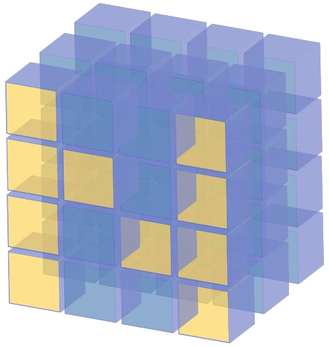
PROCESAMIENTO DIGITAL DE IMAGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Módulos principales



NumPy

matplotlib

OpenCV

pip install opencv-python

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

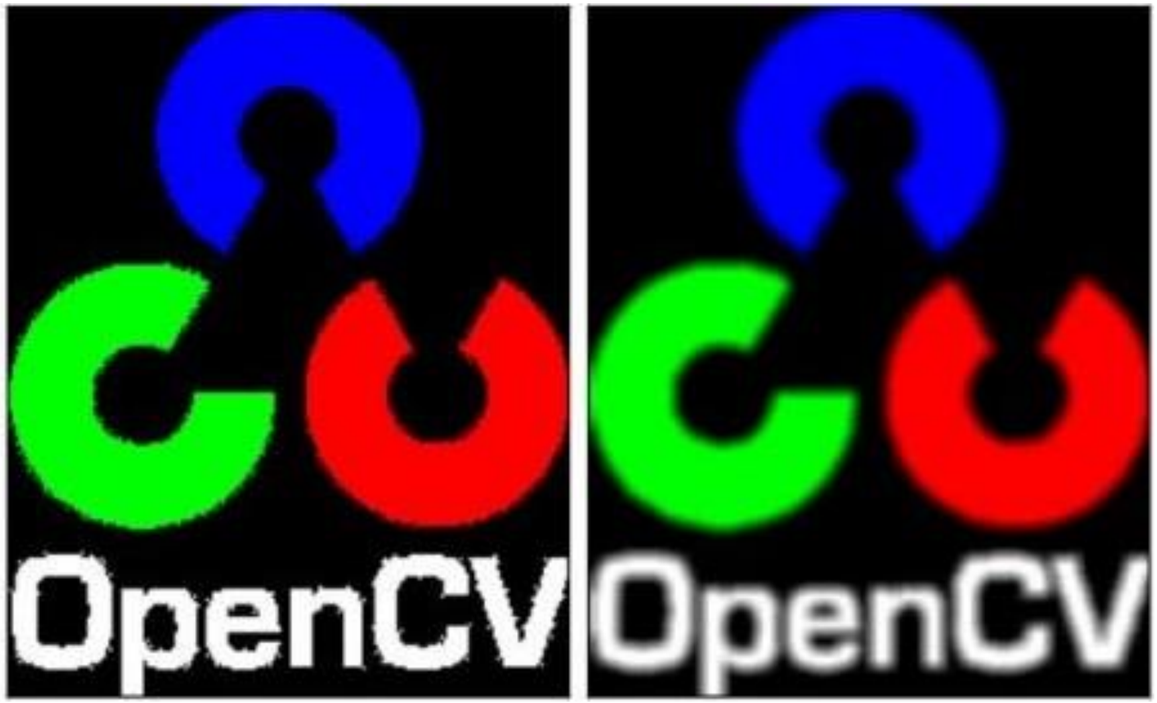
TEMAS QUE VEREMOS HOY

- Suavizado
- Convolución 2D
- Difuminado de imágenes
- Filtro de Mediana
- Filtro Gaussiano
- Transformaciones Morfológicas

PROCESAMIENTO DIGITAL DE IMÁGENES

SUAVIZADO

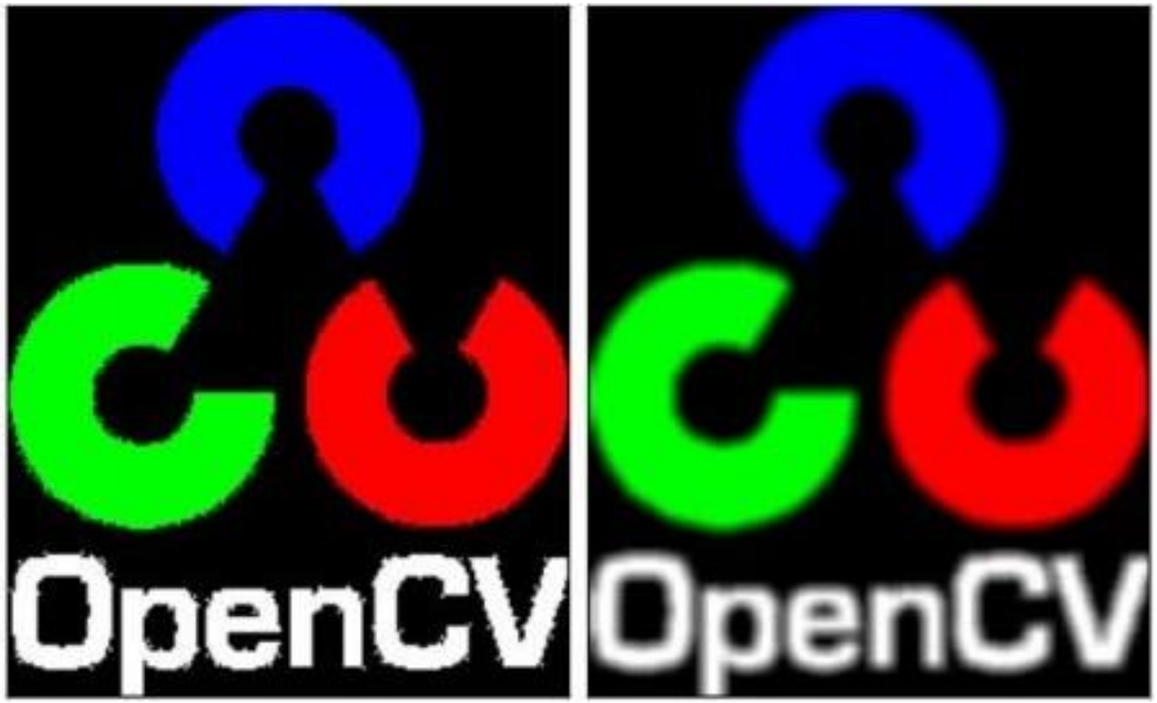
- Consiste en capturar patrones, tal que se deja fuera el ruido
- Reduce el contenido de los bordes de los objetos, permitiendo la transición entre los colores
- Se utilizan los filtros de imágenes para conseguir esto



