Project Hawking:

Engineering an Autonomous System

Tennessee Technological University

Autonomous Robotics Club [1]

IGVC 2013

[1]I, Stephen Canfield, sign that this project consisted in good engineering design effort:

Contents

1. Introduction

The Autonomous Robotics Club (ARC) of Tennessee Technological University has, with the support of Apple Mobility, had the opportunity to create an autonomous wheelchair. Wheelchairs come in all shapes and sizes, but they all have one purpose: to assist people who are unable to get around on their own feet. One major use of wheelchairs is the transport of patients in hospitals. Today, a nurse must first find a wheelchair, bring it back to the patient, and once they are done, take it back to a storage area. However, because of how busy hospitals can be, finding a free wheelchair can be difficult. We propose that if the wheelchairs could drive themselves to the nurse autonomously, the nurse's time could be more focused on taking care of the patient instead of looking for a wheelchair. Currently we are only using the base of Invacare's Pronto R2 wheelchair, but as we move forward we wish to begin making the computer and sensors less intrusive and visible so the patient may sit comfortably as a nurse escorts them to their destination.

2. Design Process and Team Organization

The ARC is an interdisciplinary club of students open to any and all students at Tennessee Tech. Since the ARC exists purely as an extracurricular and volunteer-based club at Tennessee Tech, we organized our group dynamics based on interest and skill levels. We meet twice during the week on Wednesdays and Thursdays in order to help new members learn about robotics. Typically the newer and less experienced members work on smaller BOE-Bot kits which allow them to learn basic circuits and programing skills along with building problem-solving skills. As members gain more experience and want to expand their knowledge they join us with Project Hawking. Members involved in Project Hawking also meet on Saturday; this day serves as our major work day during which we build and run tests for Hawking. Having this structure for club meetings allows us to prevent the intimidation of newer members and allows for bright ideas from anyone in regards to Project Hawking.

2.1 Design Process

The ARC has existed for several years and has competed in IGVC in years past with Andros, a bomb disposal robot. However, the club has lost many of its more experienced members; therefore, our club has a completely new generation of members and is, in a way, starting from square one. With that said, our design process is chopped into smaller segments. We take each problem in small steps which allows us to approach a fairly complex problem in a much simpler manner. This also makes testing easier and allows us to see our progress, which, in turn, helps with motivation and maintaining members' interest in the club.

2.2 Team Composition and Organization

Table 1: Project Hawking Team Members

Name	Major	Level	Hours
Justin Johnson	Electrical Engineering	Undergraduate	20
Savannah Nolen	Electrical Engineering	Undergraduate	20
Scott Hill	Mechanical Engineering	Undergraduate	60
Edward Tidwell	Mechanical Engineering	Undergraduate	50
Phillip Adams	Mechanical Engineering	Undergraduate	50
Daniel Oliver	Computer Science	Undergraduate	60
Cameron Chaparro	Computer Science	Undergraduate	40
Alex von Dollen	Computer Engineering	Undergraduate	80
Brian Peppers	Computer Science	Undergraduate	30
Leon Allen	Computer Science	Undergraduate	20

Members involved in Project Hawking can be found in Table 1. Generally, members of the Project Hawking Team fell within four major groups:

- Hardware and Electrical – Responsible for mounting sensors to Hawking's chassis, repairing failed components, and power distribution.
- Sensor Interfacing – Develop drivers for the various sensors used on the robot.
- Algorithms – Write navigation logic for avoiding obstacles, staying between lines, and reaching waypoints.

- System Integration – Responsible for the "glue" which allowed each of the individual groups to work together.

2.3 Resource Sharing and Communication

To help in making the club sustainable, we utilize an online wiki forum, subversion content management system, and public website for sharing information. This way, members who miss important meetings or discussion can remain informed as his/her schedule allowed. The forums also provide a key factor for the club's future success: the documentation of progress, success, and failures. The subversion content management system is a great way for members to review the latest revisions to code and quickly identify changes. The public webpage allows anyone to view what we have done with Project Hawking as well as upcoming events we are sponsoring or participating in, which helps in promoting the club and with recruitment.

3. Mechanical Systems

We are using an Invacare Pronto R2 electric wheel chair from Apple Mobility as the base from which our robot is built. Due to their simplicity and durability, a wheel chair chassis was chosen after the motor failures of the ARC's last two robots. Apple Mobility was happy to help when they found out that we were in need.

The Invacare Pronto R2 is an electric wheel chair, which makes it very nimble. It has an electric motor for each drive wheel like most other powered wheelchairs; because each motor drives a single wheel, the robot can provide plenty of power without fear of damaging the motor on an incline or in rough terrain.

