PROGETTO DI SISTEMI MULTIMEDIALI

ELABORAZIONE DI IMMAGINI

Barco Simone Matricola: 1052302
Graziano Grespan Matricola: 1003760
Santimaria Davide Matricola: 1097392

15 luglio 2016

Tipologia Progetto: `Elaborazione di Immagine`
Email referente: `grazianogrespan@gmail.com`

Indice

1 Guida all'uso del programma

Il programma dato è scritto in codice C++ ed è stato testato con i compilatori g++ e clang producendo correttamente i programmi oggetto ed il relativo esegubile.
E' possibile compilare il programma in un semplice passo grazie al MakeFile incluso nel file .zip.

Per ottenere una compilazione ottimale si consiglia di eseguire in ordine i seguenti comandi:

1. Spostarsi con il terminale nella cartella *sorgenti*
2. Eseguire il comando:

   ```
   make && make clean
   ```

In questo modo nella cartella corrente verrà creato il file eseguibile nominato *elaboraimmagine* nell'estensione *.out* se ci si trova in ambiente Linux oppure *.exe* se Windows .

il comando *make clean* rimuove dalle cartelle dei file *.cpp* i relativi files oggetto.

Per eseguire il programma da terminale digitare :

./elaboraimmagine[.out/.exe] <fileinp ><filtro ><dimX ><dimY ><fileout>

La stringa in input associata al filtro deve avere il seguente formato:

- *blur* : corrisponde al filtro di Blur
- *bn* : corrisponde al filtro di Bianco/Nero
- *sharpen* : corrisponde al filtro di Sharpen
- *seppia* : corrisponde al filtro Seppia
- *contrasto* : corrisponde al filtro Contrasto
- *lumin* : corrisponde al filtro Luminosità
- *negativo* : corrisponde al filtro Negativo
- *trama* : corrisponde al filtro Trama
- *zoom* : corrisponde al filtro di Zoom

Un esempio di corretta esecuzione del programma:
./elaboraimmagine /img/input.bmp bn 400 300 /img/output.bmp

2 I Filtri

Per elaborare l'immagine desiderata l'implementazione del nostro codice si occupa in primis di applicare i filtri, brevemente elencati qui sotto, e successivamente di scalare l'immagine alla dimensione voluta.

Per effettuare il resize dell'immagine è stata sviluppata la funzione Scala alla quale vengono passate le dimensioni dell'immagine di input e quelle che si vogliono dare alla destinazione in output.

Per quanto riguarda le dimensioni dell'immagine di input, esse vengono estrapolate attraverso una funzione chiamata OttieniHAL (HAL = Header - Altezza - Larghezza), che scorrendo l'header dell'immagine salva in una struttura: dimensione dell'header, larghezza immagine sorgente e altezza immagine sorgente.

In base a questi valori viene calcolato un fattore sia per larghezza che per altezza che verrà poi usato per ingrandirla o rimpicciolirla.
Il ridimensionamento è ottenuto passando tutti i pixel di ogni canale alla funzione LeggiInterpolato, la quale calcola il valore da assegnare al pixel di destinazione, valutandolo nella sorgente sui quattro pixel che si trovano superiormente, inferiormente, a sinistra e a destra rispetto a quello che si sta considerando.

Ciò serve a calcolare la nuova matrice che conterrà l'immagine ridimensionata.
Toccherà poi alla funzione SalvaBmp24 modificare l'header per inserire la nuova larghezza e la nuova altezza desiderata, aggiungendo successivamente la matrice appena trovata.

Poiché in una bitmap è necessario che il numero di byte per riga di pixels sia un multiplo di 4 byte, si è deciso di scalare inferiormente l'immagine ad una dimensione dell'asse x pari al precedente multiplo di 4 (rispetto alla dimensione data da terminale), ciò è stato effettutato per garantire un risultato finale apprezzabile.
Sebbene la scelta sia totalmente arbitraria, si sarebbe eventualmente potuto anche creare un padding per riempire le restanti righe fino al prossimo multiplo di 4, in sostituzione alla soluzione da noi adottata.

2.1 Seppia

Per applicare l'effetto seppia viene fatta una scansione di tutta la matrice dei pixel dell'immagine in input, questo per ognuno dei 3 canali RGB.

Per calcolare il nuovo valore del pixel destinazione viene richiamata la funzione *Pixel* che, in base al canale che si sta valutando, effettua la somma dei canali sorgente di quel pixel moltiplicata per specifici fattori che permetteno di modificare il valore donando l'effetto "seppia".

Figura 1: Immagine originale

Figura 2: Effetto Seppia

2.2 Sharp

Per quanto riguarda l'effetto di *sharpening* viene applicato un filtro a convoluzione con il seguente kernel:

```
{  0,   −1,    0,
  −1,    5,   −1,
   0,   −1,    0  }
```

Per ogni pixel della destinazione viene calcolato, tramite la funzione *PixelS*, un valore che rappresenta la somma dei pixels adiacenti a quello che si sta valutando, moltiplicato per il valore del kernel nella posizione corrispondente.

