

Universidade Federal de Minas Gerais  
Escola de Engenharia  
Curso de Graduação em Engenharia de Controle e Automação

## **Controle de um dispositivo *Ball and Plate* por meio de visão computacional**

Hugo Ferreira Marques

Orientador: Prof. Luciano Antônio Frezzatto Santos, Dr.

Belo Horizonte, Dezembro de 2023



## **Monografia**

### **Controle de um dispositivo *Ball and Plate* por meio de visão computacional**

Monografia submetida à banca examinadora designada pelo Colegiado Didático do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Minas Gerais, como parte dos requisitos para aprovação na disciplina Projeto Final de Curso II.

Belo Horizonte, Dezembro de 2023

# Resumo

O presente trabalho visa o controle de um sistema *Ball and Plate* por meio de visão computacional. Este sistema envolve uma superfície que pode ser rotacionada nos eixos  $x$  e  $y$ , em que se utiliza três servos motores para atingir a inclinação desejada, juntamente com uma bola que pode se movimentar livremente pela superfície. O objetivo principal é controlar a bola por meio de rotações na superfície, visando posicioná-la em uma referência pré-determinada. Para atingir esse objetivo, quatro temas principais são abordados. Em primeiro lugar, a visão computacional é empregada para determinar a posição da superfície e, posteriormente, a posição da bola. A partir disso, a cinemática inversa é utilizada para converter o ângulo desejado da superfície em angulações específicas para os servos motores. A modelagem do sistema, fundamentada na mecânica lagrangiana, é empregada para obter a planta do sistema. Por fim, o controle é realizado por meio de dois controladores PIDs, os quais são responsáveis pelo gerenciamento da posição da bola sobre a superfície, ajustando a inclinação da mesa para controlar o movimento da bola. Os resultados obtidos indicam êxito na construção do sistema, bem como na implementação de todos os recursos utilizados. No entanto, são apresentadas propostas de melhorias que visam aprimorar a robustez e precisão do sistema.

**Palavras-chaves:** *Ball and Plate*, Visão Computacional, OpenCV, Cinemática Inversa, Arduino, Modelagem em Caixa Branca.

# Abstract

This paper aims at the control of a *Ball and Plate* system through computer vision. This system involves a surface that can be rotated in the  $x$  and  $y$  axes, using three servo motors to achieve the desired inclination, along with a ball that can move freely on the surface. The main objective is to control the ball through rotations on the surface, aiming to position it at a predetermined reference. To achieve this goal, four main topics are addressed. Firstly, computer vision is employed to determine the position of the surface and subsequently the position of the ball. From this, inverse kinematics is used to convert the desired angle of the surface into specific angles for the servo motors. System modeling, based on Lagrangian mechanics, is employed to obtain the system's plant. Finally, control is carried out through two PID controllers, which are responsible for managing the position of the ball on the surface, adjusting the table inclination to control the ball's movement. The results obtained indicate success in building the system, as well as in implementing all the resources used. However, proposals for improvements are presented to enhance the robustness and precision of the system.

**Keywords:** *Ball and Plate*, Computer Vision, OpenCV, Inverse Kinematics, Arduino, White Box Modeling.

# Agradecimentos

Agradeço profundamente aos meus pais e à minha irmã pelo constante apoio e incentivo ao longo de todos os momentos da minha vida.

Agradeço aos amigos, especialmente ao Pedro Américo e à Renata, pela amizade e apoio constantes, elementos essenciais que contribuíram significativamente à minha trajetória.

Ao Thiago, meu sincero agradecimento pelo apoio e suporte intelectual ao longo da graduação.

Ao meu orientador, Luciano Frezzatto, agradeço por sua disponibilidade em auxiliar nas dúvidas que surgiram durante o desenvolvimento deste trabalho.

Por último, estendo meus agradecimentos ao Gabriel, Italo, Lucca e Marina pelo companheirismo e amizade, especialmente durante o período da pandemia.



# Sumário

<b>Lista de ilustrações . . . . .</b>	<b>vi</b>
<b>Lista de tabelas . . . . .</b>	<b>vii</b>
<b>1      INTRODUÇÃO . . . . .</b>	<b>1</b>
<b>1.1    Aplicações . . . . .</b>	<b>3</b>
<b>1.2    Objetivos . . . . .</b>	<b>3</b>
<b>2      REVISAO BIBLIOGRÁFICA . . . . .</b>	<b>4</b>
<b>3      DESCRIÇÃO DO SISTEMA E DEFINIÇÕES . . . . .</b>	<b>7</b>
<b>3.1    Construção Física . . . . .</b>	<b>7</b>
<b>3.2    Componentes utilizados . . . . .</b>	<b>8</b>
<b>3.3    Declaração das variáveis e representação do sistema . . . . .</b>	<b>9</b>
<b>4      CINEMÁTICA INVERSA . . . . .</b>	<b>11</b>
<b>4.1    Matrizes de Rotação e Translação . . . . .</b>	<b>11</b>
<b>4.1.1    Matriz de Rotação . . . . .</b>	<b>11</b>
<b>4.1.2    Matriz de Translação . . . . .</b>	<b>12</b>
<b>4.2    Cálculo da cinemática inversa . . . . .</b>	<b>14</b>
<b>4.2.1    Cálculo da Junta <math>J_1</math> . . . . .</b>	<b>14</b>
<b>4.2.2    Cálculo da junta <math>J_3</math> . . . . .</b>	<b>14</b>
<b>4.2.3    Cálculo da Junta <math>J_2</math> . . . . .</b>	<b>16</b>
<b>4.2.4    Cálculo do ângulo de inclinação <math>\gamma</math> . . . . .</b>	<b>17</b>
<b>5      MODELAGEM DO SISTEMA . . . . .</b>	<b>18</b>
<b>5.1    Energia cinética . . . . .</b>	<b>19</b>
<b>5.2    Energia Potencial . . . . .</b>	<b>20</b>
<b>5.3    Equação Euler-Lagrange . . . . .</b>	<b>20</b>
<b>5.4    Linearização do sistema . . . . .</b>	<b>22</b>

<b>6</b>	<b>VISÃO COMPUTACIONAL</b>	<b>23</b>
<b>6.1</b>	<b>Calibração da câmera</b>	<b>23</b>
<b>6.2</b>	<b>Detecção da superfície</b>	<b>25</b>
6.2.1	Estratégia Utilizada . . . . .	25
6.2.2	Implementação . . . . .	27
<b>6.3</b>	<b>Detecção da Bola</b>	<b>29</b>
6.3.1	Filtro HSV . . . . .	29
6.3.2	Obtenção da posição da bola . . . . .	30
<b>7</b>	<b>CONTROLE</b>	<b>32</b>
<b>7.1</b>	<b>Cálculo dos ganhos</b>	<b>32</b>
<b>7.2</b>	<b>Simulação</b>	<b>34</b>
<b>8</b>	<b>RESULTADOS</b>	<b>37</b>
<b>8.1</b>	<b>Interface</b>	<b>37</b>
<b>8.2</b>	<b>Modificações físicas no sistema</b>	<b>39</b>
<b>8.3</b>	<b>Dificuldades encontradas</b>	<b>40</b>
<b>8.4</b>	<b>Resultados Experimentais</b>	<b>41</b>
<b>8.5</b>	<b>Códigos Utilizados</b>	<b>42</b>
<b>9</b>	<b>CONCLUSÕES</b>	<b>44</b>
<b>9.1</b>	<b>Propostas de Continuidade</b>	<b>45</b>
	<b>REFERÊNCIAS</b>	<b>46</b>

## Lista de ilustrações

Figura 1 – Demonstraçāo do sistema <i>Ball and Plate</i> . . . . .	1
Figura 2 – Exemplo de um marcador do tipo ArUco . . . . .	4
Figura 3 – Representaçāo dos manipuladores paralelos . . . . .	8
Figura 4 – Exemplo de um ponto rotacionado . . . . .	10
Figura 5 – Exemplo de um sistema rotacionado . . . . .	12
Figura 6 – Exemplo de um sistema transladado . . . . .	13

Figura 7 – Demonstração do centro transladado (em vermelho) em decor- rencia de uma rotação da superfície em y de $-30^\circ$ . . . . .	15
Figura 8 – Possíveis posições da Junta $J_2$ , representada pelos pontos em ver- melho . . . . .	16
Figura 9 – Representação das distorções Tangenciais e Radiais, retirado do site Tangramvision . . . . .	24
Figura 10 – Imagem da câmera antes e depois de aplicação das correções de calibração . . . . .	25
Figura 11 – Definição visual de um quadro ChArUco, retirado da documen- tação da biblioteca OpenCV . . . . .	26
Figura 12 – Área impressa para detecção da superfície . . . . .	27
Figura 13 – Superfície detectada, junto com seu eixo desenhado . . . . .	28
Figura 14 – Representação da correção de inclinação, para transformar um problema de três dimensões, para duas dimensões . . . . .	28
Figura 15 – Representação visual do espaço HSV . . . . .	29
Figura 16 – Comparação da máscara original e após a aplicação das funções de <i>erode</i> e <i>dilate</i> . . . . .	30
Figura 17 – Representação do Lugar das Raízes . . . . .	33
Figura 18 – Resposta ao degrau da planta Linearizada . . . . .	34
Figura 19 – Visualização do modelo implementado no Simulink . . . . .	35
Figura 20 – Visualização do bloco “Planta Não-linearizada”do modelo imple- mentado no Simulink . . . . .	35
Figura 21 – Simulação da planta não linearizada, realizada no Simulink . . . .	36
Figura 22 – Tela para edição dos parâmetros de detecção da bola . . . . .	38
Figura 23 – Tela para exibição da Perspectiva . . . . .	38
Figura 24 – Tela para editar a inclinação da superfície . . . . .	39
Figura 25 – Tela para habilitar o PID e seus ganhos . . . . .	39
Figura 26 – Controle do sistema após a aplicação de uma perturbação na bola ao longo do eixo $x$ . . . . .	41
Figura 27 – Controle do sistema após a aplicação de uma perturbação na bola ao longo do eixo $y$ . . . . .	42

## Listas de tabelas

Tabela 1 – Tabela das variáveis essenciais para um sistema *Ball and Plate* . 9

# 1 Introdução

O sistema *Ball and Plate* é um dispositivo mecânico composto por uma placa que pode ser inclinada em duas direções perpendiculares, com o objetivo de controlar a posição de uma bola que se movimenta livremente pela superfície. Embora sua implementação seja extensamente discutida na literatura, uma variedade de abordagens tem sido explorada, incluindo o uso de motores de passo, motores servo e diferentes disposições, como plataformas de Stewart com 6 motores, bem como configurações mais simplificadas de apenas 2 eixos e outros tipos de manipuladores.

Este trabalho representa uma tentativa de aprimorar um sistema já existente, desenvolvido por Ferreira (2022), concentrando-se em uma nova implementação da cinemática inversa, além de diferentes estratégias de visão computacional e controle. O sistema em questão é composto por três motores do tipo servo que se conectam à plataforma por meio de três manipuladores paralelos, possuindo, portanto, três graus de liberdade. Cada manipulador possui duas juntas rotacionais e uma junta esférica, e o sistema como um todo pode ser observado na Figura 1:

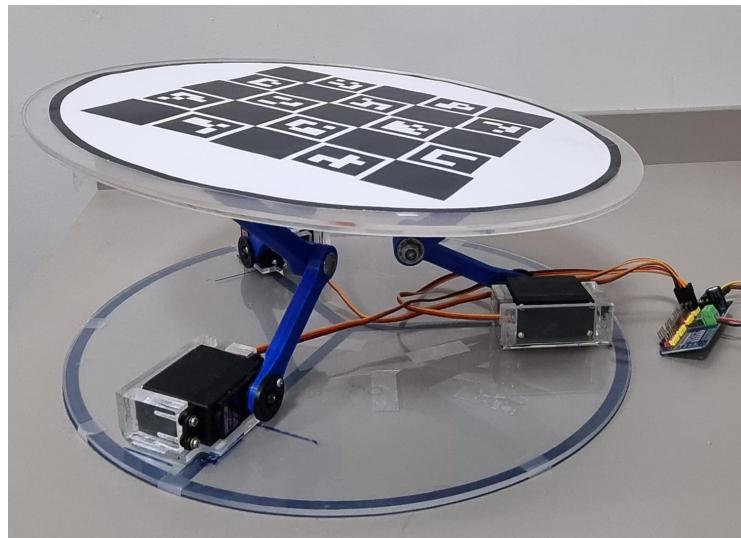


Figura 1 – Demonstração do sistema *Ball and Plate*

Na etapa de cinemática inversa, várias propriedades de matrizes de rotação e translação serão empregadas para calcular as posições das juntas do sistema a partir

de uma posição de inclinação desejada. Essa etapa é fundamental, pois determina a inclinação necessária dos servos para atingir a inclinação desejada da superfície.

