École Polytechnique Fédérale de Lausanne

De-occlusion of Occluded Vehicle Images from Drone Video

by Ogay Xavier, Dokmak Mahmoud

Bachelor Project Report

Approved by the Examining Committee:

Prof. Geroliminis Nikolaos
Main Supervisor

Tak Yura
Project Supervisor

Fonod Robert
Project Supervisor

EPFL-ENAC-INTER-LUTS
Room GC C2 389
Station 18
CH-1015 Lausanne

June 16, 2023

Contents

Abstract

In urban traffic analysis, the accurate detection of vehicles plays a crucial role in generating reliable statistics for various city management applications. However, occlusions occurring in densely populated city environments pose significant challenges to vehicle detection algorithms, leading to reduced detection rates and compromised data accuracy. To try to address this issue, we compared two of the novel models that leverage machine learning techniques for inpainting occluded vehicles, with the goal of improving the overall detection rate and enhancing the reliability of city statistics.

De-occlusion involves a two-step process: occlusion detection and inpainting. First, an occlusion detection algorithm is employed to identify regions within the traffic scene that contain occluded vehicles. Second, is an Inpainting model that given the occluded part, completes this hidden image fraction. We exclusively focused on the inpainting of images for this project.

One machine learning model is trained using a dataset of non-occluded vehicle images and the second also needs masks of the occluded part. The models learn to inpaint the occluded regions based on the available visual information. By leveraging contextual cues and vehicle appearance patterns, the model should effectively generate plausible completions of the occluded regions, restoring the missing vehicle details.

We wanted to evaluate the proposed method on a comprehensive dataset of UAV point-of-view single vehicles in urban traffic scenes, finetune it with another dataset of the LUTS lab and then comparing the ground truth image with the inpainted image to compare the results. An interesting future evaluation could be comparing the detection performance before and after applying the inpainting technique.

Github Repository: here

1 Introduction

The recent popularity and development of drones offer new promising solutions for aerial surveillance of urban areas. This method despite being widely used nowadays can probably be sorely improved by using the latest research and improvements in computer vision, probability theory, machine learning, and new dataset creation methods.

This Bachelor Project focuses on improving car detection using UAV (Unmanned Aerial Vehicles) by de-occluding the cars from the occlusions we usually find in our streets such as buildings, trees, shadows, and other obstacles that can hide parts of the vehicle when filming in a drone point of view.

In this context, we, first of all, went through a comprehensive literature review, which let us explore various studies related to car detection, de-occlusion, and inpainting, highlighting the challenges and advancements in the field. Our findings highlighted two families of models called Denoising Diffusion Probabilistic Models (DDPM) and Generative Adversarial Networks (GAN) that exhibit promising capabilities based on different mathematical models and so differ in certain aspects such as accuracy, efficiency, and robustness. At first, our relentless efforts were dedicated to unraveling the intricate workings of those models, striving to comprehend their inner workings and harness their potential. Once this part was done, we needed to test to see which one is going to meet our expectations in real-life applications and in our concrete problem of car de-occlusion.

To validate the capabilities of our models, we had to adapt the code to make it performs at our specifications and create suitable datasets for our models. We meticulously curated a few datasets employing various techniques for random mask generation. Our datasets were crafted with utmost care and precision, aiming to capture the complex nature of occlusions encountered in real-world scenarios. Once the datasets were created, we focused on the training of the models allowing the

models to learn from our own datasets.

The last part of our project is the testing part where we used different metrics and our own point of view to determine which one performs betters in our particular case and conclude which one is more likely to be used in a real-life application, taking care of the compute power each model requires. In particular, we will try to test which model performs best with small, medium, and big occlusions and which one provides us with the most satisfactory outputs.

2 Background

Definitions:

We are going to start by giving some important definitions and describing some concepts that we use throughout our project:

De-occlusion: De-occluding is the process of removing obstacles or more generally speaking occlusions that obstruct the view of a target object.

Inpainting: Inpainting consists in reconstructing damaged or missing parts of an image using surrounding information.

Markov Chain:

To keep it simple, we are not going into the details but giving a brief explanation and some properties. A Markov chain is a tool in probability theory used to describe a sequence of events (called states) where the events are related to each other in the following manner: The probability of transition from one state to another one depends only on the current state and not on the path taken that brought the system to its current state. In other words, a Markov chain can be described as "memoryless" because it disregards the past and focuses solely on the current state to determine the probabilities of transitioning to other states.

Bézier Curve:

A Bézier curve is an effective method for approximating the shape of a curve due to its utilization of parametric functions comprising a set of control points. Each Bézier curve is determined by three control points, with two representing the endpoints of the curve and the third governing its shape.

The generation of a parametric curve involves the application of linear interpolations, wherein a point is selected between two given points. By specifying the starting and ending points of the curve, the position of the third control point can be determined to establish the desired curve shape. Modifying the positions of the control points results in the alteration of the entire curve.

For instance, by placing the third control point at a distance of 30% from the origin and 70% from the endpoint, the shape of the resulting curve can be accurately computed. Furthermore, any adjustment to the position of the third control point will induce a corresponding transformation across the entirety of the curve.

GAN model *(Generative Adversarial Network)*:

GAN stands for Generative Adversarial Network and was the most famous technique used for image inpainting. GAN is in reality a more general model that is used in various domain that requires generating new inputs based on a training dataset. Generative Adversarial Networks were first introduced in 2014 [5]. GAN is a deep learning model and is composed of two neural networks, the generator and discriminator that are going to be trained at the same time in an adversarial manner.
For our case, when working with image inpainting, the training process works as follows: We provide as

input two identical images with the exception that the second one is occluded by a mask. The generator takes the masked image and begins with the help of a noise vector to create new complete images. On the other hand, the discriminator takes the unmasked image and the one created by the generator and compares them. The goal of the generator is to fool the discriminator by creating more and more meaningful images and the goal of the discriminator is to improve its ability at distinguishing real from fake images. To do this, the discriminator gives feedback to the generator to improve its generation process. Once the training is completed, we can feed the model a new image (one that hasn't been trained with) and the generator is going to generate based on its learning the missing part. GAN tries to produce sharp images and indiscernible inpainted regions. However, in numerous of our studies, we found out that the GAN model is very efficient for background completion and as one can imagine can't reconstruct a proper image when the mask is too big.

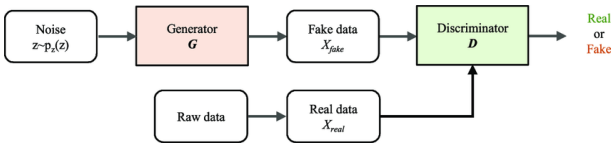


Figure 1: Vanilla GAN model scheme

DDPM *(Denoising Diffusion Probabilistic Model)*:

The first apparition of this model was for thermodynamics purposes in 2015 [18] but its popularity really arose from the Jonathan Ho study in 2020 [7]. The DDPM goal is to convert noise to a representative data sample.

"*Our goal is to define a forward (or inference) diffusion process which converts any complex data distribution into a simple, tractable, distribution, and then learn a finite-time reversal of this diffusion process which defines our generative model distribution.*"
– Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 2015 [18]

To train such a model the original images undergo progressive degradation through the introduction of noise, followed by the utilization of a neural network model for the image reconstruction using the degraded steps previously computed. This iterative process aims to eliminate the noise component at each step, gradually eroding the initial structure and distribution of the image. With sufficient iterations and high-quality data, the neural network model eventually acquires the capability to approximate the intrinsic data distribution of the original image. Consequently, starting with a noise-only input, the trained neural network can be employed to generate a novel image that is representative of the original training dataset distribution.

It happens in two steps [8] :
The forward Diffusion. In the forward diffusion process, the original image x_0 is gradually corrupted through iterative steps, following a Markov chain approach. This corruption is achieved by incrementally introducing scaled Gaussian noise. This process is repeated for a specified number of time steps T, resulting in the image representing Isotropic Gaussian at the final step, denoted as x_T. At each time step, the current image x_{t-1} is combined with the noise ϵ_{t-1} to produce the next image x_t. It's important to note that no model is utilized during this stage.

