数字图像处理综合设计与实验报告

课题名称：三维目标检测

院　　系：人工智能与自动化学院
班　　级：硕 1909 班
姓　　名：陈智勇/沈宜帆
学　　号：M201972831/M201972832
指导老师：马 杰、郑定富

自动化学院
2019 年 12 月 20 日

目 录

一、总体方案设计

课题目的：利用 Kinect v2 传感器，检测分割室内地面、桌面典型目标，且求取其最小外接矩形盒。

1. 1 方案一

由 Kinect v2 传感器采集深度图像，不进行预处理直接将原数据送进 PointNet 进行训练，训练得出分割结果，再用 OBB 算法画出最小外接矩形盒。PointNet 是第一个直接将原始点云数据输入到神经网络进行训练网络模型，它采用最大池化层，可以直接处理无序点云数据，不需要预处理，检测精度高，而且可以完成目标识别、部件分割和语义分割的任务。PointNet 网络结构如图 1 所示。

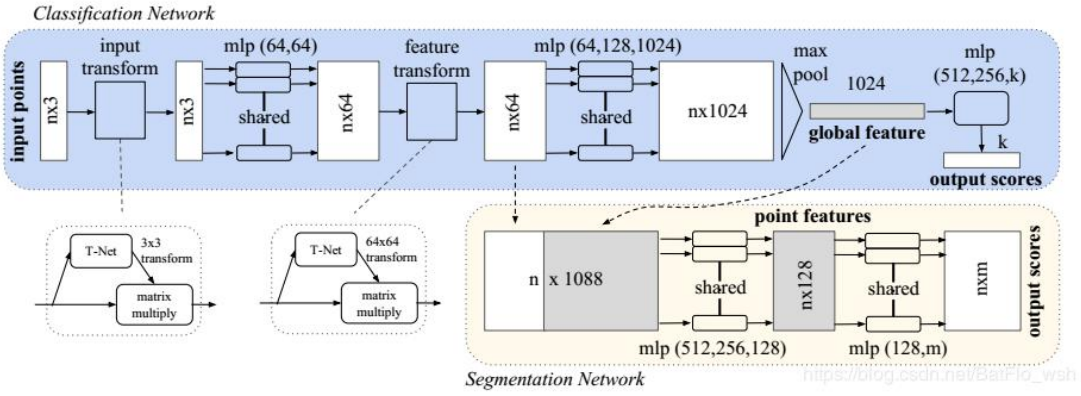

图 1　PointNet 网络结构

这种方法使用了神经网络进行训练学习，需要大量训练数据，因此精度很高，对训练集中的物体分类效果很好，但是对网络没有学习过的物体分类效果很差，识别新的物体之前需要重新学习。

1.2 方案二

方案二具体步骤如下：

（1）由 Kinect v2 深度传感器采集点云数据。

（2）预处理去除孤立点。

（3）由于点云数据太多，下采样加快计算速度。

（4）使用 RANSAC 算法找出地面，并把地面和地面上的物体分离。

（5）为了减少计算量，划分出感兴趣区域。

（6）使用欧式聚类或其它聚类方法对点云数据进行聚类。

（7）将聚类结果中点云数据最多的那一类或者几类视为目标，并打上标签。

（8）使用 OBB 算法画出目标的最小外接矩形盒，显示出来并用不同颜色显示。

这种方法思路简单，不需要大量训练数据，而且可以划分出没有见过的物体，较方案一有一定优势。

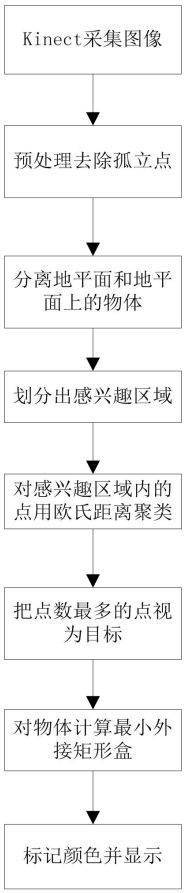

图 2 方案二流程图

1.3 方案选择

方案一具有很高的精度，但是我们训练时发现有比较多的困难，其缺点有：数据集样本少，容易过拟合；数据集的分布和实际拍摄点云分布差别大；由于网络训练时在训练集的基础上进行的，所以对于数据集中没有出现过的对象，网络可能分割不出来。我们在给定数据集的基础上训练好网络模型后，用测试集检验效果很好，但是如果将测试集之外的物体送进网络检测，发现物体并不能被很好的分割出来。

