

UN COMPENDIO DELLE LEZIONI DI
COMPUTER VISION SVOLTE IN UNIMORE
DALLA PROF RITA CUCCHIARA E DAL
PROF LORENZO BARALDI

*Il sequel di "Le fantastiche avventure nel machine
learning con Simone Calderara"*

COMPUTER

UNIMORE 2021

VISION

SCRITTI DA

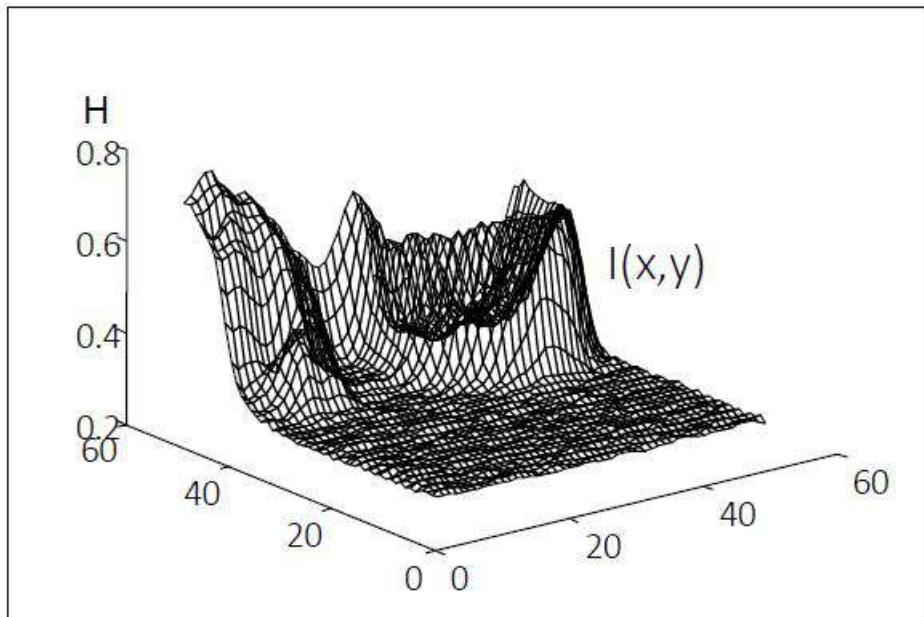
SEVERI-DE VITIS-VIGLIANISI

3. ABC OF IMAGES AND POINT OPERATIONS

Un'immagine può essere definita da due punti di vista differenti.

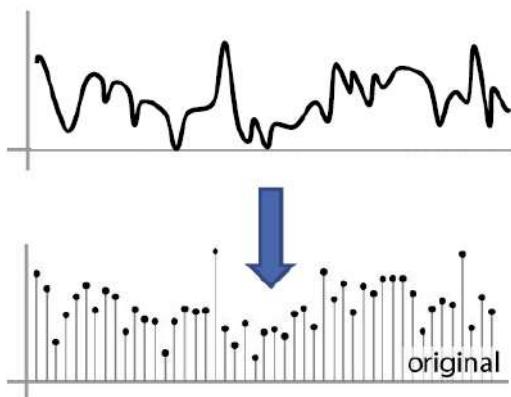
Dal primo punto di vista l'immagine è un segnale rappresentato dalla discretizzazione di una funzione continua in 2D $f(x,y)$. Da questo punto di vista l'elaborazione delle immagini è un'estensione in 2D alla teoria dei segnali definita in una sola dimensione. Si possono quindi elaborare immagini applicando ad esse tutte le elaborazioni definite dalla teoria dei segnali (es. trasformata di fourier). Il frutto che l'immagine possa essere vista come una quantizzazione di un segnale continuo in 2D è molto importante perché ci permette di applicare ad essa concetti tipici dei segnali.

In particolare l'immagine è un segnale e tale segnale può essere rappresentato sia da una funzione continua in due dimensioni che dal campionamento discreto di tale funzione.



$$I(\mathbf{x}) = f(x,y) : R^2 \rightarrow R$$

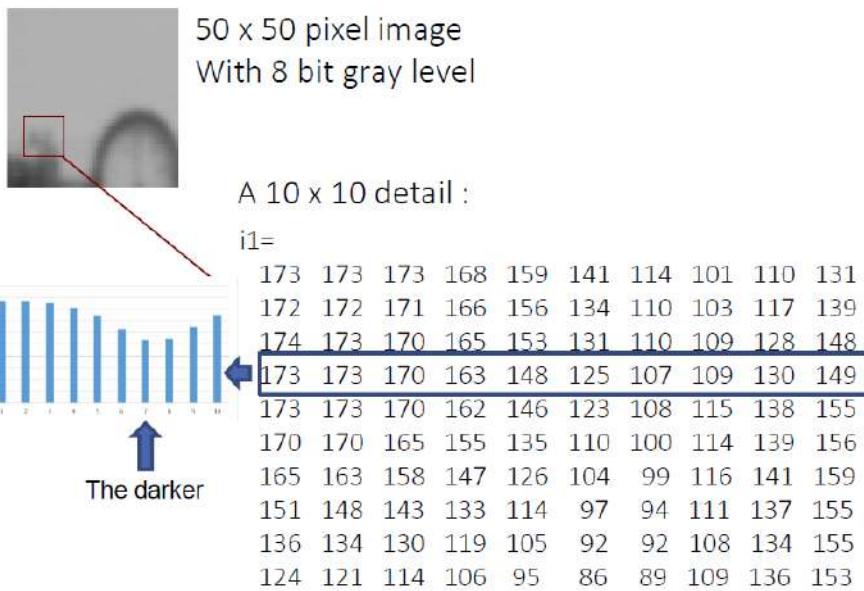
Tuttavia nella realtà l'immagine è definita soltanto in una porzione finita e quadrato di R^2 e i valori sono campionati e digitalizzati quindi in concreto l'immagine è data dalla rappresentazione discreta della funzione $f(x,y)$.



Se l'immagine è a colori, argomento che vedremo dopo, la funzione corrisponde alla seguente funzione vettoriale:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

Da un secondo punto di vista l'immagine è **una matrice 2D di pixels** digitali (numeri). Da questo punto di vista l'elaborazione delle immagini consiste nell'applicare ad esse funzioni matematiche, algoritmi o qualsiasi altro tipo di operazioni che lavorano con le matrici (algebra lineare, computer science,...).



In **computer vision** l'immagine è un **tensore 2D** talvolta usato come input, talvolta ottenuto come output. Questo significa che in computer vision esistono sia algoritmi che lavorano su immagini per estrarre informazioni e/o modelli, sia algoritmi che da informazioni/modelli costruiscono immagini.

Come detto prima i pixels sono valori numerici che rappresentano l'immagine. Il valore assunto dal pixel corrisponde ad un determinato livello di grigio o di colore dell'immagine in base a quale rappresentazione del colore viene scelta.

Le immagini rappresentate da livelli di grigio utilizzano una sola matrice (un solo canale) che contiene numeri da 0 a 255, mentre le immagini rappresentate nello spazio RGB utilizzano tre matrici (tre canali diversi) ognuna delle quali indica i livelli del proprio colore di riferimento. In particolare R esprime i livelli di rosso, G i livelli di verde e B i livelli di blu. Quindi un'immagine in bianco e nero è una matrice bidimensionale mentre un'immagine a colori è una tripla matrice bidimensionale, quindi avrà tre dimensioni (assi) invece che due.

0 → nero

255 → bianco

Oltre ad essere livelli di colore, i numeri all'interno della matrice potrebbero essere anche puntatori ad una paletta. In questo caso nello spazio RGB avremo una sola matrice di puntatori ad una paletta RGB di tre canali che rappresenta l'immagine.

256 values

A small block x=100..104, y=100..104

190	190	191	188	186
190	188	188	188	188
182	182	187	183	186
167	170	170	174	182
158	158	158	170	173



Link to RGB palette

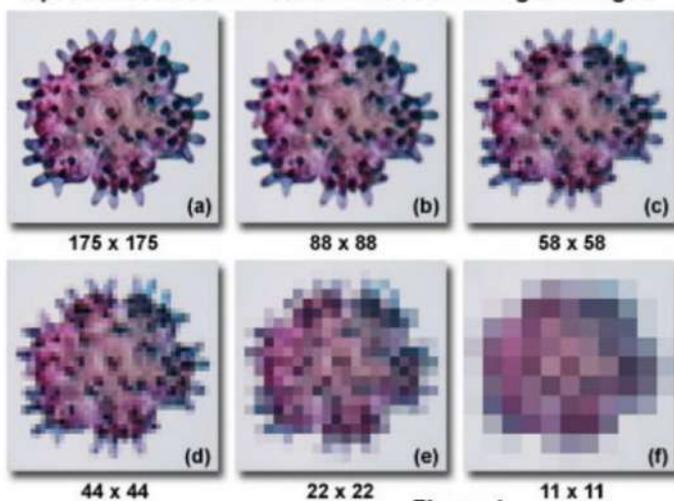
Palette:	186	1.0000	0.6118	0.3216	
	187	1.0000	0.5490	0.3529	
	188	1.0000	0.5686	0.4000	RGB value in [0.0,1.0]
	189	1.0000	0.6353	0.3255	
	190	1.0000	0.6118	0.4510	
	191	1.0000	0.6471	0.4196	

RGB value in [0.0,1.0]

Tuttavia le palette riducono notevolmente l'idea di continuità del segnale (che già è quantizzato) quindi non vengono usate in computer vision.

Un altro aspetto che si può considerare è la **risoluzione dell'immagine** o la risoluzione spaziale che corrisponde al numero di pixels lungo ogni asse. Specifichiamo che la risoluzione non è in alcun modo correlata con la dimensione dell'oggetto rappresentato.

Spatial Resolution Effect on Pixelation in Digital Images



Un altro aspetto che si può analizzare da un'immagine è l'intensità o la **luminosità (brightness)**, poiché ogni pixel è la misura dell'intensità luminosa del punto nello spazio 3D proiettato nello spazio 2D. tale luminosità dipende da diversi fattori quali la luminosità dell'oggetto, la luminosità dello sfondo, la capacità di riflettere dell'oggetto, il sensore utilizzato e così via.

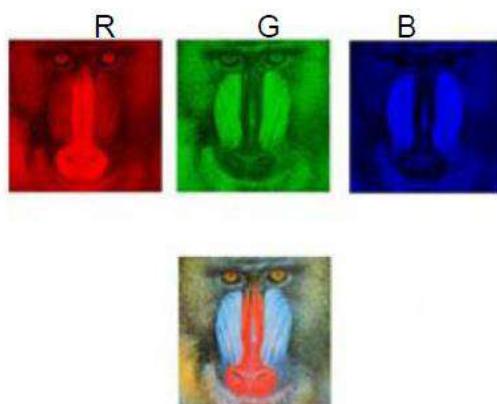
In particolare la **fotometria** è la scienza che studia la misurazione della luce, in termini di luminosità percepita all'occhio umano. Infatti la luminosità non dipende dalla lunghezza d'onda, bensì dalla distribuzione dell'energia dalla sorgente e dalla sensibilità dei sensori. Essa fornisce un significato fisico ai valori che compongono l'immagine. Le immagini monocromatiche rappresentano solo i valori di luminosità,

mentre le immagini multispettrali sono la combinazione di differenti spettri acquisiti ed ogni colore fa riferimento ad uno spettro differente. (in fisica, distribuzione in frequenza di una grandezza fisica variabile nel tempo, o in numeri d'onda di una grandezza fisica variabile nello spazio). In particolare di solito le bande incluse sono da 2 a 5 ottenendo un sistema ottico con larghezza di banda relativamente grande. Gli spettri combinati di solito sono:

- visibile (da 0,4 a 0,7 μm)
- vicino infrarosso (NIR; da 0,7 a 1 μm)
- infrarossi a onde corte (SWIR; da 1 a 1,7 μm)
- infrarossi a onde medie (MWIR; da 3,5 a 5 μm)
- o infrarossi a onde lunghe (LWIR; da 8 a 12 μm)

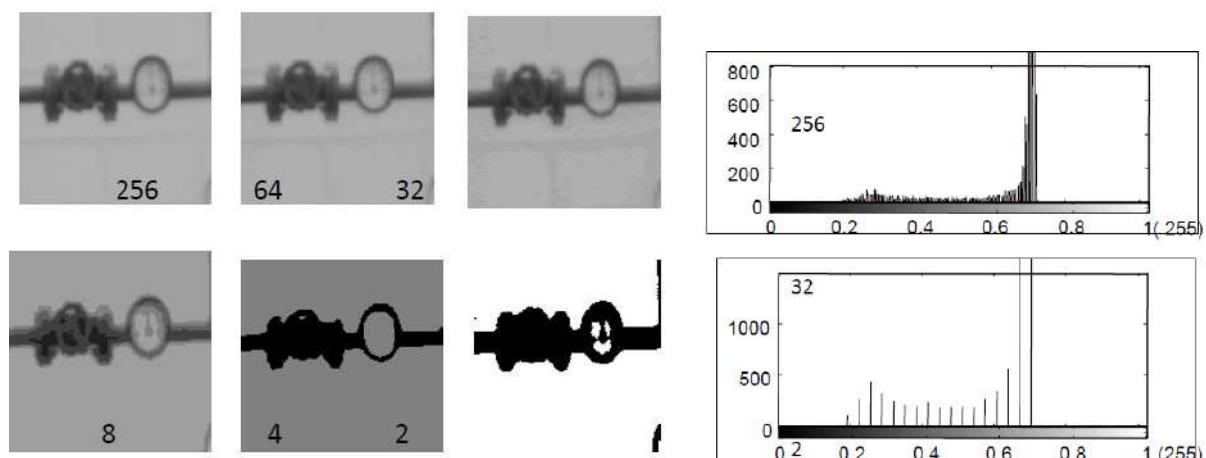
La colorimetria, invece, studia la lunghezza d'onda e l'emissione del colore.

Le immagini a colori sono strutture dati multi-array, cioè un insieme di matrici, una per ogni dimensione dello spazio dei colori. Ad esempio nello spazio RGB sono 3.



Invece le immagini grigie sono rappresentate da una sola matrice i cui valori sono dati dalla media dei tre valori corrispondenti nei tre canali RGB.

Il livello di grigio (o di colore) corrisponde al valore assunto dal pixel e ad ogni differente valore corrisponde un differente livello. Solitamente si utilizzano 256 livelli differenti ma all'occhio umano è sufficiente anche un numero inferiore.



L'operazione più semplice che può essere applicata su un'immagine è quella di ricavare il suo istogramma. L'**istogramma** è un vettore che contiene un numero di elementi pari al numero di livelli di grigio (o del

colore che stiamo considerando) dell'immagine (bins). Il valore di ciascuno di questi elementi corrisponde al numero di pixels dell'immagine che assumono quel livello di grigio, cioè il cui valore corrisponde al valore associato a quel bin. Se consideriamo 256 livelli i bin saranno 256. L'istogramma fornisce importanti informazioni per l'elaborazione dell'immagine, in particolare per il miglioramento del contrasto e per la segmentazione.

Algoritmo per costruire l'istogramma

```
for (n = 0; n < NLIV; n++)
    hist[n] = 0;
for (i = 0; i < NROW; i++)
    for (j = 0; j < NCOL; j++)
        hist [img [i][j]] ++;
```

Esistono librerie che forniscono funzioni già implementate per costruirlo, ad esempio:

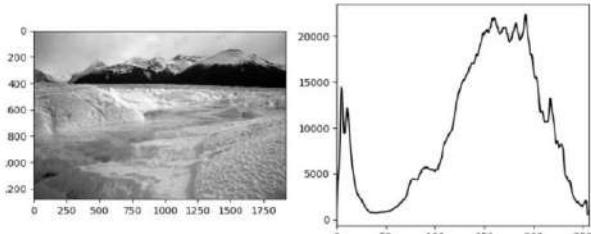
```
import cv2
import matplotlib.pyplot as plt

image = cv2.imread('dark-tones.jpg')

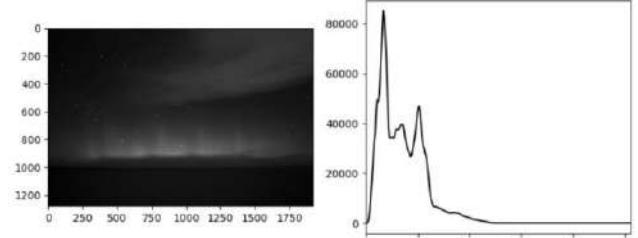
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

histogram = cv2.calcHist([gray_image], [0], None, [256], [0, 256])
plt.plot(histogram, color='k')
plt.show()
```

Sample image with light tones and histogram



Sample image with dark tones and histogram

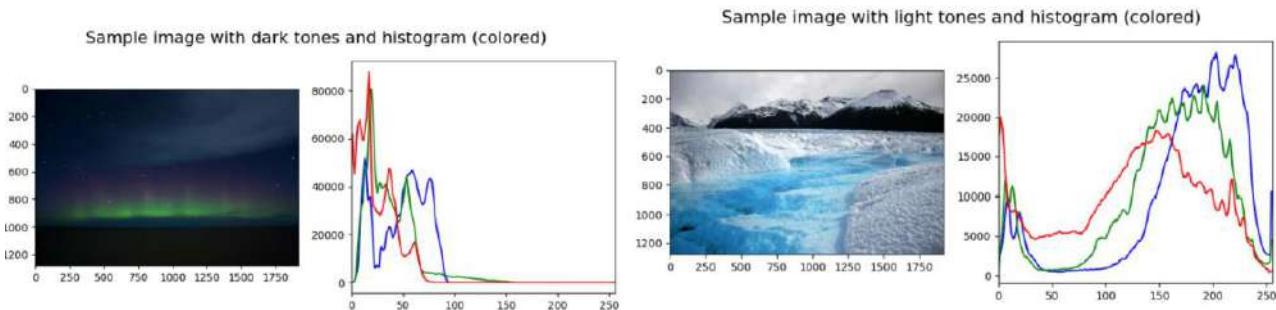


```
import cv2
import matplotlib.pyplot as plt

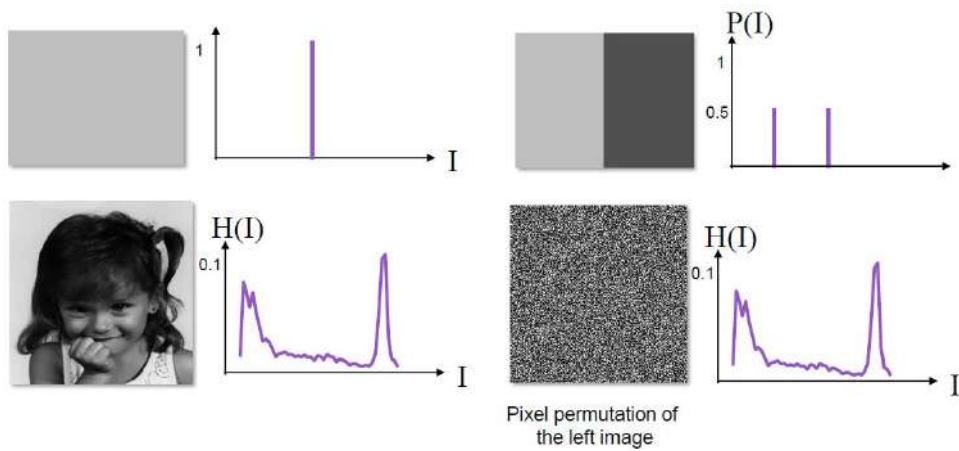
image = cv2.imread('dark-tones.jpg')

for i, col in enumerate(['b', 'g', 'r']):
    hist = cv2.calcHist([image], [i], None, [256], [0, 256])
    plt.plot(hist, color = col)
    plt.xlim([0, 256])

plt.show()
```



Il vettore che rappresenta l'istogramma può essere utilizzato come **feature vector** per la classificazione di immagini, anche se in realtà in molti casi non discrimina abbastanza perché non rappresenta completamente l'immagine. In questo caso avremo un global feature vector, cioè un vettore che considera tutti i pixel dell'immagine.



Se i dati (i pixels in questo caso) possono essere associati a variabili con distribuzione random i.i.d. indipendenti e identicamente distribuite, l'istogramma può essere visto come l'approssimazione discreta di una distribuzione di probabilità di una funzione di densità di probabilità.

(Ovvero $H(j)$, che poi normalizzando diventa $p(j)$, è la funzione di densità di probabilità mentre l'andamento di $H(j)$ per tutti i possibili j contenuti nell'intervallo considerato forma la distribuzione di probabilità.)

Considerando l'immagine $I = NxM$ con L livelli, posso definire l'istogramma nel seguente modo:

$$H(j) = \#\{x: I(x) = j\} \quad \# \text{ operator counts the number of elements}$$

Dove $x=(x,y)$

Per avere una distribuzione di probabilità, devo utilizzare la versione normalizzata dell'istogramma.

$$\sum_{j=0}^{L-1} p_I(j) = 1$$

L'istogramma normalizzato è a tutti gli effetti una distribuzione di probabilità e la probabilità che un generico pixel dell'immagine sia del livello i è data da ($p(i)$ è la funzione di densità di probabilità della distribuzione):

$$p(i) = \# \frac{\{x: I(x)=i\}}{n} \quad 0 \leq i < L$$

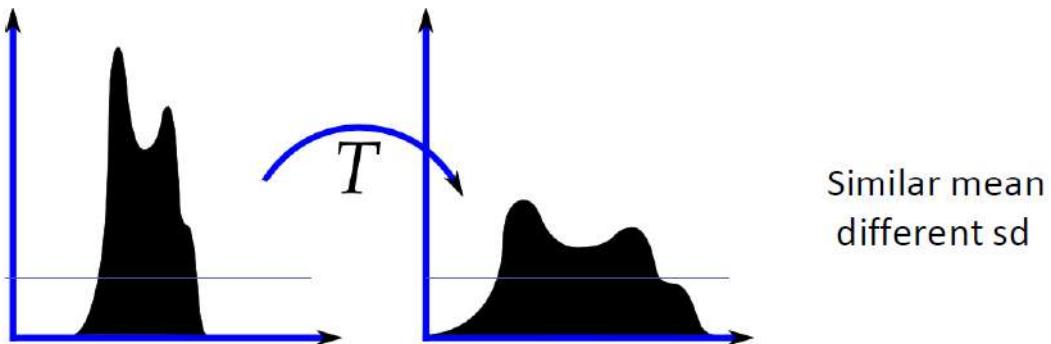
L being the total number of gray levels in the image, n being the total number of pixels in the image, and $p_x(i)$ being in fact the image's histogram for pixel value i , normalized to $[0,1]$.

Inoltre si può calcolare l'**istogramma cumulativo normalizzato** che altro non è che la funzione di distribuzione cumulativa **cdf** corrispondente a p_x . Essa restituisce la probabilità che il pixel x assuma valore inferiore al valore del livello i .

$$cdf_x(i) = \sum_{j=0}^i p_x(j)$$

Dal momento che l'istogramma è visto come una distribuzione di probabilità è possibile calcolarne media e deviazione standard:

$$\mu = \frac{1}{N} \sum_{j=0}^{J-1} j P_F(j) \quad \sigma^2 = \frac{1}{n-1} \sum_{j=0}^{L-1} (j - \mu)^2 P_F(j)$$

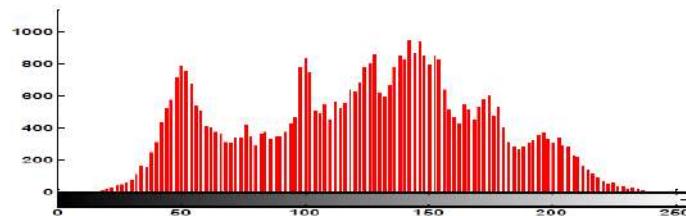


L'istogramma è fondamentale per misurare la distribuzione di una caratteristica dell'immagine come livello di grigio, livello di colore o motion, per verificare la mono-modularità per la segmentazione e per implementare tools per l'elaborazione come l'equalizzazione.

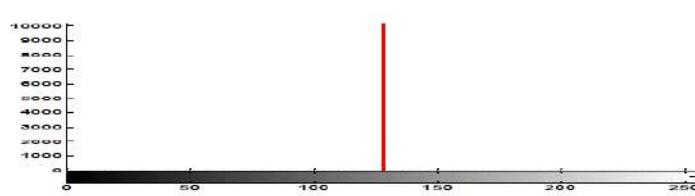
Inoltre l'istogramma è importante per calcolare l'entropia dell'immagine. L'entropia di un'immagine specifica l'incertezza nei valori dell'immagine, cioè misura l'ammontare medio di informazione richiesta per codificare i valori dell'immagine. Un evento raro fornisce più informazione rispetto ad un evento frequente.

L'entropia è la misura della dispersione dell'istogramma. Più c'è dispersione nell'istogramma più l'entropia è alta e più l'entropia è altra più è alta l'informazione fornita.

$$Entropy(I) = -\sum_k P(k) \log P(k)$$

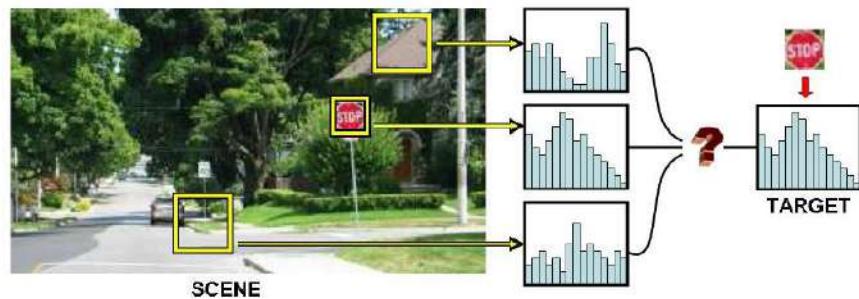


entropy=7.4635

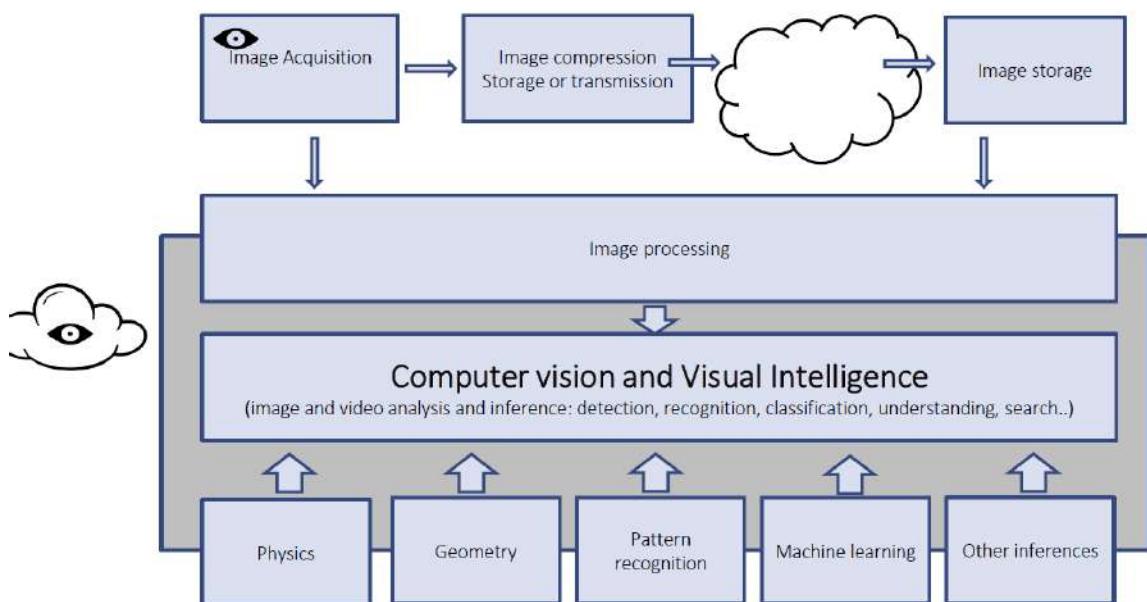


entropy=0

L'**istogramma adattivo** è l'istogramma di una porzione dell'immagine. In certi casi è utile considerare solo una porzione particolarmente rilevante dell'immagine invece dell'immagine intera.



Panoramica di processo di computer vision



L' **image processing** è la scienza che studia come elaborare automaticamente le immagini con un calcolatore per produrre altre immagini oppure per modificarle o migliorarne alcune proprietà. Sostanzialmente si occupa di creare nuove immagini elaborando immagini esistenti.

Quello che distingue la computer vision (visione artificiale) dal campo già esistente dell'elaborazione digitale di immagini è che in visione artificiale si cerca di ricostruire la struttura tridimensionale del mondo dalle immagini in modo da poter usare tale struttura come base per capire l'intera scena in cui ci troviamo.

L' image processing è un primo passo per trasformare le immagini in altre immagini su cui l'informazione e il processo di estrazione della conoscenza richiesta dal problema risultano più facili.

L' image processing è il task di base non sono nella computer vision ma anche per altri task ad essa associati come compressione, comunicazione, imaging, ecc..

Quando facciamo image processing e lavoriamo a livello di pixels, spesso si utilizza la **saturated arithmetic** che è un'aritmetica in cui gli unici valori esistenti sono gli interi compresi fra 0 e un numero massimo che solitamente è 255:

$$\begin{aligned} I'(x) &= 0 \quad \text{if } s I(x) + k < 0 \\ I'(x) &= \text{maxrange} \quad \text{if } s I(x) + k > \text{maxrange} \end{aligned}$$

Spesso è usata quando il risultato dell' image processing è un'altra immagine visibile dall'occhio umano.

Andiamo ora a vedere quali sono gli operatori utilizzati sui pixels nell' image processing. Esistono tre differenti tipologie di operatori:

- **Point operators:** il valore dei ogni pixel della nuova immagine dipende solo dal valore del pixel nella stessa posizione dell'immagine di partenza (ad esempio soglia). Con questi operatori si può ottenere il massimo parallelismo nell'elaborazione poiché ogni pixel viene elaborato indipendentemente dagli altri.

$$I'(x, y) = h(I(x, y)) \quad \text{or in vectorial form } I'(x) = h(\mathbf{I}(x))$$

Also for videos: $I'_t(x) = h(I_t(x))$

- **Local (neighborhood) operations:** il valore di ogni pixel della nuova immagine dipende dal valore del pixel nella corrispondente posizione nell'immagine di partenza e dai valori dei pixels nel suo intorno (ad esempio filtering). Con questi operatori si può avere un parallelismo medio.

$$I'(x, y) = h(I(x, y), \mathbf{x}(I(x, y)))$$

- **Global operators:** il valore di ogni pixel della nuova immagine dipende dal valore di tutti i pixels dell'immagine di partenza (ad esempio trasformata di fourier). Con questi operatori il parallelismo nell'elaborazione è nullo.

$$I'(x, y) = h(\iint I(x, y))$$

LINEAR POINT OPERATOR O OPERATORE LINEARE

Può essere applicato su una o più immagini:

$$g(x) = h(f(x)) \quad \text{or} \quad g(x) = h(f_0(x), f_1(x), \dots, f_n(x),)$$

X è il dominio D-dimensionale della funzione, che per le immagini vale 2 e rappresenta le coordinate x e y nel piano 2D del pixel che stiamo elaborando oppure le coordinate i,j dell'immagine, mentre $h()$ è l'operatore che trasforma l'immagine in un'altra immagine. Se la trasformazione $h()$ è **lineare** sarà della forma:

$$g(x) = h(f(x)) = s f(x) + k$$

$$I'(x) = h(I(x)) = s I(x) + k$$

dove s è lo **scale factor**, chiamato anche guadagno o gain o contrasto, k è la **costante di offset** chiamata anche bias o brightness. A volte s e k possono dipendere dalla posizione quindi saranno $s(x)$ e $k(x)$. Essendo un operatore lineare, per esso vale il **principio di sovrapposizione**:

$$h(f_0 + f_1) = h(f_0) + h(f_1)$$

Applicando l'operatore lineare e settando correttamente s e k si possono eseguire diverse operazioni come la variazione di luminosità, cambiando in modo proporzionale il valore dei pixels (es: +10% di luminosità -> $s=1.1$ e $k=0$) oppure fare il negativo ($s=-1$ e $k=255$).

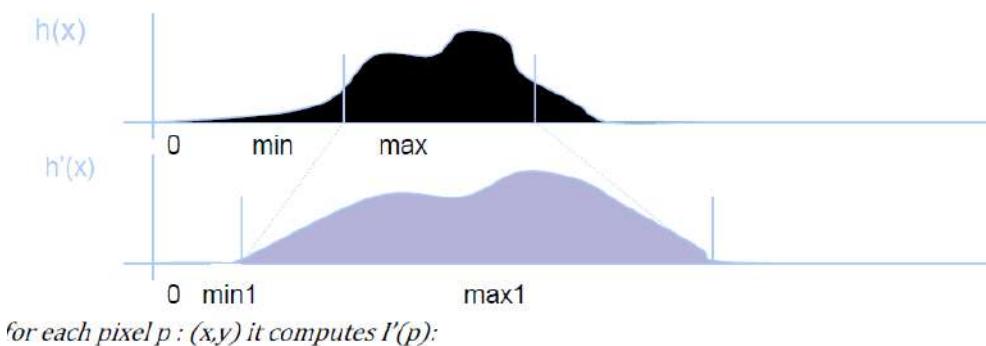
LINEAR BLEND OR DYADIC OPERATOR

E' un operatore lineare a livello di pixels che combina fra loro due immagini, facendo la loro sovrapposizione. E' usato solitamente nell'elaborazione di video per creare effetti visivi come una transizione lineare o una dissolvenza incrociata.

$$g(x) = h(f_0(x), f_1(x)) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$

CONTRAST STRETCHING

Dato un istogramma, il contrast stretching è l'espansione dei livelli di grigio o di colore dei pixels in un range dinamico. Espande quindi l'istogramma generando un nuovo intervallo di livelli cambiando anche il valore dei pixels, quindi produce una nuova immagine.



È sempre un operatore lineare e l'**algoritmo** è il seguente:

```

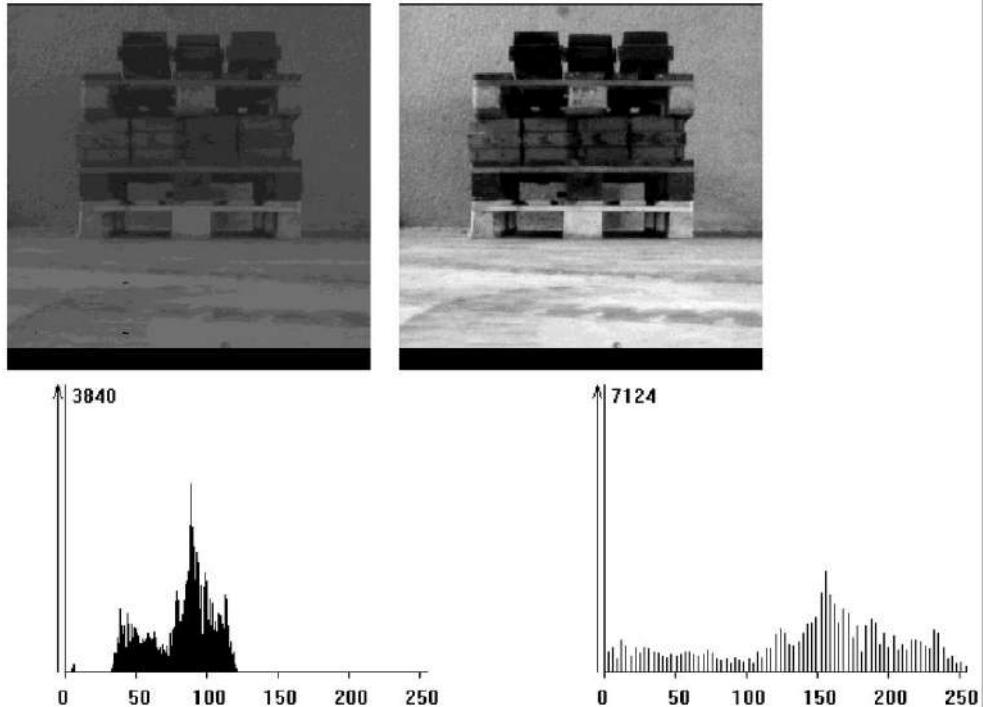
ScaleFactor= (max1-min1) / (max-min)

if (I(p)<=min)    I'(p)=min1;
if (I(p)>=max)    I'(p)= max1;
if ((I(p)>min) & (I(p)<max) )
    I'(p)= (I(p)-min) * ScaleFactor +min1;

```

$$\text{always } I'(p) = h(I(p)) = s I(p) + k$$

Esempio:



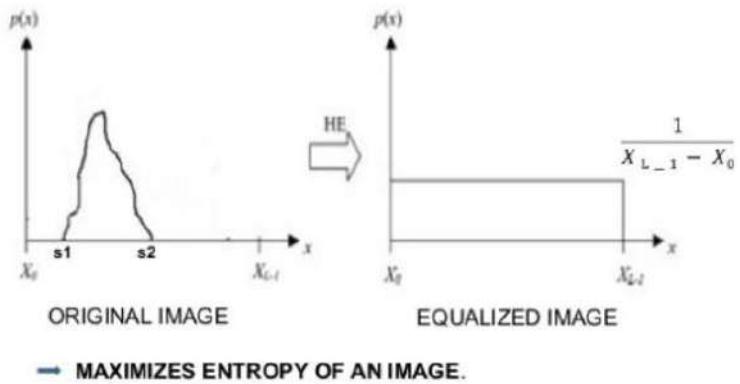
Bisogna specificare che questo operatore non cambia la distribuzione di probabilità definita dall'istogramma, allarga solo l'intervallo.

EQUALIZATION (non richiesto all'esame)

Questo operatore, invece, cambia la distribuzione di probabilità data dall'istogramma. Cambia l'istogramma per creare un'immagine migliore per la perception. Cerca di creare immagini in cui per ogni livello c'è circa lo stesso numero di pixels che ha quel valore, cerca di ricostruire una distribuzione uniforme. Non si usa nell'image processing perché introduce una modifica NON LINEARE nei dati.

L'equalizzazione, oltre a cambiare l'informazione, **massimizza l'entropia**.

$$S_k = (L - 1)cdf(x)$$

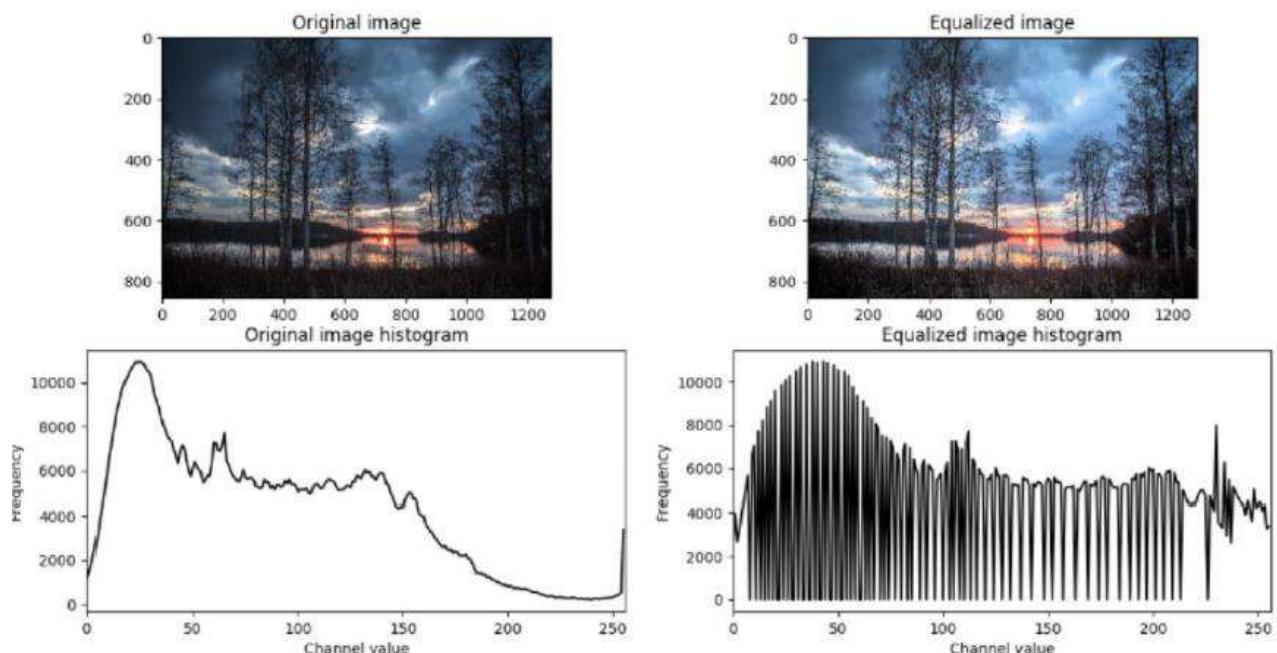


Considering $t(l)$ the cumulative h.
L gray levels, $M \times N$ Image size

$$E(l) = \max(o, \text{round}((\frac{L}{N * M}) * t(l)) - 1)$$

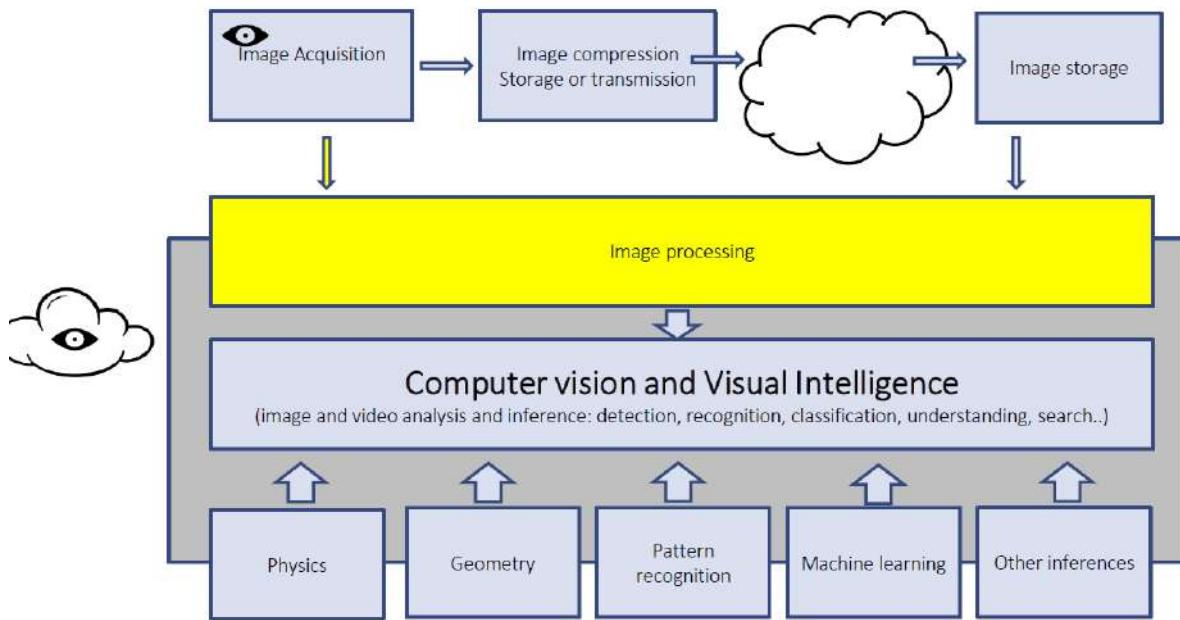
L'equalizzazione ha come obiettivo quello di pareggiare l'istogramma, modificando i valori dei pixels.

Comparison between original and equalized image on HSV



4. FILTERING

L'operatore di filtering è un neighborhood operator ed è sempre un'elaborazione dell'image processing.



Il filtering è quell'operatore dell'image processing che si occupa di eliminare il rumore o noise.

Per rumore si intende tutto ciò che non è segnale o tutto ciò che non è un'informazione utile.

Nel caso specifico delle immagini, che sono fortemente soggette a rumore, il **noise** è una variazione casuale della luminosità o del colore, ovvero una **variazione casuale del valore di pixels sparsi nell'immagine**. Può essere causato da diversi fattori come dai sensori, dalla trasmissione, ecc.

Un operatore puntuale lavora su un pixel alla volta quindi non può sapere se quel pixel è soggetto a rumore o meno se non lo confronta con quelli intorno. Quindi non è possibile eliminare il rumore o noise con un operatore puntuale, bisogna usare un neighborhood operator.

Possiamo definire tre diversi tipi di noise:

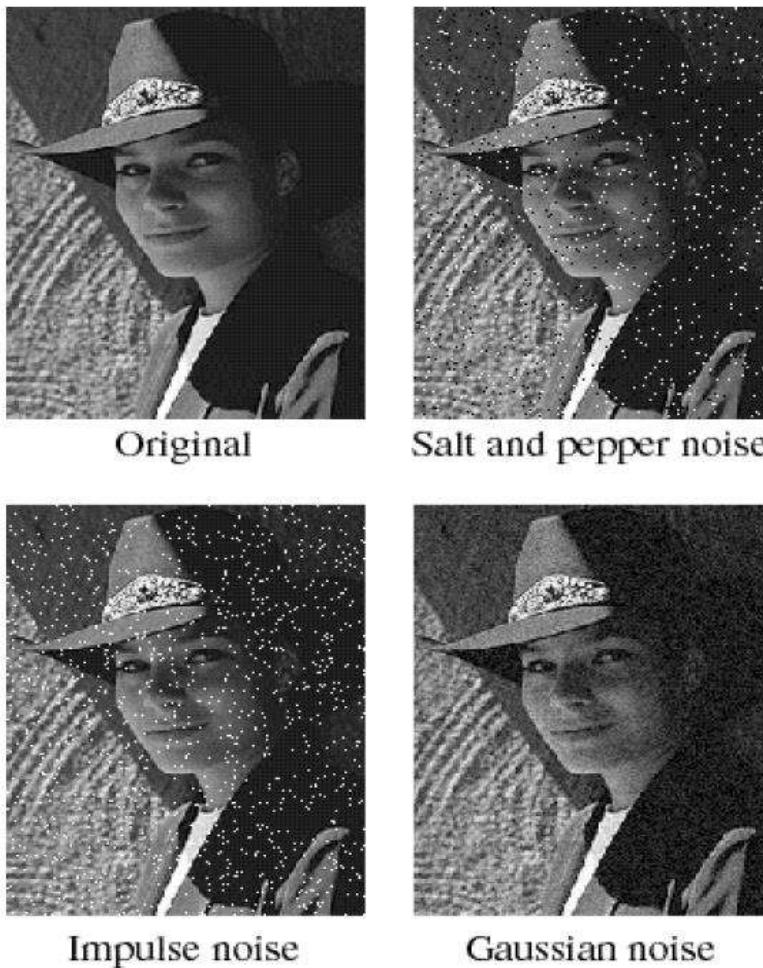
- **Signal noise:** come appena descritto, rumore dovuto alle operazioni di campionamento del segnale che va a modificarlo ed è solitamente additivo. Esso può essere misurato dal signal to noise ratio (SNR) che mette in relazione la potenza del segnale utile con quella del rumore in un qualunque sistema di acquisizione ed è calcolato come il rapporto tra la deviazione standard del segnale e la deviazione standard del rumore.

$$l'(i,j) = l(i,j) + n(i,j) \quad SNR = \sigma_s / \sigma_n$$

- **Computational noise:** è generato da errori nella computazione dei pixel dovuti per lo più ad approssimazioni o a limiti computazionali. Ad esempio la compressione produce computational noise
- **Perceptual o semantic noise:** è tutto ciò che è presente nell'immagine ma disturba la comprensione del target perché non ne fa parte. Sono fattori che disturbano la comprensione di quello che l'immagine raffigura.

Prima di eseguire i task di computer vision il signal noise deve essere eliminato. Esistono diversi tipi di signal noise da dover eliminare prima di proseguire con altri task, come:

- **Salt and pepper noise:** questo tipo di rumore corrisponde a pixel bianchi e pixels neri che in modo random appaiono nell'immagine.
- **Impulse noise:** questo tipo di errore corrisponde a pixels bianchi che in modo random appaiono nell'immagine.
- **Gaussian noise:** questo errore corrisponde a delle variazioni dell'intensità dell'immagine. Tali variazioni seguono una distribuzione Gaussiana.



In generale, a prescindere dalla classificazione vista sopra, si suppone che il rumore presente nelle immagini sia bianco. Dal punto di vista fisico un **white noise** è un segnale (o processo) caratterizzato dall'assenza di periodicità nel tempo e da ampiezza costante su tutto lo spettro di frequenze, cioè ha **densità spettrale di potenza piatta**. Questo significa che uguale potenza per ognuna delle frequenze all'interno di una data banda. È chiamato Bianco perché lavora come una luce bianca la cui densità spettrale di potenza è distribuita sulla banda visibile dai recettori dei tre colori degli occhi i quali sono approssimativamente e equamente stimolati da tale luce.

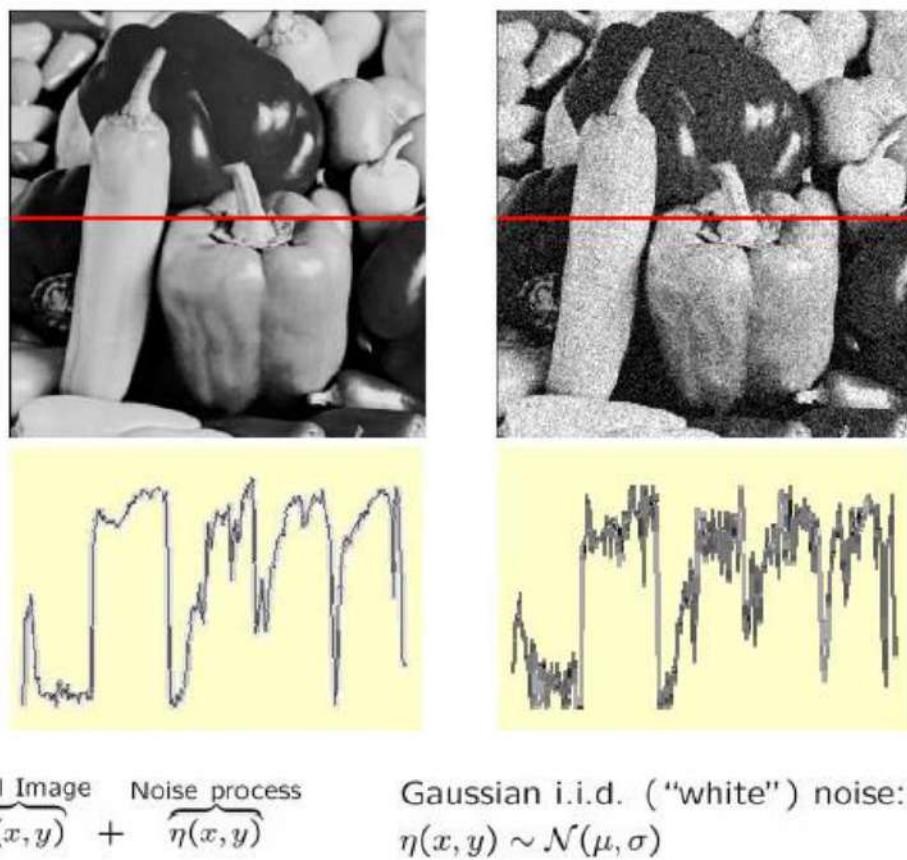
Mentre dal punto di vista statistico una è serie temporale $\{r_i\}$ (nel caso delle immagini è una serie spaziale, non temporale) formata da una **sequenza di variabili casuali non correlate fra loro con media zero e varianza finita**. Nella pratica significa che il white noise produce per ogni pixel una variazione additiva casuale e considerando tutte queste variazioni possiamo dire che sono indipendenti l'una dall'altra (no correlazione) hanno media zero e varianza finita.

Infatti il white noise ha infatti la caratteristica di essere **i.i.d.** indipendentemente e identicamente distribuito il che implica che non ci sia correlazione fra le variabili.

In particolare se la serie r_t che rappresenta il rumore bianco segue la distribuzione normale con media 0 e deviazione standard σ tale serie è chiamata **Gaussian white noise**.

Ciò che abbiamo visto fin ora, sia dal punto di vista fisico che statistico, faceva riferimento a segnali definiti nel dominio delle frequenze temporali ma vale anche nel dominio delle frequenze spaziali. Specifichiamo che noi stiamo lavorando con immagini quindi consideriamo lo spettro nel dominio spaziale e la serie che rappresenta il white noise sarà una serie spaziale in x e y .

Osserviamo il seguente esempio considerando il segnale lungo l'asse x : è presente un white noise additivo.



Se non sono specificate altre informazioni supponiamo che il rumore presente sull'immagine sia **additivo e gaussiano**, ovvero un rumore bianco che segue una distribuzione normale, con media 0. Il noise è quindi un valore stocastico aggiunto al valore di ogni pixel, non correlato in alcun modo con il segnale (l'immagine) e con i rumori aggiunti agli altri pixels. Il rumore è sempre presente nelle immagini, specialmente quando si utilizzano fotocamere a basso costo.

Altre volte il rumore può essere **impulsivo**, come nel caso salt and pepper, e quindi può modificare in modo casuale il valore di alcuni pixels.

Altre volte ancora il rumore può essere **moltiplicativo**, di solito nelle immagini mediche, oppure **speckle**, cioè additivo ma proporzionale al valore del pixel.

$$I'(x, y) = I(x, y) + nI(x, y) + no \quad (n \text{ is the noise})$$

speckle noise

Il task di **image restoration**, facente sempre parte dell'image processing, è l'elaborazione che ha l'obiettivo di migliorare la qualità dell'immagine eliminando il rumore ed eventuali artefatti. Le tecniche utilizzate per ridurre il rumore sono: smoothing, filtering.

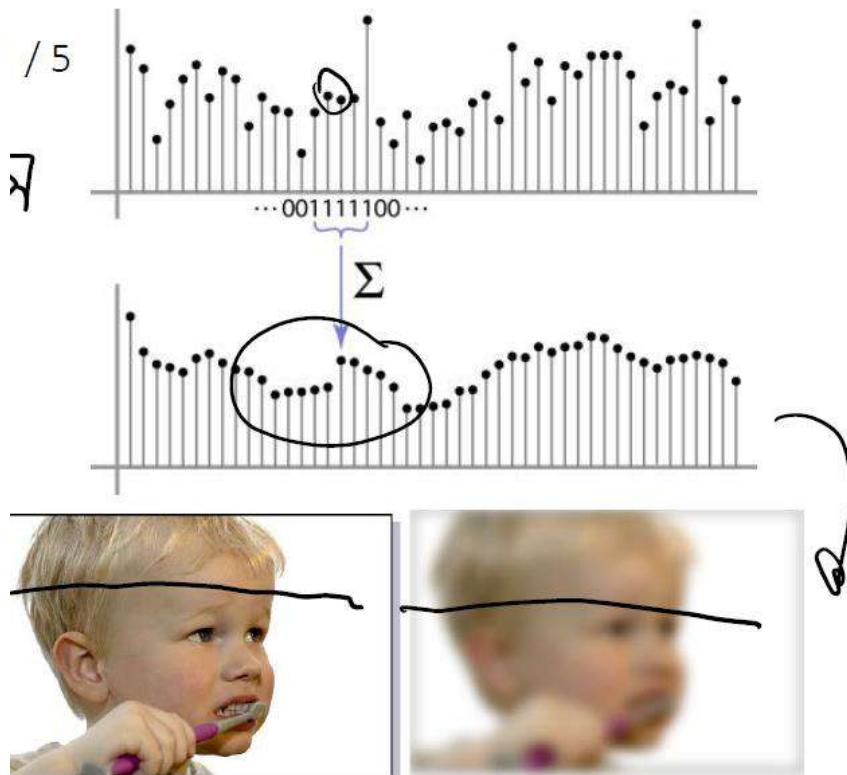
SMOOTHING

E' una tecnica usata per ridurre il rumore che consiste nell'effettuare un processo di smoothing (levigatura) dei valori dei pixels, utilizzando la media. Il valore assunto da ogni pixel viene sostituito con il valore medio fra i valori dei pixel nell' intorno e il valore di quel pixel. (neighborhood operation). E' possibile fare anche una media pesata, dando più importanza a determinati pixels piuttosto che ad altri.

$$g(x, y) = t(f(x, y))$$

La funzione t è la media dei valori dell'intorno di (x,y) . Per ogni pixel i pixel nel suo intorno cambiano, di conseguenza cambierà anche la media di quei pixels. In realtà anche la tecnica di smoothing rientra dentro alle tecniche di **linear filtering**. In questo caso il filtro utilizzato il **filtro media**.

(quello nell'immagine non è un istogramma, sono i valori dei pixels in una linea)



Lo smoothing sfuoca l'immagine perché non si sta facendo un'operazione di levigatura solo sul rumore ma anche sul segnale quindi sull'immagine.

LINEAR FILTERING

Data un'immagine iniziale F l'operazione di linear filtering consiste in un processo che da come output una nuova immagine G in cui ogni pixel è dato dalla **somma pesata dei valori del pixel in posizione corrispondente e dei pixels nell'intorno di esso nell'immagine iniziale**, usando lo stesso set di pesi ogni volta.

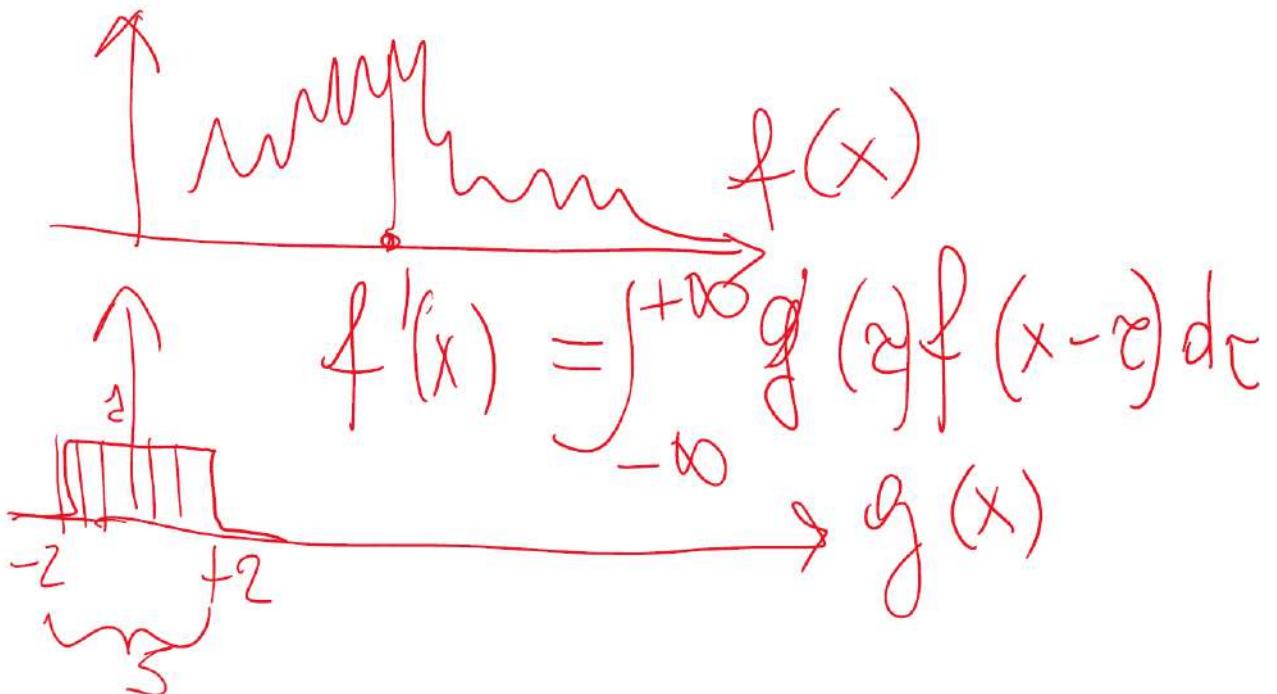
Tale operazione è **shift-invariant**, ovvero il valore del nuovo pixel dipende dal pattern contenuto nei pixels nel vicinato, non dalla loro posizione (la posizione del pixel che stiamo filtrando non influisce in alcun modo), e **lineare**, cioè l'output ottenuto per la somma di due immagini è uguale alla somma degli output calcolati indipendentemente sulle due immagini.

La configurazione dei pesi utilizzata per l'operazione di linear filtering è chiamata **kernel** del filtro mentre il processo di applicazione del filtro è solitamente chiamata **correlazione o convoluzione**.

Dopo aver ripassato la teoria sulla convoluzione

(https://biomedia4n6.uniroma3.it/teaching/tds_el/materiale_didattico/TeoriaDeiSegnali.pdf) andiamo a vedere come funziona nel nostro caso specifico. Ricordiamo che il nostro dominio è lo spazio, non il tempo t.

Applicando la definizione di **convoluzione** nel nostro caso specifico avremo $f(x)$ che in questo caso è una linea dell'immagine, $g(x)$ è il nostro filtro che in questo caso considera 2 pixel a destra e 2 pixel a sinistra del pixel in posizione x su cui stiamo facendo la convoluzione, per un totale di 5 pixels.



Essendo noi nel campo discreto non utilizzeremo l'integrale bensì useremo la seguente formula: (questa formula è scritta al contrario, in questo caso f è il filtro e g una riga dell'immagine)

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m].$$

tuttavia noi ci troviamo in uno spazio simmetrico quindi andare in una direzione o andare nella direzione opposta è la stessa cosa quindi utilizzare il segno meno o il segno più produce lo stesso risultato. Se utilizziamo il segno + non stiamo più facendo la convoluzione ma stiamo facendo la **correlazione discreta**.

Possiamo concludere dicendo che dato un segnale $I(x)$ l'applicazione di un filtro F che è definito sui valori di x compresi fra $-N$ e N si effettua facendo il **Dot- product** o la **cross correlation**:

$$F \circ I(x) = \sum_{i=-N}^N F(i)I(x+i)$$

E in due dimensioni (sull'intera immagine):

$$F \circ I(x, y) = \sum_{j=-N}^N \sum_{i=-N}^N F(i, j)I(x+i, y+j)$$

Andando ancora più nello specifico della nostra applicazione, applichiamo all'immagine $f(i,j)$ un filtro lineare che produce in output un'immagine $g(i,j)$ i cui pixel sono dati dalla somma pesata dei pixels dell'immagine iniziale $f(i,j)$ e i pesi sono forniti dal kernel o dalla maschera H che ha $h(k,l)$ come coefficienti.

$$g(i, j) = \sum_{k, l} f(i+k, j+l)h(k, l)$$

CORRELAZIONE.

$$g(i, j) = \sum_{k, l} f(i-k, j-l)h(k, l) = \sum_{k, l} f(k, l)h(i-k, j-l)$$

CONVOLUZIONE

Image is a 2D signal. The 2D convolution come from the theory of continuos signals in 2 D and is a very common operator for linear filters

$$g(i, j) = f * h = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(i-m, j-n)h(m, n)dm dn$$

$$g(i, j) = f * h = \sum_m \sum_n f(i-m, j-n)h(m, n)$$

If the Image is actually a sampled signal, all the conclusions of filtering proprieties coming from signal processing can be assumed for image processing too.

If the hypothesis is not true (that is the image is not a correct linear sampling of a continuous 2D function), then also the proprieties of convolution are not always true...

I filtri convolutivi, in quanto filtri lineari, godono delle seguenti proprietà:

-**linearity**: $h^\circ(f_0 + f_1) = h^\circ f_0 + h^\circ f_1$

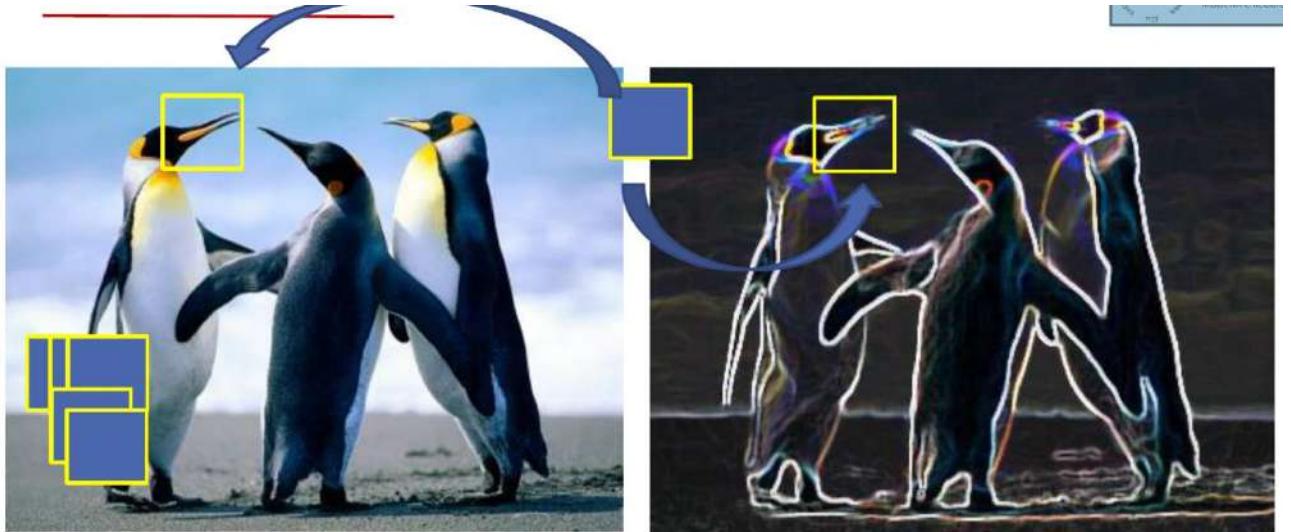
-**scalarity**: $h^\circ(kf) = kh^\circ f$

-**shift invariance**: the answer to a shifted stimulus is the shift of the answer to the stimulus

$$g(i, j) = f(i+k, j+l) \Leftrightarrow (h^\circ g)(i, j) = (h^\circ f)(i+k, j+l)$$

Come già detto prima, il valore dell'output dipende solamente dal valore di partenza del pixel ed eventualmente dai valori nel suo intorno, non dalla posizione. Nei sistemi lineari in cui vale la proprietà di shift-invariance si può applicare la convoluzione.

La convoluzione è infatti un'operazione locale, che agisce su una finestra dell'immagine di dimensione pari a quella del kernel.



Algoritmo che fa la convoluzione (applica all'immagine un filtro lineare)

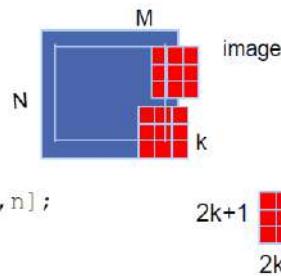
Given the image $f(i,j)$ $i=1..N, j=1..M$, and a mask or ray k ,

```

h(m,n) m=-k..+k, n=-k..+k

for (i=1+k; i<=N-k, i++)
    for (j=1+k; j<=M-k, j++)
    {
        g[i,j]=0;
        for (m=-k; m<=k, i++)
            for (n=-k; n<=k, n++)
                g[i,j]=g[i,j]+f[i-m,j-n]*h[m,n];
    }
}

```



45	60	96	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

$h(x,y)$

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	106	120	129
57	69	83	96	112	124
53	60	71	85	100	114

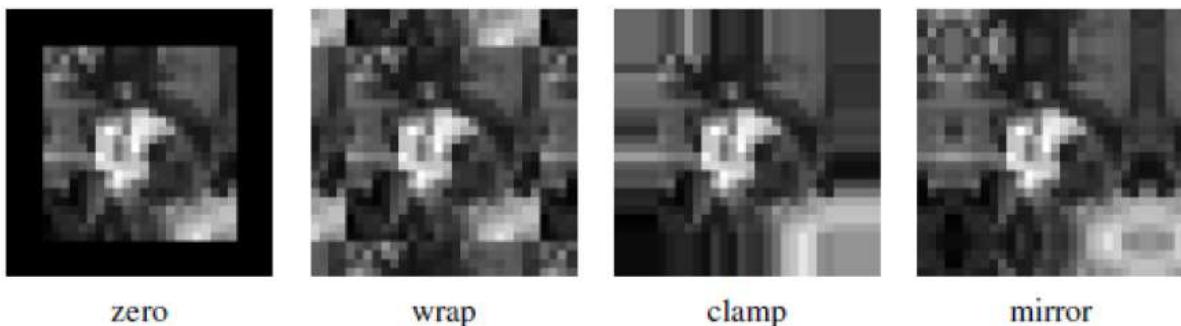
$g(x,y)$

In generale, poiché il filtro è invarianti allo spostamento (shift-invariant), usiamo per tutte le porzioni dell'immagine lo stesso kernel con gli stessi valori e la stessa dimensione. Questo causa dei problemi quando la convoluzione è applicata ai pixels sui bordi. Per risolvere questo problema si effettua il padding,

ovvero si crea un bordo di pixels aggiunti all'immagine originale. Tali pixels possono assumere diversi valori a seconda del tipo di padding usato:

- **Zero padding**: insert 0 pixels
- **Constant padding** insert a specific color in the border
- **Clamp to edge**, repeat the edge value
- **Wrap** : loop around in a toroidal configuration
- **Mirror**: reflect the edge

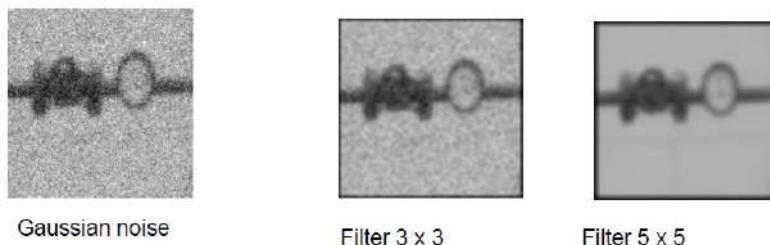
Etc..



Come detto prima, l'operazione di smoothing viene anch'essa eseguita applicando un filtro. Lo smoothing o la sfuocatura (bluriness) viene effettuata applicando un filtro media chiamato **moving average filter**. Invece di sostituire il valore del pixel che stiamo considerando con la media dei valori dei pixels nel suo vicinato, facciamo la convoluzione utilizzando un kernel avente in tutte le posizioni lo stesso valore pari a 1 diviso il numero totale di pixels dell'intorno che andiamo a considerare.

$$\begin{array}{|ccc|} \hline & & \\ \left| \begin{array}{ccc} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{array} \right| & & 3 \times 3 \text{ kernel} \\ \hline \end{array}$$

Lo smoothing non funziona quando l'errore da correggere è di tipo salt and pepper.



Quindi il moving avarage filter genera sfuocatura e levigatura (smoothing and blurriness).

Vediamo il seguente esempio in 2D:

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

0	10	20	30	30	30	20	10		
0	20	40	60	60	60	40	20		
0	30	60	90	90	90	60	30		
0	30	50	80	80	90	60	30		
0	30	50	80	80	90	60	30		
0	20	30	50	50	60	40	20		
10	20	30	30	30	30	20	10		
10	10	10	0	0	0	0	0		

Come si può notare il risultato $G[x, y]$ non è uniforme in tutte le direzioni, infatti i valori sui bordi sono molto più bassi di quelli al centro. Questo perché i pixels dell'immagine di partenza sui bordi hanno tanti zeri quindi il risultato della convoluzione in quei punti sarà più basso.

In particolare questo accade perché l'avarage kernel è **non isotropico**, ovvero i pixels dell'intorno che consideriamo non hanno tutti la stessa distanza dal pixel di riferimento. In un kernel **isotropico**, invece, tutti i pixels dell'intorno hanno la stessa distanza dal pixel che stiamo considerando.

L'applicazione di questo tipo di filtro non va bene per il salt and pepper noise perché un filtro lineare applicato a questo tipo di noise non lo elimina ma lo spalma/distribuisce nell'intorno generando una sfuocatura inutile, mentre è molto adatto per eliminate il rumore gaussiano.

Il tipo di filtro che utilizziamo dipende dai valori che contiene il suo kernel. Esistono diversi tipi di filtri oltre a quello medio.

In generale i valori numerici da inserire nel kernel del filtro, la sua dimensione e il modo in cui si ottengono (definiti o appresi?) variano a seconda del task che dobbiamo eseguire sull'immagine.

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \quad \begin{matrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{matrix}$$

Defined or learned?



(a)



(b)

It also depends on the task.



(c)



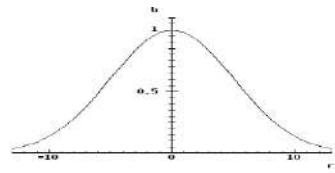
(d)

Il filtro migliore per ridurre il Gaussian noise è il **Gaussian filter** o filtro Gaussiano.

Il Gaussian filter è una maschera isotopica data da una funzione gaussiana con media 0 e deviazione standard determinata usata per fare la convoluzione sull'immagine.

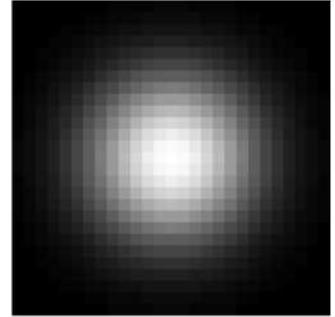
In 1D

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2}$$



In 2D

$$G(x, y) = G(x)G(y) = \frac{1}{\sigma^2 2\pi} e^{-(x^2+y^2)/2\sigma^2}$$



Ovviamente il filtro deve essere discretizzato scegliendo la sua dimensione k e la sua deviazione standard. Avremo così una maschera kxk dove solitamente k e σ sono scelte in modo tale che k sia circa 5σ . In questo modo si coprono circa il 99% dei valori della distribuzione che stiamo considerando. Solitamente, per ragioni computazionali, si utilizzano $\sigma=1$ e $k=5$.

Eg $\sigma=1$ $k=5$

$$\begin{matrix} h = & 0.0029 & 0.0131 & 0.0215 & 0.0131 & 0.0029 \\ & 0.0131 & 0.0585 & 0.0965 & 0.0585 & 0.0131 \\ & 0.0215 & 0.0965 & 0.1592 & 0.0965 & 0.0215 \\ & 0.0131 & 0.0585 & 0.0965 & 0.0585 & 0.0131 \\ & 0.0029 & 0.0131 & 0.0215 & 0.0131 & 0.0029 \end{matrix}$$

Best values

$\sigma=1$	7 X 7
$\sigma=2$	13 X 13
$\sigma=3$	19 X 19

Nell'esempio, i valori del filtro h sono i valori per x e y compresi fra -2 e +2 (essendo k=5) della distribuzione gaussiana avente $\sigma=1$.

I pesi che vanno a comporre il kernel sono campionati dalla funzione gaussiana calcolando tale valore per diversi valori assegnati a x e y. Ad esempio se la maschera è 3x3 x e y si suppongono compresi fra 1 e -1.

For x and y ranging among -1 and +1 (mask 3x3)

Too approximated...

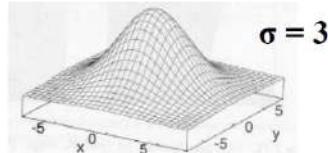
$$\frac{1}{16} \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

A mask 7x7

7×7 Gaussian mask

$$\begin{array}{ccccccc} 1 & 1 & 2 & 2 & 2 & 1 & 1 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 2 & 2 & 4 & 8 & 4 & 2 & 2 \\ 2 & 4 & 8 & 16 & 8 & 4 & 2 \\ 2 & 2 & 4 & 8 & 4 & 2 & 2 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 & 1 & 1 \end{array}$$

$\sigma = 1.4$



and 15 x15

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

15 x 15 Gaussian mask																			
2	2	3	4	5	5	6	6	6	5	4	3	2	2	3	4	5	7	7	5
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2	3	4	6	9	7
3	4	6	7	9	10	10	11	10	9	7	6	4	3	4	5	7	9	10	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4	5	7	9	11	13
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5	7	9	11	13	12
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5	7	10	12	14	16
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6	8	10	13	15	17
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6	8	10	13	15	17
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6	7	10	12	14	16
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5	7	10	12	14	16
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5	7	9	11	13	12
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4	5	7	9	10	8
3	4	6	7	9	10	10	11	10	9	7	6	4	3	4	5	7	9	10	5
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2	3	4	5	6	4
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2	3	4	5	6	5

$height = width = 5\sigma$ (subtends 98.76% of the area)

Il filtro gaussiano più usato è:

$$\frac{1}{16} \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

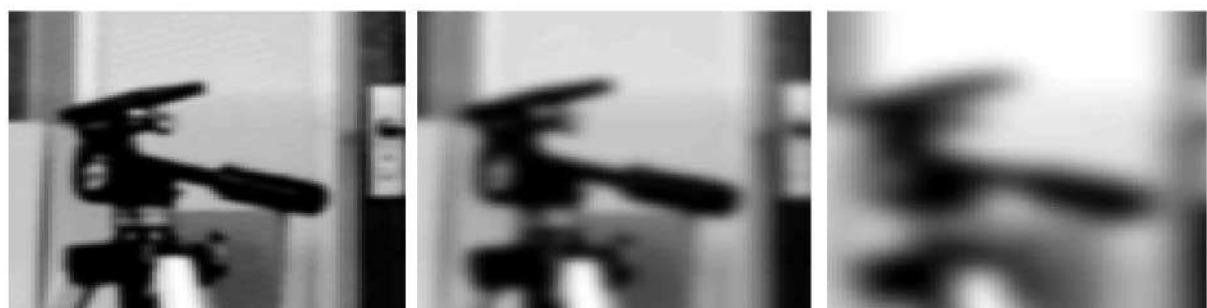
Esempi di filtri gaussiani applicati a questa immagine:



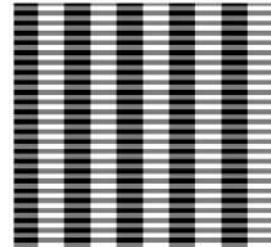
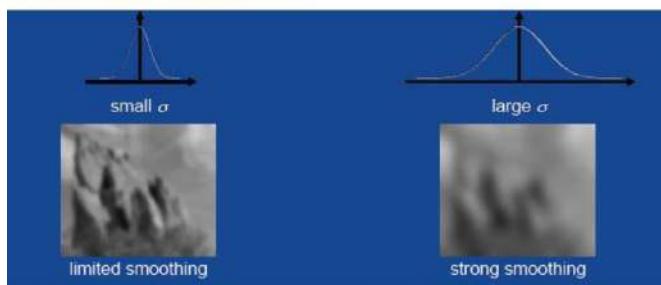
Let's use three different filters given by a Gaussian function with different shape due to different sigma

$\sigma=1, \sigma=2, \sigma=5$

Results: blurred, cleaned or also they can abstract a shape for a cognitive process...



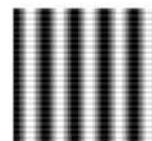
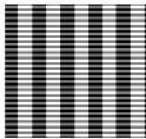
Esempio: in questo caso l'obiettivo del filtro è quello di eliminare le linee orizzontali mantenendo solo quelle verticali.



Let's suppose the horizontal lines
As a noise to be cleaned

Let's suppose this a part of a
Barcode noisy acquired..

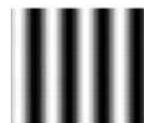
Averaging vs Gaussian Smoothing



Averaging



Gaussian



Si può notare che il gaussian filter funziona meglio perché è isotropico, quindi uniforme in tutte le direzioni, rispetto al avarage filter che non è isotropico, quindi non è uniforme in tutte le direzioni.

Un altro filtro rilevante è quello generato per fare il **cognitive smoothing**. I valori del kernel, che vengono definiti o in modo fisso oppure in modo dipendente da assunzioni sul contesto che stiamo trattando, devono essere scelti in modo tale che la convoluzione tra il kernel con quei valori e l'immagine corrisponda al cognitive snoothing.

“We see what we want to see”

Per cognitive snoothing si intende la capacità di vedere in un'immagine che raffigura più figure sovrapposte e intrecciate una figura piuttosto che un'altra. I neuroni del nostro cervello fanno esattamente la stessa cosa di uno smoothing filter. Questo concetto si chiarisce meglio con un esempio:



Noi vediamo nell'immagine sopra una donna o un teschio a seconda del nostro mood, stessa cosa vale per il filtro. Nel caso specifico scegliamo i valori numerici del kernel in modo tale che rappresentino perfettamente i mood che può avere una persona.

Se a questa immagine applichiamo un low pass filter come il gaussian filter o l'avverage filter otteniamo il teschio, mentre se applichiamo un high pass filter otteniamo la donna.

The All Is Vanity the figure can be seen as a woman looking at her reflection in a mirror, or a skull. They are ambiguous figures (also known as '**reversible figures**' or '**bistable figures**').

There is some controversy over how the All Is Vanity Ambiguous Figure works. It is generally agreed that the retinal image is constant when experiencing the illusion, but what is not agreed is whether the visual experience of the figure changes when the perspectival switch takes place between seeing the woman versus the skull, or whether the experience itself does not change, and it is some post-experiential belief, judgment, or other mental process which changes.

There is also a question about what **the role of attention** is in generating **Gestalt switches**.



In psychology, a gestalt shift is when your perception suddenly changes.

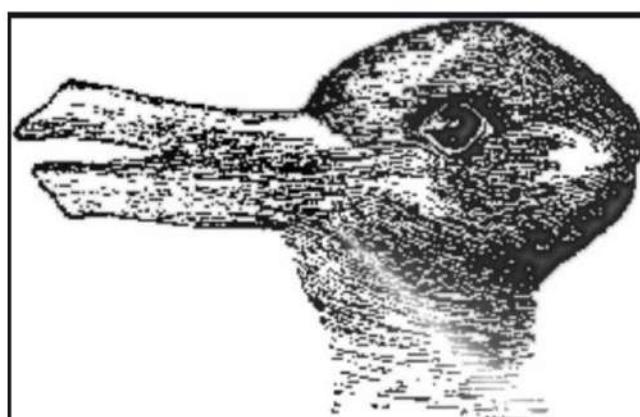
This issue is intertwined with more general questions about the **modularity of mind and cognitive penetration**

To explain: on the hypothesis that the mind is modular, a mental module is a kind of semi-independent department of the mind which deals with particular types of inputs, and gives particular types of outputs, and whose inner workings are not accessible to the conscious awareness of the person – all one can get access to are the relevant outputs.

So, in the case of visual illusions, for example, a standard way of explaining why the illusion persists even though one knows that one is experiencing an illusion is that the module, or modules, which constitute the visual system are '**cognitively impenetrable**' to some degree –

Further, there is some evidence from neuroscience that, for at least some ambiguous figures, there are significant changes in early-stage visual processing in the brain when the **Gestalt switch** is taking place, which might support the hypothesis that Gestalt switches in general are changes in the experience itself rather than in downstream mental processes like beliefs about that experience (see Kornmeier & Bach 2006, 2012).

Gestalt shift



A QUESTION FOR FUTURE
Visual intelligence?
can we create a filter capable to follow different
Mental models?

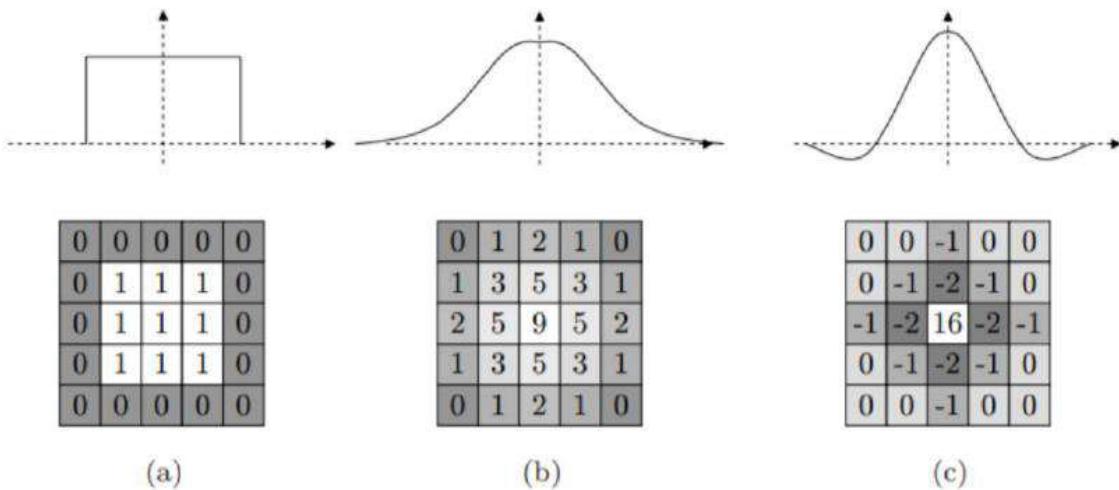
Wittgenstein (1953) formulates the paradox of gestalt switches thus: "What is incomprehensible is that nothing, and yet everything has changed, after all. That is the only way to put it"

Rivediamo ora le proprietà della convoluzione lineare. Essa è:

- Commutativa: l'ordine fra immagine di input e filtro può essere invertito senza cambiare il risultato
- Lineare: rimane valida se moltiplichiamo per un fattore di scala
- Associativa: possiamo utilizzare più filtri in cascata o un solo filtro composto dall'insieme di tutti i filtri utilizzati e il risultato non cambia
- Separabile: il kernel H può essere rappresentato come la convoluzione fra più kernel, ovvero i kernels che si ottengono da H separandolo in una coppia di kernels x e y.

Fin ora abbiamo visto solo dei low pass filter: avarage, gaussian e laplacian filter. **Laplacian filter** lo introduciamo ora però è sempre un filtro passa basso che serve per trovare i bordi presenti nell'immagine e sostanzialmente calcola la derivata seconda dell'immagine misurando la frequenza con cui cambia la derivata prima. Il laplacian filter misura l'intensità del cambiamento del valore di pixels adiacenti per capire se tale cambiamento è dovuto alla normale progressione dei valori oppure se è un cambiamento che denota la presenza di un bordo.

<https://www.l3harrisgeospatial.com/docs/laplacianfilters.html#:~:text=A%20Laplacian%20filter%20is%20an%20edge%20or%20continuous%20progression.>

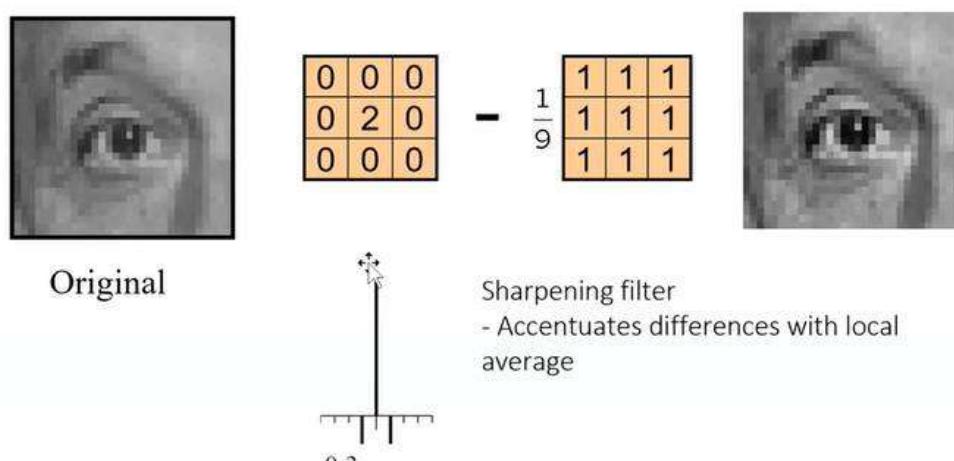


Ora analizziamo gli high pass filter.

Fin ora abbiamo analizzato i low pass filter. L'immagine che otteniamo dalla applicazione di un low pass filter è un'immagine in cui i nostri occhi vedono peggio i dettagli. Ora andiamo ad analizzare gli high pass filter che sono i filtri che applicati all'immagine ne esaltano i dettagli facendoli vedere meglio ai nostri occhi.

Sharpening

Essendo la convoluzione un operatore lineare gode delle proprietà degli operatori lineari tra cui la possibilità di effettuare combinazioni lineari. Lo sharpening viene, appunto, effettuato utilizzando due filtri lineari in cascata nel seguente modo:



Si può notare che la combinazione dei due filtri così definita calcola il negativo (circa) dei pixels nell'intorno. Ha l'obiettivo di accentuare i bordi rendendoli più evidenti.

Generalizzando questo concetto possiamo dire che eseguiamo l' **unsharp filtering** quando applichiamo un high pass filter della seguente forma:

$A >= 1$		
$w = 9A - 1$		
-1	-1	-1
-1	w	-1
-1	-1	-1

$A = 2$		
$w = 17$		
-1	-1	-1
-1	17	-1
-1	-1	-1

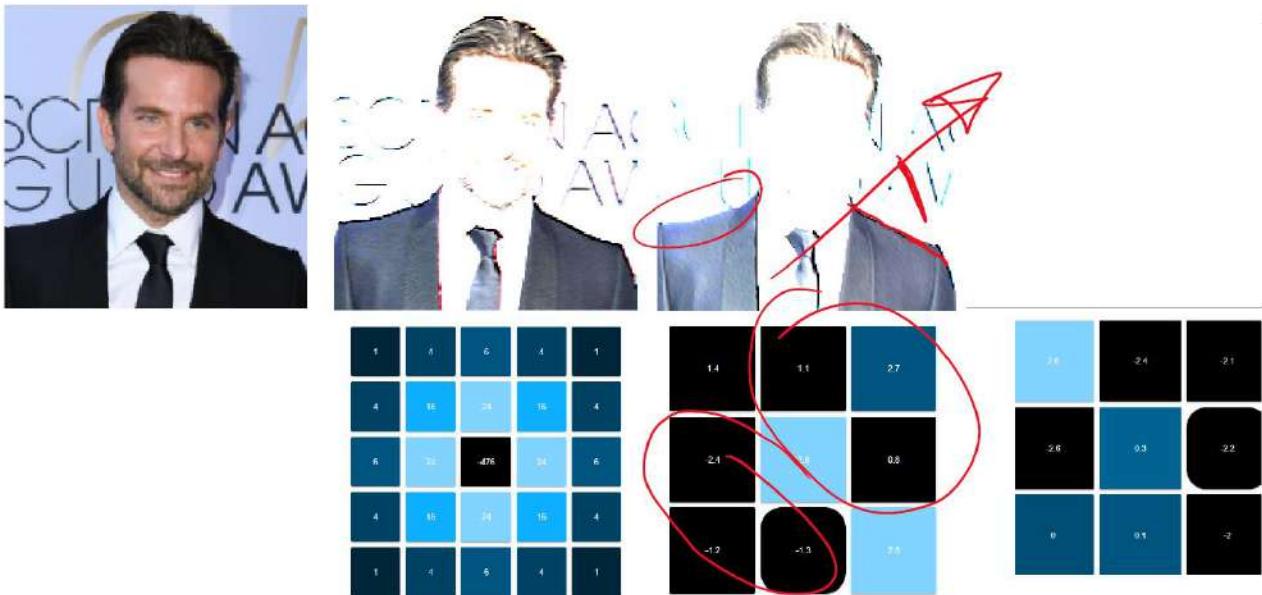


Questo filtro è utile per evidenziare transizioni dell'intensità dell'immagine come i bordi. E' anche utilizzato in photoshop con $A=1.4$.

Quiz. Which mask has been used?

Se io ho un'immagine 1000×1000 e voglio fare la convoluzione con un filtro 5×5 il numero totale di operazioni che eseguirò sarà: $10^3 \times 10^3 \times 5 \times 5$.

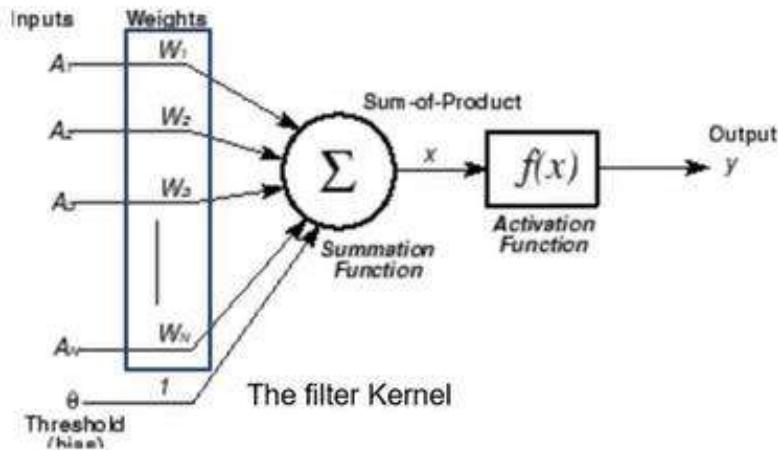
E' inoltre possibile applicare un altro tipo di filtri chiamati **bilateral filter** che evidenziano i bordi che vanno in una sola direzione e cancellano quelli che vanno nelle altre. Ad esempio, nella terza immagine sono mantenuti solo i bordi vero destra poiché in quella direzione ci sono valori positivi mentre gli altri sono cancellati. Questo filtro compie una sorta di derivata in una direzione, ne parleremo dopo.



Convolutional neural network CNN

Consideriamo l'idea di rete neurale, il problema di avere troppi parametri e come la convoluzione inserita nelle reti neurali risolva questo problema, costituendo le CNN.

Consideriamo un neurone. In generale esso può essere visto come un tipo di filtro lineare. Le reti neurali hanno la seguente struttura:



e svolgono la seguente operazione:

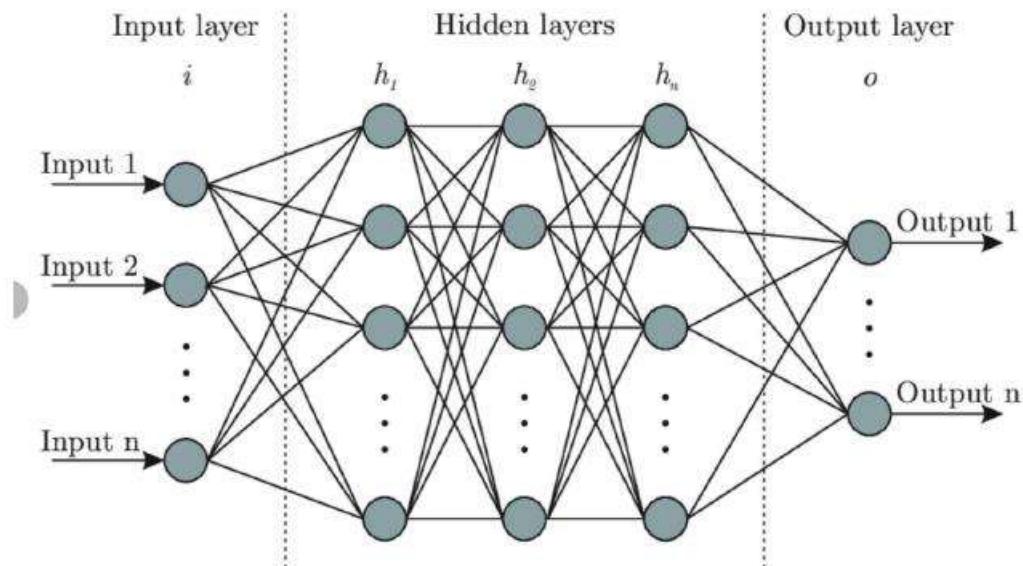
$$a(X) = \sum_{i=1}^n w_i A_i$$

Dove A_i è l'input e w_i è il set di pesi utilizzati e pesi della rete corrispondono ai valori contenuti nei filtri che vengono usati. Possiamo scegliere di usare diversi filtri in funzione dell'obiettivo che vogliamo raggiungere.

Se non sappiamo qual è il filtro migliore da utilizzare i pesi possono essere imparati.

