Emailed October 12, 2000
Due back October 18 2000

Homework 7: Camera calibration using vanishing points

This week we are looking at a way to calibrate a camera using vanishing points.

We are using the same cube and the same image as last week. As a reminder, the world coordinates of the cube vertices are

```
 2    2    2
-2    2    2
-2    2   -2
 2    2   -2
 2   -2    2
-2   -2    2
-2   -2   -2
 2   -2   -2
```

and the corresponding image points are

```
422  323
178  323
118  483
482  483
438   73
162   73
 78  117
522  117
```

1. Write a Matlab function that outputs the homogeneous coordinates of the 12 lines that are the images of the edges of the cube (these are lines in the image plane. Refer to slide 11 of the class on Projective Geometry, and use the Matlab function cross).

2. Find the homogeneous coordinates of the 3 vanishing points of the image of the cube. These are the intersections of the image lines corresponding to parallel edges of the cube (refer to slide 11 of Projective Geometry again for a method for finding intersections between lines. Refer to slide 36 of class 3 on cameras for a review on vanishing points).
NOTE: Using only 2 lines is OK. For a least square solution using 4 lines, write that the vanishing point belongs to 4 lines, and find P as the null vector for the 4x3 matrix of the system (cf . previous homework).

3. Each vanishing point is the image of a point at infinity of the form (**d**, 0), where **d** is a euclidean vector with 3 coordinates expressing the direction of a cube edge. Show that the coordinates of a vanishing point v can be expressed as **v = K R d**, where **K** is the calibration matrix and **R** is the rotation matrix between the camera and world coordinate system (use slides of Calibration class).

4. Express an edge direction **d** as a function of **K**, **R** and **v**.

5. The 3 directions of cube edges that give rise to the 3 vanishing points are mutually perpendicular, therefore the dot product between two directions is zero. Show that this condition leads to an equation in which the unknown is the calibration matrix **K**. Such an equation can be written for each of the 3 pairs of vanishing points.

Note: this equation expresses that the vanishing points belong to a conic which is the image of the so-called absolute conic.

6. The equation just found leads to nonlinear conditions between the elements of the matrix **K**, so we will not attempt to solve the system, but only verify that the matrix **K** found last week indeed is a solution. Verify that the equation above is verified for the vanishing points found in (2), for a calibration matrix in which the skew is zero, the 2 focal lengths are equal to 690, and the image center is at (300, 250).

NOTE: If you ever need to use this method, here is how to find **K**.
Let omega be the matrix that results from the combination of the inverse of **K** and its transpose in the result found in (5).
When only one focal length and the coordinates of the image center are the unknowns, omega has the form [a 0 b; 0 a c; b c 1] with a, b and c unknown. These unknowns can be found with a system involving 3 vanishing points v1, v2 and v3. The system is of the form A*[a;b;c] = rhs with

```
A = [v1(1)*v2(1) + v1(2)*v2(2), v1(3)*v2(1) + v1(1)*v2(3),  v1(3)*v2(2) +
v1(2)*v2(3);
       v1(1)*v3(1) + v1(2)*v3(2), v1(3)*v3(1) + v1(1)*v3(3),  v1(3)*v3(2) +
v1(2)*v3(3);
       v2(1)*v3(1) + v2(2)*v3(2), v2(3)*v3(1) + v2(1)*v3(3),  v2(3)*v3(2) +
v2(2)*v3(3)];
and rhs = [-1;-1;-1]
```

Then solving the system with `[a;b;c] = A \ rhs` provides omega.
Finally, the inverse of K can be found from omega by Cholesky decomposition (check the help for the Matlab function `chol`). **K** is found by inversion of that inverse and normalizing the result given the fact that K(3,3) must be equal to 1.
The result may not be as accurate as with the method described in the previous homework. In addition, it needs some modifications if one of the vanishing points is at infinity (which is the case with the image provided above). The method succeeds if the cube is positioned so that all vanishing points are finite, as is the case for the following cube image

```
cubeImage = [446, 333, 1; 204, 314, 1; 98, 462, 1; 454, 503, 1;...
             466, 75, 1; 191, 70, 1; 56, 111, 1; 489, 123, 1]
```