Y-CLOPS

BRIGHAM YOUNG UNIVERSITY

TEAM MEMBERS: Evan Andersen, Chris Archibald, Aaron Dennis, Spencer Fowers, Christopher Greco, Nicholas Jepsen, Peter Lamb, Kirt Lillywhite, Kent McGillivary, Justin Perry, Beau Tippetts, Chris Young

FACULTY STATEMENT: We, Dr. Dah-Jye Lee and Dr. James Archibald of the Department of Electrical and Computer Engineering at Brigham Young University, certify that the design and development of Y-Clops has been significant and that each student performed this work as part of a senior design project and as an extracurricular project.

Dah-Jye Lee

James Archibald

1.0 INTRODUCTION

The Intelligent Ground Vehicle team from Brigham Young University is pleased to present Y-Clops, an autonomous vehicle designed and built by 12 senior-level students in computer and electrical engineering. Using entirely new hardware and software developed specifically for the 2006 competition, Y-Clops features a unique tread system, vision-directed control, and a machine learning algorithm that allows its performance to improve with each run. Because Y-Clops employs only essential and inexpensive sensors, the design clearly demonstrates the impressive potential of inexpensive autonomous vehicles that use only passive sensors. This report details the design of Y-Clops, an exciting and innovative new competitor in the IGVC.

2.0 INNOVATIONS

Y-Clops is a revolutionary vehicle that brings several new innovations to the IGVC, including:

- Robust ***caterpillar-style tracks*** for stability, a true zero turning radius, and all-terrain operation
- An ***intelligent machine learning algorithm***
- Effective autonomous operation at 4+ miles per hour – using ***a single camera*** as input

Each of these systems, along with others explained below, was designed and implemented by our team from the ground up. During the last year we made custom treads, programmed dedicated GUIs, coded our own algorithms and sensor libraries, and designed and built our own circuit boards. We even built a custom fiberglass shell.

3.0 DESIGN METHODOLOGY

To encourage collaboration and maximize efficiency, we followed the development process recommended by Ulrich and Eppinger in "Product Design and Development," 3rd Edition, 2004. The overview of the concept generation phase, the first of the design cycle, is shown in Figure 1.

We planned and completed two full design cycles. Since our algorithms depend on the capabilities of our sensors and mechanical systems, our first priority was to develop a robust mechanical platform. Once our hardware was dependable, we designed and implemented the appropriate software. With a reliable mechanical platform already finished, we were able to efficiently test our software prototypes.

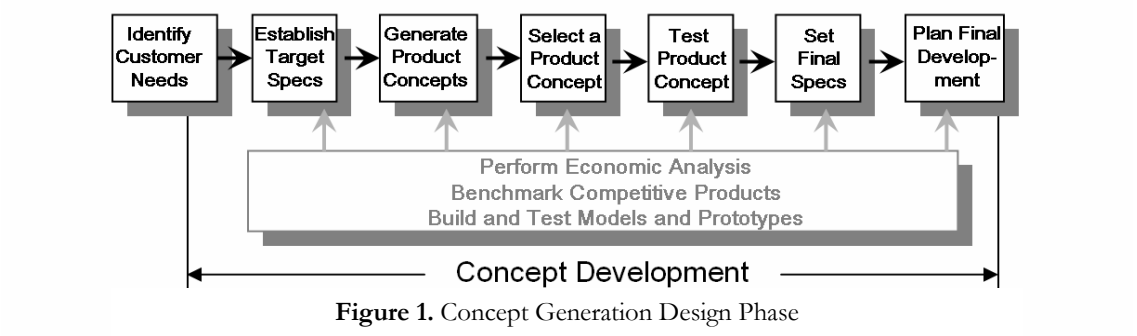


Figure 1. Concept Generation Design Phase

3.1 Exemplary Design Process

Our software development cycle is representative of our design process. First, we established preliminary specifications based on the IGVC rules and objectives. We then brainstormed image-processing strategies and narrowed our focus to three viable approaches: reactive gap-finding, machine learning, and potential fields. We decided to pursue all three approaches concurrently. To manage the design process, we divided into three sub-teams and scheduled firm deadlines and feasibility tests for each team. After a series of performance evaluations we found that while each algorithm performed well, the gap finding approach was most effective.

During a final software integration cycle, we incorporated the strengths of each algorithm into the final design. Specifically, our experience with machine learning enabled us to program a learning decision tree, which we then trained with the successful reactive algorithm. Our experience with a variety of algorithms allowed us to develop a more effective solution than any of the sub-teams developed individually.

