Ahmed Elgabaly
861054818
Raj Kumar
861139625

EE 243
Advanced Computer Vision
Professor MATTHEW J. BARTH

3D Reconstruction from Multiple Images

ABSTRACT

Our Project is 3D reconstruction from multiple images, the data set that we used is the impart data set, from the university of Surrey in England. The Algorithm that we used for the feature detection was the SURF algorithm, at first it did not give us enough points to get an accurate reconstruction but when we pushed its strongest feature threshold to the limitations of our computers we got a close to accurate reconstruction, our approach can be further refined to give a perfect reconstruction.

INTRODUCTION

What is 3D reconstruction

3D reconstruction is the process of capturing the shape and the appearance of real world objects, this process can be accomplished either by active or passive methods. If the model is allowed to change its shape in time, this is referred to as non-rigid or spatio - temporal reconstruction.

3D reconstruction methods

Active 3D reconstruction

Active 3D reconstruction methods are these methods which actively interfere with the reconstructed object, either mechanically or radio metrically. A simple example of a mechanical method would use a depth gauge to measure a distance to a rotating object put on a turntable. More applicable radiometric methods emit radiance towards the object and then measure its reflected part. Examples range from moving light sources, colored visible light, time of lasers to Microwaves and Ultrasound, which are all basically the sources used in 3D scanners.

Passive 3D reconstruction

Passive 3D reconstruction methods are those methods which do not interfere with the reconstructed object; they only use a sensor to measure the radiance reflected or emitted by the object's surface to infer its 3D structure. Typically, the sensor is an image sensor in a camera sensitive to visible light and the input to the method is a set of digital images (one, two or more) or video.

Image Based 3D reconstruction

Image based 3D reconstruction is the creation of three-dimensional models from a set of images. It is the reverse process of obtaining 2D images from 3D scenes. The essence of an image is a projection from a 3D scene onto a 2D plane, during which process the depth is lost. The 3D point corresponding to a specific image point is constrained to be on the line of sight. From a single image, it is impossible to determine which point on this line corresponds to the image point. If two images are available, then the position of a 3D point can be found as the intersection of the

two projection rays. This process is referred to as triangulation. The key for this process is the relations between multiple views which convey the information that corresponding sets of points must contain some structure and that this structure is related to the poses and the calibration of the camera.

APPROACH

1) Find interest points using SURF
2) Match interest points
3) For each matching image points x and x', Compute point X in scene using triangulation and the given projection matrices

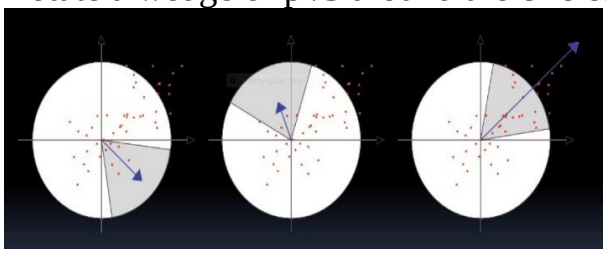
1) Find interest points using SURF (Speeded Up Robust Features)

a) Finding image interest points using Hessian Matrix

- Hessian matrix is a 2x2 matrix containing second order partial derivatives
- The determinant is equal to the product of its eigen values.
- For a Hessian matrix the eigen vectors form an orthogonal basis giving the direction of the curve
- If both eigen values are positive then it is a local min
- If both eigen values are negative then it is a local max
- Therefore if the product is positive we are at a local extremum, which is an interest point.
- By using a threshold we can detect the major features

b) Find Feature Direction

- For each feature look at the pixels in a circle of radius 6σ.
- Compute the x and y Haar transform for each pixel.
- Use the x and y values in the Cartesian space.
- Rotate a wedge of pi/3 around the circle.



Similar to how the RANSAC algorithm.

c) Generate feature vectors

- A square window of size 20σ is centered on each interest point with orientation based on the direction.
- Divide the window into 4x4 sub regions.
- For each sub-region compute the following
  - Sum of dx
  - Sum of dy
  - Sum of abs(dx)
  - Sum of abs(dy)
- Therefore SURF has a 64D feature vector consisting of the above 4 values for each of the 16 sub-regions

Why SURF ?

