# GO!GAN

## Grasping Objects with Generative Adversarial Networks

Losada de la Rosa, Fran J.

*Award date:*
2020

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

# GO!GAN: Grasping Objects with Generative Adversarial Networks

*Master Thesis*

Francisco Jose Losada de la Rosa

Supervisors:
Joaquin Vanschoren
Mike Holenderski
Nico van Engelenhoven

Eindhoven, August 2020

# Preface

The author would like to acknowledge the support, help and flexibility given by my supervisors, both from Greenhouse Labs Research (Tim Deynen and Nico van Engelenhoven) and Eindhoven University of Technology (Joaquin Vanschoren), to develop this Master Thesis.

Also, the unconditional and continuous support from my girlfriend with whom I have shared these last lock-down months on the COVID-19 pandemic, my family, friends and Greenhouse interns.

# Contents

# List of Figures

# Chapter 1

# Introduction

Image editing encompasses the process of altering images, and has been an active and interdisciplinary research topic from several decades. Images constitute the core element in different fields like marketing, television, arts, and more recently computer vision. Graphic software programs such as vector graphics editors and raster graphics have been the primary tools with which users manipulate, alter and enhance images.

In the last decades, the advances in Artificial Intelligence (AI), with improved neural networks together with increased computing power, has led to strong improvements in the field of computer vision. Computer vision aims to gain a high-level understanding of digital images and video, seeking to understand and automate tasks that were exclusive to the graphic software programs. In this context, generative models are attracting the attention of the research community and the last decade has experienced an important upturn since the Generative Adversarial Networks (GANs) appeared in 2014, which researchers as Yann Lecun claim to be the most interesting machine learning idea of the decade.

GANs were first used for image generation, as a way to infer an artistic component to an AI. Soon, GAN applications expanded to fields like image inpainting, future state prediction, style transfer and superresolution. The use of conditional GANs opened the door to learning specific image transformations, with strong enough capabilities to generate what are known as "Deep Fakes", in which a person in an image or video is edited in a way that looks like real content when it is actually fake. In the realm of "Deep Fakes", most of the existing applications transform existing components of the image into different shapes or positions. As so, mouth position can be modified so the person seems to be speaking, or the body position can be changed into different postures.

Introducing new objects into the image is one of the next challenges to be faced by generative models. It is especially challenging because 2D images lack the third dimensional depth information, finding it hard to render new objects in a spatially coherent way. In this study, we focus on the rendering of new objects that will be held by a previously existing hand in the source image. The research question of this study can be formulated as follows:

"How do we infer a hand-object grasp generation in a 2D image framework?"

The main contributions of this work are the following:

- Design a modeling framework to learn hand-object grasps adapting the Pix2Pix architecture.
- Define an evaluation metric to asses the quality of the generated images.
- Engineer the model loss to obtain better qualitative and quantitative results.

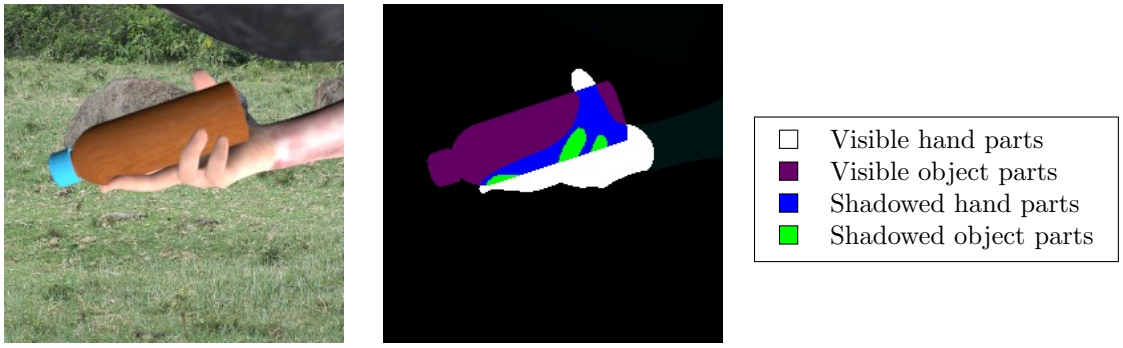
# Chapter 2

# Problem Statement

A grasp is the act of taking, holding or seizing firmly an object with our hands. It is a human primitive act that humans acquire in early stages of their life. We are born with the "palmar grasp reflex", one of our primitive reflexes that makes baby's fingers close and grasp an object placed in the infant's hand. Over the years, we develop this grasp capacity to be able to hold any kind of object shape, adapting our hand to the object characteristics. A grasp is, inevitably, a 3D action that requires our hand to understand the object to be held.

When dealing with images, there are 2D and 3D images that refer to the actual dimensions of information contained in the image. A 2D image is "flat", meaning that only the horizontal and vertical (X and Y) dimensions exist, and if turned to the side becomes a line. A 3D image adds a the third depth (Z) dimension, which allows the image to be rotated and visualized from any perspective.

Dealing with grasps in 2D is then, a challenging task that requires us to reformulate the act of grasping and understand it without depth information. To decompose the problem, we take a closer look the example of a hand grasping a bottle. We can clearly identify the two key elements that are part of the equation, that is: the hand and the object. A key point is that not all the hand is visible, and not all of the object is visible. We can identify four differentiated parts in the grasp 2.1. The parts of the hand and object which occupy a spatial position in the image in which do not shadow each other (purple, white) and the parts of the hand that occupy the same spatial position, and thus shadow each other being on the foreground or background (green, blue).




(a) Hand-object grasp (b) Hand-object interaction parts

Figure 2.1: Decomposition of the hand-object grasp attending to the 2D interaction between the hand and the object.

The core of the model learning lies in the shadowed parts of both hand and object. Our model will have two inputs, and will have one output 2.2. One of the inputs will be the hand without holding the object, and the output will be the hand holding the object. As we will see in the literature review section, conditional masks are used in generative models to provide information tailored to the task we are trying to solve, this will be our second input. As we will see in more detail in sections 5 and 7, if the mask provides too much information like what parts of the object should be shadowed, the model will not learn the grasp. On the other side, and if we provide too little information, for example, no information about the object appearance, the model will not be able to learn how to represent it.

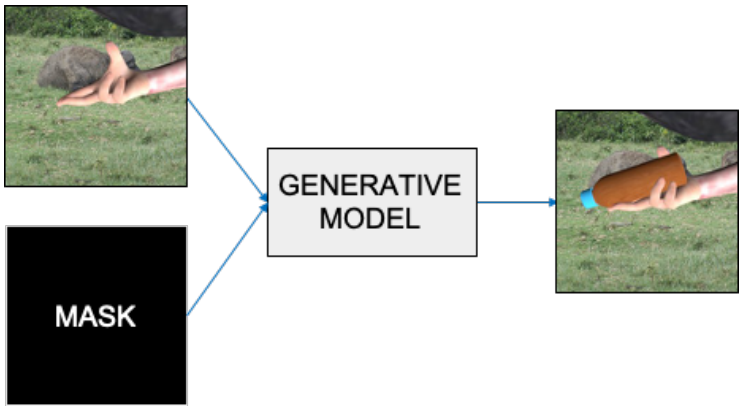



Figure 2.2: General overview of the model inputs and output.

The challenge being tackled works under very specific input conditions. To be able to learn the grasp, both hand and object must be in a position that allows the grasp. This means that there are two underlying challenges to this setting, and that need to be solved in our framework to work:

- Hand pose manipulation: The hand pose needs to be transformed to a grasp position in which the integration with the object grasp makes sense.
- Object pose manipulation: The object in the image needs to be in a specific shape and position.

# Chapter 3

# Preliminaries

In this section we introduce convolutional neural networks, providing enough background on this deep learning approach that will be used throughout the work. Then we deep dive into generative models, and particularly into Generative Adversarial Networks, that will be the base of our model and whose details need to be understood for the further design decisions to be made.

## 3.1 Convolutional Neural Networks

Images are high dimensional data structures in which different low, medium and high-level features are present. A first approach to deal with this data would be to flatten the image matrix into an array vector and use it as input to a neural network. The problem is that this approach fails to capture the spatial and temporal dependencies existing in the image, loosing a lot of valuable information needed in classification tasks.

The best way to capture all the dependencies exiting in an image, going from the low-level to the high-level features, is the use of convolutional neural networks (CNN).

### 3.1.1 Convolution

In a convolution layer, there are the following key elements:

- Kernel: 2D square that usually has the shape of 3x3 or 4x4 pixels. The kernel traverses the input image from the top left to the button right.
- Filter: Operation that is done with the pixels contained in the kernel and that results in one value contained in a neuron. That neuron is said that covers the receptive field of the pixels involved in the operation
- Stride: Value that indicates how many pixels does the kernel move to perform the next filter operation.
- Padding: By nature, the act of convoluting generates a lower dimensional image. A technique to avoid this is to populate the borders of the image with 0s in a way that the output dimension stays the same.

The kernel moves from the top right to the bottom left of the image until all the image is completed. The iterative process of moving the kernel throughout the image applying the filter operation with a certain stride and padding can be mathematically expressed as a matrix multiplication. To do that, we need to create the convolution matrix first. The convolution matrix represents in a matrix the moves of the kernel throughout the image. Each row is a flattened version

of the image that is polulated with 0s if the kernel is not in the position and with the kernel values if the kernel is placed there.

If we have a 3x3 image and a 2x2 kernel with stride one, the kernel moves 4 times, which means that the convolution matrix has four rows. Each row has 9 columns that represent each of the pixels of the 3x3 image, resulting in a (4x9) matrix. If we flatten the image as a (9x1) matrix. Now we can multiply the (4x9) convolution matrix with the (9x1) flatten image, getting a flatten (4x1) version of the 2x2 image resulting from the convolution.

In a 256x256 image, if the kernel size is 3x3, the receptive field that covers each of the filter neurons is very small. For this reason, the first convolutional layers capture low-level features of the input image. Following with this example, and with a configuration that reduces the size of the image by a factor of two, a second convolution layer of 3x3 on the 128x128 resulting image would have a receptive field on the 256x256 original image of 9x9, and medium-level features would be captured. This process goes on until the highest level features are gotten, which allows for example to classify an image by category.

### 3.1.2 Transpose Convolution

Transpose convolutions are used as an up-sampling technique to go from low-dimensional images to bigger ones. Contrary to other data-independent techniques in which the up-sampling method is predefined, transpose convolutions are able to learn from data, making them widely used in deep learning image generation.

Following with the example previously used for convolution, we have the result of the convolution (2x2) and the shape of the image that resulted in that value (3x3). If we built the convolution matrix as explained before, we get a (4x9) matrix. We cannot multiply a (4x9) matrix with the flatten image (4x1). The solution is to transpose the convolution matrix so its shape becomes (9x4) and multiply it by the (4x1) flatten image, resulting in a (9x1) flatten version of the (3x3) image before convolution we were looking for.

### 3.1.3 Activation Function

In a neural network forward pass, a neuron calculates a weighted sum of its input (neuron value multiplied by neuron weight) and adds the bias term. Once that value is obtained, the activation function takes the decision of what the output will be.

The ReLU activation 3.1 function truncates negative values to 0 and lets positive values pass unaltered. ReLU, however, often faces what is known as the "dying Relu" problem [11], in which neurons get stuck in a value that ReLU will always output 0, and does not allow the flow back of the gradients through the network. This problem is especially significant when training GANs, since the generator needs to receive the gradients from the discriminator in order to learn. Leaky ReLu 3.1 adds a small slope for negative values, which fixes the "dying Relu" problem.

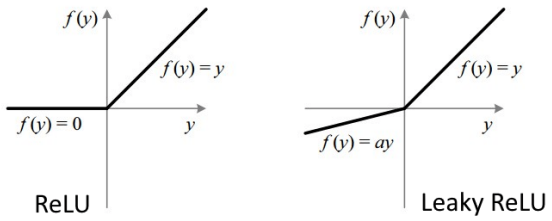



Figure 3.1: ReLU and LeakyReLU activation functions.

## 3.2 Generative Models

Generative models attempt to learn the data distribution of a training set. When modelling high-dimensional data such as images or audio, the model learns a low-dimensional distribution representation which is as similar as possible to the true data distribution.

Deep Generative Models make use of neural networks to learn those data feature representations. Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) are two approaches used in this area.

### 3.2.1 Variational Autoencoders

VAEs are composed of two main components, an encoder and a decoder. The encoder performs a dimensionality reduction of the input data, limiting the features (pixels in the case of images) that describe it. This process of gradually encoding the input to a low-level representation or latent space is a lossy compression, meaning that the original data dimension can not be recovered back. The VAEs decoder does the reverse task, going from the encoded latent space to gradually get again to a high-level representation.

Autoencoders can be trained to minimize a loss function between the input and output images, and so learning the best encoding-decoder scheme. VAEs, contrarily to vanilla autoencoders, generate latent spaces, which are continuous by design, allowing random sampling and interpolation. This feature allows the decoder not only to replicate existing images, but to randomly sample from the latent space, or generate variations on an input image.

