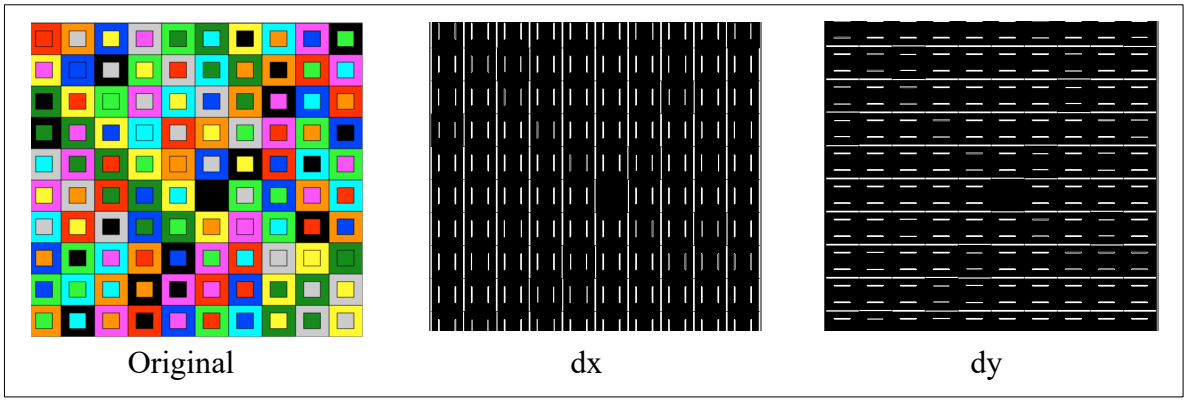
Juliette Rengot 16/02/2019

Harris Interest point dection for image processing

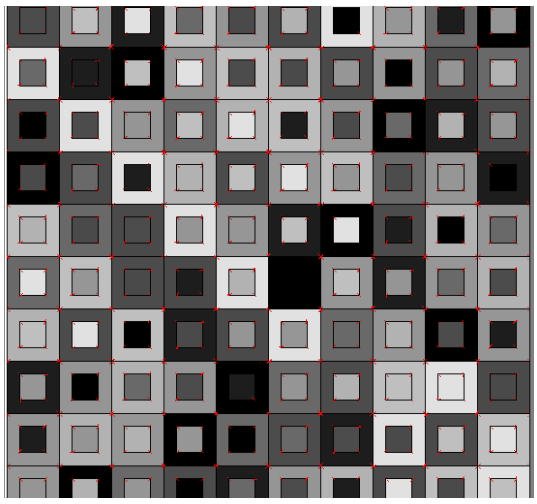
PART 1 : Harris corner detector

The Harris corner detection method is implemented in the function *findCorners* of the file *harris_corner.py*. In a first step, we compute the gradient of the image in both direction x and y. In the first case, we underline vertical lines, in the other case, we underline horizontal lines.



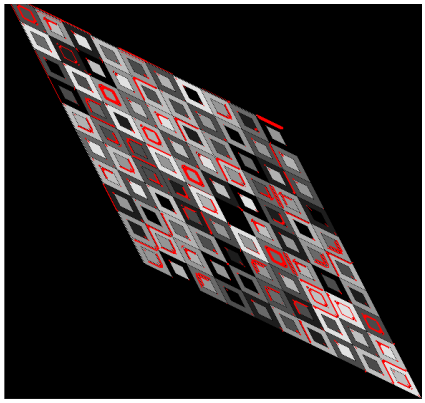
Next, we compute product images : $Ixx = dx^2$ | $Ixy = dy*dx$ | $Iyy = dy^2$
We applay smoothing function by using two times a 1D convolution with the following gaussian kernel : *1/sqrt(273)*[1, 4, 7, 4, 1]*. Now, we can compute the sums of the product images at each pixel. Then, we can create a Hessian matrix H. For each image point, we obtain the corner : response : *det(H) – k*trace(H)* where *k* is a Harris constant. First, I take *k=0,2*.

A point is considered as a corner if its corner response is bigger than a threshold. First, I take threshold = 10000. Here is what I obtain for the previous example. Detected corners are shown in red. It is exactly what was expected. We finf 2709 corners in this image.

