XI CONGRESSO DE INICIAÇÃO CIENTÍFICA DO IFSP ITAPETININGA
Itapetininga, 27, 28 e 29 de maio de 2025
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Campus Itapetininga

DESENVOLVIMENTO DE UM ROBÔ SEGUIDOR DE LINHA UTILIZANDO PYTHON E OPENCV COM BAIXO CUSTO COMPUTACIONAL

Lívia Cerqueira Sakai – Objetivo Itapetininga[1]
Prof. Esp. Josenil Ezequiel Costa - IFSP[2]

Introdução

Nos últimos anos, a tecnologia tornou-se parte integrante do cotidiano de estudantes de escolas públicas e privadas, principalmente por meio da inclusão de componentes curriculares como aulas maker, programação e robótica. Visando impulsionar e incentivar iniciativas educacionais voltadas à ciência e tecnologia, diversas organizações têm promovido competições acadêmicas. Nesse contexto, destaca-se, no Brasil, a Olimpíada Brasileira de Robótica (OBR), considerada uma das maiores do país. A OBR configura-se como uma olimpíada científica com foco na área de robótica, cuja primeira edição ocorreu em 2007, por iniciativa privada da empresa Lego Education. Atualmente, o evento é realizado anualmente, por meio de uma parceria público-privada entre o Instituto Federal do Rio Grande do Norte e a RoboCup, mantendo-se como uma olimpíada gratuita e sem fins lucrativos (OBR, 2025). A OBR contempla diferentes modalidades, como Robótica Artística, Seguidor de Linha e Resgate de Robôs. Na última modalidade, o destaque recai sobre o desafio proposto: os estudantes devem construir um robô autônomo capaz de seguir uma linha, superar diferentes tipos de obstáculos, localizar vítimas, resgatá-las e transportá-las até uma área segura. O robô utiliza a linha para sua orientação e sinais coloridos para decidir o melhor trajeto (OBR, 2025). Neste contexto, os avanços recentes em inteligência artificial proporcionaram a oportunidade de explorar a visão computacional — A visão computacional, por sua vez, é um campo da Inteligência Artificial que permite softwares identificar e interpretar objetos em imagens capturadas por câmeras (Gregersen, 2023) - Como uma poderosa ferramenta para identificação de caminhos e vítimas ao invés dos convencionais conjuntos de sensores de infravermelho e ultrassom. Este trabalho busca destacar as etapas e tecnologias necessárias para a construção de um sistema com essas capacidades, utilizando ferramentas como Python — atualmente considerada uma das linguagens mais adequadas para projetos de inteligência artificial, devido à sua sintaxe simples — e bibliotecas como a OpenCV (Open Source Computer Vision Library), que reúne algoritmos de visão computacional voltados a auxiliar o computador na interpretação e análise de imagens e vídeos (Leal; Heinen; Neves, 2019).

[1]Estudante do Ensino Fundamental Anos Finais, Colégio Objetivo Itapetininga — Itapetininga/SP. E-mail do primeiro autor: liviacerqueirasakai511@gmail.com.
[2]Especialista em Informática Aplicada a Educação, Colégio Objetivo Itapetininga, Itapetininga/SP.E-mail do autor: josenilezequiel@gmail.com

Objetivo

O presente trabalho pretende:(i)Identificar e descrever as tecnologias essenciais para a construção de um robô de resgate destinado à participação na competição OBR, com ênfase na utilização de visão computacional;(ii)Determinar a configuração mínima de hardware e software necessária para o funcionamento adequado do sistema; (iii)Documentar as etapas do processamento de imagens, bem como os procedimentos de comunicação entre os módulos de visão computacional e o microcontrolador responsável pelo controle do robô.

Metodologia

Os recursos utilizados neste trabalho limita-se ao seguinte ambiente de software descritos na Tabela 1:

Tabela 1 **-** Ambiente de Softwares

Software	Versão	Função
Python	3.9.12	Linguagem de programação desenvolvimento dos scripts, selecionada por sua ampla compatibilidade de recursos específicos para modelos de IA.
OpenCV	4.11.0.86	Biblioteca de visão computacional open-source capaz de fornecer as funções básicas para manipulação de imagem e reconhecimento
Numpy	1.26.4	Biblioteca específica para operações matemáticas complexas, auxiliando o algoritmo de visão computacional identificar linhas e objetos no vídeo
Media Pipe	0.10.21	Biblioteca que possui diversas funções de visão computacional que atua no processamento de pipelines e Machine Learning.

Fonte: do Autor, 2025

E o hardware para execução estão descritos na Tabela 2 a seguir:

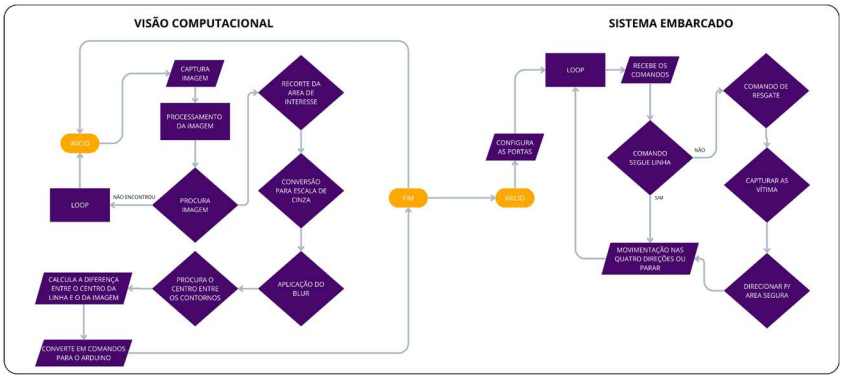
Tabela 2 - Plataforma de Hardware

Hardware	Função
Câmera	Capturar a imagem
Raspberry Pi V3	Micro computador dedicado ao processamento do algoritmo de visão computacional
Arduino Uno	Microcontrolador específico para controlar as funções básicas do robô.
Atuadores	Inclui, motores CC e Servos, sendo componentes base para o robô cumprir sua função de se movimentar e resgatar as vítimas

Fonte: do Autor, 2025

Com os componentes de hardware e software elencados, a etapa de desenvolvimento do chassi do robô está sendo elaborada paralelamente ao desenvolvimento dos softwares, por outra equipe especializada. O robô será construído utilizando um chassi de MDF desenvolvido no colégio Objetivo pela equipe de engenharia, protótipo exibido na figura x. Na parte dianteira será adicionado o sistema de garra e a webcam para facilitar a visualização da linha, vítimas e obstáculos. Para a movimentação do robô considera-se utilização de dois motores de corrente contínua capazes de fornecer o movimento para seis rodas que compõem o chassi. A movimentação do sistema de garra para captura das vítimas, será utilizado servos motores de 9g. O sistema de computador de bordo será composto por um microcomputador Raspberry Pi responsável por processar as imagens, convertendo os resultados do reconhecimento em comandos enviados ao Arduíno Uno via comunicação serial para execução das ações desejadas, além das baterias e um powerbank para fornecer energia ao sistema.

Figura 1 - DIAGRAMA DE SOFTWARE: VISÃO COMPUTACIONAL X EMBARCADOS



Fonte: do Autor, 2025

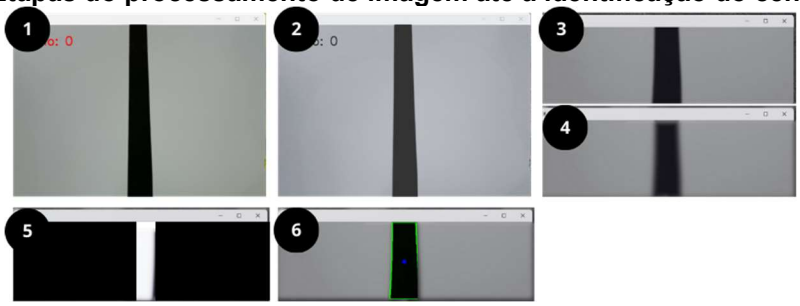
Pipeline de Visão Computacional: O processo tem início com a captura das imagens por meio de uma câmera RGB, com resolução de 640 × 480 píxeis. Essas imagens podem ser representadas como matrizes bidimensionais, em que cada elemento — denominado píxel — contém informações relativas à sua posição na matriz e à sua cor, codificada no espaço RGB (Red, Green, Blue). Cada canal de cor apresenta um intervalo de intensidade que

varia de 0 a 255, permitindo a representação de uma ampla gama de cores (Bradski, 2000). Na etapa seguinte, a imagem é redimensionada, com redução de 50% em seu tamanho, visando otimizar o uso de recursos computacionais nas fases subsequentes do processamento. Em seguida, realiza-se a conversão da imagem para escala de cinza. Essa conversão é fundamental para a redução da complexidade computacional, uma vez que a imagem em RGB possui três canais de cor altamente sensíveis a variações de iluminação, o que pode comprometer a eficácia dos algoritmos de reconhecimento (Gonzalez; Woods, 2018). A escala de cinza, por sua vez, mantém as informações essenciais de contraste e forma, utilizando somente um canal, tornando o processamento mais eficiente e menos suscetível a ruídos, sem comprometer a detecção de elementos como linhas, quadrados e círculos (Silva; Kobayashi, 2015). Posteriormente, é realizada a seleção de uma região de interesse (ROI) da imagem original. Essa etapa visa evitar a análise de áreas irrelevantes, que poderiam consumir processamento adicional e afetar negativamente a acurácia do reconhecimento. Sobre essa região recortada, aplica-se um filtro Gaussiano, que gera um leve desfoque com o propósito de eliminar ruídos — pequenas variações de cor e forma — que poderiam dificultar a detecção de bordas e contornos (Gonzalez; Woods, 2018). A imagem suavizada é então submetida ao algoritmo de Canny, que tem como objetivo a detecção de bordas (Canny, 1986). Em seguida, utiliza-se a técnica de limiarização binária com o método de Otsu, por meio da função cv2.threshold. Essa técnica converte a imagem para uma representação em preto e branco, sendo que o limiar é calculado automaticamente pelo método de Otsu, otimizando a separação entre objeto e fundo (Otsu, 1979). Além disso, utiliza-se a *flag THRESH_BINARY_INV,* que inverte a imagem, tornando os objetos escuros em claros sobre fundo escuro, facilitando o processo de extração de contornos. Com a imagem binarizada, aplica-se a função *cv2.findContours*, cuja finalidade é identificar contornos presentes na imagem (Bradski, 2000). Como o objetivo é localizar somente as formas externas, emprega-se o modo *cv2.RETR_EXTERNAL*, que descarta contornos internos (como furos e detalhes irrelevantes). Para otimizar o consumo de memória, utiliza-se também o parâmetro *CHAIN_APPROX_SIMPLE*, responsável por eliminar pontos redundantes. Na etapa seguinte, os contornos detectados são analisados: primeiramente, descartam-se aqueles que não atingem um tamanho mínimo pré-estabelecido. O contorno de maior área é então considerado o principal objeto de interesse. Sobre esse contorno, aplica-se a função cv2.moments, que permite o cálculo de propriedades geométricas, como o centroide (centro geométrico da figura). A Pipeline do reconhecimento do caminho x comando: Com o centro geométrico determinado, calcula-se o desvio em relação ao centro horizontal da imagem, permitindo inferir a direção a ser seguida. Essa informação é utilizada para definir os comandos a serem enviados ao motor, de forma que o sistema possa ajustar sua trajetória. As demais partes do algoritmo concentram-se na geração de saídas para fins de depuração e monitoramento.

Resultados

Até o momento, o sistema desenvolvido consegue emitir quatro comandos básicos de controle: parar (0), frente (1), direita (2), esquerda (3) e atrás (4). Esses comandos são gerados com base na análise da imagem tratada, especialmente na identificação do centro geométrico da linha detectada. A Figura 1 ilustra as diferentes etapas do processamento de imagem, desde a captura inicial até a identificação do centro da linha.

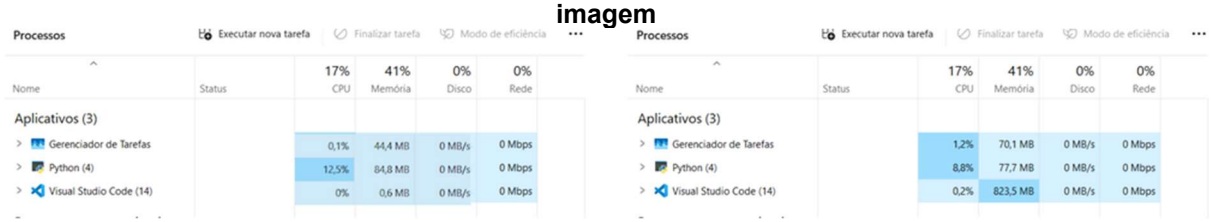
Figura 2 – Etapas do processamento de imagem até a identificação do centro da linha



Fonte: do Autor, 2025

Observa-se que a redução da imagem em 50% impacta significativamente o desempenho do sistema, proporcionando uma diminuição de aproximadamente 33% no consumo de recursos computacionais. Esse ganho se mostra relevante, especialmente em aplicações embarcadas com capacidade de processamento limitada. A Figura 3 apresenta a comparação entre o consumo de processamento com e sem a redução da imagem.

Figura 3 – Comparação do consumo de processamento com e sem redimensionamento da imagem



Fonte: do Autor, 2025

Além disso, é possível observar, conforme demonstrado na Figura 4, o **cálculo do desvio do centro geométrico** em relação ao eixo central da imagem. Esse desvio é utilizado como parâmetro de decisão para o envio dos comandos de movimento: frente, parar, direita ou esquerda. Essa conversão do desvio em comandos permite que o sistema reaja dinamicamente à posição da linha detectada no campo de visão.