At the end of the forward diffusion stage x_T, the iterative addition of noise leads to the generation of a purely noisy image that conforms to an "Isotropic Gaussian" distribution. This conceptually signifies that the distribution of the data has been transformed into a Gaussian distribution, where the variance remains consistent across all dimensions.

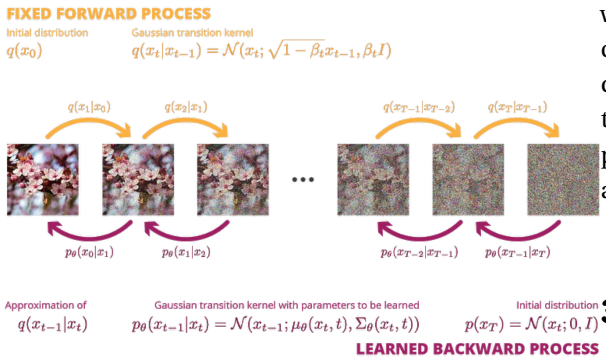


Figure 2: DDPM steps [19]

And the *Backward/Reverse Diffusion*:
During the backward or reverse diffusion stage, the objective is to undo the effects of the forward diffusion process. This is accomplished iteratively through another Markov chain approach, leveraging a neural network model. The model's task is to estimate and remove the noise that was added during the forward diffusion process.

Specifically, given a particular time step t and the corresponding noisy image x_t, the neural network model is trained to predict the noise $\bar{\epsilon}$ that was added to the image at the previous step $t-1$. By inputting x_t into the model, it generates an estimation (approximation) of the noise ϵ that was originally introduced during the forward pass at time step $t-1$.

Main differences between GAN and DDPM:

The training and generation process in diffusion models is generally more stable than in GANs due to their iterative nature.
Unlike GANs, which require the generator model to transform pure noise into an image in a single step ($x_T \rightarrow x_0$), diffusion models do not face this source of instability.
Diffusion models only require a single model for training, unlike GANs that rely on two models.
In diffusion models, the dimension of the image remains constant throughout the process, unlike GANs where the latent tensor can have varying sizes. This can pose a challenge in generating high-quality images due to limited GPU memory. However, the authors of the "Stable Diffusion" (or "latent diffusion" to be more precise [17]) approach overcome this issue by utilizing a variational autoencoder.

3 Literature Review

Our literature review brought us to discover the newest academic techniques related to our problem of car deocclusion. We quickly took note that the main concern and research in the domain were related to face recognition improvements.

More technically, when we deepened ourselves into the different papers and studies, we understoodd that the new methods rely increasingly on high-probability arguments and stochastic processes. The new probabilistic comprehension is widely increasing machine learning models' performance and extending their usability. The second huge advancement is related to the new technologies that can help to develop more precise datasets using different strategies to get big and complete ones. We can easily imagine that the development of powerful 3D creation tools such as Unreal Engine 5 permits the production of usable artificially synthesized images, masks, and data [2].

Unfortunately, many of these techniques are limited to a specific case and don't export well for our problem but we think it is an interesting sting to introduce you to the newest technologies in the domain.

In the following paragraphs, we are going to introduce and briefly describe, without getting into the details, some of the papers related to our problem that we find interesting.

The first paper that caught our intention is *PD-GAN: Probabilistic Diverse GAN for Image Inpainting* [11] a collaboration of multiple Chinese corporations and universities introducing PD-GAN their own improve-

ment of GAN model. As said, the main improvement techniques rely on more complex probabilistic computation and observations.
PD-GAN generation capabilities outperform vanilla GAN models by using context information in the contour. In fact, a classical GAN generates images randomly (following a random noise vector) while PD-GAN uses prior information by producing a coarse reconstruction of what could be the result based on a pre-trained partial convolution model and the masked region shape.
The PD-GAN pre-trained partial convolution model creates a SPDNorm matrix with the help of SPDNorm ResBlock blocks. This probabilistic matrix encapsulates lots of information about the kind of generation needed. More specifically, the SPDNorm ResBlock consists of SPDNorm Hard and SPDNorm Soft. The Hard SPDNorm controls the probability according to the distance between the pixel and hole boundary, while the Soft SPDNorm learns the probability in an adaptive process.
The Hard probabilistic diversity map D^h is only determined by the mask and transcribes the following observation: The pixels at the border have a high dependence on the known one and the pixels in the middle can be much more diverse.
The Soft probabilistic diversity map D^s is an adaptive map that is obtained by the input feature and coarse prediction through a learning process.

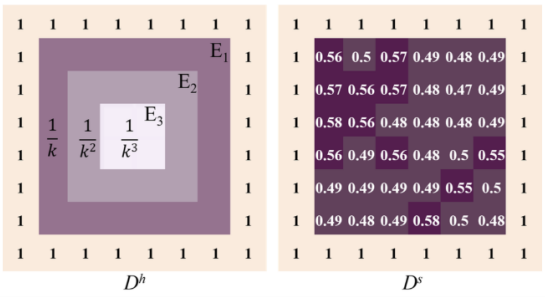


Figure 3: Hard and Soft diversity maps

The diversity is enhanced towards the hole center while is reduced towards the hole boundary. As said in the introduction, PD-GAN is one of the latest versions of a GAN model and uses new probability theory (in this case SPD Matrix) to better control and modulate the random noise. This permits this model to control the diversity of the model and produce more "smart" outcomes. We couldn't use this model for our project, as for most of the papers we read, the code was not available.

Another paper which was particulary relevant to our research was *Visualizing the Invisible: Occluded Vehicle Segmentation and Recovery* [22].
The model presented in this study focuses on the task of vehicle de-occlusion, aiming to restore the appearance of occluded vehicles. The model consists of two distinct modules: the segmentation completion network and the appearance recovery network.

The segmentation completion network is responsible for generating a realistic mask that accurately represents the boundaries of the occluded vehicle. This module employs 3D models of cars to fill in the missing regions of the vehicle, resulting in a complete and coherent segmentation mask.

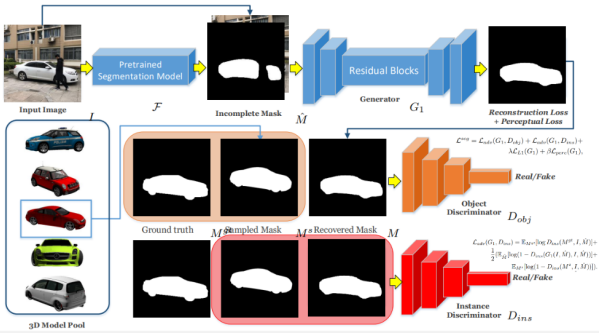


Figure 4: Segmentation Completion Network

On the other hand, the appearance recovery network is designed to recover the visual details of the occluded vehicle. Leveraging the information provided by the segmentation completion network, this module utilizes deep learning models to reconstruct the obscured appearance of the vehicle. By effectively inferring the

missing appearance information, the network generates a visually plausible representation of the occluded vehicle.

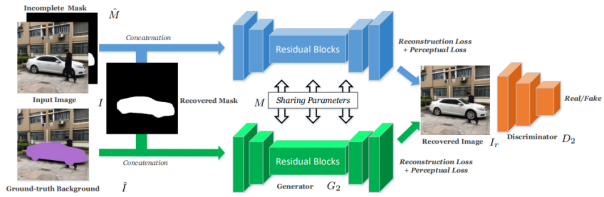


Figure 5: Appearance Recovery Network

Through the combined operation of these two modules, the model demonstrates its capability to tackle the challenge of vehicle de-occlusion. By accurately segmenting the occluded regions and recovering their appearance, the model mostly accurately restores the original visual information of the occluded vehicles.

Un-hopefully, the code was not available and we would have needed to manually label the segmentation of each of the vehicles on the ground truth images to create a training dataset alongside creating/searching diverse 3D vehicle models.

Also, the constraint applied to the mask segmentation is great but not as useful in our case as we always will have an UAV POV.