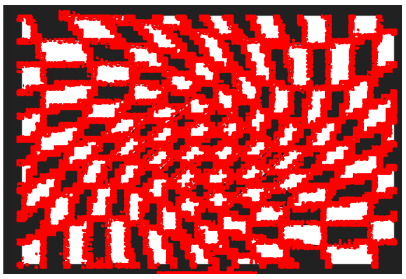
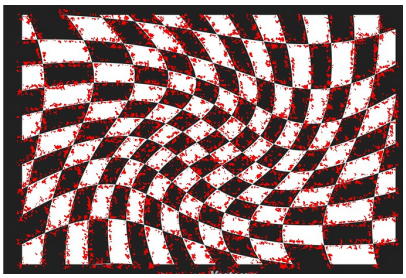
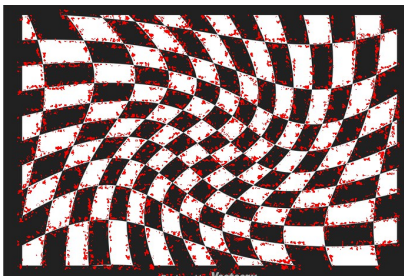
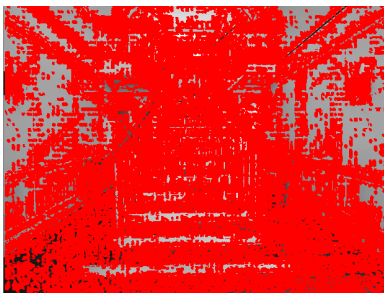


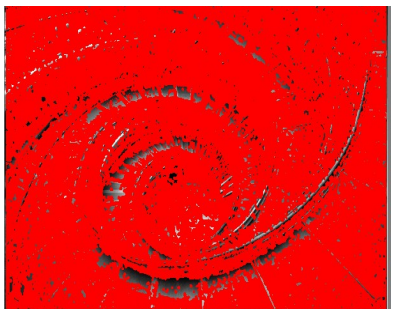
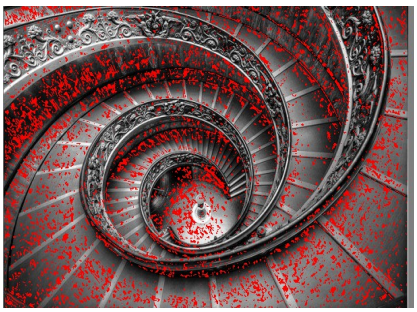
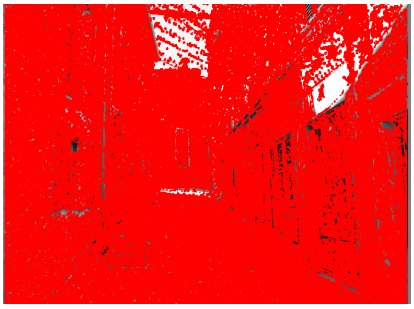
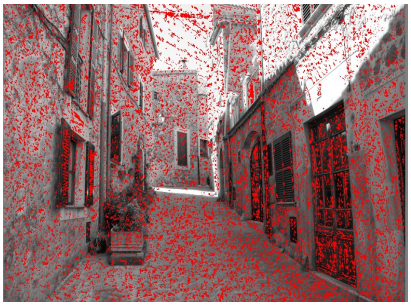
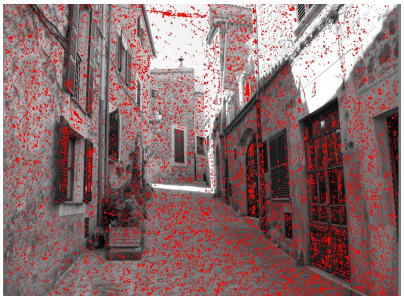
If we deform this image using *I'(x,y) = I(2x-y,-x+2y)*, we are still able to find relevant corners. To do this experiment, put *deform=True* in the code. We note that some edges are also detect as corners. We detect 8076 corners : there are many mistakes. So, harris detector is not invariant by

homographic transformation and that is a problem for the quality of the detector. We could change the parameters to suppress bad edges detection.

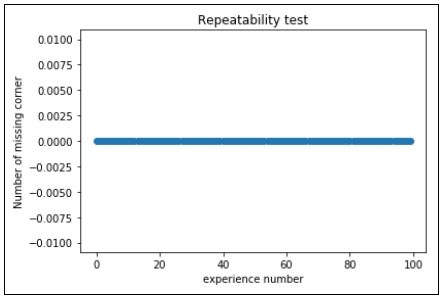


The following tables shows some results for different scenes and parameters. The tests are done on five good quality images. In all scene we are able to detect relevant corners. Distortion of space (experiment 1) does not prevent the corner detection. Moreover, the higher the thresh, the fewer the number of detected corners. It is obvious because the threshold is harder to overpass. The higher the Harris parameter k, the fewer the number of detected corners. Indeed, it makes corner response decrease. So the threshold is harder to overpass. We need to fit these parameters to get relevant harris points. We can see some that we tend to have many detected points in texture areas.

