Rozpoznávání

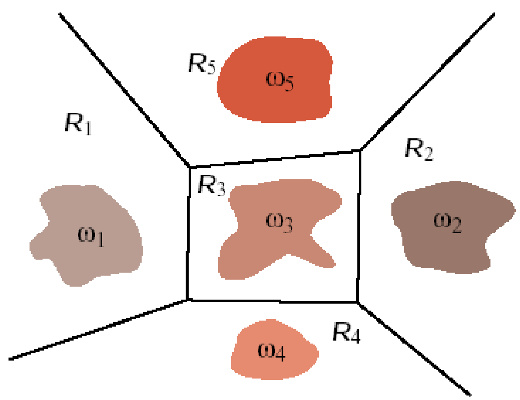
- cvičení

UTIA - ZOI

Adam Novozámský
Jitka Kostková
{novozamsky, kostkova}
@utia.cas.cz

Motivace

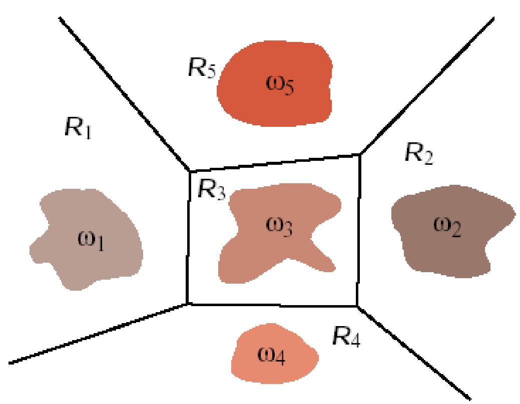
- **rozpoznávání** je **rozhodování**, jestli objekt patří do dané **třídy**
- objekt je popsán množinou **příznaků** (*n-D vektor v metrickém prostoru*)
- Jaké známe typy klasifikace/rozpoznávání?
  - rozpoznávání řízené (s **učením**) – pro ∀ třídy máme typickou množinu reprezentantů (**trénovací množina**)
  - rozpoznávání neřízené (bez **učení**) – nemáme ani trénovací množinu, ani nevíme kolik je tříd



Trénovací množina

- **reprezentativní** – typické vzorky dané třídy, všechny hlavní typy, neměly by tam být jiné vzorky
- **dostatečně velká** – k podchycení vnitřní variability
- měl by ji sestavovat odborník v dané oblasti

- ***Formální definice klasifikátorů:***
  - Každá třída je charakterizována diskriminační fcí $g(x)$
  - Klasifikace = maximalizace $g(x)$



Jaké máme klasifikátory?

- **NN-klasifikátor** (NN = **n**earest **n**eighbor)

$$g(x) = \frac{1}{dist(x, w)}$$

- Nevýhoda ?:
  - extrémně citlivá na chyby v trénovací množině a na extrémy

- Jak modifikovat ?:
  - nejbližší vzdálenost k těžištím množin - nerespektuje tvar ani počet prvků množin
  - ***k-NN:*** *k*-nejbližších bodů jedné třídy
    - Jak správně volit ***k***?
      - řádově menší než počet prvků v trénovací množině $(\mathbf{k} = \langle \mathbf{2}, \mathbf{5} \rangle)$

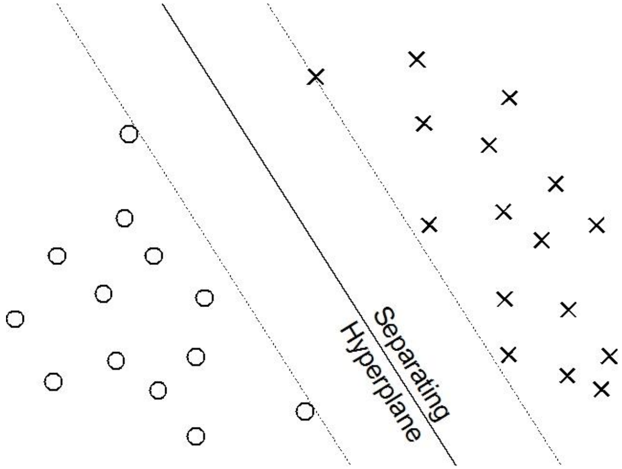
- Co se stane, pokud máme třídy, ve kterých je vždy jen jeden bod?
  - vznikne taková mozaika ~ **Voronoiovy polygony**

Další klasifikátory

- **lineární klasifikátor:**
  - mezi 2 třídami vede vždy jen jedna nadrovina – přímka
  - jednodušší hledání hranic, ale klasifikace nemusí být správná
- <u>Jak byste hranice hledali ?:</u>
  - začnu osou mezi dvěma body z různých tříd, postupně přidávám další body:
    - když padají na správnou stranu, nic s přímkou nedělám, začnu ji posouvat a naklánět teprve, až se trefím na špatnou stranu
    - lepší je ale upravovat přímku vždy, i když padají nové body na správnou stranu (*např. minimalizace rozdílu středních vzdáleností od přímky*)

