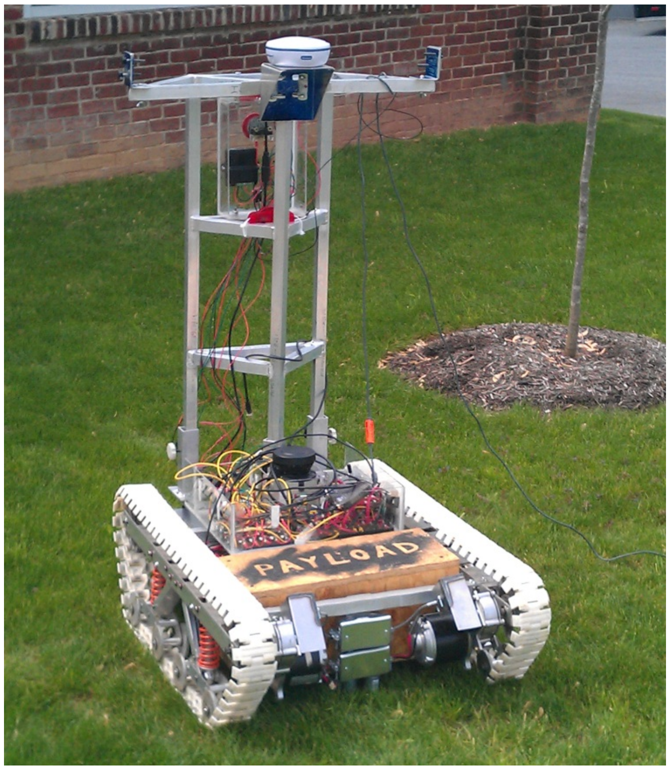
Proudly Presents:



YCP30 - Intelligent Ground Vehicle Competition 2011

<u>Team Members</u>

Richard Beasley, Ethan Benedict, Timothy Cardenuto

Matthew Heisey, Kevin Gaquin, Andrew Malachowski, Patrick McVeigh

John Powell, Matthew Small, Fernando Torres

Introduction

The senior engineering class of York College of Pennsylvania is proud to present YCP30 for the 2012 Intelligent Ground Vehicle Competition. This vehicle was designed as a capstone project by computer, electrical, and mechanical engineering students beginning in May of 2011. The 2011 vehicle from York College of Pennsylvania qualified for the competition for the first time in school history. In design for the 2012 vehicle, the team chose to keep many aspects of the vehicle similar to last year's design. The new features of our robot are tread design, the power system, navigation algorithm, and added sensor filtering. With the new adjustments to the system, the vehicle is expected to perform successfully at the 2012 competition.

Team Organization

YCP30 was designed by senior undergraduate engineering students entering their sixth semester of classes. The work done by the team was split into three main sub-teams. These teams are: structure and motion, object detection and localization, and navigation and system architecture. The structure and motion team was responsible for designing the body, frame, and ground contact for the vehicle. It was also responsible for the electrical and drive systems. The object detection and localization team is responsible for the sensors on the vehicle and mapping the data received by those sensors. Their integration with the navigation algorithm is essential to the functionality of YCP30. The navigation and system architecture team is responsible for the decision making of the vehicle. This sub-team uses the sensor input to determine a heading and velocity for the trajectory of the vehicle. Figure 1 below shows the engineering disciplines involved with each sub-team.

Faculty Advisors		
Dr. Gregory Link and Dr. Patrick Martin		
Project Team Leader		
John Powell (EE)		
Vision, Object Detection, and Localization	Decision Making and System Architecture Team	Structure and Motion Team
Tim Cardenuto (EE)	Kevin Gaquin (CE)	Matthew Small (EE)
Patrick McVeigh (EE)	Andrew Malachowski (CE)	Matthew Heisey (ME)
Richard Beasley (CE)		Fernando Torres (ME)
		Ethan Benedict (ME)
		John Powell (EE)

Table 1 – Team Organization

Design Innovations

In designing YCP30, many features are similar to the design of last year's vehicle. There are however a few areas that have been focused on and are important in allowing our robot to perform successfully this year.

1. New tread design for ease in turning and added static friction
2. Weight reduction for quicker turning and responsiveness to commands
3. New onboard computer

4. Dual webcams for increased viewing angle

To improve the vehicle's maneuverability the treads were to be redesigned to have a higher coefficient of friction, add grip to the treads to prevent them from slipping, and turn better on uneven terrain. The first test performed was a friction test to find the static and kinetic coefficients of friction of last year's robot and the potential new treads. Two new treads were designed and tested comparing the values to the treads from last year.

One of the major mechanical differences from last year's vehicle to this one is in weight reduction. The robot last year weighed roughly 200 pounds. This weight came from a robustly designed frame and oversized batteries. This year, our team has made it a goal to build a robot that weighs less than 160 pounds. The reduction in weight will help with quicker turning and responsiveness to commands. With lower weight, the motors will have to work less to move the vehicle. The less the motors have to work, the less power is used.

The vehicle this year has an onboard desktop computer. This feature is powered by an onboard lithium polymer battery. From our calculations, this lithium polymer will allow the computer to run for one hour before dying. In order to charge this battery fully, the battery must be charging for roughly three hours. While the vehicle will only be on the course for ten minutes at a time, the computer will be running for long periods of time for debugging purposes. The computer cannot be powered all day by the lithium polymer battery. An umbilical system was designed in order to allow for switching to an AC/DC converter during debugging. This umbilical allows for the power of the lithium polymer to be conserved while not shutting the computer down.

