



## DESENVOLVIMENTO DE UM ROBÔ SEGUIDOR DE LINHA UTILIZANDO PROCESSAMENTO DIGITAL DE IMAGENS

DOI: 10.37702/2175-957X.COBENGE.2025.6169

**Autores:** IGOR RODRIGUES CASSIMIRO, MARCO ANTONIO DE SOUZA LEITE CUADROS, MARCELO VICTOR FERREIRA BARBOSA, CAIO LOPES DE OLIVEIRA, NICOLAS NASÁRIO FERREIRA, ENZO MEDEIROS AGNEZ

**Resumo:** Este trabalho apresenta um robô seguidor de linha desenvolvido como atividade prática da disciplina de Processamento Digital de Imagens. O protótipo, baseado em ESP32 S3 e uma web-cam, utiliza Python com OpenCV para processamento em tempo real, aplicando Transformada de Hough e segmentação HSV. A metodologia incluiu pré-processamento de imagens (filtros e operações morfológicas), definição de ROI e cálculo de ângulos para navegação, com interface PyQt5 para ajuste de parâmetros. Os resultados comprovaram a eficácia dos métodos clássicos de PDI em robótica educacional, destacando a integração teoria-prática no ensino de engenharia. O projeto demonstrou a aplicabilidade de técnicas de visão computacional em sistemas embarcados acessíveis, reforçando a importância de abordagens interdisciplinares na formação engenheira.

**Palavras-chave:** Processamento Digital de Imagens, Visão Computacional, Robótica

# **DESENVOLVIMENTO DE UM ROBÔ SEGUIDOR DE LINHA UTILIZANDO PROCESSAMENTO DIGITAL DE IMAGENS**

## **1 INTRODUÇÃO**

A transformação digital nas indústrias, caracterizada pela Indústria 4.0, integra tecnologias como Inteligência Artificial, Robótica, Internet das Coisas (IoT) e análise avançada de dados, possibilitando automação inteligente e otimização em tempo real dos processos produtivos (ABDI, 2019). De acordo com a Agência Brasileira de Desenvolvimento Industrial, essa adoção tecnológica apresenta potencial para gerar uma economia anual de R\$ 73 bilhões no Brasil.

No entanto, persistem obstáculos significativos para a consolidação da Indústria 4.0 no país, incluindo limitações de infraestrutura tecnológica, restrições financeiras e, principalmente, desconhecimento sobre os benefícios dessas inovações para a competitividade industrial (Cordeiro, Reis e Fernandes, 2024). Esses desafios destacam a necessidade de reformular a formação de engenheiros, incorporando conteúdos atualizados e experiências práticas alinhadas às demandas do setor produtivo.

Particularmente na formação em engenharia, observa-se uma lacuna curricular em temas emergentes como Visão Computacional e IoT (CNI, 2016). O processamento digital de imagens emerge como tecnologia crítica nesse contexto, com aplicações versáteis no desenvolvimento de sistemas robóticos e automação industrial.

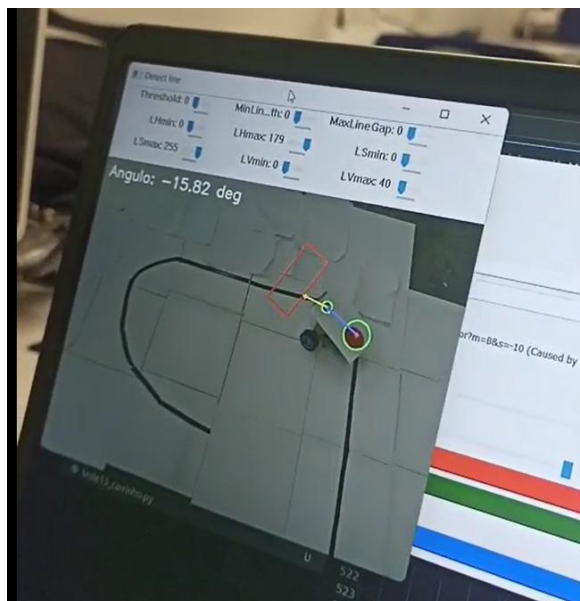
Neste trabalho, apresenta-se uma experiência prática desenvolvida no Programa de Pós-Graduação em Engenharia de Controle e Automação do IFES Serra, na disciplina de Processamento Digital de Imagens. Como mostra a Figura 1, o projeto envolveu o desenvolvimento colaborativo de um robô seguidor de linha, utilizando técnicas de visão computacional e programação em Python, com objetivo de aproximar os alunos dos conceitos fundamentais da Indústria 4.0 através da aprendizagem ativa.