Quanto à visão computacional, uma *webcam* é utilizada para capturar imagens em conjunto com marcadores do tipo ChArUco. Estes marcadores são cruciais para identificar tanto a posição relativa da superfície quanto a localização precisa da bola sobre ela. Para isso, a biblioteca OpenCV em Python é empregada. Inicialmente, a câmera é calibrada para obter os parâmetros de distorção, essenciais para a precisão das medições. A identificação dos marcadores é realizada por funções específicas da biblioteca, enquanto a posição da bola é determinada principalmente por meio de filtros de cor HSV (*Hue Saturation Value*).

Para o controle, será utilizado o tipo PID (Proporcional Integrativo Derivativo) com a finalidade de levar a bola ao referencial desejado. Basicamente, este tipo de controle é uma abordagem clássica que combina três componentes principais: proporcional, integrativo e derivativo. O componente proporcional ajusta a saída com base no erro atual, já o componente integrativo lida com acumulação de erros ao longo do tempo, enquanto o componente derivativo previne oscilações excessivas ajustando a resposta com base na taxa de variação do erro.

A realização deste projeto é motivada pela oportunidade de aplicar e reforçar os conhecimentos fundamentais adquiridos durante a graduação em Engenharia e Controle e Automação.

A Visão Computacional é crucial para detectar a bola e a superfície. A interação entre Eletrônica e Programação se destaca na montagem do circuito, comunicação computador-microcontrolador, programação e criação da GUI (*Graphical User Interface*). Os assuntos abordados em Fundamentos de Mecânica, Modelagem e Simulação, e Manipuladores Robóticos são fundamentais para a modelagem, proporcionando a base para compreender a dinâmica e realizar operações de cinemática inversa no sistema *Ball and Plate*. Os conceitos de Engenharia de Controle e Controle Digital são essenciais para implementar estratégias como o PID, permitindo ajuste preciso e equilíbrio da bola sobre a superfície.

Com a utilização dessa integração multidisciplinar, torna-se relevante considerar as aplicações práticas do sistema, às quais serão exploradas na próxima seção.

## 1.1 Aplicações

O sistema como um todo possui aplicações dedicadas apenas para fins didáticos, podendo ser utilizado, por exemplo, nas disciplinas práticas de Controle. Contudo, ao analisar o sistema de forma isolada, existem diversas aplicações para cada área.

Em relação à cinemática inversa e definição de suas inclinações, o sistema é amplamente utilizado em simuladores de carros e aviões. A precisão na determinação da angulação da superfície permite simular movimentos realistas, como curvas em um ambiente virtual, por exemplo.

Voltado à visão computacional, a utilização dos marcadores fiduciais é amplamente utilizado em ambientes industriais para calibração precisa de câmeras e para estimar o posicionamento de certos componentes. Já a detecção de objetos com filtros HSV é amplamente utilizada para identificação de cores específicas de semáforos, usada em veículos autônomos.

## 1.2 Objetivos

Este projeto possui os seguintes objetivos:

- Aprimorar o sistema já existente, tentando corrigir os problemas apresentados anteriormente;
- Implementar uma forma de visão computacional em que a posição da câmera não seja um fator que necessite de um alinhamento preciso;
- Desenvolver uma forma de cinemática inversa simples de ser implementada e calculada;
- Aplicar uma estratégia de controle do tipo PID, por meio dos itens anteriores, que seja capaz de manter a bola equilibrada em uma referência determinada;

## 2 Revisão Bibliográfica

Em Poroykov *et al.* (2020), é feito um estudo comparativo de diversos marcadores do tipo ArUco (Augmented Reality University of Cordoba). ArUco é um tipo fiducial de marcadores utilizados para detecção e rastreamento desses elementos em visão computacional e realidade aumentada, servindo também para calibração da câmera. Esse tipo de marcador consiste em um padrão de imagem em 2D, geralmente em preto, contendo pequenos dados, representados por quadrados em branco que podem ser interpretados como identificadores. Sua composição consiste em uma grade binária de 7x7 (Cada bit representado por um quadrado branco) em que o entorno dessa grade necessariamente precisa ser preto, como pode ser exemplificado na Figura 2:

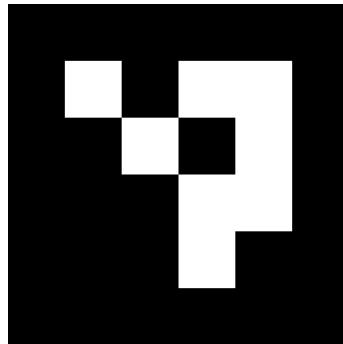


Figura 2 – Exemplo de um marcador do tipo ArUco

O estudo faz uma abordagem breve citando que existem diferentes tipos de marcadores fiduciais e cita o ArUco como um dos melhores modelos e mais utilizados dentre eles. O artigo também reporta teste utilizando ArUcos de diferentes tamanhos e em diferentes angulações, para verificar se a biblioteca gráfica utilizada (OpenCV) consegue detectar esses marcadores com precisão. Não surpreendentemente, quanto maior o ArUco e menos rotacionado ele está em relação à câmera, maior sua precisão e confiabilidade. Como pontos positivos deste artigo, têm-se a abordagem rápida e objetiva, além de uma fácil compreensão do que está sendo discutido. Seria interessante se este estudo abordasse também os diferentes tipos de marcadores, como o AprilTag e STag, além de outras técnicas, não somente dos

marcadores independentes, mas a utilização deles em conjunto, como por exemplo o ChArUco, que são vários ArUcos lado a lado e oferecem uma precisão melhor, sendo muito utilizados para calibração de câmeras e uma estimativa melhor da posição e orientação de um objeto.

A modelagem detalhada de um sistema *Ball and Plate*, utilizando mecânica lagrangiana, é abordada em Nokhbeh e Khashabi (2011). O autor realiza uma linearização assumindo que a superfície que suporta a bola, inclina-se muito pouco e tem aceleração baixa, o que torna essas hipóteses bem próximas da realidade e que permitem que a linearização não perca muita informação. Com isso, o sistema linearizado se torna desacoplado, dependente somente da aceleração da bola, em  $x$  e  $y$ , e da inclinação da superfície, nos mesmos eixos, o que o torna muito mais simples de ser controlado. Além de fazer a modelagem do sistema *Ball and Plate*, o autor também faz a modelagem do motor, uma vez que um motor CC (Corrente Contínua) será utilizado. Finalmente, para a parte de modelagem, ele realiza a transformação do sistema para o espaço de estados.

Como o sistema resultante envolve a dinâmica do *Ball and Plate* e do motor, é necessário criar um controle em cascata, em que a malha do motor é a interna (com dinâmica mais rápida) e a malha do *Ball and Plate* é a externa (mais lenta). O autor não cita qual foi a estratégia utilizada para o controle e, além disso, ele faz uma tentativa de controle para a malha interna e três tentativas para a externa. Apesar de obter resultados satisfatórios, o controlador obtido possui uma ordem elevada, chegando a possuir até 8 zeros e 8 polos, o que torna sua aplicação complexa.

Em Rodrigues *et al.* (2004) é abordado uma implementação de um controlador neuro-fuzzy para um sistema ball and beam. Este sistema é parecido com o *Ball and Plate*, porém a bola se move apenas em uma direção, sendo assim uma simplificação desta monografia. O estudo discute sobre a implementação de um modelo NEFCON (Neuro-Fuzzy Control), que é um controlador neuro-fuzzy baseado em uma rede perceptron de três camadas, utilizando portanto *backpropagation*. Nessa abordagem, a primeira camada possui os valores de entrada de erro e o delta do erro, a segunda camada representa a base de regras do sistema fuzzy e a terceira camada equivalente às funções de pertinência de saída. Com isso, o autor realiza os experimentos, tanto com regras fuzzy inferidas empiricamente quanto utilizando o método NEFCON, obtendo neste um resultado bem melhor, em que a instabilidade e o erro diminuem

significativamente.

Em Castro *et al.* (2019) é discutido um problema semelhante ao desenvolvido nesta monografia, porém é utilizado apenas 2 servo motores. O artigo apresenta uma modelagem completa baseada na mecânica lagrangiana e, também, uma modelagem em espaço de estados. Para realizar o controle do dispositivo, os autores, em Castro *et al.* (2019), optaram por um controlador do tipo LQG (*Linear Quadratic Gaussian*), cuja abordagem combina dois elementos principais: um controlador LQR (*Linear Quadratic Regulator*) e um observador de estado Kalman. Entre os pontos positivos do trabalho, destaca-se a modelagem lagrangiana feita de forma clara e de fácil compreensão, além da modelagem em espaço de estados. Embora o controlador utilizado tenha mostrado resultados satisfatórios para o sistema, seria interessante a comparação com outros tipos de controladores, para verificar se a abordagem utilizada foi a melhor dentre elas.

# 3 Descrição do sistema e Definições

Este capítulo estabelece uma base sólida para a compreensão aprofundada do sistema *Ball and Plate*, proporcionando uma explicação detalhada sobre os componentes do sistema, suas variáveis e valores, além de apresentar uma representação visual do sistema.

## 3.1 Construção Física

O sistema físico consiste em quatro componentes principais: a base, os manipuladores paralelos, a superfície e a câmera.

A base é uma superfície em acrílico, de raio de 30 cm e espessura 2 mm, em que os três servomotores estão instalados a 120° entre si, de forma que o sistema consiga se equilibrar sem dificuldades. Os servomotores possuem uma distância de 10 cm do centro e foram acoplados à base através de uma montagem em acrílico e colados usando uma fita VHB de alta resistência e durabilidade.

Para conexão da base com a superfície, foram utilizados manipuladores paralelos. Manipulador paralelo é um sistema robótico que possui uma cadeia cinemática de forma a ligar, nesse caso, o servomotor à superfície. Para isso, utilizou-se manipuladores na configuração 3RRS (3 juntas em que R e S significam, respectivamente, Rotacional e Esférico), com 2 juntas rotacionais (localizadas no servo e na junta intermediária) e uma junta esférica (localizada na extremidade, conectada à superfície) feita de neodímio, um material ferromagnético. O elo entre as 2 juntas rotacionais possui 7 cm e o elo entre a junta intermediária e a junta esférica possui 8.8 cm. Essa configuração permite que as juntas se rotacionem somente em torno dos eixos de seus motores. A Figura 3 ilustra a configuração RRS utilizada do manipulador paralelo.

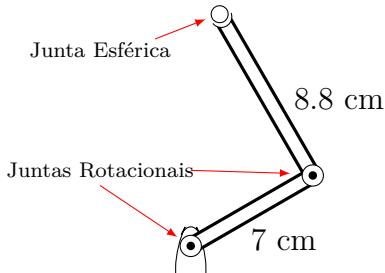


Figura 3 – Representação dos manipuladores paralelos

A superfície conectada pela base através dos manipuladores paralelos também é feita de acrílico e possui raio 30 cm e 2 mm de espessura. Acoplada a ela, está uma impressão dos ChArUcos, que serão melhor explicados no Capítulo 6, usados para detecção da superfície. Na parte inferior da superfície, estão posicionados três ímãs na mesma posição dos servomotores, ou seja, equidistantes a 120° entre si e a 10 cm do centro. Esses ímãs servem para conectar os ímãs esféricos do manipulador paralelo à superfície.

Para a realização da visão computacional, utilizou-se uma *webcam* da marca Logitech, modelo C270. Essa *webcam* possui uma resolução de 720p e é capaz de gravar em até 30 *frames* por segundo. Para fixar a câmera, foi utilizado um pedestal de microfone da Marca SYANG, fixado em uma mesa. Este pedestal possui um certo grau de liberdade para posicionar a câmera em distintas configurações.

## 3.2 Componentes utilizados

Além dos componentes descritos na construção física, também foram utilizados os seguintes componentes:

- 3 servomotores modelo MG996R de alto torque;
- Arduino Uno, para obter a angulação requerida para os servomotores via comunicação Serial e enviar as informações para o Módulo PWM;
- Módulo PWM PCA9685 para fornecer o pulso PWM de forma separada, uma vez que possui uma fonte externa e, assim, protege o Arduino. Sua comunicação é no formato I2C;

- Computador com Windows para executar toda a programação, incluindo a visão computacional, a cinemática inversa e o controle. Ao obter os ângulos requeridos, o computador envia as informações de ângulo dos servomotores através de comunicação serial.

