COMPUTER VISION REPORT ASSIGNMENT #2

Evahn LE GAL

9IE24004S

06/04/2024

PLAN

- 5-points Algorithm
- Computation of E
- Associated Epipolar Lines
- Results A-2
- Normals Surface
- Results B-2

All the code and the images can be find on my GitHub repository :
https://github.com/Vurremt/report2

STRUCTURE OF THE PROGRAM

Question A-2 :

- Part 1 : Program in C++ → Use of OpenCV to obtain essential matrix E which is/are saved in the folder resources\report1_F\A2_matrix_E\
- Part 2 : First Half of the Python → Exploitation of E and the results of first report to obtain epipolar lines of E compared to the previous F

Question B-2 :

- Part 3 : Second Half of the Python → Open data in the folder resources\photometric\ and compute the normal surface

Note : the images obtained by parts 2 and 3 are saved in the folder resources\output\

A-1 : 5-POINTS ALGORITHM

- The 5-point algorithm in stereo optics is used to determine the relative position of two cameras from points in two images of the same object.
- We use 5 pairs of points to determine the essential matrix E (and thus obtain the rotation matrix R and a translation vector t with an SVD decomposition, which can be used to reconstruct the 3D scene).
- The advantages of this method are that it is more resistant to noise than other algorithms (8-point) and is more robust to errors in point correspondence to a certain degree.
- The disadvantages are greater calculation time, as well as the possibility of obtaining several possible results due to the nature of the polynomial equation. With only 5 points, the system is generally underdetermined, meaning this leads to a space of potential solutions rather than a single solution.

A-2 : COMPUTE E

list_of_points.txt

```
img1[ 147 ; 496 ]/img2[ 255 ; 452 ]/ff0000/red
img1[ 498 ; 307 ]/img2[ 443 ; 276 ]/04ff00/green
img1[ 983 ; 639 ]/img2[ 953 ; 656 ]/1f00ff/blue
img1[ 950 ; 388 ]/img2[ 1073 ; 368 ]/cf00ff/purple
img1[ 485 ; 782 ]/img2[ 337 ; 758 ]/feff00/yellow
img1[ 1004 ; 746 ]/img2[ 1042 ; 775 ]/ff7900/orange
img1[ 1019 ; 580 ]/img2[ 1109 ; 589 ]/ffffff/white
img1[ 738 ; 490 ]/img2[ 705 ; 474 ]/00daff/cyan
```

We enter the number of points we want to compute (5 or more), here we choose 5 :

```
Number of points for the Algorithm : 5
Points loaded : (img1.x ; img1.y / img2.x ; img2.y)
147 ; 496 / 255 ; 452
498 ; 307 / 443 ; 276
983 ; 639 / 953 ; 656
950 ; 388 / 1073 ; 368
485 ; 782 / 337 ; 758
```

```
cv::Mat E, mask;
E = cv::findEssentialMat(points1_cv,points2_cv,K1,cv::RANSAC, 0.999, 1.0, mask);
int rows = E.rows;
int cols = E.cols;
```

```
Essential matrix E :
[0.000452129486073588, -9.702740843674104e-05, -0.6851281567846813;
 -3.860315884971207e-05, -0.0005087515333176008, -0.1749418792285224;
 0.2144663030753841, 0.6737940872651378, -7.956644369582442e-05;
 6.225711872626208e-05, 0.001128898646869054, -0.5721271525458089;
 -0.001217830984965176, 0.0002121721383125852, 0.4155349742532014;
 0.5307912360608228, -0.4671820425571707, 5.392514910845311e-05;
 -0.000755771305602236, 0.002013363602055954, 0.3707955311045483;
 -0.001574096248190647, 0.001345906985816005, -0.6020878462937881;
 0.3668966789120716, -0.6044654588538988, -0.0005669622279423776;
 -3.414784513018487e-06, -0.0004614536937277809, 0.1984557295998347;
 0.0005928966963506747, -9.844647327867652e-05, -0.6786862288479263;
 -0.2316139611140145, 0.6680978232330447, -0.0001536233962002647]
Dimensions : [12 ; 3]
```

E1

E2

E3

E4

We obtain E but there are 4 potentials solutions, so we saved the 4 differents matrix independentely

```
### ## # A-2 : Compute E # ## ###


Number of points loaded :  8
[[147, 496], [498, 307], [983, 639], [950, 388], [485, 782], [1004, 746], [1019,
 580], [738, 490]]
[[255, 452], [443, 276], [953, 656], [1073, 368], [337, 758], [1042, 775], [1109
, 589], [705, 474]]

F of report 1 =  [[-1.66142220e-08 -2.16144774e-05  5.65711864e-03]
 [ 6.14723209e-06  3.09199007e-06 -7.46119460e-02]
 [-1.73805365e-03  7.87369544e-02  9.94081750e-01]]
```

