Omni-Directional Hovercraft Design as a Foundation for MAV Education

Carrick Detweiler, Brent Griffin, and Heath Roehr

***Abstract*— Quad-rotor Micro Aerial Vehicles (MAVs) are used widely in research and increasingly in commercial applications as the cost of these platforms has dropped. The cost of entry, however, is still high in large part due to the time and effort involved in repairing vehicles after crashes while learning about the system design and dynamics. In this paper, we present an omni-directional hovercraft, which has dynamics similar to MAVs and can be used as an educational platform to teach students about the behavior and control of MAV-like platforms with minimal cost and effort. Teaching students about the capabilities and challenges associated with MAVs is critical for educating future engineers and scientists that will develop and use the next generation of MAVs. In addition, the hovercraft provides a safe platform for researchers to test control and coordination algorithms before trying them on higher-cost MAVs.**

I. Introduction

Quad-rotor Micro Aerial Vehicles (MAVs) and Unmanned Aerial Vehicles (UAVs) are being used widely in research labs and are finding an increasing number of commercial applications as the cost of these platforms continues to drop. It is critical to educate students how to design and control MAVs to advance future research and commercial adoption. Traditional robotics education with wheeled robots fails to address many of the fundamental challenges associated with MAVs. In particular, MAVs cannot simply stop to process data and collect sensor information since their dynamics will likely cause their position to drift. This requires an educational focus on dynamics and real-time control and should be taught on a MAV-like platform.

In our MAV-focused research lab[1], we use a variety of MAVs ranging from platforms provided by Ascending Technologies [1] to Parrot AR.Drones [2] that cost approximately \$5-10k and \$300, respectively. At \$300, the Parrot AR.Drone seems like an ideal, low-cost platform to teach and educate undergraduate students about using and developing MAVs. Our experience, however, is that the low-cost platforms have limited payloads, closed-source firmware, and limited inputs for additional sensors. In addition, while learning how to control the MAVs, students tend to quickly and frequently crash these vehicles, greatly increasing the cost and effort involved in using MAVs with the classroom.

We are grateful to NSF IIS-1116221 and NSF CNS-1217400, the UNL CSE department, and a UNL Faculty Seed Grant for supporting parts of this work.

All authors are members of the Nebraska Intelligent MoBile Unmanned Systems (NIMBUS) Lab, Computer Science and Engineering, University of Nebraska–Lincoln, Lincoln, NE 68588, USA. carrick@cse.unl.edu, bagriffin@gmail.com and hroehr@cse.unl.edu

[1]Nebraska Intelligent MoBile Unmanned Systems (NIMBUS) Lab

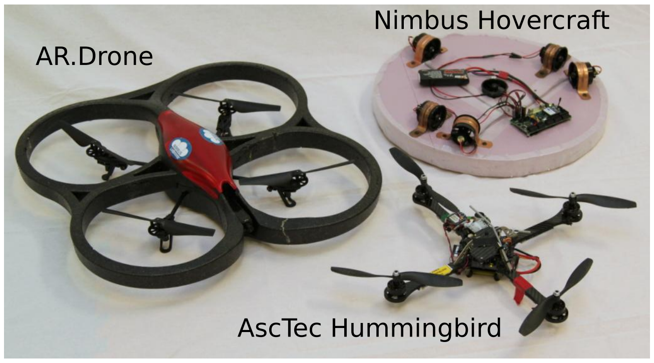

Fig. 1. MAVs and hovercraft in the NIMBUS Lab.

We have designed an omni-directional hovercraft, shown in Fig. 1, that has control architecture and dynamics similar to most quad-rotor MAVs. We have used this platform in the classroom to teach students about the fundamental challenges associated with developing and controlling MAV-like platforms in two courses: Embedded Systems (CSCE 436/836) and Robotics (CSCE 496/896). In addition, we use this platform in our research lab to prototype algorithms before moving them onto MAVs.

The hovercraft has a flexible design with typical diameters ranging from 12 to 18 inches. It is constructed from inexpensive and easily available materials. This encourages exploration of the design space and various configurations, since it is easy and inexpensive to try a different arrangement. For our classes and research we configure the thrusters to enable control of all degrees of freedom in the two dimensional plane. This makes the control and dynamics of the hovercraft very similar to a MAV.

The rest of this paper details the similarities and differences between MAVs and our omni-directional hovercraft. In Section III, we explore the dynamics of both vehicles and note that they are inertia-driven, which makes them significantly different from traditional wheel-based ground robots. In Section IV, we examine the low and high-level architecture of the hovercraft. At a low-level, we use custom designed circuit boards to control the system. At a high-level, we use Robot Operating System (ROS) [3] for control of the hovercraft, which makes it easy to substitute a hovercraft for a MAV when testing various algorithms. In Section V, we discuss our implementation and experiences using the hovercraft for a variety of tasks in the classroom. Finally, we conclude in Section VI by discussing limitations and future work. But first, we begin with the related work in Section II.

