Proposed Testbed for the Modeling and Control of a System of Autonomous Vehicles*

Joaquin D. Labrado, Berat A. Erol, Jacqueline Ortiz, Patrick Benavidez, and Mo Jamshidi
Department of Electrical and Computer Engineering
The University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA
Email: jdlabrado@gmail.com, berat.erol@utsa.edu, jaqdandori@gmail.com, patrick.benavidez@utsa.edu, moj@wacong.org

Benjamin Champion
School of Engineering
Deakin University
Waurn Ponds, VIC, AUS
Email: btch@deakin.edu.au

***Abstract*—Large scale multi-agent systems are very important in todays world because of their varying uses. The Center for Testing, Evaluation and Control of Heterogeneous Large scale Autonomous Vehicles (TECHLAV) has been tasked to conduct research on Large Scale Autonomous Systems of Vehicles (LSASV). This paper focuses on the proposed testbed system that will help model the large scale system out in the field for Modeling, Analysis and Control tasks for LSASV (MACLAV). The system will use a team of UGVs, UAVs and AUVs to navigate, interact and complete missions through an unknown area as a cohesive unit. A small private cloud provides a computational backbone to the system.**

***Index Terms*—Testbed, Modeling, Cooperative SLAM, vSLAM, Robotics, RGB-D, Cloud Computing**

I. Introduction

The field of robotics is a very broad area of research that many different people and centers are researching. One such center is the Center for Testing, Evaluation and Control of Heterogeneous Large scale Autonomous Vehicles (TECHLAV) which focuses on Large Scale Autonomous Systems of Vehicles (LSASV). LSASV include UGV's, UAV's and AUVs that work as a team to accomplish a specific mission.The main focus of the center is to develop techniques to deploy LSASV in uncertain or dynamic environments through effective interactions with human operators. The objective of this paper is to propose a framework for Modeling, Analysis and Control of LSASVs (MACLAV). The two main tasks associated with the MACLAV testbed are the modeling and analysis of LSASVs, and the cooperative localization, navigation and control of LSASVs. These main tasks are split into several subtasks including mathematical modeling, data-driven modeling, analysis, localization, navigation and control of LSASVs. Each task requires different resources, but each has a bottom line requiring access to computational power and multiple networked robotic platforms.

Multi-robot operations are an area that has been explored in the past, such as in the RoboLeader project [1] which was done to help the Army find IEDs out in the field. While these robots are operated by humans, the systems still communicate with other mobile robots to better identify targets in the field. Other places where multi-robot system have been used have been in disaster scenarios such as in [2], and [3] which show designs for robots to work as a team to accomplish a goal in a dynamic environment. There has also been research in the past that has dealt with both UAS's and UGV working as a team [4], and where UAS's and UGV worked together in a ROS environment in [5]. This paper used a UAS to land and charge its battery on a UGV platform which is a valuable skill for any system that utilizes aerial and ground robots. All these papers present multi-robot testbeds designed for much smaller systems that had fewer complications from processing and communications.

Cloud computing will be used to provide a computational backbone to robots in the system. Generated system models will require validation against captured experimental data. Localization and navigational tasks will require large datasets to develop and optimize the algorithms. Control tasks will require simulation and experimentation to ensure the controller functionality is optimal. The Robot Operating System (ROS) will be used for control and management of robots in the testbed. ROS will be used to acquire datasets, control robots, provide interfaces between device drivers, and provide the messaging platform to compute nodes on the cloud.

There are some limited examples in the literature and projects on multi-robot ROS. DAVinCI, a cloud computing framework for robot SLAM applications is proposed in [6], which proposed methods for servers to process data coming from robotic agents. It is unclear as to whether the system was tested for more than one or two robots, or even in a real-time application. A more promising approach is found in ROS 2.0 [7], where a working group has devoted significant effort at tackling the problem of creating a multiple robot, real-time version of ROS. Advances of ROS 2.0 should provide significant performance increases in the near future for ROS based systems.

The rest of this paper is structured as follows. Section II covers applications of the testbed for the tasks of MACLAV. Section III details the proposed testbed configuration. Section IV details some early validations of the testbed. Section V covers the conclusions and future work with the testbed.

* This work was supported by Grant number FA8750-15-2-0116 from Air Force Research Laboratory and OSD through a contract with North Carolina Agricultural and Technical State University.

II. Testbed Applications

Use cases for the testbed will be presented in this section. Requirements for each application will be noted as well.

