

COMPUTER VISION **REPORT 1**

BRICE Chloe – 9IE24001M

CONTENT

Project structure : What are the steps and how the code is implemented

Epipolar Geometry : Results with same plane and not same plane

Extra points: Additional computation with OpenCV

PROJECT STRUCTURE / 8-PTS ALGORITHM

This project is implemented in Python, and C++ for the OpenCV Part. Images are stored in the img folder. We will consider images with various target objects on the same plane, and images with various target objects on different planes.



The aim is to compute the fundamental matrix, and find camera parameters.

To do so, we use the 8-points algorithm : the points coordinates are described in a .txt file, which is parsed by the program. Then we can begin the computation.

First of all, we construct matrix A :

$$\begin{array}{ccc} (x_1, y_1) & \leftrightarrow & (x'_1, y'_1) \\ (x_2, y_2) & \leftrightarrow & (x'_2, y'_2) \\ (x_3, y_3) & \leftrightarrow & (x'_3, y'_3) \\ & \vdots & \\ (x_n, y_n) & \leftrightarrow & (x'_n, y'_n) \end{array}$$
$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & x'_1y_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ x_2x'_2 & x_2y'_2 & x_2 & x'_2y_2 & y_2y'_2 & y_2 & x'_2 & y'_2 & 1 \\ x_3x'_3 & x_3y'_3 & x_3 & x'_3y_3 & y_3y'_3 & y_3 & x'_3 & y'_3 & 1 \\ & & & \vdots & & & & & \\ x_nx'_n & x_ny'_n & x_n & x'_ny_n & y_ny'_n & y_n & x'_n & y'_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

$Af = 0$

known  unknown 

After following the steps of matrix decomposition of 8-points algorithm (detailed in the code) we can finally get F the fundamental matrix.

RESULTS OF SAME PLANE

A MATRIX COMPUTATION

Results of A computation :

Number of points loaded : 8

```
[[289, 483], [620, 473], [766, 607], [888, 504], [570, 532], [395, 577], [455, 611], [1211, 402]]  
[[350, 413], [675, 405], [816, 540], [947, 428], [621, 463], [451, 510], [507, 542], [1299, 308]]
```

Print A ?

[y/n] : y

```
[[1.011500e+05 1.193570e+05 2.890000e+02 1.690500e+05 1.994790e+05  
 4.830000e+02 3.500000e+02 4.130000e+02 1.000000e+00]  
[4.185000e+05 2.511000e+05 6.200000e+02 3.192750e+05 1.915650e+05  
 4.730000e+02 6.750000e+02 4.050000e+02 1.000000e+00]  
[6.250560e+05 4.136400e+05 7.660000e+02 4.953120e+05 3.277800e+05  
 6.070000e+02 8.160000e+02 5.400000e+02 1.000000e+00]  
[8.409360e+05 3.800640e+05 8.880000e+02 4.772880e+05 2.157120e+05  
 5.040000e+02 9.470000e+02 4.280000e+02 1.000000e+00]  
[3.539700e+05 2.639100e+05 5.700000e+02 3.303720e+05 2.463160e+05  
 5.320000e+02 6.210000e+02 4.630000e+02 1.000000e+00]  
[1.781450e+05 2.014500e+05 3.950000e+02 2.602270e+05 2.942700e+05  
 5.770000e+02 4.510000e+02 5.100000e+02 1.000000e+00]  
[2.306850e+05 2.466100e+05 4.550000e+02 3.097770e+05 3.311620e+05  
 6.110000e+02 5.070000e+02 5.420000e+02 1.000000e+00]  
[1.573089e+06 3.729880e+05 1.211000e+03 5.221980e+05 1.238160e+05  
 4.020000e+02 1.299000e+03 3.080000e+02 1.000000e+00]]
```

F MATRIX COMPUTATION

We can now decompose and find F :

Print U, Σ and V^T ?

[y/n] : y

U = [[0.09887603 0.26719161 0.36726067 -0.50074828 0.61999635
0.28574162

-0.20513766 -0.15835951]

[0.24931231 0.16851306 -0.25537106 -0.52122265 -0.57476512 0.30073775
0.12212523 -0.36992781]

[0.38481029 0.32116791 -0.36891667 0.5502247 0.20261617 0.51554696
-0.04868719 -0.0267031]

[0.43899974 0.0212013 -0.44160568 -0.1695766 0.37054083 -0.60549595
0.24531194 -0.13767402]

[0.23648868 0.28274237 -0.12036181 -0.24244059 -0.17577004 -0.11063064
-0.41316247 0.75962358]

[0.16096742 0.39107527 0.42514224 0.05815008 -0.07956682 -0.00572286
0.74463216 0.2760101]

[0.1970115 0.43673747 0.37444993 0.28539467 -0.25150156 -0.4145121
-0.38857339 -0.40338739]

[0.68309313 -0.61072984 0.36516366 0.0496555 -0.08010457 0.1067818
-0.06546768 0.04977497]]

Σ = [2.41468660e+06 6.94017221e+05 1.02237758e+05 1.38119674e+04
4.99194490e+01 2.17661963e+01 1.71953939e+00 1.45367205e+00]

V^T = [[8.10223783e-01 3.30740211e-01 8.21221252e-04 4.28296347e-01
2.25185846e-01 5.11116104e-04 8.85957192e-04 4.33289063e-04
1.01444183e-06]

[-5.39045957e-01 3.57945213e-01 3.18864481e-04 4.03039025e-01
6.47195077e-01 1.16970753e-03 3.88260996e-04 1.06635362e-03
1.84130755e-06]

[2.17799761e-01 -5.70245975e-01 -1.46862203e-04 -3.46223279e-01
7.12395768e-01 1.63299753e-03 1.77208248e-04 1.33559702e-03

3.38193346e-06]

[7.43598495e-02 6.61172224e-01 -8.84889602e-03 -7.30778557e-01
1.51390964e-01 -1.02062278e-02 -1.11370953e-02 -7.67286078e-03
-3.55172548e-05]

[-8.46173060e-05 -5.69841599e-03 -7.20938365e-01 7.66257636e-03
-1.67018334e-03 2.41303251e-01 -6.27725872e-01 1.66992409e-01
6.29919731e-04]

[8.58758110e-04 1.11198530e-02 7.60381105e-02 -1.24151893e-02
-2.33050070e-04 7.13537635e-01 3.47258121e-01 6.03493486e-01
3.32839837e-03]

[-5.51616219e-05 -8.62956033e-04 -2.78956369e-01 9.45806028e-04
-8.39032210e-05 -6.50281347e-01 2.46579058e-01 6.62183878e-01
-5.21014553e-03]

[8.70376040e-05 2.96510611e-04 6.29729344e-01 -4.21771065e-04
-2.49020713e-04 -9.80738725e-02 -6.51481208e-01 4.11510344e-01
-7.32157726e-03]

[-3.80054139e-07 -1.13305272e-05 3.35809659e-03 1.23752713e-05
1.11503719e-06 -6.63373167e-03 -4.24614725e-03 4.34904171e-03
9.99953885e-01]]

F = [[-4.73740750e-07 -1.13802302e-05 3.35809641e-03]
[1.22939146e-05 1.07187547e-06 -6.63373183e-03]
[-4.24614748e-03 4.34904160e-03 9.99953885e-01]]