### 3.2.2 Generative Adversarial Networks

Generative Adversarial Networks [13] were introduced by Ian Goodfellow in 2014, and soon started to be used in images with deep convolutional neural networks [25]. This new technique is inspired by games theory approach by which two networks, generator and discriminator, compete between each other. The objective is to achieve what in game theory is known as 'Nash equilibrium' by which both networks would keep improving indefinitely by competing among each other. The generator network captures the data distribution and generates and image out of that distribution, and then the discriminator network estimates the probability that the sample comes from the training data rather than from the generator. The generator learns how to make synthetic images look more real trying to mislead the discriminator, while the discriminator tries to keep classifying correctly not getting fooled by the generator.

In the vanilla GAN, Gaussian noise is fed to the generator, which after training will have learnt the data distribution of the training data. This initial configuration presents some limitations, such as the possibility of learning only one data distribution per dataset.

#### Conditional Generative Adversarial Networks

As a straightforward extension of this framework, the creation of a conditional model of GANs was presented months later and named CGAN [22]. It solved the unconditioned previous model by the addition of class labels for the data generation. These class labels were added to the random noise input received by the generator, acting as conditioning information that allows us to introduce a desired behaviour in the generated image. The generator operates in a way that learns a mapping from an observed image x and random noise vector z, to y, $G : \{x,z\} \longrightarrow y$, while the discriminator operates as a binary classifier. A new broad sub-field in GANs was born, which we will refer to as conditional image synthesis, being the task of generating images conditioned on certain input data. (Note that the term image-to-image translation is also commonly used, but we avoid to use it here due to the possibility of using image-to-image mappings using GANs unconditionally.)

### 3.2.3 Loss Functions in GANs

**Cross-entropy loss**

Also known as log loss, it is used for classification tasks, and measures the performance of a model whose output is a probability vector that represents predicted probabilities of all classes to classify, summing up to 1. This can be achieved by having the last layer activated by a softmax function for example. When we compute this loss to classify an image of class A, the loss does not penalize if the model gives a high probability to class B. Instead, it penalizes more the low probability given to class A by the model. In other words, the loss penalizes more if the probability given to the correct class is lower.

$$CE = -\sum_{x} p(x) * \log_{10} q(x) \tag{3.1}$$

Where p(x) = probability of class x in target, q(x) = probability of class x in prediction

As the loss only pays attention to the predicted probability, the other terms do not need to be calculated as they will always be zero. This can be done if the our target is a one-hot vector, by calculating only the loss for the hot class. This is known as categorical cross-entropy loss.

$$CCE = -\log_{10} q(x) \tag{3.2}$$

For binary cross-entropy, we have two classes, zero and one. The output is a predicted value that is shirked between zero and one by using a sigmoid activation function. So, in order to keep using the loss as before, we one-hot encode the two classes (1,0) and (0,1) and transform the predicted value to a probability vector, assigning to the other class the probability of one minus the prediction. The first and second term of the formula cancels out if the target is zero or one respectively.

$$BCE = -(p(x) * \log_{10} q(x) + (1 - p(x)) * (1 - \log_{10} q(x))) \tag{3.3}$$

**MAE and MSE loss**

Mean Absolute Error, also known as L1 loss, is used for regression models. It is the sum of the absolute differences between the target and predicted variable.

$$MAE = (\frac{1}{n}) \sum_{i=1}^{n} |y_i - x_i| \tag{3.4}$$

If, instead of the absolute differences, we compute the squared distances, then it becomes the Mean Square Error (MSE) or L2 loss. The mean difference between both resides in how they deal with outlier values. As L1 loss simply takes an average of differences, it is robust against outliers as they will not greatly affect the whole average. On the other hand, L2 loss squares those differences, hence making outliers have a much bigger impact on the final loss value.

$$MSE = (\frac{1}{n}) \sum_{i=1}^{n} (y_i - x_i)^2 \tag{3.5}$$

**Adversarial Loss**

To understand how the adversarial loss works, we need to take a step back to the BCE loss defined in 3.2.3. The discriminator acts as a binary classifier that aims to classify real images as real, and fake images as fake. When a real image comes into the discriminator (p(x) = 1), the second term cancels out. In the same way, when a fake image generated by the generator comes in (p(x) = 0), the first term cancels out

$$BCE(D(x), 1) = \log_{10} G(x) \tag{3.6}$$

$$BCE(G(z), 0) = \log_{10}(1 - D(G(z)) \tag{3.7}$$

Thus, the objective of the discriminator is to maximize these two functions.

$$\mathcal{L}(D) = max[\log_{10} G(x) + \log_{10}(1 - D(G(z))] \tag{3.8}$$

The generator is competing against the discriminator, trying to minimize the above equation.

$$\mathcal{L}(G) = min[\log_{10} G(x) + \log_{10}(1 - D(G(z))] \tag{3.9}$$

Putting together the objective functions of both generator and discriminator, the final GAN objective is:

$$\mathcal{L} = min_G max_D[\log_{10} G(x) + \log_{10}(1 - D(G(z)))] \tag{3.10}$$

In this original setting, the mini-max loss game proved to saturate the generator loss in the early stages of the training [13]. This made it necessary to re-frame the idea in a way that both the generator and discriminator had to maximize a loss function. To do this, the generator technique is changed, and instead of minimizing the discriminator objective, it maximizes the chances that the discriminator classifies a fake image as true.

The discriminator acts a binary classifier which tries to maximize the binary cross entropy loss, also known as log loss. The generator learns to generate samples that have a low predicted probability of being fake. This means minimizing the probability that the discriminator classifies synthetic images as synthetic.

Other adversarial loss configurations have appeared since then. The Least Squares GAN loss [20] proposes the use of a mean squared error or L2 loss in both generator and discriminator. This loss gives a higher penalty to large errors, which in practice helps to fight the vanishing gradient problem. Wasserstein GAN loss [1] proposes a new loss function based on calculating the distance between synthetic and real images with the Earth-Mover's distance. This loss function, created specifically for the GANs framework, provides useful gradients, allowing for the continued training of the models.

#### Additional Generator Losses

The standalone adversarial training can, in theory, allow the generator to learn the data distribution of the data it has been trained with. Many GANs settings pursue not the generation of new images per se, but to learn a specific transformation. This means that some characteristics of the input image should be kept after passing through the generator, and the adversarial loss can not guarantee that the learned function will map the input to the desired output. To further reduce the space of possible mapping functions, additional loss functions need to be used.

**Cycle-consistency loss**. A way to reduce the mapping function space learnt by the generator network is to ensure that the function is cycle-consistent. This loss was proposed for training with an unpaired dataset [16], but has also shown good results when using it for a paired dataset configuration [33]. This loss proved to be effective in transformations involving colors or textures like style-transfer or photograph enhancement, but performs poorly when the transformation implies big changes in the image. The original cycle-gan was done using two generators, where one learnt a transformation and the second the backward transformation. StarGAN [9] simplified the configuration to a single generator and discriminator, being able to learn both forward and backward transformations with the same generator.

**Reconstruction loss**. Another way to narrow this function space is to minimize the L1 or L2 loss between the generated image and the ground truth. In order to alleviate the channel pollution when using this loss, a channel-wise calculation has shown to be effective [33].

Perceptual loss. This builds further on the idea of the reconstruction loss. Instead of directly encouraging the generated image to match the ground truth, we encourage a match on the feature representations of encoded versions of it. To do that, we create an external CNN network and make a forward pass of both the generated and ground truth image [17]. Specific latent spaces of different layers are selected from the two images and the L1/L2 loss is computed, trying in every interaction to reduce it.

### 3.2.4 Evaluation Metrics in GANs

Different measures have been proposed to evaluate and compare GANs. Yet, there is no consensus as to which measure best captures the strengths and limitations of models and the search for a fair comparison measure is still going on.

From a strictly theoretical point of view, the adversarial training implies that only one of the two networks will finally succeed.

- Generator: Will succeed if the discriminator, acting as a binary classifier, achieves a 50 percent accuracy in classifying real and synthetic images. This is the accuracy of a random guess, and means that the generator has learnt a data distribution that generates images indistinguishable for the discriminator.
- Discriminator: Will succeed if it achieves a 100 percent accuracy, being able to correctly classify between the real and the synthetic images generated by the generator.

The training process, on the other hand, aims to keep the balance between both networks. If one of them achieves its final goal, the other one will not be able to improve, so we tend to keep a balance between them. A very good discriminator will always classify correctly, and the loss function information that the generator receives will not contain valuable insights and the generator will never learn how to improve its images. A very good generator will always fool the discriminator, so it will not be eager to generate better images, and the final results of the image will have bad quality, as the discriminator was easily fooled.

Even if we evaluate from the basis that the training process has been balanced, there will always be a bias factor coming into the equation. Depending on how good the discriminator operates, its ability to distinguish between real and fake images will greatly vary. The learnt data distribution of the generator may produce good enough images to fool the discriminator, but fail to have a photo-realistic results in terms of image quality. This happens because the data distribution aimed to be learnt has very high dimension and there are many optimal solutions that can fool the discriminator, but there is no direct relation between them and a high quality image. This is why the adversarial loss is often combined with other complementary losses that constrain the learnt data distribution. To overcome those uncertainties, two main groups of evaluation metrics [4] are used to evaluate GANs.

- Quantitative measures: Mainly "model agnostic" in that the generator is only used for the image generation. They provide an objective criteria, but sometimes may not correspond to how humans perceive and judge generated images. In conditional GANs, some measures extracted from the image quality assessment literature like Mean square Error (MSE), Peak Signal-to-Noise ratio (PSNR) and Structural Similarity (SSIM) [36] are widely used [39] [33].
- Qualitative measures: Visual examination of generated samples by humans is one of the most common and intuitive methods to rate GANs. Despite its convenience, it also has

several drawbacks. It is an expensive process and the evaluation may be biased depending on the human inspectors used. For the evaluation of conditional GANs, perceptual measures are widely used [16] [35] by doing A/B tests with tools like Amazon Mechanical Turk.

The Mean Square Error, is a pixel-wise measure calculated by averaging the squared difference between a ground truth image and the generated one. As it does not take into consideration the peak of intensity of the pixel values, images with different encoding values cannot be objectively compared using MSE. This drawback is handled by PSNR, that takes into account the maximum possible pixel value (e.g. 255 for an 8 bit representation). Both measures are pixel-wise, which means they do not quantify the image similarity in terms of structure.

$$PSNR = 10 * \log_{10} \frac{MAX^2}{MSE} \tag{3.11}$$

Where MAX is the maximum possible pixel value of the image.

The SSIM measure [36] was proposed to overcome the limitations of both MSE and PSNR. It is based on visible structures in the image and compares not only pixel-wise, but also the pixel neighbours using luminance, contrast and structure quantities. It aims to imitate the human perception, discounting aspects of the image which may not be relevant and a more reliable measure of perceptual similarity.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.12}$$

For every NxN pixels window, where $\mu_{x/y}$ stands for average, $\sigma_{x/y}$ stands for variance, $\sigma_{xy}$ stands for co-variance.

### 3.2.5 Non-convergence: Mode Collapse

Mode Collapse, also known as the Helvetica scenario, is the most common non-convergence problem encountered in GANs training [12]. This occurs when the equilibrium between generator and discriminator breaks, causing one of them outperform the other. In mode collapse, it happens that the generator maps several different input values to the same output point. Although complete mode collapse is rare, a partial mode collapse with outputs containing the same colors, textures or shapes is common.

It can be identified by visual examination of the generated samples or by reviewing the line plots of the model losses. When the discriminator loss improves to values close to zero, the generator is unable to generate real-like images that fool the discriminator. This causes the generator loss to oscillate over time between high and low values.

# Chapter 4

# Related work

In this section we explain several state-of-the-art methods for conditional GANs that will constitute the core of our model components. We also explore the more specific research being done with the use of segmentation masks for human pose manipulation, novel object pose generation, and finally in the introduction of new elements in an image, which is the closest field to our hand-object grasp generation challenge.

## 4.1 Conditional GAN frameworks

Conditional GANs allow to introduce desired behaviour by introducing into the generator what is known as a mask. The mask contains the information that the generator will use to generate the image, it is thus conditioned by it. Masks can be numeric labels, text, and most commonly, images.

### 4.1.1 Low-dimensional images

Conditional GANs rely on a the use of conditioning information to be able to learn a broad range of transformations. Pix2pix [16] stands as a common framework when using images as the condition, which is also known as image-to-image transformation. Its mayor contributions are the use of specific generator and discriminator architectures that demonstrate the best performance when dealing with this kind of problems. It also proposed GAN objective that can be used as a baseline and combined with other more problem-specific losses.

In 4.1, a target mask is introduced into the generator, which uses the information contained in it to generate the desired output. The discriminator then has the task of classifying as real or fake, pairs of images composed by the target images and target mask and generated image and target mask respectively.

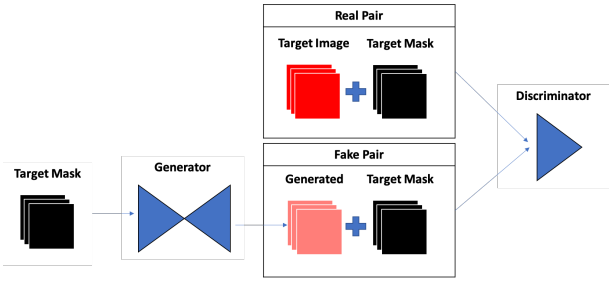



Figure 4.1: Pix2Pix Architecture

If we observe the example of the segmentation mask labels for buildings 4.2, what the generator network is learning is a transformation that will output a photo-realistic building out of the different components in the mask. The same happens with black and white images to color 4.2, the generator is learning the transformation of impainting the image in a realistic way. The idea behind is that despite the GAN is still the same in terms of architecture, the underlying idea behind shifts from image generation to image transformation. We frame the input image of the generator to a set of desired constrains in order to learn a specific transformation. In the next related work papers, we will revise the different novel ways of learning them, and the two most important mechanisms that researchers have to focus on, the loss function and the conditional mask.


