

Processamento de Imagens

Prof. MSc. Daniel Menin Tortelli

e-mail: danielmenintortelli@gmail.com

Skype: daniel.menin.tortelli

Site: <http://sites.google.com/site/danielmenintortelli/home>

Formação de uma imagem

Formação de uma imagem

- Todas as imagens digitais têm alguma origem.
- A origem e as características de uma imagem podem ter grande influência sobre o modo em que a processamos.
- Em geral, uma imagem digital pode ser formalizada como um modelo matemático que consiste em:
 - Uma representação funcional da cena (função do objeto o)
 - Um processo de captura (função de espalhamento de ponto [**PSF – Point-Spread Function**]).
 - Um ruído aditivo n .

$$\text{Imagem} = \text{PSF} * \text{função do objeto} + \text{ruído}$$
$$s = p * o + n$$

Formação de uma imagem

Os elementos essenciais neste processo são:

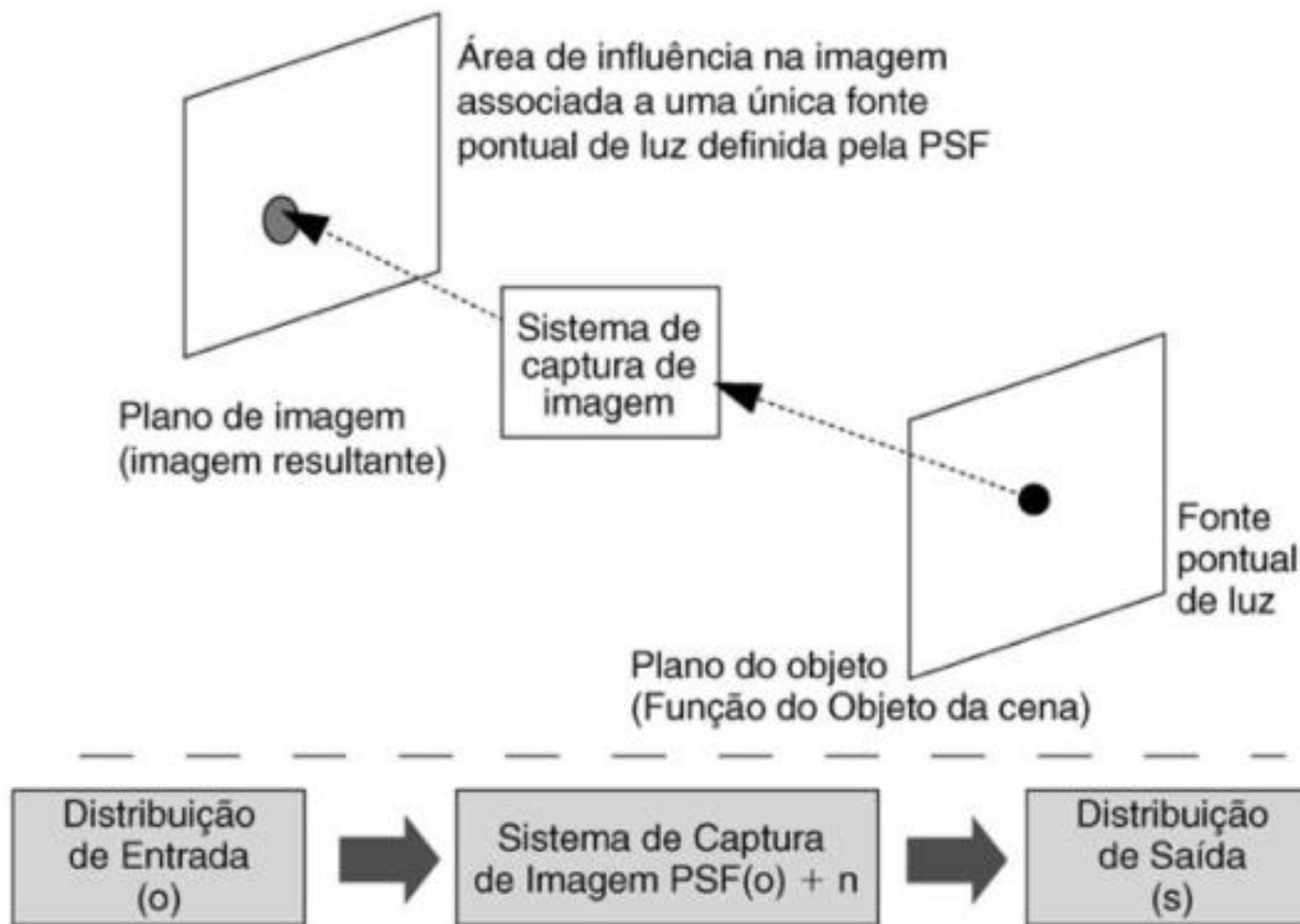
- **PSF** – Esta função descreve a forma em que a informação sobre a função do objeto se espalha em consequência da captura dos dados. Esta é uma característica do instrumento que captura a imagem (por exemplo, a câmera) e é uma função determinística (que opera na presença de ruído).
- **Função do objeto** – Esta função descreve o objeto (ou cena) cuja imagem é capturada (por exemplo, a superfície ou estrutura interna do objeto) e a forma em que a luz é refletida por esta estrutura em direção ao instrumento de captura de imagens.

Formação de uma imagem

Os elementos essenciais neste processo são:

- **Ruído** – Esta é uma função não determinística que, na melhor das hipóteses, pode ser descrita somente em termos de alguma distribuição estatística de ruído (por exemplo, gaussiana). Ruído é uma função estocástica e resulta de todos os distúrbios externos indesejados que ocorrem durante a captura da imagem.
- **Operador Convolução ($*$)** – Operação matemática que “traspassa” (efetua a convolução de) uma função com outra.

Formação de uma imagem



Visão geral do 'sistema' de formação de uma imagem e o efeito da PSF.

Formação de uma imagem

- Aqui, a função da luz refletida pelo objeto/cena (função do objeto) é transformada nos dados de representação da imagem através da convolução com a PSF.
- A PSF é uma característica do instrumento de captura de imagem (ou câmera).
- Esta função caracteriza o processo de formação (ou captura) da imagem.
- O processo é afetado por ruído.

Aquisição de Imagens

Aquisição de Imagens

- Considerando que o dispositivo de captura de imagens seja uma câmera...
- A imagem de uma cena captada pela câmera é essencialmente uma projeção do mundo 3D (ou seja, a cena) em uma representação 2D (a imagem).
- Se assumirmos que um objeto ou cena é iluminado por uma fonte de luz (possivelmente, múltiplas fontes), alguma parcela da luz será refletida em direção à câmera e capturada como uma imagem digital.

Aquisição de Imagens

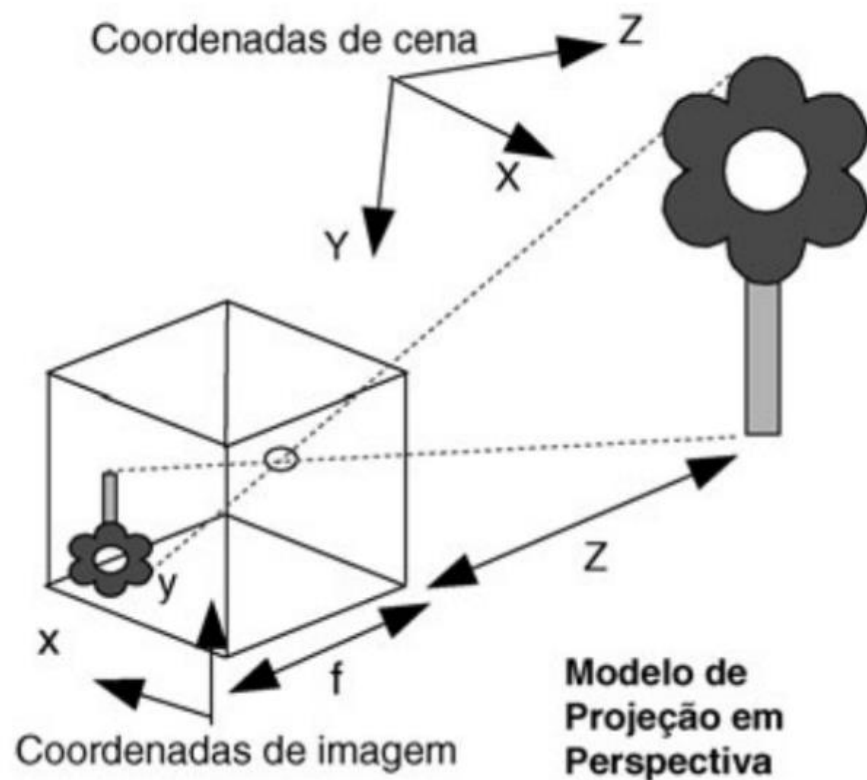
- Assumindo que tenhamos uma câmera convencional, a luz refletida que incide na lente forma uma imagem no plano de imagem da câmera, com base no modelo de projeção da câmera.
- O modelo de projeção da câmera transforma coordenadas do mundo em 3D (X, Y, Z) em coordenadas de imagem 2D (x, y) no plano de imagem.
- A maioria das imagens de cenas com que tratamos é capturada com o uso de uma **projeção em perspectiva**.

Projeção em Perspectiva

- Em geral, a projeção em perspectiva pode ser enunciada como:

$$x = f \frac{X}{Z} \quad y = f \frac{Y}{Z}$$

- Uma posição (X, Y, Z) na cena (profundidade Z) é reproduzida na posição (x, y) no plano de imagem determinado pela distância focal f da lente da câmera (distância entre a lente e o plano de imagem).



Projeção em Perspectiva

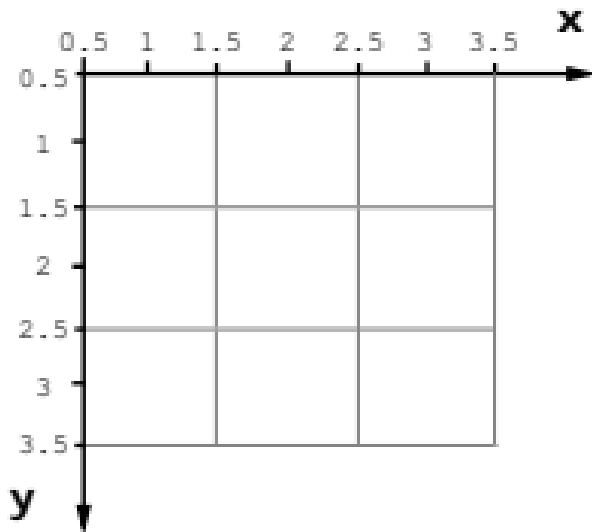
- A projeção em perspectiva tem as seguintes propriedades:
 - **Escorço (*foreshortening*):** Os tamanhos de objetos na imagem capturada dependem das distancias entre os mesmos e o visor. Assim, objetos mais afastados do visor parecerão menores na imagem.
 - **Convergência Retas:** que são paralelas na cena parecerão convergir na imagem resultante. Isto é conhecido como distorção de perspectiva; o(s) ponto(s) em que as retas paralelas parecem se encontrar recebe(m) a denominação de ponto(s) de fuga da cena.

Efeitos do modelo de projeção em perspectiva sobre a geometria da cena:

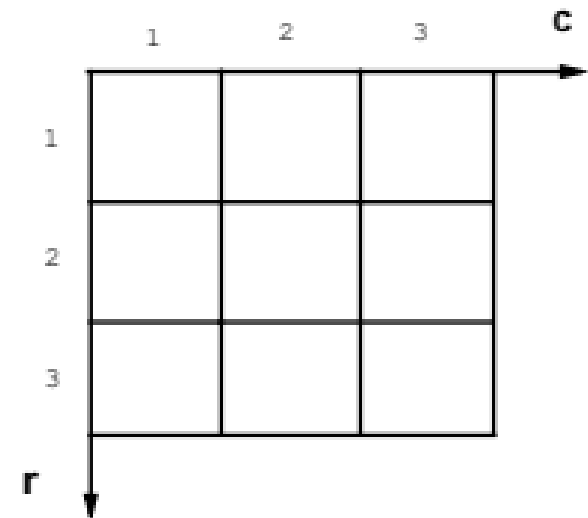


Sistema de Coordenadas da Imagem

- A localização em uma imagem pode ser expressada em vários sistemas de coordenadas.
- A quantização espacial da projeção do plano de imagem em uma grade discretizada de pixel transforma as coordenadas de imagem 2D (x, y) no plano de imagem em uma posição de pixel (c, r) .



