Generate a Birds Eye View from Fisheye Cameras using Generative Adversarial Networks

Ricardo Branco Lucas
ricardo.b.lucas@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

January 2021

Abstract

Traditional methods for generating a Bird's Eye View (BEV) are accurate for flat surfaces, but errors are introduced when slopes or protruding objects are present. This thesis aims to investigate different methods using Generative Adversarial Networks (GAN) to generate a BEV from 4 vehicle-mounted fisheye cameras, and to improve on traditional methods. I present two different model approaches, and a data collection and processing procedure. In the first approach, state-of-the-art models (Pix2pixHD, CycleGAN, AttentionGAN) are used to generate the BEV images. In the second, I propose a multi-input model, not only to generate the BEV images, but also to estimate corrected homography matrices. It was not possible to generate a BEV with current state-of-the-art Generative Adversarial Networks, given only fisheye images. Nonetheless, it is shown that without greatly increasing the methods' complexity, similar approaches yield promising results. In particular, Spatial Transformer modules demonstrate a strong potential in aiding a GAN to learn a correct mapping to the BEV.
Keywords: Bird's Eye View, Generative Adversarial Networks, Spatial Transformer Networks, Homography.

1. Introduction

Increasing concern with safety in mobility, together with costumers' interest in connectivity [18], has been shifting automotive manufacturers' efforts towards research and development in information technologies. As a result, many of them have led the field in intelligent technology solutions for safe mobility and ease of driving, in particular in autonomous driving.

Autonomous systems in vehicles need to accurately perceive and understand their surrounding environment in order to perform safe and efficient driving. Cameras are an inexpensive and popular sensor choice for these systems, and if paired with computer vision techniques, they can provide a great amount of information about the environment. Among the different types of cameras, fisheye cameras stand out for achieving high angles of view. This makes them ideal to capture the close proximity environment. By mounting four 180 degrees wide fisheye cameras on each side of the vehicle, it is easy to gain 360 degrees coverage of the surrounding environment, with some overlapping areas between adjacent cameras.

Although cameras can provide information in the 2-dimensional image plane, they lack information about the real-world coordinates. By assuming a flat earth approximation, a perspective transform is often applied to project image pixels to the ground plane. The method of applying this perspective transform is commonly referred to as Inverse Perspective Mapping (IPM) [16]. And this results in a top-down view, also known as Bird's Eye View (BEV). This resulting projection provides information on the real-world coordinates of the elements on the ground plane. As the IPM method assumes a flat earth approximation, the objects protruding out of the ground plane, slopes and irregularities in the road surface will be incorrectly mapped to the BEV image during the IPM projection. Additionally, when using the IPM for the generation of a BEV in multi-camera systems other issues also arise, such as misalignments between the different views, mostly due to accuracy errors in camera calibration.

BEV images can be utilised within tasks such as lane detection [19], road marking detection [17], free space computation [4], and path planning [33]. Since the relevant elements for these tasks are mostly present in the ground plane, the IPM approximation is usually sufficient enough to be used in these applications. An accurate mapping to the BEV could not only improve the performance and reliability of the previously mentioned tasks, but

also allow for tasks where objects like other vehicles, obstacles and pedestrians are accurately detected and located in the real-world coordinates.

With the advancement of deep learning techniques, Generative Adversarial Networks (GANs) [5] have been able to learn the mapping between images of two different domains. By exploring similar methods for the generation of the BEV from 4 fisheye camera images, a more accurate mapping could be achieved.

1.1. Objectives

The main objective of this work is to investigate different methods using Generative Adversarial Networks to generate a BEV from 4 vehicle-mounted fisheye cameras, and evaluate the capabilities of these networks to learn the correct mappings. In particular, we aim to generate a BEV image where all objects are correctly represented without distortions, and the stitching between the different fisheye images is seamless. This work also aims to find an efficient data collection procedure, and evaluate which data is best suited for learning.

Furthermore, with this work we aim to find methods that reduce the need for time-consuming tasks, such as the estimation of parameters necessary to compute the IPM, the alignment and the stitching of the different views.

1.2. Outline

Four sections will follow this introduction. Section 2 includes an overview of related work to the generation of BEV images. In section 3 we present the the approaches taken in this work. The data collection and processing process used to build the necessary datasets is explained, as well as the approaches taken in this work to reach the objectives. Section 4 presents the results produced by the neural network models, which are analysed and discussed. And section 5 presents the conclusion to the thesis and possible future work.

