Improving depth estimation in an automated privacy-preserving video processing system

Author: Zina Al-jibouri
s4360532

Internal Supervisor: Prof. T.M. Heskes

External Supervisor: J. Snijder (InfoSupport B.V.)

April 2020
Abstract

This master thesis is written in collaboration with Info Support B.V. for one of their clients, and explores the use of a conditional Generative Adversarial Network for the estimation of depth from monocular rgb images. The main research question stated is: Can a state-of-the-art neural network be trained to accurately estimate relative depth in a crowded scene from monocular recordings of a single uncalibrated camera? This question was answered through three sub-questions, which addressed the addition of biological cues, temporal information, and transfer learning from hyper-realistic video game data. The model trained was improved through addition of biological cues, temporal information and transfer learning, but was not accurate enough for the intended application of depth estimation from monocular recordings in a crowded scene.
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Introduction</strong></td>
</tr>
<tr>
<td><strong>2 Background information</strong></td>
</tr>
<tr>
<td>2.1 Object identification and tracking</td>
</tr>
<tr>
<td>2.1.1 Object detection</td>
</tr>
<tr>
<td>2.1.2 Object tracking</td>
</tr>
<tr>
<td>2.2 Depth Estimation</td>
</tr>
<tr>
<td>2.3 Generative adversarial networks</td>
</tr>
<tr>
<td>2.4 Data collection</td>
</tr>
<tr>
<td>2.5 Research questions and approach</td>
</tr>
<tr>
<td><strong>3 Methodology</strong></td>
</tr>
<tr>
<td>3.1 A study of biological cues of depth perception</td>
</tr>
<tr>
<td>3.2 Experiment 1: Generating depth maps from monocular images with a conditional Generative Adversarial Network</td>
</tr>
<tr>
<td>3.2.1 Data analysis and preprocessing</td>
</tr>
<tr>
<td>3.2.2 Model and training procedure</td>
</tr>
<tr>
<td>3.3 Experiment 2: Generating depth maps from monocular image with hyperrealistic video game data from GTA V</td>
</tr>
<tr>
<td>3.3.1 Data collection and preprocessing</td>
</tr>
<tr>
<td>3.3.2 Model and training procedure</td>
</tr>
<tr>
<td>3.4 Experiment 3: Incorporating temporal information in a model for depth estimation from monocular images with a conditional Generative Adversarial Network</td>
</tr>
<tr>
<td>3.4.1 Data analysis and preprocessing</td>
</tr>
<tr>
<td>3.4.2 Model and training procedure</td>
</tr>
<tr>
<td><strong>4 Results</strong></td>
</tr>
<tr>
<td>4.1 Experiment 1: Generating depth maps from monocular images with a conditional Generative Adversarial Network</td>
</tr>
<tr>
<td>4.2 Experiment 2: Generating depth maps from monocular image with hyperrealistic video game data from GTA V</td>
</tr>
<tr>
<td>4.2.1 Application of Region of Interest masks</td>
</tr>
<tr>
<td>4.2.2 Transfer learning with a cGAN</td>
</tr>
<tr>
<td>4.3 Experiment 3: Incorporating temporal information in a model for depth estimation from monocular images with a conditional Generative Adversarial Network</td>
</tr>
<tr>
<td><strong>5 Discussion</strong></td>
</tr>
<tr>
<td>5.1 Experiment 1: Generating depth maps from monocular images with a conditional Generative Adversarial Network</td>
</tr>
<tr>
<td>5.2 Experiment 2: Generating depth maps from monocular image with hyperrealistic video game data from GTA V</td>
</tr>
<tr>
<td>5.3 Experiment 3: Incorporating temporal information in a model for depth estimation from monocular images with a conditional Generative Adversarial Network</td>
</tr>
<tr>
<td>5.4 General Discussion</td>
</tr>
<tr>
<td>5.5 Suggestions for further research</td>
</tr>
<tr>
<td><strong>6 Conclusion</strong></td>
</tr>
</tbody>
</table>


1 Introduction

This project is done in collaboration with Radboud University Nijmegen and Info Support B.V.. Info Support B.V. has received the request for a privacy preserving video recording system from one of their clients, namely the organisation of Paaspop, a music festival in the Netherlands. Paaspop utilizes a large-scale CCTV system for the security of their visitors throughout the festival. They wish to store their recordings in compliance with privacy laws and in such a way that they are still able to analyze traffic and pedestrian patterns outside the festival terrain. This could help them improve their security and logistic processes at future festival events. Existing footage that needs to be archived has been recorded by single cameras, and future footage will most likely also be recorded with single cameras. By creating a general and extensible system, a developed method would not only be applicable to video data from Paaspop, but also to other public surveillance systems.

Video surveillance systems have become more prevalent in recent decades. Only in London, the average citizen is caught on closed circuit television (CCTV) about 300 times a day [36]. This is not surprising, as image quality has improved while the technology costs have gone down. Furthermore, developments in the field of artificial intelligence have increased the usability of surveillance data. Although video surveillance technology plays an important role in the safeguarding of public safety, this trend has raised a number of concerns, especially with regards to privacy. Cavailaro, for example, discusses how locations of people can be tracked based on surveillance data from different cameras in public spaces, allowing owners of cameras or even governments to spy on specific (public) figures [3]. Koskela argues that persistent surveillance does not only lead to privacy issues, but also affects an individual’s behaviour [36]. Surveillance can thereby threaten an individual’s autonomy, as they may not engage in harmless behaviours that are beneficial to them [54].

There are measures that can be taken to reduce the impact of video surveillance systems on the privacy of individuals. One such measure can be to automatically detect individuals and attempt to make them unidentifiable [11]. This is often done by applying facial detection software and subsequently adding noise to blur faces out. This method is perhaps the most common one, and can be found in well-known applications such as Google Street View [20]. While this method may seem effective, it does not satisfy all privacy issues. Firstly, if the software responsible for detecting faces is not 100 percent accurate, privacy is not guaranteed. This is particularly the case for crowded scenes in which faces are partially obfuscated and the software is therefore unable to detect them. This means that entities utilizing this type of software remain liable for privacy infringement. Secondly, this approach is often reversible, which means that the recordings could still be misused by authorized personnel [11, 3]. Furthermore, although this method would prevent absolute identification when a 100 percent is achieved, individuals could still be recognized from their clothing or environment, allowing for relative identification [69].

Another method that has often been suggested is to encode recorded data in such a way that none of the individuals or objects are recognizable. This method allows for storing of the altered recorded data, and could be reversed with e.g. a court order [91]. However, this does not solve the aforementioned issue of misuse by authorized personnel, or the lack of guaranteed privacy due to reversibility. Therefore, to address these privacy concerns with video surveillance systems, a method is needed that is either 100 percent accurate, or irreversible while preserving useful information in the recorded data.

Info Support B.V. has attempted to create such a solution, which is summarized in the work of Conde Moreno [10]. The proposed system describes a method for converting each frame of a 2D recording of a scene to an anonymous 3D representation of that scene.
This means that each object of interest is converted to a generic 3D model in its corresponding location in the hyperplane. This way, video data can be stored while privacy of individuals can be guaranteed. The following requirements were suggested for such a system:

- Capable of detecting the 2D location of objects of relevance every single frame of a recording.
- For each of these frames provide an estimation of the location of all detected objects within the 3D space of the scene.
- Able to provide a general reconstruction with an anonymized representation of the objects of relevance.

The resulting pipeline that is in place at Info Support B.V. consists of the following steps: 1) detection and segmentation of the objects of interest (such as people), 2) camera self-calibration to obtain the camera parameters needed for conversion from a 2D scene to a 3D scene, 3) depth estimation to calculate the approximate distance of objects to the camera viewpoint, 4) estimation of 3D object locations (based on step 2 and 3), 5) object tracking across frames to improve the above mentioned processes, 6) 3D reconstruction of the scene using generic 3D models. A schematic of this pipeline is also depicted in figure 1.

The method proposed by Conde Moreno shows promising results. The depth estimation in the system was however not accurate enough, which led to poor determination of locations in 3D space (step 4). Developing an algorithm that is capable of 3D depth estimation in video recordings with a single camera is not only relevant to the application in surveillance software, but also has important cost-effective applications within robotics, scene understandings and 3D reconstructions. Additionally, much of the existing footage has already been recorded with a single camera and needs to be archived anonymously. This thesis therefore aims to extend and improve upon the work of Conde Moreno in collaboration with Info Support B.V.. Specifically, it will focus on the improvement of depth estimation from monocular images of a single camera (step 3) by incorporating biological cues into the system and using various neural networks to generate depth maps. To achieve a comprehensive work, literature will be discussed separately for each section of the pipeline. After this, the methodology of this thesis will be elaborated, followed by an overview of the results and a discussion.

Figure 1: General pipeline of video anonymization software proposed by Conde Moreno. The system receives video frames as input and then outputs the estimated locations of detected objects on the ground plane and their class label for each of the frames. Taken from [10].
2 Background information

In this section some essential background information will be outlined. Background on the relevant aspects of the overall anonymous 3D scene reconstruction pipeline will be discussed, as well as on some novel methods that will be applied in the project.

2.1 Object identification and tracking

To be able to anonymize sensitive data in video recordings, it is first necessary to be able to identify the relevant aspects of the data, e.g. detecting and recognizing objects of interest. There are broadly two different approaches to object recognition, namely through image segmentation or semantic segmentation. The prior method involves creating bounding boxes around the objects of interest, while the latter provides a semantic label to every single pixel in the image [87, 85]. Note that in the case of image segmentation, semantic labels are not necessarily required. This means that objects could be detected without being recognized. For the intended application object recognition is essential, since the goal is to reconstruct the 3D scene with the position of objects of interest. Furthermore, temporal information should be taken into account. This could be useful as object locations are correlated across frames. In the following section, we will discuss a review of existing techniques for image segmentation and semantic segmentation, followed by a review of existing techniques for object tracking.

2.1.1 Object detection

Traditionally, visual object detection is achieved with point detectors or background subtraction. Nowadays, the state-of-the-art technique for both image segmentation and semantic segmentation is to make use of Convolutional Neural Networks (CNNs) [38]. CNNs are a class of neural networks that are inspired by the organisation of neurons. They share weights and are more computationally efficient than traditional neural networks. This also makes them less prone to overfitting. CNNs are excellent feature extractors, which is why they are able to map more complex patterns than traditional neural networks. This means that in particular for image data, it is advisable to continue training a pretrained CNN instead of training a CNN from scratch. This way the network has already captured higher level image features. This process is known as transfer learning and has been shown to significantly improve performance of CNNs [50].

In general, three categories of CNNs can be defined: sliding window-based detectors, region proposal-based detectors, and single shot detectors [10]. A problem with sliding-window based detectors such as Overfeat [70] and DPM [16] is that they do not take the context of the image into account. They can also be considered a brute force approach, in which windows of varied sizes and aspect ratios are slid over an image to detect objects. As different objects can have different aspect ratios and sizes depending on the object size and distance from the camera, this process is quite slow. While there have been attempts to improve sliding window-based detectors, their performance still lags behind that of region proposal-based detectors [52]. Therefore, they are not as popular and will also not be implemented in this work.

Region proposal-based detectors (or R-CNNs) are by far the most accurate CNNs, but their computational cost is relatively high [24]. This is because R-CNNs divide images into small, merging regions and separately extract features from each region. This process is called selective search [78]. The original R-CNN by Girshick et al. creates about 2000 regions of interest, of which each is then classified with a category-specific linear Support Vector Machine (SVM) [24]. Finally, a regression analysis is applied
to refine the resulting bounding boxes. This heavy computation makes the network unsuitable for real-time image segmentation. This would not necessarily be a problem for the current application, as it is intended to be applied to archive already existing footage.

To increase the computational performance of R-CNN, Girshick proposed a model called Fast R-CNN [23]. Fast R-CNN extracts features from the whole image before dividing the image into regions, thereby decreasing computation time in training and inferencing. In that same year, Ren et al. presented a different version of R-CNN that is more efficient than Fast R-CNN. This new model called Faster R-CNN replaces the separate feature vector for each region with an internal deep network that derives the regions of interest from feature maps [61]. To increase the performance and generalizability of Faster R-CNN, He et al. developed a new model that extends upon Faster R-CNN by adding a new output branch [28]. This new output channel consists of an object mask and allows for pixel-level semantic segmentation. Mask R-CNN outperforms its predecessors and does not have significantly higher computation time, but retains the issue of being unsuitable for real-time applications.

