ESE 568 Computer Vision
ECE, SBU, M. Subbarao
Project 1. Image Formation : Computer Animation and Augmented Reality

Draft 1.0,

Image formation in a pin-hole camera: Mapping from 3D object coordinates to 2D image coordinates under 3D rigid-body motion. (See pages 31-52 in the Szeliski's text book).

COMPUTER ANIMATION:

ROTATING AND TRANSLATING CUBE WITH TEXTURE-MAP.

1. **You are given a program with some parts of the code removed. Fill in the missing code so that the program works. There will be a quiz on this project after the submission of this project. This quiz is to make sure that you understand all parts of the project and that you completed the project yourself.**

A rigid wire-frame cube and its texture-map for one of the surfaces are given. The cube is specified in a World Coordinate System (WCS) fixed to the cube.

The origin of the world coordinate system is at T0 in the camera coordinate system (CCS) and the WCS is rotated by an angle theta0 around a given axis along the unit vector N0 passing through the origin of the origin of the WCS.

In the project, take theta0=0, and N0 is passing through V1 and V8, and pointing towards V8 (i.e. , N0 is parallel to V8-V1).

Use this to compute the initial rotation matrix R0.

The translation of the WCS wrt the CCS with time t in seconds is given by

```
T=T0+ velocity*t + 0.5 * acc * t * t;
```

T0, velocity `velocity`, and acceleration `acc` are translational vectors given in the CCS.

And the rotation angle theta of the WCS wrt the CCS along the same axis (translated with V1,V8) with time t in seconds is given by

theta = theta0 + w0*t

where theta0 and w0 are given. The above equation specifies angle theta as a function of time t with initial angle theta0 added to rotational angle w0*t where w0 is the uniform rotational/angular velocity. Note that the axis of rotation N0 always remains the same and always passes through the origin of the WCS which is translating as T wrt to the CCS.

Draw a wire-frame image of the cube at time t=0.

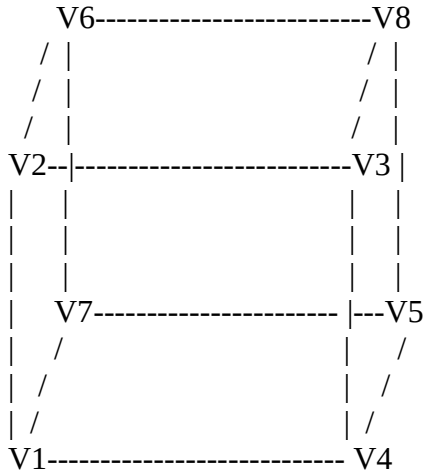
Join the lines
(v1,v2), (v2,v3), (v3,v4), (v4,v1),
(v7,v6), (v6,v8), (v8,v5), (v5,v7),
(v1,v7), (v2,v6), (v3,v8), (v4,v5).
By convention, 3D vectors are denoted with upper case letters and their corresponding 2D image vectors are represented in lower case.

The 3x3 camera parameters matrix K is given. Generate a sequence of images as a function of time t and display them as a sequence of image frames to produce the effect of a "video".

The camera matrix K of a digital camera is given by Eq. 2.59 on page 52 in the Szeliski's text book. The perspective projection image formation is given by Eq. 2.55 for any point Pw at (x,y,z). The image center is at (0,0) [not (W/2,H/2) as in the text book]. The rotation matrix is given by Eq. 2.34 on page 42. This rotation matrix is derived for a rotation axis passing through the origin of the WCS. A cube of size 5 mm X 5 mm X 5 mm has three of its edges parallel to the three axes X, Y, and Z.

If the cube is rotated by theta= Θ degrees counter-clockwise around an axis that passes through the vector **v8-v1** (and that points from **v1** to **v8** which determines the direction of rotation). Write an expression for computing the numerical values of the image coordinates of the image point of vertex **vi**, (i=1,2,…,8). All vectors and matrices like K and R must be written in explicit matrix form and computed.

Note that, the unit vector along the axis of rotation is **n=(v8-v1)/||v8-v1||**. Then you can get **[n]x** in Eq. 2.32, Also, **[n]x^2** in Eq. 2.34 is the product of **[n]x** with **[n]x** itself. This Eq. 2.34 gives the rotation matrix R.



