3-D 点阵过程

点阵经常用于测量物体表面。他们应用于机器人的航行和感觉，深度测量，立体视觉，视觉校准和先进的司机助手系统。电脑视觉系统的工具箱里的算法里提供的关于点阵的功能有下载样本，denoise，和变换点云。工具箱也有点云校准，3-D几何形状的绘制和读、写、存储、演示和对比点阵的能力。你也可以融合多元点云去用ICP算法重新构建一个3-D视觉。

从PLY或PCD中读取点阵文件

ptCloud = pcread(filename)

描述：ptCloud = pcread(filename) 从PLY或PCD中读取一个输入的文件名具体表示的向量点阵。这个功能返回到pointCloud

输入的变量：文件名

文件名，就像是一个向量。输入的文件类型必须是PLY或PCD格式的文件

PLY格式，和斯坦福三角格式从3D折像器储存3D数据一样著名。它是一种被描述成收集多边形的图像储存格式。PLY文件包括一个header，跟着一系列顶点和一系列多边形。Header具体表明了在这个文件里有多少个定点和多边形。它也表明了每个顶点之间的关联属性，比如(x,y,z)坐标系，垂线和颜色。文件格式包括两个副格式：ASCII图像和一个为了压缩储存和加速保存和加载的一个二进制版本？两种格式的header都是ASCII文本，只有数值数据不同。想了解更多PLY文件格式细节，请看目录PLY file。

PCD格式同样是储存3D数据。它是被一个被广泛应用的用于提供更多点阵需求的图书馆作者创作的。

Note：这个功能只能支持在0.7版本存储的PCD格式。并且只支持header输入是COUNT输入是1的，不支持具体描述的

输出变量：点阵

存储的点阵包括以下的PLY或PCD格式的文件：

1. 位置，存储了x、y、z的值
2. 颜色，存储了红、绿、蓝的值
3. 性质，存储了每个点的向量属性

写一个3D点阵

ptCloud = pcread('teapot.ply');

pcshow(ptCloud);

pcwrite(ptCloud,'teapotOut','PLYFormat','binary');

从windows的kinect里规划点阵颜色

这个例子展示了如何在kinect图像里规划颜色。这个例子需要图像获取工具箱软件和Kinect相机还有相机数据线

为颜色设备创造一个系统

colorDevice = imaq.VideoDevice('kinect',1)

把彩色图像的返回类型从single改到unint8

colorDevice.ReturnedDataType = 'uint8';

为深度设备创建一个系统

depthDevice = imaq.VideoDevice('kinect',2)

预置相机

step(colorDevice);

step(depthDevice);

从设备里加载一个参考系

colorImage = step(colorDevice);

depthImage = step(depthDevice);

求出点阵

ptCloud = pcfromkinect(depthDevice,depthImage,colorImage);

预置一个点阵播放器来展现3D点阵数据。坐标轴设置适用于Kinect 显现点阵图

player = pcplayer(ptCloud.XLimits,ptCloud.YLimits,ptCloud.ZLimits,...

'VerticalAxis','y','VerticalAxisDir','down');

xlabel(player.Axes,'X (m)');

ylabel(player.Axes,'Y (m)');

zlabel(player.Axes,'Z (m)');

获取500个向量数据构建点阵

for i = 1:500

colorImage = step(colorDevice);

depthImage = step(depthDevice);

ptCloud = pcfromkinect(depthDevice,depthImage,colorImage);

view(player,ptCloud);

end

展现这个物体

release(colorDevice);

release(depthDevice);

球面点阵的结构映像（把照片映在一个球上？）

一般包括600\*600的表面

numFaces = 600;

[x,y,z] = sphere(numFaces);

用default color map来构建一个球

figure;

pcshow([x(:),y(:),z(:)]);

title('Sphere with Default Color Map');

xlabel('X');

ylabel('Y');

zlabel('Z');

加载图像结构映像

I = im2double(imread('visionteam1.jpg'));

imshow(I);

重新排列图像来配合映像

J = flipud(imresize(I,size(x)));

用颜色结构构建球体

pcshow([x(:),y(:),z(:)],reshape(J,[],3));

title('Sphere with Color Texture');

xlabel('X');

ylabel('Y');

zlabel('Z');

两种点阵的区别

加载两个被kinect设备捕获的点阵

load('livingRoom');

pc1 = livingRoomData{1};

pc2 = livingRoomData{2};

