Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

31 de Janeiro de 2020

Otimização do Controle por Câmera de um Projeto Ball and Plate

Luiz Felipe Monteiro de Albuquerque Fonseca

PUC RIO

Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Otimização do Controle por Câmera de um Projeto Ball and Plate

Aluno: **Luiz Felipe Monteiro de Albuquerque Fonseca**

Orientador: **Wouter Caarls**

Trabalho apresentado com requisito parcial à conclusão do curso de Engenharia de Controle e Automação na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

Gostaria de agradecer à minha família, em especial meus pais, avós e irmãos, responsáveis pela minha formação como pessoa e os maiores incentivadores da minha formação acadêmica, sem os quais esse trabalho e tudo que levou a ele não seria possível. À minha namorada por aguentar os momentos difíceis e ser paciente mesmo quando mais desafiador. Aos meus queridos amigos Vitor Carneiro, Mateus da Silva e Felipe Ribas, com quem compartilho algumas das minhas melhores lembranças do meu tempo de graduação, assim como aos colegas do Coral PUC-Rio. Ao meu orientador Wouter Caarls, cuja disponibilidade, atenção e ajuda foram essenciais para a conclusão do meu curso. E também a Mauro Esperanza, Nikolay Rutkevich e Antonio Leite pela ajuda com este trabalho.

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Resumo

Para este projeto foi desenvolvido um sistema de controle da posição de uma bola rolante sobre uma mesa de dois eixos de liberdade cujos ângulos são movimentados por motores de corrente contínua, sendo a posição da bola observada por uma câmera. Dois controladores PID independentes foram sintonizados para que, através da atuação sobre os motores, a bola fosse levada a uma posição desejada e lá mantida. O controle da mesa é executado em um Arduino, que se encontra em comunicação serial com um programa de Python onde é feito o processamento e segmentação da imagem da câmera e o controle da posição da bola. O trabalho contempla a compreensão do tipo de controle e do modelo do sistema, desafios e resultados experimentais, incluindo um desafio final de controle de trajetória da bola.

Palavras-chave: Ball and plate, otimização, controle, PID

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Optimization of a Ball and Plate control via webcam

Abstract

This project describes the development of a control system for a rolling ball on a plate with two degrees of freedom, with each axis acted upon by a direct current motor and a camera observing the ball's position. Two independent PID controllers were tuned so that upon operating the motors the ball would be taken to a specific desired position and then stay balanced where it stood. The table's control is implemented on an Arduino board that is in contact through serial communication with a Python program, which in turn runs the processing and segmentation for the camera image as well as control of the ball's position. This work encompasses the understanding of the type of control implemented as well as the system model, experimental challenges and results, plus a final challenge of trajectory control.

Keywords: Ball and plate, optimization, control, PID

Sumário

1 Introdução

O sistema Ball & Plate é um exemplo clássico de controle, assim como o Pêndulo Invertido e o Ball & Beam. Trata-se de uma plataforma com dois graus de liberdade sobre a qual uma esfera rolante é colocada, tendo sua posição controlada pela inclinação da mesa. O desafio é portanto levar a bola até uma posição desejada e lá equilibrá-la, atuando somente sobre a angulação da mesa.

Exemplos como esse em laboratório são de alta relevância para o estudo de controle, assim como de campos de estudo associados à automação. Esse tipo de exercício também está relacionado a aplicações práticas como maximizar a captação de energia solar através do posicionamento dos painéis, manter a posição de uma câmera estável sobre uma embarcação marítima ou outras situações onde deseja-se obter equilíbrio agindo contra perturbações e instabilidades. Para o Laboratório de Controle Inteligente da PUC-Rio, havia o desejo de se ter um modelo funcional de Ball & Plate para experimentos com inteligência computacional, mais especificamente aprendizado por reforço.

Para que seja possível realizar esses experimentos, é importante montar o sistema físico, projetar um controle básico e então verificar se é de fato possível controlar essa montagem.

Nesta montagem, a mesa fica presa por dois eixos, sendo a atuação sobre os eixos feita por motores de corrente contínua. Os sensores de angulação dos eixos são encoders de giro e o sensor da posição da bola é uma câmera posicionada acima da mesa. Diversos tipos de controle podem ser utilizados para esta atuação. Neste caso, foi utilizado controle PID tanto para a angulação dos dois eixos da mesa quanto para o posicionamento da bola.

2 Fundamentação Teórica

a Controle

O controlador PID é amplamente utilizado em diversos tipos de sistemas de controle. Como colocado por Ogata, K. em seu livro *Engenharia de Controle Moderno*: "É interessante notar que mais da metade dos controladores industriais em uso atualmente empregam esquemas de controle PID ou PID modificado. [...] A utilidade dos controles PID está na sua aplicabilidade geral à maioria dos sistemas de controle. Em particular, quando o modelo matemático da planta não é conhecido e, portanto, métodos de projeto analítico não podem ser utilizados"(Ogata, 2004).