Další klasifikátory

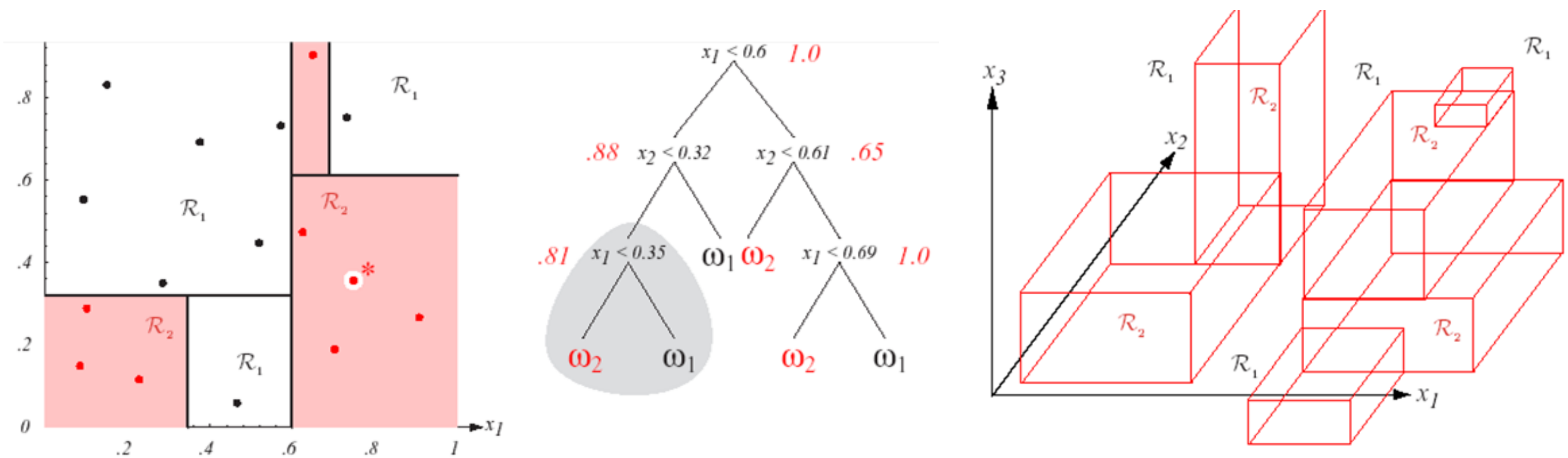
- ***SVM (support vector machine)***
  - snaží se konstruovat 2 rovnoběžné nadroviny tak, aby separovaly třídy a byly co nejdále od sebe
  - body, které tyto nadroviny protínají, se nazývají **support vectors**
  - vlastní rozhodovací nadrovina je s nimi rovnoběžná a vede mezi nimi



  - Nevýhody:
    - support v. jsou většinou extremální body
    - nezohledňuje počty bodů v množinách – rozhodovací přímku posunu v poměru k té množině, kde je více prvků
    - často nemusí existovat dvě rozdělující přímky, pokud nejsou třídy lineárně separovatelné
    - programování je náročné, protože se musejí prozkoušet všechny možnosti

Další klasifikátory

- **rozhodovací stromy:**
  - tam, kde je těžké určit metriku
  - kořen stromu je neznámý vstupní prvek, listy jsou jednotlivé třídy
  - každý rozhodovací strom se dá přepsat do binárního
  - trénování spočívá v sestavování stromu a nastavování podmínek
  - při reálných příznacích se rozhoduje na základě nerovností
  - rozhodovací hranice = hyperkvádry v prostoru



Další klasifikátory

- **Bayesův klasifikátor**

$$P(\omega_j|X) = \frac{p(X|\omega_j)P(\omega_j)}{p(X)}$$

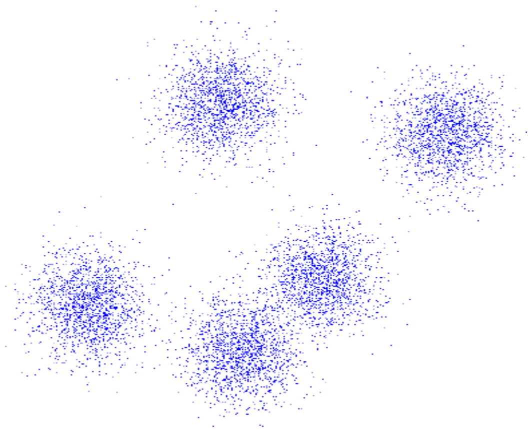
  - $p(\omega_j|X)$ … podmíněná pravděpodobnost, že se ve třídě ω_j může vyskytnout prvek X
  - $P(\omega_j)$ … pravděpodobnost i-té třídy v Ω (v reálu)
  - $p(X|\omega_j)$ … pravděpodobnost, že na prvku ze třídy *i* můžeme naměřit vektor X
  - $p(X) = \sum_{j=1}^{C} p(X|\omega_j)P(\omega_j)$ … celková pravděpodobnost