The seat, arms, and powered reclining system have all been removed to leave ample room for the computer box. Everything added to the chassis follows the ARC's "bolt-on" philosophy. The wheels, motors, and drivers are all mounted onto the chassis and have not been altered from the factory specifications. All feedback comes from the sensor array: GPS, camera, laser range finder, etc.

3.1 Computer Box

In order to protect our precious electronic systems, the ARC has manufactured a large housing (Figure 1) to secure our computer and various other electronics. It is mounted where the seat of the Invacare Pronto R2 was removed. The hinged lid of the housing allows for a large, open space in which to work on the electrical systems. The extruding aluminum tower at the front of the box is an ideal place to attach sensors, such as the camera and laser range finder. As long as the chassis can bear its size, the box can easily be moved to a different platform.

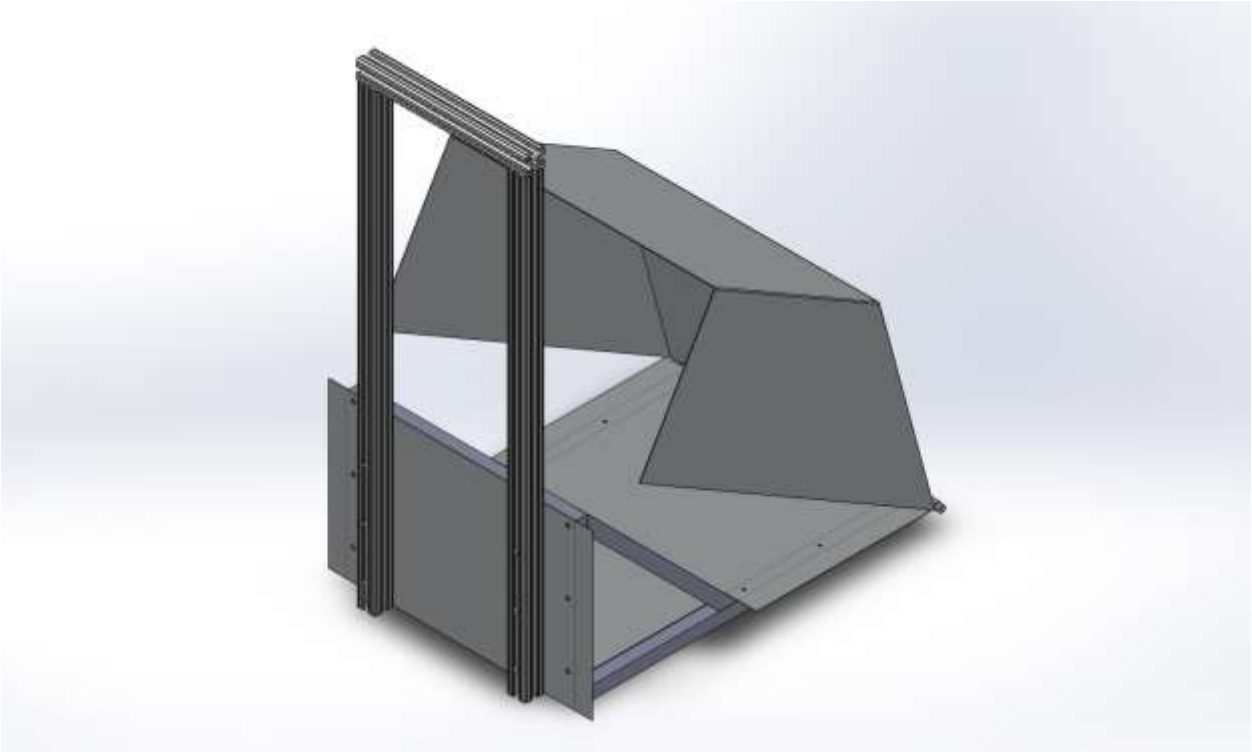


Figure 1: A 3D rendering of the computer box.

Figure 2: Logitech QuickCam Orbit AF

Figure 3: Ag Leader GPS

4.1 LMS Laser Measurement System

Hawking will be using the LMS291-S05 sensor. The LMS sensor is used for physical obstacle detection. The LMS291-S05 is a scanning laser range finder with a sensing range of 80 meters. Measurement accuracy is within 1.83 meters tolerance of the current reading for most of the sensor's range. Our LMS has a field of vision of 180 degrees.

