Matterport3D Environment Setup Notes

Setup Repository

First follow our website to setup the '*autonomous_exploration_development_environment*' repository in default settings. Then, checkout the *'distribution-matterport*' branch (replace '*distribution*' with '*melodic*' or '*noetic*' depending on the ROS version), and compile.

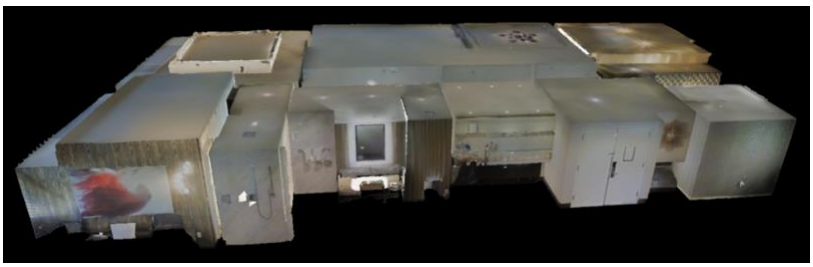
cd autonomous_exploration_development_environment
git checkout distribution-matterport && catkin_make

Prepare Environment Model

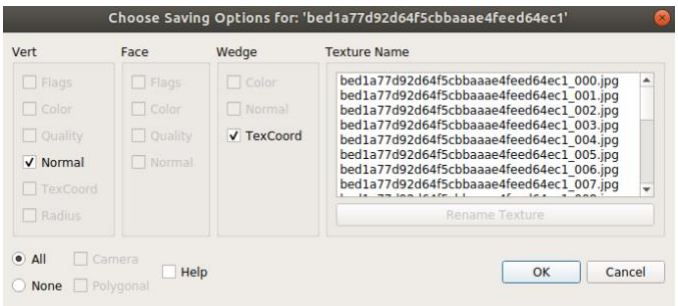
Install MeshLab (v2020.12 or later) for mesh format conversion.
MeshLab: https://www.meshlab.net

Go to Matterport3D website, sign Terms of Use, and download the environment models using the given download_mp.py script.
Matterport3D: https://niessner.github.io/Matterport

In each environment folder, you can find a matterport_mesh.zip file. Taking environment ID:17DRP5sb8fy (first downloaded mesh) as an example, unzip the matterport_mesh.zip file and load the .obj file in MeshLab, you will see a colored mesh.



Export the mesh in DAE format and save it as a matterport.dae file in the same folder with all the other mesh files, i.e. .mtl, .obj, .jpg. Then, copy all mesh files, i.e. .mtl, .obj, .jpg together with the matterport.dae file to the '*src/vehicle_simulator/mesh/matterport/meshes*' folder (do not put the files in another folder inside the '*meshes*' folder).



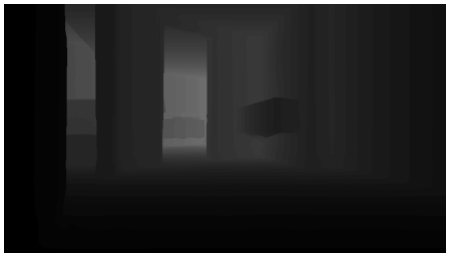
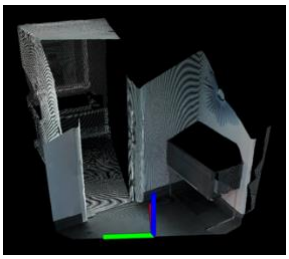
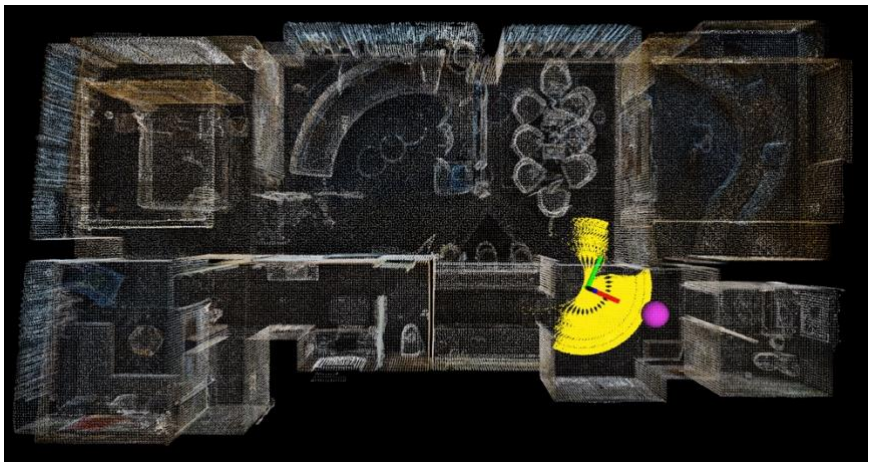
You can also find a house_segmentations.zip file. Extract the .house file, rename it to a matterport.house file, and copy it to the '*src/vehicle_simulator/mesh/matterport/segmentations*' folder. Extract the .ply file, rename it to a pointcloud.ply file, and copy it to the '*src/vehicle_simulator/mesh/matterport/preview*' folder.