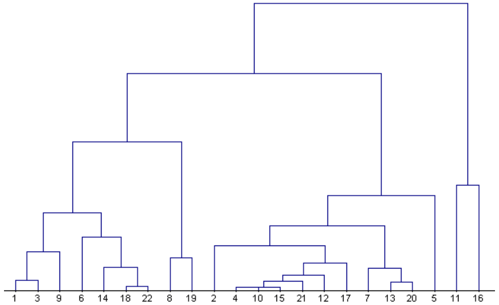
Klasifikace bez učení

- **Shluková analýza** (***clustering***)

  - Jednoduché **Wardovo kritérium**: $J = \sum_{i=1}^{N} \sum_{x \in C_i} \|x - \mu_i\|^2$

  - <u>iterační metody:</u>
    - ***N-Means Clustering***
    - ***Iterativní minimalizace J***
  - <u>hierarchické metody:</u>
    - ***Aglomerativní***
    - ***Divizivní***





Příznaky

- obecné požadavky ? :
  - **Diskriminabilita** – objekty patřící do různých tříd, by měly mít různé hodnoty příznaků (invariance jde většinou proti diskriminalitě)
  - **Robustnost** – měli bychom zajistit jen malé nepřesnosti; měly by být dosti robustní na šum
  - **Efektivnost**
  - **Nezávislost** – žádná složka vektoru příznaků není funkce jiných
  - **Úplné** – daný objekt lze přesně zrekonstruovat pomocí těchto příznaků
- Jaké známe příznaky ? :
  - **vizuální**
  - **transformační koeficienty**
  - **diferenciální**
  - **momentové**

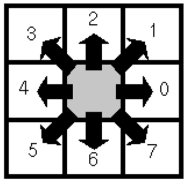
Vizuální příznaky

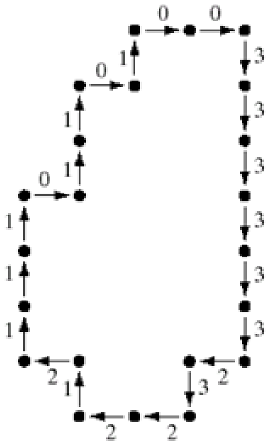
- **Kompaktnost**
  - $\frac{4\pi P}{O^2}$ … P je plocha a O je obvod
  - jde o míru podobnosti ke kruhu, kde kruh má hodnotu "1"
- **Konvexita**
  - $\frac{P(A)}{P(C_A)}$ ... jde o míru podobnosti ke konvexnímu obalu
- **Elongation (podlouhlost)**
  - poměr krátké a dlouhé strany >> míra podobnosti ke čtverci
- **Podobnost obdélníku (rectangularity)**
  - poměr plochy objektu a opsaného obdélníku >> míra podobnosti k obdélníku
- **Eulerovo číslo** – počet komponent mínus počet děr

Vizuální příznaky se někdy používají jako předklasifikace.