There are other feature detectors like Harris corner detectors, we chose SURF because our dataset did not have much of corners to detect, and corner detectors gave us a poor response, 3D reconstruction essentially needs a lot of point clouds to get a good model, which is done by SURF.

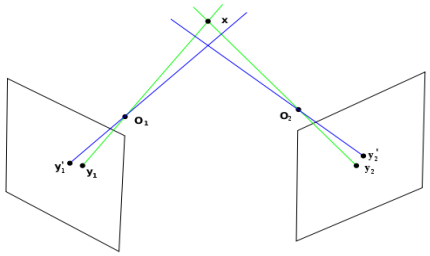
2) Match Interest points

a) Find the SURF feature in the image i.
b) Find the SURF feature in the image i+1.
c) Compare both of them for similar features.

3) For each matching image points x and x', Compute point X in scene using triangulation and the given projection matrices

a) Triangulation is the process of determining a point in 3D space given its projections onto two, or more images.
b) It is necessary to know the parameters of the camera projection function from 3D to 2D for the cameras involved, represented by the projection Matrices.
c) Each point in an image corresponds to a line in 3D space where all the points on the line are projected to the point in the image.
d) If a pair of corresponding points in two, or more images, can be found then they are the projection of a common 3D point X.

e)



Because our case here is not the ideal stereo image pair case (green line), the image points y1 and y2 cannot be measured with arbitrary accuracy. Instead points y'1 and y'2 are detected and used for the triangulation. The corresponding projection lines (blue) do not, in general, intersect in 3D space and may also not intersect with point x.

f) Re projection Errors & Valid Points

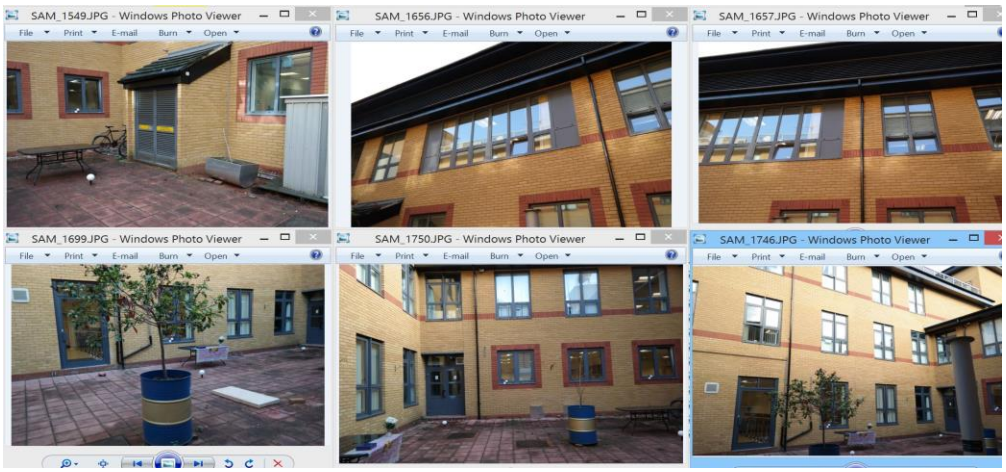
- The triangulation function in MATLAB finds an optimal intersection point for the projection lines.
- Then it computes the Euclidean distance between the Optimal point and the other intersection points from the Epi polar projection lines
- The one with the least distance is taken as the valid 3D point

DATA SET

Our data set is a group of 247 images taken from the Impart 3D reconstruction project by the University of Surrey in England. The 247 images were taken for an outdoors scene of a patio. The components of our data set are as follows

1) 247 Images around the patio taken by a Samsung NX300 smart phone
2) Projection Matrices folder containing 247 projection matrices .txt files for each Image

Below is a sample of the images



CODE

1) Save content in projection matrices files to a .mat file

Save_Projection_Matrices.m

```
x = dir('*.txt'); % load the text files information from the directory
Projection_Matrices = zeros(3,4,length(x)); % Projection matrices
variable (3x4x247)
for i = 1:length(x) % go over the text files
    Projection_Matrices = csvread(x(i).name); % read each txt file as
csv file into the Projection_Matrices Variable
end
save('Projection Matrices','Projection_Matrices'); % Save the the
Projection_Matrices variable as (Projection Matrices.mat)
```

