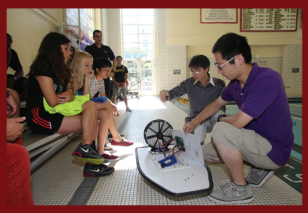
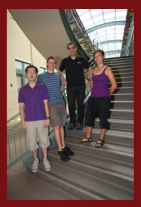
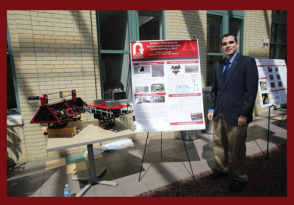
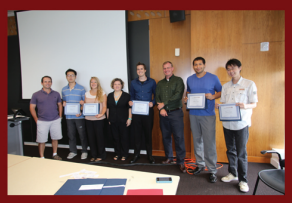
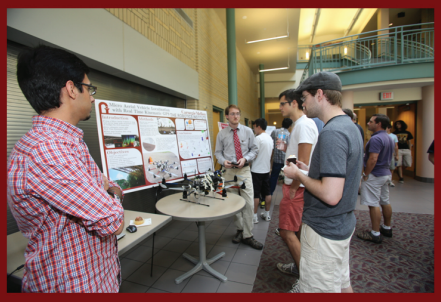
VOLUME 1 FALL 2013

ROBOTICS INSTITUTE
Summer Scholars (RISS) Working Papers

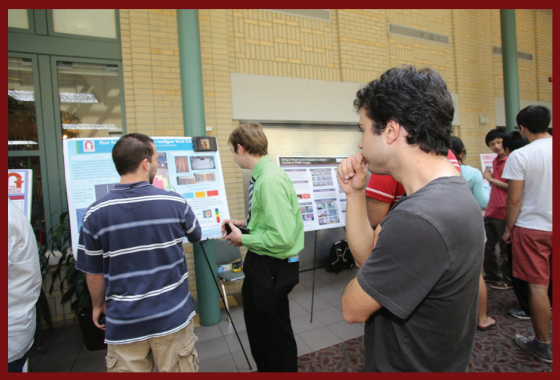
JOURNAL








Robotics Institute Summer Scholars Working Papers Journal

Volume 1 Fall 2013

We gratefully acknowledge the support of the National Science Foundation through the Research Experience for Undergraduates (REU) program (Grant # CNS 1263266).

Table of Contents

Table of Illustrations

Congratulations Cohort of 2013!

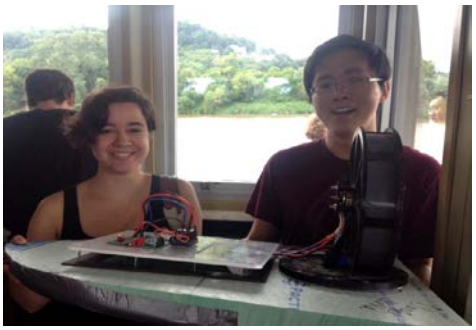
Luis Bilbraut Márquez	Universidad del Turabo
Harrison Billmers	The College of New Jersey
Philip Cooksey	California State University, Monterey Bay
Micah Corah	Rensselaer Polytechnics Institute
Julia Deeb	Georgia Institute of Technology
Adrian Escoto	University of the Pacific
Tao Fu	Nanjing University of Science and Technology
Allison Funkhouser	Trinity University
Yaonan Guan	Nanjing University of Science and Technology
Lu Hou	Nanjing University of Science and Technology
Emmanuel Johnson	North Carolina Agricultural & Technical State University
Jacqueline Kay	Swarthmore College
Aida Khosroshahi	San Jose State University
Andrey Kurenkov	Georgia Institute of Technology
Christopher Okwonko	Norfolk State University
Ellis Ratner	Bowdoin College
Karuna Relwani	University of Pittsburgh
Nelson Rivera Garcia	Universidad del Turabo
James Samotshozo	Howard University
Krishna Raj Sapkota	Jacobs University Bremen
Roger Smith	Hampton University
Stephen Smith	North Carolina State University
Ander Solorzano	Rose-Hulman Institute of Technology
Caroline Suni	Pennsylvania State University, University Park
Albert Toledo	Hampton University
Ruffin White-Magner	Rose-Hulman Institute of Technology
Micah Williams	Fort State Valley University
Troi Williams	Norfolk State University
Weilun Wu	National University of Singapore
Zesheng Xi	Nanjing University of Science and Technology
Danfei Xu	Dickinson College
Burak Yücesoy	Bilkent University
Abdullah Zafar	Carnegie Mellon University, Qatar
Wenqiang Zhou	Nanjing University of Science and Technology

To Future Summer Scholars

We write to you today to share our summer experiences with you. Not just as fellow roboticists, but as fellow summer scholars.

For those of you who are not familiar with this program, the Robotic Institute Summer Scholars (RISS for short) is a summer internship designed to expose undergraduates to research in the robotics disciplines. Here students get to learn and experience the cutting edge research in robotics while working with a diverse and passionate team of research students and professors. Students get to learn about the process, the teamwork, the documentation, and the presentation aspects that are involved in the research field. However, the RISS program was about more than just research; it was about learning about a new field, exploring graduate work, meeting innovative faculty, and forming friendships with fellow interns.

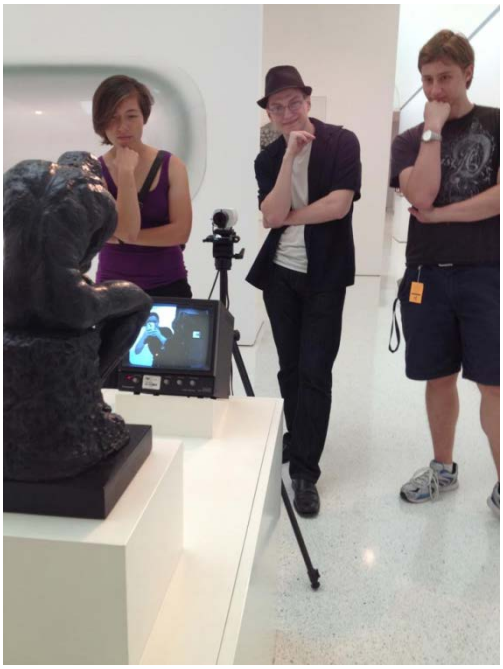
For scholars aspiring to graduate studies in robotics, this experience offers an insight to the ongoing research and applications currently under study by passionate roboticists. Aside



from technical work, this program gives the scholars an opportunity to attend thesis presentations given by Ph.D candidates, to learn about graduate fellowship programs, and to attend various technical presentations from roboticists and professors from around the various institutions. This experience offers various leadership opportunities to help strengthen teamwork and organizational qualities. We strongly advise the scholars to take advantage of all of these opportunities in order to make the most out of this internship.

Your work as a summer scholar has lasting impacts on the research community. Though this is an intimidating prospect, the friends you meet and the mentors you find at CMU will help you along the way. One of the most important things we learned that summer was that research is a collaborative process. Nothing in computer

science (or any other field for that matter) can be achieved alone. Additionally, scholars may be asked to attend the National Conference of Undergraduate Research (NCUR) in DC after the summer. Here students from all over the US present the cutting edge research that they were involved in the summer. This is a chance to meet students and program directors from various academic institutions as well as board members of the National Science Foundation (NSF). This is an extreme honor and we encourage the students to accept this opportunity given the chance.



During this program, students are also given the opportunity to give back to the community by sharing their knowledge and experience with local and international educational programs that engage students in practical, hands-on Science, Technology, Engineering, and Mathematics (STEM) activities. Other examples include showing local grade schools about the innovative, thoughtful, and creative work involved in the construction of space robots. Here, local students learn about the daunting challenges that can be solved by a group of passionate engineers working together as a team. In another experience, the RISS scholars educated and demonstrated to local environmental programs about how a swarm of surveying hoverboats collect information of polluted water bodies. From designing an optimal hull that will perform in different conditions to the full implementation and testing in a real environment, students were involved hands-on through the whole process.

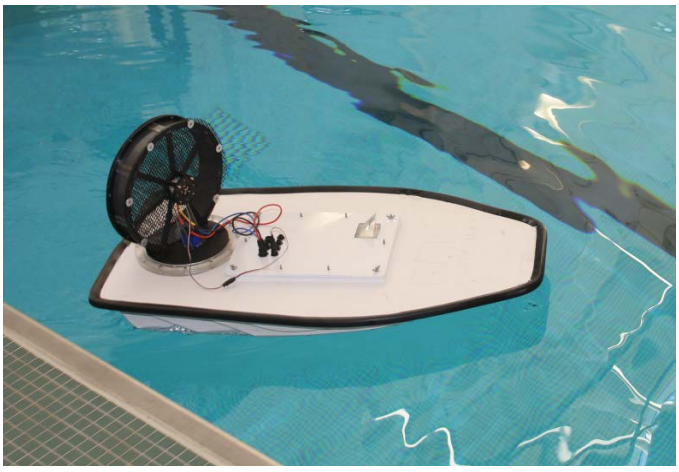
Finally, the last lesson we leave behind to the future scholars is to simply have fun. This program will enrich your understanding, expand your horizons, and strengthen your desires. However, all of these rewards are in vain unless you have fun with those around you. During free time, we encourage the scholars to explore the city, to organize social events with the entire group, and to meet other students from around the world. We truly wish you the best of luck as you endeavor in the path set in front of you and that you enjoy your time that this wonderful program offers you.

Sincerely,

Julia Deeb & Ander Solorzano
RISS Scholars 2013

2013 Robotics Institute Summer Scholars Program Partnerships

RiverQuest



CMU Cooperative Robotic Watercraft

In 2013, the Robotics Institute Summer Scholars Program (www.ri.cmu.edu/RISS) partnered with the Carnegie Cooperative Robotic Watercraft Group (http://crw-cmu.blogspot.com/) and RiverQuest (www.riverquest.org) to (1) *launch a capstone experience for undergraduate researchers that enabled the students to apply previous course and lab work to the challenges and unpredictability of designing, operating, and testing a robot in a real-world environment* and to (2) contribute to the development of robotic air boat activities for middle and high school students. For the scholars, the capstone started with an introduction to robotic air boats, opportunity to join the scholar leadership team that would plan the capstone field experience, opportunities to join practice field testing, a hull design seminar and challenge, and the culminated with an afternoon of field testing on the river. Caroline Suni, working with a team of scholars and partners, organized the field experience and translated this experience into draft curriculum for future use.

The capstone impacted the way the scholars see themselves as scientists and how they approach problem-solving. Working with robots in a real-world environment gives scholars a different perspective on what needs to be done, how it needs to be done and to some degree inspires their efforts. Scholars conducted mini-field tests over the summer in multiple environments. However, on the day of field testing on the river, scholars faced additional challenges. Heavy rains forced the opening of dams. The flow of the river was greatly increased and littered with debris. Controlling the boats and obstacle avoidance became increasingly challenging. This real-world experience impacted how these future roboticists will approach system design and problem solving.

Scholars applied their skills from a broad range of fields to examine how robotic watercraft could be used in environmental studies. The robotic air boats provide a novel way of presenting water science and environmental issues to younger students. In an era where technology is ubiquitous, fusing new technologies with these sciences can engage students in exciting hands-on and learner-driven activities. The study of nonpoint source pollution and its impact on water quality provides the nexus between RiverQuest's environmental education programs and the new technology represented by the robotic air boats. While traditional testing modes are often dependent upon water sample collection and manipulation to determine

pH, oxygen concentration, dissolved solids, and other parameters, the air boats may contribute to a robust data collection by enabling the testing of certain parameters in situ. Potential use of airboats for testing tributaries where the large vessel cannot access provides a next-step opportunity for student engagement and expanded data collection. Scholars also learned about hull design, system components, preparation and testing.

During the field experience, the scholars tested robotic boat activities designed by cohort members. The field experience design and activity development were led by summer scholar Caroline Suni and a team of scholars including Tao Fu, James Samotshozo, Ander Solorzano, Troi Williams, Zesheng Xi, and Wenqiang Zhou. The resulting activities contributed to ongoing collaboration and development of robotic boat curriculum for K-12 Pittsburgh students.

Community engagement and broader impact are important aspects of being a scientist today. Each year the summer scholars eagerly embrace multiple opportunities to share their research results, projects, and knowledge with educators and members of the community. Scholars have the opportunity to work and research alongside labs and science educators recognized for not only their innovative research but also for sharing current research with the public. Such an experience deepens the scholars' sense of responsibility to contribute to the health of our communities through science and action. Like their faculty mentors and research advisors, summer scholars are also role models and mentors for future scientists.

Many thanks to Carnegie Cooperative Robotic Watercraft group (led by Paul Scerri and George Kantor) and the RiverQuest team (led by Jeffrey Jordan, Suzi Bloom, Megan Griffin, and Gerry Balbier). Additional project guidance and support provided by CMU's Leonard Gelfand Center for Service Learning and Outreach led (by Judith Hallinen) and the Richard King Mellon Foundation contributed to the success of this partnership.

Robotics Institute Summer Scholars Program

The Robotics Institute Summer Scholars (RISS) Program is an intensive undergraduate research program at Carnegie Mellon University. Summer scholars participate in innovative research that focuses on robotics as the intelligent connection of perception to action. Scholars work with faculty, post-doctoral fellows, researchers, graduate students, and fellow summer scholars from around the world to conduct research work in:

- Intelligence: including core AI technologies, motion planning, control theory, planning under uncertainty, POMDPS, game theory, and machine learning.
- Perception: including computer vision, stereo processing, understanding ladar and 3D sensing, state-estimation, and pattern recognition.
- Action: including work mechanisms, actuators, their design and control.

Previous scholars have worked on projects ranging from distributed sensing to autonomous flight through cluttered forests. Learn more about RI participating projects at www.ri.cmu.edu/RISS and scholar contributions to this research.

Through the program, scholars are:

(1) Immersed in a guided research process that enables them to experience the thrill of discovery and to adopt the role of scientist as one that is authentically their own;

(2) Inspired to pursue careers in robotics and related STEM fields and equipped with the skills and new knowledge to seize industry and graduate school opportunities;

(3) Challenged by the interdisciplinary nature of robotics, the complexity of the research, and the vast potential to impact and improve the world's quality of life;

(4) Supported by robust student-development programming that complements the research immersion and informs the student's post research experience trajectory; and

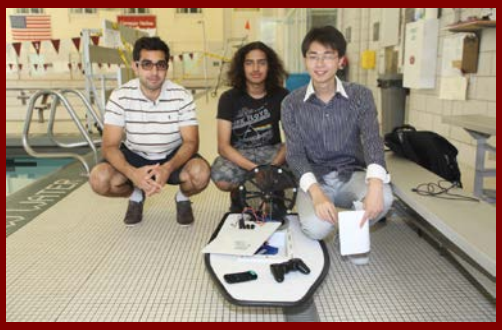
(6) New members of lifelong global community of researchers, entrepreneurs, and innovators that support, encourage, and enrich each other's lives.

The Robotics Institute at Carnegie Mellon University, the largest university-affiliated robotics research group in the world, offers a diverse breadth of research with an extensive range of applications; with over a hundred funded research projects. The Institute is a global leader in robotics research, education, and innovation. The Institute's experience, capacity, and faculty engagement extends unparalleled opportunities for students to be immersed in cutting-edge research while building in-demand STEM knowledge and skills.

The institute has seven years of experience hosting successful formal summer undergraduate research programs. The RI Summer Scholars program has grown to an average cohort size of 30 students and yields an impressive number of successful graduate school applications (at CMU and top universities around the world) and research position placements. For instance, 7 former Summer Scholars were admitted to the fall 2012 Robotics Institute incoming class (1 PhD, 2 research MS, and 4 Masters of Robotics Systems Development). For fall of 2013, there were 4 PhD and 5 masters offers of admission were extended to RISS alumni.

