TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

BAKALÁŘSKÁ PRÁCE

Liberec 2009 **Ondřej Kolmistr**

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: B 2612 – Elektrotechnika a informatika

Studijní obor: 1802R022 – Informatika a logistika

Klasifikace objektů rozpoznáváním obrazu

Classification of object by image recognition

Bakalářská práce

Autor: **Ondřej Kolmistr**

Vedoucí práce: Ing. Přemysl Svoboda

Konzultant: Ing. Martin Vlasák

V Liberci 27.5. 2009

(zde bude vloženo originální zadání)

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/200 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé BP a prohlašuji, že **souhlasím** s případným užitím své bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užití své bakalářské práce či poskytnutí licence k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum:

Podpis:

Poděkování

Na tomto místě bych rád poděkoval vedoucímu mé bakalářské práce Ing. Přemyslu Svobodovi za jeho cenné rady a trpělivou pomoc.

Děkuji tímto také svým rodičům za jejich podporu během mého studia na Technické univerzitě v Liberci.

Abstrakt

Obsahem této bakalářské práce je popis algoritmů, které se používají pro zpracování obrazu a návrh aplikace pro online klasifikaci objektů snímaných kamerou. Jako vývojové prostředí bylo použito Delphi 7. Důvodem jeho použití byla znalost tohoto prostředí z výuky a dále fakt, že některé algoritmy pro zpracování obrazu byly již vedoucím této bakalářské práce v tomto prostředí naprogramovány. Výsledná aplikace filtruje obraz snímaný z kamery, klasifikuje objekty na základě jejich obrysu nebo délky obrysu a porovnává je s referenčními objekty s databáze.

Klíčová slova: zpracování obrazu, segmentace, klasifikace objektů

Abstract

The content of this work is the description of algorithms, which are used for image processing, and design of online application for the classification of objects captured by camera. As the development environment was used Delphi 7. The reason for its use was knowledge of this environment from the teaching and the fact that some algorithms for image processing have been programmed by leader of this work in this environment. The resulting application filtered image from the camera, classify objects based on their outline or outline length, and compares it with the reference objects with the database.

Keywords: image processing, segmentation, clasification of objects

Obsah

Seznam ilustrací

Seznam tabulek

1. Úvod

Dnes, kdy jsou digitální kamery a počítače s dostatečným výpočetním výkonem již relativně velmi levné, se můžeme s programy pro rozpoznávání obrazu setkat dosti často. Mezi typické příklady jejich použití by se dala považovat například kontrola předmětů na výrobní lince a jejich třídění. Taková aplikace poskytuje téměř ideální podmínky pro snímání obrazu a klasifikaci objektů. Mezi podmínky, které tuto úlohu zjednodušují by se dalo zahrnout dostatečné osvětlení prostoru a to tak, aby jednotlivé objekty byly co nejvíce kontrastní vzhledem k okolí. Další podmínkou je například konstantní vzdálenost kamery od jednotlivých objektů. V tomto případě můžeme jednoduše určit velikost objektu. Návrh aplikace pro podobné zjednodušující podmínky se stal cílem této bakalářské práce.

