UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Luiz Henrique Susin

Controle de posição de uma esfera levitante por meio de visão computacional e aprendizado por reforço

Blumenau
2024

Luiz Henrique Susin

Controle de posição de uma esfera levitante por meio de visão computacional e aprendizado por reforço

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.
Orientador: Prof. Marcos Vinicius Matsuo, Dr.

Blumenau
2024

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Susin, Luiz Henrique
Controle de posição de uma esfera levitante por meio de visão computacional e aprendizado por reforço / Luiz Henrique Susin ; orientador, Marcos Vinicius Matsuo, 2024.
76 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Blumenau, Graduação em Engenharia de Controle e Automação, Blumenau, 2024.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Visão computacional. 3. Controle PID. 4. Machine Learning. I. Matsuo, Marcos Vinicius. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Luiz Henrique Susin

Controle de posição de uma esfera levitante por meio de visão computacional e aprendizado por reforço

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Controle e Automação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 01 de Fevereiro de 2024.

Banca Examinadora:

Documento assinado digitalmente
Marcos Vinicius Matsuo
Data: 05/02/2024 10:12:17-0300
CPF: ***.580.029-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Marcos Vinicius Matsuo, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Leonardo Mejia Rincon
Data: 05/02/2024 12:25:27-0300
CPF: ***.678.849-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Leonardo Mejia Rincon, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Mauri Ferrandin
Data: 05/02/2024 10:20:11-0300
CPF: ***.580.869-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Mauri Ferrandin, Dr.
Universidade Federal de Santa Catarina

Aos meus pais, Silvania e Luiz, dedico este trabalho. Cada página é um reflexo do apoio incondicional que recebi e da sabedoria transmitida. Obrigado por serem a estrela-guia que conduz minha jornada.

AGRADECIMENTOS

Aos meus pais, Silvania e Luiz, expresso minha profunda gratidão, cujo amor incondicional e apoio incansável foram fundamentais para a realização deste trabalho. A jornada acadêmica que culmina neste Trabalho de Conclusão de Curso foi marcada por desafios e triunfos e, em cada etapa, pude contar com a orientação sábia e o encorajamento caloroso de vocês. Seu sacrifício e dedicação moldaram não apenas meu percurso acadêmico, mas também meu caráter e visão de mundo. Este trabalho é, em grande parte, uma manifestação da influência positiva que vocês têm sobre minha vida. Agradeço por serem não apenas meus pais, mas também meus eternos mentores e inspirações.

Aos meus melhores amigos, que não apenas compartilharam risos e histórias, mas que também se tornaram pilares essenciais na minha jornada acadêmica, dedico esta conquista. Suas contribuições não se limitaram à amizade; foram alicerces sólidos que sustentaram minha formação. Cada conselho, cada desafio superado juntos, transformou meu percurso acadêmico de maneiras inestimáveis. Este trabalho é, portanto, uma homenagem não apenas ao meu esforço individual, mas também à incrível equipe de amigos que esteve ao meu lado, compartilhando conhecimento e apoio.

Ao meu orientador, Prof. Dr. Marcos Vinicius Matsuo, pela dedicação na correção e aprimoramento deste trabalho. Certamente não teria atingido esse nível de excelência sem o seu apoio, orientação e confiança em meu potencial. Sou grato por ter tido a oportunidade de aprender com você e por sua influência positiva em minha jornada acadêmica.

Ao campus Blumenau da Universidade Federal de Santa Catarina, agradeço por fornecer uma plataforma educacional excepcional que não apenas desafiou meu intelecto, mas também nutriu meu crescimento pessoal e profissional.

À dedicada equipe de professores, expresso meu reconhecimento pela orientação valiosa, pelo estímulo à excelência acadêmica e pelo comprometimento inabalável com o desenvolvimento dos estudantes. Cada aula, cada projeto e cada interação contribuiu significativamente para minha formação. Levo comigo não apenas conhecimento técnico, mas também a gratidão por fazer parte de uma comunidade educacional tão inspiradora.

À empresa ZPE Systems por confiar em meu trabalho durante o estágio obrigatório. A oportunidade de aplicar os conhecimentos adquiridos na universidade em um ambiente profissional foi inestimável para o meu desenvolvimento acadêmico e profissional. Agradeço sinceramente pela confiança depositada em mim, pela orientação recebida e pela experiência enriquecedora proporcionada pela equipe.

The difficult is that which takes a little time, the impossible is what takes a little longer. (NANSEN, 1936)

RESUMO

A necessidade de aperfeiçoamento dos métodos tradicionais de *autotuning*, como o método de Ziegler-Nichols, é destacada dada a presença de limitações destas ferramentas em sistemas dinâmicos, não lineares e em ambientes sujeitos a constantes mudanças. Tais métodos convencionais muitas vezes exigem um conhecimento aprofundado da planta e podem não ser eficazes diante de incertezas, necessitando de ajustes manuais frequentes. Este trabalho se propõe a controlar a posição de uma esfera - cuja posição é adquirida utilizando visão computacional - em um tubo utilizando um controlador PID e aprendizado por reforço com Q-Learning como uma solução para o *autotuning* de seus ganhos. Esta abordagem combina inteligência artificial com controle de processos, oferecendo uma estratégia adaptativa e flexível que supera as limitações dos métodos convencionais. O uso da visão computacional é essencial devido à sua capacidade de fornecer informações em tempo real sobre a posição da esfera. A câmera RGB utilizada no sistema captura imagens que permitem a retroalimentação, sendo mais efetiva do que outras soluções tais como sensores ultrassônicos. O uso em conjunto com aprendizado de máquina permite ajustar dinamicamente o controle em resposta à mudanças nas condições ambientais e comportamento da esfera. Tal incorporação no ajuste dinâmico dos ganhos do controlador PID representa um salto significativo em direção à automação inteligente, já aplicado na indústria. Ele permite que o sistema aprenda e adapte-se continuamente ao ambiente, otimizando seus próprios parâmetros de controle, fornecendo ao sistema a capacidade de se ajustar a novos contextos de forma autônoma, além de proporcionar uma resposta adaptativa a perturbações, promovendo a estabilidade do sistema. Assim, a combinação da visão computacional e do aprendizado por reforço não apenas oferece uma abordagem avançada para o controle do sistema, mas também estabelece um modelo para a aplicação dessas tecnologias em sistemas complexos e dinâmicos em diversos campos, desde automação industrial até pesquisa científica. Através de experimentos práticos mostrou-se que o algoritmo Q-Learning foi capaz de encontrar ganhos para o controlador que corrigiram uma dinâmica oscilatória do sistema em apenas 8 minutos (considerando as etapas de aprendizado e correção).

Palavras-chave: Visão computacional; Controle PID; Machine Learning.

ABSTRACT

The need for the refinement of traditional autotuning methods, such as the Ziegler-Nichols method, is highlighted given the presence of limitations of these tools in dynamic, nonlinear systems and in environments subject to constant changes. Such conventional methods often require an in-depth knowledge of the plant and may not be effective in the face of uncertainties, necessitating frequent manual adjustments. This paper proposes to control the position of a sphere - whose position is acquired using computer vision - in a tube using a PID controller and reinforcement learning with Q-Learning as a solution for autotuning its gains. This approach combines artificial intelligence with process control, offering an adaptive and flexible strategy that overcomes the limitations of conventional methods. The use of computer vision is essential due to its ability to provide real-time information about the sphere's position. The RGB camera used in the system captures images that allow feedback, being more effective than other solutions such as ultrasonic sensors. The use in conjunction with machine learning allows for dynamically adjusting control in response to changes in environmental conditions and the behavior of the sphere. Such incorporation into the dynamic adjustment of the PID controller's gains represents a significant leap towards intelligent automation, already applied in the industry. It allows the system to learn and continuously adapt to the environment, optimizing its own control parameters, providing the system with the ability to adjust to new contexts autonomously, as well as offering an adaptive response to disturbances, promoting system stability. Thus, the combination of computer vision and reinforcement learning not only offers an advanced approach for system control but also establishes a model for the application of these technologies in complex and dynamic systems across various fields, from industrial automation to scientific research. By using pratical experiments, it was found that the Q-Learning algorithm was able to find gains for the controller that corrected an oscillatory dynamic of the system in just 8 minutes (considering the learning and correction phases).

Keywords: Computer Vision; PID Control; Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CAD	Computer-aided Design
CC	Corrente Contínua
CCD	Charge-Coupled Device
CMOS	Complementary Metal-Oxide-Semiconductor
CNC	Comando Numérico Computadorizado
CSV	Comma-separated Values
FPS	Frames per second
HSV	Hue, Saturation, Value
IDE	Integrated Development Environment
MSE	Mean Squared Error
PCA	Análise de Componentes Principais
PI	Proporcional Integral
PID	Proporcional Integral Derivativo
PLA	Polylactic Acid
PWM	Pulse Width Modulation
RGB	Red, Green, Blue
SVM	Máquinas de Vetores de Suporte
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VGA	Video Graphics Array

SUMÁRIO

1 INTRODUÇÃO

No epicentro da Revolução Industrial, o Controle PID (Proporcional Integral Derivativo), conforme discutido por K. J. Åström e Hägglund (1995), emergiu como uma resposta engenhosa à crescente complexidade das máquinas. A obra dos autores em questão evidencia que o PID não apenas se tornou uma ferramenta indispensável na automação industrial, mas também evoluiu continuamente para enfrentar os desafios impostos por sistemas dinâmicos cada vez mais intrincados.

A incorporação da visão computacional, conforme descrita por Szeliski (2010), representa um marco na interação homem-máquina. Desde a detecção de padrões até o reconhecimento de objetos, a visão computacional transcendeu fronteiras disciplinares, influenciando setores tão diversos quanto a medicina, a automação industrial e a robótica. No cerne dessa transformação encontra-se a capacidade de processamento de informações visuais, antes limitada aos domínios humanos, agora potencializada pela tecnologia.

Ao explorar a interseção entre o PID e a visão computacional, torna-se evidente que este não é apenas um casamento por conveniência, mas sim a convergência de duas disciplinas que, juntas, oferecem um sinergismo único. Considerando os avanços nas técnicas de ajuste e modelagem de controladores PID, conforme discutido por Belanger (1995), percebemos que essa união não é apenas sobre o controle eficaz de sistemas, mas também sobre a compreensão e adaptação a ambientes complexos.

O estudo dessas interações complexas expande-se para a esfera da educação, onde projetos práticos, como demonstrações visuais das respostas do sistema à troca dos ganhos do PID, desempenham um papel vital. Essas experiências não apenas capacitam os alunos a compreender conceitos abstratos, mas também estimulam o pensamento crítico e a resolução de problemas, habilidades essenciais em um mundo onde a automação e a inteligência artificial desempenham um papel cada vez mais central.

Contudo, o desafio contemporâneo reside na implementação do aprendizado por reforço para otimização em sistemas de controle, um tema discutido por Dogru *et al.* (2022). Esta abordagem representa uma fronteira a ser desbravada, onde o equilíbrio delicado entre a eficiência prática e a complexidade algorítmica se desenha como um desafio a ser superado.

Ao explorar os benefícios dessa sinergia entre controle PID e visão computacional, percebe-se que estamos na vanguarda de uma revolução tecnológica que transcende as fronteiras dos sistemas de controle de dinâmica estocástica. O uso inovador de uma plataforma de esfera levitante para a experimentação prática representa a materialização de ideias futuristas em soluções tangíveis.

Este sistema visa não apenas melhorar a precisão e eficiência do controle em ambientes dinâmicos e incertos, mas também estabelecer uma nova referência em automação inteligente e sistemas adaptativos. O projeto foca em alcançar uma compreensão profunda

das interações entre o controle PID, a análise visual em tempo real e o aprendizado autônomo, visando criar um modelo que seja capaz de se ajustar autonomamente a novos contextos e perturbações ambientais. Além disso, pretende-se que o sistema seja uma plataforma para futuras pesquisas e aplicações em campos diversos, desde a automação industrial até avanços em pesquisa científica, propondo um paradigma onde a precisão, adaptabilidade e aprendizado contínuo convergem para criar um controle de alta performance e inteligente.

Em síntese, o trabalho não é apenas uma exploração técnica, mas uma imersão em uma narrativa mais ampla de como as disciplinas interagem, evoluem e moldam o mundo à nossa volta. À medida que se aprofunda nas complexidades do controle PID, visão computacional e aprendizado por reforço, delineia-se não apenas uma análise minuciosa dessas técnicas, mas também uma compreensão profunda de seu papel transformador na vanguarda da inovação. Este estudo, ao desvelar as interseções e desafios dessas disciplinas, contribui não apenas para o conhecimento científico, mas para o progresso incessante rumo a sistemas mais inteligentes e eficientes, refletindo a busca incansável pela excelência tecnológica na sociedade contemporânea.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver e implementar um sistema de controle avançado para a posição de uma esfera levitante, integrando de forma inovadora a visão computacional e o aprendizado por reforço para otimizar os ganhos de um controlador PID.

1.1.2 Objetivos Específicos

- Desenvolver o algoritmo de controle PID para a regulação da posição da esfera levitante.
- Implementar um sistema de visão computacional usando uma câmera VGA para monitorar em tempo real a posição da esfera.
- Criar algoritmos para processar e interpretar dados visuais, garantindo *feedback* preciso para o sistema de controle.
- Aplicar técnicas de aprendizado por reforço para otimizar dinamicamente os ganhos do controlador PID com base no desempenho do sistema.
- Realizar experimentos para encontrar os melhores parâmetros e configurações para o sistema de controle e aprendizado por reforço.
- Verificar a estabilidade e a confiabilidade do sistema em diferentes condições, garantindo sua adaptabilidade e eficiência.

- Criar uma interface para visualizar em tempo real o comportamento da esfera e a performance do controlador.
- Documentar todo o processo de desenvolvimento e análise, incluindo metodologias, resultados, e discussões.
- Sugerir aprimoramentos e direções futuras para pesquisa baseadas nos resultados e limitações observadas.

1.1.3 Organização do Trabalho

Além do capítulo de introdução, esta monografia possui outros 4 (quatro) capítulos, seguindo a disposição abaixo descrita.

- No Capítulo 2, contextualiza-se o leitor acerca das tecnologias relacionadas ao desenvolvimento do trabalho, incluindo, mas não limitando-se, as macroáreas do controle de sistemas, visão computacional e aprendizado de máquina.
- No Capítulo 3, descreve-se a proposta de solução do problema, de forma a demonstrar como o problema deve ser resolvido e a metologia utilizada para adquirir os resultados experimentais.
- No Capítulo 4, decorre-se acerca da coleta e análise de resultados.
- No Capítulo 5, apresentam-se as conclusões extraídas desta monografia, além de indicar sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados alguns conceitos fundamentais necessários para a compreensão deste trabalho. Especificamente, são discutidos alguns algoritmos básicos de visão computacional, controle PID e o seu ajuste por meio de aprendizado por reforço.

2.1 SISTEMAS DE CONTROLE

Sistemas de controle são uma pedra angular da engenharia moderna, impactando uma vasta gama de aplicações em diversos setores. Desde o controle de temperatura em residências até a complexa gestão de processos industriais, a aviação, a robótica e além, os sistemas de controle estão em toda parte, garantindo que máquinas e processos operem dentro dos parâmetros desejados, de forma eficiente e segura. Ogata (2010) define sistemas de controle como conjuntos de dispositivos ou técnicas que gerenciam, comandam, direcionam ou regulam o comportamento de outros dispositivos ou sistemas. Eles são projetados para responder a mudanças, manter a estabilidade e otimizar o desempenho, apesar das variações e perturbações externas e internas.

A importância desses sistemas no mundo moderno destaca quão fundamental são para a manutenção da qualidade, segurança, eficiência e confiabilidade em inúmeras aplicações. À medida que a tecnologia avança, também evolui a complexidade dos sistemas de controle, bem como as estratégias necessárias para gerenciá-los.

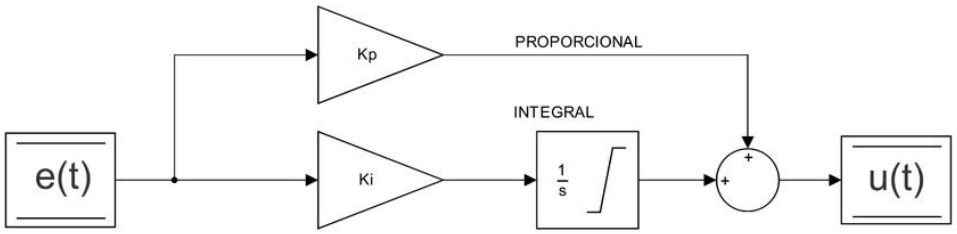
2.1.1 Controladores PI

Os controladores PI (Proporcional Integral) representam uma das estratégias de controle mais fundamentais e amplamente utilizadas na indústria. C. A. Smith (2005) descreve a ação proporcional como a resposta imediata ao erro atual - a diferença entre o valor desejado (setpoint) e o valor real (medido). Esta ação proporcional ajusta a saída do controlador proporcionalmente ao erro, proporcionando uma correção rápida. No entanto, a ação proporcional sozinha muitas vezes deixa um erro residual, conhecido como erro em regime permanente. É aqui que a ação integral entra em cena, acumulando o erro ao longo do tempo e introduzindo uma correção que visa eliminar esse erro persistente.

Os controladores PI são particularmente valorizados por sua simplicidade e eficácia. Eles são menos complexos do que os controladores PID e, muitas vezes, fornecem um desempenho satisfatório em sistemas onde uma resposta ultra-rápida não é crítica (ÅSTRÖM, 2002). No entanto, a sua eficácia pode ser limitada em sistemas mais complexos ou em situações que exigem uma resposta rápida e precisa a mudanças rápidas ou perturbações severas. A estrutura de blocos de um controlador PI pode ser vista na Figura 1.

A equação geral para um controlador PI pode ser vista na Equação (1).

Figura 1 – Diagrama de Blocos de um Controlador PI.



Fonte: Adaptado de Almeida, Salles e Carvalho (2020).

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau \tag{1}$$

Onde cada variável representa:

- $u(t)$: Saída do controlador no instante de tempo t.
- $e(t)$: Erro de medição no instante t, o qual é dado pela diferença entre o valor desejado e o valor real.
- $\int_0^t e(\tau)\, d\tau$: Integral do erro ao longo do tempo, acumulando o erro passado.
- K_p: Ganho proporcional, que determina a reação imediata ao erro atual.
- K_i: Ganho integral, que determina a reação à soma acumulada dos erros passados.

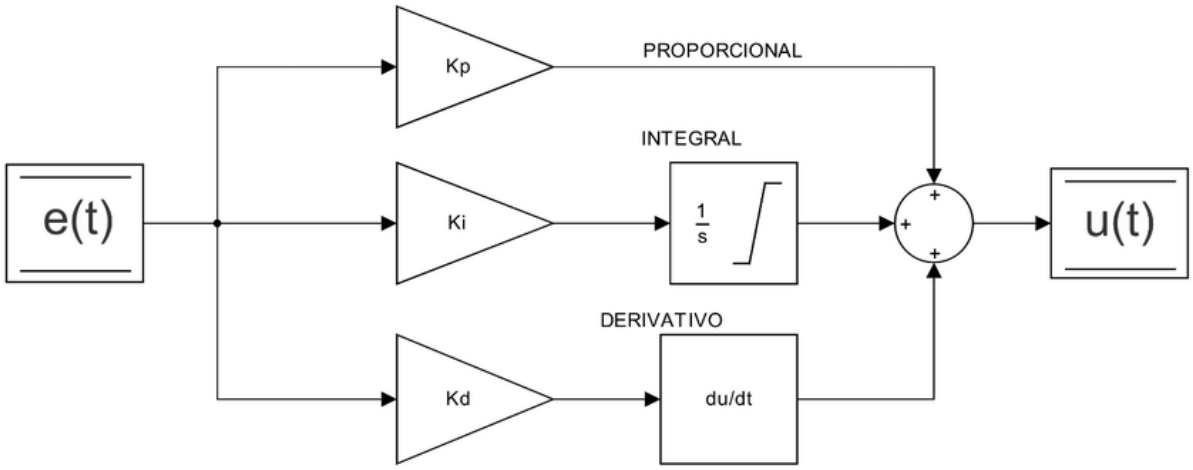
A compreensão da formulação e da estrutura desses controladores, representada pela equação geral e pelo diagrama de blocos, respectivamente, é crucial para implementá-los de forma eficaz em diversas aplicações industriais, garantindo operações estáveis e eficientes.

