

Multimedia HW1

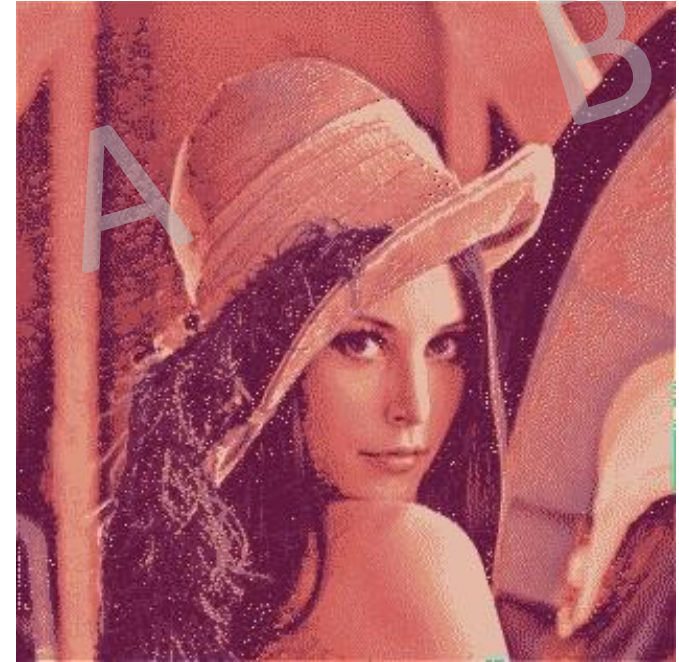
Problem1 – case: $n=3$



origin



Median-cut



Error diffusion dithering

Note : result image contain 8 color only.

Problem2



bilinear interpolation



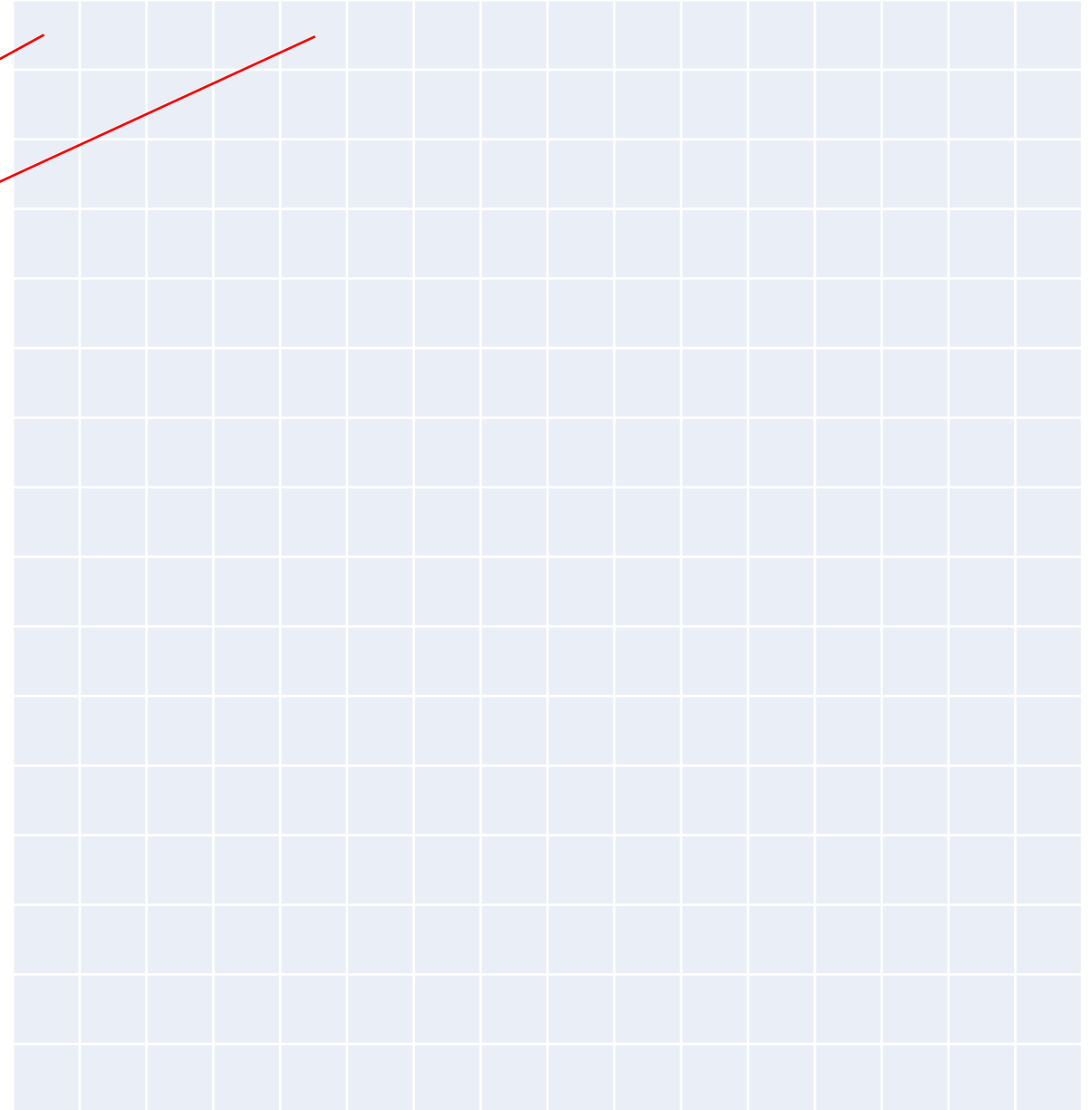
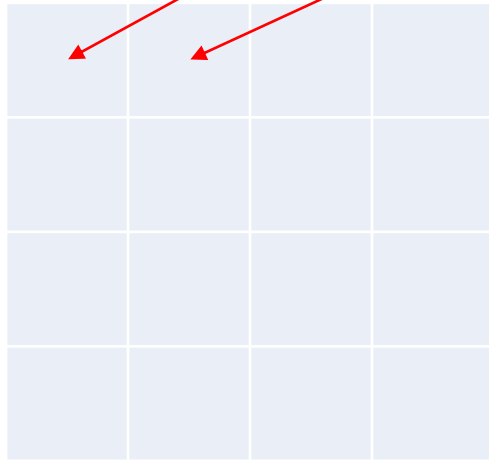
x4 upsample

