Florida Institute of Technology

Harris Institute for Assured Information

CSE 5280
Computer Graphics
Spring 2016

Class Assignment-03
(Space Carving Animation)

Student Name: **Jay Sandeepkumar Modi**

Student ID: **902292667**

Professor:

Dr. Eraldo Ribeiro

eribeiro@cs.fit.edu

Florida Institute of Technology

150 W. University Blvd, Melbourne, FL - 32901

http://www.fit.edu

Space Carving Animation:

Code:

```
% Demonstrates the projection of a set of 3-D points onto an image. The
% projection matrix was obtained from camera calibration.

tic;
% clean up memory and close all figures
close all;
clear all;

% Pose number
imNumber = '0112';

run=0;

if (run == 1 )
    minX = -0.3;
    minY =  0.1;
    minZ = -0.3;
    maxX =  0.2;
    maxY =  1.7;
    maxZ =  0.2;
    step = 0.03;
else
    minX = -2.0;
    minY = -2.0;
    minZ = -2.0;
    maxX =  2.0;
    maxY =  2.0;
    maxZ =  2.0;
    step = 0.1;
end

% These are the first frames on Cameras 1 and 2
Cam(1).im =
rgb2gray(imread(strcat('silhouettes/Silhouette1_',imNumber,'.png')));
Cam(2).im =
rgb2gray(imread(strcat('silhouettes/Silhouette2_',imNumber,'.png')));
Cam(3).im =
rgb2gray(imread(strcat('silhouettes/Silhouette3_',imNumber,'.png')));
Cam(4).im =
rgb2gray(imread(strcat('silhouettes/Silhouette4_',imNumber,'.png')));
Cam(5).im =
rgb2gray(imread(strcat('silhouettes/Silhouette5_',imNumber,'.png')));
Cam(6).im =
rgb2gray(imread(strcat('silhouettes/Silhouette6_',imNumber,'.png')));
Cam(7).im =
rgb2gray(imread(strcat('silhouettes/Silhouette7_',imNumber,'.png')));
Cam(8).im =
rgb2gray(imread(strcat('silhouettes/Silhouette8_',imNumber,'.png')));
```

```
object_point=[];

% Number of Cameras
noc = 8;

% Sample 3-D points within a cube shape centered at the origin
    for X = minX : step : maxX
        for Y = minY : step : maxY
            for Z = minZ : step : maxZ

                hit=0;
                % Display the projection of cube points as seen from
                        Cameras 1 to 8
                for iCam = 1 : noc
                    % Obtain projection matrix for camera iCam
                    P = getProjMatrix( iCam );

                    % Project 3-D points to image points
                    x = P * [ X Y Z 1 ]';

                    % Transform homogeneous coords into cartesian
                    u = x(1)/x(3);
                    v = x(2)/x(3);

                    if ( v <= size(Cam(iCam).im,1) && u <=
                              size(Cam(iCam).im,2) && 1 <= u && 1 <= v)
                        if( (Cam(iCam).im(round(v),round(u))) ~= 0 )
                            hit = hit + 1;
                        end
                    end
                end

                % Storing points if a point is hit by all the cameras.
                if(hit == noc)
                   temp = [ X Y Z ]';
                   object_point = [object_point, temp ];
                end
            end
        end
    end

%display(object_point);
minX = min(object_point(1,:));
minY = min(object_point(2,:));
minZ = min(object_point(3,:));
maxX = max(object_point(1,:));
maxY = max(object_point(2,:));
maxZ = max(object_point(3,:));

object_point=object_point';
```

