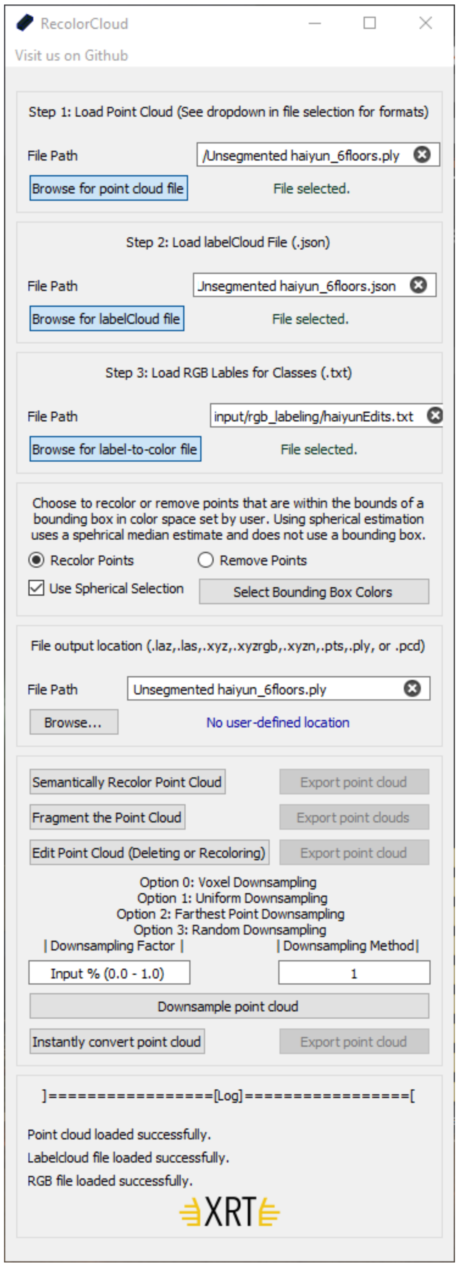
RECOLORCLOUD: AN OPEN SOURCE EDITOR FOR SEGMENTING, RECOLORING, AND CONVERTING LARGE POINT CLOUDS

Primary Developer: **Esteban Segarra Martinez**

Supporting Project: **[**LINK**]**

Grant Number: **NSF: 2120240**


RecolorCloud
Visit us on Github
Browse for point cloud file
Browse for labelCloud file
Browse for label-to-color file
Select Bounding Box Colors
Browse...
Semantically Recolor Point Cloud
Export point cloud
Fragment the Point Cloud
Export point clouds
Edit Point Cloud (Deleting or Recoloring)
Export point cloud
Downsample point cloud
Instantly convert point cloud
Export point cloud
XRT

Step 1: Load Point Cloud (See dropdown in file selection for formats)

File Path: /Unsegmented haiyun_6floors.ply

File selected.

Step 2: Load labelCloud File (.json)

File Path: Jnsegmented haiyun_6floors.json

File selected.

Step 3: Load RGB Lables for Classes (.txt)

File Path: input/rgb_labeling/haiyunEdits.txt

File selected.

Choose to recolor or remove points that are within the bounds of a bounding box in color space set by user. Using spherical estimation uses a spehrical median estimate and does not use a bounding box.

● Recolor Points ○ Remove Points

☑ Use Spherical Selection

File output location (.laz,.las,.xyz,.xyzrgb,.xyzn,.pts,.ply, or .pcd)

File Path: Unsegmented haiyun_6floors.ply

No user-defined location

Option 0: Voxel Downsampling
Option 1: Uniform Downsampling
Option 2: Farthest Point Downsampling
Option 3: Random Downsampling

| Downsampling Factor | : Input % (0.0 - 1.0)

| Downsampling Method| : 1

```
]================[Log]================[

Point cloud loaded successfully.
Labelcloud file loaded successfully.
RGB file loaded successfully.
```

CONTENTS

INTRODUCTION

RecolorCloud is a point cloud recoloring tool used for re-coloring, semantically recoloring, segmentation, and conversion of point clouds. RecolorCloud works by utilizing algorithms written specifically for recoloring points based on simple recoloring techniques. These algorithms allow the tool to recolor and semantically recolor based on selected options.

The tool was created to be a simple tool that works alongside a third-party tool called LabelCloud to create bounding boxes on a point cloud. This tool combines the output from LabelCloud, a point cloud, and a coloration file to produce a final output point cloud.