Coordenadas Espaciais



Coordenadas de Pixel

Aquisição de Imagens

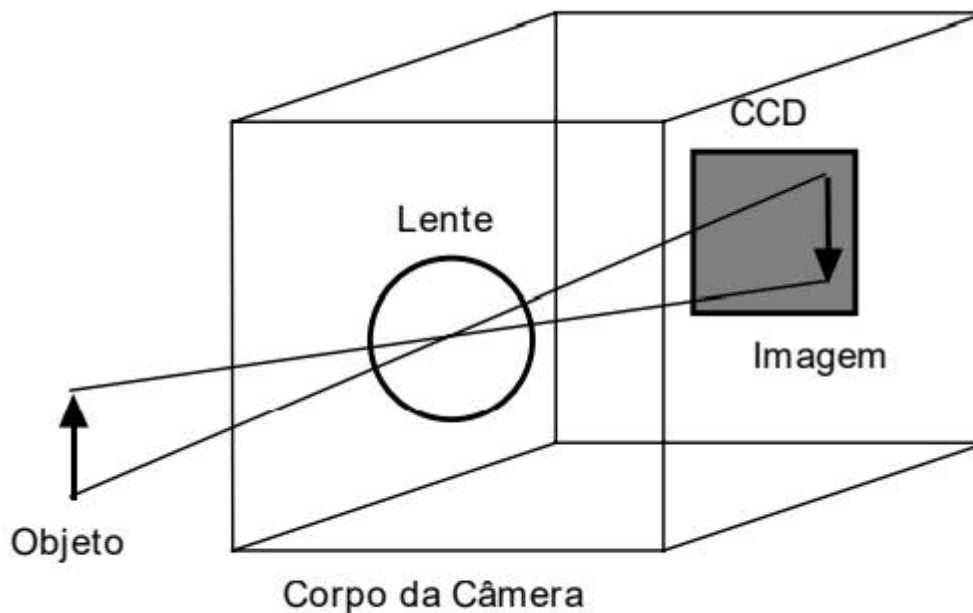
- Um dispositivo de aquisição de imagens é a câmera **CCD (*Charge Coupled Device*)**.
- Ela consiste de uma matriz de células semicondutoras fotossensíveis, que atuam como capacitores, armazenando carga elétrica proporcional à energia luminosa incidente.
- O sinal elétrico produzido é condicionado por circuitos eletrônicos especializados, produzindo à saída um **Sinal Composto de Vídeo (SCV)** analógico e monocromático.

Aquisição de Imagens

- Para a aquisição de imagens coloridas utilizando CCDs é necessário um conjunto de prismas e filtros de cor encarregados de decompor a imagem colorida em suas componentes R, G e B, cada qual capturada por um CCD independente.
- Os sinais elétricos correspondentes a cada componente são combinados posteriormente conforme o padrão de cor utilizado **NTSC** (***National Television Standards Committee***) ou **PAL** (***Phase Alternating Line***), por exemplo).

Aquisição de Imagens

- Uma câmera CCD monocromática simples consiste basicamente de um conjunto de lentes que focalizarão a imagem sobre a área fotossensível do CCD, o sensor CCD e seus circuitos complementares.



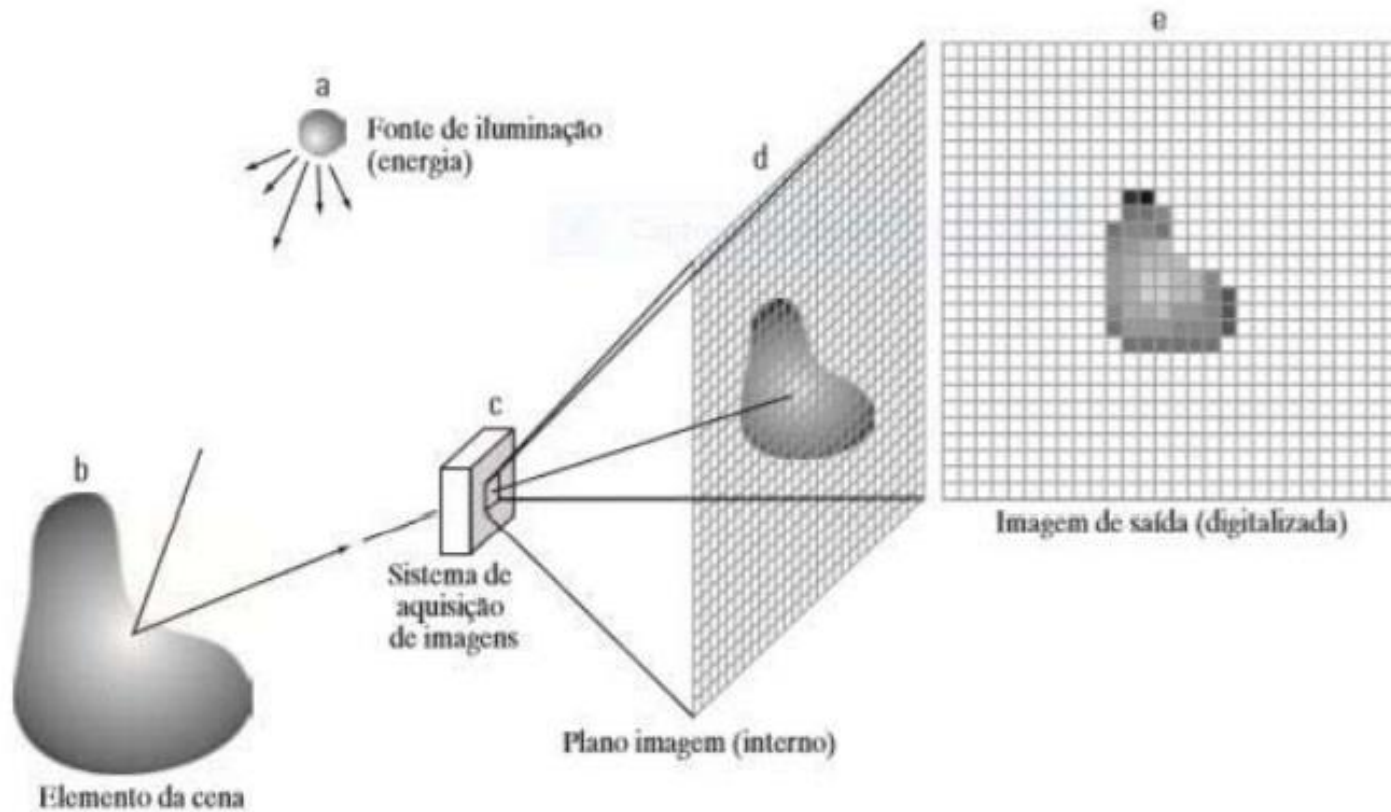
Visão esquemática de uma câmera CCD.

Aquisição de imagens utilizando sensores matriciais

- Existem inúmeros dispositivos com sensores eletromagnéticos e alguns ultrassônicos que são utilizados de forma matricial.
- Essa organização também é encontrada nas câmeras digitais.
- *A resposta de cada sensor é proporcional à integral da energia luminosa projetada sobre sua superfície. (GONZALES; WOODS, 2010).*

Aquisição de imagens utilizando sensores matriciais

Processo de aquisição de uma imagem digital: (a) Fonte de energia (“iluminação”). (b) Elemento de uma cena. (c) Sistema de aquisição de imagem. (d) Projeção da cena no plano imagem. (e) Imagem digitalizada.



Fonte: Gonzales e Woods (2010, p. 32)

Aquisição de imagens utilizando sensores matriciais

- A energia de uma fonte de iluminação é refletida de um elemento da cena.
- A primeira atribuição realizada pelo sistema de aquisição de imagens é coletar a energia de entrada e projetá-la em um plano.
- Considerando a fonte de iluminação, a entrada frontal do sistema de aquisição de imagens é constituída por uma lente ótica que projeta a cena vista sobre o plano focal da lente e produz saídas integrais da luz recebida em cada sensor.

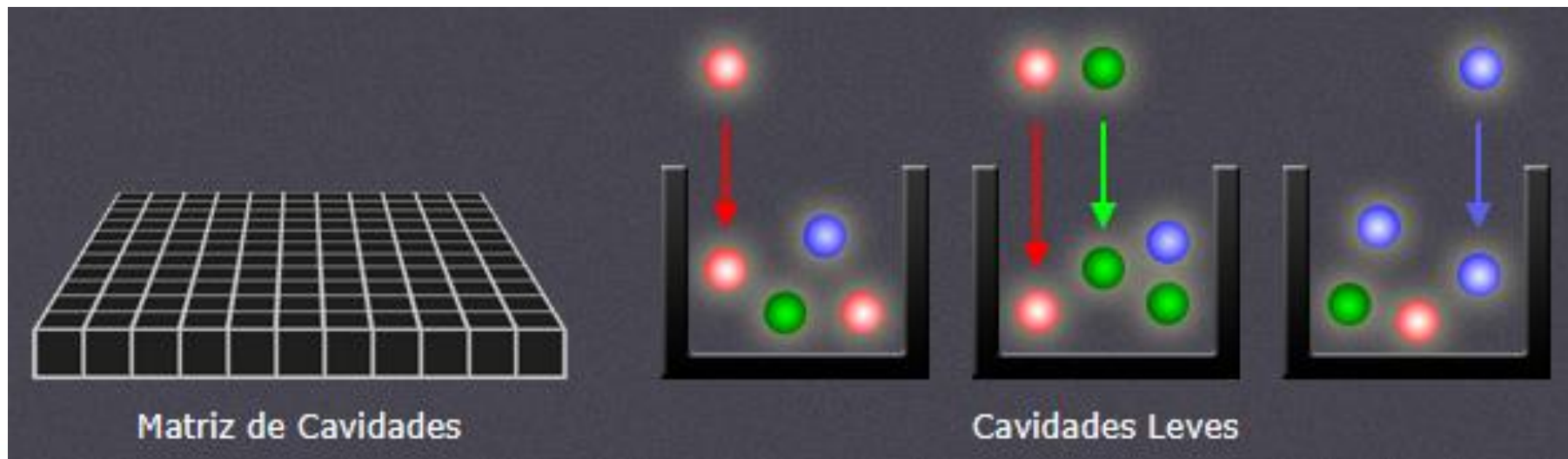
Aquisição de imagens utilizando sensores matriciais

- Enquanto isso, circuitos digitais e analógicos realizam uma varredura nas saídas de cada sensor e as convertem em um sinal analógico, que em sequência é digitalizado por outro componente do sistema de aquisição de imagens. (GONZALES; WOODS, 2010).
- Para realizar a conversão de uma imagem para o formato digital, deve-se fazer a digitalização dos valores das coordenadas da função $f(x,y)$ (amostragem) em ambas às coordenadas e na amplitude (quantização).

Sensores de Câmeras Digitais

Sensores de Câmeras Digitais

- Uma câmera digital usa uma série de milhões de minúsculas cavidades de luz ou *photosites* para registrar uma imagem.
- Quando se pressiona o botão do obturador da câmera e a exposição começa, cada um deles é descoberto para coletar fótons e armazená-los como um sinal elétrico.
- Terminada a exposição, a câmera fecha cada um desses *photosites* e tenta avaliar quantos fótons caíram em cada cavidade medindo a força do sinal elétrico.



Sensores de Câmeras Digitais

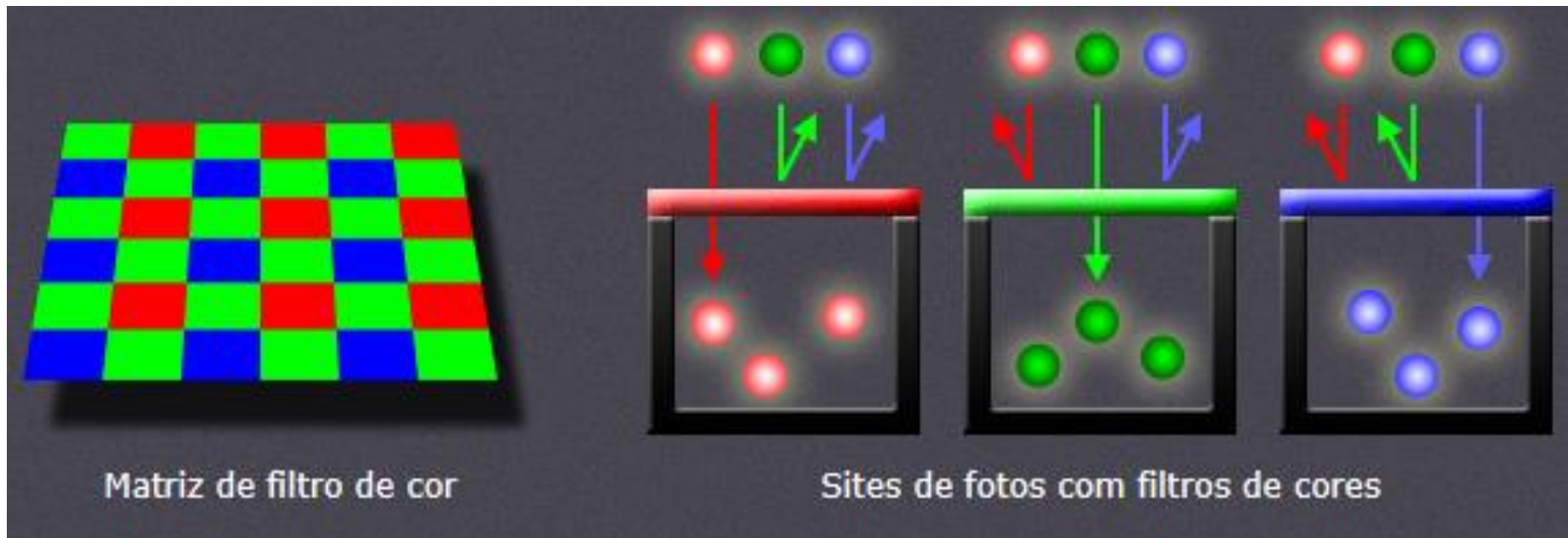
- Os sinais são então quantificados como valores digitais, com uma precisão determinada pela profundidade de bits.
- A precisão resultante pode então ser reduzida novamente dependendo de qual formato de arquivo está sendo gravado (0 - 255 para um arquivo JPEG de 8 bits).
- No entanto, a ilustração anterior criaria apenas imagens em tons de cinza, pois essas cavidades não conseguem distinguir o quanto têm de cada cor.