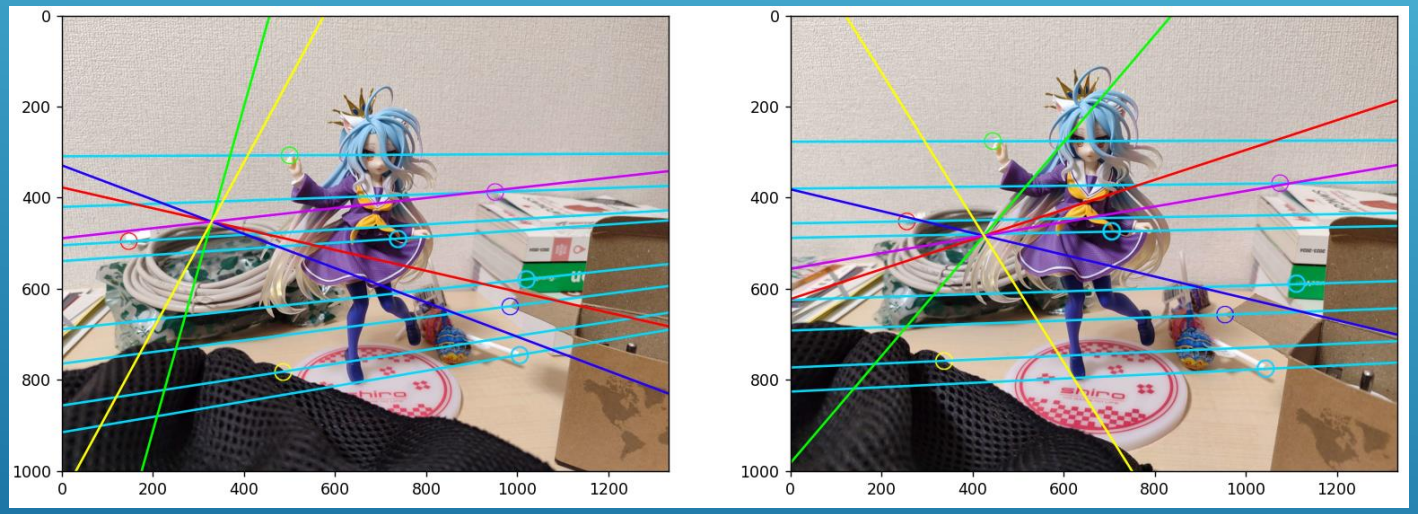
+

```
Give the name of the file with the matrix E : matrix_E2.txt

 E loaded =  [[ 6.22571187e-05  1.12889865e-03 -5.72127153e-01]
 [-1.21783098e-03  2.12172138e-04  4.15534974e-01]
 [ 5.30791236e-01 -4.67182043e-01  5.39251491e-05]]

 New F =  [[ 6.22571187e-05  1.12889865e-03 -5.72127153e-01]
 [-1.21783098e-03  2.12172138e-04  4.15534974e-01]
 [ 5.30791236e-01 -4.67182043e-01  5.39251491e-05]]

How many items do you want to process? (choose the same number as for the calcul
ation of E in C++, 5 by default for the 5 point algorithm) : 5
```

We just re-use the list of points of report 1 and we re-compute f with the same method as report 1

We choose the name of the matrix we want to use (here E2) and compute the new F as $K^T E K$ (here $K = I_3$)

In cyan, we see the epipolar lines of the report 1, and in color the epipolar lines of the 5 points with which we computed E



