STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

obor 01

Praktické využití metod digitálního zpracování obrazu

Vypracoval:
Ondřej Mikšík
Oktáva B
Gymnázium Kroměříž
Masarykovo nám. 496

2007

Obsah

1. Úvod

Práce se zabývá teoretickými základy digitálního zpracování obrazu a jejich aplikací při řešení konkrétního technického zadání.

Původním záměrem autora byla účast v soutěži autonomních robotických vozítek. Tato autíčka získávají informace o svém okolí pomocí nejrůznějších čidel, jako jsou kamery, infračervené senzory, ultrazvukové dálkoměry. Tyto údaje jsou vyhodnocovány a vozítko na ně vhodným způsobem reaguje.

Klíčovým nástrojem při řešení této problematiky jsou metody digitálního zpracování obrazu. Tento rozsáhlý obor na rozhraní matematiky a pokročilé výpočetní techniky autora zaujal natolik, že veškeré úsilí nakonec věnoval teoretickému řešení problematiky a sestrojení původně zamýšleného robotického vozítka přenechal jiným nadšencům.

Metody, jimiž se autor zabýval, si však cestu do praxe nakonec stejně našly. Další část práce popisuje řešení konkrétního technického požadavku zpřesnění kalibrace přístrojů pro výzkum živé přírody.

Závěrečná kapitola shrnuje výsledky, kterých se při řešení problematiky digitálního zpracování obrazu podařilo dosáhnout.

2. Digitální obraz

Digitální obraz chápeme jako snímek sejmutý kamerou, scannerem, nebo jiným zařízením a definujeme jej jako obrazovou funkci f(x, y), jejíž hodnoty odpovídají měřené veličině (například jasu). Při počítačovém zpracování obrazů pracujeme s digitalizovanými obrazy. Obrazová funkce f(x, y) je poté představována maticí, jejíž prvky nazýváme pixely (obrazové elementy). Tyto obrazové elementy, jsou nejmenšími, dále nedělitelnými jednotkami, což je důležité například při měření vzdáleností v obraze - nepohybujeme se v Euklidovském prostoru, ale v rastru (čtvercový, hexagonální, …) [2].

Obr. 1. Testovací dráha pokusného vozítka

Cílem první části práce je nalézt polohu „patníků“, které představují světla nalevo od chodníku v obr1. Nyní se budeme věnovat metodám, které vedou k řešení tohoto problému.

2.1. Předzpracování obrazu

Předzpracování obrazu je operace, kterou musíme provést u každého zpracovávaného obrazu. Většinou se nám totiž nepodaří získat kvalitní obraz, který má vysoký kontrast, neobsahuje šum, je ostrý, … Cílem operací prováděných při předzpracování je obraz upravit tak, aby následná lokalizace a rozpoznávání objektů bylo co nejpřesnější a nejrychlejší.

Většina operací prováděných při předzpracování je protichůdná. Při odstranění šumu se snažíme obraz vyhladit, což způsobuje rozmazání tenkých čar a hran, které jsou důležité. Naopak při ostření obrazu (zvýraznění vyšších frekvencí) dochází zároveň k zvětšení podílu obrazového šumu (protože ten je také vysokofrekvenční). Proto je většina prováděných operací závislá na hledání kompromisu.

Histogram

Jednou z nejjednodušších pomůcek při digitálním zpracování obrazu je histogram obrazu, který je ve stupních šedi [2]. Tato funkce sumarizuje hodnoty úrovně šedi v obrázku. Každý obraz má pouze jeden histogram, ale jeden histogram může odpovídat více obrazům

(když posuneme objekt v obraze před konstantním pozadím, nebude to mít na výsledný histogram vliv).

Histogram úrovně šedé je funkce, která obsahuje pro každou úroveň šedi (intensitu jasu), počet pixelů (bodů) v obraze, které dané úrovni odpovídají, ale nedává nám žádné informace o tom, kde se tyto body v obraze vyskytují. Histogram osmibitového obrazu tedy obsahuje 256 různých úrovní šedi.

Matematicky jej můžeme definovat jako

$$N = \sum_{k=0}^{255} h_k \tag{1}$$

kde h_k je celkový počet pixelů, které odpovídají úrovni šedi k.

Ekvalizace histogramu

Ekvalizace histogramu [2] je algoritmus, který změní výskyt jasových úrovní v obraze tak, aby intenzity jasu byly zastoupeny v co nejširším rozmezí a pokud možno se stejnou četností. Ekvalizací zvýšíme kontrast obrazu. Nové hodnoty jasu spočítáme jako

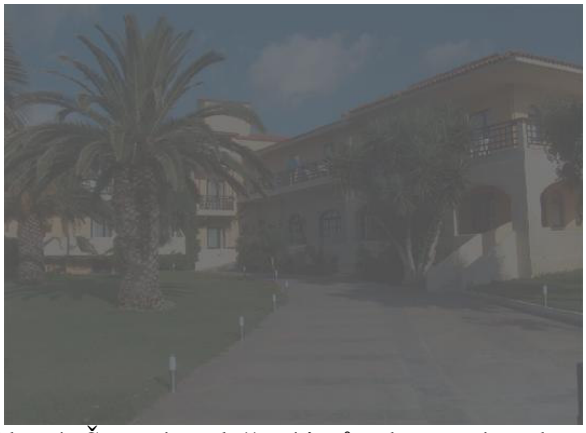
$$J = \frac{I}{xy} \sum_{i=I_0}^{i=I} h(i) \tag{2}$$

Kde I je maximální hodnota jasu v původním obraze, xy je počet pixelů původního obrazu, I_0 je nejnižší hodnota jasu v původním obrazu a h(i) je počet pixelů odpovídajících této hodnotě.

Obr. 2. Špatné rozložení jasů v obraze

Obr. 3. Obraz po ekvalizaci

Stejně je možné provádět ekvalizaci histogramu barevných obrazů. Tu můžeme provést několika způsoby, například ekvalizací každé barevné složky zvlášť. To však může vést ke změně poměrů barev v obraze, což nemusí být vždy vhodné. Další možností je provést ekvalizaci v některém alternativním barevném prostoru, například YIQ, ve kterém Y nese informace o hodnotě jasu daného pixelu a tudíž při ekvalizaci nedojde ke zkreslení barev.



Obr. 4. Špatné rozložení jasů v barevném obraze

Obr. 5. Barevný obraz po ekvalizaci

Ekvalizací histogramu zvýšíme kontrast obrazu, což nám usnadní odlišit jednotlivé objekty od sebe.

Prahování

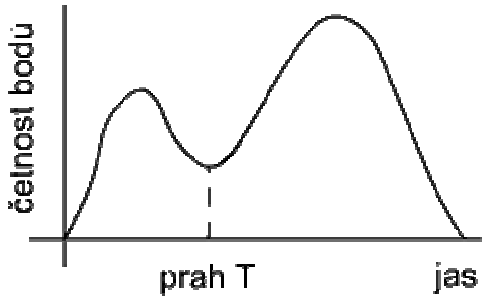
Často potřebujeme odlišit objekty zájmu od pozadí. Nejčastěji používanou metodou pro tuto operaci je prahování (thresholding) [1]. Prosté prahování funguje tak, že máme vstupní obraz a práh. Potom

$$f(i,j) = \begin{cases} a[i,j] \geq T \Rightarrow a[i,j] = 1 \\ a[i,j] < T \Rightarrow a[i,j] = 0 \end{cases} \tag{3}$$

Výsledkem této funkce je binární obraz, kde pixely, které mají jas větší nebo roven prahu jsou označeny jako 1 neboli popředí a pixely, jejichž jas je menší než práh jsou označeny jako 0 neboli pozadí.

Metoda, při které používáme pouze jeden práh, je užitečná pouze ve velmi jednoduchých aplikacích a i tak skýtá mnohá úskalí při volení prahu.

Mezi nejčastěji používané metody volby automatického prahu patří metoda standardního histogramu, metoda p-procentního prahu nebo metody váženého/vysokého gradientu.



Obr. 6. Bimodální histogram s automaticky voleným prahem

Filtrace šumu v obraze

Při filtrování šumu z jediného obrazu nám bohužel nezbývá, než se spolehnout na přebytečnost informací v obraze (obraz nemáme s čím porovnat). Budeme vycházet z toho, že sousední pixely v obraze jsou tvořeny převážně stejnými, nebo podobnými hodnotami jasů. Proto se nová hodnota pixelu odhaduje z jeho malého okolí (často čtverec s lichým počtem řádků/sloupců). Bohužel při těchto filtracích se rozmazávají hranice objektů, což stěžuje jejich následnou identifikaci a přesnou lokalizaci [3].

Jedním z nejběžnějších filtrů je prosté průměrování. Při použití tohoto filtru je obraz systematicky procházen například po řádcích (princip konvoluce) a nová hodnota pixelu je vypočtena jako aritmetický průměr pixelů, které jsou pokryty filtrační maskou (kernelem). Jak již bylo zmíněno, tento filtr velmi ničí linie, ale na druhou stranu je dobře použitelný pro filtraci rozsáhlých homogenních ploch a je velmi rychlý.

$$f(i,j) = \frac{1}{(2n+1)(2m+1)} \sum_{k=-n}^{n} \sum_{l=-m}^{m} f(i+k, j+l) \tag{4}$$

Obr. 7. Obraz s přidaným umělým šumem

Obr. 8. Prosté průměrování (5x5)

Dalším často používaným filtrem je medián. Medián je číslo, které se po uspořádání nachází uprostřed posloupnosti. Jeho nalezení zjednoduší vhodně zvolená velikost zkoumaného okolí (opět čtverec s lichým počtem řádků/sloupců). Navíc není potřeba vytvořit posloupnost, ale stačí hodnoty částečně uspořádat. Jeho nevýhody jsou, že poškozuje jemné čáry a ostré rohy v obraze.

88	95	**80**
68	100	56
82	75	60

56, 60, 68, 75, **80**, 82, 88, 95, 100

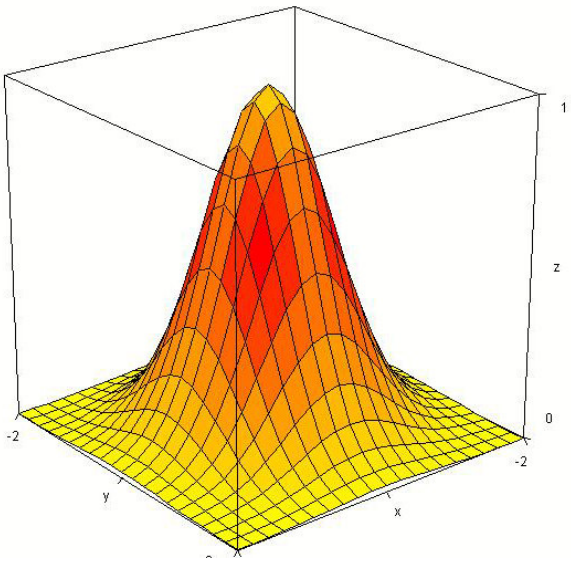
Tabulka 1. Příklad použití mediánu

Obr. 9. Obraz s přidaným umělým šumem

Obr. 10. Iterativně použit medián (3x3)

Gaussův filtr (Gaussian blur) má podobné vlastnosti jako prosté průměrování, ale konvoluční maska se spočítá z 2D tvaru Gaussiánu. Od prostého průměrování se odlišuje tím, že se počítá vážený průměr oblasti se zvýrazněním středu.