Robot	**Cost**
USC robomote	$150
iRobot Create	$220
Rice r-one	$220
UNL Hovercraft	$225-$275
LEGO Mindstorms	$250
CEENBoT	$300
Parrot AR.Drone MAV	$300
HandyBoard	$350
EPFL e-puck	$979

TABLE I

COST OF SELECTED EDUCATIONAL ROBOTS, SOME DATA FROM [4], [5].

II. RELATED WORK

With the advent of smaller and less expensive robotic platforms, purchasing robots such as the Roomba for educational purposes has become a viable way to teach students about advanced robotic principles without the need for large robotic laboratories [6], [7], [5]. Because small, yet capable robots can be purchased within the budget of a typical course, more educators can take advantage of unique opportunities to help students learn robotics with hands-on experiences. This makes it possible for students to learn about the challenges associated with the uncertainty in sensing and control on real robots. Table I summarizes the cost of a small subset of robots that are used in education, including the hovercraft developed in this paper.

In addition to ground-based robots, education and research on the control and applications of robotic quad-rotor MAVs has been rapidly expanding in recent years [8], [9], [10]. While the physical design and flight dynamics of quad-rotor MAVs is greatly simplified when compared to a fixed-wing aircraft or helicopter, the algorithms and heuristics used to control MAVs are quite complex [8].

The cost associated with purchasing and repairing MAVs (after inevitable crashes) has largely kept MAVs out of the classroom, except under highly supervised conditions that may prevent students from learning by experiencing mistakes. Quad-rotor MAVs are different from most traditional ground robots because they are omni-directional and largely inertia-driven. Ground based omni-directional robots have been designed and optimized for a variety of tasks [11], [12] and have also been used in education. However, MAVs differ from these platforms because their inertia is large compared to their input forces and they do not have friction based wheels (or wings in the case of an airplane) that can be used to turn the vehicle. In this paper, we develop an omni-directional, inertia-driven hovercraft robot that operates near the ground with characteristics similar to most quad-rotor MAVs. This inexpensive platform enables MAV research and education with a significantly lower operational cost.

In addition to hardware, software advancements with ROS have enabled rapid and reliable software engineering for robotic systems, both for educators and researchers [13]. ROS offers students a way to learn about fault-tolerant and scalable software engineering in the context of robotic engineering. We use ROS to enable modularity and the ability to use the same interfaces for both our hovercraft and MAVs (which have existing ROS interfaces).

III. SIMILARITIES IN DYNAMICS

The key characteristic that distinguishes omni-directional hovercrafts and MAVs from traditional ground-based omni-directional vehicles is that they have small input forces compared to their inertia. The speed and direction of an omni-directional wheeled vehicle can be directly controlled and adjusted by turning its wheels [14], [15], which requires little energy. Hovercrafts and MAVs, however, are subject to a much higher ratio of inertia to resistance. In order to shift directions, the current momentum must be countered with a significant new input force. If active control is not applied, it will continue to drift and/or rotate.

Hovercrafts have the same number of degrees of freedom (DOF) as MAVs when they are performing tasks independent of elevation (e.g. navigating an unknown building [16]). There is less of a connection between MAVs and hovercrafts when the MAVs are performing aggressive maneuvers [17], [18] due to the large tilt angles and changes in elevation. In this paper, we analyze MAV and hovercraft similarities under situations where speeds are low and maneuvers are smooth. We now present an overview and comparison of the inertial dynamics for MAVs and hovercrafts.

A. MAV Dynamics

Translational movement in MAVs is performed through the use of pitch and roll. Tilting the MAV will introduce a horizontal thrust component that will allow the MAV to translate (Translate Left, Fig. 2). In addition, the overall thrust must be increased to maintain the MAVs current elevation. Because pitch and roll are about perpendicular axes, using them in conjunction allows translation in any direction. A simple control scheme is to have the MAV's baseline thrust level adjust to control elevation, and then perform small deviations on opposite rotors to control pitch and roll. The equations for MAV translational movement are as follows:

$$\begin{aligned} m_m a_h &= sin\theta \sum F_t - F_d \\ m_m a_z &= cos\theta \sum F_t - m_m g \end{aligned} \qquad (1)$$

where m_m is the mass of the MAV,
a_h is the horizontal acceleration,
θ is the angle of tilt,
F_t is the force of each rotor,
F_d is the counter force from aerodynamic drag,
and a_z is the vertical acceleration.