构建并设置点阵的可视点

figure

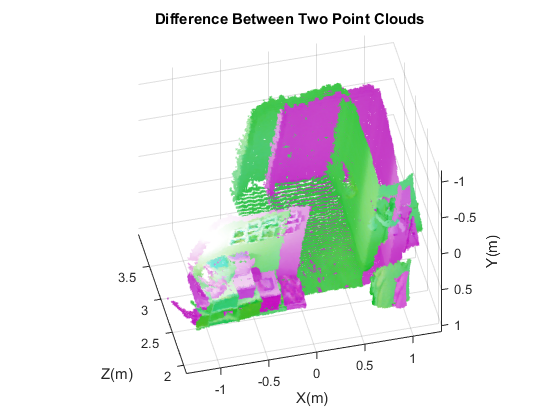
pcshowpair(pc1,pc2,'VerticalAxis','Y','VerticalAxisDir','Down')

title('Difference Between Two Point Clouds')

xlabel('X(m)')

ylabel('Y(m)')

zlabel('Z(m)')



旋转体3-D点阵

读取点阵

ptCloud = pcread(‘teapot.ply’);

定义旋转体的矩阵和3-D变换

x = pi/180;

R = [ cos(x) sin(x) 0 0

-sin(x) cos(x) 0 0

0 0 1 0

0 0 0 1];

tform = affine3d(R);

计算x、y的范围确保茶壶不会缺胳膊少腿

lower = min([ptCloud.XLimits ptCloud.YLimits]);

upper = max([ptCloud.XLimits ptCloud.YLimits]);

xlimits = [lower upper];

ylimits = [lower upper];

zlimits = ptCloud.ZLimits;

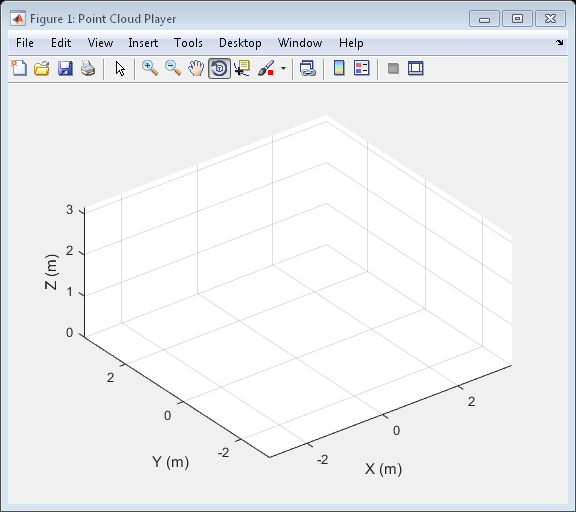
创建一个播放器并定制播放器坐标轴标签

player = pcplayer(xlimits,ylimits,zlimits);

xlabel(player.Axes,'X (m)');

ylabel(player.Axes,'Y (m)');

zlabel(player.Axes,'Z (m)');



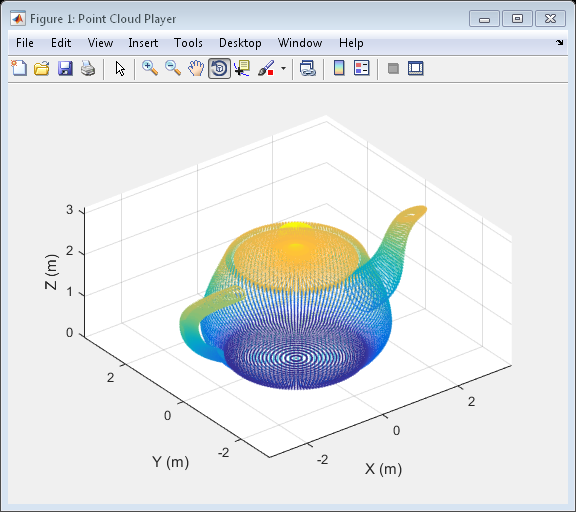
绕着z轴旋转茶壶

for i = 1:360

ptCloud = pctransform(ptCloud,tform);

view(player,ptCloud);

