Computer Vision Project Summer 2022 (Report)

Students: Mustafa Waked 318577921 and Layan Haddad 318369709

Paper: "Deep Rectangling for Image Stitching: A Learning Baseline" (March 2022)

Paper purpose: the known image rectangling methods mainly work on searching an initial mesh and optimizing a target mesh to form the mesh deformation in two stages. Then rectangular images can be generated by warping stitched images. HOWEVER, these methods only work for images with rich linear structures. Which means that portrait and landscape images would most likely suffer from distortions if these methods were applied on them. THIS PAPER CAME TO PROVIDE A SOLUTION FOR THIS ISSUE.

Paper modifications: We were provided with 2 modifications to choose from, the first one being "Editing the training set by implementing standard geometric augmentations on the data set" and the second one was "Seam Carving" which is an algorithm for content-aware image resizing, where we had to implement the algorithm using the paper by modifying some part of the code, we decided to go with the first option.

So, why there was a need to modify the training data in the first place?

In the article, particularly on page 5 (step 4), we can see that the researchers have generated 60,000 samples, and out of this big number of samples, only 5705 were chosen for the training process, and thus eliminating approximately 95% of the samples. And so by implementing standard geometric augmentations on those data the chances of eliminating further samples shall be reduced.

Our algorithm for modifying the data:

Our approach for a possible geometric augmentation:

Crop and scale: we take a certain part of the image, perform cropping on it, and then resize this part to be the same size as the original image.

Implementation: We cropped each image (‘gt’ and ‘input’) from the training data, preserving the aspect ratio of the original image. To be able to crop both the stitched image and the ‘gt’ accordingly with precision we used both their meshes (that were created for each image during the initial

training). We extracted the vertices from the mesh of the stitched image that we wanted to crop, and we chose the same vertices accordingly in the 'gt' image mesh as well. That way we made sure to crop the exact same portion from both images.

After cropping both images we scaled them back to the original image size, and then we created the mask for the new stitched image. Applying that to all the images in the training set, we now have created a new training folder that contained a new 'gt', new 'input' and a new 'mask' image to train our model with. Examples of those new input images are shown below.

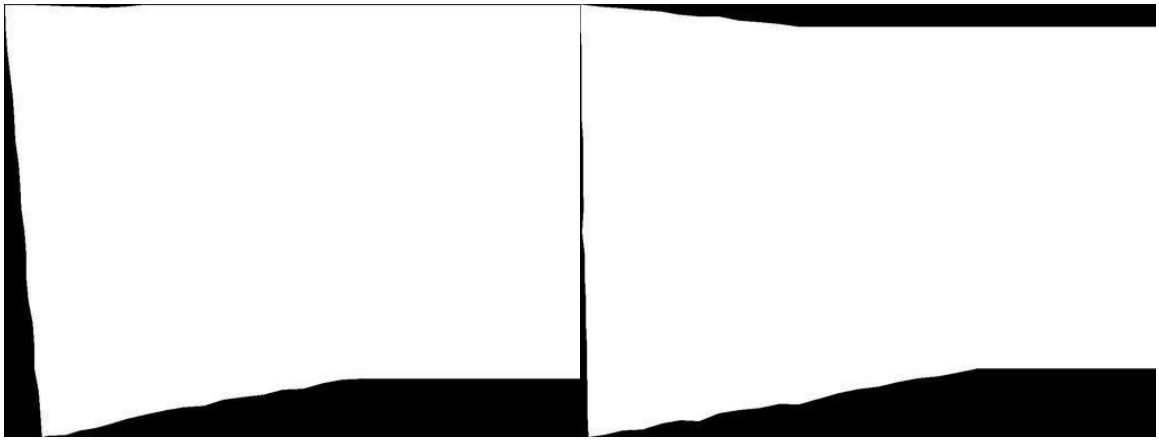
The full implementation of this geometric augmentation can be found in inference2.py, whereas if one runs the file, they shall have the new data set generated AUTOMATICALLY. PLEASE consider changing the paths of the files to match your coding environment. After running the file, we saved the results in final_res.py.