3.2 Team Structure

Each member of the Y-Clops team was assigned to a sub-team for algorithm development as well as to an additional support team, focusing on the mechanical system, sensor libraries, or business needs of the team. With two sets of teams, everyone was involved in software development while the other needs were still met. Additionally, overlapping teams fostered collaboration, ensuring that no team became isolated. We estimate that our team spent a total of 4800 hours on this project over the last year. Our team's organization is shown in Table 1.

	Name	Algorithm	Major		Name	Algorithm	Major
Mechanical	Beau Tippetts	Learning	CE	Business	Nicholas Jepsen	Potential Field	EE
	Spencer Fowers	Potential Field	CE		Peter Lamb	Potential Field	EE
	Kirt Lillywhite	Gap-finding	CE		Chris Young	Gap-finding	EE
	Chris Greco	Gap-finding	EE	Sensor	Justin Perry	Learning	EE
	Aaron Dennis	Learning	EE		Evan Andersen	Potential Field	CE
TA	Chris Archibald	Gap-finding	CE		Kent McGillivary	Learning	CE

Table 1. Team Organization

4.0 MECHANICAL FEATURES

Y-Clops' mechanical system is a faithful realization of the concept we developed during the preliminary phases of the design process. The CAD drawing featured on this report's cover represents our original Y-Clops concept. Figure 2 shows the physical implementation of this design, accurately capturing the essence of the original concept.

Figure 2. Y-Clops Mechanical System

4.1 Tread System

The tread system is by far the most distinctive visible feature of Y-Clops. Tank-like treads were initially proposed as a solution to a problem encountered by BYU's 2005 IGVC team. Our entry in last year's competition featured rear-wheel drive with free-rotating caster wheels in front.

The vehicle performed well on level ground, but unfortunately veered off course on slopes because it could not control the direction of the front wheels. This year, our team explored four potential solutions: front-wheel drive, course correction in software using a three axis compass, wheel encoders to monitor actual movement, and powering the front and back wheels using a caterpillar-tread system.

In the initial design phases, some members of the team questioned the feasibility of a tread system and the associated mechanical complexity. Proponents of the system, however, claimed that the added precision, power, and ruggedness were well worth the added complexity. The mechanical team delivered a full-scale prototype on schedule, and tests convinced the remainder of the team that treads were an effective option.

Our tread system includes 4 key elements, shown in Figure 2. The two rubber treads are made from a single snowmobile tread cut to fit the width of the vehicle. Each rubber tread is cut to 5 ½ inches wide by 6 ½ feet long and sewed together with 1/16" steel cable.

Through testing, we found that the electric motors provide ample torque for the tread system. We were concerned that the increase in traction might diminish the vehicle's turning capability, but instead found that the tread system allowed a true zero turning radius.

With the prototype treads, the vehicle experienced large vibrations on hard surfaces due to traction knobs on the tread. To address this challenge, we cut off the knobs and sanded the track smooth. We also added a spring-based shock absorption system to cushion the computer and camera, and installed a laptop hard drive in the onboard computer designed for higher shock and vibration tolerance. The vehicle now performs well on cement and experiences less vibration on grass than wheeled vehicles due to the large surface area of the treads.

4.2 Motors and Drive System

The motor system and some basic structural elements of the chassis were adapted from an Invacare Arrow Storm series wheelchair. We considered constructing our own motor and gear system, but found that the wheelchair motors and gearboxes – even if purchased new – were less expensive than generic parts of the same quality. We also felt that the wheelchair's commercially developed drive system would be robust and reliable. Since the motors were engineered to carry several hundred pounds, they provide sufficient power for this application.

The drive system includes integrated electronics for controlling the motors. This control subsystem allows us to manually control the vehicle with a joystick when not in autonomous mode.

We can also adjust parameters such as acceleration, torque, and turning sensitivity with an external programming device.

Although the platform is extremely reliable, we cannot guarantee the lifetime of the motors because they were purchased second-hand. As a precaution, we purchased commercially-available spare motors for the unlikely case of motor or gearbox failure. The entire drive subsystem can be replaced in less than one hour.

4.3 Chassis

While our team opted to use the basic structural elements of the wheelchair frame, we made numerous modifications and additions. After removing the unnecessary pieces we drilled through the frame and bolted on the remaining structural components.