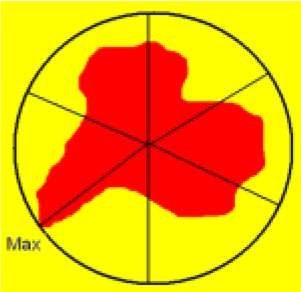
Úplné vizuální příznaky

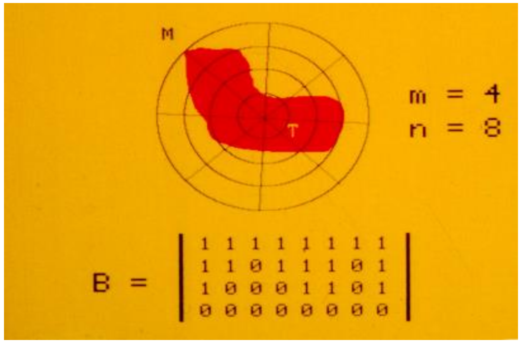
- **Řetězový kód (Chain code)**





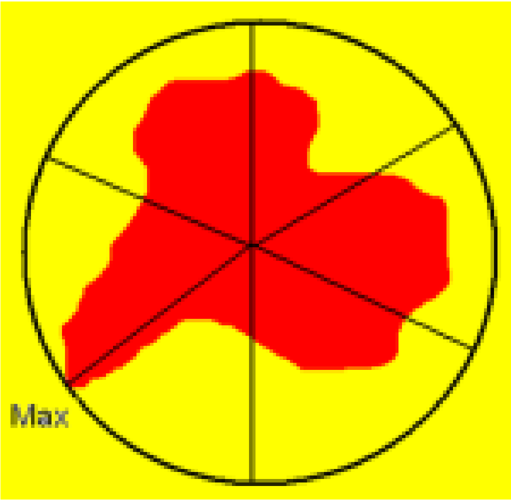
- **Polygonální aproximace**
  - nahrazuje hranici polygonem
- **Tvarový vektor (Shape vector)**
  - převzorkování v polárních souřadnicích
  - najdu těžiště
  - najdu nejvzdálenější bod od těžiště
  - vzdálenost těžiště a tohoto bodu bude poloměr kružnice
  - udělám kružnici se středem v těžišti
  - rozdělím ji na stejné výseče





Úplné vizuální příznaky

- **Tvarový vektor (shape vector)**



- Je invariantní:
  - na posun ?
    - vztaženo k těžišti
  - na otáčení ?
    - vztaženo k maximu
  - na změnu měřítka ?
    - ano, pokud vektor normalizuji první složkou

Fourierovy deskriptory

- patří do skupiny transformačních koeficientů
  - (*stejně jako wavelet transform*)
- založeny na ***Fourier shift teorému*** (FST):

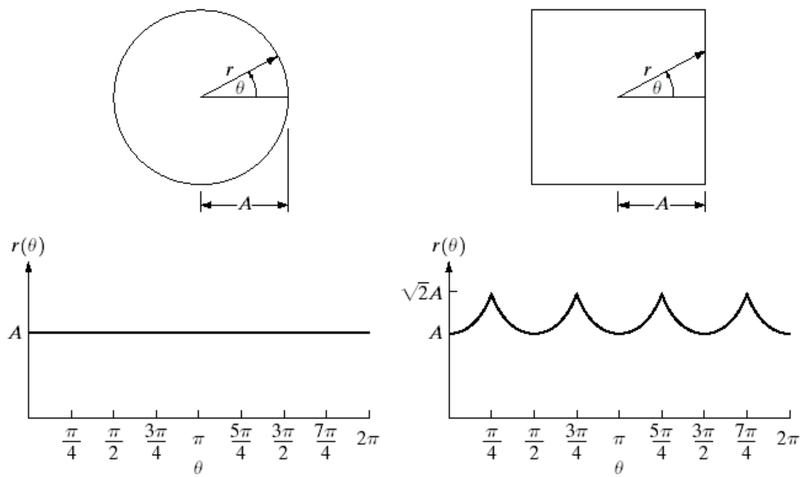
$$\mathcal{F}_{\boldsymbol{x}}[\boldsymbol{f}(\boldsymbol{x}-\boldsymbol{x}_{\mathbf{0}})](k) = e^{-2\pi i k x_0} F(k)$$

  - fourierka posunuté fce ***je jen násobkem fourierky té původní***
- Amplituda FT se při posunu nemění, fáze se definovaně posouvá

Fourierovy deskriptory

$$\mathcal{F}_x[f(x - x_0)](k) = e^{-2\pi i k x_0} F(k)$$

- Jak se FST využije ? :
  - Zkonstruujeme radiální fci:



  - radiální fce je invariantní k:
    - **posunutí** – protože to vztahuji k těžišti, nemusím uvažovat o posunutí
    - **rotaci** – radiální fce se bude pouze posouvat – tedy nezávisí na startovním bodu
  - udělám **FT** radiální fce a vezmu její amplitudu – **prvních pár koeficientů FT** prohlásím za naše hledané FOURIEROVY DESKRIPTORY

Fourierovy deskriptory

- Abychom zajistili invarianci ke změně měřítka, dělí se tato sada prvním koeficientem, což je koeficient konstantní fce – neboli střední hodnota fce:

$$F(n) = \int f(t)e^{-2\pi i n t}dt \qquad F(0) = \int f(t)dt$$

- Funguje jen pro hvězdicovité objekty a ve spojitém případě.

- ***Použití v praxi (diskrétní případ):***
  - vezmeme hranici a představíme si ji jako komplexní funkci:

$$f(t) = x(t) + iy(t)$$

  - z ní se spočítá FT a vezmou se absolutní hodnoty
  - nultý koeficient má nyní jiný význam – říká nám vzdálenost od počátku, proto jej zahodíme a používáme až ty další

Pozn.: Ve F. deskriptorech moc informace není – u FT je podstatná část informace ve fázi, kterou vůbec neuvažujeme.