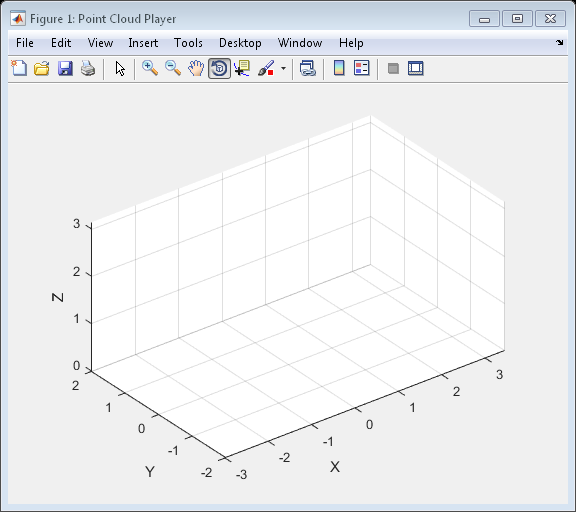
end



隐藏或显示3D点阵数据

创建一个播放器并定制坐标轴标签

player = pcplayer(ptCloud.XLimits,ptCloud.YLimits,ptCloud.ZLimits);



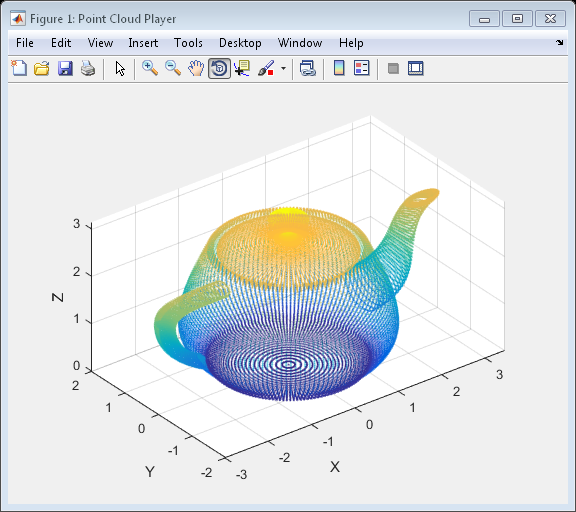
隐藏数据

Hide(player)

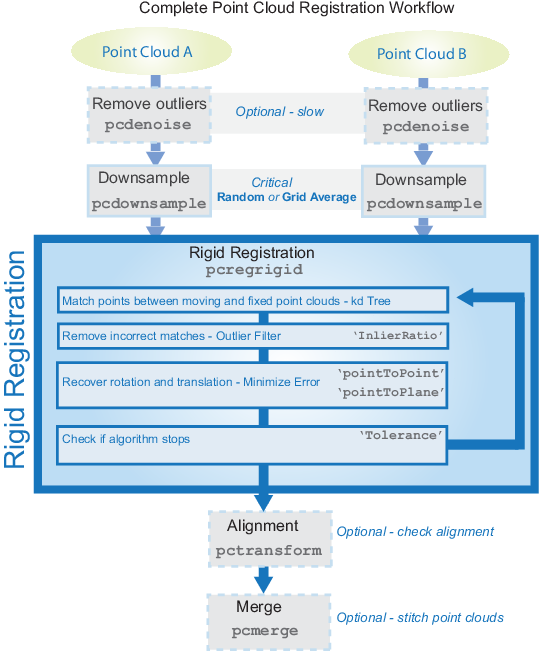
展示数据

Show(player)

View(player,ptCloud);



点阵校准的工作流程



校准两个点阵

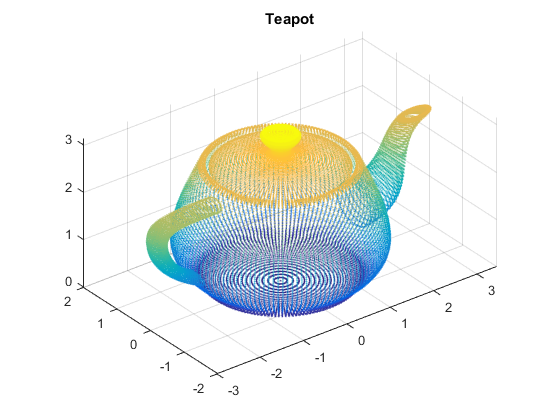
读取一个点阵数据

ptCloud = pcread('teapot.ply');

figure

pcshow(ptCloud);

title('Teapot');



创建一个变换目标，沿着z旋转30度并位移[5,5,10]

A = [cos(pi/6) sin(pi/6) 0 0; ...

-sin(pi/6) cos(pi/6) 0 0; ...

0 0 1 0; ...