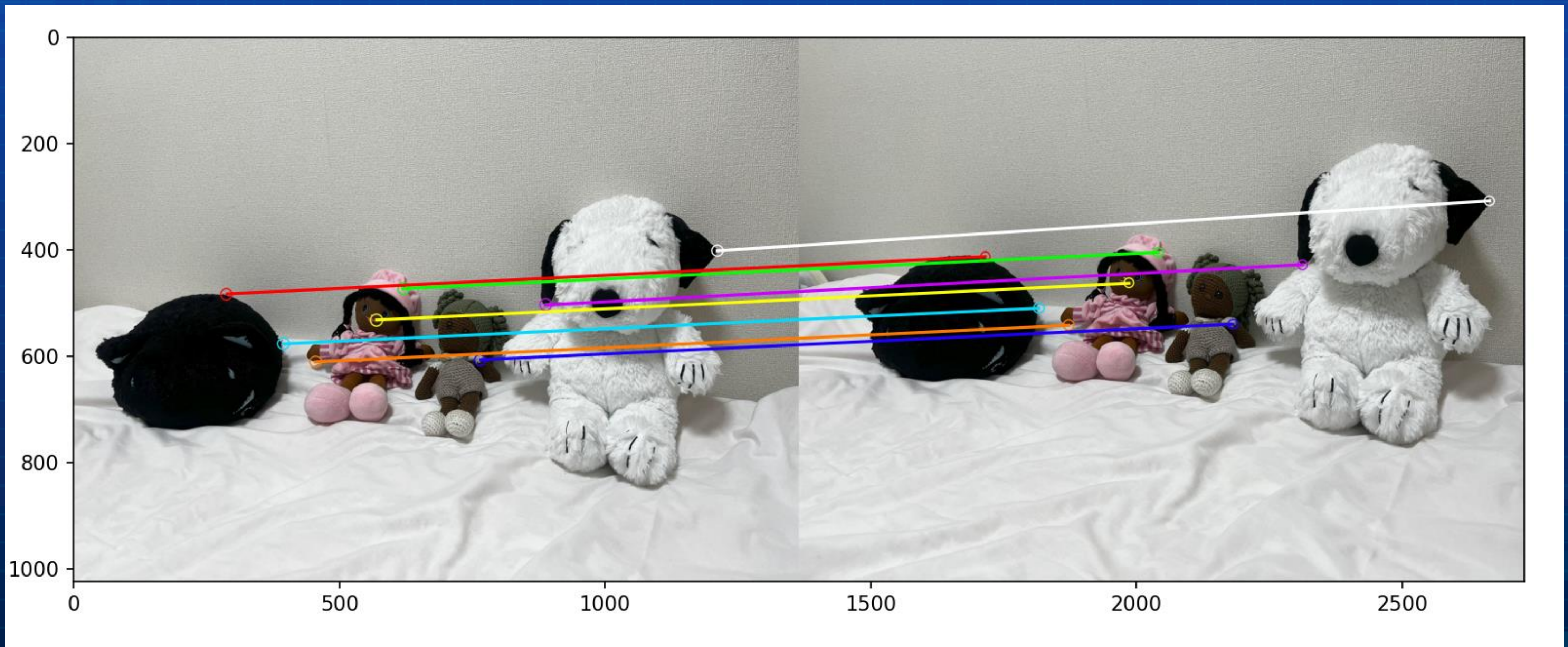
- - - - - Question A-3 : Verification of F - - - - -

The eigenvalues of F are : [9.99910772e-01 2.09169242e-19 4.37114951e-05]

The rank of F is : 2

CONNECTING LINES

Highlight of points used for computation and connecting lines



EPIPOLAR LINES

We can now also have the epipolar lines equations and evaluate the quality of the F matrix's computation.

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} f_{11}x' + f_{12}y' + f_{13} \\ f_{21}x' + f_{22}y' + f_{23} \\ f_{31}x' + f_{32}y' + f_{33} \end{bmatrix} = 0$$

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

$$ax + by + c = 0$$

- - - - - Question A-4 : Draw Epipolar Lines - - - - -

--- Epipolar lines in image 1 and image 2 ---

Img 1 : Point 0 : -0.001507747939062417 x + -0.0018881771519421936 y + 1.3099564465851883 = 0

Img 2 : Point 0 : 0.0015549021949291062 x + 0.0015778709089195515 y + -1.233648725800892 = 0

Img 1 : Point 1 : -0.0015706718409847467 x + 0.0020987700882673004 y + -0.10483381669984237 = 0

Img 2 : Point 1 : 0.0012751548606972068 x + -0.0021997040519530853 y + -0.055781496289566856 = 0

Img 1 : Point 2 : -0.003173800367823688 x + 0.0039769152343487 y + -0.11641999581155993 = 0

Img 2 : Point 2 : 0.0028533732671534888 x + -0.0037175863530931877 y + -0.4544194856417696 = 0

Img 1 : Point 3 : -0.0019612746202922683 x + 0.005467367994303006 y + -1.159757974169347 = 0

Img 2 : Point 3 : 0.0015293036921842718 x + -0.005216377614204856 y + 0.638542654521006 = 0

Img 1 : Point 4 : -0.002205143193836446 x + 0.0014970674770850853 y + 0.37670255966647215 = 0

Img 2 : Point 4 : 0.0020241828594068567 x + -0.0015674518879331663 y + -0.615076494589639 = 0

Img 1 : Point 5 : -0.0026594780871503835 x + -0.0005425198574471385 y + 1.3029525859945343 = 0

Img 2 : Point 5 : 0.0026603136475176837 x + 0.00047232279833980427 y + -1.501261298360641 = 0

Img 1 : Point 6 : -0.0030501749365206055 x + 0.0001802393748895851 y + 1.2043376582453469 = 0

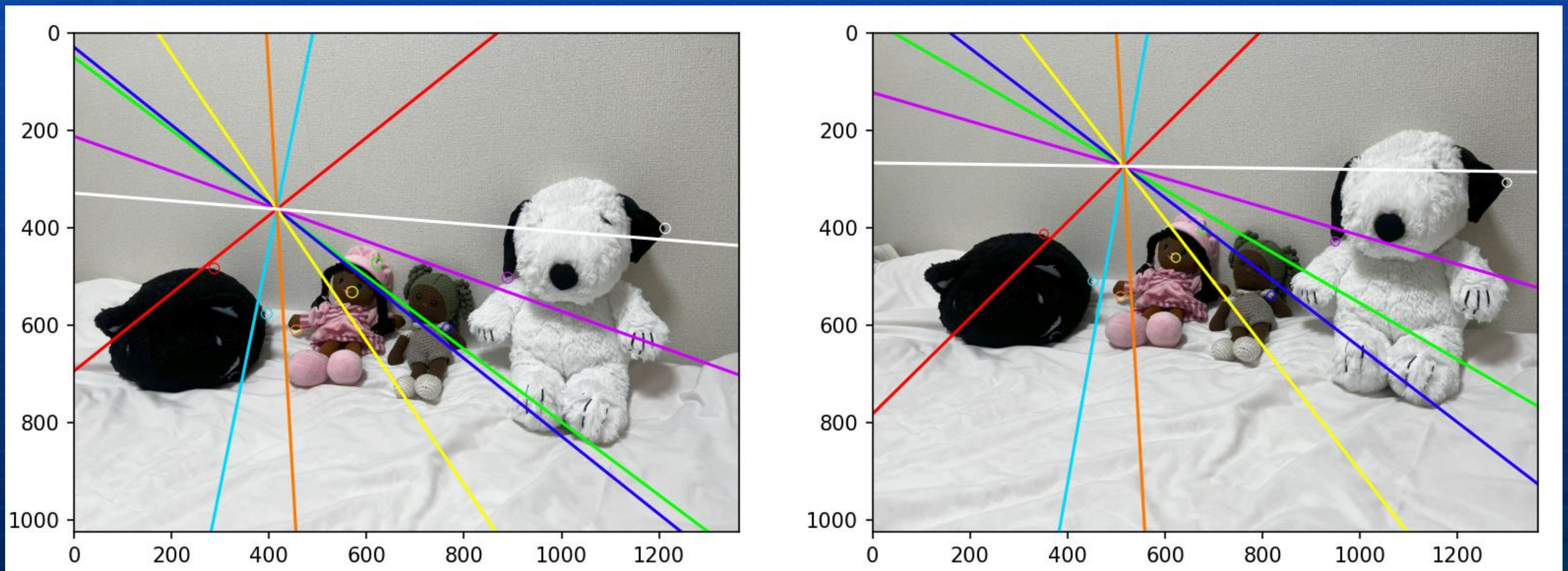
Img 2 : Point 6 : 0.0030498822988040903 x + -0.00017404724962280516 y + -1.525322396012105 = 0

Img 1 : Point 7 : -0.0007624037366786127 x + 0.009666200876406211 y + -3.176286878044844 = 0

Img 2 : Point 7 : 0.0001223061410529653 x + -0.009001523276129051 y + 2.3998484409725998 = 0

EPIPOLAR LINES

Here are the epipolar lines represented :



We can see that the results aren't very satisfactory, but still relevant; the epipolar lines are close to the corresponding points. When the target objects are on the same plane, we observe that the lines all converge to the same point, as we enforce F to be rank 2. However, we might have expected the lines to intersect outside of the image, as the camera has just moved slightly from left to right.