Figure 4.2: Pix2Pix examples. Left pair: segmentation mask transformation.Right pair: inpainting transformation

### U-net

The generator architecture presents a "U-Net" shape autoencoder [27] with skipping connections between the encoder and decoder layers 4.3. This information is shared directly without the decoder without suffering the shrinking and consequent loss of information of passing through the encoder layers until arriving to the bottleneck layer. This is especially beneficial in image-to-image translation in which a big amount of low-level information needs to be kept to achieve sharper and more realistic results.

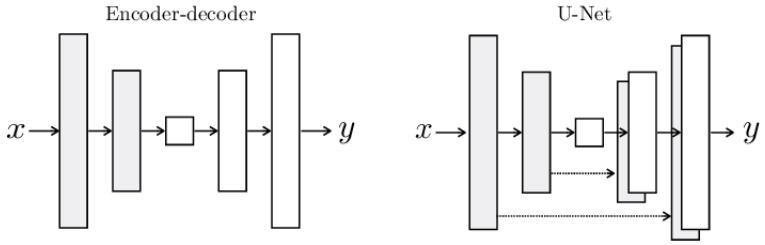



Figure 4.3: The skip-connections of the U-net are one of the key characteristics than distinguish this autoencoder from a conventional one. Image source [16]

The U-net is composed of convolutional blocks with LeakyRelu activation function, in which the number of filters consecutively double that of the previous layer. After the first convolution layer, the following blocks also contain a batch normalization layer that makes the samples output zero mean and unit variance and that has proven effective for stabilizing the training process [15].

### PatchGAN

The PatchGAN discriminator maps the input image into a one dimensional NxN output. Each of the values of that output map to a patch or receptive field of the original input image, and it is classified into real or fake. A receptive field size of 70x70 was proven to be the best trade-off. A smaller receptive field was proven to tiling artifacts, while a bigger one does not offer better visual results [16]. This configuration allows the use of fewer parameters and a faster training.

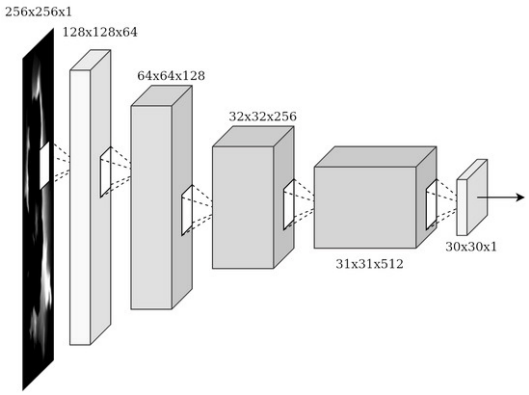



Figure 4.4: PatchGAN discriminator networ. Image source [24]

The discriminator architecture follows the same scheme as the "U-net" generator. In each transpose convolutional layer, the number of filters doubles while the image shape shrinks by a factor of two. Every block has a batch normalization, but the last one before the output layer does not.

### 4.1.2 High-dimensional images

The framework used in Pix2Pix with the U-net generator and PatchGAN discriminator has a limitation in terms of the resolution of generated images. It is intended to work with low resolution images (Eg. 128x128 or 256x256) but fails when dealing with higher resolution images. With Pix2Pix as a baseline, researchers from the same team proposed a solution to improve photorealism and resolution. First, a low-resolution image generation is learnt by the encoder-decoder generator. Once it has learned, we append to both sides a bigger convolutional block and train it as a whole to generate higher resolution. This type of generator, named "Coarse-to-fine generator" can be consequently increased in resolution by appending extra layers until the desired size is acquired.

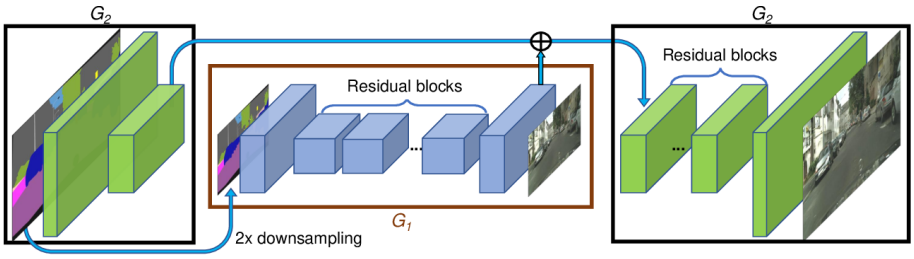



Figure 4.5: Pix2PixHD Generator architecture. Composed by embedded generators which refine and upsample the generated image. Each of them has a discriminator associated that assures a good image quality in every step. Image source [35]

This incremental resolution learning implies a change in the discriminator strategy. As the final image has a high resolution, keeping the PatchGAN previously used with a 70x70 receptive field would require a deeper network or larger convolutional layers. This is not convenient as it can potentially cause overfitting and demands a larger memory footprint for training. Alternatively, the use of several discriminators evaluating the quality of each of the different resolution images generated proves to be a better solution. All discriminators have an identical architecture, and they evaluate different aspects of the final image. The discriminator evaluating the lowest resolution image will have the finest receptive field and takes care of the small details. The discriminators

evaluating higher resolution stages will have a broader receptive field and will evaluate a more global view of the image, assuring global consistency.

## 4.2 Segmentation masks for pose manipulation

In this section we review the works that focus on pose manipulation. We will make a distinction based on the training procedure, which divide them in paired and unpaired training.

In paired training, each source image has a specific target image associated, and requires a dataset of the shape (source image, target image). Once an image is generated out of a source image, it is compared with its paired target image to calculate the losses. Unpaired training, on the other hand, resembles to an "unsupervised" way of training GANs. There is the target set of images that share a specific characteristic, for example Van Gogh painting style, and a source set that do not have that style. The generated image can be compared with any target image, as they all share the Van Gogh style.

### 4.2.1 Paired training

In the last years, several works have proposed solutions based on the use of segmentation masks to learn how to transform target person images to novel poses. The generator receives as input both target image and a conditional mask containing information on the new desired pose, and the outcome is new transformed image.

Pose Guided Person Image Generation (PG2) [19] tackles it in a two-step process in which a first coarse result is refined to keep the target person general appearance 4.6. A segmentation pose mask is created for each image with the use of a pose estimation model [6]. Each training sample is composed of 2 human poses and its pose masks. Image A and mask B go into the generator, and try to reconstruct image B. The generator focuses on learning the pose transformation, but fails to retain finer image details, causing a blurry output. To create a sharper image, a second generator inputs both target image and coarse output and learns a difference map of both of them that is later added to the coarse result.

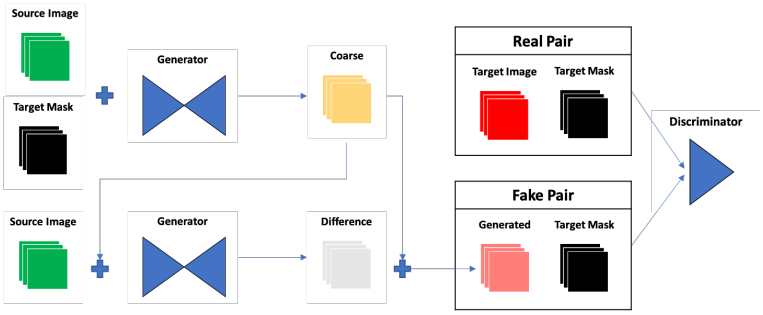



Figure 4.6: PG2 network

Another novelty introduced is the use of a pose mask loss function. This mask loss consists of giving an additional weight in the area of the image of the segmentation mask when calculating the L1 loss. By doing this, the network gets more penalized when generating a big loss in the mask area and leads to better results as shown in the experiment 7.2 carried out.

Synthesizing Images of Humans in Unseen Poses [31] does a similar task as PG2, but with a different approach. The pose mask is generated by the use of a first generator. Then, the background and foreground are generated separately to later join them with the use of the pose mask. In

addition to the adversarial loss, the pose generation makes use of a reconstruction loss 4.1 that is computed using a VGG-19 network, and proves to achieve better generation results.

$$\mathcal{L}_{perceptual}(G) = \|VGG(y) - VGG(G(x, M_y))\|_1 \tag{4.1}$$

where VGG is the VGG-19 network.

This approach is shared by other works like Dance Dance Generation: Motion Transfer for Internet Videos [39]. Specially interesting are the different loss functions used in the pose generation. Up to four different losses are used simultaneously. There is the adversarial loss, then the perceptual loss is used twice, one with a VGG-19 network and another with a pose estimation network. Finally, a feature matching loss [35] loss aims to stabilize the training as the generator has to produce natural statistics at multiple scales.

When there are multiple losses guiding the learning, the weight that you give to each of them becomes extremely important. In their experiments, the best combination found was a factor of one for the adversarial loss, and a factor of ten for the other three. This configuration is interesting in the sense that the importance of the discriminator gets reduced. This goes in consonance with other works [33] that we will analyze next.

GestureGAN [33] proposes a solution to learn how to transform hand poses into novel poses by the use of segmentation masks. It follows the same concept idea as the body pose papers presented above, but with the difference that it is a one-step process.

In order to achieve good results in a one-step process 4.7, the generator makes use of four different losses. A modified version of the reconstruction loss calculated channel-wise (three in the case of an RGB channel) solves the problem of the "channel-pollution" present in PG2 [19], which removes the need for a second step to refine the output image. It also uses the cycle-consistency loss. The adding of those two losses was mentioned to be very important to achieve significant improvement of the model performance. In terms of weight loss, the optimal configuration is in line with DanceDanceGAN [39]. It gives a low weight to the adversarial loss with a value of 1, while giving high weights to the reconstruction loss and cycle-loss with values of 100 and 10 respectively.

Another interesting twist is the discriminator input 4.7. To the previously seen configurations in which the real and the fake input was composed by the target mask and target or generated image, the source image is added, forming a triplet instead of a pair.

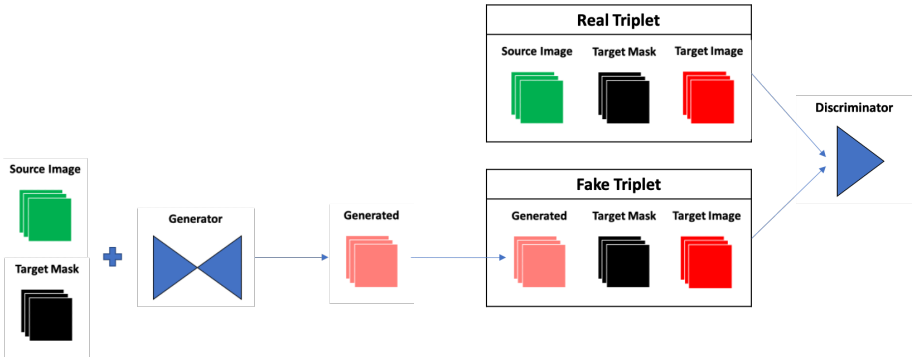



Figure 4.7: GestureGAN Architecture. The discriminator input is composed by triplets of images instead of the pairs seen in the previous works.

### 4.2.2 Unpaired training

One of the limitations when training conditional GANs is the need for paired data to train the model. For example, in the pose transformations mentioned before, a couple of images of the same person in different poses are needed. CycleGAN [15] addresses this limitation by learning transformations without the need of paired dataset, just with the use of unpaired images.

CycleGAN requires two GANs training in parallel. The first generator learns how to go from the source to the target image, and the second generator learns how to go backwards 4.8. In the learning process, the cycle-consistency loss constrains the learnt transformation of the generators to a solution that ensures that when a target image goes through both generators, the original image is reconstructed back.

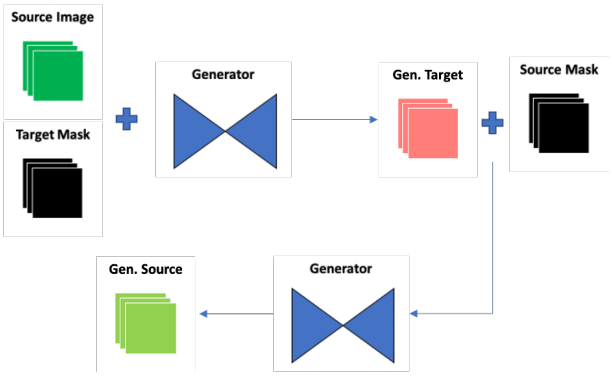



Figure 4.8: Cycle-GAN network. Each generator has a discriminator associated that encourages the network to learn.

This approach achieves great results in transformations that involve color or texture changes, like applying a specific painter style to an image, but performs worse when the task requires geometric changes in the image. In terms of computation time, the need to train two GANs doubles the resources needed. Other proposed versions of unpaired training have demonstrated good performance with the use of a single GAN, in which the generator learns both forward and backward transformation [9]. Also, the cycle-consistency loss proved to be effective when using it with paired datasets [33].

