ECE 5273
HW 2 Solution

Spring 2010 Dr. Havlicek

Note: This document contains solutions in both Matlab and traditional C.

Analytical Portion of the Solution:

Initial world coordinates are specified for five out of the eight corners of the cube. From these data, it is clear that, in units of meters, the initial world coordinates of all eight corners are as shown in the table below.

Corner	X	Y	Z
P1	0.00	0.00	0.00
P2	0.00	0.00	0.05
P3	0.05	0.00	0.05
P4	0.05	0.00	0.00
P5	0.00	0.05	0.00
P6	0.00	0.05	0.05
P7	0.05	0.05	0.05
P8	0.05	0.05	0.00

The first motion applied to the cube is a rotation by an angle of $\theta = 30^\circ$ clockwise in the X-Z plane. Note that this rotation does not change the Y world coordinate of any point on the cube. Thus, to account for the rotation, we must update the X and Z world coordinates of all eight corners.

Let P be an arbitrary one of the eight corners. Suppose that P initially has world coordinates in the X-Z plane given by X_0 and Z_0. The initial position of P is depicted in Fig. 1 below. In polar coordinates, the point P is located in the X-Z plane at an angle with respect to the positive Z-axis of $\alpha = \arctan\frac{X_0}{Z_0}$ and a radius of $R = \sqrt{X_0^2 + Z_0^2}$.

After the first rotation, the new world coordinates of the point P in the X-Z plane are given by $P' = (X_1, Z_1)$, as also shown in Fig. 1. In polar coordinates, the point P' is located in the X-Z plane at an angle with respect to the positive Z-axis of $\alpha+\theta = \arctan\frac{X_1}{Z_1}$. The radius of the point P' is the same as that of point P and is given by $R = \sqrt{X_1^2 + Z_1^2} = \sqrt{X_0^2 + Z_0^2}$.

The new coordinates X_1, Z_1 may be found by applying simple geometry. Details for finding Z_1 are shown in Fig. 2. Consider a new pair of axes X' and Z' that are rotated by $\theta = 30^\circ$ clockwise with respect to the original X and Z axes. These rotated axes are shown in Fig. 2. With respect to the X' and Z' axes, the coordinates of point P' are X_0 and Z_0.

Note that $Z_1 = \overline{AB} - \overline{BF}$. From triangle ABC, it is clear that the length of line segment $\overline{AB}$ is given by $Z_0 \cos\theta$. Now, triangles AED and HGC have all three corresponding pairs of angles equal. These triangles are therefore similar. Moreover, $\overline{AE}$ and $\overline{CH}$ are opposite sides of a rectangle and are therefore equal. This implies that AED and HGC are not only similar; they are identical triangles. So $\overline{BF} = \overline{CG} = \overline{DE} = X_0 \sin\theta$. We have then that

$$Z_1 = \overline{AB} - \overline{BF} = Z_0 \cos\theta - X_0 \sin\theta. \tag{1}$$

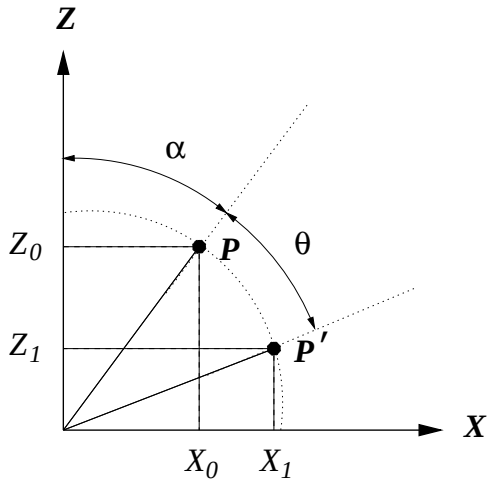

Figure 1: Geometry for the first rotation of the cube.

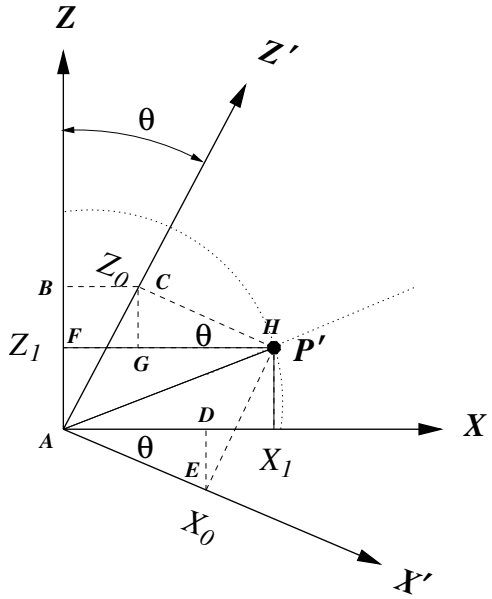

Figure 2: Details for finding Z_1.

