CSCI 574 Hw#4 : SVD and 3d object from images of its multiple views

Name: Manan Vyas

USC-ID : 7483-8632-00

Email: mvyas@usc.edu

Source Code :

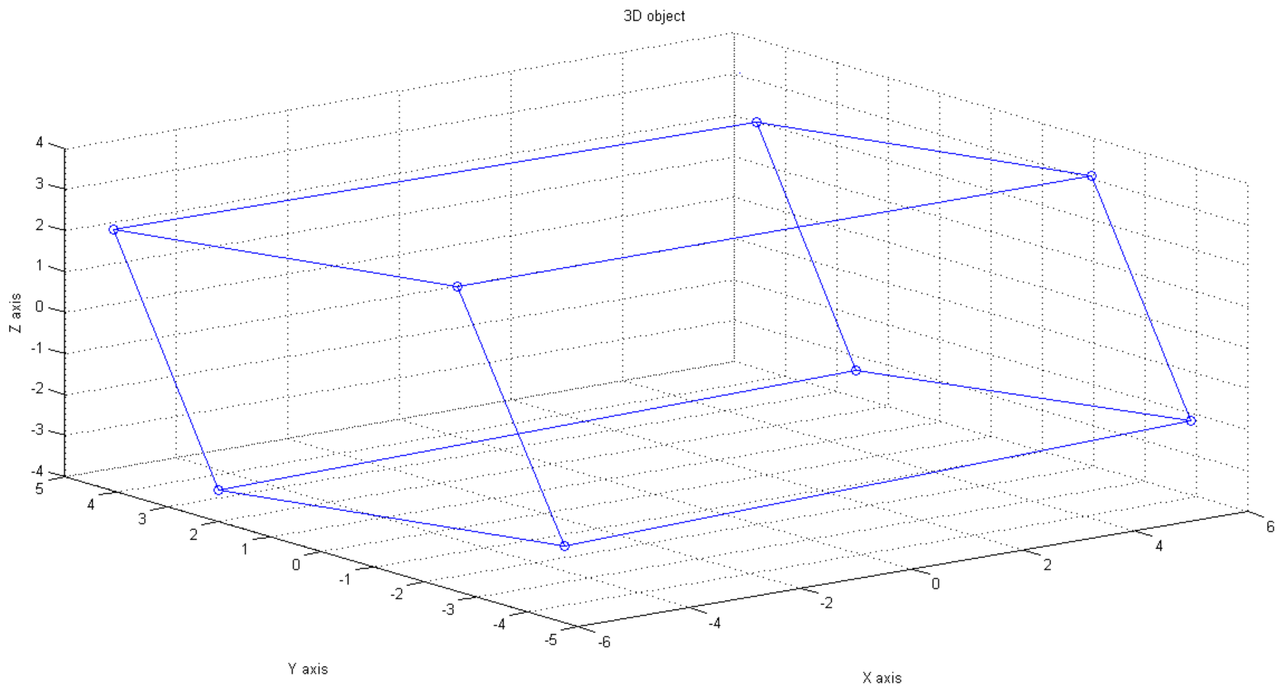
```
/* *******
    Name : Manan Vijay Vyas
    USCID: 7483863200
    Email: mvyas@usc.edu

    CSCI 574 : Homework#4

******** */

#include <opencv/cv.h>
#include <opencv2/highgui/highgui_c.h>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
#include <algorithm>
#include <opencv2/imgproc/types_c.h>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/nonfree/features2d.hpp>

using namespace cv;
using namespace std;

void process_ ()
{
    Mat U, Vt, D, A, P, temp, temp1, D1_sqrt;

    int const kNumberOfImages = 6 ;
    int const kNumberOfPoints = 8;

    // Input Co-Ordinates for homework # 4
       Mat W = (Mat_<double>(2*kNumberOfImages,kNumberOfPoints) << 130, 150, 195, 172,
       111, 130, 174, 152,
         196, 230, 159, 125, 196, 231, 158, 123,
         86, 64, 110, 134, 100, 78, 123, 147,
         119, 113, 47, 50, 131, 126, 60, 63,
         15, 38, 76, 54, 27, 49, 89, 67,
         129, 129, 72, 70, 142, 141, 84, 83,
         27, 63, 74, 35, 21, 58, 67, 27,
         18, 31, 40, 26, 36, 49, 60, 46,
         48, 72, 135, 113, 43, 67, 130,
         108, 53, 55, 76, 75, 72, 74, 96,
         95, 130, 145, 46, 37, 131, 145, 48,
         38, 29, 29, 32, 32, 53, 54, 58, 57);

    // Average matrix
    Mat W_ (Size(1,12),CV_64F);
    for (unsigned int i = 0 ; i < (2*kNumberOfImages) ; i++)
    {
        Scalar sumX_Y = cv::sum(W.row(i));
```

```
        W_.at<double>(i, 0) = ((double)sumX_Y[0]) / ((double)kNumberOfPoints);
    }

    // Compute distance for average for each co-ordinate to get the actual W matrix
    for SVD
    for (unsigned int i = 0 ; i < (2*kNumberOfImages) ; i++)
    {
       W.row(i) -= Mat::ones(Size(kNumberOfPoints, 1), CV_64F) * (W_.at<double>(i,
       0));
    }

    // Compute SVD
    SVD::compute(W, D, U, Vt);

    // Algorithm 6.2 from text book
    Rect roi1(0, 0, 3, 12);
    Rect roi2(0, 0, 3, 8);
    Mat U1(U, roi1);
    transpose(Vt, temp);
    Mat Vt1(temp, roi2);
    Mat D1 = (Mat_<double>(3,3) << D.at<double>(0,0), 0, 0, 0,
    D.at<double>(1,0), 0, 0, 0, D.at<double>(2,0));
    // the Sqrt transpose method to compute P
    sqrt(D1, D1_sqrt);
    transpose(Vt1, temp1);
    A = U1 * D1_sqrt;
    P = D1_sqrt * temp1;

    cout << "W = " << endl << W << endl;
    //cout << "D = " << endl << W << endl;
    cout << "P = " << endl << P << endl;

}

int main()
{
    process_ ();
    waitkey(0);
    return 0;
}
```

```
% %
%  CSCI 574 -- Homework 4
%  Name : Manan Vyas
%  USC ID: 7483-8632-00
%  Email: mvyas@usc.edu
%
%  --- MATLAB SCRIPT TO RECONSTRUCT THE 3D OBJECT FROM THE P MATRIX
% %


%%%%%% ---------- P without normalizing the image1-6 co-ordinates with
 %%%%%%            their (xmean,yMean)

% P = [10.68738697719022, 11.95551382929746, 10.77165019507828,
9.498434834764916, 11.08447847900898, 12.34859144897835, 11.14330366828817,
9.860168648547067;
%   -5.303208060341911, -5.001824783800161, 6.184852695940656,
5.831138723870756, -5.543576843486757, -5.263975143306675,
5.901585268985403, 5.593843579755961;
%   1.160920991146074, -4.472669793817366, -4.408059110869711,
1.373532631986038, 4.537978694365356, -1.102153197087567, -
0.8951262596487105, 4.947693301752587];

%%%%%% ---------- P without normalizing the image1-6 co-ordinates with
 %%%%%%            their (xMean,yMean)

P = [5.299128798824463, 5.58706582631356, -5.719672187109584, -
5.96018356534191, 5.687601465499774, 5.99536827459116, -5.299608428312881,
-5.589700184464575;
  -2.723651839908247, 4.114682604728753, 4.366309566008261, -
2.604716547736885, -4.236357961296556, 2.610707266262171,
2.759445666176336, -4.286418754233829;
  3.530388809558039, 2.242932374301797, 2.184358678432943,
3.403371328854095, -1.979429452208752, -3.329800258387276, -
3.678647637271629, -2.373173843279213];

% Arrange the co-ordinates
x=P(1,:);
y=P(2,:);
z=P(3,:);
%  Make a 3d plot
scatter3(x,y,z)
%  Join the points according to the connectivity order given
line(x(1:4),y(1:4),z(1:4))
line(x(5:8),y(5:8),z(5:8))
line(x(1),y(2),z(1))
line([x(1) x(5)],[y(1) y(5)],[z(1) z(5)])
line([x(2) x(6)],[y(2) y(6)],[z(2) z(6)])
line([x(3) x(7)],[y(3) y(7)],[z(3) z(7)])
line([x(4) x(8)],[y(4) y(8)],[z(4) z(8)])
line([x(4) x(1)],[y(4) y(1)],[z(4) z(1)])
line([x(5) x(8)],[y(5) y(8)],[z(5) z(8)])
```

