Software systems report:

The robotic platform developed by our team is capable of navigating an environment using the sense –think-act paradigm as its driving force. The system iteratively gathers data about its environment, as well as its place in that environment, then it processes the data, determining the action most optimal in achieving its goal, then it acts to complete that action. The general implementation of this paradigm involved performing every process of sense-think-act in parallel. Using this method proved more advantageous since many robotic applications involve small and mobile computational architecture, relatively lengthy computation times, slow sensor refresh rates, and mission-critical safety requirements.

This implementation was done in the LabVIEW programming environment: because LabVIEW offers robust hardware abstraction capabilities, built in functionalities for robotics, and a simpler, easy-to-learn interface for new users it was chosen to be our implementation software.

The architecture of our software uses a top-level module to condense and provide structure for our entire system; it is responsible for startup initialization, continuous operation, and shutdown procedures. Upon startup, this module first initializes all the sensor acquisition loops, such as those of the GPS, LIDAR, and IMU. Due to the LabVIEW’s simple to use multi-threading capabilities, every sensor loop runs independently and continuously while acquiring and then sharing sensor data through global variables, improving our system’s integration. This allows for the acquisition rates demanded of each sensor to be tailored to that sensor, rather than having every sensor operate at the same rate. For instance, the GPS updates once every second, while the LIDAR data updates at 500 Hz. The different acquisition rates allow each sensor to report at its maximum rate, giving us the most up-to-date environment model possible. All of the sensor interfaces communicate with a computer through virtual serial RS-232 ports emulated by USB adapters. These ports are opened and configured using LabVIEW's VISA tool.



The top-level module is also responsible for implementing the sense-think-act scheme mentioned earlier.

“sense” loop logic

This is the loop responsible for gathering data from which a model of the robot’s environment can be constructed. Each sensor has its own operational loop, each of which is running parallel:

* The LIDAR continuously scans the environment and then reports back with the polar representation of the angle magnitude of object surfaces that intersect the LIDAR's plane of sight.
* The GPS sensor loop acquires a longitude and latitude value for the robot’s present position, along with additional information such as altitude and absolute time. Using a known location approximation, we can improve the startup time for the GPS by setting a close estimate of the GPS's physical location.
* The IMU uses a combination of accelerometers, magnetic Hall Effect sensors, and Kalman filters to continuously report its own orientation in space with respect to the Earth's magnetic field. We use this as a magnetic North for our compass in order to guide our robot to a command heading.

The data gathered in the “sense” stage is used to update global variables, setting the stage for the think loop.

May 8, 2012

“think” loop logic

This loop takes it the current sensor data along with the current robot set mode that defines the robot's behavior, which include a “tele-operated” mode for remote operation with external Wii mote or any other handheld remote and an autonomous mode for intelligent course navigation, and uses them to generate an world model, which it then uses to determine the optimal action based on its present goal.

Part of our localization software takes in the LIDAR data when operating, an array of line vectors, containing start and end points, generated by the vision loop, which delimit the white course lines, and superimpose the line vectors on top of the LIDAR distance readings to create a new obstacle histogram. Then, using the current GPS location and the location of the next GPS waypoint, we use spherical coordinates to calculate the current distance and directional bearing to that waypoint. Using the obstacle histogram and desired bearing, we constructed a simple bearing controlled algorithm that attempts to align the robot towards a suitable opening within the histogram field. An opening is determined “suitable” when the distance and angle of the opening produce an absolute opening wider than the width of the robot’s wheelbase plus tolerances. While searching for suitable openings the robot implements a cost function based on its current difference in sub-goal heading and current heading along with a current distance. If the robot veers too far from the sub-goal heading or the current distance exceeds the specified maximum tolerance, the robot will find alternative paths that might provide a more direct path.

To elaborate on the vision loop, LabVIEW’s vision module has a feature that can collect images from the Logitech webcam located on our robot’s mast, which allows the system to quickly process them with our algorithm isolating the white lines. Although the device is capable of higher resolutions, we collect each image at 640x480 pixels to decrease processing time for the images.

By preforming a perspective transform on each image, we are able to calculate the distances to the white lines. This is possible because we used, through a tool built into LabVIEW, a grid with known distance measurements to generate the transformation. When this transformation is applied to an image on the course, the image appears to be viewed from straight above, and distances on the ground can be determined. A color threshold is placed on the image to find the white lines. These areas are then dilated and eroded to remove specks while also filling up gaps in the line. A line detection algorithm then finds the lines. These lines are returned in real world coordinates, due to the perspective transformation.



Now once an image is acquired it can be processed so that an array containing a representation of each of the multiple white course lines can be easily transposed to our mapping algorithm.

“act” loop logic

Finally, the “act” phase is when the robot uses the information determined from the gathered data to perform some action which moves the robot towards its goal. In this loop the motor control loops are initialized. From the command bearing and command velocity specified by the “think” module, the control loop attempts to drive the robot in the set direction. We calculate the specific motor velocities or RPM values by defining the drivetrain model matching our robot's own differential drivetrain. All relevant physical dimensions are specified within the model such as wheel radius, wheelbase width, and gear ratios along with clockwise and counterclockwise motor orientation. Using the system controller and implementing an integrator feedback loop, we take our command heading, which serves as a set point, and our current heading from the IMU, which serves as the control system output, and, using a basic integrator method, we adjust the robots drivetrain to orientate the robot towards its command heading.