To end our literature review, we would like to discuss a noteworthy research paper that focuses on a highly significant area of interest within the industry related to inpainting and de-occlusion which is face recognition. This recent article, titled *Complete Face Recovery GAN: Unsupervised Joint Face Rotation and De-Occlusion from a Single-View Image* [10] was authored by several Departments of Korea University in Seoul and published in 2022.

This groundbreaking article introduces a novel technique that uses a modified Generative Adversarial Network named Complete Face Recovery GANGANs (CFR-GAN) to reconstruct damage caused by performing 3D facial modeling for an image. The idea behind this article is to be able to reconstruct front faces from an image of a person oriented in any direction or occluded by diverse objects. To do so, they fine-tuned the 3D Morphable Model (3DMM), a very famous model for face recognition, that enables the generation of a comprehensive 3D representation of a face based on a single image.

The 3DMM is a highly effective technique in computer vision and computer graphics. This model is trained with a huge database of 3D face scans and their corresponding 2D images that enables to capture the statistical variations in facial shape and texture. Using this information, we can construct 3D models of a face even with a picture of a rotated head.

This exceptional and intricate technique represents a significant milestone in the realm of computer vision. Its integration of GANs not only enhances the clarity of de-occlusion and inpainting tasks but also enables the creation of comprehensive 3D facial representations, expanding the possibilities of face analysis and reconstruction.

Figure 6: Image taken from the *Complete Face Recovery GAN: Unsupervised Joint Face Rotation and De-Occlusion from a Single-View Image* qualitative results of their model on CelebA-HQ and FFHQ datasets

4 Design

After conducting thorough research on various papers and models, we have discovered two noteworthy class models that hold significant potential. As you likely gathered from the background section, the models we focused on include the very famous GAN model, as well

as a more recent model based on Denoising Diffusion Probabilistic Model (DDPM).

In order to accomplish our objective of de-occluding cars, we have made the decision to employ two enhanced and recent implementations of these models called respectively AOT-GAN and Repaint. Our aim is to leverage the inherent capabilities of these models and effectively utilize them to attain our desired outcome of de-occluding vehicle images.

In this section, we delve into a detailed description of both the AOT-GAN and Repaint models. We provide insights into their architectures, training methodologies, and underlying mathematical principles.

AOT-GAN: *Aggregated Contextual Transformations for High-Resolution Image Inpainting*

The first paper was published to the IEEE in 2021 and was written by several Chinese computer vision experts and brings new interesting optimization to GAN model. AOT-GAN stands for Aggregated Contextual-Transformation GAN. [23]

AOT-GAN introduces two significant enhancements to the training process, aimed at improving both the generator and the discriminator.
The first improvement focuses on capturing context information effectively. In a traditional GAN setup, the generator generates synthetic images solely from a noise vector. However, AOT-GAN follows the same intuition as PD-GAN [11] and leverages probability theory to enhance its context reasoning capabilities. This modification addresses a limitation of vanilla GANs, which primarily excel in completing small gaps within stationary backgrounds but struggle with understanding and reconstructing missing contents in complex scenes.
By incorporating probabilistic context reasoning, AOT-GAN aims to overcome this limitation and generate more plausible and realistic inpainting. This permits to AOT-GAN to improve its capabilities to hallucinate plausible content which demonstrates an interesting versatility in generating images.
At last, this improvement permits this particular GAN to work with big pictures and big masks which demonstrates once again the power of the model.

The second improvement is in the discriminator design. The discriminator of GAN aims to distinguish real images from those of the generator. This is mainly done patch-wisely which means that the neural network will segment the image produced by the generator and compare them with the ground truth. While this seems to be a great idea there is still one default that certainly decreases the discriminator capabilities.
In fact, the generator will only generate the missing part and keep the other parts untouched. The problem with the discriminator of a vanilla GAN is that it compares patch-wisely the fake and the real image without knowing which part of the image has been inpainted. This limitation weakened the comparison capabilities of the discriminator by noticing that a huge part is similar. This observation lead to the second improvement called mask predictor. The discriminator of AOT-GAN brought a clear improvement for synthesizing realistic fine-grained textures compared to other GAN by giving the indication to only compare the generated part with the original chunk of the image.

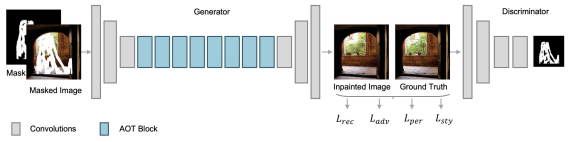


Figure 7: AOT-GAN model scheme

AOT-GAN, to be trained, takes as input a masked image and a binary mask for the generator and it also needs the ground truth image for the discriminator as you may have understood from the previous explanation. For testing, we need the masked image and a binary mask indicating the position of the mask.

The Generator, as illustrated in Figure 7, shares simi-

larities with a vanilla GAN in terms of its architecture. It comprises an encoder that utilizes several layers of standard convolutions to encode features from the input. The encoded features are then processed by a stack of multiple-layered building blocks and finally, we use multiple layers of transposed convolutions to decode features from AOT blocks to a completed image as the final result. However, the distinctive aspect lies in the construction of these blocks. AOT-GAN introduces its own AOT-Blocks, which differ from the conventional Residual Blocks employed in other GANs. These AOT-Blocks are specifically designed to ensure a coherent structure with surrounding image contexts and effectively capture various angles and scales of view within different parts of the scene. To accomplish this, the AOT-Blocks perform numerous operations to propagate information from distant contexts to the missing regions, enabling comprehensive context reasoning.

More concretely AOT-Blocks work in three steps: Splitting, Transforming, and Aggregating. At first, the AOT-Block splits the kernel of a standard convolution into equal size sub-kernels with fewer output channels. Second, each small part created will undergo different transformations such as dilation at different rates. Third, the contextual transformations from different receptive fields are finally aggregated by a concatenation followed by a standard convolution for feature fusion. To compute, how to aggregate them, they use a gated residual connection to compute the spatially-variant gate value g from x_1 by a standard convolution and a sigmoid operation. The output is the sum of g-weighted x_1 and x_2 as shown in Figure 8. One important and great observation about AOT-GAN is that it doesn't introduce any extra model parameters and computational costs.

AOT-GAN model is trained using a combination of different loss functions: a reconstruction loss, an adversarial loss, a perceptual loss, and a style loss. This has been studied to be the best to synthesize reasonable contents and clear textures for missing regions of high-resolution images.

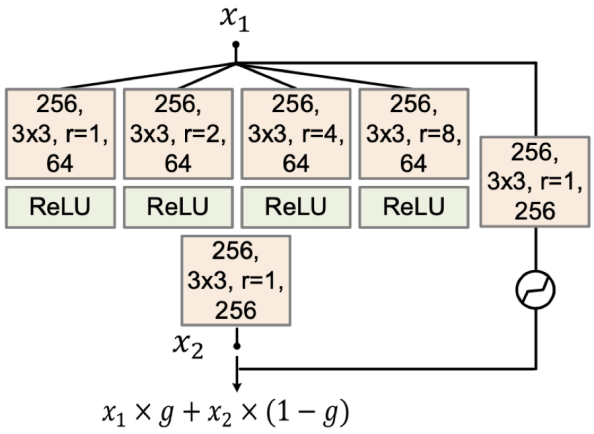


Figure 8: Design representation of an AOT-Block

The second improvement is in the design of a tailored mask-prediction task for the discriminator, which can in turn facilitates the generator to synthesize clear textures. This happens because in most of the previous GAN, the discriminator who compares the ground truth and the fake image patch-wisely either can't differentiate known patches from unknown ones (such in PatchGAN [9] which is the worst case), either can't differentiate known parts from inpainted within patches (such in HM-Patch-GAN).

The authors employ a Soft Mask-guided PatchGAN (SM-PatchGAN) approach to facilitate the synthesis of fine-grained textures. They downsample the inpainting masks and use them as ground truth for a mask-prediction task. Gaussian filtering is applied to the masks to simulate pixel propagation around the boundaries of missing regions. This process generates a soft patch-level mask that provides a probability distribution indicating the likelihood of reconstruction for different parts of the damaged areas.

