ROBOTICS & CV

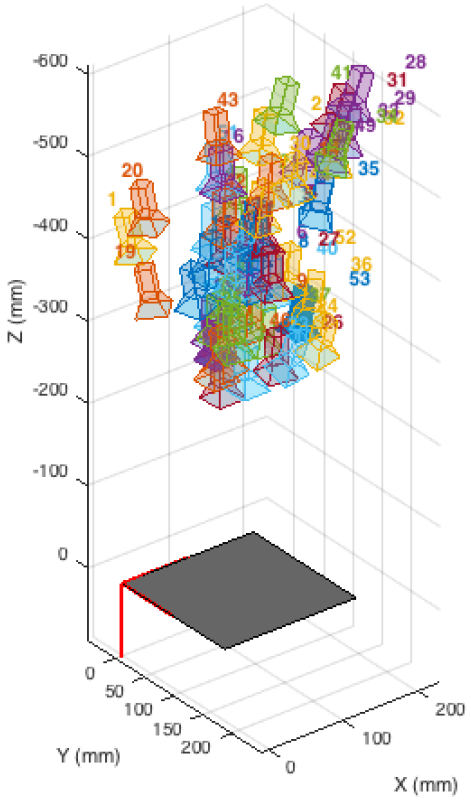
Project 3

Niral Shah

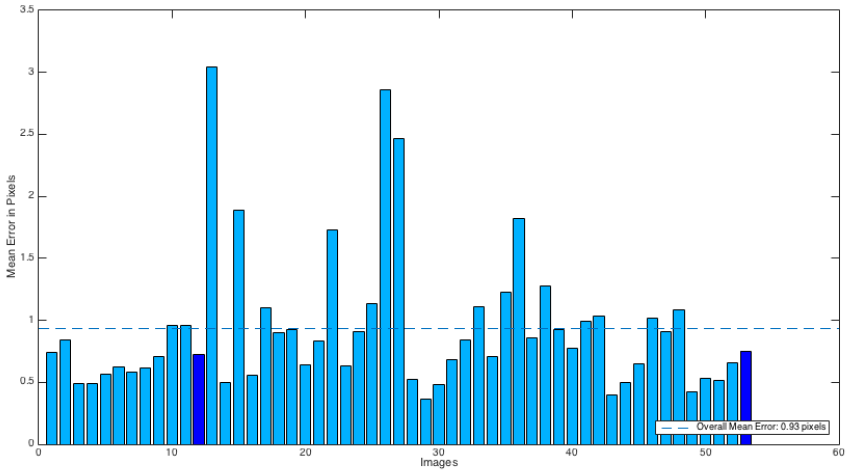
RUID:150008870 | DATE: 11/14/2016

Table of Contents

Camera Calibration:

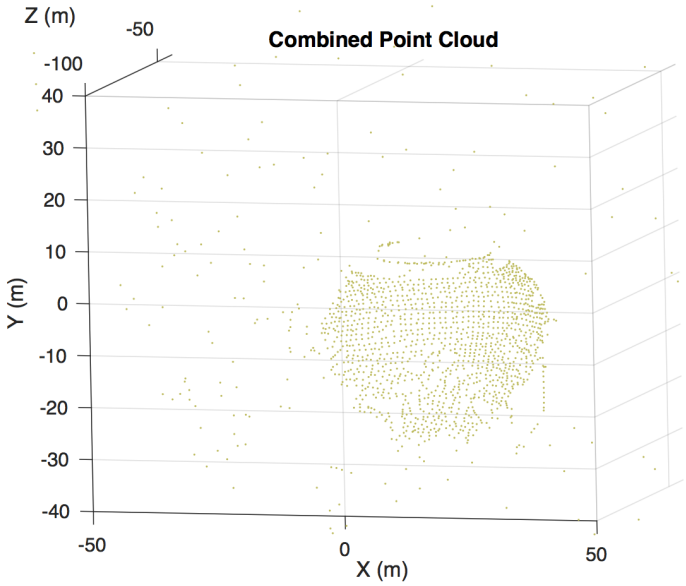


Re-projection Error:



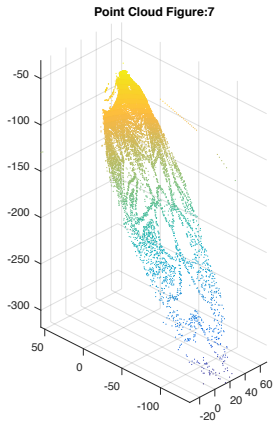
In order to reach the final result, a 3D reconstruction or point-cloud was constructed during each iteration of the loop. At the conclusion of the loop these point clouds were merged and refined.

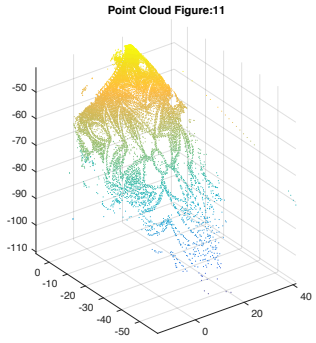
Final Result:

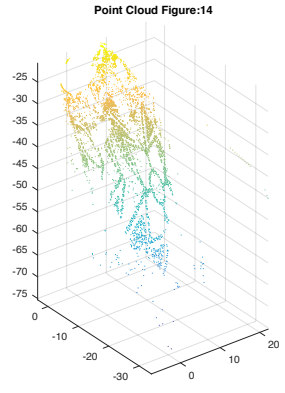


Original Image

Highlights:







Contents

Niral Shah

RCV Project 3 (FINAL COPY)
11/12/16

```
% Intrisic Matrix: (calculated w/camera calibration)
%K = [2923.04133761325,0,0;0,2913.74339399455,0;1652.84237267260,1204.46130305411,1];

%import cameraParams from calibration
load('/CV/cameraParams.mat');

% Load Image1
im1 = imread('/CVImages/Photos/FinalPics/images1.jpg');
im1 = imrotate(im1,-90);
im1 = undistortImage(im1, cameraParams); %remove lens distortion


% Initial Features:
imPts1 = detectMinEigenFeatures(rgb2gray(im1));
figure;
imshow(im1, 'InitialMagnification', 50);
title('150 Strongest Corners from the First Image');
hold on;
plot(selectStrongest(imPts1, 150));
hold off;

vSet = viewSet;
viewId = 1;
vSet = addView(vSet, viewId, 'Points', imPts1, 'Orientation', ...
    double(eye(3,3)), 'Location', ...
    double(zeros(1, 3)));


vSet = updateView(vSet, 1, 'Points', imPts1.Location);
imPts1 = imPts1.Location;
```

```
Warning: Image is too big to fit on screen; displaying at 33%
```

Detecting feature points

```
tracker = vision.PointTracker('MaxBidirectionalError', 1, 'NumPyramidLevels', 6);
initialize(tracker, imPts1, im1);
prev_im = im1;
prev_im_pts = imPts1;

all_wPoints = [];
viewIDs = 2:37;
allPoses = [];

mergeSize = 1;
Rcum = eye(3,3);
tcum = [0 0 0];
worldPts = [];
```

Identify and Match Features

```
for i = 2:37
```

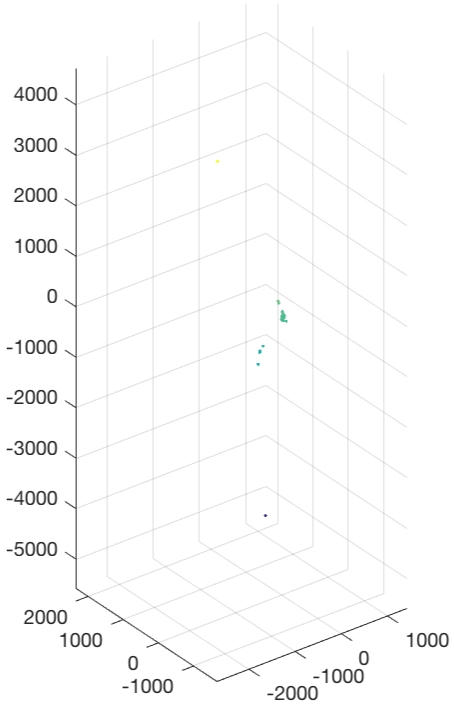
```
curr_im = imread(strcat('/CVImages/Photos/FinalPics/images',num2str(i),'.jpg'));
curr_im = imrotate(curr_im,-90);
curr_im = undistortImage(curr_im, cameraParams); %remove lens distortion
[xlen, ylen] = size(curr_im);

[imPts2, valid_ind] = step(tracker, curr_im);
vSet = addView(vSet, i, 'Points', imPts2);
matchedPoints1 = imPts1(valid_ind, :); %previous image
matchedPoints2 = imPts2(valid_ind, :); % current image

% make sure its valid points (within the image)
 inRange = matchedPoints1(:,1)<xlen & matchedPoints1(:,1)<ylen ...
            & matchedPoints2(:,2)<xlen & matchedPoints2(:,2)<ylen;

 matchedPoints1 = matchedPoints1(inRange,:);
 matchedPoints2 = matchedPoints2(inRange,:);



%Visualize correspondences
figure(1);
subplot1 = subplot(7,8,i-1);
text(.75,1.25,'Tracked Features');
showMatchedFeatures(im1, curr_im, matchedPoints1, matchedPoints2);



% calculate essential matrix:
[E, inliers] = estimateEssentialMatrix(matchedPoints1, matchedPoints2,...
cameraParams, 'Confidence', 99.99);

% inliers
inlierPoints1 = matchedPoints1(inliers, :);
inlierPoints2 = matchedPoints2(inliers, :);

ind = find(valid_ind(inRange));

indX = [ind ind];
vSet = addConnection(vSet,i-1,i, 'Matches', indX);
[orient, loc] = relativeCameraPose(E, cameraParams, inlierPoints1, inlierPoints2);


% Compute extrinsics of the second camera
[R, t] = cameraPoseToExtrinsics(orient, loc);

% relative orientation and translation
Rcum = Rcum*R;
tcum = tcum + (R*t')';

% camera matrix (left and right)
camMatrix2 = cameraMatrix(cameraParams, Rcum, -tcum*Rcum);
camMatrix1 = cameraMatrix(cameraParams, eye(3), [0 0 0]);

%track points along images
vSet = updateView(vSet, i, 'Orientation',Rcum,'Location', tcum);
camPoses = poses(vSet);

% Compute world points
worldPts = triangulate(matchedPoints1, matchedPoints2, camMatrix1, camMatrix2);
all_wPoints = [all_wPoints;worldPts];
```

```
    tracks = findTracks(vSet);

    % Merging Point Clouds
    if i >= 3
        ptCloudCurrent = pointCloud(worldPts);
        moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize);
        tform = pcregrigid(moving, fixed,'Extrapolate', true);
        if i > 3
            accumTform = affine3d(tform.T * accumTform.T);
            ptCloudAligned = pctransform(ptCloudCurrent, accumTform);
            ptCloudScene = pcmerge(ptCloudScene, ptCloudAligned, mergeSize);
        else % i == 3
            tform = pcregrigid(moving, fixed,'Extrapolate', true);
            ptCloudAligned = pctransform(ptCloudCurrent,tform);
            ptCloudScene = pcmerge(ptCloudRef, ptCloudAligned, mergeSize);
            accumTform = tform;
        end
        fixed = moving;
    else
        ptCloudRef = pointCloud(worldPts);
        gridSize = 1;
        fixed = pcdownsample(ptCloudRef, 'gridAverage', gridSize);
    end

    figure(2);
    subplot2 = subplot(7,8,i-1);
    pcshow(pointCloud(worldPts));

    figure;
    pcshow(pointCloud(worldPts));
    title(['Point Cloud Figure:' num2str(i)]);
    prev_im = curr_im;
    imPts1 = imPts2;
end
```

