Dpto. Ingeniería de Sistemas y Automática
Universidad de Málaga

COMPUTER VISION

PRÁCTICA 7: 2D projective transformations (homographies)

Concepts: 2D homography and augmented reality

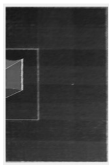
Recall from the previous lesson: a **2D homography** is a very general transformation between planes, in such a way that any two images from the same surface are related by a homography.

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

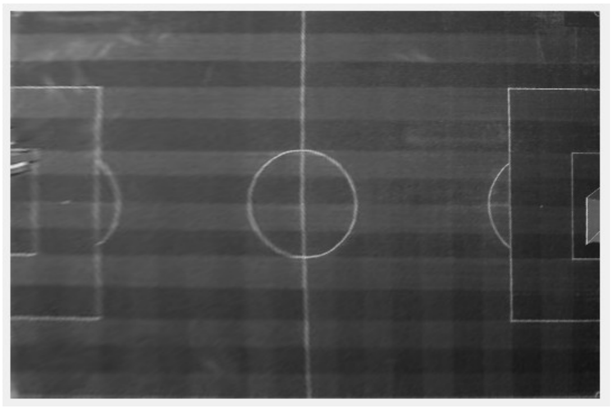
   a) Allow you to indicate four points of a plane in an image (the four corners of a table, for example. Get inspiration from the image below).

   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?

   It seems to work just fine with rectangle of different sizes. Here is the transformation of the area.



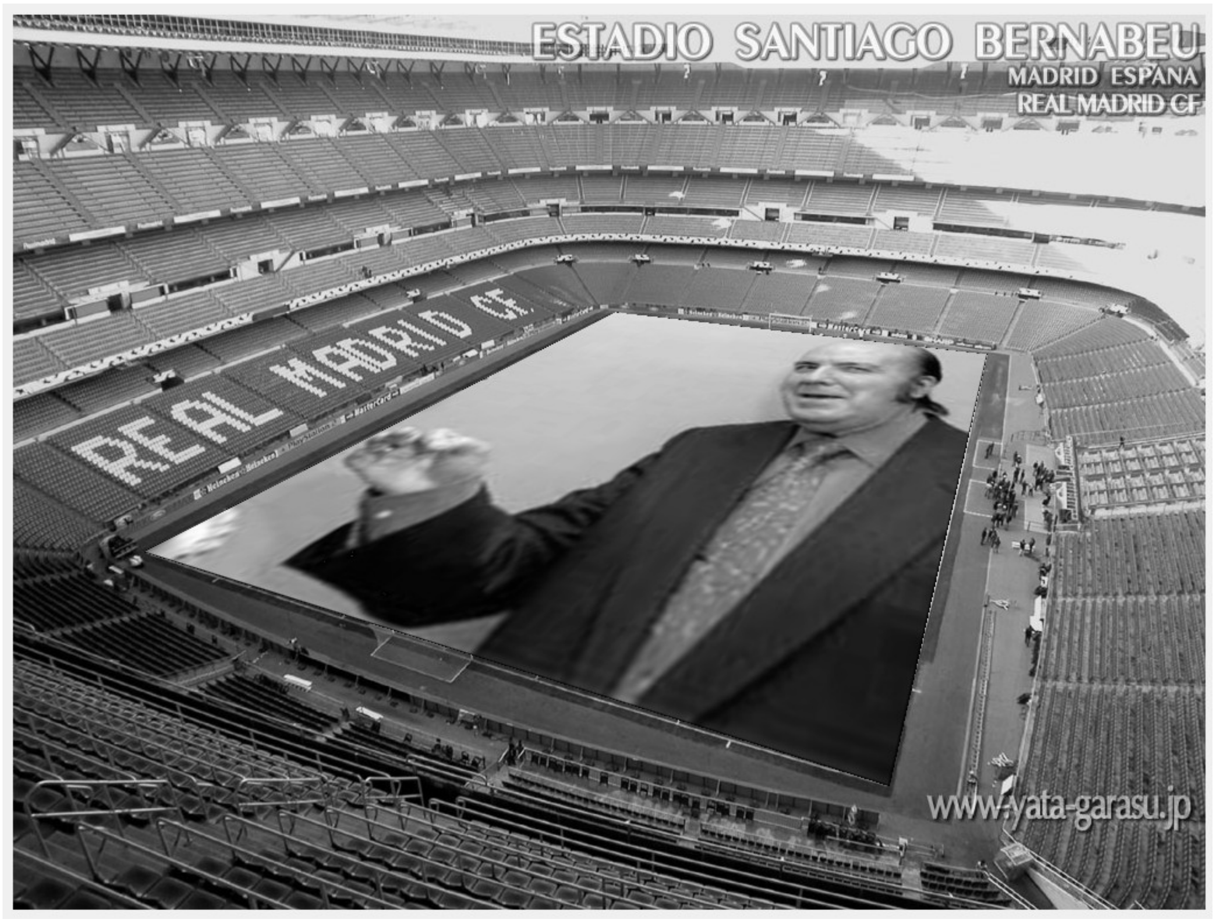
   c) Apply the computed transformation to the initial image and check that the plane perspective has been removed.
   We can see how the image lose quality on the left side. This is because less pixels strecth to cover the same distance.



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

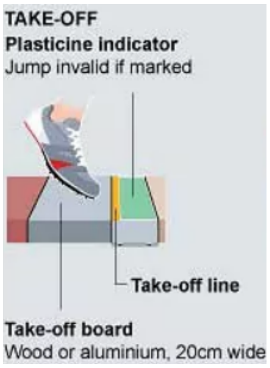
   a) Apply the inverse transformation to the chosen image.
   I've first applied the transformation of Ex.1 to the field. Then, I resized chiquito's image to that size. Finally, I used the fliptform transformation to transform it and cover the field.

   b) Put the resultant one over the initial image.
   After the transformation, I had to deal with some padding added by matlab. I removed it with a mask, cloning the football field grayscale onto the black pixels of the projected 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 over-stepped 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).

      I just copied the code from the first exercise to do this task, but I had to change some parameters to fit the 30x122cm dimensions.

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

   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?

      I've just used a cross-multiplication to calculate the distance between the tip of the sneaker and the take off line. The distance that appears on the screen seems to be the total distance rounded up. I've got around 10.5cm.