

# **Appunti Elaborazione delle Immagini**

## **Contattami:**

- personale: gianluca[at]scarpellini.dev
- campus: g.scarpellini1[at]campus.unimib.it

## **Papers e materiale**

<b>Definitions .....</b>	<b>5</b>
<i>Weber ratio.....</i>	6
<b>Filtri, istogramma, smoothing 4 ottobre.....</b>	<b>10</b>
<i>Rumore:.....</i>	11
<i>Come prendere una zona di pixel?.....</i>	11
<i>Filtro mediano .....</i>	13
<i>Alpha trimmed.....</i>	13
<i>Altri filtri di rango .....</i>	13
<i>Filtri con maschera rotante.....</i>	13
<b>Edge e sharpening (10/10) .....</b>	<b>14</b>
<b>Trasformazione geometrica (10/10) .....</b>	<b>16</b>
<b>Il colore .....</b>	<b>18</b>
<b>Algoritmi per bilanciare i colori .....</b>	<b>19</b>
<i>White balance .....</i>	19
<i>Grey world.....</i>	19
<i>White point estimation: max RGB .....</i>	19
<i>Modelli del colore.....</i>	20
<b>Segmentazione .....</b>	<b>22</b>
<i>Importante: cosa si impara?.....</i>	22
Introduzione .....	22
Segmentazione region-based .....	23
Segmentazione a soglia .....	25
Compensazione dello sfondo - illuminazione.....	26
Segmentazione per regioni .....	27
Segmentazione immagini a colori .....	28
Segmentazione nello spazio delle caratteristiche.....	30
<b>Texture .....</b>	<b>32</b>
<b>Morfologia matematica.....</b>	<b>42</b>
<i>Dilation .....</i>	43
<i>Erosion .....</i>	43
<i>Opening.....</i>	43
<i>Closing .....</i>	43
<i>Thinning .....</i>	44
<i>Edge detection .....</i>	44
<i>Trasformata hit-and-miss .....</i>	44
<i>Trasformata distanza .....</i>	44

<i>Hat transform</i> .....	45
<b>Edge detection</b> .....	<b>46</b>
<i>Canny edge detector</i> .....	47
<b>Edge linking</b> .....	<b>49</b>
<i>Rappresentazione e descrizione</i> .....	49
<b>Features</b> .....	<b>52</b>
Contorno.....	52
Chain code .....	52
Approssimazione poligonale .....	53
Signature .....	53
Minimo insieme convesso .....	53
Minimo Rettangolo Orientato .....	53
Trasformata distanza .....	54
MAT (Medial Axis Transform) .....	54
Proiezione .....	54
Scheletrizzazione .....	55
Descrizione di regioni .....	56
<b>Classificazione</b> .....	<b>59</b>
<i>Approccio statistico</i> .....	59
<i>Classificatore</i> .....	61
<i>Semplificare</i> .....	62
<i>Matching</i> .....	62
<i>Chain Code</i> .....	62
<i>Descrizione di una regione</i> .....	64
<i>Numeri di Eulero:</i> .....	65
<i>Momenti di inerzia</i> .....	65
<i>Trasformata distanza</i> .....	66
<i>Scheletrizzazione</i> .....	66



# Definitions

**Image restoration** is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation. There is no general “theory” of image enhancement. When an image is processed for visual interpretation, the viewer is the ultimate judge of how well a particular method works. For this reason, and also because beginners in the field of image processing generally find enhancement applications visually appealing, interesting, and relatively

**Image enhancement** is the process of manipulating an image so that the result is more suitable than the original for a specific application. The word **specific** is important here, because it establishes at the outset that enhancement techniques are problem oriented

**Photopic vision** is the vision of the eye under well-lit conditions (luminance level  $10$  to  $10^8$  cd/m<sup>2</sup>). In humans and many other animals, photopic vision allows color perception, **mediated by cone cells**, and a significantly higher visual acuity and temporal resolution than available with scotopic vision.

**Scotopic vision** is the vision of the [eye](#) under low light conditions.

The human eye uses three types of cones to sense light in three bands of color. The biological pigments of the cones have maximum absorption values at wavelengths of about 420 nm (blue), 534 nm (bluish-green), resp. 564 nm (yellowish-green). Their sensitivity ranges overlap to provide vision throughout the visible spectrum. The maximum efficiency is 683 lm/W at a wavelength of 555 nm (green).<sup>[1]</sup>

The wavelengths for when a person is in photopic vary with the intensity of light. For the blue-green region (500 nm), 50% of the light reaches the image point of the retina.<sup>[2]</sup>

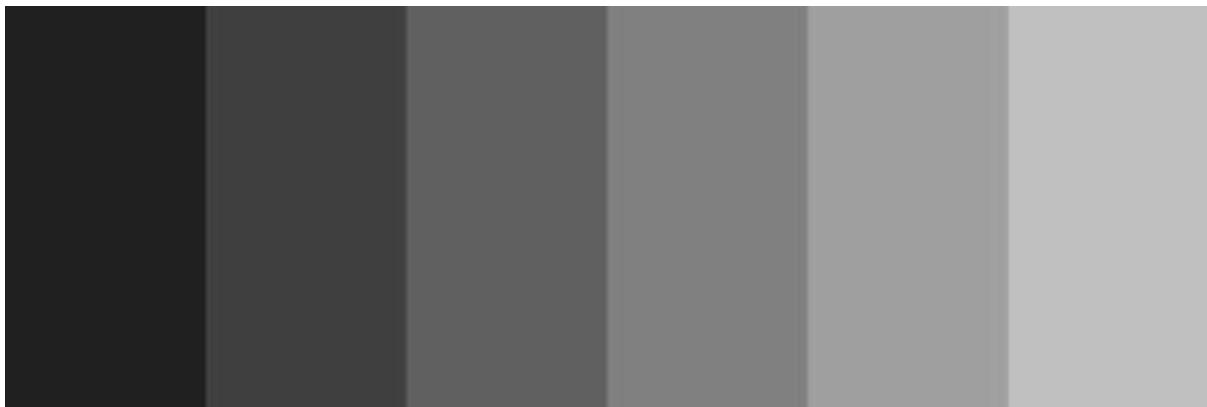
Adaptation is much faster under photopic vision; it can occur in 5 minutes for photopic vision but it can take 30 minutes to transition from photopic to scotopic.<sup>[2]</sup>

Most older adults lose photopic spatial contrast sensitivity. Adults in their 70s require about three times more contrast to detect high spatial frequencies than adults in their 20s.<sup>[3]</sup>

The human eye uses scotopic vision under low-light conditions (luminance level  $10^{-6}$  to  $10^{-3.5}$  cd/m<sup>2</sup>), and mesopic vision in intermediate conditions (luminance level  $10^{-3}$  to  $10^{0.5}$  cd/m<sup>2</sup>).

## Weber ratio

$$C = (L_{\text{fondo}} - L_{\text{Carattere}}) / (L_{\text{fondo}})$$



The electromagnetic spectrum can be expressed in terms of wavelength, frequency, or energy. Wavelength and frequency are related by the expression

$$l = c / \nu$$

### Spatial and Intensity Resolution

Intuitively, spatial resolution is a measure of the smallest discernible detail in an image. Quantitatively, spatial resolution can be stated in a number of ways, with line pairs per unit distance, and dots (pixels) per unit distance being among the most common measures

*Elaborazione Delle Immagini, 3 ottobre*

### Capitolo 3

$s = T(r)$ , dove  $r$  è il livello di grigio in input ( $f(x, y)$ ) e  $s$  è il risultato della trasformazione

Un angolo  $< 45$  comprime i livelli di grigio; un angolo  $> 45$  li stretcha, migliorando i dettagli

Non siamo interessati ai costi computazionali; usiamo una tabella contenente tutti i valori "precalcolati" (*lookup table*)

I range da 0 a  $L - 1$  può essere interpretato come **da 0 a 1**

**Un'immagine digitale è rappresentata da funzioni discrete, che devono avere valori discreti (non necessariamente interi)**

**Un'immagine è ben contrastata** quando assume tutti i valori del range dinamico

Dopo il valore massimo (limite di saturazione), il tono viene visualizzato come bianco

### Operatori puntuali lineari

**$f(n)$**  = funzione applica identicamente a ogni pixel

Due errori possibili:

- intensità non ammessa (non nei 256 valori)
- Oltre i limiti di saturazione o threshold

## Operatore di Thresholding

*Binarizza l'immagine*

$$g(n) = P * f(n) + L$$

Con  $P < 1$ , il risultato avrà un range dinamico ridotto

Con  $P > 1$ , il risultato avrà un range dinamico ingrandito

## Negativo

Se poniamo  $P = -1$ ,  $L = K - 1$  (livello massimo di intensità)

->

$g(n) = -f(n) + K - 1$  := funzione che inverte le intensità dei pixel di un'immagine

*Nota:*

*Operazioni che aumentano il contrasto amplificano anche il rumore originale*

Due immagini con istogrammi diversi possono essere uguali:

Il nostro sistema visivo compensa i diversi istogrammi, se le due immagini sono le stesse e hanno la stessa proporzione tra i livelli di grigio

Data un'immagine monocromatica con intensità tra 0 e 255, e voglio cambiare il range, usiamo le **proporzioni**

Per eliminare il “rumore impulsivo” spesso si tagliano le code (valori intorno a 0 e 255) ponendo

$$G_{min} = 1 - 5 \%$$

$$G_{max} = 95 - 99 \%$$

## Contrast stretching

Detti  $G_{max}$  e  $G_{min}$  i livelli di grigio minimo e massimo, viene usata una funzione lineare per espandere i valori nel range (0, 255)

$$g(a, b) = (f(a, b) - G_{min}) * (255 / G_{max} - G_{min})$$

## Operatori puntuali non lineari

### Scaling logaritmico

$$S = c * \log(1 + r)$$

Stratifica i toni scuri e comprime i toni chiari

Data un'immagine con molti toni (es. trasformata di Fourier, con  $10^6$  livelli di grigio), possiamo applicare un operatore logaritmico

- Se  $c = 1$ , a volte è necessario un secondo operatore che sterchi i valori per ottenere toni tra 0 e 255
- C può essere calcolata:  $c = (L - 1) / (\log R)$ ; ponendo quindi  $0 < s < c * \log(1 + R)$  e  $0 < r < R$

Uno scaling può migliorare l'enanchment, generando artefatti e ampliando il rumore

## Gamma correction

$$S = C r^\gamma$$

C e  $\gamma$  sono costanti positive

Se  $\gamma > 1$  -> espande i toni scuri e comprime i toni chiari := l'immagine si scurisce

Se  $\gamma < 1$  -> espande i toni chiari e comprime i toni scuri := l'immagine si schiarisce

Schiarendo un'immagine, i picchi di scuro si spostano verso il chiaro, e aumentano i picchi a 255 (saturazione)

*Nota: alcuni monitor usano gamma correction dell'input ricevuto prima di mostrarlo.*

*-> per visualizzare l'immagine originale, si può applicare gamma correction con  $\gamma = 1 / (\gamma \text{ del monitor})$*

## **Equalizzazione istogramma**

### **Istogramma cumulativo**

Per ogni livello, il numero di elementi del campione con valore  $\leq$  al livello corrente (accumula tutti gli individui)

-> probabilità di avere un livello di grigio  $\leq$  a K, con K rappresenta

Può essere visto come la **funzione** che mappa un certo livello di grigio alla sua probabilità

Nota: nell'equalizzare l'istogramma, è evidente che l'area sottesa all'istogramma (che si ottiene con integrale dell'istogramma) non cambia

Se noi applichiamo l'istogramma cumulativo come funzione di trasformazione, questo equalizza l'istogramma dell'immagine

Interpretando l'istogramma come funzione **discreta**

Allora l'istogramma cumulativo si ottiene dalla sommatoria delle frequenze dei livelli discreti di grigio

- Non essendo continuo, genero dei buchi nei valori dell'istogramma
- I valori possono solo crescere (stiamo applicando la funzione cumulativa, monotona non decrescente)

# Filtri, istogramma, smoothing 4 ottobre

Lavorare file salvati in jpeg **può** dare problemi: la compressione causa problemi quando si va a elaborare l'immagine

## Equalizzazione dell'istogramma locale

E' possibile definire la funzione di trasformazione della base di una "sotto-regione" dell'immagine

Nota: può anche essere di un'altra immagine

Sequenza di equalizzazioni su sotto-regioni dell'immagine

Possiamo procedere dividendo l'immagine in porzioni

1. Si valuta l'istogramma della prima porzione e si equalizza
2. Si trasla di una posizione, aggiungendo una colonna di pixel e togliendone una
3. Si continua da 1 -> non sarà necessario ricalcolare tutto l'istogramma

Es. Migliorare lo sfondo troppo scuro

1. Definisco la dimensione di una finestra di pixel
2. Per ogni finestra sull'immagine, calcolo la media e la varianza dell'intensità dei pixel
  - Posso classificare l'intorno del pixel centrale, per riconoscere se la finestra è nell sfondo o no
    - Uso la varianza per riconoscere le zone di bordo con il soggetto
      - Se sostituissi la media locale a ogni pixel, ottengo un'immagine con meno rumore, più omogenea ma meno definita
      - Se assegnassi la varianza locale a ogni pixel, ottengo ben definiti i contorni tra soggetto e sfondo

Se un algoritmo richiede molti parametri -> molto difficile e richiede molto lavoro

**Stimare i parametri tramite Machine Learning**

**Scegliere l'algoritmo migliore sulla base del risultato che si vuole ottenere**

## Slicing

Possiamo "tagliare" l'immagine in N (solitamente 8) "**bit-plane**"

Ogni bit-plane è la stessa immagine di partenza ma binaria (due soli livelli di intensità)

Lavoro solo su bit-plane, con operazioni binarie

Es. Maschere

Se prendo un'immagine, ne scelgo la parte da rimuovere e pongo a nero il resto

Posso quindi procedere con operazioni binarie per ottenere il risultato voluto

Somme e differenze tra i immagini

Possono produrre valori fuori scala (fuori da (-255, 255) )

-> Per ogni Pixel i :  $(i - \text{min}) / (\text{max} - \text{min})$

### Ridurre il rumore

Se prendo N foto in sequenza di uno stesso soggetto e le compongo, riduco il rumore e miglioro il contrasto  
Il rumore è infatti una sommatoria di valori casuali e sono distribuiti secondo una normale; la somma tende a 0 con varianza sempre minore

Mappa delle luci

Acquisire un telo bianco in un certo luogo ci permette di mappare le diverse intensità di luce  
Possiamo usare questa mappa come maschera per correggere le foto scattate nello stesso posto

*Elaborazione Delle Immagini, 6 ottobre*

Capitolo 4

## Restoration

Per rimuovere il rumore dobbiamo capire:

- se c'è
- Che tipo è
- Dove è

Diversi tipi di rumore:

- Gaussiano: l'immagine è più chiara
- Impulsivo: tipico istogramma con valore massimo e minimo a picco
- Uniforme
- Gamma
- Esponenziale
- Rayligh

Se l'istogramma tende a uniformarsi a causa del rumore, perdo la nitidezza del contorno

Un'alternativa, che prevede di agire sulle frequenze del segnale, è molto oneroso

### Rumore:

Se la varianza del rumore è alta, e la varianza del segnale è bassa, il rumore "sovrascrive" il segnale

**SNR** = scarto quadratico medio del segnale / sqmedio del rumore

ErroreMedio = varianza dell'intensità di un'immagine

Il rumore può essere generato dal dispositivo di imaging, ma anche dall'elaborazione (Es. Jpeg)

Il rumore si maschera con i dettagli (texture); l'occhio umano distingue il rumore su regioni omogenee

Il restauro tiene in considerazione il pixel e il suo neighbourhood

### Come prendere una zona di pixel?

Si usa una maschera: dato un pixel, questa griglia è posta con il centro in corrispondenza del pixel, mentre i restanti quadrati della griglia sono riempiti con i pixel vicini

Nota:

Filtri con maschere di forma diversa possono dare risultati diversi

## Correlazione vs

### Convoluzione:

La maschera data viene ruotata di  $180^\circ$  e applica all'immagine. I valori della maschera vengono moltiplicati localmente per il pixel dato e suoi vicini. I valori della matrice risultante vengono sommati, e il risultato è il valore del pixel dato filtrato

Accorgimenti:

- I valori della maschera ha sempre somma 1: in zone omogenee di un'immagine, il filtro non produce differenze
- Hanno sempre righe/colonne dispari

Maschere ed effetti:

- Valore 1 al centro, 0 nel resto: l'immagine non cambia
- Valore 1 al lato, 0 nel resto: l'immagine viene shiftata dal lato dell'uno
- Valori 1/3 in una maschera  $3 \times 3$  nella riga centrale: sfuma i bordi, rimuovendo il rumore

Sharpening:

- Applico una maschera con valore centrale 2
- Sottraggo un'immagine filtrata con maschera  $1/3 \ 1/3 \ 1/3$
- Il risultato è una maschera più nitida, rimuovendo il rumore ma anche i bordi (edge sfocati)

Posso quindi applicare un operatore che ristabilisca i bordi mantenendo le zone omogenee inalterate (filro  $-0.3, 1.7, -0.3$ )

## Smoothing

Maschere a croce: per disegni immagini tecniche: non è lo sfondo a interessare, ma le righe/ colonne

Maschere a "cerchio" (quadrata priva di spigoli): filtro per rumore gaussiano

**I filtri sono separabili:** posso applicare due vettori in alternativa a una matrice, risparmiando costi computazionali

### Nota:

Possiamo usare un istogramma come caratteristica di un certo tipo di immagine, magari opportunamente lavorata

### Perché maschere grandi generano aliasing:

Quando la maschera è piccola, ed è su zone omogenee, non cambia l'immagine.

