

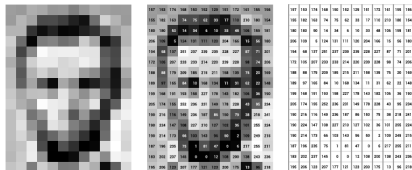
## Derivazione numerica per 'Edge detection'

Data un'immagine a colori, si vogliono identificare i pixel in un intorno dei quali si ha un brusco cambiamento dell'intensità di colore, per isolare regioni diverse dell'immagine.



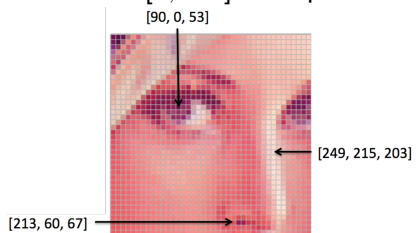
# Immagini come funzioni

Ad ogni pixel di un'immagine bw è associato un valore in  $[0, 255]$



<https://ai.stanford.edu/~syeyung/cvweb/tutorial1.html>

mentre ad ogni pixel di un'immagine a colori è associata una terna di valori in  $[0, 255]$  corrispondenti ai tre colori RGB



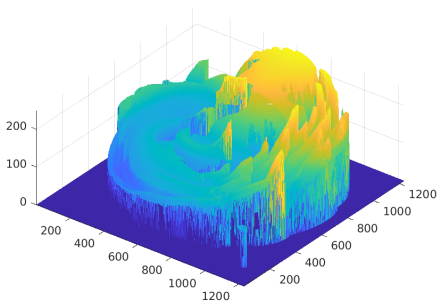
<https://ai.stanford.edu/~syeyung/cvweb/tutorial1.html>

pixel  $(j, i)$   $\leftrightarrow$  punto  $(x, y)$   
intensità di colore  $\leftrightarrow$  funzione  
 $color(j, i)$   $\leftrightarrow$   $f(x, y)$

# Immagini come funzioni



immagine



intensità di colore =  $f(x, y)$

Bruschi cambi di colore in corrispondenza di forti variazioni della funzione *intensità di colore*



voglio 'taggare' i pixel corrispondenti a  $\left| \frac{\partial f}{\partial x} \right|$  e/o  $\left| \frac{\partial f}{\partial y} \right|$  grandi.

# Osservazioni

- ▶ Non conosciamo l'espressione della funzione

$$f(x, y) = \textit{intensita' di colore}$$

ma ne conosciamo i valori in corrispondenza dei pixel, cioè conosciamo la funzione solo per punti.

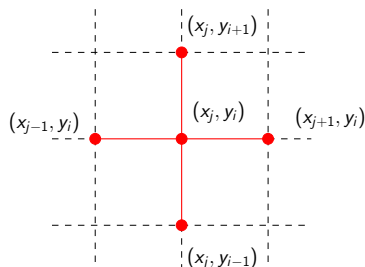
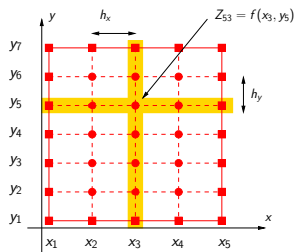
- ▶ Dobbiamo approssimare le derivate con schemi numerici che sfruttino i valori della funzione in determinati punti (i pixel).

# Calcolo numerico di derivate

L'intensità di colore  $f$  è nota solo per punti (nei pixel), quindi:

$$\frac{\partial f}{\partial x}(x_j, y_i) \simeq \frac{f(x_{j+1}, y_i) - f(x_{j-1}, y_i)}{2h_x} \quad \text{punti su una riga fissata}$$

$$\frac{\partial f}{\partial y}(x_j, y_i) \simeq \frac{f(x_j, y_{i+1}) - f(x_j, y_{i-1})}{2h_y} \quad \text{punti su una colonna fissata}$$



$h_x = h_y = 1$  è la distanza tra i pixel

# Come selezionare i punti dei contorni

1. seleziono i pixel dove la **norma del gradiente** dell'intensità di colore si discosta molto dal suo valore medio.

