Advanced Vision
Assignment 3

Thorvaldur Helgason (s1237131)
Hywel Dunn-Davies (s.....)

March 20, 2013

1 Introduction

This assignment involved analyzing 21 consecutive image frames captured with a Kinect sensor. These frames contained both RGP pixel intensity values and XYZ range values. The images showed many different angles of a cabinet containing an open book on top. Our goal was the following:

(i) Extract the three planes that formed the front of the rectangular cabinet and compute their plane equations.

(ii) Extract the set of points belonging to the book.

(iii) Compute the translation and rotation of the three planes between frames to register all views together.

(iv) Fuse all views of the book together using the translation and rotation computed in step (ii).

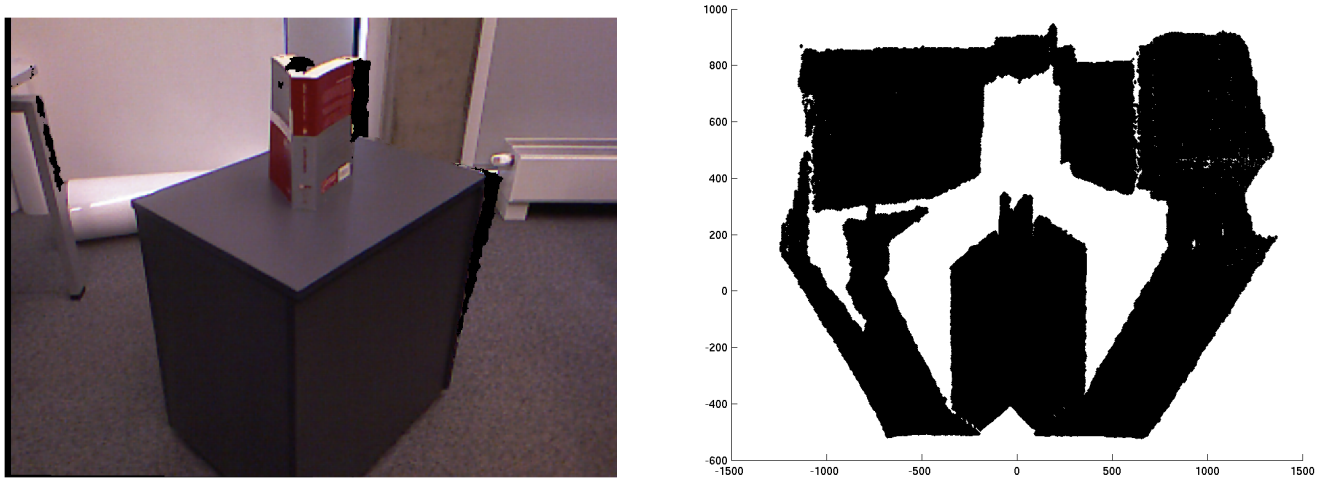
At last we evaluated our approach by looking at our 3D fused view of the book, computing the determinant of the covariance matrix between the same point in all frames, and compute the average and standard deviation of surface normal angles which have been corrected in terms of rotation.

2 Methods

2.1 Pre-processing

Before extracting the planes, we preprocessed the range data to eliminate the background since the only things we are interested are the cabinet and the book, which are in the foreground. After having observed the range data, we decided to eliminate all pixels with the range $Z > 1400$, which removed the background. Additionally we transformed the range data in X-Y space to a pointcloud which made it easier to work with and visualize (see Figure 1).

For registering views we decided to use frame number 17 as our foundation frame (see Figure 1) which we used for computing the rotation and translation between that frame and all the others.



(a) (b)

Figure 1: The image (a) and X-Y range data converted to a point cloud (b) for our foundation frame.

2.2 Plane Extraction

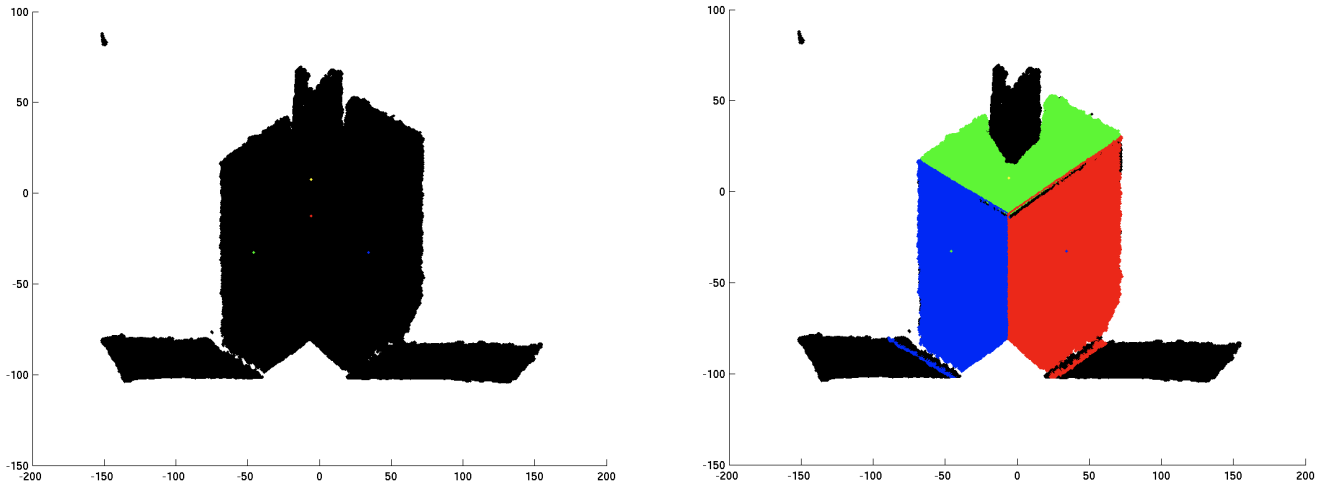
2.2.1 Patch Selection

We used region growing for extracting the points on the three planes, and managed to speed up that procedure considerably by approximating the position of the initial patches on each of the three planes. To do that we extracted the closest point in terms of Z in the range data, which was the corner of the cabinet, and then extracted three points with the following:

$$\begin{aligned} \text{rightPoint} &= (\text{cornerPoint(x)} + 40,\ \text{cornerPoint(y)} - 20,\ \text{cornerPoint(z)} + 30) \\ \text{leftPoint} &= (\text{cornerPoint(x)} - 40,\ \text{cornerPoint(y)} - 20,\ \text{cornerPoint(z)} + 30) \\ \text{topPoint} &= (\text{cornerPoint(x)},\ \text{cornerPoint(y)} + 20,\ \text{cornerPoint(z)} + 20) \end{aligned}$$

Here the rightPoint is our approximation for the initial patch for the right side of the cabinet, leftPoint for the left side, and topPoint for the top half (see Figure 2a).

By using these approximation points we found three points in the point cloud that were closest to the approximations, and each of them respectively turned out to be on the planes we wanted to extract. This meant that we could use them as out initial patches for region growing on each plane and not have to do an exhaustive search for points to serve as initial patches. The patches we acquired using these points were computed using a distance tolerance of 5



(a) (b)

Figure 2: Reduced point cloud data with the corner point (red) and the three approximation points for the three planes (a) and region growing with initialization on the closest patch to the points (b).

2.2.2 Region Growing

In our region growing algorithm we used a plane tolerance of 1 and point distance tolerance of 150. By having a low plane tolerance we acquired more accurate regions and with such a large point distance tolerance the growing only needed about 5 iterations. When the number of new points in a region exceeded the number of points already in the region plus 50 then a plane is (re)fitted to the region. The region growing stops when the least square fitting error of that plane is bigger than 30% of the number of new points.

2.3 Book Extraction

To extract the set of points that represented the book we used couple of thresholds in order to capture the book and reduce noise. First, we used the point cloud that can be seen in Figure 2a and subtracted the points belonging to the three planes. Now to eliminate other data points, such as outside the cabinate (see Figure 2b), we subtracted all points that were to the left and right of the topmost plane, and ones that were below the corner point. Finally, to eliminate noise on the topmost plane, we extracted only the points that had the distance ≥ 1 from the top plane. The result can be seen in Figure 3.

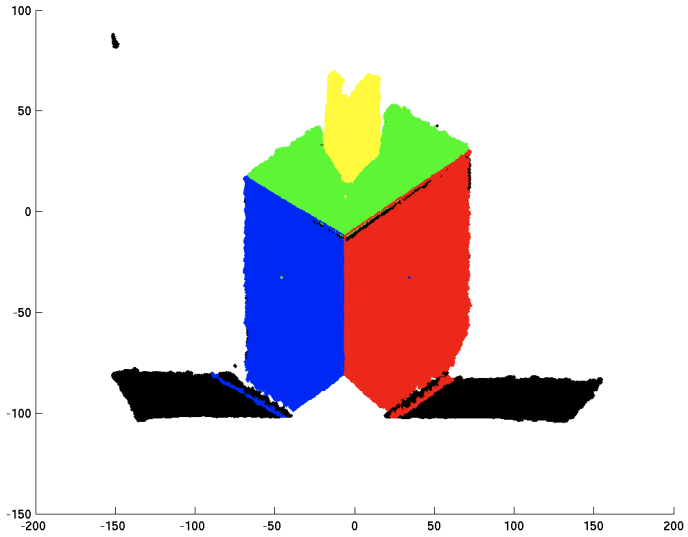


Figure 3: Book points extracted and displayed in yellow

2.4 Plane Registration

2.5 Book Fusion

3 Evaluation

We computed the angles between the surface normals of the upwards facing plane of our foundation frame and the others. The average angle between the vectors was 0.0756 and the standard deviation was 0.0599.

4 Code

4.1 run.m

```
% Run file for Assignment 3
global model planelist planenorm facelines
```

```
load('../../Data/data1.mat')
planeEq = zeros(3,4,21);
closestPoints = zeros(21,3);
bookPoints = {};

fig = figure(1);
for frameNo = 1:length(frame)
    clf;
    hold on
    [pointCloud planeEq planePoints closestPoints(frameNo,:)] = ...
        planeExtraction(frame(frameNo));
    bookPoints{frameNo} = bookExtraction(pointCloud,planeEq, ...
        planePoints,closestPoints(frameNo,:));
end;

% Computing surface normal angle mean and standard deviation
[angle rotationMatrices] = findRotationMatrices(planes);
angles = [angle(1:16) angle(18:21)];
angleMean = sum(angles) / length(angles);
squaredMeanDiff = (angles-mean).^2;
stdDevAngle = sqrt(sum(squaredMeanDiff)/length(angles));
```

4.2 Plane Extraction

4.2.1 planeExtraction.m

```
function [pointCloud planeEq planePoints closestPoints] = ...
    planeExtraction(frame)

planePoints = {};       % Points on each individual plane
bookPoints = {};        % Points on the book
closestPoints = zeros(3); % Closest point in each image, i.e. the corner

xyzFrame = frame.XYZ(:,:,:);
xFrame = xyzFrame(:,:,1);
xFrame = xFrame(:);
yFrame = xyzFrame(:,:,2);
yFrame = yFrame(:);
zFrame = xyzFrame(:,:,3);
zFrame = zFrame(:);
vectorizedFrame = [xFrame yFrame zFrame];
pointCloud = vectorizedFrame(randperm(size(vectorizedFrame,1)),:);

depthThreshold = 1400;
backgroundRowsToIgnore = find(pointCloud(:,3)>depthThreshold);
zeroRowsToIgnore = find(abs(pointCloud(:,1)) + ...
    abs(pointCloud(:,2)) + abs(pointCloud(:,3))==0);

index = true(1,size(pointCloud,1));
index(backgroundRowsToIgnore') = false;
index(zeroRowsToIgnore') = false;
pointCloud = pointCloud(index,:)./5;
```

```
plot3(pointCloud(:,1),pointCloud(:,2),pointCloud(:,3),'k.')

closestPoints = getClosestPoint(pointCloud);
plot3(closestPoints(1),closestPoints(2),3000,'r.')

[NPts,W] = size(pointCloud);
planeEq = zeros(3,4);

remaining = pointCloud;

rightPoint = [closestPoints(1)+40 ...
    closestPoints(2)-20 closestPoints(3)+30];
leftPoint = [closestPoints(1)-40 ...
    closestPoints(2)-20 closestPoints(3)+30];
topPoint = [closestPoints(1) ...
    closestPoints(2)+20 closestPoints(3)+20];

patchPoints = [rightPoint; leftPoint; topPoint]
plot3(rightPoint(:,1),rightPoint(:,2),3000,'b.')
plot3(leftPoint(:,1),leftPoint(:,2),3000,'g.')
plot3(topPoint(:,1),topPoint(:,2),3000,'y.')

for i = 1 : 3
    planePatch = getClosestPointToPoint(pointCloud,patchPoints(i,:))

    % select a random small surface patch from the remaining points
    [oldlist,plane] = select_patches(remaining,planePatch);

    plot3(oldlist(:,1),oldlist(:,2),oldlist(:,3),'y.')

    % grow patch

    growthCycles=0;

    stillgrowing = 1;

    while stillgrowing

        growthCycles = growthCycles+1

        stillgrowing = 0; %-- until we find a bad fit we keep growing

        [newlist,remaining] = ...
            getallpoints(plane,oldlist,remaining,NPts);

        [NewL,W] = size(newlist);
        [OldL,W] = size(oldlist);

        if i == 1
            plot3(newlist(:,1),newlist(:,2),newlist(:,3),'r.')
            save1=newlist;
        elseif i==2
```

```
            plot3(newlist(:,1),newlist(:,2),newlist(:,3),'b.')
            save2=newlist;
        else
            plot3(newlist(:,1),newlist(:,2),newlist(:,3),'g.')
            save3=newlist;
        end
        pause(0.01)

        if NewL > OldL + 50
            ['refitting plane']

            % refit plane
            [newplane,fit] = fitplane(newlist);
            planeEq(i,:) = newplane';
            fitThreshold = 0.3;

            if fit > fitThreshold*NewL    % bad fit - stop growing
                break
            end

            stillgrowing = 1;
            oldlist = newlist;
            plane = newplane;
        end
    end

    ['**************** Segmentation Completed']

    planePoints{i} = oldlist;
end
```

4.2.2 getClosestPoint.m

```
% gets the closest point in the image
function meanClosestPoint = getClosestPoint(pointCloud)
minDepth = min(pointCloud(:,3))
minDepthIndex = find(pointCloud(:,3)==minDepth)
closestPoints = pointCloud(minDepthIndex,:)
meanClosestPoint = mean(closestPoints)
```

4.2.3 getClosestPointToPoint.m

```
% gets the closest point in the point cloud to the point given as input
function closestPoint = getClosestPointToPoint(pointCloud, point)
distancesFromPoint = [];
l = length(pointCloud(:,1));
dist = abs(pointCloud(:,1)-point(1)) + abs(pointCloud(:,2)-point(2)) ...
    + abs(pointCloud(:,3)-point(3));
closestIndex = find(dist==min(dist));
closestPoint = pointCloud(closestIndex(1),:);
```

4.2.4 Other

Here we also used the following code from the course:

- *select_patches.m* - Slightly modified version of *select_patch.m* where the exhaustive search for points is removed and instead only the three approximated plane patches are used.
- *getallpoints.m* - Tolerance variables modified.
- *fitplane.m* - Not modified.

4.3 Book Extraction

4.4 bookExtraction.m

```
function bookPoints = ...
        bookExtraction(pointCloud,planeEq,planePoints,closestPoints)

% subtract points in R that are on planes
allPoints = [planePoints{1};planePoints{2};planePoints{3}];
[~,planeRowsToIgnore] = ismember(allPoints,pointCloud,'rows');
planeRowsToIgnore(find(planeRowsToIgnore > 0));

planeIndex = true(1,size(pointCloud,1));
planeIndex(planeRowsToIgnore') = false;
reducedPointCloud = pointCloud(planeIndex,:,:);

% subtract points too low / to left / to right of planes
minXPoint = min(planePoints{3}(:,1));
maxXPoint = max(planePoints{3}(:,1));
bookPoints = reducedPointCloud(find( ...
    (reducedPointCloud(:,1) > minXPoint) & ...
    (reducedPointCloud(:,1) < maxXPoint) & ...
    (reducedPointCloud(:,2) > closestPoints(2))),:);

% Get points above top plane
[abovePoints aboveIndices] = ...
    getPointsAbovePlane(bookPoints,planeEq(3,:));
bookPoints = bookPoints(aboveIndices,:);
plot3(bookPoints(:,1), bookPoints(:,2),bookPoints(:,3),'y.');
pause(0.01);
% saveas(fig,strcat('../../Images/BookExtraction3/',int2str(frameNo)),'png');
```

4.5 getPointsAbovePlane.m

```
function [points indices] = getPointsAbovePlane(R,plane)
R = [R ones(size(R,1),1)];
distanceFromPlane = R*plane';
distanceTolerance = -1;
indices = find(distanceFromPlane <= distanceTolerance);
points = R(indices);
```

4.6 Plane Registration

4.7 Book Fusion