Towards Heterogeneous Multi-Agent SLAM for Cooperative Map Completion

Sam Schoedel
Robotics Institute
Carnegie Mellon University
Pittsburgh, USA
sschoede@andrew.cmu.edu

Matthew Sivaprakasam
Robotics Institute
Carnegie Mellon University
Pittsburgh, USA
msivapra@andrew.cmu.edu

***Abstract*—Many downstream tasks in robotics rely on having a prior map, such as path planning, intelligent obstacle avoidance, or computationally expensive inspection of an environment that can't be done online. To create maps efficiently, individual observations from multiple robots can be aggregated into one large, dense map. However, it can be difficult to align maps from different robots and, when discrepancies between maps arise, determine which is correct. We propose a robot-agnostic algorithm to generate high fidelity point cloud maps that incorporates automatic registration and reconciles disparities by generating confidence maps for each robot. We validate our algorithm by running it in real time with maps generated by a ground and aerial vehicle. Our code is available at https://github.com/Matthewjsiv/mushr_slam**

***Index Terms*—SLAM, descriptor, map fusion, multi-agent**

I. Introduction

Using multiple robotic agents to map a single area is efficient because it allows different robot modalities to tackle different sections of the map. For example, drones used to survey an area can observe large portions of the map at a time and are easy to manipulate, but sacrifice resolution and are frequently subject to obscured views of objects under canopies. On the other hand, ground robots can achieve viewpoints that drones can't but don't necessarily have the high-level understanding of large environments to efficiently navigate the terrain. One way to obtain thorough representations of an environment is to use both types of robots in a complementary fashion. We propose using a high-level aerial-view map from a drone to inform a ground robot's trajectory as it fills in gaps. Our setup involves capturing an aerial point cloud map and then doing computation online with a ground vehicle to further explore the map and fill out unseen sections. However, our algorithm may be extended to run both vehicles, or even more than two robots, at the same time by streaming updated map information between each agent. The scenario is simplified in our case where we are storing the fused map on the ground vehicle, but in a real-world scenario each agent may be communicating with a ground station that stores the fused map and instructs each agent where to travel to continue efficient exploration of the environment.

The Iterative Closest Point (ICP) algorithm is often used for registering a source pointcloud into a target pointcloud. However, it requires a strong initialization so that it doesn't converge to incorrect solutions. The random sample consensus (RANSAC) algorithm is often used to initialize ICP, when there is a strong enough chance that one of the random guesses will be close to the solution. Unfortunately, this is not feasible for the task of registering a local map into a large global map, as it would require too many RANSAC iterations for initialization. We circumvent this issue by generating feature descriptors for square patches of the pointcloud that are then evaluated against a query with a distance function to find the best match for initialization.

Fig. 1. MuSHR vehicle (top). Simulated drone platform (bottom).