4.3.1 Fiberglass Cover

The fiberglass cover provides durable protection for the electrical components of the system and a professional finish for the vehicle. The cover encloses the computer, GPS receiver, and wireless E-stop receiver and has a watertight hatch for quick access to these components. We used MAYA and Pro-Engineer to model the cover and to create a tool path for a 5-axis milling machine.

The cover attaches to the sides of two aluminum runners that span the length of the robot. A piece of quarter-inch Plexiglas lies on top of the aluminum runners providing support for the electrical components. Each component is securely attached to the Plexiglas with bolts or industrial-strength Velcro. We opted to use Velcro because it was a simple and effective way to secure the internal components while allowing immediate removal for maintenance or replacement.

4.3.2 Shock Absorption

Each side of the vehicle employs two spring coils between the main chassis and computer platform to minimize vibration of sensitive components such as the camera and hard drive. In the front, a spring-based system consisting of a 2-pound spring attaches to the main frame and the aluminum runners, as shown in Figure 3. In the rear, we use the original wheelchair shock absorbers and springs. The springs do not eliminate vibration completely, but dampen hard jolts that could damage components.

Figure 3. Shock Absorber

4.3.3 Front Bumper

A front bumper protects the vehicle – and large stationary objects in its way – from frontal collision. If a collision occurs, the bumper presses against a

wired emergency stop to halt the vehicle. Springs attach the bumper to the robot frame, ensuring that the emergency stop is not activated unnecessarily.

4.3.4 Roll Bar

The galvanized steel roll bar provides a mount for the camera, GPS antenna, wireless antenna, and 3-axis compass. The low cost and malleability of automotive muffler pipe was ideal for this application. The roll bar extends to a height of 5'10", providing stable placement for the camera, GPS and wireless antenna. Additionally, the pipe conceals and protects cables connecting these sensors to the computer.

5.0 ELECTRICAL SYSTEM

Our electrical components were selected to give us a reliable computing platform, accurate control of the wheelchair, and power for the entire system.

5.1 Computer

We elected to use a standard desktop PC for maximum computing power. It runs Microsoft Windows XP on an Intel dual-core 2.8GHz Pentium 4 processor. As previously noted, the computer uses a laptop hard drive which has a much higher shock tolerance than standard hard drives. Since the robot is powered by batteries, we replaced the standard AC power supply with a more efficient DC to DC 24-volt power supply that connects directly to the robot's battery supply.

The computer is the central hub of our control system, integrating input from each sensor and supporting the necessary interfaces. The computer also includes 802.11g wireless access to allow remote monitoring, effective debugging, and JAUS functionality.

5.2 Power Delivery System

Y-Clops is powered by two 12-volt marine deep-cycle batteries. The batteries are connected in series to provide the 24 volts required by the motors and motor controller. A pair of batteries can be charged in two to three hours and each pair lasts for approximately 2 hours per charge. A power conversion module attached to the front of the chassis converts the 24-volt supply to meet the needs of the wheelchair drive system.

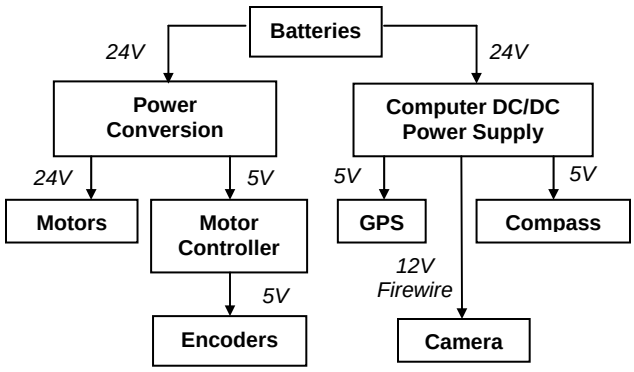


Figure 4. Power Distribution System

5.3 Control Board

The original wheelchair included a joystick drive system. The system included an inductive joystick which translated the 5 volt input voltage into a suitable drive command (0.2V to 4.5V for horizontal and vertical drive) that was then sent to the motor controller. In order to facilitate computer control we had to create a circuit that would receive serial commands via RS232 interface and translate them into these desired drive voltages.