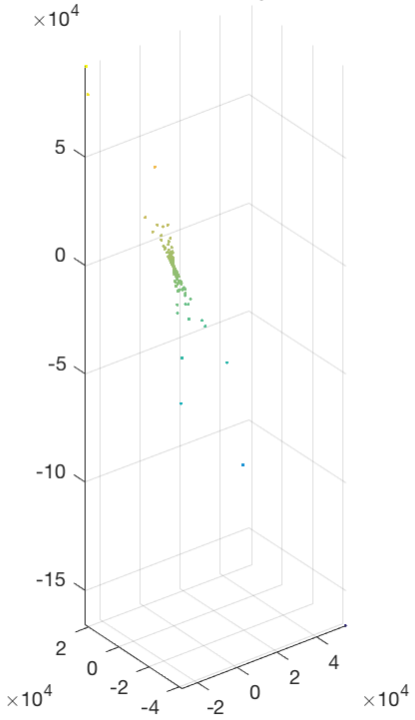
```
Warning: Maximum number of trials reached. Consider increasing the maximum
distance or decreasing the desired confidence.
Warning: Maximum number of trials reached. Consider increasing the maximum
distance or decreasing the desired confidence.
```

Matched Features

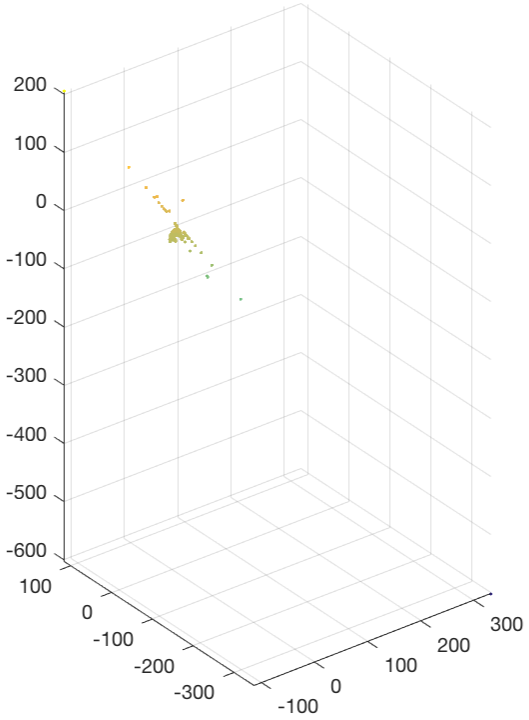
Point Cloud Figure:2



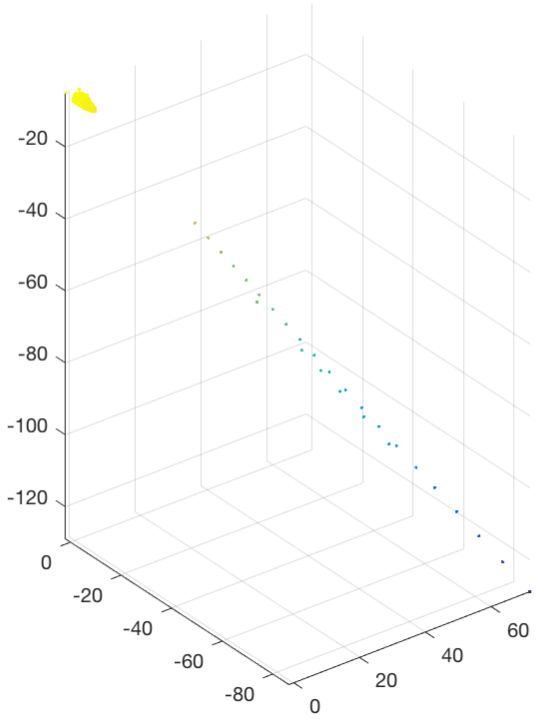
Point Cloud Figure:3



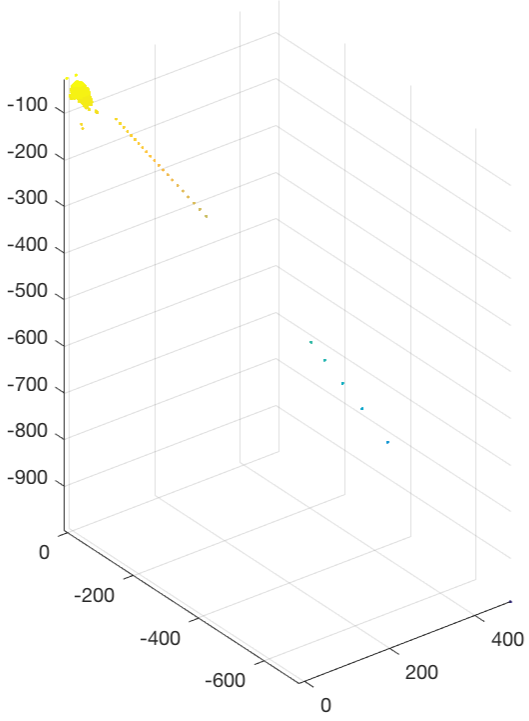
Point Cloud Figure:4



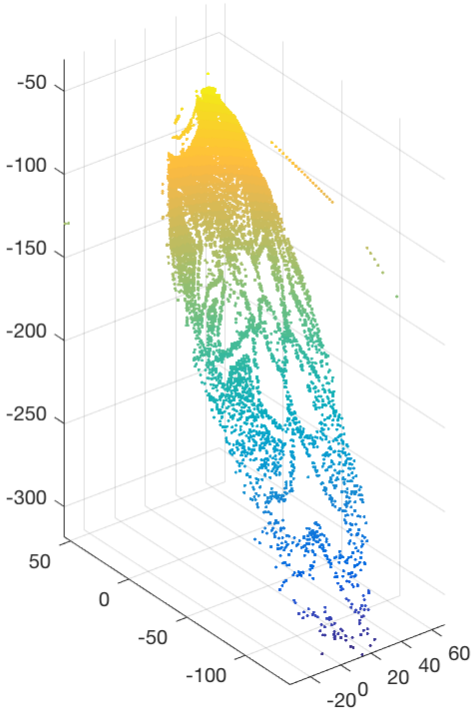
Point Cloud Figure:5



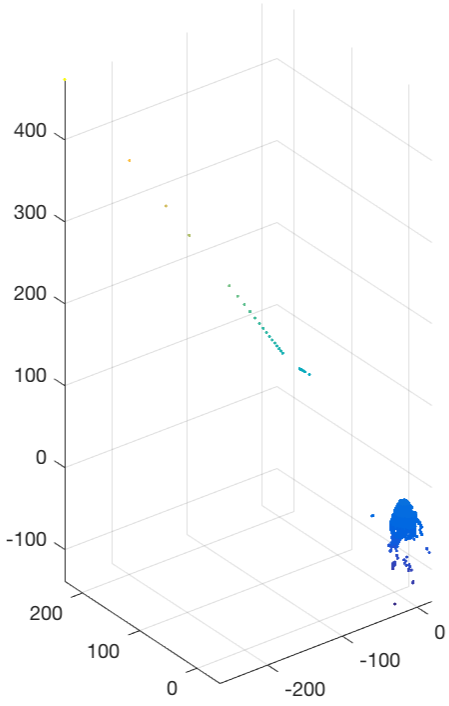
