Dpto. Ingeniería de Sistemas y Automática
Universidad de Málaga

COMPUTER VISION

PRÁCTICA 7: 2D projective transformations (homographies)

Concepts: 2D homography and augmented reality

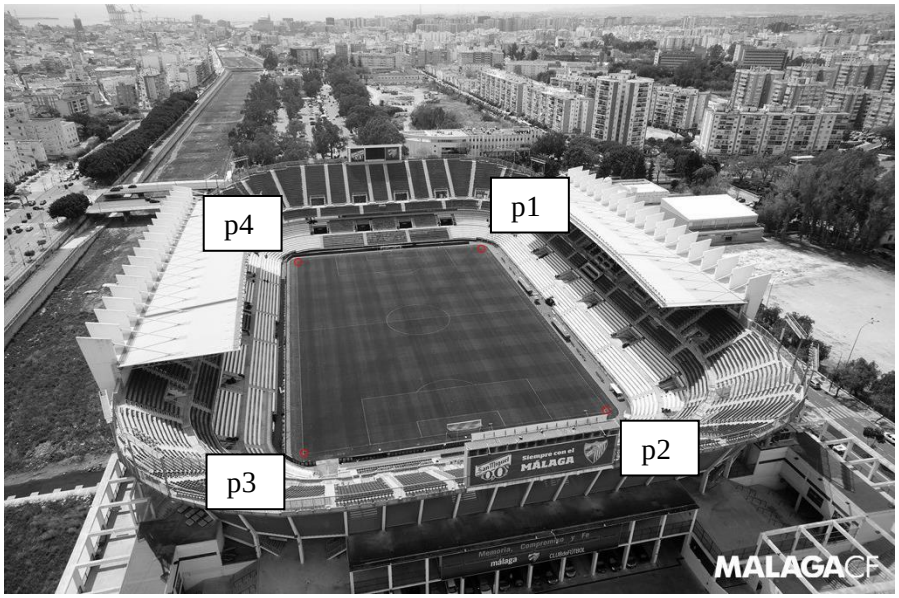
Recall from the previous lesson: a **2D homography** is a very general transformation between planes, in such a way that any two images from the same surface are related by a homography.

1. **My first homography.** Implement a code that removes the perspective of a plane in an image. For that, the code has to:

   a) Allow you to indicate four points of a plane in an image (the four corners of a table, for example. Get inspiration from the image below). *Note: if the image is not grayscale, convert it.*

```
image='la_rosaleda_2.jpg';
im=imread(image);
im=rgb2gray(im);
imshow(im);
title('Indicate the four vertices in the image');
P=ginput(4);
```

Points selection



b) Compute the homography between the chosen corners and the corners of a real rectangle (perpendicularly seen). Use the given function **homografia2D.m**, which implements the DLT method, introduced in the previous lecture. Could you use real rectangles of different size? *Note: if you use the same image as in the example, the dimensions of the soccer field in the Rosaleda Stadium are 68x105m.*

```
w=68;
h=105;
sortval=(P(:,1)-0).^2 + (P(:,2)-0).^2;
[~,sortorder] = sort(sortval);
P = P(sortorder,:);
P=[P ones(4,1)]';
Q=[0 0 1;
   w 0 1;
   0 h 1;
   w h 1]';

H=homography2d(P,Q);
```

c) Apply the computed transformation to the initial image and check that the plane perspective has been removed.

```
tf = maketform('projective',H');
img2 = imtransform(im, tf, 'XData', [1 w], 'YData', [1 h]);
figure(2)
imshow(img2);
title('Perspective rectangle');
```

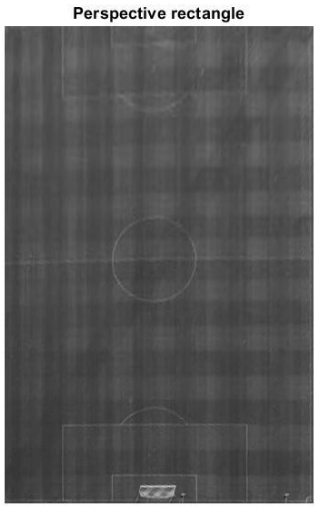


Image 1. Perpective of the field

2. **Augmented reality.** Put a rectangular image over the initial plane for doing augmented reality:

   a) Apply the inverse transformation to the chosen image.

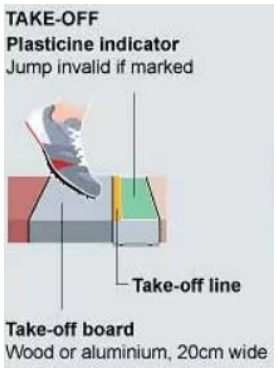
```
inv_tf = fliptform(tf);
```

   b) Put the resultant one over the initial image.

```
imageHolo = 'chiquito.jpg';
im_holo = imread(imageHolo);
im_holo = rgb2gray(im_holo);
img2 = imtransform(im, tf, 'XData', [1 w], 'YData', [1 h]);
im_holo=flip(imresize(im_holo',size(img2)),2);
[y,x]=size(im);
img2 = imtransform(im_holo, inv_tf,'XData', [1 x], 'YData', [1
y]);
ind = img2 == 0;
img3 = im.*uint8(ind)+img2.*uint8(~ind);
imshow(img3);
title('AR image');
```

Image 2. Augmented Reality on the original image

3. **Application: Long jump competition.** The *Olympics committee* has an especial work for us: to develop a computer vision program able to decide if a jump is valid (the jumper has not overstepped the plasticine board) and the distance from the step to this plasticine board.



The image to the right shows the parts of the zone where the jumper has to take-off: the take-off board (with a width of 20cm), the take-off line (included in the take-off board) and the plasticine indicator (10 cm width). In total, the area of this zone is 30x122cm.

The *Olympics committee* wants us to evaluate our software using a number of provided images, named *long_jump_0.png* to *lon_jump_4.png*, which are annotated with the distance from the jumper sneaker tip to the beginning of the plasticine indicator. With this goal, implement a code that:

1. Computes the homography between the take-off zone in the images and a rectangle with the real size (30x122cm).

```
w=30;
h=122;
image=sprintf('long_jump_%d.PNG',i);
im=imread(image);
imshow(im);
title('Indicate the four vertices in the image');
P=ginput(4);
sortval=(P(:,1)-0).^2 + (P(:,2)-0).^2;
[~,sortorder] = sort(sortval);
P = P(sortorder,:);
P=[P ones(4,1)]';
Q=[0 0 1;
   0 h 1;
   w 0 1;
   w h 1]';
H=homography2d(P,Q);
```

Although we use here ginput, we could use other method for obtain the four points. For instance, we could use a region method for separate some region and then use harris to guess where that four points are.

Image 3. Representation of the described method

2. Queries you to point at the jumper sneaker tip in the zone without perspective.

```
title('Indicate the jumper sneaker tip');
[x,y]=ginput(1);
p=[x;y;1];11111
```

3. Having the pointed coordinate, computes and reports the distance from it to the plasticine indicator. *Note: remember that it is 10cm width.* Is this distance the same as the provided one?

```
q=H*p;
q_nh(1:2) = q(1:2)/q(3);
txt=sprintf('Distance from the jumper sneaker to the plasticine
indicator: %2.6f cm',20-q_nh(1));
xlabel(txt);
```

As we see in the images, the distances are really proximate from the real value. Some are by 0.5 and some by decimals.

Image 4

Image 5

Image 6

Image 7

Image 8

4. **OPTIONAL! Application: Offside detection.** *LaLiga* has also called to our offices asking for a software to draw a line at the position of the last defender in a soccer match, making easier in this way to decide if a player was offside. To fulfill this requirement, develop a code that:

   a) Removes the perspective of the penalty area. The size of this area is 16.5x40.3m. Use the provided image *offside.jpg* for that.

```
w=16.5;
h=40.3;
figure
image='offside.jpg';
im=imread(image);
imshow(im);
title('Indicate the four vertices in the image');
P=ginput(4);
sortval=(P(:,1)-0).^2 + (P(:,2)-0).^2;
[~,sortorder] = sort(sortval);
P = P(sortorder,:);
P=[P ones(4,1)]';
Q=[0 0 1;
    0 h 1;
    w 0 1;
    w h 1]';
H=homography2d(P,Q);
```

   b) Queries you to point at the position of the last defender.

```
title('Indicate the last defender');
[x,y]=ginput(1);
p=[x;y;1];
```

   c) Draws a line in the initial image crossing that position. There are two ways for doing this (just choose one):

      i. Building the line ax + by - c = 0 in the image without perspective, and using H' to compute its coordinates in the initial image. Remind that $l = H^T l'$.
      ii. Define first a line crossing this point in the image without perspective by setting two points, and then translate these points to the initial image using the inverse homography. Finally draw a line connecting them.

```
q=H*p;
q_nh(1:2) = q(1:2)/q(3);
pq1=[q_nh(1);0;1];
pq2=[q_nh(1);h;1];
p1=inv(H)*pq1;
p1=p1(1:2)/p1(3);
p2=inv(H)*pq2;
p2=p2(1:2)/p2(3);
x=[p1(1),p2(1)];
y=[p1(2),p2(2)];
hold on
l=plot(x,y,'r');
l.LineWidth=2;
```

   d) Is the Luis Suarez goal legal?

      No, it isn't

Image 9. Verification of the goal