A abordagem pedagógica adotou os princípios de Freire (1996), enfatizando o protagonismo discente. Os alunos foram responsáveis por: (1) planejar estratégias de detecção de direção, (2) definir regiões de interesse, (3) implementar diferentes abordagens de segmentação de imagens, e (4) testar parâmetros de controle, com o professor atuando como mediador do processo.

Os resultados demonstraram que o robô seguidor de linha (Figura 1) alcançou seu objetivo funcional, comprovando a eficácia da abordagem no desenvolvimento de competências técnicas e na integração multidisciplinar de conhecimentos em processamento de imagens, controle e automação. A atividade ainda estabeleceu conexões relevantes com outros componentes curriculares, incluindo matemática aplicada e sistemas embarcados.

Este artigo está organizado em quatro seções principais: a Seção 2 apresenta a fundamentação teórica; a Seção 3 detalha a metodologia e recursos tecnológicos utilizados; a Seção 4 discute os resultados obtidos; e a Seção 5 apresenta as conclusões e perspectivas futuras, que incluem a incorporação de técnicas baseadas em redes neurais convolucionais.

Figura 1: Projeto de seguidor de linha utilizando Processamento Digital de Imagens



Fonte: Autoria Própria (2025)

## 2 FUNDAMENTAÇÃO TEÓRICA

### 2.1 Processamento Digital de Imagens (PDI)

O Processamento Digital de Imagens (PDI) compreende técnicas para manipulação e análise de imagens digitais, representadas matematicamente como funções discretas bidimensionais  $f(x,y)$ , onde  $x$  e  $y$  são coordenadas espaciais e  $f(x,y)$  representa a intensidade do pixel (GONZALEZ; WOODS, 2010). No caso de imagens coloridas no modelo RGB, utilizam-se três matrizes correspondentes aos canais vermelho, verde e azul.

As principais etapas do PDI incluem:

- Pré-processamento (filtragem e realce)
- Segmentação (isolamento de regiões de interesse)
- Análise (extração de características)

### 2.2 Transformada de Hough

A Transformada de Hough é um método robusto para detecção de formas geométricas em imagens, que mapeia pontos do espaço imagem para um espaço de parâmetros acumuladores (FERRIANI; RIBEIRO, 2005). Para detecção de círculos, utiliza-se a equação paramétrica:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \quad (1)$$

onde:

- $(x_0, y_0)$  = coordenadas do centro
- $r$  = raio do círculo

## 2.3 Detecção de Círculos com OpenCV

A implementação `cv2.HoughCircles()` do OpenCV utiliza a Transformada de Hough por gradiente (DUDA; HART, 1972), com os seguintes parâmetros ajustáveis:

Tabela 1: Parâmetros do algoritmo de Processamento Digital de Imagens

| Parâmetro | Descrição                                               |
|-----------|---------------------------------------------------------|
| dp        | Razão inversa entre resolução do acumulador e da imagem |
| minDist   | Distância mínima entre centros detectados               |
| param1    | Limiar superior para detecção de bordas                 |
| param2    | Limiar de votos no acumulador                           |
| minRadius | Raio mínimo para detecção                               |
| maxRadius | Raio máximo para detecção                               |

Fonte: Autoria Própria (2025)

## 3 METODOLOGIA

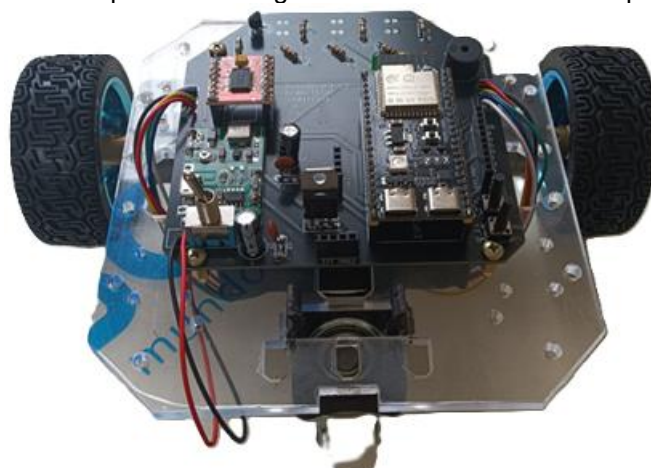
### 3.1 Desenvolvimento do Protótipo

O robô seguidor de linha foi desenvolvido no laboratório GAIIn conforme as seguintes etapas:

#### 1. Projeto e Fabricação:

- Projeto mecânico e elétrico
- Desenvolvimento do circuito impresso (PCB) personalizado
- Montagem com microcontrolador ESP32 S3 e motores

Figura 2: Protótipo do robô seguidor de linha utilizado no experimento



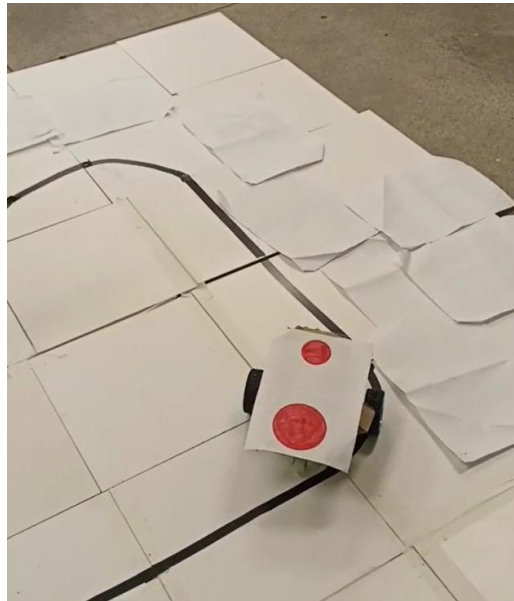
Fonte: Autoria Própria (2025)



## 2. Configuração do Ambiente:

- Pista de testes em MDF branco
- Dois círculos impressos como referência de navegação
- Linha preta contínua para seguimento

Figura 3: Pista utilizada para os testes do robô seguidor de linha



Fonte: Autoria Própria (2025)

## 3.2 Sistema de Visão Computacional

### Deteccção de Círculos:

- Utilizou-se a função `cv2.HoughCircles()` do OpenCV
- Parâmetros ajustados para detectar apenas os dois círculos de referência
- Construção de vetor de orientação entre os centros dos círculos

### Deteccção de Linha:

1. Pré-processamento:
  - Conversão para espaço de cores HSV
  - Segmentação por limiarização
  - Aplicação de operações morfológicas
2. Transformada de Hough probabilística (`cv2.HoughLinesP()`)
  - Região de Interesse (ROI) definida à frente do robô
  - Eliminação de ruídos e falsas detecções

### 3.3 Lógica de Controle

Comandos pré-setados e enviados automaticamente pelo algoritmo conforme tabela 2:

Tabela 2: Comandos enviados via interface ao robô ESP32 S3

| Comando          | Motor Esquerdo | Motor Direito |
|------------------|----------------|---------------|
| Frente           | +70%           | +70%          |
| Virar à direita  | +70%           | +50%          |
| Virar à esquerda | +50%           | +70%          |
| Ré               | -50%           | -50%          |

Fonte: Autoria Própria (2025)

#### Fluxo Automático:

1. Cálculo do ângulo entre os vetores (círculos e linha)
2. Tomada de decisão baseada no ângulo calculado
3. Envio de comandos via HTTP para a ESP32

### 3.4 Implementação do Software

#### Estrutura do Código:

O sistema foi implementado em Python utilizando OpenCV para processamento de imagens e PyQt5 para a interface gráfica, seguindo uma abordagem modular com três componentes principais. A classe CircleDetector executa a detecção de círculos através da função `cv2.HoughCircles()`, realizando conversão para escala de cinza, aplicação de filtro Gaussiano e configuração dos parâmetros de detecção (`dp`, `minDist`, `param1` e `param2`), retornando as coordenadas e raios dos círculos identificados. A classe LineDetectorInteractive processa a linha de navegação convertendo a imagem para HSV, aplicando limiarização, operações morfológicas e `cv2.HoughLinesP()` para detectar segmentos lineares dentro de uma região de interesse pré-definida. Já a classe RobotController gerencia a comunicação com o hardware através de conexão WiFi, enviando comandos HTTP para a ESP32 que controla os motores, implementando a lógica básica de movimentação (avanço, curvas e recuo).

Figura 4: Interface gráfica de comunicação desenvolvida em PyQt



Fonte: Autoria Própria (2025)

### Ciclo de Controle:

1. Captura da imagem (webcam)
2. Processamento visual
3. Cálculo da direção
4. Envio do comando
5. Atualização da interface

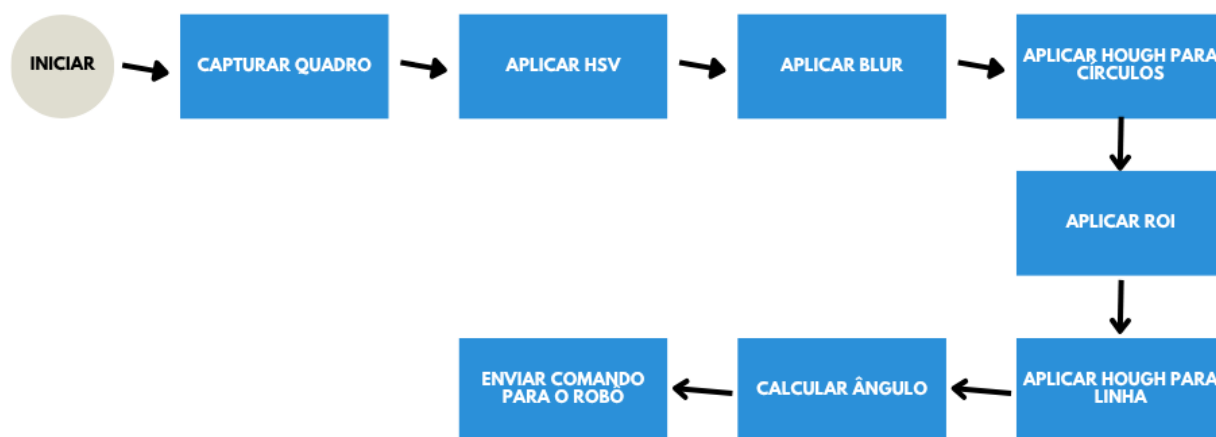
O fluxograma apresenta o ciclo completo de processamento e controle do robô seguidor de linha, conforme implementado no projeto. O fluxo segue uma sequência lógica de operações:

1. **INICIAR:** O sistema é inicializado, carregando todos os parâmetros de configuração, bibliotecas e estabelecendo a conexão com o robô.
2. **CAPTURAR QUADRO:** A câmera do sistema obtém um novo quadro (frame) do ambiente, que contém a pista com a linha de navegação e os círculos de referência.
3. **APLICAR HSV:** A imagem colorida é convertida do espaço de cores RGB para HSV, que permite melhor segmentação da linha preta e dos marcadores coloridos, sendo menos sensível a variações de iluminação.
4. **APLICAR BLUR:** Um filtro de suavização (como o Gaussiano) é aplicado para reduzir ruídos e pequenas imperfeições na imagem, preparando-a para as etapas seguintes de processamento.
5. **APLICAR HOUGH PARA CÍRCULOS:** A Transformada de Hough é utilizada para detectar os círculos de referência na imagem. Esta etapa identifica com precisão a posição e o tamanho dos marcadores circulares.

6. **APLICAR ROI:** Define-se uma Região de Interesse (ROI) à frente do robô, delimitando a área onde será realizada a detecção da linha. Isso aumenta a eficiência do processamento ao ignorar áreas irrelevantes da imagem.
7. **APLICAR HOUGH PARA LINHAS:** Na ROI definida, aplica-se a Transformada de Hough Probabilística para detectar segmentos de linha, identificando assim a trajetória que o robô deve seguir.
8. **CALCULAR ÂNGULO:** Com base nas posições dos círculos detectados e dos segmentos de linha, calcula-se o ângulo de orientação necessário para o robô seguir a trajetória correta.
9. **ENVIAR COMANDO PARA O ROBÔ:** O ângulo calculado é convertido em comandos de movimento específicos (avançar, virar à esquerda/direita ou recuar) que são enviados ao microcontrolador ESP32 via comunicação sem fio.

Este ciclo se repete continuamente enquanto o sistema está em operação, criando um laço fechado de realimentação que permite ao robô ajustar sua trajetória em tempo real, mantendo-se alinhado com a linha de navegação e utilizando os círculos como referência para orientação inicial. A abordagem mostrou-se eficiente tanto na detecção precisa dos elementos visuais quanto na resposta rápida do sistema de controle.

Figura 5: Fluxograma do algoritmo



Fonte: Autoria Própria (2025)

## 4 RESULTADOS E DISCUSSÃO

O sistema desenvolvido apresentou desempenho satisfatório nos testes realizados, validando a eficácia da abordagem baseada em visão computacional para navegação autônoma. Os resultados foram analisados sob três aspectos principais: detecção de elementos visuais, estratégia de controle e robustez do sistema, permitindo uma avaliação crítica das escolhas técnicas adotadas.

### 4.1 Detecção de Elementos Visuais

A detecção dos círculos de referência utilizando `cv2.HoughCircles()` mostrou-se estável após ajuste dos parâmetros (`dp=1.2`, `minDist=100`, `param1=50`, `param2=30`), garantindo identificação consistente dos marcadores. A formação de um vetor entre os centros dos



círculos provou-se eficiente para orientação inicial do robô, corroborando a escolha da Transformada de Hough para detecção de formas geométricas em ambientes estruturados (Gonzalez & Woods, 2010).

A segmentação da linha de navegação no espaço HSV, seguida pela aplicação da Transformada de Hough Probabilística (`cv2.HoughLinesP()`), demonstrou maior eficiência em comparação a abordagens baseadas apenas em limiarização RGB, reduzindo falsas detecções causadas por variações de iluminação. A definição de uma Região de Interesse (ROI) contribuiu significativamente para minimizar interferências de ruídos visuais, aumentando a confiabilidade do sistema.

## 4.2 Estratégia de Controle e Desempenho

O sistema de controle implementado, baseado em comandos pré-definidos (Tabela 2), respondeu adequadamente às variações de trajetória, executando movimentos de avanço, curvas e recuo conforme necessário. A lógica de decisão por ângulo relativo entre os círculos e a linha mostrou-se eficaz para navegação em percursos fechados, embora apresente limitações em trajetórias mais complexas, como curvas acentuadas ou intersecções.

A interface gráfica desenvolvida em PyQt5 permitiu monitoramento em tempo real e ajustes manuais quando necessário, facilitando a validação experimental. Contudo, observou-se que a latência na comunicação WiFi (entre o algoritmo e o ESP32) em alguns casos introduziu pequenos atrasos na resposta do robô, um aspecto que poderia ser otimizado em futuras implementações com protocolos mais eficientes (e.g., comunicação serial direta ou MQTT).

## 4.3 Robustez e Limitações

O sistema mostrou-se robusto em condições controladas de laboratório, mantendo desempenho consistente sob variações moderadas de iluminação. Entretanto, testes em ambientes com iluminação não uniforme ou fundos não brancos revelaram desafios adicionais na segmentação da linha, indicando que a abordagem atual depende criticamente do contraste entre a pista e a linha preta.

Apesar disso, os resultados confirmam que técnicas clássicas de PDI (Processamento Digital de Imagens) — como filtragem espacial, transformada de Hough e operações morfológicas — são suficientes para navegação autônoma em ambientes estruturados, alinhando-se com trabalhos anteriores na área (Duda & Hart, 1972). A escolha por métodos determinísticos (em vez de redes neurais) justificou-se pela transparência do processo decisório e menor demanda computacional, aspectos pedagógicos relevantes para o escopo do projeto.