PROCESAMIENTO DIGITAL DE IMÁGENES

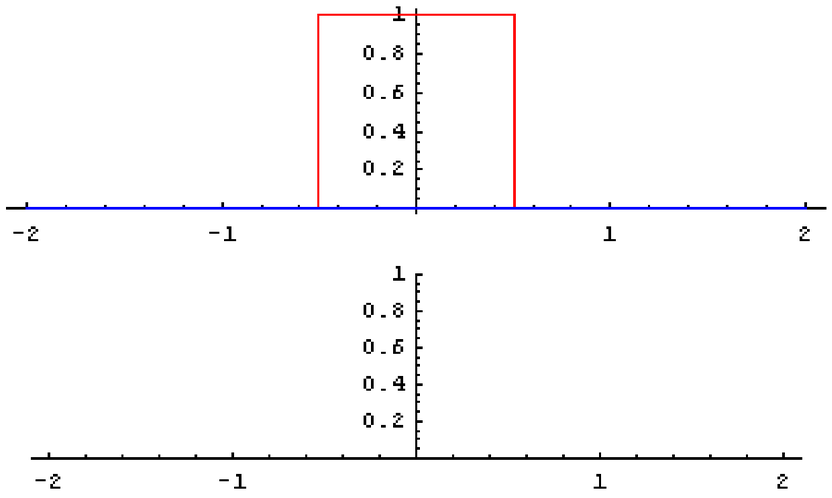
SUAVIZADO

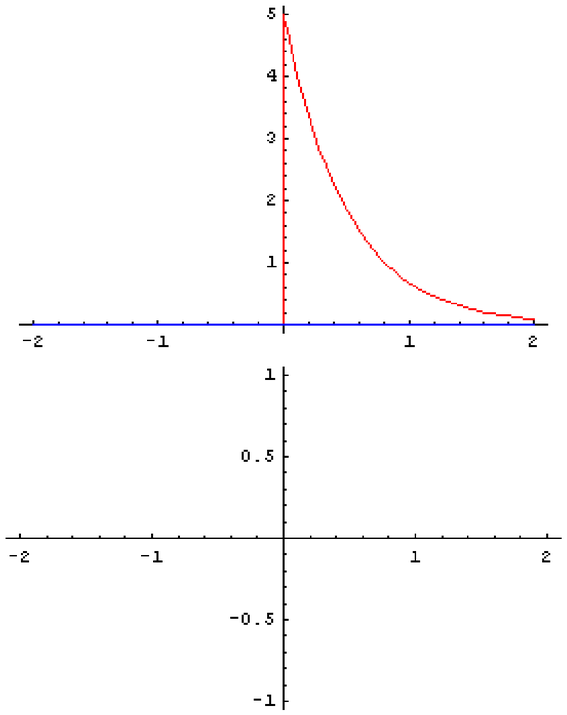
- Si podemos percibir los objetos con sus respectivas formas, conseguimos que el objeto se vea más nítido
- Si los bordes de los objetos se encuentran poco definidos, es más difícil distinguir un objeto de otro



CONVOLUCIÓN

- Es una operación que combina 2 funciones, tal que indica como la forma de una función es modificada por la otra
- Es un filtro de propósito general para las imágenes
- Es una matriz aplicada a una imagen
- Determina el valor de un píxel agregando los valores ponderados de los píxeles en la vecindad





PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

CONVOLUCIÓN 2D

- La matriz representa el filtro a aplicar
- Aplicamos el filtro a cada uno de los píxeles de la imagen
- Se estila tener la imagen en escala de grises

2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

An input image
(no padding)

1	0	1
0	0	0
0	1	0

A filter
(3x3)