Last year, the vehicle was implemented with one camera mounted at the top of the vehicle. This year, the team has implemented two cameras mounted on wings at the top of the camera tower. This will allow for increased viewing angle and ability to see the edges of the robot. Seeing the edges of the robot will allow the vehicle to see obstacles that are close to it and avoid them.

Design Specifications and Vehicle Performance Goals

Physically, the vehicle this year is designed to be at minimum dimensions as specified by the competition. The design was intended to make navigation easier as well as reduce weight. The sizing leads to lower power consumption and faster vehicle response time. The vehicle mechanical specifications are listed in table 2 below. The performance of the vehicle also has team set specifications. These specifications are based on sensor performance as well as power restrictions. The specifications are listed in table 3 below.

Weight	167 pounds including 20 pound payload
Width	25.5 inches
Length	36.5 inches
Height	50.5 inches

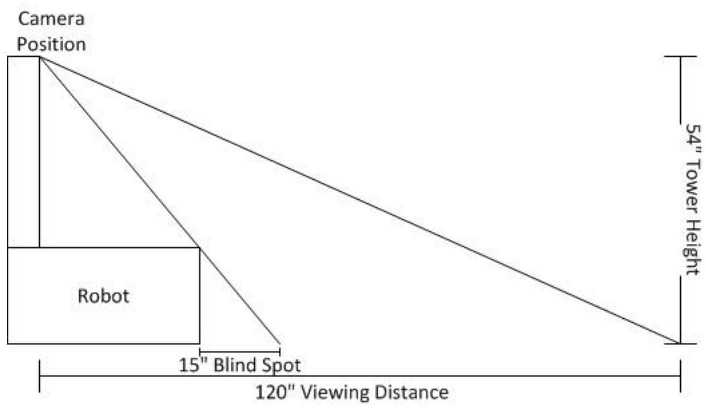
Table 2 – Physical Characteristics

Maximum Speed	10 miles per hour
System Response Time	115ms
Battery Life	One hour run time
Obstacle Detection Range	10 feet
Accuracy of GPS Waypoints	1 meter

Table 3 – Performance Statistics

Vision System

Seen below in figures 1 and 2, is the viewing angle of the cameras on the vehicle. After observing numerous other teams' line tracking failures last year due to close proximity lines running parallel to the vehicle's direction of travel, as well as observing this failure mode in our own tests, we decided to adjust our vision system to utilize a two camera setup. This setup has cameras mounted perpendicular to the tracks on either side of the robot. The cameras are placed 30" from the front of the robot at a height of 54". This allows for detecting lines parallel to the tracks that could pose a hazard while turning. It was determined that the robot should be able to both view a line and react to it turning at full speed no farther than 1 ft. away. If the robot was turning and approaching a line it should turn no farther than 6 in before halting the turn. The turning speed of the robot is calculated based on a latency of 250ms from the time the image is first captured until the motors begin to physically respond to the detection. Using these numbers the robot can turn no faster than 24" a second. This gives an angular velocity of 14.56 degrees per second.



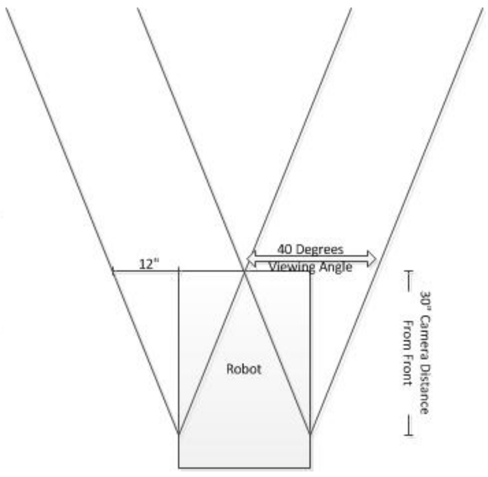


Figure 1 - Side view of the robot plane of view; Figure 2. Viewable area of Robot from top

Image processing is performed via EMGU open CV wrapper for c#. Once an image is captured, it is necessary to convert said image into a useful data set consisting of distances in a three dimensional plane. This conversion is done by utilizing a pre-calculated array that contains the actual distances, in inches, of every pixel the camera captures. This array is generated using the homography, perspective transform, and chessboard detection features available in open cv. Pre-calculating our transform array is a performance improvement over the built-in transforms functions, which require re-computation each pass. The chessboard is placed within the field of view of the camera. The distance of the board, along with its physical dimensions, are entered into the calibration program and world space points are generated and saved into a file.

Once the robots line detection algorithm captures an image, it is converted into hue, lightness, and HLS format. The lightness values are checked against a predetermined threshold value; any pixels within this range are saved into an array of camera detection objects and sent to the world map for path planning.

Object Detection

The SICK LMS111-10100 LIDAR was used for object detection by returning object distance and angle from the robot heading. The LIDAR takes measurements every .5° over a 270° spread returning 541 detections each time data is queried. These detections are grouped into objects based on inter-detection distances to reduce the number of points stored in mapping. The absolute x and y position of each object is then calculated based on the offset from the absolute robot position at the time of measurement. Finally, occlusion is performed on camera objects that are determined to be artifacts of LIDAR detected objects. The LIDAR and camera objects are then combined in a world map used by navigation.

