Platform for Communication and Coordination of Agents using the MAVLink Protocol

Ramon Soares de Melo and Arturo César Alvarez Cea
Universidade Federal de Minas Gerais, University of Santiago de Chile
ramonmelo@dcc.ufmg.br, arturo.alvarez@usach.cl

I. INTRODUCTION

The use of autonomous robot systems can be seen in many scenarios, besides the industrial use, where were first employed, mainly on object manipulation and building of items in assembly lines ([1]), is notorious the use of these autonomous systems in other activities like surveillance and regions recognition ([2], [3]), search and rescue ([4], [5]), mapping ([6]), detection and tracking ([7]), transport of objects ([8], [9]), among others.

All aforementioned activities can be executed both individually as well collaboratively, in this case called Multi-Robot Systems (MRS). The use of multiple robots presents several advantages like increased robustness and, in most cases, time reduction to accomplish a task. However, the use of such systems also brings many challenges, such as robot localization, path planning, task allocation and control.

The accomplishment of any task using a MRS involves many subproblems that must be considered. Among these problems, we can highlight the localization of the robots, path planning, coordination and the task allocation. The localization problem is related with agent itself as well the environment where the robots are working and any other object that they interact. The path planning problem can be addressed in many ways, in some cases to minimize the total time spent or the overall traveled distance for example. Regarding the team coordination, that involves task allocation among the agents, as well a study of how combine the available resources to accomplish the activity respecting all possible restrictions.

Each agent in a MRS have its own capabilities and resources, that can be shared among the other agents of the team. These resources can be the locomotion type of the vehicle, a set of sensors, or the payload among others. A team of agents should be able to share and explore these resources to complete satisfactorily one particular mission.

One of the platforms that has come into common use for the creation of autonomous agents is called ArduPilot (APM), whose main goal is to be an autopilot system to carry out autonomous missions, responsible for low-level control of different kinds of vehicles, such as the terrestrial ackerman model or flying with multiple rotor and fixed wing.

Similar to other platforms for autonomous control, the APM system uses the Micro Air Vehicle Communication Protocol (MAVLink) that describes a specification for data exchange between the vehicle and a base station. It's able to transfer telemetry information such as data from sensors (GPS, barometer, magnetometer) or send missions to the agent.

Using this toolkit, a vehicle controlled by an auto pilot and having communication with a base station, can perform activities like scanning and capturing images of a particular area, carrying out autonomous missions in a consistent and easily configurable way.

The project presented here was developed during the interchange period occurred at the University of Santiago de Chile (USACH) between April and May of 2015, together with the tutor Arturo César Alvarez Cea, which has developed a base system for communication and control agents using MAVLink protocol.

II. METHODOLOGY

The main objective of this work was the creation of a base system for communication and exchange of information between autonomous agents through MAVLink protocol. Agents must be able to share informations such as your current position, your current status (whether they are on a mission or not), what mission they are in, as well as having the ability to send missions to other agents.

The project was based on framework ROS (Robot Operating System), which provides a platform for research and development of robotic systems, possessing abstraction layers for communication, visualization and others. As control platform, the autopilot ArduPilot had been used in two different kind vehicles, a land based (figure 1a) and aerial (figure 1b). Both have global positioning sensor (GPS), altitude, and communication via radio, which can send and receive data from a base station.

A. MAVLink Protocol

The MAVLink protocol was constructed to serve as common language between a ground control station and a autonomous robot. It shows how to encode and decode data using a header-only message, describing a set of message types used to identify the message content.

A MAVLink message consists of a package with a minimum of 8 bytes and 263 bytes as maximum. As can be seen in the figure 2, the package is divided into several slots, each one with a specific description. The table I list and describe all slots of a MAVLink package.