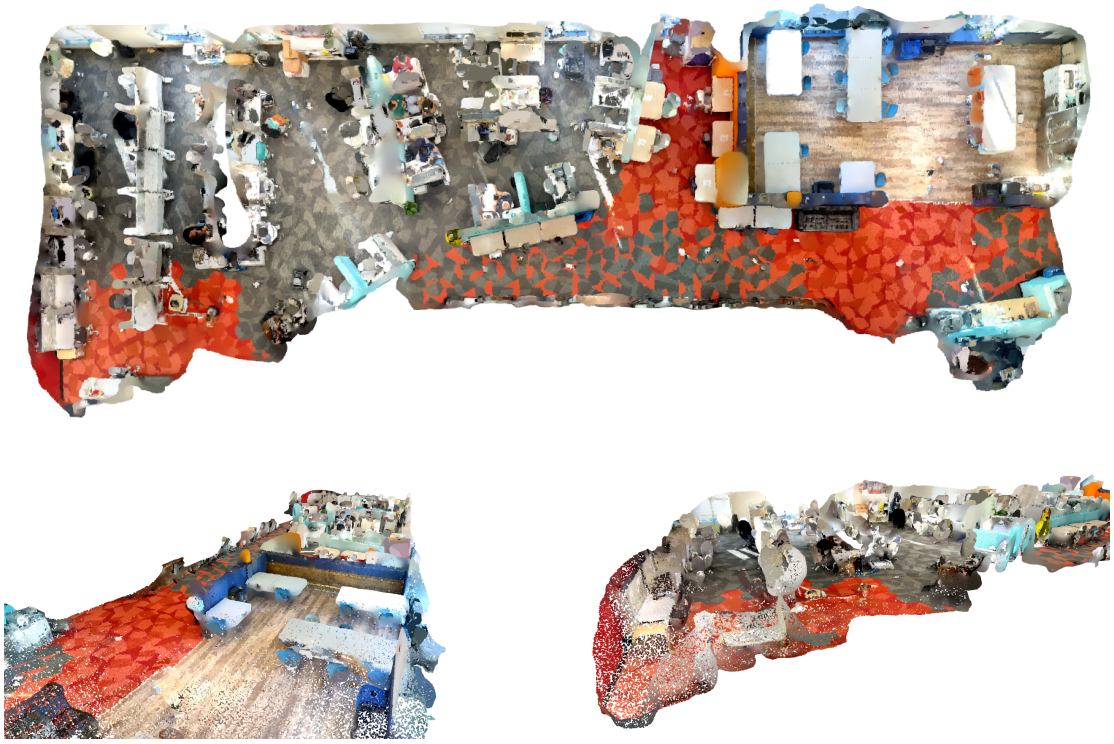

Fig. 2. Point cloud obtained from aerial sensor.

II. Related Work

Feature descriptors provide one way of registering two sets of pointclouds together or detecting loop closures. He et al. propose a way of generating features from a hand-crafted algorithm [1]. They first pre-define a set of rotations along each axis (for example describing azimuth and elevation angles), which are used to define a set of planes. The points are then projected onto these planes, and a histogram is calculated to describe the density of the points on each plane. This approach works well, but is only demonstrated on datasets such as KITTI, where the platforms the data was collected on have similar views of the environment (in contrast to the views from a drone vs a car).

Another key issue in the heterogeneous multi-agent SLAM problem is knowing when to trust one agent over the other in the map-fusion stage. This can also be considered analogous to the challenge of resolving merges from multiple viewpoints in single-agent map-fusion. Yang et al. propose an approach in which they incorporate a probabilistic component into the fusion stage [4]. By quantifying the uncertainty of candidate points, they are able to select the most-certain points for fusion.

The accumulation of points into the robot's local frame is also important. If only a single frame of points are used (e.g. one lidar scan), the sparsity can make it difficult to compute features that will be in a similar domain to the accumulated global map. Vizzo et al. propose one way of doing this through well-tuned ICP to achieve SLAM [3]. They utilize ICP to find the transform between sequential scans, and integrate the results to provide an odometry estimate without requiring other sensors such as a camera or IMU.

III. Methodology

We employ ICP to properly localize the ground vehicle's point clouds within its own reference frame to accumulate a local, dense point cloud map. We then use descriptor matching for initial localization with the aerial map and use a heuristic algorithm to fuse them into one high fidelity point cloud. The result is a map that can be continuously updated as it takes in new data from the ground vehicle. The ground vehicle can then be informed by that map to explore either locations in the existing aerial map that need to be filled in or edges of the combined map that have not yet been seen.

A. Mapping

1) Aerial Robot: Due to the infeasibility and danger of operating a drone big enough to carry a lidar sensor in the open space of our indoor test site, we simulated this data by attaching an iPhone with its lidar sensor to the end of a metal beam, raising it approximately 12 feet, and pointing it down. We then utilized the PolyCam app to record the lidar data and perform dense SLAM while we carried the "drone" around the office. This resulted in our global map shown in Fig. 2. It can be seen that it is a relatively detailed pointcloud. However, some areas are warped and all of the sections beneath surfaces (e.g. tables) are incomplete due to occlusion.

