



# ELABORAZIONE DELLE IMMAGINI

SEGMENTAZIONE DELLE IMMAGINI: CLUSTERING

## CLUSTERING

- Gli psicologi hanno identificato un serie di **fattori** che consentono al sistema visivo umano di **raggruppare** un **insieme** di **elementi**
- **Luminosità e colore**

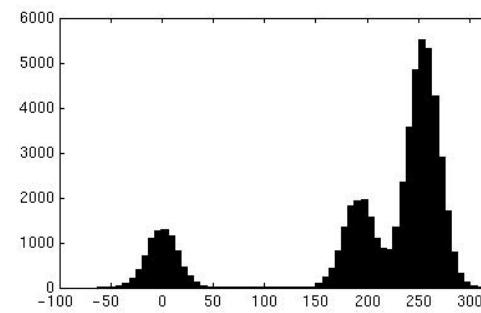
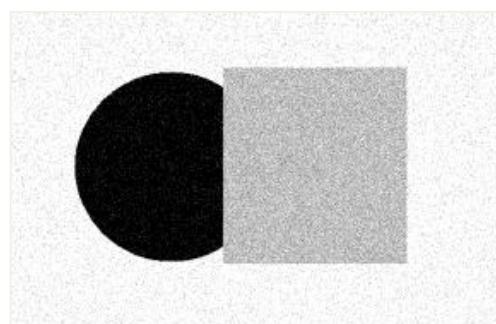
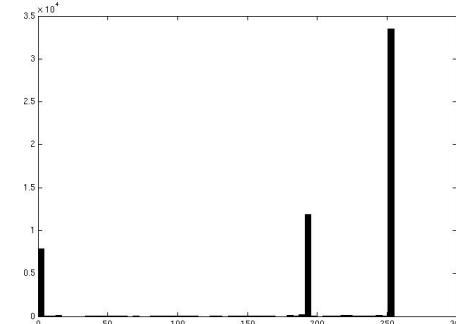
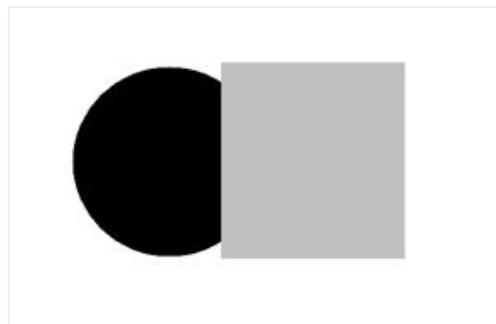
*“I stand at the window and see a house, trees, sky.  
Theoretically I might say there were 327 brightnesses  
and nuances of colour. Do I have "327"? No. I have sky, house,  
and trees.”*

**Max Wertheimer**  
**(1880-1943)**

Raggruppare pixel simili tra loro

## SOGLIATURA

- Per **segmentare** un'**immagine** in base al valore di **intensità** è possibile calcolare l'**istogramma** e trovare uno o più valori di **soglia** che separano le **mode**



Elim Parte II – Prof. A. Ferone

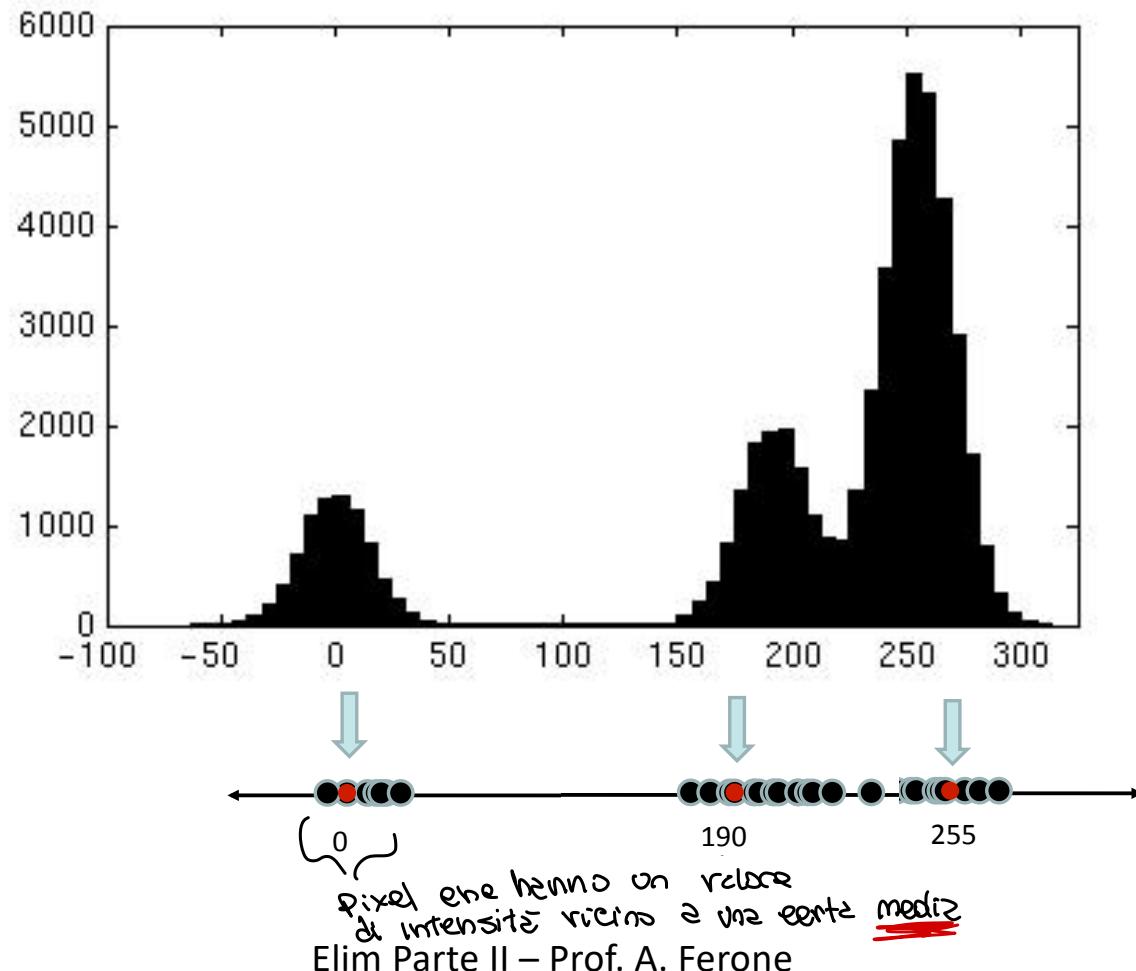
il Kmeans si propone di trovare un  
valore di intensità che si avvicina al valore  
di intensità dei pixel dell'immagine

↓  
cioè voglio trovare un certo colore  
di intensità media che attre dei  
pixel con un colore di intensità vicino  
a quello trovato

## CLUSTERING: K-MEANS

- Il **clustering** è una tecnica che consiste nel **raggruppare i pixel** in base alle caratteristiche di **intensità, colore e posizione**
- Nell'algoritmo **k-means**, si creano **k gruppi** (cluster) **disgiunti** che siano rappresentati da **k centri** (o **medie** dei valori di intensità dei pixel)
- Ogni **pixel** viene **assegnato** al **gruppo** rappresentato dalla **media** ad esso **più vicina**  
 $\downarrow_{\text{mean}}$
- I **pixel** di ogni gruppo sono **etichettati** con il **valore** del **centro** di quel **cluster** (ovvero la media)
- La scelta migliore per i **centri** è quella che minimizza la **Sum of Squared Distances** (SSD) tra tutti i **punti** ed il **centro** più vicino

