Segmentace II

Pokročilá analýza obrazu

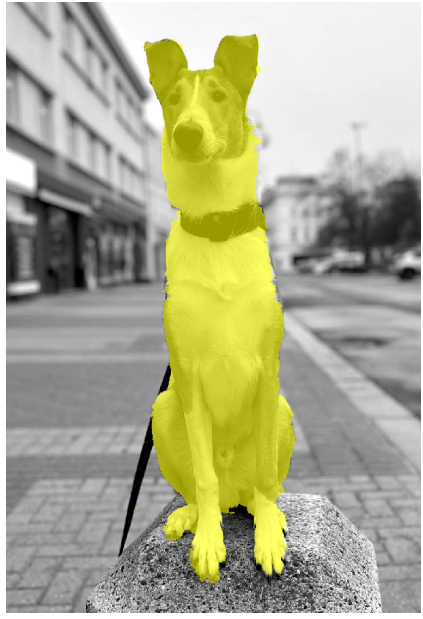
Mgr. Markéta Trnečková, Ph.D.

Obraz

- Segmentace (dělení obrazu na nějaké logické části)
  - dělení obrazu dle nějakých prudkých změn
  - dělení obrazu na regiony, které mají společné vlastnosti
  - jejich kombinace

Segmentace

- R – celý region (obraz)
- **Segmentace** – rozdělení obrazu na jednotlivé regiony $R_1, \ldots R_n$:
  - $\bigcup_{i=1}^{n} R_i = R$
  - R_i jsou spojité
  - $R_i \cap R_j = \emptyset$, pro všechna $i \neq j$
  - $Q(R_i) = TRUE$ – všechny pixely v oblasti splňují nějakou podmínku
  - sousední regiony stejnou podmínku nesplňují $Q(R_i \cup R_j) = FALSE$
- dělení obrazu na části, které splňují nějakou podmínku:
  - pixely v oblasti jsou „shodné“
  - pixely v sousedních oblastech jsou „rozdílné“



Prahování

- Díky své jednoduchosti (jak implementační, tak v časové náročnosti) je často využíváno
- Už jsme se s prahováním setkali
- Dělí obraz na regiony dle hodnot intenzity a vlastností oblastí
- **Základní prahování**:

$$T(r) = \begin{cases} s_0 & \text{pro } r < prah \\ s_1 & \text{pro } r \geq prah \end{cases}$$

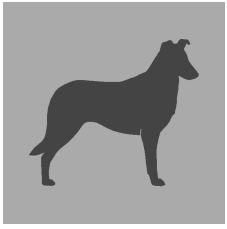
- **Volba prahu**:
  - experimentálně
  - matematicky
- Zde se práh použije na celý obrázek – **Globální prahování**
- Práh se může v obrázku měnit – **Lokální prahování** (adaptivní prahování)

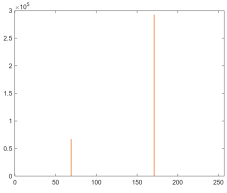
Prahování

- Kvalita prahování závisí:
  - Snadnosti separování jednotlivých peaků
  - Šumu
  - Relativní velikosti objektů a pozadí
  - Rovnoměrném osvětlení (a odrazů v obraze)

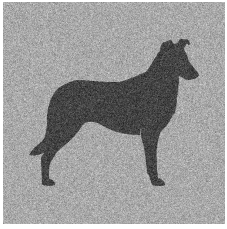
Prahování

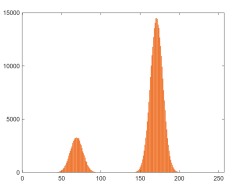
Vliv šumu



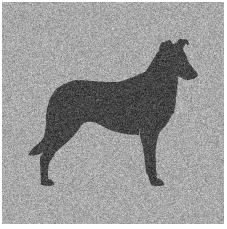


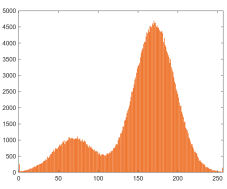
Bez šumu



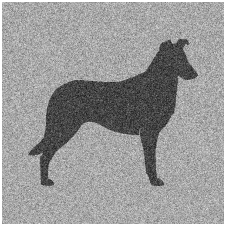


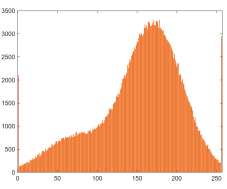
Gauss $\sigma^2 = 0.001$





Gauss $\sigma^2 = 0.01$

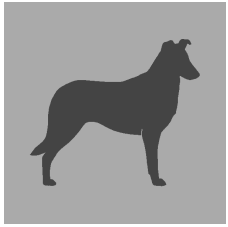


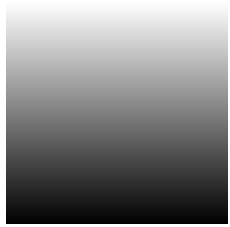


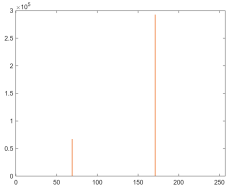
Gauss $\sigma^2 = 0.02$

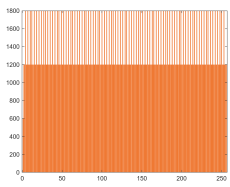
Prahování

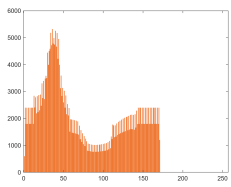
Vliv osvětlení











Originál Stín Se stínem

Globální prahování – automatický výběr prahu

- V případě bimodulárního grafu vybíráme práh mezi dvěma oblastmi
- V případě, že známe velikosti plochy, kterou objekt zabírá, využijeme této znalosti

Příklad

Jak bychom zjistili prahovou hodnotu v případě, kdy víme, jak velkou oblast objekt zabírá?

- Obecně je prahování rozdělení pixelů v obraze do dvou (a více) tříd tak, aby byly tyto dvě skupiny pixelů, co nejvíce rozdílné
- Otsu metoda je asi nejznámější z nich – maximalizuje variance mezi těmito třídami pixelů

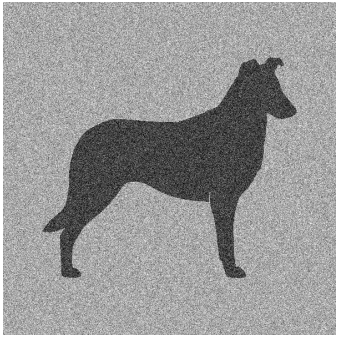
Prahování – automatický výběr prahu

Otsuova metoda

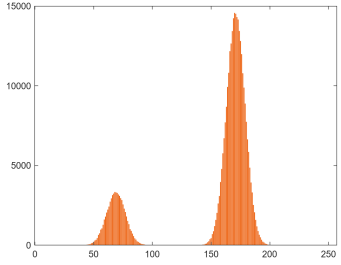
- Máme normalizovaný histogram p_i obrazu s L intenzitami:
- Zvolíme práh T, který rozdělí obraz na dvě třídy $C_0 = \{0, 1, \ldots, T\}$ a $C_1 = \{T+1, \ldots, L-1\}$
- Spočítáme váhy tříd $\omega_0(T) = \sum_{i=0}^{T} p_i$, a $\omega_1(T) = \sum_{i=T+1}^{L-1} p_i$
- Spočítáme střední hodnoty tříd $\mu_0(t) = \frac{1}{\omega_0(T)} \sum_{i=0}^{T} i\, p_i$, a $\mu_1(T) = \frac{1}{\omega_1(T)} \sum_{i=T+1}^{L-1} i\, p_i$
- Celkový průměr: $\mu_T = \sum_{i=0}^{L-1} i\, p_i$
- Otsu maximalizuje **mezitřídní rozptyl**: $\sigma_b^2(T) = \omega_0(T)\, \omega_1(T)\, (\mu_0(T) - \mu_1(T))^2$
- Optimální práh T^* je taková hodnota, pro kterou platí: $T^* = \arg\max_T \ \sigma_b^2(T)$.

Prahování – automatický výběr prahu

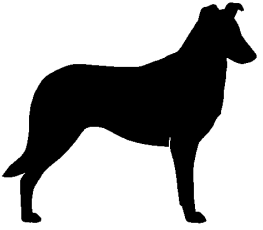
Otsuova metoda



Gauss $\sigma^2 = 0.001$



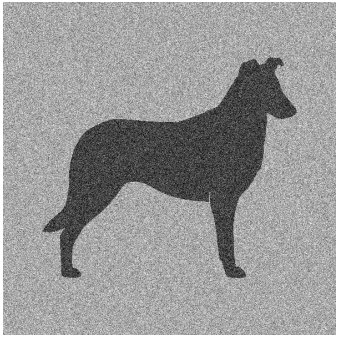
Histogram



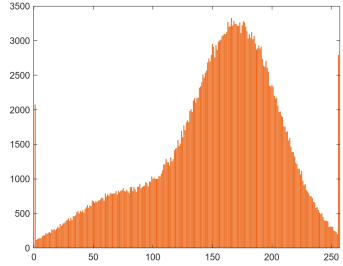
Otsuova metoda

Prahování – automatický výběr prahu

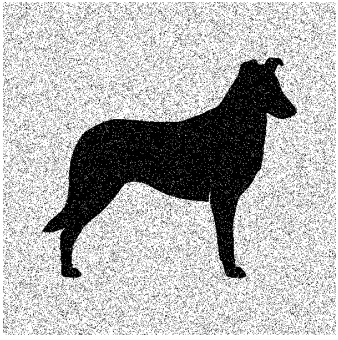
Otsuova metoda



Gauss $\sigma^2 = 0.02$



Histogram

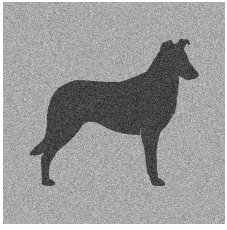


Otsuova metoda

Prahování

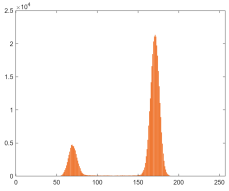
Vylepšení prahování vyhlazením

- V předchozím příkladu vidíme, že šum znemožňuje optimální prahování
- Myšlenka je snížit vliv šumu – už víme, že to můžeme udělat pomocí filtrování



Gauss $\sigma^2 = 0.02$

Vyhlazený obrázek



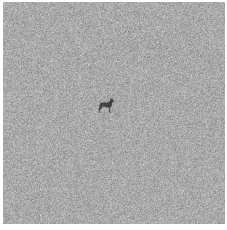
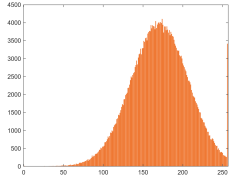
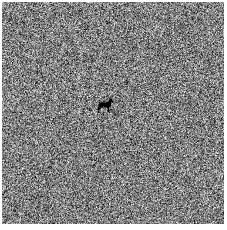
Histogram

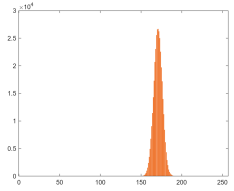
Otsuova metoda

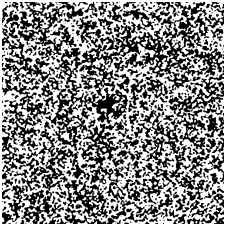
Prahování

Vylepšení prahování vyhlazením

- Další problémem Otsu metody spolu s šumem je, pokud je popředí/pozadí moc malé, pak šum představuje v histogramu větší informaci
- ani vyhlazení nepomůže





Prahování

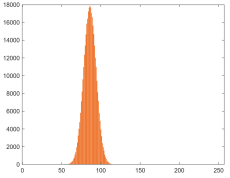
Vylepšení prahování hledáním hran

- Hledání optimálního prahu je snadné, pokud je histogram bimodulární s dvěma přibližně stejnýma peaky
- Myšlenka: k hledání prahu využijeme jen ty pixely, které leží na hraně, nebo v její blízkosti
- Histogram bude méně závislý na relativní velikosti objektů
- To, zda jsou pixely blízko hran se dá spočítat pomocí gradientu nebo Laplaceova operátoru
- **Algoritmus**:
  1. Spočítáme hrany (buď magnitudu gradientu, nebo absolutní hodnotu po použití Laplaceova operátoru)
  2. Specifikujeme prahovou hodnotu T
  3. Naprahujeme obrázek z kroku 1 prahovou hodnotou z kroku 2 $\rightarrow g_T(x, y)$. Výsledný obrázek použijeme jako masku v následujícím kroku.
  4. Spočítáme histogram pouze z pixelů $f(x, y)$ odpovídajících masce $g_T(x, y)$
  5. Z tohoto histogramu pak pomocí Otsu spočítáme ideální prahovou hodnotu

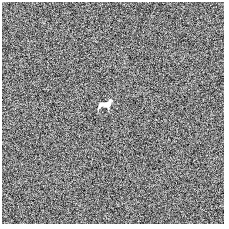
Prahování

Vylepšení prahování hledáním hran

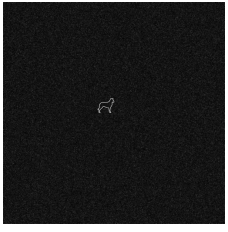
Původní obrázek



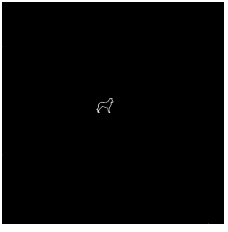
Histogram



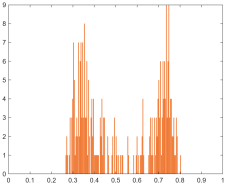
Otsu



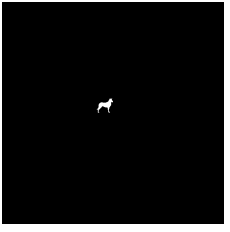
Hranový obrázek



Maska



Histogram



Otsu

Vícestupňové prahování

$$T(r) = \begin{cases} s_0 & \text{pro } r < T_1 \\ s_1 & \text{pro } T_1 \leq r < T_2 \\ \dots & \\ s_n & \text{pro } T_n \geq r \end{cases}$$

- Otsuovu metodu je možné rozšířit na prahování libovolným počtem prahů
- Histogram dělíme na tolik tříd, kolik potřebujeme
- Opět hledáme takové dělení, které maximalizuje mezitřídní varianci
- Pokud prahujeme více prahy, mnohdy hledáme dělení nejen na základě intenzity, ale i jiných vlastností

Prahování – automatický výběr prahu

Otsuova metoda 2 prahy

- Máme normalizovaný histogram p_i obrazu s L intenzitami:
- Zvolíme prahy T_1 a T_2, které rozdělí obraz na tři třídy
  $C_0 = \{0, 1, \ldots, T_1\}$, $C_1 = \{T + 1, \ldots, T_2\}$ a $C_1 = \{T_2 + 1, \ldots, L - 1\}$
- Spočítáme váhy tříd
  $\omega_0 = \sum_{i=0}^{T_1} p_i,$, $\omega_1 = \sum_{i=T_1+1}^{T_2} p_i$ a $\omega_2 = \sum_{i=T_2+1}^{L-1} p_i$
- Spočítáme střední hodnoty tříd
  $\mu_0 = \frac{1}{\omega_0} \sum_{i=0}^{T_1} i\, p_i,$, $\mu_1 = \frac{1}{\omega_1)} \sum_{i=T_1+1}^{T_2} i\, p_i$ a $\mu_2 = \frac{1}{\omega_2} \sum_{i=T_2+1}^{L-1} i\, p_i$
- Celkový průměr:
  $\mu = \sum_{i=0}^{L-1} i\, p_i$
- Otsu maximalizuje **mezitřídní rozptyl**:
  $\sigma_b^2(T_1, T_2) = \omega_0(\mu_0 - \mu)^2 + \omega_1(\mu_1 - \mu)^2 + \omega_2(\mu_2 - \mu)^2$
- Optimální je taková dvojice (T_1^*, T_2^*), pro kterou platí:
  $(T_1^*, T_2^*) = \arg\max_{0<T_1<T_2<L-1} \sigma_b^2(T_1, T_2)$.

Vícestupňové prahování

Původní obrázek

2 prahy

3 prahy

Adaptivní prahování

- Faktory jako je šum, nebo neuniformní osvětlení hrají velkou roli při hledání prahu
- Někdy může být rozmazání, nebo hledání hran nepraktické
- Variabilní (adaptivní) prahování – neexistuje jeden globální práh pro celý obraz:
  - Založené na lokálních vlastnostech
  - Založené na klouzavých průměrech

Adaptivní prahování

Původní obrázek

Obrázek se stínem

Globální Otsu

Adaptivní prahování

Lokální vlastnosti

- Hledáme prahovou hodnotu pro každý bod (x, y) dle nějaké vlastnosti okolí tohoto bodu
- Tento přístup je sice časově náročný, ale díky modernímu hardware možný
- Pro ilustraci použijeme průměr a standardní odchylku pixelů v okolí (popisují průměrnou hodnotu jasu a kontrast)
  m_{xy} průměr
  σ_{xy} standardní odchylka
  pixelů v okolí S_{xy}
- Práh: $T_{xy} = a\sigma_{xy} + bm_{xy}$ (a, b nezáporné konstanty),
  případně $T_{xy} = a\sigma_{xy} + bm_G$ (m_G je globální průměr)
- $g(x, y) = \begin{cases} 1 & \text{pro } f(x, y) > T_{xy} \\ 0 & \text{pro } f(x, y) \leq T_{xy} \end{cases}$

Adaptivní prahování

Lokální vlastnosti

- Obecně

$$g(x,y) = \begin{cases} 1 & \text{pro } Q(local) = TRUE \\ 0 & \text{pro } Q(local) = FALSE \end{cases}$$

  Q je predikát založený na lokálních parametrech

- Příklad výše

$$Q(\sigma_{xy}, m_{xy}) = \begin{cases} TRUE & \text{pro } f(x,y) > a\sigma_{xy} \text{ a } f(x,y) > bm_{xy} \\ FALSE & \text{jinak} \end{cases}$$

Adaptivní prahování

Původní obrázek

Prahové hodnoty

Adaptivní prahování

Adaptivní prahování

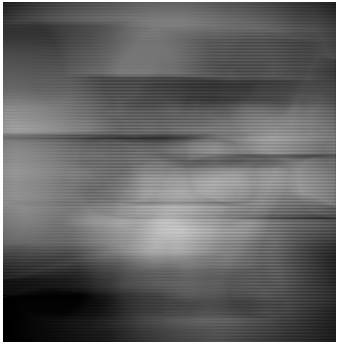
Klouzavé průměry

- Speciální případ prahování založeného na lokálních vlastnostech
- Počítáme klouzavé průměry podél skenovacích linií
- Je vhodné, pokud je důležitá rychlost
- Typicky obraz zpracováváme řádek po řádku v zik-zak pořadí
- z_{k+1} intenzita, kterou zpracováváme v kroku $k+1$
- klouzavý průměr v tomto bodě je
  $m(k+1) = \frac{1}{n} \sum_{i=k+2-n}^{k+1} z_i$
  $k \geq n-1$
- Nebo také
  $m(k+1) = m(k) + \frac{1}{n}(z_{k+1} - z_{k-n})$
  $k \geq n+1$
- n je počet bodů, které zahrnujeme do výpočtu, $m(1) = z_1$
- u krajů, kde nemáme n bodů, počítáme s body, které máme k dispozici
- $T_{xy} = cm_{xy}$, c je kladná konstanta, m_{xy} je klouzavý průměr v bodě (x, y)

Adaptivní prahování

$n = 500,\ c = 0.2$

Původní obrázek



Klouzavé průměry

Adaptivní prahování

Segmentace na regiony

- Segmentace = rozdělení na regiony, které splňují nějakou vlastnost
- Dělili jsme na základě nějaké prudké změny, nebo dle intenzity
- Dělení přímo na regiony:
  - Region growing
  - Region splitting

Segmentace na regiony

Region growing

- Spojujeme pixely do větších oblastí dle předepsaného kritéria růstu
- Začneme nějakým seedem, od kterého pokračujeme směrem dál a pixely, které splňují nějakou podmínku k tomuto regionu přidáme
- Od nově přidaných pixelů opět pokračujeme dál
- Výběr startovacích bodů (seedů) závisí na aplikaci
- Pokud nemáme žádnou informaci, kde je seed, hledáme pro každý pixel pixely splňující předepsanou podmínku
- Jestliže výsledky těchto výpočtů vytváří shluky, mohou být jako počáteční body použity ty pixely, které jsou blízko centroidu těchto shluků
- Jaké podmínky použijeme pro shlukování závisí nejen na aplikaci, ale také na vlastnostech obrázku (například na barvě)
- Samotný popis oblasti může vést k chybám (nespojitostem), pokud není brána v úvahu i sousednost pixelů
- Je také potřeba formulovat správně stop kritérium (kdy zastavit proces růstu)
- V některých aplikací můžeme také chtít brát v úvahu nejen lokální vlastnost, ale i „historii“ regionu, nebo jeho tvar

Segmentace na regiony
Region growing

- $f(x, y)$ vstupní obraz, $S(x, y)$ pole seedů (obrázek obsahující 1 v místech, seedů a 0 jinde), Q je predikát, který aplikujeme na každý pixel (x, y)
- Předpokládáme, že každý seed reprezentuje jednu oblast (region)
- Většinou bereme v úvahu 8-sousednost
- **Algoritmus**:
  1. Najdeme všechny souvislé komponenty v $S(x, y)$ a každou souvislou komponentu zmenšíme na jeden pixel; všechny takto nalezené pixely označíme hodnotou 1. Všechny ostatní pixely označíme 0
  2. Vytvoříme obrázek f_Q tak, že všem pixelům (x, y) nastavíme hodnotu 1, pokud splňují daný predikát. Jinak budou mít hodnotu 0.
  3. Výstupní obrázek g vytvoříme tak tak, že ke každému počátečnímu bodu (seedu) v S jsou připojeny všechny body s hodnotou 1 v f_Q, které jsou s tímto seedem 8-spojitě propojené.
  4. Každou spojitou komponentu v g označíme jinou hodnotou

Segmentace na regiony
Region growing

- Například víme, že region, který hledáme bude světlejší, než zbylá část obrázku.
- Prahováním obdržíme počáteční oblast, kterou následně pomocí matematické morfologie zmenšíme na 1 bod

Příklad

Jakou použijeme operaci z matematické morfologie?

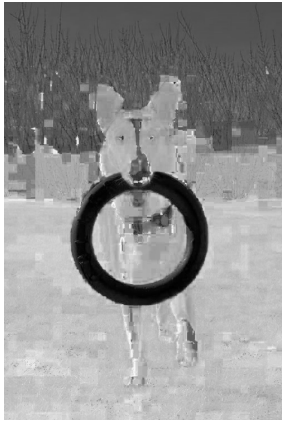
- Pak specifikujeme predikát např. sousední pixely jsou si podobné (použijeme absolutní rozdíl intenzity oproti seedu a tento rozdíl naprahujeme)

Segmentace na regiony

Region growing

- Predikát = pixely jsou v HSV podobně barevné jako seed (mapa vzdálenností barev od barvy v prostoru)

Původní obrázek



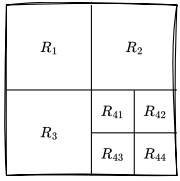
Vzdálenosti od seedu

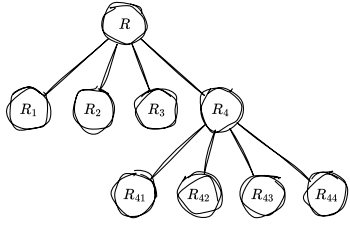
Region growing

Segmentace na regiony

Region splitting and merging

- Opačný přístup – začínáme celám obrazem jako regionem, pokud není region homogenní, pak ho dále dělíme (**splitting**) na disjunktní regiony
- R představuje celý obraz, Q daný predikát
- Region dělíme na menší a menší subregiony R_i, dokud neplatí $Q(R_i) = TRUE$
- Nejjednodušší dělení – na kvadranty
- toto dělení se dá vyjádřit pomocí tzv. **Quadtree**





Segmentace na regiony
Region splitting and merging

- Pokud bychom aplikovali jen dělení, vzniknou sousední regiony, které mají shodné vlastnosti
- Následuje tedy krok **merging**
- Spojujeme pouze sousední regiony, které splňují dohromady daný predikát $Q(R_i \cup R_j) = TRUE$
- Varianta 2: V kroku merging spojujeme sousední regiony, pokud oba dva samostatně splňují predikát (toto zrychlí proces)

Segmentace na regiony
Region splitting and merging

Původní obrázek

Region splitting and merging

Segmentace na regiony

Clusterování

- Základní myšlenka shlukování je rozdělit obraz na předem definovaný počet oblastí
- Příkladem může být **K-means** shlukování
- Každé pozorování je přidáno ke shluku s nejbližším průměrem (průměry nazýváme prototypy shluku)
- $\{z_1, z_2, \ldots z_Q\}$ je množina vzorků
- Vzorky jsou vektory a každá hodnota vektoru představuje nějaký numerický atribut pixelu
- Např. pokud pracujeme s šedotónovými obrázky, vektor obsahuje jen jednu hodnotu – intenzitu
- V RGB obrázcích představuje trojici hodnot

Segmentace na regiony
Clusterování

- Cílem shlukování metodou k-means je rozdělit množinu pozorování Q do k disjunktních množin shluků
  $C = \{C_1, C_2, \ldots, C_k\}$
  tak, aby bylo splněno následující kritérium optimality:
  $\arg\min_C \sum_{i=1}^{k} \sum_{z \in C_i} \|z - m_i\|^2$.
  m_i je střední vektor (centroid) vzorků v množině C_i
  $\|\cdot\|$ označuje normu vektoru (Obvykle se používá eukleidovská norma – prostorová vzdálenost)
- Nalezení minima je NP-těžký problém, používají se heuristiky

Segmentace na regiony

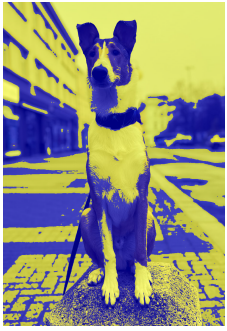
„Standardní“ algoritmus k-means

- Je dána množina pozorování $\{z_1, z_2, \ldots, z_Q\}$ a zadaná hodnota k
- **Algoritmus**:
  1. Zvolíme počáteční množinu průměrů $m_i^{(1)}$, $i = 1, 2, \ldots, k$ (mnohdy se bere náhodná množina)
  2. Každý vzorek přiřadíme do toho shluku, jehož průměr je mu nejbližší:
     $z_q \to C_i$ jestliže $\|z_q - m_i\|^2 < \|z_q - m_j\|^2$,
     $j = 1, 2, \ldots, k$, $(j \neq i)$, $q = 1, 2, \ldots, Q$
  3. Aktualizujeme centroidy shluků
     $m_i = \frac{1}{C_i} \sum_{z \in C_i} z, \qquad i = 1, 2, \ldots, k$
     C_i je počet vzorků ve shluku C_i
  4. Vypočteme eukleidovské normy rozdílů mezi vektory průměrů v aktuálním a předchozím kroku. Chyba E je součet těchto k norem.
  5. Algoritmus se ukončí, pokud $E \leq T$ (T je zadaný nezáporný práh)
     Jinak se vrátíme ke kroku 2.

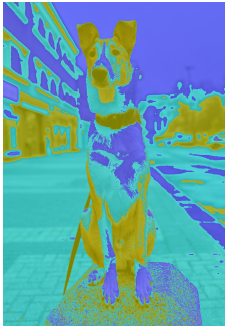
Segmentace na regiony

„Standardní“ algoritmus k-means

Původní obrázek



$k = 2$



$k = 3$

$k = 4$

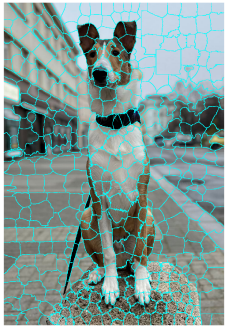
Výsledky lze vylepšit za použití například waveletů, kterým se budeme věnovat později.

Segmentace na regiony

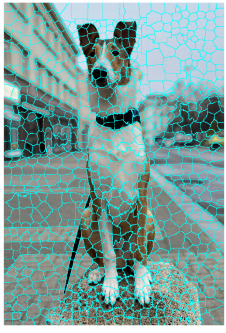
Superpixely

- Hlavní myšlenka: nahradit standardní mřížku pixelů seskupením pixelů do primitivních oblastí, které jsou smysluplnější(**superpixely**)
- Dochází ke snížení výpočetní náročnosti segmentace, tím že odstraníme nerelevantní detaily

$n = 100$



$n = 500$



$n = 1000$

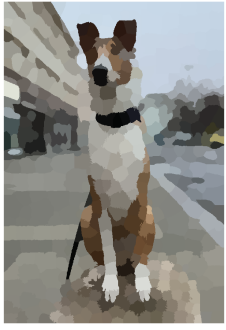
Velikost původního obrázku je 924×1351

Segmentace na regiony

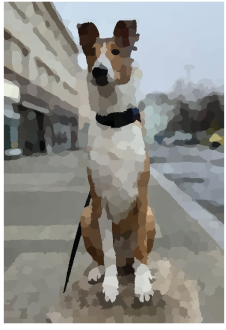
Superpixely

- Všem pixelům v jednom clusteru je přiřazená jedna hodnota

$n = 100$



$n = 500$



$n = 1000$

Segmentace na regiony
SLIC algoritmus

- Příkladem je algoritmus **SLIC** (simple linear iterative clustering)
- Vznikl modifikací k-means algoritmu
- Používáme 5 rozměrné vektory (3 barevné složky, 2 prostorové souřadnice)
  $z = [r, g, b, x, y]$
- n_{sp} požadovaný počet superpixelů, n_{tp} celkový počet pixelů v obrazu
- Počáteční centra superpixelů
  $m_i^T = [r_i,\ g_i,\ b_i,\ x_i,\ y_i], \qquad i = 1, 2, \ldots, n_{sp}$
  získáme vzorkováním obrazu na pravidelné mřížce (vzdálenost s)
- Aby byly superpixely přibližně stejně velké (tj. měly podobnou plochu), volí se rozestup mřížky $s = \sqrt{\frac{n_{tp}}{n_{sp}}}$
- Aby se zabránilo umístění centra superpixelu na hranu obrazu a zároveň se snížila pravděpodobnost, že inicializace začne v šumovém bodě, jsou počáteční centra shluků posunuta do bodu s nejmenším gradientem v okolí 3×3

Segmentace na regiony

SLIC algoritmus

- **Algoritmus**:
  1. Vypočítáme počáteční centra shluků superpixelů
     $m_i^T = [r_i, g_i, b_i, x_i, y_i], \qquad i = 1, 2, \ldots, n_{sp}$
     vzorkováním obrazu na pravidelné mřížce s krokem s
  2. Centra posuneme do bodu s nejmenším gradientem v okolí 3×3
  3. Každé poloze pixelu p v obraze nastavíme počáteční hodnoty $L(p) = -1$, $d(p) = \infty$
  4. Každému centru shluku m_i vypočítáme vzdálenost $D(p, i)$ mezi m_i a každým pixelem p v okolí $2s \times 2s$ kolem m_i
  5. Pokud $D(p, i) < d(p)$ pro pixel p nastavíme $d(p) = D(p, i)$ a $L(p) = i$
  6. C_i označuje množinu pixelů v obraze s označením $L(p) = i$
     Aktualizujeme centrum shluku
     $m_i = \frac{1}{|C_i|} \sum_{z \in C_i} z$
     $|C_i|$ je počet pixelů v množině C_i
  7. Vypočítáme eukleidovské normy rozdílů mezi vektory průměrů v aktuálním a předchozím kroku
     Reziduální chybu E určíme jako součet těchto (n_{sp}) norem
     Pokud $E < T$ (T zadaný nezáporný práh), pokračujeme následujícím krokem
     V opačném případě se vrátíme ke kroku 4
  8. Nahradíme všechny superpixely v každé oblasti C_i jejich průměrnou hodnotou m_i

Segmentace na regiony
SLIC algoritmus

- Výstupem je obraz, kde všechny pixel superpixely mají stejnou (průměrnou) hodnotu (krok 8)

Příklad

Jak bychom spočítali průměrnou barvu pixelů?

- Narozdíl od k-means nepočítáme všechny vzdálenosti, ale jen vzdálenosti v $2s \times 2s$ okolí

Segmentace na regiony
SLIC algoritmus

- Definice **vzdálenosti**:
  - pixely jsou dány barevnými složkami a prostorovými souřadnicemi (měřítka spolu nesouvisejí)
  - barevné a souřadnicové vzdálenosti zpracováváme odděleně
  - barevná vzdálenost $d_c = \left[(r_j - r_i)^2 + (g_j - g_i)^2 + (b_j - b_i)^2\right]^{1/2}$
  - prostorová vzdálenost $d_s = \left[(x_j - x_i)^2 + (y_j - y_i)^2\right]^{1/2}$
  - složená vzdálenost $D = \left[\left(\frac{d_c}{d_{cm}}\right)^2 + \left(\frac{d_s}{d_{sm}}\right)^2\right]^{1/2}$

    d_{cm}, d_{sm} maximální očekávané hodnoty d_c a d_s
  - maximální prostorová vzdálenost by měla odpovídat vzorkovacímu intervalu

    $d_{sm} = s = \sqrt{\frac{n_{tp}}{n_{sp}}}$
  - maximální barevná hodnota závisí na obraze, nastavujeme konstantní hodnotu c (relativní důležitost podobnosti barev)
  - po úpravě dostáváme

    $D = \left[d_c^2 + \left(\frac{c}{s}\right)^2 d_s^2\right]^{1/2}$

Segmentace na regiony

Superpixely

- K-means algorittmus spuštěný na obrázky získané superpixely

$n = 100$

$n = 500$

$n = 1000$

Segmentace na regiony

Grafové řezy

- Obraz dělíme na menší části, pixely v tomto přístupu chápeme jako uzly grafu
- Hledáme rozdělení grafu (graph cuts) na skupiny uzlů
- Řezy volíme dle nějakých kritérií (uzly v rámci skupiny mají vysoké hodnoty kritérií, mimo skupinu nízké)
- Pro některé případy dostáváme lepší výsledky, než metodami dříve
- Mají ale vyšší časovou náročnost

- Graf je struktura tvořená množinou uzlů V a množinou hran E ($E \subseteq V \times V$)
  $G = (V, E)$
- Grafy:
  - neorientované – pokud z $(u, v) \in E$ plyne, že $(v, u) \in E$
  - orientované – jinak

Segmentace na regiony
Grafové řezy

- **Obraz jako graf**
- Jednotlivé pixely budeme brát jako uzly, graf bude úplný a neorientovaný
- Každá hrana má definovanou váhu (graf je ohodnocený)
- Váhy budeme reprezentovat maticí W (element $w(i,j)$ říká, jaká je váha hrany spojující uzly i a j
- Váhy spočítáme jako podobnosti dvojic pixelů

Příklad

Co znamená, že je graf úplný?
Co můžeme říci o matici W?

Segmentace na regiony

Grafové řezy

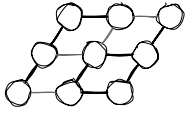
- Hledáme dělení grafu na disjunktní množiny $V_1, V_2, \ldots V_k$
- Dělení je takové, že podobnost uzlů v jednotlivých množinách je vysoká, ale podobnost uzlů mezi množinami je nízká
- Jednotlivé množiny pak bereme jako segmentované regiony
- **Řez grafu** = dělení množiny vrcholů na dvě disjunktní množiny (dělení grafu je modelování odstraněním hran spojující vrcholy mezi dvěma podmnožinami)

Segmentace na regiony

Grafové řezy

- Princip si ukážeme na grafu představující obrázek 3×3, pixely tvoří uzly grafu, hrany jsou pouze mezi sousedními pixely (4-sousednost)
  Je to jen příklad, obecně jsou hrany mezi všemi pixely!

Obrázek



Graf

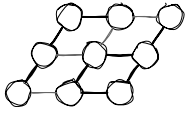
- Váhy hran se běžně počítají jako prostorová vzdálenost pixelů a podobnost ve smyslu intenzity, barvy nebo textury
- V našem příkladu váhu počítáme jako inverzi rozdílu intenzit
  $w(i,j) = 1/(|I(n_j) - I(n_i) + c|)$
  $I(n_i)$ a $I(n_j)$ jsou intenzity pixelů, c konstanta, abychom předešli dělení 0

Segmentace na regiony

Grafové řezy

- Princip si ukážeme na grafu představující obrázek 3×3, pixely tvoří uzly grafu, hrany jsou pouze mezi sousedními pixely (4-sousednost)
  Je to jen příklad, obecně jsou hrany mezi všemi pixely!

Obrázek



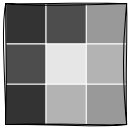
Graf

- Váhy hran se běžně počítají jako prostorová vzdálenost pixelů a podobnost ve smyslu intenzity, barvy nebo textury
- V našem příkladu váhu počítáme jako inverzi rozdílu intenzit
  $w(i,j) = 1/(|I(n_j) - I(n_i) + c|)$
  $I(n_i)$ a $I(n_j)$ jsou intenzity pixelů, c konstanta, abychom předešli dělení 0
  V obrázku odpovídá tloušťce čáry

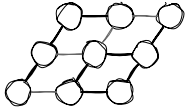
Segmentace na regiony

Grafové řezy

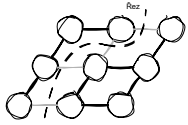
- Hrany mezi tmavými pixely, stejně jako hrany mezi světlými pixely jsou silnější, než mezi tmavými a světlými uzly
- Řez bychom provedli podél slabých hran



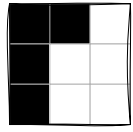
Obrázek



Graf



Řez

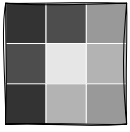


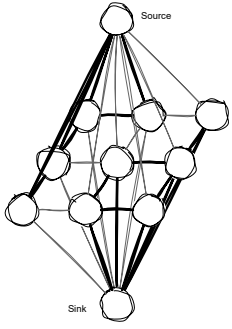
Segmentace

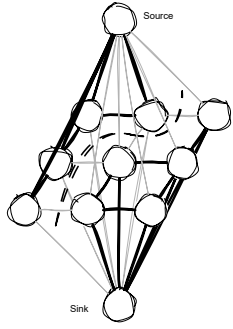
Segmentace na regiony

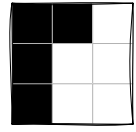
Grafové řezy

- Setkáváme se i s jinou definicí grafů, které kromě „pixelových“ uzlů mají ještě uzly terminální (**source** = zdroj a **sink** = stok)
- Oba uzly jsou spojeny se všemi „pixelovými“ uzly pomocí hran, kterým říkáme **t-link**
- Role těchto uzlů je, že určují pravděpodobnost, s jakou uzel patří do popředí, nebo do pozadí (síla t-link)









Segmentace na regiony

Minimum graph cuts

- Jakmile vyjádříme obrázek jako graf, chceme ho rozdělit na dva nebo více podgrafů
- Pixely v jednotlivých podgrafech tvoří regiony
- Ohodnocený graf můžeme interpretovat jako tok v síti a využít **minimum cut graph**
- Ten je založen an **Max-flow, min-cut theorému**
- Ten říká, že v síti je maximální množství toku, které může protékat ze zdroje do stoku, rovno hodnotě minimálního řezu
- Tento minimální řez je definován jako nejmenší celková váha hran, jejichž odstraněním by došlo k odpojení stoku od zdroje
  $\text{cut}(A, B) = \sum_{u \in A,\, v \in B} w(u, v)$.
  A, B jsou disjunktní množiny
- Optimální řez je takový, který minimalizuje hodnotu tohoto řezu

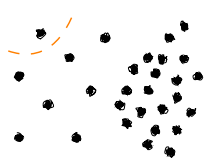
Příklad

Kolik je přibližně množství potencionálních řezů?

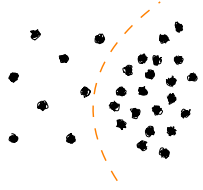
Segmentace na regiony

Minimum graph cuts

- Počet řezů je exponenciální číslo, problém vybrat ten nejmenší je tedy výpočetně náročný
- Existují efektivní algoritmy, které řeší Max-flow problém
- Díky Max-flow, min-cut theorému můžeme tyto algoritmy využít k segmentaci
- Mnohdy tyto algoritmy vedou k tomu, že jeden region je příliš malý, což v není v segmentaci žádoucí
- Viz následující uskupení, které by bylo propojeno hranami s váhou nepřímo úměrnou vzdálenosti uzlů



Min-cut



Vhodnější řez

Segmentace na regiony

Minimum graph cuts

- Abychom se tomuto vyhnuli, upravíme definici toho, co chápeme jako řez
- Místo celkové váhy hran, které spojují dvě části grafu,pracujeme s mírou „oddělenosti“
- Ta počítá cenu řezu jako zlomek z celkových hranových spojení ke všem uzlům v grafu
- Tato míra, nazývaná **normalizovaný řez** (Ncut), je definována jako
  $\mathrm{Ncut}(A,B) = \frac{\mathrm{cut}(A,B)}{\mathrm{assoc}(A,V)} + \frac{\mathrm{cut}(A,B)}{\mathrm{assoc}(B,V)}$,
- Součet vah všech hran vedoucích z uzlů podgrafu A ke všem uzlům celého grafu
  $\mathrm{assoc}(A,V) = \sum_{u\in A,\, z\in V} w(u,z)$
  Stejně definujeme i $\mathrm{assoc}(B,V)$
- Použitím Ncut vyřešíme dělení na izolovaný bod a zbytek
- Můžeme definovat míru celkové normalizované asociace uvnitř rozdělení grafu
  $\mathrm{Nassoc}(A,B) = \frac{\mathrm{assoc}(A,A)}{\mathrm{assoc}(A,V)} + \frac{\mathrm{assoc}(B,B)}{\mathrm{assoc}(B,V)}$,
  $\mathrm{assoc}(A,A)$ $(\mathrm{assoc}(B,B))$ = celkové váhy hran spojujících uzly uvnitř množiny A (B)

Segmentace na regiony

Minimum graph cuts

- Zřejmě $\mathrm{Ncut}(A, B) = 2 - \mathrm{Nassoc}(A, B)$
- Minimalizace $\mathrm{Ncut}(A, B)$ současně maximalizuje $\mathrm{Nassoc}(A, B)$
- Segmentace obrazu pomocí grafových řezů je založena na nalezení takového rozdělení, které minimalizuje $\mathrm{Ncut}(A, B)$
- Přesná minimalizace této veličiny je NP-úplná úloha
- Existuje aproximační řešení minimalizace $\mathrm{Ncut}(A, B)$
- Je založená na definici minimalizační úlohy jako problému zobecněných vlastních čísel, pro který existuje řada implementací

Segmentace na regiony

Minimum graph cuts

- V označuje množinu uzlů grafu G, A a B tvoří rozklad V, K počet uzlů ve V
- Definujme K-rozměrný vektor x, jehož prvek x_i:

$$x_i = \begin{cases} 1, & \text{je-li uzel } n_i \in A, \\ -1, & \text{je-li uzel } n_i \in B. \end{cases}$$

- $d_i = \sum_j w(i,j)$ je součet vah všech hran vedoucích z uzlu n_i ke všem ostatním uzlům

$$\mathrm{Ncut}(A,B) = \frac{\mathrm{cut}(A,B)}{\mathrm{cut}(A,V)} + \frac{\mathrm{cut}(A,B)}{\mathrm{cut}(B,V)} = \frac{\sum_{x_i>0, x_j<0} -w(i,j)x_i x_j}{\sum_{x_i>0} d_i} + \frac{\sum_{x_i<0, x_j>0} -w(i,j)x_i x_j}{\sum_{x_i<0} d_i}$$

- Chceme najít vektor x, který minimalizuje $\mathrm{Ncut}(A,B)$
- Takový vektor existuje a snadné ho najít, pokud bychom se neomezili na to, že jeho hodnoty jsou ± 1

  $(D - W)y = \lambda D y$

  D je diagonální matice $K \times K$ s diagonálními prvky d_i

  W je váhová matice $K \times K$ s prvky $w(i,j)$
- Řešením rovnice dostaneme K vlastních čísel a K vlastních vektorů
- Řešením našeho problému je vlastní vektor odpovídající druhému nejmenšímu vlastnímu číslu

Segmentace na regiony

Minimum graph cuts

- Požadovaný diskrétní vektor x lze získat ze spojitého řešení nalezením dělícího bodu, který rozdělí hodnoty prvků vlastního vektoru na dvě části
- Tento bod zvolíme tak, aby výsledná hodnota $\mathrm{Ncut}(A, B)$ byla minimální
- Všechny prvky vlastního vektoru s hodnotou větší než tento bod $\rightarrow 1$ (tyto body tvoří A) ostatní $\rightarrow -1$ (tyto body tvoří B)
- Většinou existuje několik dělících bodů, které mají podobnou hodnotu
- Některé z nich nevedou k dobrému výsledku
- Přidáváme kritérium stability a řez zvolíme, jen když je splněné
- To se získá tak, že se nejprve spočítá histogram hodnot vlastního vektoru a poté se určí poměr minimálního a maximálního počtu hodnot v histogramových třídách
- U „nejistého“ vlastního vektoru zůstávají hodnoty histogramu přibližně stejné a tento poměr je relativně vysoký
- V takovém případě dále nedělíme

Segmentace na regiony

Minimum graph cuts

- **Váhy hran grafu**:
  - Váhy na základě podobnosti intenzit
  - Váhy založené na prostorové vzdálenosti
  - Barvy, textury
  - Statistické momenty oblasti Těm se budeme věnovat později.
- Lepší je kombinace výše zmíněného, např. kombinace intenzity a vzdálenosti
- Hodnota váhy hrany mezi dvěma pixely by měla být velká, pokud jsou pixely velmi blízké jak z hlediska intenzity, tak i z hlediska polohy a měla by se zmenšovat s rostoucím rozdílem intenzit a s rostoucí vzdáleností mezi pixely

$$w(i,j) = \begin{cases} e^{-\frac{(I(n_i)-I(n_j))^2}{\sigma_I^2}} e^{-\frac{\mathrm{dist}(n_i,n_j)^2}{\sigma_d^2}}, & \text{pokud } \mathrm{dist}(n_i,n_j) < r, \\ 0, & \text{jinak.} \end{cases}$$

σ_I^2 a σ_d^2 jsou konstanty
$\mathrm{dist}(n_i, n_j)$ je například eukleidovská vzdálenost mezi dvěma uzly
r je konstanta určující, jak daleko ještě uvažujeme podobnost

Segmentace na regiony

Minimum graph cuts

1. Je dána množina příznaků. Sestavíme vážený graf
   $G = (V, E)$
   Vypočítáme váhy hran a použijme je k vytvoření matic V a D
   K požadovaný počet rozdělení grafu
2. Vyřešíme
   $(D - W)y = \lambda Dy$
   a určíme vlastní vektor odpovídající druhému nejmenšímu vlastnímu číslu
3. Pomocí vlastního vektoru z kroku 2 rozdělíme graf na dvě části nalezením dělícího bodu tak, aby byla minimalizována hodnota $\mathrm{Ncut}(A, B)$
4. Pokud počet provedených řezů ještě nedosáhl hodnoty K, rozhodneme, zda má být aktuální rozdělení dále děleno, kontrolou stability řezu
5. Je-li to nutné, rekurzivně rozdělíme již segmentované části

Segmentace na regiony

Minimum graph cuts

- Grafové řezy jsou ideálně vhodné pro získání hrubé segmentace hlavních oblastí v obraze
- Například tak, že nejprve obrázek rozostříme (klasickým průměrováním) a pak použijeme graph cut
- Níže je příklad použití Matlab Segmenteru
- Označíme body patřící popředí, body patřící pozadí a pak se počítá Graph cut