Results and Discussion:

The Matlab plot of the 3D object obtained is that of the cuboid ;



Numerical Results :

```
U1 =
[-0.2391141806936814, 0.3591881731689685, 0.2733137896788123;
  0.4033626114047415, 0.4561987591709346, -0.08861995692960935;
  -0.2580268180512466, -0.3453986667453141, -0.2059006406344539;
  0.3742723239014452, -0.1130537704226614, -0.1830522229687365;
  -0.2118805389176538, 0.2747136565845874, -0.3833080804057641;
  0.3236563146768411, -0.04529927408029328, -0.2073038050217747;
  -0.0430895726624913, 0.5415122995790902, -0.04965588072414673;
  -0.04704438599790296, 0.1274487659380349, -0.4552151205863886;
  -0.352177165328015, 0.3442793113060812, -0.05761368222552492;
  -0.1178133741569448, -0.03518682081509113, -0.4203209306784303;
  0.5306896397327526, 0.1382178578080773, 0.003028633025448235;
  -0.01443960120339757, -0.07011594497590001, -0.5118185490111985]

D1_sqrt =
[15.97367701747584, 0, 0;
  0, 10.04731541908306, 0;
  0, 0, 8.247354925496493]

V_transpose =
[0.3317413262473633, 0.3497670461347807, -0.3580686012902373, -
0.3731253335610348, 0.3560608781107387, 0.3753280016887775, -
0.3317713524891543, -0.3499319648412334;
```

-0.2710825455658695, 0.4095305495151129, 0.4345747479685215, -0.2592450260683263, -0.4216407850847759, 0.2598412767358337, 0.2746450719497934, -0.4266232894502895;

0.4280631598191475, 0.2719577845944071, 0.2648556656243873, 0.4126621637602447, -0.2400077928123834, -0.4037415981811673, -0.446039690361716, -0.2877496924429194]

P =
[5.299128798824463, 5.58706582631356, -5.719672187109584, -5.96018356534191, 5.687601465499774, 5.99536827459116, -5.299608428312881, -5.589700184464575;

-2.723651839908247, 4.114682604728753, 4.366309566008261, -2.604716547736885,-4.236357961296556, 2.610707266262171, 2.759445666176336, -4.286418754233829;

3.530388809558039, 2.242932374301797, 2.184358678432943, 3.403371328854095, -1.979429452208752, -3.329800258387276, -3.678647637271629, -2.373173843279213]

Since parallel lines remain parallel to each other and we get a perfectly reconstructed rectangular 3D solid object which involves some translation, scaling, rotation and shearing transformations , it is thus concluded that the reconstruction is indeed an affine transform of the rectangular solid.