### 3.3 Declaração das variáveis e representação do sistema

De forma a representar melhor o que foi descrito acima, nesta seção serão apresentadas as variáveis fundamentais do sistema, apresentadas de maneira organizada em uma tabela, juntamente com uma representação visual, como pode ser observado na Tabela 1 e na Figura 4

Variável	Descrição	Valor
$b$	Base, assumindo a altura inicial como a altura dos servos	$(x, y, 0)$
$s$	Superfície	$(x, y, z)$
$\mathcal{O}_{b,s}$	Origem da base e da superfície	$\mathcal{O}_b = (0, 0, 0)$ $\mathcal{O}_s = (0, 0, z)$
$r_{b,s}$	Raio da base e da superfície	$r_b = r_p = 15 \text{ cm}$
$d_{b1, 2, 3}$	Distância do eixo do motor (Junta J1) para a Origem $\mathcal{O}_b$	$d_{b1, 2, 3} = 10 \text{ cm}$
$d_{s1, 2, 3}$	Distância das juntas J3 para a Origem $\mathcal{O}_s$	$d_{s1, 2, 3} = 11 \text{ cm}$
$S_{1,2,3}$	Servos, distanciados $120^\circ$ entre si	
$J_{1,2,3}$	Juntas do manipulador paralelo Ex.: $J_{1s_1} \rightarrow$ Junta 1 do Servo 1	
$l_{1,2}$	Links(elo) entre as juntas, sendo o $l_1$ o elo entre $J_1$ e $J_2$ e $l_2$ o elo entre $J_2$ e $J_3$	$l_1 = 7 \text{ cm}$ $l_2 = 8.8 \text{ cm}$
$\alpha$	Ângulo de inclinação do eixo x da plataforma	$-20 \leq \alpha \leq 20$
$\beta$	Ângulo de inclinação do eixo y da plataforma	$-20 \leq \beta \leq 20$
$p_{x,y}$	Posição, em x e y, da bola na superfície	$(p_x, p_y)$
Váriaveis da modelagem		
$m_b$	Massa da bola	0.13 kg
$r_b$	Raio da bola	0.03 m
$I_b$	Momento de inércia da bola, assumindo uma bola oca	$\frac{2}{3}m_b r_b^2$

Tabela 1 – Tabela das variáveis essenciais para um sistema *Ball and Plate*

A imagem abaixo representa uma configuração inicial da superfície do sistema com ângulos de inclinação nulos, ou seja,  $\alpha = 0$  e  $\beta = 0$ . Nesse estado, a superfície plana serve como ponto de partida para a compreensão das variáveis do sistema descritas na Tabela 1.

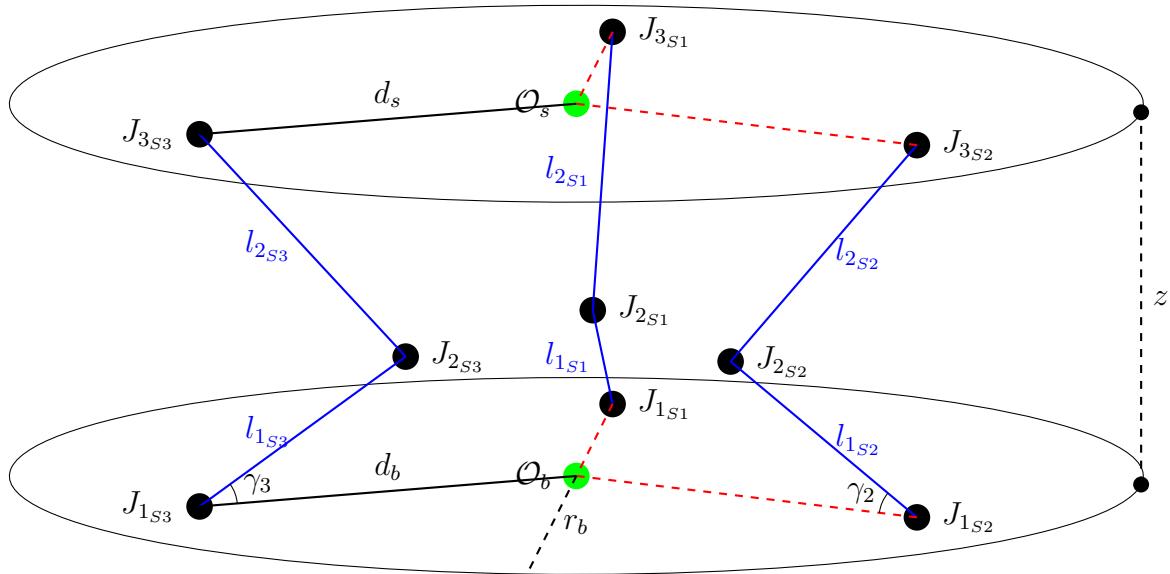


Figura 4 – Exemplo de um ponto rotacionado

## 4 Cinemática Inversa

Devido à utilização de manipuladores paralelos, são necessários cálculos que convertam a inclinação da superfície para as inclinações do servo, ou seja, deseja-se uma função que tenha como entradas os ângulos de inclinação  $\alpha$  e  $\beta$  e que retorne os ângulos de inclinação desejados  $\gamma_1, \gamma_2, \gamma_3$ :

$$f(\alpha, \beta) = (\gamma_1, \gamma_2, \gamma_3) \quad (4.1)$$

Para atingir esse objetivo, serão aplicadas manipulações matriciais e operações algébricas para determinar as inclinações desejadas dos servos.

### 4.1 Matrizes de Rotação e Translação

Para o desenvolvimento da cinemática inversa, será explorada a aplicação de diversas técnicas de manipulação de pontos e coordenadas. Para isso, serão utilizadas matrizes de rotação e translação, que serão explicadas nas subseções subsequentes.

#### 4.1.1 Matriz de Rotação

Matrizes de rotação, para o caso em 3 dimensões, são matrizes de tamanho  $3 \times 3$  que têm a finalidade de rotacionar um ponto em torno de um eixo fixo, sem alterar sua magnitude. Representado na Figura 5, é possível visualizar uma rotação no eixo  $x$ .

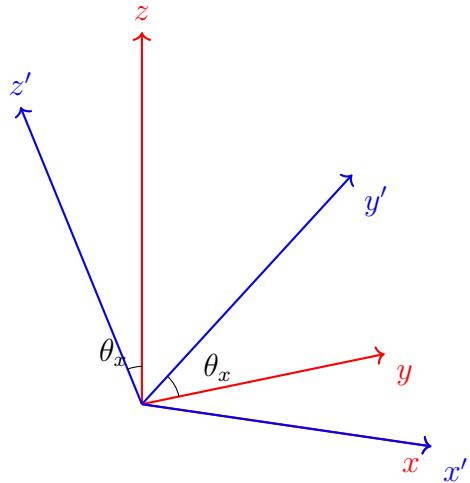


Figura 5 – Exemplo de um sistema rotacionado

As matrizes de rotação de cada eixo podem ser representadas através das seguintes matrizes:

$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \quad (4.2)$$

$$R_y(\theta_y) = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \quad (4.3)$$

$$R_z(\theta_z) = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.4)$$

Para a aplicação das matrizes de rotação é necessário multiplicá-las e por fim multiplicar pelo ponto (um vetor  $3 \times 1$ ) no qual se deseja aplicar a rotação.

#### 4.1.2 Matriz de Translação

Matrizes de translação em 3 dimensões são matrizes de tamanho  $4 \times 4$  que possuem a finalidade de deslocar objetos no espaço tridimensional, como pode ser observado na Figura 6.

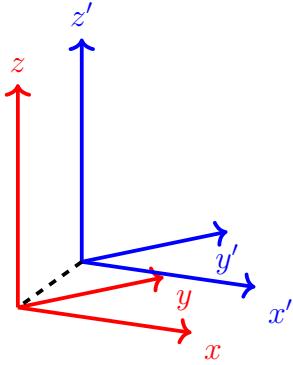


Figura 6 – Exemplo de um sistema transladado

A matriz de translação pode ser representada pela matriz da Equação (4.5), em que  $T_x$ ,  $T_y$  e  $T_z$  são as quantidades que se deseja transladar o ponto nas direções  $x$ ,  $y$  e  $z$ , respectivamente.

$$T(T_x, T_y, T_z) = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.5)$$

É necessário fazer o produto TP, em que P é o vetor representando um ponto no espaço, com uma dimensão adicional (um elemento a mais com valor 1):

$$P = [P_x \ P_y \ P_z \ 1]^T \quad (4.6)$$

Também é possível representar a translação por meio de um vetor  $3 \times 1$  em que o resultado da translação se dá pela soma do ponto com a quantidade desejada a se transladar:

$$P' = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \quad (4.7)$$

É importante destacar que é possível realizar várias manipulações em sequência, tanto de rotação quanto de translação, multiplicando cada operação em cascata e, por fim, aplicando a transformação resultante no ponto desejado. No entanto, é necessário estar ciente de que a ordem das operações de multiplicação influencia

diretamente o resultado final, uma vez que as matrizes de translação e rotação não são comutativas, o que pode resultar em uma posição diferente da configuração desejada.

## 4.2 Cálculo da cinemática inversa

Para o cálculo das inclinações dos servos desejadas, serão calculadas, primeiramente, de forma independente, a posição das juntas  $J_1$  e  $J_3$  e posteriormente, utilizar essas coordenadas obtidas para o cálculo da junta  $J_2$ .

### 4.2.1 Cálculo da Junta $J_1$

A junta  $J_1$  é a única junta em que suas coordenadas são fixas, uma vez que elas estão acopladas aos servos. Como as posições das juntas estão localizadas na mesma posição, porém rotacionadas no eixo  $z$  em  $120^\circ$  entre si, suas coordenadas ficam definidas em:

$$J_{1_{S1}} = \begin{bmatrix} 0 \\ d_b \\ 0 \end{bmatrix} \quad J_{1_{S2}} = R_z(120^\circ)J_{1_{S1}} \quad J_{1_{S3}} = R_z(240^\circ)J_{1_{S1}} \quad (4.8)$$

### 4.2.2 Cálculo da junta $J_3$

Para o cálculo da junta  $J_3$  é necessário, primeiramente, definir a altura  $z$  desejada para a superfície. Ao assumir que a superfície tenha inclinação nula ( $\alpha = \beta = 0$ ), pode-se obter as coordenadas das 3 juntas esféricas.

$$J_{3_{S1}} = \begin{bmatrix} 0 \\ d_s \\ z \end{bmatrix} \quad J_{3_{S2}} = R_z(120^\circ)J_{3_{S1}} \quad J_{3_{S3}} = R_z(240^\circ)J_{3_{S1}} \quad (4.9)$$

Ao inclinar a superfície no eixo  $x$  e  $y$ , utilizando os ângulos  $\alpha$  e  $\beta$  respectivamente, as coordenadas das juntas podem ser modificadas utilizando matrizes de rotação em  $x$  e  $y$ :

$$J_{3_{S1}}(\alpha, \beta) = R_y(\beta)R_x(\alpha)J_{3_{S1}} \quad (4.10)$$

$$J_{3S2}(\alpha, \beta) = R_y(\beta)R_x(\alpha)J_{3S2} \quad (4.11)$$

$$J_{3S3}(\alpha, \beta) = R_y(\beta)R_x(\alpha)J_{3S3} \quad (4.12)$$

Porém, ao rotacionar as 3 juntas de maneira independente, o centro da superfície também é deslocado, como mostra a Figura 7.

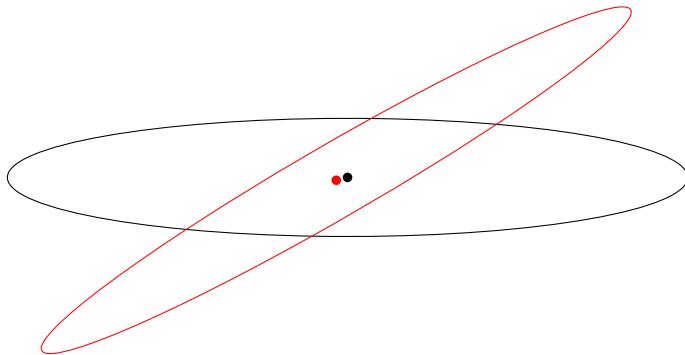


Figura 7 – Demonstração do centro transladado (em vermelho) em decorrência de uma rotação da superfície em y de  $-30^\circ$

Para manter o centro da superfície no mesmo ponto original, é calculado o quanto o centro foi transladado pela rotação provocada pelos ângulos  $\alpha$  e  $\beta$ :

$$\Delta_{\text{centro}} = \begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix} - R_y(\beta)R_x(\alpha) \begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix} \quad (4.13)$$

Com isso, todas as juntas precisam ser transladas por  $\Delta_{\text{centro}}$  para que o centro da superfície permaneça imóvel, portanto as coordenadas gerais desejadas da 3<sup>a</sup> junta podem ser definidas como

$$J_{3S1}(\alpha, \beta)' = T(\Delta_{\text{Centro}})J_{3S1}(\alpha, \beta) \quad (4.14)$$

$$J_{3S2}(\alpha, \beta)' = T(\Delta_{\text{Centro}})J_{3S2}(\alpha, \beta) \quad (4.15)$$

$$J_{3S3}(\alpha, \beta)' = T(\Delta_{\text{Centro}})J_{3S3}(\alpha, \beta) \quad (4.16)$$

### 4.2.3 Cálculo da Junta $J_2$

Ao obter a posição desejada da Junta 3 e as coordenadas fixas da Junta 1, a Junta 2 pode ser definida a partir das possíveis posições dos elos 1 e 2. Ao assumir que as juntas giram em torno somente do eixo de seus motores, uma vez que o manipulador possui 2 juntas rotacionais e 1 esférica em sua ponta, pode-se representar as possíveis posições em 2 dimensões,  $y$  e  $z$ :

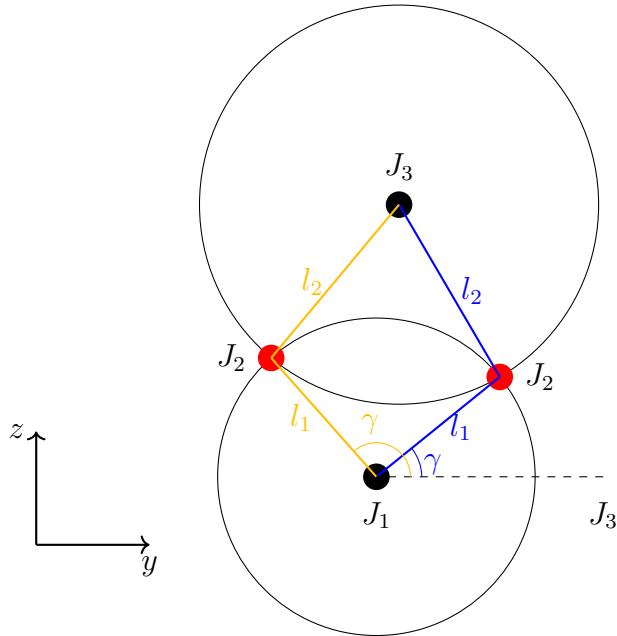


Figura 8 – Possíveis posições da Junta  $J_2$ , representada pelos pontos em vermelho

Assim, as equações dos círculos das Juntas 1 e 3 podem ser calculados como:

$$(y - y_1)^2 + (z - z_1)^2 = l_1^2 \quad (4.17)$$

$$(y - y_3)^2 + (z - z_3)^2 = l_2^2 \quad (4.18)$$

Em que  $(y_1, z_1)$  e  $(y_3, z_3)$  são as posições das juntas 1 e 3, respectivamente.