Single shot detectors compromise on accuracy to increase speed. This allows them to be utilized for real-time applications. One of the most popular single shot detectors is the You Only Look Once (YOLO) network [58]. This network consists of a single neural network that predicts bounding boxes and outputs class probabilities for each of these bounding boxes from a full image in a single pass through the network. The YOLO base model is capable of processing 45 frames per second, while the optimized network Fast YOLO is capable of processing 155 frames per second [58]. This is a significant increase in computational performance compared to Faster R-CNN, which is capable of processing 5 frames per second [61]. While YOLO does not outperform Fast R-CNN or Faster R-CNN in terms of classification accuracy, it does outperform the original R-CNN network. The YOLO network has been demonstrated to successfully be able to track pedestrians in video data [46, 35].

Another popular single shot detector is the SSD Multibox Detector network by Liu et al. [43]. Like any single shot detector, object localization and classification are done in a single forward pass through the network. This model uses default bounding boxes that each correspond to different aspect ratios and scales, allowing it to detect objects at different scales. With a mean average precision of 74.3% on the PASCAL VOC2007 test dataset at a speed of 59 frames per second, SSD Multibox outperforms Faster R-CNN in both speed and accuracy [15, 43].

Image segmentation is a fast developing field. It is therefore not surprising that new and adjusted versions of all the aforementioned networks have been developed over the years. Some examples of these are SPPnet [29], YOLOv2 [59], YOLOv3 [59] and RetinaNet [41]. To go into details about how each of them works would exceed the scope of this project, but it is relevant to know that many different networks have been developed, each with their own strengths and weaknesses.

2.1.2 Object tracking

Object tracking cannot be separated from the task of object detection in video data analysis, as it essentially adds a temporal dimension to object detection. In the past, researchers have attempted to exploit this attribute with techniques like frame differencing and motion segmentation, but as with object detection, this is now mainly done with CNNs. A CNN that has been demonstrated to be applicable to pedestrian tracking has already been described in section 2.1.1, namely the YOLO network [46]. Similar to image segmentation, identifying objects is not a requirement for tracking. However, as mentioned before, it is necessary to be able to identify objects of interest (e.g., people
or vehicles) in the intended application.

Figure 2: Examples of challenges in object tracking, including a) illumination, b) occlusion, c) deformation, d) noise corruption, e) out-of-plane rotation, and f) motion blurring. Adapted from [40].

There are some challenges in object tracking that exist but are not as prevalent in object detection. Factors that can affect tracking performance include illumination variation, partial or full occlusion of objects, and background clutters [83]. Examples of these can be seen in figure 2. While tracking approaches exist that aim to eliminate some of these problems, there is no tracking method that can take all of them into account. The data available for object tracking is also limited. Datasets such as VIVID [9], CAVIAR [17] and PETS [18] contain small objects such as humans or cars, and have static background, which makes them less suitable to generalize to applications with crowded scenes. Furthermore, most datasets do not contain annotated bounding boxes, which makes it difficult to evaluate the performance of algorithms [83].

According to Li et al., a typical visual object tracking system consists of 4 modules [40]:

- **Object initialization.** This is the module in which objects are detected and highlighted by bounding boxes, ellipses, or pixel-level semantic segmentation. It can be done by manually annotating data, but is commonly done with CNNs as explained in section 2.1.1.

- **Appearance modeling.** This has two components, visual representation and statistical modeling. The prior constructs the object descriptor based on visual features, while the latter uses statistical learning techniques to create a model for object identification.

- **Motion estimation.** In this step the motion of objects is predicted based on the current state of the object, presumed noise during the state transition, and a state evolution function $f$. Motion estimation plays an important role in visual tracking software, and is especially common in e.g. self-driving car software [8]. Motion estimation models often are created with the assumption that the speed and/or
acceleration of objects is constant [80]. They typically make use of linear regression techniques, Kalman filters, or particle filters.

- **Object localization.** This final phase consists of attempting to localize the object. Traditionally, this is done through a greedy search or maximum a posterior estimation based on the motion estimation.

An important consideration that needs to be made is whether an application will use online or offline tracking methods. As mentioned before, online tracking requires faster models for object initialization and appearance modeling, which means that some sort of Single Shot Detector is most suitable. However, this would be at the expense of detection accuracy. Furthermore, future frames cannot be considered when applying online tracking methods. For the purpose of a reliable method to anonymously store video surveillance data, an offline tracking method is sufficient, as the recordings are processed later. This does not hurt the generalization capacities of this application, as the depth estimation method is still relevant to other fields that make use of image recognition, e.g. self driving cars, and robotics.

As discussed earlier, the problem of object tracking essentially adds the dimension of time to an object detection problem. A tracking problem can therefore be framed as estimating the states of a target object in subsequent frames [83]. A distinction can be made between global and local offline tracking. Global methods perform data association simultaneously across large batches of frames, whereas local methods perform it consecutively across a couple of frames [10]. As can be expected, this means that global methods are generally slower than local methods, but preserve more information as more of the context is taken into account. As the application in this project does not require real-time tracking, global methods can be considered.

## 2.2 Depth Estimation

Binocular depth perception is one of the most demanding visual tasks that humans can perform [53]. It is therefore not surprising that machines are notoriously bad at performing this task. One of the problems is that machines have to rely solely on the disparity between two images, while humans can apply their prior knowledge about the world, such as the typical size of certain objects. Therefore, although the use of Convolutional Neural Networks for binocular depth perception has been successful when restricted to specific datasets, these methods often fail to generalize [12].

Where binocular depth estimation is a difficult problem to solve, monocular depth estimation is inherently an ill-posed problem because it is no longer possible to rely on the disparity between two images. Humans, however, are not only capable of perceiving depth with one eye, but also of estimating depth from a 2D image. This is because humans can fall back on various visual cues to perceive depth, in addition to aforementioned prior knowledge of the world [76]. Monocular cues that humans utilize to perceive depth include colour, motion parallax (works only if the object is in motion), differences in shades and texture, linear perspective, and foreshortening [53]. Efforts have been made to model these cues into neural networks. Chen et al. [7], for example, generated a pixel-wise depth prediction with a single deep network and an ordinal dataset of relative depth. The dataset consisted of 421k single training images and 74k test images annotated with relative depth values. They achieved the best results by pre-training their network on the NYU depth dataset of ground-truth depth annotated indoor images [72] and fine tuning on their own dataset of relative depth, which they named Depth in the Wild (DITW) [7].

It can be challenging to incorporate all relevant monocular cues into a single model, because the learning algorithm would need to reason about the semantic context of ob-
jective 

This is why before deep neural networks became widely accepted, techniques such as shape from shading were considered state of the art. This method tries to estimate depth based on brightness, intensity and smoothness. The disadvantage of this technique is that annotated data containing these values is required. Other techniques that have been applied focused on prior scene understanding, in which images are semantically segmented before a feature mapping is generated for each semantic class separately. Similarly, researchers have attempted the opposite: to learn semantic labels on the basis of depth.

A depth estimation method usually receives either monocular or stereo images as input, and outputs a depth map which reflects the relative depths of objects. Conde Moreno attempted to estimate depth from monocular images with an approach adapted from a paper by Godard et al. Unlike more mainstream methods, this work implements an unsupervised or self-supervised approach. Depth is not learned directly, but inferred by first learning the relationship between stereo left and right images in an unsupervised manner. This allows the model to estimate a disparity for each image, from which the depth can be estimated. An advantage of this approach is that no labeled ground-truth depth data is required, which can be more difficult to acquire. A disadvantage is that binocular training images are required, as well as additional information such as the camera focal length. Godard et al. manage to achieve a Root Mean Square Error (RMSE) of 4.863 using their unsupervised method on the Kitti dataset of autonomous vehicle camera footage. Conde Moreno applied this same method in the depth estimation pipeline to predict the depth of pedestrians and reported a RMSE of 6,344. The resulting depth maps unfortunately contained a considerable amount of artifacts and temporally inconsistent depth estimations. This led to poor ground-plane estimations, and consequently inconsistent 3D reconstructions. Therefore, the depth maps produced with the method proposed by Godard et al. do not seem to qualify as adequate depth references. The disappointing results could be due to several reasons, e.g. inaccurate focal length estimation or camera calibration, or overfitting on the training data.

Similarly to object detection and tracking, there are many different methods available in the literature for depth estimation. Convolutional neural networks surged in popularity for this application as Eigen et al. demonstrated their usability. As the goal of this project is to research depth estimation methods for monocular images, disparity based methods will not be discussed. Instead, two unique approaches to monocular depth estimation will be briefly summarized below.

One approach presented by Chakrabarti et al. emphasizes depth cues by training a neural network to predict depth derivatives of different orders, orientations and scales at every image location. This way a probability distribution is created from which depth can be estimated by harmonizing the set of network predictions to produce a single depth map. This method distinguishes itself because it deviates from a common point-wise depth value regression. The model by Chakrabarti et al. was able to achieve a pixel-wise root mean square error (RMSE) of 0.620 on the NYU v2 dataset. Tan et al. utilize the novel method of generative adversarial networks (GANs) to generate depth maps. GANs are a relatively new technique with considerable potential. Tan et al. managed to generate depth maps for the NYU Depth v2 dataset which outperforms the model of Chakrabarti et al. with a reported RMSE of 0.597. This indicates that GANs can be trained on specific datasets, although it is debated whether they can be trained to generalize. Other possible applications of GANs are also being explored, such as creating additional training samples for data augmentation. As GANs will be applied during this project, they will be discussed in more detail in section 2.3.

A distinct feature that is lacking from the method of Godard et al. that is reflected
in the results by Conde Moreno, is temporal information. In a recording, each frame is correlated to the previous frame. Similar to object tracking, including temporal information in the network should reduce errors by removing artifacts. For example, static objects should retain their depth, while moving objects change their depth depending on the direction of their movement in the context of the image. By including temporal information, background information is also implicitly encoded. As mentioned before, humans are able to detect depth with one eye by comparing the size of objects relative to the size and position of other objects [76]. Therefore, if the relation between the size of (static) objects and objects of interest can be learned by the network, this could improve its depth estimation.

In section 2.1.2, it was briefly discussed that object tracking can be framed as the addition of a temporal component to object detection. In a similar manner, a temporal component could be added to depth estimation models to track depth across frames. It is possible that considering prior and future frames can mitigate errors in depth estimation, as objects such as people should not vary significantly in depth across close frames. Some researchers have suggested methods to incorporate temporal information in depth estimation models and shown these to be effective in improving performance [84, 88].

2.3 Generative adversarial networks

Generative Adversarial Networks (GANs) have been around since their development by Radford et al. in 2014, but have seen a surge in popularity recently due to promising new research results [56, 31, 34]. Their applications vary from image-to-image translation, style-transfer and synthetic data generation. They are praised for their apparent capability to produce highly realistic data, but are notoriously difficult networks to train [32].

A generative adversarial network consists of two networks, a generator model and a discriminator model. These are trained with a game-theoretical approach, as adversaries in a minimax game [56]. The goal of the generator model is to generate images that are as similar to the training data as possible, while the goal of the discriminator model is to recognize whether the images it receives are real or fake, i.e. is a sample from the real data or a sample generated by the generator model respectively. Figure 3 shows a typical GAN architecture. Note that the generator model is not trained directly, but rather through the discriminator model. This is done by employing the following loss function:

\[
E_x [\log(D(x))] + E_z [\log(1 - D(G(z)))]
\]

Given this loss function, \(E_x\) represents the expected value over all real data instances, \(D(x)\) represents the discriminator’s estimate of the probability that real data instance \(x\) is real, \(E_z\) represents the expected value over all random inputs to the generator, and \(G(z)\) represents the generator’s output given noise \(z\). Given the definition of a GAN, the discriminator will try to maximize this loss function, while the generator will try to minimize it. This means that in theory the discriminator model will continue to improve its recognition, simultaneously forcing the generator to also improve its synthetic output [32, 76].

In this project, the intended use of a GAN would be to generate depth maps from RGB images. One possible way to do this would be to use a specific GAN architecture called a conditional Generative Adversarial Network (cGAN) [31]. The main difference between a typical GAN architecture and a cGAN architecture is that the noise vector that the generator uses is combined with auxiliary information that conditions
Figure 3: An example of a typical generative adversarial network architecture. There are two models, a discriminator model and a generator model. The discriminator receives input from either the generator or the training set, and tries to determine which set the data belongs to. The generator model draws from a random noise distribution, and generates images. Image taken from [73].

