Progress Report: Project BALANCE

Matthew Ebert; V00884117
ECE 490; Supervisor: Dr. Capson
dept. of Electrical and Computer Engineering
University of Victoria
ebertmx@gmail.com

Abstract – This report covers the progress of project BALANCE: a ball and beam visual servoing system. At the time of this report, the mechanical and electrical system are nearing completion. The software development environment has been set up, but the control system has yet to be implemented. Since previous research has been conducted on visual servoing techniques, any issues implementing the control system should be minor.

Key words –visual servoing, camera, pose, control theory, computer vision

I. Introduction

Project BALANCE, a ball and beam visual servoing system, has progressed significantly. Several setbacks caused delays in development in the software and hardware components as modules were re-implemented several times. The hardware design is complete and currently being produced. The software environment was successfully integrated with the available hardware and the control system is under development.

The background and research conducted for this system is covered in separate reports [1] [2] [3].

II. Current State

Project BALANCE is comprised of 3 subsystems: mechanical, electrical, and control.

A. Mechanical

The mechanical design consists of mostly 3D printed components. Figure 1 shows the designed assembly.

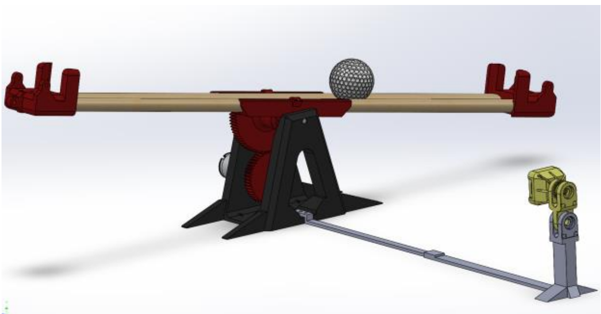

Figure 1: CAD model of the BALANCE mechanical system.

The design integrates wooden dowels, a DC motor, and a raspberry pi camera. Figure 2 shows the current produced version of this design.

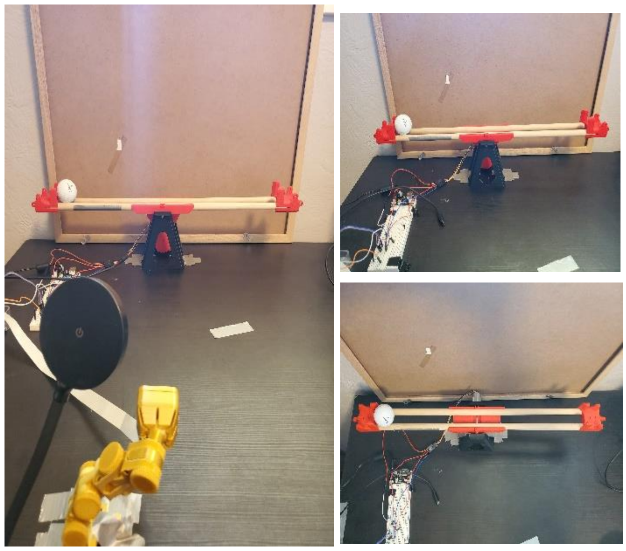

Figure 2: Current produced mechanical design for project BALANCE

B. Electical Hardware

The electrical system consists of a raspberry pi model 3 B, a BLDC motor, a raspberry pi CAM, a power supply, and motor controller circuit. The electrical schematic for the motor and raspberry pi is shown in Figure 3.

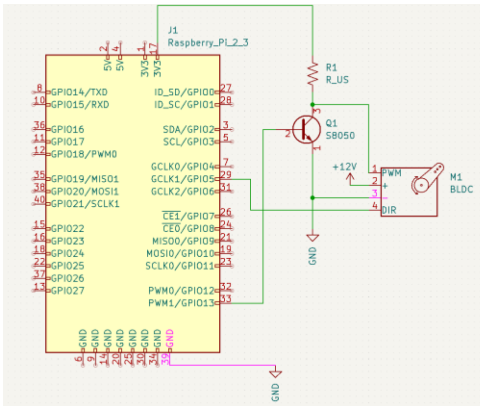

Figure 3: BALANCE electrical schematic

The raspberry pi model 3 (RPI) computer has a 4-core processor which runs a MATLAB specific version of the Raspbian OS. The RPI has multiple GPIO pins which can read, write, and produce PWM signals. It has a specified camera input which is compatible with the raspberry pi CAM.

