6.111 Final Project Proposal
Alex Hornstein
11.3.2005

One of the greatest obstacles in human/machine interaction is the communication barrier. We, as humans, speak a different language from machines. In the past, we have overcome this barrier by inventing language structures such as programming languages, which allow us to speak in a finite vocabulary that is easily translated into machine language. The problem with this is that it requires anyone who wants to communicate with computers to learn and be familiar with programming languages and concepts. We compromise by having programmers write user interfaces intended for most users, so they can use the computer without 'speaking it's language.' This allows people to use a computer, but only to the level of complexity that the programmer designs the interface, and so there are some intrinsic limitations in this system.

This project will create a new language structure that is more intuitive for humans to use. Users will use gestures to describe a path for a robot to take, as well as conditional reactions for the robot(i.e. go straight. If you hit a wall, turn left. Otherwise, keep on going straight). An FPGA will interpret these gestures using a video camera, and then direct a physical robot according to the user's instructions.

In this project, an analog color camera will be pointed at a user. The camera will output a NTSC composite signal to the FPGA, which will interface to it via the onboard ADV7185 video decoder. The FPGA will track the user's gestures by tracking a finger, which will be uniquely colored, and store the set of (x,y,t) triplets in RAM. It will also identify gestures signifying conditionals, and then compile the user's gestures and traced paths into a sort of program, which it will then "run" on a two-wheeled robot.

The robot itself has two drive wheels and two balancing castors. Position feedback is given by an optical mouse mounted on the robot slightly off the floor, communicating with the FPGA via the PS/2 protocol. Batteries are included on the robot, as well as a small H-bridge and a larger brushed motor amplifier for each motor. The robot will be tethered to the FPGA (with lots of slack). Each motor will only require two wires to control.

This project is an implementation of a new concept for human/computer interaction. Using this gesture language, someone could easily create a robot with behavior such as wall-following in a few minutes, rather than learning a new set of programming commands and then writing equivalent code for the robot. By implementing it in harware, we can have a fairly sophisticated system running on a single FPGA and a few interface chips, rather than on a full PC. This could lead to a small, cheap system that allows intuitive gesture-based programming, that can be added on to existing robot systems.

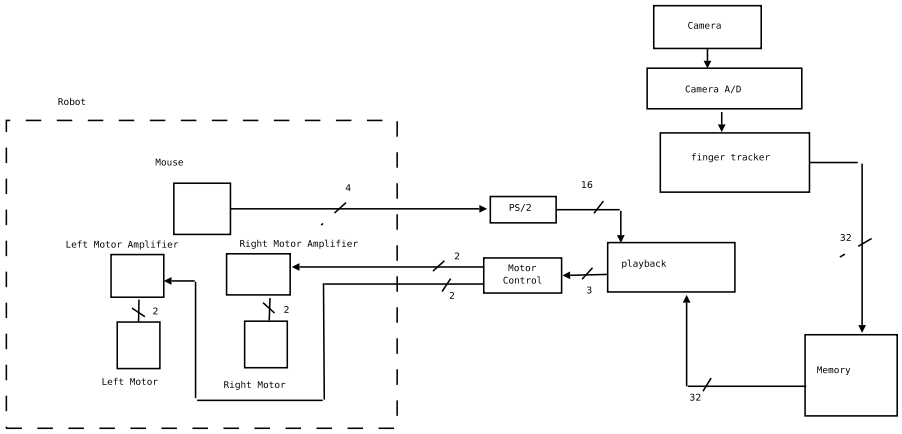

Figure 1: Block diagram of the system

The basic system, as outlined in the block diagram above, is very simple. The camera A/D module has already been implemented and is available on the course website. All the finger tracker module must do is go through a frame and average the positions of all points that are a target color/brightness. This average is taken to be the position of the finger at a given time. It then stores the set of (x,y) points in memory. When it is no longer in record mode, the playback module will 'playback' the points by assuming the robot is at 0,0, and directing the robot to the next stored point in memory, using closed/loop feedback from the PS/2 mouse. The PS/2 interface module is also already written, and is available on the course site. The motor control module is just a wrapper module that takes commands such as 'forwards' or 'left' and translates them into pin-level signals for the motor amplifiers .

1 modules

1.1 Camera A/D

The bulk of this code is already written. A module available on the course web site interfaces with the ADV7185 video decoder chip, returning a stream of YUV values for the current scan point in the camera's frame. It also returns hsync and vsync signals.

1.2 Finger Tracker

This module takes in two 10-bit data signals from the camera A/D module, as well as two one-bit hysnc and sync signals, a 'record' button from the labkit, and a 27mhZ clock. When 'record' is pressed, the module keeps a running average over one frame (one vsync) of the camera of the x,y positions of all green points in the camera field, and outputs the two 10 bit x,y positions, as well as a 9-bit memory address. It updates the address at a sample rate of 2Hz, taking a maximum of 512 data points.

