CSE 424 Final Report

Parallelize or Paralyze

Team Seek and Destroy

Group Members: Edward Andert, Eric Barber, Michael Vetrano, Roger Dolan, Shang Wang, Thomas Barry

Sponsor: Aviral Shrivastava

4/1/2013
# Table of Contents

Table of Figures .......................... 2
Table of Tables .......................... 2

**INTRODUCTION** ......................... 3

Project Description ....................... 3
Purpose .................................. 3
Users .................................. 4

**SCOPE** ................................ 5

**SCHEDULE** .............................. 6

Turret Plan ................................ 9
Car Plan .................................. 10

**DEVELOPMENT APPROACH** ............. 12

Turret Development Approach .......... 12
Car Development Approach .............. 12

**DESIGN DECISION** ...................... 13

Turret Design Decisions ................. 14
Car Design Decisions ..................... 15

**TOOLS AND TECHNOLOGY** .......... 19

Tower Tools and Technology .......... 19
  Hardware .......................... 19
  Software .......................... 20

Car Tools and Technology ............... 20
  Hardware .......................... 21
  Software .......................... 21

**RESULTS** .............................. 22
Introduction

Project Description

Team Seek and Destroy’s project consists of two autonomous robots that are independently aware of their surroundings. The car has a main goal of following a human skeletal figure around. The human is detected using a Kinect sensor at the front of the vehicle. At no time should the car stop following the human, and must dodge obstacles in the way. Voice commands may be used to stop the car at any time, and start the car again after stopping. The car has a small atom based computer, an Arduino microcontroller, and a Kinect sensor onboard to take care of all the vision and processing requirements. The other autonomous robot is called the tower. The tower’s goal is to shoot a foam dart at the car, which has an orange ball mounted on top of it. A Kinect sensor is mounted to a wooden board underneath the sensor. The orange ball is picked up by the Kinect, processed on the Ubuntu Linux laptop, and the servos on the turret adjust accordingly. The servos are adjusted taking into account the speed, distance, and trajectory of the car using an Arduino. The Arduino Nano unit utilizes two of the PWM pins to control the servos, based on a serial communication stream from the laptop from the processed Kinect sensor images. When placed together, the turret will target the car while the car targets the human. In this way, the human can aid the car in avoiding the missiles from hitting it. This is an entertaining example of how cyber physical systems can work together to form an “intelligent” demonstration.

Purpose

There are several motivations for this project. The first and foremost purpose was set by our sponsor. We were to attend the Engineering Open House hosted by ASU. In this event, school children and parents came and looked at many science exhibits, including the exhibit put on by Team Seek and Destroy. The people who came and viewed our exhibit were having a lot of fun walking around and having the car follow them, or holding the orange ball and having the turret shoot it. It was interesting how many different kinds of people interacted with the technology, and the range of questions was intriguing.
There are many people and firms researching into robot autonomy. This project provides another field of research as instead of the common model of a robot trying to interact with the human world, or a robot that can mimic human interaction, our team is creating a pair of robots whose sole purpose is to interact with each other. This brings about a spirit of healthy competition between teammates that encourages innovation whilst delving into an aspect of the robotic field that is not necessarily very well populated with researchers. The project is also developing new usage models for the Kinect sensor, such as mounting it on a mobile robot and navigating around objects. The robotic car is much the same setup as a full size car that one would see on the road, so this could also be taken as development in the field of autonomous transportation too.

The third possible use is as future military hardware prototype or model. The tower is a robot that, with a lot of modification, could be very easily outfitted with a real weapon and used in combat. This means it is a useful prototype for future military technology. The car could arguably be even more useful. The civilian sector isn’t the only sector looking into semi and fully-autonomous remote controlled robots. The car is a good prototype for a relatively inexpensive autonomous scout that doesn’t place soldiers in harm’s way. As can be seen here, this project has many motivations and potential uses in the real world.

A final approach could be that of a children’s toy. While at the open house, we realized that kids loved to play with both the turret and the car. The turret may be too dangerous to produce as a toy, but the car could not only be fun, but useful. Children would have an autonomous friend following them very similarly to a dog. It would be able to carry small things, interact when needed to, and just follow and exist when no direct interaction is needed. With a low enough final price and dedicated components, this item could be successful in the market.

**Users**

The users of this project right now are primarily those in a university setting. Though the project wouldn’t be very hard to produce copies of, it is most likely that it will remain as a single project.
or as consecutive new projects built off of the design. Therefore, future capstone groups will probably comprise most of the users of this project. The sponsor and those working with him will potentially also use the project as a demonstration at ASU events to pique students’ interests in the subject. The algorithms and robots are novel in the fact that this exact combination has not been thought of before, but the users of the project in its current state will not stretch far from academia.