## 4.3 Object pose manipulation

Objects present in 2D images are difficult to manipulate because of the absence of a third dimension. This third dimension gives the depth information of the objects and is key to perform any transformation. Overcoming this limitation is an ongoing field of research that is being addressed from different perspectives.

One of the current lines of research aim to recreate a 3D scene representation out of a 2D object. With this approach, you can rotate the object into a novel view and go back to the 2D image.

- DeepVoxels [30] builds this 3D representation by combining the feature information of the 2D object images with the use of a pinhole camera model. The model achieves a high level of detail, but it requires a training set of 479 2D poses, uniformly distributed.
- 3D-GAN [37]. The training samples are both 2D and 3D samples of different objects in Shapenet [8]. In the 3D-VAE-GAN version of the paper, a 2D image inputs an autoencoder and the feature representation of the object is learned in the encoder to later generate the

3D representation when going through the decoder. This work, standalone, does not solve the rendering back of the 2D image in a novel view, but it does half of the job.

Another line of research is the generation of novel viewpoints of objects going directly from 2D to 2D. GANs are often used. Although is currently limited to object rotations in different angles, only a single input image of the object is required.

- VariGAN [38] generates multi-view images of a target image of a person. It uses a conditional setting in which different positions are learned by the use of different word labels like "side" or "back". A coarse image is generated, which is later refined by a second network.
- IterGAN [11] uses an iterative approach in which multiple intermediate objects are generated, each of them with a small rotation angle until the target angle is achieved. In each interaction, an intermediate discriminator loss is introduced, guiding the rotation and ensuring consistency throughout the process.

## 4.4 Introducing new elements in the image

The conditional GAN models analyzed so far all have something in common. They learn to apply a transformation that modifies the already existing elements of the image. In other words, the transformations do not introduce any object which was not present beforehand. Introducing a new element in an existing image is a challenging task that is currently taking its first steps with the use of GANs. Among the challenges faced when dealing with this technique, we can cite:

- Providing the spatial information to generate the object in a specific place and rotation angle of the target image.
- Understanding the spatial 3D depth of a 2D image to be able to generate the object in a context-aware way.

During the last months, two papers have been published which try to answer these challenges.

'Object-Centric Image Generation from Layouts' [32] relies on the use of rectangular layouts with its correspondent image class. Previous works have used layouts that include pixel-level information about the different classes to generate, but failed when several layouts overlap each other [2] [31]. To overcome this limitation, the work proposes to use scene-graphs that depict positional relationships from the given spatial layouts 4.9, leveraging them to learn relationships between objects.

With this concept, they create a new loss function that aims to increase the layout fidelity of the generated images. They build a module that extracts the scene-graphs from the layouts, and then encode it to obtain the global graph features. Those features are compared with the generated image after going through an encoder to get the same dimension as the graph features.

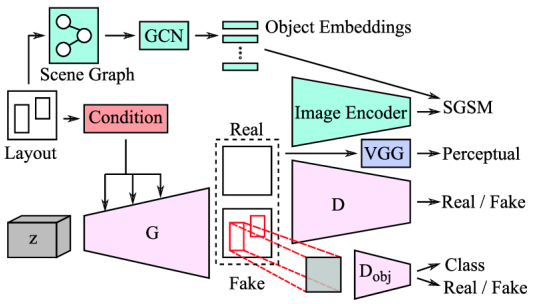



Figure 4.9: Object-centric Network. Image source [32]

The second key point is how the GAN learns what the different layout object classes must look like. In this case, the data distribution of each of the classes is learned by the generator. To evaluate the quality of the different generated objects, the paper uses an auxiliary classifier as in the AC-GAN framework [23]. This configuration makes use of two discriminators. One evaluates individually the different objects generated in their layout rectangles, while the second one evaluates the overall image.

The second paper, 'Wish You Were Here: Context-Aware Human Generation' [10] addresses the challenge of inserting humans into existing images in which other people are present. The idea is to create a segmentation mask in which different colors represent different body parts, and to place that segmentation mask in a way that is context-aware of the other people of the image.

This context-aware segmentation mask is generated in a first phase, they named 'Essence Generation Network'. It is an encoder-decoder network in which the segmentation masks of all the people of an image are present except one, which has to be generated as an output.

Once the desired segmentation mask is added, it is combined with the target person image to generate that person in the new pose. As input of the second network, the different body parts of the target person are introduced (face, hair, torso), extracted with a segmentation model. This information gets compressed through the encoder, and in the following decoder layers, the desired segmentation mask pose is added to the input. This procedure allows the network to learn the mask in tandem with the person rendering, avoiding the "pasting" effect that occurred in previous works [3]. The third network refines the coarse results for the faces to make them look more human-like. A key aspect is a perceptual loss to make the face generation closer to the feature layers obtained when going through a VGG network specially pre-trained on a face dataset [5].

## 4.5 Limitations of the state of the art

We have reviewed the most relevant papers in relation to our research direction. Different generator and discriminator architectures for conditional GANs have become a good starting point with pix2pix as a good solution for low-resolution image generation, while pix2pixHD addresses the need for higher-resolution results. We have also analyzed the different segmentation masks that are unique for each kind of transformation. Also, a review of the importance of the different loss functions and the novel ways to engineer the learning process of GANS has been done. Finally, we have dived into the work related to 3D manipulations of 2D objects and the introduction of new elements in a coherent way into 2D images.

We can now take make conclusions about the viability of the proposed framework, which, as explained in the problem statement, is constrained by the specific inputs of our model. It requires the hand and the object in a spatial position that allows the grasp.

- Hand pose manipulation. Most of the ongoing research in conditional GANs pivots around the idea of transforming existing elements present in the target image. Transforming the hand pose in order to get a specific desired pose has been recently addressed by GestureGAN. It uses a hand pose mask in order to generate that pose in the wild. Hence, the research questions related to the first challenge have already been solved by the community.
- Object pose manipulation. This is an active research challenge, which is already being addressed in two main ways. The first is to recreate 3D representations out of 2D objects, to later re-render them with a novel view. The DeepVoxels approach requires a high number of 2D image samples, and the research question of "How to learn a 3D representation out of a few samples of 2D images?" has not been answered yet. The 3D-GAN approach, even though it uses few samples, results in far from photorealistic images, and only solves half of

the problem (2D to 3D). The second one is to use GANs to render novel views directly 2D to 2D. Only object rotations [38] and people rotations [11] have been addressed.

Finally, we can also appreciate that the hand-object grasp generation is a challenge that has not been tackled yet. The closest work is "Wish you were here GAN" [10], but differs in the sense that the humans generated did not have occluded areas or close interaction with other persons in the image. There is also ongoing research on how to introduce new elements into an image [32], which is in line with the hand-object grasp challenge, as the introduced objects also need to be coherent with the other image components.

The existence of similar lines of study like Conditional GANs with target masks and how to introduce new elements into an image will give us the basis on which we will build our research.

# Chapter 5

# Methodology

As explained in 2, we want to create a model capable of learning how to generate hand-object grasps. The model will receive as inputs the hand together with a conditional mask with the information about the object. With the use of different objective losses, the model will aim to learn the grasp.

In the following section, we discuss the chosen model architecture in the field of conditional GANs. We also discuss the two main design elements that will characterize our model and constitute the core of a conditional GAN, i.e., the conditional mask and the loss function.

## 5.1 Architecture

The architecture used for this study is inspired by the Pix2Pix framework, and motivated both by the size of the dataset images most widely used in the GANs literature, together with the state-of-the-art in conditional GANs.

As seen in 4.1.1, the images with size 256x256 or smaller fall under the realm of low-dimensional images. This image size is used in the Pix2Pix architecture in which the transformation does not require several enhancement steps as in the high-dimensional Pix2PixHD architecture. These image sizes require of lower computational expenses, and are often used for research purposes. Also, several recent papers have used the Pix2Pix architecture in the last year [7] [33], proving that it is still considered as the reference for low-resolution conditional GANs.

### 5.1.1 Generator

The generator is a "U-net" shape encoder-decoder architecture (See 4.1.1). It inputs both RGB images of the hand and mask of the object, that are concatenated together at a channel level, making six 256x256 channels. The input shrinks through the encoder with convolutional operations until it arrives at a bottleneck layer. Each of the layers has skipped connections that connect directly to the equivalent decoder layer. The decoder part is equivalent to the encoder but does transpose convolutional operations to upsample the bottleneck layer and combines it with the information coming from the skip connections of the encoder. The final output is the generated three-channel 256x256 RGB image.

### 5.1.2 Discriminator

The discriminator used is a PatchGAN discriminator (See 4.1.1) with a receptive field size of 70x70, which proved to be the best trade-off for 256x256 images. It inputs a concatenation of two RGB images and outputs a one-dimensional 30x30 matrix in which each value represents a 70x70 patch of the input image concatenation.

### 5.1.3 Auxiliary classification network

We use an auxiliary VGG19 classification network trained on Imagenet [28]. Five feature representations of intermediate layers of the VGG19 network are extracted for every input image entering the network, and used afterwards to calculate the perceptual loss 5.3, which does not intervene in the adversarial training.

More precisely, the slices are feature representations of sizes 256, 128, 64, 32 and 16 of the original input image, which correspond to the 2nd, 7th, 12th, 21st, and 30th layer of the VGG-19 Torchvision implementation that follows the original implementation [29].

The overall network architecture proposed for this work, and that will be used to perform the upcoming experiments, can be represented as:

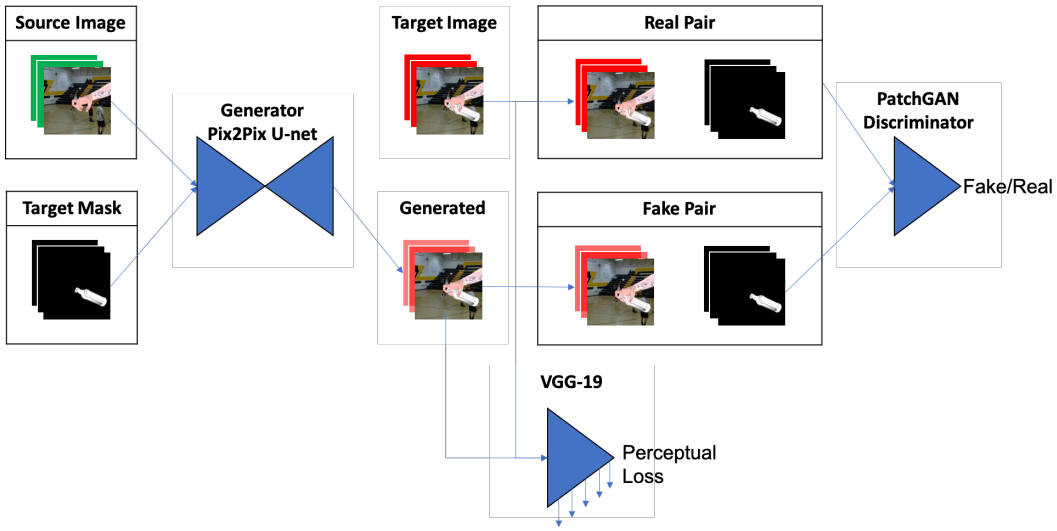



Figure 5.1: Overall Network Architecture

## 5.2 Conditional Mask

The conditional mask used for our model needs to provide information tailored to the task we are trying to solve. The input image is the hand without the object, and the output must be the same image with the hand grasping the object. We are aiming to learn the hand-object grasp generation, and this implies learning the shadowed parts of the hand and the object during the grasp.

As seen in previous works, the conditional masks are often monochromatic, especially when dealing with inferring a new pose. The source image they aimed to modify was already present in the image, thus did not need extra information about its appearance. This changes in our work, as we introduce a new object in the scene that is not present in the source and the model does not know about.

The mask to be used in our methodology is an RGB mask of the object on a black background, placed in the correct spatial position for the grasp 5.2. This configuration expects the model to learn to place the object in conjunction with the hand, learning the parts that both hand and object shadow between each other.

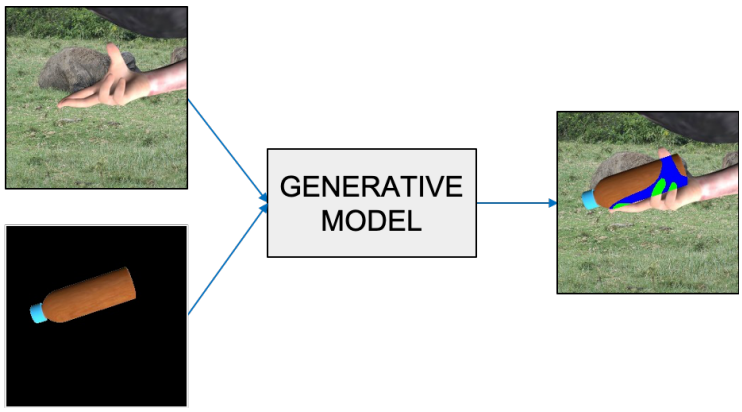



Figure 5.2: The model conditional mask is an RGB image of the object to grasp. The model is expected to learn to place in the foreground the object parts that shadow the hand (blue) and object parts shadowed by the hand (green)).

