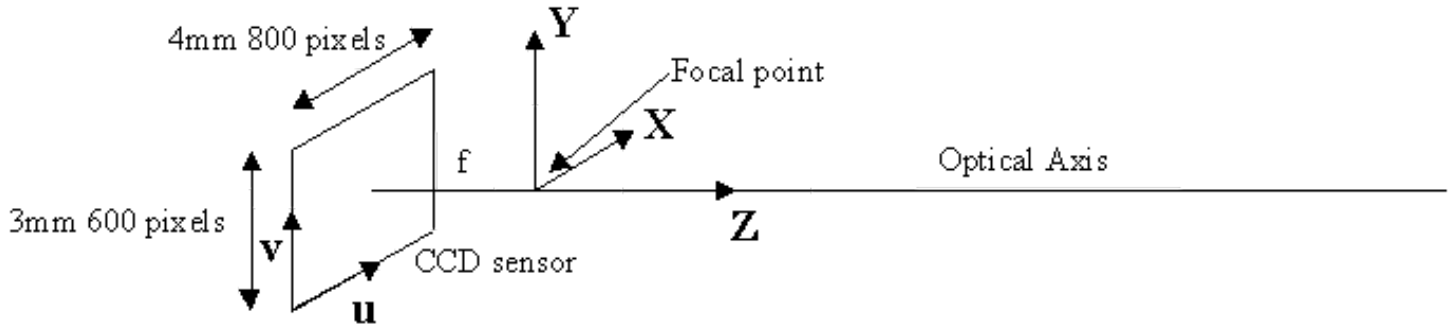
CSE527 Homework 3

Due Thu Oct 23, 2014

1. Pinhole camera model

The figure below shows a typical pinhole camera model. We construct a camera frame of reference centered at the focal point of the camera with the z axis along the principal optical axis of the device. The focal length of this camera, f, is 8mm, the CCD sensor on the focal plane is 4mm wide and 3mm tall as shown. The sensor is divided into a grid of 800 pixels in the x direction and 600pixels in the y direction. The center of the imaging array corresponds to the point where the optical axis cuts the focal plane. The pixel coordinates are indexed from the bottom right hand corner as shown.



a. {5 pts} Compute the matrix of intrinsic parameters which relates the coordinates of features given with respect to the camera frame to the pixel coordinates of the projection of that point on the sensor. That is give an A such that:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \propto A \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

b. {10 pts} Consider a wireframe cube 2 meters on side that is initially centered and axis-aligned with the cameras frame of reference; that is, initially the frame of reference of the cube and the frame of reference of the camera are initially coincident. Now consider what would happen if we first rotated the cube about the axis (3,4,-1) by 60 degrees then translated it by +10 meters along the z-axis (remember to normalize the axis to unit length). Use Matlab to compute the 4x4 matrix which represents the transformation from the blocks new frame of reference to the camera frame. Use the quaternion representation for the rotation.
c. {10 pts} Use your answers above to compute where the vertices of the cube would appear on the image in pixel coordinates. Plot these vertices in Matlab and connect the vertices by line segments using Matlab's line command.
d. {5 pts} Find out where the points at infinity corresponding to the x, y and z directions of the cubes new frame of reference would project onto the image.

2. Camera calibration

A fixed camera views a table of variable height, illuminated by a projector that projects a cross-pattern

onto the table. The system is to be calibrated by adjusting the height of the table to heights of precisely 50mm, 100mm, and 200mm. At each of these heights, the center of the cross-pattern is measured in images at the following positions (in pixels): (u, v) = (100, 250), (140, 340), and (200, 450), respectively. Notice that the projected ray of light has the same algebraic representation as the ray imaging to a camera. To simplify matters, choose a convenient world frame.

a. {5 pts} What are the
    a. perspective projection camera model and the
    b. projective camera model,

   specialized for this particular problem configuration? Give your answer as equations using homogeneous coordinates that show how a scene point is mapped to an image point.

