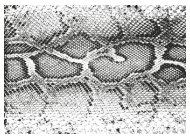
Image Processing: Homework 4

Due on January 11, 2020 at 6:00pm

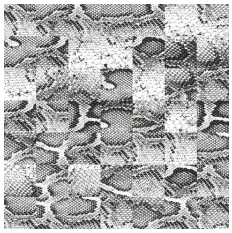
Yamin Sarcheshmehpour
94109827

Problem 1

Solution: Genrating texture with random patches.
Texture is:



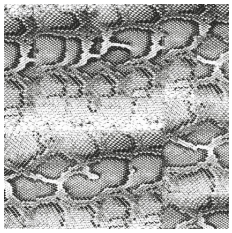
And the result is:



The code is in $Q1.py$, for runing the code please run *python* $Q1.py$.

Problem 2

Solution: Genrating texture with SSD overlap.



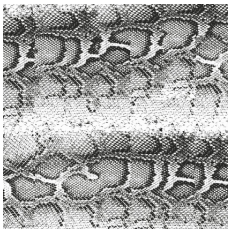
Describe functions

1. *template_matching_ssd*: It returns the overlap error of the selected patch.
2. *get_best_match*: It findes the best match for the (i, j)th block of the generated texture by choosing the best match for this place.
3. *generate_texture*: It generates the desired texture.
4. $fade\_up$: It calculates the mean of overlap for top of the selected patch.
5. $fade\_left$: It calculates the mean of overlap for left of the selected patch.

The code is in $Q2.py$, for runing the code please run *python* $Q2.py$.

Problem 3

Solution: Genrating texture with min cut.



Describe functions
The only difference between this code with the previous one is the *fading* part.

1. get_mask: This function calculates the min cut, at first it calculates weights for left, currect, right edges and then for each node it will find the min edge and will finde the best path.
2. fade_up: It calculates the mean cut of overlap for top of the selected patch using *get_mask* function.
3. fade_left: It calculates the mean cut of overlap for left of the selected patch using *get_mask* function.

The code is in *Q3.py*, for runing the code please run *python Q3.py*.

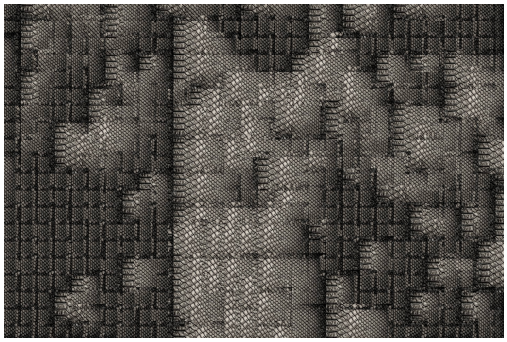
Problem 4

Solution: At this problem I have implemented texture transfer.
Texture is:

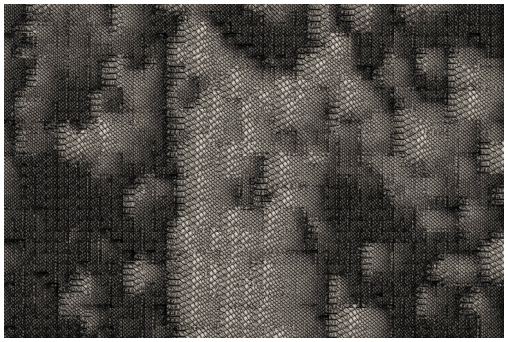
The target is:

And the result is:

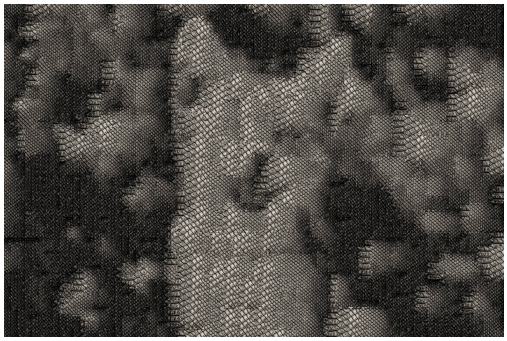
I have done this in 5 iterations, at each iteration I have fund best match patch from texture for each block which is near the target corresponding patch and has min overlap error and created the result, then I reduced the *patch* and *overlap/match* sizes. You can see the result of each iteration at follows.



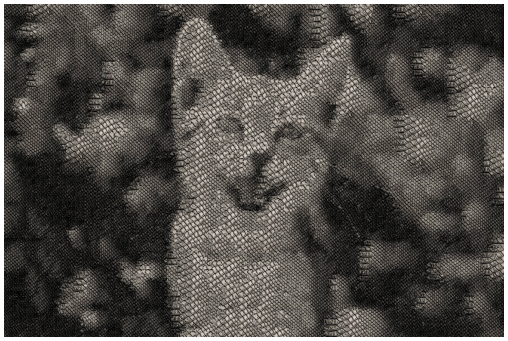
(a) iter1



(b) iter2



(c) iter3



(d) iter4

(e) iter5

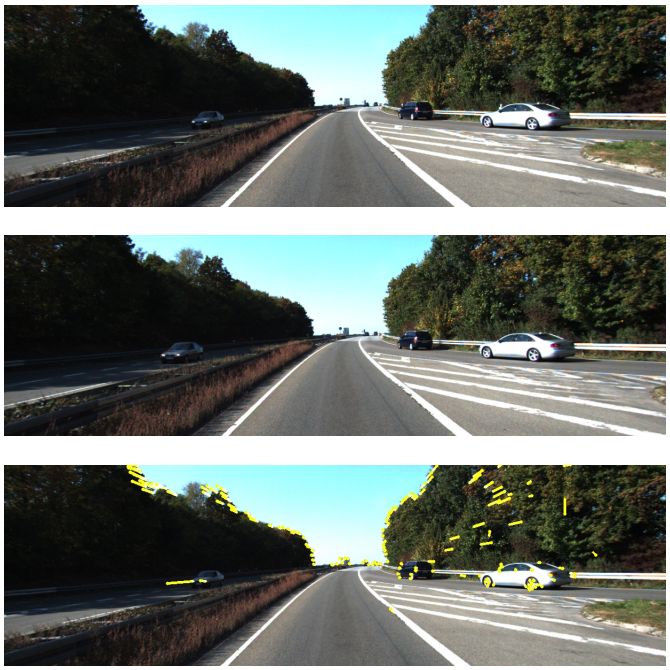
Describe functions The only difference between this code with the previous one is the *get_best_match*.

1. *get_best_match*: It findes the best match for the (i, j)th block of the generated image by choosing the best match for this place by calculating error of its overlaps and the previous patch and the original target.

The code is in *Q4.py*, for runing the code please run *python Q4.py*.

Problem 5

Solution: The Optical Flow between the following images is:



For doing this I have used $cv2.goodFeaturesToTrack$ to find the good points in the first image and then I used $cv2.calcOpticalFlowPyrLK$ to find that points in the second image.

The code is in $Q5.py$, for runing the code please run $python\ Q5.py$.