With a monochromatic mask of the object, the model would fail to render the real object appearance. This happens because the mask specifies the spatial position and shape of the object, and even though the model sees some training samples of the target object it is not able to learn the object pose manipulation. If we also provide information about the object parts to be shadowed by the object, we expect the image generation results to be better. In other words, the more information about the grasp is given by the mask, the less has to be learned by the model and thus the better performs.

Figure 5.3: Examples of two monochromatic masks that fail to learn the hand-object grasp.

## 5.3 Loss Function

To ease the understanding of the upcoming formulas, the naming convention uses 'G' and 'D' for generator and discriminator, 'x' for the source image that inputs the generator, 'M' for the conditional mask, and 'y' for the target image we aim to generate.

In every GAN framework, there is an adversarial loss that the generator network uses to try to fool the discriminator. With the chosen architecture, the discriminator is not just a single binary classifier, but a patch classifier that classifies each of the patches as fake or real, restricting the attention to the structure of local image patches. A BCE loss is used for each of the patches and averaged.

The duplets that the discriminator has to classify as real or fake are composed by concatenation of the generator target mask, and respectively the target image and generated image:

$$realDuplet = (TargetMask, TargetImage) \tag{5.1}$$

$$fakeDuplet = (TargetMask, G(x, M_y)) \tag{5.2}$$

Hence, the adversarial loss can be represented as:

$$\mathcal{L}_{adv}(D) = \mathcal{L}_{BCE}(realDuplet, 1) + (fakeDuplet, 0) \tag{5.3}$$

where BCE is the binary cross-entropy loss.

In the generator, instead of maximising the discriminator loss, we try to minimize the BCE error of a fake duplet being classified as true, which is the opposite of what the discriminator is trying to do.

$$\mathcal{L}_{adv}(G) = \mathcal{L}_{BCE}(fakeDuplet, 1) \tag{5.4}$$

In this research, the input image closely resembles a lot to the ground truth. As explained before, apart from the object area, the rest should remain unchanged. The generator shrinks the image, and despite the skip connections between the encoder and decoder, we need to train the GAN in a way that is encouraged to learn to reconstruct those areas that should not vary. This is done by adding extra losses in the generator network. Those losses aim to reduce the distance between the ground-truth and generated image.

To achieve this, the first loss we use is the L1 loss proposed in the Pix2Pix framework. With the high-frequency correctness modeled by the PatchGAN, the use of L1 loss is enough to model the low frequencies and L2 loss was not added because it generated blurry results. This blurriness has been solved by calculating the L2 loss channel-wise [33] and so we also include the L2 loss in our work. To give an added weight to the loss in the area containing the object, we use a mask loss [19] that doubles the L1 and the L2 loss for the object.

$$\mathcal{L}_{l1,l2}(G) = \|(y - G(x, M_y)) * (1 + M_y)\|_{1,2} \tag{5.5}$$

where $\|_{1,2}$ is the L1 and L2 loss, and $M_y$ is the target mask.

The perceptual loss has proven to be effective in similar conditional GAN works [35] [39] [33]. We use an auxiliary VGG19 network and we reduce the loss between feature maps of different layers between the ground truth and generated image. The use of this loss stabilizes the training process as the generator is encouraged to produce natural statistics at multiple scales.

$$\mathcal{L}_{perceptual}(G) = \|VGG(y) - VGG(G(x, M_y))\|_1 \tag{5.6}$$

where VGG is the VGG-19 network.

Finally, we also experiment with the use of the cycle-consistency loss. Our interest on this loss resides in two factors. First, it acts as an additional constraint to force the learned transformation to be cycle-consistent. Second, the backward transformation implies going from an image with an object to the same without the object, that could be an interesting approach for afterwards introducing a new object different from the one that was being grasped in the ground truth image. We use a a configuration with a single generator and discriminator [9].

$$\mathcal{L}_{cycle}(G) = \|x - G(G(x, M_y), M_x)\|_1 \tag{5.7}$$

where $M_x$ is the source mask.

This four losses will be controlling the training process of the network. Four lambda hyper-parameters will set the weight of each of the losses in the generator. The final generator loss goes as follows:

$$\mathcal{L}(G) = \lambda_{adv} * \mathcal{L}_{adv}(D) + \lambda_{l1,l2} * \mathcal{L}_{l1,l2}(D) + \lambda_{perceptual} * \mathcal{L}_{perceptual}(D) + \lambda_{cycle} * \mathcal{L}_{cycle}(D) \tag{5.8}$$

where $\lambda$ is the loss weight.

# Chapter 6

# Experimental setup

In this section we will introduce the dataset used in this study, discuss the different evaluation metrics used to measure the quality of our model, and expose the baselines to compare the performance of our design decisions.

## 6.1 Dataset

The datasets available to evaluate our method are constrained by several requirements that makes them hard to find. Firstly, they need to be having a lot of hands holding different objects: this images constitutes the target image that we want to obtain. Secondly, for each ground truth, we need paired images that do not contain the object held by the ground truth. What is more, that image without the object needs to have the hand in the same grasp position as the ground truth. This is because what our model learns is the interaction between hand and object to generate the grasp, but does not perform hand or pose manipulation that are part of the open research fields explained in 4.5.

The only dataset that we could find that fits these requirements is the Object Manipulation (ObMan) dataset. ObMan is a large-scale synthetic image dataset of hands grasping objects. This dataset was generated in 2019 for use the paper Learning joint reconstruction of hands and manipulated objects [14] that learns to estimate a 3D pose of a hand and an object being held in a 2D image. The good results obtained when using the model trained on the ObMan dataset with real images demonstrate the transferability of ObMan-trained models, validating the quality and usefulness of the proposed dataset. The main elements used in the image generation are:

- Objects: Eight kinds of object categories (bottles, bowls, cans, jars, knives, cell phones, cameras and remote controls) which add up a total of 2772 different individual object models that compose the ShapeNet [8] dataset.
- Grasps: The object grasps were generated using GraspIt [21], a grasp simulation and visualization tool that allows us to create and analyze grasps of a given 3D model with a given articulated hand model. The hand model used a modified version of MANO [26] to compose a set of 16 rigid parts, 3 parts for the phalanges of each finger and one for the hand palm.
- Body Pose: The hand model is integrated with the full body model SMLP [18] to render realistic images of embodied hands.
- Textures: The body textures were obtained from the body scans used in SURREAL [34]. Seventeen hand textures were generated by combining body textures and matched to the closest skin color body model, due to the numerous missing values that the scans had in that area.

The dataset is composed of two-hundred thousand images. The images are in a 256x256 RGB format. Each image comes in three different compositions, the complete image with the object and the hand and the hand and object standalone 7.10. Each image also comes with a metadata file with information about the object and hand depth for 3D representation, hand joints and vertices in 2D and 3D, hand poses, object references from ShapeNet and hand grasp quality from GraspIt among other information. All this metadata allows to visualize interesting information that will be useful for engineering the mask that our solution will implement.

(a) Image sample with object (b) Image sample without object (c) Image sample without hand

Figure 6.1: Each sample comes in three different compositions.

### Data Preparation

The difference in shape of the different objects in the dataset implies different hand grasps. Also, the big size of the dataset with two hundred thousand images requires significant computational resources very big. Because of this, for our experiments we have decided to use a subset of the ObMan dataset containing 350 different objects contained within the bottle category. These constitute a total of 33.278 images that we divide in train, validation and test set by creating subsets split in a stratified fashion, meaning that we keep the distribution of classes between the subsets balanced. We will use the training set to train the model and the validation set evaluate our experiments.

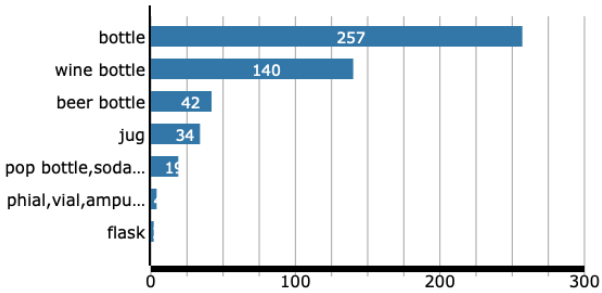



Figure 6.2: Bottle sub-categories.

To prepare the conditional mask, we make use of the metadata of the dataset to turn to black every pixel of the object images in which the object is not present. We will do this for both the source image (hand without object) and target image (hand with object).

The training process requires the dataset to be paired. We thus need to prepare the data to

generate cuatriplets of the shape 256x1024 including the source image, target image, source mask, target mask. These samples will later be decomposed into four 256x256 images that the model will use to compose the inputs to the generator and discriminator, as well as the loss inputs.

## 6.2 Model Measures

In order to assess the effectiveness of our model, we need measures that capture the quality of our results in the most objective way possible. In order to do that, and in line with previous works in the field, we will use two measures. First, a qualitative measure by visually inspecting the generated image. Second, the SSIM in the hand-object section of the image.

### 6.2.1 Quantitative: SSIM

We use the structural similarity as a quantitative measure for our model. We perform it between the ground truth image with the object, and the generated image from the generator output. It provides a value between zero and one, one being the value resulting of inputting two identical images.

Since all the generation changes occur in the area of the image of the object mask and the rest stays invariable, the SSIM measure already achieves a very high value close to one in the early stages of the training, when the grasp learning has not been learned yet. To make the measure more significant, we calculate the SSIM only in a cropped square of the image containing the object, allowing it to evaluate only the area where the changes are happening.

When calculating the SSIM, we create a difference map between the two images that reflects the biggest dissimilarities between both images. The bigger the dissimilarity, the brighter the color of the difference map gets. This is done channel wise, meaning that black represents a perfect match, and white the biggest dissimilarity. This new image can be also used as a reference to qualitatively measure the GAN learning through the training.

### 6.2.2 Qualitative: Visual Validation

Visual validation is a common technique to address the quality of image generation in GANs. It is usually addressed by the use of crowdsourcing marketplaces like Amazon Mechanical Turk in which the generated images are shown to a group of people that try to distinguish the generated from the real ones. In this work, we will do the visual validation ourselves, trying to give an objective analysis. For that purpose, we use of generated images, together with the difference map created when calculating the SSIM.

## 6.3 Baseline Methods

### 6.3.1 Loss baseline

To evaluate the performance of the objective loss of our model, we use the Pix2Pix loss proposed in the paper [16] as a baseline. The generator loss in composed by an adversarial loss together with an L1 loss amplified by a lambda parameter of value 100.

$$\mathcal{L}_{b\_pix2pix}(G) = \lambda_{adv} * \mathcal{L}_{adv}(D) + 100 * \mathcal{L}_{L1}(D) \tag{6.1}$$

### 6.3.2 Mask baseline

To evaluate the performance of the proposed mask, we compare its behaviour against the two monochromatic masks explained in the methodology section. For the monochromatic object mask 6.3a, we aim to certify the hypothesis that the model will lack the information needed to render the object appearance. For the monochromatic object and hand mask 6.3b, to certify that hand mask provides too much information of the grasp and thus the model does not learn the grasp but just extracts the information from the mask.

(a) Object on foreground

(b) Hand on foreground

Figure 6.3: Weak baselines for hand-object grasp generation. White for correct rendering, green and blue for incorrect rendering

# Chapter 7

# Experiments and results

In this section, we will explain the experiments performed in this research and the results obtained using both quantitative and qualitative measures exposed in 6.2.

In the first experiment, we evaluate the performance of the proposed object mask with respect with the baseline masks and validate the hypothesis that made us decline the use of monochromatic masks (See 6.3.2). In the second experiment, we evaluate the performance of our loss design choices with respect to the Pix2Pix baseline and find the best loss weight combination. Quantitatively, GO!GAN final objective loss outperforms a 10% the baseline, and qualitatively generates more realistic grasps with less noise and artifacts. Finally, we expose some failure grasps that GO!GAN model fails to generate.

## 7.1 Experiment 1. RGB object conditional mask

As seen throughout the report, choosing the conditional mask constitutes a key design choice for the success of the model. In this first experiment, we use the Pix2Pix model set up to evaluate the performance of the chosen mask against two weak baselines and two monochromatic masks that we have hypothesized that would not learn as expected.

Qualitative results show that with the use of a monochromatic object mask (mask 1), the model fails to learn the object rendering 7.1. This can be appreciated in the generated image in which the object appearance has the same white appearance as the object mask, suggesting that the model does not have enough information about how the object looks like and thus fail to render it. This goes in line with the previously analyzed work about object pose manipulation, in which models that accurately render an object from a novel view in 2D require of many samples of the object. The training dataset used for our model is composed of 26622 of 350 objects, that means that on average the model only sees each object 76 times from different angle sizes which is not enough for the model to learn the object pose view out of a monochromatic mask. Additionally, the grasp is not properly learned, as the fingers that should be shadowed by the object are not.