## CLUSTERING: K-MEANS



## CLUSTERING: K-MEANS

- L'obiettivo è **minimizzare** la Sum of Squared Distances (**SSD**) tra tutti i punti ed il centro (media) più vicino, ovvero minimizzare la varianza in ogni cluster

↓ x ottenere gruppi di pixel + compatti

$$c^*, \delta^* = \operatorname{argmin}_{c, \delta} \left( \frac{1}{N} \sum_j \sum_i \delta_{ij} (c_i - x_j)^2 \right)$$

Where  $x_j$  is assigned to  $c_i$

$\delta_{ij}$  (red circle) mi dice l'oggetto / pixel  $j$  è quale cluster è assegnato

voglio trovare  $c$  e  $\delta$  in modo che riduci il più possibile la funzione

↓  
c e δ devono varcare in modo che la funzione ritorni risultati sempre + piccoli

NPixel      K cluster  
 normalization  
 Cluster center  
 Data, voci di intensità  
 centri, lemedie dei cluster (possono cambiare)

cluster

$\delta_{ij}$       1      2  
 1      1      0  
 2      0      1  
 3      0      1  
 4      1      0

pixel

$N$

$\delta_{ij}$       1      2  
 1      1      0  
 2      0      1  
 3      0      1  
 4      1      0

$K$

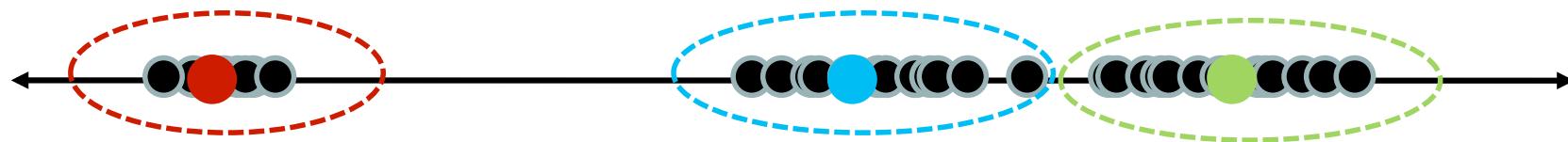
$0 = \text{False}$   
 $1 = \text{True}$

→ il pixel 2 non è assegnato al cluster 1 (0) ma al cluster 2 (1)

Elim Parte II – Prof. A. Ferone

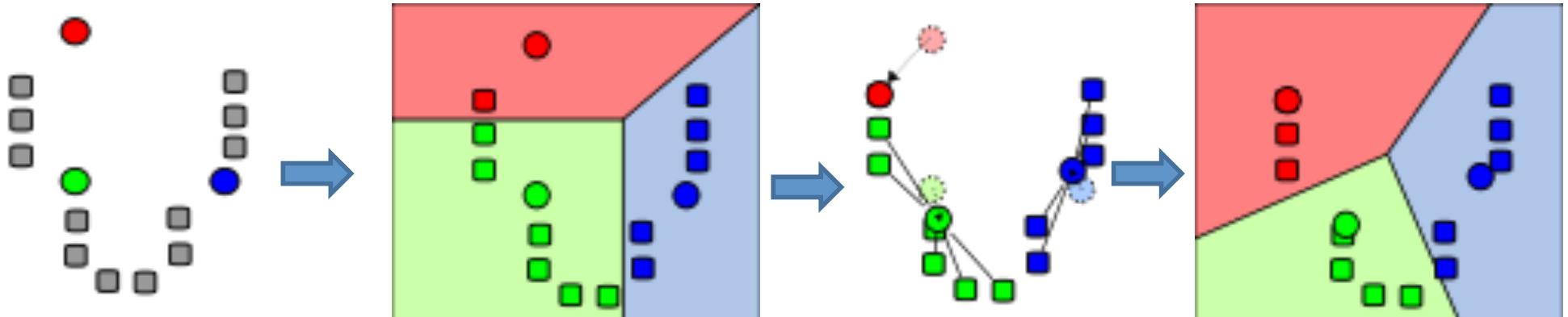
## CLUSTERING: K-MEANS

- Con questa **funzione obiettivo** possiamo risolvere il **problema**, ma quale?
- Se si **conoscono** i **centri** dei cluster, si **possono assegnare** i **pixel** al cluster con il centro più vicino
- Se si **conosce** a quale **gruppo** appartiene ogni pixel è possibile **calcolare** i **centri** di ogni gruppo



Elim Parte II – Prof. A. Ferone

# CLUSTERING: K-MEANS



0-Inizializzazione dei centri  
In maniera random  
↓ prendo 3 relativi e diventano centri iniziali

1-Assegnazione dei pixel ai cluster ( $\delta$ )  
↓ il pixel  $x_i$  lo assegno al cluster  $\Omega$  cui appartiene  
 $\bar{x}_i + r\text{etorno}$

2-Ricalcolo delle media dei relativi assegnati a ogni cluster  
↓ devo ricalcolare perché all'inizio l'assegnazione era casuale  
↓ quindi le nuove medie saranno il nuovo centro

Ripetere 1 e 2  
↓ finché le medie non sono le stesse  
↓ significa che i pixel non si spostano  
↓ Ho calcolato un minimo locale  $m_2$  non è un minimo globale

Elim Parte II – Prof. A. Ferone

## CLUSTERING: K-MEANS

- 1. Inizializzare** i centri di ogni cluster
- 2. Assegnare** ogni pixel al cluster con il centro più vicino

- 3. Aggiornare** i centri

$$\delta^t = \operatorname{argmin}_{\delta} \frac{1}{N} \sum_j^K \sum_i^K \delta_{ij} (\mathbf{c}_i^{t-1} - \mathbf{x}_j)^2$$

uno dipende  
dell'altro

$$\mathbf{c}^t = \operatorname{argmin}_{\mathbf{c}} \frac{1}{N} \sum_j^K \sum_i^K \delta_{ij}^t (\mathbf{c}_i - \mathbf{x}_j)^2$$

- 4. Ripetere** i punti 2 e 3 finché il centro (media) di ogni cluster non viene più modificato (ovvero i gruppi non vengono modificati)