$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{(x-\mu)^2}{2\sigma_x^2} - \frac{(y-\mu)^2}{2\sigma_y^2}} \tag{5}$$



Obr. 11. 2D tvar Gaussiánu, σ=2,5

Obr. 12. Obraz s přidaným umělým šumem

Obr. 13. Gaussův filtr (5x5), σ=2,5

Největším problémem při filtrování Gaussiánem je volba správného σ a okolí v kterém Gaussův filtr reaguje. Při zvolení příliš malého σ nebo okolí, je filtrace nevýrazná, ale zase nepoškozuje hrany. Při zvolení vyšších hodnot σ a filtrovaného okolí je filtrace výrazná, ale lokalizace hran přestává být přesná, popřípadě některé méně výrazné hrany zaniknou úplně.

Samozřejmě, že filtrovacích metod existuje celá řada. Mezi zde nezmíněné, ale poměrně často používané patří například Rotující maska, Binomický filtr…

2.2. Matematická morfologie

Původní využití matematické morfologie spočívalo v úpravě binárních obrazů, ale dnes se morfologie využívá i při úpravě obrazů ve stupních šedi.

Morfologii [5], [6], [7] chápeme při digitálním zpracování obrazů jako nástroj pro předzpracování a segmentaci obrazů. Jedná se o množinu metod, pomocí kterých můžeme zkoumat tvar, kostru nebo konvexní obal. Výsledek morfologických metod je závislý pouze na rozložení pixelů v obraze a není závislý na jejich hodnotách.

Základem matematické morfologie je fakt, že obrazy lze modelovat pomocí bodových množin. Morfologické operace jsou většinou založeny na zvětšování a zeštíhlování zkoumaných tvarů. Mezi základní morfologické operace při binární morfologii patří Dilatace a Eroze.

Binární dilatace

Binární dilatace se používá k zaplnění malých děr v objektech. Výsledkem binární dilatace je objekt, který je větší, než byl objekt původní.

Při binární dilataci sčítáme množinu X s množinou X‘, která je od množiny X posunuta o strukturní element B (druhá, většinou menší množina).

$$X \oplus B = \{ p \in \mathrm{E}^2 : p = x + b, x \in X \wedge b \in B \} \tag{6}$$

Což můžeme vyjádřit Minkowského součtem (sjednocení posunutých množin):

$$X \oplus B = \bigcup_{b \in B} X_b \tag{7}$$

Binární eroze

Binární eroze je duální morfologickou transformací k dilataci. Při binární erozi skládáme dvě množiny pomocí Minkowského rozdílu. Erozi také můžeme vyjádřit jako průnik všech posunů obrazu X o vektory $-b \in B$

$$X \Theta B = \bigcap_{b \in B} X_{-b} \tag{8}$$

Binární eroze se používá ke zjednodušení objektu. Objekty, které jsou menší než strukturní element, z obrazu zmizí. Výsledkem binární eroze je objekt, který je menší než původní

Binární otevření

Jedná se o erozi následovanou dilatací. Při binárním otevření odstraníme z obrazu malé objekty.

$$X \circ B = (X \Theta B) \oplus B \tag{9}$$

Pokud se obraz X po otevření strukturním elementem B nezmění, říkáme, že obraz X je otevřený vzhledem ke strukturnímu elementu B.

Binární uzavření

Jedná se o dilataci následovanou erozí. Při binárním uzavření odstraníme z obrazu malé díry.

$$X \bullet B = (X \oplus B) \Theta B \tag{10}$$

Pokud se obraz X po uzavření strukturním elementem B nezmění, říkáme, že obraz X je uzavřený vzhledem ke strukturnímu elementu B.

Důležitou vlastností dilatace a eroze je idempotence. To znamená, že po jednom otevření/uzavření je již obraz otevřen/uzavřen, takže nemá smysl je provádět vícekrát.

Hit or miss

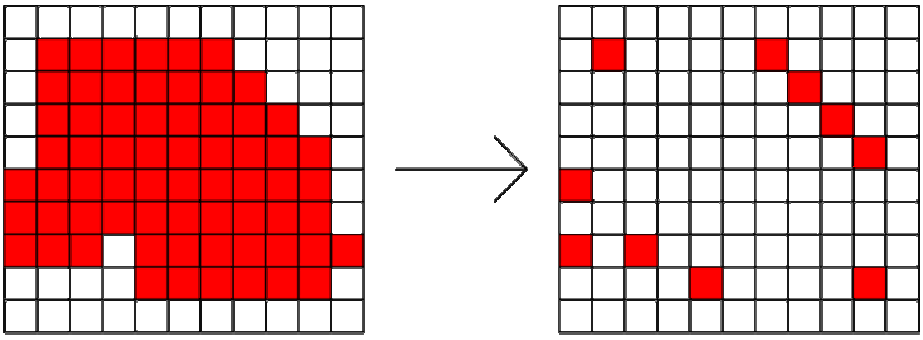
Hit or miss (tref či miň) transformace se používá k hledání zvláštních částí objektů jako jsou například rohy, nebo ke ztenčování či zvýraznění hran. Tato transformace používá složený strukturní element, pro který platí $B = (B_1, B_2)$ a zároveň $B_1 \cap B_2 = \varnothing$, přičemž B_1 testuje objekty (popředí) a B_2 testuje pozadí.

Transformaci hit or miss můžeme vyjádřit také pomocí erozí

$$X \otimes B = (X \Theta B_1) \cap (X^C \Theta B_2) \tag{11}$$

Jak již bylo uvedeno, transformaci hit or miss můžeme použít k nalezení rohů konvexních obalů objektů. Roh může být tvořen čtyřmi variantami uspořádaných pixelů, proto nadefinujeme následující čtyři masky (1 reprezentuje objekt, 0 reprezentuje pozadí a * reprezentuje libovolný pixel).

$$\begin{bmatrix} * & 1 & * \\ 0 & 1 & 1 \\ 0 & 0 & * \end{bmatrix} \quad \begin{bmatrix} * & 1 & * \\ 1 & 1 & 0 \\ * & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} * & 0 & 0 \\ 1 & 1 & 0 \\ * & 1 & * \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & * \\ 0 & 1 & 1 \\ * & 1 & * \end{bmatrix} \tag{12}$$



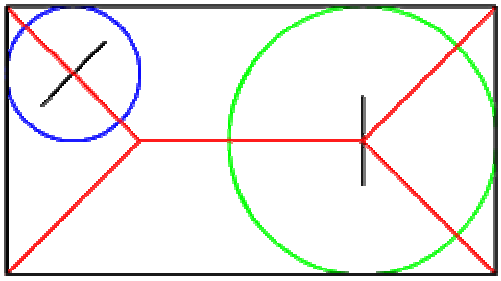
Obr. 14. Příklad použití transformace Hit or miss

V případě, že maska se shoduje s okolím zkoumaného pixelu, je daný pixel označen jako 1, jinak je označen jako 0.

Skeleton

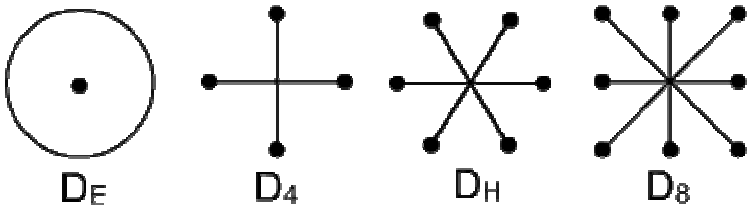
Řadu objektů má smysl reprezentovat pouze jejich kostrou. Toto zobrazení má využití například u sledování pohybů člověka.

Kostra musí být tlustá jeden pixel, měla by vést středem objektu a musí zachovávat jeho tvar. Kostru lze také definovat pomocí tzv. maximálních kruhů (nebo v prostoru pomocí koulí). Maximální kruh je takový, který je vepsaný do objektu a dotýká se jeho hran ve dvou nebo více bodech. Kostra objektu je potom tvořena sjednocením středů maximálních kruhů.



Obr. 15. Kostra obdélníku

Při zpracování digitálního obrazu se zpravidla nepohybujeme v euklidovském prostoru, proto mají kruhy podobu právě takového prostoru v jakém se zrovna pohybujeme (D_E, D_4, D_H, D_8) [2].



Obr. 16. Kruhy v různých prostorech – zleva: Euklidovský prostor, 4-okolí, hexagonální prostor, 8-okolí

Protože je ale tvoření kostry pomocí maximálních kruhů výpočetně náročné, používají se jiné algoritmy jako je koutková reprezentace (objekt je nejdříve bezeztrátově zkomprimován a poté se do něj vpisují maximální obdélníky), sekvenční ztenčování nebo vzdálenostní transformace.

Abychom mohli používat sekvenční ztenčování/ztlušťování, musíme nejdříve zavést jednoduché ztenčování/ztlušťování. Ztenčování můžeme popsat jako odečtení části ztenčované oblasti určené složeným strukturním elementem $B = (B_1, B_2)$ od objektu

$$X \oslash B = X - (X \otimes B) \tag{13}$$

Ztlušťování je duální transformací k ztenčování a můžeme popsat jako sjednocení s částí pozadí určenou strukturním elementem.

$$X \odot B = X \cup (X \otimes B) \tag{14}$$

Sekvenční ztenčování můžeme popsat také pomocí posloupnosti strukturních elementů.

$$X \oslash \{B_{(i)}\} = (((X \oslash B_{(1)}) \oslash B_{(2)}) \ldots \oslash B_{(n)}) \tag{15}$$

Analogicky můžeme popsat sekvenční ztlušťování

$$X \odot \{B_{(i)}\} = (((X \odot B_{(1)}) \odot B_{(2)}) \ldots \odot B_{(n)})$$

(16)

Existuje řada různých posloupností strukturních elementů, my si ukážeme ty, které jsou popsané v tzv. Golayově abecedě pro oktagonální (D_8) mřížku (rastr). Jednička opět reprezentuje objekt, 0 reprezentuje pozadí a * reprezentuje libovolný pixel.

$$\underset{L1}{\begin{bmatrix} 0 & 0 & 0 \\ * & 1 & * \\ 1 & 1 & 1 \end{bmatrix}} \underset{L2}{\begin{bmatrix} * & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & * \end{bmatrix}} \underset{L3}{\begin{bmatrix} 1 & * & 0 \\ 1 & 1 & 0 \\ 1 & * & 0 \end{bmatrix}} \underset{L4}{\begin{bmatrix} 1 & 1 & * \\ 1 & 1 & 0 \\ * & 0 & 0 \end{bmatrix}} \underset{L5}{\begin{bmatrix} 1 & 1 & 1 \\ * & 1 & * \\ 0 & 0 & 0 \end{bmatrix}} \underset{L6}{\begin{bmatrix} * & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & * \end{bmatrix}} \underset{L7}{\begin{bmatrix} 0 & * & 1 \\ 0 & 1 & 1 \\ 0 & * & 1 \end{bmatrix}} \underset{L8}{\begin{bmatrix} 0 & 0 & * \\ 0 & 1 & 1 \\ * & 1 & 1 \end{bmatrix}}$$

(17)

Následně můžeme obraz sekvenčně ztenčovat/ztlušťovat až dosáhneme idempotence. Podobně existují strukturní elementy pro další typy mřížek (rastrů).