A função de transferência do controlador na forma de ganhos independentes pode ser vista a seguir:

$$G_c(s) = \frac{U(s)}{E(s)} = k_p + \frac{k_i}{s} + sk_d \tag{1}$$

Podemos observar três elementos do controlador, que dão nome ao mesmo: uma parte proporcional ao erro, uma parte proporcional à integral do erro e uma à sua derivada. O erro é calculado ao subtrair do valor esperado (setpoint) o valor medido pelo sensor, ou seja, a resposta do sistema. O controlador então define a atuação sobre a planta, como pode ser visto no diagrama de blocos:

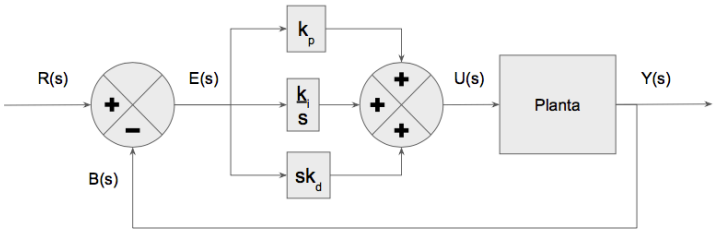

Figura 1: Diagrama de blocos de um controle PID.

Sendo o principal desafio do uso de um controlador PID a sintonização de seus ganhos, é essencial compreender os efeitos da manipulação de cada parâmetro.

O incremento do ganho proporcional k_p deixa a resposta do sistema mais rápida, porém aumenta o overshoot e a instabilidade da resposta.

O incremento do ganho integral k_i tem efeito similar, além de acabar com o erro estacionário, garantindo que o setpoint será atingido, e aumentar o tempo de estabilização. Em alguns casos, é necessário limitar o acúmulo dessa integral.

Já o incremento do ganho derivativo k_d reduz o overshoot e o tempo de estabilização, aumentando também a estabilidade do sistema e o fazendo entrar em regime estacionário, sem oscilações. Pode ser necessário filtrar sua atuação.

b Modelo

Como mencionado anteriormente, não é necessário conhecer exatamente o modelo da planta para executar um controle PID. Porém, é interessante pelo menos analisar o modelo de um sistema similar para melhor entender como funciona. Neste caso, foi estudado o modelo desenvolvido no trabalho *Visual Servoing Stabilization of a Ball and Plate Mechanism by Using an Enhanced Disturbance Rejection Control Method* (Zachi, Riveros, Lima, Gouvêa, Leite, 2017).

Nessa modelagem, foi levado em conta também um sistema com sensores para a posição da bola no plano cartesiano e o ângulo dos dois eixos da mesa, assim como a atuação por motores no mesmo. Supondo que a bola é um corpo homogêneo e simétrico, que rola pela mesa sem deslize ou perda de contato, o modelo dinâmico completo encontrado para a planta foi:

$$\ddot{\ddot{x}}_c = K_2(\frac{R_a B + K_m K_b}{J R_a})\dot{\alpha} + (\frac{-K_2 K_m}{J R_a})V_\alpha \tag{2}$$

$$\ddot{\ddot{y}}_c = K_2(\frac{\bar{R}_a \bar{B} + \bar{K}_m \bar{K}_b}{\bar{J}\bar{R}_a})\dot{\beta} + (\frac{K_2 \bar{K}_m}{\bar{J}\bar{R}_a})V_\beta \tag{3}$$

Sendo α e β os ângulos de giro da mesa $[rad]$; V_α e V_β a tensão alimentada aos motores; K_2 uma constante proporcional à gravidade e inversamente proporcional à altura da câmera; K_m a constante de torque do motor $[N.m/A]$, K_b a constante de força eletromotriz inversa $[V.s/rad]$; J a inércia do rotor $[N.M.s^2/rad]$; B o fator amortecedor do rotor $[N.m/(rad/s)]$ e R_a a resistência de armadura $[\Omega]$. As barras sobre certas variáveis na segunda equação apenas identificam que estes valores são ou podem diferir entre os dois eixos e motores, com a especificidade física de cada um.

Visto que temos tanto o controle da angulação da mesa quanto o controle da posição da bola, o modelo do projeto deve contemplar 8 variáveis de estado no total, 4 para cada eixo. São elas a posição, velocidade e aceleração da bola, além da velocidade angular do eixo da mesa. Trata-se então de um sistema de quarta ordem que vamos controlar com dois controladores PID em cascata.

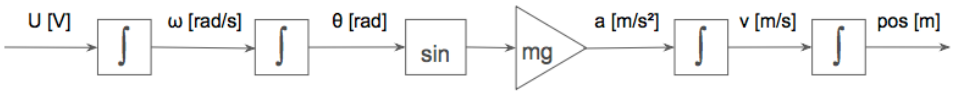

Figura 2: Diagrama de blocos da planta.

3 Desafios experimentais

O principal desafio do desenvolvimento do projeto é garantir a robustez do sistema, tanto mecânica quanto eletrônica. Isso envolve diversos fatores como atrito, folgas nas juntas, interferência e ruído.