Sensores de Câmeras Digitais

- Para capturar imagens coloridas, um filtro deve ser colocado sobre cada cavidade que permita apenas cores específicas de luz.
- Praticamente todas as câmeras digitais atuais podem capturar apenas uma das três cores primárias em cada cavidade e, portanto, descartam cerca de $2/3$ da luz recebida.
- Como resultado, a câmera precisa aproximar as outras duas cores primárias para ter cores completas em cada pixel.

Sensores de Câmeras Digitais

- O tipo mais comum de matriz de filtro de cores é chamado de “**Matriz Bayer**”, mostrado abaixo.



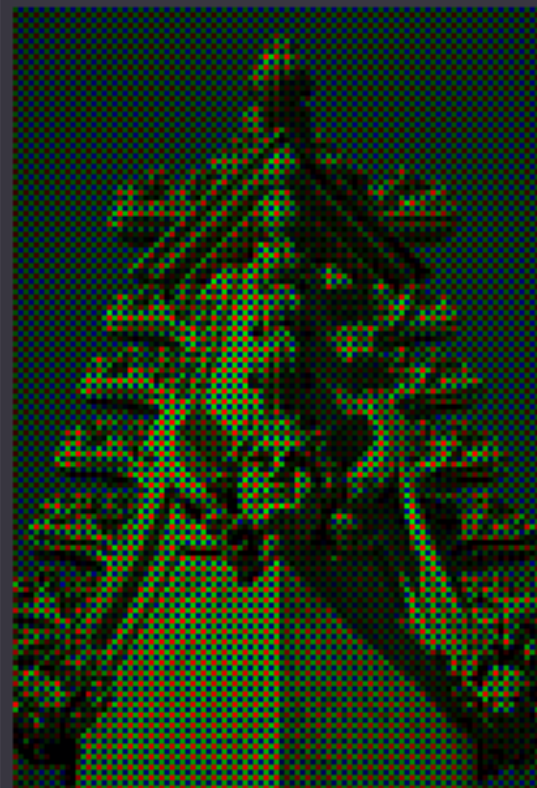
- Uma Matriz Bayer consiste em linhas alternadas de filtros vermelho-verde e verde-azul. Observe como a matriz Bayer contém duas vezes mais sensores verdes do que sensores vermelhos ou azuis.

Sensores de Câmeras Digitais

- Nem todas as câmeras digitais usam uma Matriz Bayer, mas esta é de longe a configuração mais comum.



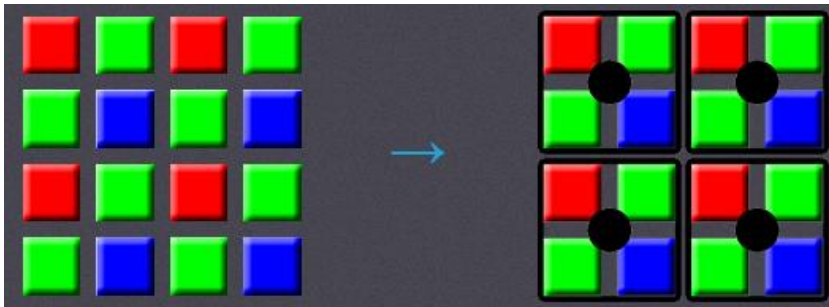
Cena original
(mostrada em 200%)



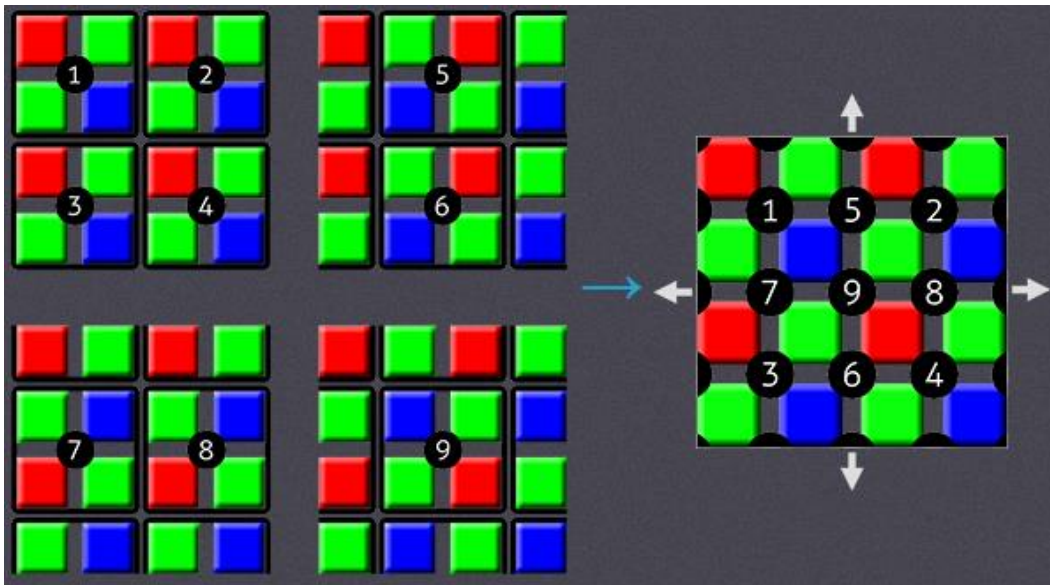
O que sua câmera vê
(através de uma matriz Bayer)

Sensores de Câmeras Digitais

- Bayer "**desmosaicing**" é o processo de traduzir esta matriz Bayer de cores primárias em uma imagem final que contém informações de cores completas em cada pixel.

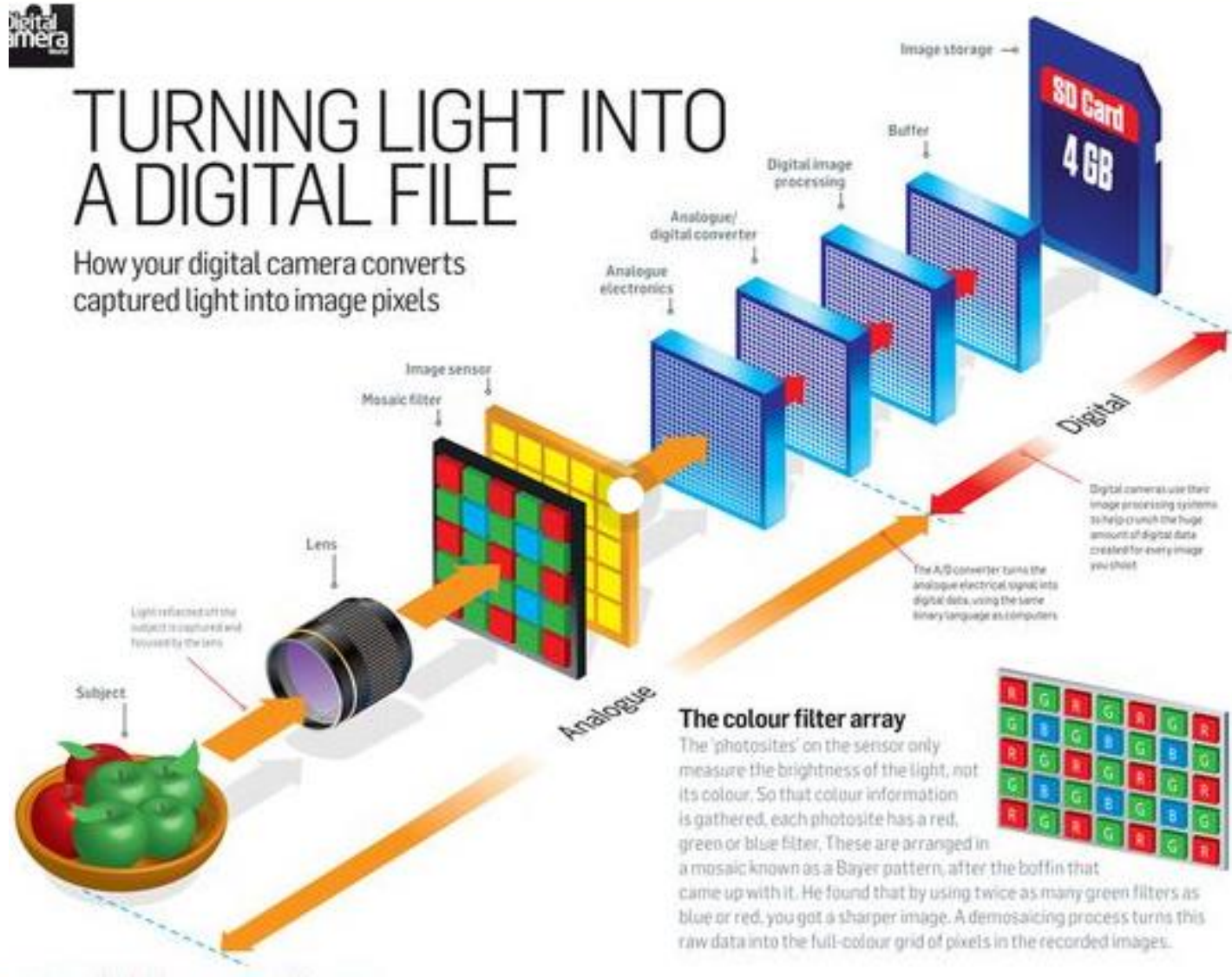


- Se a câmera tratasse todas as cores em cada matriz 2x2 como se estivessem no mesmo lugar, ela seria capaz de atingir apenas metade da resolução nas direções horizontal e vertical.



- Por outro lado, se uma câmera calculasse a cor usando várias matrizes 2x2 sobrepostas, ela poderia obter uma resolução mais alta do que seria possível com um único conjunto de matrizes 2x2.
- A seguinte combinação de matrizes 2x2 sobrepostas pode ser usada para extrair mais informações da imagem.

Sensores de Câmeras Digitais



Digitalização de Imagens

Digitalização de Imagens

- Para que uma imagem possa ser armazenada e/ ou processada em um computador, torna-se necessária sua discretização tanto em nível de coordenadas espaciais quanto de valores de brilho.
- O processo de discretização das coordenadas espaciais denomina-se **amostragem**.
- O processo de discretização dos valores de brilho denomina-se **quantização**.

Digitalização de Imagens

- O sinal analógico de vídeo obtido à saída do dispositivo de aquisição deve ser submetido a uma **discretização espacial** e em **amplitude** para tomar o formato desejável ao processamento computacional.
- **Amostragem** é o processo de discretização espacial.
- **Quantização** é processo de discretização em amplitude.

Digitalização (Amostragem)

- Basicamente, a **amostragem** converte a imagem analógica em uma matriz de ***M*** por ***N*** pontos, cada qual denominado pixel (ou elemento de imagem):

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix}$$

- Maiores valores de ***M*** e ***N*** implicam em uma imagem de maior resolução.

Digitalização (Quantização)

- A **quantização** faz com que cada um destes pixels assumam um valor inteiro, na faixa de 0 a $2^n - 1$.
- Quanto maior o valor de ***n***, maior o número de níveis de cinza presentes na imagem digitalizada.
- ***Diferentes níveis de quantização de intensidade resultam em diferentes níveis de qualidade de cor de imagem.***

Digitalização (Quantização)

- Em particular, uma significativa redução no nível de quantização pode ser feita ($256 \rightarrow 16$) antes que efeitos reais possam ser percebidos.



- No caso de sensores de cores, os canais vermelho, verde e azul são igualmente quantizados em uma representação de N bits (tipicamente 8 bits por canal, resultando em uma imagem em cores de 24 bits).