Vedi meglio funzionamento generico: https://it.wikipedia.org/wiki/Rete_neurale_artificiale

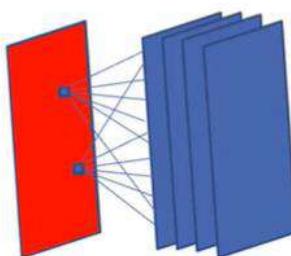
Appena le reti neurali furono inventate l'architettura utilizzata era la seguente:



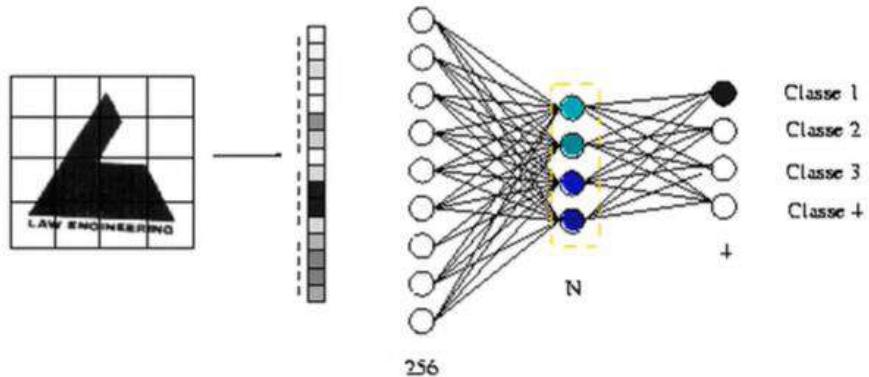
Cioè una generica architettura multi-livello in cui è presente un neurone per ogni pixel di input (non è detto che sia sempre così), in cui ogni neurone è collegato con tutti i neuroni del livello successivo e in **cui ogni neurone applica un differente filtro lineare**. Nel primo livello nascosto h_1 , **ogni neurone riceve lo stesso input (cioè l'intera immagine)**, essendo collegato a tutti i neuroni di input) e vi applica un filtro della stessa dimensione dell'immagine definito dai pesi di quel neurone. **Ogni neurone dello strato applica un filtro diverso** quindi la rete, per ogni livello, deve imparare un numero di pesi pari al numero di pesi di ogni neurone moltiplicato per il numero di neuroni presenti nel livello.

Questo significa che in questa architettura, essendoci un numero elevato di neuroni, è presente un numero troppo grande di parametri.

Ipotizziamo di avere un'immagine composta da 100x100 pixels (10.000 input) e di avere una rete che ha 10.000 pesi per ogni neurone (per un singolo filtro). Se abbiamo, nel primo livello, un neurone per ogni pixel in input avremo 10.000 neuroni con 10.000 parametri l'uno, per un totale di 10.000x10.000 parametri da ricordare. Se poi vogliamo applicare più di un filtro, ad esempio $n=50$ filtri, avremo $50 \times 10.000 \times 10.000$ differenti parametri da imparare. Ovviamente questo è un numero troppo elevato.



Facciamo un altro esempio di un'architettura molto utilizzata in passato. L'obiettivo è quello di riuscire a classificare il logo rappresentato dall'immagine data in input. L'architettura utilizzata è la seguente:



L'immagine è in bianco e nero, quindi i valori dei pixels sono o 0 o 1 (255). Dividiamo l'immagine in n blocchi, ad esempio 256, ed assegniamo ad ogni blocco un valore ottenuto dalla media dei valori dei pixels in quel blocco. In questo modo l'immagine viene rappresentata con soli n valori ottenuti in questo modo. Questo vettore che rappresenta l'immagine viene dato in input alla rete neurale in cui l'input layer è costituito da un neurone per ogni valore del vettore ed è fully connected con tutti i neuroni dello strato nascosto, che sono N. Non c'è una regola generale sulla scelta di quanti neuroni utilizzare, bisogna andare per tentativi. Per quanto detto prima, questo tipo di reti può essere usato solo per risolvere problemi computazionalmente semplici perché all'aumentare dei neuroni il numero dei parametri diventa troppo alto.

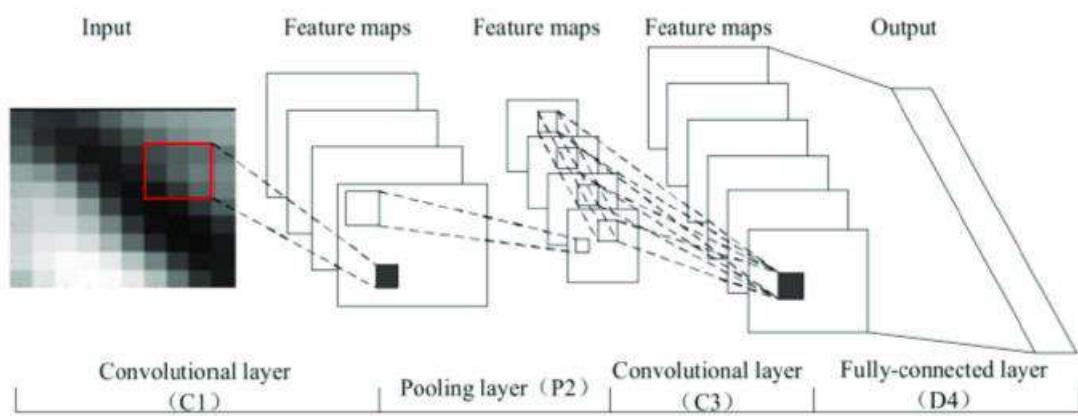
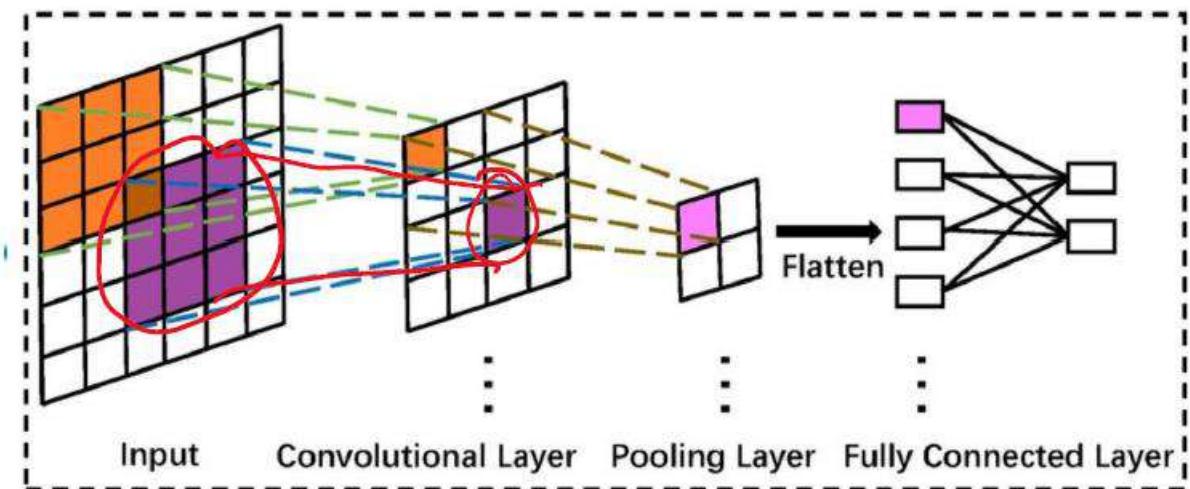
Per risolvere il problema dei troppi parametri da imparare, sono state pensate le reti neurali convolutive CNN, che sono reti neurali che implementano l'operazione di convoluzione. Andando più nel dettaglio le idee che stanno alla base delle CNN sono:

- **Receptive field:** ovvero tutti i neuroni non ricevono in input l'intera immagine bensì **ogni neurone riceve in input una sola porzione dell'immagine**, similmente a quanto accade nel cervello umano.
- **Ogni neurone utilizza lo stesso set di parametri come filtro**, ovvero lo stesso kernel, per generare l'immagine di output nel seguente modo:

$$a(X) = \sum_{i=1}^n w_i x_i$$

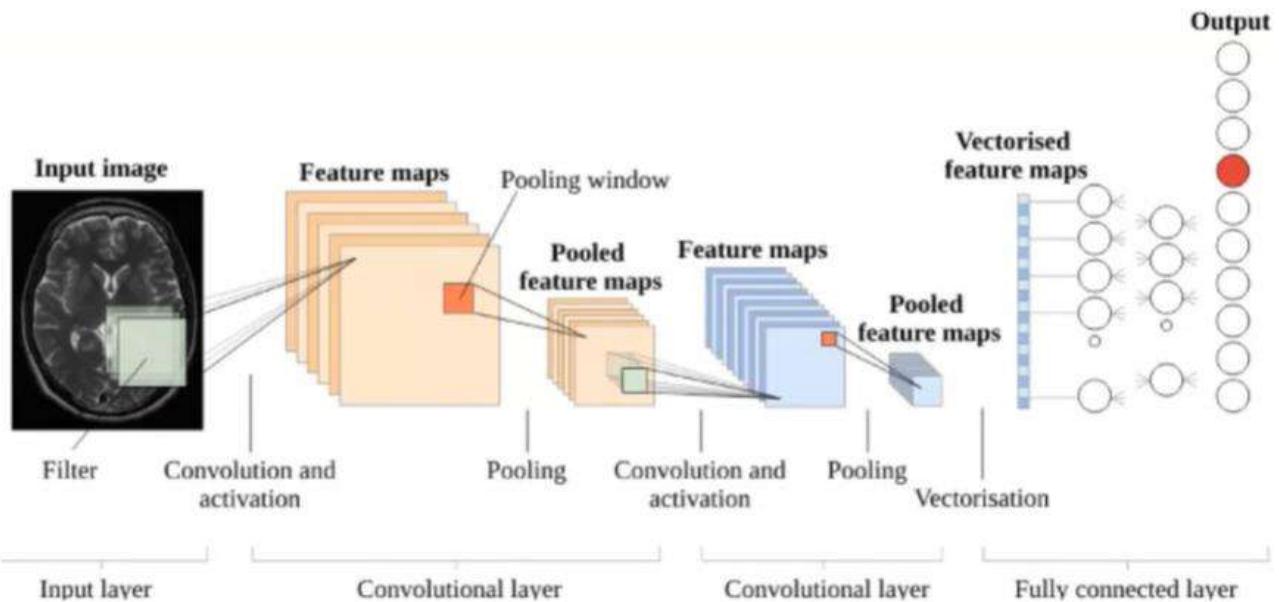
Questo non avviene nel cervello umano, è stata scelta questa strategia per ragioni computazionali. In più viene usato un filtro lineare quindi la computazione è shift-invariant.

- Più kernel diversi possono essere usati in parallelo per costruire più output diversi, solitamente chiamati **feature maps** (o feature vectors quando sono compressi). Quindi possiamo dire che i feature maps sono i risultati dell'applicazione di ognuno dei diversi filtri sull'immagine.
- I feature maps sono processati in modo non lineare da **funzioni non lineari** poi **compresse da pooling** per lavorare in multirisoluzione o per allargare il receptive field (la porzione di immagine processata dal neurone). Le funzioni non lineare servono per prendere le decisioni.
- I feature vectors sono usati per fare **inferenza finale**, ovvero per **mettere insieme le decisioni** prese fin ora utilizzando un **layer fully connected** che si occupa della classificazione. Questo viene fatto anche per ridurre l'overfitting.



**Filters (weights) can be learned by data, instead of hand-crafted.
(but sometime the model knowledge is enough and training is not needed)**

Vediamo un esempio di applicazione di una CNN:



I filtri che usiamo nella rete possono essere impostati manualmente quando sappiamo su cosa stiamo lavorando e sappiamo cosa è meglio usare per il nostro obiettivo, oppure possono essere imparati se non sappiamo nulla. Se i parametri w sono imparati gli aspetti che possono influire sull'apprendimento sono:

- Architettura
- Modello e loss function
- Dati annotati
- Gpu (potere computazionale)

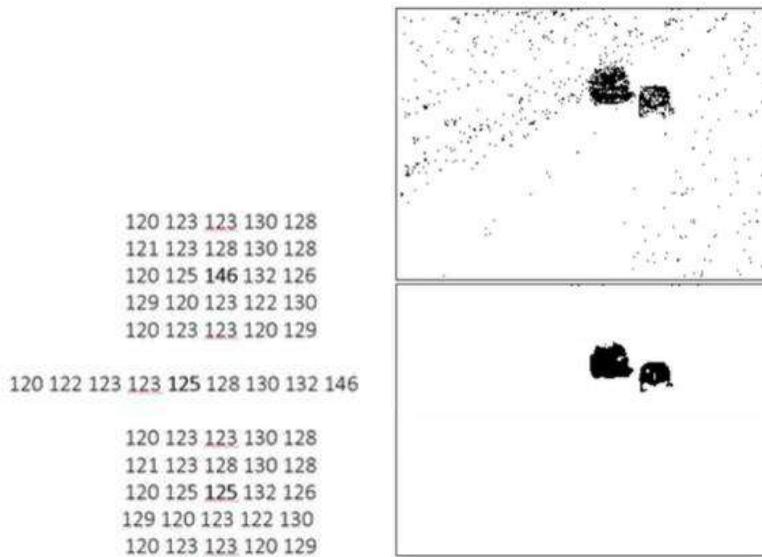
Ci sono certe occasioni in cui bisogna prendere decisioni non lineari (come si/no, sostituire un dato o usare una soglia) e in questi casi si utilizzano **filtri non lineari**. I principali filtri non lineari che vedremo sono **min max filter** e **median filter**.

Il **min max filter** è un filtro non lineare che restituisce rispettivamente il valore minimo e il valore massimo tra i valori dei pixels della porzione R di immagine che stiamo considerando. Un filtro di questo tipo è ad esempio il max pooling.

$$I'(u, v) \leftarrow \min \{I(u+i, v+j) \mid (i, j) \in R\},$$

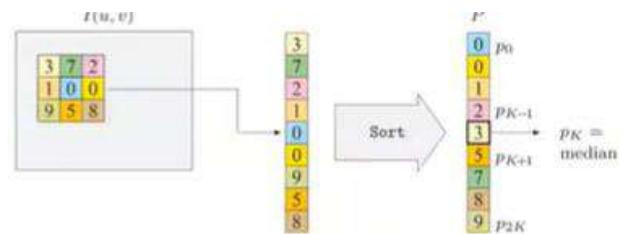
$$I'(u, v) \leftarrow \max \{I(u+i, v+j) \mid (i, j) \in R\},$$

Il **median filter** è un filtro non lineare molto utile per ridurre l'impulsive noise, come il salt and pepper noise. Il pixel dell'output corrisponde al valore mediano della regione R dell'immagine di input considerata. In particolare considero i valori della porzione dell'immagine di input che sto considerando, li metto in ordine, e individuo la mediana. Il pixel dell'immagine di output corrispondente a quella regione avrà quel valore.



In particolare questo filtro ha l'obiettivo di scartare gli outliers. L'**algoritmo applicato** è il seguente (implementa a computer):

- 1) Consider a 2D neighborhood
- 2) Order it
- 3) Choose the central value
- 4) Substitute pixel with the median one



Inoltre è possibile calcolare la mediana pesata per dare più peso a certi pixels piuttosto che ad altri. I pesi w possono essere fissi o possono dipendere dalla distanza dal pixel centrale della porzione. Dal punto di vista statistico il valore mediano $g(i,j)$ è il valore che minimizza la seguente funzione con $p=1$:

$$\sum_{k,l} w(k,l) |f(i+k, j+l) - g(i, j)|^p,$$

mentre se $p=2$ abbiamo una mediana pesata. (non ho capito perchè).

Tornando al discorso più generale, i filtri convolutivi high pass e low pass sono lineari e shift invarianti e i valori dei kernels possono essere fissi o imparati. A volte possiamo decidere di modificare i filtri in base alla porzione di immagine che stiamo elaborando (non applico lo stesso filtro a tutta l'immagine ma cambia in base alla posizione in cui sono) ottenendo così dei **filtri a valori variabili**. Il principale filtro di questo tipo che vediamo è il **bilateral filter**.

Il **bilateral filter** è un filtro in grado di ridurre il rumore mantenendo però i bordi nitidi e non sfocati. Questo è ottenuto grazie al fatto che ogni pixel è sostituito dalla media pesata dei pixels nel suo intorno e i pesi sono scelti sia in funzione della vicinanza con il pixel centrale di riferimento, secondo l'idea che il peso sarà più alto per i pixels più vicini e più basso per quelli più lontani, che in funzione della similarità con il pixel centrale secondo l'idea che più i pixel sono simili più sarà probabile che l'informazione che mi forniscono sia rilevante (un bordo). Più i pixels sono simili più il peso è alto, più sono diversi più è basso.

Grazie ai valori dei pesi riesco a mantenere memorizzata similarità e vicinanza tra i pixels.

Tutti i pesi devono essere poi normalizzati per mantenere la media locale.

Si può dire che il bilateral filter è un low pass filter in grado di rimuovere il rumore e mantiene i bordi, ovvero mantiene la derivata, che si ottiene combinando un kernel contenente valori (pesi) in base alla distanza spaziale e un kernel contenente valori (pesi) sulla base della similarità cioè della distanza in termini di colore.

Esempi:

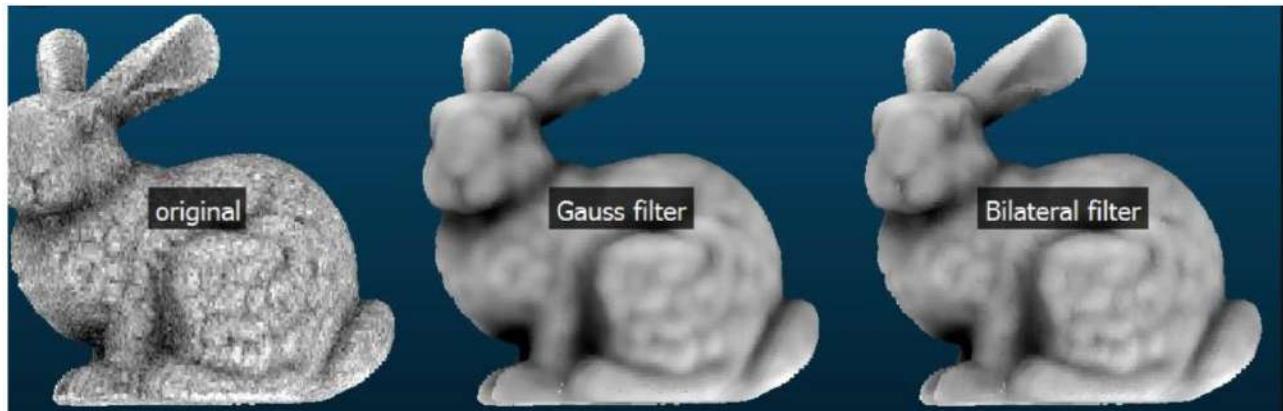




(a)



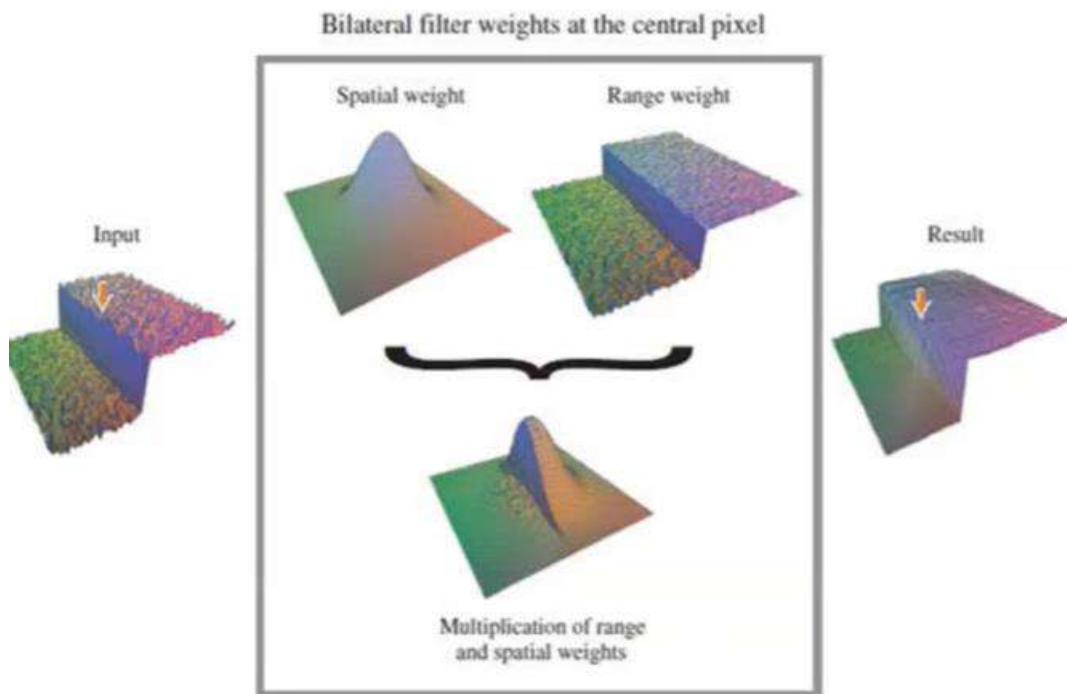
Confronto fra bilateral filter e gaussian filter:



Bilateral Filtering in Photoshop (thanks to Ben Weiss Siggraph 2006)



Ora andiamo più nello specifico a vedere come funziona il bilateral filter:



Come abbiamo detto prima, i valori del kernel del bilateral filter sono dati dalla composizione di due kernels diversi, uno che elimina il rumore ed è basato sulla distanza spaziale e uno che evidenzia i bordi ed è basato sulla similarità. Si può notare che il kernel basato sulla vicinanza sarà un kernel gaussiano in quanto i pixel nell'immediato intorno avranno valori alti e via via che ci si allontana i valori scendono (lo stesso kernel per tutta l'immagine). Mentre il kernel con valori dovuti alla similarità sarà un derivative/range kernel, ovvero un kernel che, tenendo conto della distanza fra i colori (o più in generale fra i valori) presenti, dipende dalla parte specifica di immagine che stiamo considerando e cambia in base a quale porzione consideriamo (non è shift-invariant).

La convoluzione normalizzata (standardizzata) fra la porzione dell'immagine data dall'intorno di $f(i,j)$ e il bilateral filter così definito è espressa dalla somma normalizzata rispetto ai pesi della convoluzione fra quella porzione dell'immagine e il filtro i cui pesi sono espressi da $w(i,j,k,l)$ secondo la seguente formula: in a neighborhood of $f(i,j)$ the result $g(i,j)$ is a normalized weighted sum

weights are given by
$$g(i,j) = \frac{\sum_{k,l} f(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}.$$

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right).$$

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right), \quad \text{Domain kernel}$$

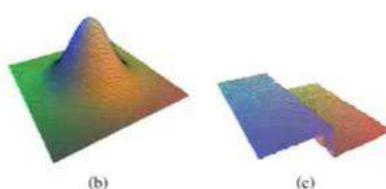
Range kernel

$$r(i,j,k,l) = \exp\left(-\frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right)$$

i	j	k	l	$d(i,j,k,l)$	$r(i,j,k,l)$	$w(i,j,k,l)$
2	1	0	1	0.0	0.0	0.0
1	0.1	0.3	0.4	0.3	0.1	0.0
2	0.6	0.8	0.6	0.5	0.0	0.0
0	0.4	0.8	1.0	0.8	0.4	0.0
1	0.3	0.6	0.8	0.6	0.5	0.0
2	0.1	0.5	0.4	0.3	0.1	0.2

(c) domain filter

(d) range filter



In color the range kernel is a vector distance

I pesi sono gli esponenziali dei due diversi tipi di kernel. (i,j) è la posizione del pixel mentre (k,l) sono le coordinate del kernel.

(imparo questa formula).

Ora andiamo a vedere come si misura la qualità del filtro applicato. Ora i filtri non lineari sono usati più frequentemente dei filtri lineari.

Per misurare la qualità del filtraggio applicato normalmente si aggiunge un rumore sintetico nel seguente modo:

$I(\mathbf{x})$ starting image

$I'(\mathbf{x}) = I(\mathbf{x}) + N(\mathbf{x})$ with noise added

$\hat{I}(\mathbf{x}) = \text{filter}(I'(\mathbf{x}))$

Dopo di che si calcola una metrica chiamata **PSNR (peak signal-to-noise ratio)**.

$$\text{PSNR} = 10 \log_{10} \frac{I_{\max}^2}{MSE}$$

Dove I_{\max} è il valore massimo del range (255 nel nostro caso) e MSE è il mean square error massimo.

$$MSE = \frac{1}{n} \sum_{\mathbf{x}} [I(\mathbf{x}) - \hat{I}(\mathbf{x})]^2$$

Ora vediamo come possiamo decidere quali sono i migliori coefficienti da inserire nei kernels per fare **denoising**. A volte i coefficienti possono essere decisi manualmente o localmente (si può decidere di avere filtri diversi per diversi tipi di immagini vedi esempio slide 65), altre volte devono essere imparati. Quando vengono imparati solitamente si punta ad avere un filtro che ha lo stesso obiettivo, che in questo caso è il denoising, per tutte le immagini quindi l'obiettivo è quello di **imparare i migliori coefficienti** possibili per l'intero set di immagini che abbiamo.

Vediamo ora come, dato un obiettivo, i coefficienti migliori possono essere imparati usando una rete neurale. La prima proposta utilizza un **multi-layer perceptron** ed è la seguente
<http://www.hamburger.com/files/neuraldenoising.pdf> mentre altre proposte prevedono l'uso di **una rete neurale convolutiva CNN** nella seguente forma:

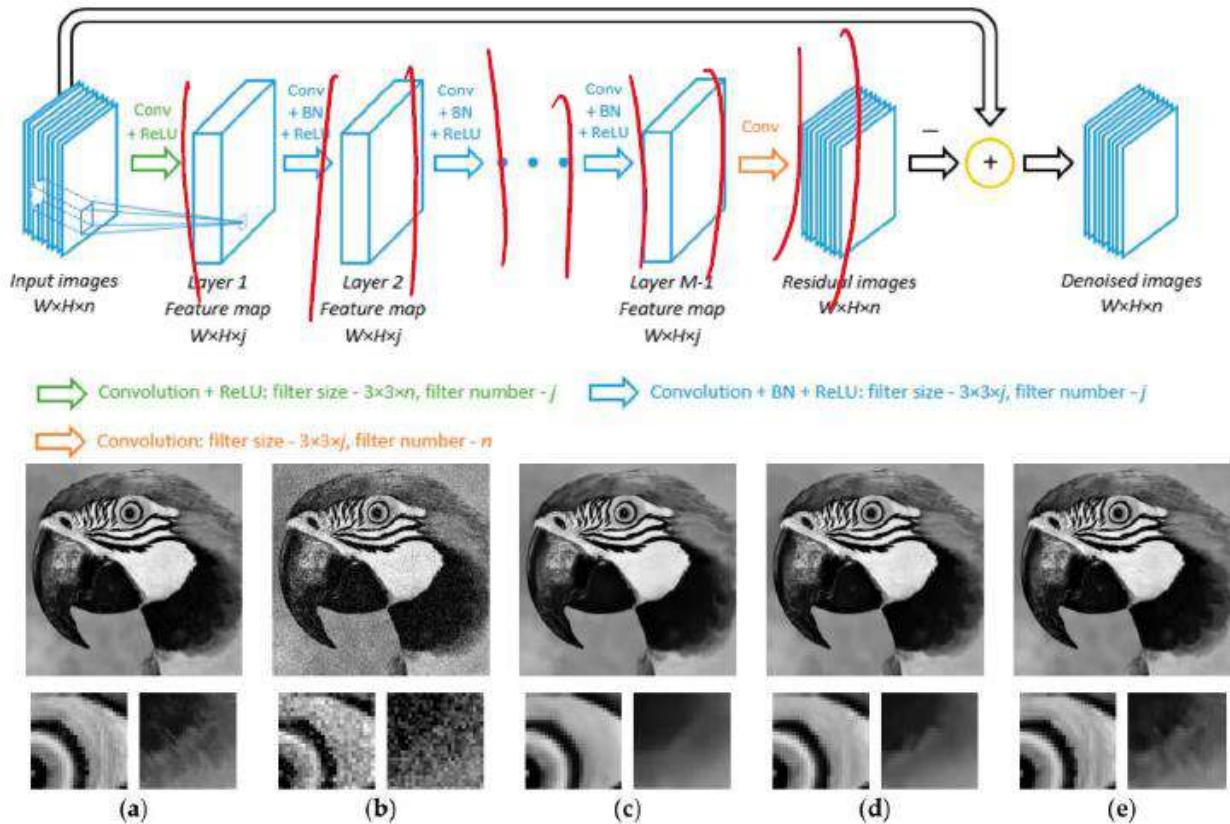


Figure 2. Comparison of single image convolutional neural network (CNN) and multi-view CNN models. (a) Ground truth; (b) Noisy image, PSNR = 20.15 dB; (c) DnCNN [28], PSNR = 29.44 dB; (d) Multi-view image denoising algorithm based on convolutional neural network (MVCNN) (3 views),

In questo caso vengono eseguiti semplici esperimenti di filtraggio multiplo su immagini impilate come input.

Questo funziona, però nella maggior parte dei casi è inutile usare un procedimento così complesso per un'operazione così semplice. Quando è possibile, ovvero si ha un minimo di conoscenza sul dominio dei dati, è preferibile settare i valori del kernel in modo manuale.

Un'altra tecnica, molto più interessante e utile della prima, per apprendere i valori del kernel è quella di utilizzare **autoencoders**. In particolare questa soluzione viene scelta per imparare la corretta ricostruzione dell'immagine quando non conosciamo il tipo di rumore presente. Questa è una tecnica molto usata per ricostruire immagini. Vedi teoria autoencoders.

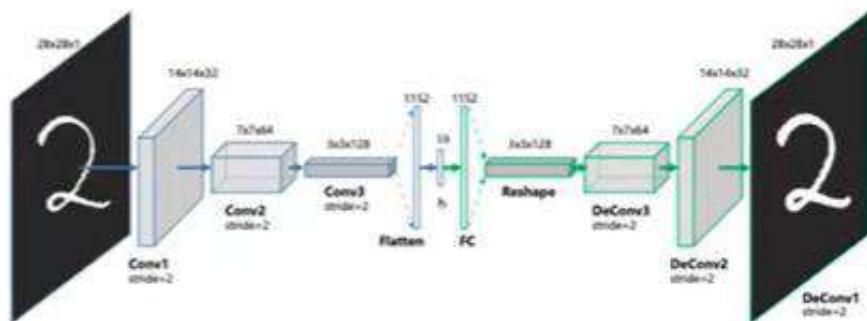
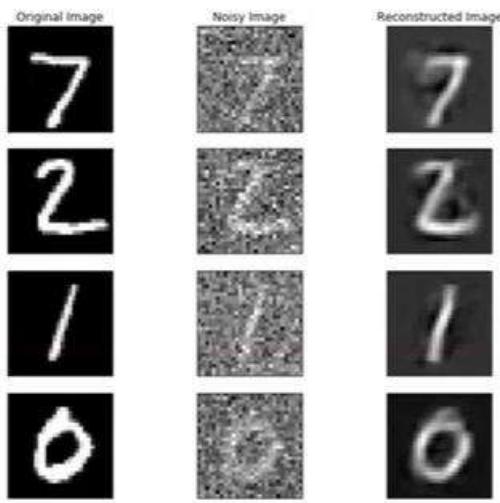
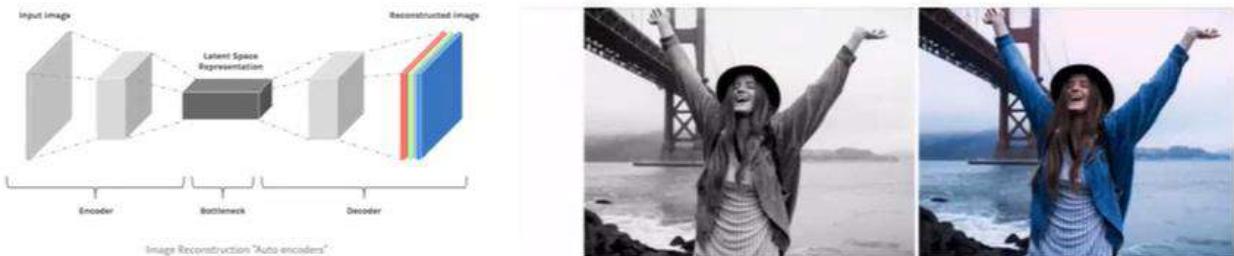


Figure (D)

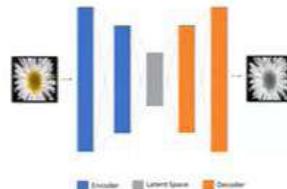


In realtà gli autoencoder possono essere utilizzati per diversi tipi di elaborazioni sulle immagini, non solo per imparare i migliori valori del kernel di denoise.

Ad esempio, un'elaborazione che può essere fatta usando autoencoder è quella di colorare le immagini. In questo caso viene imparato dai dati come colorare le immagini in differenti spazi di colore, oppure si può ottenere l'immagine in livelli di grigio partendo da un'immagine colorata. Però questo è veramente inutile.

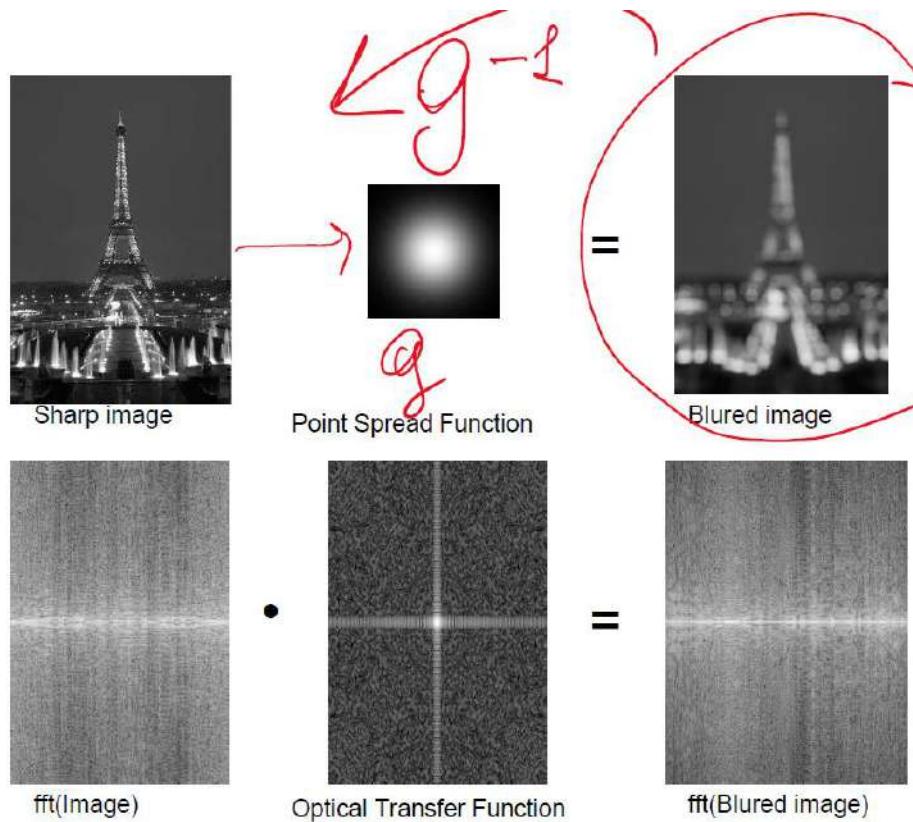


Or to create gray level images! Do not do it!
("take a hammer to crack a nut")



Un'elaborazione più interessante di quelle appena viste è il deblurring.

Il deblurring è l'elaborazione che consente di ricostruire l'immagine originale partendo da un'immagine sfocata. La sfocatura (blur) è la degradazione della nitidezza e dei contrasti nell'immagine dovuta alla perdita delle alte frequenze e può essere dovuta sia alla convoluzione con un filtro per rimuovere il rumore che a instabilità o perdita di messa a fuoco della telecamera. Questo è un problema molto complesso e ancora aperto, infatti sono numerose le attuali ricerche in questo campo.



In generale la convoluzione con un filtro rimuove il noise presente nell'immagine, tuttavia ne aggiunge un altro diverso dovuto proprio a questa operazione, motivo per cui è molto difficile ricostruire l'immagine di partenza.

$$y = x \otimes k + n$$

blurred image sharp image blur kernel noise

Se il noise fosse uguale a zero potremmo semplicemente fare la convoluzione con il filtro inverso e otterremmo l'immagine di partenza:

$$y = x \otimes k \xrightarrow{F} Y(f) = X(f) \cdot K(f)$$

$$\Downarrow$$

$$X(f) = Y(f) \cdot K(f)^{-1}$$

Tuttavia in generale il nuovo noise non è mai uguale a zero per cui se si applica il filtro inverso all'immagine sfocata senza eliminare l'additive noise presente si rischia di propagarlo notevolmente un'immagine disastrosa.

$$y = x \otimes k + n \xrightarrow{F} y(f) = X(f) \cdot K(f) + N(f)$$

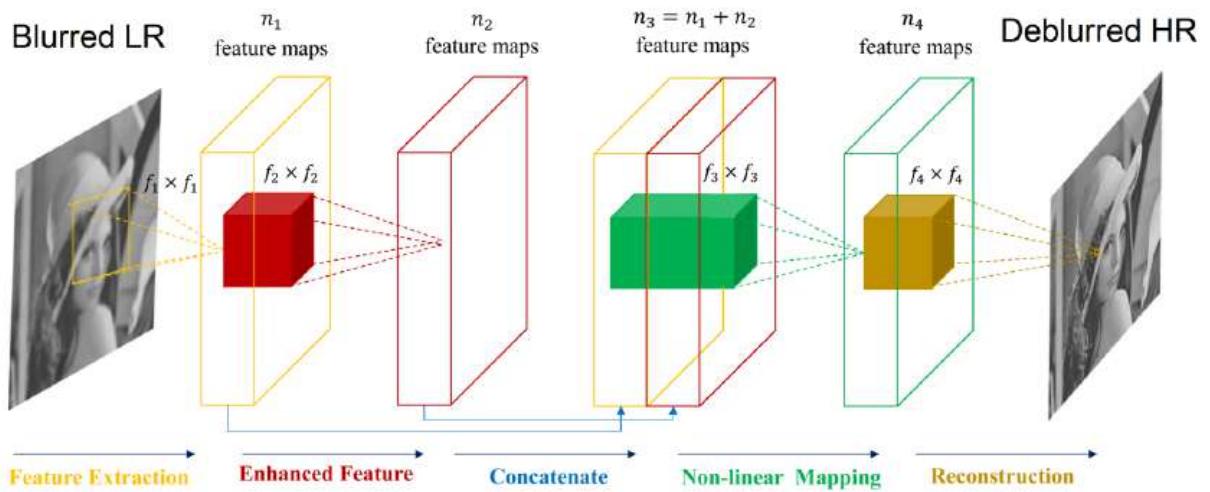
$$\downarrow$$

$$X(f) = Y(f) \cdot K(f)^{-1} + N(f) \cdot K(f)^{-1}$$



Sono attualmente presenti diverse soluzioni a questo problema ma, come detto prima, la ricerca in questo campo è ancora fortemente attiva. Vediamo ora qualcuno dei metodi utilizzati. (vedi slides 74-75).

Un metodo è l'utilizzo di una CNN per fare deblurring: si può allenare una rete neurale per fare il deblurring. La stessa architettura è usata anche per la superrisoluzione (image processing task che ha lo scopo di creare un output con risoluzione più alta rispetto all'input con lo stesso contenuto percettivo e semantico). In questo caso l'approccio è di tipo "black-box" poiché l'operazione è semplicemente imparata dai neuroni, non si applica una procedura che la esegue.

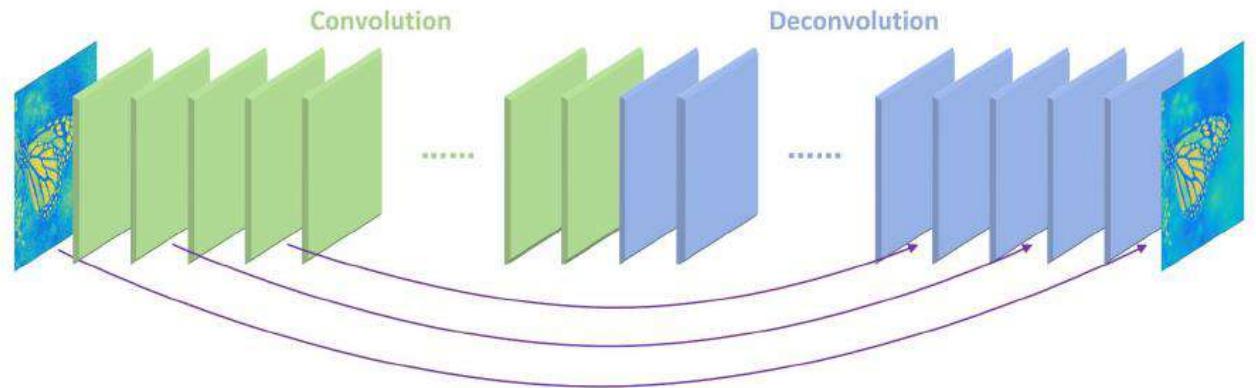


Un altro metodo prevede l'utilizzo di autoencoders.

E' lo stesso metodo che viene utilizzato per altri tasks come denoising, superrisoluzione, JPEG deblocking, non-blind image deblurring e image painting. In questo caso specifico l'autoencoder non fa compressione,

viene sempre utilizzata la stessa dimensione. Per la ricostruzione di immagini a basso livello è preferito non utilizzare né pooling né unpooling perché il pooling scarta dettagli che in questo caso possono essere utili.

Come in VGG, la dimensione del kernel sia per la convoluzione che per la deconvoluzione è settata a 3x3 ed ha mostrato ottime performance. La dimensione dell'immagine in input può essere arbitraria poiché la rete esegue essenzialmente una previsione basata sui singoli pixels, non sull'intera immagine. Vengono usati 64 feature maps per i livelli di convoluzione e deconvoluzione e 20 layers.

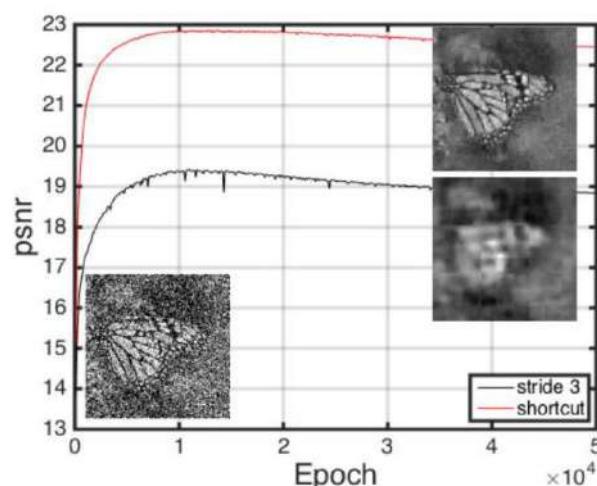


Una singola rete allenata per diversi tasks simmetrici.

La cosa interessante di questa rete è che con alcuni salti di connessione tra livelli consecutivi della rete le immagini di input e di output della rete hanno la stessa dimensione $w \times h \times c$. Infatti come si può notare dall'immagine in certi livelli di ricostruzione non si utilizza come input l'output del livello precedente ma dati provenienti da livelli più lontano. In questo paper è utilizzato $c=1$ nonostante lo stesso concetto si possa applicare anche ad immagini con più canali. Abbiamo capito che l'utilizzo di 64 feature maps per i livelli di convoluzione e di deconvoluzione ci consente di raggiungere risultati soddisfacenti nonostante l'utilizzo di più feature maps ci condurrebbe a performance leggermente migliori. Con tale architettura esistono due tipologie di reti, una con 20 livelli e una con 30 livelli.

Saltare le connessioni è importante per i dettagli. Questa rete viene allenata utilizzando come loss il mean square error MSE con distanza euclidea:

$$\mathcal{L}(\Theta) = \frac{1}{n} \sum_{n=1}^N \|\mathcal{F}(X_i; \Theta) - Y_i\|_F^2.$$



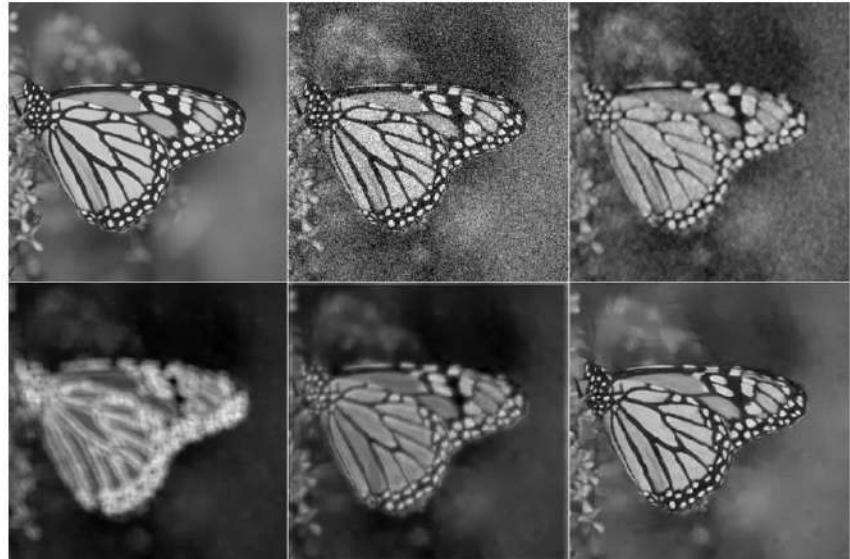
Short cut significa saltare connessioni e il grafico mostra la differenza fra l'immagine ricostruita dall'autoencoder saltando le connessioni e non saltandole.

E' possibile vedere i risultati intermedi di questa rete:

The images from top-left to

bottom-right are:
clean image,
noisy image,
output of conv-2,

output of conv-5,
output of deconv-3
output of deconv-5,



Come abbiamo detto questa architettura può essere usata anche per fare deblurring.



Details of previous papers can be found on teh NIPS16 paper <https://arxiv.org/abs/1603.09056> and on <https://arxiv.org/pdf/1606.08921.pdf> too

Ma può essere anche usata per fare impainting (ricostruire lo sfondo eliminando quello che c'è davanti) e superresolution (aumentare la risoluzione dell'immagine). Esempio di impainting:

rp43dpb0az0v9t0ws5gfeleajma13zg7ng1bwk0gk9pdjyjmc4d5bv791spfq33laf13glbgukoncktziyopishweling91m8rlwyf82azohx8ynpwf4mcbaodoinrnuole9tdasrlid7828dk5gicbn265ewjtccqjw2ceyr5bo9wyhrl2axznaf881d0bq4wy7ewjossq1pu9b3tkumnd73bc1e7bs0il14plta4ufjy6ickdrikpuvyl3qnsd0fat4ncmjhpwiwum8p7ud72x7c5yv03mqstkyfzen2nyfyd9245mw6lsbfelvlp8nrgisc3lftpqxqbs36166fvpvduzzl4apokjaaotwyk59phnvgewc5df95455qq5t1388ywqj0119gywkbnn4vvs75iw65hc3kvhbub35a90zqe8zrrg3ea9f4xw93ktom97bbhu0vcxphnphus4trfgj7x91b7ptrwh4sdyq2d3mhd073vbinv3qg9erhk24ecccxr41zyh61m3f10814tachf8ltgbnx53wl6wg0v6u7edisrftv4wsp0tg1uole27yrij9wqwd7x21h5w42wheom3ifdvpgy6sb69j2xlacp691wf2k(No THROUGHFARE PLEASE)939qm2qwfrf9mlaeslkkpm9481074juluwuk26wqnvslbr8pl5onswxxt58s4rmx65wk2nef6f3wnnibr20tg8mmmkqe5it09cgugr92szjpnn3zlmmdh1nccaltz391178pikt13ahicxsmbxj6uef37awbh3lyhgvzjrc21vky0p62hiux



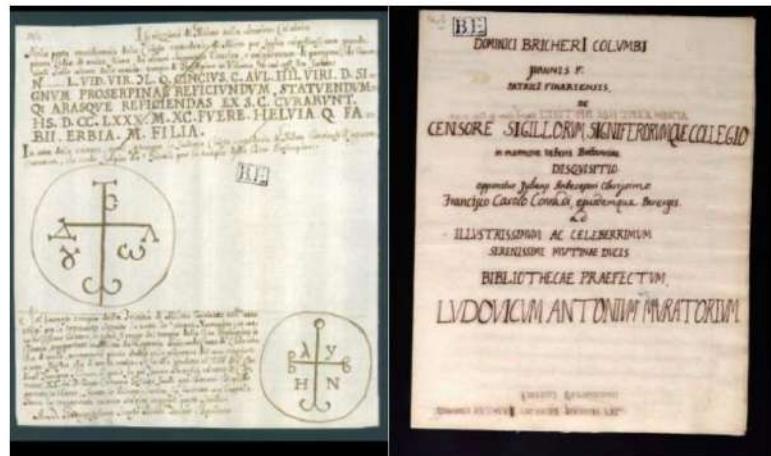
Se la rete può essere allenata in modo supervisionato si ottengono ottimi risultati ma nella maggior parte dei casi non abbiamo dati a sufficienza per trainarla in questo modo.

Vediamo alcuni esempi reali di quanto abbiamo visto fin ora:

100.000 documents of Ludovico Antonio Muratori.

1. Denoising images,
2. Image restoration
3. Then handwritten recognition

T



Hradis et al Convolutional Neural Networks for Direct Text Deblurring BMVC 2015

CNN Photo

placed by $\|w - \hat{w}\|_1$, and resulting the following function:

placed by $\|w - \hat{w}\|_1$, and resulting the following function:

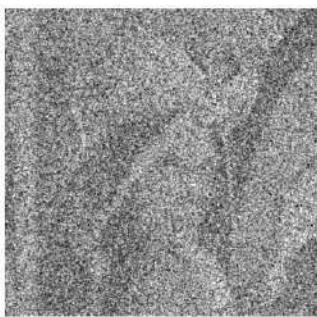
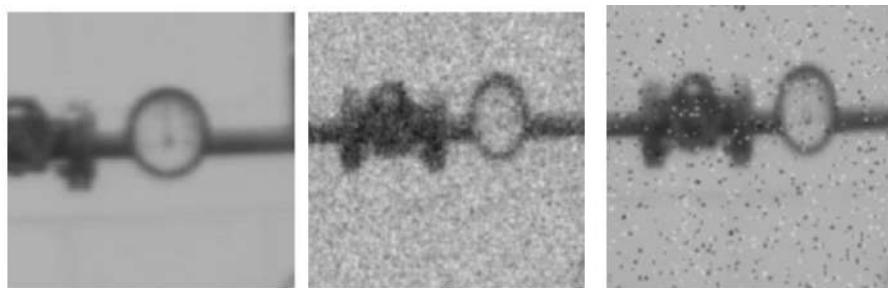
$$\min_w \|w - \hat{w}\|_1 + C \sum_i (\max(0, 1 - w_i))$$

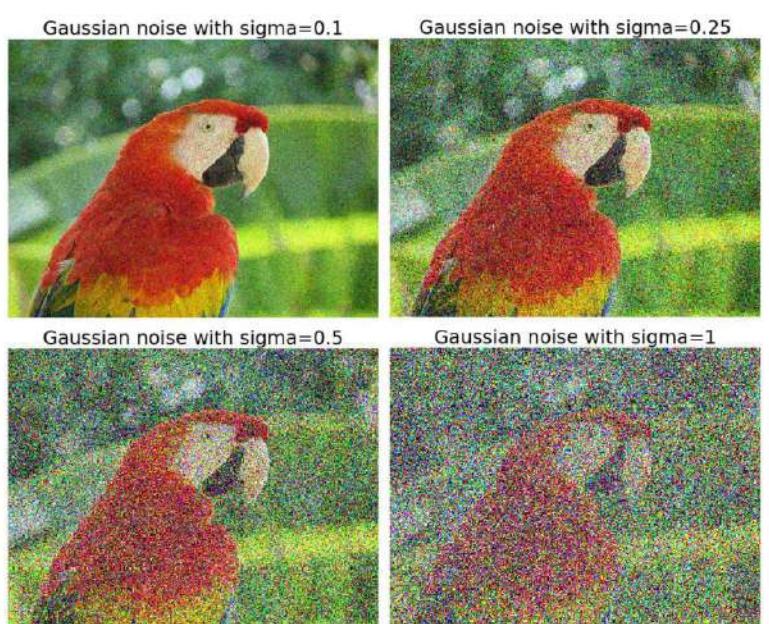
We present a simple method to deal with artifacts, we show that the proposed algorithm can also effectively process natural blurred images and low illumination images which are not handled by standard SVM.

we show that the proposed algorithm can also effectively process natural blurred images and low illumination images which are not handled by standard SVM.

We propose a SVM framework [38] so that the discrepancy between w and \hat{w} can be constrained while minimizing the classification error over D . Specifically, the regularizer $\|w\|_1$ in standard ℓ_1 -regularized linear SVM [40] is replaced by $\|w - \hat{w}\|_1$, and resulting the following object function:

the adaptive SVM framework [38] so that the discrepancy between w and \hat{w} can be constrained while minimizing the classification error over D . Specifically, the regularizer $\|w\|_1$ in standard ℓ_1 -regularized linear SVM [40] is replaced by $\|w - \hat{w}\|_1$, and resulting the following object function:





5. Edges

Questo capitolo si occuperà di analizzare la percezione visiva di umani e di sistemi cognitivi. In particolare vedremo come viene eseguita la feature extraction, quali tipi di feature esistono e la loro importanza. In una seconda parte analizzeremo più nel dettaglio la feature edge e quello che indica andando a studiare i principali algoritmi di edge detection come Sobel e Canny, vedremo in cosa consiste il contour tracing e vedremo alcune applicazioni di tali metodi.

La computer vision è la disciplina che studia **come creare modelli e algoritmi che consentono al computer di vedere** (basso livello), come **creare sistemi computazionali in grado di vedere** (medio livello) e come **creare sistemi intelligenti capaci di vedere** (alto livello).

Quanto la visione artificiale è simile alla visione umana?

Può la visione artificiale replicare la vista umana? Può la visione artificiale prendere esempio dalla visione umana e cercare di replicarla? La risposta è sì. **La computer vision si ispira nella maggior parte dei casi alla visione e alla percezione umana per creare sistemi in grado di vedere.** Infatti la “vista/visione” è parte dell’intelligenza umana mentre la visione artificiale o computer vision è parte dell’intelligenza artificiale.

L’obiettivo della visione artificiale è quello di creare sistemi che vedono meglio dell’uomo. Per fare ciò è necessario basarsi sulla vista umana e sul suo funzionamento quindi dobbiamo studiarla nel dettaglio per poterla riprodurre. In particolare lo studio di sistemi di visione artificiale si basa su due macroaree di ricerca nel campo della vista umana. Infatti la visione artificiale si basa sia sulla **psicologia percettiva e sulle scienze cognitive** per elaborare sistemi che si basano che agiscono come il cervello umano dal punto di vista dell’interpretazione di quanto viene visto, sia sulle **neuroscienze** per elaborare sistemi che abbiano la stessa struttura del cervello umano (neuroni, connessioni, corteccia cerebrale, ecc.).

Possiamo dire che i sistemi di computer vision cercano di riprodurre sia il comportamento della **mente** umana che la struttura del **cervello** umano.

La computer vision nacque negli anni 80 (partendo da sistemi in grado di fare pattern recognition) grazie a David Marr che definì la teoria della visione computazionale: “ Vision: A computational investigation into the human representation and processing of visual information” 1980”. In particolare negli anni ’80 ci furono particolari progressi in ricerche affini a questo campo che diventarono la base per proseguire la ricerca in visione artificiale. In particolare Haralick & Shapiro scrissero il libro “Robot vision”, vennero portati avanti studi su algoritmi paralleli per i supercomputers, venne elaborata una teoria di psicologia percettiva chiamata Gestalt theory e l’image processing divenne più comune.

In particolare David Marr capì che la computer vision può essere analizzata sotto tre livelli:

- **Livello computazionale:** cosa fa il sistema e perché lo fa (ad esempio quali problemi risolve e perché).
- **Livello algoritmico/rappresentazionale:** come il sistema esegue quello che deve fare e in particolare quali rappresentazioni utilizza e quali processi usa per costruire e manipolare queste rappresentazioni.
- **Livello implementativo:** come viene fisicamente realizzato il sistema (nel caso della visione biologica, quali strutture neurali e attività neuronali implementano il sistema visivo).

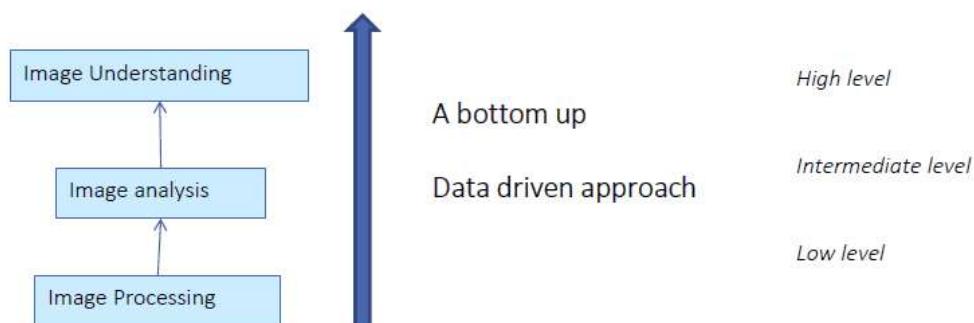
In particolare il modello illustrato da Marr nella sua teoria di computational vision era il seguente:

1. Inizialmente viene creata un’immagine 2D con i valori di intensità di ogni punto (pixel).
2. Poi viene realizzato uno sketch primordiale della scena basato sull’estrazione delle caratteristiche dei componenti fondamentali della scena come bordi, regioni, ecc.

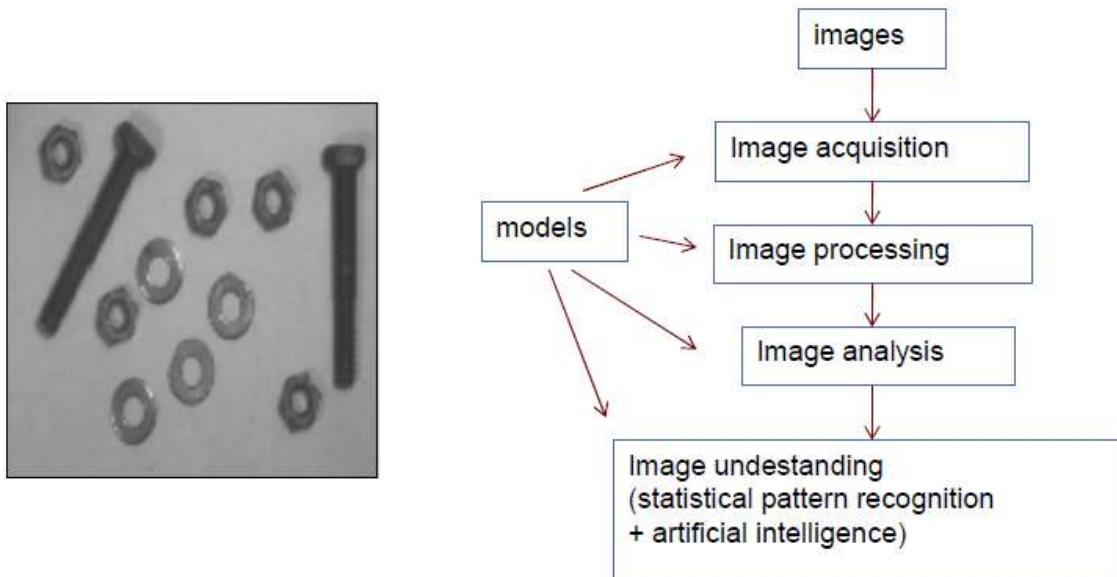
3. Poi viene realizzato uno schizzo 2.5D della scena in cui vengono riconosciute le textures, ecc. Questo sketch rappresenta il fatto che nella realtà non vediamo tutto ciò che ci circonda ma costruiamo la visione tridimensionale dell'ambiente centrata dal punto di vista dello spettatore. (vista tridimensionale viewer-centred).
4. Infine viene costruito il modello 3D in cui la scena viene visualizzata attraverso una mappa 3D e continua costruita da un punto di vista centralizzato.

In quegli anni gli studi sulle neuroscienze e sulla struttura del cervello erano ancora agli albori e le conoscenze sulla corteccia e sul funzionamento della vista erano limitate.

Sulla base del modello illustrato da Marr, uno degli approcci della computer vision (usato ancora oggi) è il seguente (è un approccio fortemente vincolato all'hardware, orientato all'applicazione e la conoscenza è incorporata nelle attività):

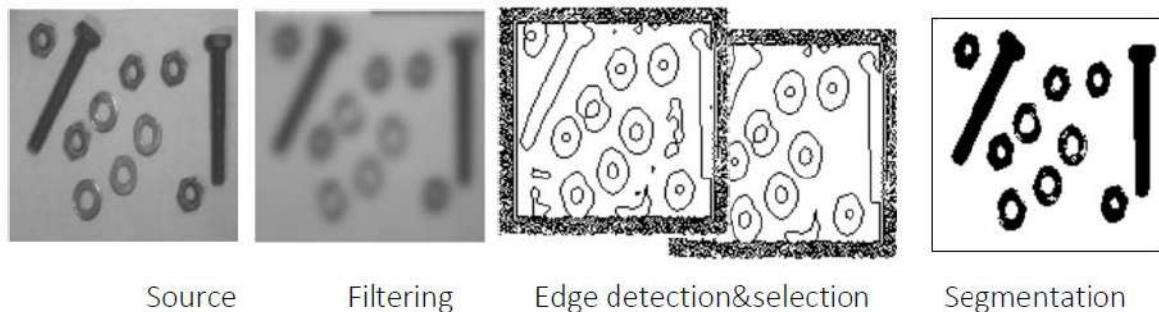


Vediamo un esempio di applicazione concreta di questo modello in un robot che deve essere in grado di capire quanti oggetti sono presenti, in quale posizione e di che tipo sono per poterli raccogliere:



Prima fase: image processing

1. Low level vision: the most classical pipeline



Rielaboro l'immagine al fine di ottenere qualcosa più facilmente analizzabile.

Seconda fase: Image analysis

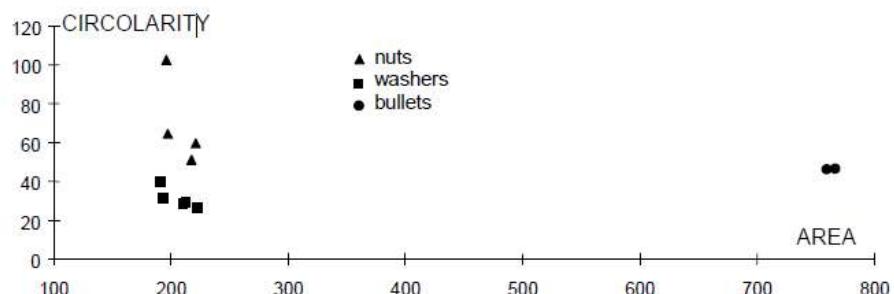
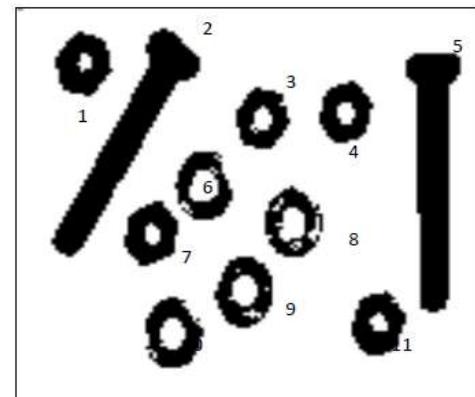
Questa fase consiste nel labeling e nel feature extraction. Le feature estratte costituiscono un feature space e in questo feature space viene costruito un feature vector per ognuno degli oggetti in modo tale che queste features descrivano al meglio gli oggetti. Poi la classificazione viene fatta sulla base di questi feature vectors. E' molto difficile scegliere le giuste features, quelle più discriminative. Lo vedremo meglio in seguito. Vengono anche contati gli oggetti in questa fase.

2. Image analysis: Labeling; feature extraction

Visual feature extraction

- Haralick circularity
- area (8-connection)

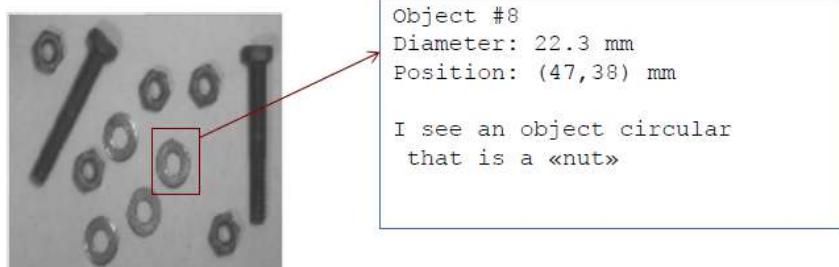
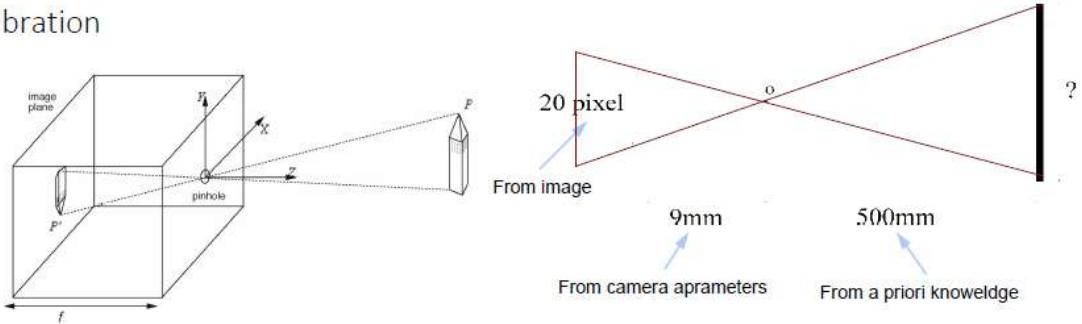
labeling



Terza fase: geometria in computer vision

Per raccogliere gli oggetti è necessario sapere la loro posizione precisa, quindi le loro coordinate. Inoltre vengono reperite altre informazioni relative alla geometria presente nella foto come la distanza fra gli oggetti.

3. Camera calibration



Quarta fase: classificazione

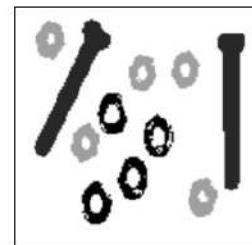
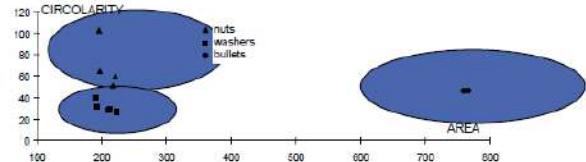
Sulla base delle informazioni estratte, gli oggetti vengono classificati. Solitamente la classificazione avviene in modo non supervisionato raggruppando gli oggetti sulla base del loro feature vector.

3. High level vision

clustering, unsupervised classification

a typical pattern recognition task, using

- Bayesian networks
- Clustering
- K-means, SVM
- Neural networks
- Genetic algorithms

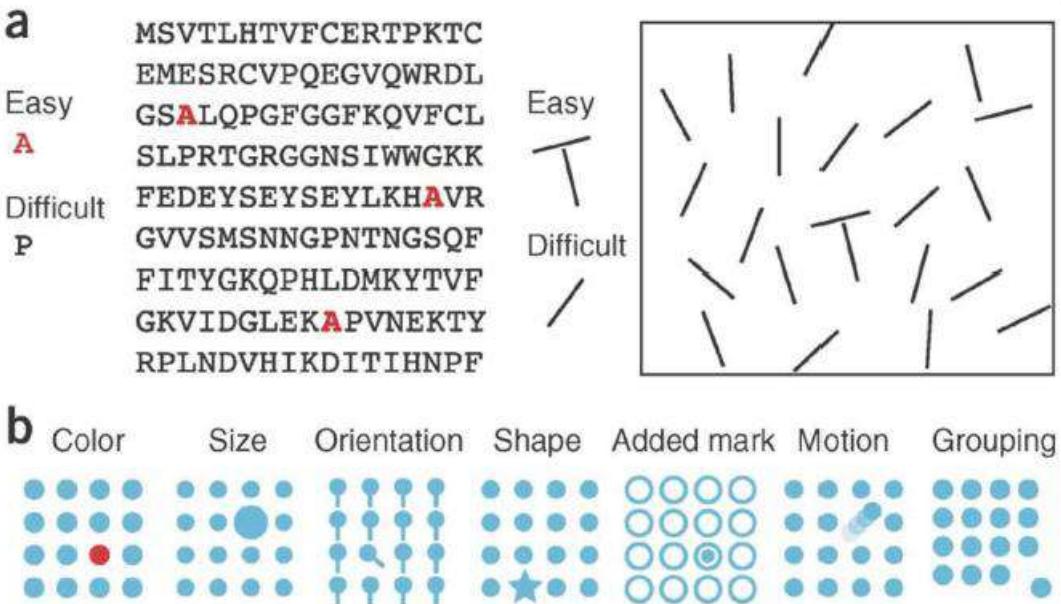


Spesso il punto più difficile di questa pipeline è proprio trovare quali sono le giuste features per rappresentare gli elementi da classificare. Questo perché spesso non conosciamo il contenuto delle immagini quindi non sappiamo quali sono le features che possano discriminare meglio gli oggetti raffigurati.

Se abbiamo a disposizione un dataset di esempio annotato possiamo allenare il modello su quel dataset così poi sarà in grado di classificare. Tuttavia nella maggior parte dei casi non abbiamo a disposizione molti esempi annotati quindi dobbiamo essere bravi a trovare le features giuste per fare learning non supervisionato.

Andiamo ora a vedere quali sono le migliori features da estrarre in accordo con studi di scienze cognitive e psicologia percettiva.

In particolare l'uomo è in grado di riconoscere una forma in base a diverse caratteristiche, come colore, dimensione, orientazione, forma, segni aggiunti, movimento o raggruppamento, più o meno di immediata percezione.



Possiamo dire anche che le features hanno pop-out diversi, ovvero diversa immediatezza di percezione. Ci sono features che il cervello umano riesce a cogliere immediatamente e sono “le prime che saltano all’occhio” e features invece per cui serve un po’ più di tempo prima che il cervello le noti. Infatti il nostro cervello è più sensibile al riconoscimento di qualcosa che è molto diverso, come ad esempio la A rossa, rispetto a qualcosa che è abbastanza simile, come le stanghette.

In particolare ci sono alcune features che saltano all’occhio subito (vedi A rossa) proprio perché il nostro cervello è in grado di processarle in parallelo, ovvero saltano all’occhio guardando tutta l’immagine nel suo complesso. Se nell’immagine delle lettere dovesse, ad esempio, trovare le lettere P dovrei processare l’immagine in serie guardando lettera per lettera per capire se è una P o meno. Questo non lo devo fare per trovare le A rosse perché saltano all’occhio subito. Quindi, ritornando alla visione artificiale, nelle immagini si possono trovare caratteristiche che possono essere processate in serie e caratteristiche che possono essere processate in parallelo.

In accordo con quanto spiegato fin ora la **feature extraction** è quel processo che, data un’immagine, estrae un vettore n-dimensionale che rappresenta alcune proprietà visive. Le features devono essere tali da riassumere correttamente il contenuto visivo dell’immagine. La feature extraction può essere assimilata ad un problema di equalizzazione o compressione. Quando si esegue è necessario sia generalizzare i dati al fine di evitare overfitting che rispettare vincoli computazionali. La feature extraction, come detto prima, è la procedura di base per molti algoritmi di computer vision.

Per progettare un sistema di visione artificiale innanzitutto bisogna capire che cosa si vuole vedere e quali sono le caratteristiche visive di cui necessiti per interpretare (classificare) quello che stai guardando. I criteri di scelta delle caratteristiche visive sono i seguenti:

- **Proprietà discriminante:** ogni feature deve assumere valori diversi per oggetti che appartengono a classi diverse. Se due oggetti appartengono a classi diverse la stessa caratteristica deve assumere due valori significativamente diversi per quei due oggetti.

- **Affidabilità:** ogni feature deve assumere valori simili per oggetti che appartengono alla stessa classe. Se due oggetti appartengono alla stessa classe la stessa caratteristica deve assumere valori simili per quei due oggetti.
- **Proprietà di indipendenza:** ogni feature deve essere indipendente dalle altre.
- **Proprietà di minima cardinalità:** Si devono scegliere meno features possibili.

Sempre intorno agli anni '80 la computer vision iniziò a considerare anche la teoria di Gestalt elaborata nel 1920 al fine di ottenere ulteriori informazioni sulle visual features e su come esse vengono recepite dal cervello umano. La teoria di Gestalt studia il modo in cui il cervello umano è capace di acquisire e mantenere percezioni significative in un mondo apparentemente caotico. Il principio centrale di tale teoria è che la mente forma un insieme globale con tendenze auto-organizzative.

La Gestalt-Forma rappresenta l'attitudine a organizzare le sensazioni elementari in figure emergenti da uno sfondo. Si ottiene, in questo modo, una figura dai contorni dettagliati, che affiora in maniera netta rispetto a uno sfondo indifferenziato, che in alcuni casi appare impercettibile. Consideriamo una serie di stimoli visivi fissi, distaccati tra loro da una manciata di secondi, che producono in noi la percezione di un solo elemento che si muove nello spazio. E' un fenomeno che a tutti capita di percepire e sperimentare soprattutto quando si è in viaggio e si osserva un'immagine fuori dal finestrino del treno o dell'auto. Questo processo è stato descritto per la prima volta da Wertheimer, uno dei capisaldi della Gestalt, che lo definì fenomeno del Phi o della persistenza percettiva degli oggetti. Quindi, l'oggetto è percepito nella sua totalità prima delle singole parti da cui è composto. Si ottiene in questo modo una figura strutturata e organizzata che diventa l'unità di misura della percezione stessa, chiaramente in relazione all'ambiente in cui si è immersi. Famose in questo ambito sono le figure geometriche ambigue, il cubo di Necker, che varia a seconda di come è percepito dal soggetto, il vaso di Rubin o la donna di Leavitt, figure aperte (senza margini), che gli occhi sono in grado di percepire come chiuse (con i bordi uniti) ovvero nella loro totalità e non come costituite da parti aperte.

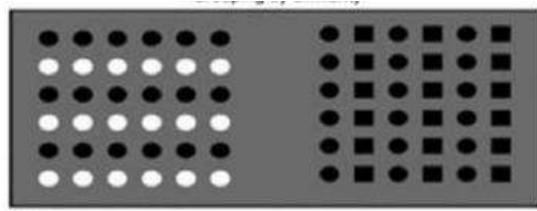
Per saperne di più: <https://www.stateofmind.it/2016/03/gestalt-teoria-terapia/>

Gli psicologi della Gestalt hanno identificato specifici principi di organizzazione percettiva:

- **Prossimità, vicinanza o similarità spaziale:** All'interno di una stessa scena o immagine, gli elementi vicini tra loro vengono percepiti come un elemento unitario.
- **Similarità:** Il cervello umano tende a mettere in relazione oggetti simili tra loro in modo automatico. Li può classificare per forma, per colore o per dimensione.
- **Continuità:** elementi simili per forma, colore o dimensione, posti uno dietro l'altro (anche secondo schema casuale) danno un'idea di unitarietà molto forte.

❖ Grouping by similarity:

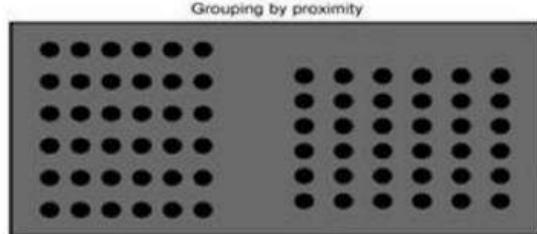
White dots grouped with white dots, squares with squares



(a)

❖ Grouping by proximity:

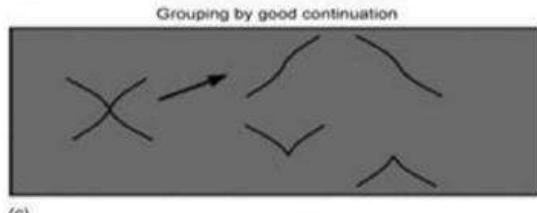
Here we perceive two separate groups of dots that are near each other



(b)

❖ Grouping by good continuation:

*On the left we perceive a single object.
When the same lines are separated we do not*



(c)

Grouping by good continuation

5

Applicando i concetti della Gestalt alla computer vision ricaviamo due proprietà aggiuntive fondamentali che le visual features devono avere:

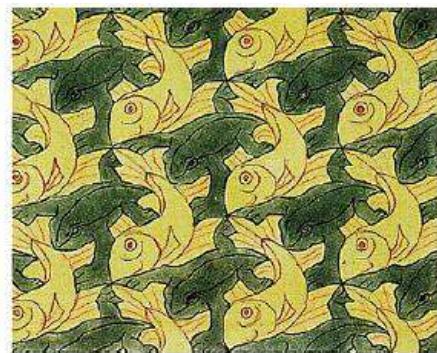
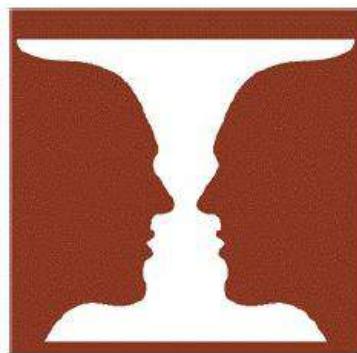
- **Proprietà invariantiva rispetto alle operazioni geometriche:** Il valore della feature non deve cambiare in seguito a scaling (traslazione in scala), contrazione geometrica, espansione, dilatazione, riflessione, rotazione, taglio, trasformazioni di similarità e tutte le loro combinazioni.
- **Proprietà di rilevanza soggettiva:** cioè la features assume più o meno importanza in base a all'obiettivo posto e a quanto quella feature può essere utile per raggiungere quell'obiettivo. Inoltre esprime l'indipendenza della feature rispetto la variazione di luminosità, rispetto allo sfondo, rispetto al campo visivo, ecc.

Ad esempio, quali sono le features che possono considerare per definire questi oggetti? E' molto difficile in questo caso scegliere caratteristiche che soddisfano tutte le proprietà elencate sopra a mano. In questo caso è meglio utilizzare il deep learning che sarà in grado da solo di estrarre e di riconoscere gli oggetti (riconoscerli a mano in questo caso è troppo difficile).



Riassumendo quanto visto fin ora, la computer vision mira a calcolare caratteristiche visuali che mostrano più somiglianza percettiva, somiglianza spaziale e somiglianza nella continuità (anche temporale) possibile in modo da poter classificare nel miglior modo possibile. Le possibili features sono colore, forma (contorni, geometria, bordi), struttura (continuità e similarità spaziale), shape 3D (profondità), motion (continuità nel tempo) e se possibile loro combinazioni e salienza.

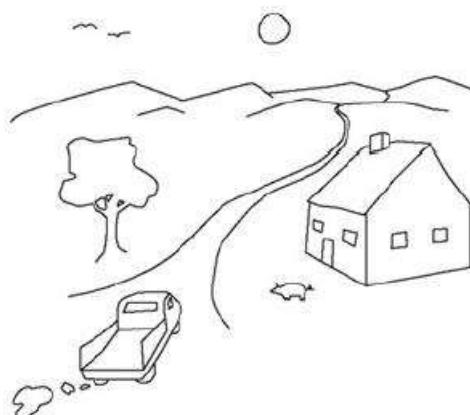
Andiamo ora a riprendere la teoria di Gestalt per analizzare una delle features più significative, ovvero i bordi, infatti gli psicologi della Gestalt hanno anche sottolineato l'importanza del fatto che il cervello umano riesce ad associare più parti della scena fra loro percependo così gli oggetti da queste parti rappresentati e riesce ad associare altre parti fra loro per andare a percepire lo sfondo dell'immagine. Quindi il cervello è in grado di fare la distinzione fra parti che compongono lo sfondo e parti che compongono gli oggetti. La separazione fra figura e sfondo può essere continua e dinamica, come nel seguente caso:



L'occhio e la mente umani non possono essere occupati nella percezione di due cose allo stesso momento, quindi deve esserci un salto veloce e continuo da una parte all'altra, ossia fra sfondo e oggetto. Per esempio i contorni nel campo visivo segnalano la presenza dei bordi degli oggetti quindi ci aiutano a percepire gli oggetti.

Possiamo dire che per riconoscere gli oggetti rappresentati sono molto importanti i **contorni** e i **bordi** (feature molto importante da estrarre, vedremo fra poco come) infatti, in accordo con la teoria di E. Kandel "Elements of neuroscience", per riconoscere gli oggetti in disegni a matita come quello in figura sono sufficienti i bordi senza colori e sfumature.

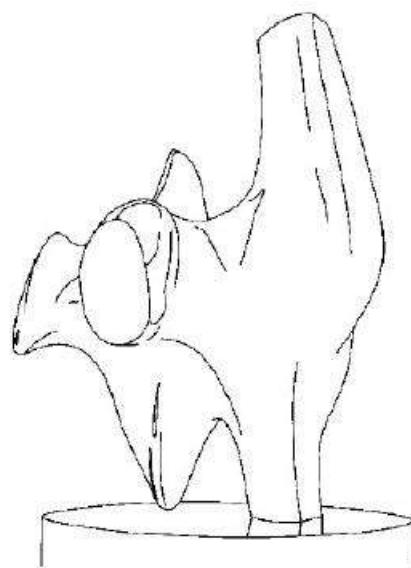
(specifichiamo un concetto che ci tornerà utile dopo, i contorni riguardano l'immagine nel suo complesso mentre il bordo è una proprietà locale, riguarda una piccola porzione dell'immagine).



Ricreare un disegno come quello in figura a partire da un'immagine è sia una compressione del dato ma anche una feature extraction dove la feature che stiamo estraendo sono proprio i bordi.

Inoltre i bordi esprimono anche le relazioni spaziali presenti fra gli oggetti che ci aiutano ad interpretare il contenuto dell'immagine.

Analizzando questo aspetto dal punto di vista della visione artificiale, le forme o gli oggetti vengono percepiti grazie alla variazione della luminosità (si intende variazione brusca, discontinuità del valore assunto da pixels adiacenti). Infatti per riconoscere i contorni di un oggetto si cerca di comprimere l'immagine in modo tale da evidenziare soltanto le forti variazioni di luminosità che appaiono in essa.



Questa è un'operazione relativamente facile per gli umani ma difficile per i sistemi artificiali poiché i contorni si presentano al confine tra regioni omogenee ma spesso la segmentazione globale è difficile quindi essi vengono costruiti analizzando la variazione a livello locale. (in parole povere prima si identificano i bordi presenti in una porzione dell'immagine per volta dopo di che si mettono insieme tutti i bordi trovati in tutte le porzioni per costruire i contorni).

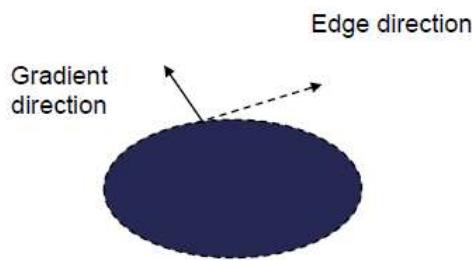
Dal punto di vista computazionale quali sono le operazioni che posso fare per estrarre i bordi degli oggetti rappresentati nelle immagini?

Iniziamo con la definizione di bordo o edge. Il **bordo** è una proprietà locale di un pixel e del suo intorno che esprime la presenza una rapida variazione dell'intensità dell'immagine. Può anche essere interpretata al contrario, ovvero esprime la posizione in cui si è verificata una rapida variazione dell'intensità.

Specifichiamo che il confine (border) è una proprietà di una regione mentre il bordo (edge) è una proprietà locale. Possiamo calcolare i confini selezionando i bordi più forti/alti.

Più tecnicamente il bordo è **un vettore con un modulo (magnitude) e una direzione** che dipende dalla variazione di luminanza (o intensità).

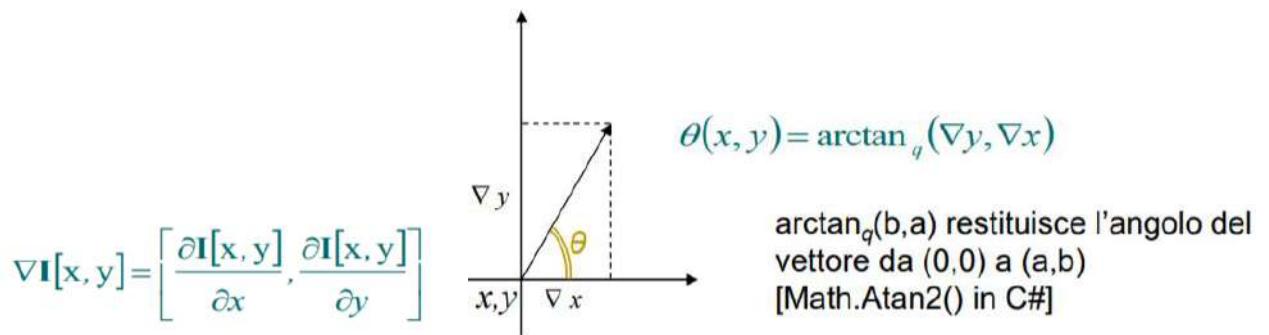
L'idea è quella di verificare se c'è o meno una continuità dei valori dei pixels nell'intorno derivando la funzione e studiando il comportamento della derivata. Possiamo quindi calcolare la variazione di luminanza o intensità come un gradiente. Il bordo ha la direzione perpendicolare alla direzione del gradiente.



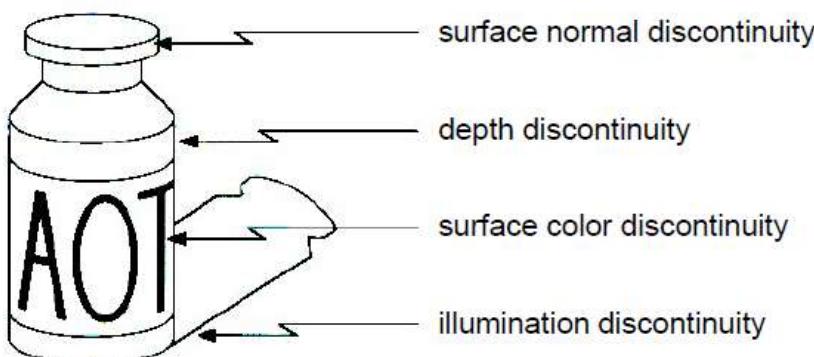
Per capire meglio questi concetti facciamo un passo indietro e ripassiamo il gradiente e il suo significato geometrico. (vedi qualche spiegazione di analisi 2 per avere definizioni matematiche).

La derivata di un segnale denota la sua variabilità. A fronte di forti variazioni locali (i.e. contorni e altri bruschi cambiamenti di intensità) la derivata assume valori elevati mentre se il segnale è costante la derivata è zero. Nel caso di segnali bidimensionali (come le immagini), si devono considerare le derivate parziali lungo asse x e y. Il gradiente è il vettore le cui componenti sono le derivate parziali nelle diverse direzioni (2 nel caso di immagini).

L'orientazione del vettore gradiente in un punto indica la direzione di maggior variazione d'intensità in quel punto dell'immagine. La direzione dell'edge è ortogonale all'orientazione del gradiente:



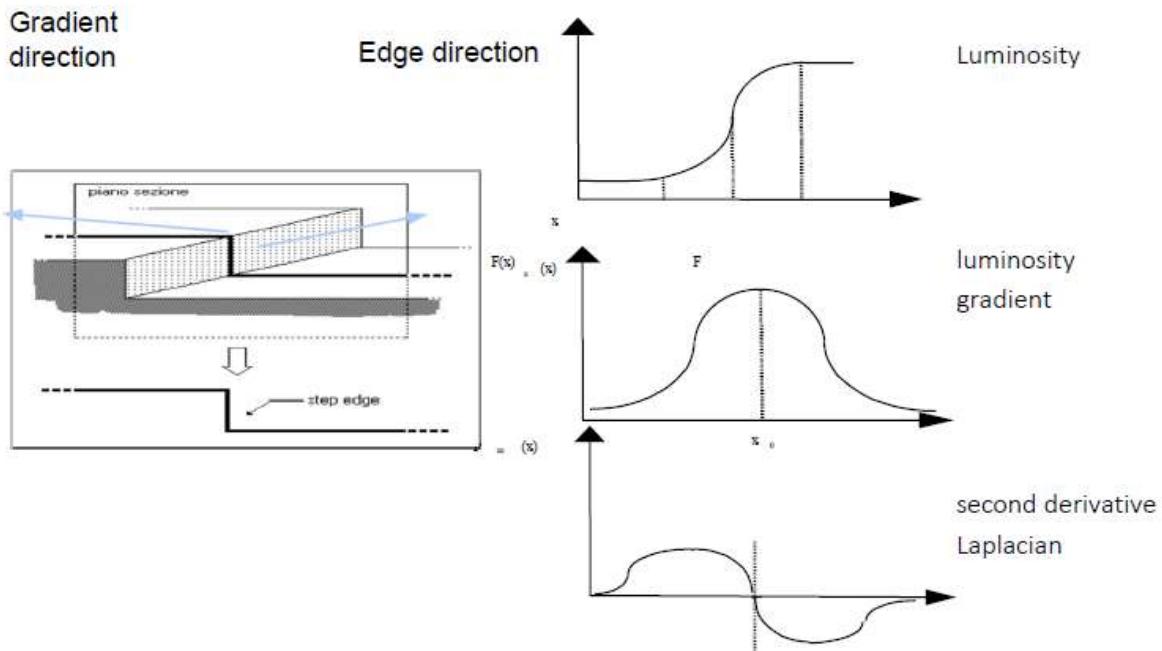
In realtà, essendo il bordo una discontinuità della funzione $f(x,y)$ che rappresenta l'immagine, non sempre è dovuto alla presenza di un contorno ma può presentarsi anche per altri motivi, come per discontinuità dovute al colore, alla superficie o all'illuminazione o alla profondità.



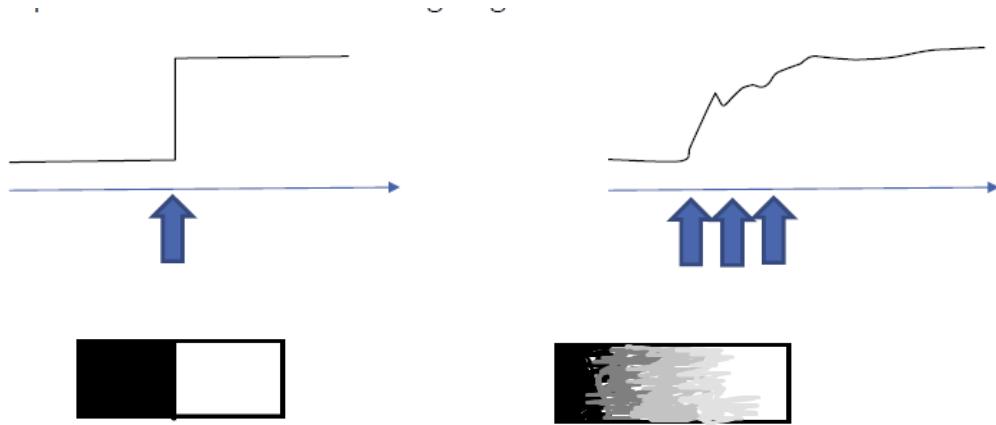
Per questa ragione, per trovare i contorni andiamo a considerare solo gli **“strong edges” cioè i vettori gradiente che hanno modulo (magnitude) più alto**. Infatti il gradiente misura la variazione dei pixels dell'immagine e più brusca (o grande, o maggiore discontinuità) è la variazione più grande è il modulo o

magnitude del gradiente. Per trovare la variazione dei pixel massima mi basta trovare il massimo valore del gradiente (del modulo o magnitude del gradiente) oppure i punti in cui si annulla la derivata seconda dell'immagine.

(massimo modulo e direzione dell'edge perpendicolare a quella del gradiente.)



Poiché il nostro obiettivo è quello di trovare i contorni dobbiamo considerare solo gli edges dovuti appunto ai bordi, quindi dovremo considerare solo quelli che hanno un “elevato gradiente” mentre si possono trascurare gli altri.



L'algoritmo di **BORDER detection** è il seguente:

- 1) Uso un operatore di **edge detection** per trovare gli edge
- 2) Seleziono solo i “**bordi forti**” in base a diversi criteri (applico una soglia al modulo del gradiente per tenere solo i punti in cui è elevato)
- 3) **Collego** e metto insieme gli edges trovati (annotazione)

Border detection:

- 1) Use of an edge detection operator (edge detector)
- 2) Selection of strong edges with some given criteria
- 3) Linking the edges (labeling)

Il problema che si può verificare applicando questo algoritmo è che il rumore può generare falsi edge.

Andiamo a vedere quali **algoritmi di EDGE detection** possono essere applicati:

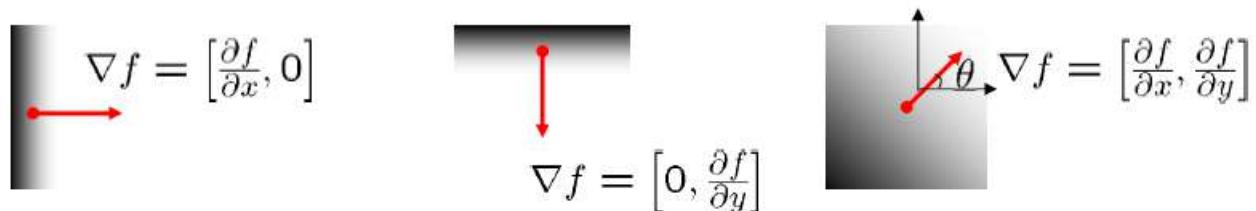
- 1) Metodi basati sul **calcolo della derivata prima**: In genere l’individuazione dei bordi si avvale di **filtri derivativi**. Il valore in ogni punto rappresenta una stima numerica del gradiente nel pixel corrispondente dell’immagine.
- 2) Tecniche sempre basate sul primo metodo con in aggiunta tecniche di **regolarizzazione che usano filtri e maschere ottimali**.
- 3) Tecniche che seguono i contorni a livello locale basate su operazioni che coinvolgono l’intorno di **bordi annotati seguite da segmentazione**.
- 4) **Classificazione** utilizzando il deep learning

Vediamo più nel dettaglio le tecniche che prevedono il calcolo della derivata prima:

Il gradiente di una funzione 2D continua $f(x,y)$ è:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

e la direzione del gradiente è (punta nel verso in cui c’è massima variazione di valore dei pixels):



$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

La **direzione del gradiente** è perpendicolare alla **direzione del bordo**. La “**forza**” del bordo corrisponde alla **magnitude del gradiente**:

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

In realtà le immagini sono funzioni discrete, non continue, quindi dobbiamo trovare il modo per calcolare la derivata di una funzione discreta. Ci sono tre possibili approssimazioni che possono esse scelte:

- Forward difference

$$\Delta_h[f](x) = f(x + h) - f(x).$$

- Backward difference

$$\nabla_h[f](x) = f(x) - f(x - h).$$

- Central difference

$$\delta_h[f](x) = f(x + \frac{1}{2}h) - f(x - \frac{1}{2}h).$$

Nel nostro caso sceglieremo la **central difference** e, data la funzione $f(x,y)$ e la sua approssimazione discreta $f(r,c)$, la calcoliamo così:

$$\frac{\partial f(r,c)}{\partial r} = f(r+1,c) - f(r-1,c)$$

$$\frac{\partial f(r,c)}{\partial c} = f(r,c+1) - f(r,c-1)$$

Si può dimostrare che queste espressioni possono essere calcolate facendo la convoluzione fra l'immagine e le seguenti maschere:

$$\begin{matrix} & 0 & -1 & 0 & & 0 & 0 & 0 \\ & 0 & 0 & 0 & & -1 & 0 & 1 \\ (y) & 0 & 1 & 0 & & 0 & 0 & 0 \end{matrix} \quad \begin{matrix} & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ (x) & & & & & & \end{matrix}$$

Quindi la convoluzione con questo tipo di filtro restituisce una stima del gradiente in quel punto.

