Computer Vision and Pattern Recognition

Giacomo Boracchi, due date: April 14, 2020

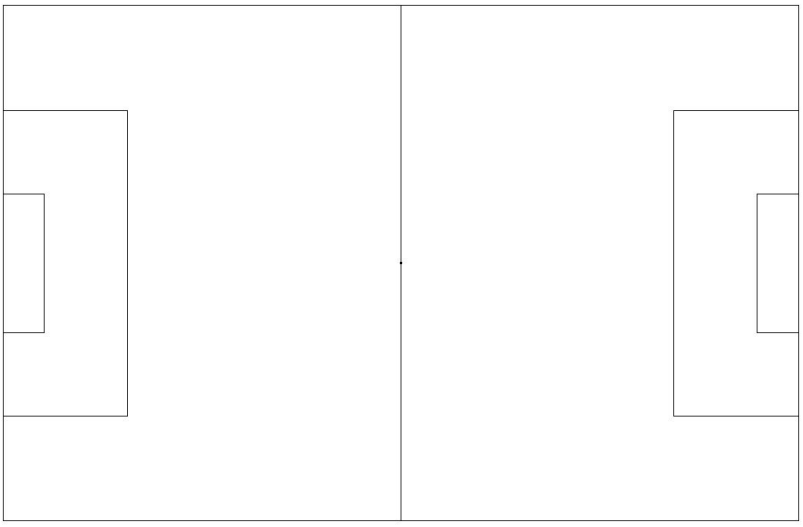
General Rules

- This assignment is **due by April 14, 23.59 CET** to be uploaded on the icorsi platform.
- You are allowed to prepare your homework in **teams of two persons**, where both members are required to equally contribute to the solution of the whole assignment.
- The name of both members must be clearly written in your submission. Nevertheless, each student should submit their copy by their own (i.e., **double submission** mode: each student should upload their assignment) so that it will be easier for us to grade them.
- Upload a well commented Python script or (better) a **Jupyter Notebook**. You can also use Matlab if you prefer.
- Try to comment your design choice.

Single View Geometry A: (10 pts)

Consider the following image ***I*** of a football pitch (Stadio Meazza, Milano).

Consider also the corresponding pitch layout ***S***, which contains a few lines placed in the correct position (1 pixel = 1cm) for the Meazza Stadium.



A scheme of the pitch layout ***S*** is also provided, or it can be re-computed from the official measures of the pitch from following resources https://en.wikipedia.org/wiki/Football_pitch https://it.wikipedia.org/wiki/Stadio_Giuseppe_Meazza

You are asked to:

- Estimate the transformation that maps points from the pitch layout ***S*** to this image ***I***
- Apply this transformation to map all the line segments of the pitch layout ***S*** over the image ***I***, visualizing all the line segments even in regions that are not depicted in the image ***I***.
- Define the centre circle ***C*** in the layout and map it to the image (mapping the conic is more convenient for answering then the bonus question)
- Look for another picture ***J*** of Stadio Meazza (possibly partially occluded), and repeat the whole process. Briefly comment the results (e.g. whether it works better or does not work as for the provided image)

The expected outcomes should look like the following. Mapping mode details of the pitch is not required, but the image might look nicer.

For fitting the transformation you are free to manually select as many point correspondences as you want, however, avoid built-in transformation and use codes developed during labs/lectures.

Watch out that you have to possibly compensate for changes in the coordinate systems in ***I*** and ***S***

Single View Geometry B: (3 pts)

We assumed you have been estimating the above mapping by manually selecting point correspondences. However, this might not always be the case since corners in this sort of image are very ambiguous and everything could be more robustly performed from lines.

Repeat the transformation fitting process by providing line correspondences as input instead of point correspondences. You are not allowed to intersect lines and use the point-estimation procedure: you need to implement the homography directly from the lines equations!

Bonus Question: dual conic to circular points (4 pts)

Rectify (bringing lines over the pitch to be orthogonal) the above image without making use of point or line correspondences nor the pitch scheme ***S***.
You know however which lines are parallel and you can use the equation of the centre circle ***C*** (which should be an ellipse in these images), as long as you have been mapping it from the pitch scheme ***S*** in the above question.