A. Cloud Based Control of LSASVs

Robotic studies or their applications that involves swarm of system, or a cooperative work task with multiple agents to accomplish the the system's ultimate goal, require data sharing among the system's agents. Alternatively, a master agent that deployed into the system would be required, which is able to collect and process the gathered data from and through the other agents, a large scale system. Our testbed has been built on such operation that aims to deal these concerns, and to create a centralized computing, processing and storing unit for all created data for entire system. It is important to recall that the system is heterogeneous, a combinations of different agents and platforms, in other words, a system of systems (SOS). Therefore, this kind of research experiments creates humongous amount of data, deals with storage problems, and most importantly requires high performance computing power for processing the data to find a meaningful outcome. We proposed a system of system that will ease these concerns with a cloud computing back-end.

Therefore, we believe in that adding the Cloud computing will bring priceless advantages into any application of a SOS. For example, our system will be able to handle all individual work tasks for LSASVs easily and results will be created and processed faster. Moreover, it will include large storage and memory capacity for storing and processing the data to work on data analytic processes, real time system simulations, cooperative control of system agents, and vSLAM experiments.

B. Modeling of LSASVs

In MACLAV, there are studies into both pure mathematical and Big Data Analytic approaches to modeling LSASVs from collected experimental data. In the Big Data Analytic approaches, the main bottleneck in performing much of the research has been the available memory and computing power in commercially available personal computers. Open source software and use of cloud computing can greatly advance the efficiency of research performed in this area. As methods are being developed for data driven approaches to modeling of LSASVs, the computational and storage requirements may change over time.

C. Cooperative VSLAM

This objective will be split into smaller objectives, that all are vastly important in order to achieve the overall main goal. Portions of the overall process are listed below:

1) Navigate an unknown terrain without a map data and create a map on the fly.
2) Reuse the created map to navigate to objective markers placed by a human controller
3) Search for distinct targets hidden in the known world.
4) UAV eye in the sky helping the UGV navigate the saved map
5) Merging maps from the UGV and UAV

The main objective of VSLAM will be to create a map of an unknown location for navigation of the area at some future point in time. Detection of features in an environment can be performed via QR tags, feature detectors in OpenCV, rg-chromaticity, FAST SURF [8], or a combination of some or all of the methods. Existence and persistence of usable features will be two of the main problems encountered in this task. Optimization and selection of the features to be stored in memory, and later to the disk, are computationally expensive operations. Without corrections to saved map data, matches to incorrect positions will cause incorrect localization data to be injected into the system.

Once the system has a completed map of its environment, the next step is to mark locations in the map for the UGV or UAV to navigate around. This step can be another computationally intensive operation that runs concurrently with feature detection and map building. Detection of where current data matches into the saved map requires not only search mechanisms, but also some kinematic modeling of the agents acquiring the image data. Here our past research into cloud-based VSLAM will be utilized in order to speed up the processing time.

Once new images can be mapped into the saved maps, tests of autonomous search behavior can be performed to create trajectories for the robots to take through the environment. This will set a baseline for all vSLAM experiments as it is the main idea behind the overall objective. Once we have a vSLAM algorithm implemented across all UGVs and UAVs the next objective is to have them find and identify the exact same target by using the same vSLAM map. This objective is not the same as the previous one because the UAV and UGV will work together to create a new map. They will operate as if they are one system and not two independent systems. This will mean that the newly created map will contain significantly more RGB-D data and features. Again, the pruning algorithm will have to be able to know what data to throw away and what data is important for retention.

Lastly, the final objective will ensure that the overall objective is accomplished by having a UAS and a UGV fully merge both of their independently created map. After all the step objectives are accomplished then the full large scale system will consist of two UAVs and two UGVs working as a team to map an environment and coordinate their movements. Each of these objectives require not only large amounts of parallel computation, but also significant network bandwidth.

III. Proposed Testbed

The robotic portion of the proposed test bed consists of Kobuki Turtlebot2s (UGV), Parrot Bebop Drones (UAV) and Erle-Robotics Erle-Copters (UAV). A private cloud system provides the computational backbone of the system.

A. Computing and Networking

1) Private Cloud "ACE Fog": A ten server (Facebook's OpenCompute V1.0 spec), private cloud running Openstack Open Source Cloud Software is installed in the Autonomous Controls Engineering (ACE) Field Lab, which we have named *ACE Fog*. One of the ten servers provides control services to the cluster, which includes authentication, image creation and storage, management and networking services. The remaining nine servers provide a bank of storage and computational power where the virtual machines reside. Figure 1 shows a single server and a collection of servers in a wood rack.

