
Universidad
Tecmilenio®

Universidad Tecmilenio

Visión computacional

Operadores locales

Introducción

Los operadores locales son los más importantes en procesamiento de imágenes. El valor de un punto depende de la vecindad local del mismo, es decir, de los valores de otros puntos de la imagen o de su entorno. Un ejemplo de este tipo de operadores es el filtrado utilizado para difuminar, suavizar o mejorar una imagen.

En este tema conocerás:

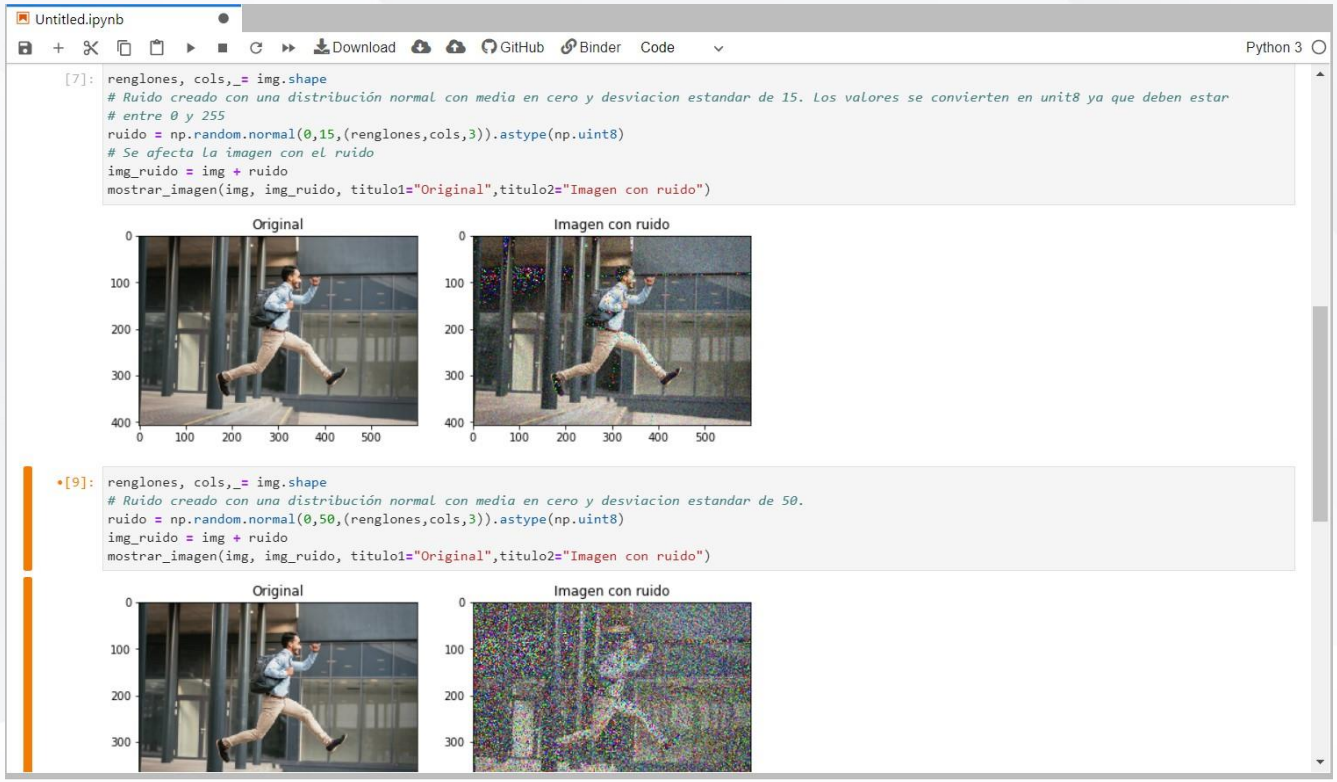
- Distintos tipos de operaciones de filtrado en imágenes.
- Transformaciones morfológicas en imágenes.
- El uso de OpenCV en Python para realizar filtrado y transformaciones morfológicas de imágenes.

Explicación

Los operadores lineales incluyen a la convolución y correlación.
El filtrado mejora una imagen eliminando el ruido existente en ella.