2.1.2 Controladores PID

Os controladores PID (Proporcional Integral Derivativo) são uma extensão dos controladores PI, incorporando uma terceira ação: a derivativa. Franklin, Powell e Emami-Naeini (2021) explicam que a ação derivativa responde à taxa de mudança do erro, oferecendo uma forma de previsão e correção proativa. Isso permite ao controlador não apenas reagir ao erro atual, como no caso proporcional, ou acumular o erro passado, como no integral, mas também antecipar a direção futura do erro. A eficácia de um controlador PID depende fortemente de sua sintonia, um processo que Dorf e R. H. Bishop (2021) descrevem como crucial e muitas vezes desafiador. Ajustar os parâmetros proporcional, integral e derivativo requer um entendimento profundo do sistema em questão e dos objetivos de controle desejados. Uma sintonia inadequada pode levar a um desempenho insatisfatório, incluindo instabilidades, sobressinal excessivo e resposta lenta. Os controladores PID são

notáveis pela sua versatilidade e eficácia em uma ampla gama de condições e tipos de sistemas. Eles são a escolha preferida em aplicações que exigem uma resposta rápida e precisa, onde a estabilidade e a minimização do erro são críticas. Sua capacidade de ser sintonizado para atender às necessidades específicas de um sistema os torna uma ferramenta indispensável em muitos contextos industriais e comerciais. O diagrama de blocos de um controlador PID pode ser visto na Figura 2.

Figura 2 – Diagrama de Blocos de um Controlador PID.



Fonte: Almeida, Salles e Carvalho (2020).

Apesar de sua eficácia, os controladores PID não são uma solução universal. Eles podem ser excessivamente complexos para sistemas simples, onde um controlador PI pode ser suficiente. Além disso, em sistemas altamente complexos, dinâmicos ou não lineares, estratégias de controle mais avançadas podem ser necessárias para alcançar o desempenho desejado.

A equação geral para um controlador PID é muito similar à equação de um controlador PI dada pela Equação (1). A diferença fundamental entre as equações reside na presença do termo derivativo na equação do PID, conforme segue:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t) \tag{2}$$

Em particular, o termo derivativo presente no controlador PID é composto por:

- $\frac{d}{dt}e(t)$: Derivada do erro em relação ao tempo, representando a taxa de variação do erro.
- K_d: Ganho derivativo, que determina a reação à taxa de mudança do erro.

Enquanto o controlador PI reage apenas ao erro atual e ao erro acumulado, o controlador PID também reage à taxa de mudança do erro. Isso permite que o PID antecipe

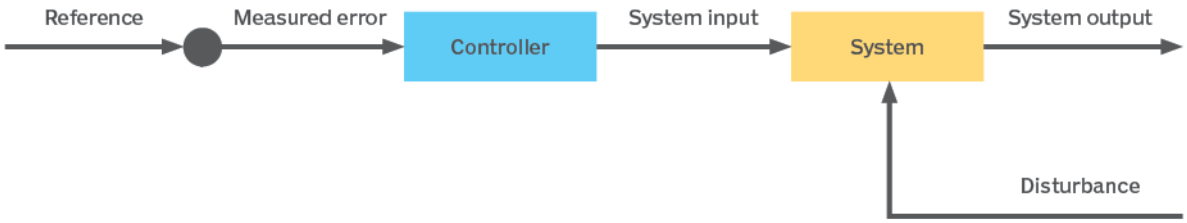
mudanças e atue de maneira mais eficaz em sistemas onde a previsão do comportamento futuro pode melhorar significativamente o controle.

2.1.3 Controle em Malha Aberta e Malha Fechada

O controle em malha aberta e malha fechada são dois métodos fundamentais utilizados em sistemas de controle para gerenciar e regular o comportamento de sistemas dinâmicos. Ambos têm suas aplicações, vantagens e desvantagens, dependendo da natureza do sistema e do objetivo desejado.

No controle em malha aberta, o sinal de controle é enviado para o atuador sem qualquer *feedback* sobre o resultado. Ou seja, o sistema não mede o resultado da ação de controle e não ajusta automaticamente sua ação com base nesse resultado (OGATA, 2010). Enquanto sistemas de malha aberta são mais simples e têm menor custo de implementação, eles não compensam distúrbios ou mudanças nas condições do sistema, o que pode resultar em menor precisão e eficiência. O diagrama de blocos que exemplifica a estrutura básica de um sistema de controle em malha aberta é ilustrado na Figura 3.

Figura 3 – Diagrama de blocos de um sistema de controle de malha aberta.



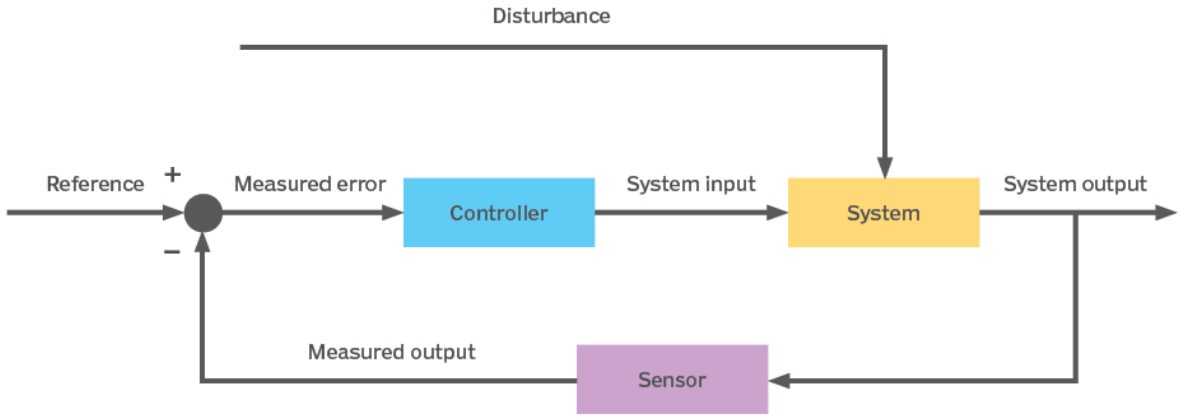
Fonte: Adaptado de Gavin Wright (2022).

Por outro lado, o controle em malha fechada, também conhecido como controle com *feedback*, utiliza informações da saída do sistema para ajustar continuamente a entrada, buscando minimizar o erro e melhorar a precisão e estabilidade do sistema (OGATA, 2010). Este método é essencial em aplicações onde precisão e resposta a perturbações são críticas. Para fins de ilustração, na Figura 4 é apresentado a estrutura básica de um sistema de controle em malha fechada.

2.1.4 Sintonia automática de controladores PI e PID

A sintonia automática (*Autotuning*) de controladores PI e PID é uma técnica avançada utilizada para ajustar automaticamente os parâmetros de controle desses sistemas, garantindo desempenho e estabilidade que atendam critérios de desempenho pré-estabelecidos. Essa abordagem é crucial em diversas aplicações industriais e de engenharia

Figura 4 – Diagrama de blocos de um sistema de controle de malha fechada.



Fonte: Gavin Wright (2022).

onde os controladores PI e PID são empregados para manter o desempenho desejado de sistemas dinâmicos.

Entre os métodos de *autotuning*, o método Ziegler-Nichols é um dos mais clássicos e amplamente utilizados, baseando-se na observação da resposta do sistema a uma perturbação. Inicialmente, os parâmetros do PID são ajustados conservadoramente e, em seguida, refinados com base na resposta observada. Isso é feito iterativamente até alcançar estabilidade e desempenho desejados (ÅSTRÖM, K.; HÄGGLUND, 2006). Alternativamente, métodos modernos como algoritmos genéticos e técnicas de otimização baseadas em enxame, como a Otimização por Enxame de Partículas, têm sido aplicados para sistemas mais complexos onde abordagens tradicionais podem não ser suficientes (BEQUETTE, 2003). O *autotuning* oferece a vantagem de adaptar os parâmetros do controlador às mudanças dinâmicas do sistema, reduzindo a necessidade de intervenção manual melhorando a eficiência e o desempenho do sistema.

Realizar processos de *autotuning* em controladores PI e PID quando há desconhecimento da planta apresenta uma série de complexidades e desafios. K.J. Åström e Hägglund (2006) apontam que a falta de um modelo preciso da planta pode levar a estimativas imprecisas dos parâmetros do controlador, resultando em desempenho subótimo ou até mesmo em instabilidade do sistema. Isso ocorre porque a eficácia do *autotuning* depende significativamente do conhecimento das características dinâmicas do sistema. Sem um entendimento claro da resposta da planta às entradas, torna-se difícil determinar os parâmetros ideais que garantirão uma resposta rápida e estável.

Bequette (2003) destaca que, em situações onde a planta é desconhecida ou altamente não-linear, métodos de *autotuning* mais avançados e adaptativos podem ser necessários. Técnicas como algoritmos genéticos ou otimização baseada em enxame podem explorar uma gama mais ampla de possíveis soluções e adaptar-se a mudanças na dinâmica

da planta. No entanto, esses métodos geralmente requerem uma quantidade significativa de dados e tempo de computação para convergir para uma solução adequada, aumentando a complexidade do processo de *autotuning*.

2.2 MODULAÇÃO POR LARGURA DE PULSO (PWM)

A técnica de Modulação por Largura de Pulso, tradução livre de PWM (Pulse Width Modulation), é uma abordagem amplamente utilizada no controle de dispositivos eletrônicos, incluindo o controle de velocidade de motores elétricos de corrente contínua (CC). Nesse contexto, a técnica PWM envolve a variação da largura dos pulsos em uma sequência de sinais digitais para controlar a quantidade de energia fornecida ao motor, permitindo um controle preciso sobre sua velocidade.

2.2.1 Princípios, Implementação e Usabilidade

Conforme descrito por Holmes e Lipo (2003), a técnica PWM controla a tensão média fornecida a um motor elétrico ajustando o tempo em que a tensão está ligada durante cada ciclo. Em essência, ao invés de fornecer uma tensão contínua, a PWM fornece uma série de pulsos de tensão. A rapidez com que o motor gira depende da média de tensão que ele recebe, que é determinada pela duração de cada pulso. Uma das principais vantagens da PWM, como discutido por Toliyat e Campbell (2003), é a sua eficiência energética. Ao contrário dos métodos de controle que dissipam energia como calor em resistores ou em dispositivos semicondutores, a PWM controla eficientemente a energia fornecida ao motor, minimizando as perdas de energia. Essa técnica é implementada usando circuitos eletrônicos chamados de controladores PWM, que podem ser microcontroladores programáveis ou circuitos integrados dedicados. Um exemplo de controlador que implementa um circuito PWM pode ser visto na Figura 5.

A modulação PWM é especialmente útil em aplicações onde é necessário um controle preciso e eficiente da velocidade ou do torque do motor, como em veículos elétricos, sistemas de condicionamento de ar, bombas e ventiladores industriais, e em robótica. Em paralelo, neste trabalho, a técnica PWM é usada para controle de uma ventoinha de computador, isto é, um motor elétrico de corrente contínua.

2.2.2 Características do Sinal PWM

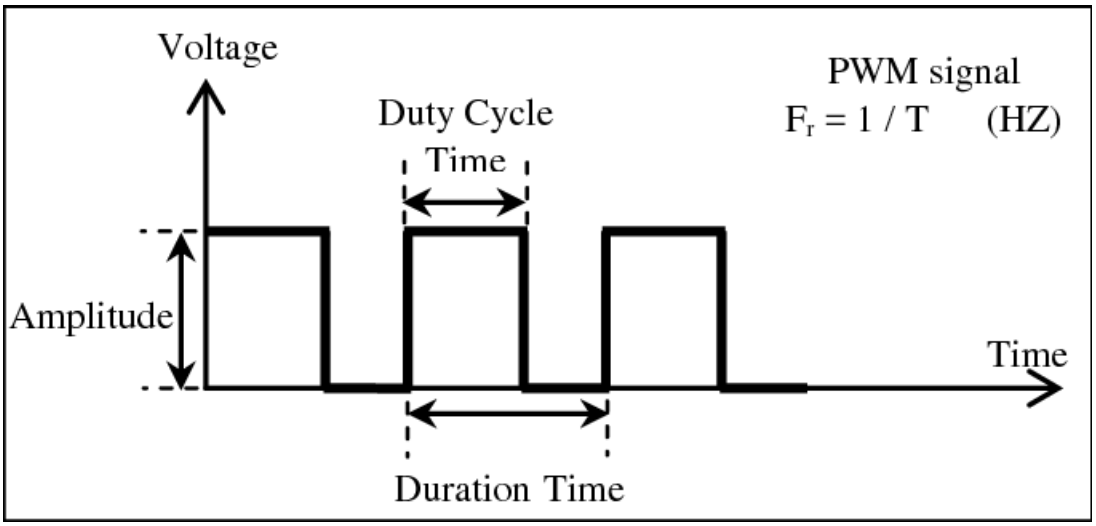
O sinal PWM é tipicamente uma forma de onda quadrada, que alterna entre um estado alto (tensão máxima) e um estado baixo (tensão mínima ou zero), criando pulsos de tensão. O ciclo de trabalho, tradução livre de *Duty Cycle*, é a característica mais crítica da onda, representando a fração ou porcentagem de um ciclo completo em que a onda está no estado alto. Por exemplo, um ciclo de trabalho de 50% significa que a tensão está alta por metade do ciclo e baixa pela outra metade. Ajustar o ciclo de trabalho altera a

Figura 5 – Vista de um controlador PWM.

Fonte: Techtonics (2023).

quantidade média de energia fornecida ao dispositivo. A Figura 6 descreve cada um dos componentes do sinal PWM.

Figura 6 – Componentes da forma de onda PWM.



Fonte: Mohamed, Elmahalawy e Harb (2013).

A frequência da onda PWM indica quantos ciclos de onda (ligado-desligado) ocorrem em um segundo. A frequência determina quão suave é o controle do dispositivo. Frequências mais altas geralmente resultam em operação mais suave, mas também podem exigir eletrônica mais complexa e suscetível a interferência eletromagnética.

2.3 VISÃO COMPUTACIONAL

Visão computacional é uma complexa área de estudo que se concentra em capacitar computadores e sistemas para interpretar e entender o mundo visual de maneira semelhante aos seres humanos. Szeliski (2010) descreve a visão computacional como um campo que não apenas tenta entender como os seres humanos visualmente percebem o ambiente ao seu redor, mas também busca replicar essas capacidades em máquinas, permitindo que elas vejam e entendam o mundo visual de forma autônoma.

Segundo Forsyth e Ponce (2011), a visão computacional é fundamentalmente sobre a interpretação de imagens. O objetivo é converter imagens em descrições do mundo que possam se integrar com outras informações e provocar ações adequadas. Isso inclui uma variedade de tarefas como reconhecimento de objetos, navegação e modelagem 3D, todas fundamentais para o desenvolvimento de sistemas inteligentes.

Gonzalez e Woods (2018) ampliam essa perspectiva ao discutir como o processamento de imagem e análise são componentes cruciais da visão computacional. Eles explicam que o processamento de imagens envolve a manipulação de imagens para melhorar sua qualidade ou extrair informações, enquanto a análise de imagem refere-se à extração de atributos estruturais que podem ser o foco de reconhecimento subsequente.

2.3.1 Espaço de Cores

O espaço de cores é uma faceta fundamental na visão computacional, fornecendo um meio para representar e manipular cores em imagens digitais. Entre os espaços de cores mais prevalentes, destacam-se o RGB (Red, Green, Blue) e o HSV (Hue, Saturation, Value), cada um com suas características e aplicações específicas.

2.3.1.1 RGB (Red, Green, Blue)

Gonzalez e Woods (2018) discutem que o modelo RGB é um dos espaços de cores mais comuns e intuitivos, baseando-se na mistura aditiva de luz. Neste modelo, as cores são representadas através da combinação das três cores primárias de luz: vermelho, verde e azul. As diferentes intensidades dessas cores primárias são combinadas para produzir uma ampla gama de cores. Este modelo é amplamente utilizado em dispositivos de captura e exibição de imagem devido à sua correspondência direta com a maneira como os sensores de câmera e os monitores funcionam. Na visão computacional, a cor é frequentemente usada como um recurso para reconhecer padrões e rastrear objetos. Jain, Flynn e Ross (2008) discutem como os componentes de cor RGB podem ser usados para diferenciar e identificar objetos dentro de uma cena. Por exemplo, o reconhecimento de sinais de trânsito, rastreamento de jogadores em um campo esportivo, ou identificação de frutas em uma linha de produção podem depender fortemente da cor. A Figura 7 demonstra os

componentes RGB, onde a imagem mais à direita mostra as componentes sobrepostas, formando a imagem final.

Figura 7 – Componentes RGB do brasão da UFSC.

Fonte: Adaptado de Identidade UFSC.

2.3.1.2 HSV (Hue, Saturation, Value)

Alvy R. Smith (1978) destaca que o modelo HSV foi projetado para se assemelhar mais à forma como os seres humanos percebem e interpretam as cores. O HSV descreve cores em termos de sua matiz (Hue), saturação (Saturation) e brilho ou valor (Value). Ao separar a informação de cor (matiz e saturação) da informação de luminosidade (valor), o HSV é particularmente útil para processamento de imagens onde as condições de iluminação podem variar, permitindo que a análise de cor seja mais consistente e robusta.

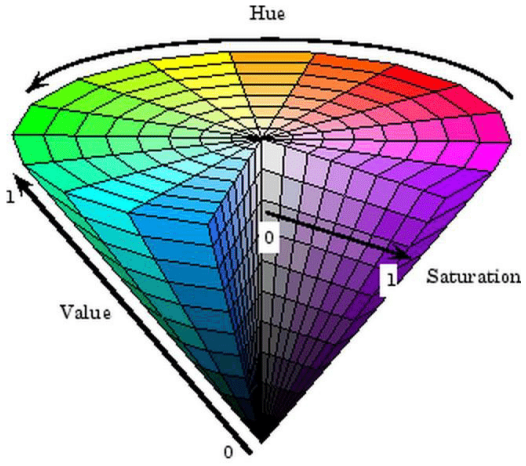
Quanto às componentes, segundo Foley, Dam e Hughes (1990), a matiz representa a cor em si, expressa como um ângulo no círculo cromático, variando de 0 a 360°, onde cada ângulo corresponde a uma cor conforme visível na Figura 8. Isso permite uma identificação de cores muito mais intuitiva e precisa em comparação com o modelo RGB. A saturação indica a intensidade ou pureza da cor. Uma saturação mais baixa significa mais presença de cinza, enquanto uma saturação mais alta indica uma cor mais pura e vívida (FOLEY; DAM; HUGHES, 1990). O brilho (valor) descreve quão claro ou escuro a cor é, variando desde o preto, no seu ponto mais baixo, ao branco, no seu ponto mais alto, com várias intensidades de cor entre elas (FOLEY; DAM; HUGHES, 1990).

Foley, Dam e Hughes (1990) discutem como a capacidade do HSV de separar informações de cor da luminosidade é útil para rastrear objetos em vídeos, onde a cor é um identificador chave, sendo essa capacidade um dos pilares deste projeto.

2.3.2 Máscaras de Imagem

A criação de máscaras é uma operação crucial em visão computacional, permitindo isolar e analisar regiões específicas de interesse em uma imagem. Vários métodos matemáticos e teorias são aplicados na criação e refinamento de máscaras, conforme descrito em diversas literaturas especializadas.

Figura 8 – Espaço de cores HSV.



Fonte: Erdoğan e Yılmaz (2014)

2.3.2.1 Segmentação por Limiarização

A segmentação por limiarização, também conhecida como *Thresholding*, é uma técnica que visa criar uma máscara que separe áreas de interesse. Gonzalez e Woods (2018) explicam que o *Thresholding* envolve a escolha (pelo projetista) de um limiar de comparação T e a classificação dos pixels da imagem com base nesse limiar. Os pixels que estão acima do limiar são designados como objetos ou áreas de interesse, enquanto os pixels abaixo do limiar são geralmente considerados como o fundo. O resultado é uma imagem binária, onde os pixels de interesse são marcados (geralmente como brancos ou 1) e o restante é desconsiderado (geralmente como pretos ou 0). Há múltiplos métodos de aplicar a segmentação, sendo os mais comuns:

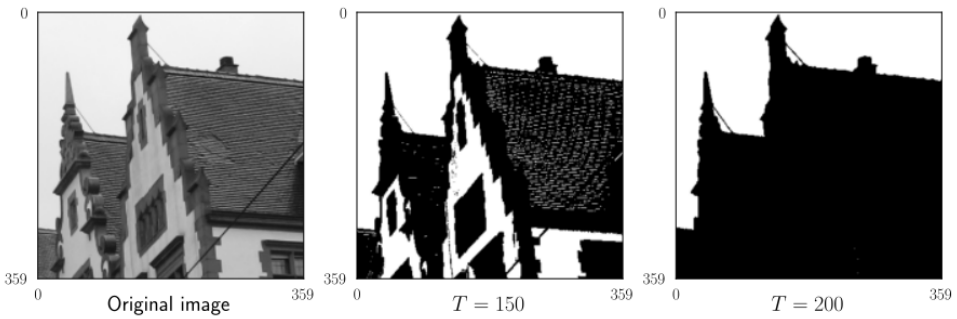
- O *thresholding* global, que aplica um único valor de limiar a toda a imagem. Este método é eficaz quando há um contraste significativo entre os objetos de interesse e o fundo (SONKA; HLAVAC; BOYLE, 2014).
- O *thresholding* adaptativo, que calcula limiares locais para diferentes regiões da imagem, sendo particularmente útil em situações onde a iluminação não é uniforme em toda a imagem (PARKER, 2010).
- O método de Otsu, que encontra automaticamente um limiar ótimo calculando o limiar que minimiza a variância intra-classe, maximizando a variância entre classes (GONZALEZ; WOODS, 2018).

A análise de um histograma torna mais simples a compreensão da técnica de segmentação por limiarização. Isso acontece porque o histograma é um gráfico que mostra a distribuição de intensidades de pixels numa imagem. Cada barra do histograma representa a frequência de uma determinada intensidade de pixel. Ao visualizar o histograma de uma

imagem, é possível rapidamente perceber onde estão concentradas as intensidades de pixel, sendo crucial para definir um limiar de segmentação apropriado, facilitando a identificação de picos e vales. Picos representam intensidades de pixel que ocorrem frequentemente e geralmente correspondem aos objetos de interesse ou ao fundo. Vales, por outro lado, são representativos de transições entre objetos e fundo. A identificação de um vale adequado entre dois picos significantes usualmente indica o local apropriado para definir o limiar de segmentação.

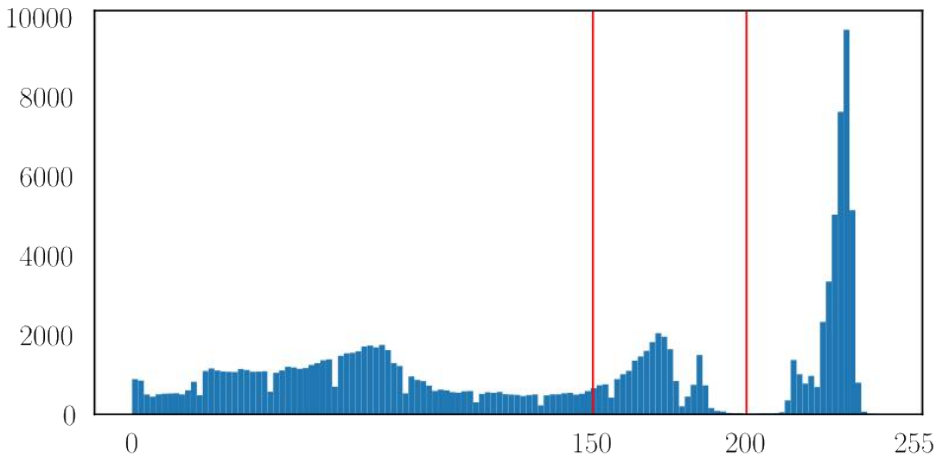
Na Figura 9 é possível visualizar exemplos de múltiplos limiares para uma mesma imagem. Tais limiares permitem segmentar de maneira simples o plano de fundo do plano frontal. O histograma para a Figura 9 pode ser encontrado na Figura 10.

Figura 9 – Exemplo de múltiplos limiares para uma imagem.



Fonte: Vincent Mazet (2023).

Figura 10 – Histograma da imagem original da Figura 9



Fonte: Vincent Mazet (2023).

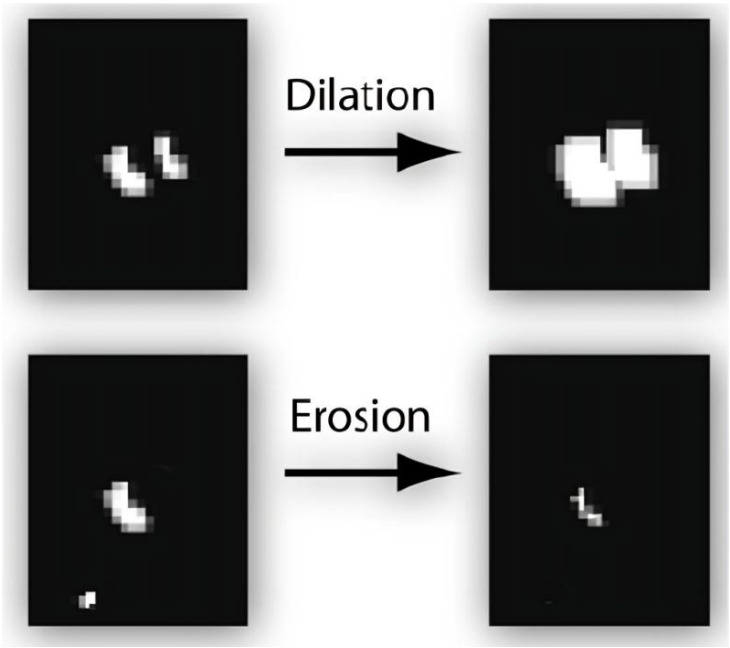
2.3.2.2 Operações Morfológicas

As operações morfológicas são um conjunto de técnicas fundamentais no processamento de imagens e visão computacional, utilizadas para extrair informações importantes sobre a estrutura e a forma dos objetos em uma imagem. Serra (1982) introduz as operações morfológicas como métodos que processam imagens com base na forma. Essas operações são aplicadas a conjuntos de pixels (geralmente binários, mas também podem ser estendidos para imagens em tons de cinza) e são definidas em termos de operadores que interagem com a imagem e um elemento estruturante, que é uma pequena forma ou padrão usado para sondar e interagir com os pixels da imagem. As duas principais operações morfológicas são erosão e dilatação, além da abertura e fechamento, que são operações resultantes da combinação de erosão e dilatação em ordens específicas.

A erosão é uma operação que "afina" os objetos em um conjunto (imagem) **A** pelo elemento estruturante **B**, removendo todos os pixels de **A** que não se encaixam completamente em **B**. Em termos práticos, isso significa que a erosão remove pixels nas bordas dos objetos, o que pode ser útil para eliminar pequenos detalhes ou separar objetos que estão ligeiramente conectados (SERRA, 1982). A Figura 11 demonstra como esse processo acontece.

A dilatação é o oposto da erosão. Ela "engrossa" os objetos em uma imagem, conforme visível na Figura 11. A dilatação de um conjunto **A** pelo elemento estruturante **B** adiciona todos os pixels que tocam **B**. Isso faz com que os objetos cresçam e possam preencher pequenos buracos ou espaços dentro de um objeto, ou conectar objetos próximos.

Figura 11 – Exemplos de operações de erosão e dilatação.



Fonte: Reda, Mateevitsi e Offord (2013).

A abertura é a erosão seguida de dilatação, suavizando os contornos dos objetos, rompendo conexões estreitas, eliminando proeminências finas ou removendo pequenos objetos (ruídos na imagem) (SERRA, 1982). Como a dilatação segue a erosão, o tamanho dos objetos (que permanecem após a erosão) não diminui significativamente, mas sua forma pode ser alterada, especialmente removendo pequenos detalhes.

O fechamento é a dilatação seguida de erosão. Ele também suaviza os contornos dos objetos, mas, ao contrário da abertura, o fechamento tende a fechar pequenos buracos e fissuras nos objetos e a conectar componentes próximos. Isso pode ser particularmente útil para preencher buracos e lacunas em objetos.

2.3.2.3 Detecção de Contornos

A detecção de contornos é uma técnica fundamental em visão computacional que envolve a identificação das bordas ou linhas que delineiam e definem as fronteiras de objetos em uma imagem. Essencialmente, contornos são as curvas que unem todos os pontos contínuos (ao longo da borda) que têm a mesma cor ou intensidade. A detecção de contornos serve a vários propósitos essenciais, entre eles a segmentação de imagens, reconhecimento de objetos, análise de formas, etc.

Um dos algoritmos mais utilizados neste meio foi proposto por Suzuki e Abe (1985). Este algoritmo é notável pela sua eficiência e precisão na detecção de estruturas de contorno. Na Figura 12 é apresentado um exemplo de detecção de contorno obtido através do algoritmo de Suzuki e Abe, onde uma esfera laranja é detectada por meio do uso da aplicação de máscaras e da técnica de contorno. Depois, é aplicado um pós-processamento que permite criar uma circunferência ao redor do contorno detectado.

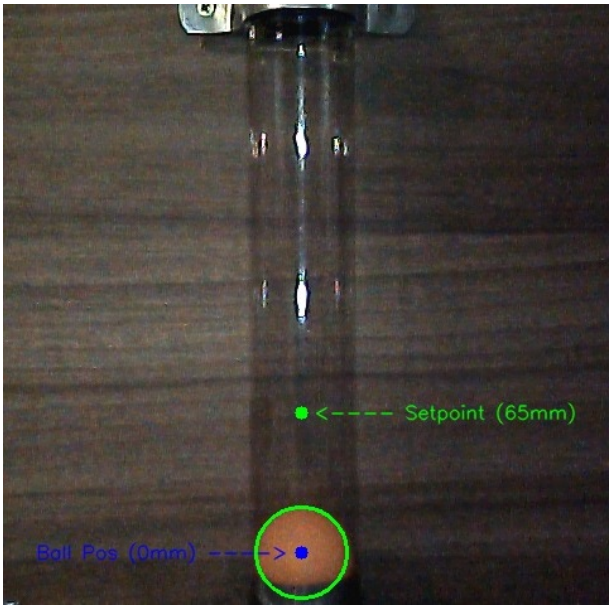
Os autores deste algoritmo focaram em uma análise topológica de imagens binárias, onde os contornos dos objetos são extraídos seguindo as bordas dos pixels. A ideia central é seguir as bordas dos objetos na imagem para identificar e isolar os contornos completos. O algoritmo utiliza um processo de seguimento de fronteira, começando de um ponto na borda do objeto e seguindo a borda até que o ponto inicial seja alcançado novamente. Durante esse processo, o algoritmo mantém um registro dos pixels visitados e sua ordem, efetivamente traçando o contorno do objeto.

Uma das contribuições significativas deste método é a capacidade de entender a hierarquia de contornos, distinguindo entre contornos externos e internos (buracos dentro dos objetos). Isso permite uma análise mais detalhada e compreensão da estrutura dos objetos na imagem.

2.4 APRENDIZADO DE MÁQUINA

O aprendizado de máquina (tradução livre de *Machine Learning*) é um ramo da inteligência artificial que se concentra no desenvolvimento de algoritmos e técnicas que

Figura 12 – Exemplos de detecção de contorno usando o algoritmo de Suzuki e Abe.



Fonte: Do autor, 2024.

permitem aos computadores aprender e melhorar com a experiência. C. M. Bishop (2007) define o aprendizado de máquina como um conjunto de métodos que podem detectar padrões em dados e usar esses padrões para prever dados futuros ou executar outras formas de tomada de decisão sob incerteza. Algoritmos de aprendizado de máquina são usualmente classificados em três categorias, a saber: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço. Os quais são abordados com mais detalhes nas seções seguintes.

2.4.1 Aprendizado supervisionado

O aprendizado supervisionado é um dos pilares fundamentais do aprendizado de máquina, onde o algoritmo é treinado em um conjunto de dados rotulado. Alpaydin (2020) descreve o aprendizado supervisionado como um processo onde o modelo é alimentado com entradas e as saídas correspondentes (ou rótulos), e o objetivo é aprender uma função que mapeie as entradas às saídas. O aprendizado supervisionado é amplamente utilizado devido à sua eficácia em uma ampla gama de tarefas. C. M. Bishop (2007) explica que uma vez treinado, o modelo pode ser usado para prever a saída de novos dados não vistos anteriormente.

Esse algoritmo é utilizado para resolver dois tipos distintos de problemas: os de classificação, com saídas discretas, e os de regressão, com saídas contínuas. Hastie, Tibshirani e Friedman (2016) discutem que nos problemas de classificação, as saídas são categorias discretas. O objetivo é prever a categoria ou classe de uma entrada. Exemplos comuns in-

cluem reconhecimento de dígitos, diagnóstico médico e detecção de e-mails spam. Murphy (2012) detalha que nos problemas de regressão, as saídas são valores contínuos. O objetivo é prever um valor numérico baseado em entradas. Exemplos incluem a previsão do preço de casas, ações ou a temperatura de um dia.

Alguns dos métodos mais comuns em aprendizado supervisionado, conforme descrito por Mitchell (1997), incluem árvores de decisão, redes neurais, SVM (Máquinas de Vetores de Suporte) e modelos lineares como regressão linear e logística.

Apesar de sua eficácia, o aprendizado supervisionado enfrenta desafios como o risco de sobreajuste (*overfitting*), onde um modelo aprende a memorizar os dados de treinamento em vez de generalizar para novos dados. Além disso, a necessidade de grandes conjuntos de dados rotulados pode ser uma limitação significativa, especialmente em domínios onde a rotulação é cara ou difícil (JORDAN; MITCHELL, 2015).

2.4.2 Aprendizado não supervisionado

O aprendizado não supervisionado é uma categoria de algoritmos de aprendizado de máquina que opera em dados não rotulados. Essa abordagem é essencial quando as informações sobre os rótulos de saída não estão disponíveis e o objetivo é descobrir a estrutura intrínseca dos dados. Em resumo, o aprendizado não supervisionado é uma forma de encontrar padrões ocultos ou agrupamentos em dados sem a orientação de um conjunto de treinamento rotulado (MURPHY, 2012). Segundo C. M. Bishop (2007), o aprendizado não supervisionado se concentra em modelos que capturam a distribuição ou a estrutura dos dados de entrada.

Algumas técnicas comuns são o agrupamento, a redução de dimensionalidade e o aprendizado de representação. Hastie, Tibshirani e Friedman (2016) explicam que o agrupamento é uma das técnicas mais comuns no aprendizado não supervisionado, onde o objetivo é dividir os dados em grupos (ou *clusters*) com base na similaridade. Métodos populares incluem K-means, agrupamento hierárquico e DBSCAN. Alpaydin (2020) descreve a redução de dimensionalidade como uma técnica usada para reduzir o número de variáveis em um conjunto de dados, preservando ao mesmo tempo a estrutura essencial. Isso é útil para visualização de dados, compressão e mitigação da maldição da dimensionalidade, um fenômeno que ocorre em problemas de análise de dados e aprendizado de máquina quando o número de variáveis ou dimensões em um conjunto de dados aumenta significativamente. Exemplos de algoritmos incluem o PCA (Análise de Componentes Principais) e o t-SNE. Goodfellow, Bengio e Courville (2016) discutem sobre o aprendizado de representação, ou *autoencoder*, ser uma técnica que busca aprender uma representação compacta dos dados. Isso pode ser usado para descompactação, remoção de ruído ou como uma etapa prévia para outras tarefas de aprendizado de máquina.

O aprendizado não supervisionado tem uma ampla gama de aplicações, conforme discutido por Jordan e Mitchell (2015), incluindo segmentação de mercado, organização

de grandes bibliotecas de documentos ou imagens, detecção de padrões anormais para detecção de fraude, entre outras. Em muitos casos, o aprendizado não supervisionado serve como uma ferramenta preliminar para entender melhor a estrutura dos dados antes de aplicar métodos de aprendizado supervisionado.

Um dos principais desafios do aprendizado não supervisionado é a falta de critérios claros para avaliar o resultado, uma vez que não existem rótulos verdadeiros para comparação (MURPHY, 2012). Além disso, a interpretação dos resultados pode ser subjetiva e dependente do contexto.

2.4.3 Aprendizado por reforço

O aprendizado por reforço é uma área do aprendizado de máquina onde um agente aprende a tomar decisões através da interação com um ambiente. Sutton e Barto (2018) descrevem o aprendizado por reforço como um processo em que um agente busca maximizar a recompensa cumulativa ao longo do tempo, tomando ações e observando os resultados dessas ações no ambiente. O aprendizado é guiado por recompensas e punições, e o agente aprende uma política que mapeia estados do ambiente para as ações que devem ser tomadas.

O agente é a entidade que aprende e toma decisões, enquanto o ambiente é o mundo externo com o qual o agente interage (RUSSELL; NORVIG, 2016).

A política é uma estratégia que o agente utiliza para determinar a próxima ação com base no estado atual do ambiente. Pode ser uma função simples ou uma estrutura de dados complexa (SUTTON; BARTO, 2018).

A recompensa é um *feedback* imediato dado ao agente após cada ação. A função de valor, por outro lado, é uma estimativa do retorno total esperado a partir de um estado ou ação ao longo do tempo (SUTTON; BARTO, 2018).

O aprendizado por reforço tem várias aplicações práticas. Mnih *et al.* (2015) demonstraram como pode ser usado para jogar *vídeo games* com desempenho sobre-humano. Outras aplicações incluem robótica, otimização de sistemas e gerenciamento de recursos. Os conceitos fundamentais envolvem conhecer o agente e o ambiente, as políticas, recompensas, função de valor e o dilema de exploração-extrapolação.

Quanto ao dilema exploração-extrapolação, Silver *et al.* (2014) discutem a situação onde o agente deve escolher entre explorar ações desconhecidas para descobrir novas recompensas ou extrapolar as ações conhecidas que já resultaram em boas recompensas.

Nesta forma de aprendizado, Sutton e Barto (2018) destacam os métodos de diferença temporal, como Q-learning e SARSA, que são capazes de aprender políticas ótimas diretamente da experiência bruta, sem um modelo do ambiente. Esses métodos atualizam as estimativas de valor com base nas diferenças temporais entre as estimativas consecutivas.

2.4.3.1 Q-learning

O Q-learning é uma técnica de aprendizado por reforço que visa encontrar a melhor política de ação para um agente, ou seja, a melhor sequência de ações a serem tomadas em um ambiente para maximizar a recompensa total. Como um método livre de modelo, o Q-learning não requer um modelo prévio do ambiente e, em vez disso, aprende a estimar o valor futuro (recompensa) de ações e estados através da experiência (SUTTON; BARTO, 2018).

A base do Q-learning está na função Q, que associa a cada par estado-ação um valor Q que representa a qualidade dessa ação no estado dado. O valor Q é uma estimativa do valor total que se pode obter, começando daquele estado, tomando aquela ação e seguindo a política ótima a partir de então (WATKINS; DAYAN, 1992). A função Q é atualizada iterativamente usando a equação de Bellman, descrita na Equação (3).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{3}$$

Cada um dos termos da equação de Bellman pode ser dito como sendo:

- $Q(s_t, a_t)$: É o valor Q atual para um dado par estado-ação. Representa a qualidade esperada da ação a_t quando no estado s_t.
- $\leftarrow$: Indica a atualização do valor Q.
- α: A taxa de aprendizado, que determina o quanto as novas informações afetarão o valor Q anterior. Valores menores fazem o aprendizado mais lento e estável, enquanto valores maiores tornam o aprendizado mais rápido e potencialmente instável.
- r_{t+1}: A recompensa imediata recebida após tomar a ação a_t no estado s_t e transicionar para o estado s_{t+1}.
- γ: O fator de desconto, que representa a importância das recompensas futuras. Um valor de γ próximo de 1 faz com que o agente valorize recompensas futuras quase tanto quanto as recompensas imediatas, enquanto um valor próximo de 0 faz com que o agente seja míope e só considere recompensas imediatas.
- $\max_a Q(s_{t+1}, a)$: O valor máximo Q para o próximo estado s_{t+1} sobre todas as ações possíveis. Isso representa a melhor recompensa futura esperada sob a política atual.
- $Q(s_t, a_t)$: O valor Q original para o estado atual e ação, antes da atualização. Este termo é subtraído para calcular a diferença entre o valor Q atualizado e o antigo (o erro da diferença temporal).

Conforme o agente interage com o ambiente e observa as recompensas resultantes de suas ações, ele atualiza os valores Q usando a diferença temporal entre a recompensa

atual e as recompensas futuras estimadas, ajustando assim suas expectativas sobre a política ótima (SUTTON; BARTO, 2018).

Um desafio no Q-learning é equilibrar a exploração de novas ações para aprender sobre elas e a exploração das ações conhecidas por serem mais recompensadoras. Uma abordagem comum é a política Epsilon-Greedy, onde, com uma pequena probabilidade epsilon, o agente escolhe uma ação aleatória (exploração), e com a probabilidade 1-epsilon, ele escolhe a ação com o maior valor Q (exploração) (MNIH *et al.*, 2015).

Por fim, sua natureza livre de modelo e capacidade de aprender diretamente da experiência tornam o Q-learning uma ferramenta poderosa para problemas onde o ambiente é complexo ou desconhecido.

2.5 COMUNICAÇÃO SERIAL

A comunicação serial assíncrona é uma técnica fundamental na transferência de dados eletrônicos, onde a informação é transmitida bit a bit ao longo de uma linha de comunicação única. Devido à sua simplicidade e eficiência em termos de cabeamento e complexidade do circuito, a comunicação serial assíncrona é amplamente utilizada em uma variedade de dispositivos e sistemas eletrônicos, desde microcontroladores simples como o ATmega328P (que está presente no Arduino UNO) até complexos sistemas de computadores (MONK, 2014).

A natureza sequencial da comunicação serial permite que ela opere eficientemente mesmo com apenas um único fio de transmissão de dados. Além disso, a comunicação serial pode ser realizada em longas distâncias, com a capacidade de operar sobre diferentes meios como fios de cobre, fibra óptica e até links sem fio (SCHERZ; MONK, 2013).