Nel concreto queste maschere sono troppo sensibili al rumore quindi ne vengono utilizzate versioni un po' diverse. Per limitare il rumore può essere utilizzata la **Prewitt mask**:

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	-1

Poi si può normalizzare dividendo per 1/6.

Questa maschera è più robusta al noise perché calcola la derivata centrale in una direzione e in contemporanea applica un avarage filter nell'altra direzione per ridurre il rumore.

Tuttavia la soluzione migliore, cioè che in assoluto approssima meglio il gradiente è l'**operatore di Sobel**. Questo filtro calcola la derivata centrale in una direzione e in contemporanea applica un gaussian filter per fare smoothing e ridurre il rumore nell'altra direzione.

The *Sobel* operators below are commonly used

$$\frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

s_x

$$\frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

s_y

Still better : Frei and Chen operator

$$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$$

(e)

Vediamo alcune delle maschere esistenti a confronto:

Gradient:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

Good Localization
Noise Sensitive
Poor Detection

Roberts (2 x 2):

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Prewitt (3 x 3):

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & 1 \end{bmatrix}$$

Sobel (3 x 3)

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & 1 \end{bmatrix}$$

Sobel (5 x 5):

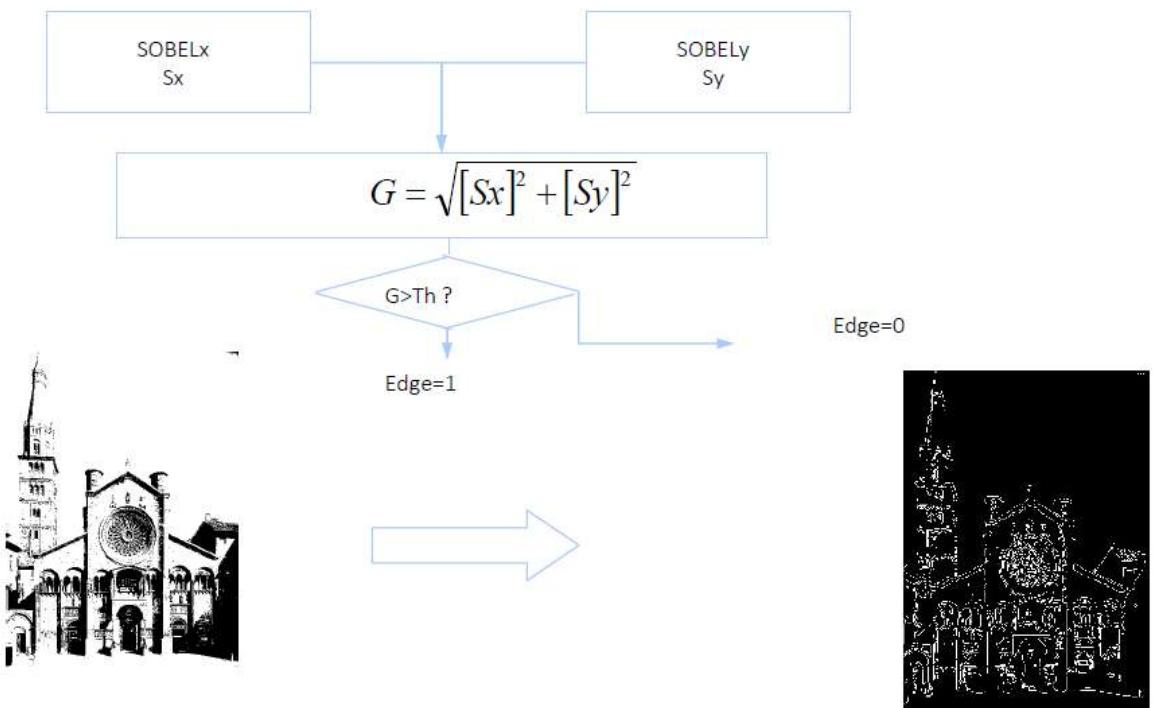
$$\begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -2 & -3 & 0 & 3 & 2 \\ -3 & -5 & 0 & 5 & 3 \\ -2 & -3 & 0 & 3 & 2 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 5 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -3 & -5 & -3 & -2 \\ -1 & -2 & -3 & -2 & -1 \end{bmatrix}$$

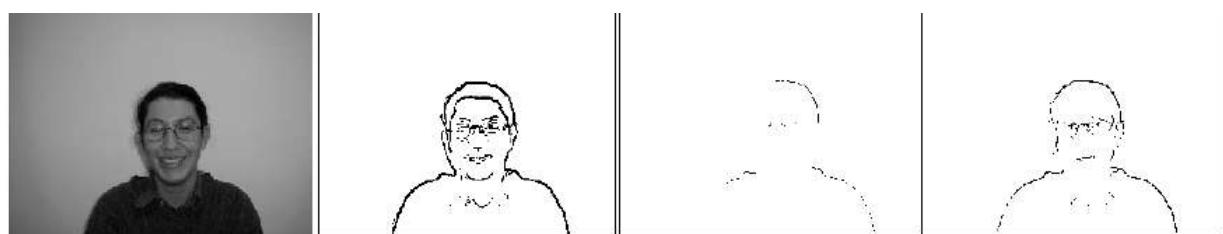
Poor Localization
Less Noise Sensitive
Good Detection

Vediamo ora l'**algoritmo di edge detector che utilizza la maschera di Sobel**: Per ogni pixel viene eseguita la convoluzione sia con la maschera S_x che con S_y , dopo di che viene calcolata la magnitudo del gradiente e vengono selezionati come edges soltanto i pixels che hanno gradiente di magnitudo superiore alla soglia. In generale la soglia si impara, è difficile impostarla a mano perché dipende molto dal tipo di immagini su cui stiamo lavorando.

For each point :



Esempi di maschere diverse a confronto:

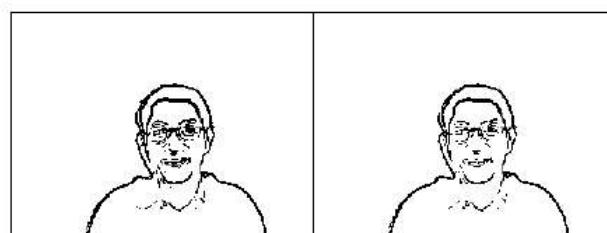


Original

Prewitt,
Th = 100

Roberts,
Th = 100

Roberts,
Th = 50

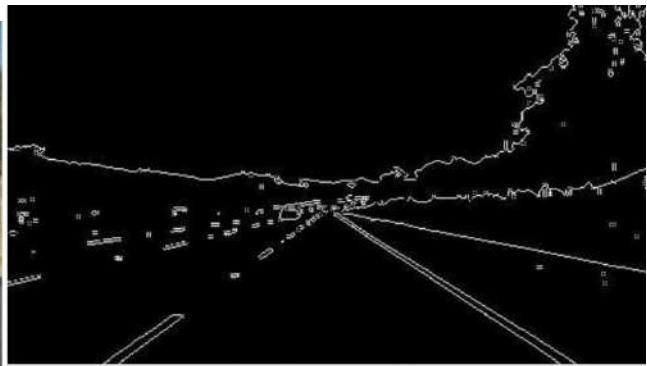


Sobel,
Th = 100

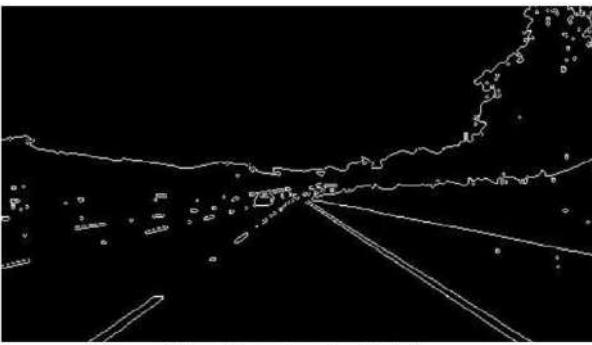
Frei and Chen
Th = 100



Original Highway Image taken from Udacity



Edge Detection by Prewitt



Edge Detection by Sobel

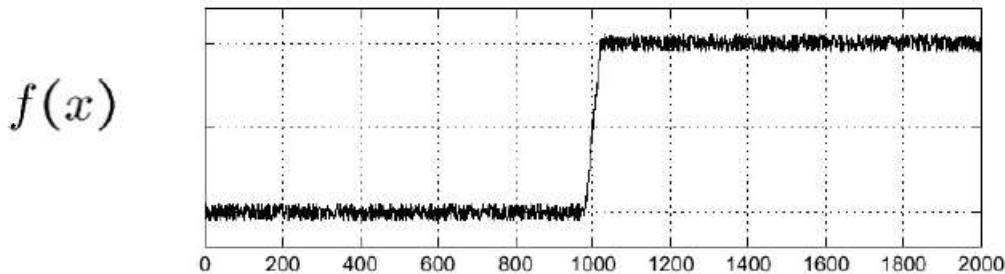
Ora analizziamo algoritmi di edge detection della seconda tipologia elencata, ovvero quelli che aggiungono tecniche di regolarizzazione e ottimizzazione.

I criteri, espressi da **Canny**, che un buon operatore deve avere sono i seguenti:

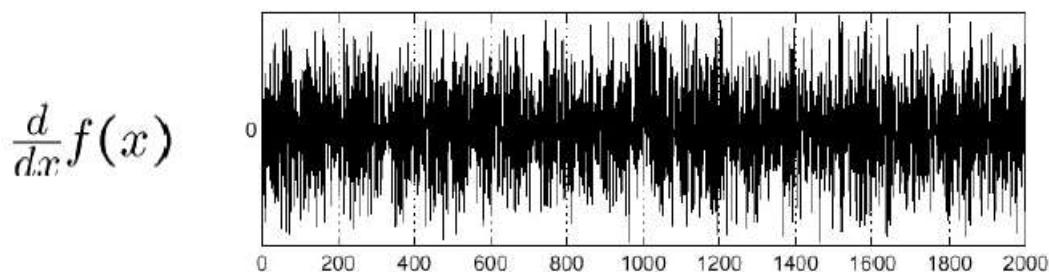
- 1) **BUON RILEVAMENTO**: si cerca di avere una bassa probabilità di errore nel riconoscere i veri bordi e i falsi bordi. Se consideriamo tutto quello che non è bordo come rumore, entrambe le probabilità sono funzioni monotone decrescenti con SNR e questo criterio corrisponde a massimizzare il signal-to-noise Ratio SNR. Più il rilevamento dei bordi è fatto bene, più alto è il SNR. (Riconoscere i bordi reali e non quelli falsi corrisponde a massimizzare il signal to noise ratio).
- 2) **BUONA LOCALIZZAZIONE**: i pixels identificati come edges dagli operatori devono essere il più vicino possibile ai bordi reali per avere una perfetta localizzazione. Più i bordi sono localizzati bene, più precisa è la loro posizione.
- 3) **UNA RISPOSTA PER OGNI SINGOLO EDGE**: Quando è presente un singolo bordo anche l'operatore dovrebbe ristornare un solo edge più vicino possibile a quello reale. Se abbiamo due risposte, una delle de è falsa.

La selezione dei bordi è un ill-posed problem, ovvero un problema per cui si cerca la soluzione migliore possibile, un determinato un trade-off ma non c'è una unica soluzione ottima.

Andiamo a vedere meglio che significa massimizzare il SNR. Consideriamo una singola riga o colonna dell'immagine e consideriamo i valori dei pixels in quella riga/colonna. Tale riga/colonna può essere vista come un segnale:



Come si può notare è presente del rumore. Se io ora calcolo la derivata di $f(x)$ ottengo una funzione che non è sufficiente per rilevare eventuali bordi.



Quando nell'immagine è presente rumore la derivata prima non è sufficiente per computare i bordi perché è troppo sensibile al rumore.

I filtri che abbiamo visto fin ora, che calcolano la derivata prima come central difference, rispondono fortemente al rumore. Poichè il rumore produce pixels che sembrano molto diversi dai loro vicini, più alto è il rumore più forte è la risposta del filtro (l'edge rilevato dal filtro).

Una possibile soluzione sarebbe quella di effettuare uno smoothing dell'immagine prima di calcolare la derivata in modo d ridurre il rumore forzando i pixels differenti dal loro intorno ad assomigliare di più ai pixels circostanti. Tuttavia invece di applicare un **filtro di smoothing e poi un filtro derivativo** è possibile applicare un **solo filtro che esegue entrambe le operazioni**. Tale filtro è chiamato **DOG** e corrisponde alla **derivata di un filtro gaussiano**. (derivo un filtro gaussiano e poi lo applico).

Tuttavia questo non è sufficiente a soddisfare tutti e tre i criteri esposti prima perché comunque non restituisce un solo edge per ogni bordo reale.

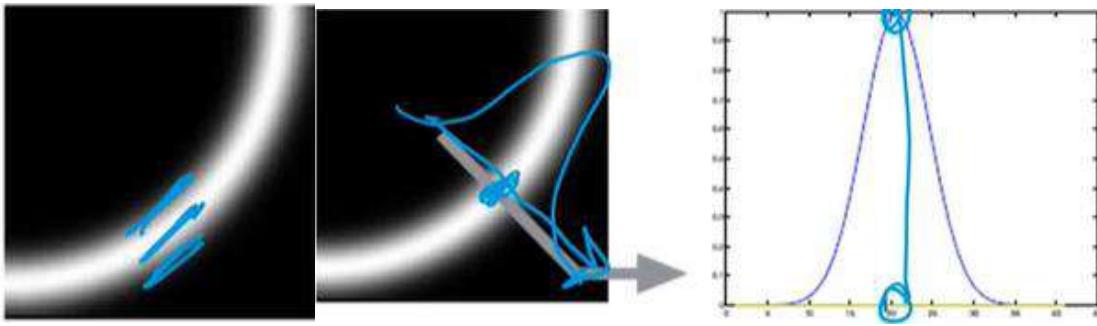
La proposta di Canny per trovare il filtro più efficiente è la seguente:

Definire il miglior filtro continuo ad una dimensione possibile e dopo discretizzarlo ed espanderlo a 2 dimensioni. Abbiamo bisogno di un filtro che:

- fa **smoothing** in modo tale da minimizzare i falsi bordi dovuti dal rumore
- Riesce a trovare la **corretta direzione del gradiente** in modo da estrarre un solo bordo
- Elimina i falsi bordi per aumentare la precisione

Il miglior filtro per fare smoothing è il **Gaussian filter**, quindi nell'algoritmo di Canny viene usato **DOG**.

Per rispondere con un solo edge invece, consideriamo il seguente esempio: se ho un'immagine di questo tipo qual è il punto che considero edge fra i possibili punti esistenti su questa linea? (quello che viene spiegato ora fa riferimento all'algoritmo Log non a Canny)



Devo trovare la reale direzione del gradiente e una volta che l'ho trovata costruisco il grafico in figura (sarebbe il grafico della derivata, per trovare il massimo della derivata prima calcolo la derivata seconda e la pongo uguale a zero.) lungo quella direzione. Il la posizione dell' edge sarà la coordinata del punto massimo di quel grafico e l'edge avrà direzione perpendicolare alla direzione del gradiente.

Algoritmo di Canny (vedi sulle altre slides, è più semplice)

1. Si fa lo smoothing dell'immagine con un filtro gaussiano 2D
2. Si trova la direzione normale \mathbf{n} agli edge locali trovati per ogni pixels che corrisponde alla direzione del gradiente.
3. Si calcola la magnitude del gradiente
4. Si trova la posizione precisa dell'edge che corrisponde al punto in cui si annulla la derivata seconda di $G*I$ calcolata nella stessa direzione del gradiente (della derivata prima). Il punto in cui si annulla la derivata seconda calcolata nella stessa direzione della direzione del gradiente corrisponde al punto in cui la derivata prima ha un massimo. Infatti **si escludono tutti gli edge tranne quello in corrispondenza del punto massimo della derivata prima**. (non-maximum suppression). (questo è quello che accade idealmente, in realtà vediamo dopo qual è il procedimento che si applica nel concreto).
5. Si applica hysteresis o thresholding

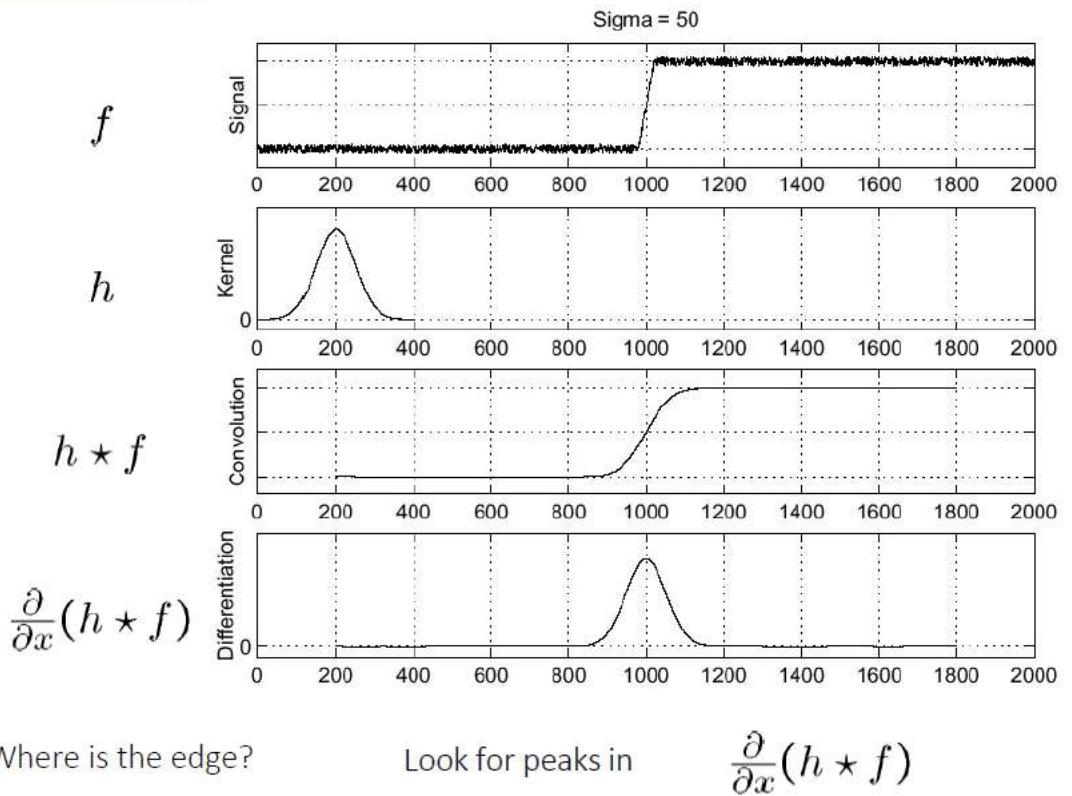
Algorithm

- 1.) Smooth image I with 2D Gaussian: $G * I$
- 2) Find local edge normal directions for each pixel
$$\mathbf{n} = \frac{\nabla(G * I)}{|\nabla(G * I)|}$$
- 3) Compute edge magnitudes $\text{Grad}(I,j)$
$$|\nabla(G * I)|$$
- 4) Locate edges by finding zero-crossings along the edge normal directions (non-maximum suppression): search for points which cross zero with second derivative
- 5) hysteresis-thresholding
$$\frac{\partial^2(G * I)}{\partial \mathbf{n}^2} = 0$$

Andiamo a vedere come realmente viene eseguito questo algoritmo. Per prima cosa è possibile applicare il filtro DOG al posto di applicare il filtro gaussiano e poi fare la derivata.

Andiamo a vedere più nel dettaglio la differenza fra fare prima lo smoothing e poi derivare e applicare il filtro Derivative of Gaussian DOG:

Solution: smooth first (doG)



Where is the edge?

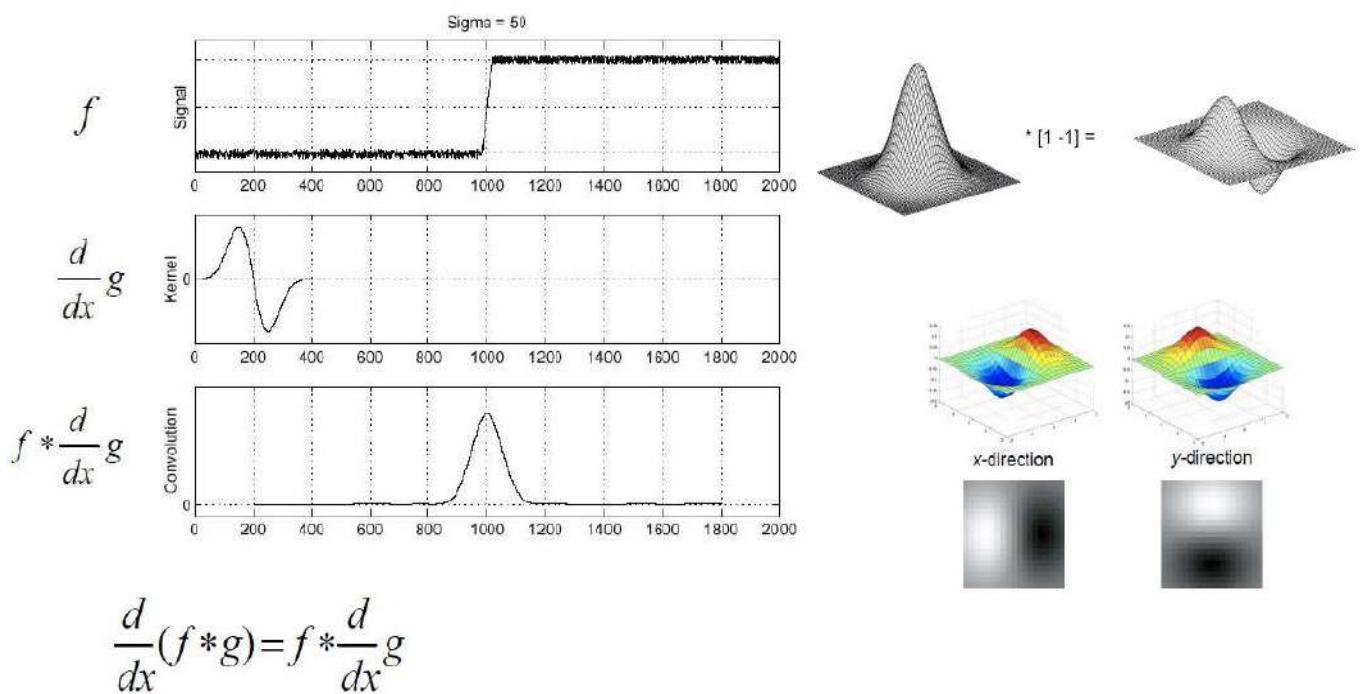
Look for peaks in

$\frac{\partial}{\partial x}(h * f)$

Dopo aver eseguito i punti 1, 2 e 3 dell'algoritmo otteniamo la direzione e la magnitudine del gradiente della convoluzione fra immagine e filtro Gaussiano. Fare la convoluzione fra l'immagine e il filtro Gaussiano e poi derivare è equivalente a fare la convoluzione fra l'immagine e la derivata del filtro gaussiano.

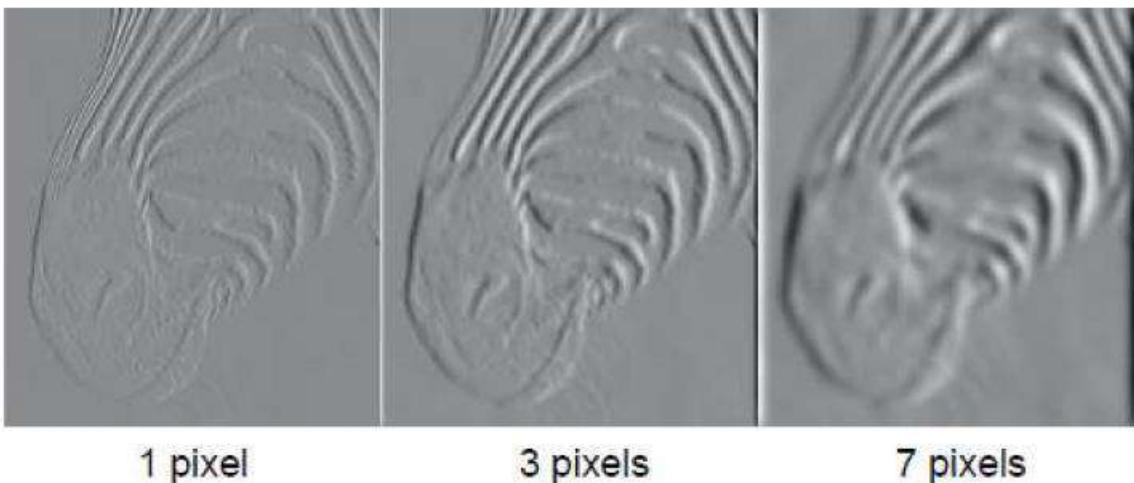
$$\frac{\partial(G_\sigma \ast I)}{\partial x} = (\frac{\partial G_\sigma}{\partial x}) \ast I$$

Nel secondo caso è meglio perché si esegue una convoluzione in meno.



Il valore di sigma del gaussian filter definito per lo smoothing è chiamato scala di smoothing. Aumentare la scala di smoothing consente di rimuovere il rumore tuttavia sfoca i bordi. A volte il multi-scale è utile.

Vediamo un esempio di Dog con sigma 1, 3 e 7:



Più sigma è alto più si rimuove rumore più vengono sfocati i bordi. Risultati con sigma differenti creano bordi spessi, noi abbiamo bisogno di bordi sottili collegati in una curva.

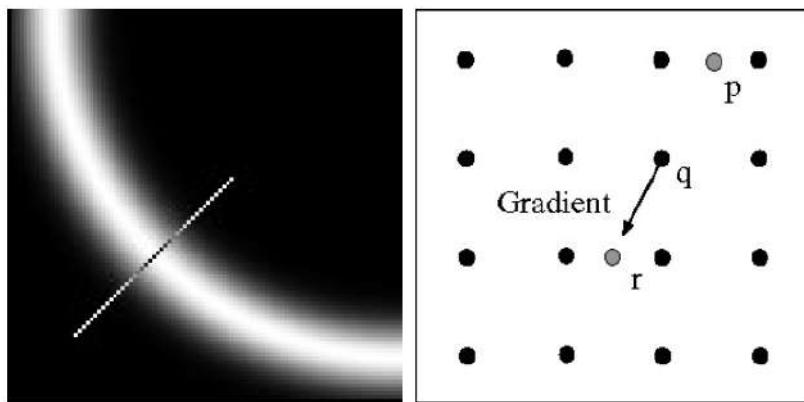
Questo è il motivo per cui dobbiamo utilizzare una strategia per selezionare un singolo bordo sottile dall'insieme spesso di bordi che otteniamo. Ad esempio in questa immagine quali bordi teniamo per ottenere un contorno sottile della zebra? Vogliamo tenere solo i bordi centrali.



E' per fare ciò che utilizziamo non-maximum suppression.

Andiamo a vedere come questo metodo è applicato nel concreto, senza veramente calcolare la derivata seconda e annullarla.

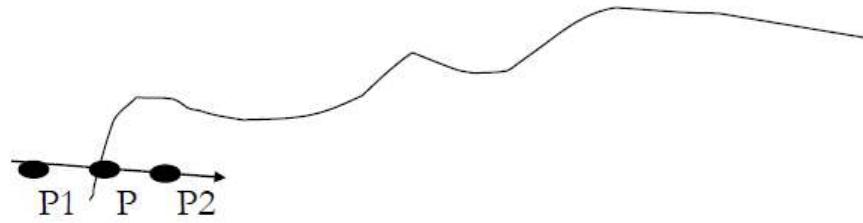
Per trovare i veri bordi verifichiamo se il pixel è il massimo locale lungo la direzione del gradiente. Quindi non selezioniamo gli edges che hanno un gradiente elevato ma solamente i massimi locali lungo le direzioni del gradiente.



Per fare ciò dobbiamo lavorare a livello di subpixels. Se mi trovo ad esempio, sul punto q e voglio trovare la direzione del gradiente devo fare l'interpolazione fra i punti nell'intorno di q. Quando trovo la direzione del gradiente confronto tutti i punti (i pixels) presenti su quella direzione e se il punto q che sto considerando ha valore massimo allora esso viene definito un edge valido.

Riassumendo, per ogni pixel candidato ad essere edge, calcolo la direzione del gradiente tramite interpolazione dei punti presenti nel suo intorno poi vado a considerare gli edges nell'intorno che sono sulla direzione del gradiente. Se il punto candidato edges, che è il punto centrale fra i punti sulla direzione del gradiente, è il punto massimo (ha il valore più alto) fra i punti lungo quella direzione, allora quello sarà considerato un edge effettivo, altrimenti viene scartato. Ripeto questo ragionamento per ogni punto candidato ad essere edge.

NON MAXIMA SUPPRESSION:



if $\begin{cases} \text{Grad } (P) \geq \text{Grad } (P_1) \\ \text{Grad } (P) \geq \text{Grad } (P_2) \end{cases} \Rightarrow P \text{ is a valid edge}$

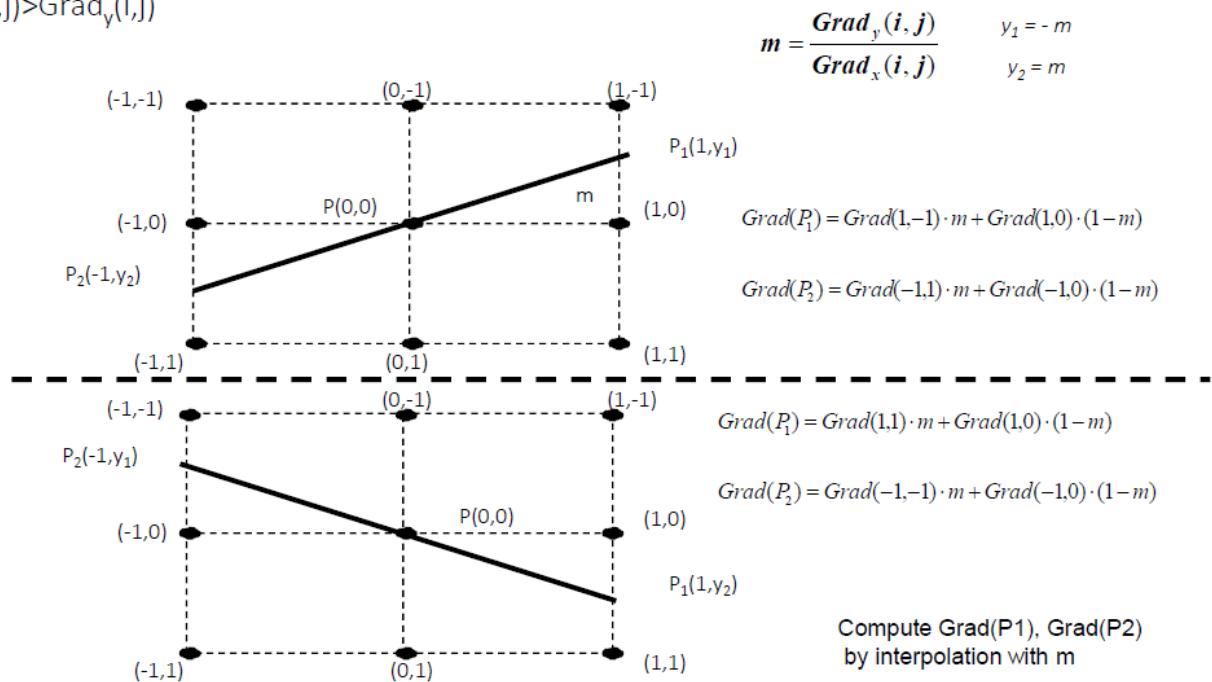
L'ipotesi sotto cui si applica questo ragionamento è che tutti i punti nell'intorno del pixel considerato abbiano la stessa direzione del gradiente. Per trovare tale direzione abbiamo bisogno di una adeguata interpolazione del gradiente dei punti dell'intorno.

Andiamo a vedere come si calcola la direzione del gradiente.

Per capire però quali sono i corretti punti dell'intorno con cui fare l'interpolazione bisogna prima trovare la pendenza del gradiente. La formula per trovare la pendenza m del gradiente cambia a seconda che il gradiente calcolato nel punto sia maggiore in x oppure in y . Nel primo grafico del seguente esempio per trovare il punto $P_1(1,y_1)$ dobbiamo fare l'interpolazione fra il punto $(1,-1)$ e $(1,0)$, mentre per trovare il punto $P_2(-1,y_2)$ dobbiamo fare l'interpolazione fra il punto $(-1,0)$ e $(-1,1)$.

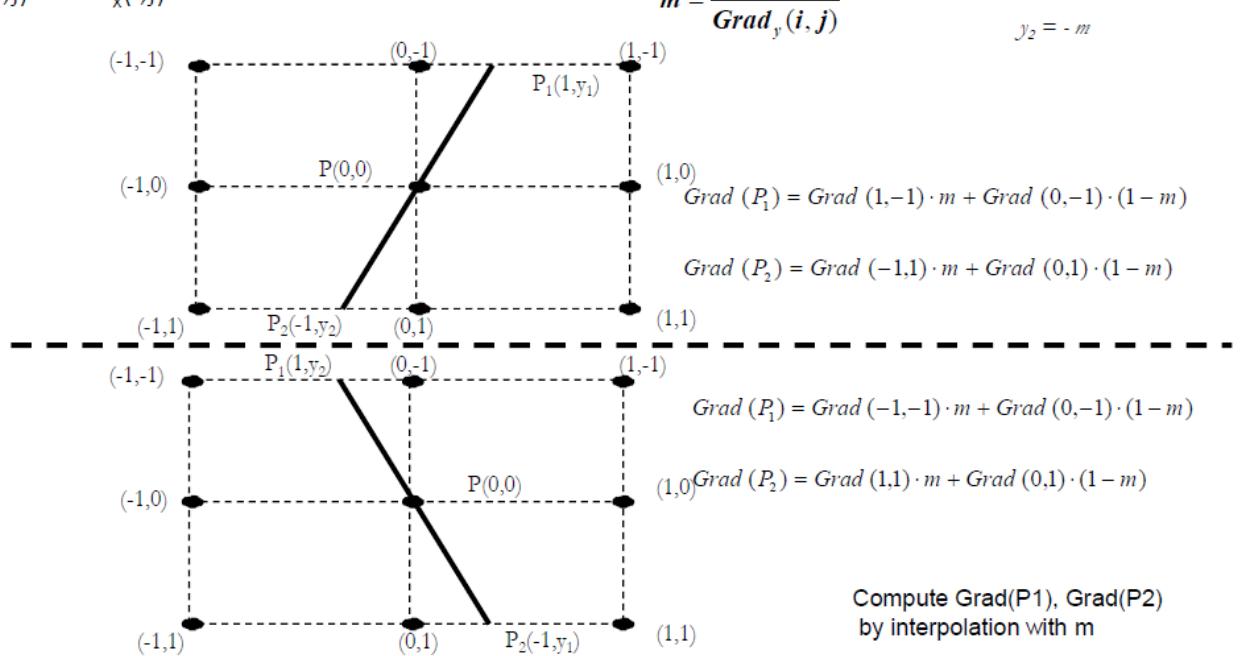
NON MAXIMA SUPPRESSION

if $\text{Grad}_x(i,j) > \text{Grad}_y(i,j)$

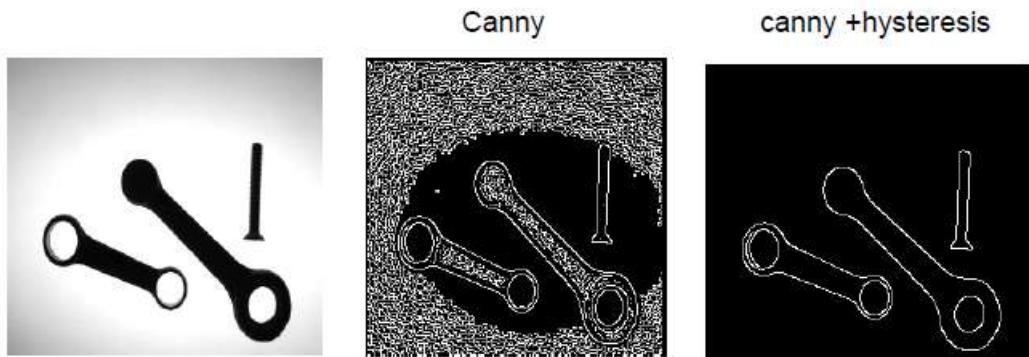


NON MAXIMA SUPPRESSION

if $\text{Grad}_y(i,j) > \text{Grad}_x(i,j)$

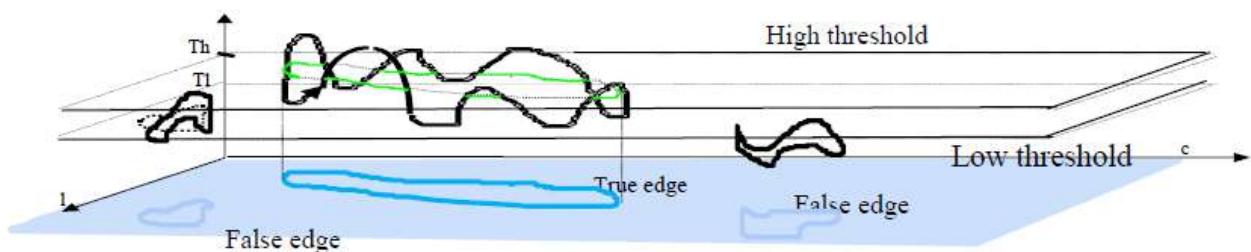


Dopo aver effettuato la soppressione dei non massimi gli edges che sono stati selezionati come validi hanno la proprietà di generare delle curve chiuse. Tuttavia fra questi dovrebbero essere selezionati solo i bordi forti. Canny ha proposto l'uso di una sogliatura basata sull'isteresi.



Andiamo a vedere come funziona la soglia con isteresi.

Consideriamo due soglie, una soglia alta per selezionare i bordi forti ed una soglia bassa per selezionare i bordi deboli ma solo quelli collegati con i bordi forti.



Normalmente:

$$T_H \cong 2 \div 3 T_L$$

Viene usata la soglia alta per cominciare la curva e la soglia bassa per continuarla. In particolare consideriamo solo i punti del gradiente che sono maggiori della soglia bassa e che hanno nell'intorno almeno un punto che ha gradiente con valore maggiore della soglia alta.

$$E_{H,L}(i,j) = \begin{cases} 1 & \text{if } G(i,j) > T_L \wedge \exists (k,l) \in N(i,j) \mid G(k,l) > T_H \\ \text{otherwise} & \end{cases}$$

Vediamo ora un esempio di applicazione di parte dell'algoritmo di Canny applicato ad un'immagine.

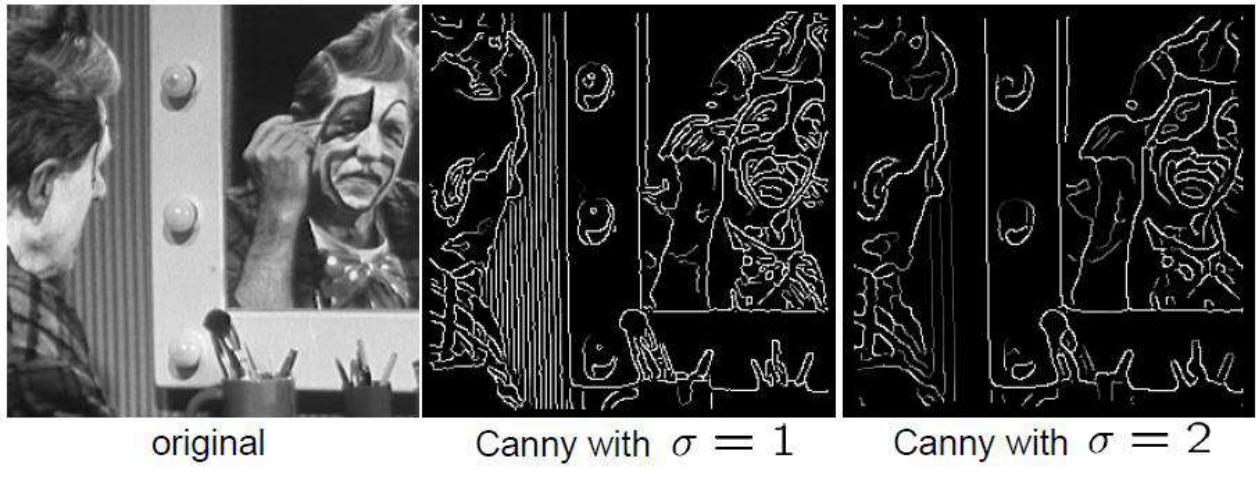


magnitude of the gradient



After non-maximum suppression

La scelta di sigma dipende dal comportamento che si desidera: sigma alto rileva bordi su larga scala, sigma basso rileva bordi più dettagliati e fini.



Andiamo ora a vedere un algoritmo di edge detection del secondo tipo alternativo. Tale **algoritmo è chiamato Laplacian of the Gaussian (LOG)**. Come già detto in precedenza, se la derivata prima ha un massimo la derivata seconda passa per lo zero in corrispondenza di quel punto. Quindi la derivata seconda si annulla in corrispondenza degli edges.

Come si calcola la derivata prima in modo robusto rispetto al rumore? Con uno smoothing senza delocalizzazione dei bordi e con una discretizzazione 2D della derivata seconda.

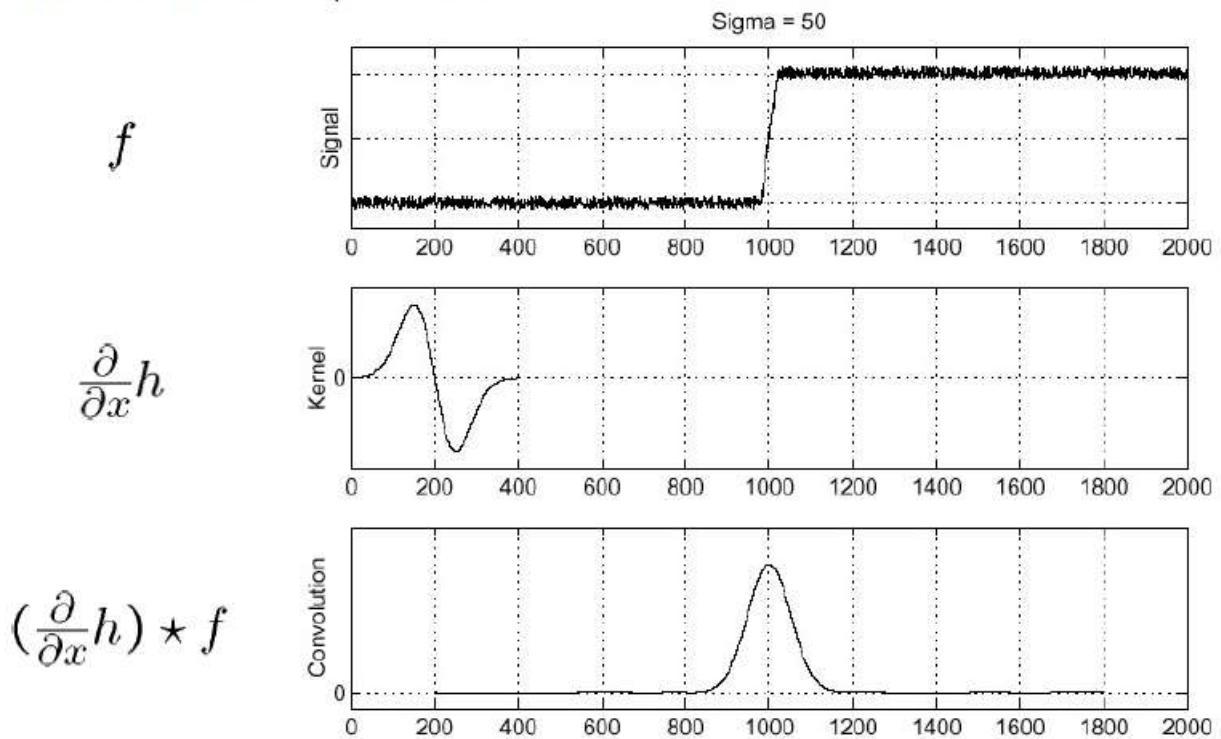
Calcolando la derivata seconda dell'immagine in due dimensioni (x e y) il gradiente diventa l'operatore laplaciano.

Sostanzialmente si calcola la derivata seconda (laplacian operator) del filtro gaussiano poi si fa la convoluzione con l'immagine. Con questo algoritmo è sufficiente la convoluzione, non è necessario imparare e usare delle soglie.

Come visto in precedenza la derivata della convoluzione fra immagine e filtro è uguale alla convoluzione fra immagine e derivata del filtro.

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

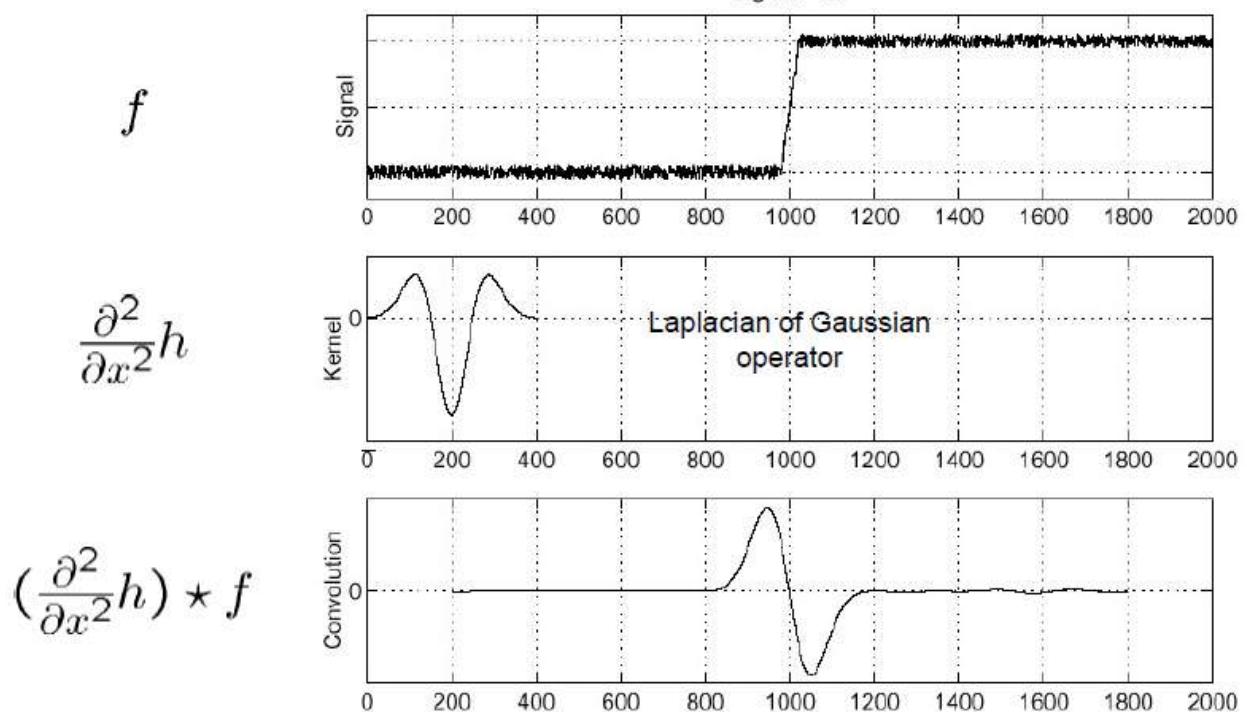
This saves us one operation:



Consider

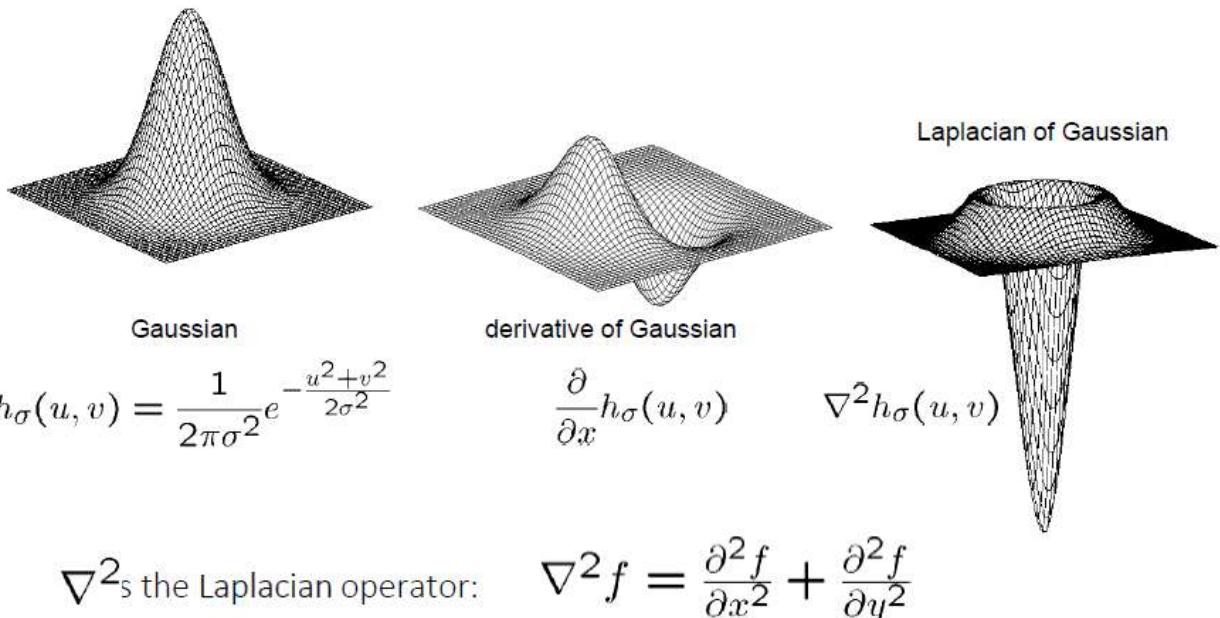
$$\frac{\partial^2}{\partial x^2}(h \star f)$$

Sigma = 50



Where is the edge?

Zero-crossings of bottom graph



$$h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

$$\frac{\partial}{\partial x} h_\sigma(u, v)$$

$$\nabla^2 h_\sigma(u, v)$$

∇^2 is the Laplacian operator: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Con il Laplacian of the Gaussian siamo in grado di capire se la derivata seconda dell'immagine passa per lo zero.

Il filtro che approssima l'operatore laplaciano è il seguente:

(LOG, Marr & Hildreth, 1980): Laplacian:

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

$$\begin{matrix} 1 & 1 & 1 \\ 1/3 & 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

- In general:

$$\begin{matrix} a & b & a \\ b & e & b \\ a & b & a \end{matrix}$$

$$\text{With } e = -(4a + 4b) \quad e - 2a + b = 1$$

Log can combine in a single operator both Gaussian+Laplacian

$$\begin{matrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{matrix}$$

$$\nabla^2(G * I)$$

Quindi per calcolare la derivata seconda dell'immagine in x e in y è sufficiente fare la convoluzione con il filtro che approssima l'operatore laplaciano. LOG applica un filtro che è sia filtro Gaussiano che Laplaciano. (fa entrambe le cose). Dopo aver applicato tale filtro si individuano i pixel che attraversano lo zero. Un pixel attraversa lo zero se il valore di tale pixel è minore di $-t$ e se uno dei pixel dell'intorno ha valore maggiore di t , dove t è un valore molto limitato. (ad esempio $t=1$ se $\sigma=1/2$).

Search for zero crossing

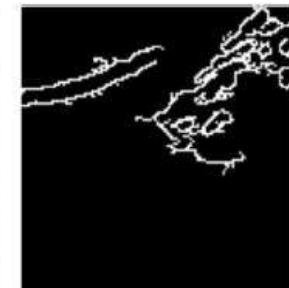
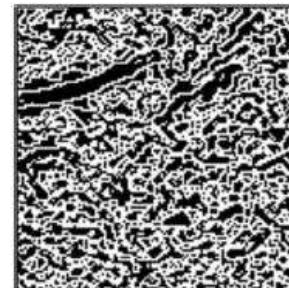
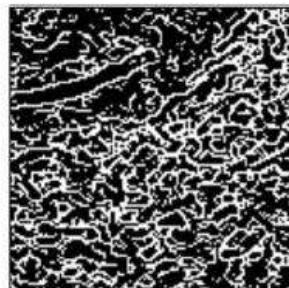
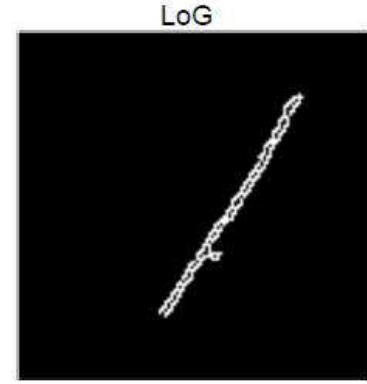
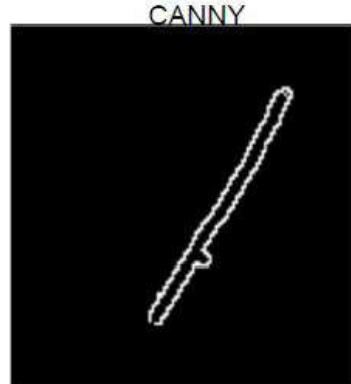
A pixel is a zero crossing if

$(I(x) < -t) \& (\text{one of neighborhood } N(I(x)) > t)$,
or viceversa;

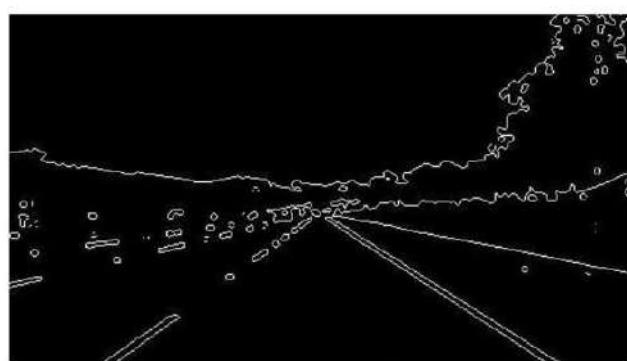
(t is very limited eg $t=1$ with Gaussians with $\sigma=1/2$)

Questo algoritmo ha il vantaggio di non essere sensibile rispetto ad una soglia tuttavia è più debole rispetto agli altri infatti presenta due problemi: Il primo è che tiene conto anche delle variazioni molto piccole dell'intensità dell'immagine, quindi è molto sensibile ai piccoli bordi chiusi e il secondo è che non lavora molto bene sulle discontinuità dell'immagine (ad esempio angoli).

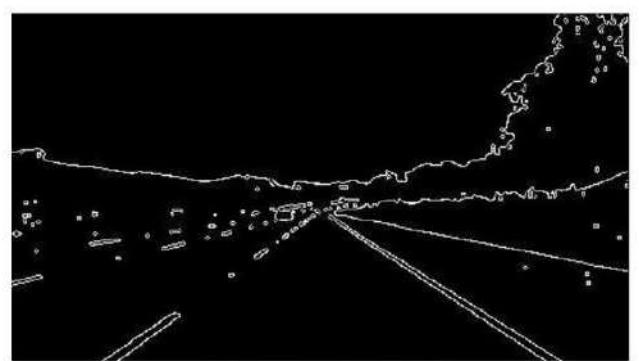
Vediamo alcuni esempi degli algoritmi appena visti:



hysteresis th

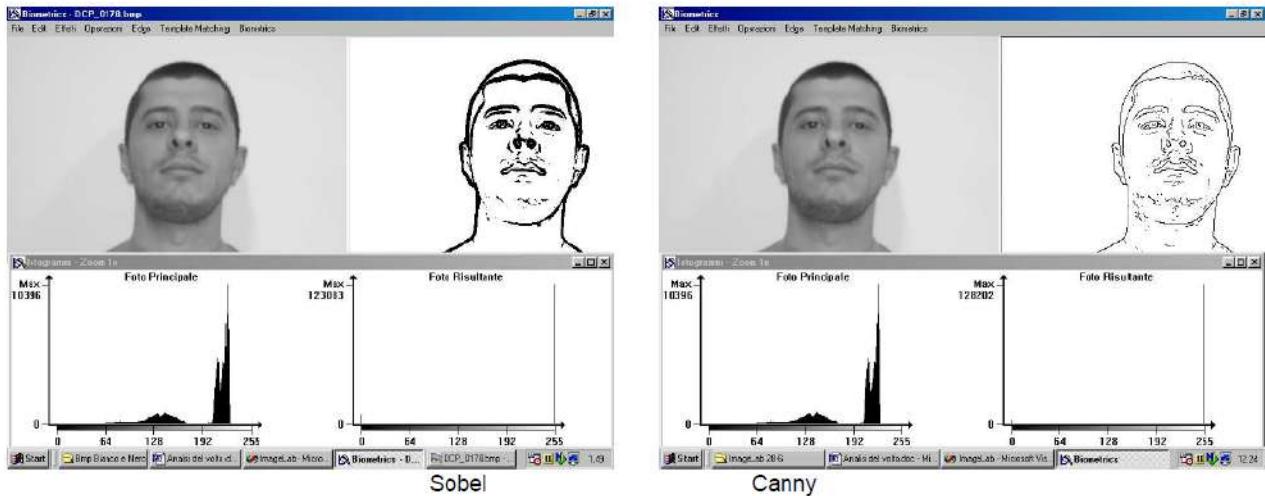


Edge Detection by LoG



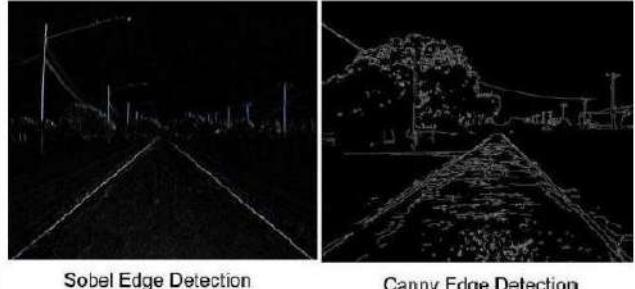
Edge Detection by Canny

Sobel often has more answers to single edges



Example of applications

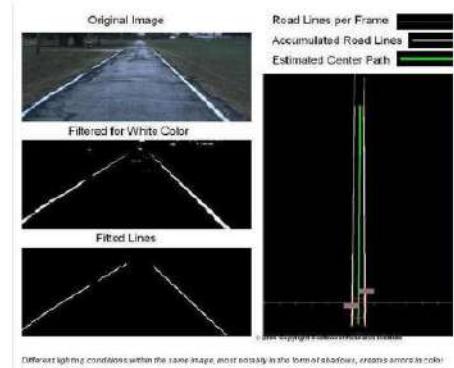
Edges



Edge detection algorithms pull out edges between objects of contrasting colors in an image.

And color segmentation

<http://www.swri.org/4org/d14/aerospace/path/vision.htm>



In molti sistemi di computer vision sono applicati questi algoritmi di edge detection. Tuttavia in molti sistemi è utilizzato il **deep learning** per fare edge detection. Vediamo alcune reti neurali proposte proprio per la edge detection.

http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Shen_DeepContour_A_Deep_2015_CVPR_paper.pdf

DeepContour: A Deep Convolutional Feature Learned by Positive-sharing Loss for Contour Detection



Figure 6. Illustration of our contour detection results on the NYUD dataset [43] for five selected example images (Depth images are not used). In each example, we see the original image and the ground truth and our results, respectively. Although our deep contour features are learned from the BSDS dataset, they can represent the object contours in NYUD dataset well.

In questo caso si insegna alla rete a rilevare i bordi allenando la rete con immagini di bordi create appositamente dall'uomo. Si riproduce un sistema in grado di rilevare i bordi come l'uomo.

Un'altra rete molto importante è CaseNet.

if you have supervised data, you can LEARN by a network their border..

Many tentatives in the last years.

CASENet CVPR2017

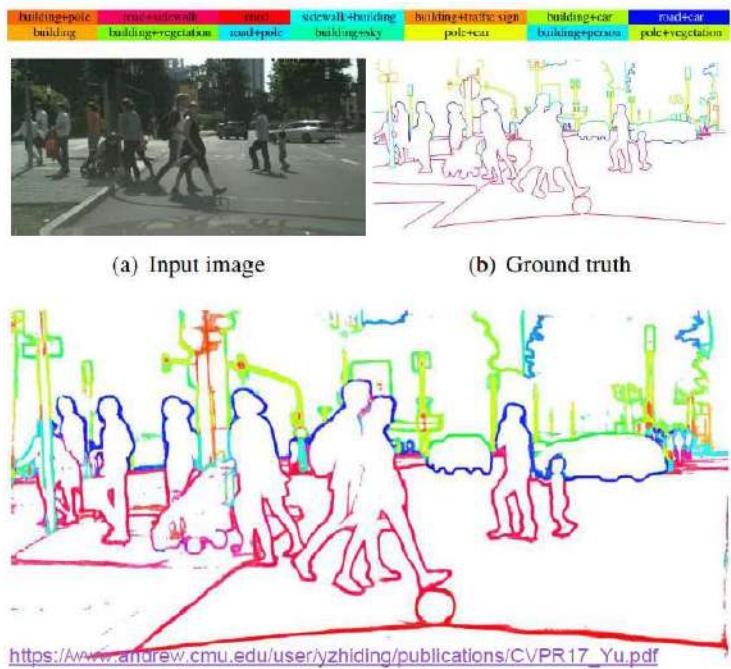
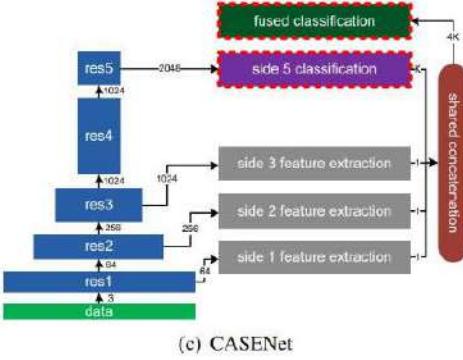
CASENet: Deep Category-Aware Semantic Edge Detection

Zhidong Yu*
Carnegie Mellon University
yuhidong@andrew.cmu.edu

Chen Feng* Ming-Yu Liu† Srikumar Ramalingam†
Mitsubishi Electric Research Laboratories (MERL)
cfeng@merl.com, mingyul@nvidia.com, srikumar@cs.utah.edu

Questa rete non è solo in grado di rilevare i contorni degli oggetti ma anche di dare loro un significato semantico.

An end-to-end network to at the same time detect edges
classify them according with some classes



In CASENet vengono usati più layers convolutivi che utilizzano le features estratte dal primo (o dai primi layers). Si ottengono i contorni e la classificazione di essi.

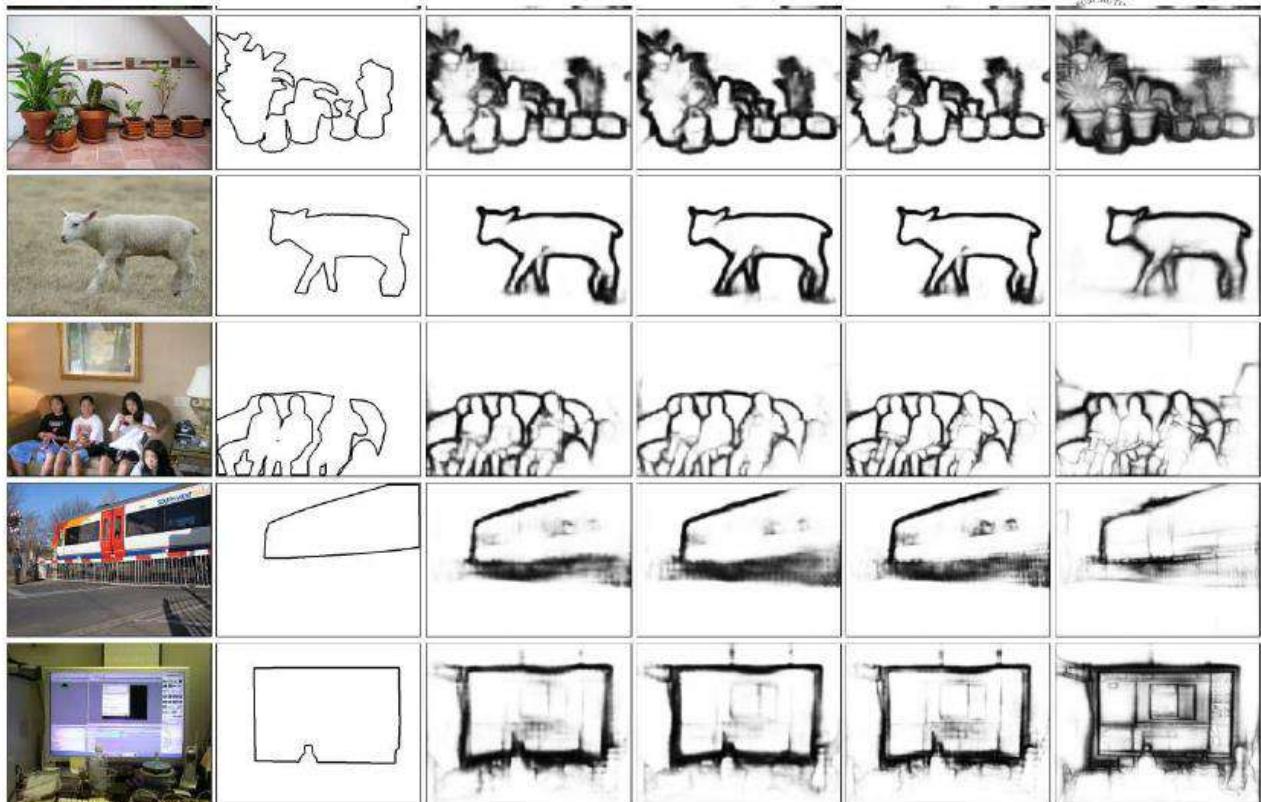


Figure 3. Class-wise prediction results of comparing methods on the SBD Dataset. Rows correspond to the predicted edges of “dining table”, “dog”, “horse”, “motorbike”, “person”, “potted plant”, “sheep”, “sofa”, “train” and “tv monitor”. Columns correspond to original image, ground truth, and results of Basic, DSN, CASENet and CASENet-VGG.

Ora andiamo ad analizzare un algoritmo della terza tipologia.

Se l'immagine è stata segmentata e binarizzata, dobbiamo solamente codificare i bordi. Andiamo a vedere un algoritmo che lavora su immagini già segmentate e binarizzate.

Chain code è una delle possibili rappresentazioni dei contorni dell'immagine e **L'algoritmo di Freeman** è un algoritmo usato anche per l'identificazione dei contorni che si basa su chain code. Partendo da un punto iniziale possiamo seguirne il contorno in senso orario o antiorario fino a che non torniamo al punto di partenza. Il chain code è un elenco che tiene conto dei cambi di direzione da un pixel a quello successivo. I possibili chain codes sono le 8 possibili direzioni dato un punto p iniziale.

0	7	6
1	p	5
2	3	4

I codici {0,2,4,6} sono le direzioni che indicano i movimenti in diagonale mentre i codici {1,3,5,7} indicano movimenti orizzontali/verticali. (il punto p è chiamato seed point).

Il contorno viene identificato dalla posizione iniziale del punto p e dal chain code.

Algoritmo di Freeman

Consideriamo un'immagine che ha background=0 (sfondo nero) e un seed point p_i . Se il contorno è composto da almeno tre punti ogni punto facente parte del contorno p_i dovrebbe avere un p_{i-1} precedente e un p_{i+1} successivo collegati in una regione 3x3 anch'essi facenti parte del contorno. Indichiamo la posizione di tali punti con il relativo chain code con valori fra 0 e 7. Tali punti corrispondono alla direzione precedente (pre) e successiva (post).

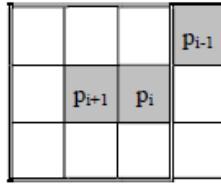
– Given $p_i \rightarrow \text{pre}=0 \text{ post}=5$.

p_{i-1}		
	p_i	p_{i+1}

- 0) Dato p_i e il suo precedente, per esempio $p_{i-1}=6$ (all'inizio possiamo fissare arbitrariamente il precedente)

2°	1°	p_{i-1}
3°	p_i	7°
4°	5°	6°

- 1) Per trovare il successivo (post) dobbiamo esplorare la regione circostante a p_i in senso antiorario partendo dal punto adiacente al precedente. Il primo punto da analizzare è infatti: $(\text{pre}(p_i)+1) \bmod 8$ poi proseguo in senso antiorario fino ad analizzare tutto l'intorno di p_i
- 2) Arrivati a questo punto abbiamo due condizioni possibili:
 - 2.1) è presente almeno un punto del contorno nell'intorno. La direzione del punto trovato (post) è data dalla chain code. Si ripete l'algoritmo fino a che non si arriva al 2.2



In the example p_{i+1} has a value equal to 1 thus $\text{pre}(p_i) = 6$ and $\text{post}(p_i) = 1$. We move to the next point p_{i+1} and we can compute its pre which is 5.

In general

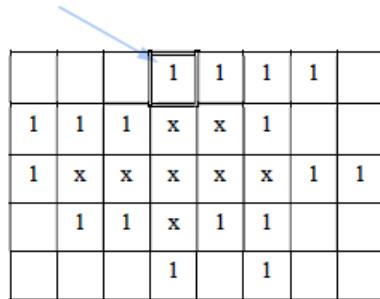
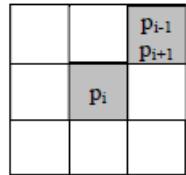
$\text{post}(p_i) = \text{freeman code } (p_i)$

$$\text{pre}(p_{i+1}) = (\text{post}(p_i) + 4) \bmod 8$$

$$p_i \leftarrow p_{i+1}$$

Continue until we find the initial point or termination

2.2) Nessun punto dell'intorno fa parte del contorno. In questo caso l'unica opzione possibile è quella di tornare indietro su p_{i-1} e ricominciare l'analisi dell'intorno considerando che p_{i-1} è uguale a p_{i+1} . $\rightarrow \text{pre}(p_{i+1}) = (\text{pre}(p_i) + 4) \bmod 8$.



Dopo aver eseguito questo algoritmo avremo un vettore di numeri che rappresentano la direzione in cui ci si deve spostare dal pixel corrente per seguire il contorno (chain code). Così vengono codificati i contorni con vettori di numeri.

(DA QUI IN POI NON SI CAPISCHE NIENTE, NON SO SE LE CHIEDE QUESTE COSE)

Un altro tipo di rappresentazione del contorno degli oggetti è tramite **spline**.

Trovare la migliore curva che passa per una serie di punti interpolandoli.

Spline è una formula matematica usata per rappresentare il punto tangente ad una curva. Una curva (come ad esempio quella di un contorno) può essere rappresentata da una funzione matematica.

Vengono usati due tipi di splines:

- **B-Spline** definisce i punti di controllo per rappresentare una curva con una funzione polinomiale di terzo grado o di grado superiore al terzo. Vedi <https://en.wikipedia.org/wiki/B-spline>
 - **Catmull Rom Spline** è la curva che meglio fa l'interpolazione tra i punti della curva e una funzione polinomiale di ordine 3. Vedi https://en.wikipedia.org/wiki/Centripetal_Catmull%E2%80%93Rom_spline

Gli oggetti possono essere segmentati trovando le curve che corrispondono ai contorni chiusi. Approcci chiamati **Active Contours** (usati nei video) consentono la rilevazione dei contorni muovendosi iterativamente verso la loro soluzione finale sulla base dell'immagine e di una opzionale forzatura da parte dell'uomo. Abbiamo due tecniche: (tecniche che si adattano ad un particolare bordo in maniera iterativa).

- **Snake:** è una curva spline bidimensionale che minimizza l'energia e si evolve verso le caratteristiche dell'immagine, come i bordi forti. Per applicare entrambe queste tecniche devo aver già applicato qualcosa prima e avere già dei punti definiti.
- **Level set:** sono tecniche che evolvono a curva come zero di una funzione caratteristica grazie la quale può essere facilmente modificata la topologia e d è possibile incorporare statistiche basate sulla regione considerata

Andiamo a vedere più nel dettaglio la tecnica di snake:

https://en.wikipedia.org/wiki/Active_contour_model

La curva è rappresentata come una funzione parametrica la cui energia deve essere minimizzata.

Parametric representation of curve

$$\mathbf{v}(s) = (x(s), y(s))$$

Energy functional consists of three terms

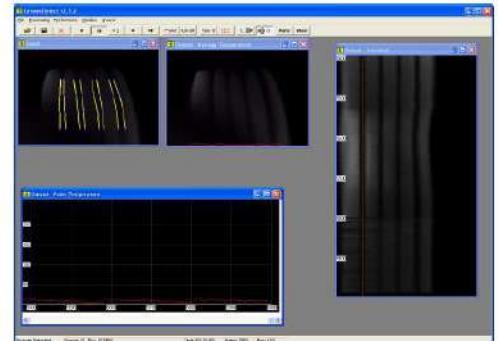
$$\mathcal{E} = \int [\mathcal{E}_{\text{int}}(\mathbf{v}(s)) + \mathcal{E}_{\text{img}}(\mathbf{v}(s)) + \mathcal{E}_{\text{con}}(\mathbf{v}(s))] ds$$

L'energia interna rappresenta la regolarità/smoothness lungo la curva e ha due componenti (resistenza all'allungamento e flessione), l'energia dell'immagine guida gli active contour verso le proprietà dell'immagine desiderate e l'energia esterna può essere usata per tenere conto di vincoli definiti dall'utente o conoscenze preliminari sulla struttura da individuare. Minimizzare tale funzione corrisponde a trovare un equilibrio tra questi tre termini. L'obiettivo è trovare il set di punti (curva) che minimizzano questa funzione dell'energia e loro corrisponderanno ai bordi.

Una volta trovati i bordi lo snake si lega a quei punti e riuscirà a seguirli nel tempo quando si spostano nell'intorno (hanno minime variazioni, tipo mi attacco al contorno di una bocca che parla e sono in grado di seguire questo contorno anche mentre la bocca parla).

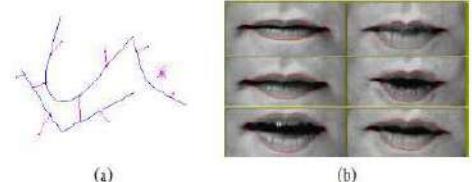
Used in many applications in image frames for tracking some boundaries in video.

An example for following weels boundaries in real time



Or for lip reading

For details see Szelinsky chapt. 5.1



Andiamo ora a vedere la tecnica Level set. Vedi qui https://en.wikipedia.org/wiki/Level-set_method

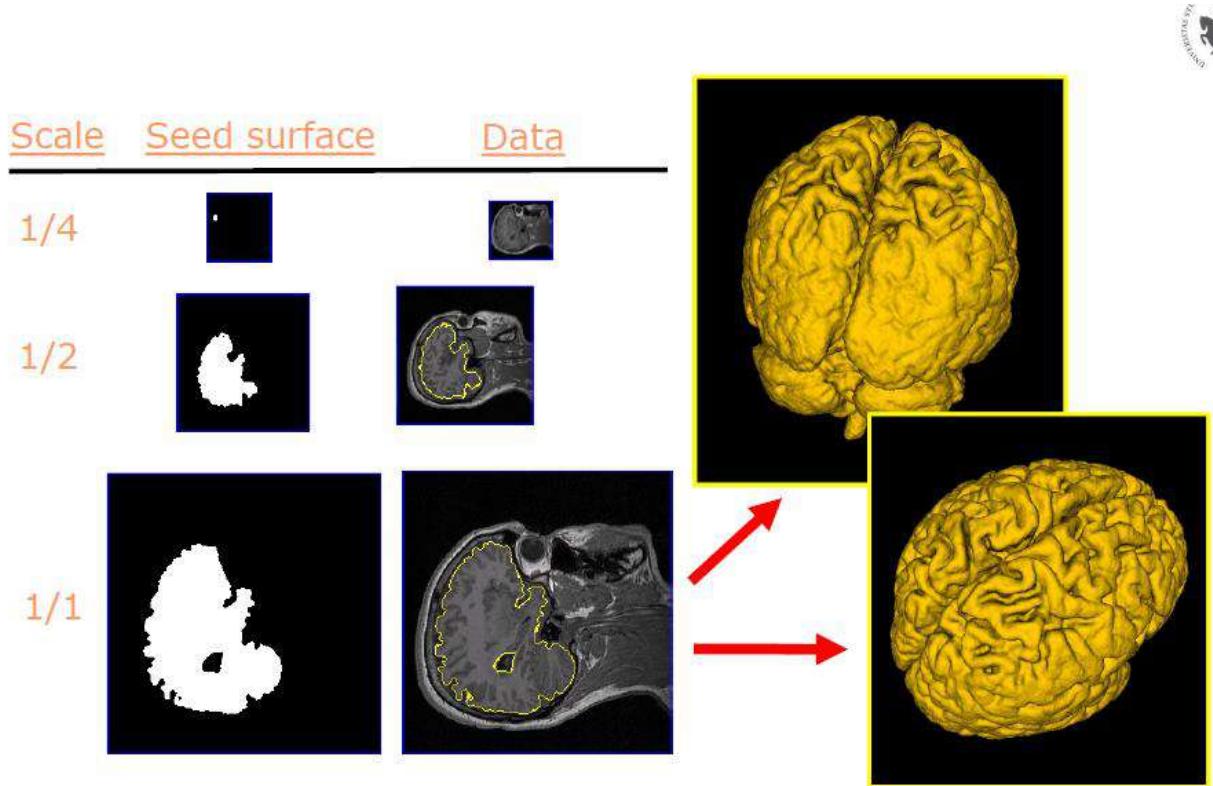
L'algoritmo Level set cerca la funzione che attraversa lo zero di una caratteristica per definire una curva.

$$\varphi(x, y, t) : \mathcal{R}^2 \times [0, T) \rightarrow \mathcal{R}$$

La curva è rappresentata in forma implicita:

$$C(p, 0) = \{(x, y) : \varphi(x, y, 0) = 0\}$$

E la curva è la funzione che attraversa lo zero:

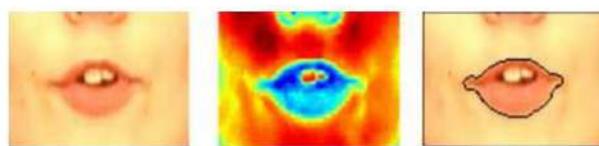


[A Level Set Method for Image Segmentation in the Presence of Intensity Inhomogeneities with Application to MRI](#), IEEE Trans. Image Processing, 2010

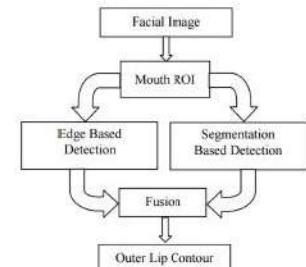
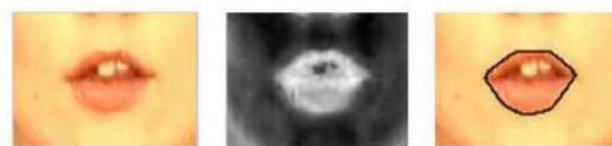
Experiments

A color transform for Lip (the blue component is not important)
And then active snakes

$$I = \frac{2G - R - 0.5B}{4}$$



And level set on gray level image



Snakes: active Contour models. International Journal of Computer Vision, vol. 1, pp. 259-288 (1987)

experiments

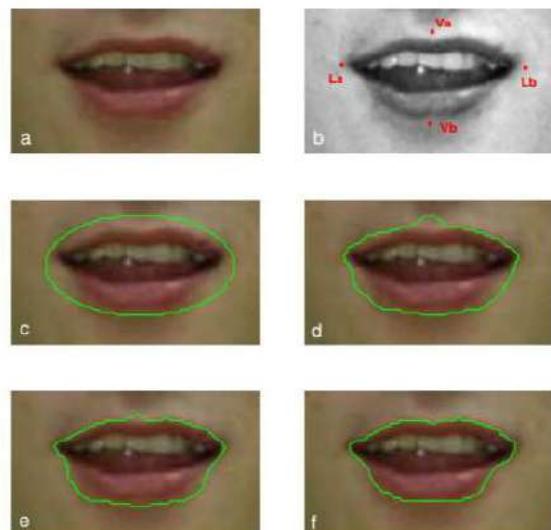


Problems for contour initialization in case of noisy images

2010 International Conference on Pattern Recognition

A Lip Contour Extraction Method Using Localized Active Contour Model with Automatic Parameter Selection

Xin Liu*, Yiu-ming Cheung*, Meng Li* and Hailin Lin†



http://www.comp.hkbu.edu.hk/~ymc/papers/conference/icpr10_publication_version.pdf

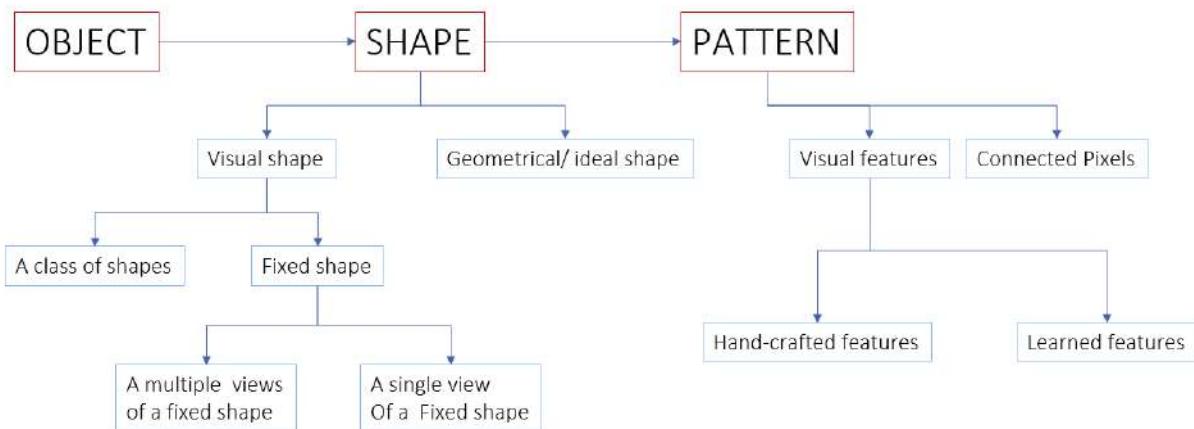
6. Shape

Prima di ricostruire l'oggetto 3D nel mondo reale è necessario individuare la sua shape in 2D e ancora prima di individuare la shape si costruisce un pattern. Il pattern è un particolare set di pixels facilmente riconoscibile (più semplice della shape), può essere ad esempio una linea o un cerchio.

(prima individuo i pattern, sulla base di essi costruisco la shape e poi sulla base della shape ricavo l'oggetto 3D).

3D World, Objects

2D view of the Object in the image



Queste sono le operazioni che possono essere eseguite sull'immagine senza nessuna conoscenza semantica.

(PARENTESI SULLA SEGMENTAZIONE CHE VEDREMO PIU' AVANTI)

In particolare l'operazione che divide le immagini in “regioni” è chiamata segmentazione. Tale suddivisione è eseguita sulla base di caratteristiche percettive, non è necessario avere conoscenze semantiche dell'immagine.

La semantic segmentation, invece, è più difficile della semplice segmentazione perché prevede la suddivisione in regioni sulla base della loro specifica semantica, quindi è necessario comprendere il significato di ciò che è rappresentato.

Segmentation



Segmentation



Semantic segmentation



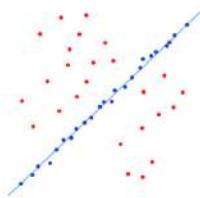
(FINE PARENTESI)

Andiamo ora a vedere una delle operazioni che non necessitano di alcuna conoscenza semantica elencata nello schema: il **riconoscimento/estrazione di pattern all'interno dell'immagine**.

Le possibili tecniche usate per la pattern recognition sono:

- **Fitting** (uso di un modello parametrico che rappresenta il pattern), trovare il miglior modello che si allinea ai punti della mia immagine.
- **Matching** (uso di un modello non parametrico che rappresenta il pattern), trovare se nell'immagine c'è qualcosa che corrisponde ad un'idea che io ho, ad un mio modello.
- **Misura delle similarità**, trovare la corretta funzione per definire su cosa (quali features) misurare la similarità e in che modo misurarla.
- utilizzo di una **rete neurale appositamente allenata**, devo avere sufficienti dati

FITTING



Fitting a parametric model

MATCHING



Matching a nonparametric model

MEASURING SIMILARITIES



Which features?
Which similarity?

Asking trained
neural
networks to do
it for us

Understanding how to define
Architecture, loss and data

Learning models, data, functions...

Andiamo ad analizzare le tecniche di pattern recognition una ad una.

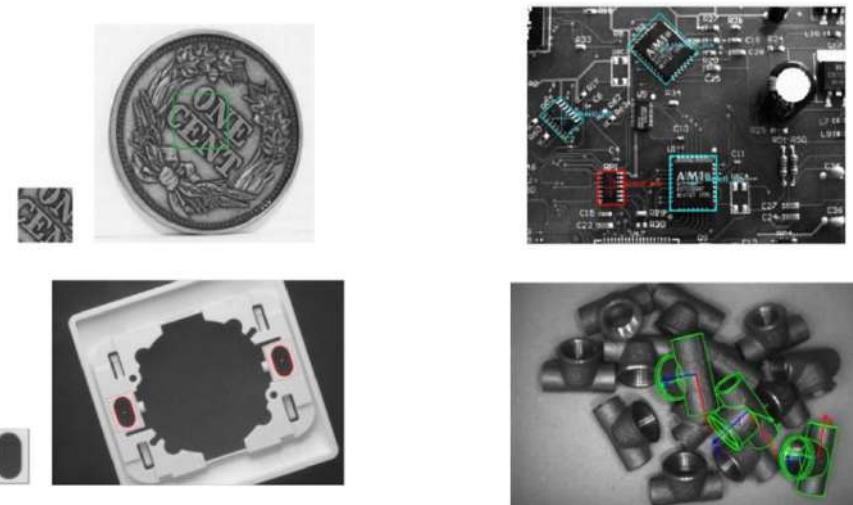
Fitting

Consiste nell'adattamento (fitting) di un modello parametrico o geometrico (il pattern) ad un insieme di punti (la porzione di immagine). Consiste nel trovare i valori dei parametri del modello capaci di generare il miglior allineamento possibile delle istanze del modello con l'insieme dei punti di riferimento (una porzione di immagine).

I principali algoritmi usati per fare fitting sono:

- Regressione lineare
- Hough transform
- RANSAC

Matching



Il matching fra un modello non parametrico (che in questo caso può essere il pattern, un template o una shape) e un insieme di punti solitamente si riferisce alla ricerca della migliore corrispondenza tra i punti del modello/template/shape e l'immagine di riferimento o l'insieme dei punti considerati. Questo implica spesso la ricerca della trasformazione (o deformazione) del modello che darà luogo alla migliore corrispondenza.

Come abbiamo già accennato in questo caso, a differenza del precedente, il modello può rappresentare sia un pattern, cioè qualcosa di molto semplice, sia un template, cioè qualcosa di più complesso, sia una shape, cioè qualcosa di ancora più complesso. Esistono i seguenti tipi di matching:

- Shape matching
- Template matching
- Iterative Closest Point (ICP)
- Global Matching

Se consideriamo lo **shape matching**, il modello non parametrico può essere rappresentato da:

- Un'immagine
- Un insieme di punti 3D
- Un vettore descrittore della shape globale
- Un vettore di punti rappresentanti features locali e relativi descrittori
- Un modello compresso e imparato

Andiamo ora a considerare il **template matching** (e wrapping).

Un template corrisponde ad una descrizione di una precisa shape fatta utilizzando i pixels. Consiste genericamente in una finestra di pixels la cui semantica ora non è rilevante e non è nota al template stesso. Il riconoscimento di templates nel concreto può risultare difficile perché si può incontrare nell'immagine lo stesso template ma con colore diverso, dimensione diversa, prospettiva diversa, rotazione diversa, ecc..



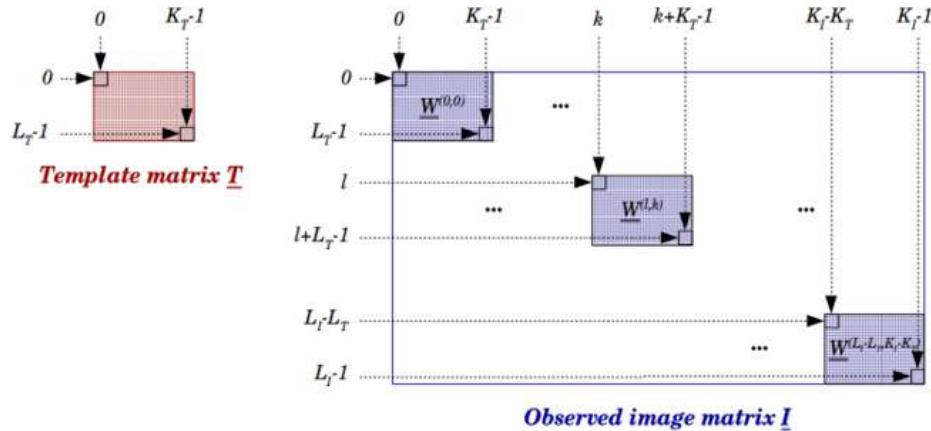
Il matching è una generica operazione che consiste nel confronto di similarità fra entità, regioni o vettori dello stesso tipo. Lo specifico template matching si ha quando la similarità viene calcolata sulla base di un determinato modello (immagine 2D) o prototipo che deve essere riconosciuto. Il riconoscimento viene fornito misurando la similarità fra il modello e l'immagine corrente.

Un buon template matching dovrebbe essere invariante (riconoscere ugualmente anche se la differenza fra immagine e template è una di queste operazioni) rispetto a:

- Scale
- Rotazione
- Traslazione
- Luminosità, ecc

Questa operazione è estremamente complessa. E' inoltre importante ricordare che template matching non è object detection. Detection significa trovare qualcosa simile al modello che abbiamo o che abbiamo imparato, matching significa trovare l'esatta (a meno delle trasformazioni sopra) corrispondenza tra modello e immagine. Ovviamente il template matching risulta più facile se vogliamo trovare l'esatto template ignorando il fatto che debba essere invariante alle trasformazioni.

Possiamo dire che il template matching, nel concreto, è la semplice ispezione dell'immagine per verificare la presenza di una data sottoimmagine che corrisponde al template. E' la ricerca in tutte le possibili posizioni dell'immagine (sliding window) e la valutazione della misura del grado di corrispondenza. Si usa l'idea di sliding window facendo corrispondere il template ad ogni porzione di immagine calcolando ogni volta la similarità. Posso utilizzare due misure di similarità diverse, o il **mean square error** (distanza euclidea fra immagine e template) oppure la **cross correlation** (faccio la convoluzione fra la porzione di immagine e il filtro che è il template, più il risultato è alto più i due sono simili).



Esistono diversi tipi di template:

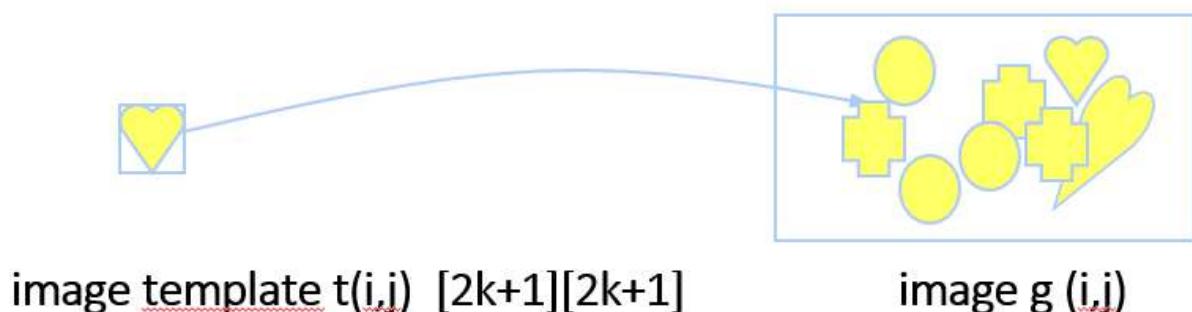
- Global template matching: si cerca la corrispondenza con l'intero oggetto rappresentato nel template
- Local template matching: si cerca una specifica visual feature
- Block matching: si cerca una finestra senza un particolare significato ma con una specifica dimensione
(spiega)

Fare il matching esatto di template in un contesto binario (solo bianco e nero) è il caso più semplice, mentre in immagini in scala di grigio o colorate è molto più difficile. Per questo motivo non si cerca il matching esatto ma si cerca un certo livello di similarità fra immagine e template. Infatti esistono due tipi di template matching, quello **pixel level**, che ricerca la corrispondenza esatta con la finestra di pixel che rappresenta il template, e quello **feature-based level** che ricerca la corrispondenza con la shape fornita dal template basandosi sulla similarità.

Esistono diversi metodi per fare il matching con diverse misure di similarità:

- Matching with cross correlation
- Matching with NCC (normalized cross correlation)
- Matching with edges (and distance transforms)
- Matching Fourier coef, or global features
- Matching keypoints (Sift)
- Matching by (NN or learned) descriptors

Andiamo a vedere il **template matching con cross-correlation** (dot product) come misura di similarità.



Per ogni punto (m,n) dell'immagine $g(i,j)$ la funzione di matching utilizzata è la cross-correlation che è definita nel seguente modo:

$$F : t(i,j) \times g(i,j) \rightarrow R$$

Function matching $F(x,y) = \sum \sum F(t(x,y)g(x,y))$

F(.) cross- correlation is the product

(let suppose the template squared, even if it is not necessary)

Cross-correlation of an image g with a t template of $(2k+1) \times (2k+1)$ pixels:

$$F(m,n) = \sum_{i=-k}^k \sum_{j=-k}^k g(m+i, n+j) * t(i,j)$$

Più F restituisce un valore elevato, più sarà elevata la similarità fra quella porzione dell'immagine e il template. Questo perché minimizzare il mean square error corrisponde a massimizzare la cross-correlation, vedremo dopo la dimostrazione.

Andiamo a vedere il **template matching con mean square error MSE** come misura di similarità.