CAMERA PARAMETERS

Calculation of camera parameters:

Both images have been taken with iPhone 13 camera. We now want to find the focal length of the camera, and the image center.

We need to find epipole to resolve Kruppa equations ; it corresponds to the null space of F matrix : so with singular value decomposition we can find the left and right epipole while taking the last column of F matrix decomposition V matrix. Let's call them e1 and e2.

We then have to extend e1 and e2 to matrix form $[x]$. Then Kruppa equations use the image of absolute dual quadric to find the camera parameters:

$$\begin{bmatrix} e_1 \end{bmatrix}^t \omega_1^* \begin{bmatrix} e_1 \end{bmatrix} \approx F \omega_2^* F^t \rightarrow 2 \text{ constraints for } \omega^* \approx AA^t$$

With the image of the absolute dual quadric, we can find A which contains the information we are interested in:

$$A = \begin{bmatrix} f\alpha_x & fs & x_0 \\ 0 & f\alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

CAMERA PARAMETERS

Results of epipoles computation:

- - - - - Question B-7 : Intrinsic Camera Parameters - - - - -

Would you like to find intrinsic camera parameters with Kruppa equations ?

[y/n] : y

Left epipole: [416.19925193 361.42418181 1.]

Right epipole: [515.73919625 273.61219166 1.]

Left epipole (matrix form): [[0. 1. 361.42418181]

[1. 0. -416.19925193]

[-361.42418181 416.19925193 0.]]

Right epipole (matrix form): [[0. 1. 273.61219166]

[1. 0. -515.73919625]

[-273.61219166 515.73919625 0.]]

Sadly, I did not find out how to find w^* with $w1^*$ and $w2^*$.

However, the focal length of iPhone 13 camera is 77 mm, and we can expect the image center to be close to the center of the image.

RESULTS OF NOT SAME PLANE

A MATRIX COMPUTATION

- - - - - Question A-2 : Compute F - - - - -

Number of points loaded : 8

[[1040, 566], [632, 271], [423, 406], [303, 313], [911, 157], [462, 546], [20, 331], [956, 379]]
[[979, 606], [733, 267], [457, 400], [369, 304], [862, 149], [476, 540], [111, 316], [888, 395]]

Print A ?

[y/n] : y

```
[[1.01816e+06 6.30240e+05 1.04000e+03 5.54114e+05 3.42996e+05 5.66000e+02
 9.79000e+02 6.06000e+02 1.00000e+00]
[4.63256e+05 1.68744e+05 6.32000e+02 1.98643e+05 7.23570e+04 2.71000e+02
 7.33000e+02 2.67000e+02 1.00000e+00]
[1.93311e+05 1.69200e+05 4.23000e+02 1.85542e+05 1.62400e+05 4.06000e+02
 4.57000e+02 4.00000e+02 1.00000e+00]
[1.11807e+05 9.21120e+04 3.03000e+02 1.15497e+05 9.51520e+04 3.13000e+02
 3.69000e+02 3.04000e+02 1.00000e+00]
[7.85282e+05 1.35739e+05 9.11000e+02 1.35334e+05 2.33930e+04 1.57000e+02
 8.62000e+02 1.49000e+02 1.00000e+00]
[2.19912e+05 2.49480e+05 4.62000e+02 2.59896e+05 2.94840e+05 5.46000e+02
 4.76000e+02 5.40000e+02 1.00000e+00]
[2.22000e+03 6.32000e+03 2.00000e+01 3.67410e+04 1.04596e+05 3.31000e+02
 1.11000e+02 3.16000e+02 1.00000e+00]
[8.48928e+05 3.77620e+05 9.56000e+02 3.36552e+05 1.49705e+05 3.79000e+02
 8.88000e+02 3.95000e+02 1.00000e+00]]
```

F MATRIX COMPUTATION

Print U, Σ and V^T ?

[y/n] : y

U = [[0.67769364 0.26800217 0.38273831 -0.26224422 0.3211915

0.10831822

-0.31457637 -0.19947534]

[0.26565745 -0.13190343 -0.0372827 0.79609234 0.3681896 0.31845242
0.16726001 0.10919319]

[0.1641376 0.28661087 -0.14897719 0.28113095 -0.37583067 -0.16140896
0.22274865 -0.75681288]

[0.09543902 0.169519 -0.11684656 0.38494506 -0.3153206 -0.33448984
-0.71950922 0.2670484]

[0.37196877 -0.65371867 -0.5154065 -0.19570528 -0.15443861 0.19148819
-0.20766923 -0.16340498]

[0.21922177 0.5569502 -0.45907424 -0.15655371 -0.22974928 0.42186939
0.19365055 0.37179433]

[0.02040961 0.16735829 -0.55432035 -0.08480461 0.61421702 -
0.52563539

0.04978119 -0.03284043]

[0.49726471 -0.18531976 0.18058237 -0.01569083 -0.25825974 -0.50792209
0.47185815 0.37174608]]

Σ = [2.00172927e+06 4.73230943e+05 1.19412560e+05 4.12089408e+04
2.01112118e+02 1.80951621e+02 1.34558155e+01 5.39976690e-01]

V^T = [[8.08283848e-01 4.00448310e-01 9.42676119e-04 3.72272145e-01

2.18471168e-01 4.62296849e-04 9.17826431e-04 4.76065119e-04

1.15489772e-06]

[-5.53012437e-01 4.05821182e-01 -3.04471434e-04 4.12308819e-01
5.99571278e-01 9.97371887e-04 -1.79958210e-04 1.00670121e-03

1.00901825e-06]

[-1.93198579e-01 6.62857023e-01 -2.04345313e-03 1.24637192e-01
-7.12554827e-01 -2.82335557e-03 -2.74509256e-03 -2.52618443e-03

-1.06235631e-05]

[-5.93873563e-02 -4.85296878e-01 4.82034384e-03 8.22072267e-01
-2.91624930e-01 3.68164181e-03 8.02631349e-03 3.31037653e-03
1.81312522e-05]

[1.36765125e-03 -6.11365077e-03 -8.41487054e-01 7.14454054e-03
8.30906778e-05 -6.94806712e-02 -5.34138690e-01 -4.09574558e-02
-1.49174379e-04]

[9.45552227e-05 -3.75303606e-03 9.69852714e-02 4.06834542e-03
2.22900072e-03 -7.11251393e-01 -6.79284693e-03 -6.96151021e-01
-2.70419271e-03]

[1.93975550e-04 -2.51335513e-03 5.30044887e-01 3.16743644e-03
-4.16775224e-04 7.04339404e-02 -8.44907156e-01 1.01967324e-02
-1.01410629e-02]

[-5.19372904e-07 2.18111625e-04 -3.88254479e-02 -1.32384825e-04
-2.37641423e-05 6.94715482e-01 2.56521030e-02 -7.15210460e-01
-6.06538185e-02]

[1.04613851e-06 -8.36660260e-06 3.16293352e-03 2.17731168e-05
-2.77278162e-06 4.09953386e-02 -7.12409281e-03 -4.52513760e-02
9.98103671e-01]]

F = [[3.23484509e-07 -8.24416313e-06 3.16293391e-03]

[2.18257454e-05 -2.78169850e-06 4.09953385e-02]

[-7.12409269e-03 -4.52513760e-02 9.98103671e-01]]