Digitalização (Quantização)

- Nas Figuras (A) a (D), ilustra-se a influência dos parâmetros de digitalização na qualidade visual de uma imagem monocromática.



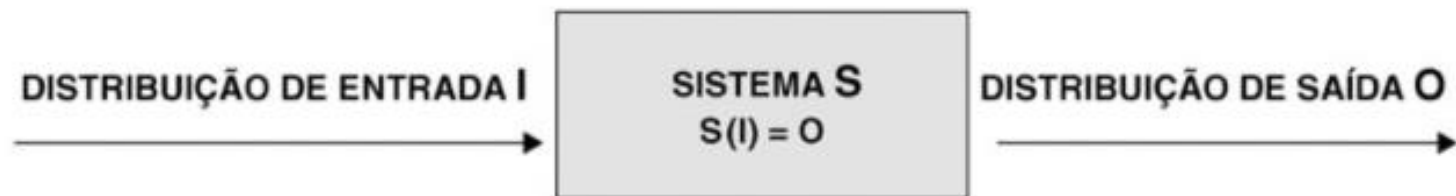
Influência da variação do número de amostras e de níveis de quantização na qualidade de uma imagem digital: (A) 200 x 200 pixels/ 256 níveis; (B) 100 x 100 pixels/ 256 níveis; (C) 25 x 25 pixels/ 256 níveis; e (D) 200 x 200 pixels/ 2 níveis.

Digitalização

- Do ponto de vista eletrônico, a digitalização consiste em uma conversão analógico-digital na qual o número de amostras do sinal contínuo por unidade de tempo indica a taxa de amostragem e o número de bits do conversor A/D utilizado determina o número de tons de cinza resultantes na imagem digitalizada.

Matemática da formação de imagens

- De um ponto de vista matemático geral, podemos ver a formação de uma imagem como um processo que transforma uma distribuição de entrada em uma distribuição de saída.
- Assim, uma simples lente pode ser vista como um 'sistema' que transforma uma distribuição espacial de luz em um domínio (plano do objeto) em uma distribuição em outro plano (plano de imagem).



Abordagem de sistema ao processo de captura de imagens. O processo é visto como um operador S que atua sobre a distribuição de entrada I para produzir a saída O .

Ruído

Ruído

- O principal obstáculo ao eficiente processamento de imagens e de sinal é, em geral, o ruído.
- Ruído significa uma pequena variação (aleatória) sofrida pelo sinal em torno de seu verdadeiro valor devido a fatores externos ou internos durante o processamento da imagem.
- Ruído é o principal problema em 99% dos casos em que as técnicas de processamento de imagens falham ou processamento adicional se torna necessário para a obtenção do resultado desejado.

Ruído



Imagem Original



Ruído Sal e Pimenta (impulsivo)

- Grande parte do domínio de processamento de imagens e de qualquer sistema de processamento de imagens é dedicada à redução e à remoção do ruído.



Exemplos de efeito de ruído em imagens digitais.

Origem do Ruído em Imagens

- O ruído é introduzido em cada estágio do processamento, captura e amostragem de imagens.
- Em imagens digitais, o ruído pode originar-se de diferentes fontes:
 - **Ruído de captura:** pode resultar de variações na iluminação, temperatura do sensor, ruído elétrico no sensor, não uniformidade do sensor, poeira no ambiente, vibração, distorção da lente, limitações de foco, saturação do sensor (luz em demasia), subexposição (deficiência de luz).
 - **Ruído de amostragem:** limitações na amostragem e na quantização de intensidade são fontes de ruído na forma de mascaramento (*aliasing*) de representação.

Origem do Ruído em Imagens

- **Ruído de processamento:** Limitações na precisão numérica (números em ponto flutuante), overflow de inteiros e aproximações matemáticas (por exemplo, $\pi = 3,142\dots$) são possíveis fontes de ruído no próprio processamento.
- **Ruído de codificação de imagem:** Muitas técnicas modernas de compressão de imagens (por exemplo, JPEG usado intrinsecamente pelas modernas câmeras digitais) são técnicas de compressão com perda. Em geral, perda = artefatos de compressão = ruído.
- **Oclusão de cena:** Na tarefa de reconhecimento de objetos, muitas vezes, esses são obscurecidos por outros.

Tipos de Ruído em Imagens

- Ignorando o ruído sistemático associado a uma causa específica (como iluminação não uniforme), o ruído em imagens pode ser caracterizado de diferentes formas.
- As duas principais caracterizações de ruído usadas no processamento de imagens são:
 - **Ruído sal e pimenta;**
 - **Ruído gaussiano.**

Tipos de Ruído em Imagens

- **Ruído sal e pimenta:**

- Causado pela introdução aleatória de pixels puramente brancos ou pretos (intensos/fracos) na imagem.
- Esse tipo de ruído também é conhecido como **ruído impulsivo**.

- **Ruído gaussiano:**

- Neste caso, a variação aleatória do sinal de imagem em torno de seu valor esperado segue a distribuição gaussiana ou normal.
- Este tipo de ruído também é conhecido como **ruído aditivo**.

Exercício 1

A engenharia da formação de uma imagem impõe limites à resolução espacial de uma imagem através do processo de digitalização.

No limite da resolução espacial, certos detalhes da imagem são perdidos.

Usando algumas imagens, explore o uso da função ***imresize()***, tanto em sua forma mais simples (usando apenas um parâmetro de escala) como especificando largura e altura (linhas e colunas) da imagem.

À medida que o tamanho da imagem é reduzido, a partir de quando certos detalhes se deterioram a ponto de impedir reconhecimento/compreensão da imagem?

Explore, ainda, o uso de ***imresize()*** para ampliar imagens. Explore os efeitos da ampliação de uma imagem usando as três funções de interpolação disponíveis para a função ***imresize()*** de Matlab.

Que diferenças você nota? (Escreva sobre as diferentes opções de interpolação empregando as funções ***tic()/toc()*** de Matlab.)

Exercício 2

Além da resolução espacial, a engenharia da formação de uma imagem também impõe limites no número de níveis de quantização (cor ou escala de cinza) em uma dada imagem.

Usando a função ***imapprox()*** de Matlab, podemos aproximar uma dada imagem representada em um certo espaço de cores (por exemplo, RGB ou escala de cinza) com um menor número de cores.

Esta função opera em uma imagem indexada (basicamente, uma imagem com uma tabela de referência de consulta para cada um de seus valores, conhecida como **mapa de cores**) – consulte “***indexed images***” na Ajuda de Matlab.

Uma imagem convencional pode ser convertida em uma imagem indexada e associada ao mapa de cores de referência através da função ***rgb2ind()*** de Matlab (ou da função ***gray2ind()***, no caso de imagens em escala de cinza).

Explore o uso de ***imapprox()*** e ***rgb2ind()/gray2ind()*** para reduzir o número de cores em imagens em cores e em escala de cinza. Experimente diferentes níveis de redução de cor.

Como suas observações se relacionam à discussão sobre a percepção de cores por humanos?

Exercício 3

A função ***imtool()*** de Matlab nos permite exibir uma imagem e nela executar diferentes operações interativas.

Use tal função para carregar o exemplo de imagem ('railway.png') e meça o comprimento em pixels dos dormentes na frente e no fundo da imagem.

O que você pode dizer sobre a distancia relativa dos dois dormentes à câmera?

Assumindo que o comprimento padrão de um dormente seja 2,5 m, que informação adicional é necessária para determinar as distancias absolutas entre esses dois itens e a câmera?

Como isto pode ser aplicado a outras áreas, como estimativa de distancia entre pessoas ou carros e a câmera?

Exercício 4

Grande Faixa Dinâmica (*HDR – High Dynamic Range*) é uma nova metodologia para captura (formação) de imagens em que a imagem é digitalizada com um numero muito maior de níveis de quantização entre os níveis mínimo e máximo de cor.

Matlab suporta a técnica HDR por meio da função ***hdrread()***.

Use essa função para carregar o exemplo de imagem HDR (office.hdr) e use a função ***tonemap()*** de Matlab para convertê-la à convencional representação RGB para visualização (com ***imshow()***).

Use as funções ***getrangefromclass()*** e ***imshow()*** de Matlab para explorar as faixas dinâmicas dessas duas versões da imagem HDR.

Exercício 5

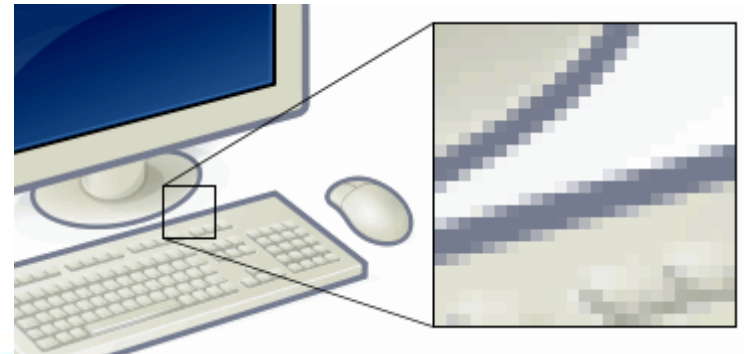
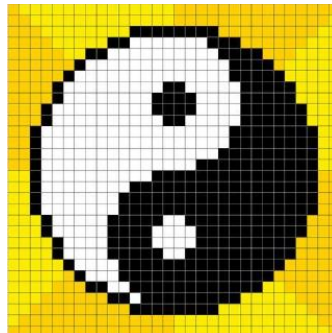
A função ***imnoise()*** de Matlab pode ser usada para adicionar ruído a imagens.

Explore a utilização dessa função, criando um script de Matlab que mostre o antes e depois da aplicação de ruídos distintos em imagens.

PIXEL

O que é um *pixel*?

- O termo pixel resulta da aglutinação das palavras inglesas “*picture element*” ou elemento de imagem.
- Indexado como uma posição (x, y) ou coluna-linha (c, r) a partir da origem da imagem, um pixel representa o menor elemento constituinte de uma imagem digital e contém um valor numérico que é a unidade básica de informação sobre a imagem, para uma dada resolução espacial e correspondentes níveis de quantização.
- Em geral, pixels contêm a resposta de cor ou intensidade da imagem na forma de pequenas amostras pontuais de luz colorida da cena.



O que é um *pixel*?

O conteúdo de informação de pixels pode variar consideravelmente, dependendo do tipo de imagem em processamento:

- Imagens em cor/escala de cinza;
- Infravermelho;
- Sistemas de captura de imagens médicas;
- Sistemas de captura de imagens de radar/sonar;
- Sistemas de captura de imagens 3D;
- Sistemas de captura de imagens científicas.



Imagem Térmica IR



Imagem de Profundidade 3D



Imagem Médica TC 3D

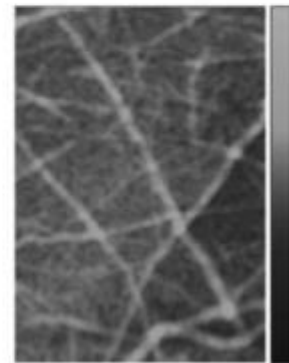


Imagem Científica
(Rik Higham, UoE (2006))

Diferentes tipos de informação de pixel.

Operações em *Pixels*

Operações em *Pixels*

- O tipo mais básico de operação de processamento de imagens é uma transformação pontual, que mapeia os valores em pontos individuais (*pixels*) na imagem de entrada a pontos correspondentes (*pixels*) na imagem de saída.

$$I_{\text{saída}} = I_A + I_B$$

$$I_{\text{saída}} = I_A + C$$

- Nos dois casos, os valores em posições individuais de *pixel* (i, j) na imagem de saída são mapeados da seguinte forma:

$$I_{\text{saída}}(i, j) = I_A(i, j) + I_B(i, j)$$

$$I_{\text{saída}}(i, j) = I_A(i, j) + C$$

Operações Lógicas e Aritméticas

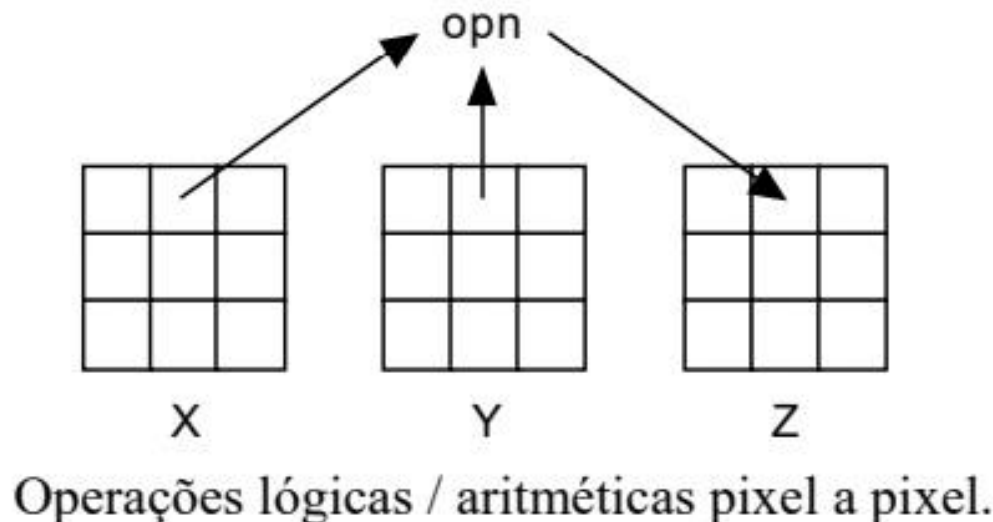
- Sabemos que após uma imagem ter sido adquirida e digitalizada, ela pode ser vista como uma matriz de inteiros e portanto pode ser manipulada numericamente utilizando operações lógicas e/ou aritméticas.
- Estas operações podem ser efetuadas *pixel a pixel* ou orientadas a vizinhança.
- No primeiro caso, elas podem ser descritas pela seguinte notação:

$$X \text{ opn } Y = Z$$

- onde X e Y podem ser imagens (matrizes) ou escalares, Z é obrigatoriamente uma matriz e **opn** é um operador aritmético (+, -, x e /) ou lógico (AND, OR, XOR) binário.