Stáhněte si balíček se zadáním:

http://zoi.utia.cas.cz/ROZ2/studijni-materialy

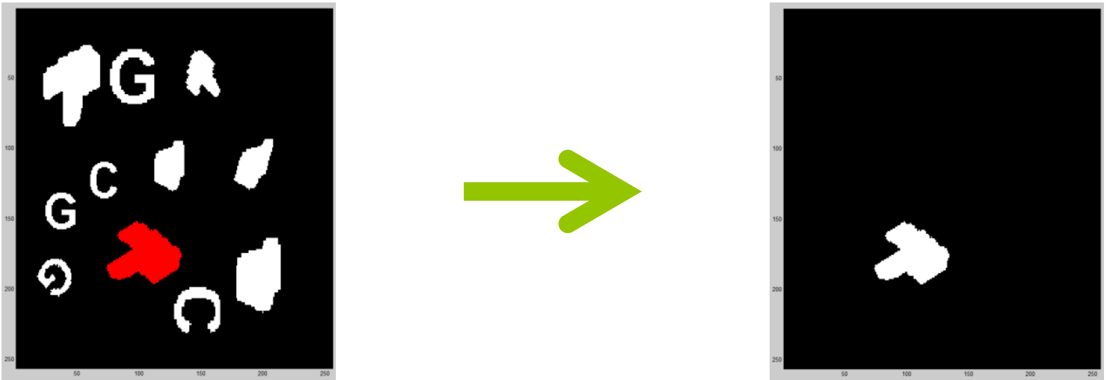
- **segm.pgm**

- **huc.m** – počítá Huovy momentové inv. z _centrálních m._
- **hun.m** – počítá Huovy m. i. z _normalizovaných m._
- **iboundary.m** – vnitřní hranice objektu S ~= 0
- **label.m** – „olabeluje“ (*označkuje*) segmenty obrazu
  - [CS, Counter] = label(Img, Pic)
    - přiřadí cifry oblastem binárního snímku >> `zobr(CS==i)`
- **zobr.m**

Programovaní (zahřívací kolo):

- doporučuji si vytvořit skript **`doAll.m`**, kam budete psát jednotlivé mezikroky, protože je budete opakovaně volat…

- vyzkoušejte si přiloženou funkci **`label(I)`**
  - zobrazte např. jen tento červený objekt:



```
[CS, N] = label(Img, 1);
zobr(CS==5);
```

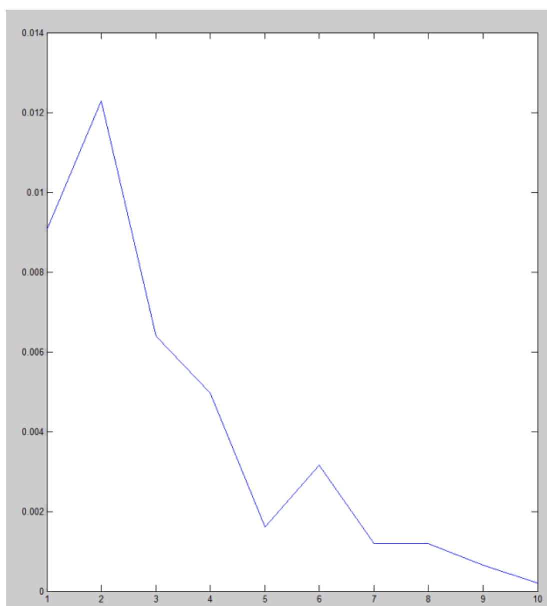
Programovaní:

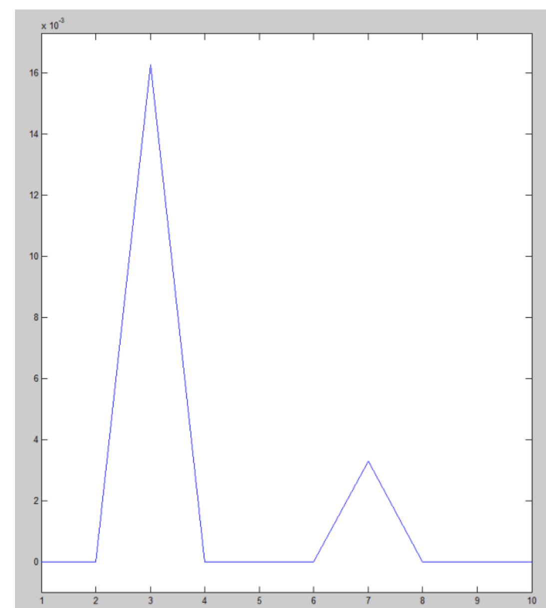
- Napište funkci na spočtení **prvních N Fourierových deskriptorů** binárního objektu:

```
function R = fourDesc(I, N)
% vraci N fourierovych dekriptoru snimku I
B = iboundary(I);
X = B(:,2);
Y = B(:,1);
F = X + 1i*Y;
FT = abs (fft(F));
R = FT(2:N+1);
R = R / length(X)^2;
```

