CV - EX 3

Amitai ovadia - 312244254

May 7, 2023

Q1

(a) The problem that this paper tackels is that of calculating the fundamental matrix F
between views of the same scene with fewer then 7 point correspondences.
Up untill now it was thought that that at least 7 points are needed to compute F,
but it becomes harder and harder to find 7 point correspondences using Automatic methods if the angle of view between the 2 cameras increases.
The article suggests a method to find F with less then 7 point correspondences using more available image imformation.
(b) The observation upon which the paper is based is that every two pencils of epipolar lines are related by an homography (there exsists such an homography).
We know that every 4 corresponding epipolar lines have the same cross ratio:
(1) $\frac{|l_1 l_2||l_3 l_4|}{|l_1 l_3||l_2 l_4|} = \frac{|l'_1 l'_2||l'_3 l'_4|}{|l'_1 l'_3||l'_2 l'_4|}$: $l_s = p_s \wedge e \iff l'_s = p'_s \wedge e'$.
From the above observation we can derive the second epipole e', the conic equation that the seconds epipole lies on and more.

Q2

Here I will elaborate on how the epipole localization actually happeneds for $N = \{4, 5, 6\}$ points correspondences.
Using the formula (1) We can substitute for $l_s = p_s \wedge e$ and reach the following formula:
$\frac{|l_1 l_2||l_3 l_4|}{|l_1 l_3||l_2 l_4|} = \frac{|l'_1 l'_2||l'_3 l'_4|}{|l'_1 l'_3||l'_2 l'_4|} \Rightarrow$
(2) $\frac{|e p_1 p_2||e p_3 p_4|}{|e p_1 p_3||e p_2 p_4|} = \frac{|e' p'_1 p'_2||e' p'_3 p'_4|}{|e' p'_1 p'_3||e' p'_2 p'_4|}$
were each variable is a vector [x, y, 1], point in 2D in homogenous coordinates.
e and e' are the epipoles.
all the pairs p_i and p'_i are points corrspondence.
And the notation of $|abc| = det\left(\begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{bmatrix}\right)$
$e' = \begin{pmatrix} e'_x & e'_y & e'_z \end{pmatrix}$ the only unknown.
We can derive from (2) the conic equation $a{e'_x}^2 + b e'_x e'_y + c{e'_y}^2 + d e'_x e'_z + e e'_y e'_z + f{e'_z}^2 = 0$
$N = 4$:
In this case we cannot, according to the paper, really find the epipole.

But we can find the conic on which the epipole lies on in the second image:
From equation (2) given e and the points correspondences, and assigning $e' = \begin{pmatrix} e'_x & e'_y & e'_z \end{pmatrix}$
we will get the conic equation that e' sits on (in the second image)
in the form of $e'^T C e' = 0$.
$N = 5$:
We can take 2 subsets of size 4 from the 5 points correspondences. compute the 2 conics C_1 and C_2 from each subset using equation (2),
and since e' lies on both of them then we will calculate their intersection.
(there are maximum 4 intersection points between 2 conics, from which 3 are already known to us).
$N = 6$:
Given 6 corresponding points and the line of one of the epipoles,
We can use three different subsets of size 4 from the points to derive three equations of the form (2)
Given those equations we can derive both epipoles.

Q3

I used 2 images.

Image 1 and the chosen 5 points $p_1, ..., p_5, e$:

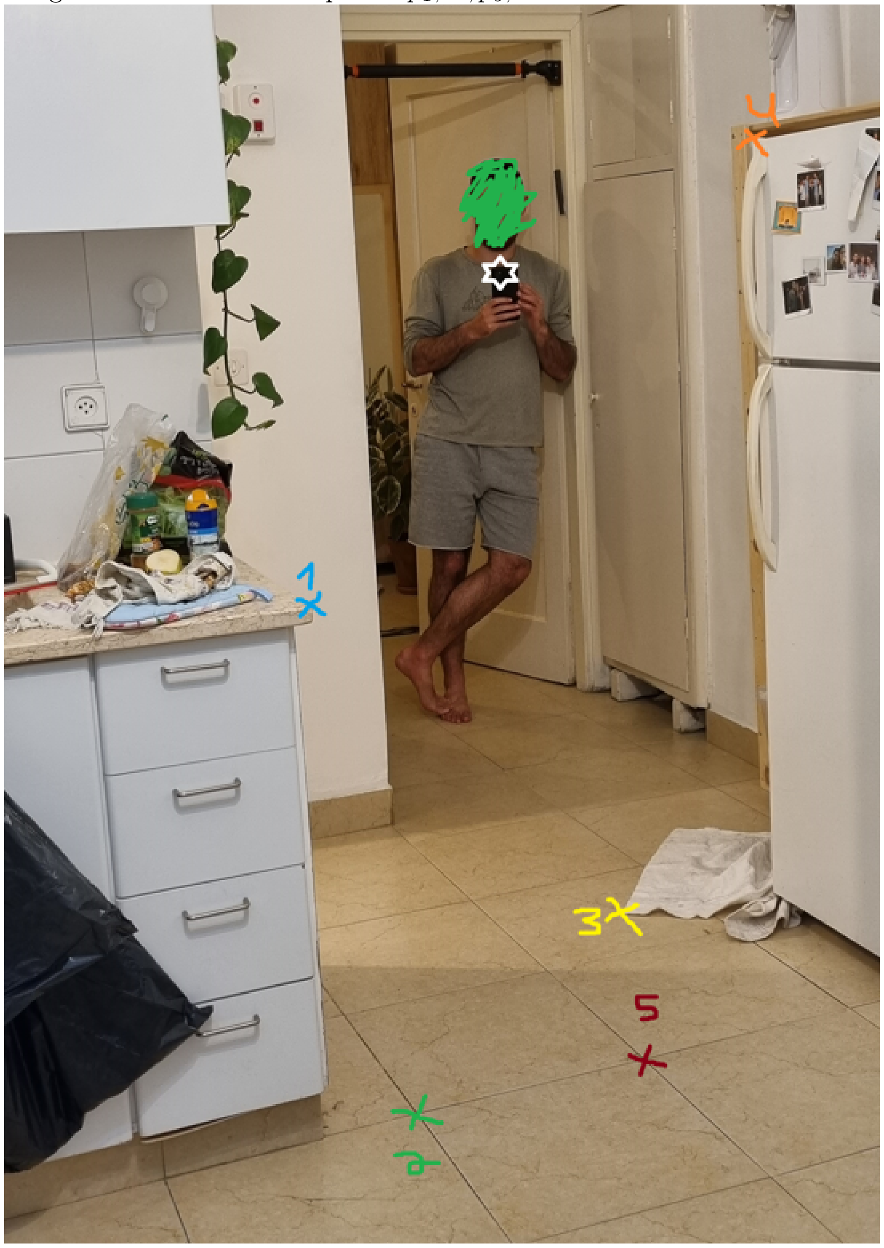
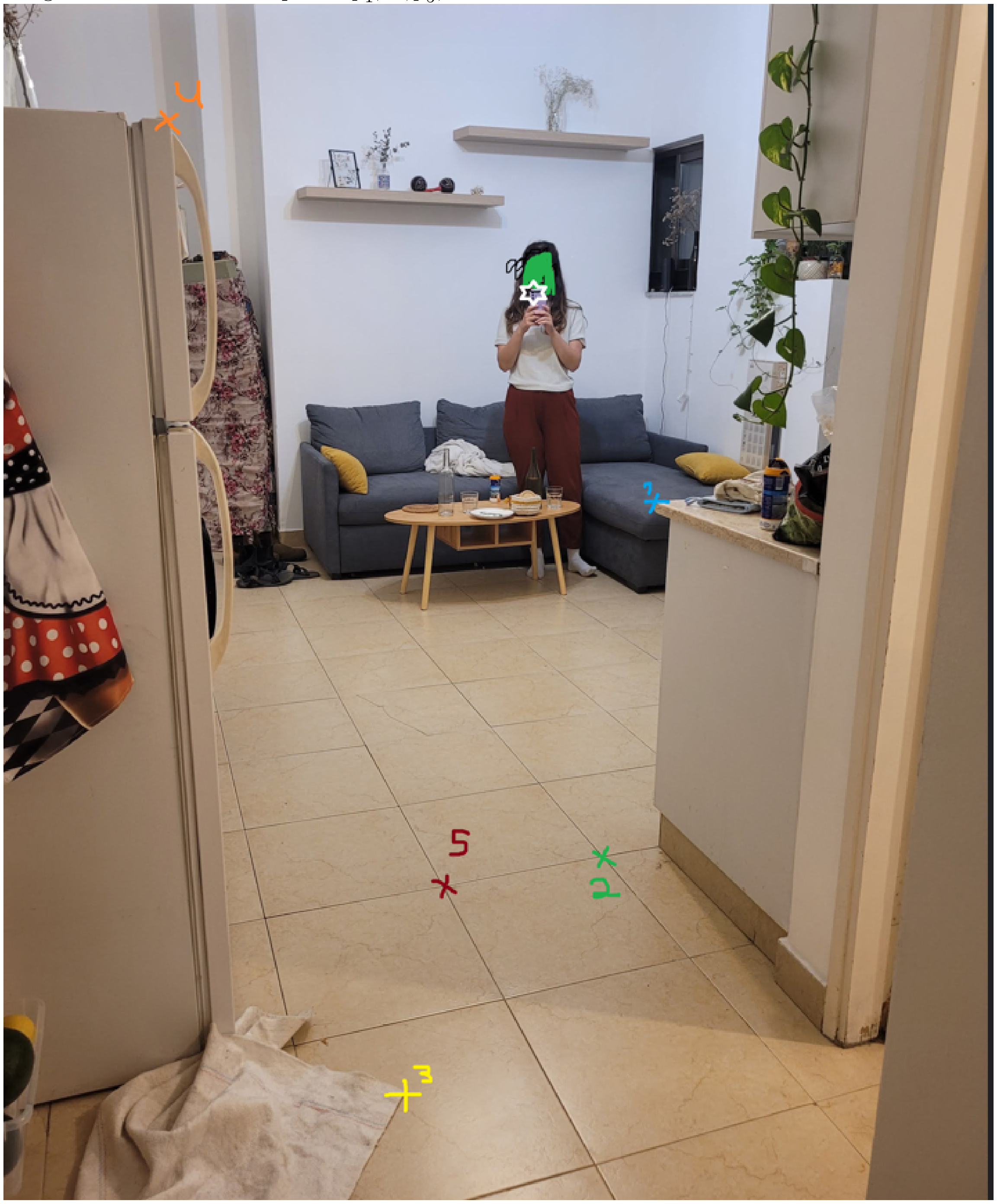