2.3. Edge detection

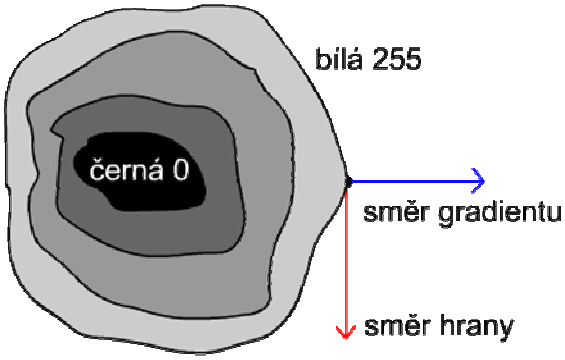
Neurofyziologické a psychologické výzkumy ukazují, že nejdůležitějšími místy v obraze pro vnímání vyšších živočichů jsou místa, kde se prudce mění hodnota jasu. Tato místa se nazývají hrany. Protože tato místa obsahují více informací o obsahu obrazu, pokusíme se je při rozpoznávání obsahu obrazu detekovat. Hrany jsou do jisté míry invariantní vůči změně osvětlení, nebo pohledu [1], [2], [8], [9].

Hrana v obraze je dána vlastnostmi obrazového elementu a jeho okolí a je určena tím, jak náhle se mění hodnota obrazové funkce $f(x, y)$. Změnu funkce dvou proměnných zkoumáme pomocí parciálních derivací. Změnu funkce popisuje její gradient, což je dvousložková vektorová funkce ∇. První složkou je velikost gradientu (modul) a druhá složka udává úhel, pod kterým hrana běží (směr hrany můžeme definovat také jako kolmý na směr gradientu). Hranou nazýváme body, které mají velké hodnoty modulu gradientu.

Gradient spojité funkce n proměnných spočítáme jako vektor parciálních derivací.

$$\nabla f(x_1, \ldots, x_n) = \left(\frac{\partial f}{\partial x_1}, \ldots\ldots, \frac{\partial f}{\partial x_n} \right)$$

(18)



Obr. 17. Souvislost hrany s gradientem

Modul gradientu pro n = 2 spočítáme jako

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{19}$$

a směr gradientu

$$\theta = \arctan\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \tag{20}$$

Obraz je ale diskrétní jasová funkce, proto nejsme schopni obrazové derivace spočítat přesně. Tyto derivace ale můžeme aproximovat pomocí diferencí.

$$\begin{aligned} \Delta_x g(x,y) &= g(x,y) - g(x-n,y) \\ \Delta_y g(x,y) &= g(x,y) - g(x,y-n) \end{aligned} \tag{21}$$

Kde n je malé celé číslo, většinou 1.

Digitální gradient poté spočítáme jako

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{22}$$

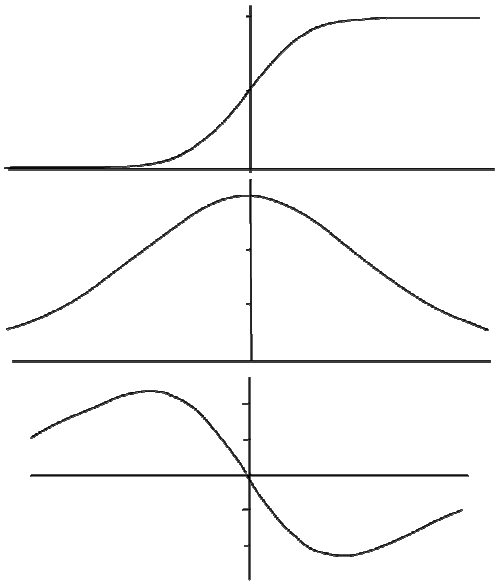
Pro urychlení výpočtu můžeme uvažovat pouze součet absolutních hodnot

$$|G| = |G_x| + |G_y| \tag{23}$$

Směr digitálního gradientu spočítáme jako:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{24}$$

Pro detekci hran používáme detektory založené na hledání maxim první derivace, nebo na hledání průchodů nulou (zero crossing) druhé derivace, což je jednodušší.



Obr. 18. Hrana, první derivace, druhá derivace

Diskrétní konvoluce

Konvoluce je matematický operátor zpracovávající dvě funkce. Nás bude zajímat diskrétní konvoluce, protože zpracováváme diskrétní jasovou funkci (obraz).

$$f(x,y) * h(x,y) = \sum_{i=-k}^{i=k} \sum_{j=-k}^{j=k} f(x-i, y-j).h(i,j) \tag{25}$$

f(x,y) je vstupní obraz a h(i,j) je konvoluční maska (konvoluční jádro)

Při diskrétní konvoluci si lze jako konvoluční jádro představit matici, kterou systematicky přikládáme na příslušný pixel v obrazu. Hodnotu jasu každého pixelu překrytého touto maticí vynásobíme hodnotou v příslušné buňce matice, sečteme a vynásobíme celkovým koeficientem konvoluční masky (například pro konvoluční masku o rozměrech 3x3 je to 0.11 = 1/9). Takto dostaneme novou hodnotu jasu pro jeden pixel.

Pouhou změnou konvoluční masky lze nadefinovat řadu operací (filtrace šumu, detekce hran, …), protože koeficienty uvnitř konvoluční masky udávají vliv hodnoty pixelu

pod nimi. Většinou používáme konvoluční masky s lichým počtem řádků a sloupců, protože potom leží reprezentativní pixel ve středu masky a také můžeme dát středu větší váhu (zvýraznit jej).

Čím je konvoluční maska větší, tím je odolnější vůči šumu v obraze, ale zároveň je výsledný obraz více rozostřen a výpočet je pomalejší.

Robertsův operátor

Robertsova konvoluční maska má rozměry jen 2x2. Nejčastěji se používá na obrazy ve stupních šedi, její výpočet je velmi rychlý, ale také je velmi náchylný k označení náhodného šumu v obraze za hranu.

Velikost gradientu se spočítá jako

$$|G| = |f(x,y) - f(x+1,y+1)| + |f(x+1,y) - f(x,y+1)| \tag{26}$$

Konvoluční matice poté mají následující tvary

$$\underset{G_x}{\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}} \qquad \underset{G_y}{\begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}} \tag{27}$$

Tyto matice dobře detekují hrany, které „běží“ pod úhly 45° a 135°. U dalších filtrů postačí uvést pouze konvoluční masky.

Obr. 19. Detekce hran Robertsovým filtrem

Operátor Prewittové

Operátor Prewittové o rozměrech matice 3x3 pro detekci horizontálních hran:

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \tag{28}$$

Operátor Prewittové o rozměrech matice 5x5 pro detekci svislých hran (transponovaná matice vůči matici pro detekci horizontálních hran):

$$\begin{bmatrix} -2 & -1 & 0 & +1 & +2 \\ -2 & -1 & 0 & +1 & +2 \\ -2 & -1 & 0 & +1 & +2 \\ -2 & -1 & 0 & +1 & +2 \\ -2 & -1 & 0 & +1 & +2 \end{bmatrix} \tag{29}$$

Další matice (pro detekci šikmých hran) opět dostaneme pootočením matice.

Obr. 20. Detekce hran pomocí Operátoru Prewittové

Sobelův operátor

Sobelův operátor se často používá pro detekci vodorovných a svislých hran. Jeho základní tvar reprezentují tyto dvě matice:

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

G_x G_y

(30)

Klidně ale můžeme chtít detekovat hrany pouze ve vodorovném směru, potom tyto matice modifikujeme:

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

(31)

Když matici pootočíme, můžeme detekovat šikmé hrany:

$$\begin{bmatrix} 0 & +1 & +2 \\ -1 & 0 & +1 \\ -2 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -2 \\ +1 & 0 & -1 \\ +2 & +1 & 0 \end{bmatrix} \quad \begin{bmatrix} +2 & +1 & 0 \\ +1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix} \quad \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & +1 \\ 0 & +1 & +2 \end{bmatrix}$$

(32)

Obr. 21. Detekce hran Sobelovým filtrem

Kirschův operátor

Kirschův operátor definuje velikost gradientu jako maximum, po aplikaci všech 8 možných směrových kombinací konvoluční masky.

$$\begin{bmatrix} +5 & -3 & -3 \\ +5 & 0 & -3 \\ +5 & -3 & -3 \end{bmatrix}$$

(33)

Obr. 22. Detekce hran Kirschovým filtrem

Laplaceův operátor

Maximům prvních parciálních derivací diskrétní jasové funkce odpovídají průchody nulou druhé parciální derivace. K tomu se výhodně používá Laplacián.

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{34}$$

Všesměrový lineární Laplaceův operátor (Laplacián) je vhodné použít v případě, že chceme detekovat hrany v obraze, ale nezajímá nás směr hrany. Tento operátor udává stejnou odezvu pro všechny směry, proto nelze získat informaci o směru hran. Lze jej využít při ostření obrazu. Mezi jeho nevýhody patří velká citlivost na šum a dvojitá odezva na tenké linie.

$$\begin{bmatrix} 0 & +1 & 0 \\ +1 & -4 & +1 \\ 0 & +1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & +1 & 0 \\ +1 & -8 & +1 \\ 0 & +1 & 0 \end{bmatrix} \quad \begin{bmatrix} 2 & +1 & 2 \\ +1 & -4 & +1 \\ 2 & +1 & 2 \end{bmatrix} \quad \begin{bmatrix} -1 & +2 & -1 \\ +2 & -4 & +2 \\ -1 & +2 & -1 \end{bmatrix} \tag{35}$$

Různé matice pro Laplaceův operátor (pro 4-okolí, 4-okolí s větší vahou prostředního pixelu, pro 8-okolí, 8-okolí s větší vahou prostředního pixelu)

Obr. 23. Detekce hran Laplaceovým filtrem

LoG, DoG

Protože Laplaceův operátor je velmi citlivý na šum, je třeba vstupní obraz nejdříve vyhladit od šumu a až poté se v něm snažit detekovat hrany. Připustíme tedy rozmazání obrazu Gaussiánem a pro odhad druhé derivace použijeme všesměrový Laplacián. Tento postup se označuje jako LoG operátor (Laplacian of Gaussian). Protože používáme lineární filtry, je možno zaměnit pořadí operací.