Operações Lógicas e Aritméticas

- Sejam duas imagens ***X*** e ***Y*** de igual tamanho.
- Estas imagens podem ser processadas *pixel a pixel* utilizando um operador aritmético ou lógico, produzindo uma terceira imagem ***Z***, cujos *pixels* correspondem ao resultado de ***X opn Y*** para cada elemento de ***X*** e ***Y***, conforme ilustra esquematicamente a figura abaixo:



Operações aritméticas *pixel a pixel*

Efeitos e aplicações das operações aritméticas sobre imagens

| Operação | Efeito sobre a imagem | Aplicações |
|---------------|---|---|
| Adição | Z é o resultado da soma dos valores de intensidade de X e Y . Se Y for um escalar positivo, Z será uma versão mais clara de X ; o acréscimo de intensidade será o próprio valor de Y . | <ul style="list-style-type: none">• Normalização de brilho³ de imagens• Remoção de ruídos |
| Subtração | Z é o resultado da diferença dos valores de intensidade de X e Y . Se Y for um escalar positivo, Z será uma versão mais escura de X ; o decréscimo de intensidade será o próprio valor de Y . | <ul style="list-style-type: none">• Detecção de diferenças entre duas imagens (eventualmente adquiridas de forma consecutiva) da mesma cena |
| Multiplicação | Z é o produto dos valores de intensidade de X e Y . Se Y for um escalar positivo, os valores de intensidade de Z serão diretamente proporcionais a X por um fator Y . | <ul style="list-style-type: none">• Calibração de brilho⁴ |
| Divisão | Z é o razão dos valores de intensidade de X pelos valores correspondentes em Y . Se Y for um escalar positivo, os valores de intensidade de Z serão inversamente proporcionais a X por um fator Y . | <ul style="list-style-type: none">• Normalização de brilho |

Operações aritméticas pixel a pixel

- Ao executarmos operações aritméticas sobre imagens, devemos tomar especial cuidado com os problemas de *underflow* ou *overflow* do resultado.
- A adição de duas imagens de 256 tons de cinza, por exemplo, pode resultar em um número maior que 255 para alguns *pixels*, ao mesmo tempo que a subtração de duas imagens pode resultar em valores negativos para alguns elementos.

Operações aritméticas pixel a pixel

- Para contornar estes problemas, existem basicamente duas alternativas:
 1. Manter os resultados intermediários em uma matriz na qual o espaço em memória alocado para cada *pixel* permita a representação de números negativos e/ou maiores que 255 e em seguida **proceder a uma normalização** destes valores intermediários;
 2. **Truncar os valores** maiores que o máximo valor permitido, bem como os valores negativos, igualando-os a 255 e 0, respectivamente.

Exercício 6

Dadas as matrizes X e Y a seguir, correspondentes a trechos 3x3 de imagens de 256 tons de cinza, adicioná-las e informar:

- (a) O resultado intermediário (sem considerações de *underflow* e *overflow*),
- (b) O resultado final utilizando normalização,
- (c) O resultado final utilizando truncamento.

$$X = \begin{bmatrix} 200 & 100 & 100 \\ 0 & 10 & 50 \\ 50 & 250 & 120 \end{bmatrix}$$

$$Y = \begin{bmatrix} 100 & 220 & 230 \\ 45 & 95 & 120 \\ 205 & 100 & 0 \end{bmatrix}$$

Exercício 6 (Resolução)

(a) O resultado intermediário (sem considerações de *underflow* e *overflow*):

$$X = \begin{bmatrix} 200 & 100 & 100 \\ 0 & 10 & 50 \\ 50 & 250 & 120 \end{bmatrix} + Y = \begin{bmatrix} 100 & 220 & 230 \\ 45 & 95 & 120 \\ 205 & 100 & 0 \end{bmatrix} = \text{(a)} \begin{bmatrix} 300 & 320 & 330 \\ 45 & 105 & 170 \\ 255 & 350 & 120 \end{bmatrix}$$

Exercício 6 (Resolução)

(b) Fazendo com que a escala $[45, 350]$ seja adequada ao intervalo $[0, 255]$, utilizando-se a relação

$$g = \frac{255}{f_{max} - f_{min}} (f - f_{min}),$$

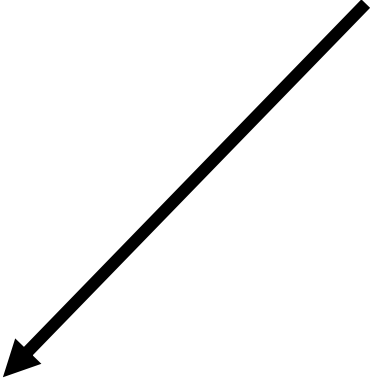
obtém-se:

$$\begin{bmatrix} 213 & 230 & 238 \\ 0 & 50 & 105 \\ 175 & 255 & 63 \end{bmatrix}$$

Exercício 6 (Resolução)

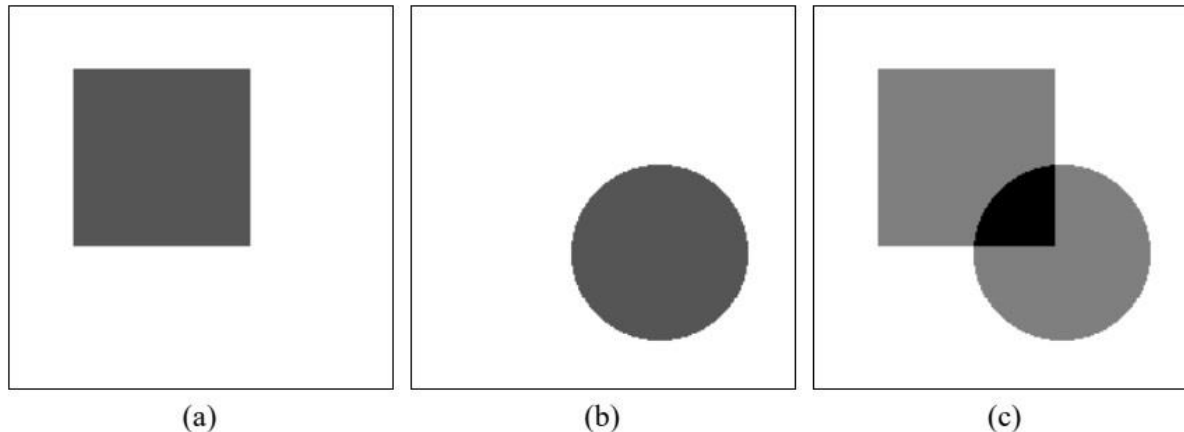
(c) Truncando os valores maiores que 255, obtém-se:

$$X = \begin{bmatrix} 200 & 100 & 100 \\ 0 & 10 & 50 \\ 50 & 250 & 120 \end{bmatrix} + Y = \begin{bmatrix} 100 & 220 & 230 \\ 45 & 95 & 120 \\ 205 & 100 & 0 \end{bmatrix} = \text{(a)} \begin{bmatrix} 300 & 320 & 330 \\ 45 & 105 & 170 \\ 255 & 350 & 120 \end{bmatrix}$$

$$\begin{bmatrix} 255 & 255 & 255 \\ 45 & 105 & 170 \\ 255 & 255 & 120 \end{bmatrix}$$


Adição

- A adição de um valor a cada valor de pixel de imagem pode ser usada para alcançar os seguintes efeitos:
 - **Ajuste de contraste:** A adição de um valor positivo constante C a cada posição de pixel aumenta o valor do pixel e, portanto, o seu brilho.
 - **Mistura:** A adição de imagens produz uma imagem composta das duas imagens de entrada. Adição ponderada pode ser usada para produzir efeitos de mistura (*blending*).



Exemplo de adição de imagens monocromáticas: (a) X , (b) Y , (c) $X + Y$ (normalizado).

Adição de valor constante

```
% Operações Aritméticas em imagens  
% PI02_Add.m  
  
%Lê a imagem Add1.jpg  
A = imread('Add1.jpg');  
  
%Exibe a imagem  
subplot(2,1,1), imshow(A);  
  
%Soma 100 a cada valor de pixel na imagem A  
B = imadd(A, 100);  
  
%Exibe a imagem resultante B  
subplot(2,1,2), imshow(B);
```



Adição de duas imagens

```
% Adição de duas imagens
% PI02_Add2.m

%Lê as imagem
A = imread('Add1.jpg');
B = imread('Add2.jpg');

%Exibe as imagens
subplot(2,2,1), imshow(A);
subplot(2,2,3), imshow(B);

%Soma as imagens -> A + B
C = A + B;

%Exibe a imagem resultante C
subplot(2,2,2), imshow(C);
```



A

+



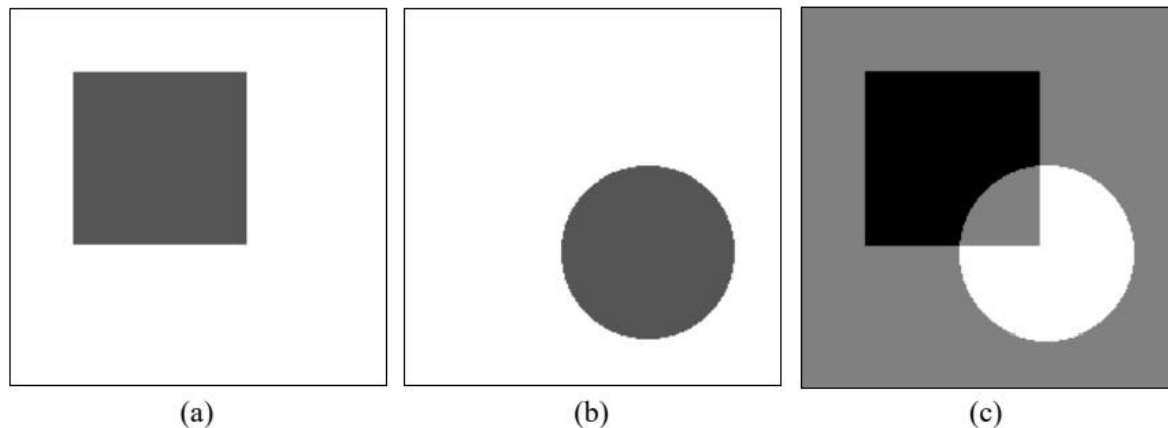
B



C

Subtração

- A subtração de um valor constante de cada posição de pixel também pode ser usada como forma básica de ajuste de contraste.
- A subtração de uma imagem de outra nos mostra a diferença entre as duas imagens.



Exemplo de subtração das imagens monocromáticas das figuras (a) e (b): $X - Y$ (normalizado).

