atividade-4

November 10, 2017

In [17]:
```
import os

import cv2
import numpy as np
from scipy.ndimage.filters import gaussian_filter

import pylab as plt
%matplotlib inline
```

In [18]:
```
path = os.getcwd() + os.sep
#path += '../../db_images/png/captcha.png'
path += '../../db_images/jpeg/captcha.jpeg'
```

In [19]:
```
thresholds = []

img = cv2.imread(path)
thresholds.append(('Original', img))

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
thresholds.append(('Gray', gray))

sobel_x = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=1)
thresholds.append(('sobel x ksize=1', sobel_x))

sobel_y = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=1)
thresholds.append(('sobel y ksize=1', sobel_y))

sobel_x = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=3)
thresholds.append(('sobel x ksize=3', sobel_x))

sobel_y = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=3)
thresholds.append(('sobel y ksize=3', sobel_y))

sobel_x = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=5)
thresholds.append(('sobel x ksize=5', sobel_x))

sobel_y = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=5)
thresholds.append(('sobel y ksize=5', sobel_y))
```

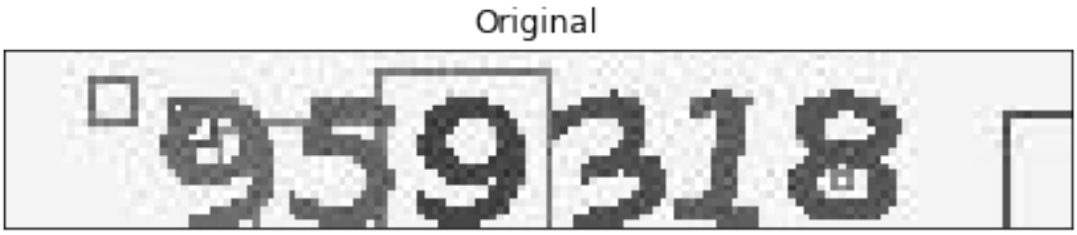
```
        edges = cv2.Canny(gray, 100, 255)
        thresholds.append(('canny lower=100, upper=255', edges))

        edges = cv2.Canny(gray, 135, 255)
        thresholds.append(('canny lower=135, upper=255', edges))

        edges = cv2.Canny(gray, 71, 255)
        thresholds.append(('canny lower=71, upper=255', edges))

        edges = cv2.Canny(gray, 100, 200)
        thresholds.append(('canny lower=100, upper=200', edges))

        gaussian_blur5 = cv2.GaussianBlur(gray, (5, 5), 0.)
        gaussian_blur3 = cv2.GaussianBlur(gray, (3, 3), 0.)
        dog = gaussian_blur5 - gaussian_blur3
        thresholds.append(('DoG left(ksize=(5, 5), std=0.) right(ksize=(3, 3), std=0.)', dog))

        gaussian_blur7 = cv2.GaussianBlur(gray, (7, 7), 0.)
        gaussian_blur5 = cv2.GaussianBlur(gray, (5, 5), 0.)
        dog = gaussian_blur7 - gaussian_blur5
        thresholds.append(('DoG left(ksize=(7, 7), std=0.) right(ksize=(5, 5), std=0.)', dog))

        gaussian_blur_left = cv2.GaussianBlur(gray, (3, 3), 0.5)
        gaussian_blur_right = cv2.GaussianBlur(gray, (3, 3), 0.)
        dog = gaussian_blur_left - gaussian_blur_right
        thresholds.append(('DoG ksize=(3, 3), left(std=0.5) right(std=0.)', dog))

        gaussian_blur_left = cv2.GaussianBlur(gray, (5, 5), 0.)
        gaussian_blur_right = cv2.GaussianBlur(gray, (5, 5), 0.5)
        dog = gaussian_blur_left - gaussian_blur_right
        thresholds.append(('DoG ksize=(5, 5), left(std=0.) right(std=0.5)', dog))
```

In [20]:
```
    def DoG(image, k=200, gamma=1):
        s1 = 0.5
        s2 = s1*k
        gauss1 = gaussian_filter(image, s1)
        gauss2 = gamma*gaussian_filter(image, s2)
        return gauss1 - gauss2

    def XDoG(image, epsilon=0.05):
        phi = 10
        difference = DoG(image, 200, 0.98)/255
        diff = difference*image

        for i in range(0, len(difference)):
            for j in range(0, len(difference[0])):
                if difference[i][j] >= epsilon:
```

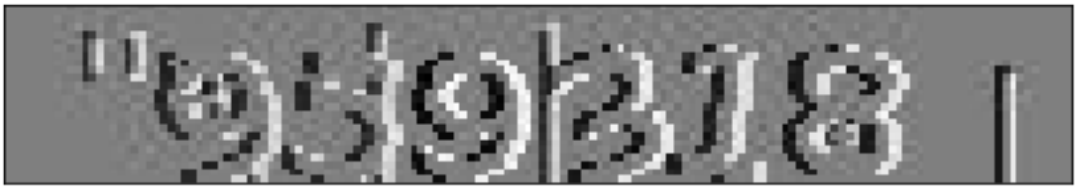
```
                    difference[i][j] = 1
                else:
                    ht = np.tanh(phi*(difference[i][j] - epsilon))
                    difference[i][j] = 1 * ht
        return difference*255
```

In [21]:
```
xdog = XDoG(gray, epsilon=0.01)
thresholds.append(('XDoG epsilon=0.01', xdog))

xdog = XDoG(gray, epsilon=0.05)
thresholds.append(('XDoG epsilon=0.05', xdog))

xdog = XDoG(gray, epsilon=0.1)
thresholds.append(('XDoG epsilon=0.1', xdog))
```

In [22]:
```
for title, image in thresholds:
    plt.figure()
    plt.title(title)
    plt.xticks([]), plt.yticks([])
    if 'original'.upper() is title.upper():
        plt.imshow(image)
    else:
        plt.imshow(image, cmap='gray')
    plt.tight_layout()
```



sobel x ksize=1



sobel y ksize=1

sobel x ksize=3

sobel y ksize=3

sobel x ksize=5

sobel y ksize=5

canny lower=100, upper=255

canny lower=135, upper=255

canny lower=71, upper=255

canny lower=100, upper=200

DoG left(ksize=(5, 5), std=0.) right(ksize=(3, 3), std=0.)

DoG left(ksize=(7, 7), std=0.) right(ksize=(5, 5), std=0.)

DoG ksize=(3, 3), left(std=0.5) right(std=0.)

DoG ksize=(5, 5), left(std=0.) right(std=0.5)

XDoG epsilon=0.01

XDoG epsilon=0.05

XDoG epsilon=0.1