System Architecture

Operating the vehicle requires the integration of the sensing network with the navigation and image processing algorithms. In order to facilitate integration the task parallel library (TPL) introduced in .NET 4.5 was used to create a custom architecture similar to ROS. Each block found in figure 3 found below represents an actor within the system. The sensing network consist of a Phidget 1055 Spatial, two US Digital E6 encoders, a Hemisphere A100, a Sick LMS111 LIDAR, and two Logitech C310 webcams producing data at 24 ms, 115 ms,50 ms, 20 ms, and 125 ms periods respectively. The Phidget and encoders pass data to a spatial filter to determine the vehicle's spatial location, while the webcam and LIDAR data are passed asynchronously to their respective filtering blocks. The filtered data is then transformed into to match the vehicles current orientation and combined into a construct aptly called a Scene Graph which the navigation algorithm utilizes for obstacle and line avoidance, ultimately passing commands to the motor controller.

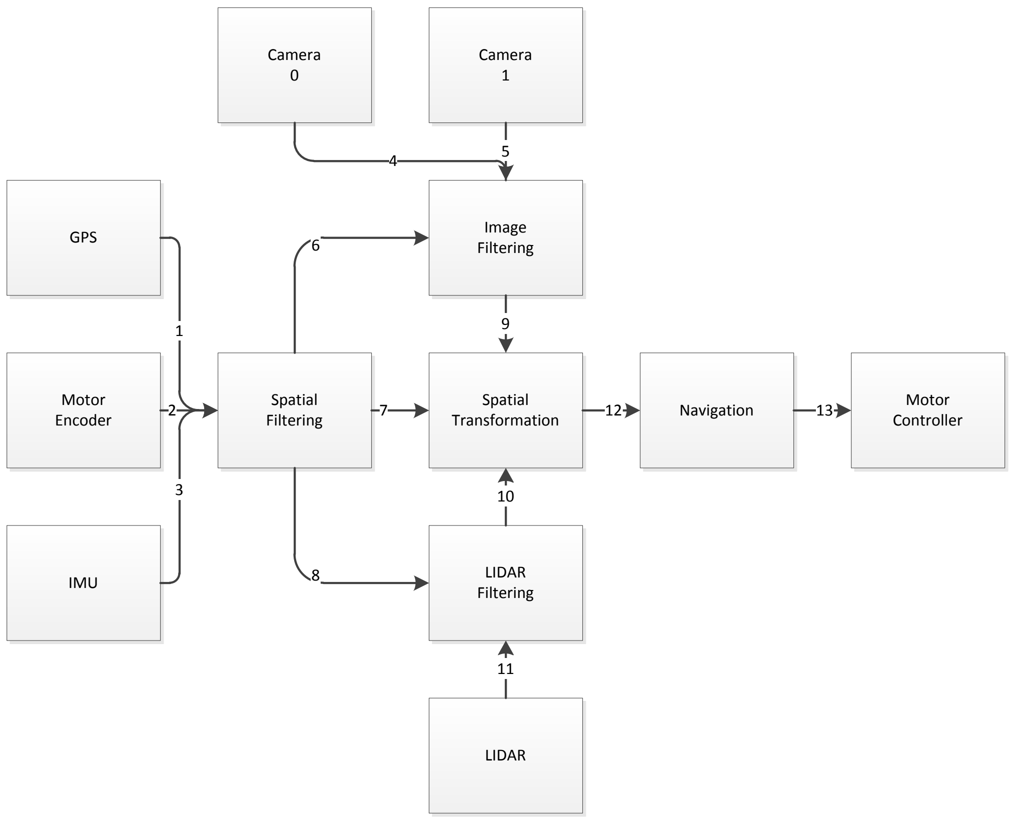


Figure 3 – System Architecture

Navigation & Mapping

After competing last year with a robot which implemented the A* algorithm, this year our team wanted an algorithm which could performs as well as A* however, takes a safer path, rather than a shorter one. After researching various algorithms, our team decided that utilizing a Potential Fields Algorithm, could give us a safer path which was more likely to not hug lines, as A* did. However, since the Potential Fields Algorithm is now being implemented, a Cartesian based map no longer efficiently fulfills the mapping task.

The reason a Cartesian based map no longer solves the task of mapping efficiently, is the Potential Field Algorithm only needs to know how far an object is away and the angle to the object. So while a Cartesian based map could be used, since it is no longer necessary to have each node of the map represented, as is needed with an A* algorithm, it is only necessary to store each object's location. Therefore, a simple list can be implemented which contains the necessary data. In this list, which we call a Scene Graph, there are actually three different sub-lists, as well as the robot's location. Each of the three sub-lists all contain similar data, the coordinate of the object based upon the initial robot's coordinates, the time when the object was found, how long the object has to live, and the other fields shown below in which are necessary for the Potential Fields Algorithm. Within the mapping a Scene Graph exists that contains data which is still important to the robot. As the data in the Scene Graph becomes old and has existed for longer than its time to die allows, it is then removed. Note if an object in the current Scene Graph was detected again, its time detected gets updated so that it will not be removed and placed in the dead Scene Graph.

