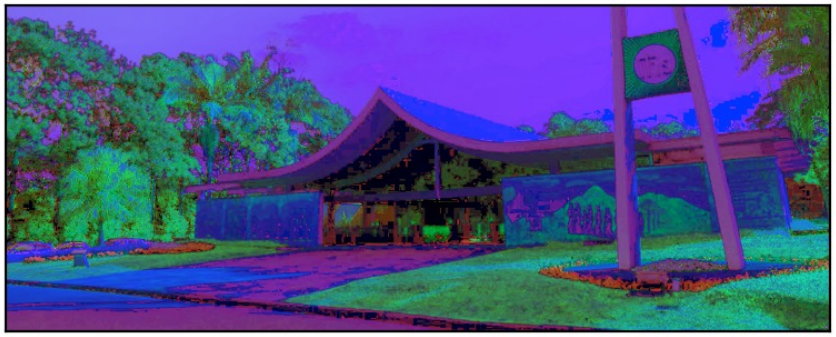
Aula Laboratório
Visão Computacional

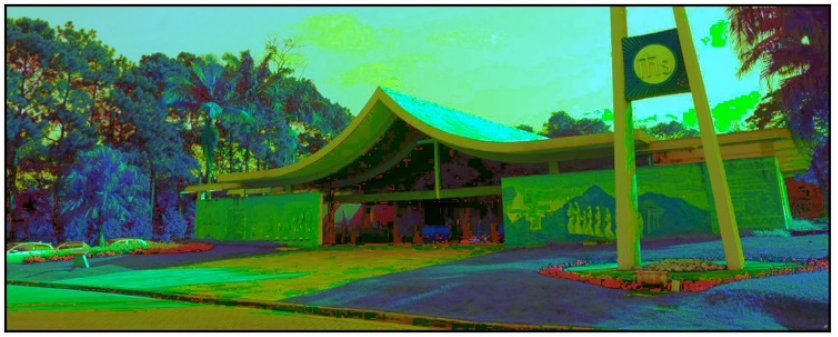
Espaço de Cores

```
# Carregans imagem
img = cv2.imread('./imagens/Burano01.jpg')
```

```
# Convertendo espaço de cores
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
img_hls = cv2.cvtColor(img, cv2.COLOR_BGR2HLS)
```

Espaço de Cores





Convertendo para Preto e Branco

```
#Convertendo para preto e branco (RGB -> Gray Scale -> BW)
img_gray = cv2.cvtColor(img_blur, cv2.COLOR_RGB2GRAY)
a = img_gray.max()
_, thresh = cv2.threshold(img_gray, a/2+100, a,cv2.THRESH_BINARY_INV)
```

```
double cv::threshold ( InputArray  src,
                       OutputArray dst,
                       double      thresh,
                       double      maxval,
                       int         type
                     )
```

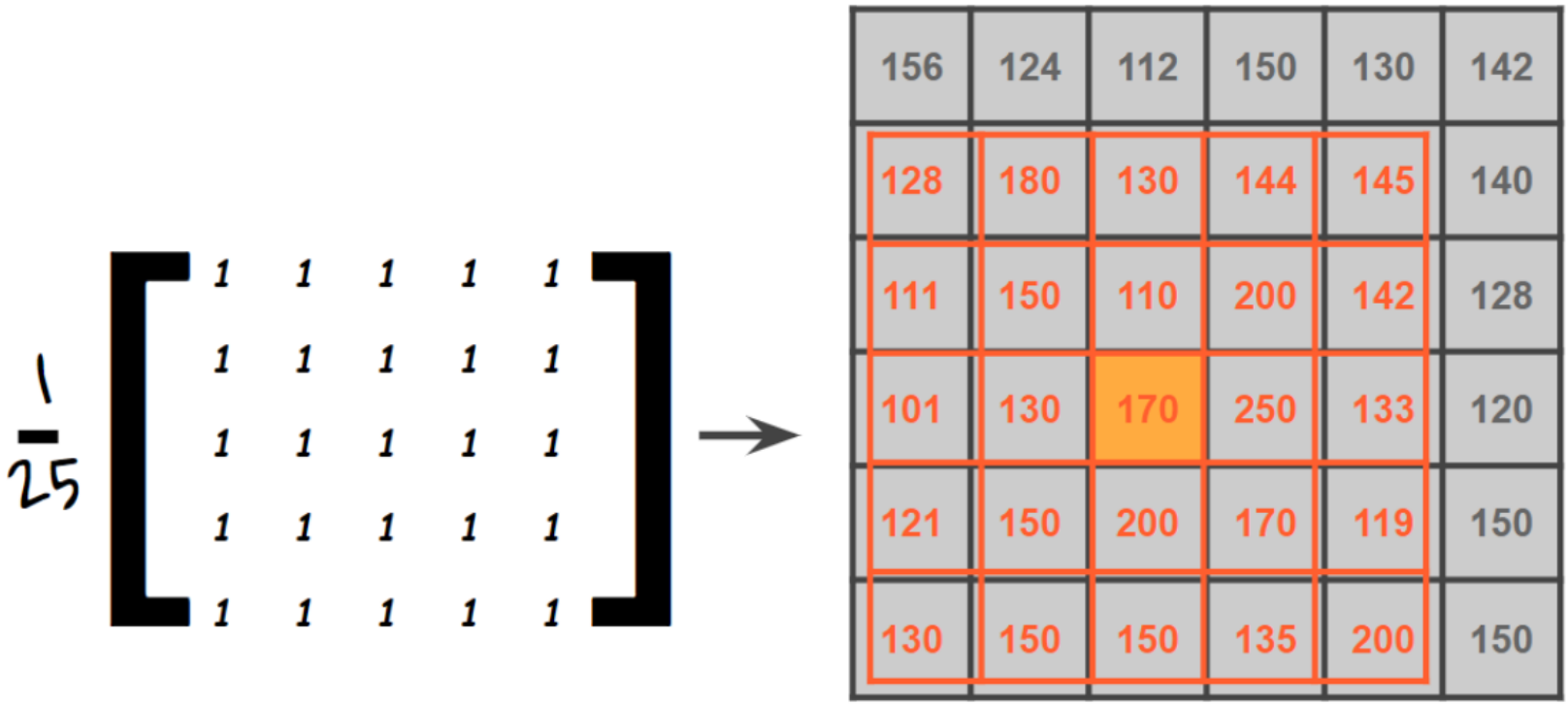
Python:

```
cv.threshold( src, thresh, maxval, type[, dst] ) -> retval, dst
```

Parameters

src	input array (multiple-channel, 8-bit or 32-bit floating point).
dst	output array of the same size and type and the same number of channels as src.
thresh	threshold value.
maxval	maximum value to use with the **THRESH_BINARY** and **THRESH_BINARY_INV** thresholding types.
type	thresholding type (see **ThresholdTypes**).

Kernel

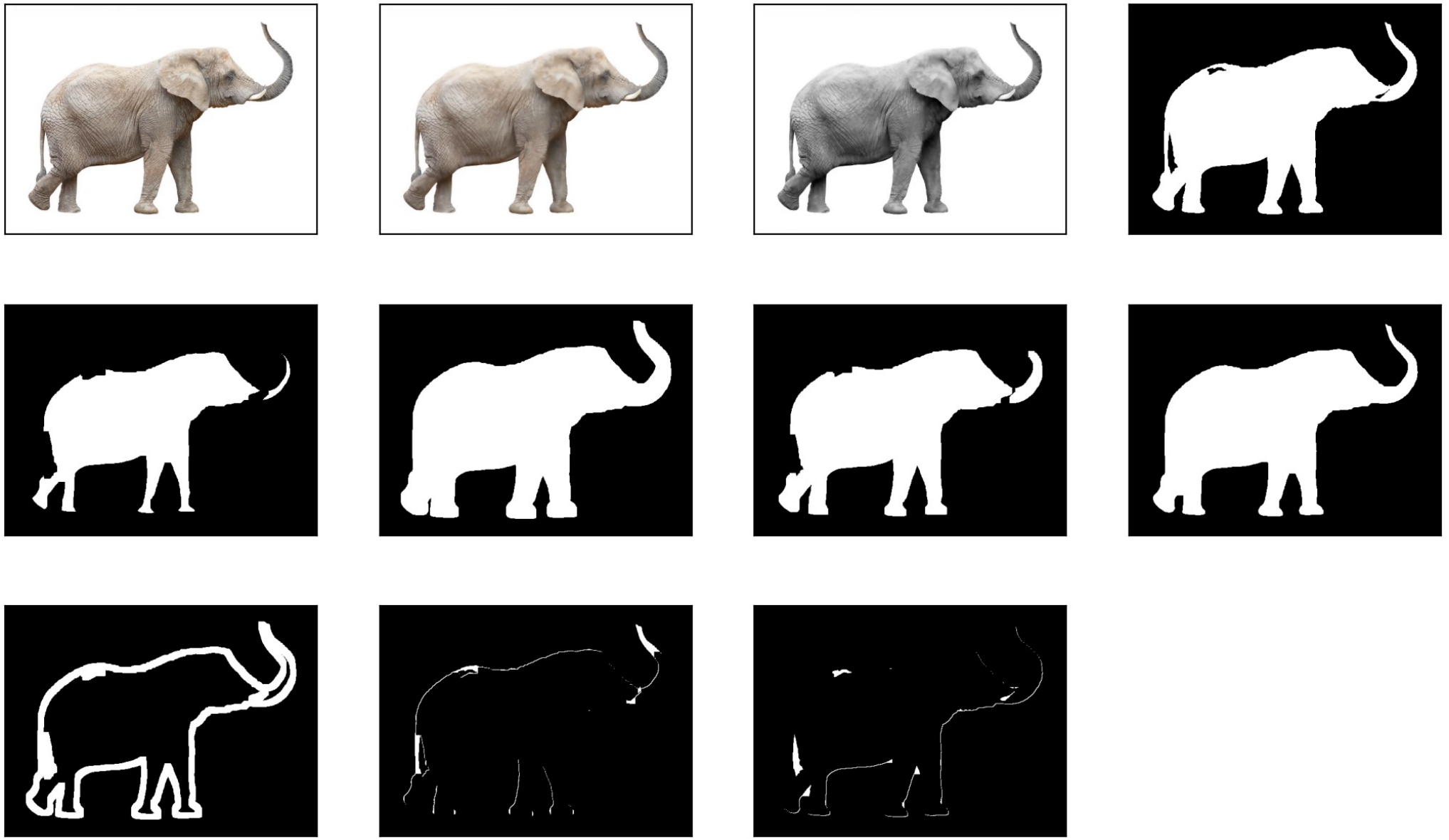



```
#preparando o "kernel"
kernel = np.ones((5,5), np.uint8)
```

Operadores Morfológicos

```
#operadores Morfologicos
img_dilate = cv2.dilate(thresh,kernel,iterations = 1)
img_erode = cv2.erode(thresh,kernel,iterations = 1)
img_open = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
img_close = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)
img_grad = cv2.morphologyEx(thresh, cv2.MORPH_GRADIENT, kernel)
img_tophat = cv2.morphologyEx(thresh, cv2.MORPH_TOPHAT, kernel)
img_blackhat = cv2.morphologyEx(thresh, cv2.MORPH_BLACKHAT, kernel)
```

Operadores Morfológicos

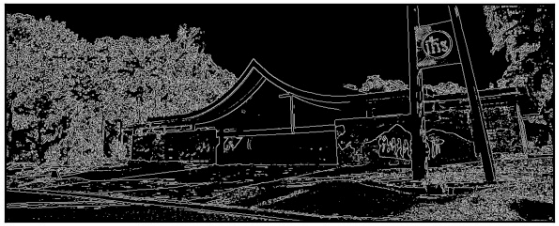


Detecção de bordas

```
# Detecção borda com Canny (sem blurry)
edges_gray = cv2.Canny(image=img_gray, threshold1=a/2, threshold2=a/2)
# Detecção borda com Canny (com blurry)
edges_blur = cv2.Canny(image=img_blur, threshold1=a/2, threshold2=a/2)
```

Detecção de bordas





Atividade Laboratório #1