Filtraggio lineare e nonlineare di immagini in Python 3

Lorenzo Gasparini

Febbraio 2015

1 Introduzione

La consegna di questa tesina è la seguente:

> "Scrivere un programma, in un linguaggio a scelta, per realizzare il filtraggio lineare e non lineare di immagini. Gli ingressi al programma sono una immagine originale, l'ordine del filtro ed i coefficienti del filtro. L'uscita è l'immagine filtrata."

Per realizzare il programma, è stato scelto il linguaggio Python nella sua versione 3.4. Questo perchè Python dispone di buone librerie per la lettura/scrittura di immagini (*Python Imaging Library*), e per la manipolazione di matrici in modo efficiente (*NumPy*).

Il risultato finale è un package contenente i metodi e le classi *pyimagefilter*, affiancato da un modulo *cli.py* che permette di utilizzare le librerie da linea di comando, per applicare i filtri.

2 Installazione

Innanzitutto bisogna ottenere la cartella del progetto. È possibile ad esempio clonare in locale il repository Github con il comando

```
$ git clone https://github.com/joined/pyimagefilter
$ cd pyimagefilter
```

oppure scaricare l'archivio da Github ed estrarlo.

Una volta ottenuto il progetto, la strada migliore per installarlo senza dover installare package a livello di sistema, è creare un *virtual enviroment* con l'interprete di Python 3.4 con il comando

```
$ virtualenv .env -p $(which python3)
```

dunque è necessario attivare l'ambiente con

```
$ source .env/bin/activate
```

A questo punto dobbiamo installare i package necessari al funzionamento del toolkit

```
$ pip install -r requirements.txt
```

ed una volta terminata l'installazione, il tool è pronto per funzionare.

3 Utilizzo

Essendo, come già detto, questo programma disposto di modulo utilizzabile da linea di comando, possiamo invocare l'help con il comando

```
$ ./cli.py -h
usage: cli.py [-h] [--average RANK] [--gauss STDEV,RANK] [--sharpen TYPE]
              [--prewitt TYPE] [--sobel TYPE] [--volterra COEFFICIENT_FILE]
              [--custom MASK] [--no-parallel] [--output OUTPUT_IMAGE]
              input_image

Toolkit for linear and nonlinear image filtering

positional arguments:
  input_image           Input image file name.

optional arguments:
  -h, --help            show this help message and exit
  --average RANK        average mask transform. rank must be unven
  --gauss STDEV,RANK    gauss average transform. rank must be uneven
  --sharpen TYPE        sharpen mask transform. available types 1,2,3
  --prewitt TYPE        prewitt mask transform. available types 1,2
  --sobel TYPE          sobel mask transform. available types 1,2
  --volterra COEFFICIENT_FILE
                        quadratic volterra filtering. file must be json
  --custom MASK         custom mask linear filter, json-style format
  --no-parallel         disable parallel execution
  --output OUTPUT_IMAGE
                        output image filename
```

Come vediamo c'è un solo argomento obbligatorio, e diversi altri opzionali.

L'argomento obbligatorio è l'immagine di input, che deve esser in uno dei formati supportati da *Python Imaging Library*, che comprendono JPEG, GIF, PNG, e altri.

Gli argomenti che non sono relativi al tipo di filtraggio sono *no-parallel* e *output*:

- *no-parallel* serve a disabilitare l'esecuzione parallela dell'operazione di filtraggio
- *output* serve per scegliere il nome del file di output.

Se il nome del file di output non è fornito, verrà scelto un nome del tipo *output_XX.jpg* dove XX è un numero di due cifre generato casualmente. Dunque, il file verrà salvato il formato JPEG.

3.1 Filtraggio lineare

Per quanto riguarda il filtraggio lineare, ci sono due opzioni:

- utilizzare una delle maschere predefinite, specificandone i parametri
- utilizzare una maschera personalizzata.

I tipi di maschera predefiniti sono 5, e accettano diversi parametri:

- *average* è la maschera che ad ogni pixel sostituisce la media aritmetica di quelli intorno a lui. Accetta come parametro la dimensione della maschera, che deve essere dispari.
- *gauss* è la maschera che ad ogni pixel sostituisce la media gaussiana di quelli intorno a lui. Come parametri accetta la deviazione standard da utilizzare nella funzione gaussiana, e la dimensione della maschera (sempre dispari).
- *sharpen* è la maschera di nitidezza che evidenzia i dettagli, di cui si possono scegliere 3 varianti.
- *prewitt* e *sobel* sono maschere che evidenziano i contorni, e hanno entrambe 2 varianti.

Per quanto riguarda la maschera personalizzata, è possibile usare l'argomento *custom* per inserirla. La maschera dev'essere in formato JSON per essere letta dal programma. Un esempio di filtraggio personalizzato con maschera 3x3 è il seguente:

```
$ ./cli.py --custom [[1,2,3],[4,5,6],[7,8,9]] image.jpg
```

3.2 Filtraggio non lineare

Il programma prevede anche la possibilità di eseguire un filtraggio non lineare con un filtro quadratico di Volterra. Per eseguire questo filtraggio sono necessarie due matrici di coefficienti, una di dimensione $N \times N$ e l'altra di dimensione $N \times N \times N \times N$, dove N è dispari.

Queste matrici di coefficienti devono essere strutturate in un file JSON, dove l'elemento A del file contiene la prima matrice, e l'elemento B contiene la seconda. Il comando per eseguire un filtraggio con kernel di Volterra è il seguente:

```
$ ./cli.py --volterra file_coefficienti.json image.jpg
```

Per un esempio della struttura del file, si veda il file *test_volterra.json*.

3.3 Concatenazione di filtri

È possibile, per come è strutturato il programma, concatenare più filtri di diverso tipo. L'ordine in cui vengono specificati i filtri viene rispettato.

Un esempio di concatenazione di filtri è il seguente

```
$ ./cli.py --average 3 --custom [[0.5]] image.jpg
```

che prima applica il filtro della media e poi applica una maschera che ad ogni pixel dimezza i valori delle componenti RGB.