Extrakce příznaků I

Pokročilá analýza obrazu

Mgr. Markéta Trnečková, Ph.D.

Extrakce příznaků

- Segmentace = dělení obrazu na nějaké logické části
- Dalším krokem je popsání segmentovaných objektů kvůli dalšímu strojovému zpracování
- **Extrakce příznaků** = detekce příznaků a jejich popis
- Detekovaným příznakům přiřazujeme kvantitativní atributy
- Dělení podle toho, na jaké části je aplikujeme
  - hranice oblastí
  - oblasti
  - celé obrazy
- **Deskriptory příznaků** = způsob, jakým popisujeme jednotlivé příznaky
- Měly by být co nejméně citlivé na změny parametrů, jako jsou měřítko, translace, osvětlení, úhel, ...

Co je příznak obrazu

- Příznak intuitivně chápeme jako charakteristický popis „něčeho“
- „Něco“ může být objekt, obraz nebo množina obrazů
- Nejprve musíme příznaky detekovat a pak je popsat
- Např. příznakem můžou být rohy objektu
  - Detekce = nalezení rohů v oblasti
  - Popis = přiřazení atributů těmto příznakům, např. jejich poloha vzhledem k ostatním rohům
- Díky detekovaným příznakům a jejich vlastnostem můžeme jednotlivé oblasti od sebe odlišovat

Co je příznak obrazu

- Jaké vlastnosti by měly příznaky mít v kontextu digitálního zpracování obrazu, abychom je mohli používat k rozlišování objektů?
- Obecné vlastnosti:
  - Měly by být nezávislé na poloze, rotaci či měřítku (otočený čtverec, je stále čtverec)
  - Měly by být nezávislé i na osvětlení, či perspektivě
- Většinou se před extrakcí příznaků obraz normalizuje

Příklad

Jak bychom obraz upravili, abychom minimalizovali vliv výrazných změn osvětlení, které komplikují detekci příznaků?

Co je příznak obrazu

Příklad

Jak bychom obraz upravili, abychom minimalizovali vliv výrazných změn osvětlení, které komplikují detekci příznaků?

- Například vyrovnání nebo specifikace histogramu
- Nezávislost:
  - **invariantnost** = pokud se hodnota deskriptoru po aplikaci transformací nemění
  - **kovariantnost** = pokud se jeho hodnota mění stejným způsobem jako transformovaný objekt

Příklad

Je plocha epileptické oblasti invariantní či kovariantní vůči translaci, rotaci a změně měřítka?

Co je příznak obrazu

- V praxi se kovariantní deskriptory převádí na invariantní pomocí vhodné normalizace
- Další dělení:
  - **lokální příznaky** = vztahují se k prvku množiny
  - **globální příznaky** = popisují celou množinu
- Deskriptory většinou přímo nepoužívá člověk, ale slouží jako vstupy do složitějších úloh
- Například registrace, rozpoznávání objektů, hledání vzorů a pod.
- Deskriptory organizujeme do tzv. **příznakových vektorů** (feature vector) (matice $1 \times n$ nebo $n \times 1$), složky jsou jednotlivé deskriptory
- Jednoduchým příkladem je RGB obraz (každý pixel je reprezentován 3 hodnotami)
- Pokud použijeme barvu jako příznak, oblast v RGB obraze lze reprezentovat jako množinu bodů v trojrozměrném prostoru
- Při použití n deskriptorů vzniká n-rozměrný příznakový prostor
- Množina příznakových vektorů = „mračno“ bodů v n-rozměrném eukleidovském prostoru

Co je příznak obrazu

- Dělení podle toho, co popisují:
  - příznaky hranic
  - příznaky oblastí
  - příznaky celých obrazů
- některé příznaky se dají použít v různých kontextech

Předzpracování hranic

- Segmentace vrací pixely, které leží na hranicích nebo které tvoří nějakou oblast
- Různé přístupy k předzpracování hranic
  - Sledování hranic (trasování)
  - Aproximace hranic
  - Podpisy
  - a jiné

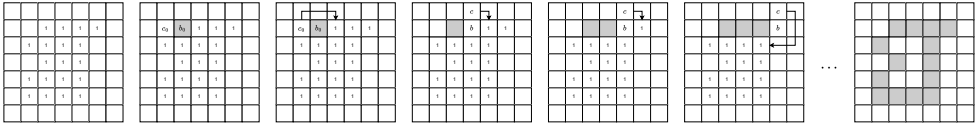
Sledování hranice (trasování)

- Některé deskriptory vyžadují sekvenci bodů, které tvoří hranici
- Někdy chceme, aby body popisující hranici oblasti byly uspořádány ve směru hodinových ručiček (někdy proti)
- Představíme algoritmus, který vytvoří uspořádanou posloupnost bodů, které tvoří hranici
- Předpoklad:
  - Pracujeme s binárními obrazy, kde body objektu mají hodnotu 1, pozadí 0
  - Okraje obrazu jsou 0 (aby objekt nesplýval s okrajem)
- Pokud by obraz obsahoval více oblastí, každou zpracováváme zvlášť

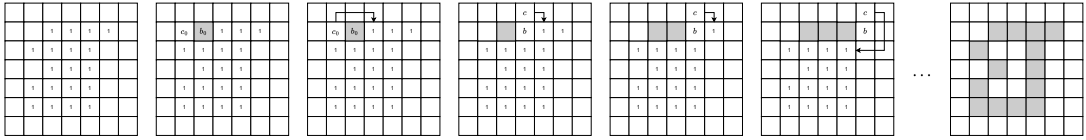
Sledování hranice (trasování)

Algoritmus

1. Počáteční bod b_0 je nejvýše a nejvíce vlevo ležící bod v obraze s hodnotou 1
   Označme c_0 jako jeho západního souseda (tento bod je vždy bodem pozadí)
   Zkoumáme 8-sousedství bodu b_0, počínaje bodem c_0 a postupujeme ve směru hodinových ručiček
   Označme b_1 první sousední bod s hodnotou 1 a c_1 bod bezprostředně předcházející bodu b_1
   Uložíme si polohu bodu b_0 pro použití v kroku 5
2. Položíme $b = b_0$ a $c = c_0$
3. Označíme 8-sousedství bodu b, počínaje bodem c a postupně ve směru hodinových ručiček, jako $n_1, n_2, \ldots, n_8$
   Najdeme první sousední bod s hodnotou 1 a označím ho n_k
4. Nastavíme $b = n_k$ a $c = n_{k-1}$
5. Opakujeme kroky 3 a 4, dokud neplatí $b = b_0$
   Posloupnost bodů b, nalezená po ukončení algoritmu, tvoří uspořádanou množinu bodů hranice



Sledování hranice (trasování)



- Bod c v kroku 4 je vždy bodem pozadí, protože n_k je první nalezený bod s hodnotou 1 při průchodu ve směru hodinových ručiček
- Tento algoritmus je znám jako **Mooreův algoritmus sledování hranice**
- Algoritmus pracuje jak pro hranice, ale i pro celé oblasti
- Vrácená hranice je většinou jednopixelová

Příklad

Jak bychom získali hranici díry uvnitř oblasti?

Příklad

Dá se tento algoritmus formulovat i pro hranici tvořenou body proti směru hodinových ručiček?

Sledování hranice (trasování)

Příklad

Jak bychom získali hranici díry uvnitř oblasti?

- Extrahujeme díry, nastavíme jim hodnotu 1 a najdeme pro ně hranici

Příklad

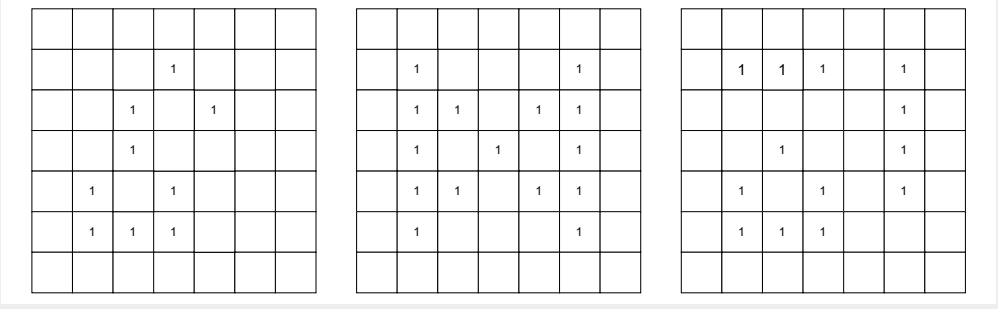
Dá se tento algoritmus formulovat i pro hranici tvořenou body proti směru hodinových ručiček?

- Ano
- Jednodušší je ale použít tuto variantu a obrátit pořadí posloupnosti bodů

Sledování hranice (trasování)

Příklad

Vyzkoušejte si sami vytvořit hranici následujících objektů.



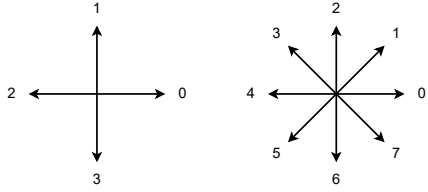
Řetězové kódy

- Reprezentace hranice pomocí spojité posloupnosti úseček
- Úsečky mají zadanou délku a směr
- Předpokládáme, že hranice jsou uzavřené a jednoduché (neprotínají se)
- Různé přístupy:
  - Freemanovy kódy (Freeman chain codes)
  - Sklonové řetězové kódy (Slope chain codes)

Řetězové kódy

Freemanovy řetězové kódy

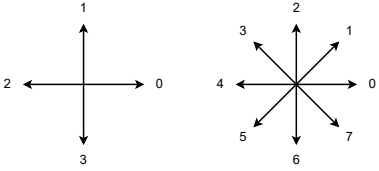
- Obvykle založeno na 4- nebo 8-sousednosti segmentů
- Směr každého segmentu je kódován číselně



- Posloupnost těchto kódů tvoří **Freemanův kód**
- Digitální obrazy jsou obvykle vzorkovány v pravidelné mřížce se stejným rozestupem v osách x a y
- Kód tvoříme sledováním hranice (např. ve směru hodinových ručiček) a přiřazení směru segmentům spojujícím sousední pixely

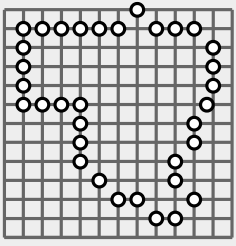
Řetězové kódy

Freemanovy řetězové kódy



Příklad

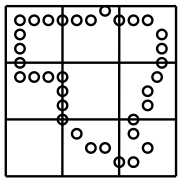
Jak by vypadal Freemanův kód pro následující obrázek?

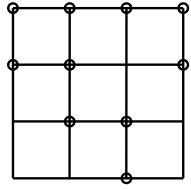


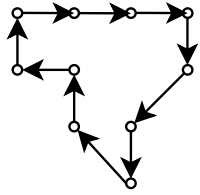
Řetězové kódy

Freemanovy řetězové kódy

- Běžně nepoužíváme takto detailní reprezentaci
  - výsledný kód je dlouhý
  - drobné nepravidelnosti způsobené šumem, nebo nepřesnou segmentací vedou ke změnám kódu nesouvisející s tvarem objektu
- Převzorkujeme hranici pomocí hrubší mřížky (její velikost závisí na konkrétní aplikaci







Příklad

Jak by vypadal Freemanův kód 3. obrázku?

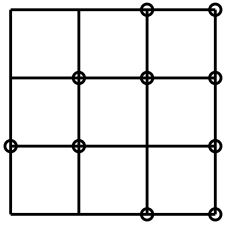
Řetězové kódy

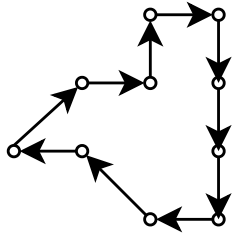
Freemanovy řetězové kódy

- Dosud jsme začínali v levém horním rohu
- Konkrétní kód závisí na volbě počátečního bodu
- To odstraníme normalizací, kdy kód považujeme za cyklickou posloupnost a zvolíme takový počátek, aby výsledné číslo mělo nejmenší hodnotu
- Normalizaci můžeme provést i ve smyslu rotace (úhly posouvají čísla kódující směry)

Příklad

Jak by vypadal Freemanův kód následujícího obrázku, který vznikl rotací obrázku z předchozího příkladu o 90 stupňů?



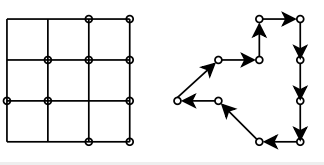


Řetězové kódy

Freemanovy řetězové kódy

Příklad

Jak by vypadal Freemanův kód následujícího obrázku, který vznikl rotací obrázku z předchozího příkladu o 90 stupňů?



- Normalizaci můžeme provést i ve smyslu rotace (úhly posouvají čísla kódující směry)
- Místo kódu směrů budeme brát derivaci (1 rozdíl)
- Rozdíl = počet změn směru

Příklad

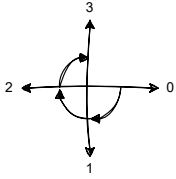
Předpokládejme, že máme kód **1 0 1 0 3 3 2 2** (u 4-sousednosti). Jak by vypadal kód rozdílů?

Řetězové kódy
Freemanovy řetězové kódy

Příklad

Předpokládejme, že máme kód **1 0 1 0 3 3 2 2** (u 4-sousednosti). Jak by vypadal kód rozdílů?

- 3 1 3 3 0 3 0



- Normalizace ve smyslu změny měřítka je realizována změnou vzorkovací mřížky

Řetězové kódy

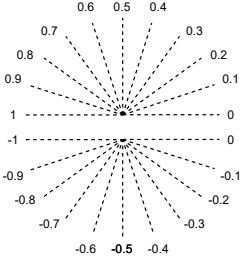
Sklonové řetězové kódy

- U Freemanova kódu potřebujeme převzorkovat obrázky, abychom odstranili drobné variace
- Převzorkování = namapování bodů původní hranice na nejbližší body ové mřížky
- Alternativou jsou **sklonové řetězové kódy**
- Bereme linie stejné délky a omotáváme je okolo hranice tak, aby se koncové body linií dotýkaly křivky
- Kód definujeme jako změny směru jednotlivých linií
- Tuto změnu normalizujeme na spojitý interval $< -1, 1 >$
- Potřebujeme definovat délku segmentu
- Sklonové kódy jsou nezávislé na rotaci o libovolný úhel (u Freemanových kódů to bylo jen o úhly 45, respektive 90 stupňů)
- Je nezávislý na pozici (translaci)
- Změnou délky segmentu můžeme zajistit nezávislost na změně měřítka

Řetězové kódy

Sklonové řetězové kódy

- U Freemanova kódu potřebujeme převzorkovat obrázky, abychom odstranili drobné variace
- Převzorkování = namapování bodů původní hranice na nejbližší body ové mřížky
- Alternativou jsou **sklonové řetězové kódy**
- Bereme linie stejné délky a omotáváme je okolo hranice tak, aby se koncové body linií dotýkaly křivky
- Kód definujeme jako změny směru jednotlivých linií
- Tuto změnu normalizujeme na spojitý interval $< -1, 1 >$



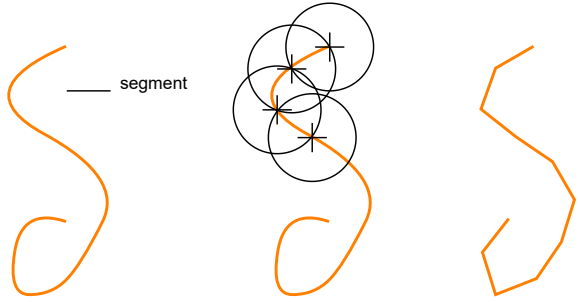
Řetězové kódy
Sklonové řetězové kódy

- Potřebujeme definovat délku segmentu
- Sklonové kódy jsou nezávislé na rotaci o libovolný úhel (u Freemanových kódů to bylo jen o úhly 45, respektive 90 stupňů)
- Je nezávislý na pozici (translaci)
- Změnou délky segmentu můžeme zajistit nezávislost na změně měřítka

Řetězové kódy

Sklonové řetězové kódy

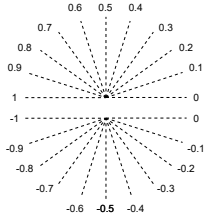
- **Postup**
  - Máme křivku (hranici), zvolíme délku segmentu
  - Specifikujeme počáteční bod a do něj umístíme jeden konec segmentu
  - Najdeme druhý konec tak, aby se dotýkal křivky a ten se stane počátkem dalšího segmentu
  - Jakmile máme segment (oba konce) najdeme úhel
    Pozitivní změna je mapována na $< 0, 1)$
    Negativní na $(-1, 0 >$
    -1 a 1 určují stejný směr jako 0 proto je nepoužíváme



Řetězové kódy

Sklonové řetězové kódy

- Délka kódu je dána přesností
- Pokud máme přesnost 0.1, tak máme 19 různých symbolů



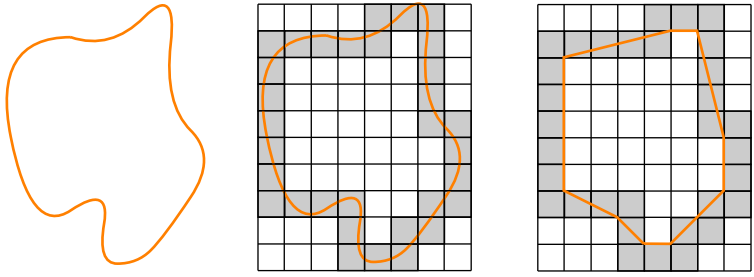
- Koncový bod křivky, nemusí být koncovým bodem posledního segmentu, ten proto zkracujeme

Příklad

Máme křivku, na kterou aplikujeme operaci zrcadlení. Jak se změní kód?

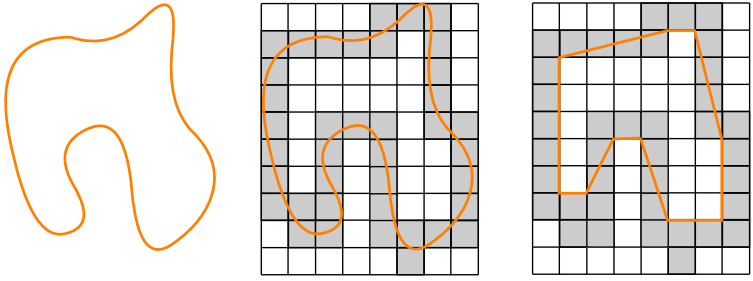
Aproximace okraje

- Každý okraj je možné s jistou mírou přesnosti aproximovat polygonem
- V případě, že počet segmentů odpovídá počtu bodů hranice jedná se o přesný popis
- Cílem aproximace je najít polygon s rozumně malým množstvím segmentů, který zachycuje podstatu tvaru
- Tento úkol není triviální a může vést k časově náročnému iterativnímu procesu
- Jedním z přístupů je využití principu **minimum perimeter polygon** (MPP)
- Okraj překryjeme sousedícími buňkami
- Tento okraj zmenšíme tak, aby byl uzavřen v buňkách, ale procházel vrcholy těchto buněk



Aproximace okraje

- Velikost buněk ovlivňuje přesnost aproximace

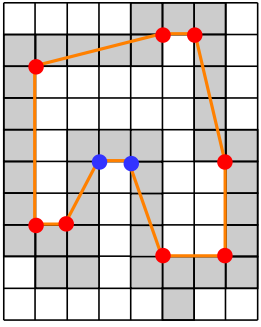


- Když se podíváme na objekt, který dostaneme, každý vrchol polygonu představuje konvexní/konkávní úsek

Příklad

Na příkladu výše určete, které vrcholy patří ke konkávním respektive konvexnímu úseku.

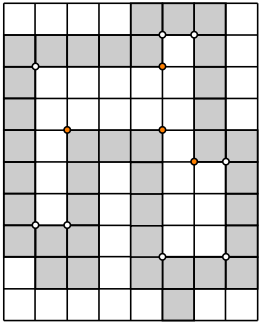
Aproximace okraje



- konvexní, konkávní
- Podívejte se na polohu konvexních a konkávních bodů v rámci buněk

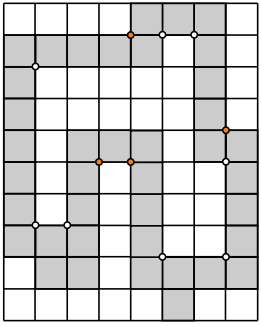
Aproximace okraje

- Množině buněk říkáme **cellular complex**
- Vezmeme vrcholy uvnitř buněk, určíme, zda patří konvexnímu, respektive konkávnímu úseku (konkávní oranžové)



Aproximace okraje

- Polygon tvoří konvexní vrcholy (označme je W) a u konkávních vrcholů vezmeme jejich protilehlé vrcholy (označme je B) viz níže



- Ne všechny vrcholy však tvoří MPP

Příklad

Můžeme určit, zda bude nejhornější levý vrchol vždy konvexní nebo konkávní? Proč?

Aproximace okraje

- Nejhornější levý vrchol bude vždy konvexní

Příklad

Jak určíme, zda je vrchol konvexní nebo konkávní?

Aproximace okraje

Příklad

Jak určíme, zda je vrchol konvexní nebo konkávní?

- Vezmeme tři vrcholy a, b a c

$$A = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix}$$

- Můžeme určit z determinantu
  $det(A) > 0$ sekvence vrcholů je proti směru hodinových ručiček (pozitivní strana)
  $det(A) = 0$ kolineární vrcholy
  $det(A) < 0$ sekvence po směru hodinových ručiček
- Z toho, zda jsou po nebo proti směru hodinových ručiček dokážeme určit, zda je b konvexní, nebo konkávní

Příklad

Jak?

Aproximace okraje

Příklad

Jak určíme, zda je vrchol konvexní nebo konkávní?

- Vezmeme tři vrcholy a, b a c

$$A = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix}$$

- Můžeme určit z determinantu
  $det(A) > 0$ sekvence vrcholů je proti směru hodinových ručiček (b leží na kladné straně přímky ac)
  $det(A) = 0$ kolineární vrcholy
  $det(A) < 0$ sekvence po směru hodinových ručiček (b leží na záporné straně přímky ac)
- Z toho, zda jsou po nebo proti směru hodinových ručiček dokážeme určit, zda je b konvexní, nebo konkávní

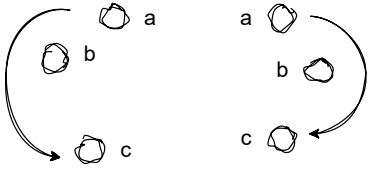
Příklad

Jak?

Aproximace okraje

Příklad

Jak?



Aproximace okraje
MPP algoritmus

- Vytvoříme seznam trojic obsahujících
  - označení vrcholu (např. $V_0, V_1, \dots$)
  - souřadnice vrcholu
  - prvek určující, zda je vrchol typu W nebo B
- Konkávní vrcholy jsou zrcadleny, vrcholy jsou sekvenčně uspořádány (proti směru hodinových ručiček) a první prvek je nejhornější levý prvek (víme, že je W)
- Tento vrchol označíme V_0
- Algoritmus používá dva „prohledávací“ body (crawlery)
  - Bílý crawler C_W – pohybuje se po konvexních vrcholech W
  - Modrý crawler C_B – pohybuje se po konkávních vrcholech B
- Tyto dva body, poslední nalezený vrchol MPP a aktuálně zkoumaný vrchol jsou vše, co je potřeba k implementaci algoritmu

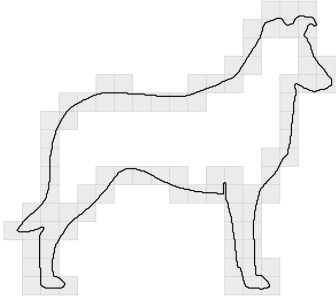
Aproximace okraje

MPP algoritmus

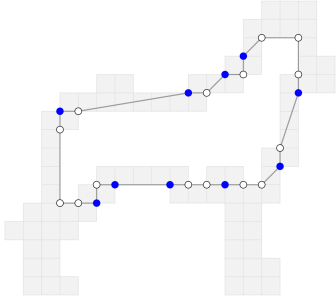
- Nastavíme $C_W = C_B = V_0$
- Označíme V_L poslední nalezený vrchol MPP a V_k aktuálně zkoumaný vrchol
- Mezi V_L, V_k a oběma crawlery mohou nastat tři situace:
  (a) V_k leží na kladné straně přímky určené dvojicí bodů (V_L, C_W), tj. $\text{sgn}(V_L, C_W, V_k) > 0$
  (b) V_k leží na záporné straně přímky (V_L, C_W) nebo je s ní kolineární, tj. $\text{sgn}(V_L, C_W, V_k) \leq 0$, a současně leží na kladné straně přímky (V_L, C_B) nebo je s ní kolineární, tj. $\text{sgn}(V_L, C_B, V_k) \geq 0$
  (c) V_k leží na záporné straně přímky (V_L, C_B), tj. $\text{sgn}(V_L, C_B, V_k) < 0$
- V případě (a) dalším vrcholem MPP je C_W a položíme $V_L = C_W$
- Nastavíme $C_W = C_B = V_L$ a pokračujeme dalším vrcholem za novým V_L
- Pokud nastane případ (b), V_k se stává kandidátem na vrchol MPP.
- Pokud je V_k konvexní (tj. W-vrchol), nastavíme $C_W = V_k$, jinak nastavíme $C_B = V_k$. Poté pokračujeme dalším vrcholem v seznamu
- Pokud nastane případ (c), dalším vrcholem MPP je C_B a položíme $V_L = C_B$
- Nastavíme $C_W = C_B = V_L$ a pokračujeme dalším vrcholem za novým V_L
- Algoritmus končí, jakmile se opět dostane k prvnímu vrcholu
- Vrcholy V_L nalezené algoritmem tvoří vrcholy MPP

Aproximace okraje

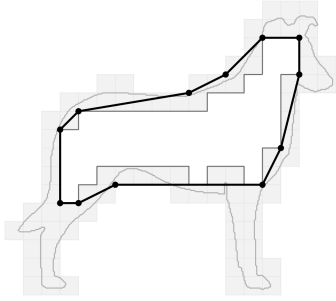
MPP algoritmus



Buňky



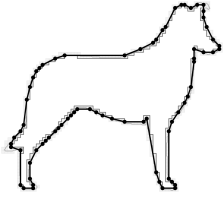
B a W vrcholy



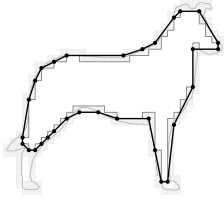
MPP

Aproximace okraje

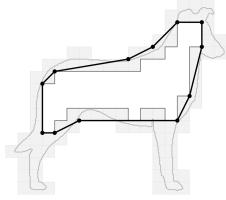
MPP algoritmus



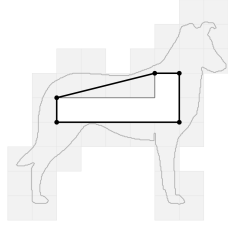
4



8



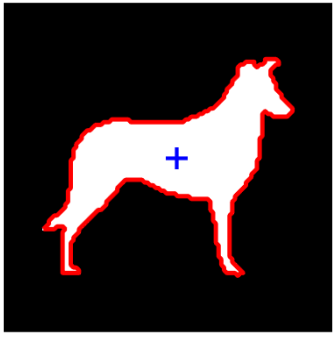
16

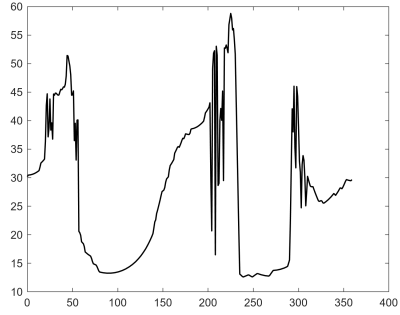


32

Signatura

- **Signatura** = jednorozměrná funkční reprezentace dvourozměrné hranice
- Je možné ji vytvořit různými způsoby
- Jedním z nejjednodušších je vykreslit vzdálenost od těžiště k hranici jako funkci úhlu



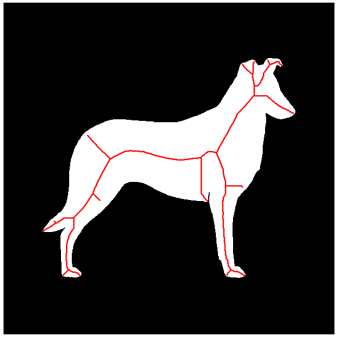


Signatura

- Jednorozměrné funkce jsou snazší na popis, než dvourozměrná informace
- Změny velikosti objektu vedou ke změnám amplitudy funkce
- Provádí se normalizace aby rozsah hodnot byl vždy $[0, 1]$
- Problém je, pokud se objevuje šum, pak normalizace nemusí vyjít správně
- Robustnější (ale výpočetně náročnější) přístup spočívá v dělení každého vzorku rozptylem signatury
  za předpokladu, že rozptyl není nulový nebo příliš malý
- Další možné signatury:
  - Podél hranice a pro každý bod zaznamenáme úhel mezi tečnou k hranici v daném bodě a referenční přímkou
  - Hustotní funkce sklonu, což je histogram hodnot úhlů tečen

Skelety, mediální osy a distanční transformace

- Podobně jako hranice souvisejí s tvarem oblasti i **skelety**
- Skelet oblasti je množina bodů v oblasti, které jsou stejně vzdálené od její hranice
- Oblast redukujeme na strom nebo graf pomocí jejího skeletu
- Jak získat skelet:
  - postupným ztenčováním oblasti (např. pomocí morfologické eroze)
  - výpočtem mediální osy oblasti pomocí efektivní implementace transformace mediální osy (MAT)

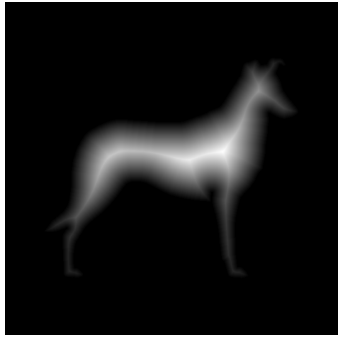


Skeletonizace pomocí morfologie

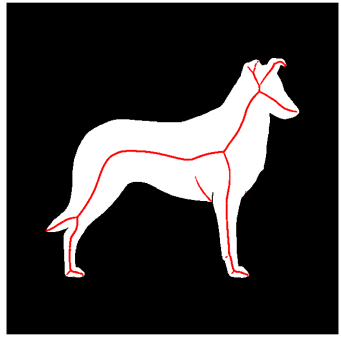
Skelety, mediální osy a distanční transformace
MAT

- MAT oblasti R s hranicí B
- Pro každý bod $p \in R$ najdeme jeho nejbližší bod v B
- Pokud má p více než jednoho takového nejbližšího souseda, patří do mediální osy oblasti R
- „Nejbližší“ soused (a tedy i výsledná MAT) závisí na zvolené metrice vzdálenosti
- Potřebujeme stanovit vzdálenosti každého vnitřního bodu ke všem bodům hranice (nepraktické)
- Využíváme distanční transformaci
- **Distanční transformace oblasti** popředí (nenulových pixelů) v pozadí nul je definována jako vzdálenost každého pixelu k nejbližšímu nenulovému pixelu
- Nás ale zajímá vzdálenost k hranici (nulovým bodům) $\rightarrow$ počítáme distanční transformaci k doplňku
- MAT skelet odpovídá hřebeni distanční transformace (tj. množině lokálních maxim)

Skelety, mediální osy a distanční transformace
MAT



Distanční transformace



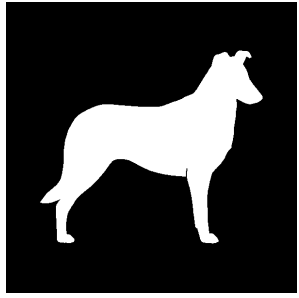
MAT

Deskriptory hranic

- **Délka hranice** – jeden z nejjednodušších deskriptorů
- Aproximace délky = počet pixelů tvořící hranici
- Křivka reprezentována řetězovým kódem s jednotkovým krokem = součet počtu horizontálních a vertikálních segmentů plus $\sqrt{2}$ krát počet diagonálních segmentů
- Hranice reprezentována polygonální křivkou = součet délek jednotlivých segmentů

- **Průměr hranice** B: $\text{diameter}(B) = \max_{i,j} D(p_i, p_j)$,
  D metrika vzdálenosti
  p_i, p_j jsou body na hranici

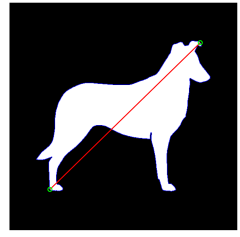
- Délka hranice (počet pixelů hranice): 1728
- Délka hranice (řetězový kód): 1966
- Průměr hranice: 553.7



Deskriptory hranic

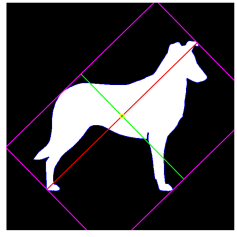
- Průměr + orientace úsečky spojující tyto dva krajní body ((x_1, y_1) a (x_2, y_2)) = **hlavní osa** (nebo nejdelší tětiva)
- Délka hlavní osy: délka $= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
- Úhel hlavní osy: úhel $= \tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right)$

- Délka hlavní osy: 553.7
- Úhel hlavní osy: -44.34 stupňů



Deskriptory hranic

- **Vedlejší osa** (nejdelší kolmá tětiva) = přímka kolmá k hlavní ose, jejíž délka je taková, že obdélník procházející čtyřmi krajními průsečíky hranice s oběma osami zcela obsahuje danou hranici
- Tomuto obdélníku říkáme **bounding box** (ohraničující obdélník)
- Poměr délek hlavní a vedlejší osy = **excentricita hranice**

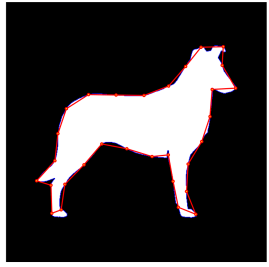


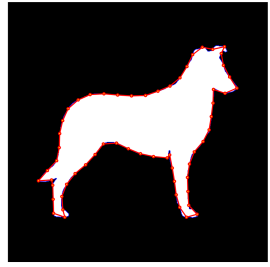
- Délka vedlejší osy: 387.97
- Excentricita: 1.4272

Deskriptory hranic

- **Zakřivení hranice** = rychlost změny sklonu
- U digitální hranice je to těžké určit (jsou lokálně „zubaté“)
- Robustnější je používat rozdíly sklonů sousedních segmentů (aproximovaných přímkami)
- Ve směru hodinových ručiček je vrchol p konvexní, pokud změna sklonu v tomto bodě je nezáporná; jinak je konkávní
- Můžeme zavádět intervaly sklonů – téměř přímkový (10°), roh ($90^\circ \pm 30^\circ$) a jiné
- Deskriptory založené na změnách sklonu = **Sklonový řetězový kód** (SCC)
- **Tortuozita** = míra „kroucení“ křivky
- Při použití SCC = součet absolutních hodnot prvků řetězového kódu $t = \sum_{i=1}^{n} a_i$

Deskriptory hranic





Počet segmentů	30	60
Délka segmentu	56.5490	29.3415
Tortuozita	393.1510	1857.5683

Příklad

Jak by vypadala Tourtuozita čtverce?

Deskriptory hranic

Tvarová čísla

- Hranice reprezentované Freemanovým řetězovým kódem, založeným na 4-směrovém kódu
- První diference s nejmenší možnou hodnotou
- Řád n tvarového čísla je definován jako počet číslic v jeho zápisu
- Pro uzavřenou hranici je n sudé a jeho hodnota omezuje počet možných různých tvarů





Řád	4	6	8	8	8
Řetězový k.	0 3 2 1	0 0 3 2 1	0 0 3 3 2 2 1 1	0 3 0 3 2 2 1 1	0 0 0 3 2 2 2 1
Rozdíl	3 3 3 3	3 0 3 3 0 3	3 0 3 0 3 0 3 0	3 3 1 3 3 0 3 0	3 0 0 3 3 0 0 3
Tvarové č.	3 3 3 3	0 3 3 0 3 3	0 3 0 3 0 3 0 3	0 3 0 3 3 1 3 3	0 0 3 3 0 0 3 3

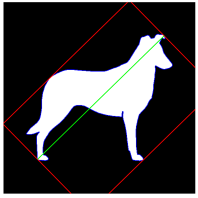
Deskriptory hranic

Tvarová čísla

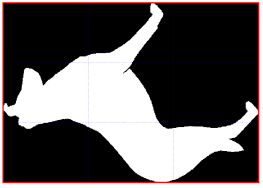
- První diference 4-směrového řetězového kódu je invariantní vůči rotaci (po násobcích 90°)
- Kódovaná hranice obecně závisí na orientaci mřížky
- Závislost lze odstranit zarovnáním mřížky řetězového kódu s bounding boxem
- Pro zvolený řád tvaru hledáme obdélník řádu n, jehož excentricita co nejlépe odpovídá excentricitě základního obdélníku
- Tento nový obdélník použijeme k určení velikosti mřížky
  Pro $n = 12$ mají všechny obdélníky řádu 12 (tj. s obvodem 12) rozměry 2×4, 3×3 a 1×5
  Pokud excentricita obdélníku 2×4 nejlépe odpovídá excentricitě základního obdélníku dané hranice, vytvoříme mřížku 2×4
- Tento postup vede k tomu, že řád může být ve výsledku vyšší, v tom případě snížíme n a postup opakujeme
- Řád je vždy sudý

Deskriptory hranic

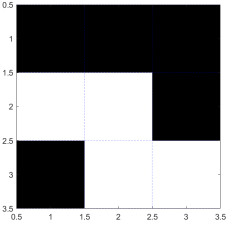
Tvarová čísla



Bounding box



Mřížka



Tvar

- $n = 12$
- Freemanův kód: 0 0 3 0 3 2 2 1 2
- 1. difference: 0 3 1 3 3 0 3 1 2
- Tvarové číslo: 0 3 1 2 0 3 1 3 3

Příklad

Jaký je řád tvarového čísla?

Deskriptory hranic

Fourierovy deskriptory

- Při průchodu hranice (např. proti směru hodinových ručiček) od dostaneme posloupnost bodů $(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_{K-1}, y_{K-1})$
- Můžeme zavést funkce $x(k) = x_k$ a $y(k) = y_k$
- Hranice: $s(k) = [x(k), y(k)]$, $k = 0, 1, 2, \ldots, K-1$
- Souřadnice můžeme chápat jako komplexní čísla $s(k) = x(k) + jy(k)$
  Osa x odpovídá reálné části a osa y imaginární části komplexní posloupnosti
- Hranice se nemělní, ale převádíme problém z 2D na 1D popis
- Diskrétní Fourierova transformace (DFT) posloupnosti $s(k)$ je
  $a(u) = \sum_{k=0}^{K-1} s(k) e^{-j2\pi uk/K}$
  $u = 0, 1, 2, \ldots, K-1$
- Komplexní koeficienty $a(u)$ se nazývají **Fourierovy deskriptory hranice**
- Inverzní transformace rekonstruuje $s(k)$:
  $s(k) = \frac{1}{K} \sum_{u=0}^{K-1} a(u) e^{j2\pi uk/K}$
  $k = 0, 1, 2, \ldots, K-1$

Deskriptory hranic

Fourierovy deskriptory

- Pokud použijeme všechny koeficienty $\rightarrow$ přesná hranice
- Pokud použijeme pouze prvních P koeficientů (a ostatní položíme rovny nule), dostaneme aproximaci
  $\hat{s}(k) = \frac{1}{K} \sum_{u=0}^{P-1} a(u) e^{j2\pi uk/K}$
- Počet bodů hranice zůstává K, ale pro rekonstrukci každého bodu se používá méně členů
- Kvůli periodicitě DFT je nutné transformaci před filtrací centrovat násobením $(-1)^x$ a po inverzi tento krok zrušit
- Z důvodu symetrie musí být počet bodů hranice i počet odstraněných koeficientů sudý (koeficienty se nulují symetricky na obou koncích spektra)

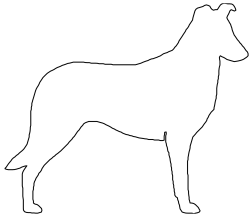
Příklad

Jak odstraníme vysokofrekvenční koeficienty?

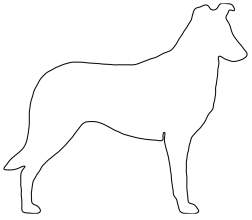
- Čím méně bude P, tím ztratíme více detailů

Deskriptory hranic

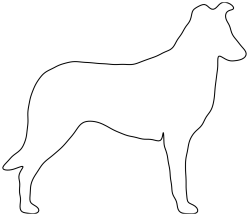
Fourierovy deskriptory



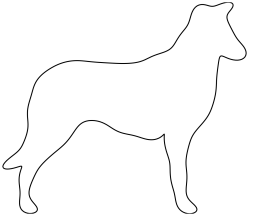
Originál



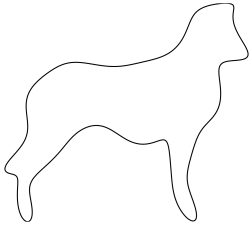
$P = 256$



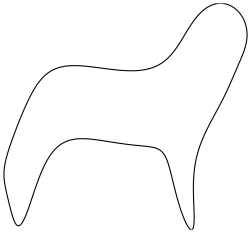
$P = 128$



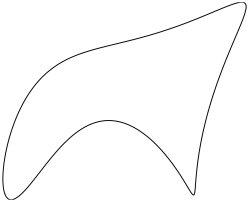
$P = 64$



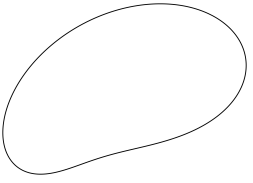
$P = 32$



$P = 16$



$P = 8$



$P = 4$

Deskriptory hranic

Fourierovy deskriptory

- Několik málo Fourierových deskriptorů postačuje k zachycení podstaty tvaru hranice
- Můžeme je použít jako příznakový vektor
- Deskriptory by měly být co nejméně citlivé na translaci, rotaci a změnu měřítka, a také na volbu počátečního bodu
- Fourierovy deskriptory nejsou vůči těmto transformacím přímo invariantní
- Jejich vliv lze vyjádřit jednoduchými transformacemi koeficientů
- Např. rotace o φ odpovídá násobení každého bodu faktorem $e^{j\varphi}$
  $s(k)e^{j\varphi}$
  $a_r(u) = a(u)e^{j\varphi}$

Transformace	**Hranice**	**Fourierův Deskriptor**
Identita	$s(k)$	$a(u)$
Rotace	$s_r(k) = s(k)e^{j\theta}$	$a_r(u) = a(u)e^{j\theta}$
Posunutí	$s_t(k) = s(k) + \Delta_{xy}$	$a_t(u) = a(u) + \Delta_{xy}, \delta(u)$
Změna měřítka	$s_s(k) = \alpha s(k)$	$a_s(u) = \alpha a(u)$

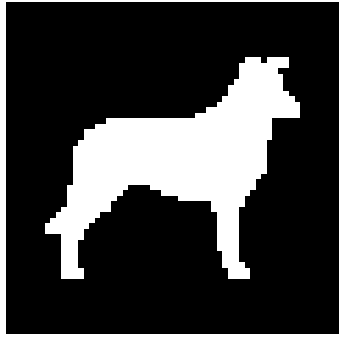
Deskriptory hranic

Statistické momenty

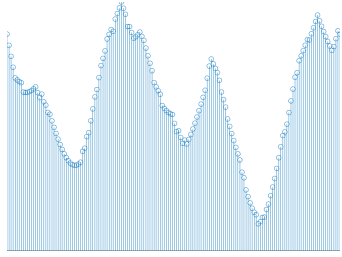
- Jednorozměrné reprezentace dvourozměrných hranic
- Například signatura, je diskrétní funkce $g(r)$ jedné proměnné r
- Amplitudu funkce g budeme považovat za diskrétní náhodnou proměnnou z
- Vytvoříme histogram amplitud $p(z_i)$, $i = 0, 1, 2, \ldots, A-1$ (pokud ho normalizujeme, dostaneme pravděpodobnost výskytu hodnoty)
  A je počet diskrétních úrovní amplitudy
- n-tý moment proměnné z vzhledem k jejímu průměru je
  $m_n(z) = \sum_{i=0}^{A-1}(z_i - m)^n p(z_i)$
  $m = \sum_{i=0}^{A-1} z_i p(z_i)$
- m je střední hodnota (průměr) proměnné z a m_2 je její rozptyl
- Obecně postačuje několik prvních momentů k rozlišení signatur odpovídajících zřetelně odlišným tvarům

Deskriptory hranic

Statistické momenty



Obrázek



Signattura

Momenty

n	1	2	3	4
$m_n(z)$	17.3862	40.0576	-76.9283	3676

Deskriptory hranic

Statistické momenty

- Alternativa: normalizace plochy pod křivkou $g(r)$ a chápat jí jako histogram
- $g(r_i)$ = pravděpodobnost výskytu hodnoty r_i (r je proměnná náhodná veličina)
- Momenty:
  $m_n(r) = \sum_{i=0}^{K-1}(r_i - m)^n g(r_i)$,
  $m = \sum_{i=0}^{K-1} r_i g(r_i)$
  K je počet bodů na hranici