2.5.1 Protocolo UART

O protocolo UART (Universal Asynchronous Receiver/Transmitter) é amplamente utilizado em sistemas embarcados e microcontroladores. Segundo Valvano (2019), este protocolo oferece uma forma eficiente de comunicação serial assíncrona, permitindo a transmissão de dados entre dispositivos através de apenas dois fios. Sua simplicidade e robustez o tornam popular em uma variedade de aplicações, desde microcontroladores simples até sistemas complexos de comunicação serial. Além disso, sua operação assíncrona elimina a necessidade de um sinal de *clock* compartilhado entre dispositivos, proporcionando flexibilidade em diferentes configurações de velocidade de comunicação (VALVANO, 2019).

2.5.2 Taxa de Baud

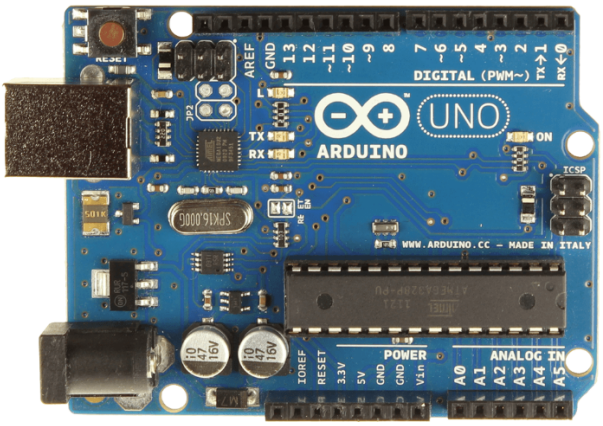
A taxa de baud, também chamada de *baudrate*, é uma medida crucial na comunicação serial que determina a velocidade da comunicação, representando o número de bits

transmitidos por segundo. Ajustar a taxa de baud corretamente é essencial para garantir uma comunicação eficaz e livre de erros entre os dispositivos (MARGOLIS, 2011). Em muitos sistemas, um símbolo pode corresponder a um bit, mas em sistemas mais avançados, um único símbolo pode representar vários bits (HAYKIN; MOHER, 2009). O termo *baud* é uma homenagem a Émile Baudot, um importante inventor na área de telegrafia, que desenvolveu um dos primeiros métodos de transmissão de dados: o código Baudot, um precursor direto das tecnologias de comunicação digital modernas (STALLINGS, 2013).

2.5.3 Comunicação Serial no Arduino

Arduino é uma plataforma de prototipagem eletrônica de código aberto composta por hardware e software simples e flexíveis. Ele foi projetado para tornar a eletrônica mais acessível a artistas, designers, hobbyistas e qualquer pessoa interessada em criar objetos ou ambientes interativos. O hardware do Arduino é uma placa de desenvolvimento com um microcontrolador, conforme visível na Figura 13, contendo entradas e saídas digitais e analógicas, além de outras interfaces. O software é uma IDE (Integrated Development Environment) que permite programar o microcontrolador usando uma linguagem de programação simples baseada em C/C++ (BANZI; SHILOH, 2014).

Figura 13 – Placa de desenvolvimento Arduino Uno R3.



Fonte: D&D Componentes (2023).

A comunicação serial via USB (Universal Serial Bus), também chamado de Barramento Serial Universal, em tradução livre, é uma das formas mais comuns de enviar e receber dados entre um Arduino e um computador ou outros dispositivos. Quando um Arduino está conectado a um computador através de USB, ele pode enviar dados para o computador para fins de monitoramento e diagnóstico ou receber comandos do computador para controlar motores, módulos e outros dispositivos (MONK, 2014). O Arduino utiliza chips específicos para converter USB para Serial, tais como o ATmega16U2. Especificamente, o protocolo serial utilizado é o UART.

2.5.4 Código G (G-Code)

O código G é uma linguagem de programação que é usada para controlar máquinas automatizadas, como impressoras 3D e máquinas-ferramentas de CNC (Comando Numérico Computadorizado). Essa linguagem é composta de comandos que instruem a máquina sobre como se mover, que caminho seguir, onde parar e outras operações relacionadas. Cada comando, começa com a letra G seguida por um número que especifica a ação a ser realizada (MATTSON, 2012). Um exemplo deste código pode ser visto na Figura 14, onde a letra F no comando G1 indica o *feedrate*, isto é, a velocidade de movimentação da máquina.

Figura 14 – Exemplo de código G para movimento espacial.

```
1 G1 X12.345 Y67.890 F1500
2 G1 X23.456 Y45.678
3 G1 X34.567 Y23.456
4 G1 X45.678 Y1.234
5 G1 X56.789 Y23.456
6 G1 X67.890 Y45.678
7 G1 X78.901 Y67.890
8 G1 X89.012 Y89.012
9 G1 X100.123 Y110.234
```

Fonte: Do autor, 2024.

Além dos movimentos básicos, o código G também pode incluir instruções para outras funções da máquina, como controlar a temperatura, a velocidade do eixo e outras configurações específicas do processo. Essa capacidade de controlar finamente as máquinas torna o código G uma ferramenta poderosa em uma variedade de aplicações industriais e de engenharia (KRAR, 2000).

2.6 CÂMERAS DIGITAIS

As câmeras digitais representam uma evolução significativa na forma como capturamos, armazenamos e compartilhamos imagens. Esses dispositivos funcionam convertendo luz em sinais eletrônicos por meio de um sensor, geralmente CCD (Charge-Coupled Device) ou CMOS (Complementary Metal-Oxide-Semiconductor), que depois são processados para formação de imagens digitais (LANGFORD; FOX; SMITH, R. S., 2015). Esses sensores são responsáveis por captar a luz e convertê-la em sinais elétricos, sendo que a qualidade e o tamanho do sensor são fatores críticos que afetam a qualidade da imagem, incluindo a resolução, a capacidade de capturar detalhes e o desempenho em condições de pouca

luz. Essa tecnologia permite não só uma revisão instantânea das imagens capturadas mas também um armazenamento e compartilhamento eficientes.

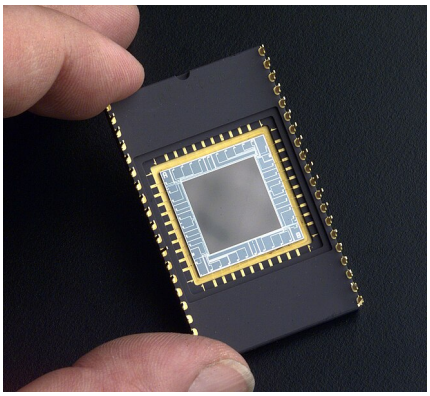
A unidade megapixel (MP) é amplamente utilizada para descrever a resolução de câmeras digitais e sensores de imagem. Um megapixel equivale a um milhão de pixels, e o número de megapixels de uma câmera indica quantos pixels individuais compõem o sensor da câmera. Langford, Fox e R. S. Smith (2015) explicam que quanto maior o número de megapixels, maior a resolução da imagem, permitindo capturar mais detalhes e possibilitando impressões maiores sem perda de qualidade.

2.6.1 Sensores CCD

Os sensores CCD são uma das tecnologias de imagem mais predominantes em câmeras digitais, telescópios, *scanners* e outros dispositivos ópticos. Eles operam convertendo a luz em cargas elétricas, que são então processadas para formar uma imagem digital. O CCD é composto por muitos pixels sensíveis à luz, onde cada pixel captura um ponto de luz da imagem e, juntos, criam a imagem completa (HOLST; LOMHEIM, 2011).

Uma das características notáveis dos sensores CCD é sua alta qualidade de imagem e sensibilidade à luz. Comparados a outras tecnologias de sensor, como CMOS, os CCDs são frequentemente reconhecidos por produzir ruído de imagem mais baixo, o que os torna particularmente valiosos em situações de baixa iluminação ou onde a qualidade da imagem é crítica, como em astrofotografia e outras aplicações científicas (JANESICK, 2001). A Figura 15 mostra um sensor CCD utilizado pela NASA, permitindo que os cientistas estudem o ultravioleta extremo, uma das partes menos exploradas do espectro eletromagnético.

Figura 15 – Sensor CCD para detecção de luz no espectro visível e ultravioleta.

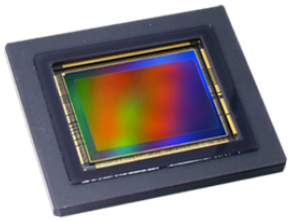


Fonte: NASA (2006).

2.6.2 Sensores CMOS

Os sensores CMOS são um tipo fundamental de sensor de imagem usado em câmeras digitais, incluindo webcams, smartphones e câmeras profissionais. Eles são conhecidos por sua eficiência energética e capacidade de integrar funções de processamento de imagem diretamente no chip, tornando-os uma escolha popular em muitas aplicações. Estes sensores operam convertendo a luz em elétrons com a ajuda de semicondutores. Cada pixel em um sensor CMOS tem seus próprios circuitos de amplificação, conversão de carga em tensão, e às vezes circuitos para outras funções de processamento. Esta arquitetura permite que os sensores CMOS tenham velocidades de leitura muito rápidas e sejam mais versáteis em termos de manipulação de imagem (HOLST; LOMHEIM, 2011). A Figura 16 mostra um sensor CMOS de alta resolução, fabricado pela empresa multinacional japonesa Canon, com finalidade industrial.

Figura 16 – Sensor CMOS de 120MP.



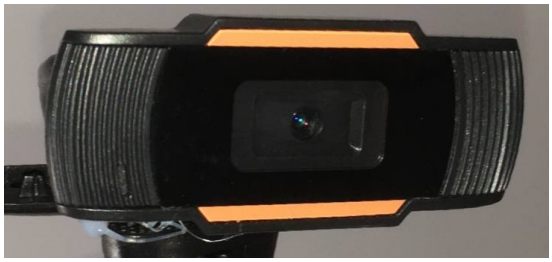
Fonte: Canon (s.d.).

2.6.3 Câmeras VGA

As câmeras VGA (Video Graphics Array) representam uma categoria de dispositivos de captura de imagem que operam com uma resolução específica de 640x480 pixels, o que equivale a cerca de 0.3 megapixels. Essa especificação vem do padrão VGA, que foi originalmente desenvolvido para monitores de computador, mas também se aplica à resolução de câmeras e outros dispositivos de imagem (LEWELL, 2010).

Apesar de sua resolução relativamente baixa em comparação com as câmeras modernas de alta resolução, as câmeras VGA têm seu lugar em muitas aplicações, especialmente devido ao seu custo relativamente baixo e requisitos modestos de processamento e armazenamento. Elas são particularmente úteis em situações onde a alta resolução não é necessária ou onde a economia de custos é uma prioridade (LEWELL, 2010). Isso pode incluir sistemas de monitoramento de segurança, dispositivos eletrônicos de consumo básicos, e iniciativas educacionais que requerem tecnologia acessível. Um exemplo de câmera VGA pode ser visto na Figura 17.

Figura 17 – Câmera VGA com sensor CMOS.



Fonte: Do autor, 2024.

3 PROPOSTA E ARQUITETURA DA SOLUÇÃO

Como introduzido no Capítulo 2, os métodos tradicionais de *autotuning* de controladores, como o método de Ziegler-Nichols, embora amplamente utilizados, enfrentam limitações significativas. Eles geralmente requerem conhecimento prévio da planta e podem não ser eficazes em sistemas dinâmicos, não lineares ou em ambientes em constante mudança. Além disso, esses métodos podem não lidar bem com incertezas e podem exigir reajustes manuais frequentes, o que é inviável em muitos cenários industriais modernos.

Uma solução promissora para superar as limitações dos métodos tradicionais de *autotuning* é a aplicação de técnicas de aprendizado por reforço, especificamente o Q-Learning. O uso deste método para aplicar a sintonia automática de controladores representa uma abordagem avançada que combina métodos de inteligência artificial com controle de processos. Em vez de depender de métodos tradicionais, que frequentemente requerem um conhecimento prévio da planta e podem não lidar bem com sistemas altamente dinâmicos ou não lineares, o aprendizado por reforço oferece uma estratégia adaptativa e flexível.

Neste trabalho, o sistema foi considerado, com base em experimentação prática, como sendo não-linear e invariante no tempo. A não-linearidade do sistema exige uma abordagem de controle sofisticada e, por isso, é justificada a utilização de um mecanismo de controle adaptativo que realize a otimização dos ganhos do controlador para manter eficácia no controle.

No aprendizado por reforço aplicado ao *autotuning*, o controlador ou o sistema de controle é considerado como um agente que interage com o ambiente (neste caso, a planta). O objetivo do agente é aprender a melhor política de ação (ou seja, a melhor configuração dos parâmetros do controlador) para maximizar uma recompensa ao longo do tempo. Essa recompensa é tipicamente definida em termos de desempenho do sistema, como minimização do erro, estabilidade e rapidez na resposta. No contexto deste trabalho, a recompensa é definida nos termos de minimização do erro, uma vez que manter a posição o mais próximo do ponto definido (*setpoint*) é fundamental para atingir o objetivo do sistema.

Para a utilização de visão computacional, faz-se fundamental possuir uma interface gráfica. Tal interface deve mostrar a esfera e sua posição atual, permitindo o monitoramento direto e intuitivo do comportamento do sistema. Isso é essencial para verificar se a esfera está seguindo a trajetória desejada e reagindo conforme esperado aos comandos do controlador. Exibir tanto a posição atual quanto o *setpoint* é crucial para avaliar a precisão do sistema de controle, permitindo identificar rapidamente desvios e ajustar o sistema, se necessário.

Gráficos do erro, do sinal de controle e da posição ao longo do tempo são informações necessárias para análise de desempenho. Tais curvas fornecem uma visão clara de como o sistema responde a diferentes condições de operação e ajudam a identificar padrões,

como oscilações ou instabilidades. Com os dados visuais e gráficos disponíveis, é mais fácil diagnosticar problemas e ajustar os parâmetros do controlador PID, bem como os algoritmos de aprendizado por reforço. Além disso, ajustes finos podem ser feitos de maneira mais informada e precisa.

Para fins de análise pós-execução, o sistema deve também contar com um sistema de registro de dados (*Data Logging*). Coletar e exportar tais dados permite uma avaliação aprofundada do desempenho do sistema. Após a execução, é possível analisar como o sistema reagiu em diferentes situações e identificar padrões de comportamento que podem não ser imediatamente evidentes durante a operação em tempo real. Além disso, tais dados podem ser fundamentais para diagnóstico de problemas com o controlador ou com a planta, uma vez que se ocorrerem falhas ou desvios no comportamento esperado, os dados coletados oferecem informações valiosas para o diagnóstico.

3.1 IMPLEMENTAÇÃO DA ESTRUTURA FÍSICA

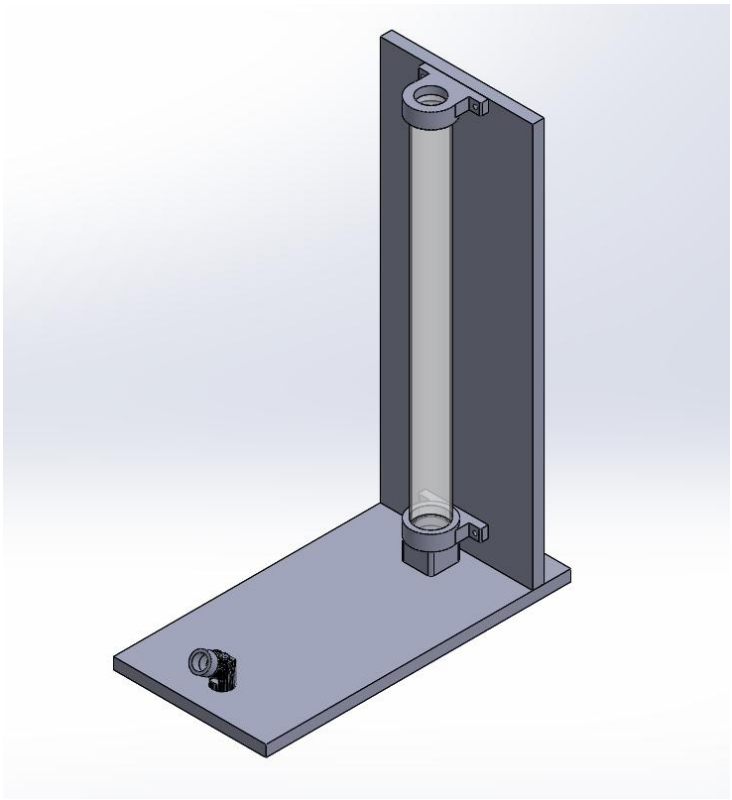
A estrutura física é responsável por permitir que a plataforma se mantenha estável e permita replicabilidade dos experimentos. Uma plataforma demasiada frágil poderia não funcionar de maneira adequada e, por essa razão, desenvolver um projeto físico robusto é fundamental para garantir a confiabilidade e a precisão dos resultados. Se a estrutura for bem projetada, minimizar-se-á as influências de vibrações e perturbações externas, assegurando a precisão na coleta de dados e no controle da posição da esfera. Essa consistência é vital para a validade científica dos resultados, pois assegura que os dados refletem as capacidades do sistema de controle e não variáveis externas. Portanto, o desenvolvimento de uma estrutura física robusta é tão essencial para o sucesso do projeto quanto o desenvolvimento do software e do controle.

O projeto da estrutura foi desenvolvido em software CAD (Computer-aided Design), conforme mostra a Figura 18.

A lista de itens utilizados na estrutura é composta por:

- **Base de Madeira:** Uma base na cor preta construída em chapa de fibra de madeira, possuindo: uma base horizontal de 40 cm de comprimento, 20 cm de largura e 16 mm de espessura; e uma base vertical de 55 cm de altura, 20 cm de largura e 16 mm de espessura. As chapas são fixadas entre si por meio do uso de um suporte de prateleira (mão francesa).
- **Esfera de Isopor:** Uma esfera de isopor preenchida com borracha e silicone acético de 35 mm de diâmetro e peso aproximado de 20g. Possuir uma cor que contraste com o fundo preto é fundamental para que seja mais visível ao sistema de visão computacional, por isso, a esfera foi mantida na cor branca.
- **Tubo de Acrílico:** Responsável por servir como guia linear para a esfera. Possui 50 cm de comprimento, 40 mm de diâmetro externo e 36 mm de diâmetro

Figura 18 – Projeto feito em CAD.



Fonte: Do autor, 2024.

interno.

- **Câmera VGA:** Possuindo baixo custo, a câmera VGA entrega resolução de 480p (640x480) a 30 FPS (Frames per second), isto é, um *frame* a cada 33 ms.
- **Monopé fotográfico:** Também conhecido como Bastão de Selfie, o monopé fotográfico é utilizado como suporte para a câmera VGA.
- **Micro-ventilador (*Cooler*) de Servidor:** Padrão 40x40mm e 28mm de altura, tensão nominal de 12 V, corrente nominal de 0,85 A, com rolamento no eixo e rotação máxima de 20.000 RPM, permitindo um fluxo de ar de 22,37 CFM. Além disso, possui um encoder, que permite saber a velocidade atual, além de um pino para controle PWM.
- **Arduino Uno R3:** O Arduino, dispositivo com um microcontrolador, é responsável por controlar o chaveamento PWM do *cooler* recebendo sinais seriais pelo barramento USB.
- **Suportes impressos em 3D:** Alguns suportes foram modelados em software CAD e impressos em impressora 3D no material PLA. Estes são utilizados para fixar tanto o tubo de acrílico quanto o monopé na estrutura de madeira.

Para montar a estrutura, as bases de madeira são presas conforme mencionado acima por meio do uso dos suportes de prateleira. O tubo de acrílico recebe os dois suportes nas extremidades, sendo que na extremidade inferior também é preso o micro-ventilador. Assim, o tubo é fixado por seus suportes na chapa vertical. O suporte para o monopé deve ser preso na base horizontal de forma a ficar na extremidade oposta à base vertical. Isso permitirá que a câmera tenha imagem completa do sistema. O monopé deve então ser acoplado ao seu suporte. Adesivo termoplástico, chamado popularmente de cola quente, pode ser utilizado para fixar o monopé de maneira que ele não se movimente com facilidade. A câmera pode então ser presa, também com adesivo termoplástico, ao monopé. O Arduino, *jumpers* e demais conexões são realizadas e posicionadas na parte traseira do dispositivo.

Ao final, a estrutura deve ter o formato representado na Figura 19.

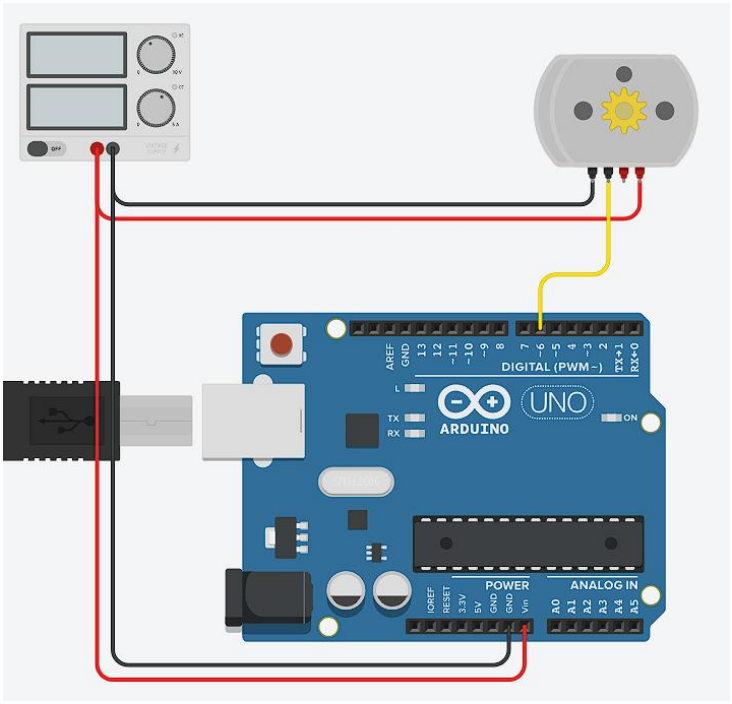
Figura 19 – Estrutura física completamente montada.