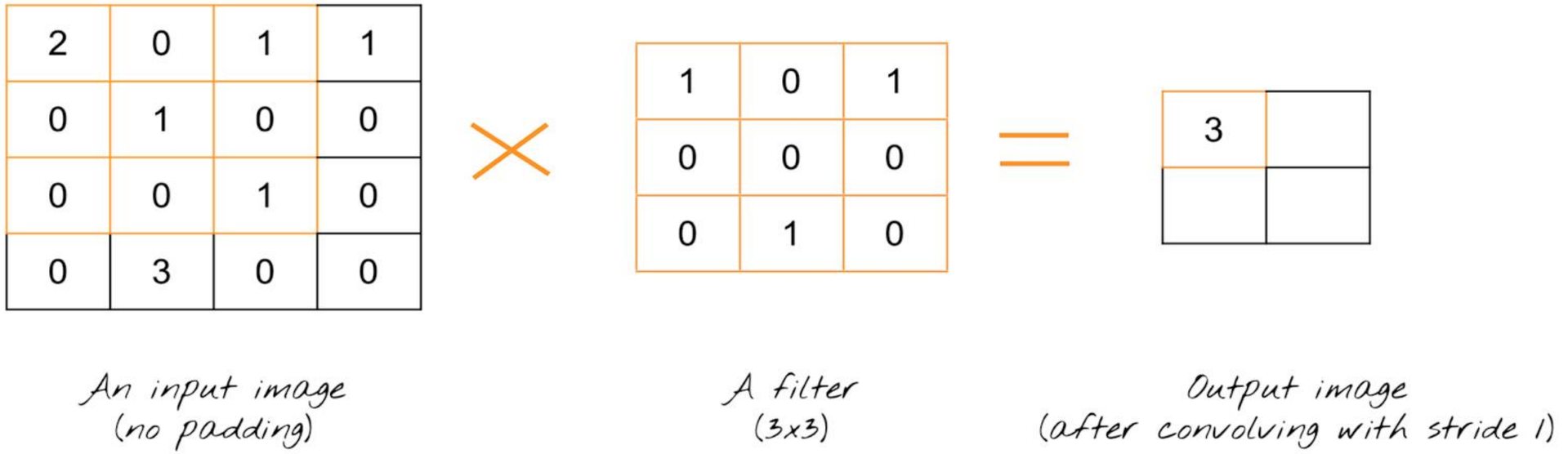
Output image
(after convolving with stride 1)

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

CONVOLUCIÓN 2D

- La matriz representa el filtro a aplicar
- Aplicamos el filtro a cada uno de los píxeles de la imagen
- Se estila tener la imagen en escala de grises

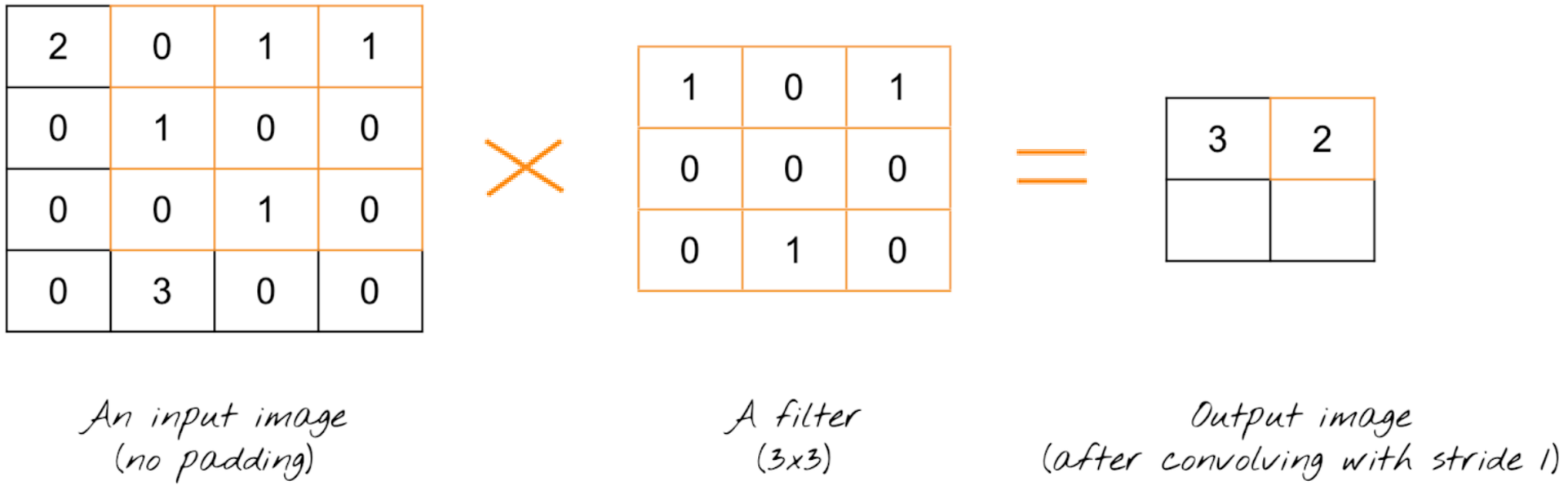


PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

CONVOLUCIÓN 2D

- La matriz representa el filtro a aplicar
- Aplicamos el filtro a cada uno de los píxeles de la imagen
- Se estila tener la imagen en escala de grises