Consideriamo un'immagine $g(m,n)$ e un template $t(i,j)$ una misura di similarità invariante solo alla traslazione è il mean square error che corrisponde alla distanza esatta fra il punto dell'immagine e il corrispondente punto del template. Per ogni punto dell'immagine (m,n) si calcola:

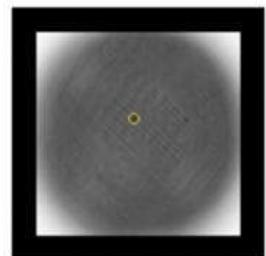
$$D_{i,j}(m,n) = \sqrt{\sum_i \sum_j (g(i+m, j+n) - t(i, j))^2}$$



Template **T**



Observed image **I**



Distance map **D**

Ovviamente più il mean square error è piccolo più alta sarà la similarità, il matching migliore è quello in cui il MSE è minimo.

Per ridurre il carico computazionale del mean square error spesso sono utilizzate metriche equivalenti ma più leggere dal punto di vista computazionale:

Mean Absolute Difference (MAD):

$$MAD_{i,j}(m,n) = \frac{1}{N} \sum_i \sum_j |g(i+m, j+n) - t(i, j)|$$

Sum of Absolute Difference (SAD):

$$SAD_{i,j}(m,n) = \sum_i \sum_j |g(i+m, j+n) - t(i, j)|$$

Dimostrazione del fatto che **massimizzare la cross correlation equivale a minimizzare il mean square error**: questo è il motivo per cui il matching migliore sarà quello che avrà la più alta cross correlation.

Considerando la formula del MSE

$$D_{i,j}(m,n) = \sqrt{\sum_i \sum_j (g(i+m, j+n) - t(i, j))^2}$$

andiamo a calcolarne una approssimazione senza la radice quadrata:

$$E2(m,n) = \sum_j \sum_i g^2(i+m, j+n) - 2g(i+m, j+n)t(i, j) + t^2(i, j)$$

L'ultimo termine è costante perché rappresenta il livello medio di grigio del template ed è possibile, supponendo che sull'intera immagine il livello medio di grigio è costante, supporlo costante. Essendo due termini costanti possiamo trascurarli. Minimizzare l'errore scritto con questa formula corrisponde a massimizzare il temine centrale poiché è preceduto dal segno meno.

Quindi il matching migliore sarà quello che massimizzerà il temine centrale, ovvero:

$$R(m,n) = \sum_i \sum_j g(i+m, j+n)t(i, j)$$

Che è proprio la formula della cross-correlation. Si cerca il punto (m,n) che massimizza $R(m,n)$.

Questo ragionamento è basato sul supporre costante il livello di grigio dell'immagine, tuttavia questa è un'approssimazione troppo irrealistica quindi spesso l'utilizzo della cross-correlation non è molto preciso.

Questo è il motivo per cui spesso la cross correlation viene normalizzata in $[0,1]$ dividendo per la media geometrica della luminosità (livello di grigio) dell'immagine e del template nel seguente modo:

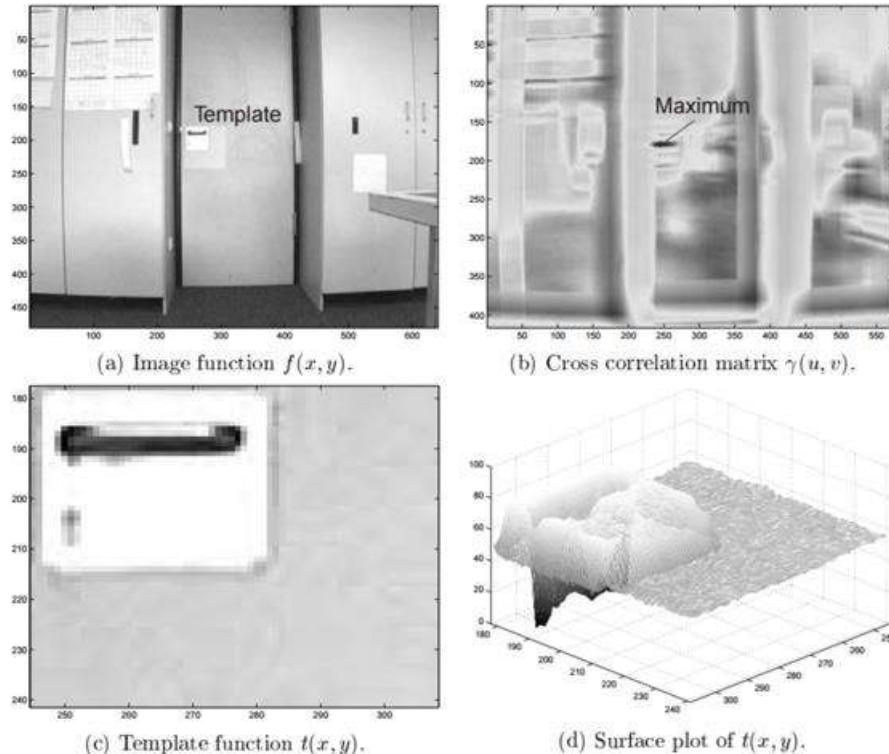
$$N(m, n) = \frac{\sum_i \sum_j g(i+m, j+n) t(i, j)}{\sqrt{\sum_i \sum_j g(i+m, j+n)^2} \sqrt{\sum_i \sum_j t(i, j)^2}}$$

Tuttavia nemmeno questo è sufficiente. Si usa al posto della cross-correlation la **NCC Normalized Cross Correlation**. In questo caso viene standardizzato il valore della cross-correlation in base al valore medio del livello di grigio nell'intorno considerato.

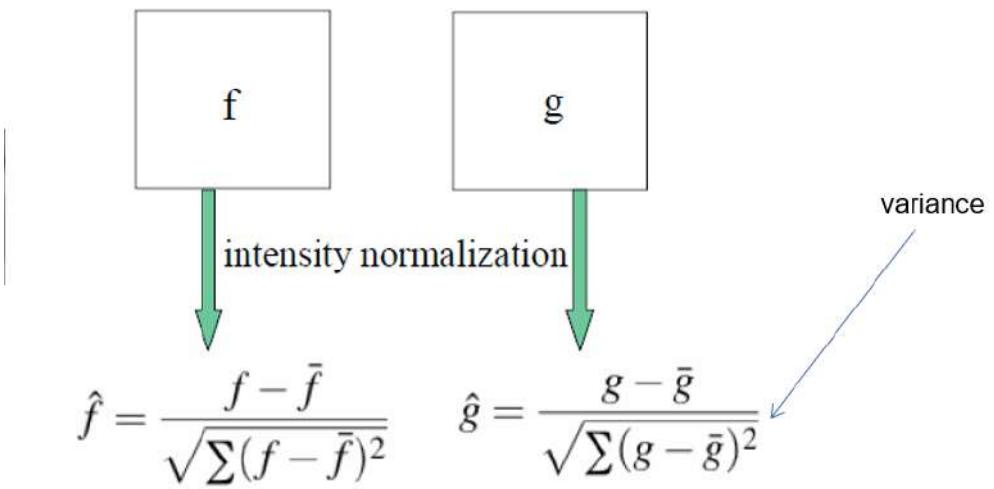
$$NCC(m, n) = \frac{\sum_i \sum_j (g(i+m, j+n) - \bar{g})(t(i, j) - \bar{t})}{\sqrt{\sum_i \sum_j ((g(i+m, j+n) - \bar{g})^2 \sum_i \sum_j (t(i, j) - \bar{t})^2)}$$

Dove g e t (con trattino sopra) sono i valori medi del livello di grigio nell'intorno.

E' utilizzata per cercare un template in un immagine o per cercare una data immagine in una sequenza di frames. E' l'esempio più semplice di tracking.

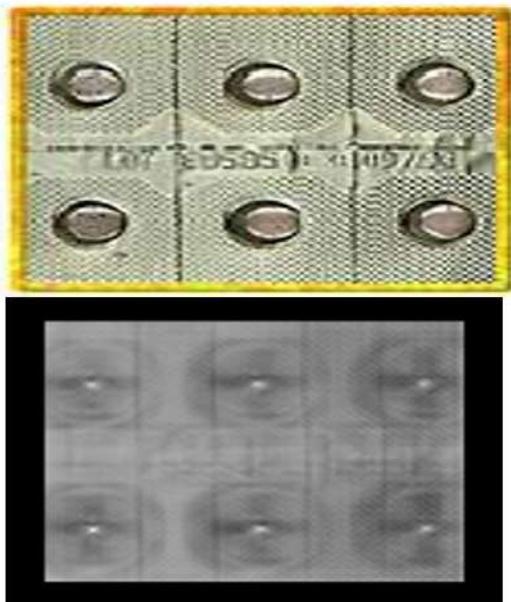


E' possibile anche calcolare NCC normalizzando prima l'intera immagine e l'intero template e poi calcolando la cross-correlation nel modo classico. In realtà questo è il metodo di template matching più utilizzato e spesso viene eseguito quando la luminosità dell'immagine e del template possono variare a causa delle condizioni di illuminazione e di esposizione:



Esempi di applicazione

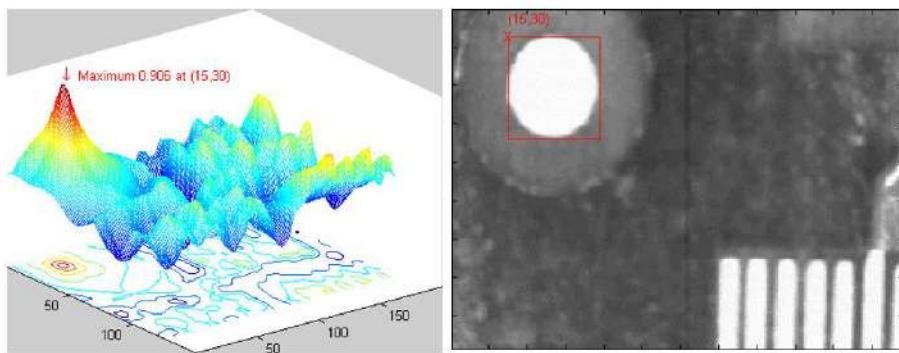
Typical in machine vision, for quality inspection

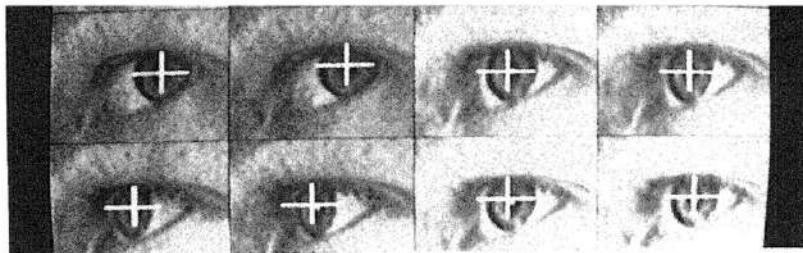


cross-correlation
thresholding



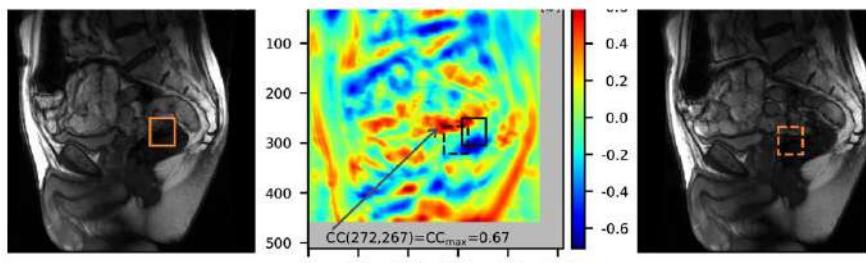
- Load printed circuit board into a machine
- Teach template image (select and store)
- Load printed circuit board
- Capture a source image and find template





Some templates are easy to be matched.....
Eye tracking,
medical image target tracking.

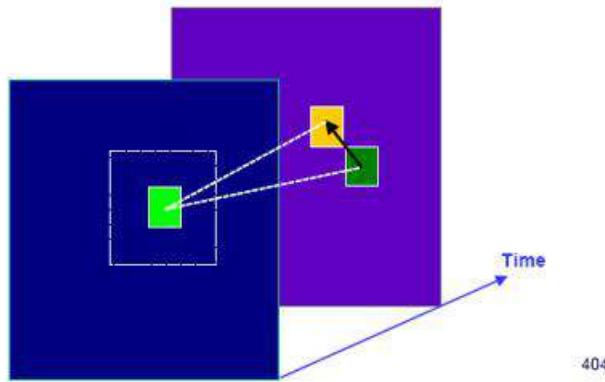
very simple when there is a single shape.. Very fast



Andiamo ora a vedere nello specifico il **block matching**.

Il block matching è utilizzato per trovare il vettore di movimento in frame consecutivi. Si cerca di capire come si muove in frame consecutivi un blocco. Data un'immagine al tempo t $I(t)$ e una al tempo $t+1$ $I(t+1)$ si cerca di capire come si è mosso un blocco $k \times k$ per calcolare il vettore movimento. In questo caso è sufficiente usare SAD per trovare la corrispondenza con il blocco solo però nel caso in cui il movimento non corrisponda ad una rotazione.

Assumes the 2D translational model, for each block, a single vector is found by minimizing the total pixel intensity difference



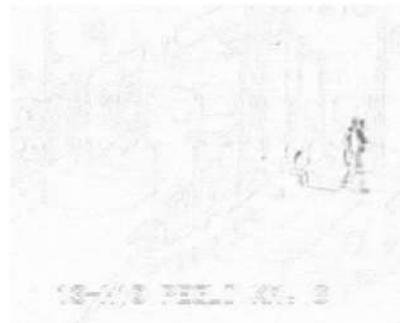
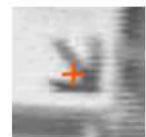
404

Vediamo alcune applicazioni di block matching:

Machine vision: defect detection, template matching

Multimedia: motion analysis compression

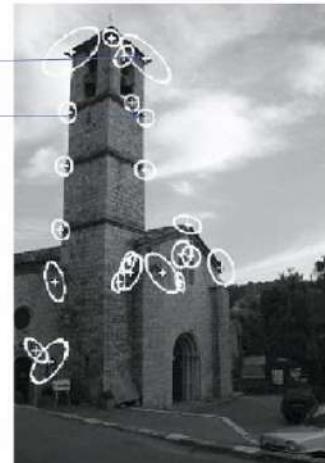
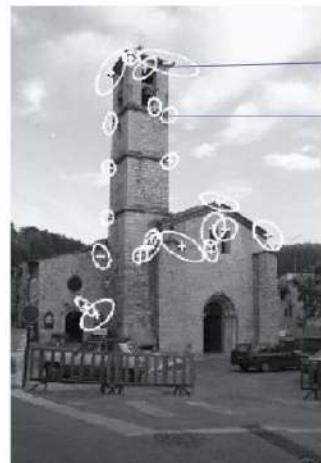
Surveillance: camera motion correction



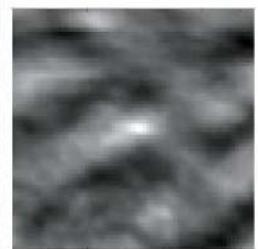
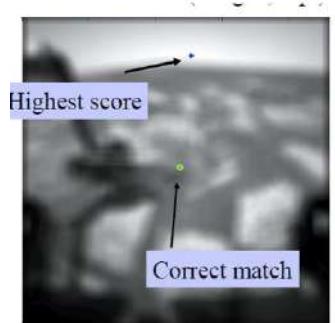
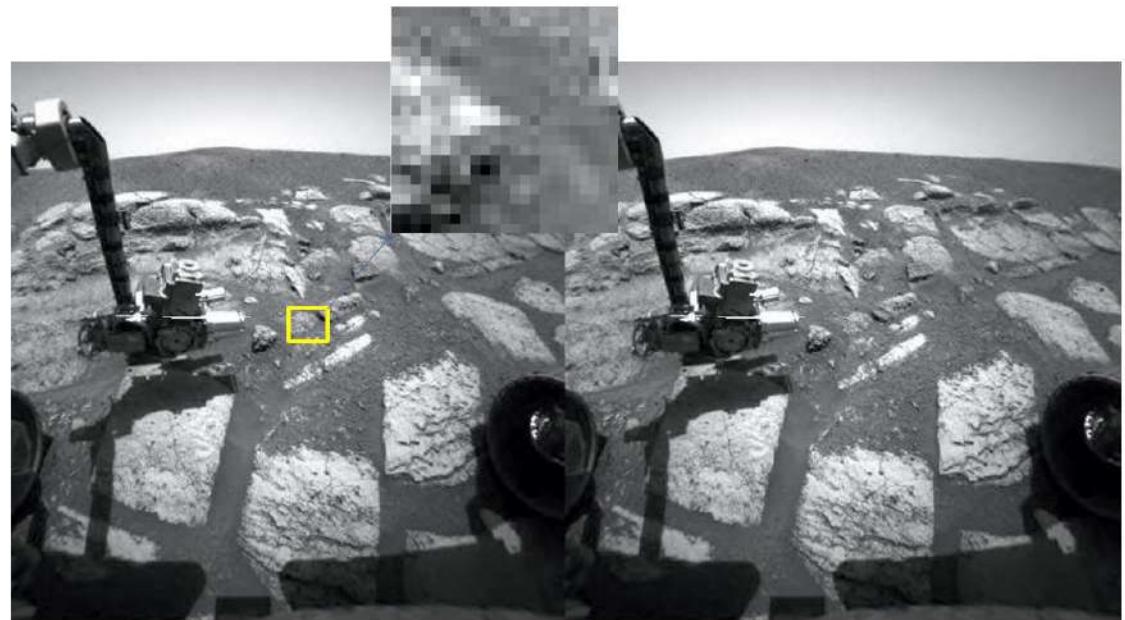
Background suppression for camera motion correction
1999

In stereo and motion correspondence

Each block can be compared in frame pairs
either we select some specific block to match (features..)



From Mars rover exploring “the captain” formation (thanks to Collins MIT)



Highest score also coincides with correct match.
Also, looks like less chances of getting a wrong match.

Logo detection in documents

Thanks to (A. Alaey)

In the context of logo recognition, scale and rotation invariance are not concerned as the logos appear at fix orientation and the resolution in document work-flow is also fixed.

Problems to be solved include the choice of a robust distance-based function, and to deal with the position invariance and the complexity issues related to the image and template sizes..
now it is not a problem..



Property	Number of document images	Number of logos	Number of logo classes
Tobacco800 [21]	1290	432	35
Subset 1	261	263	23
Subset 2	685	705	82

Dataset	Accuracy		Rec. Acc. %	Overall performance %
	Recall %	Precision %		
Tobacco800 [21]	99.31	20.58	97.90	97.22
Subset 1 of Itesoft dataset	97.21	23.85	98.82	96.06
Subset 2 of Itesoft dataset	96.31	23.19	94.10	90.63



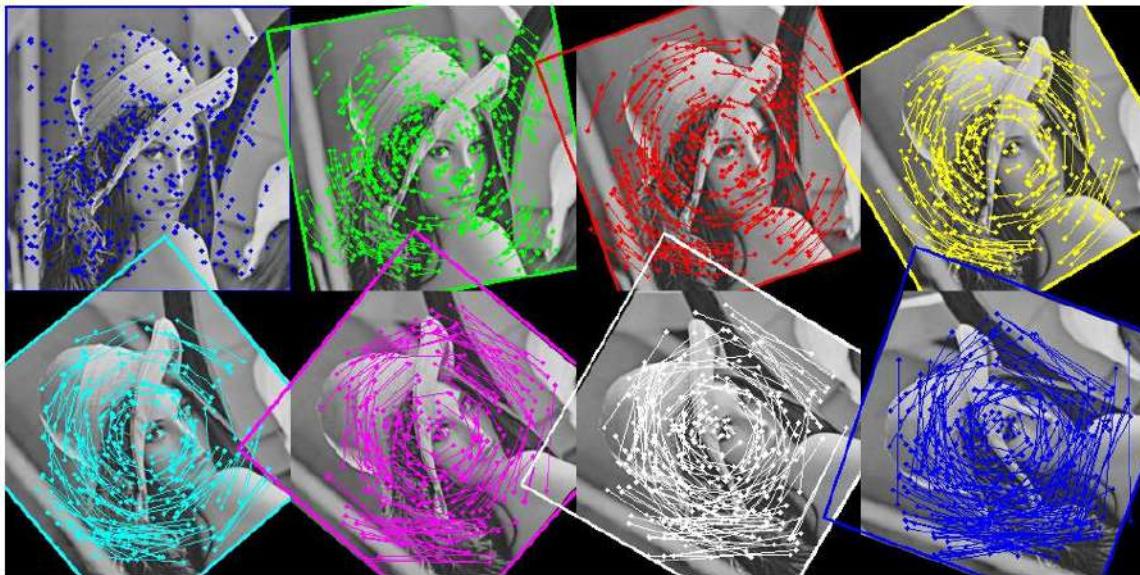
Riassumendo il template matching è usato in immagini a livelli di grigio o colorate quando so che devo trovare esattamente lo stesso template. Inoltre è usato in video per identificare una shape assumendo che essa non cambi nel tempo, in machine vision per trovare semplici template e in ocr per trovare il logo in documenti.

Tuttavia questo metodo, così definito, non funziona quando il template sull'immagine differisce dal template utilizzato nel colore, nella forma, nella dimensione, nella rotazione o in trasformazioni affini o prospettiche.

Quindi NCC non è la metrica corretta quando abbiamo differenze utilizzando nel colore, nella forma, nella dimensione, nella rotazione o in trasformazioni affini o prospettiche.

In questi casi dobbiamo:

- Trasformare il pattern: ripetere il template per ogni sua possibile rotazione o scaling, oppure fare lo scaling dell'immagine o applicare altre trasformazioni all'immagine.
- Usare descrittori compatti



Ora andiamo a studiare la geometria per capire come fare le varie trasformazioni.

Geometria

Le primitive geometriche 2D e 3D sono gli elementi costitutivi di base usati per descrivere shapes tridimensionali. Inoltre avremo a che fare con punti 2D e 3D, linee 2D, piani 3D e sistemi in coordinate omogenee.

Perché è importante riprendere concetti di geometria per studiare la computer vision?

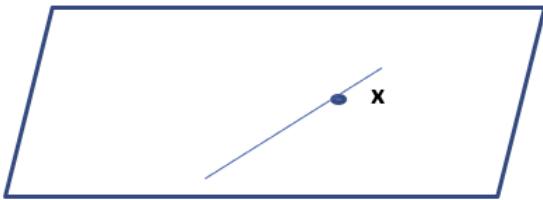
- Il modello di visione è basato su modelli fisici di lenti e su geometria ottica quindi il mondo può essere descritto da geometria 3D e dal corrispondente mapping sul piano 2D. (vedi camera models).
- Molti oggetti sono disegnati dall'uomo utilizzando tools CAD che utilizzano modelli geometrici quindi sono dominati da linee, curve, piani ecc.
- Il sistema di visione del cervello umano utilizza descrittori geometrici incorporati nel nostro sistema percettivo per riconoscere il mondo.
- La geometria è un buon framework deterministico che aggiunge noise e dati incerti descritti da modelli statistici. Inoltre la geometria è meno difficile di altri paradigmi.

Ad esempio, la trasformazione geometrica può essere utilizzata insieme al template matching per riconoscere gli oggetti da altri punti di vista (rispetto al template).

Andiamo a vedere come vengono rappresentati i punti 2D nel piano Cartesiano.

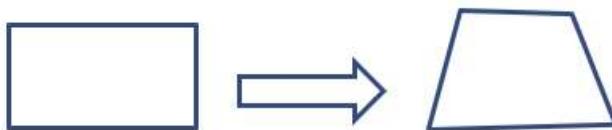
Un punto nel piano cartesiano è rappresentato nel seguente modo:

$$\mathbf{x} = (x, y) \in R^2 \quad x = \begin{bmatrix} x \\ y \end{bmatrix}$$



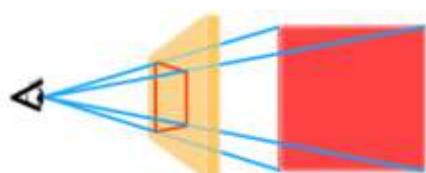
Questo tipo di coordinate vengono usate per rappresentare il piano dell'immagine e x rappresenta le coordinate dei pixels nell'immagine. Le immagini corrispondono alla trasformazione del mondo 3D nel piano dell'immagine del camera model. E un processo simile viene eseguito anche nel modello mentale degli umani.

Tuttavia l'immagine rappresentata in coordinate cartesiane 2D perde informazioni relative al corrispondente oggetto 3D che l'immagine dovrebbe rappresentare. In particolare si perde la rappresentazione di alcune primitive 3D, come la definizione di quali rette parallele si incontrano in un punto all'infinito, come informazioni sulla prospettiva oppure le informazioni sul punto di vista ed un eventuale cambio di punto di vista.



Per questo viene utilizzata la geometria proiettiva.

La geometria proiettiva è la parte della geometria che modellizza i concetti intuitivi di *prospettiva* e *orizzonte*. Definisce e studia gli enti geometrici usuali (punti, rette, ...) senza utilizzare misure o confronto di lunghezze. Può essere pensata informalmente come la geometria che nasce dal collocare il proprio occhio in un punto dello spazio, così che ogni linea che intersechi l'"occhio" appaia solo come un punto. Le grandezze degli oggetti non sono direttamente quantificabili (perché guardando il mondo con un occhio soltanto non abbiamo informazioni sulla profondità) e l'orizzonte è considerato parte integrante dello spazio. Come conseguenza, nella geometria piana proiettiva due rette si intersecano sempre, non esistono quindi due rette parallele e distinte che non hanno punti di intersezione.



La geometria proiettiva è la geometria "vista da un occhio".

In geometria, lo spazio proiettivo è lo spazio ottenuto da uno spazio euclideo (ad esempio, la retta o il piano) aggiungendo i "punti all'infinito". A seconda della dimensione, si parla quindi di retta proiettiva, piano proiettivo, ecc. vedi per dettagli https://it.wikipedia.org/wiki/Spazio_proiettivo

I punti nella geometria proiettiva sono descritti dalle **coordinate omogenee** o coordinate proiettive.

Le coordinate omogenee hanno diversi campi di applicazione tra cui la visione artificiale 3D e la grafica poiché consentono trasformazioni affini e in generale trasformazioni proiettive.

I punti sono proiettati da uno spazio di dimensione n ad uno spazio di dimensione n+1 che non possiede il centro (e possiede anche una relazione di equivalenza che nella formula non è indicata).

$$P^n = \mathbb{R}^{n+1} - \vec{0}$$

In the 2D case $x=(x,y)$ $\mathbf{x} = (x, y) \in \mathbb{R}^2 \rightarrow \tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) \in P^2$

In the 3D case $x=(x,y,z)$ $\mathbf{x} = (x, y, z) \in \mathbb{R}^3 \rightarrow \tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{z}, \tilde{w}) \in P^3$

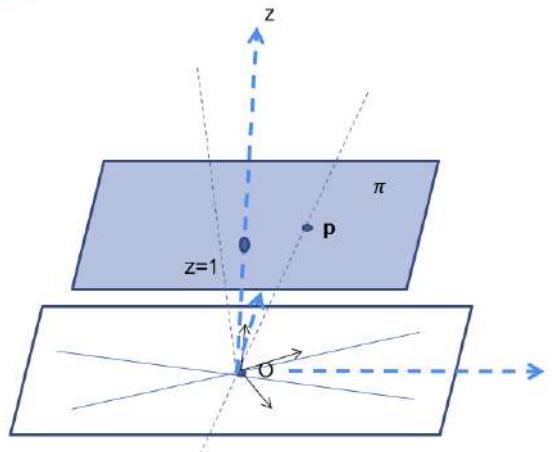
Projective plane:

Let's consider to go from 2 to 3 dimensions, and consider a plane π given in \mathbb{R}^3 by $z = 1$

Each point p represented in inhomogeneous coordinates with (x,y) in π is in biunivocal correspondence with the term $(x, y, 1)$ in $\mathbb{R}^3 - (0,0,0)$, the projective space called P^2
the line Op is the line of the coordinates (wx, wy, w) for each w

Thus, given S the set of lines through O

$$p = (x, y, 1) \in \pi \rightarrow \{(wx, wy, w) | w \in \mathbb{R}\} \in S$$



Vedi queste cose spiegate sul libro.

Consideriamo un nuovo piano parallelo al piano dell'immagine con distanza 1 dal piano dell'immagine.

Ogni punto appartenente a questo piano ha coordinate $(x,y,1)$ perché $z=1$. Se si moltiplica il punto p con qualsiasi numero diverso da zero si ottengono tutti i punti passanti per la retta che passa per Op .
Ogni punto di quella retta ha coordinate (wx,wy,w) con w appartenente a \mathbb{R} e diverso da zero.

(il centro O è il centro della camera ed è la base della prospettiva). Con questo tipo di rappresentazione si possono rappresentare tutti i punti appartenenti a tutte le possibili rette che attraversano il piano pigreco, tutte tranne le rette parallele al piano perché attraversano il piano pigreco in un punto all'infinito (sono paralleli). Le rette che intersecano il piano pigreco in un punto all'infinito vengono rappresentate con $w=0$.

Un punto $\mathbf{x}=(x,y)$ sul piano pigreco può essere rappresentato in coordinate omogenee nel seguente modo:

$$\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1) = \tilde{w}\bar{\mathbf{x}}$$

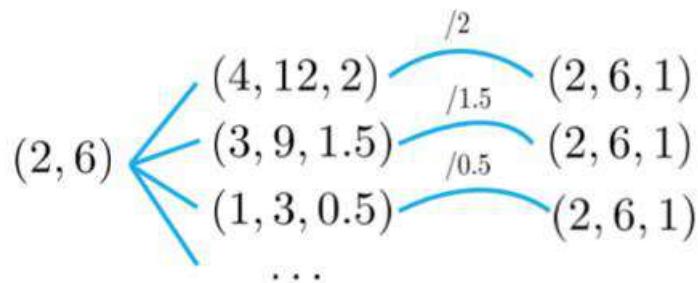
Inhomogeneous
coordinates homogeneous
coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\tilde{x} \in P^2$	is the homogeneous vector
$\bar{x} = (x, y, 1)$	is the augmented vector
$P^2 = R^3 - (0, 0, 0)$	is the projective 2D Plane

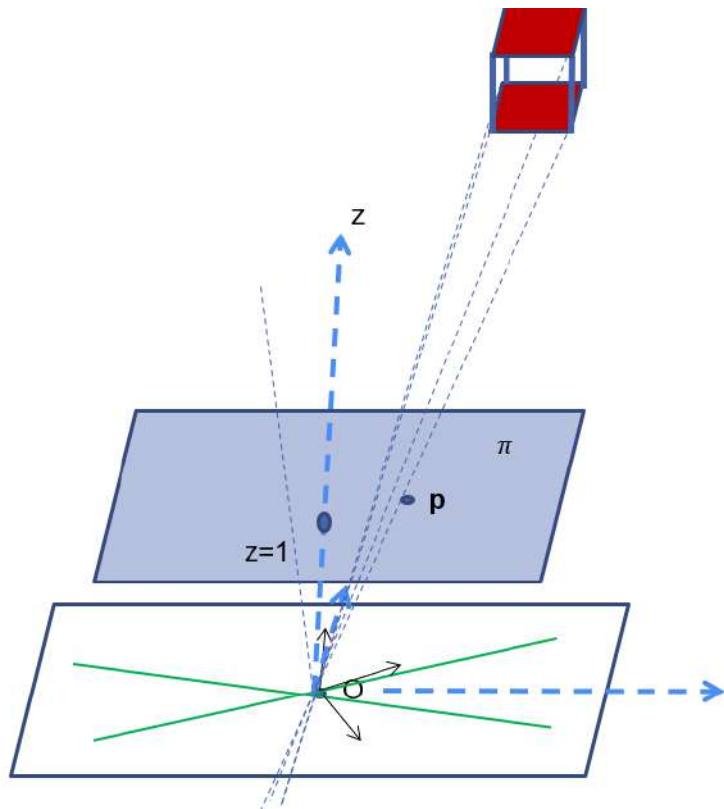
Nel piano proiettivo P^2 vettori che differiscono solamente per lo scalare w sono equivalenti. Questo significa che un punto nel piano potrebbe essere la proiezione di ogni punto della retta che interseca il piano in x, y con un certo w (questa è la trasformazione da 3D a 2D, ovvero tutti i punti di una retta vengono rappresentati con un unico punto).

Ad esempio, il punto $(2, 6)$ nel piano cartesiano corrisponde al punto $(2, 6, 1)$ nel piano proiettivo ed è equivalente a $(4, 12, 2)$ ecc..

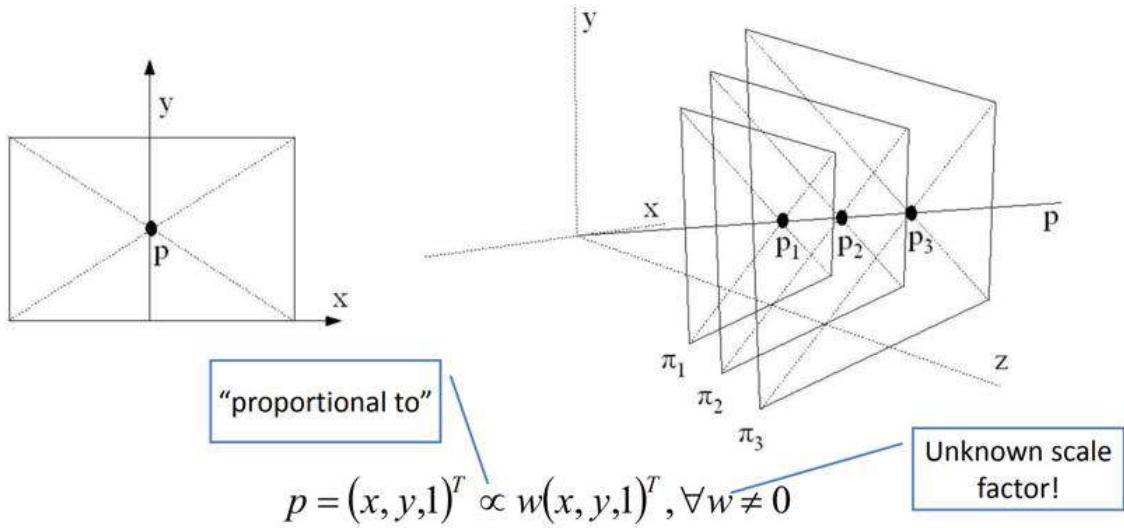


Ricordando che S è l'insieme di linee che attraversano O , alcune linee di questo insieme non intersecano il piano pigreco. (linee verdi)

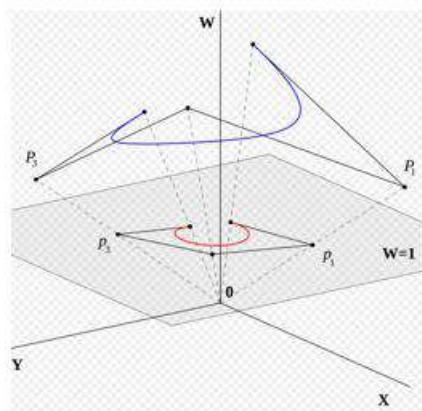
Tali linee sono parallele al piano pigreco quindi lo intersecano in un punto improprio all'infinito. I punti omogenei con $w=0$ sono chiamati punti ideali o punti all'infinito e non possono essere rappresentati in coordinate omogenee. Il piano proiettivo è un normale piano cartesiano a cui vengono aggiunti i punti all'infinito.



Sulla base di quanto appena visto possiamo dire che un vettore p rappresenta una classe di equivalenza di vettori (con differenti w).



Se noi consideriamo solo il piano con $w=1$ non possiamo capire la terza dimensione (perdiamo la terza dimensione), come succede nella videocamera con il piano dell'immagine (image plane).



Andiamo a vedere come avviene la conversione in coordinate omogenee:

1. Converting to *homogeneous* coordinates in 2D and 3D

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image
coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene
coordinates

2. Converting *from* homogeneous coordinates in 2D and 3D

2.1 If **w** non zero:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

2.2 If **w=0** no conversion(point at infinity)

Perché abbiamo bisogno di rappresentare i punti in coordinate omogenee? Perché così è più facile lavorare sui prodotti di matrice e rendere più semplice ogni computazione. Inoltre ogni trasformazione può essere rappresentata sotto forma di matrice. I punti in coordinate omogenee sono invarianti allo scale(proprio per l'equivalenza fra i vari w vista prima).

Points in homogeneous coordinates are invariant to the scale
Invariant to scaling

$$k \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{kx}{kw} \\ \frac{ky}{kw} \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}$$

Homogeneous Coordinates Cartesian Coordinates

Ritornando al nostro problema principale di rendere il template matching invarianto alle varie trasformazioni, andiamo ora ad analizzare le principali **trasformazioni 2D** che possono essere applicate ad un'immagine.

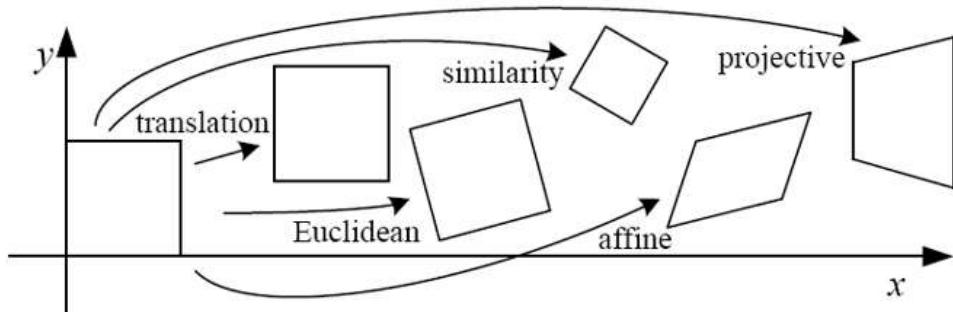


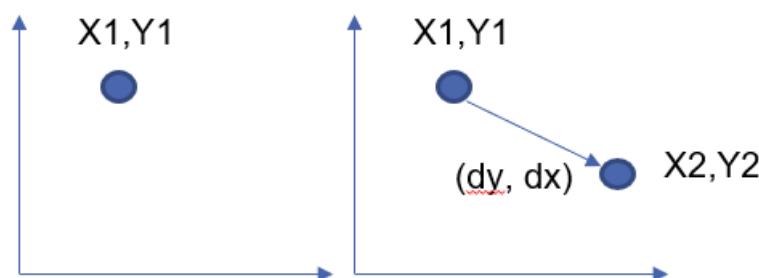
Figure 2.4: Basic set of 2D planar transformations

Traslazione 2D

In 2D la traslazione è data da:

$$X_2 = X_1 + dx$$

$$Y_2 = Y_1 + dy$$

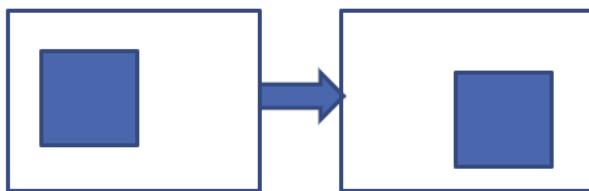


In coordinate non omogenee non è possibile definire una matrice T tale che: $P_2 = T^*P_1$. In coordinate omogenee, invece, la traslazione può essere ottenuta attraverso un prodotto matriciale: è possibile trovare una matrice T costituita dalla matrice identità e dal vettore di traslazione nel seguente modo:

$$\begin{bmatrix} X_2 \\ Y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ 1 \end{bmatrix}$$

$$\bar{x}_2 = \begin{bmatrix} I & t \\ O^T & 1 \end{bmatrix} \bar{x}_1 \quad x_2 = [I \quad t] \bar{x}_1$$

La traslazione mantiene invariati l'orientazione e l'area.



Traslazione e rotazione 2D

La trasformazione che esegue sia traslazione che rotazione contemporaneamente è chiamata “movimento del corpo rigido 2D” oppure “trasformazione euclidea 2D”.

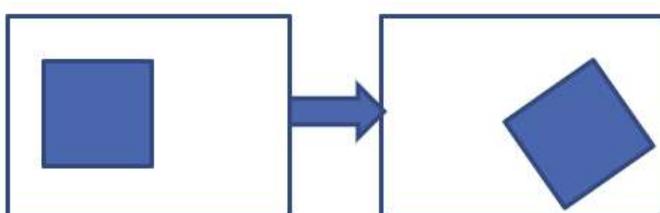
Definiamo R la matrice ortonormale di rotazione per cui valgono le seguenti proprietà:

$$RR^T = I \text{ and } |R| = 1$$

$$x' = \begin{bmatrix} R & t \end{bmatrix} \bar{x}$$

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

La trasformazione di euclidea mantiene invariata la lunghezza e la distanza euclidea.



Classe I: isometrie

Iso=stessa, metria=misura.

Tutte le trasformazioni euclidee sono isometrie. Tutte le trasformazioni appartenenti a questa classe possono essere rappresentate nella seguente forma generica:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \varepsilon \cos \theta & -\sin \theta & t_x \\ \varepsilon \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \varepsilon = \pm 1$$

orientation preserving: $\varepsilon = 1$

orientation reversing: $\varepsilon = -1$

$$x' = \mathbf{H}_E x = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{bmatrix} x \quad \mathbf{R}^T \mathbf{R} = \mathbf{I}$$

In questa matrice sono racchiuse traslazione, rotazione e verso di essa.

Le isometrie offrono **tre gradi di libertà** (DOF): 1 rotazione, 2 traslazioni. (Il numero **di gradi di libertà** corrisponde al numero **di** parametri indipendenti necessari per specificare la posizione o il movimento **di** ciascun corpo nel **sistema**).

Anche la sola traslazione e la sola rotazione sono particolari tipi di isometrie.

Le isometrie sono invarianti alla **lunghezza, all'angolo e all'area**.

There are **3DOF (1 rotation, 2 translation)**

special cases: pure rotation, pure translation

Invariants: **length, angle, area**

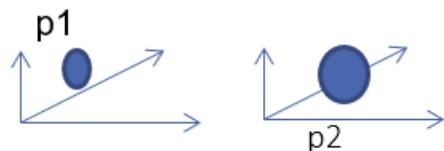


Scaling (ridimensionamento) 2D

Lo scaling in 2D è dato da un fattore s:

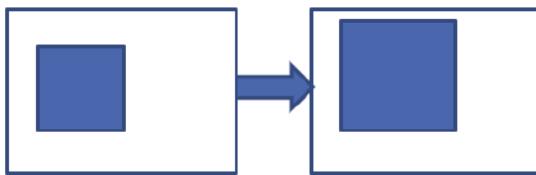
$$\overline{x_2} = [sI \quad O] \overline{x_1}$$

$$x_2 = [sI \quad t] \overline{x_1}$$



Se $s > 0$ l'oggetto viene ingrandito, se $s < 0$ l'oggetto viene rimpicciolito.

Lo scaling preserva l'orientazione ma non l'area.



Classe II: similarity trasformations (trasformazioni simili)

La similarity trasformation è la trasformazione che comprende scaling 2D, rotazione e traslazione in un'unica trasformazione. Ogni punto viene ruotato, scalato e traslato. In 2D è data da:

$$\bar{x}_2 = \begin{bmatrix} sR & t \\ O^T & 1 \end{bmatrix} \bar{x}_1$$

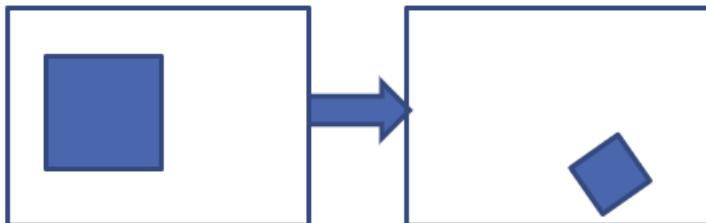
$$x_2 = [sR \quad t] \bar{x}_1$$

Scrivendo la trasmissione di similarità nel seguente modo:

$$x_2 = \begin{bmatrix} a & -b & tx \\ b & a & ty \end{bmatrix} \bar{x}_1$$

La differenza fra la Classe I e la Classe II è che il vincolo $a^2+b^2=1$, che era necessario per la classe I ora per la Classe II non è più richiesto.

Questa trasformazione preserva solo gli angoli:



La Classe II può essere anche vista come la classe I a cui è stato aggiunto anche lo scaling:

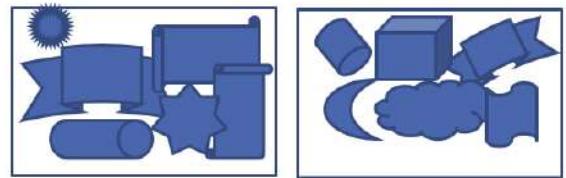
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s \cos \theta & -s \sin \theta & t_x \\ s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x' = \mathbf{H}_S x = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} x \quad \mathbf{R}^T \mathbf{R} = \mathbf{I}$$

Le trasformazioni di similarità offrono **quattro gradi di libertà** (DOF): 1 scaling, 1 rotazione, 2 traslazioni e sono anche chiamate equi-form (shape preserving) perché preservano la shape degli oggetti. Inoltre sono invarianti rispetto a lunghezza, angoli, area e linee parallele.

Invariants: **ratios of length, angle, ratios of areas, parallel lines**

*Do you recognize
Similar objects?*

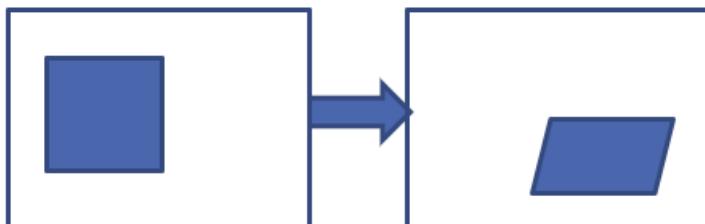


Classe III: trasformazioni affini

Se consideriamo le matrici delle trasformazioni viste fin ora e vi eliminiamo tutti i vincoli caratteristici delle trasformazioni precedenti (a e b non devono più essere multipli di seno e coseno ma possono essere qualunque numero) otteniamo la trasformazione affine che è definita in 2D da una matrice A generica 2x3:

$$\overline{x_2} = \begin{bmatrix} A \\ O^T \end{bmatrix} \overline{x_1} \quad \boldsymbol{x}_2 = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \overline{\boldsymbol{x}_1}$$

La trasformazione affine preserva solo le linee parallele, cioè se sono presenti linee parallele esse rimarranno tali anche dopo aver applicato la trasformazione mentre tutto il resto cambia.



La matrice può essere scritta nel seguente modo:

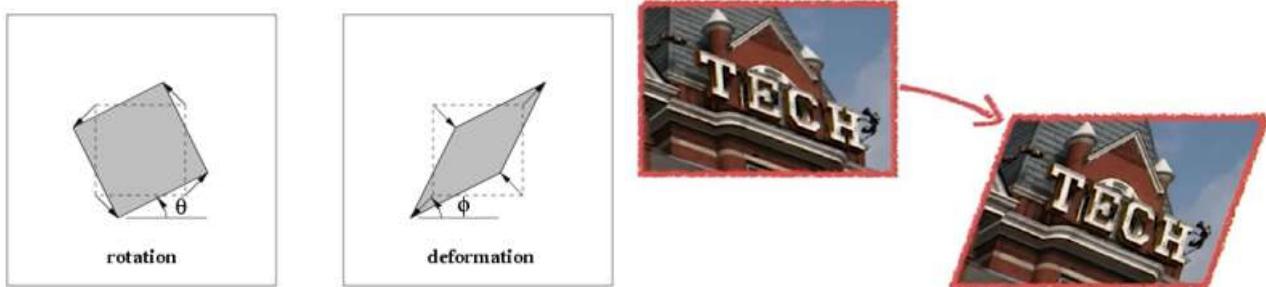
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \mathbf{x}' = \mathbf{H}_A \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

Le trasformazioni affini offrono **sei gradi di libertà** (DOF): 2 scaling, 2 rotazioni, 2 traslazioni. Inoltre sono invarianti rispetto a linee parallele, lunghezza di linee parallele e area.

There are 6DOF (2 scale, 2 rotation, 2 translation)

non-isotropic scaling! (2DOF: scale ratio and orientation)

Invariants: parallel lines, ratios of parallel lengths, ratios of areas



Classe IV: Trasformazione proiettiva o omografica

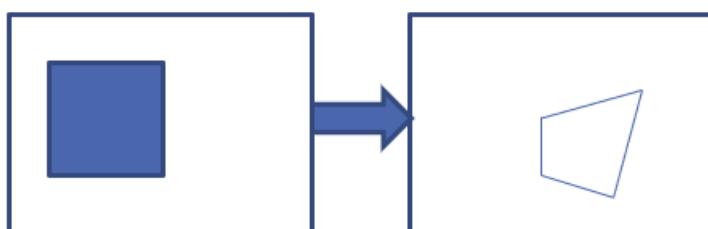
Questa trasformazione è definita attraverso la seguente matrice H 3x3:

$$\tilde{x}_2 = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tilde{x}_1$$

Essendo una matrice omogenea se due matrici H differiscono per un fattore di scaling esse sono equivalenti, quindi implementano la stessa trasformazione. Inoltre l'unico vincolo presente in questa matrice è che h_{33} sia diverso da zero ed in generale si pensa che sia uno perché tutti gli altri numeri possono essere divisi per h_{33} . Inoltre non essendoci vincoli sui numeri nella matrice le coordinate devono essere normalizzate:

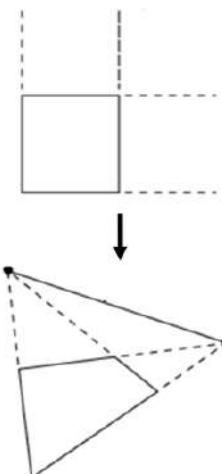
$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \text{ and } y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}.$$

Le trasformazioni proiettive preservano solo le linee rette.



$$\mathbf{x}' = \mathbf{H}_P \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^\top & v \end{bmatrix} \mathbf{x} \quad \mathbf{v} = (v_1, v_2)^\top$$

$$\tilde{\mathbf{x}}_2 = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tilde{\mathbf{x}}_1$$



There are 8 DOF (2 scale, 2 rotation, 2 translation, 2 line at infinity)
in general there are 8DOF since due to scaling h_{33} can be fixed to 1

Allows to observe
vanishing points,
horizon

Invariants: cross-ratio of four points on a line ratio of ratio

Action non-homogeneous over the plane

Le trasformazioni proiettive o omografiche offrono **otto gradi di libertà** (DOF): 2 scaling, 2 rotazioni, 2

traslazioni e 2 linee all'infinito. Inoltre sono invarianti rispetto a rapporto incrociato di quattro punti su un

rapporto lineare del rapporto. (?)

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	4	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2 \times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3 \times 3}$	8	straight lines	

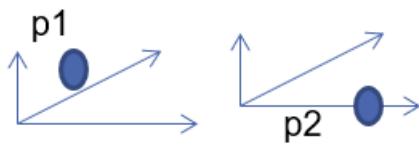
Vedi riassunto da slide 59

Ora andiamo a vedere le trasformazioni in 3D.

Traslazione 3D

La traslazione in 3D da un punto p1 ad un punto p2 è data da:

$$\begin{cases} X_2 = X_1 + dx \\ Y_2 = Y_1 + dy \\ Z_2 = Z_1 + dz \end{cases}$$



In coordinate non omogenee non esiste una matrice T che moltiplicata alle coordinate di p1 dia le coordinate del punto p2, mentre in coordinate omogenee la traslazione viene effettuata facendo il prodotto con la seguente matrice costituita dalla matrice identità e dal vettore di traslazione:

$$\bar{p}_1 = [X_1 \ Y_1 \ Z_1 \ 1]^T$$

$$\bar{p}_2 = [X_2 \ Y_2 \ Z_2 \ 1]^T$$

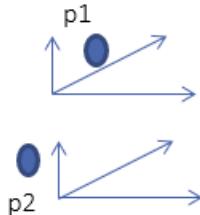
$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = T \cdot \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}$$

$$\bar{p}_2 = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix} \bar{p}_1$$

Rotazione 3D

La rotazione 3D lungo la direzione Z per il punto P1=(X1,Y1,Z1) è definita da:

$$\begin{cases} X_2 = X_1 \cos \vartheta - Y_1 \sin \vartheta \\ Y_2 = X_1 \sin \vartheta + Y_1 \cos \vartheta \\ Z_2 = Z_1 \end{cases}$$



e la corrispondente matrice di rotazione è:

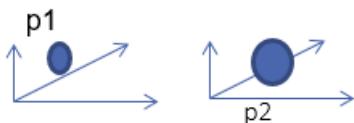
$$R_z(\alpha) = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 & 0 \\ \sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Allo stesso modo possiamo definire la matrice corrispondente alla rotazione attorno alla direzione X e attorno alla direzione Y:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha & 0 \\ 0 & \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\alpha) = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \alpha & 0 & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Scaling



Anche in 3D lo scaling è definito dalla seguente matrice:

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = S \cdot \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} \quad \overline{P}_2 = \begin{bmatrix} sI & 0 \\ 0^T & 1 \end{bmatrix} \overline{P}_1$$

Dove s è il fattore di scaling.

Anche in 3D è possibile definire le isometrie di **classe I** e le similarity trasformation di **classe II** che effettuano traslazione, rotazione e scaling nella stessa matrice 3x4.

Classe III: trasformazioni affini 3D

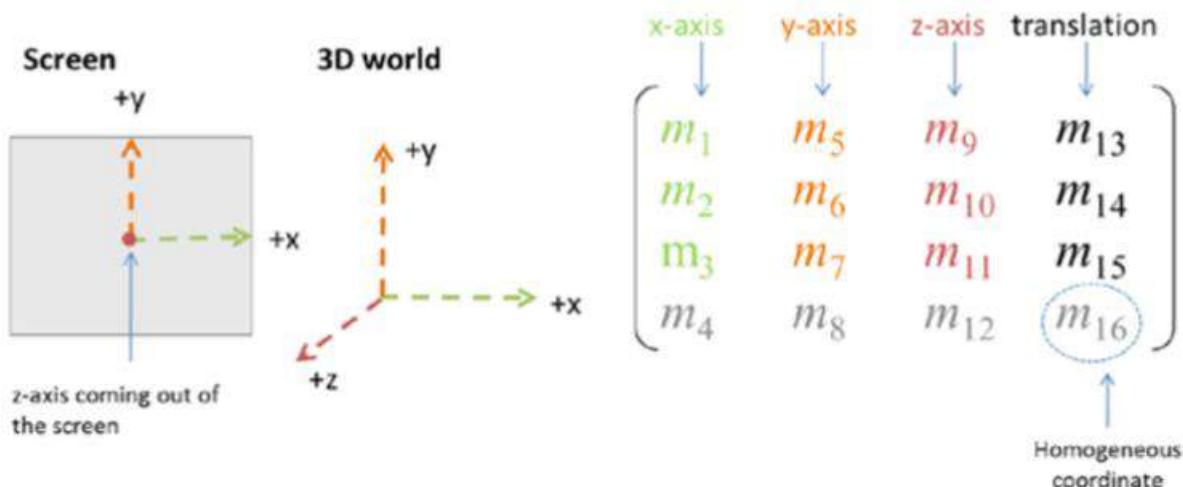
Anche in questo caso vengono preservate le linee parallele. Sono definite dalla seguente matrice 3x4:

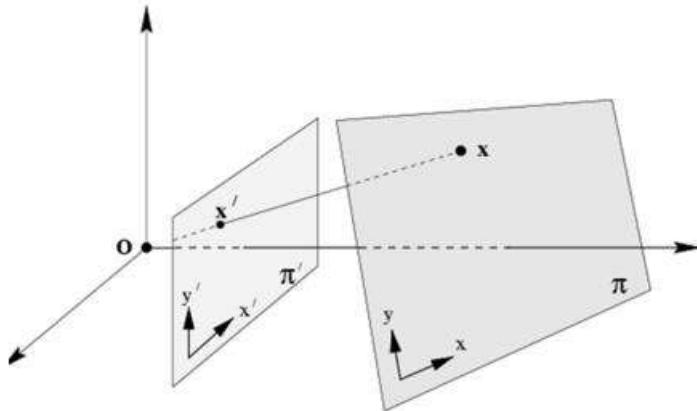
$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{bmatrix} \bar{x}.$$

Classe IV: trasformazioni proiettive 3D

Tali trasformazioni sono implementate da una arbitraria matrice omogenea H :

$$\tilde{x}' = \tilde{H}\tilde{x},$$





Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	3	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	6	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	7	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{3 \times 4}$	12	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{4 \times 4}$	15	straight lines	

Oltre ad applicare singolarmente le trasformazioni appena analizzate è possibile applicare **trasformazioni composte**. Le trasformazioni composte sono ottenute moltiplicando le coordinate per la matrice della trasformazione di ogni step. È importante seguire il corretto ordine $A \cdot B \neq B \cdot A$.

$$\textcolor{red}{T_c} = T_4 \cdot T_3 \cdot T_2 \cdot T_1$$

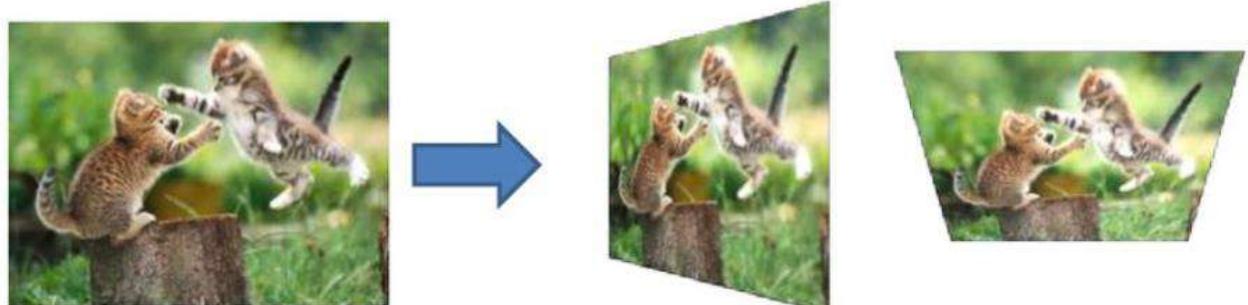
$$P_2 = \textcolor{red}{T_c} \cdot P_1 = T_4 \cdot T_3 \cdot T_2 \cdot T_1 \cdot P_1 = T_4 \cdot (T_3 \cdot (T_2 \cdot (T_1 \cdot P_1)))$$

Esempi:

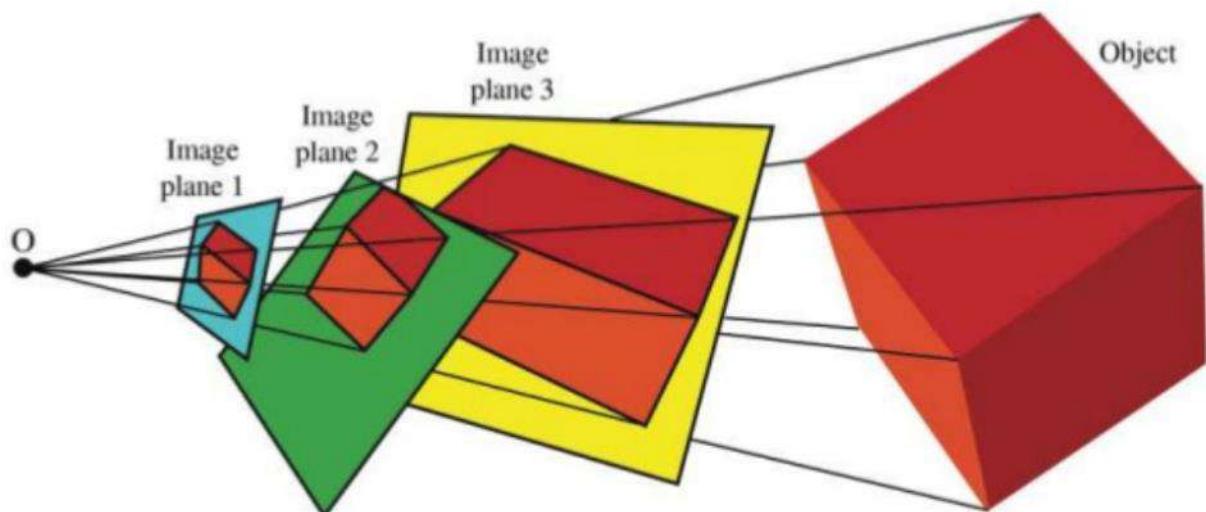
e.g. Euclidean transformations and affine transformation

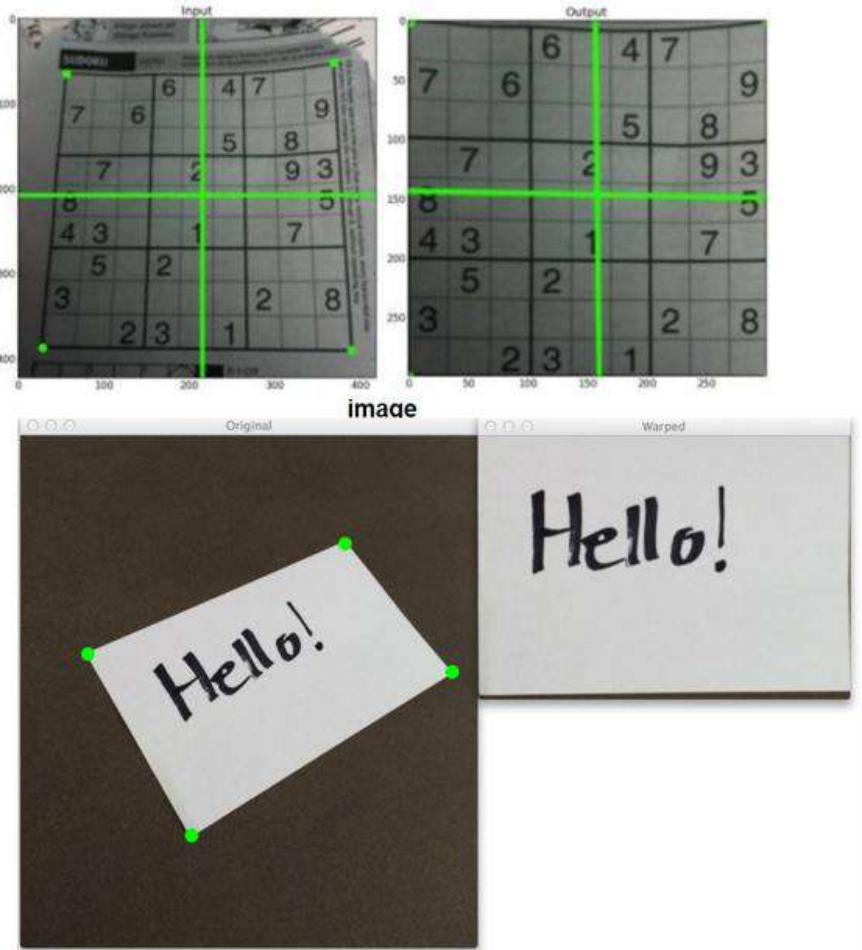


$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad \begin{aligned} x' &= u/w \\ y' &= v/w \end{aligned}$$



Le immagini formate da tutti i possibili piani che tagliano con inclinazioni diverse lo stesso fascio di raggi sono legate fra loro da trasformazioni omografiche, questo significa che per passare da una all'altra di quelle immagini devo applicarvi una trasformazione omografica.





HOMOGRAPHIC TRANSFORMATION

Come faccio a trovare la trasformazione omografica che mi consente di passare dal foglietto hello di destra a quello di sinistra? Devo trovare i valori tali per cui si crea una corrispondenza fra i quattro punti evidenziati in verde e i 4 angoli del cartellino. (?)

<http://www.ce.unipr.it/people/medici/geometry/node24.html>

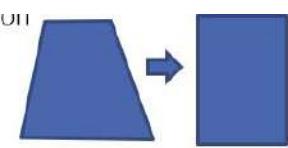
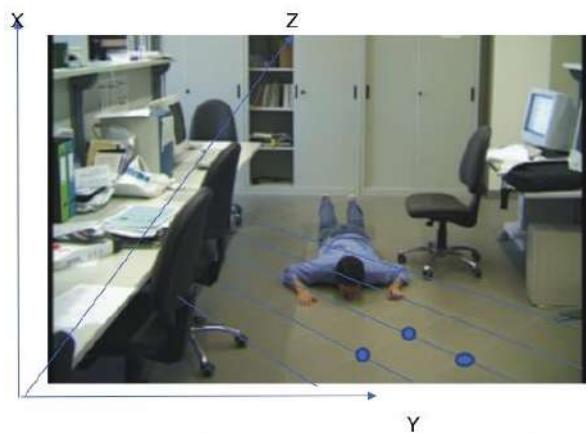
Esempio di applicazione

Se abbiamo un'immagine con distorsioni prospettiche non possiamo calcolare distanze reali (ad esempio vogliamo calcolare quanto realmente la persona sdraiata è alta) e le righe non sono parallele. Se calcoliamo una trasformazione che parte dal piano $X=0$ e fa una trasformazione omografica o prospettica per ottenere linee parallele verranno corretti anche tutti gli oggetti (ad esempio il corpo) nel piano $X=0$. Tutto ciò che non giace nel piano $X=0$ viene distorto dopo la trasformazione.

Come faccio a calcolare la trasformazione omografica che mi serve? Definisco 4 punti sul piano X e cerco di trovare la loro corrispondenza sul piano dopo la trasformazione. (ad esempio so che la piastrella dovrebbe essere quadrata quindi prendo i suoi angoli e cerco di spostarli fino a formare un quadrato).

In questo caso il piano che consideriamo è il pavimento:

Everything that is not lying in the plane X=0 is distorted after the transformation



This is the important transformation we will use to give true measures

Altro esempio:

Questa è la visuale della telecamera (con prospettiva):



Se abbiamo tre visuali diverse provenienti da tre telecamere diverse alle quali applichiamo una trasformazione omografica per preservare il piano del pavimento (vedere allo stesso modo in tutte e tre le camere e in modo frontale) la persona che passa risulta distorta in tre modi diversi (dipende da come era la prospettiva iniziale):



Ora vediamo un esempio di mosaik, ovvero una ricostruzione di un'immagine di un ambiente utilizzando più immagini dello stesso ambiente/oggetto prese da prospettive diverse.



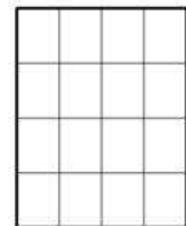
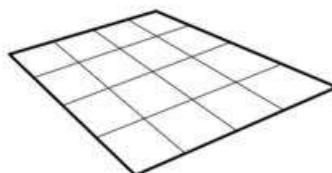
Ora entriamo più nel dettaglio delle trasformazioni omografiche per capire come vengono definite.

Un'immagine piana può essere trasformata in un'alta immagine piana:



$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} \mapsto (u/w, v/w) = (x', y').$$



Dobbiamo affrontare due problemi: come fornire la trasformazione parametrica e come calcolare la matrice H.

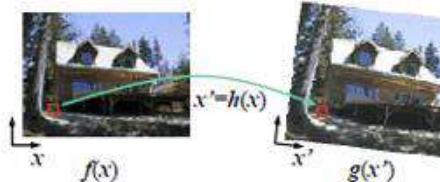
In particolare quando parliamo di voler definire la trasformazione parametrica ci chiediamo, data una trasformazione $\mathbf{x}' = \mathbf{H}\mathbf{x}$ e un'immagine $I = f(\mathbf{x})$ come faccio a creare una nuova immagine $I' = g(\mathbf{x})$? Abbiamo due possibilità: usando **forward algorithm** oppure **inverse algorithm**.

Forward algorithm

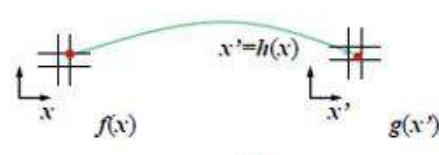
```
procedure forwardWarp( $f, h$ , out  $g$ ):
```

For every pixel x in $f(x)$

1. Compute the destination location $x' = h(x)$.
2. Copy the pixel $f(x)$ to $g(x')$.



(a)



(b)

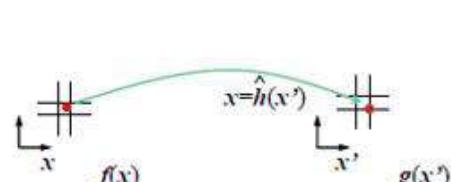
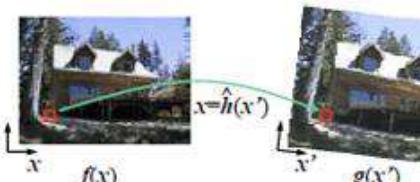
Questo algoritmo è il più semplice ma presenta alcune limitazioni: x' non può essere intero quindi è necessaria un’ulteriore trasformazione per distribuirlo nel vicinato (chiamata splatting), inoltre alcuni punti x' non possono essere definiti.

Inverse algorithm

```
procedure inverseWarp( $f, h$ , out  $g$ ):
```

For every pixel x' in $g(x')$

1. Compute the source location $x = \hat{h}(x')$
2. Resample $f(x)$ at location x and copy to $g(x')$



Questo algoritmo è definito in ogni punto (no buchi). Se la posizione iniziale non è un numero intero è possibile adottare alcuni metodi di interpolazione (vicinato, lineare, bicubica ecc.). Il kernel inverso utilizzato in questo caso è la matrice inversa di H.

Fin ora abbiamo visto come ottenere la trasformazione parametrica. Andiamo ora a vedere come si calcola la matrice H.

Poiché in 2D la matrice H è una matrice 3x3 dobbiamo trovare 9 valori. In particolare $h_{33}=1$ quindi dobbiamo trovarne solo 8. Per trovare 8 valori consideriamo 4 punti sullo stesso piano e non colineari nell’immagine di partenza e nell’immagine trasformata. Considerando 4 punti andiamo a tener conto di 8 coordinate trovandoci nel piano 2D.

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$



Select 4 points

$$P_0, P_1, P_2, P_3$$

$$P'_0, P'_1, P'_2, P'_3$$



$$H \rightarrow P_i \ i=0..3$$



$h_{00} \dots h_{21}$ are the variables

Dopo aver considerato tali punti scrivo le coordinate dei punti trasformati come espressioni dei valori della matrice h (che rappresentano le incognite del nuovo sistema) e delle vecchie coordinate.

$$\begin{aligned} x'_i &= h_{00} x_i + h_{01} y_i + h_{02} \\ y'_i &= h_{10} x_i + h_{11} y_i + h_{12} \\ z'_i &= h_{20} x_i + h_{21} y_i + 1 \end{aligned}$$

Divide with
the third eq.

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$



$$H \rightarrow P_i \ i=0..3$$



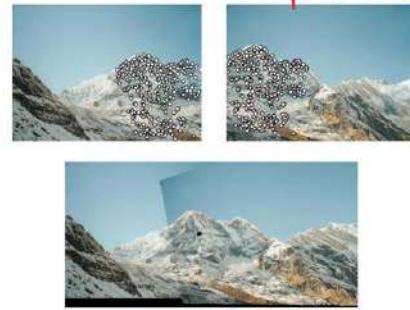
Dopo di che divido sia x che y per l'ultima equazione.

$$\begin{array}{l} \underline{x'_i = h_{00}x_i + h_{01}y_i + h_{02}} \\ y'_i = h_{10}x_i + h_{11}y_i + h_{12} \\ z'_i = h_{20}x_i + h_{21}y_i + 1 \end{array}$$

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$



H $\rightarrow P_i \ i=0..3$



$$\left\{ \begin{array}{l} x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + 1} \\ y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + 1} \end{array} \right.$$

Repeats for the 4 points ..

$$x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + 1}$$

$$y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + 1}$$

$$\left\{ \begin{array}{l} h_{00}x_i + h_{01}y_i + h_{02} - h_{20}x_i x'_i - h_{21}y_i z'_i = x'_i \\ h_{10}x_i + h_{11}y_i + h_{12} - h_{20}x_i y'_i - h_{21}y_i z'_i = y'_i \end{array} \right.$$



In matricial form ..

$$h_{00}x_i + h_{01}y_i + h_{02} - h_{20}x_i x'_i - h_{21}y_i z'_i = x'_i$$

$$h_{10}x_i + h_{11}y_i + h_{12} - h_{20}x_i y'_i - h_{21}y_i z'_i = y'_i$$

$$\left[\begin{array}{cccccc} x_0 & y_0 & 1 & 0 & 0 & 0 & -x_0 x'_0 & -y_0 x_0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -x_0 y'_0 & -y_0 y'_0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 x'_1 & -y_1 x_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 y'_1 & -y_1 y'_1 \\ x_2 & & & & & & -y_2 x_2 & \\ 0 & & & & & & -y_2 y'_2 & \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3 x'_3 & -y_3 x_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3 y'_3 & -y_3 y'_3 \end{array} \right] \left[\begin{array}{c} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \end{array} \right] = \left[\begin{array}{c} x'_0 \\ y'_0 \\ x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{array} \right]$$

A B



Più in generale, possiamo considerare $N \geq 4$ punti per trovare le 8 incognite che formano la matrice H:

Given $N \geq 4$ source points $\{(x_i, y_i)\}_{i=0}^{N-1}$ and N corresponding target points $\{(x'_i, y'_i)\}_{i=0}^{N-1}$ we have the following $2N$ equations and 8 unknowns:

$$\frac{h_{00} \cdot x_i + h_{01} \cdot y_i + h_{02}}{h_{20} \cdot x_i + h_{21} \cdot y_i + 1} = x'_i, \quad (3)$$

$$\frac{h_{10} \cdot x_i + h_{11} \cdot y_i + h_{12}}{h_{20} \cdot x_i + h_{21} \cdot y_i + 1} = y'_i. \quad (4)$$

We fixed $h_{22}=1$
Consider 4 points thus
8 coordinates
8 equations

Remember being in
A plane to plane
Trasformation 2D to 2D
It is working only if the 4
points are
on the same plane
But non collinear

We can rewrite these equations as a linear system for $i = 0, 1, \dots, N-1$:

$$h_{00} \cdot x_i + h_{01} \cdot y_i + h_{02} - h_{20} \cdot x_i \cdot x'_i - h_{21} \cdot y_i \cdot x'_i = x'_i, \quad (5)$$

$$h_{10} \cdot x_i + h_{11} \cdot y_i + h_{12} - h_{20} \cdot x_i \cdot y'_i - h_{21} \cdot y_i \cdot y'_i = y'_i. \quad (6)$$

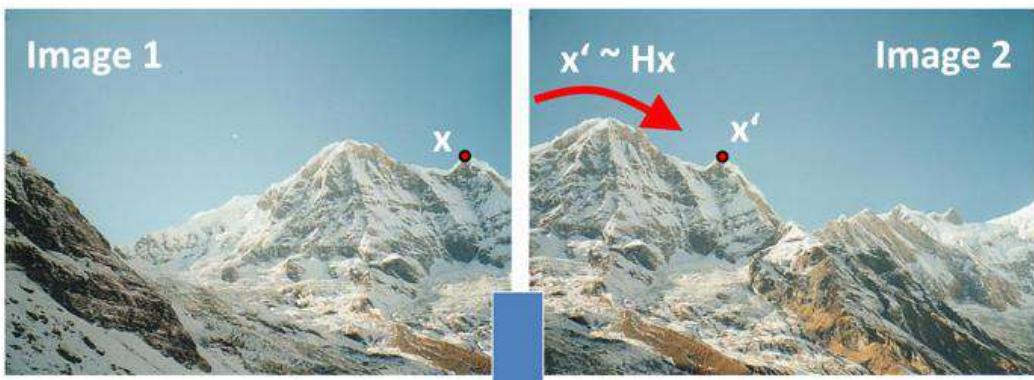
In matrix form we have $Ax = b$ where A is a $2N \times 8$ matrix and b is a $2N \times 1$ vector:

$$A = \begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & -x_0 \cdot x'_0 & -y_0 \cdot x'_0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -x_0 \cdot y'_0 & -y_0 \cdot y'_0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 \cdot x'_1 & -y_1 \cdot x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 \cdot y'_1 & -y_1 \cdot y'_1 \\ \vdots & \vdots \\ x_{N-1} & y_{N-1} & 1 & 0 & 0 & 0 & -x_{N-1} \cdot x'_{N-1} & -y_{N-1} \cdot x'_{N-1} \\ 0 & 0 & 0 & x_{N-1} & y_{N-1} & 1 & -x_{N-1} \cdot y'_{N-1} & -y_{N-1} \cdot y'_{N-1} \end{bmatrix} = \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \end{bmatrix} = \begin{bmatrix} x'_0 \\ y'_0 \\ x'_1 \\ y'_1 \\ \vdots \\ x'_{N-1} \\ y'_{N-1} \end{bmatrix} \quad (7)$$

where $x = [h_{00} \dots h_{21}]^T$ is our homography matrix in row-major order. When $N = 4$ (and no three of the source points are co-linear), solving $Ax = b$ is sufficient.

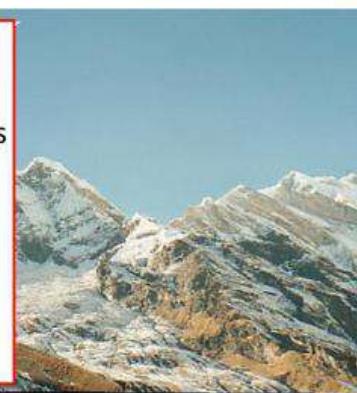
Ora andiamo a vedere le principali applicazioni delle trasformazioni omografiche.

1. "cucitura" di più immagini fra loro



Panorama stitching:

1. Undistort images
2. Find point correspondences between images
3. Compute homography H
4. Resample:
 1. Loop over image 1
 2. Project into image 2 using H
 3. Bilinear interpolation in image 2

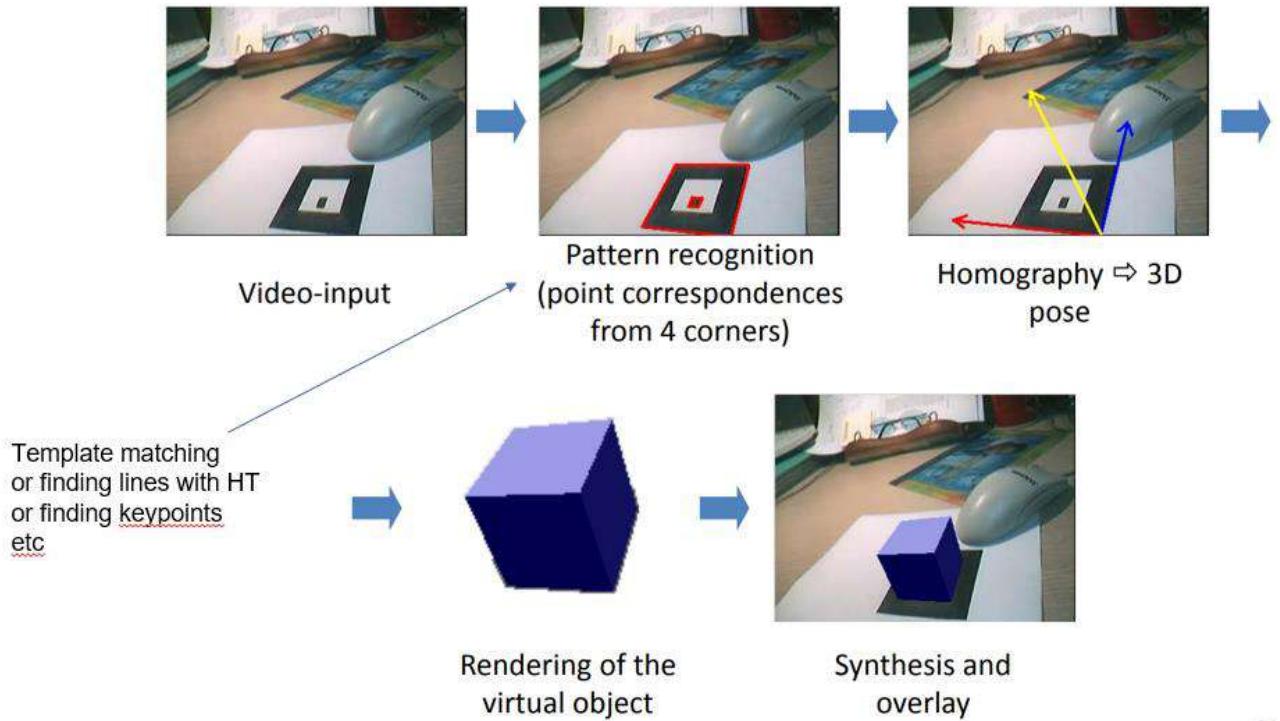


26

2. Estendere le operazioni di template matching

Dati un template e un'immagine è possibile forzare l'algoritmo di template matching creando tutte le possibili trasformazioni euclidiane, affini e omografiche e applicandole all'immagine per poi provare a fare il matching con il template. Questo metodo funziona abbastanza bene ma si può fare di meglio usando i descrittori locali.

3.realtà aumentata



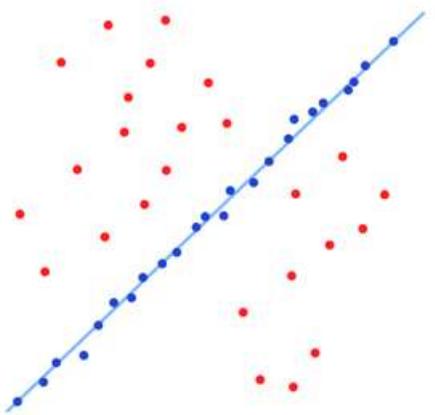
Fin ora abbiamo visto l'utilizzo di un modello basato su una **finestra di pixels più eventuale applicazione di geometria** per individuare una determinata shape. Tuttavia è possibile usare anche modelli basati su **forme geometriche** (HOUGH), modelli basati su **classi di shapes simili** (specifici detector e classificatori oppure keypoints locali e learning).

Ora andiamo a vedere i modelli che consentono la rilevazione di specifiche forme geometriche, ad esempio linee o cerchi. In molte immagini sono preset shapes che possono essere descritte da specifiche forme geometriche come linee cerchi o ellissi per questo motivo questi modelli sono utili. Questi modelli ricercano le forme geometriche nell'immagine o nella sola immagine dei bordi tuttavia considerano anche le trasformazioni affini/prospettiche. (come trovare una curva parametrica in un'immagine?)

Vediamo come rilevare linee con l'algoritmo **RANSAC ($O(N^3)$)**. Ransac (random sample consensus) è molto usato in computer vision però ci sono algoritmi molto più efficienti.

Ransac lavora su pochi punti, considerando un determinato feature space ridotto rispetto a quello dei punti dell'immagine iniziale (ad esempio l'immagine con i soli bordi o punti superiori ad una soglia).

Esso è uno dei più famosi metodi matematici iterativi per l'adattamento dei parametri di un modello matematico sulla base di un insieme di dati osservati contenenti sia valori che fanno match con il modello che outliers. È un algoritmo non deterministico, nel senso che produce un risultato ragionevole solo con una certa probabilità.



L'algoritmo consiste nel considerare due punti e definire la retta che passa per quei due punti. Dopo aver definito tale retta si conta quanti punti seguono quella retta (consenso). Ripeto questo per ogni possibile coppia di punti e la retta che avrà il consenso più alto è quella che definisco come modello. Provo tutte le possibili rette (modello) e per ognuna di quelle guardo quanti punti fittano il modello (consenso). A retta che avrà il consenso più alto corrisponde al mio modello, quindi alla retta che ho trovato nell'immagine.

Mmax=0;

For all pairs (e_1, e_2) in E

- find line equation passing through e_1 and e_2 ;

- $M=0$;

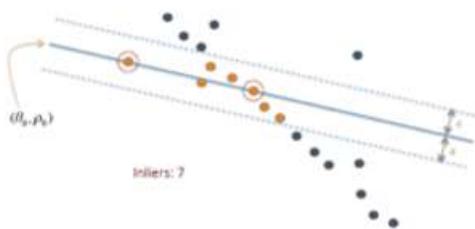
- For all e points in E

- if e fits the line (as a thresholded distance) $M++$;

- if $M > M_{max}$

- $M_{max}=M$; store e_1, e_2 ;

→ Iterative method; re-calculate the line



Questo viene ripetuto k volte, finché non si ottiene un buon adattamento. Trovo un piccolo sottoinsieme di punti e adatto il modello a quel sottoinsieme, poi calcolo quanti punti possono essere rappresentati da quel modello. Se sappiamo che il 50% dei punti che consideriamo sono outliers e adattiamo il modello a coppie casuali di punti, il 25% di tali coppie produrrà un'istanza del modello soddisfacente.

In general: the set of inliers obtained for the fitting model is **called consensus set**. (in the lines it is initially with 2 points). RANSAC iteratively repeat two steps until the consensus set in certain iteration has enough inliers.

1. - Select a random subset of the original data. Call this subset the *hypothetical inliers*. A model is fitted to the set of hypothetical inliers.
2. All other data are then tested against the fitted model. Those points that fit the estimated model well, according to some model-specific loss-function are in the *consensus set*.

The estimated model is reasonably good if the consensus set is large enough.
and the model may be improved by re-estimating it using all members of the consensus set.

The procedure is repeated a fixed number of times, each time producing either a model which is rejected because too few points are part of the consensus set, or a refined model together with a corresponding consensus set size. In the latter case, we keep the refined model if its consensus set is larger than the previously saved model.

The number of iterations depends on the probability of failure
See <https://www.youtube.com/watch?v=qN61qa3mPog> very high complexity..

Ora invece andiamo a vedere la **trasformata di Hough**. Questo è una classe di algoritmi che lavora in un feature space (gray level, color, edge, gradient, ecc.), ha performance migliori rispetto a Ransac ($O(N^2)$) e serve per rilevare curve parametriche all'interno di un'immagine.

Normalmente viene eseguita la **EHT edge-based Hough Transform** dopo aver eseguito la edge detection sull'immagine. Essa infatti corrisponde alla Hough transform sulla base dell'immagine contenente solo gli edge dell'immagine di partenza.

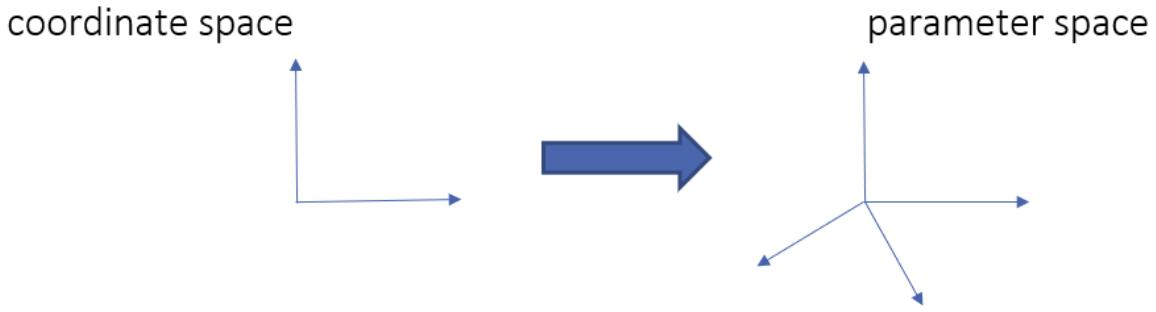
Possiamo utilizzare due tipi di Hough transform:

- **Trasformata di Hough classica** è capace di rilevare curve regolari come linee, cerchi, parabole, ellissi ecc. E' una tecnica per isolare curve facenti parte di una data shape in una data immagine. Richiede che le curve vengano espresse in forma parametrica.
- **Trasformata di Hough generalizzata** che può essere utilizzata nei casi in cui una semplice descrizione analitica delle features non è possibile.

La trasformata di Hough è in grado di rilevare curve parametriche ma non di localizzarle (non ne trova la posizione) tuttavia è abbastanza robusta al rumore e alle occlusioni nell'immagine ed è tollerante ai gaps negli edge.

L'idea è quella di fare una trasformazione da uno spazio verso un altro spazio in cui è più facile rilevare curve.

Data una shape geometrica (curva) rappresentata nella forma $f(x,p)=0$ dove x è il vettore delle coordinate nello spazio delle coordinate cartesiane e p è il vettore dei parametri nello spazio parametrico, la trasformazione di Hough trasforma i punti nello spazio delle coordinate in corrispondenti punti dello spazio parametrico in cui la curva viene evidenziata da un'accumulazione. Infatti tutti i punti che rappresentano la curva nello spazio delle coordinate sono rappresentate da un punto solo nello spazio parametrico aente coordinate corrispondenti ai parametri della curva.



$$F(x,p)=ax+by+c=0 \rightarrow (x,y) \text{ coordinate}$$

(a,b,c) parametri

Andiamo a vedere nello specifico come funziona la rilevazione di una **linea retta con la trasformata di Hough**.

Una linea retta viene rappresentata con l'equazione $y=mx+c \rightarrow mx+c-y=0$

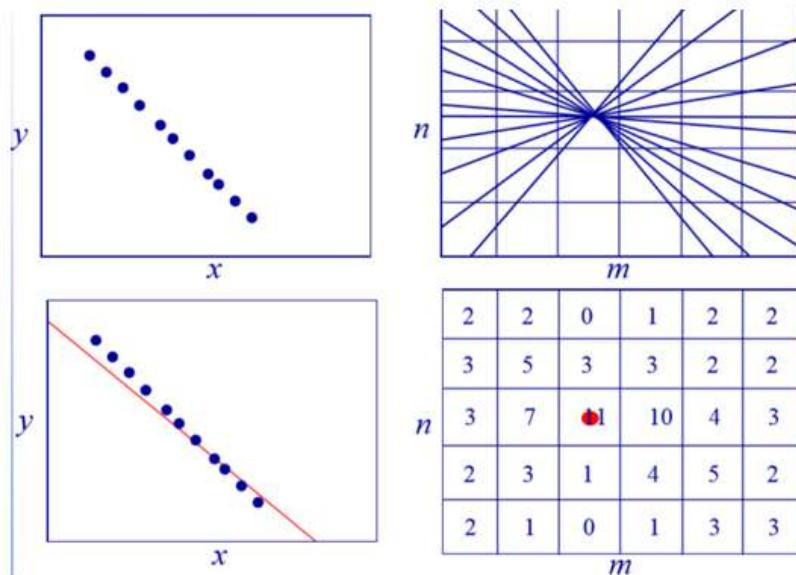
Ogni linea nel piano cartesiano corrisponde al punto (m,c) nel piano parametrico.

Ogni punto (x,y) nel piano cartesiano corrisponde ad una linea nel piano parametrico e ogni punto di questa linea fornisce i parametri per una delle infinite rette che passano per il punto (x,y) nel piano cartesiano.

I punti nello spazio cartesiano che passano per la stessa retta corrispondono alle linee che passano per lo stesso punto nello spazio parametrico. (tutti i punti della stessa retta nel piano parametrico cartesiano corrispondono tutte le rette passanti per il punto (m,c) nel piano parametrico. Quindi da una linea (infiniti punti) in uno spazio troviamo infinite linee in un altro spazio).

Lo spazio parametrico, considerato come uno spazio di accumulo discreto, è chiamato Hough space.

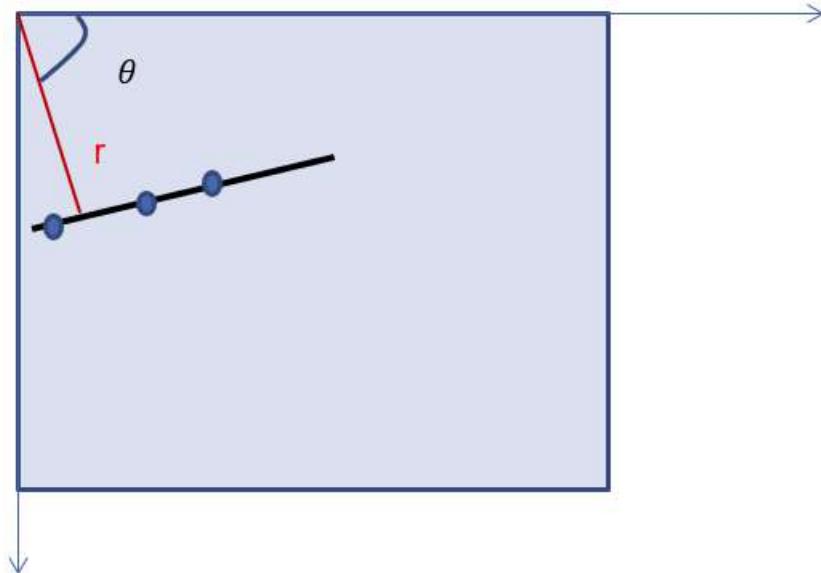
La Hough transform trasforma i punti (feature points) da punti nello spazio dell'immagine a punti nello spazio discreto di Hough dove i punti appartenenti alla stessa forma parametrica accumulano "voti" nello stesso punto dello spazio di Hough. Quindi ogni punto dello spazio dell'immagine vota per un punto nello spazio di Hough e punti sulla stessa retta voteranno per lo stesso punto. I punti dello spazio di Hough che hanno tanti voti corrispondono a potenziali parametri di una curva (retta in questo caso).



Questo metodo sarebbe molto semplice tuttavia non può essere usato perché non è possibile rappresentare linee parallele agli assi, in particolare per linee verticali in cui m è infinito.

Basandoci sempre sull'immagine composta solo dagli edges non utilizziamo più la rappresentazione $y=mx+c$ per le linee rette ma le rappresentiamo con **equazione polare**:

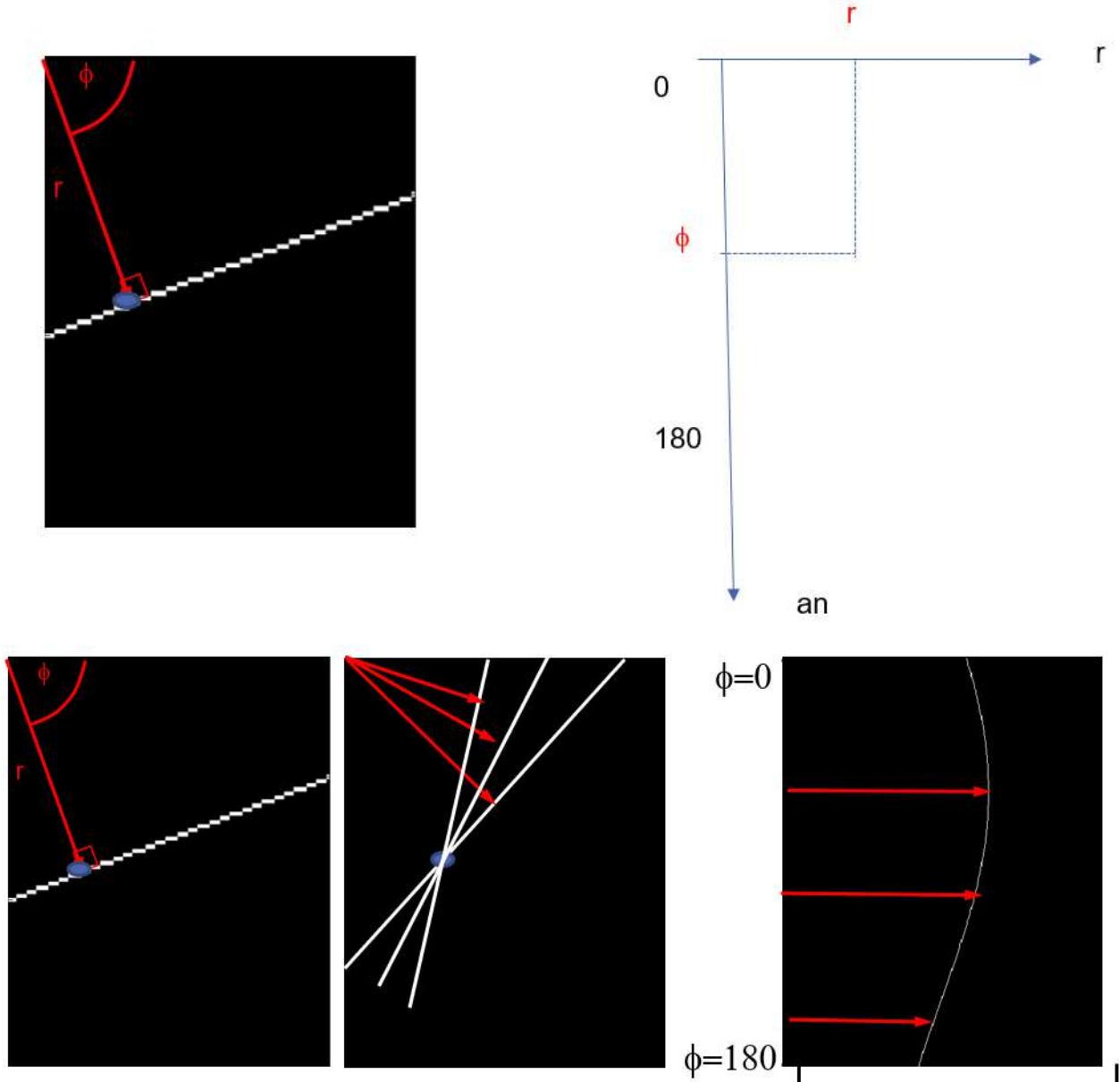
$$x\cos(\theta) + y\sin(\theta) = r$$



In questo caso i parametri che vanno a formare il nuovo spazio sono r e θ .

