Universidade Federal de Itajuba
Computação Gráfica e Processamento Digital de Imagens
Trabalho Final de PDI - Parte Teórica
Prof. Natasha Sayuri Dias Nakashima
Aluno: Mateus Eusébio de Lacerda - 31437

1

O processo de filtragem de uma imagem $f(x, y)$ no domínio da frequência consiste em:

I. Calcular a sua transformada de Fourier $F(u, v)$;

II. Multiplicar $F(u, v)$ por um filtro $H(u, v)$;

III. E calcular a transformada inversa do resultado encontrado, de forma a visualizar a modificação realizada.

Sobre o processo de Filtragem no Domínio da Frequência, responda cada uma das questões a seguir e justifique com implementação de código e resultados reais.

a) O que acontece com a imagem resultante $f'(x, y)$ após uma filtragem pelo filtro $H(u, v) = a$ onde a é uma constante real positiva?

b) O que acontece quando o filtro aplicado é truncado? Este processo gera algum artefato na imagem final processada?

c) O resultado da filtragem de uma imagem no domínio da frequência é alterado, caso a imagem $f(x, y)$ de entrada seja rotacionada?

d) Considere o experimento de aplicar repetidamente o filtro gaussiano passa-baixa $H(u, v) = e^{\frac{-D^2(u,v)}{2D_0^2}}$, para um dado valor fixo $D - 0$. Ignorando os erros de arredondamento:

   i. o que acontece se as aplicações são repetidas um número de vezes suficientemente grande K, antes de calcular a etapa (III)? Em termos de resultado, que tipo de imagem você espera obter?

   ii. E para o caso em que você aplica o filtro considerando todo o processo (a-c), um número de vezes suficientemente grande? Em termos de resultado, que tipo de imagem você espera obter?

2

Escolha entre o processamento para suavização ou realce de bordas, e explique e exemplifique um Filtro Linear e Não Linear da categoria escolhida. A exemplificação deve ser feita utilizando uma imagem em tons de cinza.

R: O processo de suavização é utilizado para reduzir o ruído de alta frequência da imagem. A Gaussiana é um importante filtro Linear de suavização que mantém pesos maiores para os píxeis mais próximos do píxel que será suavizado, enquanto que a Mediana é um filtro não-linear que ordena os píxeis vizinhos baseado em suas intensidades e substitui pelo valor mediano, ela é importante porque desconsidera píxeis "fora-da-curva"numa vizinhança, desde que eles não sejam a maioria.

I. Gaussiana: Dada uma imagem em cinza representada por uma matriz 2d, a gaussiana, recalcula a intensidade de um píxel baseada na intensidade da vizinhança, tendo uma importante característica, os píxeis mais próximos tem peso maior. Na Figura 1, podemos ter uma idéia do comportamento da função gaussiana.

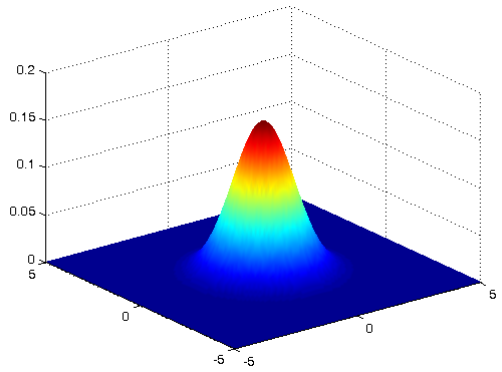


Figura 1: Função Gaussiana em 2d

Para aplicar a gaussiana numa imagem é necessário gerar um filtro $h(n, \sigma)$, onde n é o tamanho da matriz (ímpar) e σ representa a variação no peso dos píxeis do centro para os das bordas. Na Equação 1, é gerada uma matriz para um filtro gaussiano $h(n, \sigma)$ com $n = 3$ e $\sigma = 0.8$

$$h = \begin{bmatrix} 0.0571 & 0.1248 & 0.0571 \\ 0.1248 & 0.2725 & 0.1248 \\ 0.0571 & 0.1248 & 0.0571 \end{bmatrix} \tag{1}$$

Na Figura 2a temos a imagem original com ruído, enquanto que na Figura 2b temos a imagem após o filtro da gaussiana de equação 1

II. Mediana: Dada uma vizinhança do píxel a ser filtrado, a mediana ordena os píxeis vizinhos crescentemente por intensidade e substitui o píxel filtrado pelo valor do meio do vetor ordenado.

