

Képfeldolgozás a gyakorlatban

klaszterezés, alakleírók

Klaszterezés

- ▶ **Felügyelet nélküli** gépi tanuló eljárás
- ▶ Input adathalmaz: jelöltetlen (osztálycímke nélküli minták)
 $X = \{x_1, x_2, \dots, x_n\}$ mintahalmaz, ahol
 $x_i = [v_{i1}, v_{i2}, \dots, v_{ik}]$ a minták *jellemzővektora*
 $i=(1, 2, \dots, n)$
- ▶ Cél: rejtett struktúra feltárása
- ▶ Csoportok (klaszterek) automatikus létrehozása úgy, hogy az egy csoportba tartozó minták (valamilyen szempontból) **hasonlóbbak** legyenek, mint a különböző csoportba tartozó minták.
- ▶ Gyakorlati haszon:
 - a csoportok feltárása
 - esetleges kiugró minták megtalálása
 - hiányzó adatok pótlása (amennyiben az adott mintát lehet klaszterezni a meglévő értékek alapján, a csoport jellemzők alapján pótolhatók az értékek)

Klaszterezés: Példa

- ▶ Input adathalmaz:
minta: 30 személy
jellemzők:
 - magasság
 - hajhossz
- ▶ Elkülöníthető-e a nők és a férfiak csoportja?

- ▶ Az ábrán látott jelölések:
alakzatok – nem:

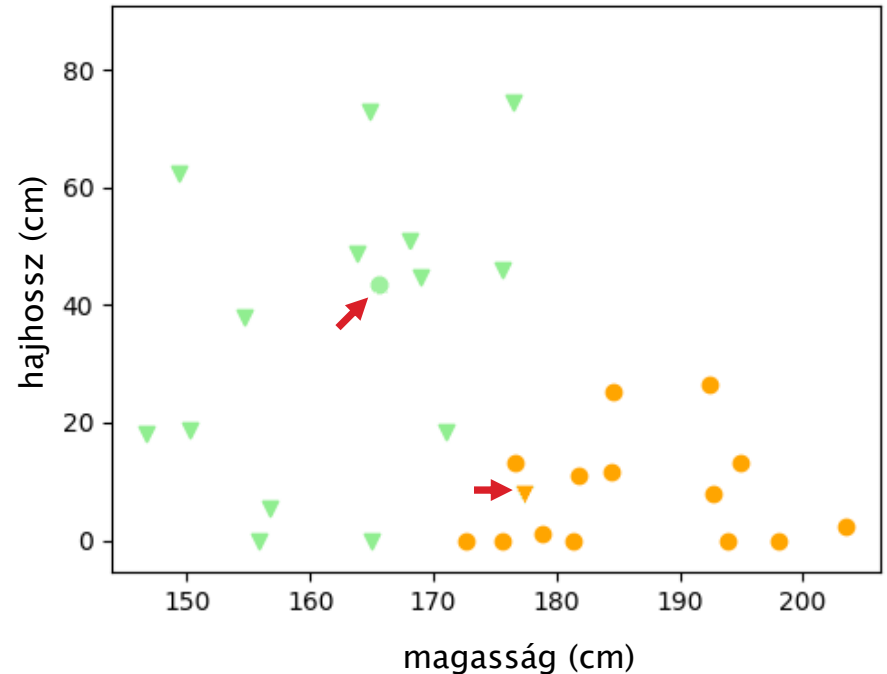
háromszög: nők

kör: férfiak

Megj: A klaszterező algoritmus számára ez ismeretlen

színek – automatikus klaszterek

piros nyíl – hibás klaszterezés (a jellemzők nem alkalmasak a tökéletes bontásra)



Lépések:

- ▶ Minden mintának meghatározzuk a jellemzővektorát
- ▶ Előfeldolgozás: (opcionális)
 - normalizálás
 - dimenzió redukció
- ▶ Klaszterezés
- ▶ Vizualizáció a létrejött klaszterek ellenőrzésére