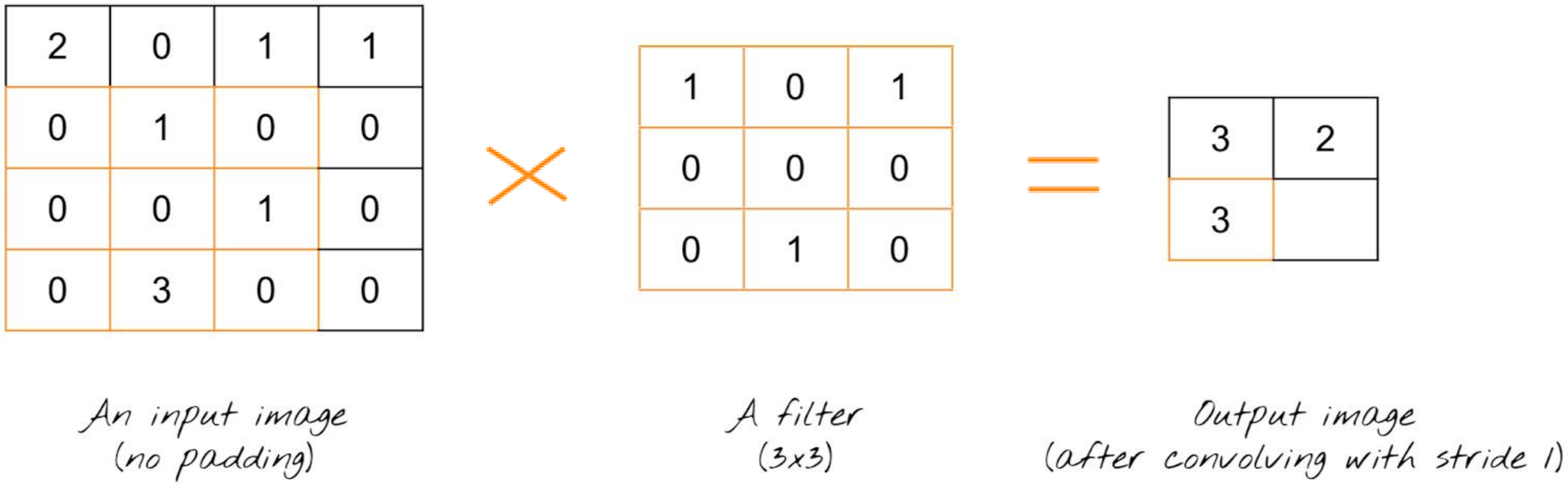


PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

CONVOLUCIÓN 2D

- La matriz representa el filtro a aplicar
- Aplicamos el filtro a cada uno de los píxeles de la imagen
- Se estila tener la imagen en escala de grises

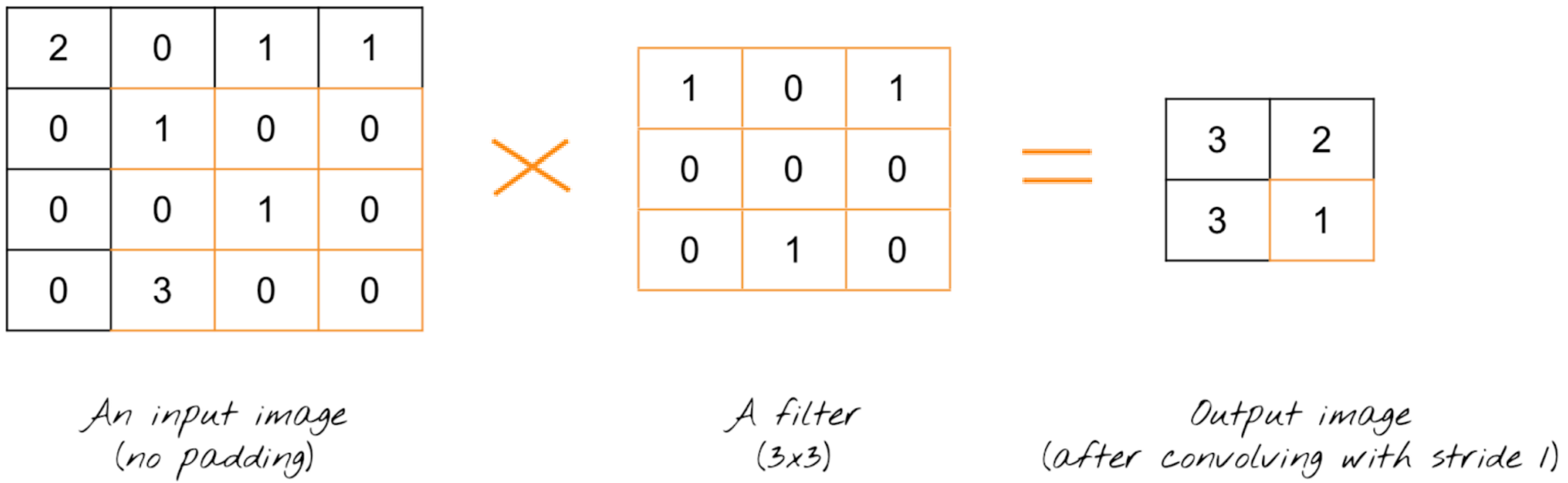


PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

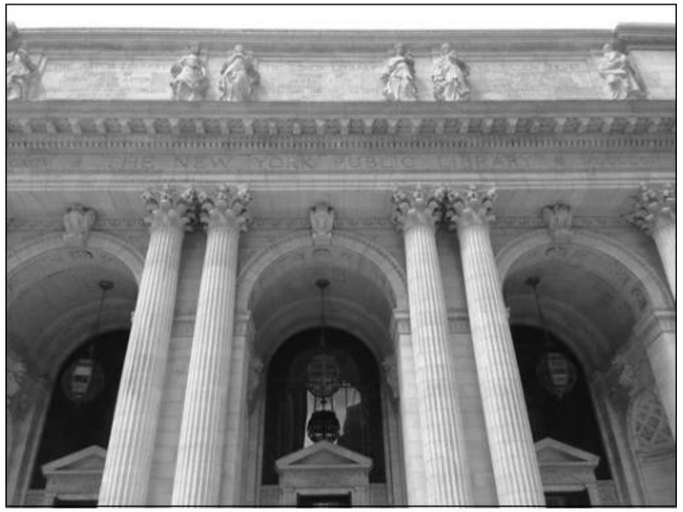
CONVOLUCIÓN 2D

- La matriz representa el filtro a aplicar
- Aplicamos el filtro a cada uno de los píxeles de la imagen
- Se estila tener la imagen en escala de grises



CONVOLUCIÓN 2D

- La matriz representa el filtro a aplicar
- Aplicamos el filtro a cada uno de los píxeles de la imagen
- Se estila tener la imagen en escala de grises



