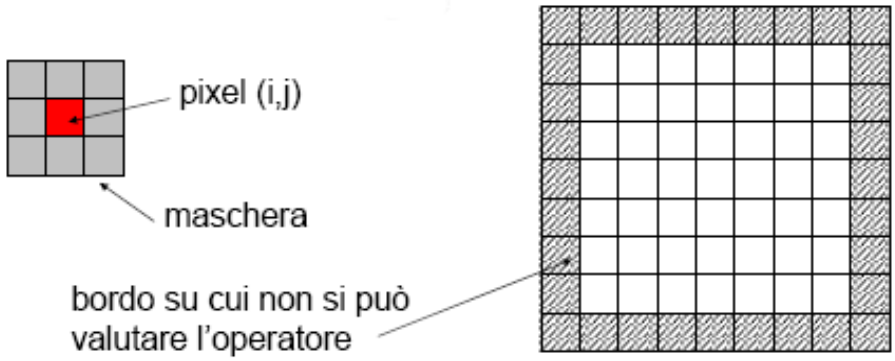
SINA **Dicembre 2005**

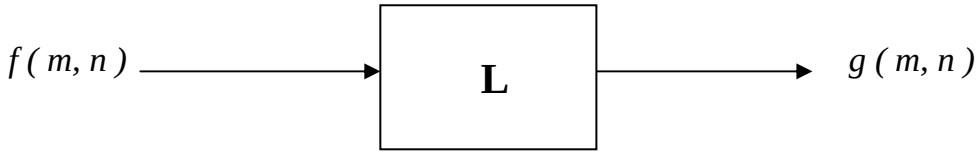
Corso di programmazione – Test Finale

Al fine di migliorare l'intelligibilità di un'immagine digitale, una tecnica di primaria importanza è il filtraggio, attraverso il quale si possono enfatizzare alcune caratteristiche dell'immagine.
Il filtraggio è una funzione di intorno, nel quale il valore che assume un pixel nell'immagine d'uscita è determinato dall'applicazione di un operatore che agisce sui valori dei pixel a lui circostanti nell'immagine di ingresso. Per questo tale operatore viene detto operatore locale.
Interpretando l'immagine come una matrice, l'operatore agisce su una finestra di punti nell'intorno del pixel a cui è applicato. L'operazione di filtraggio calcola la somma pesata dei pixel appartenenti all'intorno e si assegna il risultato al pixel dell'immagine di uscita (per semplicità, d'ora in poi tratteremo solo immagini bianco e nero, in cui l'intensità luminosa corrisponde al valore in toni di grigio dei singoli pixel dell'immagine, dove un valore uguale a 0, corrisponde al colore nero, il valore massimo rappresentabile, ad esempio 255, corrisponde al bianco ed i valori intermedia ai toni di grigio).
Questa operazione viene ripetuta per tutti i pixel dell'immagine in ingresso.
Con questo procedimento i risultati che si ottengono dall'analisi dei pixel di bordo immagine sono affetti da un errore: una parte della finestra del filtro rimane infatti fuori dall'immagine. La regione di bordo sarà tanto più larga quanto più è grande la finestra del filtro applicato.
Un modo per ovviare a questo problema è quello di copiare i pixel che giacciono sul bordo dell'immagine originale direttamente sull'immagine di uscita, in pratica non si applica i filtro sulle zone di bordo.



Il meccanismo della convoluzione

Si consideri il generico sistema **L** in figura, dove $f(m, n)$ costituisce l'ingresso al sistema e $g(m, n)$ ne è l'uscita.



Il filtraggio lineare spazio-invariante porta come risultato un valore dato dalla combinazione lineare dei valori dei pixel dell'intorno del pixel d'ingresso. Considerando l'operatore applicato con una finestra di dimensioni m x n su un'immagine X x Y , si ha

$$g(i, j) = L[f(i, j)] = \sum_{p=0}^{m} \sum_{q=0}^{n} f(i, j) h(i - p, j - q)$$

Ad esempio, per una finestra 3x3 si ha

$$h(m, n) = \begin{bmatrix} h_{-1,-1} & h_{-1,0} & h_{-1,1} \\ h_{0,-1} & h_{0,0} & h_{0,0} \\ h_{1,-1} & h_{1,0} & h_{1,1} \end{bmatrix}$$

$$\begin{aligned} & g(i, j) = \\ & = h_{-1,-1} f(i-1, j-1) + h_{-1,0} f(i-1, j) + h_{-1,+1} f(i-1, j+1) + \\ & + h_{0,-1} f(i, j-1) + h_{0,0} f(i, j) + h_{0,+1} f(i, j+1) + \\ & + h_{+1,-1} f(i+1, j-1) + h_{+1,-1} f(i+1, j) + h_{+1,+1} f(i+1, j+1) \end{aligned}$$