Tutti i punti di una linea sono rappresentati da una coppia di parametri, la distanza dall'origine r (distanza algebrica dovuta dal segno, può essere negativa) e l'angolo θ discretizzato. Qualsiasi linea nello spazio dell'immagine può essere rappresentata da una coppia (r, θ) che identifica la linea. È possibile fornire una trasformazione dallo spazio dell'immagine allo spazio parametrico, dove ogni linea dello spazio dell'immagine corrisponde ad un punto nello spazio parametrico. Per ogni punto nello spazio dell'immagine passano infinite rette (spazio discreto, non continuo) quindi ogni punto nello spazio dell'immagine identifica un numero infinito (sempre discreto) di coppie (r, θ) . Ogni punto nello spazio dell'immagine vota nello spazio parametrico per tutte le coppie corrispondenti alle rette che passano per quel punto nello spazio dell'immagine. L'insieme delle coppie (r, θ) identificate e votate da un punto forma una sinusoida e risulta ovvio che punti nello spazio dell'immagine appartenenti alla stessa retta voteranno anche per la stessa coppia. I punti che nello spazio parametrico sono maggiormente votati identificano i parametri (r, θ) corrispondenti ad una retta nello spazio dell'immagine.

Quindi ogni punto significativo dell'immagine vota per una sinusoida nello spazio parametrico. I punti che sono più votati sono quelli che corrispondono ai parametri di una retta presente nell'immagine.



L'algoritmo è il seguente:

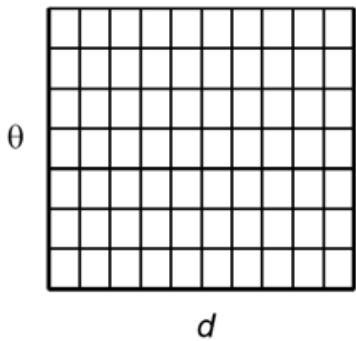
-Quantize the Hough Transform space: identify the maximum and minimum values of r and α_n (r_{min} , r_{max} , α_{min} , α_{max})

- Discretize the Hough space sampling steps D_r , D_α
- Generate an accumulator array $H(r, \alpha_n) = 0$ for all r and α ;

```
% create the Hough space
-For all edge points  $E(x_i, y_i) = 1$  in the image
  -For each angle  $\alpha_n$  in  $(\alpha_{min}, \alpha_{max})$ 
    {Compute  $r$  from the equation  $r = \text{quantize}(x_i \cos(\alpha_n) + y_i \sin(\alpha_n))$ ;
      $H(r, \alpha_n)++$ ; }

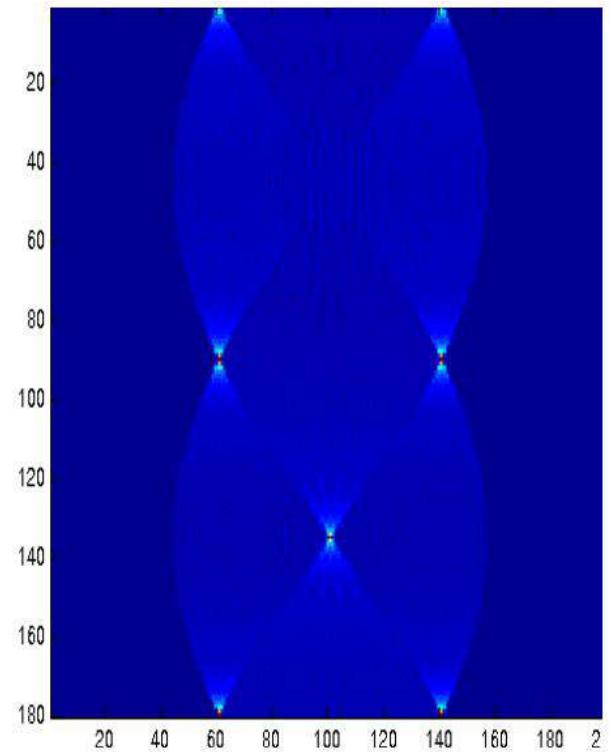
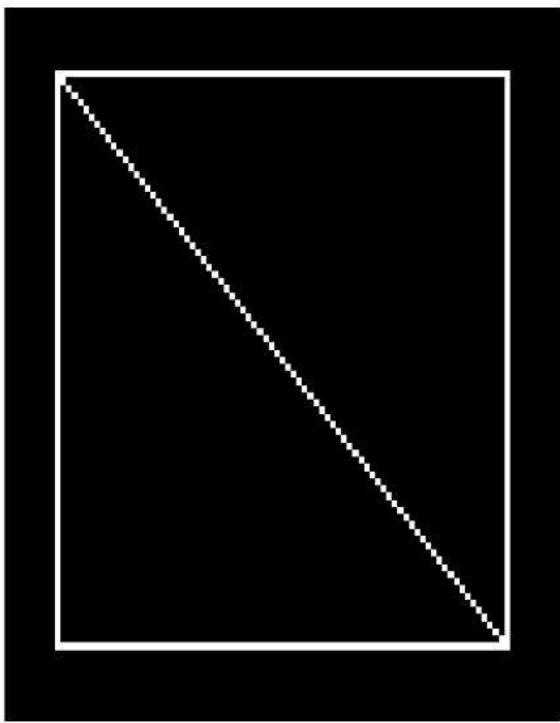
% search the lines
-For all cells in  $A(r, \alpha_n)$ 
  Search for the maximum value of  $A(r, \alpha_n)$  or over a threshold
  Calculate the equation of the line
```

H: accumulator array (votes)

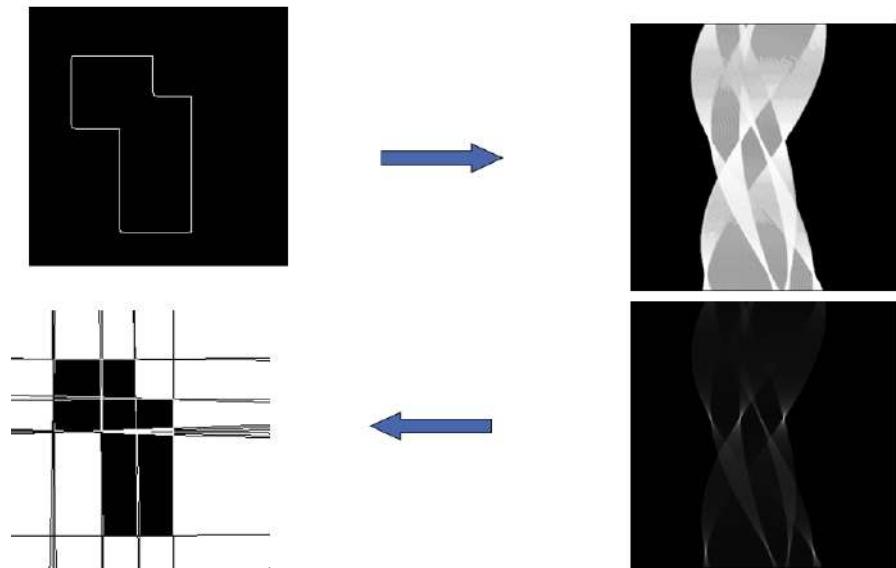


Esempio:

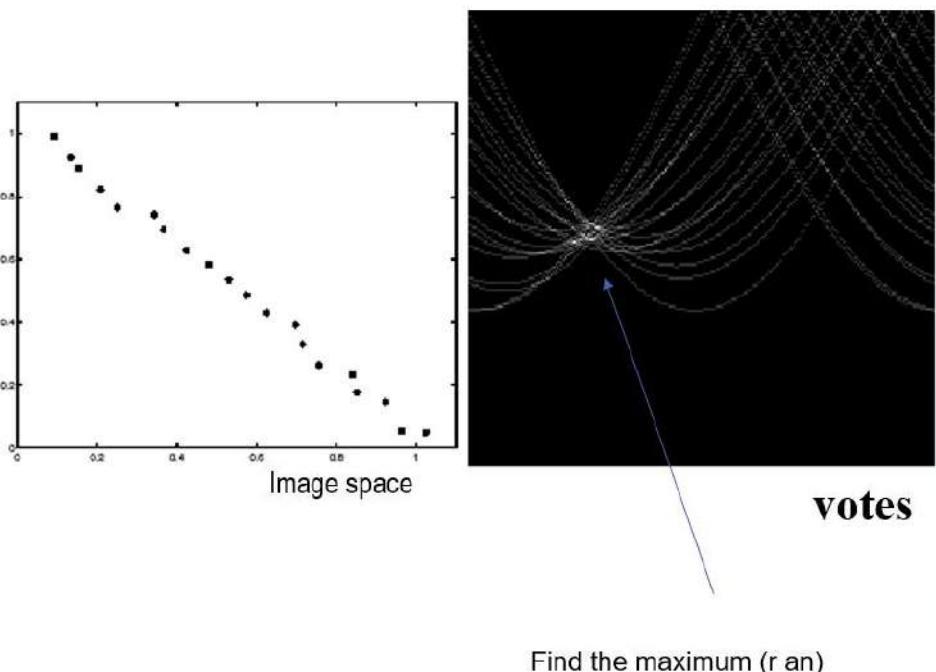
I punti più votati nello spazio parametrico (quelli bianchi) sono i punti che identificano le rette. Ogni punto è proporzionale (ha un numero di voti proporzionale) alla lunghezza della retta.



Nella seconda immagine l'asse verticale è θ mentre quello orizzontale è d . Il primo punto in basso a sinistra rappresenta la prima riga orizzontale in alto essendo $\theta=0$ e $d=60$ in quel caso. Il secondo punto andando verso destra, invece rappresenta la riga orizzontale più in basso, essendo sempre $\theta=0$ e d più grande (140). Allo stesso modo si può notare la corrispondenza di ogni punto con ogni retta.



Poiché avviene una discretizzazione dell'immagine i punti sulla stessa retta possono non essere precisissimi (vedi immagine) e inoltre possono esserci errori e può esserci rumore quindi la retta non è perfettamente precisa, per questa ragione spesso c'è più di un picco. I metodi utilizzati per trovare il punto nello spazio di hough che corrisponde alla linea retta che meglio si adatta ai punti discretizzati dell'immagine si utilizzano non maxima suppression, filtraggio dello spazio di Hough o applicazione di una soglia.

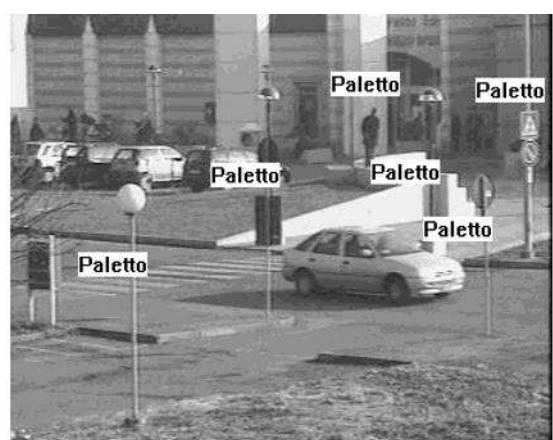
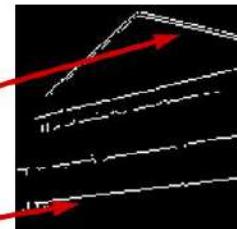
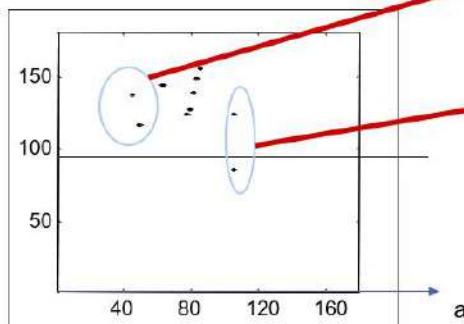


Esempi



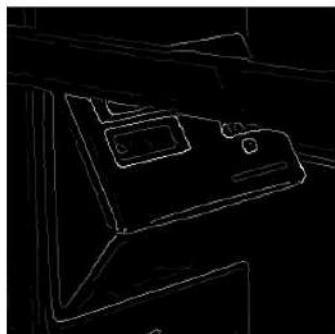
edge

H	r	a
59	-10	104
66	52	48
60	66	76
73	74	78
96	96	80
86	116	82

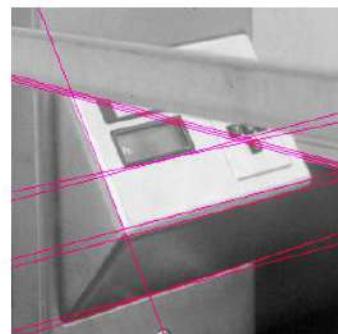




Original



Edge
Detection

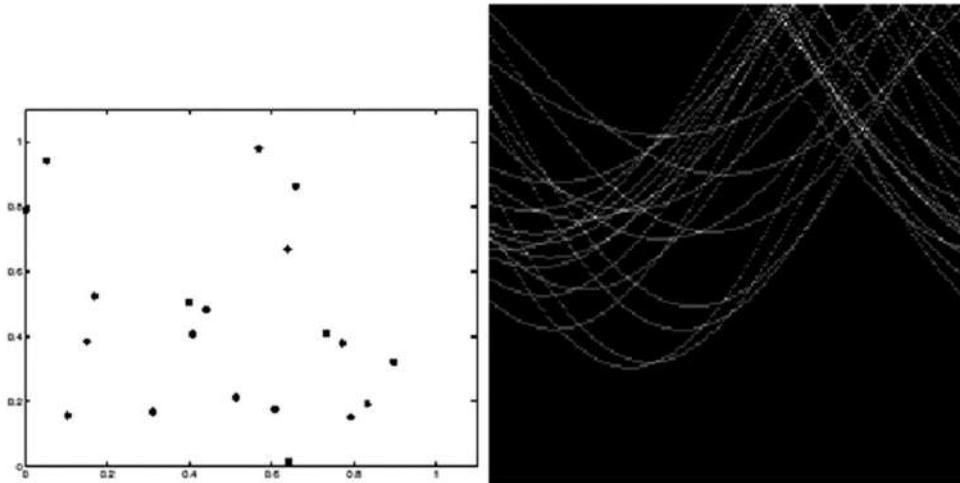


Found Lines



Parameter Space

Inoltre anche il rumore randomico può creare picchi randomici, ad esempio:



Fin ora abbiamo analizzato la edge based Hough transform. Andiamo ora a vedere alcune delle tante varianti di Hough transform esistenti.

Per ridurre l'effetto del rumore è possibile votare più di un elemento nell'array dei voti oppure applicare una post elaborazione all'immagine nello spazio di hough.

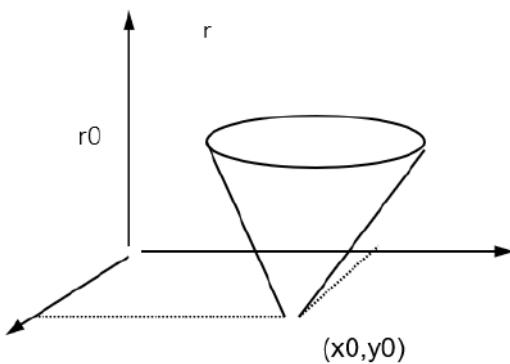
La **Gradient-based Hough transform GHT** (non ho capito) è più veloce perché ogni punto della curva parametrica nello spazio dell'immagine, invece di votare per tutte le possibili rette passanti per quel punto vota solo per la retta che è perpendicolare al gradiente calcolato in quel punto. Questo perché la direzione dell'edge è perpendicolare a quella del gradiente quindi riusciamo a ricostruire la direzione della retta per cui si deve votare risparmiando i voti per tutte le rette che non ci servono. Ovviamente potrebbe esserci del

rumore quindi si vota anche per qualche retta che ha direzione vicina a quella perpendicolare al gradiente. In più il voto può essere proporzionale alla magnitude del gradiente, in modo da non considerare gli edge più deboli.

Fin ora abbiamo visto la trasformazione di Hough usata per rilevare rette, tuttavia l'equazione parametrica della curva da rilevare può essere generalizzata.

Ad esempio, se vogliamo rilevare circonference l'equazione parametrica che useremo è

$(x - x_0)^2 + (y - y_0)^2 = r_0^2$. I parametri di questa equazione sono: $-a, b, r$ quindi lo spazio tridimensionale di Hough sarà generato da tali parametri. Ogni cerchio presente nello spazio delle immagini corrisponde ad un punto nello spazio di Hough (a, b, r) e un punto nello spazio delle immagini corrisponde ad un cono, cioè vota per tutti i possibili cerchi con ogni possibile raggio che hanno quel punto come centro. La curva ottenuta nello spazio di Hough per ogni punto presente nello spazio dell'immagine sarà un cono circolare retto. Il punto di intersezione di tutti i coni fornisce i parametri a, b, r del cerchio.



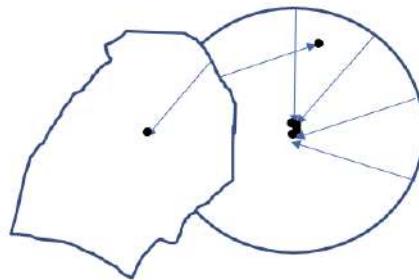
Soltamente però per il cerchio viene usata l'equazione polare:

$$x_0 = x_i - R \cos \theta$$

$$y_0 = y_i - R \sin \theta$$

Se r è noto (come spesso accade) lo spazio di hough diventa bidimensionale e ogni punto dello spazio dell'immagine vota per un cerchio avente come centro quel punto e come raggio r . L'intersezione fra tutti i cerchi dà il punto di coordinate (x_0, y_0) centro del cerchio dell'immagine.

Questa trasformazione è robusta alle viste parziali, ovvero se il cerchio non è intero ma appare solo in parte esso viene rilevato lo stesso. (robusto al rumore e robusto alle occlusioni).



Questo ragionamento può essere esteso a tutte le possibili curve parametriche:

Analytic Form Parameters Equation

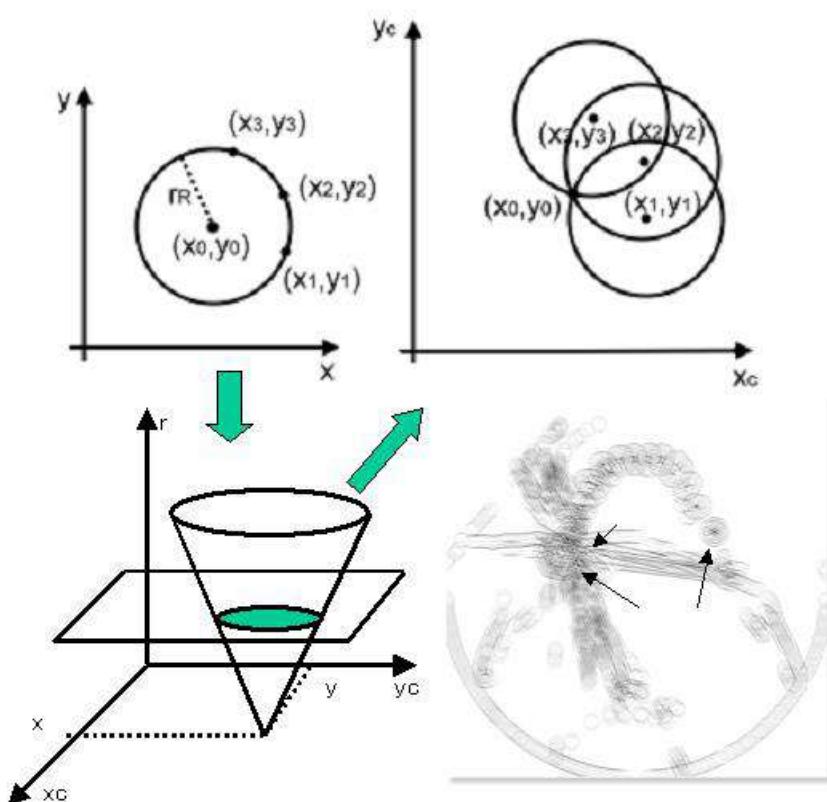
Line ρ, θ $x\cos\theta + y\sin\theta = \rho$

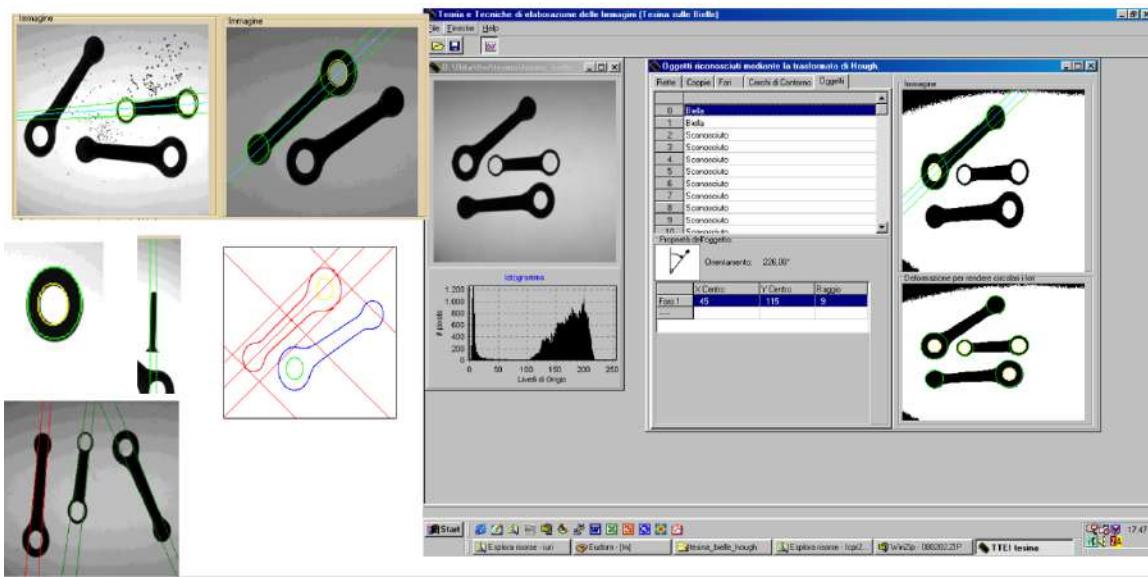
Circle x_0, y_0, r $(x-x_0)^2 + (y-y_0)^2 = r^2$

Parabole x_0, y_0, p, θ $(y-y_0)^2 = 4p(x-x_0)$

Ellipse x_0, y_0, a, b, θ $(x-x_0)^2/a^2 + (y-y_0)^2/b^2 = 1$

Esempi



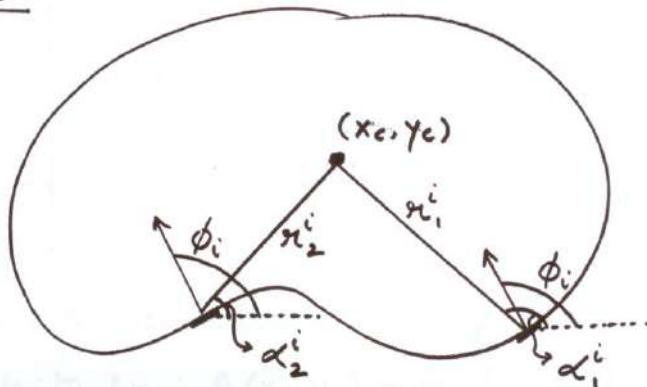


Oltre a generalizzare il modello per le curve parametriche note come circonferenze o ellissi è possibile generalizzare per curve arbitrarie.

Possiamo considerare una generica curva descritta in modo parametrico.

Rappresentiamo la generica curva nel seguente modo: dato un punto di origine xc, yc essa è una lista di punti rappresentati da una coppia (r_i, α_i) dove r_i è la distanza fra il punto i -esimo e il centro e α_i è l'angolo compreso fra l'asse e il raggio. Si crea un hash table per rappresentare tutti i punti.

Model :



(Nell'immagine phi è l'angolo fra l'asse e il gradiente.)

Per ogni possibile direzione del gradiente calcolata in ogni possibile punto della curva viene memorizzata la distanza dal centro e l'angolo con l'asse. Per una data direzione del gradiente, se la curva che consideriamo è arbitraria, potremo avere più punti della curva con r differente. (tradotto significa che punti diversi della curva hanno stessa direzione del gradiente). Tutte queste informazioni vengono salvate nella hash table che rappresenta la curva.

ϕ -Table

Edge Direction	$\vec{r} = (r, \alpha)$
ϕ_1	$\bar{r}_1^1, \bar{r}_2^1, \bar{r}_3^1$
ϕ_2	\bar{r}_1^2, \bar{r}_2^2
ϕ_i	\bar{r}_1^i, \bar{r}_2^i
ϕ_n	\bar{r}_1^n, \bar{r}_2^n

L'algoritmo che consente di rappresentare la curva generica con un punto x_c, y_c nello spazio di Hough è il seguente:

Given the edges (x_i, y_i, ϕ_i) find the object center (x_c, y_c)

1. Form an Accumulator array to hold the candidate locations of the reference point, initialized to 0 for each possible center

$$A(x_c, y_c)$$

2. For each point on the edge (x_i, y_i, ϕ_i)

1. Compute the gradient direction and determine the row of the R-Table it corresponds to, use the correspondent r
2. For each entry on the row calculate the candidate location of the reference point $x_c = x_i + r \cos \theta$
3. Increment the Accumulator value for that point $y_c = y_i + r \sin \theta$

3. The reference point location is given by the highest value in the accumulator array

Sostanzialmente, ogni punto nello spazio delle immagini vota il suo centro. Il punto che riceve più voti rappresenta la curva nello spazio di Hough.

La Hough transform generalizzata definita in questo modo, però, riconosce curve con stessa dimensione e stessa orientazione. Se scaliamo o ruotiamo la curva nell'immagine il modello così definito non la rileverà. Per rendere la trasformazione invariante allo scaling e alla rotazione dobbiamo modificare l'equazione che definisce il suo centro aggiungendo dei parametri:

$$x_c = x_i + r_k^i s \cos(\alpha_k^i + \theta)$$

$$y_c = y_i + r_k^i s \sin(\alpha_k^i + \theta)$$

$$A(x_c, y_c, s, \theta) = A(x_c, y_c, s, \theta) + 1.$$

Dove s è il fattore di scaling e θ è l'angolo di rotazione. In questo modo lo spazio di Hough avrà dimensione 4. Ogni punto di edge dell'immagine vota per più potenziali centri in base ai diversi possibili s e ai possibili θ . Il centro che viene scelto è definito dalle 4 coordinate ed è quello con maggior numero di voti.

Fin ora abbiamo visto la trasformazione di Hough su immagini 2D. E' possibile anche applicare tale trasformazione ad "immagini 3D". (non richiesto all'esame)

Hough transform in 3D

Using Kinect point cloud (x, y, z) $\mathbf{X} = \{X_j\}_{j=1, \dots, N}$ $X_j = (x_j, y_j, z_j)$.

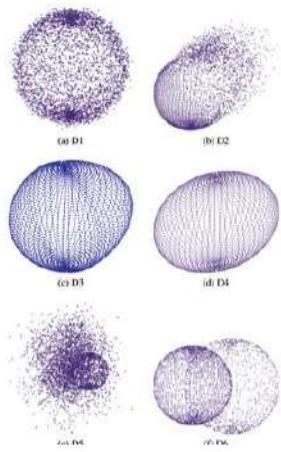
$$f(X, \Omega) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0.$$

$$p(X, \hat{\Omega}) = \begin{cases} 1, & \exists \Omega \in C_{\hat{\Omega}} : f(X, \Omega) = 0 \\ 0, & \text{otherwise} \end{cases}$$

$$H_1^{\hat{\Omega}}(\mathbf{X}) = \sum_{j=1}^N p(X_j, \hat{\Omega}),$$

Esempi:

Lane detection



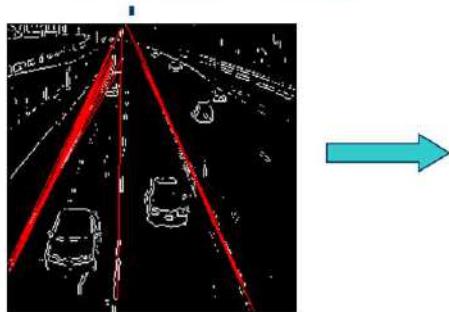
University Ave.



Loop 4 in Beijing



1. The red lines are the first 20 lines which have the biggest count numbers in Hough parameter space.
2. For each lane marking in the real scene, there are many line candidates around it.
3. There are some fake lines caused by the vehicle queue.



1. Sort the candidate lines by their position from left to right
2. Around each line cluster, choose the candidate which has the biggest count number as the lane marking in real scene
3. Delete the fake lane marking candidates
4. Calculate the mid-line of each lane (shown as green lines)





University Ave.



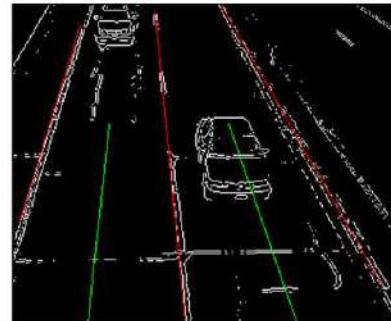
Edge image



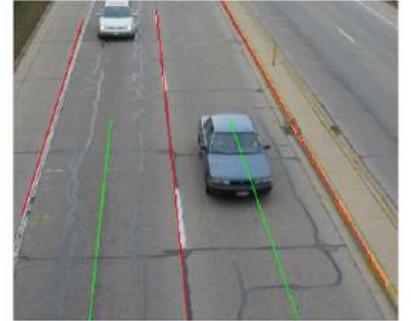
Without restriction to θ



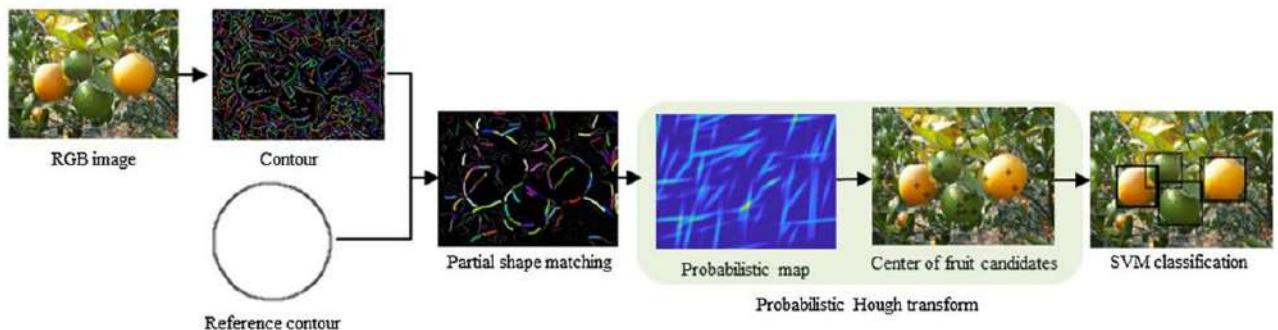
Restrict θ



Decide the lane markings



Detected lane markings



7.From image processing to AI-based vision (questo capitolo è da studiare con le slides sotto)

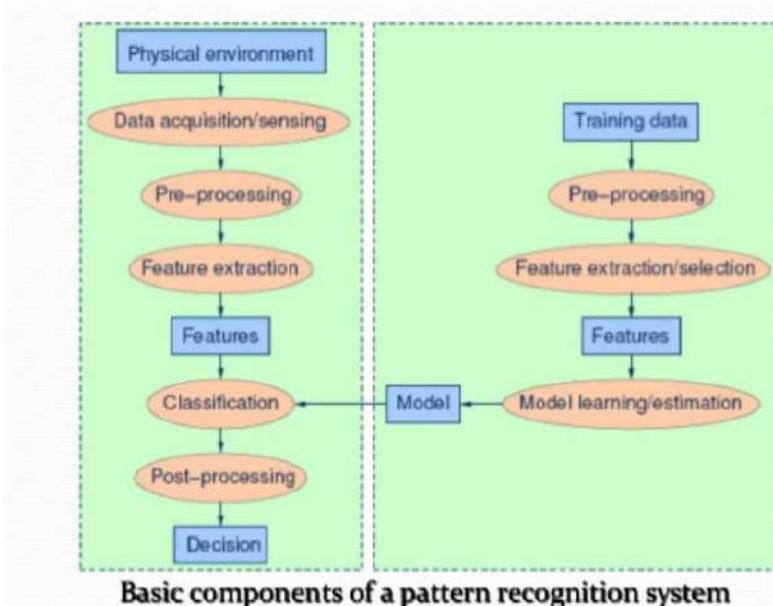
Premessa generale: studia l'esempio biometrics perché lo chiede

Nella prima parte di questo capitolo andiamo ad analizzare (anche tramite esempi) quali sono state le varie pipeline utilizzate per fare pattern recognition su immagini negli anni e quali sono gli approcci usati anche oggi.

La più vecchia pipeline di pattern recognition è la seguente:

The classical (old) pipeline in pattern recognition by images

1. Data preparation (pre-processing)
2. Feature extraction (to understand the model and define the inference engine)
3. Feature extraction for inference (e.g. classification)
4. Decision

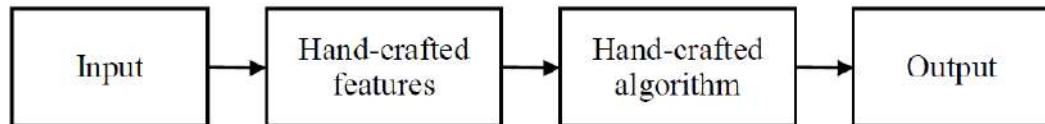


ora “pattern recognition” (che teoricamente sfrutta machine learning ed euristiche definite dall’uomo) e “machine learning” sono sinonimi:

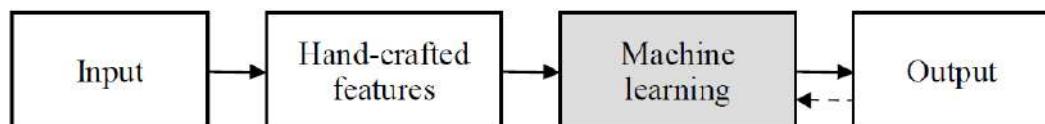


Machine learning techniques have always played an important and often central role in the development of computer vision algorithms. Computer vision in the 1970s grew out of the fields of artificial intelligence, digital image processing, and pattern recognition (now called machine learning), and one of the premiere journals in our field (*IEEE Transactions on Pattern Analysis and Machine Intelligence*) still bears testament to this heritage.

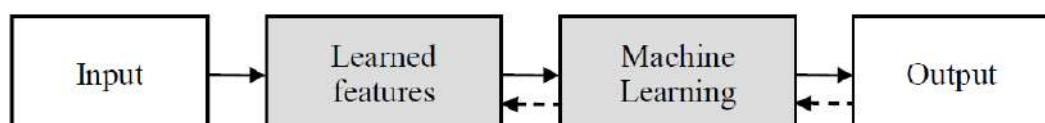
Negli anni la pipeline per lavorare su dati visuali è stata modificata dal punto di vista tecnico ma non nei task che la compongono:



(a) Traditional vision pipeline



(b) Classic machine learning pipeline



(c) Deep learning pipeline

Esempio di pipeline in campo biometrico per il riconoscimento di impronte digitali: slides 6-13.

Ora andiamo a vedere la visione dal punto di vista del processo cognitivo.

Statement 1: La visione richiede di **estrarre ciò che è caratteristico** di un target (scena, oggetto, evento oppure punti chiave): **features e descrittori**. I descrittori possono essere features vectors, tensori ma anche strutture più complesse come grafi. Le visual features dovrebbero essere rappresentative, precise, discriminatorie, invarianti e con cardinalità minima per accelerare il calcolo e fornire un ragionamento migliore concentrandosi solo sul necessario. La sfida è selezionare le features migliori per il target, selezionare la loro migliore descrizione per il processo di understanding e

Statement 2: la visione richiede un **motore di inferenza per fornire ragionamento** (associazione, memoria, deduzione logica): **Classificazione, retrieval, detection, understanding**.

Vision as a cognitive process slides: 17-21

I principali (e anche più difficili) task di intelligenza visuale sono:

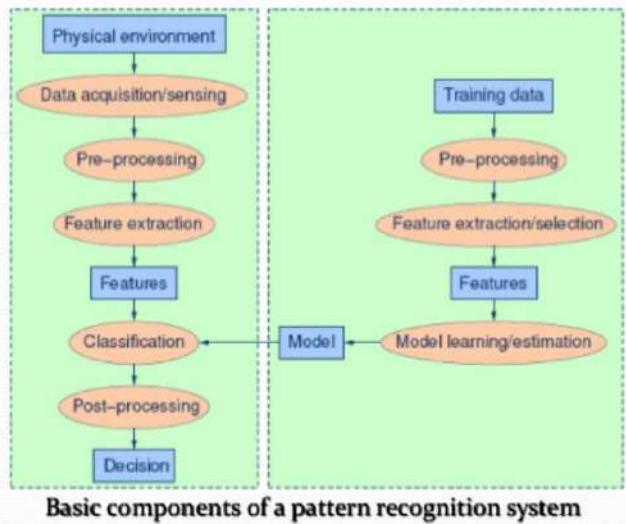
- Acquisizione dei dati visuali (pre-processing)
- Feature extraction
- Inferenza
- Azione

Andiamo a vedere come è cambiata negli anni la classica pipeline di intelligenza artificiale (di cui l'intelligenza visuale è un sottoinsieme). Elenco di pipeline e relativo campo di applicazione: 23-31

Andiamo a vedere ora due aspetti molto importanti dell'intelligenza artificiale che sono anche diventati oggetto di legislazione: l'affidabilità e la spiegabilità. Slides 32-47.

Torniamo ora ad analizzare il sistema di pattern recognition introdotto all'inizio del capitolo: i tasks che compongono la pipeline sono i seguenti:

1. Selection of DATA (Dataset, real time data, Data Augmentation...) and if necessary the DATA ANNOTATION (human /machine made, redundancy...)
2. Pre-processing (Human crafted or Ai based)
3. Feature extraction (hand-crafted features of deep learning based; architecture definition)
4. Model for classification/inference (definition of the measure and the Loss)
5. Post-processing (task based)
6. Explaining why...



Il task di model learning è un task di machine learning. Facciamo un riassunto delle principali tecniche di machine learning. Slides 49-58 (vedi cos'è eigenfaces).

Ora facciamo un elenco dei principali datasets utilizzati in computer vision e del preprocessing necessario: slides 59-67

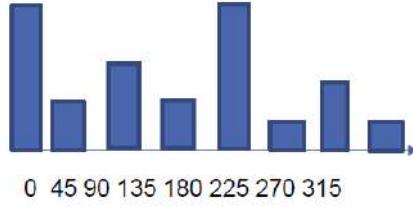
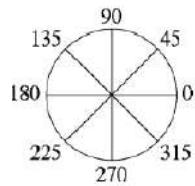
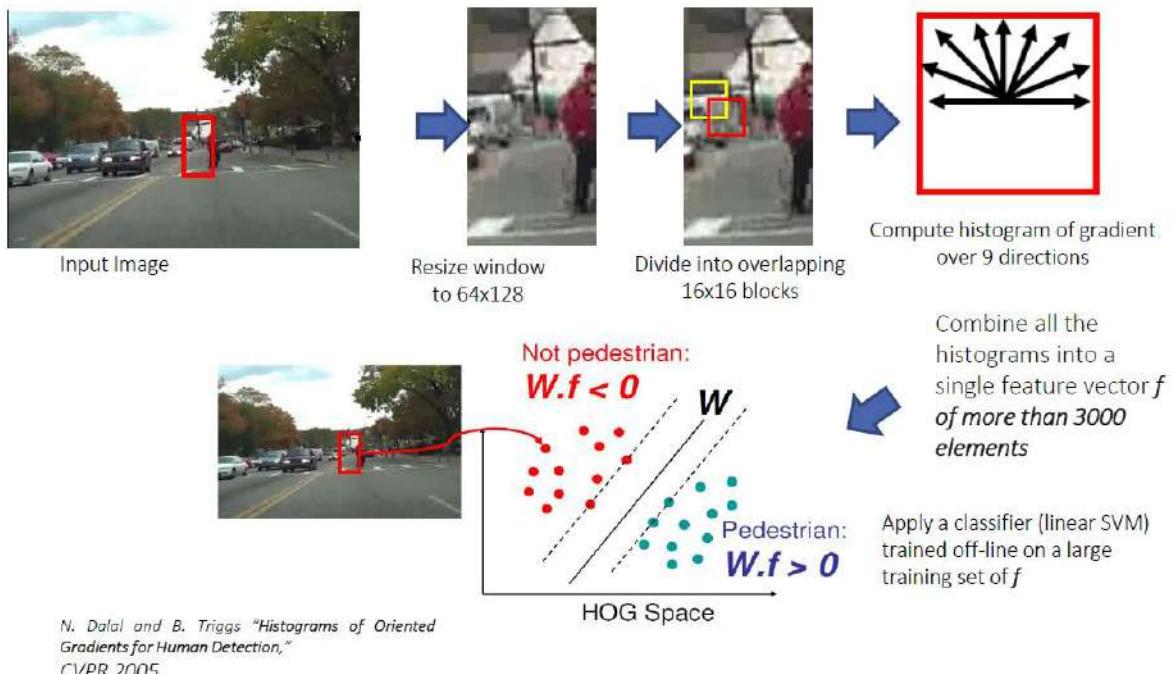
Hog e detection by hog: vedi slides 68-76

Con lo sviluppo del primo dataset raffigurante persone la prima soluzione robusta per la detection (nata per la detection di pedoni su scenario stradale, poi utilizzata per la detection di qualsiasi cosa) è arrivata. Usa una feature handcrafted per detectare la forma di una persona o di quello che dobbiamo detectare e si basa sul fatto che la forma di una persona (o di qualunque altra cosa) è più o meno costante.

L'idea è quella di utilizzare l'istogramma del gradiente. Sostanzialmente è l'istogramma che ha come bins le possibili direzioni del gradiente calcolato in ogni pixel dell'immagine e come valore la loro frequenza. Calcolando l'istogramma del gradiente in questo modo, però, si perde la dimensione spaziale quindi viene applicato il seguente metodo (hog + sliding window su ogni possibile porzione dell'immagine):

Si ridimensiona l'immagine o la porzione di immagine considerata 64x128 e su tale immagine si considerano tante finestre 16x16. Per ognuna di queste finestre si calcola il gradiente e si calcola l'istogramma del gradiente considerando solo 9 bins corrispondenti a 9 intervalli di direzione (o 9 direzioni). Facendo l'esempio della persona, avendo per lo più edges verticali l'istogramma del gradiente nella porzione che raffigura la persona identifierà un numero elevato in corrispondenza della direzione orizzontale del gradiente (direzione dell'edge e del gradiente sono perpendicolari).

Una volta fatto questo calcolo per ogni porzione si mettono in fila gli histogrammi trovati e questo vettore corrisponde appunto al feature vector dell'immagine che andrà in ingresso ad un SVM alleata per detectare persone (o quello che si vuole detectare). In questo vettore così formato sono presenti anche le informazioni posizionali relative a quale finestra dell'immagine si sta rappresentando, quindi se viene detectato qualcosa si riesce a ricostruire in quale punto dell'immagine si trova.



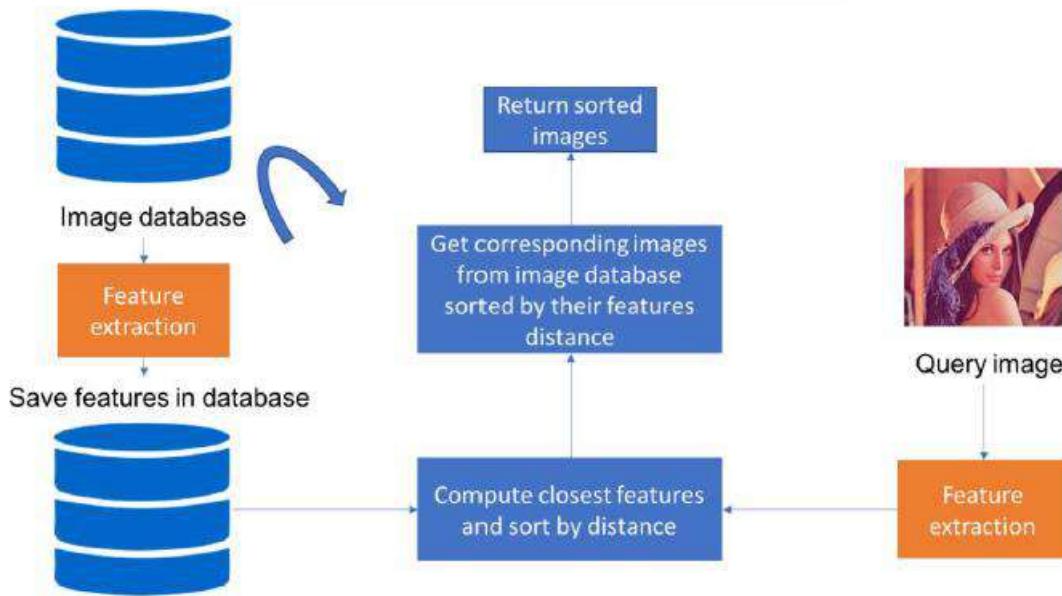
Spiegazione fatta bene: <https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/>

Altri datasets: slides 77-89

Retrieval: slide 89-99

Image retrieval è diverso da image classification. L'image retrieval consiste nel trovare gli oggetti simili all'immagine che sto considerando ed è molto complesso perché per fare ciò è necessario definire un modo per rappresentare l'immagine, definire una misura di similarità e calcolare tale similarità.

Le features che descrivono un'immagine possono essere hand crafted oppure learned based. L'insieme delle features utilizzate per rappresentare l'immagine è chiamato embedding e ci sono diversi metodi per costruirlo (slide 91). Image retrieval è anche conosciuto come content-based retrieval o query-by-example. Questo è un problema di corrispondenza fra immagini che consiste nel recuperare dal database immagini simili ad una data immagine (query image). La somiglianza fra immagine di riferimento e immagini del database viene utilizzata per classificare il database in ordine decrescente di somiglianza, quindi le prestazioni di ogni sistema di retrieval dipendono dalle metriche di similarità utilizzate. Questo problema è un problema non ancora concluso, e sono presenti diverse soluzioni applicate in diversi campi.

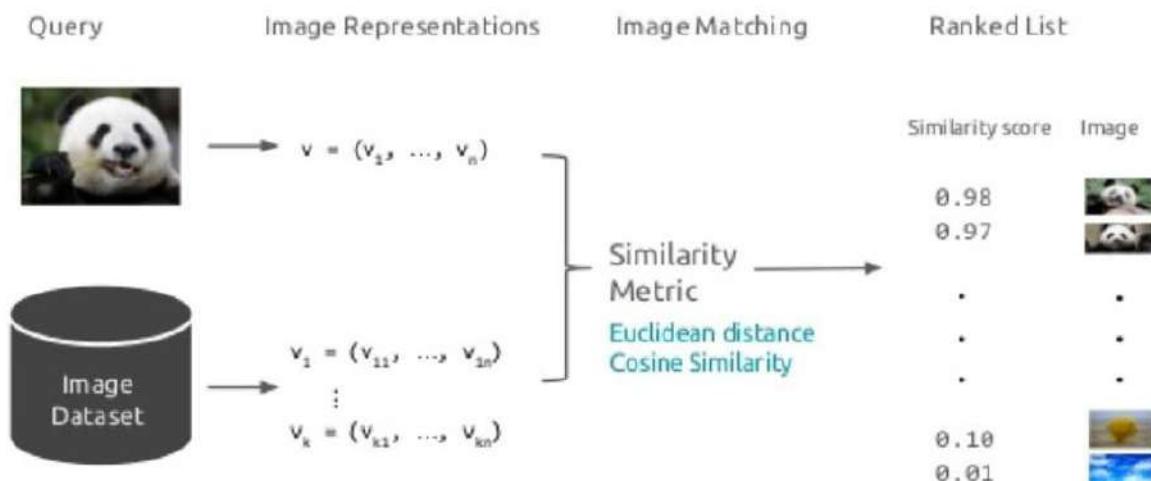


Le principali sfide sono quelle di definire la similarità corretta, garantire una determinata velocità per l'operazione e cercare di raggiungere la maggiore scalabilità possibile essendo spesso i database molto grandi e/o distribuiti.

Un esempio di feature vector per rappresentare l'immagine può essere la concatenazione dei valori dei pixel in ogni canale del color space e in questo caso la similarità è data dalla distanza fra pixels. Tuttavia questo metodo non è molto efficiente.

Come già detto più volte la pipeline di retrieval è la seguente:

The retrieval pipeline



1. creazione del feature vector

2. metriche di similarità

3. score e performance

Il retrieval non viene eseguito a livello di oggetto o di regione ma a livello di immagine/frame.

Esempio biometrics slides 98-99

Partiamo analizzando gli score e le performance elencando le varie misure esistenti usate anche nella classificazione e nel machine learning in generale.

Performance in generale: slides 100-104

Performance in biometrics e parentesi su biometrics: slides 105-112

L'identificazione è un problema di classificazione con N classi, l'autenticazione è un problema di classificazione a due classi (you and not you) e può anche essere visto come un problema di retrieval con una soglia di accettabilità (per dire che quella impronta è talmente simile alla mia da essere della stessa persona). Questo si traduce in due aspetti: accettare la persona corretta e rifiutare le persone sbagliate.

Delle specifiche metriche usate in biometrics sono:

- True Positive Rate (TPR) also referred as True Acceptance Rate (TAR) = $TP / (TP+FN) = TP / P$ (=recall)
- False positive rate (FPR) also referred as False Acceptance Rate (FAR) = $FP / (FP+TN) = FP / N$

L'idea è quella di volere basso FAR, il più possibile vicino a 0 e alto TAR, il più possibile vicino a 1. Di solito TAR@0.001 FAR =95% è un tipico requisito. (significa se su 100 persone posso accettare 5 falsi negativi e 1 falso positivo? Non sono sicura di questo).

Un aspetto molto importante è scegliere la soglia di accettazione da utilizzare.

Misure di performance per retrieval: slides 113-124

R-precision è la precisione calcolata in un dato rank. Ad esempio 10-precision è calcolata sui primi 10 risultati.

Misure di performance in recognition: slides 125-132

Misure di similarità: come misuro la similarità (o distanza) fra due feature vectors: slides 132-148

Con la distanza euclidea tutte le features hanno lo stesso peso, non ce n'è una più importante di un'altra. Lo stesso vale per il coseno.

Moving earth distance è usata per feature vectors che sono simili se sono shiftati. Non compara bin a bin i due vettori ma consente di paragonare bins che sono shiftati nello spazio. Compara gli istogrammi nell'insieme non bin a bin.

Scelta del feature vector: slides 150-190

In computer vision generare il corretto feature vector è molto difficile quindi spesso sono scelti metodi deep per generarlo. Esistono diversi tipi di feature vectors: Global descriptors (l'intera immagine), local descriptors (regioni di interesse, oggetti), local invariant descriptors (keypoints) che possono essere costruiti in diversi modi: a mano (hand-crafted), imparati, imparati con metodi deep.

Il più semplice descrittore è l'istogramma. E' anche possibile ottenere un feature vector da una rete neurale togliendo gli ultimi layers di classificazione. Si può trainare una rete per fare determinate classificazione poi utilizzare la stessa rete già trainata eliminando gli ultimi layer di classificazione per ottenere dei feature vector che verranno classificati in altro modo e riguardano contenuti diversi rispetto ciò che classificava la rete inizialmente. Questo è possibile perché quello che viene estratto nei primi layers non dipende da la

tipologia di oggetto rappresentato nell'immagine. Per ottenere un feature vector si fa il maxpooling sulle varie mappe di attivazione (vedi slides 162-163)

Feature vector sift: https://it.wikipedia.org/wiki/Scale-invariant_feature_transform

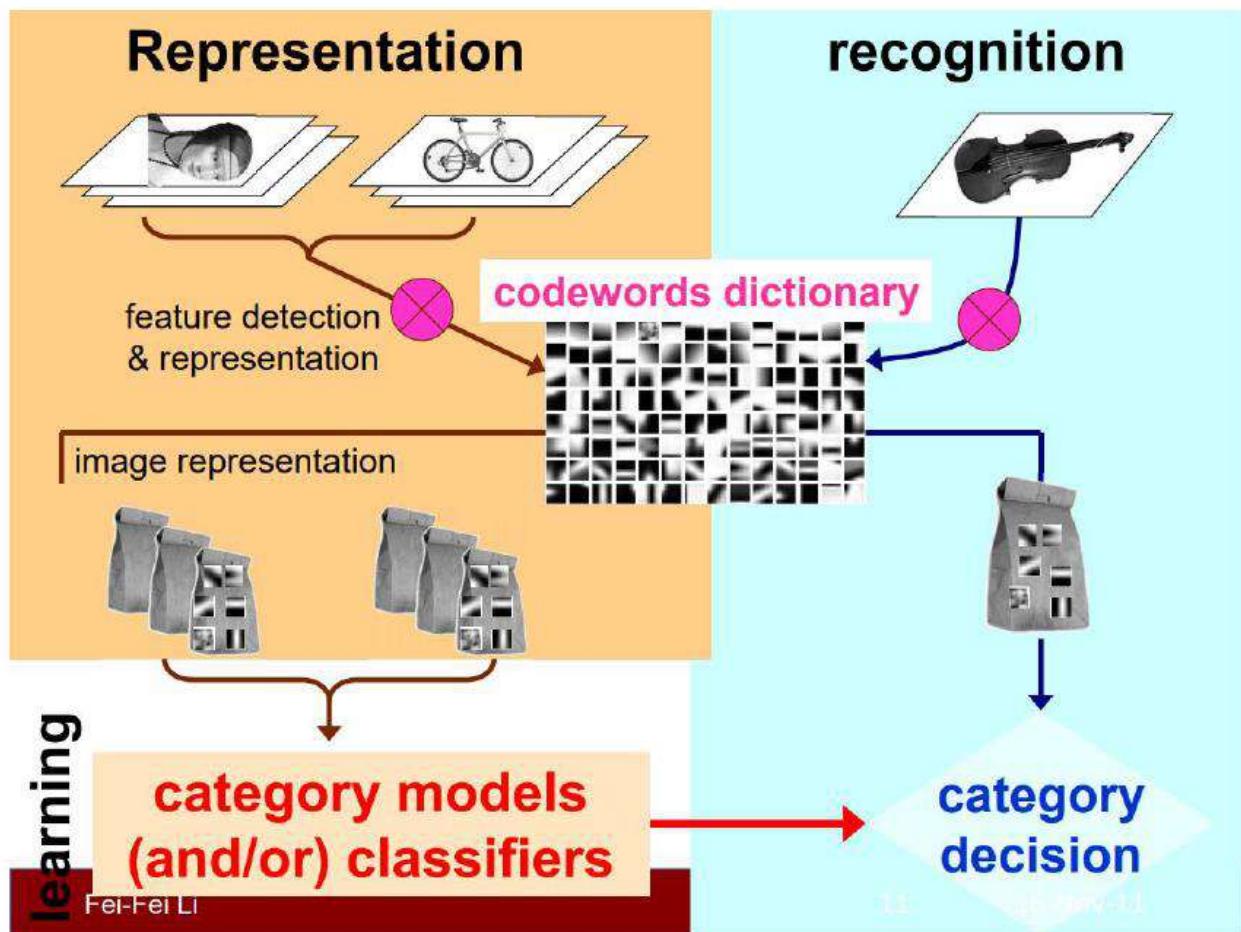
Embeddings bag of word: concetto di bag of word, utilizzo di bag of word nell'algoritmo sift, utilizzo di bag of word nelle CNN.

Il modello della borsa di parole (in inglese: *Bag-of-words model*, in sigla: BoW) è un metodo utilizzato nell'Information Retrieval e nel Elaborazione del linguaggio naturale per rappresentare documenti ignorando l'ordine delle parole. In questo modello, ogni documento è considerato in quanto contiene parole, analogamente a una borsa; ciò consente una gestione di queste basata su liste, dove ogni borsa contiene determinate parole di una lista. Nella Computer Vision si applica alla classificazione delle immagini, trattando l'immagine come caratteristiche (*feature*) di parole. In particolare, nell'object recognition, un'immagine può essere trattata come un documento e le caratteristiche rilevate in determinati punti dell'immagine si considerano "parole" visuali. Nella classificazione di documenti, la borsa di parole è un vettore sparso del numero di occorrenze delle parole, che non è altro che un istogramma sparso sul vocabolario. In Computer Vision una borsa di parole visuale è un vettore sparso di occorrenze del vocabolario di caratteristiche locali dell'immagine.

Bag of word: https://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision

Nell'immagine non c'è il concetto di word, quale visual features uso come word? Ne posso scegliere tante.

Bag of visual pattern, non importa il loro significato (occhio, capelli, ecc.) li metto tutti in un vocabolario. Creo per ogni immagine un istogramma in cui ogni bin corrisponde ad un visual feature del dizionario e il valore corrispondente al numero di volte che quella visual feature appare nell'immagine. Code word dictionary → dizionario dei visual pattern.



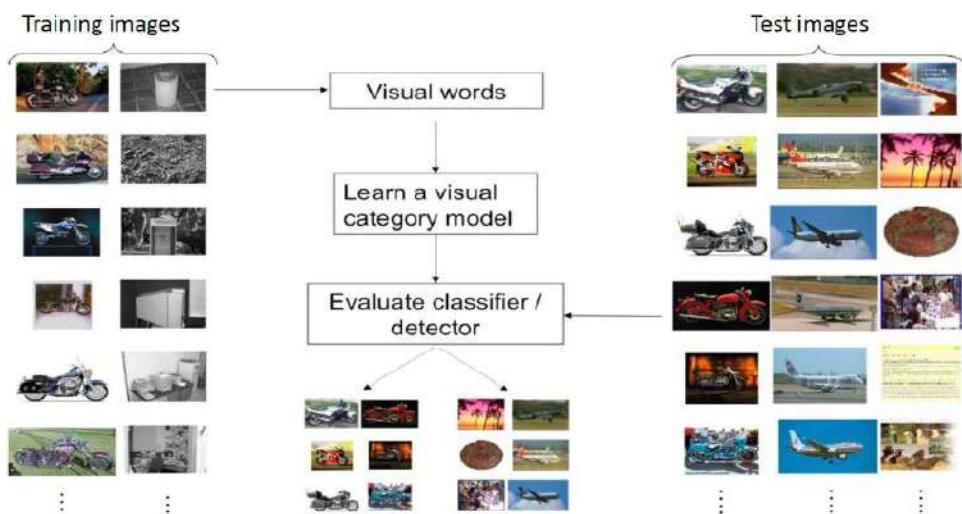
Le visual word possono essere definite a priori o possono essere imparate dal dataset e spesso sono imparate considerando features locali.

Bag of word è molto usato nella categorizzazione di oggetti nelle immagini e in tempi recenti ha iniziato ad essere usato anche per action recognition nelle sequenze video. Vdi slides 171-172.

Abbiamo un set di immagini di training ed utilizziamo i descrittori locali di queste immagini per estrarre le **visual words** e creare il codebook. Dopo di che si costruisce il **visual category model** allenando un classificatore. Tale classificatore viene allenato utilizzando le visual word per ogni classe corrispondente ad un oggetto (offline process). Dopo di che si fa la recognition, ovvero il classificatore allenato classifica l'immagine di input in accordo al visual category model imparato.

Il retrieval è l'operazione di, data una immagine o una parte di essa come query, restituire qualcosa di simile basandosi sulle visual word utilizzate.

Quindi la scelta delle visual words è molto importante. Vedi i tipi di visual words slide 174.



Gli steps sono i seguenti:

- 1)scelta delle possibili visual words (tutte le possibili scelte slides 174)
 - 2)clusterizzare tutte le possibili visual words trovate per trovare le reali visual words

Da slide 176: descrittore BoW+SIFT: prendo dei punti di interesse (patches) dell'immagine, li normalizzo e poi uso sift. Ogni patch normalizzato elaborata da sift è una visual word.

I metodi di rappresentazione delle caratteristiche trattano di come rappresentare le patch come vettori numerici. Questi vettori sono chiamati descrittori di caratteristiche. Un buon descrittore dovrebbe avere la capacità di gestire in una certa misura intensità, rotazione, scala e variazioni affini. Uno dei descrittori più famosi è SIFT (Scale-invariant feature transform). SIFT converte ogni patch in un vettore a 128 dimensioni. Dopo questo passaggio, ogni immagine è una raccolta di vettori della stessa dimensione (128 per SIFT), in cui l'ordine dei diversi vettori non ha importanza.

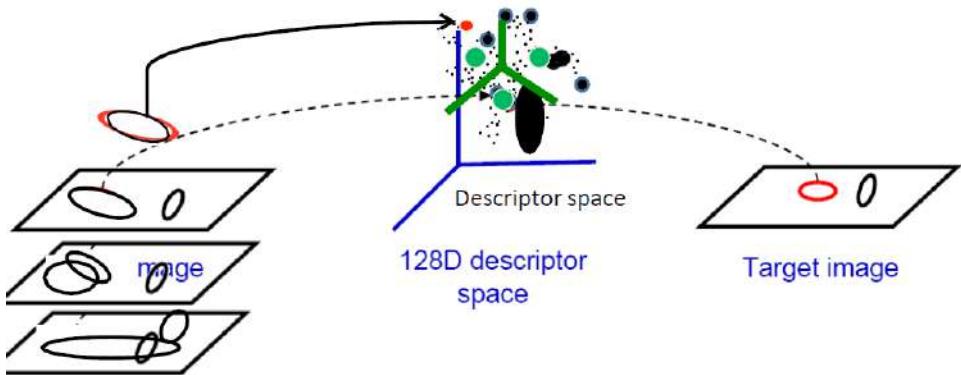
Dopo aver calcolato tutte le visual word dell'immagine le metto insieme e l'immagine in questo modo viene descritta con esse (può essere vista come un documento). Dopo di che tutte le visual word che rappresentano tutte le nostre immagini vengono messe nello spazio e vengono raggruppate in cluster per creare il dizionario. Il numero di word presenti nel vocabolario sarà pari al numero di cluster che ottengo in questa fase. Se uso k-means avrò k cluster quindi otterrò un dizionario di k visual word. Il centroide di ogni cluster diventa la word nel vocabolario.

Local features estratte dall'immagine viene rappresentata grazie a qualche algoritmo (sift per esempio) come keypoint.

Slide 179: esempio di bag of word + sift

Slide 181: ogni immagine viene descritta da un insieme di vettori di 128 dimensioni quindi l'insieme dei descrittori di tutte le immagini del dataset è un insieme di vettori molto grande (in più lo spazio ha 128 dimensioni) quindi deve essere ridotto. Per fare ciò si fa un clustering in cui si mantiene soltanto un elemento per ogni cluster, il suo centro. Questo è chiamato quantizzazione del feature space. Rimangono come descrittori di visual word solo i centroidi dei cluster e questi vanno a formare il vocabolario. (vettori simili sono raggruppati all'interno della stessa visual word).

Dopo aver creato tale dizionario ogni immagine è rappresentata in forma vettoriale sostituendo i suoi descrittori precedenti con le più vicine (ovviamente si deve definire una giusta metrica di similarità) word del vocabolario.



Facendo questo punti (vettori) di immagini differenti sono più facilmente confrontabili.

Ricordo che nelle visual word non c'è semantica.

Arrivati a questo punto abbiamo costruito il codeword di vettori (o punti di dim 128) e di conseguenza anche di visual word.

Consideriamo una query image. Per prima cosa è necessario estrarre i suoi descrittori (patch + sift ad esempio) e poi andare a trovare a quali visual word corrispondono. Una volta che abbiamo trovato con quali visual word del vocabolario questa immagine può essere descritta andiamo a costruirne l'embedding, cioè con un istogramma di visual features contenute nel codeword che appaiono nell'immagine con relativa frequenza. Così facendo, tuttavia, perdiamo le informazioni spaziali perché sappiamo quante volte una visual word appare ma non dove appare nell'immagine.

L'ultima cosa da fare è quello di passare questo embedding in un classificatore in modo che venga trovato l'oggetto rappresentato nell'immagine, oppure si possono usare metodi di learning non supervisionato per trovare immagini simili. (si può fare classificazione, ricerca o riconoscimento sulla base degli embeddings).

Ora andiamo a vedere i metodi per estrarre caratteristiche più dettagliate dalle immagini.

Keypoints e features locali: slides 190-315

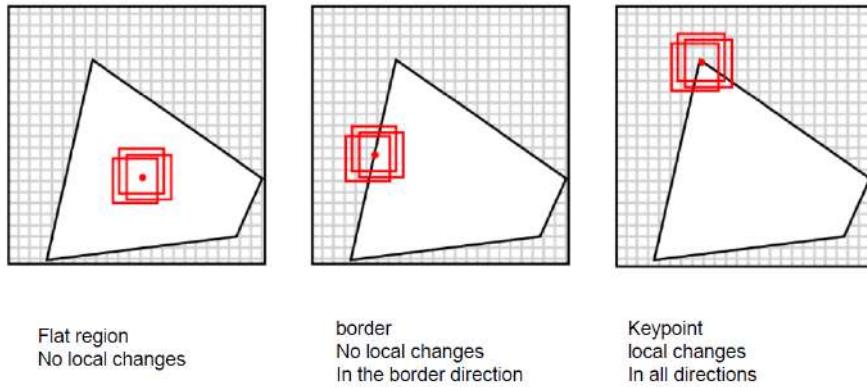
Keypoints detection è diverso da keypoints description.

Harris detector: https://it.wikipedia.org/wiki/Rilevatore_di_Harris

Harris detector è in grado di detectare punti salienti o keypoints. Tali punti sono caratterizzati da una definizione formale (percettiva, algebrica..), da una precisa posizione (localizzazione) e da eventuale invarianza a qualche trasformazione (ad esempio rotazione, luminosità ecc.). Harris detector usa il concetto di **autocorrelazione** per detectare questi punti.

Cos'è l'autocorrelazione? <https://it.wikipedia.org/wiki/Autocorrelazione>

Per capire l'autocorrelazione in modo semplice pensiamo di selezionare una potenziale finestra saliente dell'immagine ed utilizzarla come template per fare il template matching fra l'immagine stessa e tale finestra. I risultati di tale matching indicano il livello di autocorrelazione e in base ad essi si capisce se l'area potenzialmente saliente lo è o meno. Intuitivamente se la finestra matcha con tante altre regioni dell'immagine non sarà saliente e viceversa. In particolare possiamo distinguere tre macrosituazioni:



Direzioni dei grad: 0

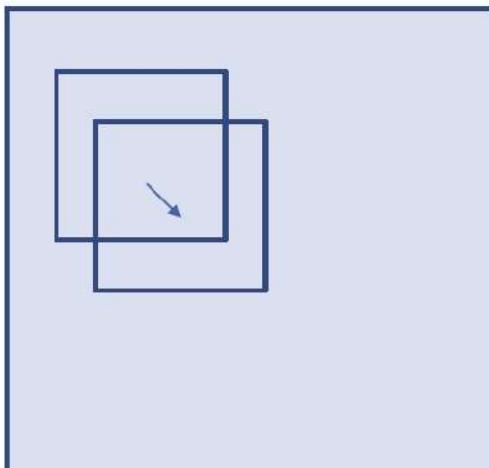
1

2

Infatti le porzioni che hanno il gradiente in almeno due direzioni sono quelle più significative e devono essere localizzate.

Si ha un picco del gradiente in corrispondenza dell'angolo

Formalizzando il concetto esposto a grandi linee sopra, dato un pixel $p(x,y)$ e una sua finestra di intorno $\Omega(p)$ facciamo il confronto (come nel template matching) di tale finestra con la finestra che si ottiene traslando la finestra stessa di una lunghezza δ in direzione d (dove d è un vettore unitario).



$$D_q(\mathbf{d}) = \sum_{\mathbf{r} \in \Omega(q)} [\mathbf{d}^\top \nabla I(\mathbf{r})]^2$$

D is the information content associated with p in the d direction, relative to the window
D can be expressed as a (u,v) vector

(sono abbastanza convinta che nella formula al posto di q ci vada p). D è il contenuto informativo associato a p traslato nella direzione d . Il vettore d , essendo un vettore unitario, può essere espresso come un vettore (u,v) .

L'informazione che stiamo cercando, cioè quanto la finestra iniziale sia saliente, è chiamato contenuto informativo della porzione di immagine che stiamo considerando ed è calcolato come la differenza di intensità $E(u,v)$ tra la finestra originale e la finestra traslata moltiplicata per una funzione w :

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

Window function $W(x, y) =$

1 in window, 0 outside or Gaussian

Interpretando nel modo corretto questo valore possiamo capire se l'area considerata inizialmente è saliente o meno.

Approssimiamo l'espressione di $E(u,v)$ ottenuta sopra utilizzando le serie di taylor:

$$\begin{aligned}
 E(u, v) &\approx \sum_{x,y} w(x, y) [I(x, y) + uI_x + vI_y - I(x, y)]^2 \\
 &= \sum_{x,y} w(x, y) [uI_x + vI_y]^2 \\
 &= \sum_{x,y} w(x, y) (u - v) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
 \end{aligned}$$

Quindi, considerando valida l'espansione di taylor (per piccole traslazioni $[u,v]$ è valida), possiamo usare la forma matriciale dell'approssimazione bilineare, essendo M la matrice derivata dell'immagine 2×2

$$E(u, v) \cong [u, v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

M è chiamata structure tensor (tensore di struttura) o anche matrice di autocorrelazione o anche matrice hessiana.

(<https://www.youmath.it/domande-a-risposte/view/6260-matrice-hessiana.html#:~:text=L'Hessiana%20di%20una%20funzione,parziali%20seconde%20della%20funzione%20f.&text=ovvero%C3%A8%20un%20modo%20compatto,di%20f%20rispetto%20a%20xj.>)

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

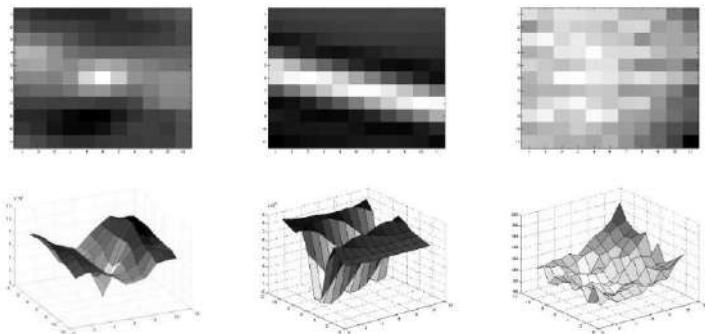
The auto-correlation matrix \mathbf{A} can be written as

$$\mathbf{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix},$$

The different Hessian matrixes
and the autocorrelation surfaces



(a)



Il punto p nell'immagine è saliente (quindi per il ragionamento visto nell'immagine prima è un angolo) se il contenuto informativo è sufficientemente significativo. Questo equivale a dire che il contenuto informativo è maggiore di una soglia scelta, ovvero:

$$\min\{d^T M d \mid d \in \mathbb{R}^2, \|d\| = 1\} > \tau$$

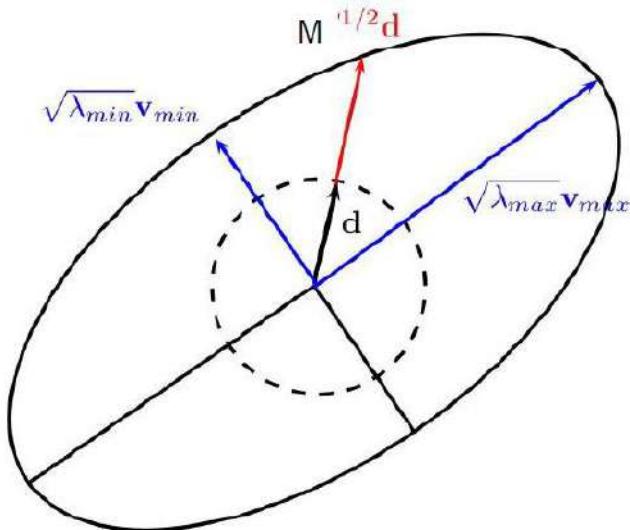
Questo equivale a dire che, per essere rilevante il punto considerato, il minimo autovalore di M deve essere minore della soglia:

$$\text{Min } \lambda > \tau$$

I due autovalori di M sono entrambi positivi essendo $d^T M d > 0$. (questo perché d trasposto per M per d equivale agli autovalori di $M \rightarrow$ perché? boh). Vedi meglio autovalori e autovettori e cerca di capire : <https://www.youmath.it/lezioni/algebra-lineare/matrici-e-vettori/735-autovalori-e-autovettori.html>

L'interpretazione geometrica di M è la seguente: (vedi slide 201)

$$M = V \Lambda V^T = (\mathbf{v}_{max} \mathbf{v}_{min}) \begin{pmatrix} \lambda_{max} & 0 \\ 0 & \lambda_{min} \end{pmatrix} \begin{pmatrix} \mathbf{v}_{max}^T \\ \mathbf{v}_{min}^T \end{pmatrix}$$



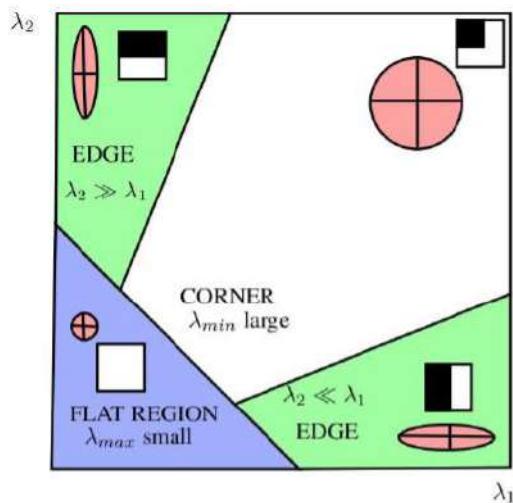
$\mathbf{v}_{min}, \mathbf{v}_{max}$ are the eigenvectors
The points of the ellipsis are
Characterized by
 $x^T M^{-1} x = 1$

The vectors of the principal axes are
Indicated.
thus they should be large enough

Se gli autovalori sono sufficientemente alti significa che ho due dimensioni possibili (due assi) e che il vettore non sarà per forza in una direzione ma potenzialmente sarà in due il che equivale a dire che il gradiente dell'immagine nella porzione ha due direzioni quindi che l'immagine è un angolo.

Infatti angoli e bordi possono essere distinti analizzando gli autovalori della matrice Hessiana calcolata sulla finestra considerata. Gli autovalori λ_1 e λ_2 di M svelano l'intensità della variazione del gradiente lungo le due direzioni ortogonali più significative.

- If $\lambda_1 \gg \lambda_2$ or $\lambda_1 \ll \lambda_2$ there is an edge
- If $\lambda_1 \sim \lambda_2$ and they are large, we have a corner
- If they are small, we are in a flat region



Riassumendo, si considera una porzione, si calcola la matrice di autocorrelazione che è la somma delle matrici Hessiane in tutti i pixels della porzione:

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Poi si calcolano i due autovalori di questa matrice. Dopo di che si fanno le considerazioni viste sopra su tali autovalori per capire di che tipo è la porzione considerata.

Nella realtà non vengono sottoposti a soglia gli autovalori ma dei valori calcolati sulla base di essi. Ci sono diverse metriche che si possono calcolare applicando diversi metodi, noi e vedremo due:

- 1) Il valore f viene sottoposto a soglia:

$$f = \frac{\text{Det}(M)}{\text{trace}(M)} f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} \quad \text{dove}$$

In general given a C matrix as the Hessian one

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$$\begin{aligned} \det(C) &= c_{11}c_{22} - c_{12}c_{21} = \lambda_{\min}\lambda_{\max} \\ \text{trace}(C) &= c_{11} + c_{22} = \lambda_{\min} + \lambda_{\max} \end{aligned}$$

- 2) Corner response: il valore R viene sottoposto a soglia:

Measure of corner response:

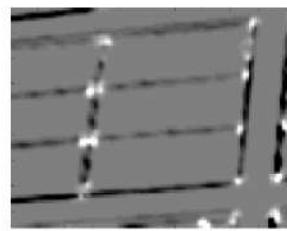
$$R = \det M - k (\text{trace } M)^2$$

$$\begin{aligned} \det M &= \lambda_1 \lambda_2 \\ \text{trace } M &= \lambda_1 + \lambda_2 \end{aligned}$$

$$(k - \text{empirical constant}, k = 0.04-0.06)$$

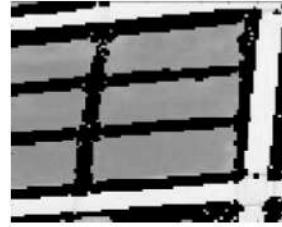
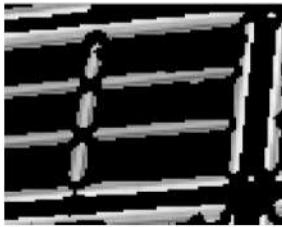
Dopo aver trovato quali sono le porzioni dell' immagine in cui ci sono potenziali angoli si applica un metodo per rilevare il massimo locale (es. non maxima suppression, oppure posso tenere solo i keypoints che sono un 10% maggiori rispetto ai punti nell'intorno) per tenere un solo punto per porzione: quello è un keypoint.

Esempi:



R computed in each point

Derivatives are computed with sobel, w is a gaussian with sigma=1



R<1000, borders

[R]<10000 flat regions

R>10.000 keypoints

Riassunto dell'algoritmo di Harris detector

Summary of harris detector algorithm

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x \cdot I_x \quad I_{y2} = I_y \cdot I_y \quad I_{xy} = I_x \cdot I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma^2} * I_{x2} \quad S_{y2} = G_{\sigma^2} * I_{y2} \quad S_{xy} = G_{\sigma^2} * I_{xy}$$

4. Define at each pixel (x, y) the matrix

$$H(x, y) = \begin{bmatrix} S_{x2}(x, y) & S_{xy}(x, y) \\ S_{xy}(x, y) & S_{y2}(x, y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

$$R = \text{Det}(H) - k(\text{Trace}(H))^2$$

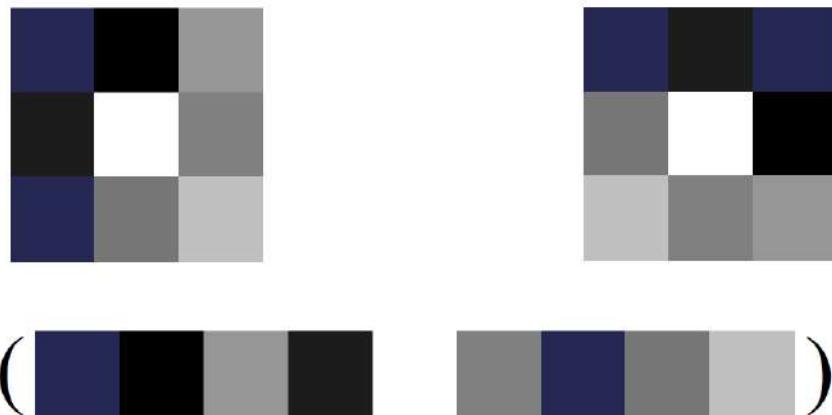
6. Threshold on value of R. Compute nonmax suppression.

Esempio slides: 207-211

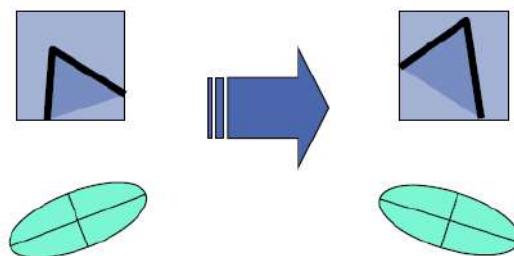
+ slide 213

I punti rilevati da harris possono essere usati come features nel Bag of word.

Come possono essere descritti questi punti rilevati? Che descrittore si usa? Il **descrittore** usato da Harris è un vettore di 9 componenti corrispondenti ai valori dei pixels nell'intorno del keypoint. Questo descrittore tuttavia, non è **invariante alla variazione di luminosità e non è invariante alla rotazione**. Perché se io ruoto una stessa immagine rilevo gli stessi keypoints ma i descrittori di quei keypoints sono diversi quindi lo stesso angolo risulterà come un angolo diverso nelle due immagini.



Il rilevamento di keypoints con algoritmo di Harris è invariante alla rotazione perché in qualunque modo venga ruotata l'immagine l'angolo mantiene le sue proprietà sfruttate da harris per rilevarlo (doppia direzione del gradiente nella finestra). (corner response è invariante alla rotazione dell'immagine).



Nonostante la detection dei keypoints fatta con harris sia buona (invariante alla rotazione, invariante alla variazione di luminosità), la rappresentazione di essi non lo è (non è invariante a nessuna dei due).

Inoltre sia il rilevamento di keypoints con Harris che il loro descrittore non sono invarianti allo scaling. Infatti un punto ad una certa risoluzione può essere visto come un angolo mentre ad un'altra solo come un bordo.

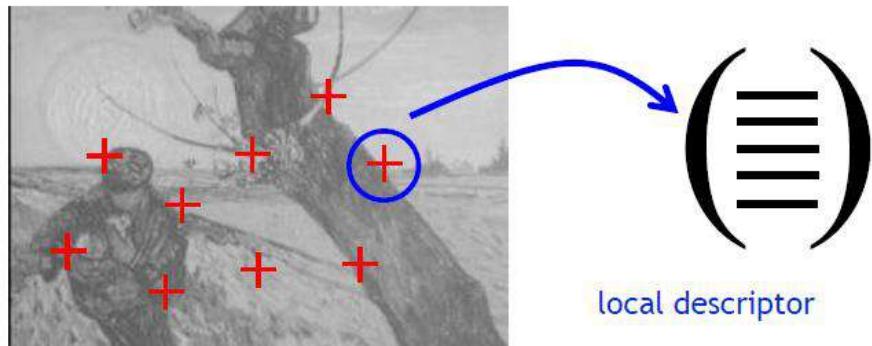
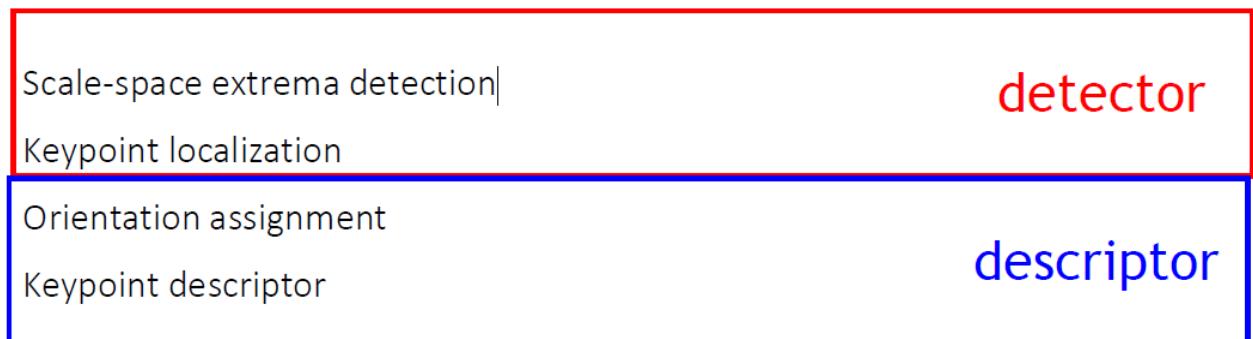
L'algoritmo che risolve i problemi di invarianza di Harris sia nel rilevamento che nel descrittore è **SIFT**.

Spiegazione buona: <https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40>

SIFT è un algoritmo in grado di rappresentare entità visive secondo le loro proprietà con parametri empiricamente determinati. Questo metodo utilizza come descrittore per l'immagine features locali individuate in corrispondenza dei keypoints individuati. La selezione dei keypoints e i loro descrittori sono robusti alle variazioni di luminosità, allo scaling e alla rotazione.

Gli steps di tale algoritmo sono i seguenti:

- Rilevamento degli estremi dello spazio scalato
- Localizzazione dei keypoints
- Assegnamento dell'orientazione (fare una sorta di standardizzazione dell'orientazione in modo da avere la stessa orientazione per ogni possibile rotazione dell'immagine)
- Descrittori dei keypoints



A 500x500 image gives about 2000 features

Il descrittore di SIFT è un vettore di 128 componenti ed è usato solo per descrivere i keypoints, che sono appunto punti dell'immagine che hanno particolari feature interessanti.

Come già detto tale algoritmo dovrà eseguire due operazioni, la detection di keypoints e la costruzione dei loro descrittori: SIFT keypoints detector e SIFT descriptor.

Algoritmo

The algorithm

1. Scale-space extrema detection
 1. Build a **Gaussian-blurred image pyramid**
 2. Subtract adjacent levels to obtain a **Difference of Gaussians (DoG) pyramid (so approximating the Laplacian of Gaussians)**
 3. Take **local extrema of the DoG filters at different scales as keypoints**
2. Keypoint localization
3. **Orientation assignment**
4. **Keypoint descriptor**

Andiamo ad analizzare ogni punto nel dettaglio:

1. Scale-space extrema detection

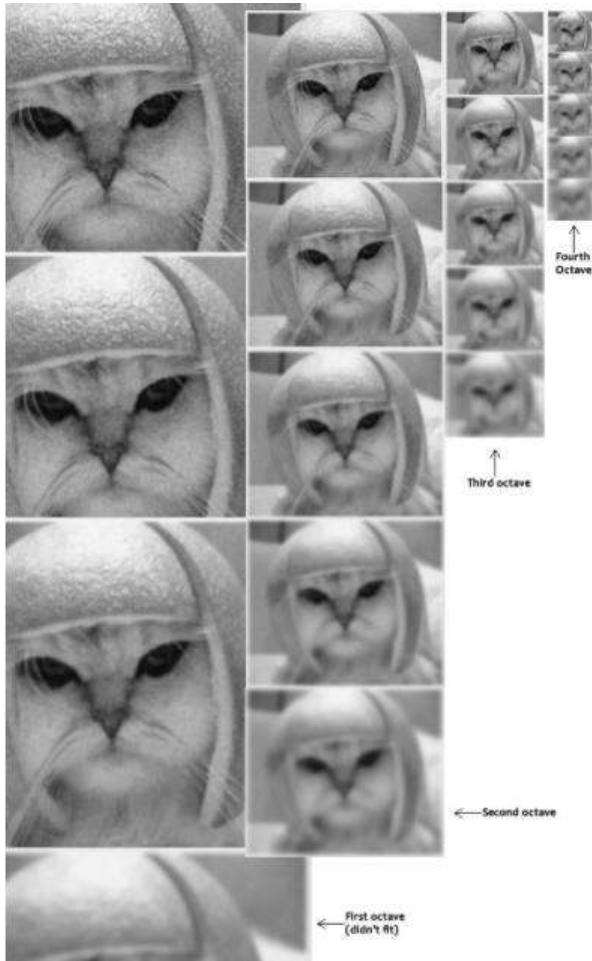
Fare detection dei punti estremi significa fare detection di quei punti che hanno particolare significato nell'immagine come ad esempio gli angoli. In questo algoritmo però si cercano i punti che sono estremi nell'immagine ma che rimangono estremi anche nella stessa immagine scalata in modo da ottenere l'invarianza rispetto lo scaling. Andiamo più nel dettaglio.

Per prima cosa dobbiamo definire lo space scale di un'immagine che è una funzione $L(x,y,\sigma)$ ottenuta facendo la convoluzione fra kernel gaussiani (blurring) con σ differenti e l'immagine iniziale a scale differenti. Lo scale-Space viene organizzato in ottave e il numero di tali ottave e le loro diverse scale dipendono dalla dimensione dell'immagine originale. Quindi generiamo diverse ottave partendo dall'immagine originale ognuna delle quali contiene immagini aventi la stessa dimensione ma sfuocate con kernel gaussiani aventi σ diversa. La dimensione delle immagini cambia solo fra ottave diverse mentre rimane la stessa dentro la stessa ottava. La dimensione dell'immagine in un'ottava è pari alla metà della dimensione delle immagini nell'ottava precedente.

Mentre i diversi σ dei diversi kernel applicati dentro la stessa ottava variano nel seguente modo:

$$\sigma_n = k^n \sigma_0 \quad (k = 2)$$

(ottave piramidali)

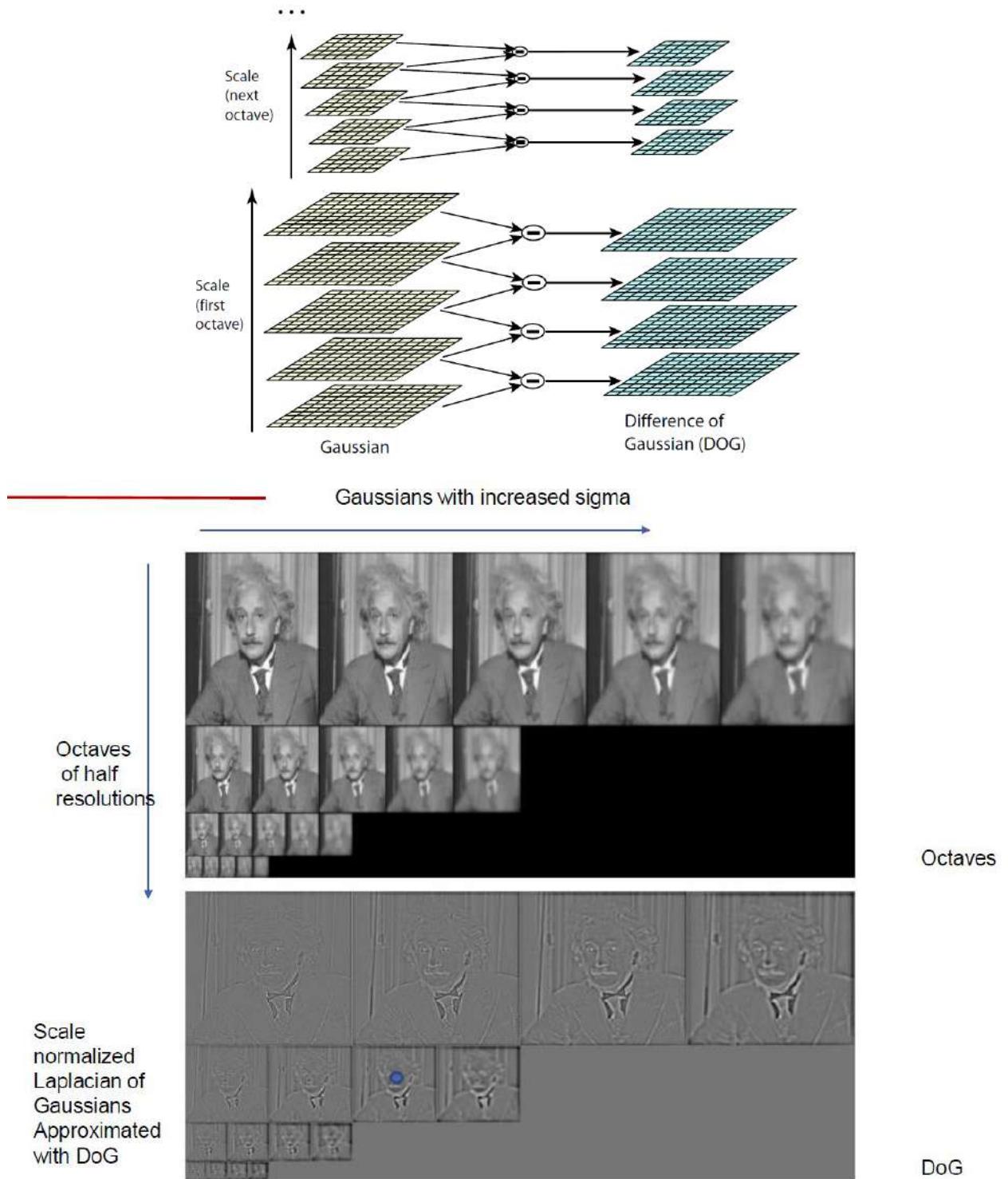


$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Arrivati a questo punto utilizziamo e ottave che abbiamo ottenuto per costruire un altro insieme di immagini il Derivative of Gaussian (DoG) definito da $D(x, y, \sigma)$. Quello che viene fatto all'interno di ogni ottava è quello di sottrarre tra loro due immagini sfuocate con filtri aventi σ differenti. In questo modo otterremo un nuovo insieme. Questa sottrazione ci consente di ottenere un'approssimazione del gradiente che ci consente di trovare i picchi della funzione dell'immagine.

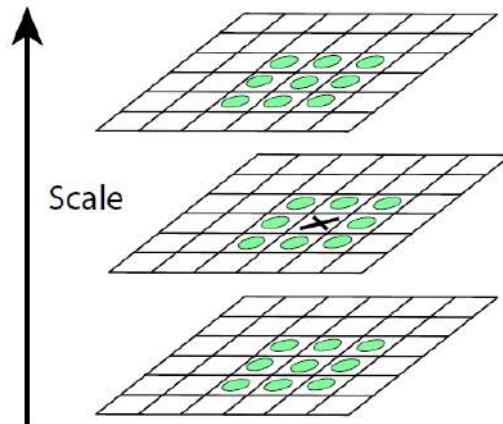
$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$



2. Keypoints Localization

Arrivati a questo punto abbiamo ottenuto un'approssimazione del gradiente invariante allo scaling. Le forti variazioni nell'immagine causano picchi sul gradiente quindi per rilevare i punti di interesse (local extrema) ci basta cercare i picchi in $D(x,y,\sigma)$.