Figura 4 – Interpretação do desvio do centro da linha em comandos de controle:

Fonte: do Autor, 2025

Conclusão

O desenvolvimento de um robô seguidor de linha utilizando Python e OpenCV demonstrou ser uma alternativa viável, acessível e eficaz para o escopo de projetos educacionais voltados à robótica e à inteligência artificial, especialmente no contexto de competições

como a Olimpíada Brasileira de Robótica. A implementação de um pipeline de visão computacional com baixo custo computacional permitiu a detecção eficiente de caminhos e a tomada de decisões em tempo real, mesmo em plataformas embarcadas com recursos limitados, como o Raspberry Pi. O sistema proposto apresentou resultados promissores, com comandos básicos de navegação sendo gerados precisamente a partir da análise de imagens, e com otimizações como a redução da resolução contribuindo significativamente para o desempenho geral. O uso combinado de bibliotecas como OpenCV, Numpy e MediaPipe mostrou-se adequado para o processamento eficiente de imagens e para a extração de informações relevantes para o controle do robô. Além de seu valor técnico, o projeto destaca-se pelo potencial pedagógico, promovendo o aprendizado interdisciplinar ao integrar conhecimentos de matemática, física, lógica de programação e eletrônica. Os próximos passos do trabalho incluem o aprimoramento da precisão do sistema de reconhecimento, a expansão dos comandos de controle e a integração de sensores adicionais que permitam ao robô lidar com cenários mais complexos, como obstáculos variados e reconhecimento de cores ou formas específicas. Com isso, espera-se que este projeto possa não somente contribuir para a participação qualificada de estudantes em competições como a OBR, mas também servir como inspiração para iniciativas de ensino que visem democratizar o acesso à robótica educacional no Brasil.

Referências

ANDRADE, Daniel Spillere. **Projeto: Robô Seguidor de Linha.** 2013. 35 f. TCC (Graduação) - Curso de Engenharia Elétrica, Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 2013. Disponível em: https://danielandrade.net/uploads/images/2014/08/relatorio-final.pdf. Acesso em: 11 maio 2025.

BRADSKI, Gary. **The OpenCV Library.** Dr. Dobb's Journal of Software Tools, 2000. Disponível em: https://docs.opencv.org/. Acesso em: 10 maio 2025.

CANNY, John. **A computational approach to edge detection**. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. PAMI-8, n. 6, p. 679–698, 1986.

DIAS, Vinícius; JUNIORF, Luiz. Seguidor de linha utilizando controlador fuzzy baseado em visão computacional. **Centro de Ciências Exatas e Tecnológicas**, Cruz das Almas, p. 376-385, 22 ago. 2018. Disponível em: https://sol.sbc.org.br/index.php/erbase/article/view/8560/8461. Acesso em: 11 maio 2025.

GONZALEZ, Rafael C.; WOODS, Richard E. **Digital Image Processing**. 4. ed. New York: Pearson, 2018.

KONFLANZ, Renato Mello; MORETTO, Marcos Antonio. Protótipo de um Algoritmo para Robô Seguidor de Piso Tátil Utilizando Visão Computacional. **Revista Eletrônica de Iniciação Científica em Computação**, Chapecó, v. 18, p. 1-16, 03 abr. 2020. Disponível em: https://seer.ufrgs.br/index.php/reic/article/view/98469/56218. Acesso em: 07 maio 2025.

LEAL, Glédson; HEINEN, Milton; NEVES, Bruno. Uma Proposta para Implementação de Robô Seguidor de Linha com Visão Computacional. **Computer On The Beach**, Bagé, v. 1, p. 822-824, 2019. Anual. Disponível em: https://periodicos.univali.br/index.php/acotb/article/view/14434. Acesso em: 07 maio 2025.

OBR (Brasil). **Manual de Regras e Instruções: presencial/ regional e estadual. Presencial/ Regional e Estadual.** 2025. Disponível em: https://obr.robocup.org.br/documentos-e-manuais/#. Acesso em: 07 maio 2025.

OTSU, Nobuyuki. A Threshold Selection Method from Gray-Level Histograms. **Ieee Transactions On Systems, Man, And Cybernetics**, [S.L.], v. 9, n. 1, p. 62-66, jan. 1979. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tsmc.1979.4310076.

SILVA, José Carlos da; KOBAYASHI, Hélio. **Processamento Digital de Imagens: princípios e aplicações**. São Paulo: Blucher, 2015.