## CLUSTERING: K-MEANS

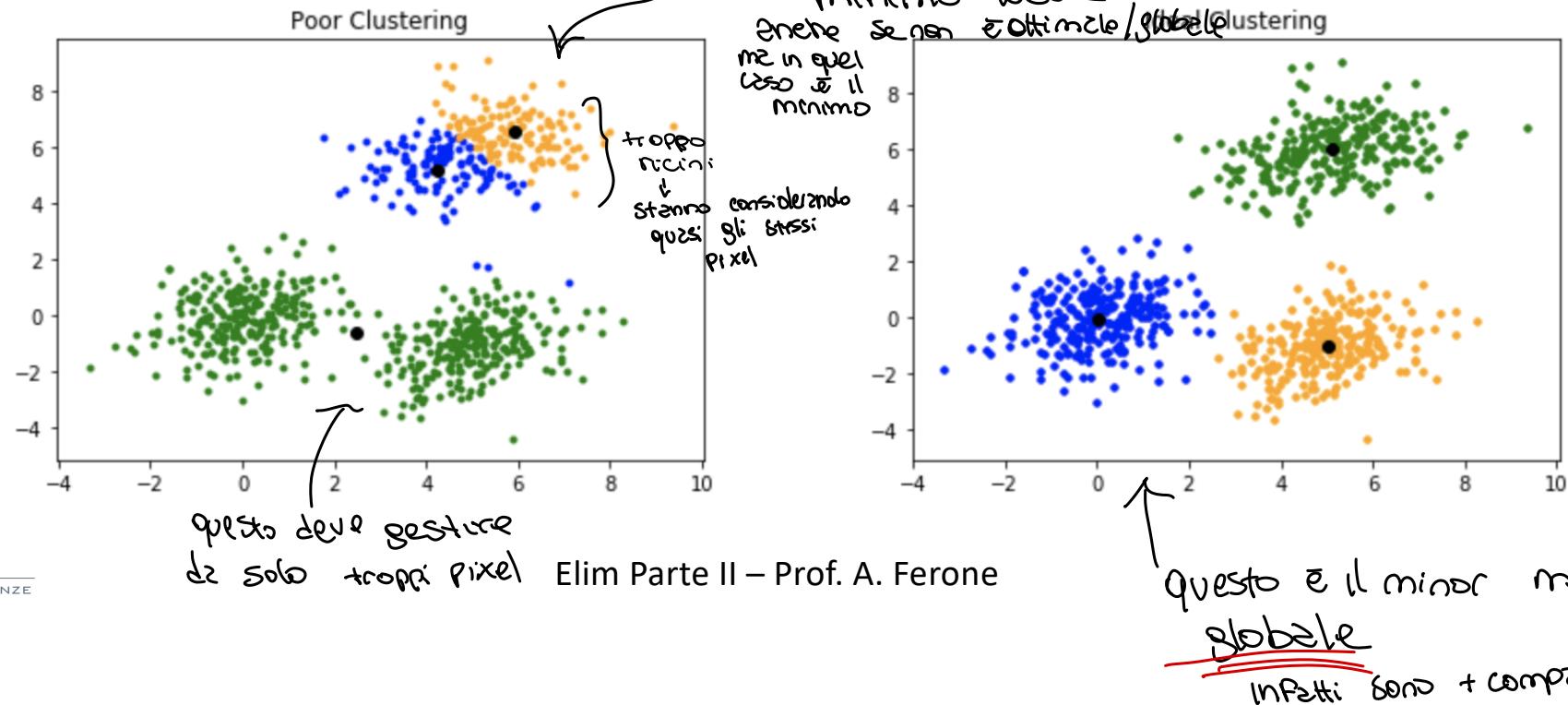
1. **Inizializzare** i centri di ogni cluster (*selego k : numero di centri che esegno random*)
2. **Assegnare** ogni pixel al cluster con il centro più vicino
  - Per ogni pixel  $p_j$  calcolare la **distanza** dai **k centri  $c_i$**  ed **assegnare  $p_j$**  al **cluster** con il centro  $c_i$  **più vicino**
3. **Aggiornare** i centri
  - **Calcolare** la **media** dei pixel di ogni cluster
4. Ripetere i punti 2 e 3 finchè il centro (media) di ogni cluster non viene più modificato (ovvero i gruppi non vengono modificati)

## CLUSTERING: K-MEANS

- Dettagli implementativi
- I **k centri iniziali** possono essere **scelti casualmente** tra i pixel dell'immagine o in maniera più oculata
- Come distanza solitamente si utilizzata la **distanza euclidea** (RGB)
- L'algoritmo potrebbe terminare in un **minimo locale** (i gruppi non vengono più modificati in corrispondenza di una soluzione non ottimale) per cui si può **rieseguire** più volte l'**algoritmo**

## CLUSTERING: K-MEANS++

- Se i **k centri iniziali** sono **tropo vicini** il risultato finale potrebbe non essere ottimale
- Per ovviare a questo problema i **centri iniziali** possono essere scelti in modo che siano **distanti** tra loro



## CLUSTERING: K-MEANS++

- L'algoritmo di **inizializzazione** prevede i seguenti passi:
- **Scegliere** il primo centro  $c_1$  in maniera **random**
- Calcolare la **distanza** di ogni **punto** da  $c_1$
- **Scegliere** tra i restanti **pixel** il prossimo centro  $c_2$  con **probabilità** direttamente **proporzionale** alla sua **distanza** da  $c_1$  (ovvero il **pixel** con valore più **lontano**)
- **Ripetere** finchè non sono stati scelti i **k centri iniziali**

→ tra di loro

Quel è il costo ?

Elim Parte II – Prof. A. Ferone

↓  
è come una matrice diagonale

## CLUSTERING: K-MEANS

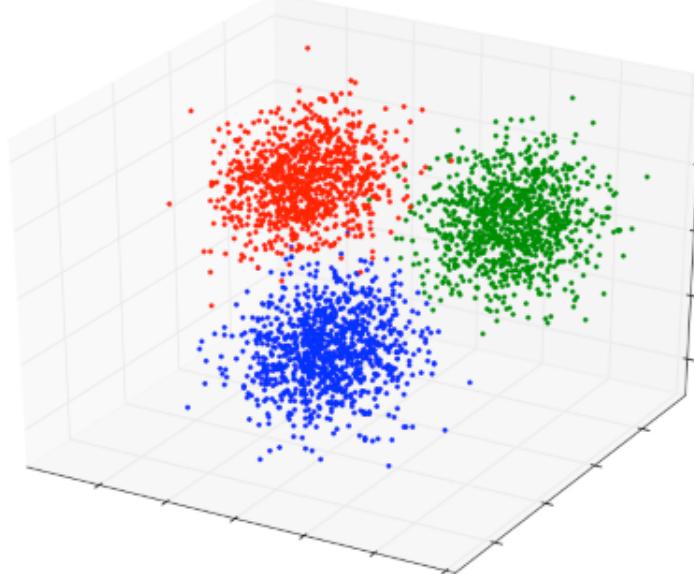


È come se stessi facendo una quantizzazione  
dei livelli di grigio

Elim Parte II – Prof. A. Ferone

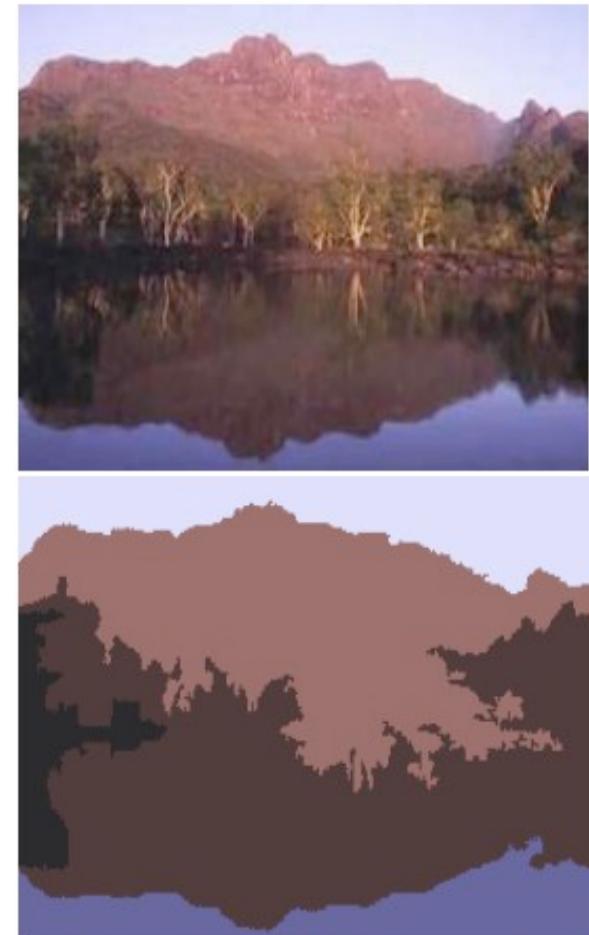
## CLUSTERING: K-MEANS

- Lo **stesso procedimento** si può utilizzare se si considerano **immagini a colori**
- In questo caso ogni **pixel** sarà un **vettore** di **tre componenti**
- I **centri** di ogni cluster saranno dei **vettori** di **tre componenti**



