Multi-Agent Autonomous Mapping of Unknown GPS-Denied Environments Using a Relative Navigation Framework

Jacob M. Olson[1], Timothy W. McLain[2] [TODO: include Matthiew Labbe?]

***Abstract*— [TODO: write the abstract]**

I. Introduction

[TODO: write the rest of the introduction (mapping/slam background)]

The remainder of the paper is organized as follows: Section II describes framework used to map the environment and background on what previous work has made this research possible. Section III details the planning and control schemes used to successfully navigate the unknown area. Then method used to combine maps of multiple agents are then explained in Section IV. Results showing and evaluating the generated maps are presented in Section V. Finally, conclusions are presented in Section VI.

II. Technical Approach

A. Problem Statement

The goal of this project is to successfully map a GPS-denied environment using multiple UAVs collaborating with eachother to explore the environment. This framework assumes that a good set of high level waypoints will be provided that ensure sufficient loop closures and coverage of the desired area. The focus in this paper is properly estimating the UAV states to allow for successful GPS-denied navigation and map merging. The network diagram showing the framework used is shown in Fig. 1 which will be described in detail in this section and throughout the paper.

B. Sensors

Since we are operating in a GPS-denied environment, we are not able to rely on GPS measurements to give us global information about where the UAVs are located. As shown in Fig. 1 the sensors used by the UAV to estimate its state are an RGB-D camera, a planar laser scanner, a LiDAR pencil-beam sensor and an IMU on the onboard flight controller. Using only these sensors and the flight computer, we must be able to accurately estimate the states of the UAV enough to send waypoints and control the attitude. Since global position is not measurable and must be estimated, it is important to not rely on the current global estimates for attitude control. [TODO: how much do I talk about sensors here?]

C. Estimation

Estimation is the most critical element in enabling autonomous flight. Without good position and attitude estimation, autonomous navigation algorithms do not function. To avoid the issue of loop closures causing instability in the controller, we separately estimate the global and relative position and do not directly rely on the global estimate to control the attitude of the UAV. The current estimates are stored in a transformation tree as shown in Fig. 2. The world frame is the inertial NED frame of the world with a static transform to the map frame. The map frame is a NWU frame set by RTAB-Map as in the starting location where it is initialized. RTAB-Map uses the odom fram to adjust for loop closures without causing spikes in the odometry. The odom frame starts with zero transformation from the map frame. Each time a new loop closure is detected, if the map optimization shifts the global position estimate, rather than alter the odometry message, causing spikes in the estimated odometry, the entire tree from odom down is shifted with the loop closure adjustment. Every time RTAB-Map creates a new node in the graph, the visual odometry node resets the keyframe transform to the current base-link transform. the base-link transform represents the current estimated position of the the UAV in the NED frame and camera-link represents the position in the NWU frame, the camera-base-link transform represents the current position of the camera in the camera frame. the waypoint manager drops the current waypoint connected to the world frame. This way a loop closure does not shift the desired gloabal position of the waypoint. The position controller controls on the error between the waypoint and base-link. This is under the assumption that the desired waypoint location is known globaly, this could be modified to set the new waypoint relative to odom frame if the position is relative to the current obstacles rather than the global position.

1) RTAB-Map: RTAB-Map is a powerful open source software package that uses graph-based SLAM with appearence based loop closures to generate high-quality, dense 3D maps of environments without the use of GPS. RTAB-Map is also able to accurately estimate position within the map with little error. [TODO: cite RTAB-Map papers] We use the visual odometry from RTAB-Map as an input to the estimation for the relative framework. We directly use the current global estimate of RTAB-Map for the position controller and waypoint manager.

RTAB-Map has been extended to build a single map from multiple different sessions at different times, but currently is not able to handle simultaneous multi-agent mapping. This paper proposes a method to extend the functionality of RTAB-Map to combine the maps of multiple agents flying simultaneously into a single map in near real time.