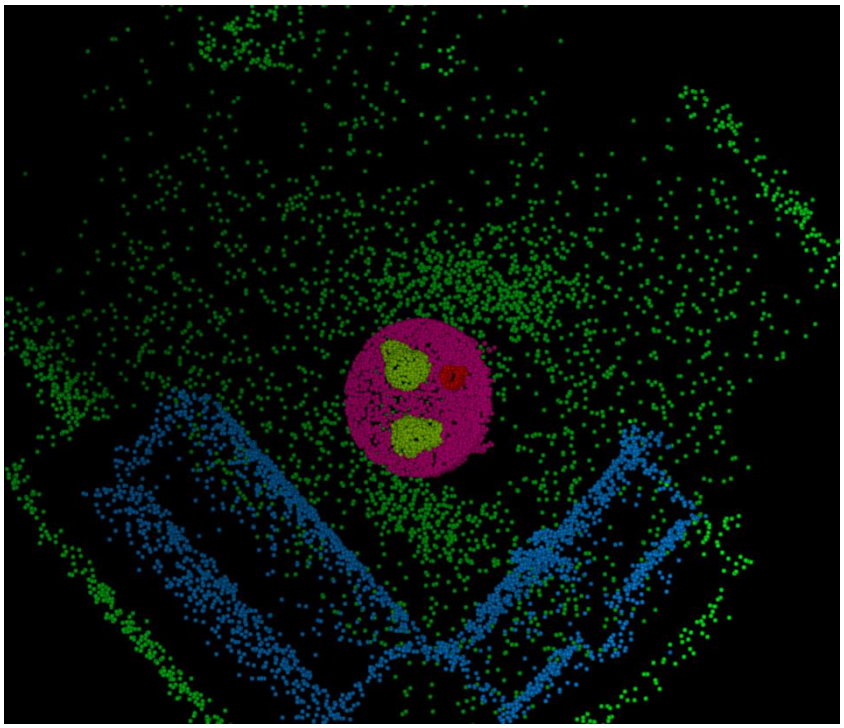

图 3 Pointnet 在测试集检测结果

从图 3 中可以看出，在测试集上，沙发、墙面、地面和茶几都被很好的分割了出来，因为采用了下采样，所以数据点比较稀疏，但是不影响分割结果。

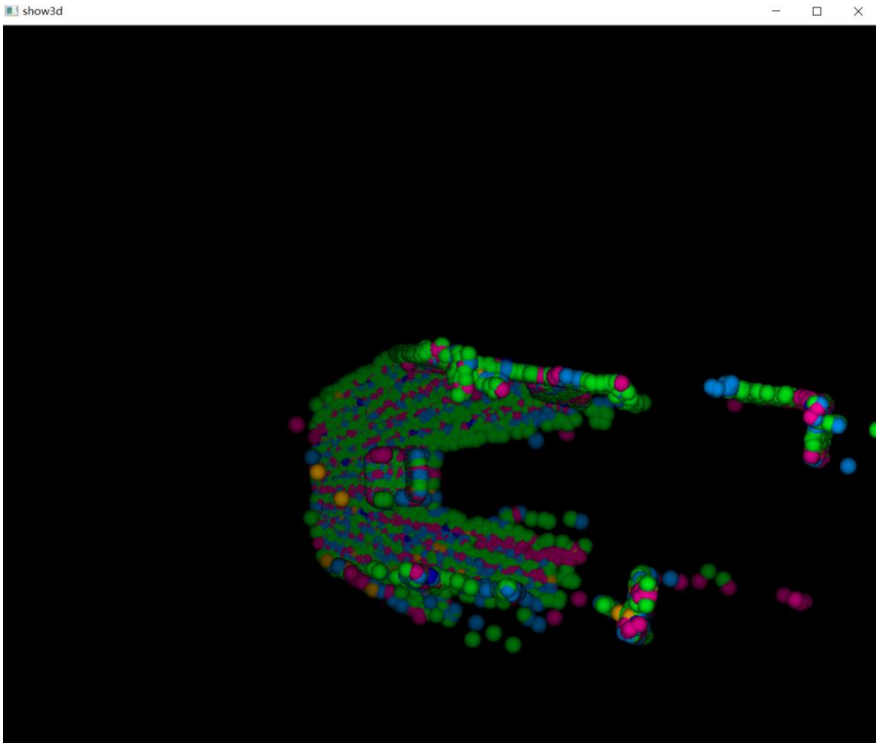

图 4 Pointnet 对实拍图像检测结果

从图 4 中可以看出，对于数据集中不存在的物体，Pointnet 不能很好的分割

出来。因此方案一不适合本次设计要求。

方案二方法简单，不需要很多的数据集来训练，可以分割出它没有见过的对象，且速度比较快，我们将在后面介绍方案二的具体细节。

本次实验我们选用方案二来实现点云对象的分割，并在 Matlab 中编写程序实现算法。

二、关键技术

2.1 寻找最大平面

在去除孤立点之后，需要划分出目标，但是如果不能确定地面就不能分割出地面上的目标，因此需要确定地面并把地面从待分割点中去除。由于在获取的点云数据中，地面处在水平面的位置，且是水平方向上的一个最大的平面，因此我们在程序中使用算法检测最大的水平面，将其视为地面，并做标记分割。

本次设计中，我们采用 RANSAC 算法，选取 x, y, z 轴中的一个方向，按照所选的方向从点云数据中找出一个最大的平面。

2.2 感兴趣区域划分

如果将所有原始的点云数据都放进算法进行检测会有很大的计算量，我们分析发现：一般情况下需要检测分割的目标比较远的背景离传感器更近，因此，如果可以去掉较远的点云数据，可以大大减少计算量，加快检测速度。

在本次设计中，我们采用了感兴趣区域划分的方法，人为划分我们感兴趣的区域，把离传感器一定距离范围内的点云数据保留，其它较远的点云数据不予计算，这一方法大大减小了计算量，加快了计算速度。

2.3 OBB（OrientedboundingBox）方向包围盒

在分割出目标之后，需要画出目标的最小外接矩形盒，本次设计中我们采用 OBB 来画目标的最小外接矩形盒。

OBB 的生成思路简单来说是根据物体表面的顶点，通过 PCA（主成分分析）获得特征向量，即 OBB 的主轴。主成分分析是一种通过正交变换，将一组可能相关的变量集合变换成一组线性不相关的变量集合，即主成分。协方差表示了两个变量之间的相关程度，协方差越小表示变量之间越独立，即相关性越小，协方

差计算公式为：

$$\mathrm{cov}(X_i,X_j) = \mathrm{E}[(X_i - \mu_i)(X_j - \mu_j)]$$

协方差矩阵为：

$$\mathrm{A} = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(x,y) & cov(y,y) & cov(y,z) \\ cov(x,z) & cov(y,z) & cov(z,z) \end{bmatrix}$$

主对角线元素表示变量方差。主对角线的元素较大则表示强信号，非对角线元素表示变量之间的协方差，较大的非对角线元素表示数据的畸变，为了减小畸变可以重新定义变量之间的线性组合，将协方差对角化。则协方差矩阵的特征向量表示 OBB 包围盒的方向，大的特征值对应大的方差，所以应该让 OBB 包围盒沿着最大特征值对应的特征向量的方向。

找到主分量之后，以主分量为 z 轴建立空间直角坐标系，计算出 OBB 包围盒的八个角点在以主分量为 z 轴建立的空间直角坐标系下的坐标，然后反变换回原来的空间直角坐标系，求出坐标。