Norma del gradiente di  $f(x, y)$ :

$$\|\nabla f(x, y)\|_2 = \sqrt{\left(\frac{\partial f}{\partial x}(x, y)\right)^2 + \left(\frac{\partial f}{\partial y}(x, y)\right)^2}$$

2. seleziono i pixel dove il valore assoluto del **laplaciano** dell'intensità di colore si discosta molto dal suo valore medio.

Laplaciano di  $f(x, y)$ :

$$\Delta f(x, y) = \frac{\partial^2 f}{\partial x^2}(x, y) + \frac{\partial^2 f}{\partial y^2}(x, y)$$

# Immagini in matlab

Scaricare *mozilla-logo-617.png* dalla pagina del corso.

```
% per leggere l'immagine dal file
```

```
filename='mozilla-logo';
```

```
type='png';
```

```
A=imread(filename,type);
```

```
% per visualizzare il contenuto di A
```

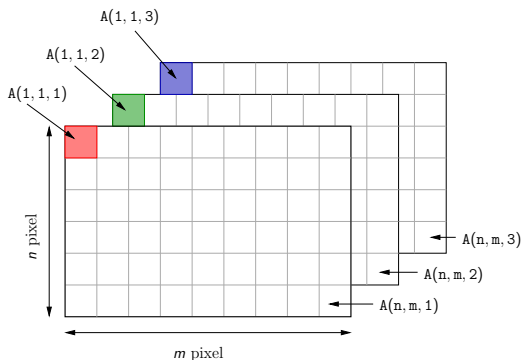
```
figure(1); clf; image(A);
```

```
% per salvare il contenuto di A nel file new_image.png
```

```
imwrite(A,'new_image',type);
```

## Come Matlab gestisce le immagini

Un'immagine di  $m \times n$  pixel è memorizzata in un array a 3 indici  
 $A = (n, m, 3)$



$n$  = numero pixel lungo la  $y$  = numero di righe di  $A$

$m$  = numero pixel lungo la  $x$  = numero di colonne di  $A$

Con intensità di colore nella posizione  $(i, j)$  definiamo la media aritmetica dei tre valori di R, G, B nel pixel  $(i, j)$ .



A è di tipo uint8: ogni elemento occupa 8bit e può assumere solo valori interi tra 0 e 255.

```
>> whos
```

Name	Size	Bytes	Class
A	617x617x3	1142067	uint8

```
% per copiare le tre pagine di A in 3 matrici  
% di tipo double (su cui fare i conti)
```

```
R=double(A(:,:,1));  
G=double(A(:,:,2));  
B=double(A(:,:,3));
```

```
% per calcolare la media
```

```
Z=(R+G+B)/3;
```

## Per cominciare

1. visualizzare le tre componenti di  $A$  ( $R$ ,  $G$ ,  $B$ ) con il comando `surf` in tre diverse figure, in rosso, verde e blue.
2. calcolare l'intensità di colore  $Z_{ij}$  e visualizzarla in una nuova finestra con il comando `mesh`.

**Osservazione.** Il comando

```
mesh(Z)
```

produce un'immagine ribaltata lungo  $y$ . Quindi, sostituire con

```
mesh(Z(n:-1:1, :)); % e non mesh(Z)
```

## Per generare la mappa di colori su una sola componente

Definiamo una matrice nulla  $64 \times 3$  e in prima colonna (corrispondente al R), salviamo 64 valori equispaziati in  $[0, 1]$ :

```
color=zeros(64,3);  
color(:,1)=linspace(0,1,64); % prima colonna = R
```

Per rappresentare la componente rossa dell'immagine:

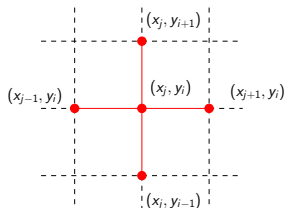
```
figure;  
p=surf(R(n:-1:1,:));  
% per togliere le linee nere della griglia  
set(p,'Edgecolor','none')  
colormap(color) % per selezionare la mappa di colori  
view([0,90]) % vista dall'alto 2D
```

Procedere analogamente con gli altri colori. Ogni volta bisogna rigenerare la matrice color, la colonna 2 per il G, la colonna 3 per il B.