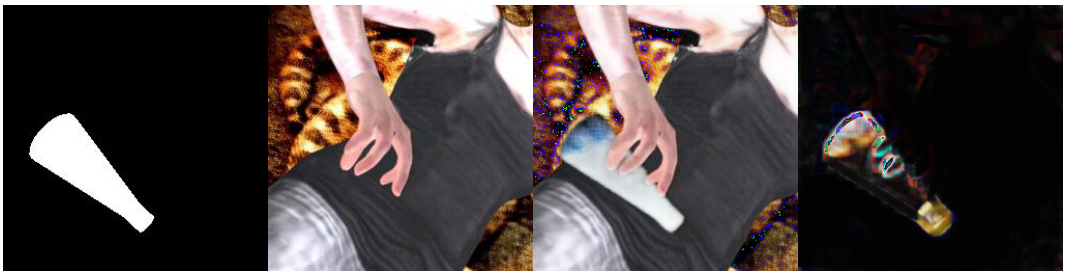


Figure 7.1: Mask 1. From left to right: object mask, source image, generated image, SSIM difference map. The network does not know the object appearance.

The use of a monochromatic object and hand mask 7.2 (mask 2) provides the model more information needed to know the hand-object grasp, more specifically the hand parts that shadow the object, and it can be seen that it succeeds from the early stages of the training 7.2. The fact that from the beginning the grasp is known proves that it is indeed not learning it, just extracting the information from the conditional mask and failing the learning purpose of the model. The object appearance is also not learned successfully, although it does perform better than in the first mask.

Figure 7.2: Mask 2. From left to right: object mask, source image, generated image, SSIM difference map. The network knows the hand-object grasp that is provided by the mask.

Finally, observe that the use of the chosen RGB object mask (mask 3) successfully learns the hand-object grasp. In the early steps of training 7.3, the model places the object in the foreground, but fails to render the fingers that should shadow the object. After the training 7.4, we can see how the hand fingers become visible, proving that the model is learning the grasp. The SSIM difference map supports the evolution of the learning, getting less bright in the fingers area as the model learns how to place the object in a coherent hand grasp. The fact that the fingers are still visible in the difference map after the training suggests that although the grasp in the fingers area has improved, it does not match the original one completely. This can be due insufficient training time, as experiments have been run for a maximum of twenty-four hours.

Figure 7.3: Mask 3. From left to right: object mask, source image, generated image, SSIM difference map. In an early training stage the object mask draws the object on the image foreground covering the hand fingers.

Figure 7.4: Mask 3. From left to right: object mask, source image, generated image, SSIM difference map. After the training, object mask draws the object being aware of the hand grasp, keeping the fingers in the foreground.

From a qualitative point of view, we measure the cropped SSIM value for the three monochromatic masks.

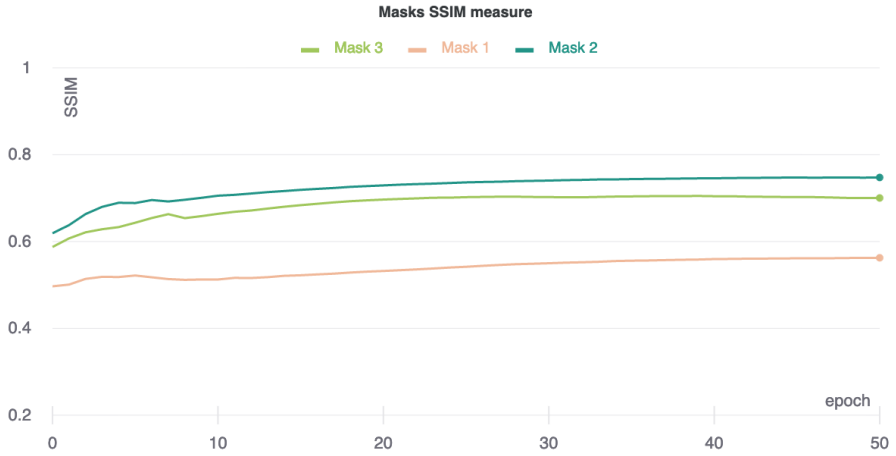



Figure 7.5: SSIM Measure comparison over the three Segmentation Masks.

Mask 1 scores a 18,5 % and 13,78 % less SSIM than the mask 2 and 3 respectively. This is in line with the previous qualitative conclusions. Because this mask does not contain hand, nor object information, the SSIM results are the lowest of the three. The second and third mask scores have a similar value range, with the experiment to a 4,72 % ahead of the third. With this, we can conclude that the SSIM gives a higher importance to the hand-grasp rather than to the object appearance. As the second mask gives the hand information, the grasp is done properly.

## 7.2 Experiment 2. Evaluate the performance of the proposed model loss function

In this experiment we evaluate the performance of our generator loss choices against the Pix2pix baseline. We measure the SSIM with the chosen conditional mask and with the Pix2Pix loss baseline (7.6 - Mask 3), and compare it with the addition of each of the new losses proposed in our model, these are the channel-wise L1 and L2 loss penalizing double on the hand-object grasp area (7.6 - L1-L2) the perceptual VGG loss (7.6 - perceptual-loss), and the cycle-consistency loss (7.6 - cycle-loss). Additionally, we add the best loss weight combination found by adjusting the weight parameters (7.6 - best-model).

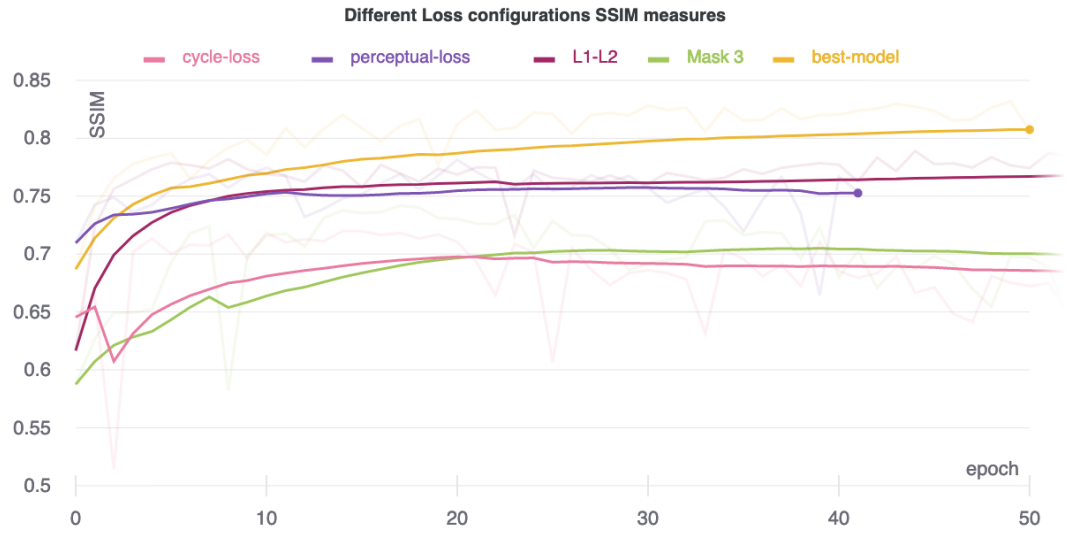



Figure 7.6: SSIM Measure comparison with the use of the different generator losses proposed.

The use of the L1 and L2 loss penalizing double on the hand-object grasp area (7.6 - L1-L2) obtains a 7.6% improvement over the Pix2Pix baseline, being the best performing loss of the three additions. The use of the perceptual loss improves the results on a 4.8% (7.6 - perceptual-loss). On the other hand, the use of the cycle-consystency loss (7.6 - cycle-loss) does not obtain a significant improvement, even with a small decrease over the Pix2Pix baseline (7.6 - Mask 3). The use of these additional losses help to stabilize the training, avoiding the mode collapse (See Appendix A).

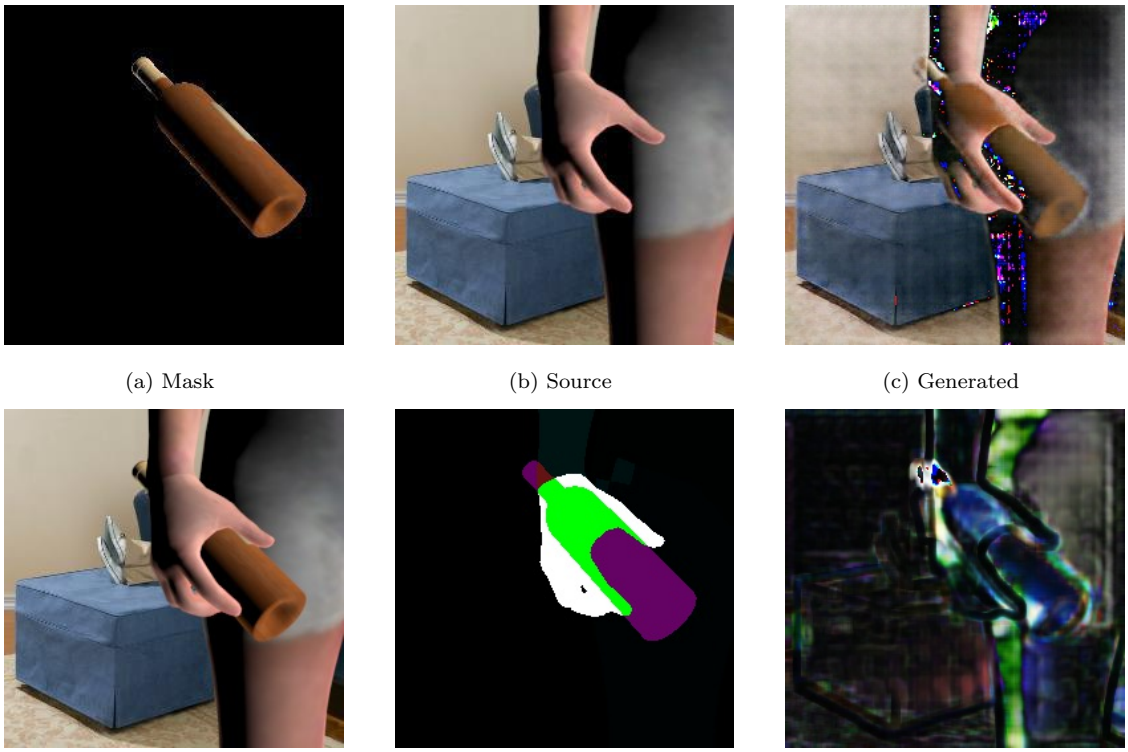
Finally, we search for the best weight loss combination trying different value combinations of $\lambda$ until finding the one that gets the best SSIM. The best runs do not contain the cycle-consistency loss, so our final loss weight selections eliminates it. The final loss objective with the best $\lambda$ values results outperforms the baseline by a 10%, obtaining a 0.8 SSIM score:

$$\mathcal{L}(G) = \mathcal{L}_{adv}(D) + 100 * \mathcal{L}_{l1,l2}(D) + 10 * \mathcal{L}_{perceptual}(D) \tag{7.1}$$

### 7.2.1 Pix2Pix baseline examples

The Pix2Pix baseline model achieves a SSIM of 0.7. The use of an RGB object mask is able to learn the grasp, as exposed in 7.1, but the qualitative results are still blurry and far from optimal. In this section we will analyze two Pix2Pix grasp generations to prove the limitations of this baseline.

In the first example, the model source image 7.7a and the conditional mask 7.7b input the model to generate the target grasp 7.7d. The generated image learns the grasp 7.7c, but the quality of the results are low, and some parts of the object that should be shadowed by the hand (green) 7.7e are not. Also, the generated image contains a high amount of noise, that result in artifacts and blurriness. These two problems can be seen in the SSIM difference mask 7.7f.

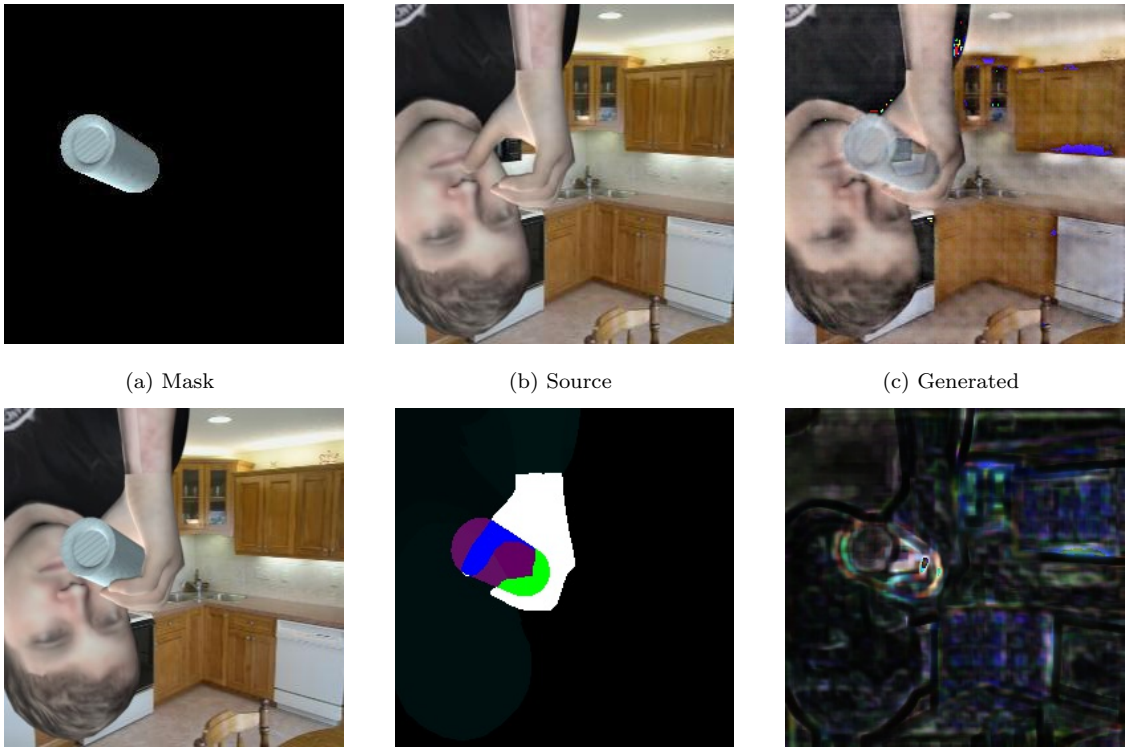


(a) Mask (b) Source (c) Generated

(d) Target (e) Grasp parts (f) SSIM diff

Figure 7.7: Pix2Pix grasp generation. Example 1.