FEATURES AND FUNCTIONALITY

- Recoloring to correct outlier points inside a bounding box
- Solid recoloring a point cloud to semantically recolor the points inside a bounding box
- Fragment a point cloud as defined by a LabelCloud bounding box
- Convert a point cloud into one of the supported file types
- Handle large scale point clouds (>600 million points)
- Functional UI for python interfacing

QUICKSTART AND INSTALLATION

1. Download the repository at [LINK]
2. Install or Launch Conda and create a new environment using the requirements.txt file provided.
3. Unpack ReColorCloud source code.
4. Launch your Conda environment.
    a. You can open the project in VS code or on a Conda console window.
    b. > `python main.py`

LABELCLOUD USAGE INSTRUCTIONS

Users should be able to load a point cloud into label cloud by following labelCloud's instructions on the repository. Users should only need to use the following command to install labelCloud:

- ➢ `pip install labelcloud`
- ➢ `Labelcloud` or `python labelcloud`

Labelcloud is limited to in its performance capabilities for visualizing large point clouds so it is recommended to *downsample* a point cloud using RecolorCloud (Preferably to one with a file size less than 1GB from our trials) and load it into LabelCloud.

EXTENDED USER GUIDE AND HOW-TO

This section will discuss the inner workings of RecolorCloud, how does it work, and how to best use it towards a project.

LOADING DATA

To load data, follow Steps 1 – 3 on the UI. They are outlined as follows:

- Step 1: Load a point cloud
    - A necessary operation for editing the point cloud. Currently RecolorCloud supports .las, .laz, .pts, .ply, .pcd, .xyz, .xyzn, and .xyzrgb. **Please be patient as point clouds may take a few minutes to load, especially large ones**. This is just the nature of Open3D and python at the moment.
- Step 2: Load a lableCloud bounding box
    - Required for editing the colors of the point cloud, fragmentation or deleting outlier colors
- Step 3: Load the text-to-label file
    - Required for changing the color of the point clouds semantically or "selecting" which bounds to edit.

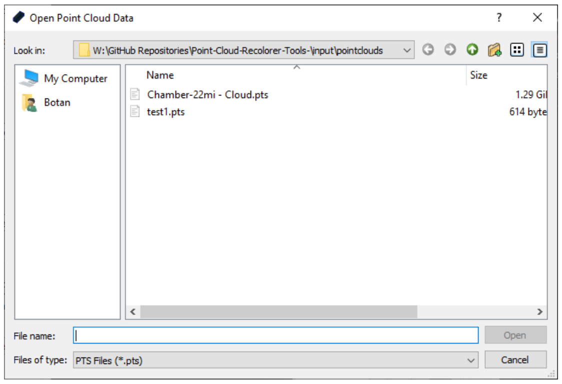


Figure 1:File selection window

Note that Step 2 and 3 and required for recoloring, deletion, and fragmentation. Downscaling and file conversion are still possible without these files.

SAVING DATA

Saving data is easily done by selecting a name or a location to store the point cloud. Users can press the "Browse…" button to select a location to directly output the point cloud. Users can navigate to a location and output the point cloud to a location on their hard drive. If no location is provided, the data will be stored locally on RecolorCloud's default output locations which are:

- Output/range_recolored
    - This is for point clouds that have been deleted or recolored
- Output/recolored_by_label
    - This is for point clouds that were sematically recolored
- Output/segmented_by_label
    - This is for point clouds that were fragmented

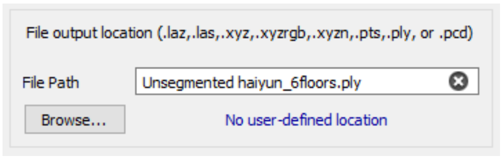

File output location (.laz,.las,.xyz,.xyzrgb,.xyzn,.pts,.ply, or .pcd)
File Path
Unsegmented haiyun_6floors.ply
Browse...
No user-defined location

Figure 2:File output location and browser

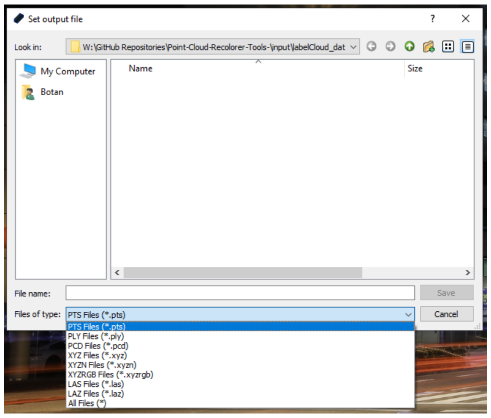


Figure 3:Save browser

DOWNSAMPLING

This section describes the capabilities of RecolorCloud for downsampling or decimating the point cloud. This process follows the convention followed by Open3D's downsampling features. Open3D supports 4 different downscaling processes detailed as follows:

- Voxel Downsampling
  - This utilizes Open3D’s function to downsample a point cloud using voxels, minimizing using small boxes and removing points in this manner.
- Uniform Downsampling
  - This uniformly downsamples the point cloud in an attempt to retain the original shape of the cloud. This is different from random downsampling, which can remove points anywhere in the point cloud
- Farthest Point Downsampling
  - This downsamples based on the furthest distance points .
- Random Downsampling
  - This downsamples the point cloud by randomly removing points until a desired percent amount is achieved.