To navigate based upon this Scene Graph, the Potential Fields Algorithm will be used [1]. The Potential Fields Algorithm works based off of two simple equations seen below in Equation 1 and 2. These Equations result in a Δx and Δy, which can then be used to generate an new vector, rather than a path. The variables associated with the following equations are as follows: $\beta = \text{constant} > 0$, $\alpha = \text{constant} > 0$, $\text{d} = \text{distance from robot to goal}$, $\text{r} = \text{radius of the goal}$, $\text{s} = \text{spread of the goal}$, $\theta = \text{angle of robot heading to goal}$.

Attractor

$$\Delta x_{att} = \begin{cases} 0, & d < r \\ \alpha(d-r)\cos(\theta), & \mathrm{r} \leq \mathrm{d} \leq \mathrm{s}+\mathrm{r} \\ \alpha s\cos(\theta), & d > s + r \end{cases} \qquad \Delta y_{att} = \begin{cases} 0, & d < r \\ \alpha(d-r)\sin(\theta), & \mathrm{r} \leq \mathrm{d} \leq \mathrm{s}+\mathrm{r} \\ \alpha s\sin(\theta), & d > s + r \end{cases}$$

Equation 1 - Equation for an attractor in the Potential Fields Algorithm

Repulsor

$$\Delta x_{rep} = \begin{cases} -sign(\cos(\theta))\infty, & d < r \\ -\beta(s+r-d)\cos(\theta), & r \leq \mathrm{d} \leq \mathrm{s}+\mathrm{r} \\ 0, & d > s + r \end{cases} \qquad \Delta y_{rep} = \begin{cases} -sign(\sin(\theta))\infty, & d < r \\ -\beta(s+r-d)\sin(\theta), & r \leq \mathrm{d} \leq \mathrm{s}+\mathrm{r} \\ 0, & d > s + r \end{cases}$$

Equation 2 - Equation for a repulsor in the Potential Fields Algorithm

By combining the two vectors a single vector can be created which will "pull" the robot to the GPS waypoint along the safest path, from the current robot's location. Figure 4 shows the results of two equations and their effects, both on each other and on their environment.

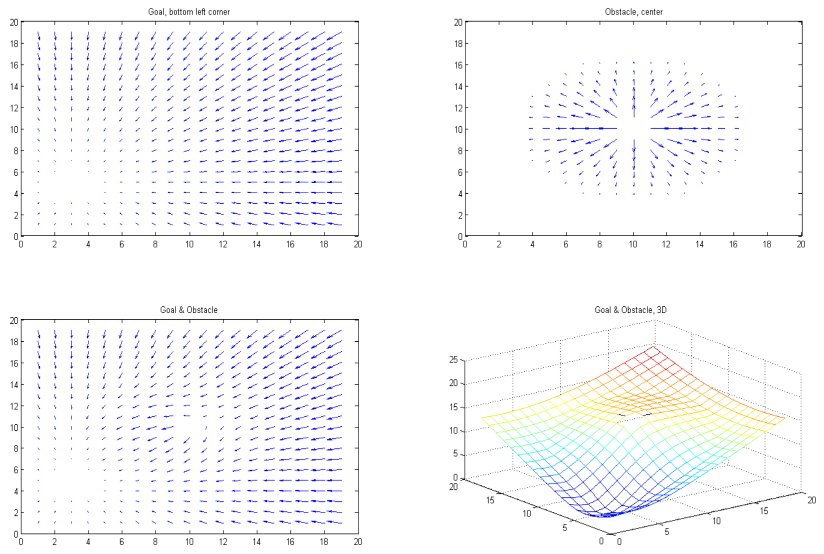


Figure 4 - Graphical representation of a Goal and obstacle in a single map.

(Top left) An attractor (GPS Waypoint) placed in the bottom left corner. (Top right) A repulsor (obstacle) placed in the center. (Bottom Left) The combination of the goal an obstacle in a single map. (Bottom right) A graphical representation of the valley created by the GPS waypoint, and the hill created by the obstacle.

To then utilize this algorithm with multiple obstacles, all that is necessary is to perform the summation on all the obstacle's resulting repulsion vectors. When repeated as the robot moves, it results produce a smooth, safe, path, an example of this can be seen in Figure 5. This simulation was performed utilizing an engine we designed and built in C# to allow for testing of any scenarios which may present themselves at the competition.

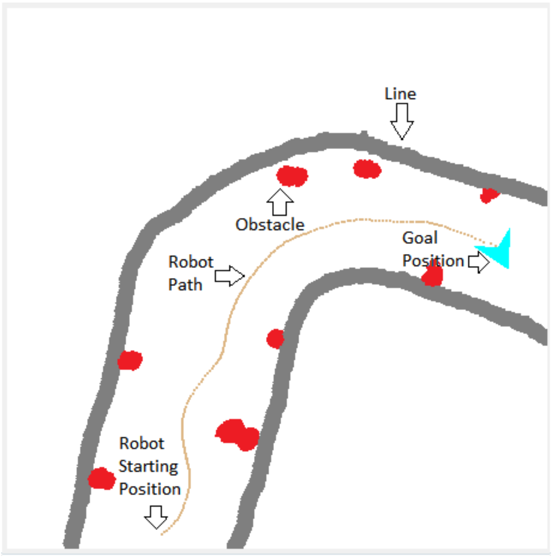


Figure 5: Test Course utilizing the Potential Fields Algorithm.