2. Related Work

Several works have taken a geometry-based approach to the generation of the BEV. In [29] and [13] the authors proposed methods to align and stitch the different views in multi-camera systems. In [20], data from a laser range finder is fused with images from the cameras, so that the IPM is not computed in the regions where obstacles are present. And in [11], a mono visual odometry algorithm is used to obtain the vehicle motion, in order to correct the IPM.

In the last few years many works emerged on Image Generation and Translation with the rise of GANs [6]. With the latest works employing attention modules that allow for the translation of images that require holistic and large shape changes [26], [12], but are still restricted to perform aligned appearance transformations. BridgeGAN [32] uses a GAN in conjunction with the IPM to bridge the large gap between the frontal view and the BEV.

In [10], the authors proposed a Spatial Transformer module that transforms the input images to improve performance on classification tasks. In [28] and [30] similar ideas to the Spatial Transformer were used to perform novel view synthesis. The work presented in [3] employs Spatial Transformer modules together with a GAN model to generate a BEV from a single frontal view, resulting in a higher quality BEV compared with the one generated solely through IPM.

The works [32] and [3] are the closest to ours, however these only generate the BEV for a single view. As far as we are aware, our work is the first to use multi-input GAN models to generate a single BEV from 4 different views.

3. Method

Most CNN models take only one image as input. In order to combine images from multiple cameras mounted on a vehicle, two procedures can be taken: use as input the 4 camera images concatenated along their channel dimension; or combine the 4 images in a 2 by 2 grid, as to create a single image. However, for the task at hand, this would result in spatial inconsistency between the input and the output images, because of the way convolutional layers operate (information in particular locations of the input is mapped to approximately the same location of the output). Therefore, other solutions had to be found. The first solution was to use popular GAN models, which are all single-input, and use as input a pre-processed BEV image from the 4 camera images by using the IPM technique. The second was to create a multi-input GAN model, with integrated Spatial Transformer modules to ensure spatial consistency. In this section, we will present these two different GAN-based approaches. But before, the data collection and processing procedures used to create a dataset are explained in detail.

3.1. Data Collection

The first step in creating a dataset from scratch is the data collection. In order to collect the necessary images we used a Volvo XC90 test car and a DJI Mavic drone. The Volvo test car was equipped with four fisheye cameras (which record synchronously), one in the front badge, one in each of the side-mirrors and one in the tailgate, meaning that each camera is pointing in a different direction and full 360 degrees coverage can be achieved. The DJI drone was equipped with a camera mounted on a stabilisation gimbal, which was set to point directly down.

The recording procedure involved driving the test

car at relatively slow speeds i.e., less than 30 km/h, on various parking lots, and having the drone following the car directly above, with the camera pointed straight down, at an altitude relative to the ground of around 50 meters. The position of the drone had to be controlled manually, which meant that it wasn't always directly over the test car nor at the same altitude. That led to the need of rectifying the drone frames in a later processing stage.

3.2. Data Processing

The goal of processing the drone frames was to correct each one: to centre the test car in the frame; align the test car symmetry line with the vertical axis of the frame; crop the image to keep the scale of the image fixed in relation to the test car; and to pair the frames with the images from the test car fisheye cameras. To achieve this, an algorithm using the OpenCV library was developed. The images resulting from the first three correction steps mentioned, can be visualised in Fig. 1.

The algorithm to rectify the drone images and synchronise them with the test car images can be portioned in 3 main stages: parameter extraction by manual point and Region of Interest (ROI) selection; tracking with optical flow and rectifying; synchronising and saving.

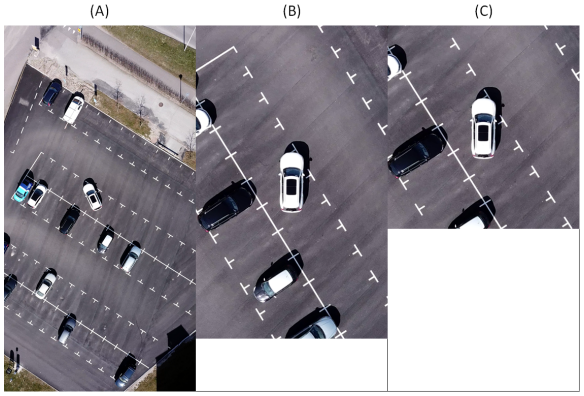

Figure 1: Images of the same frame after different rectifying steps. (A) - Original frame; (B) - Frame after alignment and centring; (C)- Frame after cropping.