Programovaní:

```
>> FD = fourDesc(padarray(ones(10),[4 4]), 10);
>> plot(FD);
>> ylim([-0.001, max(FD)+0.001]);
>>
>> FD = fourDesc(CS == 5, 10);
>> plot(FD);
```





Programovaní:

- zapište příznaky všech objektů do matice příznaků
  - *řádky = záznamy*

```
[CS, N] = label(Img, 1);
for i = 1:N
  PriFD(i,:) = fourDesc(CS == i, 5);
end
```

- Zobrazte příznakový prostor:
  - napište funkci **`zobrPriz(P)`**
    - bude zobrazovat první dvě složky příznaků

```
function zobrPriz(P)
% zobrPriz(P) - zobrazi prvni dve slozky priznakovych
vektoru
figure;
plot(P(:,1), P(:,2), 'w');
for i = 1 : size (P,1)
  text(P(i,1), P(i,2), ['\times' num2str(i)]);
end
```

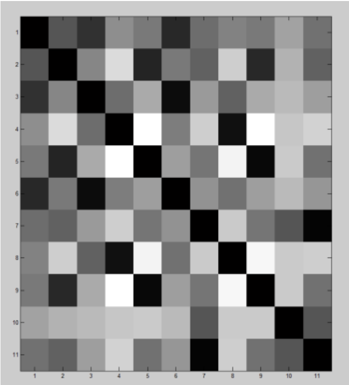
Programovaní:

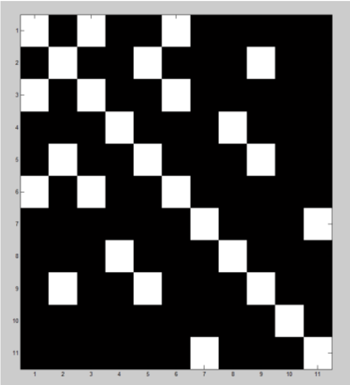
- Vytvořte funkci pro získání **distanční matice** příznakových vektorů **`distMat(Vects)`**

```
function R = distMat(Vects)
% R = distMat(Vects) - vraci distancni matici
radkovych priznakovych vektoru

N = size(Vects,1);
for i = 1 : N
  for j = 1 : N
    R(i,j) = norm (Vects(i,:)-Vects(j,:));
  end
end
```

Programovaní:

- zobrazte distanční matici a poté nalezněte práh

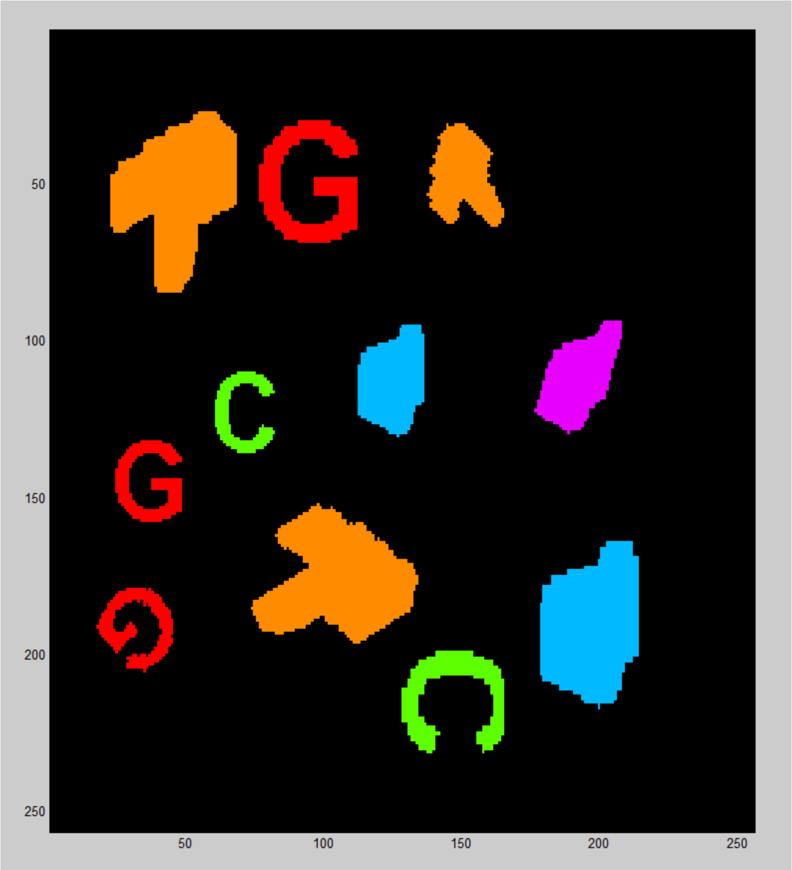




```
[CS, N] = label(Img, 1);

for i = 1:N
  PriFD(i,:) = fourDesc(CS == i, 5);
end

ClaFD = distMat(PriFD);
zobr(ClaFD < 0.007);
```