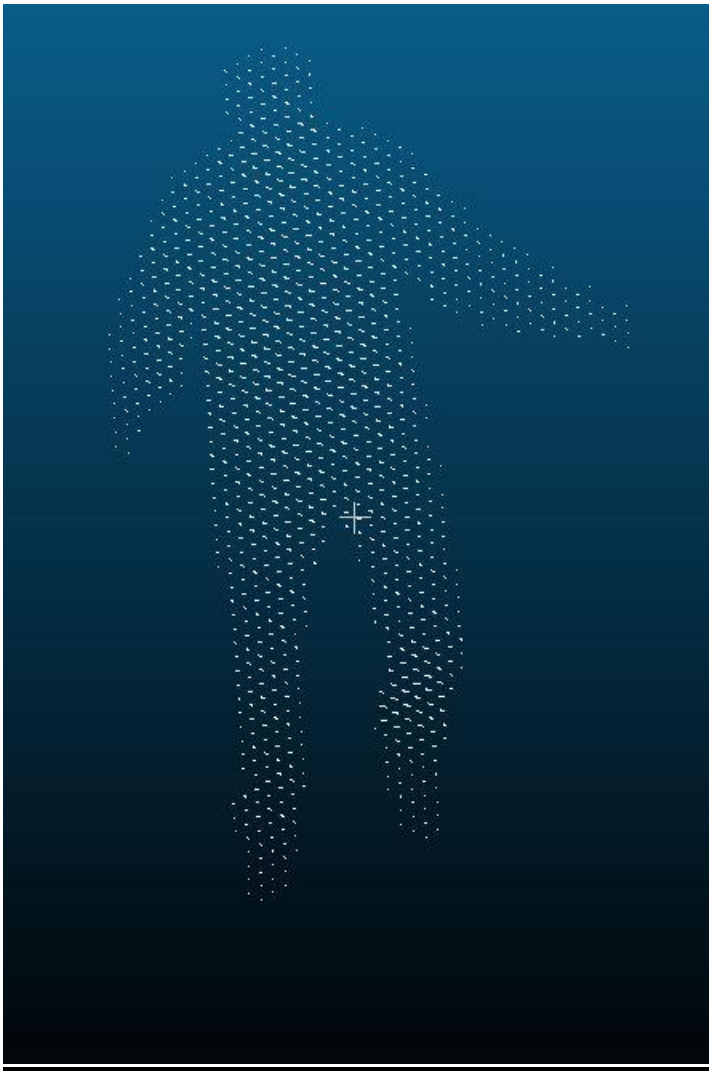
```
function P = getProjMatrix(i)
% Projection matrix for Camera i
%
% Projection matrices are hardcoded for simplicity. But, we could just
read
% them directly from the calibration files.

   Calib(1).P = [ 1.483215e+03 -7.953666e+02 -9.153119e+02 4.046004e+03; ...
                  -5.395400e+01 -1.719494e+03  3.606972e+02 3.961539e+03; ...
                  -6.991278e-02 -9.069575e-01 -1.063456e+00 4.753082e+00 ];

    Calib(2).P = [ 1.677218e+03 -7.084734e+02 5.732087e+02 5.171564e+03; ...
                  -1.814967e+02 -1.743858e+03 3.993065e+00 4.314523e+03; ...
                   8.810557e-01 -7.603295e-01 -7.786652e-01 6.074740e+00 ];

    Calib(3).P = [  9.269854e+02 -7.025415e+02 1.509225e+03 4.448627e+03; ...
                   -1.922770e+02 -1.737088e+03 -7.511179e+01 4.261084e+03;...
                    1.152985e+00 -7.936465e-01 -3.782901e-02 5.152725e+00 ];

    Calib(4).P = [ -3.529096e+02 -6.068026e+02 1.769746e+03 4.453332e+03; ...
                   -5.880053e+01 -1.765468e+03 -8.726954e+01 4.527056e+03;...
                    8.825374e-01 -7.435539e-01 7.937028e-01 6.003132e+00 ];

    Calib(5).P = [ -1.532662e+03 -7.698217e+02 8.246429e+02 3.787523e+03; ...
                    2.544591e+01 -1.720718e+03 -4.054952e+02 3.874761e+03;...
                    1.776188e-02 -9.415247e-01 1.036173e+00 4.695420e+00 ];

    Calib(6).P = [ -1.732989e+03 -5.827524e+02 -6.096631e+02 4.869964e+03;...
                    8.629205e+01 -1.771143e+03 -1.076821e+02 4.426927e+03;...
                   -9.177131e-01 -7.547954e-01 7.410083e-01 6.032102e+00 ];

    Calib(7).P = [ -8.914168e+02 -6.960165e+02 -1.548135e+03 4.017688e+03;...
                    2.466475e+02 -1.750562e+03 2.269384e+01 4.171231e+03; ...
                   -1.119922e+00 -8.401569e-01 2.586530e-04 5.181506e+00 ];

    Calib(8).P = [  4.011197e+02 -5.661652e+02 -1.788971e+03 4.441950e+03;...
                    1.039635e+02 -1.761553e+03 4.867437e+01 4.503671e+03; ...
                   -8.417806e-01 -7.415664e-01 -8.374394e-01 6.065466e+00 ];

    P = Calib( i ).P;

return
```

Steps followed:

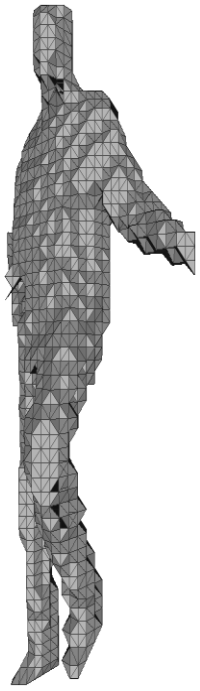
1. Take any pose number (`imNumber`) and `run` = 0 (Range = -2 to 2 and steps = 0.1), then execute the above program.
2. Now, get the value of `minX, minY, minZ, maxX, maxY, and maxZ.`
3. Now, update `run` = 1 (Range changes according to the min-max value of X, Y, and Z, and step = 0.03), and run the program again.
4. Save the object points in .txt file from the workspace.
5. Now, import this .txt file in MeshLab tool.
6. Following the procedure to create mesh from the object points.
7. Repeat steps 1-6 for all poses.
8. Save output by taking snapshot, and create animation.

Result:

Object points:



Mesh:



Note: Output animation is stored in output.gif, which can be found in zip file.