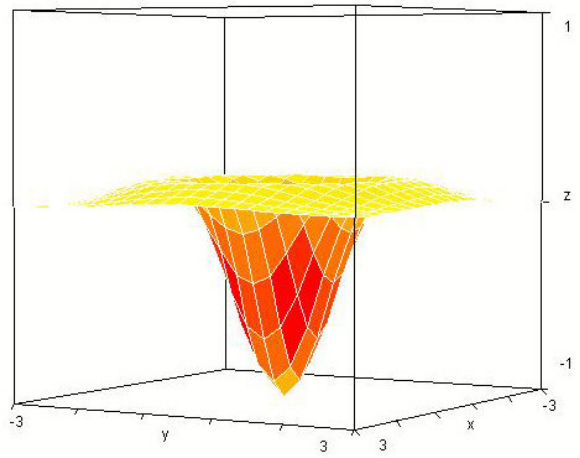
$$\nabla^2(G(x,y,\sigma) * f(x,y)) = (\nabla^2 G(x,y,\sigma)) * f(x,y) \tag{36}$$

Obr. 24. Detekce hran LoG operátorem, σ=2

Hodnoty derivace Gassiánu můžeme předpočítat, protože nezávisí na vstupním obraze. LoG operátor můžeme aproximovat pomocí diference dvou obrazů vyhlazených Gaussiánem o různém σ. Tento operátor se nazývá DoG (Difference of Gaussians).

Nevýhodou těchto operátorů je, že rozmazávají ostré tvary (rohy objektů) a mají snahu spojovat hrany do uzavřených útvarů. Dalším problémem je opět vhodné nastavení hodnot σ.

Obr. 25. Detekce hran DoG operátorem, σ_1=1, σ_2=2,5



Obr. 26. DoG

Cannyho hranový detektor

Cannyho hranový detektor je algoritmus, který zahrnuje několik kroků k dosažení co nejlepšího výsledku při detekci hran v obraze. Mezi základní požadavky patří spolehlivá detekce (tak, aby bylo nalezeno co nejvíce existujících hran), přesná lokalizace hrany (tak, aby byly pozice hrany určeny co nejpřesněji) a jednoznačnost (aby nebyly detekovány neexistující hrany).

Při detekci hran Cannyho hranovým detektorem většinou postupujeme následovně: nejdříve eliminujeme šum v obraze (většinou pomocí Gaussova filtru, ale můžeme použít i jiný filtr), poté nalezneme přibližné směry gradientu a pro každý pixel derivaci ve směru gradientu pomocí vhodné konvoluční masky. V dalším kroku nalezneme lokální maxima těchto derivací a hrany získáme pomocí prahování s hysterezí. V posledním kroku (který se ale většinou nepoužívá) se sloučí hrany, které jsme získali při různě velkých vyhlazeních.

Při použití Cannyho hranového detektoru vzniká několik problémů. Musíme vhodně zvolit σ při použití Gaussova filtru, protože když bude hodnota σ příliš malá, detektor bude detekovat i nevýznamné detaily (snažíme se jakoby o pohled z větší dálky), když bude

hodnota σ příliš velká, ztíží nám to detekci hran, jejich lokalizaci a také tím více slabých hran zanikne.

Dalším problémem je zvolení vhodné konvoluční masky pro určení velikosti a směru gradientu. Většinou se používá Sobelův operátor, protože není příliš citlivý na šum a vrací nejen velikost gradientu, ale i jeho směr.

Při prahování se snažíme potlačit krátké/slabé bezvýznamné hrany (ponecháme pouze ty, které jsou součástí hran silnějších). Tohoto nelze dosáhnout prahováním s jedním prahem, proto volíme prahy dva (minimální T_1 a maximální T_2) mezi kterými může hodnota gradientu kolísat (například vlivem šumu). V případě, že gradient posuzovaného pixelu se nachází mezi zvolenými prahy, je označen jako hrana pouze v případě, že sousedí s bodem, který již byl uznán jako hrana.

Obr. 27. Detekce hran Cannyho detektorem, σ=1,35, T_1=20, T_2=100

Další metody

Výše rozebraný přístup k rozpoznávání obrazu založený na detekci hran není pochopitelně jediný. Mezi další významné metody patří například Detektory rohů v obraze. Mezi tyto detektory patří například Moravec corner detector, který je základem dalších detektorů jako jsou Stephens-Harris detector nebo Kanade-Lucas-Tomasi (KLT) detector. Tyto detektory lze dobře využít například při automatickém vytváření panoramatických fotografií. Více viz [18], [19], [20], [21], [22].

V případě, že vstupní obraz je velmi zašuměn a potřebujeme detekovat přímky v obraze, je velmi vhodné použít Houghovu transformaci, která je založena na transformaci z Kartézského souřadnicového systému do polárního. Tato transformace byla zobecněna a je využitelná i pro detekci složitějších objektů. Více viz [10].

Další významnou skupinou jsou metody založené na Diskrétní Fourierově transformaci, zejména na její počítačově optimalizované podobě (FFT). Tyto metody obecně umožňují převést časové (plošné) průběhy signálů na frekvenční a zpět. Signál ve frekvenční doméně lze poté zpracovat pomocí propustí a jiných typů filtrů jejichž popis přesahuje rozsah této práce, viz [11].

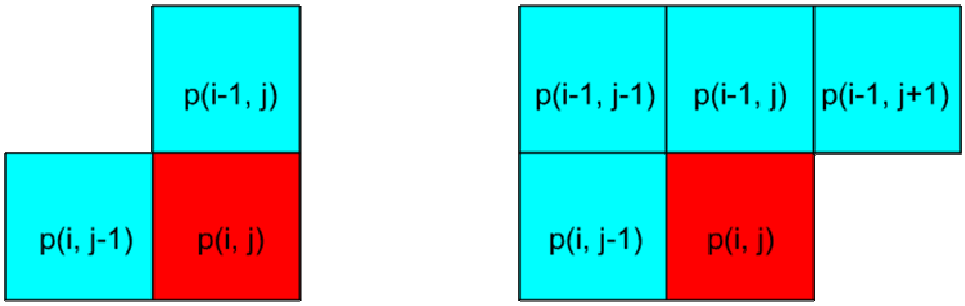
Využití FFT bylo v minulosti limitováno značnou početní náročností, která však vedla k rozvoji specializovaného hardware jako jsou signálové procesory (DSP). Velkým příslibem do budoucna jsou hradlová pole Xilinx Virtex s desítkami zabudovaných DSP řezů umožňujících paralelní zpracování.

2.4. Segmentace

Segmentace obrazu [1], [2], [3] je jeho dělení na oblasti, jejichž vlastnosti jsou v určitých ohledech stejnorodé. Typickou úlohou segmentace je odlišit objekty zájmu od sebe a také od zbývajících objektů nebo pozadí. Existuje řada metod, které lze při segmentaci využít. Mezi základní patří metody založené na zkoumání histogramu (k určení prahu při prahování), další zkoumají podobnost, nebo shodu pixelů v okolí zkoumaného pixelu (tzv. region growing), nebo naopak rozdíly mezi sousedícími pixely.

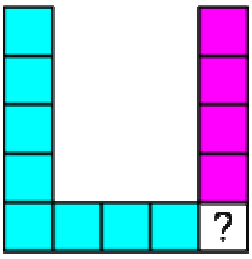
Region growing (connected components labeling)

Při obarvování objektů (neboli obohacování obrazu o pseudobarvy) máme na vstupu binární předzpracovaný, který postupným procházením pomocí obarvovací masky rozdělíme, v tomto případě obarvíme, na jednotlivé objekty. Obarvovací maska má může mít různé tvary (záleží na rastru a povolených směrech pohybu po rastru).



Obr. 28. Obarvovací masky pro 4-okolí a 8-okolí

Při obarvování může nastat problémů při určitých tvarech objektů (například písmena U, V). Proto je potřeba segmentovaný obraz projít znovu a zkontrolovat správnost segmentace.



Obr. 29. Problém u některých tvarů

Obarvováním oblastí také automaticky počítáme objekty v obraze. Počet objektů je roven nejvyšší hodnotě použité při obarvování.

Obr. 30. Obraz obohacený o pseudobarvy

Štěpení a spojování oblastí

Při implementaci tohoto algoritmu musíme obraz rozdělit do skupin a poté zkoumáme, zdali jsou sousedící oblasti stejnorodé. K tomu můžeme použít například zkoumání podobnosti jasů příslušných segmentu. V případě, že jasy sousedících segmentů jsou stejné, můžeme je spojit dohromady v jeden. V případě, že uvnitř segmentu je rozdílný jas, znamená to že segment prochází přes rozhraní dvou oblastí a proto jej rozštěpíme.

2.5. Popis segmentovaného obrazu

Před dalším zpracováním obrazu se musíme pokusit co nejlépe popsat objekty zájmu. K charakterizaci získaných objektů můžeme použít řadu primitivních metod popisujících jejich tvar, velikost, texturu a další vlastnosti [1].

Velikost

Spočítáme jako sumu všech pixelů zkoumané oblasti.

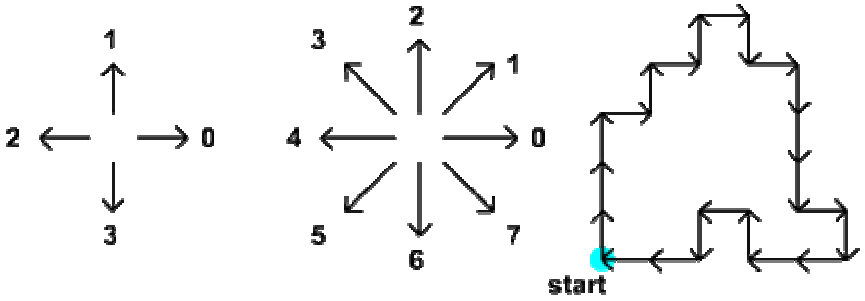
$$v = \sum_{i=0, i \in \Re}^{n} P_i \tag{37}$$

Směr

Směr určujeme pro podlouhlé objekty a určujeme jej podle delší strany opsaného obdélníka.

Řetězové kódy

Řetězové (Freemanovy) kódy se skládají z úseček jednotkové délky. Tyto úsečky jsou označeny čísly, podle toho, jak jsou orientovány. Při reprezentaci objektů pomocí řetězových kódů vyjdeme z jednoho určitého bodu a „pohybujeme“ se po hranici objektu. Tento pohyb poté zaznamenáváme pomocí řady čísel (řetězový kód). Výhodné je tento kód derivovat, protože poté se stává nezávislým na natočení tělesa, což můžeme využít při srovnávání.



