Perception module for robotics applications with OpenCV.

Stefano Leonardi
Enrico Saccon

University of Trento

Steps

- Calibration
- Unwrapping
- Detection

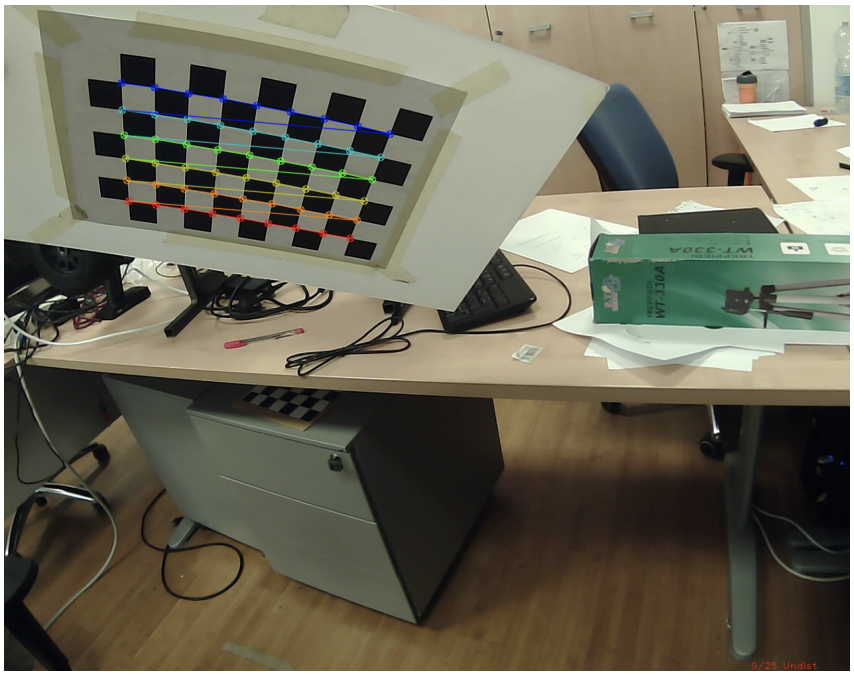
Calibration

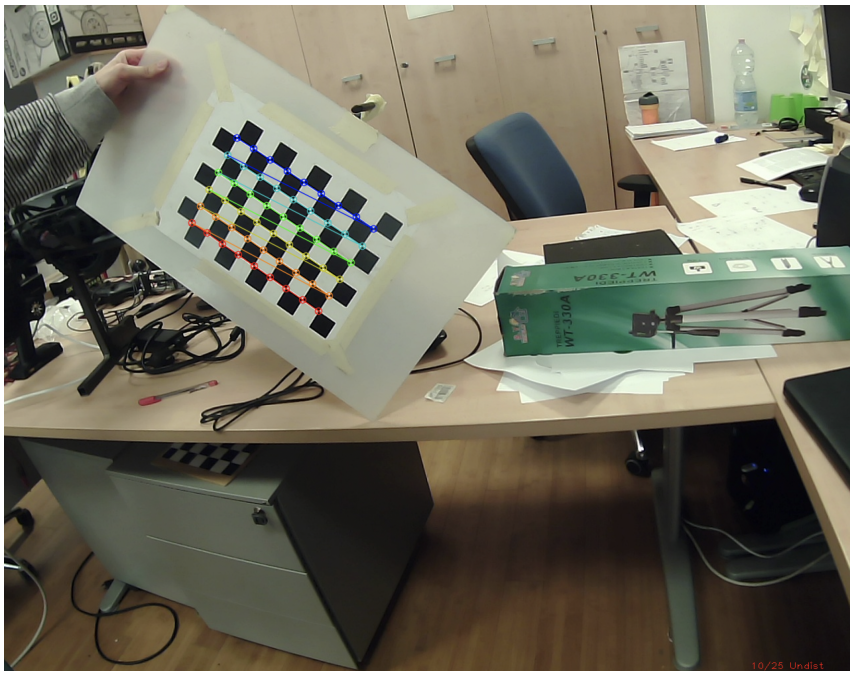
Given a set of pictures taken with a camera, the object of this step is to find the camera matrix A and the distortion coefficients $\dot{d}$.