Scope
The scope of the project has changed dramatically since the initial inception of the project. The initial scope was defined as a remote control car that could drive itself based on data points collected from a Kinect sensor and translated through an Arduino board. The Kinect sensor and on board computer were not only going to process object data, but also practice missile avoidance from the turret which was going to be shooting at it. The turret was to be a highly precise autonomous missile launcher which would mercilessly track the car. There was no scenario except for a general on floor demonstration that was thought of at this point. Upon diving further into the project, the scope has changed a bit to harbor realistic and reachable goals, and a final scenario has been thought of as a milestone to work towards.

Revising the scope was a crucial part of the entire experience, and truly made the project our own, as opposed to a hand me down from a previous group. Instead of practicing missile avoidance with the Kinect on the car, we decided to follow a human. This was done to make subsequent experiences truly involved with the person using the device, and gave it an “alive” feeling which was unique. This worked in our favor as active missile avoidance using the on board atom based computer would be next to impossible utilizing the Kinect libraries. The car still does practice avoidance, but it is from obstacles that could keep it from following the human, not from incoming missiles. The car also has voice recognition built in now, which is handy in stopping and starting the car without the use of an external computer.

The tower was revised in a minor fashion compared to the car. The turret is still targeting a round orange ball and calculating its trajectory based on this circle. There are better
algorithms in place which take into account the speed of the dart, gravity, and distance to the
target for a more accurate shot. This was not possible before because the computer tower that
was being used, a Pentium four single core, could not handle the video and physics calculations
at the same time. Moving the turret onto an i5 based laptop alleviated this issue completely
and progress could be made. The initial turret was a roughly constructed and glued together
device. The new scope called for stability, so an entirely new platform and pan and tilt system
were added. The new setup is extremely stable and allows for greater accuracy.

The final demonstration was changed from an obstacle course to a walkway. The car has
the goal of following the human, who was trying to make the car move fast and erratic enough
for the turret not to hit the car with a dart. This is a very interactive example, and worked
perfectly for the main goal of showcasing at the engineering open house. Changing the scope
for this demonstration proved to be an effective way to keep the group on task, and keep end
goals in mind. Without this demonstration, it would have been much harder to coordinate our
thought plans and make one system that could work together as it does.

### Schedule

| Original vs. Actual - Turret Plan - Semester 1 |
|-----------------|---------|---------|---------|---------|
| Week            | 3       | 4       | 1       | 2       |
| Understand Legacy Code | 3       | 4       | 1       | 2       |
| Understand Legacy Code (Actual) | 1       | 2       | 3       | 4       |
| Reorganize, Update Code |         |         | 1       | 2       | 3   |
|-------------------------------|------|------|------|------|
| Reorganize, Update Code (Actual) | ✔️   | ✔️   |      |      |
| Document Old Code             |      |      | ✔️   |      |
| Document Old Code (Actual)    |      |      | ✔️   |      |
| Run On Other Hardware         | ✔️   | ✔️   |      |      |
| Run On Other Hardware (Actual)|      |      | ✔️   |      |
| Optimizations                 |      | ✔️   |      |      |
| Optimizations (Actual)        |      | ✔️   |      |      |
| Test Demo 1                   | ✔️   | ✔️   |      |      |
| Test Demo 1 (Actual)          |      | ✔️   |      |      |

Table 1: Semester 1 Schedule Tower
| Research Software Improvements |   |   |   |   |   |   |   |   |   |
| Research Software Improvements (Actual) |   |   |   |   |   |   |   |   |   |
| Implement Firmware Improvements |   |   |   |   |   |   |   |   |   |
| Implement Firmware Improvements (Actual) |   |   |   |   |   |   |   |   |   |

**Table 2: Semester 1 Schedule Car**

### Original vs. Actual – Car and Tower Plan - Semester 2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Week</td>
<td>3 4 1 2 3 4</td>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
<td></td>
</tr>
<tr>
<td>Stage 2 Optimizations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stage 2 Optimizations (Actual)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface with sensor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface with sensor (Actual)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implement software</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implement software (Actual)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Documentation (Actual)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demonstrations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 3: Semester 2 Schedule Tower and Car

Turret Plan

Our original plan for the tower started with getting the turret firing by the middle of October. Then we would move on to analyzing the code and doing code reorganization. At the same time we would begin documenting the code as we learned more about how it worked; this was to be completed by the beginning of November. The code analysis would be completed by the end of October and we could begin moving to basic optimizations and refining the tracking algorithm for better accuracy and performance. This would be completed by the end of November to give us time to completely prepare for the demo at the end of the semester (mid-December).