Point Cloud Figure:6



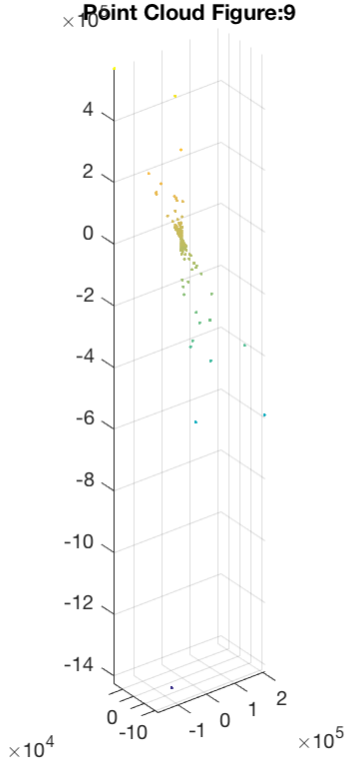
Point Cloud Figure:7



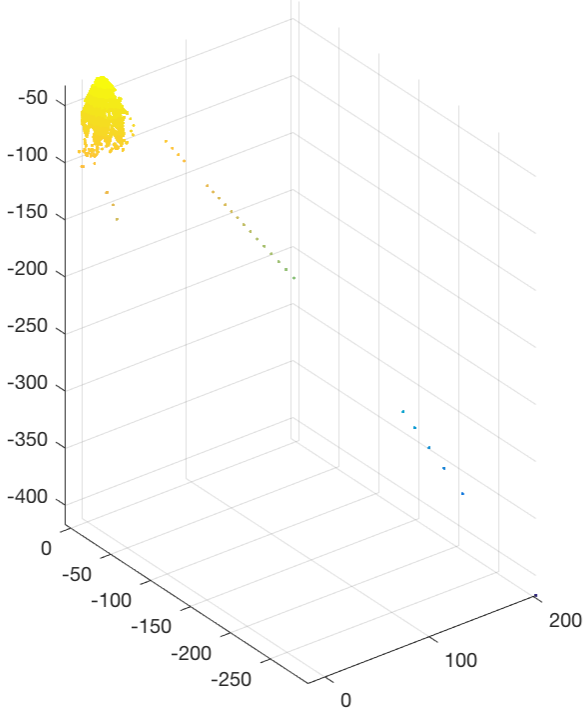
Point Cloud Figure:8



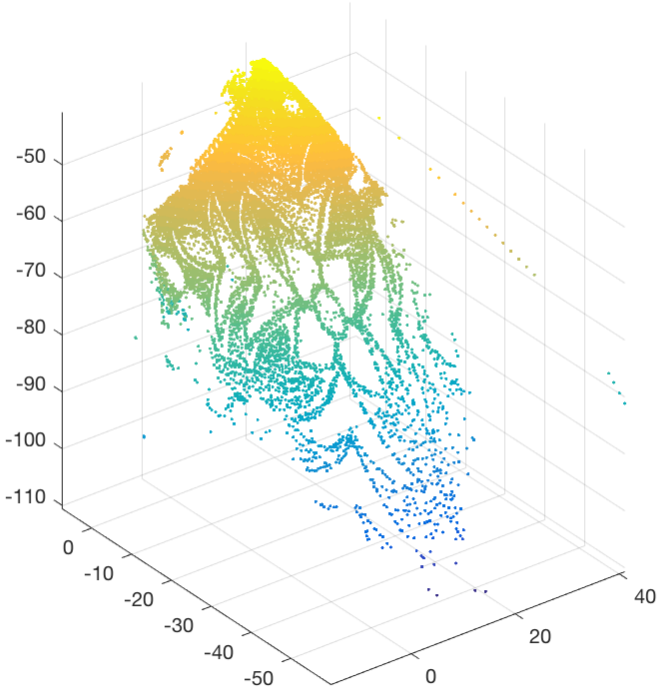
Point Cloud Figure:9



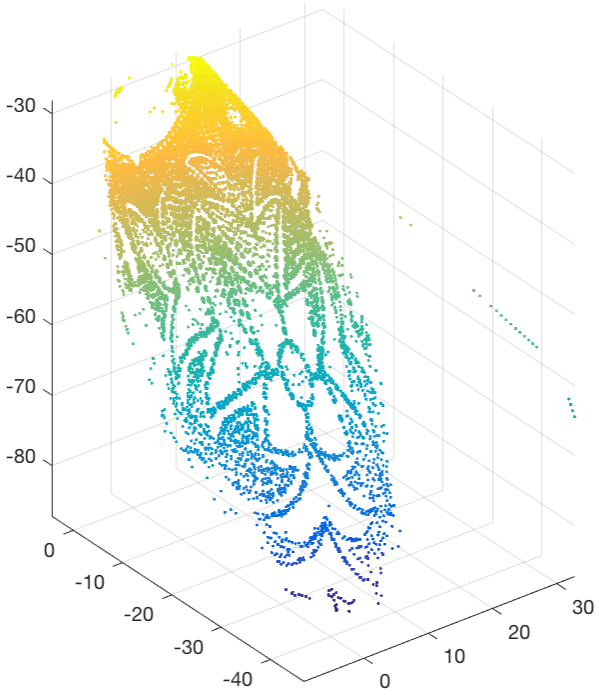
Point Cloud Figure:10



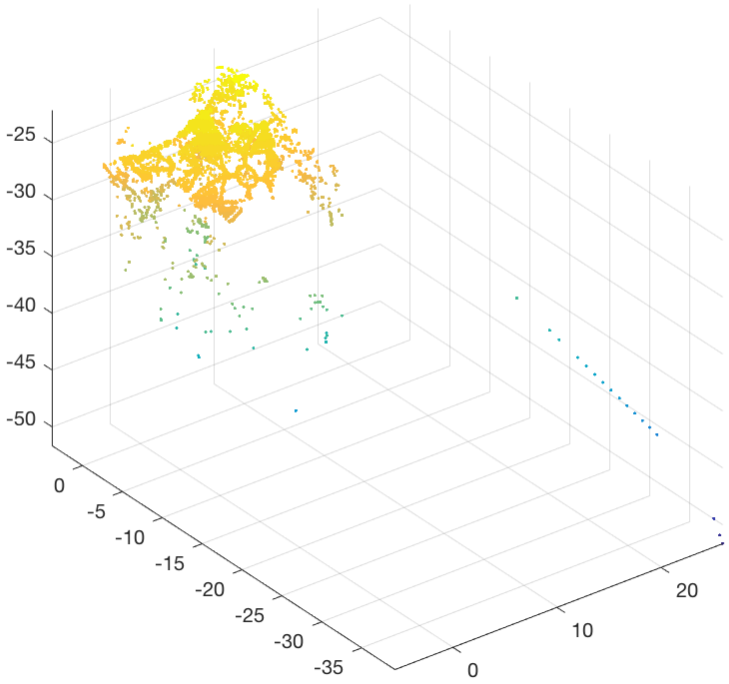
Point Cloud Figure:11



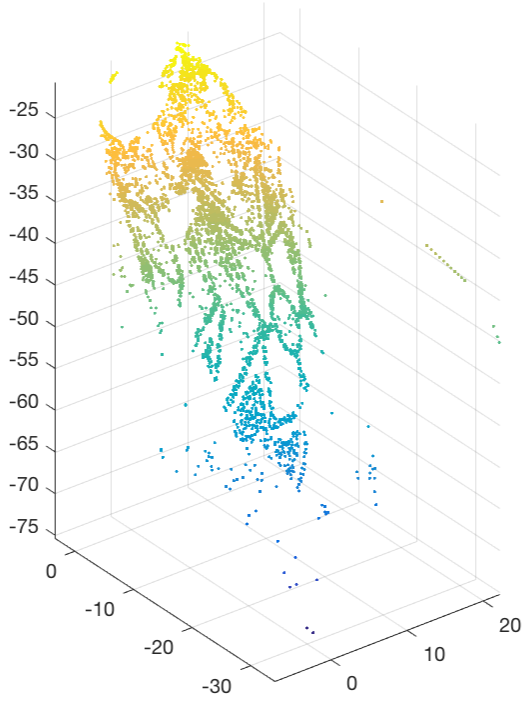
Point Cloud Figure:12



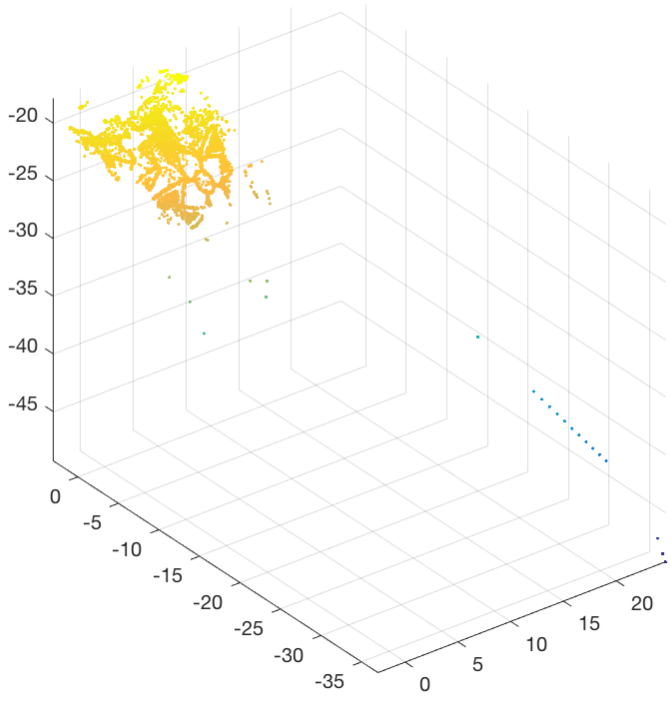
Point Cloud Figure:13



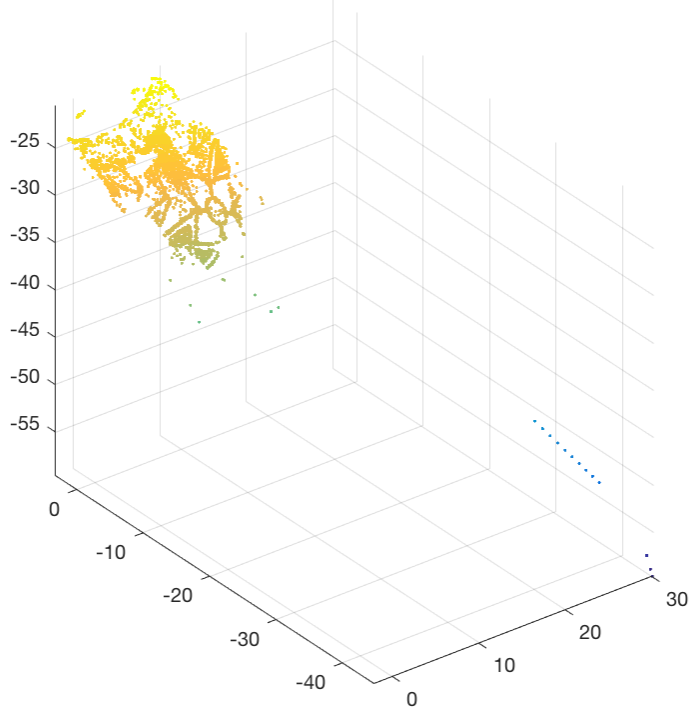
Point Cloud Figure:14



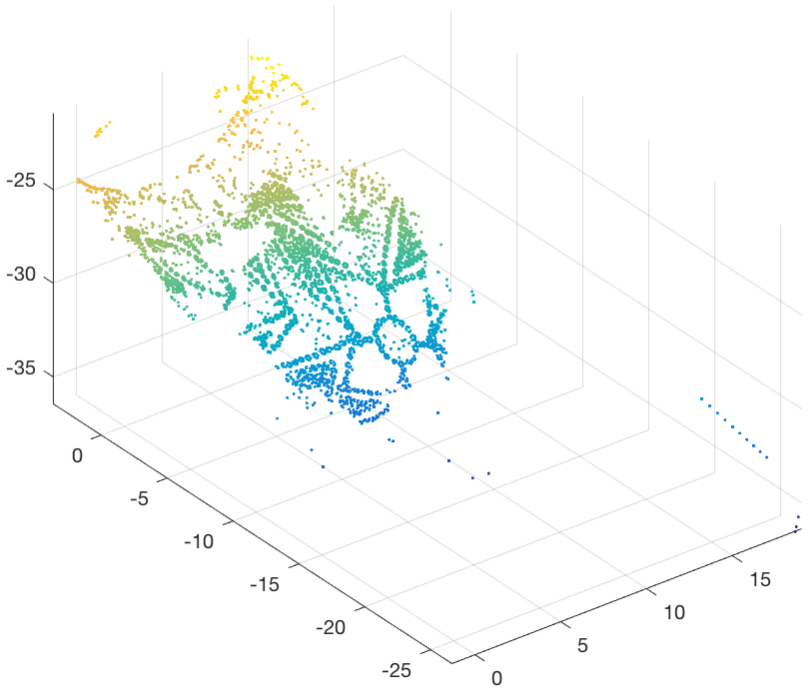
Point Cloud Figure:15