Per trovare i punti di interesse ad ogni livello di scala della piramide di DoG si confronta ogni punto di ogni immagine con i suoi 8 pixels nell'intorno, con i 9 corrispondenti pixels dell'immagine con σ precedente e con i 9 pixel corrispondenti nell'immagine con σ successivo, per un totale di 26 confronti.



Se il punto p è un estremo locale (minimo o massimo locale) nei tre livelli allora viene selezionato come keypoint.

I keypoints generati in questo modo sono troppi e molti di essi sono posizionati lungo un edge oppure non hanno abbastanza contrasto per essere definiti keypoints. In entrambi i casi non sono sufficientemente interessanti per essere rilevati come feature quindi dobbiamo eliminarli. Il metodo utilizzato per eliminare quelli che non hanno sufficiente contrasto è semplice, basta confrontare l'intensità del contrasto con una soglia minima e trascurare il punto se tale contrasto non è sufficiente.

Mentre per eliminate i keypoints che si trovano sui bordi si applica lo stesso metodo usato da Harris. Viene utilizzato lo sviluppo in serie di Taylor per approssimare lo spazio in cui ci troviamo (DoG che poi corrisponde con il gradiente) e partendo da questo si applica tutto il ragionamento fatto da Harris e si tengono solo i punti per cui l'autovalore più piccolo supera una certa soglia per far in modo che i picchi siano in due direzioni (quindi su un angolo) e non in una (quindi su un bordo).

- Reject flats:

- $|D(\hat{x})| < 0.03$

- Reject edges:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Let α be the eigenvalue with larger magnitude and β the smaller.

$$\text{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\text{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

Let $r = \alpha/\beta$.
 So $\alpha = r\beta$

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

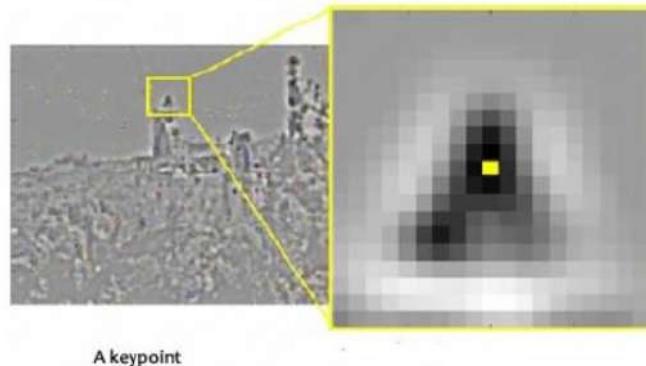
$(r+1)^2/r$ is at a min when the 2 eigenvalues are equal.

- $r < 10$

3. Orientation assignment

il descrittore del keypoint deve essere invariante all'orientazione (di conseguenza alla rotazione) quindi bisogna eseguirne una standardizzazione.

Ora conosciamo i keypoints e la scala (σ) a cui sono stati rilevati. Abbiamo già ottenuto l'invarianza allo scaling, dobbiamo ottenere quella alla rotazione.

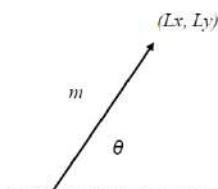
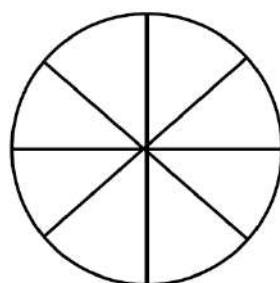


A keypoint

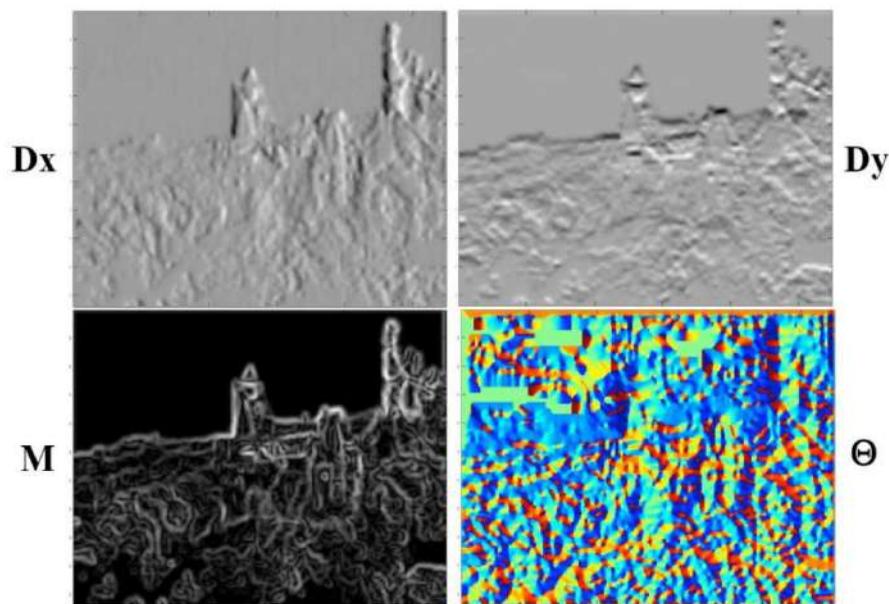
Consideriamo un intorno del keypoint in ogni immagine con σ (nelle immagini di L non di DoG) differenti e calcoliamo magnitudine e direzione del gradiente in questo intorno. Dopo di che si crea un istogramma con 36 bins che coprono tutte le 360 direzioni (intervallo di 10 direzioni per ogni bin) e si va ad aggiungere in ogni bin un valore proporzionale alla magnitudo del gradiente che ha direzione compresa nell'intervallo di direzioni associato a quel bin. Questa operazione si fa per ogni magnitudo e direzione del gradiente calcolata nell'intorno del keypoint.

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

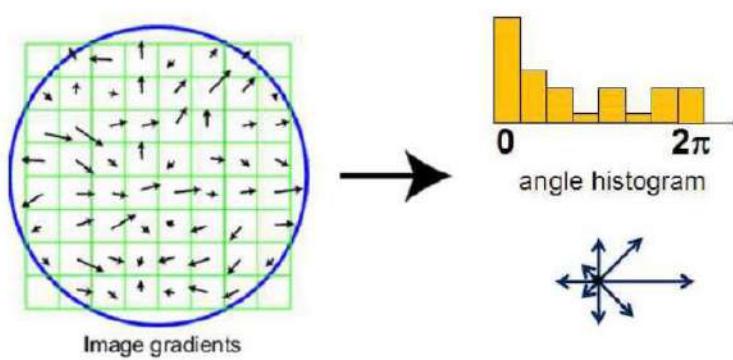
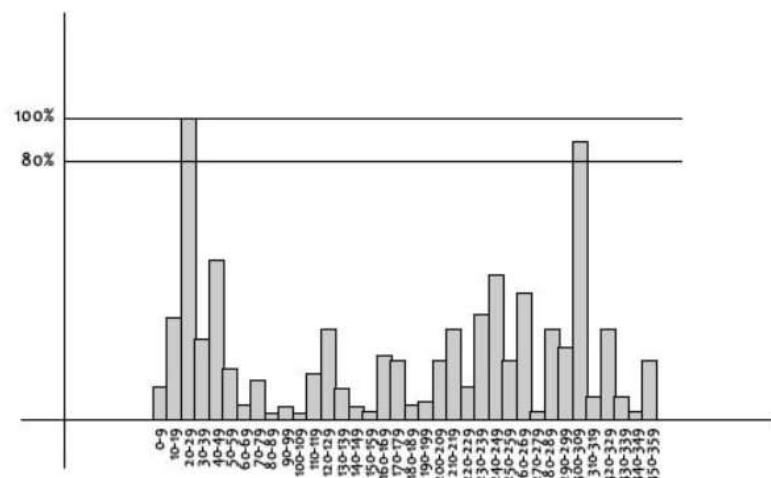
$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$



orientation histogram (36 bins evry 10 degrees)



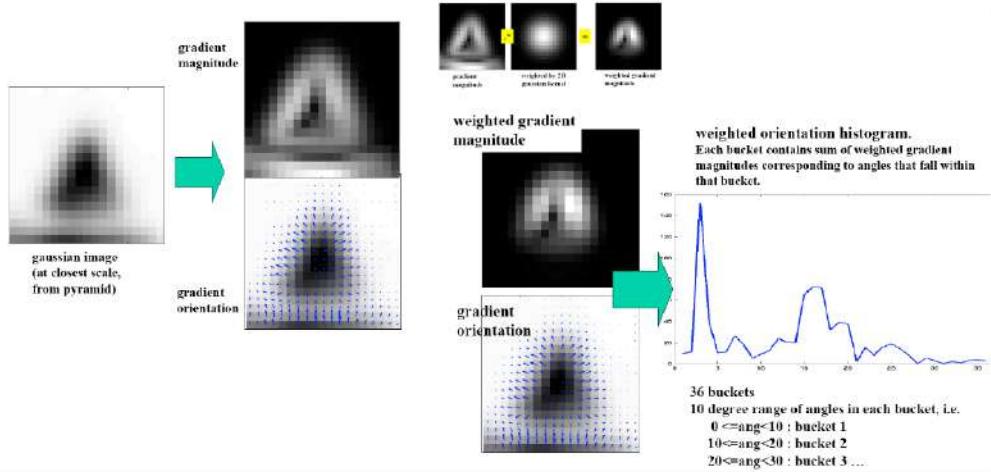
Una volta ripetuta questa operazione per ogni punto nell'intorno del keypoint l'istogramma avrà un picco da qualche parte.



- For such region

Create an histogram with 36 bins for orientation

Weight each point with Gaussian window of 1.5σ

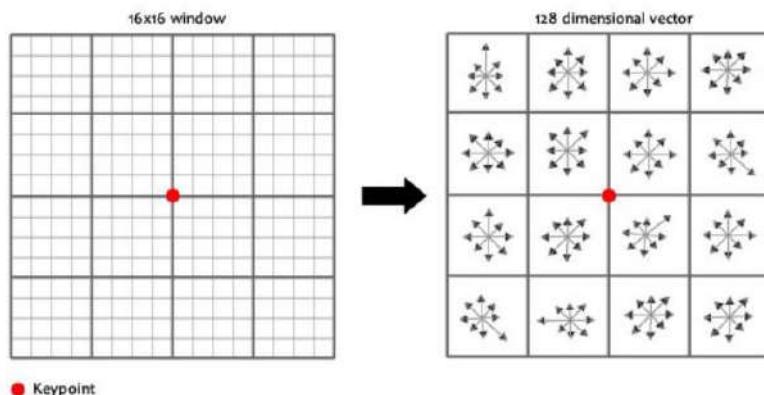


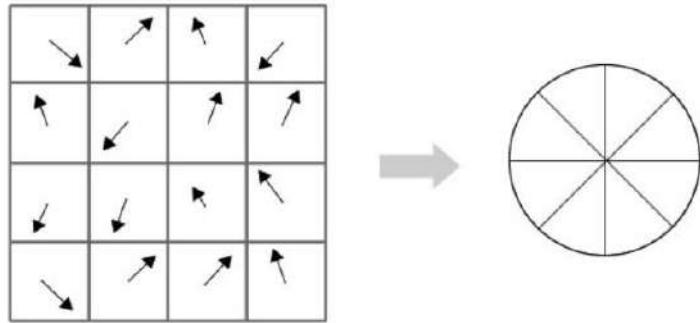
Viene considerata come orientazione la direzione corrispondente al picco più alto e anche ogni direzione che ha valore superiore all'80% di tale picco è tenuta in considerazione. In questo modo si crea un keypoint che ha stessa location e stesso scale ma diverse orientazioni. Questo rende in keypoint invariante all'orientazione.

4. Keypoint descriptor

Arrivati a questo punto ogni keypoint ha location, scale e orientazione. L'ultimo step da fare è quello di costruire il descrittore nel modo più discriminante e invariante (ad esempio a variazioni di luminosità o di punto di vista) possibile per la regione di immagine identificata dal keypoint.

Per costruire il descrittore consideriamo una finestra 16×16 attorno al keypoint e la dividiamo in 16 sottoblocchi di dimensione 4×4 . Per ogni sottoblocco costruiamo un istogramma di 8 bin rappresentante le orientazioni del gradiente in quel blocco:





Quindi in fin dei conti sono utilizzati 4×4 descriptor diversi, ognuno dei quali di 8 bin. Quindi considerando una finestra di 16×16 divisa in sottoblocchi 4×4 ognuno dei quali avente 8 bins il descrittore associato al keypoint avrà dimensione $4 \times 4 \times 8 = 128$ e corrisponderà a tutti gli istogrammi di 8 bins della direzione del gradiente calcolati nell'intorno messi in fila.

(questa ultima parte non c'è sulle slides ma per completezza la aggiungo)

This feature vector introduces a few complications. We need to get rid of them before finalizing the fingerprint.

1. **Rotation dependence** The feature vector uses gradient orientations. Clearly, if you rotate the image, everything changes. All gradient orientations also change. To achieve rotation independence, the keypoint's rotation is subtracted from each orientation. Thus each gradient orientation is relative to the keypoint's orientation.
2. **Illumination dependence** If we threshold numbers that are big, we can achieve illumination independence. So, any number (of the 128) greater than 0.2 is changed to 0.2. This resultant feature vector is normalized again. And now you have an illumination independent feature vector!

Biological motivation di sift: slide 238

Altri descrittori: slide 239-240

Applicazioni: slide 241-263

Algoritmo Bag of Word: Rileggi per completezza, dovrebbe essere uguale a quello già spiegato: slide 264-271

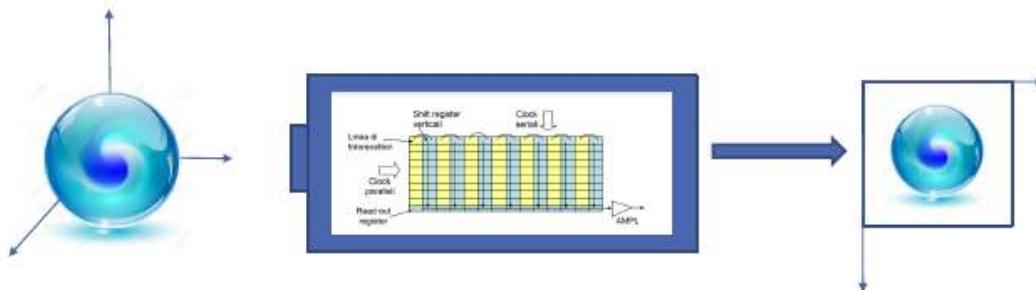
Bag of Word and Deep Learning: slide 273-277

After Bow: slide 278-320

8. The camera model

Le immagini con cui si lavora sono matrici di dati. L'immagine è una rappresentazione digitale dei segnali che vengono acquisiti da un sensore, la fotocamera.

Un sensore di immagini o un sensore ottico è un dispositivo che converte una immagine ottica in un segnale elettrico.



Esistono molti tipi di *sensori di immagini*, la prima suddivisione da fare è tra quelli che riprendono a colori e quelli che riprendono in bianco e nero - anche se ormai questi ultimi sono presenti solo in campi ristretti di utilizzo (visione notturna) - poi altre suddivisioni sono fatte in base a caratteristiche come metodo di rilevamento dei colori, tecnologia, sensibilità alla luce, risoluzione ecc.

Il principio base di funzionamento è il seguente: l'immagine viene focalizzata su una griglia composta da una miriade di piccoli sensori puntiformi i quali convertono singolarmente la luminosità rilevata. Le tecnologie più conosciute sono sicuramente la CCD e la CMOS, erroneamente si fa coincidere questi acronimi coi relativi sensori. (Però nelle slides della Rita i nomi si riferiscono ai sensori e amen).

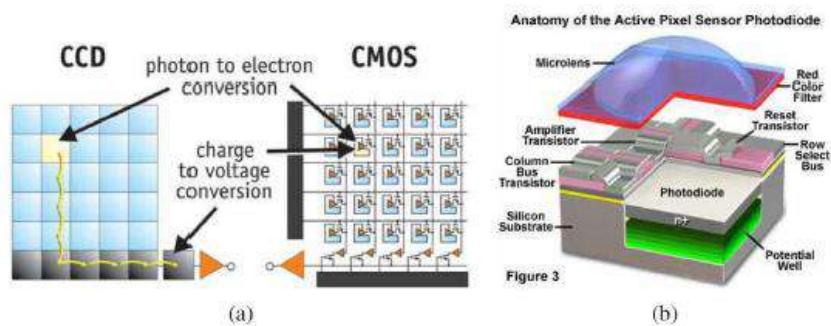


Figure 2.24 Digital imaging sensors: (a) CCDs move photogenerated charge from pixel to pixel and convert it to voltage at the output node; CMOS imagers convert charge to voltage inside each pixel (Litwiller 2005) © Photonics Spectra 2005; (b) cutaway diagram of a CMOS pixel sensor, from <http://olympus.magnet.fsu.edu/primer/digitalimaging/cmosimagesensors.html>.

Le tecnologie CDD e CMOS, a prescindere dalla fenomeno fisico con cui lavorano, garantiscono in output i valori dell'immagine discretizzati nello spazio dei pixels.

(approfondimento su questo argomento Capitolo 3 del libro di Szelinski's Book)

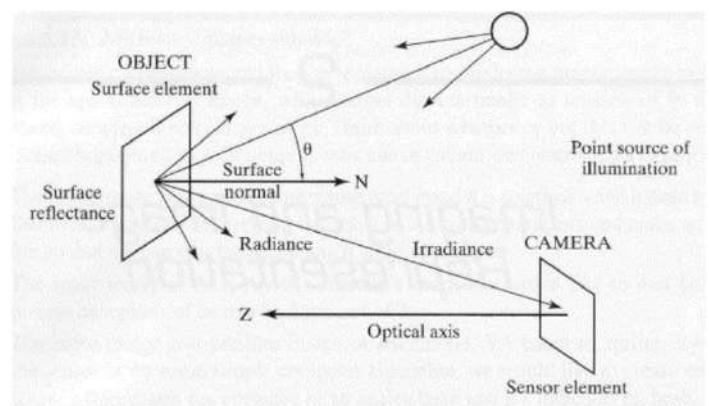
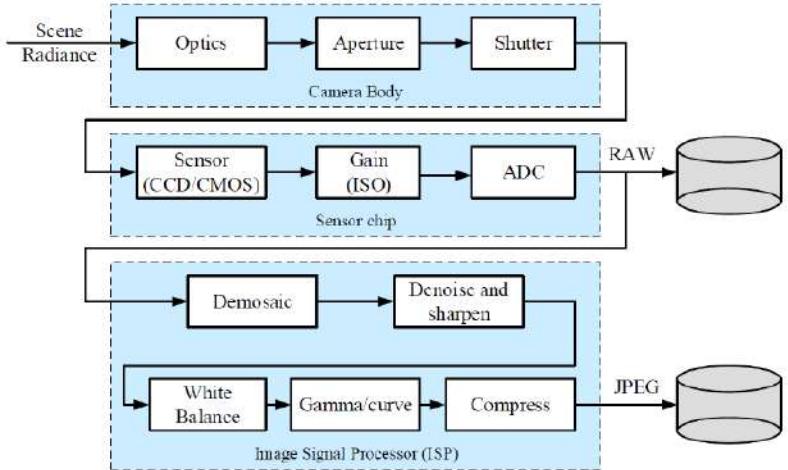
Pipeline di acquisizione

La pipeline che porta alla memorizzazione di una scena reale in un file immagine è illustrata in figura. La pipeline può essere suddivisa in tre fasi principali: acquisizione, elaborazione e memorizzazione. Nella fase di acquisizione la luce proveniente dalla scena attraversa il sistema di lenti che la indirizza verso il sensore della fotocamera (costruito con tecnologia CCD o CMOS). Quest'ultimo è composto da un alto numero di elementi fotosensibili che catturano l'energia della luce convertendola in corrente elettrica e che riescono in tal modo a determinare il valore di luminosità di ciascun pixel. Per acquisire immagini a colori è necessario scomporre la luce visibile nelle tre componenti fondamentali, corrispondenti alla lunghezza d'onda del rosso, del verde e del blu. In linea di principio bisognerebbe avere un sensore per ognuno dei colori da catturare, facendo lievitare oltremodo il prezzo delle fotocamere e introducendo svariate complicazioni tecniche. I dispositivi comunemente in commercio adottano invece un'altra soluzione: sopra il sensore viene applicata una sottile pellicola fotosensibile, detta CFA (Color Filter Array), che ha il compito di filtrare la luce e di separarla nei tre colori fondamentali in modo che ogni singolo pixel sia specializzato nella cattura selettiva di ciascuno di questi colori. In questo modo si ottiene una griglia di valori in cui ogni pixel registra il segnale relativo ad una sola componente cromatica. L'impiego dei CFA richiede la ricostruzione delle due componenti mancanti mediante un algoritmo di interpolazione (demosicing). Di solito questa elaborazione viene eseguita dal processore della fotocamera prima della memorizzazione della foto, unitamente ad altre operazioni di post-processing quali: il bilanciamento del bianco, l'elaborazione del colore, la correzione di pixel difettosi, la soppressione delle "dark currents", il miglioramento del contrasto e la correzione gamma. L'immagine ottenuta viene quindi compressa. L'algoritmo di compressione utilizzato dalla maggior parte delle fotocamere in commercio è il JPEG.

Come sappiamo però in computer vision è necessario decomprimere le immagini e lavorare con il formato RAW di queste.

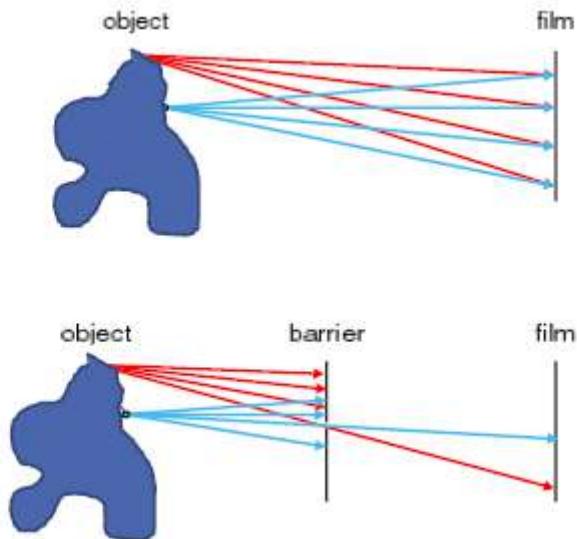
Il processo di formazione dell'immagine all'interno della fotocamera e quindi la successiva elaborazione di quest'ultima è determinato da due aspetti:

1. La geometria alla base della formazione dell'immagine
2. La fisicità della luce all'interno della scena



Per semplificare il modello di geometria dell'immagine si assume che la scena sia illuminata da una singola sorgente.

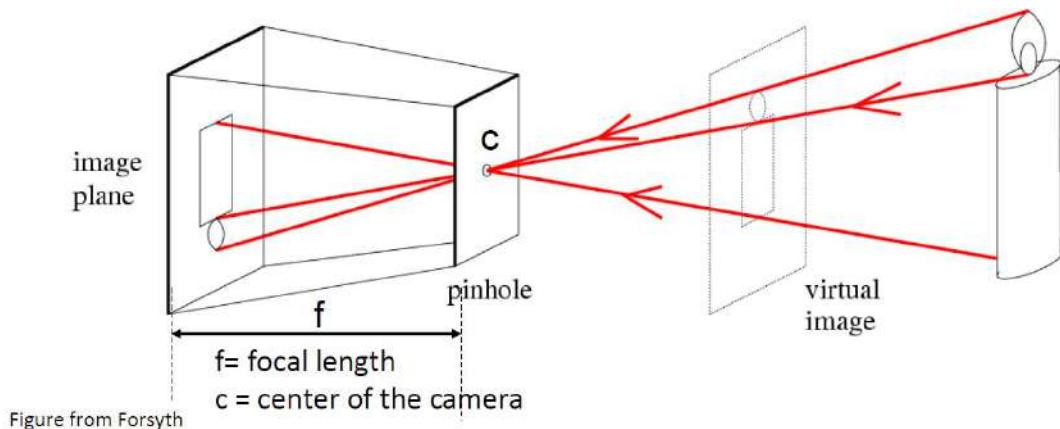
Infatti nella realtà ogni parte di un oggetto che viene illuminato a sua volta riflette luce. Per bloccare i raggi che irradiano da ogni punto dello oggetto viene aggiunta una *barrier* che riduce le sfocature e permette di attenuare l'irradiazione che parte da ogni punto della scena che vogliamo fotografare.



La tecnica che viene più utilizzata per formare un'immagine da una scena 3D, quindi per convertire da 3D a 2D tramite l'aggiunta di un barrier è il **pinhole camera model**.

Il pinhole camera model sfrutta il principio della camera oscura. La camera oscura è un dispositivo ottico composto da una scatola oscurata con un foro (pinhole) sul fronte e un piano di proiezione dell'immagine sul resto. La camera oscura tramite il foro raccoglie i raggi proiettati su qualsiasi oggetto e ottiene dall'altra parte della camera l'immagine capovolta.

Il funzionamento è intuitivo dalla foto:



La storia del Pinhole model: (leggere per completezza)

Il foro stenopeico è l'antenato della macchina fotografica. Fu scoperto nel V secolo a.c. da uno studioso cinese che osservò che la luce viaggiava in linea retta. Il filosofo **Mo Ti** osservò la formazione dell'immagine rovesciata, come succede nelle macchine fotografiche contemporanee, attraverso il foro stenopeico.

Un secolo dopo, nel IV A.C., **Aristotele** in "Problemata" affermanva che "I raggi del sole che passano per un'apertura quadrata formano un' immagine circolare la cui grandezza aumenta con l'aumentare della distanza dal foro "(1). La leggenda narra che Aristotele fosse nell'intimità della "ritirata" e notò come l'immagine del paesaggio esterno alla stanza buia dove si trovava proiettava un'immagine ribaltata sul suo vestito.

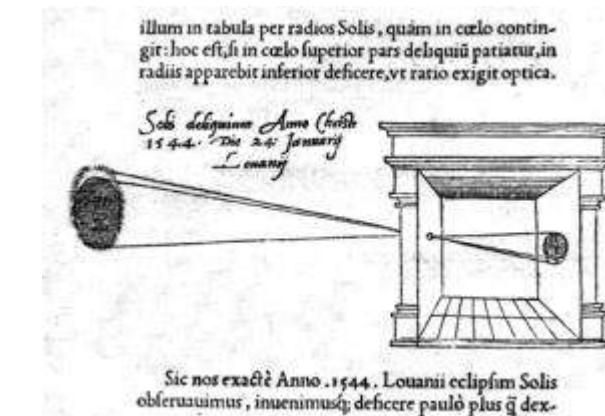
Dopo alcuni secoli, arrivò l'invenzione della prima Camera Obscura che si trasformerà in macchina fotografica molti secoli dopo. **Alhazen Ibn AlHaitam**, nato a Bassora nel 965 d.c. astronomo fisico e matematico arabo, diede una spiegazione più chiara del fenomeno. La scoperta venne utilizzata per studiare l'eclissi solare. "Se l'immagine del sole al momento di un'eclisse, purchè questa non sia totale, cade attraverso un forellino rotondo su di una superficie piana opposta, essa avrà la forma di una mezzaluna. L'immagine del sole rivela questa proprietà solo quando il foro è molto piccolo"

I testi viaggiarono attraverso il tempo e furono tradotti in diverse lingue fino ad arrivare a **Ruggero Bacone**, che ne tracciò uno schema esatto.

Gli studi arrivano fino in Italia, prima Brunelleschi usò il foro stenopeico che formava la camera obscura per riportare su carta e studiare le prospettive degli edifici e i monumenti, poi dopo qualche anno Leonardo Da Vinci.

Leonardo Da Vinci nel suo in Codice Atlantico ne descrive in maniera precisa il funzionamento denominandola oculus artificialis (occhio artificiale).

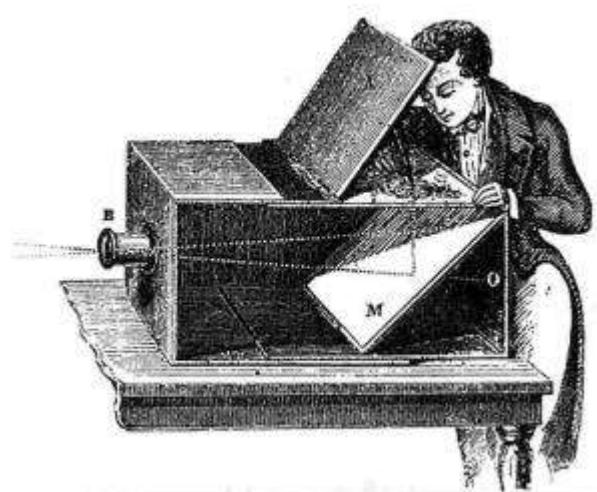
L'immagine, secondo Leonardo Da Vinci, doveva cadere su un foglio di carta "debole" (semitrasparente) per poter essere osservata e studiata. Sucessivamente l'invenzione perfezionata dall'inventore toscano fu suggerita da **Giovanni Battista della Porta**, nel trattato "Magiae Naturalis: sive de Miraculis rerum naturalium" (1558) per disegnare e ritrarre la realtà con più accuratezza. Nacque così in Italia la camera obscura.



Nei secoli l'utilizzo di questo apparecchio divenne sempre più popolare, arrivando a trasformarsi in un mezzo diffuso per la riproduzione dal 700 a metà dell'800.

Alcuni affermavano che la camera obscura, formata da una scatola con un forellino e un piano semitrasparente dove si formava l'immagine che veniva poi ricalcata, uno strumento per pittori inetti.

Questo diede il via alla controversia che si manifestò poi nella fotografia ottocentesca tra i primi fotografi e i pittori i quali accusavano i primi di essere pittori mancati. Esistono diversi tipi di camere obscure sparse per il mondo, da quelle in legno fatte per ritrarre il paesaggio o persone a vere e proprie stanze dotate di un foro nel muro che proiettava l'immagine dell'esterno all'interno della stanza. Il problema era riuscire a fissare l'immagine su un supporto e questo avvenne molto tardi, se valutiamo il tempo trascorso dalla scoperta del principio ad oggi.



Era infatti il 1822 quando **Joseph Nicéphore e Niepce** con la collaborazione di suo fratello scopre come fissare l'immagine su un supporto grazie al bitume di Giudea.

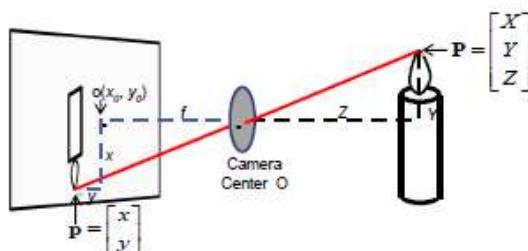
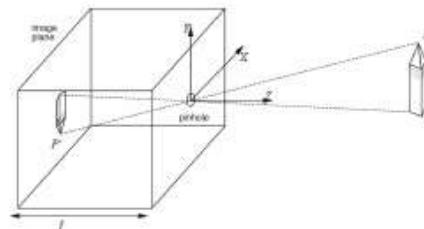
Nel 2015 Aimagelab, in occasione dell'eclissi, costruì una camera oscura con una distanza focale che potesse permettere la proiezione dell'eclissi. (Nei limiti del possibile dato che si doveva coprire la distanza dalla terra al sole).



The Pinhole MODEL as a Projection:

world coordinates → image coordinates

- Definitions:
 - Image plane: π
 - Optical center: O (or C camera center)
 - Image center: o
 - Oo: focal length (f)
 - Z: optical axis
- The model provides a perspective transform of a point $P=(X,Y,Z)$ in the 3-D world to a point $p=x=(x,y)$ on the image plane

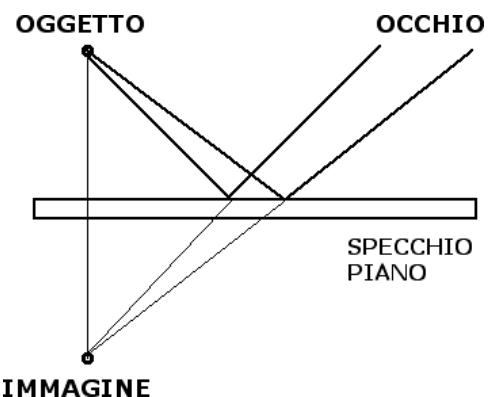


Prima di iniziare ad affrontare il tema del modello Pinhole, è necessario soffermarsi su alcuni concetti di ottica di base come quello della **riflessione**.

I raggi di luce che emergono da un oggetto, dopo la riflessione o la rifrazione da uno specchio o una lente, si incontrano in un punto specifico e formano una riproduzione dell'oggetto noto come immagine. Esistono due tipi di immagini che si formano, ovvero Immagine reale e Immagine virtuale. L'immagine reale è l'immagine che si forma quando i raggi di luce provenienti da una sorgente dopo la rifrazione o la riflessione convergono effettivamente in un punto mentre l'immagine virtuale si forma quando i raggi di luce sembrano divergere da un punto. L'immagine virtuale sembra trovarsi nel punto di divergenza. Un'immagine reale è un'immagine che può essere registrata a porte chiuse o può essere visualizzata sullo schermo mentre l'immagine virtuale non può essere visualizzata sullo schermo.

Per capire meglio la differenza tra immagine virtuale e reale consideriamo il caso di uno specchio piano. Dove appare l'immagine che si forma in uno specchio piano?

Ci è chiaro dalla nostra esperienza quotidiana che l'immagine di un oggetto che noi vediamo in uno specchio appare posizionata **dietro** lo specchio. Usando le leggi della riflessione possiamo capire la ragione di questo fenomeno. Osserviamo la figura.



I prolungamenti di due raggi riflessi si incrociano in un punto che diventa l'**immagine** dell'oggetto. Da semplici considerazioni geometriche si può dimostrare che la distanza tra l'oggetto e lo specchio è uguale alla distanza tra lo specchio e l'immagine dell'oggetto. Dal momento che tale immagine non esiste nella realtà ma si forma sul prolungamento dei raggi riflessi prodotto dal nostro cervello, si parla di **immagine virtuale**. Un'immagine si dice invece **reale** quando si forma al punto di intersezione di raggi luminosi reali.

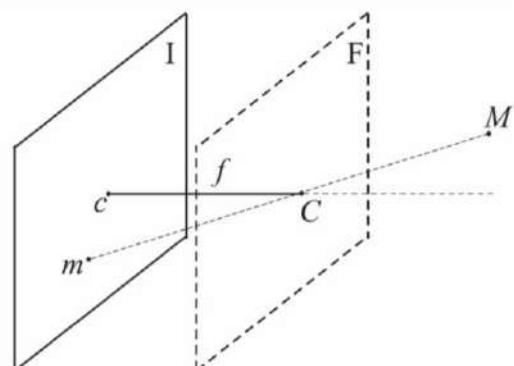
Pinhole camera con la proiezione prospettica

I parametri essenziali di un sistema pin-hole sono due:

- Il centro ottico **C**
- Un piano immagine o retina **I**

Fissati questi due parametri si derivano altri elementi notevoli:

- La distanza del centro ottico dal piano immagine, detta **distanza focale**, **f**
- L'asse ottico, ovvero la retta passante per **C** e ortogonale ad **I**
- Il punto principale **c** intersezione tra il piano immagine **I** e l'asse ottico
- Il piano focale **F** parallelo ad **I** e contenente il centro ottico.



Nella pagina seguente vedremo come è possibile individuare una relazione che lega i punti 3D ai punti sul piano tramite le ipotesi di proiezione prospettica.

Per derivare un modello analitico di una pin-hole camera, è necessario introdurre due sistemi di riferimento cartesiani:

1. Uno 3D per i punti della scena
2. Uno 2D per i punti dell'immagine

Partendo da un caso particolare:

- S.d.R destrorso (x, y, z) per la scena, centrato nel centro ottico C , con l'asse Z coincidente con l'asse ottico. Questa scelta prende il nome di **S.d.R standard**;
- S.d.R destrorso (u, v) per il piano immagine, centrato nel punto principale e con assi u e v orientati come x e y , rispettivamente.

Riprendendo dei concetti di *proiezione prospettica* consideriamo:

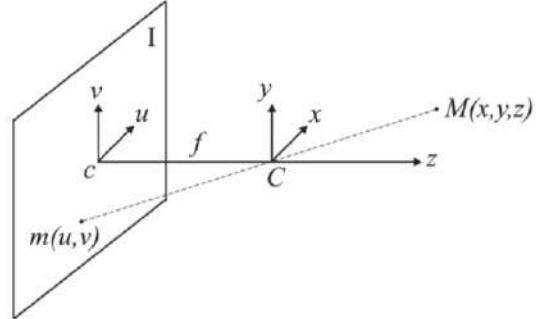
- un punto W di coordinate $w = [x, y, z]^T$ nel S.d.R standard
- la sua proiezione M su I attraverso C , di coordinate $m = [u, v]^T$

Semplici considerazioni di similitudine tra triangoli portano alla relazione

$$\frac{f}{z} = \frac{-u}{x} = \frac{-v}{y}$$

da cui la relazione **non lineare**

$$\begin{cases} u = -\frac{f}{z}x \\ v = -\frac{f}{z}y \end{cases}$$



È possibile esprimere le coordinate euclidee di m e w in coordinate omogenee:

$$\tilde{\mathbf{m}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad \tilde{\mathbf{w}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Usando le coordinate omogenee (e quindi considerando la trasformazione come tra spazi proiettivi), la proiezione prospettiva diviene una trasformazione lineare [uso slide prof]:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \xrightarrow{\text{homogeneous coordinates}} \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

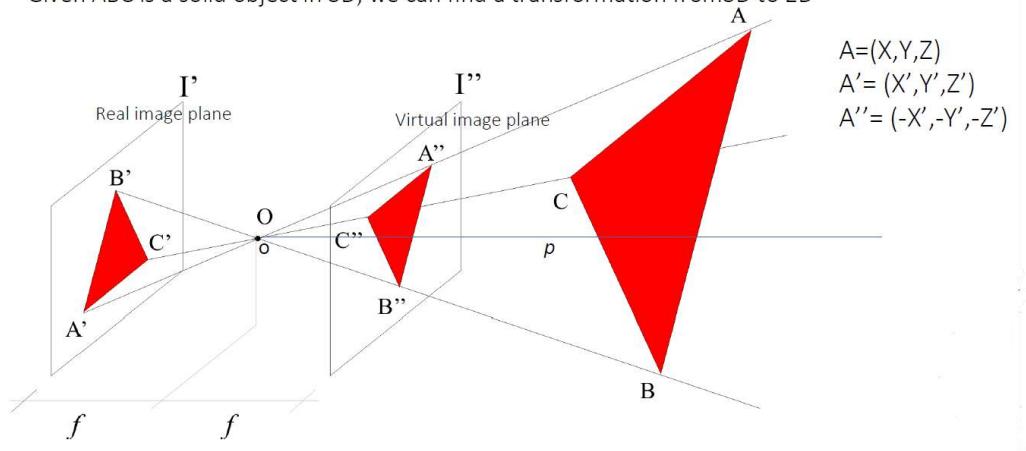
linear projection in homogeneous coordinates!

$$\begin{aligned} X' &= -f \frac{X}{Z} \\ Y' &= -f \frac{Y}{Z} \\ Z' &= -f \end{aligned}$$

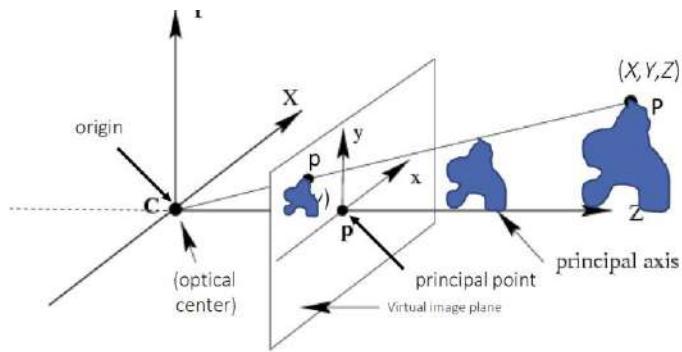
$\tilde{\mathbf{x}}' = \tilde{\mathbf{H}} \tilde{\mathbf{x}},$

where $\tilde{\mathbf{H}}$ is an arbitrary 4×4 homogeneous matrix.

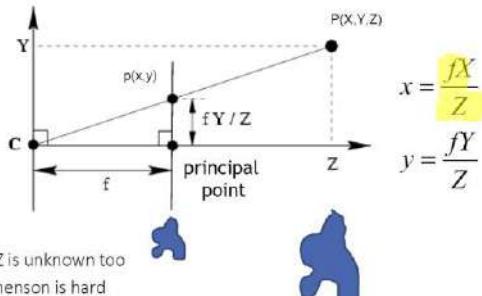
Given ABC is a solid object in 3D, we can find a transformation from 3D to 2D



Per completezza aggiungo le slide della prof da cui non si capisce niente:



Exploiting the similarity among triangles, $y:f=Y:Z$

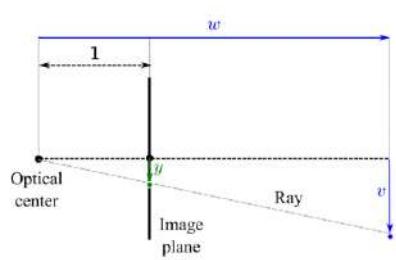


Non linear equations!

Often f is unknown, and Z is unknown too
Reconstruct the third dimension is hard

Abbiamo quindi potuto constatare che tramite le coordinate omogenee è possibile esprimere il pinhole model con una trasformazione lineare e tramite la teoria della prospettiva è possibile trasformare coordinate 3D in 2D.

Normalized camera model



Supponiamo di avere $f=1$ e l'immagine 2D con coordinate (x, y) centrato nel punto principale. Tramite la similarità tra triangoli, la posizione y nell'immagine di un punto (u, v, w) è data da il rapporto v/w . Quindi è possibile trasformare un punto $[u, v, w]$ in:

- $x = u/w$
- $y = v/w$

Allora si intuisce che **più un oggetto è lontano, più la sua proiezione si avvicina al centro dell'immagine.**

Proiezione ortografica

Un altro tipo di prospettiva è la prospettiva ortografica, le cui nozioni vengono applicate nel caso in cui un oggetto sia lontano dalla camera. L'oggetto è talmente lontano che Z è troppo grande rispetto a f , allora la distanza tra gli oggetti può considerata come costante nel caso della componente z .

$$\begin{aligned} Z_0 \pm \Delta Z & \quad X' = -f \frac{X}{Z_0} \\ \frac{\Delta Z}{Z_0} \ll 1 & \quad Y' = -f \frac{Y}{Z_0} \\ & \quad Z' = -f \end{aligned}$$

Quindi una proiezione ortografica elimina la componente z delle coordinate 3D.

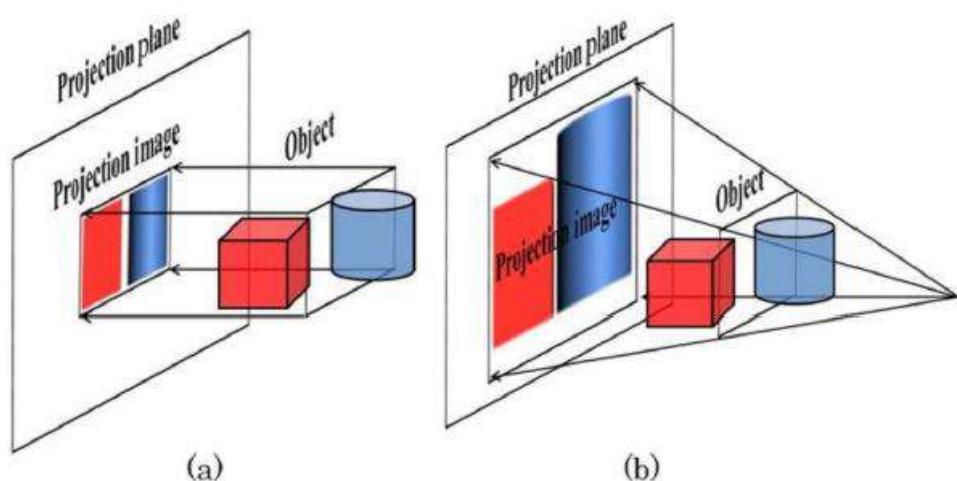
$$\mathbf{x} = [\mathbf{I}_{2 \times 2} | \mathbf{0}] \mathbf{p}.$$

If we are using homogeneous (projective) coordinates, we can write

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tilde{\mathbf{p}},$$

L'ultima colonna della matrice presenta comunque un 1 per evitare che il punto possa essere un punto all'infinito.

Orthographic (a) and perspective(b) transformation

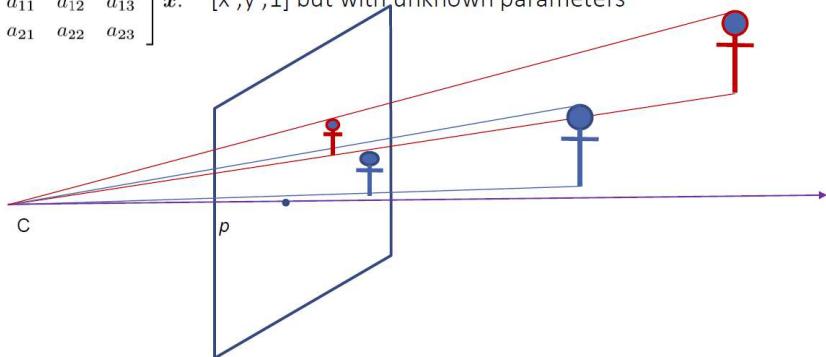


Proiezione Ortografica o ortografica, vuol dire letteralmente proiettare un oggetto da rappresentare su di un piano ad esso perpendicolare (ortognale). E' come se illuminassimo un oggetto con una torcia e ne osservassimo la sua ombra proiettata su una parete (piano) posta alle sue spalle.

La proiezione ortografica viene usato perlopiù in immagini satellitari ecc.

La perdita della terza dimensione di oggetti in una scena comporta che non si ha più informazione in merito alla distanza relativa fra gli oggetti e alle loro dimensioni. Con la proiezione prospettica invece è possibile notare che due oggetti anche se sembrano avere dimensioni diverse (uno più alto un più basso) in realtà sono semplicemente posizionati in prospettiva.

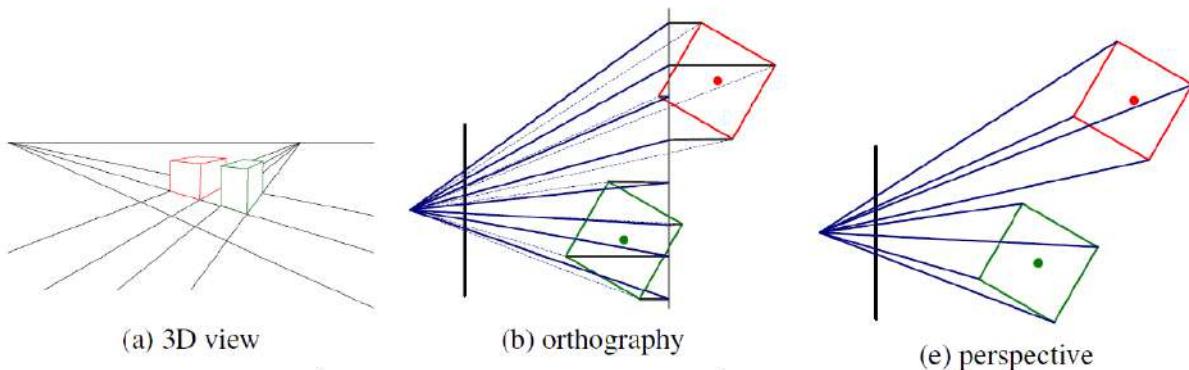
$\bar{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{bmatrix} \bar{x}$. There is a perspective transformation from a 3D point $[x,y,z,1]$ to a 2D point $[x',y',1]$ but with unknown parameters



Come è possibile vedere da questa immagine il fatto che la persona sia più lontana in prospettiva e quindi risulti più piccola non comporta che questa persona sia necessariamente 4/5 volte più bassa della persona in prospettiva più vicina all'obiettivo.

In questo corso non si farà riferimento a una proiezione ortografica, però è importante sapere che esiste una diversa modalità di proiezione degli oggetti in una scena.

Per esempio l'occhio umano dopo 6 metri non riesce più a percepire la terza dimensione.



Use of correct perspective projection indicated in 1st century B.C. frescoes
 Skill resurfaces in Renaissance: artists develop systematic methods to determine perspective projection (around 1480-1515)



Raphael



Durer, 1525

Applications of projective geometry



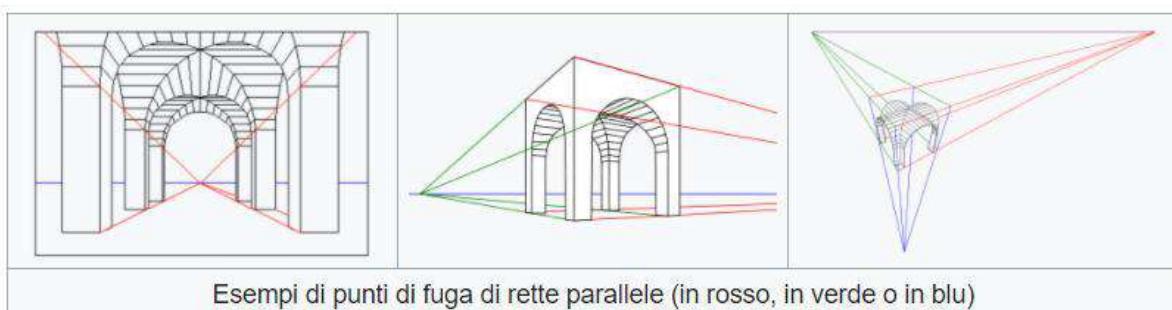
Vermeer's Music Lesson



Reconstructions by Criminisi et al.

Vermeer è stato il primo pittore che utilizzava la prospettiva senza conoscerne la geometria alla base.

Come ultimo punto da trattare nel caso delle proiezioni prospettiche è il concetto di punto di fuga o vanishing point. Il **punto di fuga o fuoco** è un punto verso il quale le linee parallele sembrano convergere. In particolare, il punto di fuga di una retta r è un punto F_r sul piano di proiezione, comune alle immagini prospettiche di ogni retta parallela a quella data.



Basi di ottica

Prima di affrontare il tema della camera calibration è importante mettere le basi su alcuni concetti di ottica.

Come abbiamo già accennato, la dimensione del pinhole può condizionare la qualità della foto. Perché più luce entra all'interno del centro focale più l'immagine sarà illuminata, e questo causerà una sfocatura dovuta al fatto che la luce da un solo punto sull'oggetto può toccare più punti sullo schermo.

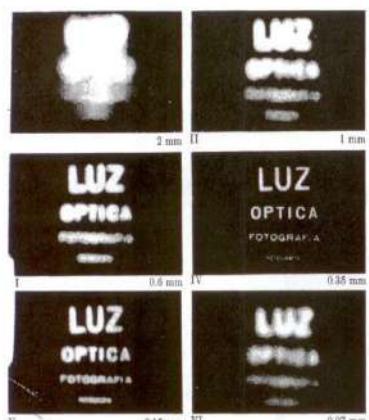
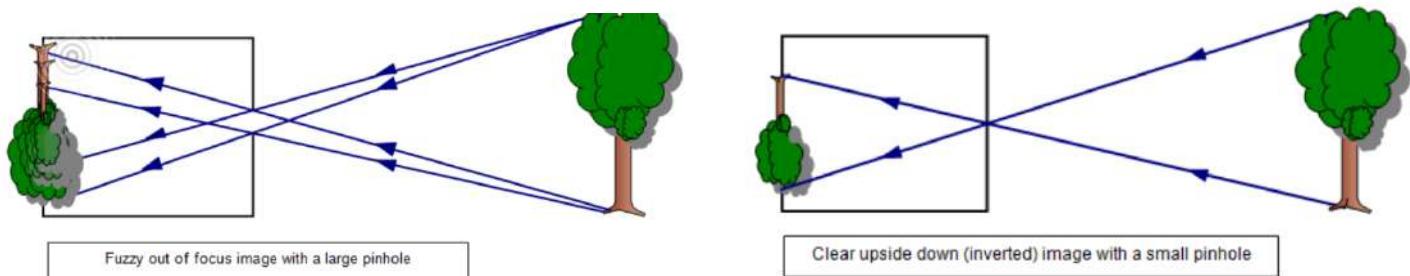
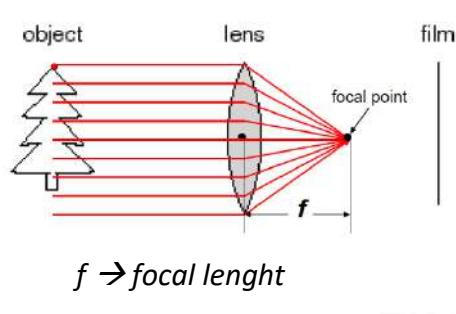
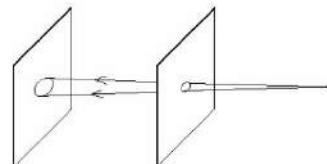


Fig. 5.96 The pinhole camera. Note the variation in image clarity as the hole diameter decreases. [Photos courtesy Dr. N. Joel, UNESCO.]

Larger

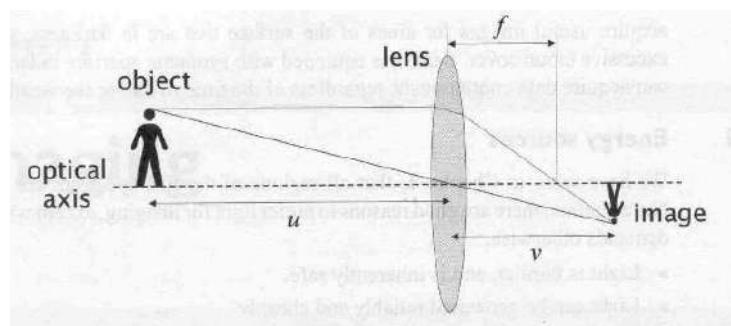
Smaller



$f \rightarrow$ focal lenght

Aggiunta della lente

Una lente è un elemento ottico che ha le proprietà di concentrare o far divergere i raggi di luce. Grazie a questa proprietà può formare immagini, reali o virtuali, di oggetti.

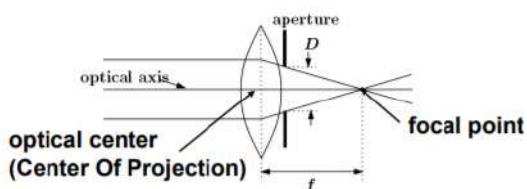


Prendendo in considerazione il solo pinhole model, abbiamo visto che se il foro si allarga allora il quantitativo di luce dalla sorgente si diffonde sull'immagine e la rende sfocata. Se il foro si restringe, il quantitativo di luce dalla sorgente si riduce allora si riduce la nitidezza dell'immagine a causa della diffrazione.

L'uso delle lenti permette di duplicare la geometria del pinhole senza dover trovare la giusta dimensione del pinhole, basta disporre gli oggetti nella scena perché siano messi a fuoco dalla lente.

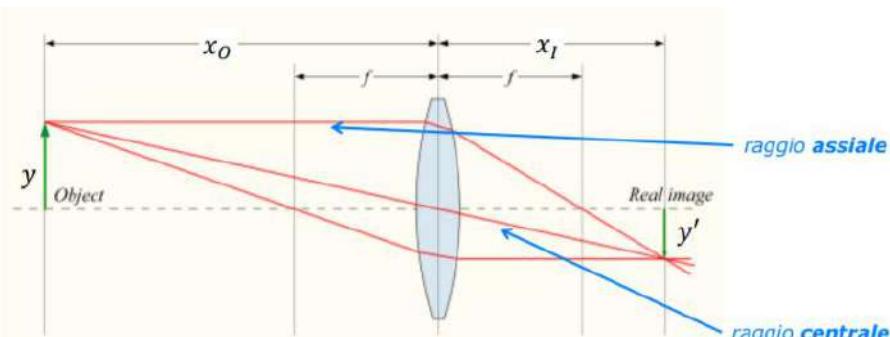
- Un pinhole cattura un raggio di luce (linea retta oggetto – immagine)
- Una lente cattura tutti i raggi di luce che raggiungono la sua apertura

L'apertura di una lente si misura con il rapporto diametro della stessa diviso la sua lunghezza focale.

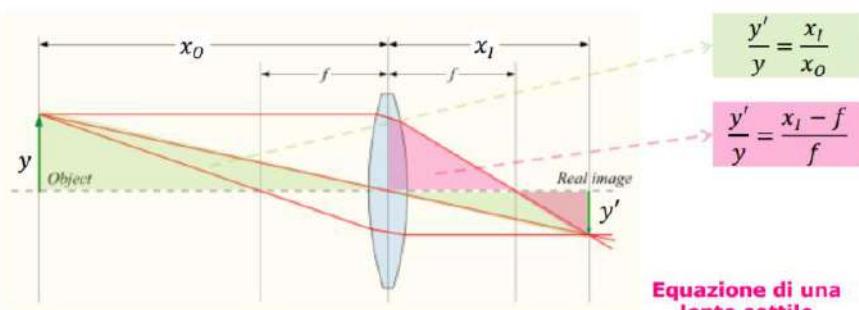


$$\text{Apertura: } A = \frac{D}{f}$$

Modello geometrico di una "lente sottile"



- Il raggio centrale, che passa per il centro della lente, non viene deviato
- Il raggio assiale parallelo all'asse ottico viene deviato e passa per il fuoco.



Equazione di una lente sottile

Eguagliando le due equazioni e dividendo per x_I :

$$\frac{x_I}{x_o} = \frac{x_I - f}{f} = \frac{x_I}{f} - 1 \rightarrow \frac{1}{x_o} = \frac{1}{f} - \frac{1}{x_I}$$

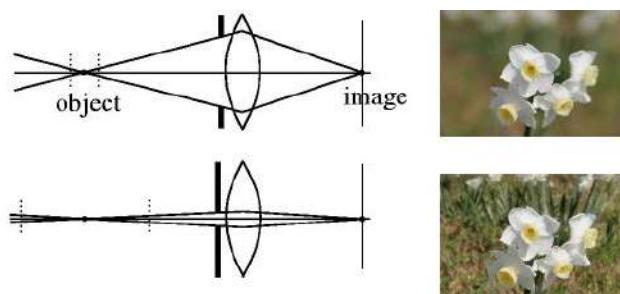
$$\frac{1}{x_I} + \frac{1}{x_o} = \frac{1}{f}$$

In sintesi, il modello geometrico non cambia particolarmente, ma con la lente $f \rightarrow x_1$ dove x_1 dipende dalla distanza dell'oggetto x_0 : $x_1 = f(x_0)$.

La **profondità di campo** rappresenta la zona in cui gli oggetti nell'immagine appaiono ancora nitidi e sufficientemente focalizzati. I fattori che incidono sulla profondità di campo sono:

- La lunghezza focale, è la distanza in mm fra il centro ottico dell'obiettivo e il piano focale che, nel caso della fotografia digitale, è rappresentato dal tensore
- La distanza dal soggetto
- L'apertura del diaframma.

Il diaframma è un apertura poligonale posta all'interno dell'obiettivo. Variando la sua apertura, è possibile scegliere la quantità di luce che andrà a colpire il sensore della macchina fotografica nel tempo di otturazione prestabilito. La profondità di campo è fortemente influenzata dall'apertura del diaframma, un diaframma molto aperto ci darà una profondità di campo ridotta e quindi soggetto in primo piano e sfondo sfocato. Il diaframma chiuso permette di avere una massima profondità di campo quindi soggetto e sfondo ben definiti.



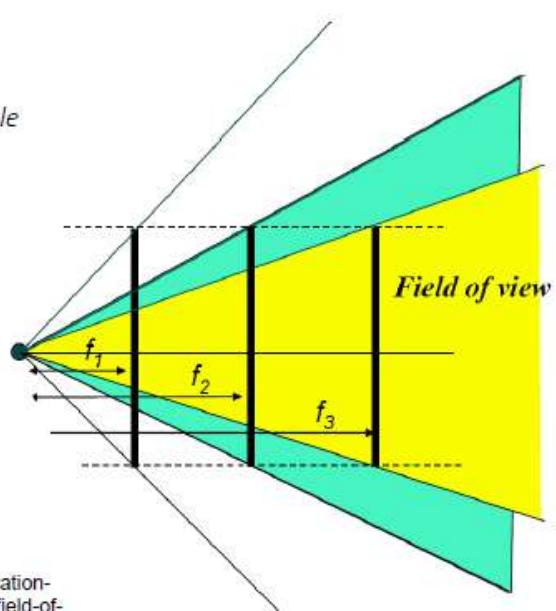
- A smaller aperture increases the range in which the object is approximately in focus

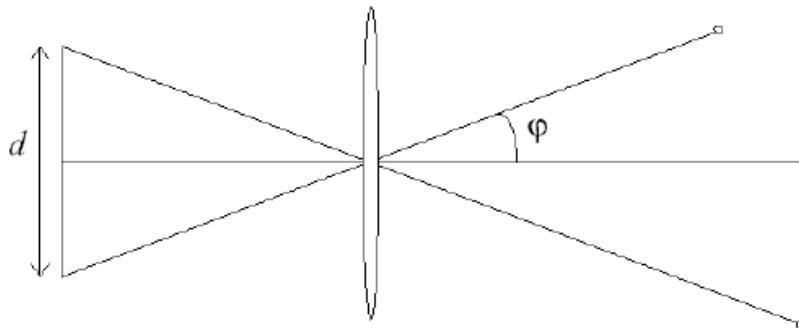
Il campo visivo è una misura angolare dello spazio che è in grado di coprire la camera con il suo obiettivo.

Field of view and focal length

As f gets smaller, image becomes more *wide angle*
more world points project onto the finite image plane

As f gets larger, image becomes more *telescopic*
smaller part of the world projects onto the finite image plane





Da questa immagine è possibile vedere la relazione tra campo visivo e lunghezza focale.

Size of field of view governed by size of the camera retina:

$$\varphi = \tan^{-1}\left(\frac{d}{2f}\right)$$

Smaller FOV = larger Focal Length

ANSWER

Fish Eye Len: si tratta di una lente estremamente grandangolare che abbraccia un angolo di campo non minore di 180 gradi. A differenza degli obiettivi grandangolari a prospettiva rettilinea, i fisheye non possono evidentemente fornire un'immagine non distorta. In particolare l'immagine è sempre più distorta quanto più ci si allontana dal centro.

Esistono diversi metodi per ottenere queste distorsioni: nella maggior parte dei fisheye per uso fotografico si opta per la cosiddetta proiezione equidistante.

Calibration

La visione artificiale è l'insieme dei processi che mirano a creare un modello approssimato del mondo reale 3D partendo da immagini bidimensionali 2D. Per fare ciò, si necessita di equazioni che leghino il mondo reale a quello descritto in una o più immagini delle medesima scena.

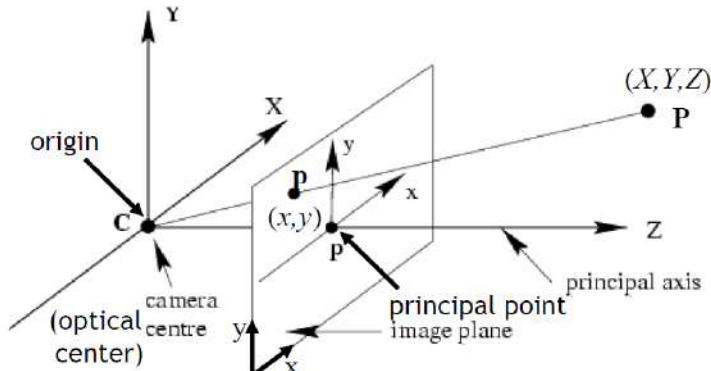
Queste equazioni sono scritte nel sistema di riferimento della telecamera e molto spesso si assume che:

- Il sistema di riferimento della telecamera è posizionato rispetto a un altro sistema di riferimento, conosciuto come sistema di riferimento mondo
- Le coordinate dei punti dell'immagine (o delle immagini) nel sistema di riferimento della telecamera possono essere ottenute dalle coordinate pixel, le uniche direttamente disponibili dall'immagine.

A questo scopo abbiamo bisogno di conoscere le caratteristiche della telecamera (o telecamere, se si parla di visione stereoscopica), conosciute come parametri intrinseci e parametri estrinseci. Il prossimo passo è quindi quello di descrivere l'esatta natura di questi parametri. Il processo che determina i parametri sia intrinseci che estrinseci è detto *camera calibration*.

Partiamo da i parametri **intrinseci**, sono parametri necessari a collegare le coordinate pixel di un'immagine con le corrispondenti coordinate nel sistema di riferimento della telecamera; dipendono infatti dalla telecamera stessa1. Il primo parametro intrinseco è **la lunghezza focale f**.

Model 1. The origin of the world coordinates is the camera center C, and the origin of the (virtual)image plane is the principal point of coordinates (0,0,f)
In homogenous coordinates, the perspective or homographic transformation becomes linear. In the very ideal case: the H matrix is simple



1. First intrinsic parameter: f

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} & h_{03} \\ h_{10} & h_{11} & h_{12} & h_{04} \\ h_{20} & h_{21} & h_{23} & h_{05} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\boxed{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}$$

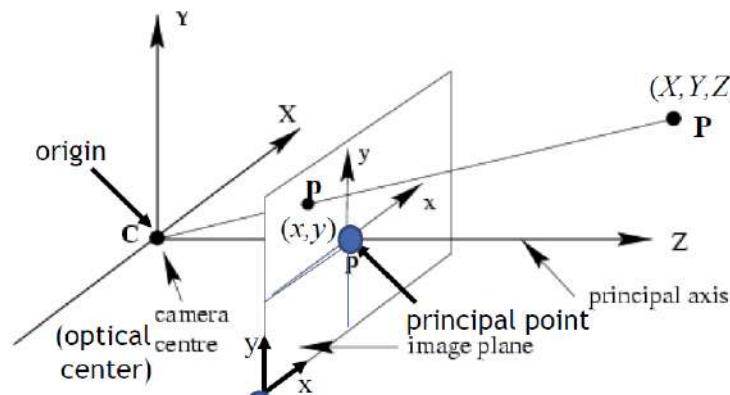
Intrinsic matrix K $[K | 0]$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

(z in the image plane does not exist
is 0 if the transformation is from 3D to 2D)

Il secondo parametro intrinseco sono le coordinate **(x0, y0)** del punto principale dell'immagine, questo punto infatti non coincide con l'origine del piano dell'immagine.

Model 2. if the origin in the image plane is translated with respect the principal point, and the pricipal point has coordinates **(x0,y0)** in the image plane (i.e. **(x0,y0,f)** in the space) we must add a TRANSLATION



2. Second pair of intrinsic parameters: x0, y0

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & x_0 & 0 \\ 0 & f & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Il terzo modello aggiunge ai parametri intrinseci la **pixelizzazione**. Nel caso in cui i pixel non siano quadrati ma per esempio rettangolari (può succedere nei video), la distanza focale deve tenere conto di una distorsione della proiezione sia sull'asse x che sull'asse y. Quindi cambiano le formule di proiezione.

Model 3: PIXELIZATION In case of non-square pixels (digital video) the focal distance has a distortion pixels are not squared and the focal length is different among x and y

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & 0 & x_0 & 0 \\ 0 & fy & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$fx = fax \\ fy = fay$$

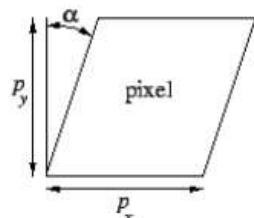


two form factors of pixelization

Using in homogenous coordinate the INTRINSIC MATRIX IS GIVEN BY FOUR PARAMETERS (which could be given by the camera, or learned by CALIBRATION)

L'ultimo parametro intrinseco è lo **skew**. Lo skew è un parametro che serve alla calibrazione nel caso in cui l'angolo compreso fra i due assi all'origine del image plain non sia retto. Lo skew è un parameter che permette di effettuare una proporzione tra l'angolo di inclinazione fra i due assi.

Model 4. The most general form considers also a 5° parameter:
the **skew** which is the real angle between X and Y axes



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & s & x_0 & 0 \\ 0 & fy & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Normally s is fixed to 0 (especially in new digital CMOS cameras)

non-square pixels (digital video)
skew

General INTRINSIC MATRIX

5 intrinsic parameters , depending on the camera

- xc, yc translation of origin
- fx, fy focal length and pixelization
- S skew

$$\mathbf{K} = \begin{bmatrix} f_x & s & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & s & xc & 0 \\ 0 & fy & yc & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

[K | 0]

- They can also be written depending on the ratio in pixelization

I parametri **estrinseci** sono parametri che definiscono la posizione e l'orientamento del sistema di riferimento della telecamera rispetto al sistema di riferimento mondiale. Il sistema di riferimento della telecamera è molto spesso sconosciuto. Un problema comune è determinare la posizione e l'orientamento rispetto a un sistema di riferimento sconosciuto, utilizzando esclusivamente informazioni ottenute attraverso le immagini catturate. I parametri estrinseci sono definiti come un set di parametri geometrici che identifica univocamente la trasformazione tra il sistema di riferimento della telecamera a noi sconosciuto e il sistema di riferimento mondo a noi noto.

Una tipica scelta per descrivere la trasformazione tra questi due sistemi di riferimento consiste nell'usare:

- Un vettore di traslazione \mathbf{T} a tre dimensioni che descrive la relativa posizione delle origini nei due sistemi di riferimento.
- Una matrice di rotazione \mathbf{R} 3×3 ortogonale, che porta gli assi corrispondenti dei due sistemi di riferimento l'uno sopra l'altro.

Rotation+ translation

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \mathbf{R}_{3 \times 3} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \mathbf{t}$$

$$\mathbf{x} \sim \mathbf{K} \underbrace{[\mathbf{R}|\mathbf{t}]}_{[\mathbf{R}|\mathbf{t}]} \mathbf{X}$$

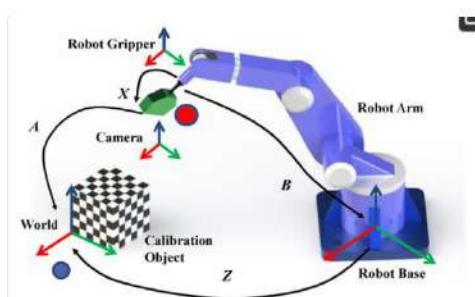
extrinsic matrix

$$p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & s & x_0 \\ 0 & fy & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & tx \\ r_{21} & r_{22} & r_{23} & ty \\ r_{31} & r_{32} & r_{33} & tz \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

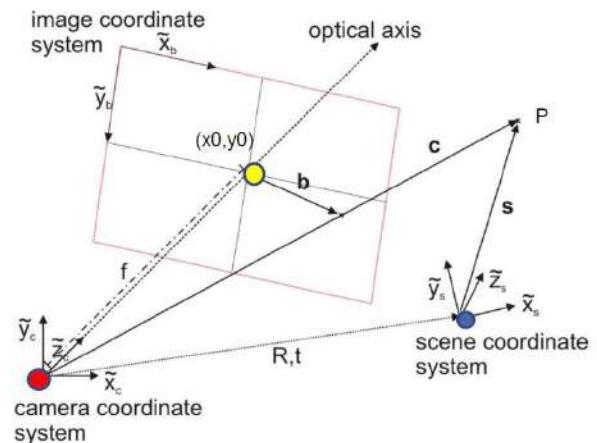
Scene (world) coordinate system: the coordinates of the world

Camera coordinate system: the coordinates of the camera

The image coordinate system

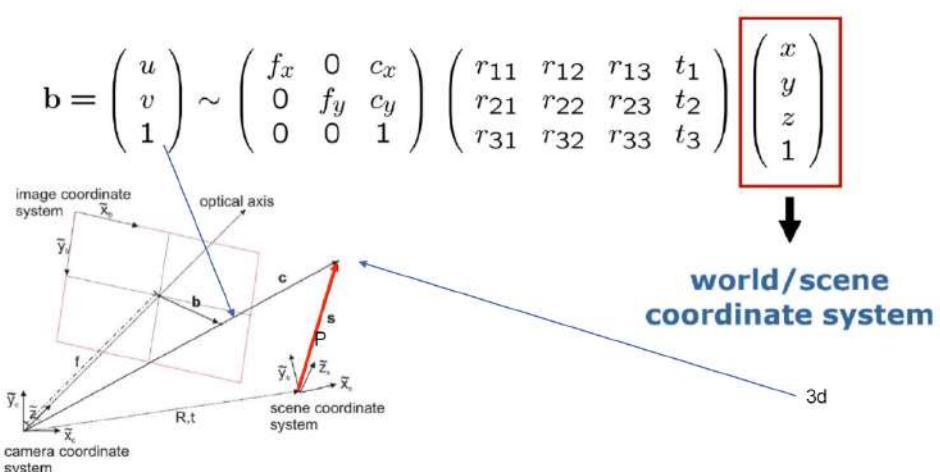


We want compute \mathbf{s} but
we have \mathbf{b} only



(spiegazione incomprensibile di queste ultime due slides, attacco slides e faccio una preghiera)

We have the measure of b with respect to the image plane and we need to know the measure of s w.r.t. the scene coordinates



Infine possiamo dire che

internal or intrinsic parameters: focal length, optical center, skew and distortion
external or extrinsic (pose): rotation and translation:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{R}|\mathbf{t}] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

This is the complete (perspective) camera model

Dove la matrice 4×4 è la matrice di prospettiva, questa presenta 11 gradi di libertà.

Gradi di libertà – Degrees of freedom

$$P = K[\mathbf{R}|\mathbf{t}]$$

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \mathbf{X}$$



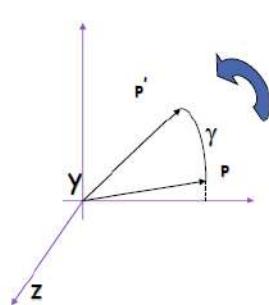
$$w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 5 & & & 6 & \\ f_x & s & cx & r_{11} & r_{12} & r_{13} & t_x \\ 0 & f_y & cy & r_{21} & r_{22} & r_{23} & t_y \\ 0 & 0 & 1 & r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Intrinsic parameters Extrinsic parameters

5 parametri dalla matrice intrinseca e 6 dalla matrice estrinseca, 3 per le rotazioni degli angoli, 3 per la traslazione.

11 elementi che devono essere calcolati.

Rotation around the coordinate axes, counter-clockwise:
The rotation matrix is the product of the three rotations



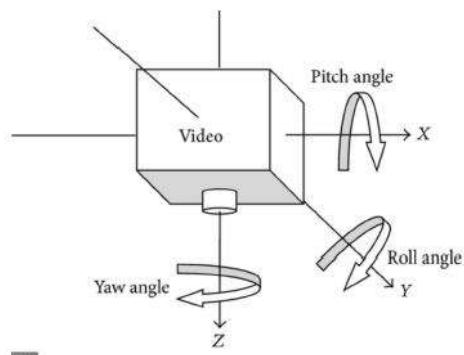
$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Le componenti di rotazione della camera sono dette:

- Pitch, l'angolo intorno all'asse x
- Roll, l'angolo intorno all'asse y
- Yaw, l'angolo intorno all'asse z



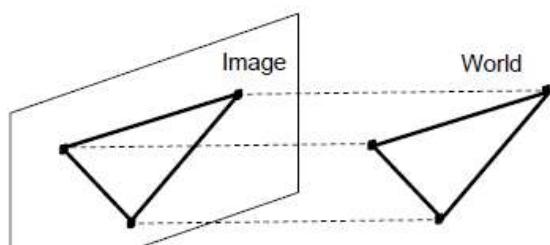
$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \varphi & 0 & \sin \varphi \\ 0 & 1 & 0 \\ -\sin \varphi & 0 & \cos \varphi \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \psi & \sin \psi \\ 0 & -\sin \psi & \cos \psi \end{pmatrix}$$

Le slide 60-61-62-63 le ha saltate a lezione ma io le metto per completezza

Orthographic projection

Special case of perspective projection

Distance from the Center to the Principal point is infinite



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

– Also called “parallel projection”: $(x, y, z) \rightarrow (x, y)$

1. Perspective

$$x = X \frac{f}{Z} \quad y = Y \frac{f}{Z}$$

2. Weak perspective

$$x = \text{const } X \quad y = \text{const } Y$$

3. Orthographic

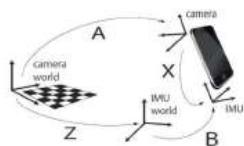
$$x = X \quad y = Y$$

Considerazioni finali su Camera Calibration:

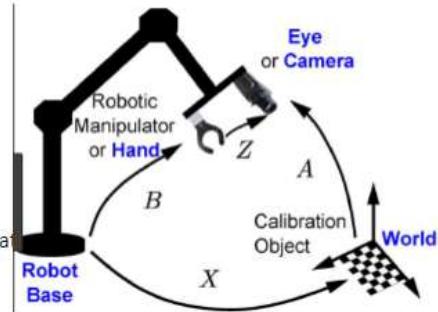
Una buona calibrazione della telecamera è importante quando:

- È necessario **ricostruire un modello dal mondo**: cioè riportare una costruzione 3D nel mondo virtuale dal mondo reale;
- È necessario **interagire con il mondo reale**: robot, coordinazione braccio-occhio;

Knowing the real world dimension by single images we can calibrate cameras

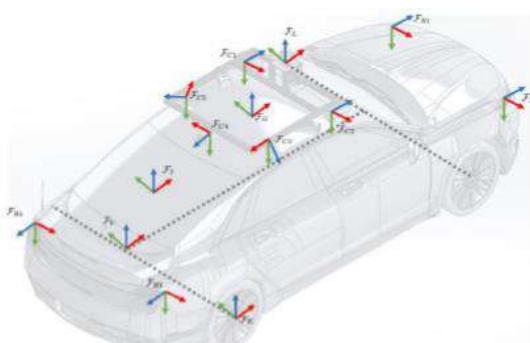


In mobile we need also the IMU inertial measurement unit calibration



Come possiamo vedere al giorno d'oggi, le automobili presentano diverse telecamere/sensori disposte sulla carrozzeria per facilitare manovre, per implementare la sicurezza ecc. Tutti questi sensori devono essere calibrati correttamente per fare in modo che ci sia un collaborazione e una condivisione dell'informazioni tra di loro.

- Data from multiple complementary sensors leads to increased accuracy and robustness
- Sensors need to be spatially and temporally calibrated
- Calibration helps transform measurements from different sensors into a common reference frame



Infine, con calibrazione della camera intendiamo la stima dei parametri intrinseci [Hartley and Zisserman 2004], i parametri di calibrazione più importanti per FOV camera (Field of view camera) sono la lunghezza focale e la distorsione.

I metodi di calibrazione che vengono usati sono i seguenti:

1. **Multiple images:** il metodo della scacchiera, dove si estraggono gli angoli. È molto preciso, richiede diverse immagini, è un metodo manuale, ma è difficile da applicare nel caso di wide DoG;
2. **Usare diverse strutture geometriche della scena,** come linee o punti di fuga; selezionare alcune linee esistenti, nel caso di scene composte da soli uomini allora una metrica deriva da una singola immagine (?)
3. **Camera self calibration:** stima i parametri intrinseci muovendo la telecamera, risolve il problema di calibrazione nel caso della distorsione usando linee epipolari (Fitzgibbon Simultaneous Linear Estimation of Multiple View Geometry and Lens Distortion. In CVPR. 2001)
4. **Deep Learning,** ma non esistono ad oggi tecniche di calibrazione con deep learning efficaci:
 - a. DeepFocal [Workman et al. 2015] predice solo la lunghezza focale e non la distorsione. Si allena con l'uso di solo poche immagini, circa 7000.
 - b. Ronget al. [2016] stima solo la distorsione radiale e non la lunghezza focale.
 - c. Hold-Geoffroyet al. [2018] stima la lunghezza focale e l'orientazione della camera, ma non la distorsione

Metodi per stimare i parametri intrinseci ed estrinseci

Partendo dal modello matematico che abbiamo visto fino ad ora:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R | t \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

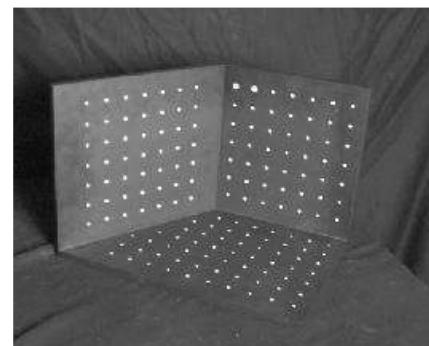
$$\mathbf{P} = \mathbf{K}[\mathbf{R}|t]$$

$$\tilde{\mathbf{P}} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & t \\ \mathbf{0}^T & 1 \end{bmatrix} = \tilde{\mathbf{K}}\mathbf{E},$$

$$\mathbf{x}_s \sim \tilde{\mathbf{P}}\bar{\mathbf{p}}_w, \quad \mathbf{v}_s = (x_s, y_s, 1, d), \quad \bar{\mathbf{p}}_w = (x_w, y_w, z_w, 1)$$

Il primo metodo che vedremo si basa sulla **conoscenza della geometria di oggetti come riferimento**. Dal punto di vista matematico è perfetto come metodo, però non viene più utilizzato nella pratica.

Nel passato per calibrare una fotocamera utilizzavano l'oggetto nell'immagine con dei punti fissi. Calcolando ogni punto sull'oggetto è possibile calcolare gli 11 parametri che servono.



$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}|t]\mathbf{X} = \mathbf{M}\mathbf{X}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Da questa notazione matriciale possiamo ottenere

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i \end{bmatrix} = \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

Solve for Projection Matrix M using least-square techniques

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 \\ \vdots & & & & & & & & & & \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -u_NX_N & -u_NY_N & -u_NZ_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -v_NX_N & -v_NY_N & -v_NZ_N \end{bmatrix} = \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

Solve for Projection Matrix M using least-square techniques



Carl Friedrich Gauss

Questo metodo guassiano però non è più utilizzato a causa del suo grosso carico computazionale.

Il secondo metodo che vedremo è, invece, il più famoso e il più utilizzato. Può essere implementato tramite OpenCV.

A Flexible New Technique for Camera Calibration

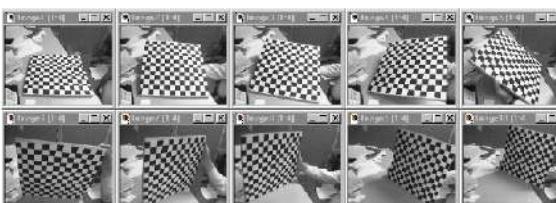
Zhengyou Zhang
Microsoft Research
IEEE Transaction on pattern analysis
And Machine intelligence, Nov. 2000



Il **metodo di Zhang** è un metodo di calibrazione semplice e prevede l'acquisizione, nella stessa immagine o in immagini successive, di pattern piani caratterizzati da giaciture diverse (traslati o ruotati). Questa tecnica sfrutta il calcolo di diverse matrici omografiche H ottenute dall'osservazione di un piano (per esempio una griglia di calibrazione con marker equispaziati - chessboard). Il grosso vantaggio risiede nel fatto che non è necessario conoscere la posizione relativa di tali piani, quindi risulta essere un metodo estremamente pratico.

Zhang ha quindi sviluppato un metodo che permette di ottenere un'equazione lineare per ricavare i parametri della camera.

Multi-plane calibration self-calibration



Images courtesy Jean-Yves Bouguet, Intel Corp.

Advantage

- Only requires a plane
- Don't have to know positions/orientations
- Good code available online!
 - Intel's OpenCV library: <http://www.intel.com/research/mri/research/opencv/>
 - Matlab version by Jean-Yves Bouguet: http://www.vision.caltech.edu/bouguet/calib_doc/index.html
 - Zhengyou Zhang's web site: <http://research.microsoft.com/~zhang/Calib/>

Quante immagini servono per poter calcolare i parametri ?

Considerando che lo skew $s=0$, quindi ci sono 4 parametri intrisici (costanti) e 6 parametri estrinseci 3 angoli e 3 rotazioni, variabili. Avendo N_i immagini con M angoli ciascuna:

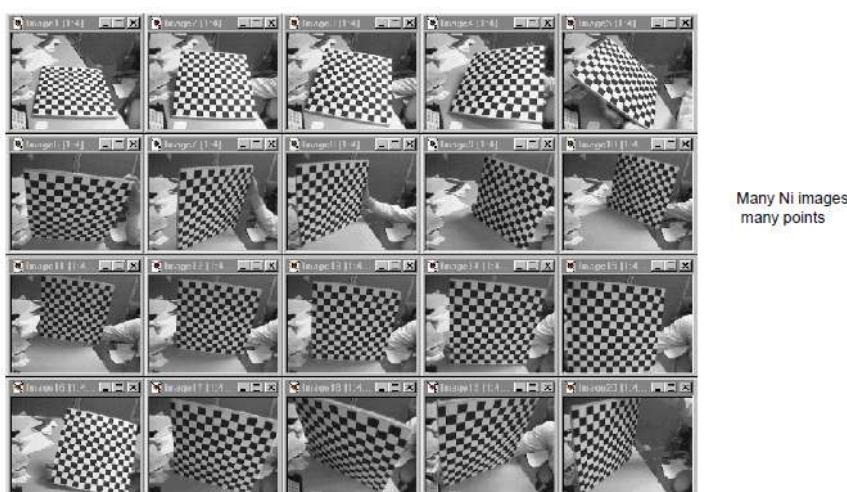
- $4 + 6 \cdot N_i$ parametri
- $2 \cdot N_i \cdot M$ constraints ($2 \cdot M$ coordinate per ogni immagine)

Deve essere

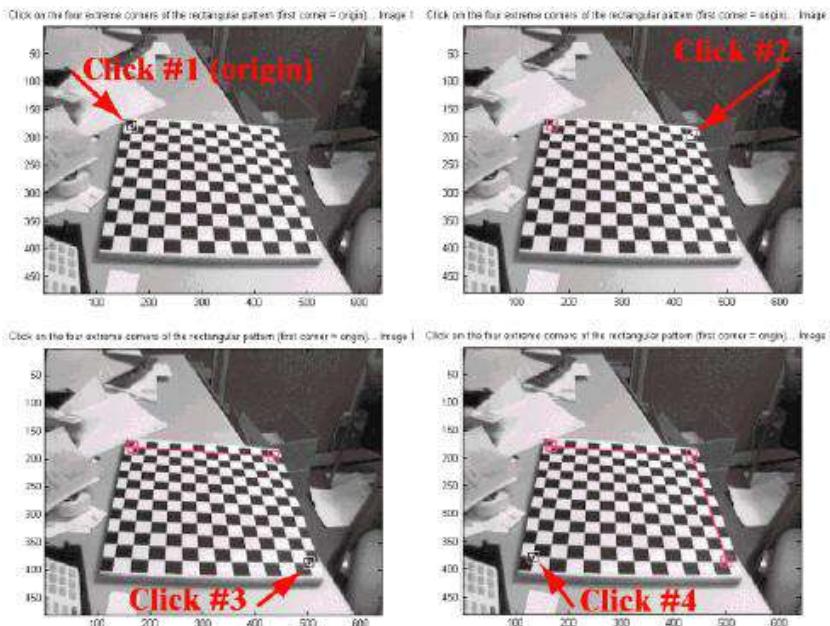
$$2 \cdot N_i \cdot M \geq 4 + 6 \cdot N_i \rightarrow N_i \geq 2 / (M-3)$$

Quindi 2 immagini con 4 punti è abbastanza.

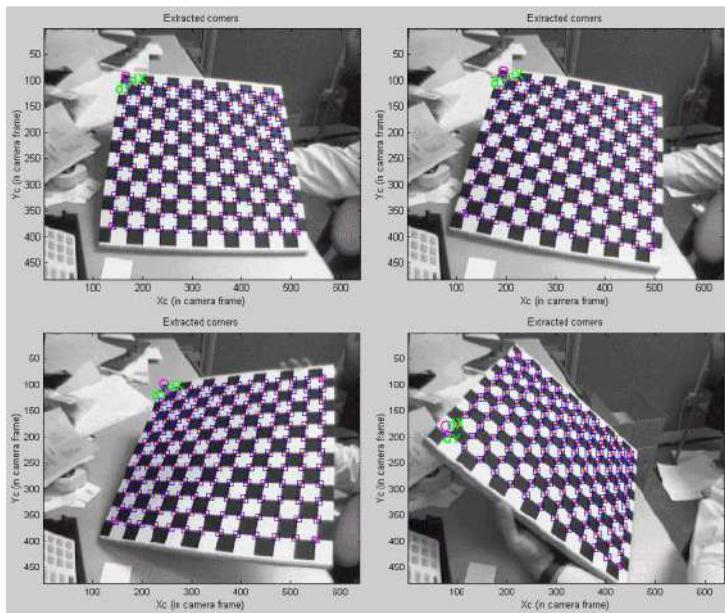
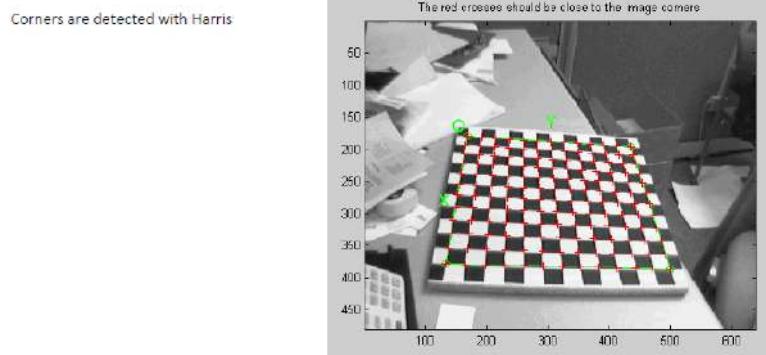
Metodo Zhang: step 1 – Data acquisition



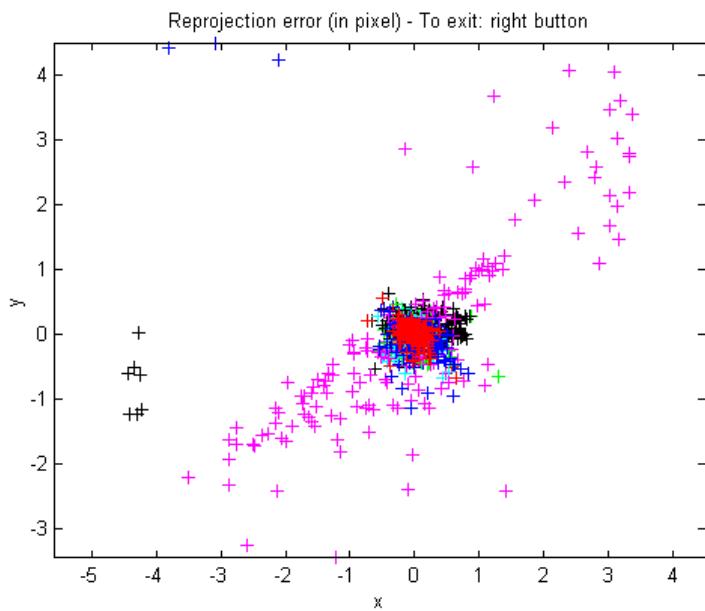
Step 2 – Specify corner order



Step 3 – Corner extraction



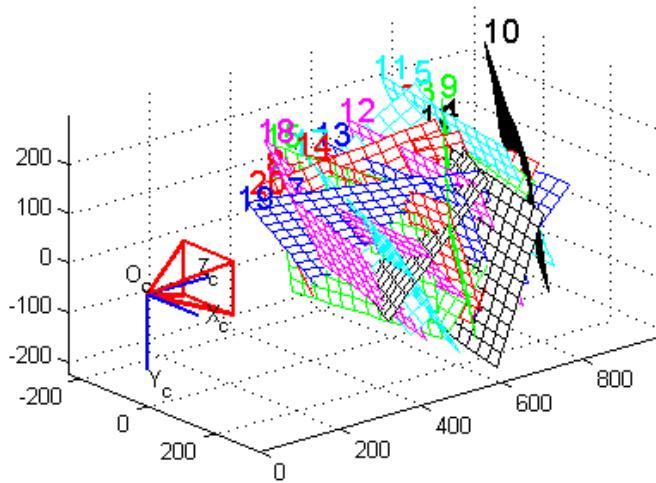
Step 4 – Minimize projection error



Calibration results after optimization (with uncertainties):

Focal Length: $f_c = [657.46298 \quad 657.94673] \pm [0.31819 \quad 0.34646]$
Principal point: $c_c = [303.13665 \quad 242.56935] \pm [0.64682 \quad 0.59218]$
Skew: $\alpha_c = [0.00000] \pm [0.00000] \Rightarrow$ angle of pixel axes =
Distortion: $k_c = [-0.25403 \quad 0.12143 \quad -0.00021 \quad 0.00002 \quad 0.00000]$
Pixel error: $err = [0.11689 \quad 0.11500]$

Step 5 – Camera calibration



Structure from Motion

(questa slide non l'ha fatta ma la metto per completezza)

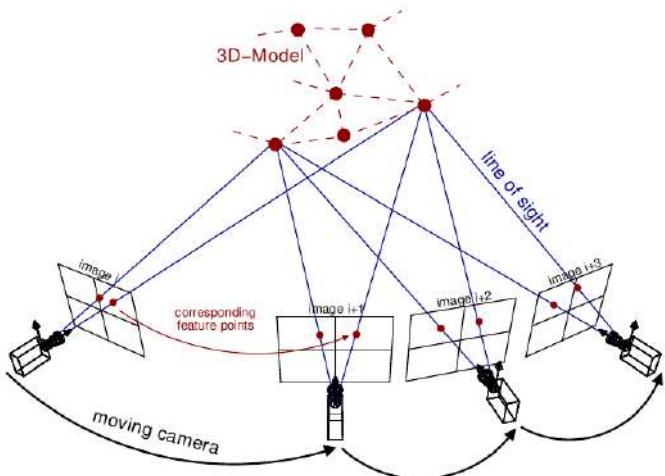
Given many points in correspondence across several images, $\{(u_{ij}, v_{ij})\}$, simultaneously compute the 3D location \mathbf{x}_i and camera (or motion) parameters $(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j)$

$$\begin{aligned}\hat{u}_{ij} &= f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \\ \hat{v}_{ij} &= g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i)\end{aligned}$$

Two main variants: calibrated, and uncalibrated (sometimes associated with Euclidean and projective reconstructions)

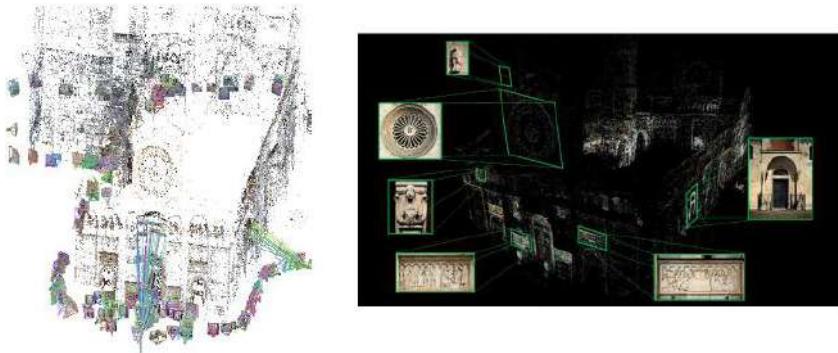
How many points do we need to match?