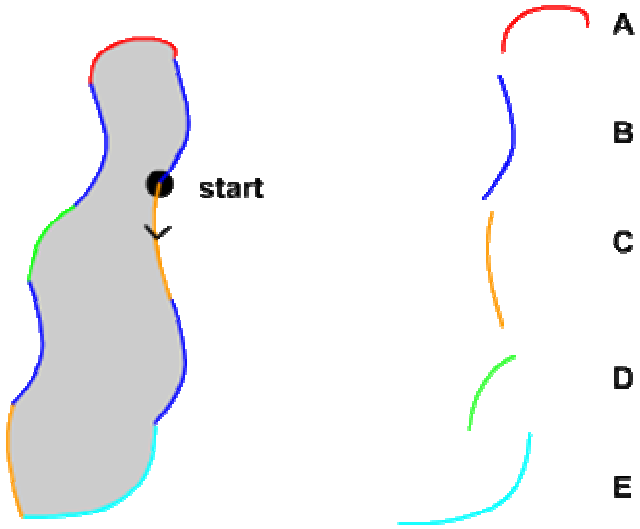
Obr. 31. Matice pro 4-okolí, 8-okolí a vzorový obrazec

Řetězový kód: 11101010303330322123 22
Derivace: 10031313313001330311 30

Derivace řetězového kódu se vypočte jako první diference kódu mod 4 nebo 8 a udává nám kolikrát je nutné obraz otočit o 90° respektive o 45° vlevo, abychom mohli pokračovat v pohybu.

Řetězové kódy mohou být dobře uplatněny při výpočtu délky hranice objektu (zde je třeba mít na paměti, že délka hranice v D_4 je u stejného objektu vždy větší než délka v D_8).

Někdy se tvar části hranice opakuje. V těchto případech je výhodné popisovat hranici řetězovým kódem po těchto stejných částech.



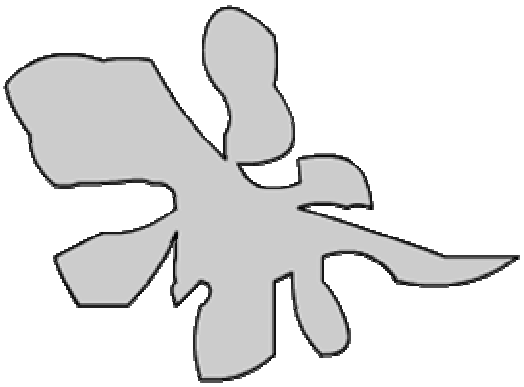
Obr. 32. Popis obrazce pomocí „Letter codes“

Řetězový kód: C, B, E, C, B, D, B, A, B

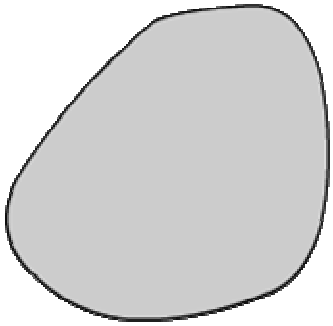
Kompaktnost

Kompaktnost se spočítá jako podíl druhé mocniny velikosti hranice a velikosti zkoumané oblasti. Nejkompaktnější tvar má kruh.

$$c = \frac{B^2}{a} \tag{38}$$



Obr. 33. Nekompaktní tvar

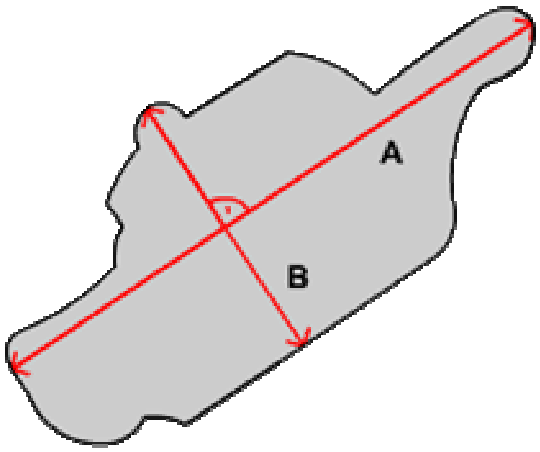


Obr. 34. Kompaktní tvar

Výstřednost

Výstřednost, neboli excentricita je poměr délek nejdelších na sebe kolmých tětiv objektu.

$$e = \frac{\max A}{\max B} \tag{39}$$

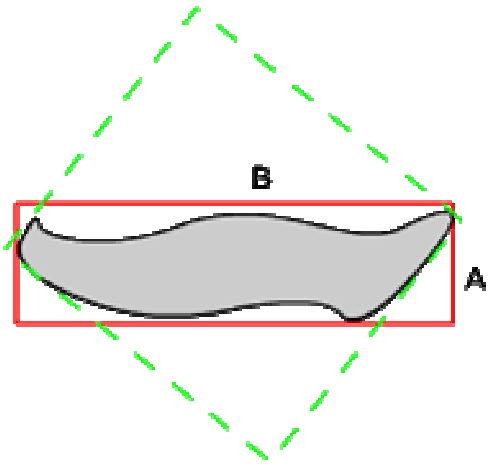


Obr. 35. Excentricita

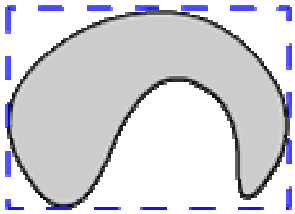
Podlouhlost

Podlouhlost vyjadřuje poměr mezi délkou a šířkou obdélníku opsaného objektu. Ze všech možných obdélníků se vybere ten s nejmenším obsahem. Pro urychlení výpočtu ale obdélník natáčíme po větších krocích (například 15°).

$$el = \frac{b}{a} \tag{40}$$



Obr. 36. Podlouhlý tvar



Obr. 37. Nepodlouhlý tvar

Pravoúhlost

Při pravoúhlosti opisujeme stejně jako u podlouhlosti objektu obdélník, ale jako míru podlouhlosti označíme maximální hodnotu F_k. Hodnoty F_k vypočteme jako podíl velikosti zkoumané plochy a plochy opsaného obdélníku.

$$F_k = \frac{a}{a_r} \tag{41}$$

RTS invariant moments

Jasová funkce obrazu se interpretuje jako hustota pravděpodobnosti dvourozměrné náhodné veličiny. Její vlastnosti můžeme vyjádřit pomocí statistických charakteristik – momentů. Momenty můžeme použít pro popis jak šedých obrazů, tak binárních. Obecně moment definujeme jako

$$m_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^p y^q f(x,y)dxdy \tag{42}$$

Kde f(x, y) je jeden pixel obrazu a p + q je řád momentu. Protože obraz je diskrétní jasová funkce, počítáme se sumami.

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} i^p j^q f(i, j)$$

(43)

Obecné momenty ale nejsou invariantní vůči posunutí, změně měřítka nebo natočení. Vůči posunutí jsou invariantní centrální momenty.

$$\mu_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (i - x_c)^p (j - y_c)^q f(i, j)$$

(44)

Kde x_c a y_c jsou souřadnice středu objektu (centroid) a spočítáme je jako

$$x_c = \frac{m_{10}}{m_{00}} \qquad y_c = \frac{m_{01}}{m_{00}}$$

(45)

Pokud se jedná o binární obraz, m_{00} reprezentuje počet pixelů tvořících popisovaný objekt. Momenty mohou být rozšířeny i o invariantnost vůči změně měřítka.

$$\vartheta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\gamma}}$$

(46)

$$\gamma = \frac{p+q}{2} + 1$$

(47)

Dále mohou být momenty rozšířeny o invariantnost vůči natočení. Druhé momenty jsou definovány jako

$$\varphi_1 = \vartheta_{20} + \vartheta_{02}$$
$$\varphi_2 = (\vartheta_{20} - \vartheta_{02}) + 4\vartheta_{11}^2$$

(48)

Dalších pět momentů je známo pro třetí smíšené momenty.

Pro většinu spočítaných hodnot chybí interpretace, pouze charakterizují popisovaný objekt. Výhodné je využít jejich invariantních vlastností ke klasifikaci objektů. Pro charakterizaci objektů používáme momenty nižších řádů. Při použití momentů vyšších řádů dochází ke zkreslení, protože jsou velmi citlivé na šum.

Další metody

Uvedené možnosti popisu rozpoznávaného objektu samozřejmě nejsou jediné. Za zmínku jistě stojí popis pomocí jasů, B-spline křivek, či aktivních kontur. Více viz [1], [2], [3].

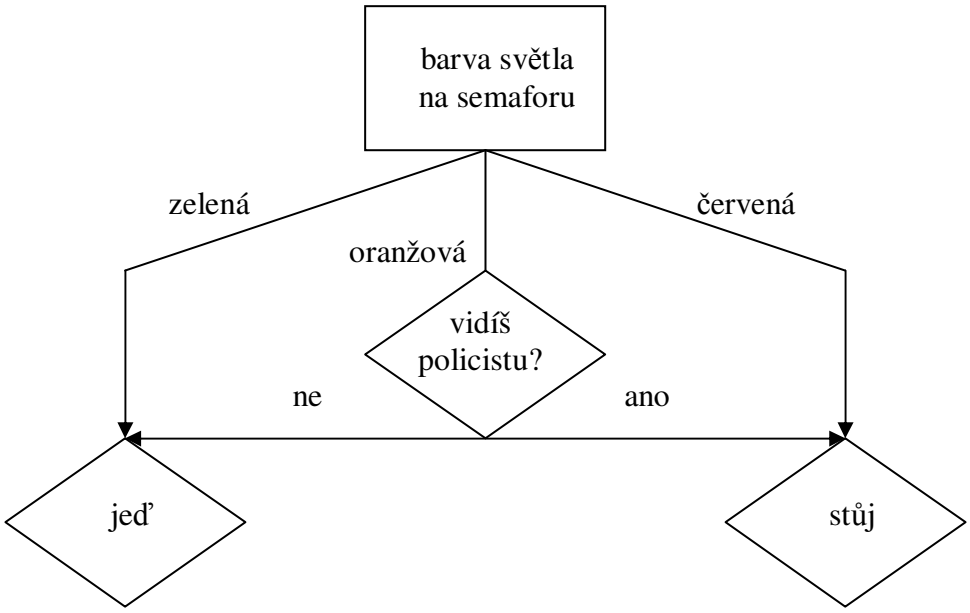
2.6. Klasifikace

Poté, co se nám podařilo oddělit objekty zájmu od pozadí, lokalizovat je a popsat pomocí jejich vhodných elementárních vlastností, nutně dospějeme do fáze, kdy dané objekty potřebujeme rozpoznat, neboli klasifikovat [12], [13], [14]. Klasifikace je proces, při kterém zařazujeme objekty na základě určitého rozhodovacího pravidla do (většinou) předem známých tříd. Objekty které jsou obsažené v jedné třídě jsou si mnohem podobnější než mezi jednotlivými třídami navzájem.