Em sua primeira versão, o motor do eixo externo da mesa não era capaz de fornecer o torque necessário para o movimento. Para resolver o problema, foi feita a adição de uma caixa de redução. Esse dispositivo, porém, também introduziu diversas dificuldades ao controle, com muito ruído mecânico por causa do backlash das engrenagens. A solução final foi a troca desta caixa de redução por um conjunto de polia e correia.

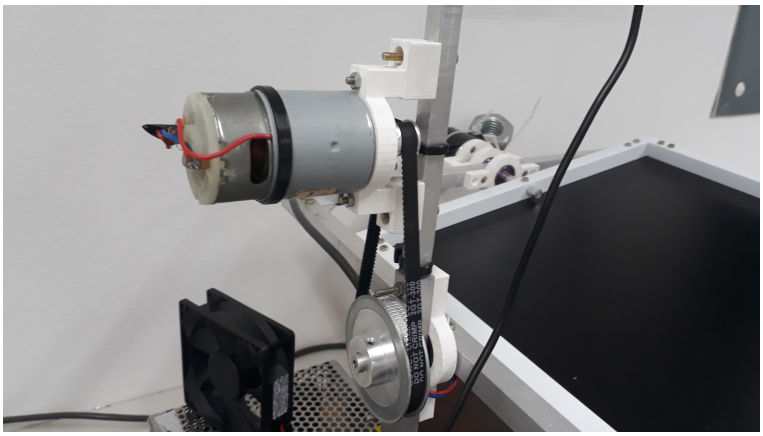

Figura 3: Foto do motor do eixo externo com polia e correia.

Outra questão envolvendo o motor é que ele introduz muito ruído eletromagnético ao sistema, que afeta diretamente a leitura dos sensores pelo Arduino. Foi verificado que esse ruído se encontrava em frequências altas e portanto poderia ser filtrado. Como havia falta de espaço no circuito provisório de condicionamento dos encoders para filtrá-lo apropriadamente, a primeira tentativa foi de fazê-lo pelo próprio programa do Arduino, medindo a ordem e o intervalo de tempo dos sinais recebidos e aceitando a leitura apenas dos condizentes com a frequência esperada. Depois foi possível fazer o filtro passa-baixas também por meio de um circuito maior com capacitores.

4 Projeto

a Montagem

A mesa possui uma placa suspensa com dois eixos livres horizontais, cada um com um motor de corrente contínua 12V e um encoder incremental de rotação com até 5000 pulsos. O motor do eixo externo, como mencionado anteriormente, possui um sistema de polia e correia para melhor atender a exigência de força, dado que o movimento nesse carrega também o peso do motor e encoder do eixo menor. Uma câmera encontra-se fixa acima da placa, ligada ao computador.

Figura 4: Foto da montagem final.

b Eletrônica

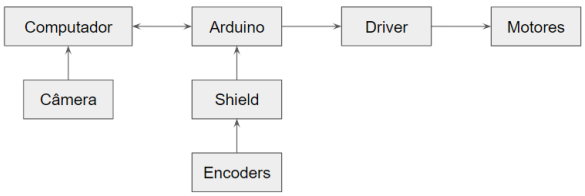

Figura 5: Conexões entre os diferentes dispositivos eletrônicos.

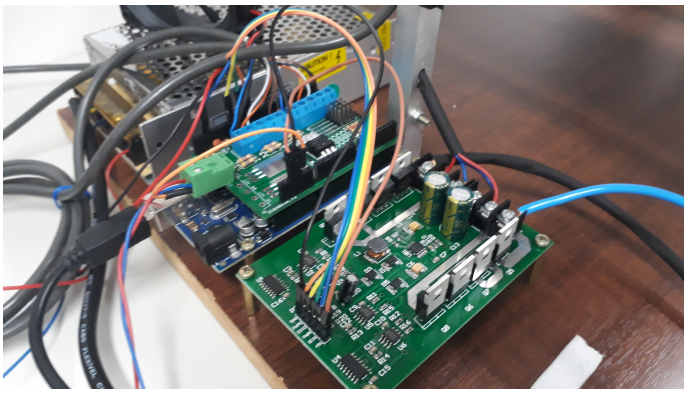

Figura 6: Foto do Arduino com shield (esq.) e o driver dos motores ao lado (dir.)

A leitura dos encoders passa por um shield de Arduino, que condiciona a sua leitura.

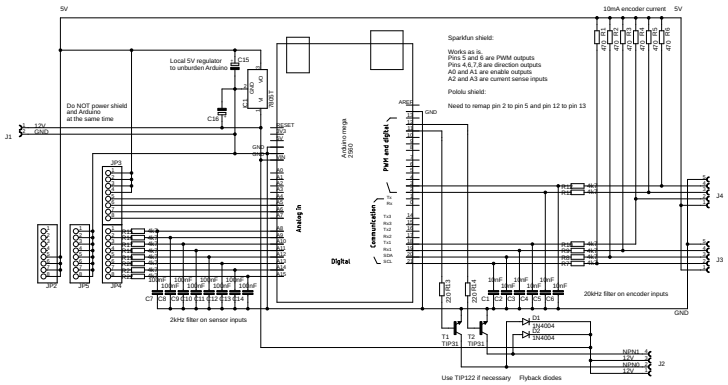

Figura 7: Esquema do shield de leitura dos encoders.