The BLDC has 5 ports which include power, PWM speed control, direction, and FG signals. The PWM signal built into the motor is inverted meaning when the PWM duty cycle is 100% the motor is off and at full speed at 0%. To integrate with the raspberry pi and prevent the motor from running at full speed on startup, a transistor circuit was used to invert the logic. The FG pin is currently nonfunctional.

The raspberry pi cam mounts directly to the mechanical design and connects to the raspberry pi through a ribbon cable. The camera module contains an OV5647 sensor which can record at 2592 × 1944 resolution with a maximum frame rate of 60 fps.

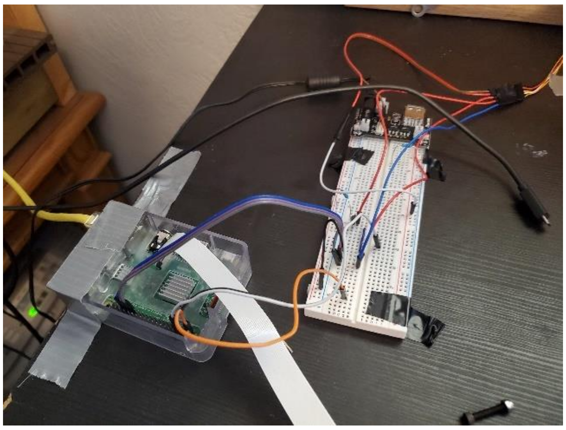

Figure 4: Current electrical system

C. Software

Currently, the control software is being developed in MATLAB and Simulink. These programs have an application which can compile their native programs into a C++ .EFL file which can run independently on the raspberry pi. This feature allows for application to be development in the MATLAB/Simulink environment but deployed onto the raspberry pi to run locally. Thus, the embedded requirement of the system is still satisfied.

Prior efforts have focused on creating the development environment and deploying programs on to the RPI. Currently, the software can read and write GPIO pins, read and process camera data, display frame data, and implement basic control on the BALANCE system. A program was completed in both MATLAB and Simulink which complete the following tasks:

1. Grab a frame from the camera
2. Binarize the image for the golf ball
3. Find the circle and calculate its center
4. Calculate which side of the system the ball is on
5. Move the beam so the ball will start rolling to the other side.
6. Repeat.

The implementations can be seen in Figure 5 and Figure 6.

Figure 5: MATLAB code for simple control of the BALANCE system

```
%A function which enacts a simple control using
visual servoing on
% a ball and beam system with a raspberry pi.
% Created by: Matthew Ebert, 2022 -07 -20
% ECE 490, University of Victoria

function simplecontrol()%#codegen
```

```
% Create a connection to the Raspberry Pi hardware
r = raspi();
middle = [311;297];
%setup camera
cam = cameraboard(r,'Resolution','640x480');
%set up PWM pin
configurePin(r, 13, 'PWM');
writePWMDutyCycle(r, 13, 0.0);
writePWMFrequency(r, 13, 8000);
%set direction pin
dir = 1;
configurePin(r,5, 'DigitalOutput');
writeDigitalPin(r,5,dir);
%set initial speed between 0 and 1.
spd = 0.05;
writePWMDutyCycle(r, 13, spd);
count = 0;
go = 1;
while count<100 %for 1000 frames
    tic;
    %grab frame from camera and process image
    I = snapshot(cam);
    img = rgb2gray(I);
    img = imgaussfilt(img,1);
    %threshhold the gray scale image to isolate
the ball
    img = imbinarize(img,0.60);
    % subplot(1,2,1);
    %imshow(img);
    %find circles of certain radius based on
binary search
    [centers, radii, metric] =
imfindcircles(img,[20 30],
'ObjectPolarity','bright');
    imshow(I);
    %draw circles
    viscircles(centers, radii,'EdgeColor','b');
    if(size(centers,2)>1)
        %if the ball is detected
        %pick direction and set speed
        delta = centers(1) - middle(1);
        alpha = centers(2) - middle(2);
     %Limit how far the beam can rotate based on
the hieght of the ball.
        if (alpha > 0 && go )
            if delta>0
               %if ball on right side, rotate left
                dir = 1;
                writeDigitalPin(r,5,dir);
                spd = abs(delta)/1000;
            elseif delta<0
               %if ball on left side, rotate right
                dir = 0;
                writeDigitalPin(r,5, dir);
                spd = abs(delta)/1000;
            end
    %set speed based on balls distance from center
            writePWMDutyCycle(r, 13, spd);
        else
            go= 0;
            writePWMDutyCycle(r, 13, 0);
        end
        if(1==dir && delta<0) || (0 ==dir &&
delta>0)
            go = 1;
        end
    end
    fps = 1/toc
    count = count+1;
    %needed for debuging
    %pause(0.00001);
end
%turn of motor
writePWMDutyCycle(r, 13, 0.0);
end
```