Klasifikátory dělíme do dvou základních skupin – příznakové a strukturální. U příznakových rozpoznáváme pomocí příznaků, což je skupina číselných charakteristik kterými je rozpoznávaný objekt popsán. Strukturní rozpoznávání je založeno na kvalitativním popisu rozpoznávaného objektu. Vlastní rozpoznávání je poté založeno na syntaktické analýze nadefinované gramatiky.

Rozhodovací stromy

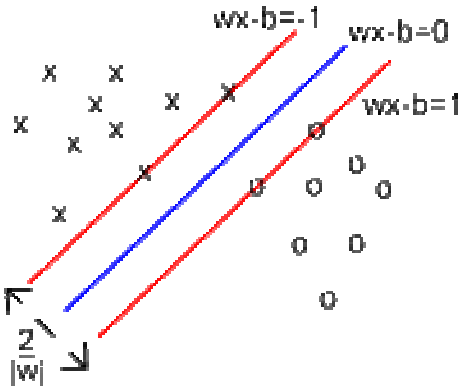
Rozhodovací stromy (Decision trees) [16] patří mezi nejjednodušší a nejoblíbenější klasifikátory. Jejich největší výhoda je v rychlosti algoritmu (strom, který procházíme pomocí booleovské logiky), dále v jejich přehlednosti, jednoduchosti a snadné rozšiřitelnosti. Rozhodovací pravidla jsou učena na základě trénovaní množiny, která zahrnuje informace o výsledné třídě. Jedná se tedy o učení s učitelem. Při vytváření rozhodovacích stromů můžeme využít algoritmů ID3 a C4.5.



Obr. 38. Příklad rozhodovacího stromu: běžné chování na křižovatce

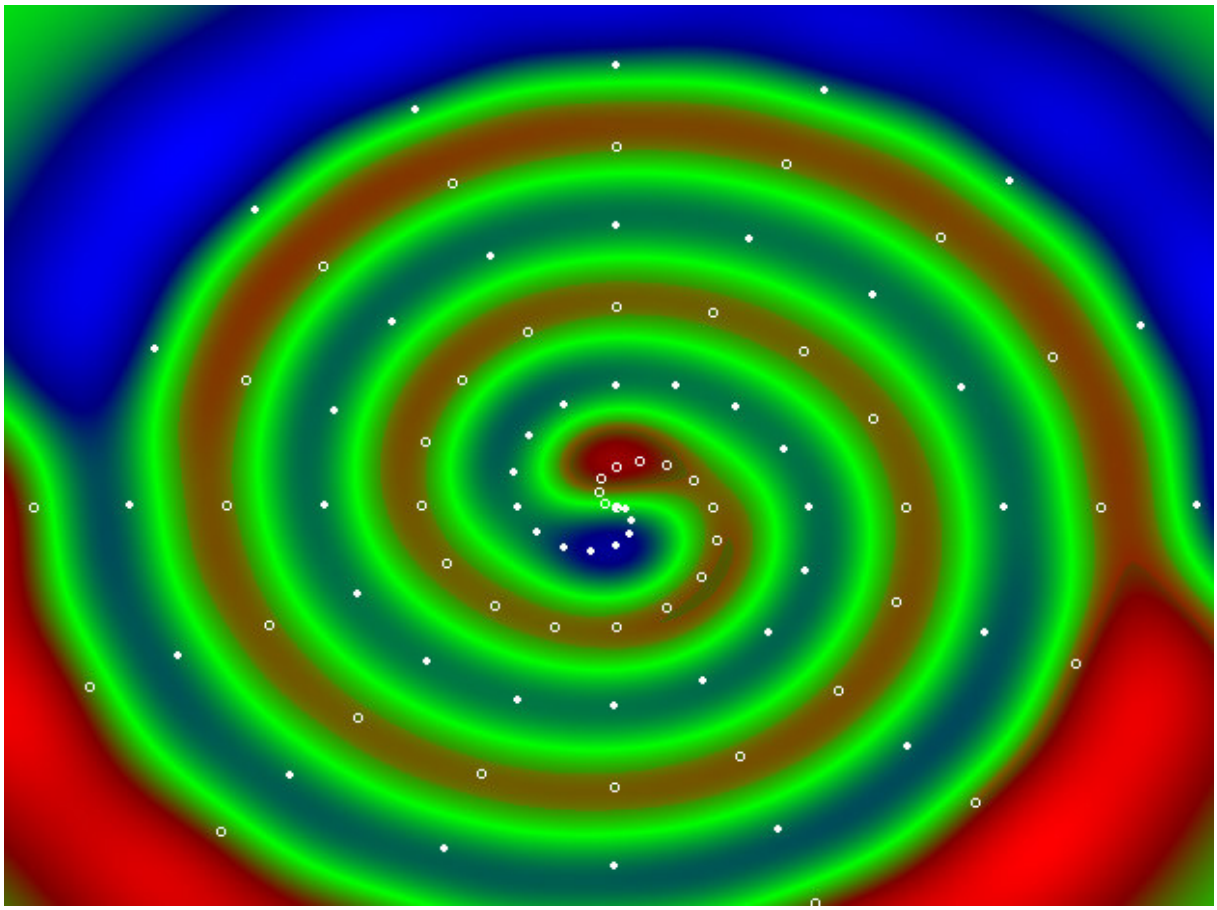
Support vector machines (SVM)

Support vector machines jsou klasifikátory [15], které jsou určeny především pro rozdělení (klasifikaci) dvou tříd. Při klasifikaci pomocí SVM od sebe oddělujeme body, které se nachází ve vícerozměrném prostoru $\mathbb{R}^n$ pomocí dělících nadrovin, které jsou n-1 dimensionální. Těchto dělících nadrovin existuje samozřejmě celá řada, proto se snažíme nalézt tu, jejíž vzdálenost od hraničních bodů jednotlivých tříd, kterými prochází pomocné vektory (support vectors) je maximální. Původně byl SVM klasifikátor pouze lineární, ale dnes jsou velmi oblíbená jeho rozšíření nelineárními funkcemi. Nejčastěji používanými funkcemi jsou polynomiální, sigmoidní, nebo Gaussova.



Obr. 39. Support vectors

SVM jsou ve své podstatě velmi podobné umělým neuronovým sítím. Hlavními výhodami SVM oproti neuronovým sítím je fakt, že při použití SVM nám postačí spočítat jedno globální minimum strukturního rizika, zatímco u neuronových sítí počítáme empirické riziko, které se ovšem nachází (zpravidla) v několika lokálních minimech. SVM je dobře využitelný u vícerozměrných prostorů.

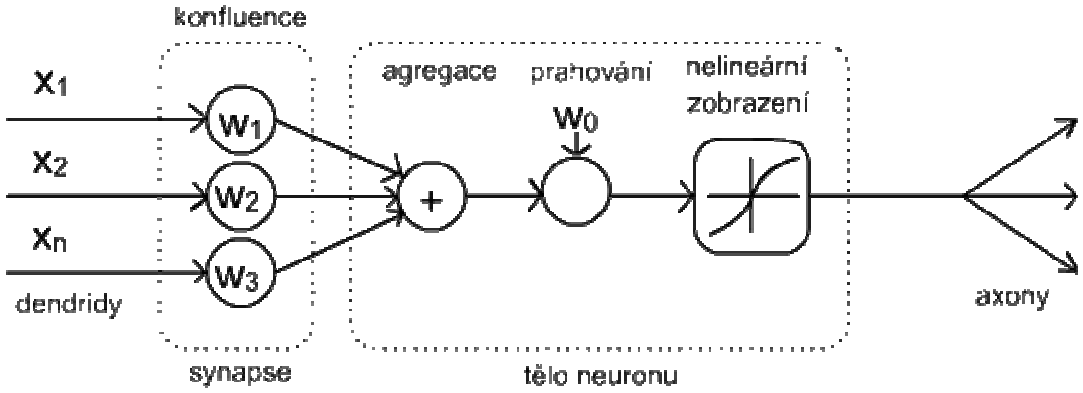


Obr. 40. Klasifikace pomocí SVM, KMOD kernel

Umělé neuronové sítě

Častým problémem při klasifikaci do tříd je mírná odlišnost od rozpoznávaného vzoru. V těchto případech klasifikace pomocí rozhodovacích stromů zpravidla selže a algoritmus klasifikace musíme předělat tak, aby byl schopen reagovat. V případě, že se ale rozhodneme použít umělou neuronovou síť, která má vhodnou topologii (strukturu) a je dobře natrénovaná, zareaguje (zobecňuje předkládaná data) a s určitou pravděpodobností zařadí rozpoznávaný obrazec do určité třídy. Neuronové sítě lze využít například při predikci chování, optimalizaci, nebo klasifikaci.

Umělé neuronové sítě (artificial neural networks) se snaží napodobit chování biologických neuronových sítí jak svou strukturou, tak svým chováním. Základním elementem neuronových sítí je nervová buňka neboli umělý neuron.



Obr. 41. Schéma umělého neuronu

Každý neuron obsahuje určitý počet vstupů, které jsou na stupu ohodnoceny vahami (bezrozměrná čísla). Poté se provede agregace těchto vstupů a jejich porovnání s prahovou hodnotou neuronu (vstupní potenciál) a následně se spočítá tzv. aktivační funkce (nejčastěji se používá sigmoidní funkce a hyperbolický tangens). Výstup této funkce je i výstupem samotného neuronu. V případě, že se jednalo o neuron ve skryté vrstvě, výstup bude použit jako vstup do další vrstvy. V případě, že se jednalo o neuron ve výstupní vrstvě, bude tento výstup výstupem celé umělé neuronové sítě.

Existuje celá řada umělých neuronových sítí. Nejjednodušší je perceptron, což je jednovrstvá síť. Dalšími neuronovými sítěmi jsou například Hopfieldova nebo Kohenenova síť (rekurentní sítě). Mezi zástupce vícevrstevných sítí (vstupní vrstva, jedna nebo více skrytých vrstev a výstupní vrstva) patří například ART síť. Při rozpoznávání obrazců má velké využití dopředná neuronová síť s učícím algoritmem back propagation.

3. Řešení ukázkové úlohy

Při řešení vzorové úlohy narážíme na několik problémů. Jedná se o fotografii pořízenou digitálním fotoaparátem, takže se nemusíme zabývat filtrování Salt and Pepper šumu, který vzniká při použití kamery, ale na první pohled vidíme, že se použití řady metod předzpracování nevyhneme.