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

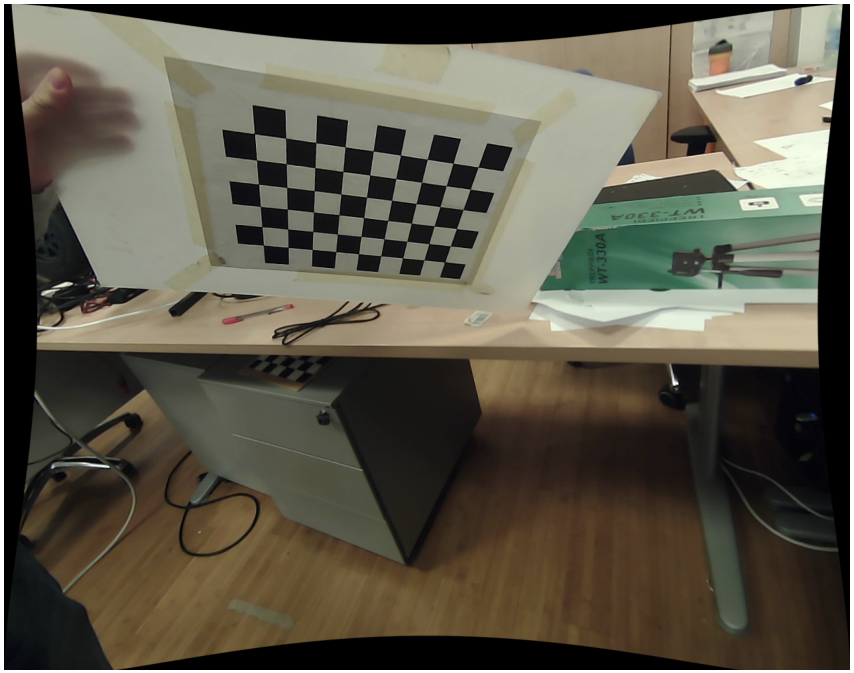
$$\begin{aligned} x_{distorted} &= x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ y_{distorted} &= y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ x_{distorted} &= x + \left[2p_1 xy + p_2\left(r^2 + 2x^2\right)\right] \\ y_{distorted} &= y + \left[p_1\left(r^2 + 2y^2\right) + 2p_2 xy\right] \\ \dot{d} &= (k_1, k_2, p_1, p_2, k_3) \end{aligned}$$