Se la maschera è abbastanza grande da contenere linee nere e lo spazio bianco in centro, genera aliasing trasformando in grigio lo spazio bianco

## Filtro mediano

Data la maschera che isola i pixel localmente, ordino i pixel e prendo la mediana (valore centrale) dei valori

Effetti: sfoca meno di un filtro di media, ma più costoso

Anche la forma della maschera può variare (croce, gaussiana, etc... )

- Bene per impulsivo e gaussiano (se debole)
- Costoso

*Applicato iterativamente, produce effetti migliori, rimuovendo il rumore senza danneggiare gli edge*

**Quando fermarsi?**

Quando si nota che la soluzione converge senza migliorare ulteriormente

## Alpha trimmed

Rumore impulsivo e gaussiano: come restaurare l'immagine con un solo filtro

1. Data la maschera, prendo i pixel
2. Gli ordino
3. Scarto le code (0, 255), che contengono il rumore impulsivo
4. Faccio la media

## Altri filtri di rango

**Minimo:** data la maschera, applico al pixel il minimo del suo vicinato

- I dettagli chiari vengono cancellati, gli scuri si espandono

**Massimo:** data la maschera, applico al pixel il massimo del suo vicinato

Possono causare danni se il rumore presente è alto: espande il rumore

## Filtri con maschera rotante

Data una maschera di dimensione dispari, calcolo la varianza di 9 “sotto-finestre” della finestra data

- Scelgo quindi la sottofinestra con la varianza minore
- Procedo assegnando al pixel centrale la media della sottomaschera scelta

Conseguenze:

- bordi più definiti
- Rimozione del rumore
- Effetto cartoon
- Aliasing

# Edge e sharpening (10/10)

Ricorda:

Il sistema visivo umano tende a schiarire i bordi per distinguere i contorni degli oggetti

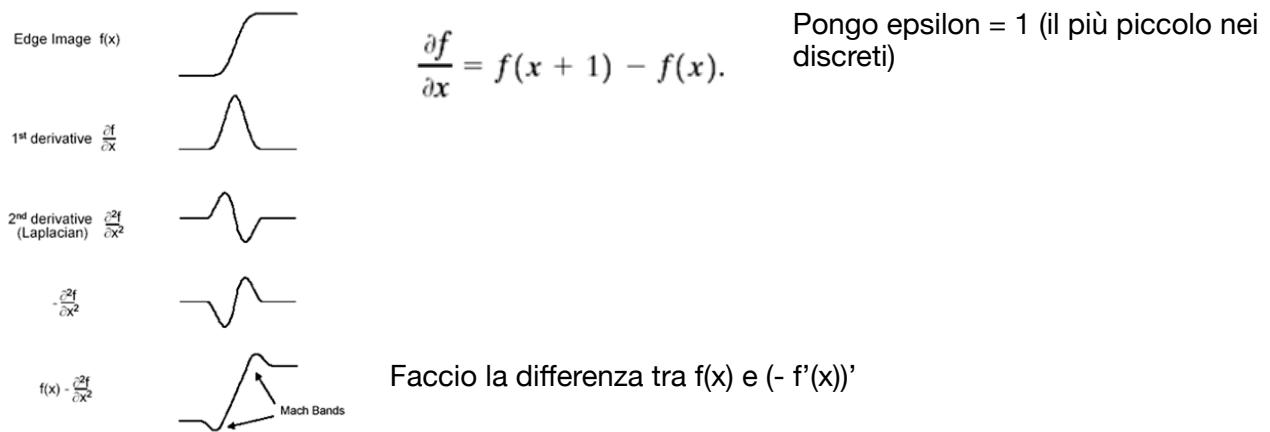


Aumenta il contrasto, ma anche il rumore

**Edge:** cambiamenti repentini dell'immagine.

**Edge detection:** cambiamenti repentini dell'immagine. Ma è necessario distinguere tra gli edge di oggetti fisici, e le ombre o altri cambiamenti dovuti alla luce

## Aumentare il contrasto sugli edge



## Forme delle derivate di un'immagine digitale

**Rampa:** cambio smooth di valori. La derivata seconda mette in evidenza solo il punto di inizio e di fine

**Spike** (punto isolato): picchi. La derivata seconda mette in evidenza il punto di picco, e i suoi edge

**Step:** cambio rapido. La derivata seconda mette in evidenza l'inizio e la fine adiacente del cambio

## Filtri Laplaciani

Molto sensibili al rumore

Mettono in evidenza inizio e fine della rampa (bande di Mach)

0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1
0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

**FIGURE 3.39**  
(a) Filter mask  
used to  
implement the  
digital Laplacian,  
as defined in  
Eq. 3.7-4.  
(b) Mask used to  
implement an  
extension of this  
equation that  
includes the  
diagonal  
neighbors. (c) and  
(d) Two other  
implementations  
of the Laplacian.

Nota: isotropic per angoli di 45 gradi

## Filtri derivativi con epsilon = 2

Considero il valore del pixel precedente e quello del successivo

### Gradiente: $D = (f'(x), f'(y))$

Un vettore, può essere codificato come direzione e distanza dal centro

Distanza dal centro:

$L_2 = \| D \| = \text{norma del vettore} = [Vx^2 + Vy^2]^{(1/2)}$

$L_1 = \text{distanza di Manhattan} = |Vx| + |Vy|$

$L_1$  è più veloce da computare, e approssima bene la distanza in spazi discreti

Uso filtri con i valori del gradiente, che possono essere positivi/negativi o solo positivi se applico il modulo

Riconoscimento automatico di difetti:

Prendo l'immagine, ne faccio l'edge detection, applico funzione di threshold e confronto l'immagine con una ottenuta con lo stesso procedimento di un'immagine di un oggetto sano

La distribuzione di edge mi permette di riconoscere la sua texture, quindi materiale (o assenza di materiale) etc.

## Filtro di sobel

Agisce sui bordi accentuandoli, ha un effetto di smoothing pesato al centro  
 $[+1 +2 +1 ; 0 0 0 -1 -2 -1]$  e la sua trasposta

## High boost

Cerca di trasformare  $f(a, b)$  in modo che rispecchi la nostra percezione degli edge

1.  $M = f \text{ conv Filtro Gaussiano}$

2.  $\text{Mask} = f - M$

3.  $\text{Result} = f + a * \text{Mask}$

Dove  $a$  è un parametro costante

## Perché i valori di una maschera devono dare 1?

Per mantenere regioni omogenee inalterate

# Trasformazione geometrica (10/10)

Spostare il pixel, riscrivendolo in una nuova posizione:  
“Raddrizzare l’immagine”

## Problemi:

Spostiamo i pixel tramite funzioni, ma c’è il rischio di ottenere valori **non** interi  
-> Buchi nell’immagine, valori sconnessi

## Soluzione: inverse mapping

Procedo al contrario, partendo dall’immagine risultato e cerco nell’originale

## Zoom

Scalare un’immagine

- possiamo prendere un pixel (x,y) e sostituirlo nell’immagine finale con 4 pixel identici
  - Immagine “a blocchi” se lo zoom è elevato

## Nearest neighbor:

Approssimo il valore di un pixel partendo dai pixel vicini

## Interpolazione dei valori

Il valore assegnato è una media pesata del valore dei 4 pixel vicini: peso di più i pixel più vicini

(Nota: nel libro è spiegata più “teorica”)

Nel caso a 3 dimensioni ( pixel ) :

Possiamo interpolare più volte su una funzione bidimensionale tratta dalla originale, fino a ottenere il valore approssimato del punto

## Campionamento:

Se il campionamento è più grande della più piccola delle strutture che stiamo campionando, si forma **aliasing** (i dettagli si fondano)

Applicare un filtro **prima** di effettuare il campionamento

-> l’immagine sottocampionata e poi zoommata è sfocata, ma non “blocchettizata”

In caso dei video:

Posso usare più frame per recuperare dettagli (interpolando i valori)

## Alcune trasformazioni geometriche

Translate: sposta i pixel

Scale: scala l'immagine

Come procedere:

- Si trasla l'oggetto nell'origine
- Si scala l'oggetto nell'origine
- Si trasla l'oggetto con l'angolo sotto a sinistra nel punto di partenza

Rotazione, traslazione, proiezione sono **trasformazioni lineari  $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$**

### Rotazione:

$$T(a, b) = [\cos(\theta), \sin(\theta); -\sin(\theta), \cos(\theta)] * [a \ b]$$

**Proiezioni su una Linea L = {c v : c elements of R, v unit vector of  $\mathbb{R}^2$ }**

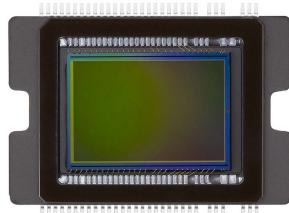
$$T(a, b) = ([a \ b] * v) * v$$

Più in generale

$$T(a, b) = (([a \ b] * v) * v) / \|v\|^2$$

# Il colore

Come si acquisisce un'immagine a colori?



La luce in ingresso attraversa un array di filtri rossi / verdi / blu, sotto i quali molti microsensori acquisiscono il valore

Sopra i filtri delle microlenti *incanalano* la luce attraverso i filtri, evitando che i fotoni *vaghino* nel sensore (producendo rumore)

Più piccolo è il sensore, più alto è il rumore

## Distribuzione dei filtri:

- 2 verdi, 1 rosso, 1 blu: **Bayer CFA**
- Il verde è la banda centrata, che corrisponde alla luminanza
  - > dà luce all'immagine

I "buchi" di ciascun immagine in scala di grigi (blu, rossa e verde) vengono interpolati

## Tecniche:

- media dei vicini
- Mediana
- ... Tecniche avanzate per evitare artefatti sui bordi

## Sensore foveon:

Tre strati di sensori, ciascuno fa passare i colori per i quali non è predisposto -> tre strati RGB permettono di ottenere un'immagine senza artefatti

Alcune operazioni di processing di una camera digitale:

- Demonic
- White balance
- Trasformazioni di spazio colore
- Elaborazione dei colori (hue, saturazione)
- Trasformazione curva di esposizione

## Problemi:

una fotocamera digitale è fatta per *foto belle*:

- non adatte per applicazioni industriali/mediche, perché non fedeli

**RGB** non è assoluto, ma relativo

Lo stesso soggetto acquisito in due punti diversi può apparire con *colori* differenti

-> l'indipendenza dalla luce, dall'esposizione etc. Può essere ottenuta tramite post-processing

- Dimensioni fortemente correlati: informazione ripetuta tra i 3 diversi colori
- Non uniformi percettivamente: la stessa varianza nello spazio rgb può generare colori molto o poco diversi
- Dipendenti dal dispositivo
- Non rappresenta la luminanza, che è *prodotta* dal verde

La fonte luminosa influenza i colori, **ma**

**Nota:** non notiamo la dominante dovuta alle diverse fonte luminose, perché il sistema visivo le neutralizza

## Come?

L'occhio si accorge della dominante e cambia la sensibilità ai diversi colori

# Algoritmi per bilanciare i colori

## White balance

Come noto il tipo di fonte luminosa?

Vado a controllare un oggetto che so essere bianco, e shift i colori

**Calcolo lo scaling dei colori come rapporto tra il valore della dominante sul bianco / 255**

Alternativa: ipotizzo ch elà media dei colori di un'immagine debba essere il **grigio**

## Grey world

1. **Calcolo la media per ciascun canale R G B**
2. Calcolo la media dell'immagine a livelli di grigio
3. Scalo ciascun canale RGB al fine che abbia la media = media dell'immagine a livelli di grigio

-> funziona se l'immagine ha molti colori, tali che la loro media possa essere effettivamente grigio  
Per evitare che una classe domini sulle altre, rendendo il Grey world inutilizzabili, posso usare il gradiente

### In Matlab:

```
grayImage = rgb2gray(rgbImage); % Convert to gray so we can get the mean luminance.  
% Extract the individual red, green, and blue color channels.  
redChannel = rgblImage(:, :, 1);  
greenChannel = rgblImage(:, :, 2);  
blueChannel = rgblImage(:, :, 3);  
meanR = mean2(redChannel);  
meanG = mean2(greenChannel);  
meanB = mean2(blueChannel);  
meanGray = mean2(grayImage);  
% Make all channels have the same mean  
redChannel = double(redChannel) * meanGray / meanR;  
greenChannel = (double(greenChannel) * meanGray / meanG);  
blueChannel = (double(blueChannel) * meanGray / meanB);  
% Recombine separate color channels into a single, true color RGB image.  
rgbImage = cat(3, redChannel, greenChannel, blueChannel);
```

## White point estimation: max RGB

Scalo rispetto al valore massimo sul canale

-> rumore impulsivo danneggia tutta l'immagine, posso tagliare le code (ad esempio i valori nel range 0 - 0.05 % e 99.95 - 100 %)

### In python

```
# white balance for every channel independently  
def wb(channel, perc = 0.05):  
    mi, ma = (np.percentile(channel, perc), np.percentile(channel, 100.0-perc))  
    channel = np.uint8(np.clip((channel-mi)*255.0/(ma-mi), 0, 255))  
    return channel  
  
imCol = np.dstack([wb(channel, 0.05) for channel in (red, green, blue)] )
```

# Modelli del colore

Spazi colore alternativi al RGB

## Sintesi additiva:

Un monitor ciascun pixel può essere rosso verde o blu

Sintesi additiva spaziale: i pixel sono così vicini che non siamo in grado di discriminare

## Sintesi sottrattiva:

I colori sono usati come filtri che sovrapposti ottengono un colore (dal bianco verso il nero)

La stampante pone colore sul foglio, parzialmente sovrapposti, ottenendo un risultato per sintesi mixata

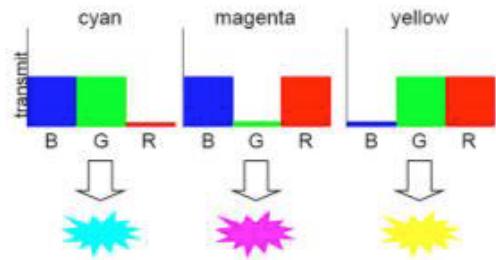
## Spazi approssimati per la stampa:

### CMY:

Ciano - magenta - giallo

### CMYK: Come funziona una stampante

Posso usare il nero come "base", ponendo nero =  $\min(C, M, Y)$  e poi aggiungendo i valori (C - nero, M - nero, Y - nero)



Un modello alternativo molto più vicino al nostro modo di "vedere" i colori sono i modelli HSL (Hue, Saturation, Lightness), HSI (Hue, Saturation and Intensity), o HSV (Hue, Saturation and value).

## HSI (tinta, saturazione, luminosità)

Tinta vista come grandezza ciclica: i colori scorrono in senso orario (coordinata radiale)

Trasformo il cubo RGB in un doppio cono, con l'asse dei grigi visto dall'alto e la saturazione a rotazione

## HSV:

Schiaccia il doppio cono in un unico cono, ottenendo un **value** (intensità) molto più approssimativa

Più semplice da calcolare

## sRGB:

Ottenuto dalla media di molti monitor, e stabilito come spazio colore standard da cui derivano hsv e hai. Legato matematicamente allo spazio colorimetrico standard CIE

Fascia consumer: ha molti problemi, ed è principalmente adottato da Device lato consumer

Color management system: software che definiscono la trasformazione da un dispositivo allo spazio CIE

## YCbCr

La trasformazione dallo spazio colore RGB a quello YCbCr è una trasformazione lineare. Se l'immagine di input ha range di valori tra 0 e 255, la trasformazione è la seguente:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (\text{EQ})$$

Il primo termine della sommatoria (offset) serve per portare i valori di output in un range positivo. Se l'immagine di input ha valori tra 0 e 1, l'offset è moltiplicato per 1/255.

## Enhancement delle immagini a colore

Posso procedere applicando trasformazioni anche a una solo delle bande. Usando gli spazi percettivi (es. HSI) possiamo intervenire solo su un asse per migliorare l'immagine.

### Riconoscere le bande:

saturazione: in RGB va dal bianco al nero lungo la scala di grigi

Tinta: in RGB appare come threshold

Intensità/brightness: scala di ciascun colore, appare simile in RGB

### Sharpening:

Si può agire sul RGB o sulla banda di intensità in HSI, ottengono circa lo stesso risultato

### Rumore:

Appare più visibile sulla tinta e meno sull'intensità



a b c

**FIGURE 6.49** HSI components of the noisy color image in Fig. 6.48(d). (a) Hue. (b) Saturation. (c) Intensity.

La media dei rumori tende ad annullare il rumore:

- Il rumore random sui tre canali sommati si annulla, da qui spiegato perché l'intensità appare "bella"
- I cambiamenti sull'asse della tinta sono invece forti

### Smoothing:

Applicato a ciascun canale RGB produce un'immagine sfocata nel complesso

### Filtro mediano:

Ottimo per il rumore impulsivo. E' necessario ordinare i **vettori RGB** secondo la loro norma per poi individuare il vettore rgb centrale ed assegnarlo nell'immagine risultato

### Edge detection

Può essere calcolato il gradiente di ciascuna banda e poi sommata, ma non è perfettamente corretto. Una versione più complessa è presente su libro

# Segmentazione

## Blue screen

Segmentazione (trova i contorni) viene eseguita

### Segmentare:

Prendo una regione caratterizzata da un colore

Ne faccio la media colore

Per ogni pixel con valore = media -> estraggo valori che cadono in una regione centrata nel pixel

### Classificatore di skin tone generale:

Scelta dello spazio colore: nello spazio cromatico, due colpi diverse di pelle sono quasi uguali salvo per la componente di brightness.

