Atividade Avaliativa PDI 2024.pdf

# **Questão 1**

**Diferencie os conceitos de amostragem e quantização no processo de digitalização de uma imagem e a consequência desses processos na imagem digital.**

A amostragem e quantização são processos que fazem parte da aquisição da imagem digital. Numa imagem capturada a partir de um sinal analógico observam-se ruídos de sinal. Essa imagem é chamada continua e precisa ser convertida para um sinal digital.

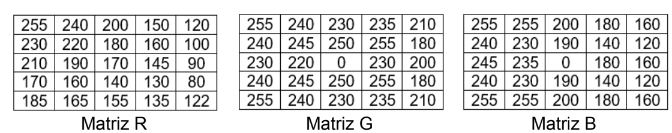
A imagem continua pode ser dividida em vários quadradinhos, conhecidos como pixels. Esse processo de dividir a imagem em pixels se chama **amostragem**. Quanto menos pixels forem usados para representar a imagem pior é sua qualidade (imagem pixelada). Em resumo, **amostragem** define quantos pixels usamos para representar a imagem.

Após dividir a imagem deve-se definir a cor que será “pintado” cada pixel. No mundo digital, não dá para representar infinitas variações de cor, então simplificamos: escolhemos um número limitado de cores ou tons de cinza. Esse processo é a **quantização**. Quantização define quantas cores ou tons diferentes cada pixel pode ter.

Na **quantização**, cada pixel é atribuído a um valor numérico que varia dentro de uma escala predefinida (ex.: 256 niveis em uma imagem de 8 bits). Quanto maior a quantização mais precisa será a representação de cores da imagem. A baixa quantização provoca um efeito conhecido como “banding” onde se tem uma imagem com transições abruptas de cores.

# Questão 2

**As matrizes abaixo correspondem às componentes RGB de uma imagem.**

****

**Calcule a imagem em tons de cinza resultante para essas três matrizes, criada:**

1. **através da Média;**
2. **através do uso de pesos conforme o slide 46 da aula de cores.**

**Comente os resultados que você encontrou.**

Para realizar essa conversão foi criado um código em Python anexado abaixo. Os resultados obtidos foram:

**Imagem em tons de cinza pela Média:**

[[255 245 210 188 163]

[238 232 207 185 133]

[232 222 203 180 130]

[218 212 193 175 127]

[232 202 188 177 164]]

**Imagem em tons de cinza com Pesos:**

[[255 242 218 203 177]

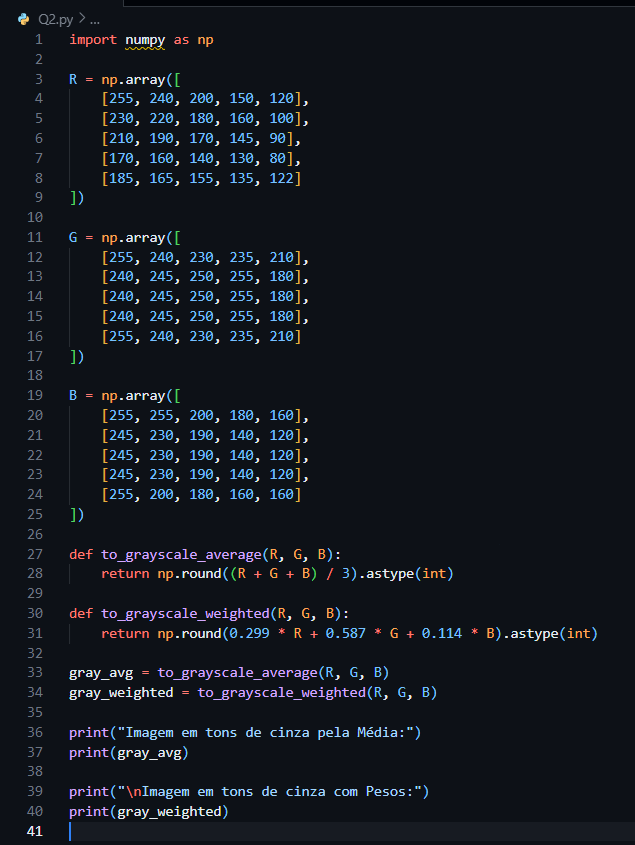
[238 236 222 213 149]