2) Create the 3D Point Cloud

Generate_3D_PointCloud.m

```
% 3D reconstruction for all 247 images SURF Threshold 200
clc; close all; clear all;

tic % Start timer

im_dir = dir('*.jpg'); % Read all the images
P = load('Projection Matrices.mat'); % Read Projection Matrices
% Initialize Point cloud variables
points3D = []; % 3D points
color = []; % 3D points colors
num_Im_dir = length(im_dir); % Number of Images

for i = 1:num_Im_dir-1 % go over the images
    disp(i); % Display the index of the current image
    im1 = imread(im_dir(i).name); % Read the Current image
    im2 = imread(im_dir(i+1).name); % Read the next image
    im1_gray = im2double(rgb2gray(im1)); % Create a gray scale version
of the Current image
    im2_gray = im2double(rgb2gray(im2)); % Create a gray scale version
of the next image

    % Extract Feature points for the current & the next image based on
the
    % MetricThreshold or the Strongest Feature Threshold for the SURF
    % algorithm where the density of the point cloud increases as the
    % MetricThreshold decreases from 1000 to 0
    im1_points = detectSURFFeatures(im1_gray, 'MetricThreshold', 400);
    im2_points = detectSURFFeatures(im2_gray, 'MetricThreshold', 400);

    % Extract Feature descriptors from the feature points
    im1_features = extractFeatures(im1_gray,im1_points);
    im2_features = extractFeatures(im2_gray,im2_points);
```

```
% Match Features descriptors between the current image & the next image
based on the
    % MaxRatio threshold where more points are matched togeather as
this
    % ratio increases from 0 to 1
    indexPairs = matchFeatures(im1_features,im2_features, 'MaxRatio',
1);
    im1_matchedpoints = im1_points(indexPairs(:,1));
    im2_matchedpoints = im2_points(indexPairs(:,2));

    % Estimate 3D Points Corresponding to matched Points between the
    % current image & the next image & calculate the reprojection
errors
    % using the Feature descriptors & the projection matrices for the
    % current image & the next image
    [curr_points3D, reprojErrors] =
triangulate(im1_matchedpoints,im2_matchedpoints, ...
        P.Proj_Matrices(:,:,i)',P.Proj_Matrices(:,:,i+1)');

    % Eliminate noisy points based on reprojection errors
    errorDists = max(sqrt(sum(reprojErrors .^ 2, 2)), [], 3);
    validIdx = errorDists < 1;
    curr_points3D = curr_points3D(validIdx, :);
    im1_matchedpoints = im1_matchedpoints(validIdx, :);
    im2_matchedpoints = im2_matchedpoints(validIdx, :);

    % Get the color of each reconstructed point using by trakcing down
the locations of the
    % matched points in the RGB current image & next image
    im1_matchedpoints = round(im1_matchedpoints.Location);
    numPixels = size(im1,1) * size(im1,2);
    allColors = reshape(im2double(im1), [numPixels, 3]);
    colorIdx = sub2ind([size(im1,1), size(im1, 2)],
im1_matchedpoints(:,2), ...
        im1_matchedpoints(:, 1));
    curr_color = allColors(colorIdx, :);

    % Concatinate current 3D points & their colors with the previous 3D
    % points & the current colors with the previous colors
    points3D = [points3D;curr_points3D];
    color = [color;curr_color];
end

toc % End timer

% Once the program finishes for the Current MetricThreshold used, copy
the
% contents of the color variable & the points3D variable into 2
seperate
% excel files
```

3) Show the reconstructed point cloud
ImportAndShowPointCloud.m

```
clc; close all; clear all;

points3D = xlsread('3DPoints_im1_to_im133_SURF200.xlsx'); % Load 3D
points
color = xlsread('Color_im1_to_im133_SURF200.xlsx'); % Load Colors of 3D
points

% Manual filtering of noise, this manual filteration is based on trial
and
% error after generating the point cloud for different
MetricThresholds,
% this filteration removes the random points generated on the edges of
the
% point cloud & do not add density or accuracy to the shape &
appearance of
% the point cloud produced
a = points3D(:,1);
b = points3D(:,2);
c = points3D(:,3);
xx = size(points3D);
for i = 1:xx(1)
    if((a(i)<-15)||(a(i)>30))
        points3D(i,:) = [0 0 0];
        color(i,:) = [0 0 0];
    elseif((b(i)<-20)||(b(i)>8))
        points3D(i,:) = [0 0 0];
        color(i,:) = [0 0 0];
    elseif((c(i)>18))
        points3D(i,:) = [0 0 0];
        color(i,:) = [0 0 0];
    end
end
points3D(end,:) = 0;
points3D(all(points3D == 0,2),:) = [];
color(all(color==0,2),:) = [];

% plot the 3D point cloud
color = uint8(255 * color); % convert the color values from 0-1 range
to 0-255 range
ptcloud = pointCloud(points3D,'color',color); % Create point cloud
struct
figure; grid on % create figure with grid lines
showPointCloud(ptcloud); % Plot the point cloud using the point cloud
struct
axis tight
xlabel('x-axis (mm)');
ylabel('y-axis (mm)');
zlabel('z-axis (mm)');
title('Reconstructed Point Cloud');
```

