Undergraduate Project Report
2022/23

Synthetic-to-realistic Object Rendering for 6D Object Pose Estimation

Name:	Zhiyang Chen
School:	International School
Class:	2019215104
QMUL Student No.:	190897332
BUPT Student No.:	2019213135
Programme:	Telecommunications Engineering with Management

Date: 25-04-2023

Table of Contents

Abstract

The fundamental objective of this project is to identify a framework that can automatically generate big data for neural network training and testing. Our aim is to use the Generated Adversarial Network (GAN) to translate objects' image style from synthetic to realistic to obtain realistic datasets for deep learning-based computer vision frameworks. To achieve this goal, we consider the Normalized Object Coordinate Space (NOCS) framework for Category-Level 6D Object Pose and Size Estimation and attempt to enhance its dataset generation method. However, the challenge of unavailability of large-scale datasets for 6D object pose and size estimation framework training and testing led the writer to introduce a spatially context-aware mixed reality data generation method. This method automatically generates large quantities of data comprising synthetic objects with a realistic appearance from the ShapeNet-Core dataset, combined with tabletop real scenes. The resulting mixed reality dataset is called CAMERA dataset. However, synthetic objects in the dataset still create a de-facto domain gap in computer vision. To reduce this gap, we trained the CycleGAN with a cycle-consistent loss function to render synthetic objects to realistic. Subsequently, we used our trained GAN model to render synthetic objects in the CAMERA dataset to realistic. These steps resulted in a more realistic CAMERA++ dataset with its original information, including mask, ground truth, and depth map. We then used this dataset in conjunction with the original CAMERA dataset to train the NOCS model simultaneously and make comparisons. After evaluating and experimenting, our improved CAMERA++ dataset enables the NOCS model to better generalize realistic data for training. In the future, we believe that our proposed data rendering method can be applied to various other Computer Vision applications.

Keywords

List of keywords.

Generated Adversarial Network, Image-to-Image translation, Data Generation, Category Level Object 6D Pose and size estimation.

摘要

This is the Chinese translation of the Abstract.

这个项目最基本的想法是找到一个很好的框架，可以自动生成神经网络训练和测试的大数据。我们的目标是使用生成对抗网络(GAN)将物体图像风格从合成转换为真实，从而为一些基于深度学习的计算机视觉框架获得真实的数据集。为此，我们考虑了一个著名的分类级 6D 对象姿态和大小估计框架——标准化对象坐标空间(NOCS)，并尝试改进其数据集生成方法。针对 6D 对象姿态和尺寸估计框架训练和测试缺乏大规模数据集的挑战，作者引入了一种空间上下文感知的混合现实数据生成方法，从 ShapeNet-Core 数据集中自动生成由具有逼真外观的合成对象组成并与桌面真实场景组合的大量数据。这个混合现实数据集被称为 CAMERA 数据集。但由于合成物体的存在，这种方法仍然会导致计算机视觉的事实上的域差距。为了减少这种事实上的域差距，我们首先用周期一致损失函数训练 CycleGAN，以将来自 ShapeNet-Core 数据集的合成对象呈现为现实。然后使用我们训练好的 GAN 模型来渲染 CAMERA 数据集中的合成对象。从这些步骤中，我们可以得到一个更真实的 CAMERA++数据集，它的原始信息(掩码，ground truth, depth map)。然后我们可以使用 CAMERA++数据集与原始 CAMERA 数据集同时训练 NOCS 模型并进行比较。经过评估和实验，我们可以看到，我们对 CAMERA++数据集的改进使 NOCS 模型能够更好地概括出现实数据用于训练。在未来的工作中，我认为我们提出的数据渲染方法可以应用到更多的计算机视觉应用中。

关键词

This is the Chinese translation of the Keywords.

生成对抗网络， 图片风格转译， 数据生成， 类别级对象 6D 姿态和大小估计

Chapter 1: Introduction

With the recent advances in deep learning, computer vision has made significant progress in various applications such as object detection, segmentation, and pose estimation. However, the accuracy of these algorithms heavily relies on the availability of large-scale labeled datasets for training and testing. Acquiring such datasets is a time-consuming and expensive process. The lack of realistic and diverse datasets is a significant bottleneck in the development of deep learning-based computer vision frameworks.

In this project, we aim to address this issue by improving the dataset generation method for the Normalized Object Coordinate Space (NOCS) Category-Level 6D Object Pose and Size Estimation framework. Our goal is to improve the CAMERA datasets provided by NOCS framework using Image-to-Image translation by Generative Adversarial Network.

To achieve this, we firstly use real objects dataset from Objectron dataset and synthetic objects datasets from ShapeNet-Core dataset to train CycleGAN model. This Generated Adversarial Network (GAN) then can translate object image style from synthetic to realistic. Then we use our trained GAN model to render synthetic objects in CAMERA dataset to realistic. This approach aims to reduce the de-facto domain gap in computer vision caused by the synthetic objects in the generated datasets.

The main contributions of this project are:

- Fulfill the synthetic-to-realistic objects rendering using CycleGAN based on the Image-to-Image translation application.
- Improve the dataset generation method for NOCS framework using synthetic-to-realistic objects rendering from GAN.
- Evaluating the performance of the improved CAMERA++ dataset compared to the original CAMERA dataset by train and test the NOCS model.
- Analyzing the effectiveness of the proposed method in reducing the de-facto domain gap in computer vision.

The report is structured as follows: In the background section, we provide an overview of the NOCS framework, the CycleGAN framework, and the CAMERA dataset generation process. In the Design and Implementation section, we describe how we train our CycleGAN model to

achieve synthetic-to-realistic object rendering and how we use CycleGAN to improve the CAMERA dataset in NOCS. The experimental comparison results between our dataset and the original dataset are presented and analyzed in the results and discussion section, followed by a discussion of the findings and limitations. Finally, we conclude the report with the main contributions, limitations, and future directions of the project in the conclusion and further work section.

Through our experiments and evaluations, we found that our proposed dataset improving method can improves the accuracy of 6D object pose estimation compared to the original dataset. We also observed that synthetic-to-realistic object rendering provided by GAN can effectively reduce the de-facto domain gap in computer vision. These findings have significant implications for improving the accuracy of deep learning-based computer vision frameworks in real-world scenarios.

Chapter 2: Background

2.1 Normalized Object Coordinate Space (NOCS) for Category-Level 6D Object Pose and Size Estimation

Category-Level 6D Object Pose and Size Estimation: Involves estimating three rotation, three translation, and three scaling parameters (dimensions) for an object instance. The solution to this problem can be visualized as a tightly oriented closed box around the object. The objects belong to known object classes, such as cameras, and training samples have been observed during training. However, this task is particularly challenging when testing with objects that cannot be seen and CAD models cannot be used. To address this issue, the NOCS framework proposes a new representation for defining a shared object space, which enables defining 6D poses and sizes of invisible objects.

Normalized Object Coordinate Space (NOCS) is a framework for category-level object 6D pose and size estimation [6]. The proposed method employs a representation of objects in a normalized coordinate space, where every object is scaled and aligned to a standard template. This representation enables the authors to accurately estimate the 6D pose and size of an object, including those that are occluded or surrounded by clutter in the scene.

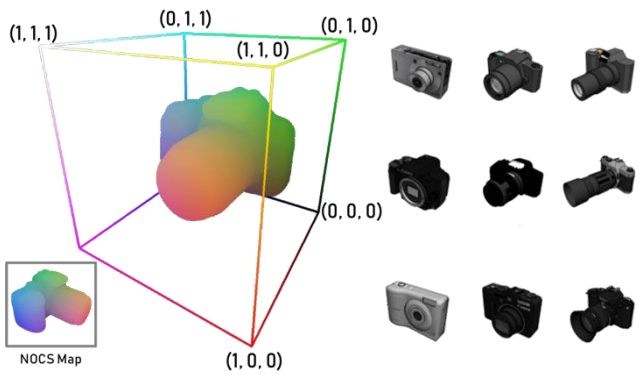


Figure 1 The Normalized Object Coordinate Space (NOCS) is a 3D space contained within a unit cube. We use this to set objects' orientation and generate NOCS map for NOCS training pipeline.

The NOCS method is capable of estimating the pose and size of objects within a particular category (e.g., cameras or bottles) without any object-specific training data, making it a highly versatile approach that can be applied to various object recognition and manipulation tasks.

The NOCS framework represents a significant advancement in the fields of computer vision and robotics, and it has the potential to enable numerous applications in areas such as autonomous driving, robotics, and augmented reality.

2.2 Mixed Reality Dataset and Limitation

Context-Aware Mixed Reality Approach (CAMERA). NOCS proposes a context-aware mixed reality approach [6] to generate large amounts of training data with ground truth for hand-scale objects. This new approach solves the limitations of previous methods, saves time, and is cost-effective. The composite object is rendered and combined into a real scene with reasonable physical location, lighting, and scale.

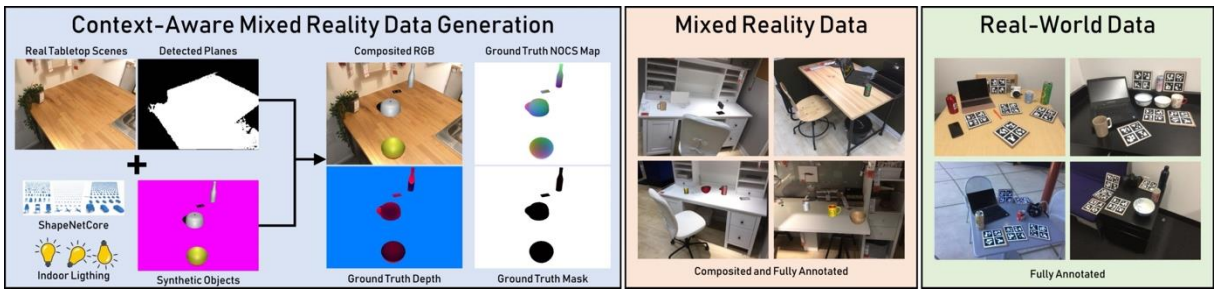


Figure 2 Context-Aware MixEd ReAlity (CAMERA) approach. The approach generate data by combining real images of tabletop scenes, using algorithms to detect plane surfaces, and render synthetic objects onto the plane surfaces (left). Since the objects are synthetic, we can obtain accurate ground truth for class label, instance mask, NOCS map, and 6D pose and size. The CAMERA dataset generation approach can generate large quantities of realistic scenes images (middle). The approach also provides a real-world dataset for training and testing (right).

Developing deep learning-based computer vision frameworks, like NOCS, face the significant challenge of limited large-scale datasets for both training and testing. The generation of realistic and diverse datasets is a challenging and time-consuming task. This task can suffer from the domain gap problem, where the performance of the model is limited due to the differences between the synthetic and real-world data distributions.

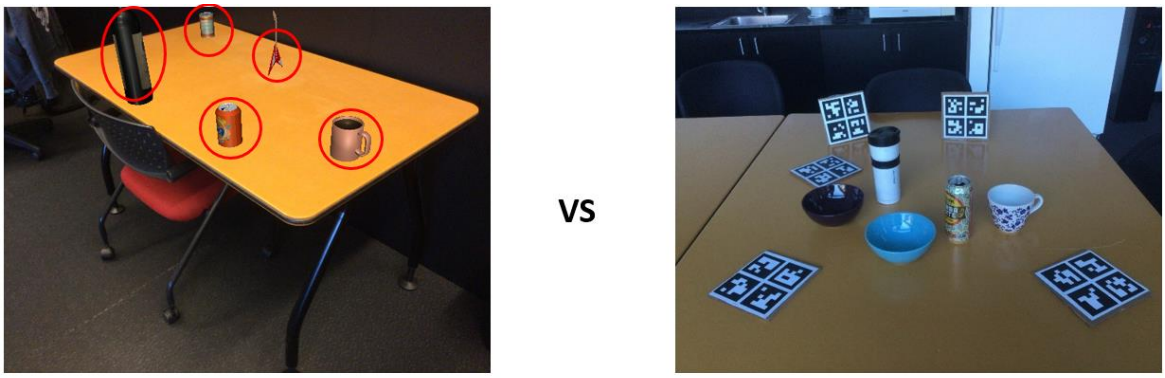


Figure 3 The left scene image is mixed reality image from CAMERA dataset, compared to the real scene image, it will occur a synthetic and real domain gap because of the synthetic objects exist.

To address the issue of the lack of large-scale datasets for deep learning-based computer vision frameworks like NOCS, researchers have proposed various dataset generation methods,

including rendering 3D models using game engines, 3D scanning real-world objects, and using Generative Adversarial Networks (GANs) to generate synthetic images that resemble real-world data. In general, they consist of two neural networks: a generator that generates synthetic data and a discriminator that distinguishes between real and synthetic data. But as the technology iteration, variety of GAN model have been proposed.

2.3 CycleGAN based Image-to-Image Translation

Generated Adversarial Network (GAN), Goodfellow et al. proposed the Generative Adversarial Network (GAN) in 2014 [1]. This forms the fundamental basis for all GANs. GAN is a framework for estimating generative models using an adversarial process. It involves simultaneous training of two models - a generator model G that captures the data distribution and a discriminator model D that estimates the probability of a sample coming from the training data rather than G. The training procedure for G involves maximizing the probability of D making an error or being fooled by G. In the space of arbitrary functions G and D, there exists a unique solution where G recovers the training data distribution and D equals $\frac{1}{2}$ everywhere. This framework has revolutionized the generation of high-quality fake data using neural networks.