a. {5 pts} In the projective camera model you defined in (a), how many degrees of freedom are there in order to calibrate this system?
b. {10 pts} Compute all of the calibration parameters for this problem (use your projective camera model from (a)). Hint: Convert the problem to the form Ax=b where x is a column vector of all the unknowns, and use Matlab to solve for the linear least squares solution. For example, in Matlab entering A = [1 2 3;4 5 6;7 8 0], then b = [366;804;351] and finally entering x = A\b will compute and print the solution x = 25.0 22.0 99.0. For this problem you ll need to compute the singular value decomposition (SVD) of A and print the eigenvector corresponding to the smallest eigenvalue; you can use the operators eig or eigs, e.g., [V,D]=eigs(A), for this purpose.
c. {10 pts} Recover the height of the table when the cross is measured in three new images at the following pixel coordinates. Show how you obtained your answers as well.
    a. (130, 310)
    b. (170, 380)
    c. (190, 300)

3. Image mosaicing

In general, two pictures taken by a camera that rotates about an axis will be related by a 2D projective transformation. You will implement a program that performs image mosaicing (read the Szeliski paper) and goes through the following steps:

a. Image acquisition: You can find a set of 4 test images from the former humanities building here. Alternatively, if you successfully complete the mosaic with your own images, you will receive extra credit {5 pts}. Remember to rotate your camera around a single axis. Here's an example of mosaics that can be created. All images are courtesy of Prof. Oliveira.
b. Image warping: Supposing that you have two input images, projectively warp img1 so that the portions of img1 that overlap with img2 are correctly aligned.
    a. {10 points} Write out the equation of the linear system that your implementation is solving. All coefficients & constants in the system must be expressed in terms of the image coordinates of the user-specified feature points. You may assume that the user selects only the minimum number of feature points necessary for warping.
    b. {10 points} Implement a function [A] = ComputeWarpMapping(img1pts, img2pts), where A is the projective transformation matrix. You can use SVD or Matlab's matrix division to solve it, but cannot use a built-in function to solve for the projection. To get good alignment results you need to choose your feature points with some care. Here are a few suggestions on what constitutes a 'good' set of feature points: While in theory 4 points are sufficient, you should choose more points, probably around 6-10 points. Try to distribute the feature points throughout the area of overlap between the two images. Avoid 'degenerative' point configurations (e.g., all points along a single line, all points very close to each other). Try to

choose feature points that are very distinctive from their surroundings, since they will make localization easy in the other image.

c. {10 points} Once the 2D projective map is computed, you can use it to warp image img1 so that it becomes aligned with img2. To do this, you need to implement a function [img1warp] = WarpImage(img1, A) that takes an image img1 and the matrix A describing the 2D projective warp and returns a new image, img1warp, that warps img1 according to A. You should use the backward mapping algorithm to implement this warp.

c. Merging the two Images {10 points}: To compute the final mosaic, you simply need to copy all pixels of img2 directly into a copy of img1warp. For every pixel in the region of overlap, use the average color between img1 and img2 for the pixel's final color.

4. Image mosaicing with SIFT (Extra credit)

{20 points} Implement a mosaic program similar to that in problem 3, but with SIFT descriptors instead of using user-defined points. You will detect SIFT points in the input images and use RANSAC to find matches between those points. After that, you can calculate the transformation matrix accordingly. For details, use this paper as guideline.

Submission

You will submit a report file which describes all the answers to the questions and Matlab codes you implemented. PDF format is preferred for the report format. Submit a compressed archive (e.g., zip, tar.gz, rar) that include all of your files.

You are free to include your codes in the report, but whether you do or don’t, you MUST submit all codes in separate Matlab .m files. Also, if you have a Matlab figure file that is large in size, do not submit the figure but instead hand in the Matlab code that can generate the figure.

Late submission policy: you will lose 10 points for every one day past deadline.