Odin1 Relocalization Map Acquisition Guide

Odin1 provides relocalization functionality and supports data exchange with the MANIFOLD handheld scanner. Three methods are available:

1. 1. Use Odin to collect .olx data, process it in MindCloud, and export a .bin map file for Odin relocalization.
1. 2. Use the MANIFOLD handheld scanner (e.g. Q9000) to collect .lx data, process it in MindCloud, and export a .bin map file for Odin relocalization.
1. 3. Use Odin's SLAM mode to generate a .bin map file for relocalization. Detailed steps follow.

1. Record .olx File with Odin (Method 1)

1.1 Record .olx File with Odin (firmware >= 0.10.0, driver >= 0.9.0 required)

1.1.1 Modify /catkin_ws/src/odin_ros_driver/config/control_command.yaml: set recorddata: 1, custom_map_mode: 0 (Odometry mode):

```
Shell
recorddata: 1           # 0: off; 1: on
***
custom_map_mode: 0      # 0: Odometry mode 1: SLAM mode 2:
Relocalization mode
```

1.1.2 Run the driver

```
Shell
cd ~/catkin_ws
# ROS2 example; for ROS1 use corresponding commands
source install/setup.bash
ros2 launch odin_ros_driver odin1_ros2.launch.py
```

1.1.3 Collect the map

After collection, press Ctrl+C to stop the driver. The .olx file will be found under /catkin_ws/src/odin_ros_driver/recorddata/. Copy the entire folder to a PC with MindCloud (version >= 0.2.8) for Step 3.

2. Record .lx Data with Q9000 (Method 2)

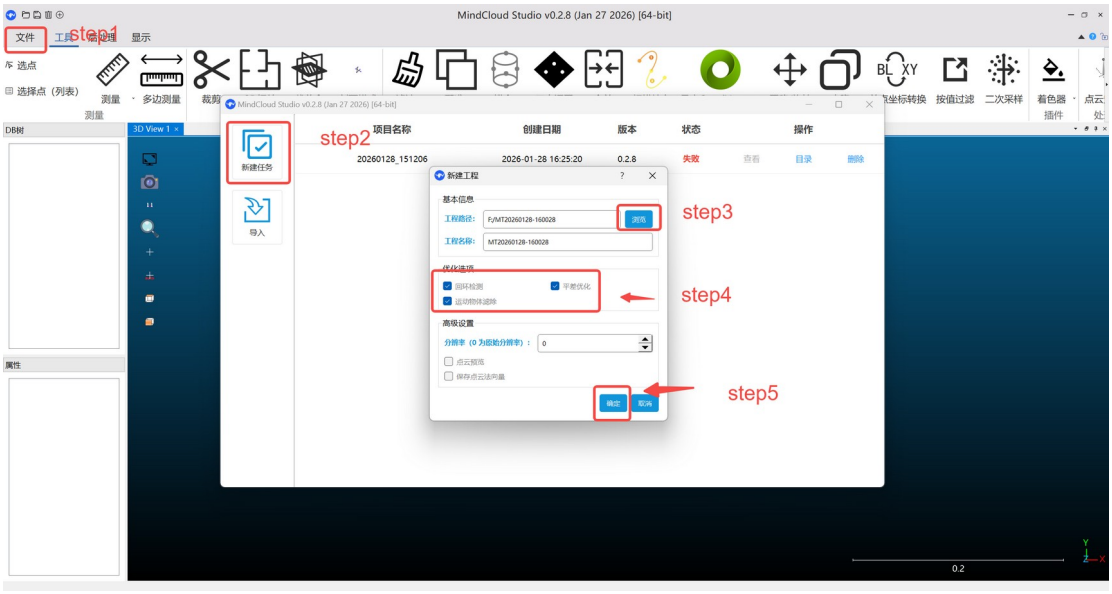
2.1 Record .lx File with Q9000

1. Collect data
1. Copy the .lx file to a PC with MindCloud (version >= 0.2.8) for Step 3.

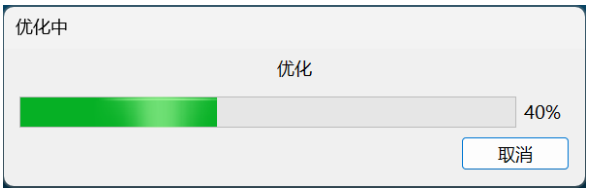
2.2 Import into MindCloud Studio and save PCD & BIN files (requires Windows)