5 5 10 1];

tform1 = affine3d(A);

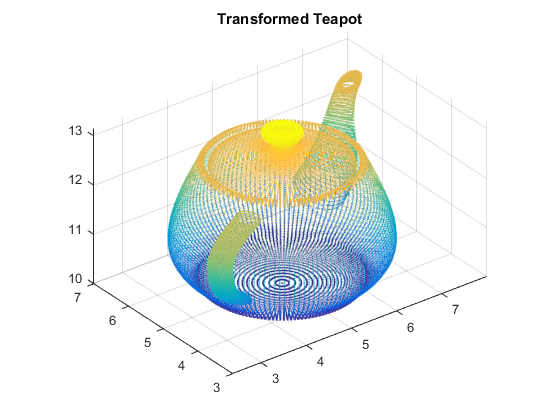
变换这个点阵

ptCloudTformed = pctransform(ptCloud,tform1);

figure

pcshow(ptCloudTformed);

title('Transformed Teapot');



应用严格的校准

tform = pcregrigid(ptCloudTformed,ptCloud,'Extrapolate',true);

比较严格严格变换后的结果

disp(tform1.T);

tform2 = invert(tform);

disp(tform2.T);

0.8660 0.5000 0 0

-0.5000 0.8660 0 0

0 0 1.0000 0

5.0000 5.0000 10.0000 1.0000

0.8660 0.5000 0.0000 0

-0.5000 0.8660 0.0000 0

0.0000 -0.0000 1.0000 0

5.0000 5.0000 10.0000 1.0000

3-D点阵的校准和连缀

这个例子展示了如何用ICP算法混合多元点阵来重新构建一个3-D画面

概述

这个例子把kinect捕获的点阵连缀并构建成了一个大的3-D画面。这个例子应用了ICP的两个相邻点阵。这种类型的重构可以被用于构建3D物体模型或3D世界地图，同时包括定位和图像

校准两个点阵

dataFile = fullfile(toolboxdir('vision'), 'visiondata', 'livingRoom.mat');

load(dataFile);

% Extract two consecutive point clouds and use the first point cloud as

% reference.

ptCloudRef = livingRoomData{1};

ptCloudCurrent = livingRoomData{2};

校准的质量取决于数据的noise和ICP算法的最初设置。你可以应用之前的步骤来过滤noise或者为你的数据设置一个合适的初始值。这里，运行含有a box grid filter的样本里的之前过程的数据，并将grid filter的大小设置成10cm。the grid filter将点阵的空间分成无数小方块。每个方块里的点被混合进一个单独的取他们x、y、z平均值的输出点。

gridSize = 0.1;

fixed = pcdownsample(ptCloudRef, 'gridAverage', gridSize);

moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize);

% Note that the downsampling step does not only speed up the registration,

% but can also improve the accuracy.

为了校准两个点阵，我们用ICP算法来估算下载样本文件上的严格的变换。我们用第一个点阵做参考，之后对原始的第二个点阵进行大致变换。我们需要另与第一个图像重合的点融合

找到严格的能使第二个点阵和第一个点阵校准上的变换，用它变换第二个点阵，使它和第一个点阵拥有相同的系统

tform = pcregrigid(moving, fixed, 'Metric','pointToPlane','Extrapolate', true);

ptCloudAligned = pctransform(ptCloudCurrent,tform);

我们现在可以用校准数据创建一个世界画面。重合的区域用1.5cm的box grid filter过滤掉。增加融合的大小来缩减点阵最终呈像的储存空间的需求，并缩减融合大小来增加图像的分解？？？

mergeSize = 0.015;

ptCloudScene = pcmerge(ptCloudRef, ptCloudAligned, mergeSize);

% Visualize the input images.

figure

subplot(2,2,1)

imshow(ptCloudRef.Color)

title('First input image')

drawnow

subplot(2,2,3)

imshow(ptCloudCurrent.Color)

title('Second input image')

drawnow

% Visualize the world scene.

subplot(2,2,[2,4])

pcshow(ptCloudScene, 'VerticalAxis','Y', 'VerticalAxisDir', 'Down')

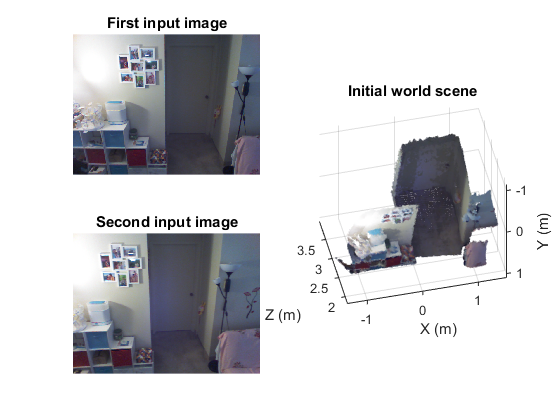
title('Initial world scene')

xlabel('X (m)')

ylabel('Y (m)')

zlabel('Z (m)')

drawnow



连接两个连续的点阵

通过重复进行之前相同的步骤来链接点阵来构建一个大的3D画面。用第一个点阵去建立一个参考系。将每个点阵都向标准参考系变换。这个变换过程是一个成对数倍增的变换。

% Store the transformation object that accumulates the transformation.