The protocol can handle various types of data, having support to fixed-size integer data, floats with single precision

(a) Rover

(b) Quadcopter

Fig. 1: Agents used on experiments

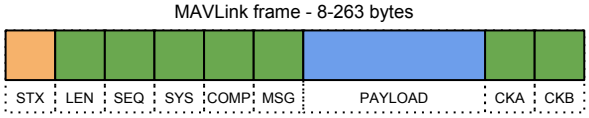


Fig. 2: MAVLink Package description

Slot	Description
STX	Indicate a new package
LEN	Indicate the total length of the payload
SEQ	Package sequence
SYS	ID of the sending MAV system
COMP	ID of the sending component inside the MAV
MSG	ID of the message describing what type of message will be in the payload
PAYLOAD	Data of the message
CKA	Checksum A (low byte)
CKB	Checksum B (high byte)

TABLE I: MAVLink Package slot description

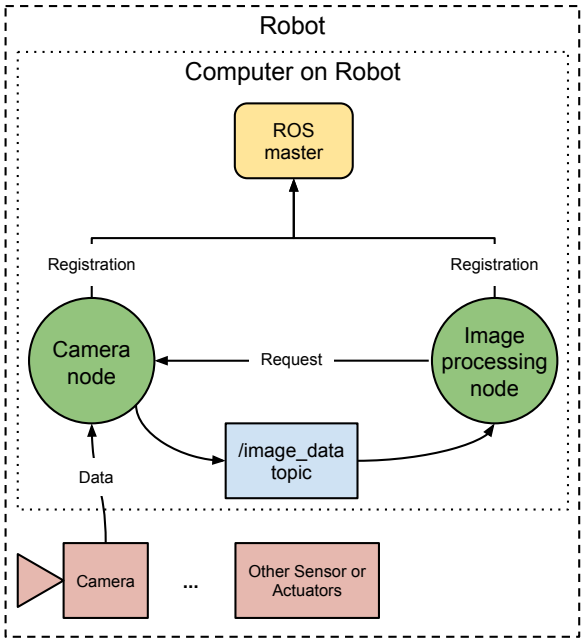


Fig. 3: ROS Environment

point, and array of these types. The supported types are: *char, uint8, int8, uint16, int16, uint32, int32, uint64, int64, float, double*.

The main objective of the protocol was the be fast and safety, but there are some drawbacks, as the need of extra bytes to track the messages, but it can detect the lost of packages and check the content.

B. ROS Communication System

The ROS environment was created aiming the decoupling between any part of the system, creating a system of communication that do not link directly two parts of the code. To create this environment, two main concepts were created, the nodes and the topics. Each node is a piece of code that executes a determined behavior, like control the velocity of the motors of a car or capture images from a camera. Each topic is a address where the data is read and write by the nodes using the actions of subscribing and/or publishing to that topic. This guarantee the possibility of the code run in different computers, like using a team of robots, each one with it individual computer for example.

The figure 3 show a representation of the ROS environment, where there are many nodes exchanging information using the topics. One important note about that type of communication is about the registration of the nodes and the initial meeting among then, that is made by the called *master node*, that waits for the initialization of any node, and any request to publish or subscribe to one topic and make all the necessary connections.

C. Project Architecture

In general, the architecture for controlling a robot using the ArduPilot platform is made by a base station which communicates with the agent via the MAVLink protocol,

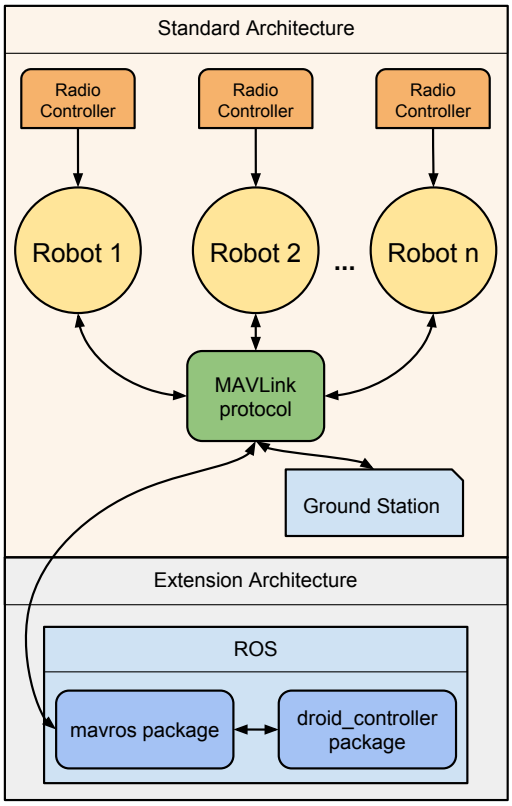


Fig. 4: System Architecture

sending commands and tasks. The onboard controller is responsible to receive and threat theses messages, making the necessary decisions to accomplish the sent objectives. However, this strategy only guarantees the control of one agent at a time, and does not allow the exchange of information between an agent team. The work developed here proposes an extension to this architecture in order to provide a communication and intelligence layer for agents, serving as basis for the development of more complex tasks that require the use of a set of robots.

Figure 4 shows the layout of the developed architecture, marking the addition in the context of communication in the pre-existing control layer. The manner in which this strategy was set aims to promote transparency about the origin of the commands received by the agents, because similar to messages sent by the base station, the system developed also communicates through MAVLink commands, which makes interaction with agents direct and simple.

The MAVLink protocol is a standard regarding communication with autonomous systems, in this way there are several libraries that abstract the low-level details in communication, providing an interface for sending and receiving data transparently. One of these libraries, implemented within the context of the ROS is the *mavros* package. This package continue in development by a group of nonprofit developers who created the tool because they see the opportunity of using these systems as autonomous intelligent agents.

The *mavros* package is able to encode and decode messages in MAVLink, extract information and then format them into ROS messages, making the communication with a ArduPilot direct to the ROS infrastructure, providing easy access to other packages for these information.

Based on this structure, was created the package *Droid Controller*, which contains a set of classes and algorithms that perform the reading of information from the agents and performs processing and exchange of information between them.

D. Droid Controller Package

The *Droid* package stacks over the *mavros* package using it connection and MAVLink message handler. The main goal was to create a set of classes that listen, store and share data from the agents and also create a interaction interface where is possible to request this data, send commands or missions. This package should serve as basis for the creation of complex missions using the shared information between the robots.

The following are the main classes of the *Droid* package:

Droid
: The Droid class is the base class to any specialized vehicle. It retains all important data about the robot like its current position, status of motors, MAVLink mode, current mission, battery status and others. Also export functions to get/set configuration parameters, send mission or arm the robot, among others.

DroidCopter
: Specific class for a aerial robot, like a quadrotor. It uses and extends the Droid class, adding behaviors to return the craft to ground based on the level of remaining battery, and perform the initialization sequence for a mission.

DroidRover
: Class that control a terrestrial robot, like a rover. Extends the Droid class to perceive for obstacles while a mission, requesting for another agent to continue the mission if it's not possible to complete it.

As a example of integration and exchange of information, the Droid class implements a follow behavior, where a agent is capable of maintain a certain distance from another based on its current global position and the position of another robot. This behavior can be extended to create a squad behavior using various robots.

III. EXPERIMENTS

The created architecture was tested in two environments: a simulation and a real. The real agents are as shown the figure 1. The primary objective was to prove that the connection and exchange of information occurs between the agents, in such way that each agent know about the others and can perform actions based on this knowledge.

To proof the *Droid* package, two simulation tests was performed: (i) the following mode, where a agent must follow and maintain a certain distance from another target agent,

and (ii) the obstacle mode, where a terrestrial agent can ask for help to a aerial agent if it detects that is not possible to complete your mission.