RTAB map does not estimate the attitude of a UAV with enough frequency to autonomously navigate so we used the relative navigation framework to estimate attitude and

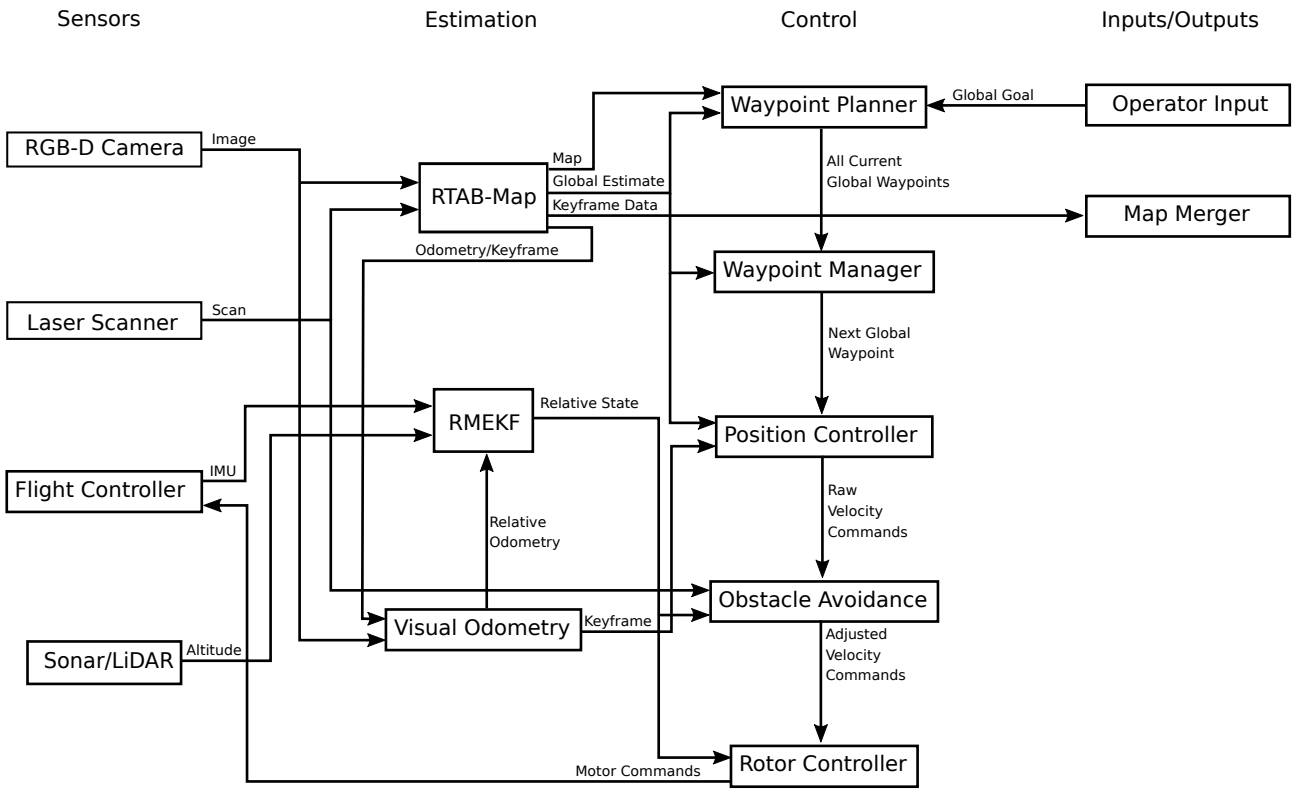

Fig. 1. The network diagram of the relative navigation framework proposed in this paper

relative state.

2) Visual Odometry: The odometry and keyframe information from RTAB-Map is used to generate a relative odometry message that is used by the relative estimator. RTAB-Map produces a global visual odometry. the visual odometry node publishes only the translation and rotation from the last keyframe that has been dropped. when RTAB-Map begins a new keyframe, the relative odometry message flags the keyframe change and resets the relative odometry. this makes it so that the large global position jumps caused by loop closure do not translate into the relative odometry and it is only tracking the transform between keyframes.

