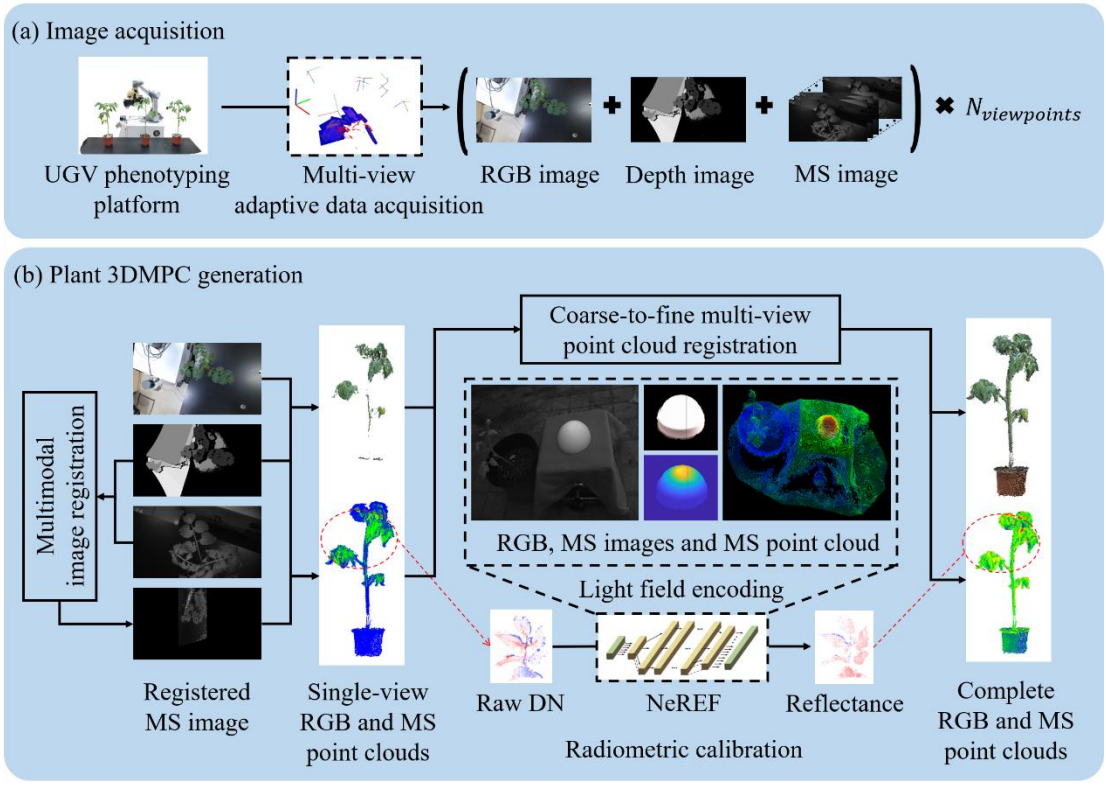
NeREF deployment pipeline: from start to application



Introduction

This document is the specific implementation method of Neural REference Field (NeREF), including several parts such as pre-data preparation, data preprocessing, dataset construction, model building and model application. Since the process involves the use of different software and the compilation and running of different types of code, we will provide data examples, software usage instructions or source code respectively. The relevant data and code can be found in Dropbox and Github.

Dropbox:
https://www.dropbox.com/scl/fo/78hl7uqd0s9ypct1lcq0k/AF0uAWLquMZqKe2cz1ohj8w?rlkey=lh8qdaxnyjsq143flrxowz6tp&st=g0yfs7bt&dl=0

Github:
https://github.com/DigBigPigForU/NeREF_pipeline

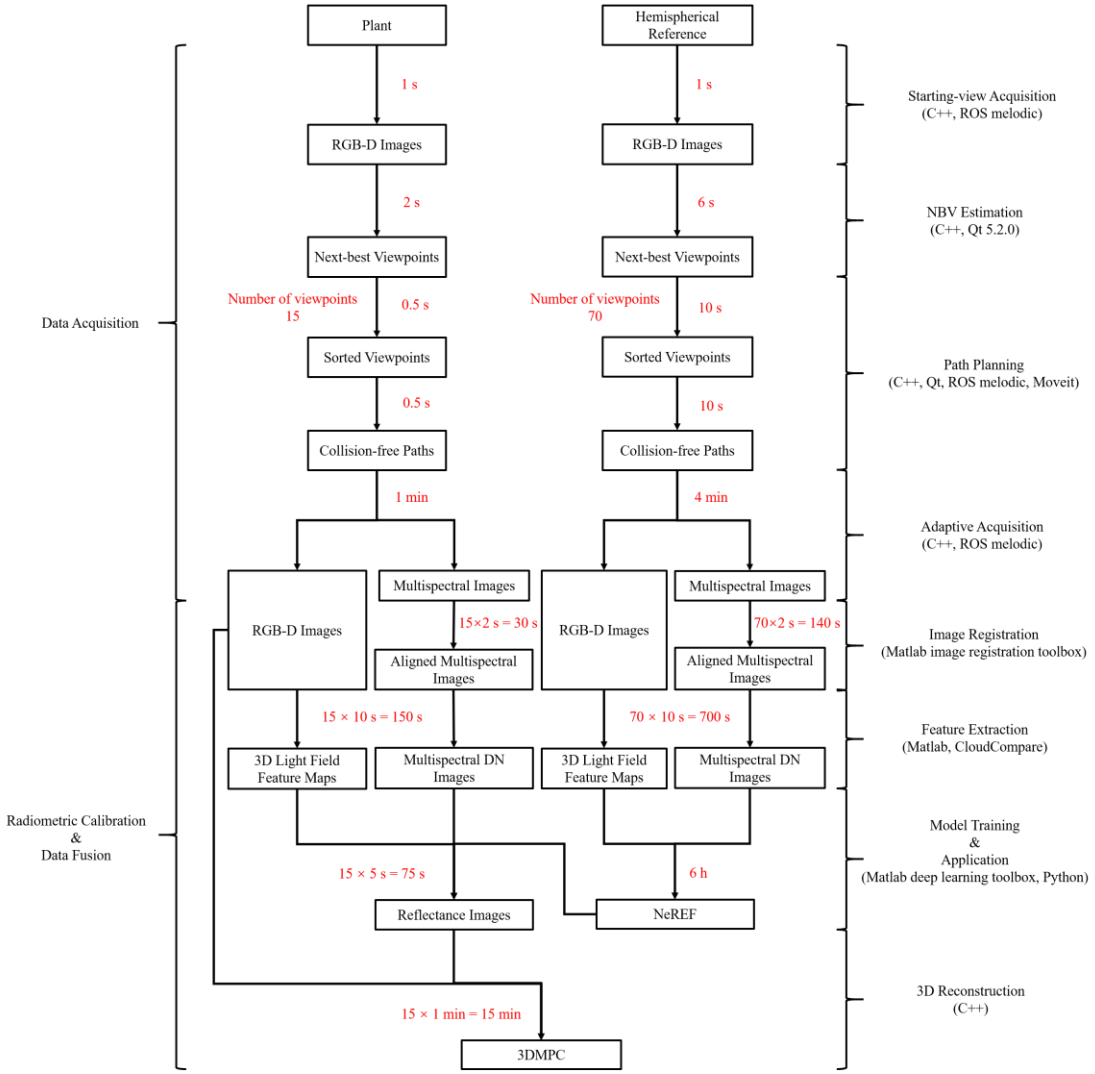


Figure 1. The full flow of NeREF modeling and application, including the software or environment used for each step and the step elapsed time.

Data preparation

Data acquisition with robotic arm

The data acquisition section contains the following parts: starting-view acquisition, Next-Best-View (NBV) estimation, path planning and adaptive acquisition. Since data acquisition needs to be adapted to the hardware, the hardware control and camera acquisition will not be expanded in detail here. If you have a robotic arm, you can configure your own Robot Operating System (ROS) environment for automatic acquisition by following the tutorial at https://www.ros.org/. In our study, we used the UR5 robotic arm and the melodic version of ROS. For programming we used C++ based on Qt 5.2.0. Rapidly-Exploring Random Trees (RRT)-based robotic arm path planning can be easily realized with Moveit. For the key parts of data acquisition, i.e., NBV estimation and viewpoint sorting, we provide source code that also runs under windows. Please refer to \ximea_dk\acthread.cpp, \ximea_dk\pcproc.cpp and \ximea_dk\pso.cpp.

Key data: images & poses

Regardless of the data acquisition method, the purpose is to acquire multi-view RGB-D images and multispectral images, as well as the pose of each viewpoint. If you don't have a robotic arm, you can also manually acquire images from multi-views using collocated cameras, and then use the Structure-from-Motion (SfM) algorithm to estimate the camera's viewpoint pose. However, special attention should be paid to the fact that the pose calculated by this method needs to be recalibrated using a standard reference.

Now, suppose you have obtained RGB-D images and multispectral images from multi-views, as well as the pose of each viewpoint. These images are saved in folders numbered for the different viewpoints and the poses of the viewpoints are recorded in Excel or .csv file. You can refer to the example data we provided (\hemi\1-28, UR_poses_with_matrix_hemi.xlsx and data_hemi.csv). If your robot is supplied with coordinates in joint space (e.g. hemi_joints_res.csv), you need to use ximea_dk\ur_kin.cpp to convert to coordinates in the base coordinate system of the robotic arm. If you get the (x, y, z, rx, ry, rz) format, you can use the xyzrxryrz2matrix.m file for conversion.

Data preprocessing

Routine calibration of multispectral images

Routine calibration of multispectral images includes elimination of various negative effects such as dark current effect, dark corner effect, etc. Eliminating the dark current effect requires first acquiring a dark current image and then subtracting it frame by frame. Dark corner calibration requires shooting against a uniform flat field and then calculating the calibration matrix, which is the ratio of all pixels to the center pixel of the image. Then multiply all the frames by this calibration matrix. If your camera is also snapshot, it is recommended to use full_frame2parts.m to split each band image from the raw image to calibrate it separately. You can refer to the example data in \split_frames. In addition, other calibrations, such as linearity calibration, cosine correction, etc., can be made according to the actual conditions of the camera. Finally, replace the images in \hemi\1-28 with the calibrated images.