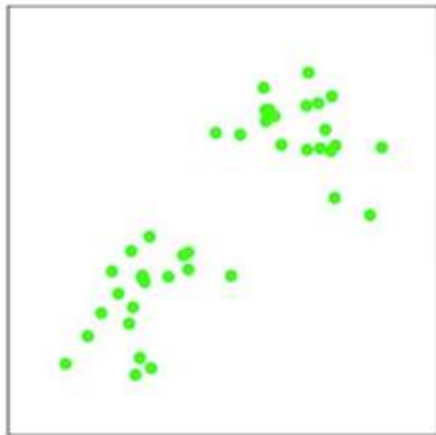
K-means (K-közép) klaszterező

▶ Algoritmus

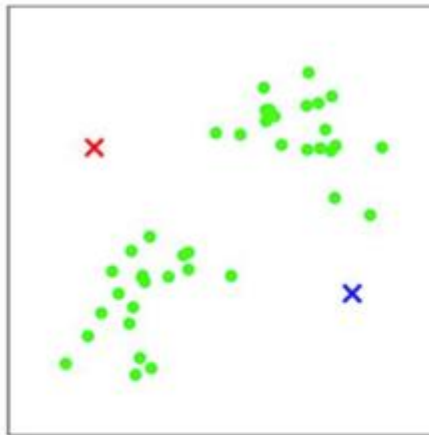
- K darab középpontok választása
(pl. véletlenszerűen, vagy *a priori* ismeret alapján)
- Ciklus, amíg van (érdemi) változás (és az iterációk száma egy határ alatt van):
 - minták besorolása a legközelebbi klaszterbe:
a klaszterközéppontot leíró vektor és a mintát leíró vektor távolsága alapján
 - klaszterközéppontok újraszámítása

K-means (K-közép) klaszterező

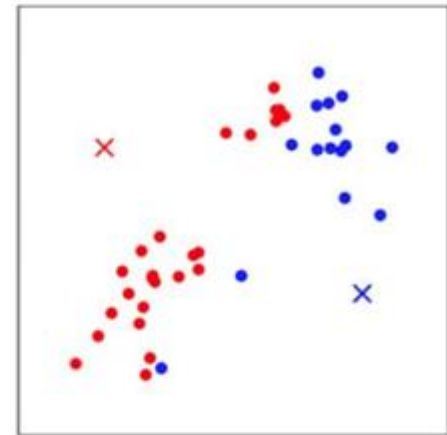
Példa $k = 2$ esetre



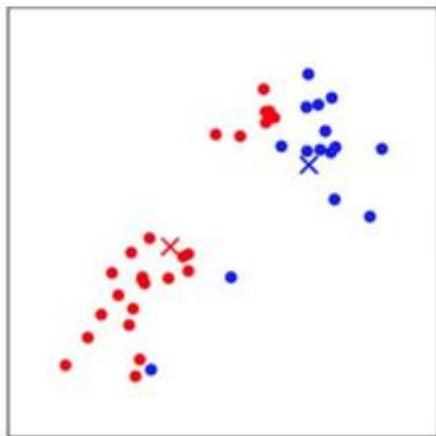
input halmaz



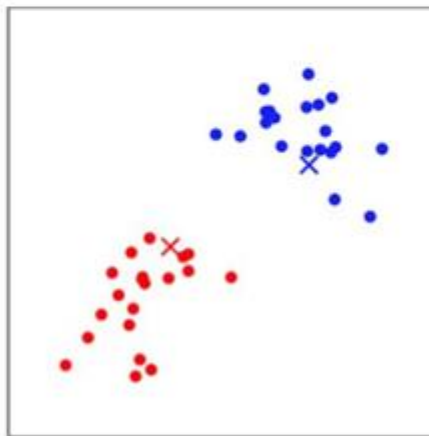
input halmaz + k véletlen
középpont



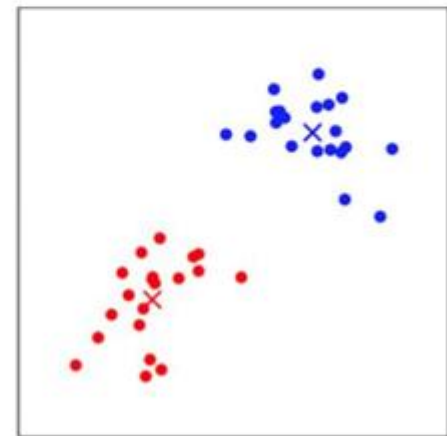
legközelebbi középponthoz való
besorolás alapján képzett klaszterek



