Master Thesis

Inter-recognition for collaborative monocular SLAM in UAV swarms

Marina Sofia Garcia Perez

Master Thesis DKE-20-05

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science of Artificial Intelligence
at the Department of Data Science and Knowledge Engineering
of the Maastricht University

Thesis Committee:

Dr. Rico Möckel
Dr. Alexia Briassouli

Maastricht University
Faculty of Science and Engineering
Department of Data Science and Knowledge Engineering

March 15, 2020

Contents

List of Figures

Chapter 1

RELATED WORK

1.1 Multi-UAV Systems

In this thesis we will work with Multi-UAV (Unmanned Aerial Vehicles, Figure 1.1[1]) Systems to perform a collaborative SLAM. This section will discuss briefly the current technology and applications concerning this topic.

Figure 1.1: Ryze Tello UAV example picture. Used in the SwarmLab at Maastricht University.

1.1.1 Technology

Aerial robotics is nowadays a fruitful research area thanks to the flexibility UAVs provide to developers, enabling highly reconfigurable systems with an affinity for artificial intelligence and autonomy. More-so, when combining several UAVs into swarms, it is expected they be more capable than single larger vehicles. These systems are not only adaptable and scalable, adding on the

[1]www.ryzerobotics.com/tello

flexibility single UAVs already presented, but also very robust and with high fault-tolerance [1][2].

We can differentiate different types of Multi-agent Systems based on the collective and individual behaviours:

- **Teams** usually have fewer agents, whose goal is to maximize a local objective. Whether it is optimal for the individuals to cooperate based on, e.g. auctions [3] or compete among each other depends on the specific problem.
- **Formations** can also be formed by a small number of agents, but the characteristic that differentiate it from teams is that each agent has a specific role (usually a placement). This is clear in flocks of birds, for example.
- Finally, **swarms** are formed by many similar (and usually simple) agents, whose local interactions generate a global emergent behaviour.

There are many different ways to model swarms, one of the first used is Reynolds [4] for flocks, which take into consideration neighbours velocity and distance to avoid collisions and obtain a global behaviour. Usually models try to use linear equations defining the state and velocity of different nodes (agents) communicated through edges with weights, defines by a certain communication topology which might be constant or change with time. UAVs themselves tend to be represented as point-masses with velocity dynamics and controlled by the lift, drag and thrust [1].

As a definition, swarms do not use pre-made roads or edges, they need to include some kind of path planning in order to traverse or explore the environment. This is specially tricky since it should avoid collisions not only with the environment, but also with other agents in the swarm. Another consideration to take into account is how to avoid blocking the path of other agents [5] which could greatly increase the time to complete or failing to complete whatsoever. This can be solved in a number of ways: from speed adjustments [6] and sequential trajectory re-planing [7] to more direct approaches like artificial potential fields [8].

Trajectory generation can be thought as an optimal motion planning or boundary value problem, if we decide to generate the trajectories without task allocation. These can be solved with Model Predictive Control [9], or with Decentralized Partially Observable Markov Decision Processes [10]. If we wish to add optimality objectives such as travel distance or battery life, the problem becomes NP-hard and needs to be solved by heuristics [11]. We could decide, on the other hand, to simultaneously plan the trajectory and assign tasks to different agents [5] since if the swarm is homogeneous any agent can be assigned any task, e.g in missile interception. This is more optimally solved in a centralized manner [9], but can be also solved with distributed auctions [12] or decentralized hierarchical strategies [13].

As to the Hardware and Software practicalities, it is important to state that to this day the most extensive research has been done in the realm of simulation as the real-world tests were until very recently not powerful enough or very expensive. Lately, the focus of the investigation has been on increasing the size of the systems and testing them outside the lab environment. In [14] we have a swarm of 50 agents with field experiments. In this case however, the limited battery of the UAVs made the simultaneous flight of all 50 agents only last 10 minutes since power consumption is another major difficulty when working with the hardware.