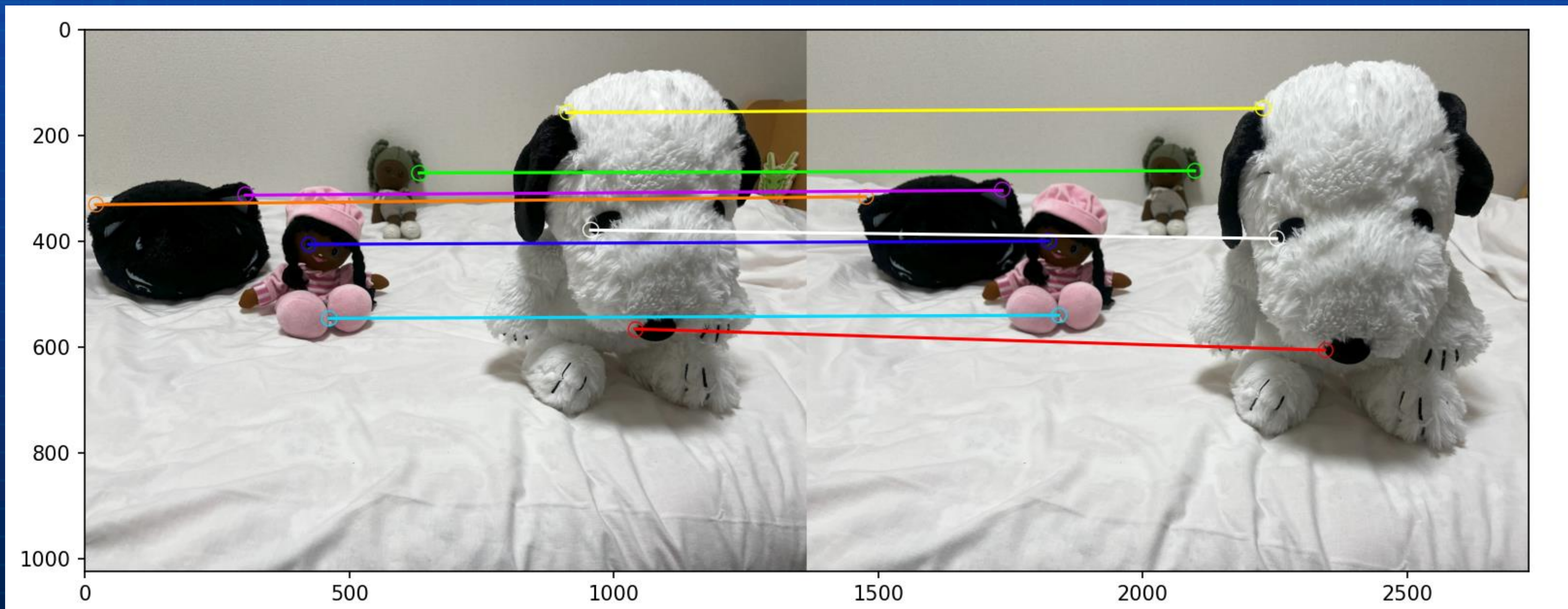
- - - - - Question A-3 : Verification of F - - - - -

The eigenvalues of F are : [9.96218920e-01 3.09784999e-19 1.88229224e-03]

The rank of F is : 2

CONNECTING LINES

Highlight of points used for computation and connecting lines



EPIPOLAR LINES

We can now also have the epipolar lines equations and evaluate the quality of the F matrix's computation.

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} f_{11}x' + f_{12}y' + f_{13} \\ f_{21}x' + f_{22}y' + f_{23} \\ f_{31}x' + f_{32}y' + f_{33} \end{bmatrix} = 0$$

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

$$ax + by + c = 0$$

--- Epipolar lines in image 1 and image 2 ---

Img 1 : Point 0 : -0.001516337612508625 x + 0.06067703396659119 y + -33.39871694369728 = 0

Img 2 : Point 0 : 0.005565703086858155 x + -0.0553997470404536 y + 27.490916548865112 = 0

Img 1 : Point 1 : 0.001198856499329558 x + 0.05625089639697005 y + -16.305973668256758 = 0

Img 2 : Point 1 : -0.0010048734789294455 x + -0.05121552742459669 y + 14.106814645321002 = 0

Img 1 : Point 2 : 1.3101078483232238e-05 x + 0.04985702477201735 y + -20.358157099206533 = 0

Img 2 : Point 2 : 0.0018739938844548115 x + -0.049868026628489 y + 18.98013216063067 = 0

Img 1 : Point 3 : 0.0007760741021622824 x + 0.048203402235311936 y + -15.387104843817452 = 0

Img 2 : Point 3 : -0.0001946185766530167 x + -0.048620029091974715 y + 14.78801360753563 = 0

Img 1 : Point 4 : 0.002213397250370055 x + 0.05939465797403172 y + -11.885319252977558 = 0

Img 2 : Point 4 : -0.0034027562737650573 x + -0.053198535308402295 y + 10.315804612969966 = 0

Img 1 : Point 5 : -0.0011349355540559956 x + 0.04988227614356312 y + -26.82870750467384 = 0

Img 2 : Point 5 : 0.004942214133005715 x + -0.050578986781168206 y + 24.842833978257868 = 0

Img 1 : Point 6 : 0.0005936851412241253 x + 0.042538979546156964 y + -14.092105443274905 = 0

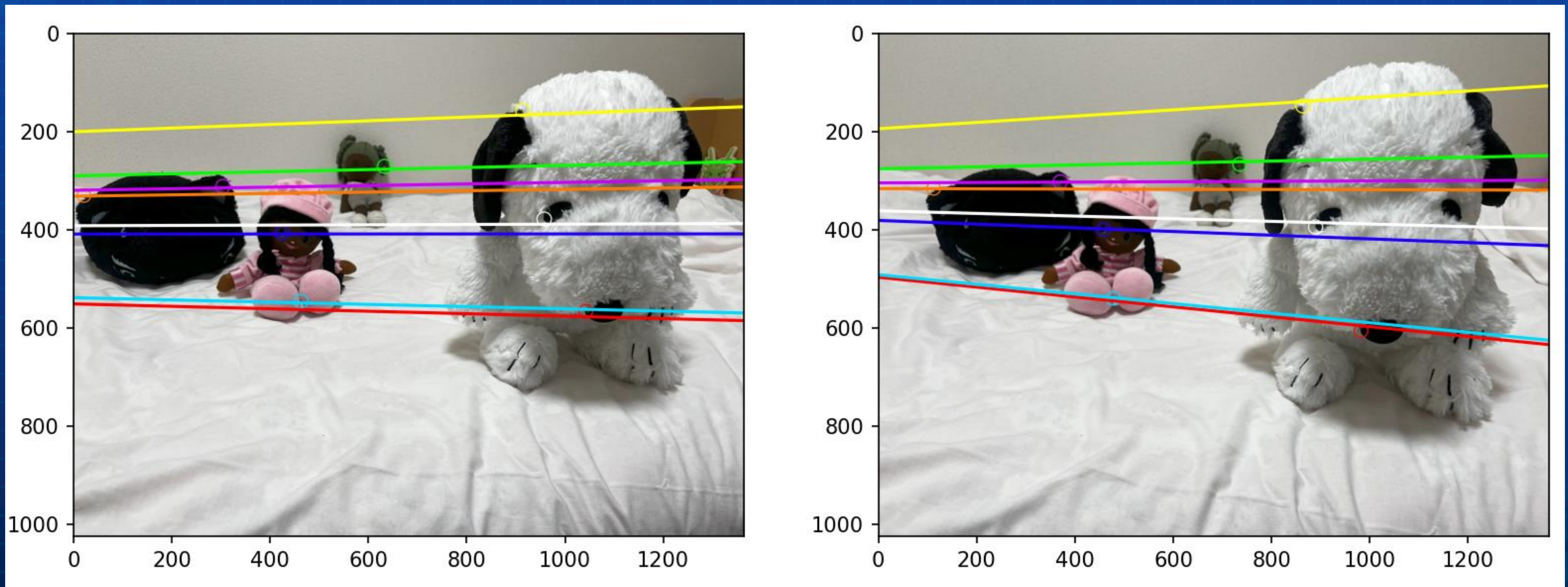
Img 2 : Point 6 : 0.0001066987240143603 x + -0.04633700149923368 y + 14.630819404719782 = 0

Img 1 : Point 7 : 0.0001937437176045111 x + 0.059277829521753825 y + -23.20238416660537 = 0

Img 2 : Point 7 : 0.0014571160027033484 x + -0.054187059717216615 y + 19.55910179406923 = 0

EPIPOLAR LINES

Here are the epipolar lines represented :



It is interesting because the results are more relevant with different planes than just one plane. Epipolar lines are closer to the points they correspond to. We can suppose that taking points on different planes allows us to have more information about how the geometry really is and how the camera renders it.