Fonte: Do autor, 2024.

3.1.1 Circuito eletrônico

O circuito do projeto envolve o Arduino Uno R3, uma fonte externa de 12V e 3A e o micro-ventilador, todos conectados entre si por meio de jumpers. O circuito pode ser visto na Figura 20.

Figura 20 – Circuito eletrônico do projeto.



Fonte: Do autor, 2024.

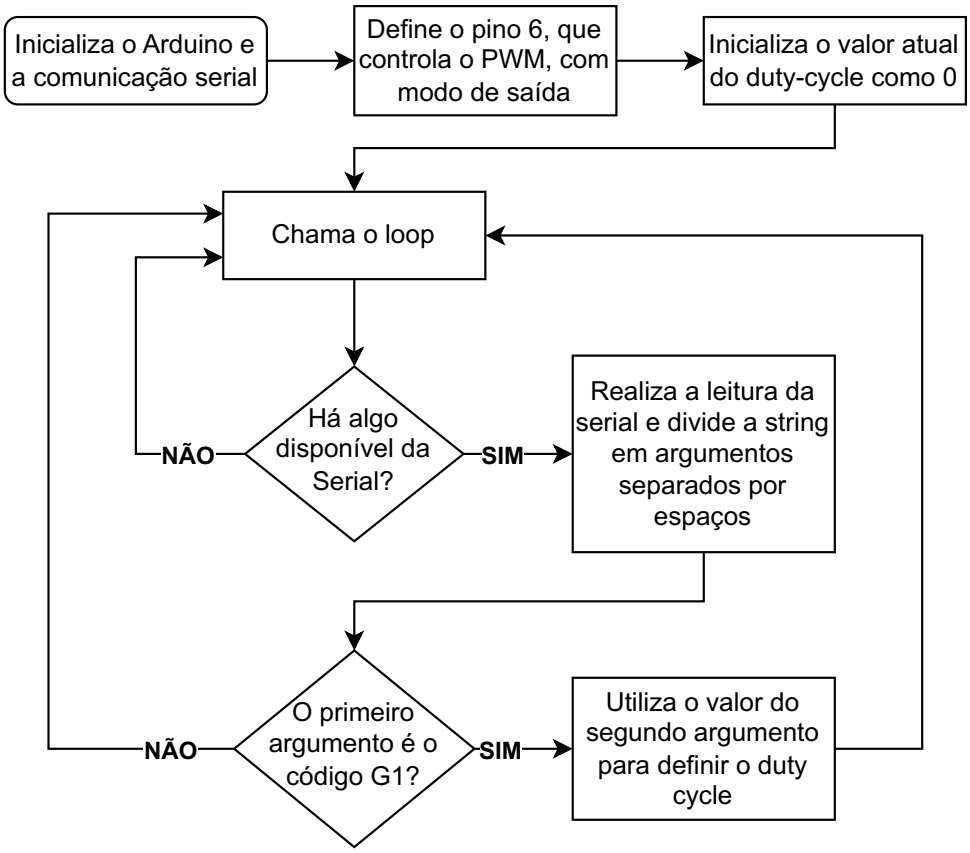
Os fios vermelhos indicam a linha de 12V, enquanto os fios pretos indicam a linha do GND (*Ground*). O fio amarelo é a saída do Arduino, onde tem-se um sinal PWM, cuja alteração do *duty-cycle* controla a rotação do motor do micro-ventilador. O último terminal do micro-ventilador não é conectado, uma vez que ele apenas indica a velocidade de rotação do motor, sendo tal informação não utilizada neste trabalho.

3.2 ARQUITETURA DO SOFTWARE

O software neste projeto foi dividido em duas partes, desenvolvidas com seus propósitos específicos. De um lado, o Arduino possui um código específico para realizar o controle do sinal PWM do micro-ventilador, intitulado *Fan Drive*. Esse código recebe um código G que indica qual *duty-cycle* deve ser aplicado à porta de controle. O fluxograma da Figura 21 demonstra como é feito o recebimento e controle. O código G foi escolhido para ser utilizado por conta da facilidade na implementação por meio de bibliotecas já existentes, além de proporcionar uma comunicação estruturada, prevenindo erros na transmissão. A

taxa de transmissão (*baudrate*) utilizada na comunicação entre o Arduino e o computador foi de 115200 bauds. Esse valor foi definido para garantir uma comunicação rápida em curta distância.

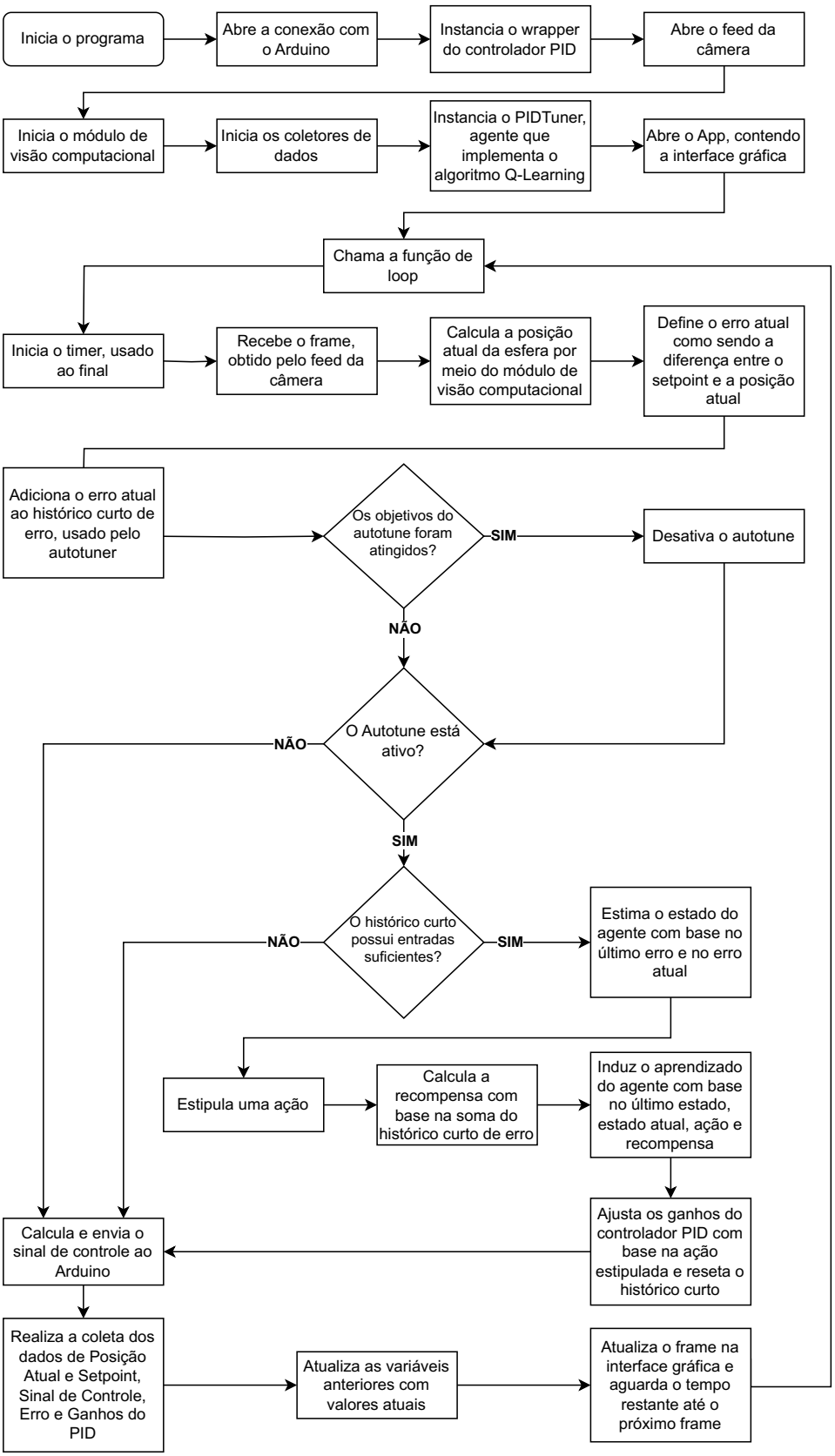
Figura 21 – Fluxograma da lógica seguida no *Fan Drive* do Arduino.



Fonte: Do autor, 2024.

De outro lado, em um computador, executa-se o software desenvolvido em Python, responsável pelo processamento do algoritmo de visão computacional, cálculo do controlador PID, aprendizado por reforço com algoritmo Q-Learning e envio do sinal de controle via comunicação serial assíncrona ao Arduino. O fluxograma da Figura 22 descreve a lógica de funcionamento desse algoritmo.

Figura 22 – Fluxograma da lógica seguida no software de controle.



Fonte: Do autor, 2024.

3.3 DESENVOLVIMENTO DO SISTEMA DE CAPTURA

Uma vez que a estrutura física está devidamente montada, o sistema de captura deve ser desenvolvido englobando os conceitos suprarreferidos quanto à visão computacional.

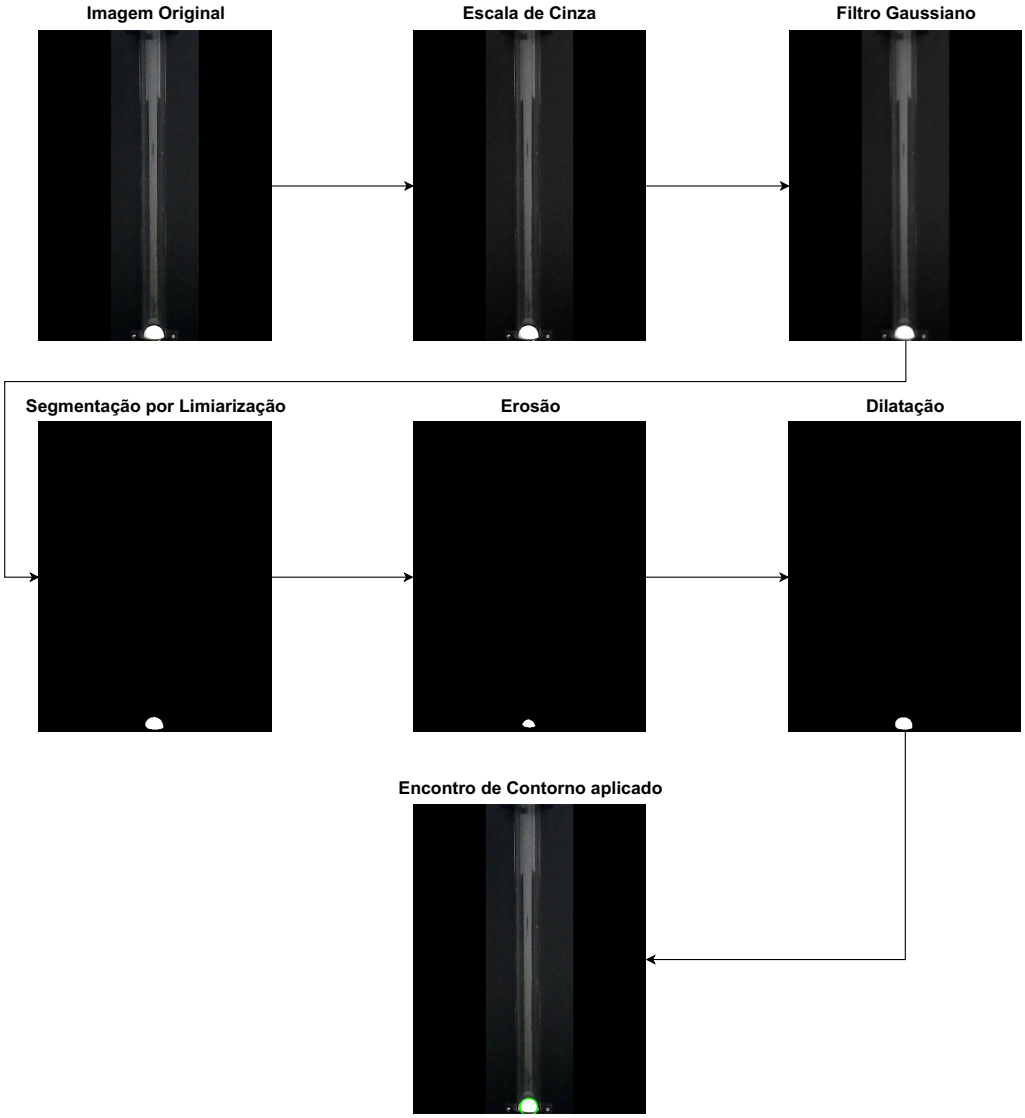
Para facilitar a implementação do sistema, o uso da biblioteca OpenCV (OPENCV, 2024) se faz fundamental, uma vez que abstrai conceitos avançados de visão computacional em funções simples para manipulação de imagens. O sistema permite, inclusive, realizar a abertura do *feed* da câmera, isto é, permite coletar em tempo real os *frames* de vídeo recebidos através da porta USB.

A primeira etapa consiste na coleta e conversão da imagem, onde realiza-se um laço de repetição contínuo para que os *frames* de vídeo sejam processados sempre que disponíveis no barramento. As imagens são então convertidas para o padrão de cores *HSV*, uma vez que este padrão facilita a detecção de cores, e são enviadas para serem pré-processadas na segunda etapa.

A segunda etapa consiste em pré-processar a imagem para a detecção de contornos. Por isso, converte-se a imagem para escala de cinza, aplica-se um filtro gaussiano para remover ruídos e cria-se uma máscara através do processo de segmentação por limiarização. A esta máscara, aplicam-se operações morfológicas de erosão e dilatação, isto é, uma operação de abertura, removendo ruídos e preparando para a detecção de contorno. Dessa forma, realiza-se a detecção utilizando o método de Suzuki e Abe (1985), implementado na biblioteca OpenCV. Por fim, é então calculada a posição do centro da esfera e de seu raio, permitindo que seja feito um desenho da circunferência da esfera com centro na posição encontrada. O diagrama que descreve este processo pode ser visto na Figura 23. A Figura 24 demonstra como a interface gráfica deve mostrar o resultado do processamento implementado.

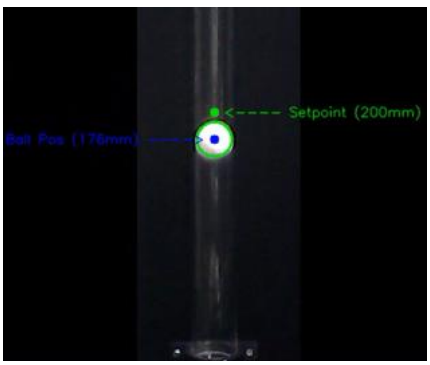
Com a posição da esfera calculada em pixels, é possível convertê-la para milímetros usando uma calibração simples, onde, sabendo a distância real em milímetro medida entre dois pontos, calcula-se quantos pixels distam entre estes pontos. Esse cálculo, porém, não é preciso, uma vez que distorções são geradas pela lente da câmera e a correção desta depende do campo de visão da câmera, abertura da lente e outros fatores, que para simplificação do projeto serão desconsiderados.

Figura 23 – Diagrama do processo de tratamento da imagem e detecção de contorno.



Fonte: Do autor, 2024.

Figura 24 – Esfera contornada na interface gráfica.



Fonte: Do autor, 2024.

3.4 COMUNICAÇÃO SERIAL ASSÍNCRONA

A comunicação serial neste projeto foi dividida em duas partes: o lado receptor, que consiste em um código no Arduino que se comunica com o micro-ventilador, e o lado transmissor, incluído no software de controle desenvolvido em Python. A classe `ArduinoSerial` é uma implementação simples em Python para estabelecer e controlar uma conexão serial com um dispositivo Arduino. Ela utiliza a biblioteca pySerial (CHRIS LIECHTI, 2020) para comunicação serial e é projetada para simplificar a tarefa de enviar comandos ou dados para o Arduino através da porta serial. O construtor da classe é chamado quando um objeto da classe é instanciado, recebendo a porta e o *baudrate* como parâmetros. O parâmetro `porta` define qual porta no sistema será utilizada para a comunicação serial. O parâmetro `baudrate` define a taxa de transmissão de dados na porta serial. Assim que esse construtor é executado, a conexão é criada.

O método `write_serial()` permite enviar dados ou comandos para o Arduino através da conexão serial ora estabelecida. Ele recebe um parâmetro `text`, que é a string de texto que deseja-se enviar pela porta serial. O método converte essa string em bytes usando a codificação `UTF-8` e acrescenta um caractere de nova linha (`\n`) no final, uma vez que a implementação no lado do Arduino espera que os comandos terminem com uma nova linha. Depois de preparar os dados, o método utiliza o objeto `conn` (que representa a conexão serial) para enviar os bytes da string para o Arduino.

No geral, a classe `ArduinoSerial` encapsula a funcionalidade de comunicação serial com um Arduino em um objeto Python reutilizável.

3.5 SISTEMA DE CÁLCULO DO PID

O controlador PID foi abstraído na classe `PIDController`, que o implementa em Python por intermédio da biblioteca simple-pid (MARTIN LUNDBERG, 2023). Tal classe possui um construtor, que configura os ganhos do controlador e recebe a instância Serial que é usada para se conectar ao Arduino. Nele, os limites de saída (0 a 255, valor mínimo e máximo das saídas PWM do microcontrolador) e o valor de referência desejado (*setpoint*), que representa a posição alvo que o sistema deve atingir, também são definidos. A escolha da biblioteca simple-pid foi a garantia de uma implementação simples e, sobretudo, já consolidada, uma vez que o objetivo não era o desenvolvimento de um sistema controlador, mas sim, de controlar um sistema físico.

O método `get_controller()` permite que outros componentes do código obtenham uma referência ao objeto controlador PID configurado. Isso é útil uma vez que há a necessidade de acessar e modificar os parâmetros do controlador durante a execução do programa.

O método `send_output()` é fundamental para o funcionamento do controlador. Ele recebe a posição atual do sistema como entrada e calcula o sinal de controle com base

nos parâmetros do controlador PID. O sinal de controle calculado representa o esforço necessário para mover a saída do sistema em direção ao valor de referência. Após calcular o sinal de controle, o método ainda envia esse valor para o Arduino por meio da instância da comunicação serial referida no construtor da classe. O sinal de controle é convertido em um código G apropriado no formato `G1 [valor]`, onde `[valor]` representa o valor calculado do sinal de controle. Esse comando é enviado ao Arduino para executar a ação de controle no sistema.

3.6 IMPLEMENTAÇÃO DO Q-LEARNING PARA *AUTOTUNING*

A classe `QLearningTuner` é uma implementação em Python do algoritmo de Q-Learning com o propósito de ajustar os parâmetros de um controlador PID de forma autônoma.

Conforme o fluxograma da Figura 22, o *autotune* possui um objetivo a ser definido. No espectro deste trabalho, o objetivo foi manter o erro médio (calculado em um período curto de 180 medidas) menor do que 10mm. Isso indicaria que o sistema está em um estado estável o suficiente e, portanto, considera-se que atingiu o seu objetivo.

O construtor da classe `QLearningTuner` recebe vários parâmetros, incluindo o espaço de estados (`state_space`), com 25 estados, o espaço de ações (`action_space`), com 27 ações possíveis, o objeto do controlador PID previamente configurado, a taxa de aprendizado (LR), definida para 0.8, o fator de desconto (DF), definido para 0.7, e a taxa de decaimento de exploração, que faz com que a taxa de exploração decaia em 0.09 por ciclo de aprendizado.

Um atributo chamado `q_table` é inicializado como uma matriz de zeros com dimensões `state_space` x `action_space`. Esta tabela armazena os valores Q, que representam a utilidade esperada de tomar uma determinada ação em um estado específico.

O método `learn()` é responsável por atualizar a tabela Q com base no *feedback* recebido do ambiente. Ele calcula um valor de predição com base na tabela Q e no último estado, um valor de objetivo com base na recompensa obtida e no estado atual, e, em seguida, atualiza o valor Q para o estado e a ação específicos. A recompensa neste algoritmo foi definida como sendo a média do erro, multiplicado pelo desvio padrão, multiplicado pela média, todos com 180 medições. Esse resultado é passado de forma a ser uma recompensa negativa e, portanto, o algoritmo procura evitar valores altos de erro. A etapa de treinamento é acompanhada do decaimento do fator de exploração, responsável por reduzir a exploração ao longo do tempo, e do algoritmo de Epsilon-Greedy, utilizado para balancear a exploração e extrapolação.