Por exemplo, dada uma matriz de vizinhança M da equação 2 que representa os píxeis a serem filtrados, a mediana primeiro realiza a ordenação dessa vizinhança e coloca num vetor V da equação 3

(a) Imagem com Ruído (b) Imagem Filtrada

Figura 2: Aplicação do Filtro Gaussiano

$$M = \begin{bmatrix} 210 & 198 & 212 \\ 87 & 235 & 240 \\ 223 & 198 & 210 \end{bmatrix} \tag{2}$$

$$V = \begin{bmatrix} 87 & 198 & 198 & 210 & 210 & 212 & 223 & 235 & 240 \end{bmatrix} \tag{3}$$

O último passo é achar o elemento do meio do vetor, que nesse caso é o 210, assim o novo valor desse píxel será 210.

Na Figura 3a temos novamente a imagem original e na figura 3b temos o resultado ao aplicar um filtro mediana de 3x3.

(a) Imagem com Ruído (b) Imagem Filtrada

Figura 3: Aplicação da Mediana

Por fim, temos o código 1, feito em Matlab, usado para filtrar as imagens usando a gaussiana e a mediana.

```
clear;
clc;
close all;

imagem = double(imread('imagem exemplo.png'));
imagem=imagem(:,:,1);
figure
imshow(uint8(imagem));

%******* GAUSSIANA *******
h = fspecial('gaussian',[3 3], 0.8)
gaussiana= imfilter(imagem,h,'circular');
figure
imshow(uint8(gaussiana));
title('Aplicacao da Gaussiana 3x3 , 0.8');

%******* MEDIANA ********
Mediana = medfilt2(imagem,[3 3]);
figure
imshow(uint8(Mediana));
title('Aplicacao da Mediana 3x3');
```

Código 1: Aplicação da Gaussiana e Mediana na Imagem

3

Desenvolva um programa que, dada a imagem colorida ($peppers\_color.tif$)**, realize um processamento na imagem de forma gerar uma imagem onde apenas os pimentões vermelhos possuem componentes de cor.**

R: Em cálculo, há casos em que coordenadas polares são mais fáceis de se trabalhar do que coordenadas cartesianas e vice-versa. Em processamento Digital de Imagens temos diferentes formas de representar uma cor, entre elas a RGB e HSV. Em que cada forma tem uma vantagem em um caso específico.

Em RGB as componentes Vermelha, Verde e Azul são somadas para formar as cores. Níveis iguais dessas componentes formam uma cor acinzentada, níveis altos formam o branco e níveis baixos formam o preto.

Em HSV, temos a componente Hue, que representa a cor propriamente dita, enquanto a Saturação representa a escala de cinza da cor e o Brilho representa o quão branco ou preta é a cor.

Quando é necessário separar cores da imagem através de uma segmentação é muito mais fácil usar o RGB. Porém quando é necessário aplicar uma escala de cinza, esbranquiçar ou escurecer uma imagem é muito mais fácil usar o HSV.

No código usado para separar as componentes do pimentão, foram usadas as componentes rgb da imagem para classificar o pimentão quanto à cor e, posteriormente, para colocar parte da imagem em cinza, foi usado o HSV, especificamente atribuindo 0 ao atributo saturação. As conversões foram feitas usando as funções rgb2hsv() e hsv2rgb() do Matlab.

Na Figura 4a pode-se observar a imagem original, e na Figura 4b, os píxeis do espaço RGB em que a componente vermelha é algumas unidades maior que a verde e maior que a azul foram mantidas do mesmo jeito, enquanto que, para os outros peixeis, a saturação foi atribuída a 0.