Posso **non** usare la banda di intensità e filtrare solo per cromatura, riconoscendo **tutti i tipi di pelle (e quindi visi etc .. )**

## Importante: cosa si impara?

- **Si può usare spazio colore non nativo del dispositivo**
- **Si può rimuovere le ombre bilanciando i bianchi tramite color balancing** (white patch individuando il bianco massimo; Grey world, la media dei colori è grigio)

---

## Introduzione

Segmentazione: dopo aver ottenuto un'immagine migliorata (tramite enhancement), eseguo il gradiente e isolo gli oggetti dallo sfondo (immagine binaria)

### Passaggio fondamentale: etichettamento delle componenti connesse (labelling)

Associa i pixel vicini assegnando a ciascuna regione connessa (e a ciascuno dei suoi pixel) una label

### Descrittore: una proprietà della regione, che possiamo usare per distinguere il tipo di regione

- Texture: la trama dell'oggetto può essere utile a definirne un descrittore

### Processi

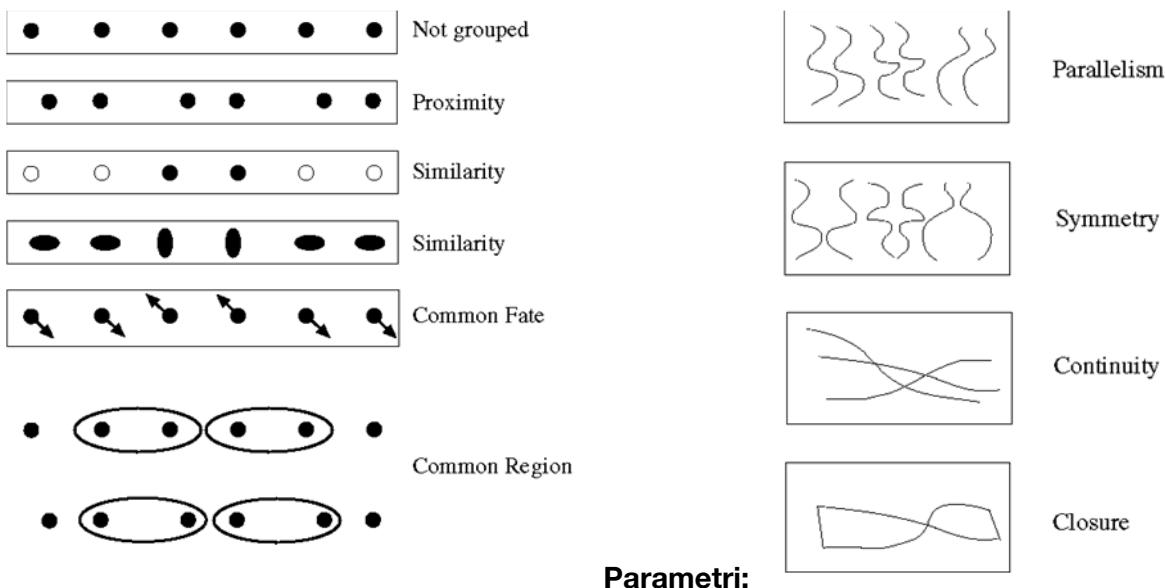
- Riconoscimento: problema a due classi, data un'immagine isola gli oggetti di interesse
- Classificazione: data un'immagine, separa gli oggetti secondo modelli e li classifico **tutti**
- Clustering: gli oggetti sono separati secondo le loro caratteristiche ricorrenti, senza modelli noti

Trovare le regioni omogenee (oggetti unici) per segmentarli è ancora un problema aperto  
-> E' necessario segmentare solo ciò che realmente serve, per semplificare il problema

### Classe unknown: classe estranea, *di rigetto*

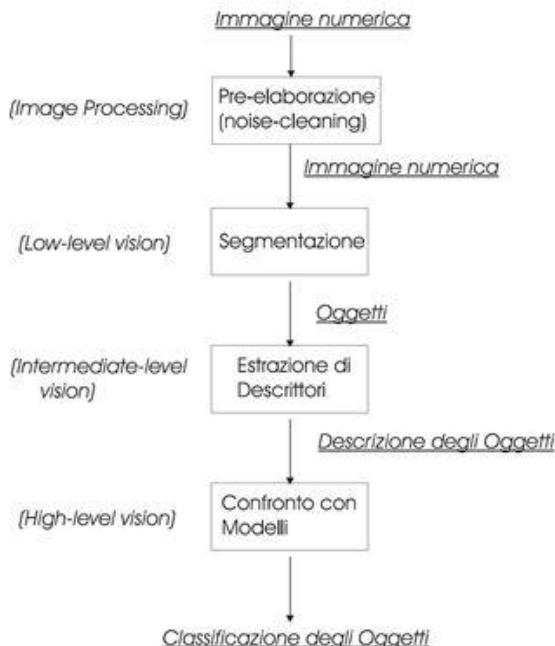
Per riconoscere le feature possiamo usare delle trasformazioni che rendono l'immagine più semplice da analizzare per l'algoritmo

## Leggi di percezione



- Prossimità
- Similarità
- Regioni comuni
- Parallelismo / simmetria

## Segmentazione region-based



### Problemi:

- Sfondo non omogeneo
- Oggetti sovrapposti

### Riconoscere gli oggetti:

- possiamo usare un *modello* da sovrapporre sull'immagine alla ricerca di oggetti simili
  - **Ma**, molto costoso, richiede di testare il modello su tutta l'immagine in tutte le rotazioni
- Posso prendere il modello ed estrarne dei descrittori (spigoli per gli oggetti rigidi, skin tone per il volto, ... )

*Come possiamo dire: tutti i pixel appartengono allo stesso oggetto?*

Se una texture rende difficile il riconoscimento dei bordi, uso un algoritmo di segmentazione **per regioni**

- Edge detection: gradiente più o meno sofisticato per riconoscer i bordi. Problema: ombre o rumore potrebbero generare bordi "aperti"

- Edge linking: *unire* le linee

Trasformata di Diaff: cerca di generare le linee mancanti

### **La segmentazione produce un insieme di regioni binarizzate**

Data un'immagine binaria, cerco gli insiemi di pixel **connessi** tra loro (adiacenti e neri)

Centroide: media di (x, y) di pixel neri connessi

Due pixel sono connessi se esiste una relazione di adiacenza spaziale, e il livelli di grigio rispettano uno specifico criterio di similarità

- 4-connettività: N4(centro) (pixel disposti a croce senza il pixel centrale)
- 8-connettività: N8(centro)

La connettività è **transitiva**

Scansionando l'immagine associo a ciascun pixel nero una label, rispettando la maschera di labeling e la proprietà transitiva

- potrebbero ancora esserci label adiacenti ma considerate "diverse"  
-> uso una tabella di equivalenza che associa le label adiacenti (proprietà transitiva) e *aggiusto il labeling*

**Nota:** possiamo per convezione non valutare le regioni sul bordo

### **Come distinguere le forme?**

- Bounding box
- Baricentro
- Area

Distanza tra pixel:

- Euclidea
- La somma euclidea può essere approssimata come  $D4 = |x - s| + |y - t|$  (distanza di (x, y) da (s, t))
- $D8 = \max(|x - s|, |y - t|)$ , massimo della differenza

# Segmentazione a soglia

Per ora trattiamo immagini con due sole soglie: chiaro / scuro, associati a oggetti / sfondo

E' possibile segmentare mediante una **soglia T**

- il risultato dipende da illuminazione, proprietà degli oggetti, riflettanza, dimensione...

In caso di un solo picco nell'istogramma, posso scegliere una tessitura per sogliare

Rimuovere uno sfondo a bande

## Soglia automatica

- migliore della manuale
- Ottenuta matematicamente
- Complessa, deve gestire le regioni
- Basata sui pixel di intorno e la posizione dell'immagine

$$T = T[x, y, p(x, y), f(x, y)]$$

## Metodi:

- Soglia centrata nell'istogramma: rapido, poco costoso, approssimativo
- Segmentazione per rapporto oggetto / sfondo: conoscendo il dominio, posso stabilire il rapporto (e quindi nell'istogramma) tra sfondo e oggetto/testo

## Raffinamento iterativo del valore di soglia

- 1) Si seleziona una soglia  $T$  (eg. valore medio).
- 2) Si segmenta l'immagine in base a  $T$ .
- 3) Si calcola il valore medio delle due classi/regioni ( $m_1, m_2$ )
- 4) Si seleziona una nuova soglia  $T = (m_1 + m_2)/2$
- 5) Si ripetono i passi 2, 3, 4 fino a quando i valori medi si stabilizzano.

## Isteresi

Accrescere le regioni ponendo una classe di *dubbio* tra due soglie

**Nota:** Se la varianza di colore di una regione è alta, allora la regione è segmentata male

- posso sfruttare questo per valutare il valore  $T$  di soglia che minimizza la somma delle varianze delle regioni ("varianza intragruppo")

## Otsu

Usa la varianza *infragruppo* (tra classi). Computato sull'istogramma (1D) dell'immagine

Divide l'istogramma in due classi:  $C1 = 0 \dots k$  e  $C2 = [k+1 \dots L-1]$

Cerca il valore  $k$  che massimizza il valore  $= (P(k) - m(k))^2 / P(k) * (1 - P(k-1))$

Dove  $P(k)$  = Probabilità di un pixel di appartenere a  $C1$  = IstogrammaCum(k) / (L - 1)

$m(k)$  = valore medio della classe  $C1$  = sommatoria per  $i = 0 \dots k$  ( $i * p(i)$ )

## Algoritmo di otsu - cambi di luce

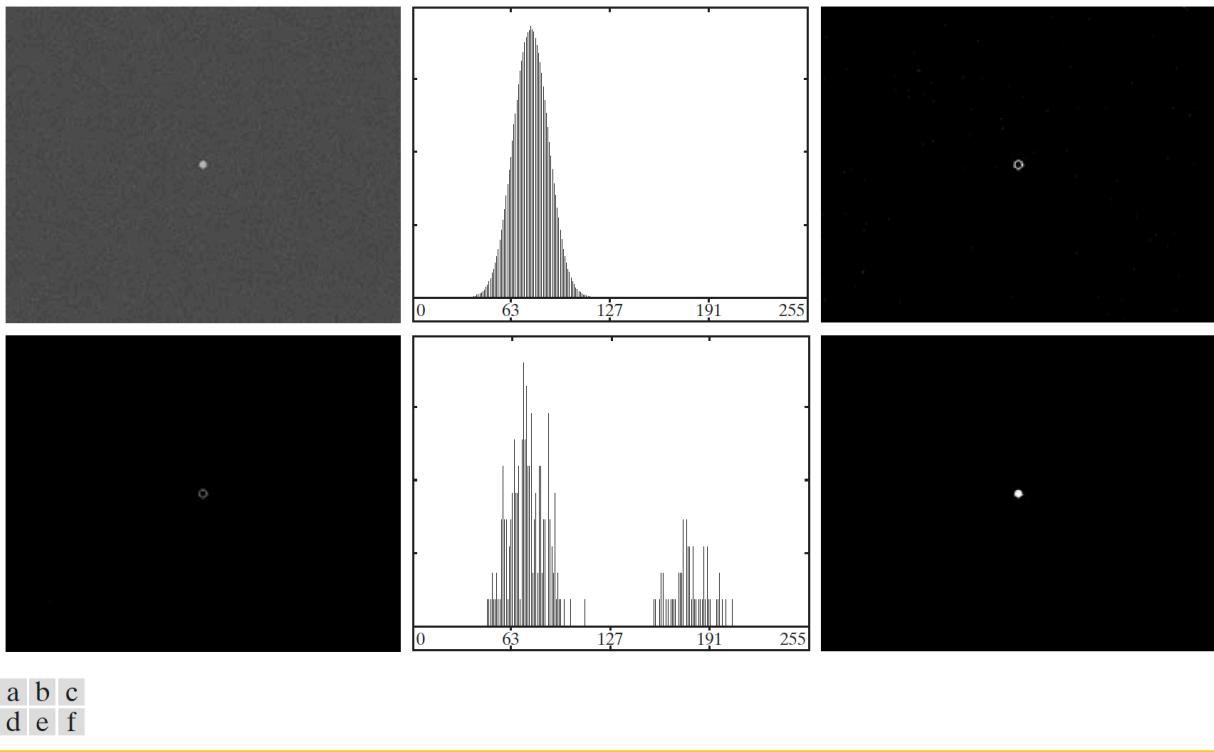
Serie di equazioni per migliorare la compensazioni in luce variabili

- Dividere un'immagine in quadranti
- Segmentare per quadranti tramite otsu (localmente)

Filtrare prima della segmentazione rimuove il rumore e i bordi. Siamo comunque in grado di soglia per una  $T$  appropriata

**Anche qui lo smoothing può fare la differenza, perché tende a dividere l'istogramma dell'immagine in due sezioni (classi) distinte**

Si può calcolare una maschera di edge, impostare di un threshold molto alto sugli edge (circa 97/98 percentili), prendere l'istogramma del valore dei soli i punti di edge per calcolare Otsu più accuratamente



**FIGURE 10.42** (a) Noisy image from Fig. 10.41(a) and (b) its histogram. (c) Gradient magnitude image thresholded at the 99.7 percentile. (d) Image formed as the product of (a) and (c). (e) Histogram of the nonzero pixels in the image in (d). (f) Result of segmenting image (a) with the Otsu threshold based on the histogram in (e). The threshold was 134, which is approximately midway between the peaks in this histogram.

## Compensazione dello sfondo - illuminazione

**Una possibilità:** stimare la luce presente per poi compensarla

**Oppure:**

Divido l'immagini in quadranti, e uso un algoritmo di soglia (es. otsu) per sfogliare ciascun quadrante indipendentemente

Una possibile soluzione al problema consiste nell'assegnare il valore di soglia determinato per il blocco esclusivamente al punto centrale del blocco, e ricavare poi la soglia per tutti gli altri pixel dell'immagine mediante un procedimento di interpolazione.

**Oppure**

Posso creare una funzione di compensazione per correggere lo sfondo eterogeneo

- traccio una retta da un punto all'altro dell'immagine, la variazione rappresenta i valori da correggere

Togliendo il gradiente (o considerando solo le parti di immagini con gradiente alto) ottengo un'immagine ridotta, con un istogramma tendenzialmente bimodale e sogliabile

### Esempio: algoritmo per segmentare le scritte bianche dallo sfondo

Uso un filtro centrato nel colore bianco e con una certa varianza, e soglio sulla base di questo filtro

Classifico le componenti connesse

Rimuovo regioni troppo grandi

## Soglia per regioni

Valutare un valore di threshold per T per ogni punto (x, y) dell'immagine. Due possibili formule

$$T_{xy} = aS_{xy} + bM_{xy}$$

$$T_{xy} = aS_{xy} + bM_g$$

Dove  $S_{xy}$  e  $M_{xy}$  sono rispettivamente la s.d e la media di un intorno del punto (x, y); mentre  $M_g$  è la media globale

L'approccio può essere esteso a una segmentazione tramite Predicati Logici:

Esempio:

$$Q(S_{xy}, M_{xy}) = \text{true if } f(x, y) > a * S_{xy} \text{ and } f(x, y) > b * M_{xy}$$

Per ogni x, y, si valutano i valori  $S_{xy}$  e  $M_{xy}$  dell'intorno e si pone il predicato logico

**Nota:** si possono risparmiare computazione tenendo conto dell'overlapping degli intorni di pixel adiacenti

## Segmentazione per regioni

- Region growing
- Split & merge

Problemi di merge:

Unire regioni tra loro simili, senza transitività. La segmentazione non può dipendere dall'ordine

### Grafo delle adiacenze

Creo un grafo delle regioni da unire. Associo a ogni nodo le caratteristiche delle regioni, legate da archi che rappresentano l'adiacenza.