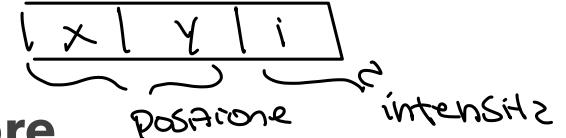
Elim Parte II – Prof. A. Ferone

## CLUSTERING: K-MEANS



Elim Parte II – Prof. A. Ferone

## CLUSTERING: K-MEANS

- Pixel con **valori simili** si possono trovare anche molto **distanti** all'interno dell'immagine  $\Rightarrow$  viola il principio di ~~coerenza~~ che devono essere **componenti connesse**
- Quando si **segmenta** un'immagine cerchiamo **gruppi** di **pixel "compatti"**
- È possibile utilizzare l'**informazione** relativa alla **posizione** dei **pixel** nell'immagine per ottenere **regioni** più **compatte**
- Ogni **pixel** sarà rappresentato da un **vettore** le cui **componenti** saranno
  - Le **coordinate spaziali** ed il livello di **intensità**  $\rightarrow$  
  - Le **coordinate spaziali** e le tre **componenti del colore** 

↓ Il fatto che ho componenti intensità e posizione insieme non mi garantisce componenti connesse

Elim Parte II – Prof. A. Ferone

## CLUSTERING: K-MEANS



## CLUSTERING: K-MEANS

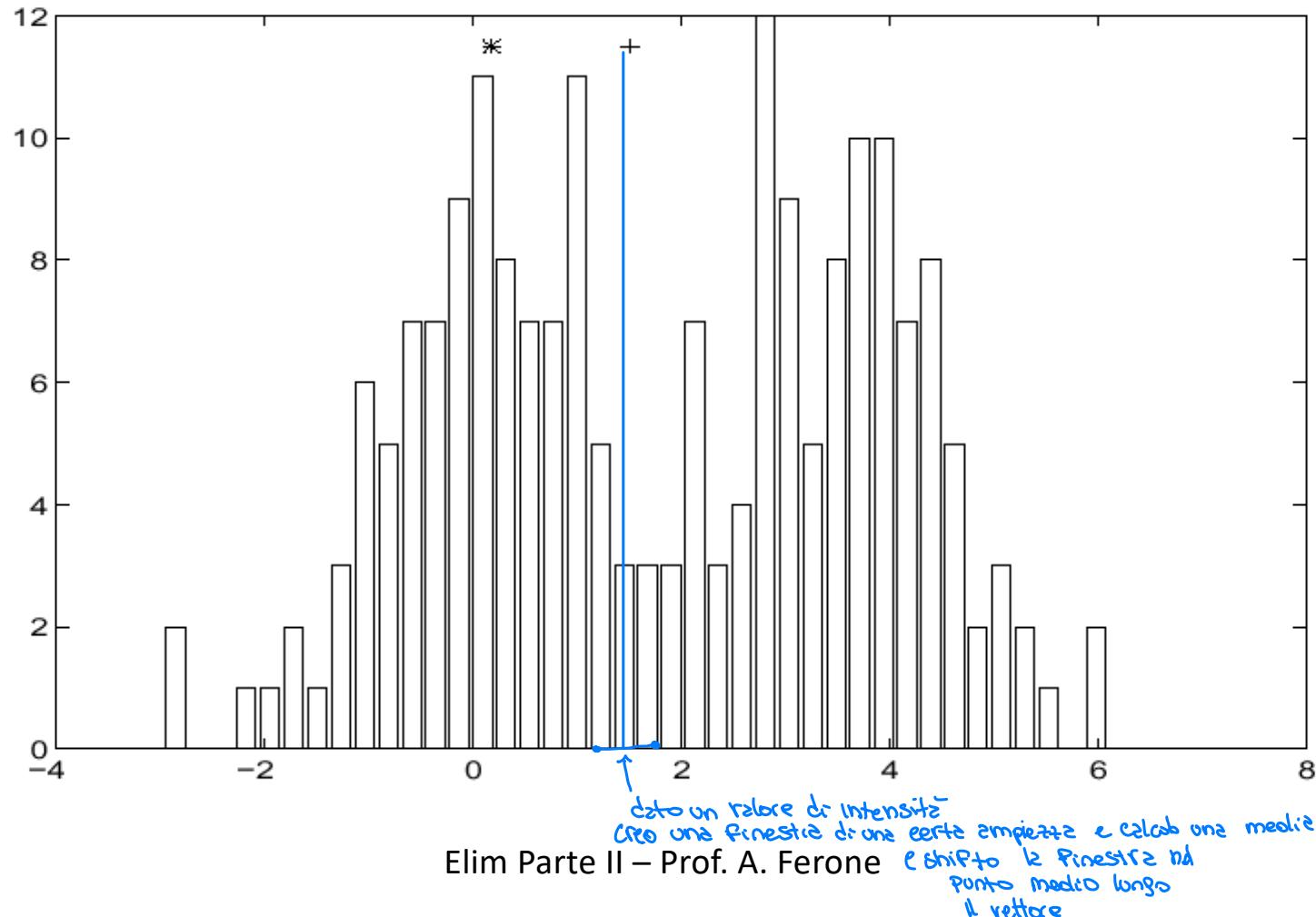
- Alcune considerazioni
  - 1. Come **scegliere** il valore **k**?
  - 2. Come valutare la **qualità** dei **cluster** al variare di k?
  - 3. Come sono le **prestazioni** con **immagini** di **grandi** dimensioni? (la complessità è **O(kNd)** dove k è il numero di cluster e d il numero di feature)

Nel caso di intensità  $d=1$  ( $\downarrow$ )  
di colore e posizione  $d=5$  ( $x, Y, R, G, B$ )  
di intensità + posizione  $d=3$  ( $x, Y, i$ )  
di colore  $d=3$  (RGB)

