**Part D: Computer Vision** 2 marks. Look up Content aware fill in Photoshop (or Resynthesize and heal in GIMP), discuss a computer vision algorithm that might help do what the tool does. If you want the ‘right’ (incomprehensible) answer, the algorithm was developed by P. Harrison in his 2005 PhD Dissertation for the University of Monash (<http://www.logarithmic.net/pfh/thesis>), but I’d rather you try and think about it on your own. 500-1000 words.

(RNNs are complicated, too complicated for this course apparently so I hope this is fine being this brief.)

A computer vision algorithm that might help with a content aware fill in a system like photo-shop or gimp could be the use of an RNN in order to fill in the hole where the image was erased.

The process would start with another item being removed from the scene using a mix of edge detection and noise reduction, this leaves us with a smooth slightly textured image that we can use as our canvas.

From there the RNN comes in. A recurrent neural network is (surprisingly) a type of neural network, this means that it can be trained on arbitrary data to predict novel outcomes. The interesting part of a recurrent neural network would be its ability to store behavior inside the network itself, and work with this self-built storage to actually replicate the data that it has built its network around.

Now you may think, this is crazy how would I load in an entire image, you load in the pixels! 1920 x 1080 pixels is only 2,073,600 pixels! Actually that’s a lot, especially when you consider that each pixel is going to be an input node and needs a subsequent output node as well. Luckily we’re working with selected smaller images using snipping tools and such in Photoshop so the image should be a much easier to handle amount of pixels.

The pixels come with 4 channels, each would need to be its vector again however, so we’ll first convert our image in black and white before pushing it into our machine. Then we create a network of x input nodes immediately pooled into a layer of radu nodes to both scale the value to a value between 0 and 1, as well as assign it to either a 0 or 1 for one hot encoding. Basically this step just makes the network learn much faster.

This is where our (very briefly described) network begins to work. As a neural network learns the weights of the values on each node changes based on various factors, this value for the node can be considered, what the machine currently thinks about that thought, if you trace the network you will find that each node is a result of what the machine thought of a combination of pixels. Meaning this is the algorithm thinking about a picture, making connections, and ingraining them into its structure.

We then need to generate our fill, so to do this we collect sample images from the area surrounding the area erased, this will give our network the knowledge of that patterns that it is trying to replicate after we train our machine on this data. We then leverage the benefit of an RNN and for lack of a better word, make it a circle, this now means that the image that is put into the RNN, is also the image that is specified by the output nodes. At this point the fancy part of the RNN starts and as the network runs we take our samples of the output layer of nodes to use as replacement images.

These images are then placed on the deleted area and using edge detection the best one is selected.

Image -> Normalize -> RNN -> Learn pattern -> Loop RNN -> Collect output vectors -> Select best image