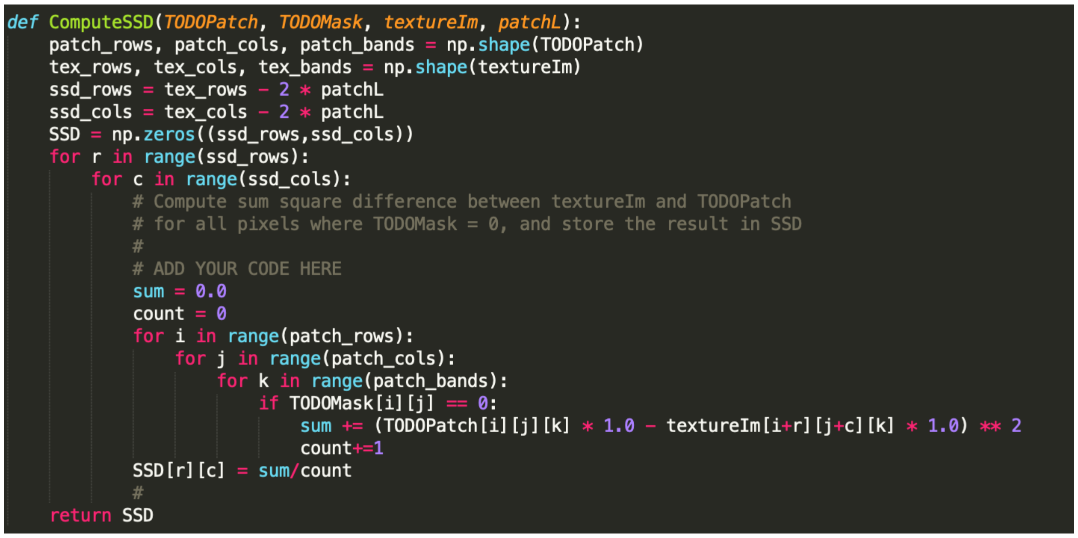
CPSC425-A3

Eloise Peng - 13812169

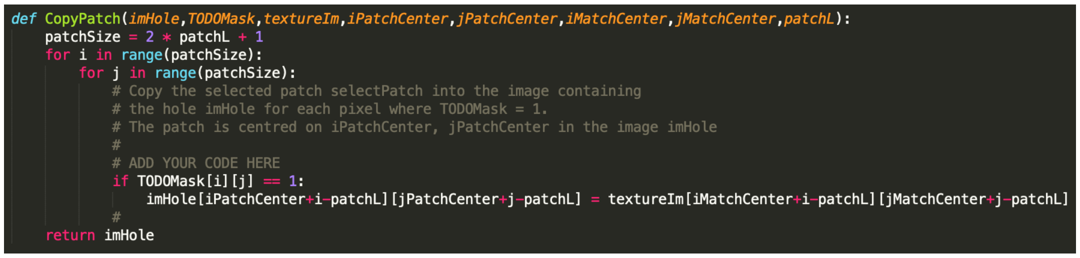
October 2019

1 Q4



```python
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))
    for r in range(ssd_rows):
        for c in range(ssd_cols):
            # Compute sum square difference between textureIm and TODOPatch
            # for all pixels where TODOMask = 0, and store the result in SSD
            #
            # ADD YOUR CODE HERE
            sum = 0.0
            count = 0
            for i in range(patch_rows):
                for j in range(patch_cols):
                    for k in range(patch_bands):
                        if TODOMask[i][j] == 0:
                            sum += (TODOPatch[i][j][k] * 1.0 - textureIm[i+r][j+c][k] * 1.0) ** 2
                            count+=1
            SSD[r][c] = sum/count
            #
    return SSD
```