Formalmente, un operador lineal se distribuye sobre una suma ponderada de imágenes, sean las imágenes f y g y los escalares a y b, el operador lineal L es: $L(af + bg) = aLf + bLg$

El **ruido** es provocado, generalmente, por una mala cámara o compresión de la imagen y da lugar a imágenes borrosas. El objetivo del filtrado es obtener imágenes más nítidas.



Explicación

Los **filtros de suavizado** promedian los pixeles en el vecindario de un píxel y se conocen como filtros pasabajos.
El **kernel** (matriz utilizada para operaciones de enfoque, realce, detección de bordes, etc.) promedia los puntos en un vecindario para obtener el suavizado.

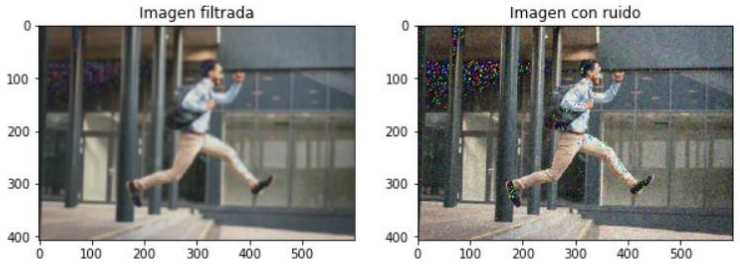
Untitled.ipynb

Download GitHub Binder Code Python 3

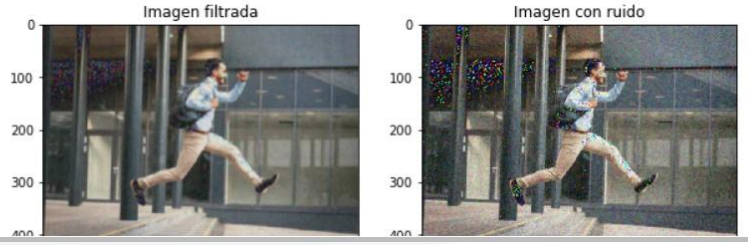
```
[16]: # Kernel de 6x6 con valores de 1/36
      kernel = np.ones((6,6))/36

[19]: img_filtrada = cv.filter2D(src=img_ruido, ddepth=-1, kernel=kernel)

[20]: mostrar_imagen(img_filtrada, img_ruido,titulo1="Imagen filtrada",titulo2="Imagen con ruido")
```




```
[22]: # Kernel de 4x4 con valores de 1/16
      kernel = np.ones((4,4))/16
      img_filtrada = cv.filter2D(src=img_ruido, ddepth=-1, kernel=kernel)
      mostrar_imagen(img_filtrada, img_ruido,titulo1="Imagen filtrada",titulo2="Imagen con ruido")
```



Explicación

El **desenfoque Gaussiano** se logra con la función *GaussianBlur* que convoluciona la imagen original con el kernel gaussiano especificado; filtra el ruido, pero conserva mejor los bordes. Los parámetros del filtro son la imagen de entrada, el tamaño del kernel y las desviaciones estándares del kernel en la dirección del eje X y del eje Y (sigmaX y sigmaY, respectivamente). Un valor mayor de sigma dará una imagen más borrosa.