As seen in figure 1, the downsampling techniques are also described in the UI. The UI is designed to allow the user to input the downscaling factor on the **left text input** and the method to be used on the **right text input**.

- Downscaling factor: A user would input a value between 0.0 – 1.0, representing the total percentage value that should remain after the decimation process.
- Downsampling method: A user inputs between 0 – 3 and selects between any of the previously mentioned techniques.

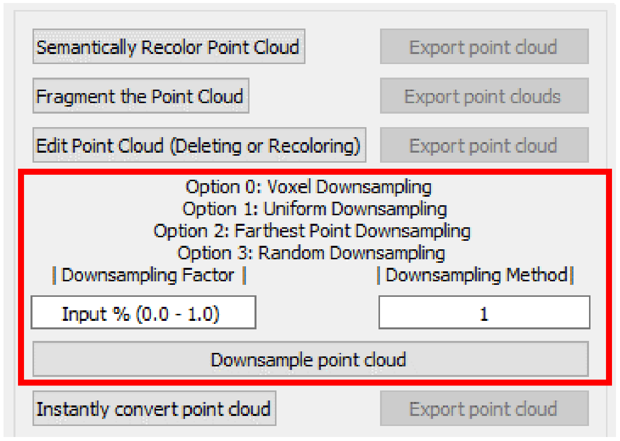

Semantically Recolor Point Cloud
Export point cloud
Fragment the Point Cloud
Export point clouds
Edit Point Cloud (Deleting or Recoloring)
Export point cloud
Option 0: Voxel Downsampling
Option 1: Uniform Downsampling
Option 2: Farthest Point Downsampling
Option 3: Random Downsampling
| Downsampling Factor |
| Downsampling Method|
Input % (0.0 - 1.0)
1
Downsample point cloud
Instantly convert point cloud
Export point cloud

Figure 4: Downsampling module from the RecolorCloud UI

Upon finishing downsampling, the export point cloud button will be enabled at the bottom right of the tool.

SEGMENTATION

Segmentation is not directly done by RecolorCloud directly, this step is done by LabelCloud. This editor exports bounding boxes by importing the point cloud and presenting a UI for placing and editing bounding boxes. As an example, here’s a preview of Greek Park with its full bounding boxes.

The bounding boxes are exported from LabelCloud using the **centroid_rel** format. This means that all centroid rotations are relative and avoids exporting them incorrectly.

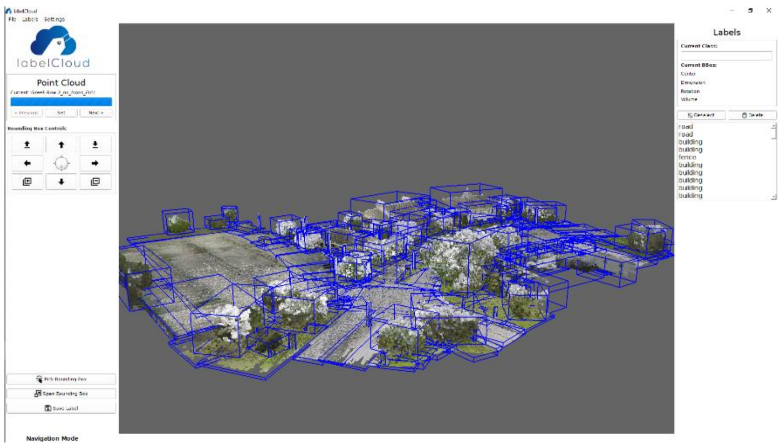


Figure 5: LabelCloud with Bounds shown for Greek Park.

Bounding boxes are imported using **step 2**.

FRAGMENTATION

Fragmentation is done by loading the bounding boxes from the segmentation step and importing them. These are created by creating multiple point clouds from the primary point cloud as defined by the bounding boxes and setting flags that enable or disable them from consideration in the fragmentation algorithm.

As an example, I show an example line from a label-to-color file:

```
car 255 255 255 1
potato 128 128 128 0
label r g b enabled-flag
```

In this example, the flag for car is **enabled** while potato is **disabled**. This means that all bounds labeled as car will be included for export while potato will be ignored. Please keep in mind the name of the label **is case sensitive** and as a **single string with no spaces**. This is because labels could be confused by other similar-named labels, of which the user should attempt best practices while labeling the bounding boxes. Any space in this between strings is interpreted as a new variable. For reference, the last line of the file is left to inform the user of the formatting but is optional.