## Approssimazione delle derivate parziali

Uso la formula delle differenze finite centrate. Ricordo che:  
 $m$ =numero pixel lungo la direzione  $x$  (=numero colonne),  
 $n$ =numero pixel lungo la direzione  $y$  (=numero righe),  
non posso calcolare la d.f. centrata sui pixel della corona più esterna.

```
gx=zeros(n,m); gy=zeros(n,m);  
for i=2:n-1  
    for j=2:m-1  
        gx(i,j)=(Z(i,j+1)-Z(i,j-1))/2;  
        % le righe delle immagini sono ordinate alto-basso  
        gy(i,j)=-(Z(i+1,j)-Z(i-1,j))/2;  
    end  
end
```



## Come selezionare i punti “giusti”

Sia  $g(x, y) = \|\nabla f(x, y)\|_2$ , e  $\bar{g}$  è il valor medio di  $g$  su tutti i pixel.  
Per isolare i contorni, dobbiamo selezionare i punti  $(x, y)$  in cui il valore  $g(x, y)$  si discosta sensibilmente dal valor medio  $\bar{g}$ .  
Un criterio possibile è: seleziono i punti  $(x, y)$  in cui

$$\frac{|g(x, y) - \bar{g}|}{\bar{g}} > 0.5$$
$$\Downarrow$$
$$g(x, y) > 1.5 \bar{g}.$$

```

% norma del gradiente
norma_grad=sqrt(gx.^2+gy.^2);
% valor medio
valor_medio_grad=sum(sum(norma_grad))/(n*m);
% inicializzo una matrice con pixel tutti bianchi
edge_grad=255*uint8(ones(n,m));
% se la norma del gradiente in un pixel e' alta,
%     salvo uno 0 (nero)
for j=1:m
    for i=1:n
        if norma_grad(i,j)> 1.5*valor_medio_grad
            edge_grad(i,j)=0;
        end
    end
end
% visualizzo edge_grad
figure;
image(edge_grad); colormap(gray(256))

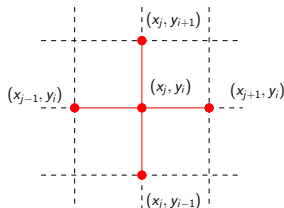
```

# Approssimazione del laplaciano

Calcolo il laplaciano con la formula delle differenze finite centrate:

$$\begin{aligned}\Delta(x_j, y_i) &= \frac{\partial^2 f}{\partial x^2}(x_j, y_i) + \frac{\partial^2 f}{\partial y^2}(x_j, y_i) \\ &\simeq \frac{f(x_{j+1}, y_i) - 2f(x_j, y_i) + f(x_{j-1}, y_i)}{h^2} + \\ &\quad \frac{f(x_j, y_{i+1}) - 2f(x_j, y_i) + f(x_j, y_{i-1})}{h^2}\end{aligned}$$

e poi il suo valore assoluto.



```

abs_lap=zeros(n,m);
for i=2:n-1
  for j=2:m-1
    abs_lap(i,j)=abs((Z(i,j+1)-2*Z(i,j)+Z(i,j-1))/2+...
                    (Z(i+1,j)-2*Z(i,j)+Z(i-1,j))/2);
  end
end

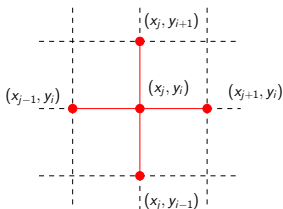
```

e il valor medio:

```

valor_medio_lap=sum(sum(abs_lap))/(n*m);

```





per isolare i contorni, dobbiamo selezionare i pixel  $(i,j)$  in cui il valore `abs_lap` si discosta sensibilmente dal `valor_medio_lap`. Un criterio possibile è:

```
abs_lap(i,j) > 1.5*valore_media_lap
```

```
% inizializzo una matrice con pixel tutti bianchi  
edge_lap=255*uint8(ones(n,m));  
% se abs(lap) in un pixel e' alto, salvo uno 0 (nero)  
for j=1:m  
    for i=1:n  
        if abs_lap(i,j)> 1.5*valor_medio_lap  
            edge_lap(i,j)=0;  
        end  
    end  
end  
% visualizzo edge_lap  
figure;  
image(edge_lap); colormap(gray(256))
```