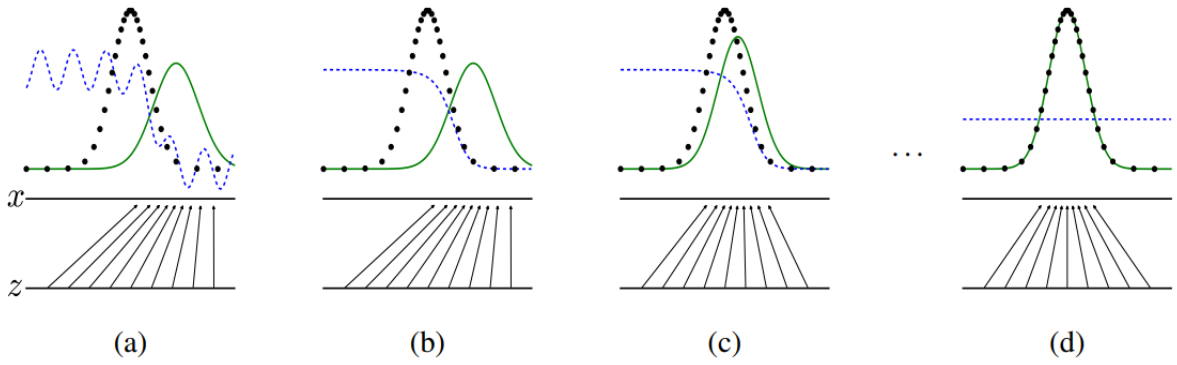


Figure 4 Generative adversarial nets are trained by simultaneously updating the discriminative distribution (*D*, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_x from those of the generative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which *z* is sampled, in this case uniformly. The horizontal line above is part of the domain of x. The upward arrows show how the mapping $x = G(z)$ imposes the non-uniform distribution pg on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to pdata and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$. (c) After an update to G, gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = \frac{1}{2}$.

Since the original model proposed by Goodfellow et al in 2014, GANs have undergone significant improvements and have found numerous applications in various fields. Researchers have re-imagined and enhanced every aspect of the model, including architecture, loss functions, datasets, and evaluation metrics. In 2022, a survey paper was published that provided an overview of GAN architecture and its various applications and algorithms. [9].

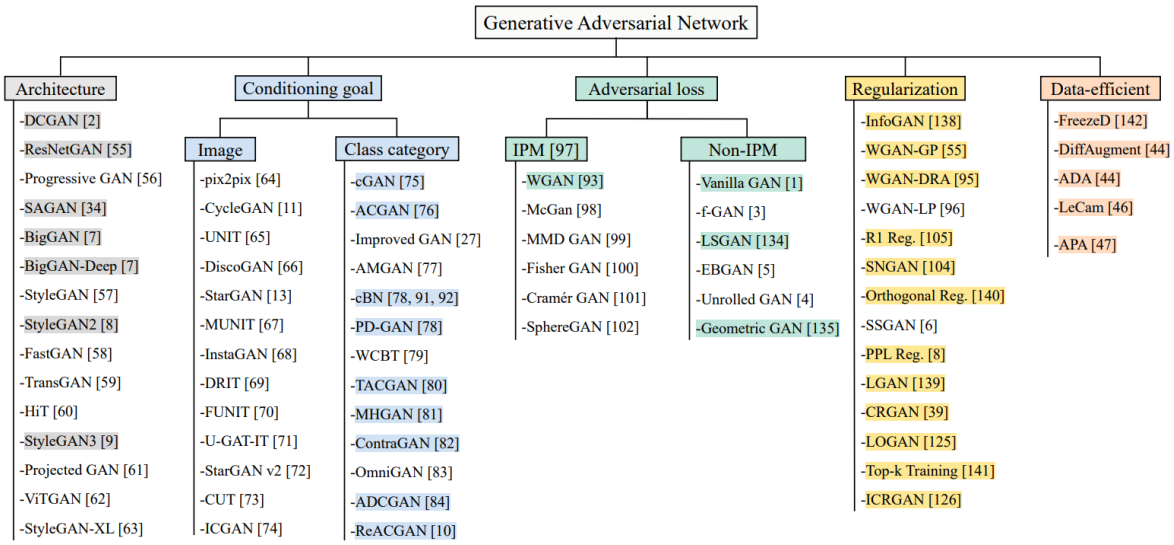


Figure 5 GANs architecture

Image-to-Image Translation. The concept of image-to-image translation has been explored in the past, with Hertzmann et al.'s Image Analogies using a non-parametric texture model based on a single input-output training image pair. Other methods rely on a dataset of input-output examples to learn a parametric translation function using CNNs. The CycleGAN approach extends the "pix2pix" framework proposed by Isola et al [8], which uses a conditional generative adversarial network to map input to output images. This approach has been applied to various tasks, and unlike the previous methods, CycleGAN can learn mappings without paired training examples.

CycleGAN. The CycleGAN architecture has become one of the most widely used GAN models for image-to-image translation between two domains that do not have paired training data [2]. By design, CycleGAN can perform unsupervised image-to-image translation, allowing it to learn how to translate images between domains without paired training data. In this project, we propose using the image-to-image translation function of CycleGAN to achieve our goal of generating realistic renderings from synthetic images. To do this, we use two sets of unpaired training data: real object images and synthetic object images. We train

CycleGAN on each dataset to learn the style of each domain. Once trained, we can use one side of the CycleGAN to translate the synthetic style to the realistic style, effectively achieving synthetic-to-realistic rendering.

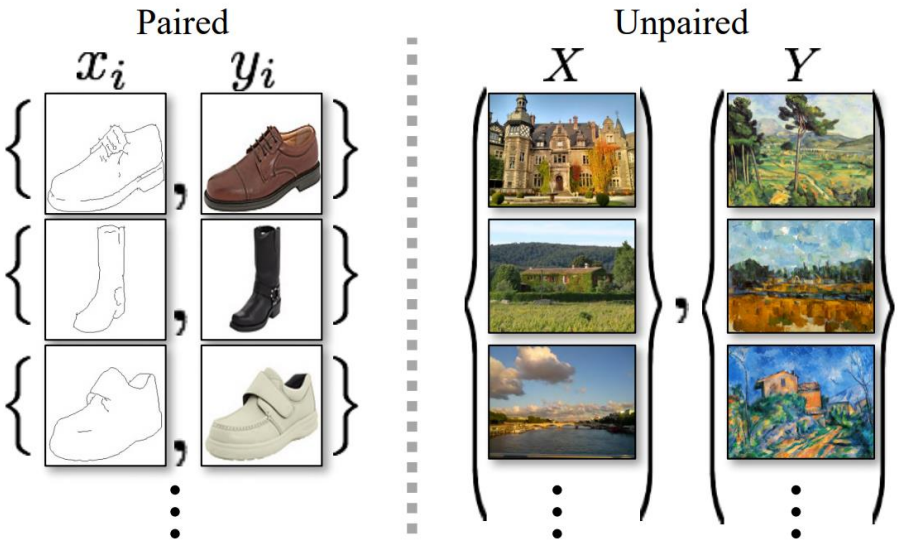


Figure 6 Paired training data (left) consists of training examples $\{x_i, y_i\}_{(i=1)}^{N}$, where the correspondence between x_i and y_i exists. CycleGAN instead by considering unpaired training data (right), consisting of a source set $\{x_i\}_{(i=1)}^{N}$ $(x_i \in X)$ and a target set $\{y_j\}_{(j=1)}^{M}$, $(y_j \in Y)$, with no information provided as to which x_i matches which y_i .

CycleGAN's use of cycle-consistency loss is a key innovation that enforces consistency between the original and translated images when they are returned to their original domain. This loss term ensures that the translated image is meaningful and preserves important characteristics of the original image.

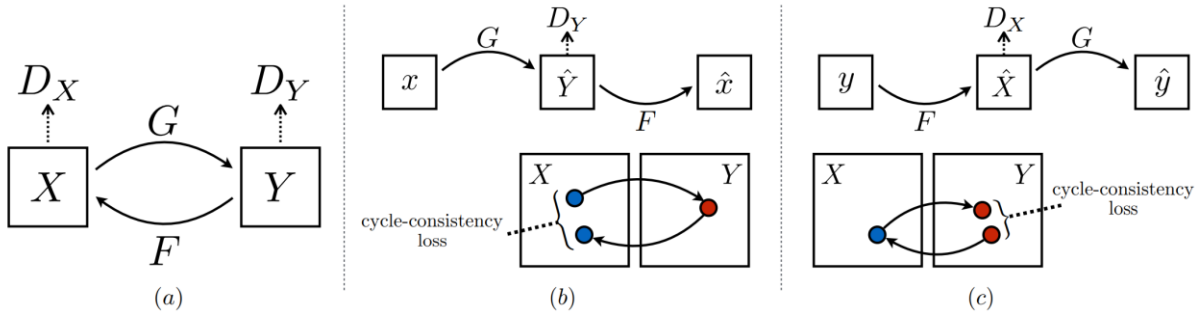


Figure 7 (a) The CycleGAN model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two cycle consistency losses that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

The CycleGAN framework comprises of two generators and two discriminators, which

respectively translate images between two domains and distinguish between real and fake images. The training of these models is done in an adversarial manner to minimize the loss function, which combines adversarial, cycle-consistency, and identity losses. The cycle-consistency loss ensures that the translation is consistent with the original image when translated back to its original domain, while the identity loss encourages the generators to preserve the original image when translating it to the same domain.

CycleGAN has had a significant impact on the field of image-to-image translation and has been used for various applications such as style transfer, image colorization, and domain adaptation. Moreover, its success has inspired the development of several variants and extensions, including StarGAN, DualGAN, and DiscoGAN, which have further advanced the state-of-the-art in unsupervised image-to-image translation.

2.4 Summary

In this project, we propose a dataset improve method for NOCS framework. It can be described as using synthetic-to-realistic object rendering to render synthetic objects in CAMERA dataset to realistic by GAN. The GAN we used is CycleGAN, which is trained to translate synthetic object images to realistic images based on its Image-to-Image translation application. Compared to the original dataset, our improved dataset can be more realistic and reduce the domain gap that caused by the synthetic objects. The proposed method can improve the performance of the NOCS framework for object 6D pose and size estimation, furthermore, it aims to generate large amounts of realistic data that resemble real-world data distributions and reduce the domain gap problem.

Chapter 3: Design and Implementation

In this section, we describe the design and implementation of our proposed framework for Synthetic-to-realistic Object Rendering for 6D Object Pose and Size Estimation. Our framework consists of five main steps: (1) Object Extraction, (2) Synthetic Object Rendering, (3) CycleGAN Training and Loss Functions, (4) CycleGAN-based Object Rendering, and (5) 6D Object Pose Estimation using NOCS model. We describe each of these steps in detail below.

3.1 Real Object Extraction

To obtain a real-world dataset of objects, we used the Objectron open-source dataset [4]. Objectron is a collection of short, object-centric videos that capture daily life objects from various angles. We used Mask R-CNN [5], a state-of-the-art object detection model, to detect the real objects' masks and extract the real objects one by one from the videos. We then resized the images to a resolution of 512x512 and saved them as .png files.

3.2 Synthetic Object Rendering

We also needed a synthetic dataset of objects to augment the real-world dataset. To obtain this dataset, we used the ShapeNet-Core dataset [3], which is a large collection of 3D models of objects in various categories. We rendered the 3D models using Blender, a popular 3D rendering software. We chose 3 object categories (bottle, camera, laptop) from the ShapeNet-Core dataset and obtained 3000 images for each category by rendering the corresponding 3D models from ShapeNet-Core. We saved the synthetic images as .png files with a resolution of 512x512.

3.3 CycleGAN Training, and Loss Function

To reduce the domain gap between the synthetic and real-world datasets, we used CycleGAN, a state-of-the-art image-to-image translation model. We trained a CycleGAN model on the

combined real and synthetic datasets to translate synthetic object images into realistic object images. We used the dataset we made, 4K real-world object images and 3K synthetic object images as our training and testing set. The model was trained for 200 epochs with a learning rate of 0.0002 and a batch size of 1.

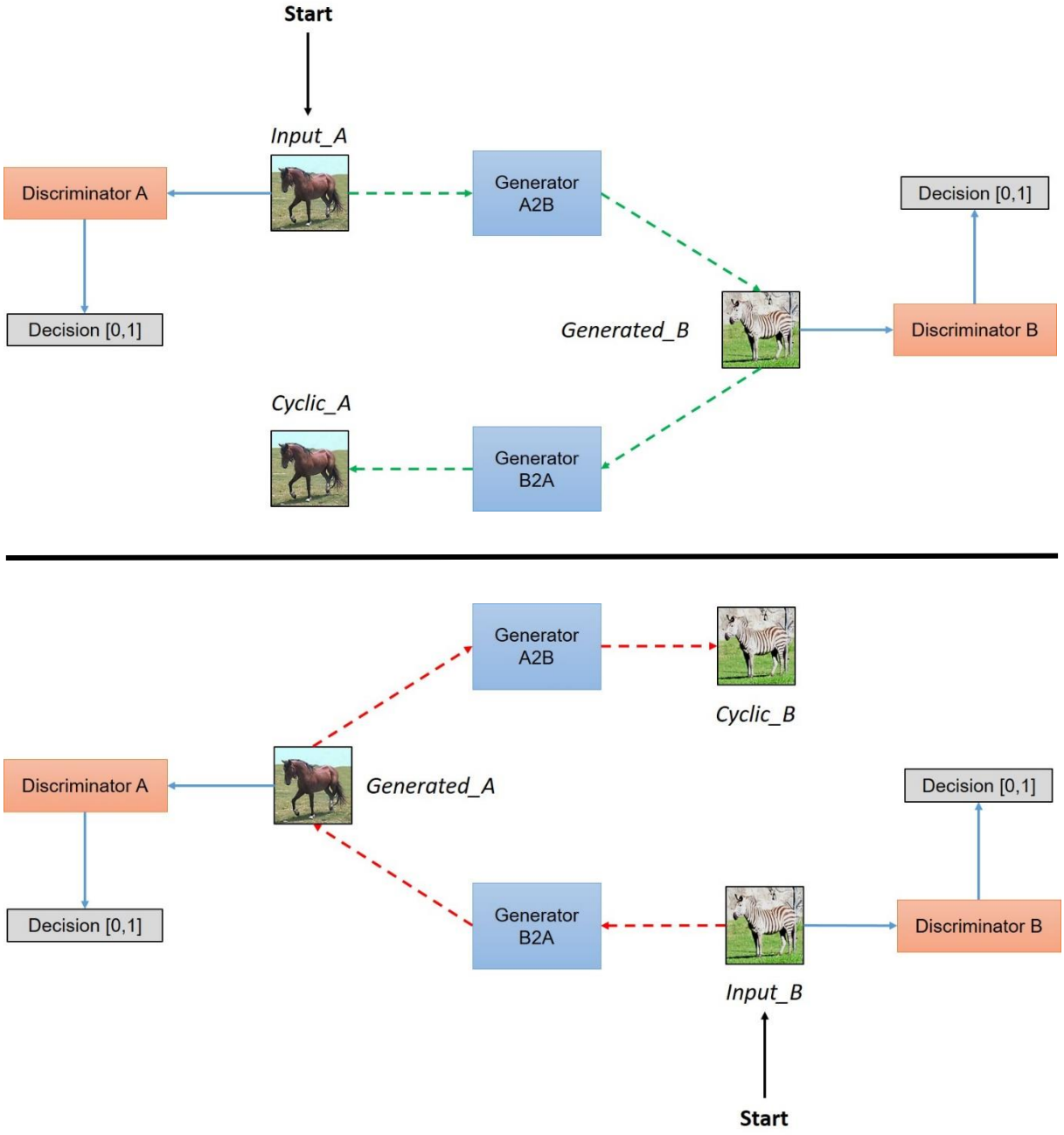


Figure 8 CycleGAN Network Architecture. The CycleGAN simultaneously train two generators and two discriminators. By this, it can achieve image-to-image translation and reconstruct the input image by using the other generator.

