Applying Image Inpainting to Video Post Production

Bryce Haley bhaley@sfu.ca
Di Wang dwa92@sfu.ca
Hyukho Kwon hyukhok@sfu.ca
Joshil Patel joshilp@sfu.ca
Kino Roy kroy@sfu.ca

December 14, 2018

Abstract

This paper attempts to apply existing U-net Architecture presented in an earlier NVIDIA's paper[1] for image in-painting to videos without recruiting the use of a time sensitive models such as an LSTM. The network was trained on 512×512 images from the ImageNet set. Then tested on publicly available video in 720p. Strong results were achieved when infilling some noisy backgrounds such as ocean waves but displayed flickering and other artifacts when trying to reproduce more structured backgrounds.

1 Motivation

In the Visual Effects (VFX) industry, *inpainting* is a process where artists remove unwanted objects or artifacts in a film clip. Often, inpainting is done by creating a mask and stenciling each image frame, then painting in the resulting hole. Large sequences can be labour intensive and tedious, thus creating a bottlneck in the VFX pipeline, often slowing down the process for later departments. Using machine learning to automate the inpainting process would therefore, be fruitful in decreasing this bottleneck and could be useful for the VFX industry as a whole.

2 Approach

Image Inpainting for Irregular Holes Using Partial Convolutions (Liu et al. 2018) shows convincing results for predicting masked-out portions of an image [1]. This paper will attempt to extend the work by Liu et al. to apply their image inpainting process to a VFX application.

2.1 Partial Convolution

Convolutional neural networks have been the standard for image inpainting methods in the past, and have shown excellent performance at extracting high-level features of images. Liu et al. proposes a *partial* convolution network where only masked areas are passed as inputs. This approach appears to outperform previous methods for irregular masks.

Let $\mathbf{W}$ the weights for the convolution filter and b be the corresponding bias. $\mathbf{X}$ are the pixel values for the current convolution window and $\mathbf{M}$ is the corresponding binary mask.

$$x' = \begin{cases} \mathbf{W}^T(\mathbf{X} \odot \mathbf{M}), & \text{if sum}(\mathbf{M}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The mask is updated after each partial convolution. If the convolution was able to condition its output on at least one valid input value, then remove the mask for that location.

$$m' = \begin{cases} 1, & \text{if sum}(\mathbf{M}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The loss function is defined as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{valid}} + 6\mathcal{L}_{\text{hole}} + 0.05\mathcal{L}_{\text{perceptual}} + 120(\mathcal{L}_{\text{style}_{\text{out}}} + \mathcal{L}_{\text{style}_{\text{comp}}}) + 0.1\mathcal{L}_{\text{tv}} \tag{3}$$

where,

$$L_{\text{valid}} = ||M \odot (I_{\text{out}} - I_{\text{gt}})||_1 \text{ } (I_{\text{out}}\text{is the network prediction, } I_{\text{gt}} \text{ the ground truth})$$

$$L_{\text{hole}} = ||(1 - M) \odot (I_{\text{out}} - I_{\text{gt}})|| \text{ } (M \text{ is the initial binary mask})$$

$$L_{\text{perceptual}} = \sum_{n=0}^{N-1} ||\psi_n(I_{\text{out}}) - \psi_n(I_{\text{gt}})||_1 + \sum_{n=0}^{N-1} ||\psi_n(I_{\text{comp}}) - \psi_n(I_{\text{gt}})||_1$$

Auto-correlation (Gram Matrix) Style Loss:

$$L_{\text{style}_{\text{out}}} = \sum_{n=0}^{N-1} ||K_n((\psi_n(I_{\text{out}}))^T(\psi_n(I_{\text{out}}) - (\psi_n(I_{\text{gt}}))^T(\psi_n(I_{\text{gt}})))||_1$$

$$L_{\text{style}_{\text{comp}}} = \sum_{n=0}^{N-1} ||K_n((\psi_n(I_{\text{comp}}))^T(\psi_n(I_{\text{comp}}) - (\psi_n(I_{\text{gt}}))^T(\psi_n(I_{\text{gt}})))||_1$$

Smoothing Penalty:

$$L_{\text{tv}} = \sum_{(i,j) \in P, (i,j+1) \in P} ||I_{\text{comp}}^{i,j+1} - I_{\text{comp}}^{i,j}||_1 + \sum_{(i,j) \in P, (i+1,j) \in P} ||I_{\text{comp}}^{i+1,j} - I_{\text{comp}}^{i,j}||_1$$