Ao subtrair (4.17) e (4.18) obtém-se a seguinte equação para  $z$ :

$$z = -\frac{(y_1 - y_3)}{(z_1 - z_3)}y - \frac{(l_1^2 - l_2^2) - (y_1^2 - y_3^2) - (z_1^2 - z_3^2)}{2(z_1 - z_3)} \quad (4.19)$$

É possível reescrever (4.19) introduzindo novas constantes  $c_z$  e  $d_z$ :

$$z = -c_z y - d_z \quad (4.20)$$

Sendo

$$c_z = \frac{(y_1 - y_3)}{(z_1 - z_3)} \quad (4.21)$$

$$d_z = \frac{(l_1^2 - l_2^2) - (y_1^2 - y_3^2) - (z_1^2 - z_3^2)}{2(z_1 - z_3)} \quad (4.22)$$

Com isso, substituindo (4.20) em (4.17), e após algumas manipulações algébricas, é possível obter a seguinte equação de segundo grau

$$(1 + c_z^2)y^2 + (-2y_1 + 2c_z d_z + 2z_1 c_z)y + (y_1^2 + d_z^2 + 2z_1 d_z + z_1^2 - r_1^2) = 0 \quad (4.23)$$

Ao obter as 2 raízes em  $y$  do sistema, é substituída a resposta em (4.20) para obter as possíveis coordenadas da junta  $J_2$  ( $y_2, z_2$ ) do sistema.

#### 4.2.4 Cálculo do ângulo de inclinação $\gamma$

Com as duas raízes em (4.23), é possível calcular seus respectivos  $\gamma$ 's por meio da seguinte relação trigonométrica:

$$\gamma_{1,2} = \sin^{-1} \left( \frac{z_2 - z_1}{l_1} \right) \quad (4.24)$$

Das 2 soluções obtidas, é escolhida aquela que o  $\gamma$  está mais próximo de zero, ou seja, a que produz um esforço menor para os servos.

Com isso, de forma resumida, para obter a inclinação desejada do sistema, deve-se seguir os seguintes passos:

1. Obter as coordenadas fixas da junta  $J_1$ , visto na Subseção 4.2.1;
2. Obter as coordenadas, dada uma inclinação em  $\alpha$  e  $\beta$  da junta  $J_3$ , visto na Subseção 4.2.2;
3. Rotacionar as coordenadas em  $z$  das 2 juntas,  $J_1$  e  $J_3$ , caso se deseje obter a inclinação dos servos 2 e 3, em  $-120^\circ$  e  $-240^\circ$  respectivamente, para trabalhar em 2 dimensões;
4. Aplicar (4.23) nas coordenadas rotacionadas das juntas  $J_1$  e  $J_3$  e obter a inclinação  $\gamma$  mais próxima de zero.

## 5 Modelagem do sistema

Antes de realizar a modelagem do sistema, assume-se que:

- A bola não irá se deslizar na superfície;
- Toda fricção será desprezada;
- A bola é simétrica e homogênea;
- Em todo o momento a bola estará em contato com a superfície;

Com isso, para a modelagem do sistema será utilizada a mecânica Lagrangiana, que relaciona a conservação de energia mecânica com o momento linear de um sistema:

$$L = T - E \quad (5.1)$$

A partir da mecânica hamiltoniana, que define que entre todos os caminhos possíveis que um sistema dinâmico tem para realizar o movimento entre dois pontos, o caminho escolhido será aquele que possui a menor diferença entre as energias cinéticas e potenciais, obtendo-se a equação de Euler-Lagrange:

$$\frac{\partial L}{\partial q_i} - \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_i} \right) = Q_i \quad (5.2)$$

Sendo:

- T: A energia cinética do sistema
- E: A energia potencial do sistema
- $q_i$ : Os graus de liberdade do sistema, sendo eles  $q_i \in \{p_x, p_y, \alpha, \beta\}$
- $Q_i$ : As forças generalizadas externas.

## 5.1 Energia cinética

A energia cinética do sistema pode ser calculada de forma independente, como a soma das energias cinéticas da bola e da superfície:

$$T = T_b + T_s \quad (5.3)$$

A energia cinética da bola é a soma da energia rotacional e translacional:

$$T_{br} = \frac{1}{2} I_b (\omega_x^2 + \omega_y^2) \quad (5.4)$$

$$T_{bt} = \frac{1}{2} m_b (v_x^2 + v_y^2) \quad (5.5)$$

sendo  $\omega$  e  $v$  as velocidades angulares e lineares para cada coordenada. É possível obter a velocidade angular em termos da linear por meio da seguinte relação:

$$v = \omega r \quad (5.6)$$

Assumindo a velocidade linear como a primeira derivada da posição, é possível reescrever as equações (5.4) e (5.5) na forma

$$T_{br} = \frac{1}{2} \frac{I_b}{r_b^2} (\dot{p}_x^2 + \dot{p}_y^2) \quad (5.7)$$

$$T_{bt} = \frac{1}{2} m_b (\dot{p}_x^2 + \dot{p}_y^2) \quad (5.8)$$

Para obter a energia cinética total da bola, somam-se os dois componentes, rotacional e translacional:

$$\begin{aligned} T_b &= T_{br} + T_{bt} \\ &= \frac{1}{2} \frac{I_b}{r_b^2} (\dot{p}_x^2 + \dot{p}_y^2) + \frac{1}{2} m_b (\dot{p}_x^2 + \dot{p}_y^2) \\ &= \frac{1}{2} \left( \frac{I_b}{r_b^2} + m_b \right) (\dot{p}_x^2 + \dot{p}_y^2) \end{aligned} \quad (5.9)$$

De forma análoga, a energia cinética da superfície é composta pela soma da energia rotacional e translacional da superfície com a bola, assumindo que a bola está em uma posição  $(p_x, p_y)$ :

$$T_{sr} = \frac{1}{2} (I_b + I_s) (\dot{\alpha}^2 + \dot{\beta}^2) \quad (5.10)$$

$$\begin{aligned} T_{st} &= \frac{1}{2}m_b(p_x\dot{\alpha} + p_y\dot{\beta})^2 \\ &= \frac{1}{2}m_b(p_x^2\dot{\alpha}^2 + 2p_x\dot{\alpha}p_y\dot{\beta} + p_y^2\dot{\beta}^2) \end{aligned} \quad (5.11)$$

Para obter a energia cinética total da superfície, somam-se os dois componentes, rotacional e translacional:

$$\begin{aligned} T_s &= T_{sr} + T_{st} \\ &= \frac{1}{2}(I_b + I_s)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2}m_b(p_x^2\dot{\alpha}^2 + 2p_x\dot{\alpha}p_y\dot{\beta} + p_y^2\dot{\beta}^2) \end{aligned} \quad (5.12)$$

Dessa maneira, a energia cinética total do sistema é a soma da energia cinética total da bola e da superfície:

$$\begin{aligned} T &= \frac{1}{2}\left(\frac{I_b}{r_b^2} + m_b\right)(\dot{p}_x^2 + \dot{p}_y^2) + \frac{1}{2}(I_b + I_s)(\dot{\alpha}^2 + \dot{\beta}^2) \\ &\quad + \frac{1}{2}m_b(p_x^2\dot{\alpha}^2 + 2p_x\dot{\alpha}p_y\dot{\beta} + p_y^2\dot{\beta}^2) \end{aligned} \quad (5.13)$$

## 5.2 Energia Potencial

A energia potencial gravitacional da bola relativa à superfície pode ser calculada como:

$$E_p = m_bgh = m_bg(p_x \sin(\alpha) + p_y \sin(\beta)) \quad (5.14)$$

## 5.3 Equação Euler-Lagrange

Ao obter todos os termos necessários para a obtenção da equação de Euler-Lagrange, deriva-se os termos em relação a todos os graus de liberdade descritos em sua definição:

$$\frac{\partial L}{\partial p_x} = m_b\dot{\alpha}(\dot{\alpha}p_x + p_y\dot{\beta}) - m_bg \sin(\alpha) \quad (5.15)$$

$$\frac{\partial L}{\partial \dot{p}_x} = \left(\frac{I_b}{r_b^2} + m_b\right)\dot{p}_x \quad (5.16)$$

$$\frac{\partial L}{\partial p_y} = m_b\dot{\beta}(\dot{\beta}p_y + p_x\dot{\alpha}) - m_bg \sin(\beta) \quad (5.17)$$

$$\frac{\partial L}{\partial \dot{p}_y} = \left( \frac{I_b}{r_b^2} + m_b \right) \dot{p}_y \quad (5.18)$$

$$\frac{\partial L}{\partial \alpha} = m_b g p_x \cos(\alpha) \quad (5.19)$$

$$\frac{\partial L}{\partial \dot{\alpha}} = (I_b + I_s) \dot{\alpha} + m_b p_x (p_x \dot{\alpha} + p_y \dot{\beta}) \quad (5.20)$$

$$\frac{\partial L}{\partial \dot{\beta}} = m_b g p_y \cos(\beta) \quad (5.21)$$

$$\frac{\partial L}{\partial \dot{\beta}} = (I_b + I_s) \dot{\alpha} + m_b p_y (p_y \dot{\alpha} + p_x \dot{\alpha}) \quad (5.22)$$

A partir das derivadas acima, é possível obter as 4 equações do sistema. Para este projeto, serão utilizadas apenas as que possuem uma relação entre o estado da bola e o estado da superfície. As outras equações mostram o efeito de um torque externo no sistema como um todo e não serão utilizadas. Com isso:

$$\frac{\partial L}{\partial p_x} - \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{p}_x} \right) = m_b \dot{\alpha} (\dot{\alpha} p_x + p_y \dot{\beta}) - m_b g \sin(\alpha) - \left( \frac{I_b}{r_b^2} + m_b \right) \ddot{p}_x \quad (5.23)$$

$$\frac{\partial L}{\partial p_y} - \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{p}_y} \right) = m_b \dot{\beta} (\dot{\beta} p_y + p_x \dot{\alpha}) - m_b g \sin(\beta) - \left( \frac{I_b}{r_b^2} + m_b \right) \ddot{p}_y \quad (5.24)$$

$$\begin{aligned} \frac{\partial L}{\partial \alpha} - \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\alpha}} \right) &= m_b g p_x \cos(\alpha) - (I_b + I_s) \ddot{\alpha} - m_b (2p_x \dot{p}_x \dot{\alpha} + p_x^2 \ddot{\alpha}) \\ &\quad - m_b (\dot{p}_x p_y \dot{\beta} + p_x \dot{p}_y \dot{\beta} + p_x p_y \ddot{\beta}) \end{aligned} \quad (5.25)$$

$$\begin{aligned} \frac{\partial L}{\partial \beta} - \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\beta}} \right) &= m_b g p_y \cos(\beta) - (I_b + I_s) \ddot{\beta} - m_b (2p_y \dot{p}_y \dot{\beta} + p_y^2 \ddot{\beta}) \\ &\quad - m_b (\dot{p}_x p_y \dot{\beta} + p_x \dot{p}_y \dot{\beta} + p_x p_y \ddot{\beta}) \end{aligned} \quad (5.26)$$

As equações (5.23) e (5.24) representam o movimento da bola na superfície e como a fricção, atrito e outros fatores externos são desprezados; o sistema pode ser considerado conservativo, em que essas duas equações podem ser igualadas a zero. Com isso é possível obter as equações em termos das acelerações  $\ddot{p}_x$  e  $\ddot{p}_y$ :

$$\ddot{p}_x = \frac{m}{m + I_b/r_b^2} \left( p_x \dot{\alpha}^2 + p_y \dot{\alpha} \dot{\beta} - g \sin(\alpha) \right) \quad (5.27)$$

$$\ddot{p}_y = \frac{m}{m + I_b/r_b^2} \left( p_y \dot{\beta}^2 + p_x \dot{\alpha} \dot{\beta} - g \sin(\beta) \right) \quad (5.28)$$