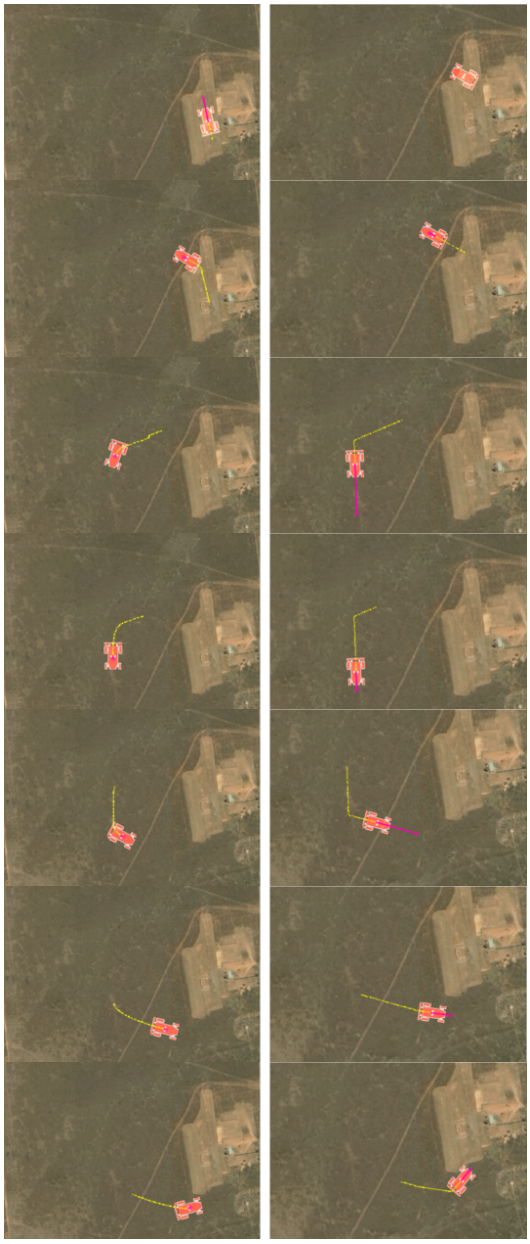


Fig. 5: Two rovers in simulation, the right rover executes his mission while the left rover follows it.

The following mode test is demonstrated in the figure 5 (simulation environment) and 7 (real experiment), where one robot is executing a determined mission, but sharing its current position the other agents in the team, and another robot use this information to follow that agent, but keeping a configurable safety distance.

The test shown in the figure 6 demonstrate the behavior where a agent founds a obstacle which prevents your mission conclusion, and then ask to another agent to finish it, sending the remaining mission and returning the home.

These tests were developed to demonstrate that the architecture extension works to communicate and take decisions based on data sent by the agents, performing actions using the resources available for the team of robots.

Fig. 7: Real experiment showing the follow mode using a rover vehicle and a quadrotor drone.

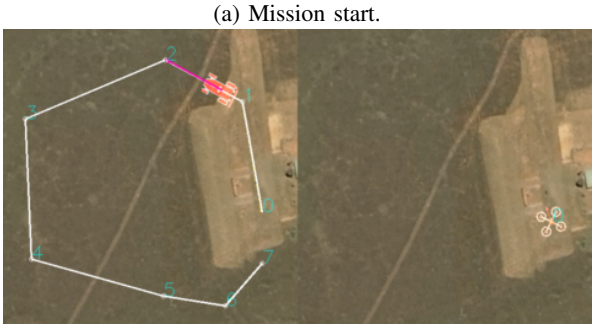
IV. CONCLUSIONS AND FUTURE WORK

In this work was presented a extension of the standard architecture of control of a autonomous agent using the ArduPilot autopilot controller communicating using the MAVLink protocol that guarantees communication and information exchange between a team of robots. The experiments showed the capabilities of the extension, using the available data to take control of the agents.

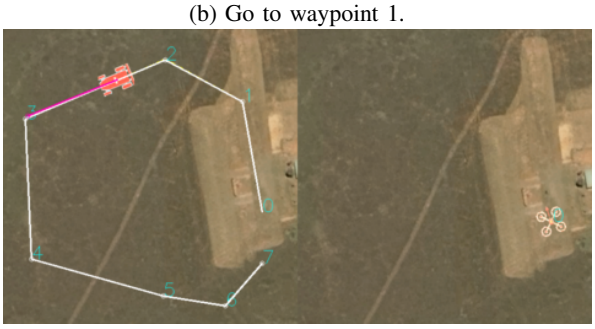
Future directions include the addition of a squad control, where multiple agents can perform missions of surveillance or search, exchanging data about the supervised area among then and with a land base. Another improvement is to add

(a) Mission start.

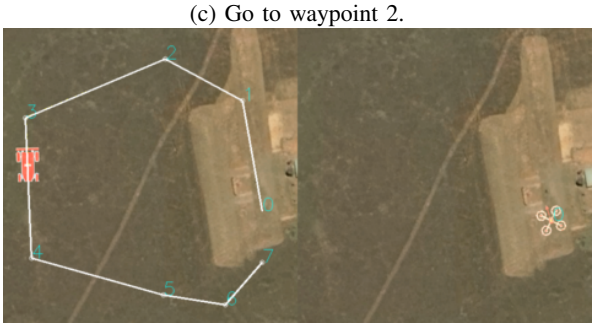
(b) Go to waypoint 1.