PHOTO GALLERY

RISS Presentations 2013



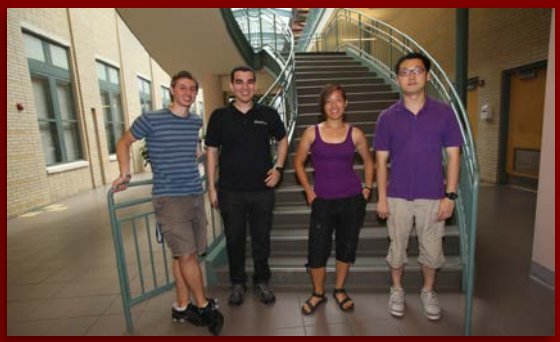
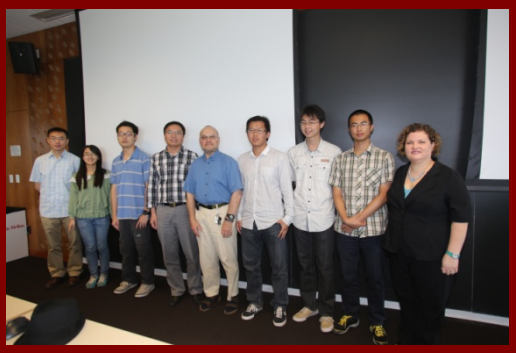
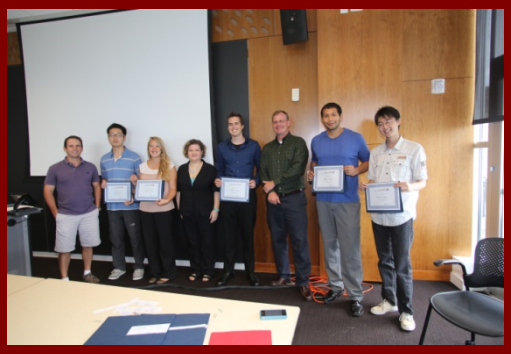


PHOTO GALLERY

RISS Ceremony 2013





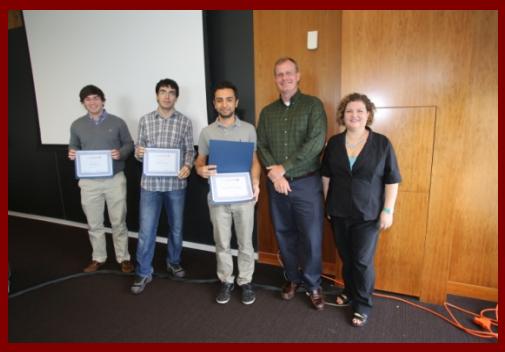
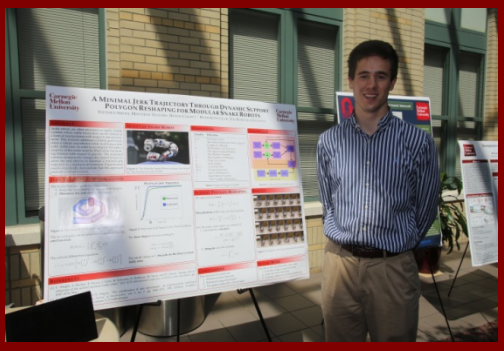
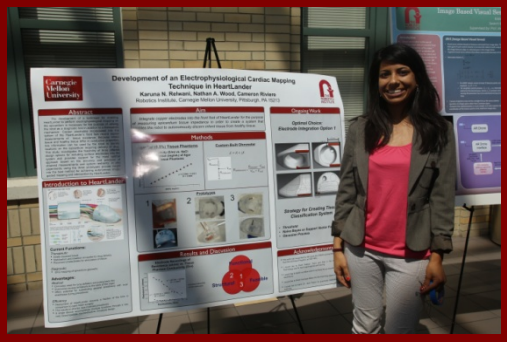
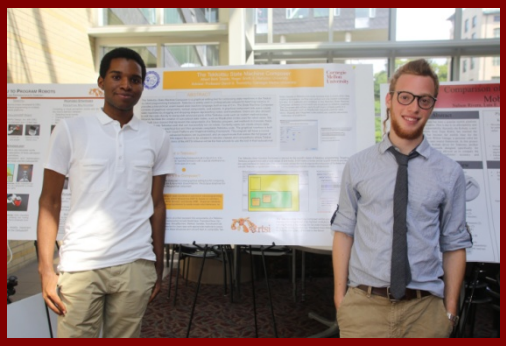


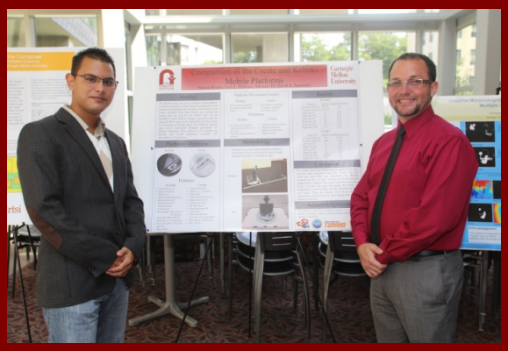
PHOTO GALLERY

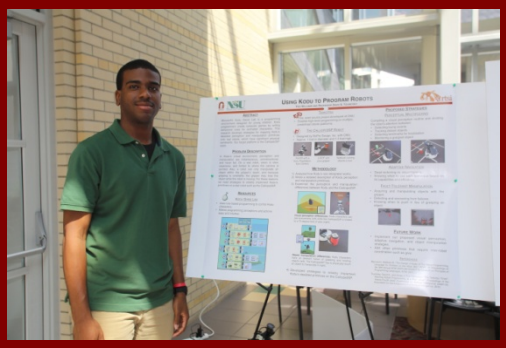
RISS Posters 2013

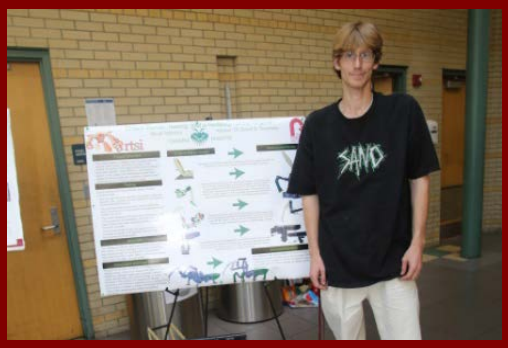


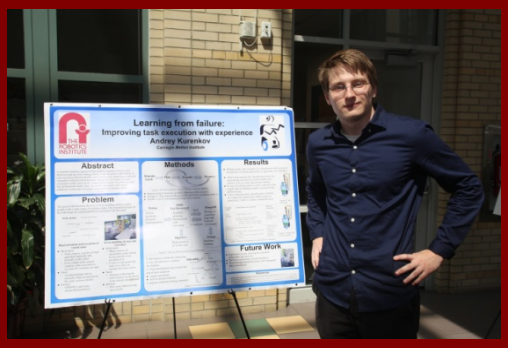


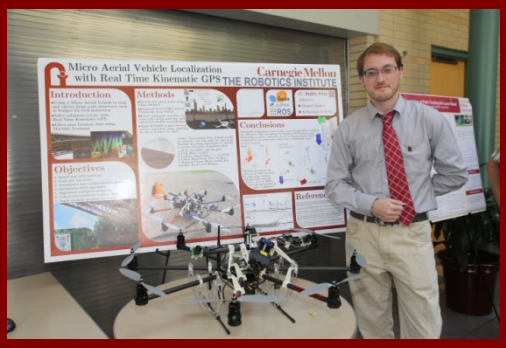












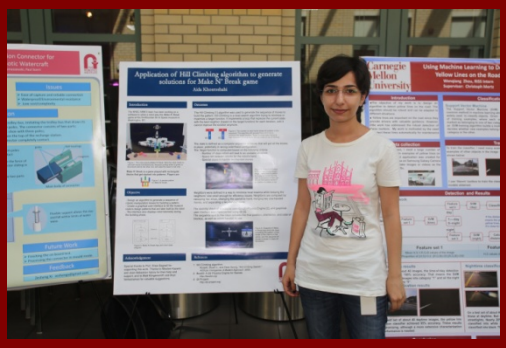


PHOTO GALLERY
RISS 2013

Working Papers

Authors

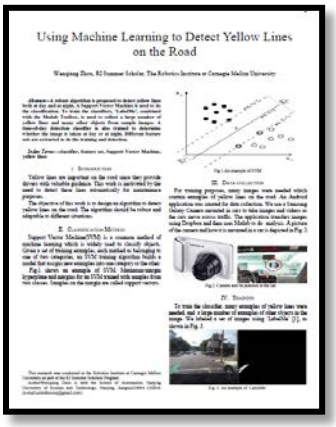

Software for Cooperative Assembly of an Airplane Wing Ladder Using Mobile Robots


Multi Sensor Fusion and 2D State Estimation of Lunar Rover


Designing a PID Control Algorithm for a Lunar Tyrolean Robotic Survey Platform


Airboats in Pittsburgh and Nanjing


Improved Localization and Mapping for Miniature Aerial Vehicles (September 2013)


Using Machine Learning to Detect Yellow Lines on the Road

Micah Corah

Software for Cooperative Assembly of an Airplane. Wing Ladder Using Mobile Robots

Lu Hou

Multi Sensor Fusion and 2D State Estimation of Lunar Rover

Ander Solorzano

Designing a PID Control Algorithm for a Lunar Tyrolean Robotic Survey Platform

Zesheng Xi and Yaonan Guan

Airboats in Pittsburgh and Nanjing

Burak Yücesoy

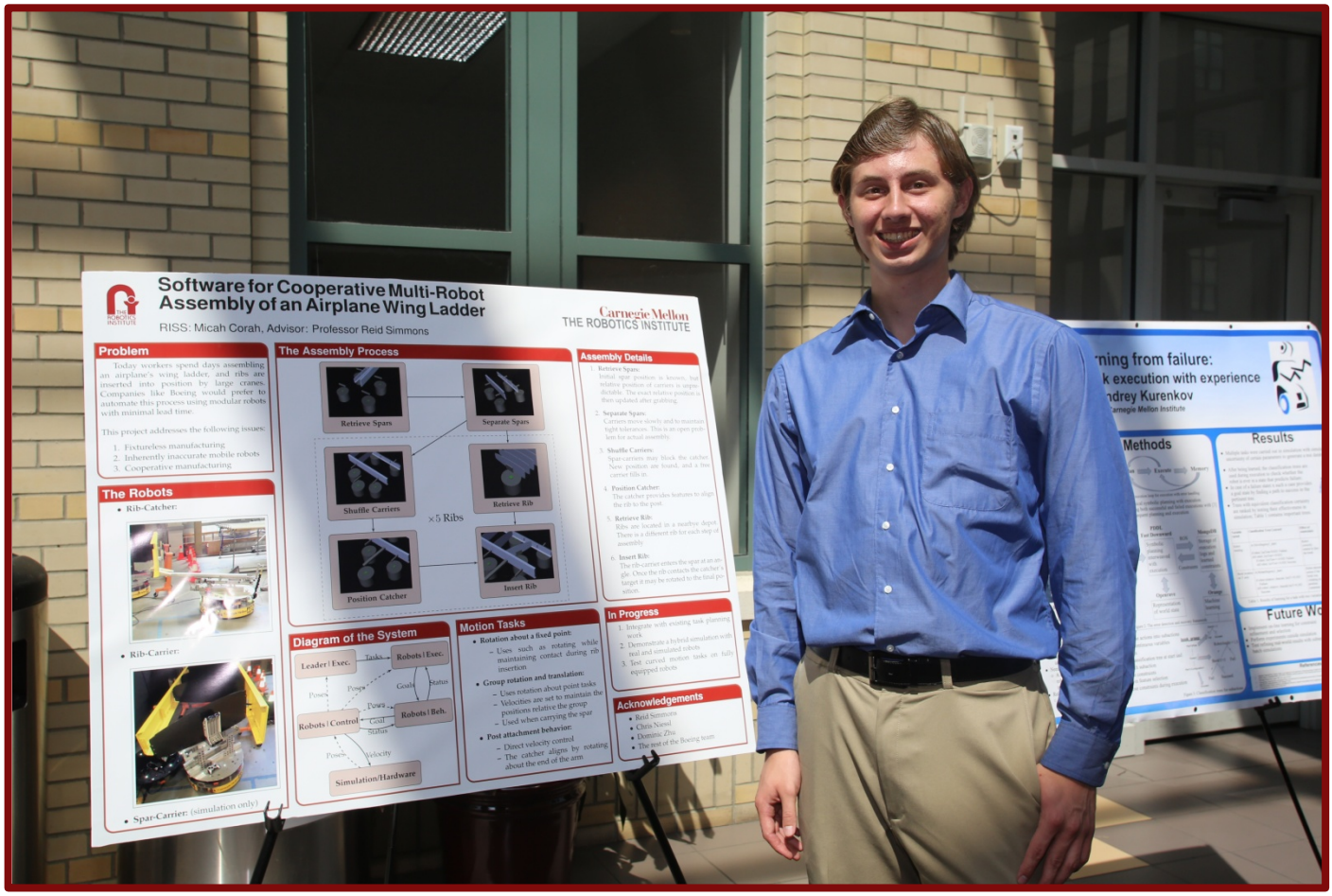
Improved Localization and Mapping for Miniature Aerial Vehicles

Wenqiang Zhou

Using Machine Learning to Detect Yellow Lines on the Road

Micah Corah

RISS 2013



Software for Cooperative Assembly of an Airplane Wing Ladder Using Mobile Robots

Micah Corah, Chris Niessl, and Reid Simmons

I. Introduction

TODAY airplane wings are assembled by large groups of workers over a period of days who use cranes to position heavy ribs inside large wing-spars. Because this is a slow process, airplane manufacturers simply cannot keep up with demand. Automation is a desirable option for increasing production. However, the assembly process is difficult for robots for some of the same reasons as it is for humans:

- Ribs and spars are heavy
- Airplane wings are large compared to people and robots
- Small holes holes impose tight positional tolerances

When mounted on tracks, robotic arms are able to move around the entire length of a wing. This kind of setup implies installation of large and expensive fixtures. We believe that robotic arms are unsatisfactory, as are factory workers.

A group of mobile robots could instead be used to assemble airplane wings with no need for such fixtures. Given access to the necessary parts a group of mobile robots could potentially assemble airplane wings *having beeng placed in an otherwise empty warehouse with absolutely no fixtures*. Before achieving this, a number of problems must be addressed. The robots used on this project are innaccurate and cannot on their own satisfy the tolerances. As no fixtures in the evironment will be allowed, the robots then provide their own fixtures for fine positioning of the parts. At the same time the assembly process must be coordinated at the grand scale as all of the robots must arrive at the correct positions at a specified time in order to efficiently perform the assembly process.

This paper is a technical report discussing software development primarily for a simulation of the coordinated assembly process but also relating to components that will run on the physical robots. First, the robots being used are introduced. High-level details of the assembly process are covered as are many of the details and difficulties encountered during the implementation. We finish with a discussion of integration of a planning layer and implementation of a hybrid simulation where simulated robots can dynamically switch to represent real robots for parts of the assembly process.

II. The Robots

Assembly is performed by robots with identical omnidirectional bases having three different kinds of tooling. These robots are the rib-carrier (Figure 1), rib-catcher (Figure 2), and spar-carrier.

A. The Rib-Carrier

The rib-carrier is responsible for carrying and inserting ribs. The top will rotate compliantly in all directions to allow the rib to be passively aligned to the spar. A column lift will lower the fingers once the rib is inserted, allowing the rib-carrier to escape under the assembly. Additional sensors will be integrated as the problem of rib-insertion is solved.

B. The Rib-Catcher

The function of the rib-catcher is to provide fixtures for fine positioning of the rib. Pictured is a metal target that attaches flush with the post (attachment point) on the spar. Detections of impact of the rib with the target will be used as part of the process of servoing the rib into place.

C. The Spar-Carrier

Currently the spar-carriers only exist in simulation with no physical equivalent. Instead, the wing-ladder is held in place by a set of jack-stands. The spar-carrier robots are responsible for retrieving the spar and holding it in place.

III. The Simulation Environment

A. Simulated Robots

In simulation, the robots are composed of a small collection cylinders and cuboids representative of their main physical features. These robots go through the motions of the assembly process without simulation of lower level details or dynamics. The assembly process is performed by seven robots, five spar-carriers (four to carry the spars and one which is used intermittently), one rib-catcher, and one spar-catcher.

B. The Wing-Ladder

The simulated wing-ladder is made up of two spars and five ribs. The ribs have holes for fasteners on either end and are inserted into the spar and fixed to posts (attachment points) on the spars. The ribs are spaced at roughly equal distances with one rib at each end of the spar. The ribs each fall on the same side of their respective posts except that one of the end ribs is flipped such that the end ribs are each attached outside of the posts.

The wing-ladder is the only object in the simulation taken directly from the real world, derived from the original CAD model (Figure 3). This provides a one-to-one correspondence between real and simulated components allowing the simulated assembly process to be related directly, even simultaneously, to the real assembly process despite incomplete simulation of the robots and of dynamics. The real wing-ladder does not have properties such as identical ribs or the centroids of each rib all falling on a single plane or axis. In order to deal with this

the reference frames used for each part of the ladder are not based on strict alignment to certain features but instead offset from the reference frame of the original CAD model, which is the center of the rib on the larger end.

C. Abstractions in the Environment

The abstractions in the environment are the objects of the simulated environment without any graphical representation. Each however places strong contraints on the simulation. Note that, although these are abstracted in simulation, some or all would be physical elements of a complete implementation. For instance, a device at some location would likely be necessary to load ribs or spars onto the robots, defining the parts depot. Likewise, the assembly-station may be a well-defined section part of a factory. These abstractions are as follows:

1) The Parts-Depot: This is the physical space where the ribs and spars are initially located. The parts are each placed next to each other and are rotated ninety degrees from their positions when assembled.

2) The Assembly-Station: The assembly-station is the location where the wing-ladder will be assembled. For the purpose of the simulation few assertions are made about this or other spaces except that the intersection of any two is empty.

3) Corridors: At the moment corridors have not been implemented explicitly in the software but are the pathways that robots may follow when moving between parts-depots or assembly-stations. Implicitly, the paths that the robots follow when picking up parts can be thought of as corridors.

4) Intersections: Intersections are the regions where corridors would intersect. Motion through the intersections is therefore managed explicitly in order to prevent collisions. These like corridors have not yet been implemented.

5) Parking Spaces: These are explicit locations where a single robot may sit idle with the guarantee that no other robot will collide while the robot remains in that space. Currently parking spaces are defined by the initial positions of the robots. In the future, when additional robots and wing-ladders are added, parking lots with dynamically-allocated spaces will be defined. These may for instance be defined by a point indicating the location of the first space, a vector indicating the direction of additional spaces, and a maximum number of parking spaces in the lot.

IV. The Assembly Process

The assembly process for the wing-ladder is composed of five basic steps: retrieval of the spars, positioning of the spars, rib-retrieval, shuffling of the spar-carriers when necessary, attachment of the rib-catcher, and insertion of the rib. This process is demonstrated in Figure 4. The summary below covers the basic logic of this process while details and technical issues will be explained more thoroughly later.

1) Retrieval of the spars from the depot is performed by a group of carriers. The number of carriers and hold locations are calculated based on the length of the spar. Spar-carriers are selected and made to approach and grab the spar. This group then carries the spar to the assembly station while maintaining rigid formation in order to prevent collisions.
2) Once the spars are in the assembly station, they are moved to their relative positions for assembly. Tight tolerances are crucial. This step is an open problem in the physical assembly process and even somewhat difficult in simulation.
3) The rib-carrier moves to the depot picking up the appropriate rib and then carries the rib to a position in front of the spars.
4) The rib-catcher attaches to the post on the spar in preparation for insertion of the rib.