As we said in background section, the CycleGAN model needs unpaired dataset for training and testing. Paired datasets involve manually mapping an image in one domain to an image in

another domain, such that they share various features. This mapping defines a meaningful transformation of an image from one domain to another. In contrast, unpaired datasets lack this predefined transformation, so a generator must be trained to map an input image from one domain to an image in another domain. To ensure a meaningful mapping, the generator must share features that can be used to map the output image back to the input image, which is achieved by using two generators in a cycle. The first generator converts an image from the input domain to the target domain, while the second generator converts the generated image back to the input domain. The generator and discriminator play a game where the generator tries to generate images close to the original images in the target domain, and the discriminator tries to distinguish between real and generated images. The Nash equilibrium is achieved when the generator's distribution becomes the same as the desired distribution.

The loss function of the CycleGAN used was a combination of generator adversarial loss, discriminator loss and cycle-consistent loss. Since we need to train two generators and two discriminators together, we need to design and decide the loss functions in a way that can achieve these goals:

1. Each discriminator should approve all the original real images in the corresponding categories. (Basic idea about GAN)
2. Each discriminator should know and can reject all the fake images which are generated by Generators to fool them. (Basic idea about GAN)
3. Generators should make the discriminators approve all the fake generated images, in order to fool the discriminators. (Basic idea about GAN)
4. The generated image should save the property of the original image, so if we want to generate fake images using one generator for example $Generator_{A\to B}$, then we should be able to generate back to the original images using the other generator $Generator_{B\to A}$. we must make them satisfy the proposed cyclic-consistency. (Innovation idea about CycleGAN)

```
# define loss functions
self.criterionGAN = networks.GANLoss(opt.gan_mode).to(self.device)  #
define GAN loss.
self.criterionCycle = torch.nn.L1Loss()
self.criterionIdt = torch.nn.L1Loss()
```

3.3.1 Discriminator loss

Discriminator must be trained to satisfy the distribution for original real images should be as

close to 1, and vice versa for discriminator B. So Discriminator A can be design to minimize $(Discriminator_A(a) - 1)^2$ and same goes for B as well $(Discriminator_B(b) - 1)^2$. Also, discriminator should be able to distinguish fake generated images and real original images, a good discriminator need predicting 0 for fake images produced by the generator. Discriminators can be design to minimize $(Discriminator_A(Generator_{B\to A}(b)))^2$, and $(Discriminator_B(Generator_{A\to B}(a)))^2$

```
def backward_D_basic(self, netD, real, fake):
    """Calculate GAN loss for the discriminator

    Parameters:
        netD (network)      -- the discriminator D
        real (tensor array) -- real images
        fake (tensor array) -- images generated by a generator

    Return the discriminator loss.
    We also call loss_D.backward() to calculate the gradients.
    """
    # Real
    pred_real = netD(real)
    loss_D_real = self.criterionGAN(pred_real, True)
    # Fake
    pred_fake = netD(fake.detach())
    loss_D_fake = self.criterionGAN(pred_fake, False)
    # Combined loss and calculate gradients
    loss_D = (loss_D_real + loss_D_fake) * 0.5
    loss_D.backward()
    return loss_D

def backward_D_A(self):
    """Calculate GAN loss for discriminator D_A"""
    fake_B = self.fake_B_pool.query(self.fake_B)
    self.loss_D_A = self.backward_D_basic(self.netD_A, self.real_B, fake_B)

def backward_D_B(self):
    """Calculate GAN loss for discriminator D_B"""
    fake_A = self.fake_A_pool.query(self.fake_A)
    self.loss_D_B = self.backward_D_basic(self.netD_B, self.real_A, fake_A)
```

3.3.2 Generator loss

The generator should be able to fool the discriminator about the fake images generated by them. We could achieve this if the distribution detected by the discriminator of the fake-generated images is as close to 1 as possible. So the generators can be design to minimize $(Discriminator_B(Generator_{A\to B}(a)) - 1)^2$, and $(Discriminator_A(Generator_{B\to A}(b)) - 1)^2$

```
# GAN loss D_A(G_A(A))
self.loss_G_A = self.criterionGAN(self.netD_A(self.fake_B), True)
# GAN loss D_B(G_B(B))
self.loss_G_B = self.criterionGAN(self.netD_B(self.fake_A), True)
```

3.3.3 Cycle-consistency loss

And the most important one is the cycle-consistency loss which makes we are able to get the fake generated image back to the original one using another generator. To achieve this, the difference between the cycle reconstruction image and the original one should be as small as possible.

```
# Forward cycle loss || G_B(G_A(A)) - A||
self.loss_cycle_A = self.criterionCycle(self.rec_A, self.real_A) * lambda_A
# Backward cycle loss || G_A(G_B(B)) - B||
self.loss_cycle_B = self.criterionCycle(self.rec_B, self.real_B) * lambda_B
```

3.3.4 Putting Them Together

With the loss function defined, all needed to train the model is to minimize the loss function by changing the model parameters, we then calculate the gradients of the loss function.

```
# combined loss and calculate gradients
self.loss_G = self.loss_G_A + self.loss_G_B + self.loss_cycle_A +
self.loss_cycle_B + self.loss_idt_A + self.loss_idt_B
self.loss_G.backward()
```

3.3.5 CycleGAN Training Pipeline

When training CycleGAN, we use the synthetic dataset and realistic dataset as input, train two generators to generate corresponding output, from synthetic to realistic and from real to synthetic. Then each by using the other generator to reconstruct the original input. We use the discriminator to calculate the discriminator loss and the generator loss, and use the reconstruction image to calculate the cycle-consistency loss. We apply L1 norm for each loss and calculate the gradients together with optimize the network weights in every iteration during the backward progress.

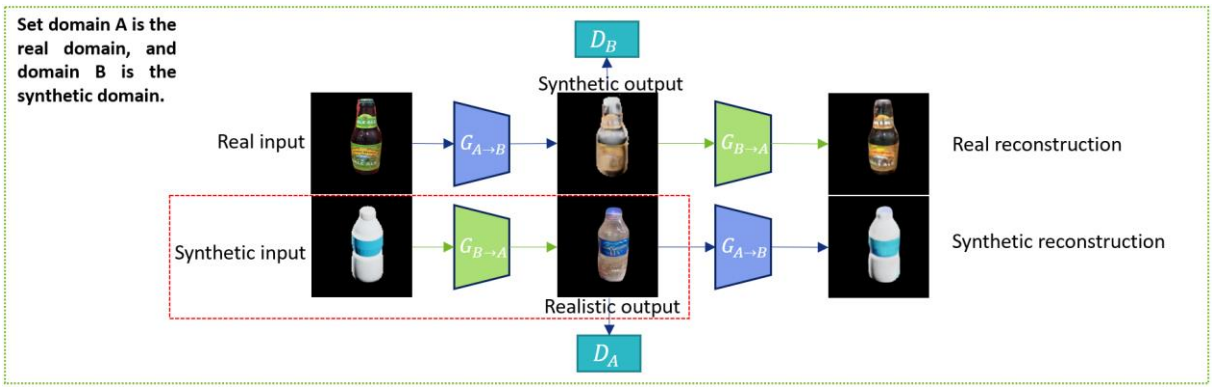


Figure 9 CycleGAN Training Details. We set domain A is real domain and domain B is synthetic domain,

the diagram shows images and networks in CycleGAN training process. What we want to use here is $G_{B\to A}$ which can achieve synthetic to realistic object rendering.

3.3.6 Mask Guided loss

In the testing (rendering) process, we find that our CycleGAN model may not generate the fake realistic image as the same shape as the real synthetic input. To solve this problem and improve our results, we proposed a new mask loss that can be used during the test process. It is possible to include a mask-guided loss function that encourages the network to generate the result only on the region of interest (ROI) defined by the input mask. Since we can easily get the mask information for the synthetic dataset. This idea can be similar to eq8. in [7].

The objective of the mask-guided loss is to ensure that the output of G only changes the pixels inside the masked region of the synthetic bottle, and does not modify the pixels outside this region. This can be achieved by adding a mask-guided L1 loss to the CycleGAN objective, as follows:

$$L_{Mask} = lambda_{mask} * ||M * (G(x) - x)||\_1$$

Where $lambda_{mask}$ is a hyperparameter that controls the weight of the mask-guided loss, x is the synthetic bottle image, M is the mask image of the synthetic input, and ||.||_1 denotes the L1 norm.

```
def mask_loss(self, mask, real_synthetic, fake_realistic):
    """Calculate mask loss to improve the rendered results

    Parameters:
        mask (tensor array) -- mask information
        real_synthetic (tensor array) -- real synthetic images
        fake_realistic (tensor array) -- fake realistic images generated by
a generator
        # mask: tensor of shape (batch_size, 1, H, W)
        # fake: tensor of shape (batch_size, C, H, W)
        # real: tensor of shape (batch_size, C, H, W)
    """
    fake_masked = fake_realistic * mask
    real_masked = real_synthetic * mask
    loss = torch.nn.L1Loss(fake_masked, real_masked)

    return loss
```

By adding the mask-guided loss to the CycleGAN objective during the testing process, the network is encouraged to generate realistic bottle images that preserve the shape and details of the synthetic bottle in the masked region, while ignoring the other regions. This should improve the quality of the generated images and reduce the loss of information due to the imperfect mask extraction process.

3.4 CycleGAN-based Object Rendering

Then by using our trained CycleGAN model, we can transfer the synthetic objects in CAMERA dataset to realistic. Since the CAMERA dataset has provided all the objects' mask information, what we can do here is use mask to extract the synthetic objects and send these synthetic objects into our CycleGAN model. Then we can obtain realistic objects, by using the mask information again, we can put these realistic objects back in their position to replace the original synthetic objects. Thus, we can make the CAMERA dataset more realistic, and what we want is to reduce the domain gap through this pipeline.

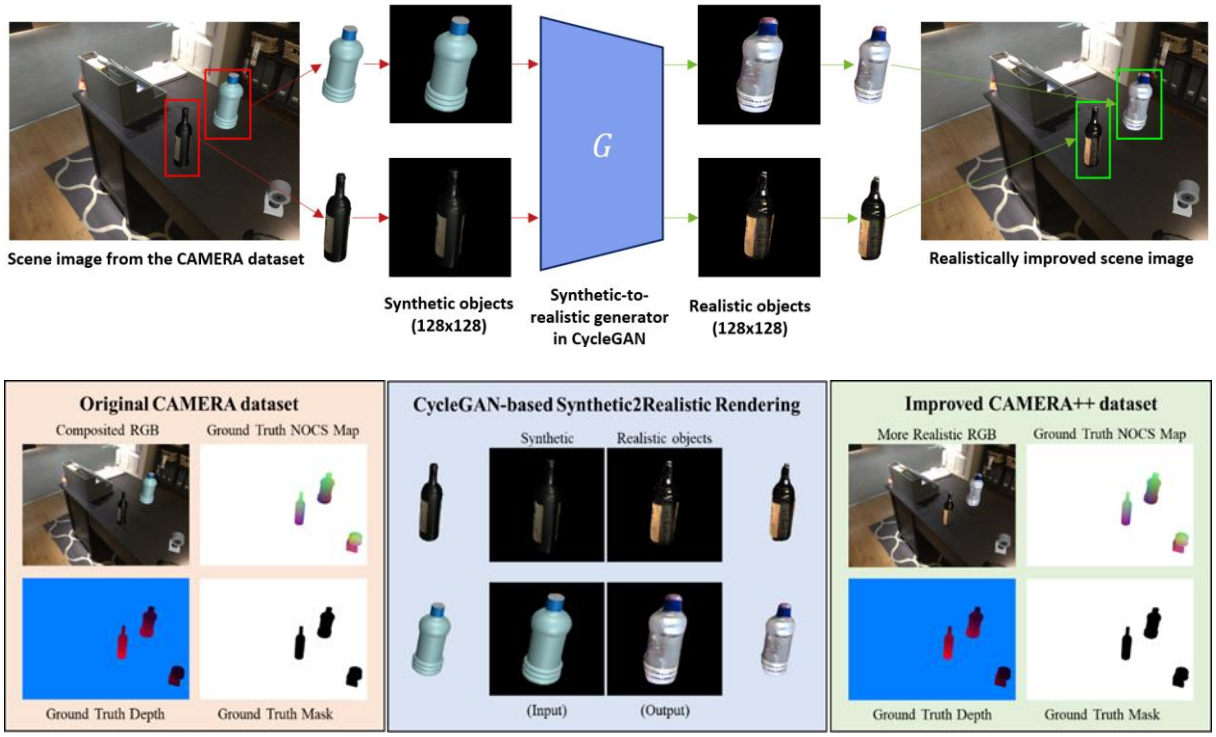


Figure 10 Our dataset improved pipeline using the trained CycleGAN model. Since the NOCS object 6D pose and size estimation framework only need RGB input image, our method is to use the provided mask information and the original composited RGB image from the CAMERA dataset to extract the synthetic objects (in this example we use bottle) and use CycleGAN to render the synthetic objects to realistic. Then by using the mask information again, we can put the realistic result back into the CAMERA RGB again to obtain a more realistic CAMERA++ dataset.