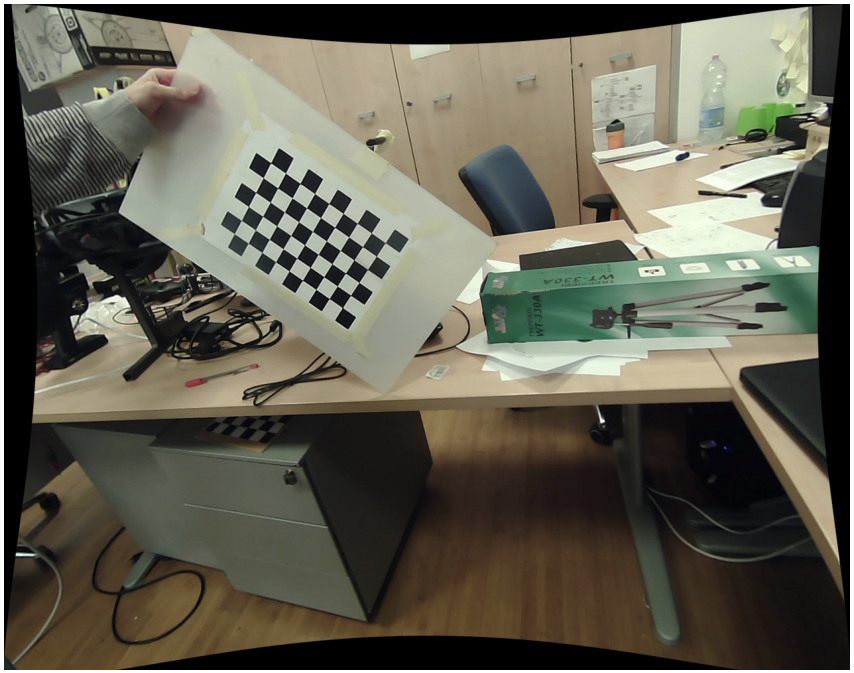
Calibration





Calibration





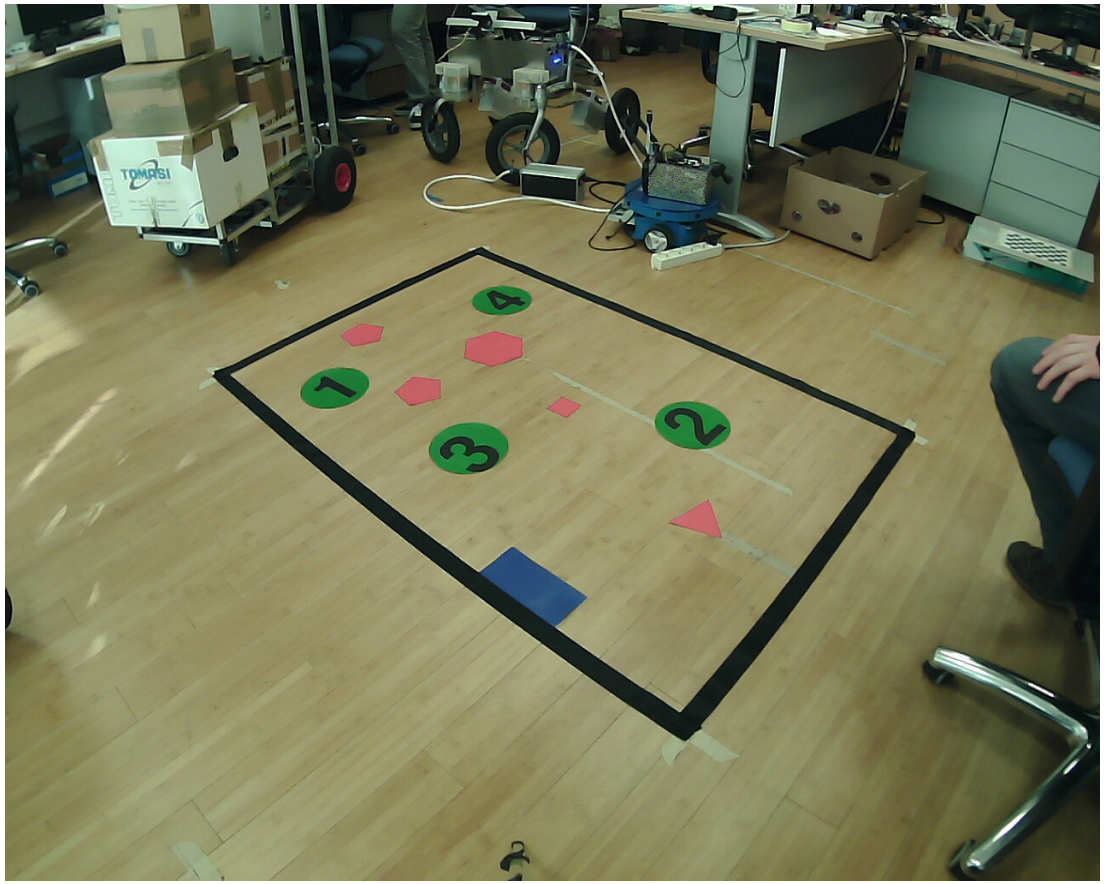
$$A = \begin{bmatrix} 8.4247565095622963 \times 10^2 & 0 & 6.3709750745251142 \times 10^2 \\ 0 & 8.4247565095622963 \times 10^2 & 4.9404840840221556 \times 10^2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\dot{d} = (-2.5214446354851400 \times 10^{-1}, 7.2467634259951161 \times 10^{-2},$$