Figure 5. Control Board Layout

Our team designed the schematic and printed circuit board layout of the circuit, shown in Figure 5, and had it professionally milled. After assembling the board, we programmed it to produce the required voltages with a pulse-width-modulated signal. The board includes an 8 MHz Atmega8 microprocessor with several general purpose input/output (GPIO) pins. The processor communicates with the computer's serial port via an RS232 chip to receive drive commands from the computer and transmit data from the quadrature encoders to the computer. Two additional GPIO pins output the necessary pulse-width modulated control signals directly to the wheelchair motor controller.

6.0 SENSORS

We carefully selected sensors for Y-Clops to provide sufficient information to autonomously make intelligent decisions. In addition, we made a conscious effort to use as few sensors as possible while extracting as much information as possible from those sensors employed. The result is a simple, low-cost platform that is nevertheless capable of meeting all the demands of the IGVC.

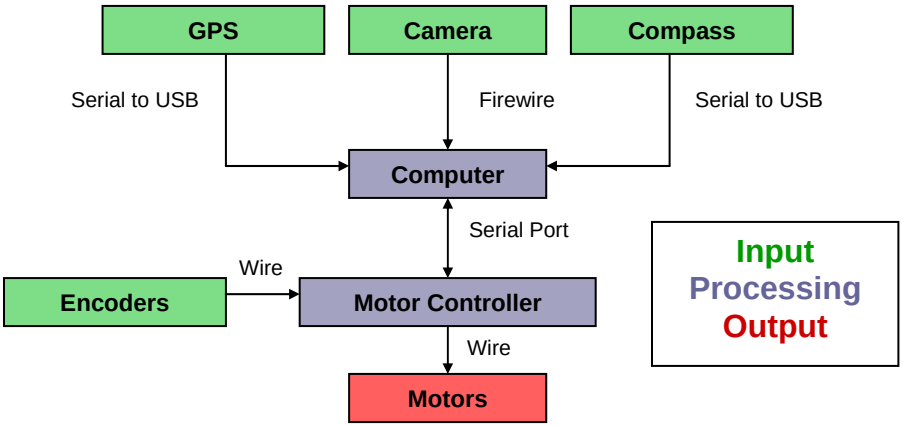


Figure 6. Sensor System Integration

Although the rules of the IGVC do not reflect this, there are many applications of autonomous vehicles where active sensors are undesirable, either due to interference with the sensors of other

vehicles or because it is undesirable to allow observers to easily distinguish autonomous vehicles by their emissions. To maximize the applicability of our approach, we opted to fully explore the capabilities of passive sensors. Most importantly, we chose to rely on vision sensors rather than laser range finders.

6.1 Camera - MicroPix C-640 Firewire with 6mm lens

This camera transmits video over a Firewire interface at 30 frames per second. Through testing, we determined that a 6mm lens provides a field of vision appropriate for our algorithms.

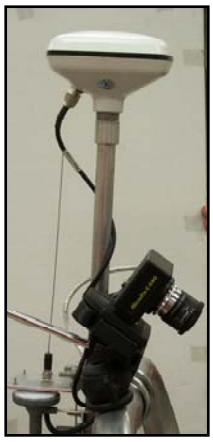


Figure 7. GPS Antenna & Camera

6.2 GPS - CSI Wireless MGL-3 antenna with MiniMax receiver

Our differential GPS system is accurate within 0.5 meters and refreshes at 5 Hz, well within our target specification of 1 meter accuracy. The frequent updates enable the vehicle to drive quickly while maintaining an accurate course in the navigation competition.

6.3 3-axis Compass - Honeywell HMR3000-D21-232

We chose a 3-axis compass to measure tilt of the vehicle, as well as direction in a plane. This decision addressed the previously mentioned problem in which last year's vehicle veered on hills. Sensing that Y-Clops is on a hill enables the vehicle to adjust turn commands appropriately. The compass updates at more than 14 times a second allowing high speed during navigation.

Figure 8. Compass

6.4 Encoders - US Digital incremental rotary shaft - H5S-110

The wheel encoders also measure distance traveled and provide feedback information whenever the vehicle is running. The encoders may be used for navigation in case of GPS failure or poor GPS signal strength.

We attached the encoders to the bottom of the front axles, which are not powered directly, but turn as the treads pass over them. The encoders output a quadrature square wave pulse, which is interpreted by our controller board and communicated to the main computer.