Per evitare che i valori dei singoli pixel, per effetto dell'applicazione del filtro, escano dall'insieme dei valori rappresentabili (ad es. in un immagine a 256 livelli di grigio si usano 8 bit per rappresentare il livello di grigio di un singolo pixel, quindi l'insieme del valori rappresentabili va da 0 a 255) si usa un fattore di normalizzazione M

$$g(i, j) = L[f(i, j)] = \frac{1}{M} \sum_{p=0}^{m} \sum_{q=0}^{n} f(i, j) h(i - p, j - q)$$

La somma dei coefficienti di convoluzione fornisce la costante di normalizzazione:

$$M = \sum_{m} \sum_{n} h_{m,n}$$

A volte ci sono anche numeri negativi nella maschera, è dunque possibile che la somma dei coefficienti di convoluzione dia un risultato nullo. Per implementare quindi un algoritmo di filtraggio adatto per un calcolatore, che non può dividere per zero, bisogna avere l'accortezza di porre la costante di normalizzazione pari a uno se la somma dei coefficienti di convoluzione fosse zero.

Il centro del quadrato corrisponde al pixel che si sta calcolando, mentre gli altri numeri sono i pixel del suo intorno.

Il processo di filtraggio digitale consiste nel moltiplicare ciascun pixel dell'intorno per il coefficiente di convoluzione corrispondente, e dividere il risultato ottenuto per la costante di normalizzazione.

Bisogna sottolineare che possono essere usate maschere di qualsiasi forma e dimensione, purché sia possibile individuarne un centro: ad esempio non si possono utilizzare matrici 2x2 o 4x6. Tuttavia le maschere 3x3 sono quelle comunemente utilizzate, poiché rendono semplici le operazioni di filtraggio.

Edge detection: il filtro di Sobel

Una delle applicazioni dei filtri digitali è quelli di ridurre il contenuto informativo delle immagini al fine di capire quello che contiene, ricavando informazioni a livello aggregato (non sul singolo pixel) con procedure di estrazione delle caratteristiche salienti.
Un tipo di informazione aggregata, utilizzato in numerose tecniche di analisi, è quella relativa all'estrazione dei contorni degli oggetti presenti nella scena. I metodi più semplici estraggono i contorni mediante sistemi derivativi per estrarre le zone di discontinuità di luminosità.
Risultati migliori si possono ottenere con operazioni di tipo derivativo non lineari, ad esempio con l'algoritmo di **Sobel**.
L'algoritmo di Sobel prevede l'applicazione di kernel per l'estrazione dei contorni verticali o orizzontali dell'mmagine:

$$H_{vert} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_{oriz} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Immagine originale

Filtro di Sobel orizzontale

Filtro di Sobel verticale

Noise removal: il filtro Mediano

Un'altra applicazione dei filtri digitali consiste nel miglioramento delle immagini: ad esempio l'applicazione di appositi filtri che eliminano (o riducono) il rumore presente nelle immagini.
Un approccio di questo tipo si avvale del *filtro mediano*, per effetto del qual un pixel è sostituito dal valore mediano dei pixel del suo intorno: il mediano M di un insieme di valori è tale che metà dei valori sono minori di M e metà dei valori sono maggiori di M.
In altre parole, ponendo i valori presenti nell'intono del pixel in un vettore di dimensione M e riordinandoli, si sostituirà al pixel il valore mediano del vettore, ovvero il valore che sta nell'elemento con indice (M/2)
Quindi l'effetto del filtro mediano è di forzare i pixel ad assumere un valore uguale a quello di uno dei pixel circostanti, eliminando così eventuali spike isolati di intensità, che sono quelli con cui di solito si manifesta il cosiddetto *rumore impulsivo*(o "salt-and-pepper").

