Kathy Camenzind
Alex Huang
Xavier Zapien

Final Project Proposal: Pong.iRL

6.111 Fall 2018

Table of Contents

Overview

Inspired by the pong lab, our team has decided to build a robotic system to play air hockey against a human opponent, which is essentially the same game as pong but with an added degree of motion. The air hockey robot consists of a machine vision module, air hockey robot logic module, and a motion control module:

- *Machine vision module*: responsible for processing camera VGA data, buffering it into a frame and then processing the frame to isolate the position of the puck and paddle. It then passes this information to the air hockey robot logic. We anticipate this module to be the most challenging aspect of the design.

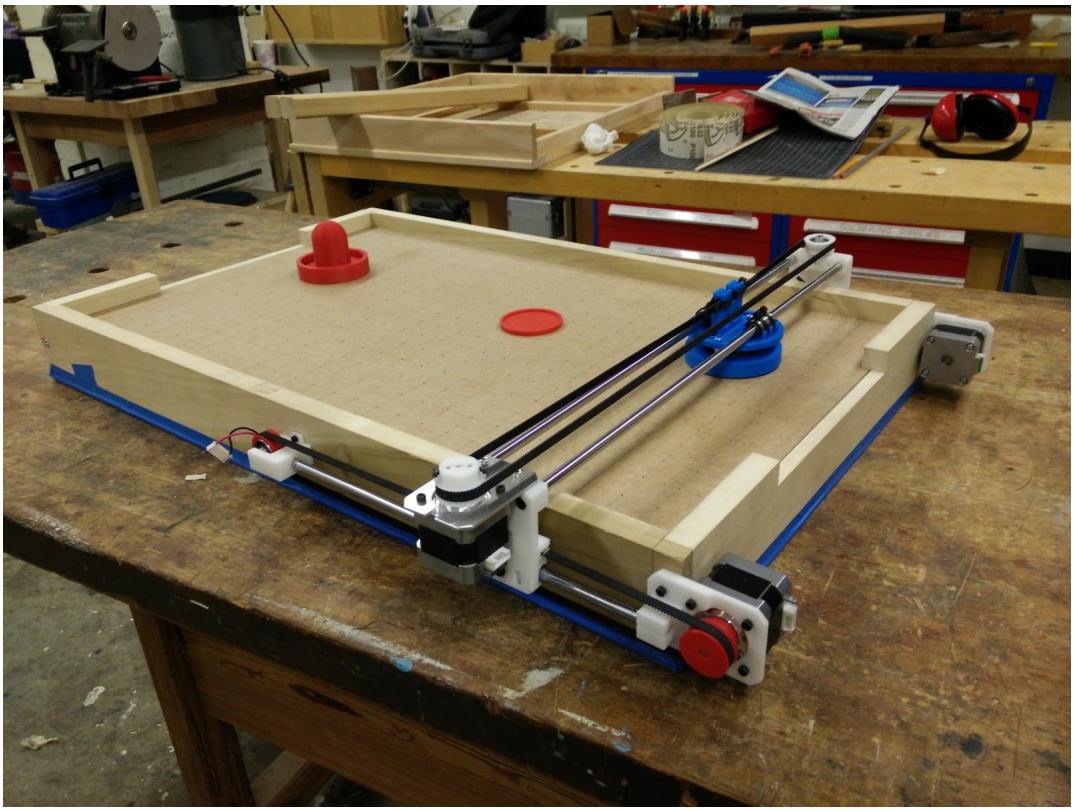
- *Air hockey robot logic module*: implements an “AI” that predicts the trajectory of the puck (accounting for bounce off the walls) and outputs a desired paddle position based on a game strategy which we will implement.

- *Motion control module*: controls how to actuate stepper motors to move paddle. An important consideration with stepper motors driving a load is that they will need to be ramped (accelerated) to avoid stalling. This module will ensure that we move the paddle in the most efficient way possible.

Implementation

The Air Hockey Robot

There is a substantial mechanical element to this project, but we already have it built:

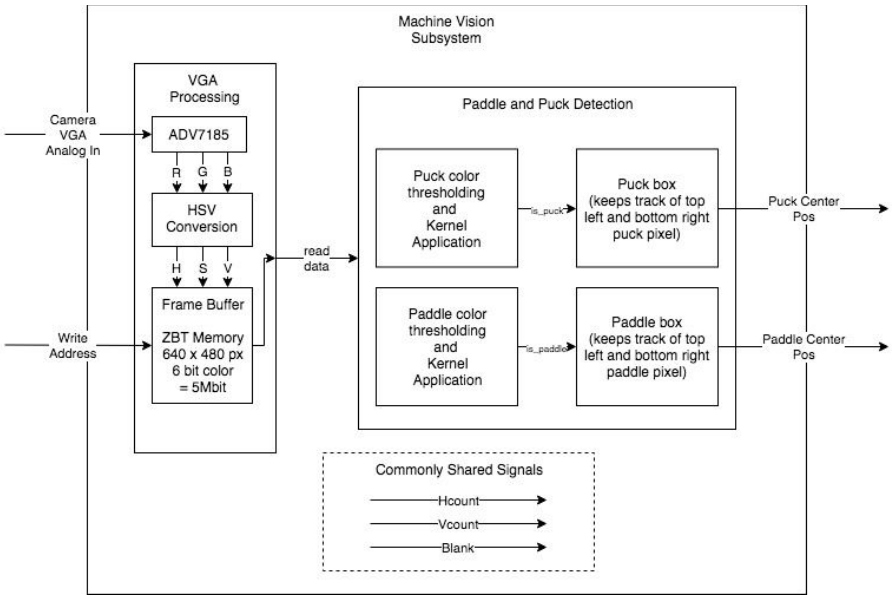


This is a miniature air hockey table that we have custom built; it measures roughly 18” x 29”. There is a set of rails that allows the gantry (the paddle + rail + motor assembly) to

move on the X axis and another set that allows the paddle (pictured in blue) to move on the Y axis. These parts are driven by 3 stepper motors. The table itself is essentially a big enclosed chamber with two computer fans underneath to draw air in. Tiny holes are drilled in a gridlike formation on the table top to allow air streams to escape. This airstream levitates the puck, which can slide around with little friction.

Machine Vision Module

Description:



The machine vision module is responsible for handling camera data and identifying the paddle and puck. The camera will be mounted on the air hockey table and provide a birds eye view. Its data is passed to the ADV7185 NTSC decoder, which outputs RGB digital data. To save space, we are considering using only the top 6 bits of this color information. This data is then fed to the HSV conversion module which, as the name implies, outputs the HSV values of the pixel. We choose to use HSV because inevitably we will need to play with threshold values to pick up the colors of the paddle and puck and having HSV allows for more intuitive tuning of these parameters. The HSV data will be stored in a ZBT memory frame buffer.

The paddle and puck detection module will then read this data given a specific Hcount and Vcount. An erosion and dilation kernel will be applied to filter out noise and the top-left and bottom-right pixel will be kept. Potential problems of doing this may be

noise causing issues if the erosion and dilation kernel does not adequately filter. A potential solution may be also to include some temporal filtering of the position of the puck and paddle as these are physical objects that cannot move very quickly relative to the processing time of the FPGA. In order to keep the filtering as simple as possible, we will also attach a generous light source so illumination is ideal and noise is reduced. In addition to this, we will also require that players wear a monochrome sleeve when they are playing the game so as to not introduce false positives in our identification pipeline.

Seeing as it may almost be impossible to debug, we will have stream the camera view and overlay boxes which bound the paddle and puck positions so we can debug this module.

The output of this module - namely, the x and y positions of the paddle and puck will be passed to the AI subsystem using a shared register.

Hardware Usage:

Hcount: 10 bits to account for 640 pixels
Vcount: 9 bits to account for 480 pixels
Read/Write address: 19 bits to account for 640 * 480 possible addresses
Puck,Paddle positions: 10 and 9 bit registers for x and y
R, G, B: 6 bits for each color
H, S, V: 6 bits for each parameter

ZBT: 5.5 Mb of usage
HSV Conversion Module: 2x 16 bit multiplier IP

Robot Logic Module

The two arguments passed into the air hockey robot logic module are the paddle and puck positions. Since the camera is 640x480 pixels (aspect ratio 4:3), and the board is 29” x 18” (aspect ratio 4:2.48), there will be some extra camera pixels off of the board on the top, and the number of used pixels is actually 640x397. The coordinates for both (which represent the middle of the object) need to be $\log_2(640)+\log_2(397) = 19$ bits wide.

The first part of the module tracks the puck and its trajectory. It consists of three submodules: a velocity calculator, a deflection calculator, and a trajectory predictor. The

velocity calculator tracks stores 2 previous puck positions and calculates the current velocity. The pos[i] -> pos[i-1] gives us the currently velocity and direction. However, if the direction is considerably different from pos[i-1] -> pos[i-2], then we bounced off of a wall. We can then use the deflection calculator, which takes in the previous and current position, determines the wall that was hit, and calculates the new velocity. Lastly, the trajectory predictor uses these values to predict whether the puck will reach the goal, at what coordinates, and at what time.