However, by using the Potential Fields Algorithm, situations can exist in which the vector generated has magnitude of zero, i.e. the robot is stuck in a local minima. Therefore, to solve this issue a different path planning algorithm must be used to get out of the minima generated. After researching various algorithms, the conclusion was made to utilize an algorithm similar to A*. But A* is based upon Graph Theory which requires vertices and edges, which while they exist in a Cartesian map, they do not exist within a Scene Graph. Therefore an adaption on A* was created; this adaption, which we have called G*, relies on creating the vertices and edges is needed. A vertex is created by expanding out from a previous vertex the move distance, the desired distance between any two adjacent vertices. The edge is created by utilizing the distance-plus-cost heuristic generated when a new vertex is created. The distance-plus-cost is based upon the number of obstacles within the given vertex, and if it is a turn or not. Other than the generation of vertices and edges, G* runs exactly as A* would. It still returns path, that path is then used on top of the Potential Fields Algorithm. To do this the first vertex outside of the robot's width is added to the Scene Graph as an attractor, then the Potential Fields algorithm runs normally. However, utilizing the G* vertex rather than and GPS waypoint as its attractor. This same process is then repeated each time the navigation algorithm is called, while a shortest path algorithm is still being utilized the resulting path resembles that of a safer path rather than the shortest path.

Waypoint Navigation

An A100 Smart Antenna GPS is utilized to track the robots position in latitude and longitude coordinates. This sensors measurements are necessary to calculate the great circle distance from the robot to the given competition GPS waypoints, which is performed using the Haversine formula where is the goal waypoint and is the current robot location.

The GPS is also used to assist the encoders with calculating changes in x, y position by using the same formula. This GPS has an accuracy of within .6 meters at a 95% confidence which will ensure the robot can navigate to within 1 meter of waypoints.

Position Estimation

This year, the 2012 team has worked to implement a Kalman Filter estimation algorithm. This version was used over the original Kalman Filter due to the non-linear nature of robotic movement. The measurement data passed to the filter is the position coordinates as measured by the sensors. A Kalman gain is then calculated based on the variances of the measurements and process. The innovation is then estimated using the weighted difference between the measurement and the previously predicted state. This innovation is added to the predicted state to get the current state.

Finally the next state and process variance is calculated using the navigation input and the jacobians of the odometry model. The EKF process described as three steps of Measure, Estimate, and Predict are looped and will maintain an accurate estimate of the current state based on control input and encoder data.

Electrical System

Computer Power

In the design of the electrical system, the powering of the computer was the first step. In year's past, the team has used an on board laptop as the decision making center of the vehicle. This year, the team determined that designing an on board desktop would be best for computational speed. In order to power this portion of the system, a 25.9V, 12.6Ah lithium polymer battery was selected. It is designed to be able to sustain power to the computer and sensors for up to an hour. The batteries for the motor system are rate at 2-12V, 20Ah. These batteries are also estimated to deliver a run time of an hour.

Umbilical Design

The computer will be run all day when at the competition. As a result, the computer battery will die quite quickly if another source of power is not provided. In order to allow for full computer operation while the lithium polymer battery charges, an umbilical power supply was designed. The umbilical consists of an

AC/DC power converter that is plugged in to the computer to power it. It is not convenient however to disconnect the lithium polymer battery and connect the alternate power source as this would require shutting down and restarting the onboard computer, disrupting work and reducing the rate at which testing and improvements can be made. In order to make this switch without any disconnections or power shut down, a procedure has been put in place to help ease the situation. While the computer is running, the operator may plug in the alternative power supply. At this moment, the lithium polymer and the AC/DC power supply are both powering the computer. In order to do this safely, there is diode protection on both power supplies. Without this diode protection, the power source with the higher voltage would attempt to charge the other power source. The diodes prevent this issue. Once both power sources are connected, a switch may be flipped to disconnect the lithium polymer from the computer. Power is not lost to the computer and now the battery may be charged. After the switch is used, the battery is now connected to an outlet on the vehicle where the charger can be attached. This umbilical is shown below in figure 6.

