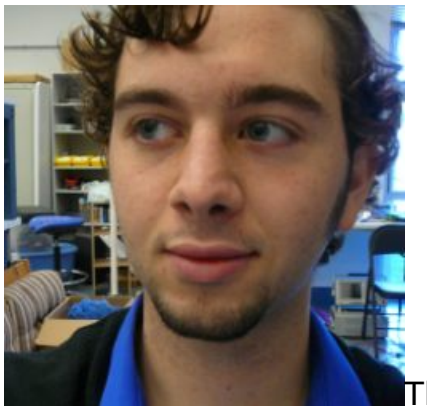
Creating an Autonomous Robot for the IGVC Autonomous Challenge

Gray Thomas - Olin College of Engineering

 The Olin IGVC team is an academic student group from Olin College in Needham, MA. We designed our robot, Athena, to compete in the IGVC autonomous challenge. The IGVC challenges participating robots to traverse a grassy path defined by simple spray-painted lines and filled with obstacles such as traffic barrels and buckets.

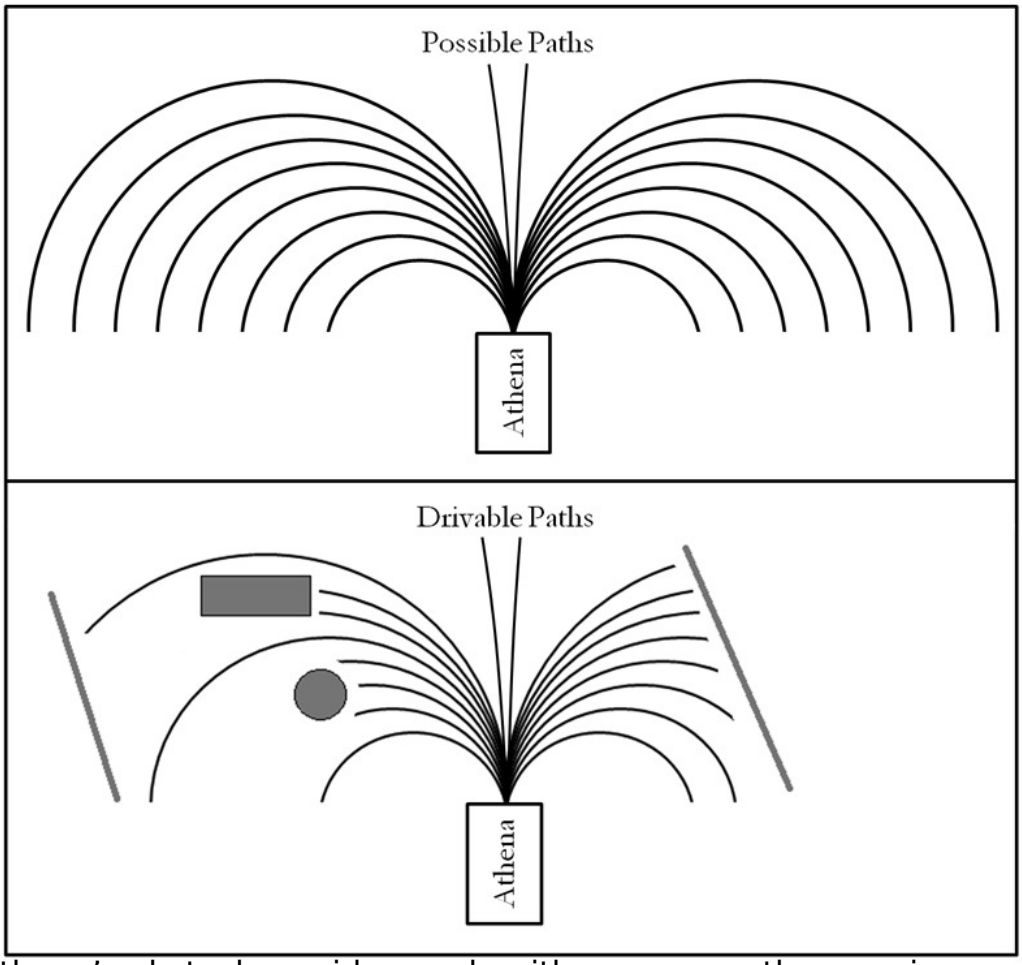
Performing well in this competition tests both robot vision and cognition speed because an inability to safely move faster than they can avoid obstacles frequently slows down robots. Previously, we used NI LabVIEW [1] software on two desktop computers in the back of the robot to read the input from our SICK LIDAR and control the steering and drive motors, but this year we switched to an FPGA-based solution using the NI sbRIO-9642 [2]. We had seven programmers on our team, but only one member was experienced with FPGAs. However, thanks to the LabVIEW FPGA Module, [3] we picked it up much faster than we would have if we were forced to learn VHDL or Verilog.

Athena is a modified standup electric utility vehicle consisting of a motorized tricycle with front wheel drive and steering. She has a rotating SICK LIDAR on a four-bar linkage with a motor for her main obstacle avoidance sensor and a fisheye lens-equipped camera on her mast to detect lines. She is powered by two large, deep discharge batteries and power is carefully regulated to protect the more delicate components from the motors. During the competition, some of our custom level shifter circuits exploded and blew out some of the NI Single-Board RIO [4] components. This was a major setback, but an NI engineer was willing to help us on a weekend to identify the blown components and assemble a workaround.

Computing

Athena's brain consists of one laptop computer connected via a CAT5 cable to an NI sbRIO-9642, which includes a real-time processor and a Xilinx Spartan-3 FPGA with two million gates. Our team chose to explore and extend a subsumption architecture purposed by Rodney Brooks [5]. The idea of using a subsumption architecture with an FPGA seems to be relatively novel in the greater robotics community. Because we used subsumption, our design process was a tightly coupled development and test cycle with little to no vehicle simulation. One of our goals this year was to increase cognition speed so the average effective operation velocity could exceed two miles per hour. Therefore, we worked very hard to ensure that our obstacle avoidance sensor could update fast enough to support such a speed. By using the parallel processing capabilities of an FPGA, Athena processes LIDAR data scans at 70 Hz whereas last year's PC-based robot vehicle, Brian, could only process scans at 10 Hz.

Obstacle Avoidance Algorithms



Athena’s obstacle avoidance algorithm removes the massive occupancy grids required for the standard D*Algorithm by using a custom algorithm, adapted from Virginia Tech, which efficiently avoids obstacles and tends to generate relatively efficient paths without large memory requirements. This algorithm runs on the FPGA and updates LIDAR scans at 70 Hz. Athena processes the effect of a data point on each of 18 parallel paths 12,600 times per second. Therefore, each robot path requires 112 inches of travel to be examined for collisions. The FPGA passes the available path lengths to the real-time target, which uses this information to subsume a desire to go towards a particular heading. For the autonomous competition, that desire was simply to move forwards.

Extensibility

To build more complicated robotics systems, a code base must develop over time. To achieve this, software teams need to preserve abstraction barriers between code modules that have been previously developed. This becomes challenging, however, if computing occurs solely on CPUs. While software may not require interaction with other code modules, parallel processes must vie for the same finite number of cores. This means that while you do not explicitly design interaction, parallel threads and processes are managed by the operating system's scheduler. As a result, they frequently interact in ways that even developers who understand the entire system cannot predict or comprehend.

This problem is greatly reduced in the Athena architecture. Code developed for the FPGA functions the same with a parallel architecture, regardless of whether it is running as one process or along with several other processes on the FPGA chip. It is true that resources are still finite and if they run out, the code fails to compile. However, upon successful compilation, a developer knows that an FPGA module will not interact with other modules because the two different programs share no resources. As a result, the developer can preserve a perfect abstraction barrier.

Vision

The vehicle attempts to increase cognition rates and use various computing resources like PCs, real-time targets, and FPGAs, and promote system-level design thinking across the vehicle's many technical domains. We decided to put our vision processing on the laptop CPUs where complex floating-point computations like matrix operations can be easily performed.

We made use of the excellent vision development module in LabVIEW to completely avoid writing the basic components of powerful vision algorithms. We quickly pieced together a line finding system with five filter choices, a fisheye distortion remover, and a Hough transform—which we would not have had time for if we had to write our own RGB threshold filter.

Next year, we plan to put the vision processing on the real-time target and FPGA to match the obstacle avoidance speed because vision was the new limiting factor in speed.

Conclusion

Although we were troubled with hardware failures in our circuitry at the competition, our FPGA-based approach was appreciated by the judges, who awarded us third place in the design competition. Hours before the end of the four-day competition, we hacked together enough circuitry workarounds to prove we could avoid lines and obstacles. Unfortunately, with so little time left to tune the filters, the two runs we made were very disappointing. But thanks to the extensibility of our architecture, we can focus almost entirely on vision for next year. Without our NI Single-Board RIO and the LabVIEW FPGA, none of our technical innovations would be possible. And without support from the National Instruments R&D staff on the weekend of the competition, we would never have advanced in the competition.

Watch a Video of Athena’s Early Obstacle Avoidance Algorithm Testing

Source URL (retrieved on *01/25/2015 - 7:35am*):

http://www.ecnmag.com/blogs/2010/10/creating-autonomous-robot-igvc-autonomous-challenge

Links:
[1] http://www.ni.com/labview/
[2] http://sine.ni.com/nips/cds/view/p/lang/en/nid/205900
[3] http://sine.ni.com/nips/cds/view/p/lang/en/nid/11834
[4] http://www.ni.com/singleboard/
[5] http://en.wikipedia.org/wiki/Subsumption_architecture