# Holefill.py

**from** **PIL** **import** Image, ImageDraw

**import** **numpy** **as** **np**

**import** **random**

**import** **os.path**

**import** **pickle**

*# Functions for you to complete*

**def** ComputeSSD(TODOPatch, TODOMask, textureIm, patchL):

patch\_rows, patch\_cols, patch\_bands = np.shape(TODOPatch)

tex\_rows, tex\_cols, tex\_bands = np.shape(textureIm)

ssd\_rows = tex\_rows - 2 \* patchL

ssd\_cols = tex\_cols - 2 \* patchL

SSD = np.zeros((ssd\_rows,ssd\_cols))

*# Tuples of points where patch is not empty*

p = np.where(TODOMask == 0)

zippedP = zip(p[0], p[1])

*# Must change the data unit to be float as the arguments*

*# are given in unit8 which are not suitable for calculations*

textureIm = textureIm.astype('float')

TODOPatch = textureIm.astype('float')

*# for every way of sliding the patch over the image,*

*# it computes the sdd of that match on the given points*

*# where the patch is non empty*

*# Compute sum square difference between textureIm and TODOPatch*

*# for all pixels where TODOMask = 0, and store the result in SSD*

**for** r **in** range(ssd\_rows):

**for** c **in** range(ssd\_cols):

*# For the non-empty points*

**for** (x, y) **in** zippedP:

*# Sum over RGB (three channels)*

**for** d **in** range(patch\_bands):

SSD[r][c] += (textureIm[r][c][d] - TODOPatch[x][y][d])\*\*2

**return** SSD

**def** CopyPatch(imHole,TODOMask,textureIm,iPatchCenter,jPatchCenter, iMatchCenter,jMatchCenter,patchL):

patchSize = 2 \* patchL + 1

*# Empty points coordinates*

p = np.where(TODOMask == 1)

zippedP = zip(p[0], p[1])

**for** i **in** range(patchSize):

**for** j **in** range(patchSize):

*# Copy the selected patch selectPatch into the image containing*

*# the hole imHole for each pixel where TODOMask = 1.*

*# The patch is centred on iPatchCenter, jPatchCenter in the image imHole*

**if** (i, j) **in** zippedP:

*#Hole coordinates*

iH = i+iPatchCenter - patchSize/2

jH = j+jPatchCenter - patchSize/2

*#Image coordinates*

iI = i+iMatchCenter - patchSize/2

jI = j+jMatchCenter - patchSize/2

*#Copy the RGB channels from texture to the hole*

**for** d **in** range(3):

imHole[iH][jH][d] = textureIm[iI][jI][d]

**return** imHole

#

# Helper functions in the python file is the same as was given (not included here)

#

*# Main script starts here*

*# Constants*

*# Change patchL to change the patch size used (patch size is 2 \*patchL + 1)*

patchL = 10

patchSize = 2\*patchL+1

*# Standard deviation for random patch selection*

randomPatchSD = 1

#

# Rest of the python file is the same as given (not included here)

#

Q6) The algorithm performed poorly due to the fact that the background is not really uniformly distributed like the donkey one for example (with the grass). My picture had a weird, asymmetrical texture in the background of the hole I was trying to replace.

Q7) patchL : This is used to offset from the hole to find a suitable texture to replace with. The bigger the patchL the bigger the sample size; which may effect calculations (if it’s too close to the edge of the image, or is larger than the image itself, it will throw an error). A larger offset may reduce accuracy of the patches taken from sample textures resulting in unrealistic ones.

randompatchSD: is used to calculate a gaussian value with 0 as the mean, however it is compared to the size of the Hole, and the minimum is taken between them. So a large value would not effect the calculations, but a small value would mean the texture finding algorithm would take a longer time to go through the pixels of the picture to find a good match.