Analisando os valores do circuito do shield, podemos calcular a frequência de corte do filtro passa-baixas mencionado anteriormente. Considerando o capacitor de 10nF do próprio encoder em série com o de 100nF do shield:

$$\frac{1}{C_{eq}} = \frac{1}{100nF} + \frac{1}{10nF} = 0.11 \times 10^9$$

$$C_{eq} = 9,09nF$$

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi(4,7\times10^3)(9,09\times10^{-9})} = 3,73\text{kHz}$$

A saída do Arduino passa por um driver alimentado por uma fonte externa para fazer o controle dos motores.

c Programação

O controle ocorre em dois ciclos com dois controladores PID independentes. A partir dos dados dos sensores (encoders e câmera) os controladores em Arduino e Python definem a atuação sobre os motores.

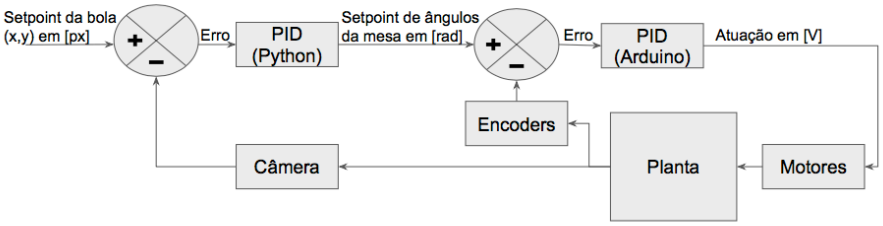

Figura 8: Diagrama de ciclos de controle.

1 Arduino / Software Embarcado

Na programação embarcada é feito o controle da posição da mesa em seus dois eixos, recebendo os dados dos encoders e enviando a atuação para os motores. O controle utilizado foi um PID, com ganhos específicos para cada motor, que recebe do Python através de comunicação serial o setpoint de inclinação de cada eixo da mesa e os leva para o ângulo pedido.

Antes de tudo, é imprescindível para a programação do Arduino definir corretamente os pinos, ou seja, as conexões que estamos fazendo na placa do Arduino. Nas tabelas a seguir vemos as ligações feitas.

Pino na Placa	Variável no Código	Entrada do Driver
6	PIN_MOTOR2_PWM	PWM1
13	PIN_MOTOR2_DIR	DIR1
5	PIN_MOTOR1_PWM	PWM2
8	PIN_MOTOR1_DIR	DIR2

Tabela 1: Pinos referentes à atuação dos motores.

Para controlar o driver dos motores, são necessárias as 4 saídas vistas acima, sendo elas para cada motor: um sinal PWM, que define a aceleração do motor, e um sinal lógico, que define o sentido do giro.

Pino na Placa	Variável no Código
21	PIN_ENCODER0_A
20	PIN_ENCODER0_B
19	PIN_ENCODER0_Z
18	PIN_ENCODER1_A
2	PIN_ENCODER1_B
3	PIN_ENCODER1_Z

Tabela 2: Pinos referentes à leitura dos encoders no shield.

Os encoders possuem 3 saídas cada, ligadas à placa do Arduino através do Shield visto nas figuras 6 e 7, sendo elas A, B e Z. Trata-se de sinais lógicos que nos dão a posição do encoder. A leitura desses sinais foi feita por uma biblioteca desenvolvida para o próprio shield. Pelos sinais A e B, sabemos quando e o quanto o encoder está girando (com precisão de 10.000 pontos), e para qual direção. Já o sinal Z nos indica a posição ”zero”, que é necessária ser encontrada para a inicialização da execução do nosso programa. Na função de setup, além de acionar os pinos utilizados, também acionamos a leitura dos encoders e mudamos o timer do Arduino, reduzindo o prescaler para aumentar a frequência do PWM e melhorar a atuação sobre os motores.

```
open serial communication
first axis homing
second axis homing
serial write ’S’
```

Após esse processo de Homing, é enviado o caracter S para a porta serial, que então inicia o controle e a comunicação com o programa em Python na função loop.

---

if header = True **then**
  setpoints ← serial read
  header ← False
else
  if serial read = 0xFE **then**
    header ← True
  end if
end if

Tendo os setpoints para ambos os eixos, é iniciado o controle PID, com ganhos diferentes para cada eixo.