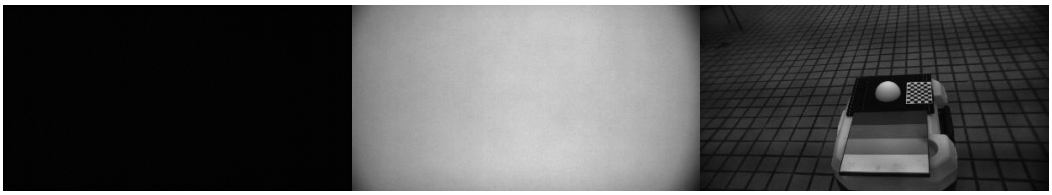


Figure 2. The dark current image, the dark corner image and the example image of the hemispherical reference.

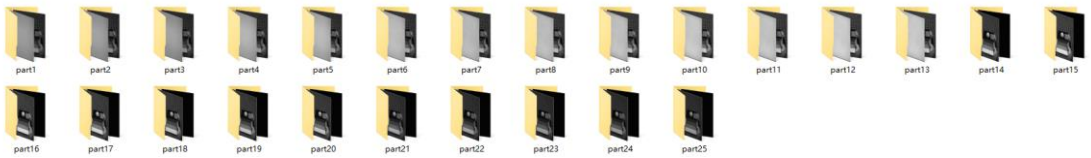


Figure 3. Split images of each band.

Image segmentation

The RGB image and the multispectral image are first imported into the Matlab workspace. Then the plant pixels were segmented using Matlab Color Thresholding Toolbox and Image Segmentation Toolbox as follows.

Figure 4. Import RGB image in Matlab.

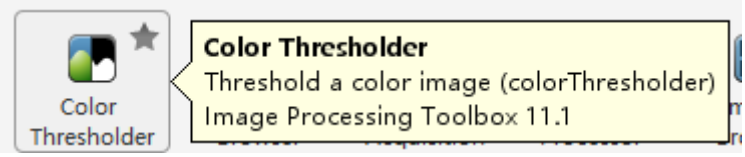


Figure 5. Color thresholder toolbox in Matlab.

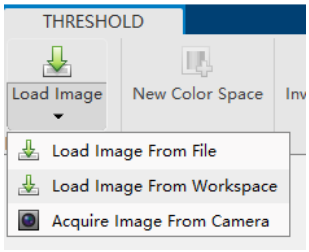


Figure 6. Load image from the workspace.

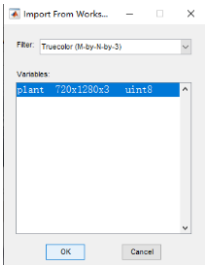


Figure 7. Choose the plant RGB image.

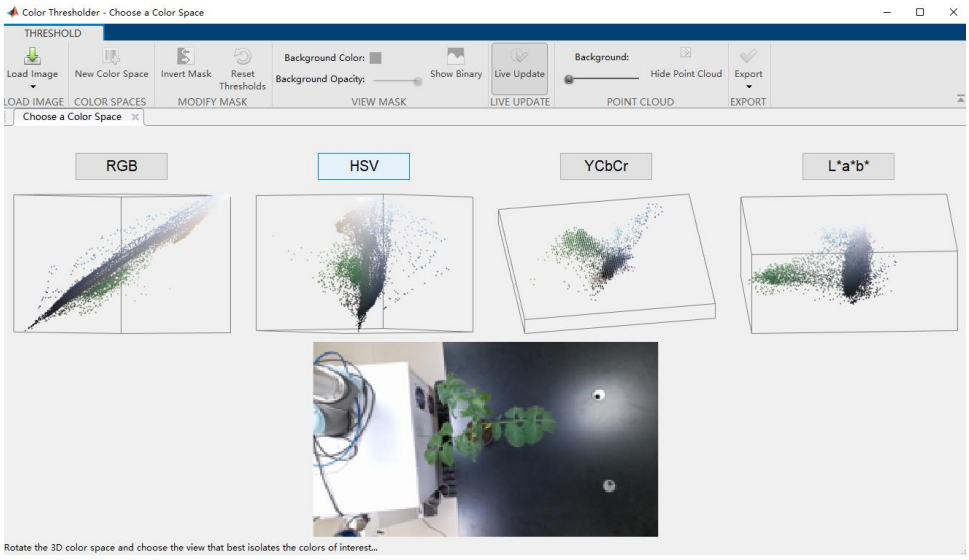


Figure 8. Choose the HSV color space.

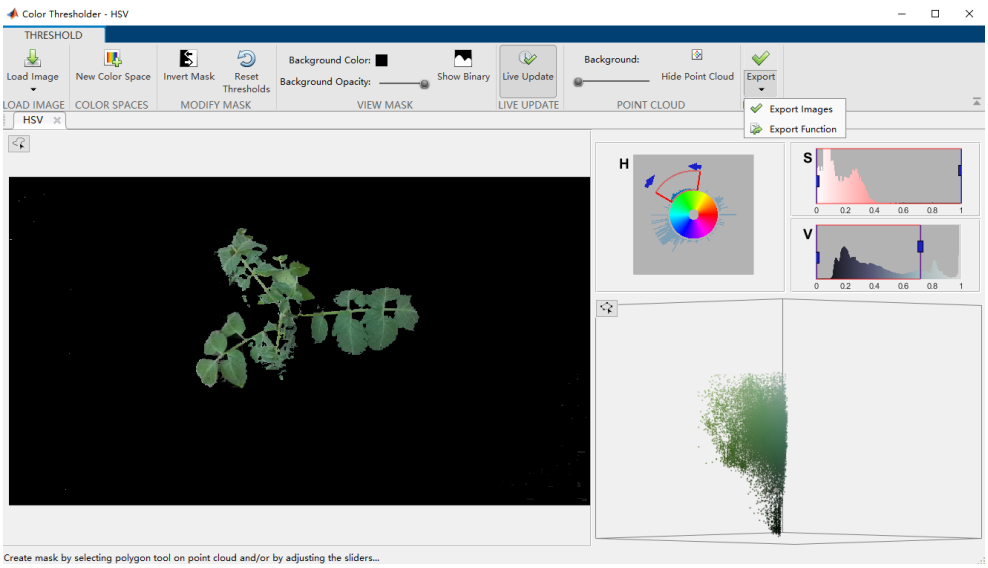


Figure 9. Select the appropriate threshold and export the function code.

Untitled*

```
function [BW,maskedRGBImage] = createMask(RGB)
%createMask  Threshold RGB image using auto-generated code from colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace and
%  range for each channel of the colorspace were set within the app. The
%  segmentation mask is returned in BW, and a composite of the mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 16-Jun-2024
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2hsv(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.226;
channel1Max = 0.416;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.000;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 0.718;
```

Figure 10. You can use this code for batch processing.

Figure 11. Import multispectral image into the workspace. (here are band 2 and band 7)

Figure 12. Calculate the NDVI image.

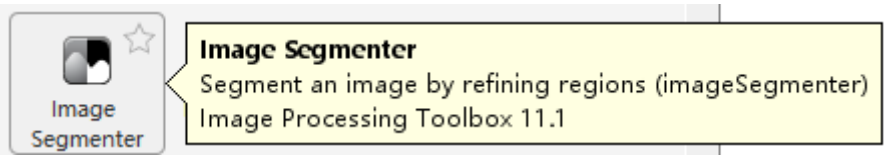


Figure 13. Use the image segmenter toolbox for segmentation.

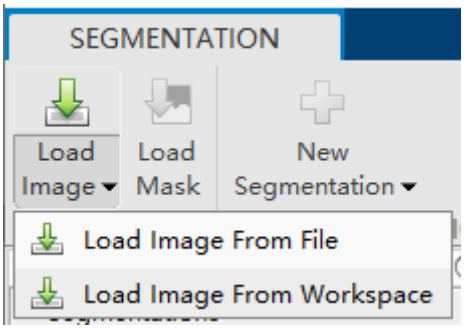


Figure 14. Load the image from the workspace.

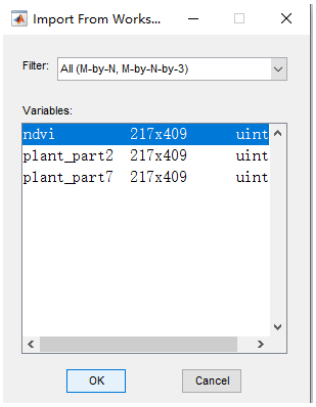


Figure 15. Choose the NDVI image.

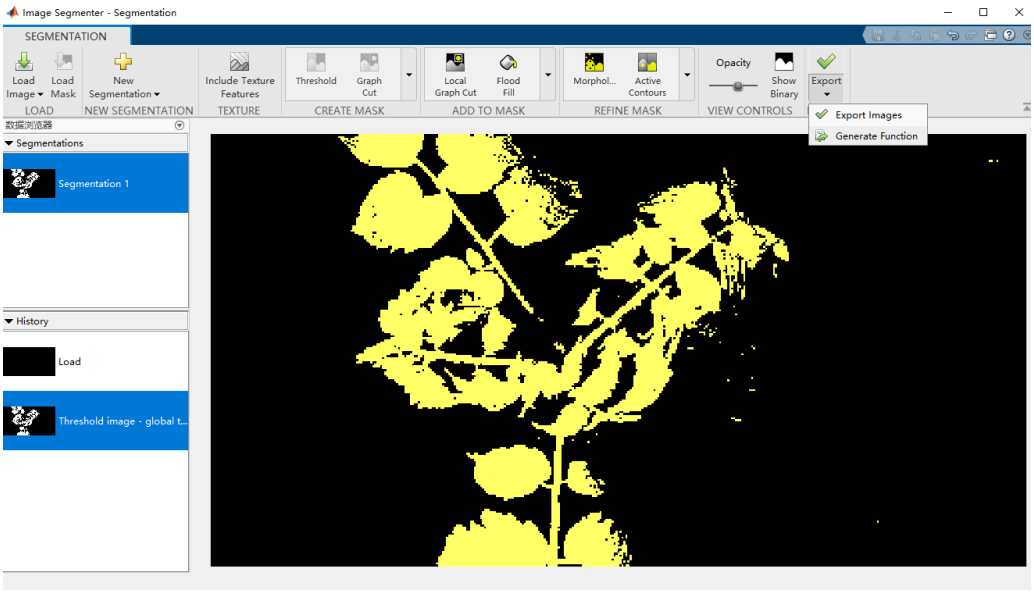


Figure 16. Select the appropriate threshold and export the function code.

```
Untitled2*
function [BW,maskedImage] = segmentImage(X)
%segmentImage Segment image using auto-generated code from imageSegmenter app
%  [BW,MASKEDIMAGE] = segmentImage(X) segments image X using auto-generated
%  code from the imageSegmenter app. The final segmentation is returned in
%  BW, and a masked image is returned in MASKEDIMAGE.

% Auto-generated by imageSegmenter app on 16-Jun-2024
%----------------------------------------------------


% Adjust data to span data range.
X = imadjust(X);

% Threshold image - global threshold
BW = imbinarize(X);

% Create masked image.
maskedImage = X;
maskedImage(~BW) = 0;
end
```

Figure 17. You can use this code for batch processing.