3. Using MindCloud

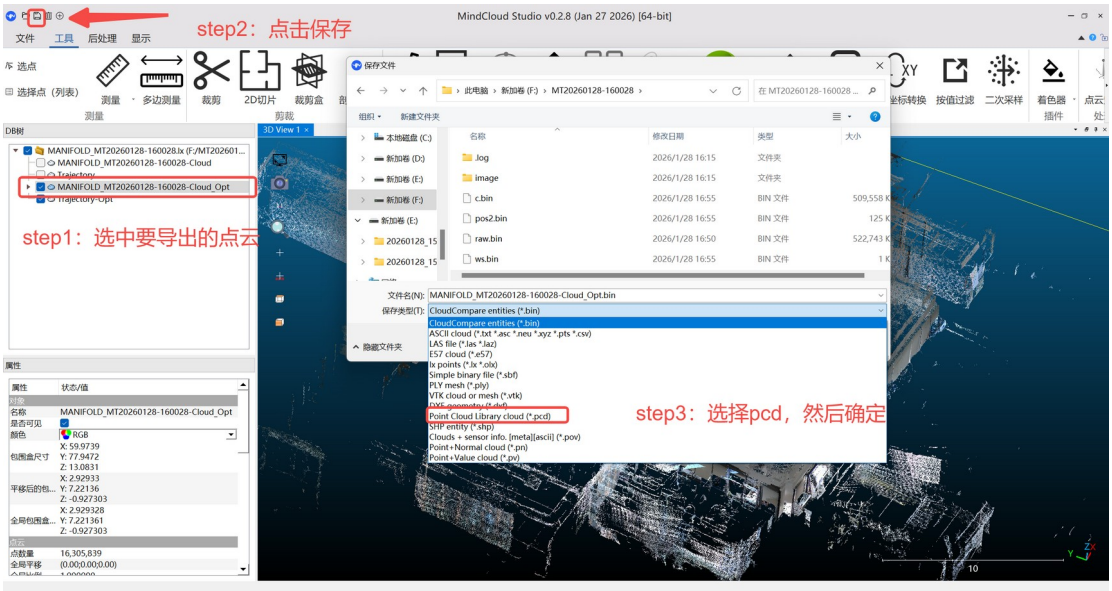
3.1 Process .olx / .lx data in MindCloud



