Sistemi Intelligenti Naturali e Artificiali a.a. 2008/2009

Esercitazione 2: istogrammi

Testo.
Scrivere le funzioni per:
a) calcolare l'istogramma di un'immagine
b) eseguire lo "stretch" dell'istogramma di un'immagine. I valori di *min* e *max* per l'operazione di stretch vengono passati come parametri alla funzione.
c) equalizzare l'istogramma di un'immagine

In tutti i casi e' utile (fatelo), scrivere una funzione che disegna un istogramma su un'immagine di tipo *ImageClassMono*, cosi' che poi sia possibile salvarlo e visualizzarlo.

Testare le funzioni nel seguente modo:

1) Aprire l'immagine *lena.pgm* (nella cartella "immagini" fornita con l'esercitazione).
2) Calcolare l'istogramma e salvarlo su un'immagine pgm (es: *lena-histo.pgm*)
3) Eseguire lo stretch dell'istogramma tra due valori a piacere. Calcolatene l'istogramma. Salvare immagine e istogramma cosi' ottenuti.
4) Equalizzare l'istogramma dell'immagine di partenza (*lena.pgm*). Calcolare l'istogramma. Salvare immagine e istogramma cosi' ottenuti.

Si veda anche la traccia della funzione *main* fornita piu' sotto.

Traccia di soluzione.
Creare una nuova cartella e create i file per compilare il codice (makefile e main.cpp). Potete utilizzare i file dell'esercitazione 1 e modificarli di conseguenza.

- La funzione per il calcolo dell'istogramma avra' come prototipo:

*void computeHisto(ImageClassMono &img, int *histo);*

Dove *img* e' l'immagine della quale si vuole calcolare l'istogramma (toni di grigio), mentre *histo* e' il vettore che conterra' l'istogramma. La funzione *computeHisto* deve andare a riempire i valori *histo[0] .. histo[255]*, in maniera opportuna (si veda quanto fatto a lezione). A tal fine e' utile utilizzare la funzione *get(r,c)* implementata durante l'esercitazione precedente.

Per visualizzare il risultato scrivere una funzione che disegna un istogramma in un'immagine *ImageClassMono*:

*void drawHistogram(int *histo, ImageClassMono &img);*

La funzione in questo caso deve convertire i valori dell'istogramma contenuti nel vettore *histo* in un'immagine 2D.

- La funzione di *stretch* dell'istogramma ha il seguente prototipo:

void stretchHistogram(ImageClassMono &in, ImageClassMono &out, int min, int max);

Dove *in* e' l'immagine di partenza e *out* l'immagine ottenuta dopo l'operazione di stretch dell'istogramma. *min* e *max* sono invece i valori da utilizzare per la creazione della tabella di look up.

- Prototipo funzione di equalizzazione:

void equalizeHistogram(ImageClassMono &in, ImageClassMono &out);

Dove *in* e' l'immagine di partenza e *out* l'immagine ottenuta dopo l'operazione di equalizzazione dell'istogramma.

Programma main:
Per testare le funzioni precedenti realizzate il seguente programma:

```
int main() {
        ImageClassMono img;

        // legge l'immagine dal disco
        ReadPgm(img, "lena.pgm");

        // allocazione del vettore che conterra' l'istogramma
        int *histogram=new int [256];

        // calcola l'istogramma
        computeHistogram(img, histogram);

        // disegna l'istogramma su un'immagine 256x256
        ImageClassMono histoImg;
        histoImg.resize(256, 256);
        drawHistogram(histogram, histoImg);

        // salva l'istogramma su disco
        SavePgm(histoImg, "lena-histo.pgm");

        // stretch dell'istogramma
        ImageClassMono stretched;
        stretched.resize(img.getWidth(), img.getHeight());
        stretchHistogram(img, stretched, 10, 230);

        computeHistogram(stretched, histogram);
        drawHistogram(histogram, histoImg);
        SavePgm(histoImg, "stretched-histo.pgm");

        // equalizzazione
        ImageClassMono equalized;
        equalized.resize(img.getWidth(), img.getHeight());

        equalizeHistogram(img, equalized):
        computeHistogram(equalized, histogram);
        drawHistogram(histogram, histoImg);
        SavePgm(histoImg, "equalized-histo.pgm");

        // distrugge il vettore dell'istogramma
        delete [] histogram;
}
```