The first frame that is fed to the algorithm will follow the first stage, parameter extraction by manual point and ROI selection. Here the user has to identify and manually select two points on the frame that correspond to the test car extremities that belong to its symmetry line. These points will then be used to calculate the angle γ relative to the vertical axis, the length (in pixels) l and the midpoint pixel coordinates C between the points. In this process, given two points in the image coordinate system, $A(x_A,\ y_A)$ and $B(x_B,\ y_B)$, that form the line segment AB, represented in Fig. 2, the parameters γ, l and C can be easily calculated:

$$l = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2} \tag{1}$$

$$C = (\frac{x_A + x_B}{2},\ \frac{y_A + y_B}{2}) \tag{2}$$

$$(x_B,\ y_B) = (x_A + l\sin(\gamma),\ y_A - l\cos(\gamma)) \tag{3}$$

$$\gamma = \arctan2(\frac{x_B - x_A}{y_A - y_B}) \tag{4}$$

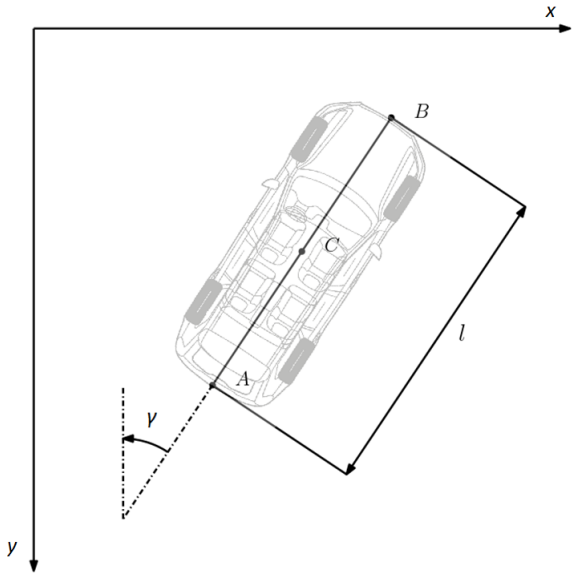

Figure 2: Definition of the image coordinate system, representation of the points A and B, and parameters γ, l and C in relation to the test car.

By using this process to calculate γ, l and C, from a set of two known points, we can easily perform the necessary correction for any given frame.

These first calculated parameters, from the manually selected points will serve to rectify the first frame and to pre-rectify all of the following frames. This means that now only the changes from this first rectified frame need to be tracked, and that can be done automatically. But first, in order to track the changes in orientation, position and scale of the test car relative to the first frame, a new set of two good points to track needs to be selected. Here the user will not select the points, but instead define two ROIs over the test car. The ShiTomasi Corner Detection algorithm [22] will be ran on the ROIs and select the best points for tracking.

After the first frame, all the frames will follow the procedures in stage two. Where optical flow

is calculated, by the Lucas–Kanade method [15], on the current frame with the coordinates from the previous set of tracking points, and output a new set of tracking points. From this new set, new parameters, reflecting the changes from the previous frame, are calculated and then full rectification of the frame can be completed.

At stage three the drone frames are matched with a set of 4 fisheye camera images that correspond to the same scene. This is done by resorting to timestamps from the images and checking for each fisheye images set, the drone image that is closest in time to it, and pairing them together.

The images from the test car fisheye cameras were also pre-processed into two more classes of images, which we will refer to as: Undistorted and Classic BEV. In the Undistorted set, the distortion caused by the fisheye camera lenses is removed, and in the Classic BEV image, the images from each individual camera are projected onto the ground plane (using IPM), stitched together and blended to form a single BEV image, as seen in Fig. 3. This preprocessing is outside of the scope of this thesis, as these processes were already developed and implemented by Volvo Cars.

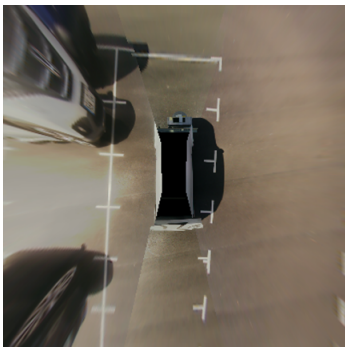

Figure 3: IPM generated BEV, or Classic BEV

3.3. Single-Input Model

As first approach, we propose the use of popular single-input GAN models. Due to the large gap between the fisheye camera views and the BEV, an pre-generated BEV, through IPM, will instead be used as input to the networks. This will be the Classic BEV, mentioned in the previous section, as it covers the same spatial region as the desired target output image.