Középpontok újraszámítása



legközelebbi középponthoz való
besorolás alapján képzett klaszterek



Középpontok újraszámítása

Feladat: K-means

- ▶ Generál adathalmazt a magasság, hajhossz példának megfelelően
- ▶ Az adat mátrixot 15 férfi és 15 női jellemzőknek megfelelő adattal töltsd fel:
 - a férfiak magasságánál legyen 186 a nőknél legyen 160 a középérték.
 - a férfiak hajhosszánál legyen 10 a nőknél legyen 45 a középérték.
 - a szórásokat állítsd tetszés szerint.

```
...
#include <random>
...
int main(){

    std::default_random_engine generator;
    std::normal_distribution<float> male_height(186.0f, 30.0f);
    ...
    Mat data(30, 2, CV_32F);
    // férfiak adatai
    for(int i = 0; i<15; ++i){
        data.at<float>(i, 0) = male_height(generator);
        data.at<float>(i, 1) = ...
    }
    // nők adatai
    for(int i = 15; i<30; ++i){
        ...
    }
}
```

Feladat: K-means

Hozz létre egy kritériumot, ami megadja, hogy hány iteráció után, milyen változás alatt álljon le a K-közép eljárás. Pl:

```
TermCriteria krit(  
    TermCriteria::Type::EPS|TermCriteria::Type::MAX_ITER,  
    100, 1.0);
```

Végezd el a klaszterezést. A k értéke legyen 2, mert két klasztert akarunk. A bestLabels üres. centers alapértelmezett marad.

```
kmeans( InputArray data, // a jellemzőmátrix  
    int k, // a klaszterek maximális száma  
    InputOutputArray bestLabels, // véletlen középpontnál üres Mat  
    TermCriteria krit, //a leállási feltétel  
    int attempt, //kísérletek száma  
    int flag, // inicializálási mód, pl. KMEANS_RANDOM_CENTERS,  
    OutputArray centers=noArray() //üres Mat, a klaszterközepponok  
);
```


Feladat: K-means

Hozz létre egy 300 x 300-as **fehér** képet és jelenítsd meg rajta pl.
`drawMarkers(InputOutputArray img, Point pt, Scalar color,
MARKER_FLAG shape, ..)` függvénnnyel a pontokat.

A `bestLabels.at<int>(i)` adja meg, hogy az *i*. adatot melyik klaszterbe tette a gép, ezért a rajzoláshoz használhatod:

```
int shape = cluster_labels.at<int>(i) == 0 ?  
                MARKER_CROSS : MARKER_DIAMOND;
```

A színt változtathatod annak megfelelően, hogy eredetileg női vagy férfi jellemzőhöz tartozott-e az *i*. adat.

Jellemzők (képfeldolgozás)

- ▶ Ideális esetben a minták olyan leírói, amelyek értékei az azonos osztályba tartozó mintákra nézve hasonlóak, a különböző osztályba tartozó mintáknál pedig nem.
- ▶ Alapvető kategóriák:
 - Intenzitásérték, színezet
 - Textúraleírók
 - Alakleírók
- Előbbiek az objektumok szegmentálásra is alkalmasak lehetnek
- Utóbbi esetben abból indulunk ki, hogy az objektum már megvan

Milyen jellemzőt válasszak?

▶ Problémafüggő

▶ Szeder vs. málna:

- Alakleíró? nem
- Textúraleíró? nem
- Szín? igen



▶ Szeder vs. áfonya

- Alakleíró? pl: körszerűség
- Textúraleíró? igen
- Szín? nem



Feladat: Képpontok klaszterezése szín alapján /kvantálás/

▶ Input:

- egy színes kép (img)

▶ Minták:

- maguk a képpontok
- minták száma: $\text{img.rows} * \text{img.cols}$

▶ Jellemzők:

- a képpontok L, a, b értékei (CIE Lab színteret fogunk használni)
- a jellemzők száma: 3

Lépések: az adathalmaz előállítása

- ▶ Olvass be egy képet színesbe
 - ▶ Konvertáld át CV_32F-be (`convertTo`)
 - mert a Kmeans CV_32F-et vár
 - ▶ Konvertáld át CIE Lab színtérbe (`cvtColor`)
 - mert tudjuk, hogy ebben a színtérben jobban tudjuk mérni a színek közti eltérést, mint RGB-be
 - ▶ Állítsd elő a minták (pixelelek) jellemzővektorát:
 - 1 csatornás, $kep.rows * kep.cols$ soros, 3 oszlopos mátrixszá formáljuk
- ```
Mat data = lab.reshape(1, img.rows*img.cols)
```

*A reshape függvénynek a csatornák (1) és a sorok számát ( $img.rows * img.cols$ ) adatuk meg, az oszlopot automatikusan számolja.*