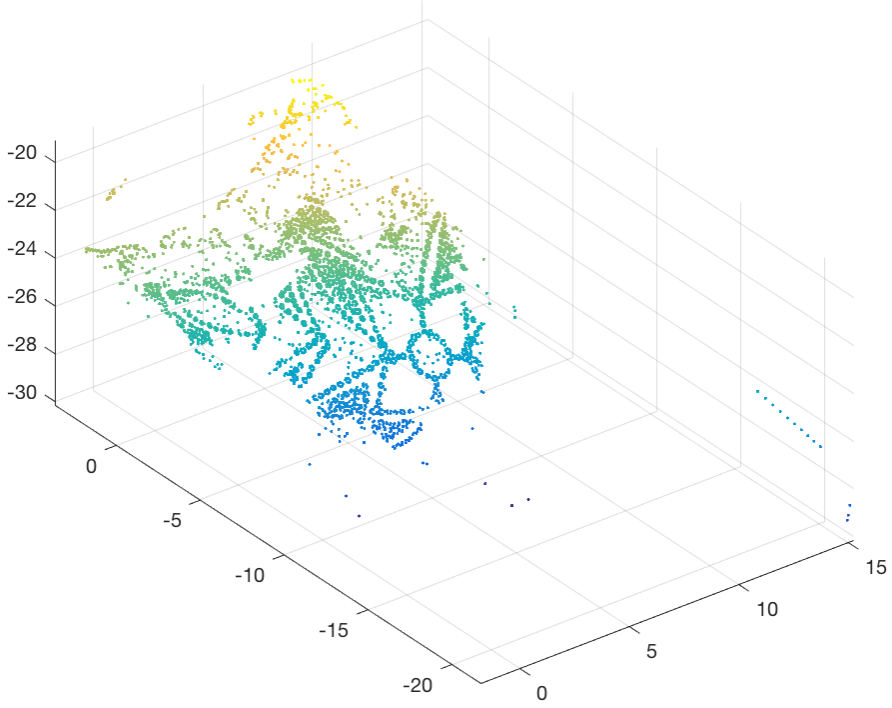
Point Cloud Figure:16



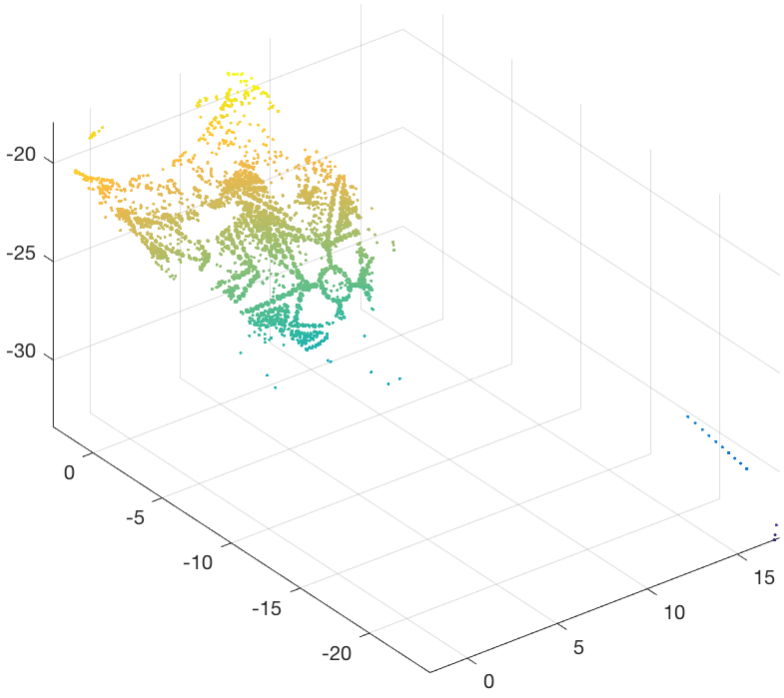
Point Cloud Figure:17



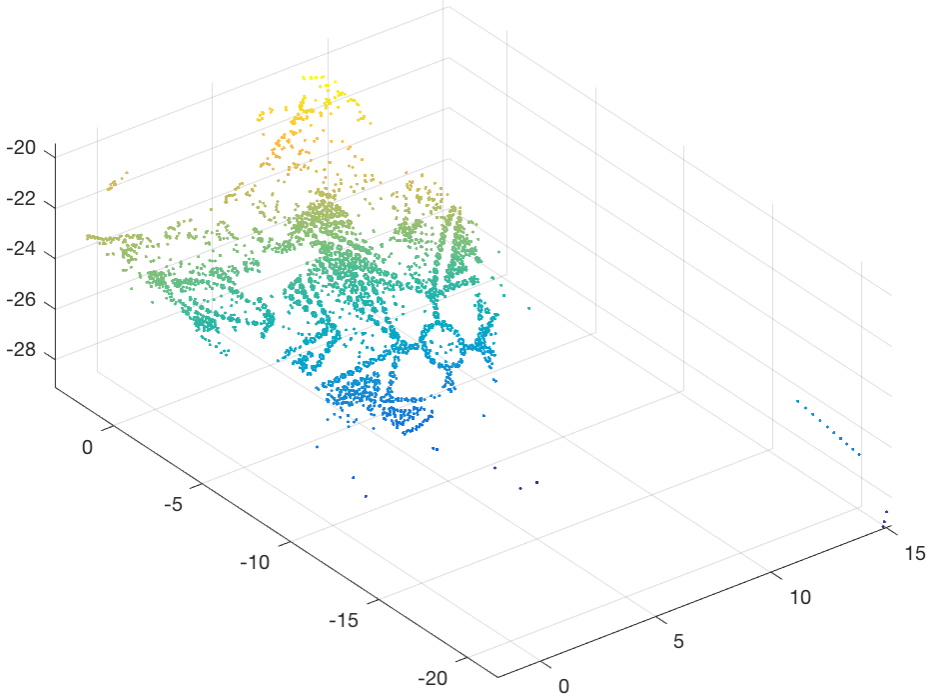
Point Cloud Figure:18



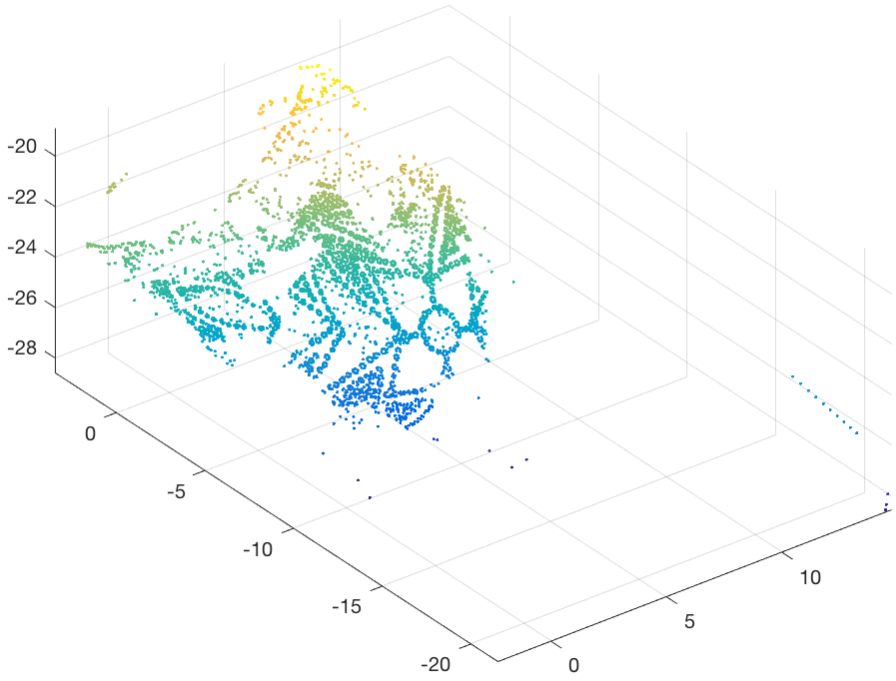
Point Cloud Figure:19



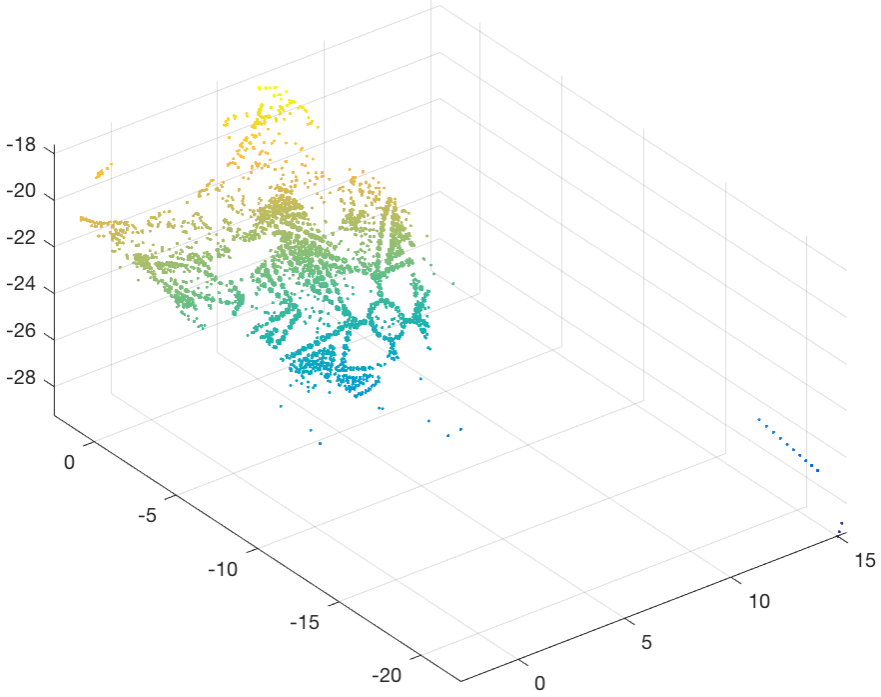
Point Cloud Figure:20



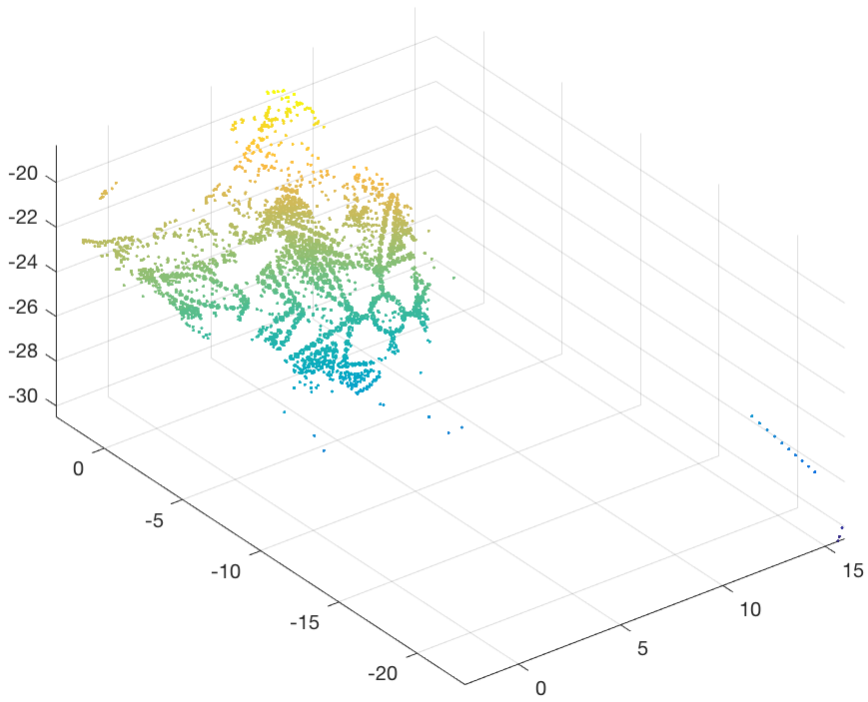
Point Cloud Figure:21



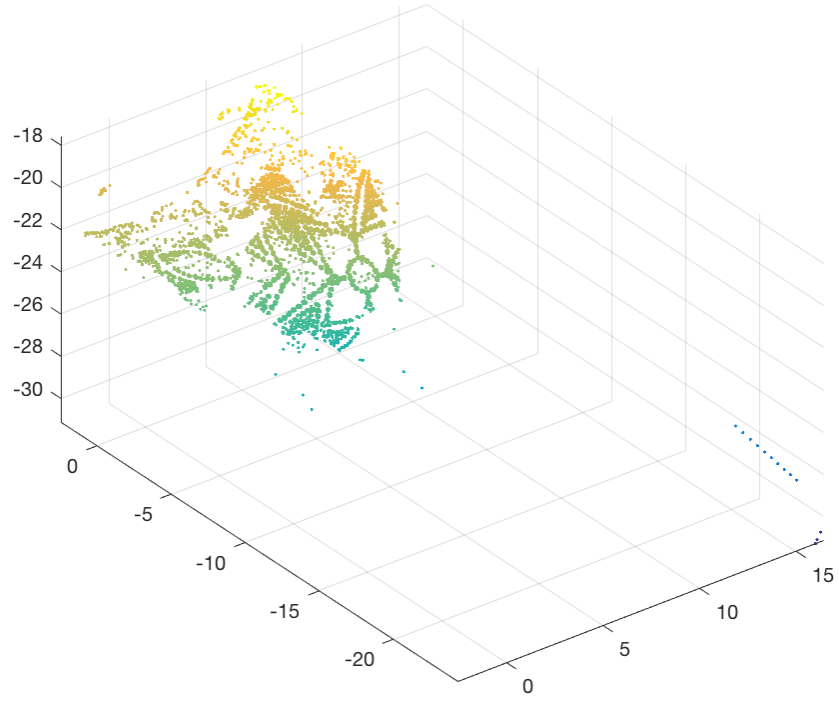
Point Cloud Figure:22



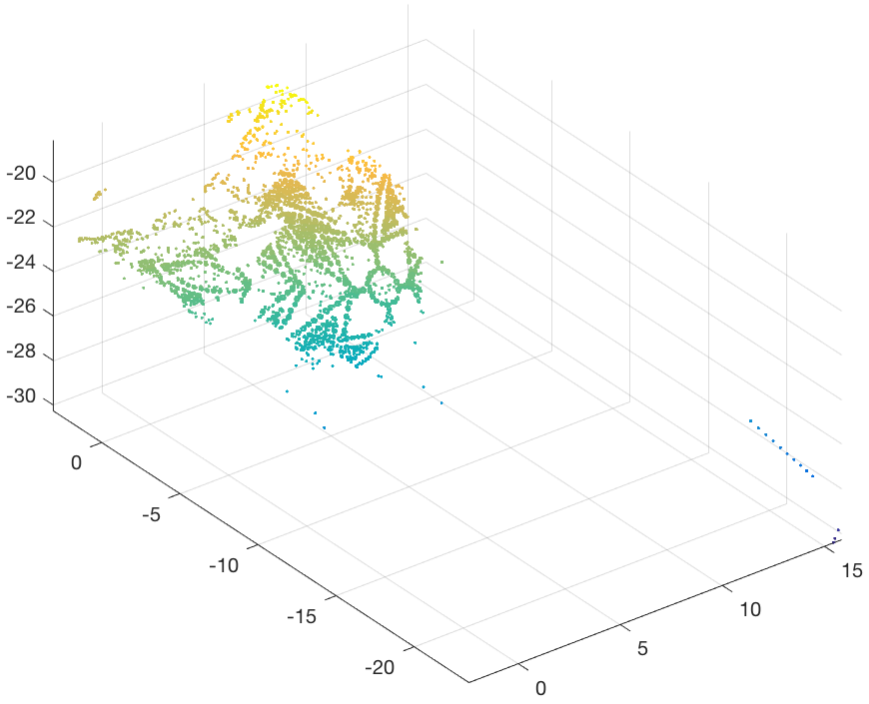
Point Cloud Figure:23



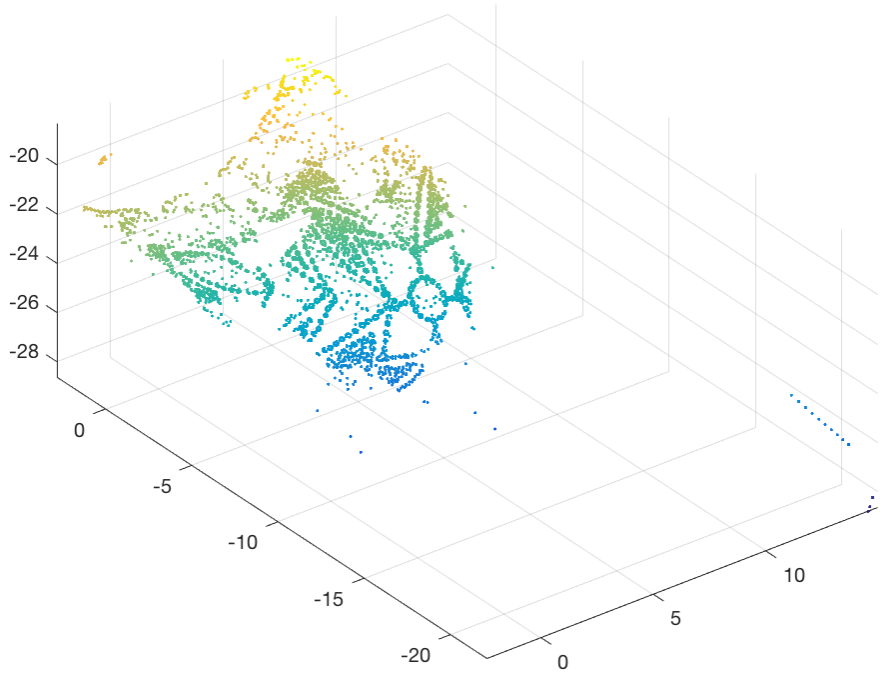
Point Cloud Figure:24



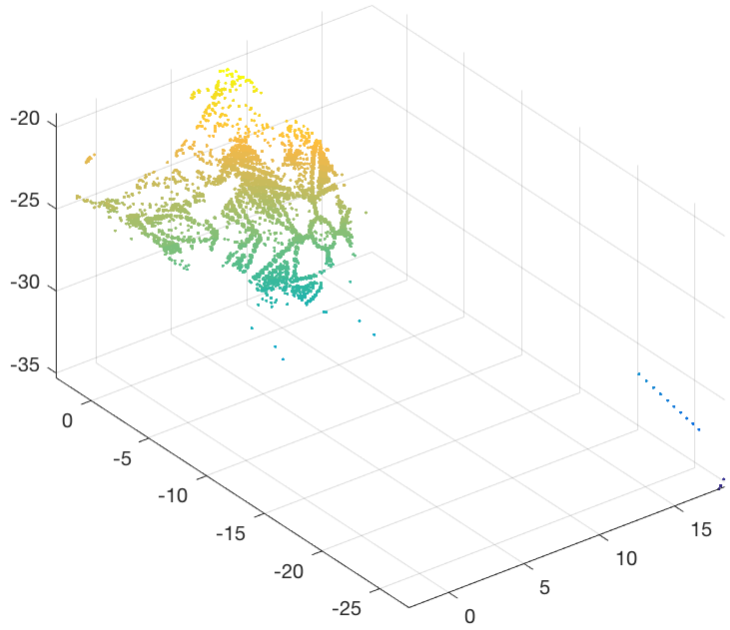