Original training data:

'gt':

'input':

‘mask’:

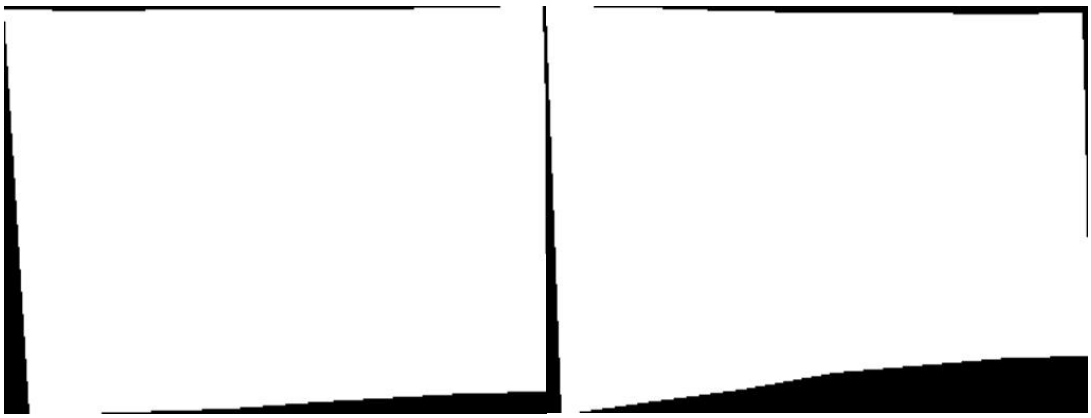


Our new training data:

‘gt’:

‘input’:

‘mask’:



The inference2.py path file: DeepRectangling/Codes/inference.py2

The final_rest.py path file: DeepRectangling/Codes/Data/DIR-D/final_res

Training with the new generated data:

Training can be performed exactly as described in the paper readme file. With few differences: since none of us have GPU on our laptops, we decided to use google COLAB that offers a perfectly free training environment with a GPU. Again, please be aware of the files path before starting the training.

To make sure the model trains smoothly, please make sure to add the following code cells to your colab notebook:

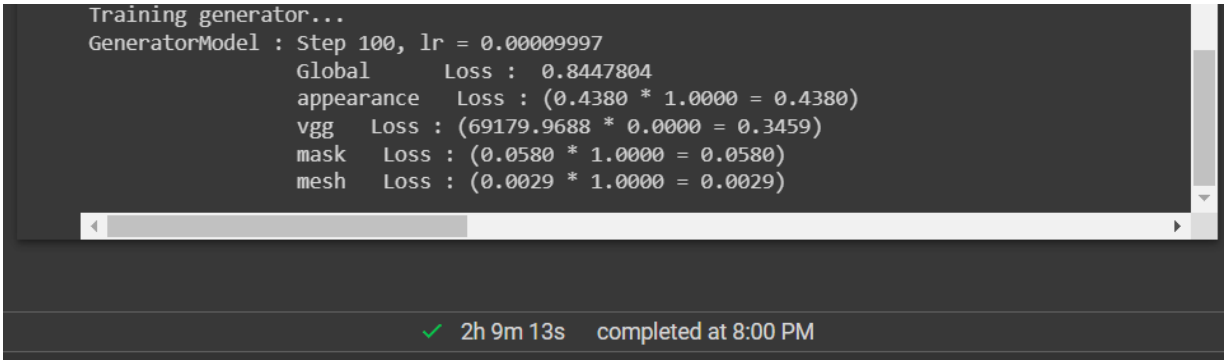
```
cd drive/MyDrive/DeepRectangling/Codes    #the path to get to the file that contains train.py

!pip install tensorflow==1.13.1  #installing tensorflow 1.13.1 (as requested by the original paper)

import tensorflow as tf                #checking that you have the right version of tensorflow installed
print(tf.__version__)

!pip uninstall keras        #the appropriate keras version for tensorflow 1.13.1 is 2.2.3 !pip
install keras==2.2.3
```

!pip install numpy==1.18.1 #installing the right numpy version

import numpy #checking numpy version print(numpy.__version__)

!python train.py #train the model

Training results (for 100 iterations, sadly we couldn't do more because our google colab GPU usage has run out, but you get the idea 😉):



Testing the model trained with the new data:

After training, please follow the instructions on the readme file of the paper to preform the testing. (Be sure to change the files paths as well).

RESULTS

The testing results are supposed to be better than the original paper results, since by generating new training data with these augmentations, the mesh of each image is ought to get more suitable for the content of the image, and thus we get better rectangling results (psnr wise). Further more, by using these augmentations on further samples, the amount of the eliminated samples is supposed to be less than 95%.

Best,

Mustafa and Layan, University of Haifa, Sept 2022.