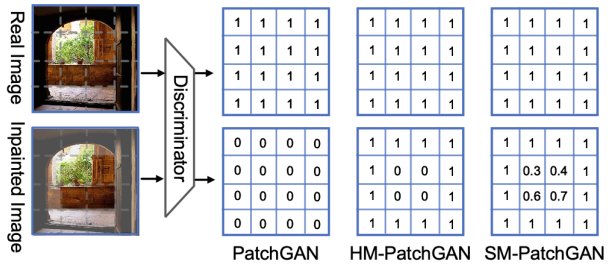


Figure 9: Use of the mask & probability of reconstruction to inpaint missing region

REPAINT: *Inpainting using Denoising Diffusion Probabilistic Models*

The second paper, authored by the Computer Vision Lab of ETH Zürich and presented in the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition in 2022, introduces a novel technique for image inpainting. [13]

Following extensive research, Repaint has emerged as the most promising model for our project as the newest image generator such as DALL-E 2 [16] and Midjourney [17] model are Diffusion-based Image Generation models. It leverages the principles of Denoising Diffusion Probabilistic Models (DDPM) and presents a totally new technique that stands out from classic GAN models while offering more than interesting results.

The training process is totally different from everything we have presented thus far and relies on Markov chains, discussed in the Background.

The first thing to note, training doesn't require a mask, the training relies on learning the distribution of raw images in the training set. More precisely, during the training, DDPM is going to transform an image x_0 to a totally indistinguishable image $x_T \sim \mathcal{N}(0,1)$ in T steps by defining a diffusion process that will follow the properties of a Markov chain of T states. More precisely, each step-image depends only on the previous one and the nature of the transformation is the pixel-wise addition of i.i.d. Gaussian noise with variance β_t (if we consider we are at timestep t ie. going from state x_{t-1} to x_t) and with mean a scaling of the previous sample mean x_{t-1} with $\sqrt{1-\beta_t}$ according to a variance schedule which gives us the following state-transition:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Furthermore, By using the Gaussian and independence properties of the noise added at each step, we can, as in stationary Markov chain, calculate the total noise variance added to transform state x_0 to x_t as $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ which let us rewrite the above expression by :

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t I)$$

Having this Markov chain, the real "training" can start and consist of computing the reverse Markov chain (the transition of a state x_t to a state x_{t-1}). Since the original Markov chains add i.i.d Gaussian noise at each step we can easily imagine that the reverse Markov's chain transition follows Gaussian distribution :

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t|t), \Sigma_\theta(x_t|t))$$

This reverse model is modeled by a neural network that predicts the parameters $\mu_\theta(x_t|t)$ and $\Sigma_\theta(x_t|t)$ of the Gaussian distribution that represents the probability of transition in the reverse Markov chain.

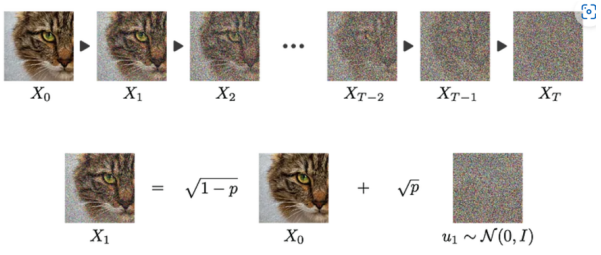


Figure 10: Markov chain of an image

The Repaint model uses a pre-trained unconditional DDPM as the generative prior and "only" condition the generation process as we can read in the following

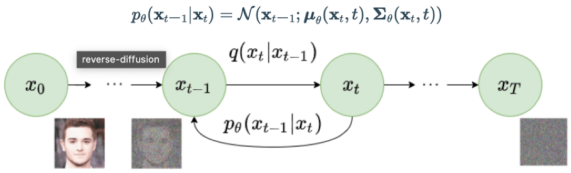


Figure 11: Forward and Backward process

quotation.

"*We propose RePaint: an inpainting method that solely leverages an off-the-shelf unconditionally trained DDPM. Specifically, instead of learning a mask-conditional generative model, we condition the generation process by sampling from the given pixels during the reverse diffusion iterations. Remarkably, our model is therefore not trained for the inpainting task itself. This has two important advantages. First, it allows our network to generalize to any mask during inference. Second, it enables our network to learn more semantic generation capabilities since it has a powerful DDPM image synthesis prior*" – RePaint: Inpainting using Denoising Diffusion Probabilistic Models, 2022 [13]

This means that the training process needs to be performed only on a DDPM model and thus this model in our project will not require masks for the training part.

To understand how Repaint infer masked images, ie generate the masked part with the help of the pre-trained DDPM model, let's denote the ground truth image as x, the unknown pixels as $m \odot x$ and the known pixels as $(1-m) \odot x$.
At a given state x_t, to determine the next state in the reverse Markov chain x_{t-1} we do the following: We first should dissociate the part of the image that is known and the inpainted part x_t^{known} and $x_t^{unknown}$ because we should not modify the known part. To reconstruct x_{t-1}^{known} we can use the direct formula on x_t to denoise it but to denoise $x_t^{unknown}$ to get $x_{t-1}^{unknown}$ we sample similar data from the training. This gives us the following formula to go from state x_t to x_{t-1} :

$$x_{t-1}^{known} \sim \mathcal{N}(\bar{\alpha}_t x_0, (1-\alpha_t) I)$$

$$x_{t-1}^{unknown} \sim \mathcal{N}(\mu_\theta(x_t|t), \Sigma_\theta(x_t|t))$$

$$x_{t-1} = m \odot x_{t-1}^{known} + (1-m) \odot x_{t-1}^{unknown}$$

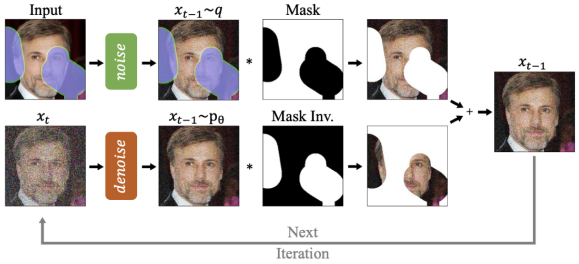


Figure 12: Overview of the approach

Using this scheme, Repaint is able to inpaint only masked regions with a Denoising Diffusion Process model while keeping the surrounding of the mask intact in the final result.

5 Implementation

Datasets Creation

Two distinct datasets were curated specifically for the purpose of vehicle de-occlusion, with the objective of restoring the appearance of occluded vehicles. In order to facilitate the training and evaluation process, certain preprocessing steps were undertaken on the datasets such as eliminating some occluded vehicles.

Firstly, the ground truth images were resized to the desired dimensions, ensuring consistency and compatibility with the model architecture. This resizing operation was performed to establish a standardized format

for the ground truth images, enabling effective comparison and analysis during the training and evaluation stages.

Additionally, to simulate occluded regions within the dataset, random masks were generated for all images. These masks were created using a randomization process in which we can choose the probability of occurrence of each mask shape, introducing variations and occlusions that could mimic real-world scenarios. By incorporating those various masks shape, the datasets captured a diverse range of occlusion patterns and scenarios, contributing to a more comprehensive and representative training and evaluation framework.

The creation of these curated datasets, encompassing resized ground truth images and randomly generated masks, provides a robust foundation for conducting vehicle de-occlusion tasks.

VRAI: ***Vehicle Re-identification for Aerial Image*** **[20]**

"VRAI is a large-scale vehicle ReID dataset for UAV-based intelligent applications. The dataset consists of 137, 613 images of 13, 022 vehicle instances. The images of each vehicle instance are captured by cameras of two DJI consumer UAVs at different locations, with a variety of view angles and flight-altitudes (15m to 80m)." – Opendatalab Introduction for VRAI dataset, 2019 [20]

For the VRAI datasets, we only worked with the train part already split on the original dataset. Which made us work with 66,113 images. To improve the pool of images we could merge their test, train, and dev part of the dataset and then use our script to split it again in a test and train part.