Lego due regioni alla volta, e si aggiorna di volta in volta il grafo. Pongo un metodo di ordine nell'unione delle regioni (unisco sempre le due più simili)

Posso anche stabilire delle regole di unione:

- si fondano su pareri di esperti del dominio, non su dataset

### Clustering:

Partizioniamo un insieme di oggetti

(Classificazione: Le classi emergono dai dati)

# Segmentazione immagini a colori

Una possibilità:

Considerare RGB come tre immagini distinte in scale di grigio

**Colori omogenee -> oggetti omogenei**

**Trova picchi e valli sull'istogramma:**

1. Filtro l'istogramma
  2. Devo saper distinguere rumore da informazione
  3. La derivata prima mi permette di studiare i punti di picco; i segni della derivata mi permettono di capire se è un picco o una valle
- 1. Zero crossing**

Filtraggio migliore:

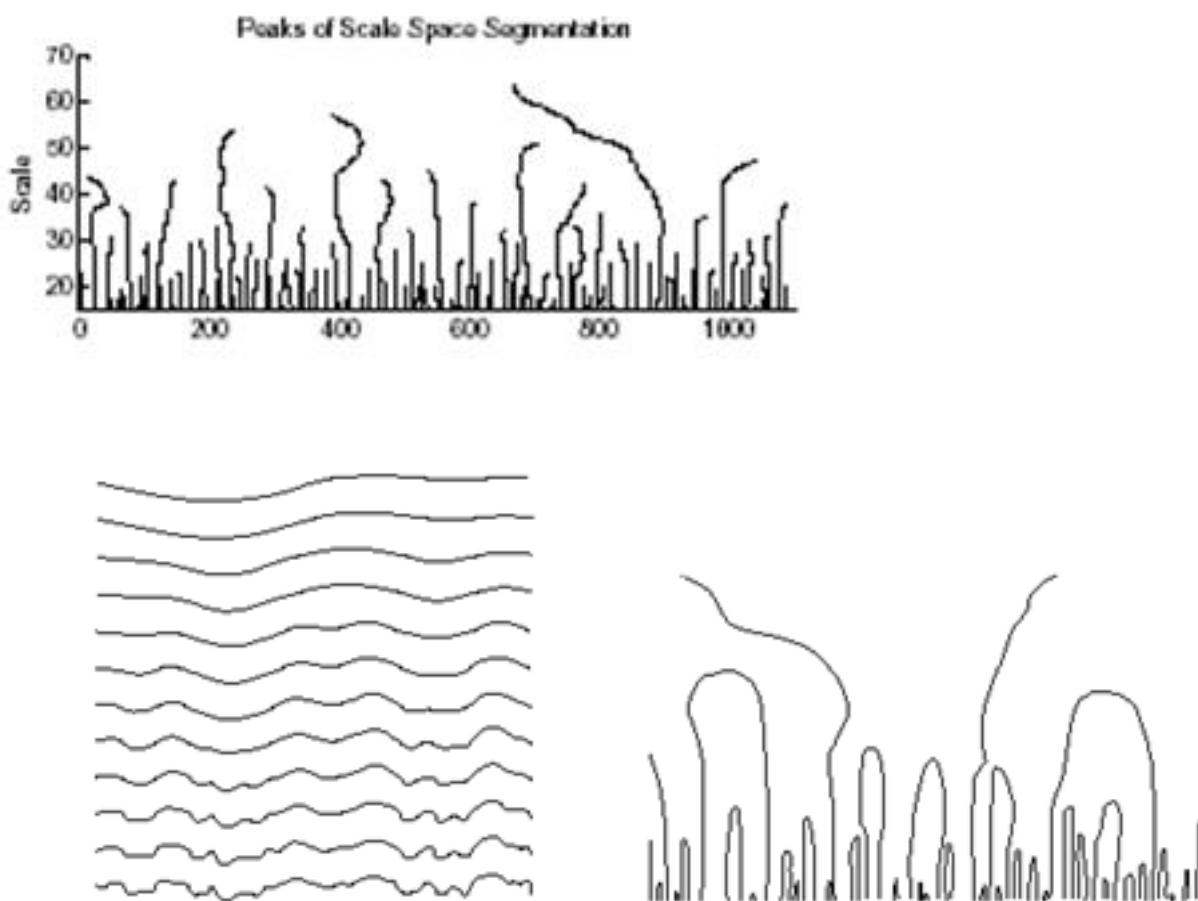
Uso una gaussiana con una data varianza

**Nota:** all'aumentare della varianza il segnale si leviga sempre più

**Scale space image:**

Ottenuto dalla sovrapposizione di istogrammi smussati con gaussiane

**Noto che al variare della varianza la posizione delle valli si spostano**



**Taglio il grafico e ne prendo una fetta**

**Prendo i picchi della fetta**

**Li localizzo precisamente alla base ripercorrendo all'indietro il segmento**

### **Spazi colore cilindrici:**

La tinta (hue), essendo ciclica, bisogna analizzare congiuntamente due cicli completi (per l'istogramma a cavallo tra  $0^\circ$  e  $359^\circ$ )

### **Ombre:**

L'ombra cambia il livello di grigio (intensità / lightness / value) lasciando invariate tinta e saturazione

-> posso scartare l'intensità e usare le altre grandezze

### **Modelli del colore**

Caratteristiche in base alle quali scegliere il modello colore:

1. Indipendenza dal dispositivo
2. Uniformità percettiva
3. Linearità vs .Non-linearità := *costi computazionali costosi*
4. *Presenza di singolarità* := se  $\text{saturazione} = 0 \rightarrow$  non abbiamo informazione circa la tinta
5. Costo della trasformazione
6. Intuitività delle coordinate := *più facile lavorare in spazi cilindrici*
7. Robustezza ai cambiamenti nelle condizioni di imaging

# Segmentazione nello spazio delle caratteristiche

Una componente fa da maschera alle componenti successive

- posso aggiungere altre *dimensioni* per trovare una dimensione in cui dei cluster si differenziano, per segmentare gli oggetti di un'immagine

## Clustering

Somiglianza tra i gruppi; i gruppi emergono dai **dati**

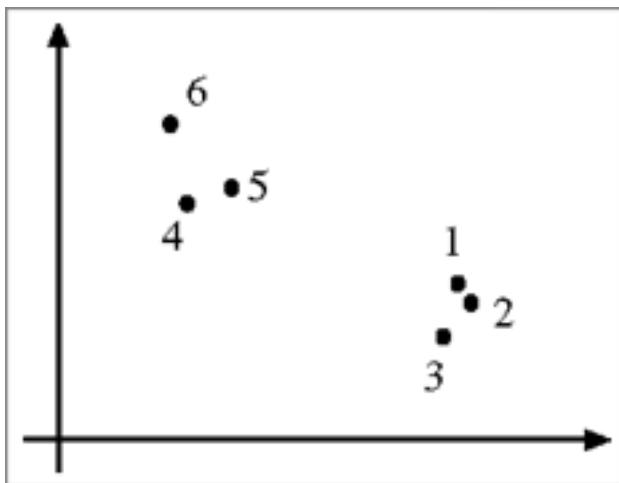
Problema **np completo**

-> usiamo algoritmi che restituiscono clustering subottimali

## Clustering gerarchico

Creo un albero unendo i due cluster più vicini finché non ottengo un unico cluster

I metodi gerarchici permettono di applicare criteri di somiglianza “ad-hoc” per controllare l’aggregazione e la divisione. Sono quindi da preferirsi quando non si dispone di uno spazio metrico delle feature



## K-means

Form K-means clusters from a set of  $n$ -dimensional vectors.

1. Set  $ic$  (iteration count) to 1.
2. Choose randomly a set of  $K$  means  $m_1(1), m_2(1), \dots, m_K(1)$ .
3. For each vector  $x_i$  compute  $D(x_i, m_k(ic))$  for each  $k = 1, \dots, K$  and assign  $x_i$  to the cluster  $C_j$  with the nearest mean.
4. Increment  $ic$  by 1 and update the means to get a new set  $m_1(ic), m_2(ic), \dots, m_K(ic)$ .
5. Repeat steps 3 and 4 until  $C_k(ic) = C_k(ic + 1)$  for all  $k$ .

**Algorithm 10.1** K-Means Clustering.

## **Posso combinarlo con ragion margining in caso di risultati non ottimali**

### **Isodata**

Parto da un seed, lo faccio crescere e assegno i pixel. Se un cluster è grande lo splitto, se due cluster tendono ad avvicinarsi li unisco. Si trovano i centroidi (i pixel medi più significativi) e si assegnando al centrale i pixel vicini

### **Algoritmo**

**Data** la partizione iniziale, per ogni punto P (rappresentato dal suo vettore delle feature) viene determinato il centroide più vicino

**Se**

la distanza è minore di una certa soglia T1, il punto P viene assegnato al cluster e la media del cluster viene aggiornata,

**altrimenti**

P diventa il centroide di un nuovo cluster

Se la distanza minima fra i centroidi di due cluster è minore di una data T2 soglia i cluster vengono fusi (si ridetermina la posizione del nuovo centroide).

**Ricorda:**

Il clustering restituisce gli oggetti colorati in modo omogeneo (**no texture**)

**CONTA SOLO IL DOMINIO / OBIETTIVO : LA SEGMENTAZIONE DEVE LAVORARE BENE PER RAGGIUNGERE L'OBIETTIVO**

**Possibili errori nella segmentazione:**

- Sotto-segmentazione
- Sovra-segmentazione
- Cattiva localizzazione dei contorni

**Valutazione di una segmentazione**

*Le regioni estratte devono essere:*

- uniformi ed omogenee.
- senza buchi interni.
- differenti dalle regioni adiacenti (secondo il criterio di uniformità).
- con confini semplici e non seghettati
- E soprattutto: mi permette di raggiungere l'obiettivo?**

# Texture

*Struttura composta da un elevato numero di elementi più o meno simili, nessuno dei quali si presenta singolarmente percepibile*

**Problema:** data una regione, riconoscere la texture che si ripete nella regione: dipende dalla dimensione della regione

**Aumentando** le dimensioni della finestra si ha quindi una maggiore stazionarietà. Ma anche una **minore** risoluzione dei confini fra regioni adiacenti di texture diversa ed un **maggiore** tempo di calcolo.

Categorie di operazioni su texture:

- riconoscimento di una texture
- Classificazione della texture
- Segmentazione di texture

Texture:

- Descrittore (vettore di valori o scalare)
- Funzione di distanza

**Approcci:**

- Statistico
- Strutturale

Analisi morfologica tesa a determinare la tipologia delle primitive (cerchi, quadrati, ecc.) dette **texel**

I tele possono essere primitive o immagini complesse ripetute

La texture si descrive secondo:

- **granularità**: dimensione delle aree caratterizzate da variazioni di intensità luminosa
- **direzione**: presenza di una direzione preferenziale lungo cui si dispongono le variazioni di intensità luminosa
- **ripetitività**: presenza di configurazioni ricorrenti
- **contrasto**: visibilità della texture rispetto all'immagine

Distinguere due texture:

1. Creo l'istogramma della texture
2. Posso usare la varianza e normalizzarla

$R = 1 - 1 / ( \sigma^2(z) )$  := descrittore che indica la varianza di varianza con un valore tra 0 e 1

Nota: posso impiegare più feature diverse per classificare e distinguere le texture

Valuto le caratteristiche in base ma quanto separano tra texture

Combino le caratteristiche tra loro indipendenti

Posso ottenere la texture **dopo** aver analizzato l'immagine: cerco sempre di semplificare il processo decisionale prima di prendere una decisione

Posso classificare le texture usando l'istogramma come descrittore

## Riquantizzazione

- Clustering
- Ridurre i bit per l'immagine

Moment	Expression	Measure of Texture
Mean	$m = \sum_{i=0}^{L-1} z_i p(z_i)$	A measure of average intensity.
Standard deviation	$\sigma = \sqrt{\mu_2} = \sqrt{\sigma^2}$	A measure of average contrast.
Smoothness	$R = 1 - 1/(1 + \sigma^2)$	Measures the relative smoothness of the intensity in a region. $R$ is 0 for a region of constant intensity and approaches 1 for regions with large excursions in the values of its intensity levels. In practice, the variance, $\sigma^2$ , used in this measure is normalized to the range [0, 1] by dividing it by $(L - 1)^2$ .
Third moment	$\mu_3 = \sum_{i=0}^{L-1} (z_i - m)^3 p(z_i)$	Measures the skewness of a histogram. This measure is 0 for symmetric histograms; positive by histograms skewed to the right about the mean; and negative for histograms skewed to the left. Values of this measure are brought into a range of values comparable to the other five measures by dividing $\mu_3$ by $(L - 1)^2$ , the same divisor we used to normalize the variance.
Uniformity	$U = \sum_{i=0}^{L-1} p^2(z_i)$	Measures uniformity. This measure is maximum when all intensity values are equal (maximally uniform) and decreases from there.
Entropy	$e = -\sum_{i=0}^{L-1} p(z_i) \log_2 p(z_i)$	A measure of randomness.

### Third moment

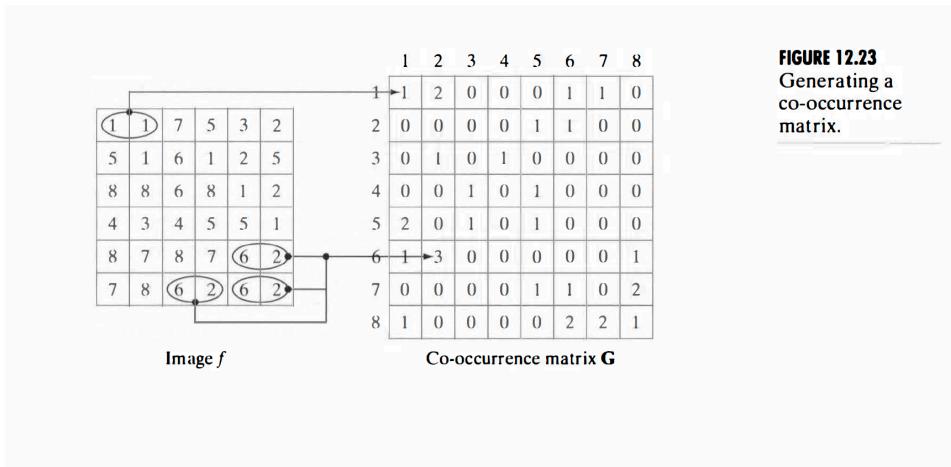
Measures the skewness of a histogram.

This measure is 0 for symmetric histograms; positive by histograms skewed to the right about the mean; and negative for histograms skewed to the left. Values of this measure are brought into a range of values comparable to the other five measures by dividing  $\mu_3$  by  $(L - 1)^2$ , the same divisor we used to normalize the variance.

## Matrice di co-occorenza

Alcuni descrittori per la matrice di co-occorenza:

Un'immagine regolare crea una matrice di co-occorenza *regolare*. Scegliere le feature con cura; scegliendo quelle che più caratterizzano le varie texture



**FIGURE 12.23**  
Generating a co-occurrence matrix.

Function `graycomatrix` in the Image Processing Toolbox computes co-occurrence matrices. The syntax in which we are interested is `comatrix`

`[ GS , FS ] = graycomatrix ( f , ' NumLevels ' , n , ' Offset ' , offsets )`

This syntax generates a series of cooccurrence matrices stored in GS. The number of matrices generated depends on the number of rows in the  $q \times 2$  matrix, offsets. Each row of this matrix has the form [ row\_offset , col\_offset ] , where row\_offset specifies the number of rows between the pixel of interest and its neighbors, and similarly for col\_offset. For instance, offsets = [ 0 1 ] for the example in Fig. 12.23.

Parameter **NumLevels** specifies the number of level "bands" into which the intensities off are divided, as explained earlier (the default is 8), and FS is the resulting image, which is used by the function to generate GS. For example, we generate the co-occurrence matrix in Fig. 12.23 as follows

```
>> f2 = imread('Fig1224(b).tif');
>> G2 = graycomatrix(f2,'NumLevels', 256);
>> G2n = G2/sum(G2(:)); % Normalized matrix.
>> stats2 = graycoprops(G2, 'all'); % Descriptors.
```

Property	Description	Formula
'Contrast'	Returns a measure of the intensity contrast between a pixel and its neighbor over the entire image. Range = [0 (size(G, 1) - 1)^2] Contrast is 0 for a constant image.	$\sum_{i=1}^K \sum_{j=1}^K (i - j)^2 p_{ij}$
'Correlation'	Returns a measure of how correlated a pixel is to its neighbor over the entire image. Range = [-1 1] Correlation is 1 or -1 for a perfectly positively or negatively correlated image, respectively. Correlation is NaN for a constant image.	$\sum_{i=1}^K \sum_{j=1}^K \frac{(i - m_r)(j - m_c)p_{ij}}{\sigma_r \sigma_c}$ $\sigma_r \neq 0; \sigma_c \neq 0$
'Energy'	Returns the sum of squared elements in G. Range = [0 1] Energy is 1 for a constant image.	$\sum_{i=1}^K \sum_{j=1}^K p_{ij}^2$
'Homogeneity'	Returns a value that measures the closeness of the distribution of elements in the G to the diagonal of G. Range = [0 1] Homogeneity is 1 for a diagonal G.	$\sum_{i=1}^K \sum_{j=1}^K \frac{p_{ij}}{1 +  i - j }$
'All'	Computes all the properties.	

```

>> maxProbability2 = max(G2n(:));
>> contrast2 = stats2.Contrast;
>> corr2 = stats2.Correlation;
>> energy2 = stats2.Energy;
>> hom2 = stats2.Homogeneity;
>> for I = 1:size(G2n, 1);
    sumcols(I) = sum(-G2n(I,1:end).*log2(G2n(I,1:end)...
                      + eps));
end
>> entropy2 = sum(sumcols);

```

## Texture - maschere energetiche

Possiamo riconoscere una texture tramite delle maschere:

Dove la maschera risponde bene significa che riconosce la texture. Associamo un vettore di caratteristiche a ciascun punto

Maschera di Laws - procedura:

- Per ogni pixel si sottrae la media del suo intorno (per ridurre l'effetto dell'illuminazione)
- si calcolano le maschere (5x5)
- si calcolano le seguenti feature