Point Cloud Figure:26



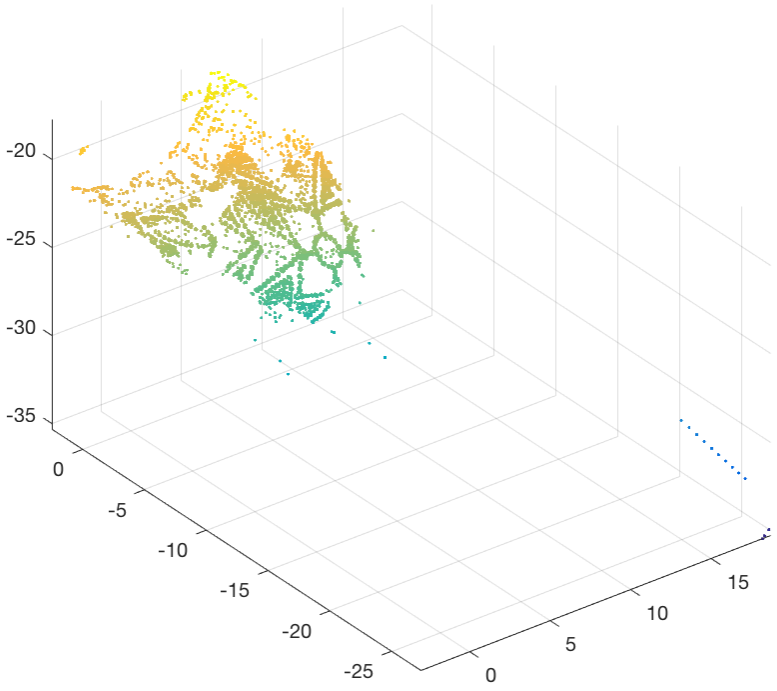
Point Cloud Figure:25



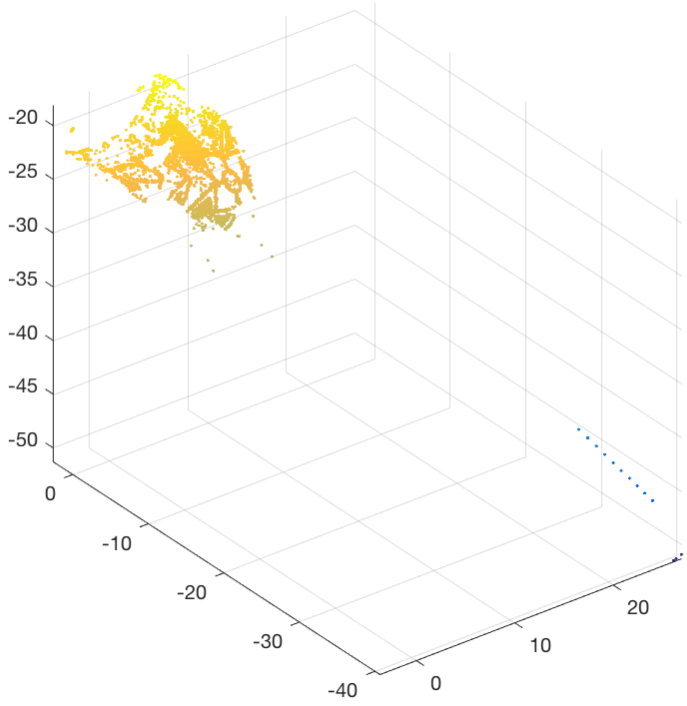
Point Cloud Figure:27



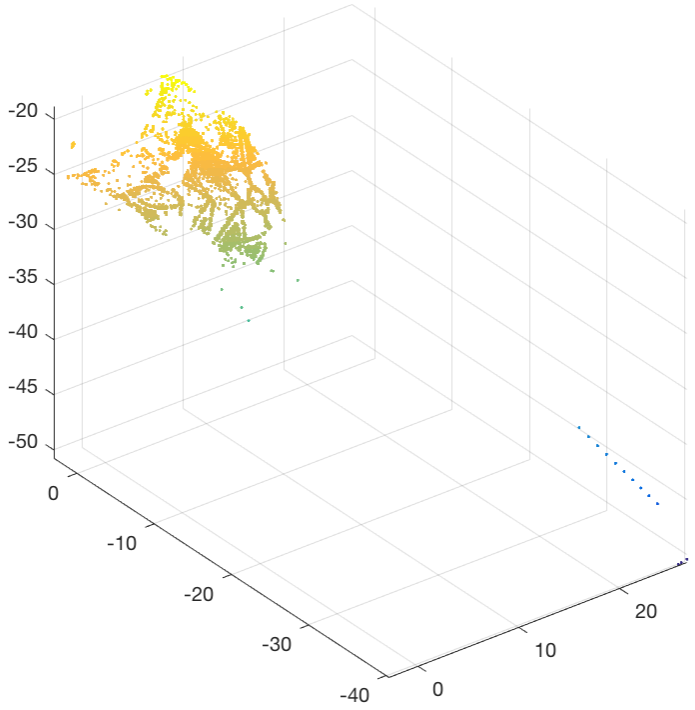
Point Cloud Figure:28



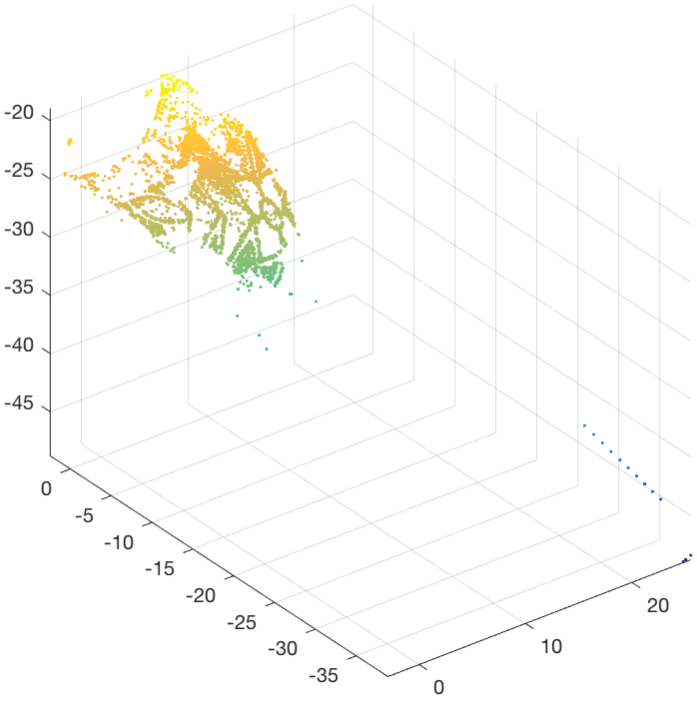
Point Cloud Figure:30



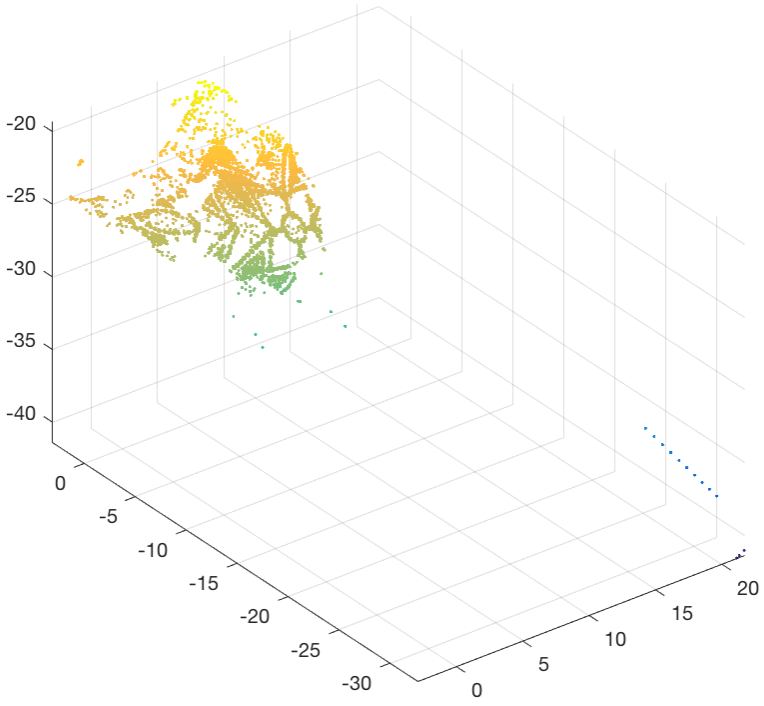
Point Cloud Figure:31



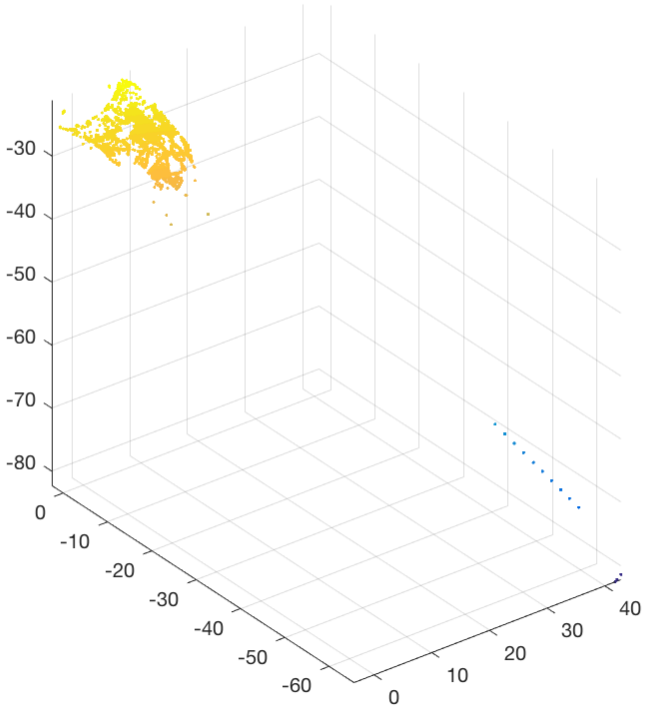
Point Cloud Figure:32



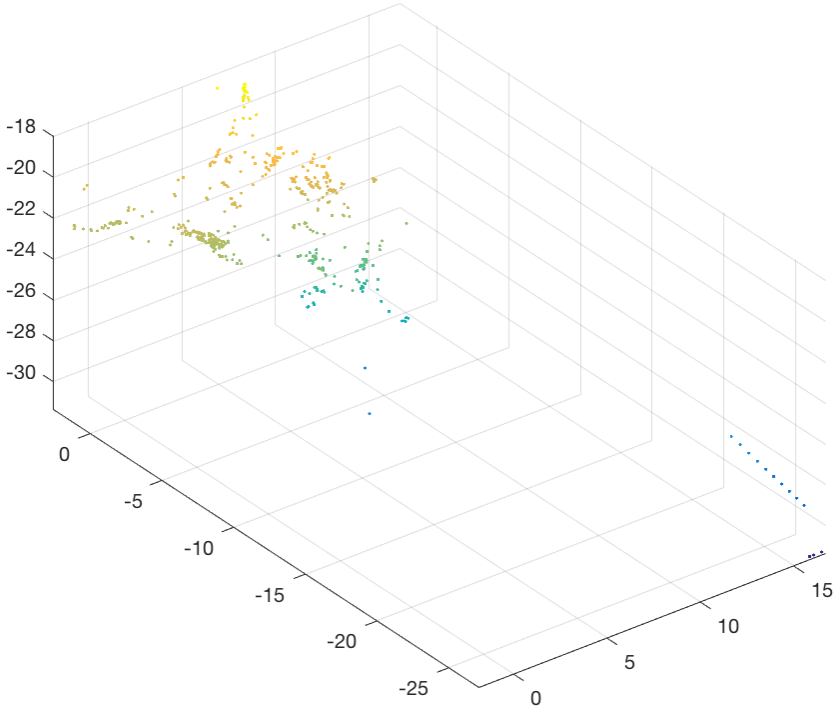
Point Cloud Figure:29



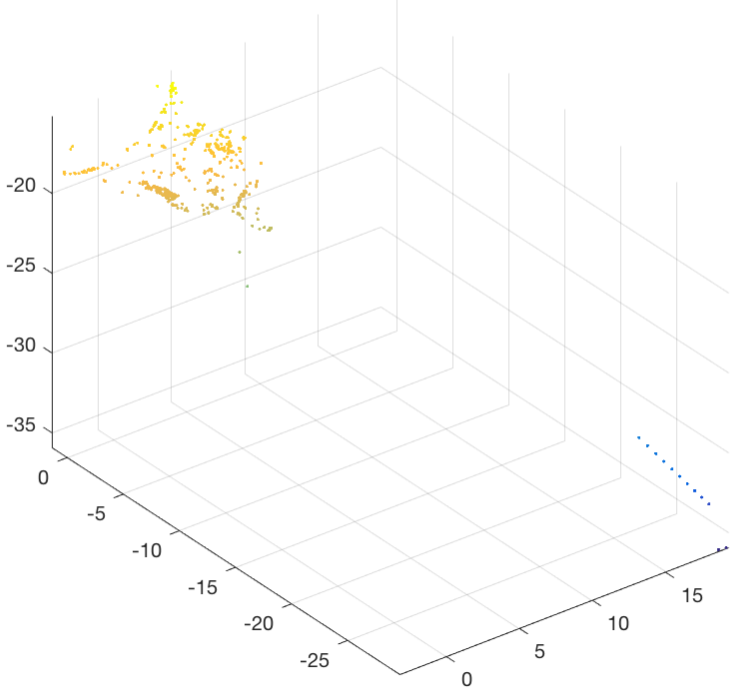
Point Cloud Figure:33



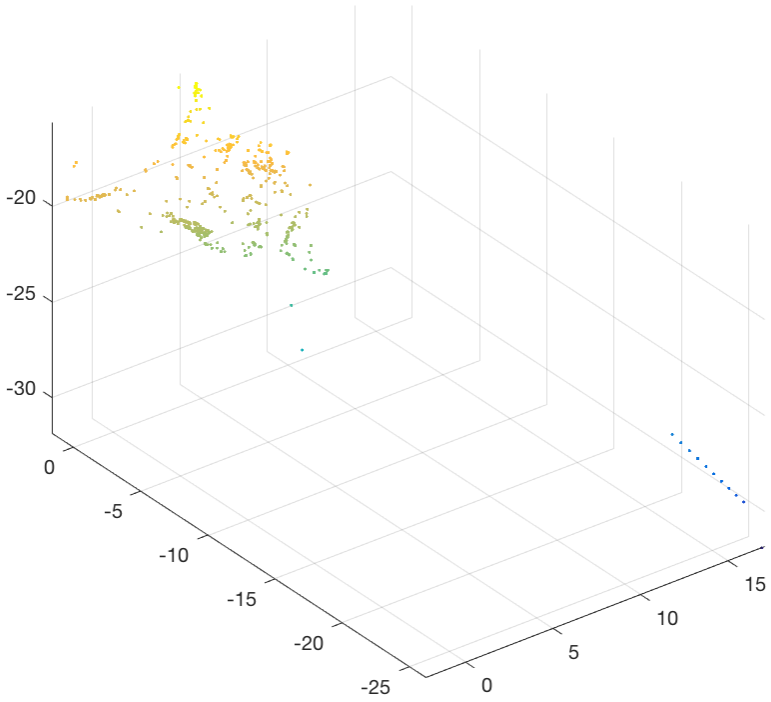
Point Cloud Figure:34