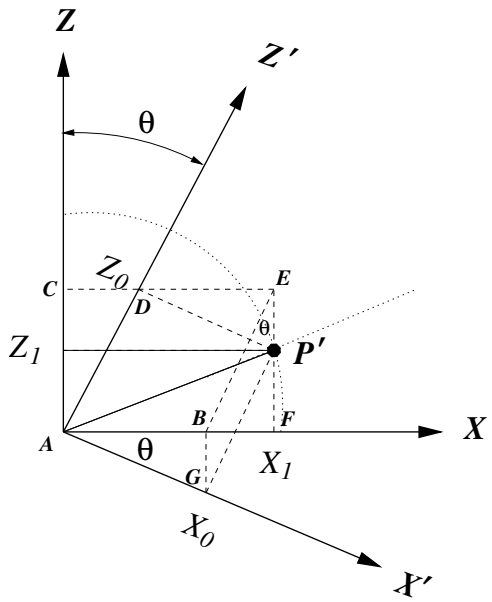

Figure 3: Details for finding X_1.

Details for finding X_1 are shown in Fig. 3. Note that $X_1 = \overline{AB} + \overline{BF}$. From triangle ABG, we have that $\overline{AB} = X_0 \cos\theta$.

Now, triangles ACD and EFB have all three pairs of corresponding angles equal and are therefore similar. Also, $\overline{AC} = \overline{FE}$, since they are opposite sides of a rectangle. Thus, triangles ACD and EFB have all pairs of corresponding sides equal. In particular, $\overline{BF} = \overline{CD} = Z_0 \sin\theta$. We have then that

$$X_1 = \overline{BF} + \overline{AB} = Z_0 \sin\theta + X_0 \cos\theta. \tag{2}$$

Writing (1) and (2) together, we obtain the coordinate transformation

$$\begin{aligned} Z_1 &= Z_0 \cos\theta - X_0 \sin\theta, & (3)\\ X_1 &= Z_0 \sin\theta + X_0 \cos\theta. & (4) \end{aligned}$$

Therefore, for each of the eight corners of the cube, the X and Z world coordinates can be updated to account for the first rotation by applying (3) and (4) to the initial X-Z coordinates of the corner.

The second motion applied to the cube is a rotation by an angle of $\psi = 20°$ counterclockwise in the Y-Z plane. With respect to the positive Y-axis, this is a rotation *clockwise* by $\psi = -20°$. Note that this rotation does not change the X world coordinate of any point on the cube. Thus, to account for the rotation, we must update the Y and Z world coordinates of all eight corners.

Let P be an arbitrary one of the eight corners. Suppose that, prior to the second rotation, P has world coordinates in the Y-Z plane given by Y_0 and Z_0. The position of P prior to the second rotation is depicted in Fig. 4 below. In polar coordinates, the point P is located in the Y-Z plane at an angle with respect to the positive Y-axis of $\beta = \arctan\frac{Z_0}{Y_0}$ and a radius of $R = \sqrt{Z_0^2 + Y_0^2}$.

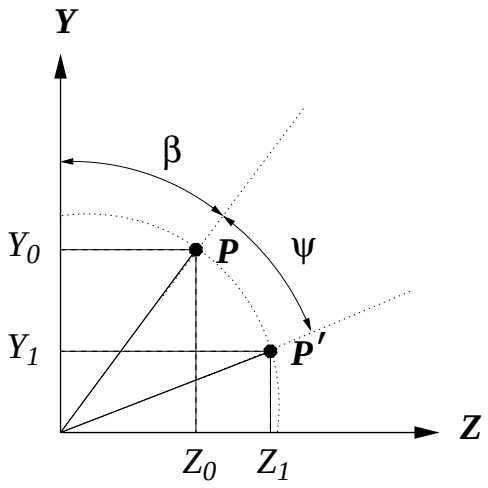

Figure 4: Geometry for the second rotation of the cube.

After the second rotation, the new world coordinates of the point P in the Y-Z plane are given by $P' = (Y_1, Z_1)$, as shown in Fig. 4. Applying arguments identical to those used for finding Z_1 and X_1 in the first rotation above, we obtain the coordinate transformation

$$Y_1 = Y_0 \cos\psi - Z_0 \sin\psi, \quad (5)$$
$$Z_1 = Y_0 \sin\psi + Z_0 \cos\psi. \quad (6)$$

Therefore, for each of the eight corners of the cube, the Y and Z world coordinates can be updated to account for the second rotation by applying (5) and (6) to the Y-Z coordinates obtained after the first rotation.