L5E5/E5L5	L5S5/S5L5
L5R5/R5L5	E5E5
E5S5/S5E5	E5R5/R5E5
S5S5	S5R5/R5S5
R5R5	

Nota: i descrittori che usiamo **non** sono invarianti per scala

Kernel name	Kernel value using 1-D kernels	description - features extracted from texture
$L_3 L_3$	$L_3^T L_3$	expressing grey level intensity of 3 neighbouring pixels in both horizontal and vertical direction
$L_3 E_3$	$L_3^T E_3$	edge detection in horizontal direction and grey level intensity in vertical direction
$L_3 S_3$	$L_3^T S_3$	spot detection in horizontal direction and grey level intensity in vertical direction
$E_3 L_3$	$E_3^T L_3$	grey level intensity in horizontal direction and edge detection in vertical direction
$E_3 E_3$	$E_3^T E_3$	edge detection in both vertical and horizontal direction
$E_3 S_3$	$E_3^T S_3$	spot detection in horizontal direction and edge detection in vertical direction
$S_3 L_3$	$S_3^T L_3$	grey level intensity in horizontal direction and spot detection in vertical direction
$S_3 E_3$	$S_3^T E_3$	edge detection in horizontal direction and spot detection in vertical direction
$S_3 S_3$	$S_3^T S_3$	spot detection in both horizontal and vertical direction

possible  $5 \times 5$  Laws' masks

L5L5	E5L5	S5L5	W5L5	R5L5
L5E5	E5E5	S5E5	W5E5	R5E5
L5S5	E5S5	S5S5	W5S5	R5S5
L5W5	E5W5	S5W5	W5W5	R5W5
L5R5	E5R5	S5R5	W5R5	R5R5

**Momenti di inerzia:** features invarianti a molte variazioni (scala, rotazione, ...)

We compute the moment invariants using function `invmoments`:

```
>> phi = invmoments(f);
```

are the moment invariants of the original image. Usually, the values of moment invariants are small and vary by several orders of magnitude, as you can see:

```
>> format short e  
>> phi  
phi =  
1.3610e-003 7.4724e-008 3.8821e-011 4.2244e-011  
4.3017e-022 1.1437e-014 -1.6561e-021
```

We bring these numbers into a range easier to analyze by reducing their dynamic range using a  $\log_{10}$  transformation. We also wish to preserve the sign of the original quantities:

```
>> format short  
>> phinorm = -sign(phi).*log10(abs(phi))  
phinorm =  
2.8662 7.1265 10.4109 10.3742 21.3674 13.9417 -20.7809
```

*Nota: assente nel libro*

## LBP di Base

### LBP generalizzato

Si usa un intorno di punti P e un raggio R per valutare il valore di P  
Invariate ai cambi di illuminazione locale, MA non alla rotazione

Omogenee: se sono omogenee, la loro differenza mantiene il segno

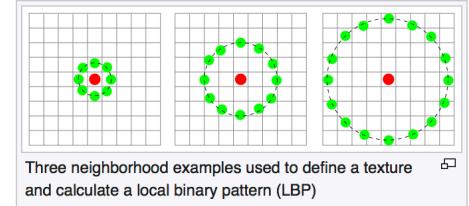
#### Invarianza alla rotazione:

Preso il descrittore calcolato, lo ruoto fino ad avere il pixel nel punto minimo

### Scegliere le feature che permettono di ottenere lo scopo prefissato

The LBP feature vector, in its simplest form, is created in the following manner:

- Divide the examined window into cells (e.g. 16x16 pixels for each cell).
- For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.
- Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
- Compute the [histogram](#), over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional [feature vector](#).
- Optionally normalize the histogram.
- Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.



Three neighborhood examples used to define a texture and calculate a local binary pattern (LBP)

The feature vector can now be processed using the [Support vector machine](#), [extreme learning machines](#), or some other machine-learning algorithm to classify images. Such classifiers can be used for face recognition or texture analysis.

A useful extension to the original operator is the so-called uniform pattern<sup>[8]</sup>, which can be used to reduce the length of the feature vector and implement a simple rotation invariant descriptor. This idea is motivated by the fact that some binary patterns occur more commonly in texture images than others. A local binary pattern is called uniform if the binary pattern contains at most two 0-1 or 1-0 transitions. For example, 00010000(2 transitions) is a uniform pattern, 01010100(6 transitions) is not. In the computation of the LBP histogram, the histogram has a separate bin for every uniform pattern, and all non-uniform patterns are assigned to a single bin. Using uniform patterns, the length of the feature vector for a single cell reduces from 256 to 59. The 58 uniform binary patterns correspond to the integers 0, 1, 2, 3, 4, 6, 7, 8, 12, 14, 15, 16, 24, 28, 30, 31, 32, 48, 56, 60, 62, 63, 64, 96, 112, 120, 124, 126, 127, 128, 129, 131, 135, 143, 159, 191, 192, 193, 195, 199, 207, 223, 224, 225, 227, 231, 239, 240, 241, 243, 247, 248, 249, 251, 252, 253, 254 and 255.

### Oggetti in moto

La differenza tra fotogrammi mi permette di individuare i movimenti

Distinguere background e foreground

Il background cambia: luce, posizione degli oggetti, scena, posizione della telecamera; devo identificare i cambiamenti

### Video-sorveglianza:

La camera è statica: sfogliatura e mappa del cambiamento mi permette di individuare i cambiamenti

### Monitorare i movimenti:

Separare la persona dalla sfondo, e vederne i movimenti nel tempo

### Problemi nel change detection:

- Ombre viste come oggetti
- cambi di luce
- Waveing trees: movimenti della scena naturale. Individuare regioni i cui cambiamenti minimi non sono rilevanti
- Foreground aperture: oggetti con colazione uniforme su sfondo uniforme, gli spostamenti minimi sono difficili da rilevare
- Camouflage: stesso colore dello sfondo

### Algoritmo single difference:

Due frame consecutivi, differenza che viene poi sogliata per T

Problemi:

- Ghosting: tanto maggiori quanto il soggetto è veloce
- Foreground aperture
- Individua solo gli oggetti in movimento continuo: se si ferma, non esiste più

Difficile settare il valore di soglia

### **Algoritmo double difference:**

Prendo tre frame consecutivi e li combino tra loro

- Se gli oggetti si muovono più velocemente dei frame/sec della camera, il double difference individua i movimenti
- Riesco a eliminare le aree di ghost
- Individua solo gli oggetti in movimento continuo: se si ferma, non esiste più

### **Algoritmi sulla differenza dal background**

Se prendiamo tanti frame e li mediamo, la media tenderà al background, rimuovendo gli oggetti  
Rimuovo i problemi precedenti

Uso i dati per calcolare i valori delle soglie da utilizzare (ad esempio, cambiamenti di luce sul muro)

**Problema:** il background non si aggiorna mai, e se cambiasse produrrebbe sempre un rumore

### **Aggiornamento del background:**

Usare una media di frame per aggiornare il background, e il foreground, **ma a velocità diverse**

### **Aggiornamento a due pesi:**

Usare due valori alpha e beta diversi per riconoscere background e foreground

### **Algoritmi statistici**

Regione costante permette di stimare il cambiamento con una statistica gaussiana

Usiamo **media e varianza**

**Queste statistiche possono essere aggiornate anziché ricalcolate**

- Waving trees: Sul background c'è molta differenza, che potrebbe dar problemi

### **Inframe difference model**

Possiamo usare altri descrittori (esempio, valore massimo)

### **Algoritmo W4**

Differenza tra frame, può causare falsi positivi

### **Algoritmi statistici: block NNCF**

Consideriamo patch anziché pixel: divido l'immagine in regioni di pixel.

Blocchettizzazione, ma statistiche più robuste. Normalizzando le regioni di pixel ottengo una parziale indipendenza dai cambiamenti di luce

### **Tracking**

Individuo una texture in uno frame, e visualizzo come la texture si è spostata da un frame all'altro

### **Problema con le ombre, che non devono essere parte del soggetto**

-> considero la hue (l'ombra ha tinta scura)

### **Problema con l'illuminazione non uniforme**

-> white balancing

### **Cambiamento di stato della descrizione di un soggetto:**

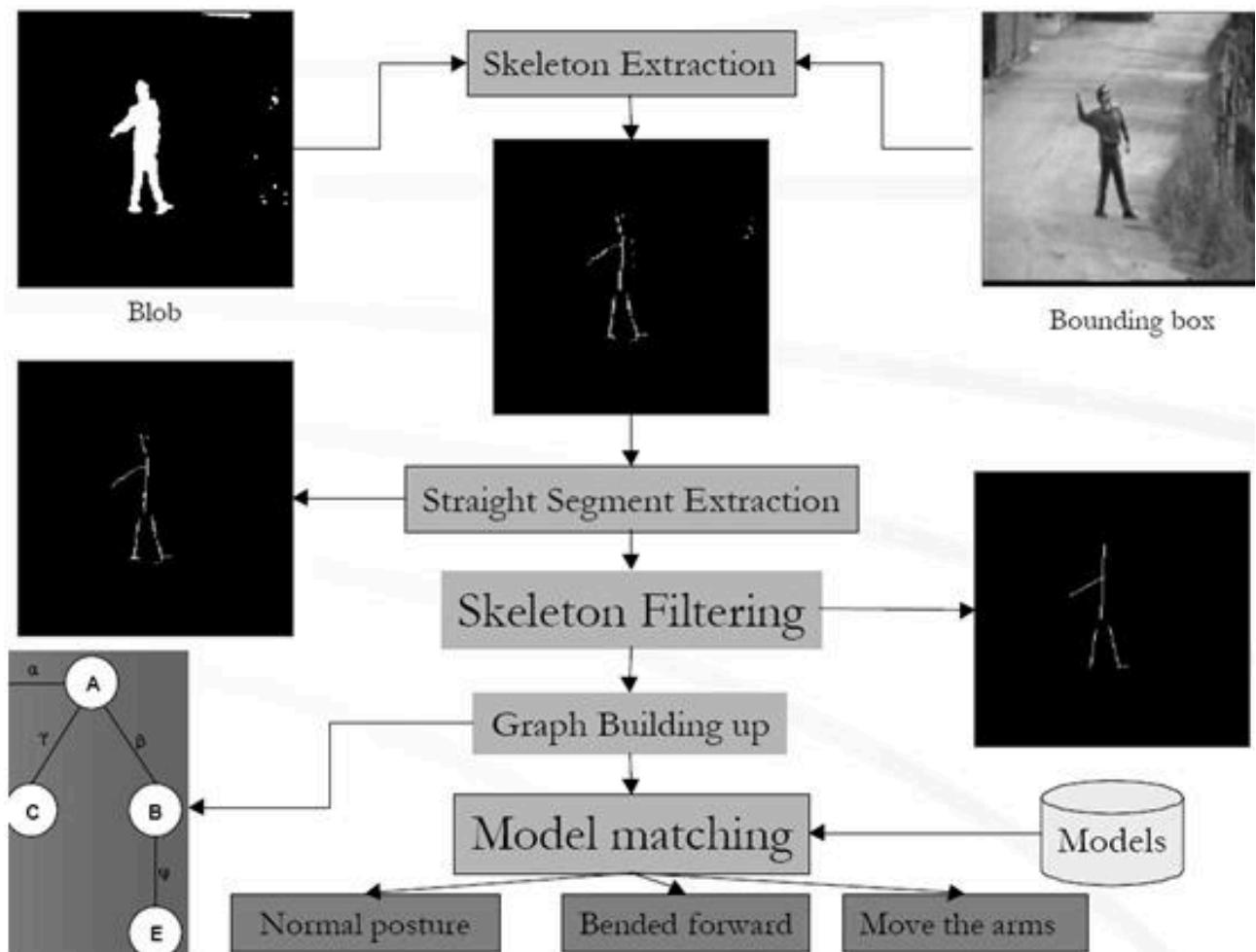
Se il soggetto *cambia*, ad esempio vestiti, segno il cambiamento di stato per continuare a seguirlo

## Classificare un comportamento

Segnare la traiettoria del movimento, e il soggetto del movimento, per poter classificare il comportamento

## Skeleton

Forma del soggetto segmentato e *minimalizzato*. **Posso usare lo skeleton per riconoscere il soggetto**



## Quantizzazione

### Naive color quantization

Diminuisco i bit per canali (da 8 bit per canale a 8 bit per tutti e tre i canali RGB)

*Molto semplificativa -> può essere usata in preprocessing per riconoscimento / calcolo automatico, non per rappresentazione*

La tavola di conversione tra corrispondenti RGB è detta *palette* (*tavolozza di colori*)

Possiamo usare una *look up table* per ridurre i costi computazionali

In un'immagine ciascun pixel è un puntatore nella look up table

-> aggiornando la look up table l'immagine *cambia*

### Palette safe RGB

216 colori, palette standard usata per il web

## Riconoscere oggetti colorati

Quantizzazione in HSV, i colori mantengono la stessa tinta e quantizzo saturazione e intensità  
-> posso quindi riconoscere gli oggetti per colore

Quantizzazione per “categorie di colori”

-> pochi colori, in cui sono mappati i colori dell’immagine da quantizzare  
-> può essere utile per la segmentazione / descrizione con il colore

### **Calcolare la palette**

Clustering veloce per raggruppare i colori da usare nella palette

-> opportune statistiche dell’immagine

Es:

- **Arbib's algorithm**

- **Algoritmo Median Cut Algorithm**

- Come funziona?

1. Prendo un rettangolo che contiene tutti i colori
2. Prendo il semiasse più lungo
3. Divido in modo da avere la stessa quantità di colori a destra e a sinistra
4. Itero per ottenere un numero di “cubotti”
5. Rappresento i cubetti con i loro centroidi

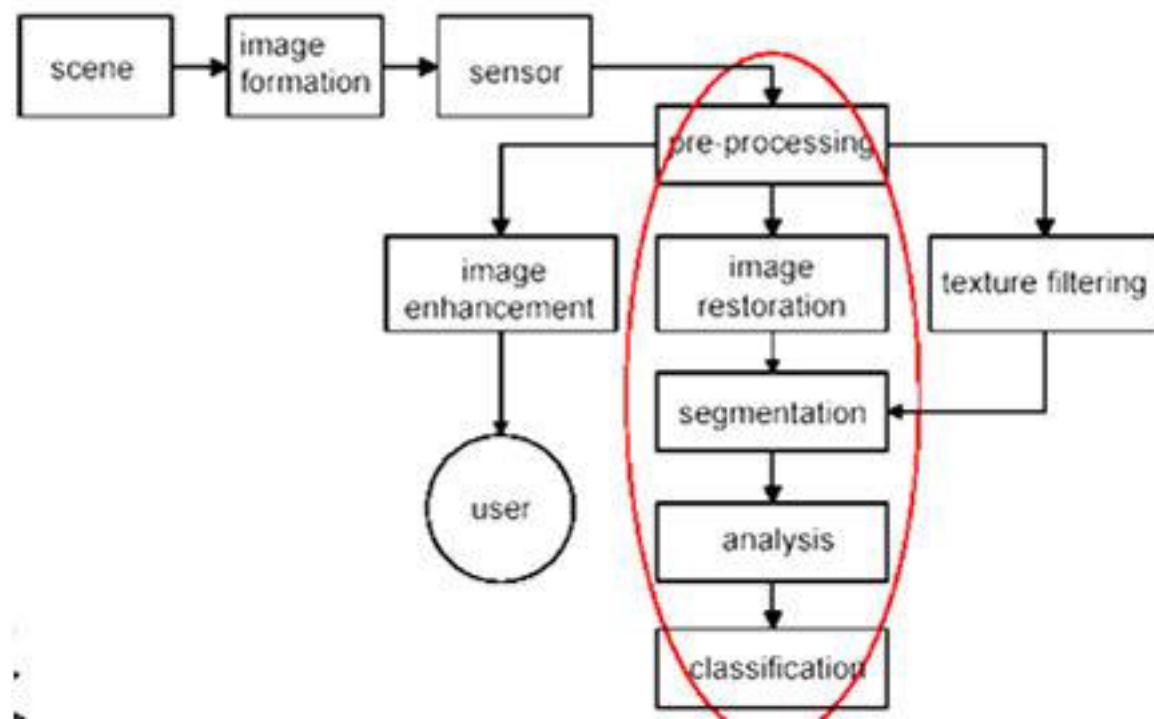
### **Implementation for color quantization**

Suppose we have an image with an arbitrary number of pixels and want to generate a palette of 16 colors. Put all the pixels of the image (that is, their RGB values) in a **bucket**. Find out which color channel (red, green, or blue) among the pixels in the bucket has the greatest range, then sort the pixels according to that channel's values. For example, if the blue channel has the greatest range, then a pixel with an RGB value of (32, 8, 16) is less than a pixel with an RGB value of (1, 2, 24), because  $16 < 24$ . **After the bucket has been sorted, move the upper half of the pixels into a new bucket.** (It is this step that gives the median cut algorithm its name; the buckets are divided into two at the median of the list of pixels.) **Repeat the process on both buckets, giving you 4 buckets, then repeat on all 4 buckets, giving you 8 buckets, then repeat on all 8, giving you 16 buckets. Average the pixels in each bucket and you have a palette of 16 colors.**

Since the number of buckets doubles with each iteration, this algorithm can only generate a palette with a number of colors that is a power of two. To generate, say, a 12-color palette, one might first generate a 16-color palette and merge some of the colors in some way.

# Morfologia matematica

## Morfologia binaria



Sistema di operatori algebrici che permette di :

- eliminare distorsioni di forma
- Caratterizzare morfologicamente gli oggetti, per scomporre gli oggetti

Si applicano sui pixel di un'immagine

**Elemento strutturante:** immagini di dimensione ridotte che parametrizzano le operazioni

**Operazioni:**

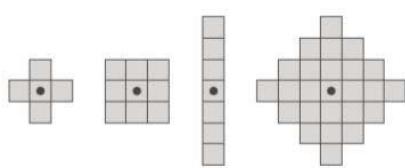
- **Dilation**
- **Erosion**

Dette anche somma e sottrazione di **Minkowski**.

Operatori più complessi sono costruiti come combinazione dei precedenti

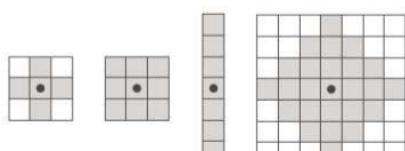
- apertura (opening),
- Chiusura (closing), ....