It is important to note that with the assumption of constant elevation, translation becomes a function of tilt. Once horizontal acceleration has begun, the only real limitations on speed are maintaining elevation with enough component of $\sum F_t$, and drag which increases non-linearly with speed.

A MAV rotates by using the drag the rotor blades experience opposite to their direction of rotation, which creates a torque on the vehicle. Since a quad-rotor MAV has two pairs of rotors turning clockwise and counterclockwise, an overall moment is generated by strategically increasing and decreasing the speed of the rotors, in turn altering the drag experienced by each pair. Since the drag of each rotor is

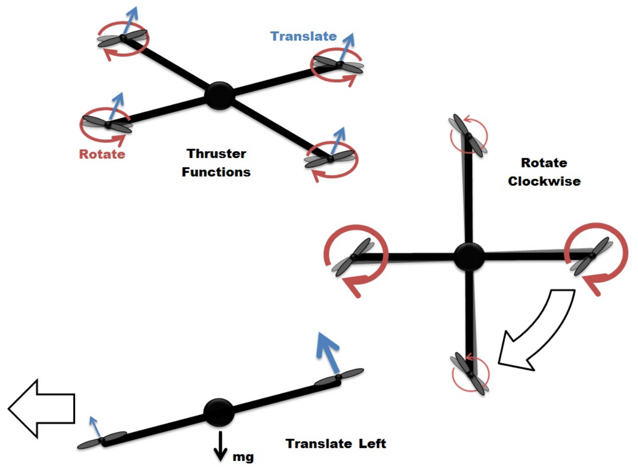

Fig. 2. Schematic of the MAV forces to achieve rotation and translation.

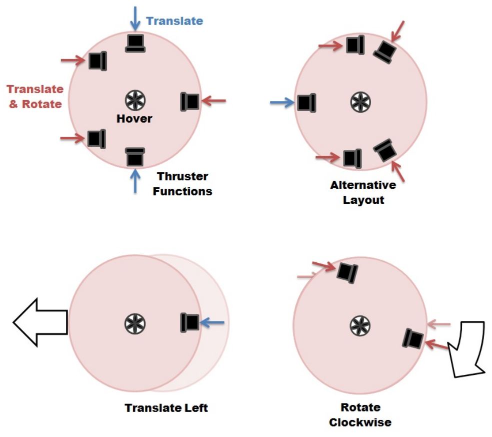

Fig. 3. Schematic of the hovercraft thruster layouts and forces to achieve rotation and translation.

proportional to the square of its rotational velocity [19], a controlled unbalance can be introduced to rotate the MAV. Fig. 2 shows that speeding up rotors with clockwise drag and slowing down those with counterclockwise drag causes the MAV to rotate clockwise. This does not induce additional pitch or roll since the pairs are opposite one another and the component of thrust remains balanced across the center of the MAV. The rotational dynamics are:

$$I_m\alpha = \sum M_t - M_d \tag{2}$$

where I_m is the rotational inertia of the MAV,
α is the angular acceleration,
M_t is the drag moment of each rotor,
and M_d is the counter moment from aerodynamic drag.

B. *Hovercraft Dynamics and Comparison*

Translational movement of hovercrafts along the ground plane is controlled by the one directional force output of each thruster. Ideally, coordinated thrusters pointing in at least three directions allow omni-directional acceleration and deceleration (Fig. 3) in the same fashion as tilt does for MAVs flying with small tilt angles (Eqn. 1). The equation for hovercraft translational dynamics is:

$$m_h a = \sum F_t - F_d \tag{3}$$

where m_h is the mass of the hovercraft,
a is the acceleration,
F_t is the force of each thruster,
and F_d is the counter force from skirt drag.

Hovercraft speed, just as with MAVs, is limited by drag and will reach a maximum when the force of drag is equal to the thruster force input to the system. The drag on the hovercraft is primarily due to limited contact between the inflated skirt and the ground. It is important to note that pure translation (i.e. translation without rotation) when using a single thruster can only occur if the thruster is aligned with the hovercraft's center of mass. If a thruster has a perpendicular offset from the hovercraft's center of mass with respect to the direction of its input force, it will introduce a moment about the center of mass (Fig. 3). Each moment is equal to the cross product of distance from the center of mass of the hovercraft to the thruster and thruster force. To perform rotation without translation, rotational thrusters must be countered with an equal magnitude of opposing thrust input, but by translational thrusters aligned with the center of mass. Just as rotation results from generated rotor moments on a MAV (Eqn. 2), rotation on the hovercraft is given by the following equation:

$$I_h\alpha = \sum F_t \times r - M_d \tag{4}$$

where I_h is the rotational inertia of the hovercraft,
α is the angular acceleration,
F_t is the force of each thruster,
r is the distance to each thruster from the center of mass,
and M_d is the counter moment from skirt drag.

