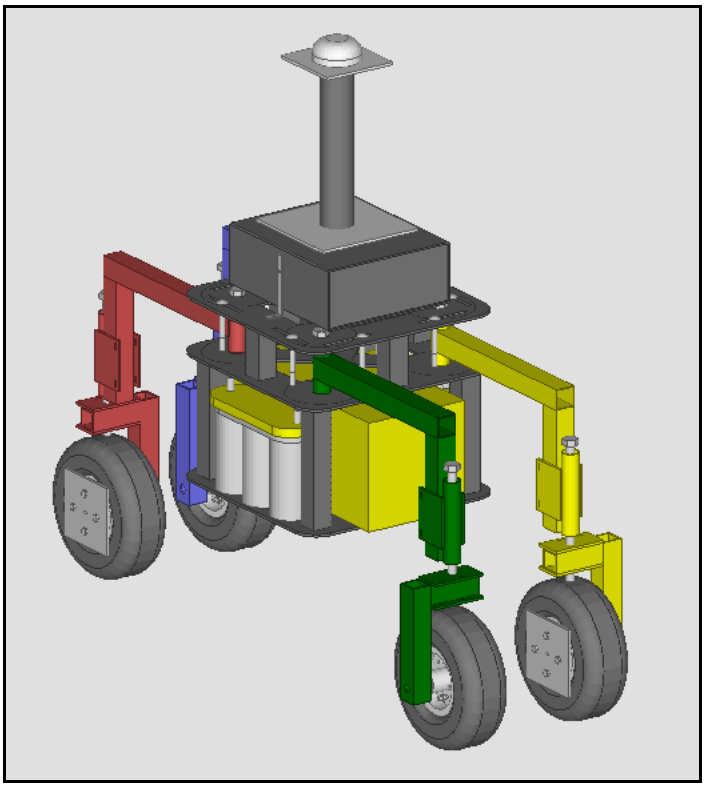
RDB3K



An Unique Omni-directional 4-Wheeled Platform
Governed by ARTMAP Neural Networks

Volunteer Team Members:

Core Members:	Contributing Members:
Programming, Concept, and Schematics: Gary Stein (Senior CS / EE)	Electronics, Funding: Tim Roberts Electronics, Machining: Richard Andres
CAD, Frame, and Construction Geoff Decker (Sophomore CpE)	Mechanical: Scott Stickler, Michael Barry

Dr. Fernando Gonzalez

Print Signed Date

I certify that the work done by all students on this project is consistent with a senior design course and that the vehicle has been created for this year's competition.

Robotic Platform Overview:

The RDB3K platform has 4 “legs”, however, each leg has a direct motor to wheel system attached that provides a driving force in a single direction of movement. On top of the drive wheel, there is also a directional motor that turns the wheel in any direction. It allows for each leg to pick a direction to drive toward. The combination of the two motors allows each leg to be omni-directional since it can travel in any direction. The design is to allow each leg subsystem to act almost independently of the rest of the vehicle. Each leg has its own set of sensors. With this independent set of sensors, the leg makes decisions on where to travel. To prevent conflicts between the individual leg systems, a control center is used. This control center uses a voting system to resolve conflicts, which could halt vehicle progress. With individually moving legs, the vehicle shape changes dynamically. This changing state can navigate obstacles more efficiently.

Design Process:

The desired concept of RDB3K is to create a unique platform. The initial theory behind RDB3K was to create a vehicle with four-wheel drive system that could travel in any direction using a reactive planner. Many ideas were proposed and experimented upon in a competitive manner until the most feasible was selected. The challenge of navigating through different terrain and obstacles makes a reactionary vehicle ideal and a proven method for dealing with multidimensional data is to use Neural Networks. By training the vehicle instead of giving it defined rule sets, the platform is able to react to ever changing environments. Instead of one central hub controlling all movement of the vehicle, four control systems communicating with each other allows for a more robust vehicle. As a result, all goals of the Intelligent Ground Vehicle Competition can be met.

Power System:

The entire power system for RDB3K is from two, 6 Volt Optima 55 Ah. These batteries were chosen for the stability of output voltage, long battery life, and ability to fit within the design space. The two batteries are strung in series to provide 12V to the motor and DC-DC switching power supply. The DC-DC switching power supply provides the necessary regulated 12V, 5V, and 3.3V to the main computer motherboard and 5V to the additional TTL level electronic circuitry. The regular 12V signal has additional fuses to provide safety in case of circuitry failure. During a rest state, the batteries provide enough power to run the boards for 24 hours. With motor draw, the usable time is cut down to 5 hours.

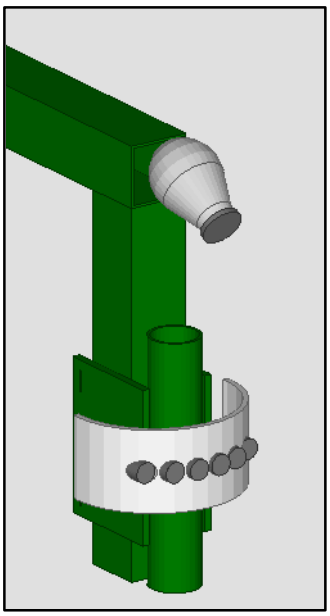
Safety System:

The main safety concerns of a rapid prototype system were addressed as much as possible without hindering the functionality of the system. Standardized wire coloring schemes, fuses, current limiting protection, emergency lights, and multiple emergency stopping systems reduce risk of an accidents. The E-Stop mechanism functions by using a Visonic Ltd. wireless receiver in the 315 MHz band. It can receive inputs from two different wireless transmitters, one of which is mounted on the frame of the vehicle and the other carried by the vehicle's operator. The receiver sets off an interrupt per microcontroller. In turn, each microcontroller sets its emergency light on and puts its motors in a halted state.

Sensors Suite:

The sensors on the platform provide interfaces with the real world to determine information about the current status of the platform. The Garmin GPS provides WAAS enabled positioning to obtain better than 3 meters of accuracy. The vision system runs off of four D-Link USB Web cams. Each is positioned at the top of each leg's "knee" joint providing relative

camera information versus the position of the leg. 12 Ultrasonic range finders are separated into groups of 3 and attached in a 45-degree separation pattern on the "shin" of the leg. The leg positioning is done using gears and a potentiometer to obtain a split analog voltage that can feed back the angle of the leg from the "hip" joint. Each leg's directional motor is accompanied by a 500 clicks per revolution quadrature encoder to measure the positioning of the wheels and the current direction of travel. The serial based Gravis Stinger joystick also provides data from the user for either training purposes or driving the vehicle around in manual mode.



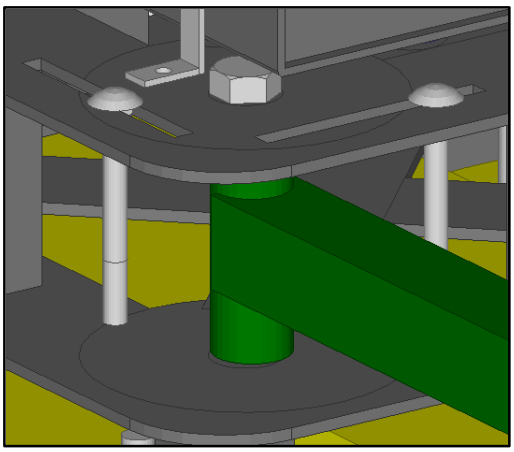
Motor Control:

The motor and underlying control system is accomplished using four PIC 16F877 microcontrollers, with each mounted individually on each leg. The same circuitry is used on all legs. Internally, the drive motor is driven by a 100 KHz PWM to a self made power transistor board to step up the voltage to a Pittman Electric Motor that is geared 1:1 to the output shaft. The drive motor rotates the 10-inch wheel at approximately 71 RPM providing a maximum speed of 2.1 MPH. The directional motor is also driven using a 100 KHz PWM, however, it drives a NAND gate logical array in conjunction with a directional pin to an H-Bridge to allow bi-directional motion. On both circuits, a current sensing resistor feeds Analog to Digital pins on the microcontroller to protect the motors and circuitry from stall in case of emergency.

The microcontrollers receive desired velocity in the form of direction and speed information from the main computer over RS-232. The microcontroller then attempts to achieve these results using a PID controller that was converted into the Z-domain to adjust for sampling

period. All the PIC controllers are programmed in C using the CCS PCW C compiler. The feedback is provided from the directional motor in the form of a quadrature signal that is decoded into motor direction and speed. This information is used to track the fork's position, which is attached through a 4:1 gear ratio. An LED board provides visual feedback of position and emergency status.

Mechanical Setup:



One of the more unique aspects of the RDB3K frame is pivoting and shape changing ability. This ability is accomplished through a system of bearings to allow free spinning motion for each of the legs. The main portion of the frame is a 3 layer shelving design to store all the necessary components. This includes batteries, payload, power supply, and main computer. Each leg is then attached to the body using a bolt and bearing system to allow the legs to reconfigure their position. The hip joint itself is not motorized and the vehicle cannot change its shape automatically in this manner. Hip joints move due to the dynamics of steering and drive motors. Carriage bolts are used to restrict hip rotation for balance protection in case of computation malfunction of drive direction.

The steering motor is mounted on the bottom of each leg to power rotation of each wheel. It does this using a chain and sprockets to a shaft mounted within a bolt and bearing system. This allows for 360-degrees of motion. The drive motor is mounted to the rotating fork housing the wheel. It receives its electricity from a system of two carbon brushes that bridge the gap between the non-rotating leg and the rotating fork and touches two circular conductive pads. The

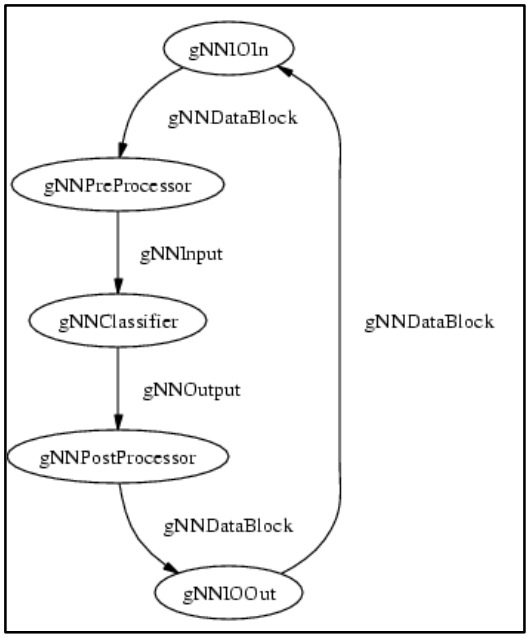
drive motor then attaches to a bracket mounted to one side of the wheel with an equal sized gear to provide motion. The other side of the wheel is attached to the fork using a bolt and bearing to provide an "airplane landing gear" style of wheel support from only one side.

Computer Setup:

An embedded small form factor computer board provides the vehicle's main intelligence. The Biscuit, from Advantech, contains a mobile Intel Pentium 4 2.0 GHz processor, 512 Megabytes of RAM, 4 USB root hubs, 4 RS232 Serial ports, and an extra PCI Serial card that provides 2 additional serial ports. A TV-Transmitter is attached to the TV-Out of the board to provide remote display to a base station. The computer is the central hub of the intelligence that is provided by external microcontrollers, GPS, and cameras.

The computer data acquisition is accomplished through the serial and USB ports. Each of the serial ports is attached to an individual microcontroller that provides feedback with respect to the wheel angle, leg angle, and sonar values. The microcontroller is attached to the three sonar modules over an I^2C serial interface. It samples each and returns the distance detected in meters from sonar every 76ms. Camera acquisition is done through USB Web cams and provides view with respect to the leg angle. The four USB root hubs are important because the amount of data passed from each camera would saturate the bus at its capturing rate of 15 fps. The GPS provides a 5 Hz position update over RS-232 serial signal with latitude, longitude, direction, and speed. Because the direction and speed information is not accurate for such low speed maneuvering, its value in the final positioning is minimal. The GPS information is processed through an additional filtering state to reduce jitter and account for satellite or signal drop. The information from the vector sum of the different legs provides a form of ded-reckoning to use as position estimation and GPS validation.

Software Platform:

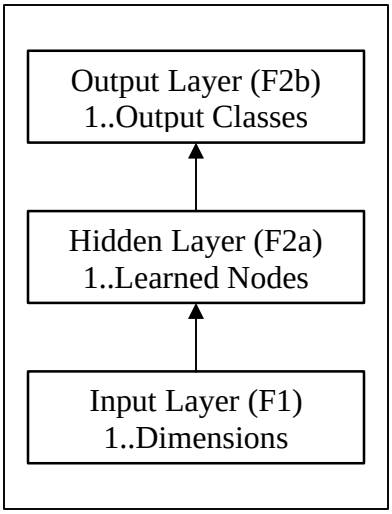


The software platform has been generalized to operate on run time without prior knowledge of the functional system blocks that are going to be running in the program. The abstraction from the classifier from the IO allows different file types to be used or different ARTMAPS to be used based on settings interpreted automatically through virtual functions. Although for different functional blocks to work, an interface must be written to convert it to a standard type. The general pipeline defined by the program is IO-In, Pre-Processor, Classifier, Post-Processor, and IO-Out. Following that chain of events produces an abstract system that can be built on using inheritance. The IO-In functions will relate to the serial ports, images, text files, cameras, and joysticks. The Pre-Processor translates those IO data blocks into useable information by converting color space and breaking up images or convert the GPS into position information. The Pre-Processor produces the Input blocks for the Classifier. The Classifier becomes the various ARTMAPS that process the data and produce output blocks for the post-processor. The post-processor does the image rotation and converts the output into motor commands to the IO-Out. The IO-Out does the physical communication to the output devices like the screen or the motor microcontrollers units.

By breaking the process into smaller blocks, the program can be done in a pipeline and reduces the overhead involved with waiting for input. In addition, it lends itself well to parallelism because different cameras will be capturing at different times, and instead of

processing one at a time, multiple images can be done while at different stages of the pipeline. The program is constructed in abstract with C++. Meaning that base functions are made and the additional functions added on will use the functions of the parent or their own functions using virtual members if overloaded. Using the idea of polymorphism, a pointer also can be made to the base class and during runtime, and the system will call the correct internal functions accordingly. By using this system of plug and play, once base classes are built, it should be easy to add additional ARTMAPs, sensors, or even different frames. The architecture is not specific to the one robotic frame but could be modified using the same abstract classes to be used on other frames.

Main Intelligence:



Almost the entire intelligence of the system is based on the ARTMAP Neural Network. The Adaptive Resonance Theory with mapping (ARTMAP) form of neural network was developed originally by Carpenter, Grossberg, and Reynolds in 1991 as a competitive form of network that eliminates bi-directional weights. The model used by this ARTMAP has three layers. A presentation layer (F1) that consists of the input for the system usually modified for use with the neural network. This modification usually normalizes each input value between 0 and 1 to be properly handled by most algorithms. The next layer is the hidden layer (F2a), which consists of nodes that store information about that particular mapping cluster. The final layer is the output layer (F2b) that contains the mapped output for a particular node. There are bottom up connections from the input layer to the hidden layer that determine the weight of how much an input maps to that node. There are also

connections from the hidden layer to the output layer to determine which output that hidden node maps to. The input layer size is determined by the dimensionality of the input data and the number of outputs determines the output layer size. The only variable layer is the hidden layer whose size is determined by the data that is presented to it.

The specific calculations for each of the different forms of ARTMAP are typically very different from each other, however, they share certain properties. Most systems use some form of geometry to represent the grouping around sets of inputs. The value returned of how close it matches a node, Rho, should always be within the range between 0 and 1. This allows for easy comparison to the bar-Rho value, which represents the vigilance on a scale from 0 to 100% to make logical sense in context of how sure the Neural Network is. The bottom up values that are calculated are not required to be between 0 and 1, but are chosen to be small positive real numbers for sorting purposes.

The vehicle was tested with two forms of ARTMAP, Fuzzy and Gaussian. The Fuzzy ARTMAP (FAM) uses magnitude of vectors and the Fuzzy Min operator to do its calculations. The magnitude of the vector is simply the sum of the vectors parts and the fuzzy min is the smallest of each vector location between two vectors. The Gaussian ARTMAP (GAM) uses standard deviations and natural logarithms to do calculations. In addition, GAM needs to keep track of the number of times that node was selected. GAM was chosen for the image classification system for its better ability to work with noisy data and less percent missed. FAM was chosen for the driving system, because of the higher speed on large dimensional data.

Vision System:

The vision system consists of several parts that are abstracted with respect to the calling functions. But the main two phases of the vision system are the camera capture and the image

pre-processing stage. The camera capture is done through the Video 4 Linux (V4L) interface in Linux. V4L is a system of headers to standardize the API for capturing from video devices. It is able to abstract the physical source of the data (USB, firewire, frame grabber) to allow one standard way to capture data and interpret into a computer readable format. More specifically, one layer below the API call interaction is the ov518 kernel driver that converts the compressed USB camera data into an Y'CbCr Image stream. That is then copied into an array, and then presented to the preprocessor.

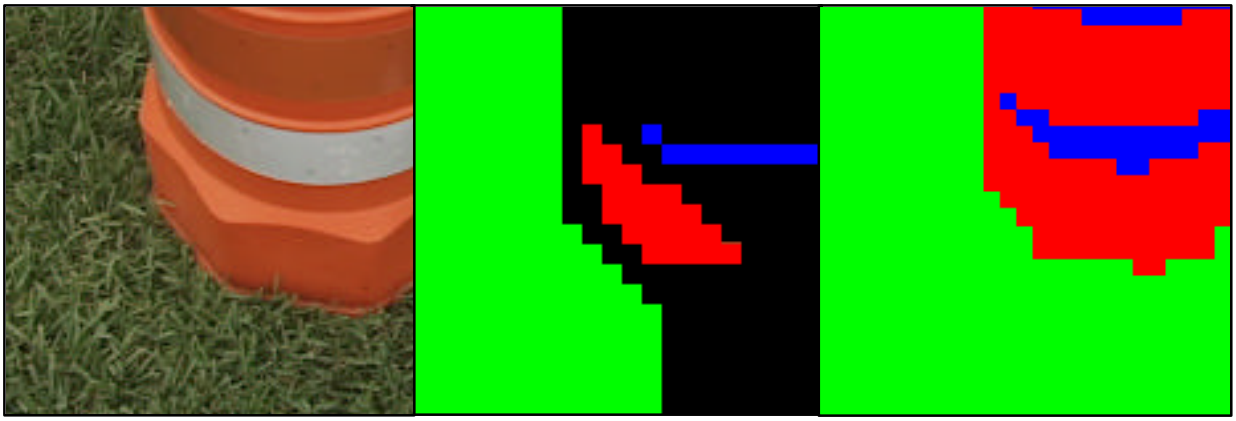
The image then must be broken up into subsections based on a size parameter that must be an integer divisor of both the width and height of the image to produce small blocks. The reason for the division is to take advantage of the similarity of localized data to help with classification. When the size parameter is very low then number of inputs presented to the Neural Network increases, but the amount of different data the blocks can overlap is minimal. However when the size parameter is high, the number of inputs is smaller and the amount of overlapping data is greater. A happy median needs to be made between those two different advantages, less overlap and less nodes. Decreasing the block size a factor of 2 increases the number of inputs a factor of 4. Additionally, the block size and overlap really cannot be larger than the smallest object being classified since that object's data could be lost in the surrounding data. Typically, the input captured by the cameras is 320x240 pixels, experimentally the block size of 8x8 seemed to work well with respect to both overlap and number of inputs. For a typical image at block size 8, the number of inputs is 1200.

The sub-blocks then need to be normalized versus the total image so they can be presented equally with blocks from other images. The theory behind the way the blocks are normalized consists of the idea that any information should be accurately estimated by using its

mean and standard deviation. This process reduces the block into a simple set of numbers that can then be presented to the ARTMAP. The mean and standard deviation of the entire image are calculated in order to properly normalize the sub blocks. Each of the sub blocks then has individual mean and standard deviation taken.

Then the entire images input pattern is pressed on each block to normalize its value between 0 and 1. To normalize the average color between 0 and 1 a division by 255 is used. The standard deviation of the block for each color is divided by four times the standard deviation of the entire image. This produces a 6 dimensional string of data for the standard deviation and average of each color channel. This data is now ready to be presented to the ARTMAP as an input.

For the ARTMAP, the input comes in as an array between 0 and 1 and a bit-mask created by the oracle of output labels that match the input blocks. The aforementioned normalization process calculates the inputs from stored images during the training phase and from the camera during the performance phase. For the training phase a good set of test images are selected that



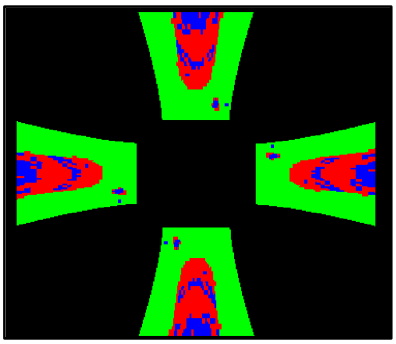
should represent all of the objects that will be classified. The images will be processed to provide the input of the input/output pair. The human oracle creates the output section. The Oracle manually reviews the test images and classifies each sub-block to its pertaining output. Those are stored in a bit-mask file representing the outputs, which are then read by the program

as the output section of the input/output pair. Those two pieces of data, input and output, are presented to the ARTMAP for the training phase until stabilized. Then those values of the hidden nodes and their connections to the output layer are saved to disk for later processing. During the performance phase, the hidden node values are read back in and only the inputs can be calculated from the incoming camera images. The output of the performance phase should return the same output labels as presented originally in the training section.

Driving System:

ARTMAP is also used to make decisions about where to drive based on the inputs. The driving system receives a series of inputs from different sensors that are generally abstracted and combined together. The output of the previous vision system needs to be placed in the local map. First the classified camera data needs to be transformed from the image projection to the flat surface. Then that image is rotated into the local map with relation to the leg angles at the hip. The ultrasonic range finders are then added as additional projected obstacles into the local map. The GPS is used as a measure of how much the vehicle wishes to go in one direction or another for the navigation challenge only, contributing two more dimensions. In the training section, the joystick provides the output section of the input/output pair for training.

The projection of the image on the ground plane is based upon the camera iris for what the cone of vision the camera has. The height affects the distance and size of the projection on the ground but not the shape. The angle versus normal affects both the shape and size of the projected image.



Once the image is projected, the image is then translated to a fixed point equal to the normal leg placement, which is done by adding the nominal leg position to the projected frame position. And finally, the image is rotated based on the leg angle into position by a pivot around that point equal to the current leg angle. The current leg angle is already tracked by the potentiometer on the leg and taking the offset versus the initial position gives the rotation angle. That information is then processed for the degree of closeness of obstacles and the context of the situation. Then the driver information from the trainer is used with the context information to generate similar driving habits to how the human Oracle drove the course.

Autonomous Challenge:

The Autonomous Challenge is accomplished using the training set of information provided by the Oracle Driver. The intelligence is created as a rather reactive system based on a series of contexts of its current situation with respect to vehicle location, direction of motion, and obstacles. The ARTMAP Neural Network then compares the situation it is in currently to the training data it was provided making a decision about where it should travel next. The human driver provides information on driving habits and obstacle avoidance during training. The actual motion the vehicle chooses is not preprogrammed but learned. However in particular situations like the Autonomous Challenge, a reactive system will not work. A secondary system needed to be implemented to backtrack to previous locations. After back tracking a new path is chosen.

With respect to the ramp, the vehicle will similarly act based on its training data. The training data it is given would be to go relatively slowly in a zigzag pattern, similar to a switch back, to prevent speed issues and other ramp issues. The vehicle has enough torque to travel up the ramp straight, but for a safety reasons the other method was chosen. Dash lines on the course are accounted for through the block system of vision processing. The lines will be recognized

and enlarged into blocks. Those blocks then fill some of the gap and provide a context to be avoided. Since the cameras can see several yards ahead the general tendency is to avoid all lines. Each leg system would work to a vector to determine that direction.

Navigation Challenge:

The Navigation Challenge provides the additional challenge of arriving at waypoints on top of obstacle avoidance. The given GPS waypoints themselves are sorted into the most appropriate order based on the minimum total distance from the current point, and are decided through a pre-processing stage. The GPS waypoints are added as weights to the neural network, then relative direction of the desired point and current location are added for both the Longitude and Latitude. Again, it is based on the training data, which shows the method of driving in the direction of the waypoint. Since the direction and obstacles are both inputs, the context of desired direction and obstacles, should relate to the training data to avoid the obstacle as the more important information.

The Navigation challenge also added particular issues surrounding legs that could decide different directions to reach the same location. A voting system to determine the most appropriate path is used. This system is added to the post-processing stage to prevent such issues. As stated before, the particular path to the waypoint is undetermined, but the main processing stage is managing the current location from the GPS and filtering system. This system also determines if a waypoint has been reached and chooses the next point. Otherwise the Neural Network based driver manages the desired vector for each leg.

Budget:

Item	Quantity	Item Cost	Total Cost
Frame	1	1500	1500
Cameras	4	30	120
GPS	1	300	300
Biscuit	1	350	350
Serial Board	1	45	45
Drive Motors	4	105	420
Steer Motors	4	120	480
Sprockets	8	10	80
Chain	2	35	70
Sonar	12	15	180
Batteries	2	135	270
Boards	3	33	99
Receiver	1	50	50
Microprocessors	4	8	32
Transmitter	2	30	60
Misc Electrical	1	100	100
Total:			4156

Conclusion:

Using the unique frame of RDB3K in combination with the ARTMAP based Neural Network classifier and driver, the vehicle will be able to perform all of the tasks required by the IGVC competition. The Omni-directional motion and vehicle's ability to reconfigure sets the RDB3K apart from traditional robots that have participated in the past.