(a) Facebook OpenCompute V1.0 Server

(b) Server Rack

Fig. 1: "ACE Fog" private cloud installed in UTSA's ACE Field Lab

a) Configuration Specifications: The current configuration has the capacity for 216 VCPUs that can consume up to 493.5GB of RAM and 40.1TB of disk space. These figures account for overhead of the host server OS on the physical servers. The servers are all connected to a TrendNet TEG-S16DG gigabit ethernet switch.

b) Installation Method: Installation of OpenStack on the servers was accomplished via OpenStack Configurator for Academic Research (OSCAR) [9], which prepares Openstack-Ansible [10] for use on small clusters.

2) Network: In a system of robots requiring the processing of visual data, there needs to be a capable network fabric to support them. A Linksys AC3200 TriBand WiFi router was acquired for the wireless link to the mobile robots. This tri-band router provides 3.2Gbps total throughput on the wireless links. *ACE Fog* connects into the router via a wired connection with the TrendNet gigabit ethernet switch. Virtual compute nodes on *ACE Fog* can connect to the building network via software-based routing tools in OpenStack.

3) Software: ROS is used as an integrating tool in our testbed configuration. Communication between processes is handled via ROS messages. Distributed processing is currently being performed using ROS in a method similar in functionality to that of MPI. The *multimaster_fkie* ROS package [11] is used to synchronize ROS topics across multiple machines. Namespaces are used to isolate individual robot's topics across all synchronized nodes in the system. For namespacing, a script on each robot computer sets the specific namespace via the *ROS_NAMESPACE* system environment variable [12] before *multimaster_fkie* and the appropriate driver nodes are started.

B. Robotic Agents

Each vehicle is configured with the necessary hardware to accomplish the tasks for MACLAV. Software differences between MACLAV tasks are possible on the Kobuki Turtlebot2 and Erle-Copter systems as each are configured with a general purpose ARM-based processor with open source control software. Figure 2 shows the proposed vehicles and their assigned roles in a mapping operation.

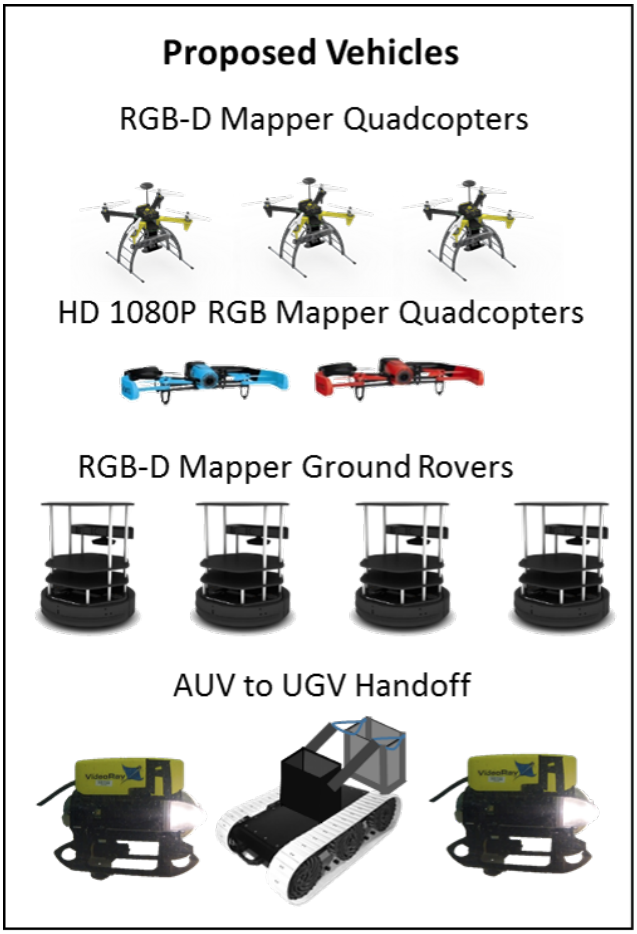


Fig. 2: Proposed Testbed Vehicles – Erle-Copters as the "RGB-D Mapper Quadcopters", Parrot Bebop UAVs as the "HD 1080P RGB Mapper Quadcopters" and Kobuki Turtlebot2s as the "RGB-D Mapper Ground Rovers"