Careful planning of thruster placement and control implementation allows a hovercraft to rotate and translate in any direction, in varying magnitude, in the same fashion as a MAV. Thrusters add momentum to the system, and this momentum is typically higher than the drag. This means that sufficient planning must be made in advance to coordinate a change in direction or to stop.

The payload capacity of the hovercraft is proportional to the thrust generated by the downward-facing thruster, the hovercraft diameter, efficiency, and the diameter of the downward-facing thruster:

$$P = \frac{F_t D_h \eta}{A_t} \tag{5}$$

where P is the payload of the hovercraft,
F_t is the force produced by the downward-facing thruster,
D_h is the diameter of the hovercraft,
η is the efficiency,
and A_t is the area of the downward-facing thruster.
Efficiency accounts for the air that escapes underneath the skirt of the hovercraft. It varies depending on the uniformity of the skirt and the smoothness of the ground surface. Efficiency can be determined experimentally for different configurations.

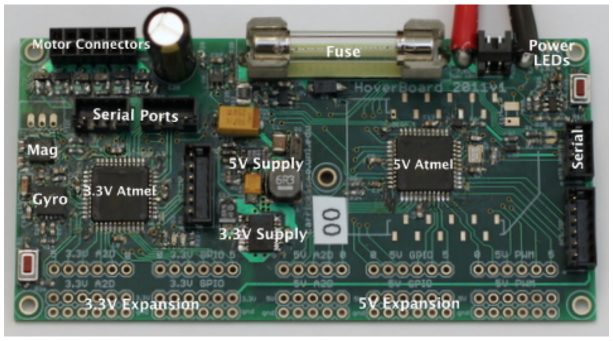

Fig. 4. Picture of the hoverboard that provides low-level control.

A final principal attribute shared between hovercrafts and MAVs is the importance of balanced and controlled corrections. Because control is based on the force input of placed thrusters or rotors, changes to either vehicle that unbalance the center of mass or rotational inertia can dramatically change the handling of the vehicle. If too many components are placed on one side of the hovercraft, for example, the shift in the center of mass will cause previously aligned translational thrusters to generate moments and rotations when not intended. Best results are obtained by mechanically balancing the system and then using feedback control to maintain constant heading to correct for unwanted rotation from unaligned thrusters or uneven drag. The similarities in dynamics between hovercrafts and MAVs show that omni-directional hovercrafts can be used to learn about the fundamentals of control and component placement that are directly applicable to MAVs.

IV. Similarities in Architecture

We designed the control architecture of the hovercraft similar to that of most MAVs. At a low-level there is a pair of microcontrollers that perform motor control, interface with sensors, and provide an abstracted interface for the higher-level system. The high-level control is implemented in ROS, with an interface that is compatible with existing ROS MAV control nodes. In this section, we detail both the low and high-level control systems and architectures.

A. Low-Level Architecture

The hoverboard was designed to easily interface with a wide variety of sensors and to provide base onboard sensors that are similar to those found on most MAVs. At the low-level the hovercraft is controlled by the hoverboard, shown in Fig. 4. The hoverboard has a pair of ATmega1284P processors, with one operating at 8MHz at 3.3V and the other at 20MHz at 5.0V. The two processors are connected over an I2C bus that is also exposed externally to enable easy expansion. Nearly all MAVs are controlled by multiple processors, so when designing the hoverboard, we decided to put two lower-end processors instead of a single more capable processor. This gives students in embedded systems-type courses the opportunity to learn how to implement protocols to communicate between the processors and they are also able to learn how to balance tasks between the processors to ensure real-time operation.

The hoverboard has a magnetometer and a one-axis gyroscope. These are used to control the rotation of the hovercraft and to teach students about data fusion on embedded systems. These are the same sensors found on most MAVs, minus the accelerometer, which is not needed since the hovercraft always stays level relative to the ground.

The hoverboard also has current feedback sensors, analog sensor inputs, digital I/O pins, serial, SPI ports, and PWM outputs. Since the hoverboard has both 5V and 3.3V inputs and outputs, it is possible to interface with nearly any type of sensor or actuator. We have added range finders, line detectors, bump sensors, accelerometers, servos, computer mice sensors, and a number of other devices. The ability to expand makes it easy to do a wide variety of course projects.