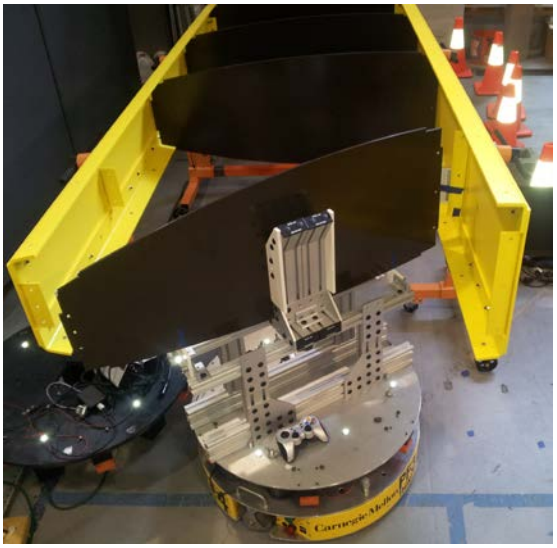


Fig. 1. The Rib-Carrier in position for rib-insertion

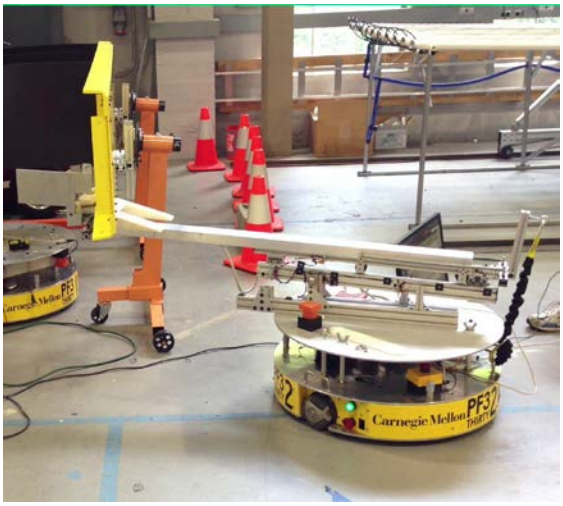


Fig. 2. The Spar-Catcher aligning to the spar during the post-attachment process

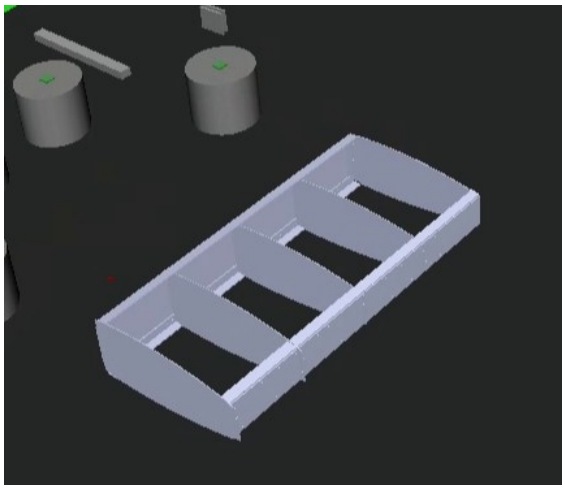


Fig. 3. The wing-ladder after being assembled

5) Often, one of the spar-carriers will be in the way of the rib-catcher. When this occurs, a free spar-carrier takes up an alternative position on the spar, and the offending spar-carrier is parked.
6) The rib-carrier enters the spar at an angle such that there is sufficient clearance on each side of the rib. Once the edge of the rib taps the target on the spar-catcher a fine

disengages and exits the wing-ladder.

V. Software Structure

As this project involves large numbers of robots operating simultaneously the software stack includes a number of processes running in parallel. IPC (Inter Process Communication) provides non-blocking communication between all of the processes when running in simulation or on the physical hardware [1]. We use the Syndicate architecture which is composed of three layers of abstraction: planning, executive, and behavioral running simultaneously on multiple robots [2]. As of yet, the planning has not been included. The executive is implemented using TDL (Task Description Language) [3]. TDL is used to initiate execution of high-level tasks such as moving from one point to another or grabbing an object. TDL maintains constraints between tasks and provides a variety of options for how a task is executed and completed. The behavioral layer is composed of blocks each representing a robot behavior generally involving closed loop interaction with the environment, such as moving forward until a switch is activated. Below these are an intermediate robot controller, hardware level control, and a simulator based in OpenRave. The relationships between each of these processes is diagrammed in Figure 5.

A. The Leader Process

The leader is an executive layer process responsible for spawning tasks related to the entire group of robots which then spawn individual tasks on each of the robots. For instance, the

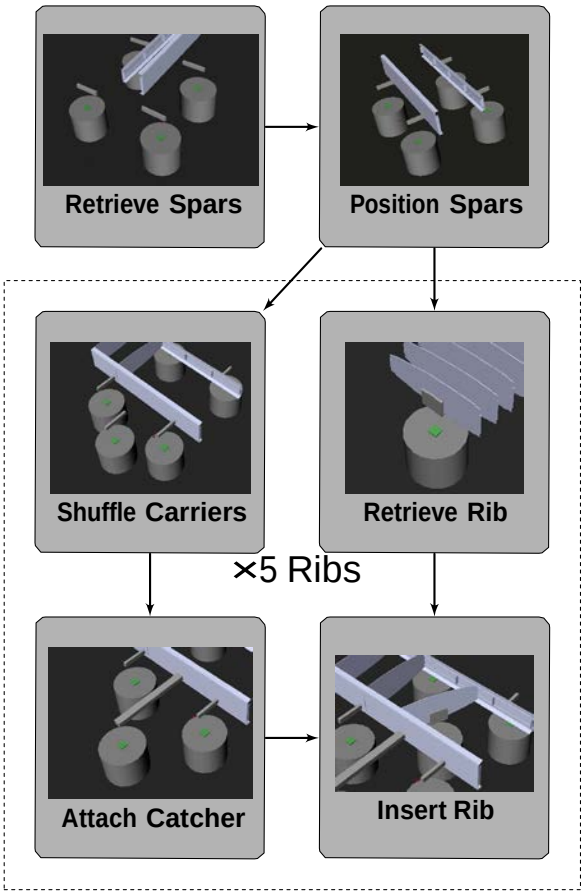


Fig. 4. A diagram of the steps of the assembly process

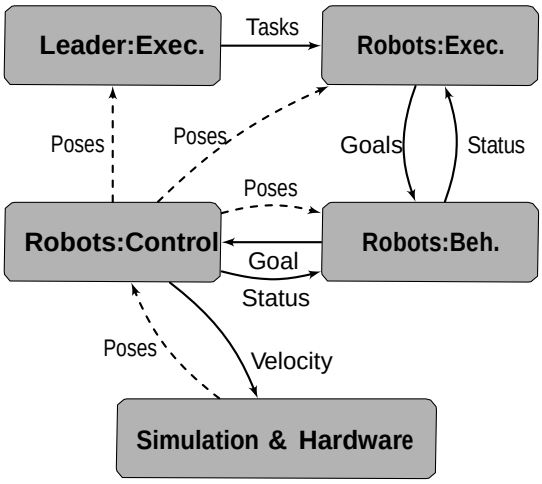


Fig. 5. A simplified diagram of the flow of control and information between processes

top level wing-assembly task is spawned on the leader resulting in high level tasks being spawned for insertion of each rib and then motion tasks being spawned on each of the robots. As tasks are being executed the leader also maintains some high-level data on the individual robots related to their state and location. This process will also be important later on as it will be primarily responsible for communication with the planner layer which will soon be integrated into the software stack.

B. Robot Processes

Each robot is associated with an executive, behavioral, and control process. The executive process maintains tasks associated with the robot and also data relevant its current state. With few exceptions the behavioral layer communicates and monitors position goals obtained from the executive to the controller. The robot controller provides several modes of velocity control for point to point motion tasks. The controller is also responsible for interfacing with the simulated and physical robots and is responsible for distributing robot pose information.

C. The Simulator

The robots are simulated using a plugin to OpenRave which duplicates the interface to the physical robots. The simulator itself is devoid of dynamics and other complexities. Robot position is calculated by discretely integrating commanded robot velocities in real time. Note that there are absolutely no artificial sources of error and that simulated robots are assumed to exactly follow the commanded velocities as they are given. The simulator also provides grab commands that allow a robot to pick up any intersecting simulated object.

VI. Motion Primitives

Most tasks on the executive eventually spawn some motion primitive. Most commonly, the primitive is either a straight line motion or an in-place rotation. This is a side-effect of using primarily taxi-cab style motions. When used together, these span the configuration space of the robots. For the most part, these primitives are sufficient, but during some parts of the assembly process, motion is constrained. Additional primitives were implemented to deal with such situations.

A. Rotation About a Point

The constraints encountered during the assembly process are generally of the form of a rigid relationship of the robot to some fixed point in the environment. Such tasks require the robot to be able to rotate relative to an arbitrary point. Two examples are the rib-insertion and post-attachment tasks. During rib-insertion, the rib is rotated into place while maintaining contact with the target. During rib-insertion, the rib-catcher rotates about the end of its arm in order to align itself perpendicularly to the spar while also maintianing contact. Thus, a motion primitive was implemented which takes a point and an angle to address this issue. This is then executed by the controller (discussed later). Previous to this work, a velocity controller mode for curved motion had been implemented. However, this computed velocities appropriate for differential drive robots. The velocity controller was rewritten to compute velocities for omnidirectional bases instead as a part of implementation of this motion primitive.

B. Primitives for Rigid Motion of Groups

Another type of constraint is a rigid relationship between two or more robots. This constraint occurs in the simulation when the spar-carriers must move the spar. Note that the motion of the group of robots as a whole is unconstrained. For simple translations, the existing primitives are sufficient. If all of the robots move with the same velocities the rigid relationship is maintained. However, rotation of the spar is necessary in addition to translation. As the spar and spar-carriers together do not have any meaningful center of rotation, a point in the plane can be chosen arbitrarily. If all of the robots in the group rotate about a single point with the same radial velocity the rigid relationship is maintained. The task for rotation about a point can then be extended to rotations of groups of robots. The only requirement is to determine a valid rotational velocity. For each kind of robot, values are set for maximum rotational and translational velocities. The maximum rotational velocity about the center is limited by each of these. Naturally, rotational velocity about the centerpoint is no greater than the minimum of the robots' maximum rotational velocities. Also, given a rotational velocity ω about a point a distance $\boldsymbol{r}$ from the robot and translational velocity V_{max}:

$$V_{max} \geq \omega \times \boldsymbol{r}.$$

Given a valid rotational velocity, the rotation about a point primitive can be run on each of the robots in the group while maintianing an ideally rigid relationship. Using this, three motion primitives were added for motion of groups of robots. The simplest spawns straight line primitives on each of the robots; the second rotates the group about a point; and the third combines the two, taking initial and final positions as input. In the case of the last, note that the motions can be varied without changing the end conditions by moving the positions in their local frames (equivalent to moving a single robot's center of rotation). Using this property, a group can complete a motion using a single rotation and no translation or rotate about the centroid of the group before translating. With both translation and rotation, groups of robots can reach any point in their configuration space. When applied to the physical robots, the controller will have to be able to react to innacuracies in the motions of the robots relative to each other. This work is likely a good starting point to drive development of such a controller for the robots. This result is sufficient as the simulation is only intended to demonstrate the process of assembly, not simulating dynamics or contact.

VII. Control and Latency

Although the simulator attempts to provide idealized robot motion, the control remains far from ideal due to the non-deterministic nature of parallel execution. Generally, when

running the simulation, twenty-three processes and a number of threads run and communicate concurrently. Often, this all occurs on a single virtual machine, severely taxing resources. For the most part, the simulation tends to run smoothly. However, at a minimum, there are motion innaccuracies similar or greater in magnitude to those on the physical robots. Latency is observed as a result of the large quantity of processes all running round-robin on a single machine. In the best case, pose information on the controller is always slightly old as are the velocities that the simulator integrates based on a real-time clock. In the worst case, such as if the simulator is configured to send out pose updates too quickly (the controller reponds to each pose update with a velocity update), communication may slow almost to a halt, and robots may continue moving based on either old velocity commands or velocity commands computed with old positions for even several seconds. Old positions may also have negative effects on the executive.

There are a variety of ways of dealing with the latency issue. As mentioned previously, lowering the update rate reduces traffic. An appropriate update rate was found heuristically with an appropriate trade-off between increasing latency and decreasing accuracy. Message queue lengths are also set to one to eliminate old messages. In addition to reducing latency, measures were taken to increase the resilience to motion errors. At the executive layer, when precise relative positions must be known, robot positions are sampled after an action is completed. For instance, the spar-carriers may sometimes report succesfully reaching the locations of their holds on the spar when they are actually outside tolerances. Because the precise location of the simulated spars are already known, the transform from each spar-carrier to its spar is updated after grabbing the spar. The simulation is then resilient to error in spar-carrier positions at this step so long as the grabbing spar-carrier is in collision with the spar. With small tolerances and high speeds a robot can potentially jump across the goal region. As mentioned previously, this kind of problem can have serious consequences for curved motions. To deal with this, a feature was added to the curved motion controller to detect jumps from $(-90^\circ, 0^\circ)$ and $(0^\circ, 90^\circ)$ where the goal is 0° on the circle the robot is moving in. When this happens an existing motion finalization mode that moves the robot directly to the goal is triggered.

VIII. Toward Planning Integration and Additional Assemblies

The simulation described in this paper features assembly of only a single wing-ladder. Reasonable results were achieved with hard-coded steps. As the number of wing-ladders increase and factors such as failure to complete a task are included in the simulation, the need for an actual planning layer increases. Development of the planning layer is outside of the scope of this paper. However, an important part of integration of work on the planning layer is that the executive layer be designed in a way that simplifies the interface between the layers. This is achieved through features that can easily afford various levels of abstraction to the designers of the planner. The planner being developed will use information on the connectivity of the environment, the structure of the assembly task, and information on the time to complete tasks to move robots around the environment to the four assembly stations and depots and to schedule tasks using a just-in-time model. Note that the planner will not have a spacial or geometric representation of the environment. However, the geometric and spacial components each affect parts of the task structure. For example, the carrier shuffle is not performed if there is no carrier in the way of the rib-catcher. Use of abstraction eliminates the need to create rules on the planner to describe these interactions.

In order to simplify this abstraction the concept of workspaces has been added to the executive layer. The set of robot workspaces is the set of vertices of the planner's graph of the environment. The executive then must guarantee that there can be no interactions between robots in different workspaces. The designers of the planner are then *free to cede any desired level of control of robots within a workspace to the executive*. Therefore the executive can be allowed to assign sub-tasks to robots within a workspace as a part of execution of a task assigned by the planner. Workspaces are assigned to robots implicitly by motion tasks assigned by the planner. The planner may assign a task such as, "cross `Intersection1`, and enter `Station1`." Preceding assignment of the task, the robot workspace would have been `Intersection1`. After completing the task, the robot would be assigned the current workspace `Station1`.

Two additional pieces of information are also tracked by the executive: robot names (which are derived from robot type) and robot state. State may be information such as which rib a robot is carrying, which hold a robot has been assigned to on the spar, or whether a robot is currently free to be assigned tasks by the executive.

The result of storing this information is that the planner may be designed to manage tasks to whatever level of detail is desired. The command to insert a rib may have as few parameters as the assembly station it applies to. Given a rib-carrier with a rib, spar-carriers holding the spars, and sufficient additional robots all in the assembly-station, the executive can infer all subtasks and assignments necessary to complete the rib-insertion task. Just as easily the name of the rib-carrier could be used, in which case the executive would simply look up the rib-carriers workspace and state as part of inferrring further subtasks. On the other hand, designers of the planner may deem necessary explicit assignment of robots to a rib-insertion task. In this case, the executive may skip looking up which robots are in the workspace or look up names and locations to ensure the validity of the assignment or possibly to determine whether to wait until a robot has arrived in the workspace to start the task. The executive is therefore flexible to the needs of those designing the planner instead of being a driving force in the development of the planner, by not having strict requirements on the information needed before executing a task.

IX. Development of a Hybrid Simulation

The simulation portrays the overlapping steps of the assembly process but lacks finer details such as dynamics.

The tooling simply has not been designed with simulation in mind and furthermore was being developed in parallel to the simulation. Instead of simulating the hardware we chose to use the physical hardware to drive a subset of the steps of the simulation. We therebye can demonstrate a level of accuracy in simulation when a real robot can be substituted for a simulated robot without consequence.

A number of technical challenges were overcome to implement this idea:

- There must be some toggle that causes a simulated robot to track a real robot
- The toggle should not be enacted when the real robot is not present
- The reference frames of the real and simulated environments must be related
- Simulated and real time must be related

Naively, implementation of the toggle was initially approached with the idea that the simulated robot could simply be told to stop publishing positions and listening to velocity commands and instead just listen for positions being published by the real counterpart. Both the real and simulated robots would need to listen to some message and then switch to the appropriate mode. However, there are a number of difficulties and problems with this method:

- The real and possibly also simulated robots would have to send messages indicating their presence to syndicate
- An initialization step would be necessary specifying whether simulated or real robots are active
- The environments would either have to have the same reference frame or software would have to be implemented to allow the controller to use a transform between the environments
- The velocity controller would have to be able to distinguish between position information from simulated and real sources

Overall this would require a large amount of modification and effort for a cosmetic feature, especially considering potentially issues with the controller.

For this reason another option was initially implemented where the simulation contains a robot dedicated to tracking the real robot. This eliminates the serious technical issues but has consequences on the assembly process. As there is no parts depot in the real environment, a simulated robot would ideally pick up a rib and move to the spar before switching to tracking the real robot. Instead the real robot would have to have some other way of obtaining a rib. Since the simulated robot would not have performed the insertion, it may not end up in the position it usually would post-insertion. The process would have to be modified to deal with it potentially not being in ths position, or the simulated rib-carrier would have to be moved to this position by some other method.

Fortunately, when the previously discussed options are combined these complications go away. In our implementation there is a simulated robot that can be made to track a corresponding real robot, but these robots have different names as demonstrated in figure 6. Before the real robot performs a task, the simulation is passed the name of the real robot

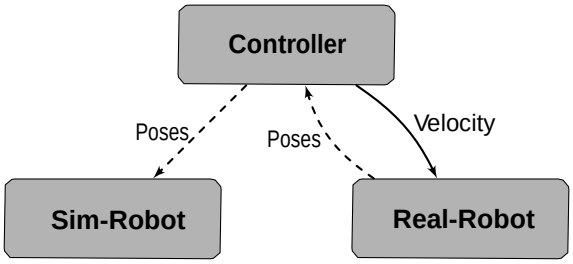


Fig. 6. Diagram of a simulated robot, "Sim-Robot" tracking a real robot, "Real-Robot"

and also a transform between the environments. To ensure a smooth transition before the simulated robot snaps to the position of the real robot, a task should be spawned to move the real robot into some starting position *before* the switch, and once this task has been completed, another task should be spawned to move the simulated robot to the position of the real robot. This immediately eliminates issues related to the controller as there is now one instance of the controller for both the simulated and real robots. Since the real robot and corresponding simulated robot are represented by the same robot in the simulation there are also no issues with acquiring the simulated rib or differences in final position.