3.1.1 Wait for processing to complete



3.1.2 Save PCD point cloud



3.1.3 Save .bin map

3.1.4 Copy the generated PCD point cloud and .bin map to the Ubuntu PC

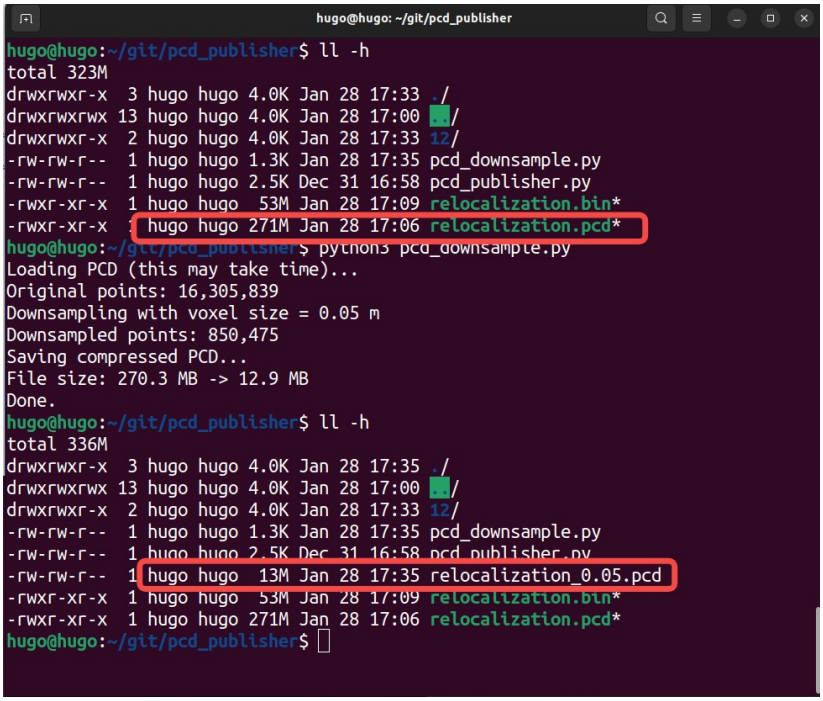
3.2 Point Cloud Downsampling

1. The raw point cloud can be very large. To display it in rviz, downsampling is needed.
1. Use the following script to downsample the PCD file to under 20 MB (recommended for smooth rviz performance):

```
Shell
#!/usr/bin/env python3
"""
超大 PCD 降采样工具（适合 RViz）
"""

###################### 可修改区域 ######################
INPUT_PCD  =
"/home/hugo/git/pcd_publisher/relocalization.pcd.pcd"
# 原始 4.3G PCD
OUTPUT_PCD =
"/home/hugo/git/pcd_publisher/relocalization_0.05.pcd"
# 输出压缩后
VOXEL_SIZE = 0.05                                  # 体素大小（单位:
米）
########################################################

import open3d as o3d
import os

def main():
    print("Loading PCD (this may take time)...")
    pcd = o3d.io.read_point_cloud(INPUT_PCD)
```

```
    print(f"Original points: {len(pcd.points):,}")

    print(f"Downsampling with voxel size = {VOXEL_SIZE} m")
    pcd_ds = pcd.voxel_down_sample(voxel_size=VOXEL_SIZE)

    print(f"Downsampled points: {len(pcd_ds.points):,}")

    print("Saving compressed PCD...")
    o3d.io.write_point_cloud(
        OUTPUT_PCD,
        pcd_ds,
        write_ascii=False,   # binary
        compressed=True      # open3d 压缩
    )

    orig = os.path.getsize(INPUT_PCD) / 1024 / 1024
    out  = os.path.getsize(OUTPUT_PCD) / 1024 / 1024

    print(f"File size: {orig:.1f} MB -> {out:.1f} MB")
    print("Done.")

if __name__ == "__main__":
    main()
```

1. After downsampling:



3.3 Run Relocalization and Publish PCD

3.3.1 Set relocalization mode and enter the correct .bin file path:

```
Shell
  custom_map_mode: 2      # 0: Odometry mode 1: SLAM mode 2:
Relocalization mode
  custom_init_pos: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
  relocalization_map_abs_path:
"/home/hugo/git/odin_0.8.0/map/relocalization.bin" # must be
set for Relocalization mode or will fail
```