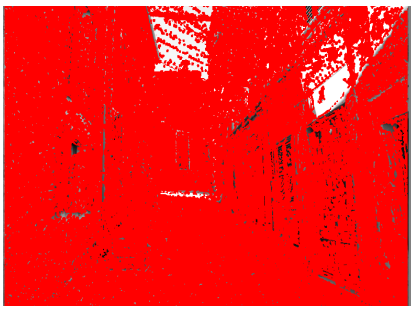
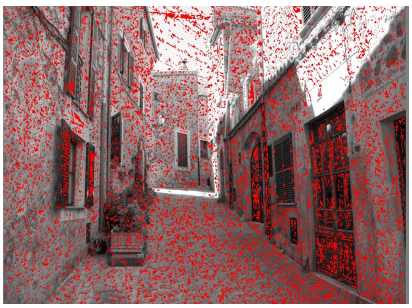
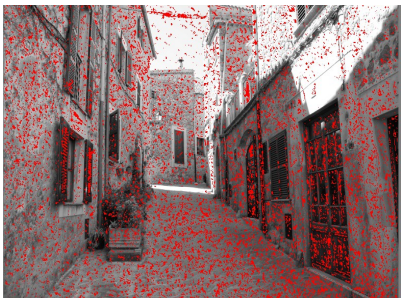
thresh=10000 k=0.04	thresh=10000 k=0,2	thresh=1000000 k=0.2
 158171 detected corners	 42601 detected corners	 31029 detected corners
 225515 detected corners	 43181 detected corners	 31018 detected corners
 101203 detected corners	 13571 detected corners	 9670 detected corners

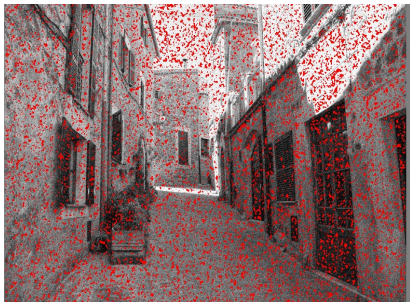
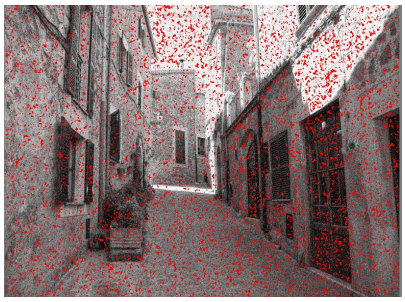
 215590 detected corners	 32349 detected corners	 24027 detected corners
 364701 detected corners	 66732 detected corners	 50027 detected corners

We can note than the repeatability of the experiment is good because we found common detected corners from one experiment to another. The evaluate more precicely this aspect, we work with the image 'notreDame2.png'. We store the corner list of a first experiment. We repeat the same experiment 100 times and we count how many point in the reference list are not in the new detected corners. Here is what we get : we always detect the same 3670 corners. To do this experiment, put *repeat_test=True* in the code. So there is rebeatability.

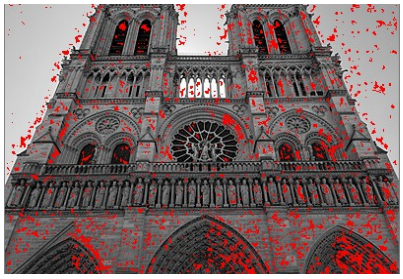
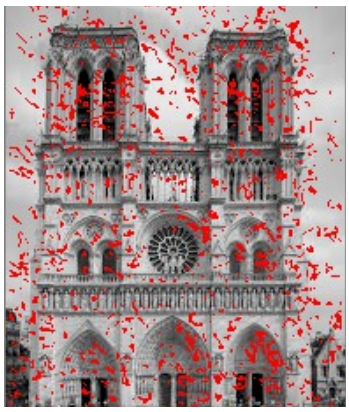
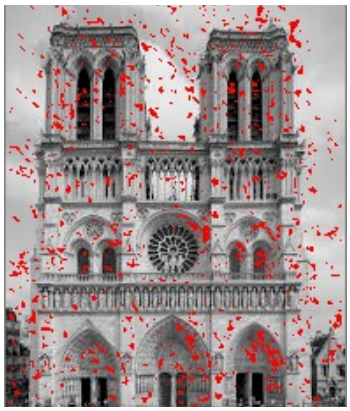


Next, we can try to add a gaussian noise (with a standard deviation of 25) to an image and see the impact on the harris detection performances. The obtained detection is deteriored by noise. Detected points seem to be random (cf sky area for example). We can find again the same relevant detected corners. So, Harris detection is sensitive to noise. It is a drawback for detectors.

thresh=10000 | k=0.04	thresh=10000 | k=0,2	thresh=1000000 | k=0,2
Original image		
 364701 detected corners	 66732 detected corners	 50027 detected corners