X. Conclusions

This paper has presented a summary of software development work for multi-robot assembly of an airplane wing-ladder and a simulation of the assembly process. For background, we begin with a description of the robots, the environment, and the basic steps of the assembly process. Development of both hardware and software for this project continues, and the simulation is a self-contained subproject demonstrating work on software development and the large scale assembly process. We then move on to describe the software structure and communication between robots and processes. The motion primitives are the basic components of the assembly process, and performing the assembly process requires a set of primitives that satisfy the constraints at each step. During implementation, large overheads related to running so many robots and processes were recognized and delt with. As components from the executive down were completed, we began to look upward and developed components to support integration of the planning layer.

Cooperative assembly with mobile robots is not a new concept. This project is set apart by both its scale and granularity. The simulation as described in this paper uses seven robots to assemble a single wing-ladder. This is being extended to multiple wing-ladders and even more robots making the planner critical. Through various subprojects development extends downward through software (some of which being the focus of this paper) and hardware from tooling to the mobile bases themselves.

REFERENCES

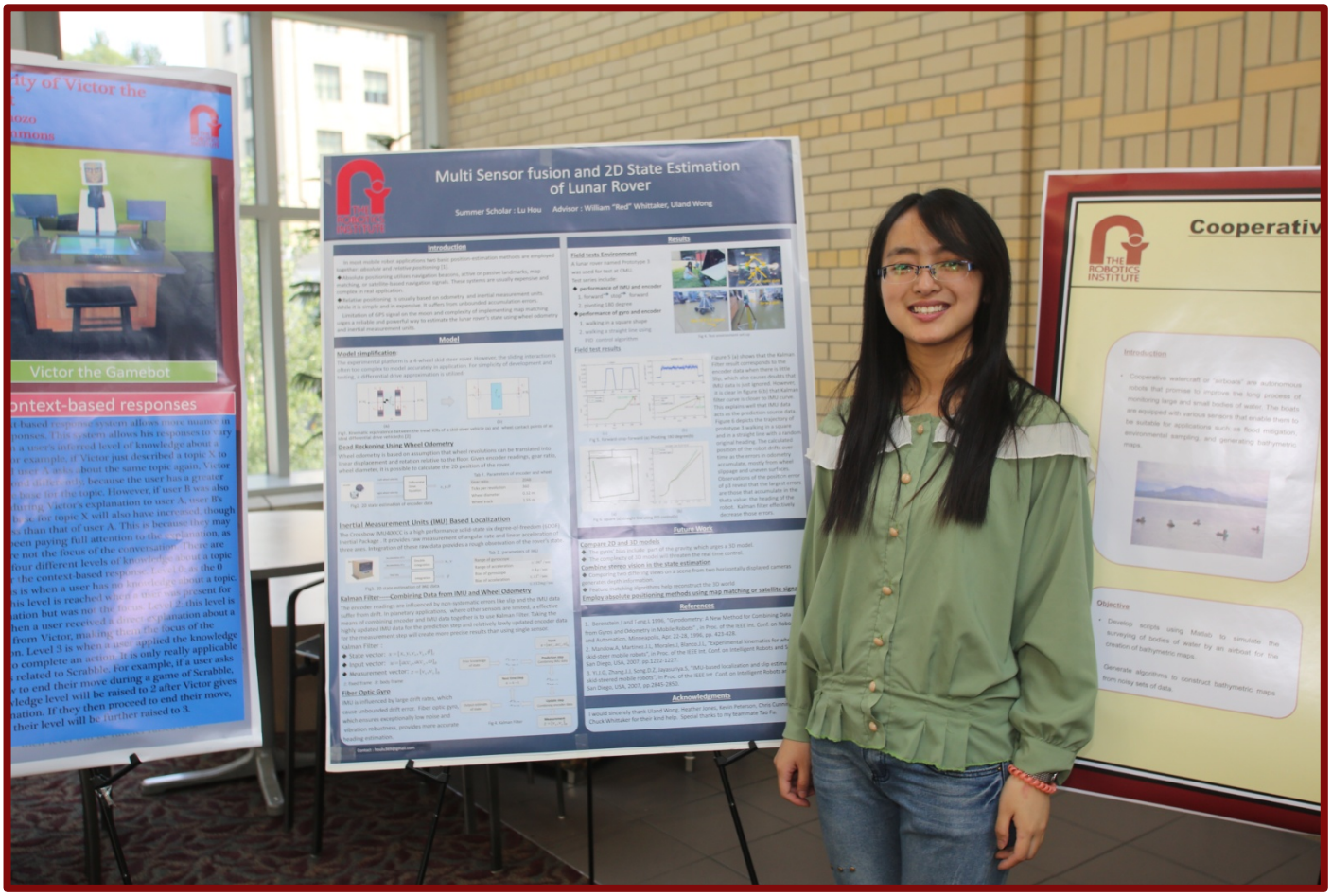
[1] R. Simmons and D. James, "Inter-process communication: A reference manual," *Robotics Institute, Carnegie Mellon University*, April 2001.

[2] B. Sellner, F. W. Heger, L. M. Hiatt, R. Simmons, and S. Singh, "Coordinated multiagent teams and sliding autonomy for large-scale assembly," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1425–1444, 2006.

[3] R. Simmons and D. Apfelbaum, "A task description language for robot control," vol. 3, pp. 1931–1937, 1998.

Lu Hou

RISS 2013



Nanjing University of Science and Technology (NUST) has been a partner in the CMU Robotics Institute international graduate education program – the Master of Science in Robotics Technology (MS-RT) since 2010. NUST became a special RI Summer Scholars program partner in 2013. The working paper journal articles here represent a sampling of the range of projects on which NUST students have collaborated.

Multi Sensor Fusion and 2D State Estimation of Lunar Rover

Lu Hou

***Abstract*—Wheeled skid-steer localization poses special challenges due to the complexity of kinematics and wheel/ground interaction. This paper provides two possible approaches to estimating the lunar rover's 2D state fusing information from wheel odometry, fiber optic gyroscope (FOG) and inertial measurement units (IMU). The first method applies a differential drive model after analyzing the kinematics of the lunar rover and uses wheel odometry and FOG to estimate the rover's state. The second effective means of combining encoder and IMU data together is to use a discrete Kalman Filter (KF). Both methods show proficiency in some particular applications. While using a fiber optic gyro can reduce heading error caused by lateral slippage, the Kalman Filter is able to calculate the rover's state without introducing the IMU's drift resulting from integration of accelerations and non-systematic errors of wheel odometry like slippage. Finally, we tested and validated the simplified model and KF with a laser range finder.**

***Index Terms*—differential drive model, Kalman Filter**

I. Introduction

Wheeled skid-steering mechanisms are widely used for all-terrain vehicles, especially true of some field applications like planetary exploration. The lunar rover named prototype 3 (Fig. 2.a) is a typical wheeled skid-steering vehicle.

A robust robot system relies much on its localization. Basically, two kinds of positioning methods are widely applied for mobile robot localization: absolute positioning and relative positioning [1]. Absolute positioning methods usually utilize expensive and complex systems like map matching and Global Positioning System (GPS) signal processing while relative positioning is based on the inertial measurement units and odometry. Considering the limitation of GPS signal on the moon and complexity of implementing map matching, it is urgent to build a reliable and powerful way to estimate the lunar rover's state using relative positioning methods.

The sliding interaction for a skid steer rover is often too complex for on-board calculation and real time control. A differential drive approximation is proposed based on studies that analyze the equivalence between these two models. Wheel odometry suffers from non-systematic errors like slippage, thus the accuracy of the heading estimation is significantly affected. Using a fiber optic gyro to provide the angle information independently will effectively help overcome this problem. In addition, building a Kalman Filter that combines both the wheel odometry and IMU data will provide the rover with both localization and slippage information.

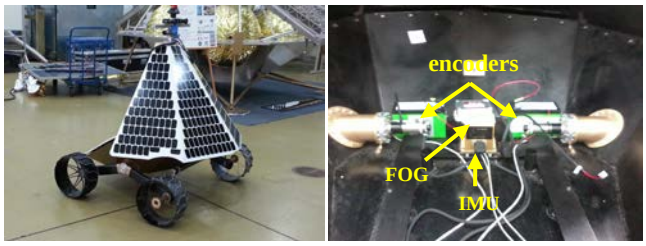


Fig. 1. a) Lunar Rover Prototype 3 b) on-board sensor suite of Prototype 3

II. Differential Drive Model

A. Model simplification

The experimental platform is a 4-wheel skid steer rover. However, the sliding interaction is often too complex to model accurately in application and it is not possible to predict the exact motion of the vehicle only according to the controlling inputs. So it is a necessity to obtain an optimized kinematic model for skid-steer vehicles.

Some work has already been done in studying the dynamics and model simplification in order to improve the on-board computation of a skid-steer vehicle. The kinematic equivalence between skid-steering and differential drive vehicles has been proposed for tracked vehicles [2].

Since the wheels on each side of Prototype 3 are connected together and always have the same angular velocity, the kinematics of Prototype 3 is similar to those of a tracked vehicle. For simplicity of development and testing, a differential drive approximation is utilized for Prototype 3.

However, once the differential drive model is employed for the lunar rover, we need to consider the turning efficiency of the rover, which has been studied in [3].

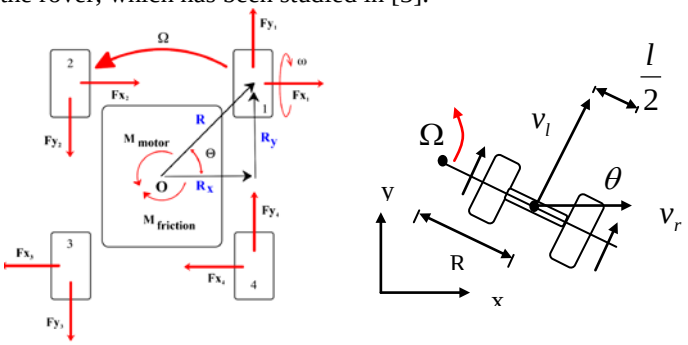


Fig. 2. a) Skid Steer Kinematics [3] b) simplified differential drive model

Lu Hou is with the Computer Science and Engineering Department, Nanjing University of Science and Technology in Jiangsu Province, China. Email: houlu369@gmail.com

This research was conducted at Carnegie Mellon University under the guidance of William "Red" Whittaker and Uland Wong.

The ideal turning rate Ω without longitudinal slippage for an standard Ackerman steering geometry is described by Eq. 1

$$\Omega = \frac{\omega r}{R} \tag{1}$$

where ω is the wheel angular velocity in radians per second, r is the wheel radius, and R is the distance from the center of the wheels to the center of rotation of the rover.

To calculate the rate of rotation Ω (Fig.2.a) for a skid steer rover, the actual turning radius which is reflected by the angle θ must be taken into account.

$$\Omega = \frac{\omega r}{R}\cos(\theta) \tag{2}$$

Denote l as wheel track (the distance between the centre of the two wheels), we can simplify the model Prototype 3 into a differential drive model by replacing l with $\frac{l}{\cos(\theta)}$. Namely, change its wheel track from into the diagonal distance of front left wheel and rear right wheel.

And then it is straightforward to calculate the robot's state using the differential drive equation based on the encoder readings.

B. *Differential drive kinematics*

Differential drive mechanisms consist of two wheels which are mounted on a common axis [4], wheels on both sides can be driven forward or backward. It is possible to change the trajectories the robot takes by varying the velocities of the two wheels.

If the rover's position at time t is (x, y), with a heading direction θ relative to the X axis. Then the instantaneous centre of curvature (ICC) location is described as

$$ICC = [x - R\sin(\theta), y + R\cos(\theta)] \tag{3}$$

Given the velocities V_l, V_r, we can get the rotation radius R (the distance between ICC and the center of the two wheels) and rate of rotation Ω using equation (3).

$$R = \frac{l}{2}\frac{V_l + V_r}{V_l - V_r}, \Omega = \frac{V_r - V_l}{l} \tag{4}$$

then the robot's pose at ($t + \Delta t$) is

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\Omega\Delta t) & -\sin(\Omega\Delta t) & 0 \\ \sin(\Omega\Delta t) & \cos(\Omega\Delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \Omega\Delta t \end{bmatrix} \tag{5}$$

C. *Fiber optic gyroscope (FOG)*

Non-systematic errors are inevitable in wheel odometry. The wheels will encounter slippage when pivoting and turn less than the expected degree as a result. One way to overcome this problem is to adapt the coefficient of the wheel track according to the real texture of the terrain. However, whenever the terrain changes, one needs to do a series of tests before the optimized coefficient is found, which is not convenient. Another means of giving better pivoting performance is employing a fiber optic gyroscope which has high accuracy and is not affected by slippage for the heading measure independently.

In this way, the rover's heading angle θ is derived directly from the fiber-optic gyroscope and the Cartesian coordinates (x, y) are calculated using dead reckoning equations:

$$\begin{aligned} x' &= x + R \times (\sin(\omega t + \theta) - \sin(\theta)); \\ y' &= y - R \times (\cos(\omega t + \theta) - \cos(\theta)); \end{aligned} \tag{6}$$

D. *Experiments and results*

The on-board sensor suit of the lunar rover shown in Fig.1.b . We use a DSP-3000 fiber optic gyroscope from KVH Industries.Inc and a Crossbow 400CC IMU for attitude measurements. The optical wheel encoder readings and motion control are implemented on Galil Controller. The control system is realized using Robot Operating System (ROS) with the control algorithm and Kalman filter design located in the higher level and the PID-based motor control located at the lower level. A laser range finder system is employed to provide the robot's absolute position information in all experiments.

In order to test the performance of the simplified differential drive model and the improved differential drive model with a FOG. We set the trajectory as a circle with a radius of 3m for the rover, the state estimations by differential drive equation with and without FOG are given in the graph.

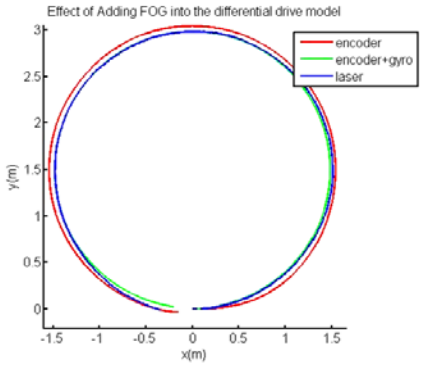


Fig. 3. Robot state estimation by differential drive equation with and without FOG compared to absolute position from laser data

The differential drive model works well in the state estimation for the rover. The improved one with a FOG providing the heading angle information shows greater accuracy. We have found an effective way of estimating the rover's state without introducing the complex skid steer model. This also helps save large amount of on-board calculation and makes real time control more robust.

III. Discrete Extended Kalman Filter Design

A. *Discrete Kalman Filter*

The encoder readings are influenced by non-systematic errors like slippage and the IMU data suffer from drift. In planetary applications, where other sensors are limited, an effective means of combining encoder and IMU data together is using discrete Kalman Filter [5].

The discrete Kalman Filter predicts the future state of the system $\hat{X}(k|k-1)$ based on the available system model F and projects the state error covariance matrix $P(k|k-1)$ using the time update equations

$$\begin{aligned} &\hat{X}(k|k-1) = F(k)\hat{X}(k-1|k-1) + B(k)u(k) \\ &P(k|k-1) = F(k)P(k-1|k-1)F^T(k|k-1) + B(k)QB^T(k) \end{aligned} \tag{7}$$

Once the measurement $z(k)$ becomes available, the Kalman gain matrix will be computed to incorporate the measurement into the state estimation $\hat{X}(k|k)$. The state error covariance for the updated state estimate $P(k|k)$ will also be computed using the following measurement update equations:

$$\begin{aligned} &\hat{X}(k|k) = \hat{X}(k|k-1) + W(k)(z(k) - H(k)\hat{X}(k|k-1)) \\ &W(k) = P(k|k-1)H^T(k)S^{-1}(k) \\ &S(k) = H(k)P(k|k-1)H^T(k) + R \\ &P(k|k) = (I - W(k)H(k))P(k|k-1)(I - W(k)H(k))^T \\ &+ W(k)RW^T(k) \end{aligned} \tag{8}$$

Where I is an identity matrix, and system $A(k)$, input $B(k)$ along with measurement $H(k)$ matrices are defined according to the detailed model in the following part.

B. Lunar Rover Model Setup

We denote two related frames for the lunar rover's model: a fixed global frame $I(x, y, z)$ and the rover's body frame $I(x, y, z)$ as shown in Fig. 2. b. Given the headed direction θ, it is simple to calculate the transformation matrix from the body frame B to the global frame I using 2D rotation matrix:

$$C_B^I = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{9}$$

The lunar rover's planar Cartesian coordinates (x, y) and heading θ describe the pose of the rover. The highly updated IMU readings (~100Hz) are treated as inputs to the system. and the velocities along X, Y axes calculated from encoders are treated as the measurements. Denote vectors with I as a subscript means accelerations, velocities or positions measured in the global frame while B means measured in the body frame.

The rover is modeled by the following kinematic equations representing the position of the mid-axis and the orientation in the global frame [6].

$$\begin{bmatrix} \dot{x} \\ y \end{bmatrix}_I = \begin{bmatrix} v_x \\ v_y \end{bmatrix}_I, \begin{bmatrix} \dot{v}_x \\ v_y \end{bmatrix}_I = C_B^I \bullet \begin{bmatrix} acc_x \\ acc_y \end{bmatrix}_B \tag{10}$$

Thus the state vector of the Kalman Filter can be written as $X = [x, y, v_x, v_y, \theta]_I$ and the input vector as $u = [acc_x, acc_y, \omega]_B$. According to the kinematics discussed above, the system $A(k)$, input $B(k)$ matrices for the Kalman Filter are given below:

$$A(k) = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, B(k) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \Delta t\cos(\theta) & -\Delta t\sin(\theta) & 0 \\ \Delta t\sin(\theta) & \Delta t\cos(\theta) & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \tag{11}$$

We consider the velocities along X, Y axes in the body frame $[v_x, v_y]_B$ as the measurement vector z for the KF design. The velocity $[V_l, V_r]$ is directly obtained from the encoder readings, and the relationship between the encoder reading is given as follows (assuming that there is no slippage).

$$\begin{aligned} v_x &= 0 \\ v_y &= \frac{1}{2}(V_l + V_r) \end{aligned} \tag{12}$$

The estimation of yaw angle θ is calculated by directly integrating the angular velocity given by the IMU. Therefore, the transformation matrix C_B^I given in Eq. (9) and its transpose matrix C_I^B is then calculated. The measurement vector $z = [v_x, v_y]_B$ can be related to the current state using the following measurement matrix:

$$H(k) = \begin{bmatrix} 0 & 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & 0 & -\sin(\theta) & \cos(\theta) & 0 \end{bmatrix} \tag{13}$$

C. Localization simulations

The filter was used to estimate the rover state (position and orientation) in planar terrain by fusing the odometry, FOG, and IMU data. It takes the highly updated (~100Hz) IMU raw data for the prediction step and relatively lowly updated (~10Hz) encoder data for the time update step if available.

Assuming that the system noises are uncorrelated and time-invariant, we will get a diagonal and time-invariant system noise covariance matrix. The system position noise standard deviation for the X, Y coordinate, the system velocity noise standard deviation in the fixed frame and the orientation noise standard deviation are guessed as follows:

$$\begin{aligned} \sigma_x &= \sigma_y = 0.01m \\ \sigma_{vx} &= \sigma_{vy} = 0.01m/s \\ \sigma_\theta &= 0.01rad \end{aligned} \tag{14}$$

We then obtain the system noise covariance matrix

$$Q = 0.01 \bullet I_5 \tag{15}$$

The KF initial state $X(0)$ is taken to be equal to zero and the initial state error covariance matrix is initialized to be equal to the system error noise covariance

$$P(0) = Q. \quad (16)$$

For the prediction step of the KF estimates, the input noise covariance matrix is initialized as

$$\gamma = 0.01 \bullet I_3$$

For the time update step, the measurement noise covariance matrix is set to relatively smaller values

$$R = 0.0001 \bullet I_2$$

D. *Experiments and results*

The acceleration of IMU raw data suffers from great drift, especially after a long period of time. While at the same time, the wheel odometry cannot providing convincing information when the rover is pivoting or encountering lateral slippage. So the robot was set to run in a square shape which contains pivoting part to test the performance of the Kalman Filter.

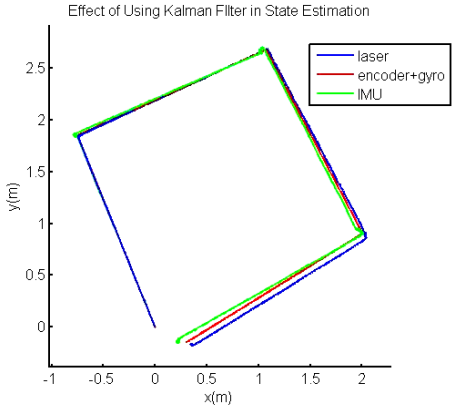


Fig. 4. Robot state estimation by Kalman Filter compared to estimation from differential drive equation and absolute position from laser data

From the trajectory recorded by the laser range finder, it is explicit that when the rover pivoted at the corner, it underwent obvious shake and slippage all the time. However, the wheel odometry method which used the encoder and FOG pair shows no roughness at the four corners at all. Namely, it can not convey information about the rover's state under some circumstances like pivoting, which is quite meaningful in rover controlling. The Kalman Filter, though did not show much improvement in the overall estimation, it clearly reflect the shake information when pivoting. This is because it utilizes the IMU data for the prediction step. So the Kalman Filter offers a nice estimation of the rover's state as wheel odometry does, while at the same time, providing the slippage information.

IV. CONCLUSION

Localization for wheeled skid-steer vehicles are challenging because of the complexity of kinematics and wheel/ground interaction. This paper provides two means of estimating the lunar rover's 2D state by fusing information from wheel odometry, fiber optic gyroscope (FOG) and inertial measurement units (IMU). The first method applys a differential drive model as a simplification and uses wheel odometry and FOG to estimate the rover's state. The second method employs a discrete Kalman Filter which combines encoder and IMU data together to provide estimation for the rover. These two methods are easy to implement on a rover and have different applications. While using a fiber optic gyro can reduce heading error caused by lateral slippage, the Kalman Filter is able to calculate the rover's state without introducing the IMU's drift resulting from integration of accelerations and non-systematic errors of wheel odometry like slippage. The performance of these two methods are tested and validated with a laser range finder.

Although our tests only cover several simple trajectories, we are optimistic that these two methods will give good performance in more complex motions because our tests have covered basic parts like pivoting, turning, walking forward and backward. Intricate trajectories are composed with these primary parts.

ACKNOWLEDGEMENT

I would like to sincerely thank Uland Wong, William "Red" Whittaker, Heather Jones, Kevin Peterson, Chris Cunningham, and Chuck Whittaker for their kind help. Special thanks to my teammate Tao Fu.

REFERENCES

[1] Borenstein.J and Feng.L 1996, *Gyrodometry: A New Method for Combining Data from Gyros and Odometry in Mobile Robots*, in Proc. of the IEEE Int. Conf. on Robotics and Automation, Minneapolis, Apr. 22-28, 1996, pp. 423-428.

[2] Mandow.A, Martinez.J.L, Morales.J, Blanco.J.L, *Experimental kinematics for wheeled skid-steer mobile robots*, in Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems, San Diego, USA, 2007, pp.1222-1227.

[3] Daniel Flippo, Richard Heller and David P. Miller, *Turning Efficiency Prediction for Skid Steer Robots Using Single Wheel Testing*, Springer Berlin Heidelberg, 2010, pp 479-488.

[4] Dudek and Jenkin, Computational Principles of Mobile Robotics. Online available:http://chess.eecs.berkeley.edu/eecs149/documentation/differentialDrive.pdf

[5] Yi.J.G, Zhang.J.J, Song.D.Z, Jayasuriya.S, *IMU-based localization and slip estimation for skid-steered mobile robots*, in Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems, San Diego, USA, 2007, pp.2845-2850.

[6]. P. Y. C. Hwang R. G. Brown. *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions.* John Wiley Sons, Inc, third edition, 1997.

Ander Solorzano

RISS 2013

Designing a PID Control Algorithm for a Lunar Tyrolean Robotic Survey Platform

Ander A Solorzano (ander@solorzano.com)
Rose-Hulman Institute of Technology '13
Carnegie-Mellon University Robotics Institute Summer Scholar Program
Supervisor: Dr. William (Red) Whittaker

ABSTRACT

This paper presents the methodology for designing and simulating a motion control algorithm for a Tyrolean-based descent and traversal robotic platform. A PID controller is developed which can perform a variety of traversal scanning trajectories including pulse and sine wave. The desired trajectory of the robot is first created using MATLAB and then utilized to set the velocities of each of the motors with respect to time. The path planning algorithm is modular and can be easily changed depending on the environmental constraints. The control algorithm deployed on the physical platform will then read the time v. velocity vector for each of the motors and compute the error between the ideal and the measurement to rectify its path trajectory. The parameters of a simulated plant model with a respectable uniform noise were used to calculate the desired PID gains of the motors. Simulation demonstrated that a stable controller with 5.82% overshoot, a settling time of 3.2 seconds, and control effort gain of 0.937 is achievable. The sampling rate of the discretized system done in the simulation is 0.005 seconds. A simulation for various velocities was performed to observe the steady state response of the controller.

***Keywords*—Tyrolean, PID, sky moonlight, moon, simulation, survey, discrete-time**

I. INTRODUCTION

First discovered in 2009 by Japan's Kaguya spacecraft and further inspected by NASA's Lunar Reconnaissance Orbiter (LRO), the "moon skylights" pose great interest for scientists since it can potentially serve as the foundation for possible colonization attempts due to natural protection from deadly electromagnetic radiation, meteorite bombardments, and large temperature variations [1][2]. Furthermore, scientists wish to explore these locations since it may consist of a network of underground lava tubes that can contain useful resources for future space exploration and colonization [1]. The mission objective consists of creating a robotic platform, called *Tyrobot*, which will be deployed to the moon in the near future. The robot is tasked with exploring the "moon skylights" or moon pit locations.

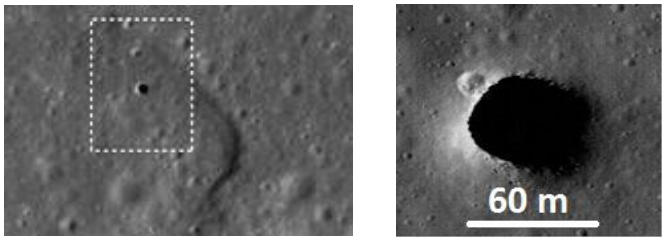


Figure 1. **Snapshot taken of the first moon skylight located on Marius Hills.** This picture was among the first set of photos captured by Japan's Kaguya spacecraft as it orbited the moon. The height of the dashed rectangle is about 1 km in length. The pit is large enough to fit the White House completely inside [1][2].

Figure 2. **A moon pit and the Capitol Building.** This figure helps grasp the scale and magnitude of the moon pit locations recently discovered on the moon.

This paper discusses and explains the control algorithm needed to deploy a robotic platform that can successfully explore and scan these locations in Earth-based test locations that resemble the moon pits. The *Tyrobot* consists of suspended platform that travels along a tightrope that is anchored down across the diameter of the pit. Once the robot is successfully suspended and deployed, it can traverse along the tightrope and lower or raise the carriage as commanded by a pre-determined path trajectory. As the robot moves along its trajectory, the attached sensor package will take thousands of laser scans and video data that can provide useful information about the composition and structure of these locations. Future improvements of the *Tyrobot* may even deploy a rover to autonomously explore the unknown surface and determine whether they consist of a network of underground lava tubes. These robots could be armed with radar-penetrating technologies to provide data and accurate models showing the stability and structural design of the underground lava tubes. Work on underground modeling applications on Earth has already been developed that maps out caves and tunnels using laser-range technologies [10].

Figure 3. **The proposed stages and lunar missions of the *Tyrobot*.** From left to right, the lunar lander will land in a location close to the moon skylight. It will then deploy a rover that will secure anchor points around the perimeter of the skylight and deploy the Tyrobot. The Tyrobot will then take scans of the wall structure and surface to discover information about the

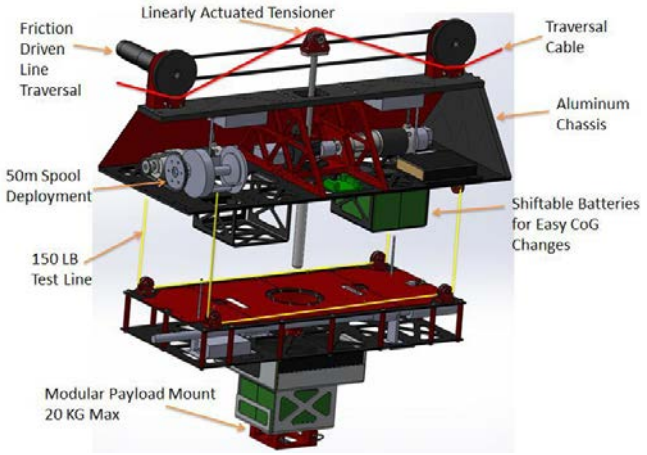


Figure 4. **Tyrobot robotic platform design.** This figure shows the overall design of the Tyrobot consisting of two sections: the carriage (top section) and platform (bottom section). The carriage houses all the actuators and motor controllers. The platform houses the computer and sensor package (not shown). The carriage and the platform communicate over radio signal.

Once the robot is deployed and suspended on the tightrope, the carriage will be able to traverse the width of the pit while lowering or raising the platform for data acquisition. The carriage's sensor packet, named *Ferret*, contains a LIDAR, an IMU, a camera, and a tracking prism. These sensors are used to capture video data about the pit and provide a 3D model. The tracking prism and IMU can provide the position and velocity of the platform as it moves. Sensor synchronization is provided via clock distribution system to match sensor scans, line tension, and position as measured by the ground survey system. The *Ferret* firmly attaches to the bottom of the platform (not shown) and remains static through the run.

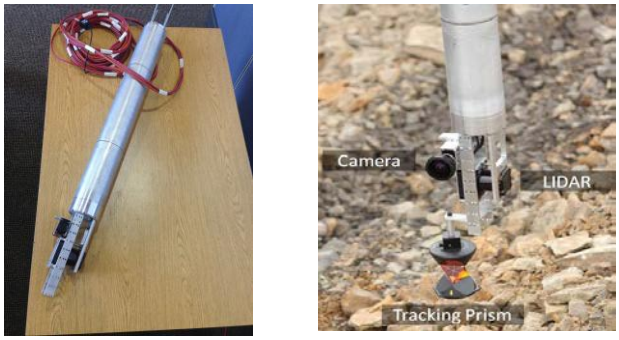


Figure 5. **Ferret sensor package.** This figure shows the sensor package that it is firmly attached to the bottom of the platform. This device is responsible for acquiring video data showing stratigraphic sequence of a rock wall and mapping a 3D model of a pit.

This paper focuses on the design, initial tests, and simulation of an algorithm that will control the traversal (x-axis) and winch (z-axis) motors via a PID closed-loop controller. Although other common controllers were also considered, like LQR and Ackermann pole-placement, a PID controller was implemented for this application to show proof of concept and due to the robustness and reliability that it can offer in uncertain domains. For future designs of the robot, a more complex and adaptive controller could be implemented once enough realistic field data is acquired.

Since this work consists of an innovative application, development of an effective control algorithm was based on several earth-based applications, like the control algorithm of a hoist, the control algorithm of the ACROBOTER, and the control theory of CMU's Ballbot, and mobile control lectures from MIT and Brown University [3-8]. Perhaps the most relevant work done that resembles a line survey platform consist of the *Expliner* robot [11][12]. This robot is used to perform tests and inspections on high-voltage lines while traversing along the lines. Several ideas and concepts concerning the mechanical design and control architecture were considered for the *Tyrobot.*

II. PROCEDURE

A. Plant modeling via simulated data and initial PID testing

The first step was to acquire some simulated data from a virtual motor since the actual device and unknown system was not constructed at the time. The simulated plant model acquired sets the motor velocities to a reference value and includes some simulated noise. The model records the Linux CPU time versus simulated motor velocity in ticks. A uniform 10% noise was added in the simulation.

In experimentation, the motor load should be incorporated in a physical test and measured by running in open-loop while creating a data log over time as velocities change to acquire a closer estimation of the motors by including inertia values, load of the motors, power consumption, and power output. The change in inertia and load can then be used to create a more accurate description of the plant which will then yield a more stable controller.

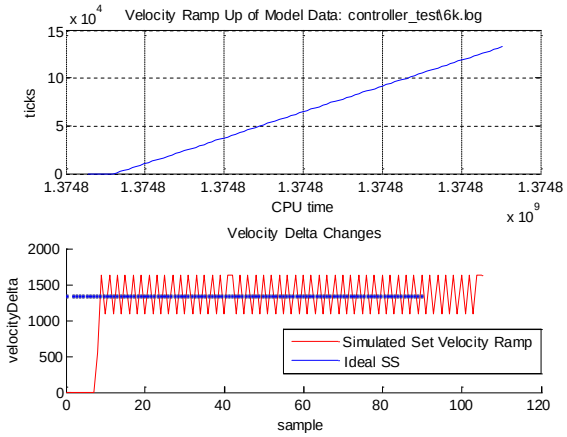


Figure 6. **Simulated plant model for PID tuning.** The figure on the top plots the accumulation of ticks as a function of CPU time. The figure on the bottom shows the change in tick position (i.e. velocity).

After acquiring a simulated velocity log file for 4 different settings, the corresponding differences of the motor velocity (in ticks) were plotted as shown in Figure 6. The ideal steady-

state value (Ideal SS) can then be computed after isolating the noise. This ideal reference value is then used to come up with the desired motor inputs and to compute the controller gains.

B. Designing and tuning a PID controller for Tyrolean traversal

The design and tuning of the PID controller was done using MATLAB software tools. Using Simulink and Sisotool utilities, a PID controller was designed and tuned for the simulated plant models that would minimize the error or difference between the actual value and the command value [5][6]. At the time, several physical constraints of the robot were estimated as shown below.

Known system requirements for *Tyrobot*:

- Max traversal speed: 20 cm/s
- Max raise/lower speed (i.e. winch motor): 20 cm/s
- Operational time: 2.5 hours
- Max slope: 20 degrees
- Max payload power draw: 50 W
- Max deployment depth: 50 m
- Max payload mass: 15 kg
- Max carriage mass: 25 kg

Types of operation expected for the *Tyrobot*:

- Data acquisition while traversing tightrope;
- Raise/lower platform and traverse to follow pre-planned trajectory as defined by ground operator;
- Allow radio control communication for emergency retrieval;
- Computer communicate wirelessly to carriage compartment containing motor controllers;
- Increase angle wrap of tightrope to prevent slip during operation;
- Device subjected to pendulum motion and environmental changes such as wind;

The constraints and controller requirements taken into consideration consisted of having a percent overshoot (P.O.) of less than 15% and a rise time of less than 5 seconds. In an ideal scenario, these constraints would correlate to having a control architecture that achieves stability within 5 seconds. Since the robot's max speed is a slow 20 cm/s, achieving steady-state operation in this time makes the robot more reliable and robust to changes caused from abruptly changing speeds. However, due to the nature of the simulate log file for the motors that was used to create the controller, physical system limitations are not being taken into account. This includes the control effort, given the max speed, that the motor can physically attain and stall torque power when the motors need to hold the rotational speed of the platform at zero. The controller was discretized using zero-order hold and a sampling time selected consisted of 0.005 seconds was selected.