As to the sensors used for pose estimation, there are two big categories. External sensors such as GPS are used in popular UAV displays like the ones performed by Intel[2] and EHang[3]. Another cheaper but less accurate option is ultra-whideband(UWB). On-board sensors such as cameras, lidar and the inertial measuring units (IMUs) are the second category. This inputs can be then used to feed a Visual inertial odometry [15] or a multi-agent SLAM.

referir a donde lo explique

1.1.2 Applications

Multi-UAV systems have the ability to obtain information simultaneously from different locations, this together with its robustness to errors thanks to the redundancy the system has, makes it ideal for obtaining data about an area and the objects inside it. There are 3 main applications derived from this definition:

- **Target search and track**: In applications such as search and rescue or hostile UAVs. A group of aerial agents can either see different regions resulting in quick global information or see the same region, obtaining quick and reliable information of a small section. Similarly, a high elevation of the vehicle results in a large field of view with a high error.

- **Surveillance and monitoring**: As opposed to the previous application where we had targets, surveillance and monitoring need to cover an area instead of focusing on several points. The difference between surveillance and monitoring is the required frequency in which all areas must be visited, which is not required in surveillance.

- **Mapping**: In mapping we not only need to observe the environment, as in previous applications, but also obtain a representation of it. As of now, only a very small number of UAVs have been able to be tested outside lab-settings because of the technical difficulties it entrails. Since trying to map without knowing the locations inside the map of the agents is not possible, this application is directly linked with Simultaneous Location and Mapping (SLAM), and more specifically collaborative SLAM. Collaborative SLAM is the focus of this thesis and will be further researched in section .

poner seccion

[2] www.intel.com/content/www/us/en/technology-innovation/aerial-technology-light-show.html

[3] www.ehang.com/news/249.html

1.2 SLAM

This thesis main topic is collaborative SLAM, which is a subcategory of SLAM involving several agents. To better understand it, we first need to investigate regular SLAM.

SLAM (Simultaneous Location and Mapping) is an egg-chicken problem, since we need to simultaneously map the environment relative to the agent inputs, and to find the agent position inside that same map. There are multiple applications where SLAM is needed since the perception problem is a main subject in autonomous robotics nowadays. Whenever a map is unknown or we are in a GPS-denied environment, SLAM is usually needed to interact with the surroundings. Some examples could be exploration missions, search and rescue, maintenance and cleaning, etc.

Chapter 2

RESEARCH QUESTIONS

As seen in Chapter 1, the field of collaborative SLAM is quite new and still has room for improvement. In every State of the Art approach to this day, it is needed to fuse maps when two cameras have looked at the same place, taking that place as reference for inter-camera loop closure (map fusion). In this thesis we will research into a different novel situation. Instead of having two agents look at the same place, we will try to have agents identify each other and use this information to improve the quality of the end map.

With this premise we can formulate the following research questions and sub-questions:

- (Main goal) Is it possible to improve the generated map and/or locations of the agents of a collaborative monocular SLAM thanks to adding information on agent inter-recognition?
  - Is it possible to use said information to fuse maps?
  - Is it possible to use it to improve the accuracy of already fused maps?
  - Could formations benefit from this information? Which formations would be best?
- What are the best approach and metrics to evaluate the accuracy of the maps and trajectories calculated?
- Could this be transferable to hardware and real-world applications?

Chapter 3

METHODOLOGY

In this chapter we will discuss the methodology used, i.e. the techniques and technologies used to design and develop our solution.

3.1 CCN-SLAM

3.2 Simulation

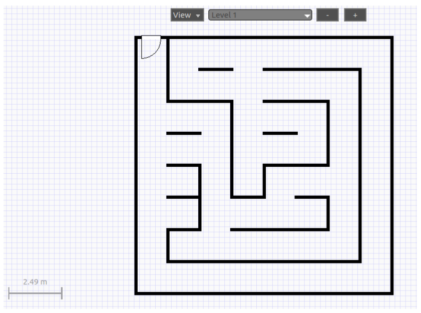
Working with real life robots and environments can be quite challenging. It is first of all expensive since we need to obtain the real hardware needed (UAVs, controls, cameras, measuring sensors, etc), which can total a fair amount even without taking repairs and broken units into account. It is also time expensive, since the setup for the tests, the actual tests and latter cleaning all take time from qualified staff. We should also consider the big penalties we could potentially suffer from in case of an accident, mostly considering the learning environment in which we are working which is more prone to human errors. This could be a minor inconvenience, time or money wise, or result is a complete paralysis of the work. Lastly, working in real scenarios is a challenge in terms of measure. The real map, the real localization of the UAVs and any perturbances are usually unknown, resulting in accuracy measurements based either on qualitative attributes which give little to no information on the quality of the solution, or on approximate measures at best.