*Ha megjeleníted a data-t, akkor egy csíkszerű képet kell látnod.*

# Klaszterezés megvalósítása

- ▶ Hozz létre egy kritériumot, ami megadja, hogy hány iteráció után, milyen változás alatt álljon le a K-közép eljárás. Pl:

```
TermCriteria krit(
 TermCriteria::Type::EPS | TermCriteria::Type::MAX_ITER,
 100, 0.001);
```

ha nem konvergál az eljárásod  
vedd magasabbra



- ▶ Végezd el a klaszterezést:

```
kmeans(InputArray data, // a jellemzőmátrix
 int k, // a klaszterek maximális száma
 InputOutputArray bestLabels, // véletlen középpontnál üres Mat
 TermCriteria krit, //a leállási feltétel
 int attempt, //kísérletek száma
 int flag, // inicializálási mód, pl. KMEANS_RANDOM_CENTERS,
 OutputArray centers=noArray() //üres Mat, a klaszterközepponok
);
```

# Feladat: a klaszterezett kép megjelenítése

2 sor, 3 oszlop  
Lab kép

|   |     |     |     |
|---|-----|-----|-----|
| L | 0.2 | 0.2 | 0.8 |
| a | 0.9 | 0.9 | 0.0 |
| b | 0.9 | 0.7 | 0.1 |
| L | 0.2 | 0.8 | 0.8 |
| a | 0.8 | 0.0 | 0.0 |
| b | 0.9 | 0.0 | 0.1 |

minden pixelt külön mintának tekintünk, 3 jellemzővel: L, a b  
ezért az data mátrix 6x3-as

*reshape*

|     |     |     |
|-----|-----|-----|
| 0.2 | 0.9 | 0.9 |
| 0.2 | 0.9 | 0.7 |
| 0.8 | 0.0 | 0.1 |
| 0.2 | 0.8 | 0.9 |
| 0.8 | 0.0 | 0.0 |
| 0.8 | 0.0 | 0.1 |

*kmeans (K=2-vel)*

a `bestLabels` megadja,  
hogy a data mátrix *i.* sora melyik  
klaszterbe került (típus: int)

|   |
|---|
| 0 |
| 0 |
| 1 |
| 0 |
| 1 |
| 1 |

a `centers` megadja, a kaszter középpontokhoz  
tartozó jellemzőket  
( `klaszterek_száma` x `jellmezők_száma` méretű,  
float típusú )

|     |      |      |
|-----|------|------|
| 0.2 | 0.87 | 0.82 |
| 0.8 | 0.0  | 0.07 |

# Megjelenítés

A centers mátrix a klaszterbe eső pixelek átlagos színértékeit tartalmazza. L, a, b értékek állnak benne, de külön-külön oszlopban (float, float, float)

- ▶ A három oszlopot vond össze, hogy 1 Vec3f-es oszlop (Lab) legyen:

```
Mat cluster_colors_lab = centers.reshape(3, centers.rows);
```

- ▶ Konvertáld vissza BGR színtérbe (cvtColor)
- ▶ Konvertáld vissza CV\_8UC3 típusú (convertTo), legyen *cluster\_colors* a mátrix neve

A bestLabels mátrix jelenleg egyetlen `img.rows*img.cols` sorból álló vektor. Formázzuk vissza képszerűvé, hogy az (i, j). képponthoz tartozó címke a mátrix (i, j). pontján legyen:

```
Mat labels = bestLabels.reshape(1, img.rows);
```

- ▶ Hozz létre egy képet az eredeti kép méretével és típusával és **színezd ki**:
  - Járd be: 

```
for (int i = ...,
 for(int j = ...
```
  - Tudd meg, hogy melyik klaszterhez tartozik a pixel: `int idx = labels.at<int>(i, j)`  
Ha az `idx` `[0, cluster_colors.rows-1]` tartományba esik:
  - Állítsd be a képpont színét: `dest.at<Vec3b>(i, j) = cluster_colors.at<Vec3b>(idx)`



# Alakleírók

Kiindulási pont: a leírni kívánt objektum

Például: körvonal, maszk, maszk alatti képrész

- ▶ Terület: `contourArea(kontúr, orientáció);`
- ▶ Kerület: `arcLength(kontúr, zárt-e);`
- ▶ Befoglaló téglalap: `boundingRect(kontúr);`
- ▶ Minimális területű befoglaló téglalap:  
`RotatedRect minAreaRect(kontúr);`
- ▶ Befoglaló ellipszis:  
`RotatedRect fitEllipse(kontúr)`
- ▶ Befoglaló kör:  
`minEnclosingCircle(InputArray kontúr, Point2f& kp, float& sugar);`

# Alakleírók II.

▶ konvexitás =  $\frac{\text{konvex burok kerülete}}{\text{objektum kerülete}}$

cv::convexHull(contour, convHull, false, true );

◦ cirkularitás =  $\frac{\text{terület}}{\text{kerület}^2}$  vagy  $\frac{4\pi \cdot \text{terület}}{\text{kerület}^2}$  (körre maximális)

◦ kompaktság =  $\frac{\text{kerület}^2}{\text{terület}}$  vagy  $\frac{4 \cdot \text{terület}}{\pi \cdot \text{átmérő}}$  vagy  $\frac{\sqrt{4\pi \cdot \text{terület}}}{\text{átmérő}}$

◦ átmérő =  $\max_{p,q \in C} d(p, q)$  C: kontúr

◦ ekvivalensKörAtmerő =  $2\sqrt{\frac{\text{terület}}{\pi}}$

◦ szálHossz =  $0.25(\text{kerület} + \sqrt{\text{kerület}^2 - 16 \cdot \text{terület}})$

◦ szálSzelesség =  $0.25(\text{kerület} - \sqrt{\text{kerület}^2 - 16 \cdot \text{terület}})$

◦ excentricitás =  $\frac{\text{főtengely hossza}}{\text{melléktengely hossza}}$

- Tengelyek meghatározása pl. a befoglaló ellipszis alapján vagy a leghosszabb obj-n belüli szakaszt véve főtengelynek, és arra merőleges leghosszabbat melléktengelynek.

# Momentumok

## ▶ OpenCV-ben

Moments  $m = \text{moments}(\text{kép/kontúr, bináris-e?});$  //bináris-e a kép (kontúrnál alap)

Bináris-e: NEM  $\rightarrow I(x,y)$  a világosságkód érték,  $x, y$  – a kép koordinátái

Bináris-e: IGEN  $\rightarrow x, y$  csak az objektumpontok koordinátái,  $I(x,y) = 1$ -nek tekintve

## ▶ Jellemzők:

$m.mpq$ , ahol  $pq$ : 00, 10, 01, 11, 20, 02, 21, 12, 30, 03

- A  $p+q$ -adrendű térbeli (geometriai) momentum számítása

$$m_{pq} = \sum_x \sum_y x^p y^q I(x, y)$$

## ▶ Súlypont számítása:

$$C(x, y) = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$$

(Bináris képnél a koordináták átlaga, szürkeskálásnál az intenzitásértékkel súlyozott átlaga)

# Centrális momentumok

- ▶ OpenCV-ben

Moments  $m = \text{moments}(\text{kép}, \text{bináris-e});$  //bináris-e: maszk vagy kép

Bináris-e: NEM  $\rightarrow I(x,y)$  a világosságkód érték,  $x, y$  – a kép koordinátái

Bináris-e: IGEN  $\rightarrow x, y$  csak az objektumpontok koordinátái,  $I(x,y) = 1$ -nek tekintve

- ▶ Jellemzők:

$m.\text{mu}_{pq}$ , ahol  $pq$ : 11, 20, 02, 21, 12, 03, 30

- A  $p+q$ -adrendű centrális momentum számítása

$$mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y)$$

# Normalizált centrális momentumok

- ▶ OpenCV-ben

Moments  $m = \text{moments}(\text{kép, bináris-e});$

Bináris-e: NEM  $\rightarrow I(x,y)$  a világosságkód érték,  $x, y$  – a kép koordinátái

Bináris-e: IGEN  $\rightarrow x, y$  csak az objektumpontok koordinátái,  $I(x,y) = 1$ -nek tekintve

- ▶ Jellemzők:

$m.\text{nu}_{pq}$ , ahol  $pq$ : 11, 20, 02, 21, 12, 30, 03

- A  $p+q$ -adrendű normalizált centrális momentum számítása

$$\text{nu}_{pq} = \frac{\mu_{qp}}{m_{00}^{1+(p+q)/2}}$$

a  $p+q$ -ad rendű centrális momentum

Az  $\text{nu}_{00}=1$  és az  $\text{nu}_{10}=\mu_{10}=\mu_{01}=\mu_{10}=0$  ezért nem tárolja őket az opencv.

# HU momentumok

## ▶ OpenCV-ben

```
Moments m = moments(kép, bináris-e); //bináris-e: maszk vagy kép
double hu[7]; //momentumok tárolására
HuMoments humoments(m, hu); //HU momentumok számítása
```

- ▶ Normalizált centrális momentumokból számolhatók
- ▶ Affin transzformációkra invariáns (skálázás, forgatásra, ...)  
(digitális világ: az eredeti és az elfogatott/skálázott... obeejektum HU momentumai különbözhetnek)
- ▶ A tulajdonságaiból következően
  - hasznos pl. ha a felismerendő alakzat forgatás invariáns
  - ne használd, ha a felismerendő alakzatnak fontos a mérete, orientációja

# Klaszterezés alakleírók alapján

- ▶ Töltsd le az alábbi sorozatot

[https://arato.inf.unideb.hu/szeghalmy.szilvia/kepfeld/img/kekszek\\_vegyes.zip](https://arato.inf.unideb.hu/szeghalmy.szilvia/kepfeld/img/kekszek_vegyes.zip)

- ▶ 18 kép van a mappában
- ▶ 3 választott alakleírót kell használnod
- ▶ Hozd létre a jellemzőmátrixot: 18 sor, 3 oszlop CV\_32F
- ▶ Olvasd be az  $i$ . képet ( $i = 1, 2, \dots, 18$ )
  - Küszöböld (javasolt módszer: THRESH\_OTSU | THRESH\_INV)
  - Eressz rá egy mediánszűrőt, pl. 5x5-ös ablakmérettel.
  - findContours-sal szedd ki a külső kontúrt (az obj. kontúrt használd a jellemzők számításához)
  - A jellemzőmátrix megfelelő  $(i-1)$ . sorát töltsd fel az általad választott jellemzőkkel.  
data.at<float>(i-1, 0) = \_\_\_\_\_  
data.at<float>(i-1, 1) = \_\_\_\_\_  
data.at<float>(i-1, 2) = \_\_\_\_\_

# Klaszterezés alakleírók alapján

- ▶ Hívd meg a `kmeans` függvényt.
- ▶ A középpontok most nem hordoznak jól vizualizálható értéket, ezért nem kell lekérni:

```
TermCriteria krit(
 TermCriteria::Type::EPS | TermCriteria::Type::MAX_ITER,
 100, 0.001);

Mat bestLabels;
kmeans(data, 3, bestLabels, krit, 5,
 KMEANS_RANDOM_CENTERS, noArray());
```

- ▶ Olvasd be újra a képeket ugyanolyan sorrendben, ahogy korábban tetted.
- ▶ Mentsd el a képet különböző mappákba/vagy jelenítsd meg különböző címkékkel a *kmeans* által adott besorolás alapján:  
 az *i*. kép besorolását a `bestLabels.at<int>(i)` adja meg
- ▶ Hasonlítsd össze az eredményt a szomszédodéval. Ki választott jobb jellemzőket?



# Klaszterezés alakleírók alapján

- ▶ Nézzük meg, hogy segít-e ha standardizáljuk az adatokat
- ▶ Építsük be az alábbi lépéseket a klaszterezés elé:
  - Az adat mátrix minden oszlopára (egy-egy jellemző:  $F$ ):
    - Határozzuk meg az átlagértéket ( $\mu$ ) és a szórást ( $\sigma$ )
    - Számoljuk ki az új értékeket :

$$F' = (F - \mu) / \sigma$$

```
for (int i = 0; i < data.cols; ++i) {
 Scalar mean, dev;
 meanStdDev(data.col(i), mean, dev);
 data.col(i) = (data.col(i) - mean) / dev;
}
```

# Klaszterezés alakleírók alapján

- ▶ Hívd meg a `kmeans` függvényt.
- ▶ A középpontok most nem hordoznak jól vizualizálható értéket, ezért nem kell lekérni:

```
TermCriteria krit(
 TermCriteria::Type::EPS | TermCriteria::Type::MAX_ITER,
 100, 0.001);

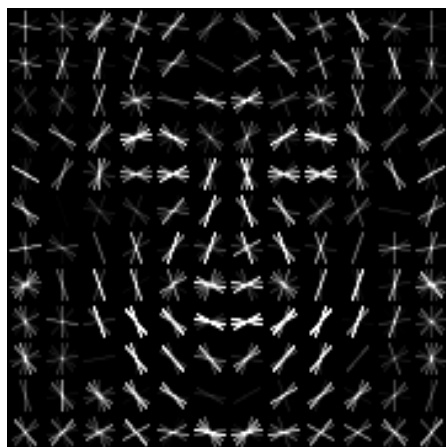
Mat bestLabels;
kmeans(data, 3, bestLabels, krit, 5,
 KMEANS_RANDOM_CENTERS, noArray());
```

- ▶ Olvasd be újra a képeket ugyanolyan sorrendben, ahogy korábban tetted.
- ▶ Mentsd el a képet különböző mappákba/vagy jelenítsd meg különböző címkékkel a *kmeans* által adott besorolás alapján:  
 az *i*. kép besorolását a `bestLabels.at<int>(i)` adja meg
- ▶ Hasonlítsd össze az eredményt a szomszédodéval. Ki választott jobb jellemzőket?