In the second example, the model source image 7.8a and the conditional mask 7.8b input the model to generate the target grasp 7.8d. The generated image partially learns the grasp 7.8c. The object parts that shadow the hand 7.8e (blue) are rendered correctly, but fails in the hand parts that shadow the object 7.8e (green). Again, the generated image contains a high amount of noise, that result in artifacts and blurriness 7.8f.



(a) Mask (b) Source (c) Generated

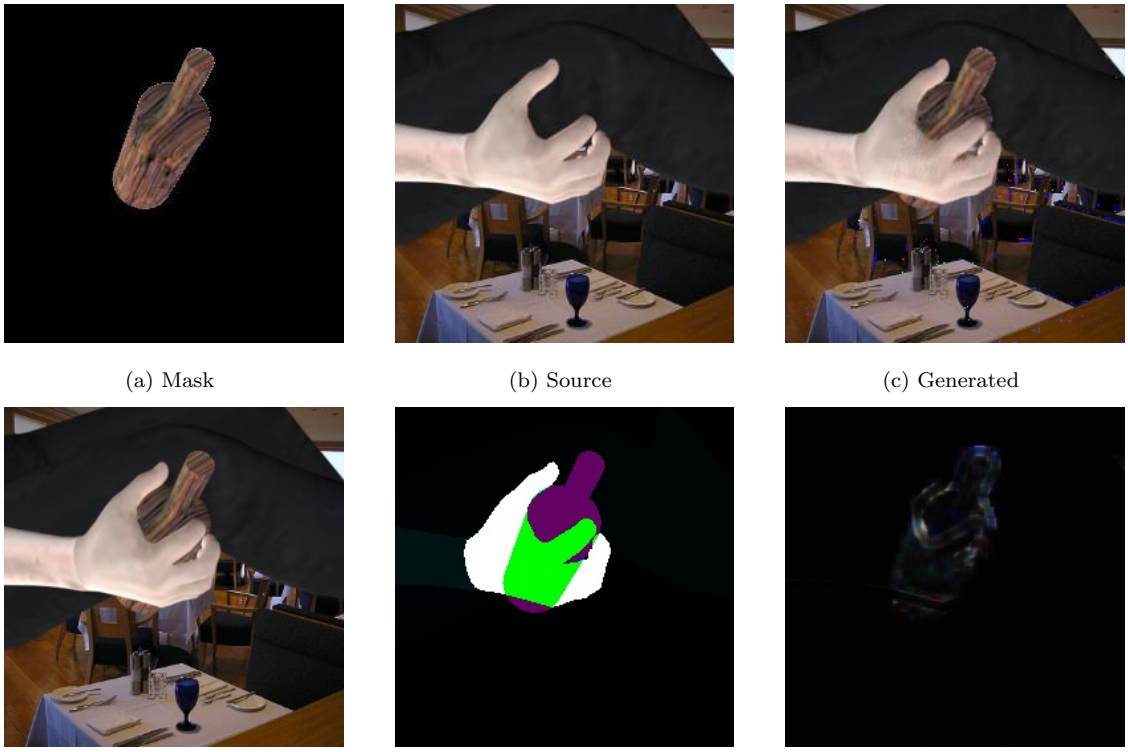
(d) Target (e) Grasp parts (f) SSIM diff

Figure 7.8: Pix2Pix grasp generation. Example 2.

### 7.2.2 GO!GAN model examples

The GO!GAN model achieves a 10% better SSIM than the Pix2Pix baseline, getting a SSIM of 0.8. The qualitative improvements are also significant. Blurriness and artifact reduction can be observed, together with the generation of better grasps in the areas in which hand and object shadow to each other. We will analyze in detail the image generation of two grasps (See Appendix C for additional examples)

In the first example, the model source image 7.9a and the conditional mask 7.9b input the model to generate the target grasp 7.9d. The model succeeds in learning the grasp, and generates an image 7.9c that renders the hand on the foreground when the object 7.9a occupies the same spatial coordinates as the hand of the source image 7.9b, thus learning the hand parts that shadow the object (green) 7.9e. The SSIM difference map 7.9f between the generated and target image shows a slight difference in the edge between the hand and the object that can be appreciated in the 7.9c image in we look closely.

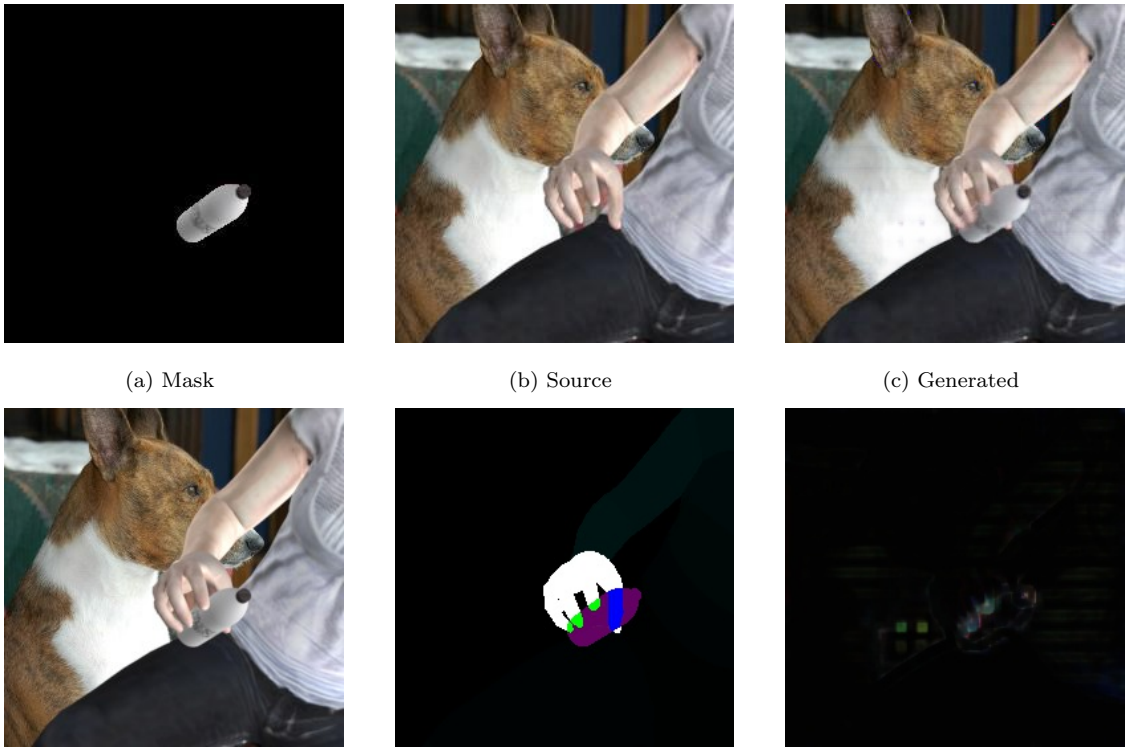


(a) Mask (b) Source (c) Generated

(d) Target (e) Grasp parts (f) SSIM diff

Figure 7.9: GO!GAN grasp generation. Example 1

In the second example, the model source image 7.10a and the conditional mask 7.10b input the model to generate the target grasp 7.10d. The model succeeds to generate an image 7.10c that renders the correct grasp, showing the hand in the foreground when the hand fingers have to shadow the object 7.10e (green), and showing the object in the foreground when the object has to shadow the object 7.10e (blue). The SSIM difference map 7.10f between the generated and target image shows a difference in the fingers that should shadow the object, that can be appreciated in the generated image 7.10c as they are a bit blurry.



(a) Mask (b) Source (c) Generated

(d) Target (e) Grasp parts (f) SSIM diff

Figure 7.10: GO!GAN grasp generation. Example 2.

### Failure cases

The hand-object grasp generation is not always learned properly. There are cases in which a correct grasp comes together with shiny artifacts in the fingers 7.11. Others in which one of the grasp areas is not properly learned. Specially challenging are the areas in which hand and object shadow to each other. In 7.12 the fingers that should be shadowed by the object are not. In 7.13, the object should be shadowed by the hand and it is not.

Figure 7.11: From left to right: object mask, source image, generated image, SSIM difference map. Grasp artifacts in the hand parts that shadow the object.

Figure 7.12: From left to right: object mask, source image, generated image, SSIM difference map. The hand parts that should be shadowed by the object are not.

Figure 7.13: From left to right: object mask, source image, generated image, SSIM difference map. Grasp failure, keeping the object in the foreground.

# Chapter 8

# Conclusions

In this research, we present a personalized conditional modeling framework which is able to learn hand-object grasp generation in 2D images. The framework has been customized to focus on the learning of the interaction between hand and object, which in a 2D environment implies the learning of the parts of the hand and object that shadow each other. Under this context, our model receives two inputs, the source hand and a conditional RGB mask of the object to be grasped, and generates an output image with the hand grasping the aforementioned object.

We focus this research on the use of Generative Adversarial Networks, specifically using the Pix2Pix architecture [40] as a starting point. The reasons why this architecture suits our purpose were thoroughly discussed and each of the components of the network generator and discriminator were explained in detail. In our implementation, we have focused on the design of a conditional mask that better suits the learning purpose of the model, and the modified the loss function of the generator in order to achieve better results.

In addition, we studied and explored the different evaluation metrics that better suit our model; analyzing the obtained results from both qualitative and quantitative results. Furthermore, we have compared our results against challenging baselines for both mask and loss function choices. In the experiment section, we have first validated our hypothesis of using an RGB object mask for targeting the learning towards the hand-object grasp, rejecting the use of a monochromatic mask as seen in previous literature works. We have also validated the use of each of the generator losses added to the model, and found the best weight combination in terms of quantitative results, outperforming our baseline model on a 10%. Finally, we have also reviewed qualitatively the results generated by our model, identifying its strengths and also acknowledging its weaknesses.

The hand-object grasp generation requires both hand and object to be in the same spatial position before inputting the model. Therefore, a future line of research could be how to address these initial starting conditions. A good starting point could be to use already existing images with hand grasps together with segmentation models like [14] to get a target hand and object pose to be used in our source image. This line of research has to come inevitably in hand with improvements on the research of hand and object pose manipulation, being the second one the one that still have bigger challenges to face.

A second future line of research is the extension of the presented model into different hand-object grasps categories. We have validated its use in the bottles category, but the existence of an infinite number of object shapes presents difficult challenges to face. Moreover, we have validated the effectiveness of the model on a synthetic dataset due to the lack of real image datasets meeting the initial input requirements. It would be interesting to test the way the model can generalize to real image datasets. That requires the creation of new datasets for it.

