# Documentation for Using VirT&R

## Introduction

This document is intended to act as a more detailed ReadMe for using VirT&R (virtual teach and repeat). It will do this by guiding a new user through the required software dependencies and programs needed, as well as the steps involved in moving data products between them.

The entire process for making a virtual teach graph can be done on the ASRL asterisk remote server, and some effort has been made to ensure that the pipeline is as cohesive and integrated as possible (less need for moving files from place to place to be used/edited). Further effort is required, but it is an improvement over the working version used to produce results in the VirT&R paper.

The desired end result that will be added to this document, once achieved, is a single program where commands are entered in the terminal to perform all actions specific to the programs used with launch files all in one place rather than needing to use commands specific to those software programs in separate places.

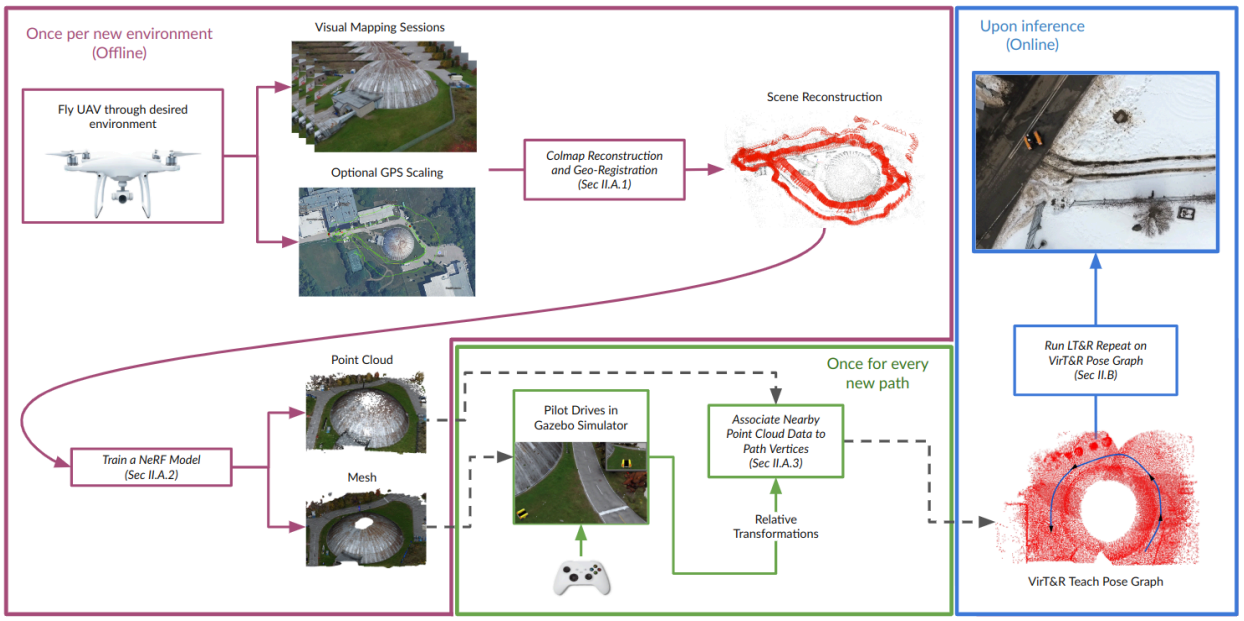
# Required Software

The main software used in the VirT&R pipeline includes: **Visual Teach and Repeat** (virtual teach and repeat branch), **Gazebo**, **Blender**, **Colmap**, **NerfStudio**, and some custom scripts for initial data processing post UAV scene capture and gazebo path logging (that can be found in the virtual teach and repeat branch's package vtr_virtual_teach>scripts).

NOTE: Vtr3 is set up to use Ros Humble (Ros 2) and the specific Gazebo installation that is compatible with the model of the clearpath warthog requires Ros Noetic (Ros 1). Switching between these ros versions will be necessary and, to keep things simple, it is recommended to use two different terminals running the docker container so there is no switching back and forth. The procedure flows from using Ros 1 to Ros 2 and a switch should only need to be made once anyways.

# Phases

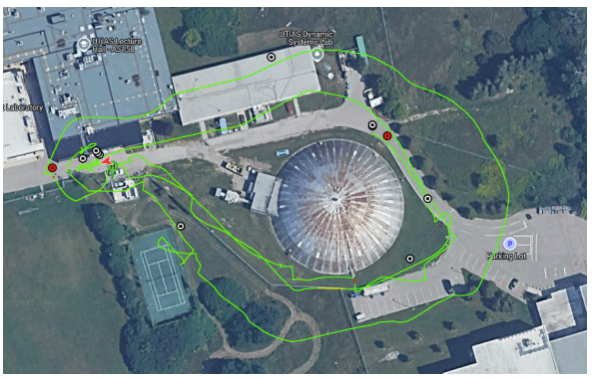
Making a virtual teach graph consists of three phases (not to be confused with the overall three phases of the whole VirT&R pipeline. The three phases, in order, are: UAV data processing,NeRF training, and point cloud/mesh path association for pose graph creation. These phases can be seen in **II.A.1**, **II.A.2**, and **II.A.3** of the VirT&R paper as shown in the overall pipeline diagram below.



# Phase 1: UAV Data Processing

The following steps detail how to use and fly the DJI Phantom 4 UAV, extract frames and georegister them, and scale them down to be ready for NeRF training.

1. Unbox the DJI Phantom 4 UAV and connect your phone via usb cable to the controller (make sure you have the DJI Go 4 app downloaded).
2. Ensure there is an SD card in the UAV (slot on the side) with 20+ GB of space.
3. Go outside to a safe spot that isn't busy to set up the UAV by placing it on the ground and putting on the propellers.
4. Turn the UAV and controller on, then open the DJI Go 4 app.
    a. If you haven't already, go through the compass and IMU calibration steps in the menu at the top right corner of the screen.
5. Now that you are ready to fly (the UAV has 10+ satellites connected, is warmed up, and the lights are flashing green), move the UAV to the starting point you desire for scene capture if not already there.
6. Take off with the automated take off button.
    a. To fly the drone, the right joystick is used for translation (forwards/backwards/left/right) and the left joystick is used for elevation and yaw. The scroll wheel at the top left of the controller controls the camera angle.
7. The goal is to fly three passes around the desired environment to capture it sufficiently. There should be an inwards facing pass at 20 m, a nadir facing path over the robot's desired path at 15 m, and a POV facing path over the robot's desired trajectory at 5-10 m. This picture shows these three 'loops' through the environment around the Mars Dome at UTIAS (comes from a step further in the process) but is useful to show you what must be flown:

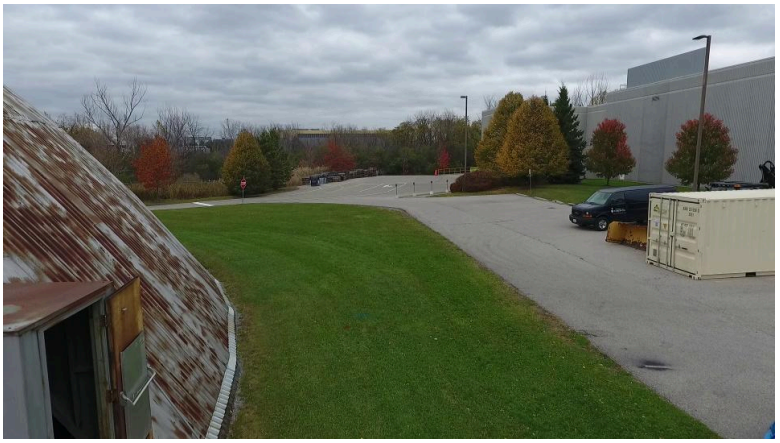


8. The first pass to fly is the ~20 m, 45 degree inwards facing pass.
    a. Identify the 'center' of your scene (ie. with the Mars Dome Loop, the center was roughly the center of the Mars Dome).
    b. Fly the drone up to around 20 m AGL (visually assess if there will be trees or terrain in the way at any time before flying the route).
    c. Tilt the camera until it is at a 45 degree angle, and spin the UAV using the yaw control so the center of the scene is being captured.
    d. Start the video recording.

e. Begin flying the UAV sideways in a curved path (doesn't need to be a perfect circle if there is some small area that sticks out that you want to capture) at a moderate and safe speed given your location and wind conditions.
f. Follow the UAV as you fly it, ensuring through the viewer on your phone that the center of the scene is mostly in view and your desired scene is being accurately captured.
g. Once you have returned to where you started, depending on battery life, land the drone, or prepare its position and camera angle for the next pass.

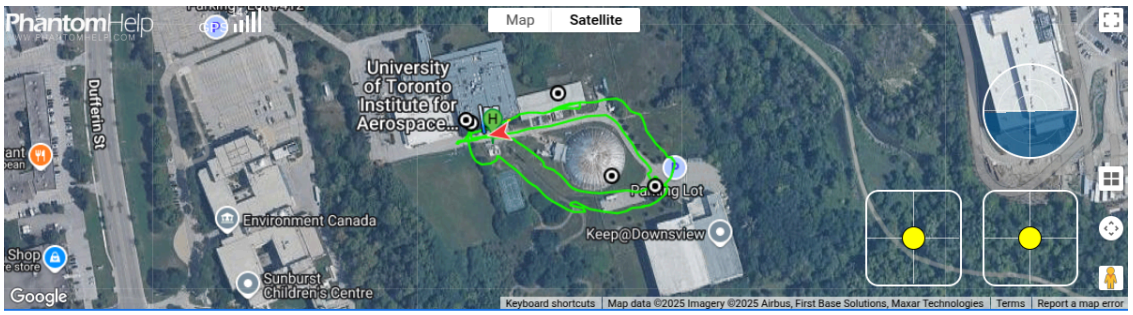
9. The next pass to fly is the 10 m forward, nadir facing pass.
    a. This pass uses the same 'center' of your scene and flies around it in a direction of travel that roughly follows what you want the UGV to follow.
    b. Fly the drone up to around 10 m AGL (visually assess if there will be trees or terrain in the way at any time before flying the route - make sure you're ready to adjust your course to give at least 1 m clearance for these obstacles).
    c. Tilt the camera until it is facing straight down, and spin the UAV using the yaw control so it is facing forward, ready to travel the path the UGV will follow.
    d. Start the video recording.
    e. Begin flying the UAV along your desired path around the center at a moderate and safe speed given your location and wind conditions.
    f. Follow the UAV as you fly it, ensuring through the viewer on your phone that the scene is being smoothly captured.
    g. Once you have returned to where you started, depending on battery life, land the drone, or prepare its position and camera angle for the next pass.

10. The final pass to fly is the 5-8 m forward, POV facing pass.
    a. This pass uses the same ‘center’ of your scene as well, but there is more flexibility in this pass to fly in/out/and around interesting features in the scene to capture it if need be (over hands, large pillars, narrow entrances etc). This pass also flies in the direction of travel that roughly follows what you want the UGV to follow.
    b. Fly the drone up to around 5-8 m AGL (visually assess if there will be trees or terrain in the way at any time before flying the route - make sure you’re ready to adjust your course to give at least 1 m clearance for these obstacles).
    c. Tilt the camera until it is facing straight forward, and spin the UAV using the yaw control so it is facing forward, ready to travel the path the UGV will follow.
    d. Start the video recording.
    e. Begin flying the UAV along your desired path around the center at a moderate and safe speed given your location and wind conditions.
    f. Follow the UAV as you fly it, ensuring through the viewer on your phone that the scene is being smoothly captured.
    g. Once you have returned to where you started, land the drone and turn it and the controller off.



11. Return inside to your computer with the SD card after packing away the drone by reversing the steps you took to unpack it. Please charge any dead batteries.
12. Connect your phone to your laptop via USB and open the DJI Go 4 folder in your phone’s file system.
13. Find a folder called FlightLogs, copy the file generated for the date and time you just flew (or any other date/time if you’re revisiting this process).
14. Move the file to your computer into whatever file system you have arranged for this project.
15. Open this website: https://www.phantomhelp.com/logviewer/upload/
    a. Upload your flight log TXT file by doing the captcha and browsing for the file, it may take a minute to load depending on the size
    b. Ensure the flight trajectory looks right based on your expectation of what to find in this file.
    c. Click the export CSV button and wait for the download to finish.
    d. Extract the downloaded file so you have a TXT file you can open.
    e. Open and read the TXT file.

f. Go to column BN and make sure there are three instances of Camera.isVideo switching from False to True (should match the number of videos you took).
g. Close the file.




Download KML | Download CSV | Download Flight Log

Map Size: **Small** | Medium | Large

►1x ■

| Time | Flight Mode | GPS | IMU Altitude | VPS Altitude | Speed | Home Distance | Battery | Battery Voltage | Cell 1 | Cell 2 | Cell 3 | Cell 4 | Cell Deviation | Message |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0m 0.4s | Starting Motors | 14 satellites | 0 ft | 0.3 ft | 0 mph | 25.8 ft | 77% | 16.034 V | 4.01 V | 4.01 V | 4.01 V | 4.01 V | 0.008 V | Data R |
| 0m 1.6s | Auto Takeoff | 14 satellites | 0 ft | 0.3 ft | 0 mph | 26.0 ft | 77% | 15.971 V | 3.98 V | 3.99 V | 4.00 V | 4.00 V | 0.016 V | Flight |
| 0m 2.4s | Auto Takeoff | 14 satellites | 0 ft | 0.3 ft | 0.2 mph | 0 ft | 77% | 15.526 V | 3.89 V | 3.89 V | 3.89 V | 3.87 V | 0.023 V | Home |
| 0m 3.4s | Auto Takeoff | 13 satellites | 1.0 ft | 2.0 ft | 0.4 mph | 1.1 ft | 77% | 15.526 V | 3.89 V | 3.89 V | 3.89 V | 3.87 V | 0.023 V | |

16. Connect and open the SD card's file system on your computer.
17. Move the video files you took to wherever you have your file system for this project set up.
18. In a new terminal, use exiftool to find the metadata timestamps for the start of the videos you took like this:

    ***EXAMPLE:***
    exiftool /home/desiree/ASRL/Thesis/BuddySystemDatasets/feb19Dome/trainingVids/DJI_0016.MP4

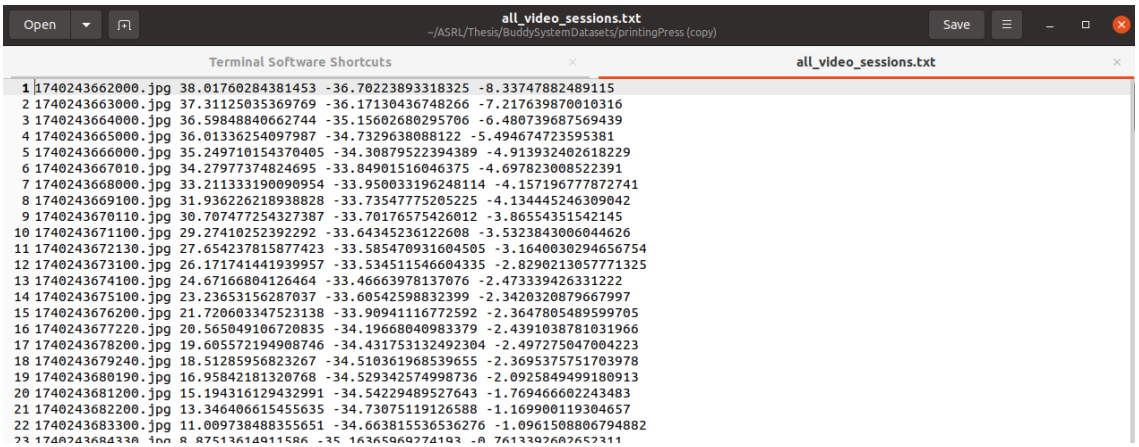
19. Look for "Create Date", record that time, and then use an online epoch time converter to obtain the time in epoch time.

```
desiree@Desiree-ThinkPad-P53:~$ exiftool /home/desiree/ASRL/Thesis/BuddySystemDatasets/DomeOrbit2/Trai
ningVids/DJI_0014.MP4
ExifTool Version Number         : 11.88
File Name                       : DJI_0014.MP4
Directory                       : /home/desiree/ASRL/Thesis/BuddySystemDatasets/DomeOrbit2/TrainingVid
s
File Size                       : 1060 MB
File Modification Date/Time     : 2024:10:31 12:29:18-04:00
File Access Date/Time           : 2025:03:10 21:29:39-04:00
File Inode Change Date/Time     : 2024:11:28 14:46:51-05:00
File Permissions                : rw-r--r--
File Type                       : MP4
File Type Extension             : mp4
MIME Type                       : video/mp4
Major Brand                     : MP4 Base w/ AVC ext [ISO 14496-12:2005]
Minor Version                   : 2014.2.0
Compatible Brands               : avc1, isom
Media Data Size                 : 1111473683
Media Data Offset               : 40
Movie Header Version            : 0
Create Date                     : 2024:10:31 16:25:25
Modify Date                     : 2024:10:31 16:25:25
Time Scale                      : 30000
Duration                        : 0:03:42
Preferred Rate                  : 1
Preferred Volume                : 100.00%
```

20. Repeat steps 17 and 18 for every video you recorded so you wind up with the following information:

    ***EXAMPLE:***
    Dome w/ spray paint - (num vids matches num true segments)
    DJI_0001: POV, (12.29.31) (12:30:33 = 1738949433000)
    DJI_0002: Straight down (12.36.54) (12:37:42 = 1738949862000)
    DJI_0003: Orbit (12.45.06) (12:45:40= 1738950340000)

21. Now, open the vtr_virtual_teach package within the vtr3 project you cloned from the following repo: ***Not Online Yet***
22. Open the script: GPSRegistration4Colmap.py
    a. Replace input_file and video_paths with your paths to the FlightLog and videos.
    b. Put the epoch times in start_epoch_times for eac video. NOTE: if multiple sessions of UAV flight were required (ie a battery died and the UAV had to be shut on and off again, there will be multiple FlightLogs, only put videos and timestamps that go with the FlightLog you are currently using together).
    c. Update the output_dir to direct the images and files made to somewhere in your project file system.
    d. Run the script, check the terminal output to make sure it checks out with the data you expect to find.
23. Now you have a folder of images and a TXT file that has image names and their GPS location on each line (called all_video_sessions.txt).

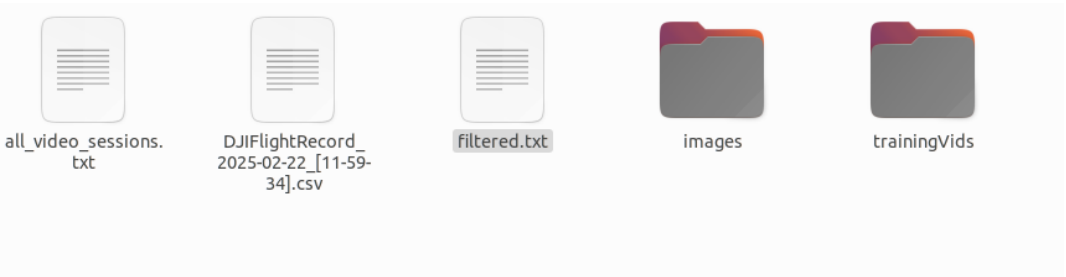
24. Now open the folder with all the images, and go through them all to delete blurry ones, duplicates, or instances of prolonged standstill footage that may confuse colmap's SfM. These images may come from repeated parts of the trajectory due to imprecise flying like in this small section:

25. Then run imageFilterandCoordScaler.py by using the following command in the terminal replaced with your specific paths (where you can determine the scale factor but it should be between 1/60 and 1/100:

    ***EXAMPLE:***

    python3 imageFilterandCoordScaler.py --image_folder /home/desiree/ASRL/Thesis/BuddySystemDatasets/Feb7thFlights/Grassy/colmap/images --txt_file /home/desiree/ASRL/Thesis/BuddySystemDatasets/Feb7thFlights/Grassy/all_video_sessions.txt --output_file /home/desiree/ASRL/Thesis/BuddySystemDatasets/Feb7thFlights/Grassy/colmap/filtered.txt --scale 0.01

26. Now you have the images and TXT file you will need for Colmap in the next phase.



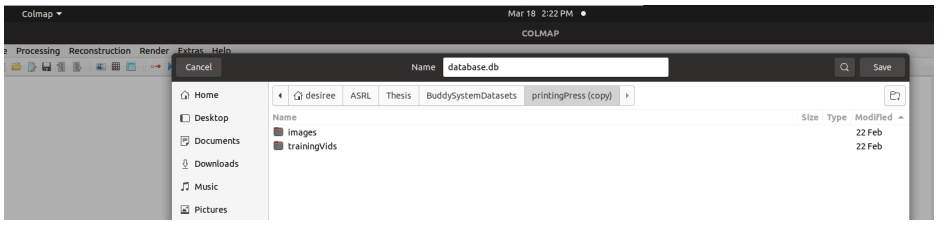
# Phase 2: NeRF Training

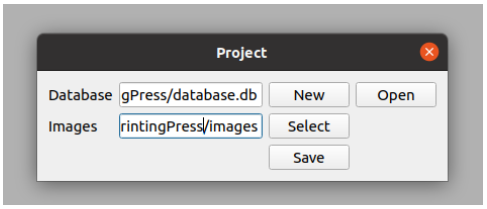
The following steps detail doing the scene reconstruction and scaling with Colmap and the NeRF training with Nerfstudio. If you have set up VirT&R with the docker image made for the project, all commands should be run in a terminal inside this docker image to work.

1. Type the following into the terminal to open Colmap:

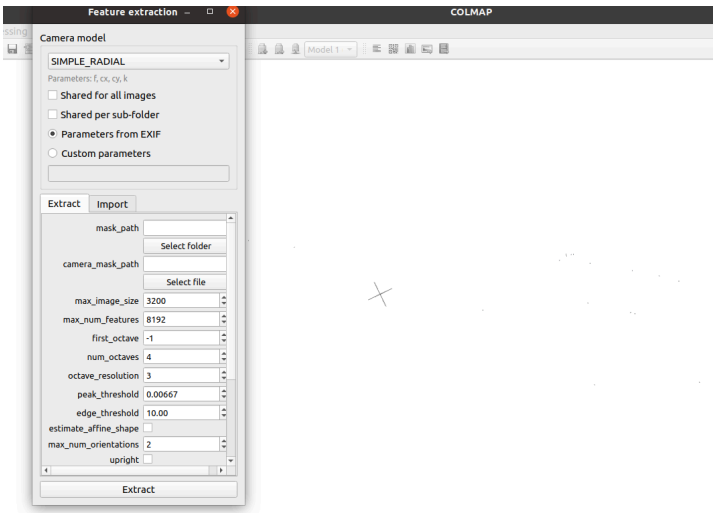
   ***EXAMPLE:***
   colmap gui

2. Go to File, and make a new Database called database.db wherever you want in your project file system (this should be in the directory that has the folder of images from the previous phase's last step). Then give the path to that folder of images where it asks, and save the project as seen here:

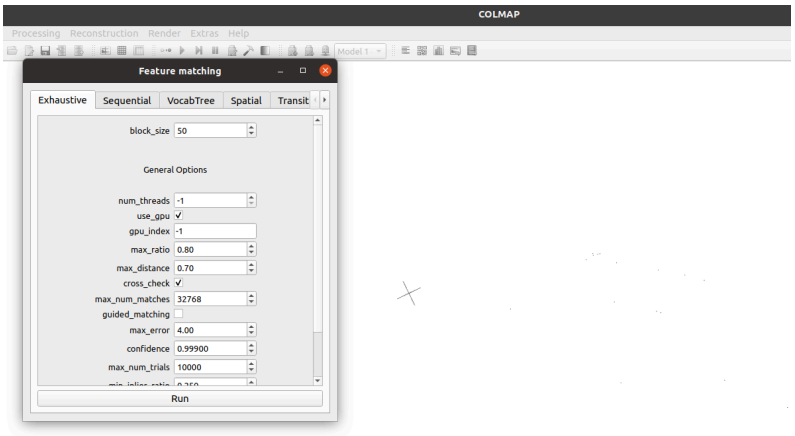


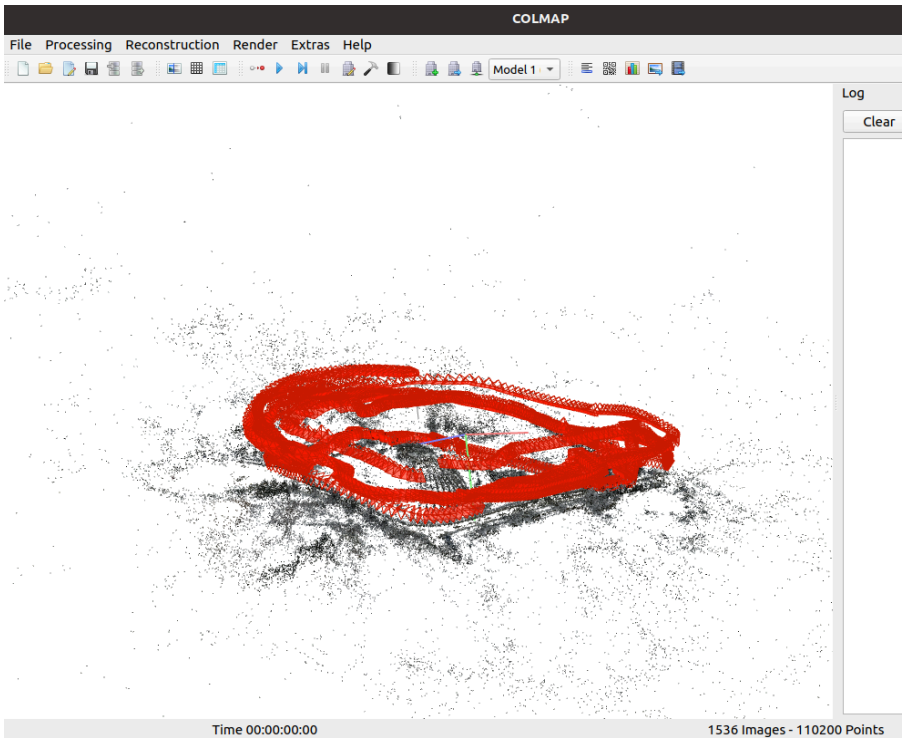

3. Go to Processing>Feature Extraction, and don't change any options, then click Extract (this should take a minute or so depending on the images).



4. Then go to Processing>Feature Matching, and make sure you run an exhaustive matching, then click run (this should take 5-10 mins depending on images).




5. Then go to Reconstruction>Start Reconstruction (this should take 20-30 mins depending on images).
6. Wait for the reconstruction to finish, you should see along the bottom bar, that the time has stopped elapsing and the number of images matches the number of images in the folder you provided as input.




**(the log is empty and the time is 00:00:00 here because I reopened this project to take this picture, for you it will reflect the total time taken for the command and the log will show the reconstruction components)**

7. Make sure to save and export this model by going to File>Export Model.
8. Close Colmap.

9. Next, you need to scale down this reconstruction by doing the following in the terminal:

   ***EXAMPLE:***

```
colmap model_aligner \
    --input_path
/home/desiree/ASRL/Thesis/BuddySystemDatasets/feb19Dome/colmap/reconstructionAttempt1 \
    --output_path
/home/desiree/ASRL/Thesis/BuddySystemDatasets/feb19Dome/colmap/reconstructionAttempt1/1_60Sc
aled \
    --ref_images_path
/home/desiree/ASRL/Thesis/BuddySystemDatasets/feb19Dome/colmap/reconstructionAttempt1/filtered.
txt \
    --ref_is_gps 0 \
    --alignment_max_error 1.0
```

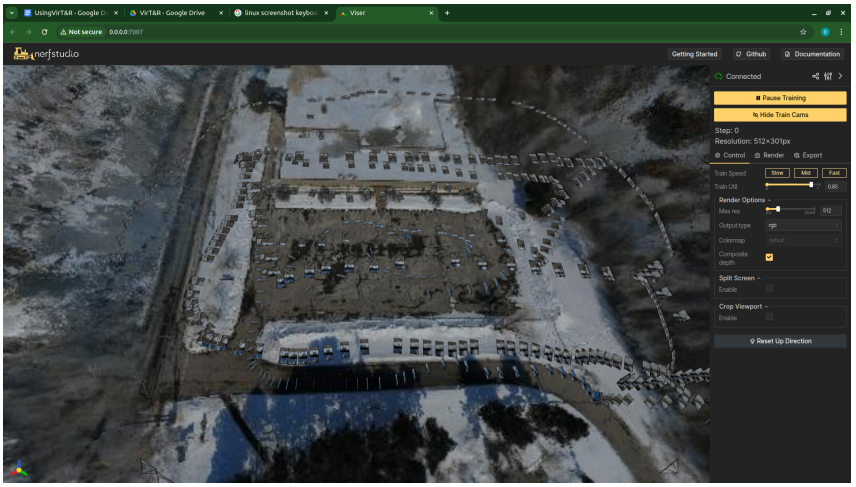
10. Now you have three .bin files that need to be transferred to the folder for this dataset in the nerfstudio codebase data folder (nerfstudio>data>nerfstudio>{project_name}).
    a. Put these three files in a folder called colmap>1_100Scaled along with the folder of images you gave to colmap.


11. Use the following command to train a NeRF of the scene:

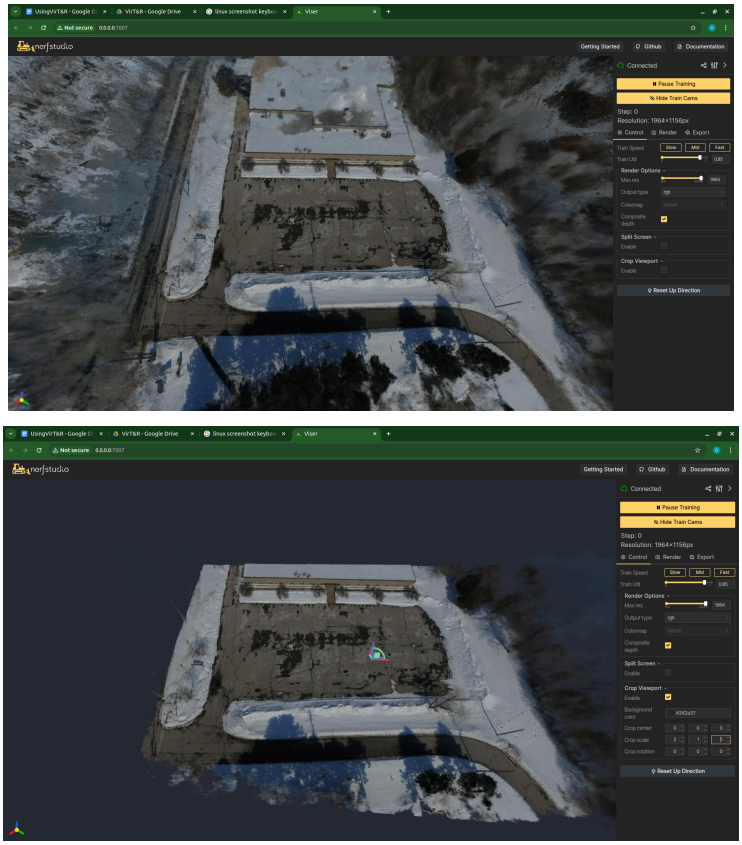
    ***EXAMPLE:***

```
ns-train nerfacto --vis viewer --data data/nerfstudio/feb19Dome --viewer.quit-on-train-completion False
colmap --colmap_path colmap/1_60Scaled --downscale_factor=1 --orientation_method=none
--center_method=none --assume_colmap_world_coordinate_convention=False
```

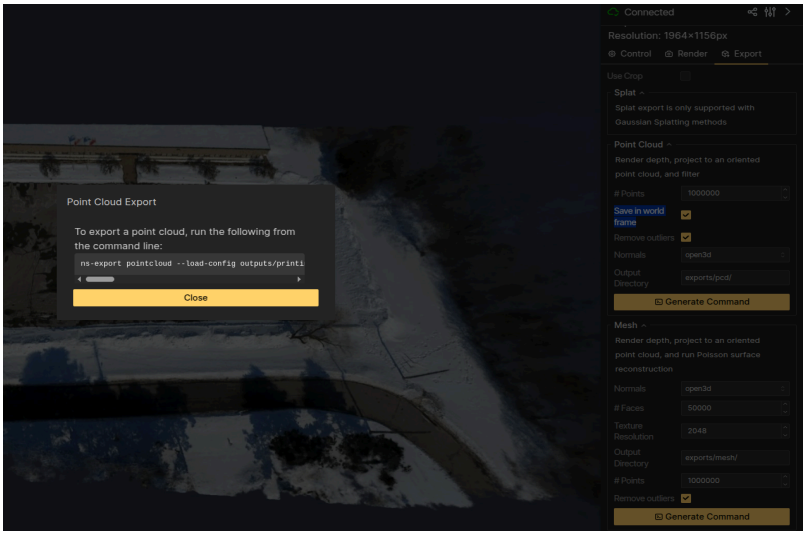
12. Once the NeRF is done training, ctrl+click the HTTP link to open the viewer from the terminal.



13. Examine the NeRF has trained to a decent quality. (Use the resolution slider bar). Crop the scene so only the area the UGV will be driving in is saved using the crop box option on the side panel.
    a. Try to get rid of blurry artifacts in the sky and surroundings by doing this.




14. Go to the exports side panel and generate the commands for exporting the point cloud and mesh, (adjust the number of points and faces in the point cloud and mesh respectively to be an order of magnitude higher than the default - MAKE SURE TO SAVE POINT CLOUD IN WORLD FRAME):



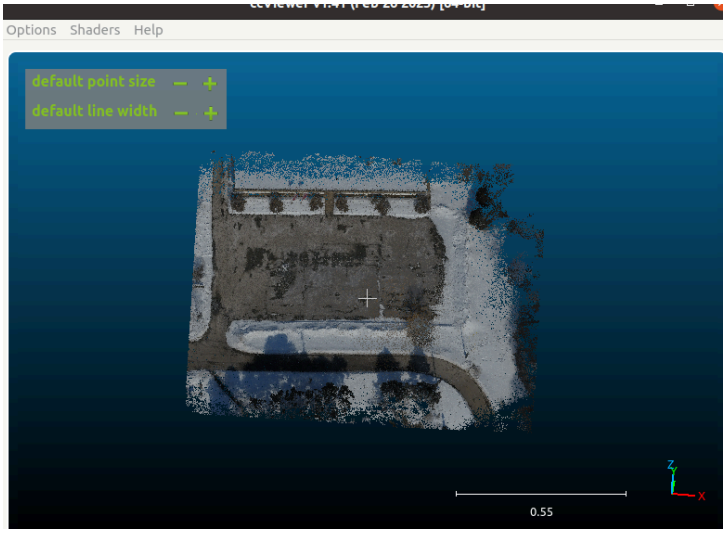
15. Close the viewer and quit training the scene, then in that terminal enter the generated commands:

***EXAMPLE:***

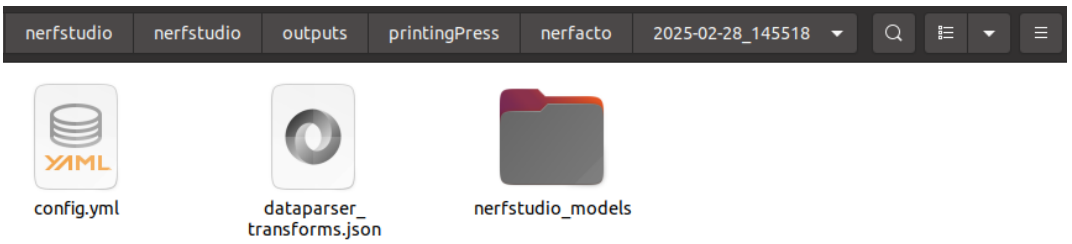
ns-export pointcloud --load-config outputs/feb19Dome/nerfacto/2025-02-19_171352/config.yml --output-dir exports/pcd/ --num-points 1000000 --remove-outliers True --normal-method open3d --save-world-frame True --obb_center 0.3997362713 0.1145217735 -0.0983033677 --obb_rotation 0.8704366783 -0.0422337007 0.2335225570 --obb_scale 1.2000000477 1.0000000000 0.4000000060

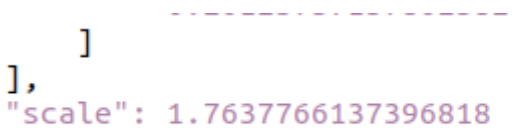
ns-export poisson --load-config outputs/feb19Dome/nerfacto/2025-02-19_171352/config.yml --output-dir exports/mesh/ --target-num-faces 50000 --num-pixels-per-side 4096 --num-points 1000000 --remove-outliers True --normal-method open3d --obb_center 0.3997362713 0.1145217735 -0.0983033677 --obb_rotation 0.8704366783 -0.0422337007 0.2335225570 --obb_scale 1.2000000477 1.0000000000 0.4000000060

16. Now, return to the terminal and run these two commands, they may take up to 15 minutes depending on the scene size.




17. Go to nerfstudio>outputs>{project_name}>{date_time}>data_parser.json and look for “scale: 0.34193143508020946” at the bottom (the number will be different, but record it for later).

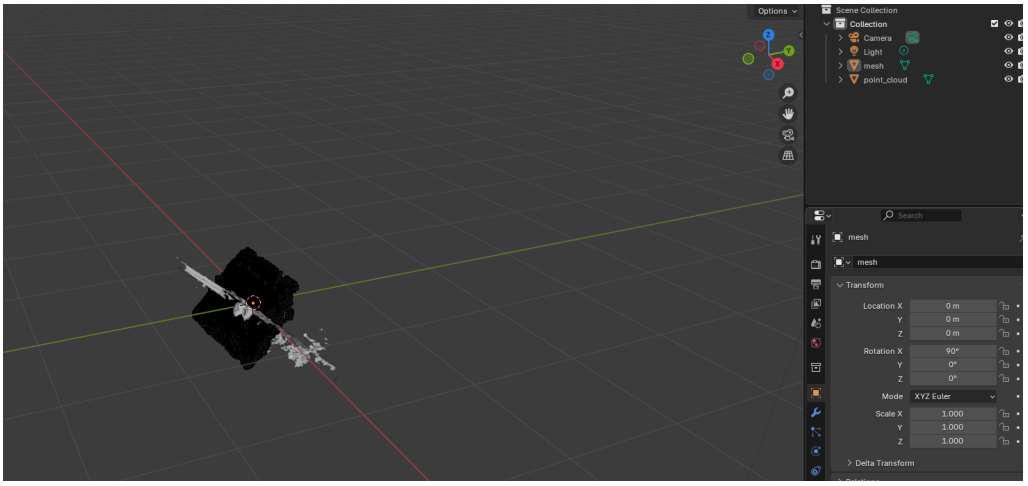


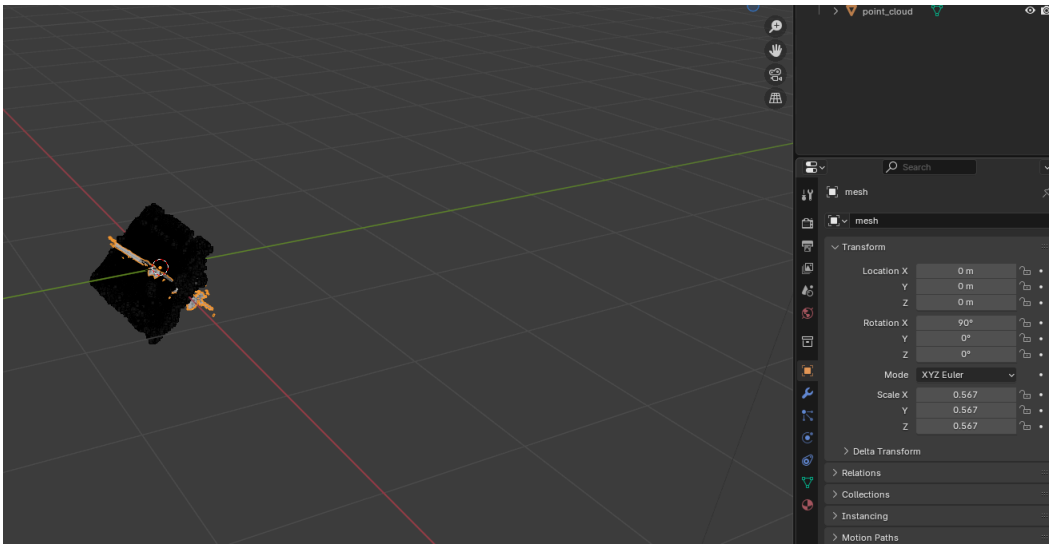
# Phase 3: PointCloud and Path Association

This section can be further segmented into actions to prepare and use the NeRF Mesh and actions to prepare and use the NeRF Point cloud. Both parts involve using Blender, but the Mesh process involves gazebo and some custom scripts, whereas the PointCloud is used in some custom scripts and then given right to VirT&R.

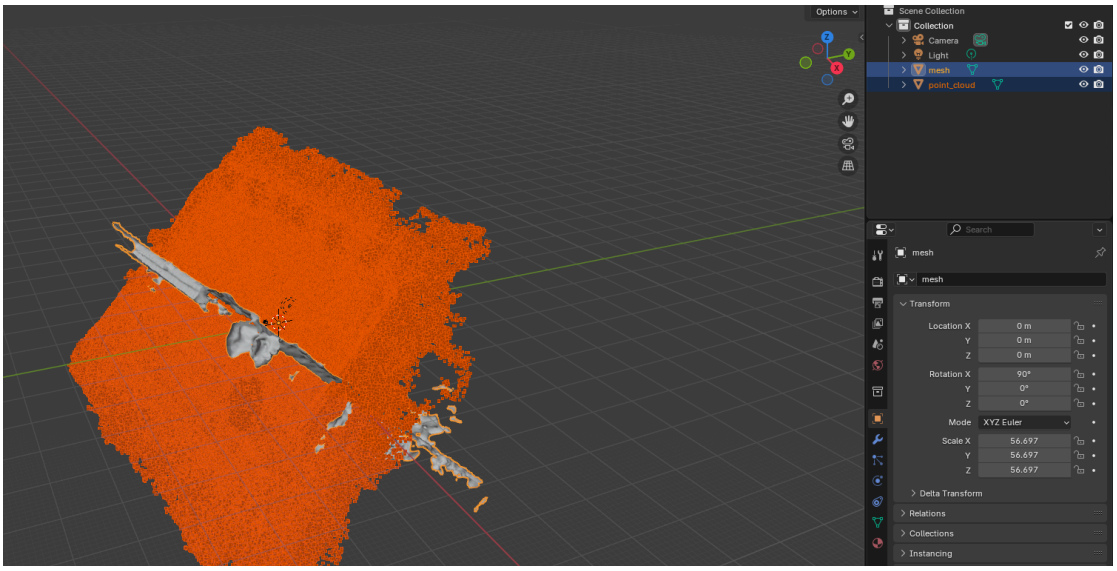
1. Now that the mesh has been generated, it needs to be taken from nerfstudio>exports>Mesh>{project_name} to the folder used for the project in your file system (make sure to grab all 4 files produced).
2. Also move the point_cloud.ply file from nerfstudio>exports>PointCloud>{project_name} to somewhere in your project file system.
3. Open blender to a blank project and import the .obj file and .ply file.



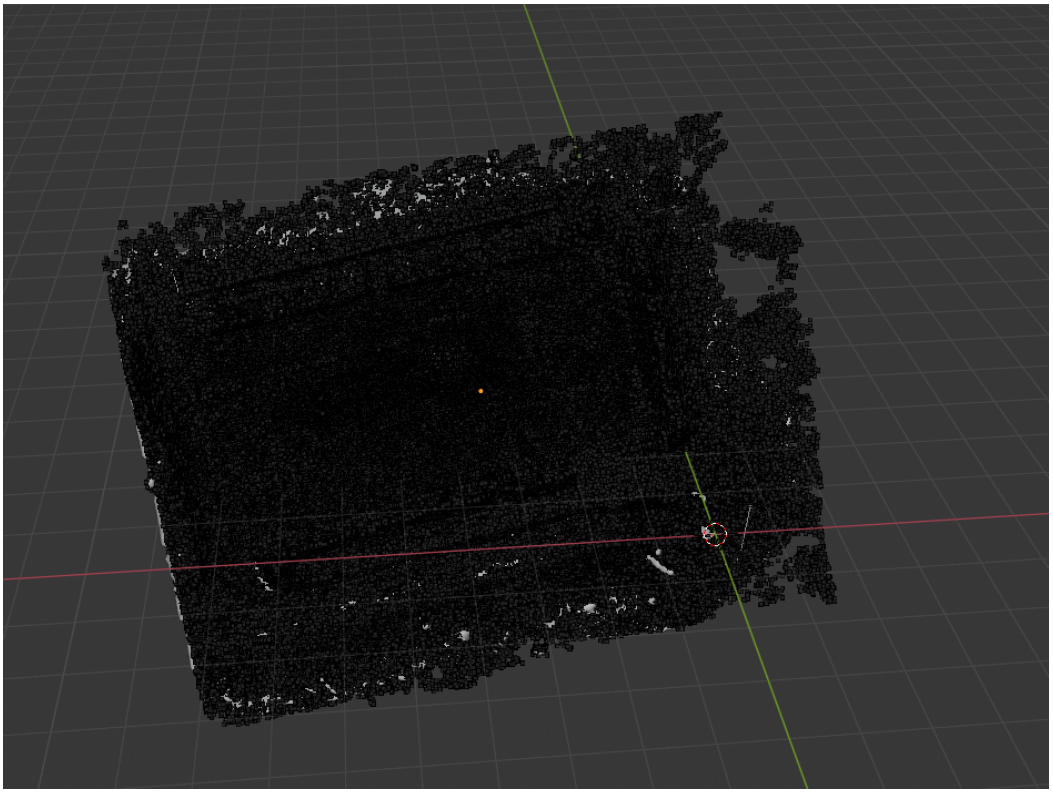

4. Scale up the mesh by the previously saved number from the .json by using the x,y,z scale toggles on the right side panel.



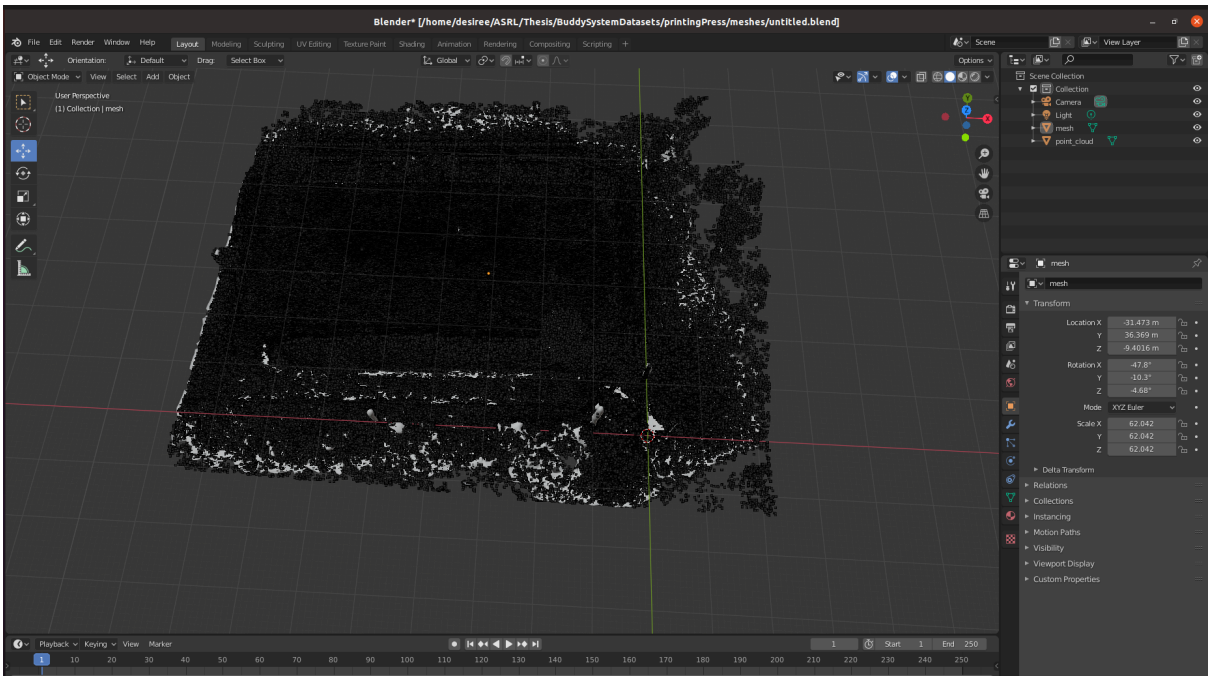
5. Then shift+click on both the mesh and point cloud, press S, and type in the number intentionally used for downscaling with Colmap.



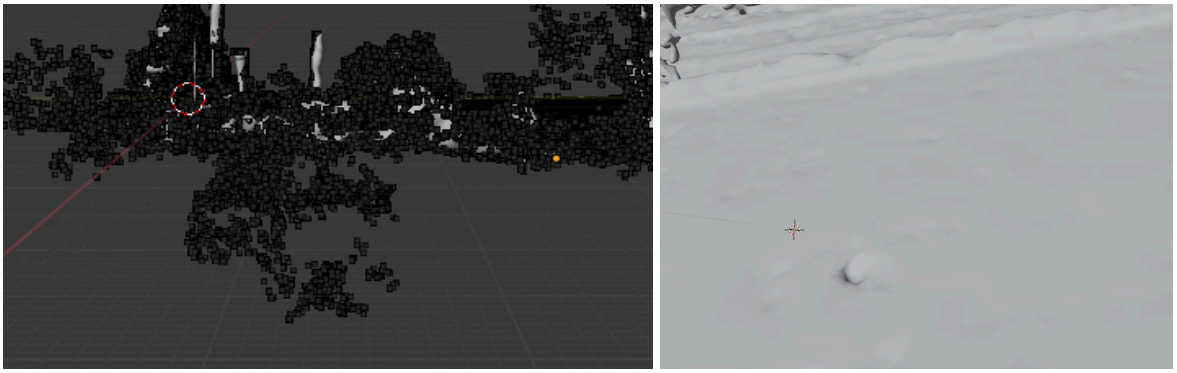

6. The mesh usually loads in 90 degrees rotated from the point cloud, so this can be undone as well. Now the mesh and point cloud should be perfectly aligned.



7. Rotate them both together so the blender origin is where the gazebo world origin should be (where the UGV will spawn) slightly above the ground.



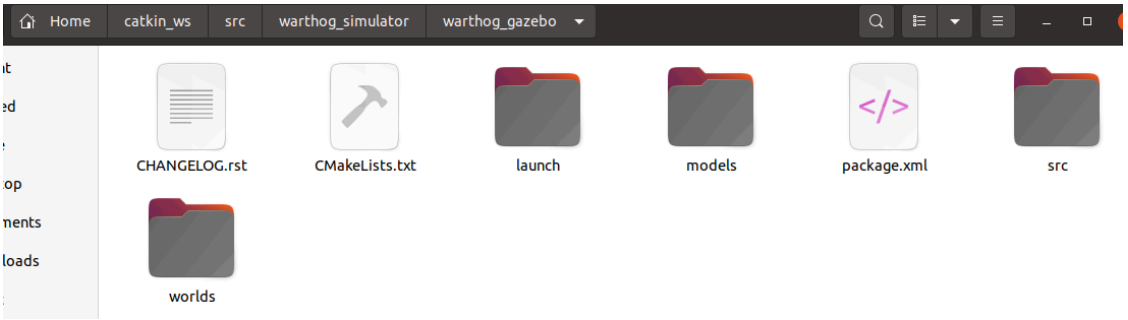

8. Crop and clean both the mesh and the point cloud to remove odd artifacts.
    a. You may need to smooth the mesh out a bit by going to edit mode, selecting vertices, right clicking, and using the smooth feature, don't do this too much as it blurs the ground.

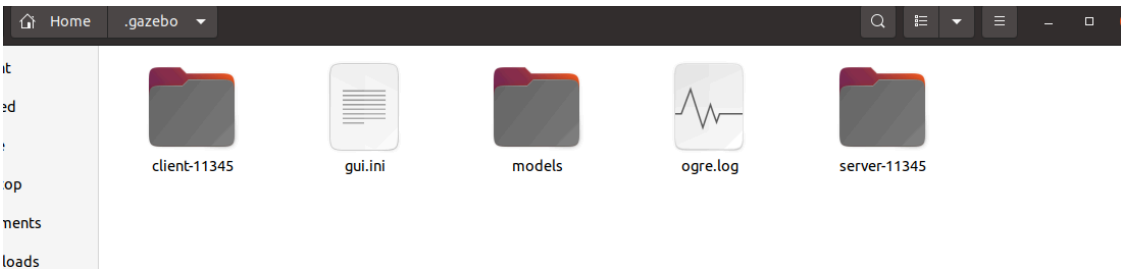


9. Save the mesh as a .dae file, and save the point cloud and a .ply file using the File>Export function in the top left corner.

# Process for Gazebo Sim:

1. The .dae mesh and associated material_0.png need to be moved to the warthog_simulator>warthog_gazebo>models (located in catkin_ws workspace) and .gazebo>models folders (located in the home directory).




2. Then, in the warthog_simulator>warthog_gazebo folder, .launch and .world files need to be created in xml script for the desired simulation that specifies the mesh that will be included. Examples can be found in the google drive catkin_ws>src>warthog_simulatior>warthog_gazebo folder.

Open | press_mesh.world ~/catkin_ws/src/warthog_simulator/warthog_gazebo/worlds | Save

Terminal Software Shortcuts | press_mesh.launch | press_mesh.world

```
<sdf version='1.6'>
  <world name='press_mesh'>
    <!-- Omni Light for Full Uniform Lighting -->
    <light name="full_light" type="point">
      <pose frame="">0 0 5 0 0 0</pose> <!-- Positioned above the scene -->
      <diffuse>1.0 1.0 1.0 1</diffuse> <!-- White light -->
      <specular>1.0 1.0 1.0 1</specular> <!-- White highlights -->
      <attenuation>
        <range>1000</range>
        <constant>1.0</constant>  <!-- No attenuation -->
        <linear>0.0</linear>
        <quadratic>0.0</quadratic>
      </attenuation>
      <cast_shadows>0</cast_shadows> <!-- No shadows -->
    </light>

    <model name='press'>
      <static>1</static>
      <link name='press'>
        <collision name='collision'>
          <geometry>
            <mesh>
              <uri>model://press_mesh/meshes/mesh.dae</uri>
              <size>1 1</size>
            </mesh>
          </geometry>
        </collision>
        <visual name='visual'>
          <cast_shadows>0</cast_shadows> <!-- No shadows for visual -->
          <geometry>
            <mesh>
              <uri>model://press_mesh/meshes/mesh.dae</uri>
              <size>1 1</size>
            </mesh>
          </geometry>
        </visual>
        <velocity_decay>
          <linear>0</linear>
          <angular>0</angular>
        </velocity_decay>
        <self_collide>0</self_collide>
        <kinematic>0</kinematic>
        <gravity>1</gravity>
      </link>
    </model>
```

**(full file examples can be found in the repo!)**

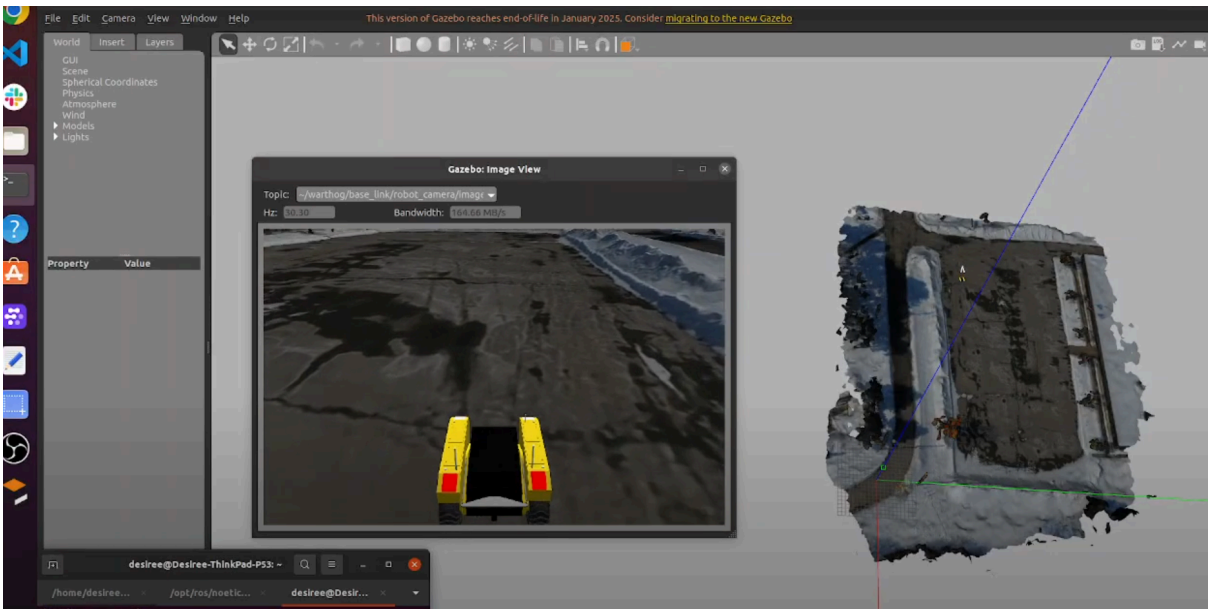
3. Now Gazebo can be opened by typing the following command in a terminal window:

   ***EXAMPLE:***
   roslaunch warthog_gazebo {project_name}_mesh_world.launch

4. Then in new terminal windows, use the following command to set up keyboard or joystick control:

   ***EXAMPLE:***
   rosrun teleop_twist_keyboard teleop_twist_keyboard.py OR roslaunch teleop_twist_joy teleop.launch

5. To open the second window of the warthog's POV camera, access and open the window by going to the top left and finding it here: Window>Topic Visualization> /warthog/base_link/robot_camera/image




6. Lastly, steer the warthog to the place you want to begin the path using the joystick and the run the following command to record the path:

   ***EXAMPLE:***
   rosrun warthog_gazebo_path_publisher save_path.py

7. The path saved in a txt file from this will be given to the virT&R package later.

# Process for Associating Point Cloud Submaps:

1. To begin to use vtr_virtual_teach ensure the following commands have been run:

   ***EXAMPLE:***
   source ${VTRSRC}/main/install/setup.bash
   source /opt/ros/humble/setup.bash
   cd home/desiree/ASRL/vtr3/src/main
   echo $ROS_DISTRO

2. Then replace the file paths in the following locations with paths to where you have the final point cloud, mesh path, and output directory:
   a. auto cloud = loadPointCloud("/home/desiree/ASRL/vtr3/data/grassy/point_cloud.pcd");
   b. std::string odometry_csv_path = "/home/desiree/ASRL/vtr3/data/grassy/nerf_gazebo_relative_transforms.csv";
   c. std::string graph_path = "/home/desiree/ASRL/vtr3/data/grassy/graph";
3. Rebuild and then run the project (make sure you're in the 'main' folder when you do this):

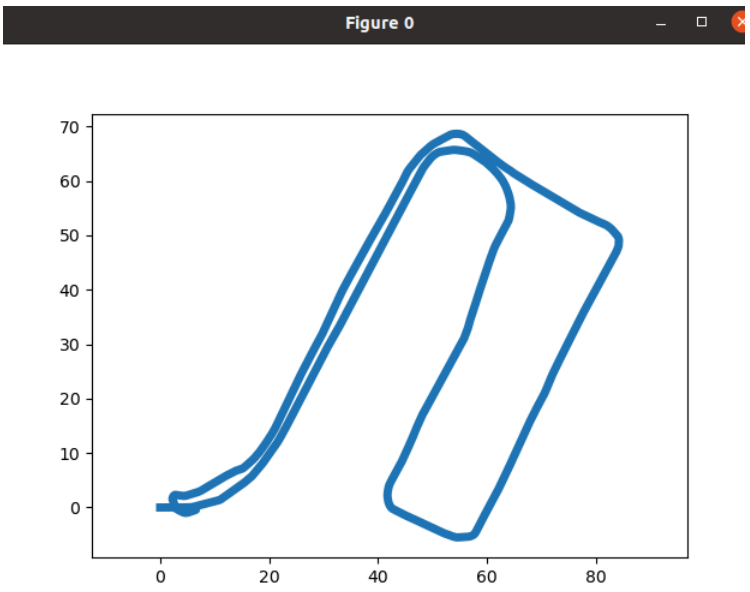
   ***EXAMPLE:***
   colcon build --packages-select vtr_virtual_teach

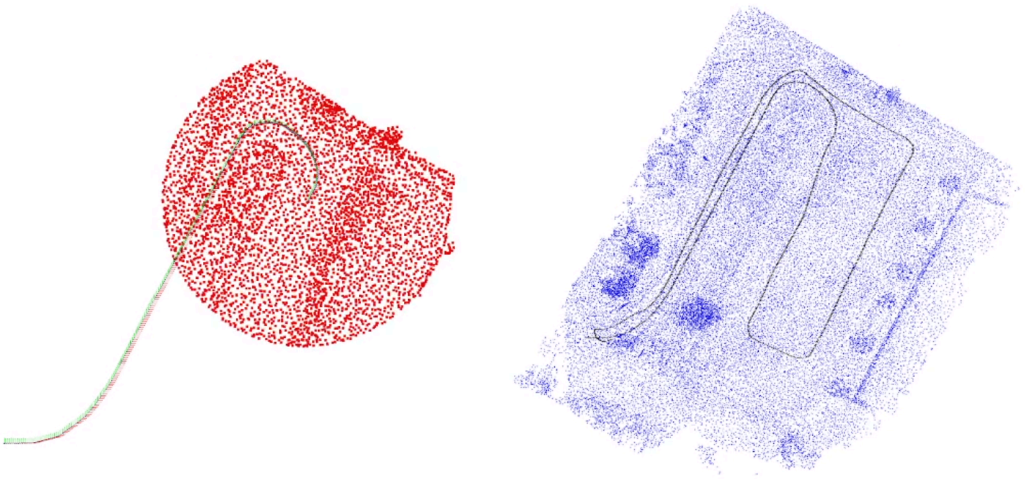
   ros2 run vtr_virtual_teach generate_global_map

4. Now a pose graph has been generated for the path driven through the mesh. Before using it online with LT&R it should be checked using the python tools to ensure the route is as expected.
5. To use the vtr3_python_tools, you must be in the vtr3>vtr3_posegraph_tools>vtr3_pose_graph directory and type the following commands to show what the teach path looks like and then to plot the submaps along the teach path (red point clouds are at the vertex, blue point clouds are being pointed to from the vertex):

   ***EXAMPLE:***
   python3 samples/plot_teach_path.py -g /home/desiree/ASRL/vtr3/data/ExistingGraph/pose_graph

   python3 src/vtr_odom_extractor/plot_submaps.py -g /home/desiree/ASRL/vtr3/data/nerf_with_odom_path/graph



Once the virtually-generated pose graph has been made using the path driven on the mesh from gazebo and the point cloud from the NeRF model, and it has been verified to follow the expected route with the pose graph tools, it can be loaded into the vtr3>temp>{project_name} folder to be used online on the Warthog with LT&R just as normal pose graphs are. This is the part of the pipeline that remains unchanged as it is only the process for making the teach map that is altered here.