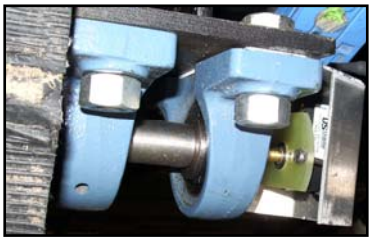


Figure 9. Encoder Interface

7.0 SOFTWARE

Superior software is key to Y-Clops' superior performance. To give Y-Clops the ability to steer intelligently, we implemented a decision-tree, which is a powerful machine learning algorithm,

along with a unique training method. We have designed and coded various graphical user interfaces (GUIs) that enable us to efficiently run, modify, and train our algorithm. With the exception of the OpenCV vision processing library, members of the Y-Clops team wrote all the code that we execute, including the GUIs. Our code is written primarily in C, C++, and Java.

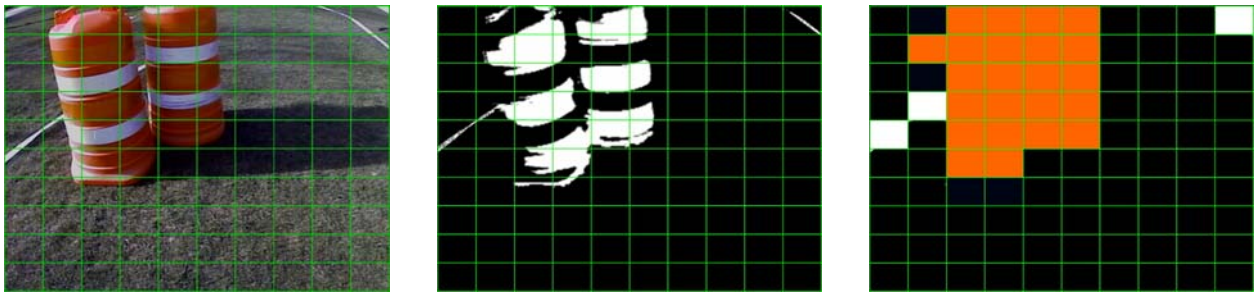
7.1 Image Processing

To interpret images from the camera, Y-Clops extracts color information, specifically about orange and white. First, the image data is converted to hue, saturation, and luminosity (HSL) values. These HSL values allow for faster processing of the image and also allow us to easily isolate color ranges of interest. For example, white appears with high luminosity. Both the autonomous and navigation programs generate commands for the vehicle based on information about orange obstacles and white lines.

Color segmentation is a fast, reliable method for extracting relevant color data from images. However, as lighting conditions change and shadows appear, color segmentation may miss important features of the image. Additionally, wide color ranges may propagate noise to subsequent processing steps.

To solve these problems, Y-Clops monitors ambient light intensity with the camera and dynamically adjusts the exposure time to adapt to lighting changes. If a cloud obscures the sun, the camera settings are automatically adjusted to maintain an accurate view of the environment.

In addition to color segmentation, Y-Clops is configured with alternate image processing capabilities. One method, called a Laplacian Pyramid, isolates features within a certain size range that also have sufficiently defined edges, regardless of color. With this approach, thin features like lines appear consistently without noise, regardless of lighting conditions.



Figures 10-12 (L to R). Original Image, Color-processed Image, 10x10 Grid Data Representation Result

7.2 Data Representation

The end result of image processing is a data representation useful to our software algorithms. Each algorithm uses the same representation, so the image processing is common to

each algorithm. The image processing software first divides the image into a 10x10 grid of cells. After orange and white have been isolated, the software compares the percentage of white or orange pixels in each cell with a pre-determined threshold. If the percentage of pixels of a given color is above the threshold, that cell is considered to be either white or orange. The final 10x10 grid then becomes input to our software for both the autonomous and navigation portions of the competition. This process is illustrated in Figures 10-12.

7.3 Autonomous Software

Throughout the development of our autonomous algorithm, our goal was to make Y-Clops adaptive and trainable to deal with a variety of courses and special exceptions. To accomplish this, we decided to use a decision tree implementation developed specifically for this purpose. We also designed a system to train this decision tree with a reactive gap-finding algorithm, which is described below. Additionally, we developed a method and tools to save and alter decision trees, allowing the algorithm to adapt and improve with each run.

7.3.1 Why a Decision Tree?

Several factors influenced our decision to use a decision tree:

1) The decision tree requires far less training time than other machine-learning algorithms we tested, such as Q-learning and artificial neural networks.
2) Decision trees consider many cells at once and can even consider past images in making a decision. Our simple reactive algorithm made decisions based on the single most important cell in an image.
3) Our decision tree can be trained, and its performance improves with additional training. With the tools we developed, we can easily review and adjust the behavior of an algorithm – drastic improvement can be seen from one run to the next. Last year, BYU's vehicle failed to improve after its first run of the competition. The decision tree allows Y-Clops to avoid the same shortcoming.
4) Finally, considering past images gives Y-Clops short-term memory. Adding memory into last year's code would have been a herculean task. The training process for the decision tree, however, was configured to handle memory with minimal effort.