A study using several models was conducted to determine if the current state-of-the-art GAN models could correct the distortions and errors introduced by the IPM, camera calibration and stitching when generating the Classic BEV images, and at the same time retain the capacity to generate the ground plane details and improve on the overall image quality.

Four models were trained: Pix2pix [9], Pix2pixHD [27], CycleGAN [31] and AttentionGAN [26]. Each of these models fall into one of three different categories. Pix2pix and Pix2pixHD are supervised GAN models, CycleGAN is an unsupervised GAN model and AttentionGAN is a unsupervised GAN model that uses attention. With this, we aimed to evaluate which type of GAN model is best suited to the problem in question and to the given data.

Each model was either implemented according to the respective paper or trained using the official publicly available implementation. For training we used all the default hyperparameters and no alterations to the architecture or loss functions of the models were done.

3.4. Multi-Input Model

For the second approach, we propose a multi-input model, which takes as input 4 Undistorted images from the 4 fisheye cameras, and outputs a BEV image. Drawing inspiration from [3], we integrate Spatial Transformer modules to ensure spatial consistency. And due to its simplicity and extensive use in image-to-image translation tasks we also based our core architecture on the popular Pix2pixHD model [27].

3.4.1 Generator

While our generator follows the traditional downsample-bottleneck-upsample architecture, in order to build an architecture for multiple inputs and one output, our model has separate input paths for each input image. Each path is composed of a series of downslampling layers and at least a Spatial Transformer Residual (STRes) block [3], that contains a Spatial Transformer module [10] followed by at least a single ResNet block [7]. The intuition behind the use of the STRes block, was "that the slight blurring that occurs as a result of each perspective transformation is restored by the ResNet block that follows it" [3]. The Spatial Transformer module can be placed at any location in the network, therefore several locations were considered, as seen in generator diagram presented in Fig. 4. These locations will be later discussed.

The features from each input path are then concatenated into a single feature map, which is then convoluted through a modified ResNet block to reduce the number of channels of the output feature map. From here the feature maps go through a series of upsampling layers until the network ends with the tanh() activation function.

3.4.2 The Spatial Transformer Module

The goal of using the Spatial Transformer (ST) module is to ensure spatial consistency between the feature maps and the respective ground truth. The

ST module tries to ensure this spatial consistency by learning transformations that are approximate to a projection of the image pixels from the Undistorted images to the ground plane. A ST is a fully differential module that can be easily integrated into any existing model and is composed by three components: the Localisation Network, the Grid Generator and the Sampler.

The Localisation Network task is to estimate transformation matrix parameters θ_{estim}, from an input volume I. It is represented, in our case, by a simple Convolutional Neural Network (CNN), but it could also be any state-of-the-art classification network, for example EfficientNet [25], as long as we regress to the right number of parameters needed for the transformation we want to do. In our case, we want to estimate an homography matrix, therefore we regress the CNN to 8 parameters. The Grid Generator generates a grid of coordinates in the input I, corresponding to each pixel from the output O. The Sampler uses the parameters of the transformation and applies it to the input I, using bi-linear interpolation, resulting in an output volume O.

A ST module is not trivial to train for large perspective transformations. To stabilise its training, we define an initial transformation matrix, θ_{init}, with an approximate parameterisation of the desired homography matrix, and multiply it with θ_{estim}, which is the matrix estimated by the ST, to calculate a final homography, θ. The following equation represents the matrix multiplication:

$$\theta = \theta_{estim} \cdot \theta_{init} \tag{5}$$

An θ_{init} was estimated for each fisheye camera. This estimation was done by selecting four approximately corresponding points on a matching pair of images, composed of an Undistorted image and a Drone image (we assume that the Drone images plane is a good approximation to the ground plane). Using these points, it is possible to arrive at a homography estimation. The same θ_{init} was used for each model.

This way, θ_{estim} is actually learning a perturbation to correct θ_{init}. Which means that θ_{estim} needs to take the form of an identity matrix at initialisation. In order to get this initialisation, the weights of the last layer of the Localisation Network are initialised with zeros and the bias with the identity matrix.