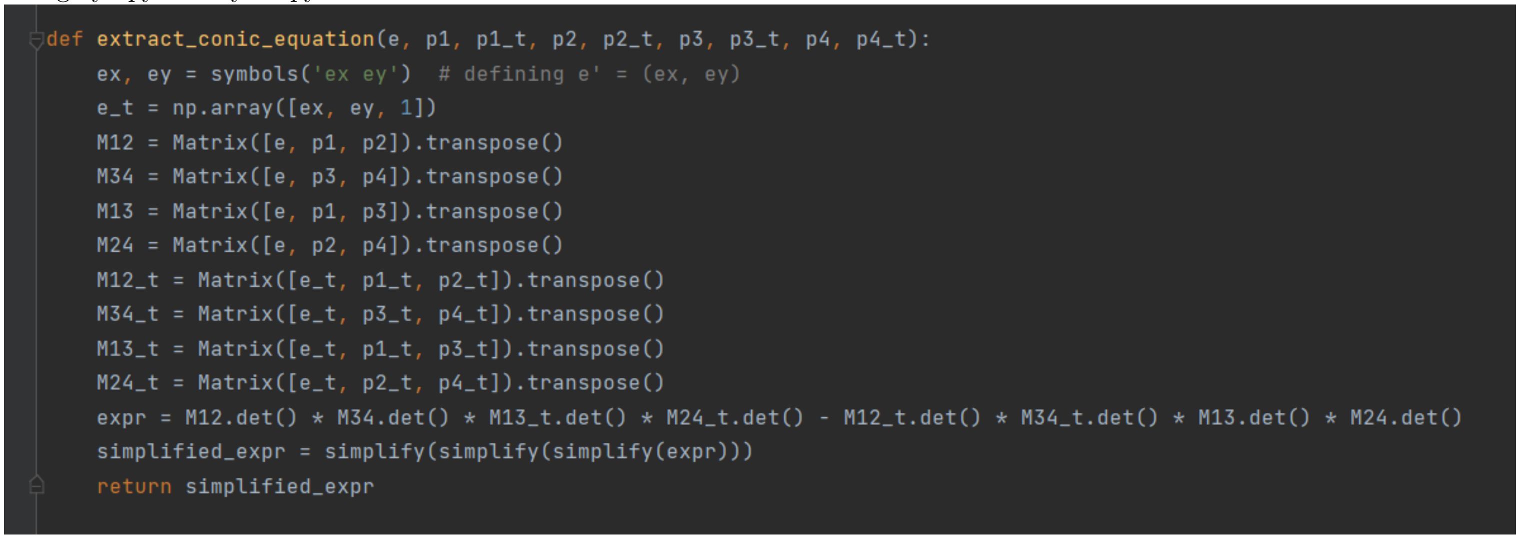


