Elaborazione di Segnali Multimediali
a.a. 2017/2018

Elaborazioni nel dominio spaziale (2)

L.Verdoliva

In questo laboratorio proseguiamo lo studio sulle elaborazioni spaziali, in particolare studieremo il bit-plane slicing ed esamineremo sia le operazioni aritmetiche che quelle geometriche.

1 Bit-plane slicing

Consideriamo un'immagine in cui ogni livello è rappresentato su 8 bit. E' possibile suddividere l'immagine in bit-plane, cioè in piani in cui si rappresentano ognuno dei bit da quello meno significativo (bit-plane 1) a quello più significativo (bit-plane 8). La decomposizione di un'immagine digitale in bit-plane (*bit-plane slicing*) è molto utile per comprendere l'importanza che ogni bit ha nel rappresentare l'immagine e quindi se il numero di bit usato nella quantizzazione è adeguato. Estraiamo i bit-plane dell'immagine `frattale.jpg` usando il comandi matlab `bitget`:

```
B = bitget(x,8);      % estrazione bit-plane più significativo
imshow(B,[0 1]);      % visualizzazione
```

Scrivete uno script dal nome bit_plane.m in cui estraete e visualizzate tutti i bit-plane dell'immagine. Potete memorizzare i bit-plane in una struttura tridimensionale in cui Bi = bitplane(:,:,i).

1.1 Esercizi proposti

1. *Ricostruzione mediante bit-plane*. Ponete a zero i bit-plane meno significativi di un'immagine (usate la funzione `bitset` di matlab) e visualizzare il risultato, al variare del numero di bit-plane che utilizzate nel processo di ricostruzione. Questo esperimento vi permette di stabilire fino a che punto (almeno da un punto di vista percettivo) è possibile diminuire il numero di livelli usati nel processo di quantizzazione.

2. *Esempio di Watermarking*. Provate adesso a realizzare una forma molto semplice di Watermarking, che consiste nell'inserire una firma digitale all'interno di un'immagine. Sostituite il bit-plane meno significativo dell'immagine lena.y con l'immagine binaria marchio.y, quest'ultima ha dimensioni 350×350 quindi è necessario estrarre una sezione delle stesse dimensioni dell'immagine lena.y. Provate poi a ricostruire l'immagine e visualizzatela, noterete che da un punto di vista visivo l'immagine non ha subito modifiche percettibili.

2 Operazioni aritmetiche

Le operazioni aritmetiche (somma/sottrazione, prodotto/divisione) coinvolgono una o più immagini e si effettuano pixel per pixel. In matlab sono molto semplici da realizzare, dato che è possibile effettuare queste operazioni direttamente sulle matrici.

In particolare, fare la sottrazione tra due immagini vi permette di scoprire le differenze che esistono tra le due. Provate allora a visualizzare l'immagine frattale.jpg, e quella in cui sono stati posti a zero i 4 bit-plane meno significativi. Noterete come da un punto di vista visivo sono molto simili, fatene allora la differenza e visualizzatela a schermo.

3 Operazioni geometriche

Le operazioni geometriche consentono di ottenere, a partire da un'immagine x una nuova immagine y nella quale i valori di luminosità sono gli stessi, ma sono prelevati in posizioni diverse da quelle originali, in modo da cambiare solo la *forma* degli oggetti.

3.1 Ridimensionamento

Rimpicciolire un'immagine (*zoom in*) di un fattore intero è estremamente semplice da realizzare in matlab. Supponiamo per esempio di volerla dimezzare, allora:

```
[M,N]=size(x);
y=x(1:2:M,1:2:N);              % decimazione per 2 lungo le due dimensioni
imshow(y, [0 255]);            % visualizzazione
```

Questa operazione ci permette di modificare la risoluzione spaziale dell'immagine e consiste di fatto nell'abbassare (in numerico) la frequenza di campionamento del segnale. Se volessimo invece rimpicciolirla di un fattore non intero, realizzando per esempio la trasformazione $y(m,n) = x\left(\frac{3}{2}m, \frac{3}{2}n\right)$ bisogna fare più attenzione perchè occorre assegnare correttamente i valori di intensità in uscita, come per esempio $y(1,1) = x\left(\frac{3}{2}, \frac{3}{2}\right)$; quest'ultimo valore non è definito nell'immagine in ingresso per cui bisogna determinarlo mediante interpolazione usando la funzione `interp2`. La sintassi di questa funzione è

```
y=interp2(n,m,x,np,mp,metodo);
```

dove `n` e `m` rappresentano le coordinate della griglia su cui è definita l'immagine di partenza `x`, `np` e `mp` sono invece le coordinate dei valori che vogliamo determinare e `metodo` è il tipo di interpolazione. Con l'opzione `nearest` si effettua un'interpolazione nearest neighbor, mentre con `linear` di tipo bilineare. Nel nostro caso:

```
[M,N] = size(x);
[n,m] = meshgrid(1:N,1:M);
[np,mp] = meshgrid(1:3/2:N,1:3/2:M);
y = interp2(n,m,x,np,mp,'linear');
imshow(y, [0 255]);
```

La funzione `meshgrid` di Matlab permette facilmente di creare la matrice in cui in ogni posizione ci sono le coordinate spaziali, in particolare `n` è una matrice $M \times N$ che contiene il valore della coordinata spaziale

lungo la direzione verticale, `m` analogamente conterrà i valori lungo la direzione orizzontale. Attenzione, i comandi `meshgrid` e `interp2` interpretano le coordinate come *ascisse e ordinate* non come righe e colonne. Ecco perché bisogna invertire l'ordine di `m` e `n`: infatti l'indice di riga, comportando uno spostamento verticale equivale all'ordinata (viceversa per l'indice di colonna).