3.4.3 Discriminator

The discriminator used is the same as in [27]. It is a multi-scale discriminator, meaning that it is actually composed of 3 discriminators that have the same architecture. Each discriminator will be trained for a different scale, "specifically, we downsample the real and synthesised high-resolution images by a factor of 2 and 4 to create an image pyramid of 3 scales" [27]. This downsampling is achieved through a simple average pooling operation, with stride of 2. The discriminators have a fully convolutional architecture [14], which is also commonly referred to as "PatchGAN".

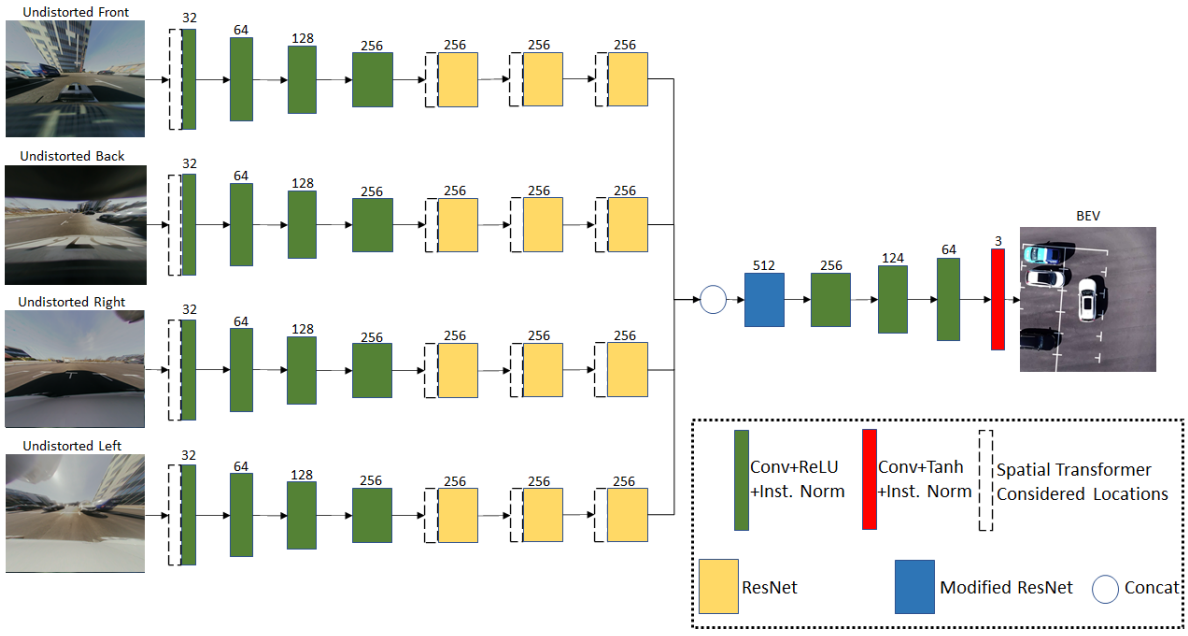

Figure 4: Architecture of the multi-input model generator network. The number of filters is represented for each block.

3.4.4 Losses

Our objective function is the same as in [3], which in term stems from [27]. It is composed of an adversarial loss $\mathcal{L}_{GAN}$, a feature matching loss $\mathcal{L}_{FM}$ and a perceptual loss $\mathcal{L}_{VGG}$.

The adversarial loss $\mathcal{L}_{GAN}$, is defined as:

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}_{(x,y)\sim P_{data}(x,y)}[logD(y,x)] + \mathbb{E}_{y\sim P_{data}(y)}[log(1-logD(y,G(y)))] \quad (6)$$

where $\mathbb{E}$ denotes the expected value, G the generator and D the discriminator.

By using the multi-scale discriminators, D_1, D_2, D_3, together with the generator, G, the learning objective becomes:

In the perceptual loss, Eq. 7, a VVG16 [23] pretrained network is used. Feature maps from intermediate layers of the network are extracted for both the real, and generated images, and the difference between them is calculated using the L_1-distance. This encourages the generated image and real image to have similar low and high-level feature representations extracted from loss network.

$$\mathcal{L}_{VGG}(G) = \mathbb{E}_{(x,y)\sim P_{data}(x,y)} \sum_{i=1}^{n} \frac{1}{w_i} ||VGG^{(i)}(x) - VGG^{(i)}(G(y))|| \quad (7)$$

Here, y is the input/label image and x is the real/target image. The weight variable, $w_i = 2^{n-i}$, is used to scale the importance of each layer, i, used in the loss, both in Eq. 7 and 8, where n is the number of intermediate layers utilised, and it was set at 4.