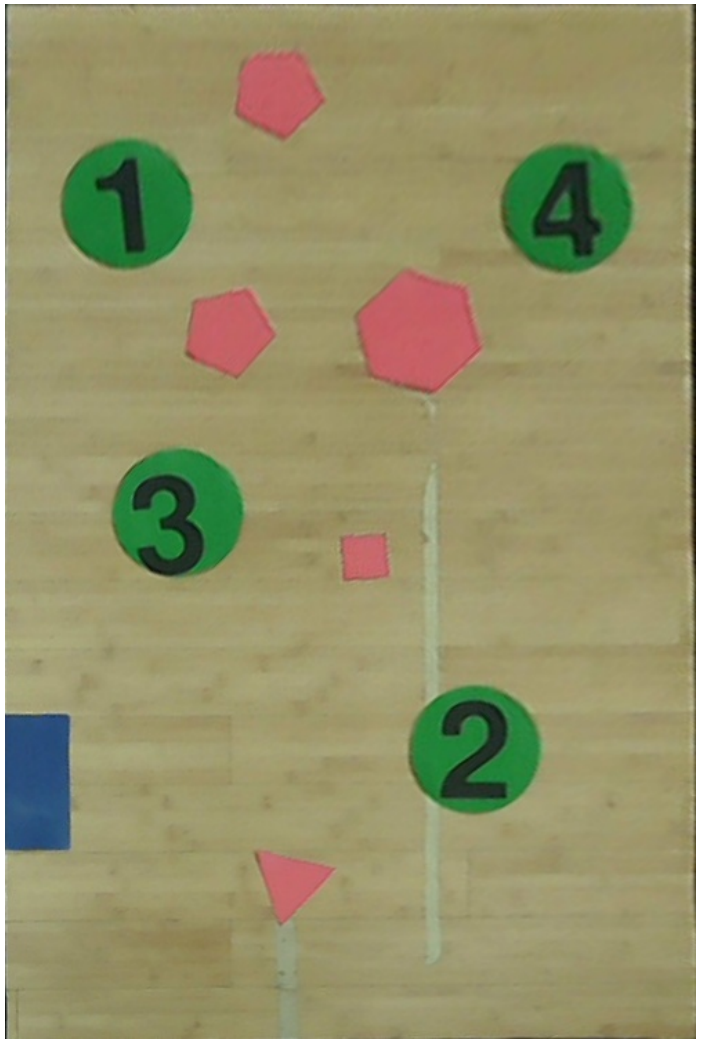
$$-3.7212601356153754 \times 10^{-3}, 4.3313659139950872 \times 10^{-4}, 0)$$

Unwrapping

Once the camera matrix and the distortion coefficients are known, the next step is to focus on the target, that is the circuit, and then to "un-distort" it.