Simulation solves all these issues. It does not require expensive hardware except a potent computer, which is recommended either way for latter calculations. A simple and optimized simulation will of course help in this aspect, as will using a simplified graphic interface. This is an aspect we will keep in consideration since removing the GUI will also render the simulation much faster since it will only depend on computation time and not on the visualization. Accidents in simulation are, of course, completely without consequence, which will help us when trying out different new ideas without fear. Lastly, and maybe most important, simulation is fully known. This will allow us to use a proper accuracy measurement to assess our work, and will be useful to simplify measurement problems along the way.

For all these reasons we decided to work on simulation instead of real world. As we will work in ROS environment, we will use Gazebo [1] as simulator, more specifically version 7.0. Gazebo is widely used inside the ROS community for its robustness and efficiency, mostly concerning populations of robots and complex environments, which makes it ideal for our purpose.

3.2.1 Environment

The first thing needed for our simulation was an environment for the UAVs to interact with and ultimately to map. A simple maze was created using Gazebo Building editor (Figure 3.1a) for this purpose. The finished model can be seen in Figure 3.1b.



(a) Gazebo Building editor view of the maze environment created.

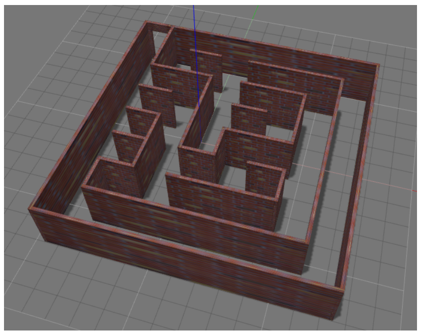

(b) Gazebo maze model in simulation.

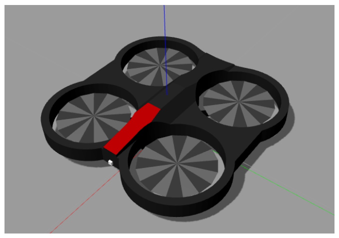
Figure 3.1: Maze environment simulation.

3.2.2 UAV

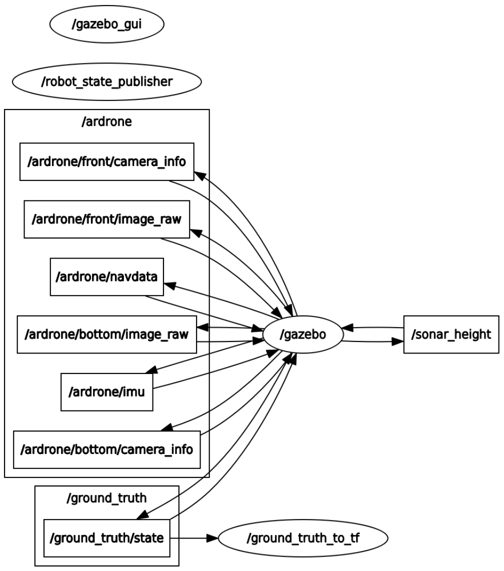
There are many UAV fully controlled models developed for Gazebo, which is one of the advantages this simulator has. In this work we decided to use the TUM Ardrone 2.0 simulation [2], more specifically the ported to Kinetic version[3], which is the ROS version we are working with. The model can be seen in Figure 3.2a.

The controller structure is shown in Figure 3.2b as an rqt_graph showing the ROS nodes as bubbles and the ROS topics as squares. Of special interest are the ardrone topics, which gives us information on the sensors this simulation provides, and the ground truth which could be very helpful for testing.

[1]www.gazebosim.org
[2]www.wiki.ros.org/tum_simulator
[3]www.github.com/angelsantamaria/tum_simulator



(a) TUM Ardrone 2.0 simulation model.