3.5 6D Object Pose Estimation using NOCS Model

To evaluate the effectiveness of our improved dataset, we trained and tested the NOCS model on both the original CAMERA dataset and the new CAMERA++ dataset generated by the CycleGAN model. We used Tensorflow 1.14.0 and Keras 2.3.0 to implement the NOCS model and trained it on both the CAMERA and CAMERA++ datasets, both giving 9K images for training and 1K images for validation. We used an Adam optimizer with a learning rate of

0.001 and a batch size of 16. We evaluated the performance of the NOCS model on a real-world test dataset using the Average Precision (AP) metric.

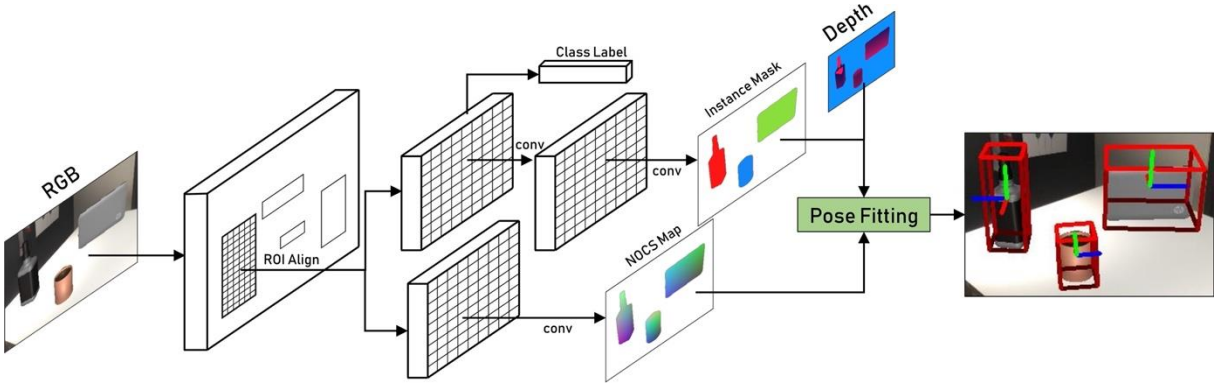


Figure 11 The inputs needed for the NOCS framework are the RGB images together with depth images of a tabletop scene containing many objects. The CNN then predicts the NOCS map (color-coded) for each object and the corresponding class label and instance mask in the RGB image. The NOCS framework then uses each object's NOCS map and the depth image to obtain the full metric 6D pose and size (axes and tight red bounding boxes).

After finishing the training and testing process, we can then make a comparison of the NOCS models trained by two datasets. If the model trained by the improved data shows a good AP score on the object category (i.e., bottle) we rendered by CycleGAN. Then it can truly show that our data improvement method can truly reduce the domain gap influence.

3.6 Summary

In this section, we have described our proposed framework for Synthetic-to-realistic Object Rendering for 6D Object Pose Estimation. We have shown how we obtained a real-world dataset of objects using Objectron and a synthetic dataset of objects using ShapeNet-Core. We have also shown how we trained the CycleGAN model and used CycleGAN to render synthetic object images into realistic object images, and how we evaluated the improved CAMERA++ dataset by train and testing the NOCS model. In the next section, we present the results of our experiments and evaluate the performance of our framework.

Chapter 4: Results and Discussion

In this section, I will show the results of synthetic-to-realistic object rendering based on the trained CycleGAN model, the improved CAMERA++ dataset and the comparison of the performance of NOCS framework trained by the original CAMERA dataset and our improved CAMERA++ dataset.

4.1 Synthetic-to-realistic Object Rendering based on CycleGAN

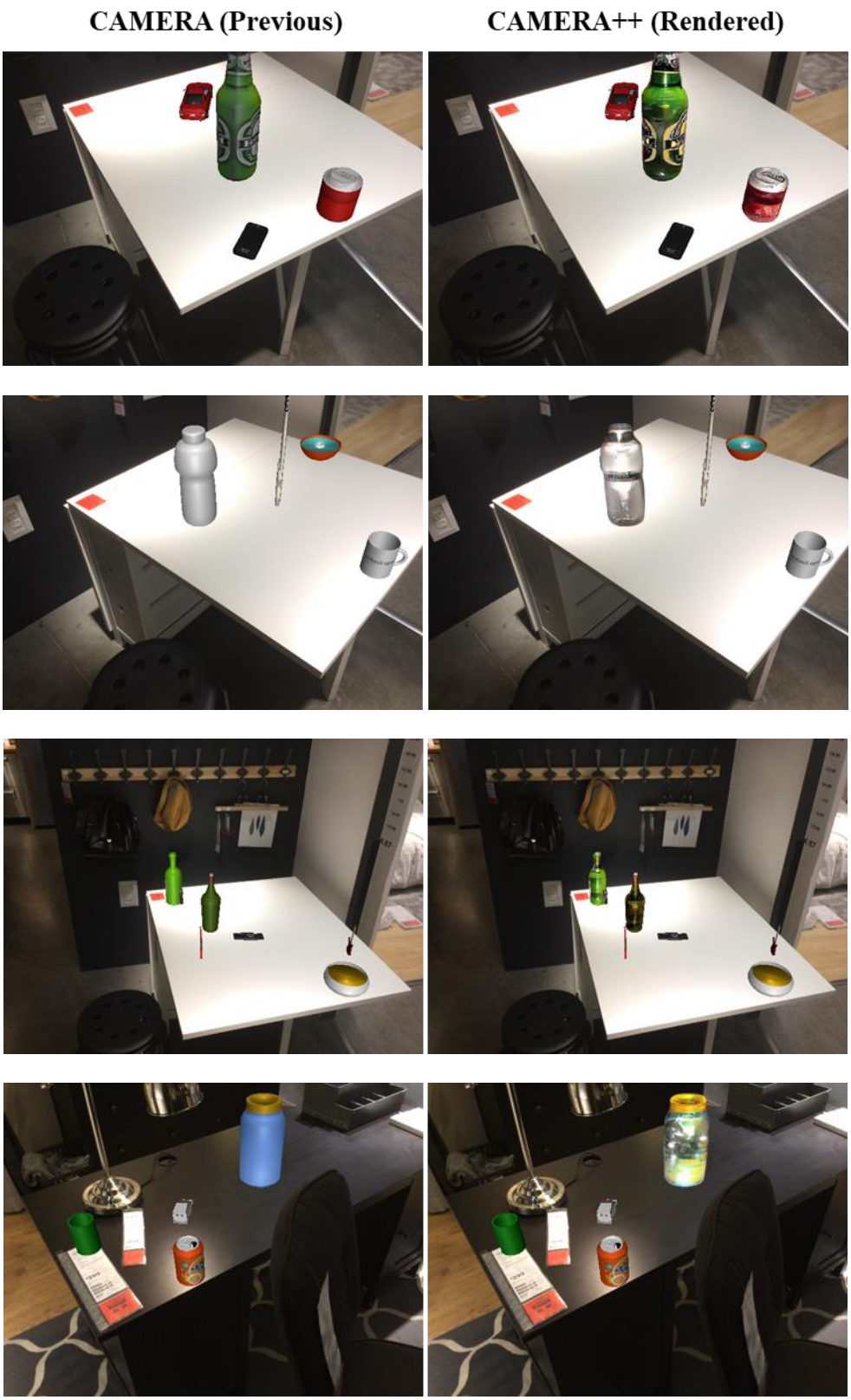
The CycleGAN model will train two generator networks simultaneously and only real synthetic to realistic fake image generator is what we want. So, we write a script to only use the generator we want and use ShapetNet-Core synthetic images as input to get corresponding realistic images.

(A) (a) (B) (b) (C) (c)

Figure 12 Some examples of synthetic-to-realistic rendering by CycleGAN model. Left is synthetic objects(A)(B)(C), right is realistically rendered objects (a)(b)(c).

4.2 Improved CAMERA++ dataset

Since the original CAMERA dataset has mask information for each scene image, we use this mask information to extract synthetic objects in CAMERA dataset and apply these images as the input of our trained CycleGAN model. This process can be the object rendering process. Through this, we can get realistic object output, then by putting them back using the mask information again, we can obtain a more realistic CAMERA mixed reality dataset.


CAMERA (Previous)
CAMERA++ (Rendered)

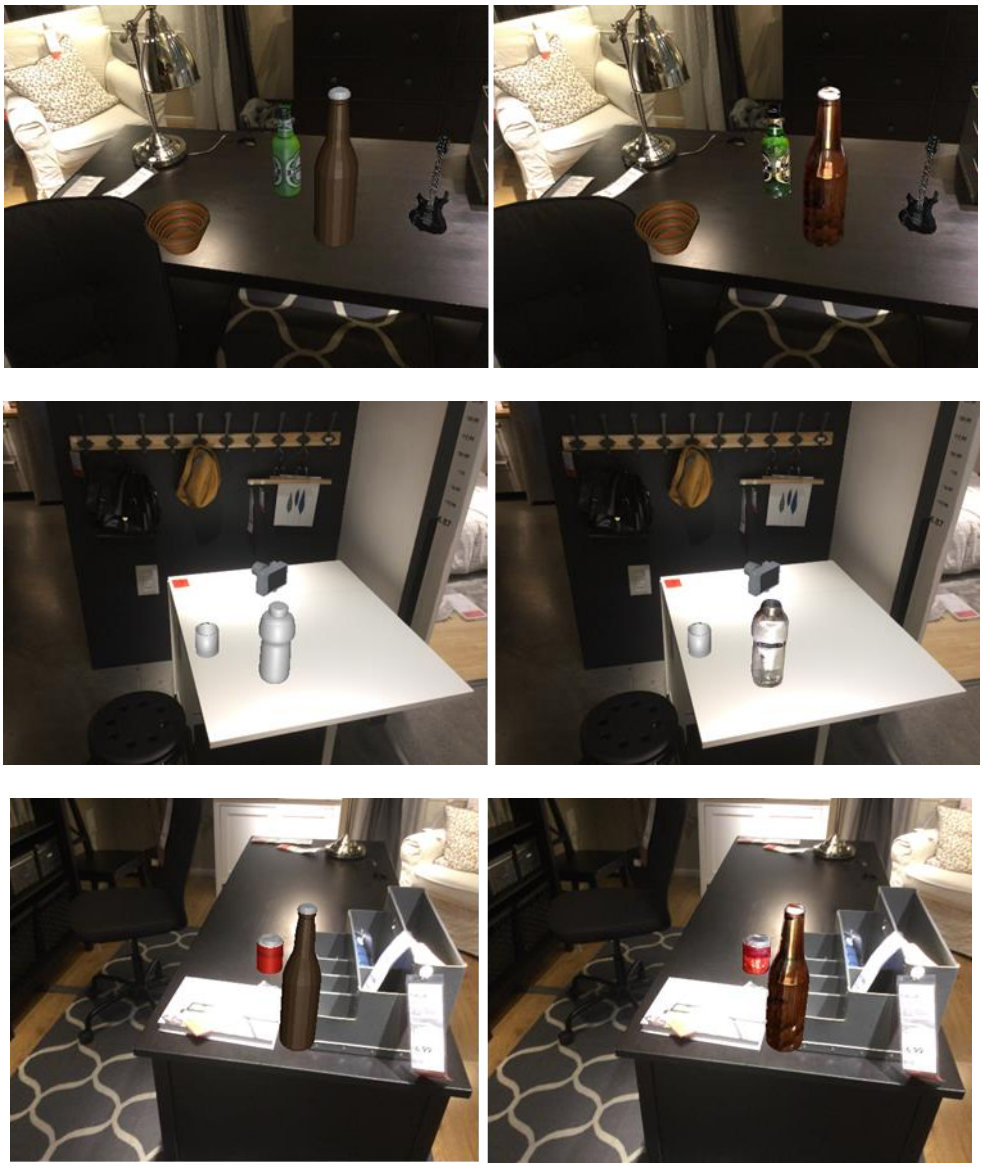


Figure 13 Apply trained CycleGAN model on CAMERA mixed reality dataset to get a more realistic CAMERA++ dataset (We focus on bottle category).

4.3 Improvement of the NOCS Framework

We use some metrics to evaluate the performance of NOCS model. The intersection over union (IoU) metric is used to evaluate 3D detection and object dimension estimation, this is used to evaluate the overlap between the bounding box and the ground truth. For 6D pose estimation, average precision is reported for object instances with an error less than m cm for translation and n◦ for rotation.

Here we first train the NOCS model use 10K original CAMERA dataset, and then use 10K realistically improved CAMERA++ dataset to train another NOCS model. Then we use 2.5K real scene images to evaluate the NOCS model's performance. Since NOCS framework is for category level object estimation, and our trained CycleGAN model is for bottle rendering, we

just need to see the bottle average precision in the evaluation diagrams.