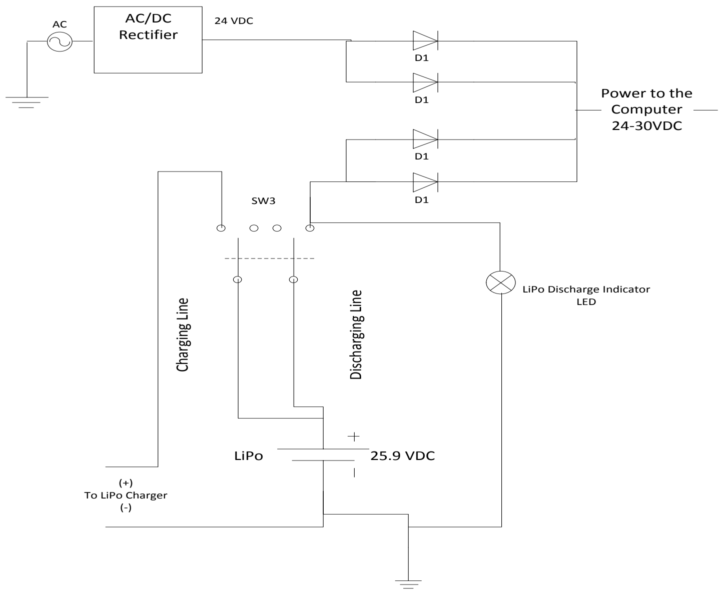


Figure 6 – Umbilical Design

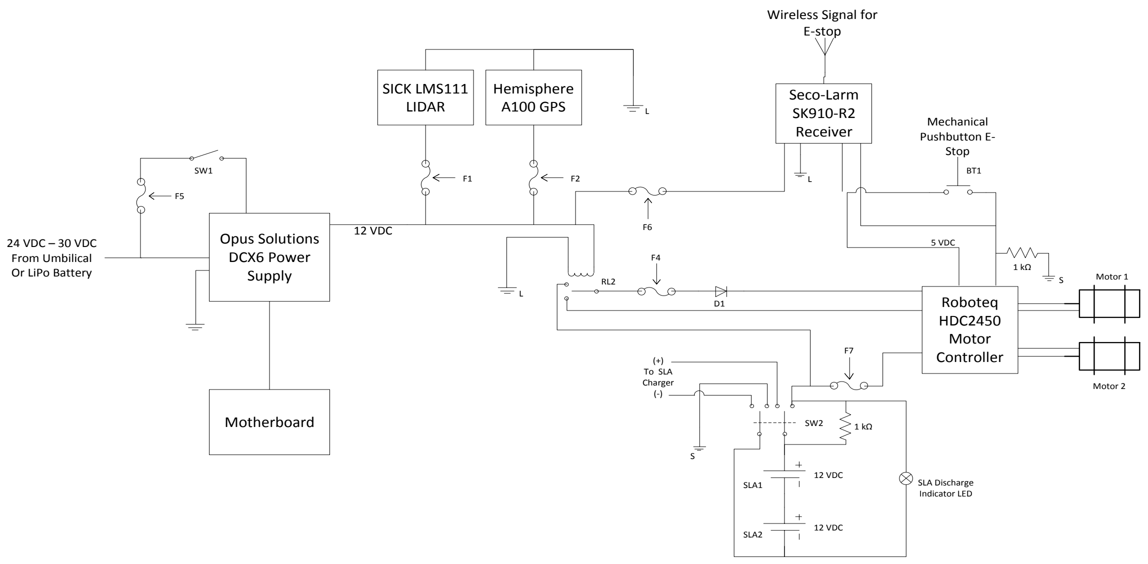

Wireless Signal for E-stop
Seco-Larm SK910-R2 Receiver
SICK LMS111 LIDAR
Hemisphere A100 GPS
Mechanical Pushbutton E-Stop
BT1
SW1
F5
F1
F2
F6
F4
F7
RL2
D1
L
S
12 VDC
5 VDC
1 kΩ
24 VDC – 30 VDC From Umbilical Or LiPo Battery
Opus Solutions DCX6 Power Supply
Motherboard
Roboteq HDC2450 Motor Controller
Motor 1
Motor 2
(+) To SLA Charger (-)
SW2
SLA1
SLA2
SLA Discharge Indicator LED

Figure 7 - Power System on Vehicle

Mechanical Design

Camera Tower

The camera tower had to be redesigned this year in order to properly support the two cameras that our robot design requires. A framework of aluminum members had given previous IGVC team's great success in maximizing rigidity while minimizing weight, and so the idea was reused for this year. The resonant frequency for the camera tower was discovered during analysis, and resonance was determined to occur outside the operating range of the robot.

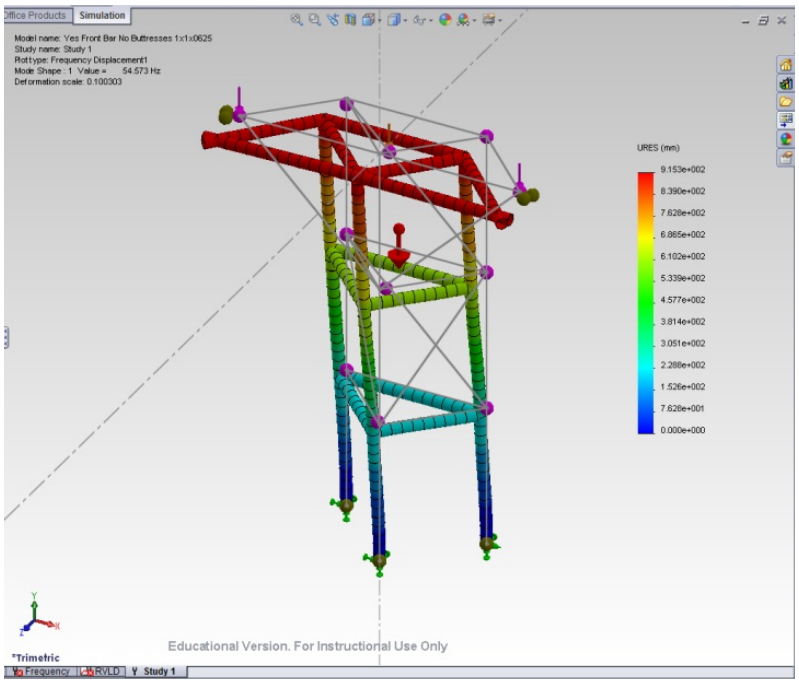

Model name: Yes Front Bar No Buttresses 1x1x0625
Study name: Study 1
Plot type: Frequency Displacement1
Mode Shape : 1 Value = 54.573 Hz
Deformation scale: 0.100303