---

past angle ← angle
angle ← $\frac{2\pi}{10000}$ read encoder
error ← setpoint - angle
D ← filter*D + (1-filter) (past angle - angle)
I ← I + error
I limited between $-I_{max}$ and I_{max}

---

control ← k_P error + k_I I + k_D D
PWM control ← control limited between -1000 and 1000

---

if absolute(PWM control) $>$ 1 **then**
  write(absolute(PWM control) mapped from min_{PWM} to max_{PWM}) to PIN_MOTOR_PWM
else
  write(1) to PIN_MOTOR_PWM
end if
write(PWM control $>$ 0) to PIN_MOTOR_DIR

A atuação sobre os motores é feita então enviando o sinal PWM para os motores e a direção de giro, essa que depende do sinal da variável de controle.

Valores menores que 1 da variável de controle são enviados como 1 para o driver pois ele não funciona com PWM de 0% e precisa que sempre seja enviado algum pulso.

2 Comunicação

A comunicação serial entre os progrmas de Python e Arduino é iniciada com um header de um byte `0xFE` indicando o início da mensagem. Como a transmissão da mensagem em si também é feita byte a byte e como estamos usando valores pequenos para ângulos em radianos e portanto trabalhando os valores em sistema de vírgula fixa, o número é multiplicado por um valor grande (1048576) ao ser enviado e dividido pelo mesmo ao ser recebido, além de usar funções de *pack* e *unpack* de ambos os lados.

3 Python / PC

No Python, o programa é referente ao processamento de imagem da câmera e o controle da posição da bola. Antes de reiniciar sua execução, é sempre importante garantir que a leitura serial e a leitura da câmera foram encerradas, para poder reabrí-las no programa. Os dois comandos a seguir são recomendados para o garantir:

```
if arduino is open then
    close serial communication
end if
if camera is open then
    close camera
    close OpenCV
end if
```

Para a captura da imagem, é utilizada a biblioteca *OpenCV*, com a resolução 640x480px. Além da câmera, todas as variáveis necessárias são iniciadas assim como a comunicação com o Arduino.

```
open serial communication
camera ← OpenCV VideoCapture
initial read ← serial read
if initial read ≠ 'S' then
    Print("Não foi possível iniciar comunicação com o Arduino.")
    close serial communication
    exit program
end if
```

Então é iniciado o ciclo de execução. Recebendo a imagem da câmera, é feita uma filtragem no espectro de cores e assim calculada a posição da bola. A partir da posição da bola, o controlador PID define o setpoint que envia ao Arduino para cada eixo da mesa. Os ganhos são os mesmos para ambos os eixos, porém os limites da atuação integral diferem.

while True **do**
  OpenCV ball search
  if ball found **then**
    $ball_x \leftarrow$ ball x-axis midpoint
    $ball_y \leftarrow$ ball y-axis midpoint
  else
    $ball_x \leftarrow goal_x$
    $ball_y \leftarrow goal_y$
  end if

  past error $\leftarrow$ error
  error $\leftarrow$ goal position - ball position
  D $\leftarrow$ error - past error
  I $\leftarrow$ I + error
  I limited between $-I_{max}$ and I_{max}

  plate axis setpoint $\leftarrow k_P$ error + k_I I + k_D D
  plate axis setpoint variation limited to $\pm 0,05 rad$ every cicle
  serial write(0xFE)
  serial write(plate axis setpoint)

  if ball found **then**
    Print(data)
  end if
  show ball image

  if keyboard input = 'q' **then**
    break while loop
  end if
end while

O setpoint enviado para o Arduino é limitado a uma variação de $0,05 rad$ por ciclo para assim gerar uma trajetória para o controlador de ângulos da mesa. Isso foi implementado pois foi possível observar experimentalmente que a atuação do controlador fica menos precisa com mudanças mais bruscas de setpoints.

Para encontrar a bola, a imagem é trabalhada no OpenCV, a levando para o espaço de cores HSV e a filtrando de acordo com limiares medidos para cada elemento de cor encontrados na bola. É gerada assim uma máscara da imagem com apenas os pontos da bola.

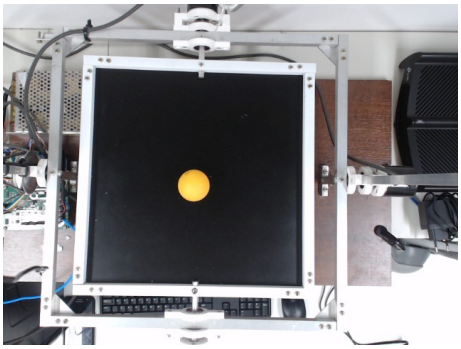

Figura 9: Imagem inicial capturada pela câmera.

Limiares referentes a	Limiar mínimo	Limiar máximo
H	15	40
S	90	240
V	150	255

Tabela 3: Limiares HSV utilizados para a bola laranja.

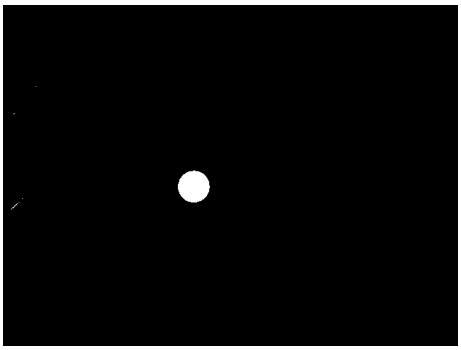

Figura 10: Máscara do filtro no espaço HSV.