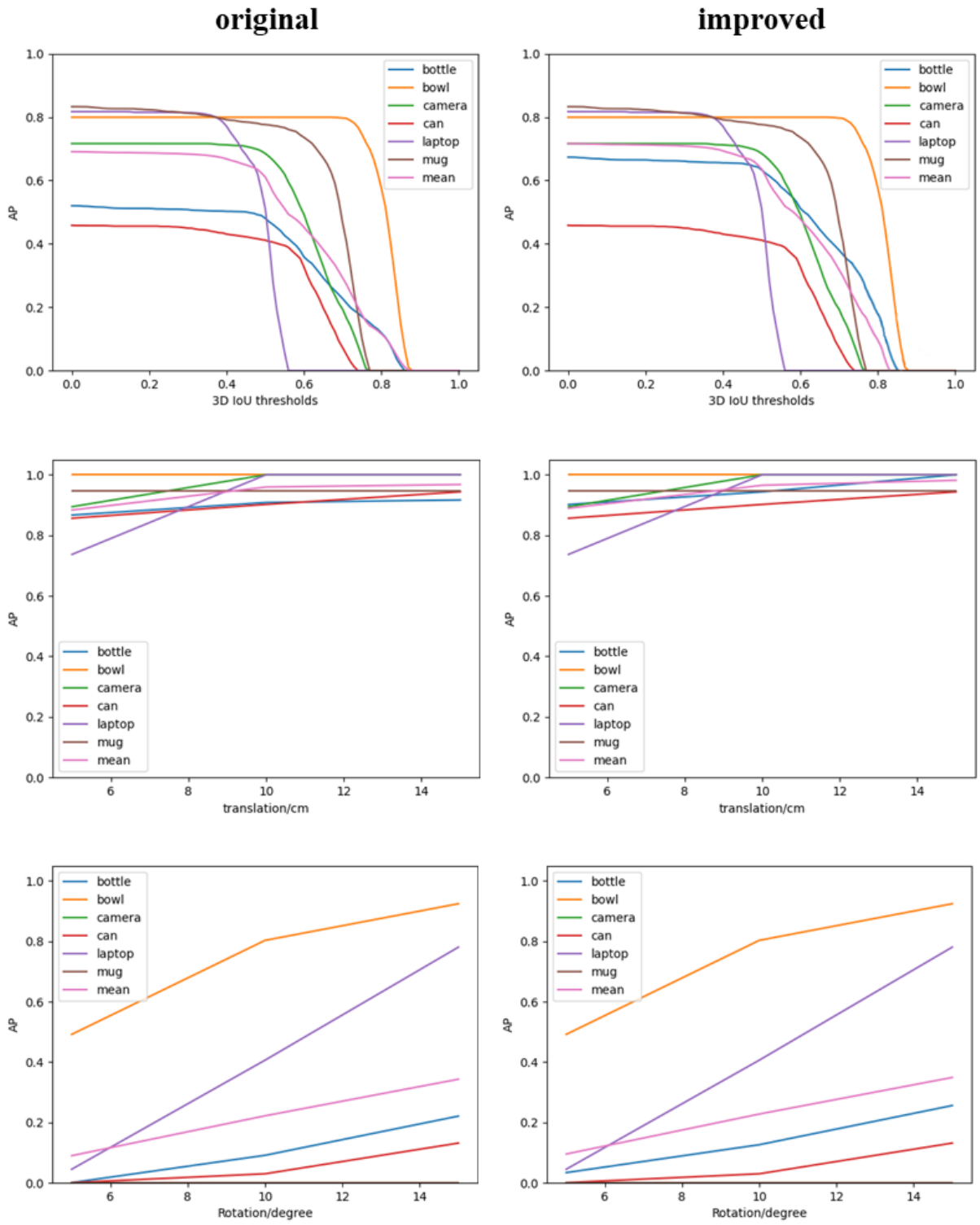


Figure 14 Diagrams of 3D detection and 6D pose estimation results. (We focus on bottle category)

After the detection and evaluation of each model, we can make a comparison here. We can see that the bottle category detection has a better average precision after trained by our improved

CAMERA++ dataset. The average precision of 3D IoU metric increase obviously, close to 15% increased at threshold 0.25 and nearly 10% increased at threshold 0.5. As for the error range metrics in translation and rotation, they all get slightly increased.

Dataset For Training	**mean Average Precision (%)**							
	$3D_{IoU25}$	$3D_{IoU50}$	5° 5 cm	5° 10 cm	10° 5 cm	10° 10 cm	15° 5 cm	15° 10 cm
CAMERA 10K	68.5	61.1	8.5	9.0	21.3	21.7	30.6	33.8
CAMERA++ 10K	71.1	63.9	8.6	9.6	22.0	22.3	32.3	35.3
CAMERA 300K	84.4	76.9	9.8	10.8	24.0	24.5	34.7	36.8

Table 1. Dataset used in training the NOCS model and mean value of the average precision for each metrics

The table shows the mean value of the average precision for each case and makes a performance comparison between the original mixed reality CAMERA dataset and our improved more realistic CAMERA++ dataset.

Through this comparison, we could say our method for realistic training dataset rendering can truly improved the 6D object pose estimation models performance especially the NOCS framework that used mixed reality dataset before. Moreover, this can also demonstrate using realistic rendering for mixed reality dataset can truly reduce the domain gap between the synthetic objects and the real scenes.

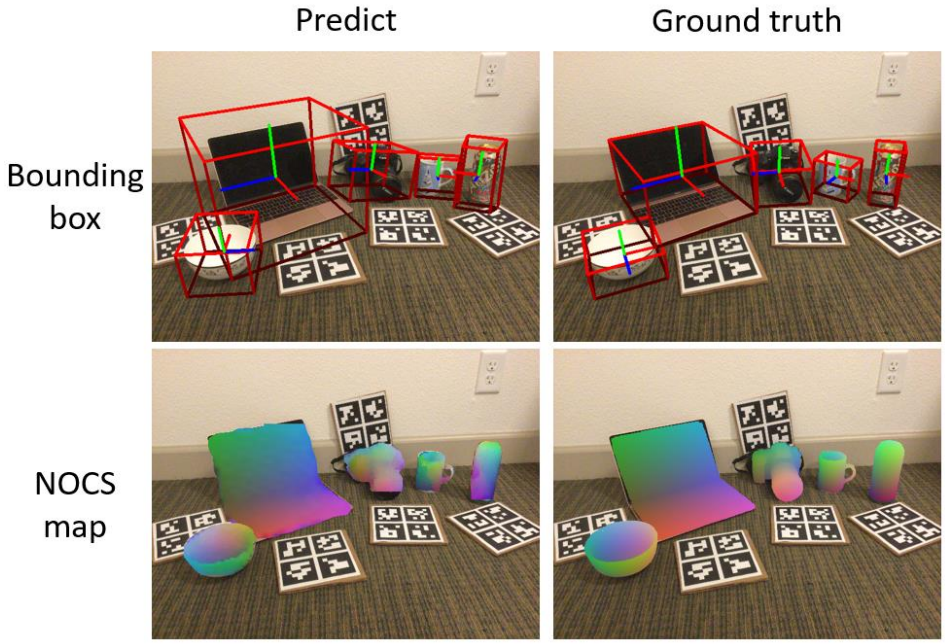


Figure 15 Real test results. The top row shows the quality of the 6D pose (axis) and size estimation (red tight bounding box). The bottom row shows the predicted NOCS maps with color coded.

Chapter 5: Conclusion and Further Work

We proposed a dataset-generation pipeline to improve the performance of the NOCS for category-level 6D pose and size estimation framework, our method for synthetic-to-realistic rendering can truly reduce the domain gap caused by synthetic objects in the mixed-reality dataset. This project gives verification about Generative Adversarial Networks can be used in dataset generation in the computer vision domain, rendering large quantities of realistic data from the synthetic dataset is an ideal direction for CV dataset generation in the future. Through this, we can not only save time and money but also obtain precise labels and masks along with the data generation process.

As for further work, our generation results by CycleGAN show some drawbacks, in this situation, trying other Image2Image translation GAN models may get better results and performance than using CycleGAN. What’s more, we can try our dataset generation method on other Computer Vision frameworks.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014. https://arxiv.org/pdf/1406.2661.pdf

[2] Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV (2017). https://arxiv.org/pdf/1703.10593.pdf

[3] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012, 2015. https://shapenet.org/

[4] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, M. Grundmann. Objectron: A Large Scale Dataset of Object-Centric Videos in the Wild with Pose Annotations. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2021. https://github.com/google-research-datasets/Objectron/

[5] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. ´ In Computer Vision (ICCV), 2017 IEEE International Conference on, pages 2980–2988. IEEE, 2017. https://arxiv.org/pdf/1703.06870.pdf

[6] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song and L. J. Guibas, "Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 2637-2646, doi: 10.1109/CVPR.2019.00275. https://arxiv.org/pdf/1901.02970.pdf

[7] S. Gu, J. Bao, H. Yang, D. Chen, F. Wen and L. Yuan, "Mask-Guided Portrait Editing With Conditional GANs," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 3431-3440, doi: 10.1109/CVPR.2019.00355. https://arxiv.org/pdf/1905.10346.pdf

[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In CVPR, 2017. https://arxiv.org/pdf/1611.07004.pdf

[9] M. Kang, J. Shin, J. Park. StudioGAN: A Taxonomy and Benchmark of GANs for Image Synthesis. 2022. https://arxiv.org/pdf/2206.09479.pdf

Acknowledgement

Here I want to give my biggest acknowledgement to my supervisor. He not only teaches me a lot of knowledge in Computer Vision, but also points out the possible directions of my final project to me. When I meet some bottlenecks or get into some trouble during my project, he is always here to help me and give me constructed advice.

Next, I would like to thank my classmates, my friends and especially my roomies. They are my best partners who help me a lot in my daily life and accompany me to face the obstacles during my project time together.

Finally, my parents are the people that I mostly want to give acknowledgement to. They support me for all the 22 years up to now, it’s my turn now to give my gratefulness back to them.

Appendix

Disclaimer

This report is submitted as part requirement for the undergraduate degree programme at Queen Mary University of London, and Beijing University of Posts and Telecommunications. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

BUPT No.: 2019213135

QM No.: 190897332

Full Name (Pin Yin): Chen Zhiyang

Full Name (Chinese): 陈志扬

Signature:

Date: 2023.4.25

Project specification

Include your project specification, part 1 and part 2 here. It must be the final version submitted to QMPlus.

北京邮电大学 本科毕业设计（论文）任务书

Project Specification Form

Part 1 – Supervisor

论文题目 **Project Title**	Synthetic-to-realistic Object Rendering for 6D Object Pose Estimation		
题目分类 **Scope**	Multimedia and Vision	Research	Software
主要内容 **Project description**	Datasets of images with real objects and their annotated labels are important for training neural networks to estimate 6D object pose. However, capturing and annotation the images with real objects are time-consuming and costly. The goal of this project is to investigate a new model that can render synthetic objects to have a realistic appearance. Pytorch or Tensorflow will be used as a deep learning framework.		
关键词 **Keywords**	generative adversarial networks, image-to-image translation, deep learning		
主要任务 **Main tasks**	**1** Literature survey and understanding of generative adversarial networks (GANs) and synthetic object rendering		
	2 Implementation of the baseline research in GANs for image-to-image translation		
	3 Algorithm design and its implementation		
	4 Evaluation on a dataset		
主要成果 **Measurable outcomes**	**1** Demonstration of understanding in generative adversarial networks (GANs) for image-to-image translation		
	2 A novel algorithm and its implementation		
	3 A performance evaluation of GAN-based realisting object rendering		

北京邮电大学 本科毕业设计（论文）任务书

Project Specification Form

Part 2 - Student

学院 School	International School	**专业 Programme**	**Telecommunications Engineering with Management**		
姓 Family name	Chen	**名 First Name**	Zhiyang		
BUPT 学号 BUPT number	2019213135	**QM 学号 QM number**	190897332	**班级 Class**	2019215104

论文题目 **Project Title**	Synthetic-to-realistic Object Rendering for 6D Object Pose Estimation
论文概述 **Project outline** **Write about 500-800 words** **Please refer to Project Student Handbook section 3.2**	Datasets of images with real objects and their annotated labels are important for training neural networks to estimate 6D object pose. However, capturing and annotation the images with real objects are time-consuming and costly. For example, if we want to analyse the object 6D Pose, we need lots of images that contain various objects in different environments as datasets to train and test our algorithms and models. It's really a waste of time and money to collect each category of object, such as bottles, cups, laptops, cameras, and so on. Also, in computer vision, we need to arrange the environment of the images, for example, we need to put calibration check boards to help the camera detection. It's ridiculous to just do these things by ourselves. The goal of this project is to investigate a new model that can render synthetic objects to have a realistic appearance. Also, generate their corresponding annotated labels (ground truth) together. What we want is automatically generate real objects images datasets which can be used in the deep learning frameworks, so that we can save money and time. Through this project, we can use our personal model to generate realistic datasets automatically by just using some open-source synthetic datasets. For the synthetic datasets, we initially want to use the 'ShapeNet' dataset, which is an open-source dataset containing a variety of 3D synthetic objects. For the rendering progress, we want to first use Blender Python API to automatically generate 2D synthetic images, and together with real objects dataset 'Objectron' to make our own datasets. Then, use this dataset as Generated Adversarial Network (GAN) training and testing dataset. Here, we use Generated Adversarial Network (GAN) as our main framework for object rendering from synthetic to realistic using the Image-to-Image application in Generated Adversarial Network (GAN) area. During the project, Python is the basic programming language we want to use for scripts, and Pytorch or Tensorflow will be used as a deep learning framework. Paper list: • Basic: GAN • GAN based Image-to-image translation: cGAN Pix2Pix, Cycle GAN, UNIT, DRIT • Dataset to be considered: ShapeNet, Objectron • 6D Pose estimation: NOCS
道德规范 **Ethics**	Please confirm by checking the box: ☒ I confirm that I have discussed ethical issues with my supervisor.