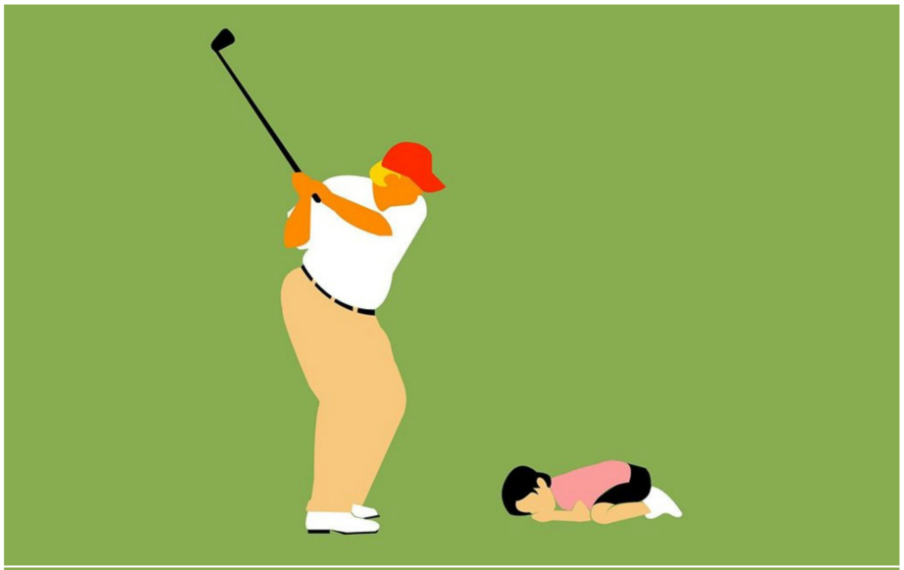
2 Q5

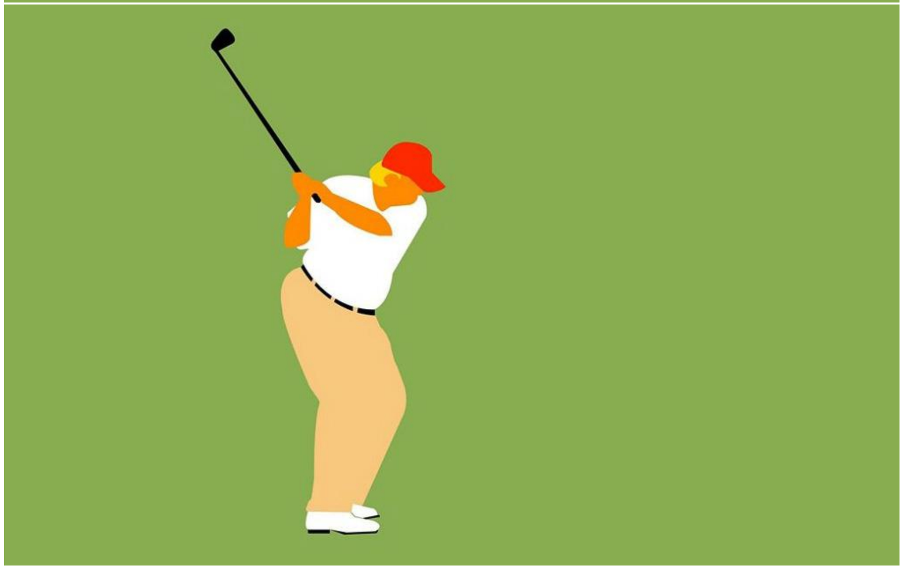


```python
def CopyPatch(imHole,TODOMask,textureIm,iPatchCenter,jPatchCenter,iMatchCenter,jMatchCenter,patchL):
    patchSize = 2 * patchL + 1
    for i in range(patchSize):
        for j in range(patchSize):
            # Copy the selected patch selectPatch into the image containing
            # the hole imHole for each pixel where TODOMask = 1.
            # The patch is centred on iPatchCenter, jPatchCenter in the image imHole
            #
            # ADD YOUR CODE HERE
            if TODOMask[i][j] == 1:
                imHole[iPatchCenter+i-patchL][jPatchCenter+j-patchL] = textureIm[iMatchCenter+i-patchL][jMatchCenter+j-patchL]
            #
    return imHole
```