Single View Geometry C: (5 pts + 1 bonus)

You are given two pictures ***I1*** and ***I2*** of the opposite sides of the Stadio Meazza, Milano.

You need to reconstruct the full pitch by stitching the two pictures. The whole procedure is simplified if you jointly map the two images to the scheme ***S*** **.** Here is a short guideline to solve this problem

1. Map the two images ***I1*** and ***I2*** over the provided pitch scheme ***S***. To estimate this transformation you can select as many point correspondences as you need, possibly adding the centre circle to the scheme ***S.***

2. Stitch the two transformed images by selecting in each pixel of the output image the color form ***I1*** or ***I2*** .

NB there are different ways for merging the two images. Here I have selected in each pixel of the output image the color having the maximum intensity between the corresponding pixel in ***I1*** and ***I2*** . but you can use more sophisticated ways to obtain better results (and gain a bonus point!).

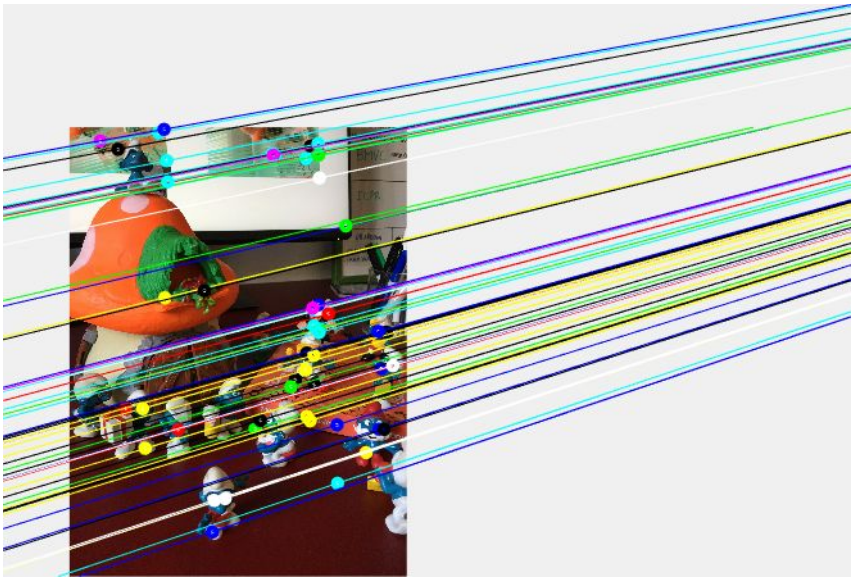
Multi View Geometry: (7 pts)

Consider the following pair of images (**imgV1** left, **imgV2** right)

Which are provided together with sparse correspondences in the array **X12** (each column of **X12** stacks a pair of corresponding points: the first three rows refer to coordinates in the first image, the last three rows refer to coordinates in the second image).

Part1, You are asked to:

- Estimate the fundamental matrix mapping **imgV1** in **imgV2**
- In both images, for each match provided, plot epipolar lines passing through each matching point. Compute the epipoles and display their location on the image plane.



Part2

You will also be provided with:

- Two low-resolution copies of these (**imgS1**, **imgS2**) and of the corresponding matches (**X12s**)
- Two fundamental matrices F13 and F23 that map **imgS1** and **imgS2**, respectively, to a third view where to perform epipolar transfer.
- The grayscale image of the third view (**imgS3gray**)

You are asked to:

- Perform epipolar transfer between **imgS1** and **imgS2** to **imgS3gray** by manually selecting correspondences over the hats of each smurf / puppet. Provide a visualization similar to the one shown in the rightmost image below over **imgS3gray**

- Add colors to **imgS3gray** by means of epipolar transfer of the matches **X12s**. Select a neighbor size (e.g. 3x3) where to paste in **imgS3gray** the colors of the pixels from **imgS1**.

- Briefly comment these results and whether there are errors

NB to solve this part of the assignment you need to load a matlab workspace. To do so, you need to load it in Python. This is done by simply the function **loadmap** which can be imported from **scipy.io**

>from scipy.io import loadmat

then you will simply need to load the workspace using it

>my_workspace = loadmap('epipolarWrks.mat')