- 2 frames: (\mathbf{R}, \mathbf{t}) : 5 dof + $3n$ point locations $\leq 4n$ point measurements $\Rightarrow n \geq 5$
- k frames: $6(k-1)-1 + 3n \leq 2kn$
- always want to use many more



SfM è una tecnica di fotogrammetria che permette di stimare un modello 3D, sovrapponendo sequenze di immagini 2D che vengono scattate dalla stessa camera da diverse angolazioni come è possibile vedere nell'immagine.

Alla base di questa tecnica c'è una considerazione molto semplice, gli esseri umani percepiscono molte informazioni sul ambiente tridimensionale in cui si trovano muovendosi intorno ad esso. Un osservatore in movimento percepisce gli oggetti in torno a lui in modalità diverse in base alla distanza da essi, allo stesso modo utilizzando foto dello stesso oggetto ma da angolazioni diverse è possibile ottenere informazioni sulla struttura tridimensionale dell'oggetto stesso.



Distortion

La totalità delle telecamere commerciali devia dal modello della *pin-hole* camera e tale deviazione è generalmente tanto maggiore quanto grande è il campo visivo della camera: siccome ogni ottica è composta da un certo numero di lenti, la distorsione deriva dalle non idealità nella fase di produzione e di assemblaggio dell'ottica. Ottenere infatti una lente non distorcente è un processo estremamente costoso e soprattutto nelle applicazioni a basso costo dove bisogna fare affidamento a ottiche economiche risulta un problema molto evidente.

Queste non idealità generano una distorsione non lineare difficilmente modellizzabile e, anche per il fatto che tale distorsione dipende dall'interazione tra la lente e il sensore, i produttori di lenti normalmente non forniscono, o non riescono a fornire, informazioni geometriche su come rappresentare tale distorsione.

È importante osservare che il modello della *pin-hole* camera è valido solamente se l'immagine su cui si lavora è non distorta pertanto calibrare, ovvero correggere la distorsione geometrica, è un prerequisito per ricostruire in maniera accurata la tridimensionalità della scena osservata.

Dal punto di vista del raggio ottico, la distorsione introdotta dalla lente si pone tra il mondo e il *pin-hole*. L'equazione della camera pin-hole modificata con la distorsione dell'ottica si trasforma in

$$\mathbf{p} = \mathbf{K}f_d([\mathbf{Rt}]\mathbf{x})$$

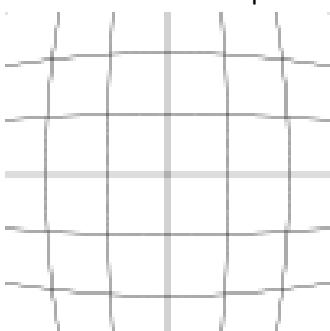
Con questo formalismo la distorsione f_d trasforma un punto da coordinate non-distorte in coordinate distorte. Questa scelta, rispetto alla formulazione inversa, viene da considerazioni puramente pratiche: siccome l'obiettivo è quello di avere un'immagine in uscita densa e non-distorta, è necessario calcolare quella funzione che trasforma appunto un punto non-distorto in un punto distorto.

In generale i contributi distorcenti della lente si dividono in:

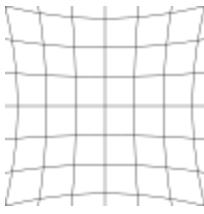
- **Radiali**, diretti lungo la direttrice che unisce il punto al centro di distorsione
- **Tangenziali**, che sono perpendicolari alla direttrice

I contributi di distorsione tangenziali (e altri contributi qui non citati) sono normalmente piccoli mentre la distorsione radiale è sempre rilevabile e, man mano che la distanza focale diventa corta, in generale aumenta di intensità

Le distorsioni radiali possono essere suddivise a loro volta in

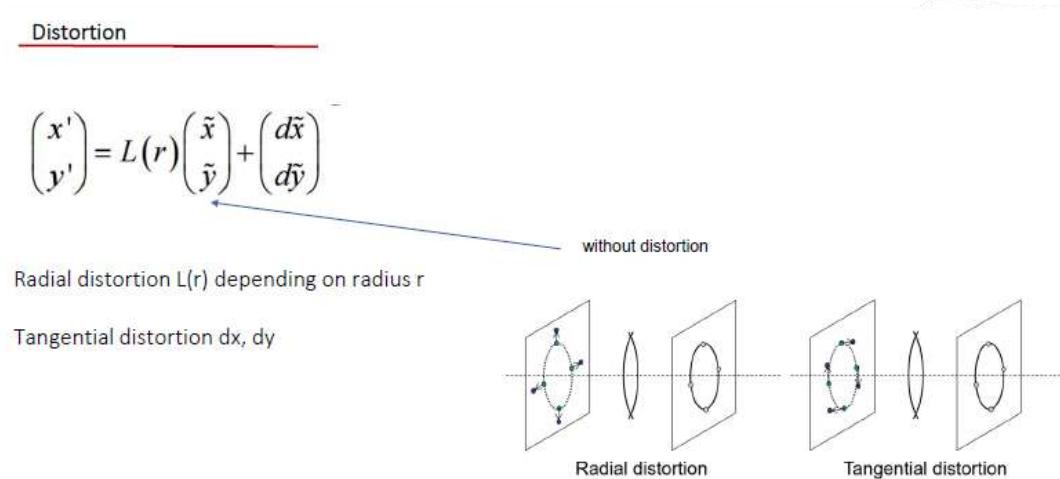


1. *Barrel distortion (distorsione a botte)*, di solito causata dalle lenti grandangolari. L'effetto della barrel distortion è quello di un'immagine mappata attorno ad una sfera (barile). Questo tipo di distorsione viene utilizzata per esempio negli obiettivi fisheye. Si verifica quando l'ingrandimento della lente diminuisce con l'aumentare della distanza dal punto principale. In presenza di questo effetto, linee rette verticali che si trovano vicino al bordo dell'immagine appariranno curve verso l'esterno.



2. *Pin Cushion distortion*, di solito causate dai teleobiettivi. Si verifica quando, al contrario, l'ingrandimento aumenta con l'aumentare della distanza dal punto principale quindi, in presenza di questo effetto, linee rette verticali che si trovano vicino al bordo dell'immagine appariranno curve verso l'interno.

Tangential distortion, che è come una rotazione dell'immagine, causata dal fatto che la lente non è uniforme, si verifica in direzione ortogonale rispetto alla direzione radiale.



Nell'equazione di sopra si definisce una relazione generale tra il punto ideale (x, y) e l'effettivo punto immagine distorto osservato. (\check{x}, \check{y})

In tutta l'immagine esiste un solo punto (x_d, y_d) , definito centro di distorsione, dove la distorsione non produce effetti. Per questo punto $(x, y) = (\check{x}, \check{y})$.

Per definire la distorsione è necessario operare in una nuova serie di coordinate, relative al centro di distorsione:

$$\begin{aligned} \bar{x} &= x - x_d \\ \bar{y} &= y - y_d \end{aligned}$$

Il centro di distorsione è normalmente vicino a $(0, 0)$ ma non c'è nessuna garanzia che coincida con il *principal point*. In diversi articoli viene infatti proposto, come approssimazione, ignorare il centro di distorsione e far coincidere il centro di distorsione con il *principal point* o considerare solamente il termine di *decentering distortion*.

La formulazione classica di *Brown-Conrady* modella la distorsione della lente sotto forma di scostamento:

$$\begin{aligned} \check{x} &= x + \delta_x(\bar{x}, \bar{y}) \\ \check{y} &= y + \delta_y(\bar{x}, \bar{y}) \end{aligned}$$

There is a non linear relation depending to the r distance to the center

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = L(r) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} + \begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix} \quad r = \sqrt{(\tilde{x} - \tilde{x}_c)^2 + (\tilde{y} - \tilde{y}_c)^2}$$

L approximation using Taylor $L(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots$

**Radial distortion
of a point x_c, y_c**

$$\begin{aligned} \hat{x}_c &= x_c(1 + \kappa_1 r_c^2 + \kappa_2 r_c^4) \\ \hat{y}_c &= y_c(1 + \kappa_1 r_c^2 + \kappa_2 r_c^4), \end{aligned}$$

Tangential distortion

$$\begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix} = \begin{pmatrix} 2p_1 \tilde{x}\tilde{y} + p_2(r^2 + 2\tilde{x}^2) \\ p_1(r^2 + 2\tilde{y}^2) + 2p_2 \tilde{x}\tilde{y} \end{pmatrix}$$

Tali scostamenti possono essere suddivisi per contributi:

radial distortion

Lo scostamento dovuto alla distorsione radiale ha equazione

$$\begin{aligned} \delta_x^r &= \bar{x} f_r(r) \\ \delta_y^r &= \bar{y} f_r(r) \end{aligned} \quad (7.10)$$

dove $f_r(r)$ è una funzione solo del raggio $r = \sqrt{\bar{x}^2 + \bar{y}^2}$, distanza euclidea tra il punto e il centro di distorsione, e con il vincolo $f_r(0) = 1$.

La funzione $f_r(r)$ della distorsione radiale non è un modello conosciuto ma può essere approssimata attraverso i primi termini dello sviluppo in serie:

$$f_r(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots \quad (7.11)$$

La presenza delle sole potenze multiple di 2 è dovuta alla simmetria della funzione f_r .

Alla fine con contributo radiale e tangenziale si va a definire la formula della distorsione di decentramento

decentering distortion

È normalmente causata dall'assemblaggio errato della lente e dei diversi componenti che compongono l'ottica. Il modello di *Brown-Conrad*y rappresenta il contributo di decentramento nella forma

$$\begin{aligned} \delta_x^{(t)} &= (p_1(r^2 + 2\bar{x}^2) + 2p_2 \bar{x}\bar{y})(1 + p_3 r^2 + \dots) \\ \delta_y^{(t)} &= (p_2(r^2 + 2\bar{y}^2) + 2p_1 \bar{x}\bar{y})(1 + p_3 r^2 + \dots) \end{aligned} \quad (7.13)$$

Questo contributo è costituito sia da una parte radiale che da una parte tangenziale.

Inserendo tutti questi contributi all'interno dell'equazione (7.9), il modello *Brown-Conrad*y complessivo si scrive come

$$\begin{aligned} \bar{x} &= x - x_d \\ \bar{y} &= y - y_d \\ r &= \sqrt{\bar{x}^2 + \bar{y}^2} \\ \bar{x} &= x + \bar{x}(k_1 r^2 + \dots) + (p_1(r^2 + 2\bar{x}^2) + 2p_2 \bar{x}\bar{y})(1 + p_3 r^2 + \dots) + s_1 r^2 + \dots \\ \bar{y} &= y + \bar{y}(k_1 r^2 + \dots) + (2p_1 \bar{x}\bar{y} + p_2(r^2 + 2\bar{y}^2))(1 + p_3 r^2 + \dots) + s_2 r^2 + \dots \end{aligned} \quad (7.14)$$

Di fatto la distorsione radiale è dominante e, in buona parte delle applicazioni, i primi termini sono più che sufficienti.

Per esempio OpenCV modella la distorsione con il modello *R3P1*: 3 termini radiali (k_1, k_2, k_3) e il termine di decentramento di primo grado (p_1, p_2).

Modeling distortion

Distortion-Free:

With radial Distortion

$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

$$1. \text{ Project } (X, Y, Z) \text{ to "normalized" image coordinates} \quad x_n = \frac{X}{Z}, \quad y_n = \frac{Y}{Z}$$

$$x_n = \frac{X}{Z}$$

$$y_n = \frac{Y}{Z}$$

$$r^2 = x_n^2 + y_n^2$$

$$x_d = x_n \left(1 + \kappa_1 r^2 + \kappa_2 r^4 \right)$$

$$y_d = y_n \left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)$$

3. Apply focal length
translate image center

$$x = fx_d + x_c$$

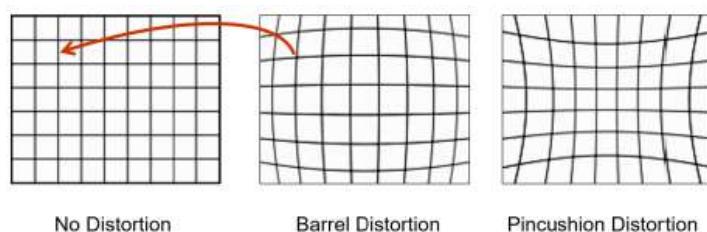
$$v = fv_d + v_c$$

To model lens distortion

To model lens distortion
Use above projection operation instead of standard projection matrix multiplication

k1,k2.. Radial distortion parameters

Con la distorsione si aggiunge al modello pinhole visto in precedenza, e quindi ai parametri intrinseci ed estrinseci, i parametri di distorsione radiale e tangenziale.



- Radial distance from Image Center:

$$r_u = r_d + k_1 r_d^3$$

Conoscendo r_d , è possibile trasformare il punto distorto ad uno senza distorsione, k cambia in base alla lente e una volta che si conosce il k è facile correggere la distorsione.

La prof consiglia di leggere alcuni paper sulla correzione della distorsione:

- R. Cucchiara, C. Grana, A. Prati, R. Vezzani «A HoughTransform-basedmethodfor RadialLens DistortionCorrection» ICIAP 2003

RADIAL COMPONENTS

$$k_1, k_2$$

TANGENT COMPONENTS

19-2

$$\begin{cases} x = x_0 + x_0(k_1r^2 + k_2r^4) + (2p_1x_0y_0 + p_2(r^2 + 2x_0^2)) \\ y = y_0 + y_0(k_1r^2 + k_2r^4) + (2p_2x_0y_0 + p_1(r^2 + 2y_0^2)) \end{cases}$$

THE 4 PARAMETERS CAN BE ESTIMATED OR COMPUTED WITH AN EXHAUSTIVE SEARCH.



Simple correction

given $pd=(x_d, y_d)$ the distorted pixel, the undistorted pixel is $pu=(x_u, y_u)$ a simplified method with cx, cy center of distortion

$$x_u = c_x + (x_d - c_x)(1 + k_1 r_d^2 + k_2 r_d^4 + \dots)$$

$$y_u = c_y + (y_d - c_y)(1 + k_1 r_d^2 + k_2 r_d^4 + \dots)$$

It corrects the 90%

$$x_u = x_d + (x_d - c_x)(k_1 r_d^2)$$

$$y_u = y_d + (y_d - c_y)(k_1 r_d^2)$$

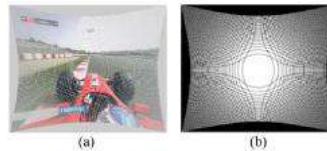


Fig. 1. Example of incomplete mapping on the undistorted image. (b) shows the locus of points for which the coordinates are defined.

And using some approximations r_u computed from the center distance of (x_u, y_u)

$$x_d = c_x + (x_u - c_x) \frac{r_d}{r_u}$$

$$y_d = c_y + (y_u - c_y) \frac{r_d}{r_u}$$

How to find them? An exhaustive simple search

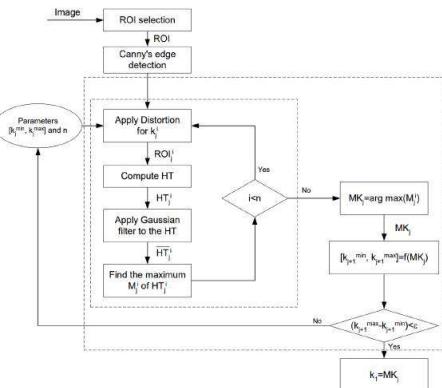
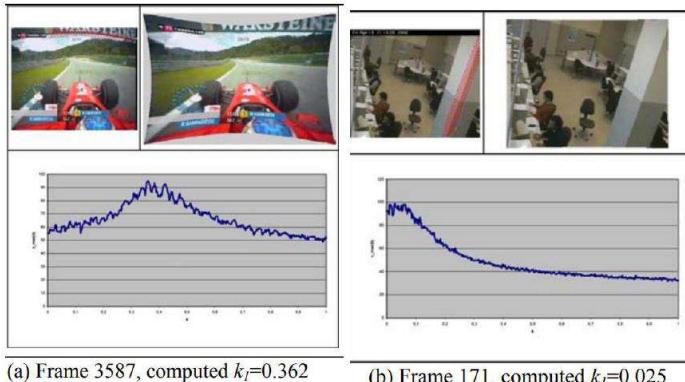


Fig. 3. Data flow of the HTRDC proposed system.

https://www.researchgate.net/publication/221355669_A_Hough_transform-based_method_for_radial_lens_distortion_correction

- Brito radial distortion, self calibration 2013, utilizzo di linee epipolari
http://openaccess.thecvf.com/content_cvpr_2013/papers/Brito_Radial_Distortion_Self-Calibration_2013_CVPR_paper.pdf

Radial Distortion Self-Calibration

José Henrique Brito
 IPCA, Barcelos, Portugal
 Centro Algoritmi, Universidade do Minho, Portugal
 josehbrito@gmail.com

Kevin Kösler
 ETH Zürich, Switzerland
 kevin.koesler@inf.ethz.ch

Roland Angst*
 Stanford University
 ETH Zürich, Switzerland
 rangst@stanford.edu

Marc Pollefeys
 ETH Zürich, Switzerland
 marc.pollefeys@inf.ethz.ch



Figure 6. Two rows of three images taken with different cameras (with different amounts of radial distortion). For the center image the estimated radial distortion center is visualized by showing the two straight epipolar lines. As compared to a chessboard-based calibration by the method of [13] the distortion center is 2.5% away (top example) and 6.5% away (bottom example) from the ground truth position (fraction given with respect to image size).

- [Bogdan et al DeepCalib: a deep learning approach for automatic intrinsic calibration of wide field-of-view camera SIGGRAPH 2018](#)

Salta slide spherical model a lezione ha detto che non lo chiede, si tratta di un'alternativa del pinhole model.

Leggere le slide da 107 a 110, le attacco qui sotto.

Estimation with CNN



It aims to estimate the focal length f and the distortion parameter ξ .

It creates an enormous dataset of fisheye pictures from web

Propose architectures

train them.

Dataset

This dataset contains about 67,000 high-resolution 9104×4552 px panoramas acquired in various scenes, such as indoor/outdoor, urban/natural and bright/dark.



Figure 2: Given an input panorama, we automatically generate images with different focal lengths f and distortion values ξ , via the unified spherical model [Barreto 2006; Mei and Rives 2007]².

Reimplement the method

Adopt for feature Inception-V3

The paper proposes 3 architectures in both the classification and regression.

For the classification problem: softmax as the activation function for the output layer(s) and cross entropy for the loss function.

For the regression problem: the sigmoid activation in the output layer(s) and the logcosh loss.

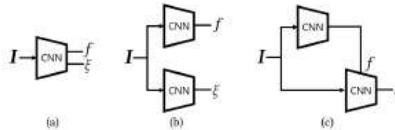


Figure 3: Illustration of the three network architectures: SingleNet (a), DualNet (b) and SeqNet (c). The input is the image I to calibrate, and the outputs are the focal length f and the distortion parameter ξ . In SeqNet, the value of the focal length f estimated by the first network is concatenated into one of the dense layers of the second network.

Parameters

Dataset: millions of images with a resolution of 299×299 px from all the panoramic images of the SUN360

For classification training:

focal lengths on a range between 50 and 500px with a step size of 10,
 distortion values on a range between 0 and 1.2 with a step size of 0.02 (discrete dataset).

For regression training:

randomly sampled the values of focal length and distortion parameter on the same ranges

dataset split into three subsets: 80% for training, 10% for testing, and 10% for validation.

(Each original panorama is used exclusively for training, or testing, or validation, i.e., none of the original panoramas belongs to more than one dataset. =

data augmentation by randomly adding Gaussian noise, modifying the brightness and contrast, and image mirroring. Inception V3 is pretrained on the ImageNet dataset, and we further train them on our generated dataset with early stop strategy to prevent overfitting learning rate to 10⁻⁵ and used a batch size of 64 for both classification and regression.

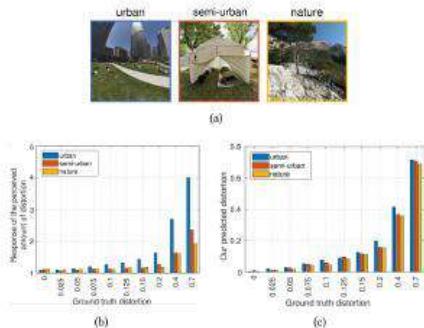


Figure 6: Analysis of the distortion on different scene types.
 (a) Representative images of the three scene types: urban, semi-urban and nature. (b) User study results on the amount of perceived distortion with respect to the ground truth distortion parameter value. (c) Comparison of our estimated distortion parameter values vs. ground truth.

calibration methods. All the methods use multiple checkerboard pictures, while ours require a single picture of a general scene.

Method	f	distortion	mean error (px)
Avenir 2.8mm			
Ours	954	0.77	3.40
Mei	1973	1.48	0.12
OpenCV Fisheye	N/A	N/A	N/A
OpenCV Brown	796	-0.30, 0.17, -0.00, -0.00, -0.07	0.20
Scaramuzza	788	-788.50, 0, 3.55 ⁻⁴ , 2.11 ⁻⁷ , -2.62 ⁻¹⁰	1.01
Avenir 4mm			
Ours	1189	0.47	2.10
Mei	2270	0.97	0.13
OpenCV Fisheye	N/A	N/A	N/A
OpenCV Brown	1158	-0.27, 0.28, 0.00, 0.00, -0.21	0.29
Scaramuzza	1157	-1157.40, 0.00, 2.89 ⁻¹ , -2.45 ⁻⁷ , 1.79 ⁻¹⁰	0.54
GoPro			
Ours	1182	0.82	1.11
Mei	1561	1.28	0.12
OpenCV Fisheye	787	0.15, -0.77, 1.61, -0.98	0.7
OpenCV Brown	N/A	N/A	N/A
Scaramuzza	791	-791.4, 0.00, 0.00, -1.12 ⁻⁴ , 3.10 ⁻¹⁰	0.82
Fisheye			
Ours	777	1.01	1.27
Mei	1351	1.79	0.10
OpenCV Fisheye	488	-0.89, 0.65, -1.79, 1.13	1.05
OpenCV Brown	N/A	N/A	N/A
Scaramuzza	487	-487.7, 0, 8.18 ⁻¹ , -4.39 ⁻⁷ , 4.32 ⁻¹⁰	1.48

Vanishing point

I vanishing point o punti all'infinito, sono i punti in cui in una proiezione prospettica le rette convergono.

We do not know the 3D to 2D matrix projection

$$\mathbf{P} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{H}\mathbf{P}$$

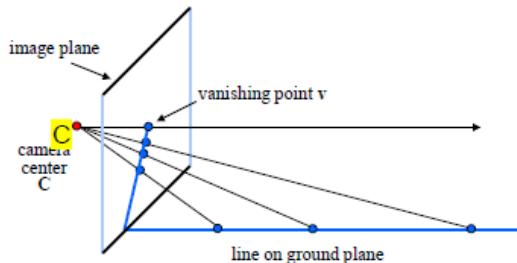
Any two parallel lines have the same vanishing point v

The ray from C through v is parallel to the lines

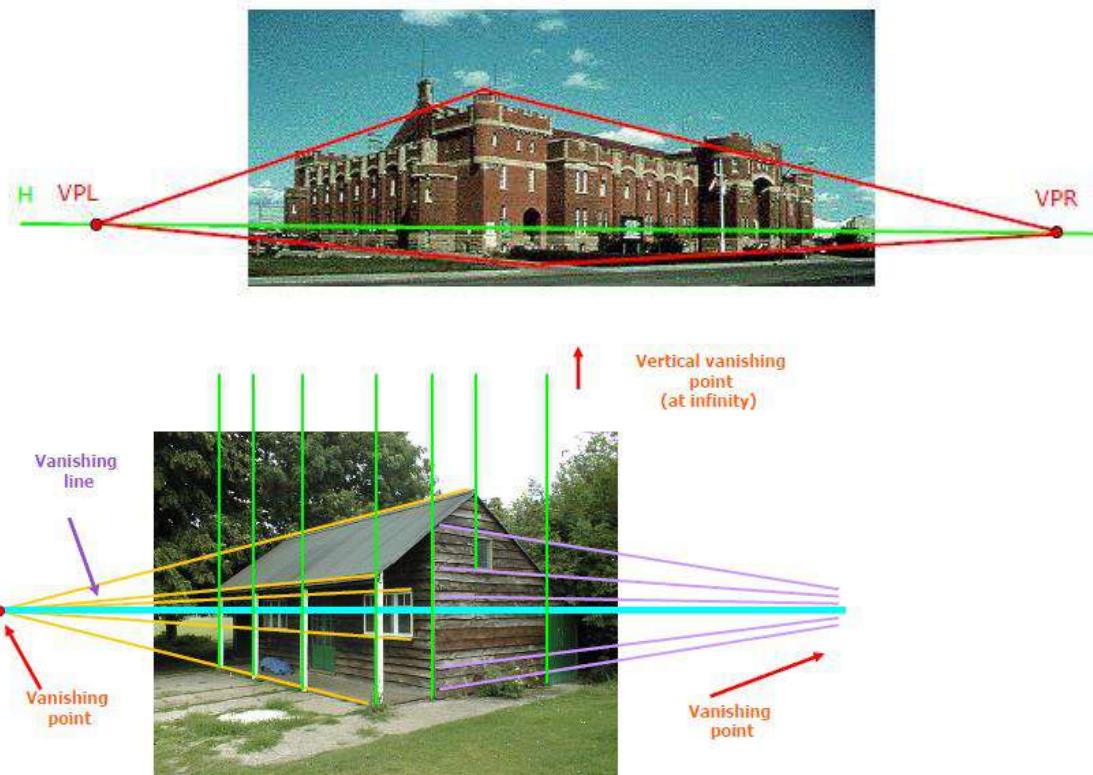
When??

An image may have more than one vanishing point
in fact every pixel is a potential vanishing point

How can we compute the vanishing point?



Le immagini possono avere dei punti all'infinito anche al di fuori dell'immagine stessa, infatti i punti all'infinito di un'immagine sono 3, rispettivamente per gli assi x, y e z.

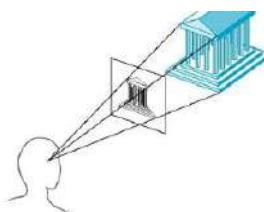


Ogni insieme di rette parallele su un piano definisce un punto all'infinito. L'unione di tutti i punti all'infinito sullo stesso piano è detta vanishing line/retta proiettiva, che nell'immagine corrisponde all'orizzonte.

Projectors converge at center of projection
One principal face parallel to projection plane
One vanishing point for cube



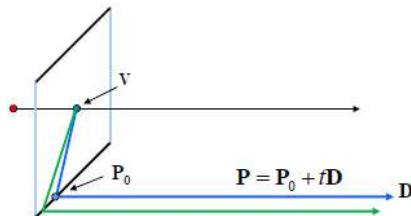
On principal direction parallel to projection plane
Two vanishing points for cube



Nella prima immagine da sinistra,
si proietta un solo punto
all'infinito, nella seconda 2 e nella
terza 3.

Queste slides le ha saltate nella spiegazione, le aggiungo per completezza:

Computing vanishing points



$$\mathbf{P}_t = \begin{bmatrix} P_x + tD_x \\ P_y + tD_y \\ P_z + tD_z \\ 1 \end{bmatrix} \cong \begin{bmatrix} P_x / t + D_x \\ P_y / t + D_y \\ P_z / t + D_z \\ 1/t \end{bmatrix} \quad t \rightarrow \infty \quad \mathbf{P}_{\infty} \cong \begin{bmatrix} D_x \\ D_y \\ D_z \\ 0 \end{bmatrix}$$

Properties

\mathbf{P}_{∞} is a point at *infinity*, \mathbf{v} is its projection

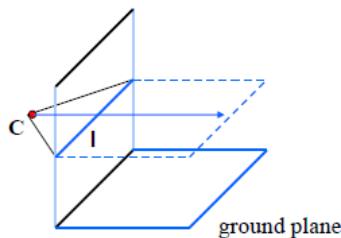
They depend only on line *direction*

Parallel lines $\mathbf{P}_0 + tD$, $\mathbf{P}_1 + tD$ intersect at \mathbf{P}_{∞}

See algorithm:

<http://www-2.cs.cmu.edu/~ph/869/www/notes/vanishing.txt>

Computing the horizon



Properties:

I is intersection of horizontal plane through C with image plane

Compute I from two sets of parallel lines on ground plane

All points at same height as C project to I

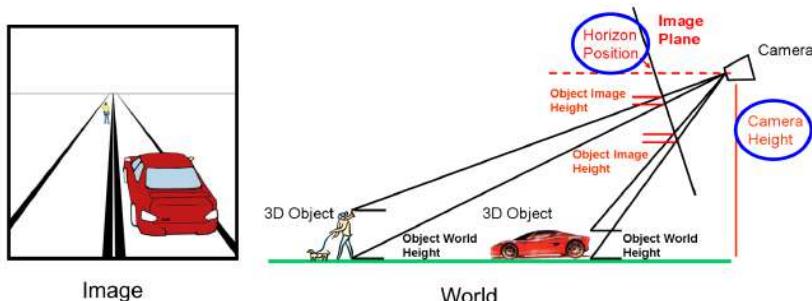
points higher than C project above I

Provides way of comparing height of objects in the scene

La teoria proiettiva trova una giusta applicazione nel calcolo della distanza per esempio nei sensori delle automobili:

Computing distance

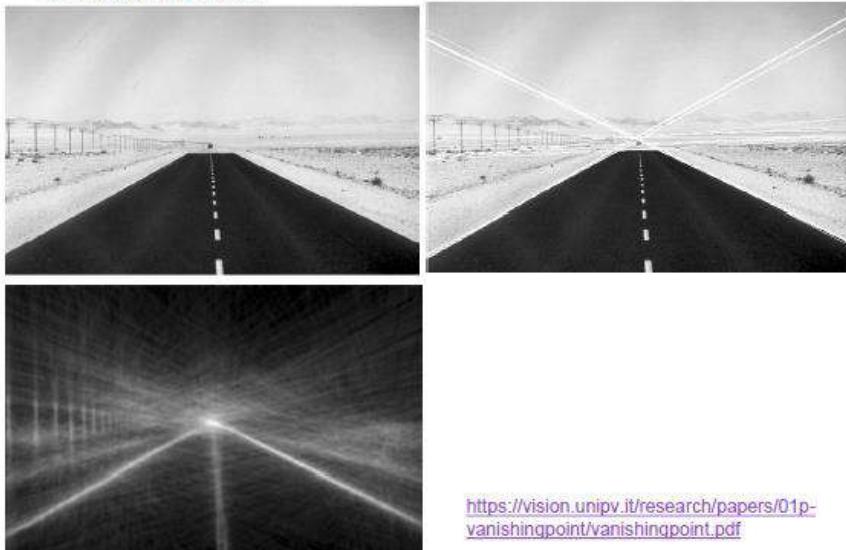
To compute the distance in the real 3D world we must know the perspective transformation
We need to know the distance between image plane and camera center . i.e. focal length.



È possibile calcolare i vanishing point usando Hough Transform

Compute vanishing points

Use Hough transform

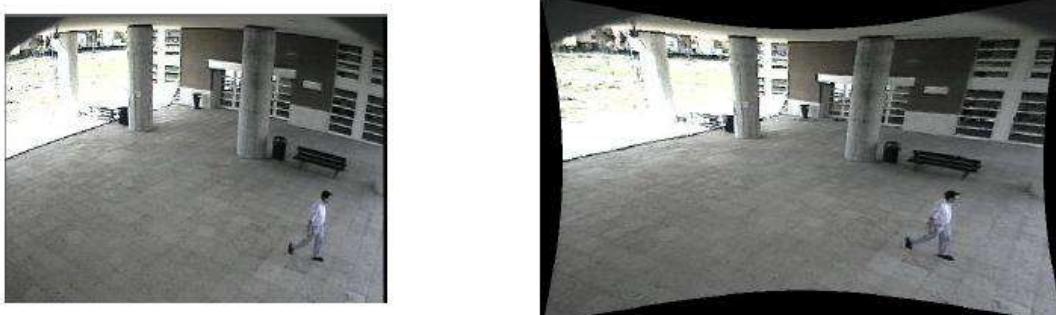


<https://vision.unipv.it/research/papers/01p-vanishingpoint/vanishingpoint.pdf>

AlImageLab ha ricercato nel campo dei vanishing point. L'esempio di applicazione oggetto di ricerca è il seguente.

Come comprendere l'altezza di una persona:

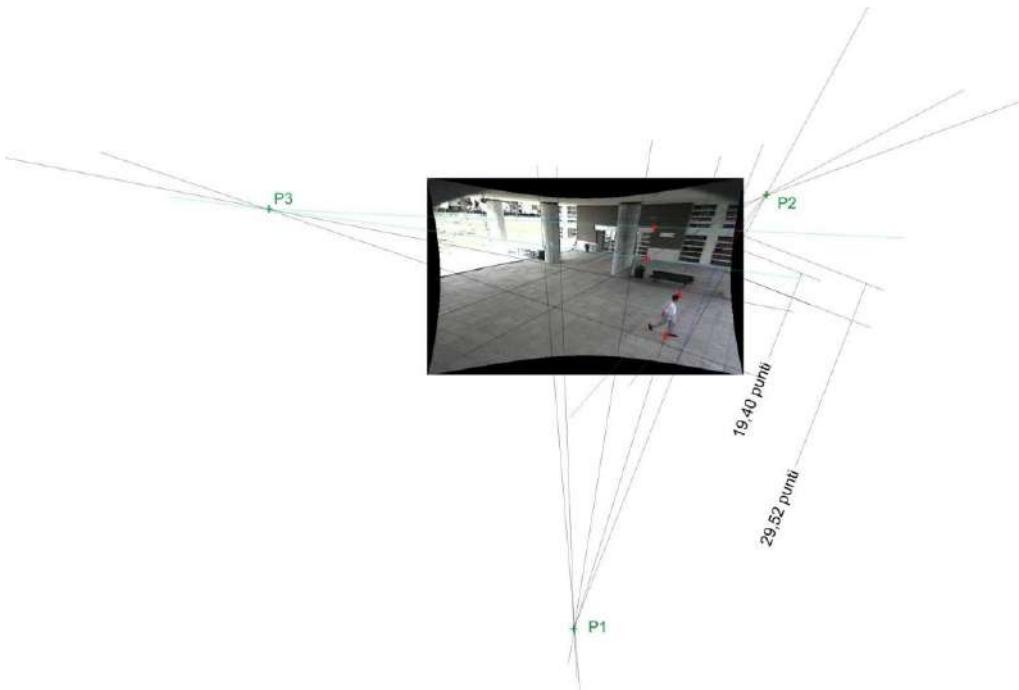
1. Correzione della distorsione della lente



2. Si seleziona 4 punti che sono sull'immagine, in corrispondenza del terreno



3. Estrarre i punti all'infinito e quindi poter calcolare le corrispondenze di lunghezza



Ratio between length

P Person's length: 19,4 points

D door's length: 29,52

DCM Door's length in cm: 270

PCM Person length in cm: ?

PCM:DCM=P:D

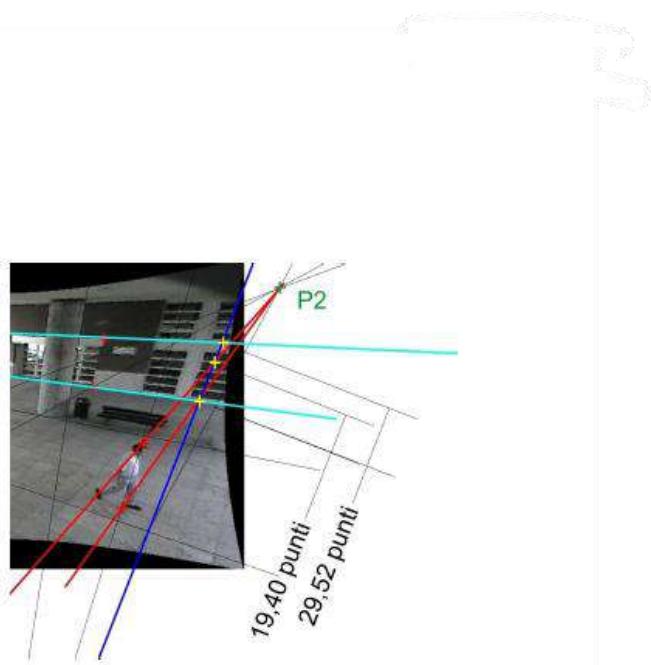
PCM= P * DCM/D

→ cm: 177,4390244

this without an explicit calibration

the alternative is to have the matrix
of projective transformation

And the calibration



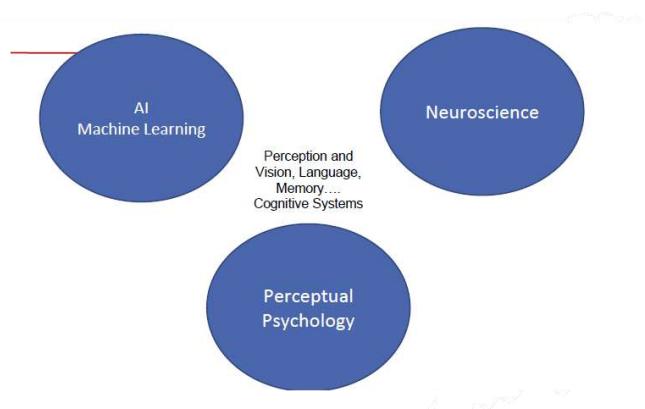
Per saperne di più leggere il paper (non lo chiede)

- <http://pages.cs.wisc.edu/~dyer/ai-qual/criminisi-dagm02.pdf>
- <http://www.cs.cmu.edu/~ph/869/papers/Criminisi99.pdf>

Il link sottostante conduce ad un paper in cui viene usata la teoria dei vanishing point per la road detection con stereocamera. (La prof ha fatto riferimento a lezione dell'uso di vanishing point per stereo vision).

<https://hal.archives-ouvertes.fr/hal-01678548/document>

9. Computer Vision and Neuroscience – Segmentation



Quello che abbiamo studiato e visto finora sulla computer vision è strettamente legato ai processi cognitivi che avvengono nella mente umana. Sia quello percettivo, legato ad un fattore psicologico, sia quello neurologico, che invece fa riferimento al funzionamento della corteccia cerebrale e dei neuroni.

La computer vision e i visual descriptors partono dalla conoscenza di questi due processi umani fisiologici.

Now : The neuroscience aims at understanding how information sensed by eyes become vision, how thoughts become memory, how behavior comes from biology

[E. Kandel, The age of the unconscious 2012]

Neuroscience results are young as the Deep Learning results are.

We have NOW a lot to learn (probably less in the past)

Our Brain is one example of a complete working (well) and evolved in million years system. Probably it is not the only one we could create...

Un altro aspetto importante della mente umana da considerare è quello della memoria, la memoria si divide in memoria a breve e a lungo termine. La **memoria a breve termine** (MBT) contiene le informazioni per un periodo di tempo molto breve, solitamente il tempo stimato corrisponde a una decina di secondi. Dopo questo tempo, la traccia decade. La **memoria a lungo termine** (MLT) è un archivio avente capacità quasi illimitata, dove sono conservate tutte le esperienze e le conoscenze acquisite nel corso della vita e quelle che corrispondono al nostro carattere o temperamento.

La memoria nel machine learning viene incorporata nella parte di setting dei parametri durante il training.

Un esempio di memoria a lungo termine nell'intelligenza artificiale è dato da memorie in grado di selezionare architetture e iperparametri. O anche da Long and Short Term Memory in RNN.

(parentesi su reti neurali ricorrenti)

Una **rete neurale ricorrente** è una rete in cui alcuni neuroni sono collegati in loop tra loro. Tipicamente i valori di uscita di uno strato di un livello superiore sono utilizzati in ingresso di uno strato di livello inferiore. Quest'interconnessione tra strati permette l'utilizzo di uno degli strati come memoria di stato, e consente, in ingresso di una sequenza temporale di valori, modellarne un comportamento dinamico temporale dalle informazioni ricevute agli istante di tempo precedente. In altri casi lo strato è costituito da un insieme di neuroni dotato di connessioni molto sparse che innesca una caotica, impiegata per l'addestramento di una parte successiva della rete, come avviene per le echo state network. Ciò le rende applicabilità a compiti di analisi predittiva su sequenze di dati, quali possono essere ad esempio il riconoscimento della grafia o il riconoscimento vocale.

A mostly complete chart of
Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

Perceptron (P)



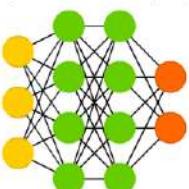
Feed Forward (FF)



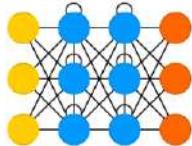
Radial Basis Network (RBF)



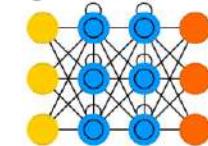
Deep Feed Forward (DFF)



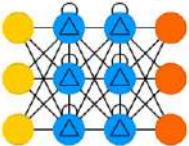
Recurrent Neural Network (RNN)



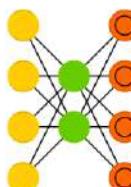
Long / Short Term Memory (LSTM)



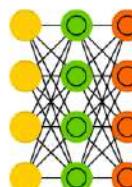
Gated Recurrent Unit (GRU)



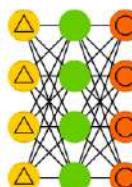
Auto Encoder (AE)



Variational AE (VAE)



Denoising AE (DAE)



Sparse AE (SAE)



Markov Chain (MC)



Hopfield Network (HN)



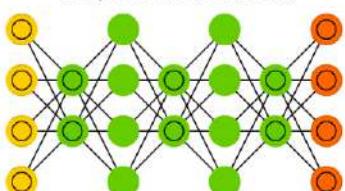
Boltzmann Machine (BM)



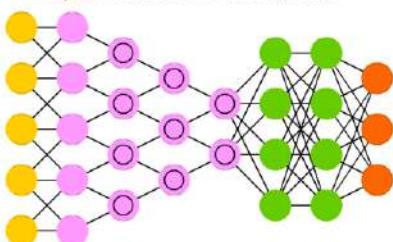
Restricted BM (RBM)



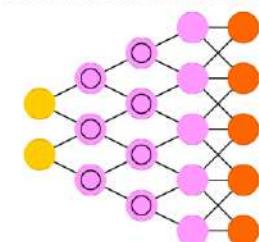
Deep Belief Network (DBN)



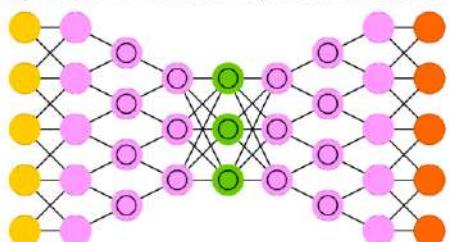
Deep Convolutional Network (DCN)



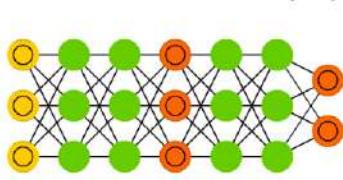
Deconvolutional Network (DN)



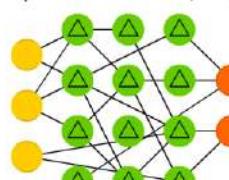
Deep Convolutional Inverse Graphics Network (DCIGN)



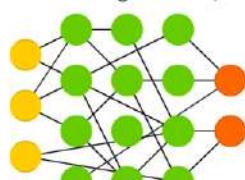
Generative Adversarial Network (GAN)



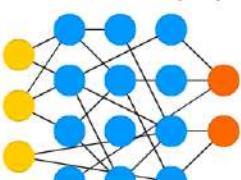
Liquid State Machine (LSM)



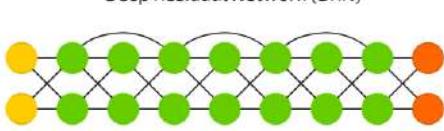
Extreme Learning Machine (ELM)



Echo State Network (ESN)



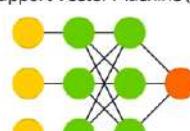
Deep Residual Network (DRN)



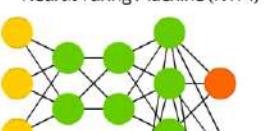
Kohonen Network (KN)



Support Vector Machine (SVM)



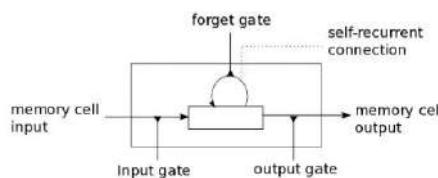
Neural Turing Machine (NTM)



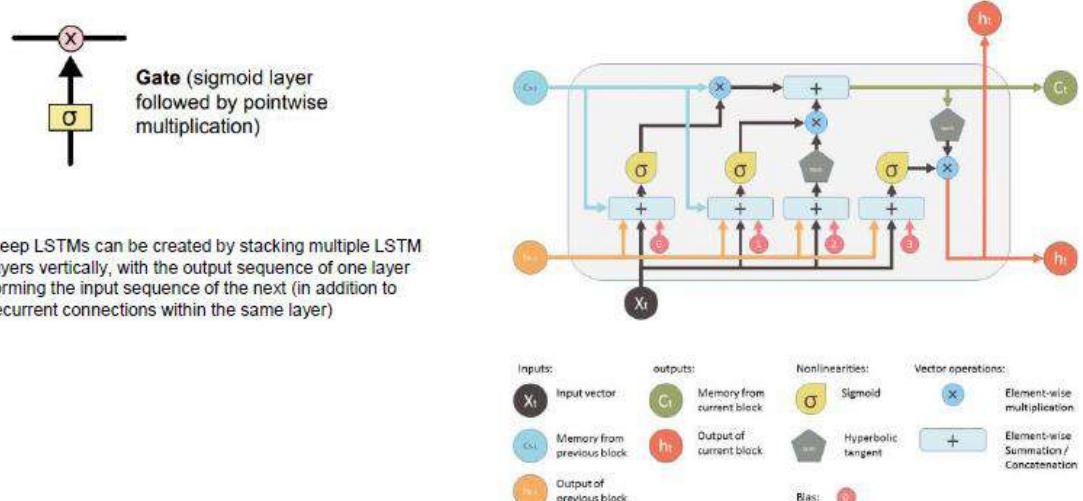
Long short-term memory

Si tratta di una rete neurale ricorrente che differentemente dalle standard feedforward reti neurali presenta delle, cosidette, connessioni di feedback. LSTM presenta un'architettura in grado di elaborare sia singoli dati che una sequenza, come ad esempio un video completo. Può essere utilizzata per il riconoscimento vocale e il riconoscimento della calligrafia.

Il processo di trasferimento dei dati è lo stesso delle reti neurali ricorrenti standard. Tuttavia, l'operazione di propagazione delle informazioni è diversa. Durante il propagarsi dei dati nella rete, si effettua un'operazione di decisione in merito a quale informazioni/dati mantenere e quali invece scartare. La decisione sulle informazioni da mantenere viene effettuata all'interno di cells e i responsabili sono i gates. LSTM presenta appunto dei gates, che si aprono e si chiudono in base al flusso di informazioni che scorre nella rete. Solo i dati rilevanti vengono selezionati in sequenza per effettuare una previsione efficace.

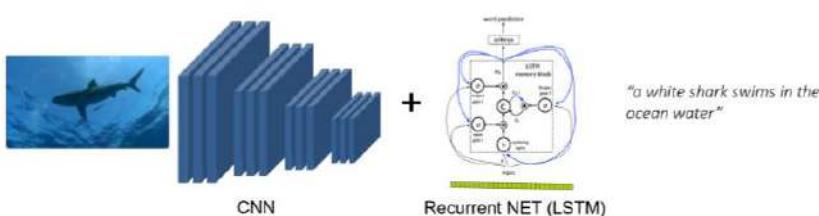


LSTM Memory Cell



LSTM non è più tanto utilizzata come in passato, perché non si applica in maniera performante con architetture parallele.

Ma presentiamo un esempio che permette di ricreare un processo cognitivo della mente umana.



Base technical idea:

- Use CNN as feature extractor and RNNs as language models

Daniel Kahneman, è uno psicologo, vincitore del premio Nobel 2002, ha studiato i meccanismi della memoria nella psicologia. Le sue considerazioni sono raccolte nel libro “Thinking, fast and slow”. Kahneman ha individuato due sistemi di memorizzazione nella mente umana:

- **Sistema 1**
 - È veloce
 - Si definiscono le caratteristiche di quello che viene memorizzato, a livello di inconscio e in maniera automatica
 - Senza avere il controllo della memorizzazione e a volte nemmeno la coscienza di stare memorizzando
 - Il ruolo di questo sistema è valutare la situazione
 - Costituisce il 98% del nostro pensiero
- **Sistema 2**
 - È lento
 - Si definiscono tutte le caratteristiche, con coscienza, si tratta di un processo mentale controllato e razionale
 - Coscienti di stare memorizzando
 - Cerca di trovare nuove o mancanti informazioni, effettua decisione
 - Costituisce il 2% del nostro pensiero

Nel contesto dei sistemi cognitivi si sono individuate le sfide che deve superare il Deep Learning nel futuro.

All three Turing Award winners agree on Deep Learning's problems

Yann LeCun presented the top 3 challenges to Deep Learning:

1. Learning with fewer labeled samples
2. Learning to reason
3. Learning to plan complex action sequences

Geoff Hinton — "It's about the problems with CNNs and why they're rubbish":

1. CNNs not good at dealing with rotation or scaling
2. CNNs do not understand images in terms of objects and their parts
3. CNNs are brittle against adversarial examples

Yoshua Bengio — "Neural Networks need to develop consciousness":

1. Should generalize faster from fewer examples
2. Learn better models from the world, like common sense
3. Get better at "System 2" thinking (slower, methodological thinking as opposed to fast recognition)

Per maggiori informazioni su le innovazioni effettuate da Hinton con le Capsule Networks

<https://ilgeneralecluster.it/neural%20networks/theory/2019/01/29/capsule-networks.html>

Eric Kendal ha vinto il premio Nobel nel 2000 in psicologia e in medicina per le sue ricerche sulla psicologia alla base dello spazio di memorizzazione nei neuroni. Le sue sperimentazioni sono state effettuate sulla Aplysia, una lumaca di mare con 20K cellule nervose, molto simili alla mente umana ma di numero decisamente inferiore. Kandel voleva studiare cosa succede a livello molecolare e nervoso nel momento in cui si crea una short-term o long-term memory.

Kendal ha individuato due tipi di memoria:

Il primo più complesso, richiede l'uso dell'ippocampo e viene detta memoria esplicita. Mentre la seconda più semplice, che permette di effettuare operazioni in maniera automatica, come dopo che si è imparato a guidare, è detta memoria implicita.

Kendal prosegue i suoi studi facendo capo al *riduzionismo*, si tratta di un pensiero scientifico del secolo scorso secondo il quale tutta la realtà fisica possa essere in definitiva "ridotta" (e spiegata) in termini di particelle materiali e dei loro movimenti. Quindi i fenomeni psicologici e mentali, possono essere ricondotti e spiegati con la neurofisiologia.

Citando Kendal:

"You've got to see how a memory is formed. You've got to see how information comes into the hippocampus and how it is stored over the long term. We found it very complicated to analyze. Rather than studying the most complex form of memory in a very complicated animal, we had to take the most simple form --an implicit form of memory --in a very simple animal. And I focused in on the marine snail Aplysia. My colleagues and I found that learning involves alterations in the strength of communication between nerve cells in the synapses. And these synapses are plastic --they can be modified by learning. If you produce a short-term memory --if you look up a telephone number you just remember for a short period of time you have a transient change in the strength of communication. But if you have a long-term memory, you alter the expression of genes in the brain and you grow new synaptic connections. So as I tell my friends, if you remember anything about this conversation, you will have a different brain than you started out with before the conversation."

La mente umana è molto più vasta di quella dell'animale su cui Kendal ha fatto i suoi esperimenti, inoltre presenta una parte inconscia che ancora è quasi sconosciuta. Ha senso pensare che la mente umana si adatta a dei meccanismi di pura evoluzione e conservazioni come poi ha scoperto Darwin, ma quello che accade realmente a livello fisiologico ancora non si conosce.

Una parte della fisiologia neuronale però parla di plasticità sinaptica. La **plasticità sinaptica**, particolare tipo di plasticità neuronale (insieme alla plasticità intrinseca relativa alle proprietà elettriche intrinseche dei neuroni, alla plasticità strutturale relativa alla loro struttura assonale o dendritica e alla neurogenesi capace di introdurre nuovi elementi neuronali in un circuito), è la capacità del sistema nervoso di modificare l'intensità delle relazioni interneuronali (sinapsi), di instaurarne di nuove e di eliminarne alcune. Questa proprietà permette al sistema nervoso di modificare la sua struttura e la sua funzionalità in modo più o meno duraturo e dipendente dagli eventi che li influenzano come ad esempio l'esperienza.

Considerazioni finali:

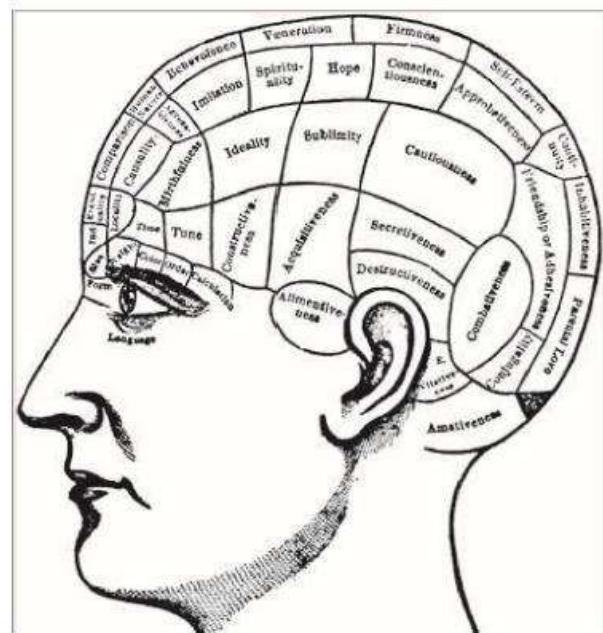
- Comprendere come funziona la mente nei processi di apprendimento è importante per lo sviluppo e l'evoluzione di intelligenza artificiale
- Ancora oggi non si conosce il funzionamento della nostra mente al 100%
- Lavorare con architetture semplici è importante per comprendere il modello
- La sinapsi ha un ruolo importante nell'apprendimento, così anche la plasticità deve essere studiata più approfonditamente.

To know something in addition



Simple neuroscience and art

- The age of Unconsciousness (E.Kendal also in Italian)
 - Art and neuroscience (E. Kendal also in Italian)
 - Principle of Neuroscience (E. Kendal et al 4 ed)
 - -----
 - Neuroscience stated in 1800 as Frenology
 - In 1800 Frenology a map of 35 cognitive and emotional areas
 - Let's consider what is related with VISION

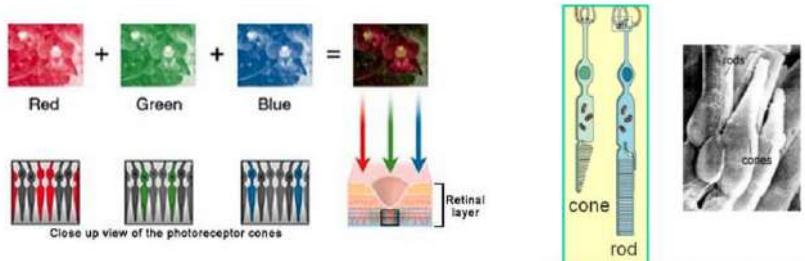


Confronto hardware: (scrivo quello che ha detto la prof ma non so se ha molto senso, non essendo lei un oculista)

- Occhi, sono due sensori
 - Cornea, sono le lenti con un indice di rifrazione pari a 1.33, biconvessa lente, con distanza focale variabile (siamo noi che la controlliamo)
 - Pupilla, è il diaframma della nostra vista
 - Coni e bastoncelli, sono fotorecettori in grado di convertire impulsi e elettrici in luce (sono circa da 6 a 120 milioni, 11 bit di risoluzione, 4 bit di risoluzione spettrale, ogni occhio ha un campo visivo che copre più o meno 150° → in questa porzione dell'occhio avviene un prima processione dei dati)
 - Retina, ci presenta una prima idea del piano visivo, la retina è la parte più connessa ai neuroni, quindi image processing = retina processing
 - Nervo ottico, nel nervo ottico avviene la connessione, si crea una compressione dei dati, si trasmettono i dati a 10fps (100ms)
 - Corteccia e mente, sono gli attuatori della visione

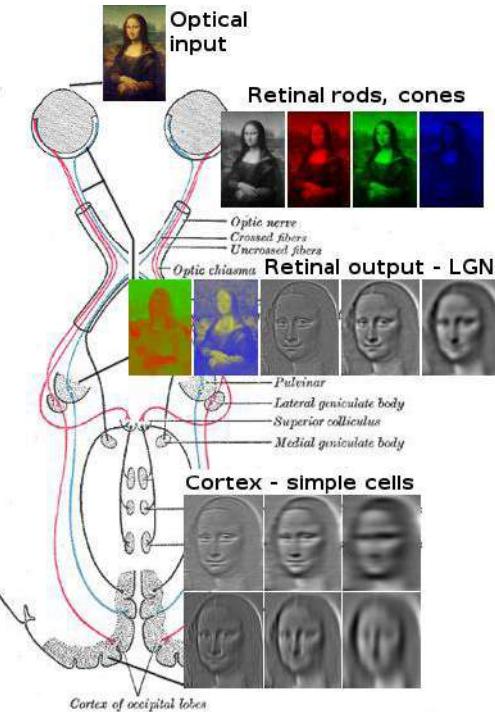
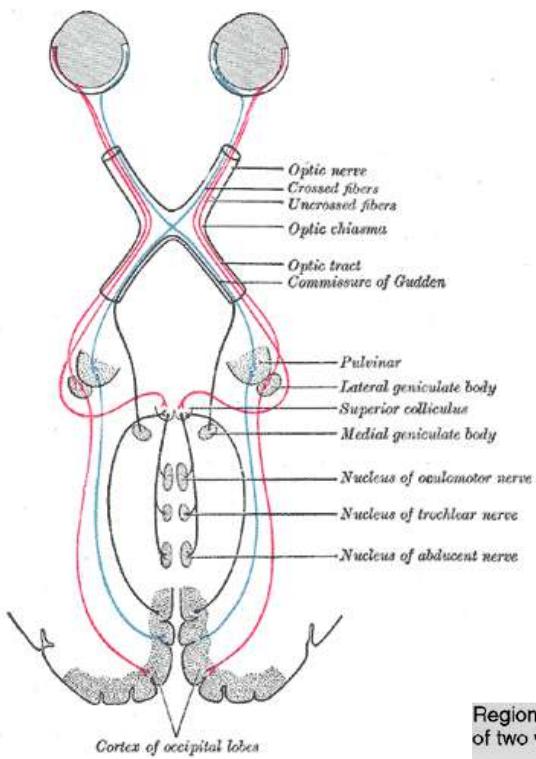
L'acquisizione dell'immagine avviene a livello dei coni e dei bastoncelli, i particolare:

- I coni sono meno sensibili alla luce, ma sono sensibili ai colori, il modello RGB deriva proprio dalla percezione dei colori a livello dei coni

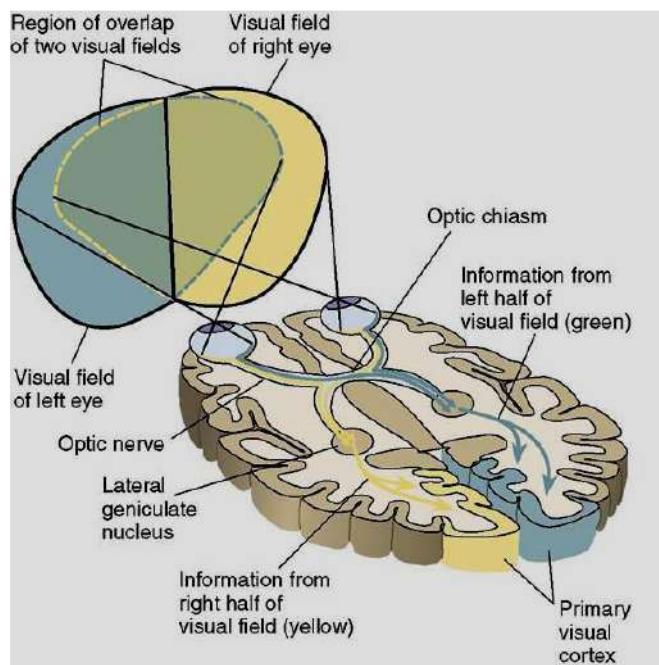


- I bastoncelli sono più sensibili alla luce, ci permettono di vedere anche di notte, contribuiscono ad una visione a livello grayscale

Il modello di percezione visiva nell'immagine affianco e sotto, è un modello antecedente a diverse scoperte che si sono fatte negli anni, risale al 1918. Si è supposto che ci fossero due percezioni di immagini, una a livello della corteccia e una a livello della retina. Ora, invece, si conosce che l'intera percezione avviene a livello di retina, e poi questa viene processata a livello della corteccia.



- Preprocessing** a livello di retina
- Trasmissione** tramite il nero ottico
- Calcolo computazionale** avviene nella corteccia (image analysis) e nella mente



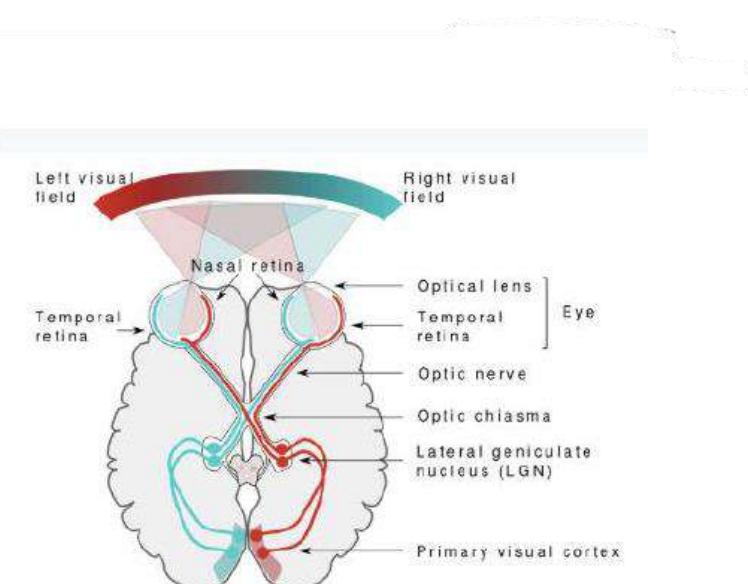
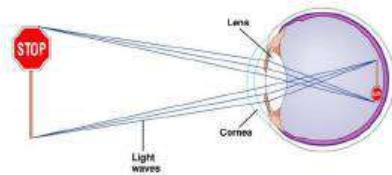
IN MODO PIU' CHIARO → Il modo in cui vediamo le cose è il risultato di un processo complesso: ancora prima di arrivare a vedere qualcosa, una serie di passaggi singoli ha luogo nell'occhio e nel cervello. Stiamo parlando del percorso retino-corticale, che inizia sull'occhio e si sviluppa fino al nostro cervello. In parole semplici, noi vediamo perché l'occhio umano assorbe la luce dall'ambiente circostante e la raccoglie sulla cornea, generando una prima impressione visiva. Entrambi gli occhi trasmettono quindi questa immagine al cervello attraverso il nervo ottico. Lì l'immagine viene elaborata, dando luogo a ciò che noi chiamiamo "visione". La luce forma la base di tutto ciò che vediamo. Nella più totale oscurità, siamo virtualmente ciechi.

Nello specifico, ciò significa che per poter vedere un oggetto, questo deve essere necessariamente esposto a una qualche forma di luce. La luce viene riflessa dall'oggetto ed elaborata dal nostro apparato visivo. Se guardiamo un albero, i nostri occhi assorbono la luce riflessa dall'albero: i raggi innanzitutto penetrano la congiuntiva e la cornea. Successivamente, attraversano la camera anteriore e la pupilla. Infine la luce colpisce il cristallino, dove viene raccolta e trasferita alla retina fotosensibile. Qui l'informazione visiva viene acquisita e smistata: i bastoncelli permettono di distinguere tra giorno e notte, mentre i coni hanno la funzione di farci vedere i colori e i contrasti. L'informazione elaborata viene trasferita al nervo ottico, che la trasmette direttamente al cervello, dove verrà esaminata ancora una volta, interpretata e consolidata per formare l'immagine che infine vediamo.

Nonostante le scoperte approfondite sull'anatomia dell'occhio umano e la sua struttura, molte domande sul funzionamento della nostra coscienza rimangono senza risposta. Pertanto, mentre sappiamo quali parti del cervello sono maggiormente coinvolte quando vediamo qualcosa, nessuno ha ancora capito come questo dia vita al mondo così come lo percepiamo.

Data transmission now

1. Acquisition by two sensors
2. Pre-processing in the retina
3. Compression through the optical nerve
4. Data are collected in the LGN of thalamus
5. Go to the primary visual cortex



Some pre-processing in smart-camera

I recenti studi sulla visione hanno fatto emergere come a livello del LGN avviene un "incontro" tra l'immagine che vediamo e la nostra memoria e la nostra esperienza.

ABC della mente umana nel campo della visione

- Il **mesencefalo (midbrain)**, controlla numerosi sensori e funzioni motori, inclusi il movimento degli occhi e le coordinate visive e i riflessi uditivi.
- Il **desencefalo (midbrain)** contiene due strutture:

- Il talamo, processa la maggior parte delle informazioni che raggiungono la corteccia cerebrale dal sistema nervoso centrale, contiene LGN;
- L'ipotalamo, regola le funzioni automatica, endocrine e viscerali
- L'**emisfero cerebrale** è rivestito esternamente da uno strato di corteccia cerebrale e tre strutture profonde:
 - I gangli alla base, regola le prestazioni motorie;
 - L'ippocampo, regola gli aspetti della memoria;
 - I nuclei amigdaloidi coordinano l'attività autonoma e risposte endocrine degli stati emotivi

A livello dei sistemi cognitivi la corteccia cerebrale è la parte più importante da studiare e da conoscere.

Si possono dividere le informazioni in due grandi insiemi

1. BOTTOM UP INFORMATION: si tratta di dati elaborati nel cervello, a livello computazionale si estraggono bordi e angoli, si riconoscono gli oggetti di base e le facce. Si estraggono gli oggetti dalla prospettiva, si riducono le ambiguità, tramite queste elaborazioni comprendiamo l'arte;
2. TOP DOWN INFORMATION: l'elaborazione avviene ad un livello più inconscio, tramite l'attenzione, l'osservazione passata, le informazioni apprese e l'esperienza personale. Risolve altre ambiguità, si creano ipotesi, si associano i dati visivi all'esperienza. Utile per comprendere l'arte astratta.

Inizia una parte che la prof ha fatto molto velocemente e che vuole che sappiamo per conoscenza e curiosità.

Image processing a livello della retina

Nella retina avviene il processing delle immagini, si riduce il contrasto, si effettua la correzione del rumore, si raddrizzano i contorni, l'immagine passa attraverso bastoncelli e coni per raggiungere il nero ottico.

Il campo recettivo è la parte di retina in cui le sue cellule vengono stimolate, dove avviene la combinazione dei segnali per determinare i contorni delle cose che guardiamo (corrispettivo ad un filtro laplaciano).

Le cellule della retina presentano un campo a loro volta che può essere modellato come un gaussiano, e i campi si possono calcolare come differenze di gaussiani (DOG).

Ogni gruppo di cellule è in grado di acquisire una variazione luminosa.

Retinal image processing

- Filtering, --- noise reduction, bilinear filters etc... -- noise deblurring using homography
- Jiaia Jia Hong Kong : see **Forward motion deblurring** (ICCV2013)
- Similar in **retinal ganglion cells** they changes polarities
- Providing deblurring
- In neuroscience →
- Two types of ganglion cells:
- Large magnocellular ganglion cells (M cells) transmit information for Movement, location, depth
- Smaller parvocellular ganglion cells (Pcells) transmit information for Colour, Form, texture
- ➔ **specific vision descriptors (HANDCRAFT OR DEEP) for different features, similar to different cells**

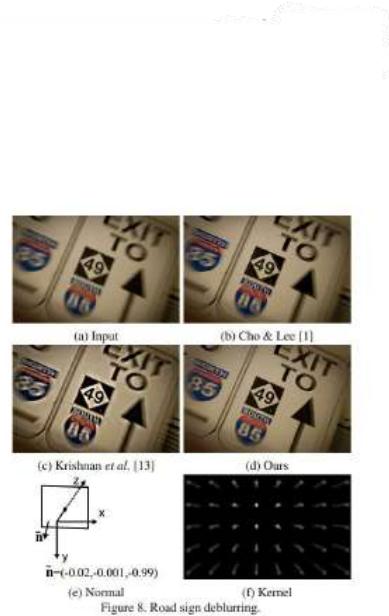
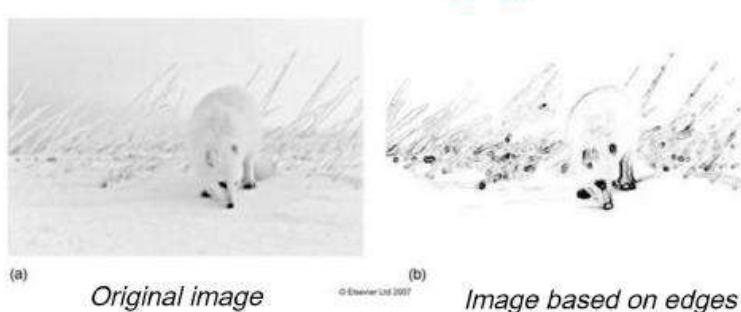


Figure 8. Road sign deblurring.

What happen in retinal ganglion cells?

- They do gradient and edge detection!
 - The signals arriving for natural ganglion cells are edge based; then there is a lateral inhibition for enhancing edges where light is changing
 - Uniform parts are less interesting



-  As the Canny edge detector

From <http://slideplayer.com/slide/3241743>

Finisce qui la parte fatta velocemente, da qui in poi si è soffermata di più.

Per provare a creare una similarità tra quello che avviene nella nostra mente e quello che viene implementato nella computer vision:

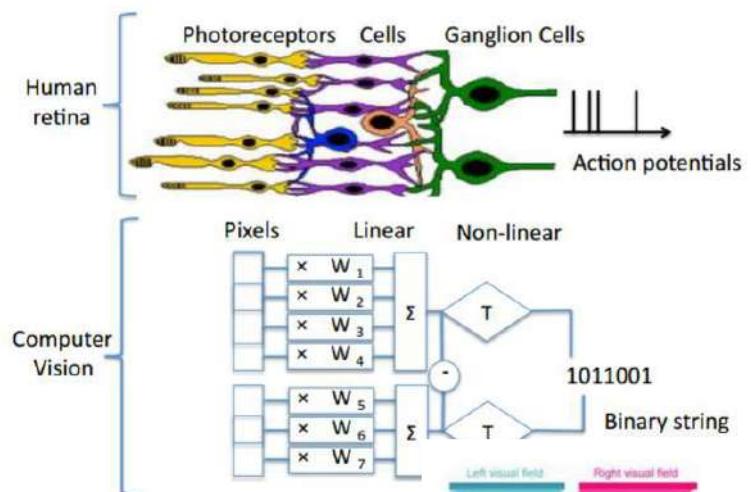
Dalla retina alla corteccia → COMPRESSIONE

Nella retina avviene la percezione dell'immagine tramite i 100 milioni di fotorecettori presenti nell'occhio umano, mentre la compressione avviene a livello di assoni nel nervo ottico che sono circa 1 milione.

Comparazione con gli autoencoder.

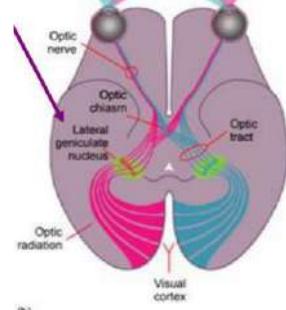
Dal nervo ottico al LGN

L'informazione compressa dalle cellule del ganglio retinico arrivano al LGN.



1. La compressione dei segnali viene parzialmente eseguita nella retina
 2. LGN ha il compito di provvedere ad un filtraggio dell'immagine
 3. Processo dinamico dell'informazioni, si selezionano i segnali utili nel campo visivo, coinvolgendo anche le informazioni sul movimento
 4. Trasferimento dei dati da V1 a V5 della corteccia.

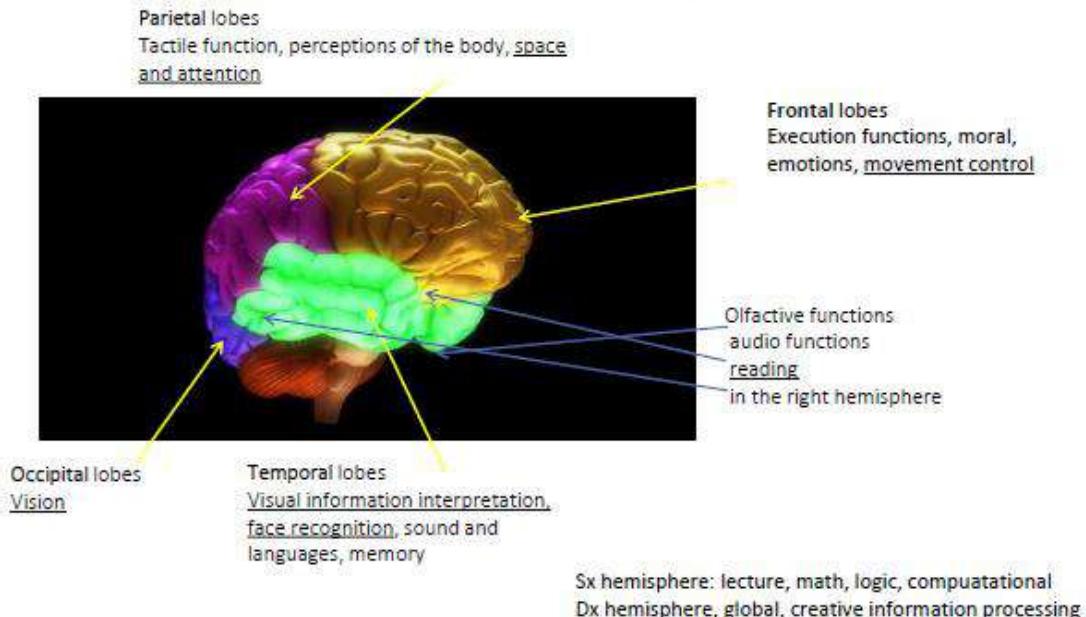
Comparazione con l'attivazione non lineare delle CNN, compressione, campionatura e selezione.



Ogni parte della nostra corteccia cerebrale ha un compito:

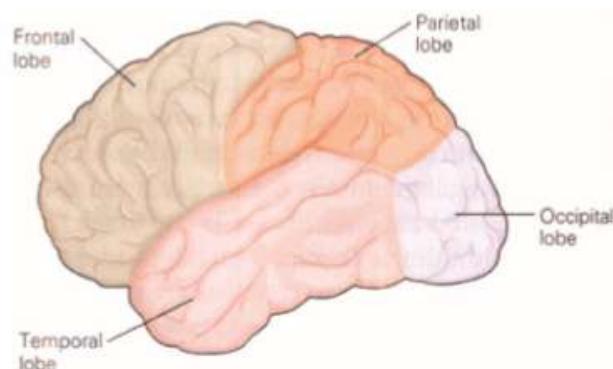
Brain-CORTEX

- Two hemispheres covered by the cortex about 10 billion of neurons.



Multimodality

The Cortex



- the frontal lobe is largely concerned with planning future action and with the control of movement;
- the parietal lobe with somatic sensation, with forming a body image, and with relating one's body image with extrapersonal space; (the where)
- the occipital lobe with vision;
- the temporal lobe with hearing; (and the what) and through its deep structures—the hippocampus and the amygdaloid nuclei—with aspects of learning, memory, and emotion.

Multimodality

Each lobe has several characteristic: deep infoldings (a favored evolutionary strategy for packing in more cells in a limited space). The crests of these convolutions are called gyri, while the intervening grooves are called sulci or fissures.

Semir Zeki ed Eric Kendal hanno proseguito con gli studi di neurofisiologia e hanno potuto appurare che nella nostra corteccia non c'è in realtà una sola area adibita alla visione, ma sono 5 aree connesse alla visione.

- V1, riordina i segnali che arrivano dalle diverse task visive, è una sorta di image processing come lo abbiamo studiato fino ad oggi;
- V2 e V3 sono adibite alla percezione della forma → edge detection;
- V4 è relativa ai colori;
- V5 è relativa al movimento e alla direzione.

Un concetto importante è che nella corteccia primaria relativa alla zona V1 i neuroni sono sensibili a orientazione, direzione del movimento, colore:

- L'orientazione è usata per determinare i contorni
- La direzione è usata per tracciare gli oggetti (capire se sono pericolosi)
- I colori sono usati per prevenire la mimetizzazione degli oggetti
- Le disparità binoculare per percepire la disparità di entrambi gli occhi

→ Quindi in V1 corrisponde ad operazioni come keypoints detection, local receptors e descriptors, edge detection, convolutive descriptors

Attacco slides su cui non si è soffermata e ha saltato slides da 40 a 45.

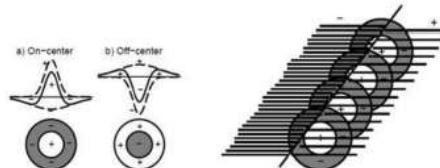
Edge detection



- We (humans and computers!) provide edge detection by computing differential derivatives in the brightness space
- Simple V1 cells combine data into edge detectors, enable detection of shapes, while other detectors work on color and texture

Different detectors with

- Different orientation
- high frequency for narrow bands and fast changes
- low frequency of wide bands gentle changes
- polarization is used for dark-light or viceversa



- ...Gradient and thresholds, convolutive filters...

Se interessati leggere il paper

Now selection of similar coarse and narrow filters in Deep Learning..
see

Iosif Kokkinos Inria France
"PUSHING THE BOUNDARIES OF
BOUNDARY DETECTION USING DEEP
LEARNING " [ICRL 2016]

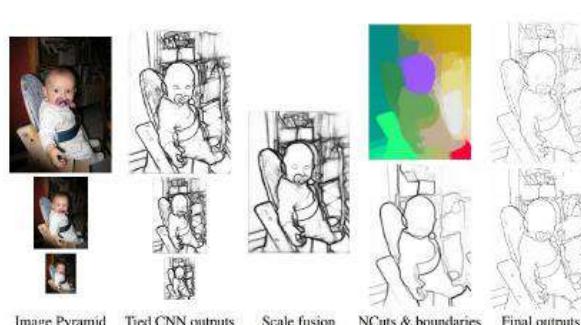
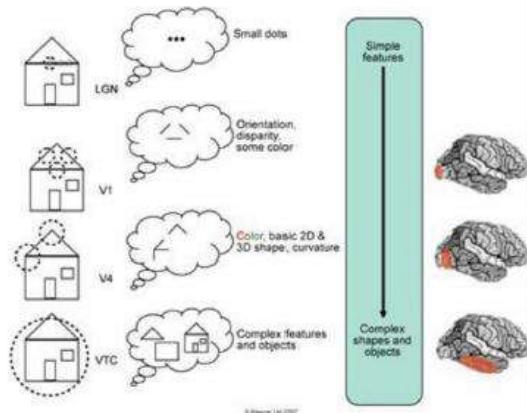


Figure 3: Overview of the main computation stages in our system: an input image is processed at three different scales in order to obtain multi-scale information. The three scales are fused and sent as input to the Normalized Cuts algorithm, that delivers eigenvectors (we show the first three of eight dimensions as an RGB image) and the resulting 'Spectral Boundaries'. The latter are fused with the original boundary map, nonmaximum suppressed, and optionally thresholded (bottom row). All stages are implemented in Caffe, requiring less than a second on an Nvidia Titan GPU.

More complex tasks

- From image processing to recognition
- From V1 (as a hub) through V4 and the Ventral Temporal Cortex (VTC) neurons respond to complex stimuli
- Retina and LGN extract dots (...*keypoints*)
- V1 dots are combined in edges (...*labeling*)
- V4 edges are combined into shapes and color features
- In VTC are combined into objects (...*object detection*)

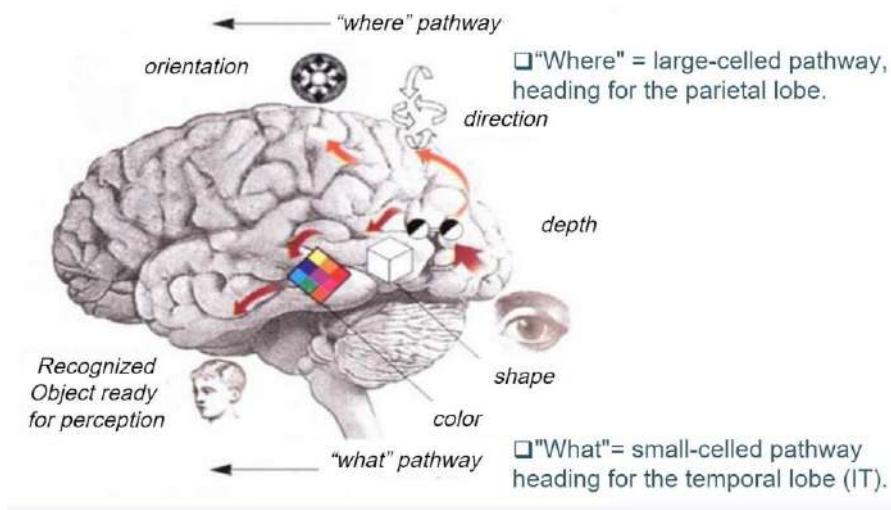


Uno dei campi applicativi di maggiore interesse in computer vision è quello del Tracking, cioè poter individuare oggetti in movimento, un'applicazione per questo campo può essere obstacle detection.

Per comprendere il problema del tracking fino in fondo è stato utile come Kendal abbia diviso gli stimoli che dalla realtà arrivano alla retina in due sentieri paralleli:

1. La via del cosa: cioè percezione del colore, della forma, della texture dell'oggetto
2. Le via del dove: cioè percezione della localizzazione e della direzione di quel dato oggetto

Il riconoscimento e le localizzazioni sono fatti in parallelo. → Concetto molto importante nell'object tracking.



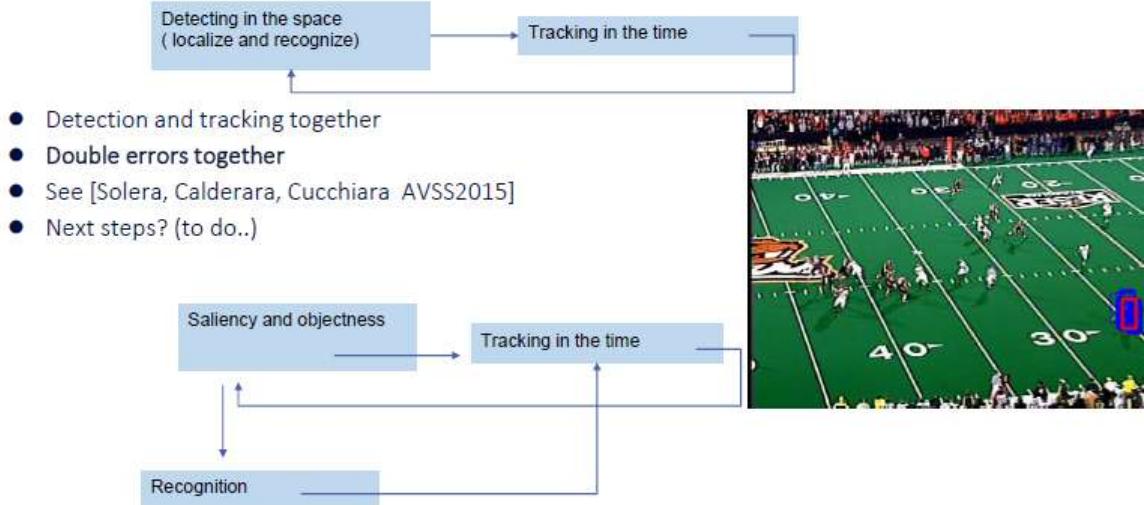
Dopo una prima percezione nella area del cosa, l'informazione viene processata di nuovo in maniera topdown.

- a. Si eliminano i dettagli che non sono rilevanti per il contesto
- b. Si cerca di comprendere la coerenza di quello che vediamo (ha una testa? Ha degli arti?)
- c. Si cerca di astrarre le features essenziali che permettono di discriminare gli oggetti fra di loro
- d. Si confrontano le immagini e le informazioni con le immagini e le informazioni che abbiamo elaborato nella nostra esperienza personale.

Una considerazione importante da fare è che la visione umana non è solo un processo di informazioni assemblate a livello dei nostri organi visivi e recettivi, comprende una parte importante di psicologia e percezione della realtà che può essere anche soggettiva.

Come abbiamo già detto precedentemente OBJECT TRACKING è un campo di applicazione molto importante delle nozioni viste finora.

- Typical (current) sequential pipeline



Si conclude così il percorso nelle analogie tra neuroscienza, neurofisiologia, psicologia e computer vision.

I sistemi cognitivi e la computer vision basano le loro architetture sul processo cognitivo che avviene nella mente umana, ma gli studi di neurologia sulla visione sono molto recenti e si conosce ancora poco sull'argomento.

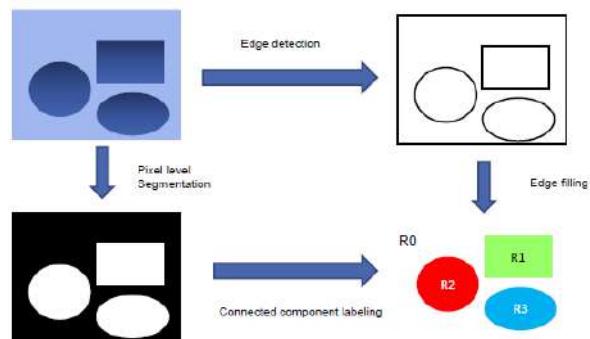
The way of what: SEGMENTATION

Data un'immagine che presenta una regione di interesse in primo piano e un background, il compito della segmentazione è quello di individuare la regione o il segmento dell'immagine nella regione di interesse.

Questo tipo di operazione si può fare partendo da una semplice edge detection per poi andare a riempire gli edge che sono stati individuati.

Oppure facendo una segmentazione a livello di pixel ed è quello che vedremo in questa sezione.

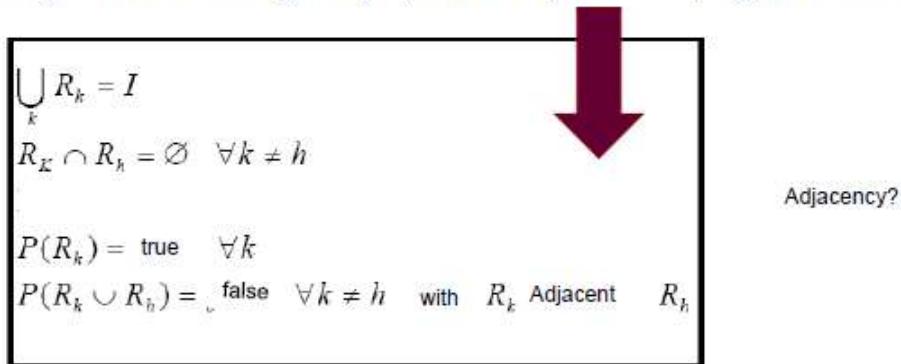
La segmentazione è l'operazione da eseguire prima del riconoscimento dell'oggetto. Tramite la segmentazione, infatti, è possibile distinguere tra l'oggetto e il background. Dopo aver determinato la forma e la posizione dell'oggetto possono essere calcolati la dimensione, il centroide, l'orientamento.



È importante distinguere tra la segmentazione percettiva e quella semantica, a livello percettivo si estrae semplicemente l'oggetto dal background in cui si trova nell'immagine ma non si effettua alcuna considerazione semantica. Per questo motivo la segmentazione si può intendere come un problema di raggruppamento, trova diverse applicazioni nella cluster analysis. Infatti quello che si effettua con la segmentazione è il raggruppamento di pixels simili. Sia la segmentazione che il raggruppamento sono problemi indefiniti (illdefined).

La segmentazione consiste nel dividere le immagini in insieme di punti omogenei chiamati *regioni* dell'immagine (che spesso corrispondono a oggetti reali nel mondo 3D) date features visive. Spesso una *regione* può essere chiamata *oggetto* se viene effettuata una associazione semantica ad essa.

DEF) *Segmentation: Given an image I and a predicate P (homogeneity criterium, based on a visual feature) image segmentation means finding a partition S of I in a set of regions R_{1..R_n} so that*

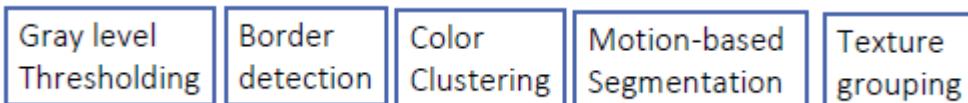


Le regioni in cui l'immagine viene suddivisa devono soddisfare alcune proprietà:

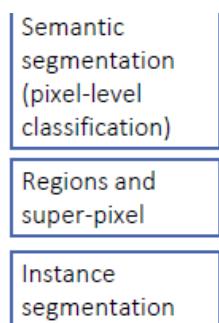
- Distinte, se nessun pixel è condiviso da due regioni
- Complete, tutti i pixel dell'immagine sono assegnati ad almeno una regione della partizione
- Connesse, tutti i pixel appartenenti alla stessa regione sono connessi
- Omogenee, tutte le regioni sono omogenee rispetto ad un criterio fissato (intensità, colore, texture..)

Possiamo quindi concludere che ciascun pixel di una regione è simile agli altri della stessa regione per una qualche proprietà o caratteristica (colore, intensità, texture). Regioni adiacenti sono significativamente differenti rispetto ad almeno una di queste caratteristiche.

Ci sono diversi metodi che possono effettuare segmentation



Ovviamente alla base della segmentazioni possono esserci diversi criteri di suddivisione delle regioni, ne vediamo alcuni come:



Il metodo più comune per fare segmentazione di regioni in un'immagine è quello del color clustering. AlImageLab ha infatti lavorato ad un paper dal titolo "Comparison of color clustering algorithms for segmentation of dermatological images".

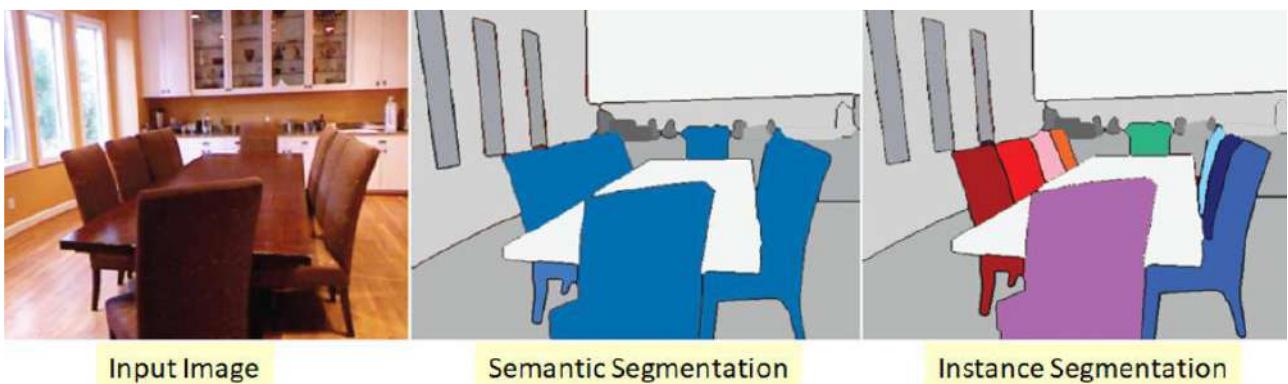
Semantic segmentation

La segmentazione semantica è un problema di classificazione, ogni pixel è classificato considerando un'etichetta e dalla classificazione di ogni pixel si determinano le regioni dell'immagine. Si tratta di un problema di supervised learning, oggi è un approccio molto usato grazie all'uso di grandi database di immagini annotati, vedi ImageNet.

La segmentazione semantica è utile per applicazioni in una varietà di settori che richiedono mappe immagine precise, come ad esempio:

- Guida autonoma: per identificare un percorso percorribile dalle auto separando la strada da ostacoli come pedoni, marciapiedi, pali e altre auto
- Ispezione industriale: per rilevare difetti nei materiali, come ad esempio nell'ispezione dei wafer
- Immagini satellitari: per identificare montagne, fiumi, deserti e altri tipi di terreno
- Imaging medicale: per analizzare e rilevare anomalie cancerose nelle cellule
- Visione robotica: per identificare e muoversi tra oggetti e terreni

Una approccio più evoluto della semantic segmentation è l'instance segmentation. Con l'instance segmentation le cose si complicano perché non basta individuare regioni che sottostanno ad uno stesso criterio semantico, ma ogni oggetto che fa parte di queste regioni viene a sua volta segmentato. Necessita per questo motivo di una conoscenza a priori del contesto.

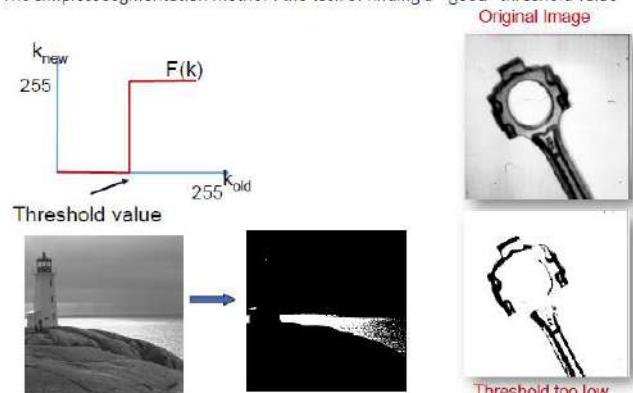


Metodo di segmentazione - Thresholding

Il thresholding o sogliatura è il metodo più antico di segmentazione dell'immagine. Da un'immagine a livelli di grigio, la sogliatura restituisce un'immagine binaria, a cui a sua volta viene associato il cosiddetto *bimodal histogram*.