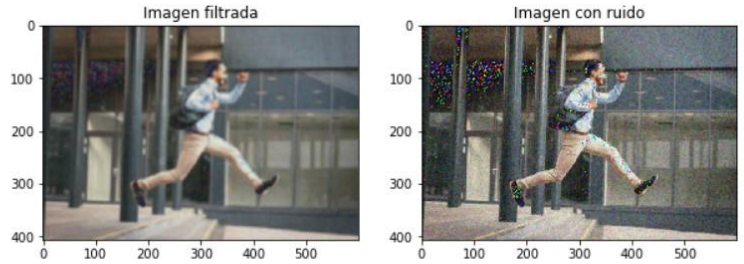
Untitled.ipynb

Download GitHub Binder Code Python 3

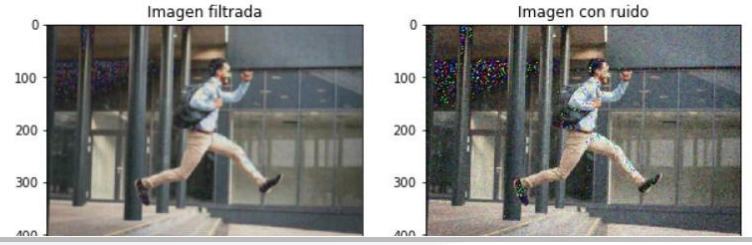
```
[16]: # Kernel de 6x6 con valores de 1/36
      kernel = np.ones((6,6))/36
```

```
[19]: img_filtrada = cv.filter2D(src=img_ruido, ddepth=-1, kernel=kernel)
```

```
[20]: mostrar_imagen(img_filtrada, img_ruido,titulo1="Imagen filtrada",titulo2="Imagen con ruido")
```




```
[22]: # Kernel de 4x4 con valores de 1/16
      kernel = np.ones((4,4))/16
      img_filtrada = cv.filter2D(src=img_ruido, ddepth=-1, kernel=kernel)
      mostrar_imagen(img_filtrada, img_ruido,titulo1="Imagen filtrada",titulo2="Imagen con ruido")
```



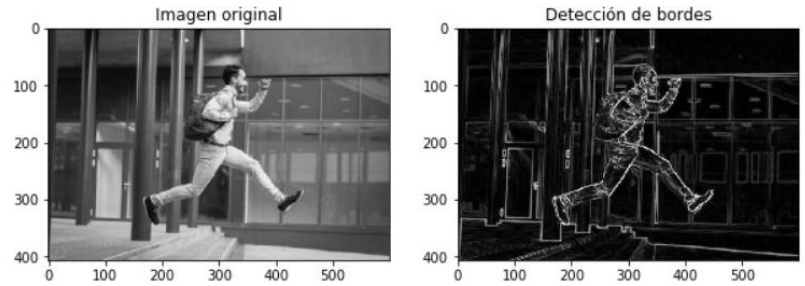
Explicación

Los bordes de una imagen se pueden identificar donde cambian de intensidad los pixeles. El gradiente de una función facilita la obtención de la razón de cambio y es posible obtener una aproximación con la convolución. Uno de los métodos para calcular el gradiente es el **detector de bordes de Sobel** que combina varias convoluciones y calcula la magnitud del resultado final.

Untitled.ipynb

Download GitHub Binder Code Python 3

```
[55]: img_grises = cv.imread('933517366.jpg', cv.IMREAD_GRAYSCALE)
#Filtrado con GaussianBlur con un kernel de 3 x 3
img_grises = cv.GaussianBlur(img_grises,(3,3),sigmaX=0.1,sigmaY=0.1)
ddepth = cv.CV_16S
# Filtrado de la imagen en la dirección del eje X
grad_x = cv.Sobel(src=img_grises, ddepth=ddepth, dx=1, dy=0, ksize=3)
# Filtrado de la imagen en la dirección del eje Y
grad_y = cv.Sobel(src=img_grises, ddepth=ddepth, dx=0, dy=1, ksize=3)
# Ajuste de valores entre 0 y 255
abs_grad_x = cv.convertScaleAbs(grad_x)
abs_grad_y = cv.convertScaleAbs(grad_y)
# Derivadas en la dirección de eje X y Y
grad = cv.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)
mostrar_imagen(img_grises , grad, titulo1="Imagen original",titulo2="Detección de bordes")
```




```
[ ]:
```

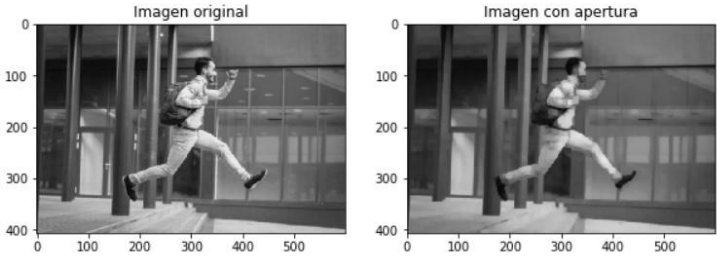
Explicación

Hay dos tipos de operadores morfológicos que se utilizan para eliminar ruido en imágenes: erosión y dilatación. La erosión adelgaza ciertas áreas de la imagen; la dilatación, por el contrario, expande ciertas regiones de la imagen. Cuando se tiene una erosión y después una dilatación se conoce como **apertura** y se utiliza para eliminar el ruido en las imágenes. El **cierre** es una dilatación seguida de una erosión.