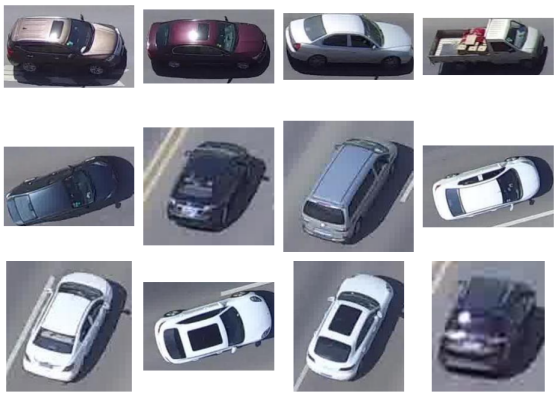


Figure 13: Sample of VRAI images

Pully: ***Vehicle from a UAV POV extracted from Video of Pully parking***

This dataset was provided by the EPFL LUTS lab which preprocessed some videos of Pully parking by extracting each detected vehicle. It contains 31,239 images and is including every detected vehicle such that few of them are mostly hidden by occlusion, mostly trees, and vegetation.

To sort those occluded images, as we want a clean dataset for training, we used a script to detect the overall and center patch ratio of green in the images (vegetation color). If one of those two ratios exceeded some arbitrary thresholds setup by our hands we split those green-intensive images to not be included in the training process. This method permits the exclusion of most of the occluded cars without the need of manual intervention, at the expense of any green vehicle.

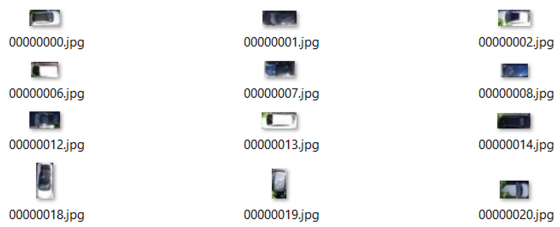


Figure 14: Sample of PULLY images

Dataset Creation Python script

The first attempt to create a dataset was by using PyTorch such that we could use already implemented dataset creation functions which are GPU optimized to accelerate the process of all images. But we didn't have the knowledge of using those PyTorch GPU usage libraries alongside the mask creation.

As the process of resizing and renaming images is pretty straightforward forward we decided to implement the whole dataset creation pipeline ourselves and use the CPU as we are working with small images anyway (ie. 128 x 128 px)

Our goal was to provide a script processing the given image data and create a mask for each of them. A mask is an image of the same size as the ground truth image but composed of only two colors. One color represents the part of the image to be kept and the second one is the patch to be inpainted. We also let the script user choose the most options to configure the dataset creation.
Such as:

- Where to store the train and test images
- Which ratio to use of how much of the data is to be used in the training and testing (ie. 80:20)
- The color of the mask and of the background
- The ability to resize the image or to keep the original size
- The option to choose the ratio of the original image to be hidden
- The choice in the frequency of apparition of each shape as a mask

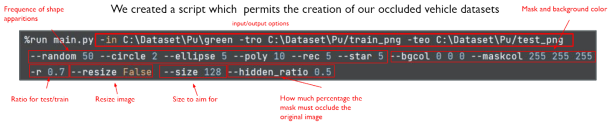


Figure 15: Script run command

Test/Train ratio:
The data separation between the training and testing process is crucial. As if we train our model with certain images, it will obviously perform especially well as it is some "known" values. To avoid this problem we select some random images in the total original image data and isolate them in another folder so they are not used in the training process. Those images are kept for testing purposes. They will be used to assess the model performance by using "unknown" images and evaluate how great it performs on those with appropriate metrics. So in our script, we let the user choose the ratio of train images with the following command option:
–train_ratio [ratio between 0 and 1]
or
-r [ratio between 0 and 1]

To choose the input folder of the complete data:
–data_path [path to the images]
or
-in [path to the images]

To choose the output folders:
–train_data_output [path to the train folder]
or
-tro [path to the train folder]
and
–test_data_output [path to the test folder]
or
-teo [path to the test folder]

Mask/Background color:
For our case, the mask color has a great significance as Repaint uses white color [255, 255, 255] as part of inpaint whereas AOT-GAN uses black color [0, 0,

0] for this task. So our script includes the following command:
To choose the mask color:
–maskcol [the color to use [0-255, 0-255, 0-255]]

To choose the background color:
–bgcol [[the color to use [0-255, 0-255, 0-255]]

Size of the image:
For both of our models, we need to specify the square size of the images we want to work with. They can load images out of this specified size by automatically resizing them but it induces the cropping of those ones. To avoid that cropping, as we want to train images on the whole vehicle, we add the ability to resize images while keeping their original aspect ratio. This has the disadvantage of adding color padding to the border of the image (here in white).
You can also keep the original size of the image so the mask generated will take the appropriate size.

Figure 16: Cropped car

Figure 17: Our script resized car

The following command permits us to address size with our script:
To choose if you want to resize images:
–resize [True/False]

To choose the square size of the output masks and images:
–size [size of the image ie. 128]

Shape generation
To create a mask we decided to create multiple shapes. Rectangles, polygons, stars, circles, ellipsis, and random shapes can be generated by the script. It uses a library named *Cairo* to draw those shapes on a 2D grid. Each shape has random properties at each mask generation, such as the number of branches for the stars or the number of edges for polygons.

```
def drawStar( ctx,
              outerRadius,
              innerRadius,
              numVertices=5):

    circ = 2 * math.pi
    angle = circ / (numVertices * 2)
    coords = []
    for ii in range(numVertices * 2):
        r = innerRadius if ii % 2 else outerRadius
        coords.append([math.sin(angle * ii) * r,
        math.cos(angle * ii) * r])
    for ii in coords:
        ctx.line_to(ii[0], ii[1])
    return ctx
```

Figure 18: Example of star generation

As the shapes are generated in a 2D grid system of coordinates, we needed to make their positions random and also make sure that those shape doesn't go out of the original size of the image. But the most complicated challenge was definitely to generate a random shape.
As easy as it could sound, creating a single random shape in computer science is not so obvious. Our first idea was to randomly choose some points of the canvas and join them and if any of the edges crossed one another reselect other points and so on.
It was not an optimal solution and didn't really represent some real-life occlusion mask. So we abandoned this idea and headed toward the idea of 2D Gaussian noise. We could use the random noise to generate a noise image, then control its size using Gaussian blur, and finally apply a threshold to only have the two colors required as a mask.
This method resulted in great results but we could not really define the center of the random shape nor could we constrain the noise to generate only one shape.

Figure 19: Gaussian Noise based mask : *as you can see there is two white shapes*

Finally, to create a random shape we came back to select random points but with some constraints, and thanks to the help of Bézier curve (details in section 2) we could achieve a similar result with a much more manageable single shape.

Figure 20: Bézier curve based mask

Shape Frequency:

To really add liberty to our mask generation script we wanted to make the user able to choose if a shape should appear or not and in which proportion.
To do so, the user needs to specify a number for each shape and its apparition will be weighted on the overall sum of all those given number.
For example, if we give 2 for rectangle, 3 for circle and 5 for polygons, the rectangle shape will have the probability $\frac{2}{2+3+5}$ to appear in a mask.
You need to specify the number 0 for a specific shape to not appear. The following commands are available:

To choose the relative weight of the apparition of rectangles:
–rec [the weight of rectangle apparition]

To choose the relative weight of apparition of polygons:
–poly [the weight of polygon apparition]

To choose the relative weight of the apparition of stars:
–star [the weight of star apparition]

To choose the relative weight of the apparition of circles:
–circle [the weight of circle apparition]

To choose the relative weight of the apparition of ellipsis:
–ellipse [the weight of ellipsis apparition]

To choose the relative weight of the apparition of random shapes:
–random [the weight of random shape apparition]

Ratio of occlusion:

We wanted to let the script user have the choice of how much of the original image should be occluded. It can be useful for evaluation purposes such as having the tool to say our model has those metric results on a 50% occluded vehicles dataset and these metric results on a 90% occluded vehicles dataset.
The implementation of such an option was challenging as we need to generate a shape given an area that we computed relatively to the given hidden ratio. For the rectangle, it was pretty easy as the area formula is straight forward but you can imagine that the formula for a star or even a random shape is significantly sophisticated.
For complex shape, we decided to use an approximation of area as it would require much more computational power for not so needed precision.
For the training purpose we made this hidden ratio randomly vary from 10% to 90% for each mask so the model can be efficient on multiple cases and not bounded to a specific range of occlusion.