Figure 4: An example of a typical generative adversarial network architecture. There are two models, a discriminator model and a generator model. The discriminator receives input from either the generator or the training set, and tries to determine which set the data belongs to. The generator model draws from a random noise distribution, and generates images. Image taken from [73].

As the GAN discriminator and generator network share a loss function and perform each training step sequentially, certain challenges arise. Perhaps biggest challenge is to achieve a stable training configuration. In an ideal scenario, both networks would settle around a Nash equilibrium. However, as the generator starts out producing random noise, the discriminator usually improves faster. As a result, the gradients of the generator will become so sparse that it will fail to learn at all [75]. There are some measures that can be taken to counteract this. One measure that is almost always taken in practise is to let the generator maximize log $D(G(z))$, which is equivalent to letting the
generator minimize \( \log(1 - D(G(z))) \), but produces larger gradients \[56\].

A related scenario that can lead to unstable training is the case in which the generator finds a solution that can fool the discriminator temporarily, i.e. ‘average’ several cases of the input data for maximization of short term gains. This situation is known as mode collapse \[56\]. Mode collapse can be identified by manually inspecting the generated images or by implementing minibatch discrimination. Minibatch discrimination is a technique in which all samples of a minibatch are compared to check for similarities that could indicate that the generator is producing the same output for each input sample \[66\].

Aside from training instability, a big challenge in adversarial training is that evaluation of the network performance is difficult. As the training loss of both models will start to hover around an equilibrium while both networks improve, the losses themselves no longer reflect the performance of the overall system. This is why model evaluation is still regularly done by manual inspection of the output \[2\]. This method is not only time consuming, but also lacks objectivity. Some researchers have suggested solutions such as visualizing discriminator features, or using a pre-trained Inception v3 model to rate the diversity and quality of the generated images \[2\]. Thus far, no consensus has been reached on a single approach. Therefore the issue of model evaluation in GANs remains an open research problem. This is less of a problem with conditional GANs, as their output can be compared to a ground-truth.

2.4 Data collection

The availability of ground-truth data remains one of the main challenges in neural network research. This is particularly true for computer vision problems, because gathering ground truth data is often expensive. In image segmentation data for example, ground-truth data needs to be annotated by hand and often remains ambiguous due to differences in interpretation by the annotator. For depth data, expensive scanners such as the LiDAR scanner are required \[22\]. This limitation often leads to multiple research projects centred around the few available datasets and arguably to overfitting on these datasets. This can be considered undesirable as it hurts the generalizability of models. Furthermore, a situation arises in which the available data leads the direction of research, rather than exploring new areas of interest.

To solve the challenge of obtaining annotated ground-truth data, researchers have recently tried to extract this data from 3D video games \[63, 26, 37\]. Using synthetic data to train neural networks is not a novel approach. While synthetic data is often generated by the process of data augmentation, the advantage of using data from hyperrealistic games such as Grand Theft Auto V and The Witcher 3 is that they provide visual data that is not only already annotated, but can also simulate a range of environments and weather conditions \[71\]. Additionally, video game data are inexpensive to acquire. The video frames and accompanying semantic and depth information can be extracted by adding a wrapper to the commonly used DirectX® rendering API \[37\].

Several researchers have shown that initial training on data gathered from video games is effective in improving the performance of neural networks in various image processing tasks. Richter et al., for example, extracted 25 thousand images from a photorealistic open-world computer game for image segmentation \[63\]. They showed that a model pretrained on video game data and only a third of the original CamVid ground-truth data outperforms a model trained on all of the ground-truth data \[63\].

Similarly, Krähenbühl, Shafaei et al., and Haji-Esmaeili and Montazer have been able to demonstrate that networks pretrained on video game frames provide better depth estimations than networks that were only trained on (limited) real world data \[26, 57, 71\]. Additionally, Krähenbühl also shows that video game networks generate better depth
estimations than networks pretrained on other synthetic datasets, suggesting that video game data resembles ground-truth data more closely than synthetic data derived from existing datasets [26]. It is therefore worth considering obtaining video game data for training purposes in this project.

2.5 Research questions and approach

Conde Moreno’s proposed pipeline takes 2D monocular video data as input and converts it to an anonymous 3D reconstruction [10]. This work aims to extend this system by improving the depth estimation method. Therefore, in the context of developing a privacy-preserving system that can reliably and accurately reconstruct a 3D scene from an outdoor single monocular camera recording, in which the movement and locations of the present objects of interest (such as people and vehicles) are preserved, the following overarching research question is proposed:

Q1. Can a state-of-the-art neural network be trained to accurately estimate relative depth in a crowded scene from monocular recordings of a single uncalibrated camera?

To answer this question, a series of experiments will be conducted. First, the option of using the relatively new technique of Generative Adversarial Networks will be explored. Specifically, conditional GANs will be used. Conditional GANs differ from traditional GANs in the fact that both the generator and discriminator are fed the input map, which allows for a loss that penalizes the joint configuration of the output [31]. This means that unlike with unconditional GANs, the output pixels of conditional GANs are considered dependent on each other. This is important because context plays such an important role into depth estimation [67, 81]. Additionally, for each input there is only one ground depth truth, since an image only has a single depth map. This also makes conditional GANs a suitable solution.

Further improvement to the network will be done by exploration of the three sub-questions stated below. This will allow for specific improvement approaches based on existing literature. As mentioned before, humans are not only capable of monocular depth estimation, but also of estimating depth from 2D images [53]. By investigating and exploiting biological cues in our machine learning algorithms, it could be possible to emulate human depth vision and thereby improve performance. Therefore, the first sub-question is formulated as follows:

Q1a. Does the incorporation of biological cues improve the performance of a single image depth estimation model?

Answering this question consists of two components: reviewing literature on which biological features are important for image depth estimation in humans, and implementing the relevant biological cues into a model. As emphasized before, constructing a 3D representation of a single monocular 2D image is a complex problem, since the global context of the image needs to be considered [67]. 3D position estimation based on single images is also an underspecified problem, because existing research relies on the use of stereo images. Since the data for this project consists of video data, an attempt can be made to exploit the correlations between frames to improve the performance of the depth estimation model. This leads to the following sub-question:

Q1b. Can we improve the results of a depth estimation network by including temporal information?
The last sub-question that will be addressed involves a recurring problem in computer vision, namely the limited availability of ground-truth training data for supervised models. This drives researchers to areas of research with available data, and leads to an extensive range of experiments conducted with identical data. While the latter is not necessarily a problem, it restricts the generalizability of presented statistical models. As discussed in section 2.4 a limited group of papers have recently turned their attention to hyperrealistic video games in an attempt to solve the lack of training data [37]. They found that this method allows for automatic and inexpensive collection of data, and that when combined with real-world training data, leads to better model performance. Therefore, the final sub-question is stated as:

Q1c. Can we improve the results of a depth estimation network by pre-training on hyperrealistic video game data?

3 Methodology

This chapter concerns the experimental designs of this project. First, a study of biological depth estimation is conducted, which explores biological depth cues and their possible applications in computer vision. After that, a series of experiments is done with the goal of answering the research questions stated in section 2.5. A total of three experiments were performed. Experiment 1 is described in section 3.2 and explores the use of generative adversarial networks for depth estimation of monocular images. Experiment 2 is described in section 3.3 and concerns the addition of video game data to improve a generative adversarial network. Finally, section 3.4 explains experiment 3, which includes temporal information in a generative adversarial network.

3.1 A study of biological cues of depth perception

Humans use over a dozen different cues to be able to perceive depth. There are different ways to categorize these cues. Firstly, depth cues can be divided into monocular cues and binocular cues, where the former involves the estimation of depth with one eye and the latter concerns depth estimation with two eyes. Depth cues can also be divided into either pictorial cues, motion cues, or physiological cues [74]. Which cues are used does not only depend on the viewing distance, but also on the environment that depth is perceived in [68].

Both humans and machines rely on the disparity between stereo images for binocular depth perception [53]. Disparity is most effective as a depth cue at shorter distances. Figure 5 shows an example of disparity in human binocular vision. The two circles represent a pair of eyes and the red and yellow squares represent two different objects. The objects are detected in different parts of retina, which is the tissue in the back of the eye that is responsible for converting light into neural signals. In humans, disparity is defined as the difference in angular position on the retina [48, 49, 81]. The disparity in figure 5 equals \( \alpha - \beta \). Angles \( \alpha \) and \( \beta \) tell us something about the relative disparity between the red and yellow object, but this disparity might be similar for objects that are at other distances because human eyes are not static. Therefore, to deduce the absolute distance to objects, humans require knowledge of the so-called vergence angle, represented in figure 5 by \( V_1 \). The vergence angle is the angle between the visual axes from the two eyes [48]. Knowledge about the vergence angle is not necessary for animals whose eyes are in fixed position, because then the same points in the retinas always correspond to the same head-centred space [48]. Arguably, knowledge about the vergence angle is not required for applications of computer vision if camera’s are static.
Humans derive information about the vergence angle from oculomotor cues. They utilize kinesthetic sensations from the extraocular eye muscles to judge distances up to 10 meters, i.e. the position of the eye muscles provides information about the distance of objects. When objects are closer, the angle between the two viewing axes will be larger than when objects are further away. In human vision, vergence always refers to convergence (turning eyes inwards). Therefore this cue for binocular depth perception is known as convergence [49].

Similar to convergence, muscular tension in the ciliary body (the part of the eye that controls the lens shape to keep objects into focus) can be a depth cue for humans [49, 48]. This process is called accommodation and is only effective at depth estimations of objects up to 2 meters [49]. This cue is considered a monocular depth cue, because it mainly is involved in monocular vision, but it has been shown to interact with convergence [51]. Accommodation and convergence are physiological cues of depth perception. It would in theory be possible to implement these oculomotor cues into computer vision. We would then need a non-static camera and keep track of its movement and (variable) focal length. However, in the intended application the camera will be static, so accommodation and convergence are not as usable.

There are various non-physiological cues that play a role in monocular depth perception. A study by Surdick et al. investigated the importance of seven different depth cues at a viewing distance of one and two meters [74]. They found that the most effective depth cues at those distances were linear perspective (LP), foreshortening (FS) and texture gradient (TX). These three cues are closely related, as they all concern object information relative to the ground plane [74]. Figure 6 shows an example for each of these three cues. As shown, linear perspective concerns parallel lines that come closer together in the distance. The point in the distance at which the lines meet is considered the vanishing point. Texture gradient works in a similar manner, where objects near the viewer appear to have more pronounced texture. As seen in [68], the texture of the ocean gradually reduces as a function of the viewing distance. Figure 6d depicts an example of foreshortening. Foreshortening is the process in which objects that are further away are shortened by contracting in the direction of depth. This is the case for the boy’s hand and arm in the picture. Ivanov et al. showed that foreshortening of a single surface can be utilized to perceive slant as effectively as texture cues [33].

LP, TX and FS are all examples of pictorial depth cues. Another pictorial depth cue that was recently found to be important to depth estimation is blur. Blur occurs due to depth-of-focus limitation in the eye and depth variation within a scene [45]. Objects...
outside of the focus area of the eye are blurred, with objects closer to the point of focus being blurred less than objects further away from that point. At first, blur was only considered as a qualitative depth cue, i.e. it was believed it supported other depth cues, but was not an accurate predictor of depth by itself. However, more recent findings have shown that blur is in fact capable of being an independent depth cue [80]. Blur was also found to be closely linked to disparity [30]. It seems that as disparity decreases for larger viewing distances, blur increasingly contributes to depth estimation.

Other pictorial depth cues include brightness, illumination, shading, and elevation, density, and geometry. While Surdick et al. found that differences in brightness were not important as a depth cue for distances of one or two meters, they argue that it could still be an important depth cue for shorter distances [74]. In the same study, Surdick et al. found that relative brightness, while believed to be an important depth cue, was not so effective as a depth predictor at 1 or 2 meters. However, it was found that it was indeed effective at smaller distances. The fact that brightness and relative brightness are only effective at distances smaller than 1 meter make them less relevant to depth estimation in outdoor surveillance scenes.