Please discuss ethical issues with your supervisor using the ethics checklist in Project Handbook Appendix 1.	Summary of ethical issues: (write "None" if no ethical issues) None
中期目标 **Mid-term target.** **It must be tangible outcomes, E.g. software, hardware or simulation.** **It will be assessed at the mid-term oral.**	● Literature survey about GANs and 6D Pose Estimation ● Make my own training datasets for image-to-image translation by using real object images and rendered synthetic object images ● Run different image-to-image translation GAN models using my own datasets and evaluate the rendered result

Work Plan (Gantt Chart)

Fill in the sub-tasks and insert a letter X in the cells to show the extent of each task

	Nov 1-15	Nov 16-30	Dec 1-15	Dec 16-31	Jan 1-15	Jan 16-31	Feb 1-15	Feb 16-28	Mar 1-15	Mar 16-31	Apr 1-15	Apr 16-30
Task 1 [Literature survey and understanding of generative adversarial networks (GANs) and synthetic object rendering]												
Study CS231n Convolutional Neural Networks for Visual Recognition course	X	X										
Literature survey about GANs and 6D Pose Estimation	X	X	X	X	X	X	X	X				
Synthetic object dataset processing and rendering using Blender Python API (from 3D to 2D)		X	X									
Task 2 [Implementation of the baseline research in GANs for image-to-image translation]												
Run different image-to-image translation GAN models using default datasets	X	X										
Make my own training datasets for image-to-image translation by using real object images and rendered synthetic object images		X	X	X	X							
Run different image-to-image translation GAN models using my own datasets and evaluate the rendered result			X	X	X	X	X					
Task 3 [Algorithm design and its implementation]												
Develop a new image-to-image translation model for the realistic rendering of synthetic objects						X	X	X	X	X		
Find the best performance Synthetic-to-realistic Object Rendering framework among all algorithms						X	X	X	X	X		
Writing mid-term report						X	X					
Task 4 [Evaluation on a dataset]												
Run 6D Pose Estimation model using default datasets					X	X						
Evaluate the rendered datasets by evaluating the results from the 6D Pose Estimation models						X	X	X	X	X		
Writing final report									X	X	X	X

Early-term progress report

Include your project early-term progress report here. It must be the final version submitted to QMPlus.

北京邮电大学 本科毕业设计（论文）初期进度报告

Project Early-term Progress Report

学院 School	International School	**专业 Programme**	**Telecommunications Engineering with Management**		
姓 Family name	Chen	**名 First Name**	Zhiyang		
BUPT 学号 BUPT number	2019213135	**QM 学号 QM number**	190897332	**班级 Class**	2019215104
论文题目 Project Title	Synthetic-to-realistic Object Rendering for 6D Object Pose Estimation				

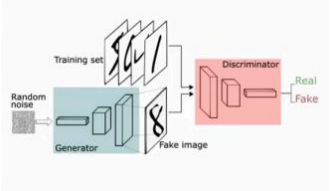
已完成工作 Finished work:

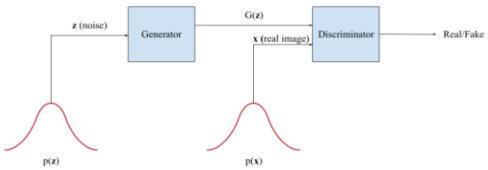
[This is a short report (2-3 pages) stating what progress you have made on your project to date **with evidence**. Please refer to the project handbook section 3.3. – DELETE THIS LINE and WHITE SPACES]

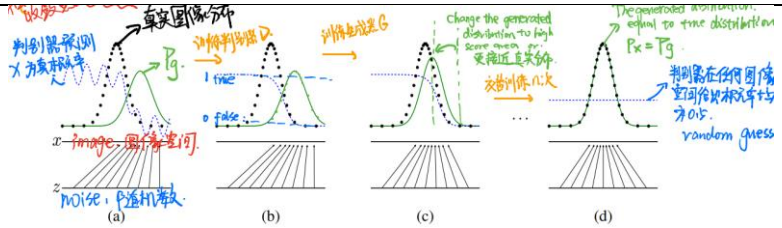
• Summary of literature review (with a list of references)

Datasets of images with real objects and their annotated labels are important for training neural networks to estimate 6D object pose. However, capturing and annotation the images with real objects are time-consuming and costly. For example, if we want to analyse the object 6D Pose, we need lots of images that contain various objects in different environments as datasets to train and test our algorithms and models. It's really a waste of time and money to collect each category of object, such as bottles, cups, laptops, cameras, and so on. To solve this problem, this project wants to investigate a new model that can render synthetic objects to have a realistic appearance. Also, generate their corresponding annotated labels (ground truth) together. What we want is automatically generate real objects images datasets which can be used in the deep learning frameworks, so that we can save money and time.

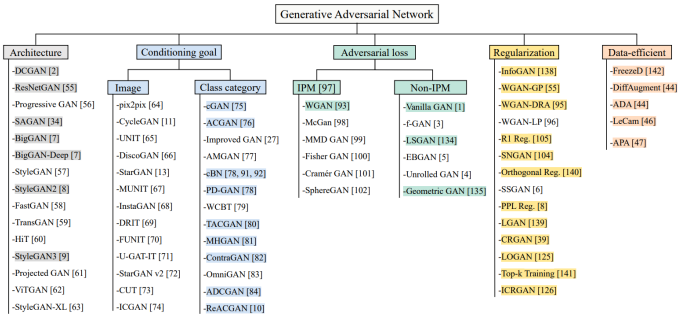
Initially, we need to review Generated Adversarial Network (GAN), a framework proposed in 2014 by Goodfellow et al. This is a framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. This framework revolutionized how we can generate fake high-quality data with neural networks.



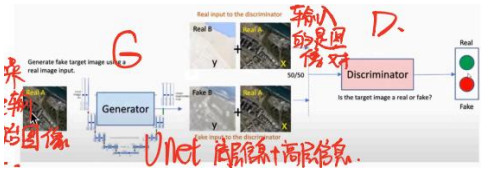




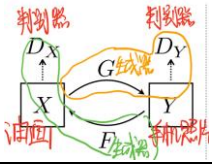
Then after GAN was proposed, many applications and algorithms based on GAN idea appeared. GANs have progressed dramatically from the original models that were proposed by Goodfellow et al. Every aspect from the model architecture, loss functions, and inputs to the datasets we can use and metrics we can evaluate them on have been re-imagined and enhanced by researchers. A survey paper published in 2022 gave a GANs architecture.



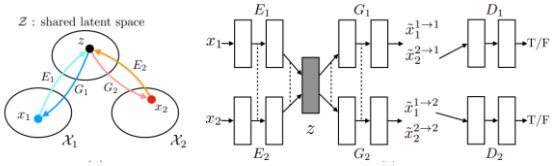
What we need to consider is GAN based Image-to-Image Translation application area. We want to use this image translation function to achieve synthetic to realistic rendering process. Image-to-Image GANs have similarly revolutionized how we can generate hyper-realistic images. Instead of using a random distribution of data as an input, these GANs use image inputs to generate other images. Image-to-Image GANs can be applied to numerous domains of Machine Learning and Computer Vision such as style transfer, colorizing, inpainting, superresolution, future state prediction, object transfiguration, photo editing and enhancement, pose morphing, data augmentation, and many more. Pix2Pix was one of the first GANs that not only achieved Image-to-Image translation but could also, in theory, be generalized to any dataset given sufficient training data and be able to model any of the aforementioned domains without having to alter the model.
I have reviewed many models, and each of these models introduced new advances in how Image-to-Image GANs are created and evaluated and set a new state-of-the-art in visual quality. Each one of these models has its own uniqueness in performing image-to-image translations. Some of these unique features can be summarized as:
– **Pix2Pix** was one of the first models that could be generalized to any dataset without altering its loss function.



– **CycleGAN** removed the limitation of needing paired data to train an Image-to-Image GAN.

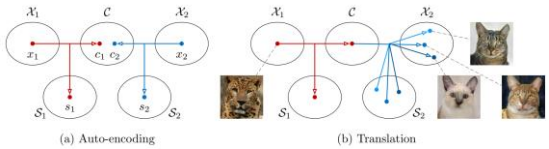


– **UNIT** make a shared-latent space assumption and propose an unsupervised image-to-image translation framework based on Coupled GANs.



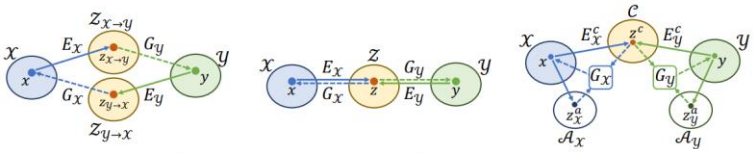
– StarGAN allowed Image-to-Image GANs to map between more than 2 classes with a single model and learn features common to all classes.

– **MUNIT** introduced multi-modal image translation to allow for one image to map to many styles.



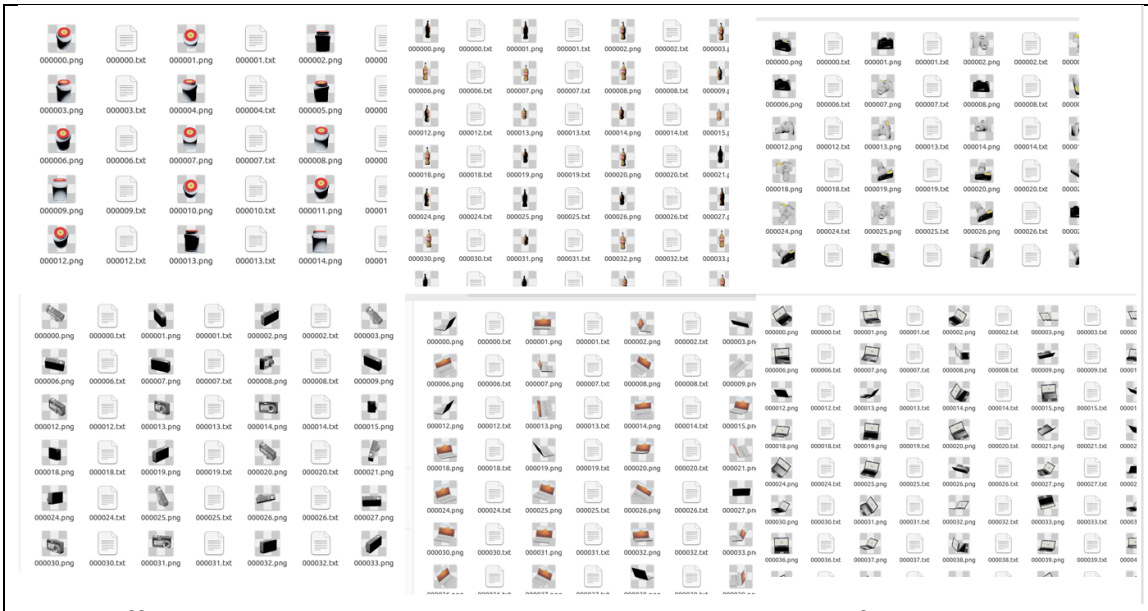
– StarGAN v2 expanded on MUNIT and StarGAN to perform multi-modal translation between more than two classes with a single model.

– **DRIT** present an approach based on disentangled representation for generating diverse outputs without paired training images.



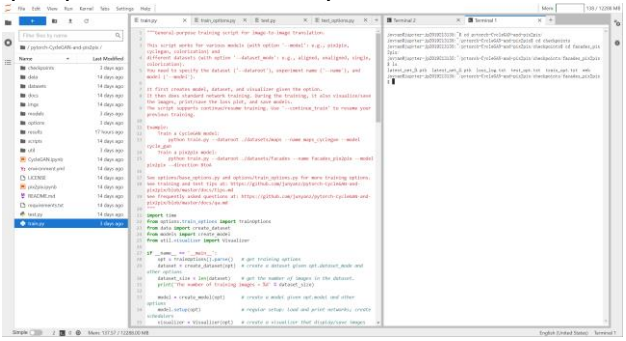
Paper list:
Basic: GAN
GAN based Image-to-image translation: cGAN Pix2Pix, Cycle GAN, UNIT, DRIT, MUNIT
Dataset to be considered: ShapeNet, Objectron
6D Pose estimation: NOCS

• Summary of work was done (add as much details as you have)
- Synthetic object dataset processing and rendering using Blender Python API (from 3D to 2D).
Download bottle, camera, and laptop categories in Shapenet dataset, using blender python API to change .obj files to .gltf files and finally rendered to .png files. From this progress, we successfully got synthetic 2D images.



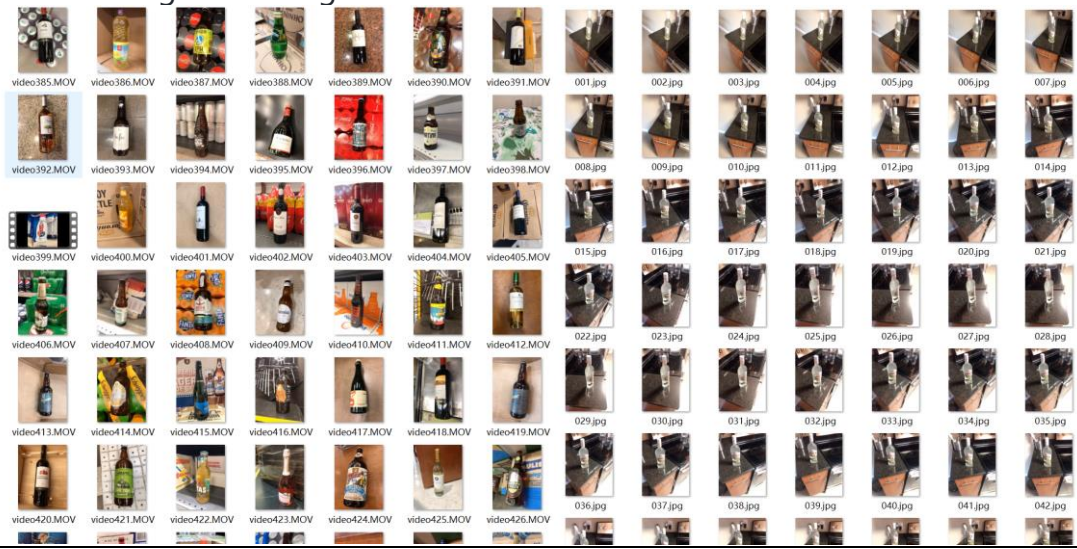
- Run different image-to-image translation GAN models using default datasets

Training & testing the pix2pix model and the Cycle-GAN model on the QM server



- Make my own training datasets for image-to-image translation by using real object images and rendered synthetic object images

After getting the synthetic 2D images, we then turn to find realistic 2D images. We checked the Objectron datasets containing of short, object-centric video clips. In each video, the camera moves around and above the object and captures it from different views. Using the functions from opencv, I successfully processed the video into 2D images referring to each frame.