(b) TUM Ardrone 2.0 controller structure.

Figure 3.2: UAV simulation.

The UAVs will not be autonomous, instead we will control them manually. When implementing into the real world our solution, we might decide to add autonomy to the agents depending on the application at hand. However, for our purpose which is testing a SLAM solution, autonomy is not needed and flying the UAVs manually gives more flexibility to our tests.

The tum_simulation wiki gives us information on how to control the UAV. Terminal commands are available but not very convenient. Instead, we will use a PS3 controller connected via USB to fly the UAV according to our desired path. Figure 3.3, provided by the wiki, shows a schematic of the connectivity needed to use a PS3 controller. The USB signal is taken to a Joy node which is in turn used as input for the ardrone joystick controller, which feeds into our Gazebo simulation. In our case, we used ardrone_tutorials[4] instead of ardrone_joystick, since it is implemented for ROS indigo. The final rqt graph generated for the ardrone simulation with a PS3 controller is shown in Figure 3.4.

[4]www.github.com/mikehamer/ardrone_tutorials

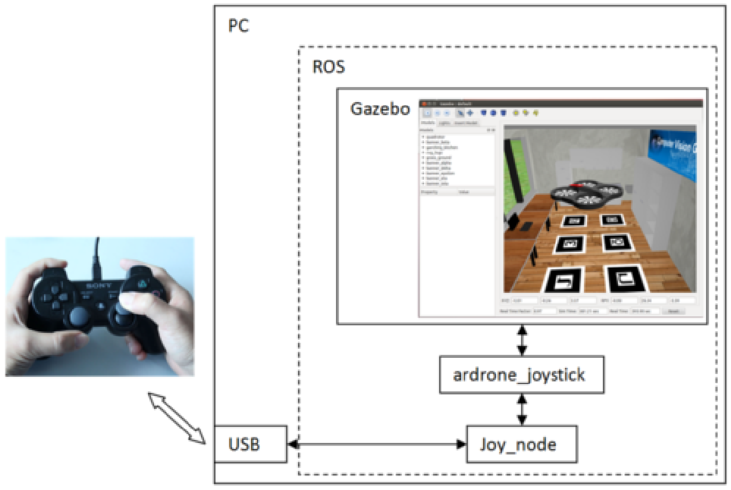


Figure 3.3: Schematic structure for the Ardrone TUM simulation with a PS3 controller.

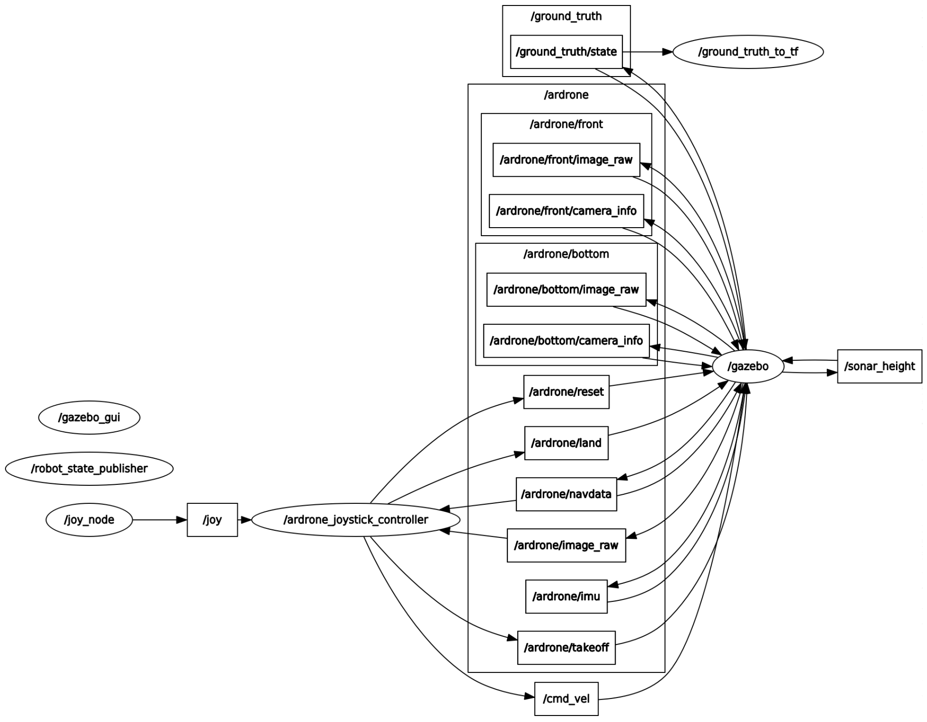


Figure 3.4: TUM Ardrone structure with a PS3 controller connected.