In Matlab è possibile effettuare una trasformazione geometrica affine anche specificando direttamente la matrice di trasformazione $\mathbf{A}$ attraverso il comando `affine2d`. Supponiamo di voler ingrandire una sezione di 25×50 pixel intorno all'occhio di lena:

```
x = double(imread('lena.jpg'));
x = x(253:277,241:290);
A = [2 0 0; 0 2 0; 0 0 1];
tform = affine2d(A);
y1 = imwarp(x, tform, 'nearest');
y2 = imwarp(x, tform, 'bilinear');
y3 = imwarp(x, tform, 'bicubic');
subplot(3,1,1); imshow(y1,[0 255]); title('interpolazione nearest');
subplot(3,1,2); imshow(y1,[0 255]); title('interpolazione bilinear');
subplot(3,1,3); imshow(y1,[0 255]); title('interpolazione bicubic');
```

Notate l'effetto di blocchettatura causato dall'interpolazione con opzione 'nearest' rispetto a 'bilinear' e 'bicubic'. Fate attenzione al fatto che i valori da usare per il ridimensionamento devono essere pari all'inverso di quelli definiti nella trasformazione di coordinate. Questo è legato al fatto che in Matlab la definizione di matrice affine è diversa da quella che abbiamo usato in teoria. In particolare, teoricamente:

$$[m', n', 1] = [m, n, 1]\,\mathbf{T}$$

mentre in Matlab si ha:

$$[n, m, 1] = [n', m', 1]\,\mathbf{A}$$

Ci sono quindi due differenze fondamentali: un'inversione di righe e colonne e un'inversione della matrice stessa. I comandi Matlab che ci permettono di ottenere $\mathbf{A}$ a partire da $\mathbf{T}$ sono i seguenti:

```
A = inv(T([2,1,3],[2,1,3]));
```

Nel toolbox image è presente la funzione `imresize` che permette di effettuare il ridimensionamento di un'immagine. Per esempio se si vuole rimpicciolire l'immagine di un fattore 0.75 con interpolazione bilineare:

```
y = imresize(x, 0.75, 'bilinear');
imshow(y, [0 255]);
```

Chiaramente si può anche ingrandire l'immagine se il fattore scelto è maggiore di 1; inoltre, anziché specificare tale fattore si possono fissare le dimensioni che deve avere la nuova immagine (se però non si fa attenzione a conservare il rapporto d'aspetto, si crea distorsione nell'immagine).

3.2 Traslazioni e rotazioni

Proviamo a realizzare la traslazione di un'immagine e scriviamo il codice definendo le griglie e usando `interp2`. Se vogliamo traslare verso sinistra e verso l'alto l'immagine considerata, il codice è:

```
x = double(imread('lena.jpg'));
[M,N] = size(x);
[n,m] = meshgrid(1:N,1:M);
mp = m+50; np = n+100;
y = interp2(n,m,x,np,mp,'bilinear');
subplot(1,2,1); imshow(x,[0 255]); title('originale');
subplot(1,2,2); imshow(y,[0 255]); title('traslata');
```

Fate attenzione all'uso del comando `imwarp` quando volete realizzare una traslazione. Provate infatti a digitare i comandi:

```
x = double(imread('lena.jpg'));
T = [1 0 0; 0 1 0; -100 -50 1];
tform = affine2d(T);
y = imwarp(x, tform);
subplot(1,2,1); imshow(x,[0 255]); title('originale');
subplot(1,2,2); imshow(y,[0 255]); title('traslata');
```

L'immagine traslata risulta identica all'originale e questo perché di default la funzione `imwarp` determina il rettangolo in cui mostrare l'uscita usando il sistema di coordinate proprio dell'uscita. Definendo opportunamente il parametro `OutputView` potete stabilire in quale spazio di uscita specificare il risultato. Se si vuole ottenere l'immagine nella stessa regione dell'immagini originale:

```
Rx = imref2d(size(x));
y = imwarp(x, tform, 'OutputView', Rx);
```

La funzione `imref2d` serve per definire una regione dello spazio. E' anche possibile modificare il colore per i pixel esterni al dominio dell'immagine. Se per esempio si vuole che abbiano colore bianco:

```
y = imwarp(x, tform, 'OutputView', Rx, 'FillValues', 255);
```

Oppure potete inserire una gradazione di grigio specificando un valore tra 0 (nero) e 255 (bianco).
Nel caso in cui si vuole effettuare la rotazione di un'immagine:

```
x = double(imread('lena.jpg'));
T = [cos(pi/4) sin(pi/4) 0; -sin(pi/4) cos(pi/4) 0; 0 0 1];
tform = affine2d(T);
y = imwarp(x, tform);
subplot(1,2,1); imshow(x,[0 255]); title('originale');
subplot(1,2,2); imshow(y,[0 255]); title('ruotata');
```

Confrontate questo risultato con quello che ottereste direttamente con la funzione `imrotate`.

3.3 Esercizi proposti

1. *Distorsione*. Scrivete la funzione che realizza la distorsione di un'immagine lungo la direzione verticale e orizzontale e che abbia il prototipo: `function y=deforma(x,c,d)`. Scegliete un'immagine e al variare dei parametri `c` e `d` osservate il tipo di distorsione.

2. *Combinazione di operazioni geometriche*. La combinazione di diverse trasformazioni affini è ancora una trasformazione affine, che può essere ottenuta tramite il prodotto (matriciale) delle matrici che le definiscono. Scrivete allora una funzione dal prototipo `function y=rot_shear(x,theta,c)` per realizzare una rotazione e poi una distorsione verticale (attenzione all'ordine!). Create l'immagine di ingresso usando il seguente comando `x = checkerboard(50);` in modo da generare una scacchiera su cui le modifiche risultano essere più facilmente visibili.