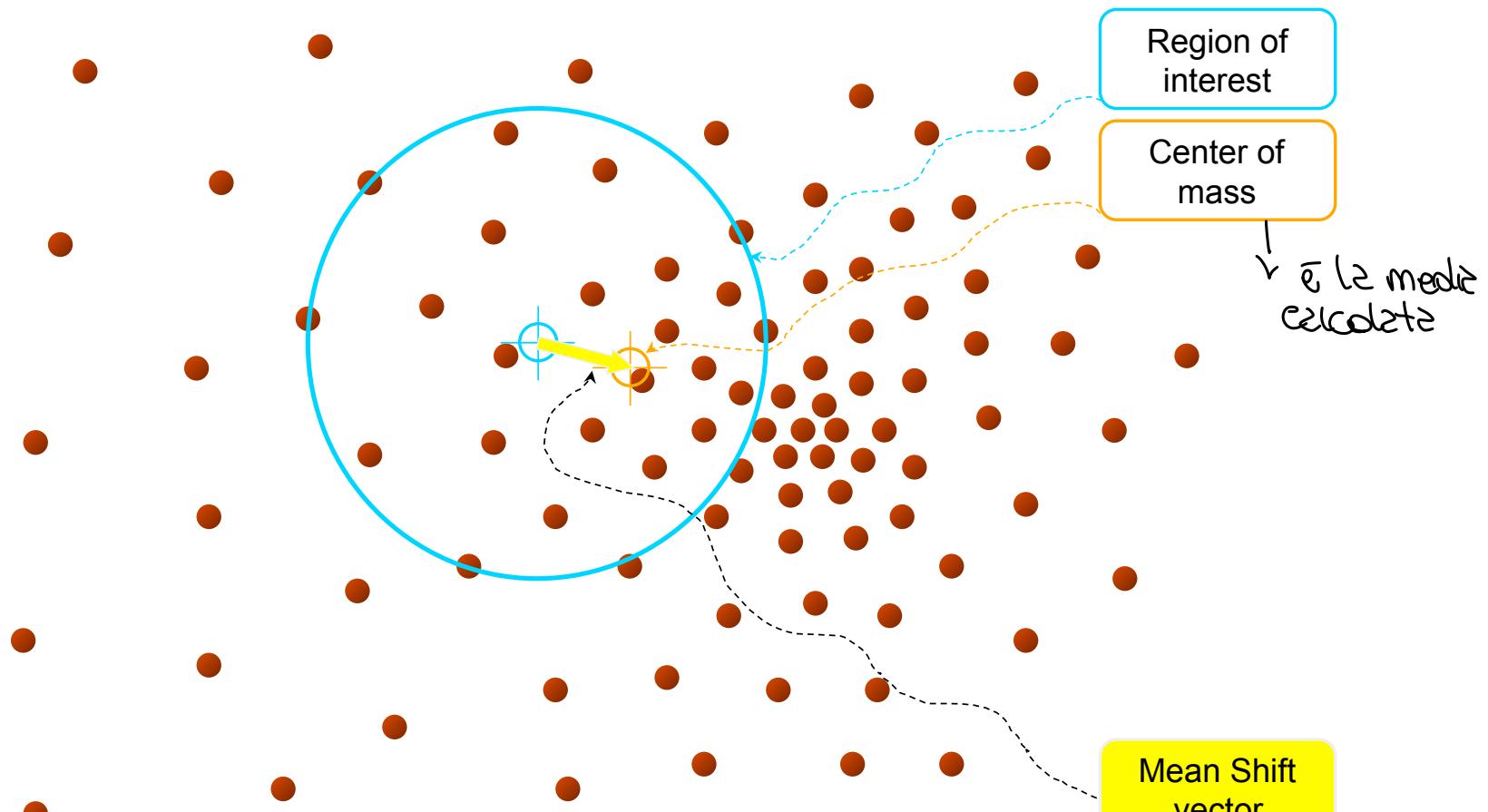
## CLUSTERING: MEAN-SHIFT

- **Mean-shift** è un **algoritmo** più **avanzato** per la segmentazione
- Consiste nel **cercare**, per ogni valore, la **moda** più **vicina** nello spazio delle feature (intensità) *(cerca i picchi senza sapere quali cluster sono)* *(i picchi)*
- Per **ogni valore** di intensità si considera una **finestra** di **ampiezza W**
- Si **calcola** la **media** (mean) nella finestra **W**
- Si **sposta** (shift) la **finestra** sulla **media calcolata** e si **ripete** il procedimento finchè la media non cambia