The following command can be used to control

this hidden ratio parameter:
–hidden_ratio [the hidden ratio between 0 and 1]
or
-hr [the hidden ratio between 0 and 1]

Thresh-holding color:
As masks are composed of two unique colors, we want to avoid a gradient of color which can happen when resizing because of interpolation and when generating shape with the python Cairo library used in our script.

To address this problem, we apply a threshold to color on all image pixels just before saving the mask thus constraining the usage of binary color. This implementation is really easy as the Intel python library *Open-CV* has a function exactly for that purpose

```
import cv2
...
ret, th = cv2.threshold(tmp,
            127,255, cv2.THRESH_BINARY)
th = cv2.cvtColor(th, cv2.COLOR_BGR2GRAY)
```

White/Black binary threshold of mask

Figure 21: Train Ground Truth Samples

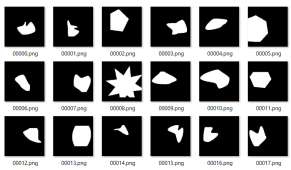


Figure 22: Train Masks Samples

Model Training Pipeline

After creating our training dataset, which is only resized ground truth images for Repaint and resized ground truth with associated black-on-white background mask for AOT-GAN, we needed to run the training process of both those models. As those two models required about ten V100 GPUs running for about eight to fifteen days in their original paper release, we also needed an appropriate infrastructure to proceed with our pieces of training.

Scitas: ***Scientific IT & Application Support***

Our first attempt to accede to such an infrastructure was through Scitas, the EPFL structure for such a project. It took some time for us to have access to this cluster but we finally had a comfortable budget to make our computations.

Scitas has many clusters and we chose to work with Izar. Izar is an Intel Xeon-Gold-based cluster, available to the EPFL research community which is aimed at GPU usage. That was exactly what we needed but the taming of this new beast was not without issue.

The Izar cluster uses Lmod, an Environment Module System, which neither of us was familiarized with. So we learned to manage this new environment and to use the Slurm Workload Wanager to launch our tasks. But when finally launching those tasks, we realized that Lmod couldn't let us install the latest TensorBoard and Pytorch module. This made us unable to launch the training process on Scitas. We have nevertheless let the scripts used on GitHub with some instructions if someone is interested in finding a way of launching those training.

```
#!/bin/bash
#SBATCH --chdir /home/xogay
#SBATCH --partition=gpu
#SBATCH --qos gpu
#SBATCH --gres gpu:1
#SBATCH --time 00:05:00
#SBATCH --requeue

echo STARTING AT $(date)
echo "Job run at: $(hostname)"

module load gcc/8.4.0-cuda py-torchvision/0.6.1 python/3.7.7 mvapi
ch2/2.3.4 py-tensorflow/2.3.1-cuda-mpi py-torch/1.6.0-cuda-openmp

source /home/xogay/venvs/aot_inpainting/bin/activate
echo "Activating conda environment..."
eval "$(conda shell.bash hook)"
conda env create -f /home/xogay/env.yml
conda activate inpainting
conda install pytorch torchvision -c pytorch
module load gcc/8.4.0-cuda py-torchvision/0.6.1 python/3.7.7 mvapi
ch2/2.3.4 py-tensorflow/2.3.1-cuda-mpi py-torch/1.6.0-cuda-openmp
echo "Activated virtual environment:"
echo $CONDA_DEFAULT_ENV
echo "bash ${@:1}"
bash "${@:1}"

cd /home/xogay/AOT-GAN-for-Inpainting/src
python train.py --dir_image "/home/xogay/AOT-GAN-for-Inpainting/sr
c/data/new/gt" --dir_mask "/home/xogay/AOT-GAN-for-Inpainting/src/
data/mask" --data_train "" --data_test "" --mask_type ""  --image_
size 128  --batch_size 16 --print_every 1000  --save_every 1000 --
save_dir "/home/xogay/AOT-GAN-for-Inpainting/trained"

deactivate
echo FINISHED AT $(date)
```

Figure 23: Sbatch script for launching AOT-GAN : *Doesn't work as Pytorch module is not up to date*

Google Collab

We then used Google Colab Pro to train our models as they proposed the location of V100 and A100 GPUs while having a user-friendly environment. This permitted us to continue our work but at the expense of time as just uploading one of our datasets took at least four hours and the setup of each project needed to be redone at every run of the Google Colab file as those run on Virtual Machines. All Interactive Python Notebooks are given on the Github repo.

Evaluation Metrics

We set up a pipeline that also needs a GPU to compute the evaluation results of our models. Those are the metrics we selected to evaluate our work:

- **L1/Manhattan Norm**: Compute the mean absolute error between the generated image and the original one to determine the per-pixel reconstruction accuracy.
- **L2/Euclidean Norm**: Compute the mean squared error between the generated image and the original one to determine the per-pixel reconstruction accuracy.
- **SSIM**: It's a metric based on the human perception that outputs a number between 0 and 1 based on the similarity of three aspects: luminance, contrast & structure. An output of 1 means it's the same image.
- **PSNR**: It computes the ratio between the maximum possible power of the image and the power of the noise that is introduced during the reconstruction. The output is expressed as the logarithm of the PSNR ratio in db. A greater output means better performance.
- **LPIPS**: LPIPS is used to judge the perceptual similarity between two images. LPIPS essentially computes the similarity between the activations of two image patches for some pre-defined network. This measure has been shown to match human perception well. A low LPIPS score means that image patches are perceptually similar.
- **MAE**: measures the average magnitude of the errors in a set of predictions ie. the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.
- **FID**: The Fréchet distance computes the distance between two probability distributions: the distribution of the reconstructed images and the real images. A smaller output is a sign of good performance.

AOT-GAN

The main difficulties encountered in the training of AOT-GAN were its lack of popularity, lack of documentation, and lack of error messages.
We didn't manage to train this model correctly and could not discover why.

We launched a training right after the end of the dataset creation and let it runs for almost one million iterations. It was a flaw on our side as we lost some time training this model for almost a week and just realized afterward that the trained GAN model was not outputting what we expected (see Table 5).
That error taught us to test the viability of our model before letting the training run to not waste time and resources on a failing process. But this result, of the training process not converging, seems to have kept track during all our effort.
The issue is that nothing seems to be incorrect in the code and that we have no error messages to give us a direction of the problem to search for.

After a careful reading of the code, we realized that the model was able to read the JPEG and the PNG format for the ground truth image but only the PNG format for masks. Correcting the datasets format did

not affect the results.
In exploring the original images and masks, given in the sample example, we figured out that the mask had not three color channels but only a grayscale channel. After implementing that correction to the dataset script and regeneration of it, the newly trained model seemed to also not converge (see Table 6)
As a last effort, we tirelessly tried to change some training parameters, tried to understand some of the most complex parts of the code, and printed the loaded image at different parts of the training process to see if images and masks were correctly read which were indeed well loaded.
We also tried to finetune the pre-trained model on the dataset Places2 provided by the authors of AOT-GAN, but along the training process, we could progressively see the degradation of the output images (See Table 7). Given all the experiments we did on AOT training, we would say that a problem most probably should occur in the data reading/loading but we did not manage to find how to fix it as we ran out of time.