As equações (5.29) e (5.30) representam o efeito do torque externo no sistema, sendo considerado assim um sistema não conservativo. Essas equações podem ser descartadas, uma vez em que se assume que seus momentos de torque são desprezíveis.

$$\begin{aligned} \tau_x &= m_b g p_x \cos(\alpha) - (I_b + I_s) \ddot{\alpha} - m_b (2p_x \dot{p}_x \dot{\alpha} + p_x^2 \ddot{\alpha}) \\ &\quad - m_b (p_x \dot{p}_y \dot{\beta} + p_x \dot{p}_y \dot{\beta} + p_x p_y \ddot{\beta}) \end{aligned} \quad (5.29)$$

$$\begin{aligned} \tau_y &= m_b g p_y \cos(\beta) - (I_b + I_s) \ddot{\beta} - m_b (2p_y \dot{p}_y \dot{\beta} + p_y^2 \ddot{\beta}) \\ &\quad - m_b (p_x \dot{p}_y \dot{\beta} + p_x \dot{p}_y \dot{\beta} + p_x p_y \ddot{\beta}) \end{aligned} \quad (5.30)$$

O momento de Inércia de uma bola oca é dado pela expressão

$$I_b = \frac{2}{3} m r_b^2 \quad (5.31)$$

Substituindo (5.31) em (5.27) e (5.28), têm-se as equações finais não linearizadas:

$$\ddot{p}_x = \frac{3}{5} \left( p_x \dot{\alpha}^2 + p_y \dot{\alpha} \dot{\beta} - g \sin(\alpha) \right) \quad (5.32)$$

$$\ddot{p}_y = \frac{3}{5} \left( p_y \dot{\beta}^2 + p_x \dot{\alpha} \dot{\beta} - g \sin(\beta) \right) \quad (5.33)$$

## 5.4 Linearização do sistema

É possível linearizar o sistema fazendo algumas simplificações. Para isso, será considerado que:

- As inclinações em  $\alpha$  e  $\beta$  serão bastante pequenas:  $\alpha \ll 1$  e  $\beta \ll 1 \Rightarrow \sin(\alpha) \simeq \alpha$  e  $\sin(\beta) \simeq \beta$
- A velocidade dos ângulos  $\dot{\alpha}$  e  $\dot{\beta}$  serão muito pequenas, podendo seus produtos ser desprezados ( $\dot{\alpha} \ll 1$  e  $\dot{\beta} \ll 1 \Rightarrow \dot{\alpha}^2 \simeq 0$ ,  $\dot{\beta}^2 \simeq 0$ ,  $\dot{\alpha} \dot{\beta} \simeq 0$ ).

Com isso, o sistema linearizado final é dado por

$$\ddot{p}_x + \frac{3}{5} g \alpha = 0 \quad (5.34)$$

$$\ddot{p}_y + \frac{3}{5} g \beta = 0 \quad (5.35)$$

# 6 Visão computacional

Neste capítulo serão abordados os caminhos para realização da visão computacional no sistema.

A visão computacional será empregada primordialmente na detecção da bola e na determinação de sua posição em relação à superfície, sendo necessário também a detecção da própria superfície. Para desempenhar essas funções, será necessária também a calibração da câmera para que as imagens obtidas sejam o mais próximo da realidade. Tais identificações desempenham um papel importante no controle do sistema, uma vez que a posição da bola é um parâmetro de entrada para o sistema de controle, como será detalhado no Capítulo 7.

Para a realização da visão computacional será utilizada a biblioteca OpenCV, como vista em Bradski (2000), na linguagem Python.

## 6.1 Calibração da câmera

A calibração da câmera é um passo fundamental no processo de visão computacional, uma vez que toda câmera possui distorções intrínsecas inerentes à sua construção. Entre as distorções mais comuns, destacam-se as distorções tangenciais e radiais.

A distorção tangencial está relacionada à falta de alinhamento entre a lente e o sensor da câmera. Quando este fenômeno ocorre, há uma sensação de que alguns elementos estão a distâncias diferentes do que eles realmente estão. Já a distorção radial ocorre quando os raios de luz que passam pela lente são desviados em ângulos distintos do que aqueles que passam em seu centro, gerando bordas curvadas.

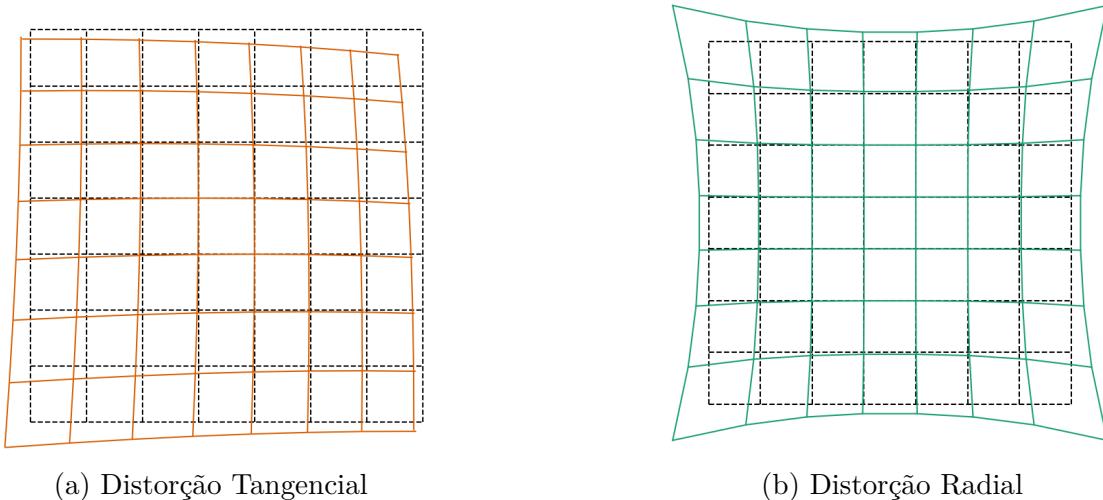


Figura 9 – Representação das distorções Tangenciais e Radiais, retirado do site Tangramvision

Ao calibrar a câmera, é possível determinar os parâmetros de distorção e criar um modelo que permite a correção das imagens posteriormente. Isso é essencial para garantir que as medidas e informações obtidas a partir das imagens sejam precisas e confiáveis, o que é crítico em aplicações de visão computacional.

O cálculo da distorção não será abordado nessa monografia, uma vez que dentro da biblioteca OpenCV já existe uma função (*calibrateCameraCharucoExtended*) que calcula seus valores. Para utilizar essa função, é necessário capturar várias fotos com a câmera que se deseja calibrar. Essas fotos devem conter um quadro do tipo CharUco (de dimensões conhecidas), que é uma combinação de um painel quadriculado com identificadores ArUco, em vários ângulos e posições diferentes.

Após a captura de todas as imagens e a utilização da função mencionada, é retornada uma série de coeficientes que são necessários para a correção da distorção. Esses coeficientes são usados em outra função da biblioteca OpenCV (*undistort*), que realiza a correção das imagens e as retornam sem as deformidades. A diferença entre uma imagem original e a imagem após a correção pode ser visualizada na Figura 10

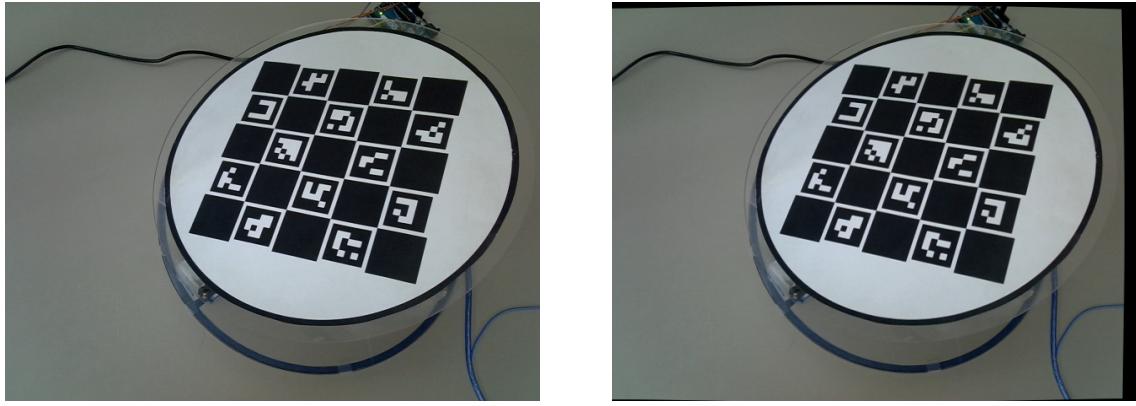


Figura 10 – Imagem da câmera antes e depois de aplicação das correções de calibração

Para facilitar esse processo, desenvolveu-se um *script* no qual, ao pressionar a tecla ‘A’, uma nova captura era realizada e armazenada no banco de dados. Após a captura das imagens desejadas, ao pressionar a tecla ‘C’, o programa inicia automaticamente o processo de calibração da câmera, utilizando todas as imagens capturadas, além de salvar os coeficientes de distorção em um arquivo *.json*, que será importado em outras aplicações.

## 6.2 Detecção da superfície

A detecção da superfície é uma etapa crítica para determinar sua posição relativa na imagem, especialmente considerando que a câmera não é fixa e pode sofrer movimentos. Essas alterações na posição da superfície na imagem podem resultar em variações no centro da imagem, o que, por sua vez, causaria mudanças indesejadas na posição da bola.

### 6.2.1 Estratégia Utilizada

Uma abordagem anteriormente considerada, conforme discutido por Ferreira (2022), envolvia o ajuste manual da área da superfície, alinhando o diâmetro da superfície com o diâmetro visível na imagem. No entanto, essa abordagem apresenta duas desvantagens: requer ajustes manuais sempre que o programa for iniciado ou

houver uma pequena alteração na posição do sistema e exige que a câmera esteja posicionada perpendicularmente ao sistema, o que pode ser desafiador de calibrar e propenso a erros que afetariam a detecção da bola.

Portanto, para esta monografia, optou-se por uma abordagem diferente na detecção da superfície. Nessa nova abordagem, serão utilizados marcadores do tipo ChArUco, que representam uma combinação de painéis quadriculados com elementos ArUco. Essa escolha é motivada pela necessidade de equilibrar precisão e facilidade de detecção. Marcadores quadriculados são altamente precisos na detecção de cantos, mas menos precisos na identificação de seu padrão global em comparação com os marcadores ArUco. Além disso, os painéis quadriculados requerem que todos os cantos estejam visíveis o tempo todo, o que seria problemático quando a bola percorre a superfície, podendo obstruir a detecção. Portanto, os detectores ChArUco combinam a facilidade de identificação dos marcadores ArUco com a alta precisão de detecção de cantos dos painéis quadriculados, tornando-os uma escolha adequada para este projeto.

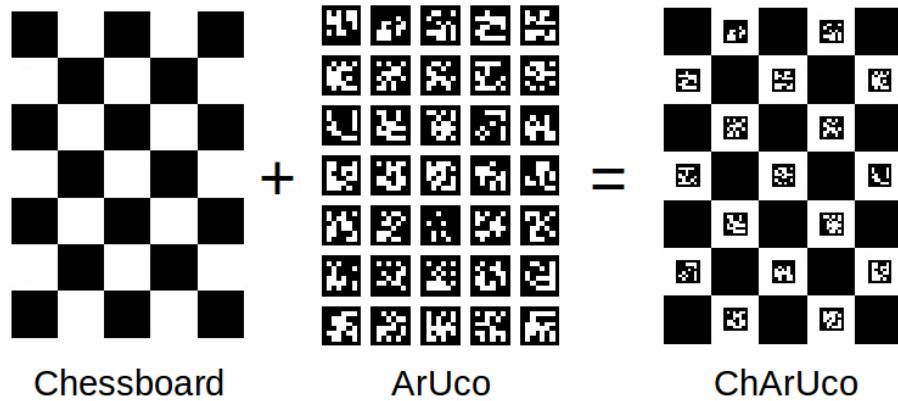


Figura 11 – Definição visual de um quadro ChArUco, retirado da documentação da biblioteca OpenCV

Essa abordagem visa automatizar a detecção tanto do raio da superfície quanto de sua inclinação, corrigindo assim quaisquer erros de posicionamento da câmera. No entanto, ela também apresenta desafios, incluindo uma maior complexidade e processamento, já que as posições do marcador precisa ser calculada. Além disso, há considerações estéticas, uma vez que o marcador ChArUco precisa ser acoplado à superfície, introduzindo elementos visuais adicionais à plataforma.

