Calibration of the video clip

The first step is to calibrate the video clip so that we have a bird view of the pool. To do that, we compute an **homography** for one image. We assume that the camera does not move and that there is no distortion. Thus, we can use the **homography** to calibrate the entire video clip.

To compute the **homography**, we need the coordinates of four points in the original image (we call this list **src**) and the coordinate of these four points in the calibrated image (we call this list **dst**).

To have notions of distances in the calibrated image, we need to give the coordinate of the top left corner and the bottom right corner of the calibrated image (we call this list **extreme values**). Pages 2 and 3 shows a way to compute **scr**, **dst,** the **homography** matrix and the **extreme values**.

The calibrated images have the same size as the original images.

This is the format of the .txt file :

	Type	Unit	Number of elements
Name of the video clip	string		1
scr	int	pixel	8
dst	int	pixel	8
Homography matrix	float		9
Extreme values	float	meter	4

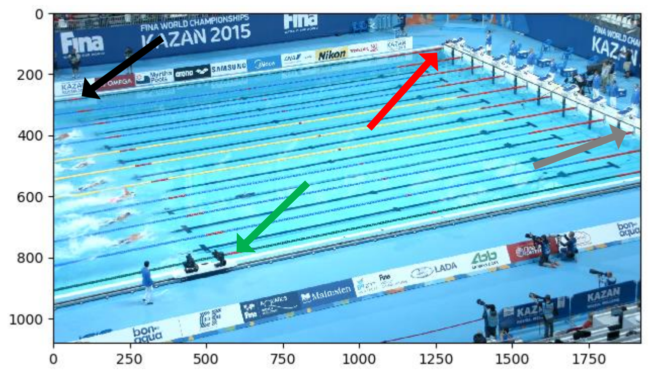
Given by the user :

The coordinates in pixel of the four points in the original image:

src = [[66. 284.], [1268. 118.], [593. 786.], [1888. 391.]]

The coordinates in meters of the four points :

[[25. 0.], [50. 0.], [25. 25.], [50. 20.]]



Computed by the computer :

The coordinates in pixel of the four points in an image that would show the full pool :

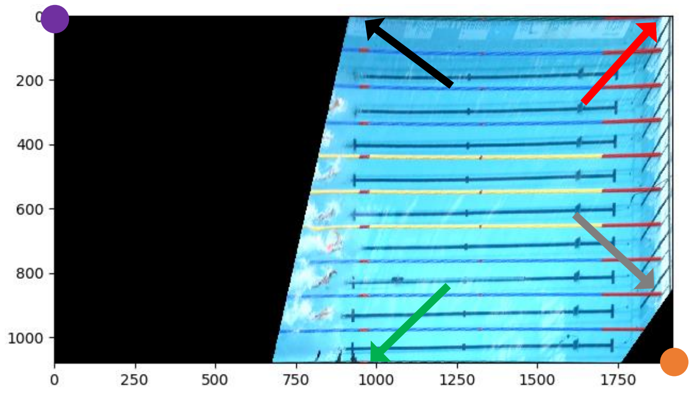
[[960. 0.], [1883. 0.], [960. 1080.], [1883. 864.]]

1883 = (50 + 1) x width / 52 # +1 to be sure to have all the pool, hence 52 instead of 50

1080 = 25 x height / 25

The coordinates in meters of the top left point and the bottom right point:

[[-1. 0.], [51. 25.]] *# Caution ! The first point on the left starts at the meter -1.*

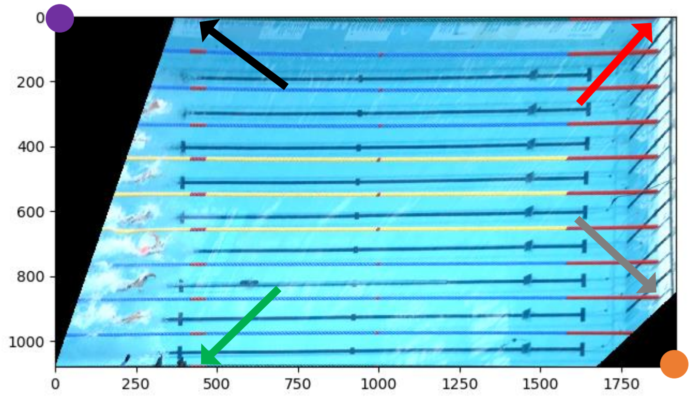


The coordinates in pixel of the four points in the final image :

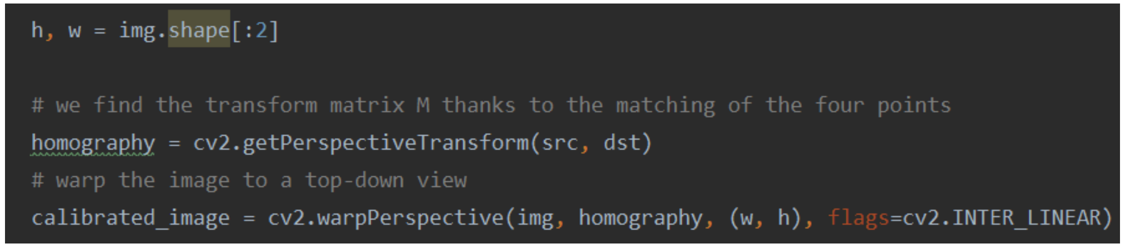
dst = [[439. 0.], [1865. 0.], [439. 1080.], [1865. 864.]]

The coordinates in meters of the top left point and the bottom right point:

Extreme values = [[17.31 0.], [50.97 25.]]



To compute the **homography**, we use this code :



```python
h, w = img.shape[:2]

# we find the transform matrix M thanks to the matching of the four points
homography = cv2.getPerspectiveTransform(src, dst)
# warp the image to a top-down view
calibrated_image = cv2.warpPerspective(img, homography, (w, h), flags=cv2.INTER_LINEAR)
```