Thresholding:

The simplest segmentation method : the task of finding a "good" threshold value



Original Image



Binary Image



Threshold too low

Threshold too high

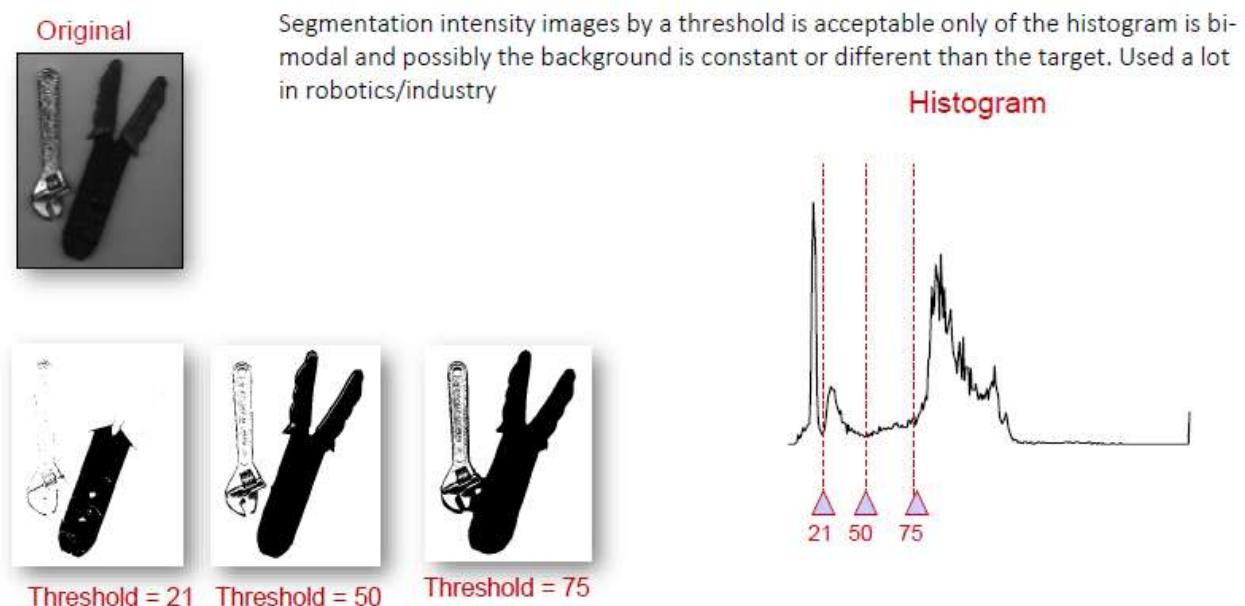
Durante un processo di sogliatura, singoli pixel dell'immagine sono catalogati come "pixel oggetto" se il loro valore è maggiore di una certa soglia e come "pixel di sfondo" se il valore è sotto la soglia. Solitamente l'immagine binaria in uscita ha valore pari a "1" dove è presente l'oggetto e pari a "0" per lo sfondo, ottenendo quindi un'immagine in bianco e nero.

Ovvero per un'immagine originale a livelli di grigio sogliata

$I_{originale}(x)$ la sogliatura si ottiene così:

$$I_{binaria}(x) = \begin{cases} 0 & \text{se } I_{originale}(x) < \text{soglia} \\ 1 & \text{se } I_{originale}(x) \geq \text{soglia} \end{cases}$$

Dove x rappresenta il *digital number* (DN) ovvero il valore del pixel nell'immagine.



Il problema di un metodo come la sogliatura è la scelta della soglia, perché è necessario individuare un valore T di intensità capace di dividere l'immagine in regioni di pixels.

- Given an image $I(x)=f(x)$ ($x=(i,j)$), it is transformed to $g(x)$
- $g(x) = Thresh(f(x), T)$

```
For each (i)
  For each (j)
    if (f(i, j) >= T)  g(i, j)=1;
                        else g(i, j)=0;
```

La soglia può essere:

- Globale se $T = T(f)$, quindi applicata in maniera uniforme su tutta l'immagine
- Adattiva se $T = T(f, W(i,j))$, se dipende dalla regione o window W dell'immagine in cui è calcolata.

È possibile definire T automaticamente ?

Di solito si conosce sempre l'immagine, o il contesto in cui si effettuano algoritmi di visone; ma se questo non è il caso la segmentazione può avvenire solo se il contrasto tra background e foreground dell'immagine è alto, tanto da avere un bimodal histogram.

Otsu thresholding (1979)

Il **metodo Otsu** è un metodo di sogliatura automatica dell'istogramma nelle immagini digitali. L'algoritmo presume che nell'immagine da sogliare siano presenti due sole classi e quindi calcola la soglia ottima per separare queste due classi minimizzando la varianza intra classe.

Si tratta di un metodo abbastanza vecchio ma comunque ancora oggi ben performante.

Algoritmo step-by-step:

1. Calcola per una data soglia t :

- a. $q_1(t), q_2(t)$, le probabilità a priori che un pixel appartenga o meno al gruppo 1 o al gruppo 2

$$q_1(t) = \sum_{i=1}^t P(i) \quad q_2(t) = \sum_{i=t+1}^L P(i)$$

Dove $P(i)$ è la probabilità a posteriori calcolata sull'istogramma (normalizzato).

Calcolo della probabilità a posteriori:

Dati n pixels p di un'immagine $P(i) = \#\{p: I(p) = i\}/n \rightarrow P(i) = H(i)/n$

2. Calcola la media e la varianza

$$\begin{aligned} \mu_1(t) &= \sum_{i=1}^t iP(i)/q_1(t) & \sigma_1^2(t) &= \sum_{i=1}^t [i - \mu_1(t)]^2 P(i)/q_1(t) \\ \mu_2(t) &= \sum_{i=t+1}^L iP(i)/q_2(t) & \sigma_2^2(t) &= \sum_{i=t+1}^L [i - \mu_2(t)]^2 P(i)/q_2(t) \end{aligned}$$

3. Calcola la varianza all'interno del gruppo

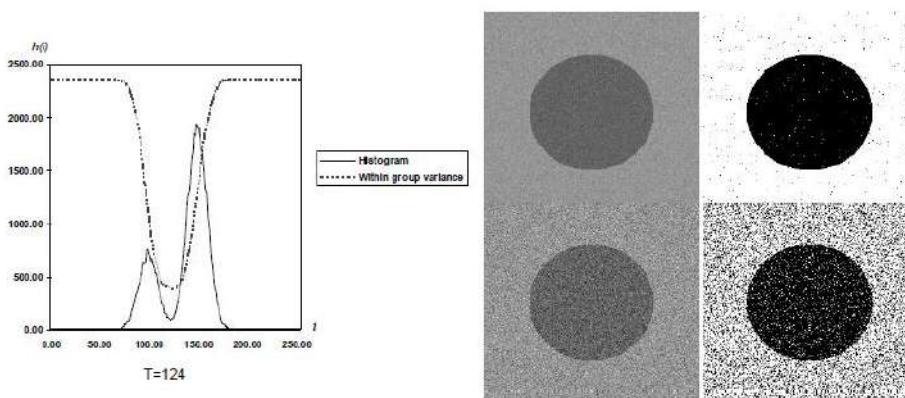
$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

Varianza all'interno del gruppo: più piccola è la wgv, allora sono più compatti al valore medio i due gruppi

4. Minimization

For each $t=1, \dots, L$ cerca il valore minimo calcolato al punto 3

Otsu's Threshold example



L'idea qui è che la varianza sia minimizzata. È molto usato come metodo quando si conosce che il background dell'immagine è costante. Ma quando il background non è costante è meglio usare una sogliatura adattiva.

Sogliatura adattiva

Metodo molto usato negli scanner per esempio.

Invece di calcolare una soglia univoca per una sola intera immagine, si calcola T solo una finestra $W(i, j)$. Le dimensioni della finestra dipendono dal problema e dalla variabilità dei dati.

T può essere:

- $T = \text{mean}(W)$
 - $T = \text{median}(W)$
 - $T = (\text{Max}(W) - \text{Min}(W)) / 2$
- Sperimentalmente è stato appurato che il caso migliore è $T = \text{mean}(W) - C$

Dove C è una costante calcolata con qualche metodo globale (non è specificato nelle slides).

● Algorithm (Linda Shapiro in the 1980'):

1) An initial threshold T is chosen, randomly or according to any other method.

2) The image is segmented into object and background pixels as described above, creating two sets; given $f(p)$ the pixel value:

$$G_1 = \{f(p) : f(p) > T\} \text{ (object pixels)}$$

$$G_2 = \{f(p) : f(p) \leq T\} \text{ (background pixels)}$$

3) The average of each set is computed.

$$m_1 = \text{mean } G_1 \quad m_2 = \text{mean } G_2$$

4) A new threshold is created that is the average of m_1 and m_2 $T' = (m_1 + m_2)/2$

5) Go back to step two, now using the new threshold computed in step four, keep repeating until the new threshold matches the one before it (i.e. until convergence has been reached).

Do you recognize this pattern recognition method?

L'algoritmo iterativa introdotto da Shapiro negli anni 80 è un caso speciale unidimensionale del *k-means*, il quale ha provato che si può convergere ad un minimo locale.

Cerca di ottenere il partizionamento dei dati minimizzando il valore della somma dei quadrati delle distanze di ciascun pattern con il prototipo più vicino (criterio del mean square error). Genera K cluster $\{C_1, C_2, \dots, C_K\}$ tali che il pattern ni appartiene esattamente ad un unico cluster e quel cluster sia C_i .

Nearest Neighbour Clustering



- Repeat
- 1) Compute the centers of classes m_1 and m_2
 - 2) Redefine the two groups so that
- all grey levels in set 1 are nearer to cluster centre m_1 and all grey levels in set 2 are nearer to cluster centre m_2
 - Repeat it until none of the pixel labels changes anymore.

$$|g_1(k) - m_1| < |g_1(k) - m_2| \quad k = 1..N_1$$

$$|g_2(k) - m_2| < |g_2(k) - m_1| \quad k = 1..N_2$$

Esempio di K Means

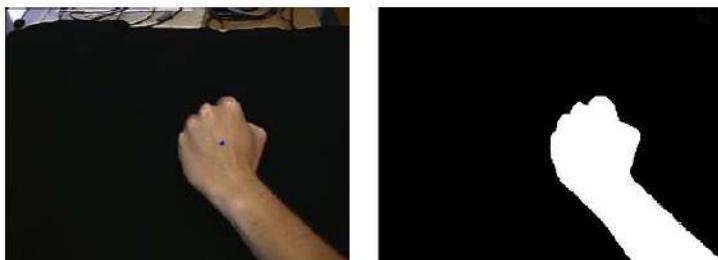


Image Segmentation when K=6

È possibile applicare algoritmi di sogliatura anche ad immagini RGB. Un semplice approccio potrebbe essere quello di scomporre l'immagine nei suoi tre canali RGB e sogliarli tutti indipendentemente. Le tre immagini binarie sono poi processate attraverso un'operazione logica di and.



Oppure è possibile usare lo spazio HSV se lo spazio di colore ha significato.



Labeling

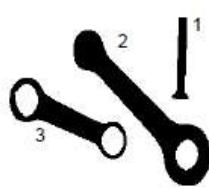
Labeling o etichettatura è un problema ad ampio spettro della computer vision, che può essere associato anche alla segmentazione. Significa infatti identificare segmenti di pixels che sono “connessi” o “adiacenti” con la stessa label. Come abbiamo visto fino ad ora, con un approccio tipico di sogliatura si presentano solo due classi background e foreground, ma nella stessa immagine possono esserci oggetti diversi che quindi hanno bisogno di etichette diverse.



Original image



Otsu threshold



after Morphology and Labeling

I problemi che sorgono con la labeling sono causati dal fatto che non si è ancora stabilito un concetto univoco di adiacenza e distanza tra due pixels.

Le misure di distanza che possono essere applicate sono:

- La distanza Euclidea, computazionalmente più complessa;
- La distanza CityBlock, le cui direzioni di movimento sono N,S,E,O

$$D4(i, j)(h, k) = |i - h| + |j - k|$$

Il minimo numero di passi in una griglia per raggiungere un punto;

- La distanza Chessboard, è in grado di muoversi in diagonale sui pixel

$$D8(i, j)(h, k) = \max\{|i - h|, |j - k|\}$$

Il numero di passi del re in una scacchiera

Connected component labeling (CCL): trasforma un'immagine binaria in input (già effettuata la segmentazione background foreground) in una simbolica in cui tutti i pixels che appartengono allo stesso componente hanno la stessa label;

Pixels Adiacency (connection in 2D): due pixels sono adiacenti se la loro distanza è uguale al range di discretizzazione dell'immagine

Connessioni tra pixels:

4-neighborhood

$$N_4(p) = \{q \in L \mid |p_x - q_x| + |p_y - q_y| \leq 1\}$$

8-neighborhood

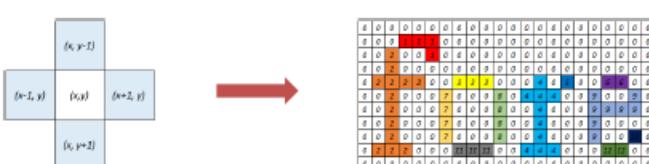
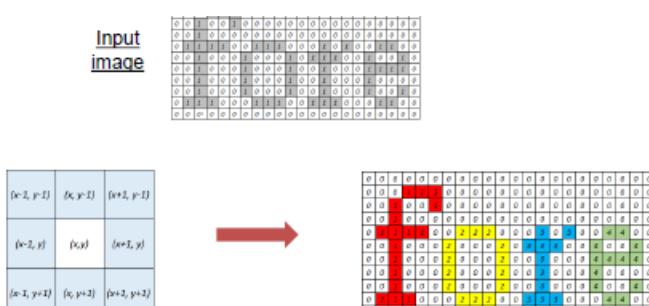
$$N_8(p) = \{q \in L \mid \max(|p_x - q_x|, |p_y - q_y|) \leq 1\}$$

Dove L è un'immagine definita su due layer L and I(p) è il valore al pixel p(x, y) su L

Background pixels: F = {p ∈ L | I(p)=0}

Foreground pixels: B = {p ∈ L | I(p) = 1}

Scelta la misura di distanza da utilizzare gli algoritmi di labeling sono deterministici e univocamente definiti.



Gli algoritmi di labeling possono dividersi in **3 macro categorie**. Ricordiamo la definizione di CCL Labeling: algoritmi che associano un'unica etichetta ad ogni regione connessa dell'immagine.

- I. **Multiscan**
 - a. Algoritmi iterativi, lenti ma buoni nel caso di allocazioni di risorse
 - b. Algoritmi classici
- II. **2 – Steps**
 - a. Algoritmi classici su classi equivalenti (?) (veloce ma povero nel caso di allocazioni di risorse)
 - b. Run-length
 - c. Decision table e algoritmi con Tree
- III. **Labels-propagation algorithm**
 - a. Contour tracing and filling (openCV)

Classic Labeling Rosenfeld

Si tratta di un algoritmo del 1966. È un approccio classico al problema del labeling e utilizza un **raster scan** sull'immagine. (Raster scan = linee di scansione, sono le linee in cui è suddivisa un immagine sottoposta a scansione elettronica).

L'output è un immagine contenente le labeling risultato e l'algoritmo mantiene anche una *equivalence table* in cui sono contenute le "ridondanze" (equivalenze) dei pixels e il riferimento al loro neighborhood.

Viene poi processata la tabella, tramite degli algoritmi di sort e vengono rimosse le ridondanze. Dalla tabella ordinata si aggiornano le label sull'immagine.

Il problema di questo algoritmo è il suo costo computazionale, bisogna infatti memorizzare sia l'immagine che l'equivalence table, inoltre bisogna applicare un algoritmo di sort.

Classic iterative multiscan Labeling (Haralick)

Haralick propose un miglioramento dell'algoritmo esposto sopra nel 1981. In questo approccio non viene usata l'equivalence table, ma si effettuano iterativamente dei passaggi di forward e di backward con il raster scan sull'immagine di output per risolvere le ridondanze (equivalenze). Nei passaggi di risoluzione si lavora sui pixels di neighborhood.

Questo approccio è meno costoso a livello di memoria, ma è comunque computazionalmente complesso mano a mano che la dimensione dell'immagine binaria aumenta.

Labels propagation

Si tratta dell'approccio più utilizzato negli ultimi anni, presenta due passi fondamentali:

1. Ricerca con raster scan e etichettare i pixels senza label; (inizializzazione)
2. Propagare la stessa label sui pixels che sono connessi fra di loro.

I pro di questo approccio:

- Non necessita di memoria per la equivalence table
- Questo tipo di algoritmi con labels propagation dato che le label di inizializzazione non cambiano possono essere utilizzati per calcolare le features di shape degli oggetti.

I contro di questo approccio:

- Non sono cache friendly, quindi non performanti in termini di lettura su memoria;
- Non sono adatti per una implementazioni parallela, dato che non si fondano sul solo raster scan come quelli classici.

Esempi di Labeling Propagation:

Iterative multiscan Algorithm

- I. Inizializzazione: ogni pixel dell'immagine è numerato con una label consecutiva
- | | | |
|--------|--|----------------------|
| before | | After initialization |
|--------|--|----------------------|

<code>0 1 1 0 1 1 0</code>		<code>0 1 2 0 3 4 0</code>
<code>0 1 1 0 1 1 0</code>		<code>0 5 6 0 7 8 0</code>
<code>0 1 1 1 1 1 0</code>		<code>0 9 10 11 12 13 0</code>

- II. Top down: si scannerizza dall'alto verso il basso da sinistra a destra per dare la stessa label a pixels connessi;
- III. Bottom up: si scannerizza dal basso verso l'alto da destra a sinistra per dare la stessa label a pixel connessi;
- IV. Ripetere step II e III tante volte quante servono per far smettere alle label di cambiare.

`min_topdown_neighborhood(i,j)` and `min_bottomup_neighborhood(i,j)` return the minum label between adjacent pixels (i,j) :

top-down:	bottom-up:
<code>1 2 3</code>	<code>0 0 0</code>
<code>4 p 0</code>	<code>0 p 4</code>
<code>0 0 0</code>	<code>3 2 1</code>

`min_topdown_neighborhood(i,j)` returns 1 `min_bottomup_neighborhood(i,j)` returns 1

After the step

top-down:	bottom-up:
<code>0 1 1 0 3 3 0</code>	<code>0 1 1 0 1 1 0</code>
<code>0 1 1 0 3 3 0</code>	<code>0 1 1 0 1 1 0</code>
<code>0 1 1 1 1 1 0</code>	<code>0 1 1 1 1 1 0</code>

- Problem: the number of iterations can be to high → very slow

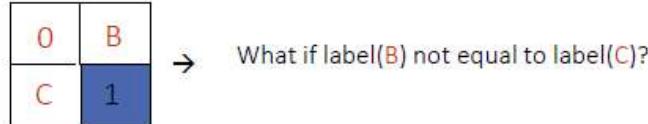
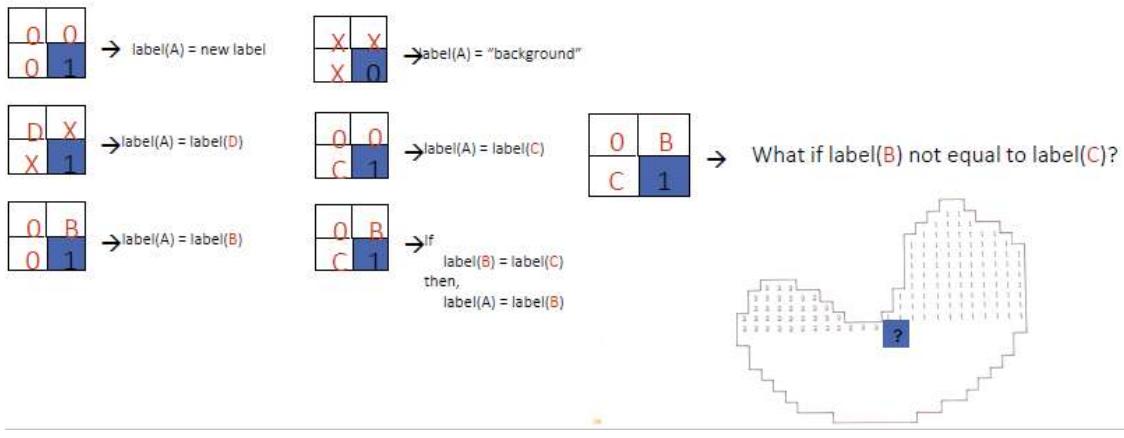
Two scan algorithm + Equivalence table

- I. Nel caso in cui ad un pixel può equivalere più di una label allora gli viene assegnata la label con valore più basso e la relazione di equivalenza viene memorizzata in una tabella
- II. La tabella viene scansionata e ordinata fino a che tutte le equivalenze non sono risolte

image	First step
<code>0 1 1 0 1 1 0</code>	<code>0 1 1 0 2 2 0</code>
<code>0 1 1 0 1 1 0</code>	<code>0 1 1 0 2 2 0</code>
<code>0 1 1 1 1 0 1</code>	<code>0 1 1 1 1 0 3</code>
	→ Equivalence
	<code>1=2</code>
	<code>2=3</code>
	Second step
	<code>0 1 1 0 1 1 0</code>
	<code>0 1 1 0 1 1 0</code>
	<code>0 1 1 1 1 1 0</code>

The time to process the equivalence table can be too high

Equivalence table:



Solution:

Let: $\text{label}(A) = \text{label}(B) = 2$

Create EQUIVALENCE TABLE

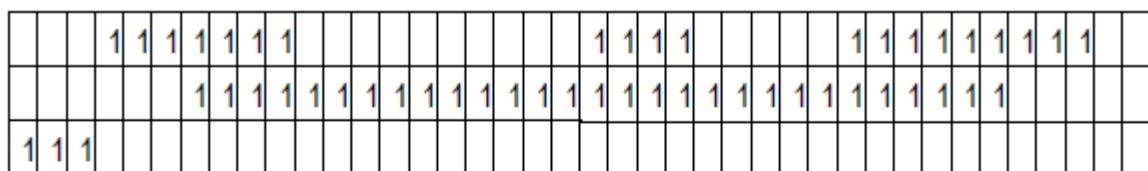
Resolve Equivalence in Second Pass

The slow part is solving the equivalence table
and find the unions . Possibly in a single pass

$2 \equiv 1$
$7 \equiv 3,6,4$
\vdots

Run Length (solo nominato a lezione inserisco per completezza)

Schwartz 1985, è un approccio nato per evitare di memorizzare l'immagine di output e quindi risparmiare memoria. Si produce una rappresentazione compatta dell'equivalenze di label.



	ROW	START COL	END COL	PERM LABEL
1	1	4	10	1
2	1	20	24	2
3	1	30	38	3
4	2	7	35	1

In questo modo, dopo una scannerizzazione forward e una backward, l'output può essere una struttura ausiliaria utilizzata per dedurre la label del pixel.

(Dal web)

L'algoritmo **Run-length** è il primo algoritmo di compressione per le immagini. Si tratta di un metodo di tipo **lossless** (senza perdita di dati) che riduce la ripetizione di elementi uguali all'interno del file. Per ogni elemento trovato uguale, l'algoritmo sostituisce un **RUN**, ovvero un insieme di 3 caratteri: il primo indica il codice identificativo del RUN, il secondo il carattere sostituito, ed infine il terzo il numero di volte che il carattere deve essere ripetuto.

Union find

Samet nel 1986 presenta una soluzione semilineare per il problema delle equivalence. Il nome dell'algoritmo deriva dalla struttura dati **union-find** che utilizza. **Union-find** è una struttura dati derivante dal concetto di partizione, per cui dato un insieme finito di elementi a volte risulta utile partizionarli in insiemi disgiunti. Le due operazioni possibili su questa struttura dati sono:

- **Ricerca**: determina in quale insieme si trova un particolare elemento, o se due elementi appartengono allo stesso insieme
- **Unione**: combina o fonde due insiemi in un unico insieme

L'altra operazione è **Crea**, tramite la quale è possibile dato un insieme creare la partizione formata solo da singoletti.

L'algoritmo in questione è composto da due passi:

1. Crea un file intermedio contenente gli elementi dell'immagine e le classi di equivalenza;
2. Il secondo passo processa il file creato al passo 1 in ordine inverso e assegna una label finale ad ogni elemento dell'immagine.

Quindi mentre un pixel viene calcolato, la label di equivalenza viene risolta. Gli approcci precedenti partivano con una collezione di labels e come ultimo passo risolvevano le equivalenze, in questo caso invece per ogni pixel viene aggiornata la struttura dati.

Nel 2003 Suzuki et al. Hanno ripreso l'approccio di Haralick sulla strategia multiscan sull'immagine, includendovi piccolo array di equivalenza: fornendo un'appendice sotto forma di una LUT di tutti possibili neighborhood, che permette di ridurre tempi e costi di calcolo evitando operazioni non necessarie.

Decision Tree

Nel 2007, He (in collaborazione con Suzuki) ha proposto un altro approccio veloce nella forma di un algoritmo a due scansioni. La struttura dati utilizzata per gestire la risoluzione della label è implementata utilizzando tre array per collegare gli insiemi di classi equivalenti senza l'uso di un puntatore.

Adottando questa struttura dati, sono stati quindi proposti due algoritmi:

1. Viene utilizzata una prima scansione run-based, in cui un albero decisionale ottimizza l'esplorazione sui neighborhood per applicare il merging solo quando necessario.
2. Grana et al. 2010: migliorarono l'approccio al punto 1 con l'utilizzo della tabella delle decisioni per implementare un decision tree applicabile anche su grandi neighborhood

Approccio generale:

- I. Nel primo passo, si assegna una label provvisoria per ogni "oggetto" nell'immagine;
- II. Nel secondo passo, si integrano le label provvisorie ad una sola

Tutti questi algoritmi utilizzano delle scan mask:

- | | | |
|---|---|---|
| p | q | r |
| s | x | |

a	b	c	d	e	f
g	h	i	j	k	l
m	n	o	p		
q	r	s	t		

n1	n2	n3
n4	a	
n5	b	

(Rosenfeld backward mask)

(Grana mask)

(He mask)

- and on a strategy to solve equivalences between labels (Union-Find).

Pro:

- Sono cache friendly
- Possibilità di implementazioni parallele dato che sono algoritmi di tipo raster-scan

Cons:

- Il calcolo delle features shape richiede una spesa computazionale abbastanza onerosa;
- Necessitano di memoria per memorizzare l'equivalenza tra le label.

Decision Table

Si tratta di un approccio importante (lo chiede sicuro).

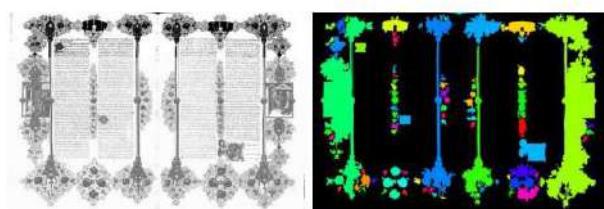
L'approccio con decision table è tipico dell'intelligenza artificiale. Permette la costruzione di un algoritmo che permette di selezionare automaticamente la scelta migliore tra tutte.

Un esempio nella tabella sottostante:

statement section	conditions			actions				
	c ₁	c ₂	c ₃	a ₁	a ₂	a ₃	a ₄	a ₅
r ¹	0	0	0					
r ²	0	0	1			1		
r ³	0	1	0				1	
r ⁴	0	1	1			1	1	
r ⁵	1	0	0					1
r ⁶	1	0	1	1	1	1		
r ⁷	1	1	0				1	1
r ⁸	1	1	1		1	1	1	
condition outcomes			action entries					

Optimized Block-based Connected Components Labeling with Decision Trees

- Model the neighborhood exploration problem with decision tables
- Construct the decision table
- Convert decision tables to optimal decision trees
- Automatic code generation from the decision tree



- AND Decision Table: si devono effettuare tutte le azioni*
- OR Decision Table: si possono effettuare una delle azioni equivalenti. **

*Fisher, D.L. (1966) "Data, Documentation and Decision Tables" CommACM Vol. 9 No. 1 (Jan. 1966)

** C. Grana,D. Borghesani, R. Cucchiara, (2010) "Optimized Block-based Connected Components Labeling with Decision Trees" IEEE Transactions on Image Processing, 2010

Esempio:

p	q	r
s	x	

- For each pixel x , we need to analyze a neighborhood of 4 pixels: p, q, r, s
- the algorithm aims to avoid the need to check ALL conditions. Which is the best action?*
- Conditions: x, p, q, r, s are foreground/background
- Actions:
 - no action,
 - assignment
 - merges of 2 or more labels

x	p	q	r	s	no action		new label				assign				merge			
					x = p	x = q	x = r	x = s	x = p+q	x = p+r	x = p+s	x = q+r	x = q+s	x = r+s	x = p+q+r	x = p+q+s	x = p+r+s	x = q+r+s
0	-	-	-	-	1													
1	0	0	0	0		1												
1	1	0	0	0			1											
1	0	1	0	0				1										
1	0	0	1	0					1									
1	0	0	0	1						1								
1	1	1	0	0							1							
1	1	0	1	0								1						
1	1	0	0	1									1					
1	0	1	1	0										1				
1	0	1	0	1											1			
1	0	0	1	1												1		
1	1	1	1	0													1	
1	1	1	0	1														1
1	1	0	1	1														1
1	0	1	1	1														1
1	1	1	1	1														1

- Some actions are useless: previous operations have already solved equivalences between connected pixels

-

p	q	r
s	x	

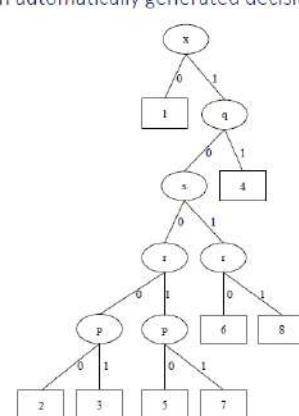
No need to merge $p+q$:
assign p or q

p	q	r
s	x	

No need to merge $p+r+s$:
merge $p+r$ or $s+r$

- The Block-Based with Decision Trees (BBDT) algorithm^[7] employs a new scanning technique that moves on a 2×2 pixel grid over the image which is optimized by an automatically generated decision tree^[8].

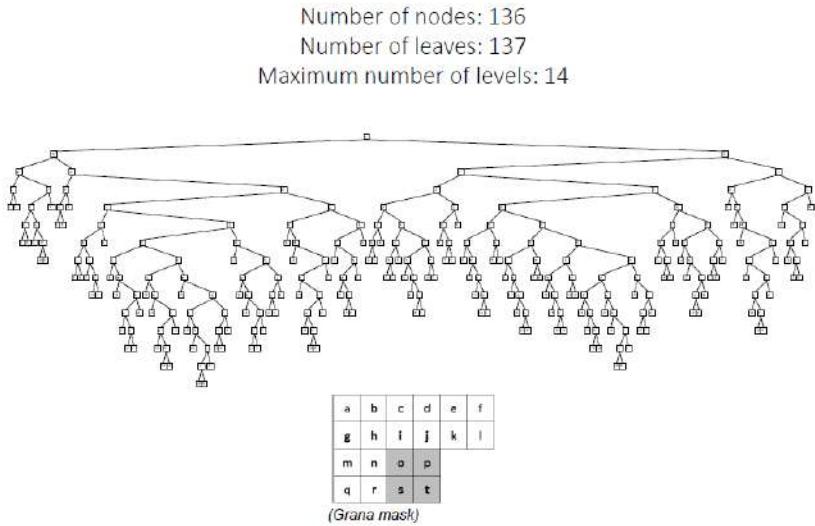
x	p	q	r	s	no action		refuse		assign		merge											
					x < 0	x >= 0	x < 0	x >= 0	x = p	x = q	x = r	x = s	x = p+q	x = p+r	x = p+s	x = q+r	x = q+s	x = r+s	x = p+q+r	x = p+q+s	x = p+r+s	x = q+r+s
0	-	-	-	-	1																	
1	0	0	0	0		1																
1	1	0	0	0			1															
1	0	1	0	0				1														
1	0	0	1	0					1													
1	1	1	0	0						1												
1	1	0	1	0							1											
1	1	1	0	1								1										
1	1	0	1	1									1									
1	0	1	1	0										1								
1	1	1	0	0											1							
1	1	1	0	1												1						
1	1	0	1	1													1					
1	0	1	1	1														1				
1	1	1	1	0															1			
1	1	1	1	1																1		



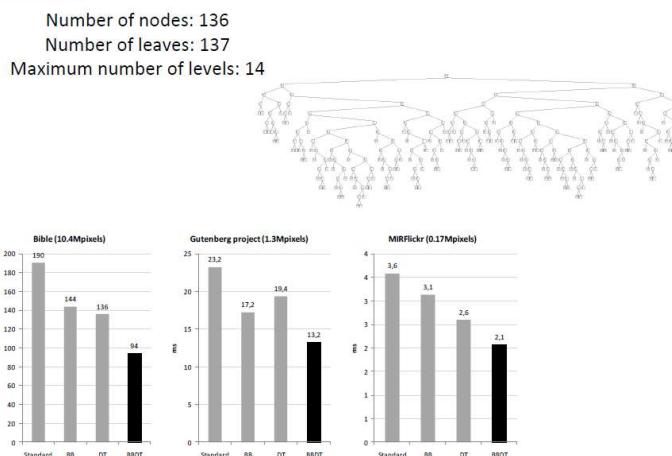
[7] Grana, Costantino, Daniele Borghesani, and Rita Cucchiara. "Optimized block-based connected components labeling with decision trees." *IEEE Transactions on Image Processing* 19.6 (2010): 1596-1609.
 [8] H. Schumacher and K. C. Sevcik, "The synthetic approach to decision table conversion," *Commun. ACM*, vol. 19, no. 6, pp. 343–351, 1976.

Slides su cui non si è soffermata:

Two scan (BBDT)



Going even faster...



TWO SCAN (DRAG)



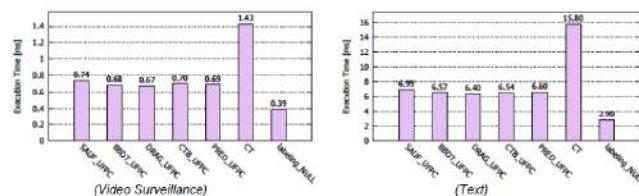
- Connected components labeling on DRAGs algorithm (or simply DRAG)^[9] models the decision problems as Directed Acyclic Graphs with a root (Directed Rooted Acyclic Graphs). The advantage of this representation is that a DRAG, differently from decision trees, will contain only the minimum number of nodes required to reach the leaf corresponding to a set of condition values.



YACCLAB



YACCLAB (*Yet Another Connected Components Labeling Benchmark*)^[10,11] is an open source C++ project to test CCL algorithms under extremely variable points of view, running and testing algorithms on a collection of datasets which cover most of the application fields.



[10] Grana, Costantino, et al. "YACCLAB-Yet another connected components labeling benchmark." *Pattern Recognition (ICPR)*, 2016 23rd International Conference on. IEEE, 2016.

[11] Isolelli, Federico, et al. "Toward reliable experiments on the performance of Connected Components Labeling algorithms." *Journal of Real-Time Image Processing* (2018): 1-16.

OpenCV



- Version 2.4: no specific implementation for CCL but,

```
Python:  
image, contours, hierarchy = cv.findContours( image, mode, method[, contours[, hierarchy[, offset]]])
```

```
Python:  
retval = cv.contourArea( contour[, oriented] )
```

- Version 3.0: introduces an implementation of the SAUF algorithm for CCL (4- and 8-connection);
- Version 3.2: introduces an implementation of the BBDT algorithm for CCL (8-connection);
- Version 3.4: introduces a parallel implementation of both SAUF and BBDT algorithms.

Delle slides sopra non vuole sapere i dettagli.

È importante quindi sapere che con segmentation non si effettua solo un'operazione di classificazione dei pixels, ma anche di aggregazione. Labeling e segmentation permettono di distinguere un oggetto dall'altro in un immagine.

Abstract di un paper per completezza:

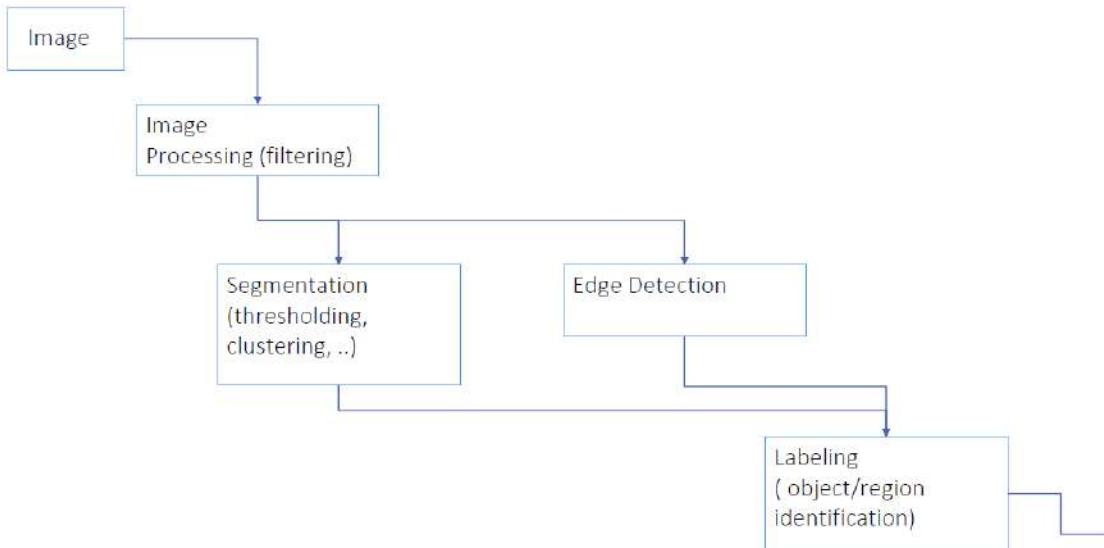
Neuroscience 2018

THE NEURAL BASIS OF IMAGE SEGMENTATION IN THE PRIMATE BRAIN

Anitha Pasupathy

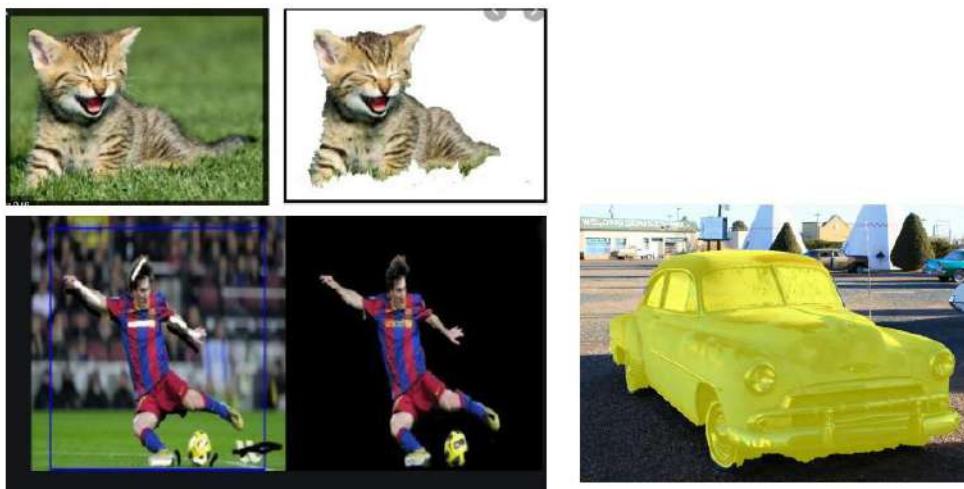
Our visual system rapidly and effortlessly segments scenes into component objects but the underlying neural basis is unknown. We studied single neurons in area V4 while monkeys discriminated partially occluded shapes. We found that many neurons tuned to boundary curvature maintained their shape selectivity over a large range of occlusion levels as compared to neurons that are not tuned to boundary curvature. This lends support to the hypothesis that segmentation in the face of occlusion may be solved by contour grouping.

e nevertheless perceive the world not as a uniform pixelated representation, but as a meaningful arrangement of objects and regions. This is achieved by a process called image segmentation which takes as its input the **continuous retinal representation** and parses it into components that ultimately underlie the percept that is the brain's best guess for the current state of the outside world. Image segmentation facilitates scene understanding and makes our interactions with the world around us more effective.



Segmentation of foreground vs background – Scontornamento delle immagini

L'approccio che vedremo da questo momento in più non assume più di avere immagini semplici e binarie. Si chiama infatti *interactive segmentation*.



FB segmentation è un approccio molto importante per la Computer Vision, ma anche difficile da gestire. Se poi affrontato senza l'aiuto della semantica, diventa un vero e proprio problema di unsupervised clustering.

Possiamo dividere gli algoritmi di FB segmentation in 3 macrocatergorie:

A. CLUSTERING METHODS IN TWO CLASSES

- Region splitting
- Region merging, clustering agglomerativo
- Superpixel method (IMPORTANTE)

B. K-means based

- K-means clustering + Gaussian (modello parametrico)
- Mean – shift

C. GRAPH BASED METHODS (importanti)

- Normalized cut
- Grabcut e graphcut

Mean shift

Utile per F/B Clustering, per multiple region segmentation e anche per tracking.

Ogni punto dello spazio di features viene interpretato come una densità di probabilità. Le regioni dense nello spazio delle features corrispondono a massimi locali o a mode. Per ogni data point, viene applicato il gradient descent sulla densità stimata fino alla convergenza. Il punto che converge rappresenta la moda della funzione di densità.

→ Tutti i punti associati allo stesso punto stazionario appartengono allo stesso cluster.

- The mean shift algorithm seeks *modes* of the given set of points

- Choose kernel and bandwidth h

Kernel density estimation function

- For each point:

- Center a window on that point

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

- Compute the mean of the data in the search window

$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}$$

- Center the search window at the new mean location

Gaussian kernel

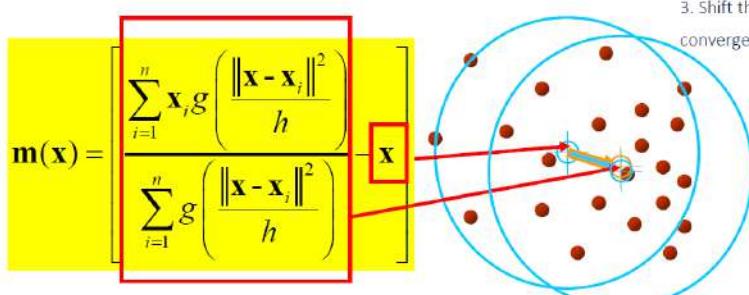
- Repeat (b,c) until convergence

- Assign points that lead to nearby modes to the same cluster

Questo algoritmo può essere utilizzato anche per semplice tracking, se applicato su frames di immagini successive.

Simple Mean Shift procedure:

- Fix a window around each data point.
- Compute the mean of data within the window.
- Shift the window to the mean and repeat till convergence.



Per approfondimenti:

<http://saravananthirumuruganathan.wordpress.com/2010/04/01/introduction-to-mean-shift-algorithm/>

Segmentation by graph-based methods

Quando le immagini sono complesse quindi non riconducibili a binarie si utilizzano dei grafici per poter segmentarle, si tratta di metodi interattivi.

L'obiettivo è quello di avere label binarie per il background e il foreground.

Approccio: ottimizzazione di un funzione di Energia (Labeling) = data + smoothness, la funzione deve essere minimizzata.

Dove:

- Data, per ogni pixel, la probabilità che appartenga a B o a F
- Smoothness (regolarizzazione), per coppie di pixel nello stesso neighborhood, si associa una penalità (abbassando i pesi) nel caso in cui i pixel siano di colori molto diversi

Paper di riferimento:

Yuri Boykov, Marie-Pierre Jolly Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images (ICCV 2001), vol. I, pp. 105-112, 2001

Shi Malik Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(22):888–905

C. Rother, V. Kolmogorov, A. Blake. GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts. ACM Transactions on Graphics (SIGGRAPH'04), 2004

Dall'immagine al grafico:

- Ad ogni pixel corrisponde un nodo
- Arco tra ogni coppia di pixel
- Calcolo di similarità per ogni arco che connette una coppia di pixel (w_{ij})

La similarità può essere sia a livello di colore che a livello di spazio all'interno dell'immagine.

Con un algoritmo di tipo Graph Cuts è possibile dividere il grafico in segmenti e quindi l'immagine in regioni.

Si taglano gli archi che hanno bassa similarità, pixels simili dovrebbero essere nello stesso segmento di grafico.

Gli algoritmi di segmentazione con i grafici:

1. Min Cut
2. Normalized Cut
3. Graphcut
4. Grabcut con Graphcut

Min Cut

Dato un grafico $G = \{V, E, W\}$, dove V è il nodo, E i suoi edges e W la matrice di affinità. La matrice di affinità associa un peso ad ogni edge in E .

Il taglio (cut) di un grafico è la partizione di V in due sottoinsieme A e B tali che $A \cup B = V, A \cap B = \emptyset$

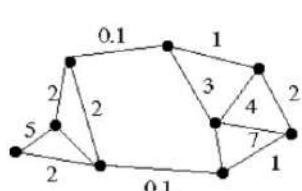
Il min cut di un grafico è il taglio che partiziona G in due segmenti disgiunti tali che la somma dei pesi associati agli edges tra i due segmenti è minima.

$$C_{min}(A, B) = \sum_{u \in A, v \in B} W_{uv}.$$

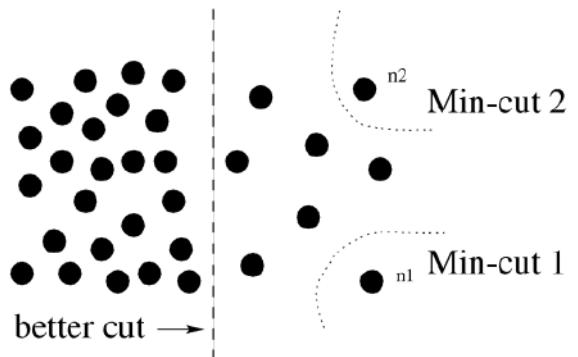
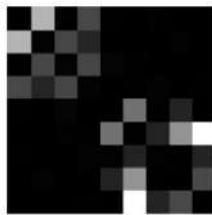
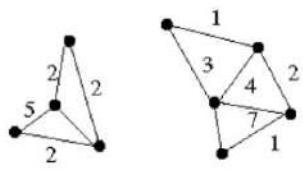
Ma si tratta di un problema NP-Hard.

Nonostante molti algoritmi siano in grado di trovare il taglio migliore o la migliore partizione. Min cut è in grado di selezionare l'edge con pesi minimi per effettuare il taglio e questo lo rende un buonissimo algoritmo.

Minimum cut example

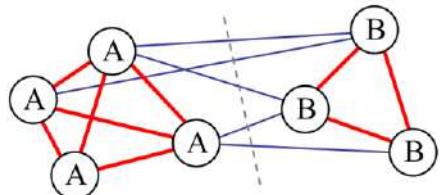


Minimum cut example



Normalized cut graph (NP complete)

Si suppongo due tipi di connessioni tra i nodi, forti e deboli. Deboli in blu, forti in rosso.



	A	B	sum
A	$assoc(A, A)$	$cut(A, B)$	$assoc(A, V)$
B	$cut(B, A)$	$assoc(B, B)$	$assoc(B, V)$
sum	$assoc(A, V)$	$assoc(B, v)$	

Il taglio in questo algoritmo penalizza i segmenti larghi.

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}, \quad cut(A, B) = \sum_{u \in A, v \in B} w(u, v).$$

$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$$

Dato $assoc(A/B, V)$ = somma dei pesi di tutti gli edges che toccano A o B, dove w è la similarità tra i pixels.

Dato il taglio $cut(A, B)$ come la somma di tutti i pesi dal nodo A a quello B.

COME CALCOLARE IL PESO TRA DUE PIXELS ?

I pesi possono essere ottenuti tramite un metodo supervised oppure definiti in maniera unsupervised come il prodotto di due fattori:

- **L'affinità in termini di colore**, preso un kernel gaussiano, considerando la distanza del vettore colore $F(i)$ e $F(j)$;
- **L'affinità in termini di prossimità spaziale**, se i pixels sono distanti più di una certa soglia definita r , per evitare calcoli inutili allora la prossimità è 0, oppure si definire un kernel Gaussiano per la misura della distanza

- Note that
- small sigma group only very near points
- Large sigma accept affinity of far away points

$$w_{ij} = e^{-\frac{\|F(i) - F(j)\|_2^2}{\sigma_f^2}} * \begin{cases} e^{-\frac{\|X(i) - X(j)\|_2^2}{\sigma_X^2}} & \text{if } \|X(i) - X(j)\|_2 < r \\ 0 & \text{otherwise.} \end{cases}$$

(L'algoritmo di normalizzazione graphcut ricorsivo può essere risolto usando gli autovalori della matrice D che contengono i pesi.)

Porzioni di paper che ha spiegato velocemente:

2.1 Computing the Optimal Partition

Given a partition of nodes of a graph, V , into two sets A and B , let x be an $N = |V|$ dimensional indicator vector, $x_i = 1$ if node i is in A and -1 , otherwise. Let $d(i) = \sum_j w(i, j)$ be the total connection from node i to all other nodes. With the definitions x and d , we can rewrite $Ncut(A, B)$ as:

$$\begin{aligned} Ncut(A, B) &= \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(B, A)}{assoc(B, V)} \\ &= \frac{\sum_{(x_i > 0, x_j < 0)} -w_{ij}x_i x_j}{\sum_{x_i > 0} d_i} \\ &\quad + \frac{\sum_{(x_i < 0, x_j > 0)} -w_{ij}x_i x_j}{\sum_{x_i < 0} d_i}. \end{aligned}$$

Let D be an $N \times N$ diagonal matrix with d on its diagonal, W be an $N \times N$ symmetrical matrix with $W(i, j) = w_{ij}$,

$$k = \frac{\sum_{x_i > 0} d_i}{\sum_i d_i},$$

RECURSIVE NORMALIZED CUTS

Our grouping algorithm consists of the following steps:

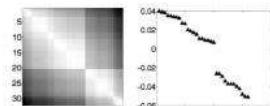
1. Given an image or image sequence, set up a weighted graph $G = (V, E)$ and set the weight on the edge connecting two nodes to be a measure of the similarity between the two nodes.
2. Solve $(D - W)x = \lambda Dx$ for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with the second smallest eigenvalue to bipartition the graph.
4. Decide if the current partition should be subdivided and recursively repartition the segmented parts if necessary.

Why is it connected with eigenvalues?
See Szelinski book 5.2

The weights depend on the distance in color (F) and in the spatial distance (X) if are close enough $< r$ otherwise 0

$$w_{ij} = e^{-\frac{\|F(i) - F(j)\|_2^2}{\sigma_f^2}} * \begin{cases} e^{-\frac{\|X(i) - X(j)\|_2^2}{\sigma_X^2}} & \text{if } \|X(i) - X(j)\|_2 < r \\ 0 & \text{otherwise.} \end{cases}$$

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}x = \lambda x.$$



Detail

<http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf>

(vedi esempi a slides 137 – 138)

Pro: si tratta di un framework generico, può essere usato con diverse features e formule di affinità, permette di avere dei segmenti regolari

Contro: bisogna scegliere il numero di segmenti, richiede un alto livello di storage e complessità di tipo NP Hard, bias per la partizione in segmenti uguali

Usage: nel caso di segmentazione in cui si vogliono segmenti regolari.

Gli algoritmi di segmentazione vogliono raggruppare pixels simili e vogliono distinguere tra i pixels che fanno parte da regioni differenti. L'idea è quella di calcolare le statistiche di appartenenza o meno ad una regione sommando i pixels che appartengono allo stesso neighborhood. È possibile formulare questo approccio tramite una funzione energetica sui pixels con un processo stocastico Markoviano.

(Si definisce processo stocastico markoviano (o di Markov), un processo aleatorio in cui la probabilità di transizione che determina il passaggio a uno stato di sistema dipende solo dallo stato del sistema immediatamente precedente (proprietà di Markov) e non da *come* si è giunti a questo stato. Viceversa si dice *processo non markoviano* un processo aleatorio per cui non vale la proprietà di Markov.)

L'idea di GrabCut è la seguente:

- a. L'utente applica una bounding box in rosso, tutto al di fuori della box è background;
- b. L'algoritmo determina la distribuzione di colori per l'oggetto e per il background, applica una binary segmentation;
- c. Il processo è ripetuto finché la distribuzione di colori migliora.

"GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts

Carsten Rother*

Vladimir Kolmogorov†
Microsoft Research Cambridge, UK

Andrew Blake‡



Come abbiamo visto GrabCut usa GraphCut.

Segue spiegazione Graphcut.

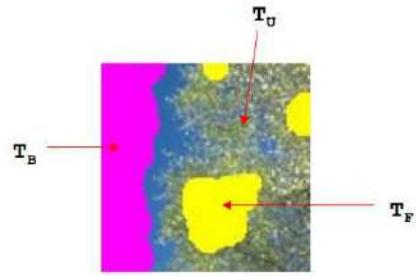
GraphCut

GraphCut è una generalizzazione della normalized cut che usa istogrammi.

- The image is an array $z = (z_1, \dots, z_N)$ of grey values indexed by the single index n .
- The segmentation of the image is an alpha-channel, or, a series of opacity values $\alpha = (\alpha_1, \dots, \alpha_N)$ at each pixel with $0 \leq \alpha_n \leq 1$.
- The parameter θ describes the foreground/background grey-level distributions. i.e. a pair of histogram of gray values:

$$\theta = \{h(z; \alpha), \alpha = 0, 1\} \iff \begin{array}{c} \text{Histogram of } z \\ \text{at } \alpha = 0 \\ \text{at } \alpha = 1 \end{array}$$

User provides a trimap $T = \{T_F, T_B, T_U\}$ which partitions the image into 3 regions: foreground, background, unknown.



- Note that these histograms are directly assembled from the trimaps T_B and T_F
- Re-pose the segmentation task:
➤ The segmentation task is to infer the unknown opacity values α from image z and the model θ .

Segmentation come la minimizzazione di energia: una funzione di energia E è definita così che il suo minimo corrisponda a una buona segmentazione.

Si utilizza la formula della Gibbs Energy

$$E(\alpha, \theta, z) = U(\alpha, \theta, z) + V(\alpha, z)$$

Dove U valuta l'adattamento dell'opacità α ai dati z (cioè dà un buon punteggio (punteggio basso) se α sembra che sia coerente con l'istogramma).

$$U(\alpha, \theta, z) = \sum_n -\log h(z_n; \alpha_n)$$

V è una termine di smoothness che penalizza nel caso in cui ci sia troppa disparità tra pixels dello stesso neighborhood.

$$V(\alpha, z) = \gamma \sum_{(m,n) \in C} dis(m,n)^{-1} [\alpha_m \neq \alpha_n] \exp -\beta (z_m - z_n)^2,$$

$$E(\alpha, \theta, z) = U(\alpha, \theta, z) + V(\alpha, z)$$

Minizzare l'energia, definendo alfa uguale a 0 ad 1.

- Given the energy model we can obtain a segmentation by finding

$$\alpha = \arg \min_{\alpha} E(\alpha, \theta)$$

- Which can be solved using a minimum cut algorithm which gives you a hard segmentation, $\alpha = \{0, 1\}$, of the object.

Risolvere un MRF (Markovian random field) con graph cuts

Si aggiungo due nodi al grafo:

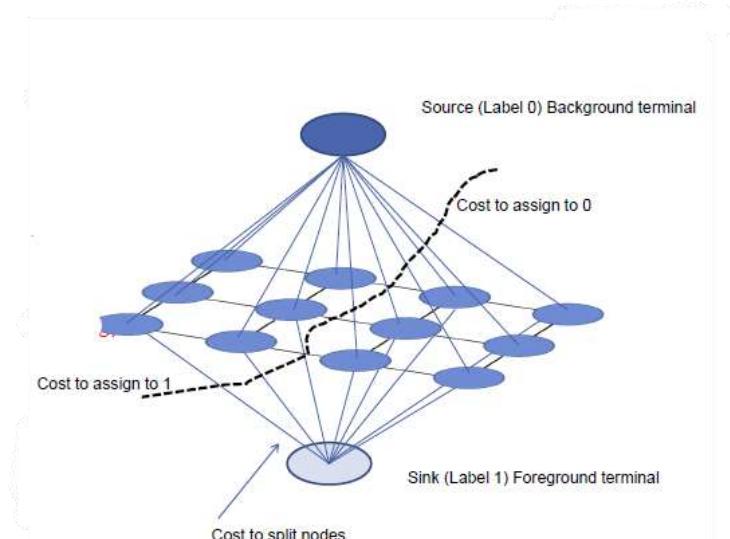
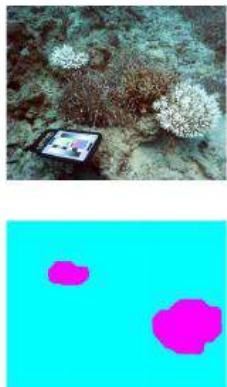
- Il nodo **source** (Background) S
- Il nodo **sink** (Foreground) F

I pixels che sono più compatibili con il foreground o con il background presentano connessioni più forti al rispettivo nodo source o sink. Pixels adiacenti con smoothness più alta presentano anche essi connessioni più forti.

(non ci ho capito molto metto link per slides di università Illinois da cui la prof ha rubato queste)

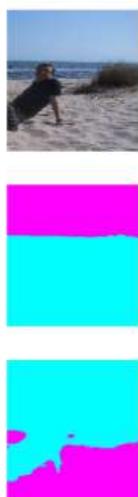
https://courses.engr.illinois.edu/cs543/sp2011/lectures/Lecture%2012%20-%20MRFs%20and%20Graph%20Cut%20Segmentation%20-%20Vision_Spring2011.pdf

- Bold are the selected initially



Graph cut

- If there is no a single foreground



The edge weight between pixel i and j will be denoted W_{ij}^I and the terminal weights between pixel i and the source (s) and sink (t) as W_i^s and W_i^t respectively and are given by

$$W_{ij}^I = e^{(-\frac{r(i,j)}{\sigma_R})} e^{(-\frac{\|\mathbf{w}(i)-\mathbf{w}(j)\|^2}{\sigma_W^2})} \quad (2)$$

$$W_i^s = \frac{p(\mathbf{w}(i)|i \in s)}{p(\mathbf{w}(i)|i \in s) + p(\mathbf{w}(i)|i \in t)} \quad (3)$$

$$W_i^t = \frac{p(\mathbf{w}(i)|i \in t)}{p(\mathbf{w}(i)|i \in s) + p(\mathbf{w}(i)|i \in t)} \quad (4)$$

Here $\|\cdot\|$ denotes the euclidian norm, $r(i,j)$ the distance between pixel i and j and λ , σ_R and σ_W are tuning parameters weighing the importance of the different features. Hence, W_{ij}^I contains the inter-pixel similarity, that ensures that the segmentation more coherent. W_i^s and W_i^t describes how likely a pixel is to being background and foreground respectively.

Grabcut

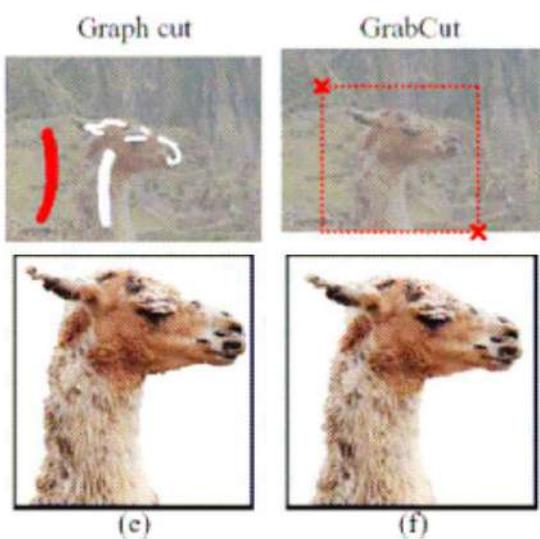
Il Grabcut come prima cosa specifica solo il background dall'uso di una bounding box.



[Specifying background only]

Partendo dalla bounding box inizializza tutto quello che ne è al di fuori come background, quello che è presente all'interno della bounding box deve essere classificato tra background e foreground pixel per pixels.

Non viene più utilizzato il modello ad istogramma ma il Gaussian Mixture Model (GMM). La procedura è iterativa.



Dato che ogni pixel z_n è in RGB è preferibile utilizzare al posto degli istogrammi un Gaussian Mixture Model (GMM). In particolare si utilizzano 2 modelli Gaussiani con full-covariance e k componenti, due perché uno è per il background e l'altro per il foreground.

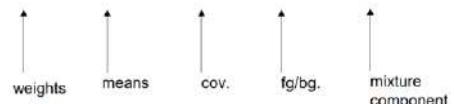
Si aggiunge al modello di Graphcut un vettore $\mathbf{k} = \{k_1 \dots k_n\}$, dove k_i è in $\{1 \dots K\}$.

k_i assegna al pixel z_i ad uno dei componenti GMM (F o B).

Il modello si complica.

Our θ becomes

$$\theta = \{\pi(a, k), \mu(a, k), \Sigma(a, k), a=0, 1, k=1 \dots K\}$$



- Must incorporate \mathbf{k} into our model:

$$E(\alpha, \mathbf{k}, \theta, z) = U(\alpha, \mathbf{k}, \theta, z) + V(\alpha, z)$$

where

$$U(\alpha, \mathbf{k}, \theta, z) = \sum_n D(\alpha_n, k_n, \theta, z_n)$$

- $D(\alpha_n, k_n, \theta, z_n) = -\log p(z_n | \alpha_n, k_n, \theta) - \log \pi(\alpha_n, k_n)$
- Where $\pi(\cdot)$ is a set of mixture weights which satisfy the constraint:

$$D(\alpha_n, k_n, \theta, z_n) = -\log \pi(\alpha_n, k_n) + \frac{1}{2} \log \det \Sigma(\alpha_n, k_n) + \frac{1}{2} [z_n - \mu(\alpha_n, k_n)]^\top \Sigma(\alpha_n, k_n)^{-1} [z_n - \mu(\alpha_n, k_n)].$$

GrabCut



Variables

- x_i : pixel
 - $y_i \in \{0, 1\}$: foreground/background label {0, 1}
 - $k_i \in \{0, \dots, K-1\}$: GMM mixture component
 - θ : GMM model parameters
 - $I = \{z_1, \dots, z_m\}$: RGB image
- Unary Term $\varphi(x_i, y_i, \theta, k_i)$
- log of GMM probability

Pairwise Term

$$\psi(y_i, y_j, x_i, x_j) = \gamma[y_i \neq y_j] \exp(-\beta \|x_i - x_j\|^2)$$

$$E(\alpha, \mathbf{k}, \theta, z) = U(\alpha, \mathbf{k}, \theta, z) + V(\alpha, z)$$

$$E(x, y, \theta, k) = \sum_i \varphi(x_i, y_i, \theta, k_i) + \sum_{ij} \psi(y_i, y_j, x_i, x_j)$$

Step by step:

1. Definisci il grafo (di solo è four-connected, o a volte eight-connected);
2. Inizializzazione: definire i potenziali unari (Color histogram o mixture Gaussian model per background e foreground);
3. Definire potenziali a coppie;
4. Applica il graphcuts;
5. Ritorna al 2, usando le label correnti per calcolare foreground o background.

Di solito Grabcut itera dalle 5 alle 10 volte.

Summary of Algorithm



CERCA DI CAPIRE LA DIFFERENZA TRA UNARY
E PAIRWISE POTENTIALS

- Start with T_B and T_U as inputs.
- We use the input to initialize the foreground and background GMMs
- Using the GMM's we solve an energy minimization problem via min-cut, which corresponds to an initial segmentation of the object, T'_B and T'_U .
- Repeat the procedure with T'_B and T'_U as inputs until convergence ($T_F = T'_U$).
- User interaction then repeat either min-cut or all steps.

Initialisation

- User initialises trimap T by supplying only T_B . The foreground is set to $T_F = 0$; $T_U = \bar{T}_B$, complement of the background.
- Initialise $\alpha_n = 0$ for $n \in T_B$ and $\alpha_n = 1$ for $n \in T_U$.
- Background and foreground GMMs initialised from sets $\alpha_n = 0$ and $\alpha_n = 1$ respectively.



Details Minimization



1. Each pixel of the unknown group is assigned to one gaussian (iterating through all 1..K, since K si small)

1. Assign GMM components to pixels: for each n in T_U ,

$$k_n := \arg \min_{k_n} D_n(\alpha_n, k_n, \theta, z_n).$$

2. Learn GMM parameters from data z :

$$\underline{\theta} := \arg \min_{\underline{\theta}} U(\underline{\alpha}, \underline{k}, \underline{\theta}, \underline{z})$$



- For example, to find the Gaussian parameters for a component k in the foreground:

- Find the set of pixels Z_k assigned to k by the k -vector.
- Find μ, Σ , in the standard fashion.
- Update the weights at $\pi(a, k) := |Z_k| / \sum_k |Z_k|$

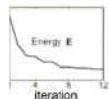
Details Estimate segmentation

3. *Estimate segmentation:* use min cut to solve:

$$\min_{\{\alpha_n; n \in T_U\}} \min_k E(\underline{\alpha}, k, \underline{\theta}, \underline{z}).$$

4. Repeat from step 1, until convergence.

- Now that k and θ are known, we can solve for the opacity values using a minimum-cut algorithm, and reapply the min-cut until convergence.
- Each iteration eats away at the unknown region and continues to minimize the energy function.



Pro:

- È molto veloce a livello di inferenza;
- Può essere usato per recognition o high-level priors;
- Si applica ad un largo range di problemi (stereo, image labeling, recognition);

Contro:

- Non si può applicare in tutti i casi (solo quelli associativi);
- A bisogno di termini unari (non utilizzabile per segmentazione generica).

La questione è che nel caso di molte immagini anche metodi efficienti possono risultare lenti. Una soluzione è Superpixels.

Superpixels segmentation



Partendo dai pixel che compongono un'immagine, un superpixel può essere definito come un gruppo di pixels che condividono caratteristiche comuni (intensità, posizione, ...).

L'uso di superpixels risulta utile nell'ambito della computer vision per diverse ragioni:

- Presentano maggiori informazioni dei pixels classici;
- Presentano un significativo percettivo, cioè condividono proprietà visive;
- Permettono di ottenere una rappresentazione delle immagini compatta e conveniente nel caso di problemi computazionali;

Gli aspetti negativi dell'uso di questo metodo di segmentazione sono che il calcolo dei superpixels è a sua volta computazionalmente pesante, inoltre c'è il rischio di perdere informazioni importanti in merito a edges.

L'algoritmo più noto che usa superpixels è SLIC. **SLIC**, Simple Linear Iterative Clustering, non necessita di grosse risorse computazionali. L'idea è quella di utilizzare clustering per andare a raggruppare i pixels che hanno caratteristiche simili in termini di prossimità spaziale e colore.

Per quanto riguarda il colore, non viene più espresso in RGB ma in CIELAB.

Lo Spazio colore Lab o CIELAB o CIE 1976 (L^* , a^* , b^*) è uno spazio colore-opponente con la dimensione L per la luminosità a e b per le dimensioni colore-opponente, basato sulle coordinate dello spazio colore non lineare compresso CIE XYZ. Le coordinate di Hunter 1948 sono quindi L , a e b .

SLIC presenta un spazio five-dimensional:

- L, a, b i valori di CIELAB;
- x, y le coordinate dei pixels.

Il problema è che la distanza tra due colori di pixels nello spazio CIELAB è limitata, mentre la distanza xy spaziale tra due pixels dipende dalla dimensione dell'immagine. Per questa ragione non è possibile utilizzare la distanza Euclidea in uno spazio 5D così definito, è necessario normalizzare la distanza spaziale. Per poter clusterizzare pixels in uno spazio 5D, viene introdotta una nuova misura di distanza che considera la dimensione dei superpixels.

SLIC prende come input K , un numero di superpixels che hanno approssimativamente la stessa dimensione. Per un'immagine con N pixels la dimensione approssimata di ogni superpixels è N/K pixels. Allora per ogni superpixels di uguale dimensione, ci sarà un superpixel al centro di ogni griglia con $S=\sqrt{N/K}$.

All'inizio di ogni algoritmo si prendono K superpixels con centro del cluster $C_k = [l_k, a_k, b_k, x_k, y_k]$ con $k = [1, K]$ preso S ad intervalli regolari nella griglia. Allora l'area approssimata di ogni superpixels è pari a S^2 . I pixels associati ad ogni cluster center sono in un area $2S \times 2S$ intorno al superpixel centrale sul piano xy .

L'area $2S \times 2S$ è l'area di ricerca per i pixels che sono più vicini ad ogni centro del cluster.

- D_s is the sum of the Lab distance
- the xy plane distance normalized by the grid interval S .
- A variable m is introduced in D_s allowing us to control the compactness of superpixel.

Distance measure D_s is defined as follows.

$$d_{\text{lab}} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$$

$$D_s = d_{\text{lab}} + (m / S) * d_{xy}$$

- The greater the value of m , the more spatial proximity is emphasized and the more compact the cluster.
- This value can be in the range [1, 20]. Authors of the algorithm have chosen $m=10$.

PROCESSO ITERATIVO:

1. Presi K cluster centers con approssimativamente uguale dimensione;
 2. Disporli sul piano in posizioni corrispondenti al valore di gradiente minimo (più basso) in un neighborhood 3x3. (questo passaggio viene fatto per evitare di posizionare il centro del cluster sugli edge);
Image gradients are computed as

$$G(x, y) = || I(x+1, y) - I(x-1, y) ||^2 + || I(x, y+1) - I(x, y-1) ||^2$$

where $I(x, y)$ is the Lab vector corresponding to the pixel at position (x, y) , and $||.||$ is the L2 norm.
This takes into account both color and intensity information.
 3. Ogni pixel viene associato al center cluster più vicino, il pixel in questione deve essere contenuto nell'area di ricerca;
 4. Ogni volta che un pixel viene associato ad un cluster center il centro viene di nuovo ricalcolato come la media dei vettori Labxy di tutti i pixels associati a quel cluster.
- SLIC is a k-means clustering performed on image pixels in a five dimensional position and color space

Algorithm 1 Efficient superpixel segmentation

```

1: Initialize cluster centers  $C_k = [l_k, a_k, b_k, x_k, y_k]^T$  by sampling pixels at regular grid steps  $S$ .
2: Perturb cluster centers in an  $n \times n$  neighborhood, to the lowest gradient position.
3: repeat
4:   for each cluster center  $C_k$  do
5:     Assign the best matching pixels from a  $2S \times 2S$  square neighborhood around the cluster center according to the distance measure (Eq. 1). →
6:   end for
7:   Compute new cluster centers and residual error  $E$  {L1 distance between previous centers and recomputed centers}
8: until  $E \leq$  threshold
9: Enforce connectivity.

```

Pixel-Superpixel association: Associate each pixel to the nearest superpixel center in the five-dimensional space, i.e., compute the new superpixel assignment at each pixel p ,

$$H_p^t = \arg \min_{i \in \{0, \dots, m-1\}} D(I_p, S_i^{t-1}), \quad (1)$$

$S_i^t = \frac{1}{Z_i^t} \sum_{p \in H_p^t} I_p$, Zit is K at the iteration t
Sit is the centroid Ck at the iteration t

http://www.kev-smith.com/papers/SLIC_Superpixels.pdf

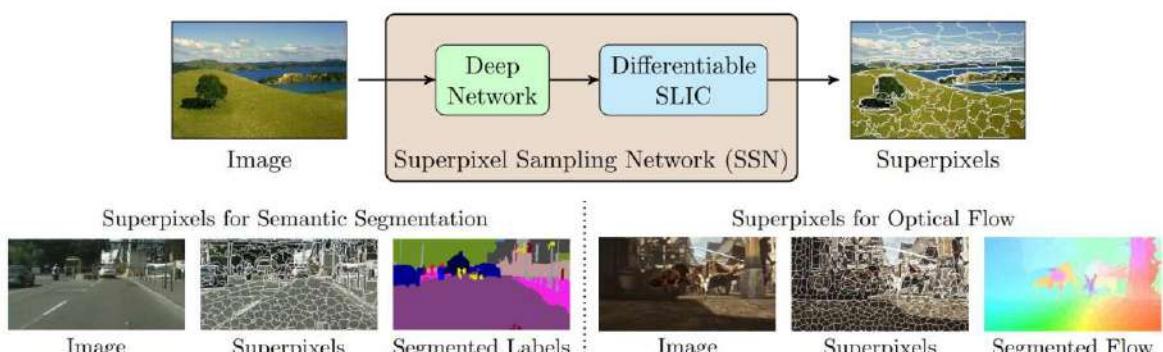
N	Number of pixels in the input image
K	Number of Superpixels used to segment the input image
N/K	Approximate size of each superpixel
$S = \sqrt{N}/K$	For roughly equally sized superpixels there would be a superpixel centre at every grid interval S



Look at <https://www.youtube.com/watch?v=-hmUbB-Y8R0>

Lorenzo Baraldi ha fatto la tesi sui superpixels, guardare se interessati slides 166.

L'evoluzione di SLICN è connessa alla volontà di usare deep learning in questo tipo di approcci. **Superpixel sampling network**, definisce le features associando una Deep Network ad ogni pixels (invece di CIELABxy). La differenziabilità non è più ottenuta usando *nearest neighborhood* nel clustering, ma ottenendo una loss differenziabile con SLIC.



Funziona poco meglio, molto utile per la semantic segmentation o l'optical flow.

La ricerca per SSN è andata avanti:

http://openaccess.thecvf.com/content_ECCV_2018/papers/Varun_Jampani_Superpixel_Sampling_Networks_ECCV_2018_paper.pdf

Da qui in poi attacco slides di confronto per l'evoluzione delle SSN ma vuole sapere solo le linee generali.

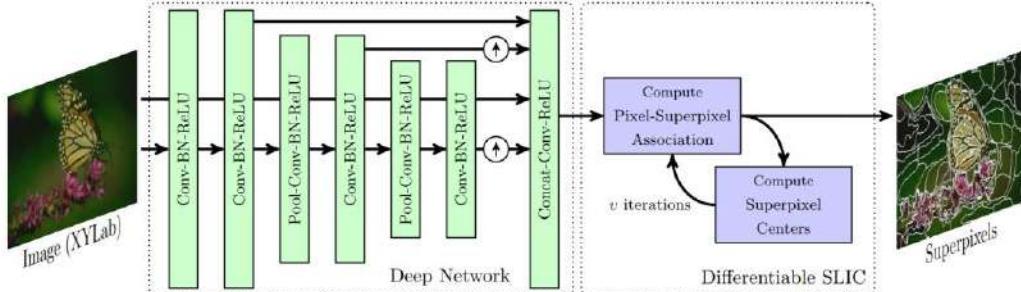


Fig. 3: **Computation flow of SSN.** Our network is composed of a series of convolution layers interleaved with Batch Norm (BN) and ReLU nonlinearities. \uparrow denotes bilinear upsampling to the original image resolution. The features from CNNs are then passed onto iterative updates in the differentiable SLIC to generate superpixels.

New results (CVPR2020)



CNN problems having in input superpixels

Two networks to high resolution
And good boundary
tested on BSDS500 and NYUv2

Superpixel Segmentation with Fully Convolutional Networks

Fengting Yang Qian Sun
The Pennsylvania State University
fuy34@psu.edu, uestcqs@gmail.com

Hailin Jin
Adobe Research
hjin@adobe.com

Zihan Zhou
The Pennsylvania State University
zzhou@ist.psu.edu

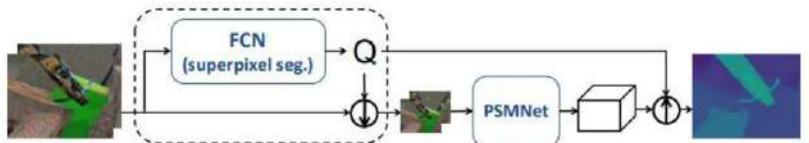
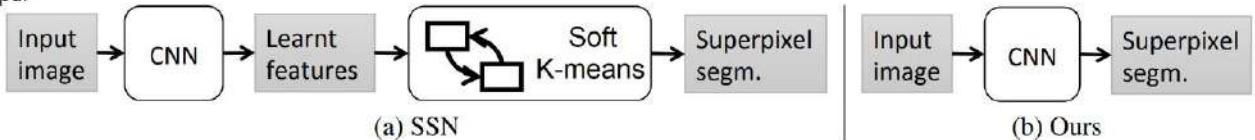


Figure 1. An illustration of our superpixel-based downsampling/upsampling scheme for deep networks. In this figure, we choose PSMNet [7] for stereo matching as our task network. The high-res input images are first downsampled using the superpixel association matrix Q predicted by our superpixel segmentation network. To generate a high-res disparity map, we use the same matrix Q to upsample the low-res disparity volume predicted by PSMNet for final disparity regression.

https://openaccess.thecvf.com/content_CVPR_2020/papers/Yang_Superpixel_Segmentation_With_Fully_Convolutional_Networks_CVPR_2020_paper.pdf



- To use DNN
- You must have a differentiable assignment

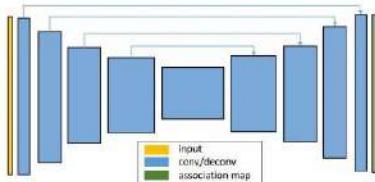


Figure 4. Our simple encoder-decoder architecture for superpixel segmentation. Please refer to the supplementary materials for detailed specifications.

“seed”). Then, the final superpixel segmentation is obtained by finding a mapping which assigns each pixel $\mathbf{p} = (u, v)$ to one of the seeds $\mathbf{s} = (i, j)$. Mathematically, we can write the mapping as $g_s(\mathbf{p}) = g_{i,j}(u, v) = 1$ if the (u, v) -th pixel belongs to the (i, j) -th superpixel, and 0 otherwise.

Superpixel with fully cnn

$\mathbf{F}(\mathbf{p})$ is the property of the pixel, it could be the CIELAB vector or an N dimensional one-hot encoding vector as a semantic label (to achieve superpixel semantic regions)

N is the set of surrounding superpixels and $q(\mathbf{p})$ is the predicted probability to be assigned to s

Thus for each \mathbf{p} the reconstructed $\mathbf{f}'(\mathbf{p})$ property and location

Finally the loss is given by two terms as in SLIC

One the distance in the property and the second is a measure of compactness and m is a weight and S is the sampling interval

Two loss functions

Measures
achievable segmentation accuracy (ASA),
boundary recall and precision (BR,BP),
compactness (CO).

ASA quantifies the achievable accuracy for segmentation using the superpixels as preprocessing step, BR and BP measure the boundary adherence of superpixels given the ground truth, whereas CO assesses the compactness of superpixels.

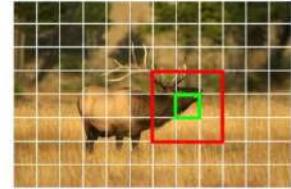


Figure 2. Illustration of \mathcal{N}_p . For each pixel \mathbf{p} in the green box, we consider the 9 grid cells in the red box for assignment.

ing a deep neural network. To make our objective function differentiable, we replace the hard assignment G with a soft association map $Q \in \mathbb{R}^{H \times W \times |\mathcal{N}_p|}$. Here, the entry $q_s(\mathbf{p})$ represents the probability that a pixel \mathbf{p} is assigned to each $s \in \mathcal{N}_p$, such that $\sum_{s \in \mathcal{N}_p} q_s(\mathbf{p}) = 1$. Finally, the superpixels are obtained by assigning each pixel to the grid cell with the highest probability: $\mathbf{s}^* = \arg \max_s q_s(\mathbf{p})$.



Given the predicted association map Q , we can compute the center of any superpixel s , $\mathbf{c}_s = (\mathbf{u}_s, \mathbf{l}_s)$ where \mathbf{u}_s is the property vector and \mathbf{l}_s is the location vector, as follows:

$$\mathbf{u}_s = \frac{\sum_{\mathbf{p}:s \in \mathcal{N}_p} \mathbf{f}(\mathbf{p}) \cdot q_s(\mathbf{p})}{\sum_{\mathbf{p}:s \in \mathcal{N}_p} q_s(\mathbf{p})}, \quad \mathbf{l}_s = \frac{\sum_{\mathbf{p}:s \in \mathcal{N}_p} \mathbf{p} \cdot q_s(\mathbf{p})}{\sum_{\mathbf{p}:s \in \mathcal{N}_p} q_s(\mathbf{p})}. \quad (1)$$

Given \mathbf{p} and \mathbf{p}' :

$$\mathbf{f}'(\mathbf{p}) = \sum_{s \in \mathcal{N}_p} \mathbf{u}_s \cdot q_s(\mathbf{p}), \quad \mathbf{p}' = \sum_{s \in \mathcal{N}_p} \mathbf{l}_s \cdot q_s(\mathbf{p}). \quad (2)$$

$$L(Q) = \sum_{\mathbf{p}} \text{dist}(\mathbf{f}(\mathbf{p}), \mathbf{f}'(\mathbf{p})) + \frac{m}{S} \|\mathbf{p} - \mathbf{p}'\|_2, \quad (3)$$



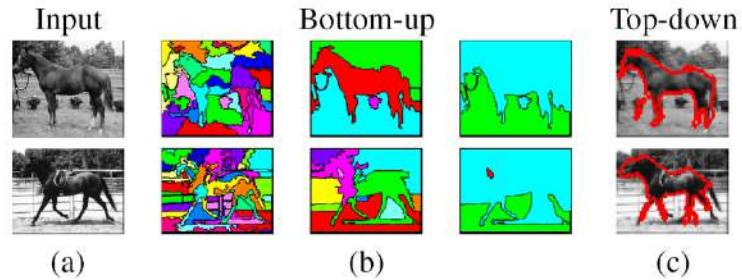
In this paper, we consider two different choices of $\mathbf{f}(\mathbf{p})$. First, we choose the CIELAB color vector and use the ℓ_2 norm as the distance measure. This leads to an objective function similar to the original SLIC method [1]:

$$L_{SLIC}(Q) = \sum_{\mathbf{p}} \|\mathbf{f}_{col}(\mathbf{p}) - \mathbf{f}'_{col}(\mathbf{p})\|_2 + \frac{m}{S} \|\mathbf{p} - \mathbf{p}'\|_2. \quad (4)$$

Second, following [16], we choose the one-hot encoding vector of semantic labels and use cross-entropy $E(\cdot, \cdot)$ as the distance measure:

$$L_{sem}(Q) = \sum_{\mathbf{p}} E(\mathbf{f}_{sem}(\mathbf{p}), \mathbf{f}'_{sem}(\mathbf{p})) + \frac{m}{S} \|\mathbf{p} - \mathbf{p}'\|_2. \quad (5)$$

- It can be done in bottom up or top down manner
 - Bottom-up: group tokens with similar features
 - Top-down: group tokens that likely belong to the same object



[Levin and Weiss 2006]

Segmentation senza l'uso della semantica è equivalente a fare raggruppamento. Può essere fatta a livello di immagine o di regioni all'interno dell'immagine. Superpixels è un approccio utile nel caso di region proposal e object detection.

SALIENCY

Ultimo argomento di questo pacco di slides.

Visual saliency è una qualità percettiva e soggettiva che fa risaltare alcuni elementi del mondo che vediamo rispetto a quelli a loro vicini, cioè gli elementi salienti sono quelli che attirano l'attenzione visiva di chi guarda.

La saliency è un concetto che direziona gli studi nell'ambito della visual intelligence. Se prima l'interesse era rivolto a come potere vedere quello che una persona vede, con la saliency ci si sposta su cosa una persona vede. Con il cosa si intende, quello che in un'immagine o in un contesto attira l'attenzione visiva.

Slide 212 213 esempio.

Quando un essere umano guarda un'immagine o un video, non focalizza il suo sguardo su tutte le regioni dell'immagine o del video con la stessa intensità, ma viene invece attratto dalle regioni più salienti e rilevanti della scena. Questo problema, chiamato predizione della salienza di immagini e video, è stato studiato per molti anni nell'ambito delle neuroscienze e dell'intelligenza artificiale risultando in diversi algoritmi in grado di emulare il focus dell'attenzione degli esseri umani.



Treisman&Gelade A feature-integration theory of attention Cogn. Psych. 1980

Koch & Ulmann, Shift in selective visual attention: toward the underlying neural circuitry Hum. Neurobiol., 1984

Dagli studi di Itti e Koch (2001) e da Fecteau e Munoz (2006).

Il modello di visibilità globale di una scena viene catturato in una stimulus-salience map che evidenzia parti della scena che sono oggetto di attenzione perlopiù in merito a stimoli. (Itti e Koch) L'elaborazione neurale delle informazioni visive è influenzata da fattori cognitivi topdown, come l'aspettativa, la memoria o l'attenzione selettiva. Questi fattori cognitivi creano una motivational salience map che una volta integrata con la stimulus-salience map forma la priority map che guida il comportamento visuomotorio.

(dalle slides)

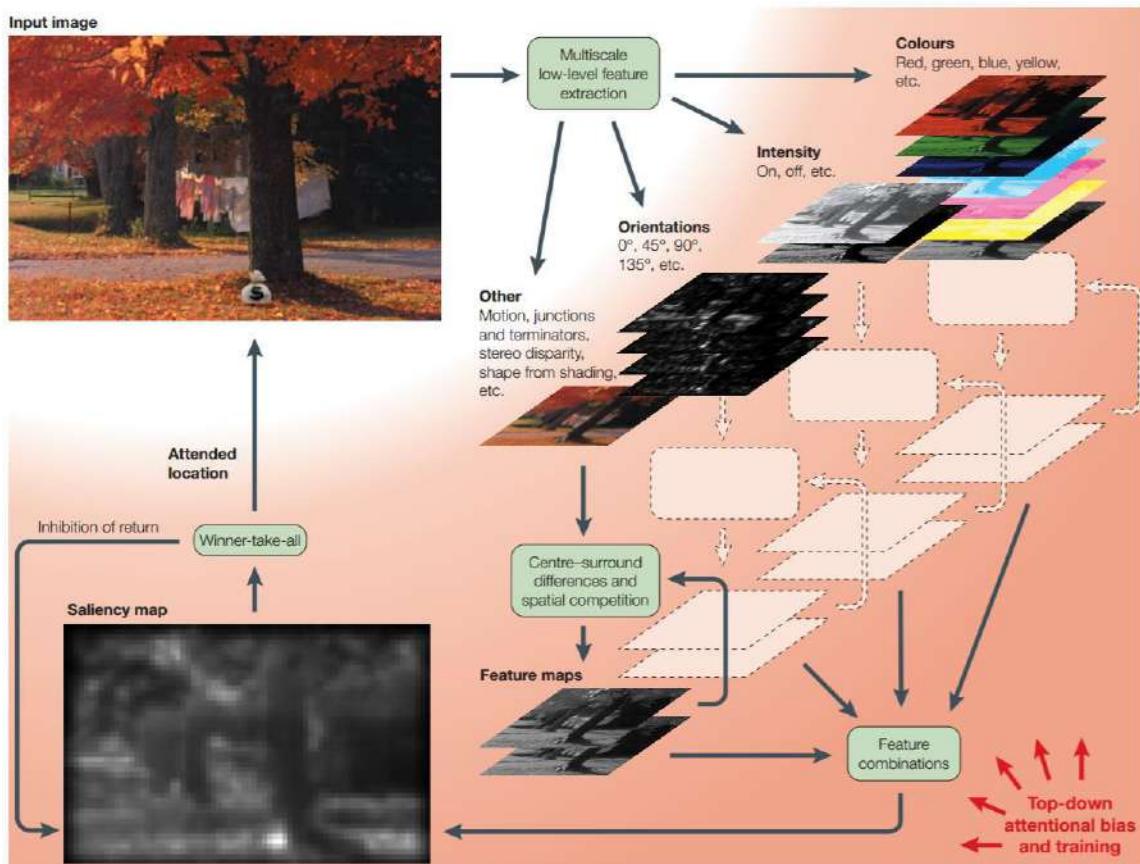
- **Saliency map**, calcola dei punti sull'immagine evidenziandoli, il calcolo avviene basandosi su features visive di basso livello come i colori più vivaci, gli edges orientati e la motion;
- **Priority map**, in questa mappa vengono integrate le informazioni ottenute con bottom-up nella saliency map con quelle relative al task e all'obiettivo iniziali;

Attention:*

- 'graded attention' representations (pre-selection)
- 'attentional target' representations (post-selection).
- overt attention (i.e. the point in visual space that is being fixated)

* R. Veale, Z. Hafed, M. Yoshida How is visual salience computed in the brain? Insights from behaviour, neurobiology and modelling [Philos Trans R Soc Lond B Biol Sci.](#) 2017

Itti e Koch



Il modello di riferimento è Itti Koch:

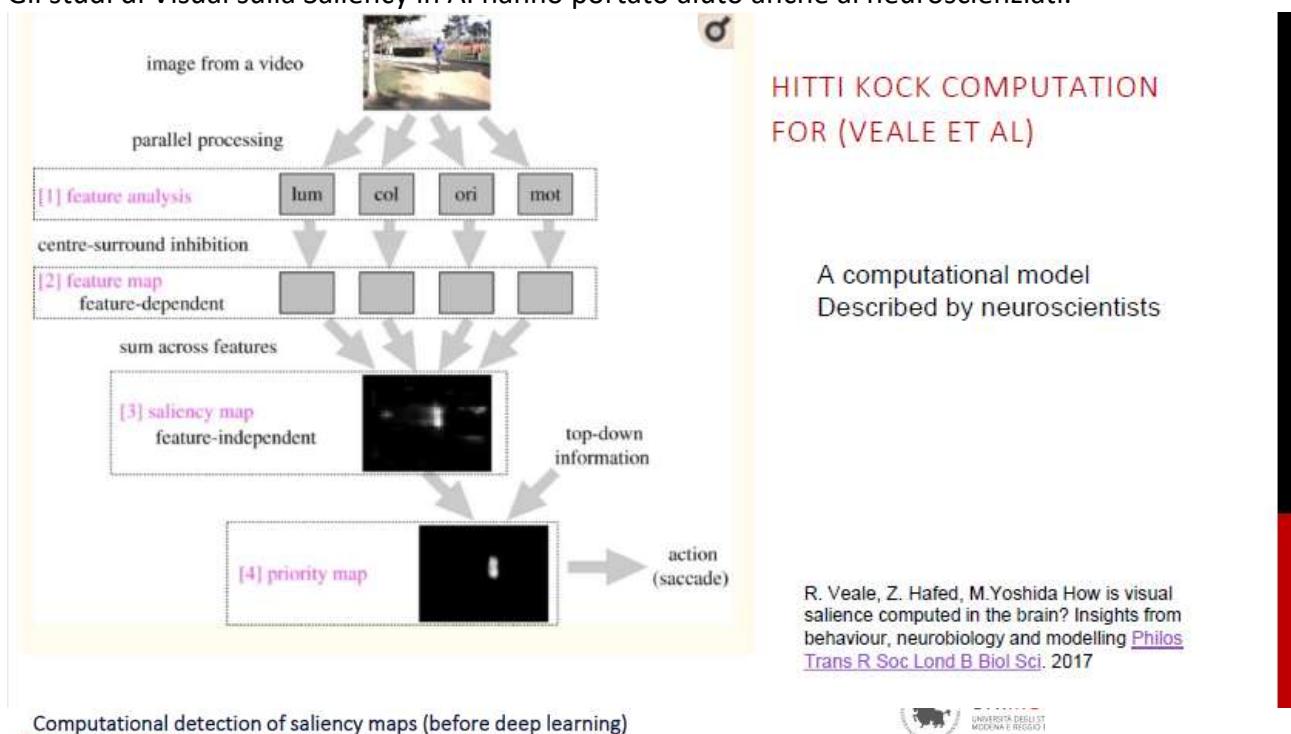
- i. Le mappe di features rappresentano features visive di base come colore, orientazione, luminosità e motion, si determina tramite queste features quali punti sulla mappa sono "differenti" rispetto a quelli che li circondano su diverse scale spaziali;
- ii. Una volta che si è normalizzato il tutto allora si ottiene la prima saliency map.

La saliency map è poi usata per determinare i target di attenzione.

Dagli studi di Veale et al.

Un modello di salience computazionale descrive fattori visivi di basso livello come colore, orientazione, illuminazione e motion che sono combinati in una singola global map, questa mappa rappresenta la 'salience' relativa ad ogni punto della mappa.

Gli studi di Visual sulla Saliency in AI hanno portato aiuto anche ai neuroscienziati.



- **LOW LEVEL FEATURES**

- '80—2000 **Itti Koch**: combination color+gradient+orientation in a winner-take-all unsupervised neural network
- 2006 NIPS **Perona et al.** Graph-based Visual Saliency as a graph of low level features
- ...

- **ADDING MEMORY and knowledge-based higher level FEATURES** (Faces, people, text..)

- 2009 ICCV Torralba et al
- 2012 ICPR Biorg ICPR
- 2013 ICCV Sclaroff et al.

(vedi per completezza ma non importanti 221 – 232)

Almage Lab work on Saliency → SAM

Marcella Cornia, L. Baraldi, G. Serra, and R. Cucchiara. A DeepMulti-Level Network for SaliencyPrediction, Proc. of ICPR 2016. (beforeatCVPRWoV2016)

(Rivedere lezione da minuto 1:37 fino a fine 19/05 – tot 20 min)

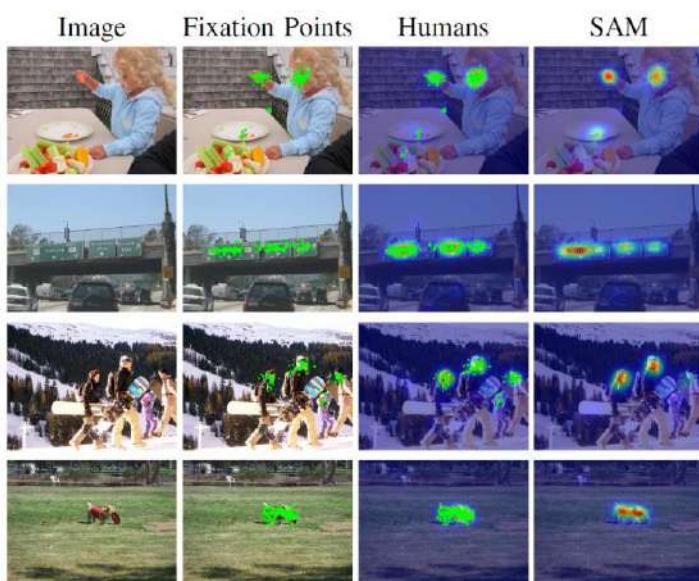
Slides da 236 a 263

Dal web:

Il lavoro presentato introduce un nuovo modello per la predizione della salienza su immagini basato interamente su tecniche di Deep Learning. Nel dettaglio, il modello proposto, chiamato Saliency Attentive Model (SAM), è composto da una rete convolutiva ricorrente che rifinisce la mappa di salienza predetta in modo iterativo grazie a meccanismi neurali attivati che identificano le aree più importanti della scena. Inoltre, l'architettura introdotta tiene conto di un'importante proprietà dello sguardo degli esseri umani: quando l'immagine non contiene regioni particolarmente rilevanti, gli esseri umani tendono a focalizzarsi maggiormente sul centro della scena portando così ad avere un forte bias centrale nelle mappe di salienza. Per incorporare questo bias centrale, SAM integra una serie di "prior map", imparate durante la fase di addestramento, che permettono di pesare maggiormente il centro della scena durante la predizione delle mappe di salienza.

Come la maggior parte dei modelli basati su Deep Learning, SAM è stato addestrato su grandi quantità di dati in cui ogni immagine è disponibile con la corrispettiva mappa di salienza ottenuta mediando le osservazioni di un numero consistente di persone. Tipicamente, questi dati vengono raccolti grazie a sistemi di eye-tracking che registrano i punti di fissazione degli esseri umani durante l'osservazione dell'immagine e dai quali è poi possibile stimare la mappa di salienza per quella immagine.

Per valutare le prestazioni del modello proposto, sono stati fatti numerosi esperimenti su diversi benchmark pubblici per la predizione della salienza su immagini. In particolare, SAM è stato testato sul dataset più grande disponibile in letteratura per questo task, SALICON, e sul MIT Saliency Benchmark che contiene una grande diversità tra le immagini che lo compongono ed è il benchmark di maggiore riferimento nell'ambito della predizione della salienza. I risultati sperimentali hanno dimostrato che il modello proposto è stato in grado di superare largamente lo stato dell'arte in questo task, posizionandosi al primo posto nelle classifiche di entrambi i benchmark considerati.



<https://arxiv.org/pdf/1611.09571.pdf> (intero Papers di Marcella Cornia)

