CPS296.1- Homework 1

Due on September 10, 2007

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.

1. A certain object in front of a camera with a thin lens with a focal length of 16 millimeters is well focused when the focal distance is 16.2 millimeters. How far is the object from the camera?

2. A 4 centimeter diameter ping-pong ball is placed one meter away from a camera. The ball is in focus at a focal distance of 16 millimeters. The image of the ball is 60 pixels wide and 80 pixels tall.

(a) What are the horizontal and vertical focal distances in pixels? [Hint: draw a picture.]

(b) Determine the dimensions of a pixel in microns.

(c) If the sensor is 480 pixels tall and 640 pixels wide, what is its size in millimeters? [Note: this is not going to be one of the standard sizes.]

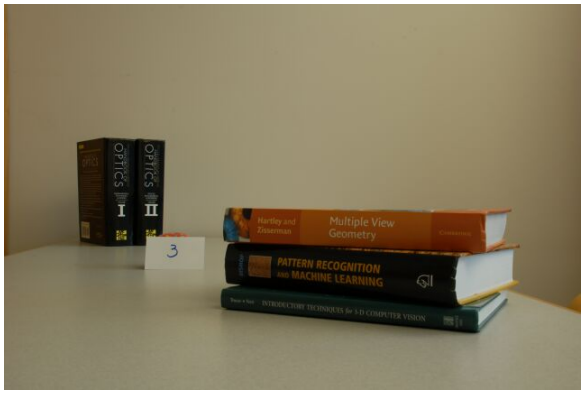
3. Prove that the perspective projection of a line is still a line. Hint: the algebraic proof is brute-force and tedious. A geometric proof (reasoning about lines and planes) is very simple. Full credit for either style.

4. The images `1.jpg`, `2.jpg`, `3.jpg`, `4.jpg` are reproduced below at a reduced scale, and are available on the class homework page at full resolution. They were taken with four different combinations of camera gain and aperture: the two apertures were $f/25$ and $f/4$, and the two gains were ISO 1600 (high gain) and ISO 100 (low gain). For each picture, the camera automatically chose an exposure time that would expose the image well.

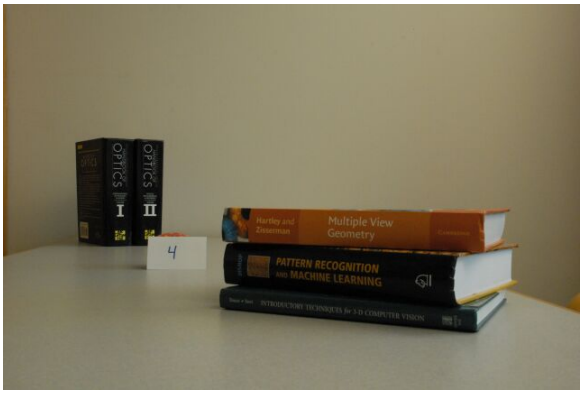
(a) Hand in a table with the picture number and the corresponding gain and aperture. Explain your answer. [Hint: this question is hard to answer from the reproductions below. Look at the original images instead.]

1.jpg. Gain: _____ . Aperture: _____

2.jpg. Gain: _____ . Aperture: _____



3.jpg. Gain: _____ . Aperture: _____



4.jpg. Gain: _____ . Aperture: _____

(b) The image that was taken with aperture $f/25$ and gain ISO 1600 ended up requiring an exposure time of 1/5 of a second. Estimate the exposure times for the other three pictures, assuming that scene lighting did not change between images. The ISO number is proportional to gain, and image brightness is proportional to the area of the lens aperture. Explain your reasoning.

5. The Matlab function in the file `makeGrid.m` available on the class homework page has header

```
function [x, y] = makeGrid(m, n)
```

and returns the x and y coordinates of $2mn$ points on a square grid with m horizontal and m vertical lines, with n points on each line. The output arrays `x` and `y` have n rows and $2m$ columns, and are arranged so that the `plot` command does the right thing. For instance, to see a 7×7 grid, type

```
[x,y] = makeGrid(7, 30);
plot(x, y, 'b')
axis equal
axis off
figure(gcf)
```

In this code, the command `axis equal` gives the grid the proper aspect ratio.

(a) Write a Matlab function with header

```
[xd, yd] = distort(x, y, k)
```

that takes a grid as output by `makeGrid` and applies the fourth-order lens distortion function described under *Quantitative Aspects of Lens Distortion* in the lecture notes on *Image Formation*. The last argument `k` to `distort` should be a vector with two entries defined as follows:

$$\texttt{k(1)} = k_2 \quad \text{and} \quad \texttt{k(2)} = k_4$$

(left-hand sides are Matlab variables and right-hand sides are symbols used in the notes). Hand in your code and the graph obtained by plotting the coordinates produced by the following code:

```
[x,y] = makeGrid(7, 30);
[xd, yd] = distort(x, y, [0.15 -0.04]);
clf
plot(xd, yd, 'b')
printGrid
```

The `printGrid.m` file is available on the class web page as well. You get half the credit for writing `distort` correctly, and the remaining half for doing it without `for` loops.

(b) The lens distortion equations

$$\begin{aligned} x_d &= x(1 + k_2 r^2 + k_4 r^4) \\ y_d &= y(1 + k_2 r^2 + k_4 r^4) \end{aligned}$$

are linear in k_2 and k_4. One can bring all the terms with k_2 or k_4 to the left-hand side, and all remaining terms to the right-hand side to obtain equations of the following form

$$\begin{aligned} a_{11} k_2 + a_{12} k_4 &= b_1 \\ a_{21} k_2 + a_{22} k_4 &= b_2 \end{aligned}$$

with suitable definitions of the coefficients. With N points one can write $2N$ equations of this form, which can be packaged into an over-constrained linear system of the form

$$A\mathbf{k} = \mathbf{b}$$

where A is a $2N \times 2$ matrix, $\mathbf{k}$ is a 2-dimensional vector, and $\mathbf{b}$ is a $2N$-dimensional vector. In Matlab, such a system can be solved with the single line

```
k = A \ b;
```

and the vector `k` then contains the least-squares estimate of k_2 and k_4.

Write a function with header

```
function k = distortion(x, y, xd, yd)
```

that takes arrays of coordinates as computed by `makeGrid` and `distort` and estimates the two-entry vector `k` of distortion coefficients as explained above. Check that running your function on the data you produced earlier returns `k = [0.15 -0.04]`.

This time you get 80 percent of the credit for doing this right, and the remaining 20 percent for not using `for` loops. [Hint: to transform a matrix `x` into a vector in Matlab, just say `x = x(:);`]

(c) [Optional problem. No credit.] Of course, in real life data is noisy, so the distortion estimates you'll get with data that comes from image measurements will be approximate. Play with your code by adding a little Gaussian noise (with the Matlab function `randn`) to `xd` and `yd`, and see what happens.