1) UGV – Kobuki TurtleBot2: For the land rovers, the Turtlebot2 was selected due to its customizable capability and open source software. The rover is equipped with a Yujin Robot Kobuki base, a 14.8V Lithium-Ion battery, and a Hardkernel ODROID XU4 minicomputer. In addition, the rovers come with cliff sensors (left, center, right), wheel drop sensors (left, right), a single axis gyro and motor overload protection.

a) ARM-based Embedded Computer: The ODROID XU4 minicomputer was selected as the embedded computer for the UGVs. It features a Samsung Exynos 5422 octa-core CPU, 2GB DDR3 RAM, USB 2.0/3.0 and a 64GB eMMC card for storage. A Meanwell DC-DC converter was connected to the Kobuki's 12V 5A output to supply power for the ODROID XU4 (5V/4A requirement).

b) Inertial Measurement Unit: Included with the base Kobuki platform is a single axis gyro sensor to obtain a heading angle relative to the starting position of the robot when it is powered on. To obtain better directional awareness, a BOSCH BNO055 Inertial Measuring Unit was added to provide absolute orientation to the system. This IMU integrates multiple sensors to obtain a stable absolute output: a triaxial 14-bit accelerometer, a triaxial 16-bit gyroscope and a triaxial magnetometer.

c) Forward Facing RGB-D Camera: An ASUS Xtion Pro Live RGB-D camera is installed on the TurtleBot2 for color, infrared and depth sensing. The camera is placed in a fixed position near the center of mass of the vehicle.

2) UAV – Parrot Bebop Drone: The Parrot Bebop Drone is a quadcopter designed to be controlled via IEEE 802.11 b/g/n/ac Wi-Fi with an Android or iOS device. Onboard the drone there is a dual-core custom Parrot CPU with a quad-core GPU and 8GB of flash memory. It runs on Linux thus it can be controlled by ROS.

a) Sensors: It also comes with the following sensors: magnetometer, gyroscope, 3-axes accelerometer, ultrasound sensor and pressure sensor.

b) Customization: No additional hardware has been added. Modifications are made to the host network settings to enable the drone to connect to an existing network, rather than an ad-hoc connection.

3) UAV – Customized Erle-Copter: The other quadcopter, the Erle-Copter from Erle Robotics, is Linux based with ROS support. It utilizes the Erle-Brain2 hardware autopilot and the AMP flight stack.

a) Inertial Measurement Unit: The Erle-Copter computational platform, the Erle-Brain2, provides a combination of a Raspberry Pi2 (RPI2) and an APM compatible autopilot board that mates to the RPI2 GPIO expansion connector.

b) Bottom Facing RGB Camera: A 60fps Raspberry Pi Camera was included with the Erle-Brain2 autopilot. The camera connects directly into a dedicated port on the RPI2. Originally it was positioned in a forward facing direction directly above the USB connections. With the number of bulky USB connections necessary in our configuration, the camera needed to be repositioned away from the ports. The camera was moved to the bottom of the drone to assist in better controlling the planar drift of the quadcopter.

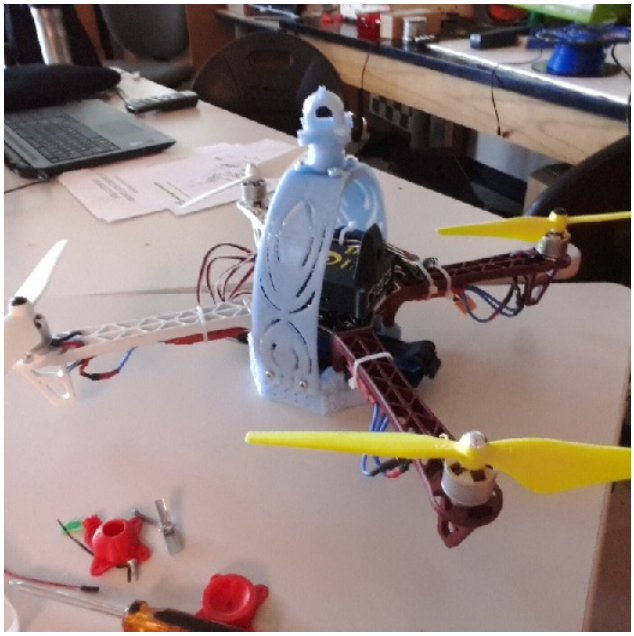
c) Front Facing RGB-D Camera: An ASUS Xtion Pro Live RGB-D camera is installed on the Erle-Copter for color, infrared and depth sensing. The camera is placed in a fixed position along the center of mass axis, at a point on top of the quadcopter.