1.3 Memory

The memory can be fairly small, specifically, a block of RAM 20 bits X 512. The RAM blocks within the FPGA are sufficient.

1.4 Playback

The playback module takes in two 10-bit data signals from the memory, two 8-bit delta x,y signals from the PS/2 module, and a 'playback' button from the labkit. It outputs a 3 bit high-level control signal to the Motor Control module. The Playback module is responsible for examining a path in memory and recreating it on the physical robot, when the playback button is pressed.
The module essentially runs the algorithm:

- Load the x,y values from the current sample point
- If robot's current x,y values are not equal to the values from memory, output proper command to motor control module (forward, back, left,right)

- Add deltaX and deltaY from the PS/2 module to the robot's X and Y values, and repeat

- If the robot's X,Y values are equal to the values from memory, increment the memory counter, and repeat.

1.5 PS/2

The PS/2 module is already written and is available(for a keyboard) on the course site. I will have to modify it for a mouse datastream, which is three 11-bit frames. The module will output two 8-bit values for the change in X and Y since the last update, as well as a 1-bit value to signify an update of the data values.

1.6 Motor Control

This module is essentially a ROM. It takes in a 3 bit enumerated value for forward, back, left, right, stop motion commands, and outputs two four-bit values to the left and right motor H-bridges that correspond to the desired motions

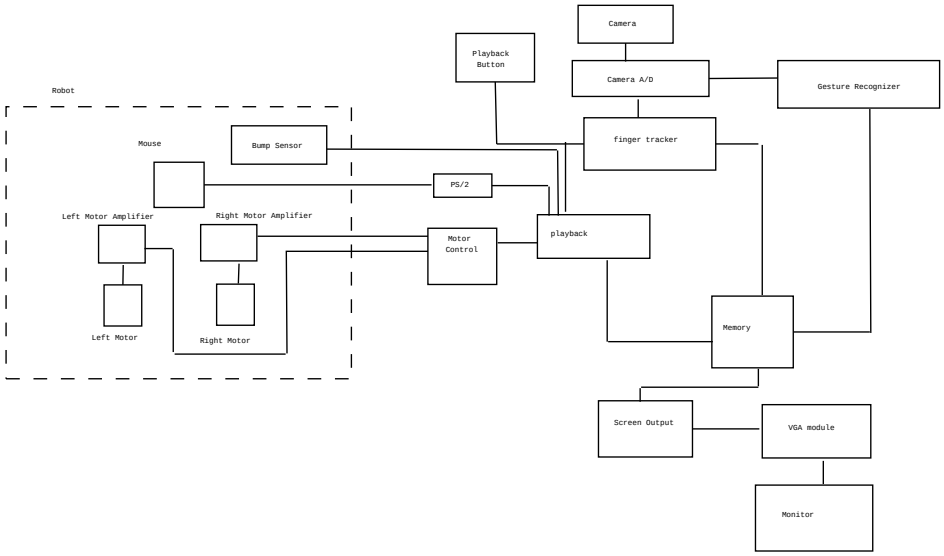

Figure 2: Block diagram of the more advanced system

A more advanced system would implement a gesture recognizer, which would look at the camera frame simultaneously with the finger tracker, to recognize gestures for conditional expressions, etc. There could also be a module which would display the currently stored points on a VGA monitor. Finally, a simple sensor, such as a bump sensor, could be implemented on the robot, to allow for evaluation of conditional statements entered by the user.

2 advanced system modules

2.1 Memory

The memory can be fairly small, specifically, a block of RAM 21 bits X 512. The additional bit stores information about conditionals, which is generated by the gesture recognizer module.

2.2 Gesture Recognizer

This module takes in the same camera signal that the finger tracker does, and maintains the same memory counter, offset by 20 bits. It outputs a 1 bit data bit and a 9 bit memory address. It also looks through the camera frame, but for a different pattern, such as a blue and red square, which could signify a conditional, at which point it would output a 1 as its data bit, to signify a conditional, and it would be stored in memory along with the X,Y sample.

2.3 Screen Output

This module goes through the current sample points and outputs the points to a VGA module to display them on a computer monitor, for debugging purposes.

2.4 VGA module

This module is already written. It was part of the pong lab.

2.5 Playback Module

The playback module now takes as an input a bump sensor from the robot. If the sensor registers a bump, it will then follow the points with a conditional bit set, otherwise it will follow the points without a conditional bit.

3 Implementation

The robot is already built. I would like to implement control starting with low-level modules and working up. I would like to first built the PS/2 and motor control modules, and test open-loop feedback, using buttons on the labkit for control, in place of the playback module. I would then like to implement a playback module that can do closed-loop feedback based on a preset list of points in a rom. Then I would like to add a memory and build the finger tracker module, as which point the screen output module will be invaluable for debugging purposes. Then, I would like to add the gesture recognizer module.
Testing this system is fairly straightforward: If the robot goes to points the user sets, the system works. Later, once the gesture recognizer moudle is built, if the robot follows a user-defined 'program,' it is working.