Our execution of this plan has been largely within our expectations. The turret was firing at the end of October (slightly later than expected) and code reorganization has been as expected. Documentation within the code has significantly improved with this reorganization and was completed on time. In addition, some basic optimizations to the code have been completed by end of December as expected. We have also added the goal of making the turret run on other computers than the one in the lab. The expected completion time is mid-January for this addition.
Our plan for second semester was to continue to improve the turret algorithm by using real world physics calculations to improve accuracy on both the target and the missile being fired. The expected completion was beginning of February for the target and beginning of March for the missile. This would lead into our second demo in the middle of March. Time after this date would be spent on creating a more mechanically stable turret (completion by end of March), complete documentation including a developer’s guide (completion by end of April), and any further optimizations we could find (completion by end of April). Then we have the final demo at the end of the second semester. Our second semester plan was significantly impacted by our sponsor’s request to be done with the project by the end of February for the Engineering Open House. This tightened our schedule and limited the scope of some of our improvements. The documentation was to be completed by the end of April.

The second semester plan was very nearly followed; we executed stage two turret optimizations, which included real-world physics calculations and some noise reduction to the incoming image data. During this time the turret’s mechanical stability was drastically improved. We began documenting the code at the end of February and proceeded to complete all documentation, including user manual, developers guide, and the website by the end of April. The code documentation itself took longer than anticipated and has been completed at the end of April along with the rest of the documentation. In addition, work on the website began two weeks earlier in March than was anticipated. This was primarily due to the decision to work on all documentation-related tasks as soon as the Engineering Open House was completed.

**Car Plan**

The plan originally was to complete the car in 4 parts. First, it was necessary to understand the legacy code and hardware. This was scheduled with a soft deadline; it was hoped that this could be completed before November, but it wasn’t crucial, as long as the end of semester demonstration was met. It was also important in order to nail specific deadlines as well, so that reasonable estimates for deliverables could be made. After this step, the schedule could be adjusted and refined. Next, we planned to complete enough of the car functionality to
allow for an autonomous car to drive directly forward, or in a circular path for the first semester demonstration. This was the approximate original plan for semester 1. Semester 2 also had two deliverable dates, with one midterm demonstration, and one final demonstration. The midterm demonstration required at the least to have autonomous driving of the vehicle, or the ability to navigate directly to a target destination without any obstacle avoidance or any interaction from the user. At this point, there was not planned to be any interaction with the turret. Finally, the final demonstration was where it was hoped that the two projects would come together. Any obstacle avoidance on the car, including the reactive avoidance to any projectiles from the turret, would have to be finished at this point. The car would also have to navigate completely autonomously towards its goal.

The plan was an approximation of the features that was desired at each deliverable date. As the project got underway, it became apparent that some features would be easily implemented, and some would not make it in time for the deadlines. The first semester deliverables were met without much trouble, and the car was able to drive autonomously in circles with the ability to be manually controlled if necessary. However, by the second semester, we were informed that all functionality of the car must be completed by the end of February for the Engineering Open House. For this reason, the schedule for semester 2 was hastily revised to allow for the change in expectations. The midterm deliverable was the deadline to include obstacle avoidance in addition to the autonomous driving features that were already scheduled. The autonomous vehicle goals were slightly revised to allow for the car to follow a human instead of a static goal endpoint. The legacy software was decided to be completely scrapped, and the remaining time would be spent revising it for the needs of the new goal. The first part of the remaining time between the beginning of the second semester was going to be the time allocated for interfacing between the car's hardware and software, and the second half of this time was to be spent on improving the tracking, implementing object recognition, and adding suitable path-finding. The rest of the semester was to be spent on documentation, demonstrations, and updating the website as requested by the sponsor.
In the second semester, the plan was roughly followed. The interfacing with hardware was completed on schedule, and all features were implemented for the Open House, with the exception that the path-finding was not reliable enough to demonstrate effectively. The remainder of the semester did not follow the plan as well as was hoped. A voice control feature was requested and added, and the path-finding was improved, pushing back the updates to the website and documentation that were originally planned. While the schedule was only approximately followed and some features were not as impressive as initially planned, all deliverables were met for the deadline successfully.