1. Generate a gray-level image of size 600x600 with pixel size p= 0.010 mm which shows the lines joining the edges between the vertices vi (i= 1 to 8) of the cube. A 3D line can be shown to project

onto the image plane as 2D line in a pin-hole camera. Use a value of 0 for the background and 255 for pixels on the lines. The edges connect the following pairs of vertices.

(v1,v2), (v2,v3), (v3 , v4), (v4, v1),

(v7, v6), (v6,v8), (v8,v5),(v5,v7)

(v1,v7), (v2,v6), (v3,v8), (v4,v5)

Note that you can compute the pixels lying on the image line connecting two image points q1(x1,y1) and q2(x2,y2) as follows. A vector equation for the line is
q = q1 + c * u where u is a unit vector along q2-q1, and c is a scalar number in the range 0 to d where d is the length of the vector q2-q1, in units of pixels. The row and column indices (i,j) are computed as follows. Let for an image of size MxN, q=(x,y). Then, i=M/2 – y/0.010, and j = N/2+x/0.010.

2. Suppose that the cube is rotating with a uniform angular velocity of **w0** degrees per second, and simultaneously translating with a uniform velocity of **velocity** mm per sec, generate the image of the wire frame cube with the texture map on one face, as a function of time and display the image sequence as a function of time.

3. Texture map of one of the cube faces is:

In the following MATLAB code, fill in the missing pieces of code. There will be quiz to test that you understand all parts of this code, and that you completed the project yourself.

AUGMENTED REALITY

The video of a real stationary 3D scene is given as a 600x600 gray-level image that does not change with time. This background video image of a real 3D scene needs to be augmented for a viewer to show a rotating and translating cube in the foreground of the 3D scene. Superimpose the image sequence generated in your previous section with a given foreground image. All pixels of the background image that are not occluded by the cube must retain their original values in the 3D scene, and all pixels that are occluded by the cube must have the pixel values generated from the cube.

AR for Stereo Vision

In addition to the camera in the previous section, a binocular stereo camera is created by placing a second camera. The all the axes in the CCS of the second camera is perfectly aligned with the corresponding axes of the first camera with the only difference being that the origin of the CCS of the second camera is located at (10, 0, 0) mm. Note that 10 mm is the baseline distance along the X-axis of the first camera.

Compute the animation sequence with the output of the first camera shown in red color, and the output of the second camera shown in blue color. The output of both cameras must be shown on the same single output color image frame. When this color image frame is viewed with a Red-Blue eye-glasses on the left and right eyes respectively, it would produce the perception of a 3D cube translating and rotating in 3D space.

In the following MATLAB code, fill in the missing pieces of code. There will be quiz to test that you understand all parts of this code, and that you completed the project yourself.