Repaint

We expressed some concerns about the Repaint training. As we are filling in missing parts to keep the aspect ratio of images with a uniform color and the training process for Repaint is the pipeline of Guided Diffusion [6] which is a DDPM. It means that the model is learning to generate the whole image and those images have artificial patches to fill the image. We feared that this could lead to a major issue in the inferring part. So we had got two ideas:

Figure 24: Square patch technique

Figure 25: Deformation mapping technique

Either extracts a squared image from the video instead of a patch of the size of the vehicle. This solution made us wonder if having extra non-vehicle information for the training could help the generation or, on the contrary, make the model prone to hallucinations.
It also raised another problem as we want to train the model on VRAI which only gives us patches of the vehicle. So we lacked those data and could only use the Pully dataset.

We came up with another idea which was to use a bijective mapping to deform before and after the model application. This would make the model only inpaint deformed vehicles and would require more pre-process and post-process work.

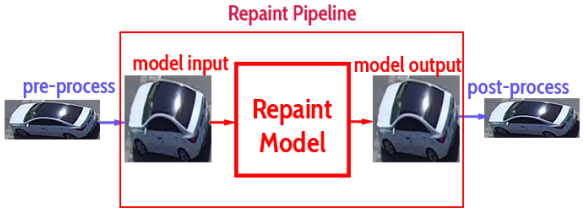


Figure 26: Repaint deformation mapping pipeline

The deformation mapping technique requires us to write in the image metadata its original size. The model also needs to keep intact this metadata and after the inference of Repaint we need to retrieve this original size to restore the image aspect ratio afterward.

The major difficulties we encountered with Repaint were the fact that the training process uses another Open-Source project named Guided-Diffusion [6] from OpenAI.
Such a separation exposed us to some severe model incompatibilities, as at the time Repaint was developed the architecture of Guided diffusion was probably different. After the exploration of the code of both models, we find out that it was some parameters mismatch between them.
We finally achieved to make that training process work by loading the parameters of Guided Diffusion directly by reading the configurations file of Repaint. Unhappily, this breakthrough happened just before our code deadline and we could not train the model as much as we would have liked.

6 Evaluation

The initial results were obtained by utilizing the pre-trained model provided by the authors. AOT-GAN was trained on Places2, an extensive dataset comprising diverse scenes and locations worldwide. Notably, Places2 contains images of architectural and street views.
In contrast, Repaint was pre-trained on ImageNet, a comprehensive dataset with a wide range of image types, with a notable focus on dog-related images.
Tables 1 to 4 showcase the compelling results we obtained. The disparity between the two models is evident.
Repaint tends to generate structured outputs that sometimes lack coherence by hallucinating the desired output. This is particularly evident in Table 1 and Table 2, where it hallucinates, respectively, a yellow salamander texture and a white dog. On the other hand, AOT-GAN produces more blurry outputs that align with the surrounding colors for all tested images. This observation, combined with the fact that AOT-GAN was pre-trained on a more urban dataset, may explain its superior performance at this stage.
Nevertheless, the results from Repaint were also promising due to their sharpness. Additionally, the hallucinations are not problematic in our case, as the desired reconstructions would be always car-shaped. Figure 27 demonstrates our initial intuition with our metrics. AOT-GAN outperforms Repaint for small and big masks.

	AOT small	RP small	AOT big	RP big
l1	0.0906	0.1469	0.1364	0.1997
l2	0.0258	0.0683	0.046	0.086
ssim	0.7409	0.6573	0.531	0.4253
psnr	18.769	15.9563	14.2656	11.3369
lpips	2.3577	2.3411	2.7835	2.9692
mae	0.1316	0.2055	0.1895	0.2625
fid	337	322.99	311.92	316.922

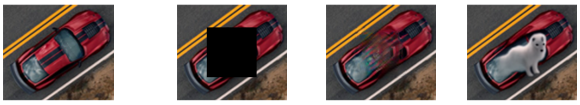
underlined is best score

Figure 27: Evaluation metrics on Vehicle dataset with original pre-trained model

Table 1: Example of midterm result

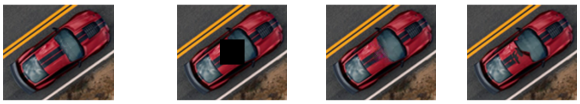
Ground Truth **Masked GT** **AOT** **DDPM**

Table 2: Example of midterm result



Ground Truth **Masked GT** **AOT** **DDPM**

Table 3: Example of midterm result



Ground Truth **Masked GT** **AOT** **DDPM**

Table 4: Example of midterm result

Ground Truth **Masked GT** **AOT** **DDPM**

After this, we decided to train on our own dataset. The results are depicted in Tables 5 to 10. The results in Table 5, Table 6 & Table 7 shows the non-convergence of AOT-GAN after three distinct training. The two first training are two tries of training from scratch and the third one is an attempt of fine-tuning places2.
In the three cases, AOT-GAN didn't converge. This could maybe be explained by the fact that training a GAN model or more precisely two Neural Networks simultaneously is very hard and often leads to such results. In all cases, the results we get are black or mauve inpainting in the masked region which could indicate a problem in image reading.
This didn't occur for Repaint where it did converge as we can see in Table 8, Table 9 & Table 10. We have got some results but unfortunately, the inpainted part is blurry and not as sharp as we expected considering the testing with the pre-trained model. This could be explained by the fact that such a model needs a lot more training with a larger data set with sharp images. The performance metric applied to Repaint confirms our visual intuition that at this stage of training, the model still does not perform as well as with the pre-trained model.

Table 5: AOT-GAN after one week of training results

Image 1 **Image 2**

Table 6: Second AOT-GAN training results

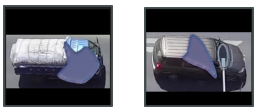


Image 1 **Image 2**

Table 7: AOT-GAN results by fine-tuning the pre-trained model

Image 1 **Image 2**

Table 8: Repaint result example

Masked **Reconstructed**

Table 9: Repaint result example

Masked **Reconstructed**

Table 10: Repaint result example

Masked **Reconstructed**

Table 11: Repaint Performance

Metric	Value
l1	0.0463
l2	0.0123
ssim	0.8226
psnr	22.0886
lpips	1.3556

7 Conclusion

In conclusion, throughout our Bachelor Project, we conducted thorough research, implemented various techniques, and obtained significant results to fulfill our goal of improving car detection by de-occluding car images from drones.

We began by carefully selecting two of the newest inpainting models available in the field. These models were chosen based on their potential to address our specific problem and the availability of their source code. Subsequently, we extensively studied these models to understand their capabilities and suitability for our project. We successfully tested these models on our custom dataset using pre-trained weights, enabling us to evaluate their performance in the context of car de-occlusion.

To enhance the diversity and relevance of our training and evaluation data, we developed a script capable of creating occlusion datasets specifically tailored to vehicles from a UAV's point of view. This script allowed us to generate multiple datasets encompassing a wide range of occlusion scenarios.

To quantitatively assess the performance of the selected models, we established a mean notebook that facilitated the evaluation process using selected image metrics. This notebook served as a valuable tool for comparing and analyzing the results obtained from the models.

The efficient execution of our experiments was ensured by setting up two distinct pipelines on Google Colab and partially on SCITAS. These pipelines enabled us to run and train the Repaint and AOT GAN models effectively, leveraging the available computational resources.

We trained both models on our custom-created dataset. While the results obtained from the AOT GAN model were not convincing due to convergence issues, the outcomes from the Repaint model were particularly encouraging.
Although the generated images still exhibited some blurriness, it is evident that with increased computational capabilities, longer training times, and larger datasets, the Repaint model has the potential to produce impressive results by effectively reconstructing the vehicles within the images.

Furthermore, we have compiled all our scripts and implementations into a comprehensive GitHub repository, providing detailed explanations of our methodology and offering guidance on how to utilize SCITAS and our code for further research and development.

In summary, we hope our Bachelor Project has yielded valuable insights and advancements in the field of car detection by de-occluding car images from drones. We have conducted extensive research, developed a custom occlusion dataset, evaluated the models using robust metrics, and implemented efficient pipelines for training and testing. The results achieved thus far, especially with the DDPM prmomising model, showcase the potential for significant improvements in car de-occlusion.