Após fazer a operação lógica & entre a máscara e a imagem da câmera bit a bit, possuímos uma imagem com apenas a bola.

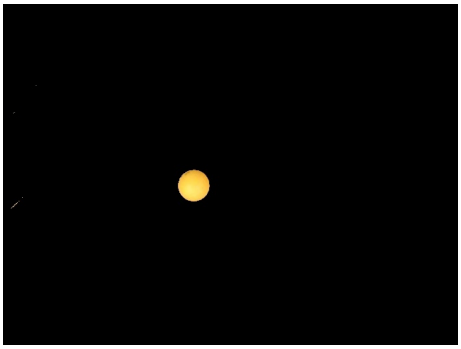

Figura 11: Imagem com apenas a bola.

Após passar essa imagem para escala de cinza e mais um passo de filtragem, é calculada a posição do centro da bola e suas coordenadas em x e y. Novamente um PID controla a posição, se comunicando com o Arduino pela sua porta serial (utilizando a biblioteca *serial*) e enviando para ele os setpoints de inclinação necessários para centralizá-la. Caso a bola não seja encontrada (ou seja, o número de pontos na imagem da bola for muito pequeno), os setpoints enviados são os que equilibram a mesa paralela ao chão.

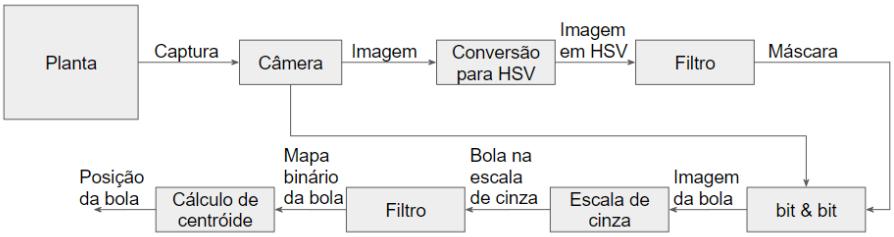

Figura 12: Diagrama do processamento de imagem.

As posições da bola ao longo da execução são salvas em uma matriz e exibidas no terminal. O mapeamento salvo de sua posição pode ser visto após interromper o programa e os dados ficam salvos em uma planilha.

5 Resultados experimentais

Dado que temos dois ciclos de controle, um para o ângulo da mesa e outro para o posicionamento da bola, faz sentido analisar o controlador de atuação direta primeiro. A seguir vemos os testes relativos à mesa, em que um setpoint senoidal era enviado para ambos os eixos e no qual analisamos o resultado real.

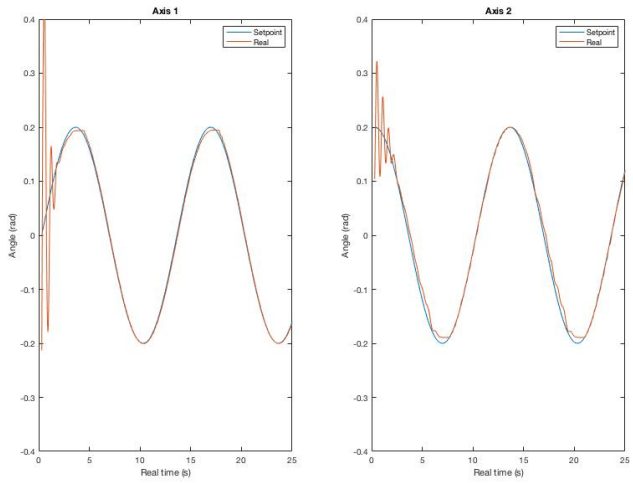

Figura 13: Gráficos com setpoint enviado e ângulo medido em radianos para cada eixo.

Vemos um pequeno undershoot no resultado real (entre $0,2 \times 10^{-3}$ e $10,9 \times 10^{-3}$ radianos), tendendo especialmente para um dos lados em cada eixo, além de um leve atraso (de $0,06$ a $0,44$ segundos) dentro do esperado pela natureza do controlador PID.

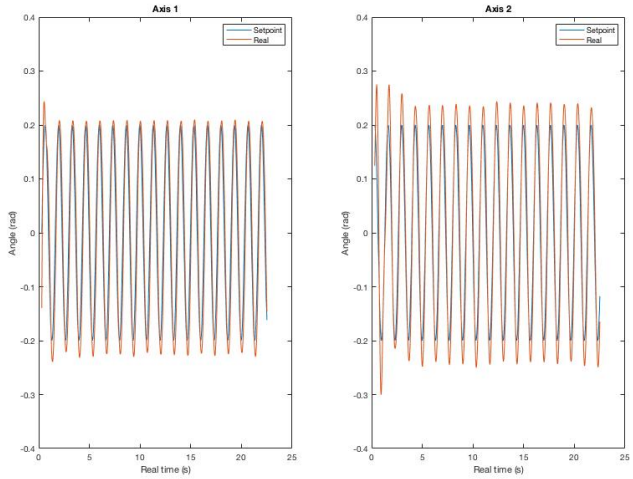

Figura 14: Setpoint enviado e ângulo medido com oscilações mais rápidas (frequência maior).