7.3.2 What is a Decision Tree?

A decision tree is a type of classifying data structure. The tree may be considered a hierarchical set of rules. Given input divided according to various attributes, the tree determines the proper output. The training process identifies the most important attributes of input data, and

creates a tree of simple questions that can be asked about the input. The result of these questions is the output corresponding with a given input. The beauty of a decision tree approach is that the programmers do not have to specify which attributes are important! The decision tree identifies key features on its own using the given training data. Furthermore, a decision tree does not require every combination of inputs to be tested exhaustively. The tree determines which attributes contribute the most to the decision process and ignores less important attributes.

After training is complete, the decision tree can quickly and accurately respond to conditions it has never encountered. For Y-Clops, the input to the decision tree consists of three 10x10 grids: from the current image, the image 3 seconds ago, and the image 6 seconds ago. Figure 13 shows a simple example of a decision tree for a 2x1 grid, giving us outputs of straight, left, right, or stop.

We initially trained the decision tree with a human driver. However, a human perceives much more than a camera and therefore uses information not available to Y-Clops. Instead, we trained the tree with a reactive algorithm, which performs very well on its own.

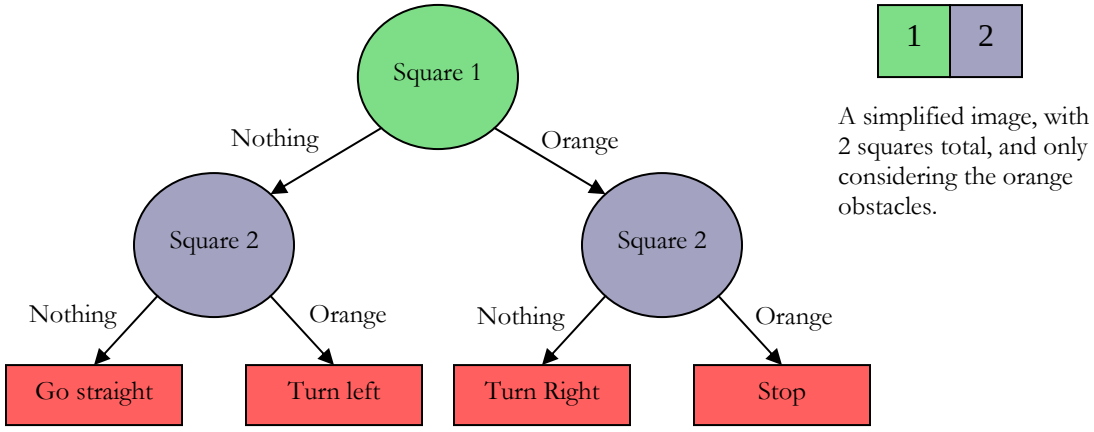


Figure 13. Simple Decision Tree Example

7.3.3 Training Algorithm

The reactive gap-finding algorithm we developed uses a look-up-table (LUT) representing the importance of each cell in the 10x10 grid. The algorithm finds the orange or white cell value of highest importance, and then references another LUT to determine the turn command associated with that cell. To decide which direction to go around obstacles, the algorithm finds the distance to the nearest white on each side of the obstacle. The algorithm then sends the turn command that will steer the robot toward the biggest gap, avoiding traps and dead-ends. This algorithm has been tested extensively and has proven accurate and reliable. However, it incorrectly handles certain arrangements of barrels and dashed lines, making the decision tree superior.

7.3.4 Tools

Three tools streamline the process of building, manipulating, and using decision trees. First, the main GUI allows us to run our software and gather information from our reactive agent. Several runs are then opened in the review GUI, where noisy data is removed and problems are analyzed. That information is then passed to the decision tree trainer which generates a decision tree file. This file can then be used in the main GUI for the autonomous algorithm and any problems with the tree can be edited within the review GUI. These GUIs allow us to monitor and direct the training of the decision tree, to ensure that Y-Clops learns correctly.