Aside from pictorial cues, motion cues also play a role in depth estimation. The most important motion cue is motion parallax [64]. This cue refers to the phenomenon in which objects closer to the viewer are perceived as moving faster than objects further away from the viewer. This can occur when either the viewer is in motion or the observed object is in motion. Rogers and Graham demonstrated through a series of experiments that motion parallax can be used as an effective depth cue in the absence of other depth cues. Furthermore, motion cues were shown to be especially effective in combination with static scenery [60]. This makes motion parallax an interesting candidate for the intended application, since static cameras are used to capture the scenes.

Now that we have discussed different biological depth cues, we should consider how to model them into our application. One possible approach would be to alter the data to emphasize the depth cues that we want to incorporate. This could perhaps increase the effectiveness of these cues. Of the pictorial depth cues LP, TX and FS seem to be most effective for our application. The texture gradient could perhaps be enhanced relative to distance with the use of an edge detector, but linear perspective and foreshortening both rely on the position of the horizon. It is therefore unclear how these could exactly be emphasized in the training data.

A different approach would be to use a network such as a conditional GAN to implicitly learn some of these cues. The advantage of a conditional GAN over a traditional Convolutional Neural Network is that the loss is learned, so that in theory any possible structure difference between the output and the target can be penalized [31]. This makes it possible to penalize inconsistencies between depth cues in the target and output images. Additionally, by making use of video data we can model motion cues such as

---

**Figure 6**: Examples of important cues in monocular depth perception. From left to right: linear perspective (LP), texture gradient (TX) and foreshortening (FS).
motion parallax.

3.2 Experiment 1: Generating depth maps from monocular images with a conditional Generative Adversarial Network

A general description of conditional Generative Adversarial Networks (cGANs) was provided in section 2.3. In the previous section on biological cues, we briefly discussed why cGANs are possibly better at modelling biological cues than traditional CNNs. In addition to that, cGANs should be capable of taking context into account [31]. This would in theory allow pictorial cues such as texture gradient, linear perspective and foreshortening to be learned. For that reason, a cGAN was trained in this experiment to generate depth maps from monocular RGB camera images. To our knowledge, this has only been done once before with a slightly different approach [21]. The results of this experiment will therefore contribute to this relatively unexplored approach of depth estimation. Furthermore, the trained model will serve as a baseline to measure improvements in following experiments.

In the following subsections the data will be described, and the specific model architecture will be discussed. The results of this experiment are summarized in section 4.1 and discussed in section 5.1.

3.2.1 Data analysis and preprocessing

For this experiment, the labeled NYU version 2 dataset was used [72]. This dataset consists of 1429 RGB and depth image pairs. The depth images are displayed in a HxWxN matrix of in-painted depth maps, where H, W, and N represent the image height and width, and the number of images respectively. The depth values consist of values between 0 and 10, which measure the distance in meters from the camera. The RGB images and depth maps were preprocessed before training. First, they were reduced from 64x480 pixels to 256x256 pixels with a k-nearest neighbours technique to speed up training time. Next, the image and depth pairs were normalized between the values -1 and 1. At first, an experiment was done with a random train and test set of 1200 training images and 249 test images to find the optimal model parameters. Final training and testing pairs were set according to the popular ‘Eigen split’, in which 795 images are reserved for training and 654 images are reserved for testing [12]. This allows for better comparisons to existing work, although it leads to less training images.

When examining the NYU version 2 dataset, the average depth value across all depth maps is 2.80 meters and the median depth value is 2.51 meters. The average and median depth value distributions per image in the dataset are presented in figure 7. Looking at these distributions, it is clear that most objects have a distance between 2 and 3 meters from the camera. There are fewer than 50 images that contain an average depth value of less than approximately 1.5 meters or larger than 4.5 meters. The sparsity of images with objects at these distances could indicate that a model would find these more difficult to learn.

One way to treat class imbalance or to increase the dataset size is through data augmentations. Augmentations can be done in multiple ways, so when performing augmentations it is important to choose a method appropriate for the type of data used. One possible way to increase the amount of data of minority classes is by applying algorithms such as the Synthetic Minority Oversampling Method Technique (SMOTE) to generate synthetic examples of the minority class [8]. This is often applied in combination with a downsampling of the majority class. Alternatively, it is possible to transform the existing data to create new data samples. This can be done by applying applications such as e.g. random flips, the addition of Gaussian noise, or elastic deformation [55].
(a) Average depth values in meters from the camera per image.

(b) Median depth values in meters from the camera per image.

Figure 7: Average and median depth values per image of the NYU version 2 dataset [72]. The x-axes show the depth values in meters from the camera and the y-axes represent the number of images with these values. The distribution of depth values show that the majority of images have average and median depth values between 2 and 3 meters from the camera.

Wong et al. distinguishes these two different approaches of augmentations as feature-space transformation and data-space transformations respectively [82]. They found that data-space transformations were more effective than feature-space transformations for the improvement of CNNs on the MNIST dataset [82]. As the model used in this experiment utilizes a CNN (U-Net) architecture, a similar result would be expected for this experiment. Therefore, the decision was made to only do augmentations in data-space with the transformations suggested in [31]. This means that mirroring and random jittering were applied to each depth map by first resizing the depth map randomly to 286x286 pixels with a k-nearest neighbour method, before randomly flipping the image vertically with a 50 percent chance. However, this proved to introduce too much noise into the system and worsened the model performance. The results of this can be found in section 4.1. Therefore in the final model training, only a vertical flip with 50 percent chance was preserved.

3.2.2 Model and training procedure

A conditional Generative Adversarial Network (cGAN) was trained to generate depth maps from input RGB images. The problem is treated as an image translation problem, as the assumption is made that the input RGB images and the output depth maps share the same underlying structure. It was mentioned before that cGANs differ from conventional GANs, because both the discriminator and generator are allowed to observe the input images [31]. This makes that the output of the generator model is conditioned on the input image and will resemble a depth map.

The model used in this experiment follows the so-called ‘pix2pix’ implementation proposed in the paper by [isola et al.31]. The generator network in this GAN incorporates an encoder-decoder structure with skip connections between mirrored layers. This architecture, based on the so-called U-Net, takes the input and progressively downsamples it, before passing it through a ‘bottleneck’ and upsampling again [65]. The skipped connections allow information to be shared directly across the network, making it possible for low-level information that is shared in the input and output to get through the bottleneck.

The generator consists of 7 downampling blocks and 7 upsampling blocks separated by a bottleneck layer. Each downsampling and upsampling block contains 2 convolution layers or 2 deconvolution layers respectively. This architecture is followed by a final deconvolution layer with a hyperbolic tangent activation function, which maps the
output to values between -1 and 1. The discriminator in turn consists of 6 convolution layers with a leaky ReLu activation. The network architecture can be viewed in Figure 8. The discriminator architecture, named patchGAN, tries to classify if NxN patches of the image are real or fake. [Isola et al.] found that a discriminator in combination with patches of 70x70 pixels was the most effective [31]. By applying penalization on image patches rather than the entire image, the network can run faster with fewer parameters and model high-frequency structure. The assumption is made that the low-frequency structure will be modelled correctly by adding an L1 term, since L1 loss measures the distance between the input and output images [31]. The output of the discriminator layers is passed through a sigmoid function, predicting the likelihood that an input is a real translation of the source image.

![The generator](image1.png)

![The discriminator](image2.png)

Figure 8: A depiction of the network architecture. The top part of the image shows the generator encoder-decoder structure, with each of the blocks containing 2 convolution or deconvolution layers respectively. The encoder and decoder layers are also connected to allow the flow of information past the bottleneck. The generator input consists of an input rgb image and the output consists of a generated depth map. The discriminator shown in the lower half of the image consists of 6 convolution layers. The 30x30x1 image in the last layer represents a patch of 70x70 pixels. The discriminator receives an rgb image and a depth map as input and will output a prediction on whether the image was ground truth or fake.

Both the discriminator and generator apply batch normalization. Training was done with a batch size equalling 1, which is equivalent to instance normalization.
normalization has been shown to drastically increase performance in image generation tasks compared to batch normalization [79]. However, using a batch size of 1 increases training time.

As mentioned before, it is conventional to add a random noise vector $z$ to the generator input. The use of this random noise vector increases the robustness of the model against noise in the input data. However, the creators of the pix2pix architecture found that the network learns to ignore this noise [31]. Therefore, it was not implemented in this experiment. Instead, dropout was applied during training in the discriminator network.

The generator and the discriminator are trained simultaneously, with the following objective function:

$$G^* = \arg\min_{G} \max_{D} \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

Within this function the generator and discriminator loss as well as the L1 distance are respectively defined as:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))]$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1]$$

The discriminator is thereby trained directly on the input and target images, whereas the generator tries to minimize the loss predicted by the discriminator on ‘real’ images and the L1 loss between the generated and target images. This is achieved by defining another model that stacks the generator on top of the discriminator. As training was stable, it was not necessary to implement a Wasserstein loss. Both the generator and discriminator weights are initialized by drawing from a Gaussian distribution with mean 0 and standard deviation 0.02 [31]. During training, the discriminator loss is weighted so that the discriminator optimizes at half the speed of the generator. Additionally, the generator loss is weighted so that MAE is taken into consideration more strongly than the discriminator loss with a ratio of 1 to 100. In this manner, plausible images are encouraged and the risks of mode collapse are reduced.

[Isola et al.] suggest to use an ADAM optimizer with beta coefficients $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for both the discriminator and the generator. As the generator’s performance quickly tends to become worse than the discriminator’s, the training process can suffer from instability. As discussed before in section 2.3, the potentially sparse gradients can prevent the generator from learning [2]. As this indeed occurred in the first iterations of training, measures were taken to counter this. Different learning rates and optimizers were experimented with, and the model proved to be very sensitive to changes in this parameter. Eventually the discriminator optimizer was replaced with a stochastic gradient descent with learning rate 0.0002 and no momentum. Furthermore, the real labels of the depth maps in the discriminator were smoothed, as suggested by [Müller et al.] [37]. Label smoothing is a technique in which the labels of the ground truth are altered to weaken the certainty of the discriminator on its predictions of the real world data. In this case, the labels of the depth maps fed to the discriminator were altered to randomly be assigned to values between 0.7 and 1.2 for real world depth images. By applying these three techniques, the stability problems of the network resolved and both networks settled around an equilibrium during training time.

A single training cycle of the entire network proceeds as follows: a random preprocessed image and corresponding depth map is loaded into memory and vertically flipped with a 50% chance. This image and depth pair are then fed to the generator. After the generator makes its prediction, the discriminator is given the same input image together
with the generator output. It also receives the input image and the ground truth depth map. Next, weights are adjusted by back-propagating through the entire network. One epoch consists of \( \frac{\text{Number of training images}}{\text{Batch size}} \) training steps. The model ran for 100 epochs, i.e. 79500 training steps. After each epoch, the model outputs predictions for three random images and a score to indicate the performance. The metric that was used to compute the score simply compares the network generated output on a pixel-level with the ground truth depth map. This so-called difference score is computed in the following manner:

\[
\text{Difference}(\text{generator output}, \text{ground truth}) = \frac{|\text{generator output} - \text{ground truth}|}{\text{Number of pixels}}
\]

Note that this score is only an indicator that is used during training to monitor performance and not as a final evaluation of the model. For the model evaluation a number of benchmark metrics for depth estimation models are used to calculate the performance on the test set. These include \( \delta < 1.25, \delta < 1.25^2, \delta < 1.25^3 \), RMSE, AbsRel, SqRel, and RMSElog \([12, 25, 76]\). The definitions of each of these metrics are stated below.

Threshold: \% of max\( \left( \frac{y_i}{y_i'}, \frac{y_i'}{y_i} \right) = \delta < \text{thr} \)

Abs Relative difference: \( \frac{1}{|T|} \sum_{y \in T} \frac{|y - y'|}{y'} \)

Squared Relative difference: \( \frac{1}{|T|} \sum_{y \in T} \frac{|y - y'|^2}{y'}^2 \)

RMSE (linear): \( \sqrt{\frac{1}{|T|} \sum_{y \in T} (y_i - y_i')^2} \)

RMSE (log): \( \sqrt{\frac{1}{|T|} \sum_{y \in T} (\log y_i - \log y_i')^2} \)

RMSE (log, scale-invariant): \( \frac{1}{2n} \sum_{i=1}^{n} (\log y_i - \log y_i' + \alpha(y, y'))^2 \)

### 3.3 Experiment 2: Generating depth maps from monocular image with hyperrealistic video game data from GTA V

The lack of (annotated) ground-truth data for computer vision experiments was extensively discussed in section 2.4. To summarize, more data would likely improve depth estimation models, but data collection is often expensive because it needs to be annotated by hand. In this experiment data will therefore be gathered from the hyperrealistic video game Grand Theft Auto V. This will provide data that is already annotated and can simulate different environments. The same model as in experiment 1 will be pretrained on the game data and then trained on the same real world data. Based on previous research, it is expected that a model trained sequentially on the synthetic data and the real world ground-truth data will outperform a model that is solely trained on the real world ground-truth data \([26, 71, 37, 63]\).