The controller takes in desired and simulated velocity commands and sets the gain for the unknown system to achieve those values within the specified constraints. In the physical implementation, position and velocity feedback is provided via encoders attached to the motors. Using the clock synchronization from the computer, the control algorithm deployed on the survey platform will receive a list of velocity commands with a time stamp attached to it. It will then verify that the actual velocity commands are matching the reference values and corresponding timestamps.

	Run 1	Run 2	Run 3	Run 4
Sampling Time (s)	0.010	0.010	0.005	0.005
Filter Coefficient (N)	2.430	2.430	2.430	2.430
Rise Time (s)	1.360	0.670	1.030	0.340
Settling Time (s)	4.210	2.110	3.200	1.060
P.O. (%)	5.880	6.040	5.820	6.040
Controller Effort	0.709	1.400	0.937	2.830
Kp	1.101	2.197	1.451	4.383
Ki	0.010	0.851	0.017	0.154
Kd	-0.519	-0.519	-0.519	-0.519

Table 1. **PID configurations with varying controller constraints.** The controller gains were computed for various controller requirements including rise time, P.O., and sampling time.

Each of the controller configurations shown above were modeled and tuned to determine a feasible result for the simulated plant. Configuration 3 (i.e. Run 3) was chosen due to its settling time, overshoot, and control effort gains (highlighted). The process for selecting these parameters relied on the control system requirements and the type of application being developed. In a physical scenario, the controller's max effort gain is estimated to be around 0.93 thus allowing a gain of 0.07 (i.e. 7%) in case more effort is needed to achieve settling motion. Since fast and abrupt changes to the velocities are detrimental to the *Tyrobot*'s operation, a settling time of 3.20 seconds is acceptable. Once the controller gains were set, the controller was tested for various inputs to observe its stability and settling time. Also, only PID configurations were tested (i.e. P,D, PI, PD not considered) since the P and D controller affect the rise time and P.O. while the I controller helps with the steady-state settling time [5][6][9].

C. Path making program for Tyrolean survey

To take data of area of interest (e.g. a moon pit or an Earth-based test location), a path planner algorithm was created that generates a velocity command file containing time, traversal motor velocity, and winch motor velocity vectors. Although physical data was not available at the time of this work, the paths created are vertical in path to minimize the work of the winch motor due to gravity.

The program that makes the desired Tyrolean trajectory (pulse wave or sine wave) is modular and can be easily changed to accommodate for the test site. In particular, the width and height of the pit location (in meters), an offset distance (to account for unexpected irregularities in the site and hanging objects from the Tyrobot), and the period of the wave can be edited and entered as needed. Once the pit

settings are measured and defined, the user needs to specify a run time (in seconds) for the data gathering trajectory. The top left coordinate of the pit is defined as the origin (0,0) while the winch motion of travel goes along the negative z-axis.

The functions used to make the paths are parameterized with respect to time so a velocity-time command file can be created. The algorithm does not take into account the slack of the traversal rope. It assumes an ideal case scenario where the traversal rope is perfectly tensioned. Thus the physical experimentation will be initially affected by these unaccounted forces. These repercussions should be dealt when data becomes available.

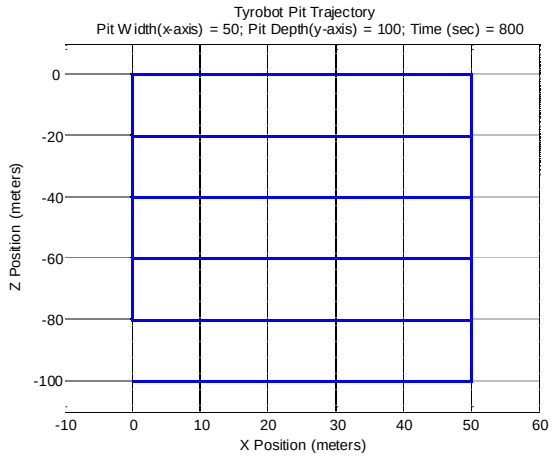


Figure 7. **Pulse wave Tyrolean trajectory.** The user can select a step-like trajectory for data gathering once the pit dimensions and run time are properly tested. Here the robot starts at the origin and proceeds along the blue line. Once it gets to the bottom, the Tyrobot does a second sweep of the bottom before proceeding upwards if time remains.

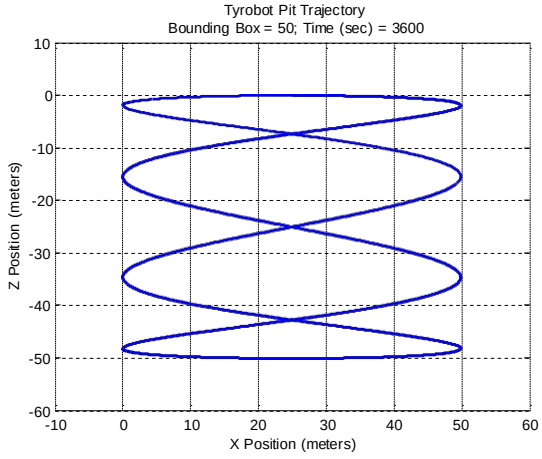


Figure 8. **Sine wave Tyrolean trajectory.** The user can select a sine-like trajectory for data gathering once a bounding box dimension of the test area (in meters) and the run time are specified. Here the Tyrobot starts at the top middle point and proceeds along the blue line. If time remains, the robot just keeps on repeating its path.

From the generated path files, the position of each of the motors can be known independently and plotted versus time.

Although not currently designed for use, another file containing the position and time of the traversal and winch motion is also created by this program. This can later be tested against encoder feedback and data from the tracking prism to obtain the error of the trajectory and refine the control algorithm to take into account such errors. From the position-time command file and plot, the derivative is then taken to compute and output the velocity-time command file for the traversal and winch motors.

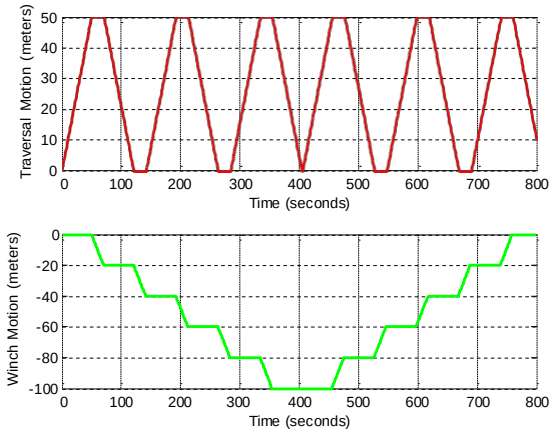


Figure 9. **Position versus time plot for the traversal and winch motions when using a step wave.**

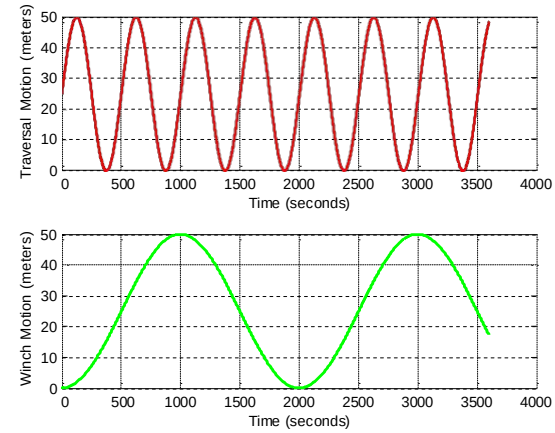


Figure 10. **Position versus time plot for the traversal and winch motions when using a sine wave.**

From the parameterized traversal and winch motion commands, the positions of the motors with respect to time can be known. In the figures above, the red plots represent the traversal positioning while the green represent the winch positioning through the timed run for data acquisition. It is important to keep in mind that the step wave increments or decrements discretely which in turn affects the velocity commands by changing the commands abruptly rather than gradually. Thus the sine wave poses a trajectory that can be stable due to its gradual motion.

Looking at the velocity plots (Figure 11 and Figure 12), one can predict how the changing of speeds will alter the swinging

motion of the platform especially when the platform is at its lowest points. Simply changing the gain of the speeds and having the platform operate very slowly (e.g. max motor velocity for traversal and winch is 10%) can dampen the swinging motion of the platform especially as it reaches its lowest descent points. To achieve higher stability and minimize the oscillations due to swinging motion, data from entire testing scenario that incorporates the pendulum motions, the slack of the traversal rope, and the external forces acting on the Tyrobot can be modeled to tune the control algorithm.

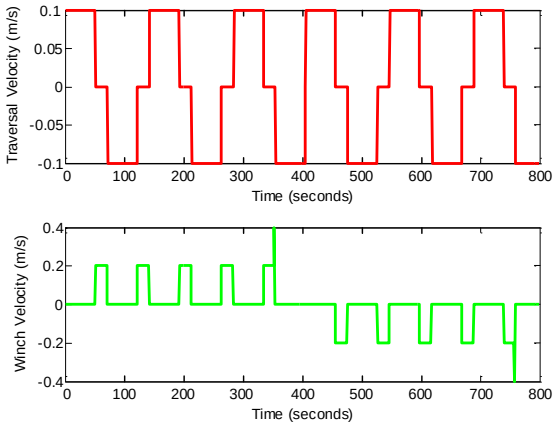


Figure 11. **Velocity versus time plot for the traversal and winch motions when using a step wave.** Using a limiter of 10% output, the max traversal velocity is 0.1 m/s (system requirement). The winch was capped at 20% output.

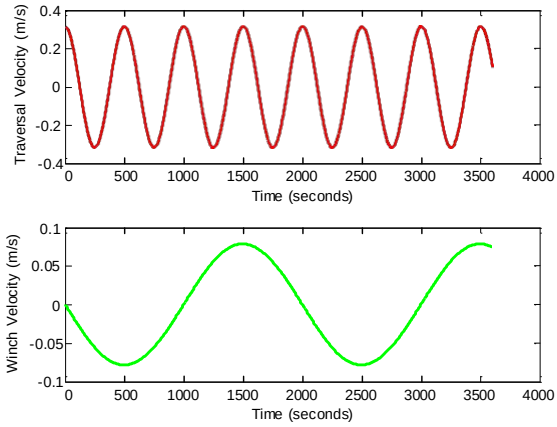


Figure 12. **Velocity versus time plot for the traversal and winch motions when using a sine wave.**

III. RESULTS

A. *PID simulation results for Tyrolean velocities*

After obtaining the desired controller configurations and controller gains from some desired system requirements, the controller was tested for different speeds to achieve its steady-state response. The simulation shows that the stability of the scenario is achievable in less than 5 seconds with less than 10% overshoot. Figure 13 shows the simulated tests done in MATLAB.

The next step would be to test the controller against the physical hardware in a real testing scenario to observe a more adequate system response. Due to lack of raw data and unexpected forces not taken into account when performing the simulation, it is expected that the controller might need tuning after several dry runs. The control effort yielded by the controller maintains the motor under 100% of maximum output.

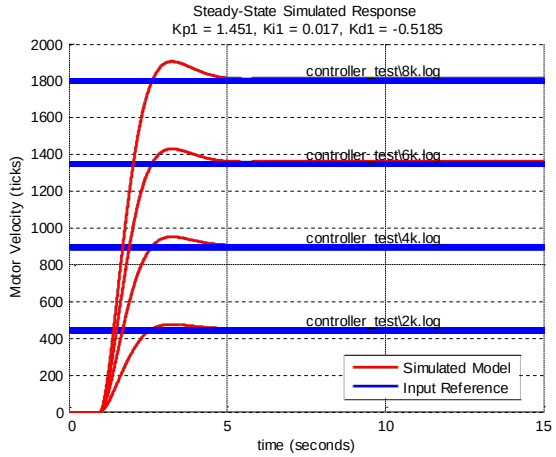


Figure 13. **PID controller steady-state response.** This plot shows the working PID controller done in simulation for several velocity reference inputs.

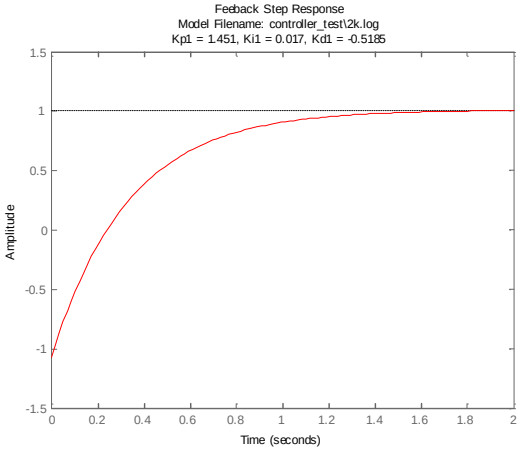


Figure 14 **Feedback step response plot.** This shows the feedback step response of the simulated controller.

IV. FUTURE WORK

Future work will focus on tuning and perfecting the controller once actual test data becomes available as scenarios of the real surveying conditions are performed.

In the short term, possibilities for tuning the controller include gathering time data logs with motor velocity and position values as reported by the encoders. From there a more accurate description of the plant can be correctly modeled and a controller can thus be designed that also includes acceleration and jerk. By limiting the acceleration, the stability and robustness of the control architecture can be improved. Also, the corresponding velocities of the motors, in either cm/s or m/s, have to be measured against ticks/s when the motor is under load conditions. From there a more accurate representation of the path and velocity command file can be created.

In the long term, a more accurate projection of the trajectory can be obtained that takes into account the swinging motion of the pendulum as a function of distance away from the carriage. The errors and differences of the path trajectory due to the slack of Tyrolean rope can also be calculated and accounted for. In addition, IMU data (accelerometer and gyroscope) from the sensor package located in the platform can be used to minimize the damping of pendulum, thus reducing uncertainties in the system.

V. SUMMARY

My work consisted in creating and simulating a control algorithm for a Tyrolean survey robotic platform that will be launched to the moon. Simulation validates that stability of the system is achievable in 3.2 seconds with an overshoot of 5.82% and control effort gain of 0.930. Tests of the physical device are needed to perfect the control algorithm for smooth data acquisition. Future work will focus on perfecting the algorithm by implementing unaccounted variables.

VI. REFERENCES

[1] B. Handwerk. (2009, 10, 26). *First Moon "Skylight" Found – Could House Lunar Base?* [Online]. Available: http://news.nationalgeographic.com/news/2009/10/091026-moon-skylight-lunar-base.html

[2] D. Coulter. (2010, 7, 12). *Down the Lunar Rabbit-Hole.* [Online]. Available: http://science.nasa.gov/science-news/science-at-nasa/2010/12jul_rabbithole/

[3] W. Su. "A Model Reference-Based Adaptive PID Controller for Robot Motion Control of Not Explicitly Known Systems." Vol. 12. No 3. September 2007. Int'l Journal of Intelligent Control and Systems

[4] G. Stepan *et al*. "ACROBOTER: a ceiling based crawling, hoisting and swinging service robot platform," Budapest Univ. of Tech. and Econ.

[5] C. Jenkins. "Lecture 4: Control Theory and Robot Dynamics" in *CS148-Building Intelligent Robots.* Brown Univ.

[6] C. Batten. "Control for Mobile Robots" in *Maslab IAP Robotics.* Jan. 2005.

[7] U. Nagarajan, G. Kantor, and R. Hollis. "Integrated Planning and Control for Graceful Navigation of Shape-Accelerated Underactuated Balancing Mobile Robots." May 2012. IEEE ICRA. CMU Press.

[8] U. Nagarajan, G. Kantor, and R. Hollis. "Trajectory Planning and Control of an Underactuated Dynamically Stable Single Spherical Wheeled Mobile Robot." May 2009. IEEE ICRA. Kobe Int'l Conference Center. CMU Press.

[9] R. Throne, *Linear Control Systems*, Rose-Hulman Institute of Technology, Chapter 9, 10, 11, 14.

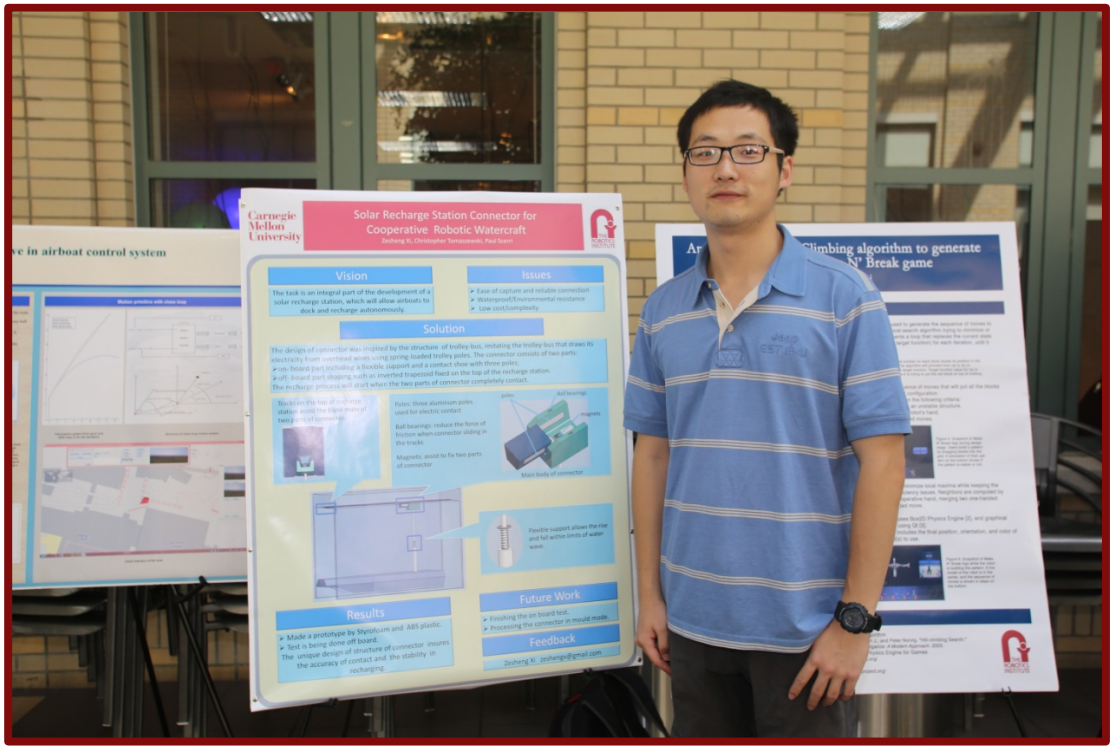
[10] U. Wong, A. Morris, C. Lea, J. Lee, C. Whittaker, B. Garney, W. Whittaker. *Comparative Evaluation of Range Sensing Technologies for Underground Void Modeling.* In Proc. Intelligent Robotics and Systems (IROS), 2011.