3) RMEKF: The Relative Multiplicative Extended Kalman Filter (RMEKF) was shown to successfully estimate the UAV's relative state sufficient to autonomously navigate in GPS-denied environments that had been previously mapped. Thus far, however, it has not been extended to estimation and navigation in unknown and unmapped environments, this paper proposes a method to extend the functionality to these environments.

The RMEKF takes as inputs the IMU measurement from the flight controller, the relative visual odometry based on a keyframe approach, and the attitude measurement from the LiDAR pencil beam sensor. Then using the multirotor dynamics model, is able to accurately estimate the relative state of the UAV. It is important to note that the RMEKF makes no effort to estimate the global position of the UAV. This makes it so that large corrections in the estimated global position that happen with loop closures do not cause the estimator to diverge, and therefore cause stability issues in the velocity and attitude controllers onboard the UAV. The RMEKF functionality and results are further detailed in [TODO: cite relative nav papers].

D. Control

To sucessfully control a UAV in a GPS-denied environment, the control must be segmented into different categories to take advantage of both global and relative estimates. The control scheme cascades from an input map, current estimated position, and current goal position and outputs the motor commands at the end.

1) Waypoint Planner: The first stage of the control is the waypoint planner. It takes the global goal, current known obstacles, and current global estimates as obstacles and outputs a set of waypoints to reach the goal while avoiding the current waypoints. This stage will be further explained in Section III.

2) Waypoint Manager: After receiving the current set of waypoints, the waypoint manager selects the appropriate waypoint for the UAV to fly towards and sends the global location to the position controller. The waypoint manager monitors the position and angle error between the current estimated position and the current waypoint and when the error crosses below a user-defined threshold value, the next waypoint is sent to the position controller.