Nearest interpolation



Problem2

- Hint : Inverse mapping



Problem3



Directory structure

- You can add directory “**exp**” to put your others experiment image.
- 可以額外增加自己的function set : `utils.py`

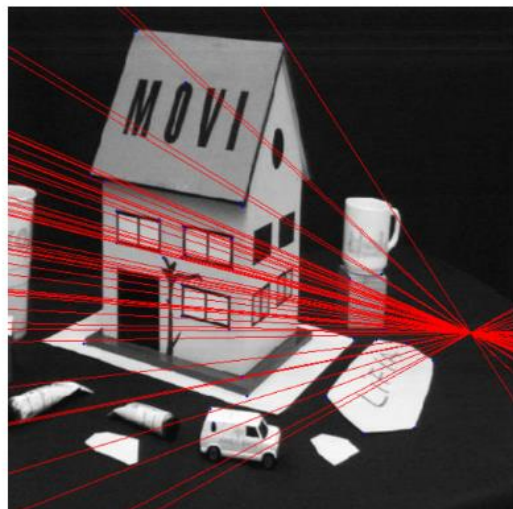
```
i HW1_111062547
├── 1.py
├── 2.py
├── 3.py
├── README
├── img -> The image program need
│   ├── Lenna.jpg
│   ├── bee.jpg
│   └── lake.jpg
├── out -> output image
│   ├── Y_hist.jpg
│   ├── Y_hist_gamma.jpg
│   ├── bee_linear.jpg
│   ├── bee_near.jpg
│   ├── error_diffusion_dithering_3.jpg
│   ├── error_diffusion_dithering_6.jpg
│   ├── gamma_img.jpg
│   ├── median_cut3.jpg
│   └── median_cut6.jpg
└── report.pdf

4 directories, 17 files
```

Report—example 参考

- Explain how to implement the code.
- The output result for each question.

Result:
a_img1



1. Fundamental Matrix Estimation

(a) Compute Fundamental Matrix without normalization

```
def Problem_ab(pts_1,pts_2,nlen):  
    """  
    least-square eight-point algorithm  
    """  
    # solve  $Ax=0$  where  $x$  the fundamental matrix  
    A=np.zeros((nlen,9))  
  
    for i in range(nlen):  
        A[i,:]=np.array([pts_1[i,0]*pts_2[i,0],pts_1[i,0]*pts_2[i,1],pts_1[i,0]*pts_2[i,2],  
        pts_1[i,1]*pts_2[i,0],pts_1[i,1]*pts_2[i,1],pts_1[i,1]*pts_2[i,2],  
        pts_1[i,2]*pts_2[i,0],pts_1[i,2]*pts_2[i,1],pts_1[i,2]*pts_2[i,2]])  
  
    U,S,VT=np.linalg.svd(A)  
  
    F=VT[-1,:].reshape((3,3))  
  
    # now we need to enforce F to rank 2 use SVD to find the approximate matrix under frobenius norm  
    U,S,VT=np.linalg.svd(F)  
    F=U[:, :2]@np.diag(S[:2])@VT[:2, :]  
    # print("rank %d" % np.linalg.matrix_rank(F))  
    print("Fundamental matrix: ")  
    print(F)  
  
    return F
```

Step:

1. Use SVD to solve the least square solution that is eigenvector corresponding to the last eigenvalue. The solution x is the element of fundamental matrix F .
2. F is not rank 3, we use SVD to find the best approximate solution constrain with rank 2 under frobenius norm.
3. show the fundamental matrix

```
not normal :  
Fundamental matrix:  
[[-5.63087200e-06 -2.77622828e-05  1.07623595e-02]  
 [ 2.74976583e-05 -6.74748522e-06 -1.22519240e-02]  
 [-6.42650411e-03  1.52182033e-02 -9.99730547e-01]]
```

Code – example 参考

```
1  def Problem_a():
2      # todo : implement problem a
3      pass
4
5  def Problem_b():
6      # todo : implement problem a
7      pass
8
9  if __name__ == '__main__':
10     # problem a
11     Problem_a()
12
13     # problem b
14     Problem_b()
15
```