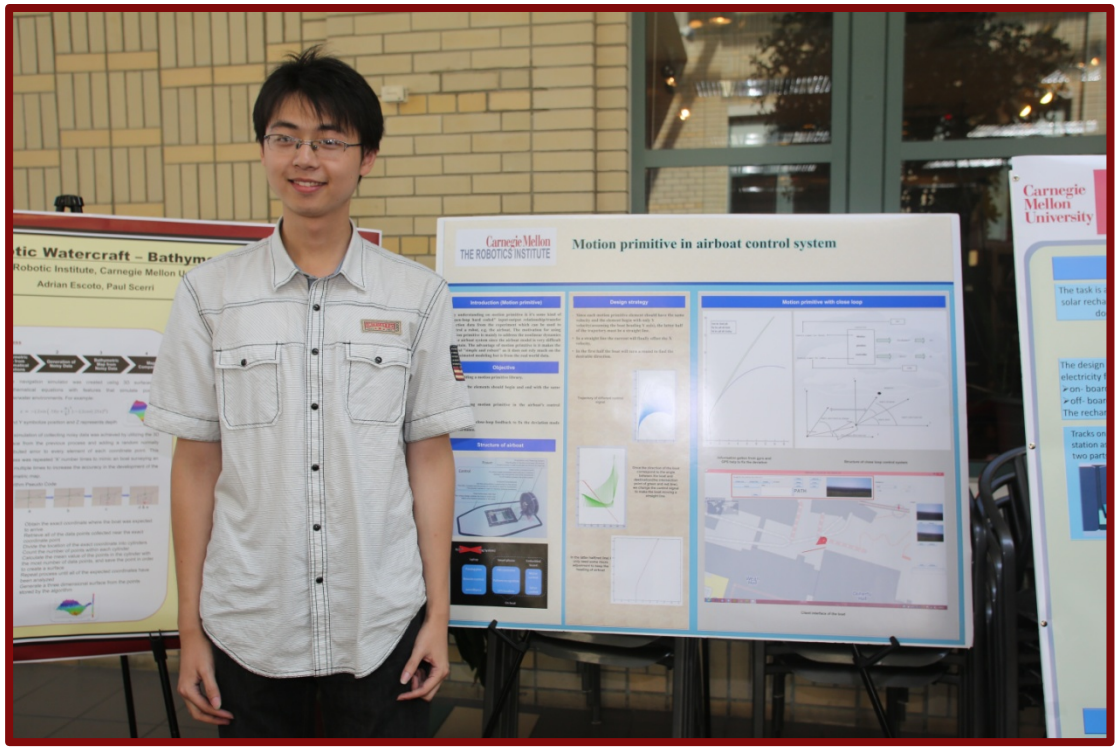
[11] P. Debenest, M. Guarnieri, K. Takita, *et al*. "Expliner – Robot for Inspection of Transmission Lines." May 2008. IEEE Int'l Conf. on Robotics and Automation, pp. 3978-3984.

[12] P. Debenest, M. Guarnieri, K. Takita, *et al*. "Expliner – Toward a Practical Robot for Inspection of High-Voltage Lines." May 2008.

Zesheng Xi and Yaonan Guan

RISS 2013





Airboats in Pittsburgh and Nanjing

Zesheng Xi , Yaonan Guan
Nanjing University of Science and Technology, Nanjing, 210094, China
zeshengx@gmail.com
tcgyn.student@sina.com

Introduction

Multi-robot systems (MRS) have great promise for revolutionizing the way a variety of important and complex tasks are performed. In our research as summer scholars with Carnegie Mellon Robotics Institute Summer Scholars Program,we are developing teams of Cooperative Robotic Watercraft (CRW) for critical applications including flood response, water monitoring and security.

A core design of our airboats is to use a commercial Android Smartphone to work as a CPU in our whole system, providing the computing, camera and communication for the boat. Moreover, using Android phones give us a relatively open and powerful development environment to try our idea. We work mainly in the Smartphone because of the access to multiple modes of communication like Wi-Fi and all kinds of data transferred from the sensors. For communicate with sensors, motors and servos, we use a relatively inexpensive microcontroller board named Arduino Mega to provide an array of digital and analog I/O for controlling the fan shroud, gyros and other external sensors modules.

Our job in lab

What we've done in our lab is to change the control system from PID to Motion Primitive. Unlike PID, a close-loop control method, Motion Primitives are elementary state trajectories used to produce motions in the position space. We can select appropriate elements based on the river environment parameters and combine them sequentially to produce more complicated and graceful trajectories. We created all the elements in Matlab and the boats worked pretty well in the simulation. In our strategy the boats could autonomously change the gain of thrust based on the trajectory elements. The gain will be increased if the boats only need to go straight to the waypoint while the gain will be decreased to move smoothly in turning.

Test in Pittsburgh

The first test was done at a lake near CMU. Our boats were only turning circles during the whole test because of the inexact model of the boats used to create all the elements. We took George's advice to build a close-loop system to help our boats follow the trajectories. Notice that both the great inertial and current influence caused significant overshoot when we try to control the direction of the boats. We refer to the PID algorithm and add the idea of differential in our close-loop system. We tested our code several times at the lake near CMU, where the water was calm, similar to the environment in my simulation.

Near the end of August, John, Ardalan and I took two boats to Ohio, the longest river in Pittsburgh. Testing was performed in several parts of the river with different current influence. This test showed that this Motion Primitive strategy did not take advantage in the straight line as we originally expected, which caused us to reevaluate our control algorithm. The boats could not keep their direction in the straight line due to the current influence and the monotone input signal in straight line. So we apply PID to handle such slight change in direction.

Another test was done in the activity “River Quest”, in which each group made their own boats and competed with others with their boats. We sent three boats, two of which ran the PID and one ran my Motion Primitive. Winds cause significantly larger waves than the boats had encountered before, which reduced performance and the boats were not able to come back automatically because of the low gain in thrust we set for smooth turn.

Fig.1. Final test in Ohio River

Airboats arrived in China

After 20 days leaving from Pittsburgh, we met the Airboats again in Nanjing, China. Flying thousands of miles across Pacific Ocean, the boats arrived in China on September 15. As we know, a good gain takes long pain. The boats were stuck in customs once they arrived. Fortunately, after we translating some necessary document, the boats finally ended their long trip and arrived at the destination----- Nanjing University of Science and Technology (NUST).

Assembling

Opening the package of the boats is as excited as opening the Christmas present. It’s amazing that you can receive a package that contains two robots. Before that, in our mind, robot should be complicated, delicate and hard to maintain. You can barely image to use common express to transport robots across half of the world and without worrying about some damages on them. In fact, we spent less than two hours to reassemble two boats by using some simple tools, then, following the checklist, do every step to test the boat. Consequently, both of boats are in good condition, which prove the robustness of Airboat, again.

Fig.2. Complete Airboats

Test in NUST

To celebrate the 60th anniversary of NUST, we did the demo of the Airboat in a pond within campus. Actually, it's the first time for Airboat running in China and it adds another spot in the track of Airboat's worldwide test, beside Australia, Philippines and Qatar. The demo caught the eyes and made many people to be curious about the Airboat. Since the interface for Airboat is designed user-friendly, operators can use the console to control boat with simple train. Thus we can take easy to let people try boat by themselves. We found that is a good approach to extend robotic technology by showing people some robots, which can be easily used by no specialist.

Fig.3. Doing the demo in campus

Attending an exhibition

Just in time, there is a science park, near the campus, holding an exhibition about robots. We bring the Airboat to attend. Comparing with other robots, the Airboat seems to be weird, since others are much more complicated in structure. However, these robots needs to be prepared and modulated in several days before demonstrating, while Airboat can be used directly once we arrived. Also, Airboat is the cheapest robot in the

exhibition and it can be an autonomous platform for a broad set of applications including water quality monitoring, flood disaster mitigation and depth buoy verification. Therefore, Airboat is practical and flexible for multi-purpose application.

Fig.4. Demonstrating Airboat in the exhibition

Workshop by Prof. Paul Scerri

We are appreciated that Prof. Paul Scerri can give us several precious workshops, sharing his inspiration and experience about multi-robots system, especially Airboat. His masterly speech deeply impressed us and broadens our vision in robotic field.

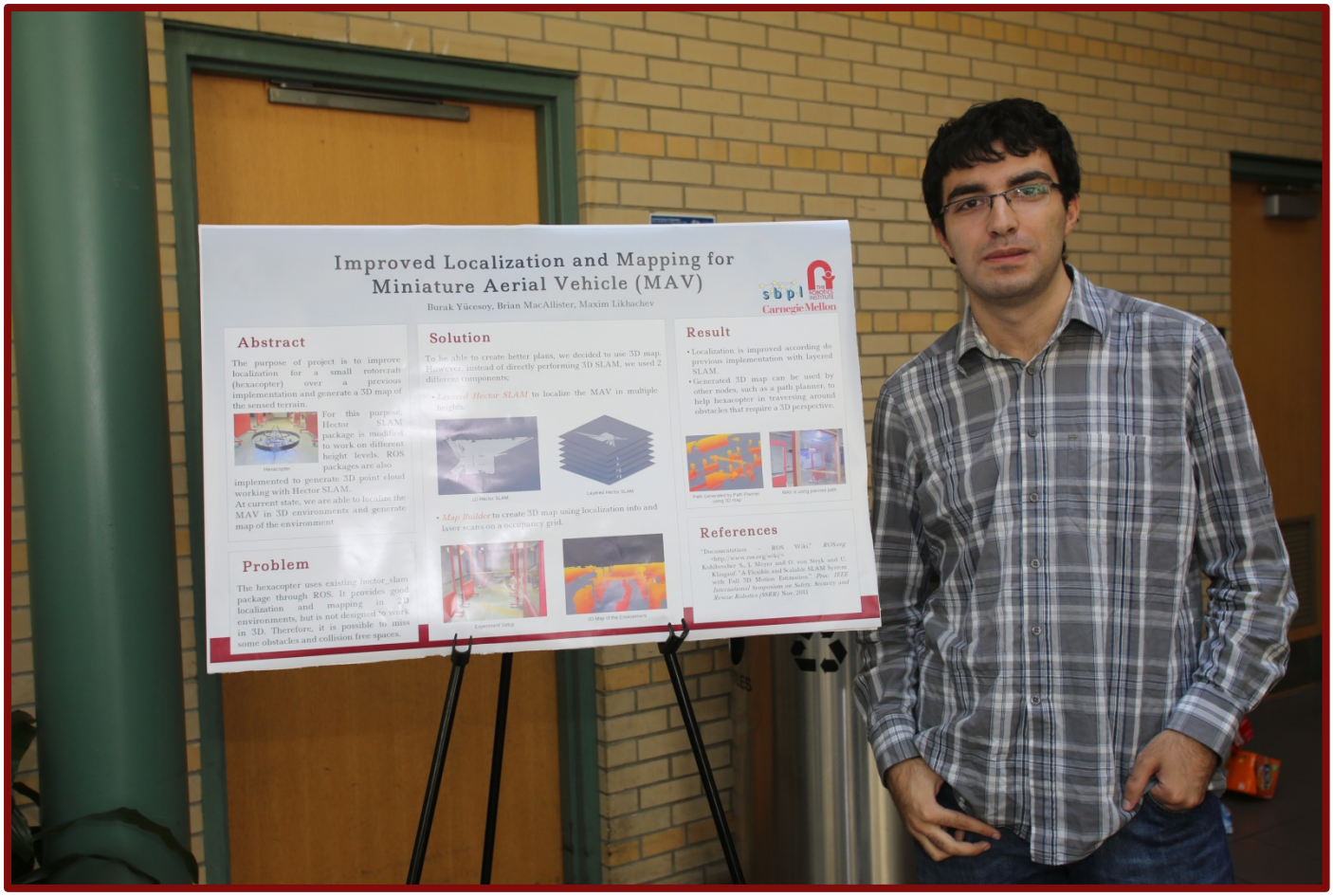
Fig.4. Prof. Paul Scerri is giving a presentation about information collection

Acknowledgements

The experience in Pittsburgh will be an unforgettable memory for us. We would like to express gratitude to our supervisor Paul Scerri and all those who have helped us during this summer and will continue to help us in future. Also, it's glad that we can meet such talented students and become friends with some of them in Pittsburgh.

Burak Yücesoy

RISS 2013



Improved Localization and Mapping for Miniature Aerial Vehicles (September 2013)

Burak Yucesoy

***Abstract*—Online mapping of the environment has been a growing interest for robotics researchers in many applications. The main focus of this study is to create fast, accurate and online 3D maps to support flight planning algorithms for aerial robot platforms. To accomplish this goal, we modified the existed Hector SLAM library to localize our robot platform in 2D maps of the environment at different altitude levels. Then, we used height and orientation to create 3D map of the environment. The performance of our 3D map is tested on an aerial robot platform following the plan generated by using our map to navigate around 3D obstacles. Flight planning algorithm created collision-free flight plans with the use of proposed 3D online mapping algorithm. We also showed that the aerial platform can perform the new flight plans during its flight.**

***Index Terms*— Simultaneous Localization and Mapping, 3D SLAM, Aerial Vehicle, Aerial Platform, Flight Planning**

I. Introduction

MOST online mapping techniques prefer the creation of 2D maps, since 2D mapping can be done more accurately with relatively less computational power than 3D maps. In fact, 2D maps are accurate enough for most cases where the robot's movement is restricted by two axes and there is no considerable amount of movement in the third axis. However, this is not the case for aerial vehicles, which are supposed to move in 3D environments. Working with aerial platforms also requires maintaining an accurate 4D position estimate of the platform while they are moving at a reasonable speed. Representing a 3D environment with 2D maps causes the flight planner to lose some relevant information about the environment, which can have significant importance for path planning.

Consider the case that robot platform tries to find a path to navigate from one point to another. In a 3D environment, such a flight plan may require movements in all three axes. Hence, working with only a 2D projection of the 3D environment may result in failing to find the optimum path or may result in no possible solutions even if they exist. For example, a 1 m × 1 m × 1 m box will cover its projection in a 2D map and the flight planner will try to avoid this region for creating a flight plan. However, there will be some free space above the box in a 3D representation of the environment, which can be used to create a more efficient flight plan.

The case mentioned above is just a simple example of the problems which can be encountered during flight planning in a 3D environment with 2D maps. Therefore, 3D perception of the environment is necessary for high performance locomotion of aerial vehicles. However, 3D mapping of an environment may require high computational power and its online implementation can be problematic for most robotic systems. Thus, we propose a layered 3D SLAM approach to create fast 3D maps of the environment by utilizing the Hector SLAM [1] algorithm.

Our mapping algorithm divides the height axis of the environment into thin layers with a fixed height. Measurement of the robot altitude gives us the layer in which our robot is currently moving. Then, the Hector SLAM algorithm is used for localizing the 2D position of our robot in the environment. Finally, our algorithm uses the 2D position information coming from Hector SLAM and orientation of the robot to register newly seen obstacles in the 3D occupancy grid.

II. Experiment Setup and Robot Specifications

A miniature aerial vehicle (Fig. 1) is used for experiments. The robot flies via its six rotors mounted on carbon fiber rods. PID controllers are used to regulate system behavior in different axes. The main computer of the robot runs Robot Operating System (ROS) on Kubuntu and all the software are implemented in C++.

Fig. 1. The robot used in experiments

Two Hokuyo UTM-30LX Laser Scanners are mounted to the main body of the robot. One of the laser scanners is facing

This paper was submitted to RISS Journal at 29 September 2013.

B. Yucesoy is an undergraduate student at Department of Computer Engineering, Bilkent University, Ankara 06800, Turkey (e-mail: burakyucesoy@gmail.com.tr)

down to measure the altitude of the robot and the other one is facing the front to sense the environment. The range of these laser scanners is 30 m and it has 270º field of view. These laser scanners work at 40Hz. The robot also has an inertial measurement unit to measure its orientation.

Besides the robot setup, it should be noted that the Hector SLAM algorithm. Hector SLAM is an open source SLAM algorithm, which is compatible with ROS. It uses the occupancy grid map approach, where the probability of being an obstacle is assigned to each cell in the map. It uses laser scan inputs to detect the obstacles in the environment. Below, you can see a 2D map generated with the Hector SLAM library (Fig. 2).

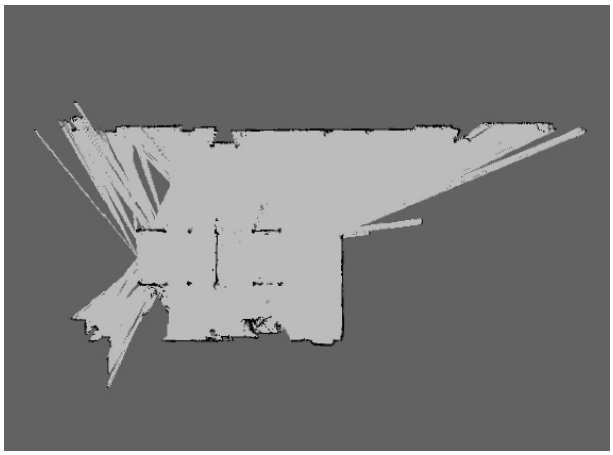


Fig. 2. 2D map created by Hector SLAM

III. CONTRIBUTION

The main contribution of this paper is to propose a new technique for fast and accurate 3D mapping of the unknown environment. The proposed layered 3D SLAM algorithm can be divided into three major parts: altitude estimation, localization and mapping.

A. Altitude Estimation

To measure altitude, a Hokuyo Laser Scanner is used facing downward. It returns the distances of the points below the robot. However, some of these points may come from obstacles below the robot rather than the ground. Therefore, the direct use of these points to estimate altitude will give an incorrect estimation since the distance of these points will not be the actual altitude of the robot with respect to ground. To have a better estimation of the robot height, we remove outlier points and then calculate arithmetic mean of the remaining points as the altitude of the robot.