In the following MATLAB code, fill in the missing pieces of code. There will be quiz to test that you understand all parts of this code, and that you completed the project yourself.

```
%define 8 points of the cube in world coordinate

length = 10;
V1= [ 0 0 0 ];
V2= [ 0 length 0 ];
V3= [ length length 0 ];
V4= [ length 0 0 ];
V5= [ length 0 length ];
V6= [ 0 length length ];
V7= [ 0 0 length ];
V8= [ length length length ];


% Find the unit vector u81 corresponding to the axis of rotation
which is along (V8-V1).

From u81, compute the 3x3 matrix N in Eq. 2.32 used for
computing the rotation matrix R in eq. 2.34

??????????????????

T0 = [ -20 -25 500 ]; % origin of object coordinate system in mm
```

```
%T0 = [ -30 -20 500 ]; % origin of object coordinate system


%set given values

f=40; %focal length in mm
%f=input('f=');

Initialize the 3x3 camera matrix K given the focal length

K= ??????????

velocity=[2  9  7 ]; % translational velocity

theta0=0;

w0=20;% angular velocity in deg/sec

p=0.01;%pixel size(mm)
Rows = 600; %image size
Cols = 600;  % image size
A = zeros(Rows, Cols); %output image
r0= round(Rows/2);
c0 = round(Cols/2);


% You are given a rectangle/square in 3D space specified by its
% corners at 3D position vectors V1, V2, V3, V4.
% You are also given a rectangular/square graylevel image
% tmap of size r x c.
% This image is to be "painted" on the 3D rectangle/square, and
% for each pixel at position (i,j),
% the corresponding 3D coordinates
% X(i,j), Y(i,j), and Z(i,j), should be computed,
% and that 3D point is
% associated with the brightness given by tmap(i,j).
%
% Find the unit vectors corresponding to u21=(V2-V1)/|(V2-V1)|
% and u41= (V4-V1)/|(v4-V1), and compute X(i,j), Y(i,j), and
Z(i,j).

% Compute the unit vector u21 along (V2-V1) and
% Compute the unit vector u41 along (V4-V1) and

h=? % height = distance from v2 to v1
w=? % width = distance from v4 to v1
u21=??????????????
```

```
u41=??????????????


% For each pixel of texture map, compute its (X,Y,Z) values
%
tmap = imread('einstein50x50v.jpg'); % texture map image
[ r c ] = size(tmap);
X=zeros(r,c);
Y=zeros(r,c);
Z=zeros(r,c);

for i = 1 : r
    for j = 1 :
%incorrect: p1  = V1 + (i-1)* u41* (h/r)+ (j-1)* u21*(w/c);
          p1  = V1 + (i-1)* u21* (h/r)  + (j-1)* u41*(w/c);

         X(i,j)=p1(1);

         Y(i,j)=  ??
         Z(i,j)=??;
      end
end

acc = [ 0.0  -0.80 0 ];  %acceleration

for t=0:0.2:24    % Generate a sequence of images
                  % as a function of time

    theta=theta0+w0*t;
    T=T0+ velocity*t + 0.5 * acc * t * t;

% Compute the Rotation matrix from N and theta

    R= ??????????????

    %find the image position of vertices

    v=Map2Da(K,R,T,V1);
    v1 = MapIndex(v,c0,r0,p);
    v=Map2Da(K,R,T,V2);
    v2 = MapIndex(v,c0,r0,p);


??????????????????????????

    % Draw edges of the cube
```

```
    A = zeros(Rows, Cols);
    A = Line(A, v1,v2);

??????????????????????????

    % Add texture map to one face.
   for i = 1 : r
       for j = 1 : c
           p1  = [ X(i,j) Y(i,j) Z(i,j) ];
```

%Find the row and column indices (ir,jr) in integers that give
%the image position of point p1 in A.
% Use the same method as for the corners of the cube above.

```
 ??????????????????????????

           if((ir>0)&&(jr>0) && (ir<=Rows) && (jr<=Cols))
               A(ir,jr)=tmap(i,j);
           end
       end   % In a general case, you may need to
             % fill up gaps in A(ir,jr)
             % through interpolation. But, in this project,
             % you can skip interpolation. The output will not
             % look nice due to the gaps.
 end

    A=mat2gray(A);
    imshow(A);
%   pause
% pause if you want to display frame by frame
% and press return to display the next frame

end



%function for rotation and translation
function [ v ] =Map2Da( K,R,T,Vi)
    P=K*[R T']*[Vi 1]';
    w1=P(3,1);
    v(1)=P(1,1)/w1;

    v(2)= ?????????????;
```

```
end


%function for mapping image coordinates in mm to
% row and column index of the image, with pixel size p mm and
% image center at (r0,c0)
function [ v ] =MapIndex( u,c0,r0,p )
    v(1)= round(r0-u(2)/p);
    v(2)=round(c0+u(1)/p);
end


%function for line
% Draw line from v1 to v2 in image A
function [A] = Line(A, v1, v2)

d=sqrt((v1-v2)*(v1-v2)');

ui=(v2(1)-v1(1))/d;
uj=(v2(2)-v1(2))/d;

i=v1(1);
j=v1(2);
[ rows cols ] = size(A);
for K=0:round(d)
    i=i+ui;
    j=j+uj;

    ir=round(i);
    jr=round(j);

    if((ir>0)&&(jr>0) && (ir<rows) && (jr<cols))
       A(ir,jr)=255;
    end

end

end
```