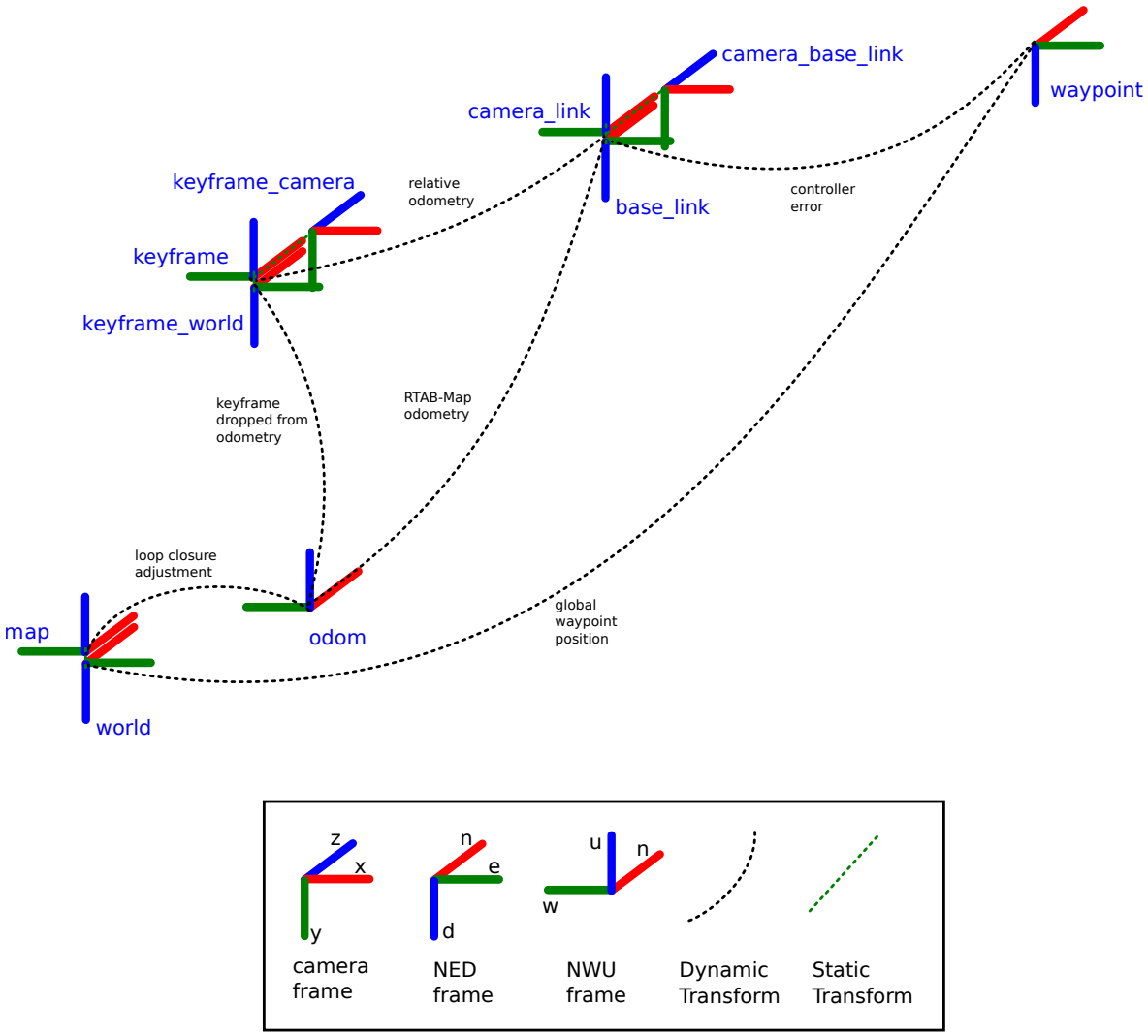

Fig. 2. The transformation tree of the reference frames used in estimation and control.

3) Position Controller: The position controller uses the error between the current estimated position of the UAV and the next waypoint. The goal of the position controller is to drive the error to zero. Since it operates in the error space of the the UAV rather than the state space, sudden shifts in the UAVs position estimate caused by loop closures have minimal effect on the controller and it is able to continue controlling the error to zero. The output of the position controller is a velocity command for the UAV.

4) Obstacle Avoidance: Before passing the velocity control into the attitude controller, it is filtered through an obstacle avoidance node as described in [TODO: cite obstacle avoidance], which uses the current relative estimates and obstacles detected py the planar laser scanner and using a potential fields method pushes away from obstacles and towards the commanded velocity input. The obstacle avoidance node then sends the modified velocity command to the attitude controller.

5) Attitude Controller: The attitude controller is a conventional PID controller that takes the velocity input commands and outputs the motor commands to the flight controller.

E. Inputs/Outputs

The input for each agent in the system is the desired goal location either from operator input or from a high level path planner. Each agent then sends the keyframe data consisting of the color and depth images and features from the images to the map merger where the maps are combined into a single node. The map merger will be further explored in Section IV.

III. Planning

The planner used in this project is an RRT planner with path smoothing and reactive path adjustments.

A. Global Goal Following with Relative Estimation

As mentioned earlier, the estimation that is critical to keeping the UAV airborne makes no attempt to estimate the

global position of the UAV, but only its relative state with respect to the last keyframe. By doing so, the global estimate of the position does not have to be continuous and is able to slide and adjust for detected loop closures without affecting the covariance of the estimator. Because there is no need to

B. Reactive Path Planning

The path planner uses reactive path planning to continuously fly as the known map is updated. As seen in Fig. 3. The UAV plans a simple path to the goal which avoids the obstacles that it can see when it starts flying. After it begins flying, the obstacle map is updated and the current flight path is periodically checked for collisions with new obstacles. If collisions are found, it modifies the path to avoid the newly discovered obstacles. This process continues until the agent is able to successfully reach the goal. Since the path is updated any time an obstacle is detected in its path, no prior knowledge of the environment is required to begin flying.