Multimodal image registration

The purpose of this step is to align the depth image and the multispectral image. The segmented G-channel image and the 7th band (740.7 nm) image are first extracted into the workspace. Matlab code 'J = imhistmatch(I,ref)' adjusts the histogram of the 2-D grayscale or truecolor image I such that the histogram approximately matches the histogram of the reference image ref. Next, use the Matlab image registration toolbox to align the adjusted image pairs, which can be found in the following process.

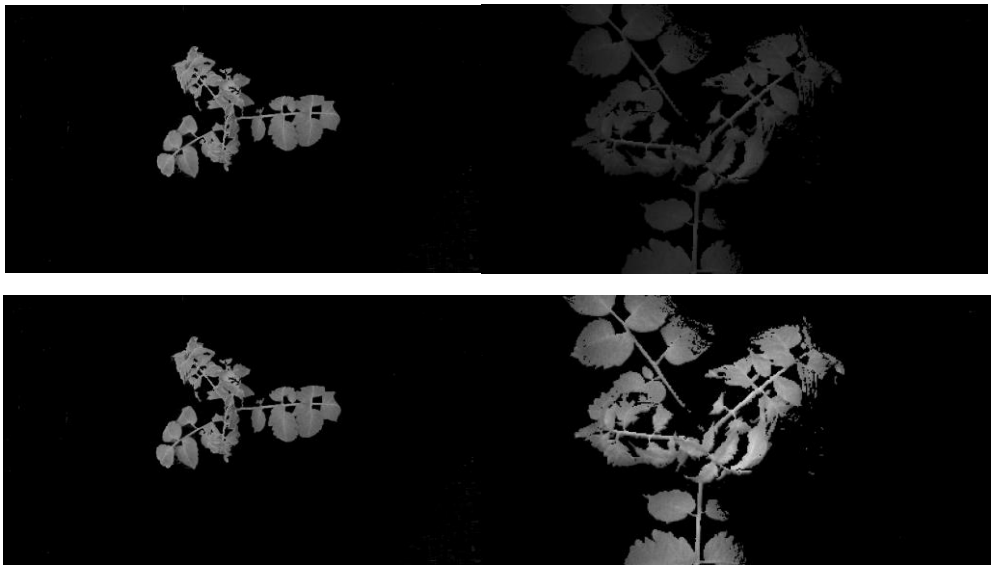


Figure 18. The segmented G-channel image and the 7th band (740.7 nm) image before and after adjusting.

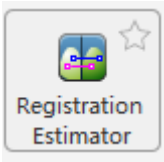


Figure 19. Use the registration estimator toolbox to align the G and 7th image.

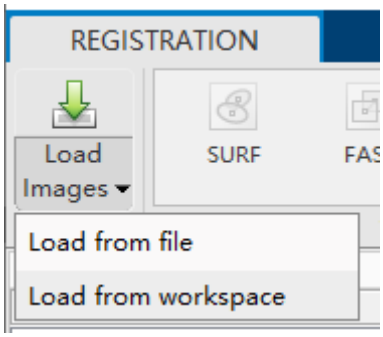


Figure 20. Load the image pair from the workspace.

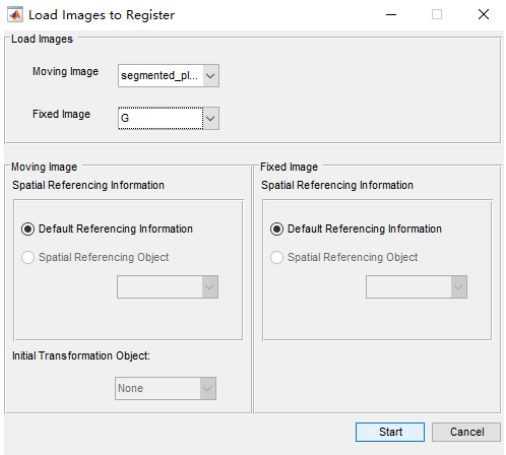


Figure 21. Set up fixed and moving images。

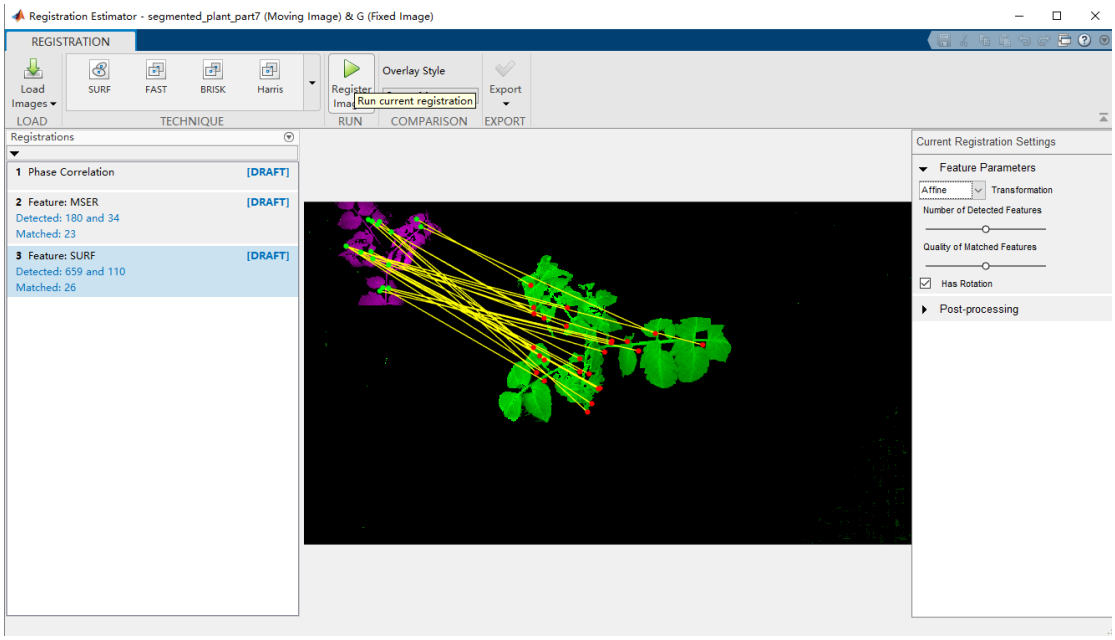


Figure 22. Select SURF algorithm and appropriate transformation and thresholds.

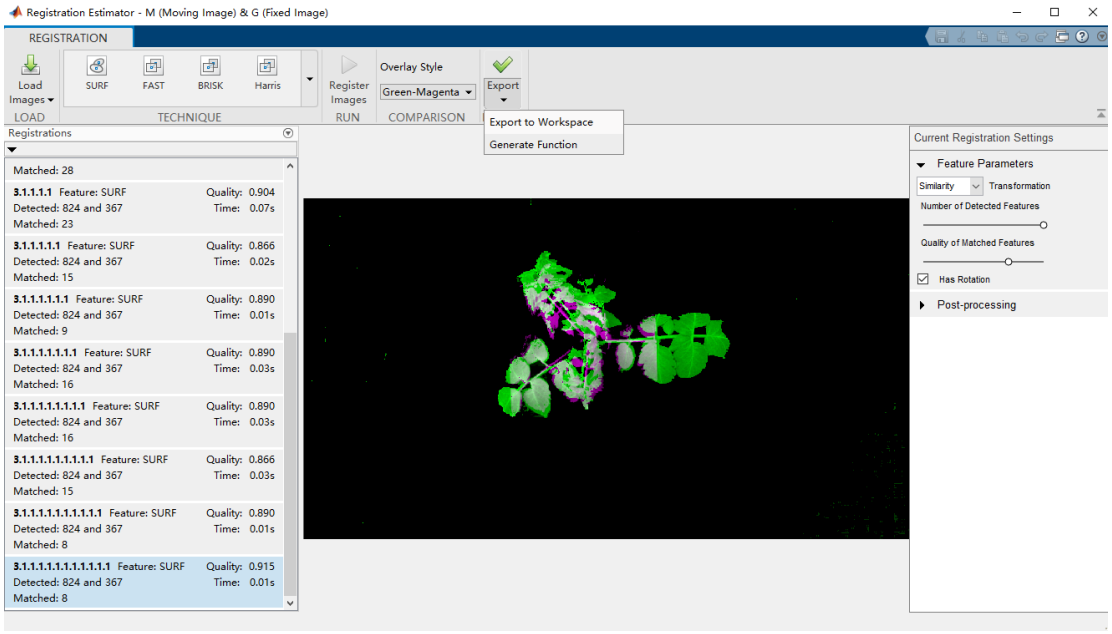


Figure 23. Export the function code for batch processing.