Launch System

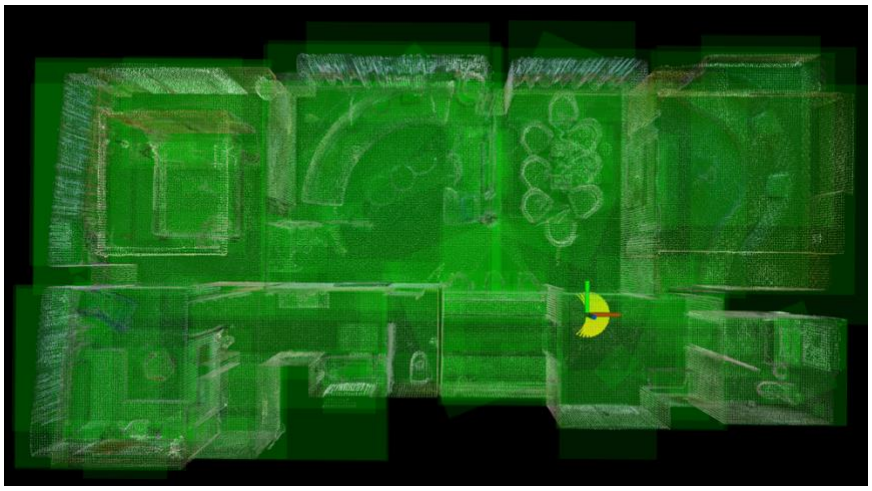
In a terminal, source the ROS workspace and launch the system.

source devel/setup.sh
roslaunch vehicle_simulator system_matterport.launch

Now, users can use the '*Waypoint*' button in RVIZ to drive the vehicle around. The rendered RGB images (left image), depth images (middle image), and registered scans should show in RVIZ. To visualize the point cloud messages (right image) corresponding to the depth images, click 'Panels->*Displays*' in RVIZ and check '*depthCloud*'. To view the overall map (bottom image), explored areas, and vehicle trajectory, check '*overallMap*', '*exploredAreas*', and '*trajectory*'. The camera configuration is set in the '*src/vehicle_simulator/urdf/rgbd_camera.urdf.xacro*' file.

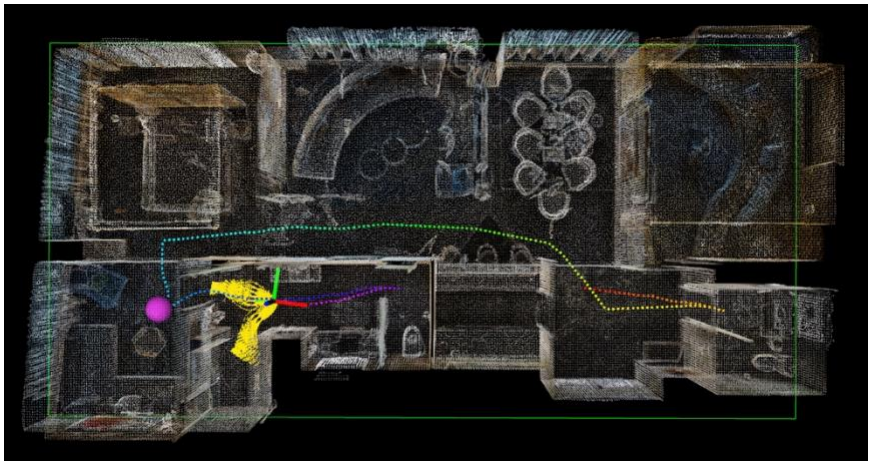





To view the region segmentations (top image below) and object segmentations (bottom image below), click '*Panels->Displays*' in RVIZ and check '*regionMarkers*' and '*objectMarkers*'. The segmentation data is also published as '*sensor_msgs*::*PointCloud2*' typed messages on ROS topics '*/region*_segmentations' and '*/o*bject_segmentations'. The point definitions are in the '*src/segmentation_proc/src/segmentationProc.cpp*' file. The region labels and object categories are retained from the Matterport3D original definitions.