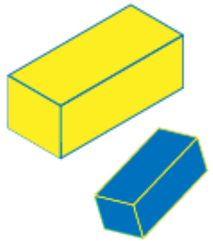
×

-1	-1	-1
-1	8	-1
-1	-1	-1

=

PROCESAMIENTO DIGITAL DE IMÁGENES

Convolucionar una imagen



filter2D : Aplica una convolución a una imagen haciendo uso de un kernel

Sintaxis: cv2.filter2D(imagen, profundidad, kernel)

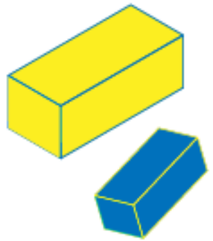
Parametros:
imagen : Imagen a la cual aplicar la convolución
profundidad: Es la profundidad deseada para la imagen resultante
kernel: Filtro a aplicar en la imagen. En el caso que la imagen tenga varios canales, se puede dividir la imagen en sus distintos canales para aplicar diferentes filtros en cada canal. Los valores deben estar como punto flotante

Nota La profundidad es la precisión que tiene cada píxel. Define la cantidad de bits a usar y si se usa «unsigned, int, float o double». Si es igual a -1, entonces la imagen resultante tiene la misma precisión que la imagen original

Ejemplo:

```python
import cv2
import numpy as np

image = cv2.imread("image.jpg")

kernel = np.ones((11, 11), np.float32)
width, height = kernel.shape
kernel /= (width * height)

image_convoluted = cv2.filter2D(image, -1, kernel)

cv2.imshow("Imagen", image)
cv2.imshow("Imagen con la convolución", image_convoluted)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

DIFUMINADO DE IMÁGENES



- El difuminado de una imagen se obtiene convolucionando una imagen
- Se aplica un kernel de filtro pasa bajo para la convolución
- De este modo, se elimina el contenido con alta frecuencia, como es el ruido y los bordes de la imagen
- Con esto, se suele obtener imágenes con los bordes más borrosos
- Sin embargo, existen filtros que eliminan el ruido sin afectando poco a los bordes
- En OpenCV, tenemos 3 tipos: Promedio, mediana y guassiano

DIFUMINADO DE IMÁGENES (PROMEDIO)

- También conocido como filtro pasa bajo
- Un filtro pasa bajo es la base de la mayoría de los métodos de suavizado
- Una imagen se suaviza disminuyendo la disparidad entre los valores de píxeles al promediar los píxeles de la ventana del kernel (vecindad)
- Retenemos la información de baja frecuencia en la imagen
- Reducimos la información de alta frecuencia en la imagen
- Definimos la cantidad de elementos del filtro como «N»

$$N = 9$$

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Convolucionar una imagen

blur : Desenfoca/Difumina una imagen mediante el filtro de cuadro normalizado

Sintaxis: cv2.blur(imagen, tamaño)

Parametros:
imagen : Imagen a la cual aplicar la convolución
tamaño : Tamaño del kernel como una tupla (ancho, alto)

Nota El filtro usado es:

$$\frac{1}{\text{ancho}*\text{alto}} * \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

PROCESAMIENTO DIGITAL DE IMÁGENES

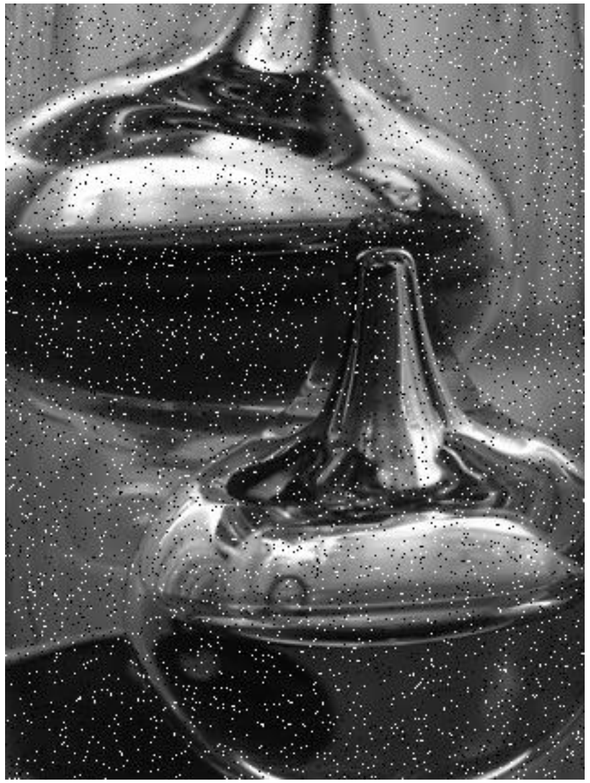
UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Ejemplo:

```
import cv2

image = cv2.imread("image.jpg")

kernel_size = (7, 7)

image_convoluted = cv2.blur(image, kernel_size)

cv2.imshow("Imagen", image)
cv2.imshow("Imagen con la convolución", image_convoluted)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

DIFUMINADO DE IMÁGENES (MEDIANA)

- Calcula la mediana de todos los píxeles en la ventana del kernel (vecindad y el píxel central)
- El píxel central se reemplaza con el valor calculado
- Permite eliminar ruido conocido como sal y pimienta
- El ruido sal y pimienta se caracteriza por cubrir de forma dispersa la imagen con una serie de píxeles blancos y negros



PROCESAMIENTO DIGITAL DE IMÁGENES

Convolucionar una imagen

medianBlur : Desenfoca/Difumina una imagen mediante el filtro mediana

Sintaxis: cv2.medianBlur(imagen, tamaño)

Parametros:
imagen : Imagen a la cual aplicar la convolución
tamaño : Tamaño del kernel