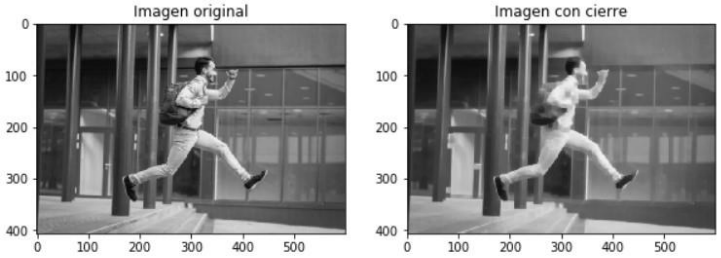
Untitled.ipynb

Download GitHub Binder Code Python 3

[87]:
```
img_grises = cv.imread('933517366.jpg', cv.IMREAD_GRAYSCALE)
kernel = np.ones((7,7),np.uint8)
apertura = cv.morphologyEx(img_grises, cv.MORPH_OPEN, kernel)
mostrar_imagen(img_grises, apertura, titulo1="Imagen original",titulo2="Imagen con apertura")
```



[88]:
```
img_grises = cv.imread('933517366.jpg', cv.IMREAD_GRAYSCALE)
kernel = np.ones((7,7),np.uint8)
apertura = cv.morphologyEx(img_grises, cv.MORPH_CLOSE, kernel)
mostrar_imagen(img_grises, apertura, titulo1="Imagen original",titulo2="Imagen con cierre")
```



Actividad

Se han definido y mostrado algunos filtros lineales. Investiga cuales son los filtros no lineales, aplícalos a los ejemplos anteriores y compara los resultados. ¿Qué diferencias existen entre filtros lineales o no lineales?

Cierre

Los operadores locales se dividen en operadores lineales y morfológicos, los primeros se utilizan comúnmente para mejorar imágenes y los segundos para identificar la forma o silueta de las imágenes.

Estas transformaciones se realizan en el dominio espacial de la imagen, es decir, en el plano que contiene a los pixeles de la imagen.

Universidad Tecmilenio

Visión computacional

Estructura local

Introducción

Al observar imágenes, incluso aquellas que contienen representaciones abstractas de cosas que no se han visto antes, la atención se centra en ciertas características locales: puntos, líneas, bordes, cruces, esquinas, etc.

Es posible describir una imagen como un conjunto de estos “prototipos de imagen” o elementos locales, incluso agruparlos con la finalidad de reconocerlos y utilizarlos como base del proceso de percepción.

En este tema conocerás:
- El concepto de parche en procesamiento de imágenes.
- El suavizado gaussiano en imágenes.
- La librería *SciPy* para el suavizado de imágenes.
- El concepto de espacio de escala y sus aplicaciones.

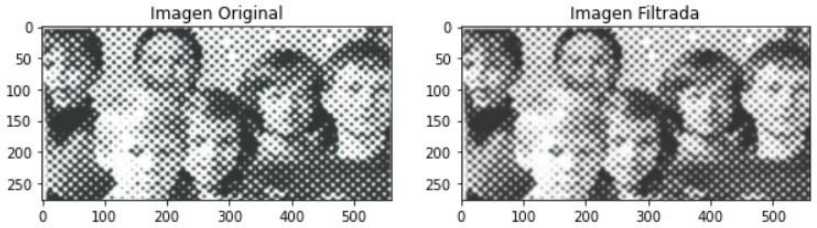
Explicación

Una aplicación utilizada para mejorar imágenes donde se selecciona la escala del vecindario local (puntos dentro de una región de la imagen) para enfocar regiones donde se pretende identificar los bordes (estructura local), es el **suavizado gaussiano**. Esta técnica asocia el tamaño del vecindario con el parámetro sigma. El filtro detecta los pixeles donde hay cambios de intensidad de forma brusca (bordes) y los suaviza (reduce la variación de ese cambio de intensidad).

