UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

ALLEX LEMOS DE SOUZA PINHEIRO
DÉBORA DIANA GONÇALVES DOS SANTOS
GUILHERME HENRIQUE SANTOS ARAÚJO
MIGUEL LUCAS SANTANA FREIRE

RELATÓRIO

TRABALHO PRÁTICO

São Cristóvão - SE
2025

ALLEX LEMOS DE SOUZA PINHEIRO
DÉBORA DIANA GONÇALVES DOS SANTOS
GUILHERME HENRIQUE SANTOS ARAÚJO
MIGUEL LUCAS SANTANA FREIRE

RELATÓRIO

TRABALHO PRÁTICO

Trabalho apresentado à Profª. Drª. Beatriz Trinchão como requisito de avaliação da Disciplina Processamento de Imagens da Universidade Federal de Sergipe - UFS.

São Cristóvão–SE
2025

1. INTRODUÇÃO

Este trabalho apresenta o desenvolvimento de um sistema de visão computacional para a empresa fictícia *GenericStuff*. O objetivo do sistema é processar imagens binárias capturadas em uma linha de produção para identificar, contar e classificar objetos em duas categorias: objetos com furos e objetos sem furos (sólidos). A solução foi implementada em linguagem Java, utilizando algoritmos fundamentais de processamento de imagens digitais.

2. IMPLEMENTAÇÃO TÉCNICA

Para resolver o problema da contagem e classificação, utilizamos a técnica de Flood Fill (Preenchimento por Inundação). O algoritmo opera em três etapas principais:

a. **Isolamento do Fundo:** Iniciamos uma inundação a partir da coordenada (0,0) para identificar todo o fundo externo da imagem.
b. **Detecção de Objetos:** Percorremos a matriz de pixels. Ao encontrar um pixel de valor 1 (objeto) que ainda não foi visitado, iniciamos um novo processo de inundação. Conforme especificado, utilizamos a vizinhança-8 para definir a conectividade dos objetos.
c. **Classificação (Detecção de Furos):** Durante o preenchimento de um objeto, verificamos seus vizinhos. Se o objeto tocar em algum pixel de fundo (0) que *não* foi marcado como "fundo externo" na etapa 1, concluímos que se trata de um furo interno. Para os furos, utilizamos a vizinhança-4.

Soluções para Problemas Encontrados Durante o desenvolvimento, enfrentamos dois desafios principais:

- **Bordas da Imagem:** Para evitar verificações complexas de limites da matriz e erros de *ArrayIndexOutOfBounds*, implementamos a técnica de Padding. A imagem carregada é inserida em uma matriz maior, cercada por uma borda de pixels de fundo.
- **Estouro de Pilha (StackOverflow):** A abordagem recursiva tradicional do *Flood Fill* falhava em imagens grandes devido ao limite da pilha de execução do Java. A solução foi implementar uma versão iterativa do algoritmo, utilizando uma estrutura de dados `Deque` (onde os elementos podem ser inseridos ou excluídos no início ou fim) para gerenciar os pixels a serem visitados.

3. COMPILAÇÃO E EXECUÇÃO

O projeto foi desenvolvido em Java (JDK 17+) e não utiliza bibliotecas externas de processamento de imagem. O código fonte encontra-se na pasta `src` e as imagens de teste na pasta `samples`.

3.1. NO LINUX/MACOS:

1. **Compilar:** `javac -d bin src/inspector/*.java`

2. **Executar:** `sh run.sh samples/teste.pbm`

No Windows:

1. **Compilar:** Execute os comandos `mkdir bin` e `javac -d bin src\inspector\*.java`.
2. **Executar**: `run.bat samples\teste.pbm`

4. CONTEXTO E ANÁLISE ÉTICA

4.1. POTENCIAL DE USO

O algoritmo desenvolvido tem aplicação direta em cenários de Controle de Qualidade Industrial. Um exemplo prático seria uma fábrica de arruelas ou porcas metálicas. O sistema visual poderia ser acoplado a uma esteira para verificar se as peças foram estampadas corretamente (com furo) ou se estão defeituosas (sem furo/sólidas), acionando um mecanismo de descarte automático para as peças falhas.

4.2. ADAPTAÇÕES NECESSÁRIAS

Para viabilizar o uso no mundo real, o sistema necessitaria de:

- **Pré-processamento:** Câmeras industriais capturam imagens coloridas (RGB). Seria necessário implementar uma etapa de *Limiarização* para converter a imagem real em binária (Preto e Branco) antes de passar para o nosso algoritmo.
- **Hardware de Atuação:** Integração com Controladores Lógicos Programáveis para soprar ou empurrar as peças defeituosas para fora da esteira em tempo real.

4.3. ANÁLISE ÉTICA

A introdução desta tecnologia levanta algumas questões éticas importantes:

- **Substituição de Trabalho:** A automação da inspeção visual substitui operadores humanos. Embora isso possa gerar desemprego a curto prazo, também retira o ser humano de uma tarefa repetitiva, monótona e ergonomicamente nociva, permitindo a realocação para funções de supervisão mais complexas.
- **Responsabilidade e Segurança:** Em aplicações críticas, como para as peças de freios de aviões, uma falha no algoritmo poderia ter consequências catastróficas.

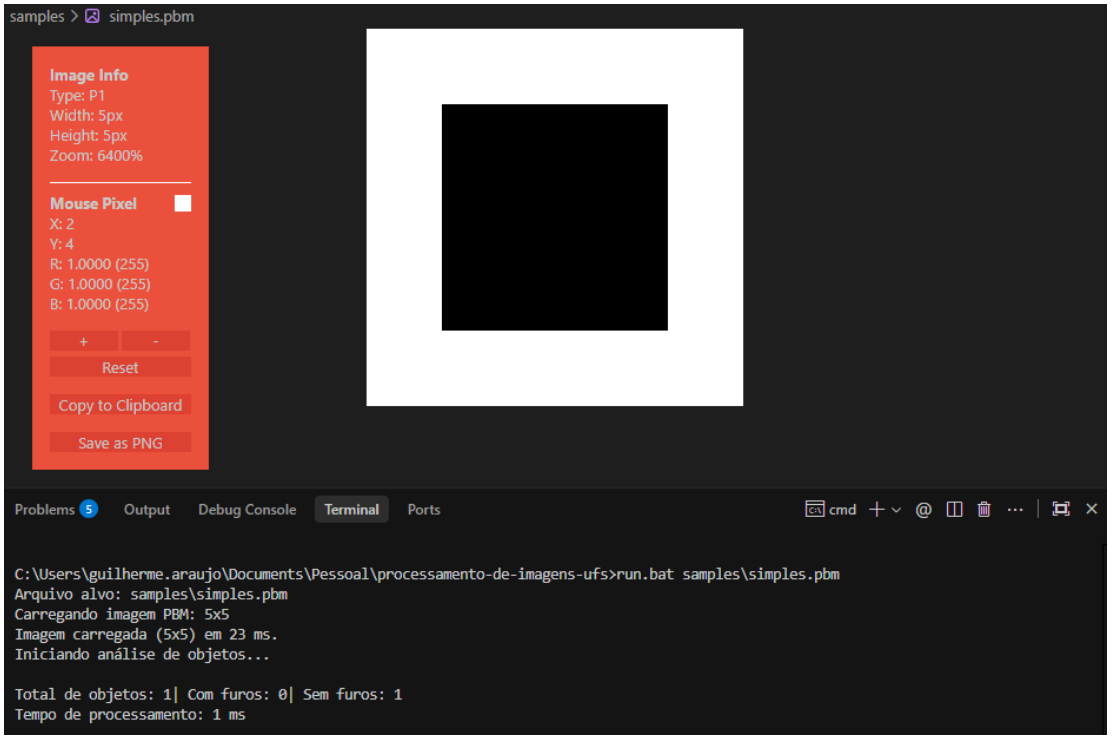
  Eticamente, o software deve ser exaustivamente testado e possuir margens de segurança claras, não devendo ser vendido como "infalível".

5. RESULTADOS

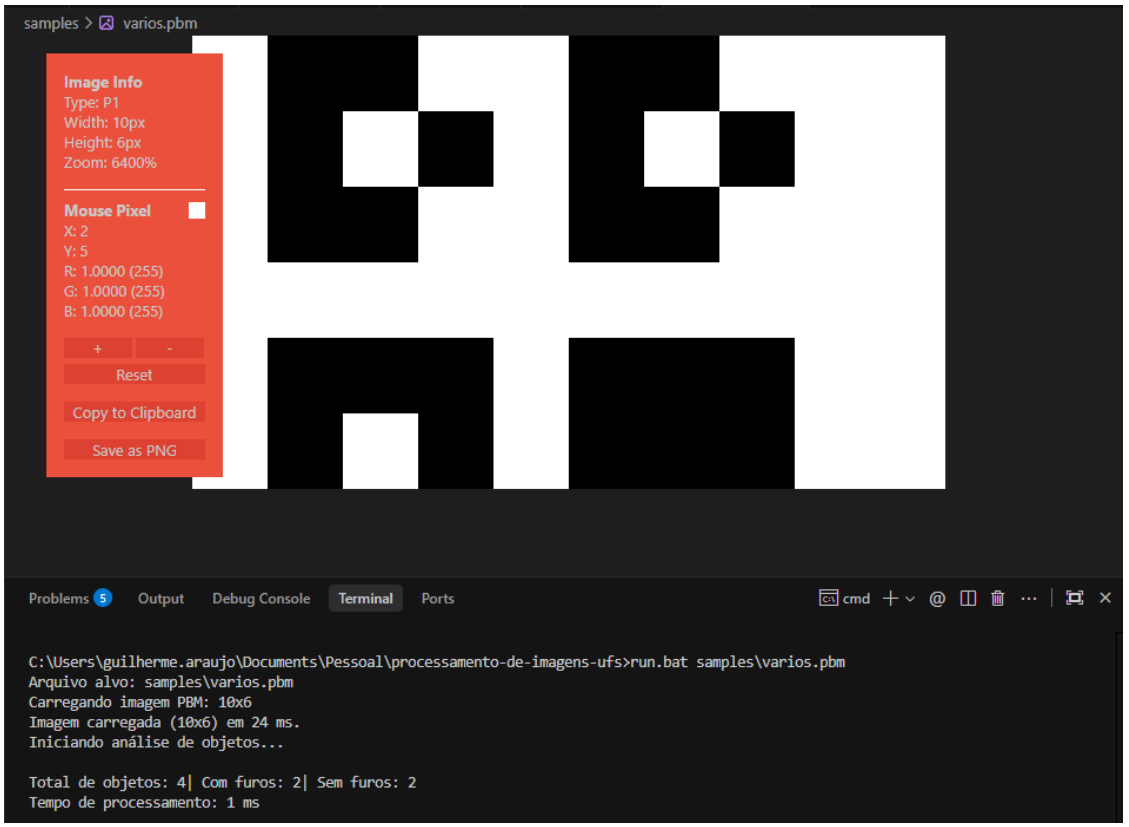
Link para o repositório: https://github.com/Slotov7/processamento-de-imagens-ufs

Teste 1 (teste.pbm)

Teste 2 (simples.pbm)



Teste 3 (varios.pbm)



REFERÊNCIAS

GONZALEZ, Rafael C.; WOODS, Richard E. ***Processamento Digital de Imagens***. 3. ed. São Paulo: Pearson, 2010.

ORACLE. ***Java Documentation***. Disponível em: https://docs.oracle.com/en/java/. Acesso em: 20 jan. 2026.