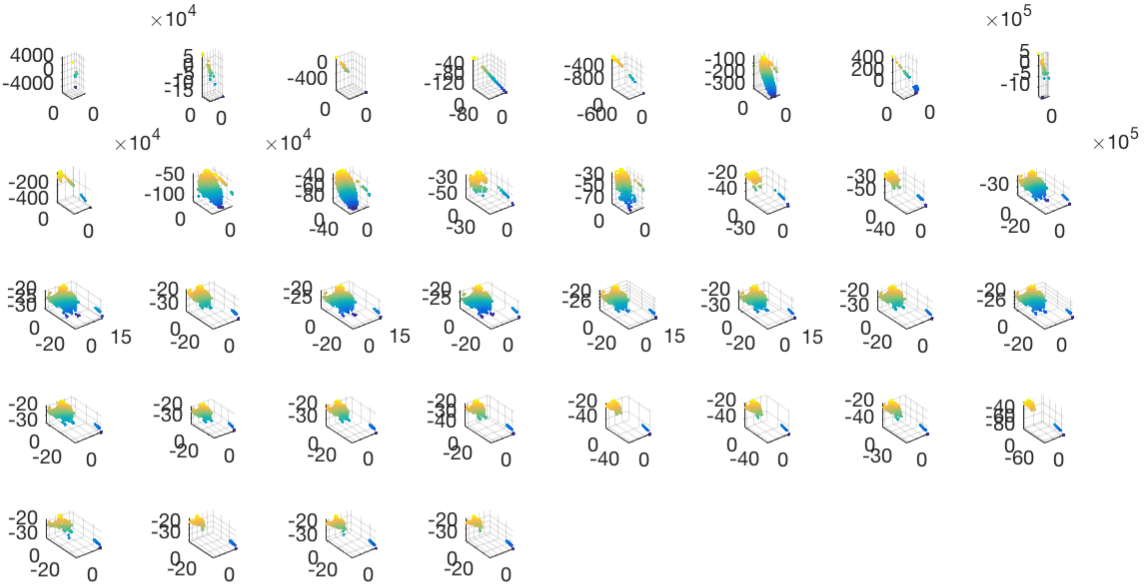
Point Cloud Figure:35



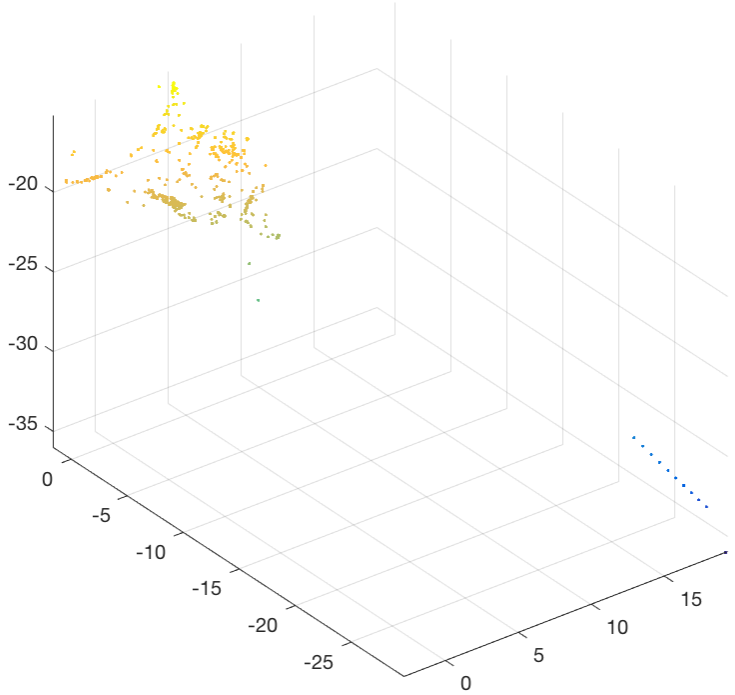
Point Cloud Figure:36



Subplot of each point cloud

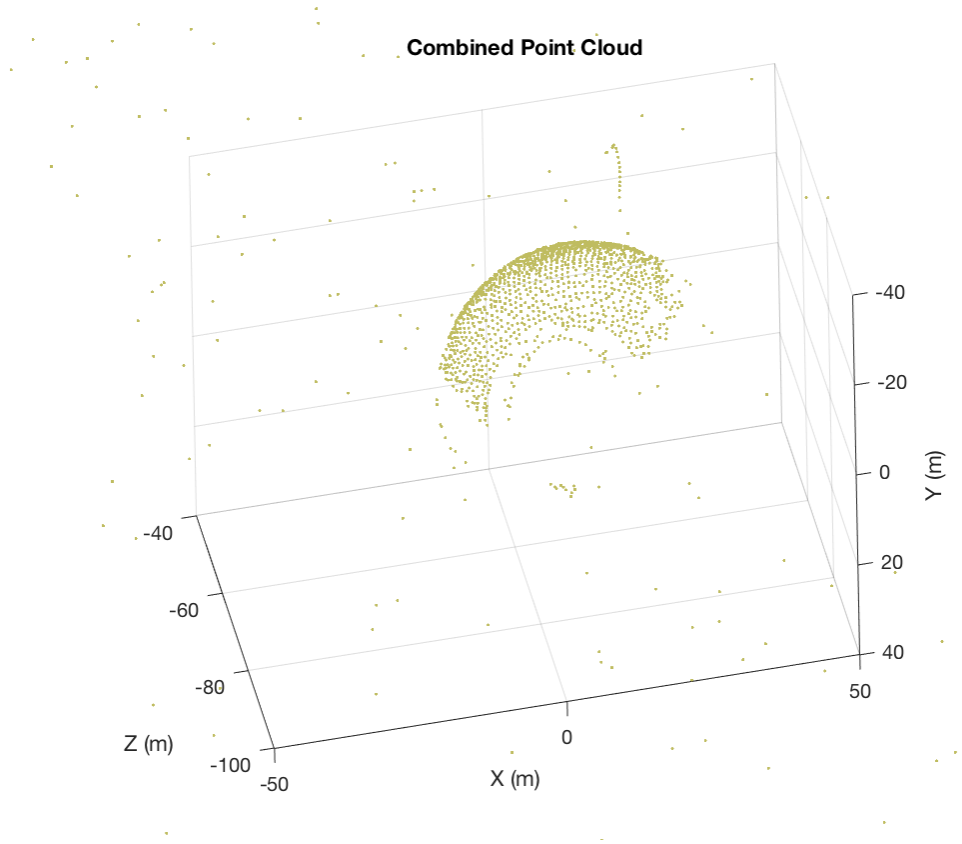


Point Cloud Figure:37



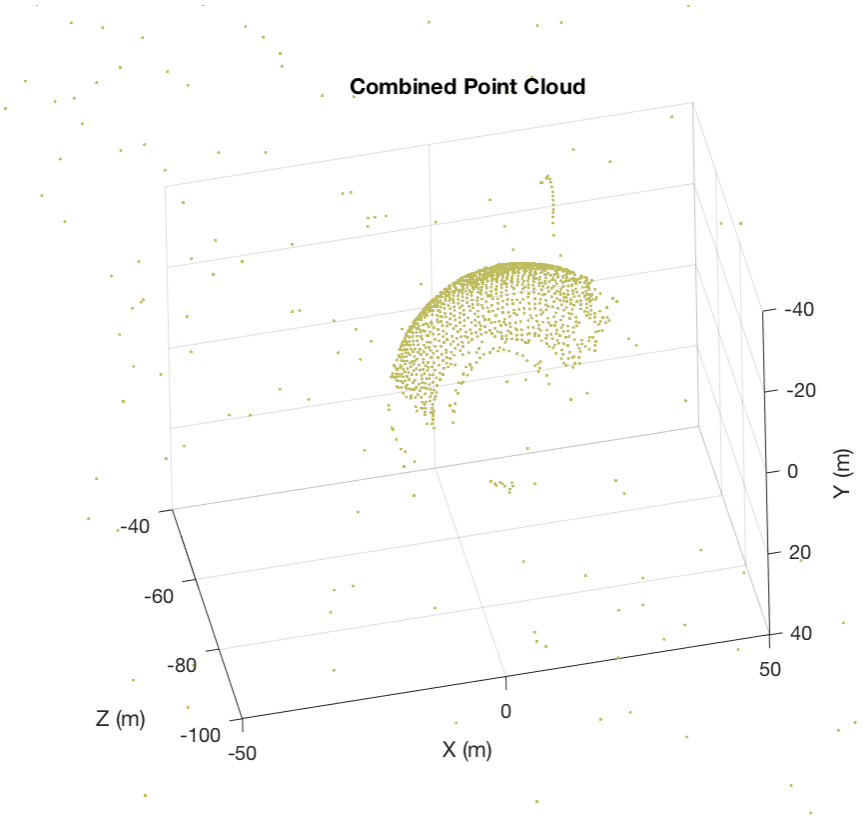
Total Point Cloud

```
figure;
hAxes = pcshow(ptCloudScene, 'VerticalAxis','Y', 'VerticalAxisDir', 'Down');

% Set the axes property for faster rendering
hAxes.CameraViewAngleMode = 'auto';
hScatter = hAxes.Children;
angle = -pi/10;


pcshow(ptCloudScene, 'VerticalAxis','Y', 'VerticalAxisDir', 'Down', ...
        'Parent', hAxes);