Development Approach

Turret Development Approach

The development approach of the turret sub-project was focused primarily around an incremental agile-based development cycle. The turret team organized project goals and subtasks around project deadlines and time estimates. As development took place, estimates were refined and used to better estimate the time taken for future tasks. In addition, hardware improvements to the turret’s stability were executed in parallel to the code development. The code was developed by using GIT for version control. With GIT all changes to the code would be easily accessible and available, in addition to allowing for rollbacks and other useful code revision features.

The turret team would meet with the car team and/or the sponsor to discuss changes to the list of tasks and or execution status as often as twice a week. These meetings also served to flesh out demos and end-goals.

Car Development Approach

The main approach to development of the Car was to follow an Iterative and Incremental development model. Specifically, the Car team followed an agile development process of iterative and incremental development. This allowed the team to implement features as needed for each deliverable, rather than accomplishing the same amount of work without having a viable demonstration at each deadline. An additional advantage to using this
development model was the emphasis that it placed upon self-organization and cross-functional teams, and the encouragement of adaptive planning. This was crucial, as the team of 3 was too small to have a formal process and leader.

Each of the main requirements of an agile process were followed. This includes self-organization, planning, and delivering working software to present as opposed to emphasis on documents or design. This also includes customer collaboration and the ability to respond to feedback. Each of these principles was met by the Car team. The team was required to be self-motivated, especially without a specific manager to manage schedules and assignments. During the second semester, meetings with the sponsor, our customer, were increasingly commonplace. It was usual for a new feature to be presented at each meeting, where feedback was accepted and used to improve the car. The agile process fit best with the Car team, and was a major reason for the success of the project.

**Design Decision**

The main goals of the project were focused around demonstrating the capabilities of simple technology in an entertaining fashion through the interaction of an autonomous vehicle and turret. An expressed desire of the sponsor was to have some sort of human interaction with both the car and the turret. In addition, a sub-goal of the project was to demonstrate the significant advantages that could be obtained with parallelism. Specifically, the idea was to demonstrate the speed at which parallelism would allow a limited set of hardware to respond to a stimulus, which in this case was the turret firing a projectile at a speed that would be difficult to detect and avoid. Unfortunately, given the tools and technology available within a reasonable budget, it was determined that the sub-goal of demonstrating parallelism was not reasonably achievable. Instead, with agreement and support from the sponsor, the teams decided to focus primarily on the first goal of providing an entertaining demonstration between an autonomous car and turret, with human interaction provided to keep it interesting while still remaining autonomous.
**Turret Design Decisions**

The turret’s design is largely influenced by the previous team’s work. The libraries, languages, and hardware used were “done” when the project began. This includes the decision to use the OpenCV and OpenKinect libraries to do image processing and to write the code in C/C++. The original code was written primarily as C code, but had some C++ constructs and libraries being used and as such were compiled as C++. The code was reorganized in stages to better modularize and understand the underlying execution. The first stage consisted of splitting the logic being executed in the sizable main function into smaller functions. Then, for the second stage, these functions were reorganized into a “turret” C++ class for those functions. In order to provide more useful error messages and code readability, exceptions and other C++ coding conventions were utilized in this reorganization. General code cleanup also took place during this stage, removing many extraneous variables and useless function calls. Due to this code rewrite, the compiler can now compile the code with heavy optimizations (-O3).

With a primary focus on the first objective, the turret’s main focus was on ensuring reliable execution and operation of the target finding algorithm and its real-time processing. With this in mind, work was done to ensure that USB initialization of the turret, servos, and Kinect were correct and centered upon startup. The program also was adjusted to be able to change firing modes on the fly without recompiling for demo purposes. There were also some optimizations to the image processing portion of the algorithm to better recognize the target. This included identifying the largest orange object in the picture, rather than any orange in the picture and also filtering out small objects in the background. This allowed the turret to operate without jitter while moving, due to constant micro-readjustments. The software uses three windows to visualize what is being seen by the Kinect and what is detected by the algorithm. This is useful in both debugging and demoing. When firing the turret, depth information is used along with x-y position in the algorithm to track the time it should take to reach the target and adjust the firing angle accordingly. This approach augments the existing Kalman filter that is used to estimate the target’s next position using only the 2d Kinect camera image.
In addition to the software decisions, the hardware was mechanically stabilized as to perform smoother and less jittery operation. This included remounting the turret on a new stand and stabilizing it using clamps to a table. This proved to be much more useful and sturdy than the previous design which was permanently screwed into the top of an old Dell Pentium 4 computer tower. In addition the turret’s mount was also physically separated from the Kinect mount so as to keep their motion independent.