Para a implementação deste sistema, foi selecionado o uso de um quadro ChArUco 5x5, no qual os quadrados pretos medem 3.6 cm e os marcadores ArUcos inseridos nos quadrados brancos têm dimensões de 3.0 cm. Isso resulta em um quadro com dimensões totais de 18x18 cm, acompanhado por uma borda externa circular com um diâmetro de 30 cm, conforme ilustrado na Figura 12. É importante destacar que para a detecção correta do marcador, é necessário que os identificadores ArUcos possuam o entorno em branco.

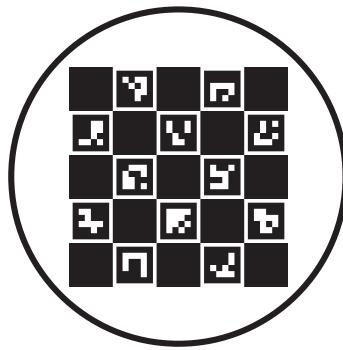


Figura 12 – Área impressa para detecção da superfície

### 6.2.2 Implementação

De forma resumida, para a detecção da posição relativa do ChArUco utiliza-se três funções principais, nativas da biblioteca OpenCV. A primeira função, *detectMarkers*, é responsável por detectar todos os ArUcos do sistema e retorna quais os marcadores foram detectados, junto com seus identificadores. Em seguida, é utilizada a função *interpolateCornersCharuco* para interpolar todos os ArUcos obtidos e verificar se eles fazem parte de um quadro ChArUco determinado, retornando assim seu identificador e a posição em 2D do ChArUco na imagem. Por fim, é utilizada a função mais importante da detecção, *estimatePoseCharucoBoard* que, a partir da posição obtida em 2D e dos parâmetros intrínsecos da câmera, consegue estimar a matriz de rotação e translação em 3D do quadro, referente ao seu canto superior esquerdo. Após a utilização dessa função, é possível também desenhar o eixo detectado na imagem, como pode ser observado na Figura 13

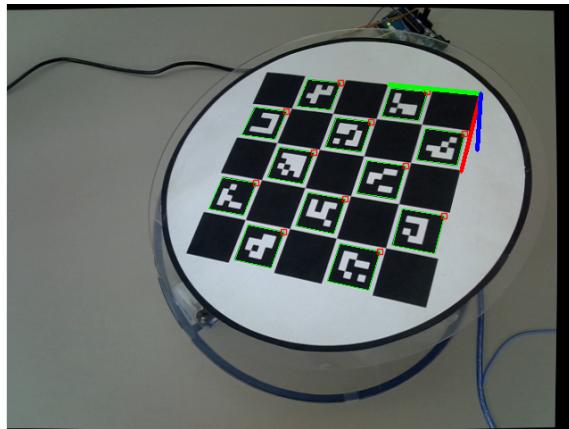
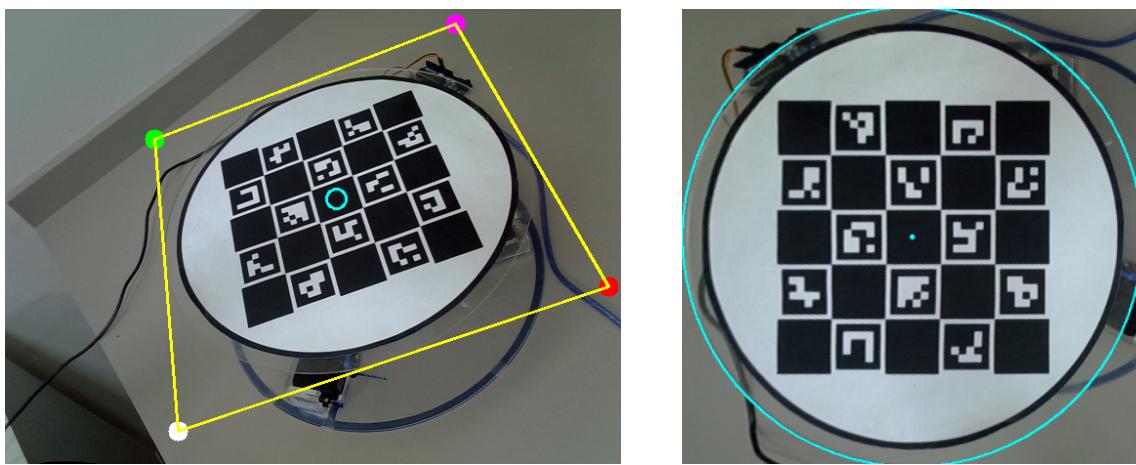


Figura 13 – Superfície detectada, junto com seu eixo desenhado

Em seguida, são utilizadas essas matrizes de rotação e translação, junto com os parâmetros intrínsecos da câmera para fazer a correção da inclinação. A correção da inclinação é muito importante para simplificar os cálculos, uma vez que ela passa o problema de três dimensões, para um problema em apenas duas dimensões. Para fazer essa conversão, basicamente é necessário estimar quatro pontos em três dimensões e informar onde deseja que cada um desses pontos seja definido. Para o caso desse sistema, deseja-se obter um quadrado em que a superfície circular esteja inscrita nele, conforme a Figura 14.



(a) Pontos obtidos para correção da inclinação

(b) Correção realizada

Figura 14 – Representação da correção de inclinação, para transformar um problema de três dimensões, para duas dimensões

Para obter estes quatro pontos, é utilizada a função *projectPoints* do OpenCV. Sabendo que os lados do quadrado estão exatamente a 6 cm em *x* e *y* do quadrado interno do ChArUco, é possível transladar o ponto de forma a encontrar esses vértices. Após a obtenção dos pontos em 3D, foi definido que esses pontos correspondiam às coordenadas em duas dimensões (0,0), (0,600), (600,0) e (600,600). Estes números foram escolhidos de forma a facilitar a conversão de pixels para metros, uma vez que o quadrado possui lado de exatamente de 30 cm. Com os pontos da superfície e sua projeção desejada, a função *warpPerspective* é utilizada para fazer essa transformação exibida na Figura 14. A partir desta imagem, de tamanho (600,600) que a bola irá ser detectada. Dessa maneira, a obtenção das coordenadas e cálculo de sua posição ficará bem mais simples de ser executada.

## 6.3 Detecção da Bola

### 6.3.1 Filtro HSV

Na detecção da bola de tênis de mesa laranja, foi empregada a escala de cores HSV (Matiz, Saturação e Valor). Esta escala foi escolhida no lugar do modelo RGB, devido à sua capacidade de filtrar cores específicas de forma mais eficiente. Os parâmetros HSV permitem não apenas identificar a cor da bola, mas também considerar sua intensidade e brilho. Ao selecionar uma faixa de valores que engloba desde o laranja claro até o laranja escuro, torna-se possível detectar a bola com precisão, adaptando-se a variações de luminosidade e saturação.

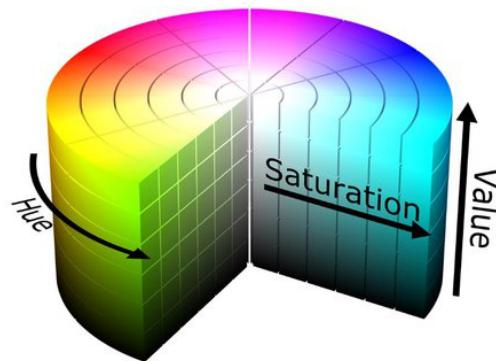
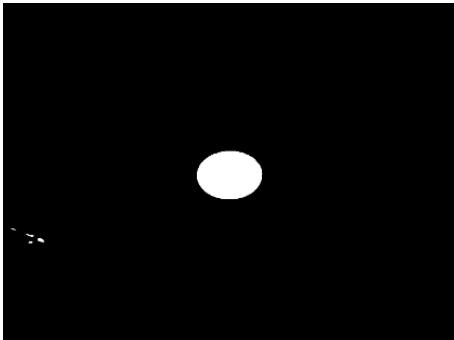


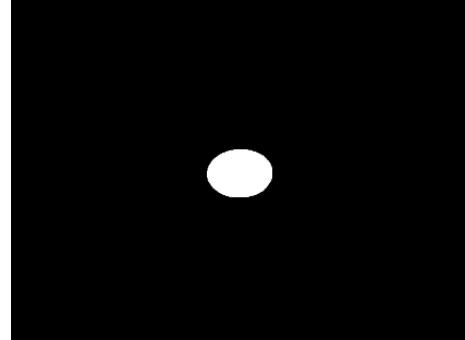
Figura 15 – Representação visual do espaço HSV

A detecção é feita de forma bem facilitada através da função *inRange*, em que a imagem em HSV é fornecida, bem como os valores mínimos e máximos de HSV. Através dessa função, é retornada uma máscara contendo apenas o conteúdo dentro da faixa definida.

Após a obtenção dessa máscara, são realizadas 2 operações sobre ela com a finalidade de reduzir os ruídos, são elas a *erode* e *dilate*. A função *erode* tem a finalidade de “corroer” a imagem, ou seja, remover pequenas identificações na imagem. Já a função *dilate* tem a finalidade de aumentar a área de um objeto. É possível, através do uso das 2 funções, remover pequenas identificações erradas e obter uma máscara mais precisa, conforme a Figura 21



(a) Máscara sem as correções



(b) Máscara com as correções

Figura 16 – Comparação da máscara original e após a aplicação das funções de *erode* e *dilate*

### 6.3.2 Obtenção da posição da bola

Ao obter a máscara contendo apenas o contorno da bola, a função do OpenCV *findContours* é utilizada para transformar a imagem do contorno em coordenadas em que esses contornos são detectados, ou seja, onde há uma diferença muito grande entre os pixels próximos (Para esse caso, do preto para o branco). Ao obter essas coordenadas, uma nova função é utilizada, *minEnclosingCircle*, em que a partir desses pontos, ela retorna o círculo de menor área possível que abrange essas coordenadas. Com isso, essa função retorna, em pixels, a posição em x e y da bolinha e seu raio.

Como observado na Subseção 6.2.2, a imagem de referência, contendo a superfície tratada, é inscrita em um quadrado de 600x600 pixels, sendo sua origem no canto

superior esquerdo da imagem, enquanto que as coordenadas definidas para o controle são em uma superfície de 30x30 cm e com a origem em seu centro. Para converter a posição em x e y da bola, de pixels para metros, e transladar a origem para o centro, uma simples operação pode ser feita, em que:

$$pos_{metros} = \frac{\frac{pos_{pixels}}{2} - 150}{10} \quad (6.1)$$

Ao obter a resposta da Equação 6.1, é possível utilizar as coordenadas da bola para realizar o controle do sistema.

# 7 Controle

Para a concretização deste estudo, o controle da posição da bolinha em relação à superfície é essencial. Para essa finalidade, optou-se pelo método PID, uma técnica clássica de controle simples e amplamente utilizada. Este controlador consiste em uma parte proporcional, integral e derivativa, em que sua equação, no domínio do tempo, pode ser expressa como:

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad (7.1)$$

Em que  $K_p$ ,  $K_i$ , e  $K_d$  representam os ganhos proporcional, integrativo e derivativo, respectivamente, enquanto  $e(t)$  denota o erro, ou seja, a diferença entre a referência desejada e o valor medido. Essa estrutura permite ao controlador PID ajustar dinamicamente a saída  $u(t)$  para minimizar o erro ao longo do tempo.

Como demonstrado pelas equações (5.34) e (5.35) após a linearização, é possível desacoplar o sistema, resultando em duas equações distintas. Assim, para efetuar o controle da planta, serão projetados dois controladores independentes a partir do modelo linearizado: um para o controle da posição da bola em x e outro para o controle da posição da bola em y. Com isso, dado que essas duas plantas compartilham o mesmo comportamento, e assumindo que a dinâmica de todos os servomotores são equivalentes, seus ganhos podem ser calculados de maneira semelhante, o que será abordado na próxima seção.