三、图像采集系统设计

图像采集系统由如下几部分组成：室内场景，Kinect 2.0 传感器，计算机，Matlab 工具包。

Kinect 2.0 是微软在 2014 年 10 月发布的第二代 Kinect，它是一种 3D 体感摄影机(开发代号"Project Natal")，同时它导入了即时动态捕捉、影像辨识、麦克风输入、语音辨识、社群互动等功能。Kinect 2.0 具有深度传感器，已经被广泛应用于深度图像的获取，本次设计的图像采集就是通过 Matlab 中的工具包调用函数控制 Kinect 2.0 完成的。

四、程序设计和运行结果

4.1 Kinect 采集数据

我们使用 Matlab 编程。在 Matlab 中 Kinect 图像采集的硬件工具包即可调用函数打开 Kinect 2.0 采集数据。设计思路：首先使用 videoinput()指定视频输入对象，程序中指定 Kinect 为视频输入对象；由于我们只需要单帧数据，所以指

定触发器帧率为 1，即一次只获取一帧图像，且触发重复次数为 1，以确保每次运行只获取一帧图像；接着使用 triggerconfig(Depth,'manual')手动触发传感器；启动深度传感器获取数据并保存深度数据到 imgdepth；用 pcfromkinect()函数从深度对象中提取点云数据保存到 ptCloud；停止 Kinect 相机。

在实验中，我们采集了几幅不同物体的图像。

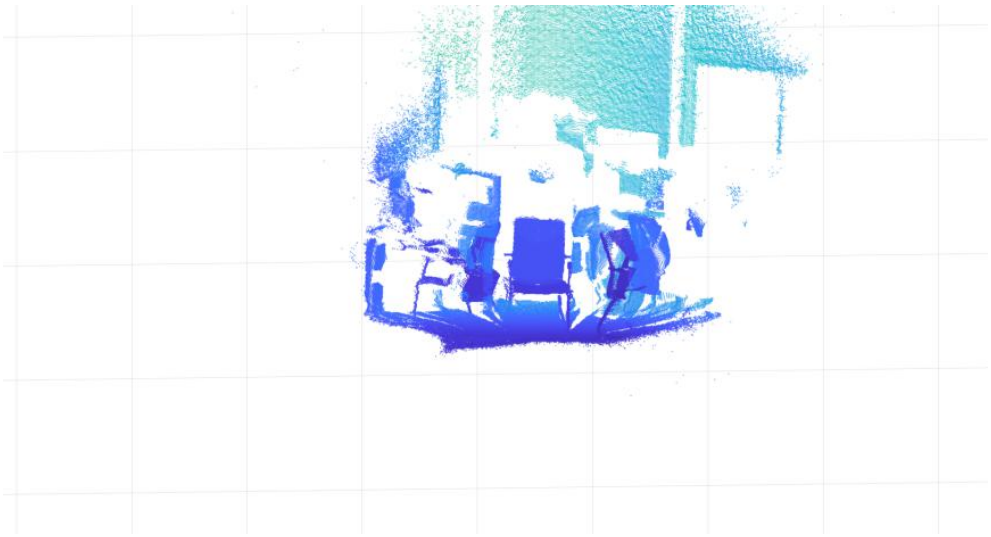

图 5 室内地面典型目标（椅子）深度图像

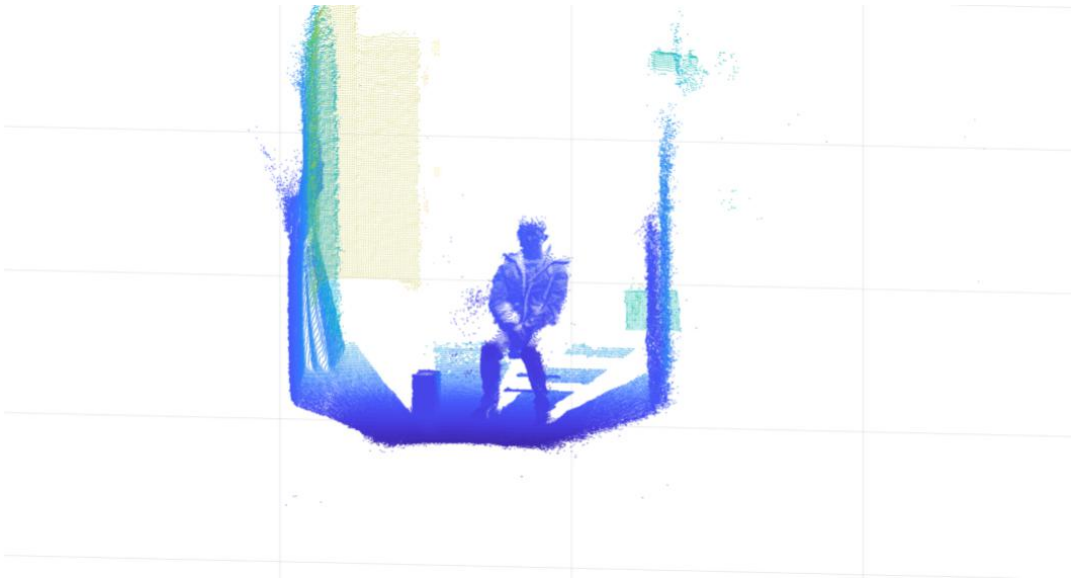

图 6 特定目标（盒子和人）深度图像

程序如下：

```
%% kinect 深度图像采集
% 创建深度传感器对象
```

```
Depth = videoinput('kinect', 2);

% 设置触发器帧数为 1
Depth.FramesPerTrigger = 1;
% 触发重复次数设置为 1
Depth.TriggerRepeat = 1;
% 配置 kinect 手动触发传感器
triggerconfig(Depth, 'manual');
% 启动深度传感器对象
start(Depth);
% 触发摄像头获取数据
trigger(Depth);
% 这里仅获取点云数据，抛弃其他数据 ts_depth，metaData_Depth
[imgDepth, ~, ~] = getdata(Depth);
% 从 kinect 深度对象提取点云数据
ptCloud = pcfromkinect(Depth, imgDepth);
% 停止 kinect 相机
stop(Depth);
```