accumTform = tform;

figure

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

title('Updated world scene')

% Set the axes property for faster rendering

hAxes.CameraViewAngleMode = 'auto';

hScatter = hAxes.Children;

for i = 3:length(livingRoomData)

ptCloudCurrent = livingRoomData{i};

% Use previous moving point cloud as reference.

fixed = moving;

moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize);

% Apply ICP registration.

tform = pcregrigid(moving, fixed, 'Metric','pointToPlane','Extrapolate', true);

% Transform the current point cloud to the reference coordinate system

% defined by the first point cloud.

accumTform = affine3d(tform.T \* accumTform.T);

ptCloudAligned = pctransform(ptCloudCurrent, accumTform);

% Update the world scene.

ptCloudScene = pcmerge(ptCloudScene, ptCloudAligned, mergeSize);

% Visualize the world scene.

hScatter.XData = ptCloudScene.Location(:,1);

hScatter.YData = ptCloudScene.Location(:,2);

hScatter.ZData = ptCloudScene.Location(:,3);

hScatter.CData = ptCloudScene.Color;

drawnow('limitrate')

end

% During the recording, the Kinect was pointing downward. To visualize the

% result more easily, let's transform the data so that the ground plane is

% parallel to the X-Z plane.

angle = -pi/10;

A = [1,0,0,0;...

0, cos(angle), sin(angle), 0; ...

0, -sin(angle), cos(angle), 0; ...

0 0 0 1];

ptCloudScene = pctransform(ptCloudScene, affine3d(A));

pcshow(ptCloudScene, 'VerticalAxis','Y', 'VerticalAxisDir', 'Down', ...

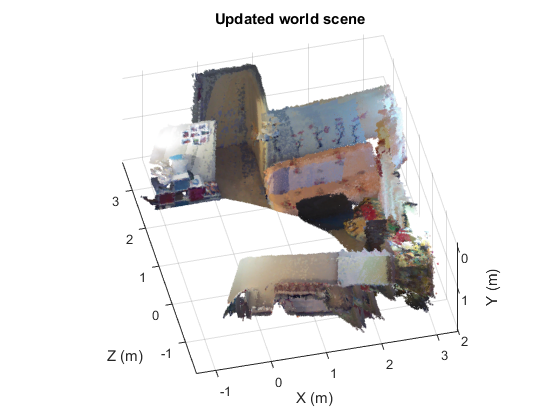
'Parent', hAxes)

title('Updated world scene')

xlabel('X (m)')

ylabel('Y (m)')

zlabel('Z (m)')



从点阵中看透多平面

展现并标注点阵

figure

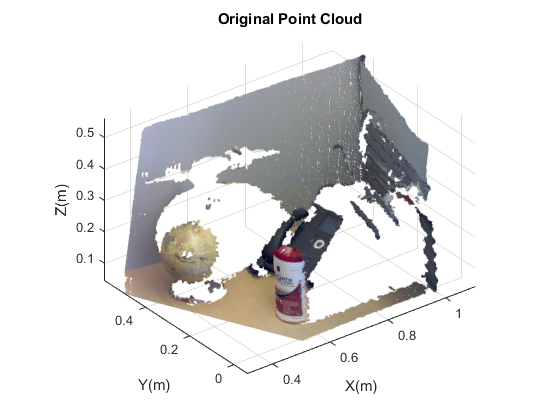
pcshow(ptCloud)

xlabel('X(m)')

ylabel('Y(m)')

zlabel('Z(m)')

title('Original Point Cloud')



设置点到平面的最大距离（2cm）

maxDistance = 0.02;

设置平面的向量属性

referenceVector = [0,0,1];

设置最大的角距离为5度

maxAngularDistance = 5;

看第一个平面，标签并将它从第一个点阵里抽出来

[model1,inlierIndices,outlierIndices] = pcfitplane(ptCloud,...

maxDistance,referenceVector,maxAngularDistance);

plane1 = select(ptCloud,inlierIndices);

remainPtCloud = select(ptCloud,outlierIndices);

设置相关的区域来限定寻找第二个平面，左边的墙

roi = [-inf,inf;0.4,inf;-inf,inf];

sampleIndices = findPointsInROI(remainPtCloud,roi);

找到并从点阵中将它抽出

[model2,inlierIndices,outlierIndices] = pcfitplane(remainPtCloud,...

maxDistance,'SampleIndices',sampleIndices);

plane2 = select(remainPtCloud,inlierIndices);

remainPtCloud = select(remainPtCloud,outlierIndices);

构建两个平面和这个点阵

figure

pcshow(plane1)

title('First Plane')

figure

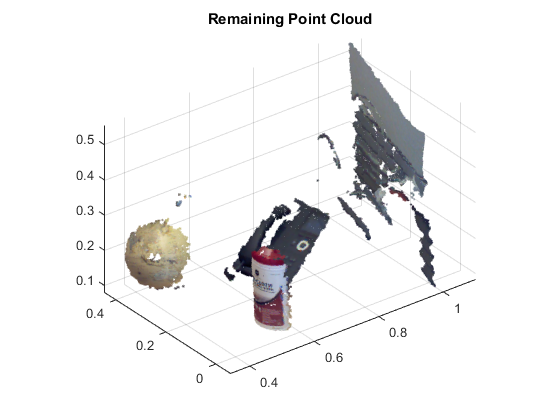
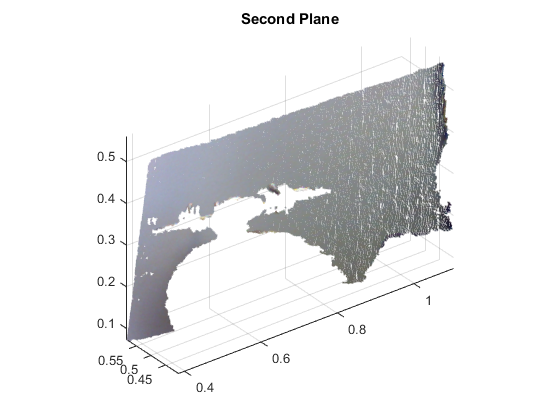
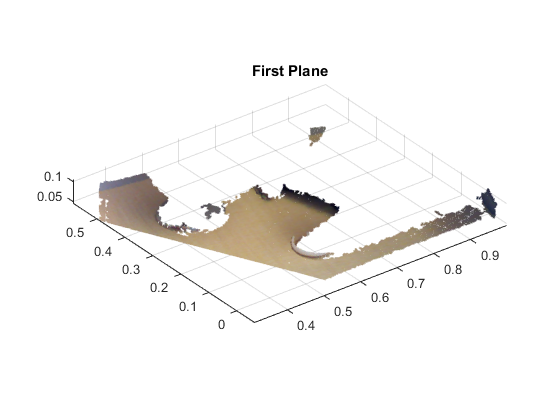
pcshow(plane2)

title('Second Plane')

figure

pcshow(remainPtCloud)

title('Remaining Point Cloud')



从点阵中抽取柱体

展示点阵

figure

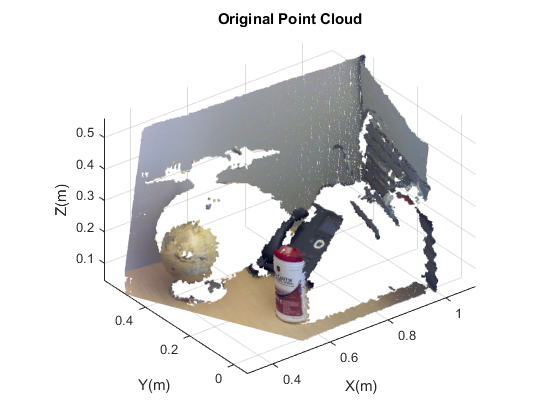
pcshow(ptCloud)

xlabel('X(m)')

ylabel('Y(m)')

zlabel('Z(m)')

title('Original Point Cloud')



设置点到柱体的最大值

maxDistance = 0.005;

设置搜索的区域范围

roi = [0.4,0.6,-inf,0.2,0.1,inf];

sampleIndices = findPointsInROI(ptCloud,roi);

设置方向限制

referenceVector = [0,0,1];

发现柱体并将它用详细的内层点抽取出来

[model,inlierIndices] = pcfitcylinder(ptCloud,maxDistance,...

referenceVector,'SampleIndices',sampleIndices);

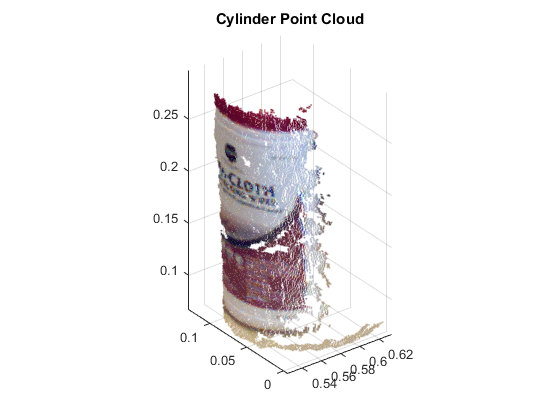
pc = select(ptCloud,inlierIndices);

构建抽出的柱体

figure

pcshow(pc)

title('Cylinder Point Cloud')



在点阵中发现球体

展示原始点阵

figure

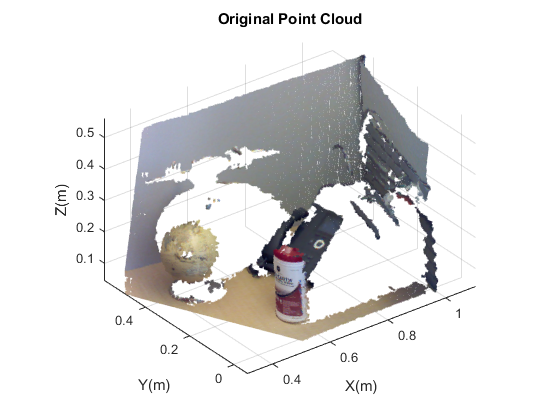
pcshow(ptCloud)

xlabel('X(m)')

ylabel('Y(m)')

zlabel('Z(m)')

title('Original Point Cloud')



设置点到球体的最大值为1cm

maxDistance = 0.01;

设置roi来限制搜索范围

roi = [-inf,0.5,0.2,0.4,0.1,inf];

sampleIndices = findPointsInROI(ptCloud,roi);

发现球并将它从点阵中抽取出来

[model,inlierIndices] = pcfitsphere(ptCloud,maxDistance,...

'SampleIndices',sampleIndices);

globe = select(ptCloud,inlierIndices);

将它构建出来

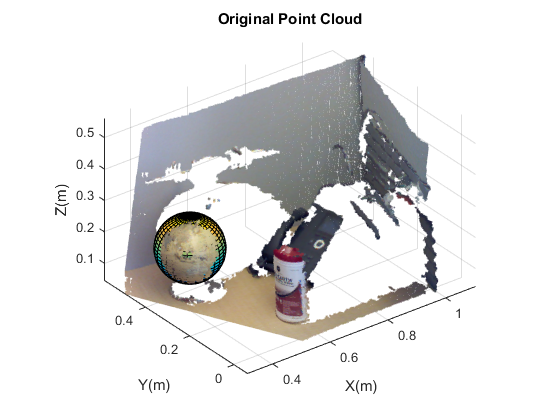
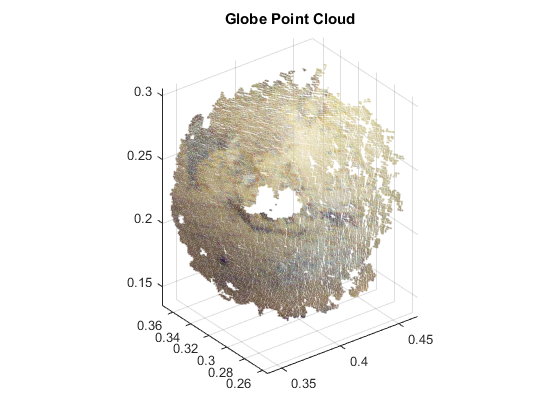
hold on

plot(model)

figure

pcshow(globe)

title('Globe Point Cloud')



点阵的应用

1. 点阵校准流程（前面有）
2. 校准和连接（前面有）
3. 用box grid filter来融合两个同样的点阵

创建两个同样的点阵

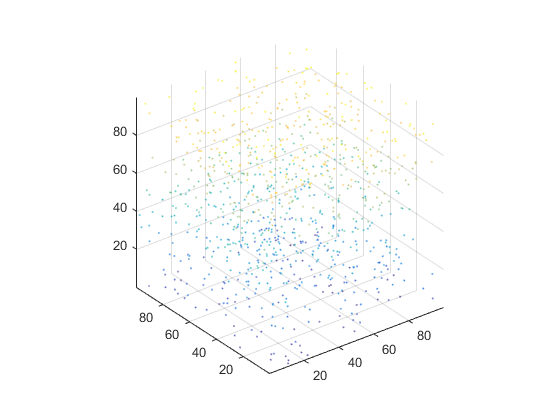
ptCloudA = pointCloud(100\*rand(1000,3));

ptCloudB = copy(ptCloudA);

融合两个点阵

ptCloud = pcmerge(ptCloudA,ptCloudB,1);

pcshow(ptCloud);



1. 将不需要的部分移出有杂志的点阵

创建一个点阵

gv = 0:0.01:1;

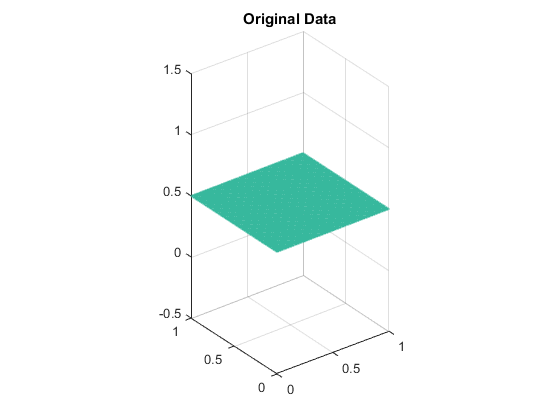
[X,Y] = meshgrid(gv,gv);

ptCloud = pointCloud([X(:),Y(:),0.5\*ones(numel(X),1)]);

figure

pcshow(ptCloud);

title('Original Data');



加入形状相同的随机的noise

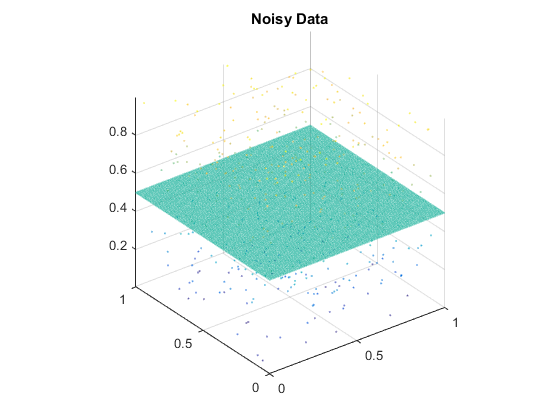
noise = rand(500, 3);

ptCloudA = pointCloud([ptCloud.Location; noise]);

figure

pcshow(ptCloudA);

title('Noisy Data');



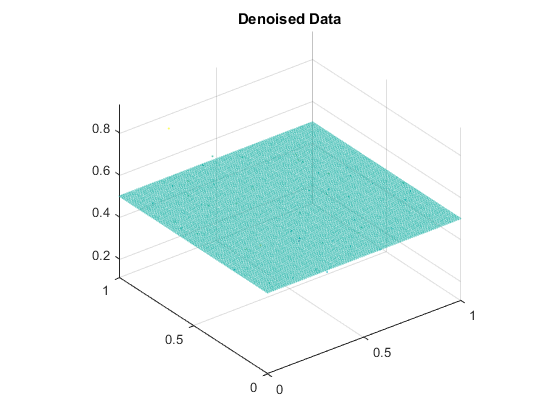
移出不需要的

ptCloudB = pcdenoise(ptCloudA);

figure;

pcshow(ptCloudB);

title('Denoised Data');



1. 用box grid filter加载点阵样本

读取点阵

ptCloud = pcread('teapot.ply');

设置3-D 分辨率（0.1\*0.1\*0,1）

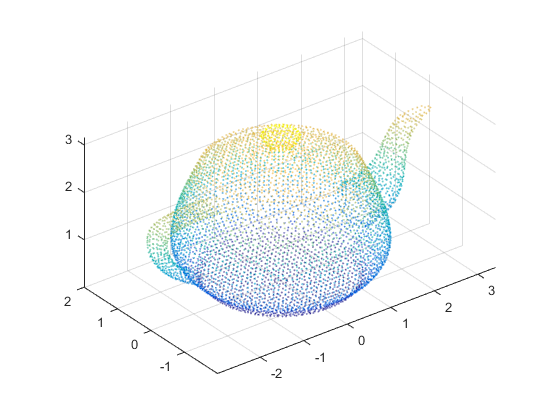
gridStep = 0.1;

ptCloudA = pcdownsample(ptCloud,'gridAverage',gridStep);

看样本数据

figure;

pcshow(ptCloudA);



和用合适的步长的点阵比较

stepSize = floor(ptCloud.Count/ptCloudA.Count);

indices = 1:stepSize:ptCloud.Count;

ptCloudB = select(ptCloud, indices);

figure;

pcshow(ptCloudB);