Nota El tamaño del kernel debe ser un entero positivo impar y mayor a 1

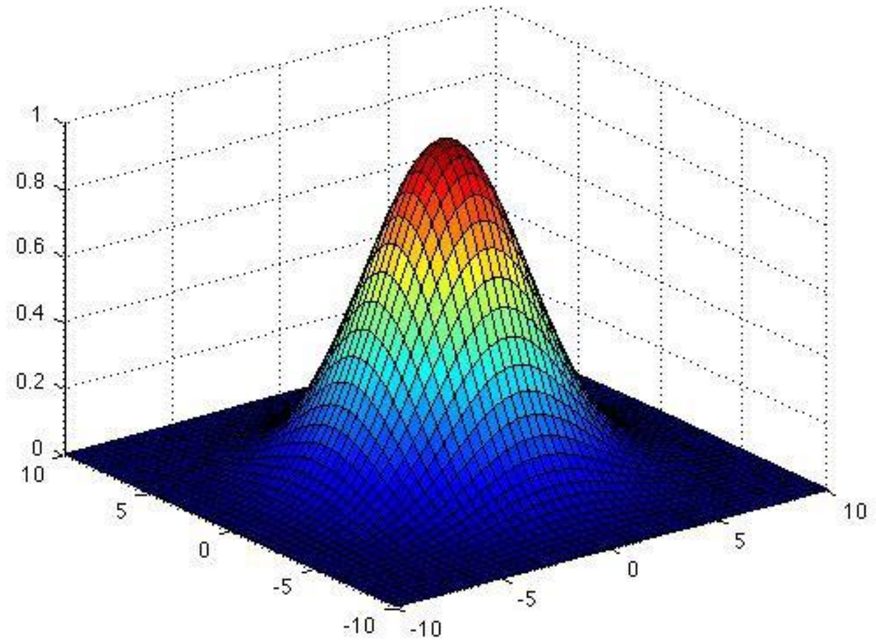
PROCESAMIENTO DIGITAL DE IMÁGENES

Ejemplo:

```
import cv2

image = cv2.imread("image_with_noise_salt_and_pepper.png")

image_convoluted = cv2.medianBlur(image, 9)

cv2.imshow("Imagen", image)
cv2.imshow("Imagen con la convolución", image_convoluted)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

DIFUMINADO DE IMÁGENES (GAUSSIANO)

- No se utiliza un kernel con los coeficientes iguales (como en el caso del Promedio)
- Se utiliza un kernel gaussiano
- Cuanto más cerca se encuentre un píxel del píxel central, más afecta al promedio ponderado usado para establecer el nuevo valor del píxel central
- Se hace la asunción que los píxeles más cercanos al píxel central están más cerca del valor real de dicho píxel, razón por la cual influyen más

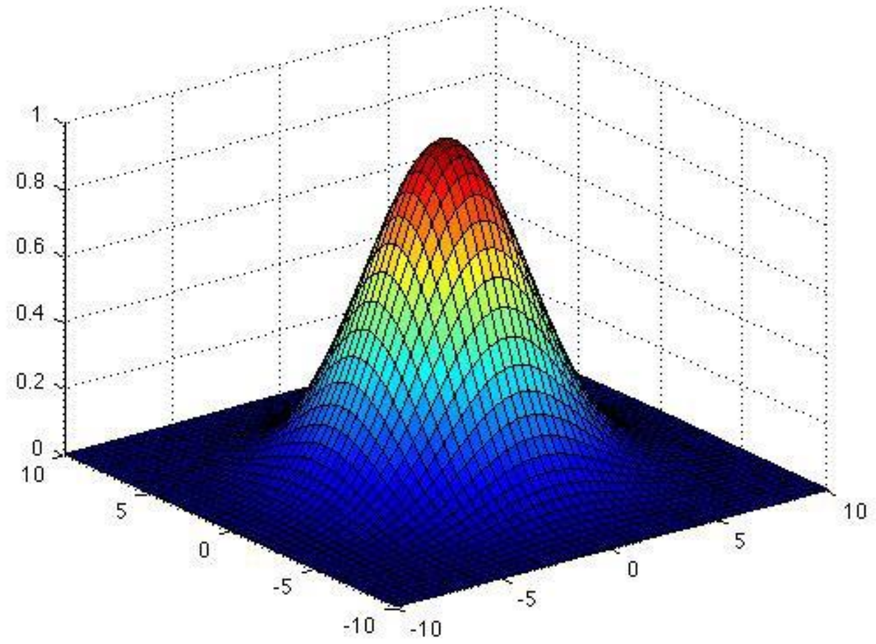


PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

DIFUMINADO DE IMÁGENES (GAUSSIANO)

- Representamos la desviación estándar mediante sigma
- Dado que se trabaja en 3 dimensiones, se tiene un valor sigma para el eje X e Y
- El valor de sigma para el eje X puede ser distinto que el del eje Y
- Sigma indica que tan ancha debe ser la curva dentro del kernel, similar a como sigma define el ancho de una curva de distribución normal



PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Convolucionar una imagen

GaussianBlur : Desenfoca/Difumina una imagen mediante el filtro gaussiano

Sintaxis: cv2.GaussianBlur(imagen, tamaño, sigma_X[, nueva_imagen[, sigma_Y]])

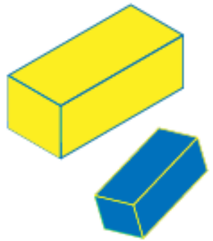
Parametros:
imagen : Imagen a la cual aplicar la convolución
tamaño : Tamaño del kernel como una tupla (ancho, alto)
sigma_X : Desviación estándar en el eje X
nueva_imagen : Imagen resultante, con el mismo tamaño que la imagen recibida
sigma_Y : Desviación estándar en el eje Y

Nota Si no se define sigma_Y, entonces obtendrá el mismo valor que sigma_X. Si ambos sigma son iguales a cero entonces se calcularán automáticamente

Ejemplo:

```python
import cv2

image = cv2.imread("image.jpg")

kernel_size = (7, 7)
sigma_X = 0

image_convoluted = cv2.GaussianBlur(image, kernel_size, sigma_X)

cv2.imshow("Imagen", image)
cv2.imshow("Imagen con la convolución", image_convoluted)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

TRANSFORMACIONES MORFOLÓGICAS



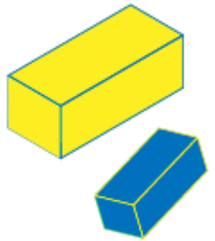
- Son operaciones basadas en la forma de la imagen
- Normalmente estas operaciones se aplican sobre imágenes binarias
- Se requiere de la imagen y del kernel
- El kernel establece el tipo de operación
- Se verá las siguientes transformaciones morfológicas:

Erosión
Dilatación
Apertura
Cierre
Gradiente Morfológico

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

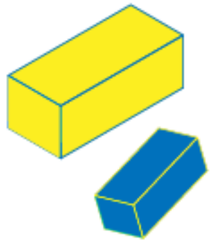
TRANSFORMACIONES MORFOLÓGICAS (EROSIÓN)



- Está basado en la erosión del suelo
- El kernel se mueve a través de la imagen mientras la erosiona
- La imagen está compuesta por píxeles con el valor de 0 o 255
- Un píxel será considerado como 255 cuando todos los píxeles dentro de la ventana del kernel sean 255
- Caso contrario, se erosiona (se convierte el píxel al valor de 0)
- Esto permite que los píxeles cerca de los bordes de los objetos sean descartados, lo cual depende del tamaño del kernel
- Sirve para eliminar pequeños ruidos de color blanco o separar 2 objetos conectados

PROCESAMIENTO DIGITAL DE IMÁGENES

Erocionar una imagen



erode : Erosiona una imagen

Sintaxis: cv2.erode(imagen, kernel)

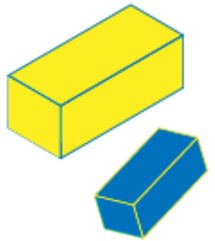
Parametros:
imagen : Imagen a la cual aplicar la erosión
kernel: Operación a aplicar en la imagen

Nota En el caso de tener una imagen con varios canales, la erosión se procesa de manera independiente en cada uno de los canales. La eroción se puede aplicar varias veces, para ello, se usa el parámetro «iterations» dentro del método «erode», con el fin de indicar la cantidad de iteraciones. Por ejemplo, cv2.erode(imagen, kernel, iterations=3)

Ejemplo:

```python
import cv2
import numpy as np

image_as_BGR = cv2.imread("image_on_black_and_white.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

kernel = np.ones((5, 5), np.float32)
width, height = kernel.shape
kernel /= (width * height)

image_eroded = cv2.erode(image_as_GRAY, kernel)

cv2.imshow("Imagen", image_as_GRAY)
cv2.imshow("Imagen erosionada", image_eroded)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

TRANSFORMACIONES MORFOLÓGICAS (DILATACIÓN)



- Es el opuesto de la erosión
- El kernel se mueve a través de la imagen mientras la dilata
- La imagen está compuesta por píxeles con el valor de 0 o 255
- Un píxel será considerado como 255 si al menos un píxel dentro de la ventana del kernel es 255
- De este modo, se termina aumentando el tamaño de los objetos (con color blanco) en la imagen
- Para eliminar ruido, normalmente luego de la erosión se usa la dilatación
- Esto es debido que la erosión al eliminar los ruidos de color blanco también encoge los objetos. Por ello, la dilatación permite recuperar el tamaño inicial
- También es útil para unir objetos separados o partes rotas de un objeto

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Dilatar una imagen

dilate : Dilata una imagen

Sintaxis: cv2.dilate(imagen, kernel)

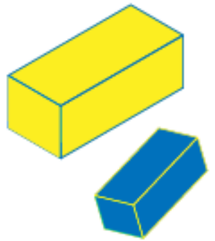
Parametros:
imagen : Imagen a la cual aplicar la dilatación
kernel: Operación a aplicar en la imagen

Nota En el caso de tener una imagen con varios canales, la erosión se procesa de manera independiente en cada uno de los canales. La dilatación se puede aplicar varias veces, para ello, se usa el parámetro «iterations» dentro del método «dilate», con el fin de indicar la cantidad de iteraciones. Por ejemplo, cv2.dilate(imagen, kernel, iterations=3)

Ejemplo:

```
import cv2
import numpy as np

image_as_BGR = cv2.imread("image_on_black_and_white.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

kernel = np.ones((5, 5), np.float32)
width, height = kernel.shape
kernel /= (width * height)

image_dilated = cv2.dilate(image_as_GRAY, kernel)

cv2.imshow("Imagen", image_as_GRAY)
cv2.imshow("Imagen dilatada", image_dilated)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

TRANSFORMACIONES MORFOLÓGICAS (APERTURA)



- Consiste en aplicar una erosión seguida de una dilatación
- El kernel se mueve a través de la imagen mientras la erosiona para luego dilatarla
- La imagen está compuesta por píxeles con el valor de 0 o 255
- Es útil para eliminar ruido
- En OpenCV viene definido por el flag cv2.MORPH_OPEN

PROCESAMIENTO DIGITAL DE IMÁGENES

Aplicar una trasformación morfológica avanzada a una imagen

morphologyEx : Permite ejecutar una transformación morfológica avanzada

Sintaxis: cv2.morphologyEx(imagen, tipo, kernel)