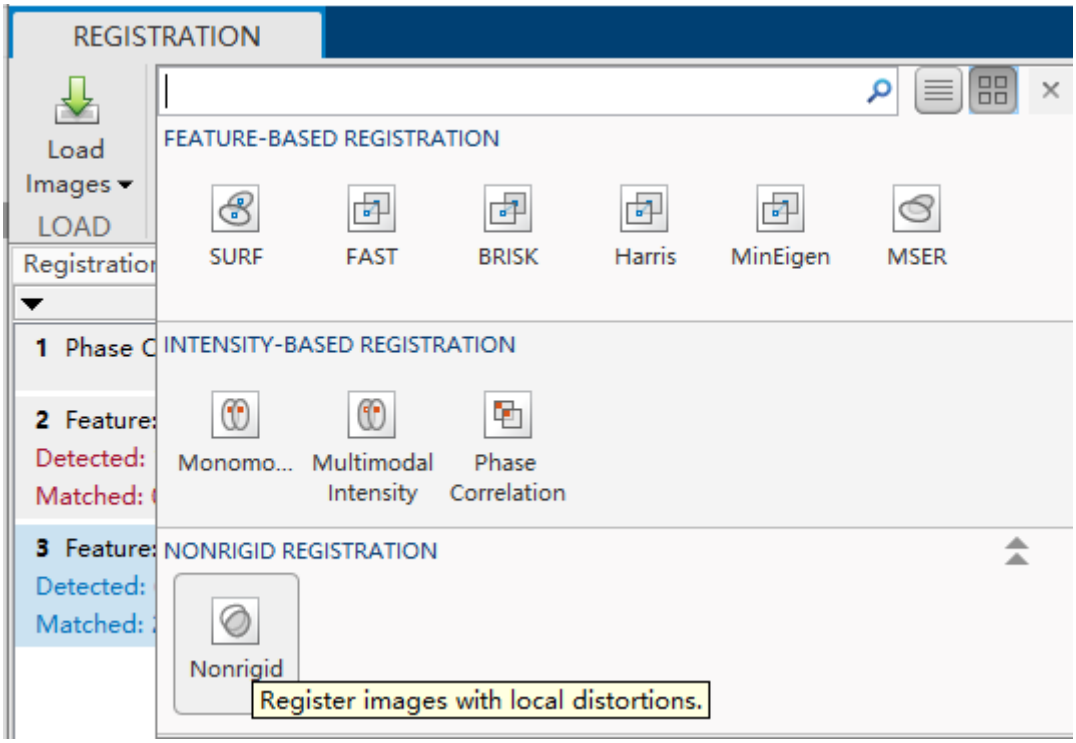


Figure 24. Import SURF results and use non-rigid methods.

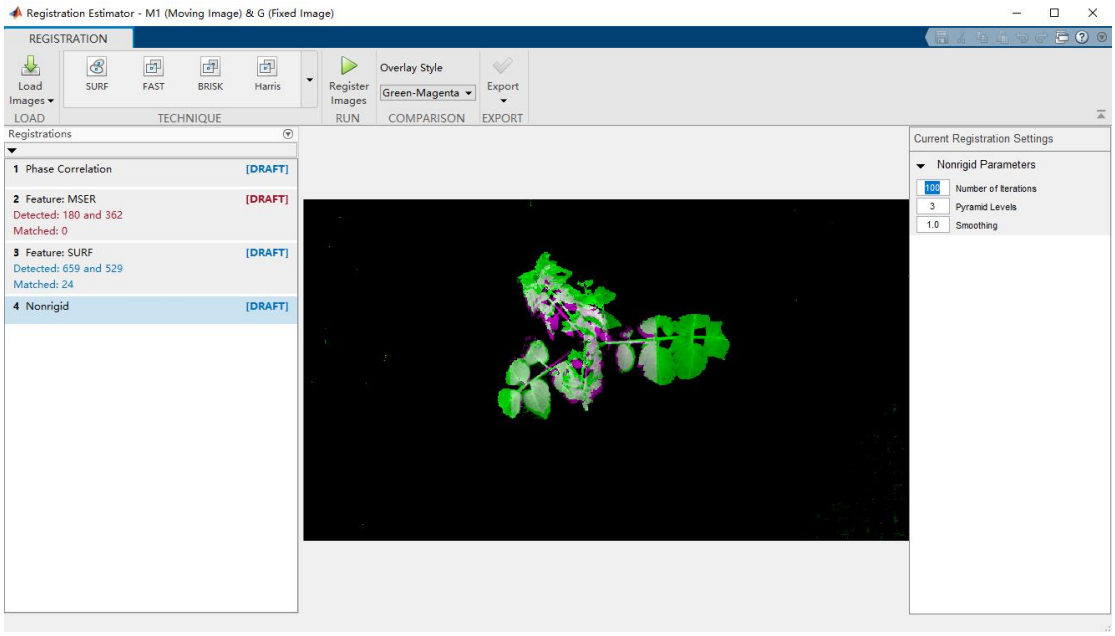


Figure 25. Select appropriate thresholds.

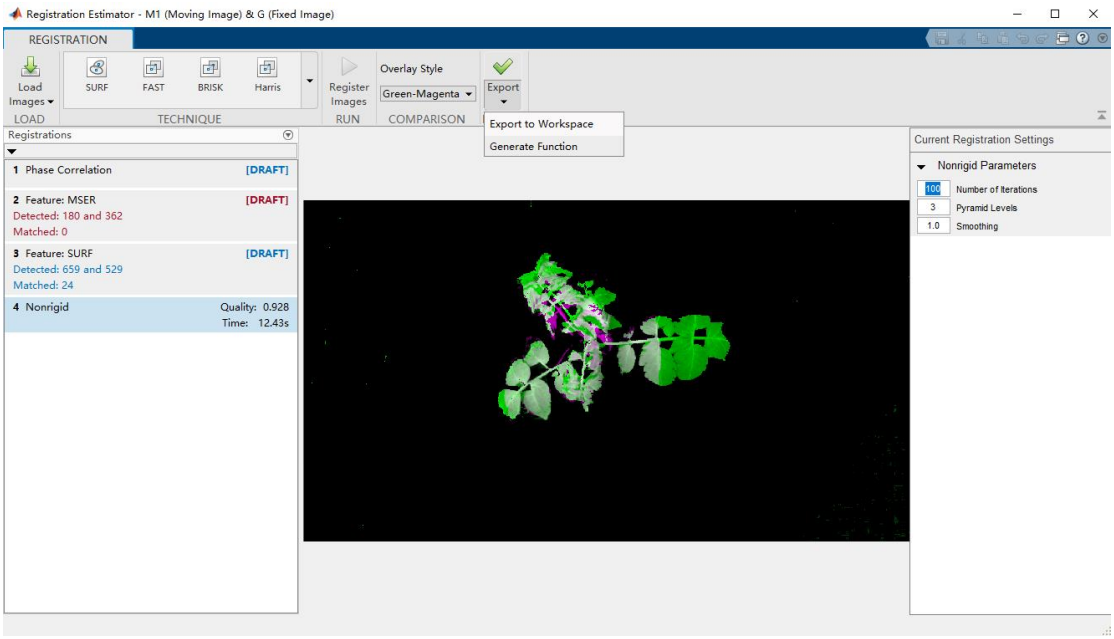


Figure 26. Export the function code for batch processing. For different band images acquired from the same viewpoint, please use the same image transformation matrix or deformation field.

After completing the above steps, you can place the aligned images in the appropriate folder in \hemi\1-28. Note that the above examples use plants, and the approach is the same for hemispherical reference. An example of aligned hemispherical reference images is provided in /register_MS_hemi.

Figure 27. Aligned multispectral images of hemispherical reference.

Single-frame 3DMPC Reconstruction

This step generates the single-frame 3D Multispectral Point Cloud (3DMPC) with Digital Number (DN) values for feature extraction, not the final reflectance point cloud. We provide a C++ code (\RGBDFrames2PC\main.cpp) that can be used to generate 3DMPC in .txt format (\hemi\N_base_pc_with_zero.txt). This code can transform the point cloud from the camera coordinate system to the robotic arm base coordinate system according to the provided viewpoint pose while converting from 2D to 3D (Sample data with all backgrounds removed is detailed in hemis.bin, which can be loaded directly into CloudCompare).

```
//txt point cloud and dataset vx vy vz nx ny nz ms_1-25
ofstream zos(save_file_path);
for (int i = 0; i<Num; i++)
{
    if (Z[i] == 0 || (unsigned(R[i])==0 && unsigned(G[i]) == 0 && unsigned(B[i]) == 0)) //Z ==0 or rgb ==0
    {
        continue;
    }
    else
    {
        zos << X[i] << " " << Y[i] << " " << Z[i] << " " << unsigned(R[i]) << " " << unsigned(G[i])<< " " << unsigned(B[i]);
        for (int b = 0; b<25; b++)
        {
           zos  << " " <<  unsigned(MS_vec[b][i]);
        }
        zos  << " " << VX[i] << " " << VY[i] << " " << VZ[i];
/         zos  << " " << VX[i] << " " << VY[i] << " " << VZ[i] << " " << NX[i] << " " << NY[i]<< " " << NZ[i];
    }
    zos << endl;
}
```

Figure 28. Key code for generating .txt DN point cloud files. Each row represents a point sample, from columns 1-34 are X Y Z R G B Band1-25 VX VY VZ (in the coordinate system of the robotic arm base). The length units here are all meter, and the range of DN values and RGBs are all 0-255.

Dataset construction

Observation vector estimation

The previous \RGBDFrames2PC\main.cpp has calculated VX VY VZ, which can be imported into CloudCompare for visualization as scalar field (SFs).

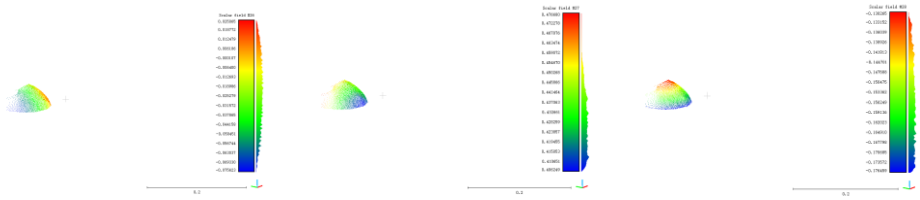


Figure 29. Visualization of VX VY VZ.

Normal vector estimation

This step imports the point cloud into CloudCompare for calculation and then exports the results as SFs. Similarly, if batch processing is required, C++ code based on point cloud library (PCL) or Open3D-based Python code can be written.