Cerchiamo di “**regolarizzare**” le forme per semplificare la segmentazione



**Centro** dell'elemento strutturante può non essere il centroide dell'oggetto

Può essere visto come la punta del pennarello, e il punto centrale (in nero) è la punta dell'applicazione



I restanti pixel dell'elemento “riempiono” l'immagine (binarie) con il valore

Pennarello per **colmare** zone mancanti più piccole dell'elemento ristrutturante

## Dilation

- Unisce regioni la cui distanza è più piccola del filtro
- Riempie buchi

### Proprietà

Posso usare le proprietà per rendere l'applicazione computazionalmente conveniente

- **Commutativa**
- **Associativa**
- **Estensiva:** l'insieme originario è contenuto nell'insieme dilatato
- **Crescente** ->

$$A \subseteq C \Rightarrow A \oplus B \subseteq C \oplus B$$

## ROI

Region of interest. Dentro regioni *rettangolari* posso individuare il testo e analizzarlo

## Erosion

### Complemento della dilation

La punta è sul bordo della forma, e cancello quello che tocca l'elemento strutturante  
Elimino elementi dell'immagine più sottili della dimensioni dell'elemento strutturante  
-> assottiglio la forma

### Contare persone

Anziché contare direttamente le componenti connesse, uso l'operazione di erosion per dividere le regioni

Domanda:

- Contare monete sovrapposte per non più del 50 %
- > **Uso una funzione di erode per distinguere le regioni**

- Dimensione dei quadrati di dimensione  $> 15 \times 15$

- **Erosion con elemento strutturante 16, per ottenere il centro dei quadrati**

## Opening

Apertura di A con B

Erosione con B e di un'espansione con B dell'immagine erosa

Ottimo per trovare oggetti in un'immagine secondo le loro dimensioni:

- Monete tra loro sovrapposte
- Macro-regioni
- Angoli, figure, linee verticali: uso una maschera rettangolare sottile e lunga in verticale, scegliendo le dimensioni corrette
- Oggetti di certe forme (circolari, ...): discriminò sempre per dimensione; uso una maschera che elimini gli oggetti indesiderati e lasci quelli cercati, e poi uso una deletion per recuperare la regione dell'oggetto rimosso

## Closing

Riempire i buchi: cercare di regolarizzare le regioni con difetti

Dilation seguita da erosion -> chiudo i buchi, e successivamente riporto i bordi dilatati e ancora aperti alle loro dimensioni originali

### Filtraggio morfologico:

Usare numeri diversi di erosion e deletion per ottenere l'effetto desiderato, magari cambiando l'elemento strutturante. Ricorda: si possono usare forme di maschere tra loro diverse

## Thinning

Trovare la linea mediana di figure: skeleton (persone, etc.)

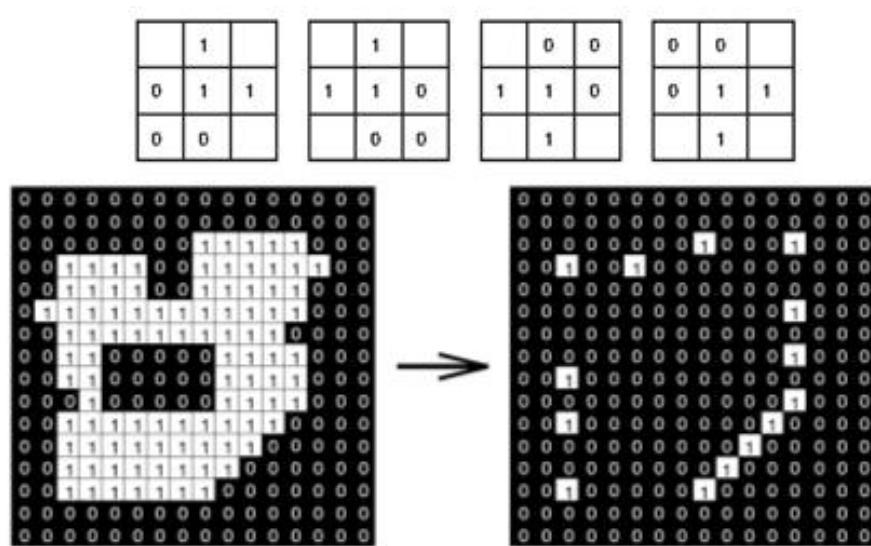
## Edge detection

Bordo interno: Originale - Erosion dell'originale con una maschera

$$A - (A \ominus B)$$

## Trasformata hit-and-miss

Individuare features nell'immagine



- Corner di un'immagine. Possiamo individuarli ragionando sulla forma dello spigolo, e sul fatto che si trova tra immagine e background
- Punti in cui si toccano le linee, punti isolati: appropriate maschere in cui combinazioni punto di applicazione è circondato da 0 che **riproduce il pattern da individuare: template matching**

## Trasformata distanza

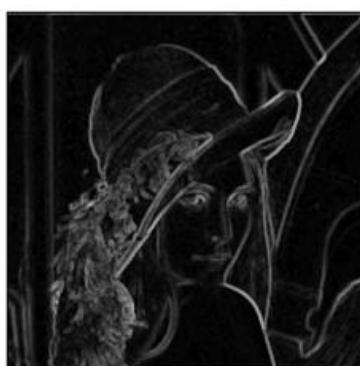
Itero l'erosione con una maschera a croce (4-connettori) o quadrata (8-connettori), e etichetto ciascun pixel di bordo con il numero dell'iterazione: ottengo la distanza di ciascuna curva di bordo dal background

### Gradiente morfologico

$$\text{MorphGrad}(f) = (f \oplus k) - (f \ominus k)$$



Immagine iniziale



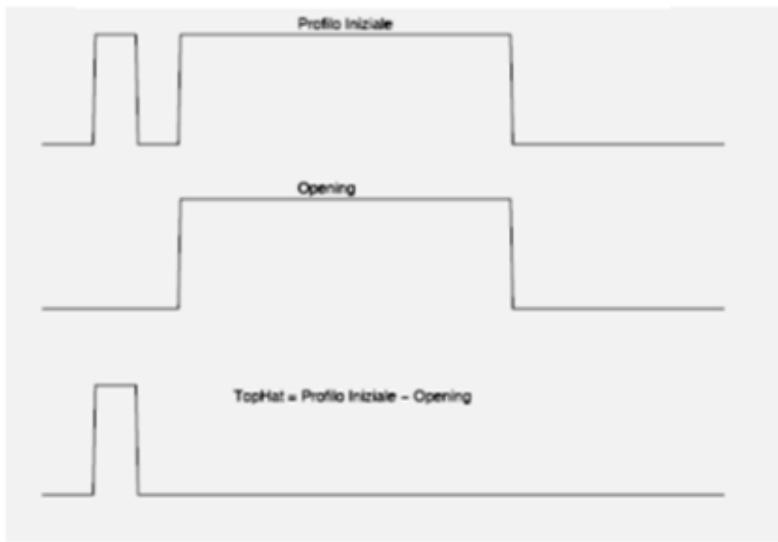
Gradiente Morfologico

Differenza tra immagine erosa e immagine dilatata: bordi dell'immagine

**Nota: indica anche la forza del bordo (a livelli di grigio)  
-> possibile thresholding**

## Hat trasform

$$TopHat(f) = f - (f \circ k)$$



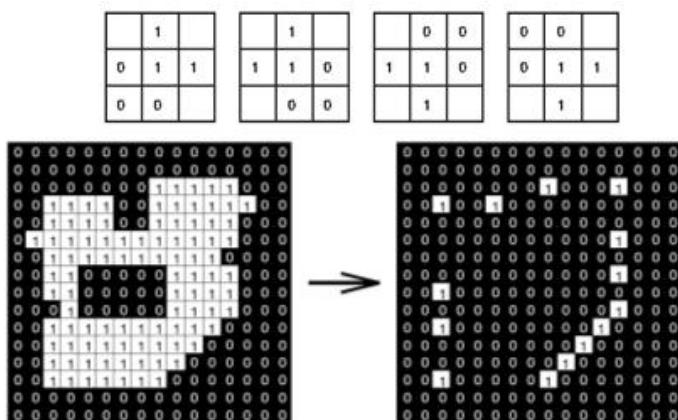
Se eseguo tante erosion e tante deletion rimane una parte chiara (più dello sfondo) degli oggetti cercati (più piccolo di una certa area) e tramite thresholding più chiaro di una certa soglia

L'opening elimina i picchi della funzione più piccoli dell'elemento strutturante, la differenza fra la funzione originaria e l'opening è una funzione in cui sono presenti solo tali picchi. La trasformata Top-Hat consente quindi di evidenziare e/o individuare i picchi della funzione di dimensione inferiore all'elemento strutturante. Nel caso delle immagini l'operatore viene impiegato per evidenziare e/o individuare piccoli (rispetto all'elemento strutturante) dettagli chiari.

## Trasformata Hit and Miss

$$A \otimes (B_1, B_0) = (A \ominus B_1) \cap (A^c \ominus B_0) \quad B_1 \cap B_0 = \emptyset.$$

I quattro elementi strutturanti che possono essere utilizzati per individuare i “corner” in un’immagine binaria (si tratta dello stesso elemento base nelle 4 possibili orientazioni). L’output è l’OR delle quattro operazioni.  
 $B_1$  e  $B_0$  sono rispettivamente l’insieme dei punti a “1” e l’insieme dei punti a “0” dell’elemento strutturante  $B$ .



# Edge detection

Edge: cambiamento repentino di un'immagine  
Riconoscere la texture e distinguere per edge e texture

**Region based:** usato per individuare edge in oggetti con texture  
Distinguere sulla base di descrittori, per evitare ombre o riflessi

## Derivata prima:

- nulla dove il segnale è piatto
  - Positiva quando il segnale è crescente
  - Negativa quando il segnale è decrescente
  - Modulo proporzionale all'entità della variazione
- Più il cambiamento è repentino, più la derivata prima vale

**Nota:** dobbiamo parlare di segno della derivata prima, e dell'angolo del gradiente

## Procedura:

Filtro derivativo -> modulo -> sogliatura T -> maschera da usare nell'immagine originale per isolare il segnale

**Direzione:** tangente all'edge; massimo cambiamento

*Problema: il rumore*

Il risultato della derivata prima amplia il rumore: il risultato sarà la somma del segnale + quella del rumore

## Soluzioni:

- maschere più grandi
- Filtraggio per ridurre il rumore (media, guassiano, mediano, ...)

*Semplificare computazionalmente: creo un filtro combinando filtro di riduzione del rumore e filtro di edge (Smoothing \* Derivativo)*

## Operatori:

### Prewitt

Combino un filtro media ( $\begin{bmatrix} -1 & 0 & +1 \end{bmatrix}$ ) con una matrice di edge orizzontali - verticali

### Sobel

Combino l'idea del filtro guassiano per gli edge (più peso al centro  $\begin{bmatrix} +1 & +2 & +1 \end{bmatrix}$ ) e lo combino con un filtro di media

### Derivata della funzione gaussiana

Distribuzione gaussiana; derivo la funzione rispetto a x ed y: maschera che combina le due funzioni

Variando la sigma tolgo più rumore (e ingrandisco gli edge)

*Problema: i bordi sono più spessi di un pixel -> posso poi usare un algoritmo di thinning per estrarre i bordi spessi un pixel, usando le informazioni (direzione e magnitudo) disponibile*

## Derivata seconda:

Sul flesso si annulla: **zero crossing**. Da qui possiamo individuare il picco della derivata prima, ossia l'edge massimo. Un punto spesso un pixel si può ottenere sfogliando gli edge e successivamente tramite zero-crossing estrarre i punti di annullamento della derivata seconda (il centro dell'edge)

**In alternativa, si può cercare il punto di cambio di segno tra positivo e negativo**

**Operatore: Laplaciano**

Problema: molto sensibile al rumore. Da applicare **dopo** lo smoothing

**Operatore: Laplaciano del gaussiano LoG**

Derivata seconda della gaussiana. Detector ottenuto dalla derivata seconda di un filtro di smoothing gaussiano.

**Algoritmo:**

1. Soglio il gradiente
2. Applico LoG
3. Zero crossing di LoG -> ottengo i punti di annullamento (ma molto rumore)
4. And tra (3) e (1) -> edge

*Problemi: edge interrotti, rumore ancora presente, ...*

Ottimo per individuare oggetti in condizioni ottimali, con background omogeneo e in immagini prive di texture.

## Canny edge detector

Modello matematico per edge e rumore

Idealmente: modello della realtà tramite assunzioni: edge a gradino con una certa altezza e con un certo rumore

- 1) usa la derivata prima
- 2) Usa un meccanismo di isteresi: doppia soglia per distinguere edge, non edge e "via di mezzo"

**Proprietà:**

- buona rilevazione: minimizzare la probabilità di errori
- Buona localizzazione: distanza tra l'edge individuato e la sua effettiva posizione è minima
- Risposta singola: unica risposta per ogni edge

La forma del detector è approssimabile dalla deviata di una gaussiana. Si ispira all'implementazione di LoG (laplaciana di una gaussiana, implementata nella Mexican Hat Trasform)

Uso la direzione dell'edge:

Se due edge guardano nella stessa direzione, questi possono essere connessi;  
Altrimenti si può supporre che siano relativi a due diversi oggetti

**Algoritmo**

1. Smooth dell'immagine con un filtro di gauss con parametro  $\sigma$
2. Calcolo del gradiente dell'immagine
3. Identificazione dei possibili edge (non-maximum-suppression):
  1. Per ogni regione 3x3 di un pixel di edge, si assegna al valore centrale:
  2.  $M(a, b)$  (magnitudo del gradiente nel punto) se  $M(a, b) > M$ (nei due punti di direzione dell'edge adiacenti ad  $(a, b)$ )
  3. 0 altrimenti
4. Rimozione degli edge non rilevanti (isteresi) mediante analisi a doppia soglia ( $a, b$ )
  1. Si pongono due soglie  $T_1$  e  $T_2$ :  $T_2 > T_1$
  2. Si generano due immagini di Magnitudo dell'edge, una per ogni soglia
  3. Si rimuovono dalle immagini di LowEdge i valori di HighEdge
  4. Per ogni punto di HighEdge, si pone il punto nell'immagine risultato
  5. si prendono tutti i punti 8 connessi di LowEdge al punto preso e si pongono nell'immagine risultato
5. Thickening dell'edge (ad esempio tramite morfologia matematica)

**Non-maximum suppression**

Vengono selezionati solo i valori massimi nella direzione del gradiente

**Tre parametri:** deviazione standard, soglia superiore, soglia inferiore

**Estensione a COLORI**

- Usare la luminanza e procedere all'erge detection su questo canale
- Distinguere tra i tre canali RGB e combinare i risultati con AND o OR
- **Metodo di zenzo**

**Nota:** se cambio spazio colore, e suo HSV - HSI, ricorda che la Hue è circolare la distanza tra due tinte potrebbe non essere immediata

# Edge linking

Legare edge che potrebbero essere interrotti

## Metodi locali

- Se due edge hanno stessa direzione e distanza minima, li fondiamo chiudendo l'interruzione
  - Nota: funziona per piccole distanze

*Esempio:* posso trovare le targhe individuando (per shape) i rettangoli posti dietro la macchina.  
Edge linking può essere utile per unire i punti della targa

## Trasformata di Hough

*Operatore globale:* analizza tutti i pixel per prendere decisioni

Si sentano criteri globali per individuare i punti di edge

Analizzare collinearità in modo efficiente:

### Accumulo di evidenza:

Copie (m, q) comuni a molte rette, una per ogni punto di edge

Ottengo una matrice di accumulazione contenente le informazioni sulle rette presenti nell'immagine

### Alternativa:

Rappresentare una retta come: distanza dall'origine, angolo del vettore perpendicolare alla retta.  
Ottengo un fascio di iperboli secondo lo stesso ragionamento

### Nota:

Non ho bisogno di una connessione nell'immagine originale per linkare gli edge: anche con poche informazioni posso ottenere un buon risultato

*Problema: le rette, che ottengo ridisegnando dalla matrice l'immagine originale, vanno a infinito*

Punto: coppia di parametri associati a n rette

Due punti collinari -> avranno parametri uguali -> la matrice nei parametri esprime il numero di pixel per la retta espressa dai parametri

Faccio sopravvivere solo le regioni con gradiente diverso da 0

- 1) trovo gli edge
- 2) Soglia sulla magnitudo per ottenere bordi forti
- 3) Trasformata di hough
- 4) Massimi locali (peaks)
- 5) Disegno le rette
- 6) Sovrappongo all'immagine originale
- 7) Unisco rette distanti meno di una certa distanza
- 8) Tolgo le rette fuori dalle regioni di alto gradiente

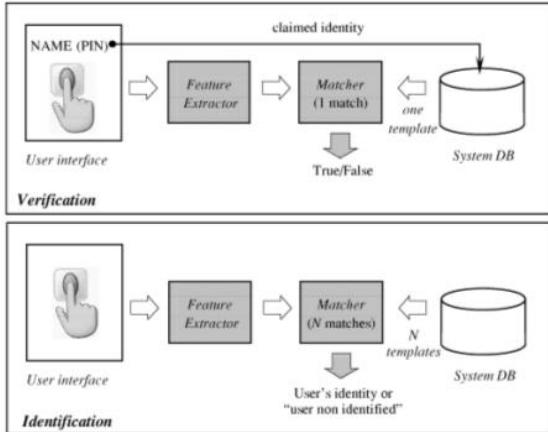
Nota: soglio i valori dei punti di accumulazione, per prendere solo le rette composte più da t pixel:  
possiamo interpretare t come lunghezza minima della retta da cercare

**Può essere generalizzata a qualsiasi figura geometrica: circonferenza, ellissi, ... cambiando la forma dell'equazione sulla base dell'equazione parametrica delle figure; cambiano i parametri**

# Rappresentazione e descrizione

*Problema di sicurezza:* lascio i dati sensibili in locale alla persona, per non salvare i dati in locale

## Riconoscimento vs classificazione



Riconoscimento: corrispondenza tra il template e un template estratto dal database

Classificazione: riconoscere per ogni template la corrispondenza con un template nel database

### Template matching

Ricerca di una sotto-immagine (template) in un'immagine -> più è corretta la sovrapposizione tra template e porzione di immagine, più il valore è alto

Nota: forza bruta; rotazione, scala, luce possono dare problemi e richiedono numerose computazioni

### Rigid template matching

SSD = sum of squared difference; differenza dei quadrati di due valori

Se template e porzione di immagine sono uguali, trovo i valori per massimizzare  $2 * \text{TemplateTrasposto} * \text{Immagine}$

NSSD = normalizzo dividendo per il modulo, per rendere indipendente al contrasto

ZNSSD = normalizzo per la differenza del valore dal valore medio, in modulo

$(\text{Immagine} - \text{immagine media}) * (\text{template} - \text{template medio})$

**Nota:** bisogna sogliare i valori dell'immagine ottenuta dalla correlazione con l'immagine di template (normalizzo e sottraggo template/immagine della sua media)

- SSD: distanza tra template e immagine (minore è la distanza, più simili sono)
- NCC: coefficiente di correlazione (maggiore è il coefficiente, più template / immagine sono correlati)

### Problema: controllo

Soglio un po' più basso, trovo diversi punti possibili, verifico con altri metodi quali di questi punti matchano con il template

-> non ho falsi negativi (soglia molto bassa, *recall oriented*)

Nota: Z significa **zero-mean**, si adotta per traslare il coefficiente in modo che abbia media **zero**

### Non ho risolto il problema della scala:

1. Posso usare più template, uno per ogni scala -> *tempo di calcolo elevato*

-> template matching con descrittori più potenti

2. Da problemi su template tra loro simili (faccia, ... )

## **Multi risoluzione**

Costruisco una piramide di immagini, ciascuna scalata rispetto alla precedente, e cerco il template nell'immagine a più scale per trovare la corrispondenza a un certo livello di scala

## **Matching / Riconoscimento**

1. Rappresentazione / descrizione del modello
2. Riconoscimento: Processo decisionale che accetta / rifiuta secondo una certa soglia
  - La soglia può essere ricavata da un dataset di training
3. Classificazione: assegnare una classe ai dati di input

*Problema: riconoscere il volto*

1. Uso lo skin-tone (caratteristica), ma come la rappresento?
  - posso usare l'area di pixel di skin -> non distingue tra volto e braccia ...
2. Devo rielaborare l'immagine, estraendo informazioni utili
3. Posso localizzare l'istogramma, distinguendo l'area del volto e successivamente occhi, ...

# Features

Idealmente la descrizione dovrebbe essere:

- sufficiente (per il raggiungimento dello scopo)
- di ampia applicabilità (capace di descrivere molte classi di oggetti)
- non ambigua (due oggetti distinti non possono avere la medesima rappresentazione)
- unica (ogni oggetto distinto ha una univoca descrizione)
- stabile rispetto al rumore
- invariante per rotazione, traslazione e scala (se richiesto)
- generativa (è possibile ricostruire l'oggetto a partire dalla sua rappresentazione)
- conveniente (per gli scopi dell'applicazione)

La **rappresentazione** di una regione può essere basata su caratteristiche esterne (cioè del boundary), su caratteristiche interne (cioè legate ai pixel che la compongono), o su caratteristiche topologiche (cioè legate a caratteristiche che rimangono inalterate quando una figura viene deformata a piacere). In generale, una rappresentazione esterna conveniente quando sono importanti le caratteristiche di forma della regione, mentre una rappresentazione interna è più adeguata se si ritengono importanti caratteristiche della superficie, come il colore o la tessitura. Spesso più tipi di caratteristiche sono importanti (per esempio, nel retrieval di immagini basato sul contenuto). Una volta scelte le caratteristiche da rappresentare, ne va effettuata una **descrizione** appropriata allo scopo finale della elaborazione, mediante appositi descrittori

---

## Contorno

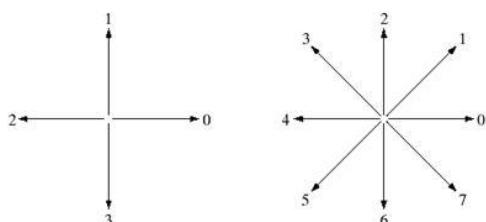
Per l'estrazione del contorno di un oggetto (ovvero della sequenza ordinata di pixel che lo compongono) viene normalmente impiegata una semplice tecnica di inseguimento che percorre il bordo sempre nella stessa direzione fino ad incontrare un pixel già visitato.

---

## Chain code

Il chain code (o Codice di Freeman) descrive esattamente oggetti binari. Il contorno è considerato come un percorso 4- o 8-connesso. Dato un punto di inizio ( $x,y$ ), si codifica la sequenza ordinata dei singoli passi di cui consiste il contorno. I passi sono ordinati come in figura. La codifica della forma è la permutazione che produce il numero più piccolo

a b  
FIGURE 11.1  
Direction  
numbers for  
(a) 4-directional  
chain code, and  
(b) 8-directional  
chain code.



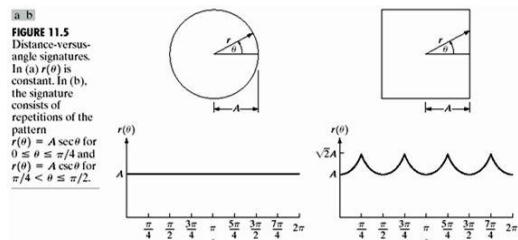
## Approssimazione poligonale

Polygono di minimo perimetro. Dato il boundary, lo si pensi contenuto in un insieme di celle concatenate. Il poligono di minimo perimetro quello che si adatta meglio alla geometria della striscia di celle, appoggiandosi alle pareti delle celle e in genere seguendo il percorso più breve.

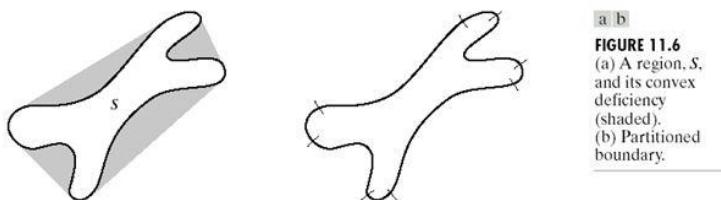
## Signature

- Riduce la dimensione dei dati. **Invariante per traslazione**
- Per ottenere l'invarianza per rotazione si sceglie
- Il punto di origine nel punto pi. lontano dal centroide
- Diverse strategie per ottenere **l'invarianza per scala**: l'idea e' di normalizzare
- I valori nel range [0,1]

La signature permette di semplificare la descrizione delle forme passando da due ad una dimensione (2D->1D). Dalle signature si possono estrarre informazioni ancora più compatte in forma di istogramma o momenti di inerzia.



## Minimo insieme convesso



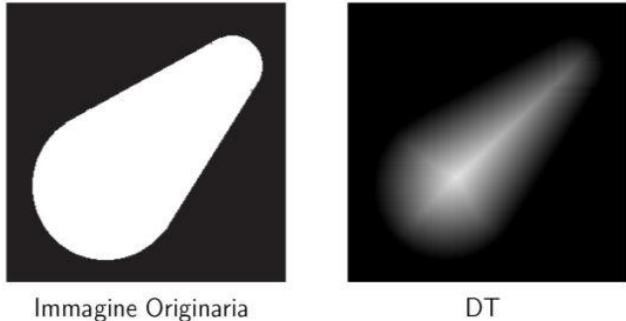
## Minimo Rettangolo Orientato

1. Si calcola l'asse maggiore
2. Si calcola l'asse minore (il suo ortogonale passante per il baricentro)
3. Si calcolano i 4 punti di massima distanza dagli assi C1,C2,C3,C4
4. Si calcolano gli assi paralleli agli assi maggiori e minori passanti per C1,,C4 (l1,l2,w1,w2)
5. Si calcolano i vertici del **Minimo Rettangolo (orientato)**
6. V1,V2,V3 e V4 come intersezione di tali assi

---

## Trasformata distanza

La trasformata distanza di  $F$  rispetto a  $F^*$  è una replica di  $F$  in cui i pixel sono etichettati con il valore della loro distanza da  $F^*$ , calcolata secondo una data metrica.



---

## MAT (Medial Axis Transform)

La MAT è definita come segue:

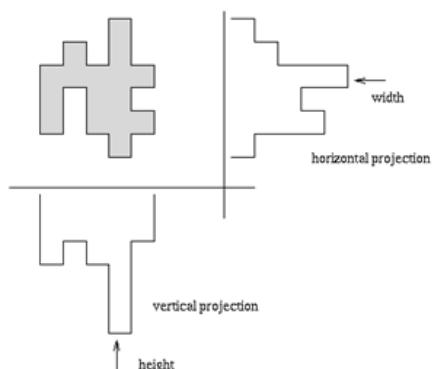
- Un punto  $p$  dell'oggetto appartiene all'asse mediano ("scheletro") se, detta  $d$  la distanza minima fra  $p$  ed il contorno della figura, esistono almeno due punti del contorno situati a distanza  $d$  da  $p$ .
- La MAT è definita nei punti appartenenti all'asse mediano ed il suo valore è dato dalla distanza minima del punto dal contorno

---

## Proiezione

Un utile descrittore di tipo globale fornito dalle proiezioni che forniscono la distribuzione dei pixel della regione secondo alcune direzioni. La proiezione verticale è definita come il numero di pixel

appartenenti alla regione in ogni colonna. La proiezione orizzontale è definita come il numero di pixel appartenenti alla regione in ogni riga. È possibile definire anche proiezioni diagonali, che contano il numero di pixel appartenenti alla regione sulle diagonali.

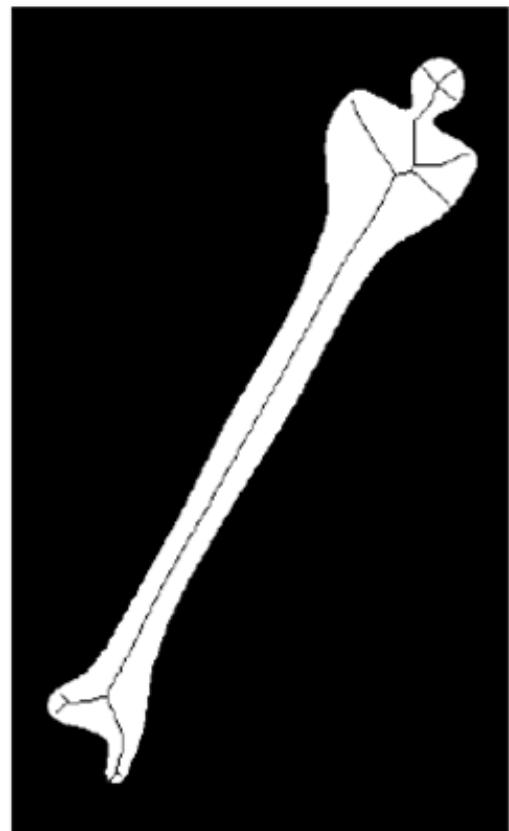


## Scheletrizzazione

In numerose applicazioni si ha a che fare con immagini che contengono linee, oggetti sottili, oggetti di forma allungata. In questi casi la forma dell'oggetto è rappresentabile efficacemente mediante una "linea sottile" localizzata approssimativamente in corrispondenza dell'asse mediano. Si pensi, a titolo di esempio, alle applicazioni di OCR.

Proprietà generali:

- lo scheletro deve consistere di linee di spessore unitario
- le proprietà topologiche dell'oggetto devono rimanere immutate
- lo scheletro deve coincidere il più possibile con la linea mediana
- deve essere poco sensibile al rumore e deve essere stabile



$$a) 2 \leq N(p_1) \leq 6$$

**Step 1** b)  $T(p_1) = 1$

$$c) p_2 \bullet p_4 \bullet p_6 = 0$$

$$d) p_2 \bullet p_4 \bullet p_8 = 0$$

$$a) 2 \leq N(p_1) \leq 6$$

**Step 2** b)  $T(p_1) = 1$

$$c') p_2 \bullet p_4 \bullet p_8 = 0$$

$$d') p_2 \bullet p_6 \bullet p_8 = 0$$

---

## Descrizione di regioni

L' **area A**, nel caso più semplice, è il numero dei pixel della regione  
Il **perimetro P** è definito come la lunghezza del contorno di una regione.

Il **Frastagliamento f** (a volte e' chiamato compattezza C) stima il grado di irregolarità della forma considerata (assume valore 1 quando la regione è un circolare)

La **Compattezza**, detta anche "Form Factor", è definita in base all'area ed il perimetro dell'oggetto.

$$C = \frac{4\pi A}{P^2}$$

### Rettangolarità:

$$R(O) = A(O) / A(R)$$

dove A (R) è l'area del minimo rettangolo che contiene l'oggetto. Dipende dall'orientamento della regione.

Il **diametro** B di un contorno può essere definito come  
**Diam (B)= max h,k [ Dist(ph, pk) ]**

Il **minimo rettangolo** può essere definito a partire dal diametro

Il rapporto dei lati del rettangolo è una misura di **eccentricità** della forma e ci dice quanto l'oggetto è allungato

**Orientamento:** angolo tra la direzione diametro dell'oggetto e il verso positivo dell'asse orizzontale

### Numero di Eulero:

$$E = C - H$$

dove C di componenti connessi dell'immagine e H numero di buchi (lacune)

- può assumere valori negativi
- invariante a trasformazioni rubber sheet è una proprietà

## Momenti

Alcuni momenti corrispondono a proprietà che abbiamo già introdotto. Si può dimostrare che l'insieme infinito dei momenti determina univocamente la forma dell'oggetto. E' possibile quindi utilizzare i momenti come descrittori di forma e distinguere oggetti aventi forma diversa sulla base di un sotto-insieme, opportunamente determinato, dell'insieme dei momenti.

**Per p=q=0, si ottiene l'area dell'oggetto**

$$m_{00} = \sum_x \sum_y f(x, y)$$

## Baricentro

Con i momenti di ordine 1 si individuano le coordinate del centroide o baricentro dell'oggetto.

$$m_{10} = \sum_x \sum_y x f(x, y) \quad m_{01} = \sum_x \sum_y y f(x, y)$$

$$x_c = \frac{m_{10}}{m_{00}} \quad y_c = \frac{m_{01}}{m_{00}}$$

### Descrittori invarianti rispetto a traslazione-scala

E' possibile ottenere dei momenti invarianti rispetto alla traslazione, che vengono detti **momenti centrali**, considerando un sistema di riferimento baricentrico (traslando nel centroide)

I **momenti centrali non sono invarianti rispetto alla rotazione**. I momenti centrali possono quindi essere usati come descrittori di forma solo se l'orientamento degli oggetti e' fisso.

Per ottenere dei momenti invarianti alla variazioni di scala, si considerano i momenti centrali normalizzati:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} \quad \text{dove} \quad \gamma = \frac{p+q}{2} + 1$$

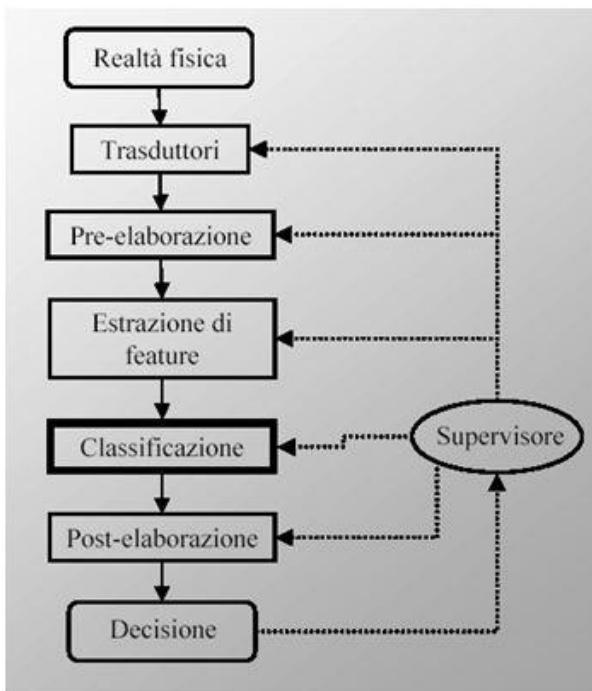
**Spread (S)** rappresenta il momento d'inerzia della regione intorno al suo centroide, ed è invariante per rotazione e per traslazione.