# Bibliography


[1] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *ArXiv*, abs/1701.07875, 2017. 9

[2] Oron Ashual and L. Wolf. Specifying object attributes and relations in interactive scene generation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4560–4568, 2019. 18

[3] Guha Balakrishnan, Amy Zhao, Adrian V. Dalca, Frédo Durand, and John V. Guttag. Synthesizing images of humans in unseen poses. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8340–8348, 2018. 19

[4] Ali Borji. Pros and cons of gan evaluation measures. *Comput. Vis. Image Underst.*, 179:41–65, 2019. 10

[5] Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. *2018 13th IEEE International Conference on Automatic Face  Gesture Recognition (FG 2018)*, pages 67–74, 2018. 19

[6] Zhe Cao, Gines Hidalgo Martinez, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 15

[7] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A. Efros. Everybody dance now. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5932–5941, 2019. 21

[8] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *ArXiv*, abs/1512.03012, 2015. 17, 26

[9] Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018. 9, 17, 25

[10] Oran Gafni and L. Wolf. Wish you were here: Context-aware human generation. *ArXiv*, abs/2005.10663, 2020. 19, 20

[11] Ysbrand Galama and Thomas Mensink. Itergans: Iterative gans to learn and control 3d object transformation. *Comput. Vis. Image Underst.*, 189, 2019. 6, 18, 20

[12] Ian J. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *ArXiv*, abs/1701.00160, 2017. 11

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 7, 9

[14] Yana Hasson, Gül Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11799–11808, 2019. 26, 39

[15] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *ArXiv*, abs/1711.03213, 2018. 13, 17

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2016. vii, 9, 11, 12, 13, 29

[17] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 10

[18] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: a skinned multi-person linear model. *ACM Trans. Graph.*, 34:248:1–248:16, 2015. 26

[19] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *NIPS*, 2017. 15, 16, 24

[20] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhixiang Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821, 2017. 9

[21] Andrew T. Miller and Peter K. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics  Automation Magazine*, 11:110–122, 2004. 26

[22] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784, 2014. 7

[23] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017. 19

[24] Emmanouil Panagiotou, Georgios Chochlakis, Lazaros Grammatikopoulos, and Eleni Charou. Generating elevation surface from a single rgb remotely sensed image using deep learning. *Remote. Sens.*, 12:2002, 2020. vii, 14

[25] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. 7

[26] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands. *ACM Transactions on Graphics (TOG)*, 36:1 – 17, 2017. 26

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015. 13

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 22

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015. 22

[30] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2441, 2019. 17

[31] Wei Sun and Tianfu Wu. Image synthesis from reconfigurable layout and style. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10530–10539, 2019. 15, 18

[32] Tristan Sylvain, Pengchuan Zhang, Yoshua Bengio, R. Devon Hjelm, and Shikhar Sharma. Object-centric image generation from layouts. *ArXiv*, abs/2003.07449, 2020. vii, 18, 20

[33] Hao Tang, Wei Wang, Dan Xu, Yan Yan, and Nicu Sebe. Gesturegan for hand gesture-to-gesture translation in the wild. *Proceedings of the 26th ACM international conference on Multimedia*, 2018. 9, 10, 16, 17, 21, 24

[34] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4627–4635, 2017. 26

[35] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018. vii, 11, 14, 16, 24

[36] Zhengjiang. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004. 10, 11

[37] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Joshua B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS*, 2016. 17

[38] Bo Zhao, Xiao Wu, Zhi-Qi Cheng, Hao Liu, and Jiashi Feng. Multi-view image generation from a single-view. *Proceedings of the 26th ACM international conference on Multimedia*, 2018. 18, 20

[39] Yipin Zhou, Zhaowen Wang, Chen Fang, Trung Bui, and Tamara L. Berg. Dance dance generation: Motion transfer for internet videos. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1208–1216, 2019. 10, 16, 24

[40] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017. 39

# Appendix A

# Mode Collapse

In the first experiments using the Pix2Pix baseline loss, we observe that sometimes the training fails, and after some epochs with a stable SSIM improvement, the values suffer a sudden drop and are unable to recover. In order to identify the problem, we make use of the generator and discriminator loss plots to clarify what is happening.

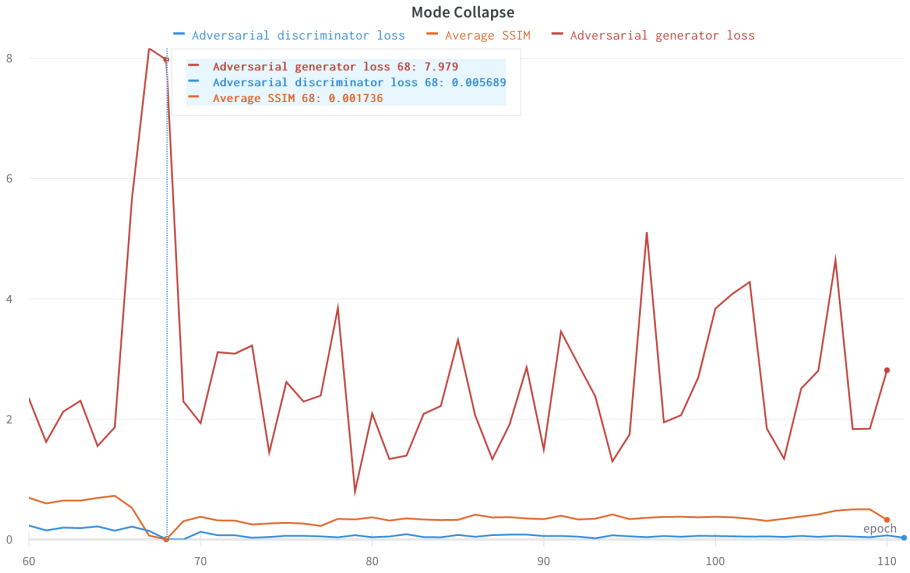



Figure A.1: Adversarial loss behaviour when mode collapse happens.

In the graph above, we can see that the discriminator loss decreases very fast. This means that the discriminator gets very good at classifying between real and fake images too early, and the generator is not able to generate good enough images. In the epoch 68, we see that the discriminator loss hits a minimum, and is able to perfectly classify the images. In this situation, the adversarial loss information getting to the generator becomes useless, the generation of images cannot improve and the training collapses.

A direct consequence is that the generator starts generating images that look alike, with similar color and texture patterns. In our case, we can see the generation of a pattern that is repeated in every generated image.

Figure A.2: Mode collapse example 1

Figure A.3: Mode collapse example 2

# Appendix B

# Object/Hand foreground baselines

In the different parts of the hand-object grasp 2.1 presented in the problem statement 2, we saw that there were parts in which hand and object occupied the same position of the 2D image, and therefore one shadowed the other. We can create two additional baselines that can render correctly three out of the four parts of the image:

- Object on foreground: We place the object directly on top of the hand image. This approach would succeed in rendering three of the four identified grasp zones, but fail in the object parts shadowed by the hand B.1b.
- Hand on foreground: We place the hand directly on top of the object image. This approach would succeed in rendering three of the four identified grasp zones, but fail in the hand parts shadowed by the hand B.1c.

(a) Hand-object (b) Object on foreground (c) Hand on foreground

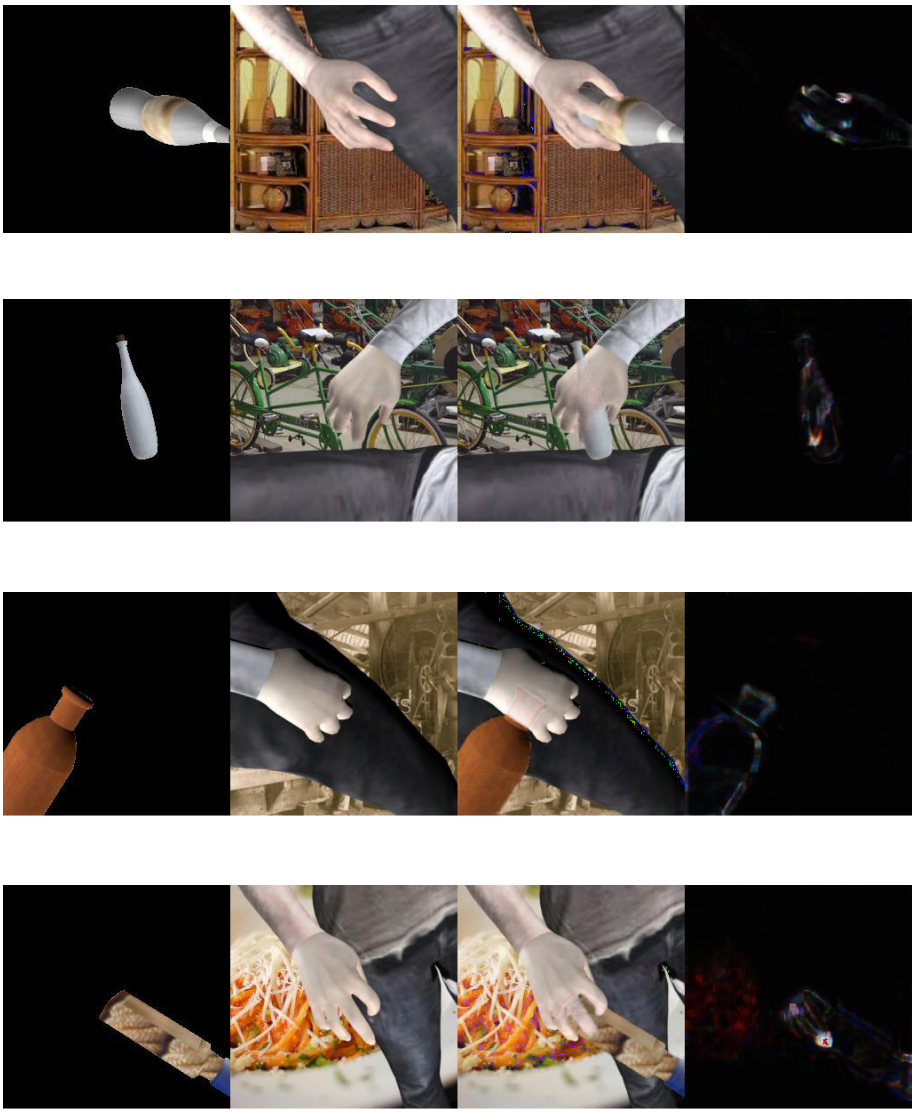
Figure B.1: Additional baselines for hand-object grasp generation. White for correct rendering, green and blue for incorrect rendering.

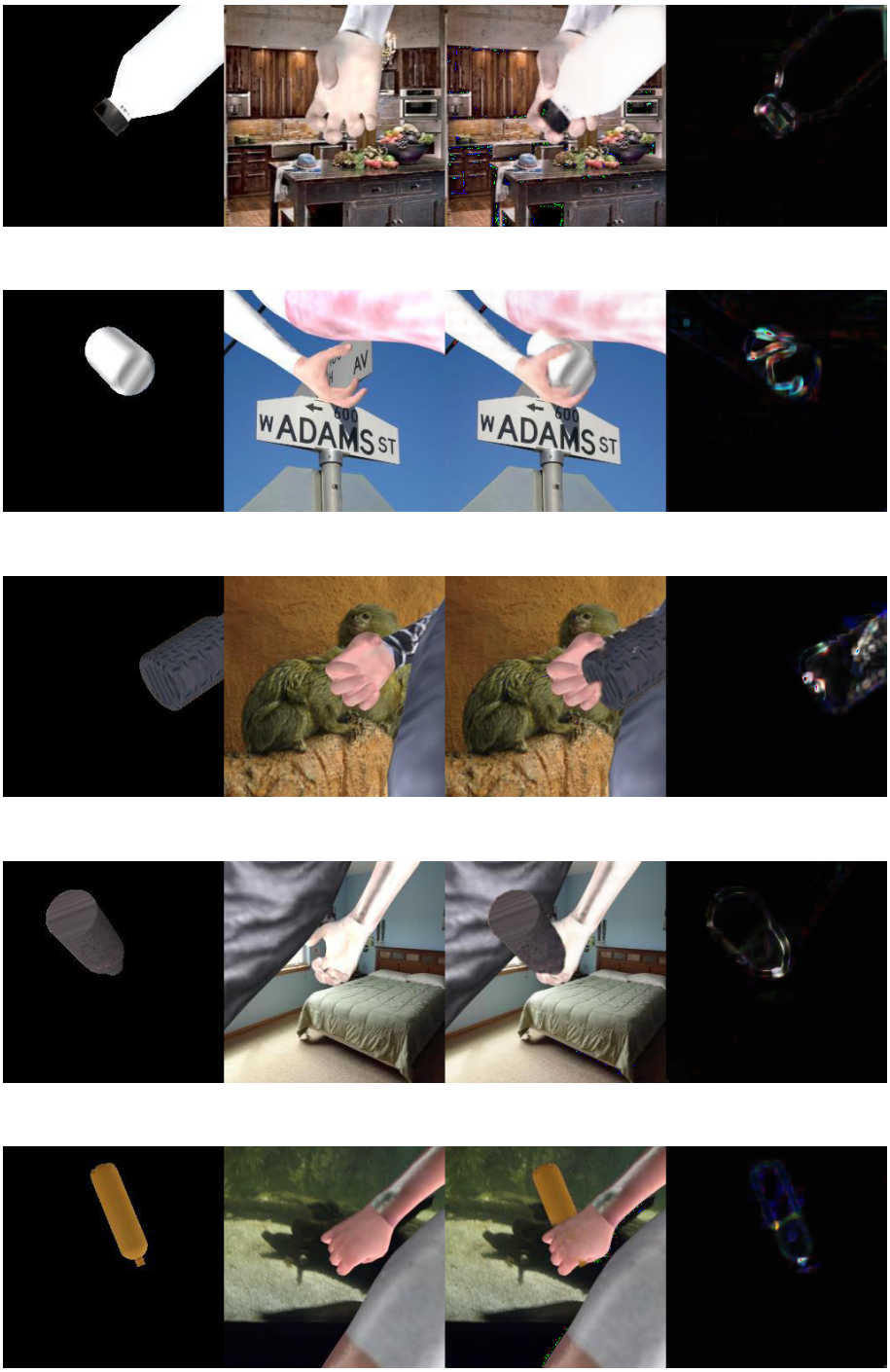
Both baselines with object or hand in foreground obtain high SSIM values of 0.83 and 0.89 respectively. As explained in 6.3.2, both of these baselines fail to learn one of the four areas identified on a hand-object grasp. The fact that these baselines score a higher SSIM than the models suggests that the SSIM does not penalise enough the situations in which the shadowed parts of the hand or the object are incorrectly rendered. As we explained in 3.2.4, the use of a quantitative GAN measure to evaluate the performance of the model is an active research field, and although the SSIM measure of the object area is the one that best suits our work, it still needs to come together with a qualitative inspection, as we see that the shadowed parts of the hand-object grasp are not sufficiently penalized.

# Appendix C

# Additional generated samples

From left to right: object mask, source image, generated image, SSIM difference map.




W ADAMS ST
W ADAMS ST