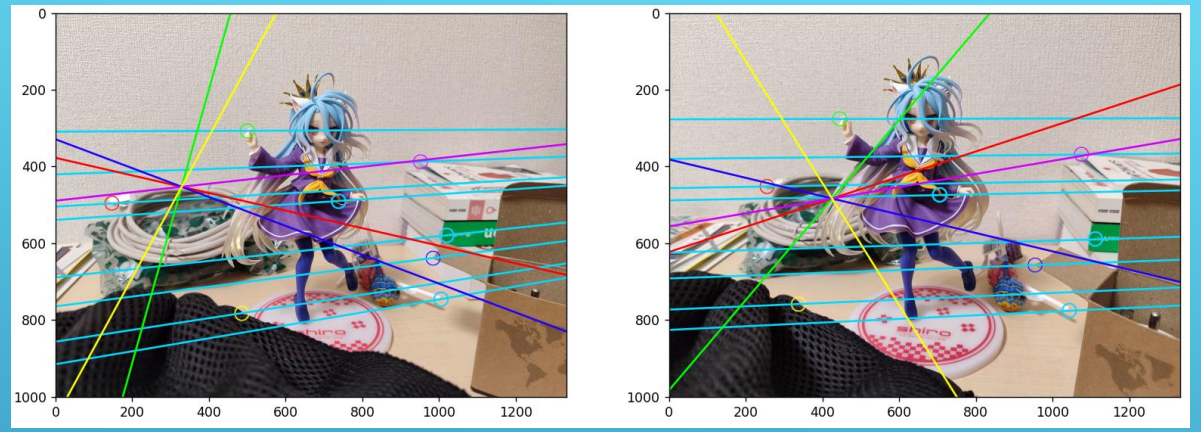
A-2 : F AND EPIPOLAR LINES

A-2 : F AND EPIPOLAR LINES

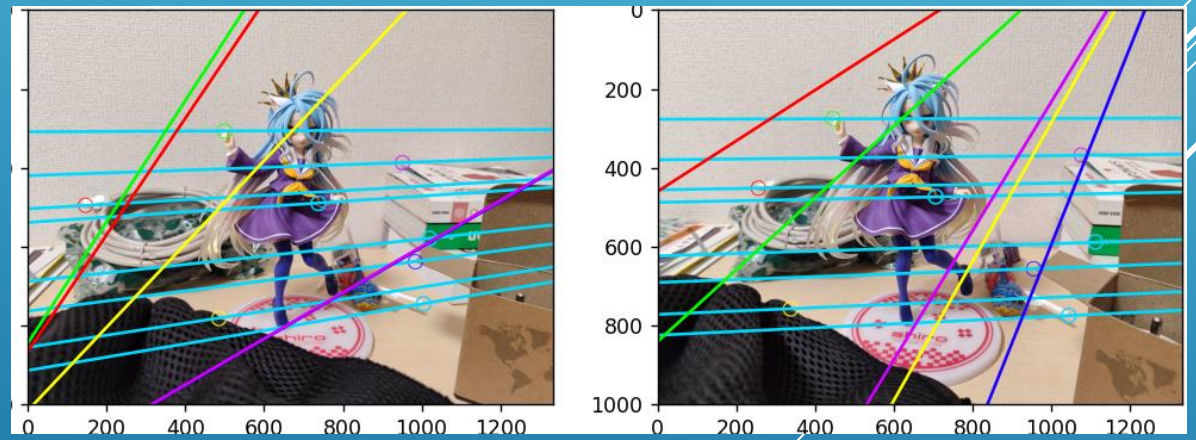
Among the possible essential matrices E (when there are several) there would need to be a way to merge them. **RANSAC's algorithm** is a good way to do this, but was not implemented here because it was too difficult, and we couldn't find enough help on the internet about it. Which means we get several E's, and we have to try them all.

We see in the images on the right that E2 is quite correct, but that E1 is completely wrong. We can't be sure which one is better until we tested it.