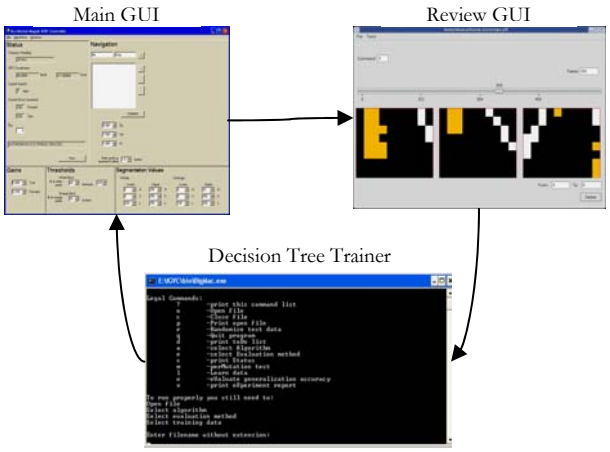


Figure 14. Interaction of Autonomous Software Tools

7.4 Navigation Software

Our navigation algorithm employs the GPS, 3-axis compass, and color-camera as sensors. Our basic algorithm consists of the steps shown in Figure 15.

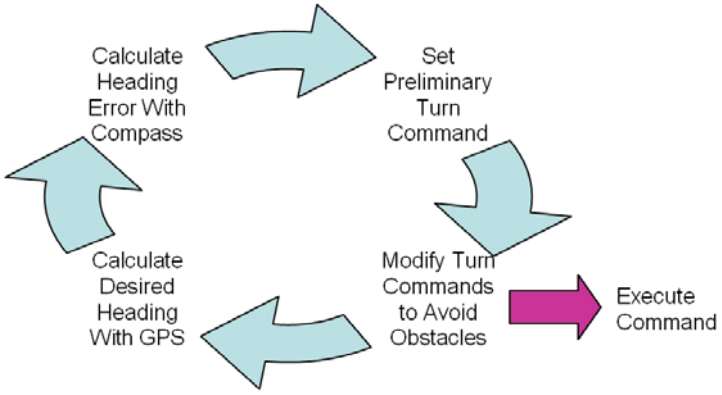


Figure 15. Main Driving Loop of Navigation Software

We also programmed a GUI to plan a path between waypoints. This program loads and displays the waypoints we specify, allowing us to draw a recommended path for Y-Clops. The program then outputs a new file with waypoints listed in the ideal order. These waypoints can be classified as goal waypoints (those specified by the IGVC) or intermediate waypoints (those we specify to facilitate a smoother path. The main GUI allows us to specify a required distance we must reach before marking a waypoint as visited, and this value can be different for goal waypoints and intermediate waypoints. Our GUI allows us to plan an intelligent path between the waypoints, but even without a recommended course, Y-Clops can efficiently reach each waypoint with the

rapidly updating sensors and obstacle avoidance. After the last waypoint is reached, Y-Clops will return to the starting box and orient itself toward true North.

8. SAFETY

Safety was a priority at every step of the Y-Clops design process. In addition to meeting IGVC standards of a wireless and hardware emergency stops, we exceeded the requirements with an additional E-stop coupled to the front bumper of the vehicle. This additional E-stop is spring-loaded, so that if Y-Clops strikes a sizeable object, the springs compress and the E-stop depresses, stopping the robot. This ensures that Y-Clops will not continue driving into stationary objects if the wireless E-stop is not pressed soon enough. We have intentionally designed the front bumper to not engage the E-stop for slight contact, such as with a barrel.

The wireless E-stop has proven effective from distances greater than 70 feet. The manual E-stop, located prominently at the rear of the vehicle, is hard-wired to disconnect the computer from the hardware motor controller to instantly stop the vehicle.



Figures 16-18 (L to R). Y-Clops' Front Bumper, Rear, and Wireless E-stops

9. SPECIFICATIONS

9.1 Speed

The maximum speed of Y-Clops may be adjusted by programming the wheelchair firmware. According to the published wheelchair specifications, the maximum rated speed is 6.5 mph. Our design, using custom treads with the wheelchair motors, runs at a maximum speed of 5.85 mph. For the autonomous and navigation competitions, Y-Clops' speed will be limited to 4.52 miles per hour in firmware.

9.2 Ramp Climbing Ability

Y-Clops can climb up and down a 14º incline, exceeding the wheelchair's 9º rating. The vehicle has sufficient traction and power to climb grades greater than 14º, but the weight of the batteries at the rear of the vehicle could compromise vehicle stability.