The main submodule is the robot AI, which outputs the desired robot position. At its most simple implementation, the AI will ignore the puck when it's travelling towards the opponent, and move the paddle to the predicted spot where the puck will reach the goal after it's been hit by the opponent. This submodule will output to the motion control module a desired paddle position.

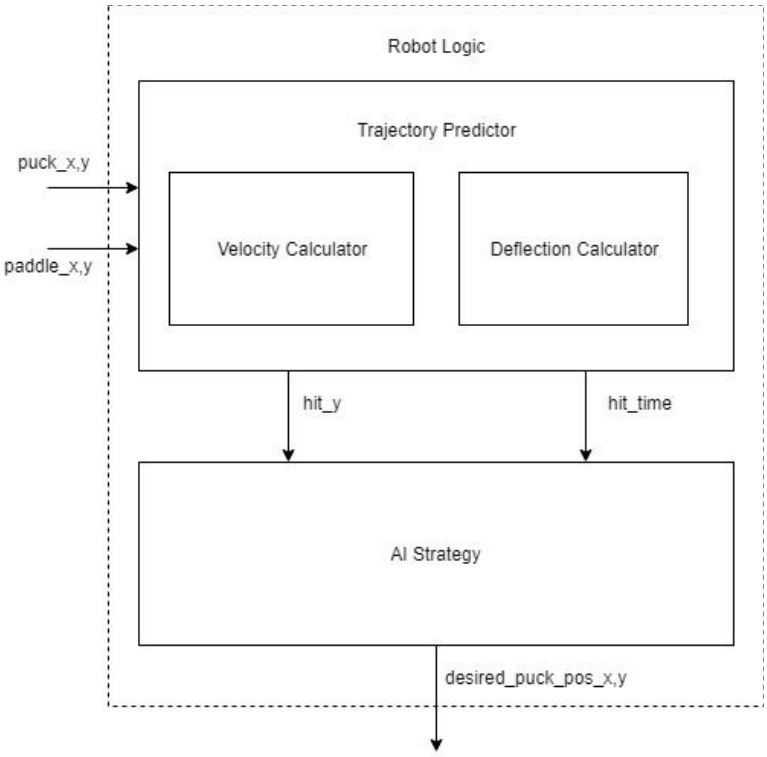
There are many ways to implement a more complex version of this AI. Some ideas are adding a second control phase: (1) moving into position to block the puck from scoring and (2) moving forwards to hit the puck. We could also implement a more aggressive strategy, where the trajectory submodule would output an array of (position, time) values of the puck's trajectory on the AI's side of the board, instead of just where it will reach the goal. It could then decide whether it has to move backwards at all, versus hitting the puck while far forward and giving the opponent less time to react.

Hardware Usage:

Inputs
$puck_{x,y}$: 19 bits
$paddle_{x,y}$: 19 bits

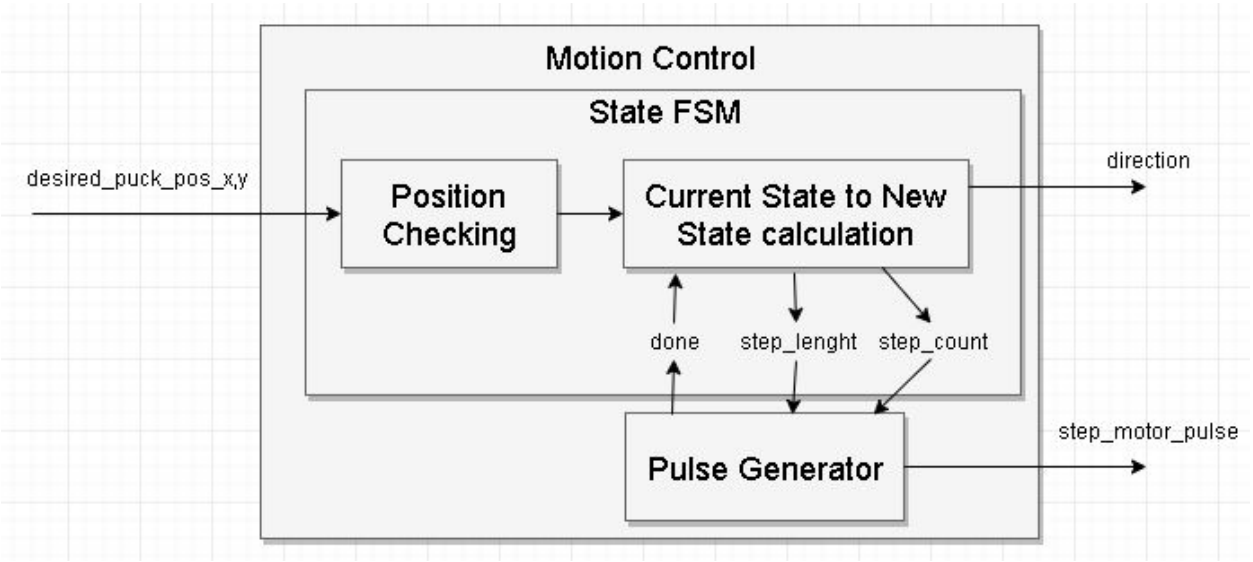
Outputs:
$paddle_{x,y}$: 19 bits

Internal:
Since our clock will be running at a high speed, we'll likely add a pipeline this in between the trajectory predictor and AI strategy modules. Other than this, hardware is just a small number of arithmetic operators,



and possibly an FSM to implement the AI if we choose a more complex strategy.

Motion Control Module



Initially, the motion control module will be interfacing with one of the three stepper motors to control the Y position of the blue paddle. This module contain 2 sub-modules: State FSM and a Pulse generator. Given the outputs of the AI Module, the position checker submodule inside of the FSM will first check that this position is obtainable by the paddle. This error checking is vital for the safety of the paddle. If the position is valid, the 9 bits are sent to the current and new state calculator. This section will compare the last two desired puck positions and generate three outputs: direction, step_lenght, and step_count. Direction is fed directly onto the A4988 Stepper Motor Driver Carrier and will dictate on what direction the robot will move. Step_count will be a 15 bit value that will be fed into the pulse generator sub module. Step_count will be the number of steps needed for the motor to reach the new desired position. Step_length will also be sent to the pulse generator. This will currently be fixed to make a step length of 2us, but after testing for stalling, this value might be changed to prevent damage to the motors.

The Pulse Generator submodule will create a pulse out of the step_count and step_lenght values. The pulse will be a 5V 2us pulse that will happen every 4us until the step_count reaches zero. Once the final pulse has been delivered, the robot assumes that it has reached it final destination and asserts a HIGH on its done output. This

allows the state transition to only send new data to the pulse generator only when it has finished writing the last position to the paddle.

One large constraint of this module are the physical restrictions of the acceleration in the stepper motors. For that reason, we will have to test how many steps and how fast we need these steps to be in order for the system to react on time. We will also have to work on making sure our pulses do not drive too much current through our motors.

Hardware Usage:

Inputs
desired_puck_pos_x_y: 19 bits

Outputs:
direction: 1 bit
Step_motor_pulse: 1 bit at 500kHz

External: A4988 DMOS Microstepping Driver

Logistics

Goals

Our baseline goal is to have all of these modules separately working, with a basic robot AI. This means that we have the machine vision module successfully able to detect the position of the puck and paddle. The robot AI will be purely defensive, stay at the back, and like in pong just have a 1D range of motion in front of the goal. The motion control module, given inputs, will be able to move the paddle to the desired position (1D motion).

Our expected level of functionality is to be able to connect the three modules, and to have a slightly more complex motion control and AI that moves the paddle in 2 dimensions. This will still largely be a defensive strategy, but will be able to move the puck forward when it’s moving slowly enough.

Our reach goals are as follows:

- Account for bounces off of two walls (corner cases) in the puck trajectory predictor
- Have an aggressive offensive strategy that allows for hitting the puck forward and/or doing trick shots
- Maximize stepper motor ramp up speed
- Have better image prediction not based solely on color so that players don't have to wear a monochrome sleeve

We also have backup plans for the more complex components. In particular, if the machine vision module proves too memory-intensive to effectively implement, we may instead use an IR sensor to point-detect where the puck and paddles are.

Division of Work

Given that there are both 3 modules and 3 team members, we have a natural division of work of 1 module per person. Alex is in charge of the machine vision module, Kathy the robot AI module, and Xavier the motion control module. However, given the varying complexities of these modules, the additional modules that will have to be made for debugging and testing, and the tight coupling that has to happen between these modules, we plan on working collaboratively and having somewhat flexible responsibilities since it's hard to know which modules are going to need the most debugging before working on them.

Specifically, we will all work together on the actual machine vision algorithm portion of the machine vision module since that module is significantly more complicated than the other two. We will also work together on the physical setup of the camera and board together since that impacts all of the modules.

Projected Timeline

11/1 - Proposal done
11/8 - Have camera communicating with Nexus with output to computer monitor, test driving stepper motors with FPGA
11/15 - Implement the machine vision module, implement the trajectory predictor
11/22 - Connect the modules and test initial implementation
11/29 - Develop a more strategic robot AI and improve machine vision

12/6 - Run tests and fine tune
12/10 - Final project checkoff