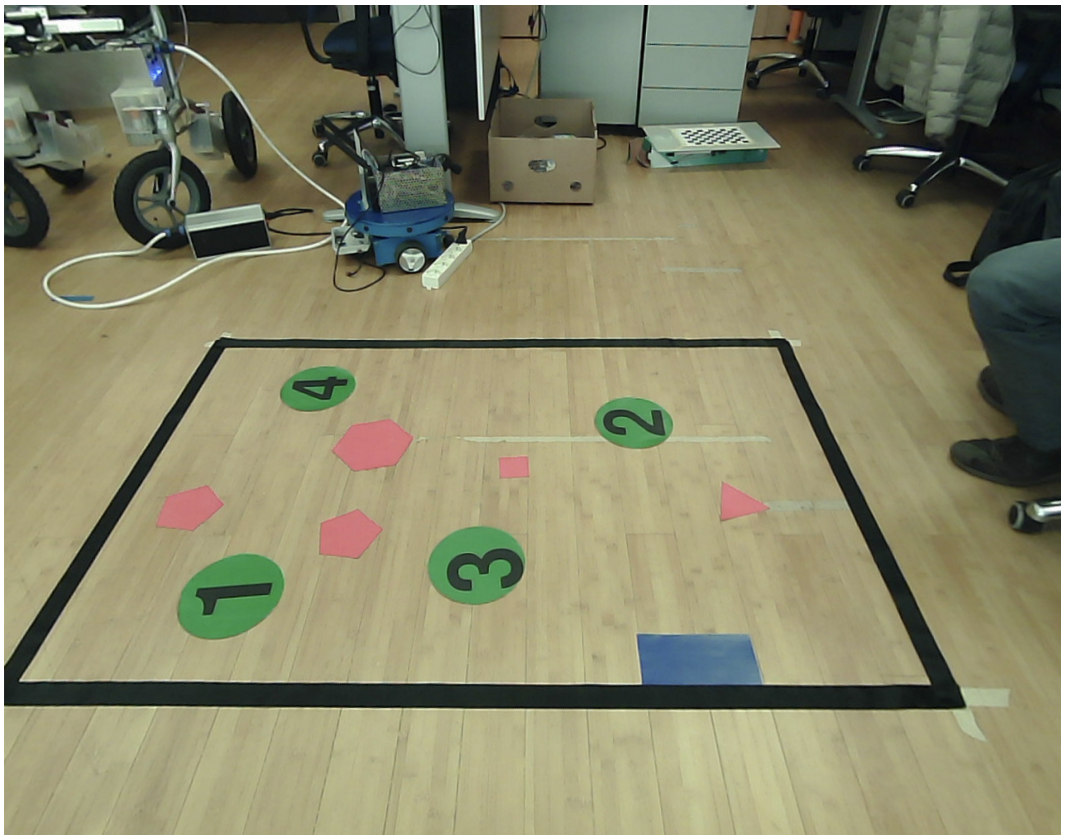


Unwrapping



Unwrapping

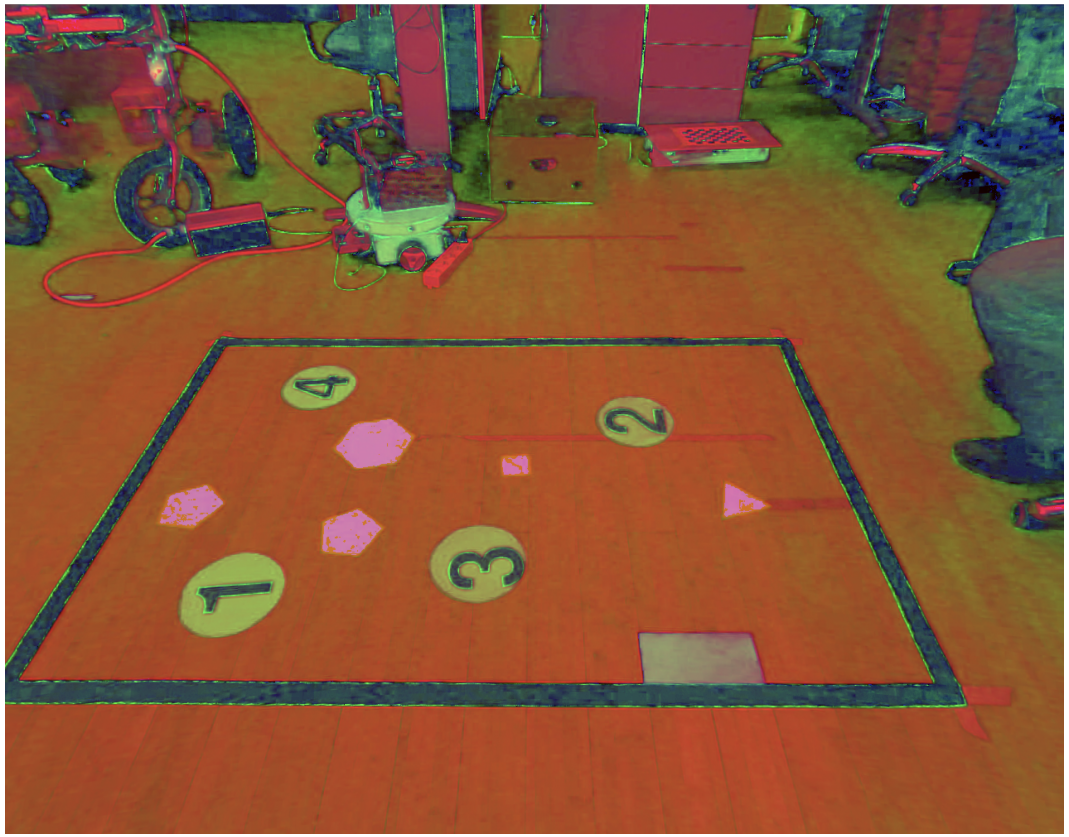
Straighten the image using the distortion coefficients.




```
loadCoefficients(calib_file, camera_matrix, dist_coeffs);
undistort(or_img, fix_img, camera_matrix, dist_coeffs);
```