It is also able to navigate through complex maps. Since it is using RRT to plan the paths, as long there is a possible path to the goal, it will find a way to reach the goal, and in most cases, it will find a path to the goal in far less time than an exhaustive search method would. We decided to not extend the planner to use RRT* to allow it to be more random in the flight path to encourage more exploration of the map while flying paths.

IV. Map Merging

The map merging process proposed in this paper is able to generate a combined map from multiple agents on a base station computer in near real time during while the UAVs are still simultaneously mapping the environment.

To merge the maps in near real time, each time a new keyframe is initialized, the data from the keyframe consisting of an XYZRGB (color and depth) pointcloud, and the features extracted from the color image are stored in a database. The pointcloud is generated using the color and depth images from the RGB-D camera onboard and the features are generated from either SIFT/SURF or ORB using opencv. The database refered to as RGB-D Cache in Fig. 4 is hosted on the base station computer.

Once the database has been initialized the maps are periodically merged using functions from an instance of RTAB-Map running on the base station computer very similarly to how individual maps are generated for each UAV. The first step is to search for loop closures using a bag-of-words with the features in each keyframe, this process looks for both loop closures an agent has with itself and with other agents, each time a loop closure is found an edge is added to combined map graph with an estimated transformation between images. After finding all loop closures with the current dataset, the graph is optimized using the pose graph optimizer built into RTAB-Map, the optimized pose graph is then sent to the map assembler along with the XYZRGB pointcloud from each keyframe where the pointclouds are combined according to the optimized graph edges. This generates a single map with all keyframes that can be connected into a single graph. This map is then processed to remove some noise and filter out the ceiling to make the map more understandable to the operator.

V. Results and Discussion

A. Simulation

B. Hardware

VI. Conclusions

References

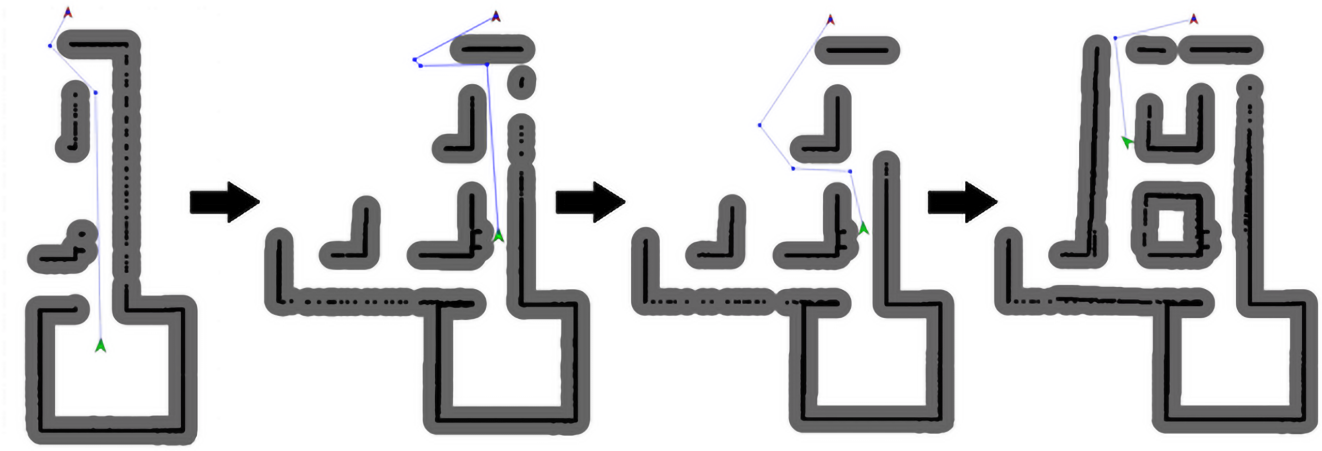

Fig. 3. An example of how the reactive path planner works as the UAV flies the planned path. The current estimated position is marked by the green arrow, the current goal position is marked by the red arrow, and the current path planned is marked with the blue lines. Detected obstacles with their respective safety buffers are represented with black and grey respectively.