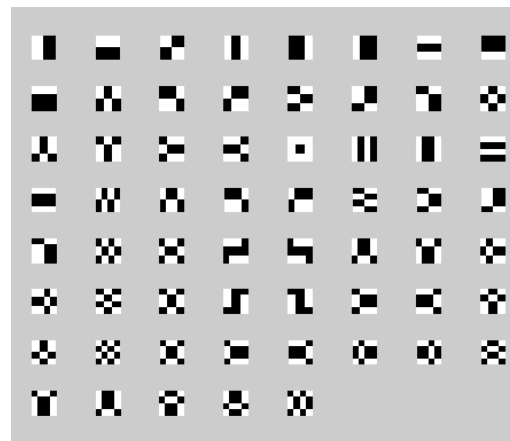
# Ismertebb textúraleírók

- ▶ LBP (local binary pattern) + hisztogram
- ▶ GLCM: (graylevel co-occurrence matrix) szürkeskálás együttes előfordulási mátrixra épülő (Harlick) jellemzők
- ▶ HOG: Gradiens-irány hisztogram
- ▶ Haar jellemzők
- ▶ Szűrőrendszerek által adott jellemzők

HOG



Haar



# LBP (Local Binary Pattern)

- ▶ A középpontot és a szomszédos pontok hasonlítása:

$$\text{LBP}_8 = (n1 > p) (n2 > p) (n3 > p) \dots (n8 > p)$$

true: 1; false: 0

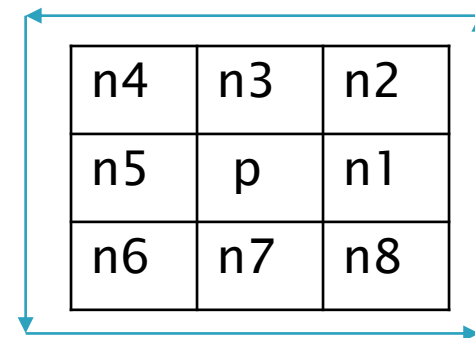
- ▶ Példa:

$$\text{LBP}_8 = (180 > 160)(140 > 160)(200 > 160) \dots (200 > 160) =$$

**10110001**

Kettes számrendszerbeli előjelnélküli egésznek tekinthetjük a kapott eredményt.

Leíró készítése: a leírni kívánt ablakot kisebb környezetekre, tipikusan 8x8-as vagy 16x16-os blokkokra osztjuk. Minden pontra kiszámítjuk az LBP-t és hisztogramot készítünk.



|     |     |     |
|-----|-----|-----|
| 210 | 200 | 140 |
| 140 | 160 | 180 |
| 138 | 144 | 200 |

*Megj: A klasszikus eljárás 8 szomszéddal dolgozik, de sok variáció van*

# GLCM: Gray Level Co-occurrence Matrix

- ▶ A GLCM mátrix megadja, hogy egy  $i - j$  intenzitásértékpár hányszor fordul elő a képen  $\alpha$  irányba  $d$  távolságra.
- ▶ A mátrix építés előtt az intenzitásértékek számát csökkentjük a képen, pl. 4, 8 szintes képekkel dolgozunk (  $I/= 64$ ;  $I/= 32$  )
- ▶ Pl: 4 szintes kép;  $\alpha = 0$  (vízszintes);  $d = 1$  (egy távolság)

|   |   |   |   |
|---|---|---|---|
| 2 | 0 | 2 | 1 |
| 1 | 2 | 1 | 1 |
| 3 | 2 | 0 | 3 |

|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 2 | 2 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |

# GLCM: Gray Level Co-occurrence Matrix

- ▶ Gyakran használt Harlick jellemzők:  
(L a kép szintjeinek száma => L×L-es a GLCM mátrix)

$$Contrast = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} (GLCM(i, j) (i - j)^2)$$