Figure 8: Deflection at the first resonant frequency of the tower ($\omega_n = 54.573$ Hz)

Tread Design:

This year, the team decided that it is necessary to redesign the treads on the vehicle. This desire came in an effort to minimize slip and help ease turning. In order to print these treads, a 3D printer was used to print ABS plastic. Two new designs were printed using this styrene ABS plastic. With these two designs, analysis was done to see which choice was best for the vehicle. The two designs used were a gap tread design and a v tread design. These new designs were analyzed for strength and coefficients of friction. The static coefficient of friction of the "V" tread design was too close to 1.0; therefore motors will require more power to start moving and turning. Also as a result, it will be harder to maneuver, which affects the vehicles performance. The gap tread design had a lower coefficient of friction than the v tread. It did however have a higher coefficient of friction than last year's treads which will allow for better maneuvering and less slip. Also in an effort to increase maneuverability, this year wings were added on the side of the treads. This was done so that the robot can avoid getting stuck in gaps on holes in the ground while performing turns.

The 3D printer used to print these treads has the ability to print ABS plastic in layers, making it possible to print specimens in different directions at different strengths. A material test was performed to determine what direction was the strongest under tension. The material test was performed using dog bones printed in four different directions. The weakest ultimate stress occurs when the direction used is standing vertical (1860 psi), from the FEA the maximum stress is 1530 psi, validating ABS plastic as the material to be used. A tear out shear stress test was also performed to validate the treads.

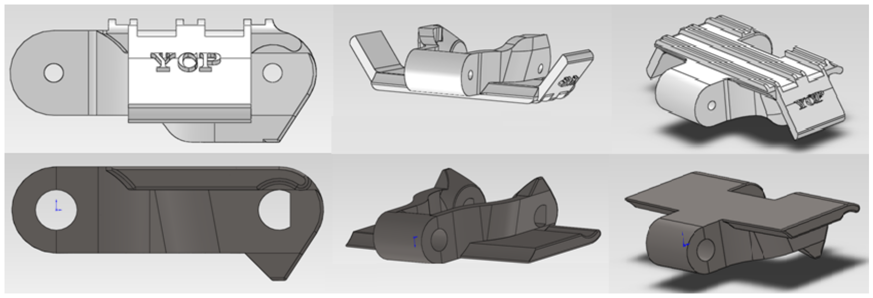


Figure 9: New tread design (top) and old tread design (bottom)

Tread design	Static coefficient of Friction	Kinetic coefficient of Friction
Gap tread (new design)	0.84	0.65
V tread (new design)	0.95	0.71
Rexnord tread (old tread)	0.55	0.5

Table 4 – Friction coefficients of different tread designs

A new pin was designed in order to ease stress concentration in the treads. The new pin design consisted of a 1/8 inch steel rod that is not press fitted into the treads. The pins are secured by two e-clips to prevent them from sliding out. The new design reduces shear stress by 47% (from 687 psi to 365 psi) at the point of interest where the pin is attached, the section that is most likely to break. The following figures show the finite element analysis performed to treads.

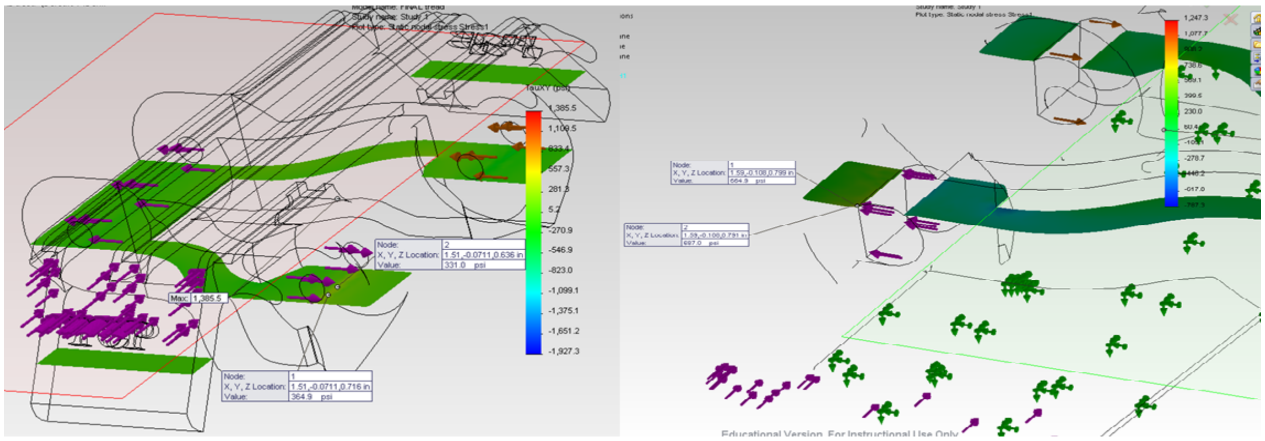


Figure 10: Finite element analysis of gap tread design (left) and old tread design (right)

Frame Analysis

Due to the overdesign of last year's frame, the 2012 Structure and Motion Sub-team decided that a material and cross-section size analysis was needed to confirm the best material for the 2012 robot frame. In order to analyze various materials, a unit needed to be established that would be able to sort through all the options. This unit would include the properties wanted on the numerator and the properties that where unwanted on denominator.