The feature matching loss, Eq. 8, is related to perceptual losses, but instead of using an auxiliary network, the feature maps are directly extracted from intermediate layers of the discriminators.

$$\mathcal{L}_{FM}(G,D_j) = \mathbb{E}_{(x,y)\sim P_{data}(x,y)} \sum_{i=1}^{n} \frac{1}{w_i} ||D_j^{(i)}(y,x) - D_j^{(i)}(y,G(y))|| \quad (8)$$

By using the multi-scale discriminators, D_1, D_2, D_3, together with the generator, G, the full learning objective becomes:

$$\mathcal{L}_{total} = \min_G \left(\left(\max_{D_1,D_2,D_3} \sum_{j=1,2,3} \mathcal{L}_{GAN}(G,D_j) \right) + \lambda_{FM} \sum_{j=1,2,3} \mathcal{L}_{FM}(G,D_j) + \lambda_{VGG}\mathcal{L}_{VGG}(G) \right) \quad (9)$$

where λ_{FM} and λ_{VGG} are empirically set at 5 and 2, respectively.

3.4.5 Architecture Variations

Three different variations of the generator architecture were studied. In each different architecture, the variations consisted of modifying where the spatial transformer modules were located. In order to maintain the same learning capacity across the different models, the number of ResNet blocks was kept constant.

In the first variation (STN1-Encoder) the ST module is placed at the start of the network, thus transforming the input image before any convolution operation. Therefore, the input to the rest of the network will already be aligned with the ground truth, but it misses features that are only present in the Undistorted image. In the second variation (STN1-Bottleneck), we place the ST module at the start of the bottleneck, to perform the appropriate transformation of the feature maps encoded from the input images. For the third variation (STN3-Bottleneck), we followed the approach in [3]. This means placing a ST module before each ResNet block, in the bottleneck of the network, to perform incremental transformations to the ground plane, Fig. 5.

Figure 5: Visualisation of 3 incremental transformations applied to a Undistorted frontal image.

3.4.6 Implementation Details

Each model was trained for 250000 iterations with a batch size of 1, which was enough for the results to converge. Usually these models should be trained for more iterations but overfitting was already being observed before 250000 iterations. The Learning Rate is initially set at 0.0001, after 125000 iterations a linear scheduler decayed the learning rate to 0, at 250000 iterations. The input image size was set at 512x512.

3.5. Evaluation

Evaluation of generated images from GANs is a difficult problem [2]. With many works relying in qualitative perceptual studies and quantitative measures. Due to time constraints we did not perform a qualitative perceptual study, and instead relied on two of the most popular quantitative measures, the Frechet-Inception Distance (FID) and the Kernel-Inception Distance (KID).

The Frechet Inception Distance (FID) [8] score

was proposed as an improvement over the Inception Score [21]. As with the Inception Score, the FID score uses the Inception v3 classification model [24] to capture features from an input image. The Inception v3 model is then run for two collections, one containing the real images, and another containing the generated images. The features of each collection are then summarised as a multi-variate Gaussian. Then the distance between these two multivariate Gaussian distributions is calculated using the Frechet distance. A lower FID score indicates that generated images more closely match the statistical properties of real images.

The recently proposed Kernel-Inception Distance (KID) [1], computes the squared Maximum Mean Discrepancy between the feature representations of real and generated images. The feature representations are again extracted from the Inception v3 model. In contrast to the FID score, KID has an unbiased estimator, which makes it more reliable, especially when there are fewer test images than the dimensionality of the inception features. As it is with FID, a lower KID score indicates that generated images more closely match the statistical properties of real images.

To run these measures an evaluation dataset was defined. This dataset contains 159 images, all taken from the same isolated route that was not seen by the models during training.

Besides evaluating the different GAN models, we also evaluate how well the models could learn to generate a BEV given different sets of data. For this, 3 different training datasets were created. Each dataset was created in order to present a different degree of difficulty. Dataset 3 presents the scenarios that should be easiest for the model to learn, while Dataset 1 presents the scenarios that should be the most challenging, as it contains features that are harder to learn, such as poles and overhead trees, that are less present in Dataset 1 and 2.

In order to evaluate how the different datasets affect the models learning of the BEV, the same model, Pix2pixHD was trained for 250000 iterations for each of the three different datasets.

4. Results

4.1. Dataset

The results from the dataset evaluation, described in the previous section, where the same model was trained with different datasets, are presented in the following table.