The final motion of the cube is a translation of 1.00 meters in the direction of the positive Z-axis. For each corner, this motion is accounted for by adding 1.00 to the Z-coordinate obtained after the second rotation.

Once the final coordinates are known for each of the eight corners, the projection equations for the pinhole camera model can be used to project the eight corners onto the image plane. The camera image is then obtained by projecting straight lines in the real world (the cube edges) to straight lines on the focal plane. This forms a closed polygon where some edges are occluded (hidden).

Matlab Solution:

- Image (numerical values of the coordinates are given in the C solution):

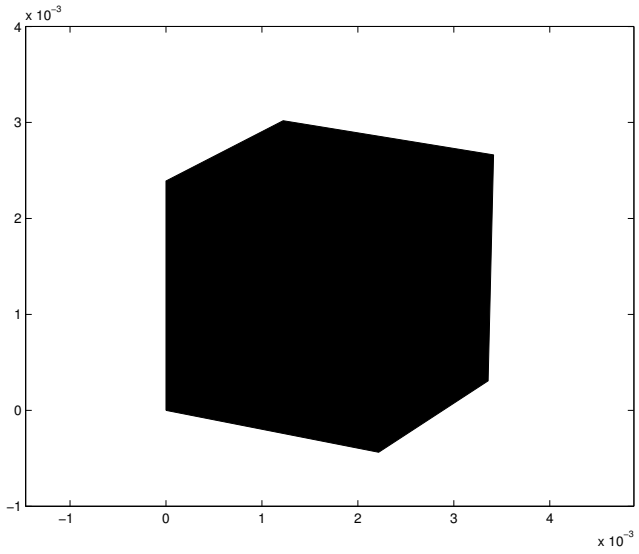

Figure 5: Image of the cube that is obtained on the camera focal plane.

- **Matlab m-file listing:**

```
%
% hw02.m
%
% 02/06/03 jph
%
%    03/06/05: corrected error in calculation of xx and yy.
%

%
% Initialize the world coordinates of the eight cube corners
%
P(1).X = 0.00; P(1).Y = 0.00; P(1).Z = 0.00;
P(2).X = 0.00; P(2).Y = 0.00; P(2).Z = 0.05;
P(3).X = 0.05; P(3).Y = 0.00; P(3).Z = 0.05;
P(4).X = 0.05; P(4).Y = 0.00; P(4).Z = 0.00;
P(5).X = 0.00; P(5).Y = 0.05; P(5).Z = 0.00;
P(6).X = 0.00; P(6).Y = 0.05; P(6).Z = 0.05;
P(7).X = 0.05; P(7).Y = 0.05; P(7).Z = 0.05;
P(8).X = 0.05; P(8).Y = 0.05; P(8).Z = 0.00;
%
% First rotation - update X & Z coordinates of all eight points
%
Theta = 30.0 * 2.0 * pi / 360.0;
for i=1:8
  Zold = P(i).Z;
  Xold = P(i).X;
  P(i).Z = cos(Theta) * Zold - sin(Theta) * Xold;
  P(i).X = sin(Theta) * Zold + cos(Theta) * Xold;
end
%
```

```
% Second rotation - update Y & Z coordinates of all eight points
%
Theta = -20.0 * 2.0 * pi / 360.0;
for i=1:8
  Yold = P(i).Y;
  Zold = P(i).Z;
  P(i).Y = cos(Theta) * Yold - sin(Theta) * Zold;
  P(i).Z = sin(Theta) * Yold + cos(Theta) * Zold;
end
%
% Translation - update Z coordinates of all eight points
%
for i=1:8
  P(i).Z = P(i).Z + 1.0;
end
%
% Project all eight points into the image plane
%
f = 0.05;
for i=1:8
  P(i).x = f/P(i).Z * P(i).X;
  P(i).y = f/P(i).Z * P(i).Y;
  P(i).z = 0.0;
end
%
% plot the resulting image
%
order = [1 5 6 7 3 4]; % this ordering enables 'fill' to deal
                       % with the occluded edges
xx = zeros(6,1);
yy = zeros(6,1);
for i=1:6
  xx(i) = P(order(i)).x;
  yy(i) = P(order(i)).y;
end
fill(xx,yy,[0 0 0]);
axis([-0.001 0.004 -0.001 0.004],'equal');
print -deps ML02.eps
print -dtiff ML02.tif
print -dps ML02.ps
```