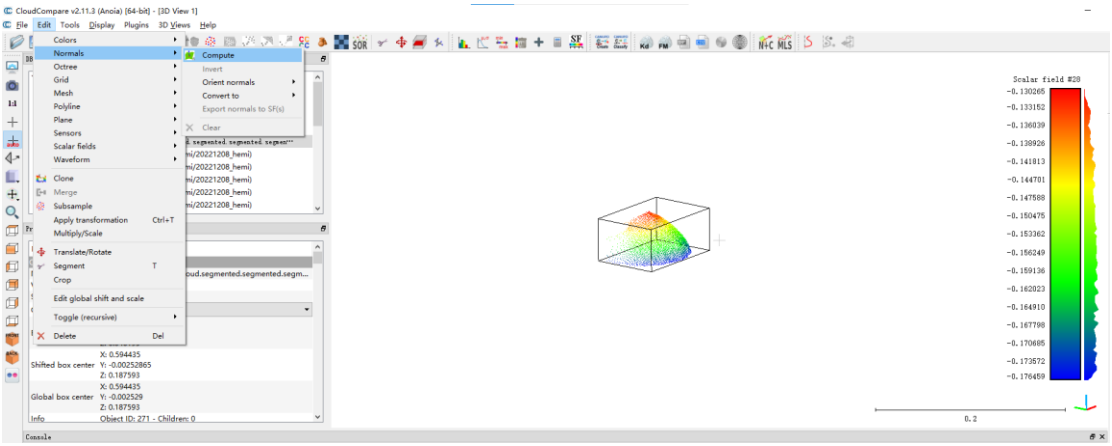


Figure 30. Calculating normal distributions with CloudCompare.

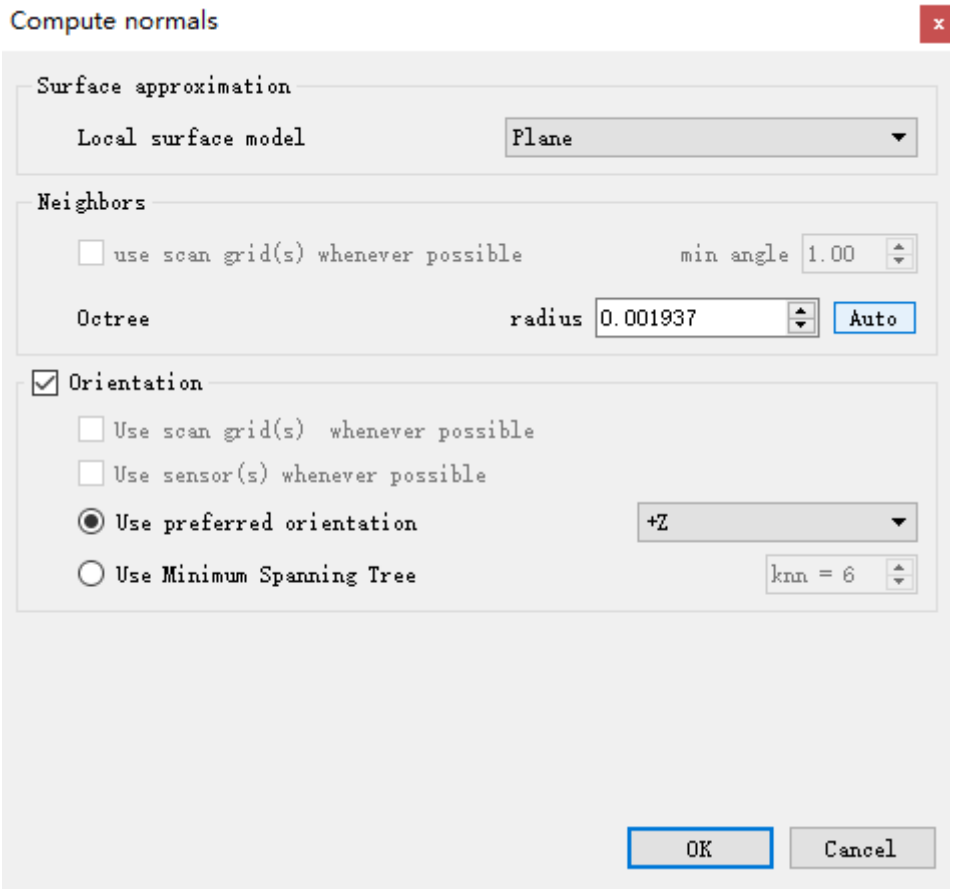


Figure 31. Set the parameters, must use +Z for preferred orientation.

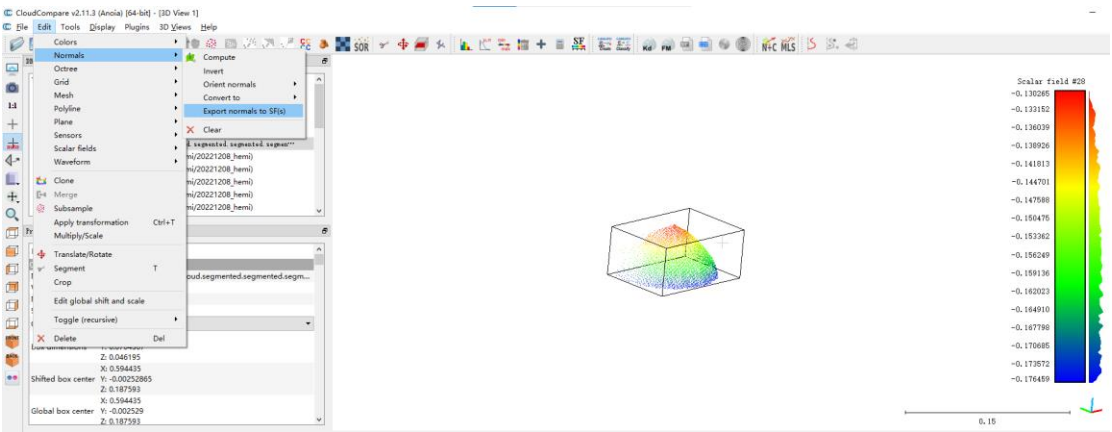


Figure 32. Export normal to SFs.

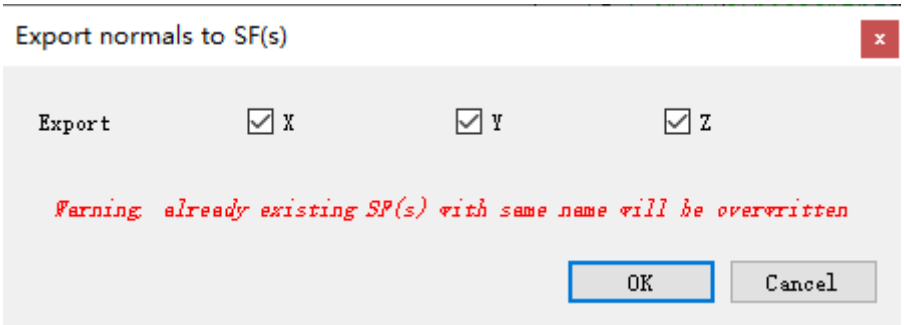


Figure 33. Select all axes.

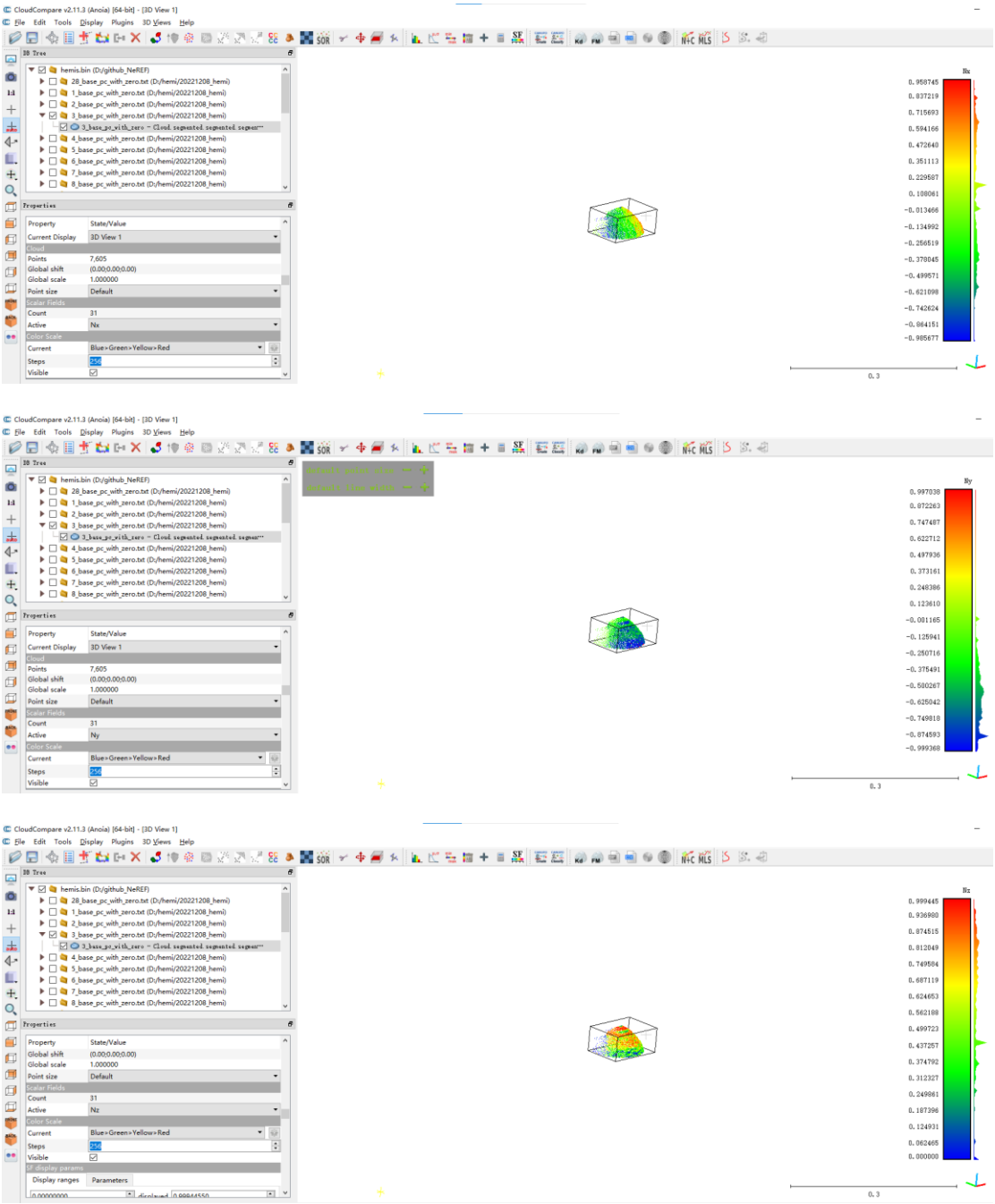


Figure 34. Visualization of NX NY NZ.

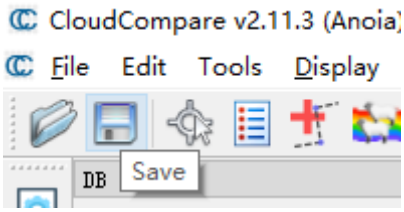


Figure 35. Export the estimated normals.

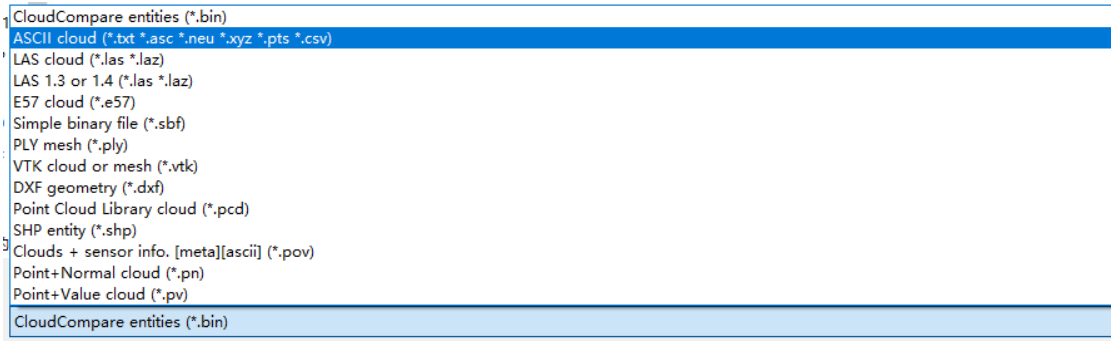


Figure 36. Select the ASCII cloud format.

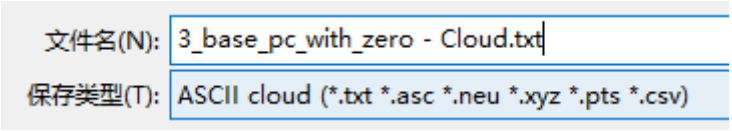


Figure 37. Add .txt suffix.

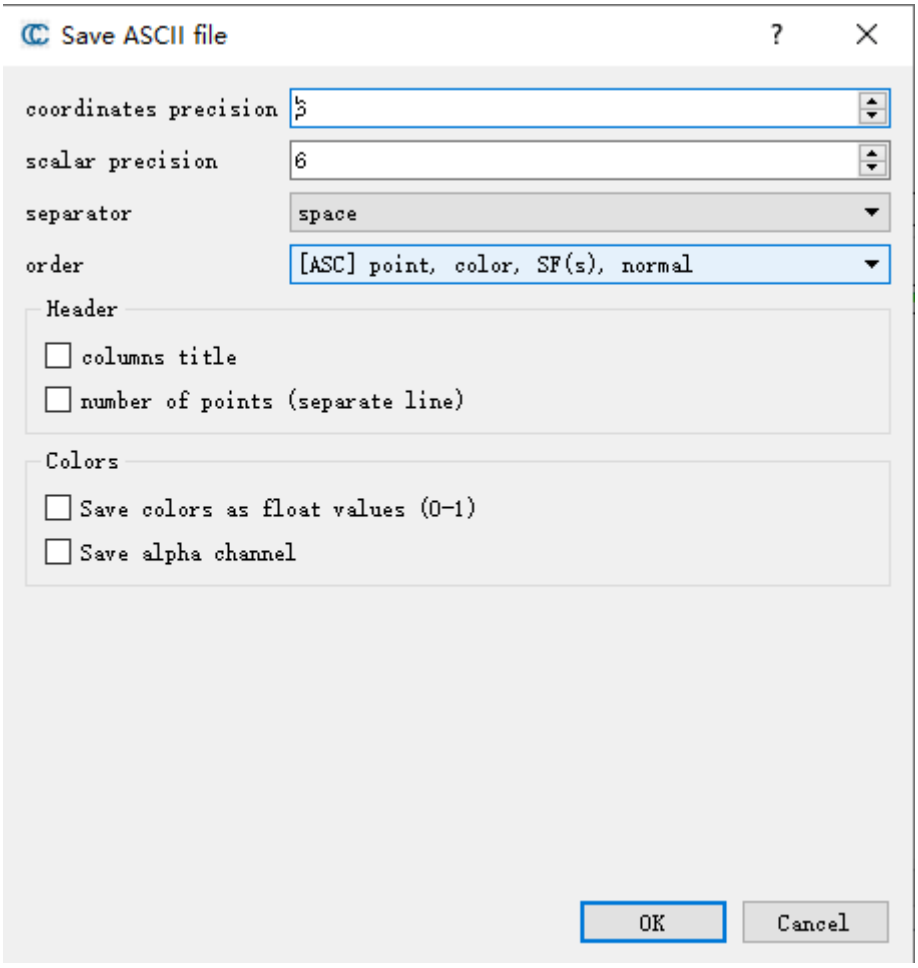


Figure 38. Saving the normals after all SFs to get a .txt point cloud in the format of X Y Z R G B Band1-25 VX VY VZ NX NY NZ.

DN value normalization

This step can be done directly in CloudCompare by simply dividing the SFs of all DN values

by the maximum value. Batch processing can be done in Excel and Matlab (see data.txt, data_for_training.xlsx, dataset.mat and normalized_dataset.mat).

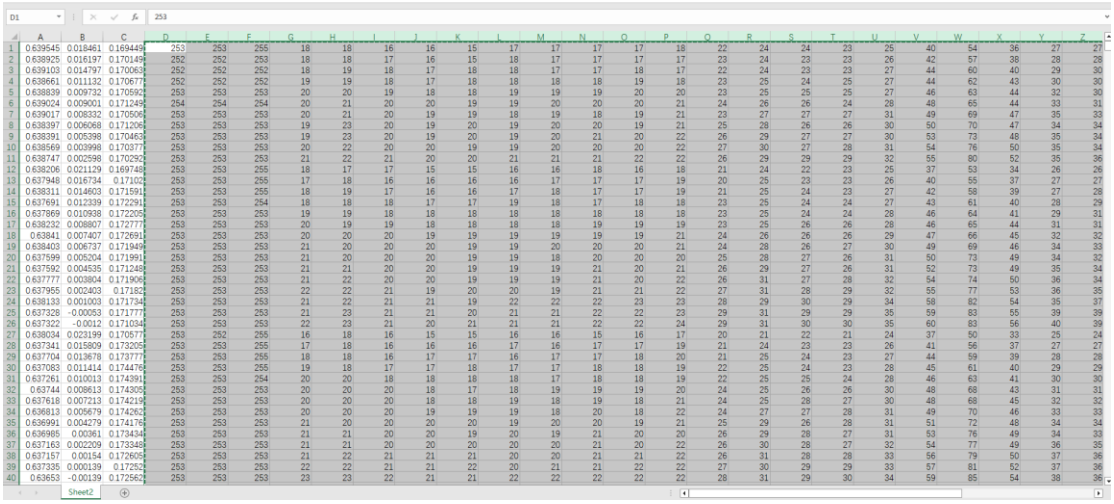


Figure 39. Aggregating the point cloud data from all the viewpoints gives you the original dataset (RGB and DN are in the range of 0-255).

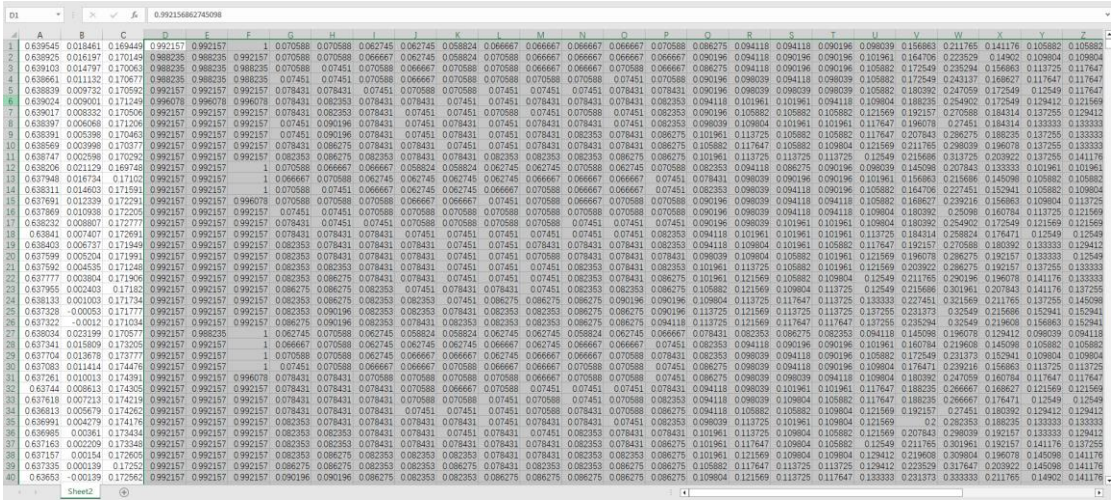


Figure 40. Normalized dataset (RGB and DN are in the range of 0-1).

Model building

Modeling with Matlab

The NeREF can be built using Matlab's deep learning toolbox. Although we have extracted the 3D light field features manually, we still recommend that you use the GPU (e.g. NVIDIA GeForce GTX 3090) for faster training given the huge amount of data. Here we provide a matlab script (NeREF_training.m) for reference.

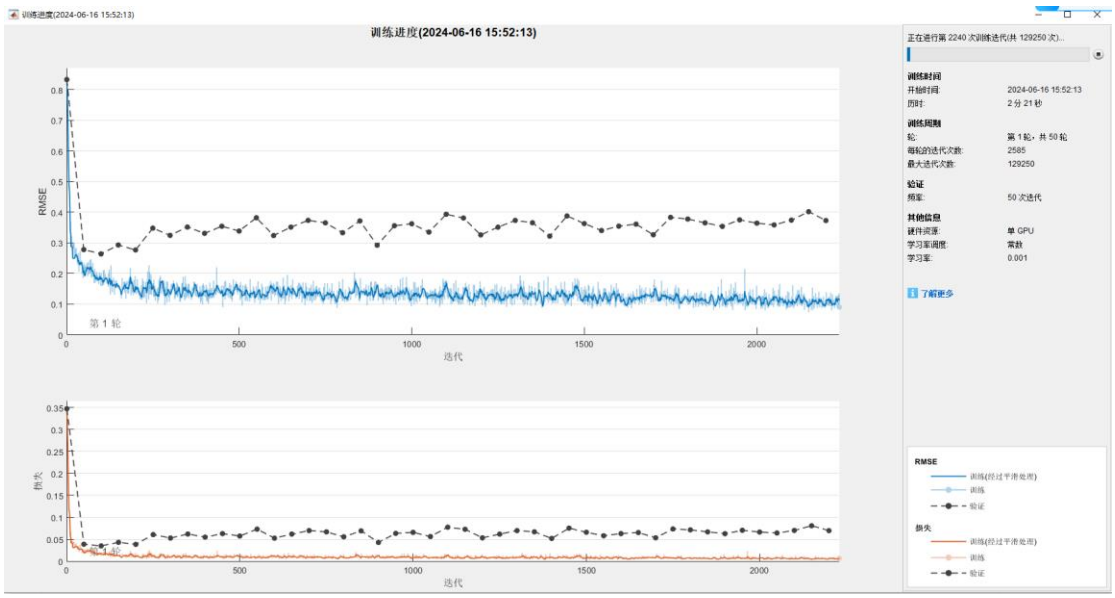


Figure 41. Train the NeREF model.

Modeling with Python

We also provide a code (NeREF_training.py) for training NeREF via Python using TensorFlow and Keras. Ensure you have TensorFlow and scikit-learn installed in your Python environment. You can install them using: pip install tensorflow scikit-learn. This code should be run in a Python environment where your data x and y are already loaded as NumPy arrays. If they are stored in files, you can load them using np.load or any other appropriate method.

Model application

Predicting reflectance for plants

The prediction step is very similar to the training step described above, and only the 3D light field features extracted from the single-frame plant 3DMPC are inputted into NeREF to predict the corresponding reference DN value. There is no need to repeat it here. Finally, dividing the original DN value of the plant 3DMPC by the predicted reference DN value gives the plant reflectance.

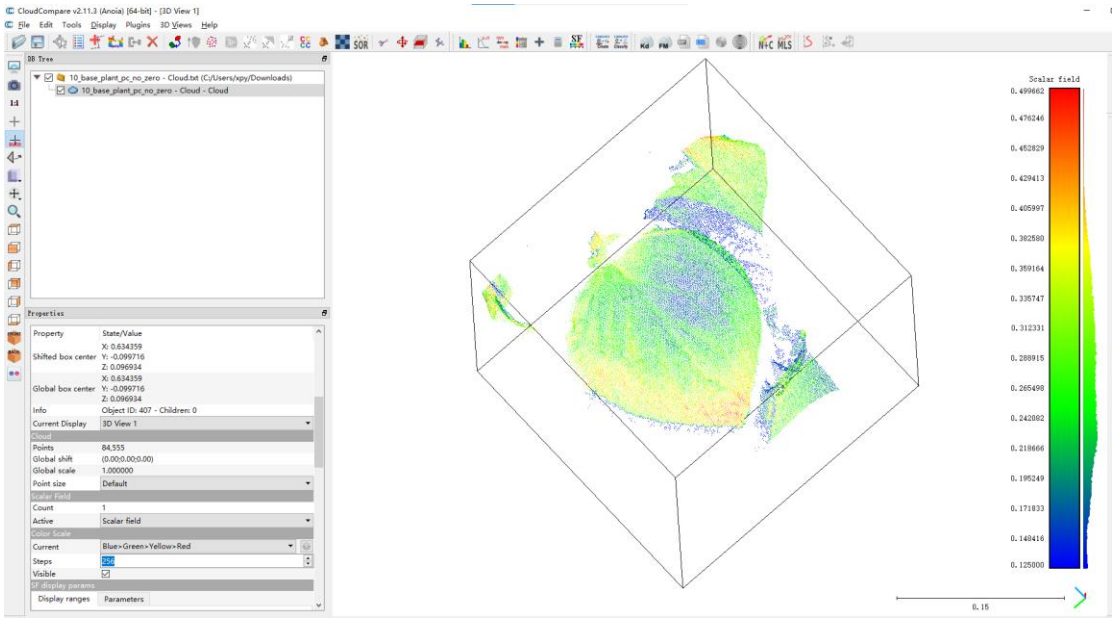


Figure 42. Visualization of the plant reflectance point cloud.

Fusion of multi-view 3DMPCs

The fusion of all single-frame plant reflectance point clouds requires two stages of coarse and fine registration, which is the same as general point cloud registration. Coarse registration has been done in the step of 'Single-frame 3DMPC Reconstruction'. Fine registration requires the utilization of the Iterative Closest Point (ICP) algorithm or its variant. Here we can use the built-in ICP algorithm in CloudCompare and visualize the results. You can select a point cloud as the reference and then register all other views to that point cloud. ICP algorithms are available in PCL and Open3D, if you need to batch process you can use C++ or Python.

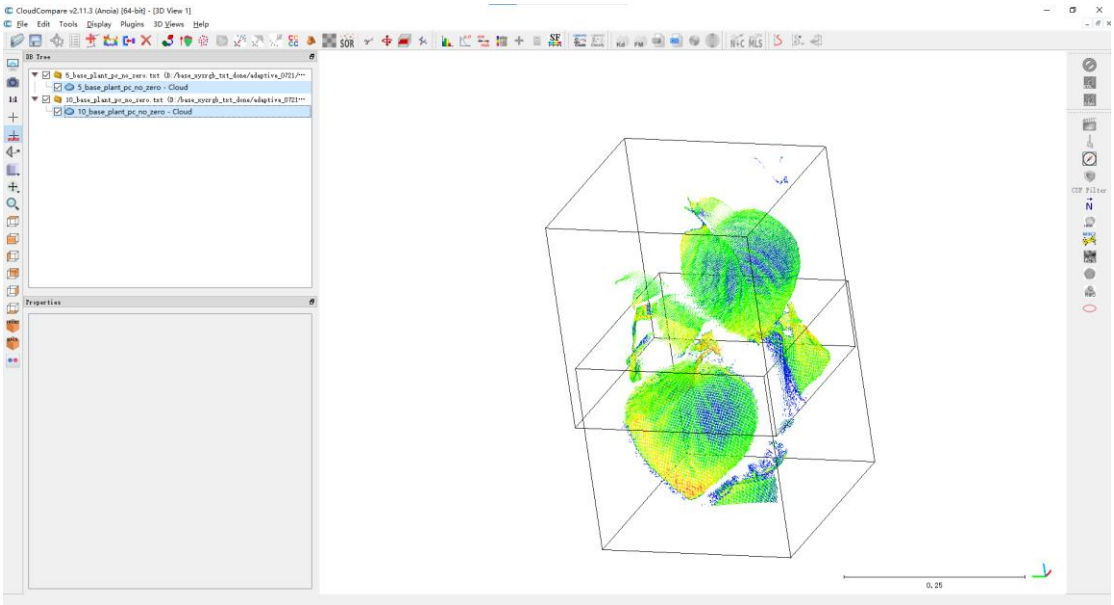


Figure 43. Import the reference and all the to-be-aligned 3DMPCs.

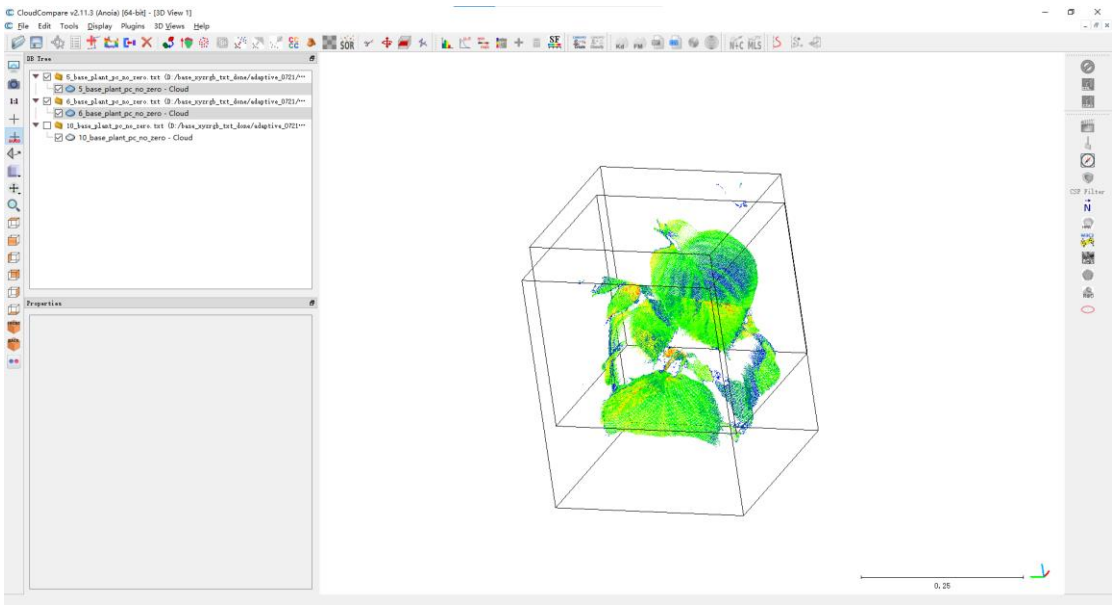


Figure 44. Select the reference and one of the 3DMPCs to be aligned.