set(gca,'xlim',[-50 50],'ylim',[-40 40],'zlim',[-100 -40]);
title('Combined Point Cloud');
xlabel('X (m)');
ylabel('Y (m)');
zlabel('Z (m)');
```



Bundle Adjustment

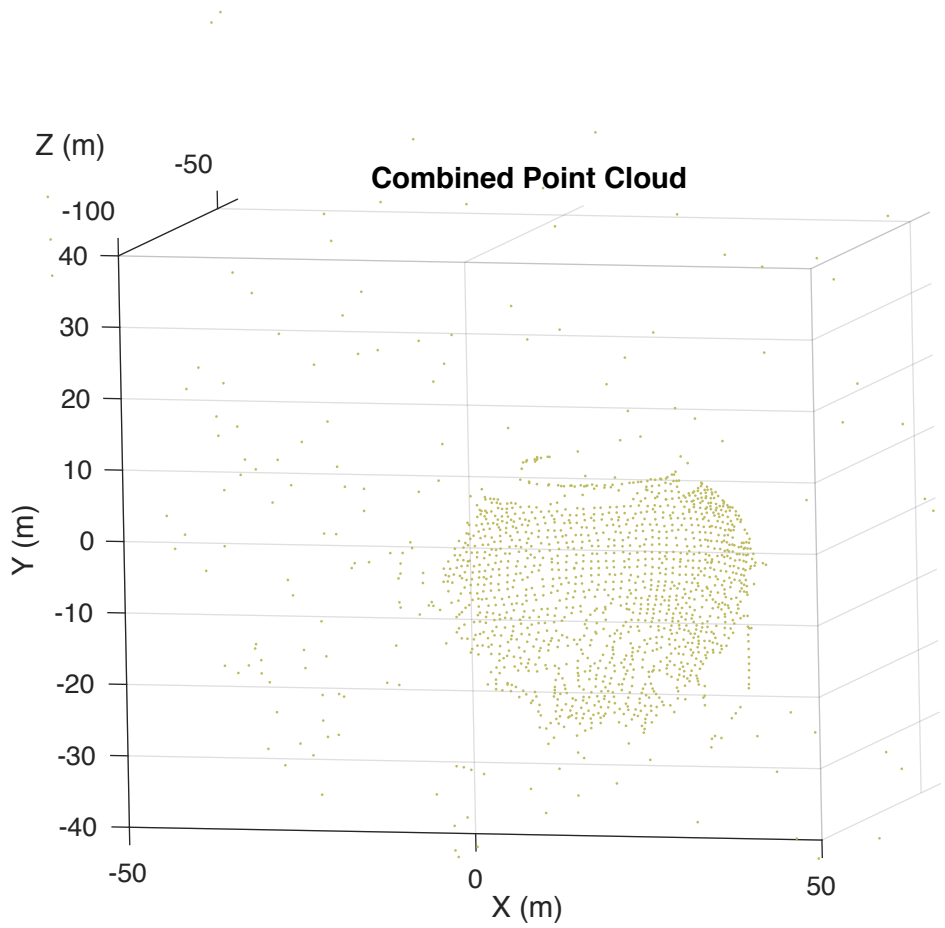
```
tracks = findTracks(vSet);

% Find point tracks across all views.
camPoses = poses(vSet);

% Triangulate locations for the world points
rWorldPts =  triangulateMultiview(tracks, camPoses, cameraParams);

% Refine the world points
[rWorldPts, camPoses, reprojectionErrors] = bundleAdjustment(...
   rWorldPts, tracks, camPoses, cameraParams, 'FixedViewId', 1, ...
    'PointsUndistorted', true);
```

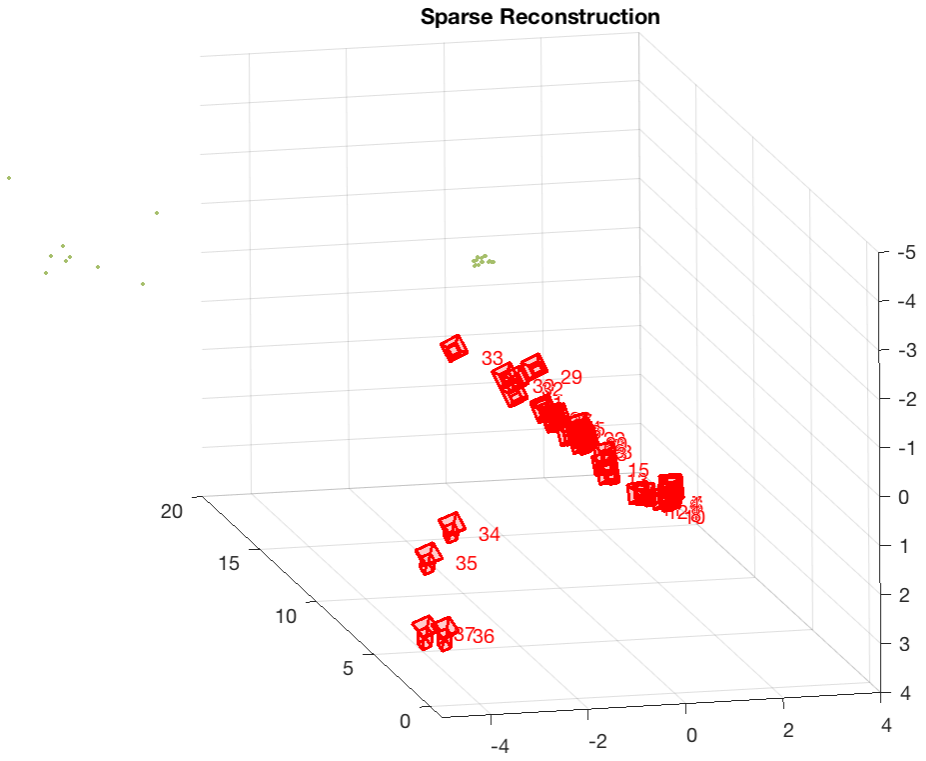


Sparse Reconstruction

```
figure;
plotCamera(camPoses, 'Size', 0.2);
hold on

%Exclude noisy 3-D world points.
good_ind = (reprojectionErrors < 5);

% Display the dense 3-D world points.
pcshow(rWorldPts(good_ind, :), 'VerticalAxis', 'y', 'VerticalAxisDir', 'down', ...
    'MarkerSize', 45);
grid on
hold off

% Specify the viewing volume.
loc1 = camPoses.Location{1};
xlim([loc1(1)-5, loc1(1)+4]);
ylim([loc1(2)-5, loc1(2)+4]);
zlim([loc1(3)-1, loc1(3)+20]);
camorbit(0, -30);
title('Sparse Reconstruction');
```

Point Cloud (another angle)





Published with MATLAB® R2016b

Sparse Recon (another angle)