C Solution:

- Images:

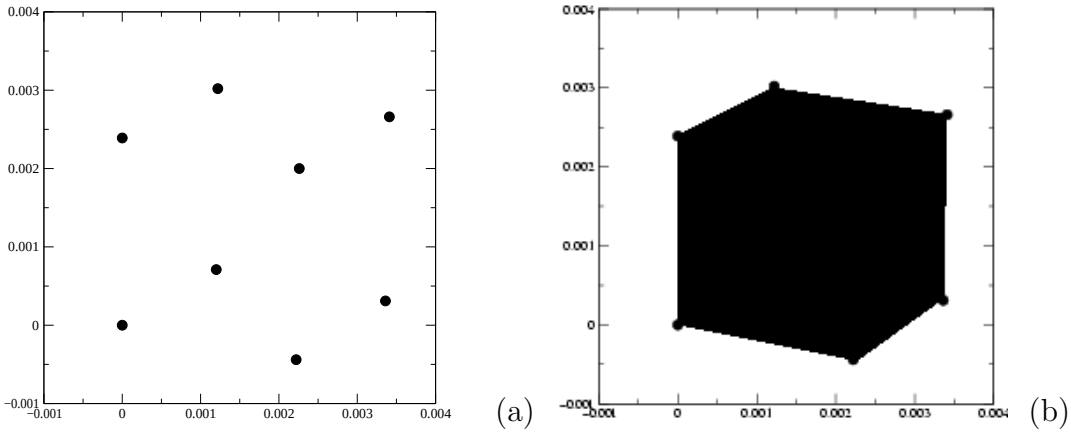

Figure 6: (a) Projections of the eight cube corners on the focal plane. (b) Image of the cube that is obtained on the camera focal plane.

- Coordinates of the points after each step:

```
After First Rotation:
P1: X=  0.00000         Y=  0.00000         Z=  0.00000
P2: X=  0.02500         Y=  0.00000         Z=  0.04330
P3: X=  0.06830         Y=  0.00000         Z=  0.01830
P4: X=  0.04330         Y=  0.00000         Z= -0.02500
P5: X=  0.00000         Y=  0.05000         Z=  0.00000
P6: X=  0.02500         Y=  0.05000         Z=  0.04330
P7: X=  0.06830         Y=  0.05000         Z=  0.01830
P8: X=  0.04330         Y=  0.05000         Z= -0.02500

After Second Rotation:
P1: X=  0.00000         Y=  0.00000         Z=  0.00000
P2: X=  0.02500         Y=  0.01481         Z=  0.04069
P3: X=  0.06830         Y=  0.00626         Z=  0.01720
P4: X=  0.04330         Y= -0.00855         Z= -0.02349
P5: X=  0.00000         Y=  0.04698         Z= -0.01710
P6: X=  0.02500         Y=  0.06179         Z=  0.02359
P7: X=  0.06830         Y=  0.05324         Z=  0.00010
P8: X=  0.04330         Y=  0.03843         Z= -0.04059

After Translation:
P1: X=  0.00000         Y=  0.00000         Z=  1.00000
P2: X=  0.02500         Y=  0.01481         Z=  1.04069
P3: X=  0.06830         Y=  0.00626         Z=  1.01720
P4: X=  0.04330         Y= -0.00855         Z=  0.97651
P5: X=  0.00000         Y=  0.04698         Z=  0.98290
P6: X=  0.02500         Y=  0.06179         Z=  1.02359
```

```
      P7: X=  0.06830         Y=  0.05324         Z=  1.00010
      P8: X=  0.04330         Y=  0.03843         Z=  0.95941

      In the Image Plane:
      P1: x=  0.00000         y=  0.00000
      P2: x=  0.00120         y=  0.00071
      P3: x=  0.00336         y=  0.00031
      P4: x=  0.00222         y= -0.00044
      P5: x=  0.00000         y=  0.00239
      P6: x=  0.00122         y=  0.00302
      P7: x=  0.00341         y=  0.00266
      P8: x=  0.00226         y=  0.00200
```

