CPS296.1- Homework 1

Due on February 7, 2006

Questions may continue on the back. Please write clearly. What I cannot read, I will not grade. Typed homework is preferable. A good compromise is to type the words and write the math by hand.

This assignment describes a simple lens distortion calibration procedure, and you will be asked to carry out its key steps. You will find all files related to this homework, including this document, on the class web page. First, the procedure is outlined in its general lines. Details, with homework questions, follow.

WARNING: NONE OF THE QUESTIONS BELOW IS HARD. THE ONLY DIFFICULTY IS TO AVOID MISTAKES. CONSEQUENTLY, MOST OF YOUR GRADE WILL COME FROM PRODUCING AN ACCEPTABLE UNDISTORTED IMAGE AT THE END. Matlab is recommended. If you use anything else, you will have to obtain your own image libraries, an image viewer (see the Resources in the class web page), and write your own version of `unwarp`. In Matlab, you can read an image with `imread`, and write it with `imwrite`. A good display is obtained with `imagesc` (scaled version of `image`).

If pixels are not square, a given angular distance between two points in the image can cover some number of pixels horizontally and a different number of pixels vertically. In standard CCD cameras, the aspect ratio of a pixel is often different from one. A simple way to measure this ratio is to place a table-tennis ball in the center of the field of view and compute its elongation. For instance, suppose that the ball measures 30 pixels in width and 45 in height. Then, since the true image of the ball is circular, there are 45 pixels vertically in the same space in which 30 pixels fit horizontally. The aspect ratio is therefore $45/30 = 1.5$, in the sense that pixels are 1.5 wider than they are tall.

The *field of view* is the angular distance between two points at the opposite edges of the image, either horizontally or vertically. Thus, there are two fields of view, one horizontal and one vertical. To measure the horizontal field of view, place a planar object (such as a sheet of paper with some lines on it) in front of the camera and move it until it fills the image horizontally. Then measure the distance between the camera and the object, as well as the width of the object. Simple trigonometry yields the horizontal field of view. The vertical field can be measured similarly.

To clarify this computation, if the object is d cm wide and the distance between object and camera is D, then the field of view is

$$\phi = 2 \arctan(\frac{d}{2D}) .$$

This leaves the radial lens distortion parameters to be calibrated from scratch. A sheet with concentric circles of linearly increasing radii, held flat and perpendicular to the viewing direction, is photographed approximately centered in the digitized image. If there were no distortion, the circle radii measured in the image would be proportional to the radii of the true circles. In the presence of distortion, on the other hand, different circles are magnified differently in the image. The calibration procedure below measures the image radii, and finds a polynomial that maps the distorted radii to the true radii. The coefficients of this polynomial are called the *distortion coefficients*. Given any image produced by the same camera-lens combination, the distortion coefficients allow to unwarp the image, that is, to deform it geometrically so as to eliminate or at least reduce distortion.

The center of the circle pattern is marked by hand in the image, as well as a few points on the circles along some radial lines. If radial distortion is modelled by an even-symmetric, fourth order polynomial, the coefficients of this polynomial can be computed by a linear fitting procedure from the distances from the center to each of the circles. The details will be explored in the problem questions below.

To perform the calibration, you will use the image `circles.gif`, and Matlab to inspect the image and measure coordinates in it. To test the calibration, you will unwarp the distorted image `lab.gif`, and check that lines appear straight after the operation.

1. The file `circles.gif` contains an image produced by holding a piece of paper with evenly spaced rings up to the camera. Use `imagesc` to examine this file. Please note that the rings at the periphery of the image appear thinner than the ones closer to the center. This is the result of lens distortion.

Clicking on the image after running the `ginput` function will collect the image coordinates required below. These are your "image measurements." Make sure you figure out which is the row and which is the column coordinate for each entry. Make the image large on the screen, so measurements are more accurate.

(a) Assume that the optical center of the image is at the center of the small black dot in the middle. Give your best estimate of the row and column coordinates r and c of this point, obtained by clicking the mouse where you think the center is.

(b) Confirm that the aspect ratio of the pixels is one by measuring the width and height in pixels of a circle and checking that they are nearly equal. State which circles you used (counting from the center), what measurements you made, and your calculations.

(c) Determine the horizontal and vertical fields of view of the camera. The radii of the ring boundaries start at 1 cm and are 1 cm apart (that is, the radius of the white circle in the middle is 1 cm; the outer radius of the black ring immediately enclosing it is 2cm; and so forth). The paper with the rings drawn on it was positioned 17 centimeters from the center of projection of the camera, and orthogonal to the optical axis. Again, describe your measurements and calculations. This result cannot be very accurate, because the rings are not tangent to the image boundaries.

(d) What is the focal distance used for this image, in pixels? Show your reasoning.