4.2 预处理删除离群点并下采样

在用 Kinect 获取深度数据的的过程中不可避免会有很多离群点，在检测目标之前还需要预处理去除这些噪声，程序中使用 pcdenoise()函数去除离群点，用 no_noise 存储删除离群点后的点云数据。当然这时的点云数据可能还是很多，因此可以通过下采样来减少数据量，从而减少计算量，用 percentage 表示下采样的采样率。去离群点和下采样代码为：

```
% 删除离群点，NumNeighbors 是最近邻点数量，Threshold 是离群点阈值
[no_noise, inlierIndices, outlierIndices] = pcdenoise(patch, ...
    'NumNeighbors', 40, 'Threshold', 0.05);
% 下采样部分（为了加速使用，可以注释）
percentage = 0.4;
```

no_noise = pcdownsample(no_noise, 'random', percentage);

4.3 地平面分割

我们使用 RANSAC 算法检测和匹配地平面，需要提前确定一个待检测平面法线方向。因为地面的点云不可能全部正好处于一个平面，因此需要选取一个偏离的阈值，距离小于这个阈值就视为地平面的点，否则不是地面点。实验中我们选择的误差为 2 厘米。分割地平面代码如下：

```
%% 分割地平面和平面上的物体
% 找到地面平面并移除地面平面点。使用 RANSAC 算法检测和匹配地面平面。
% 平面的法线方向应大致沿 y 轴向上指向。所有被划分到地面的点必须
% 在拟合的地面平面的 2 厘米以内。

maxDistance = 0.02; % in meters
referenceVector = [0, 1, 0];
[~, Ground, outliers] = pcfitplane(no_noise, maxDistance, referenceVector);

% 选择不属于地平面一部分的点。
pcWithoutGround = select(no_noise, outliers);
```

4.4 对感兴趣区域中的点云进行聚类

为了减少不必要的计算，实验中我们对点云数据进行感兴趣区域划分，感兴趣区域范围可用 x=[xmin, xmax] y=[ymin ymax] z=[zmin, zmax]表示，并用欧式聚类对感兴趣区域内的点云进行聚类，设定的最小欧氏距离为 3 厘米，把点云数最多的几类别视为目标。代码如下：

```
%% 感兴趣区域划分
% 除去地面标识点后，检索 x=[xmin, xmax] y=[ymin ymax] z=[zmin, zmax] 范围内的点
roi = [-1 0.8 -1 1 1 2];
Maybe_target_index = findPointsInROI(pcWithoutGround,roi);
Maybe_target = select(pcWithoutGround, Maybe_target_index);
```

```
%% 基于欧氏距离进行分割
% 设定最小欧氏距离，分割后提取所有类别中点数量最多的当做目标
minDistance = 0.03;
% 根据欧氏距离进行分割
[labels,~] = pcsegdist(Maybe_target,minDistance);

% 对 labels 中元素出现次数排序，左边次数从上到下递减，右边对应的标签
k=sort(labels');
w=diff(find([1 diff(k)==1 1]));
LabelsSeq = sortrows([w',unique(k)'], 'descend');

% 分割结果中出现最多的两个类别当做物体
target_index_1 = find(labels==LabelsSeq(1,2));
target_1 = select(Maybe_target, target_index_1);

target_index_2 = find(labels==LabelsSeq(2,2));
target_2 = select(Maybe_target, target_index_2);
% pcshow(target);
```

4.5 计算 OBB 包围盒

代码中计算 OBB 包围盒的函数为 calc_OriBoundingBox()，该函数实际上是计算出了 OBB 包围盒的八个角点，绘制包围盒边框的的函数为 OBB_box_line()。程序如下：

```
%% 计算 OriBoundingBox
cornerPoints = calc_OriBoundingBox(double(target. Location));
boxLine = OBB_box_line(cornerPoints);
```