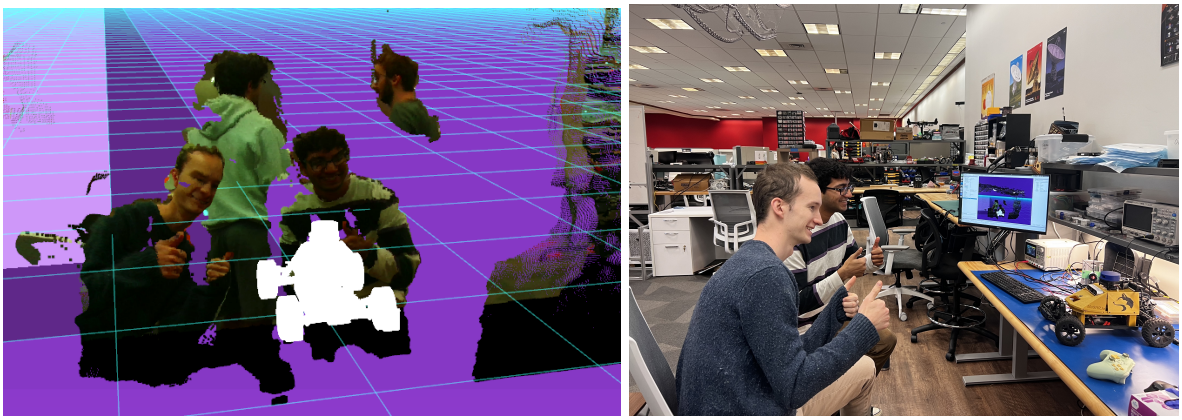

Fig. 3. Example point cloud generated by Intel RealSense D435i onboard the MuSHR robot.

2) Ground Robot: Our ground hardware setup requires two separate components, the ground vehicle and real-time point cloud capture. For the ground vehicle we used a MuSHR robot, an autonomous racecar project developed by students at the University of Washington [2]. The assembled MuSHR robot with depth sensor is shown in Fig. 1. The 2D lidar on top of the robot is unused, as it does not generate enough data and is mounted in a way that makes it infeasible for us to reconstruct a dense 3D map. Instead, we attached an Intel RealSense D435i depth camera to provide a point cloud view of the environment in front of the robot. We integrated the RealSense depth camera with the MuSHR's existing software stack using ROS. An example point cloud from the system is shown in Fig. 3.

We integrated the MuSHR stack with the open-source KISS-ICP algorithm [3] in order to generate local maps in real-time, an example of which is shown in Fig. 4. Unfortunately, we found the RealSense sensor to be too noisy, resulting in maps that we could not robustly register into the global map. To constrain the problem, we instead used the same approach we used to collect the aerial map as shown in Fig. 5.

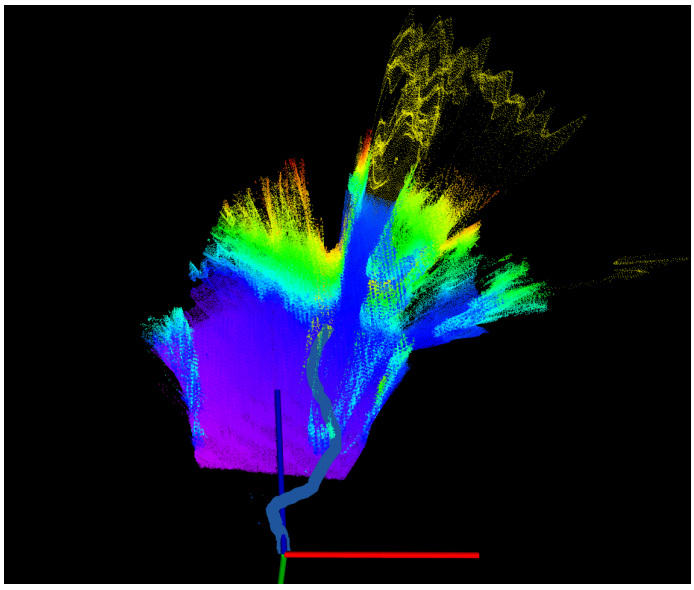

Fig. 4. Ground vehicle internal map representation generated using KISS-ICP. Current scan is in yellow and the colored pointcloud shows the accumulated points.

B. Ground to Aerial Registration

1) Descriptor Matching: The descriptor matching stage involves computing a descriptor from the local map produced by the ground-vehicle, and finding which descriptor from the global map contains the closest match. We experimented with two descriptors: M2DP [1] and our own custom descriptor that utilizes both geometric and visual information. To generate the descriptors, we first pre-process the data following these steps:

1) Subtract the mean
2) Downsample to 0.06m resolution
3) Crop to a local 3.5x3.5m patch
4) Remove points above 1.8m

where we crop by height to account for the fact that the ground vehicle will not see points above a certain height. For M2DP, we compute 16 bins along θ, 16 bins along ρ, 3 bins along azimuth, and 3 bins along elevation. For our custom descriptor, we bin the points by height into 8 bins. For each bin, we calculate:

1) Proportion of points in that bin (1 element)
2) Standard deviation along each axis (3 elements)
3) Average color of the points (3 elements)

This gives us a 56-element descriptor (7 features for each of 8 bins).

We generate the descriptors for the global map offline. We start with a 3.5x3.5m window of points and sliding it across the map with a stride of 0.25m, storing a mapping between a feature vector for each window and its XY location in the map, discarding areas with less than 500 points. This then allows us at run-time to take the local map, compute its descriptor, and then obtain an estimate of its location in the global map by finding which global descriptor is closest. The closest vector is that which has the smallest euclidean distance from the local

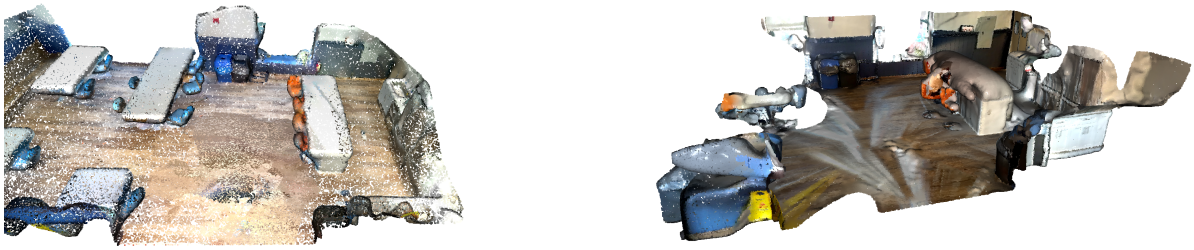

Fig. 5. Point cloud maps from drone (left) and car (right).

vector.

2) ICP for Refinement: While the descriptors allow us to obtain an estimate of the local map in the global map, the accuracy is upper bounded by the stride used to compute the global descriptors, which means that the estimate in this case can be up to 0.25m wrong. To refine this estimate, we use ICP initialized with the estimate provided by the descriptor matching.

It should be noted that the descriptor matching only provides an XY location and no rotation estimate. For simplicity, in this work we manually ensure that the local and global maps are oriented similarly so that only the XY location needs to be estimated. In reality, the rotation estimate can be solved by adding a RANSAC stage before the ICP state. By constraining the random samples to the approximate location in the global map, a good initialization for ICP should be able to be obtained within reasonable time.

C. Global Map Fusion

Once the aerial and ground maps are registered, we combine them into a confidence map for each pixel in each map in such a way that map features are preserved while incorrectly placed points are deleted. The heuristic we use is three-fold. First, for each point cloud, we determine the minimum distance between each point in that cloud and all of the points in the other cloud. Points with large minimum distances to the other cloud are given low confidence and points with small minimum distances are given high confidence. This heuristic assumes points that are close together are more likely to be correct because both vehicles agree on their locations. However, points in the same general location in both maps may be far enough apart for all of them to have low confidence. This would result in removing all points from some areas of the map, which we don't want since our goal is to reduce the number of holes in the map. Thus, we must choose one set of points to keep. We break these ties by adding a second heuristic, which is the nominal sensor orientation of each robot. We first compute the normals for each point using the location of surrounding points and then compare those normals to the sensor direction of the robot that obtained them. Because the aerial vehicle takes sensor data primarily from a birds-eye view, the confidence values for points from the aerial map with normals pointing up are increased. For the ground vehicle, since the camera is mounted forward on the car, the confidence values for points from the ground map with normals pointing to the side are increased.

Fig. 6. Descriptor matching success and failure cases.

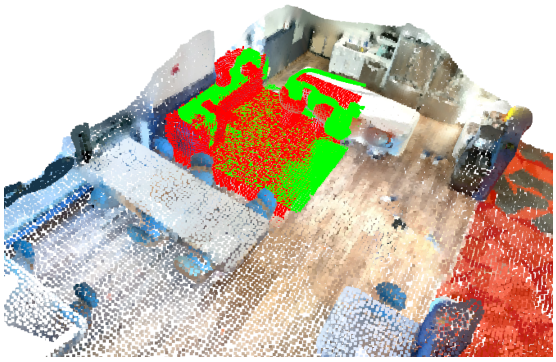

Fig. 7. Descriptor matching results from querying the local map on the global map.

IV. Experiments & Results

A. Ground to Aerial Registration

1) Pipeline Testing: Prior to integrating with the rest of the system, we made sure that our descriptor matching algorithm itself worked. This was done by pre-computing the global descriptors as previously described, and randomly choosing a location in the global map to generate a new descriptor from. To test, we ran our algorithm and ensured that the XY location corresponding to the closest descriptor in the offline bank was close to the random query. This was a sufficient test since the random location was of higher precision than the stride at which the offline descriptors were computed, meaning that the random query will not have been previously seen. An example success and failure of our algorithm can be shown in Fig. 6.

In practice, we found that our custom descriptor performed better than M2DP, most likely due to the addition of color information, as well as inherent properties in the maps we collected.

2) Local-to-Global Testing: Upon affirming our descriptor matching pipeline, we tested it using the local map generated from the ground robot. This was done by choosing a point in the local map and following the pre-processing steps and matching steps outlined in the methodology section. Overall, we found this approach to be successful, despite the different viewpoints of the aerial and ground vehicle. An example is shown in Fig. 7, where the local query is in green and global patch in red.

3) ICP Refinement: The final stage in registering the local maps into the global maps is ICP refinement. Using the initialization from the descriptor matching, we initialize an ICP algorithm to increase the accuracy of the estimate. An example result can be seen in Fig. 8, where the denser area comes from the inclusion of the local map. Note that the high-level features between both maps line up, such as the floor and walls, and with the inclusion of the local map we get a more accurate representation under occluded areas like the desks.

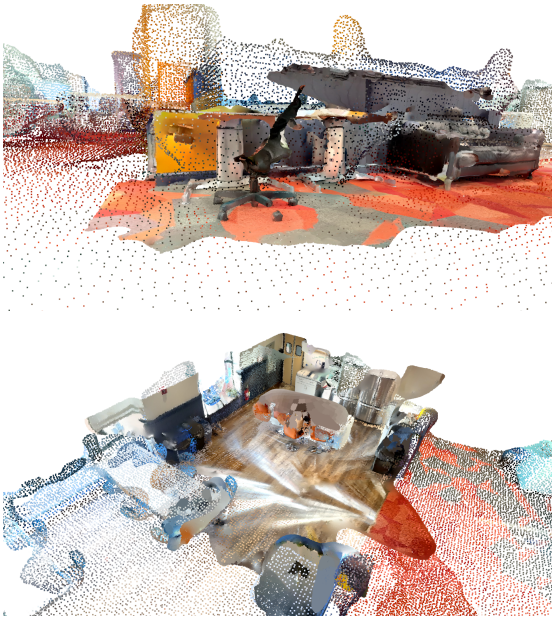

Fig. 8. Example final registration result after running both descriptor matching and ICP refinement.

B. Map Fusion

Running our heuristic map fusion algorithm on the registered point clouds results in a new map that includes points from both vehicles while taking into account the probability of each point existing given the prior that it was measured from a given robot. The confidence levels for each map are shown in Fig. 9.

Points with normals facing upward in the aerial map, shown in the left part of the image, have high confidence compared to points with normals pointing to the side. The opposite is true for points in the ground map, shown in the right side of the image. Pixels in the floor are still high confidence because both robots measured them to be in the same location. Deleting pixels below a certain hyperparameter threshold and then combining the point clouds into one map representation results in the map shown in Fig. 10

The result is a point cloud that is far more dense in the areas viewed by both the aerial and ground vehicles that captures geometries underneath areas previously occluded from the drone's point of view.