#### 3.3.1 Data collection and preprocessing

Although papers that utilize data from the video game GTA V have been written (e.g. \([26, 71]\), extracting data from the game is not a trivial task. There are several challenges. Firstly, manufacturers of video games do not endorse reverse-engineering of their games. This means that there are no out-of-the-box resources available to extract game data, and a system needs to be created from scratch. Luckily, game communities often create software that can hook into the game engine in order to make modifications to the game. A disadvantage of this, however, is that this kind of software is often developed when
the game is initially released, and not kept up to date. Furthermore, documentation of
the software and code readability are extremely limited.

To extract data from GTA V, both the methods proposed by Krähenbühl and Haji-
Esmæi\textsuperscript{l}i and Montazer were implemented. However, both projects are a couple of years
old and neither functioned fully with the current version of the game. While the on-
screen frames were successfully collected, the extracted depth and stencil images were
corrupted. Subsequently, data was not gathered as planned, but instead taken from
the Closed VirtualScapes dataset \[57\]. The Closed VirtualScapes dataset consists of
8371 scenes recorded in GTA V from one camera positioned on each side of a car. The
scenes captured from these four cameras resulted in 89196 chronological frames with
corresponding stencil data, depth data, and .json data that describes the properties of
each frame. This means that although it was not possible to simulate different weather
conditions as the data was collected directly from the game, still plenty of data could
be used.

The depth maps in the VirtualScapes dataset contain Normalized Device Coordi-
nates. As the goal is to predict depth in meters, the VirtualScapes dataset needed to be
preprocessed first. The conversion to meters is computationally expensive, so a subset
of images was first randomly selected to form a train and test set. 24000 images were
added to the train set and 2400 images were added to the test set. Initially, the train
and test set were balanced to contain images from different in-game times, simulating
three world states: dusk/twilight, day, and night. However, the model struggled with
the prediction of depth for nighttime images. Although it is desirable for the final ap-
plication to be able to predict depth from images taken at different times of day, the
decrease in model performance and the lack of nighttime images in the real world dataset
led to the decision to replace the dusk/twilight and nighttime images. This means that
24000 daytime image and depth map pairs with clear weather conditions were utilized
for training of the final model, and 2400 daytime image and depth map pairs with clear
weather conditions were reserved for testing.

Similar to experiment 1, the depth maps are displayed in a HxWxN matrix of in-
painted depth maps, where H, W, and N represent the image height and width, and the
number of images respectively. After a transformation to distance in meters, the depth
values consist of values between 0 and 10000 meters, which measure the distance in
meters from the camera. All images and depth map pairs were resized to 256x256x3. To
maintain the aspect ratio, the images were first cropped from the centre to the smallest
dimension of 1057 pixels, so that the image dimensions became 1057x1057x3, and then
resized with a k-nearest neighbour technique.

Figure 10 shows an example distribution of distances within a depth map from the
VirtualScapes dataset. As can be seen, there is a large portion of the image with depth
values of 10000 meters. This part of the image represents the sky, which is clipped at
10000 meters in the game rendering pipeline, although it technically represents infinite
depth. After initial training iterations showed that the wide range of depth data biased
the model predictions to larger differences, it was decided to leave out the sky when
making predictions. To this purpose, a binary mask was created of the same dimensions
as the depth maps. This mask is defined by setting pixel values for corresponding depth
pixels with larger values than a certain threshold to 0 and otherwise to 1. In this manner,
predictions for the larger depths can be set to 0 by multiplying the predicted depth map
with the binary mask. Since we want the model to have a similar range of depth as the
real-world application, the threshold was set at 200 meters. Figure 9 gives an example
of a depth map before and after the application of a binary attention mask.
3.3.2 Model and training procedure

A model was pre-trained on the VirtualScapes dataset and then trained on the NYU version 2 dataset. The same generative model that was used in experiment 1 was applied in this experiment to ensure fair comparison between performance on the real world dataset. However, a slight modification was made to the manner in which the synthetic VirtualScapes data was trained. Binary masks were created to nullify the model prediction of large depth values in the model evaluation. This was done to eliminate sky pixels and reduce the range of depth values. Furthermore, the created binary masks were applied during training time as an attention map. Attention maps,
also called binary region of interest (ROI) masks, try to focus the model on a certain area of the image. Application of these ROI masks have proven to boost performance of CNNs in various computer vision problems [14, 13, 27]. Research indicates that applying the attention mask after the first convolutional layer of the network is most effective [13]. Furthermore, this same research suggests that it does not matter if the mask is multiplied or added to the output of the first layer of the network. In the VirtualScapes training pipeline, the mask was multiplied with the output of the first generator layer and the first discriminator layer. The rest of the network architecture remained identical to the network used in experiment 1 (see: 3.2). A simple representation of the training the pipeline can be seen in Figure 11. To investigate the effectiveness of the ROI mask, a separate model was also trained without this addition.

Figure 11: A depiction of the network architecture with ROI mask. The top part of the image shows the generator encoder-decoder structure, with each of the blocks containing 2 convolution or deconvolution layers respectively. The first encoder block of both the generator and the discriminator is split so that both the input images and the mask pass through a convolutional layer before the outputs are multiplied together.

While transfer learning is an established technique in the training procedure of deep networks, it has not often been demonstrated in GANs. One of the challenges of transfer learning in GANs is that the training stability can be compromised when switching datasets. In particular, we face the familiar problem of the discriminator quickly outperforming the generative model and preventing it from learning. If all the generator, discriminator and placeholder model weights were used for initialisation and the optimizer state was restored this already happened after one epoch. In the final stable version of the model a new optimizer was initialised and the weights of the discriminator,
generator and placeholder model were reused, but all the model weights were retrained with the starter learning rate of 0.0002. The model was trained on the VirtualScapes dataset until it no longer improved as indicated by the difference score, and then further trained on the NYU version 2 data in the same manner. The results are shown in section 4.2 and are discussed in section 5.2. Model evaluation is done according to the same benchmark metrics as in experiment 1, namely $\delta < 1.25, \delta < 1.25^2, \delta < 1.25^3$, RMSE, AbsRel, SqRel, and RMSElog. 12, 25, 76.

3.4 Experiment 3: Incorporating temporal information in a model for depth estimation from monocular images with a conditional Generative Adversarial Network

This experiment will investigate the effects of adding temporal information to the adversarial model. As discussed in section 2.2, temporal information could improve model performance because prior and future frames can provide information about likely depths. For example, a person that is 2 meters away will most likely not move 4 meters away in the next frame and vice versa. Furthermore, the inclusion of temporal information also implicitly incorporates one of the monocular depth cues that was found to be important to human depth estimation: motion parallax. As described in section 3.1 motion parallax refers to the perception of distance relative to velocity: objects that are closer to the observer seem to move faster than objects at a distance. Given enough examples, the model may learn this relationship and thereby improve its performance.

3.4.1 Data analysis and preprocessing

Both the NYU version 2 dataset and the VirtualScapes dataset consist of video sequences, but the subset of these datasets that were used in previous experiments consisted of randomly selected frames. This was done to ensure a variety in training and testing data. For this experiment, however, a sequence of chronological frames was required for each training step. The NYU version 2 dataset is quite small and does not contain one continuous video, but rather several small ones. This makes it less suitable to be split up into sequences. Therefore this experiment will only use data from the VirtualScapes dataset.

The frame rate of the VirtualScapes dataset equals 24 frames per second. We judged that 2 seconds of data should be sufficient for the model to learn context. This would mean that 48 frames would be included in a training sequence. However, taking the training time into consideration, we decided to cut down the amount of frames to 12. This means that in every sequence of 48 frames, every 4th frame was used. As in the previous experiment, only images captured at in-game "day" time with clear weather conditions were considered. 24000 consecutive training images were selected, equally divided over four cameras to create more variety in the training images. This means that four video sequences of 6000 frames were acquired for training. In the same manner, 9600 frames were selected from the four cameras for testing. As we want a set amount of sequences in the test set, the frame rate was immediately reduced to the predetermined frame rate of 6 frames per second. This resulted in a test set with 2400 images in total.

The images in the train and test set were cropped to a size of 1057x1057 pixels to maintain the aspect ratio. They were then resized to 256x256x3 pixels with a k-nearest neighbour technique. The depth maps were processed in a similar manner and converted to depth values in meters as described in experiment 2 (see: 3.3). Furthermore, binary ROI masks were constructed to eliminate large depth values such as the sky pixels from the simulations. The threshold for the ROI masks was set to a value of 200 meters, where depth values smaller than 200 were encoded with value 1 and depth values larger
than 200 were encoded with value 0.

3.4.2 Model and training procedure

In this experiment a recurrent conditional Generative Adversarial Network was trained on a subset of the VirtualScapes dataset. The model will be compared to the previous ROI GANDepth model to investigate the possible benefits of adding temporal information. This is done by training ROI GANDepth from scratch with the training process described in section 3.3 on the same dataset as the recurrent cGAN model. To allow for a fair comparison, the model architecture of previous experiments was preserved as much as possible in the implementation of the recurrent cGAN model. The cGAN model was therefore based on a paper by [Rezaei et al.] that implements a conditional recurrent GAN based on the pix2pix architecture from [Isola et al.] and applies it on medical time-series data [62]. Similar to this paper, the bottleneck layer in the generator was replaced with a bidirectional LSTM layer. This allows the model to not only consider the relationship between the current frame and prior frames, but also future frames. This is suitable for the final application, as the system requested by Paaspop is purposed for existing offline video data. The discriminator architecture has also been altered to include recursercency, replacing the final dense layer with a bidirectional LSTM layer. The altered network architecture can be viewed in figure 12.

The training process of the recurrent cGAN model is identical to the one described in section 3.3, with the exception that the model takes a sequence of 12 chronological rgb images and ROI masks as input during each training step, and outputs a sequence of 12 corresponding depth maps. Furthermore, to prevent overlap between the frame sequences of different cameras, a camera is randomly selected during each training step, after which a random index is chosen from which the sequence of 12 frames will be drawn. In this manner, the model was trained on the chronological VirtualScapes dataset for 0.5 epoch with 12000 training steps.

Model evaluation was again done according to the same benchmark metrics as in experiment 1 and 2, namely \( \delta < 1.25, \delta < 1.25^2, \delta < 1.25^3, \) RMSE, AbsRel, SqRel, and RMSElog [12, 25, 76]. As in previous experiments, the pixel-wise average score for each of these benchmarks is calculated per image in the test set. The results of this experiment are described in section 4.3 and discussed in section 5.3.
Figure 12: A depiction of the recurrent conditional generative adversarial network architecture. The top part of the image shows the generator encoder-decoder structure, with each of the blocks containing 2 convolution or deconvolution layers respectively. To add recurrence the bottleneck layer in the generator was replaced with a bi-directional LSTM layer. The final layer of the discriminator was also replaced with a bi-directional LSTM layer. To pass the ROI masks through the network the first encoder block of both the generator and the discriminator are split so that the input images and the masks pass through a convolutional layer before the outputs are multiplied together.

4 Results

In this section the results of all experiments are reported. Section 4.1 summarizes the results of experiment 1, "Generating depth maps from monocular images with a conditional Generative Adversarial Network. Section 4.2 summarizes the results of experiment 2, "Generating depth maps from monocular images with hyperrealistic video game data from GTA V". Lastly, section 4.3 describes the results of experiment 3, "Incorporating temporal information in a model for depth estimation from monocular images with a conditional Generative Adversarial Network".

4.1 Experiment 1: Generating depth maps from monocular images with a conditional Generative Adversarial Network

In this section, the results of experiment 1 are outlined. As discussed in section 3.2, a GAN was trained on the NYU version 2 data. In the initial phases of the experiment, a training set of 1200 images was used. This training set was augmented with vertical flips and random jitter to create training sets of 2400 and 4800 image and depth map pairs respectively. Both SGD and Adam optimizers were applied in the discriminator to each of these datasets. The results of these simulations are presented in table [ ]
Table 1: Simulation results for the cGAN described in section 3.2. The model ran for 100 epochs with batch size 1 and learning rate=0.0002. The column headers indicate whether an Adam or SGD optimizer was used for the discriminator and how many training images were used. High scale-invariant loss stands out, which indicates that the cGAN does not have a consistent error across samples. Overall the model with an SGD optimizer and 1200 training images had the best performance while maintaining training stability. *Mode collapse of the discriminator occurred, which led to no more improvements.

The results presented in table 1 seem to indicate that data augmentations did not improve the model’s performance, as the score of the model with 1200, 2400, and 4800 images are similar. In fact, the model trained with only 1200 images in combination with an SGD optimizer seems to perform slightly better than other SGD models trained with more data. Furthermore, each of the models with a discriminator utilizing Adam eventually displayed mode collapse. On the basis of these results the discriminator of the final model was optimized with a SGD optimizer. It was also decided to leave out the more aggressive augmentation random jitter and instead only do vertical flips, as described in section 3.2. Figure 13 displays some examples of the model output.

After the final training parameters were set, a simulation of the NYU version 2 data was run with the Eigen split. The results of this simulation are summarized in figure 14 and table 2 in which the final generative model is referred to as GANDepth. Figure 14 shows the average and median depth per image for both the ground-truth depth maps and the generated depth maps. The distributions of the generated depth maps are more narrow, indicating that the model had difficulty with less frequent depth values. This is reflected in the results displayed in table 2 where $\delta < 1, 25^2$ and $\delta < 1, 25^3$ is substantially lower than $\delta < 1, 25^3$. In this table, the results of the original Eigen paper are also displayed for comparison. Note that although GANDepth seems to perform less well than the original Eigen model, Eigen et al., utilized 120k extra training images as they processed additional raw data from the NYU version 2 dataset, which lay out of the scope of this work. Similarly to the results of the previous simulation, the relatively high scale invariant loss stands out.

<table>
<thead>
<tr>
<th>threshold $\delta &lt; 1, 25$</th>
<th>Adam* 1200</th>
<th>SGD 1200</th>
<th>Adam* 2400</th>
<th>SGD 2400</th>
<th>Adam* 4800</th>
<th>SGD 4800</th>
</tr>
</thead>
<tbody>
<tr>
<td>threshold $\delta &lt; 1, 25^2$</td>
<td>0.77 0.77</td>
<td>0.76 0.76</td>
<td>0.76 0.76</td>
<td>0.77 0.77</td>
<td>Higher is better</td>
<td></td>
</tr>
<tr>
<td>threshold $\delta &lt; 1, 25^3$</td>
<td>0.91 0.91</td>
<td>0.91 0.91</td>
<td>0.91 0.91</td>
<td>0.92 0.92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>abs relative difference</td>
<td>0.28 0.27</td>
<td>0.29 0.28</td>
<td>0.28 0.28</td>
<td>0.29 0.29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sq relative difference</td>
<td>0.40 0.39</td>
<td>0.42 0.41</td>
<td>0.43 0.43</td>
<td>0.42 0.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSE (linear)</td>
<td>1.22 0.98</td>
<td>1.22 1.23</td>
<td>1.21 1.19</td>
<td>Lower is better</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSE (log)</td>
<td>0.36 0.37</td>
<td>0.37 0.37</td>
<td>0.37 0.37</td>
<td>0.37 0.37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSE (log, scale inv.)</td>
<td>29.19 28.72</td>
<td>30.84 30.18</td>
<td>30.91 31.17</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 13: Examples of the model output from the first set of simulations on the NYU version 2 dataset. Each output figure contains three images, of which the first represents the input RGB image, the second represents the model generated depth map, and the third represents the ground-truth depth map. It is difficult to judge the images by sight, but 13a and 13d seem to represent the object close to the camera better than 13a and 13b. Each of the models seem to have difficulty with the silhouette of the fitness machine. In particular, 13b and 13c lack the long upper part completely.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Eigen</th>
<th>GANDepth</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta &lt; 1, 25$</td>
<td>0.62</td>
<td>0.48</td>
</tr>
<tr>
<td>$\delta &lt; 1, 25^2$</td>
<td>0.89</td>
<td>0.77</td>
</tr>
<tr>
<td>$\delta &lt; 1, 25^3$</td>
<td>0.97</td>
<td>0.91</td>
</tr>
<tr>
<td>abs relative difference</td>
<td>0.22</td>
<td>0.34</td>
</tr>
<tr>
<td>sqr relative difference</td>
<td>0.21</td>
<td>0.45</td>
</tr>
<tr>
<td>RMSE (linear)</td>
<td>0.91</td>
<td>1.00</td>
</tr>
<tr>
<td>RMSE (log)</td>
<td>0.29</td>
<td>0.36</td>
</tr>
<tr>
<td>RMSE (log, scale inv.)</td>
<td>0.22</td>
<td>27.56</td>
</tr>
</tbody>
</table>

Table 2: Simulation results for the GANDepth model described in section 3.3.2. The model ran for 100 epochs with batch size 1 and learning rate 0.0002. The ‘Eigen’ column shows the results documented by Eigen et al. [12]. GANDepth performs worse than the Eigen model, but Eigen used 120k additional training images. In particular, the $\delta$ values indicate that the objects closer to the camera are predicted less accurately. This is consistent with the results displayed in 14. High scale-invariant loss stands out, which indicates that the model does not have a consistent error across samples.
Figure 14: Average (a) and median (b) depth values per image of the NYU version 2 dataset [72] for the Eigen test ground truth depth maps and the GANDepth generated depth maps. The x-axes of the scatter plots show the true depth values in meters from the camera and the y-axes of the scatter plots represent the predicted depth values in meters from the camera. Additionally, each axis is supplemented with a corresponding histogram that shows the distribution of depth values in meters of the NYU version 2 test set. The distribution of depth values show that predictions center around the mean depth values. For the generated images, there are no average depth values smaller than 1.41 meters or larger than 4.13 meters, whereas the smallest and largest average value for the target images equal 1.01 meters and 6.76 meters respectively. The smallest and largest median depth values show a similar trend: 1.44 and 4.05 for the generated depth maps, and 0.93 and 6.87 for the ground-truth depth maps.
4.2 Experiment 2: Generating depth maps from monocular image with hyperrealistic video game data from GTA V

This section describes the results of experiment 2. As discussed in section 3.3, a cGAN was trained on the synthetic VirtualScapes dataset with the goal of improving the performance on the real-world dataset through transfer learning. This section is divided into two parts. In the first part we will discuss the results with regard to the application of Region of Interest masks on the VirtualScapes dataset, and in the second part we will discuss the simulations results of transfer learning from the VirtualScapes dataset to the real-world NYU version 2 dataset.

4.2.1 Application of Region of Interest masks

As mentioned before, the conditional generative model was initially trained with images taken at various in-game times. However, because the model performed very poorly on night-time images these were removed from the final training set. An example output from the model on the night-time images can be seen in figure 15.

After removal of the night-time images, it became clear that the sky depth values and subsequent wide range of depth values was still a problem. As discussed in section 3.3, ROI masks were therefore applied to remove the sky pixels. The results of this simulation are summarized in 3.

![Figure 15: An example output of the GANDepth model during training time on the VirtualScapes dataset. Each column represents a single image, in which the top row shows the rgb input, the second row displays the model generated depth map, the third row shows the ground-truth depth map and the fourth row is a visual representation of the difference score. As discussed, the model performs significantly worse on the night-time images (column 1 and 2). In particular, it cannot distinguish the sky from the rest of the image, whereas it has no trouble doing so when presented with an image taken at dusk (column 3).](image)
Table 3: Simulation results for the GANDepth simulations described in section 3.3 with only the Virtualscapes data. The model ran for 3 epochs of 24000 training steps with batch size 1 and a learning rate of 0.0002 for both the generator and the discriminator. The ‘GANDepth’ column displays the results of the model trained on the test set without a ROI mask. It is clear that the results of the GANDepth model with a Region of Interest (ROI) mask are significantly better than the one without. Again, high scale-invariant loss stands out.

The GANDepth model was trained with and without a ROI mask. The results of these two simulations are summarized in table 3. In both simulations, the mask was applied at evaluation time so that the model did not make any predictions for the masked area.

The results of the model improved dramatically with application of the ROI mask. To see if this difference in performance was significant, a binomial test was executed for each of the metrics. Each of the scores per image in the test set were binary encoded to represent whether the added ROI mask improved the GANDepth performance, with null hypothesis that the hypothetical distribution with and without ROI mask would be equal. The formulation of the alternative hypothesis depended on the metric evaluated, as some metrics indicate better performance with a higher value, when others indicate a better performance when they are small. A binomial one-tail test showed that the performance of GANDepth ROI was systematically better (p < 0.05) for each of the metrics tested, as displayed in table 4.
Table 4: This table shows the mean and p-values from a binomial one-tail test on the individual GANDepth model predictions of the VirtualScapes dataset with and without ROI mask. The predictions in the test set were binary encoded, where 0 indicates that GANDepth had a higher value, and 1 indicates that GANDepth ROI had a higher value. For the metrics $\delta < 1.25$, $\delta < 1.25^2$, $\delta < 1.25^3$, RMSE, AbsRel, SqRel, and RMSElog, the majority of the GANdepth ROI predictions were indeed better, i.e. $\mu = 1$ for the metrics in which higher values indicate better performance, and $\mu = 0$ for metrics in which smaller values indicate a better performance. This indicates that GANDepth ROI outperformed GANDepth for each of the metrics.

An overview of the depth value distributions of the predictions of GANDepth and GANDepth ROI can be found in figure 16. Similarly to the results of experiment 1, it seems that the less frequent depth values are predicted less accurately. Furthermore, it becomes clear that GANDepth ROI generated depth maps are closer to the true depth values than GANDepth generated depth maps.
Figure 16: Average and median depth values per image of the VirtualScapes test ground truth depth maps and the GANDepth and ROI GANDepth generated depth maps. The x-axes of the scatter plots show the true depth values in meters from the camera and the y-axes of the scatter plots represent the predicted depth values in meters from the camera by the respective models. Additionally, each axis is supplemented with a corresponding histogram that shows the distribution of depth values in meters. Similar to previous results, the distribution of depth values show that less frequent depth values are predicted less accurately. Furthermore, it’s clear that the predictions of the ROI GANdepth model are closer to the true depth values.
4.2.2 Transfer learning with a cGAN

As GANDepth ROI was found to be the best performing model, it was used to continue training on the NYU version 2 dataset. The results of this simulation are summarized in table 5 and figure 17. The results indicate that GANDepth pre-trained on the VirtualScapes dataset outperforms GANDepth trained solely on the NYU version 2 dataset. All $\delta < \text{threshold}$ scores are higher, and all error metrics are lower. To verify these results, a binomial one-tail test was executed, of which the results are summarized in table 6. The results indicate that VirtualScapes GANDepth outperforms GANDepth in a majority of cases for each of the tested metrics.

The distributions displayed in 17 still indicate that the model predictions centre more around the mean. However, recall that the results of experiment 1 (section 4.1) showed a minimum average depth value of 1.41 meters, a maximum average depth value of 4.13 meters, a minimum median depth value of 1.44 meters, and a maximum median depth value of 4.05 meters. This means that the VirtualScapes model shows a slightly wider range of depth values, as well as a slight shift to larger depth values.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Eigen</th>
<th>GANDepth</th>
<th>VirtualScapes</th>
<th>Higher is better</th>
<th>Lower is better</th>
</tr>
</thead>
<tbody>
<tr>
<td>threshold $\delta &lt; 1,25$</td>
<td>0.62</td>
<td>0.48</td>
<td>0.49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>threshold $\delta &lt; 1,25^2$</td>
<td>0.89</td>
<td>0.77</td>
<td>0.79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>threshold $\delta &lt; 1,25^3$</td>
<td>0.97</td>
<td>0.91</td>
<td>0.92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>abs relative difference</td>
<td>0.22</td>
<td>0.34</td>
<td>0.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sqr relative difference</td>
<td>0.21</td>
<td>0.45</td>
<td>0.37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSE (linear)</td>
<td>0.91</td>
<td>1.00</td>
<td>0.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSE (log)</td>
<td>0.29</td>
<td>0.36</td>
<td>0.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSE (log, scale inv.)</td>
<td>0.22</td>
<td>27.56</td>
<td>26.97</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Simulation result comparison for the GANDepth and ROI GANDepth simulations described in section 3.3. The GANDepth model results were already described in 4.1. The ROI GANDepth trained on the VirtualScapes dataset for 3 epochs of 24000 training steps with batch size 1 and a learning rate of 0.0002 for both the generator and the discriminator. When this training finished, it was trained for an additional 65 epochs on the NYU version 2 dataset. While VirtualScapes GANDepth does not outperform Eigen, it shows improved results from the original GANDepth model. The $\delta$ values are slightly higher, and absolute relative difference, square relative distance, linear and log RMSE, and the scale-invariant error have all gone down. Similar to all prior results, the scale-invariant error remains high.
Figure 17: Average (a) and median (b) depth values per image of the NYU version 2 dataset as predicted by ROI GANDepth \cite{72}. The x-axes of the scatter plots show the true depth values in meters from the camera and the y-axes of the scatter plots represent the predicted depth values in meters from the camera. Additionally, each axis is supplemented with a corresponding histogram that shows the distribution of depth values in meters in the corresponding datasets. The distribution of depth values is in line with all prior results, which indicate that the model is less accurate for less frequently occurring depth values. The smallest average depth value for the generated images is 1.44 meters and the largest average depth value for the generated images is 4.41 meters, whereas the smallest and largest average value for the target depth maps equal 1.01 meters and 6.76 meters respectively. The smallest and largest median depth values for the generated and target depth maps are 1.47 and 4.47; and 0.93 and 6.87 respectively.
Table 6: This table shows the mean and p-values from a binomial one-tail test on the predictions of Virtualscapes GANDepth and GANDepth on the NYU version 2 dataset. The predictions in the test set were binary encoded, where 0 indicates that GANDepth had a higher value, and 1 indicates that VirtualScapes GANDepth had a higher value. This means that if the models perform similarly, we expect a $\mu$ of 0.5. For the metrics $\delta < 1.25$, $\delta < 1.25^2$, and $\delta < 1.25^3$, $\mu$ was indeed larger than 0.5. For the metrics RMSE, AbsRel, SqRel, and RMSElog, $\mu < 0.5$, indicating that Virtualscapes GANDepth systematically outperforms GANDepth.

4.3 Experiment 3: Incorporating temporal information in a model for depth estimation from monocular images with a conditional Generative Adversarial Network

In this section we will examine the results of experiment 3. As explained in methods section 3.4, two simulations were done on a subset of chronological frames of the VirtualScapes dataset. The first simulation is done with the GANDepth ROI model and serves as the baseline model. The second simulation is done with the recurrent GANDepth ROI model. From now on we will refer to this second model as rGANDepth ROI. The results of the performed simulations are presented in table 7.
Table 7: Simulation results for the GANDepth ROI and rGANDepth ROI models on a chronological subset of the VirtualScapes data, as described in section 3.4. The GANDepth ROI model ran for 5 epochs of 24000 training steps with batch size 1 and a learning rate of 0.0002 for both the generator and the discriminator. The rGANDepth ROI model was trained for 0.5 epoch, which totals 12000 training steps with batch size of 1. Note however that rGANDepth ROI was trained with sequences of 12 images, which is why training time was significantly slower. The rGANDepth ROI model also had a learning rate of 0.0002 for both the generator and discriminator. The results indicate that the rGANDepth ROI model outperforms the GANDepth ROI model for each of the calculated metrics.

The results summarised in table 7 indicate that the rGANDepth ROI model outperformed the GANDepth ROI model. To investigate whether this difference was significant, a binomial test was executed for each of the metrics. As with previous experiments, each of the scores per image in the test set were binary encoded to represent whether rGANDepth ROI performed better than GANDepth ROI, with null hypothesis that the probability that the score for the rGANDepth ROI is higher than that of the GANDepth ROI is equal to 0.5. Similar to prior experiments the formulation of the alternate hypothesis depended on the metric evaluated, as some metrics indicate better performance with a higher value, whereas others indicate a better performance with a smaller value. A binomial one-tail test showed that the improvement of rGANDepth ROI was indeed significant (p < 0.05) for each of the metrics listed. The results of the statistical test are summarized in table 8.
Figure 18: Average (a) and median (b) depth values per test image of the VirtualScapes dataset as predicted by rGANDepth ROI [57]. The x-axes of the scatter plots show the true depth values in meters from the camera and the y-axes of the scatter plots represent the predicted depth values in meters from the camera. Additionally, each axis is supplemented with a corresponding histogram that shows the distribution of depth values in meters in the corresponding datasets.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Mean (μ)</th>
<th>p-value</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>threshold δ &lt; 1, 25</td>
<td>0.76</td>
<td>p &lt; 1 \cdot 10^{-8}</td>
<td>Higher is better</td>
</tr>
<tr>
<td>threshold δ &lt; 1, 25^2</td>
<td>0.71</td>
<td>p &lt; 1 \cdot 10^{-8}</td>
<td></td>
</tr>
<tr>
<td>threshold δ &lt; 1, 25^3</td>
<td>0.63</td>
<td>p &lt; 1 \cdot 10^{-8}</td>
<td></td>
</tr>
<tr>
<td>abs relative difference</td>
<td>0.36</td>
<td>p &lt; 1 \cdot 10^{-8}</td>
<td>Lower is better</td>
</tr>
<tr>
<td>sqr relative difference</td>
<td>0.40</td>
<td>p &lt; 1 \cdot 10^{-8}</td>
<td></td>
</tr>
<tr>
<td>RMSE (linear)</td>
<td>0.39</td>
<td>p &lt; 1 \cdot 10^{-8}</td>
<td></td>
</tr>
<tr>
<td>RMSE (log)</td>
<td>0.28</td>
<td>p &lt; 1 \cdot 10^{-8}</td>
<td></td>
</tr>
<tr>
<td>RMSE (log, scale inv.)</td>
<td>0.24</td>
<td>p &lt; 1 \cdot 10^{-8}</td>
<td></td>
</tr>
</tbody>
</table>

Table 8: This table displays the mean and p-values from a binomial one-tail test on the predictions of GANDepth ROI and rGANDepth ROI on a chronological subset of the VirtualScapes dataset. The predictions in the test set were binary encoded, where 0 indicates that GANDepth ROI had a higher value, and 1 indicates that rGANDepth ROI had a higher value. This means that if the models perform similarly, we expect a μ of 0.5. For the metrics δ < 1.25, δ < 1.25^2, and δ < 1.25^3, μ was indeed larger than 0.5. For the metrics RMSE, AbsRel, SqRel, and RMSElog, μ < 0.5, indicating that rGANDepth ROI outperforms GANDepth ROI significantly in each of the measured metrics.

An overview of the predictions from both models can be found in figure 18. As with previous experiments, we can see that both models have a tendency to regress to the mean. This is indicated by the fact that for rGANDepth ROI the amount of predictions between 5 and 10 meters are similar to the amount of predictions between 10 and 15 meters, while this distribution is more skewed in the ground-truth data. Furthermore, we see a slight curve downward that indicates that predictions tend to underestimate the depth values slightly.
5 Discussion

In this section the results for each of the experiments will be discussed separately, followed by a general discussion of the results and their implications, and suggestions for future research.

5.1 Experiment 1: Generating depth maps from monocular images with a conditional Generative Adversarial Network

In this experiment depth maps were generated from monocular images with the use of a conditional Generative Adversarial network. The goal of the experiment was to investigate whether and with what accuracy a cGAN would be able to predict depth from monocular rgb images, and to set a baseline result to measure improvements in following experiments.

As discussed in section 4.1, an optimizer was chosen for the discriminator based on a series of trial runs with different settings. Modifications to the generator optimizer were also attempted in preliminary runs, but this was found to destabilize the training process too much. This immediately highlights the main challenge of this project, which was to set training parameters in such a way that the training stability of the model was not compromised. As is indicated by the results in table 1, training the discriminator model with an Adam optimizer as was suggested in the paper by Isola et al. unfailingly led to mode collapse [31]. It could be that this problem was avoided in Isola et al.’s implementation because they simply had more training data available.

Another possibility is that the image translation problem of monocular rgb images to depth maps was more complicated than the image translation problem of edges to rgb images, or the aerial photos to Google maps images that were explored by Isola et al. This is supported by the fact that adding augmentations that introduced noise, such as random jitter, reduced the model’s accuracy. A possible method to solve this could be to increase the resolution of the training and test images. Although this would increase training time, it would preserve more data and perhaps even make augmentations that introduce noise possible to counter overfitting. Although overfitting is not a widely discussed issue in GANs, the generator model did show signs of overfitting after about 40 training epochs as performance on the test set declined after this point in time.

The final model named GANDepth was trained with an SGD optimizer in the discriminator. This led to slower convergence, but a stable training process. The results produced by this model are described in section 4.1. The results summarized in table 2 and figure 14 show that the model regresses to the mean, which leads to poorer predictions of small and large depth values. The model also performs worse than the baseline model by Eigen et al. on the same train/test split. It is important to keep in mind, however, that Eigen et al. was able to use 120k additional training images, while the GANDepth model was trained on the original dataset with 795 training images and random vertical flips. This means that GANDepth was able to achieve similar results to the baseline paper with much less training data.

Something that stands out in the results is that while most metrics display similar results, the scale invariant loss is much higher for the generative model GANDepth than for the Eigen et al. model. This loss measures the relationships between points in the scene, irrespective of the absolute global scale [12]. High scale-invariant loss therefore implies that the errors made by the GANDepth model were not consistent across predictions. This is not a good sign, as it implies that the model does not generalize well. It is possible that this inconsistent error is caused by the fact that the generator is successfully fooling the discriminator with images that do not model the underlying relation between rgb images and depth. This behaviour can be countered by
swapping discriminators during training time.

Manually inspecting the GANDepth predictions reveals that the model seems to particularly struggle with depths of thin objects that are attached to walls, e.g. a projection screen or a painting. The edges of the objects will be visible in output depth maps, but the overall object will be predicted as being closer than it is. Another noticeable feature of the model predictions is that the model cannot predict depth accurately for windows. This is understandable, as a window would in fact display variable depth depending on what is shown behind it, theoretically ranging from zero (a wall directly behind it) to infinite (the sky). Furthermore, inconsistent light conditions in rooms with windows complicate the problem. It is possible that more training data will facilitate accurate predictions for these specific cases.

5.2 Experiment 2: Generating depth maps from monocular image with hyperrealistic video game data from GTA V

This experiment tested whether pretraining a conditional Generative Adversarial Network on hyperrealistic video game data from GTA V would improve its predictions of depth maps from monocular rgb images. Unfortunately it was not possible to extract data from the game directly, which limited the possibility of simulating different environments and weather conditions. However, the Virtualscapes dataset still provided a considerable amount of data collected from the game [57]. Images from different times of day were tested, but results showed that the model struggled with predictions of images representing in-game nighttime. This seemed to mainly be caused by the sky, which is a large part of each of the images. It is therefore possible that predictions for images at nighttime would have been more accurate after exclusion of the sky pixels, which was a feature that was added later. As no nighttime images were included in the real-world dataset, however, it was better to exclude them.

Application of the ROI mask also proved to improve the prediction accuracy significantly (see: table 3. It is not surprising that this was so effective, as it taught the model a smaller range of depth distances and thereby prevented predictions of the large depth values that were excluded. This reduces the values of the error metrics significantly, even for the scale invariant error. When looking at the overall distribution of average and median depth values (see: figure 16), it’s also clear that the ROI GANDepth model predictions follow a similar range and distribution as the target depth maps. Similar to experiment 1, the predicted distributions are more narrow than the target distributions, showing that less frequent depth values are less easily predicted. This is inevitable, as a model would always regress to the mean values.

Another noticeable feature in the depth distributions is that there is an over-representation of the amount of pixels with value 0. The model learns this bias and overestimates the amount of depth predictions at 0. This problem could also be solved by generating a more balanced dataset with the use of ROI masks.