4.5 检测结果显示

给每一个点云数据分配颜色标签，地面点云用绿色表示，目标用红色表示，OBB 包围盒边框用黑色表示，其余背景点云用蓝色表示。检测结果如下所示：

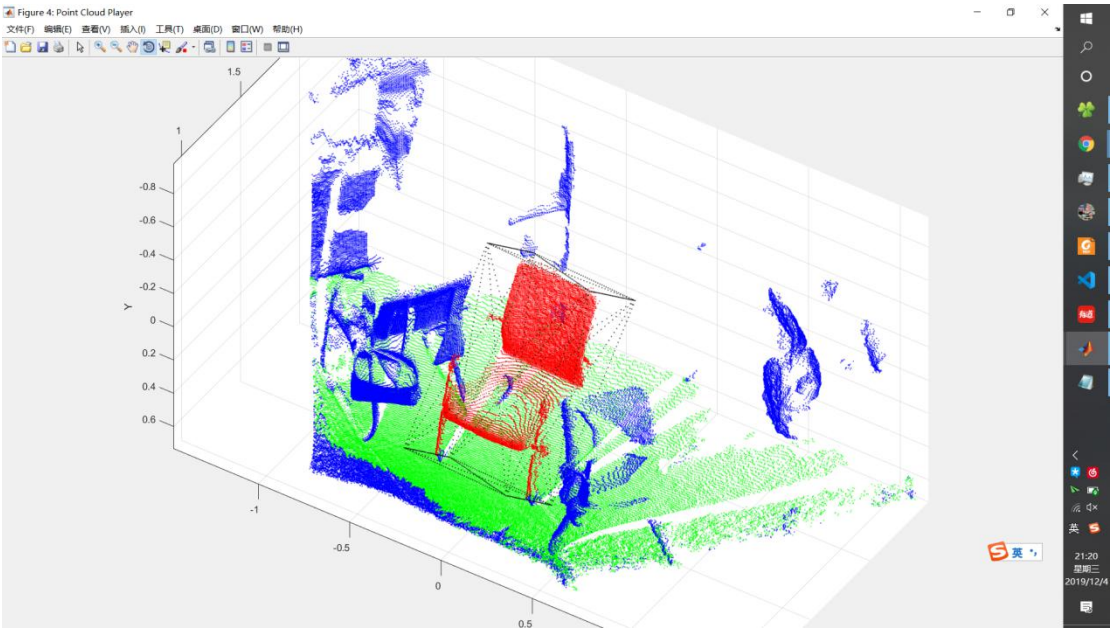

图 7 室内地面典型目标（椅子）图像分割结果图

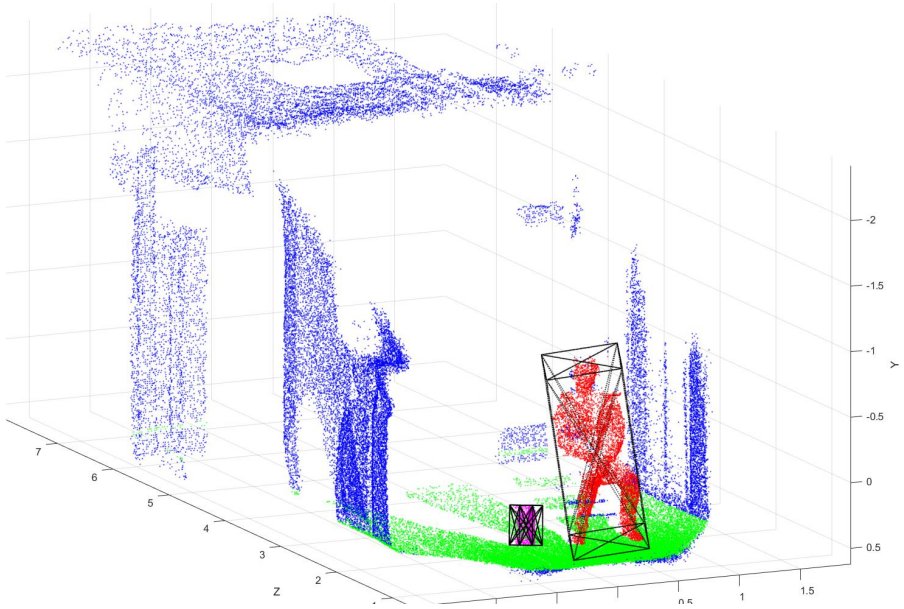

图 8 特定目标（盒子和人）图像分割结果图

代码如下：

```
%% 颜色标记
% 将颜色标签附加到点云中的每个点。使用绿色显示地面平面和红色的物体
labelSize    = [no_noise.Count, 1];
colorLabels = zeros(labelSize, 'single');
boxLineSize = [boxLine.Count, 1];
```

```
boxLineLabels = 3*ones(boxLineSize, 'single');
% 设置用于标记点的颜色表。
colors = [0 0 1; ...  % 蓝色为未标记的点(0 0 1); 指定为[R, G, B]
          0 1 0; ...  % 绿色的为地面平面点(0 1 0)
          1 0 0; ...  % 红色的为物体(1 0 0)
          0 0 0];     % 黑色的为自我附近点(0 0 0)

blueIdx  = 0; % 整个点云最初是蓝色的
greenIdx = 1; % 绿色标签（地面）
redIdx   = 2; % 红色标签（物体）
blackIdx = 3; % 黑色标签（边框）

% 标出地平面点。
colorLabels(Ground) = greenIdx;
% 根据标签对疑似目标点进行染色(索引传递到 no_noise)
colorLabels((outliers(Maybe_target_index(target_index)))) = redIdx;

%% 结果显示
% 将所有标记的点绘制到点云播放器中。使用前面设置的数字颜色标签。
player_3 = pcplayer(no_noise. XLimits, no_noise. YLimits,
no_noise. ZLimits, . . .
    'VerticalAxis', 'y', 'VerticalAxisDir', 'down');
colormap(player_3. Axes, colors)
% points1(inPlanePointIndices, :) = [];

% 直接将边框点云和标签加在后面一起显示
view(player_3, [no_noise. Location; boxLine. Location],
[colorLabels; boxLineLabels]);
title(player_3. Axes, 'Segmented Point Cloud');
```

五、实验结果及分析

5.1 实验结果

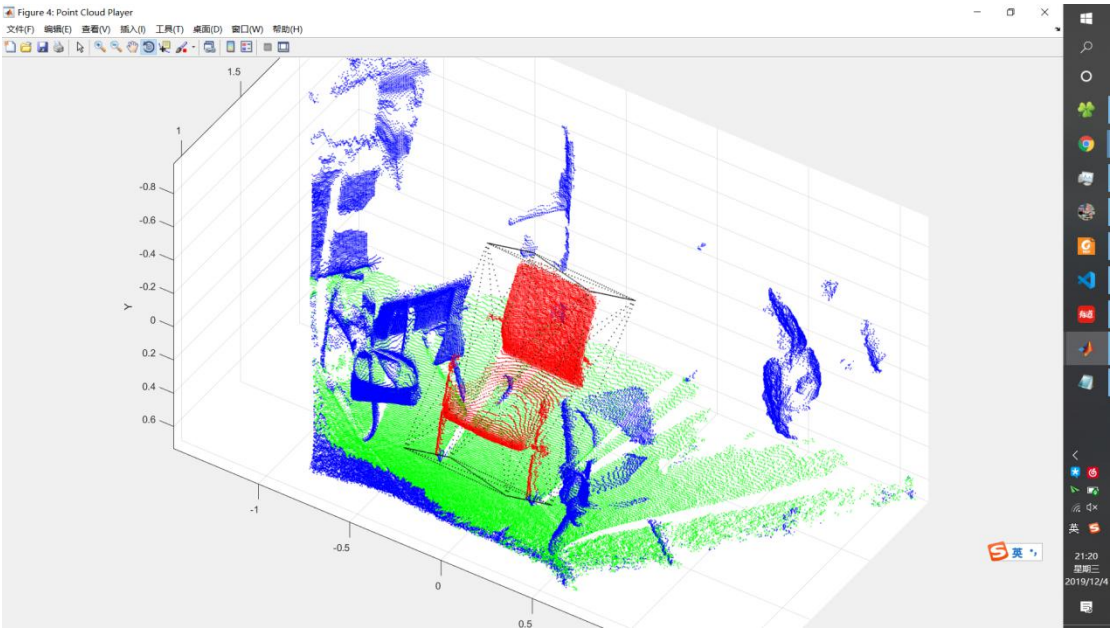

图 9 室内地面典型目标（椅子）图像分割结果图

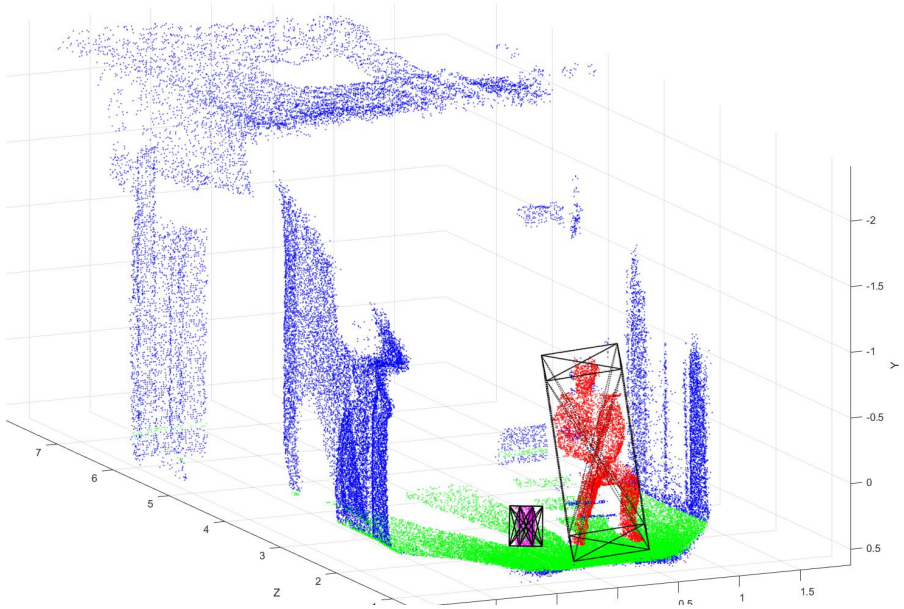

图 10 特定目标（盒子和人）图像分割结果图

5.2 结果分析

从结果图中可以看出，算法可以将点云大致分为三类：目标、地面和背景。在单目标图像中，椅子标为红色，OBB 包围盒是倾斜的，这是因为椅子有部分

被遮挡，获取深度图像时无法拍出全部目标，因此画出来的 OBB 包围盒只是对已有的数据进行包围的。

除了可以检测单目标外，算法还可以分割出多目标，图 10 演示的是图像中有特定目标（盒子和人）的分割结果，从图中可以看出，目标依然可以很好的分割出来。

六、心得体会

本学期数字图像处理综合设计与实验的课程，我们小组选择的是三维目标检测的课题。选定课题之后，组员分工合作、查阅文献，了解到目标对三维数据的几种处理方法，对 Pointnet 网络进行了复现，发现该网络模型不能满足要求后及时改变思路，采用了传统方法在 Matlab 上重新编程，最后的效果基本符合要求。在完成课题的过程中，我们将数字图像处理的理论知识运用于实际，更好地巩固了数字图像处理的理论知识体系。在实验过程中，我们还遇到了一些困难，比如该出和确定最大平面、如何检测出多个目标等等。在经过小组成员的交流协作之后，这些问题一一得到了解决，提升了我们团队协作的能力。

最后，非常感谢老师们，在我们遇到问题时候耐心地答疑解惑，在了解我们的进度和方法后给出专业指导意见，这些帮助都给我们带来了新的思路，让我们学到了更多的思考方式。

七、参考文献

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv:1612.00593, 2016.

附录

主程序代码如下：

```
%% kinect 深度图像采集
% %  创建深度传感器对象
% Depth = videoinput('kinect',2);
%
% %  设置触发器帧数为 1
% Depth.FramesPerTrigger = 1;
% %  触发重复次数设置为 1
% Depth.TriggerRepeat = 1;
% %  配置 kinect 手动触发传感器
% triggerconfig(Depth,'manual');
% %  启动深度传感器对象
% start(Depth);
% %  触发摄像头获取数据
% trigger(Depth);
% %  这里仅获取点云数据，抛弃其他数据 ts_depth，metaData_Depth
% [imgDepth, ~, ~] = getdata(Depth);
%
% %  从 kinect 深度对象提取点云数据
% ptCloud = pcfromkinect(Depth,imgDepth);
% %  停止 kinect 相机
% stop(Depth);
% pcwrite(ptCloud,'test2.ply','Encoding','binary');

%%  初始化点云显示框
%  设置垂直轴为 y 轴，垂直轴方向 down

% player = pcplayer(ptCloud.XLimits, ptCloud.YLimits, ptCloud.ZLimits,...
%         'VerticalAxis', 'y', 'VerticalAxisDir', 'down');
%
% %  设置坐标轴
% xlabel(player.Axes, 'X (m)');
% ylabel(player.Axes, 'Y (m)');
% zlabel(player.Axes, 'Z (m)');
% %  显示点云数据
% view(player, ptCloud);

%%  点云文件读取
ptCloud = pcread('test.ply');
patch = ptCloud;

%  删除离群点，NumNeighbors 是最近邻点数量，Threshold 是离群点阈值
```

```
[no_noise,inlierIndices,outlierIndices] = pcdenoise(patch,...
    'NumNeighbors', 40,'Threshold', 0.05);

% 下采样部分（为了加速使用，可以注释）
percentage = 0.4;
no_noise = pcdownsample(no_noise, 'random',percentage);

%% 分割地平面和平面上的物体
% 找到地面平面并移除地面平面点。使用 RANSAC 算法检测和匹配地面平面。
% 平面的法线方向应大致沿 y 轴向上指向。所有被划分到地面的点必须
% 在拟合的地面平面的 2 厘米以内。

maxDistance = 0.02; % in meters
referenceVector = [0, 1, 0];
[~, Ground, outliers] = pcfitplane(no_noise, maxDistance, referenceVector);

% 选择不属于地平面一部分的点。
pcWithoutGround = select(no_noise, outliers);

%% 感兴趣区域划分
% 除去地面标识点后，检索 x=[-0.5, 0.5] y=[-0.5 0.5] z=[1, 2]范围内的点
roi = [-1 0.8 -1 1 1 2];
Maybe_target_index = findPointsInROI(pcWithoutGround,roi);
Maybe_target = select(pcWithoutGround, Maybe_target_index);
%% 基于欧氏距离进行分割
% 设定最小欧氏距离，分割后提取所有类别中点数量最多的当做目标
minDistance = 0.03;
% 根据欧氏距离进行分割
[labels,~] = pcsegdist(Maybe_target,minDistance);

% 对 labels 中元素出现次数排序，左边次数从上到下递减，右边对应的标签
k=sort(labels');
w=diff(find([1 diff(k)==1 1]));
LabelsSeq = sortrows([w',unique(k)'], 'descend');

% 分割结果中出现最多的两个类别当做物体
target_index_1 = find(labels==LabelsSeq(1,2));
target_1 = select(Maybe_target, target_index_1);

target_index_2 = find(labels==LabelsSeq(2,2));
target_2 = select(Maybe_target, target_index_2);
% pcshow(target);

%% 计算 OriBoundingBox
```

```
cornerPoints_1 = calc_OriBoundingBox(double(target_1.Location));
boxLine_1 = OBB_box_line(cornerPoints_1);

cornerPoints_2 = calc_OriBoundingBox(double(target_2.Location));
boxLine_2 = OBB_box_line(cornerPoints_2);

%% 颜色标记
% 设置用于标记点的颜色表。
colors = [0 0 1; ...   % 蓝色为未标记的点(0 0 1); 指定为[R，G，B]
          0 1 0; ...   % 绿色的为地面平面点(0 1 0)
          1 0 0; ...   % 红色的为物体 1
          1 0 1; ...   % 紫色的为物体 2
          0 0 0];      % 黑色的为自我附近点(0 0 0)

blueIdx   = 0; % 整个点云最初是蓝色的
greenIdx = 1; % 绿色标签（地面）
redIdx    = 2; % 红色标签（物体 1）
purpleIdx = 3; % 紫色标签（物体 2）
blackIdx = 4; % 黑色标签（边框）
% 将颜色标签附加到点云中的每个点。使用绿色显示地面平面和红色的物体
labelSize    = [no_noise.Count, 1];
colorLabels = zeros(labelSize, 'single');

boxLineLabels_1 = blackIdx*ones([boxLine_1.Count, 1], 'single');
boxLineLabels_2 = blackIdx*ones([boxLine_2.Count, 1], 'single');


% 标出地平面点。
colorLabels(Ground) = greenIdx;
% 根据标签对疑似目标点进行染色(索引传递到 no_noise）
colorLabels((outliers(Maybe_target_index(target_index_1)))) = redIdx;
colorLabels((outliers(Maybe_target_index(target_index_2)))) = purpleIdx;

%% 结果显示
% 将所有标记的点绘制到点云播放器中。使用前面设置的数字颜色标签。
player_3 = pcplayer(no_noise.XLimits, no_noise.YLimits, no_noise.ZLimits,...
    'VerticalAxis', 'y', 'VerticalAxisDir', 'down');
colormap(player_3.Axes, colors)
% points1(inPlanePointIndices, :) = [];
% 直接将边框点云和标签加在后面一起显示
view(player_3, [no_noise.Location;boxLine_1.Location;boxLine_2.Location],...
    [colorLabels;boxLineLabels_1;boxLineLabels_2]);
title(player_3.Axes, 'Segmented Point Cloud');
```

计算最小 OBB 包围盒角点函数代码如下：

```
% @author: Svenja (st100333@stud.uni-stuttgart.de)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 拟合给定点云数据的最小 OBB 边界框角点
%
% Input:
%           点云数据点 x,y,z 坐标
%           输入格式：n*3 矩阵
% Output:
%           矩形框 8 角点 x,y,z 坐标
%           输出格式：8*3  矩阵
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function cornerpoints = calc_OriBoundingBox(data)
[n,dim] = size(data);

% 计算点云数据的凸包，得到包围点云的最小多面体
% （虽然不算也可以做，但是对于大部分情况凸包可以更好地逼近最小边界框
convH = convhull(data(:,1),data(:,2),data(:,3));

% convhull 函数求解过后，只需要得到凸包上的点就行了，相当于包裹物体的外壳
convH_points = data(convH(:),:);

% 求解协方差矩阵
nK = length(convH_points(:,1));
C = [convH_points(:,1)-sum(convH_points(:,1))/nK,convH_points(:,2)-sum(convH_points(:,2))/nK,convH_points(:,3)-sum(convH_points(:,3))/nK];
cov = C'*C;

% 利用奇异值分解得到数据矩阵的主成分
[U,V,D] = svd(cov);

% 坐标系变换成由 V 的特征向量张成的坐标系
I = [1 0 0; 0 1 0; 0 0 1];
B_traf = U'\I;
data_traf = data;
for i = 1:n
    data_traf(i,:) = data(i,:)*B_traf;
end
```

```
% 计算变换坐标系后的点云数据沿轴线方向的边框
cornerpoints = zeros(8,3);
x = data_traf(:,1);
y = data_traf(:,2);
z = data_traf(:,3);
cornerpoints(1,:) = [min(x), min(y), min(z)];
cornerpoints(2,:) = [max(x), min(y), min(z)];
cornerpoints(3,:) = [max(x), max(y), min(z)];
cornerpoints(4,:) = [min(x), max(y), min(z)];
cornerpoints(5,:) = [min(x), max(y), max(z)];
cornerpoints(6,:) = [max(x), max(y), max(z)];
cornerpoints(7,:) = [max(x), min(y), max(z)];
cornerpoints(8,:) = [min(x), min(y), max(z)];

% 将坐标系变换回原始的笛卡尔坐标系
for j = 1:8
    cornerpoints(j,:) = cornerpoints(j,:)*B_traf';
end
end
```

画线函数代码如下：

```
% 2019/12/5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 这是一个在角点之间插入大量直线分布的点云来模拟边框的函数
%
% Input:
%             矩形盒边框的 8 个角点
%             输入格式：8*3 矩阵
% Output:
%             两两角点间直线上分布的点云
%             输出格式：一个 pointCloud 类
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function box = OBB_box_line(cornerpoints)
%输入 OriBoundingBox 的 8 个角点，得到矩形框架的点云

%% 获取 8 个角点的坐标
point1 = single(cornerpoints(1, :, :));
point2 = single(cornerpoints(2, :, :));
point3 = single(cornerpoints(3, :, :));
point4 = single(cornerpoints(4, :, :));
point5 = single(cornerpoints(5, :, :));
```

```
point6 = single(cornerpoints(6, :, :));
point7 = single(cornerpoints(7, :, :));
point8 = single(cornerpoints(8, :, :));

%% 角点之间两两插值 100 个点
line1 = [linspace(point1(1), point2(1), 100)', ...
         linspace(point1(2), point2(2), 100)', ...
         linspace(point1(3), point2(3), 100)'];
line2 = [linspace(point1(1), point3(1), 100)', ...
         linspace(point1(2), point3(2), 100)', ...
         linspace(point1(3), point3(3), 100)'];
line3 = [linspace(point1(1), point4(1), 100)', ...
         linspace(point1(2), point4(2), 100)', ...
         linspace(point1(3), point4(3), 100)'];
line4 = [linspace(point1(1), point5(1), 100)', ...
         linspace(point1(2), point5(2), 100)', ...
         linspace(point1(3), point5(3), 100)'];
line5 = [linspace(point1(1), point6(1), 100)', ...
         linspace(point1(2), point6(2), 100)', ...
         linspace(point1(3), point6(3), 100)'];
line6 = [linspace(point1(1), point7(1), 100)', ...
         linspace(point1(2), point7(2), 100)', ...
         linspace(point1(3), point7(3), 100)'];
line7 = [linspace(point1(1), point8(1), 100)', ...
         linspace(point1(2), point8(2), 100)', ...
         linspace(point1(3), point8(3), 100)'];
line8 = [linspace(point2(1), point3(1), 100)', ...
         linspace(point2(2), point3(2), 100)', ...
         linspace(point2(3), point3(3), 100)'];
line9 = [linspace(point2(1), point4(1), 100)', ...
         linspace(point2(2), point4(2), 100)', ...
         linspace(point2(3), point4(3), 100)'];
line10= [linspace(point2(1), point5(1), 100)', ...
         linspace(point2(2), point5(2), 100)', ...
         linspace(point2(3), point5(3), 100)'];
line11= [linspace(point2(1), point6(1), 100)', ...
         linspace(point2(2), point6(2), 100)', ...
         linspace(point2(3), point6(3), 100)'];
line12= [linspace(point2(1), point7(1), 100)', ...
         linspace(point2(2), point7(2), 100)', ...
         linspace(point2(3), point7(3), 100)'];
line13= [linspace(point2(1), point8(1), 100)', ...
         linspace(point2(2), point8(2), 100)', ...
         linspace(point2(3), point8(3), 100)'];
```

```
line14= [linspace(point3(1), point4(1), 100)', ...
         linspace(point3(2), point4(2), 100)', ...
         linspace(point3(3), point4(3), 100)'];
line15= [linspace(point3(1), point5(1), 100)', ...
         linspace(point3(2), point5(2), 100)', ...
         linspace(point3(3), point5(3), 100)'];
line16= [linspace(point3(1), point6(1), 100)', ...
         linspace(point3(2), point6(2), 100)', ...
         linspace(point3(3), point6(3), 100)'];
line17= [linspace(point3(1), point7(1), 100)', ...
         linspace(point3(2), point7(2), 100)', ...
         linspace(point3(3), point7(3), 100)'];
line18= [linspace(point3(1), point8(1), 100)', ...
         linspace(point3(2), point8(2), 100)', ...
         linspace(point3(3), point8(3), 100)'];
line19= [linspace(point4(1), point5(1), 100)', ...
         linspace(point4(2), point5(2), 100)', ...
         linspace(point4(3), point5(3), 100)'];
line20= [linspace(point4(1), point6(1), 100)', ...
         linspace(point4(2), point6(2), 100)', ...
         linspace(point4(3), point6(3), 100)'];
line21= [linspace(point4(1), point7(1), 100)', ...
         linspace(point4(2), point7(2), 100)', ...
         linspace(point4(3), point7(3), 100)'];
line22= [linspace(point4(1), point8(1), 100)', ...
         linspace(point4(2), point8(2), 100)', ...
         linspace(point4(3), point8(3), 100)'];
line23= [linspace(point5(1), point6(1), 100)', ...
         linspace(point5(2), point6(2), 100)', ...
         linspace(point5(3), point6(3), 100)'];
line24= [linspace(point5(1), point7(1), 100)', ...
         linspace(point5(2), point7(2), 100)', ...
         linspace(point5(3), point7(3), 100)'];
line25= [linspace(point5(1), point8(1), 100)', ...
         linspace(point5(2), point8(2), 100)', ...
         linspace(point5(3), point8(3), 100)'];
line26= [linspace(point6(1), point7(1), 100)', ...
         linspace(point6(2), point7(2), 100)', ...
         linspace(point6(3), point7(3), 100)'];
line27= [linspace(point6(1), point8(1), 100)', ...
         linspace(point6(2), point8(2), 100)', ...
         linspace(point6(3), point8(3), 100)'];
line28= [linspace(point7(1), point8(1), 100)', ...
         linspace(point7(2), point8(2), 100)', ...
```

```
            linspace(point7(3), point8(3), 100)'];
%% 集合所有线成为一个点云矩形框
lineCloud = [line1; line2; line3; line4; line5;...
                line6; line7; line8; line9; line10;...
                line11;line12;line13;line14;line15;...
                line16;line17;line18;line19;line20;...
                line21;line22;line23;line24;line25;...
                line26;line27;line28];
box = pointCloud(lineCloud);
end
```