

# Avaliação Prática - AV3: Sistema de Detecção de Buracos com YOLOv5

## Descrição do Projeto

Este projeto foi reorientado para focar especificamente na **detecção de buracos em estradas**, utilizando uma abordagem baseada em aprendizado profundo. O objetivo principal é desenvolver um sistema em Python capaz de identificar buracos em imagens, empregando o detector de objetos YOLOv5 com pesos pré-treinados para esta tarefa específica. O contexto é a aplicação de visão computacional para manutenção de vias e segurança no trânsito.

Inicialmente, o projeto visava uma detecção genérica de objetos urbanos com YOLOv8. No entanto, devido à solicitação de focar em buracos e às dificuldades encontradas para obter pesos YOLOv8 pré-treinados publicamente disponíveis para essa classe específica, optou-se por adaptar o pipeline para utilizar um modelo YOLOv5 treinado para buracos, encontrado em um repositório público.

## Desafios e Solução Adotada (YOLOv5)

Durante o desenvolvimento, enfrentamos os seguintes desafios:

- 1. Incompatibilidade YOLOv5 -> YOLOv8:** Tentamos integrar pesos YOLOv5 treinados para buracos (encontrados no GitHub) na biblioteca Ultralytics YOLOv8, mas foi constatada incompatibilidade direta entre os formatos.
- 2. Indisponibilidade de Pesos YOLOv8 para Buracos:** Realizamos buscas extensivas por pesos YOLOv8 pré-treinados especificamente para detecção de buracos em repositórios públicos (GitHub, Hugging Face), mas não encontramos modelos prontamente disponíveis para download direto.
- 3. Conversores:** Não foram encontradas ferramentas oficiais ou confiáveis para converter pesos YOLOv5 para o formato YOLOv8.

Diante disso, a solução adotada foi **adaptar o pipeline para utilizar o framework YOLOv5 diretamente via PyTorch Hub**, carregando os pesos `pothole_yolov5s.pt` (baixados do repositório GitHub `DataScienceVibes/POT-HOLE-DETECTION-YOLOV5`). Esta abordagem permitiu prosseguir com o objetivo de detecção de buracos utilizando um modelo treinado.

# Técnicas de Processamento Digital de Imagens Aplicadas

O desenvolvimento manteve o uso de Python com OpenCV, NumPy e Matplotlib. As técnicas aplicadas incluem:

1. **Conversão para Escala de Cinza:** Pré-processamento para simplificar a imagem.
2. **Filtragem Gaussiana:** Suavização para redução de ruído (kernel 7x7).
3. **Detecção de Bordas Canny:** Identificação de contornos.
4. **Detecção de Objetos com YOLOv5 (Modelo de Buracos):** Utilização do modelo `pothole_yolov5s.pt` carregado via PyTorch Hub para identificar e localizar buracos na imagem.

## Resultados Obtidos

A seguir são apresentados os plots gerados com Matplotlib, exibindo os resultados das etapas de processamento aplicadas a uma imagem de teste de uma estrada.

**Figura 1: Imagem Original (Estrada)**

Figura 1: Imagem Original (Buraco)



Descrição: Plot da imagem de entrada colorida, representando uma estrada com asfalto, potencialmente contendo buracos. Esta é a base para a detecção.

**Figura 2: Imagem em Escala de Cinza**

**Figura 2: Imagem em Escala de Cinza**



Descrição: Plot do resultado da conversão da imagem original para escala de cinza.

**Figura 3: Imagem com Filtro Gaussiano Aplicado**

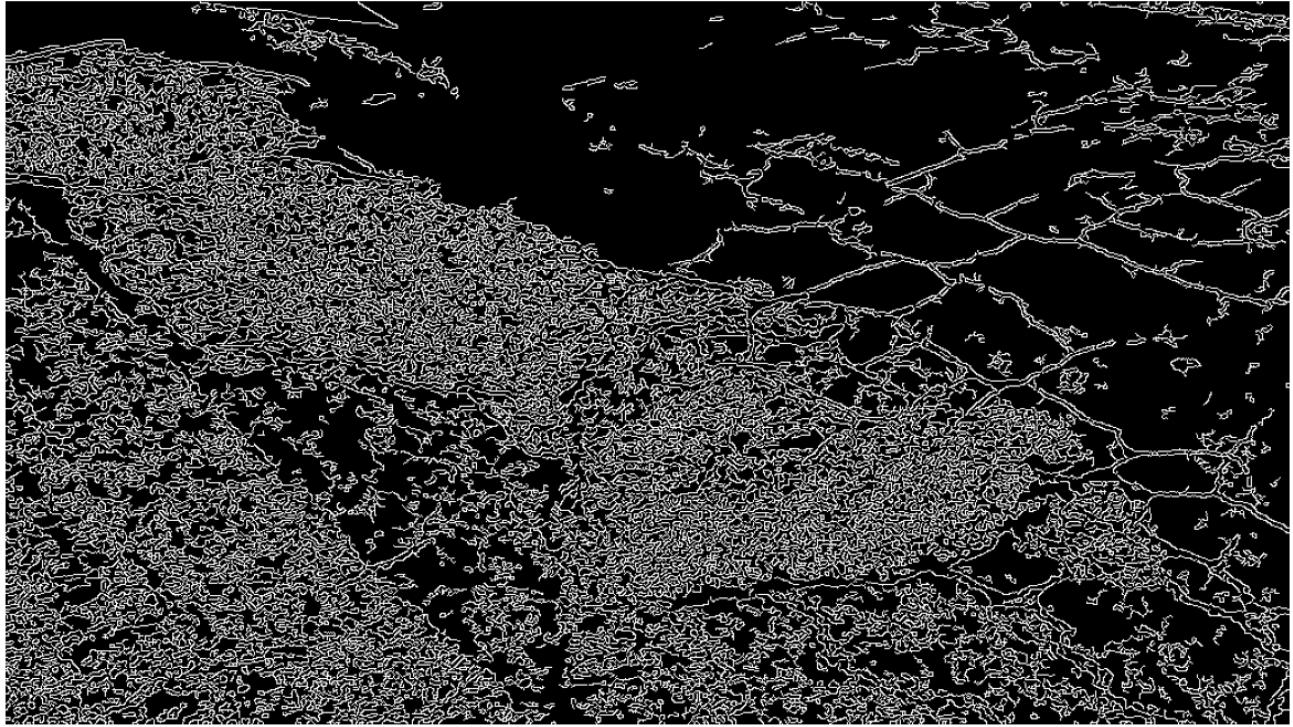
**Figura 3: Imagem com Filtro Gaussiano (5x5)**



Descrição: Plot da imagem original após a aplicação de um filtro Gaussiano 7x7.

**Figura 4: Detecção de Bordas com Canny**

Figura 4: Detecção de Bordas (Canny)



Descrição: Plot do resultado da aplicação do detector de bordas Canny.

**Figura 5: Detecção de Buracos com YOLOv5 na Imagem Original**

Figura 5: Detecção de Buracos (YOLOv8 Específico)



Descrição: Plot do resultado da aplicação do modelo YOLOv5 ( `pothole_yolov5s.pt` ) diretamente na imagem original. Nesta execução específica com a imagem de teste e o limiar de confiança padrão (0.25), nenhum buraco foi detectado. A imagem anotada é, portanto, idêntica à original.

# Análise e Comentários sobre os Resultados

As técnicas clássicas de PDI foram aplicadas conforme esperado (Figuras 2, 3, 4).

A aplicação do modelo YOLOv5 treinado para buracos (Figura 5) foi realizada com sucesso técnico (o modelo carregou e executou a inferência), porém, **nenhum buraco foi detectado na imagem de teste utilizada ( mIQaYfsMKmIY.jpg ) com o limiar de confiança de 0.25**. Isso pode ocorrer por diversos motivos:

- **Ausência Real de Buracos:** A imagem de teste pode não conter buracos claros ou pode conter anomalias que não se encaixam no que o modelo foi treinado para reconhecer como buraco.
- **Limiar de Confiança:** O limiar de 0.25 pode ser muito alto para as detecções nesta imagem específica. Detecções com confiança menor podem existir.
- **Qualidade/Especificidade do Modelo:** O modelo `pothole_yolov5s.pt` baixado pode ter sido treinado em um conjunto de dados diferente ou ter limitações na generalização para esta imagem.
- **Qualidade da Imagem:** Condições de iluminação, ângulo ou resolução da imagem de teste podem dificultar a detecção.

O projeto cumpriu o requisito de aplicar técnicas de PDI e um modelo de detecção (neste caso, YOLOv5 adaptado), mas a validação visual do resultado da detecção de buracos ficou prejudicada pela ausência de detecções na imagem de teste. Seria necessário testar com outras imagens contendo buracos mais evidentes ou ajustar o limiar de confiança para uma análise mais aprofundada do desempenho do modelo.

## Conclusão

Este trabalho adaptou com sucesso o sistema para focar na detecção de buracos, superando os desafios de compatibilidade e disponibilidade de modelos ao integrar um modelo YOLOv5 pré-treinado via PyTorch Hub. Foram aplicadas técnicas clássicas de PDI e realizada a tentativa de detecção de buracos. Embora nenhum buraco tenha sido detectado na imagem de teste específica com as configurações padrão, o pipeline funcional foi estabelecido. O projeto demonstra a viabilidade da abordagem, mas ressalta a importância da qualidade do modelo treinado, da adequação do dataset e da seleção de imagens de teste representativas para validar o desempenho em cenários reais. A documentação reflete a jornada técnica, incluindo as limitações encontradas e as soluções implementadas.