RESULTS

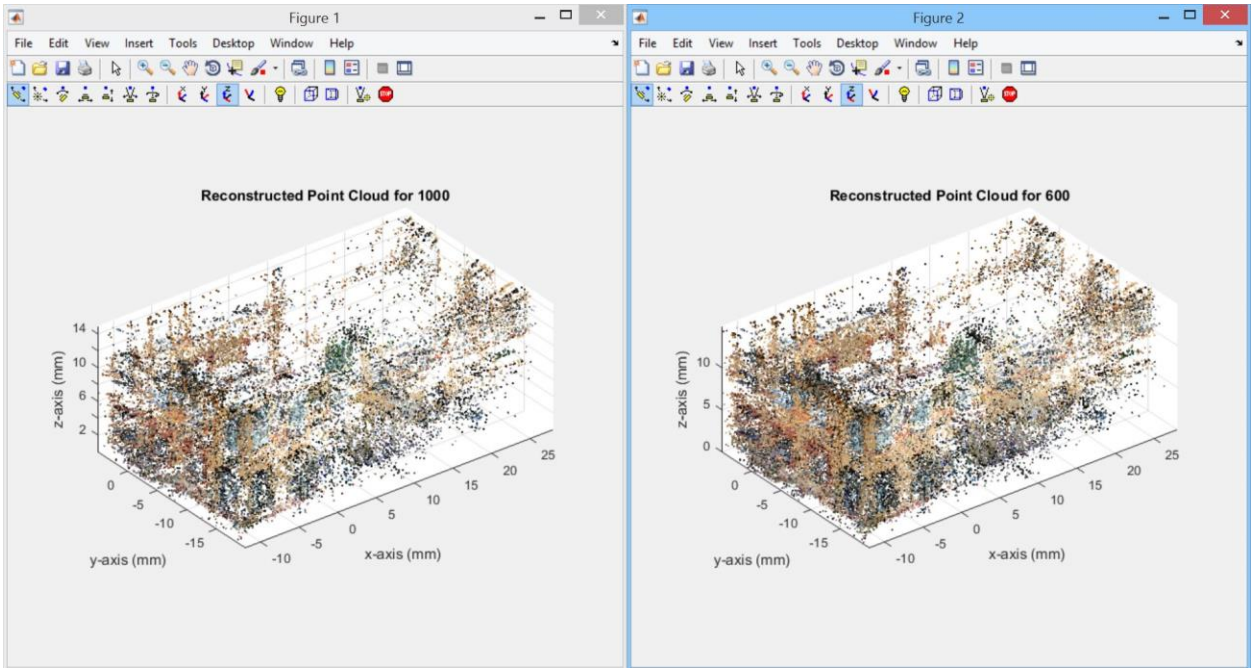
Our results varied based on the value of the MetricThreshold for the SURF algorithm, we started from 1000 & decreased it to 200, this was the time it took to create the 3D point cloud increased from 8 - 20 minutes to 5 – 52 hours & the number of points increased from around 53000 points to 300000 points and the density & the accuracy of the point cloud increased.