This past year the club has been using the Hokuyo URG-04LX-UG01 Laser Range Finder, however, we have been having problems with sunlight interfering with the Hokuyo and causing the diode to overheat. They temporarily fixed this by making a metal housing unit, but we have discovered that the Hokuyo can only be used reliably in indoor low-lit conditions. The laser inside the unit is not intense enough for it to work properly outside or even in brightly lit environments. Thus the club has decided to utilize the unit previously used on one of our past robots, the LMS291-S05, in hopes of better performance in outdoor conditions.

4.2 Cameras

Hawking currently utilizes one main camera (Figure 2) that is mounted approximately 5 feet from the ground; this setup allows us to see both wheels of the robot. We acquired the QuickCam Orbit AF by Logitech in hopes of utilizing the X, Y rotational axes: this feature would allow the robot to pan the camera, which would in turn allow it to look up a steep incline or

look for something outside of its range of view when it cannot physically move. With a moderately powerful computer we can use the camera for real time object detection, line and edge detection, and color sensing. Utilizing the camera along with the laser range finder, the robot can get a pretty good idea what is in its way.

4.3 Global Positioning System

Hawking's GPS sensor (Figure 3) is an Ag Leader GPS 1500 4. This GPS is a smart antenna that tracks GPS and SBAS (WAAS and EGNOS) signals allowing for sub-meter accuracy. The GPS automatically finds and connects these signals, outputting them as a standard NMEA string at 1Hz through a serial connection. Since this GPS is designed for agricultural services, it is easily mountable and nearly impervious to any operating conditions.

5. Electronics and Computer Systems

5.1 Emergency Stop

Considering the requirement for IGVC 2013, our Emergency Stop system has been reworked and the system guarantees the motion of the robot comes to a complete halt when activated. We accomplish this by means of a robust DP/DT (double pole/double throw) relay with high current capacity contacts and an emergency stop toggle button. The E-stop switch is located on the main control panel, centered at the rear of the robot, providing easy access and visibility. The relay and wireless E-stop components are located inside a project enclosure and are mounted to the base of the enclosure to prevent any damage from jostling during operation.

The Wireless E-stop is connected in series with the E-stop system, and it includes a wireless receiver connected to the relay. By means of a key fob transmitter, the Wireless E-stop can be activated, opening up the circuit that feeds power to Hawking's motors, making him safe in the event of an emergency shutdown scenario.

5.2 Main Computer

In the past the club has used the Intel D945GCLF2 Mini-ITX Motherboard, however, we have come to a limitation with image processing. This is probably because we are not currently taking advantage of parallel processing in our code. To continue with our development we have decided to switch to a club member's old laptop. We do this primarily for convince since it has its own onboard battery and screen. This makes for easy set-ups, along with fast debugging and "in-run" code changes.

Therefore, the computer that is currently in use is the Dell Inspiron 6400. With the Intel Core Duo T2400 with 5 Mb of cache clocked at 1.83 GHz we get a noticeable improvement from the Intel D945GCLF2 Atom's 1.6 GHz processor. With the slight performance difference and the major conveyance of testing, we will probably stick with the laptop form factor for our robots' computer system in the future.

5.3 Power and Battery Systems

All of the internal and external systems run from one pure sine wave inverter connected to a 12-volt battery from the wheelchair's 24-volt battery supply. Each system's 120-volt power supplies are connected to the inverter via a surge protector for power distribution. This enables easy replacement of system components without being restricted to a specific battery type with a specific voltage.

Because the pure sine wave inverter is located near our central processing unit inside a metal enclosure, it is necessary to protect the processing unit and other components from the electro-magnetic fields being created by the inverter. This is done by enclosing the inverter inside of what has been coined the "Faraday Cage." The Faraday Cage is a prism constructed from brass mesh that prevents electro-magnetic fields from escaping the enclosure.

6. Software Strategies

6.1 General Software Philosophy

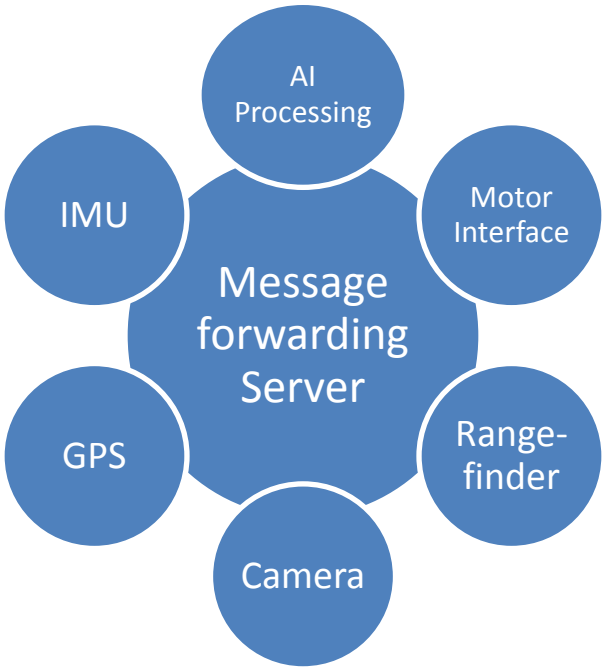
Since our club operates as a revolving door for students interested in robotics at TTU, we realize that our work must be simple, robust, and useful enough to be used and assimilated by future students wishing to gain a foothold in the area of autonomous robotics. Due to this