Obr. 42. Vstupní obraz

Nejdříve převedeme barevný obrázek do stupňů šedi. Protože je ale lidské oko různě citlivé na různé barvy, nebudeme výslednou barvu počítat pouhým průměrem, ale použijeme následující empirický vztah

$$p(x, y) = 0.3\mathrm{R} + 0.59\mathrm{G} + 0.11\mathrm{B} \tag{49}$$

Obr. 43. Vstupní obraz převeden do stupňů šedi

V dalším kroku je třeba provést ekvalizaci histogramu, protože vstupní obraz je málo kontrastní.

Obr. 44. Obraz po ekvalizaci histogramu

Obraz, který jsme získali po aplikaci předchozích dvou kroků je však velmi podrobný. V případě, že bychom se pokusili již nyní rozpoznat hrany obrazu a klasifikovat jeho objekty, bylo by to velmi náročné jak na rychlost, tak správnost klasifikace. Proto se potřebujeme na obraz „podívat z větší vzdálenosti“. K tomuto kroku můžeme využít matematickou morfologii. Po aplikaci otevření a uzavření nám většina nepodstatných detailů z obrazu vymizí.

Obr. 45. Provedeno morfologické otevření a uzavření

Nyní již můžeme použít některý z hranových detektorů k detekci hranic objektů. Při řešení této ukázkové úlohy se mi nejvíce osvědčil Sobelův hranový detektor.

Obr. 46. Detekce hran Sobelovým filtrem

V dalším kroku provedeme prahování, pomocí kterého odstraníme další nepodstatné hrany. V ukázkové úloze používám prahování s pevně nastaveným prahem na hodnotu 128.

Obr. 47. Prahování, T_1=128

Nyní obraz invertujeme. To znamená, že v tomto případě (binární obraz) pouze zaměníme bílou a černou barvu. Jinak se invertovaná barva vypočte jako

$$n(x, y) = 255 - p(x, y) \tag{50}$$

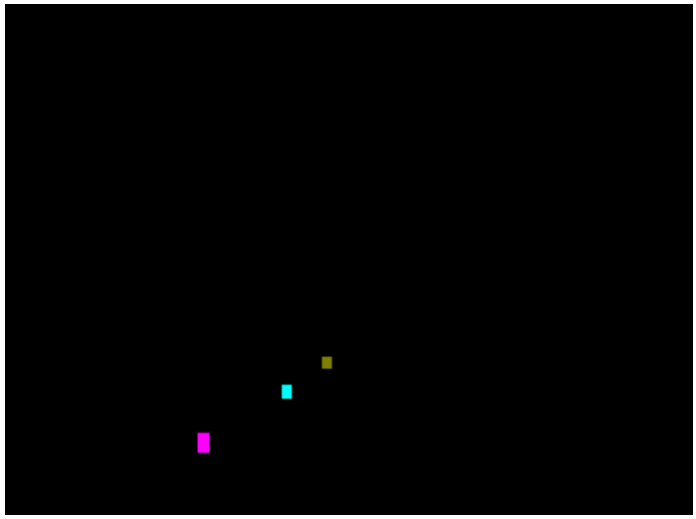
Obr. 48. Invertovaný obraz

Dále je třeba označit jednotlivé objekty. K tomuto používám segmentační metodu narůstání regionů.

Obr. 49. Obohacení obrazu o pseudobarvy

Takto označené objekty je třeba co nejlépe popsat. K popisu bychom mohli využít řady primitiv, jako jsou pravoúhlost, podlouhlost nebo směr, avšak plně nám postačí využít statistických momentů.

V posledním kroku je třeba popsané objekty klasifikovat. Protože se jedná o poměrně jednoznačně definované objekty a nepředpokládá se, že se budou nějak výrazně od sebe odlišovat, rozhodl jsem se pro klasifikaci použít rozhodovací strom.



Obr. 50. Detekované objekty

Nyní již zbývá „pouze“ postavit autíčko a nechat ho jezdit podle takto identifikovaných objektů.

4. Praktická aplikace

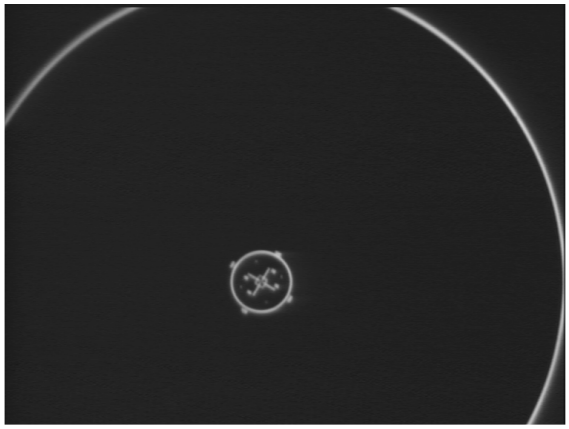
Každý čtenář této práce, který měl jen trochu možnost poznat úskalí technické praxe, jistě rozpoznal nadsázku v poslední větě minulé kapitoly. Při zjišťování možností realizace autíčka s autonomním řízením autor narazil na tým zajímavých lidí, kteří jej upozornili na rozsah rutinních prací nezbytných pro realizaci, zároveň však projevili zájem o pomoc s řešením zajímavého technického problému metodami digitálního zpracování obrazu.

Jedná se o kalibrace přístrojů, které slouží pro výzkum v oblasti LIVE SCIENCE (živé přírody). Tyto přístroje se skládají z rychlého přesného manipulátoru, mikropipetovací jednotky a dalších podpůrných uzlů. Pro kalibrace podélného posuvu (X) se samozřejmě používá laserový interferometr, který pracuje s přesností lepší než 1 mikrometr. Příčnou osu (Y) však bohužel takto kalibrovat nelze. Běžný odražeč se tam prostě nedá umístit tak, aby umožňoval proměření osy Y nebo dokonce měření pravoúhlosti.

Výrobce přístrojů si proto pomohl malým mikroskopem s kamerkou, který umístil do držáku místo pipetovací jednotky a zaměřuje značky, které byly s přesností lepší než 1 mikron zhotoveny pomocí elektronového litografu na kalibračním skle.

Obsluha najede na požadovanou souřadnici a pak navolí takovou korekci, až vidí hledanou značku na požadovaném místě monitoru. Tato ruční metoda je zdlouhavá, pracná a dosažitelná přesnost je 5 mikronů.

Na vstupu této již praktické úlohy máme obraz ve stupních šedi, který je mírně zašuměn. Cílem je lokalizovat střed křížku, který je zároveň i středem kružnice.



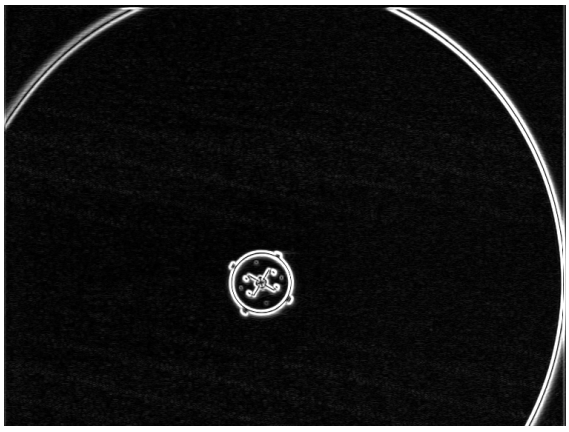
Obr. 51. Vstupní obraz

Oproti předchozí ukázkové úloze se zde klade mnohem větší důraz na přesnost lokalizace rozpoznaných objektů a proto si nemůžeme dovolit příliš modifikovat zpracovávaný obraz.

Vstupní obrazy jsou již ve stupních šedi, tudíž je nemusíme převádět z barevného spektra. Dalším zjednodušením oproti ukázkové aplikaci je fakt, že obrazy jsou snímané za konstantních světelných podmínek, takže nedochází k podexponování, přeexponování expozice a nemusíme proto provádět jasové korekce. Jak již bylo poznamenáno výše, vstupní obrazy jsou mírně zašuměné, protože jsou snímané automaticky z kamery bez manuálního

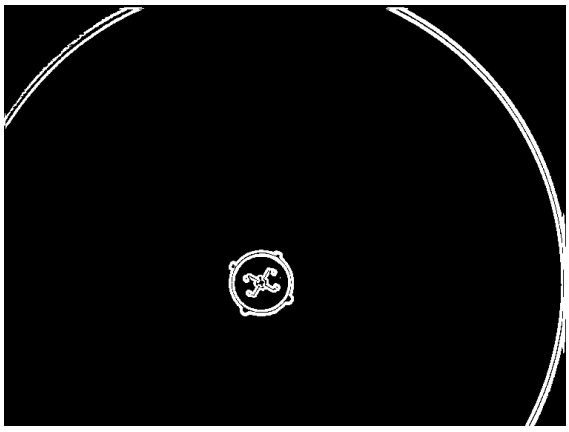
ostření. Protože v této úloze velmi záleží na přesnosti lokalizace rozpoznaných objektů a protože vstupní obraz není „zahlcen“ nepodstatnými detaily, nebudeme morfologii v této aplikaci uplatňovat.

Přistoupíme tedy rovnou k rozpoznávání hran. Na základě řady pokusů opět zvolíme Sobelův hranový detektor. Ostatní detektory buď vytváří nespojité hrany, což zbytečně ztěžuje klasifikaci, nebo velmi reagují i na šum v obraze a zkreslují rozpoznávanou hranu. Stejně tak nemůžeme použít Cannyho hranový detektor, protože v jeho první části je obraz vyhlazován pomocí Gaussova filtru, což přináší nebezpečí mírného posunutí hrany.



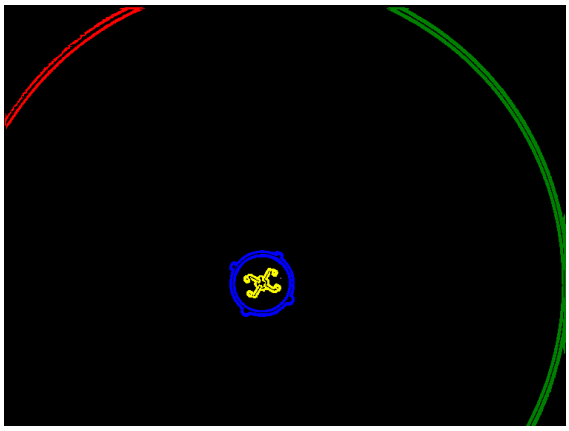
Obr. 52. Detekce hran Sobelovým filtrem

Nyní je potřeba použít prahování. Opět použijeme pevně nastavený prah a opět jej nastavíme na hodnotu 128. V případě, že by se nám rozpoznávaný objekt hodně „trhal“, můžeme použít prah trochu nižší (například 100), ale klasifikace bude pravděpodobně opět méně přesná.



Obr. 53. Prahování, T_1=128

Dalším krokem je označení objektů pomocí narůstání regionů, pro popis objektů opět použijeme RST invariantní momenty (zaměřovaná značka se může různě natáčet, měnit polohu).