3 Q6

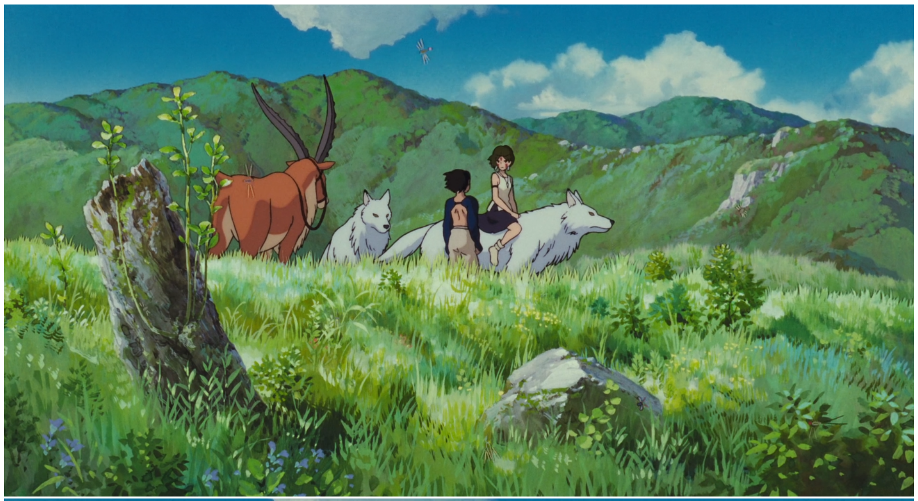
Good Sample:

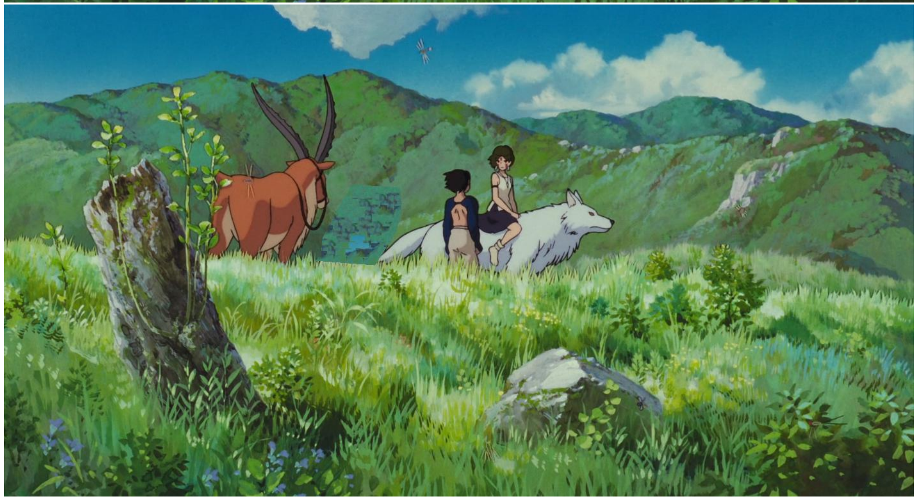




The synthesis work very well here as the texture of the background are very smooth and predictable (only one color with same lightness).

NOT Satisfied sample:



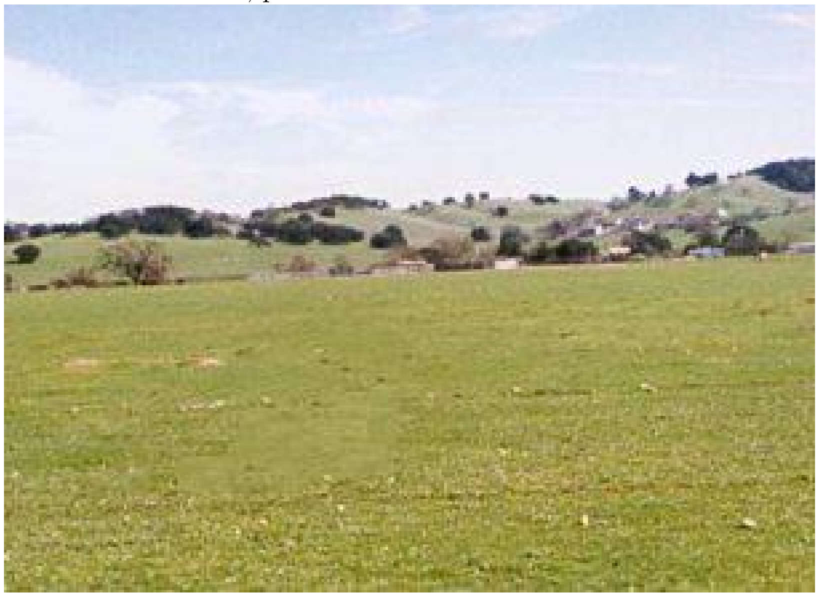


This synthesis is not too optimal as the texture of background are very complicated that the image contains a lot of details on the lines and color tones. The synthesis would work batter with a smoother and more systematic patterned surfaces. However, as the image background is a formation of different levels of green, the matching patch can fool human eyes at some point similar to Camouflage. So the result picture will look better from distance.