Car Design Decisions

With a focus on the first objective, the functionality of the car was revised to allow for the most interesting and entertaining demonstration that would also showcase the power of an autonomous vehicle. This desired functionality at the start was to move autonomously, to locate a destination and be able to navigate to it, and to avoid obstacles along the way. In order to achieve these goals, the current tools and equipment available from the last team was evaluated. With the given equipment, it would have been difficult to detect projectiles with accuracy enough to avoid them. Also, due to the realization that the car would mostly operate inside, it was found that GPS navigation, as originally planned in order to navigate to a destination, would be inaccurate and unreliable. Instead, the Car team focused on developing a car that would autonomously navigate to follow a human, providing the interaction desired by the sponsor.

The hardware received from the last team to work on this project was set up well, but the software was difficult to understand and was poorly documented. It was immediately decided to keep all of the hardware, including the specific RC car, Arduino, Atom Board, and Kinect that represented the physical components of the Car. An attempt was made to reuse the software, yet it was found to be not worth the time spent, especially with the alterations to the requirements and goals. Instead, the team started from scratch. The car was effectively split into 5 components, of which there would be two that the team would modify. The first and lowest level component was the car itself, which was an RC car which could be controlled with an RC controller, another component. Neither of these were able to be easily modified other than altering some settings. The next level component was the Arduino, which was used to
directly control the motors and servos of the car itself. A Microsoft Kinect was used as well to provide input to the Atom Board. The Atom Board itself was the main bulk of the software for this project, while the Arduino was the other component that could be modified through software. For a detailed diagram of the individual roles of each component, please see figure 1.

![Figure 1: Component roles (car)](image)

The RC car was more than capable of moving at desirable speeds in order to meet the requirements of the project. It was actually more powerful than desired, and particular care had to be exercised when using it. If there was uncontrolled input, such as from a bug from a higher layer of control, the RC car would likely damage any of the components of the car. It was quickly decided that a fail-safe would need to be implemented. This fail-safe was implemented as an override, a way to take manual control of the car regardless of the input from the Arduino or Atom Board for any reason.

The Arduino, with a 20MHz clock speed, was significantly slower than the Atom Board. The Arduino was effectively the interface between the Atom Board and the RC car. The majority
of the work was left for the Atom Board, but it was decided that the processing power of the Arduino was still powerful enough that the Arduino was designed to be an independent component with its own specific logic and functionality, as opposed to simply passing the commands of the Atom Board to the motor and servo. At this level, the car was limited in speed and control so that erroneous input from the Atom Board would not cause harm to the device. However, it was still possible for correct commands to be issued to the Arduino that would cause it to physically crash into an obstacle, and so it was at this layer that the manual override was implemented. Any errors in the Arduino would prevent manual control from being taken, so it was crucial that the device was simple and well tested to be bug-free. A flow chart detailing the logic of the Arduino are available in Figure 2 below.

Figure 2: Arduino flow chart (car)

The Atom Board contained most of the features that would allow the car to meet the end goals of the project. While it was possible that the Arduino could drive the car, the Atom Board contained the feedback of its physical progress in the world. The Atom Board
accomplished this by receiving input from the Kinect, which provided the input necessary to navigate or avoid obstacles. It also provided the listening device necessary for voice commands. This input passed to the Atom Board and was processed into simple instructions which would inform the Arduino how to control the car for the duration of each instruction. To achieve some form of parallelism, the Atom Board software was designed to be multi-threaded, and crude performance metrics were made to measure the performance of the execution, yet this was no longer a main goal and was not focused upon except during the initial design. The Atom Board also was the subject of the majority of the relevant design decisions. The provided Kinect was a Microsoft product, and the Atom Board already contained a copy of the Windows 7 OS, so it was decided that the team would stick with the Microsoft platform and develop in C# for speed of development and reliability. The software itself was kept minimal for simplicity of testing and understanding. The software stack used by the previous team, including the use of Microsoft Robotics Studio and the ArduPlane Arduino libraries, was no longer used in keeping with the intent of simplicity. Details on the design of the software running on the Atom Board are shown in the diagram below, Figure 3.
Tools and Technology

Tower Tools and Technology

The Tower project was passed down from a Capstone Team last semester. That team put into effect most of the hardware and software principals that the project still works on today. Though large improvements were made to the design of the program and the design of the physical components, we thought it best to keep most of the hardware and software the same in order to mitigate reinventing of the wheel. However, much of the code was actually ported so that it can now be run on different platforms with different hardware configurations. It was not tested outside of our two main systems however.