If we follow where the epipolar lines are converging for image 2 (right), we can see where the image 1 (left) was taken. It isn't really the case with image 1, where it looks more just like a translation of the camera.

NOT COMPUTED POINTS

- - - - - Question A-5 : Epipolar Lines of Nor Computed Points - - - - -

Would you like to take others feature points that are not included in the calculation of the F matrix ?

[y/n] : y

Number of points loaded : 4

[[1193, 590], [837, 584], [572, 2223], [103, 391]]

[[1202, 644], [805, 606], [672, 218], [179, 379]]

Img 1 : Point 0 : $-0.0017574787658980167 x + 0.06543847064235711 y + -36.70694190182368 = 0$

Img 2 : Point 0 : $0.006139014105800579 x + -0.05672786476346295 y + 28.958733561948517 = 0$

Img 1 : Point 1 : $-0.0015726239171116735 x + 0.05687935427110306 y + -32.159124816377144 = 0$

Img 2 : Point 1 : $0.005892899148262566 x + -0.05377625249812072 y + 27.586757058834362 = 0$

Img 1 : Point 2 : $0.0015830879376425655 x + 0.0550558291557234 y + -13.654086588872804 = 0$

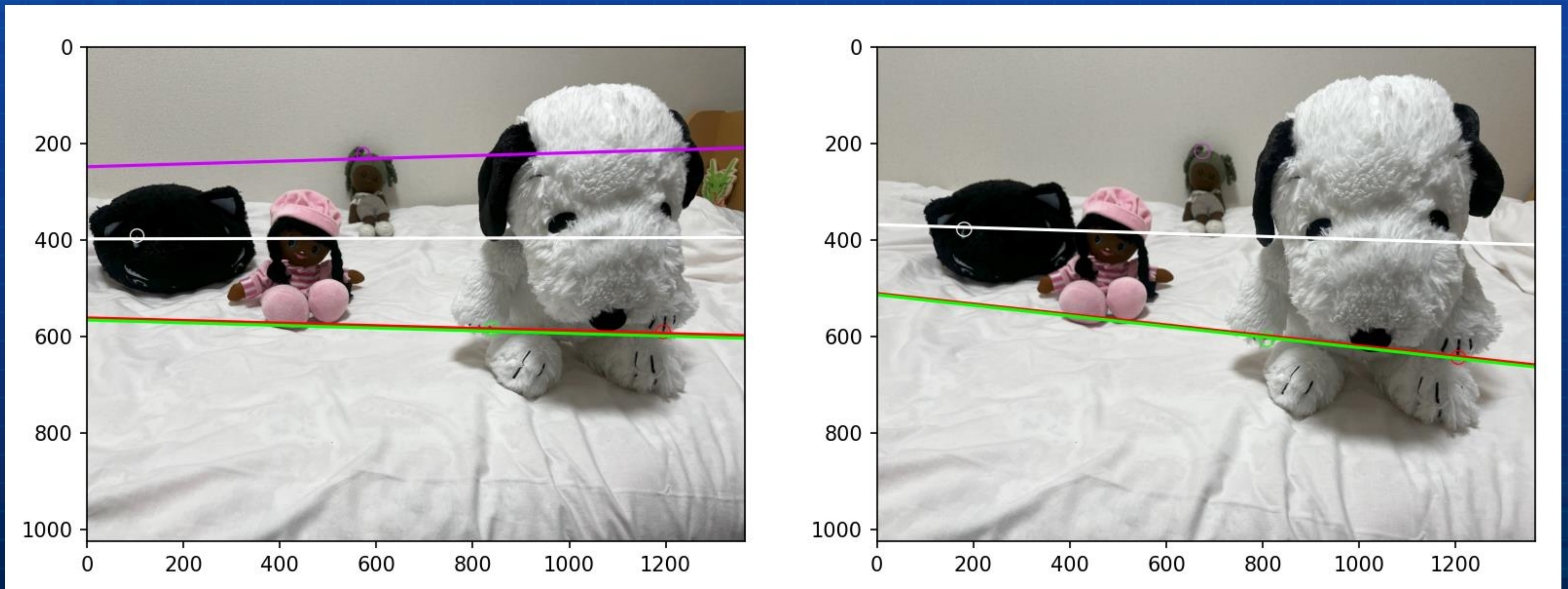
Img 2 : Point 2 : $0.04157957242518923 x + -0.05615075311722926 y + 93.93993943446068 = 0$

Img 1 : Point 3 : $9.629981065418071e-05 x + 0.04384788322597767 y + -17.4273804359029 = 0$

Img 2 : Point 3 : $0.001443092660861148 x + -0.04718816894929152 y + 17.35306323143909 = 0$

NOT COMPUTED POINTS

Now, we are showing the epipolar lines which corresponds to other points than the ones used for the fundamental matrix's computation.



The results seems relevant, similar to the last: epipolar lines are close to the points, and point of view is well rendered with image 2 but not perfectly with image 1. It just seems that purple epipolar line is an outlier considering image 2.

CAMERA PARAMETERS

Results of epipoles computation:

- - - - - Question B-7 : Intrinsic Camera Parameters - - - - -

Would you like to find intrinsic camera parameters with Kruppa equations ?

[y/n] : y

Left epipole: [-5.62717423e+03 4.09809435e+02 1.00000000e+00]

Right epipole: [-1.83859926e+03 3.11514400e+02 1.00000000e+00]

Left epipole (matrix form): [[0.00000000e+00 1.00000000e+00 4.09809435e+02]

[1.00000000e+00 0.00000000e+00 5.62717423e+03]

[-4.09809435e+02 -5.62717423e+03 0.00000000e+00]]

Right epipole (matrix form): [[0.00000000e+00 1.00000000e+00 3.11514400e+02]

[1.00000000e+00 0.00000000e+00 1.83859926e+03]

[-3.11514400e+02 -1.83859926e+03 0.00000000e+00]]

Epipoles have homogeneous coordinates, and can then be used to compute the intrinsic camera's parameters with Kruppa equations.

EXTRA POINTS OPENCV

8-points algorithm, as the name suggests, only use 8 points. But what about more points ? We can extract and match more feature points thanks to OpenCV algorithms, such as ORB or AKAZE for instance.