	FID	**KID**
Dataset 1	1.587837524	24.40750003
Dataset 2	1.8459935	30.11656404
Dataset 3	1.964971619	32.08196163

Table 1: FID and KID score for the different datasets trained on the Pix2pixHD model.

From table 1, we note that the model trained on Dataset 1, which should have been the most challenging dataset, performed the best. While the model trained on Dataset 3, which should have been the least challenging dataset, performed the worst. The explanation that we find for these results is that all models overfitted the data. With Dataset 3, which has the least amount of images thus overfitting the most and performing worse. Dataset 2 and Dataset 1 followed with better results as these contained more data and overfitted less.

Unfortunately, the only conclusion that we can arise from this study, is that the datasets didn't contain enough images to train these models without causing overfitting. This is, the models will learn to translate the images in the training set well, but have difficulty to generalise with unseen data.

4.2. Sinlge-Input Model

When evaluating the results from the first approach, it is possible to take several findings. From table 2, where the evaluation scores are presented, it is noted that Pix2pixHD obtained the best score by a large margin, outperforming all the other models in both the ground plane details and on the generation of other cars, being able to remove most of the distortions from the other cars and mapping then to correct position, and represent well defined road details (we must also note that Pix2pixHD was trained with a larger size image). Pix2pix performed the worse, generating very blurry images. CycleGAN and AttentionGAN obtained similar scores, with the latter obtaining a marginally better score.

	FID	**KID**
Pix2pix	4.264246216	78.70054245
Pix2pixHD	1.425966034	22.57144451
CycleGAN	3.478949585	64.3399477
AttentionGAN	3.465979309	64.1622901

Table 2: FID and KID score for the different single-input models trained on Dataset 1.

4.3. Multi-Input Model

From the model variations described in section 3.4.5, all were able to generate a BEV from the 4 Undistorted images. However, the overall quality of the generated images remained relatively low. We believe that this is mostly due to model overfitting, which is caused by the reduced amount of training

Figure 6: Samples from 5 different scenes, (A) to (E), generated by the three different model variations (STN3-Bottleneck, STN1-Bottleneck, STN1-Encoder), and the respective ground truth.

data and the lack of data augmentation. Another reason could be the size of the input images, that is 4 times smaller than the images that are used to generate the Classic BEV. Nevertheless, it is still possible to compare the different model variations and consequently, take away conclusions.

From the generated images, it is possible to assess two main characteristics of the generated images: how well the ground plane and road markings details are represented, and how the distortions from the other cars are corrected.

When observing the ground plane details in Fig. 6, the STN1-Encoder variation was the one that was able to generate them with more similarity to the ground truth, this is particularly observable at scene (C), where most of the road markings at the bottom and top-left corner of the image are generated, while for the other variations it is harder to make out those details. Also, in scene (A), on the left side of the image the road details are again more accurately portrayed on the variation STN1-Encoder. Comparing STN1-Bottleneck and STN3-Bottleneck, it is possible to observe that the latter is able to reproduce details better, as seen in scene (A) and (B), where the small dark drain on the left side of the image is faintly generated with STN3-Bottleneck, while it is not generated at all with STN1-Bottleneck. We argue that this difference is due to the incremental transformation performed on STN3-Bottleneck.

When looking at how the other cars are generated, all variations generate similar results, although it is noticeable that the STN1-Encoder model variation generated cars are slightly more blurry and incorrectly shaped, compared with the other variations.

We also verified that the ST modules are generally able to correct the initial homography to a transformation that will align the images more closely with the ground truth. Although, when training the STN1-Encoder model, sometimes the ST module would completely fail to learn a reasonable transformation, which would happen after an weight update deviated the estimated transformation by a large amount from the correct one. Nonetheless, this could be solved by decreasing the learning rate, which would stabilise the learning process.

	FID	KID
STN3-Bottleneck	2.10153244	36.19625866
STN1-Bottleneck	2.234797058	37.92501092
STN1-Encoder	2.124474487	36.25893891

Table 3: FID and KID score from our multi-input model variations.

From the scores in table 3, STN3-Bottleneck performed the best and STN1-Bottleneck performed the worse. We interpret that the better score for STN3-Bottleneck is due to a combination of good ground plane details, and better representation of other cars. While, the STN1-Bottleneck model was penalised for the lack of ground plane details and the STN1-Encoder model was favoured for its better representability of the ground plane details.