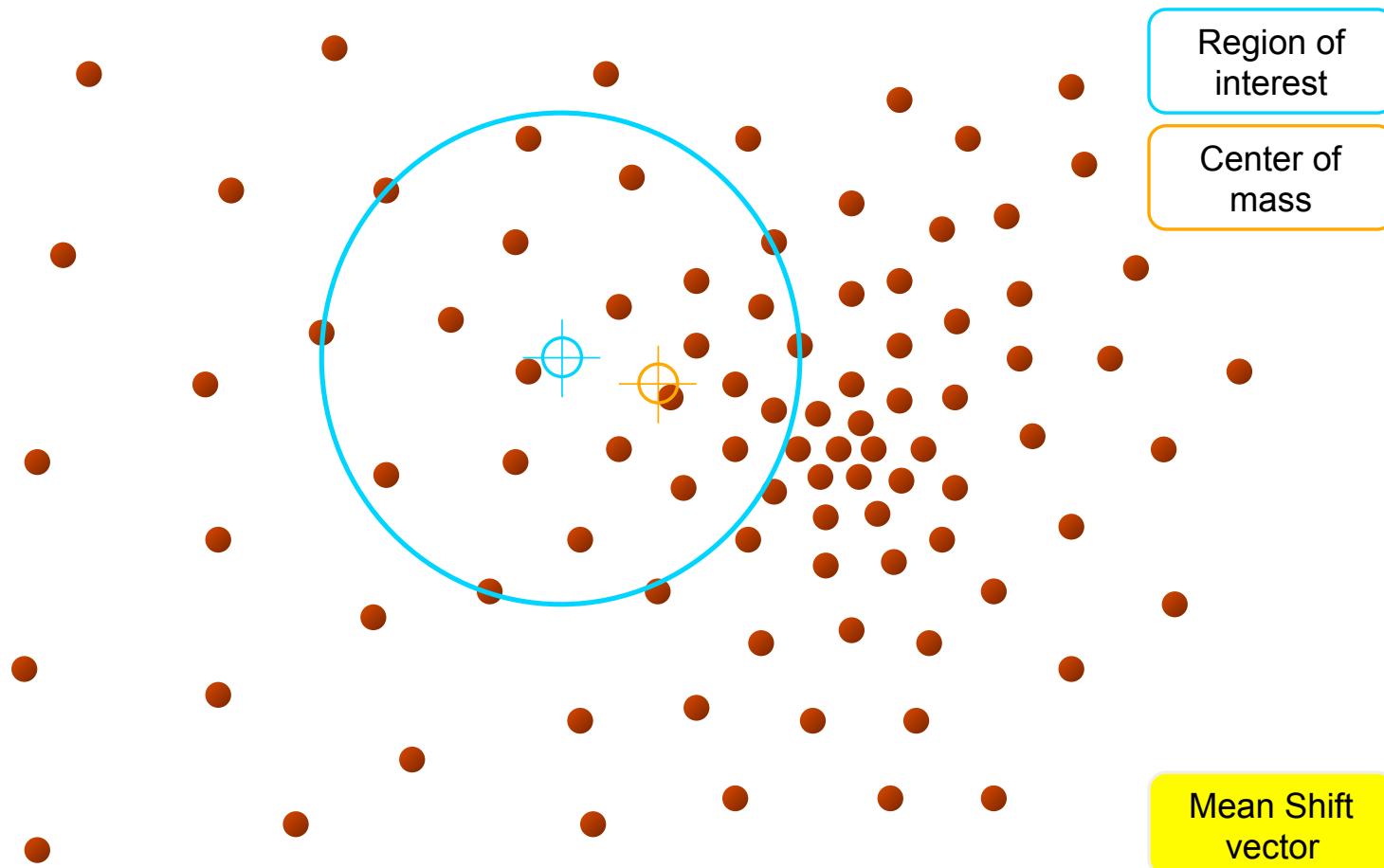
## CLUSTERING: MEAN-SHIFT



## CLUSTERING: MEAN-SHIFT

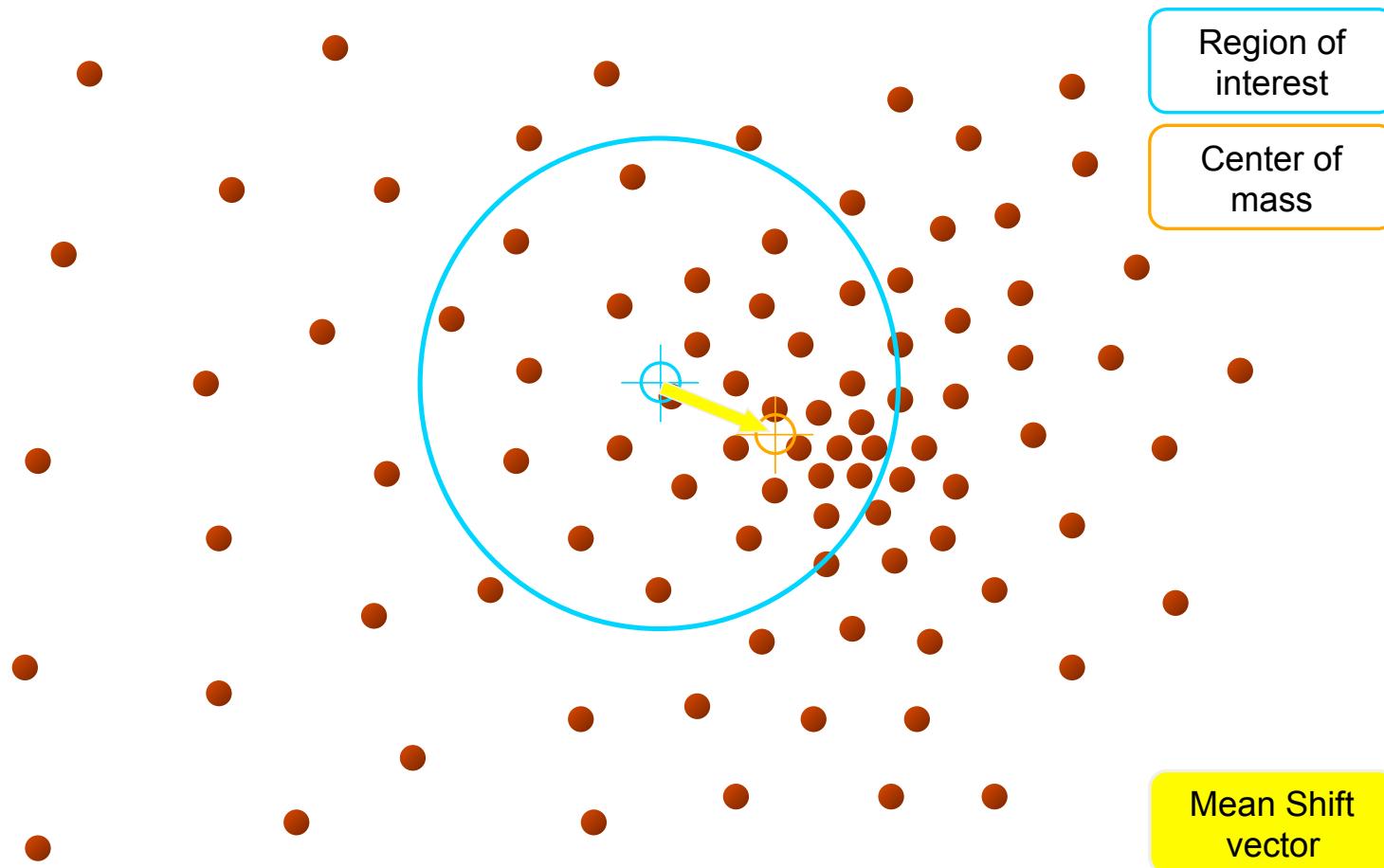


## CLUSTERING: MEAN-SHIFT

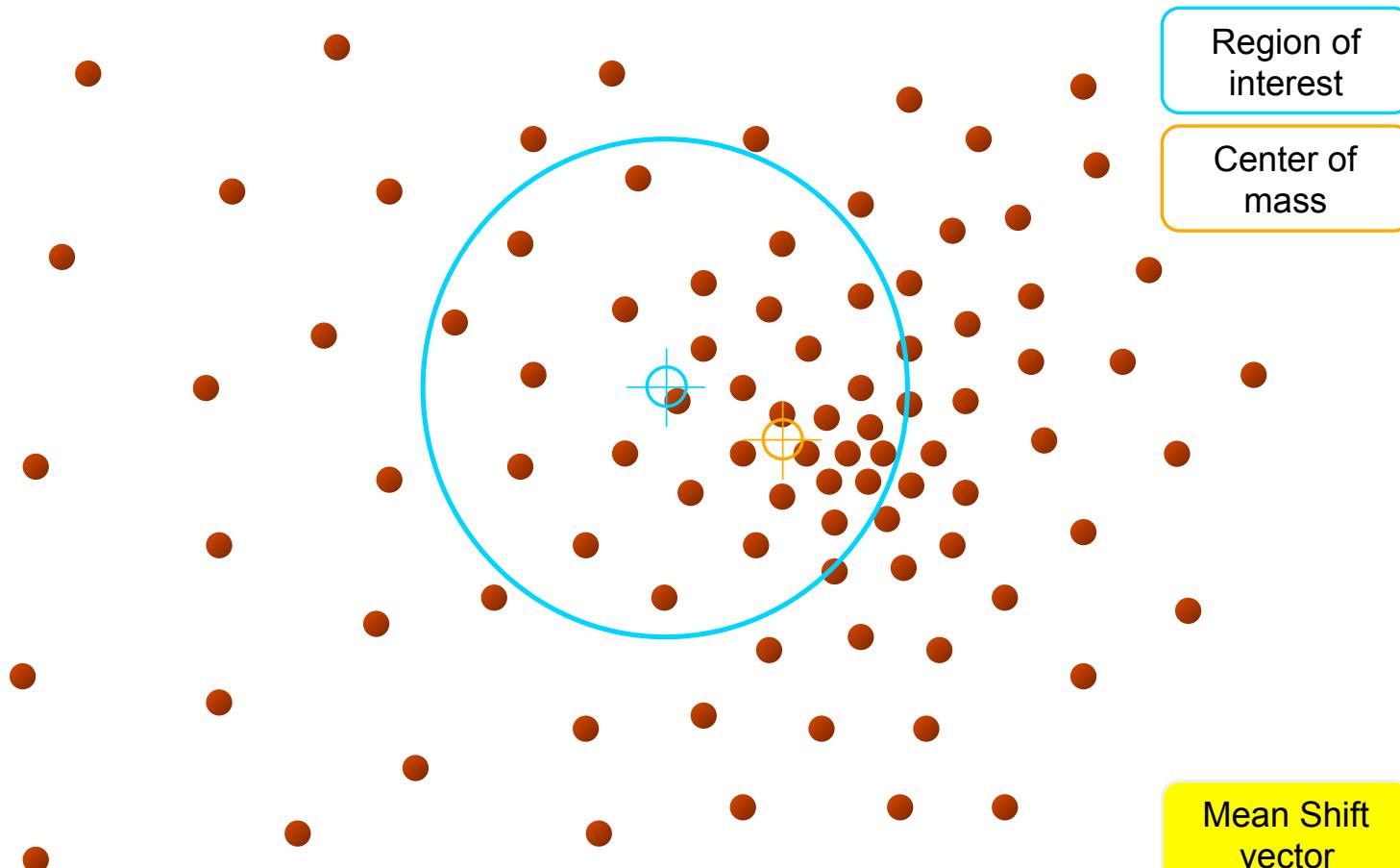


Elim Parte II – Prof. A. Ferone

## CLUSTERING: MEAN-SHIFT

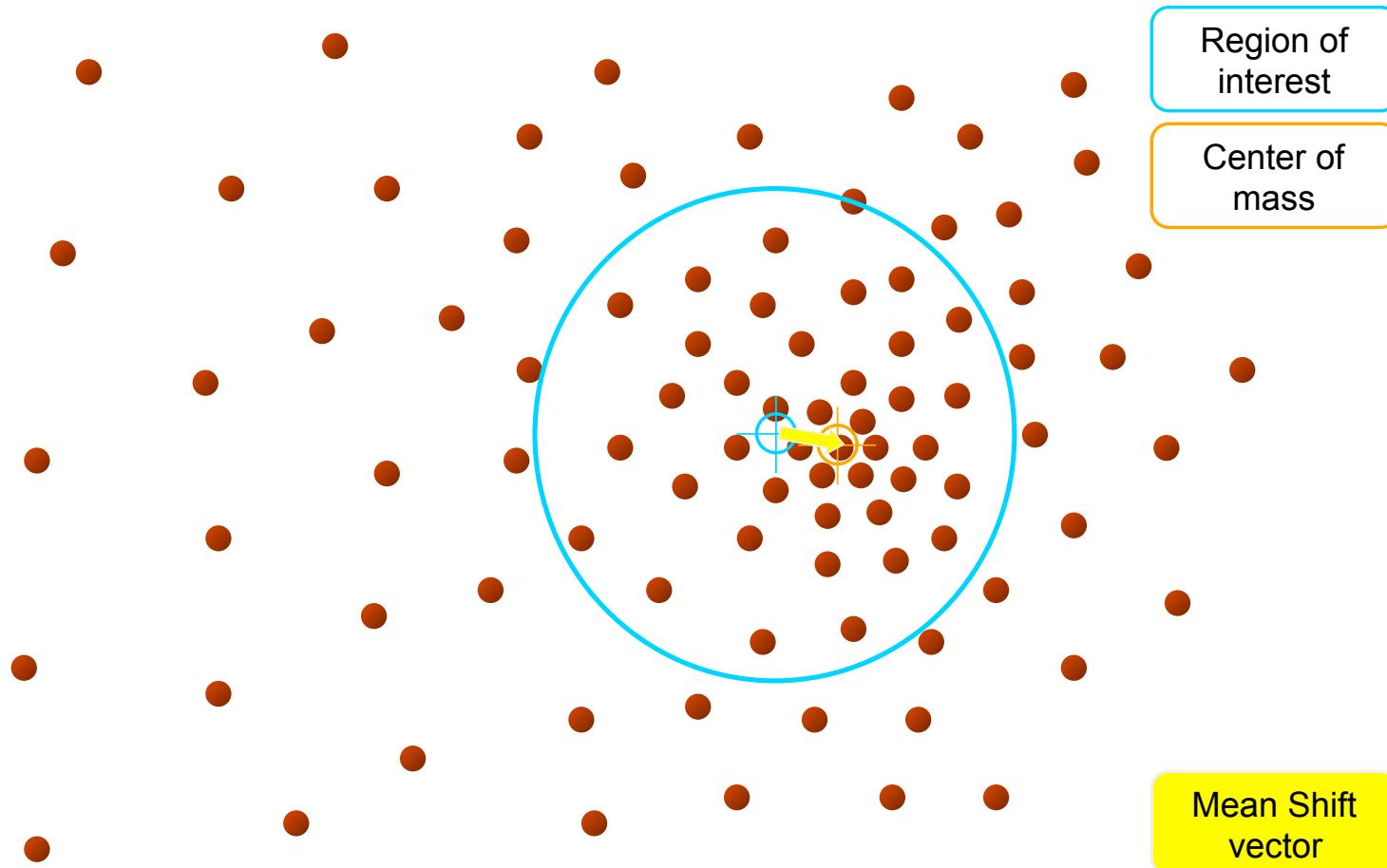


## CLUSTERING: MEAN-SHIFT



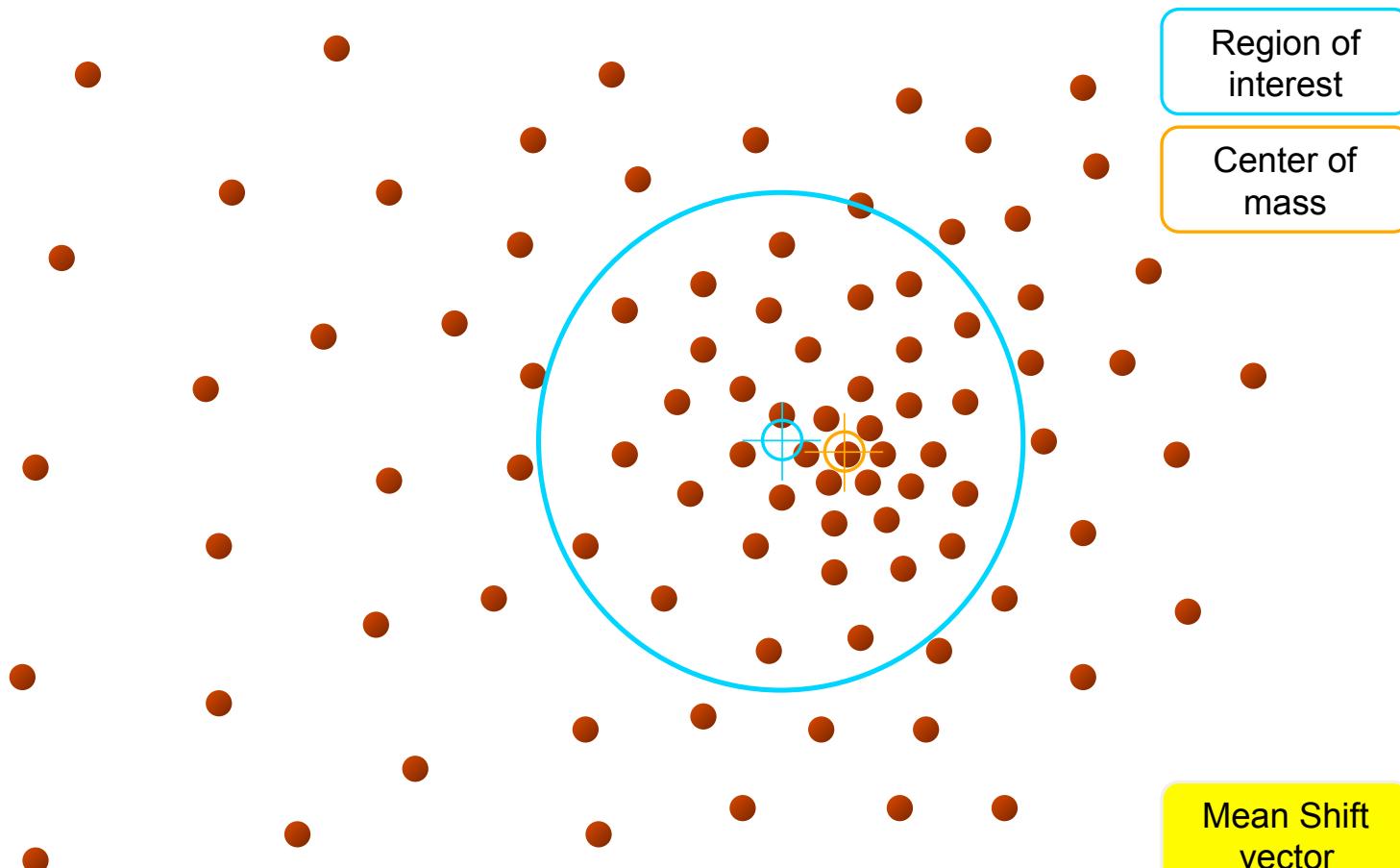
Elim Parte II – Prof. A. Ferone

## CLUSTERING: MEAN-SHIFT



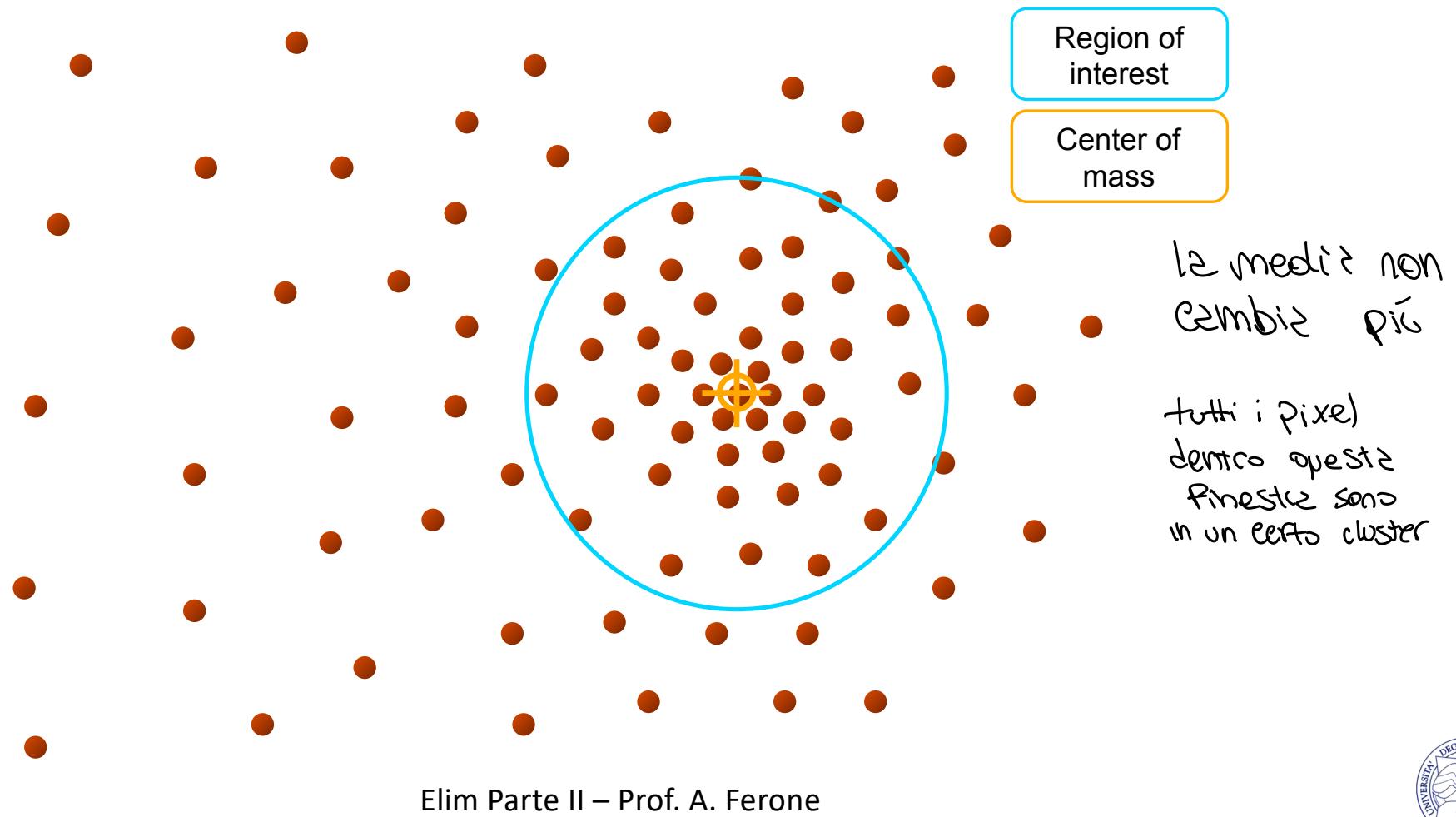
Elim Parte II – Prof. A. Ferone

## CLUSTERING: MEAN-SHIFT



Elim Parte II – Prof. A. Ferone

## CLUSTERING: MEAN-SHIFT

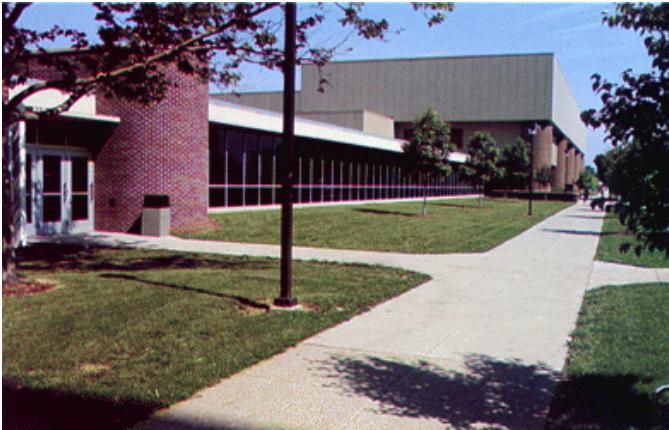


## CLUSTERING: MEAN-SHIFT

```
for p in f(x,y)
    while |p_outi-p_outi-1|>th
        shift=n=0
        for pt in f(x,y)
            if pt ∈ W      se il valore appartiene alla finestra
                shift+=pt  incrementa il valore
                n++         numero di pixel
        p_out=shift/n  media dell'iterazione
    ↓
    è il valore del cluster
