5.

6.


shutterstock
www.shutterstock.com · 73249894


www.shutterstock.com · 553662235

For this example it performed poorly because there is not really a texture to be captured. In order for it to look normal it would need to fill that space with probably another vegetable, but with our algorithm it will take from the edges of the holes and build using similar points. That would not work for an example like this because instead you just get a jumbled mess as it is matching different coloured vegetables into the hole.

7. patchL determines how big the patch is. It represents the length of the patch from the centre of the patch. So, if patchL was 3, the overall patch would be 7x7, because the patch would stretch 3 points going each ways from the centre, plus one for the centre point. If the patchL is too small, it will have trouble capturing the true texture and patterns of the image. The patch needs to be large enough to be able to get enough context of the image to replicate the texture properly. But there is a trade-off as if it is too big then it will just start copying the patterns over and it will look odd as the eyes will notice the identical patterns.

randomPatchSD is used to select the matching patch that will be used. It uses a random gaussian distribution based off the value of randomPatchSD to determine which one to select. This means that 95% of the time they will pick values within 2*SD of the best fit. So, if the value is very small it will more likely pick the optimal patch every time because 2*SD will be close to optimal choice. This may make the image look too similar. If it is too big then the 2*SD could range quite far away from optimal choice. This makes it more likely to pick patches that are too

random which will produce a poor result as it will pick some patches that are too different to what is needed.