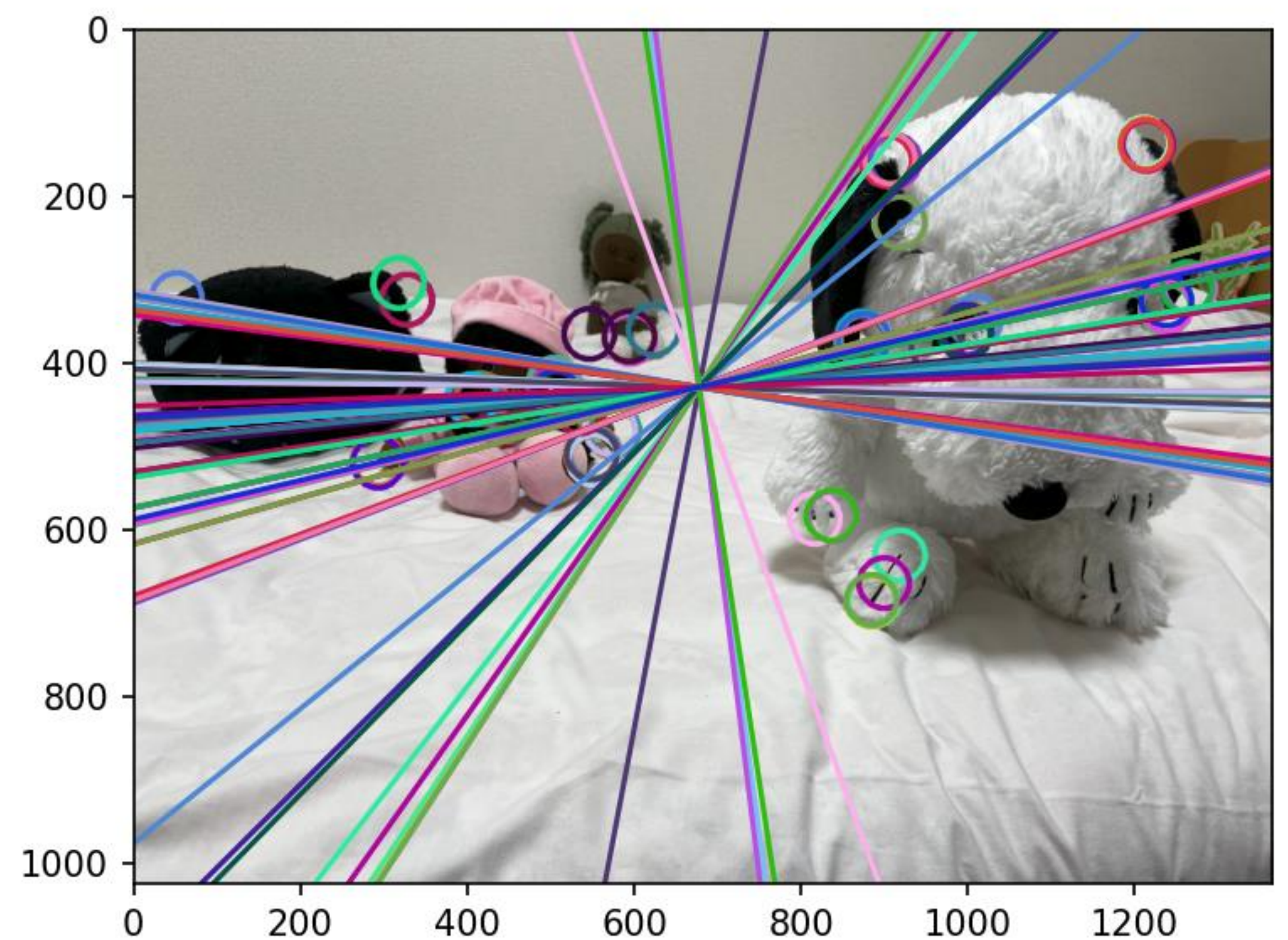
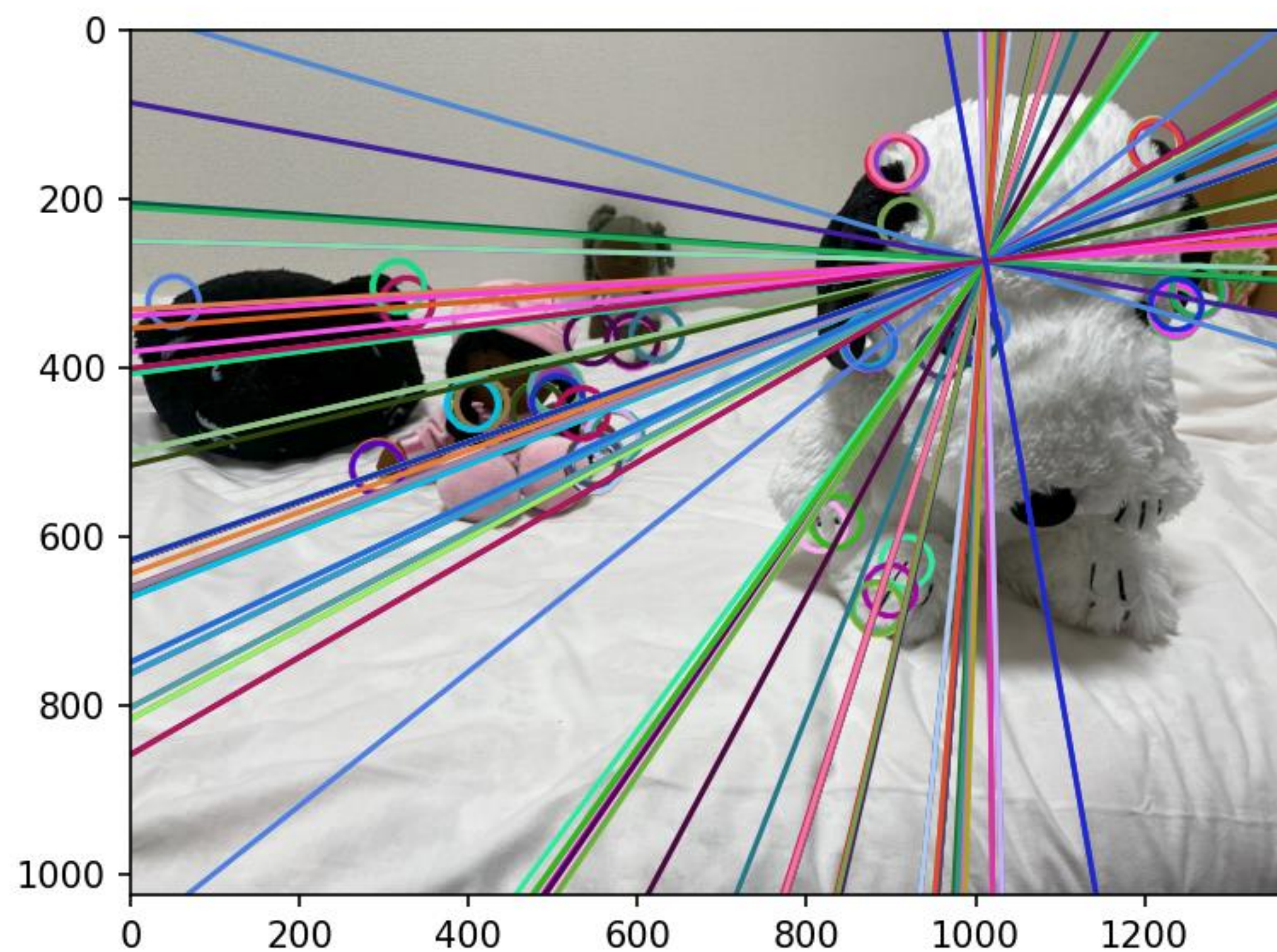
First of all, we have to find the feature points and corresponding matching points ; we give a max number of points, and then we use a threshold based on distance to keep more relevant points.

ALGO	PTS/THRESHOLD	REMAINING
ORB	200/60	107
AKAZE	200/60	28
AKAZE	300/60	50

We see that AKAZE is filtering more points ; as the C++-algorithm is sensibly fast, we can easily go up to 300 points, to keep 50 points at the end.