Subtração de valor constante

```
% Operações Aritméticas em imagens  
% PI02_Subtract.m  
  
%Lê a imagem Add1.jpg  
A = imread('Add1.jpg');  
  
%Exibe a imagem  
subplot(2,1,1), imshow(A);  
  
%Soma 100 a cada valor de pixel na imagem A  
B = imsubtract(A, 100);  
  
%Exibe a imagem resultante B  
subplot(2,1,2), imshow(B);
```



Subtração de duas imagens

```
% Subtração de duas imagens  
% PI02_Subtract2.m
```

```
% Lê as imagens  
A = imread('Add1.jpg');  
B = imread('Add2.jpg');
```

```
% Exibe as imagens  
subplot(2,2,1), imshow(A);  
subplot(2,2,2), imshow(B);
```

```
% Subtrai as imagens -> A - B  
C = A - B;  
D = imsubtract(A, B);
```

```
% Exibe as imagens resultantes C e D  
subplot(2,2,3), imshow(C);  
subplot(2,2,4), imshow(D);
```



A



B

—



C



D

Diferença Absoluta

- Uma variação útil da subtração é a **Diferença Absoluta** $|I_A - I_B|$ entre imagens.
- Isso evita o potencial problema de *overflow* de inteiro quando a diferença se torna negativa.

```
% Diferença Absoluta de duas imagens
% PI02_imabsdiff.mlx

%Lê as imagens
A = imread('Add1.jpg');
B = imread('Add2.jpg');

%Exibe as imagens
subplot(2,2,1), imshow(A);
subplot(2,2,2), imshow(B);

%Diferença Absoluta das imagens -> |A - B|
C = imabsdiff(A, B);

%Subtrai as imagens -> A - B
D = imsubtract(A, B);

%Exibe as imagens resultantes C e D
subplot(2,2,3), imshow(C);
subplot(2,2,4), imshow(D);
```



A



B



C



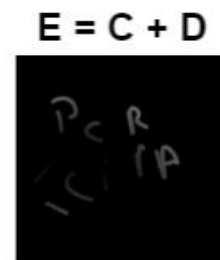
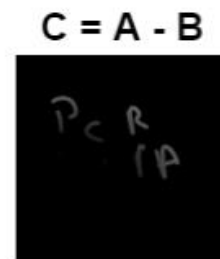
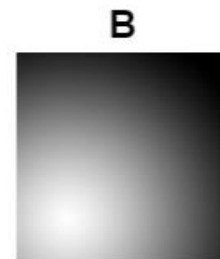
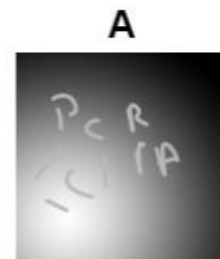
D

Diferença entre duas imagens

```
% Lê uma imagem
A = imread("back1.jpg");
B = imread("back2.jpg");

% Diferença entre duas imagens
C = A - B;
D = B - A;
E = C + D;

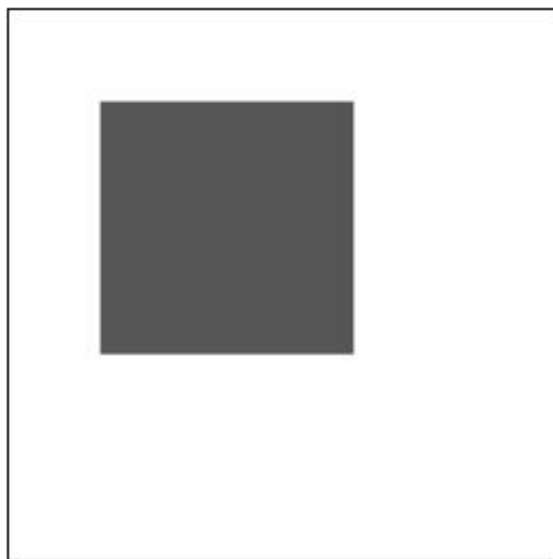
% Exibe a imagem final
subplot(3,2,1), imshow(A); title('A');
subplot(3,2,2), imshow(B); title('B');
subplot(3,2,3), imshow(C); title('C = A - B');
subplot(3,2,4), imshow(D); title('D = B - A');
subplot(3,2,5), imshow(E); title('E = C + D');
```



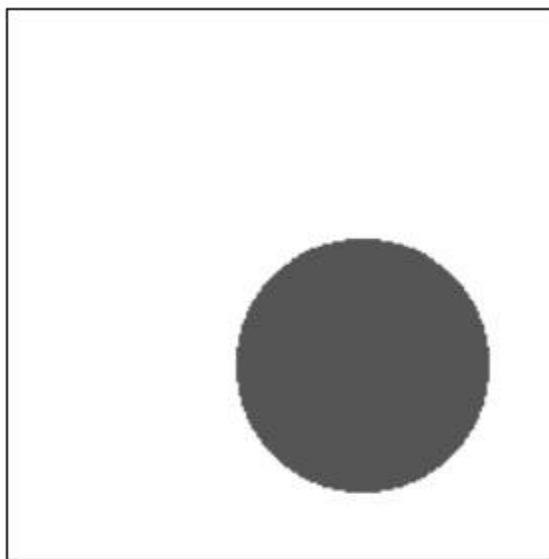
Multiplicação e Divisão

- Multiplicação e Divisão podem ser usadas como meios simples de ajuste de contraste e como extensão à adição/subtração.
 - Por exemplo, redução de contraste em 25% = divisão por 4;
 - Aumento de contraste em 50% = multiplicação por 1,5.
- Às vezes, esse procedimento é referido como ajuste de cor de imagens (*image colour scaling*).
- Do mesmo modo, a divisão pode ser usada para discriminar uma imagem de outra;
 - A divisão de uma imagem por outra tem 1,0 como resultado quando os valores de pixel são idênticos, e um resultado diferente de 1,0 quando ocorrem diferenças.

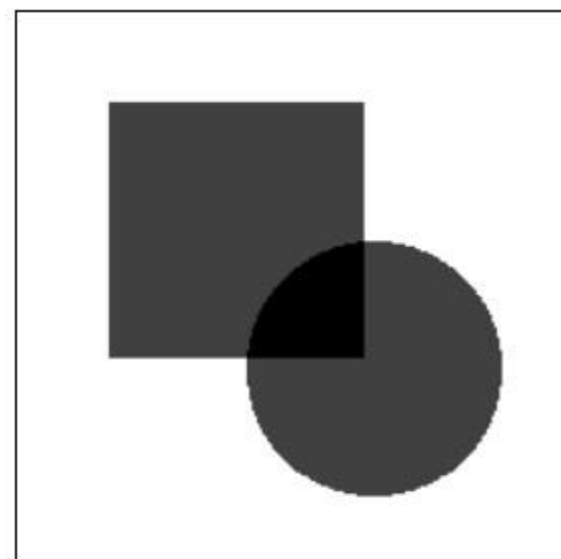
Multiplicação



(a)



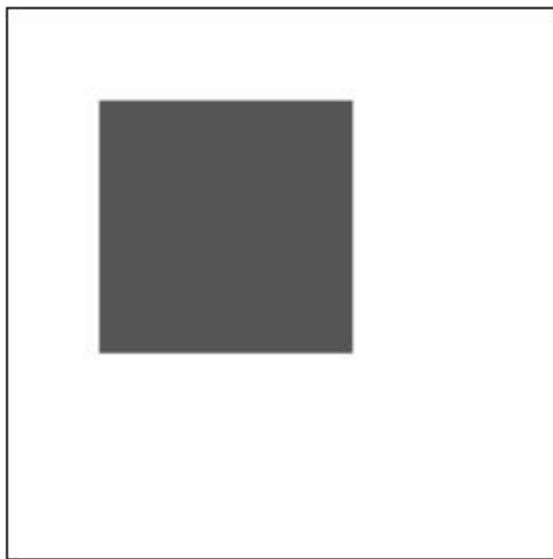
(b)



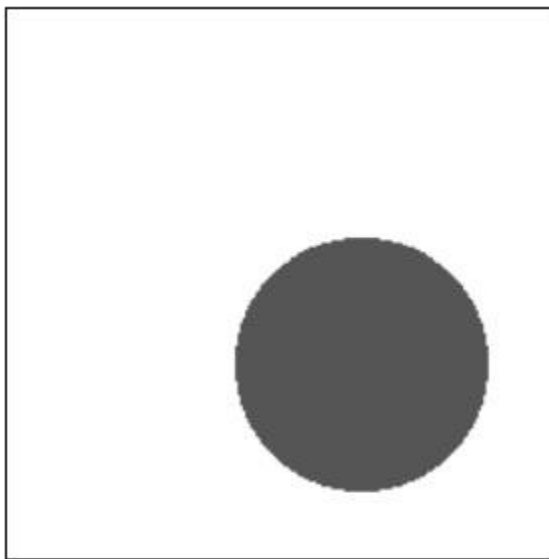
(c)

Exemplo de multiplicação das imagens monocromáticas das figuras (a) e (b): XY (normalizado).

Divisão



(a)



(b)



(c)

Exemplo de divisão de imagens monocromáticas das figuras (a) e (b): X / Y (normalizado).

Multiplicação e Divisão

```
% Multiplicação e Divisão de duas imagens  
% PI02_MultiplyDivide.mlx
```

```
% Lê as imagens
```

```
A = imread('Add1.jpg');
```

```
% Multiplicação por valor constante
```

```
% Aumenta o contraste em 50%
```

```
B = immultiply(A, 1.5);
```

```
% Divisão por valor constante
```

```
% Diminui o contraste em 25%
```

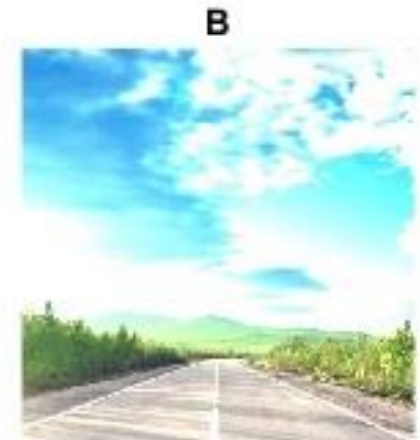
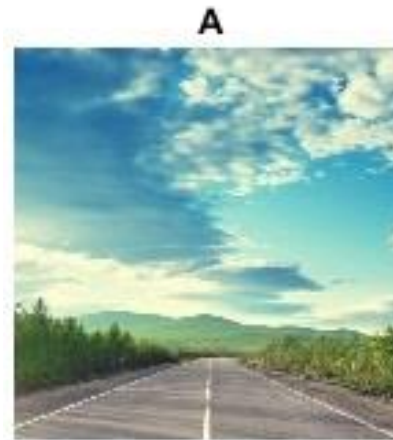
```
C = imdivide(A, 4);
```

```
% Exibe as imagens
```

```
subplot(2,2,1), imshow(A), title('A');
```

```
subplot(2,2,2), imshow(B), title('B');
```

```
subplot(2,2,4), imshow(C), title('C');
```



Combinação Linear (Blending)

- A operação de *Blending* produz a mistura de duas imagens do mesmo tamanho, pela combinação da multiplicação de um escalar e adição de imagens.
- Similar a adição de pixels, o valor de cada pixel na imagem de saída é uma combinação linear dos valores correspondentes de cada pixel das imagens de entrada.
- O coeficiente da combinação linear é especificado pelo usuário, e ele define a proporção pela qual as imagens devem ser dimensionadas antes de serem combinadas.

Combinação Linear (Blending)

- Essas proporções são aplicadas de tal forma que os valores dos pixels de saída não excedam o valor máximo do pixel.
- A imagem resultante é calculada usando a seguinte fórmula:

$$R(m,n) = C \times P(m,n) + (1-C) \times Q(m,n).$$

- **P** e **Q** são as duas imagens de entrada.
- **C** é a taxa de mistura (*blending ratio*), que determina a influência de cada imagem de entrada na imagem de saída.
- **C** também pode ser um fator constante para todos os pixels na imagem ou pode ser determinado para cada pixel separadamente, usando uma máscara. O tamanho da máscara deve ser idêntico ao tamanho das imagens.