Unwrapping

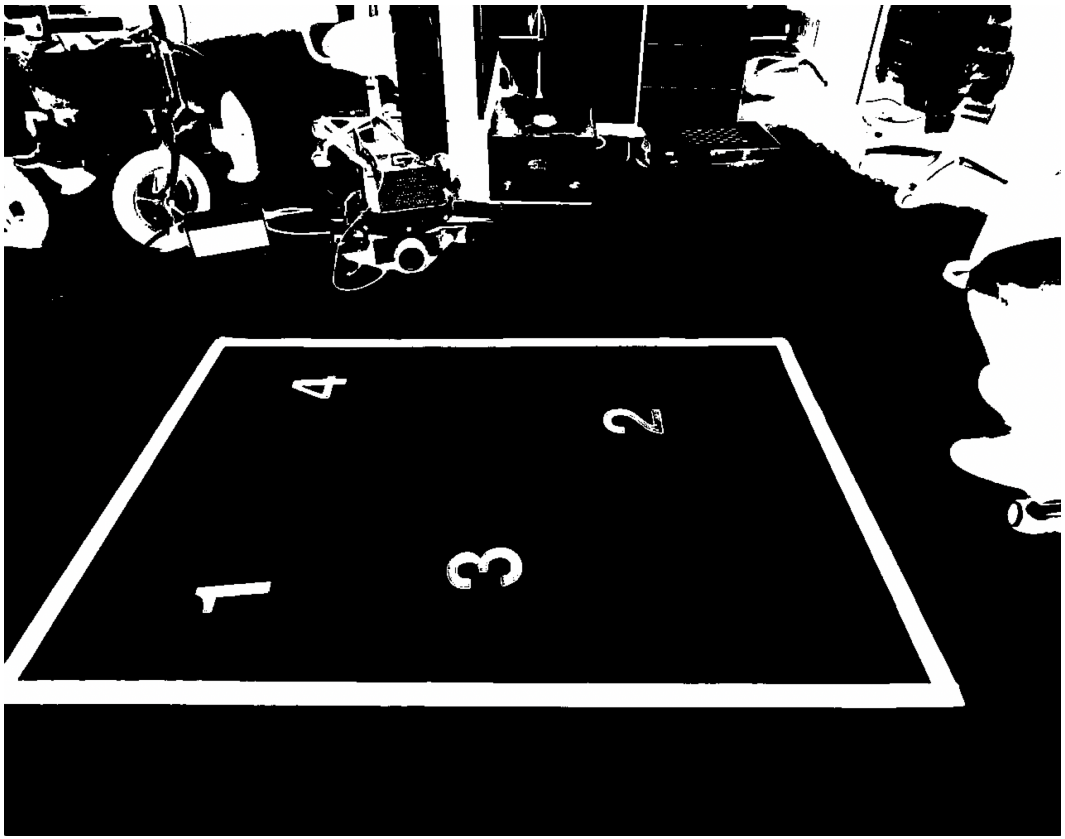
Then the image is converted from RGB to HSV.




```
cvtColor(fix_img, hsv_img, COLOR_BGR2HSV);
```

Unwrapping

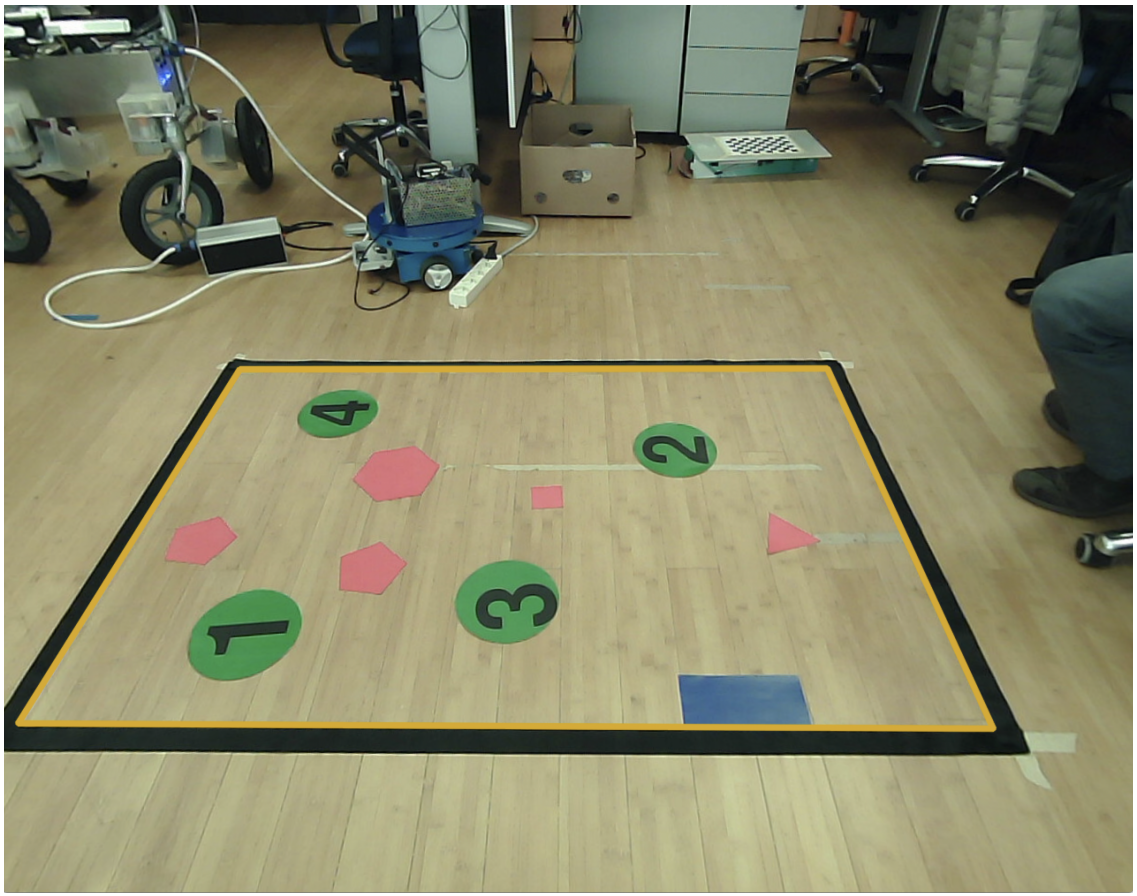
A black filter is added to highlight the black lines and areas.




```
FileNode Bm = fs_xml["blackMask"];
//<blackMask>0 0 0 179 255 70</blackMask>
inRange(hsv_img, Scalar(Bm[0], Bm[1], Bm[2]), Scalar(Bm[3], Bm[4], Bm[5]),
 ↪    black_mask);
```