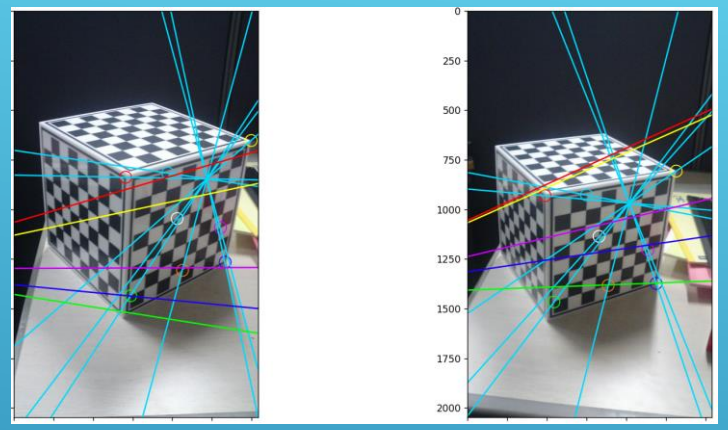
5 points : `matrix_E2.txt`



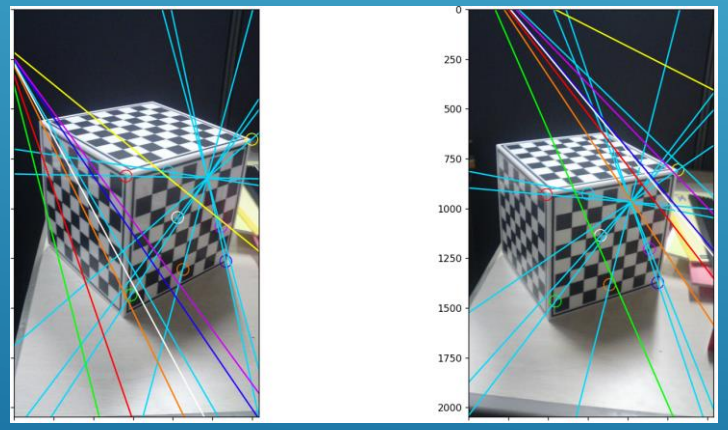
5 points : `matrix_E1.txt`



5 points : `matrix_E2.txt`



8 points : `matrix_E1.txt`



On the images on the left (used in report 1 because the epipolar lines were perfect), we can see that the matrices are not precise at all. We can also have more than 5 points to increase the precision, for example 8 points gives us a single matrix E1, but it is also false.

This can come from the imprecision of our points, from the calculation of E by the `findEssentialMat()` function of OpenCV, but especially from K. In fact, we do not know the intrinsic matrix of our camera, so we take by default $K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, but this causes a lot of errors.

A-2 : F AND EPIPOLAR LINES

B-2 : PHOTOMETRIC STEREO METHOD

Algorithm :

- Image acquisition: Three images of the object are chosen under known lighting directions. We pass the images in grayscale for greater calculation simplicity.
- Calculation of the reflectance curves: For each pixel of the image, we find the coordinates (p, q) as the intersection of the three reflectance curves $R1(p, q)$, $R2(p, q)$ and $R3(p, q)$ corresponding to the three images captured according to the light intensities.
- Estimation of surface normals: We solve a system of linear equations to estimate the vectors normal to the surface of the object, which indicate the orientation of each point on the surface relative to the light sources.
- Conversion to color image: We normalize the normal vectors and convert each pixel into color values, to obtain a color image which visually represents these normal vectors.

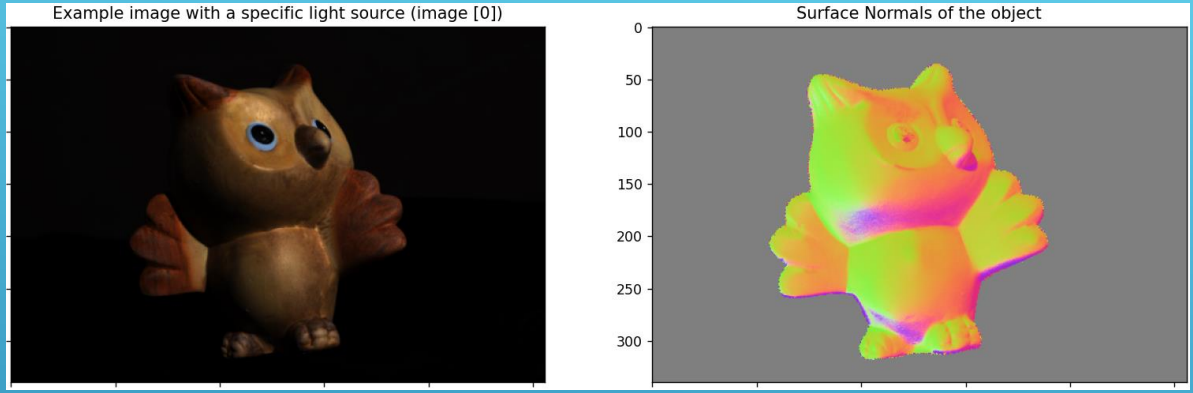
lights.txt

```
0.403259 0.480808 0.778592
0.0982272 0.163712 0.981606
-0.0654826 0.180077 0.98147
-0.127999 0.431998 0.892745
-0.328606 0.485085 0.810377
-0.110339 0.53593 0.837021
0.239071 0.41439 0.878138
0.0642302 0.417497 0.906406
0.12931 0.339438 0.931698
0.0323953 0.340151 0.939813
0.0985318 0.0492659 0.993914
-0.16119 0.354617 0.921013
```