Hardware

• Arduino Micro
Microsoft Kinect
- Intel Core i5
- USB to USB Mini cord
- Microsoft Kinect
- Dual Servos
- Pan and Tilt Servo Mount
- Dream Cheeky Electronic Nerf Dart Gun

**Software**
- Ubuntu
- ArduinoCC
- OpenCV

The turret team was using a core i5 laptop computer running Ubuntu Linux. This computer is responsible for interfacing with a Kinect sensor for vision and depth tracking, a USB driven turret to shoot darts, and an Arduino board to control the servos which are responsible for aiming the turret. However, the code has been ported and scripted so that it will easily run on any computer with a Ubuntu operating system.

The Microsoft Kinect has two RGB cameras and one infrared ray sensor. The tower uses the Microsoft Kinect to get the video stream for object recognition and the depth map for 3D position calculation. The code that controls the Kinect is done through OpenCV libraries, which are cross platform. Currently, however, the code is only proted to run on Ubuntu. This is because the controlling drivers of the Arduino and Dream Cheeky turret are kernel calls specific to Ubuntu.

**Car Tools and Technology**

The majority of the technological decisions were based on the work of the previous team. Radically modifying the existing hardware architecture would have been counterproductive. The car hardware consists of components that can be broken into several layers. A quick overview of the hardware equipment and relevant software helps provide an overview of the components and software to be utilized in the development of the project. The hardware of the car is set up into 3 main components: The Atom Board, which controls the
bulk of the software and information processing, the Arduino, which directly controls the motors/servos, and the RC car itself. Shown below is a list of the hardware and software components the car is made of.

**Hardware**
- Arduino (specifically designed for ArduPlane from DiyDrones.com)
- Microsoft Kinect
- Intel Atom Board
- Traxxas RC Car
- RC remote
- Li/Po Battery for Atom Board
- Li/Po Battery for RC Car
- Li/Po Battery Charger
- USB to USB Mini cord
- Power Cord for Atom Board

**Software**
- Microsoft Windows 7
- ArduinoCC
- Microsoft Visual Studio
- Microsoft Kinect SDK

The project uses a modified Traxxas Slash RC car. Parts found on the car include an RC antenna, main drive motor, and servo control for front wheel steering. In order to send servo commands to the steering and motors, the inputs and outputs linked to the respective servos have been rerouted to the Arduino. Since the inputs for control of the car of the Arduino initially routed to the antenna, the inputs from the antenna are rerouted to the Arduino. Next the atom computer is responsible for the image processing, passing the movements commands to the Arduino. The final component on top is a Kinect that feeds in the raw image data for visual detection and recognition. In regards to power, the hardware utilizes to LiPo batteries, one for the car and one for the computer.

The current setup of the software is divided between the Arduino and atom computer. An Arduino can easily be programmed through interface of the computer and the code is very
simple to interpret and understand. Using the Arduino CC, the Arduino can be uploaded with executable code ready to run. In the project, the Arduino simply passes interprets movement data and relay control to the servos. On the computer, the car program is developed on the Microsoft Windows 7 platform. The heart of the software is handled through Microsoft Visual Studio, where the Microsoft Kinect SDK adds the benefit of using libraries for vision detection and object detection.

Results

Tower Results

The tower team has had good results optimizing and improving the previous team’s tower implementation. Any computer with a Ubuntu build, OpenCV, and the source code can now be used to run the tower. Image data is captured from the Kinect and is transferred to the computer for image processing and target detection. We use the OpenCV library to recognize the orange ball from the image by setting a color and general shape filter. After we obtain the object position in the image, we are able to adjust the turret angle to point to the target. By capturing and processing data in real time. The turret is able to track the target pretty well. We calculate the velocity of the car to predict the next movement of the target. We also measured the shooting speed of the Nerf dart, and used the equation of uniformly retarded motion to get the time the Nerf dart arrived at the target. Combine these information together, we can adjust the turret shooting angle to improve the accuracy of the shooting. In addition, the Arduino runs a smoothing algorithm to make sure the turret does not make any sudden jerking movements that could harm the mechanics and make it miss the target.

We set two shooting mechanisms. One is auto shooting, where the turret shoots automatically whenever it detects the object is in its firing range. The other is manual shooting, where it shoots when the “f” key is pressed. The current shooting accuracy is around 75-90% with some measure of human intuition on when to press the button to fire. For example, a higher accuracy can be achieved by waiting for the car to stop before shooting it, or shooting it as it drives towards the tower rather than away. The alternative is using a prediction algorithm
of the computer to figure out when to fire. This prediction of the best launch point is most accurate when the target is moving in a straight line or not moving. Accuracy has been found to be around 60% when shooting at the car following a human. This result is very promising.

