Computação Gráfica

Desenhando Imagens e Formas no OpenCV
(Aula 5)

Leonardo Medeiros

Instituto Federal de Alagoas

22 de Abril de 2017

Roteiro

1. Introdução Imagens e Formas
2. Operações em Imagens
3. Prática
4. Bibliografia

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Introdução a Imagens e Formas

Como iremos ver nesta aula, utilizando *OpenCV* poderemos criar imagens e formas nas imagens. Desta maneira, conseguiremos identificar e destacar regiões nas imagens e até mesmo escrever nestas regiões.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

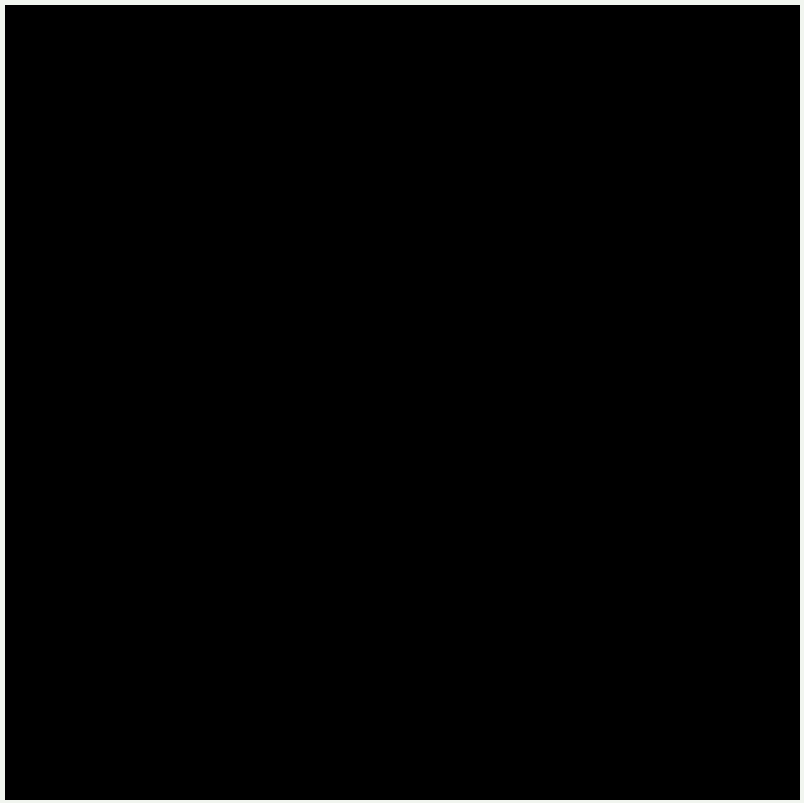
Vamos brincar um pouco

Sem blá, blá, blá. Vamos programar :D.

Desenhar um Quadrado Vazio

```
import cv2
import numpy as np

# Create a black image
image = np.zeros((512,512,3), np.uint8)

# Can we make this in black and white?
image_bw = np.zeros((512,512), np.uint8)

cv2.imshow("Black Rectangle (Color)", image)
cv2.imshow("Black Rectangle (B&W)", image_bw)

cv2.imwrite('../img/blankRectangle.jpg', image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Imagem Quadrado Vazio

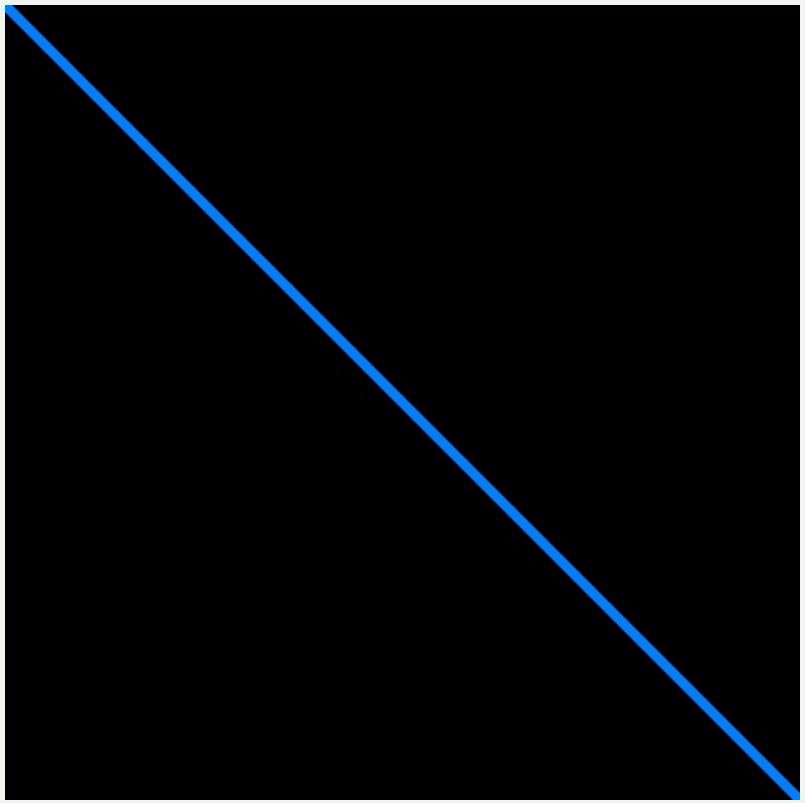


Desenhar uma Linha Azul De Ponta a Ponta

```
import cv2
import numpy as np

# Draw a diagonal blue line of thickness of 5 pixels
image = np.zeros((512,512,3), np.uint8)
cv2.line(image, (0,0), (511,511), (255,127,0), 5)
cv2.imshow("Blue Line", image)
cv2.imwrite('../img/blankBlueLine.jpg', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Imagem Quadrado Linha Azul De Ponta a Ponta

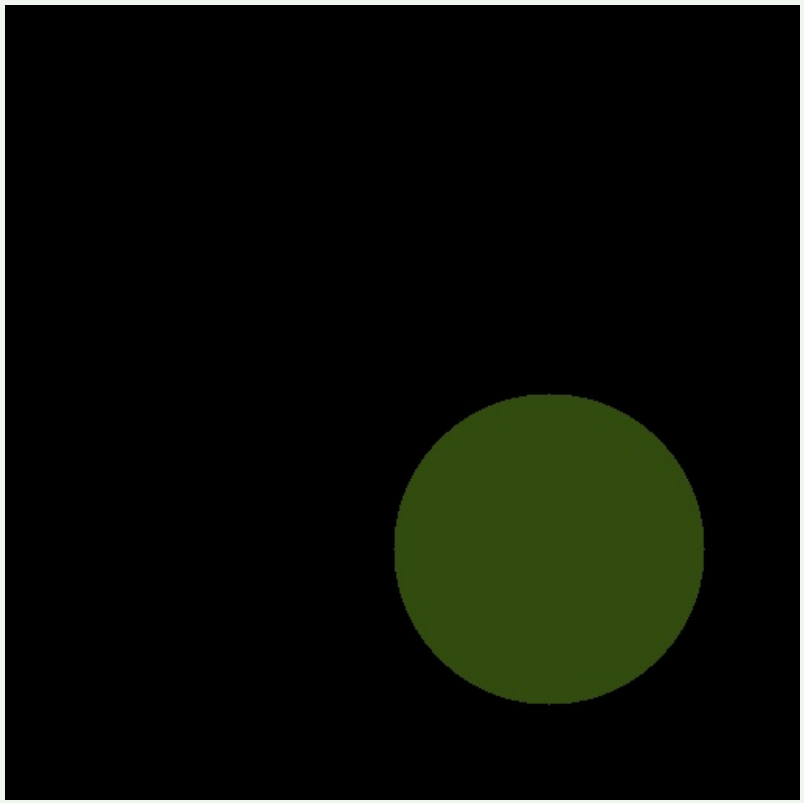


Desenhar um Círculo Preenchido

```
import cv2
import numpy as np

image = np.zeros((512,512,3), np.uint8)

cv2.circle(image, (350, 350), 100, (15,75,50), -1)
cv2.imshow("Circle", image)
cv2.imwrite('../img/circle.jpg', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

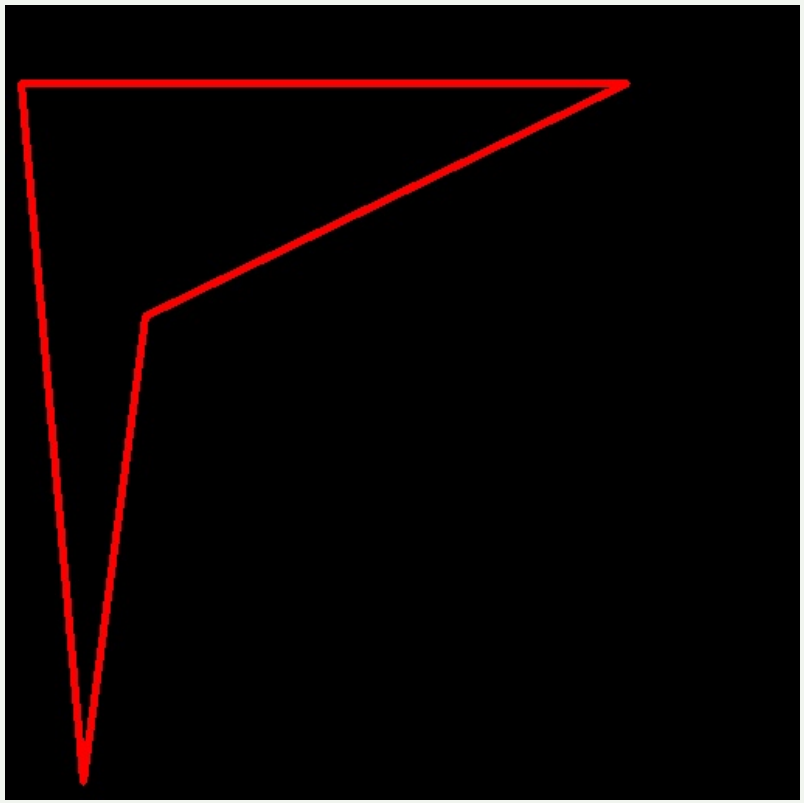
Imagem de um Círculo Preenchido



Desenhar um Polígono

```
import cv2
import numpy as np

image = np.zeros((512,512,3), np.uint8)

# Let's define four points
pts = np.array( [[10,50], [400,50], [90,200],
    [50,500]], np.int32)

# Let's now reshape our points in form  required by
    polylines
pts = pts.reshape((-1,1,2))

cv2.polylines(image, [pts], True, (0,0,255), 3)
cv2.imshow("Polygon", image)
cv2.imwrite('../img/polygon.jpg', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Imagem de um Polígono

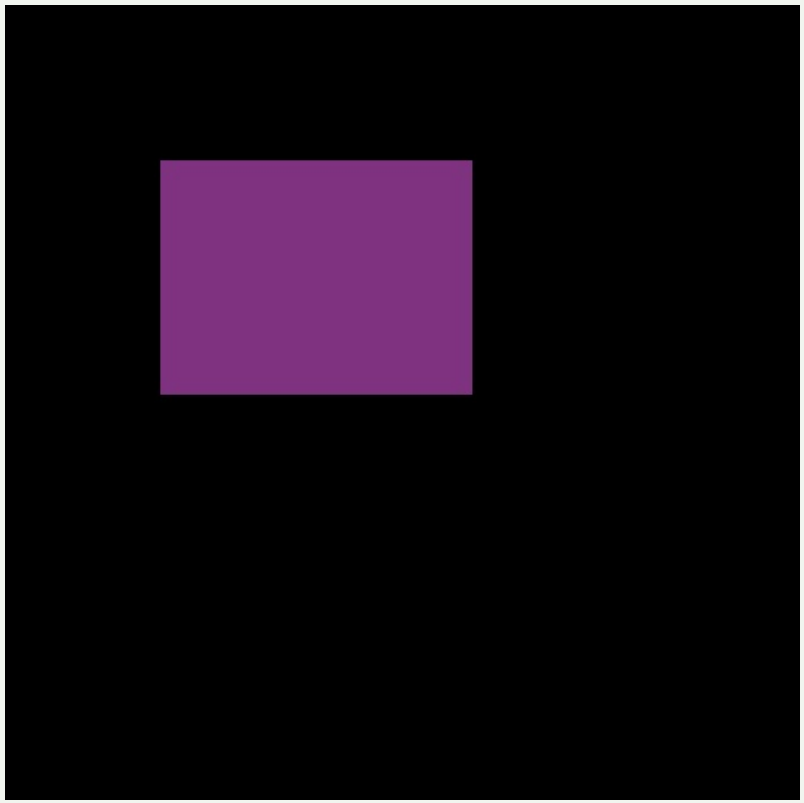


Desenhar um Retângulo

```
import cv2
import numpy as np

# Draw a Rectangle in
image = np.zeros((512,512,3), np.uint8)

cv2.rectangle(image, (100,100), (300,250), (127,50,127)
    , -1)
cv2.imshow("Rectangle", image)
cv2.imwrite('../img/rectangle.jpg', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Imagem de um Retângulo



Escrever um Texto

```
# coding=cp1252
import cv2
import numpy as np

image = np.zeros((512,512,3), np.uint8)

cv2.putText(image, 'Alo IFAL', (75,290), cv2.
    FONT_HERSHEY_COMPLEX, 2, (100,170,0), 3)
cv2.imshow("Alo IFAL Titulo", image)
cv2.imwrite('../img/text.jpg', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Desenho de Texto

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Operações em imagens podem ser realizadas:

- pontualmente nos *pixels*;
- em partes da imagem
  - fixas ou
  - dependendo de algum contexto; e,
- em toda a imagem .

4.1. Operações Pontuais

4.1.1. Operações Aritméticas

4.1.2. Operações Lógicas

4.1. Operações Pontuais

O *pixel*, na posição (x_i,y_i), da imagem resultante depende apenas do *pixel* na imagem original.

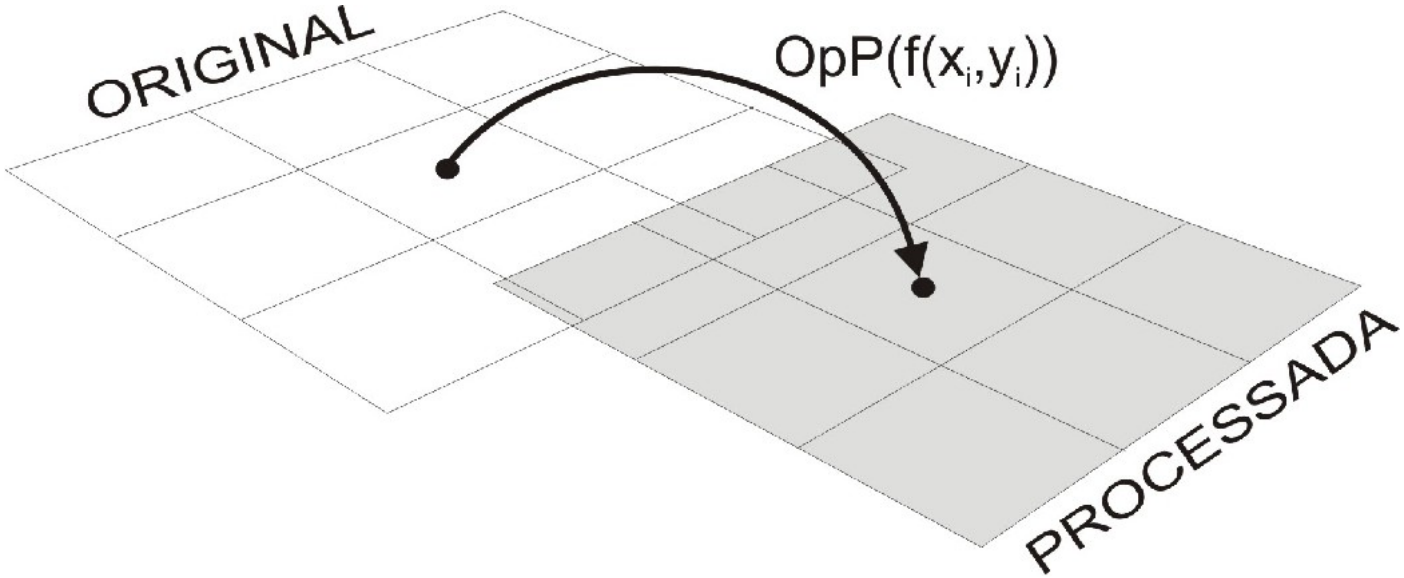


Figura 4.1 – Esquema de operações pontuais em imagens.

Algumas operações eram apenas características de cor ou luminância ja foram vistas no cap 2:

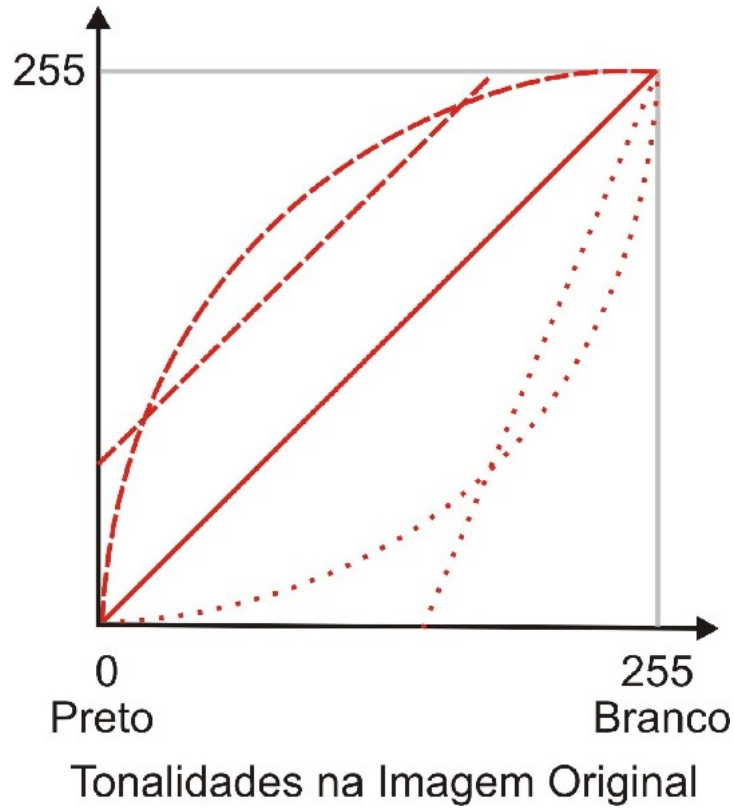
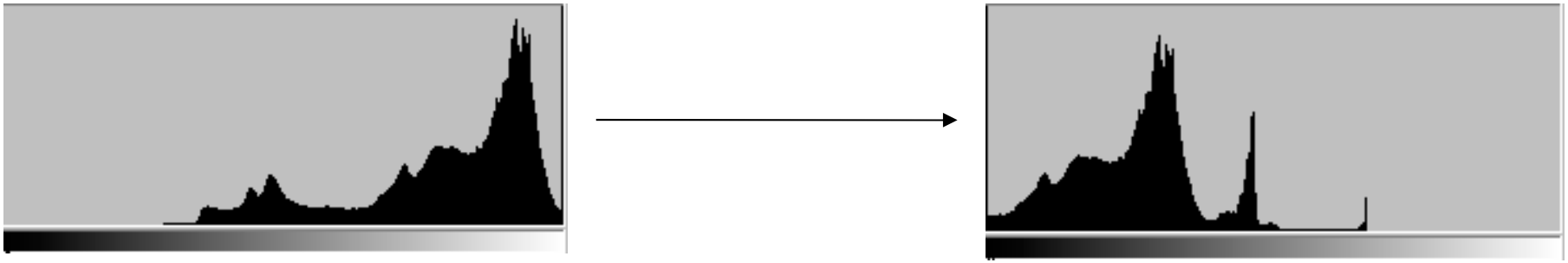


Figura 4.2 – Esquema de mudanças de tons para Imagem em 256 tons de cinza.

O processamento pode levar em consideração dados globais da imagem, como por exemplo, o histograma.



As operações locais *pixel*-a-*pixel* de duas imagens podem ser descritas pela expressão:

$$Z = (X \; OpP \; Y) \qquad (4.3)$$

OpP é um operador qualquer aritmético ou lógico.

4.1.1. Operações Aritméticas

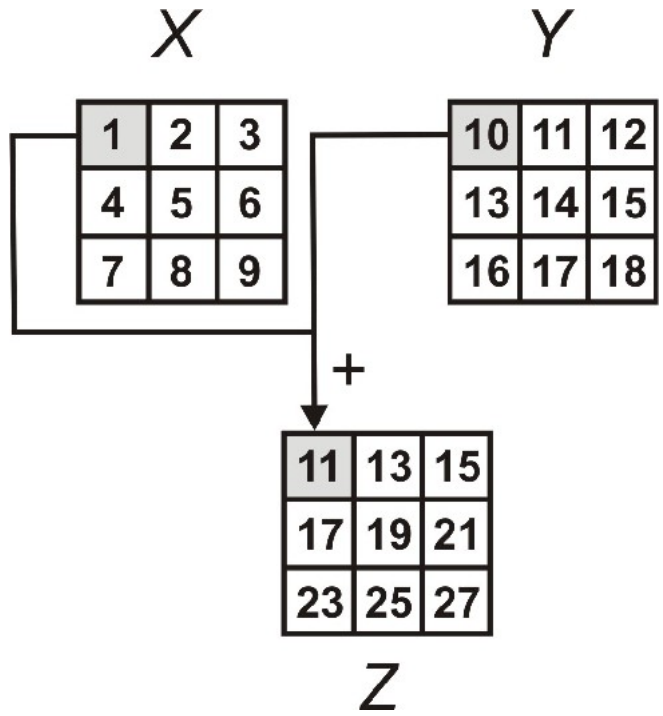
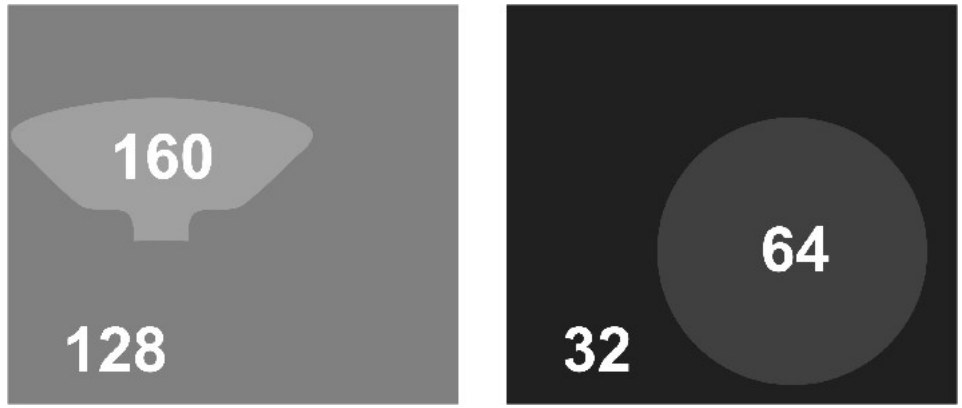
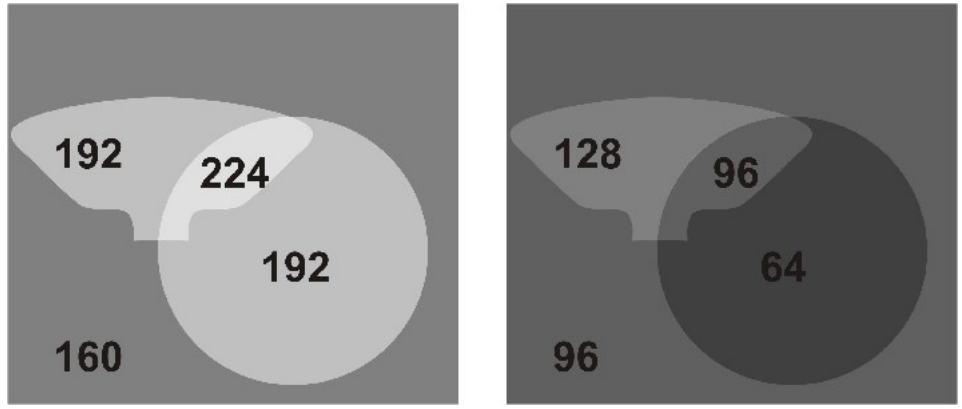


Figura 4.3 – Exemplo de operação aritmética de soma



(a) imagem X (b) imagem Y

Figura 4.4 – Imagens X e Y utilizadas como exemplos.



(a) Z = X+Y (b) Z=X-Y

Figura 4.5 – Exemplos de operações aritméticas com as imagens da Figura 4.4.

Imagens e os Sistemas Lineares

Sistemas Lineares

Os Sistemas Lineares e as operações permitidas por estes são extremamente importantes para o processamento de imagens. Isto se deve ao fato de que a maioria das soluções do mundo real realizada por processamento de imagens são resolvidos por meio de operações em sistemas lineares.

Sistema de Imagens

O Sistema de imagens pode ser descrito por uma operação S que é linear se as duas entradas de valores X e Y e os dois valores escalares a e b atenderem:
$S\{aX + bY\} = aS\{X\} + bS\{Y\}$.

Operações em Imagens

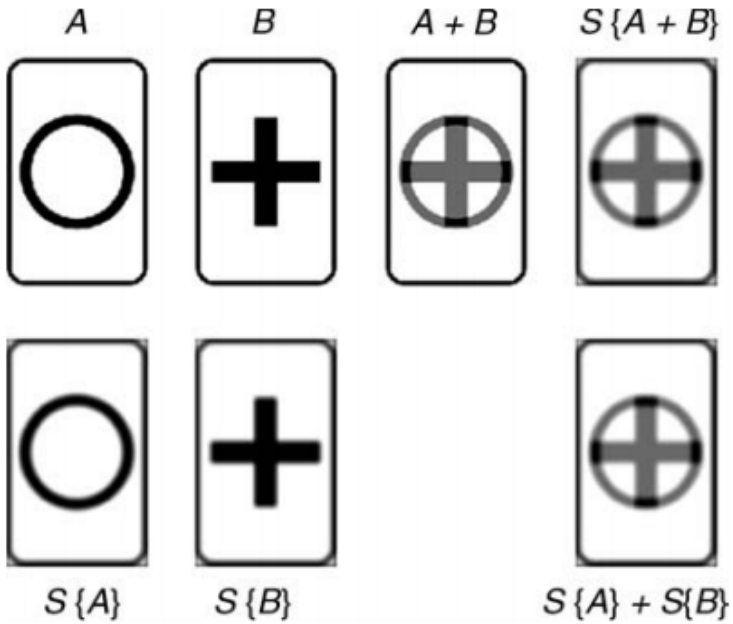


Figure 2.3 Demonstrating the action of a linear operator *S*. Applying the operator (a blur in this case) to the inputs and then linearly combining the results produces the same result as linearly combining them and then applying the operator

Em outras palavras, a aplicação do operador linear em duas entradas deverá então combinar os valores para uma saída ponderada.

Operação Soma

```
import cv2
import numpy as np

imgrectangle = cv2.imread('../img/rectangle.jpg')
imgcircle = cv2.imread('../img/circle.jpg')
imgtext = cv2.imread('../img/text.jpg')

image = imgcircle + imgrectangle + imgtext

cv2.imshow("Imagem", image)
cv2.imwrite('../img/operacaosoma.jpg', image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

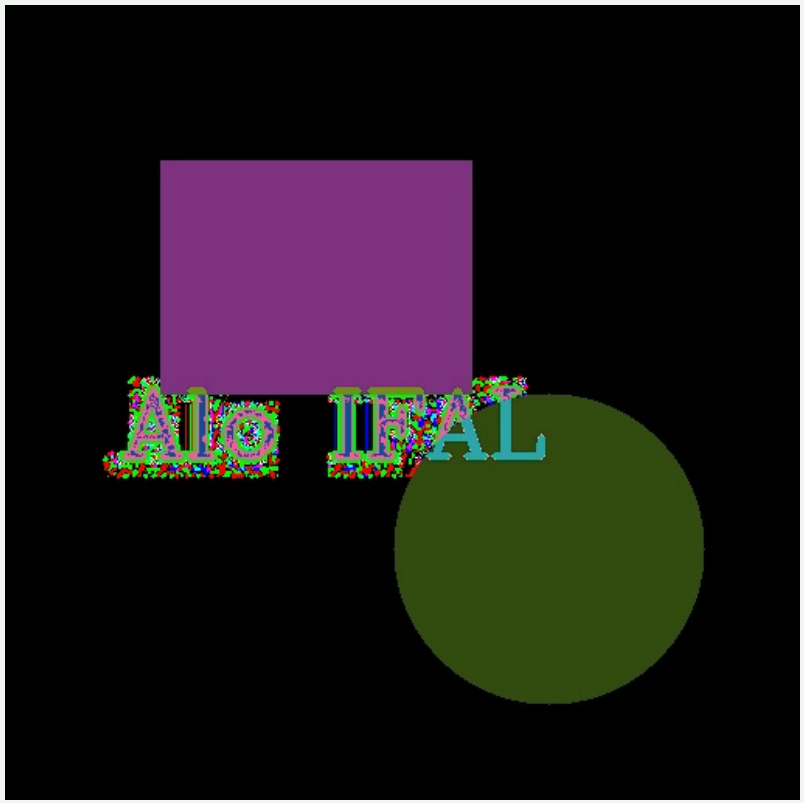
Imagem da Operação Soma

Operação Subtração

```
import cv2
import numpy as np

imgrectangle = cv2.imread('../img/rectangle.jpg')
imgcircle = cv2.imread('../img/circle.jpg')
imgtext = cv2.imread('../img/text.jpg')

image = imgcircle + imgrectangle - imgtext

cv2.imshow("Imagem", image)
cv2.imwrite('../img/operacaosubtracao.jpg', image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Imagem da Operação Subtracao



4.1.2. Operações Lógicas

Figura 4.7 – Objetos X e Y utilizados como modelo.

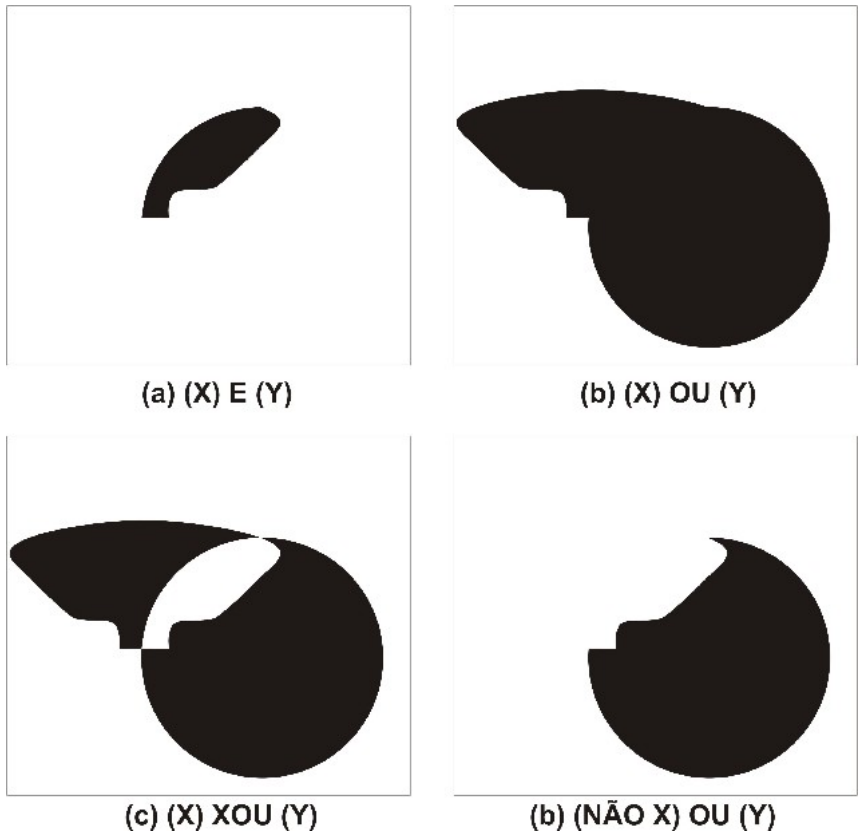


Figura 4.8 - Exemplos de operações lógicas com objetos da figura 4.10.

Equivalentes as operações de União, Interseção e Subtração de conjuntos

Operações Lógicas

Operador BitWise

Vamos agora utilizar as funções *BitWise* para realizar operações lógicas nas imagens.

Desenhando Quadrado e Elipse

```
import cv2
import numpy as np

# Making a sqare
square = np.zeros((300, 300), np.uint8)
cv2.rectangle(square, (50, 50), (250, 250), 255, -2)
cv2.imshow("Square", square)
cv2.imwrite('../img/square.jpg', square)
cv2.waitKey(0)

# Making a ellipse
ellipse = np.zeros((300, 300), np.uint8)
cv2.ellipse(ellipse, (150, 150), (150, 150), 30, 0,
    180, 255, -1)
cv2.imshow("Ellipse", ellipse)
cv2.imwrite('../img/ellipse.jpg', ellipse)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Desenhando Quadrado e Elipse

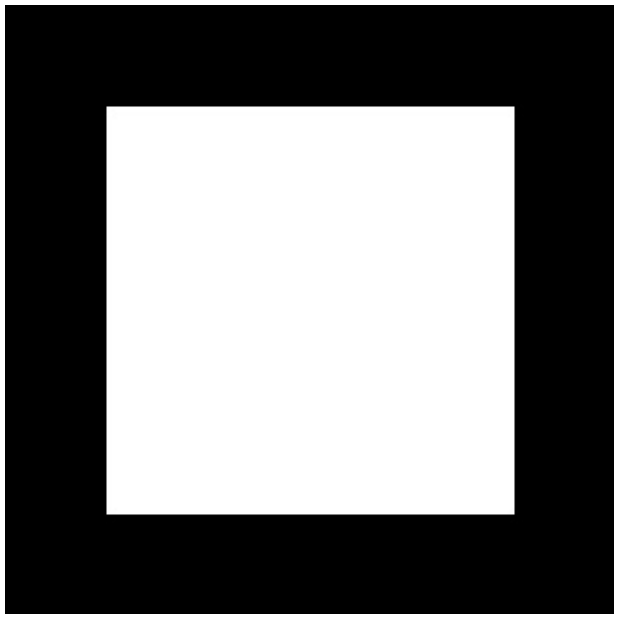


Figura: Quadrado

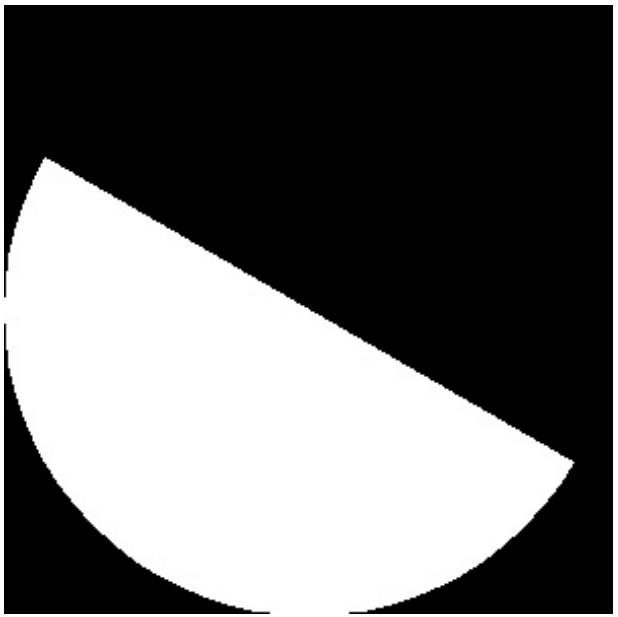


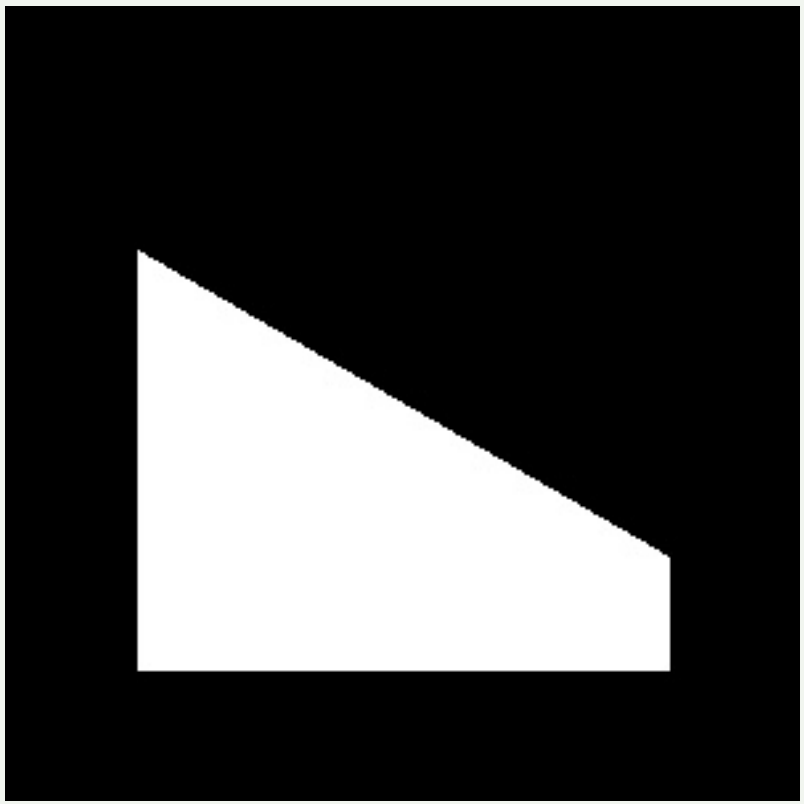
Figura: Elipse

Operação de Lógica - AND

```
import cv2
import numpy as np

square = np.zeros((300, 300), np.uint8)
cv2.rectangle(square, (50, 50), (250, 250), 255, -2)

ellipse = np.zeros((300, 300), np.uint8)
cv2.ellipse(ellipse, (150, 150), (150, 150), 30, 0,
    180, 255, -1)

imageand = cv2.bitwise_and(square, ellipse)
cv2.imshow("AND", imageand)
cv2.imwrite('../img/bitwiseoperationand.jpg', imageand)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Resultado Operação de Lógica - AND

Operação AND

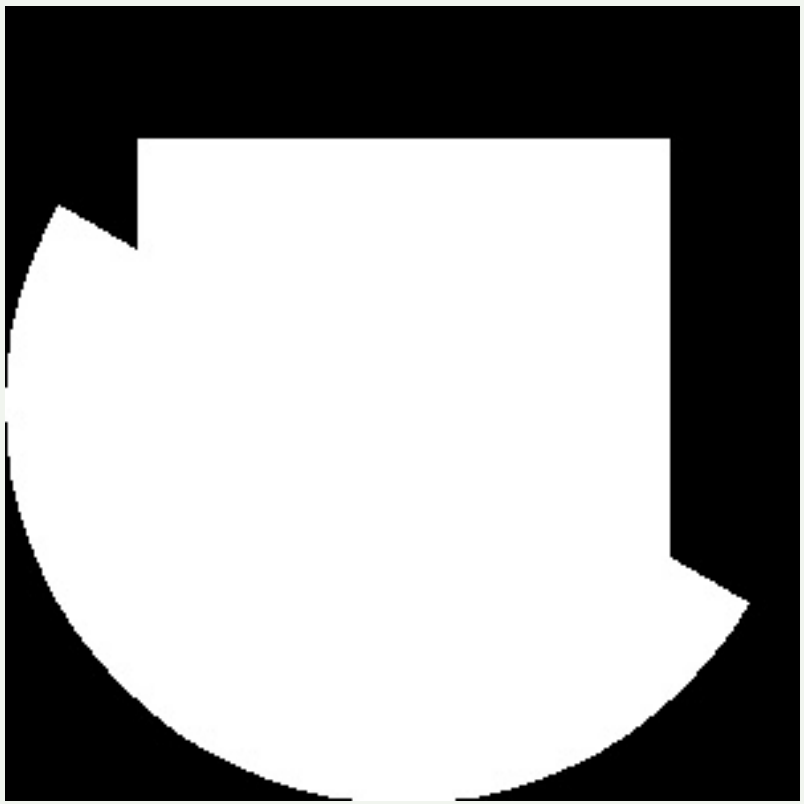


Operação de Lógica - OR

```
import cv2
import numpy as np

square = np.zeros((300, 300), np.uint8)
cv2.rectangle(square, (50, 50), (250, 250), 255, -2)

ellipse = np.zeros((300, 300), np.uint8)
cv2.ellipse(ellipse, (150, 150), (150, 150), 30, 0,
    180, 255, -1)

imageor = cv2.bitwise_or(square, ellipse)
cv2.imshow("OR", imageor)
cv2.imwrite('../img/bitwiseoperationor.jpg', imageor)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Resultado Operação de Lógica - OR

Operação OR

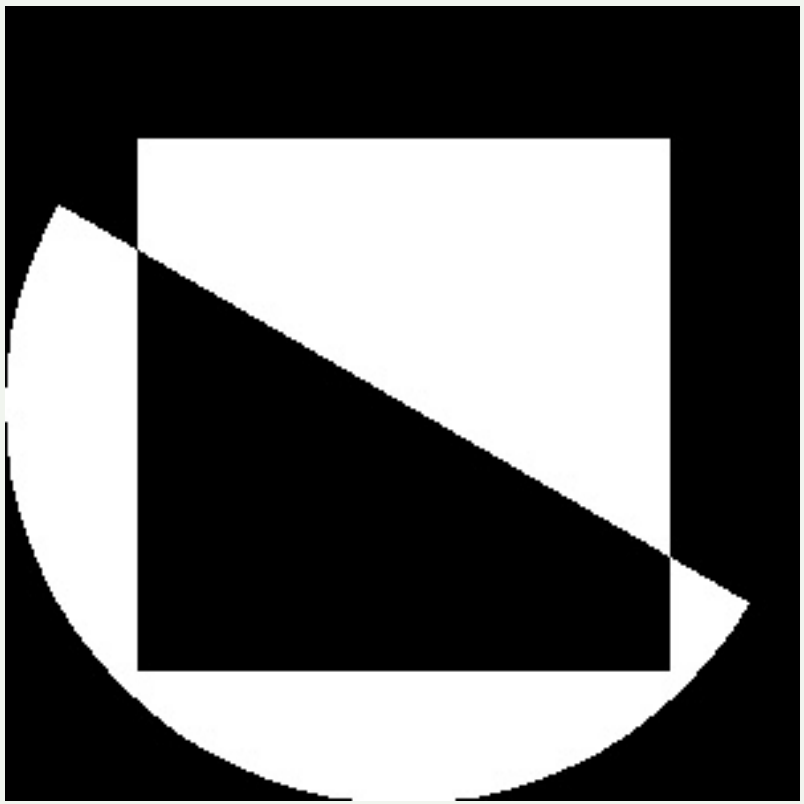


Operação de Lógica - XOR

```
import cv2
import numpy as np

square = np.zeros((300, 300), np.uint8)
cv2.rectangle(square, (50, 50), (250, 250), 255, -2)

ellipse = np.zeros((300, 300), np.uint8)
cv2.ellipse(ellipse, (150, 150), (150, 150), 30, 0,
    180, 255, -1)

imagexor = cv2.bitwise_xor(square, ellipse)
cv2.imshow("XOR", imagexor)
cv2.imwrite('../img/bitwiseoperationxor.jpg', imagexor)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Resultado Operação de Lógica - XOR

Operação XOR



Operação de Lógica - NOT

```
import cv2
import numpy as np

square = np.zeros((300, 300), np.uint8)
cv2.rectangle(square, (50, 50), (250, 250), 255, -2)

ellipse = np.zeros((300, 300), np.uint8)
cv2.ellipse(ellipse, (150, 150), (150, 150), 30, 0,
    180, 255, -1)

imagenot = cv2.bitwise_not(square, ellipse)
cv2.imshow("NOT", imagenot)
cv2.imwrite('../img/bitwiseoperationnot.jpg', imagenot)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Resultado Operação de Lógica - NOT

Operação NOT

Bibliografia

- Solomon C., Breckon T. **Fundamentals of Digital Image Processing.. A Practical Approach with Examples in Matlab** (Wiley-Blackwell, 2011)(ISBN 9780470844724)(en)(355s)
- A. Conci, E. Azevedo e F.R. Leta - **Computação Gráfica: volume 2 (Processamento e Análise de Imagens Digitais)**, Campus/Elsevier. 2008 - ISBN 85-352-1253-3.
- Open CV Tutorials
- Tutorial de ColorSpace `http://docs.opencv.org/trunk/df/d9d/tutorial_py_colorspaces.html`

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

DÚVIDAS ?

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS