L’uso della convoluzione per l’*image filtering*

Matteo Rota
Principi e Modelli della Percezione
Università degli Studi di Milano
A.A. 2023-2024

Indice

- La convoluzione
- Kernel di convoluzione nell'*image processing*
- Applicazione dell'*identity kernel*
- Sfocatura di un'immagine utilizzando un kernel di convoluzione 2D personalizzato
- *Gaussian blurring*
- *Median blurring*
- Nitidezza di un'immagine usando kernel di convoluzione 2D personalizzati
- Filtraggio bilaterale

La convoluzione

- Che cos’è?
- La convoluzione nell’elaborazione delle immagini

Che cos’è?

La **convoluzione** è un’operazione matematica che combina due funzioni per descrivere la loro sovrapposizione: vengono fatte «scorrere» l’una sull’altra, moltiplicando i valori della funzione in ogni punto di sovrapposizione e aggiungendo i prodotti con l’obiettivo di creare una nuova funzione. Quest’ultima rappresenta il modo in cui le due funzioni originali interagiscono tra di loro.

Formalmente…

La convoluzione è un integrale che esprime la quantità di sovrapposizione di una funzione $f(t)$ mentre viene spostata sulla funzione $g(t)$.

$$(f * g)(t) \approx^{def} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)dr$$

La convoluzione nell’elaborazione delle immagini

È possibile utilizzare il filtraggio convoluzionale nell’elaborazione delle immagini, implementando algoritmi come il rilevamento dei contorni, l’ottimizzazione delle immagini e la sfocatura delle immagini.

Tutto questo viene eseguito applicando il corretto kernel (o matrice) di convoluzione.

Sfocatura dell’immagine usando la convoluzione con un filtro di media.

Kernel di convoluzione nell'*image processing*

7

- Introduzione
- Metodo di utilizzo
- Risultato finale

Introduzione

Nell'ambito *dell'image processing* (elaborazione delle immagini), un **kernel di convoluzione** è una matrice 2D utilizzata per filtrare le immagini.

Generalmente, si tratta di una matrice quadrata *MxN*, nella quale *M* e *N* sono numeri interi dispari.

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

Kernel di convoluzione 2D 3x3.

Metodo di utilizzo

Il filtraggio di un'immagine si ottiene seguendo questi passaggi:

1. Supporre che il centro del kernel sia posizionato su uno specifico pixel *p* in un'immagine.
2. Moltiplicare il valore di ciascun elemento nel kernel (facendo riferimento alla matrice dell'immagine precedente) per l'elemento pixel corrispondente (ovvero l'intensità del pixel) nell'immagine sorgente.
3. Sommare il risultato delle moltiplicazioni e calcolare la media.
4. Sostituire il valore del pixel *p* con il valore medio appena calcolato.

Risultato finale

Seguendo i passaggi precedenti per ogni pixel dell'immagine sorgente e utilizzando il kernel di convoluzione, l'immagine filtrata risulterà sfocata.

Motivazione: l'operazione di convoluzione con quel kernel ha un effetto di media, che tende a sfocare o levigare l'immagine.

Modificando i valori del kernel, si può anche ottenere un effetto di nitidezza.

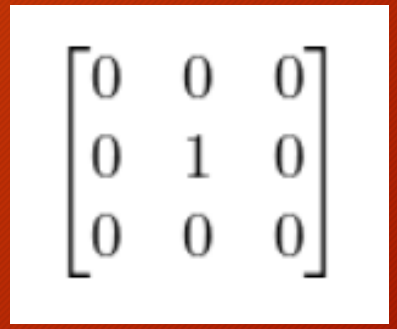
Applicazione dell'*identity kernel*

- Nozione teorica
- Esempio pratico

Nozione teorica

L'***identity kernel*** (kernel di identità) è una matrice quadrata, nella quale l'elemento centrale è 1 e tutti gli altri elementi sono 0.

L'aspetto principale di questo tipo di matrice è che moltiplicandola con una qualsiasi altra matrice si ottiene la matrice originale.



Identity kernel 3x3.

Esempio pratico

Codice Python

```python
#importo le librerie OpenCV e Numpy
import cv2
import numpy as np

#leggo l'immagine sorgente
img = cv2.imread('Gioconda.jpg')

#creo l'identity kernel
k1 = np.array([[0, 0, 0],
               [0, 1, 0],
               [0, 0, 0]])

#applico la funzione filter2D()
identity = cv2.filter2D(src=img, ddepth=-1, kernel=k1)

#visualizzo l'immagine originale e quella filtrata
cv2.imshow('Originale', img)
cv2.imshow('Identity', identity)

#salvo l'immagine filtrata su disco
cv2.waitKey()
cv2.imwrite('identity.jpg', identity)
cv2.destroyAllWindows()
```

Risultato finale

Immagine originale

Immagine filtrata con l'identity kernel

Sfocatura di un’immagine utilizzando un kernel di convoluzione 2D personalizzato

- Nozione teorica e codice
  - *filter2D()* di OpenCV
    - Risultato finale
- Alternativa: funzione integrata *blur()*

Nozione teorica e codice

Per sfocare un'immagine utilizzando un **kernel di convoluzione 2D personalizzato**, è necessario usare una funzione che consenta tutto questo, ovvero *filter2D()* di OpenCV.

Viene usata una matrice 5x5, composta solo da 1. Tuttavia, prima di effettuare l'operazione di sfocatura, è necessario avere tutti i valori normalizzati: ogni elemento del kernel viene diviso per il numero di elementi del kernel, ovvero 25, assicurando così di avere tutti valori compresi nell'intervallo [0,1].

Successivamente è possibile applicare la funzione *filter2D()* per sfocare l'immagine.

```
#importo le librerie OpenCV e Numpy
import cv2
import numpy as np

#leggo l'immagine sorgente
img = cv2.imread('Gioconda.jpg')

#creo il kernel per la sfocatura
k2 = np.ones((5, 5), np.float32) / 25

#applico la funzione filter2D()
img2 = cv2.filter2D(src=img, ddepth=-1, kernel=k2)

#visualizzo l'immagine originale e quella filtrata
cv2.imshow('Originale', img)
cv2.imshow('Kernel Blur', img2)

#salvo l'immagine filtrata su disco
cv2.waitKey()
cv2.imwrite('blur_kernel.jpg', img2)
cv2.destroyAllWindows()
```

filter2D() di OpenCV

La funzione ***filter2D()*** richiede 3 argomenti di input:

- Primo argomento: immagine sorgente.
- Secondo argomento: *ddepth*, ovvero la profondità dell'immagine risultante.
- Terzo argomento: *kernel*, da applicare all'immagine sorgente.

Risultato finale

Immagine originale

Immagine filtrata

Alternativa: funzione integrata *blur()*

È una funzione fornita da OpenCV che consente sempre di sfocare un'immagine, senza necessità però di definire un kernel. Viene specificata solo la dimensione del kernel usando *ksize* come argomento di input.

La funzione di sfocatura *blur()* creerà quindi internamente un kernel di sfocatura e lo applicherà all'immagine sorgente.

```
#importo le librerie OpenCV e Numpy
import cv2
import numpy as np

#leggo l'immagine sorgente
img = cv2.imread('Gioconda.jpg')

#applico la funzione blur()
img2 = cv2.blur(src=img, ksize=(5,5))

#visualizzo l'immagine originale e quella filtrata
cv2.imshow('Originale', img)
cv2.imshow('Blurred', img2)

#salvo l'immagine filtrata su disco
cv2.waitKey()
cv2.imwrite('blur.jpg', img2)
cv2.destroyAllWindows()
```

Gaussian blurring

- Nozione teorica
- *GaussianBlur()* di OpenCV
- Codice e risultato finale

Nozione teorica

La sfocatura gaussiana (***gaussian blurring***) utilizza un filtro gaussiano che esegue una media ponderata: vengono ponderati i valori dei pixel, in base alla loro distanza dal centro del kernel (i pixel più lontani dal centro hanno meno influenza sulla media ponderata).

GaussianBlur() di OpenCV

A tale scopo è possibile usare la funzione ***GaussianBlur()*** di OpenCV.

La funzione richiede quattro argomenti di input:

- Primo argomento: *src*, ovvero l'immagine sorgente da filtrare.
- Secondo argomento: *ksize*, ovvero la dimensione del kernel gaussiano.
- Terzo argomento: *sigmaX*, ovvero la deviazione standard del kernel gaussiano nella direzione X (orizzontale).
- Quarto argomento: *sigmaY*, ovvero la deviazione standard del kernel gaussiano nella direzione Y (verticale).

Codice e risultato finale

Codice Python

```python
#importo le librerie OpenCV e Numpy
import cv2
import numpy as np

#leggo l'immagine sorgente
img = cv2.imread('Gioconda.jpg')

#applico la funzione GaussianBlur()
img2 = cv2.GaussianBlur(src=img, ksize=(5,5), sigmaX=0, sigmaY=0)

#visualizzo l'immagine originale e quella filtrata
cv2.imshow('Originale', img)
cv2.imshow('Gaussian Blurred', img2)

#salvo l'immagine filtrata su disco
cv2.waitKey()
cv2.imwrite('gaussian_blur.jpg', img2)
cv2.destroyAllWindows()
```

Risultato finale

Immagine originale

Immagine filtrata

Median blurring

- *medianBlur()* di OpenCV
- Codice e risultato finale

medianBlur() di OpenCV

La sfocatura mediana (***median blurring***) consente di andare a sfocare un'immagine sostituendo ogni pixel nell'immagine sorgente con il valore mediano dei pixel dell'immagine kernel.

Può essere utilizzata la funzione ***medianBlur()*** di OpenCV, la quale richiede 2 argomenti di input:

- Primo argomento: immagine sorgente.
- Secondo argomento: dimensione del kernel (numero intero positivo dispari).

Codice e risultato finale

Codice Python

```
#importo le librerie OpenCV e Numpy
import cv2
import numpy as np

#leggo l'immagine sorgente
img = cv2.imread('Gioconda.jpg')

#applico la funzione medianBlur()
img2 = cv2.medianBlur(src=img, ksize=5)

#visualizzo l'immagine originale e quella filtrata
cv2.imshow('Originale', img)
cv2.imshow('Median Blurred', img2)

#salvo l'immagine filtrata su disco
cv2.waitKey()
cv2.imwrite('median_blur.jpg', img2)
cv2.destroyAllWindows()
```

Risultato finale

Immagine originale

Immagine filtrata

Nitidezza di un’immagine usando kernel di convoluzione 2D personalizzati

- Codice e risultato finale

Codice e risultato finale

Codice Python

```
#importo le librerie OpenCV e Numpy
import cv2
import numpy as np

#leggo l'immagine sorgente
img = cv2.imread('Gioconda.jpg')

#creo il kernel
k1 = np.array([[0, -1,  0],
               [-1,  5, -1],
               [0, -1,  0]])

#applico la funzione filter2D()
img2 = cv2.filter2D(src=img, ddepth=-1, kernel=k1)

#visualizzo l'immagine originale e quella filtrata
cv2.imshow('Originale', img)
cv2.imshow('Sharpened', img2)

#salvo l'immagine filtrata su disco
cv2.waitKey()
cv2.imwrite('sharp_image.jpg', img2)
cv2.destroyAllWindows()
```

Risultato finale

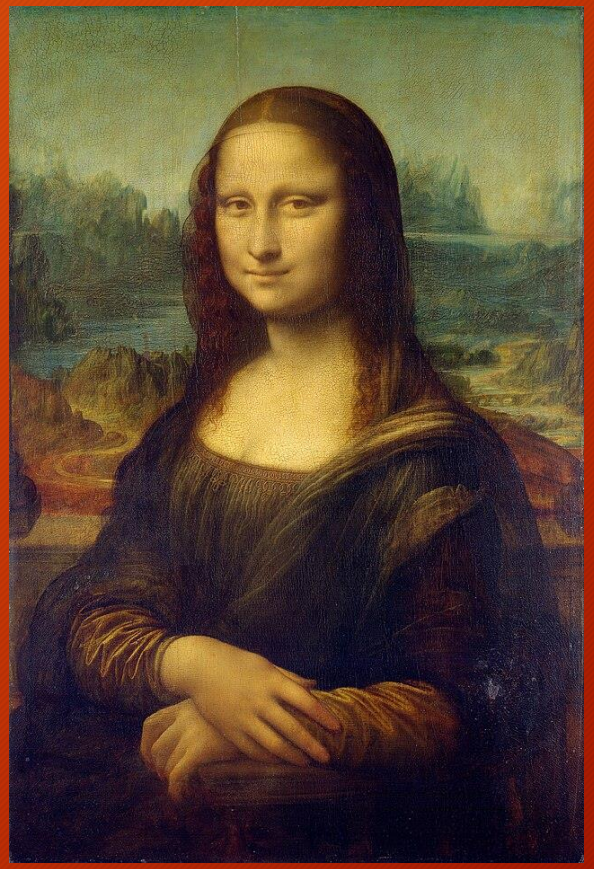


Immagine originale

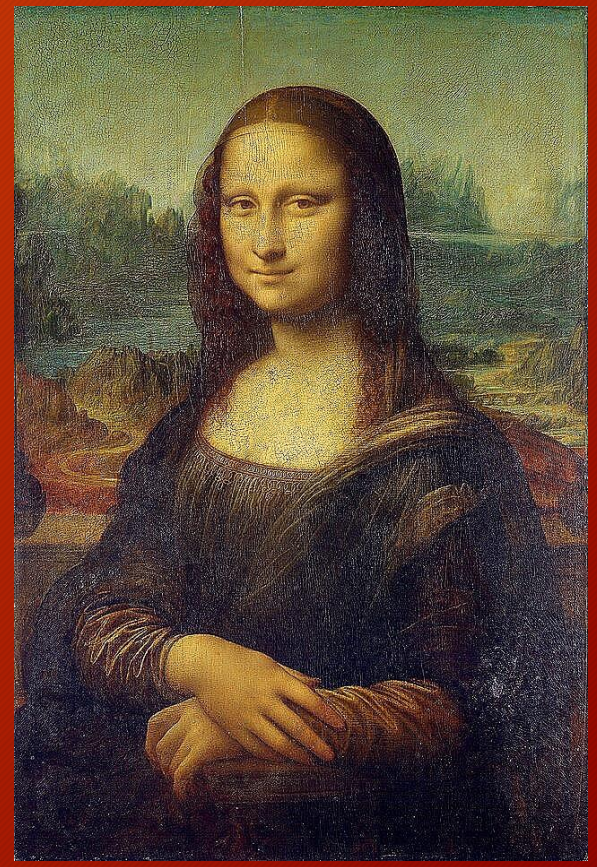


Immagine filtrata

Filtraggio bilaterale

- A cosa serve?
- *bilateralFilter()* in OpenCV
- Codice e risultato finale

A cosa serve?

Il **filtraggio bilaterale** è una tecnica di filtraggio che consente di sfocare un'immagine preservando però alcuni dettagli dell'immagine originale.

Questa tecnica applica il filtro selettivamente per sfocare pixel di intensità simile in un vicinato, tenendo intatti i bordi nitidi, ove possibile. In particolare, viene applicato un filtro gaussiano all'immagine, considerando anche la variazione di intensità dei pixel adiacenti per ridurre al minimo la sfocatura vicino ai bordi.

bilateralFilter() in OpenCV

La funzione ***bilateralFilter()*** di OpenCV consente di effettuare il filtraggio bilaterale descritto precedentemente.

Richiede 4 argomenti di input:

1. Primo argomento: immagine sorgente.
2. Secondo argomento: *d*, ovvero il diametro del vicinato dei pixel utilizzato per il filtraggio.
3. Terzo argomento: *sigmaColor*, definisce la deviazione standard della distribuzione dell'intensità del colore.
4. Quarto argomento: *sigmaSpace*, definisce la deviazione standard della distribuzione spaziale.

Il valore finale, ovvero ponderato, per un pixel nell'immagine filtrata è un prodotto del suo peso spaziale e di intensità.

Codice e risultato finale

Codice Python

```python
#importo le librerie OpenCV e Numpy
import cv2
import numpy as np

#leggo l'immagine sorgente
img = cv2.imread('Gioconda.jpg')

#applico la funzione bilateralFilter()
img2 = cv2.bilateralFilter(src=img, d=9, sigmaColor=75, sigmaSpace=75)

#visualizzo l'immagine originale e quella filtrata
cv2.imshow('Originale', img)
cv2.imshow('Bilateral Filtering', img2)

#salvo l'immagine filtrata su disco
cv2.waitKey(0)
cv2.imwrite('bilateral_filtering.jpg', img2)
cv2.destroyAllWindows()
```

Risultato finale

Immagine originale

Immagine filtrata

Sitografia

- https://it.mathworks.com/discovery/convolution.html
- https://learnopencv.com/image-filtering-using-convolution-in-opencv/