Ejercicios de Visión Artificial

Ejercicio 1. Suavizado de Imágenes

OpenCV - Python

Ejercicio 1. Suavizado de Imágenes (I)

- Para el filtrado de la media vamos a emplear la función: ***cv2.blur***
  - Parámetros de Entrada:
    - imagen de entrada
    - tamaño de la máscara de convolución. Por ejemplo: (5,5)
  - Parámetros de Salida:
    - imagen de salida suavizada con el filtro de la media
  - Ejemplo de llamada a la función:
    - ***blur = cv2.blur(img,(5,5))***
- Para el filtrado gaussiano vamos a emplear la función: ***cv2.GaussianBlur***
  - Se emplea de forma similar a la función blur pero tiene uno o dos parámetros más de entrada para indicar las desviaciones estándar en x e y (si sólo ponemos un cero la función las calcula automáticamente a partir del tamaño de la máscara)
  - Ejemplo de llamada a la función:
    - ***blur = cv2.GaussianBlur(img,(5,5),0)***

Ejercicio 1. Suavizado de Imágenes (II)

- Tareas a realizar en el ejercicio:
  - Abrir imagen **numeros.png**
  - Suavizar la imagen con distintos tamaños para la máscara empleando el **filtro de la media**
  - Suavizar la imagen con distintos tamaños para la máscara empleando el **filtro gaussiano**
  - Repetir el proceso con la imágenes **marte.bmp** y **patron_texto.bmp**

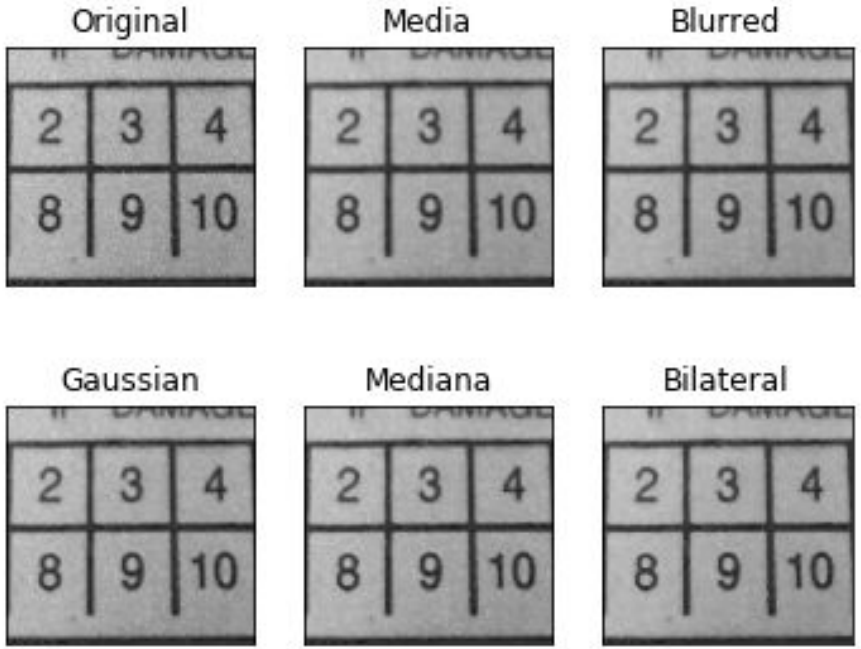
- Más información en:
  - https://docs.opencv.org/3.4/d4/d13/tutorial_py_filtering.html

Ejercicio 1. Suavizado de Imágenes (III)

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('numeros.png')
kernel = np.ones((5,5),np.float32)/25

dst = cv2.filter2D(img,-1,kernel)
blur = cv2.blur(img,(5,5))
plt.subplot(131),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(132),plt.imshow(dst),plt.title('Media')
plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()


gaussianBlur = cv2.GaussianBlur(img,(5,5),0)
median = cv2.medianBlur(img,5)
bilateral = cv2.bilateralFilter(img,9,75,75)
plt.subplot(131),plt.imshow(gaussianBlur),plt.title('Gaussian')
plt.xticks([]), plt.yticks([])
plt.subplot(132),plt.imshow(median),plt.title('Mediana')
plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(bilateral),plt.title('Bilateral')
plt.xticks([]), plt.yticks([])
plt.show()

```

Ejercicio 1. Suavizado de Imágenes (IV)

