UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

VIDEO OUTPAINTING USING CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

BY

BRAIDEN A. MAGGIA
Norman, Oklahoma
2019

VIDEO OUTPAINTING USING CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS

A MASTER'S THESIS APPROVED FOR THE GALLOGLY COLLEGE OF ENGINEERING

BY THE COMMITTEE CONSISTING OF

Acknowledgements

Thank you to my thesis committee members, Dr. Grant, Dr. Jabrzemski and Dr. Hougen. I would like to thank my wife and parents for encouraging me to pursue my goals no matter what they are. I would also like to thank my high school technology teachers Paul Wollenberg and Mike Wilson for sparking my interest in computer science.

Abstract

Recent advancements in machine learning and neural networks have pushed the boundaries of what computers can achieve. Generative adversarial networks are a specific type of neural network that have proved wildly successful at content generation tasks. With this success, filling in missing sections of images or videos became a research topic of interest. Research in video inpainting has made steady progress throughout the years focusing on filling missing content in the center of a frame while research on video outpainting, which focuses on filling missing sections on the edge of the frame, has not. This thesis focuses on outpainting research by using conditional generative adversarial networks (cGANs) which apply a condition, such as an input image, to a generative adversarial network (GAN) in order to reformat traditional 4:3 video into a modern 16:9 format. This is accomplished by using a cGAN typically used for image-to-image translation and adapting it to generate the missing content from video frames. Although generated frames are not capable of accurately reconstructing missing content, this process is capable of producing context aware video that many times seamlessly blends with the original frame. The results of this research provide a glimpse of the possibility of using conditional generative adversarial networks for video outpainting.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this section an overview of previous work is discussed, the research objective for the thesis is presented and the results from experimentation are briefly noted.

1.1 Overview

Machine learning has been used to solve many problems that would be incredibly hard or impossible to solve with traditional algorithms. Many approaches to machine learning, such as neural networks, were formed due to the wide range of problems to be solved. Originally based off of the human brain, *neural networks* can feature multiple layers and thousands of neurons interconnect together to form a neural network (Engelbrecht, 2007). The adaptable nature of neural networks encouraged broad research into various architectures and different forms of learning. Neural networks can now be found as the industry standard for many machine learning problems.

With computational power increasing each year, neural networks have become increasingly complex, featuring networks with many hidden layers each containing thousands of neurons. These complicated neural networks formed a new sub-category called *deep neural networks* and with them more complicated problems were solved (Goodfellow et al., 2016). Eventually, convolutional neural networks were developed which made groundbreaking progress in problems such as computer vision. These networks featured a smaller layer which is sparsely connected to the previous layer, forcing the layer to focus on whether a feature is present instead of where it is (Goodfellow et al., 2016).

Generative modeling tasks such as image generation were difficult for traditional neural networks as the network had to model an extremely complex output space. This changed with the development of generative adversarial networks (Goodfellow et al., 2016). *Generative adversarial networks* (GANs) are able to take a random input vector and consistently produce a realistic image from it due to a separate network that was trained to detect real images from fake and feed this information back to the generator to learn from. Due to the modular nature of generative adversarial networks, they have been adapted to a wide range of generation problems, as any network can be used for the discriminator or generator. This has led to an abundant amount of research to occur in the past couple years adapting the generator discriminator architecture to a wide range of problems (Goodfellow et al., 2016).

Image inpainting is a class of problems focused on filling missing sections of an image with a content aware result. Due to the complexity of images, neural networks quickly became the favorite method of addressing such problems. Over the years, many advances in image inpainting have been made with each result becoming progressively realistic. Some solutions applied generative adversarial networks conditionally by inputting a conditional input such as an input image as well as a random vector to fill the missing sections with impressive results (Isola et al., 2017).

Video inpainting built off of image inpainting but has only been a popular research topic for the past couple of years. Many of the same techniques used in image inpainting were applied to video with varying levels of success. The temporal nature of video led to research into flow-based video inpainting methods where missing pixels were generated by finding them in other sections of the video (Xu et al., 2019a). This has increasingly been the popular method for video inpainting as some of the results made have been astonishing. Flow-based video inpainting can be computationally expensive and is limited in its generative ability when pixels are found in other sections of video, thus making it a go-to method for only a subclass of problems.

1.2 Research Objective

Video inpainting research has focused on filling content in the center of the screen. This is especially true for flow-based inpainting methods, as the main purpose is filling missing pixels by finding them in other frames. Little research has been done on *video outpainting*, also known as video extrapolation, focuses on filling missing sections outside of the frame. Outpainting video is typically harder as there are little references in future frames and little context from neighboring edges. This is an interesting problem and could immediately be used for remastering classic films. Traditional film was shot at a 4:3 aspect ratio which when shown on modern 16:9 displays leaves a significant amount of the screen black. Flow-based methods would struggle to fill this section since the information rarely appears elsewhere, as most shots are either short or remain fixed. Due to this, a different method for video outpainting should be explored making generative adversarial networks a potential option. This thesis builds on a popular image-to-image translation framework that had success with image inpainting (Isola et al., 2017), with its architecture adapted to work with video. This method is not only a possible solution for generating video content, but is also a less computationally expensive method for video outpainting. This conditional generative adversarial neural network uses a U-Net based generator originally developed for biomedical imaging segmentation that can take an input image and translate it to a desired output image (Ronneberger et al., 2015). This image is then passed to the discriminator which incorporates a Markovian discriminator nicknamed PatchGAN that forces the generator to model crisp images by critiquing an image on the scale of patches (Li and Wand, 2016). The output of the discriminator is used to train the generator until the desired output is achieved. The original purpose of this cGAN was for a generic approach to image-to-image translation, which means depending on the film trained on, its output will vary (Isola et al., 2017). This has the benefit for being a general solution for a wide range of films.

1.3 Results Overview

The model trained on two films over the course of three days. The discriminator was then removed from the generator and a new film was passed to the generator for the results. The generator was able to produce context aware content for the edges of the film which in most cases blended seamlessly with the original frame. Due to the salient nature of video, most action occurs in the center of the frame taking attention away from the edges of the screen. Although the generated frames were not accurate to the original content, they blended well enough to not take away from the attention of a viewer. Many early attempts at video

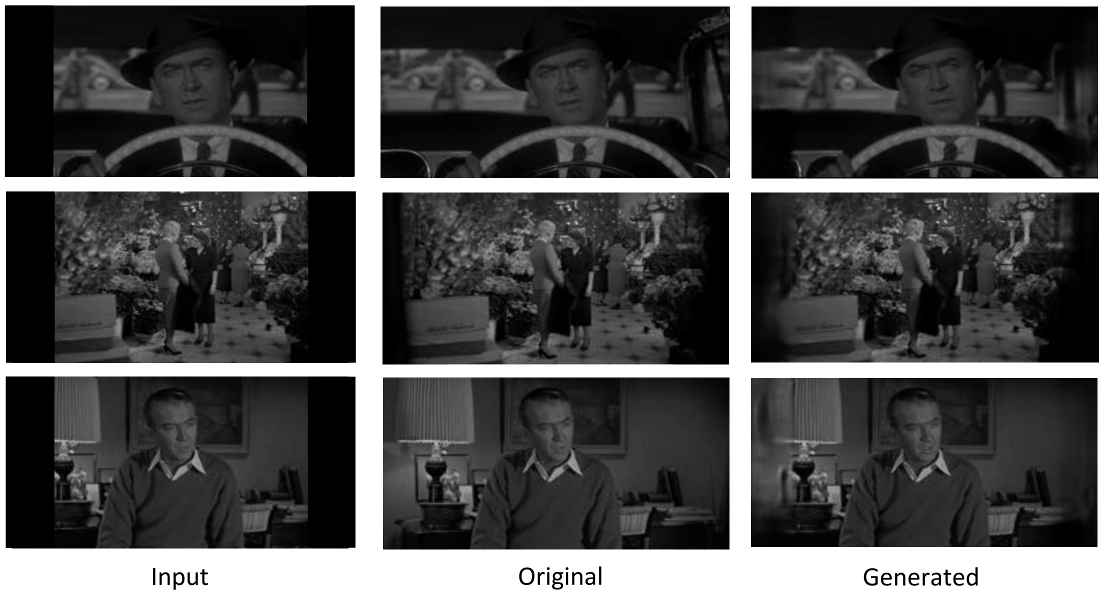

Figure 1.1: Random generated frames

inpainting resulted in temporally inconsistent frames that result in the generated sections warping over time. The generated content from this work rarely had this issue (see Figure A.1), but failed to accurately generate content moving on and off screen (see Figure A.5). Using structural similarity index measure (SSIM) and peak signal to noise ratio (PSNR) as measures for image similarity, the results were found to have a 27.451 PSNR, which has a range of 0-30, and a 0.8122 SSIM, which has a range of 0-1 when measured across the entire output frame. When measuring generated content only, the results had a 23.23 PSNR and a 0.5483 SSIM. These metrics show that the original image is correctly reproduced while the generated edges are context aware and consistent with the original frame. The quality of the results were surprising, especially given the short training time. With more training and modifications to the generator and discriminator to take advantage of the temporal structure of film an impressive result could be achieved.

Chapter 2

Related Work

This covers all related work that this thesis builds on. This includes artificial neural networks, the learning process, differentiable generator networks, generative adversarial networks, outpainting techniques and flow-guided video inpainting processes.

2.1 Artificial Neural Networks

Artificial neural networks were originally developed and researched based on the anatomical construction of the human brain (Engelbrecht, 2007). The brain consists of billions of basic computational nodes called *neurons*. These are interconnected by *synapses* which allow signals to be communicated between different neurons. This structure creates a highly parallel biological computer that is capable of completing tasks, such as image recognition, faster than any computer. This has sparked research in artificially modeling the brain in computers, which results in the creation artificial neural networks. A neural network can be described as a nonlinear mapping from $\mathbb{R}^A$ to $\mathbb{R}^B$

$$f_{NN} : \mathbb{R}^A \rightarrow \mathbb{R}^B \tag{2.1}$$

where A and B are the dimension of the input and output target and f_{NN} is a set of nonlinear functions, or the neural network itself. These neural networks have gained popularity in recent years as more complex and dense neural networks are able to be created on modern computers. Even with the advancement of computers, the most complex artificial neural networks are still extremely small in comparison to the human brain.

2.1.1 Artificial Neuron

Artificial neurons are the basic building blocks of artificial neural networks (Engelbrecht, 2007). They are computer models of biological neurons and as such operate in a similar way. An artificial neuron is simply a nonlinear mapping from $\mathbb{R}^S$ to an output defined by an activation function defined on some range such as $[0, 1]$. This mapping is defined as

$$f_{AN} : \mathbb{R}^S \rightarrow [0, 1] \text{ or } [-1, 1] \tag{2.2}$$

where S is the number of input signals to the artificial neuron and the output $[0, 1]$ or $[-1, 1]$ changing depending on the activation function chosen. Some of these activation functions are inclusive, where they are able to reach their bounds while others are exclusive and are not able to. This input can be thought of as a vector of S input signals. These signals could come from the input of the neural network, or from other neurons. Each input signal is typically given an associated

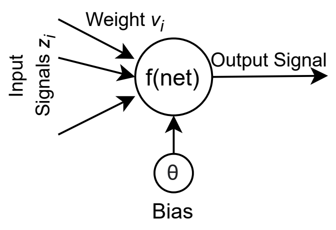

Figure 2.1: Artificial neuron

weight to determine the importance of that input signal to the current neuron. The neuron computes the net input by combining all of the vectors and is then influenced by a threshold value θ also called the bias. The signal is then applied an activation function to compute its output signal. This computation can be visually depicted in Figure 2.1.

The net input signal can be computed by a weighted sum referred as *summation units*.

This weighted sum is formally defined as

$$net = \sum_{i=1}^{S} z_i v_i \tag{2.3}$$

where S is the number of input signals, z_i is the input signals and v_i is the weight of that input signal.

After combining the input signals, an activation function is used to shape the output of the neuron. There are many activation functions, but typically are monotonically increasing mappings where

$$f_{AN}(-\infty) = 0 \text{ or } f_{AN}(-\infty) = -1 \text{ and } f_{AN}(\infty) = 1. \tag{2.4}$$

Rectified linear unit (ReLU) and leaky rectified linear unit (Leaky ReLU) activation functions are exceptions to this. ReLU is bounded $(0, \infty)$

$$f_{AN}(net) = max(0, net) \tag{2.5}$$

and leaky ReLU is bounded $(-\infty, \infty)$

$$f_{AN}(net) = \begin{cases} \beta net & net < 0 \\ net & net \geq 0 \end{cases} \tag{2.6}$$

with negatives values having a lesser slope due to a constant β being applied.

A hyperbolic tangent function is specifically used in a layer of the model and is bounded by $(-1, 1)$ defined as

$$f_{AN}(net) = e^{net} - e^{-net}. \tag{2.7}$$

2.1.2 Feed Forward Networks

Feed-forward neural networks are the basis of many other neural networks. They are the basis of convolutional neural networks and are used for many commercial applications. They are called feed-forward networks as the data from the input moves from the input layer to any number of hidden layers until it reaches the final output (Engelbrecht, 2007). Unlike other neural networks, a feed-forward network does not have any feedback connections. Feed-forward networks are considered neural networks as they are roughly inspired by neuroscience. The output of a feed-forward neural network given an input is calculated with a single forward pass through the network layers. This is done by computing the output signal for each neuron in a layer, moving forward through the network after each layers' computation is complete. This creates the nonlinear mapping (2.2).

2.1.3 Convolutional Neural Networks

Convolutional neural networks have been used to make massive strides in areas such as computer vision (Goodfellow et al., 2016). A *convolutional neural network* is simply a neural network that uses a mathematical operation called convolution (defined in Figure 2.2) in one or more of its layers. Traditional neural network layers use matrix multiplication to combine the input signals with the parameter weights. This matrix multiplication operation describes the relationship between

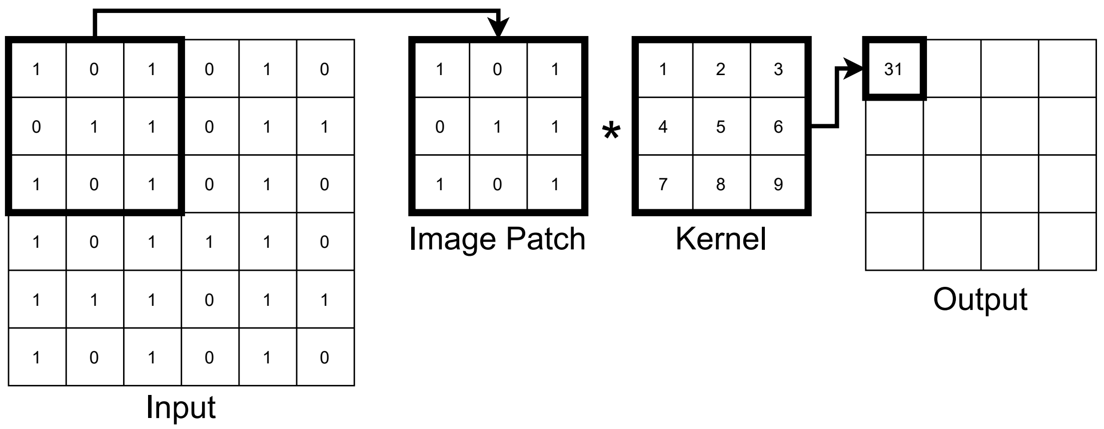

Figure 2.2: Convolutional layer

each input and output unit (2.3). Thus, in standard feed-forward neural networks, all output units interact with all input units no matter the parameters weights of the connections. Convolutional neural networks do not have the same approach and are instead sparsely connected. This is accomplished by making a layer, called the kernel, smaller than the input (2.2). This results in a layer that can detect meaningful features in an image by looking at smaller sections of the input and finding meaning in these sections. With fewer parameters to be trained, the neural network has a higher statistical efficiency, as well as requires fewer operations and less memory. Typically, the input and kernel are multi-dimensional arrays of parameters that will be modified by the learning algorithm with entries that are not explicitly stored resulting in 0. This allows an equation for the convolution to be defined as a summation over the number of array elements. This convolution with an 2-D input image I, a 2-D kernel K and an output S can be written as

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{2.8}$$

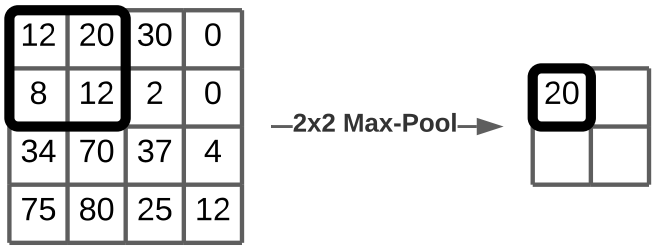

Figure 2.3: Max pooling example

A convolution layer can also apply a stride to manipulate when the convolution operation is computed. A *stride* dictates how many pixels the image patch should move by after completing an operation. For example, when the stride is set to 1, the convolution will be applied each time after moving a pixel. If the stride is set to the same size as the image patch, no duplicate pixels will be included for each convolution operation. Applying a stride to a convolution layer changes the output layer size as less cells are being computed.

Convolution layers are typically paired with a pooling stage. This *pooling stage* modifies the output by instead taking a summary statistic of nearby outputs (2.3). For example, the *max pooling* operation reports the maximum output found in a rectangular neighborhood. Other pooling operations include averaging the rectangular neighborhood or taking the L^2 norm on the neighborhood. Pooling helps the output invariant to small translations of the input. This gives precedence to an output that is more focused on determining whether a feature is present instead of where it is located, which is why convolutional neural networks have been adopted and used in most computer vision tasks.

Convolution often needs to be reversed in some way, which is common in encoder-decoder models. Encoders compress an input to a specific size while a decoder does the opposite. This process is called *deconvolution* and is an inverse to convolution. The process of deconvolution is not always to recreate the original input as encoder decoder models have been used for many segmentation tasks which require an input to be transformed to an output.

2.2 Learning

Training a neural network is an entirely mathematical operation. There are multiple training methods used for neural networks, but supervised learning is the only method used in this paper. *Supervised learning* is the process of training a neural network given an input-target pair sampled from training data (Engelbrecht, 2007). This generates an unknown function that the neural network must create to successfully transform an input to an output. There are many ways to optimize a neural network to learn this transformation, but the most popular method is gradient-descent optimization. Gradient-descent optimization occurs in two stages, a forward pass of the neural network that calculates the output from an input training pattern, and a backward propagation pass which sends an error signal back through the neural network. Weights for individual neurons are then adjusted using this signal.

2.2.1 Gradient Based Learning

Gradient-based learning attempts to minimize a function $f(x)$ by altering x (Engelbrecht, 2007). This function is typically called the *objective function* as the function represents the goal to be learned, but can be called other names such as the cost/loss function. When optimizing a function $f(x)$, a gradient vector is taken $\nabla f(x)$, computed by back-propagation (defined in Subsection 2.2.2). This gradient vector points toward a local maximum, so taking the negative of this points toward the local minima $-\nabla f(x)$. When optimizing neural networks, finding a good local minima is typically the goal, as finding the global minimum can be extremely difficult, time consuming and resource intensive. After calculating the gradient vector pointing toward the local minima, a step is taken toward that direction $\eta \nabla f(x)$ where η is the learning rate. The *learning rate* is a scalar that changes the size of the steps taken toward the local minimum. Choosing the correct learning rate is important, as steps that are too small will result in learning taking excessive time, while too large of a learning rate might skip over local minima.

2.2.2 Back Propagation

When a training set is sent through a neural network an output is computed. This step is called *forward-propagation* and once it is computed an error signal is created and sent through the network to adjust weights of individual neurons (Engelbrecht, 2007). This second pass back through the network is called *back-propagation*. Back-propagation is required to compute the gradient and ultimately learn from the training data. Back-propagation specifically uses the chain rule of calculus to compute the derivatives of functions by composing other functions of known derivatives. This is done by computing the gradient of each layer one by one, working through the network.

2.3 Differentiable Generator Networks

Differentiable generator networks are designed to generate data. For example, if a neural network is designed to take input images and generate labels, a differentiable generator network would perform the opposite task: take a label and generate an image. Differentiable generator networks are the basis of generative adversarial networks. The generative model transforms samples of a latent variable u to samples or distributions using a differentiable function $g(u; \theta^g)$ (Goodfellow et al., 2016). This differentiable function is typically a neural network. Differentiable generator networks can be thought of as a means of generating samples with the architecture of the generator dictating the possible distributions to sample from. The parameters fed to the generator then select a sample from that distribution. Generating samples from complicated distributions which are difficult to integrate over require a feed-forward network to represent a family of functions g whose parameters are set by training data. Differentiable generator networks were created from the success of feed-forward networks and classification using gradient descent. Generative modeling has been found to be more difficult than classification as it must learn a much larger output space. When generating data, the data does not specify the inputs and the outputs while typical supervised learning does and thus only needs to learn how to produce the mapping. With generative modeling, the generator must find and arrange an input space in a specific way as well as learn how to map the input to the output. To have success using a differentiable generator network, a better approach for training when only given an input must be used.

2.4 Generative Adversarial Networks

Generative adversarial networks (GANs) are a generative modeling method based on differentiable generator networks (Goodfellow et al., 2016). Generative adversarial networks attempt to fix the issue of training only given an input by pairing

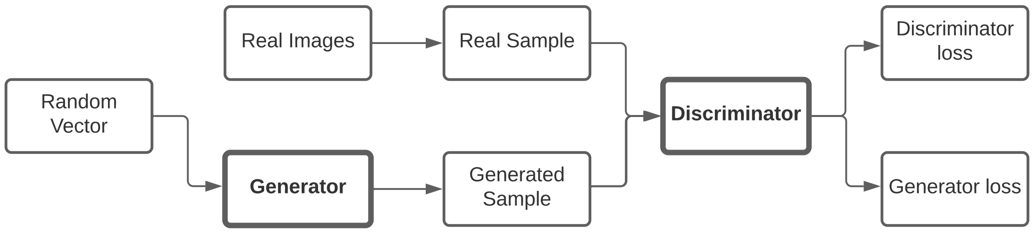

Figure 2.4: Generative adversarial network

the generator network with a discriminator network that judges its output (2.4). This is done by a game in which the generator and discriminator compete. The generator produces samples directly like a differentiable generator network, producing samples $x_g = g(u; \theta^{(g)})$. The discriminator network then attempts to determine between samples from the generator and samples drawn from the training data. The discriminator produces a probability value $d(x_g; \theta^{(d)})$ indicating the probability that x_g is a real training example rather than a fake produced by the generator. Learning can then be done by a zero-sum game where $v(\theta^{(g)}, \theta^{(d)})$ determines the payoff for the discriminator. The generator then receives payoff $-v(\theta^{(g)}, \theta^{(d)})$. The generator and discriminator compete for the higher payoff until convergence where the samples produced by the generator are indistinguishable from real data and the discriminator is always outputting $\frac{1}{2}$, as it can no longer tell the difference. Once training is complete with a generative adversarial network, the discriminator can be discarded and the generator can be used for further samples. Generative adversarial networks are designed to learn any distribution, but research has indicated that convergence can be an issue unless parameters are correctly set when designing the model.

2.4.1 Conditional Generative Adversarial Networks

Conditional generative adversarial networks (cGANs) are very similar to normal GANs, but instead of learning a generative model, it learns a conditional generative model (Isola et al., 2017). This is necessary for tasks in which the output needs to reflect an input, such as inpainting or outpainting. While generative adversarial networks typically take a random input vector and map it to an output $G : z \rightarrow y$, conditional generative adversarial networks instead learn a mapping with a condition x, and a random noise vector z to an output y, $G : \{x, z\} \rightarrow y$. Due to the modular nature of generative adversarial networks, the discriminator can be left unchanged and training can remain the same.

2.4.2 U-Net Generator

The *U-Net convolutional network* is a specific type of encoder-decoder network originally developed for biomedical image segmentation (Ronneberger et al., 2015). An encoder-decoder network has a series of convolutional networks that slowly downsample the input until a bottleneck layer, where the process is reversed using deconvolution to output a high resolution image. This architecture encourages semantic image segmentation which is why it has been adapted for image-to-image translation. The unique feature of a U-Net convolutional neural network is the addition of skip connections between layers of equal size. *Skip connections* are applied by taking the corresponding layer from the encoder and appending it to the decoder. In a typical encoder-decoder network, information can be lost when going through the bottleneck. Due to image translation using much of the same features between its input and output, sharing information between the downsampling layers and upsampling layers helps the output more accurately reflect

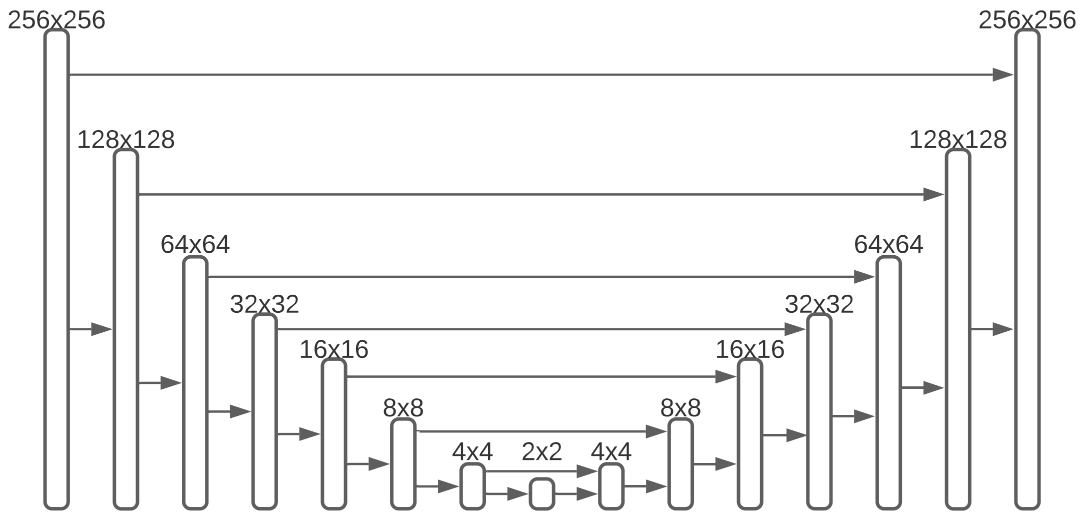

Figure 2.5: U-Net

the input.

2.4.3 Markovian Generative Adversarial Networks

Image synthesis problems typically encounter the same issue: capture the complex structure of images in a learnable and efficient way. Most have tackled this common problem by using Markov random field models that can statistically characterize an image by patches of local pixels (Li and Wand, 2016). Markov random fields are similar to a Bayesian networks in their representation of dependencies with the main different being Bayesian networks are directed and acyclic while Markov random fields are undirected and may be cyclic. Most Markovian models are able to produce surprising visual results but are computationally expensive to produce. In traditional deep Markovian models, iterative back-propagation is carried out to estimate each pixel causing a low resolution image to take minutes to synthesize. This can be fixed by applying a strided convolutional network resulting in a feed-forward model that is able to synthesize images 500 times faster and at the same quality. Using this model as a discriminator has numerous benefits over traditional L2 and L1 loss. L2 and L1 loss generally produce blurry results for image generation problems as they tend to only capture low frequencies. Markov random fields are able to model high-frequencies making a Markovian model suitable for image generation where crispness is needed.

2.5 Image Outpainting

Many image outpainting methods have been developed based on successful image inpainting techniques. All techniques discussed use generative adversarial networks to create the painted sections. Ying and Bovik (2020); Krishnan et al. (2019); Xu et al. (2019b); Xiao et al. (2020); Guo et al. (2019); Yang et al. (2019) all use an encoder-decoder network with skip connections between layers similar to a U-Net (Ronneberger et al., 2015). Each paper describes that using the skip connections results in a higher quality image with many showing a comparison between adding skip connections and removing them. Krishnan et al. (2019) specifically uses the image-to-image translation described by Isola et al. (2017) with the addition of blending technique to smooth the transition between real and generated. Mastan and Raman (2021); Wang et al. (2019) attempt to solve the image outpainting problem by finding context features in an image using an encoder-decoder network then generating the image using a generative adversarial network. Lin et al. (2021); Xu et al. (2020) both use the addition of an edge completion network to generate missing edges. Lin et al. (2021) uses a three-step process that starts with a rough output created by an image-to-image translation network that is then fed into an edge-prediction network. The course output and predicted edge are then fed into a final network that refines the output. Xu et al. (2020) similarly uses an edge prediction network for image outpainting; however, the edge is computed without the use of a rough output. The output from the edge-prediction network is then fed into a generative adversarial network where the output is created. Cheng et al. (2021); Guo et al. (2020); Yang et al. (2019) iteratively generate missing sections of an image by slowly expanding outward from the image by the scale of patches. After creating the entire image another network is then used to remove the seam between generated sections. Kim et al. (2021) uses an interesting outpainting technique by first splitting the image in the center and moving the halves oppositely toward the edges. The section in the middle is then generated using traditional image inpainting techniques with the output being split in the middle and reordered to its original position. Although this method produces surprising results, it generally only works for scenic images and would not correctly generate objects moving on and off screen due to its process.

2.6 Flow-Guided Video Inpainting

Flow-guided video inpainting attempts to fill missing pixels by finding them in a previous or future frame of video (Xu et al., 2019a). This is a complicated process which follows three steps. First, video is converted to a flow completion network which attempts to extrapolate the movement of objects in the scene. Naturally, the sections that need to be inpainted will not have information in this flow completion network, so the network iteratively fills this missing information making the section to patch smaller and smaller. Next, this flow-guided video is used to find missing pixels in other frames which are then propagated to fill the missing information. Using this method, propagated pixels are temporally consistent allowing for less warping and jitters. Again, this process of propagating pixels is done iteratively making the inpainted section of video smaller and smaller. When this process is complete, not all pixels will be found leaving some sections of video needing further inpainting. This moves to the third step where traditional image inpainting techniques are used to generate the missing content. This content is then propagated through the network using the previously computed flow-guided video, making this purely generated content temporally consistent. Again, this is done iteratively across the video until all frames are computed. Flow-guided video inpainting has the benefit of being very temporally consistent and in most cases is able to fill missing sections accurately. The process is most successful when the content to be filled is found elsewhere in the video, and is least successful when it is never shown. When generating missing content for still video shots, the network would be forced to generate the content for each frame as it could not use the flow completion network. The network has not been tested for these cases and given its traditional inpainting focused algorithm, it would likely fail to accurately extrapolate the frames. Flow-guided video inpainting can also be extremely resource intensive as it is both computationally and spatially expensive.

Chapter 3

Experimental Design

The conditional generative adversarial network used here for video outpainting is based on an image-to-image translation cGAN described by Isola et al. (2017). This conditional generative adversarial network uses a U-Net architecture for its generator, taking advantage of the encoder-decoder that encourages semantic image segmentation needed for image-to-image translation. It also uses the skip connections designed in the U-Net architecture to share similar information such as edges between the encoder and decoder layers. The use of this generator allows for a generic approach to any image translation tasks, including outpainting. The discriminator uses the PatchGAN architecture described in Isola et al. (2017). Due to the GAN needing to model high-frequencies, using a patch of size $N \times N$ that is run convolutionally across the input image is sufficient to create the output of the discriminator. Doing so allows for the discriminator to quickly form an output instead of analyzing the entire image, which would take considerable time. Training is completed by a 16:9 film with similar content as the film to be translated. This training film is then separated into frames and cropped to make training pairs. Once the training is complete on the similar film, the generator can be used to re-format the original film.

3.1 Generator

The generator uses an encoder-decoder with skip connections between the two. This architecture is called a U-Net architecture and it allows for semantic image segmentation which makes it a good fit for image translation (Ronneberger et al., 2015). The encoder uses convolutional layers with batch normalization to slowly downsample the input by a factor of two for each layer until the bottleneck layer. Batch normalization is used to help the training process by standardizing the inputs to a layer, thus stabilizing the learning process. A leaky ReLU activation is used with a slope of 0.2 for all layers in the encoder. Using this reduces the chances of the gradients becoming 0 causing the dying ReLU problem where the neuron will only output 0 and is unlikely to recover. There are eight layers in the encoder with each layer increasing the filters. The first layer is the input image, followed by a layer with 64 filters. It is followed by a layer with 128 filters, then 256 filters and finally four layers of 512 filters. The decoder picks up from where the encoder left off with some changes to each layer. The decoder consists of seven deconvolution layers each of which upsample the image by a factor of two. Deconvolution is completed by doubling the image size then running a convolutional filter to refine the image. Each layer uses batch normalization and dropout with a ReLU activation function instead of the encoder's leaky ReLU activation function. *Dropout*, which randomly drops nodes in both visible and hidden layers, is used in place of standard Gaussian noise for input into the generator as the generator tends to ignore the noise. Dropout is only applied in the encoder side of the U-Net. The dropout rate used in the decoder is 50%. The decoder begins with four layers of 512 filters, followed by a layer of 256 filters, 128 filters and lastly 64 filters. The final result is sent through a final convolutional layer using a tanh activation function which maps the number of output channels. Skip connections between the encoder and decoder layers are added with the exception of the bottleneck layer. These connections send information directly to the decoder layers and concatenate their activation functions to the decoder's activation functions. This architecture is shown in Figure 2.2.

3.2 Discriminator

The discriminator is only used when training the conditional generative adversarial network. Its architecture uses four convolutional layers with increasing filters for each. Similar to the encoder, each of the layers in the discriminator uses a convolutional layer with batch-normalization and a leaky ReLU activation function with a slope of 0.2. Before the first layer, the produced image and the conditioning image are concatenated to produce the input. This is then passed to a layer with 64 filters, followed by a layer of 128 filters, 256 filters and finally a layer of 512 filters. After the last layer, a convolution is applied to map the result to a 1-dimensional output which will be used for training. This is based on the architecture described in Isola et al. (2017) also called PatchGAN. It was designed with the intent to model high-frequency correctness as L2 loss was found to produce blurry results. The architecture penalizes structure of the scale of a patch with the average of all responses creating the output of the discriminator. This architecture has the benefit of modeling high-frequency structure while having fewer parameters and running faster than traditional Markovian discriminators.

3.3 Training

Training and testing was done using two Alfred Hitchcock movies, specifically *North by Northwest* and *Vertigo*. These movies were chosen as they were released in an aspect ratio of 4:3 as well as 16:9. When choosing a training set, the film also needs to be similar to the testing film as the model will only generate images similar to its training set. The films were both exported to frames at 20 frames per second, resized to 256x256 with both sets cropping them to include black borders. The training set also kept original images to be used with the discriminator. The conditional generative adversarial network trained on the frames for three days with only one epoch as it did not run out of training data.

Chapter 4

Results

Evaluating the output of synthesized images or video in a quantitative manor is an open problem. Ultimately the success of the output depends on people's ability to determine real from fake. With this in mind, there are rudimentary metrics that can be used to get a rough sense on the quality of the output versus the original. Signal-to-noise ration, or SNR, peak signal-to-noise ratio, or PSNR, and structural similarity index measure, or SSIM, can be used to quantitatively define how well a reconstructed image matches the original (Gonzalez and Woods, 2018). *PSNR* is a ratio between the information-bearing signal power and the level of noise. It uses mean-squared error but scales it according to the image range on a logarithmic scale as most images have a wide dynamic range. PSNR outputs a measurement representing the amount of noise in the image with a lower number meaning less noise and a larger number meaning more noise. *SNR* is the same as PSNR with the difference that SNR is relative to signal while PSNR is relative to the peak dynamic range. This makes PSNR a much better measure when comparing unrelated images. *SSIM* differs from PSNR as SSIM focuses more perceived visual quality. Due to this, SSIM is many times preferred

Generated Image vs Ground Truth (based on 5375 images)		
Signal to Noise Ratio	Peak Signal to Noise Ratio	Structural Similarity Index Measure
11.3967	27.4518	0.8122

Table 4.1: PSNR and SSIM results from an average of 5375 images

Generated Images vs Ground Truth (Generated Portion Only) (Based on 5375 images)		
Signal to Noise Ratio	Peak Signal to Noise Ratio	Structural Similarity Index Measure
5.6298	23.2305	0.5483

Table 4.2: PSNR and SSIM results (generated sections only) from an average of 5375 images

to over PSNR as a way of determining image or video quality. SSIM extracts three features from images: luminance, contrast and structure. These were chosen as the most important features an image have that people perceive. *Luminance* is calculated by averaging over all pixel values while *contrast* takes the standard deviation of all pixel values. *Structure* is based off the idea that spatially close pixels have strong inter-dependencies that make up the important structures of an image. This makes SSIM perform better for most image comparisons over PSNR.

The model was trained for three days on two Alfred Hitchcock movies, specifically *Rear Window* and *North-by-Northwest*. For training, the discriminator must have original images as well as generated. These films were chosen with this in mind as they have both 4:3 and 16:9 versions. After training, the model was used to generate frames from another Alfred Hitchcock movie, *Vertigo*. Once generating these frames they were compared against their originals using SNR, PSNR and SSIM and averaged for the 5,375 generated images. These statistics are typically used to measure video compression, with PSNR's numbers ranging depending on the bit-depth of the image. The generated images resulted in an average PSNR of 27 (4.1), with anything over 30 typically being the standard for lossy compression (Gonzalez and Woods, 2018) while SNR resulted in 11 with anything over 10 considered acceptable. SSIM is easier to measure as its range is between 0 and 1 with the closer to 1 being exactly the same and 0 meaning no similarity. The average SSIM for the generated images resulted in 0.8122. SNR, PSNR and SSIM was also computed for generated sections of the frame only, resulting in a 23.23 PSNR, 5.6 SNR and a 0.5483 SSIM (4.2). This shows that the generated frames were able to fill the cropped edges with a context aware result, but little results accurately recreated the frame. Images with simple objects near the edge like buildings or a car seat produced better results (see Figures A.1 and A.2) with the edges being contextually extended. Moving objects coming into and out of the frame were not reconstructed on the edges as to be expected as the model does not exploit the temporal nature of video (see Figures A.4 and A.6).

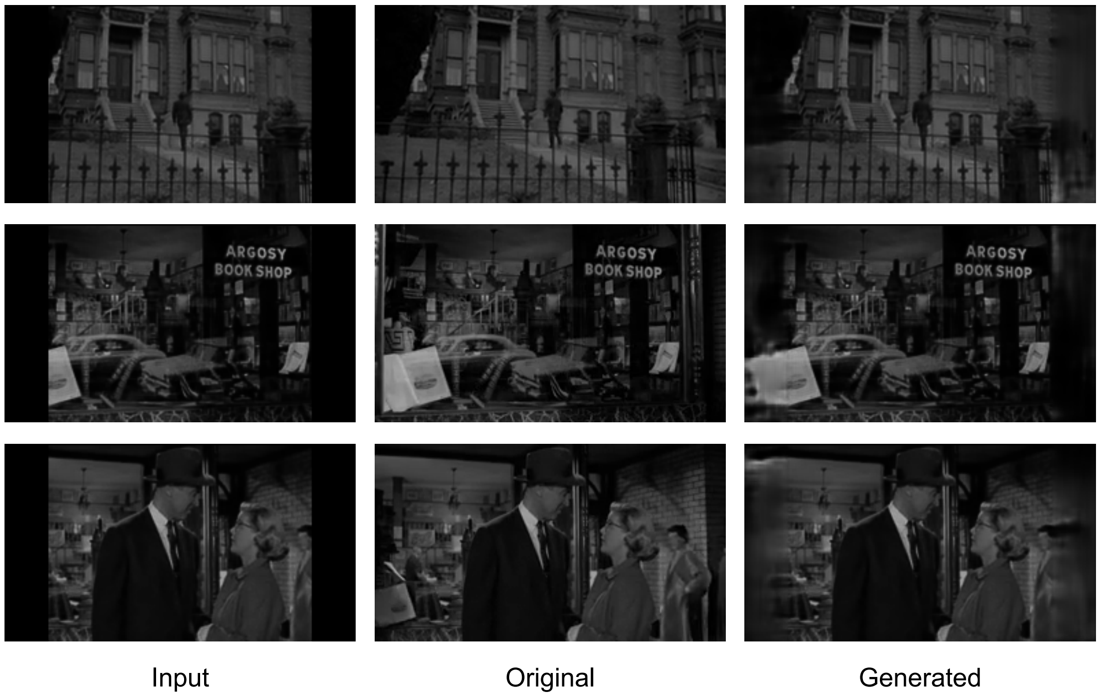

Figure 4.1: Input, generated and original images

Chapter 5

Discussion

Video inpainting has been an area of interest for researchers over the past few years with new techniques and advancements constantly evolving off of their predecessors. This research focuses on filling information with all edges giving context to the missing section. However, this thesis tested the possibility of reformatting video by generating the missing content on its edges. This posed a new problem unfit for many state-of-the-art methods, such as flow-based video inpainting relying on the missing section being revealed at some point in the frames. Many shots in movies lack this information as they are locked off with the information off screen never being shown. Image-to-Image translation proved to be a possible solution due to its adaptability, providing excellent results on a wide range of image translation problems. Although the generated content was not accurate to the original frames, it was able to fill the edges with a context aware result that did not distract from the rest of the video (see Figure A.2). Due to the salient sections of the image or video remaining intact, the viewer rarely notices inconsistencies toward the side of the screen. The human brain is wired to fill in information from the peripheral view of the eye, so generating context aware imagery proved less distracting than the typical black border in most instances. One exception to this case is when objects enter or leave the screen as their movements grabs the attention of a viewer and the model is unable to generate the content accurately (see Figure A.5). When training the model, it seems imperative to find a film related to the one to be reformatted as the model will learn to generate frames similar to its training set. If the training film does not have similarities with the target film, it will inaccurately reconstruct the frames and lead to undesirable results.

Many video inpainting methods create undesirable effects due to the generated sections warping through time. As the model used in this paper does not take advantage of the temporal structure of video or use previous frames when generating new content, warping content over time should be present in the generated frames. Surprisingly, it seems warping rarely occurs in generated content and in many instances is not distracting (see Figures A.1 and A.7). This is potentially due to many of the shots staying locked in place, allowing the model to generate a very similar result frame to frame. Further testing would have to occur to determine if warping occurs more in motion-heavy scenes.

A benefit to using cGANs for video outpainting over flow-based methods is the memory requirement. Flow-based methods must analyze previous frames in order to generate their content. In many cases, this leads to more accurate results but is computationally expense and requires a large amount of memory not found in most commercial GPUs. For computing high resolution video, using flow-based methods would be nearly impossible on consumer products. This was not the case with cGANs as the model used to generate the images was able to create multiple frames per second on a consumer GPU with the ability to increase the resolution without GPU memory running out.

Previous work in image outpainting shows similar results (Krishnan et al., 2019; Guo et al., 2020; Xu et al., 2020). Edges are filled with a context aware results which is typically blurred in comparison to the rest of the image. Certain outpainting methods achieve better results but are more specialized for specific generation tasks (Ying and Bovik, 2020; Yang et al., 2019; Cheng et al., 2021). As the variety of frames in a movie could be large, having a generic approach that produces reasonable results for most to all frames is preferable to a system that may achieve great results for some frames but poor results for others. All previous image outpainting methods have not been tested with video, so undesirable warping effects may be present. Most image outpainting techniques researched have followed a very similar architecture as used in this paper, so there is a high likelihood that other results would be similar to the ones found here.

Chapter 6

Conclusions

While video inpainting has been a popular research topic in recent years, the specialization of reformatting video which requires outpainting frames has not. Traditional flow-based methods might fail to accurately fill the edges of a scene, as it typically relies on the missing section of the image being shown at some point (Xu et al., 2019a). While this may not be an issue for most video segments, this can cause issues in film when the content may never be shown. Using a similar model to the cGAN proposed by Isola et al. (2017) for general image translation, a model was trained that was able to generate context-aware imagery to fill the edges of a film. Although it rarely achieved photorealistic results, due to the salient nature of video the generated sections proved to enhance the video enough where many times they were not noticeable to a viewer (see Figure A.8). The model was able to generate convincing results with most static shots and fell short when objects moved on or off of the screen (see Figure A.5). Many previous works with video inpainting found the temporal structure of video caused the generated sections to warp slowly over time creating distractions for an observer (Xu et al., 2019a). This effect did not occur to the same degree in the results for the method used in this thesis, most likely due to a lack of motion in most shots and the generated content being less detailed than traditional inpainting work making distortions less noticeable. The quality of results from the cGAN proved surprising, with better results being possible given a longer training time. The model also proved to be efficient compared to other video inpainting techniques which would allow higher resolution images to be generated in less time. Video outpainting research is still early in its development and this research shows that there are many ways to achieve the same goal.

Chapter 7

Future Work

The work shown in this paper could be built on in several ways. Higher quality results may be achieved by using a modified outpainting technique that uses a refinement process. Lin et al. (2021) uses a process of generating a coarse output which is refined by a generated edge map. Incorporating a similar structure for refining edges may produce highly detailed results but may make the frames temporally inconsistent. The training time and image resolution of the input could certainly be improved which could potentially give more valuable results. The discriminator could be modified from its original Patch-GAN design to give better feedback to the generator. This could possibly be fixed by increasing the size of the patch as some of the inconsistencies with training might come from the limited scope that the discriminator can see at one time. Changing the patch size will slow training as more information will have to be computed; however, when the resolution of the input image grows the discriminator could potentially degrade in performance and will fail to do its job correctly as the patch size will be too small for the image. The best extension of this research would be implementing a way for the generator to take advantage of the temporal

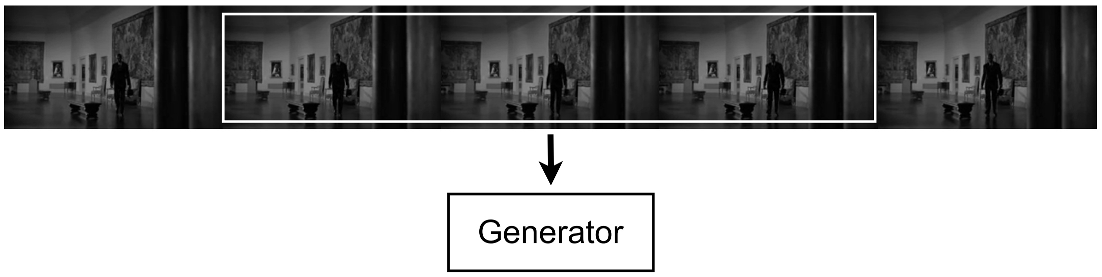

Figure 7.1: Sliding window

structure of video. In its current state, the generator produces frames without any knowledge of previous or future frames. This could be alleviated by implementing a sliding window for the generator to acknowledge multiple frames at a time. This would benefit the results in numerous ways as objects that move on and off screen would potentially be tracked, resulting in smoother results. Implementing a sliding window would also alleviate some warping that occurs frame-to-frame as the generated content would be more temporally consistent.

Another possible solution would be a recurrent neural network structure that feeds the previously generated results back into itself, thus becoming more content aware. This would hypothetically help with frame warping, but would fail to look ahead at future frames. Either change would most likely produce better results as it would become more time aware with the downside being longer computation times.

Bibliography

Yen-Chi Cheng, Chieh Hubert Lin, Hsin-Ying Lee, Jian Ren, Sergey Tulyakov, and Ming-Hsuan Yang. In&out: Diverse image outpainting via gan inversion. *arXiv:2104.00675 [cs]*, April 2021. URL `http://arxiv.org/abs/2104.00675`. arXiv: 2104.00675.

Andries P. Engelbrecht. *Computational Intelligence: An Introduction.* Wiley Publishing, 2nd edition, 2007. ISBN 0470035617.

Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Pearson, 2018. ISBN 9780133356724. URL `https://books.google.com/books?id=0F05vgAACAAJ`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. URL `http://www.deeplearningbook.org`.

Dewen Guo, Jie Feng, and Bingfeng Zhou. Structure-aware image expansion with global attention. In *SIGGRAPH Asia 2019 Technical Briefs*, SA '19, pages 13–16, New York, NY, USA, November 2019. Association for Computing Machinery. ISBN 978-1-4503-6945-9. doi: 10.1145/3355088.3365161. URL `http://doi.org/10.1145/3355088.3365161`.

Dongsheng Guo, Hongzhi Liu, Haoru Zhao, Yunhao Cheng, Qingwei Song, Zhaorui Gu, Haiyong Zheng, and Bing Zheng. Spiral generative network for image extrapolation. In *European Conference on Computer Vision*, pages 701–717. Springer, 2020.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. doi: 10.1109/CVPR.2017.632.

Kyunghun Kim, Yeohun Yun, Keon-Woo Kang, Kyeongbo Kong, Siyeong Lee, and Suk-Ju Kang. Painting outside as inside: Edge guided image outpainting via bidirectional rearrangement with progressive step learning. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages

2121–2129, Waikoloa, HI, USA, January 2021. IEEE. ISBN 978-1-66540-477-8. doi: 10.1109/WACV48630.2021.00217. URL `https://ieeexplore.ieee.org/document/9423167/`.

Dilip Krishnan, Piotr Teterwak, Aaron Sarna, Aaron Maschinot, Ce Liu, David Belanger, and William Freeman. Boundless: Generative adversarial networks for image extension. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10520–10529, Seoul, Korea (South), October 2019. IEEE. ISBN 978-1-72814-803-8. doi: 10.1109/ICCV.2019.01062. URL `https://ieeexplore.ieee.org/document/9008290/`.

Chuan Li and Michael Wand. Precomputed real-time texture synthesis with Markovian generative adversarial networks. *ArXiv*, abs/1604.04382, 2016.

Han Lin, Maurice Pagnucco, and Yang Song. Edge guided progressively generative image outpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 806–815, 2021. URL `https://openaccess.thecvf.com/content/CVPR2021W/NTIRE/html/Lin_Edge_Guided_Progressively_Generative_Image_Outpainting_CVPRW_2021_paper.html`.

Indra Deep Mastan and Shanmuganathan Raman. DeepCFL: Deep contextual features learning from a single image. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2896–2905, Waikoloa, HI, USA, January 2021. IEEE. ISBN 978-1-66540-477-8. doi: 10.1109/WACV48630.2021.00294. URL `https://ieeexplore.ieee.org/document/9423133/`.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

Yi Wang, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Wide-context semantic image extrapolation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1399–1408, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-72813-293-8. doi: 10.1109/CVPR.2019.00149. URL `https://ieeexplore.ieee.org/document/8953261/`.

Qingguo Xiao, Guangyao Li, and Qiaochuan Chen. Image outpainting: Hallucinating beyond the image. *IEEE Access*, 8:173576–173583, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.3024861.

Rui Xu, Xiaoxiao Li, Bolei Zhou, and Chen Change Loy. Deep flow-guided video inpainting. In *2019 IEEE/CVF Conference on Computer Vision and*

Pattern Recognition (CVPR), pages 3718–3727, 2019a. doi: 10.1109/CVPR.2019.00384.

Shuzhen Xu, Jin Wang, and Qing Zhu. Gradual image outpainting with pixel to pixel mapping. In *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, pages 1507–1510, October 2019b. doi: 10.1109/EITCE47263.2019.9094870.

Shuzhen Xu, Jin Wang, and Qing Zhu. Progressive image painting outside the box with edge domain. *Journal of Intelligent & Fuzzy Systems*, 39 (1):371–381, July 2020. ISSN 10641246. doi: 10.3233/JIFS-191310. URL `http://libraries.ou.edu/access.aspx?url=http://search.ebscohost.com/login.aspx?direct=true&db=aph&AN=144656370&site=ehost-live`. Publisher: IOS Press.

Zongxin Yang, Jian Dong, Ping Liu, Yi Yang, and Shuicheng Yan. Very long natural scenery image prediction by outpainting. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10560–10569, Seoul, Korea (South), October 2019. IEEE. ISBN 978-1-72814-803-8. doi: 10.1109/ICCV.2019.01066. URL `https://ieeexplore.ieee.org/document/9010032/`.

Zhenqiang Ying and Alan Bovik. 180-degree outpainting from a single image. *arXiv:2001.04568 [cs]*, January 2020. URL `http://arxiv.org/abs/2001.04568`. arXiv: 2001.04568.

Appendix A

Additional Frames

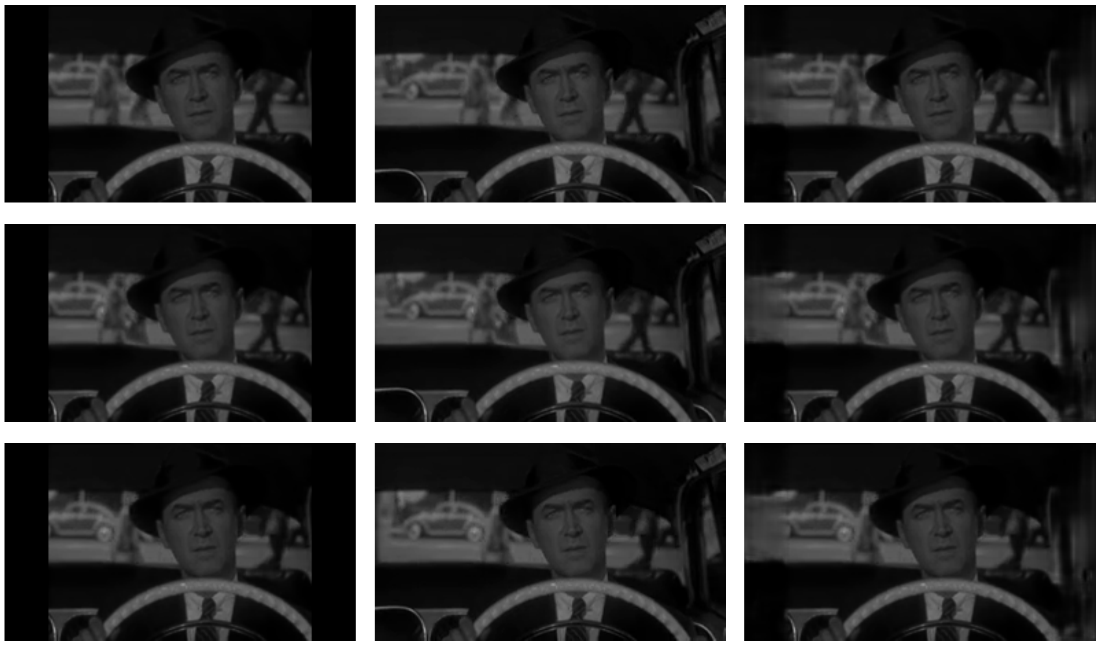

Figure A.1: Sequential frames showing temporal consistency

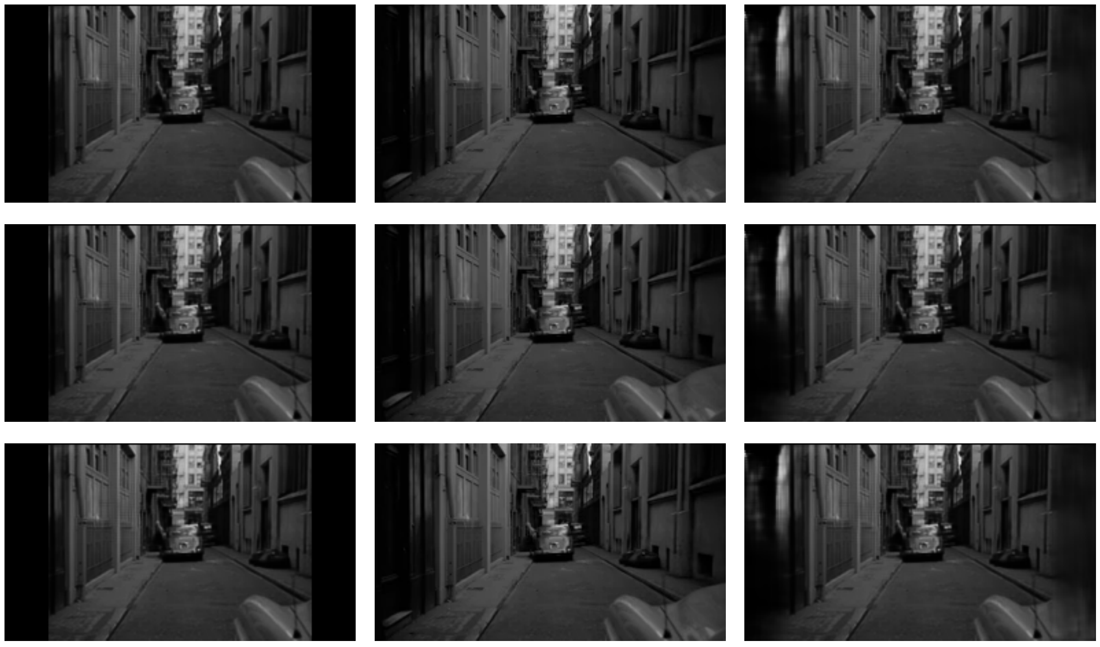

Figure A.2: Sequential frames showing temporal consistency

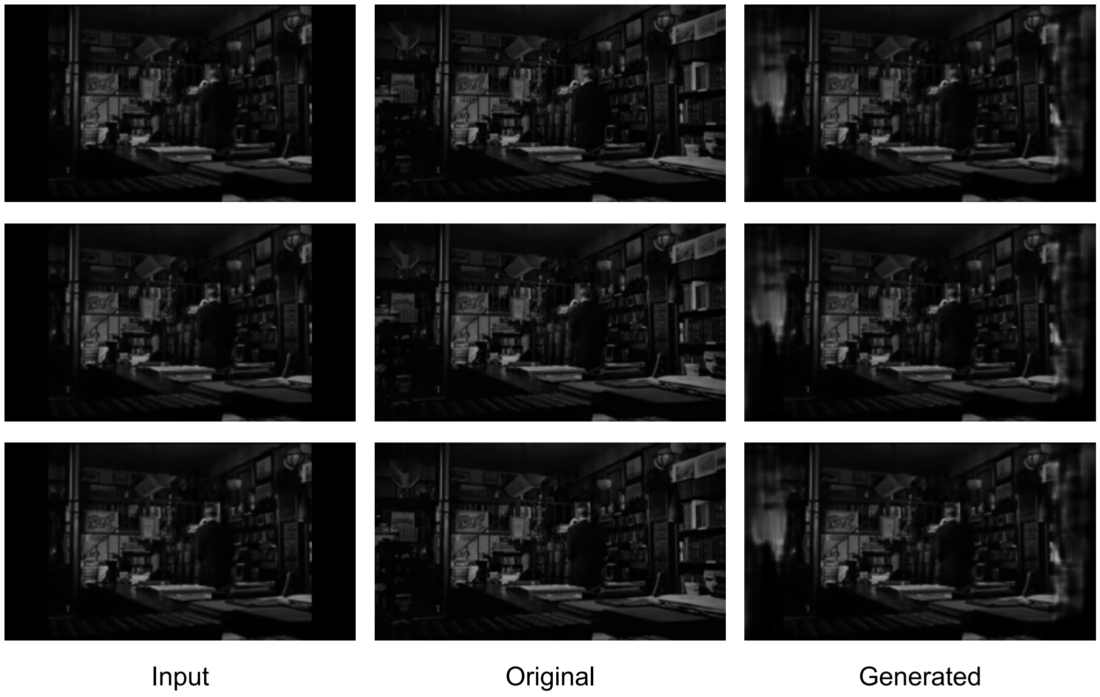

Figure A.3: Sequential frames showing temporal consistency

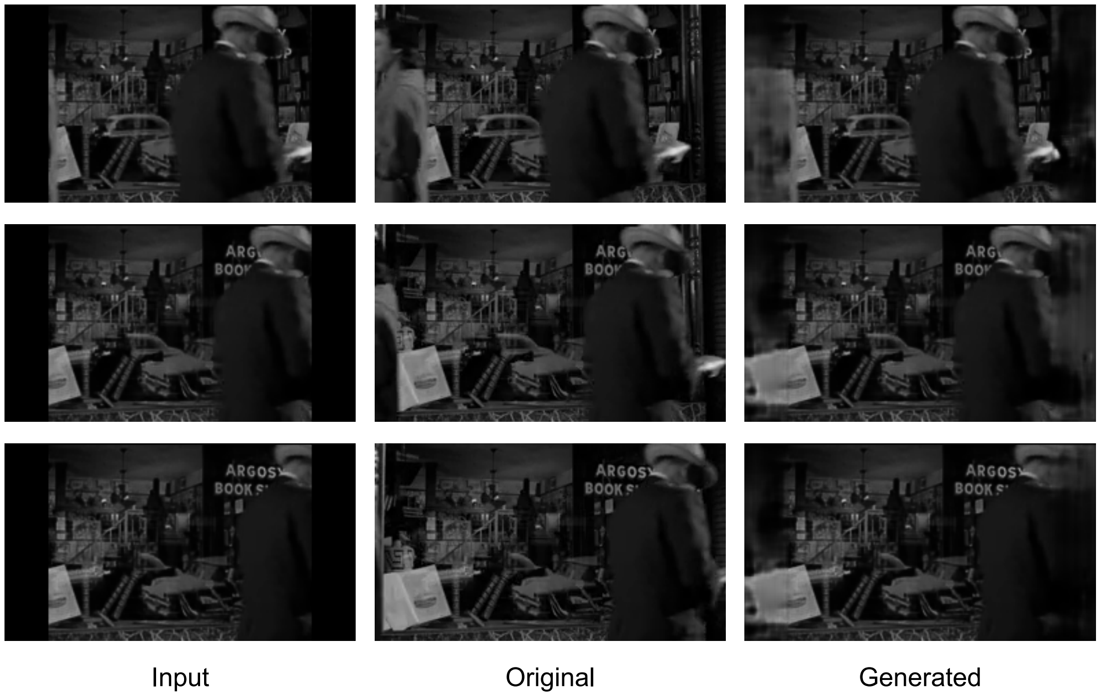

Figure A.4: Sequential frames of object moving into frame

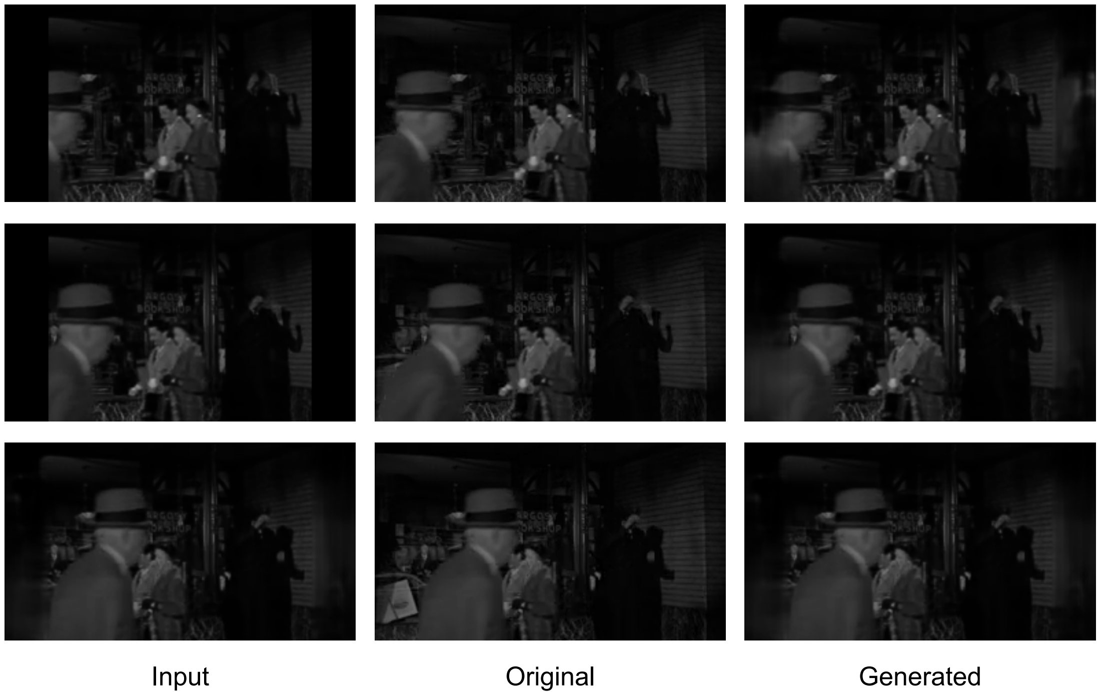