Programovaní:

- vykreslení tříd dle oprahované distanční matice
  - **`zobrTridy(CS,D)`**
    - použijte funkci `colormap([0,0,0; hsv(N)]);`
      - `N ~ počet objektů, resp. tříd`



Programovaní:

```
function zobrTridy (CS, D)
% zobrTridy(CS,D) - zobrazi segmentovany obrazek CS
klasifikovany dle distancni matice D

N = size(D,1);
for i = 1:N
  for j = i+1:N
    if D(i,j)
      CS(CS==j) = i;
    end
  end
end

figure;
image(CS+1);
colormap([0,0,0; hsv(N)]);
```

Momentové invarianty

- **momenty** jsou projekcí funkce obrázku do polynomiální báze
- Napište obecný moment $M_{pq}^{(f)}$ obrázku $f(x, y)$? :

$$M_{pq}^{(f)} = \iint_D p_{pq}(x, y) f(x, y) dxdy$$

  - $p, q \epsilon \mathbb{N}^+$
  - $r = p + q$ je stupeň momentu
    - $p_{00}(x, y), p_{10}(x, y), \cdots, p_{kj}(x, y)$ je polynomiální báze funkcí definovaných na D

Momentové invarianty

- Napište **geometrický moment** $m_{pq}^{(f)}$ obrázku $f(x,y)$

$$m_{pq}^{(f)} = \iint_{-\infty}^{\infty} x^p y^q f(x,y)\,dxdy$$

  - $m_{00}^{(f)}$ <u>? :</u>
    - "hmotnost" obrázku – pro binární obrázky je to plocha
  - <u>souřadnice těžiště ?:</u>
    - $x_t = \frac{m_{10}}{m_{00}}, y_t = \frac{m_{01}}{m_{00}}$
  - Pokud považujeme obrázek za hustotu pravděpodobnosti a normalizujeme $m_{00} = 1$, pak jsou:
    - m_{10} a m_{01} střední hodnoty
    - m_{20} a m_{02} jsou vertikální a horizontální rozptyly

Momentové invarianty

vzhledem ke geometrickým transformacím obrazu

- **invariant k T** - **centrální geometrický moment**

$$\mu_{pq} = \iint\limits_{-\infty}^{\infty} (x - x_t)^p (y - y_t)^q f(x,y)\,dxdy$$

  - kde $x_t = \frac{m_{10}}{m_{00}}$, $y_t = \frac{m_{01}}{m_{00}}$
  - ***pozn.:***
    - $\mu_{01} = \mu_{10} = 0$
    - $\mu_{00} = m_{00}$

$$\mu_{pq} = \sum_{k=0}^{p} \sum_{j=0}^{q} \binom{p}{k} \binom{q}{j} (-1)^{k+j}\, {x_t}^k {y_t}^j m_{p-k,q-j}$$

Momentové invarianty

vzhledem ke geometrickým transformacím obrazu

- **invariant k T a rovnoměrnému S** - **normalizovaný centrální moment**

$$\upsilon_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\omega}}$$

  - kde $\omega = \frac{p+q}{2} + 1$
  - ***Důkaz.:***

$$\mu'_{pq} = \iint_{-\infty}^{\infty} (x' - x'_t)^p \left(y' - y'_t\right)^q f'(x', y') dx' dy' =$$

$$= \iint_{-\infty}^{\infty} s^p (x - x_t)^p s^q (y - y_t)^q f(x, y) s^2 dx dy = s^{p+q+2} \mu_{pq}$$

    - *dále:* $\mu'_{00} = s^2 \mu_{00}$
    - *potom tedy:*
    - $\upsilon'_{pq} = \frac{\mu'_{pq}}{\mu'^{\omega}_{00}} = \frac{s^{p+q+2}\mu_{pq}}{(s^2\mu_{00})^{\omega}} = \upsilon_{pq}$
    - *z toho tedy vyplývá:*
      - $\frac{s^{p+q+2}}{s^{2\omega}} = 1 \rightarrow 2\omega = p + q + 2 \rightarrow \omega = \frac{p+q}{2} + 1$