2. As you noticed, the circles get closer together toward the edge of the screen. This effect is due to the radially symmetric distortion of the camera lens, which you are about to determine. We use the following distortion model:

$$x_d = x[1 + (k_1 + k_2\rho^2 + k_3\rho^4)] \quad (1)$$
$$y_d = y[1 + (k_1 + k_2\rho^2 + k_3\rho^4)] \quad (2)$$

where $\rho = \sqrt{x^2 + y^2}$ is the radial distance of a pixel at (x, y) from the image center in the ideal, undistorted image. The distorted coordinates x_d, y_d are your image measurements in the distorted image, relative to the image center. The coordinates x, y are coordinates you would measure in the absence of distortion.

(a) It is easiest to measure coordinates horizontally in the image. On the scan line through the central dot in the pattern, measure the distorted column coordinates c_d of the white circles, then obtain the values of x_d as

$$x_d = c_d - c$$

where c is the column coordinate of the center dot. Write the vector of values x_d from left to right in the image.

(b) To obtain the corresponding "true" values x, assume that the smallest circle in the image is undistorted. For its two intersections with the middle scan line, we have $x = x_d$. From this, and assuming that the target with the pattern is exactly fronto-parallel, infer the other values of x. Write the resulting vector of values, from left to right in the image. Plot x_d as a function of x. Label the axes, including the units of measure.

(c) We only use the first of the two equations (1) and (2). Since along the middle scan line we have $y_d = 0$, we have $\rho^2 = x^2$, and equation (1) can be rewritten as follows:

$$\delta = x_d - x = k_1 x + k_2 x^3 + k_3 x^5 \ . \quad (3)$$

Thus, an even-symmetric distortion leads to an odd-symmetric polynomial for the difference $\delta = x_d - x$. Plot the difference δ for the vectors of values you computed above. Label the coordinate axes, including the unit of measure.

(d) Determine the coefficients k_1, k_2, k_3 of the fifth-order, odd-symmetric polynomial of the form (3) that best fits the values of δ you found above in the least-squares sense. If you don't remember how to do least-squares fitting, the idea is to solve a linear system of equations:

$$M\mathbf{k} = \delta \ .$$

This system is formed as follows: let $\mathbf{x}$ be a column vector that collects your values of x, and let $\mathbf{x}$.^3 and $\mathbf{x}$.^5 be the two column vectors of the third and fifth powers of the entries of $\mathbf{x}$, respectively. Then, $M = [x, x.\hat{}3, x.\hat{}5]$. Also, let $\mathbf{k}$ be the column vector with entries k_1, k_2, k_3. The right-hand side δ is the vector of differences you computed in the previous question. You should get a value of k_2 several orders of magnitude smaller than k_1, and a value of k_3 several orders of magnitude smaller than k_2. This is because x contains large numbers, and the third and fifth powers are very large indeed. This could be avoided by normalizing image coordinates. As long as you use double precision (as Matlab does by default), this is not necessary. Write the values of k_1, k_2, k_3 with five significant digits. [Hint: in Matlab, the system above is solved as `k = M\delta`. Also, print the values separately as `k(1), k(2), k(3)`, or else the smaller values will all look equal to zero. Alternatively, do `format long` to display more digits.]

(e) Plot the function δ as given in (3) on the same diagram on which you plotted the values you measured for δ. Do not expect an exact match, but the two plots should be rather close to each other.

3. Now that you have obtained your model for the distortion of the camera, use the supplied function `unwarp.m` to test your solution. This function takes an image `in`, a vector `k` with the three values of k_1, k_2, k_3, and a vector `center` with the row and column coordinates (in this order!) of the center dot, and returns an "unwarped" image. You may ignore the fourth argument `mode` or, equivalently, set it to `'linear'`. Run the function on the image read (with `imread`) from `lab.gif`, which was taken with the same camera. As you can see by inspecting `lab.gif`, straight lines in the world are severely curved in this picture. With your calibration, this should improve to a noticeable extent. Print (use `print -deps` if you want to include the result into a Postscript or PDF file; use `print -dps` if you just want to send the file to a Postscript printer) and hand in the original and unwarped images, side to side. If your image looks substantially better than the input, you are done. Otherwise, you may want to check your solution. Given all the assumptions and simplifications, you may not obtain a perfect output.

A Programming Note

A C program that does the same as `unwarp.m` would scan the output image in a doubly-nested `for` loop, use the distortion parameters to determine which point in the input image corresponds to the current pixel in the output image, retrieve that point (by using interpolation to approximate the value of a pixel with non-integer coordinates), and place the value in the current pixel of the output image.

In Matlab, the doubly-nested loop would make the program very slow, because the file with the code is interpreted, rather than compiled. Please study how `unwarp.m` is written to avoid loops. Of course, the doubly-nested loop is merely pushed into `interp2`, but that is fast, because it is a compiled function.

It is a useful exercise (not required for this homework) to rewrite this function in Matlab, but with the doubly-nested `for` loop as described above, and compare the running times of the two versions.