1. We did this step by loading the “01” folder into the working directory of the colab notebook, and then editing the original pictures to include the overlaid rectangles. There was some confusion as to how these rectangle coordinates were defined in the text files, so the team decided to use ordered pairs of rectangles, with each ordered pair describing the top left and bottom right coordinates of the rectangles. So for example, if in a text file for the positive rectangles, a list of (300, 350) and (200, 250) was given, a positive rectangle would be created, whose top left corner is at (300,250) and bottom right corner is at (200,250). The team could not find any documentation to describe whether this is the correct interpretation or not, but if it is not the correct interpretation, it would be very easy to slightly edit our code to fix the problem.
2. This step was done by loading the linked github repo into our project. The team was not sure what an RGB-D image was, so we looked it up. We found the following image:



Using the github repo given, we created code that created two similar pictures, one picture representing the RGB channels of the image, the other picture representing the depth (D) channel. Some odd holes can be seen in the D pictures generated, and we believe this is caused by improper data capture in the point cloud files. Further support can be found for this in problem 5, discussed later.

1. Problem 3 was simple enough. We started by saving the YUV and depth images of each picture taken. After this, we cropped these images to the sizes of the rectangles given in the data, again following the same convention stated in Problem 1. After cropping the YUV images, it was simple to display them. However, for the depth images, we had to normalize them to properly display the depth information after cropping them. These normalized images can be seen in the colab notebook.
2. (Someone else do this one)
3. For this problem, we used open3d, an open-source python library, to convert the text files to .ply files. After doing this, we saved the .ply files locally to our computers, and again displayed them using open3d. Dr. Sarkar said it was okay to not use ROS for this problem, as we did not have time to set up a virtual Ubuntu machine, which is required for ROS. A few generated point clouds can be seen below. In the second point cloud shown, it can be seen that the data captured is not of great quality, having many holes around the stapler scanned. These holes are the reason for the odd holes in the problem 2 Depth images generated.