Figure A.5: Sequential frames of object moving out of frame

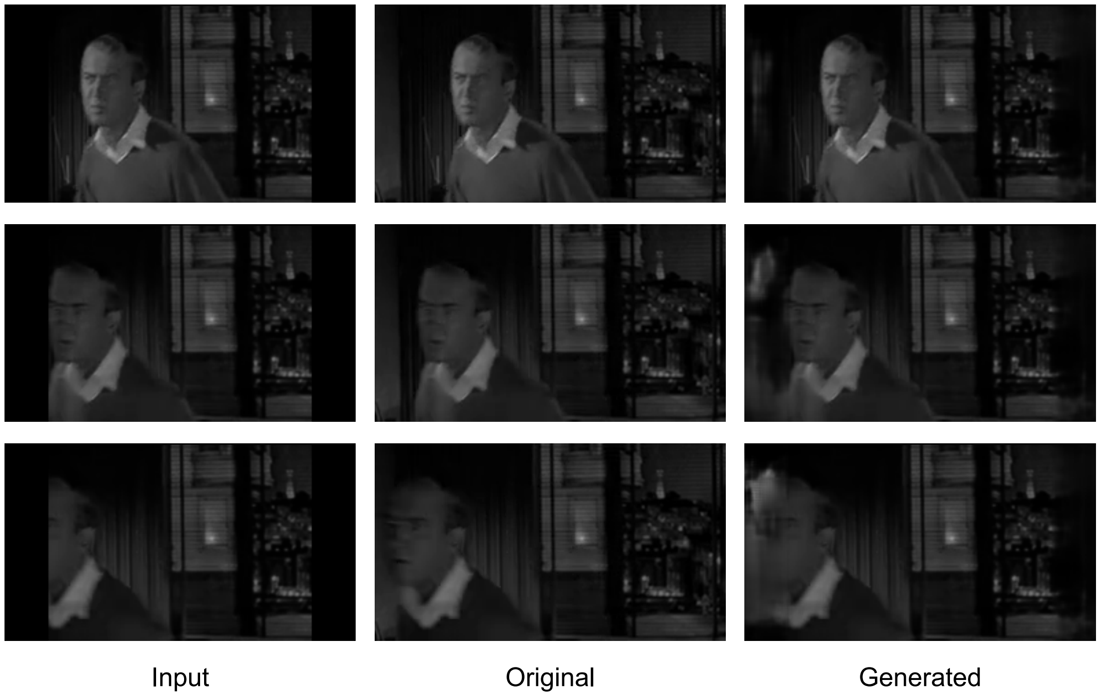

Figure A.6: Sequential frames of object moving out of frame

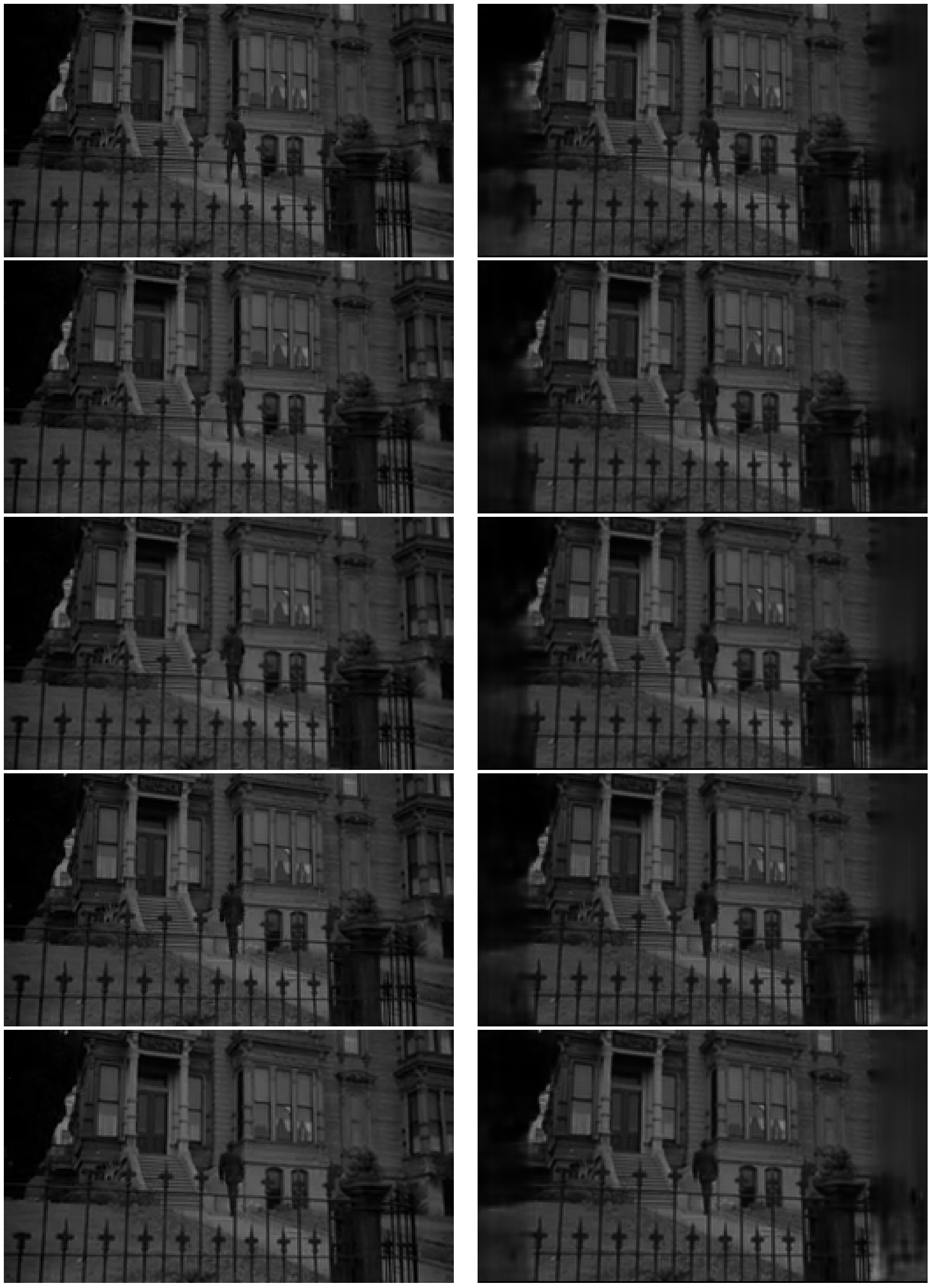

Figure A.7: Consecutive Frames from Panning Shot

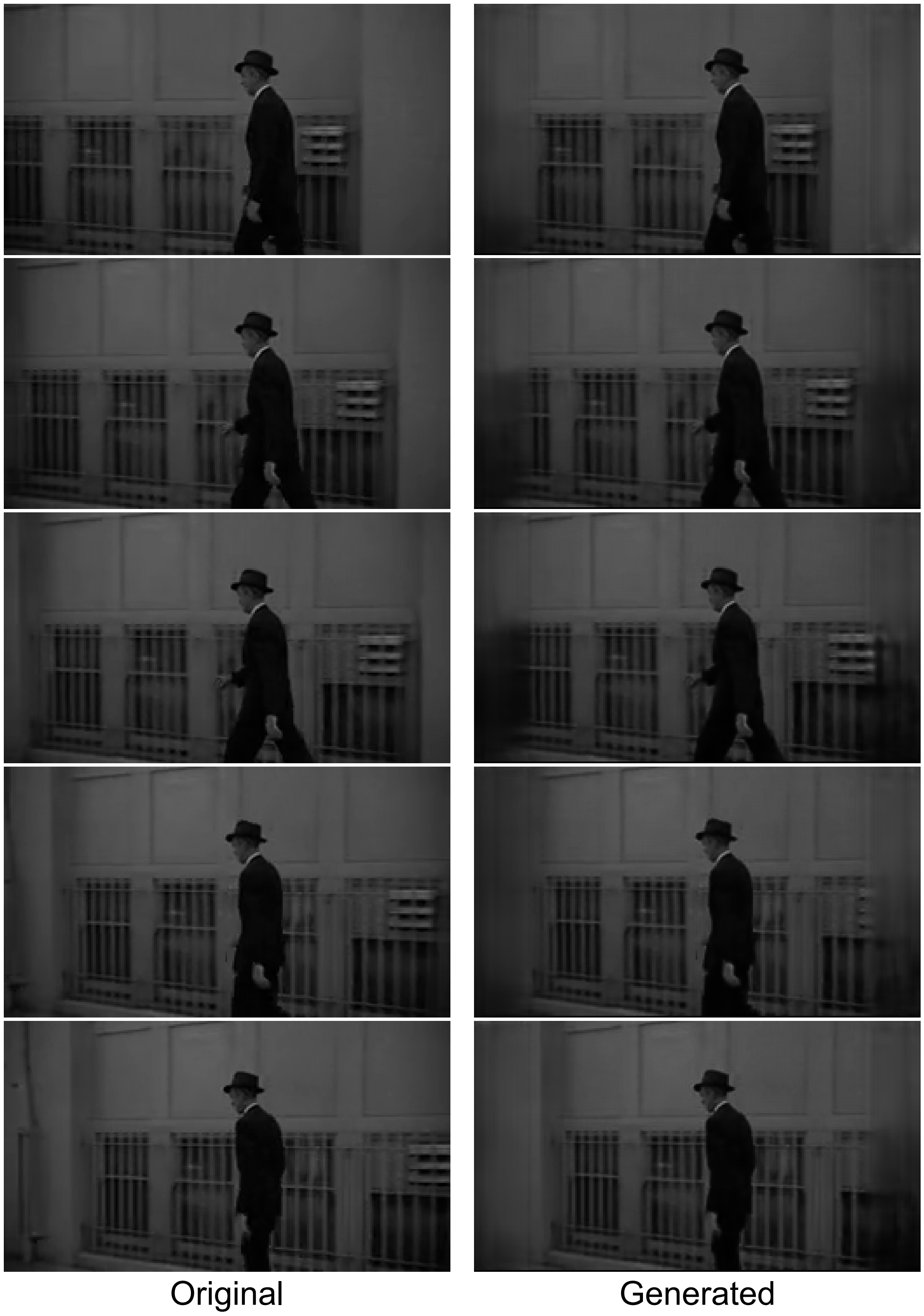

Figure A.8: Consecutive Frames from Panning Shot

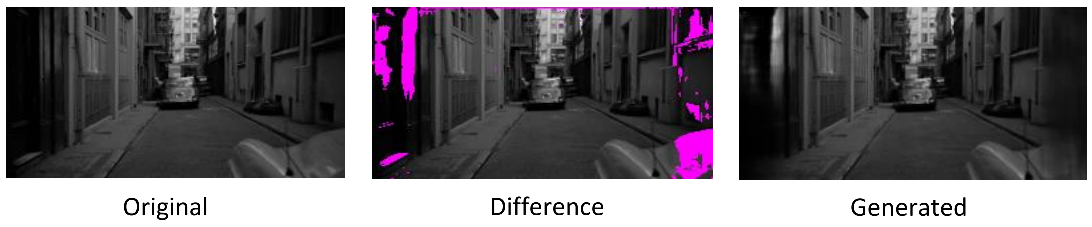

Figure A.9: Original and Generated Frames with Difference Highlighted