V. Conclusion

We developed a new point cloud descriptor for feature matching, applied ICP for refined point cloud registration, and created a heuristic algorithm for reconciling disparities between points. This work makes progress in multi-agent map completing assuming computation is being done on one robot, but can be extended to any number of vehicles operating at the same time. This work may also be used as a starting point for more intelligent path planning with multiple agents exploring an environment. During point cloud fusion, we generate both point cloud density from the fused map and uncertainty metrics

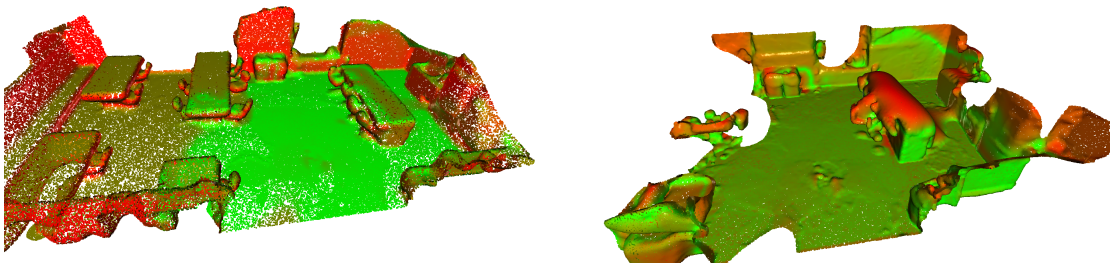

Fig. 9. Confidence maps from drone (left) and car (right).

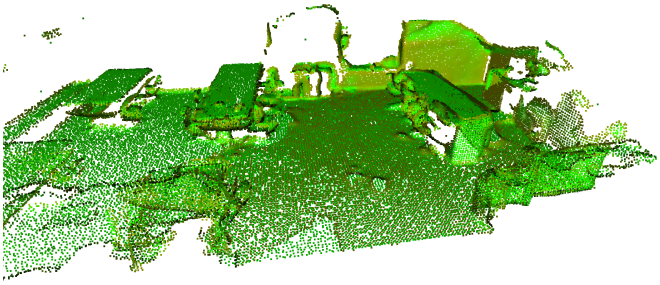

Fig. 10. Fused map.

from the prior aerial map that can be used to inform agents where to travel to obtain more information and fill out the map. Additionally, confidence estimation as shown in this work was fine-tuned for operation with two robots, but can be extended to any number by simply feeding the heuristic fusion algorithm each robot's most common viewing angle. There are a number of potential future directions based off of this work. Voxel carving could be implemented as a supplement to help eliminate some of the artifacts that are left over after our confidence-based fusion. There could also be some improvements made to the confidence estimation, drawing inspiration from sensor-models rather than relying solely on the orientation of the normals. Additionally, we want to keep recent measurements that capture geometry which did not exist in the prior map and discard measurements from the prior map that capture geometry which does not exist in the most recent map. Future work can take this into account simply by increasing the confidence of recently measured points.

References

[1] Li He, Xiaolong Wang, and Hong Zhang. "M2DP: A novel 3D point cloud descriptor and its application in loop closure detection". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 231–237. DOI: 10.1109/IROS.2016.7759060.

[2] Siddhartha S. Srinivasa et al. "MuSHR: A Low-Cost, Open-Source Robotic Racecar for Education and Research". In: *CoRR* abs/1908.08031 (2019).

[3] Ignacio Vizzo et al. "KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way". In: *IEEE Robotics and Automation Letters (RA-L)* 8.2 (2023), pp. 1029–1036. DOI: 10.1109/LRA.2023.3236571.

[4] Jun Yang, Dong Li, and Steven L. Waslander. "Probabilistic Multi-View Fusion of Active Stereo Depth Maps for Robotic Bin-Picking". In: *CoRR* abs/2103.10968 (2021). arXiv: 2103.10968. URL: https://arxiv.org/abs/2103.10968.