(a) Imagem Original (b) Separação do Vermelho

Figura 4: Separação do Vermelho

Como uma atividade complementar, foram separadas as cores amarela e verde da imagem, de modo que, dessa vez, foi atribuído 0 de saturação para o vermelho e as outras cores, conforme pode ser observado nas figuras 5a e 5b.

Em ambas as separações, os pontos muito escuros foram separados em cinza usando HSV, isso pode ser observado na linha 34 do código 2. Importante deixar claro que a

imagem original do material de aula (*pepers_color.tif*) não abriu no Matlab, portanto foi utilizada outra imagem idêntica e com mesma resolução no formato ".jpg"retirada da internet que funcionou no Matlab.

(a) Imagem Original (b) Separação do Verde e Amarelo

Figura 5: Separação do Verde e Amarelo

```
clear;
clc;
close all;

imagem_rgb = double(imread("peppers_color.jpg"));
figure
imshow(uint8(imagem_rgb));


[x,y,~]    = size(imagem_rgb);  %x=comprimento, y=largura
red      = imagem_rgb(:,:,1);   %separa a componente vermelha da imagem
green    = imagem_rgb(:,:,2);   %separa a componente verde de imagem
blue     = imagem_rgb(:,:,3);   %separa a componente azul da imagem

imagem_hsv = rgb2hsv(imagem_rgb);
hue        = imagem_hsv(:,:,1); %separa o hue
saturation = imagem_hsv(:,:,2); %separa a saturacao
luminance  = imagem_hsv(:,:,3); %separa a luminancia

saturation_red   = saturation;
saturation_green = saturation;

for i=1:x  % a variavel i percorre o comprimento da imagem
   for j=1:y %a variavel j percorre a largura da imagem
        %se o pixel nao e vermelho, o torna cinza (sem cor)
        if((red(i,j)<green(i,j)+30) || (red(i,j)<blue(i,j)))
                    saturation_red(i,j)=0; %torna cinza
        end
        %se o pixel nao e verde/amarelo, o torna cinza
        if((green(i,j)<red(i,j)-30) || (green(i,j)<blue(i,j)))
                    saturation_green(i,j)=0; %torna conza
        end
        %se o brilho do pixel e muito baixo, tambem o torna cinza
        if(luminance(i,j)<100)
                  saturation_red (i,j)=0;
```

```
                    saturation_green(i,j)=0;
        end
    end
end
%recupera a saturacao da imagem vermelha e imprime
imagem_hsv(:,:,2)=saturation_red;
imagem_final= hsv2rgb(imagem_hsv);
figure
imshow(uint8(imagem_final));

%recupera a saturacao da imagem verde/amarela e imprime
imagem_hsv(:,:,2)=saturation_green;
imagem_final= hsv2rgb(imagem_hsv);
figure
imshow(uint8(imagem_final));
```

Código 2: Separação de Cores

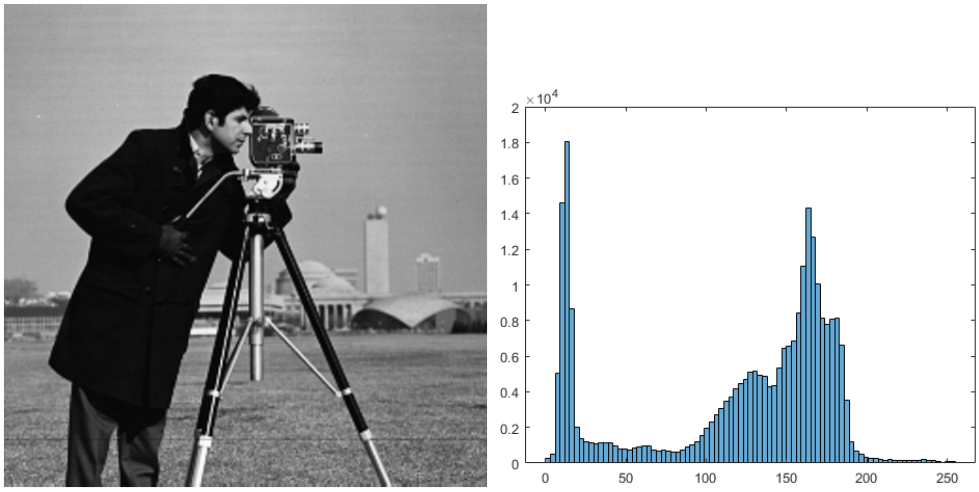
4

Explique qual a importância da avaliação do Histograma para o processamento de imagens e o que é a equalização de histograma.

R: Quando precisamos realizar uma transformação radiométrica na imagem, seja um threshold, tornar os píxeis mais claros, um aumento de contraste ou alguma operação que leva em consideração a intensidade dos píxeis, a intensidade dos píxeis de saida, será a intensidade dos píxeis de entrada transformada por alguma função para todos píxeis da imagem.

Podemos então usar o histograma como uma função de densidade-probabilidade, onde, atraves do grafico podemos ter uma ideia da intensidade media dos píxeis, pois o histograma revela a quantidade de píxeis que existem em cada faixa de intensidade de 0 a 255.

Por exemplo, dada uma imagem da Figura 6a, pode-se usar o histograma para entender melhor a distribuição da intensidade dos seus píxeis. E, na Figura 6b, temos o histograma da imagem. É possível perceber que há poucos píxeis com a intensidade superior a 200, e, ao olhar a imagem, realmente há pouco branco. Dessa forma, através da análise do histograma, observa-se que é possível melhorar a imagem aumentando a intensidade dos píxeis(multiplicando por um escalar), de modo a aproveitar o espaço de intensidade entre 200 e 255.E também há muitos píxeis na faixa de intensidade de 10 a 20, a imagem pode melhorar com a distribuição desses píxeis



(a) Imagem do Fotógrafo (b) Histograma do Fotógrafo

Figura 6: Imagem e Histograma

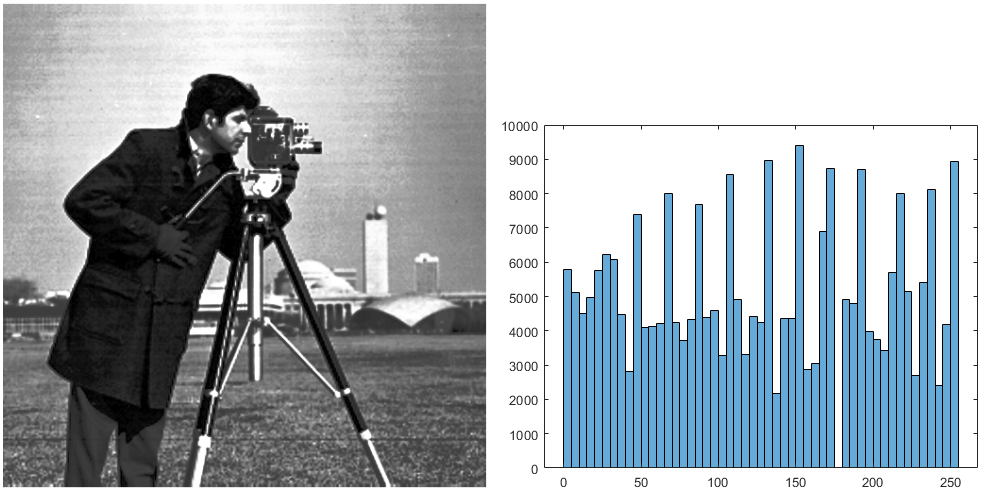
A equalização do histograma é usada para tentar equalizar uma imagem que esteja muito escura ou muito clara, procurando distribuir uniformemente a intensidade dos píxeis pela imagem, de modo a usar todas as intensidades.

No caso da imagem do fotógrafo, é possível perceber que há poucos píxeis com intensidade acima de 200, portanto uma equalização do histograma daquela imagem deve resultar em uma imagem mais clara.

Na Figura 7a temos a imagem do fotógrafo depois da equalização, é possível perceber que os detalhes dos bolsos casaco, que na Figura 6a era puramente preto, agora aparecem, isso se deve à distribuição dos píxeis com intensidade de 10 a 20. Porém o céu e a câmera ficaram claros demais, e nesse ponto a equalização do histograma não foi boa para aquele

espaço vago entre 200 e 255. Portanto cabe interpretar, para cada imagem se é válida ou não a aplicação da equalização do histograma.

Na Figura 7b é possível perceber, que no histograma equalizado a distribuição da intensidade dos píxeis fica muito mais uniforme do que no caso da Figura 6b.



(a) Imagem do Fotógrafo Equalizada (b) Histograma Equalizado

Figura 7: Imagem e Histograma Equalizados

A geração e equalização do histograma foi feita usando o *Matlab* e suas funções *histogram()* e *histeq()*, através do Código 3.

```
clear;
clc;
close all;

%********* IMAGEM *********
imagem = imread('cameraman.tif');
figure
imshow(imagem);

%******* HISTOGRAMA *******
figure
histogram(imagem);

%****** EQUALIZACAO *******
img_eq = histeq(imagem);
figure
imshow(img_eq);

%* HISTOGRAMA EQUALIZADO **
figure
histogram(img_eq);
```

Código 3: Histograma e Equalização

5

A segmentação de imagens é uma das partes essenciais na área de processamento de imagens. Avalie cada uma das questões a seguir, justificando sua resposta.

a) **A detecção de bordas é a determinação dos limites de um objeto em uma imagem, envolvendo a avaliação da variação nos níveis de cinza dos píxeis em uma vizinhança, sendo uma das formas de segmentação de imagens.**

R: A detecção de bordas ajuda **sim** a determinar limites de um objeto, existem por exemplo o filtro laplaciano e o de sobel que funcionam baseados na diferença de intensidade de um píxel com os píxeis vizinhos. **Porém**, a detecção de bordas **não** é uma forma de segmentação. A detecção de bordas pode ser usada antes da segmentação para ajudar e definir os limites do objeto a ser segmentado. Por exemplo, pode-se aplicar um threshold no resultado da detecção de bordas para obter a forma de um objeto. Nesse caso o threshold que é a segmentação.

b) **Filtros Gaussianos são utilizados para detecção de bordas, todos embasados em gradientes calculados sobre os níveis de cinza de uma imagem.**

R: O filtro gaussiano **não** é um filtro de detecção de bordas, ele é um filtro de suavização. Ele calcula o novo valor do píxel baseado na intensidade dos píxeis vizinhos, com um peso maior nos píxeis mais próximos. Ao contrário de um filtro de detecção de bordas, que ajuda a dar ênfase nas diferenças, ele ajuda a diminuir as diferenças "borrando"um pouco a imagem, e é muito bom para tirar ruído.

c) **A detecção de bordas é um processo de segmentação de imagens com princípio iguais às técnicas que agrupam píxeis vizinhos que compartilham determinado atributo.**

R: A detecção de bordas **não** é segmentação e **não** usa princípios iguais às técnicas que agrupam píxeis vizinhos com determinado atributo. A detecção de bordas funciona baseada na diferença de píxeis vizinhos, enquanto que nas técnicas de agrupamento os píxeis são agrupados baseados em propriedades similares como nível de cinza, textura, cor, etc.

d) **Crescimento de regiões é um método de detecção de bordas embasado em gradientes, usando como critério a disparidade de valores entre píxeis vizinhos.**

R: O crescimento de regiões **não** é um método de detecção de bordas, mas de segmentação. Nele, Um píxel é comparado com sua adjacência, se há muita disparidade entre os píxeis comparados eles são considerados de regiões diferentes, então ele **usa** como critério a disparidade entre píxeis vizinhos para fazer uma segmentação de determinada região, não para detectar bordas.

e) **O operador LoG (Laplacian of Gaussian) é empregado para detecção de bordas, usando os cruzamentos de zero na determinação dos píxeis que formam o limiar entre um objeto e outro em uma imagem.**

R: Como os filtros de realce de bordas geralmente realçam o ruído junto, geralmente é necessário utilizar um filtro de suavização para suavizar o ruído antes do filtro de realce de bordas. O laplaciano da gaussiana é nada mais que um filtro '2 em 1', primeiro ele realiza a gaussiana para suavizar, e com o resultado da gaussiana é aplicado o filtro laplaciano para realçar as diferenças na intensidade dos píxeis vizinhos. O LoG **não** usa cruzamentos de zero na determinação dos píxeis que formam limiares entre objetos. Ele é simplesmente um filtro gaussiano seguido por um filtro laplaciano.

6

Explique como a Morfologia matemática é utilizada no processamento de imagens. Explique também as duas principais operações Erosão e Dilatação e como elas podem ser combinadas para obtenção de melhores resultados.

R: A morfologia matemática foi inicialmente aplicada em imagens binárias de objetos. Ela pode ser utilizada para eliminar ruídos, mas principalmente para alterar a forma de objetos. Seu funcionamento consiste na comparação de objetos(inicialmente em binário) com um elemento estruturante, e através das operações de interseção ou diferença de conjuntos realiza a dilatação ou erosão do objeto.

Por exemplo, na Figura 8 temos um exemplo de dilatação onde temos um objeto binário em azul e um elemento estruturante em forma de cruz. O centro do elemento estruturante percorrerá todos os píxeis do objeto, e aqueles píxeis do objeto serão somados a todos os píxeis que o elemento estruturante atingiu enquanto percorria o objeto.

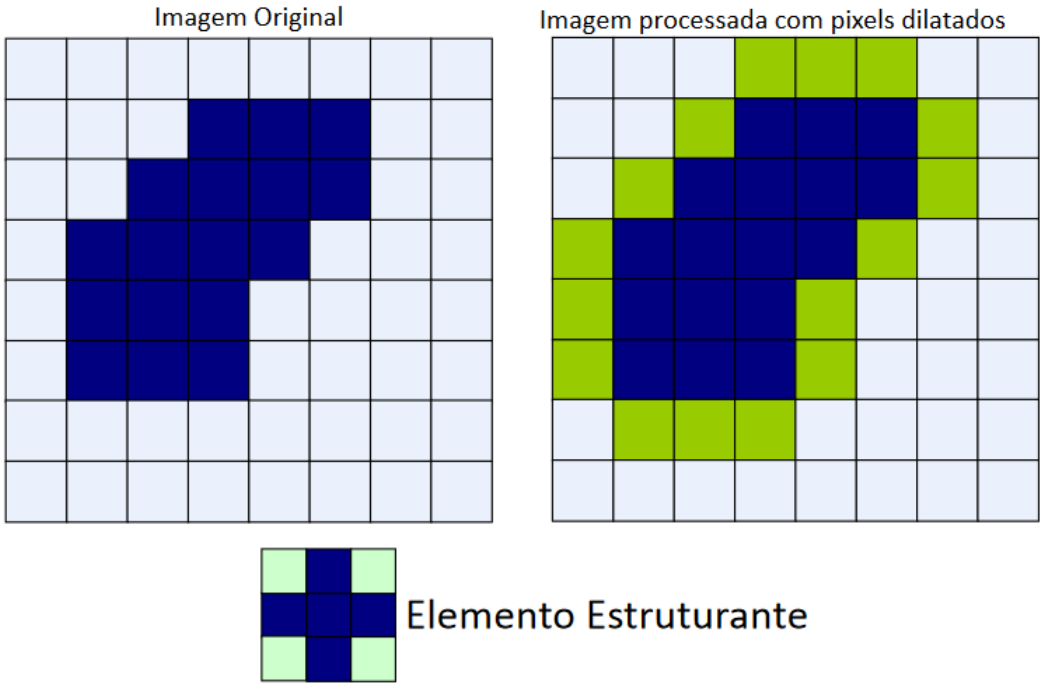


Figura 8: Exemplo de Dilatação

A erosão é a operação inversa, Dessa vez centro do elemento estruturante percorre o exterior do objeto e todos os píxeis do objeto atingidos pelo elemento estruturante durante esse percurso serão subtraídos do objeto, como podemo observar na figura 9.