Code:

```python
VIDEO_PATH = r'C:\Users\Admin\Downloads\Datasets\bottle_video\video.MOV'  # 视频地址
EXTRACT_FOLDER = r'C:\Users\Admin\Downloads\Datasets\bottle_image'  # 存放帧图片的位置
EXTRACT_FREQUENCY = 30  # 帧提取频率


# 主操作
def extract_frames(video_path, dst_folder, index):
    # 实例化视频对象
    video = cv2.VideoCapture(video_path)
    frame_count = 0

    # 循环遍历视频中的所有帧
    while True:
        # 逐帧读取
        _, frame = video.read()
        frame = cv2.flip(frame, 0)
        if frame is None:
            break
        # 按照设置的频率保存图片
        if frame_count % EXTRACT_FREQUENCY == 0:
            # 设置保存文件名
            save_path = "{}/{:>03d}.jpg".format(dst_folder, index)
            # 保存图片
            cv2.imwrite(save_path, frame)
            index += 1  # 保存图片数+1
        frame_count += 1  # 读取视频帧数+1

    # 视频总帧数
    print(f'the number of frames: {frame_count}')
    # 打印出所提取图片的总数
    print("Totally save {:d} imgs".format(index - 1))

    # 计算FPS 方法一 get()
    (major_ver, minor_ver, subminor_ver) = (cv2.__version__).split('.')  # Find OpenCV version
    # (major_ver, minor_ver, subminor_ver) = (4, 5, 4)
    if int(major_ver) < 3:
        fps = video.get(cv2.cv.CV_CAP_PROP_FPS)  # 获取当前版本opencv的FPS
        print("Frames per second using video.get(cv2.cv.CV_CAP_PROP_FPS): {0}".format(fps))
    else:
        fps = video.get(cv2.CAP_PROP_FPS)  # 获取当前版本opencv的FPS
        print("Frames per second using video.get(cv2.CAP_PROP_FPS) : {0}".format(fps))

    video.release()

def main():
    # 递归删除之前存放帧图片的文件夹，并新建一个
    try:
        shutil.rmtree(EXTRACT_FOLDER)
    except OSError:
        pass
    os.mkdir(EXTRACT_FOLDER)
    # 抽取帧图片，并保存到指定路径
    extract_frames(VIDEO_PATH, EXTRACT_FOLDER, 1)


if __name__ == '__main__':
    main()
```

Then, using scripts to combine synthetic 2D images and realistic images and make the dataset aligned, make it fit for pix2pix or Cycle-GAN training and testing.

make_dataset_aligned.py | combine_A_and_B.py

```python
import os
import numpy as np
import cv2
import argparse
from multiprocessing import Pool


def image_write(path_A, path_B, path_AB):
    im_A = cv2.imread(path_A, 1) # python2: cv2.CV_LOAD_IMAGE_COLOR; python3: cv2.IMREAD_COLOR
    im_B = cv2.imread(path_B, 1) # python2: cv2.CV_LOAD_IMAGE_COLOR; python3: cv2.IMREAD_COLOR
    im_AB = np.concatenate([im_A, im_B], 1)
    cv2.imwrite(path_AB, im_AB)


parser = argparse.ArgumentParser('create image pairs')
parser.add_argument('--fold_A', dest='fold_A', help='input directory for image A', type=str,
default='../dataset/50kshoes_edges')
parser.add_argument('--fold_B', dest='fold_B', help='input directory for image B', type=str,
default='../dataset/50kshoes_jpg')
parser.add_argument('--fold_AB', dest='fold_AB', help='output directory', type=str, default='../dataset/test_AB')
parser.add_argument('--num_imgs', dest='num_imgs', help='number of images', type=int, default=1000000)
parser.add_argument('--use_AB', dest='use_AB', help='if true: (0001_A, 0001_B) to (0001_AB)',
action='store_true')
parser.add_argument('--no_multiprocessing', dest='no_multiprocessing', help='If used, chooses single CPU
execution instead of parallel execution', action='store_true',default=False)
args = parser.parse_args()

for arg in vars(args):
    print('[%s] = ' % arg, getattr(args, arg))

splits = os.listdir(args.fold_A)

if not args.no_multiprocessing:
    pool=Pool()

for sp in splits:
    img_fold_A = os.path.join(args.fold_A, sp)
    img_fold_B = os.path.join(args.fold_B, sp)
    img_list = os.listdir(img_fold_A)
    if args.use_AB:
        img_list = [img_path for img_path in img_list if '_A.' in img_path]

    num_imgs = min(args.num_imgs, len(img_list))
    print('split = %s, use %d/%d images' % (sp, num_imgs, len(img_list)))
    img_fold_AB = os.path.join(args.fold_AB, sp)
    if not os.path.isdir(img_fold_AB):
        os.makedirs(img_fold_AB)
    print('split = %s, number of images = %d' % (sp, num_imgs))
    for n in range(num_imgs):
        name_A = img_list[n]
        path_A = os.path.join(img_fold_A, name_A)
        if args.use_AB:
            name_B = name_A.replace('_A.', '_B.')
        else:
            name_B = name_A
        path_B = os.path.join(img_fold_B, name_B)
        if os.path.isfile(path_A) and os.path.isfile(path_B):
            name_AB = name_A
            if args.use_AB:
                name_AB = name_AB.replace('_A.', '.')  # remove _A
            path_AB = os.path.join(img_fold_AB, name_AB)
            if not args.no_multiprocessing:
```

• Problems were faced

Problem such as library cannot install cost a lot of time. During the process I encountered blender python API library ‘bpy’ can just fit python 3.7 but other libraries are installed on python 3.10, that made the scripts cannot use.

• Solutions were found

It is hard to solve on the Windows system cause of the difference between Unix and Linux and the command difference. So I download a VMware virtual machine and Ubuntu os, and then by using Linux command, I successfully rendered the synthetic objects to 2D images.

是否符合进度？On schedule as per GANTT chart?

YES

下一步 Next steps:

Using my own dataset to train and test the pix2pix and Cycle-GAN model, then analyse the result.

Mid-term progress report

Include your project mid-term progress report here. It must be the final version submitted to QMPlus.

北京邮电大学 本科毕业设计（论文）中期进度报告

Project Mid-term Progress Report

学院 School	International School	**专业 Programme**	**Telecommunications Engineering with Management**		
姓 Family name	Chen	**名 First Name**	Zhiyang		
BUPT 学号 BUPT number	2019213135	**QM 学号 QM number**	190897332	**班级 Class**	2019215104
论文题目 Project Title	Synthetic-to-realistic Object Rendering for 6D Object Pose Estimation				

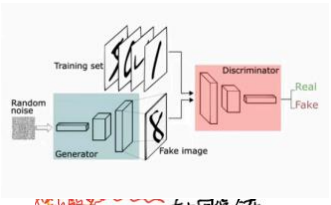
是否完成任务书中所定的中期目标？Targets met (as set in the Specification)?

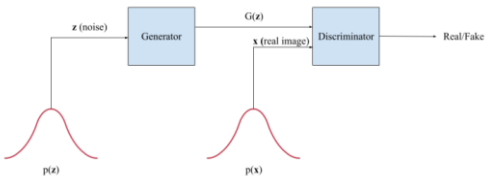
YES

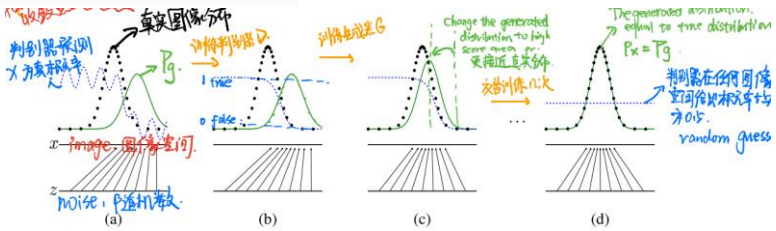
已完成工作 Finished work:

- Literature Survey about GANs and 6D Pose Estimation

Initially, we need to review Generated Adversarial Network (GAN), a framework proposed in 2014 by Goodfellow et al. This is a framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. This framework revolutionized how we can generate fake high-quality data with neural networks.

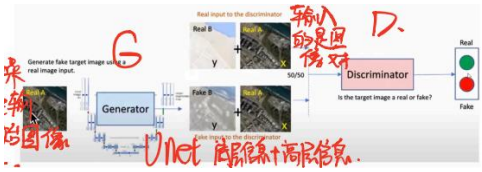




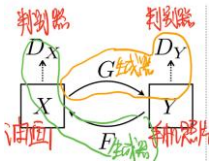


What we need to consider is GAN based Image-to-Image Translation application area. We want to use this image translation function to achieve synthetic to realistic rendering process. Image-to-Image GANs have similarly revolutionized how we can generate hyper-realistic images. Instead of using a random distribution of data as an input, these GANs use image inputs to generate other images. Image-to-Image GANs can be applied to numerous domains of Machine Learning and Computer Vision such as style transfer,

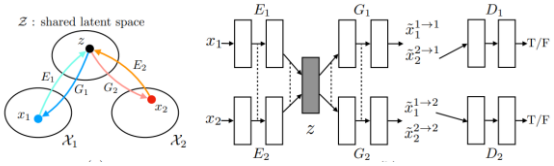
colorizing, inpainting, superresolution, future state prediction, object transfiguration, photo editing and enhancement, pose morphing, data augmentation, and many more. Pix2Pix was one of the first GANs that not only achieved Image-to-Image translation but could also, in theory, be generalized to any dataset given sufficient training data and be able to model any of the aforementioned domains without having to alter the model.
I have reviewed many models, and each of these models introduced new advances in how Image-to-Image GANs are created and evaluated and set a new state-of-the-art in visual quality. Each one of these models has its own uniqueness in performing image-to-image translations. Some of these unique features can be summarized as:
– **Pix2Pix** was one of the first models that could be generalized to any dataset without altering its loss function.



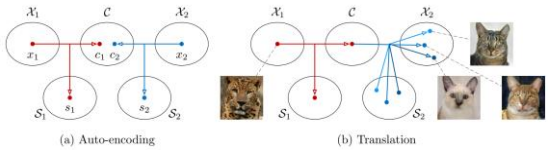
– **CycleGAN** removed the limitation of needing paired data to train an Image-to-Image GAN.



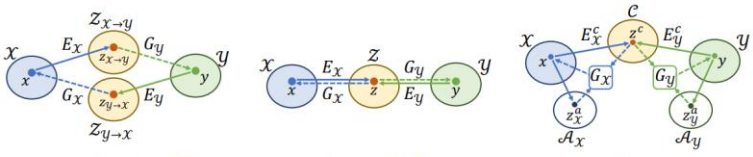
– **UNIT** make a shared-latent space assumption and propose an unsupervised image-to-image translation framework based on Coupled GANs.



– StarGAN allowed Image-to-Image GANs to map between more than 2 classes with a single model and learn features common to all classes.
– **MUNIT** introduced multi-modal image translation to allow for one image to map to many styles.

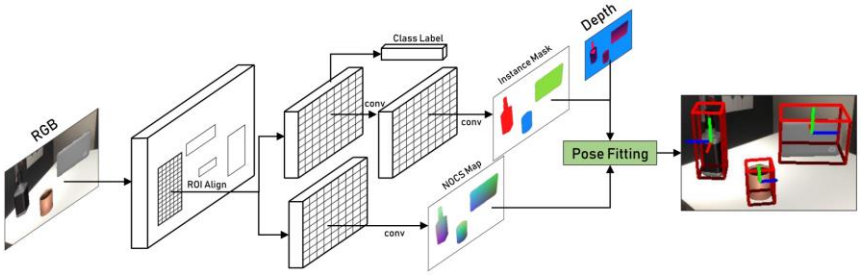


– StarGAN v2 expanded on MUNIT and StarGAN to perform multi-modal translation between more than two classes with a single model.
– **DRIT** present an approach based on disentangled representation for generating diverse outputs without paired training images.

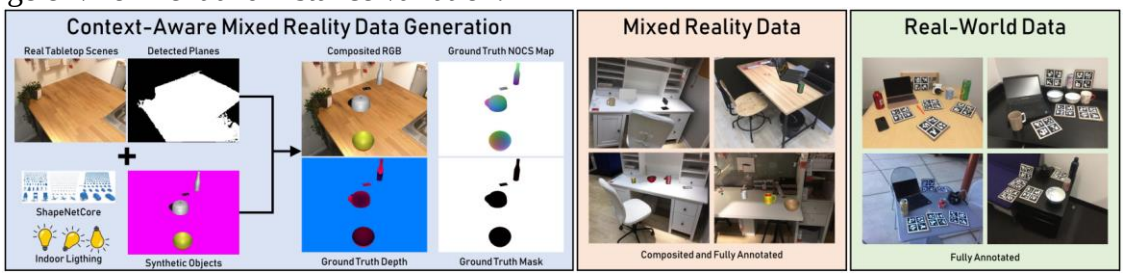


On the other hand, I also review the paper for 6D Pose Estimation. The framework is called Normalized Object Coordinate Space (NOCS) for Category-Level 6D Object Pose and Size Estimation. That paper introduces NOCS, a shared canonical representation for all possible

object instances within a category. The goal of this paper is to estimate the 6D pose and dimensions of unseen object instances in an RGB-D image.



To train the network, the paper presents a new context-aware technique to generate large amounts of fully annotated mixed-reality data. To further improve the model and evaluate its performance on real data, they also provide a fully annotated real-world dataset with large environment and instance variation.