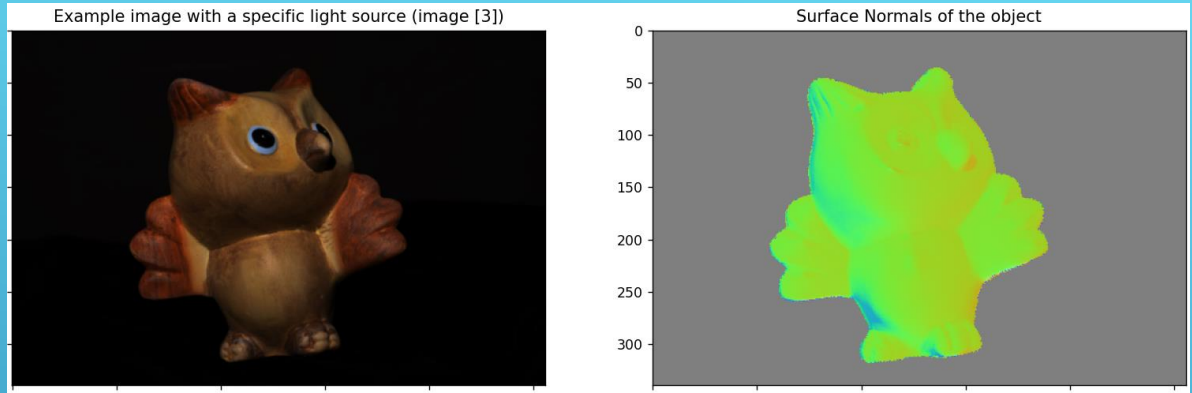
Extract only 3 lights (first 0th, last 11th)

```
### ## # B-2 : Photometric Stereo Method # ## ###


Type the camera numbers (between 0 and 11, need 3 cameras)

Camera number you want to use : 0
Camera number you want to use : 5
Camera number you want to use : 11

S_list :
 [[0.403259, 0.480808, 0.778592], [-0.110339, 0.53593, 0.837021],
[-0.16119, 0.354617, 0.921013]]
```

With lights 0 ; 5 ; 11



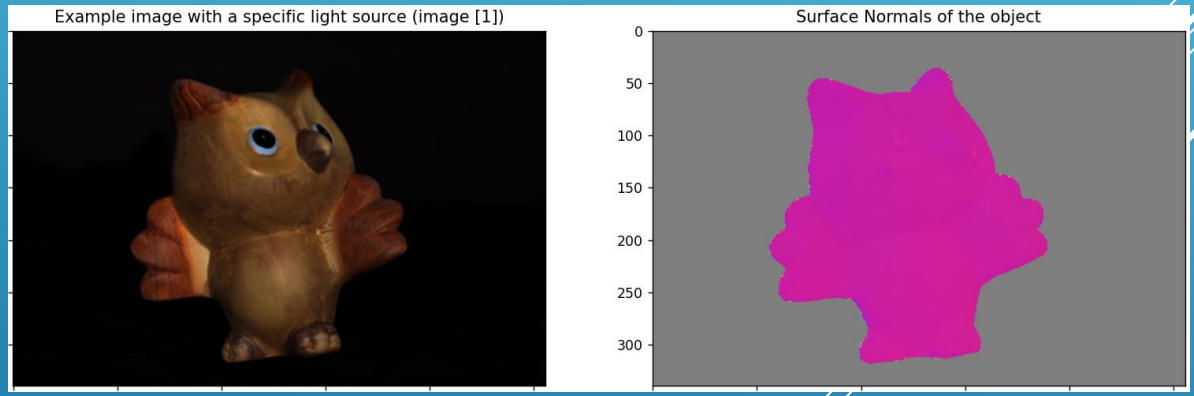
With lights 3 ; 6 ; 9



We clearly see that choosing our lights has a huge impact on the result. Indeed, since the lights are "sorted" in the file, we can define "near" light points and other "far away" ones. Indeed, lights 4 and 5 are very close spatially, and the same for all the lights in the list. We therefore notice that when we take 3 images with very close lights (1, 2 and 3), the lighting is "a bit the same" for all 3, the calculation of the normal is therefore completely distorted and we obtain a image with very little variation.

Conversely, when we take very opposite lights (0, 5 and 11), we obtain a very detailed normal map, because the lights cover a large part of the object, and we can extract a greater number of elements from it. Finally, we notice that a case of average reconciliation (3, 6 and 9) gives an intermediate result, which makes sense.

With lights 1 ; 2 ; 3



B-2 : PHOTOMETRIC STEREO METHOD