fact, documentation and good coding practices are crucial. Additionally, since software architecture choices strongly impact the modularity of our systems, we found it wise to use a heavily modular and object-oriented approach to our software design.

To achieve this modular approach we have utilized socket programing. This allows us to treat each sensor as separate clients, along with the main AI. Furthermore, we can write separate clients to override the AI and control the robot through the sad client for debugging purposes. This approach also allows us to split the load through multiple computers, if we wish, by setting up a local network between them. The computers can share the overall programs, for example, by having the vision processing on one, and the sensors, AI, and the server running on the other.

The final goal is to have an interface simple enough that a curious freshman could pick up and, in single day, accomplish simple demonstrations of autonomy and sensor aggregation. Doing this serves a dual purpose as well: if the system is simple enough to be used by novices then it certainly must be mature enough as a robust and simple strap-on solution for autonomy.

7. Systems Integration



7.1 Vision

From our past experiences, vision is the trickiest aspect to implement properly. Lighting, glare, and motion blur seek to destroy any hope of gleaning useful information the robot can use to guide itself. Of course it is paramount that the robot stays within the lines; therefore, a given line detection scheme should be as close to 99% sensitivity as possible. But typically, maximizing sensitivity comes at the cost of specificity. That is, we can trivially identify all white lines by simply writing a predictor that disregards its input and always "sees" all possible white lines. The true positive rate and the false positive rate are at odds; we don't want a robot running away from phantom white lines, but we certainly can't have a robot cross a real white line. Furthermore, given that the state-of-the-art vision algorithms for robustly identifying even simple objects in arbitrary orientations, scale, and positions are still lacking, simple white line detection on an IGVC course becomes a significant problem. Often in IGVC, the robot can barely see in front of itself: a switchback, for instance, almost completely obscures the robot's perceptions of what comes after it.

With these challenges comes several important restrictions to any algorithm: it must be tolerant to lighting changes (overcast, sunny, raining, etc.), motion blur (uneven terrain), glare (reflectors, shiny colored paint), and should not depend on seeing far away (switchbacks, object occlusion).

Loosely speaking, the robot should look close to itself for oblong moderate-sized connected regions of bright, near-white color. Thus, our proposed algorithm must find the white colors in the image (white filter), filter out glare from the tips of grass and white from pieces of hay (blob analysis), and finally detect "line-like" pieces (Hough transform).

7.1.1 White Filter

Creating a white filter is quiet challenging due to the fact that "white" is not always white. This unfortunate fact comes from the internal mechanics of cameras themselves which also don't know true white and must guess this color using a technique called white balancing. Improper white balancing may result in a "blue haze" over the entire image, for instance, when

photographs of the beach, ocean, and sky are taken; thus, a white filter must account for the possibility that a “haze” might be present in an image. Our technique, which we find quite robust, is one of color segmentation using 3-dimensional k-means clustering.

K-means clustering is a simple algorithm which seeks, creates random cluster of points, calculates the distances from the centroid of a cluster to all its constituent points, and then tries to minimize this distance iteratively. In our case, we treat each pixel as a point in 3-dimensional Cartesian space, with (x,y,z) equal to (R,G,B). We can think of the entire 8-bit RGB color space as a 3-dimensional cube where black (0,0,0) is in one corner and white (255,255,255) is in the opposite corner. Like colors will be “close” via Euclidean distance, so we can appropriately apply k-means clustering as a technique to find dominant colors in an image. We chose RGB space instead of HSV (Hue Saturation Value), because finding the “whites” cluster in RGB space is as simple as finding the largest R+G+B sum. We also found that, in practice, k = 5 suffices to capture white (and white only) as the lightest cluster. The lightest cluster is the cluster whose centroid is furthest in the “white corner” of the RGB cube.