O método `choose_action()` decide qual ação tomar em um determinado estado, com base na estratégia de exploração e extrapolação. Se a estratégia de *autotune* estiver ativa, ele pode deve escolher uma ação aleatória com base no espaço de ações. Se estiver desativado, a ação seria o valor máximo da tabela Q para o estado atual.

O método `get_state()` recebe dois erros do controlador PID, erro atual e erro anterior, determinando o estado do sistema para o Q-Learning. Ele categoriza o estado com base no tamanho do erro e na tendência do erro (aumentando ou estável/decrescendo), retornando um valor que representa o estado.

O fator de estado de erro (α, escolhida arbitrariamente) é determinado pelas seguintes condições:

$$\alpha = \begin{cases} 4, & \text{se erro atual } \leq 10\text{mm (Erro muito baixo)} \\ 3, & \text{se erro atual } \leq 15\text{mm (Erro baixo)} \\ 2, & \text{se erro atual } \leq 30\text{mm (Erro moderado)} \\ 1, & \text{se erro atual } \leq 50\text{mm (Erro alto)} \\ 0, & \text{qualquer outro caso (Erro muito alto)} \end{cases} \tag{4}$$

A tendência do erro (β, escolhida arbitrariamente) depende da proporção do erro, calculado entre a média do erro atual dividido pela média do erro anterior. Seu comportamento é determinado pelas seguintes condições:

$$\beta = \begin{cases} 4, & \text{se a proporção } < 0.8 \text{ (Erro decrescendo rapidamente)} \\ 3, & \text{se a proporção } < 0.95 \text{ (Erro decrescendo)} \\ 2, & \text{se a proporção } \leq 1.05 \text{ (Erro estável)} \\ 1, & \text{se a proporção } \leq 1.2 \text{ (Erro crescendo)} \\ 0, & \text{qualquer outro caso (Erro crescendo rapidamente)} \end{cases} \tag{5}$$

A Equação (6) descreve qual será o estado resultante (γ, escolhida arbitrariamente) com base nas informações coletadas.

$$\gamma = 2\alpha + \beta \tag{6}$$

Por fim, o método `tune_pid` ajusta os parâmetros do controlador PID com base na ação tomada pelo algoritmo Q-Learning. Ele recebe um índice de ação, que corresponde a uma das ações possíveis definidas no método e representada na matriz (7). Com base nesse índice, ele ajusta um ou mais ganhos do controlador para otimizar o desempenho do sistema.

$$\mathbf{A} = \begin{bmatrix} (-1 & -1 & -1) & (-1 & 0 & -1) & (-1 & 1 & -1) \\ (-1 & -1 & 0) & (-1 & 0 & 0) & (-1 & 1 & 0) \\ (-1 & -1 & 1) & (-1 & 0 & 1) & (-1 & 1 & 1) \\ (0 & -1 & -1) & (0 & 0 & -1) & (0 & 1 & -1) \\ (0 & -1 & 0) & (0 & 0 & 0) & (0 & 1 & 0) \\ (0 & -1 & 1) & (0 & 0 & 1) & (0 & 1 & 1) \\ (1 & -1 & -1) & (1 & 0 & -1) & (1 & 1 & -1) \\ (1 & -1 & 0) & (1 & 0 & 0) & (1 & 1 & 0) \\ (1 & -1 & 1) & (1 & 0 & 1) & (1 & 1 & 1) \end{bmatrix} \tag{7}$$

As equações que definem cada um dos ganhos são dadas por:

$$\begin{aligned} Kp &= Kp_{atual} + \mathbf{A}[i][j][k] \cdot 0.01 \\ Ki &= Ki_{atual} + \mathbf{A}[i][j][k] \cdot 0.01 \\ Kd &= Kd_{atual} + \mathbf{A}[i][j][k] \cdot 0.01 \end{aligned} \tag{8}$$

onde $\mathbf{A}[i][j]$ corresponde ao conjunto de uma ação na matriz de ações, e o parâmetro k varia de 0 a 2 para cada um dos ganhos receber uma das ações.

3.7 IMPLEMENTAÇÃO DO SISTEMA DE REGISTRO DE DADOS

O `DataLogger` é uma classe em Python projetada para registrar dados ao longo do tempo. Sua função principal é coletar, armazenar e salvar os dados em um formato de arquivo CSV (Comma-separated Values) ao final do experimento, garantindo que os dados requisitados estejam disponíveis para análise posterior. A Figura 25 exemplifica a estrutura os dados de um CSV produzido pelo datalogger.

A classe `DataLogger` é inicializada com um nome único. Seu construtor cria uma lista vazia para armazenar os dados e registra a instância em um dicionário de *loggers*, onde o nome é a chave e o objeto `DataLogger` é o valor associado.

A função `log_data()` é usada para registrar dados. Ela aceita um número variável de argumentos nomeados e os armazena em um dicionário que inclui o tempo decorrido desde o início do registro. Esses dados são então adicionados à lista de dados.

A função `get_last_data()` permite recuperar um número específico de entradas mais recentes da lista de dados. Isso é útil para análises em tempo real ou para revisar os dados mais recentes.

A função `save_to_csv()` permite salvar os dados coletados em um arquivo CSV. Ele gera um nome de arquivo com base no nome do *logger* e na data e hora atual, e escreve os dados no arquivo.

A função de classe `get_logger()` permite obter um *logger* existente com base em seu nome. Isso é útil quando se deseja acessar um *logger* específico de qualquer lugar do código.

Figura 25 – Dados de um CSV com os dados do sinal de controle.

```
time,signal
0.0,0.0
0.05011892318725586,0.0
0.06352353096008301,0.0
0.11721086502075195,0.0
0.14821076393127441,0.0
0.384235143661499,0.0
0.4172368049621582,58.78506452828343
0.45423173904418945,27.054000000091037
0.5018215179443359,27.852999999944586
0.5497362613677979,28.651999999798136
0.5943691730499268,29.45100000014645
0.6316852569580078,30.25
0.8036842346191406,33.15700000003562
0.848686933517456,33.95599999988917
0.8861238956451416,34.5
```

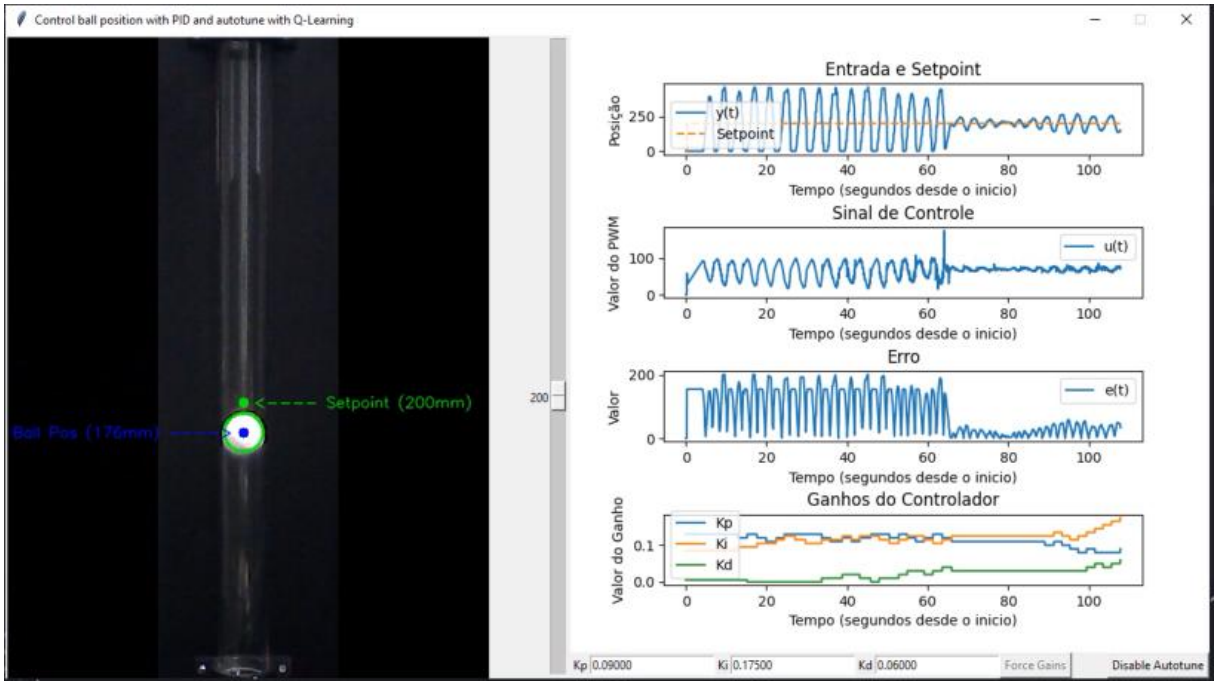
Fonte: Do autor, 2024.

3.8 DESENVOLVIMENTO DA INTERFACE GRÁFICA

De forma a tornar possível ao usuário alterar o *setpoint* em tempo real, projetou-se uma interface gráfica que permite não apenas alterar o *setpoint*, mas também visualizar a resposta da câmera (com o contorno da esfera), definir os ganhos do controlador PID manualmente, habilitar e desabilitar o sistema de *autotuning* e visualizar os gráficos de posição e *setpoint*, sinal de controle, erro e ganhos do controlador, todos em relação ao tempo. Isso é fundamental para permitir que haja uma análise clara e em tempo real durante o processo de *autotuning*, evidenciando se está havendo convergência ou divergência no aprendizado com Q-Learning.

A interface, desenvolvida em Python, utiliza a biblioteca padrão `tkinter` para todos os objetos. Os gráficos, porém, são renderizados a partir do `DataLogger` usando a biblioteca `Matplotlib` (MATPLOTLIB, 2024). Tal biblioteca se integra à interface com `tkinter` por meio de uma extensão embarcada, de forma que é possível plotar e atualizar gráficos em tempo real em uma interface gráfica. A figura Figura 26 exibe o resultado final do desenvolvimento da interface.

Figura 26 – Interface gráfica durante um experimento.



Fonte: Do autor, 2024.

3.9 METODOLOGIA DE ANÁLISE DE DADOS

Com o objetivo de avaliar o desempenho do sistema implementado são utilizadas neste trabalho as seguintes métricas:

- Média do Erro de Posição;
- Desvio Padrão do Erro de Posição;
- Valores Máximos e Mínimos do Erro;
- Erro Quadrático Médio (MSE);

Tais métricas são fundamentais para verificação do desempenho de um sistema de controle, permitindo análise e comparação com outros sistemas. No contexto deste trabalho, tais métricas podem são utilizadas para comparar como o sistema se comporta quando o algoritmo de aprendizado por reforço está ativo e quando não está. As definições das métricas listadas anteriormente são apresentadas a seguir.

3.9.0.1 Média

Aplicada ao erro, é calculada como a média dos valores dos erros de posição ao longo do tempo.

$$\text{Média} = \frac{1}{N}\sum_{i=1}^{N} \text{x}(i)$$

3.9.0.2 Desvio Padrão

Aplicado ao erro, fornece uma medida da variação ou dispersão dos erros em torno da média. Um valor baixo indica que o erro se mantém consistentemente próximo da média.

$$\text{Desvio Padrão} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{x}(i) - \text{Média})^2}$$

3.9.0.3 Valores Máximos e Mínimos

Aplicado ao erro, indica os picos de erro máximos e mínimos, úteis para identificar os extremos de desempenho do sistema.

$$\text{Valor Máximo} = \max_{i=1}^{N} \text{x}(i)$$

$$\text{Valor Mínimo} = \min_{i=1}^{N} \text{x}(i)$$

3.9.0.4 Erro Quadrático Médio (MSE)

Fornece uma medida da média dos quadrados dos erros, sendo comum em sistemas de controle para avaliar a precisão.

$$\text{MSE do Erro} = \frac{1}{N}\sum_{i=1}^{N}(\text{Erro}(i))^2$$

4 COLETA E ANÁLISE DE RESULTADOS

A coleta dos resultados deste trabalho se dá pelo uso do próprio software desenvolvido. Após a execução de um experimento, todos os dados coletados pelo sistema são salvos e analisados.

Especificamente, neste capítulo são apresentados e discutidos os 4 experimentos realizados, a saber:

- **Ensaio 1 - Variância no tempo:** Composto por 5 realizações onde o mesmo *setpoint* é definido, bem como os mesmos ganhos do controlador PID. O objetivo é verificar se o sistema varia no tempo, se comportando de maneira diferente em cada realização mesmo possuindo o mesmo cenário de operação.
- **Ensaio 2 - Linearidade:** Composto por 3 realizações, cada uma com magnitudes de degrau diferentes aplicadas à entrada. Nesse ensaio, o objetivo é verificar se o sistema é linear por meio da análise da dinâmica da resposta do sistema (dada pela posição da esfera).
- **Ensaio 3 - Controlador PID:** Composto por 1 realização que parte dos melhores ganhos do controlador PID, encontrados empiricamente.
- **Ensaio 4 - Controlador com *Autotuning*:** Composto por 1 realização cuja os ganhos do controlador PID foram escolhidos arbitrariamente de maneira a gerar uma dinâmica levemente oscilatória. O objetivo é que, após habilitar o *autotune*, o sistema seja capaz de aprender durante uma fase e, após o aprendizado, corrigir de forma autônoma a dinâmica do sistema.

Ao final, compara-se o desempenho com o sistema do Ensaio 3, cuja os ganhos do controlador PID mantiveram-se fixos, com os do Ensaio 4, cuja os ganhos foram dinamicamente alterados pelo algoritmo de *autotuning*.

O processamento dos dados obtidos e a análise de cada experimento foi realizada usando as bibliotecas `Pandas` (PANDAS, 2024), para carregar os arquivos CSV, `Matplotlib` para gerar os gráficos, e `NumPy` (NUMPY, 2024) para outros cálculos.

É importante notar que a calibração do sistema que permite a conversão de pixels para milímetros foi feita de forma que a distância em pixels entre a base visível e o topo visível fosse proporcional a 480mm, distância medida utilizando uma trena. Apesar de a lente da câmera gerar distorções na imagem, tal conversão se mostra suficiente para controlar o sistema, ainda que adicione um pequeno erro de medição.

4.1 ENSAIO 1: VARIÂNCIA NO TEMPO

A variância no tempo é um aspecto crítico na análise e no projeto de sistemas de controle, pois indica que as propriedades do sistema podem se alterar ao longo do tempo. A relevância de identificar e entender a variância temporal em um sistema de controle

reside na capacidade de desenvolver modelos e estratégias de controle mais robustos e adaptativos. Em sistemas com alta variância no tempo, as respostas a entradas idênticas podem variar significativamente em momentos distintos, exigindo uma abordagem de controle que possa se adaptar e ajustar continuamente à dinâmica mutável do sistema.

Para verificar a existência (ou não) da característica de variância no tempo no sistema considerado neste trabalho, executou-se 5 realizações do ensaio, onde são analisados os sinais da saída do sistema em cada realização.

Em cada experimento, os ganhos do controlador PID foram os mesmos, dados por:

Tabela 1 – Ganhos para as sessões de ensaio de variância no tempo.

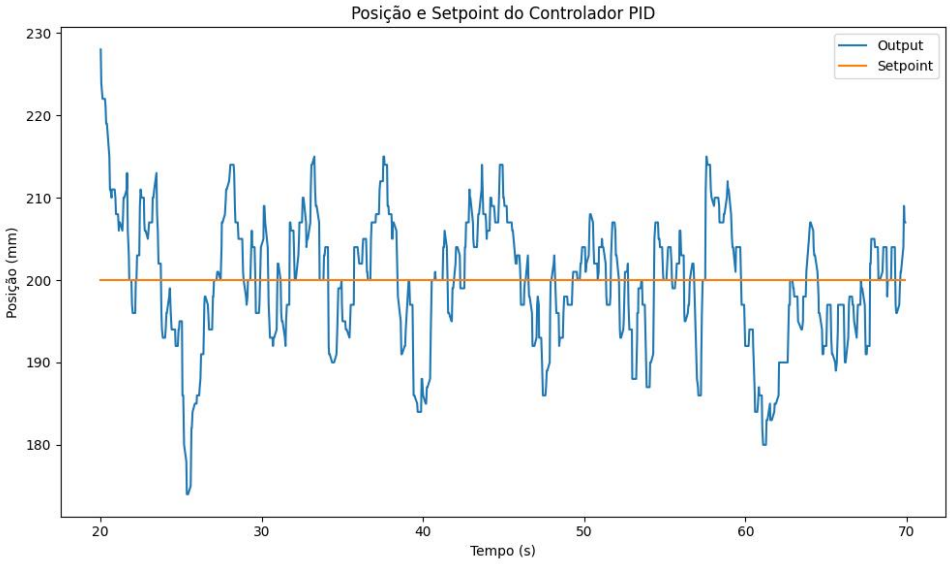
Kp	**Ki**	**Kd**
0.065	0.025	0.050

Fonte: Do autor, 2024.

Tais ganhos foram encontrados de maneira empírica, sem utilização de mecanismos de *autotune* ou cálculos do controlador, de forma a ter o menor erro médio possível. Estes foram encontrados mantendo um *setpoint* de 300mm, mas apresentam comportamento estável ao longo de múltiplas alturas.

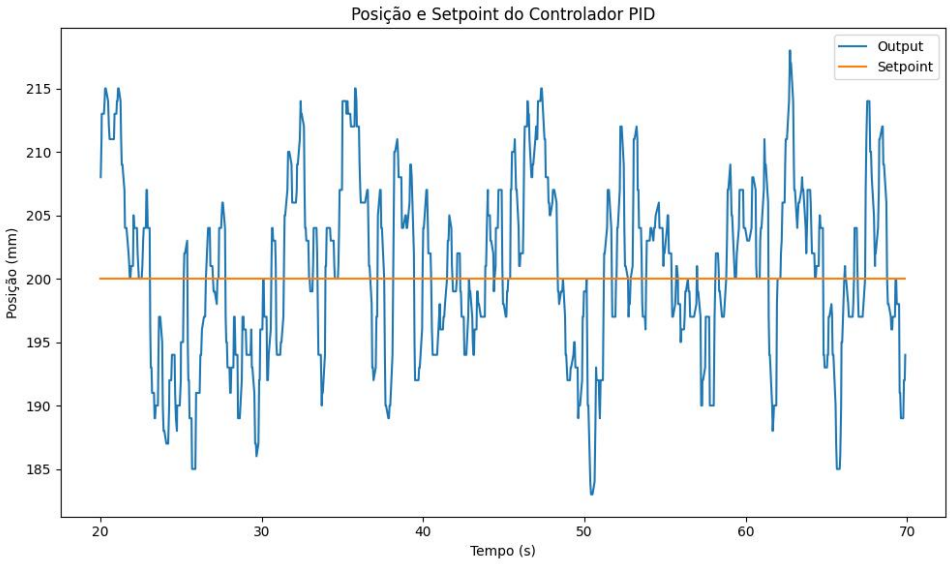
Todas as realizações tiveram o degrau aplicado no exato mesmo instante e partiram do valor de saída zero, com o sistema completamente estático. O degrau possuía magnitude de 200mm. Os dados analisados iniciam aos 20 segundos de experimento e terminam aos 70. Os gráficos a seguir demonstram o comportamento em cada uma das realizações.

Figura 27 – Gráfico da Realização 1 do Ensaio de Variância no Tempo.



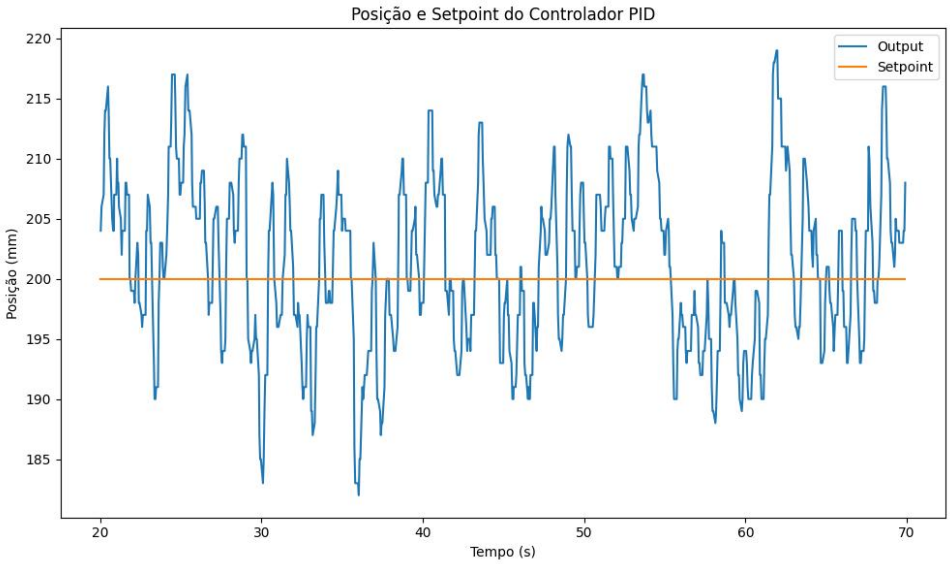
Fonte: Do autor, 2024.