## 7.1 Cálculo dos ganhos

As funções transferência do sistema linearizado podem ser obtidas a partir das equações (5.34) e (5.35) e expressas como:

$$G_x(s) = \frac{-5.886}{s^2} \quad (7.2)$$

$$G_y(s) = \frac{-5.886}{s^2} \quad (7.3)$$

Para o cálculo dos ganhos, foi-se utilizado o método do lugar das raízes. Para isso, ajustou-se o ganho do controlador e os 2 zeros para que o sistema tivesse pouco *overshoot* (menos de 20%) e um tempo de acomodação aceitável (em comparação com outros trabalhos). O ajuste dos zeros complexos, realizado através da ferramenta *sisotool* do MATLAB, foi posicionado de forma a atrair os polos da função transferência em malha fechada para o lado esquerdo do sistema, o que garante sua estabilidade. Com isso, obteve-se o seguinte lugar das raízes:

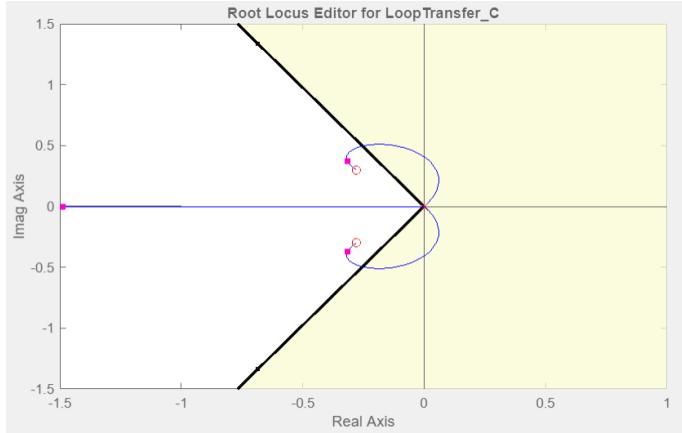


Figura 17 – Representação do Lugar das Raízes

Os ganhos do controlador para essa configuração foram:

$$K_p = -0.2 \quad K_i = -0.06 \quad K_d = -0.36 \quad (7.4)$$

Dessa forma, o controlador teve a seguinte função transferência:

$$C(s) = -0.36 \frac{(s^2 + 0.5556s + 0.1667)}{s} \quad (7.5)$$

Ao simular a resposta ao degrau do sistema com os ganhos calculados, obteve-se a seguinte resposta:

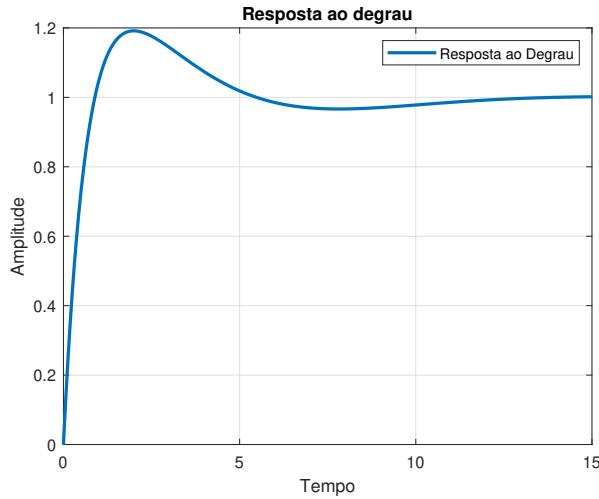


Figura 18 – Resposta ao degrau da planta Linearizada

Com base na resposta ao degrau, é possível analisar que o *overshoot* apresentou-se próximo aos 20%, conforme projetado. Em relação ao tempo de acomodação, observa-se que a planta demorou um pouco a se estabilizar, cerca de 10 segundos. Esse comportamento pode ser atribuído ao ajuste feito nos ganhos de modo que a superfície se inclinasse pouco, configurado justamente para se assemelhar à planta real, em que os ângulos de inclinação  $\alpha$  e  $\beta$  foram limitados a não ultrapassar 20°.

## 7.2 Simulação

Após o cálculo dos ganhos, realizou-se uma simulação, utilizando o modelo não linearizado no Simulink. Ao adaptar as Equações (5.32) e (5.33) para o Simulink, obteve-se o seguinte modelo:

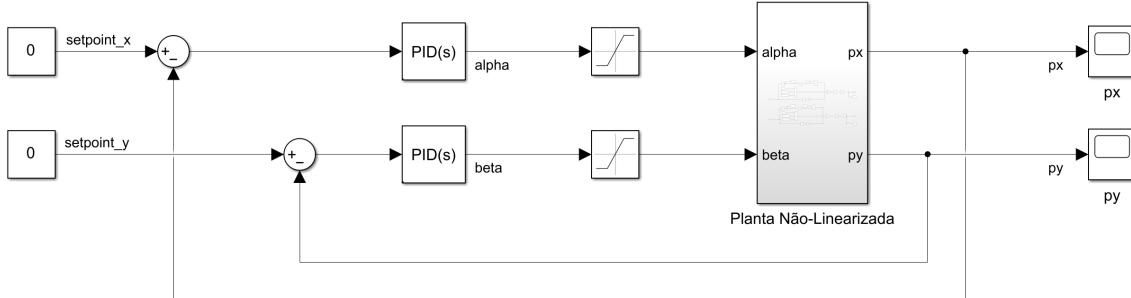


Figura 19 – Visualização do modelo implementado no Simulink

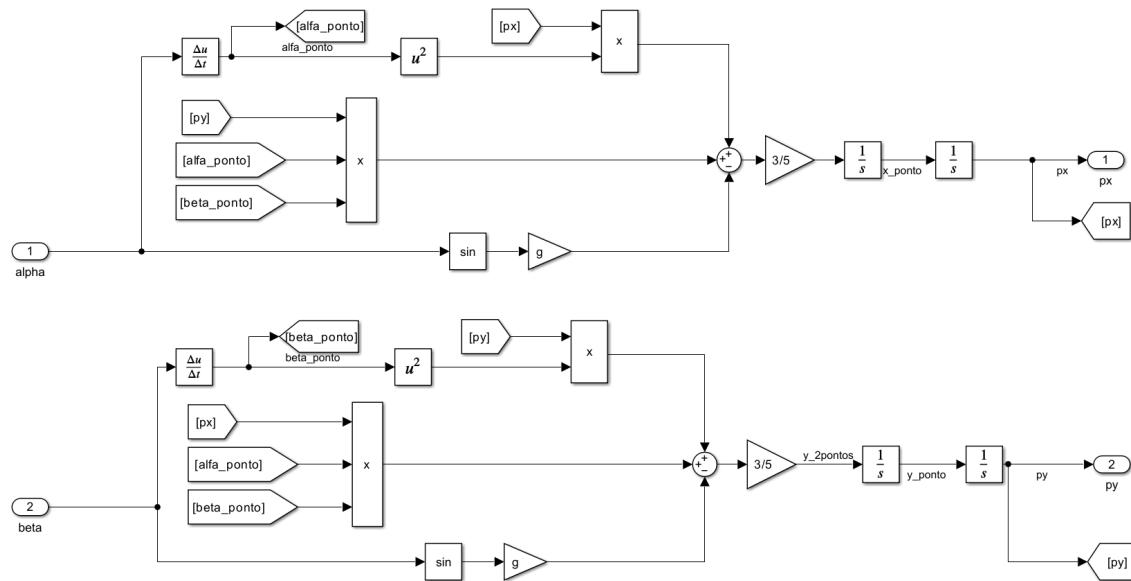


Figura 20 – Visualização do bloco “Planta Não-linearizada” do modelo implementado no Simulink

Após realizar a simulação por 20 segundos, definindo a posição inicial da bola em (5 cm, 5 cm), obteve-se os seguintes resultados, para a posição da bola em x e y:

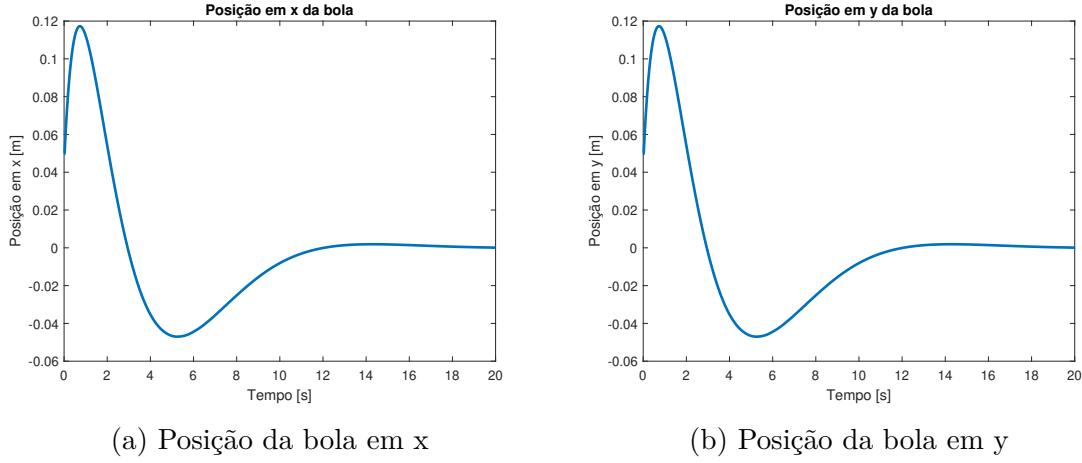


Figura 21 – Simulação da planta não linearizada, realizada no Simulink

Ao implementar os ganhos da planta linearizada no modelo não linearizado, nota-se uma alteração no comportamento da planta. Ao estabelecer uma posição inicial de 5 cm, observa-se um *overshoot* até atingir os 12 cm, seguido por um processo de estabilização. Apesar dessa variação no comportamento, a capacidade de equilibrar a bola na superfície é mantida, considerando que a superfície possui um raio de 15 cm. Esse comportamento indica que, para a planta real, ajustes empíricos são necessários para melhorar a resposta do sistema.

# 8 Resultados

Após a definição e implementação em programa da Cinemática inversa, Visão computacional, Modelagem e Controle, necessitou-se integrar todos esses tópicos no sistema físico com o objetivo de equilibrar a bola em uma posição desejada. Para isso, primeiramente, criou-se uma interface em que integrasse todos esses tópicos em um só lugar. Além disso, foi necessário realizar modificações físicas no sistema em que foram encontradas diversas dificuldades para sua implementação. As alterações foram fundamentais para aplicar as propostas definidas ao longo deste trabalho e efetuar o controle no sistema.

## 8.1 Interface

Com o intuito de otimizar a execução do projeto, foi desenvolvida uma interface utilizando o PyQt, na qual são concentradas todas as informações mais importantes e comandos necessários para visualizar e editar parâmetros do sistema. Essa abordagem elimina a necessidade de reiniciar o programa a cada ajuste de parâmetro. Além disso, por meio desta interface, é possível visualizar as imagens capturadas pela câmera, tanto na forma original quanto com a sua perspectiva corrigida.

A interface é composta por quatro telas principais, sendo a primeira voltada para a visualização da detecção da bola e ajuste dos parâmetros HSV correspondentes, conforme exemplificado na Figura 23.

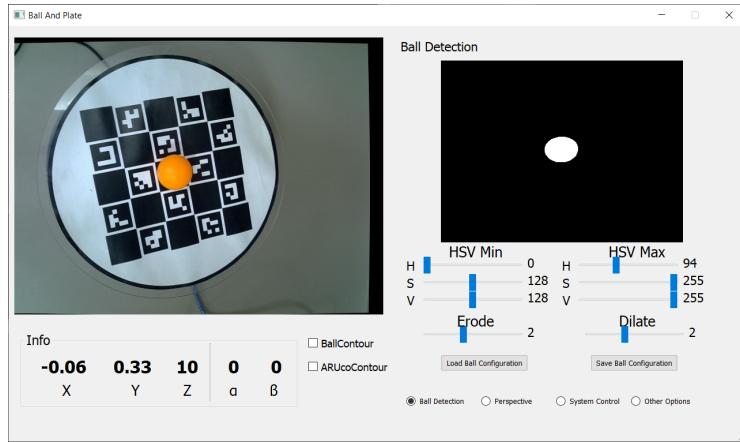


Figura 22 – Tela para edição dos parâmetros de detecção da bola

Outra tela foi desenvolvida para a exibição da perspectiva corrigida:

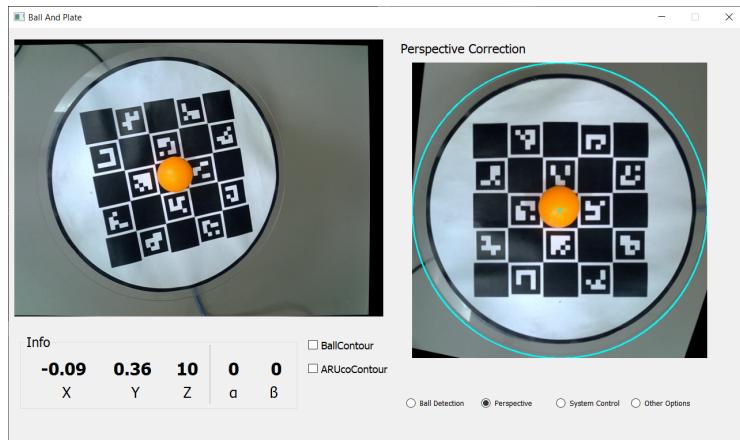


Figura 23 – Tela para exibição da Perspectiva

Além disso, foram criadas duas telas relacionadas ao controle. A primeira permite a inserção manual do grau de inclinação, representado por  $\alpha$  e  $\beta$ , quando o controle está desativado. Adicionalmente, é possível ativar o controle PID e ajustar seus parâmetros, conforme apresentado nas Figuras 24 e 25.