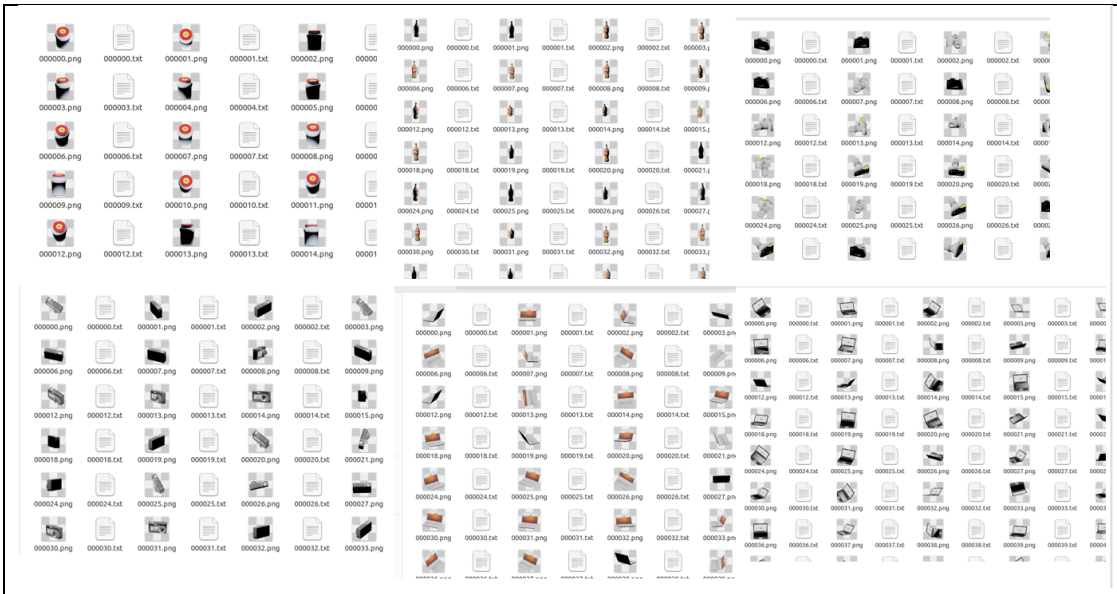
The goal of my project is to use the image-to-image translation GAN application to train a good model, which can generate realistic objects image from synthetic object datasets so that we can improve this NOCS paper's datasets gathered method.

- Baseline implementation (pix2pix, CycleGAN) (horse->zebra)

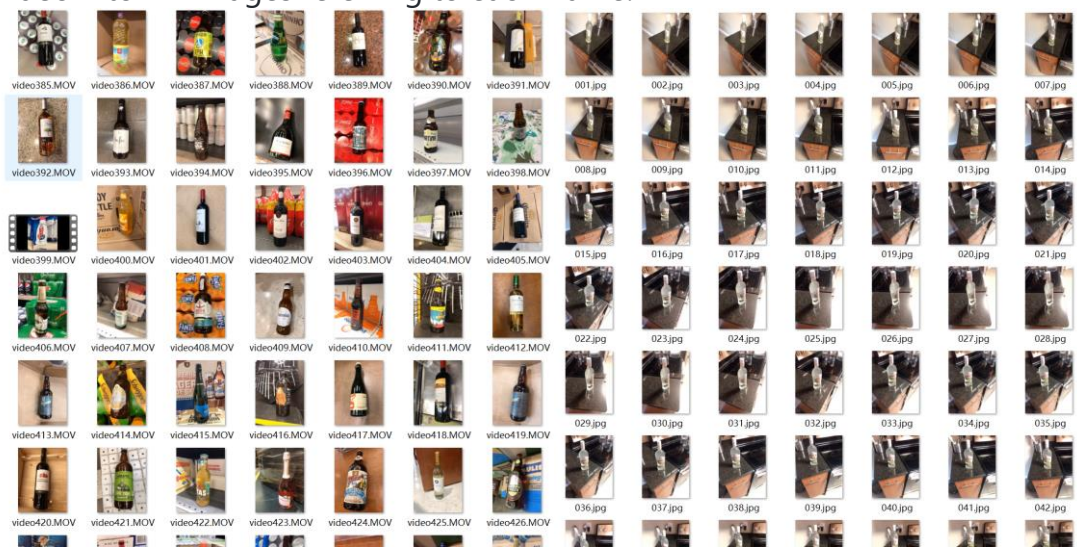
- Make my own datasets for image-to-image translation model training by using real object images(Objectron dataset) and rendered synthetic object images(ShapeNet dataset)

- Synthetic object dataset processing and rendering using Blender Python API (from 3D to 2D).

Download bottle, camera, and laptop categories in Shapenet dataset, using blender python API to change .obj files to .gltf files and finally rendered to .png files. From this progress, we successfully got synthetic 2D images.



- After getting the synthetic 2D images, we then turn to find realistic 2D images. We checked the Objectron datasets containing short, object-centric video clips. In each video, the camera moves around and above the object and captures it from different views. Using the functions from OpenCV, we successfully processed the video into 2D images referring to each frame.



- Train the CycleGAN model using my own object datasets and evaluate the rendered result

I first train the laptop dataset since the laptop objects have the same shape, but after I got the test results, I found that my dataset have very different camera orientations and the real

images contained a lot of environmental content which can influence the model effect.

尚需完成的任务 Work to do:

Run different image-to-image translation GAN models(UNIT, MUNIT) using my own datasets(laptop, bottle, camera) and evaluate the rendered result

Evaluate the rendered datasets by evaluating the results from the 6D Pose Estimation models

Develop a new image-to-image translation model for the realistic rendering of synthetic objects

存在问题 Problems:

1. Python libraries cannot install such as ‘bpy’.
2. The resolution of the rendering results is low and has many noises.
3. Model training cost lots of time, 5 minutes for each epoch.

拟采取的办法 Solutions:

1. Using VMware virtual machine, Ubuntu OS and Blender 2.7 to render ‘ShapeNet’ 3D object files to 2D images.
2. Using masks to only contain the objects and remove the environment.
3. Revise the value of ‘batch-size’ to 16 or 32 to make the training process faster.

论文结构 Structure of the final report: (Chapter headings and section sub headings)

Abstract
Introduction
Related Work
Background and Overview
Datasets
Model Design
Experiments and Results
Conclusion

Supervision log

Include your project supervision log here.

北京邮电大学 本科毕业设计（论文）教师指导记录表

Project Supervision Log

学院 School	International School	**专业 Programme**	**Telecommunications Engineering with Management**		
姓 Family name	Chen	**名 First Name**	Zhiyang		
BUPT 学号 BUPT number	2019213135	**QM 学号 QM number**	190897332	**班级 Class**	2019215104
论文题目 Project Title	Synthetic-to-realistic Object Rendering for 6D Object Pose Estimation				

Please record supervision log using the format below:

Date: dd-mm-yyyy
Supervision type: face-to-face meeting/online meeting/email/other (please specify)
Summary:

Date: 06-9-2022
Supervision type: online meeting
Summary: discussed the GAN paper

Date: 21-9-2022
Supervision type: online meeting
Summary: discussed the GAN paper

Date: 05-10-2022
Supervision type: online meeting
Summary: discussed the 6D pose estimation paper

Date: 02-11-2022
Supervision type: online meeting
Summary: discussed the open-source datasets choosing

Date: 16-11-2022
Supervision type: online meeting
Summary: discussed the project specification

Date: 30-11-2022
Supervision type: online meeting
Summary: discussed the draft specification

Date: 11-01-2023
Supervision type: online meeting
Summary: recall the process have been done before the exam

Date: 25-01-2023

Supervision type: online meeting
Summary: discussed the Cycle-GAN training datasets

Date: 08-02-2023
Supervision type: online meeting
Summary: discussed the Cycle-GAN training process

Date: 17-02-2023
Supervision type: online meeting
Summary: discussed the mid-term slides and the report

Date: 22-02-2023
Supervision type: online meeting
Summary: discussed the mid-term slides and the report

Date: 10-03-2023
Supervision type: online meeting
Summary: discussed the 6D Pose estimation

Date: 24-03-2023
Supervision type: online meeting
Summary: discussed the 6D Pose estimation

Date: 05-04-2023
Supervision type: online meeting
Summary: discussed the result of CycleGAN and mask loss

Date: 14-04-2023
Supervision type: online meeting
Summary: Mock Viva

Date: 21-04-2023
Supervision type: online meeting
Summary: discussed the final report and slides

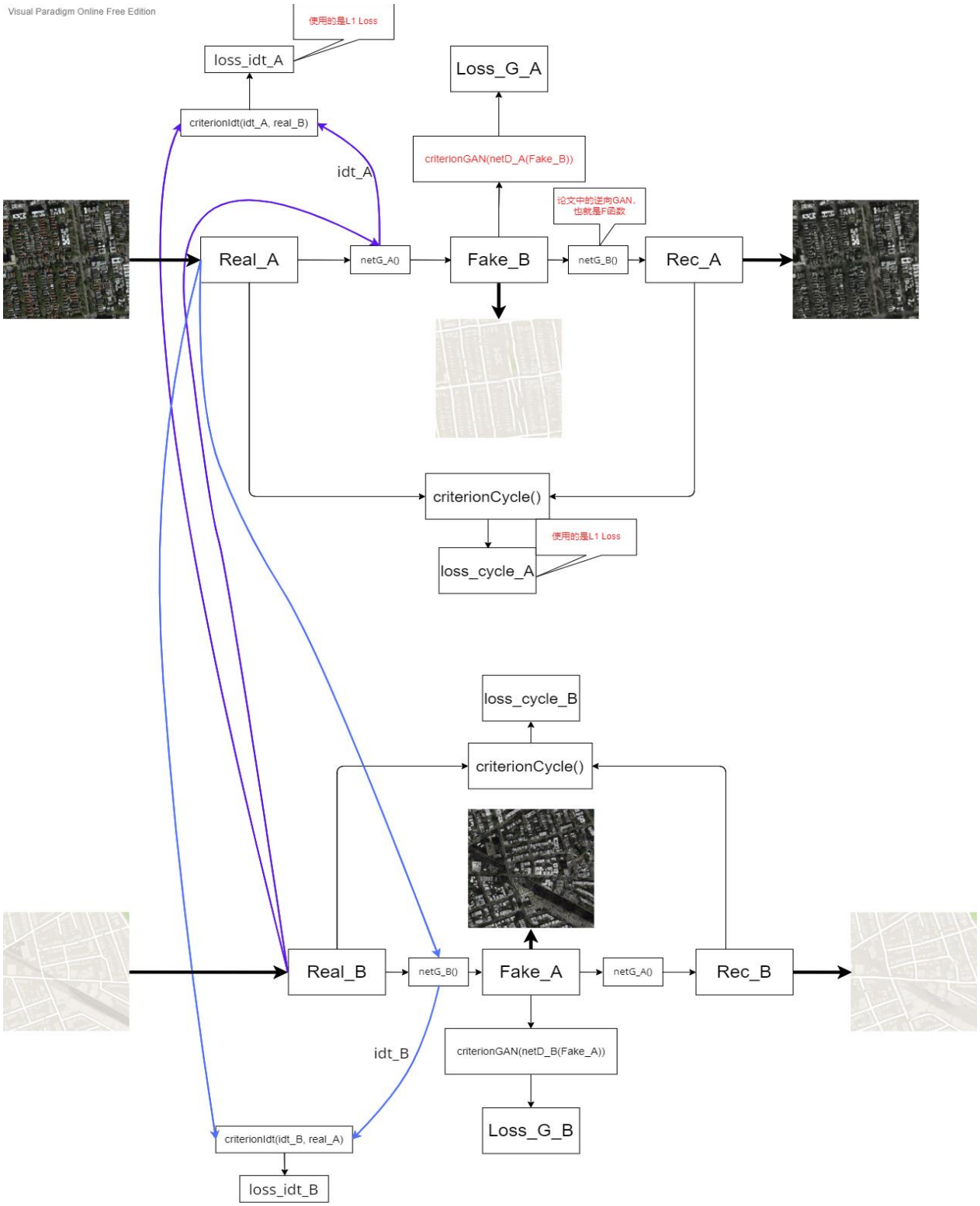
Additional Appendices (as needed)

- **Training and Testing environments:**

For CycleGAN: (Run on QueenMary IT Server)
Repository address: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
OS: Linux 5.18.13-100.fc35.x86_64
RAM: 500GB
Python version: 3.10.5
Python libraries:
pytorch=1.4.0
torchvision=0.5.0
dominate=2.4.0
visdom=0.1.8
wandb=0.15.0

For NOCS 6D pose and size estimation:
Repository address: https://github.com/hughw19/NOCS_CVPR2019
OS: Windows11
RAM: 32GB
CPU: i5-10400F GPU: RTX3060
Python version: 3.6.13
CUDA version: 10.0.13
CuDNN version: 7.4.1.5
Python libraries:
tensorflow-gpu=1.14.0
keras=2.3.0
scipy=1.2.1
h5py=2.1.0
open3d=0.15.1
numpy=1.19.5
pycocotools=2.0.6
opencv-python=4.3.0
pillow=8.4.0
cython=0.29.33
matplotlib=3.3.4
imgaug=0.4.0
scikit-image=0.17.2
IPython=7.16.3

- **Draft figure about CycleGAN loss functions, networks, and the input/output**



loss_G = loss_G_A + loss_G_B + loss_cycle_A + loss_cycle_B + loss_idt_A + loss_idt_B

Risk and environmental impact assessment

- **The open-source dataset used to train CycleGAN is restricted due to copyright or other issues**

Likehood level L: 1. Rare. May happen in exceptional circumstances
Consequence level C: 4. Major. Problem so severe that it is unlikely the project can be completed. Some aspects may be salvageable.
Rating: Moderate Risk

- **The QM server used by training the model is disconnected**

Likehood level L: 1. Rare. May happen in exceptional circumstances
Consequence level C: 3. Very Serious. Will cause a significant disruption to project progress but completion still possible.
Rating: Low risk

- **The Image-to-Image Translation function of the CycleGAN cannot achieve synthetic-to-realistic object rendering.**

Likehood level L: 0. Impossible. Cannot happen.
Consequence level C: 2. Serious. Might cause slight disruption to project progress but will not prevent completion.
Rating: No Risk

- **Objects rendering based on CycleGAN or GANs may perform well on bottle category but not on other categories**

Likehood level L: 2. Unlikely. Could happen at some time
Consequence level C: 1. Minor. Undesirable but something that can be handled without affecting the overall progress of the project.
Rating: Low risk