Lab 4

Hanz Cuevas Velásquez, Bob Fisher
Advanced Vision
School of Informatics, University of Edinburgh

Week 4, 2018

Today's lab will teach you how to read and write point-cloud data. a point-cloud data is a set of points in the tridemensional coordinate system, x, y, and z and represent the surface of an object; it can also encode the color information R,G, and B of the object.

1 Reading and writing point-cloud data

The two most used formats of a point-cloud file are Polygon file format (`.ply`) and Point Cloud Data (`.pcd`). We will start the lab by reading and showing the point-cloud file `bottles.ply`[1].

```
pc = pcread('Images/bottles.ply'); %Reading the point-cloud
showPointCloud(pc) %Showing the point-cloud
% For version of matlab > R2015a the command can be replaced by pcshow(pc)
```

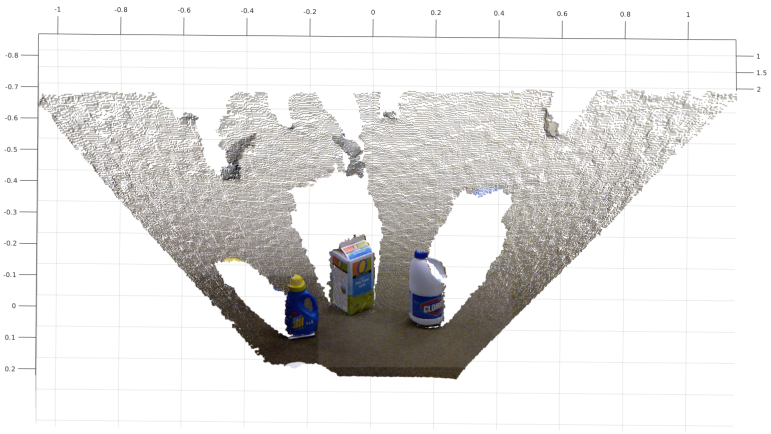

Figure 1: Plot of the point-cloud data.

We will observe an image similar to figure 1. We can rotate the point-cloud using the rotate 3D button. Each point in the plot has an x, y, and z which indicates where in the space is that point. Each point also has 3 other values assigned to it, the RGB dimensions. To describe this relation more in depth, first we will observe what is the data stored in the point-cloud, for that we will write `pc` in the command window.

```
%Write pc without semiclon
pc
```

[1] Image taken from the Point Cloud Library dataset: https://github.com/PointCloudLibrary/pcl/tree/master/test

```
%Output
pointCloud with properties:

 Location: [307200x3 single]
    Color: [307200x3 uint8]
   Normal: []
Intensity: []
    Count: 307200
  XLimits: [-1.0608 1.1525]
  YLimits: [-0.8692 0.2197]
  ZLimits: [0.5010 2.0630]
```

Here we can observe two important properties, the location and color. `Location` stores the x, y, and z coordinates of each point in the point-cloud and `Color` stores the color (r,g, and b) assigned to those points, E.g. a point p with coordinates $(1, 1, 1)$ may have the following color values $(255, 23, 52)$. Both properties encode the data in a list of size $[n \times 3]$ where n is the number of points our data has, and 3 represents the x, y, and z coordinates for `Location` or r, g, and b for `Color`. We can access to the content of these properties using the following code `pc.Color` and `pc.Location`. We can also observe that our point-cloud has 307200 points. Usually, when we collect some 3D data with color, we know the resolution or size of the image; these points are the flat version of the total number of pixels in the image. In our case, it has a size of $480 \times 640 = 307200$.