Momentové invarianty

vzhledem ke geometrickým transformacím obrazu

- **invariant k R**
  - M.K. Hu, 1962 – 7 invariantů třetího řádu:

$$\phi_1 = \mu_{20} + \mu_{02}$$

$$\phi_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2$$

$$\phi_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2$$

$$\phi_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2$$

$$\begin{aligned}\phi_5 = {} & (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})((\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2) \\ + {} & (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})(3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2)\end{aligned}$$

$$\begin{aligned}\phi_6 = {} & (\mu_{20} - \mu_{02})((\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2) + \\ & 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03})\end{aligned}$$

$$\begin{aligned}\phi_7 = {} & (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})((\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2) \\ - {} & (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})(3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2)\end{aligned}$$

- Těžko se hledají, ale dají se lehce prokázat. Pokud do nich dosadíme transformační vztahy pro rotaci:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

Momentové invarianty
vzhledem ke geometrickým transformacím obrazu

- **invariant k R**
  - M.K. Hu, 1962 – 7 invariantů třetího řádu:
    - Problémy:
      - závislost: $\phi_3 = \frac{\phi_5^2+\phi_7^2}{\phi_4^3}$
      - neúplnost
  - proto konstruujeme rotační invarianty z **komplexních momentů**:

$$c_{pq}^{(f)} = \iint\limits_{-\infty}^{\infty} (x+iy)^p (x-iy)^q f(x,y)\,dx\,dy$$

  - Nechť $n \geq 1$ a $k_i, p_i, q_i \in \mathbb{N}^+$, i $\in \hat{n}$ a nechť $\sum_{i=0}^{n} k_i(p_i - q_i) = 0$
    - pak:

$$I = \prod_{i=1}^{n} c_{p_i q_i}^{k_i}$$

    - je **invariant k rotaci**

Programovaní:

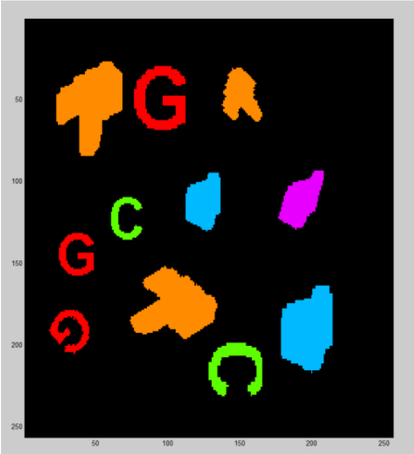
- klasifikujte objekty na obrázku **segm.pgm** pomocí **Huových centrálních** a **normalizovaných** invariantů
  - použijte huc.m a hun.m

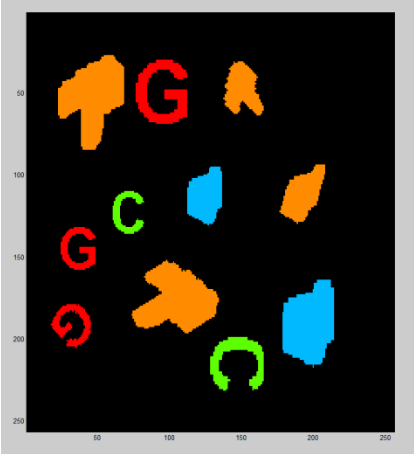
```
Img = double(~imread('segm.pgm'));
[CS, N] = label(Img, 1);
for i = 1:N
  PriFD(i,:) = fourDesc(CS == i, 5)';
  PriHun(i,:) = hun(CS == i);
  PriHuc(i,:) = huc(CS == i);
end
ClaFD = distMat(PriFD);
ClaHun = distMat(PriHun);
ClaHuc = distMat(PriHuc);

zobrTridy (CS, ClaFD < 0.007);
zobrTridy (CS, ClaHun<0.01);
zobrTridy (CS, ClaHuc<0.1e20);
```

Programovaní:

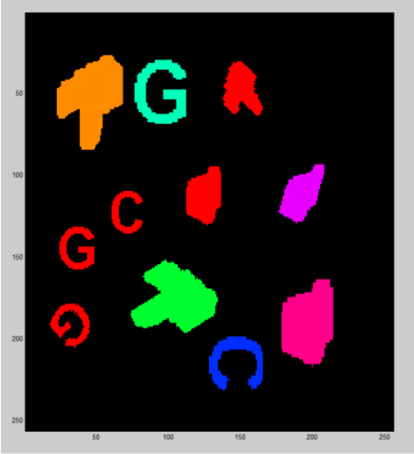
- zkuste různé hodnoty pro prahování **Huových centrálních** a **normalizovaných** invariantů

```
zobrTridy (CS, ClaFD < 0.007);
```





```
zobrTridy (CS, ClaHun<0.01);
```



```
zobrTridy (CS, ClaHuc<1e20);
```

Děkuji za pozornost !

UTIA - ZOI

Adam Novozámský
Jitka Kostková
{novozamsky, kostkova}
@utia.cas.cz