d) Hardware Customizations: The Erle-Copter must be reconfigured out of the box in order to fly it. Some components are not well mated as received from Erle Robotics. Landing gear legs did not reach the ground with the selection of a larger lithium polymer battery. The built-in camera was positioned in a poor location given the close proximity to the USB connections, where a USB WiFi dongle with a long antenna is to be placed. Additional clearance on the underbelly of the drone is provided by 3D printed landing gear leg extensions. The 60fps camera is moved to the bottom of the quadcopter to provide visual feedback from under the quadcopter. Cable extensions for the USB connections were purchased to allow components to be positioned in more optimal positions.

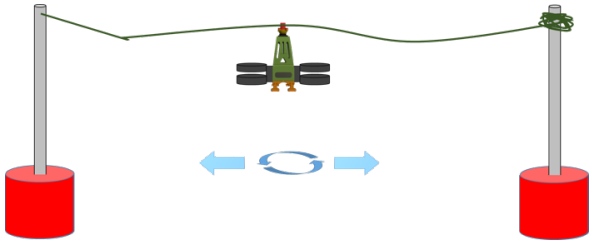
e) Safety Hardware: A 3D printed frame was added to the quadcopter in order to attach it to a tether or a long pole. The connection between the frame and the safety device (tether/pole) is provided via a low friction 3D printed ball and socket joint. Figure 3 shows an Erle-Copter configured with tethering hardware.

4) AUV – Customized VideoRay Pro3: Underwater robotics has become a growing field in recent years. These types of systems have many different applications, such as the collection of underwater resources, mine removal, etc. To be able to accommodate for this medium, the VideoRay Pro 3 robots have been added into the swarm. The VideoRay Pro 3 robot is a commercial robot produced by VideoRay and contains an internal compass, depth sensor, external lights as well as a black and white fixed rear facing camera and a tilt enabled color front-facing camera.

Like most robots that operate within the underwater domain, communication to the units is a challenging problem. Currently, most underwater robots require a tether to be able to communicate back to the surface. Other commercial options are either not suitable for many testing environments, such as a pool, as well as the bandwidth being available being too small for many applications, such as streaming vision information. This is due to high frequency radio waves, these being used by most common forms of wireless communication, being absorbed by water. This is compounded when swarm robotics is considered as each underwater robot would require the use of most, if not all, of the entire bandwidth to communicate at a single point in time, therefore preventing two or more robots from communicating simultaneously. Due this this, the VideoRay robots communicate back to a computer located on the surface via a tether. This computer also contains all of the Artificial Intelligence for the systems, as this cannot be directly uploaded into the robots. This allows for relatively fast communication between the units to be achieved, as well as enabling the robots to be able to communicate directly with



(a) Erle-Copter with Safety Hardware



(b) Tethering hardware

Fig. 3: Erle-Copter with tethering safety hardware for flying outdoors (per campus safety department restrictions)

the outside world, such as to the other robots in the swarm as well as to the cloud, as the wireless communication preformed by these systems can be routed back through the computer.

To increase the sensing functionality of the robots, an attachment to these robots have been implemented containing a microcontroller, an IMU and a RS485 to TTL converter. This allows for the significantly more accurate roll, pitch and yaw data to be transferred back to the surface, thus allowing for more complicated autonomous procedures to be preformed. Additionally an arm has been added to the base of the robot to allow the system to be able to collected objects located on the pool floor.

5) UGV – MMP40 Tracked Vehicle: A tracked robot has also been included into the swarm. This tracked robot contains an odroid, enabling it to run ROS, an IMU and encoders to enable localization as well as a kenict sensor. The main purpose of this robot is to work in tandem with the underwater robots. The MPU will be able to drive up-to the edge of a pool and stop, without entering the water. A basket, attached to the top of the robot, is then able to be lowered into the pool to allow the transfer of object from the underwater robots to the land rover. The localization issue with the underwater robots can also be partially solved using this method, as during the transfer stage the position of the underwater robots can be reset based of the more accurate position of the land rover. []