I **momenti del secondo ordine** sono usati per determinare gli assi principali d'inerzia dell'oggetto. Questi forniscono utili indicazioni sull'orientazione dell'oggetto e sulla sua forma:

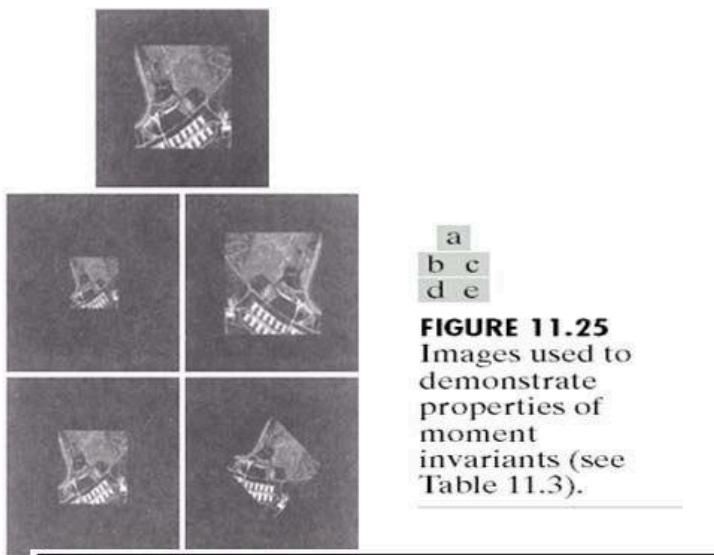
**Ellisse con assi a, b.** I valori di a, b sono invarianti per traslazione e **rotazione** in quanto dipendono dalla distribuzione spaziale della forma, ma non dalla sua posizione relativa agli assi dell'immagine. I momenti del secondo ordine definiscono un'approssimazione dell'oggetto definita *image ellipse*. **Questa è un'ellisse avente la stessa area, orientazione, eccentricità dell'oggetto ed è centrata in corrispondenza del suo centroide.**

$$M'_{mn} = \sum_{i=0}^{n_r-1} \sum_{j=0}^{n_c-1} (i - i_b)^m (j - j_c)^n F(i, j)$$

## Momenti di Hu



Hu ha mostrato che a partire dai **momenti centrali** e' possibile definire dei descrittori che risultano essere invarianti rispetto a traslazione, rotazione e scaling. Hu ha provato l'invarianza dei suoi descrittori rispetto a traslazione, **rotazione e scaling nel caso continuo**. Nell'applicazione pratica su immagini digitali queste grandezze si dimostrano generalmente ragionevolmente costanti per versioni modificate di una stessa forma. Le variazioni sono dovute agli inevitabili errori di discretizzazione.



**FIGURE 11.25**  
Images used to demonstrate properties of moment invariants (see Table 11.3).

Invariant (Log)	Original	Half Size	Mirrored	Rotated 2°	Rotated 45°
$\phi_1$	6.249	6.226	6.919	6.253	6.318
$\phi_2$	17.180	16.954	19.955	17.270	16.803
$\phi_3$	22.655	23.531	26.689	22.836	19.724
$\phi_4$	22.919	24.236	26.901	23.130	20.437
$\phi_5$	45.749	48.349	53.724	46.136	40.525
$\phi_6$	31.830	32.916	37.134	32.068	29.315
$\phi_7$	45.589	48.343	53.590	46.017	40.470

**TABLE 11.3**  
Moment invariants for the images in Figs. 11.25(a)-(e).

# Classificazione

## Progetto di applicazioni di classificazione

*Teoria:*

- Descrivere gli oggetti in termini di feature
- Specificare uno spazio di interpretazione
- Fornire un mapping (training set per legare un template al suo significato)

*Pratica:*

- Studiare la letteratura per vedere come sono affrontati problemi simili: trova feature e algoritmi utili
- Raccogliere una casistica di immagini che rappresentano l'oggetto da classificare
- Sperimentare

**Approcci:**

1. Template matching
2. Approccio statistico: estraggo descrittori che classificano tramite statistiche
3. Approccio strutturale (sintattico):
4. Reti neurali

## Approccio statistico

Definire le classi da impiegare

**Classificazione esclusiva:** un pattern appartiene a una sola classe: valuto la probabilità di appartenenza, e scelgo la classe *più probabile*

**Classificazione fuzzy:** assegnare più classi allo stesso oggetto

**Albero decisionale:**

Creo un albero decisionale caratterizzando ciascuna classe con dei descrittori

Viene costruito un albero che, per ciascun nodo, ha una regola di decisione che distingue i descrittori, al fine di ottenere la classe

*Problema: determinare le feature di un problema reale*

Parlare con il committente è fondamentale per acquisire le informazioni di dominio necessarie. Non tutte le feature, soprattutto se prese singolarmente, sono completamente discriminanti

-> devo **combinare le feature**

Inoltre, è necessario pesare le classi, se queste non hanno lo stesso valore, per minimizzare errori sulle classi *con più valore per la nostra attività*

-> Creo uno spazio delle caratteristiche (a più dimensioni) con i dati di training, per creare una soglia (**decision boundary**) per cui classificare

- L'overfitting può essere generato dalla mancanza di dati sufficienti

*Problema: numero di feature*

Usare le feature con cura: usare feature tra loro incompatibili, o che non separano correttamente i dati, o tra loro **correlate**, può creare problemi al classificatore

**Possibili combinazione per i dati:**

20% di test (*dati reali, da non usare per creare il classificatore ma per valutarlo*)

60% dati di training

20% dati per testare il training

## **Come procede la classificazione**

**Pre-elaborazione:** prima parte del corso, filtro per migliorare le immagini da riconoscere

### **Estrazione di feature:**

Uso feature appropriate per il punto successivo

### **Classificazione:**

Uso le feature per distinguere tra classi

### **Decisione:**

Prendo una decisione

### *Nota:*

Documentare come si giunge alle conclusioni (nel progetto e nel lavoro)

**Il metodo adottato per definire i parametri e ottenere i risultati è importante e va documentato**

# Classificatore

## Classificatore a minima distanza:

Calcolo la media di un vettore di feature dell'oggetto da classificare, e lo assegno alla classe il cui baricentro distante meno dal vettore. Si crea una petizione dello spazio in classi

*Problema: esistono oggetti che non rientrano nelle classi (unknown); dobbiamo avere una classe anche per i non-classificati*

## Classificatore a parallelepipedo

Creare dei *rettangoli* che raggruppino i dati, in modo che i *rettangoli* (classi) siano tra loro distinti, la regione non racchiusa in parallelepipedi sono unknown

## Classificatore bayesiano

Usa l'idea del classificatore di parallelepipedo ma con modello di gauss. Teorico, ideale, **ma** pone dei limiti alle statistiche e necessita di molti dati

## Classificatore NN

Dobbiamo normalizzare le feature per usarle come caratteristiche

## Classificatore KNN

Nearest neighbor classifier. Assegno a un vettore di feature la classe del vettore già classificato più vicino

-> su questa idea, posso estendere la dimensione del neighbor (con dati a sufficienza) assegnando al vettore la classe più diffusa negli N vettori più vicini

Con migliaia di dati il costo di ricerca nei dati è molto oneroso

## Nel nostro caso è ottimo

Nota: le feature devono essere normalizzate (la feature è un vettore, che può essere normalizzato dividendo per la distanza dall'origine)

*Errori di indecisione:* ci sono zone non assegnate a nessuna classe

## Support vector machine

**Idea:** mappare lo spazio delle feature in uno spazio vettoriale  $N + 1$ , che posso poi essere piegato per permettere una linea che divida le feature

In machine learning, support vector machines (SVMs, also support vector networks are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

## Classificatori pixel-based

Il pixel appartiene / non appartiene alla classe da classificare

Nota:

- se ci sono nel training set valori *troppo alti o troppo bassi* vengono "clippati", ridotti o aumentati per rientrare in un range
- Attenzione: l'aspetto può ingannare;
- Testare immagini diverse

Nel progetto, è importante mostrare i test fatti sui classificatori, rappresentando la correttezza tramite matrice di confusione

Spiegare perché fallisce, mostrando le immagini che non funzionano

## Semplificare

- Il colore è necessario? Altrimenti a livello di grigio
- Calcolo statistiche sull'immagine che mi rappresentano le informazioni
- Valutare Edge detection

## Matching

### Distanza di Hausdorff

Valutare la distanza di tutti i punti con tutti di template i punti dell'immagine (matching del contorno):

Problema: deve considerare tutti i punti del contorno

Estraggo il contorno e lo rappresento come un poligono in 1D (**semplifico da 2d a monodimensionale; quest'ultima può essere ulteriormente descritta**)

### Rappresentazione del dominio

Usiamo delle descrizioni che devono essere

- sufficiente (per il raggiungimento dello scopo)
- di ampia applicabilità (capace di descrivere molte classi di oggetti)
- non ambigua (due oggetti distinti non possono avere la medesima rappresentazione)
- unica (ogni oggetto distinto ha una univoca descrizione)
- stabile rispetto al rumore
- invariante per rotazione, traslazione e scala (se richiesto) (Es: area, perimetro, rapporto tra i due)
- generativa (è possibile ricostruire l'oggetto a partire dalla sua rappresentazione)
- conveniente (per gli scopi dell'applicazione)

Le segmentazioni non sono perfette -> la classificazione deve essere robusta e distinguere queste cose

La rappresentazione di una regione può essere basata su caratteristiche esterne (cioè del boundary), su caratteristiche interne (cioè legate ai pixel che la compongono), o su caratteristiche topologiche (cioè legate a caratteristiche che rimangono inalterate quando una figura viene deformata a piacere). In generale, una rappresentazione esterna è conveniente quando sono importanti le caratteristiche di forma della regione, mentre una rappresentazione interna più adeguata se si ritengono importanti caratteristiche della superficie, come il colore o la tessitura.

### Proprietà topologica

Lo scheletro è invariato

## Chain Code

Considero la possibilità di *muovermi* lungo il contorno solo in determinate direzioni, approssimando la bounding box dell'immagine di cui estraggo il contorno. Ogni direzione è rappresentata con un codice (1, 2, 3, 4 se uso 4-connessioni)

-> Ottengo una catena di codici che rappresenta il contorno

### Invarianza alla rotazione

Posso confrontare due chain code permutandone una, per verificare se i due contorni sono molto simili (ossia siano codificate in modo simile)

### Potenza del chain code

Chain code è una grande approssimazione, molto sensibile al rumore

### Approssimazione poligonale

Descrivo una forma come un poligono, prendendo:

- Parto sempre dai due punti più distanti ( o meglio, due punti estremi dell'asse di inerzia maggiore)
- Valuto il poligono
- Angolo dei vertici del poligono e numero di vertici -> prendo i segmenti che si incrociano e calcolo l'angolo

### Invarianze:

- Rotazione: prendo sempre i due punti più lontani
- Traslazione: prendo area, perimetro, ...
- Invarianza per scala: normalizzo per [0 ... 1]

### -> Signature

La signature è un vettore di valori che mantengono le invarianze e semplificano il poligono

Si estrae ponendo un centro del poligono, e valutando la distanza del centro dal contorno al rotare del raggio

-> posso confrontare due signature dalla distanza euclidea

### Insieme convesso

Insieme che rende il poligono è convesso. La differenza tra l'insieme e l'originale può essere usato per costruire diversi descrittori topologici

-> numero componenti connesse, numero buchi, numero insenature, ...

### Convessità e concavità

#### Concavo

Se almeno un angolo del poligono è  $> 180^\circ$  -> il poligono è concavo

Altrimenti, è **convesso**

Se esiste un segmento che unisce due vertici del poligono e che passa per il background

-> il poligono è **concavo**

# Descrizione di una regione

## Proiezione

La forma bidimensionale la proietto su x e y, creando due vettori. Passo da 2d a 1d, ottenendo due vettori tra loro confrontabili

E' molto utile per non fare il labeling delle componenti connesse, se si analizza un testo

Problema: riconoscere cifre, trovare e distinguere il testo dalle immagini

Distinguere i caratteri

## Frastagliamento

Il **Frastagliamento f** (a volte è chiamato compattezza C) stima il grado di irregolarità della forma considerata (assume valore 1 quando la regione è un circolare)

## Area

Numero di pixel appartenenti all'immagine

## Perimetro

Numero di pixel di contorno (4-connettori o 8-connettori).

Si sceglie le connessioni più coerenti

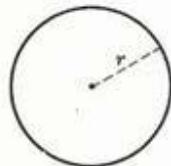
## Rapporto area/perimetro

Ottimo come descrittore che prende area e perimetro delle componenti connesse

*Dati cerchi rettangoli e quadrati di area causale e con "scala" diversa, individuare le forme*

## Form Factor

$$\begin{aligned} \text{Area} &= \pi r^2 \\ \text{Perimetro} &= 2\pi r \end{aligned}$$



$$C = \frac{4\pi \cdot \pi r^2}{4\pi^2 r^2} = 1 \quad (\text{costante})$$



$$C = \frac{4\pi \cdot l^2}{16 l^2} = \frac{\pi}{4} \cong 0.7854 \quad (\text{costante})$$

E' definita in base al rapporto area / perimetro; rappresenta il livello di "compattezza" della forma.  
Il cerchio è la massima compattezza; più ci si allontana

## Diametro

## Minimo rettangolo

## Asse di inerzia

Se i pixel sono una popolazione, l'asse di inerzia è la massima distanza tra una coppia di pixel e passa per il baricentro.

## Orientamento

Angolo tra la direzione del diametro e il verso positivo dell'asse orizzontale

## Numero di Eulero:

$E = N$  componenti connesse - numero di buchi

### Contare numero di buchi:

Conto regioni dello sfondo, rimuovendo le regioni che toccano il contorno

Su più caratteri:

- Prendo le componenti connesse e ne estratto la bounding box
- Proietta il livello di grigio sull'asse X (**proiezione**); affatto l'immagine verticalmente e trovo le regioni di gap bianchi > T (**trova le differenze**)

### Implementazione:

#### Bit Quads

Si confronta l'immagine con dei pattern  $2 \times 2$  che permettono di ottenere il numero di Eulero

## Momenti di inerzia

Volume (sommatoria del prodotto di pixel elevati alla  $(n, m)$  per il livello di grigio)

Funzionano per regioni tra loro speculari

**Ordine:** esponenti  $(n, m)$  ai pixel

Si indicano i momenti in base alla coppia  $(n, m)$

Alcuni:

$M_{00}$  = area

$M_{10}$  = baricentro rispetto a x

$M_{01}$  = baricentro rispetto a y

Invarianza:

- si calcola il momento rispetto al baricentro per evitare problemi sul punto di inizio
- Si divide per l'area ( $M_{00}$ )  $^{\wedge} (n + m / 2 + 1)$  per ottenere l'invarianza rispetto all'area
- Scala e rotazione, speculare

### Possono essere applicati anche a immagini in scala di grigio

Si ottengono i momenti di **Hu**

**7 Descrittori statistici che mantengono numerose invarianze, utili per matching tra immagini uguali ma con diversa scala, rotazione, ...**

*Template:*

- Si calcola la **ROI**: si segmenta per individuare un candidato
- Non si usano tutti i pixel, ma si estraggono i momenti e si usano come descrittori per matchare con un template dato

*Altri descrittori derivati dai momenti di inerzia:*

- **Ellisse di inerzia equivalente (fit ellipse)**
- **Orientamento**
- **Thickness**

Minimo rettangolo orientato:

- calcolo gli assi di inerzia e il loro orientamento
- Calcolo il rettangolo a partire da questi assi: trovo i 4 punti la cui distanza dagli assi è massima

## Trasformata distanza

Esempio:

Campo di erba secca; applica il fuoco ai bordi -> il fronte del fuoco si alza e si sposta alla stessa velocità verso il centro. I fronti si incontrano nei punti centrali

Posso usare la trasformata distanza per estrarre lo scheletro, e descrivere lo scheletro. Ad esempio, descrittori come:

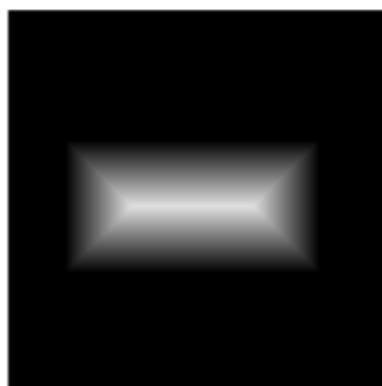
- numero joint
- Numero segmenti

Generativa:

A ogni pixel viene assegnato il raggio della sua distanza dal bordo. Quest'informazione può essere usata per rigenerare l'immagine dopo essere stata erosa



Immagine Originaria



DT

### Medial axis trasform

- Un punto p dell'oggetto appartiene all'asse mediano ("scheletro") se, detta d la distanza minima fra p ed il contorno della figura, esistono almeno due punti del contorno situati a distanza d da p
- La MAT e' definita nei punti appartenenti all'asse mediano ed il suo valore e' dato dalla distanza minima del punto dal contorno.
- Gli algoritmi piu' diffusi si basano sulla valutazione della distanza fra i punti interni ed il contorno della figura o lo sfondo. La definizione della MAT dipende ovviamente dal tipo di distanza utilizzata.

Risponde con forza in caso di difetti dell'oggetto, generando scheletri molto frastagliati  
-> può essere usata per individuare i difetti in controllo della produzione

OCR: riconoscimento delle lettere

- generare lo scheletro e usarlo come descrittore per riconoscere l'oggetto indipendentemente dalla dimensione del carattere

## Scheletrizzazione

Si impostano delle regole per riconoscere il contorno e l'oggetto, rimuovendo i pixel sino allo scheletro

Nota: possono generarsi punti di biforcazione.

Problema: come individuali ed elimino se  $< T$

Individuare: template matching. Lo scheletro ha spessore un pixel, posso contare i vicini (se  $> 3$ , biforcazione)

Trovo i punti di biforcazione, seguire i rami calcolando la distanza in T