Figura 3: Immagine originale

Figura 4: Effetto Sharp

2.3 Luminosità

Il filtro Luminosità permette l'aumento e la diminuzione della luminosita dell'immagine.

Se il fattore di scala, inserito da input, è negativo allora l'immagine verrà scurita dividendo il valore del pixel sorgente per il valore assoluto del fattore.
Nel caso opposto, quindi se il fattore di scala è positivo, verrà aumentata la luminosità dell'immagine moltiplicando il pixel per il fattore.
In ogni caso il risultato verrà impostato a 255 o 0 se supera questi limiti.

Figura 5: Immagine originale

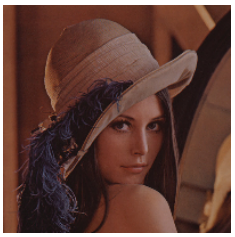

Figura 6: Diminuzione luminosità

Figura 7: Aumento luminosità

2.4 Contrasto

È possibile regolare il contrasto attraverso la funzione *Contrasto* che inizialmente calcola il fattore di correzione del contrasto attraverso la formula:

$$F = \frac{259 \times (C + 255)}{255 \times (259 - C)} \tag{1}$$

C è il valore inserito da input.

Viene poi estratto il pixel dall'immagine sorgente, per ogni canale colore e viene modificato con il fattore di correzione del contrasto calcolato all'inizio.

Figura 8: Immagine originale

Figura 9: Diminuzione contrasto

Figura 10: Aumento contrasto

2.5 Bianco/Nero

Il filtro BN permette la trasformazione dell'immagine da colori a scala di grigi.

Ogni canale di ogni pixel sorgente, viene moltiplicato per un fattore fisso, il valore risultante viene successivamente riassociato ad ogni canale di ogni pixel destinazione.

Figura 11: Immagine originale

Figura 12: Effetto Bianco/Nero

2.6 Blur

Per applicare l'effetto Blur si è utilizzato il filtro Gaussiano.
Il procedimento consiste nel convolvere l'immagine con una risposta all'impulso consistente in una campana di Gauss centrata e normalizzata per aver somma unitaria, si tratta di un filtro separabile, quindi molto efficiente se applicato prima per righe e poi per colonne (o viceversa).

Si osservi che la trasformata di Fourier di una Gaussiana è ancora una Gaussiana, quindi il filtro Gaussiano approssima un filtro passa-basso. Se da una immagine eliminiamo poi le basse frequenze, ottenute con il filtro Gaussiano, rimangono le alte frequenze, ovvero i bordi.

Figura 13: Immagine originale

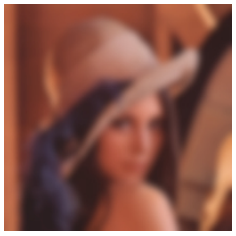

Figura 14: Effetto Blur

2.7 Zoom

Il filtro Zoom, come dice il nome stesso, si occupa di ingradire l'immagine sorgente e produrre in output un' immagine che ha un rapporto d'ingradimento pari al parametro definito dall'utente. Così come la funzione di *Scala* , questo filtro esegue un'interpolazione dei pixels sorgenti causando una degradazione del risultato finale; dunque, con valori alti di zoom, l'immagine diverrebbe sgranata.

Il punto d'origine nel quale si applica il filtro Zoom è stato impostato centralmente, ma è eventualmente possibile parametrizzare i valori X ed Y per spostare la maschera di zoom nel punto desiderato.

Figura 15: Immagine originale

Figura 16: Effetto Zoom 2x

2.8 Negativo

Per ogni pixel dell'immagine sorgente si effettua il calcolo del complementare; 255 viene sottratto al valore del pixel del canale analizzato. Applicato su tutti e tre i canali R-G-B otteniamo l'immagine invertita.

Figura 17: Immagine originale

Figura 18: Effetto Negativo

2.9 Trama

Il filtro Trama consente di modificare l'immagine impiegando un particolare effetto puntiforme. Attraverso un parametro impostato dall'utente è possibile aumentare oppure diminuire la consistenza della trama da applicare all'immagine.

Il filtro in questione si occupa, molto semplicemente, di scorrere la bitmap sorgente saltando a intervalli regolari alcuni pixels. I pixels non letti dalla sorgente sono automaticamente valutati nell'immagine destinazione come pixel privi di colori e di conseguenza neri. Questo intervallo è definito dall' utente e nel codice si traduce in una maggiore o minore quantità di pixel della sorgente non letti.

Figura 19: Immagine originale

Figura 20: Effetto Trama