For wireless external control the 5V processor has a Zigbee radio. The 3.3V processor drives the hoverboard thrusters using a PWM signal to control MOSEFTs, which enables variable speed, uni-directional control of the thrusters (they can push, but not pull). The hovercraft is typically powered by a 7.4V two cell LiPo battery that is monitored by on-board circuitry to disable operation if the voltage drops too low. The system is protected from over current or short circuits with an 8A replaceable fuse.

The hovercraft was used as the core platform for an embedded systems course. In the course the students learned to configure and control most of the peripherals on the hoverboard and implemented data fusion algorithms, proportional-integral-derivative (PID) rotation controllers, communication protocols, and task schedulers. The design of the hoverboard is available online, but much of the functionality could also be replicated using a pair of Arduinos along with motor control shields.

B. High-Level Control in ROS

Robotic Operating System (ROS) is a meta-operating system framework that enables distributed control of a robotic system. In addition, ROS greatly simplifies the inherent complexity of engineering software for robotic systems. The architecture of ROS is based on a collection of computational units (referred to as nodes) that communicate with one another via a set of distinct publishers and subscribers (collectively referred to as topics). The nodes are highly cohesive and loosely coupled, making a system built with ROS evolvable, fault-tolerant, and scalable. Further, ROS nodes in the same system are capable of running on different computers (or robots), enabling off-line control and real time communication. Because ROS nodes can be written in either python or C++, object-oriented programming can be leveraged to increase modularity and encapsulation.

ROS is widely used by MAV researchers and there are open-source implementations for controlling both the MAVs used in our research (Parrot AR.Drone and Ascending Technologies Hummingbird). This, along with node modularity, makes ROS an ideal choice for our MAV-hovercraft, cross-compatible implementation. As shown in Fig. 5, we logically separate the ROS nodes in our system into four categories:

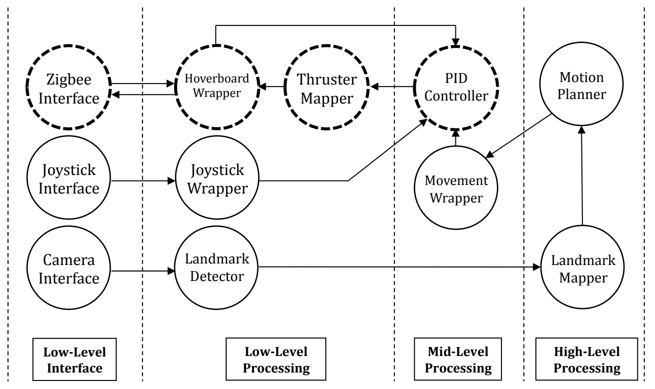

Fig. 5. ROS control architecture. Dashed nodes are easily replaced with MAV-specific nodes, enabling algorithm prototyping on hovercraft.

(1) low-level interface, (2) low-level processing, (3) mid-level processing, and (4) high-level processing.

The nodes in the low-level interface are responsible for low level communication and interfacing with devices such as radios and sensors. These nodes subscribe to (i.e. consume) topics that contain low level data that is directly received from or destined for sensors, thrusters, rotors, etc. They are responsible for converting low level data to and from serial message formats. In the event the data is outbound (from other ROS nodes) these nodes instruct the operating system to send them via the appropriate output device. Inbound data (from sensors or radios) is appropriately converted and published (i.e. emitted) to other ROS nodes.

The primary responsibility of low-level nodes is the abstraction to or from low level commands. These nodes subscribe to topics that are published from the low-level interface nodes, abstract the data in some way, and then publish the abstracted data to other ROS nodes. They also subscribe to topics that send abstracted data from higher level nodes that must be converted into low-level commands that are then published to the low-level interface nodes.

The mid-level processing nodes are central to the system. They often subscribe to and publish many topics. They are equipped to process both lower level data and higher level data (in terms of abstraction). In essence, they perform the required system functions that bridge the gap between the low-level nodes and high-level nodes.

High-level processing nodes generally interact with the most abstracted and processed data. In addition, they typically implement the most sophisticated and complex features of the system. Because many fine-grain and platform-specific details are abstracted away at this level, the developer is free to implement complex algorithms without having to manage large amounts of system implementation complexity.

ROS Example Implementation:

Fig. 5 presents an example system that uses a camera to identify landmarks (in this case, barcodes), map their locations, and then plan motions within the map based on the location of the landmarks. In addition, the system provides support for a manual override (i.e. the user can use a joystick to introduce movement commands at any time).

In this example, there are three low-level interface nodes: an interface for a Zigbee radio, joystick, and camera. These nodes enable communication with their respective devices and bridge the gap between the operating system and ROS.

Each low-level interface node publishes and subscribes to a low-level processing node[2]. For example, messages received from the joystick (via a low-level interface node) may have a high-precision data value for each button or joystick on the device in the form of a multi-dimensional array. However, other higher-level ROS nodes may require that this be abstracted into a simple integer-valued format. A low-level processing node would handle this abstraction.

At the mid-level, a PID controller is shown that is responsible for maintaining the desired heading and reducing undesired translational drift, as well as subscribing to and publishing movement commands when they are required (acting as a movement command proxy of sorts).

In our example, there are two high-level processing nodes. The landmark mapping node maintains a map of the known environment that the motion planner node can use to determine which movement commands are required. The motion planner node does not have to administer or control drift, platform-specific message formats, or deal with image data formats. All of this processing is handled in lower levels and only relevant, concise information is published to the high-level processing nodes.

ROS Modularity:

The majority of this system could also be used directly by a MAV with little modification (assuming the MAV is operating in a two dimensional plane). The four dashed nodes in Fig. 5 are the only nodes that would need to be modified if a MAV were used in lieu of our hovercraft.

The modifications that are required to adapt this system for a MAV are low-level and platform specific. For example, the Zigbee interface node must be modified to compute messages in a format the MAV can understand. Similarly, the thruster mapper node must be replaced by a rotor mapper node (assuming the MAV requires individual rotor commands). The PID controller node may only require that its control gain parameters be adjusted. In any case, replacing nodes or modifying node parameters is as simple as editing an XML file that is used to launch the ROS nodes.

In the lab setting, a researcher can develop and test a complex algorithm using our hovercraft implementation, and then run the same algorithm using a MAV. If modifications need to be made after switching to a MAV (or from a MAV back to the hovercraft), they can be made to the high-level processing nodes with little to no change required in the mid-level processing nodes.

For educational use, individual courses can focus on particular layers or touch on nodes at all levels. Classes focused on hardware, sensors, or other device interfaces may focus on developing low-level nodes. Courses on control-

[2]In our example there exists a one-to-one relationship between low-level interface nodes and low-level processing nodes, however this is not required and may not be the case in other examples.

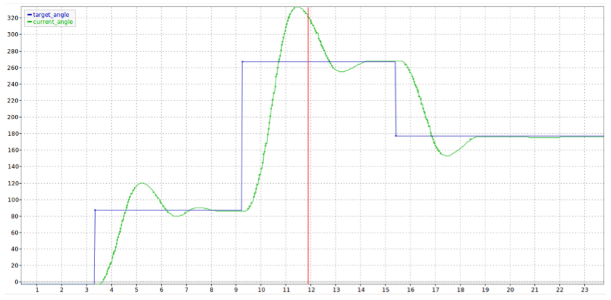

Fig. 6. PID controller step response to changing angles.

theory may focus on the mid-level control processing tasks, while courses on artificial intelligence may work on high-level nodes. A more general topics robotics course may address nodes at all levels.

V. Implementation and Experiences

In this section, we provide additional details on the hovercraft implementation. We also discuss our experiences with the hovercraft when teaching two courses: Embedded Systems (CSCE 436/836) and Robotics (CSCE 496/896). Both of these courses had upper-level undergraduates as well as graduate students. The Embedded Systems course had mostly Computer Science and Computer Engineering students, while the Robotics course also had students from Mechanical Engineering, Electrical Engineering, and Physics.

A. Hovercraft Implementation Details

The hovercrafts are constructed from inexpensive materials that are available at local and online hardware and hobby shops. The base is made of 1.5 inch rigid foam insulation and the skirt, which directs air to provide lift, is cut from a 5 mil plastic sheet. Thrust is provided by up to 6 standard RC ducted fans. Control and processing is enabled by a custom-designed circuit board detailed in Section IV-A. The total cost of the platform is $275 for small build quantities and drops to $225 for medium sized builds. This compares favorably to other educational robotics platforms as shown in Table I. The driving costs are the control board ($150 for small quantities), motors ($60 total), and battery ($40). The body components cost less than $25.

The low cost of the hovercraft's body components encourages exploration of different configurations and layouts. In the courses using this platform, students have built circular bases with diameters ranging from 12 to 18 inches and some groups experimented with different shapes for the base, although all groups ended up preferring the circular bases that had better lift characteristics in practice. There are multiple tradeoffs to consider when choosing the base size. Larger bases tend to allow for larger payloads and are less sensitive to weight imbalances. Having a large base and payload, however, decreases the rate at which the hovercraft can accelerate.