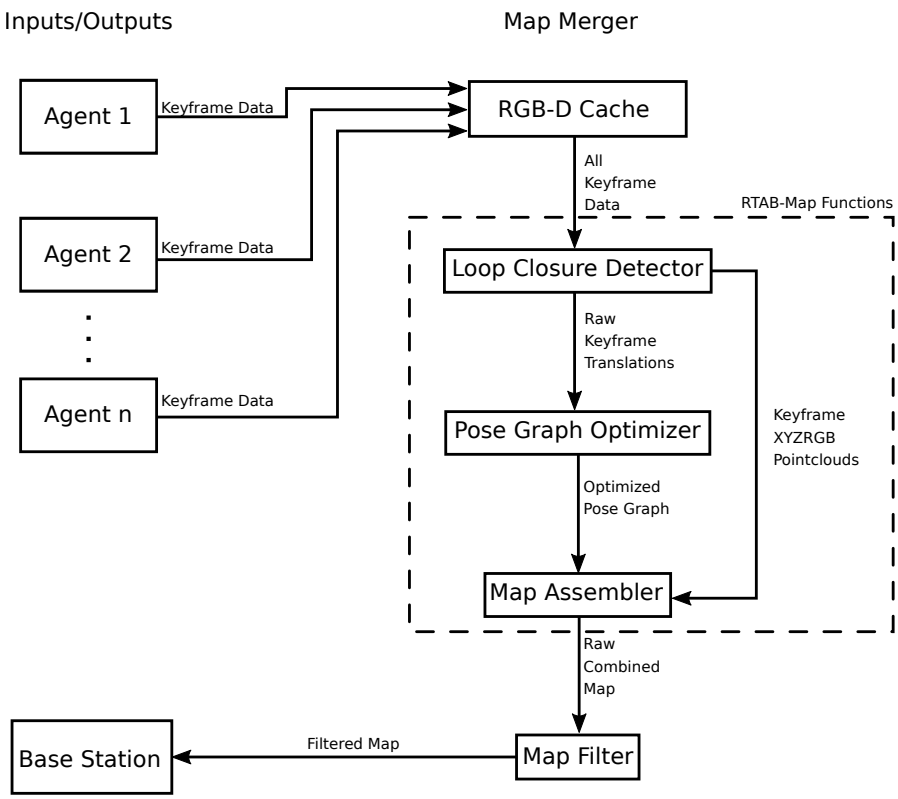

Fig. 4. The network diagram for the multi-agent map merging node proposed in this section.