[232 227 219 209 146]

[220 218 210 205 143]

[234 213 202 197 178]]

A conversão pela média trata os três canais de forma igual. Ignorando o fator de percepção humana que detecta as cores vermelha e azul com mais dificuldade, dessa forma a imagem gerada pela média tem mais contraste enquanto a imagem convertida com pesos é mais suave e se adapta melhor a percepção da visão.



# Questão 3

**Considere uma imagem em tons de cinza, cujo histograma, em uma escala de 0 a 255, tem o formato abaixo:**

**Uma imagem contendo Diagrama

Descrição gerada automaticamente**

**Cite duas informações que esse histograma traz sobre essa imagem, explicando sua resposta.**

O histograma é uma representação gráfica que mostra a quantidade de pixels de cada tom de cor numa escala de cores (no caso acima temos 256 tons). Pode-se perceber nesse grafico que existe uma concentração de prixels num intervalo pequeno de tons, o que mostra um baixo contraste na imagem, ou seja , é dificil diferenciar formas pois a s cores estão com tons muito próximos.

# Questão 4

Suponha que um filtro Box 3x3 (matriz no slide 105 da aula de filtragem) é usado para processar uma imagem. Em seguida, um filtro laplaciano 3x3 (slide 129 da aula de filtragem) é usado na imagem processada pelo filtro Box. Ou seja, temos:

I\_Final = Laplaciano (Box(I\_original))

Calcule como deve ser uma máscara única que faria o mesmo que os filtros

Laplaciano e Box. Apresente todos os cálculos.

Filtro box =

Filtro Laplaciano =

Para obter a máscara única que realiza as duas operações, deve-se multiplicar a máscara do filtro Laplaciano pelo filtro Box. Essa multiplicação deve ser realizada usando a convolução.

Filtro resultante (i, j) =

Demonstrando os cálculos:

= 0

= 2/3

= 2/3

# Questão 5

**Qual a consequência, para o contraste de uma imagem (em tons de cinza com tons entre 0 e 255), a aplicação de um filtro passa-baixa? Explique**

A aplicação de um filtro passa-baixa em uma imagem em tons de cinza, com valores de intensidade entre 0 e 255, resulta em uma suavização da imagem que reduz o contraste geral. Esse tipo de filtro é projetado para atenuar as altas frequências presentes na imagem, ou seja, ele reduz as variações rápidas de intensidade que são responsáveis por detalhes finos e bordas nítidas.

**Funcionamento do Filtro Passa-Baixa:** Filtros passa-baixa, como o filtro de média (Box) e o filtro gaussiano, funcionam aplicando uma média ponderada dos pixels vizinhos. Eles substituem cada pixel pelo valor médio ou próximo da média de seus vizinhos, de modo que áreas com grandes variações passam a ter valores de intensidade mais homogêneos.

Altas frequências, que representam mudanças bruscas entre pixels claros e escuros, são suavizadas, enquanto baixas frequências, que representam variações graduais, são mantidas. Como resultado, as bordas e detalhes nítidos tendem a desaparecer, pois suas transições rápidas são reduzidas.

**Exemplo numérico:**

Imagem =

Filtro box =

Resultado =

**Impacto no Contraste:** O contraste de uma imagem depende das diferenças entre os tons claros e escuros. Quando aplicamos um filtro passa-baixa, essas diferenças são reduzidas, pois as áreas de brilho intenso e sombra passam a ter valores de intensidade mais próximos. Isso gera uma imagem com menos definição entre regiões de diferentes intensidades, tornando as transições menos perceptíveis e diminuindo a percepção de contraste.

Portanto, ao aplicar um filtro passa-baixa em uma imagem, o resultado é uma suavização da imagem que leva a uma redução do contraste. Esse processo é útil quando queremos remover ruídos ou suavizar uma imagem, mas não é ideal quando desejamos preservar ou realçar detalhes finos, como bordas ou texturas, que são importantes para a nitidez da imagem.

# Questão 6

**Qual o resultado da aplicação dos filtros abaixo a uma imagem e cite uma aplicação para cada um:**

1. **Filtro Passa-Baixa**

O filtro passa-baixa suaviza a imagem ao reduzir as altas frequências, que são responsáveis por detalhes finos e bordas nítidas. Como resultado, a imagem torna-se mais suave, com menos ruído e uma aparência "lavada". As transições entre áreas claras e escuras se tornam mais gradativas, resultando em uma redução do contraste.

Uma aplicação comum do filtro passa-baixa é na redução de ruídos em imagens digitais, especialmente em fotos tiradas em condições de baixa luminosidade. Ele é amplamente utilizado em pré-processamento de imagens para preparar dados antes de outras operações, como a segmentação ou detecção de bordas.

1. **Filtro Passa-Alta**

O filtro passa-alta, por outro lado, destaca as altas frequências da imagem, realçando bordas e detalhes finos. Ao subtrair as componentes de baixa frequência (que correspondem a áreas mais suaves da imagem), o filtro produz uma imagem com maior nitidez e definição. O resultado é uma imagem com bordas mais marcadas e contrastes acentuados.

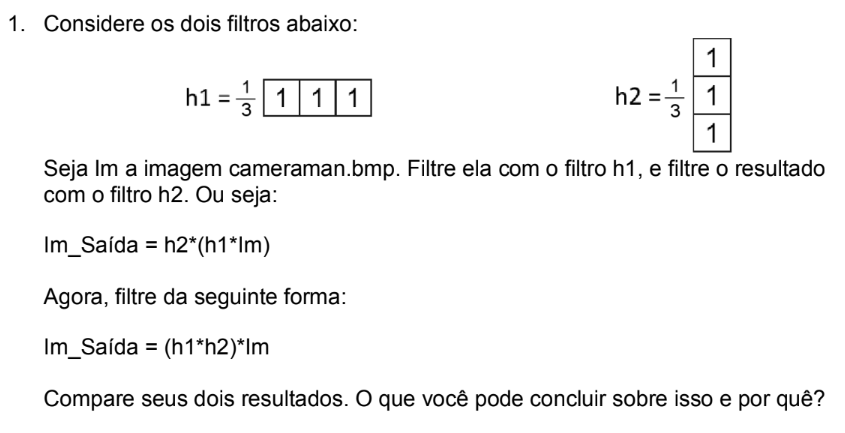
Um uso típico do filtro passa-alta é em técnicas de realce de bordas, que são essenciais em aplicações de visão computacional, como detecção de objetos e reconhecimento facial. Ao realçar bordas, o filtro ajuda a melhorar a precisão em algoritmos de reconhecimento e classificação.

1. **Filtro Passa-Faixa**

O filtro passa-faixa combina as propriedades dos filtros passa-baixa e passa-alta, permitindo a passagem de uma faixa específica de frequências. O resultado é uma imagem que mantém certas características de detalhes, enquanto remove componentes muito baixas ou muito altas. Isso pode resultar em uma imagem que destaca elementos em uma faixa de frequência específica, proporcionando uma representação equilibrada de detalhes.

Filtros passa-faixa são frequentemente usados em análise de texturas, onde é desejável focar em padrões específicos em uma imagem. Eles são úteis em processamento de imagens médicas, como a análise de imagens de ressonância magnética, onde diferentes faixas de frequências podem destacar características específicas de tecidos.

# Atividade Teórico-prática 1



Na primeira abordagem, a aplicação dos filtros é feita sequencialmente, primeiro h1 que é um filtro que atua suavizando a imagem horizontalmente e depois h2 que suaviza a imagem verticalmente. Nessa abordagem percebe-se que a imagem sofre um desfoque, porém é um desfoque mais “grosseiro”, menos suave.

Na segunda abordagem os filtros são combinados resultando em uma máscara bidimensional através de uma operação de convolução sendo:

h1 x h2 =

Essa operação é uma forma de suavização isotrópica gerando um desfoque que afeta a imagem igualmente em todas as direções (horizontal e vertical). Em termos computacionais, o filtro combinado economiza operações, pois evita o processamento sequencial ao realizar o efeito de ambos os filtros em uma única operação de convolução.

Foi criado um código python para visualizar melhor os resultados:

Foto em preto e branco de homem em pé

Descrição gerada automaticamente



# Atividade Prática 1

**Considere a imagem Tools.bmp. Através de técnicas de processamento de imagens, crie um algoritmo que conte, de forma automática, quantas ferramentas existem nessa imagem. Apesar da contagem de objetos em uma cena ser um problema típico de visão computacional, ele pode ser resolvido por processamento de imagens, dependendo das condições da cena.**

No código desenvolvido, a imagem **Tools.bmp** é primeiro carregada em escala de cinza para simplificar o processamento, uma vez que as informações de cor não são necessárias para a contagem de objetos.

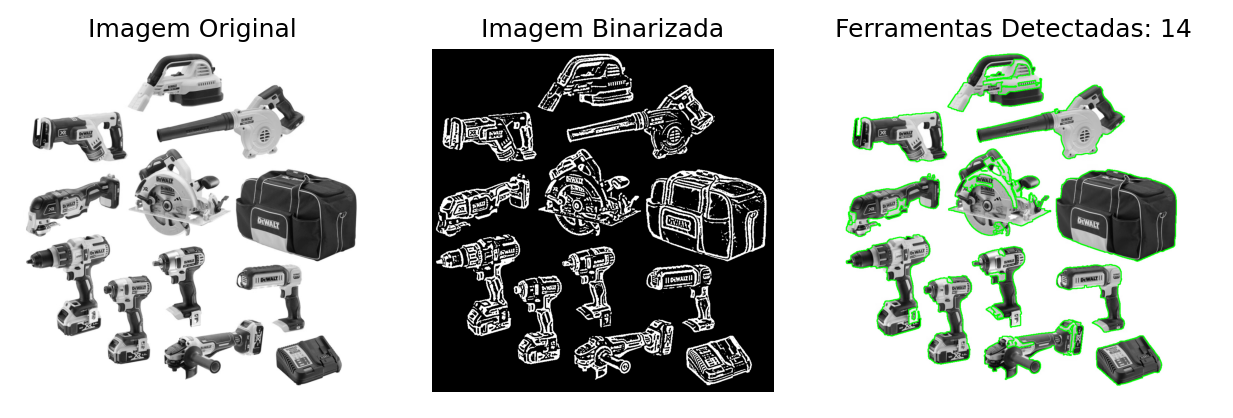
A seguir, é aplicada uma suavização por filtro Gaussiano, o que reduz ruídos na imagem. Essa etapa ajuda a suavizar as transições abruptas de pixel, evitando que pequenos detalhes ou imperfeições sejam erroneamente detectados como objetos independentes.

Em seguida, é aplicado um **limiar adaptativo** para transformar a imagem em uma versão binária (preto e branco). O limiar adaptativo é utilizado porque ele ajusta automaticamente o limiar em pequenas regiões da imagem, o que compensa variações de iluminação e melhora a separação das ferramentas do fundo. Como resultado, obtemos uma imagem binária onde as ferramentas aparecem em branco sobre um fundo preto.

Para tornar a segmentação mais robusta, é necessário aplicar uma operação morfológica de fechamento (dilatação seguida de erosão) que conecta partes próximas de uma mesma ferramenta e preenche pequenas lacunas. Essa operação resulta em contornos mais coesos, ajudando na contagem precisa dos objetos.

Com a imagem binária pronta, é possível detectar os contornos das áreas brancas (as ferramentas) na imagem. Cada contorno encontrado representa um objeto, e essa etapa permite identificar e isolar as ferramentas na cena.

Por fim, filtramos os contornos detectados com base em uma área mínima, removendo pequenas regiões que não correspondem a ferramentas reais. Isso reduz a chance de contarmos ruídos ou pequenos fragmentos como objetos separados. Depois de filtrados, os contornos restantes representam as ferramentas, e o código conta e exibe o número total detectado, além de destacar os contornos das ferramentas na imagem para uma verificação visual.



# Atividade Prática 2

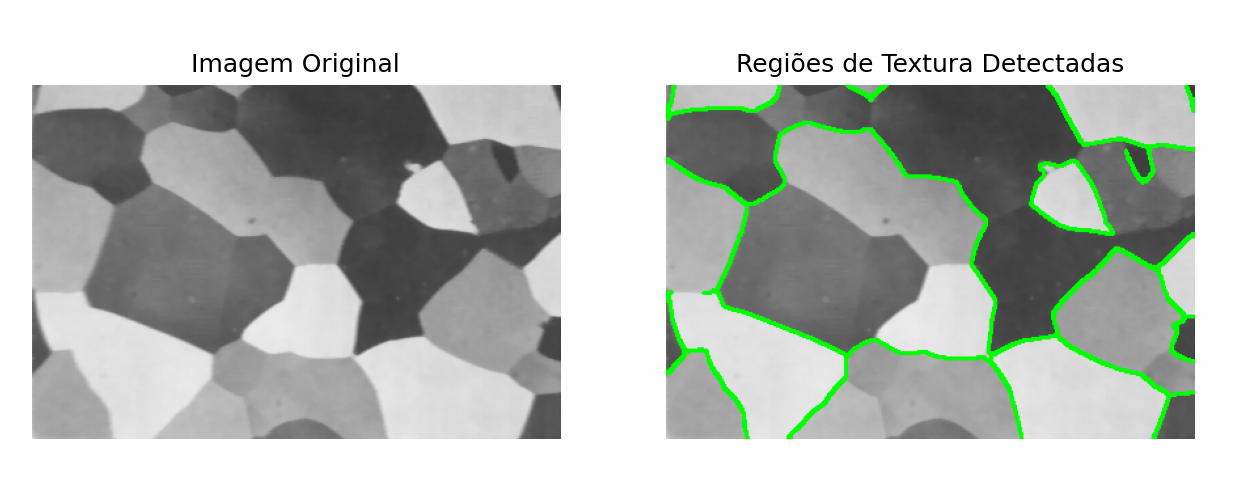
**Na imagem alumgrns.bmp, implemente um algoritmo para contar automaticamente, sem informação prévia do usuário, quantas regiões de diferentes texturas existem na imagem aproximadamente.**

Primeiro, a imagem alumgrns.bmp foi carregada e convertida para tons de cinza para simplificar o processamento, removendo informações de cor que poderiam introduzir ruído. Em seguida, aplicou-se um filtro Gaussiano para suavizar a imagem e reduzir interferências de pequenos ruídos, mantendo o foco nas principais texturas.

Com a imagem suavizada, foi utilizada a detecção de bordas de Canny, configurando limiares adequados (80 e 180) para identificar bem os contornos entre áreas de diferentes texturas. Isso permitiu que as regiões fossem destacadas sem segmentação excessiva.

A imagem de bordas foi então binarizada para facilitar a extração de contornos, e uma operação de fechamento morfológico foi aplicada para conectar áreas próximas e criar regiões mais coesas. Esse passo foi fundamental para garantir que regiões com a mesma textura não fossem separadas por pequenos espaços, evitando contagens incorretas.

Finalmente, os contornos das regiões de textura foram extraídos e contados. O número de contornos encontrados corresponde ao total de regiões de textura distinta na imagem. Para melhor interpretação visual, os contornos foram desenhados sobre a imagem original.



Texto

Descrição gerada automaticamente

# Atividade Prática 3 – A/B

O código tem como objetivo detectar a presença de um instrumento médico em um conjunto de imagens, utilizando uma técnica de template matching. O processo começa com a importação das bibliotecas necessárias: cv2, os e numpy. A biblioteca cv2 é utilizada para operações de processamento de imagens, enquanto os gerencia arquivos e diretórios, e numpy manipula arrays e operações numéricas.

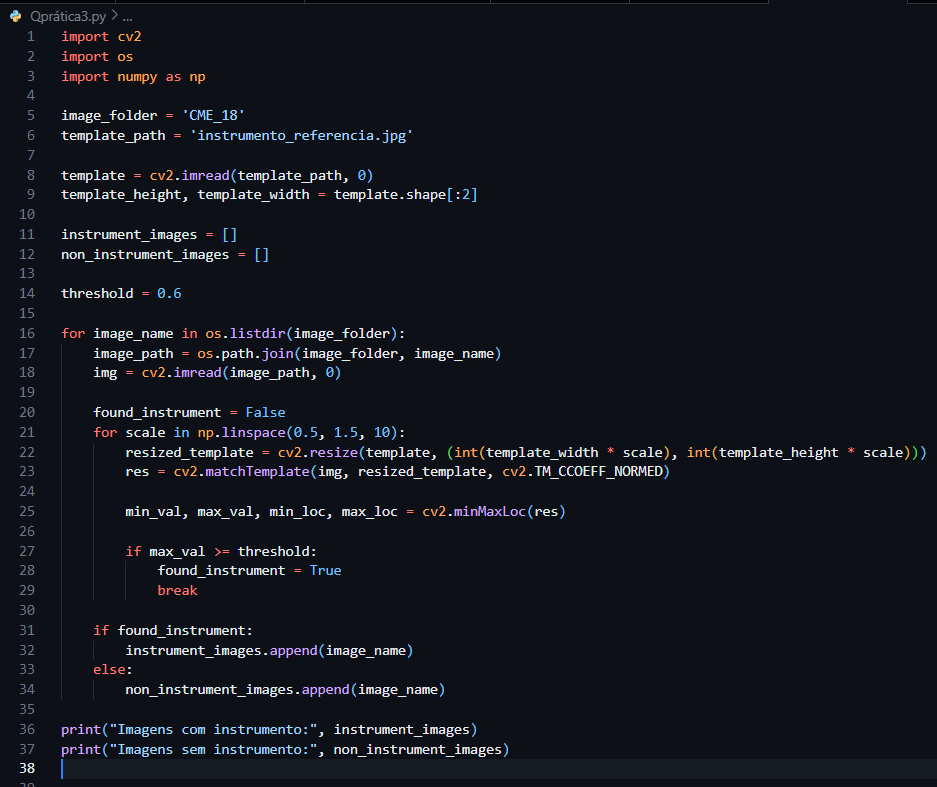
Inicialmente, define-se o caminho da pasta que contém as imagens a serem analisadas e a localização do template do instrumento, que serve como referência para a detecção. Em seguida, o template é carregado em escala de cinza, o que é crucial para simplificar a comparação entre as imagens, já que a detecção de padrões pode ser feita de forma mais eficiente em imagens monocromáticas.

O código então inicializa duas listas: uma para armazenar os nomes das imagens que contêm o instrumento e outra para aquelas que não contêm. Um limite de similaridade (threshold) é estabelecido para determinar se uma imagem é considerada como contendo o instrumento ou não. Este valor é ajustável e pode ser modificado conforme necessário.