When designing a frame, flexural rigidity is the key aspect. Flexural rigidity is how rigid the structure is; therefore, it is critical in designing a frame. In order for the suspension to be more effective, a frame must be rigid. Flexural rigidity is given by *EI*, where *E* is the Modulus of Elasticity and *I* is the moment of inertia. Various materials were analyzed for the frame and were narrowed down using this unit. Table 1 shows the materials that were selected to be analyzed.

Tread beam

Materials	Cross-section	Cost per Foot ($)	Weight (lb)	EI (Kip-in^2)	EI/WC (Kip-in^2/lb($/ft))
Aluminum 6061- T6	2" X 1.5" x.125"thick Rectangular Tube	5.33	2.588	2804	**203**
Aluminum 6063-T52	2" X 1.5" x.125"thick Rectangular Tube	2.01	2.561	2777	**216**
A513 Steel	2" X 1.5" x.083"thick Rectangular Tube	4.93	5.108	5931	**235**

Crossbeam

Materials	Cross-section	Cost per Foot ($)	Weight (lb)	EI (Kip-in^2)	EI/WC (Kip-in^2/lb($/ft))
Aluminum 6063-T52	1" square tube x .065" thick	3.64	1.379	345	**327**
HR Mild Steel A36	1" square tube x .065" thick	2.07	2.244	1032	**222**
A513 Steel	1" square tube x .065" thick	1.99	2.244	1057	**237**

Table 5 – Materials selected to be analyzed for frame use

6061- T6 aluminum was determined to be the best material to use for the frame. Finite Element Analysis (FEA) was used to verify that both that this material would be strong enough for the frame. The tread beam was loaded with 1/6 of the robots total weight distributed at the point where the three crossbeams would be attached. Figure 2 shows the FEA performed on the tread beam, the maximum stress seen was 1220 psi.

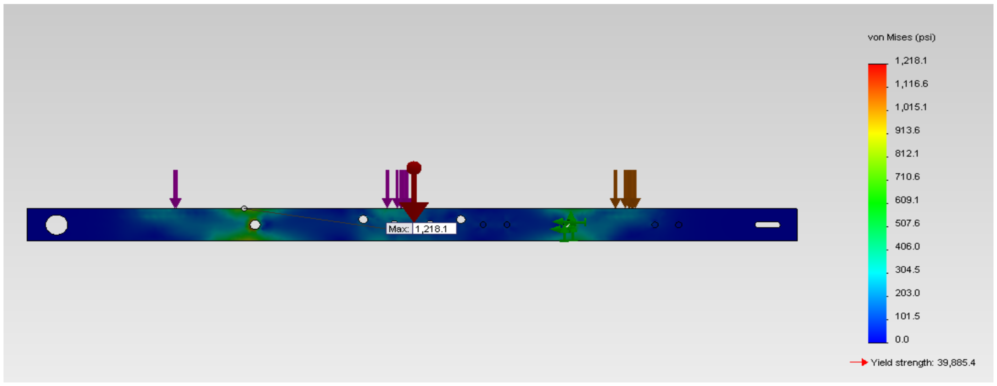


Figure 11: FEA of treadbeam

The design shown in Figure 1 was decided upon for various reasons. First, it satisfies the constraint that the Lidar needs to be placed so that it can see the back corners of the robot. Additionally, the design keeps the motors away from the CPU enclosure; therefore, motor noise is not an issue for the CPU. Also, the robots center of mass is only .08 in. from the geometric center towards the front of the robot, allowing the suspension to function as designed. Furthermore, the robots center of gravity is approximately 8.5 in. from the ground. This is an improvement of approximately 1.5 in. from last year's design. The weight of this design turned out to be 150lbs; however, while there are holes that show where bolts need to be added, the bolts themselves are not added. These bolts will add weight which will take the robot slightly over the 150lbs design limit.

Budget for YCP30

To keep the cost of our project down, several components used on this year's vehicle have been recycled from the vehicle entered last year. The LIDAR, GPS, motors and encoders were reused from last year to this one.

Item	Price	Quantity	Total
Suspension	$ 143.24	1	$143.24
Frame	$ 171.28	1	$171.28
Camera Tower	$ 71.29	1	$71.29
Camera Mounts	$ 96.00	1	$96.00
Treads	$ 1,100.00	1	$1,100.00
LIDAR	$ 5,000.00	1	$5,000.00
GPS	$ 2,000.00	1	$2,000.00
Webcams	$ 35.00	2	$70.00
Batteries/ Chargers	$ 531.85	1	$531.85
AC/DC Converter	$ 290.16	1	$290.16
Electrical Components	$ 288.00	1	$288.00
Motors	$ 85.00	2	$170.00
Encoders	$ 85.10	2	$170.20
Computer	$ 1,045.94	1	$1,045.94
Motor Controller	$ 675.00	1	$675.00
Total			$11,822.96

Table 6 – budget for 2012 Vehicle

The ten students on the design team for YCP30 have spent two semesters working on designing, manufacturing, assembling, and testing the vehicle. Each student has spent more than 500 hours on this project. These 500 hours include everything from researching part availabilities and prices, spending time in team meetings, and constructing the robot. In total, the project has been worked on for over 5000 hours.

References

[1] M. Goodrich, "Potential Fields Tutorial", Brigham Young University, Computer Science.