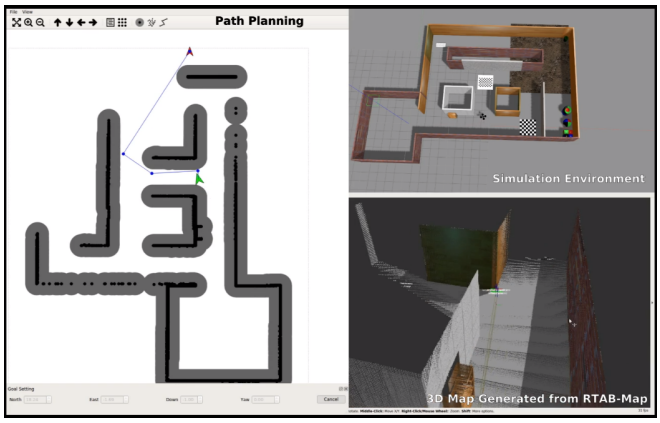

Fig. 5. Setup used for the simulation results

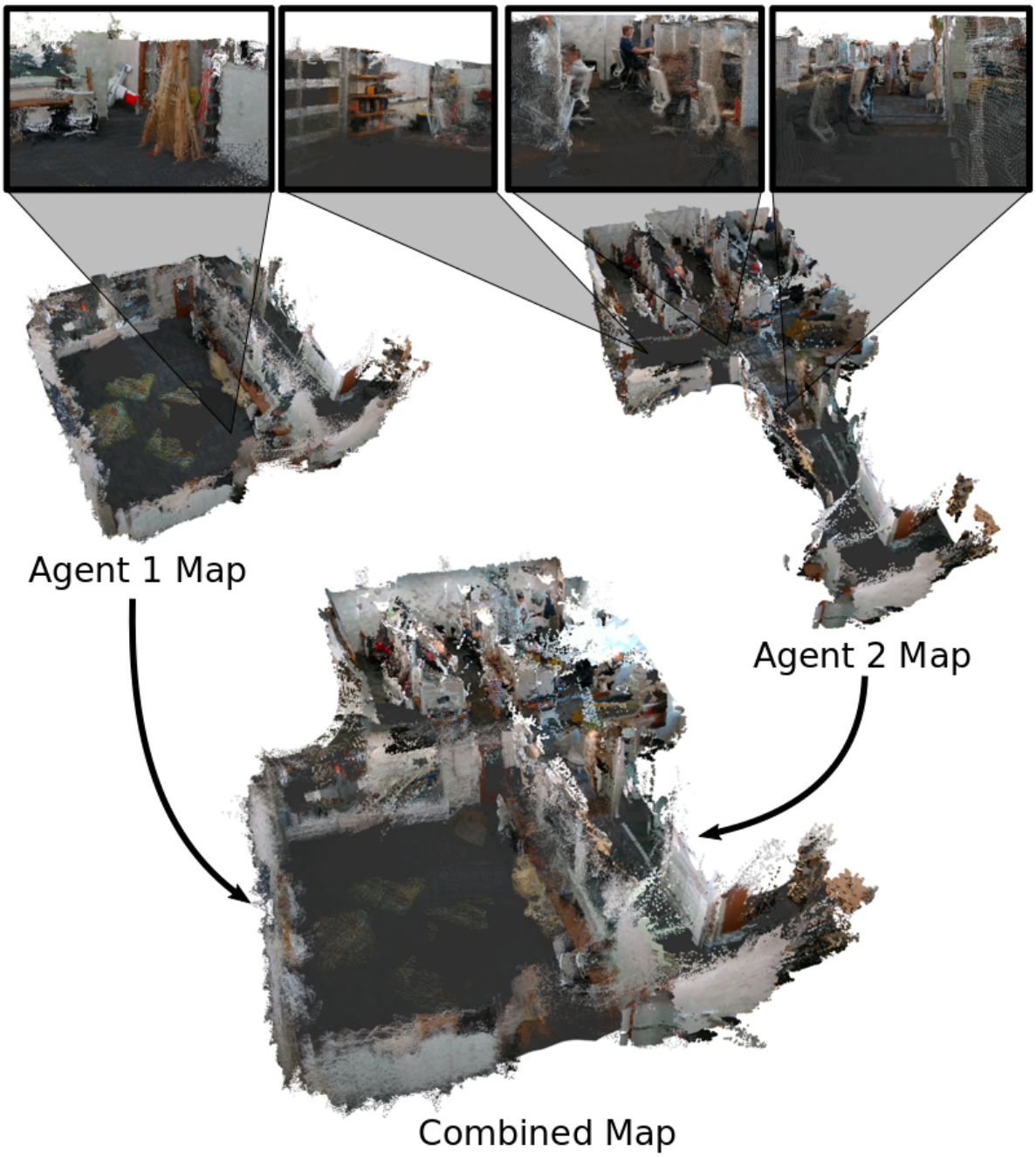

Fig. 6. Example of hardware results of merging maps from two agents into a single map in a cluttered environment. Above the individual agent maps are examples of the detail in the pointclouds when zooming in.