Compared to MAVs, the hovercrafts are very easy and quick to build and modify. In addition, the hovercrafts have significant payload capabilities that make it easy to mount new hardware and sensors. The hovercrafts can hover with a

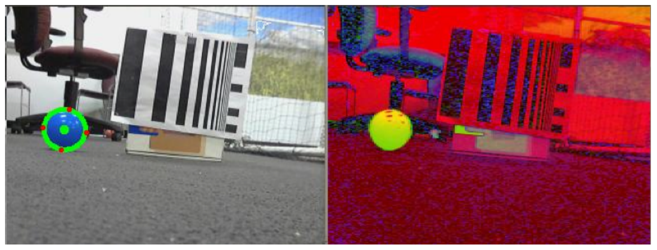

Fig. 7. Ball detected (left) based on the HSV image (right). Also seen is a scale-invariant landmark that can be identified for localization.

payload of over 2.5 kg (e.g. a netbook computer), but with this much additional mass it accelerates very slowly. In typical course projects, students have added range finders, line detectors, bump sensors, accelerometers, servos, computer mouse sensors, cameras, and a number of other devices. Fig. 8 shows a picture of the hovercraft configured with a ball collection mechanism and camera, which was part of the final project competition and is detailed in Section V-C.

B. Feedback Control

One of the early projects for both courses was the implementation of a PID controller to enable precise heading control of the hovercraft based on feedback from the gyroscope and magnetometer. This uses the base hardware on the hoverboard and can be implemented either on the microcontrollers or at a higher level in ROS. Actively controlling the angle is important even when trying to do pure translations as small misalignments in the translational thrusters will add torques that quickly add up to large rotational velocities if uncorrected. This is similar to quad-rotor MAVs, where control of each degree of freedom will impact the other degrees of freedom.

Fig. 6 shows the impulse response of a PD controller when changing the target angle. In practice no integral component was needed since the hovercraft has low friction and no external rotational forces. Note that there is a large initial overshoot, which is due to the rotational inertia associated with the hovercraft and is difficult to avoid with a PD controller without significantly slowing the response rate. In practice, we found that this level of overshoot is acceptable for most applications which require smooth changes in angle and not stepwise changes. The overshoot can be reduced by adding a feed-forward component to the controller.

C. Ball Detection, Following, and Capture

The final project for the Robotics (CSCE 496/896) course involved collecting as many balls at known locations as possible in an environment augmented with scale-invariant visual landmarks [20]. This combined techniques learned in prior assignments including visual localization and navigation, ball detecting (see Fig. 7), and visual servoing. In addition, this project required augmenting the hovercraft with a gripper or ball collection mechanism, see Fig. 8.

These tasks required vision processing, which was initially performed using a small Gumstix processor onboard the hovercraft. Unfortunately, the operational area had poor WiFi bandwidth so ultimately students used wired webcams. Since

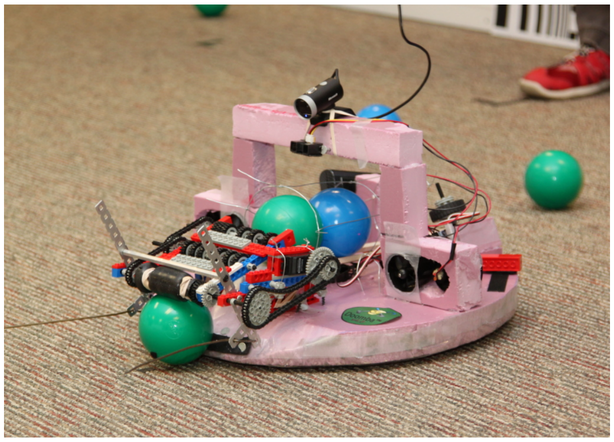

Fig. 8. Hovercraft augmented with a ball-collection mechanism.

landmarks were spread out, most groups only used the visual landmarks to opportunistically localize the hovercraft whenever a landmark was visible. This is because dead reckoning with the hovercraft (and also MAVs) is extremely imprecise and cannot be relied upon for significant distances.

Detecting balls was performed by thresholding in the HSV color-space and then looking for round objects as shown in Fig. 7. While the balls were static in the competition, the hovercrafts still needed to dynamically track and adjust their trajectories when approaching a ball.

Each group developed a different mechanism for ball collection. Most used more traditional graspers to pick up a single ball and then drop it off at a target location. One group instead developed a ball collection mechanism that collected many balls at once as seen in Fig. 8. This worked well since less time was spent traveling to dropoff locations.

VI. Conclusions and Future Work

In this article, we presented an overview of an omni-directional hovercraft platform and compared its dynamics to quad-rotor MAVs and more traditional ground robots. We also examined the similarities in the low and high-level control design of the hovercraft and MAVs that makes interoperability easy. Finally, we looked at some of course assignments that used the hovercraft.