B. Localization

At the beginning of the algorithm, we initialize 2D maps (layers) on top of each other with a fixed height. A sample illustration of these 2D maps can be seen in Fig. 3.

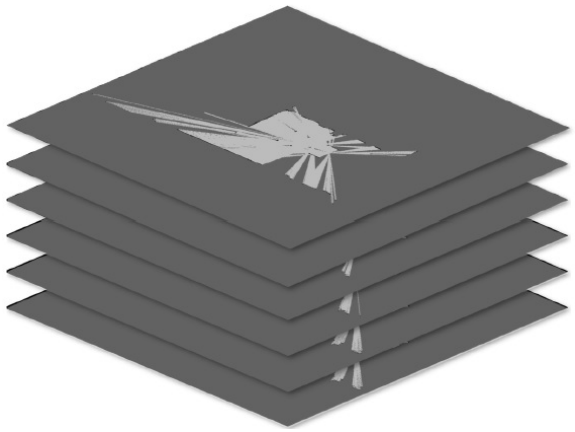


Fig. 3. Layers of 2D maps using for localization purposes

After estimating the altitude of the robot, our algorithm finds the layer, in which the robot is operating. Then, the Hector SLAM algorithm is executed by only using this 2D map. The position estimation part of the Hector SLAM algorithm requires an initial estimation to approximate the current position. In this step our algorithm uses the previous position estimation to feed a new run of the position estimation algorithm, even if the robot was in a different layer on the previous run of the position estimation algorithm. Even if the robot changes its layer, the change in the position will be small in two consecutive runs of the position estimation algorithm. Therefore, the previous position is a good estimation for the current position.

C. Mapping

In the previous steps of the algorithm, the altitude and position on 2D-plane are found. The inertial measurement unit is also used to find the orientation of the robot. After finding the position and orientation of the robot, we just register laser scans into 3D occupancy grid. In this step some points are filtered out based on the number of times they seen.

IV. EXPERIMENTS

In the experiments, a wooden stick is placed between the starting and target point (Fig. 4). In Fig. 4, the starting point is marked with “A” and the target point is marked with “B”.

Corresponding 3D map of this environment, created by the layered 3D SLAM algorithm, can be seen at Fig. 5.

Fig. 4. Experiment setup

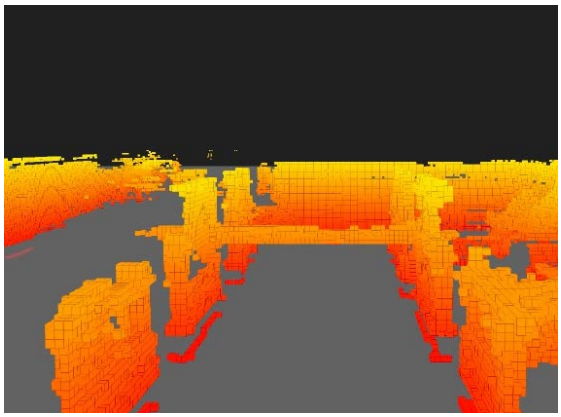


Fig. 5. Corresponding 3D map

For this setup, the planner creates a flight plan using the collision free space over the stick. In Fig. 6 the planned path for flight is illustrated as dashed green line. Our proposed solution, layered SLAM, can continue to localize the robot while it was following this flight plan that went over the stick and at the same time updating the map. In Fig. 7 the robot is shown while performing generated flight plan. In these photos, robot starts its flight, approaches to the stick, goes up, goes over the stick, goes down, approaches to its destination and lands respectively.

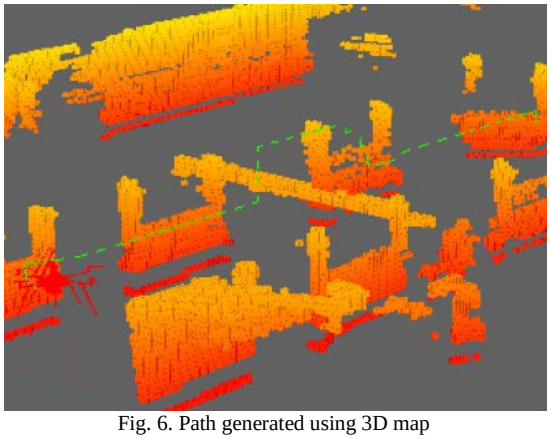


Fig. 6. Path generated using 3D map

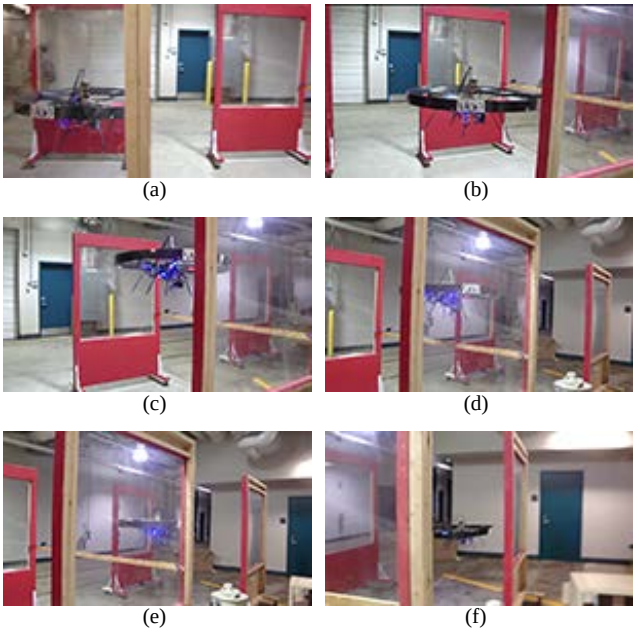


Fig. 7. Robot navigates in the environment using flight plan

V. Acknowledgements

I would like to thank Prof. Maxim Likhachev and Brian MacAllister for their supervision, Rachel Burcin and all CMU RISS coordinators for being kind, helpful and supportive during the RISS program, Prof. Uluc Saranli, Ismail Uyanik and Murat Kirtay for being excellent role models as academics.

VI. References

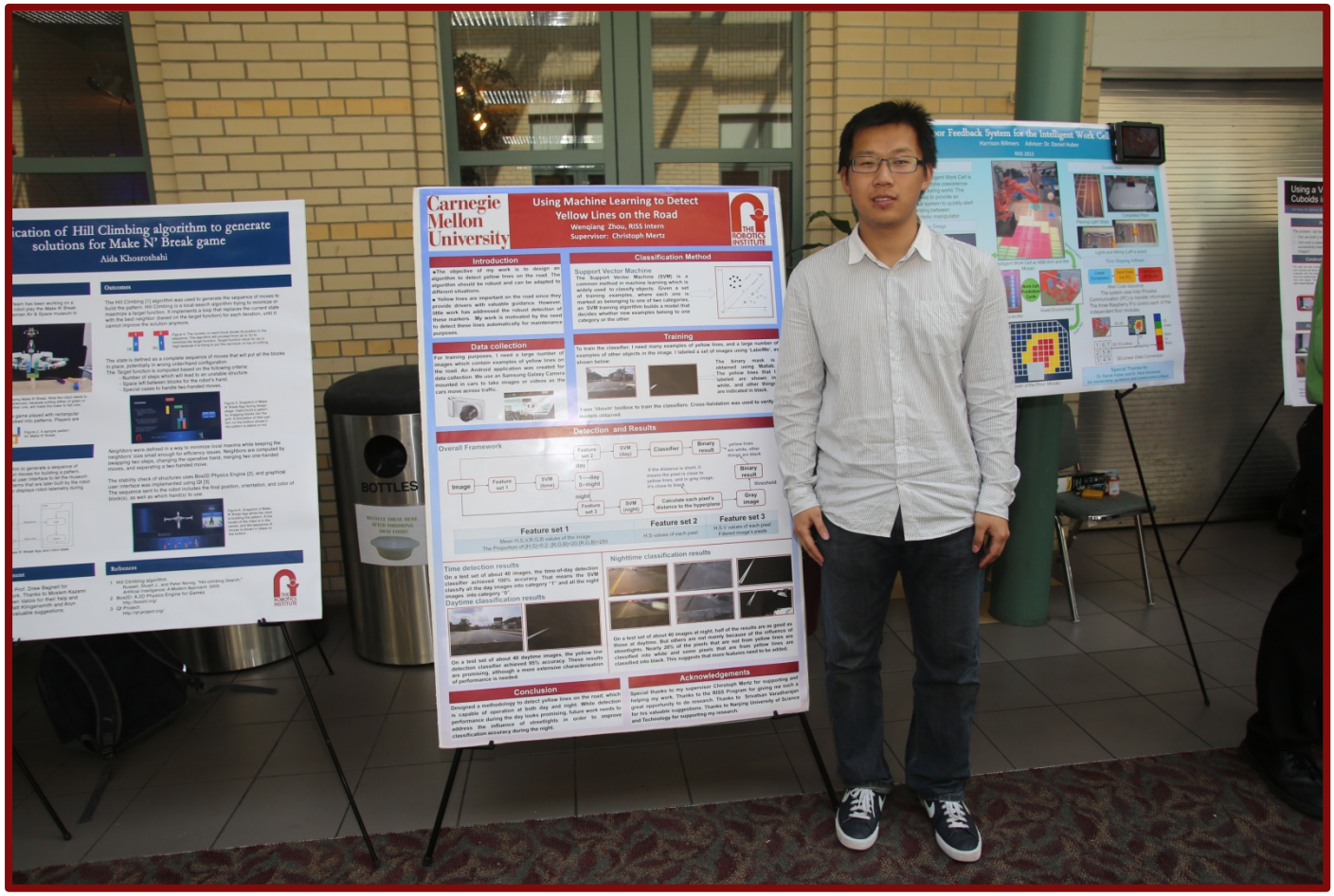
Kohlbrecher, S.; Von Stryk, O.; Meyer, J.; Klingauf, U., "A flexible and scalable SLAM system with full 3D motion estimation," Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on , vol., no., pp.155,160, 1-5 Nov. 2011

Burak Yucesoy is in his senior year to of a B.Sc. degree in Computer Engineering at the Bilkent University, Ankara, Turkey..

He was a Summer Intern at Search-Based Planning Laboratory, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, during summer 2013 as part of the RISS program. He is currently an undergraduate member of Bilkent DRACO Research Group.

Wenqiang Zhou

RISS 2013



Using Machine Learning to Detect Yellow Lines on the Road

Wenqiang Zhou, RI Summer Scholar, The Robotics Institute at Carnegie Mellon University

***Abstract*—A robust algorithm is proposed to detect yellow lines both at day and at night. A Support Vector Machine is used to do the classification. To train the classifiers, 'LabelMe', combined with the Matlab Toolbox, is used to collect a large number of yellow lines and many other objects from sample images. A time detection classifier is also trained to determine whether the image is taken at day or at night. Different feature sets are extracted to do the training and detection.**

***Index Terms*—classifier, feature set, Support Vector Machine, yellow lines**

I. Introduction

Yellow lines are important on the road since they provide drivers with valuable guidance. This work is motivated by the need to detect these lines automatically for maintenance purposes.

The objective of this work is to design an algorithm to detect yellow lines on the road. The algorithm should be robust and adaptable to different situations.

II. Classification Method

Support Vector Machine (SVM) is a common method of machine learning which is widely used to classify objects. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into the one category or the other.

Fig. 1 shows an example of SVM. Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called support vectors.

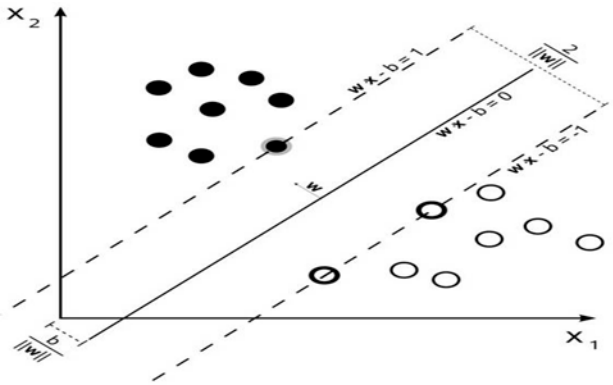


Fig. 1.An example of SVM

III. Data Collection

For training , many images were needed which contain examples of yellow lines on the road. An Android application was created for data collection. We use a Samsung Galaxy Camera mounted in cars to take images and videos as the cars move across traffic. The application transfers images using Dropbox and then uses Matlab to do the analysis. A picture of the camera and how it is mounted in a car is depicted in Fig. 2.

Fig. 2. Camera and its position in the car.

IV. Training

To train the classifier, many examples of yellow lines were needed, and a large number of examples of other objects in the image. We labeled a set of images using 'LabelMe', as shown in Fig. 3.

This research was conducted at the Robotics Institute at Carnegie Mellon University as part of the RI Summer Scholars Program.

Author Wenqiang Zhou is with the School of Automation, Nanjing University of Science and Technology, Nanjing, Jiangsu 210094 CHINA (e-mail:autozhouwq@gmail.com).

Fig. 3. An example of 'LabelMe'

The 'libsvm' toolbox was used to train the classifiers. Cross-Validation was used to verify the models obtained. The binary mask shown in Fig. 3 was obtained using Matlab. The yellow lines are shown in white, and other things are indicated in black.

V. Detection

To detect yellow lines, we need to consider different situations. Yellow lines are different at daytime or at nighttime. We extracted different features and trained two classifiers. Before detection, we need to know whether the image was taken at day or night. Therefore, we trained another classifier to do time-of-day detection. The overall framework is shown in Fig. 4.

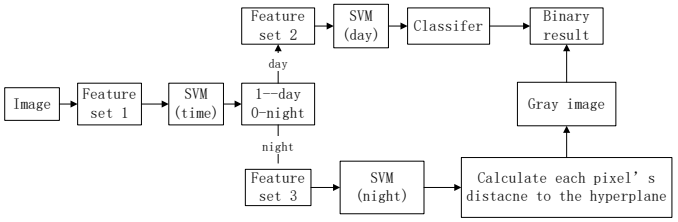


Fig. 4. Overall Framework of Detection

The feature sets used are listed in TABLE I.

TABLE I
FEATURE SETS FOR DETECTION

Feature set 1	Feature set 2	Feature set 3
Mean H,S,V,R,G,B values of the image The Proportion of:{H,S}<0.2; {R,G,B}<20;{R,G,B}>250	H,S values of each pixel	H,S,V values of each pixel Filtered image pixels

VI. Results

A. Time-Of-Day Detection Results

On a test set of about 40 images, the time-of-day detection classifier achieved 100% accuracy. That means the SVM classified all the day images into category "1" and all the night images into category "0".

B. Daytime Detection Results

On a test set of about 40 daytime images, the yellow lines detection classifier achieved 95% accuracy. These results are promising, although a more extensive characterization of performance is needed. Fig. 5 shows an example of daytime detection results.

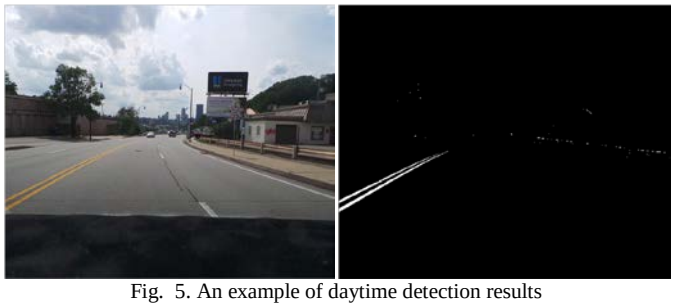


Fig. 5. An example of daytime detection results

C. Nighttime Classification Results

On a test set of about 40 images at night, half of the results are as good as those at daytime. But others are not mainly because of the influence of streetlights. Nearly 20% of the pixels that are not from yellow lines are classified into white and some pixels that are from yellow lines are classified into black. This suggests that more features need to be added. Some examples are depicted in Fig. 6.

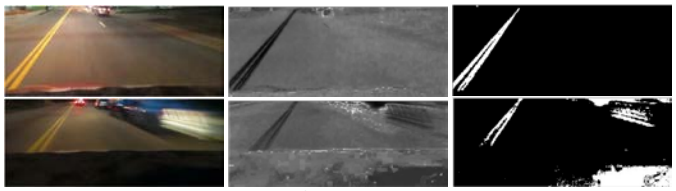


Fig. 6. Examples of nighttime detection results

VII. Conclusion

A methodology to detect yellow lines on the road was designed, which is capable of operation at both day and night. While detection performance during the day looks promising, future work needs to address the influence of streetlights in order to improve classification accuracy during the night.

Summary of My Experience and Future Work

A. Research

The Robotics Institute Summer Scholars Program at Carnegie Mellon University gave me a lot of opportunities to access high technology in robots. My research mainly focuses on Computer Vision (CV), which is a very hot topic these years. I also worked on Android Programming on a Ubuntu Operating System. After my summer research, I knew more about CV, including some hot research areas, using machine learning to do classification, filtering and some basic algorithms in CV. I also got more familiar with Linux Operating System and gained some experience in Robot Operating System(ROS). More importantly, by talking with my supervisor and some graduate students I know the important steps to do research and how to present my research.

B. Activities

Besides research, I also took part in many activities, such as presenting my research to American high school teachers and mechanical hull design contest. Through these activities, I worked with many students from all over the world and understood more about their cultures. I also improved my oral

English and the ability to communicate with students from different cultures. In addition, I made many friends both from China and other counties. I believe these experiences will benefit me a lot for my future career.

C. Future work

First, I will continue to improve my algorithm to detect yellow lines at night. I am considering extracting more features in order to improve detection accuracy at night. I hope that I can still collaborate with my lab to continue my research. Second, I want to detect other objects, such as cars or persons on the road, which is much difficult than detecting yellow lines. Since I got some experience in ROS, I would also like to use ROS to operate on intelligent robot 'NAO' in my home university.

ACKNOWLEDGMENTS

Special thanks to my supervisor Christoph Mertz for helping me with my research. Thanks to Srivatsan Varadharajan for his valuable suggestions. Thanks to the RISS Program.

Carnegie Mellon
THE ROBOTICS INSTITUTE

www.ri.cmu.edu/RISS