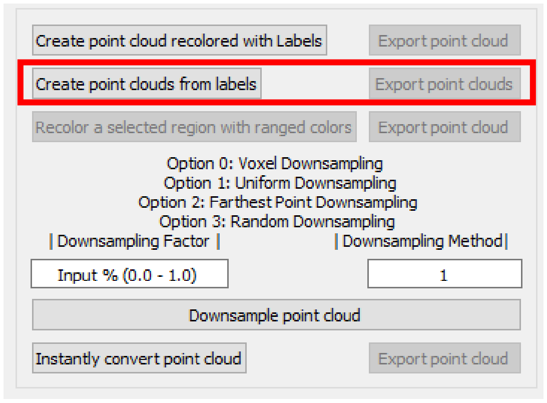

Create point cloud recolored with Labels
Export point cloud
Create point clouds from labels
Export point clouds
Recolor a selected region with ranged colors
Export point cloud
Option 0: Voxel Downsampling
Option 1: Uniform Downsampling
Option 2: Farthest Point Downsampling
Option 3: Random Downsampling
| Downsampling Factor |
| Downsampling Method|
Input % (0.0 - 1.0)
1
Downsample point cloud
Instantly convert point cloud
Export point cloud

Figure 6: The Framentation Button for RecolorCloud (Boxed in Red). The export button is enabled once a point cloud has been successfully processed.

To fragment a point cloud, users would load Steps 1 -3, verify their settings, and press on the **Create point clouds from Lables** button as shown in **Figure 3**. Users should keep in mind that point clouds may be exported to the **local /output/segmented_by_label folder**. There should be multiple point clouds that represent the individual bounds of the point cloud.

DELETION

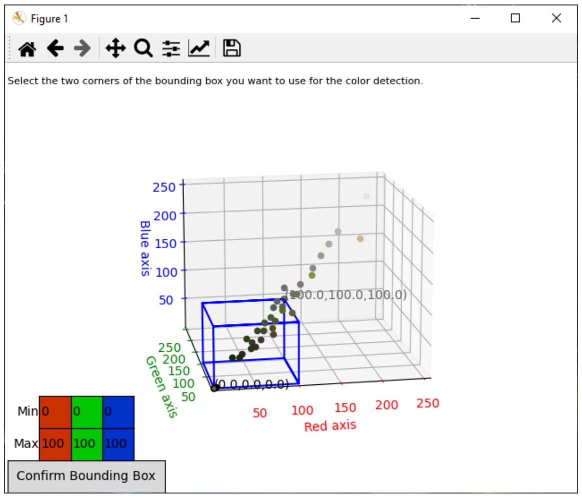
Points on a point cloud are all represented by a position and a RGB color value. Some may be more complex or have different color standards, but the general premise remains the same. Deletion in the context of RecolorCloud is the ability to remove specific points with colors that fall outside RGB color distance in the RGB color space.

Deletion is separated into two different categories:

- RGB Bounding
- Spherical Selection

Both categories work in the RGB color space and operate only on **active bounding boxes** imported in steps 2 -3.

RGB BOUNDING


Select the two corners of the bounding box you want to use for the color detection.

Min	0	0	0
Max	100	100	100

Figure 7: Bounding box selection. This screen shows a simplified median of the colors from active bounding boxes.

In this case, points are removed from the point cloud if the color is outside the RGB bounding box created by the user in the previous step. Points that are inside the box are kept.

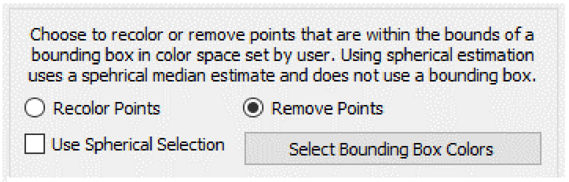

Choose to recolor or remove points that are within the bounds of a bounding box in color space set by user. Using spherical estimation uses a spehrical median estimate and does not use a bounding box.

Figure 8: Deletion with RGB bounding

Users create a bounding box by pressing the "**Select Bounding Box Colors**" button and following the UI to create a bounding box. The bounding box is created by typing the maximum corner bound of the bounding box and the minimum bound of the box. Clicking anywhere and closing the window or pressing the **"confirm bounding box"** button will save the values. Figure 4 shows the setup for this technique.

When using the bounding box selector, please note that the dots placed in the 3D matplotlib window represent a small selection of the unique colors found inside the currently active bounding boxes. Note that since this represents only a small selection of the potential colors in the bounding boxes, some colors may be missed.

SPHERICAL SELECTION

In the RGB space, the median color is found by using Numpy's median tool and used to create a centroid from which a colors within a sphere are selected. Color points that are within the distance of the sphere are kept and any colors outside the sphere are discarded and removed from the point cloud.

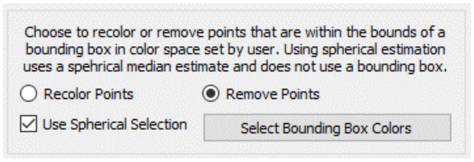
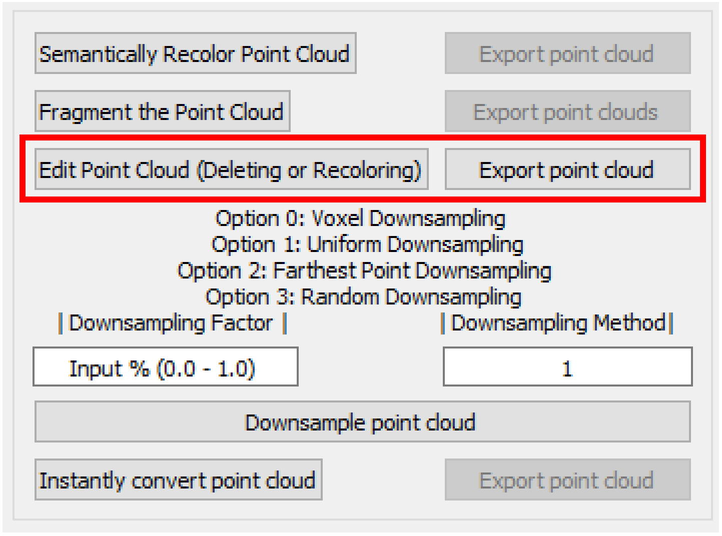

Choose to recolor or remove points that are within the bounds of a bounding box in color space set by user. Using spherical estimation uses a spehrical median estimate and does not use a bounding box.

Figure 9:Deletion with spherical selection.

By pressing the "**Use Spherical Estimate**" checkbox, users can enable the spherical estimate strategy. At present, the radius of the spherical estimate is a color distance of 85.

CONFIRMING DELETION

When you have performed your deletion selections, you will be able to press on the left button that says "**Edit Point Cloud (Deleting or Recoloring)**". When the processing is complete, the right export button will become active. **Please note that deletion and recoloring may take a few minutes to process.**


Semantically Recolor Point Cloud
Export point cloud
Fragment the Point Cloud
Export point clouds
Edit Point Cloud (Deleting or Recoloring)
Export point cloud
Option 0: Voxel Downsampling
Option 1: Uniform Downsampling
Option 2: Farthest Point Downsampling
Option 3: Random Downsampling
| Downsampling Factor |
| Downsampling Method |
Input % (0.0 - 1.0)
1
Downsample point cloud
Instantly convert point cloud
Export point cloud

The way you know your point cloud processing is close to being complete is by looking at the python window, looking at the total amount of bounding boxes to be processed and looking at the presently processed bounding box.

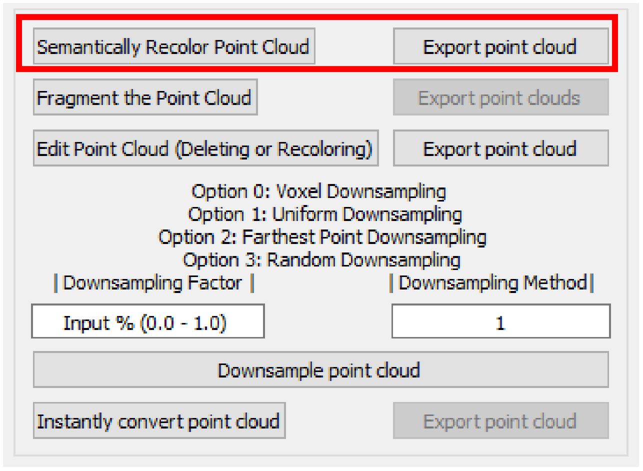
RECOLORING

As with deletion, recoloring requires bounding boxes and a label-to-color file from steps 2 -3. There are a couple of recoloring strategies as described below. In this technique, points that fall outside the criteria are not removed but are recolored or retained.

SEMANTIC SEGMENTATION

This is the simplest algorithm, where objects are recolored based on the colors they are given through the labels and the lable-to-color file. This simply converts the points inside the bounding boxes into solidly colored points according to the label-to-color file.

This does not depend on spherical or RGB bounding boxes since all points are going to be uniformly colored. As in deletion, pressing the "**Semantically Recolor Point Cloud**" button will start the recoloring process. Once it is done, the export button will be unblocked.



Option 0: Voxel Downsampling
Option 1: Uniform Downsampling
Option 2: Farthest Point Downsampling
Option 3: Random Downsampling

Downsampling Factor		Downsampling Method
Input % (0.0 - 1.0)	1

Figure 10: Segmentation recoloring settings

Note that this process as-is will create two preview windows, one with preview bounding boxes and a second one where the point cloud has been recolored. **The preview windows will temporarily pause the execution of the processing so you must close the preview windows to move onto the next steps.** Figure 8 shows these preview windows in action.

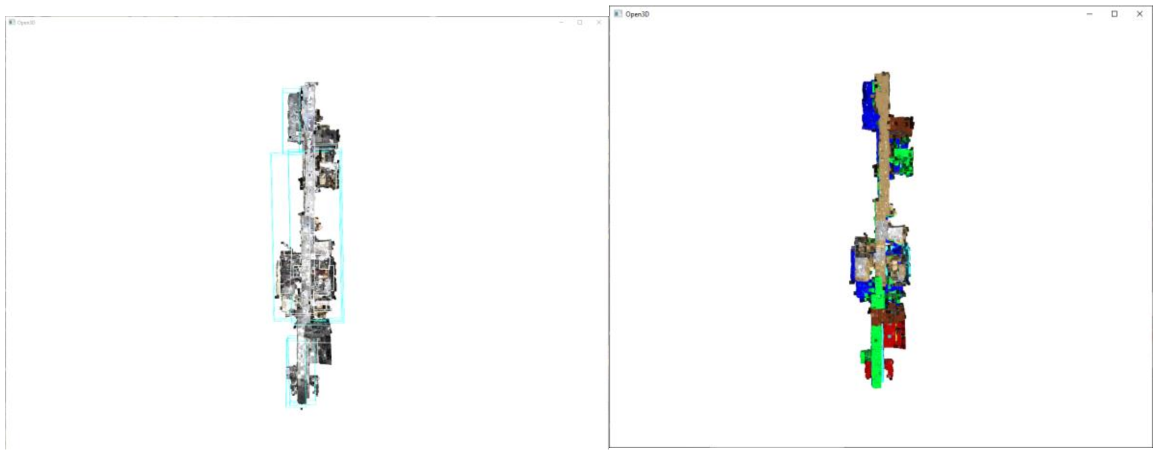


Figure 11:Preview windows

RGB BOUNDING

This process is similar to that of deletion process where a bounding box is created for the RGB color space of the point clouds in bounding boxes**. An automatic bounding box is created for all possible colors from the active bounding boxes of the point cloud**.

When a user creates a bounding box in the RGB window, they are creating a new bounding box with a new centroid. The recoloring algorithm then recolors the points from the original centroid to match the user's centroid. This means that the colors that are seen on the window are then scaled or moved relative to the target user centroid.

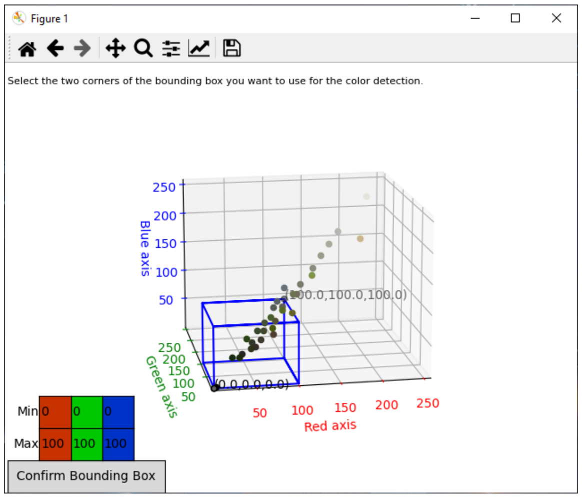
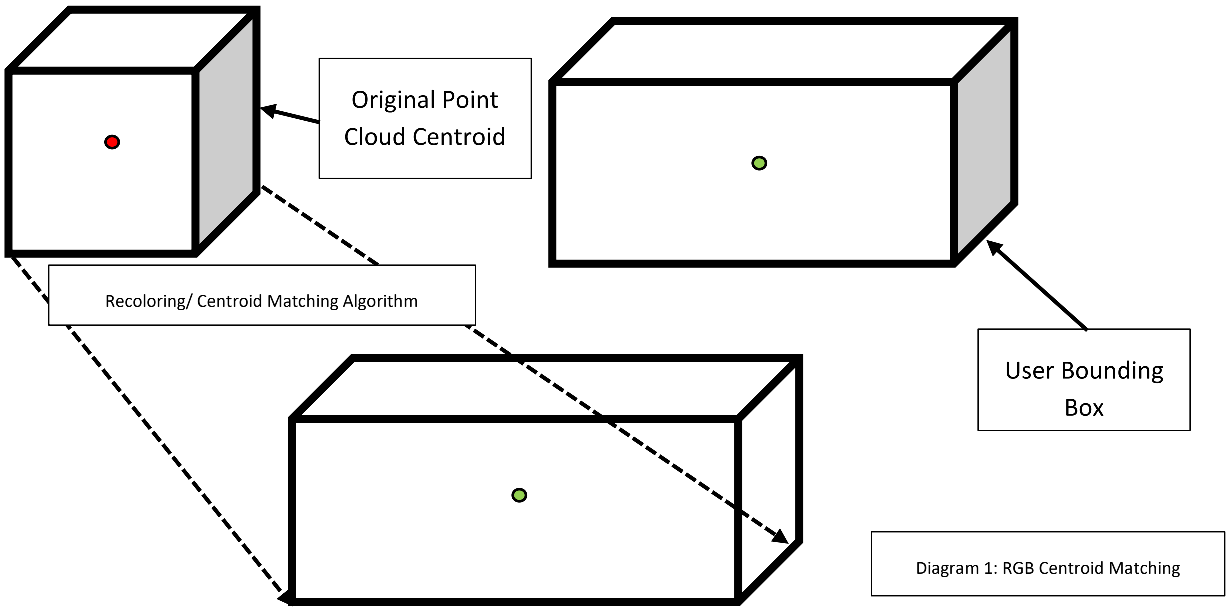


Figure 1

Select the two corners of the bounding box you want to use for the color detection.

Min	0	0	0
Max	100	100	100

Figure 12: Bounding box window similar to that of the deletion menu.

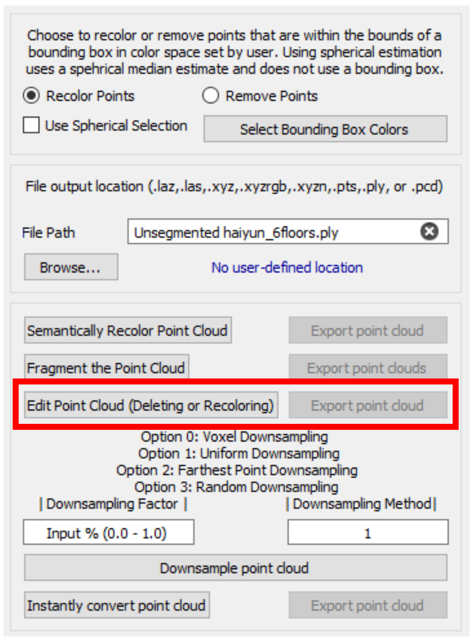


Original Point Cloud Centroid
Recoloring/ Centroid Matching Algorithm
User Bounding Box

Diagram 1: RGB Centroid Matching

As show in diagram 1, the automatically created bounding box in the RGB space is scaled to match the user's bounding box. After processing the scaling and changes, the colors inside the automatic bounding box are moved to their relative locations where they will be in the user's bounding box. This means that colors can be hue shifted or moved between different colors without removing them and automatically recoloring them. They will also retain their relative shading as when they were registered.

The process is started when the user pressed the "**Edit Point Cloud (Deleting or Recoloring)**". Similar to segmentation, a preview window will be displayed to show where the bounding boxes are displayed on the as well as a post-recoloring preview window. After the recoloring process is completed, the "Export Point Cloud" window will be enabled. Figure 10 shows where the buttons are on the user interface.



Choose to recolor or remove points that are within the bounds of a bounding box in color space set by user. Using spherical estimation uses a spehrical median estimate and does not use a bounding box.

● Recolor Points ○ Remove Points

☐ Use Spherical Selection

Select Bounding Box Colors
File output location (.laz,.las,.xyz,.xyzrgb,.xyzn,.pts,.ply, or .pcd)
File Path
Unsegmented haiyun_6floors.ply
Browse...
No user-defined location
Semantically Recolor Point Cloud
Export point cloud
Fragment the Point Cloud
Export point clouds
Edit Point Cloud (Deleting or Recoloring)
Export point cloud
Option 0: Voxel Downsampling
Option 1: Uniform Downsampling
Option 2: Farthest Point Downsampling
Option 3: Random Downsampling
| Downsampling Factor |
| Downsampling Method|
Input % (0.0 - 1.0)
1
Downsample point cloud
Instantly convert point cloud
Export point cloud

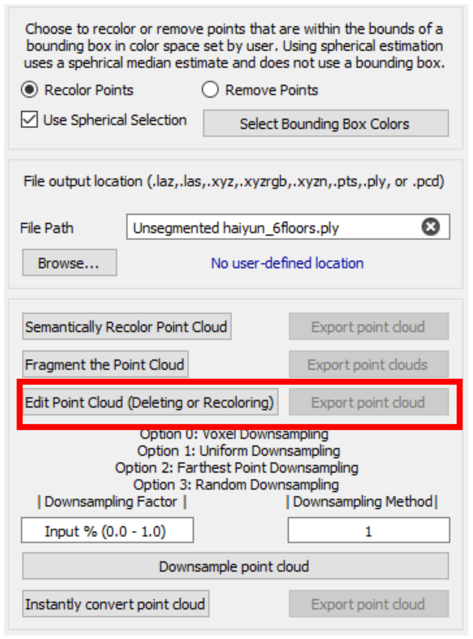
Figure 13: RGB Bounding box recoloring is similar to sphereical recoloring however the "Use Spherical estimate" checkbox is unticked.

SPHERICAL RECOLORING

Spherical Recoloring is similar to the spherical selection technique, where a sphere is generated on the centroid of all unique colors per each point cloud selection per bounding box. All colors within the range of the sphere's centroid are unchanged, however all outliers to the inside of the sphere are recolored using random colors that are inside the sphere. This means that any colors that are effectively outside the median of the point cloud will be recolored.

Due to the nature of this algorithm, the more points of similar or closely related colors are present, the more likely it is that the centroid will be closer to the majority of that color. Therefore, if a tree has more white points than green points it will tend to repaint the entire tree white as it considers the white points the greater of the two. The way to fix this is to follow these steps:

- Import unmodified point cloud
- Remove most white points using RGB bounding box deletion
- Export/save cleared point cloud
- Import cleared point cloud
- Recolor using spherical selection
- Export final point cloud



Choose to recolor or remove points that are within the bounds of a bounding box in color space set by user. Using spherical estimation uses a spehrical median estimate and does not use a bounding box.

● Recolor Points ○ Remove Points

☑ Use Spherical Selection

Select Bounding Box Colors
File output location (.laz,.las,.xyz,.xyzrgb,.xyzn,.pts,.ply, or .pcd)
File Path
Unsegmented haiyun_6floors.ply
Browse...
No user-defined location
Semantically Recolor Point Cloud
Export point cloud
Fragment the Point Cloud
Export point clouds
Edit Point Cloud (Deleting or Recoloring)
Export point cloud
Option 0: Voxel Downsampling
Option 1: Uniform Downsampling
Option 2: Farthest Point Downsampling
Option 3: Random Downsampling
| Downsampling Factor |
| Downsampling Method|
Input % (0.0 - 1.0)
1
Downsample point cloud
Instantly convert point cloud
Export point cloud

Figure 14: Recoloring using sphereical selection

The process is started when the user pressed the "**Edit Point Cloud (Deleting or Recoloring)**". Similar to segmentation, a preview window will be displayed to show where the bounding boxes are displayed on the as well as a post-recoloring preview window. After the recoloring process is completed, the "Export Point Cloud" window will be enabled. Figure 11 shows where the buttons are on the user interface.

LARGE POINT CLOUD SUPPORT

By the nature of python and Open3D, point clouds are loaded into the application through a local python script. These are loaded into RAM through Open3D and is available until the python script is closed or crashes. The amount of RAM used depends on the size of the point cloud to be loaded. For example, it might be expected that a point cloud of approximately 600 million points of around 70GB might take up approximately 70GB of RAM. Similarly, a smaller point cloud will take less than this amount. Unfortunately, **Open3D requires to load the point cloud data locally**, the amount of data required to perform operations can be expected to be approximately the file size of the target point cloud. Loading large point clouds **is slow**, however with patience, they will eventually load into RecolorCloud and will perform operations.

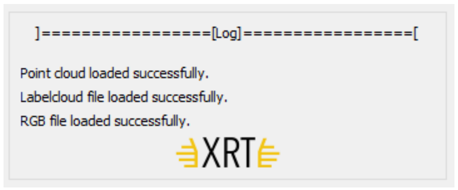
CONVERSION

Open3D supports six file types to be loaded and processed natively. Laspy then supports the .las and .laz file formats when loading those file types. Sometimes there are errors when loading certain files due to incomplete support of Open3D and Laspy at certain points. In the future, we plan to expand and improve the support for these and future point cloud file types.

OTHER FEATURES

LOGGING

RecolorCloud has a logger that is shown on the UI to show what is the current status (such as point cloud loading faliures). Please keep track of it in case of errors and also the python console in case errors are encountered during runtime.



```
]=================[Log]=================[

Point cloud loaded successfully.
Labelcloud file loaded successfully.
RGB file loaded successfully.
```

Figure 15:Logger

LIMITATIONS

A NOTE ON RECAP CONVERTED FILES (.PTS)

Files processed by Autodesk Recap can be exported into .pts files, however they are unreadable by Open3D due to how Recap stores them as color is stored in a different manner and includes three additional columns for alpha or normal calculations. These last two cause the issue where open3D cannot recognize the file despite being technically compatible. A future update will attempt to remedy this problem. In the meanwhile, the workaround is to export the point cloud as a .e57 into Cloud Compare and export from there into either a .ply, .las, or .pts file. Sometimes Cloud Compare will fail to export as a .pts so be wary of this issue.

The suggested process for a Recap project is as follows:

- Load the project on Recap
- Under export, export the project as a .e57 point cloud
- Open the project in an editor such as Cloud Compare
- Export the project as a .ply, .pts, or .pcd
- Open the point cloud in RecolorCloud

We plan to fix these issues in a future version that can import .e57 files directly into RecolorCloud and skipping the Cloud Compare requirement.

CLOSING NOTES

Thank you for the opportunity to use our application. The following are involved with this project:

- Esteban Segarra Martinez (Primary Developer)
- Mykola Maslych (Paper editor and writer)
- Robert Michlowitz (Provided Greek Park Dataset)
- Lori Walters (Provided Greek Park Dataset)
- Ryan McMahan (Project guidance, paper editor)