Image 2 and the chosen 5 points $p'_1, ..., p'_5, e'$:



Next I wanted to find the epipole e' in image 2 using 5 points correspondences and the other epipole e in the image 1.
As was displayed before, We can take 2 subsets of size 4 from the 5 points correspondences.
compute the 2 conics C_1 and C_2 from each pair, and since e' lies on both of them then
we will calculate their intersection. (there are 4 intersection points, from which 3 are already known to us).
For deriving a Conic equation I simplified this expression: (2) $\frac{|ep_1p_2||ep_3p_4|}{|ep_1p_3||ep_2p_4|} = \frac{|e'p'_1p'_2||e'p'_3p'_4|}{|e'p'_1p'_3||e'p'_2p'_4|}$
Using sympy library in python:



```
def extract_conic_equation(e, p1, p1_t, p2, p2_t, p3, p3_t, p4, p4_t):
    ex, ey = symbols('ex ey')  # defining e' = (ex, ey)
    e_t = np.array([ex, ey, 1])
    M12 = Matrix([e, p1, p2]).transpose()
    M34 = Matrix([e, p3, p4]).transpose()
    M13 = Matrix([e, p1, p3]).transpose()
    M24 = Matrix([e, p2, p4]).transpose()
    M12_t = Matrix([e_t, p1_t, p2_t]).transpose()
    M34_t = Matrix([e_t, p3_t, p4_t]).transpose()
    M13_t = Matrix([e_t, p1_t, p3_t]).transpose()
    M24_t = Matrix([e_t, p2_t, p4_t]).transpose()
    expr = M12.det() * M34.det() * M13_t.det() * M24_t.det() - M12_t.det() * M34_t.det() * M13.det() * M24.det()
    simplified_expr = simplify(simplify(simplify(expr)))
    return simplified_expr
```

I got this expressions for the 2 conics:
For the first conic:
66051945006480660*ex**2 + 336532119013873O2*ex*ey - 178114233005131948O2*ex
+ 116495460745496226*ey**2 - 418562018934895221600*ey + 159380879831564658173214
And for the second conic:
-151522235553106704*ex**2 + 107314687356800088*ex*ey + 106033433046068744568*ex
- 111572529633243360*ey**2 + 209690470146049894320*ey - 111747240006289762625520

Then I looped through all the image pixels that when plugged into this expression were under a certain threshold (10^{21}) .
I found all the intersection clusters of points of the 2 conics (4 intersections, using kmeans), eliminated the 3 known points (in blue Xs),
and I was left with the epipole (in red X).