O próximo passo envolve o processamento de cada imagem na pasta. Para cada imagem, o código é capaz de ler a imagem em escala de cinza e inicializa uma variável que indica se o instrumento foi encontrado. A detecção do template é realizada em várias escalas, variando de 50% a 150% do tamanho original do template. Essa abordagem em múltiplas escalas é importante para lidar com variações de tamanho do instrumento nas imagens, garantindo que a detecção não seja comprometida se o instrumento aparecer maior ou menor.

Dentro do loop de escalas, o template é redimensionado e a função cv2.matchTemplate é aplicada. Essa função compara o template com a imagem, produzindo uma matriz de resultados que reflete a similaridade em cada posição. O código então utiliza cv2.minMaxLoc para encontrar o valor máximo de correspondência e suas coordenadas correspondentes. Se o valor máximo encontrado for maior ou igual ao limite definido, isso indica que o instrumento foi detectado na imagem, e a variável de controle é atualizada para refletir essa condição.

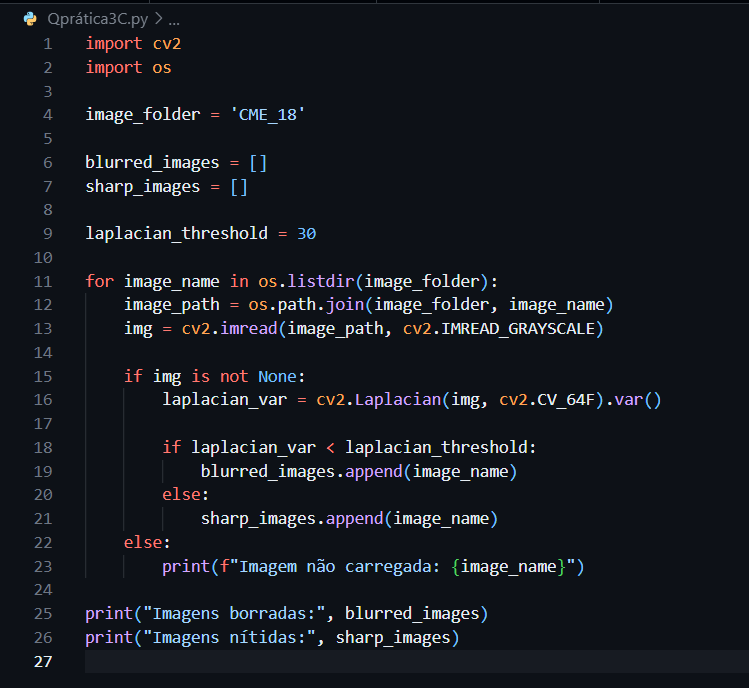
Após tentar todas as escalas para uma imagem específica, o código classifica a imagem com base na presença ou ausência do instrumento. As imagens são agrupadas nas listas de acordo com o resultado da detecção. Por fim, o código imprime as listas de imagens que contêm o instrumento e aquelas que não o contêm, fornecendo um resultado claro sobre a presença do objeto nas imagens analisadas.



# Atividade Prática 3 – C

O código foi desenvolvido para classificar imagens em uma pasta específica em duas categorias: borradas e nítidas. Utilizando a biblioteca OpenCV, ele inicia carregando as imagens em escala de cinza, o que simplifica o processamento ao eliminar informações de cor. Para avaliar a nitidez de cada imagem, o código aplica a técnica do Laplaciano, que é um método de filtragem utilizado para detectar bordas e áreas de alta frequência. A partir da imagem filtrada, é calculada a variância do Laplaciano, que serve como um indicador da qualidade da imagem.

Um limite ajustável para a variância é definido, onde imagens com variância abaixo desse limite são classificadas como borradas, enquanto aquelas acima são consideradas nítidas. Para organizar os resultados, duas listas são criadas: uma para armazenar os nomes das imagens borradas e outra para as nítidas. O código também inclui uma verificação para garantir que cada imagem foi carregada corretamente, notificando o usuário caso alguma imagem não possa ser lida. Dessa forma, o algoritmo fornece uma classificação clara das imagens, facilitando a identificação de problemas de foco.



# Atividade Prática 3 – D