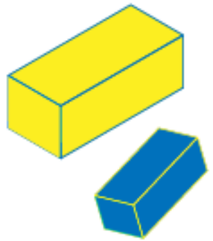
Parametros:
imagen : Imagen a la cual aplicar la transformación morfológica
tipo : Tipo de transformación morfológica a realizar
kernel: Operación a aplicar en la imagen

Nota Mediante el parámetro «iterations» podemos indicar cuantas veces se debe elaborar la erosión seguida de la dilatación (para el flag cv2.MORPH_OPEN). Por ejemplo, cv2.morphologyEx(image, tipo, kernel, iterations=3)

Ejemplo:

```
import cv2
import numpy as np

image_as_BGR = cv2.imread("image_on_black_and_white_for_opening.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

kernel = np.ones((5, 5), np.float32)
width, height = kernel.shape
kernel /= (width * height)

image_with_aperture = cv2.morphologyEx(image_as_GRAY, cv2.MORPH_OPEN, kernel)

cv2.imshow("Imagen", image_as_GRAY)
cv2.imshow("Imagen con apertura", image_with_aperture)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

TRANSFORMACIONES MORFOLÓGICAS (CIERRE)



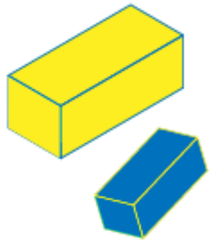
- Es el opuesto de la Apertura
- Consiste en aplicar una dilatación seguida de una erosión
- El kernel se mueve a través de la imagen mientras la erosiona para luego dilatarla
- La imagen está compuesta por píxeles con el valor de 0 o 255
- Es útil para cerrar pequeños agujeros o pequeños puntos negros dentro de los objetos
- En OpenCV viene definido por el flag cv2.MORPH_CLOSE

PROCESAMIENTO DIGITAL DE IMÁGENES

Ejemplo:

```python
import cv2
import numpy as np

image_as_BGR = cv2.imread("image_on_black_and_white_for_closing.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

kernel = np.ones((5, 5), np.float32)
width, height = kernel.shape
kernel /= (width * height)

image_with_aperture = cv2.morphologyEx(image_as_GRAY, cv2.MORPH_CLOSE, kernel)

cv2.imshow("Imagen", image_as_GRAY)
cv2.imshow("Imagen con cierre", image_with_aperture)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

TRANSFORMACIONES MORFOLÓGICAS (GRADIENTE MORFOLÓGICO)



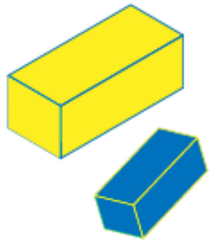
- Consiste en la diferencia entre la dilatación y la erosión en una imagen
- El kernel se mueve a través de la imagen mientras la erosiona para luego dilatarla
- La imagen está compuesta por píxeles con el valor de 0 o 255
- El resultado se verá como si se tuviera el contorno del objeto
- En OpenCV viene definido por el flag cv2.MORPH_GRADIENT

PROCESAMIENTO DIGITAL DE IMÁGENES

Ejemplo:

```python
import cv2
import numpy as np

image_as_BGR = cv2.imread("image_on_black_and_white.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

kernel = np.ones((5, 5), np.float32)
width, height = kernel.shape
kernel /= (width * height)

image_with_aperture = cv2.morphologyEx(image_as_GRAY, cv2.MORPH_GRADIENT, kernel)

cv2.imshow("Imagen", image_as_GRAY)
cv2.imshow("Imagen con cierre", image_with_aperture)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

TRANSFORMACIONES MORFOLÓGICAS (SOMBRERO DE COPA)



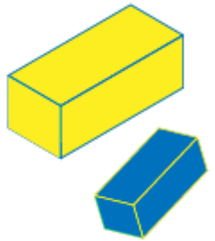
- Consiste en la diferencia entre la imagen original y la apertura de la imagen
- El kernel se mueve a través de la imagen mientras la erosiona para luego dilatarla
- La imagen está compuesta por píxeles con el valor de 0 o 255
- En OpenCV viene definido por el flag cv2.MORPH_TOPHAT

PROCESAMIENTO DIGITAL DE IMÁGENES

Ejemplo:

```
import cv2
import numpy as np

image_as_BGR = cv2.imread("image_on_black_and_white.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

kernel = np.ones((9, 9), np.float32)
width, height = kernel.shape
kernel /= (width * height)

image_with_aperture = cv2.morphologyEx(image_as_GRAY, cv2.MORPH_TOPHAT, kernel)

cv2.imshow("Imagen", image_as_GRAY)
cv2.imshow("Imagen con cierre", image_with_aperture)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

TRANSFORMACIONES MORFOLÓGICAS (SOMBRERO NEGRO)



- Consiste en la diferencia entre el cierre (de la imagen original) y la imagen original
- El kernel se mueve a través de la imagen mientras la erosiona para luego dilatarla
- La imagen está compuesta por píxeles con el valor de 0 o 255
- En OpenCV viene definido por el flag cv2.MORPH_BLACKHAT

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Ejemplo:

```python
import cv2
import numpy as np

image_as_BGR = cv2.imread("image_on_black_and_white.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

kernel = np.ones((9, 9), np.float32)
width, height = kernel.shape
kernel /= (width * height)

image_with_aperture = cv2.morphologyEx(image_as_GRAY, cv2.MORPH_BLACKHAT, kernel)

cv2.imshow("Imagen", image_as_GRAY)
cv2.imshow("Imagen con cierre", image_with_aperture)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO