

Homework for Computer Vision (Como) and IAS (Milano) Courses, 2010/2011

Version 1.0. Updated versions may be released in the future, with minor changes which will be clearly marked.

Deadline

2010 Feb 14, 10:00. See other important rules at the end of the document.

Introduction

Consider perspective images depicting sport playing fields, such as in the links below (you could also shoot your own images or find other ones).

- <http://tinyurl.com/36lgwzj>
- <http://tinyurl.com/2ugz14o>
- <http://tinyurl.com/3ya3v2y>
- <http://tinyurl.com/2vu6qyj>
- <http://tinyurl.com/34zaq7f>
- <http://tinyurl.com/3xkbjux>

You can solve the following problems using one or more of these images – use the ones which seem more convenient and/or interesting.

Part 1 – Geometry

1. Consider a rectangle in the scene with known dimensions. Identify its projection in the image (by clicking its four corners, use `getpts`) and draw it over the original image (e.g. `imshow` and `plot`).

2. Find the image of line at the infinity corresponding to the plane of the rectangle. OPTIONAL: find or shoot an image where the horizon is visible (at least in part), then verify that the line at the infinity you found is close to the horizon.
3. Define a convenient reference frame in the playing field (unit of measure will be meters); by using the four points you identified, find an homography which brings scene points expressed in such reference frame into their projection on the image. Test by projecting the four points, and additional meaningful points.
4. Find the inverse homography and rectify the playing field.
5. Write a function which allows you to click on a point of the image (such as the position of a player), and:
 - displays its coordinates and/or some meaningful information (such as the distance from the goal);
 - draws a line passing through the point, which is parallel to a meaningful line in the scene (such as the offside line);
 - draws a quadrangle which represents the projection of a square centered on the player and lying on the plane of the playing field; an example is the indicator visible at the mouse position when using Google street view;
 - OPTIONAL: draws an ellipse which is the projection of a circle on the same plane of the playing field, centered on the clicked point; an example is the indicator of the active player in some video games (<http://tinyurl.com/3a2gmff>).
6. Consider now $N = 5$ couples of points with known position in the scene and in the image. Leave one of the points out (let's call it a). Estimate the homography from the resulting $N - 1 = 4$ points. Now pretend you do not know the scene position of a : estimate it by transforming its image using the homography you just found; as you actually know the "real" position, measure the resulting error (in meters). Compute the average of this error when a is each of the $N = 5$ points.
7. Describe at least one technique for estimating the homography when more than 4 correspondences are available, and optionally implement it.
8. OPTIONAL: consider point 5, and repeat the steps with $N > 5$. Then you will need to estimate the homography using $N - 1 > 4$ points. Estimate the average error.
9. OPTIONAL: plot how the average error changes when the number of points for estimating the homography increases. You can input once for all a large number of correspondences (say, 10), then compute the homography using a subset of them (but at least 4), and compute the error using the remaining ones, possibly considering different permutations.

10. Consider an image where you see the projection of a circle and you can estimate the projection of the line at the infinity for the plane where the circle lies. Find the conic corresponding to the projection of the circle from $N = 5$ points manually clicked. Find the projection of the line at the infinity. Discuss how you can rectify the image and **OPTIONALLY** implement the rectification. **OPTIONALLY** discuss how you can find the conic with more accuracy when using $N > 5$ points.
11. Consider the last of the images linked in the introduction; find the vertical vanishing point. Briefly discuss how the camera could be calibrated and under which assumptions, by finding constraints on the image of the absolute conic. **OPTIONALLY** discuss how the height of the lighting poles can be estimated with respect to the length of the playing field. **OPTIONALLY** implement the camera calibration.

Part 2 – Image Processing

1. Implement the harris corner detector from scratch in a function. Apply the detector to an image, select the 10 largest peaks and mark them in the image. How many of such points could be really considered salient points? Repeat after resizing the image to be 1/2 or 1/4 of the original size (use `imresize`).
2. Consider the problem of finding the edges of the lines of the playing field. Discuss the results of the Canny detector when varying the σ parameter; also, discuss the effect of applying the Canny edge detector on an image resized to be 1/2 or 1/4 of the size (use `imresize`).
3. Consider the hough algorithm for finding lines, and apply it to the edges you could get in the previous step. Use MATLAB function `hough`. Draw all the lines which have peaks in the hough matrix which are larger than a given threshold; discuss what happens when changing such threshold. **OPTIONAL**: briefly describe how the matlab functions `houghpeaks` and `houghlines` work.
4. Write a script which draws the line passing through a point clicked by the user in the image (use `getpts` for finding its coordinates), which has the highest peak in the hough matrix. In order to implement this, you have to identify the elements of the hough matrix that correspond (with some tolerance) to lines passing through the clicked point.
5. Write a script that:
 - computes the hue value of a pixel clicked by the user; use `rgb2hsv` to compute hue values.
 - modifies the image in such a way that all pixels with a similar hue value are changed to black.

Discuss: what does “similar” hue mean?

6. OPTIONAL: instead of sampling the hue value at the single pixel that the user clicked, consider the *average* hue in a rectangular area of the image selected by the user (try `imcrop`). Careful: how to compute the average hue? (hint: not just with `mean`). Compute the mean and the standard deviation of the hue value of the pixels inside the area, and define a confidence interval. Test your script on a red object.
7. In many scenarios, you already know that the lines you are looking for have a specific color over a specific background, and/or that they originate by edges close by. Discuss how these constraints can help you improve the performance of the line detector. OPTIONAL: implement some of the proposed improvements.

Homework Rules

Submission

The homework must be submitted before the deadline written at the beginning of the document.

Submission consists in a mail to: `alessandro.giusti@polimi.it`, possibly sent from your `@stud.polimi.it` address, with the following subject: `CV Homework Name Surname` or `IAS Homework Name Surname`.

You will receive a confirmation email shortly after the deadline. Contact `alessandro.giusti@polimi.it` if you do not receive a confirmation email few hours after the deadline.

Your homework must be contained in a `.zip` archive – max 4MB.

The zip must contain:

- a short PDF file, no more than 1500 words plus formulas and figures. If possible, the total length should not exceed 6 pages. Do not include code in the PDF (except isolated significant lines if strictly needed). The PDF must describe your solution to the homework problems and show results, if applicable. Be clear and concise.

The PDF should include the final results of your matlab code (it should not be necessary to run the code in order to see your results). All figures in the PDF must be created by your script (or derived from data created by the script). If you include figures created in other ways, note it.

- all the code you have written, in an easily runnable form, with instructions if needed.

Language

You can submit your homework either in english or italian. No bonus/malus is applied for either language.

Programming language

Using matlab (or octave) is suggested but not required.

You can use other languages (C, C++, Java, Python, OCAML...) if you wish. Please note that, in any case, you must always submit all the source code, and provide a simple way to compile/run the code either in windows or linux (or both). We will not accept code which only runs under mac. Implementation using languages alternative to matlab is subject to a bonus in order to account for the increased difficulty.

Text/code ownership, collaboration

A limited amount of collaboration on high-level or specific issues is allowed and encouraged; however:

- You must be the author of all the text and figures you submit in the PDF.
- You must be the author of every line of code you submit (this implies that you must fully understand every single line of code you submit). In some situations, especially when solving "extra" points, using external code is tolerated. However, whenever you use external libraries or copy/paste code found somewhere on the internet (for example from the MATLAB file central), you must explicitly state this and cite the source in the PDF and in the relevant sections of your code. If possible, show that you understand how that code works.

Extras

If you are interested, you are encouraged to go beyond what's strictly required in the text and improve/personalize your work (provided that the basic requirements are met).