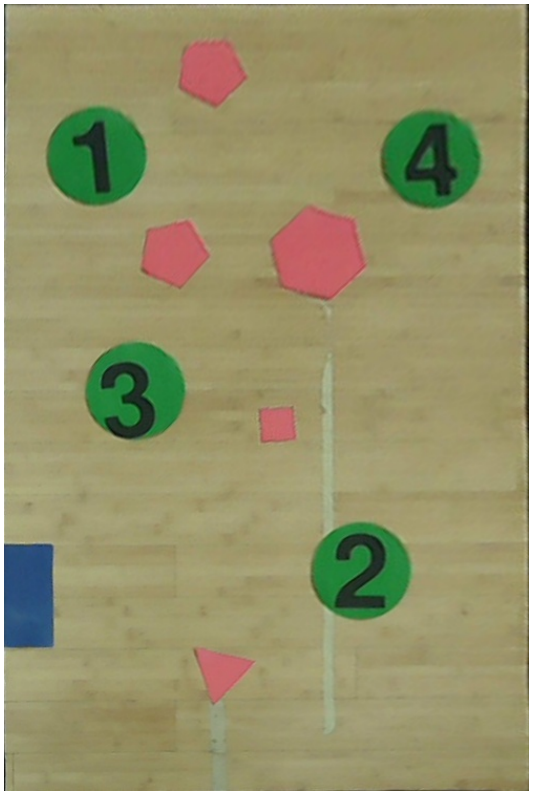
Unwrapping

Then the largest area enclosed by black lines is selected.



Unwrapping

At the end the image is cropped and turned.



Detection

The last phase consists of two more steps:

- Shape detection;
- Digit detection;

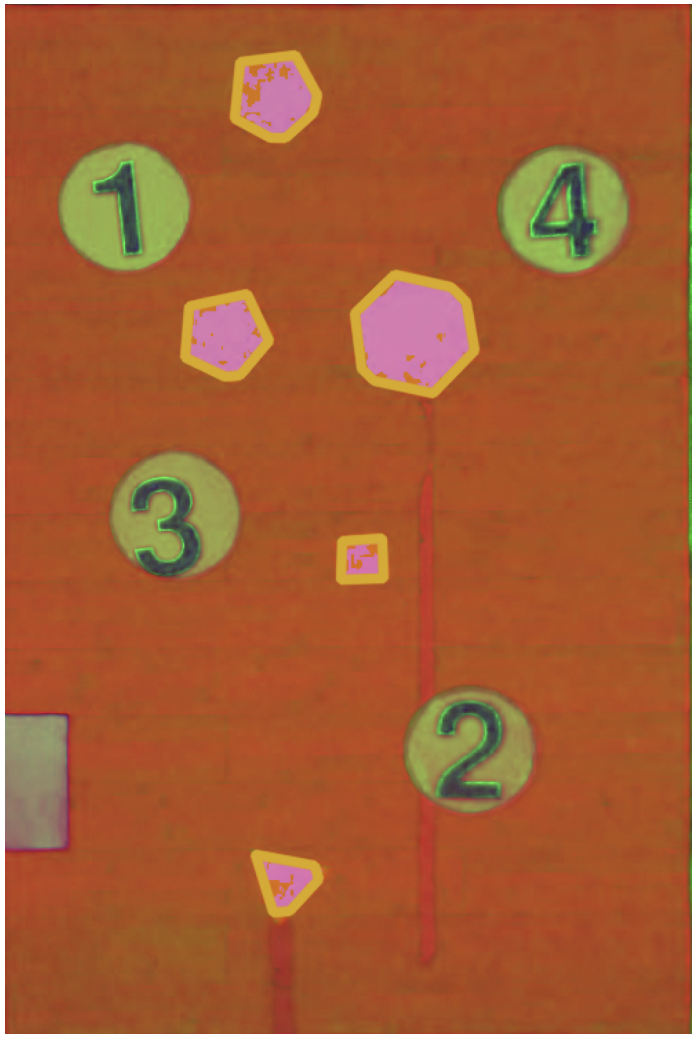
Shape detection

The shape detection consists of scanning the image to locate the various obstacles, targets and the arriving point.
Various filer are here used to achieve this goal:

- The obstacles are red, so a red filter is going to be used: 15 100 140 160 255 255;
- The targets are green, so a green filter is going to be used: 50 65 45 70 255 200;
- The target point is blue, so a blue filter is going to be used: 100 100 40 140 200 170;

The various items are going to be stored in a file with the coordinates of their angles.

Shape detection



Digit detection

The goal of this last phase is to consider the area around the targets, which is our ROI, and try to identify the number inside it.
This can be done through templates or through Tesseract API. For having better results the area around each target is:

- Cropped;
- Resized to 200×200 *px*;
- Given a black filter;
- Passed through a function `erode_dilation` which aims to removing noise and isolate the number;
- Rotate so that the number is going to be vertical.

At last the digit detection is being done with templates since the success rate has been greater than using Tesseract.