5. Conclusions

Generating a Bird's Eye View (BEV) with current state-of-the-art Generative Adversarial Networks (GANs), given only fisheye images proved to not be yet possible. Nonetheless, it is shown that without greatly increasing the methods complexity, similar approaches yield promising results. In particular, Spatial Transformer (ST) modules demonstrate a strong potential in aiding a GAN to learn a correct mapping to the BEV.

Of the two proposed methods, both the single-input and the multi-input approaches presented viable options and produced similar results. The single-input approach is simpler in its implementation, but its potential is limited by the information contained a BEV generated through traditional methods. On the other hand, the multi-input models using ST modules are harder to train, but their possibilities are still very much open for future experimentation.

By training paired and unpaired GAN models, it is observed, that although paired models result in overall higher quality and better metric scores, unpaired models demonstrate higher accuracy at generating ground plane details. Our multi-input model was only implemented in a paired configuration, consequently, we believe that an unpaired/unsupervised implementation should also be explored.

Our multi-input models showed that is possible to generate a BEV, with reasonable quality, from 4 undistorted images, by resorting to ST modules that ensure spatial consistency. Of the model variations, the STN1-Encoder was able to generate better ground plane details, although for this model the ST module was more unstable while training. On the other hand, the STN3-Bottleneck better corrected the other cars distortions and was easier to train. Therefore, it is recommended that a model combining both variations is explored. Such model would contain ST modules both in the encoder and in the bottleneck of the generator, and follow a U-Net style architecture [**?**] with skip connections. Also, only simple CNN architectures were used for the localization networks in the ST modules, which proved to be sufficient. Nonetheless, we believe that other architectures are to be explored in order to stabilize the training process.

This study also stresses how crucial a large quantity of data is to avoid model overfitting. While the recorded 6276 samples proved to be enough to conduct comparative studies, a dataset containing at least 15000 samples is recommended to train a robust model, capable of inferring in parking lot scenarios.

References

[1] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[2] A. Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[3] T. Bruls, H. Porav, L. Kunze, and P. Newman. The right (angled) perspective: Improving the understanding of road scenes using boosted inverse perspective mapping.

[4] P. Cerri and P. Grisleri. Free space detection on highways using time correlation between stabilized sub-pixel precision ipm images. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2223–2228. IEEE, 2005.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. Reporter: arXiv:1406.2661 [cs, stat].

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition.

[8] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks.

[10] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks.

[11] J. Jeong and A. Kim. Adaptive inverse perspective mapping for lane map generation with slam. In *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 38–41, 2016.

[12] J. Kim, M. Kim, H. Kang, and K. Lee. U-GAT-IT: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation.

[13] Y. Liu and B. Zhang. Photometric alignment for surround view camera system. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1827–1831. ISSN: 2381-8549.

[14] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation.

[15] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[16] H. Mallot, H. Bülthoff, J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological cybernetics*, 64:177–85, 02 1991.

[17] B. Mathibela, P. Newman, and I. Posner. Reading the road: road marking classification and interpretation. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):2072–2081, 2015.

[18] N. Müller, D. Mohr, A. Krieg, P. Gao, H.-W. Kaas, A. Krieger, and R. Hensley. The road to 2020 and beyond: What's driving the global automotive industry?

[19] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018.

[20] M. Oliveira, V. Santos, and A. D. Sappa. Multimodal inverse perspective mapping. 24:108–121.

[21] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[22] J. Shi et al. Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, pages 593–600. IEEE, 1994.

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[25] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[26] H. Tang, H. Liu, D. Xu, P. H. S. Torr, and N. Sebe. AttentionGAN: Unpaired image-to-image translation using attention-guided generative adversarial networks.

[27] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs.

[28] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision.

[29] B. Zhang, V. Appia, I. Pekkucuksen, Y. Liu, A. U. Batur, P. Shastry, S. Liu, S. Sivasankaran, and K. Chitnis. A surround view camera solution for embedded systems. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 676–681. ISSN: 2160-7516.

[30] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow.

[31] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks.

[32] X. Zhu, Z. Yin, J. Shi, H. Li, and D. Lin. Generative adversarial frontal view to bird view synthesis. In *2018 International Conference on 3D Vision (3DV)*, pages 454–463. ISSN: 2378-3826.

[33] A. Zyner, S. Worrall, and E. Nebot. Naturalistic driver intention and path prediction using recurrent neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1584–1594, 2019.