- **C program listing:**

```
/*
 * hw02-2.c:
 *
 *  For a cube specified in three-space by the coordinates of its vertices,
 *  perform two rotations and one tanslation.  Then project the cube into
 *  the image plane.
 *
 *  Print out intermediate answers for grading
 *
 * 2/15/2002 jph
 *
 */

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>

/*
 * Type for a point in upright pinhole camera model (NOTES p. 1.17)
 */
struct point {
  float     X;        /* X-coordinate in the real world */
  float     Y;        /* Y-coordinate in the real world */
  float     Z;        /* Z-coordinate in the real world */
  float     x;        /* x-coordinate in the image plane */
  float     y;        /* y-coordinate in the image plane */
};
typedef struct point point;
typedef point *Ppoint;


/*--------------------------------------------------------------------------*/
/*   MAIN                                                                   */
/*--------------------------------------------------------------------------*/

main(argc,argv)

  int     argc;
  char   *argv[];
{

  int      i;                    /* loop counter */
  point    P[8];                 /* array of cube corners */
  float    f;                    /* camera focal length in meters */
```

```
 double   Theta;                /* rotation angle */
 double   Xold;                 /* old X coordinate of a point */
 double   Yold;                 /* old Y coordinate of a point */
 double   Zold;                 /* old Z coordinate of a point */

/*
 * Initialize the world coordinates of the eight cube corners
 */
 P[0].X = 0.00; P[0].Y = 0.00; P[0].Z = 0.00;
 P[1].X = 0.00; P[1].Y = 0.00; P[1].Z = 0.05;
 P[2].X = 0.05; P[2].Y = 0.00; P[2].Z = 0.05;
 P[3].X = 0.05; P[3].Y = 0.00; P[3].Z = 0.00;
 P[4].X = 0.00; P[4].Y = 0.05; P[4].Z = 0.00;
 P[5].X = 0.00; P[5].Y = 0.05; P[5].Z = 0.05;
 P[6].X = 0.05; P[6].Y = 0.05; P[6].Z = 0.05;
 P[7].X = 0.05; P[7].Y = 0.05; P[7].Z = 0.00;

/*
 * First rotation - update X & Z coordinates of all eight points
 */
 Theta = (double)30.0 * (double)2.0 * M_PI / (double)360.0;
 for (i=0; i < 8; i++) {
   Zold = (double)P[i].Z;
   Xold = (double)P[i].X;
   P[i].Z = (float)(cos(Theta)*Zold - sin(Theta)*Xold);
   P[i].X = (float)(sin(Theta)*Zold + cos(Theta)*Xold);
 }

/*
 * Report
 */
 printf("\nAfter First Rotation:\n");
 for (i=0; i < 8; i++) {
   printf("P%d: X=%9.5f         Y=%9.5f         Z=%9.5f\n",
          i+1,P[i].X,P[i].Y,P[i].Z);
 }
 printf("\n\n");

/*
 * Second rotation - update Y & Z coordinates of all eight points
 */
 Theta = (double)-20.0 * (double)2.0 * M_PI / (double)360.0;
 for (i=0; i < 8; i++) {
   Yold = (double)P[i].Y;
   Zold = (double)P[i].Z;
   P[i].Y = (float)(cos(Theta)*Yold - sin(Theta)*Zold);
   P[i].Z = (float)(sin(Theta)*Yold + cos(Theta)*Zold);
 }

/*
 * Report
 */
 printf("\nAfter Second Rotation:\n");
 for (i=0; i < 8; i++) {
   printf("P%d: X=%9.5f         Y=%9.5f         Z=%9.5f\n",
          i+1,P[i].X,P[i].Y,P[i].Z);
 }
 printf("\n\n");

/*
 * Translation:  update Z coordinates of all eight points
 */
 for (i=0; i < 8; i++) {
   P[i].Z += (float)1.0;
```

```
  }

 /*
  * Report
  */
  printf("\nAfter Translation:\n");
  for (i=0; i < 8; i++) {
    printf("P%d: X=%9.5f         Y=%9.5f         Z=%9.5f\n",
           i+1,P[i].X,P[i].Y,P[i].Z);
  }
  printf("\n\n");

 /*
  * Project all eight points into the image plane
  */
  f = (float)0.05;
  for (i=0; i < 8; i++) {
    if (P[i].Z == (float)0.0) {
      printf("\nError: point %d has Z=0.0\n\n",i);
      exit(-1);
    } else {
      P[i].x = f/P[i].Z * P[i].X;
      P[i].y = f/P[i].Z * P[i].Y;
    }
  }

 /*
  * Report
  */
  printf("\n\n");
  for (i=0; i < 8; i++) {
    printf("P%d: x=%9.5f         y=%9.5f\n",i+1,P[i].x,P[i].y);
  }
  printf("\n\n");
  return;
} /*----------------  Main  ----------------------------------------------*/
```