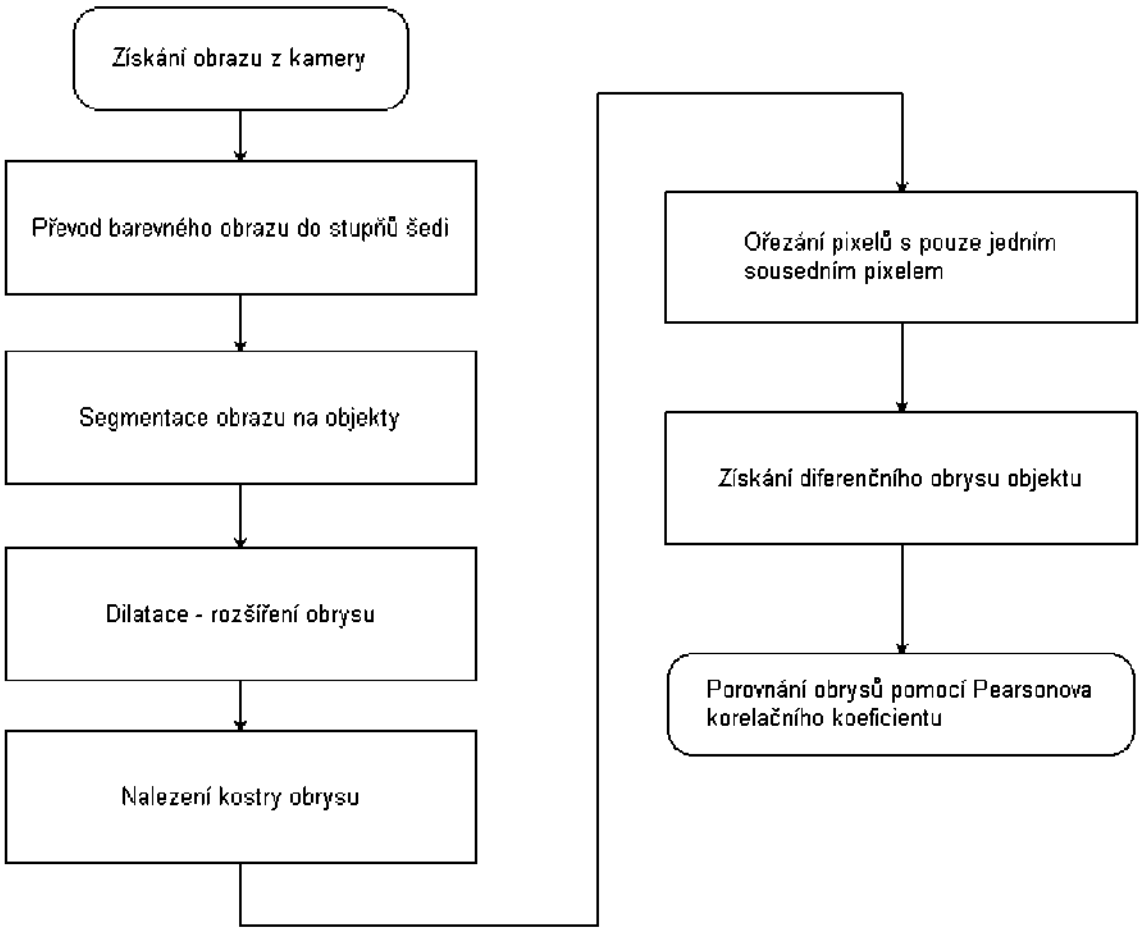
2. Počítačové vidění

Počítačové vidění je technický obor, zabývající se napodobením funkcí lidského vidění pomocí umělých prostředků. Vizuální informace jsou pro člověka velmi podstatné a lze z nich získat mnoho informací o jeho okolí, a různých objektech v něm. Počítačové vidění je velmi komplexní disciplína, která se v současné době neustále rozvíjí. I přesto je interpretace obrazu reálné scény velmi obtížná.

Činnost počítačového vidění je velmi složitá. Dala by se v podstatě rozdělit do dvou úrovní. V nižší úrovni jsou algoritmy, jejichž účelem je analyzovat vstupní obrazová data. Tato data jsou v podobě dvourozměrného pole. Z obrazových dat musí algoritmy nižší úrovně získat informace pro algoritmy vyšší úrovně. V nižší úrovní probíhá například potlačení šumu, segmentace obrazu na objekty a podobně. Tyto algoritmy se často nazývají souhrnně jako zpracování obrazu (*image processing*). Funkcí algoritmů ve vyšší úrovni je interpretace získaných dat. K tomu se používají techniky umělé inteligence a různé náročné algoritmy. Tato úroveň se nazývá počítačové vidění (*computer vision*).

3. Princip činnosti aplikace pro zpracování obrazu a klasifikaci objektů

Následující vývojový diagram znázorňuje funkci programu pro zpracování obrazu a klasifikaci objektů. Je z něho zřejmá návaznost jednotlivých naprogramovaných filtrů zpracovávajících obraz na vstupní DShow filtr, zajišťující získání obrazu z kamery.



Obrázek 1 – Schéma aplikace pro zpracování obrazu

3.1 Získání obrazových dat

Činnost aplikace začíná získáním obrazových dat. Tato data se získávají z kamery připojené k počítači, případně je možno do programu načíst data ze souboru. Připojená kamera snímá optické veličiny, a převádí je na elektrický signál. Vstupní optickou veličinou je jas jednotlivých barevných složek. Firmware kamery zajišťuje připojení k počítači přes univerzální rozhraní USB. Program z kamery získá matici pixelů. Každý pixel obsahuje tři byty informací, představující 256 hodnot pro červenou, zelenou a modrou složku barevného prostoru.