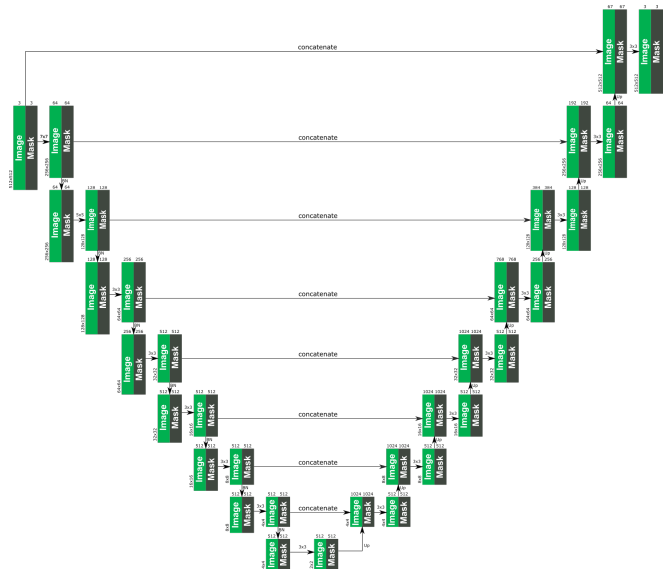

Figure 1: U-Net graph

2.2 U-Net Architecture

A U-shaped Network (U-Net) is used to learn image features and to predict pixels in the masked area. The left half of the U-Net is the shrinking stage. The shrinking stage uses down-pooling to aggregate information over large areas of the input images [6]. It is an essential step to learn key features (low-frequency) while putting less weight on details (high-frequency). However, down-pooling results in reduced resolution [6].

The right side of the U-Net is the refining stage. It firstly uses up-convolutional layers to extend features by replicating pixels. In order to recover the reduced resolution, the U-Net "concatenates" the layer with corresponding features from the left layer in each convolution [6]. This then allows the U-Net to predicts the output image which preserves both high-level features passed down from the shrinking state and fine local features provided from the left layers [6].

3 Data

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2017 dataset was used for training the model [3]. This was also one of the data sets Liu et al. used for training in their original implementation [1].

For testing data, seven short video clips were sourced online [3][4]. These clips were then rendered as a sequence of images to be used for the prediction model. Mask data was manually created per clip, and also rendered out as a sequence.

4 Code

A Python-based implementation of the NVIDIA paper that uses Keras and TensorFlow, created by Mathias Gruber, can be found on his GitHub repo [2].

Gruber implements the the following:

1. Creating random irregular masks
2. Implementing and testing the implementation of the PConv2D layer
3. Implementing and testing the U-Net architecture with PConv2D layers
4. Training and testing the final architecture on ImageNet

Our work in extending the code base involved modifying it to automate accepting image-sequence pairs representing a frame in a video, coupled with its corresponding mask. The masks we input were manually generated, as opposed to using Gruber's code to generate random masks, as this does not serve our purposes. We then output the frames in an organized way to easily be played as a video frame-by-frame.

As the model only accepts images of 512×512 pixels in size, Gruber implemented convenience functions in his model which, disect an input image into multiple images of the correct size, and then stitch multiple outputs of the model back together into the final prediction for that image. This allowed us to use high resolution videos.

5 Results

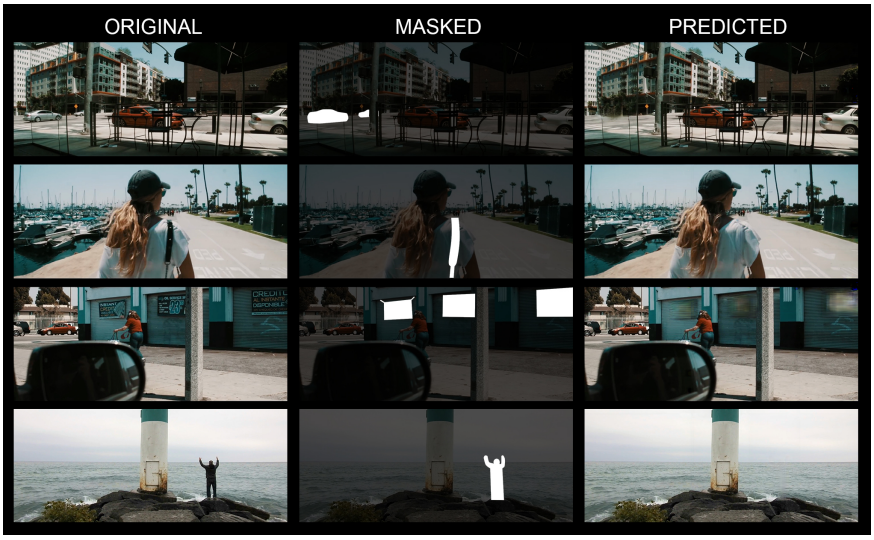

Figure 2: Frames from each of the four test videos comparing ground-truth, masked input, and model prediction.