3.3.2 Run the driver

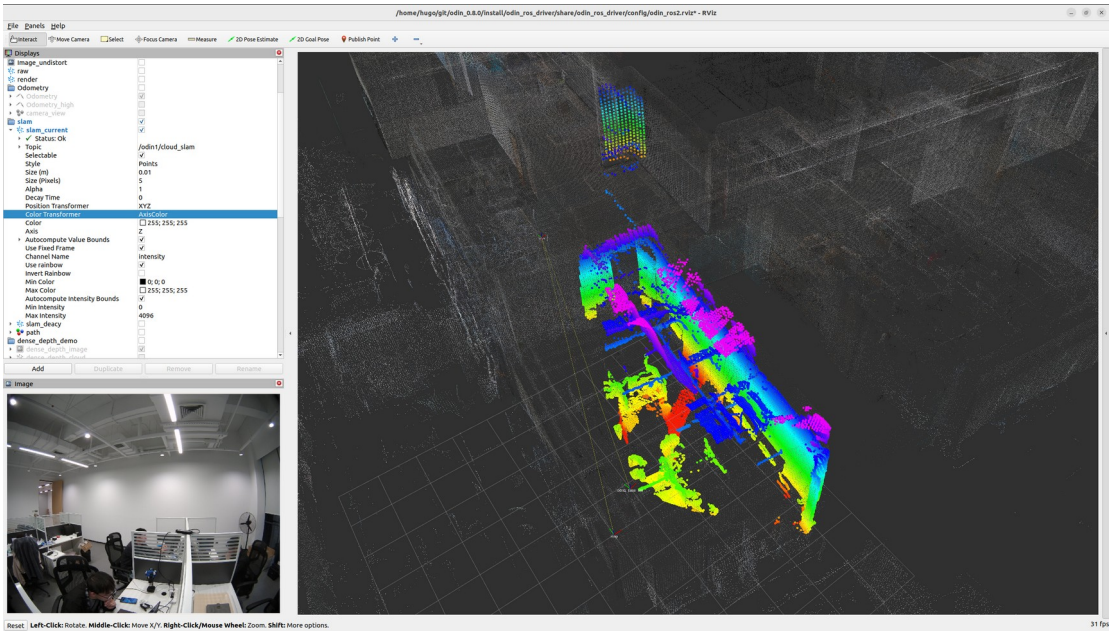
3.3.3 Use the following script to publish the downsampled PCD into ROS:

```
Shell

#!/usr/bin/env python3
"""
PCD (XYZRGB) -> ROS2 PointCloud2
使用 open3d，直接 python3 运行
"""

###################### 可修改区域 ######################
PCD_PATH  =
"/home/hugo/git/pcd_publisher/relocalization_0.05.pcd"   #
<<< 改这里
FRAME_ID = "map"
TOPIC    = "/pcd_points"
PUB_HZ   = 1.0
#########################################################

import rclpy
from rclpy.node import Node

import numpy as np
import open3d as o3d
import struct

from std_msgs.msg import Header
from sensor_msgs.msg import PointCloud2, PointField
from sensor_msgs_py import point_cloud2

def rgb_to_float(r, g, b):
    """RGB(uint8) -> packed float32"""
    rgb_uint32 = (int(r) << 16) | (int(g) << 8) | int(b)
    return struct.unpack("f", struct.pack("I", rgb_uint32))
```

```
[0]

class PCDPublisher(Node):
    def __init__(self):
        super().__init__("pcd_rgb_publisher")

        self.pub = self.create_publisher(PointCloud2, TOPIC,
10)
        self.timer = self.create_timer(1.0 / PUB_HZ,
self.timer_cb)

        self.cloud_msg = self.load_pcd(PCD_PATH)
        self.get_logger().info(f"Loaded PCD with RGB:
{PCD_PATH}")

    def load_pcd(self, path: str) -> PointCloud2:
        pcd = o3d.io.read_point_cloud(path)

        points = np.asarray(pcd.points, dtype=np.float32)
        colors = np.asarray(pcd.colors, dtype=np.float32)  #
[0,1]

        cloud_data = []
        for i in range(points.shape[0]):
            r = int(colors[i, 0] * 255)
            g = int(colors[i, 1] * 255)
            b = int(colors[i, 2] * 255)
            rgb_f = rgb_to_float(r, g, b)

            cloud_data.append((
                points[i, 0],
                points[i, 1],
                points[i, 2],
                rgb_f
            ))

        header = Header()
        header.frame_id = FRAME_ID

        fields = [
           PointField(name="x",   offset=0,
datatype=PointField.FLOAT32, count=1),
           PointField(name="y",   offset=4,
datatype=PointField.FLOAT32, count=1),
```

```
            PointField(name="z",   offset=8,
datatype=PointField.FLOAT32, count=1),
            PointField(name="rgb", offset=12,
datatype=PointField.FLOAT32, count=1),
]

        return point_cloud2.create_cloud(header, fields,
cloud_data)

    def timer_cb(self):
        self.cloud_msg.header.stamp =
self.get_clock().now().to_msg()
        self.pub.publish(self.cloud_msg)

def main():
    rclpy.init()
    node = PCDPublisher()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()

if __name__ == "__main__":
    main()
```

3.3.4 Expected result:



4. Record Relocalization Map Using Odin (Method 3)

4.1 Record .bin Map

1. Set customer_map_mode: 1, recorddata: 0 in control_command.yaml

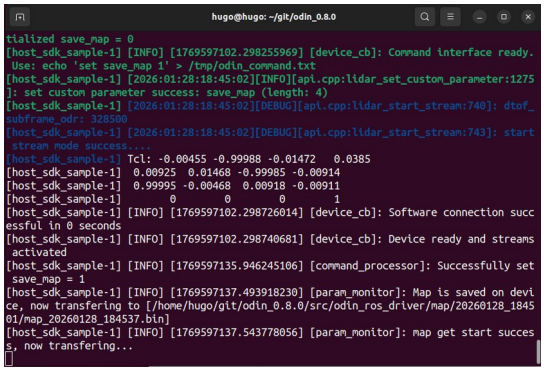
```
Shell
recorddata: 0            # 0: off; 1: on
***
custom_map_mode: 1       # 0: Odometry mode 1: SLAM mode 2:
Relocalization mode
```

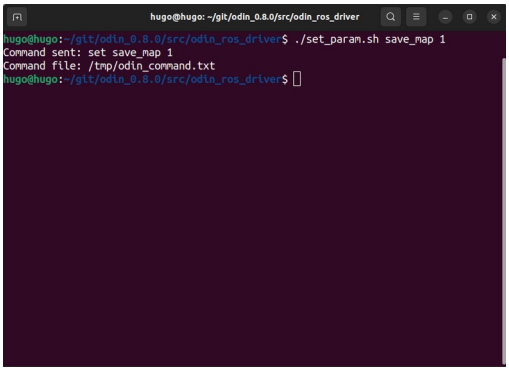
1. Run the driver to collect data

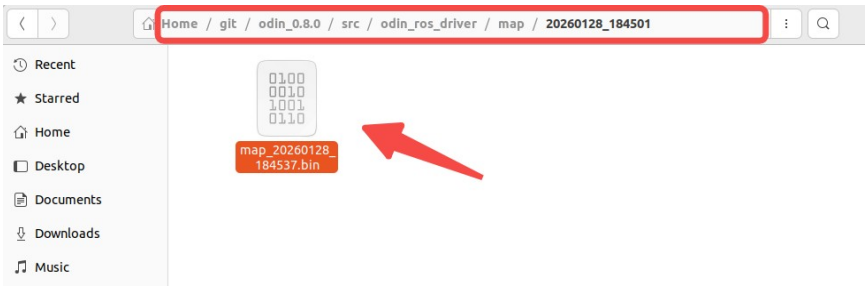
```
Shell
cd ~/catkin_ws
# ROS2 example; for ROS1 use corresponding commands
source install/setup.bash
ros2 launch odin_ros_driver odin1_ros2.launch.py
```

1. After map collection, do NOT press Ctrl+C directly. Open a new terminal, navigate to the driver path, and run ./set_param.sh save_map 1 to save the map.

```
Shell
cd ~/catkin_ws/src/odin_ros_driver
./set_param.sh save_map 1
# Wait until the terminal confirms the map is saved before
stopping the driver
```







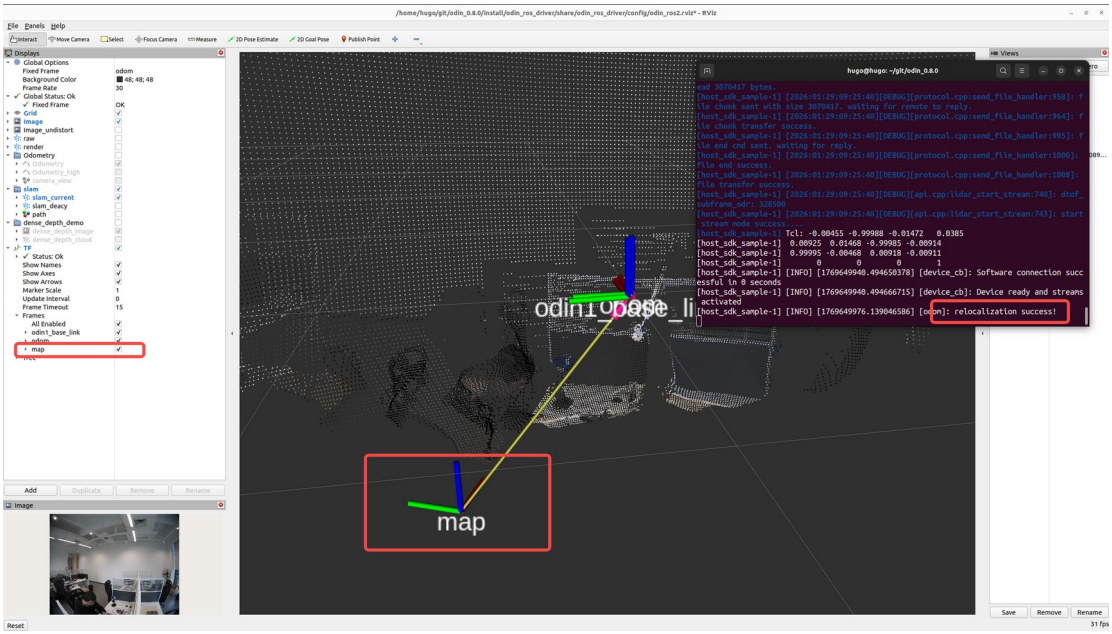
4.2 Run Relocalization Directly

1. Set customer_map_mode: 2 in control_command.yaml and configure the .bin file path:

```
Shell
custom_map_mode: 2      # 0: Odometry mode 1: SLAM mode 2:
Relocalization mode
custom_init_pos: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
relocalization_map_abs_path:
"/home/hugo/git/odin_0.8.0/src/odin_ros_driver/map/20260128_
184501/map_20260128_184537.bin" # must be set for
Relocalization mode or will fail
```