Já com aumento da frequência, passamos a observar um overshoot (de até 44×10^{-3} radianos), principalmente para o lado oposto do undershoot visto anteriormente, porém com uma menor instabilidade no começo da execução. O atraso porém, fica menor (no máximo 0,07 segundos).

O erro tanto no undershoot de movimentos mais devagares quanto no overshoot de mudanças de setpoint mais radicais fica dentro de uma escala aceitável para o experimento envolvendo a bola.

Tendo então o controlador da posição da mesa já sintonizado e com resposta com pouco erro em relação ao setpoint enviado, partimos para o controle da posição da bola.

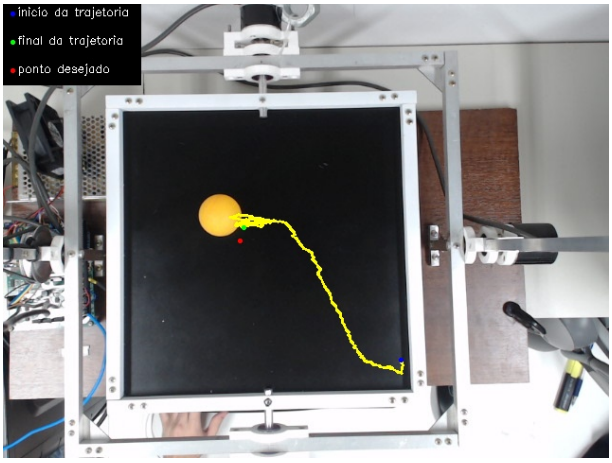

Figura 15: Foto da visão da câmera ao final de uma execução, com a trajetória executada.

Podemos visualizar os resultados em cada eixo visando uma melhor análise da performance do controlador. Sendo o eixo 1 o horizontal na imagem acima e o eixo 2 o vertical:

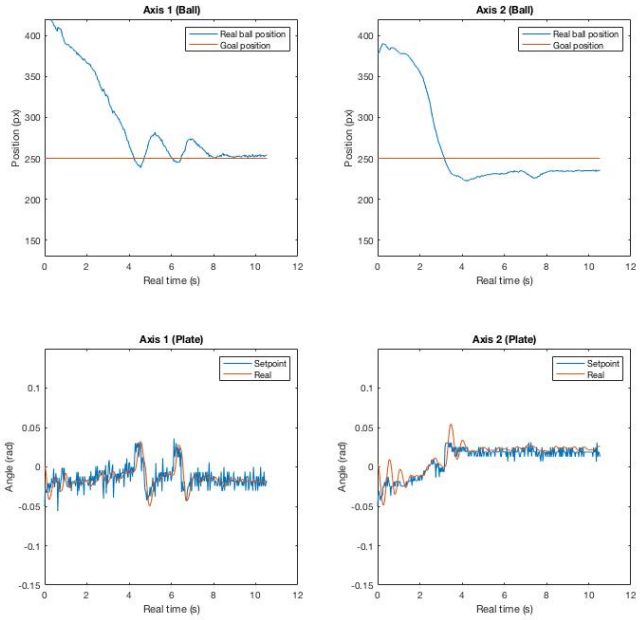

Figura 16: Gráficos da trajetória da bola e do ângulo da mesa em cada eixo.

É possível observar que o tempo de estabilização de ambos os eixos é similar (por volta de 8 segundos), tendo oscilações no primeiro eixo (ação do componente derivativo) e ajuste de erro no segundo (ação do componente integral). Observamos também um pequeno erro estacionário no segundo eixo (16 pixels).

Dado que o resultado do controle da posição da bola foi bom o suficiente, foi possível fazer também um teste de execução de trajetória da bola. Mudando o setpoint de um ponto fixo para uma série de pontos em movimento circular.

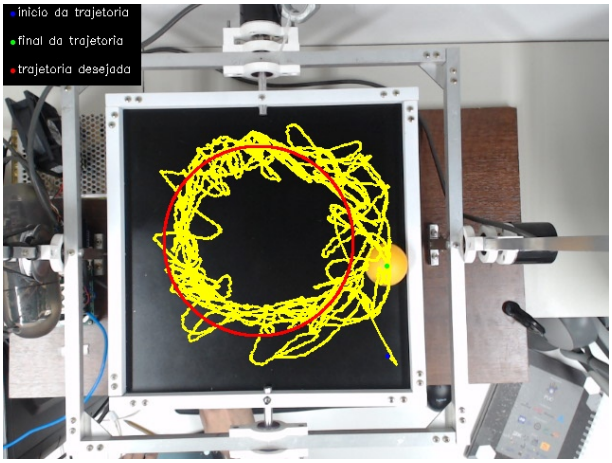

Figura 17: Foto da visão da câmera ao final de uma execução, com as trajetórias desejada e executada.

