic segmentations. Therefore these results are not discussed further.

Pure Simulation Panoptic Evaluation Results:

	PQ	SQ	RQ	#categories
All	82.266	91.577	89.973	7
Things	81.135	86.423	93.795	4
Stuff	83.773	98.449	84.877	3

CycleGan Panoptic Evaluation Results:

	PQ	SQ	RQ	#categories
All	77.978	88.707	87.683	7
Things	75.024	83.365	89.877	4
Stuff	81.917	95.829	84.758	3

CycleGan Adjusted Panoptic Evaluation Results:

	PQ	SQ	RQ	#categories
All	79.557	89.615	88.689	7
Things	77.205	84.198	91.548	4
Stuff	82.692	96.839	84.878	3

Munit Panoptic Evaluation Results:

	PQ	SQ	RQ	#categories
All	70.379	84.607	82.287	7
Things	64.288	79.161	81.001	4
Stuff	78.500	91.868	84.002	3

Munit adjusted Panoptic Evaluation Results:

	PQ	SQ	RQ	#categories
All	73.621	86.276	84.765	7

| Things | 68.979 | 80.769 | 85.179 | 4 |
| Stuff | 79.811 | 93.619 | 84.214 | 3 |