Obr. 54. Segmentace obrazu

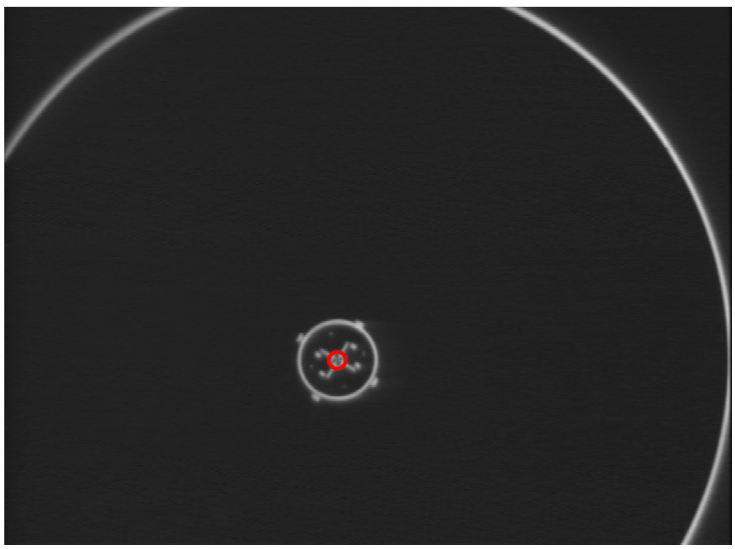
Při klasifikaci se snažíme poznat křížek uvnitř kružnice a spočítat jeho střed. V některých případech (velmi rozostřený obraz) se nám to však nemusí podařit a proto se ještě alternativně snažíme rozpoznat kružnici a z ní určit zaměřovaný střed. Použití kružnice k určení pozice zaměřovaného středu však může být méně přesné.

Na trénovacím obrázku si předpočítáme hodnoty RST invariantních momentů rozpoznávaného objektu. Poté se pomocí nich snažíme rozpoznat objekt zájmu v obraze nalézt jeho střed.

$$\varphi_1 = 12,315$$
$$\varphi_2 = 7,825 \tag{51}$$

Následným zpracováním vstupního obrázku (rozměrech 758x568 pixelů) jsme lokalizovali střed zaměřovaného křížku na souřadnicích

$$C = [347,375] \tag{52}$$



Obr. 55. Lokalizace zaměřovaného středu

5. Závěr

Práce shrnuje dosavadní zkušenosti autora s problematikou digitálního zpracování obrazu. První část je věnována teoretickým základům jasových korekcí, morfologie, hledání hran, segmentace, popisu obrazu a klasifikace. Jednotlivé metody jsou předvedeny na názorných příkladech.

Další kapitola se zabývá řešením ukázkové úlohy. Na vstupní obrázek byly aplikovány jasové a morfologické aplikace, binarizace, filtr pro hledání hran, segmentační metoda Region growing, následné popsání objektů a jejich klasifikace. Výsledkem bylo nalezení požadovaných objektů v zadaném obraze.

Poté následuje podrobný popis využití těchto metod pro řešení úlohy z technické praxe. Jejím cílem byla lokalizace značky používané při kalibracích přístrojů pro výzkum živé přírody.

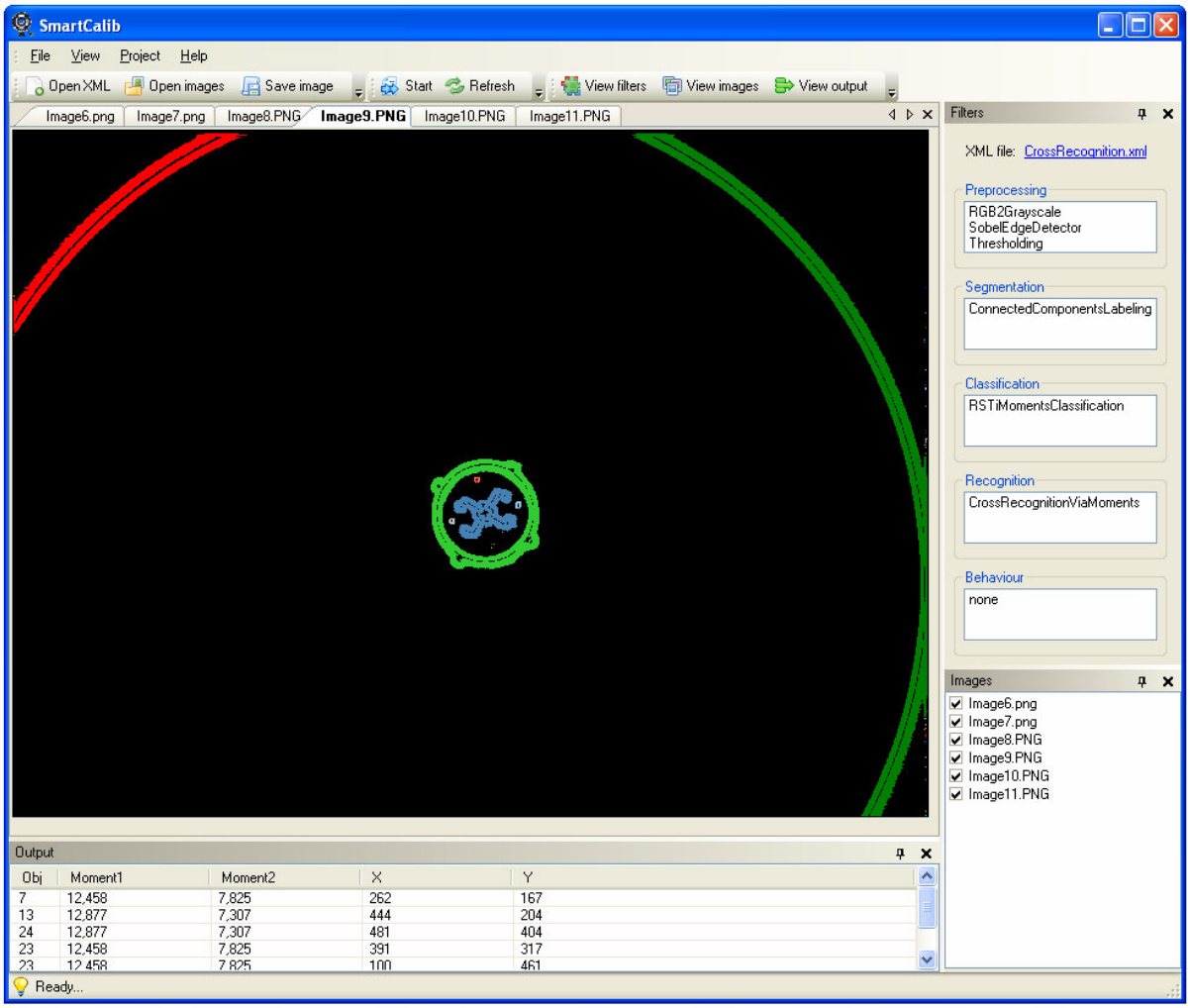
Původní vizuální vyhodnocení kalibračního obrazce bylo pracné, časově náročné a nepříliš přesné. Nalezení značky pomocí metod digitálního zpracování obrazu umožňuje zrychlení a zpřesnění kalibračního procesu.

Digitální zpracování obrazu je nesmírně zajímavý obor, který díky rozvoji svého matematického aparátu i technických prostředků bude stále více pronikat do běžného života.

Příloha A – Program SmartCalib

K řešení praktické aplikace popsané v kapitole 4 autor vytvořil program SmartCalib v jazyce c#. Tento program však není omezen pouze na rozpoznávání středu kalibrační značky, ale lze jej použít i k experimentování s metodami digitálního zpracování obrazu popsanými v kapitole 2 a dalšími.

Celý program byl navrhnut jako modulově řešená aplikace pomocí pluginů. To znamená, že uživatel není nijak omezen v množství funkcí, protože si může další filtry naprogramovat (za předpokladu, že zná metody digitálního zpracování obrazu), pouze jeho třídy musí dědit příslušnou abstraktní třídu podle typu pluginu (předzpracování, segmentace, klasifikace, rozpoznávání, chování). Průběh samotného rozpoznávání je pak nastaven v konfiguračním XML souboru.



Obr. 56. Program SmartCalib

Seznam literatury

[1] Šonka M., Hlaváč V., Boyle R.: Image Processing, Analysis, and Machine Vision, Chapman&Hall, 1993

[2] Gonzalez R. C., Woods R. E.: Digital Image Processing, Addison-Wesley, 1992

[3] Russ J. C.: The Image Processing Handbook, CRC Press, 2002

[4] Pratt W. K.: Digital Image Processing, John Wiley & Sons, 1978

[5] Goutsias J., Vincent L., Bloomberg D. S.: Mathematical Morphology and Its Applications to Image and Signal Processing, Kluwer Academic Publisher, 2000

[6] Serra J.: Image Analysis and Mathematical Morphology, Academic Press, 1982

[7] Serra J.: Image Analysis and Mathematical Morphology, Volume 2: Theoretical Advances, Academic Press, 1988

[8] Marr D., Hildrith E.: Theory of Edge Detection, Proceedings of the Royal Society of London, 1980

[9] Canny J.: A Computational Approach to Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 1986

[10] Leavers V.F.: Shape Detection in Computer Vision Using the Hough Transform, Springer-Verlag, 1992

[11] Ramirez R.W.: The Fft, Fundamentals and Concepts, Prentice Hall, 1984

[12] Duda R. O., Hart P. E., Stork D. G.: Pattern Classification 2nd ed., John Wiley & Sons, 2001

[13] Schlesinger M. I., Hlaváč V.: Ten lectures on statistical and syntactic pattern recognition, Kluwer Academic Publisher, 2002

[14] Fu, K.S.: Digital Pattern Recognition, Springer-Verlag, 1976

[15] Taylor J. S., Cristianini N.: Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000

[16] Witten I. H., Frank E.: Data Mining: Practical Machine Learning Tools and Techniques 2nd ed., Morgan Kaufmann, 2005

[17] Bishop C. M.: Neural Networks for Pattern Recognition, Oxford University Press 1996

[18] Moravec H. P.: Towards Automatic Visual Obstacle Avoidance, International Joint Conference on Artificial Intelligence, 1977

[19] Harris C., Stephens M.: A combined corner and edge detector, Proceedings of the 4th Alvey Vision Conference, 1998

[20] Smith S. M., Brady J. M.: SUSAN - a new approach to low level image processing. Int. Journal of Computer Vision, 1997

[21] Lucas B. D., Kanade T.: An Iterative Image Registration Technique with an Application to Stereo Vision, International Joint Conference on Artifcial Intelligence, 1981

[22] Tomasi C., Kanade T.: Detection and Tracking of Point Features, Carnegie Mellon University Technical Report, 1991