    a cui pt è stato
    assegnato
```

```
for p in f(x,y)
    while |p_outi-p_outi-1|>th
        shift=n=0
        for pt in f(x,y)
            d=dist(p,pt)
            w=kernel(d,W)  la finestra è un kernel,
            shift+=pt * w  una funzione gaussiana
            n+=w
            p_out=shift/n
    ↑
    + preciso ma
    cost di più
```

## CLUSTERING: MEAN-SHIFT



Elim Parte II – Prof. A. Ferone

## CLUSTERING: MEAN-SHIFT

- Alcune considerazioni
  - 1. Non è necessario specificare il **numero** di **cluster**
  - 2. L'**ampiezza** della **finestra** determina il risultato finale + è grande - dettagliata sull'immagine
  - 3. Come valutare la **qualità** dei **cluster** al variare della finestra?
  - 4. Come sono le **prestazioni** con immagini di grandi dimensioni? (la complessità è **O(kN<sup>2</sup>d)** dove k è il numero di iterazioni e d il numero di feature)



# CLUSTERING: KMEANS

```
double cv::kmeans(  
    cv::InputArray      data,  
    int                 K,  
    cv::InputOutputArray bestLabels,  
    cv::TermCriteria    criteria,  
    int                 attempts,  
    int                 flags,  
    cv::OutputArray     centers = cv::noArray() // (optional) found centers  
);
```

# CLUSTERING: MEAN-SHIFT OPENCV

```
void cv::pyrMeanShiftFiltering(  
    cv::InputArray    src,                      // 8-bit, Nc=3 image  
    cv::OutputArray   dst,                     // 8-bit, Nc=3, same size as src  
    cv::double        sp,                      // Spatial window radius  
    cv::double        sr,                      // Color window radius  
    int              maxLevel = 1,            // Max pyramid level  
    cv::TermCriteria termcrit = TermCriteria(  
        cv::TermCriteria::MAX_ITER | cv::TermCriteria::EPS,  
        5,  
        1  
    )  
);
```

# ESERCIZI

- Provare il codice k-means e mean shift
- Implementare l'algoritmo k-means

# ESERCIZI

- Provare il codice k-means e mean shift
- Implementare l'algoritmo k-means