References

[1] Jiayuan Dong, Liyan Zhang, Hanwang Zhang, and Weichen Liu. "Occlusion-Aware GAN for Face De-Occlusion in the Wild". In: *2020 IEEE International Conference on Multimedia and Expo (ICME)*. 2020, pp. 1–6. DOI: 10.1109/ICME46284.2020.9102788.

[2] Federico Fulgeri, Matteo Fabbri, Stefano Alletto, Simone Calderara, and Rita Cucchiara. *Can Adversarial Networks Hallucinate Occluded People With a Plausible Aspect?* 2019. arXiv: 1901.08097 [cs.CV].

[3] Federico Fulgeri, Matteo Fabbri, Stefano Alletto, Simone Calderara, and Rita Cucchiara. "Can adversarial networks hallucinate occluded people with a plausible aspect?" In: *Computer Vision and Image Understanding* 182 (2019), pp. 71–80.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[6] *Guided-Diffusion, GitHub Repository*. https://github.com/openai/guided-diffusion. Accessed: 2023-04.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[8] *Introduction to Diffusion Models for Image Generation – A Comprehensive Guide*. https://learnopencv.com/image-generation-using-diffusion-models/. Accessed: 2023-06-13.

[9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: 1611.07004 [cs.CV].

[10] Yeong-Joon Ju, Gun-Hee Lee, Jung-Ho Hong, and Seong-Whan Lee. "Complete Face Recovery GAN: Unsupervised Joint Face Rotation and De-Occlusion from a Single-View Image". In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 1173–1183. DOI: 10.1109/WACV51458.2022.00124.

[11] Hongyu Liu, Ziyu Wan, Wei Huang, Yibing Song, Xintong Han, and Jing Liao. *PD-GAN: Probabilistic Diverse GAN for Image Inpainting*. 2021. arXiv: 2105.02201 [cs.CV].

[12] Hongyu Liu, Ziyu Wan, Wei Huang, Yibing Song, Xintong Han, and Jing Liao. "Pd-gan: Probabilistic diverse gan for image inpainting". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9371–9381.

[13] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. "Repaint: Inpainting using denoising diffusion probabilistic models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11461–11471.

[14] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[15] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[16] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. arXiv: 2204.06125 [cs.CV].

[17] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV].

[18] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: 1503.03585 [cs.LG].

[19] *Understanding Diffusion Probabilistic Models (DPMs)*. https://towardsdatascience.com/understanding-diffusion-probabilistic-models-dpms-1940329d6048. Accessed: 2023-06-13.

[20] *VRAI (Vehicle Re-identification for Aerial Image)*. https://opendatalab.com/VRAI. Accessed: 2023-05-11.

[21] Xingqian Xu, Shant Navasardyan, Vahram Tadevosyan, Andranik Sargsyan, Yadong Mu, and Humphrey Shi. "Image Completion with Heterogeneously Filtered Spectral Hints". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 4591–4601.

[22] Xiaosheng Yan, Feigege Wang, Wenxi Liu, Yuanlong Yu, Shengfeng He, and Jia Pan. "Visualizing the invisible: Occluded vehicle segmentation and recovery". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7618–7627.

[23] Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo. *Aggregated Contextual Transformations for High-Resolution Image Inpainting*. 2021. arXiv: 2104.01431 [cs.CV].

[24] Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo. "Aggregated contextual transformations for high-resolution image inpainting". In: *IEEE Transactions on Visualization and Computer Graphics* (2022).

[25] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. "Large scale image completion via co-modulated generative adversarial networks". In: *arXiv preprint arXiv:2103.10428* (2021).

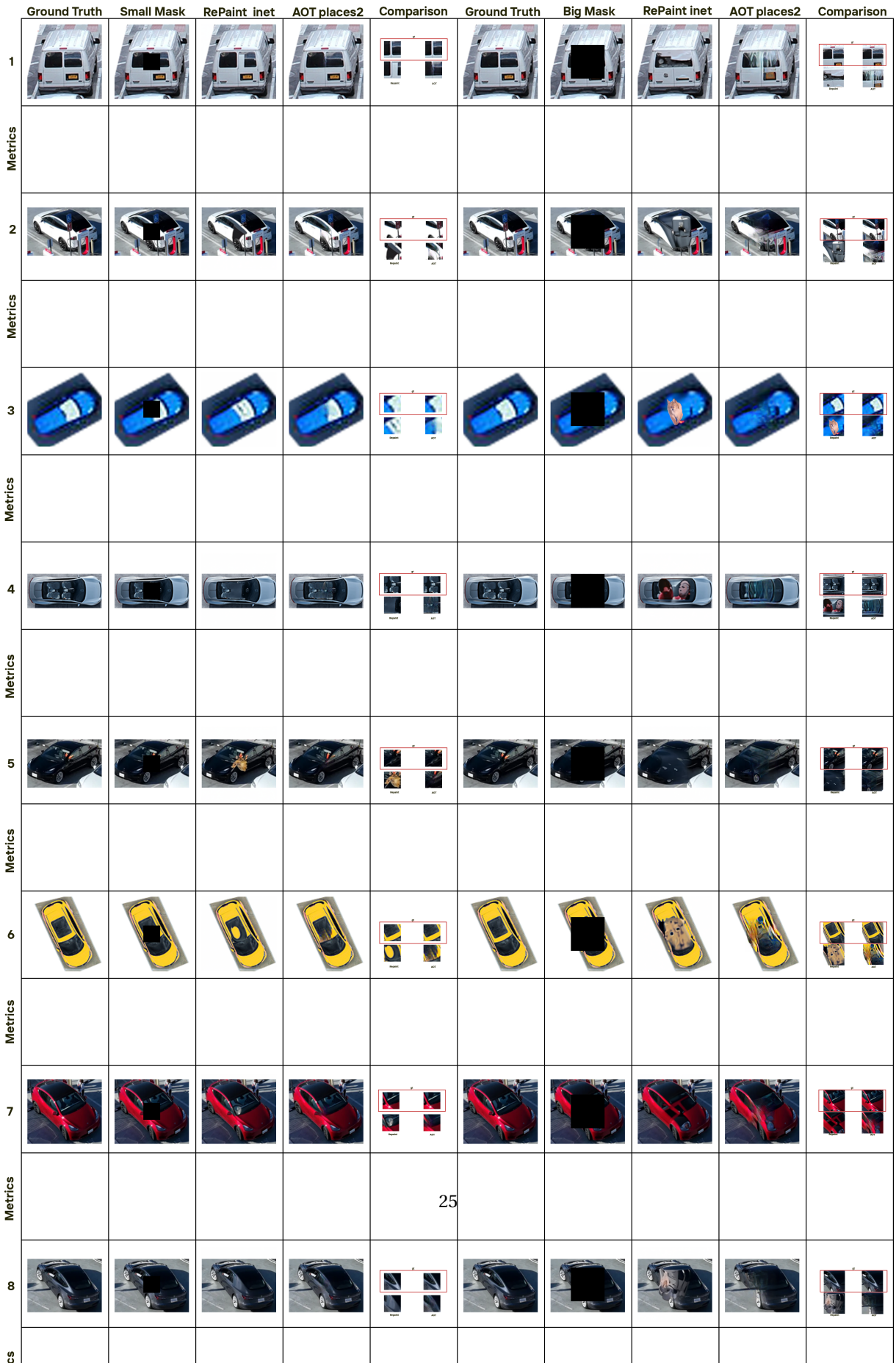
Appendix

Mid-Semester Results

Here are the datasets generated: here

Here are some of the models for AOT-GAN: here

Here are some of the models for Repaint: here


Ground Truth
Small Mask
RePaint inet
AOT places2
Comparison
Ground Truth
Big Mask
RePaint inet
AOT places2
Comparison
1
Metrics
2
Metrics
3
Metrics
4
Metrics
5
Metrics
6
Metrics
7
Metrics
8


9
Metrics
10
Metrics
11
Metrics
12
Metrics
13
Metrics
14
Metrics
15
Metrics
16