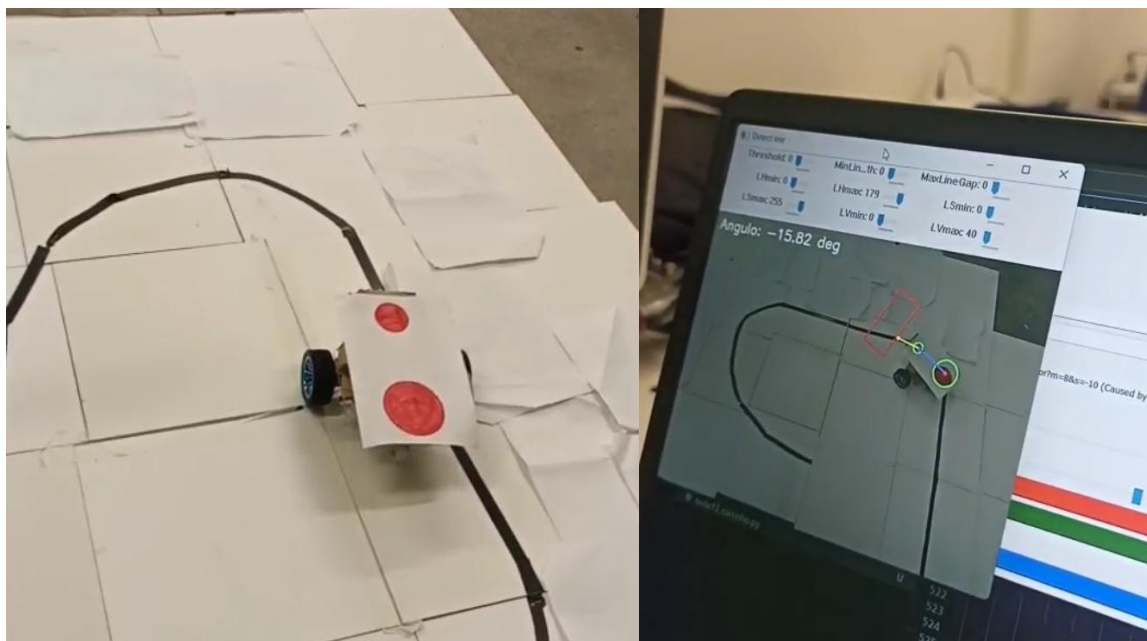
## 4.4 Perspectivas de Melhoria

Embora a solução atual atenda aos objetivos propostos, dois aprimoramentos são sugeridos:

- Implementação de filtros adaptativos (e.g., CLAHE para normalização de iluminação) para aumentar robustez em cenários dinâmicos.
- Exploração de métodos híbridos, combinando técnicas clássicas com redes neurais convolucionais (CNNs) para detecção de padrões em cenários mais complexos.

Essas melhorias poderiam estender a aplicabilidade do sistema sem comprometer sua eficiência computacional, um requisito crítico para plataformas embarcadas como a ESP32.

Figura 6: Prática em funcionamento



Fonte: Autoria Própria (2025)

## 5 CONSIDERAÇÕES FINAIS

Este trabalho demonstrou a viabilidade prática de integrar técnicas consolidadas de visão computacional - como segmentação por cor no espaço HSV, operações morfológicas e a Transformada de Hough - com sistemas embarcados baseados no microcontrolador ESP32 para controle de um robô móvel autônomo. A abordagem adotada proporcionou um valioso ambiente de aprendizagem para os alunos de pós-graduação, permitindo o desenvolvimento simultâneo de competências técnicas e analíticas no campo de automação e visão computacional.

Os resultados obtidos confirmaram que métodos clássicos de processamento de imagens, quando devidamente implementados e ajustados, podem oferecer soluções eficazes para problemas de navegação robótica em ambientes controlados. O sistema desenvolvido mostrou-se capaz de detectar com precisão os marcadores visuais e seguir a trajetória proposta, validando as escolhas técnicas adotadas no projeto.

Como perspectiva para trabalhos futuros, sugere-se a exploração de duas principais vertentes: a implementação de redes neurais convolucionais para aumentar a robustez do sistema em condições variáveis, e a combinação entre métodos tradicionais e técnicas de aprendizado de máquina, buscando aproveitar as vantagens de ambas as abordagens. Estas evoluções poderiam permitir a aplicação do sistema em cenários mais complexos, mantendo ao mesmo tempo o caráter pedagógico que se mostrou tão eficaz neste projeto.

## 6 REFERÊNCIAS

ABDI – AGÊNCIA BRASILEIRA DE DESENVOLVIMENTO INDUSTRIAL. **Indústria 4.0: a nova revolução industrial**. Brasília: ABDI, 2019.

CORDEIRO, R.; REIS, D.; FERNANDES, L. **Desafios para a Indústria 4.0 no Brasil**. *Revista Brasileira de Engenharia*, v. 45, n. 2, p. 23-39, 2024.

CNI – CONFEDERAÇÃO NACIONAL DA INDÚSTRIA. **Indústria 4.0 e o futuro do trabalho**. Brasília: CNI, 2016.

DUDA, R. O.; HART, P. E. **Use of the Hough transformation to detect lines and curves in pictures**. *Communications of the ACM*, v. 15, n. 1, p. 11–15, jan. 1972.

FERRIANI, A.; RIBEIRO, R. **Transformada de Hough na detecção de formas geométricas**. Anais, 2005.

FREIRE, Paulo. **Pedagogia da autonomia: saberes necessários à prática educativa**. 25. ed. São Paulo: Paz e Terra, 1996.

GONZALEZ, R. C.; WOODS, R. E. **Digital image processing**. 3. ed. Upper Saddle River: Prentice Hall, 2010.