4 Q7

If randomPatchSD is too small, we only choose the couple best matches out of the patches, this might lead to over replicated patches and it might also be redundantly copying some noises.

If randomPatchSD is too large, it will choose the patches randomly, which will highly diffuse the chance to pick up the best match out of it. The outcome will might random artifacts that are not suppose to be there at the first place.

randomPatchSD = 1, patchL = 10


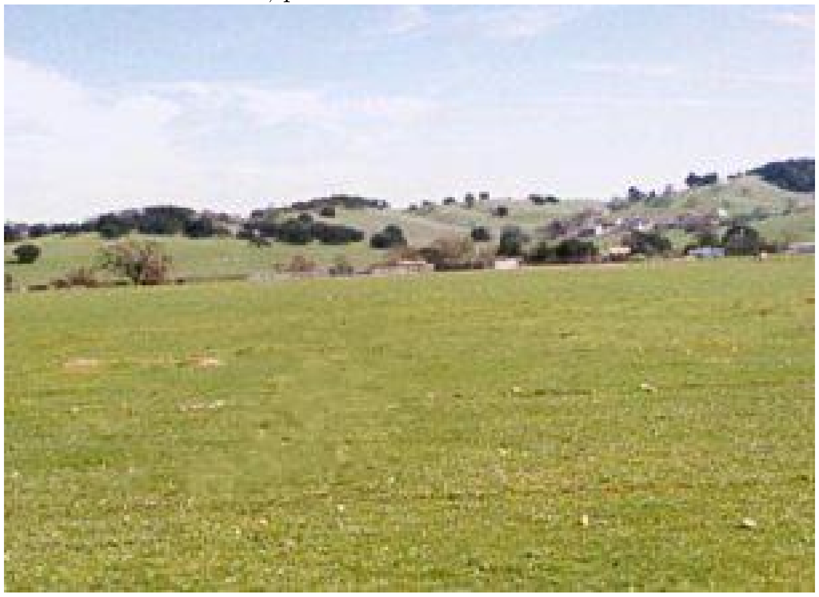
randomPatchSD = 5, patchL = 10

randomPatchSD = 10, patchL = 10

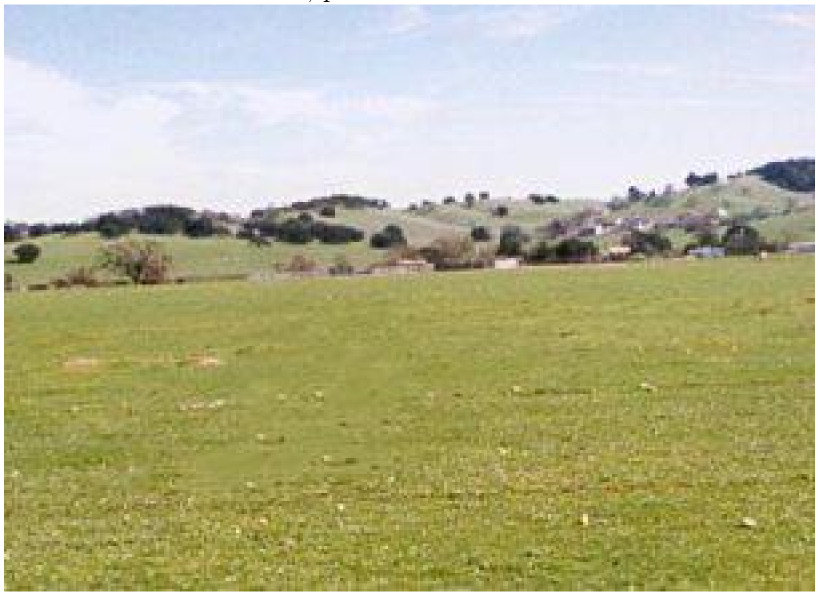
If patchL is too small, we might only capture segment of the actual pattern which might lead to a unnecessary repetitions (i.e. moire). It also taking a longer time to find the best match since there are more patches to choose from.

If patchL is too large, it might leads to uneven/rough artifacts as window is bigger which might include irrelevant patches.

randomPatchSD = 2.2, patchL = 1



randomPatchSD = 2.2, patchL = 10



randomPatchSD = 2.2, patchL = 20