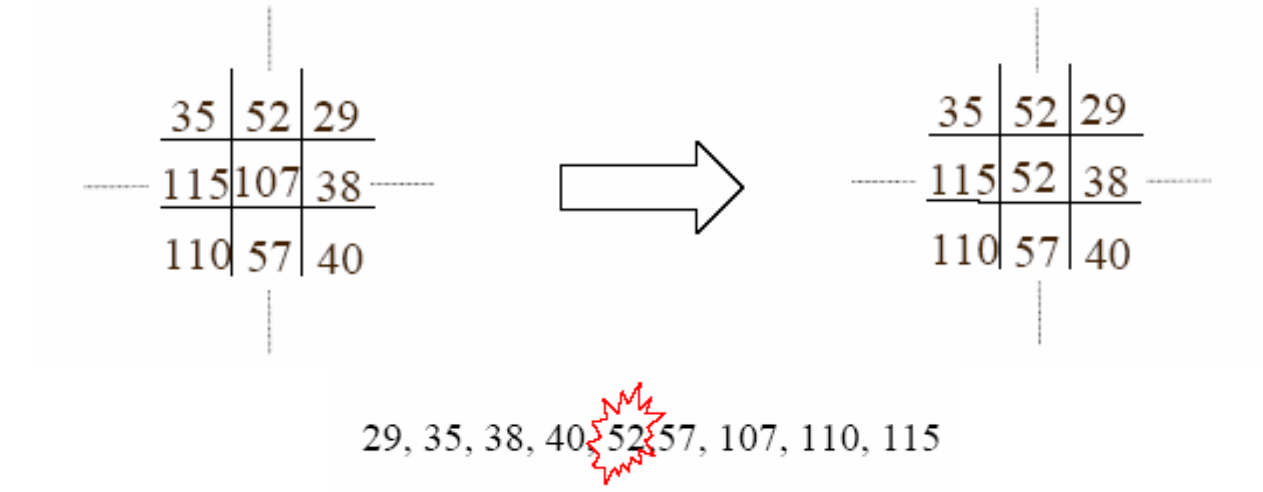



Immagine originale


Immagine con rumore "salt-and-pepper"


Risultato del filtro mediano

Test di programmazione

Sulla base di quanto detto in precedenza, produrre una classe che implementi un filtro digitale definibile a piacere dall'utente.
Tale filtro dovrà avere matrice di convoluzione (ovvero la matrice che contiene i coefficienti da applicare all'immagine) di dimensione a piacere, purché quadrata.
Il coefficiente di normalizzazione sarà calcolato automaticamente (sulla base di quanto detto sopra) a partire dai valori dei coefficienti della matrice
E' fornita con il presente testo una classe (chiamata *CPGMImage*), debitamente documentata, che si occupa di aprire un file contenente l'immagine in formato PGM (Portable Gray Map) e di organizzare l'immagine in memoria in un'apposita struttura dati e funzioni membro in grado di produrre le dimensioni dell'immagine, il valore del tono di grigio di un singolo pixel e modificarlo, così come salvare l'immagine in un file di output.
Il candidato non dovrà occuparsi dei dettagli dell'implementazione della suddetta classe, dovrà solo utilizzarla nel suo progetto, basandosi sulle specifiche fornite nella documentazione.
La classe che si richiede dovrà contenere:

- Costruttore con parametri: vengono passati a questo costruttore le dimensioni della matrice di convoluzione e la matrice di convoluzione stessa (organizzata in un vettore bidimensionale a valori interi). I coefficienti della matrice dovranno

essere copiati in un'apposita struttura dati privata presente nella classe. Verrà assegnato di conseguenza anche il valore del coefficiente di normalizzazione, calcolato in base ai valori presenti nella matrice.

- Costruttore vuoto: crea un'istanza della classe senza inizializzare i valori della matrice di convoluzione.
- Funzione che permette di settare la matrice di convoluzione (utile nel caso si sia creata l'istanza tramite il costruttore vuoto o nel caso in cui si vogliano modificare i coefficienti). Questa funzione modificherà il coefficiente di normalizzazione in maniera opportuna.
- Funzione che permette di recuperare la matrice di convoluzione, copiando i dati in un'apposita struttura dati (omogenea a quella usata nel costruttore con parametri) passata come parametro alla funzione e già allocata di dimensioni opportune.
- Funzione che applica il filtro, a cui verranno passate come parametri due istanze della classe immagine, debitamente allocate e dimensionate correttamente, una per l'immagine originale ed una per l'immagine risultato dell'applicazione del filtro (l'immagine originale non deve essere modificata, ma il risultato del filtraggio deve essere scritto nell'immagine di destinazione)

Porre particolare attenzione all'applicazione del filtro sulle zone di bordo dell'immagine, inoltre inserire nel codice che implementa le funzioni di cui sopra le verifiche sulla loro applicabilità, quali:

- applicazione del filtro su immagini non debitamente dimensionate (dimensione nulla oppure dimensioni dell'immagine di destinazione non conformi all'originale)
- applicazione del filtro senza aver definito una matrice di convoluzione.
- dimensioni della matrice di convoluzione errate rispetto all'immagine su cui si sta tentando di applicare il filtro.
- qualsiasi altra errata applicazione delle funzioni membro della classe, che pregiudicherebbero l'esecuzione del codice presente nelle stesse.

Oltre alla classe di cui sopra, il candidato dovrà fornire un programma che la utilizza, che operi nel seguente modo:

1. crei due istanze della classe di filtro generico, assegnando ad una la matrice di convoluzione corrispondente al filtro di Sobel verticale e all'altra quello orizzontale
2. carichi il memoria (usando la classe per immagini fornita) l'immagine presente nel file "*original.pgm*" (anch'esso fornito con il presente testo)
3. applichi i due filtri precedentemente creati su questa immagine, ottenendo due immagini risultato dell'applicazione dei filtri di Sobel (che devono essere debitamente create e dimensionate in precedenza)
4. salvi le due immagini così ottenute su disco.

Aggiunta per il lavoro in gruppo (max. 2 persone)

Sulla base di quanto detto in precedenza, produrre una classe che implementi un filtro digitale di tipo mediano per la rimozione del rumore "salt-and-pepper".

E' fornita con il presente testo una classe (chiamata *CVectSort*), debitamente documentata, che si occupa di ordinare un vettore di interi che gli viene passato come parametro e di estrarne il valore mediano. Il candidato non dovrà occuparsi dei

dettagli dell'implementazione della suddetta classe, dovrà solo utilizzarla nel suo progetto, basandosi sulle specifiche fornite nella documentazione.
L'intorno su cui calcolare il valore mediano dovrà essere una matrice di pixel di dimensione 3x3.
La classe che si richiede dovrà contenere:

- Costruttore e distruttore: il costruttore della classe non necessita di alcun parametro, poiché l'intorno del pixel (e quindi la dimensione del vettore di interi da ordinare) su cui calcolare il filtro è preso a dimensione fissa di 9 elementi (una matrice 3x3).
- Funzione che applica il filtro, a cui verranno passate come parametri due istanze della classe immagine, debitamente allocate e dimensionate correttamente, una per l'immagine originale ed una per l'immagine risultato dell'applicazione del filtro (l'immagine originale non deve essere modificata, ma il risultato del filtraggio deve essere scritto nell'immagine di destinazione)

Per ogni altra osservazione circa l'implementazione generale della classe, si rimanda al testo dell'esercizio sul filtro di Sobel.
Oltre alla classe di cui sopra, il candidato dovrà fornire un programma che la utilizza, che operi nel seguente modo:

1. crei un'istanza della classe di filtro mediano
2. carichi il memoria (usando la classe per immagini fornita) l'immagine presente nel file "*noise.pgm*" (anch'essa fornita con il presente testo)
3. crei un'immagine vuota delle dimensioni di quella originale appena caricata, che conterrà il risultato dell'applicazione del filtro
4. applichi il filtro sull'immagine "*noise.pgm*"
5. salvi l'immagine risultato del filtro su disco.

É richiesta la consegna di:

- Il codice C++ che implementa la classe del filtro generico e del filtro mediano (solo nel caso di lavoro di gruppo).
- La documentazione delle classi di cui sopra, consistente in un file di testo che descrive il funzionamento delle funzioni membro richieste sopra, in modo che un ipotetico utilizzatore della classe sia in grado di usarla (descrizione dello scopo della funzione, se necessario accenno al suo funzionamento, valori di input richiesti e dati di output prodotti) sul genere di quella fornita in allegato per la classe immagine. Utilizzare come modello per tale documentazione quella fornita con la classe immagine PGM.
- Il programma che esegue il filtro di sobel verticale e orizzontale sull'immagine "*original.pgm*".
- Il programma che applica il filtro mediano per la riduzione del rumore all'immagine "*noise.pgm*" (solo nel caso di lavoro di gruppo).