A dilatação e Erosão podem ser combinadas numa ordem específica para realizar outras operações morfológicas, como a Abertura e o Fechamento. A abertura consiste em realizar uma erosão na imagem e depois uma dilatação, geralmente é usada para evidenciar os "buracos"da imagem. Enquanto que o fechamento é uma dilatação seguida de uma erosão, que geralmente é utilizado para "fechar"os buracos da imagem.

A abertura também é usada para remover pequenas conexões entre objetos, enquanto que o fechamento é usado para remover buracos. 10a é mostrado um exemplo de uma Figura original, em 10b é mostrado essa figura depois da abertura, importante notar que a conexão entre os dois objetos somem, porém o buraco no meio do objeto da direita não

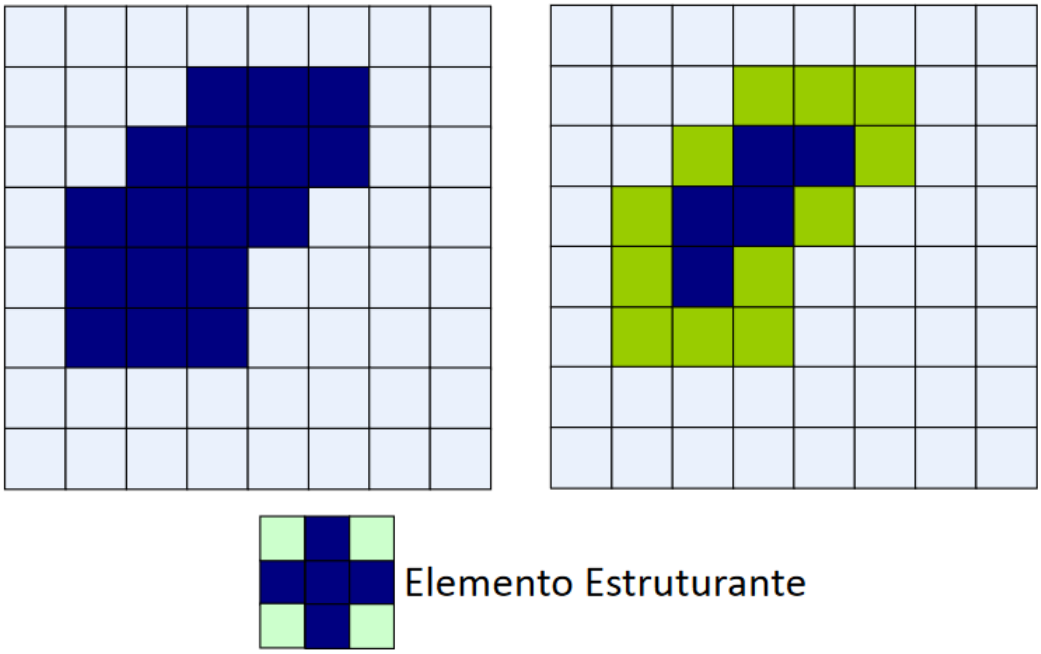


Figura 9: Exemplo de Erosão

some. E em 10c pode ser observado o fechamento, onde o buraco é removido.

(a) Original (b) Abertura (c) Fechamento

Figura 10: Abertura e Fechamento da Imagem Binária

Por fim, existem prós e contras ao escolher a operação de abertura ou fechamento num objeto. Além dessas, há várias outras maneiras de fazer operações morfológicas num objeto, há várias sequencias de aberturas e fechamentos que se feitas na ordem certa ajudam a dar ênfase nas características importantes de um objeto e descartar características não-importantes. Esses tipos de filtros com várias aberturas e fechamentos, são chamados de Filtros Alternados Sequenciais.