The benefit of color clustering over traditional threshold-based methods is clear to see: thresholds such as (R > 200 AND B > 200), for instance, are very dependent on the current lighting conditions. This threshold (200) could change significantly if the white line is in the shadow of a barrel or the amount of light form the sun. With color segmentation, we only need to guarantee that the lines are of uniform color and that they are very light compared to the other scenery.

7.1.2 Blob Analysis

The cluster-based white filter does a great job at capturing white under varying light conditions, but it doesn’t provide a way to filter out other types of white matter that aren’t part of lines. Hay, glare, and white barrels contribute to the white cluster as well and will be captured by the white filter; thus, we need a way to distinguish from the tips of grass catching the sun and white paint on grass. We can do this by contouring the binary image output from the white filter and removing any contours that produce an inner area less than a small amount. In effect, this action will remove small “blobs” of white, while keeping the large blobs;

these blobs are fully connected regions. A line segment will thus look like a large rectangle, while glare will look like small patches. By using contours we end up with edges, similar to a Canny transform. This effect will help us for our final step, the Hough transform.

7.1.3 Hough Transform

We employ the Hough transform to look for line segments. We thus can single out the oblong blobs because their long contours provide us with two lines each of identical slope. As Hough is normally used in conjunction with Canny, the contouring of our image serves as a suitable replacement for this first step. Finally, to characterize our findings, we simply find the average angle of line by treating out Hough lines as vectors and using an optimized arctangent estimate. The final output is then either: NULL or <angle>, depending on whether or not lines were found and what the average angle is respectively.

7.2 GPS-Reader

The GPS software polls the GPS unit at 3Hz. Each parcel of information is broadcasted to the artificial intelligence along with a timestamp. All of the GPS coordinates are recorded and Kalman filters are applied to normalize the data and remove spurious values. The degree of accuracy rapidly decreases as the time since the last update increases. If the signal is dropped then that information is broadcasted as well.

7.3 Navigation

Topic sentence needed here. And define Seek. Seek is the only operational state; it is, in general, a very simple algorithm. Given a set of waypoints (GPS coordinates), Hawking navigates to each of them and stops at the last one. All sensor readings are stored in a giant grid with a variable resolution. Any data stored in this giant grid is “fuzzy,” that is each tile contains the probability that there is something in that cell. As such, this allows for the possibility of sensor uncertainty and positioning uncertainty.

The algorithm will perform a simple A-star search to the next waypoint, choosing the

path with the least likelihood of obstacle collision. The algorithm will not backtrack unless there is no other choice. As new sensor data is read in, the map grid is updated and a new path is calculated as necessary. Once given a calculated path, it is a simple matter to follow it.

Since the search algorithm needs accurate position information to accurately follow a path, the GPS unit is extremely important. But since the GPS is not accurate to the needed degree, SLAM (Simultaneous-Localization-And-Mapping) is utilized as well. Since SLAM is inherently more accurate indoors, it provides just enough information combined with GPS to give sufficient results for mapping and path following.

8. Cost Table

Table 2 gives a summary of major components used in the production of Hawking, along with their estimated cost based on prices from a variety of online sources, and their actual cost to the ARC. It also includes an estimate for minor components used, labeled “Miscellaneous Components.”

Table 2: Major components used in Project Hawking

Item	Estimated Cost	Cost to ARC
SICK Rangefinder	$10,000	$0
AG Leader APS 1500	$1,000	$1,000
Dell Inspiron 6400	$150	$0
NETGEAR Wireless-G Router	$55	$55
Invacare Pronto R2	$3500	$0
Computer Electronics Box	$200	$200
Sunforce Pro Series Inverter	$400	$200
Logitech QuickCam Orbit AF HD Web Camera	$150	$150
Miscellaneous Components	$200	$200
TOTAL	$15655	$1805

9. Predicted Performance

The software will allow up to about 5mph on straightaways with no barriers in sight. The average will be about 3mph to allow for proper white line detection. The motor battery life is many hours but has never been field-tested to its duration. The limiting time factor is the onboard laptop's battery life which is approximately an hour of all systems running. The duration can be increased if the workload is split between multiple onboard laptops. Reaction speed is equal to the update rate of the sensor giving the information to react to. The navigation algorithm reaction time is so fast that it is negligible compared to the sensor processing.

10. Conclusion

We, the Autonomous Robotics Club of Tennessee Tech University are very proud of our accomplishments this year. We have designed a working prototype of an autonomous wheelchair, moving away from what our past members have accomplished and paving the way for a new generation within the ARC. Our club is beginning to grow due to some other, beginner appropriate projects we have started. With all this in mind we cannot wait to see what the future holds for Tennessee Tech's Autonomous Robotics Club.