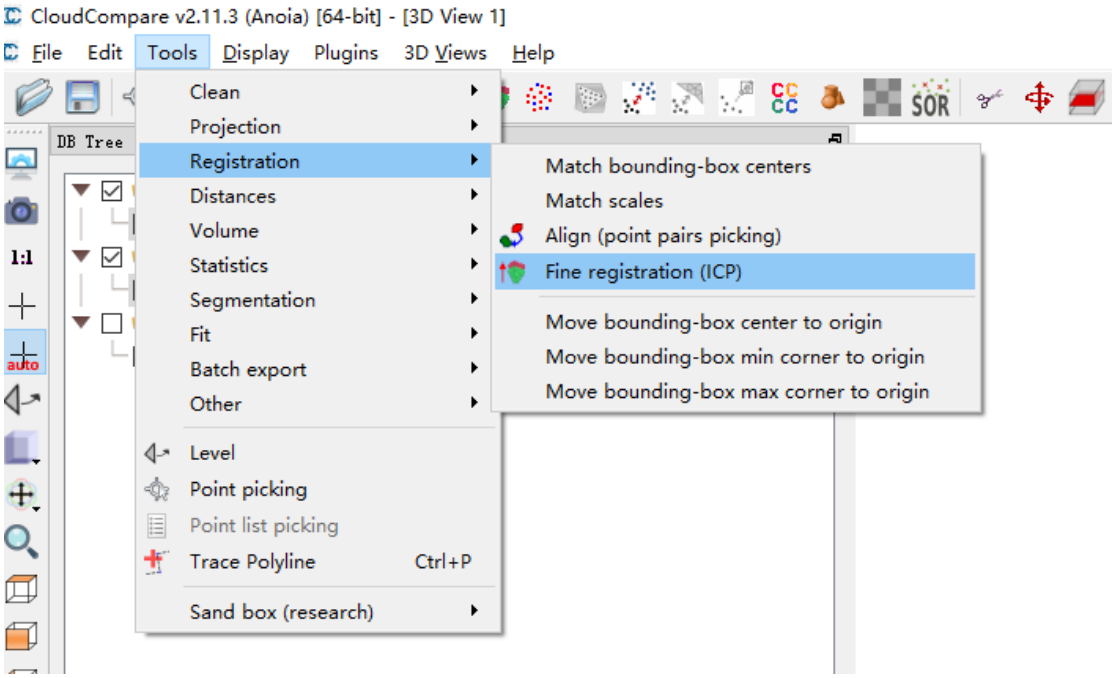


Figure 45. Use the ICP algorithm to perform the fine registration.

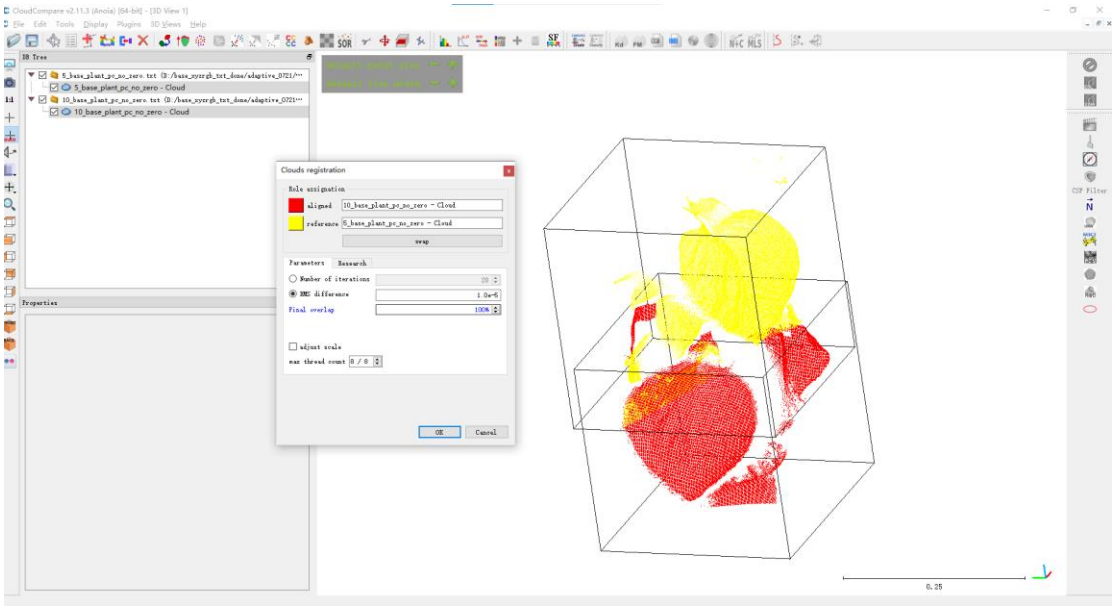


Figure 46. Set up the aligned and the reference 3DMPC as well as the parameters.

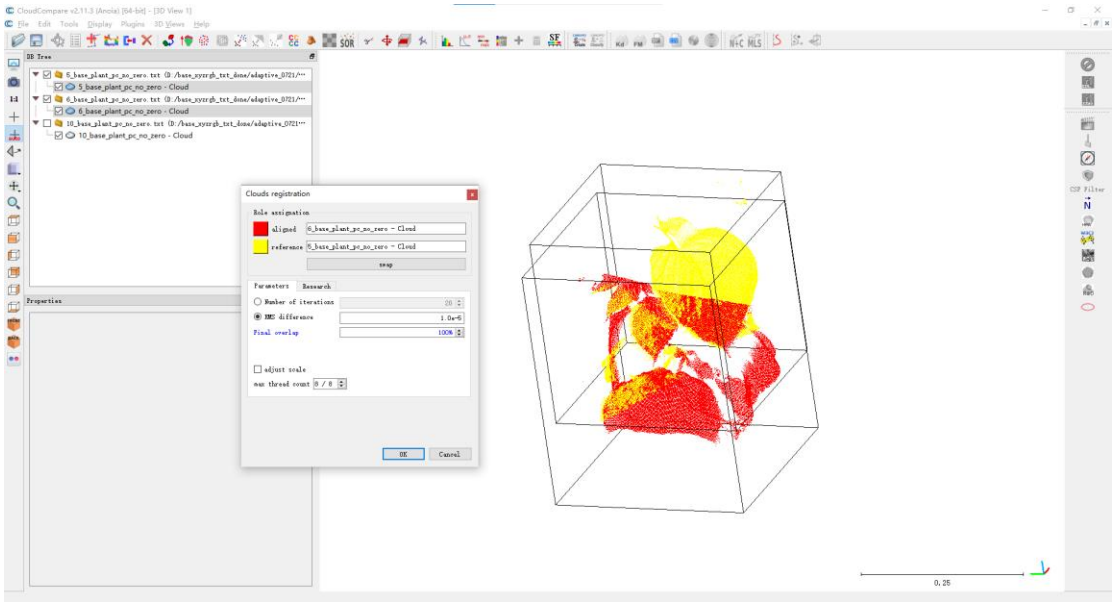


Figure 47. Use the same operation for all other 3DMPCs to be aligned.

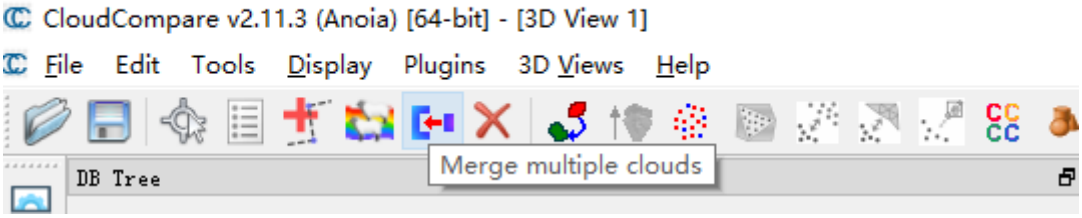


Figure 48. Merge the reference 3DMPC with all other aligned 3DMPCs.

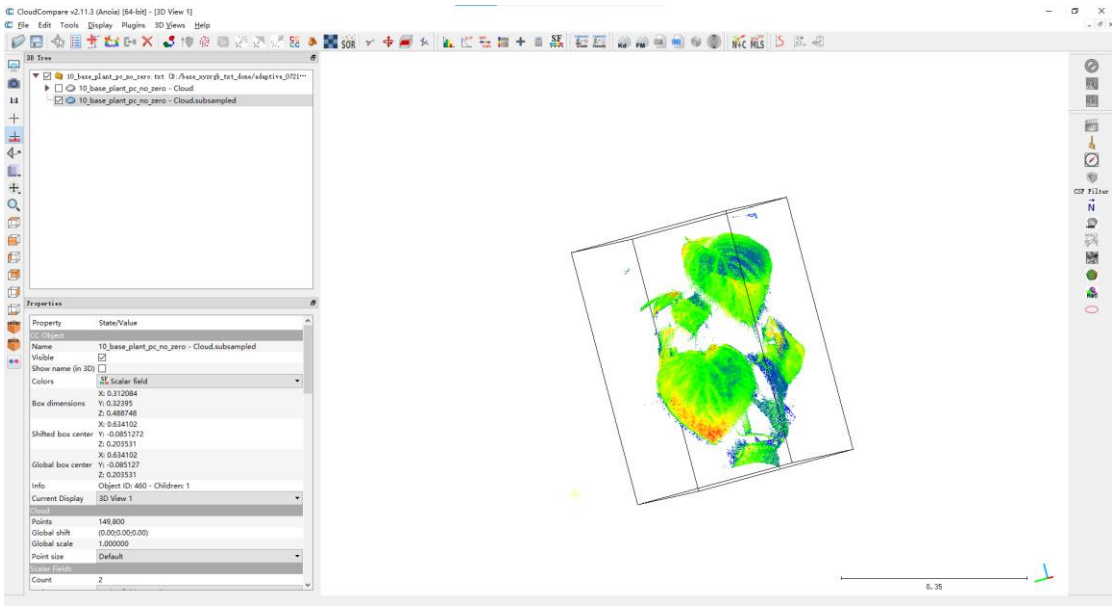


Figure 49. Use downsampling and filtering to generate 3DMPC of the complete plant, which can be exported for additional analysis.