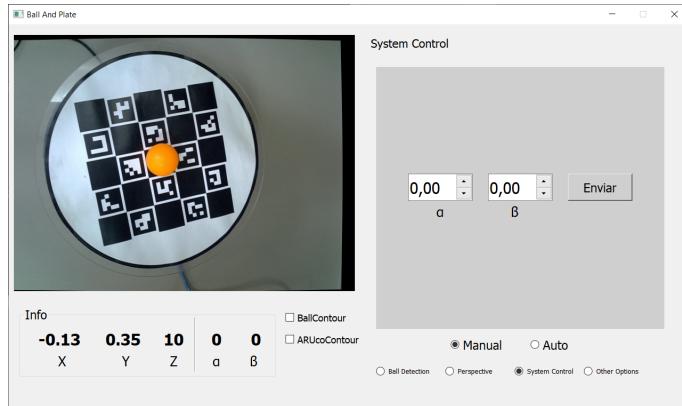


Figura 24 – Tela para editar a inclinação da superfície

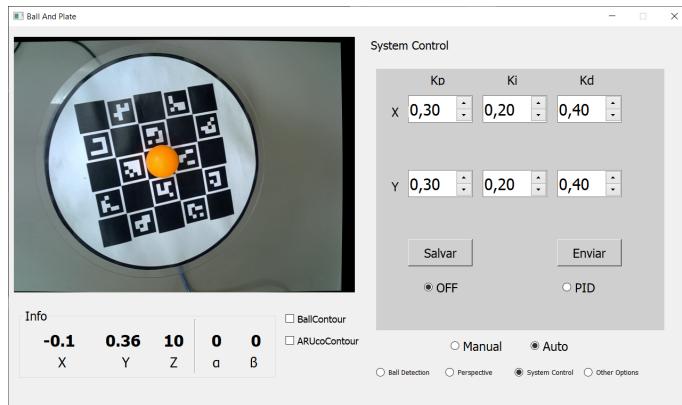


Figura 25 – Tela para habilitar o PID e seus ganhos

## 8.2 Modificações físicas no sistema

Como o sistema físico já existia, sendo o mesmo visto no trabalho de Ferreira (2022), realizou-se algumas modificações para aumentar a robustez e garantir o controle do sistema. As modificações feitas foram:

- Troca dos servos para o modelo MG996R visando aumentar o torque do sistema;
- Troca da superfície para uma mais leve. A superfície de acrílico anterior possuía 244 g e foi trocada por uma nova de 180 g. Essa troca permitiu que

os servos efetuassem menos esforço e, consequentemente, garantiu um melhor posicionamento da superfície;

- Posicionamento dos servos e ímãs com maior precisão. Foi refeito todos seus posicionamentos uma vez que eles estavam colocados com um pequeno erro. Esta ação permitiu que eliminasse um possível fator negativo na implementação da cinemática inversa.
  - Além disso, alterou-se as distâncias dos servos e ímãs do centro para 10 cm ao invés de 11 cm. Esta mudança serviu para acomodar os servos inteiramente dentro da base e também para garantir uma maior robustez na movimentação da superfície, uma vez que, com a configuração antiga, ao realizar mudanças bruscas o ímã esférico se descolava do ímã da superfície.
- Adição do módulo PWM PCA8695 para o controle dos servomotores. Foi utilizado esse módulo principalmente devido ao isolamento da alimentação para o Arduino e para os servomotores. Na configuração antiga, os servomotores eram ligados diretamente ao Arduino, o que, dependendo do torque exigido, poderia ultrapassar os limites do microcontrolador, resultando na sua queima.
- Remoção da Raspberry Pi do sistema devido à lentidão causada pela visão computacional. Para contornar essa limitação, a Raspberry Pi foi substituída por um computador e Arduino, com comunicação serial estabelecida para a troca de informações.

### 8.3 Dificuldades encontradas

Durante a realização do sistema foram encontrados diversos problemas, principalmente relacionados ao sistema físico, especificamente em relação aos servomotores utilizados. Foi observada uma folga nos ângulos definidos pelos servos, resultando em pequenos movimentos não previstos que impactavam a configuração da inclinação da superfície. Adicionalmente, diante de mínimas variações na inclinação da superfície, os servos enfrentavam dificuldades em acompanhar essas alterações. Isso

ocorria devido à magnitude reduzida dos ângulos a serem ajustados após a cinemática inversa, resultando em uma resolução insuficiente por parte dos servos.

## 8.4 Resultados Experimentais

Para realizar o teste na planta física, foi necessário ajustar os ganhos de forma empírica para obter um melhor resultado. Para isso, realizou diversos testes e observou o comportamento da bola em relação à superfície e após alguns testes, chegou nos seguintes ganhos:

$$K_p = 0.28 \quad K_i = 0.12 \quad K_d = 0.40 \quad (8.1)$$

Observa-se uma alteração no sinal dos ganhos, uma vez que a convenção utilizada para a posição da bolinha invertia suas coordenadas, assim, necessitando também inverter o sinal dos ganhos.

Dois testes foram realizados para avaliar o sistema controlador. O primeiro envolveu a perturbação da bola, inicialmente em repouso, com um toque direcionado ao eixo  $x$ , conforme ilustrado na Figura 26.

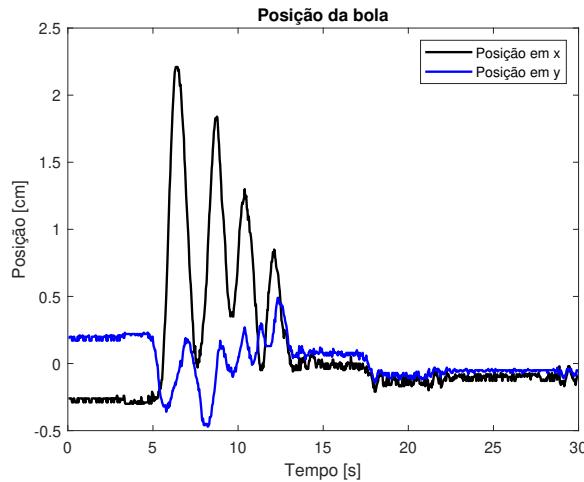


Figura 26 – Controle do sistema após a aplicação de uma perturbação na bola ao longo do eixo  $x$

O segundo teste consistiu em perturbar a bola, agora com uma magnitude maior e em uma direção diagonal, afetando ambos os eixos  $x$  e  $y$ , conforme apresentado na Figura 27.

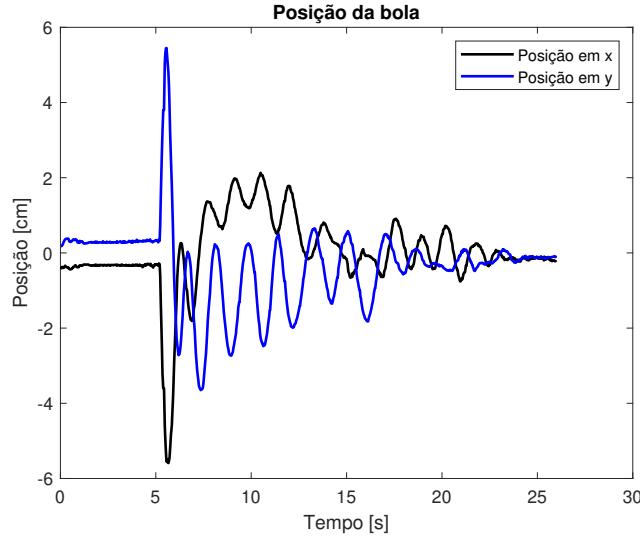


Figura 27 – Controle do sistema após a aplicação de uma perturbação na bola ao longo do eixo  $y$

Em ambos os casos, observou-se que a bola conseguiu seguir a referência após aproximadamente 15 a 20 segundos. Para o teste com uma perturbação menor, as oscilações foram mais sutis, uma vez que o distúrbio foi aplicado apenas no eixo  $x$ . No segundo teste, com uma perturbação de maior magnitude, a bola se deslocou rapidamente para cerca de 6cm do centro, resultando em oscilações mais significativas até atingir a estabilização. Apesar dessas oscilações, fica evidente que o controle foi bem-sucedido em manter a bola sob controle.

## 8.5 Códigos Utilizados

O código fonte desenvolvido para este projeto encontra-se disponível em um repositório hospedado no GitHub. Para acessar e explorar, recomenda-se visitar seu repositório através do link <<https://github.com/HugoFM2/BallAndPlate>>.

Todo o material, desde implementações relacionadas à visão computacional até aspectos cruciais de controle e modelagem, estão presentes neste repositório. Adicio-

nalmente, na mesma plataforma, encontra-se a imagem necessária para a impressão do ChArUco da superfície e também a imagem utilizada para a calibração do sistema. Além disso, também são disponibilizados os arquivos para a impressão 3D dos manipuladores paralelos.

## 9 Conclusões

Através deste sistema, foi possível rever vários conceitos vistos durante a graduação, o que permitiu construir e aprimorar a planta *Ball and Plate* para futura utilização por alunos da Universidade Federal de Minas Gerais.

Em relação à cinemática inversa, a estratégia abordada visou simplificar o cálculo feito por Ferreira (2022). Com isso, cada angulação do servo pode ser calculada de forma independente, precisando apenas da inclinação em  $\alpha$  e  $\beta$  desejada. Ao testar a cinemática inversa de forma manual, pode-se observar que os resultados foram satisfatórios, porém, devido à pequena folga dos servomotores, citado na seção 8.3, existe um pequeno erro que afetava sua inclinação.

Já voltado à visão computacional, implementou-se uma estratégia mais complexa em relação ao trabalho anterior, porém muito mais robusta, em que não é necessário o ajuste fino da posição da câmera e a leitura da posição da bola é muito mais precisa, independente da inclinação da superfície e da câmera. Este método, apesar de ser computacionalmente bem mais custoso (o que proporcionou a troca da Raspberry Pi para um computador com melhor processamento), fornece resultados muito mais precisos ao mesmo tempo em que aborda conceitos mais complexos da visão computacional.

A modelagem feita através da mecânica lagrangiana, provou-se estar correta, uma vez que ao projetar o controlador em torno da planta obtida, foi possível chegar em um controlador que pudesse estabilizar a bola na superfície. Em conjunto com a modelagem, o controle PID apresentou um desempenho satisfatório em relação à resposta e ao erro em regime permanente, sendo restrita apenas pelos limites físicos dos servomotores, abordado na seção 8.3.

Na próxima seção, alguns pontos observados podem ser levados em conta como propostas de melhoria, caso seja de interesse a continuidade deste trabalho.

## 9.1 Propostas de Continuidade

Primeiramente, devido à necessidade de maior capacidade de processamento para as funções de detecção dos marcadores fiduciais, optou-se pela utilização de um computador em substituição à Raspberry Pi, uma vez que a placa apresentava uma taxa de frames em torno de 3 fps (0.3 s), o que se revelou inviável para o propósito. Uma alternativa avaliada foi a tradução do código de Python para C++, visando otimizar a eficiência do código e acelerar sua execução.

Um dos gargalos encontrados no sistema foi a utilização de servomotores genéricos. A substituição desses servomotores por motores originais ou até mesmo por motores de passo deve garantir maior estabilidade e robustez na definição da inclinação da superfície.

Uma proposta adicional visa aprimorar a organização da parte eletrônica do sistema. Para isso, sugere-se a implementação de uma base inferior que possa integrar toda a parte eletrônica, incluindo o Arduino e o módulo PCA9685. Essa abordagem visa otimizar a disposição dos componentes, contribuindo para uma melhor organização do sistema como um todo.

Em relação ao controle, propõe-se a implementação de técnicas mais robustas visando aprimorar a resposta do sistema e seu seguimento de referências. Técnicas como o Controle Linear Quadrático (LQR) ou o controle fuzzy, conforme discutido em Singh e Bhushan (2020), apresentam-se como opções interessantes para elevar o desempenho do sistema Ball and Plate.

## Referências

- BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
- CASTRO, R. da S.; BARTH, J. M. O.; FLORES, J. V.; SALTON, A. T. Modelagem e implementacao de um sistema ball and plate controlado por servo-visao. 2019. Disponível em: <<http://www.sbai2013.ufc.br/pdfs/4635.pdf>>.
- DORF, R.; BISHOP, R. *Sistemas de controle modernos*. LTC, 2009. ISBN 9788521617143. Disponível em: <<https://books.google.com.br/books?id=pT4bQAAACAAJ>>.
- FERREIRA, L. T. Montagem, modelagem e simulação de um sistema ball and plate com manipulador paralelo de três graus de liberdade. 2022.
- MA'ARIF, A.; SETIAWAN, N. R.; RAHAYU, E. S. Embedded control system of dc motor using microcontroller arduino and pid algorithm. *IT Journal Research and Development*, 2021. Disponível em: <<https://api.semanticscholar.org/CorpusID:233627047>>.
- NOKHBEH, M.; KHASHABI, D. Modelling and control of ball-plate system. 2011. Disponível em: <<https://danielkhashabi.com/files/2011\LinearControl/16.pdf>>.
- POROYKOV, A.; KALUGIN, P.; SHITOV, S.; LAPITSKAYA, I. Modeling aruco markers images for accuracy analysis of their 3d pose estimation. 2020. Disponível em: <<https://ceur-ws.org/Vol-2744/short14.pdf>>.
- RODRIGUES, M. C.; MAITELLI, A. L.; ARAÚJO, F. M. U. Controle neuro-fuzzy com treinamento em tempo real aplicado a um sistema ball and beam. 2004. Disponível em: <<https://www.dca.ufrn.br/~maitelli/FTP/artigos/MarconiCBA04.pdf>>.
- SINGH, R.; BHUSHAN, B. Real-time control of ball balancer using neural integrated fuzzy controller. 2020. Disponível em: <<https://dl.acm.org/doi/abs/10.1007/s10462-018-9658-7>>.