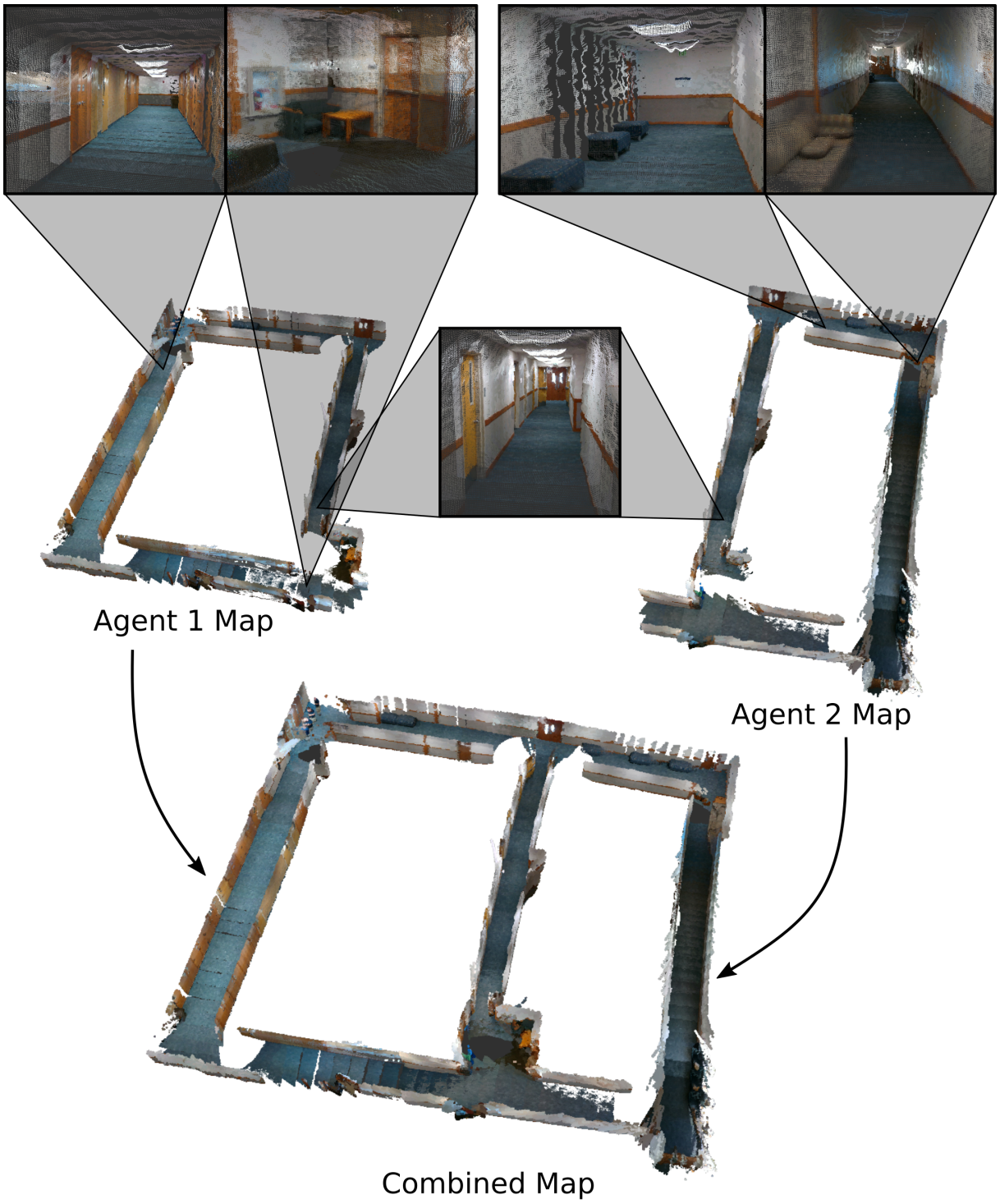

Fig. 7. Example of hardware results of merging maps from two agents into a single map in a simpler hallway environment. Above the individual agent maps are examples of the detail in the pointclouds when zooming in.