The hovercraft worked well in these classes to teach students about the challenges associated with developing and controlling robots with significant momentum that cannot easily or precisely stop or change direction. This forms a foundation that prepares students to develop and control MAVs in future courses and careers. In addition, we have found that the hovercraft is a useful platform for developing and testing algorithms in our research lab before implementing them on an actual MAV.

While the hovercraft has similar dynamics to a quad-rotor MAV in that it is an inertia-driven platform, there are some differences. Mainly, MAVs pitch and roll to translate, whereas the hovercraft does not. For some sensing (e.g. taking pictures) and control tasks the pitch and roll can have a significant impact. As such, the hovercraft is best compared to MAVs that are moving slowly and not performing aggressive maneuvers.

In the future, we plan to work on reducing the cost of the electronics to reach a target cost of under $200. At this price point the hovercraft will be accessible to a larger set of users. We also aim to develop followup courses that use MAVs by building on the experience and knowledge the students gained while working with the hovercraft.

Acknowledgment

Thanks to Brian Julian for initial ideas and discussions about hovercraft, Tim Echtenkamp and Tyler Lemburg for developing initial control and interfaces, NIMBUS Lab members, UNL CSE Department, and especiall the many students in CSCE436/836 and CSCE496/896.

References

[1] "Ascending technologies." [Online]. Available: http://www.asctec.de
[2] "Parrot AR.Drone." [Online]. Available: http://ardrone.parrot.com/
[3] "Robot Operating System." [Online]. Available: http://www.ros.org/
[4] J. McLurkin, S. Rixner, M. O'Malley, A. Lynch, and T. Barr, "A low-cost multi-robot system for research, teaching, and outreach," in *Distributed Autonomous Robotic Systems*, 2010.
[5] B. Tribelhorn and Z. Dodds, "Evaluating the roomba: A low-cost, ubiquitous platform for robotics research and education," in *Robotics and Automation*, 2007, pp. 1393–1399.
[6] B. Dickinson, O. Jenkins, M. Moseley, D. Bloom, and D. Hartmann, "Roomba pac-man: Teaching autonomous robotics through embodied gaming," in *AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education*, 2007, pp. 35–39.
[7] D. Housten and W. Regli, "Low-Cost localization for educational robotic platforms via an external Fixed-Position camera," in *AAAI AI Education Colloquium*, 2008.
[8] K. Dantu, B. Kate, J. Waterman, P. Bailis, and M. Welsh, "Programming micro-aerial vehicle swarms with karma," in *Conference on Embedded Networked Sensor Systems*, 2011.
[9] J. Shepherd III and K. Tumer, "Robust neuro-control for a micro quadrotor," in *Genetic and evolutionary computation*, 2010, pp. 1131–1138.
[10] P. Sujit, A. Sinha, and D. Ghose, "Multiple UAV task allocation using negotiation," in *autonomous agents and multiagent systems*, 2006, pp. 471–478.
[11] H. Asama, M. Sato, L. Bogoni, H. Kaetsu, A. Mitsumoto, and I. Endo, "Development of an omni-directional mobile robot with 3 DOF decoupling drive mechanism," in *Robotics and Automation*, vol. 2, 1995, pp. 1925–1930.
[12] O. Diegel, A. Badve, G. Bright, J. Potgieter, and S. Tlale, "Improved mecanum wheel design for omni-directional robots," in *Australasian Conference on Robotics and Automation*, 2002, pp. 117–121.
[13] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.
[14] P. Muir and C. Neuman, "Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot," in *International Conference on Robotics and Automation*, vol. 4, mar 1987, pp. 1772 – 1778.
[15] I. Khan and M. Spenko, "Dynamics and control of an omnidirectional unmanned ground vehicle," in *Intelligent Robots and Systems (IROS)*, oct. 2009, pp. 4110 –4115.
[16] A. Bachrach, R. He, and N. Roy, "Autonomous flight in unstructured and unknown indoor environments," in *The European Micro Aerial Vehicle Conference and Flight Competition 2009*, 2009.
[17] H. Huang, G. Hoffmann, S. Waslander, and C. Tomlin, "Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering," in *International Conference on Robotics and Automation (ICRA)*, may 2009, pp. 3277 –3282.
[18] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *International Symposium on Experimental Robotics (ISER)*, Dec 2010.
[19] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *Robotics Automation Magazine, IEEE*, vol. 17, no. 3, pp. 56 –65, sept. 2010.
[20] D. Scharstein and A. Briggs, "Real-time recognition of self-similar landmarks," *Image and Vision Computing*, vol. 19, no. 11, pp. 763–772, sept. 2001.