In a second terminal, go to the '*autonomous_exploration_development_environment*' folder, source the ROS workspace, then use the command line below to run a waypoint example. The vehicle will follow the waypoints and drive inside the navigation boundary (waypoints and navigation boundary are made for environment ID:17DRP5sb8fy specifically).

roslaunch waypoint_example waypoint_example_matterport.launch



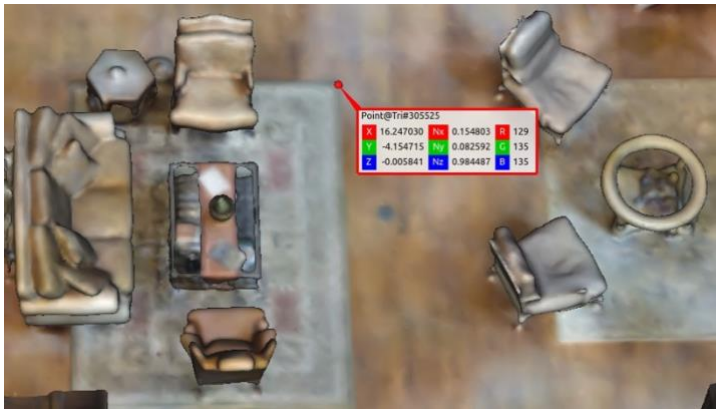
To view the mesh and vehicle in Gazebo GUI, set '*gazebo_gui = true*' in '*src/vehicle_simulator/launch/system_matterport.launch*'. To configure the vehicle to drive forward and backward, set '*twoWayDrive = true*' in '*src/local_planner/launch/local_planer.launch*'. To start the vehicle at a different position or orientation, set '*vehicleX*', '*vehicleY*', '*terrainZ*', '*vehicleYaw*' in '*src/vehicle_simulator/launch/system_matterport.launch*'.

Choose Start Position and Navigation Boundary

Install CloudCompare (v2.11.1 or later) for point cloud viewing and editing.
CloudCompare: https://www.danielgm.net/cc

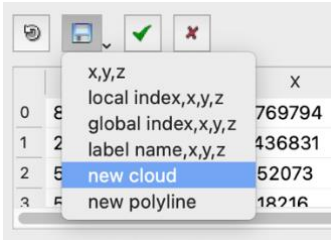
Matterport3D environment models often have multiple floors. We recommend using CloudCompare to choose the vehicle start position on the desired floor. Load the .ply file extracted from the house_segmentations.zip file. Then, you can use 'Tools->Segmentation ->Cross Section' to crop off the ceiling and unused floors to reveal the start area.

Use 'Tools/Point picking' to choose the vehicle start position. You can use these values to set '*vehicleX*', '*vehicleY*', '*terrainZ*' in '*src/vehicle_simulator/launch/system_matterport.launch*'. You can further save the cropped point cloud as a pointcloud.ply file and replace the one in the '*src/vehicle_simulator/mesh/matterport/preview*' folder.



To choose the navigation boundary, use 'Tools/Point list picking' to pick a list of points, save the points as 'new cloud' (top image), then a point cloud named 'Picking list' will show up in

CloudCompare. Use 'Edit->Multiply/Scale' and set 'Scale (z) = 0' to put the z-values to zero (bottom image), then save the point cloud in PLY format as a boundary_matterport.ply file (choose ASCII format). Now, users can copy the file to the 'src/waypoint_example/data' folder and run the waypoint example to test the navigation boundary.



Offline Processing with Habitat

Install Habitat for image rendering (first install Anaconda).
Habitat: https://github.com/facebookresearch/habitat-sim
Anaconda: https://www.anaconda.com

The recommended way of installing Habitat is using Conda packages, requiring python>=3.7 and cmake>=3.10. After installing Conda, use the command lines below to install Habitat.

conda create -n habitat python=3.8 cmake=3.14.0
conda activate habitat

For a computer with a display attached, use

conda install habitat-sim=0.2.1 -c conda-forge -c aihabitat

If you want to install an older version, e.g. v0.1.7, use

conda install habitat-sim=0.1.7 -c conda-forge -c aihabitat

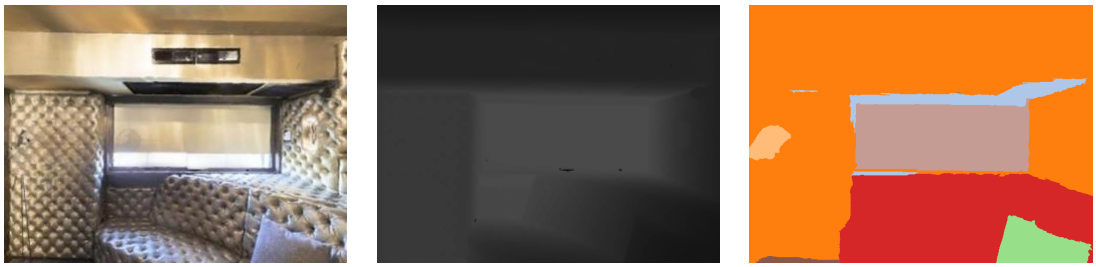
Then, use the script from Matterport3D to download the environment models with '*--task habitat*' flag, requiring Python 2.7. This will download a mp3d_habitat.zip file prepared for Habitat. Extract the files with the same environment ID: 17DRP5sb8fy used by the system.

python2 download_mp.py --task habitat -o data_download_dir

In a terminal, clone the stable branch of the habitat-sim repository and run the example.py script for a test. Users should see RGB images (left image), depth images (middle image), and

semantic images (right image) saved in the '*example*' folder. If the images are saved correctly, it means Habitat is installed correctly.

git clone --branch stable https://github.com/facebookresearch/habitat-sim.git
cd habitat-sim/examples
*python3 example.py --scene extracted_mp3d_habitat_dir/environment_id.glb *
--semantic_sensor --depth_sensor --save_png



Our system records a trajectory_timestamp.txt file in the '*src/vehicle_simulator/log*' folder for every run. The trajectory_timestamp.txt file is viewable with a text editor, where each line is a pose in the sequence of x (m), y (m), z (m), roll (rad), pitch (rad), yaw (rad), time from start (sec). Locate the trajectory_timestamp.txt file from the run that you want to post-process. In a terminal, go to the '*src/segmentation_proc/scripts*' folder and run a command line.

*python3 habitat_offline_v0.x.x.py --scene extracted_mp3d_habitat_dir/environment_id.glb *
--trajectory trajectory_file_dir/trajectory_timestamp.txt --save_dir image_saving_dir

This will render and save RGB images, depth images, and semantic images in the *'image_saving_dir*' folder as specified in the command line. The images are rendered for every pose in the trajectory_timestamp.txt file.

Note that the system needs to launch in native Ubuntu and the habitat_offline_v0.x.x.py script needs to run in the conda-habitat environment. Before launching the system, use the command line below to exit the conda-habitat environment.

conda deactivate

Before running the habitat_offline_v0.x.x.py script, use this command line to enter the conda-habitat environment.

conda activate habitat

Online Processing with Habitat (Ubuntu 20.04 only)

It is possible to run Habitat with the system in parallel on an Ubuntu 20.04 computer. Install ROS Noetic in the conda-habitat environment in addition to the nominal ROS Noetic installation in native Ubuntu.

conda activate habitat
conda install -c conda-forge -c robostack ros-noetic-desktop

In a terminal, launch the system. In a second terminal, go to the '*src/segmentation_proc/scripts*' folder and run the habitat_online_v0.x.x.py script (only available on the '*noetic-matterport*' branch).

conda activate habitat
python3 habitat_online_v0.x.x.py --scene extracted_mp3d_habitat_dir/environment_id.glb

Now, users can view the RGB images, depth images, and semantic images rendered by Habitat in RVIZ. Click 'Panels->*Displays*' and check '*HabitatRGBImage*', '*HabitatDepthImage*', and '*HabitatSemanticImage*'. The image formats are kept the same with the images saved in offline processing. In depth images, a pixel value multiplied by 10/255 is the distance (m).