Combinação Linear (Blending)

```
P = imread('blend1.tif');
Q = imread('blend2.tif');

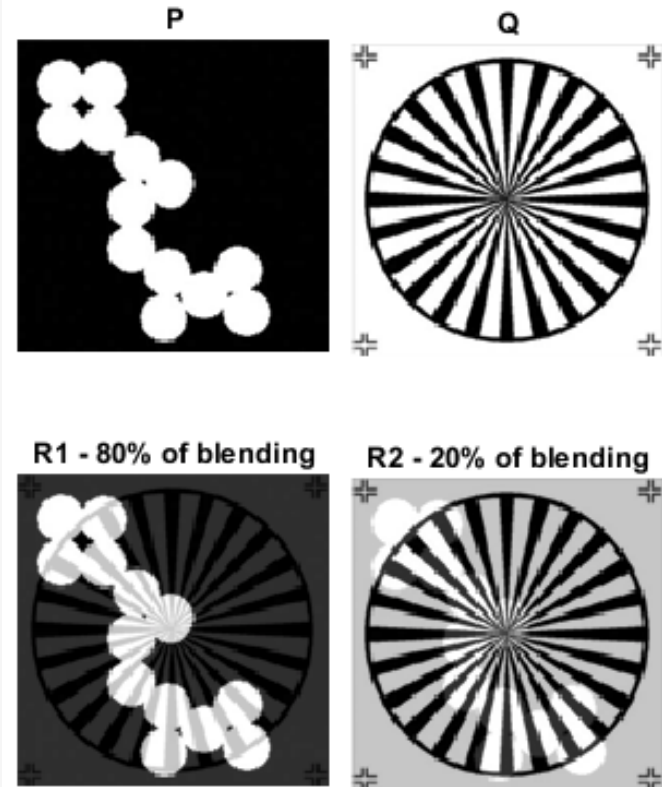
%Blend Factor
x = 0.8;

% Combinação Linear (Blending Equation)
R1 = imlincomb(x, P, 1-x, Q);

%Blend Factor
x = 0.2;

% Combinação Linear (Blending Equation)
R2 = imlincomb(x, P, 1-x, Q);

subplot(2,2,1), imshow(P), title('P');
subplot(2,2,2), imshow(Q), title('Q');
subplot(2,2,3), imshow(R1), title('R1 - 80% of blending');
subplot(2,2,4), imshow(R2), title('R2 - 20% of blending');
```



Combinação Linear (Média de duas imagens)

- Os quatro operadores definidos até agora são considerados os operadores aritméticos fundamentais de processamento de imagem.
- Em qualquer caso, você pode combinar essas funções aritméticas básicas para realizar uma serie de operações.
- Por exemplo, para calcular a **média de duas imagens**, pode-se usar a equação:

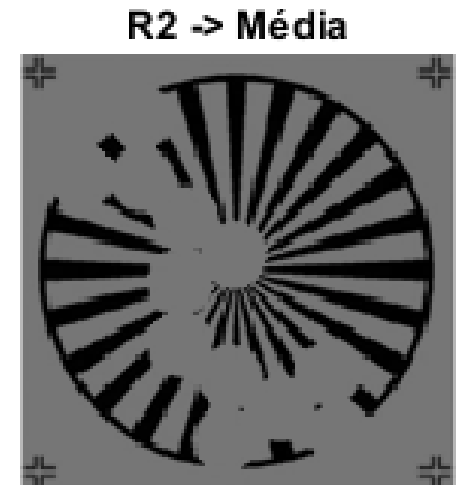
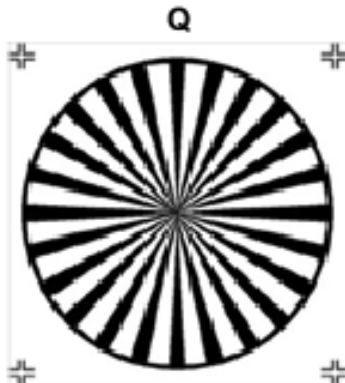
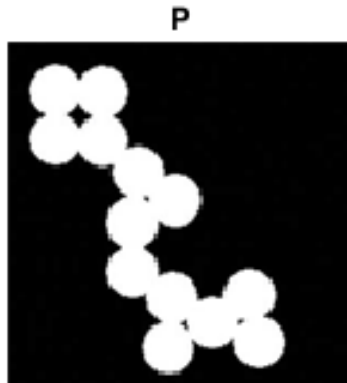
$$R(i, j) = \frac{P(i, j) + Q(i, j)}{2},$$

Combinação Linear (Média de duas imagens)

- Outras formas de calcular a média de duas imagens usando uma variação da equação anterior:
 - 1) Somar as imagens **P** e **Q** primeiro e, depois, dividir a soma por **2**.
 - 2) Dividir ambas as imagens **P** e **Q** por **2** e, depois, somar os resultados.
 - 3) Multiplicar **P** e **Q** por **0.5** e, depois, somar os resultados.
 - 4) Somar **P** e **Q** e, depois, multiplicar o resultado por **0.5**

Combinação Linear (Média de duas imagens)

```
P = imread('blend1.tif');  
Q = imread('blend2.tif');  
  
% Calcula a média das imagens  
R1 = (P + Q);  
R2 = R1 ./ 2;  
  
subplot(2,2,1), imshow(P), title('P');  
subplot(2,2,2), imshow(Q), title('Q');  
subplot(2,2,3), imshow(R1), title('R1 -> Soma');  
subplot(2,2,4), imshow(R2), title('R2 -> Média');
```



Operações lógicas pixel a pixel

- Podemos efetuar operações lógicas convencionais, como NOT, OR, XOR e AND, entre imagens.
- Em geral, uma operação lógica é efetuada entre bits correspondentes à representação da imagem em pixels (ou seja, operação bit a bit).

NOT

- **NOT (inversão):** Esta operação inverte a representação da imagem.
- No caso mais simples de uma imagem binária, os pixels de fundo (preto) se tornam frontais (brancos) e vice-versa.
- No caso de imagens em escala de cinza ou em cores, o procedimento consiste em substituir cada valor de pixel $I_{entrada}(i, j)$ da seguinte forma:

$$I_{saída}(i, j) = MAX - I_{saída}(i, j)$$

em que MAX é o máximo valor possível na dada representação da imagem. Assim, para uma imagem em escala de cinza de 8 bits (ou para canais de 8 bits de uma imagem em cores), $MAX = 255$.

NOT

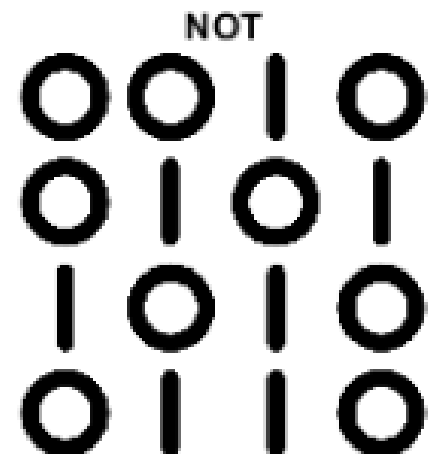
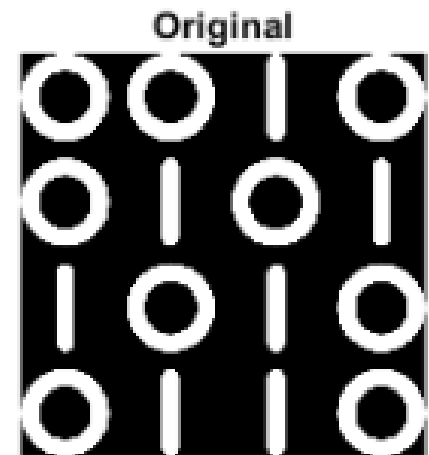
```
% Operações Lógicas em imagens
% PI02_NOT.m

% Lê a imagem Original
A = imread('binary.png');

% Exibe a imagem original
subplot(1,2,1), imshow(A), title('Original');

% Inverte a imagem (NOT)
B = imcomplement(A);

% Exibe a imagem invertida B
subplot(1,2,2), imshow(B), title('NOT');
```



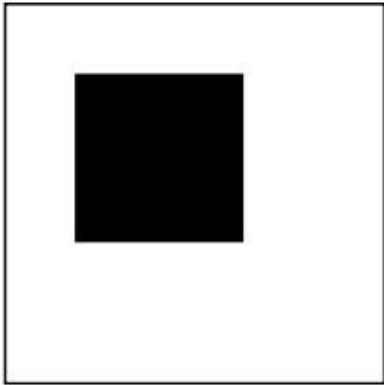
Operações lógicas pixel a pixel

Todas as operações lógicas (ou booleanas) conhecidas podem ser aplicadas entre imagens, inclusive a operação de complemento (NOT), que é uma operação unária (requer apenas um operando).

Operações lógicas podem ser efetuadas em imagens com qualquer número de níveis de cinza, mas são melhor compreendidas quando vistas em imagens binárias.

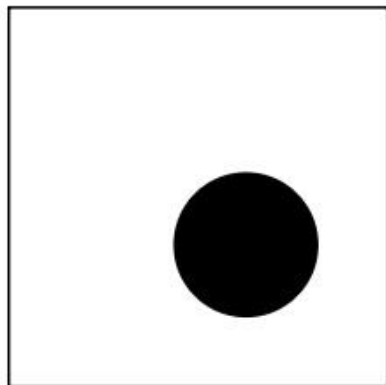
As figuras a seguir ilustram as operações AND, OR, XOR e NOT aplicadas a imagens com múltiplos tons de cinza.

Operações lógicas pixel a pixel



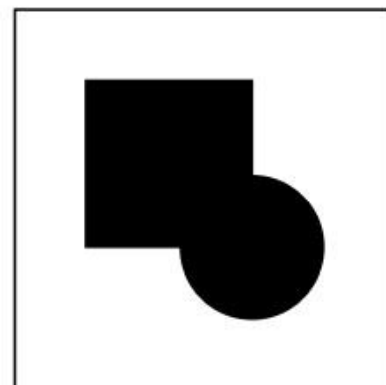
X

(a)



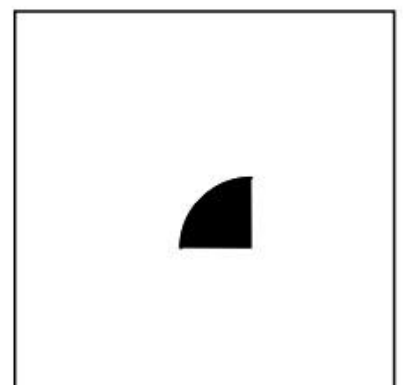
Y

(b)



X and Y

(c)



X or Y

(d)



X xor Y

(e)



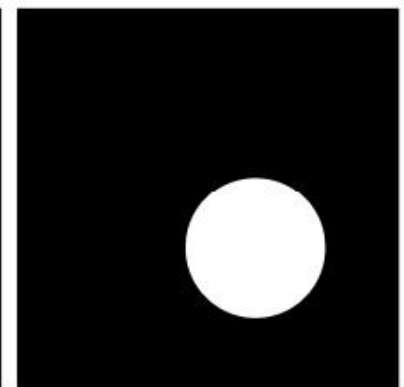
(not X) and Y

(f)



not X

(g)



not Y

(h)

Operações lógicas pixel a pixel

```
% Operações Lógicas em imagens
% PI02_AND_OR_XOR.mlx

% Lê as imagens em grayScale
A = imread('graySquare.bmp');
B = imread('grayCircle.bmp');

% Converte as imagens para Binary
A = imbinarize(A);
B = imbinarize(B);

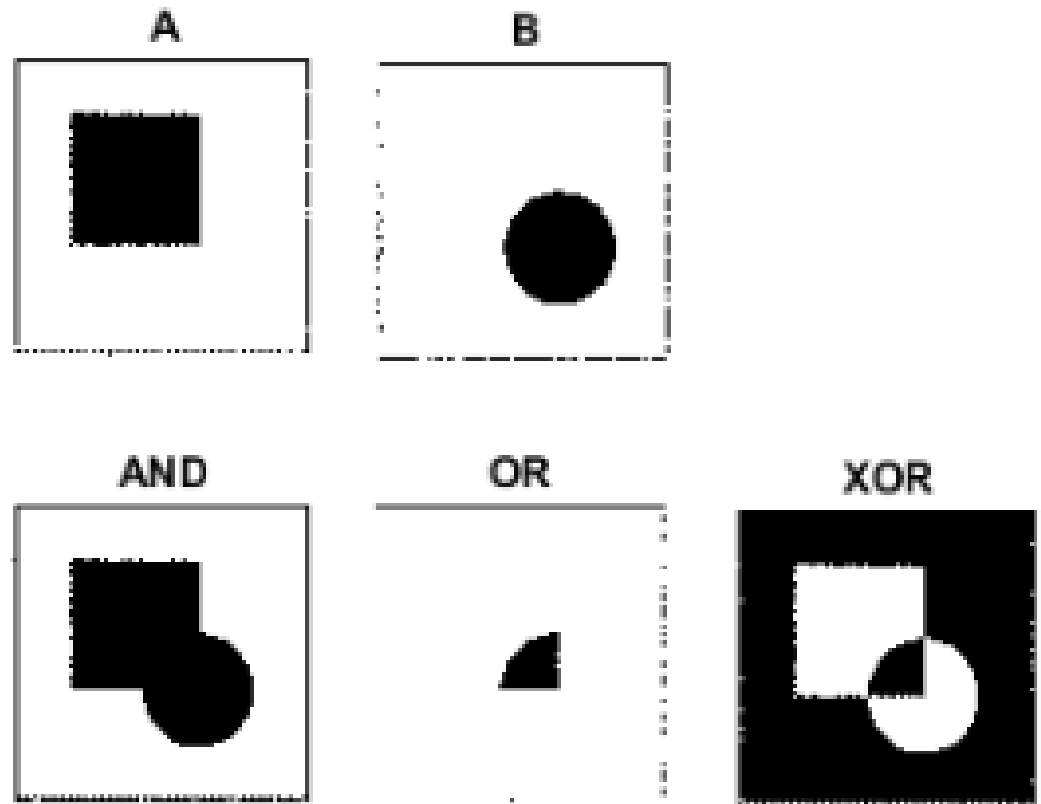
% Exibe as imagens binárias
subplot(3,2,1), imshow(A), title('A');
subplot(3,2,2), imshow(B), title('B');

% AND
% AND = (A & B);
AND = and(A, B);

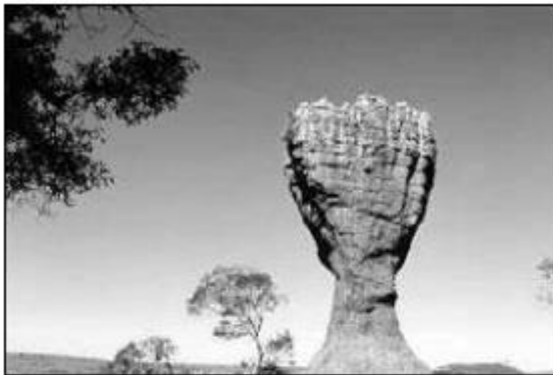
% OR
% OR = (A | B);
OR = or(A, B);

% XOR
XOR = xor(A, B);

% Exibe as operações AND, OR e XOR
subplot(3,2,3), imshow(AND), title('AND');
subplot(3,2,4), imshow(OR), title('OR');
subplot(3,2,5), imshow(XOR), title('XOR');
```