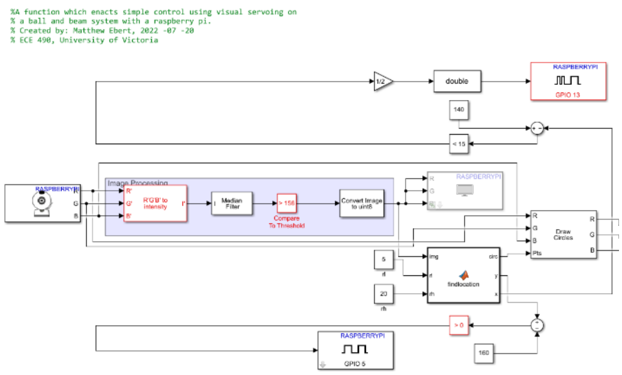

Figure 6: Simple Simulink control system for BALANCE system

Several tests have been preformed which verify the operation of some parts of the electrical, mechanical, and software subsystems. These will be detailed in the final report.

III. Problems and Set Backs

Several issues caused changes to the original design.

A. Change of Controller

Initially, an ESP32 microcontroller was selected to implement the visual servoing system. This device was replaced with the raspberry pi. The reason for this switch was due to issues with building the project with OpenCV. For currently unknown reason, any portion of the OpenCV library would not compile for the ESP32 processor. Since several computer vision processes are intended to be implemented with BALANCE, rather than manually creating these functions, a different controller was selected.

Additionally, development with the ESP32 was slow with our current skills and understanding of the system. Switching to the raspberry pi, allowed for a more familiar development environment to be used.

B. Motor feedback signal

Originally, the beam angle was to be calculated with the built-in encoder on the BLDC. However, after several tests, no relation could be found between the motor speed, and the feedback signal. Consequently, the encoder is assumed not operational and alternate methods of angle calculation are needed.

IV. Future Development

Further development includes completing the mechanical design and developing the full visual servoing system.

The mechanical design development will involve creating a constant reference point for the camera and stabilizing the system's motion.

The software component requires the most development. The following steps are planned:

1. Find the camera's intrinsic calibration parameters
2. Find the control system constants (J, L, m) and implement the transfer function into code.
3. Use CV to find the angle of the beam
4. Control the BALANCE system with IBVS
   a. Create an error function with the feature jacobian
   b. Apply the error function to the system transfer function to control the motor
   c. Use proportional control on the BALANCE system.
5. Control the BALANCE system with PBVS
   a. Use calibrated camera to estimate the pose of the golf ball and system
   b. Create an error function with these estimations and apply to transfer function and system
   c. Use proportional control on the BALANCE system.
6. Experiment with different types of control to improve the BALANCE system

V. Conclusion

The base development requirements of project BALANCE have been complete. These include the mechanical and electrical sub systems and the development environment for the software control. The control system is in its infancy but can now progress quickly since the other systems are operational. Most of the control and visual servoing theory have been covered in previous research; therefore, no major issues are expected when implementing the rest of the system. The use of MATLAB and Simulink should also speed up development due to the large number of resources available for these programs.

VI. References

[1] M. Ebert, "ECE490: Virtual Servoing Project Proposal," dept. of Electrical and Computer Engineering, University of Victoria, 2022.

[2] M. Ebert, "Review of Visual Servoing Fundamentals," dept. of Electrical and Computer Engineering, University of Victoria, 2022.

[3] M. Ebert, "Survey and Review of Computer Vision Theory and Methods," Department of Electrical and Computer Engineering, University of Victoria, 2022.