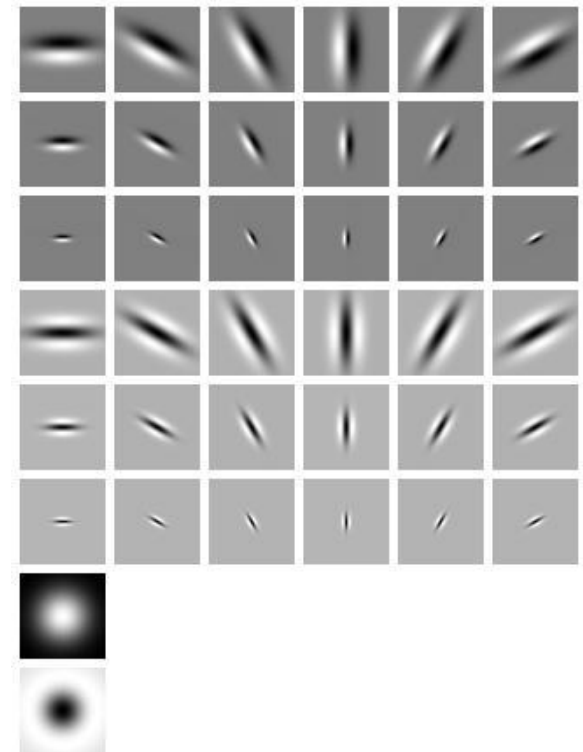
$$Energy = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} GLCM(i, j)^2$$

$$Entropy = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} (GLCM(i, j) (-\log_2 GLCM(i, j)))$$

$$Homogeneity = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} \frac{GLCM(i, j)}{1 + |i - j|}$$

# Feladat: Klaszterezés textúra alapján, szűrőrendszerrel

- ▶ Használj egy olyan képet, amelyen többféle textúra látható. Pl.:  
texture.png
- ▶ Töltsd le a Feature.h-t és a Feature.cpp-t. Add hozzá a projektedhez.
- ▶ A *MaximumResponse8* függvényét fogjuk használni:  
Az ábrán egy sorban lévő szűrők egymás elforgatottjai, ezek közül csak annak a szűrőnek az eredményét használjuk, amelyik a maximális választ adja.
- ▶ A függvény szürkeskálás képet vár és 8 képet ad vissza egy vektorban. `vector<Mat> dest;`



# Feladat: Klaszterezés textúra alapján

- ▶ Készítsünk egyetlen 8 csatornás mátrixot a vektorokból (merge)
- ▶ Készíts belőle sorvektort:
  - `Mat data = merged_mat.reshape(1, img.rows*img.cols)`

- ▶ Klaszterezz ( $k = 5$ )

```
TermCriteria krit(TermCriteria::Type::EPS |
 TermCriteria::Type::MAX_ITER, 100, 0.001);
```

```
Mat bestLabels;
kmeans(data, 5, bestLabels, krit, 5, KMEANS_RANDOM_CENTERS,
 noArray());
```

A bestLabels-t formázzuk képpé:

```
Mat dest1 = bestLabels.reshape(1, img.rows)
```

Konvertáljuk CV\_8U-ra, hogy meg tudjuk jeleníteni az eredményt:

```
Mat dest2;
dest1.convertTo(dest2, CV_8UC3, 50);
//az 50-es szorzó, csak a jobb láthatóság miatt van
```



# Feladat: Klaszterezés textúra alapján

- ▶ Megjelenítés:
  - ▶ Hozz létre egy eredeti képpel azonos méretű képet (CV\_8UC3)
  - ▶ Hozz létre egy legalább K méretű Vec3b tömböt és töltsd fel véletlenszerű színekkel.
  - ▶ Járd be a kmeans által megadott címkéket (korábban bestLabels)
    - ▶ Ha a címke értéke  $[0, K[$  tartományba esik, akkor rendeld hozzá a megfelelő képponthoz a címkének megfelelő színt.
    - ▶ Egyébként rendelj hozzá tiszta feketét.

# Feladat: Klaszterezés textúra alapján

- ▶ Megjelenítés:
  - ▶ Hozz létre egy eredeti képpel azonos méretű képet (CV\_8UC3)
  - ▶ Hozz létre egy legalább K méretű Vec3b tömböt és töltsd fel véletlenszerű színekkel.
  - ▶ Járd be a kmeans által megadott címkéket (korábban bestLabels)
    - ▶ Ha a címke értéke  $[0, K[$  tartományba esik, akkor rendeld hozzá a megfelelő képponthoz a címkének megfelelő színt.
    - ▶ Egyébként rendelj hozzá tiszta feketét.