1. Run the driver

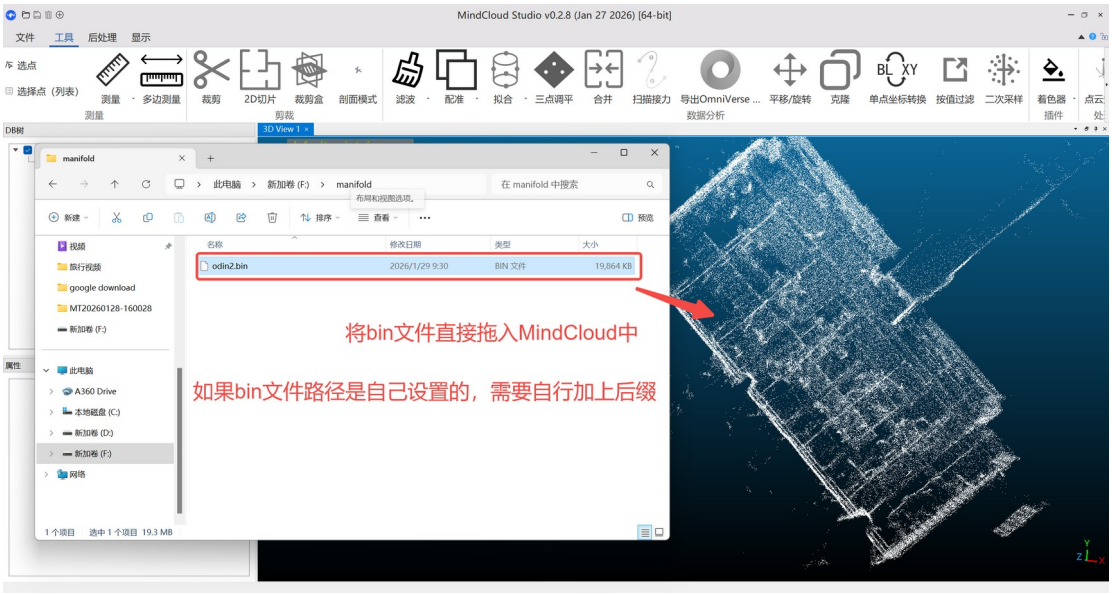
1. On successful localization: rviz shows the map frame; the terminal prints relocalization success.



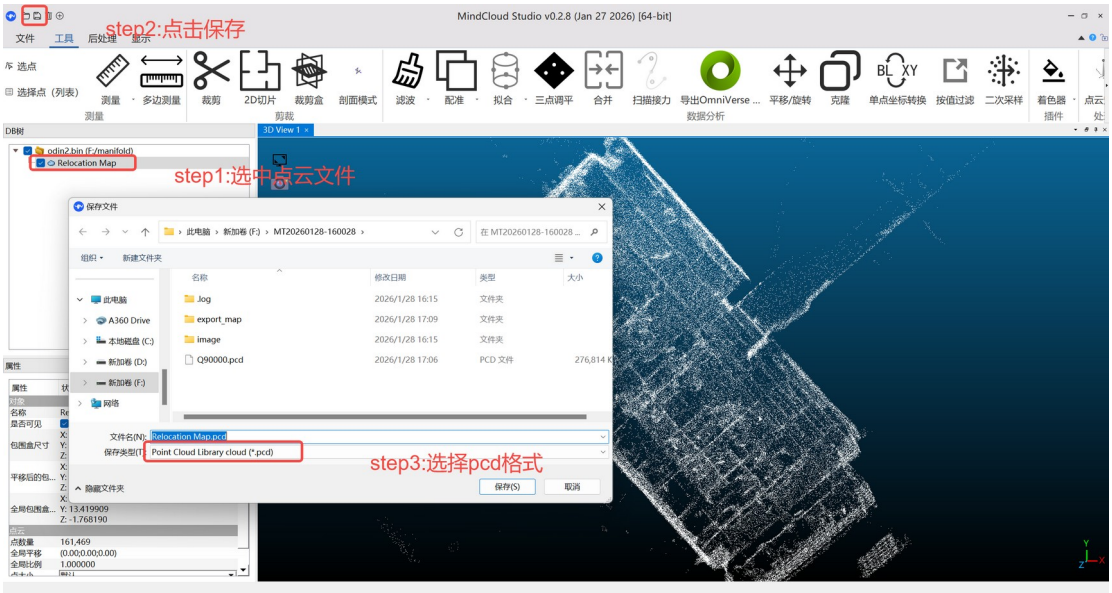
4.3 Export .bin Map to PCD

1. Copy the .bin file exported by Odin to a PC with MindCloud 0.2.8

1. Drag the .bin file directly into MindCloud



1. Export PCD format point cloud



4.4 Display PCD in rviz

1. Copy the PCD to the Ubuntu PC (where the driver is installed)

1. Follow the same steps as 1.5 to load it. Note: .bin-exported PCD has no RGB info. You may write your own publisher as needed.

Note: All code samples above are AI-generated demos; no technical support is provided for them.