Your task is to reconstruct the color image using the color property in the pointcloud. For that, you have to do the following:

- Get the r, g, and b color from the list `pc.Color`.
- Reshape the 3 lists into a matrix of size $[480 \times 640]$. Use the command `reshape(list, [row, col])` where `list` is the variable you want to reshape, and `row` and `col` represent the dimensions of the new shape.

```
color_pc = pc.Color;
r = %(Code: get the r color)
g = %(Code: get the g color)
b = %(Code: get the b color)
rec_r = %(reshape r)
rec_g = %(reshape g)
rec_b = %(reshape b)
new_rgb(:,:,1) = rec_r;
new_rgb(:,:,2) = rec_g;
new_rgb(:,:,3) = rec_b;
imshow(new_rgb)
```

Now that we have our image reconstructed, we want to find in which position (location) is the third bottle from left in figure 1, and segment it in the point-cloud. The first step is to segment the bottle in the Colour space (2D space or color matrix) and find a mask. After we segmented the bottle, we have to relate the pixels of the bottle in our 2D image to the points in our point-cloud. We can reshape the mask and use it on the `pc.Color` list. We can also use the different notations Matlab has to index a matrix to find the relationship between the 2D space and the point-cloud; we will implement the second. Matlab, apart from allowing us to access to a matrix using the rows and columns as indices, it lets us access to the values of the matrix using linear indices. These linear indices are numbers from 1 to the total number of pixels, starting from the top left of the matrix, going down to the end of the first column and so on until it reaches the bottom right of the matrix. See figure 2 for a better description of this relation.

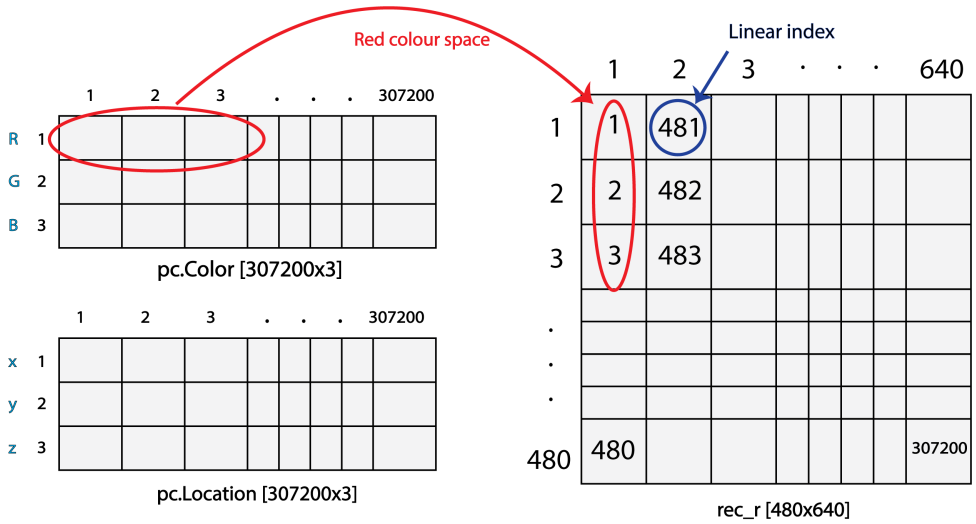

Figure 2: Showing the relation between the color and location matrix index and the linear index of a matrix of an image. On the left, the color and location matrix of the pointcloud, on the right the red color matrix we reshaped.

For this part of the lab, we will use the `find` command to find the linear indices of our matrix that have the pixels of our bottle. These linear indices are the same indices of our flat `Color` and `Location` matrices, as seen in the previous figure. In other words, the indices we get with this command are also the indices we can use in the `pc.Location` variable. In this exercise we will also learn how to create and save a new point-cloud data with `pointCloud` and `pcwrite`. Your task is the following:

- Segment and show the third bottle from figure 1 (white bottle with blue bottle cap).
- use `find` to find the pixels that do not belong to the bottle. E.g `find(x<=t)` finds all the linear indices of the matrix `x` that are less or equal to a threshold `t`. HINT: Use the mask you found to segment the bottle.

```
%(Code: Segment the third bottle and name the mask "threshold")

indx_xyz_no=%(Code: find the indices of the pixels where our bottle is not located)

xyz_pc = pc.Location;

%Eliminating the indices where our bottle is not located.
xyz_pc(indx_xyz_no,:)=[];
color_pc(indx_xyz_no,:)=[];

%Creating a new point-cloud
new_pc = pointCloud(xyz_pc, 'Color', color_pc);

%Saving the new point-cloud
pcwrite(new_pc,'single_bottle.ply');
showPointCloud(new_pc)
```

We will have the following result (figure 3c):

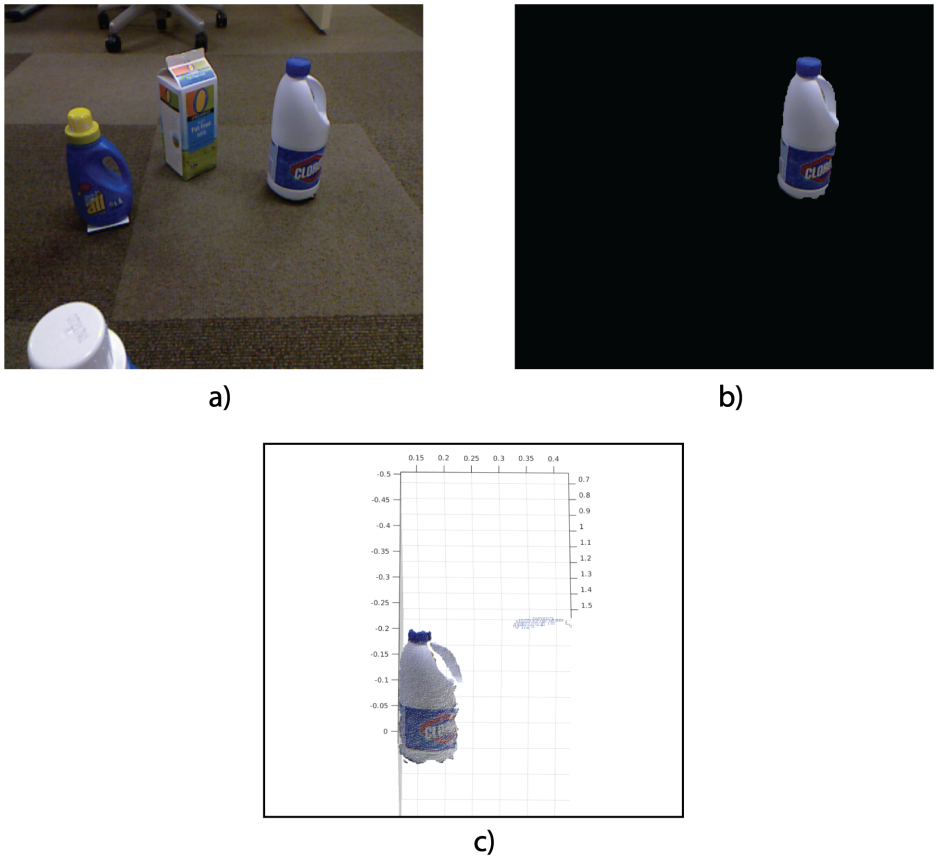

Figure 3: a) 2D image, b) Segmented bottle, c) Point-cloud plot of the segmented bottle.

In figure 3c, we can observe that we do not have a complete image of the bottle. This is because the depth sensor was not capable of detecting the depth values of those regions. If we go to the **Workspace** window, on the right of the main window, and double click on the variable `xyz_pc`, we can observe that there are several `NaN`, values. These values indicate that there is no depth information in those regions, as seen in 4. The student should think about the reasons why the sensor did not get the depth information in some parts of the image.

xyz_pc

13209x3 single

	1	2	3
1	0.1320	-0.0235	0.8500
2	0.1320	-0.0219	0.8500
3	0.1320	-0.0202	0.8500
4	0.1214	-0.0171	0.7820
5	0.1214	-0.0156	0.7820
6	NaN	NaN	NaN
7	NaN	NaN	NaN
8	0.1336	-0.0235	0.8500
9	0.1333	-0.0218	0.8480

Figure 4: Values inside `xyz_pc` variable.