**Car Results**

The development of the autonomous car solution has been a two semester project. All the software was created from the ground up, with the team concentrating on developing a software interface with the car, Arduino, computer and Kinect. This car is intended to be a fully autonomous vehicle that interacts with users and avoid obstacles. The current implementation of the Seek and Destroy project entails an autonomous car that identifies the skeletal structure of a person, locks onto the person in frame, and proceeds to follow that person.

The key functions currently implemented in the car design include basic voice commands, human skeleton detection, motor control, object detection, and a manual Remote Control override. Voice recognition provides a basic, yet additional feature where the user can verbally express start or stop. In regards to image detection, the project successfully uses the Microsoft Kinect to detect the skeletal structure of a person to be followed. Using the visual data gathered from the Kinect, commands are than relayed to the Arduino and passed to the motor and steering servo controls.

In its current state, we have implemented a speed/distance detection algorithm and refined motor control to improve functionality. This feedback loop helps to accommodate for the precision of the motor and steering servos. The servos lack the required precision and in implementation, can skew the results (e.g. turn too far left, drive further then intended). In terms of object avoidance, this feature is currently partially implemented. The software is capable of using the Kinect to obtain depth plane recognition for object detection. As described in the design section above, the filters and identifies changes in the depth plane. In the implementation of object avoidance, the program still struggles to properly identify an object. In its current state, However, object avoidance struggles with inconsistent image processing and false positive object detection, prohibiting a complete and fully functioning implementation.
Lessons Learned

Developing an autonomous robot is a huge challenge to overcome and with any project, there are problems to be learned from. When designing, implementing, and testing the robots, we learned many valuable lessons. The first and foremost of these lesson is our team needed more communication with the sponsor. During the first semester we learned the sponsor’s feedback was important for defining the direction of the project. We initially thought we were communicating with the sponsor well enough, but from input, we learned the updates sent out were far too vague. We were able to fix this problem by having more consistent face to face meetings and correspondences through email. Only after increasing our coordination with the sponsor did we realize how important and detrimental to the success of the project.

In the design of the project, the car team learned a lot. None of us were familiar with image recognition algorithms. Over the course of the project design, we were able to learn the necessary computer vision techniques to follow a person. Also while developing the design, we learned the importance of using libraries and tools, such as the Microsoft Kinect SDK. This SDK proved vital in our team understanding the operation of the Kinect. This includes how to send Kinect data and process images. However, the most valuable lessons learned were the ones from overcoming challenges and relate directly to the interfacing of the components and the movement algorithm. When testing new software, we realized autonomous solution can be unpredictable at times. We had runs where the car would take off, or run into the wall, but we quickly learned from these incidents. We learned the necessity for taking the proper precautions when testing software on hardware. This helped us to understand the importance of a proper test environment and the reason a fail-safe was implemented for good measure. Another important lesson came from the testing and debugging of the car movement and tracking algorithm. We realized the servos were inconsistent and required frequent calibration. This arose the problem what we were sending the car to do was not necessarily happens. As a result, we struggled with testing and debugging. Finally, we developed a feedback loop and
realized how much easier fixing a problem is when you know what is going on. We learned the importance of knowing the difference between the sent data and performed data.

The final lesson we learned was how manage communication between a group and properly document and creating a repository. Due to the lack of documentation from the previous team, we had zero working code to be reused for the car. Since we began the software design from scratch, we learned the entire process of the software development, and the challenges accompanied with each iteration of the project. We learned to properly manage version control as we progressed from different phases of the project. Also, we found module testing to be invaluable when debugging, fine tuning, and optimizing parts of the software. In the end, the car team gained the experience of developing project and all its associated challenges. We overcame our challenges to create a successful project.

Member Contributions
For the current Team Seek and Destroy project, members Eddie Andert, Tom Barry, and Roger Dolan collaborated together to implement an autonomous car. Team members Eric Barber, Shang Wang, and Michael Vetrano collaborated to implement the tower. In splitting the workload and developing the both pieces of the project, the work contributed by each member was as follows:

Edward Andert
- Helped design and debug Arduino Code for car (failsafe, byte sequence, bit meanings, turn distances).
- Helped test and debug code for the car project.
- Added sound recognition for voice commands.
- Distributed workloads to teammates and coordinated meetings and demos.
- Coordinated demo at open house (signup, signage, demo video)
- Filmed and edited the final video.
- Compiled and edited final presentation.
- Compiled and edited final report.
- Presentation to sponsor’s class
• Demonstrated at open house