Figure 2 displays the prediction results on four chosen sample videos. In order of the images from top to bottom, the objects items removed were two background vehicles in row 1, the bag strap in row 2, the three signs in row 3, and the standing person in row 4.

5.1 Successes

On a per-frame basis, the prediction model managed to recreate missing masked areas. Row 2 and 4 produced convincing predictions. On a sequential basis, the motion of the shot was also convincing for both rows due to the organic matter of the objects removed.

5.2 Failures

On some random single frames for Row 1, the predictions were convincing, however failed on many of the other frames. In Row 2, the predictions were unconvincing, as the prediction failed to re-produce a similar texture in neighbouring areas to the mask. Sequential predictions showed significant flickering artifacts.

6 Analysis of the Results

If the masked pixels have low-frequency details, the U-Net produces convincing predictions. Low-frequency means the changes in pixel values are relatively small, either with respect to x and y coordinates, or with respect to the timeframe t. Because the down-convolution pass captures the key features and eliminates noises, U-Net is not robust enough to recover high-frequency details even with the concatenation process.

Another visible artifact is flickering. The input only takes single tuple of (image, mask), which is why sequential predictions fail as there is no guarantee that the transition from one frame to the next is smooth.

7 Future Work

The drawback of a single tuple (image, mask) is that the U-Net is insufficient to train for smooth transitions among a sequence of frames. Therefore, the output footage shows flickering artifacts. To address this issue, it may be beneficial to feed U-Net a pair of tuples:

$$[(\text{image}_k, \text{mask}_k), (\text{image}_{k+1}, \text{mask}_{k+1})]$$

Each pixel is represented by $p(x, y, t) = I$, where I is the pixel intensity, (x, y) is the pixel coordinate on the image, and t is the timeframe. Track the motion of key-features between the two images by measuring horizontal velocity $\frac{dx}{dt}$, and vertical velocity $\frac{dy}{dt}$.

To maintain smooth transitions between frames, the change of velocity $u = \frac{d^2x}{dt^2}$ and $v = \frac{d^2y}{dt^2}$ should be small. The following loss function is borrowed from a smoothness constraint concept within Optic Flow [7]:

$$\mathcal{L}_{\text{smoothness}} = \sum_{\text{image}} [(\bigtriangledown v)^2 + (\bigtriangledown v)^2]$$

$\mathcal{L}_{\text{smoothness}}$ is then added to the old $\mathcal{L}_{\text{total}}$ loss function:

$$\begin{aligned}\mathcal{L}_{\text{total}} = w_0 \cdot \mathcal{L}_{\text{valid}} + w_1 \cdot \mathcal{L}_{\text{hole}} + w_2 \cdot \mathcal{L}_{\text{perceptual}} + w_3 \cdot \mathcal{L}_{\text{style}_{\text{out}}} \\ + w_4 \cdot \mathcal{L}_{\text{style}_{\text{comp}}} + w_4 \cdot \mathcal{L}_{\text{tv}} + w_5 \cdot \mathcal{L}_{\text{smoothness}}\end{aligned}$$

As of now w_i is unknown. Finding the right combination of weights of different loss functions will be future work.

8 Team Member Contributions

All members contributed evenly to this project, which includes research, implementation, analysis, reporting, and poster presentation.

References

[1] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, Bryan Catanzaro, (2018) Image Inpainting for Irregular Holes Using Partial Convolutions, *ECCV*

[2] MathiasGruber, (2018) PConv-Keras, *GitHub Repository*, https://github.com/MathiasGruber/PConv-Keras

[3] Deng, J. and Dong, W. and Socher, R. and Li, L.-J. and Li, K. and Fei-Fei, L., (2017) ImageNet: A Large-Scale Hierarchical Image Database, *CVPR09*, http://image-net.org/challenges/LSVRC/

[4] ID 3444753, (2016) Untitled *PixaBay*, https://pixabay.com/en/videos/waves-energy-exercise-zen-6183/

[5] Matteo Bertoli Visuals, (2016) Back to BMPCC! (Random footage) - Blackmagic Pocket Cinema Camera, https://www.youtube.com/watch?v=7qB5gsUboHY

[6] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazırbas, Vladimir Golko, Patrick van der Smagt, Daniel Cremers, Thomas Brox, (2015) FlowNet: Learning Optical Flow with Convolutional Networks

[7] Berthold K.P. Horn, Brian G. Schunck, (1980) Determining Optical Flow