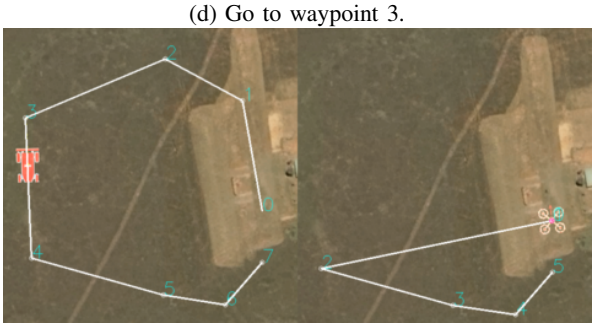
(c) Go to waypoint 2.



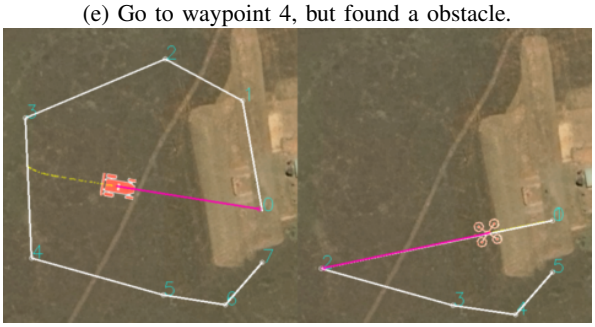
(d) Go to waypoint 3.



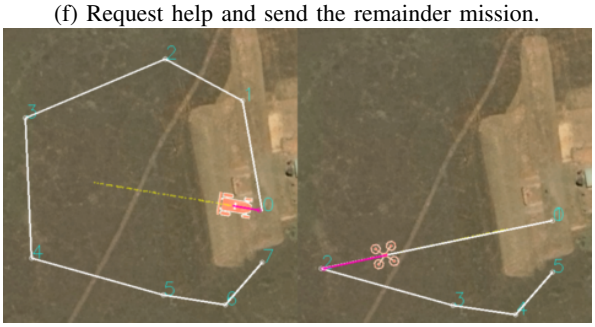
(e) Go to waypoint 4, but found a obstacle.



(f) Request help and send the remainder mission.

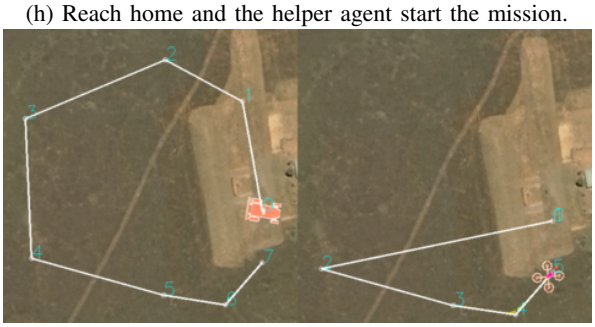


(g) Return to home while other agent take the mission.



(h) Reach home and the helper agent start the mission.

(i) The agent continue the mission.



(j) Mission complete by the other agent.

Fig. 6: Mission execution by the rover agent, that founds a obstacles and ask for help to another agent.

support for other types of robots and uses the individual abilities to improve the team.

V. Acknowledgment

I would like to thank the opportunity to have participated in research at the University of Santiago de Chile, to teachers Arturo Alvarez Cea and Gonzalo Acuna, by the knowledge exchange, and SISAR Ltda (www.sisar.cl) for providing the robots used during the experiments. They were moments of learning and exchange of knowledge that have been and will be shared by both sides in future research to come.

References

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge, MA, USA: MIT Press, 2011.

[2] H. Tanner, "Switched uav-ugv cooperation scheme for target detection," 2007, pp. 3457–3462.

[3] P. Sujit and S. Saripalli, "An empirical evaluation of co-ordination strategies for an auv and uav," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 70, no. 1-4, pp. 373–384, 2013.

[4] J. Casper and R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, no. 3, pp. 367–385, 2003.

[5] R. Murphy, "Human-robot interaction in rescue robotics," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 34, no. 2, pp. 138–153, 2004.

[6] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 5321–5326.

[7] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 16–26, 2006, cited By (since 1996)84.

[8] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, vol. 30, no. 1, pp. 73–86, 2011.

[9] J. Fink, M. Ani Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles," 2008, pp. 1471–1476.