Bibliography

[1] S.-J. Chung, A. Paranjape, P. Dames, S. Shen, and V. Kumar, “A survey on aerial swarm robotics,” *IEEE Transactions on Robotics*, Jul. 2018. DOI: 10.1109/TRO.2018.2857475.

[2] F. Hadaegh, S.-J. Chung, and H. Manohara, “On development of 100-gram-class spacecraft for swarm applications,” *IEEE Systems Journal*, vol. 10, pp. 1–12, Jan. 2014. DOI: 10.1109/JSYST.2014.2327972.

[3] D. P. Bertsekas, “Auction algorithms for network flow problems: A tutorial introduction,” *Computational Optimization and Applications*, vol. 1, pp. 7–66, 1992.

[4] C. Reynolds, “Flocks, herds, and schools: A distributed behavioral model,” *ACM SIGGRAPH Computer Graphics*, vol. 21, pp. 25–34, Jul. 1987. DOI: 10.1145/280811.281008.

[5] M. Turpin, N. Michael, and V. Kumar, “Capt: Concurrent assignment and planning of trajectories for multiple robots,” *International Journal of Robotics Research*, vol. 33, pp. 98–112, Jan. 2014. DOI: 10.1177/0278364913515307.

[6] S. Mehdi, V. Cichella, T. Marinho, and N. Hovakimyan, “Collision avoidance in multi-vehicle cooperative missions using speed adjustment,” English (US), in *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, ser. 2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017, United States: Institute of Electrical and Electronics Engineers Inc., Jan. 2018, pp. 2152–2157. DOI: 10.1109/CDC.2017.8263963.

[7] D. Morgan, S.-J. Chung, and F. Hadaegh, “Model predictive control of swarms of spacecraft using sequential convex programming,” *Journal of Guidance, Control, and Dynamics*, vol. 37, pp. 1–16, Apr. 2014. DOI: 10.2514/1.G000218.

[8] M. Wolf and J. Burdick, “Artificial potential functions for highway driving with collision avoidance,” Jun. 2008, pp. 3731–3736. DOI: 10.1109/ROBOT.2008.4543783.

[9] D. Morgan, G. Subramanian, S.-J. Chung, and F. Hadaegh, “Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming,” *The International Journal of Robotics Research*, vol. 35, Feb. 2016. DOI: 10.1177/0278364916632065.

[10] S. Omidshafiei, A.-a. Agha-mohammadi, C. Amato, and J. How, “Decentralized control of partially observable markov decision processes using belief space macro-actions,” Feb. 2015.

[11] J. Yu and S. LaValle, “Optimal multi-robot path planning on graphs: Complete algorithms and effective heuristics,” Jul. 2015.

[12] M. Zavlanos, L. Spesivtsev, and G. Pappas, “A distributed auction algorithm for the assignment problem,” Dec. 2008.

[13] J. Yu, S.-J. Chung, and P. Voulgaris, “Target assignment in robotic networks: Distance optimality guarantees and hierarchical strategies,” *IEEE Transactions on Automatic Control*, vol. 60, Aug. 2013. DOI: 10.1109/TAC.2014.2344291.

[14] T. Chung, M. Clement, M. Day, K. Jones, D. Davis, and M. Jones, “Live-fly, large-scale field experimentation for large numbers of fixed-wing uavs,” May 2016, pp. 1255–1262. DOI: 10.1109/ICRA.2016.7487257.

[15] A. Weinstein, A. Cho, G. Loianno, and V. Kumar, “Vio-swarm: A swarm of vision based quadrotors,” *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, Jan. 2018. DOI: 10.1109/LRA.2018.2800119.