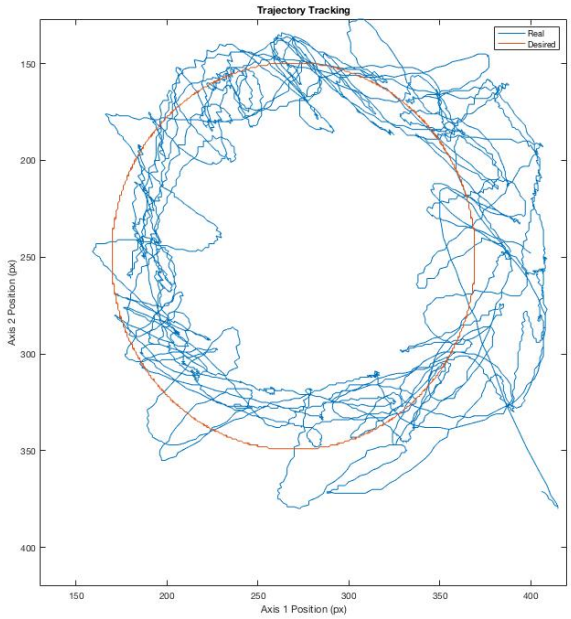

Figura 18: Gráfico da trajetória sobre o plano $\mathbb{R}^2$.

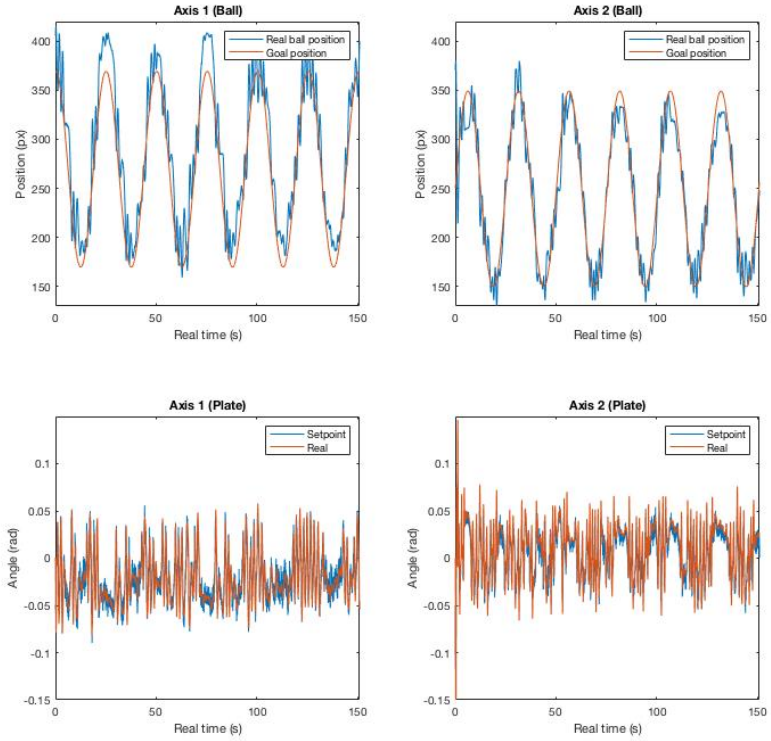

Figura 19: Gráficos da trajetória da bola e do ângulo da mesa em cada eixo.

No gráfico dos eixos individuais é possível ver que apesar de ter alguma oscilação, o controle da posição da bola acompanha a trajetória desejada bem aproximadamente. A média do erro absoluto em regime permanente é de 22,45px no eixo 1 e 13,25px no eixo 2, erros pequenos levando em consideração uma imagem de 640x480px.

6 Conclusão

Fica evidente portanto que é possível controlar este sistema. Mesmo com o controlador PID não sendo ideal para execução de controle de trajetórias, foi possível obter um resultado razoavelmente satisfatório. Estima-se que resultados ainda melhores podem ser alcançados com um controlador PID com ganhos adaptativos e outros tipos de controles mais apropriados ao modelo da planta.

Referências

Andrews, G., Colasuonno, C. and Herrmann, A.*Ball on plate balancing system, Technical report*. Rensselaer Polytechnic Institute, 2004.

Ogata, K. *Engenharia de Controle Moderno*. Rio de Janeiro: Prentice Hall, 2004.

Murphy, R. *Introduction to AI Robotics*. The MIT Press, 2000.

Zachi, A. R. L., Riveros, S. R. D., Lima, P. P. P., Gouvêa, J. A., Leite, A. C. *Visual Servoing Stabilization of a Ball and Plate Mechanism by Using an Enhanced Disturbance Rejection Control Method*. COBEM, 2017.