Table of Results

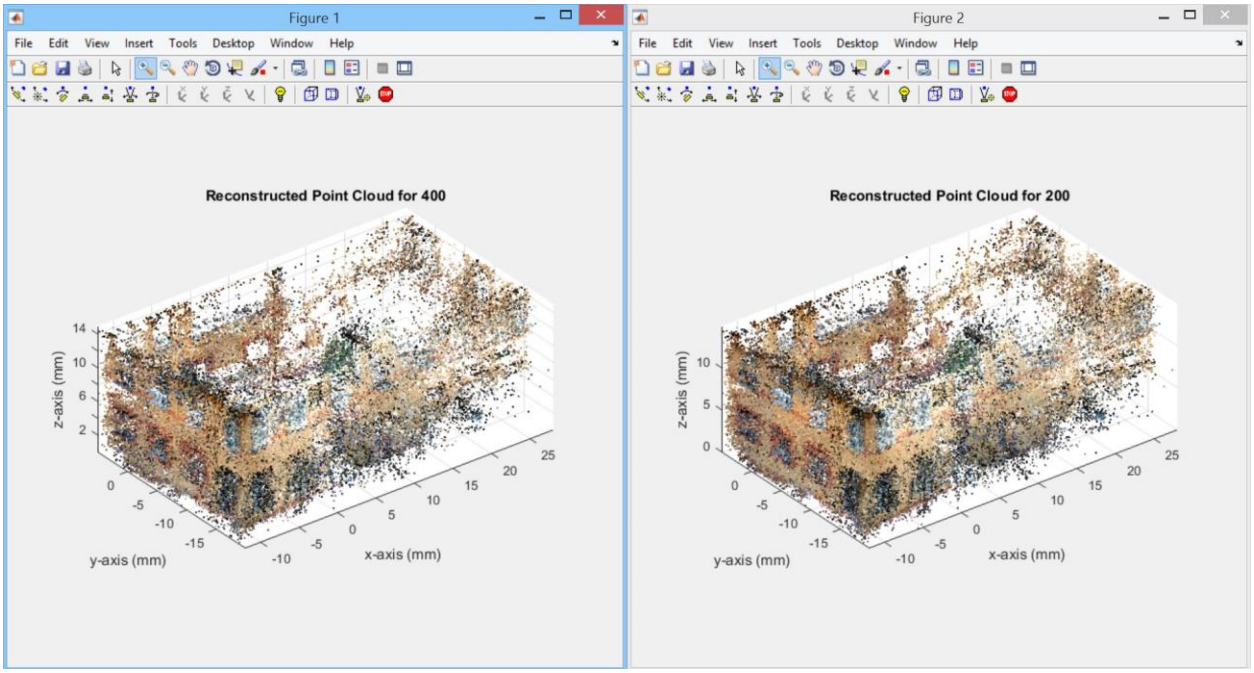
MetricThreshold	Approximate Time Taken for 3D point cloud creation	Approximate Number of points produced
1000	10 min	53000
600	20 min	92000
400	5 hours	110000
200	52 hours	300000

Reconstructed Point cloud figures

MetricThreshold = 1000 VS MetricThreshold = 600



MetricThreshold = 400 VS MetricThrehsold = 200



CONCLUSION

As we decreased the SURF algorithm MetricThreshold we kept getting better results. However the minimum value we could try is 200 due to hardware limitations; the time taken to produce the point cloud increased drastically therefore the following improvements can be done to this Project to increase the density and improve the accuracy of the reconstructed point cloud

1) Using a SURF metric threshold less than 200 can give a lot more points
2) GPU implementation to get faster results
3) Use another coding language faster than MATLAB, for example C++.
4) Combine the SURF algorithm with other feature descriptor algorithms while using maximum thresholds for all of them.
5) Use different feature descriptors for a perfectly reconstructing certain parts of the point cloud where other descriptors fail.

These improvements can efficiently increase the number of points produced to millions where at that time an accurate dense 3D point cloud can be reconstructed.

REFERENCES

1) MATLAB computer vision tool box
   http://www.mathworks.com/products/computer-vision/
2) 3D reconstruction
   http://en.wikipedia.org/wiki/3D_reconstruction
3) 3D reconstruction from multiple images
   http://en.wikipedia.org/wiki/3D_reconstruction_from_multiple_images
4) SURF
   http://www.sci.utah.edu/~fletcher/CS7960/slides/Scott.pdf
5) Class lecture notes - PPT 8
6) Class lecture notes - PPT 9
7) IMPART Data set
   http://cvssp.org/impart/
   H. Kim and A. Hilton, "Influence of Colour and Feature Geometry on Multi-modal 3D Point Clouds Data Registration," Proc. 3DV, 2014.