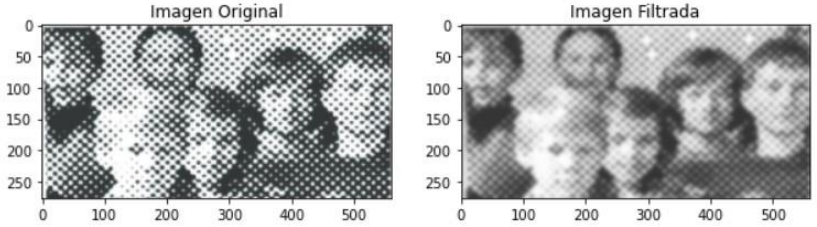
Launcher | Untitled1.ipynb

Code | git | Run as Pipeline | Python

[55]:
```
from scipy.ndimage import gaussian_filter
img_res = gaussian_filter(img, sigma=1)
mostrar_imagen(img, img_res,titulo1="Imagen Original",titulo2="Imagen Filtrada")
```



[56]:
```
from scipy.ndimage import gaussian_filter
img_res = gaussian_filter(img, sigma=3)
mostrar_imagen(img, img_res,titulo1="Imagen Original",titulo2="Imagen Filtrada")
```

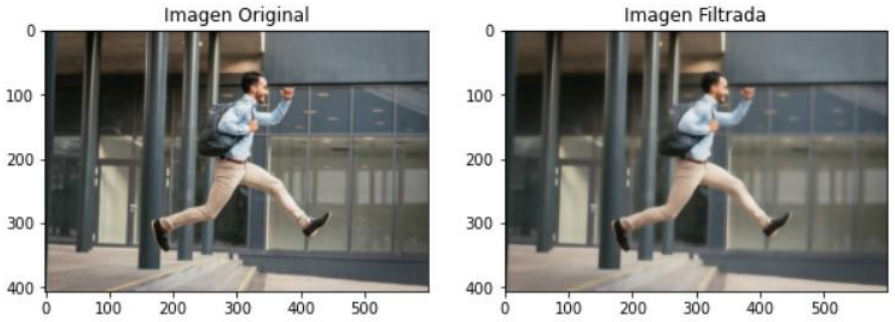


[ ]:

Explicación

El filtrado bilateral utiliza el filtro gaussiano para remover el ruido en una imagen sin perder detalle de los bordes.