EXTRA POINTS OPENCV ORB

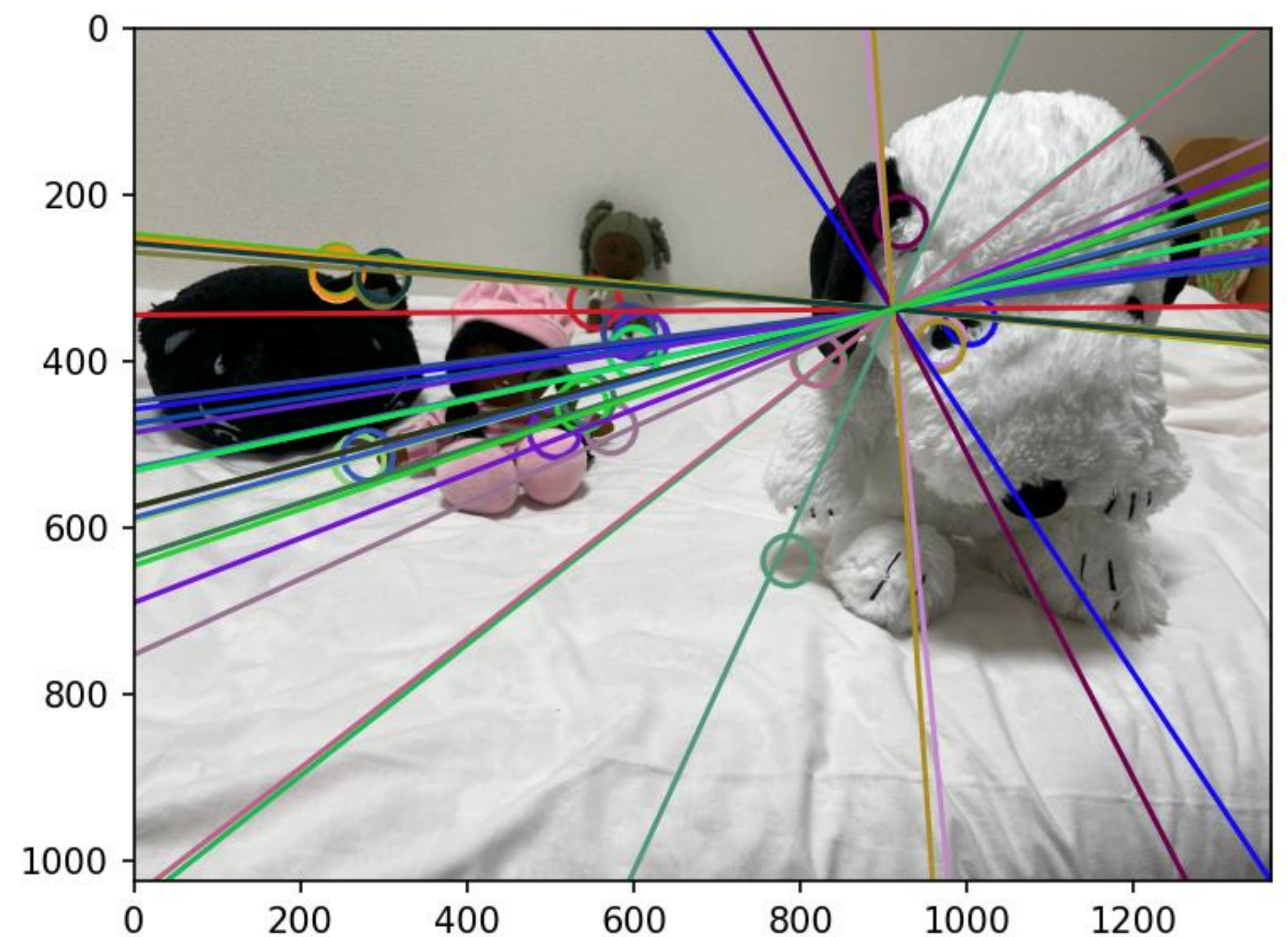
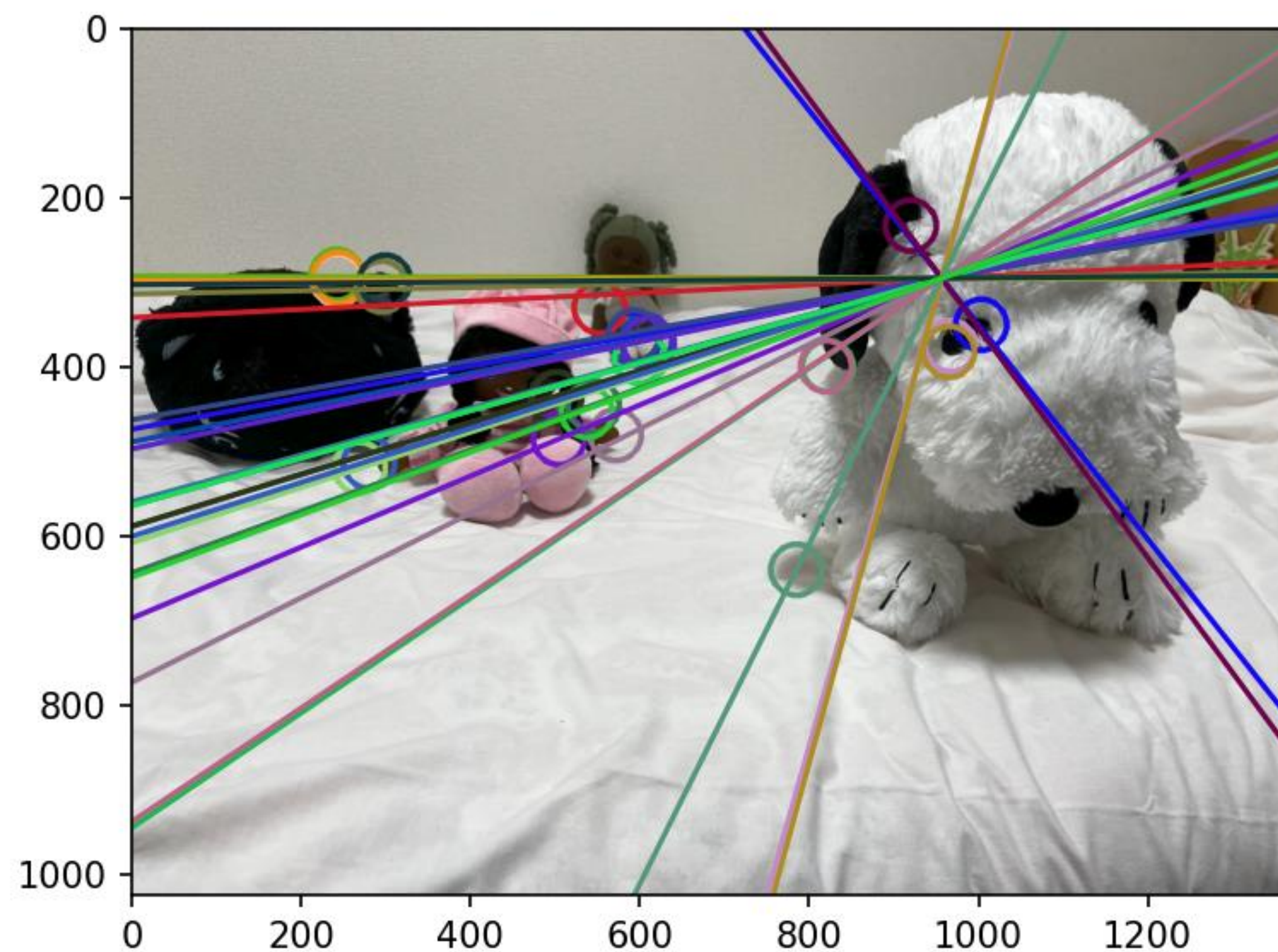
Here we see that ORB shows not so good results; we also have a lot of points remaining, and those gathered to the same areas. It appears also that image 2 on the right has better results : epipolar lines are closer to corresponding points... However we can't exploit so much those results.



Remaining points : 107

EXTRA POINTS OPENCV AKAZE

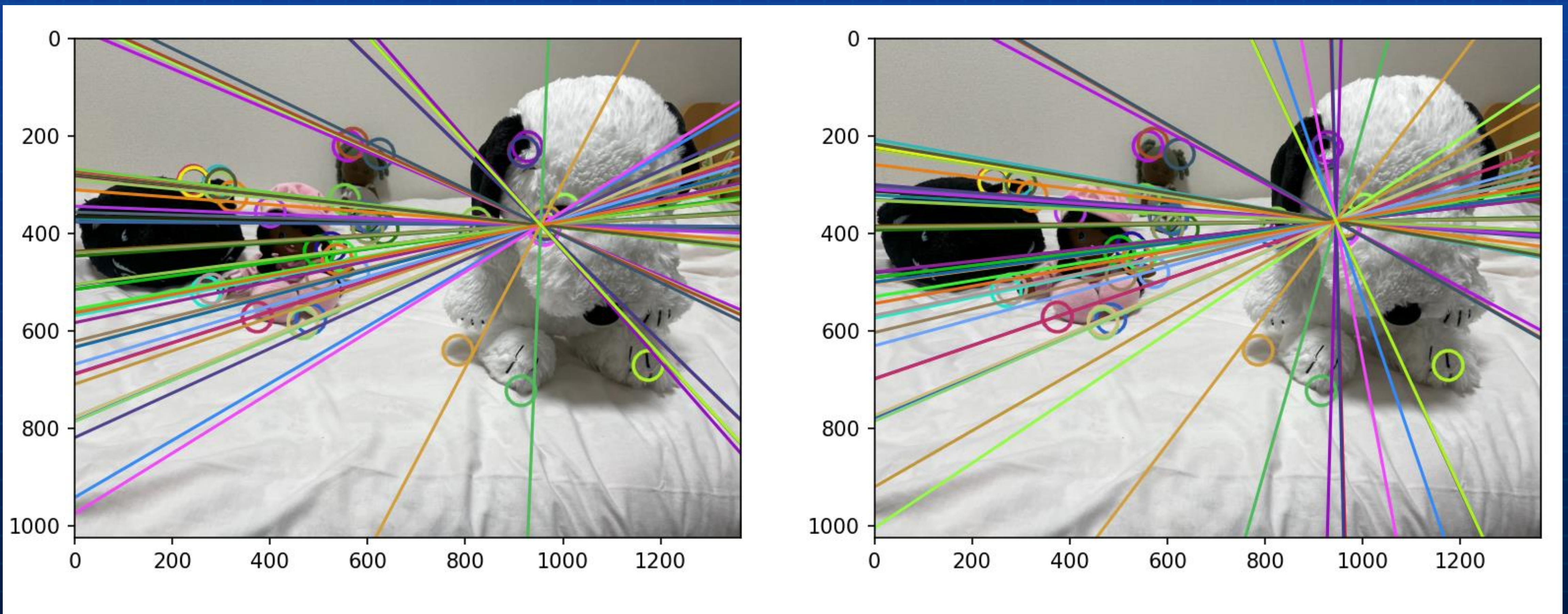
Now we try to do the same with AKAZE algorithm. The results are more satisfying, as we see that a lot of epipolar lines pass through corresponding points, or are closer to them.



Remaining points : 28

EXTRA POINTS OPENCV AKAZE

It is even more noticeable with 50 remaining points.



Remaining points : 50

RESULTS OF NOT SAME PLANE

- - - - Question A-6 : Compute f with more than 8 points - - - -

Would you like to take more than 8 points in the calculation of the F matrix ?

(Make sure you already run the C++ program for generate in the folder extra_points/ images and associated .txt)

[y/n] : y

Number of points loaded : 50

[[559, 222], [294, 293], [961, 393], [600, 391], [501, 437], [599, 390], [918, 233], [607, 370], [593, 367], [524, 430], [483, 576], [243, 298], [242, 292], [467, 586], [269, 520], [552, 331], [244, 296], [592, 364], [373, 572], [504, 486], [966, 376], [...]] * just a sample of the lines shown during execution

[913, 723], [500, 437], [1002, 351], [926, 223], [374, 572], [840, 395], [467, 580],

[631, 392], [919, 234], [280, 514], [964, 383], [821, 400], [569, 216], [610, 373], [819, 400], [400, 357], [295, 306], [600, 391], [623, 238], [821, 379], [322, 325],

[571, 481], [960, 395], [1173, 672], [299, 299], [784, 640], [545, 447], [537, 454], [1173, 671]]

[[662, 216], [365, 285], [891, 407], [682, 389], [536, 433], [681, 388], [848, 233], [693, 367], [677, 365], [556, 426], [498, 570], [325, 287], [322, 280], [481,

578], [311, 504], [645, 327], [322, 284], [676, 362], [389, 561], [529, 484], [897, 392], [862, 759], [535, 433], [929, 365], [860, 223], [391, 561], [803, 406], [482,

573], [715, 389], [849, 235], [321, 496], [896, 398], [795, 416], [670, 211], [694, 369], [792, 417], [116, 323], [365, 298], [682, 389], [725, 234], [790, 395], [393,

317], [610, 480], [889, 410], [1160, 734], [368, 291], [762, 662], [578, 444], [569, 449], [1160, 733]]

F = [[1.35979797e-07 -9.33261975e-06 3.36785794e-03]

[1.08724177e-05 1.97366601e-06 -1.10193553e-02]

[-4.28830138e-03 8.15362187e-03 9.99891174e-01]]

Would you want to print epipolar lines equations ?

[y/n] : y

--- Epipolar lines in image 1 and image 2 ---

Img 1 : Point 0 : 0.0014420306972398754 x + -0.003395502921132364 y +

-0.07778201527181139 = 0

Img 2 : Point 0 : -0.0017986119543639896 x + 0.0033748412835758407 y + 0.4362268925304733 = 0

Img 1 : Point 1 : 0.000757693934654329 x + -0.006488428009985994 y + 1.7584434037518526 = 0

Img 2 : Point 1 : -0.001062704947138908 x + 0.005988115804190496 y + -1.2386296848742606 = 0

Img 1 : Point 46 : -0.0027067197323266306 x + -0.001428006117167287 y + 3.129903201414044 = 0

Img 2 : Point 46 : 0.002776654079355424 x + 0.0020999942298381667 y + -3.4120955723387967 = 0

Img 1 : Point 47 : -0.0006972289091034135 x + -0.0038587901547624606 y + 2.1414610873618356 = 0

Img 2 : Point 47 : 0.0006457783006366324 x + 0.003949572811392009 y + -2.090278053304371 = 0

Img 1 : Point 48 : -0.0007451158260352638 x + -0.003946773583620108 y + 2.220823909140293 = 0

Img 2 : Point 48 : 0.0007207973858387087 x + 0.004038049431446001 y + -2.194356403665653 = 0

Img 1 : Point 49 : -0.003315215775482918 x + 0.0030393463955483025 y + 2.002066405012565 = 0

Img 2 : Point 49 : 0.0031665951676735716 x + -0.001469211207404588 y + -2.443598847689767 = 0

AKAZE / ORB

What are the main differences between AKAZE and ORB? How can we explain those results?

In fact, ORB is using FAST method for the keypoints, and is an efficient algorithm but it can miss details, and not render the depth correctly.

AKAZE is more specialized regarding different scales and points of view, which are convenient in our case. AKAZE's feature descriptors are more relevant while dealing with the change of planes, angles...

ORB is using binary descriptors, which lead to fast computation, but AKAZE also uses gradient image to be more precise.

Despite ORB and AKAZE both use ratio test, and ORB being especially fast, AKAZE is still more relevant in our case of study.

CONCLUSION

Thanks to only two images and a camera, we have been able to compute the fundamental matrix and recover the epipolar geometry of the scene. We can use various methods such as 8-point algorithm, get help to generate more matching points with the algorithms of OpenCV...

The results generally fit the reality, however we still have sometimes many gaps and part of the images are working better than others. We saw for example that the images with target object not on the same plane allow us to compute a more accurate F matrix.

We can explain the occasional poor results with many factors:

- The camera used has a lot of post processing and meaningful distortion, as an iPhone mobile camera
- The 8-point algorithm is sensitive to noise
- We didn't normalize the coordinates

What we can improve next time: a more controlled movement between the two images, more suitable images with less noise, normalization...

However even with homemade pictures we have been able to analyze the epipolar geometry and work on stereo basics.