Noisy image		
		
393561 detected corners	52790 detected corners	38440 detected corners

Now, we wonder about what append if we change point of view. We do corner detection on another cathedral view. It seems to have a coherence between the results. Changes in view points is well adressed by Harris corner detection.

thresh=10000 | k=0.04	thresh=10000 | k=0,2	thresh=10000000 | k=0.04
		
101203 detected corners	13571 detected corners	9670 detected corners
		
42921 detected corners	4590 detected corners	2906 detected corners

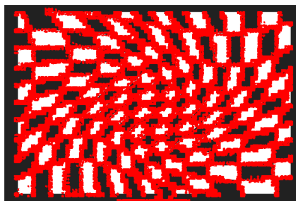
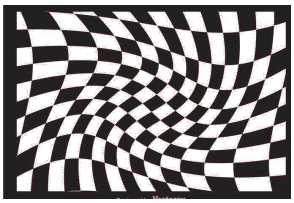
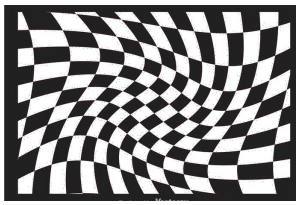
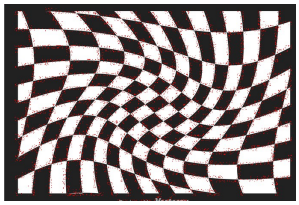
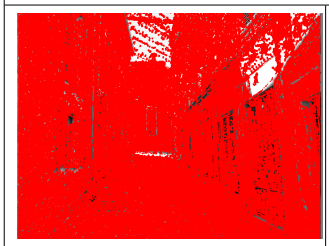
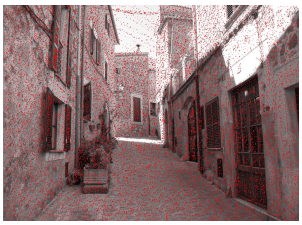
Last, we can note that features are not envely distributed. In the following example, there are more points in the right parts of cathedral arches.

PART 2 : Non maximum suppression (NMS)

We try to improve the results by suppressing points that are not local maximum. First we consider a neighbourhood of (2*k+1)*(2*k+1) pixels. If the maximal response is at the center, we keep the corner. Otherwise, we suppress it. To do this experiment, we have to change the *k_NMS* parameter in the code. The following examples are computed with *thresh=10000* and *k=0.04*.

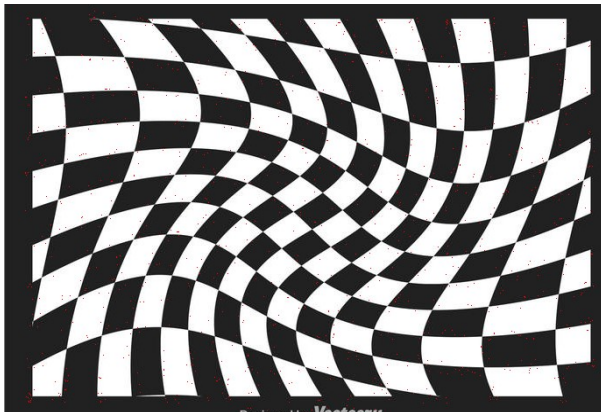
We aim at selecting relevant points. The high the NMS factor, the lower the number of detected corners. We can also use almost largest condition (last experiment),if we need a to select more points.

NMS increase a lot Harris detection performances by keeping only relevant detected points. It still leads to uneven distribution of features and it is still noise dependant.

k_NMS = 0	k_NMS = 1	k_NMS = 3	k_NMS = 0,9
 158171 corners	 5315 corners	 2302 corners	 14080 corners
Original image			
 364701 corners	 15370 corners	 5698 corners	 53683 corners
Noisy image			
 393561 corners	 16310 corners	 6063 corners	 43899 corners

PART 3 : Adaptive Non maximum suppression (ANMS)

Now, we test the adaptive method. We suppress point that are non maximum only if there close to other maximum corners. To do this experiment, we have to change the *N_ANMS* and c_*ANMS* parameters in the code. The following examples are done for *thresh=100000, k=0,2 N_ANMS=1000* and c_*ANMS=0,9*.






We obtained evenly distributed detected corners. It enables the descriptors to give information on the whole image and not just on some caracteristic parts. It is an important property for the features we are looking for. The computation time is bigger but the result is better.

The following example shows a noisy image. We obtain something really close to detected corners of the origianal image. Features are placed on relevant image points even if there is a gaussian noise. That is a really good property.

To conclude, a good keypoint detector needs to be invariant to view change, noise, illumination conditions, orientations, change of scale... They also need to be located in all images areas. So, Harris corner detectors, when selected with ANMS, are a simple and efficient methods. We don't suffer from noise or view point change and we collect features on evey parts of the image.