**Eric Barber**

• Built platform for turret (Physically, hammer and tools)
• Helped assemble pan and tilt mount for turret
• Aided in calibration of pan and tilt system
• Soldered numerous pieces in the Dream Cheeky turret board and Arduino Micro.
• Error checked problematic code
• Debugging and test for the turret on numerous occasions (Engineering Open House, internal meetings before deadlines, etc.)
• Turret sections of the introduction, purpose, scope, and description for the final report and presentation
• Presentation to sponsor’s class
• Demonstrated at open house
• Wrote user’s guide

**Michael Vetrano**

• Turret time-to-target estimation improvements
• Change turret firing mechanism without recompile
• Code organization improvements and refinements
• Status report 2 turret achievements/final release content and presentation
• Contributed to updated mechanical turret design
• Debugging and test for the turret on numerous occasions (Engineering Open House, internal meetings before deadlines, etc.)
• Turret pan and tilt calibration/code changes
• Turret sections of the schedule, development approach, and design decisions of the final report and presentation
• Presentation to sponsor’s class
Demonstrated at the Engineering Open House
Turret Developer’s Guide
All Website Updates

Roger Dolan
- Worked to design the code for the Arduino interface with the computer and motors
- Contributed to implementation of the path algorithm used to track a person
- Extensive system testing of car features including servo calibration (software/hardware), manual override, and the path algorithm
- Helped with the redesign and optimization of the code
- Wrote the car technology, results and lessons learned for the Final Report and the car results and lessons learned for the presentation
- Presentation to sponsor’s class
- Demonstrated at the Engineering Open House
- Contributed to half of the Car Developer’s Guide

Shang Wang
- Filter the background noise to stabilize the turret.
- Detect the contour of each object.
- Calculate the size of each object in the image and keep track of the largest one.
- Test the accuracy and refine the shooting algorithm.
- Wrote the car technology, results and lessons learned for the Final Report and the tower results and lessons learned for the presentation
- Comment the code and write documentation.
- Presentation to sponsor’s class
- Demonstrated at the Engineering Open House
Tom Barry

• Basic design and implementation of the Car Atom Board
• Redesign and implementation of final version of code for Arduino
• Contributed to implementation of basic humanoid tracking, object recognition, manual override, and object avoidance for Car
• System testing of Car features, including humanoid tracking, distance calculation, manual override, object recognition, and object avoidance
• Car Sections of the Schedule, Development Approach, and Design Decisions of the Final Report and Presentation
• Demonstrated at the Engineering Open House
• Contributed half of the Car Developer’s Guide

Conclusion

Over the course of the year of Capstone, Team Seek and Destroy successfully completed what we set out to accomplish. The final products are free of major bugs and accomplishing the goals that were set for them. The tower is successfully shooting the bass with a higher accuracy and a higher tracking frame rate. The tower was built a more stable base, and the tracking algorithm was smoothed out to eliminate needless jerking. The car is successfully driving autonomously to a specified target. The car can follow a human around with great accuracy and speed. The car has minor object detection built in, although admittedly the avoidance maneuver is miscalculated. The car also takes voice commands like go and stop to start and stop the following algorithm.

Working on this project was not always easy. The project entailed combining and utilizing many skills that the team was not familiar with. One of the big ones was communication with the sponsor. Another was dealing with scope changes and directing the teams work towards an area that needed it. Programming of computer systems that interact with the physical word is not the same as working in the virtual world. This also posed a challenge for the team.
Project management was also a large factor in this project. Defining team member roles was key to making sure the project was on track. By the end of the project, the roles of our members were fairly well defined; however it was not always that way as there was some uncertainty at the start. We also had to hold ourselves accountable to the deadlines that we set ourselves. Time management was a real problem at times, especially when all the members of the team would get busy as a deadline or demonstration was coming up. In the end, we worked through all of this and ended up learning a thing or two. The project taught the team many lessons. Some lessons were on how things work in a group dynamic, others on how to create working and accurate cyber physical systems. In the end, this Capstone class taught the team far more than any one of us would have thought it would.

The future for this project has been speculated to involve turning the robotic car into a product. This product could be used as an entertainment and learning tool for children. Alternative uses include as a novelty item (everyone wants a robot) and as a programming tool for schools and universities. The robot needs a vast amount of improvement for this including a complete program redesign since our programming was never meant to house more functionality than it already does. In addition new hardware would be needed which would include stepping away from using the heavily limited Kinect.

Regardless of where the project ends up in the future, it was a fitting end to our undergraduate degrees. And with a final parting farewell to Capstone and our undergraduate degrees; this is Team Seek and Destroy signing off...