Figura 28 – Gráfico da Realização 2 do Ensaio de Variância no Tempo.



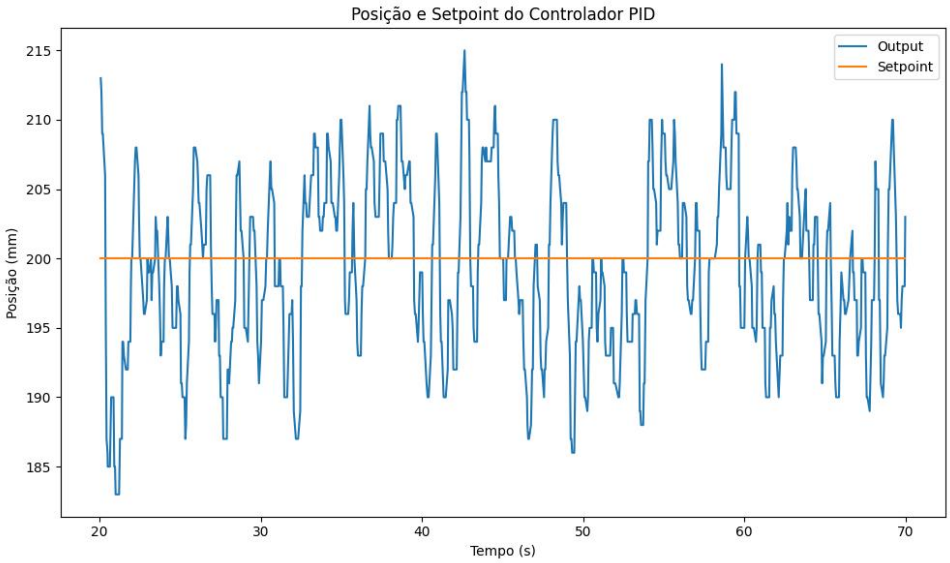
Fonte: Do autor, 2024.

Figura 29 – Gráfico da Realização 3 do Ensaio de Variância no Tempo.



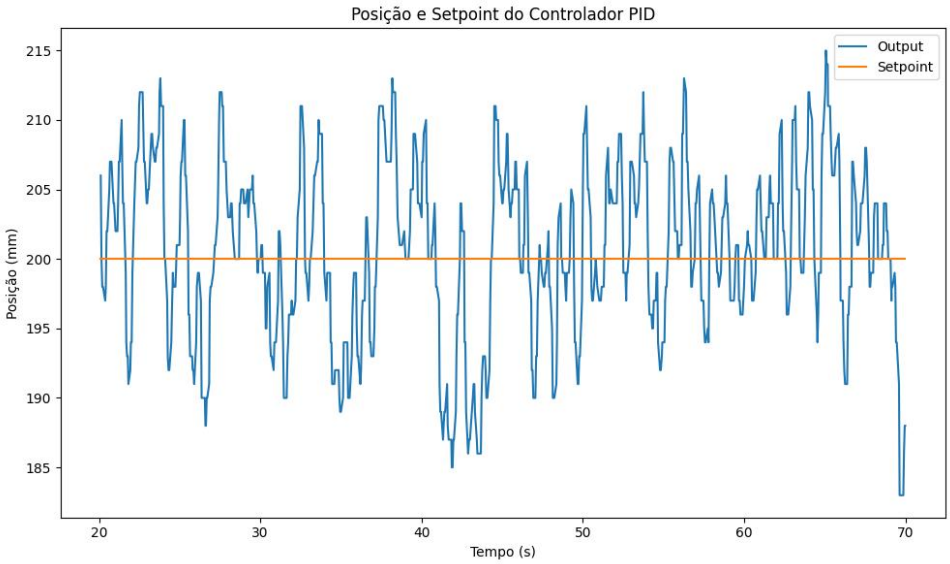
Fonte: Do autor, 2024.

Figura 30 – Gráfico da Realização 4 do Ensaio de Variância no Tempo.



Fonte: Do autor, 2024.

Figura 31 – Gráfico da Realização 5 do Ensaio de Variância no Tempo.



Fonte: Do autor, 2024.

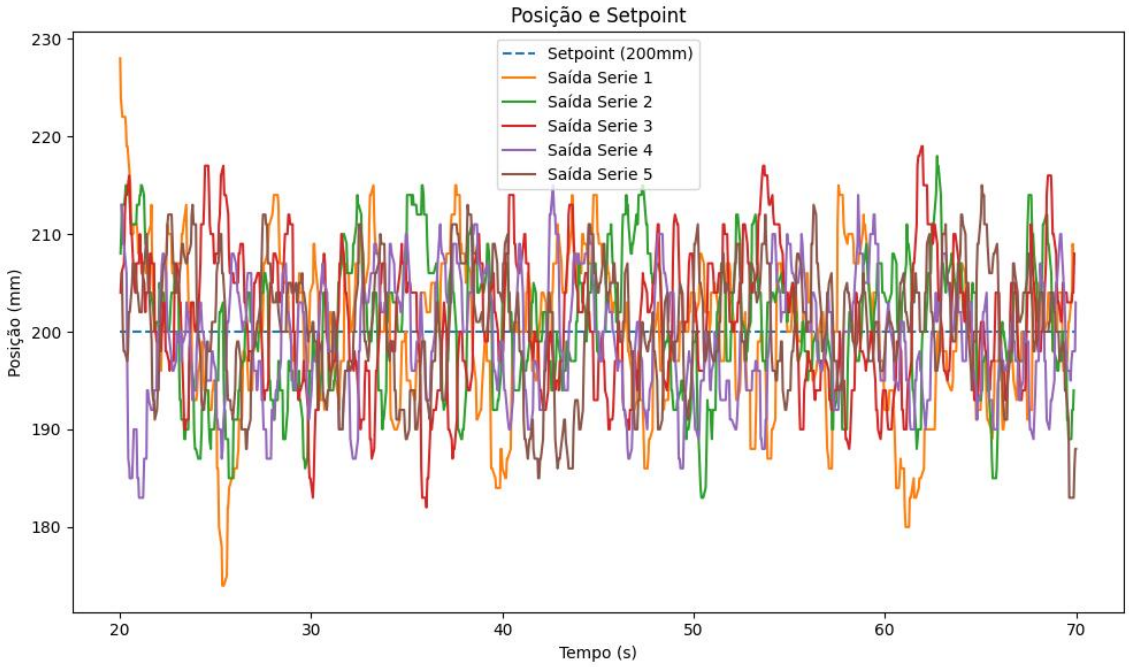
Por meio da análise dos gráficos das realizações 1 a 5, é evidente que o sistema não se comporta da mesma maneira em cada realização. Contudo, as respostas no tempo ainda são muito próximas do *setpoint* definido, conforme análise feita numericamente na Tabela 2. É também possível analisar o gráfico da Figura 32, que sobrepõe os dados e revela visualmente a diferença dos sinais. Por essa razão, tais diferenças podem ter origem em uma componente estocástica, como a rotação da esfera ao redor do próprio eixo em decorrência do regime turbulento do ar que flui dentro do duto. Além disso, há de se considerar um erro de medição, tanto causado pela distorção da imagem, quanto pelo ruído gerado pelo sensor da câmera. Dado estas constatações, o sistema pode ser considerado invariante no tempo.

Tabela 2 – Resultado do experimento de variância no tempo.

	Métricas			
	Pos. Média	**Desvio Padrão**	**Pos. Máxima**	**Pos. Mínima**
Realização 1	199.81	8.08	228	174
Realização 2	200.61	7.07	218	183
Realização 3	201.20	7.10	219	182
Realização 4	199.13	6.35	215	183
Realização 5	200.51	6.36	215	183

Fonte: Do autor, 2024.

Figura 32 – Gráfico sobreposto do Ensaio de Variância no Tempo.



Fonte: Do autor, 2024.

4.2 ENSAIO 2: LINEARIDADE

Conhecer a dinâmica do sistema é fundamental para o projeto adequado de um controlador. A não-linearidade pode introduzir complicações e impactar o projeto, a análise e o desempenho do sistema de controle. Modelos lineares não podem representar com precisão sistemas não-lineares, o que pode levar a previsões errôneas e projetos inadequados de controladores. Em muitos casos, sistemas não-lineares podem ser mais difíceis de prever e controlar devido à sua sensibilidade a condições iniciais e à presença de múltiplos pontos de equilíbrio ou soluções instáveis.

Para verificar se o sistema tratado neste trabalho possui comportamento não-linear, conduziu-se um experimento com a finalidade de analisar a resposta temporal do sistema ao introduzir diferentes degraus na sua entrada.

Em cada experimento, os ganhos do controlador PID foram os mesmos, dados por:

Tabela 3 – Ganhos para as sessões de ensaio de linearidade.

Kp	**Ki**	**Kd**
0.065	0.025	0.050

Fonte: Do autor, 2024.

O experimento foi conduzido de forma a introduzir na entrada do sistema degraus de 200, 300 e 400mm, respectivamente. Todos foram aplicados no mesmo instante e partiram do sistema em repouso. Para encontrar o tempo de assentamento (t_s), considerou-se $\pm 5\%$ de erro aceitável. Os resultados do experimento podem ser vistos no gráfico da Figura 33, cuja análise permite extrair os dados apresentados na Tabela 4.

Tabela 4 – Tempos de assentamento do ensaio de linearidade

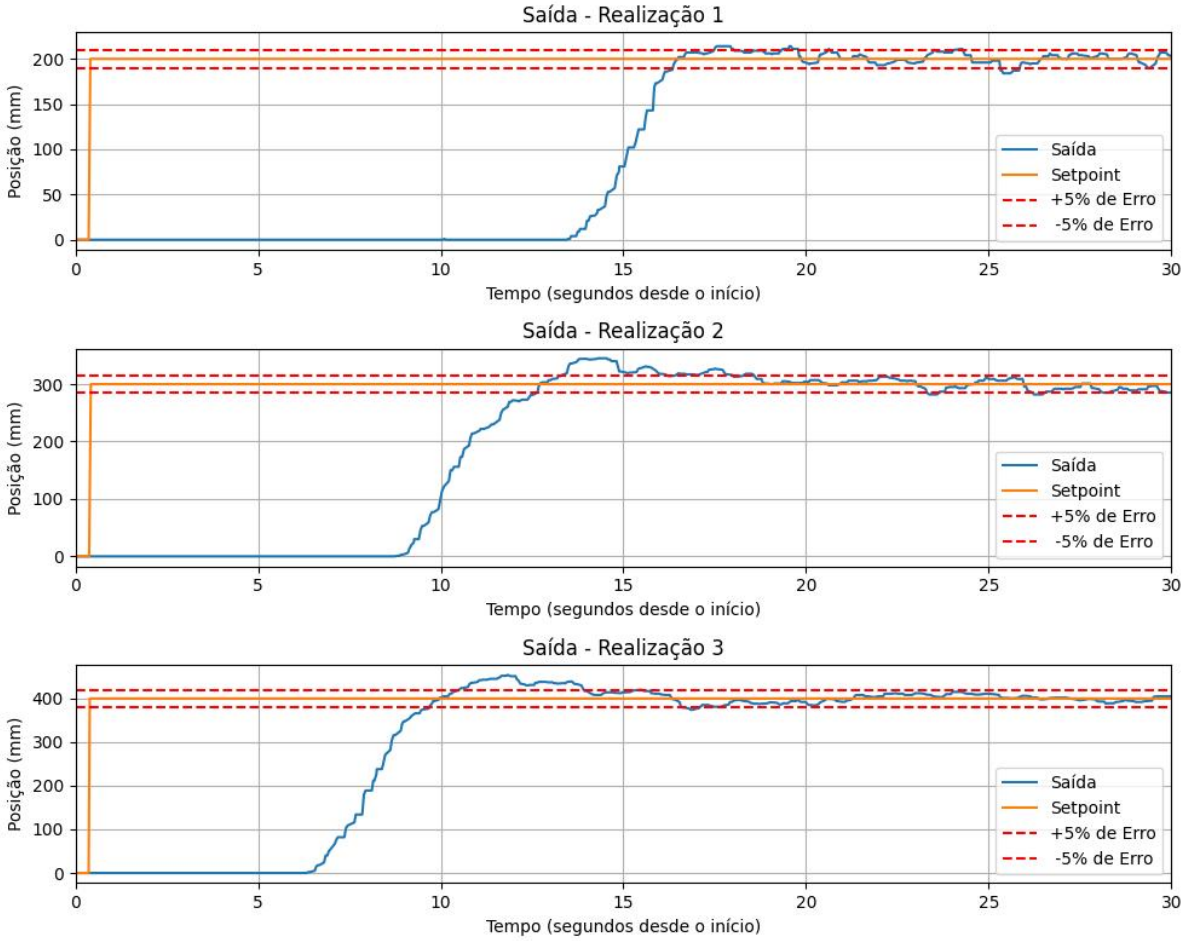
Realização	**Setpoint**	t_s
1	200mm	19.79s
2	300mm	18.56s
3	400mm	15.78s

Fonte: Do autor, 2024.

Conforme análise da Tabela 4, nota-se que o sistema apresenta comportamento não-linear, tendo em vista que o tempo de acomodação da saída varia conforme o nível do degrau apresentado na entrada.

Além disso, é importante pontuar que o sistema apresenta indícios de não-linearidade decorrentes também da presença de saturações no atuador (o controle do *duty-cycle* só pode ser feito entre 0 e 255) e pela presença de zonas mortas (uma vez que para valores de *duty-cycle* inferiores a 72 (28%), o sistema não reage). A queda da esfera acontece totalmente por gravidade, possuindo o sistema uma quantidade considerável de arrasto após haver direcionamento do fluxo de ar para dentro do tubo.

Figura 33 – Gráficos do ensaio de linearidade.



Fonte: Do autor, 2024.

Por essa razão, analisando os resultados apresentados neste experimento juntamente com aqueles discutidos na Seção 4.1 pode-se afirmar que o sistema é não-linear e invariante no tempo.

4.3 ENSAIO 3: CONTROLADOR PID

Este ensaio tem por objetivo analisar como o sistema se comporta quando controlado pelo controlador PID ajustado com os melhores ganhos obtidos via tentativa e erro para o cenário considerado, sem *autotuning*. Estes dados serão comparados na próxima seção com os resultados do Ensaio 4.

Especificamente neste ensaio, o *setpoint* foi definido como 350mm e foram utilizados os seguintes ganhos do controlador PID:

Tabela 5 – Ganhos do controlador PID para o ensaio 3.

Kp	**Ki**	**Kd**
0.065	0.025	0.050

Fonte: Do autor, 2024.

Conforme o gráfico da Figura 34, a reposta do sistema se manteve estável durante todo o teste, com leve sobressinal (cerca de 15%).

Na Tabela 6 são apresentadas algumas métricas de desempenho calculadas para o sinal de erro no intervalo de 27.7 a 181.4s (que corresponde a uma região onde a resposta do sistema está estabilizada).

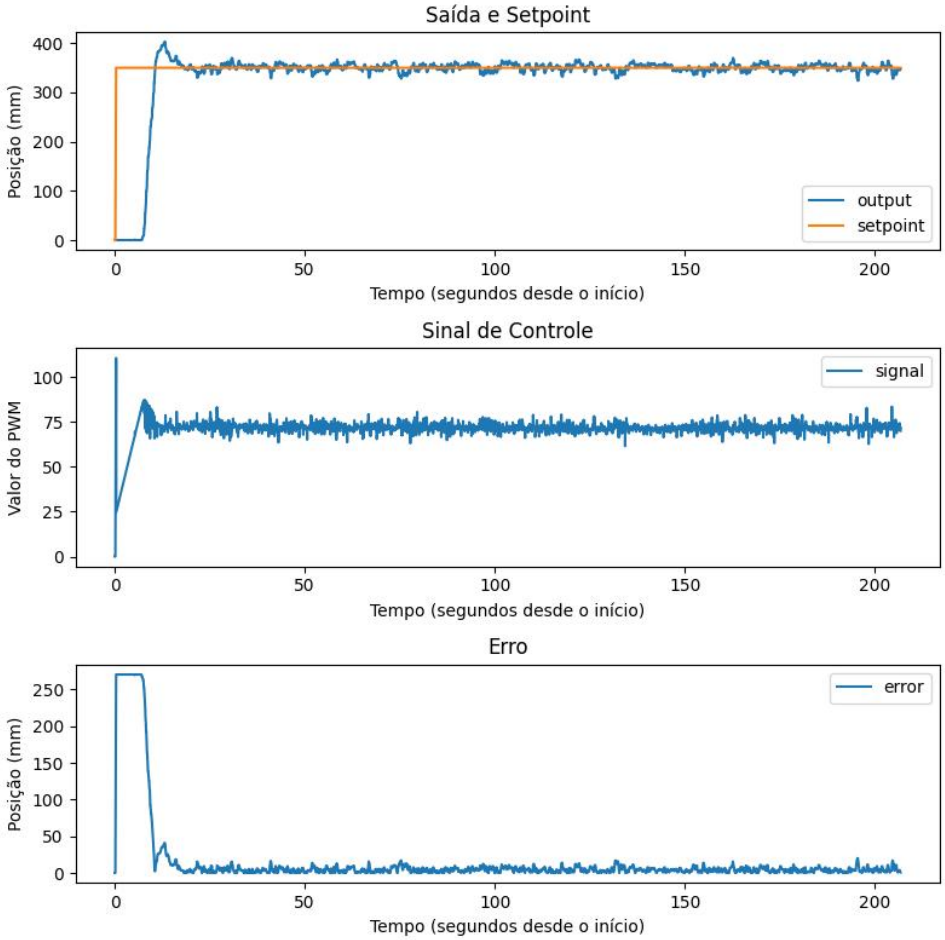
Tabela 6 – Comparação das métricas aplicadas ao erro

Métrica	**Valor no intervalo**
Média	4.42
Desvio Padrão	3.35
Máximo	17.00
MSE	30.83

Fonte: Do autor, 2024.

Ao analisar os dados da Tabela 6, é seguro afirmar que o controlador PID funciona de acordo com o esperado, sendo capaz de manter o erro com uma média de 4.42mm, inferior a 1,3% do *setpoint*. Além disso, possui um baixo desvio padrão, no valor de 3.35mm, indicando que a dispersão em torno da média é baixa e, por tanto, o erro se mantém próximo da média.

Figura 34 – Gráficos de saída, sinal de controle e erro do ensaio 3.



Fonte: Do autor, 2024.

4.4 ENSAIO 4: CONTROLADOR COM *AUTOTUNING*

Como já citado anteriormente na introdução deste capítulo, métodos de *autotuning* tradicionais geralmente requerem conhecimento prévio da planta e podem não ser eficazes em sistemas não-lineares, condição que se aplica a este sistema. Nesta situação, aplicar o aprendizado por reforço se faz de grande valia, uma vez que o objetivo é aprender a melhor política de ação para maximizar uma recompensa ao longo do tempo, definida em termos de minimização do erro. Essa abordagem é particularmente importante para manter a posição o mais próximo possível do ponto definido (*setpoint*), sendo crucial para atingir o objetivo do sistema.

Para verificar se o mecanismo de *autotuning* desenvolvido de fato funciona, escolheu-se para o controlador PID ganhos iniciais que gerariam uma resposta oscilatória. Tais ganhos são visíveis na Tabela 7.

Tabela 7 – Ganhos utilizados para gerar o comportamento oscilatório no ensaio 4.

Kp	**Ki**	**Kd**
0.020	0.040	0.050

Fonte: Do autor, 2024.

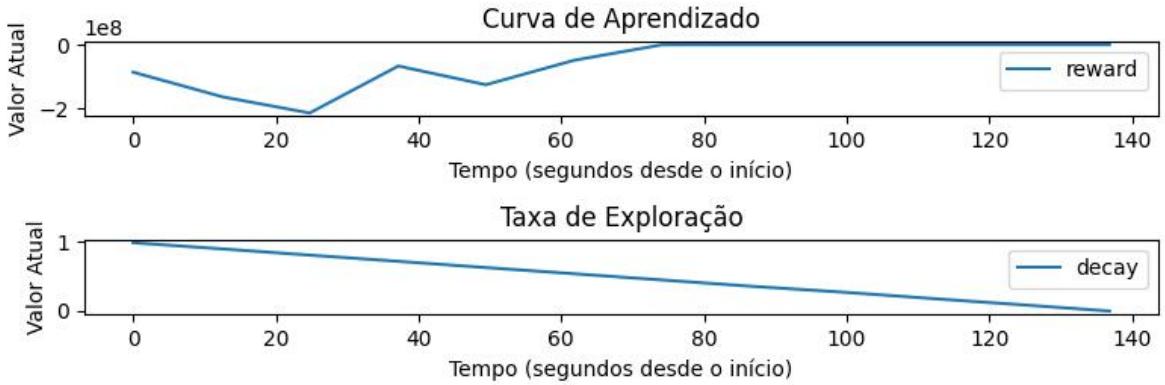
O objetivo do *autotuning* é manter o sistema por aproximadamente 15 segundos com erro médio inferior a 5mm, atingindo 180 medições. Todas as sessões foram conduzidas mantendo um *setpoint* de 200mm, isto é, o objetivo era manter o erro médio abaixo de 5% do valor do *setpoint*. Dito isso, após ser iniciado (indicado na primeira alteração no gráfico de ganhos do PID), o algoritmo deve encontrar os ganhos que atinjam tal objetivo e desligar automaticamente tão logo atingir.