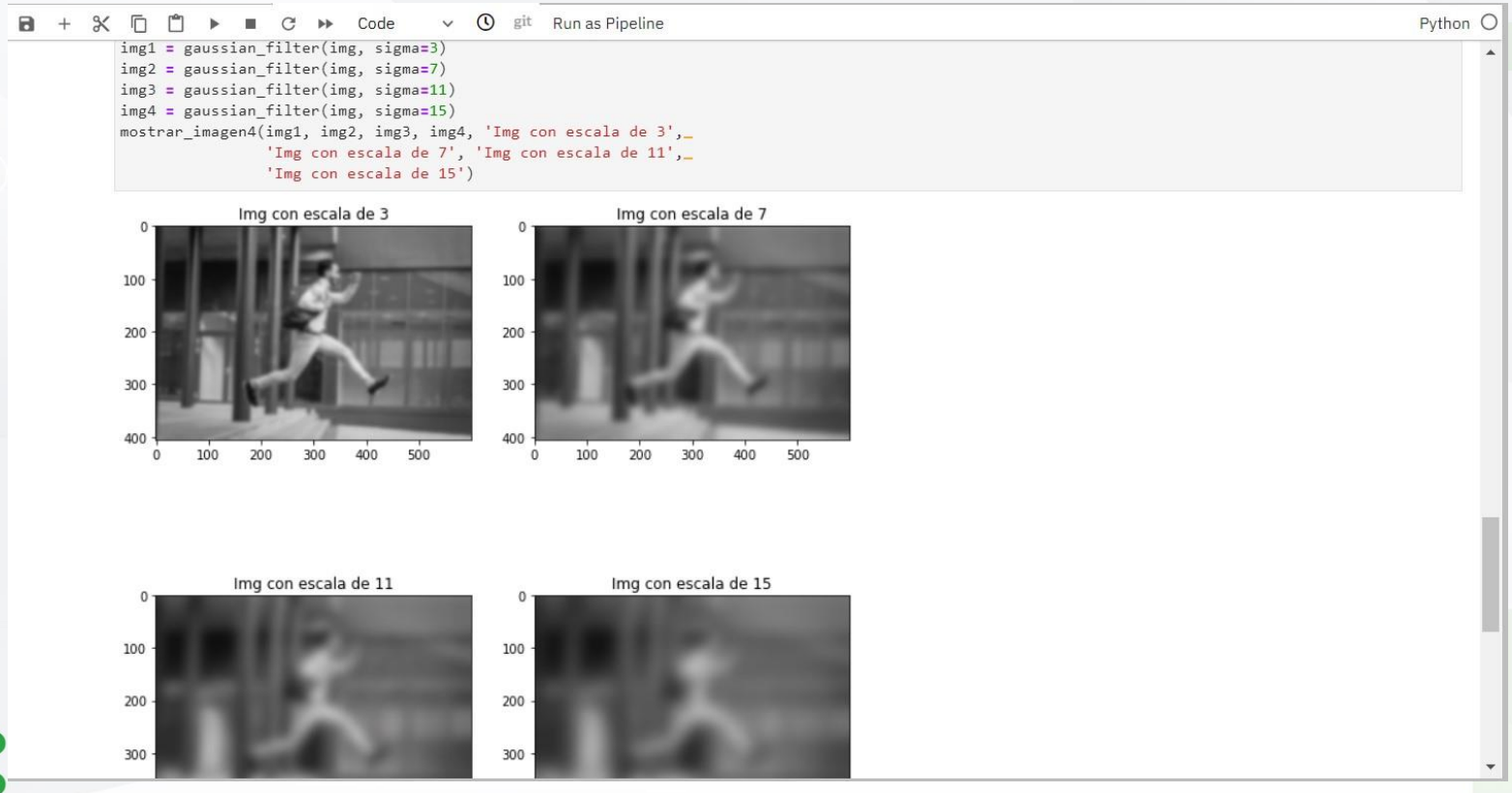
```
[59]: img = cv.imread('933517366.jpg')
      img_res = cv.bilateralFilter(img, 11, 120, 120)
      mostrar_imagen(img, img_res, titulo1="Imagen Original", titulo2="Imagen Filtrada")
```




```
[ ]:
```

Explicación

El espacio de escala es un marco de referencia de la representación de una imagen (o señal) que permite la representación de múltiples escalas de forma simultánea, es decir, generar una familia de imágenes derivadas de una particular a varios niveles de escala.
Al construir una representación en el espacio de escalas, es importante hacerlo a partir de una escala fina hacia una mayor, con la finalidad de simplificar los detalles de las imágenes. Los espacios de escala se pueden utilizar en tareas de segmentación de imagen, suavizamiento de imagen, suavizamiento manteniendo bordes de imágenes, borrado de detalles en imágenes y detección de esquinas.



Actividad

Busca una imagen de un rostro sin ningún tipo de maquillaje o retoque digital. Aplica un filtro gaussiano con diferentes valores de sigma ¿Qué mejoras observas? ¿Se parece a algún filtro que actualmente ocupan los smartphone al tomar fotos de rostros?

Cierre

Los filtros de suavizado gaussiano son comúnmente utilizados para reducir el ruido en imágenes digitalizadas. Hay distintas aplicaciones donde se usan y el resultado que se obtiene depende de la desviación estándar a lo largo de las dimensiones de la imagen.

Los objetos que observamos alrededor de nosotros son significativos a cierta escala. Es posible observar de forma adecuada un terrón de azúcar sobre una mesa, pero si ese mismo terrón de azúcar se observará desde la vía láctea simplemente no existiría. Esta característica multiescala es común en todos los objetos y el espacio de escalas intenta replicar este comportamiento en las imágenes digitalizadas.