```
In [33]: runfile('/Users/arroyo/suavizado.py', wdir='/Users/arroyo')
```



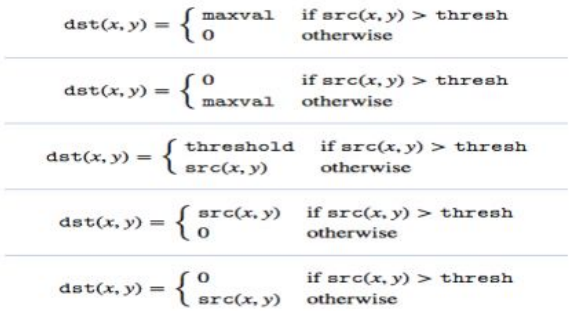
Ejercicios de Visión Artificial

Ejercicio 2. Umbralización de Imágenes

OpenCV - Python

Ejercicio 2. Umbralización de Imágenes (I)

- Segmentar imágenes empleando un método de umbral fijo (***cv2.threshold***) y un método de umbral adaptativo (***cv2.adaptiveThreshold***):
  - Función: *cv2.threshold*
  - Parámetros de Entrada:
    - imagen en niveles de gris sobre la que se va a realizar la umbralización
    - valor del umbral (0..255)
    - valor de salida para representar los píxeles que estén por encima del umbral
    - modo de umbralización
      - cv.THRESH_BINARY
      - cv.THRESH_BINARY_INV
      - cv.THRESH_TRUNC
      - cv.THRESH_TOZERO
      - cv.THRESH_TOZERO_INV
    - Parámetros de Salida:
      - valor booleano
      - imagen de salida binaria una vez umbralizada (pero en niveles de gris)



Ejercicio 2. Umbralización de Imágenes (II)

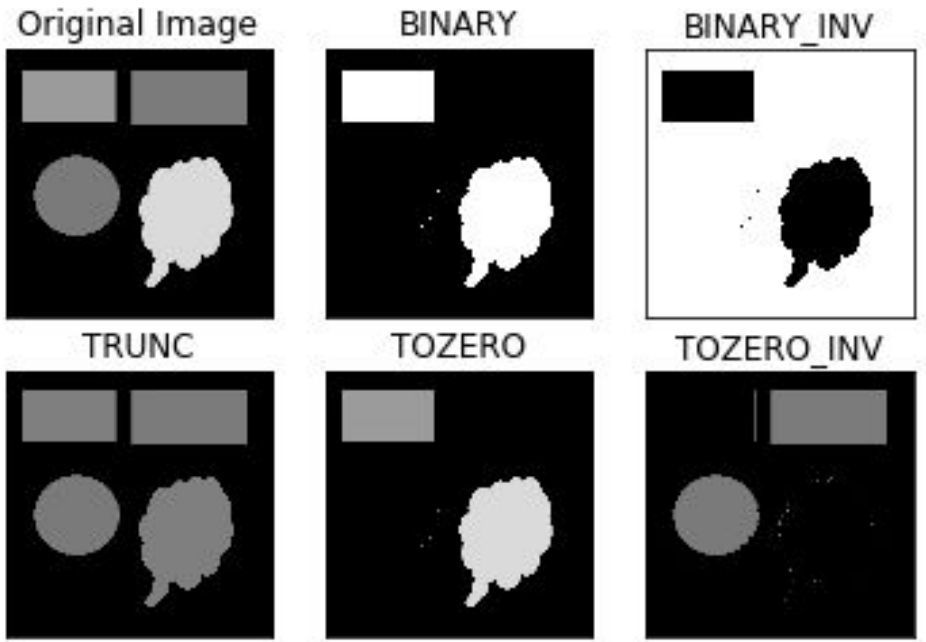
- Tareas a realizar en el ejercicio:
  - Abrir imagen **formas_gris.png**
  - Binarizar imagen mediante umbralización fija empleando la función ***cv2.threshold***
  - Analizar la imagen para establecer el mejor umbral para segmentar todas las figuras presentes en la imagen
  - Probar con distintos valores de umbral y con los distintos modos de umbralización para conseguir separar las 4 formas que aparecen en la imagen formas.png
  - Hacer lo mismo pero empleando la función ***cv2.adaptiveThreshold***
  - Hacer lo mismo pero empleando el método OTSU para imágenes bimodales:
  - Emplear la imagen **numeros.png** en lugar de la imagen **formas.png**
  - Ampliar información en:
    - https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html

Ejercicio 2. Umbralización de Imágenes (III)

```
1  import cv2
2  from matplotlib import pyplot as plt
3
4  img = cv2.imread('formas_gris.jpg')
5
6  ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
7  ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
8  ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
9  ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
10 ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)
11 titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
12 images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
13 for i in xrange(6):
14     plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
15     plt.title(titles[i])
16     plt.xticks([]),plt.yticks([])
17 plt.show()
18
```

Ejercicio 2. Umbralización de Imágenes (IV)

```
In [24]: runfile('/Users/arroyo/umbralizado.py', wdir='/Users/arroyo')
```



Ejercicio 2. Umbralización de Imágenes (V)

```
1  import cv2
2  from matplotlib import pyplot as plt
3
4  img = cv2.imread('formas_gris.jpg', cv2.IMREAD_GRAYSCALE)
5  img = cv2.medianBlur(img,5)
6
7  ret,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
8  th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,11,2)
9  th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,11,2)
10 titles = ['Original Image', 'Global Threshold (v = 127)',
11             'Adaptive Mean Threshold', 'Adaptive Gaussian Threshold']
12 images = [img, th1, th2, th3]
13 for i in xrange(4):
14     plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
15     plt.title(titles[i])
16     plt.xticks([]),plt.yticks([])
17 plt.show()
18
```

Ejercicio 2. Umbralización de Imágenes (VI)

```
In [32]: runfile('/Users/arroyo/umbralizadoAdaptativo.py', wdir='/Users/arroyo')
```