TEAK Guillaume 40157390 CONCORDIA UNIVERSITY 14/02/2020

Computer Vision Report

Feature detection :

- I've implemented the Feature Detection in the python file : "Feature_Detection", I tried to seperate the code into functions to make it easier to read.

- gradient_x() and gradient_y() are 2 functions that I use to apply Sobels filters on an image that I pass in parameter.

- Find_Local_Peaks(matrix) is a function I created to find the coordinates of the maximum peak in a matrix, each time we encounter a new peak, its coordinates are replaced in the array_with_the_max_peak_coordinates

- Most of the feature is then implemented in the function Find_Corners() which returns actually, in the same format of the image passed in parameter, the C_Responses we were asked to find, those responses are floats.

- This function basically do the Harris algorithm, for the non-maximum suppresion, I've chosen to keep the last value in case of many same peaks values.

For the treshold, instead of affecting it a random value, I gave him like a percentage of the maximum value in my array of c_responses :

- tresh = 0.05 * c_response.max() so it's kinda more effective.

- For the display of corners, in a first time after finding the corners, I paint in blue the one pixel which represents the interest point in the copy of the image array .

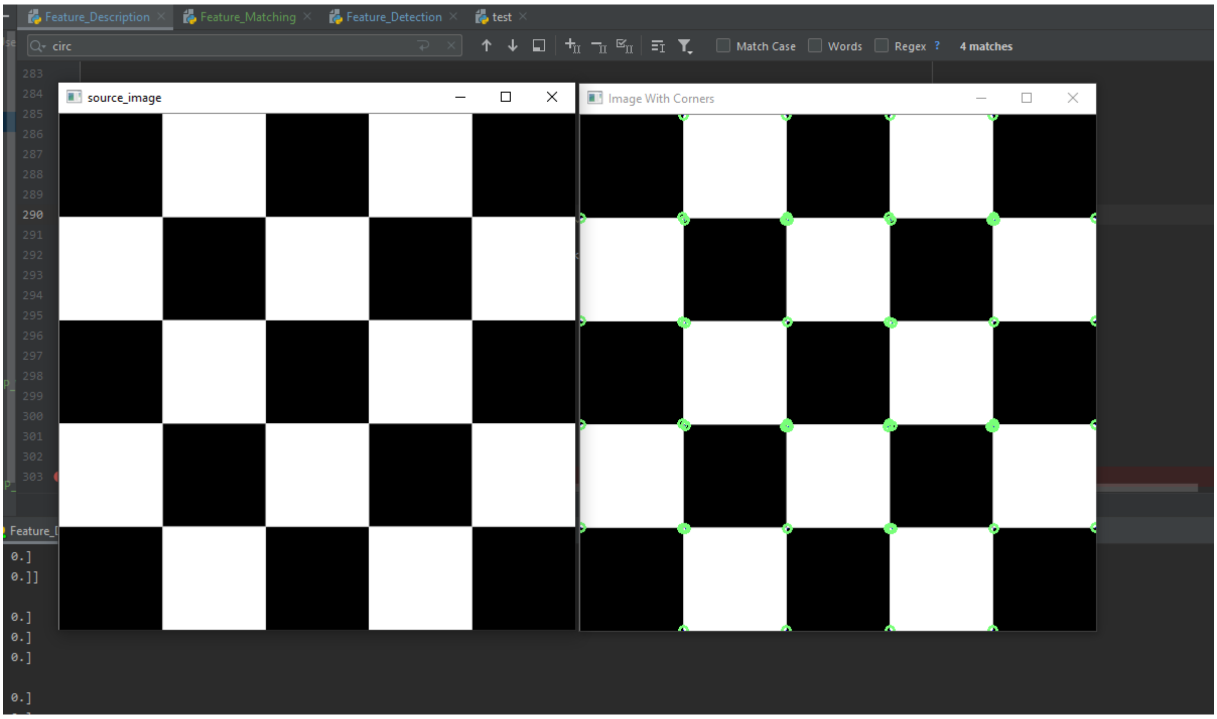
- But at the end I added cv2.DrawCircle to stress out those corners.

Overall : - The feature is implemented and the corners are clearly visibles, I can modify the threshold at will in order to show less or more corners.

RESULTS :

```
With a tresh of tresh = 0.05 * c_response.max()
```

With a chessboard image



```
0.]
0.]]

0.]
0.]
0.]

0.]
```

With Yosemite1 / Yosemite2 :

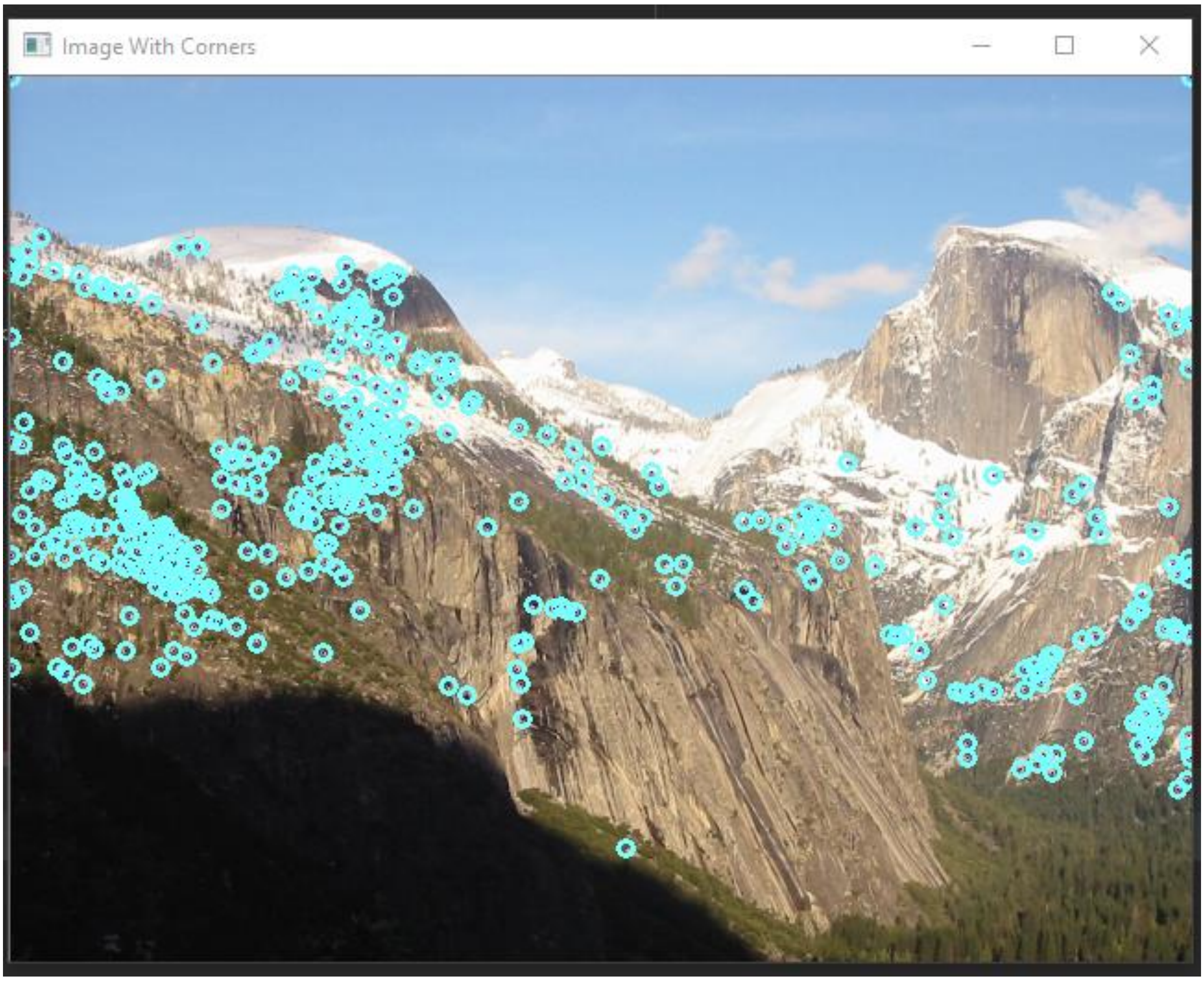
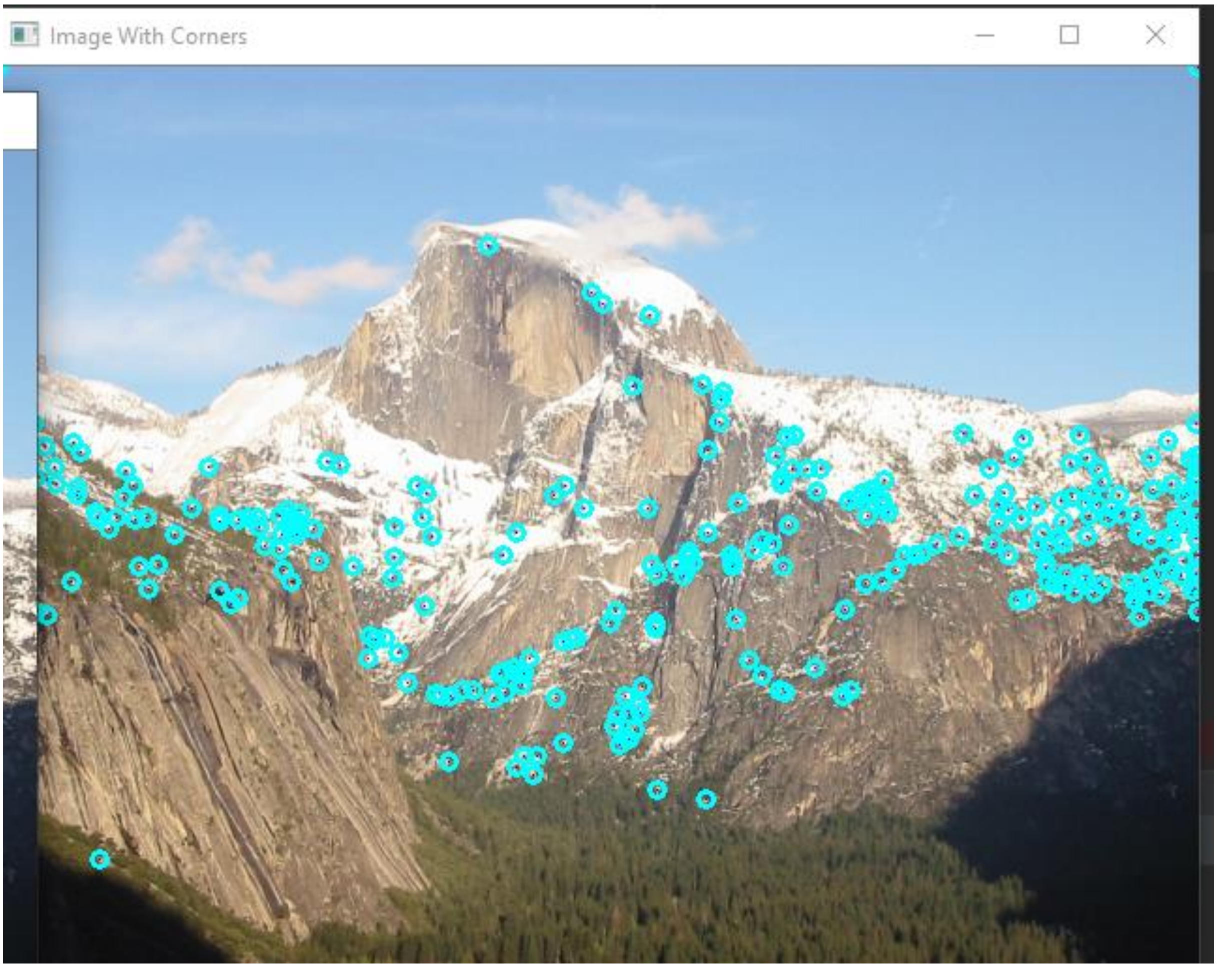
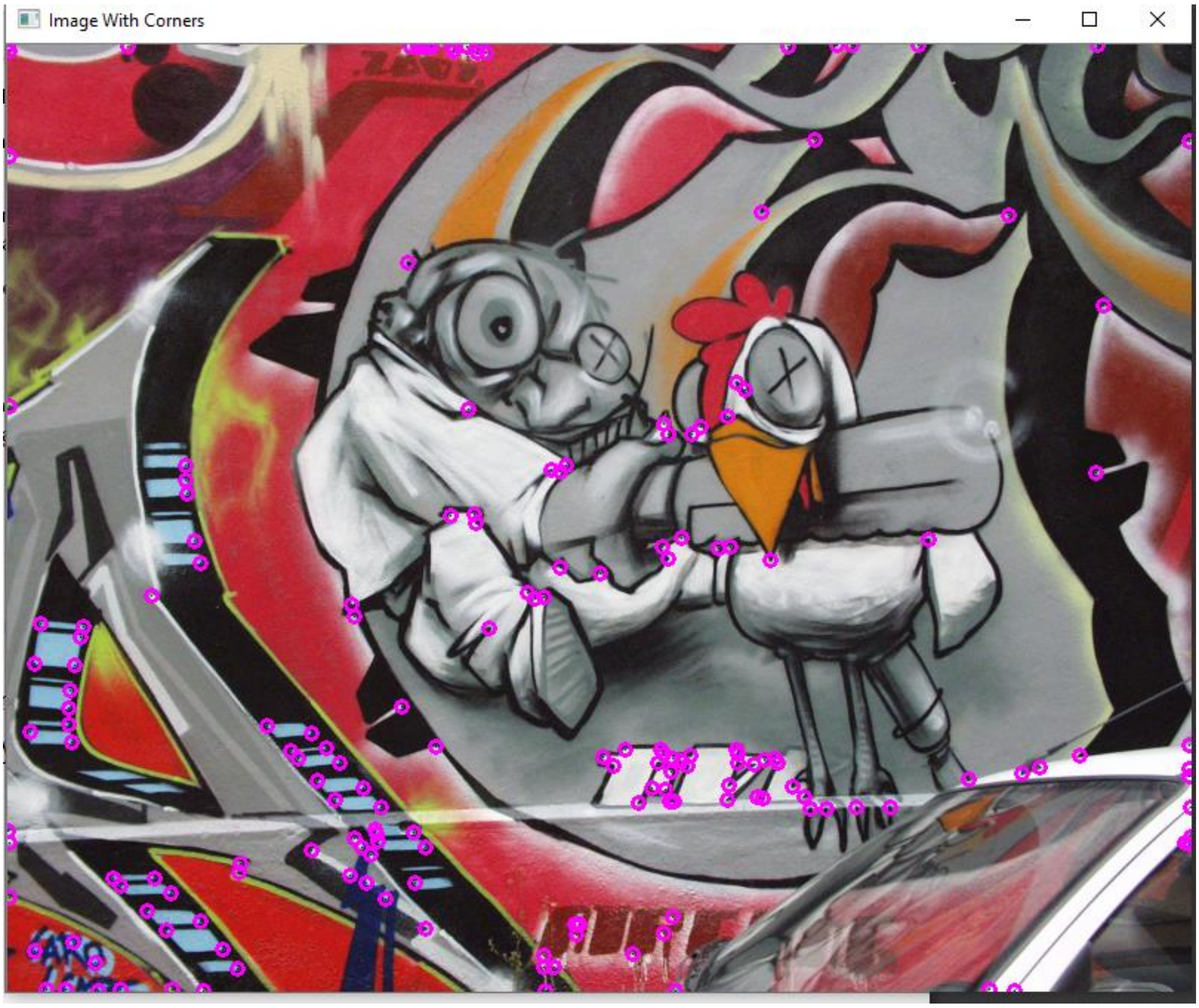


Image With Corners



With image1.ppm :



Feature Description :

- I copied and pasted what was in Feature Detection and continue the implementation in the python file "Feature_Description",

- The structure is the same, tried to implements functions for visibility.

- After getting the c_responses in the previous feature, I placed them in the main() within matrix_with_C_reponses

- Most of the feature is implemented into the function Assign_Orientation()

Small functions explanations first :

- Descriptor_Normalized(hist, treshold=0.2) is a function I implemented to normalize the descriptor(histogram), it goes through the hist vector, then set to 2.0 any value that exceed it.

- blockshaped() is a sympathic function I found on the net to slice an array in uniform parts, I used that algo to slice the 16x16 window into 16 4x4 cells.

- normalize(v) to normalize a vector, used later to normalize the 128 values we got from adding the histogramms.

- DrawCircle to make it more easier to see the corners.

Assign_Orientation() :

- We have to work on a grey image, I did some condition on size matching between

- I used a lot of for loops, and also a offset to be able to slide the window .

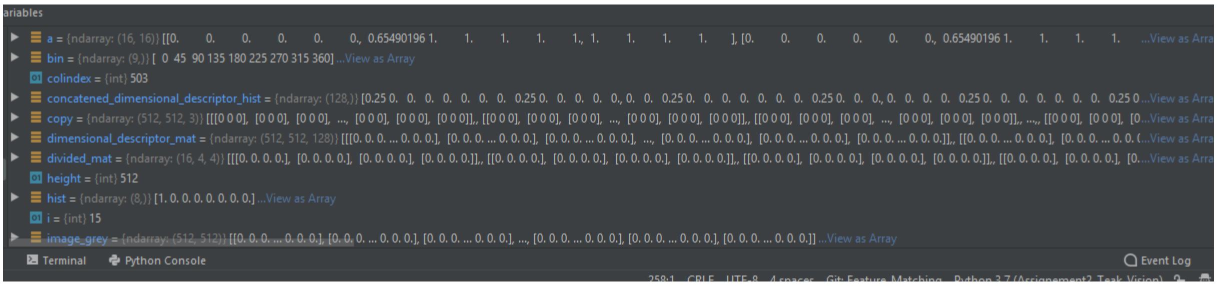
- I calculated the angles that I place in there theta_mat, had to use math libary, and convert Radians into degrees with the formula

- By following the Assignement rules, I created again a array that keeps the size of the inputed image. It has the descriptors values everywhere there is an interest point .

- We add every concatened 128 values vector into their respective interest point in the dimensional_descriptor_mat.

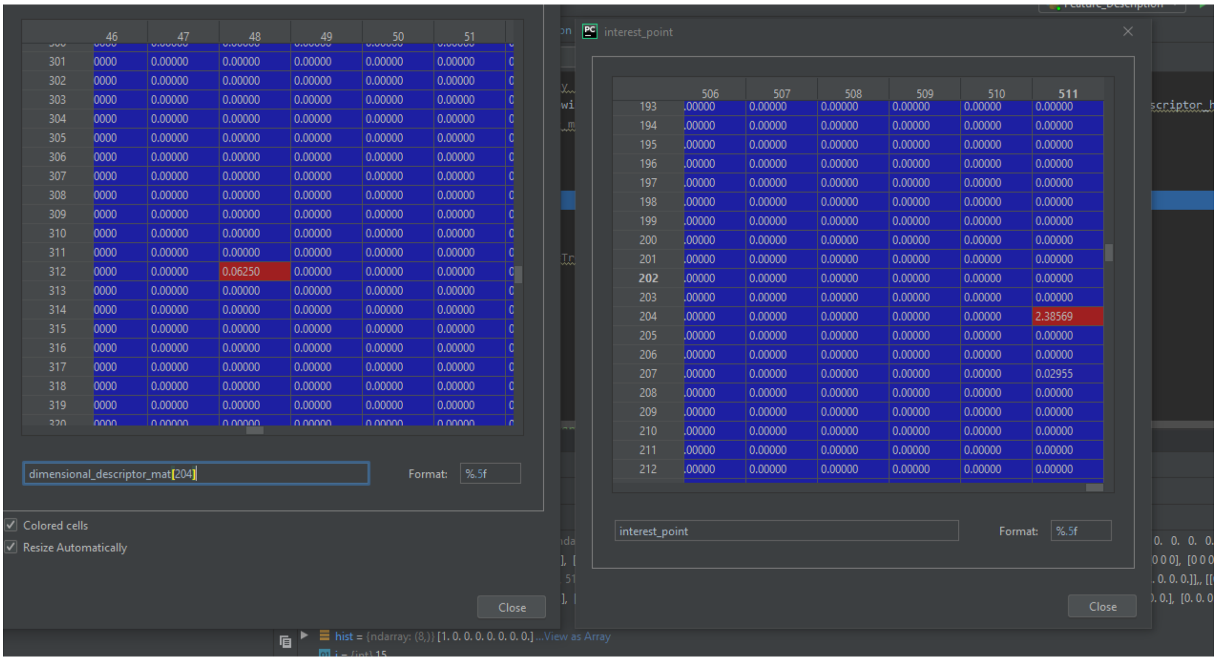
Overall : - The feature is done, I obtain a matrix (3 dimensional) the size of the image, in this matrix, at the place of an interest point, we have the correspondant 128 values that allows us to compare features in different images next.

We are gonna observe the array with descriptor ;



```
a = {ndarray: (16, 16)} [[0.       0.       0.       0.       0.       0.,  0.65490196 1.       1.       1.       1.       1.,  1.       1.       1.       1.       ], [0.       0.       0.       0.       0.       0.,  0.65490196 1.       1.       1.       1.       ...View as Arra
bin = {ndarray: (9,)} [  0  45  90 135 180 225 270 315 360] ...View as Array
colindex = {int} 503
concatened_dimensional_descriptor_hist = {ndarray: (128,)} [0.25 0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.   0.,  0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.,  0.   0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.25 0...View as Arra
copy = {ndarray: (512, 512, 3)} [[[0 0 0],  [0 0 0],  [0 0 0],  ...,  [0 0 0],  [0 0 0],  [0 0 0]],, [[0 0 0],  [0 0 0],  [0 0 0],  ...,  [0 0 0],  [0 0 0],  [0 0 0]],, [[0 0 0],  [0 0 0],  [0 0 0],  ...,  [0 0 0],  [0 0 0],  [0 0 0]],, ...,, [[0 0 0],  [0 0 0],  [0...View as Arra
dimensional_descriptor_mat = {ndarray: (512, 512, 128)} [[[0. 0. 0. ... 0. 0. 0.],  [0. 0. 0. ... 0. 0. 0.],  [0. 0. 0. ... 0. 0. 0.],  ...,  [0. 0. 0. ... 0. 0. 0.],  [0. 0. 0. ... 0. 0. 0.],  [0. 0. 0. ... 0. 0. 0.]],, [[0. 0. 0. ... 0. 0. 0.],  [0. 0. 0. ... 0. 0. (...View as Arra
divided_mat = {ndarray: (16, 4, 4)} [[[0. 0. 0. 0.],  [0. 0. 0. 0.],  [0. 0. 0. 0.],  [0. 0. 0. 0.]],, [[0. 0. 0. 0.],  [0. 0. 0. 0.],  [0. 0. 0. 0.],  [0. 0. 0. 0.]],, [[0. 0. 0. 0.],  [0. 0. 0. 0.],  [0. 0. 0. 0.],  [0. 0. 0. 0.]],, [[0. 0. 0. 0.],  [0. 0. 0. 0.],  [0...View as Arra
height = {int} 512
hist = {ndarray: (8,)} [1. 0. 0. 0. 0. 0. 0. 0.] ...View as Array
i = {int} 15
image_grey = {ndarray: (512, 512)} [[0. 0. 0. ... 0. 0. 0.], [0. 0. 0. ... 0. 0. 0.], [0. 0. 0. ... 0. 0. 0.], ..., [0. 0. 0. ... 0. 0. 0.], [0. 0. 0. ... 0. 0. 0.], [0. 0. 0. ... 0. 0. 0.]] ...View as Array
```

Terminal
Python Console
Event Log
258:1 CRLF UTF-8 4 spaces Git: Feature Matching Python 3.7 (Assignement2 Teak Vision)



	46	47	48	49	50	51
301	0000	0.00000	0.00000	0.00000	0.00000	0.00000
302	0000	0.00000	0.00000	0.00000	0.00000	0.00000
303	0000	0.00000	0.00000	0.00000	0.00000	0.00000
304	0000	0.00000	0.00000	0.00000	0.00000	0.00000
305	0000	0.00000	0.00000	0.00000	0.00000	0.00000
306	0000	0.00000	0.00000	0.00000	0.00000	0.00000
307	0000	0.00000	0.00000	0.00000	0.00000	0.00000
308	0000	0.00000	0.00000	0.00000	0.00000	0.00000
309	0000	0.00000	0.00000	0.00000	0.00000	0.00000
310	0000	0.00000	0.00000	0.00000	0.00000	0.00000
311	0000	0.00000	0.00000	0.00000	0.00000	0.00000
312	0000	0.00000	0.06250	0.00000	0.00000	0.00000
313	0000	0.00000	0.00000	0.00000	0.00000	0.00000
314	0000	0.00000	0.00000	0.00000	0.00000	0.00000
315	0000	0.00000	0.00000	0.00000	0.00000	0.00000
316	0000	0.00000	0.00000	0.00000	0.00000	0.00000
317	0000	0.00000	0.00000	0.00000	0.00000	0.00000
318	0000	0.00000	0.00000	0.00000	0.00000	0.00000
319	0000	0.00000	0.00000	0.00000	0.00000	0.00000

dimensional_descriptor_mat[204]   Format: %.5f

☑ Colored cells

☑ Resize Automatically

hist = {ndarray: (8,)} [1. 0. 0. 0. 0. 0. 0. 0.] ...View as Array

interest_point

	506	507	508	509	510	511
193	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
194	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
195	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
196	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
197	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
198	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
199	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
200	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
201	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
202	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
203	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
204	.00000	0.00000	0.00000	0.00000	0.00000	2.38569
205	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
206	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
207	.00000	0.00000	0.00000	0.00000	0.00000	0.02955
208	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
209	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
210	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
211	.00000	0.00000	0.00000	0.00000	0.00000	0.00000
212	.00000	0.00000	0.00000	0.00000	0.00000	0.00000

interest_point   Format: %.5f

Feature Matching :

- Same as before, we re-use our codes and ameliorate it into the file "Feature_Matching", we have now our descriptor matrix.

- To calculate the SSD, I use a function I created SS_DIstance that takes 2 feature matrix as parameters, I've created 4 vectors

- descriptors1 = []

  descriptors2 = []

  descriptors1_coord = []

  descriptors2_coord = []

The 2 first allows me to extract only the vectors which are filled with values, descriptor_coord allow me to know where in the big matrix those vectors are.

- I go through the 2 matrix and look for the Minimum SSD, between 2 patches, when the minimum is found, I can get back its coordinates in the big matrix thanks to descriptor_coord.

OVERALL : - The feature can give you the coordiantes of the two best match between patches between the two images you input.