9.3 Reaction Time and Obstacle Detection

The obstacle avoidance system scans the surroundings at 29 frames per second. The camera detects obstacles as close as 1.5 feet and as far as 18 feet in front of the vehicle. At 4.5 mph, Y-Clops travels 2.7 inches between frames. This means that Y-Clops receives new information about the world every 2.7 inches that it travels.

9.4 Battery Life

In full autonomous mode, the batteries discharge to 30% capacity in approximately two hours. Batteries charge completely in less than 3 hours, and discharged batteries can be quickly swapped with fully-charged replacement batteries.

9.5 Dead ends, Dashed Lines, Traps, and Potholes

Our primary strategy for dealing with dead ends and traps is to avoid them altogether. However, if Y-clops detects that no clear path exists, it stops and turns in place until it finds an obstacle-free path. Additionally, the decision tree framework has allowed us to train Y-clops to recognize and avoid traps and dead ends.

The short-term memory of the decision tree allows Y-Clops to maintain a consistent direction even with dashed lines. Potholes are detected with the camera and avoided like barrels.

10. COST

Item	Qty.	Retail Value	Total Cost	Our Cost
MicroPix C-640 firewire	2	$990.00	$1,980.00	$0.00
CSI wireless MiniMax GPS	1	$1,899.00	$1,899.00	$1,899.00
CSI wireless MGL-3 GPS Antenna	1	$250.00	$250.00	$250.00
Honeywell HMR3000-D21-232	1	$699.00	$699.00	$699.00
US Digital incremental rotary shaft - H5S-110	2	$59.85	$119.70	$119.70
2.8 GHz Pentium D 1GB RAM 60GB HD	1	$770.00	$770.00	$770.00
Invacare Arrow Storm RWD Wheelchair	1	$8,000.00	$8,000.00	$28.00
Linear DX DXR-702 2 channel	1	$72.65	$72.65	$72.65
Wireless transmitter - 1 channel	1	$37.00	$37.00	$37.00
Bearings	4	$10.00	$40.00	$40.00
Encoders	2	$50.00	$100.00	$100.00
Misc hardware	n/a	$150.00	$150.00	$150.00
Tracks	1	$350.00	$350.00	$350.00
Roll bar	1	$20.00	$20.00	$20.00
Batteries	2	$60.00	$120.00	$120.00
Fiberglass shell	1	$200.00	$200.00	$200.00
Total			$14,807.40	$4,855.35

Table 2. Costs of Y-Clops System Components

11. CONCLUSION

BYU's 2006 IGVC entry is a unique and innovative autonomous vehicle. Because our team designed and developed the mechanical and electrical systems, the software, and the algorithms specifically for this project, each component was designed with the whole system in mind, increasing overall performance and usability. Among its most innovative components, Y-Clops includes a tread system that gives it a robust and stable base, a true zero turn radius, and superior traction. Y-Clops also includes a unique decision-tree mechanism that gives it the ability to learn and improve with experience. Our design makes the most of its sensors, extracting all necessary information from its single color camera. Finally, our handcrafted fiberglass shell gives Y-Clops a professional and powerful look, clearly suggesting speed and confidence.

The Y-Clops team is proud to be a part of the 2006 IGVC. We are confident that Y-Clops will be a very competitive entry, and we feel that innovative elements in our design will prove useful in a wide variety of future autonomous vehicles.

Figure 19. Y-Clops in Action

APPENDIX: TABLE OF Y-CLOPS SPECIFICATIONS

	Theoretical	Actual
Dimensions		
Height (excluding GPS antenna)	--	5’ 10’’
Length	--	3’ 3.5’’
Width	--	2’ 7’’
Weight	--	250lb
Speed		
Mode 3	4.52 mph	4.52 mph
Mode 4	5.85 mph	5.85 mph
Ramp Climbing		
Uphill	10º	14.5º
Downhill	10º	19.5º
Power		
Battery life	--	2 hours
Battery voltage	24 V	24 V
Average driving current used	38 A	30 A
Computer power supply	300 W	300 W
Total power used	900 W	720 W
Sensors		
Camera (frames per sec)	30	29
GPS (updates per sec)	5	5
compass (updates per sec)	14	14
Encoders (updates per sec)	300	29
GPS accuracy	± 0.5m	± 0.5m
Maximum distance obstacles detected (feet)	18	18
Minimum distance obstacles detected (feet)	1.5	1.5