Operações lógicas pixel a pixel



(a)



(b)



(c)

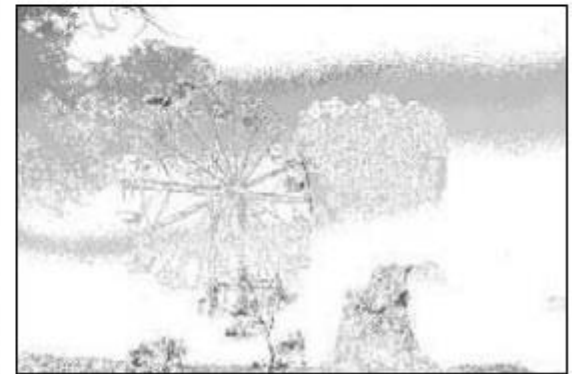
Exemplo de operação AND entre imagens monocromáticas: (a) X , (b) Y , (c) $X \wedge Y$.



(a)



(b)



(c)

Exemplo de operação OR entre imagens monocromáticas: (a) X , (b) Y , (c) $X \vee Y$.

Operações lógicas pixel a pixel



(a)



(b)

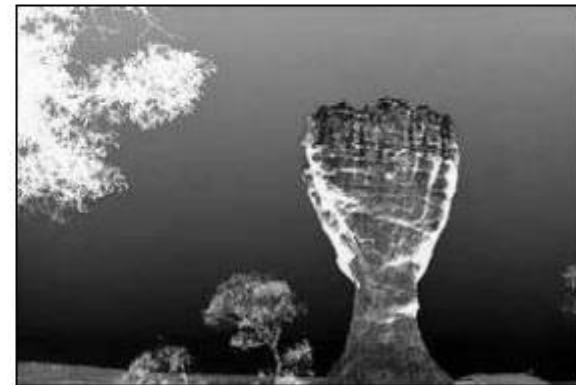


(c)

Exemplo de operação XOR entre imagens monocromáticas: (a) X , (b) Y , (c) $X \oplus Y$.



(a)



(b)

Exemplo de operação NOT sobre imagem monocromática: (a) X , (b) $\text{NOT } X$.

Exercício 7

Crie um programa em alguma linguagem de programação que gere duas matrizes quadradas $M \times N$.

Cada matriz deve ter seus valores gerados aleatoriamente, no intervalo de 0 – 255.

Em seguida, faça a soma das duas matrizes e apresente os três resultados possíveis, seguindo as regras para tratar o *underflow* e *overflow* (A, B e C) citadas anteriormente...

Exercício 8

Crie um programa em alguma linguagem leia duas imagens **P** e **Q** em escala de cinza com as mesmas dimensões $M \times N$.

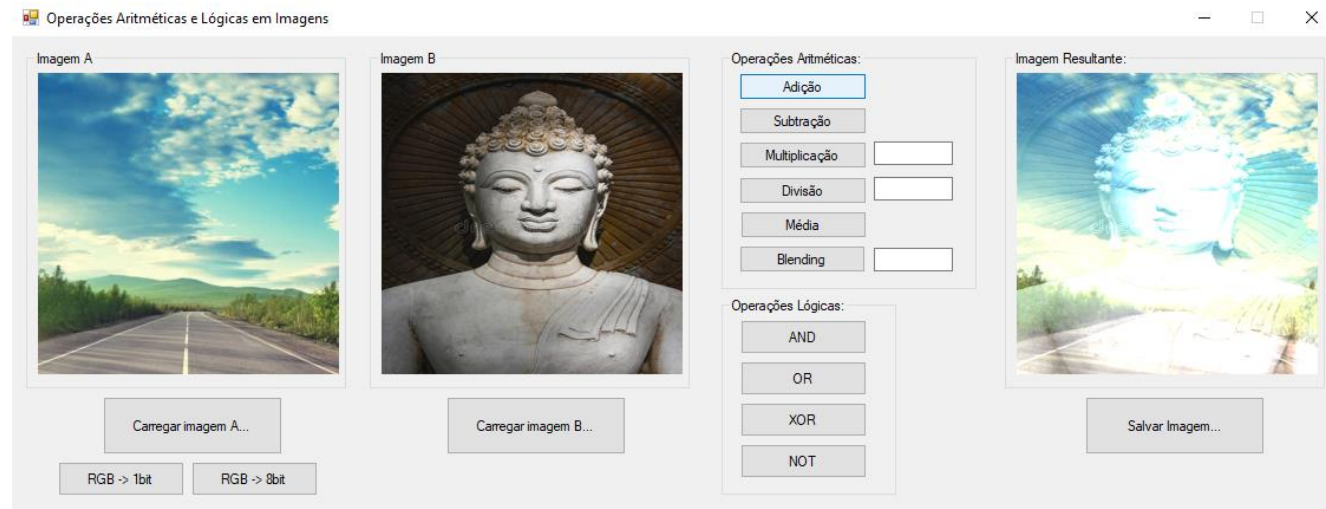
Em seguida, crie um menu com as seguintes opções que serão calculadas para o par de imagens de entrada. A saída será uma imagem **R** em escala de cinza com a devida operação realizada:

Operações Aritméticas:

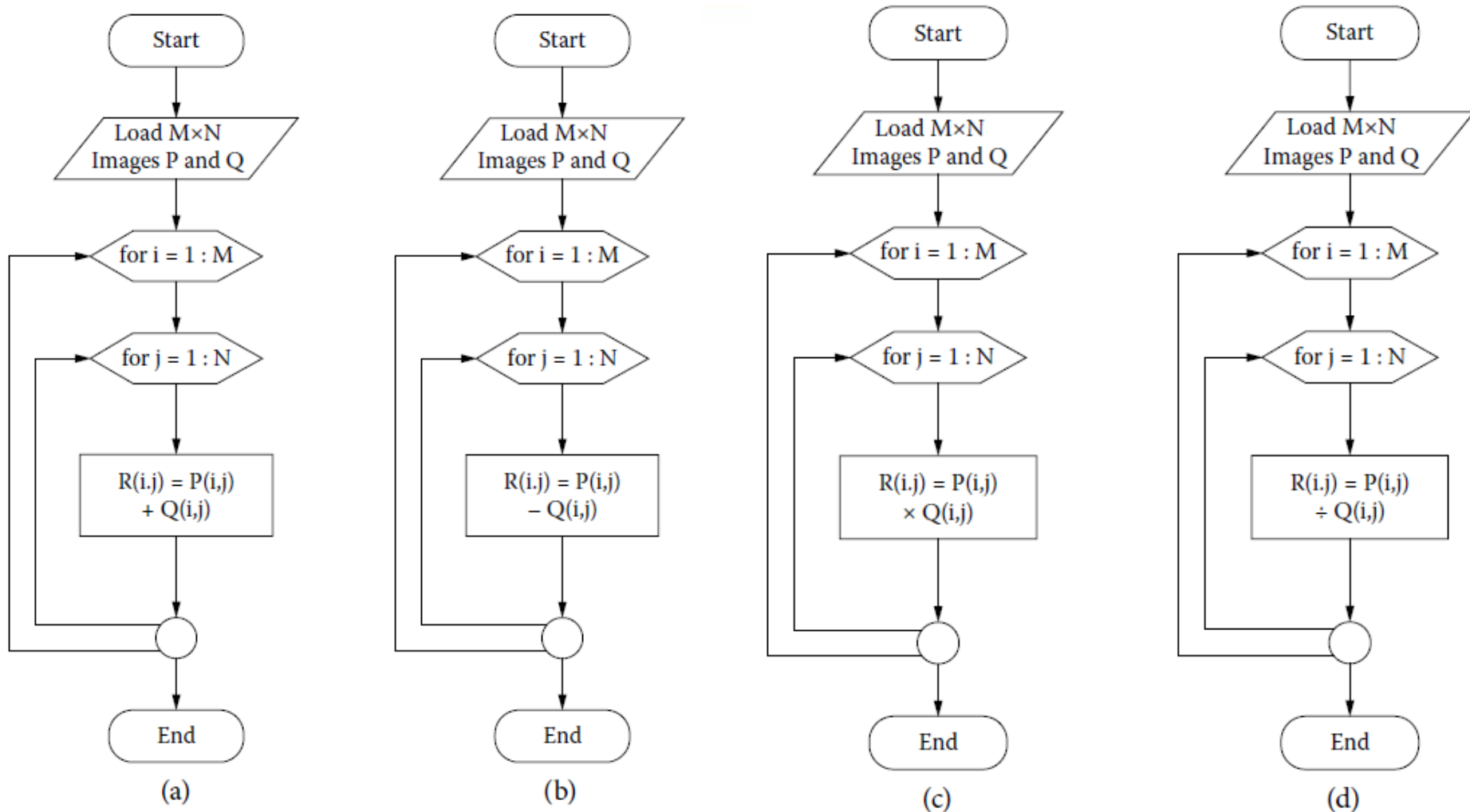
- 1) Adição
- 2) Subtração
- 3) Multiplicação
- 4) Divisão
- 5) Média
- 6) Blending

Operações Lógicas:

- 7) AND
- 8) OR
- 9) XOR
- 10) NOT

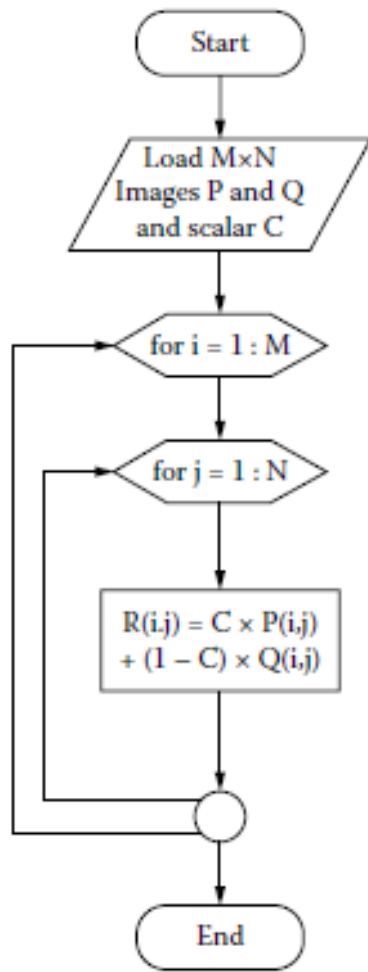


Exercício 8



Flow charts for algorithmic implementation of (a) image addition, (b) image subtraction, (c) image multiplication, (d) image division.

Exercício 8



Algorithm for image blending operation.

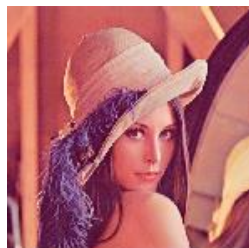
Exercício 9

Usando o Matlab, leia uma imagem de entrada **P** colorida ou em escala de cinza.

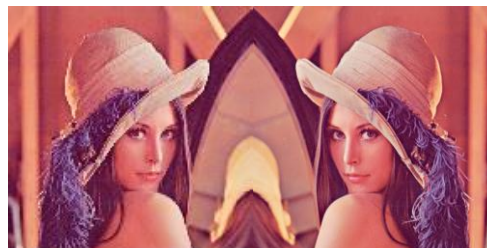
Em seguida, produza uma imagem **R** de saída com o mesmo número de linhas de **P**, porém, com o **dobro das colunas** da imagem **P**.

A porção adicional de colunas deve ser usada para acomodar a imagem espelhada da imagem de entrada **P**.

Exemplo:



P



R

Em seguida, desenvolva uma aplicação em alguma linguagem de programação que faça exatamente a mesma coisa.