O processo de uso do *autotuning* é divido em duas etapas: treinamento, onde o agente fará variações de forma autônoma a fim de aprender como suas ações influenciam no ambiente; e teste, onde o sistema recebe novamente os ganhos iniciais do controlador PID e o algoritmo deve extrapolar até convergir para novos ganhos cuja resposta do sistema atinja os objetivos especificados.

A etapa de treinamento é acompanhada pelo gráfico de decaimento do fator de exploração (disponível na Figura 35), que utiliza o algoritmo Epsilon-Greedy para balancear a exploração e extrapolação. Essa etapa levou cerca de 3 minutos desde a inicialização do *autotune* para ser finalizada. Após sua finalização, o algoritmo Q-Learning passa a fazer apenas extrapolação, utilizando apenas os valores da tabela Q para definir as ações mediante o estado atual do sistema.

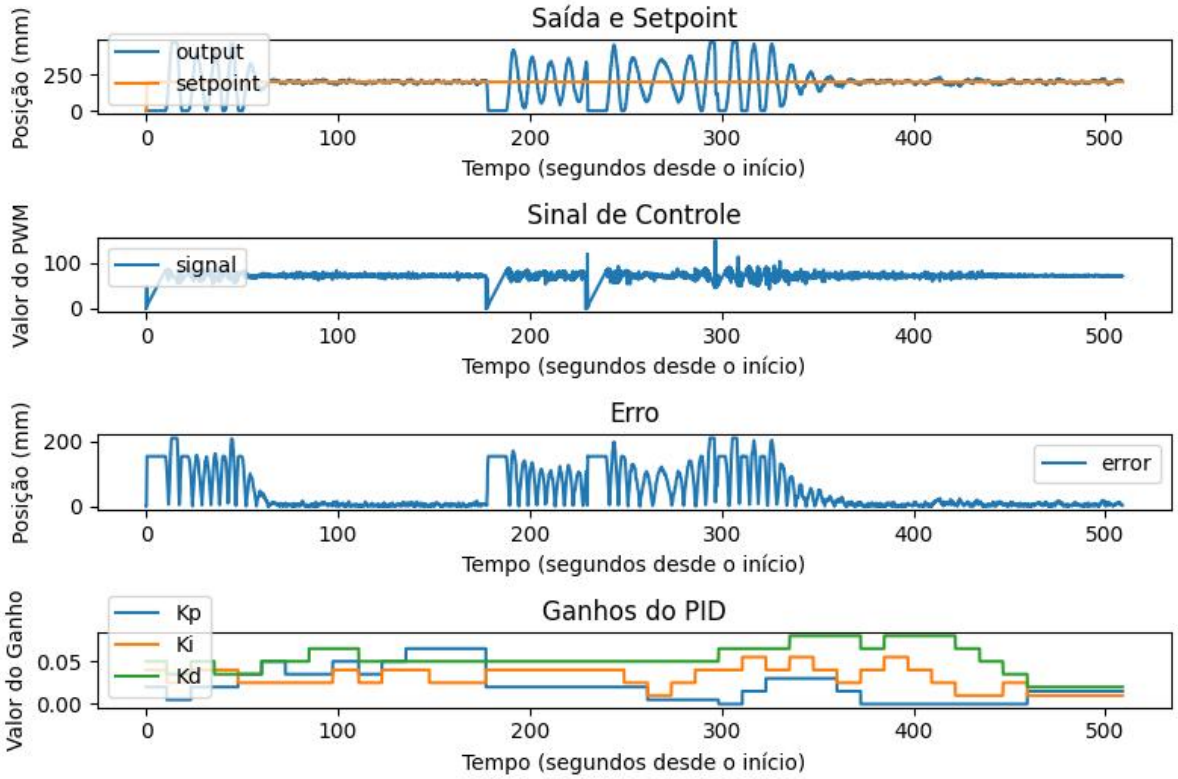
Em uma primeira análise, foi verificado, através do uso do gráfico do decaimento da taxa de exploração e do gráfico da taxa de aprendizado, vistos na Figura 35, que houve convergência, isto é, o algoritmo aprendeu, convergindo para minimização do erro. Contudo, isso não significa necessariamente que o algoritmo não está enviesado. Para garantir que não esteja, a etapa de teste é fundamental.

Figura 35 – Gráficos da taxa de exploração e taxa de aprendizado do ensaio 4.



Fonte: Do autor, 2024.

Figura 36 – Gráficos de saída, sinal de controle, erro e ganhos do PID do ensaio 4.



Fonte: Do autor, 2024.

Analisando o gráfico da Figura 36, no intervalo inicial, especialmente entre os 9 e 62 segundos, é notável a presença do sinal oscilatório na saída causada pelo controle ineficiente devido aos ganhos incorretos. Neste intervalo, o algoritmo Q-Learning já começa a preencher a tabela Q, variando entre ações de exploração e extrapolação. Para cada ação escolhida, há uma variação dos ganhos do controlador PID.

Para detectar se de fato houve convergência e, portanto, se o algoritmo aprendeu, foi inserido, após a etapa de aprendizado, especificamente no instante 178s, os mesmos ganhos oscilatórios iniciais.

Como resultado, o sistema passa a ter comportamento oscilatório novamente. O *autotune* só foi ativado no instante 248s. A partir de então, ele começa a variar os ganhos como forma de encontrar os ganhos ideias. Então, no instante 458s, cerca de 3 minutos e meio depois, há a convergência para um valor considerável aceitável dado o objetivo do *autotune*.

Os ganhos encontrados podem ser vistos a seguir:

Tabela 8 – Ganhos encontrados pelo *autotune* no ensaio 4.

Kp	**Ki**	**Kd**
0.017	0.011	0.022

Fonte: Do autor, 2024.

Durante o instante 460s a 508s, o sistema operou com estes ganhos, de forma que foi possível analisar numericamente o comportamento do erro, como segue:

Tabela 9 – Comparação das métricas aplicadas ao erro

Métrica	**Valor no intervalo**
Média	6.07
Desvio Padrão	3.89
Máximo	18.55
MSE	52.06

Fonte: Do autor, 2024.

Quando comparado a Tabela 6 do Ensaio 3, os valores encontrados apresentam pouca discrepância, de forma que há uma diferença de apenas 1.65mm na média e 0,54mm no desvio padrão. Todavia, é possível afirmar que os ganhos encontrados pelo *autotune* são menos eficientes que os ganhos encontrados e otimizados empiricamente. É válido mencionar, no entanto, que o sistema de *autotune* foi capaz de corrigir um comportamento oscilatório, aprender a forma correta de regular os ganhos e corrigir de forma autônoma em um intervalo de 8 minutos e 27 segundos, enquanto a regulagem manual exigiu horas de testes.

5 CONCLUSÃO

Neste trabalho, foi apresentado a implementação de um sistema de controle da posição de uma esfera utilizando visão computacional e aprendizado por reforço, o qual foi utilizado como método de *autotune* para ajustar os ganhos de um controlador *PID*. O intuito principal era que, mesmo sem o conhecimento do modelo matemático da planta, fosse possível realizar o controle adaptativo em tempo real, sem pré-treinamento, com a finalidade de reduzir o erro. Tal objetivo não é trivial em situações práticas, como em aplicações industriais, mas no contexto específico da planta montada foi atingido com êxito.

Com base nos resultados apresentados no capítulo anterior, pode-se concluir que o algoritmo de *autotune* foi eficaz em atingir os objetivos propostos, tendo demonstrado ser capaz de ajustar de forma autônoma os parâmetros do controlador PID, resultando em uma redução significativa do erro e melhorando o desempenho do sistema controlado. As experiências realizadas demonstraram também que a hipótese inicial de que o sistema seria não-linear e invariante no tempo foi verificada.

Apesar de o algoritmo de *autotune* com Q-Learning não ter sido capaz de entregar a mesma eficiência que um ajuste manual dos ganhos, a economia de tempo para o operador é significativa, uma vez que o método não requer interação e foi capaz de encontrar ganhos que mantiveram o sistema estável e com baixo erro médio, estando dentro dos objetivos estipulados inicialmente.

Ainda que realizar o procedimento de ajuste dos ganhos do controlador utilizando métodos tradicionais seja mais comum, este trabalho demonstra que é possível atingir objetivos complexos utilizando soluções modernas. Nesse âmbito, sugere-se para trabalhos futuros a aplicação deste mesmo modelo de aprendizado em plantas diversas, a fim de analisar sua eficácia em uma gama maior de cenários. Como adicional, os futuros autores podem comparar o desempenho do algoritmo proposto com outras técnicas de *autotuning* existentes, a fim de avaliar suas vantagens e limitações relativas.

REFERÊNCIAS

ALMEIDA, Gabriel C. S.; SALLES, Rafael S.; CARVALHO, Adelson S. Implementação de Controladores Híbridos PID-Fuzzy: Análise de Desempenho em uma Planta de Nível. *In*: XXIII Congresso Brasileiro de Automática. [*S.l.*: *s.n.*], nov. 2020.

ALPAYDIN, E. **Introduction to Machine Learning**. 4ª edição. Cambridge, MA: The MIT Press, 2020. ISBN 978-0262043793.

ÅSTRÖM, K. J. **Control System Design**. 1ª edição. Santa Barbara, CA, USA: Department of Mechanical & Environmental Engineering, University of California, 2002.

ÅSTRÖM, K. J.; HÄGGLUND, T. **PID Controllers: Theory, Design, and Tuning**. 2ª edição. Raleigh, NC, USA: Instrument Society of America, 1995. ISBN 1-55617-516-7.

ÅSTRÖM, K.J.; HÄGGLUND, T. **Advanced PID Control**. [*S.l.*]: ISA - The Instrumentation, Systems, e Automation Society, 2006. ISBN 9781556179426.

BANZI, M.; SHILOH, M. **Getting Started with Arduino: The Open Source Electronics Prototyping Platform**. 3ª edição. Sebastopol, CA, USA: Maker Media, Inc, 2014. ISBN 978-1449363338.

BELANGER, P. R. **Control Engineering: A Modern Approach**. 1ª edição. Oxford: Oxford University Press, 1995. ISBN 978-0030134890.

BEQUETTE, B. W. **Process Control: Modeling, Design, and Simulation**. [*S.l.*]: Prentice Hall, 2003. (Prentice-Hall international series in the physical and chemical engineering sciences). ISBN 9780133536409.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. New York: Springer, 2007. ISBN 978-0387310732.

CANON. **120MXS: Ultra-high 120MP CMOS sensor**. [*S.l.*]: Canon CMOS Sensors. Disponível em: https://canon-cmos-sensors.com/cmos-sensors/. Acesso em: 5 de janeiro de 2024.

CHRIS LIECHTI. **pySerial**. [*S.l.*: *s.n.*], 2020. Disponível em: https://pyserial.readthedocs.io/en/latest/pyserial.html. Acesso em: 17 de dezembro de 2023.

D&D COMPONENTES. **Arduino Uno R3**. [*S.l.*]: D&D Componentes, 2023. Disponível em: https://www.dedcomponentes.com.br/produto/arduino-uno-r3-original-da-italia. Acesso em: 5 de janeiro de 2024.

DOGRU, Oguzhan; VELSWAMY, Kirubakaran; IBRAHIM, Fadi; WU, Yuqi; SUNDARAMOORTHY, Arun S.; HUANG, Biao; XU, Shu; NIXON, Mark; BELL, Noel. Reinforcement learning approach to autonomous PID tuning. **Computers & Chemical Engineering**, v. 161, p. 107760, 2022. ISSN 0098-1354.

DORF, R. C.; BISHOP, R. H. **Modern Control Systems**. 14ª edição. Harlow, Reino Unido: Pearson, 2021. ISBN 978-0137307098.

ERDOĞAN, Kemal; YILMAZ, Nihat. Shifting Colors to Overcome not Realizing Objects Problem due to Color Vision Deficiency. Kuala Lumpur, Malaysia, dez. 2014.

FOLEY, J. D.; DAM, A. van; HUGHES, John F. **Computer Graphics: Principles and Practice**. 2ª edição. Boston: Addison-Wesley, 1990. ISBN 978-0201121100.

FORSYTH, D. A.; PONCE, J. **Computer Vision: A Modern Approach**. 2ª edição. Essex, Inglaterra: Pearson, 2011. ISBN 978-0136085928.

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. **Feedback Control of Dynamic Systems**. 8ª edição. Harlow, UK: Pearson, 2021. ISBN 978-0137516834.

GAVIN WRIGHT. **Closed-loop control system**. [*S.l.*]: TechTarget, mai. 2022. Disponível em: https://www.techtarget.com/whatis/definition/closed-loop-control-system. Acesso em: 5 de janeiro de 2024.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 4ª edição. New York: Pearson, 2018. ISBN 978-0133356724.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. 1ª edição. Cambridge, MA: MIT Press, 2016. ISBN 978-0262337373.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning**. 2ª edição. New York: Springer, 2016. ISBN 978-0387848570.

HAYKIN, S.; MOHER, M. **Communication Systems**. 5ª edição. New York: John Wiley & Sons, 2009. ISBN 978-0471697909.

HOLMES, D. G.; LIPO, T. A. **Pulse Width Modulation for Power Converters: Principles and Practice**. 1ª edição. [s.l.]: Wiley-IEEE Press, 2003. ISBN 978-0471208143.

HOLST, G. C.; LOMHEIM, T. S. **CMOS/CCD Sensors and Camera Systems**. 2ª edição. Florida, USA: SPIE Press, 2011. ISBN 978-1510627116.

JAIN, A. K.; FLYNN, P. J.; ROSS, A. A. (Ed.). **Handbook of Biometrics**. 1ª edição. New York: Springer, 2008. ISBN 978-0387710402.

JANESICK, J. R. **Scientific Charge-Coupled Devices**. 1ª edição. Bellingham, WA: SPIE Press, 2001. ISBN 978-0819436986.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. **Science**, v. 349, n. 6245, p. 255–260, 2015. eprint: https://www.science.org/doi/pdf/10.1126/science.aaa8415.

KRAR, S. **CNC Simplified**. 1ª edição. New York: Industrial Press, Inc., 2000. ISBN 978-0831131333.

LANGFORD, M.; FOX, A.; SMITH, R. S. **Langford's Basic Photography: The Guide for Serious Photographers**. 10ª edição. New York: Focal Press, 2015. ISBN 978-1136096693.

LEWELL, J. **Digital Photography for Next to Nothing: Free and Low-Cost Hardware and Software to Help You Shoot Like a Pro**. 1ª edição. Chichester, UK: John Wiley & Sons, 2010. ISBN 978-0470970584.

MARGOLIS, M. **Arduino Cookbook: Recipes to Begin, Expand, and Enhance Your Projects**. 2ª edição. Sebastopol, CA, USA: O'Reilly Media, Inc., 2011. ISBN 978-1449313876.

MARTIN LUNDBERG. **Simple PID**. [*S.l.*: *s.n.*], 2023. Disponível em: https://simple-pid.readthedocs.io/en/latest/. Acesso em: 17 de dezembro de 2023.

MATPLOTLIB. **Matplotlib**. [*S.l.*: *s.n.*], 2024. Disponível em: https://matplotlib.org/. Acesso em: 17 de janeiro de 2024.

MATTSON, M. **CNC Programming: Principles and Applications**. 1ª edição. Clifton Park, NY, USA: Delmar Cengage Learning, 2012. ISBN 978-1418060992.

MITCHELL, T. M. **Machine Learning**. 1ª edição. New York: McGraw-Hill, 1997. ISBN 978-0070428072.

MNIH, Volodymyr *et al.* Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015.

MOHAMED, Mokhtar; ELMAHALAWY, Ahmed; HARB, Hany. Developing the pulse width modulation tool (PWMT) for two timer mechanism technique in microcontrollers. *In*: 148–153.

MONK, S. **Programming Arduino: Getting Started with Sketches**. 2ª edição. New York: McGraw-Hill, 2014. ISBN 978-0071784221.

MURPHY, K. P. **Machine learning: a probabilistic perspective**. 1ª edição. Cambridge, MA: MIT, 2012. ISBN 978-0262018029.

NASA. **Delta-Doped Charged Coupled Devices (CCD) for Ultra-Violet and Visible Detection**. [*S.l.*]: NASA, 2006. Disponível em: https://pt.wikipedia.org/wiki/Ficheiro:Delta-Doped_Charged_Coupled_Devices_(CCD)_for_Ultra-Violet_and_Visible_Detection.jpg. Acesso em: 5 de janeiro de 2024.

NUMPY. **NumPy**. [*S.l.*: *s.n.*], 2024. Disponível em: https://numpy.org/. Acesso em: 17 de janeiro de 2024.

OGATA, K. **Modern Control Engineering**. 5ª edição. Upper Saddle River, NJ, USA: Prentice Hall, 2010. ISBN 978-0136156734.

OPENCV. **Open Source Computer Vision Library**. [*S.l.*: *s.n.*], 2024. Disponível em: https://docs.opencv.org/. Acesso em: 2 de janeiro de 2024.

PANDAS. **Pandas**. [*S.l.*: *s.n.*], 2024. Disponível em: https://pandas.pydata.org/. Acesso em: 17 de janeiro de 2024.

PARKER, J. R. **Algorithms for Image Processing and Computer Vision**. 2ª edição. Indianapolis, IN, USA: Wiley Publishing, 2010. ISBN 978-0470643853.

REDA, Khairi; MATEEVITSI, Victor; OFFORD, Catherine. A human-computer collaborative workflow for the acquisition and analysis of terrestrial insect movement in behavioral field studies. **EURASIP Journal on Image and Video Processing**, v. 2013, p. 48, dez. 2013.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3ª edição. Upper Saddle River, NJ, USA: Pearson, 2016. ISBN 978-0136042594.

SCHERZ, P.; MONK, S. **Practical Electronics for Inventors**. 3ª edição. New York: McGraw-Hill, 2013. ISBN 978-0071771344.

SERRA, J. **Image Analysis and Mathematical Morphology**. 1ª edição. Orlando, FL, USA: Academic Press, 1982. ISBN 978-0126372403.

SILVER, David; LEVER, Guy; HEESS, Nicolas; DEGRIS, Thomas; WIERSTRA, Daan; RIEDMILLER, Martin. Deterministic Policy Gradient Algorithms. *In*: XING, Eric P.; JEBARA, Tony (Ed.), 1. **Proceedings of the 31st International Conference on Machine Learning**. Bejing, China: PMLR, jun. 2014. v. 32. (Proceedings of Machine Learning Research, 1), p. 387–395.

SMITH, Alvy R. Color Gamut Transform Pairs. **SIGGRAPH Comput. Graph.**, Association for Computing Machinery, New York, NY, USA, v. 12, n. 3, p. 12–19, mai. 1978. ISSN 0097-8930.

SMITH, C. A. **Principles and Practice of Automatic Process Control**. 3ª edição. New York: John Wiley & Sons, 2005. ISBN 978-0471431909.

SONKA, M.; HLAVAC, V.; BOYLE, R. **Image Processing, Analysis, and Machine Vision**. 4ª edição. Toronto: Cengage Learning, 2014. ISBN 978-1305192157.

STALLINGS, W. **Data and Computer Communications**. 10ª edição. Essex, Inglaterra: Pearson, 2013. ISBN 978-0137561704.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. 2ª edição. Cambridge: MIT Press, 2018. ISBN 978-0262039246.

SUZUKI, Satoshi; ABE, Keiichi. Topological structural analysis of digitized binary images by border following. **Computer Vision, Graphics, and Image Processing**, v. 30, n. 1, p. 32–46, 1985. ISSN 0734-189X.

SZELISKI, P. R. **Computer Vision: Algorithms and Applications**. 1ª edição. London: Springer, 2010. ISBN 978-1848829350.

TECHTONICS. **DC Motor Speed Control Regulator**. [*S.l.*]: Techtonics, 2023. Disponível em: https://www.techtonics.in/dc-motor-speed-control-regulator-pulse-pwm-12v-40v-10a. Acesso em: 1 de janeiro de 2024.

TOLIYAT, H. A.; CAMPBELL, S. G. **DSP-Based Electromechanical Motion Control**. 1ª edição. New York: CRC Press, 2003. ISBN 978-0367394967.

VALVANO, J. W. **Embedded Systems: Introduction to ARM Cortex-M Microcontrollers**. 5ª edição. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2019. ISBN 978-1477508992.

VINCENT MAZET. **Histogram thresholding**. [*S.l.*]: Basics of Image Processing, 2023. Disponível em: https://vincmazet.github.io/bip/segmentation/histogram.html. Acesso em: 5 de janeiro de 2024.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. **Machine Learning**, v. 8, n. 3, p. 279–292, 1992.