As discussed before, transfer learning in GANs is an open research topic. The implementation of a GAN consists of three networks with corresponding weights: the discriminator network, the generator network and the placeholder network that connects them. When applying transfer learning in a CNN, the weights are copied and the higher layers may be frozen while the lower layers are trained [50]. This is done because higher layers are thought to represent more general image features, while lower layers could represent more specific image features. This approach was not successful when retraining a cGAN from the Virtualscapes dataset on the NYU version 2 dataset. Specifically, training stability was lost as the discriminator quickly outperformed the generator. A likely explanation is that the NYU version 2 dataset contains a much smaller range of depth (1-10 meters) than the Virtualscapes dataset (0-200 with ROI mask), which makes the

41
generator model worse at predicting smaller differences in depths. This would explain why the discriminator model finds it easy to distinguish between the real and generated depth maps. This problem was not resolved when initialising a new discriminator and retaining the generator weights, even though the generator model utilizes a higher learning rate and ADAM optimizer. This suggests that although the problem formulation remains the same, the two datasets are different. It is possible that a transfer from the Virtualscapes dataset to an outdoor dataset with similar depth range such as KITTI would allow the generator to learn faster.

Another possibility is that the generator has overfitted on its discriminator. This phenomenon has been reported in previous research and occurs because the generator uses the discriminator weights to optimize its own weights [1, 77]. One way to combat this is to swap between several discriminators during training time, so this is something that could be done in future projects.

Alternatively, the model is simply unable to capture the transformation from monocular images to depth maps. It could be that the problem is so complex that the model would require more layers or many more training examples to learn it. This is not unimaginable, as humans spend a lifetime learning depth cues that would allow them monocular vision, such as relative distances. Depth prediction from monocular images unfortunately remains an ill-posed problem, as we do not possess disparity information and rgb images can appear similar but correspond to different depth maps.

Despite these issues, the results summarized in table 5 show that the model significantly improves on all metrics when pretrained on the Virtualscapes dataset, even when only the weights are used for initialization and the datasets have different depth ranges. The model is still not able to outperform the Eigen et al. model, but is able to make more accurate predictions on the NYU version 2 dataset than the GANDepth model that was not pretrained. Additionally, the pretrained model was able to reach an equilibrium faster. As mentioned before, these improvements may have been larger when transferring to an outdoor dataset with similar depth range. This kind of dataset could be derived from the benchmark KITTI dataset by converting it to a monocular setting, which fell outside the scope of this project.

When examining the distributions of the depth predictions of the pretrained GANDepth ROI model on the NYU version 2 dataset, it is interesting to see that it shows a slightly wider range of depth values than GANDepth, as well as a slight shift to larger depth values. This is most likely a consequence of the wider (and large) range in depth values in the Virtualscapes dataset.

The improvement of the model predictions support prior research indicating that videogame data is realistic enough to transfer to a real world dataset. This means that the subquestion Q1c, posed in section 2.5 ”Can we improve the results of a depth estimation network by pre-training on hyperrealistic video game data?” is answered positively. It seems that realistic video games such as GTA V provide a source for extra depth data that can be utilized to improve model performance. This is also promising for other areas in computer vision that require more data, such as image segmentation. The only drawback of using videogame data is that it is difficult to extract this data from the game. Game companies do not often facilitate data extraction, and community code is not updated and typically lacks documentation. Computer science researchers should therefore perhaps directly approach game companies for access to game data. If financial compensation is expected, it would most likely be less expensive and time-consuming than collecting and annotating a large real-world dataset.
5.3 Experiment 3: Incorporating temporal information in a model for depth estimation from monocular images with a conditional Generative Adversarial Network

The goal of this experiment was to test whether adding temporal information to a conditional generative model of depth estimation would improve its performance. As we discussed in section 3.1, motion parallax is an important monocular depth cue for humans. Rogers and Graham even showed that it can be used to estimate depth independent of other depth cues [64]. Therefore, we expected a model trained on video data to perform better than a model that did not observe any context from its input image. Additionally, we expected a model trained on video data to further improve its predictions by reducing artifacts and inconsistencies across frames in its depth predictions.

To investigate if this was the case, a recurrent conditional Generative Adversarial network was trained on video data acquired from the VirtualScapes dataset [57]. The model referred to as rGANDepth ROI was trained on sequences of 12 frames that were taken from 2 seconds of in-game video recordings. Longer sequences and higher frame rates were considered, but did not seem feasible with the long training times required. Despite the relatively low frame rate, rGANDepth ROI outperformed the baseline model without recurrency that was trained on the same dataset. This suggests that temporal information is indeed valuable in learning the relation between monocular rgb images and depth maps.

Although the results are promising, there is perhaps less improvement than expected. rGANDepth ROI significantly outperforms GANDepth ROI on the test set, but the difference in scores is not very large. As mentioned before, rGANDepth ROI was expected to improve by learning motion parallax and being able to average inconsistent depth predictions across frames. It is possible that 2 seconds of data in combination with the low frame rate was simply not enough to learn the relation between depth and motion. This could mean that the model only benefited from the chronological frames by being able to generate more consistent predictions.

Another possible reason that the improvement of rGANDepth ROI was smaller than expected could be that GANDepth ROI had a slight advantage. Both the rGANDepth ROI model and the baseline model were trained on the same dataset with a similar amount of images, but the baseline model received single random training samples while the rGANDepth ROI model received random sequences of images. Since scenes somewhat overlap within 2 seconds of data, the baseline model has most likely received more variable input data, which should theoretically be beneficial to its performance on the test set. It seems that the temporal information is informative enough to negate this effect, but this could explain why the difference in performance is relatively small. This phenomena could be tested by running the baseline model with the exact same training images as the temporal model received.

The distributions of predictions displayed in figure 18 show that both the GANDepth ROI and rGANDepth ROI model regress to the mean, which results in less accurate predictions for depth values that occur less frequently. This is not surprising, as it was also visible in the results of the previous two experiments. Furthermore, it seems that for this same reason the models have a tendency to underestimate the depth values more often than overestimate.

In terms of absolute scores on each of the metrics, both GANDepth ROI and rGANDepth ROI performed better than the GANDepth ROI model that was trained on the VirtualScapes dataset in experiment 2 (see: table 3). This is probably due to the fact that in this experiment the data from the test set was more similar to the data from the training set than in experiment 2. Additionally, the test images themselves were more similar to each other because they consist of chronological frames. This means that if a
model happened to perform well on a test image it would also do well on the images that are similar. We tried to prevent this by acquiring the test data and training data from 4 different cameras, but because of the nature of the data it’s not possible to prevent these similarities completely.

5.4 General Discussion

In the previous chapter we have discussed the results from each of the experiments separately. In this section we will address the original research questions stated in section 2.5. The goal of this work was to answer the following main research question: Can a state-of-the-art neural network be trained to accurately estimate relative depth in a crowded scene from monocular recordings of a single uncalibrated camera? This question was explored through three sub-questions.

Our first sub-question asked: Does the incorporation of biological cues improve the performance of a single image depth estimation model? To answer this question, we implemented and tested a conditional generative adversarial network to estimate depth from monocular rgb images. As discussed before, a cGAN should be more context-aware than a traditional CNN. This makes it a more plausible candidate from a biological standpoint. It should also help the network learn relative depth and the relation between certain objects and depth, which is prior knowledge of the world that humans use to estimate depth. In addition to this, we further attempted to utilize biological cues by introducing time into the model. This would allow the model to learn the relation between motion and depth, a biological cue known as motion parallax. From these two experiments and their results, we can conclude that biological cues may indeed improve the performance of a single image depth estimation model. While the cGAN model could not reproduce the results on the real-world dataset from recent papers that utilize traditional CNN, it needed only a fraction of the data that these papers used [12, 25, 4]. This indicates that with more data and some adjustments, there is a potential to outperform these traditional models. As hypothesised, training with video data also improved the model’s performance. This further strengthens the belief that borrowing from biology can help improve models of depth estimation.

This also leads to the answer of our second sub-question: Can we improve the results of a depth estimation network by including temporal information? The results from experiment 3 (see: section 4.3) indicate that inclusion of temporal information improves network performance on depth predictions. As discussed in our answer from the previous sub-question, we hypothesize that this is at least in part due to ability of the network’s ability to learn the relation between motion and depth. We additionally speculate that by adding temporal information the network is able to remove inconsistencies from its depth predictions across frames. As discussed in section 5.3 however, the improvement we saw from the addition of temporal information was less than expected. Several possible reasons were listed, but more research would need to be done into why this was the case. The most straightforward starting point would be to experiment with different frame rates and sequence lengths. This was unfortunately not possible within the scope of this project.

Our last sub-question concerned the addition of video game data: Can we improve the result of a depth estimation network by pre-training on hyperrealistic video game data? Acquisition of data for machine learning from video games is relatively unexplored topic, but the results we have seen are quite positive. Although transfer learning with (conditional) GANs is an open research question, we were able to improve the performance of our cGAN by training the network on the synthetic VirtualScapes dataset, gathered from the hyperrealistic video game GTA V [57]. The improvement we observed is in line with previous work on this topic that was done with traditional CNNs.
We therefore believe that as video games are becoming more realistic, researchers should explore this option more. If the use of video game data becomes more prevalent in research, there is also an opportunity for scientists and game developers to work together to improve accessibility to these resources. This would be beneficial to the research community, as particularly in computer vision, the research is driven by the available data, due to the costs and time associated with data acquisition and annotation.

Now that we have answered all of our sub-questions, we return to our main question: Can a state-of-the-art neural network be trained to accurately estimate relative depth in a crowded scene from monocular recordings of a single uncalibrated camera? While we have seen that we can improve the performance of a state-of-the-art neural network on depth estimation from monocular rgb images, our results do not show the prediction accuracy required for an application that can reliably estimate depth in a crowded scene from monocular recordings. It is possible that our generative model could be improved with more data, but estimating depth from monocular images remains an ill-posed problem. It is therefore unclear if the desired accuracy could be achieved. However, as mentioned before, this work was made in collaboration with Info Support B.V. towards the goal of improving a privacy-preserving pipeline. The purpose of this system is not to estimate depth online, but to archive existing monocular video data. We therefore think that for this application it may be feasible to experiment with additional (outdoor) data and post-processing techniques to improve the depth estimation. For instance, Eigen et al. utilizes a separate network to align pixels of object boundaries. A similar concept was very recently developed by Luo et al., who developed a geometric algorithm that forced predictions in monocular video to be consistent across frames. A similar but more naive approach would be to combine object segmentations and tracking methods with depth predictions to force depth consistency across objects over time.

For the specific application of archiving monocular video data in compliance with privacy regulation, perhaps another possible solution could also be to create a 2D anonymous reconstruction instead of a 3D anonymous reconstruction. This could be done with e.g. an edge detector to draw the lines representing people and other objects of interest, and would eliminate the problem of estimating depth from monocular images entirely.

### 5.5 Suggestions for further research

Some suggestions for further research were already discussed in the previous section. There are a couple of topics, however, that were not yet mentioned.

First of all, each of the experiments done in this work used images of 256x256 pixels. While this reduced the training time, it is possible that this resolution leads to too much data loss. It would therefore be interesting to experiment with different image resolutions and augmentations to see if this improves the network’s performance.

Secondly, we discussed biological cues of depth perception in section 3.1. There are some, however, that we did not try to implement. As context and temporal data seemed to be beneficial to the network, the addition of other depth cues might also improve a network of depth estimation. Chen et al., for example, released a dataset of relative depth that could be used for training.

Lastly, it would be interesting to explore the findings of this work with different architectures, such as the new StyleGAN or even traditional CNNs. As the pix2pix network that was used benefited from adding biological cues, temporal information and additional (video game) training data, it would be good to see if these findings are reproducible in combination with different networks.
6 Conclusion

This work explored the possibilities of estimating depth from monocular rgb images with a state-of-the-art conditional Generative Adversarial Network. This is an ill-posed problem, because depth is derived from disparity, which requires stereo data. The purpose of this work was to improve depth estimations within a privacy-preserving 3D reconstruction pipeline for a client of Info Support B.V.. However, on a range of 0 to 200 meters, our model has an error margin of approximately 11 meters. We therefore did not achieve the accuracy required for reliable depth predictions in a crowded scene.

We did find that a cGAN is able to achieve similar results to a traditional CNN with significantly less data. Furthermore, we showed that the addition of biological cues and temporal information improves the accuracy of depth predictions of a cGAN and may also improve the performance of traditional CNNs. Lastly, we showed that it is possible to significantly increase the accuracy of depth predictions by a cGAN through transfer learning from hyperrealistic video game data to a real world dataset. We therefore believe that there is a lot of unexplored potential in this area for computer vision research problems in general.
References