IV. Early Validations of Testbed Components

Several proof-of-concept projects have been completed to develop this testbed. In one project, multiple ArDrone 2.0 quadcopters were controlled using the *tum_ardrone* ROS package to fly in a formation. An XBox Kinect camera placed in a fixed position was used as an observer for providing location data of the quadcopters to the controller. The camera detected pose of the quadcopters using tags from the *ar_track_alvar* ROS package [13]. Another project involved use of the *ar_track_alvar* ROS package for positional data on the location of UGV from the perspective of a UAV in [14]. The UAV was controlled to land on the navigational tag using a fuzzy logic controller to control its descent. In an unpublished work, we have also demonstrated the capability of using a cloud node to directly process visual data from UGVs. In [15], we have demonstrated a cloud-based VSLAM algorithm. A more recent project has been used to test the scalability of ROS multiple master software packages, where we decided on *multimaster_fkie* as being the most complete project available. Each of these projects have experimentally proven the usefulness and capability of the testbed that we have developed for MACLAV.

V. Conclusions and Future Work

Each of the robotic systems mentioned in Section III have been configured to meet the needs of the tasks mentioned in this paper. Future modifications to the robots will be supported via platform modularity, 3D printing capabilities and software based expansion. *ACE Fog* is being configured by users to meet the needs of their tasks. Higher level controllers that manage LSASV level dynamics are being developed to be run on the private cloud *ACE Fog*.

References

[1] M. G. Snyder, Z. Qu, J. Y. Chen, and M. J. Barnes, "Roboleader for reconnaissance by a team of robotic vehicles," in *Collaborative Technologies and Systems (CTS), 2010 International Symposium on*. IEEE, 2010, pp. 522–530.

[2] M. Gansari and C. Buiu, "Building a slam capable heterogeneous multi-robot system with a kinect sensor," in *Electronics, Computers and Artificial Intelligence (ECAI), 2014 6th International Conference on*. IEEE, 2014, pp. 85–89.

[3] A. Chiou and C. Wynn, "Urban search and rescue robots in test arenas: Scaled modeling of disasters to test intelligent robot prototyping," in *Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*. IEEE, 2009, pp. 200–205.

[4] O. De Silva, G. K. Mann, and R. G. Gosine, "Development of a relative localization scheme for ground-aerial multi-robot systems," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 870–875.

[5] F. Cocchioni, V. Pierfelice, A. Benini, A. Mancini, E. Frontoni, P. Zingaretti, G. Ippoliti, and S. Longhi, "Unmanned ground and aerial vehicles in extended range indoor and outdoor missions," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, 2014, pp. 374–382.

[6] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, and G. W. Kit, "Davinci: A cloud computing framework for service robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3084–3089.

[7] Open Source Robotics Foundation, Inc., "ROS 2.0 Design." [Online]. Available: http://design.ros2.org/

[8] M. Du, J. Wang, J. Li, H. Cao, G. Cui, J. Fang, J. Lv, and X. Chen, "Robot robust object recognition based on fast surf feature matching," in *Chinese Automation Congress (CAC), 2013*. IEEE, 2013, pp. 581–586.

[9] UTSA Cloud and Big Data Lab, "OpenStack Configurator for Academic Research (OSCAR) Documentation." [Online]. Available: http://cloudandbigdatalab.github.io/OSCAR/

[10] OpenStack-Ansible Contributors, "OpenStack-Ansible Installation Guide." [Online]. Available: http://docs.openstack.org/developer/openstack-ansible/install-guide/

[11] S. H. Juan and F. H. Cotarelo, "Multi-master ros systems," 2015. [Online]. Available: http://www.iri.upc.edu/files/scidoc/1607-Multi-master-ROS-systems.pdf

[12] I. Open Source Robotics Foundation, "Ros environment variables," 2013. [Online]. Available: http://wiki.ros.org/ROS/EnvironmentVariables

[13] S. Niekum, "ar_track_alvar-ros wiki," 2013. [Online]. Available: http://wiki.ros.org/ar_track_alvar

[14] P. Benavidez, J. Lambert, A. Jaimes, and M. Jamshidi, "Landing of an ardrone 2.0 quadcopter on a mobile base using fuzzy logic," in *World Automation Congress (WAC), 2014*. IEEE, 2014, pp. 803–812.

[15] P. Benavidez, M. Muppidi, P. Rad, J. J. Prevost, M. Jamshidi, and L. Brown, "Cloud-based realtime robotic visual slam," in *Systems Conference (SysCon), 2015 9th Annual IEEE International*. IEEE, 2015, pp. 773–777.