3.2 Filtrování obrazových dat

Po získání dat je program za použití filtrů pro zpracování obrazu upraví. Nejprve provede převod barevného obrazu na jednotlivé stupně šedi. Tímto sice ztratíme informace o barvě objektů i pozadí obrazu, ale pro další zpracování tím dojde ke zjednodušení. Dále se na obraz ve stupních šedi aplikuje některý ze segmentačních algoritmů. Tyto algoritmy mají za úkol detekovat jednotlivé objekty v obraze. Objekty se ovšem musí lišit svým jasem oproti převládajícímu pozadí obrázku, nebo musí mít hrany, kde se jas mění. Po detekci se obrys objektu zvýrazní filtrem, který jednotlivé pixely obrysu rozšíří podle zadané masky. Takto upravený obrys objektu by již měl být spojitý. Poslední filtr provede nalezení kostry obrysu. Základní vlastností takovéto kostry je, že její šířka je jeden pixel. V posledním kroku filtrování se provede odstranění slepých výběžků ze skeletu. Výsledkem filtrování obrazových dat je tedy takový obraz, kde je objekt znázorněn pouze svým obrysem. Šířka toho obrysu je jeden pixel a musí být splněna spojitost celého obrysu.

3.3 Klasifikace objektů z filtrovaného obrazu

Obraz, který je již upraven předchozími filtry, poté procházíme algoritmem, který uloží směry vektorů mezi každými pátými po sobě jdoucími pixely. V tomto poli následně odečteme jednotlivé po sobě jdoucí hodnoty a získáme tak relativní změny úhlů v obrysu objektu. Pole změn těchto úhlů poté pomocí Pearsonova korelačního koeficientu porovnáváme s dalšími objekty a zjišťujeme jejich podobnost. Další možnost klasifikace je podle pouhé délky obrysu.

4. Delphi 7

Pro realizaci aplikace jsem se rozhodl použít vývojové prostředí Delphi 7. Toto rozhodnutí jsem učinil ze dvou důvodů. Jednak je to v podstatě jediné prostředí, se kterým jsem byl do konce druhého ročníku bakalářského studia seznámen, navíc vedoucí této bakalářské práce již některé filtry pro zpracování obrazu, včetně komunikace s kamerou v tomto prostředí, naprogramoval.

Delphi je grafické vývojové prostředí vyvíjené firmou Borland, později její dceřinou firmou CodeGear, které je určené pro vytváření aplikací na platformě Windows v objektové nástavbě jazyku Pascal, Objekt Pascalu. Delphi obsahuje systém RAD, který při vytváření grafického rozhraní automaticky generuje kostru zdrojového kódu, což vede k urychlení vývoje aplikace.

Při programování v Delphi se ve velké míře používají komponenty. To jsou malé balíčky funkcí, vykonávající určitou činnost. Například otevírání a ukládání souborů, zobrazování obrázků, komunikace s databázemi, zobrazování grafů a podobně. Výhodou je i možnost tvoření vlastních komponent přímo pro naši potřebu.

5. Práce s kamerou

Pro práci s kamerou se používá DirectShow (DShow), což je rozhraní pro programování aplikací (API) od společnosti Microsoft. DirectShow je součástí Microsoft DirectX. To je programátorská knihovna obsahující nástroje pro tvorbu například počítačových her a dalších multimediálních aplikací, vytvořená pro použití pod operačním systémem Windows.

V DShow je komplexní úkol, například přehrávání videa, rozdělen do posloupnosti jednodušších úkolů, známých jako filtry. Jednotlivé filtry mají vstupní a výstupní rozhraní pro připojení na další filtry. Pro realizaci složitějšího úkolu musí programátor vytvořit instance použitých filtrů a poté filtry propojit do určené posloupnost.

Filtry se dělí do tří skupin:

- Zdrojové – zajišťují tok zdrojových dat, například čtením ze souboru.
- Transformační – upravují data ze vstupních filtrů, provádějí například dekompresi.
- Výstupní – realizují výstupní operace, například vykreslují obraz na obrazovku nebo zapisují do souboru.

V našem programu je využíván zdrojový filtr zajišťující ovládání kamery. Po startu programu se do ComboBoxu nahraje seznam dostupných zařízení. Po výběru některého zařízení se v druhém ComboBoxu zobrazí seznam rozlišení, kterými vybrané zařízení disponuje. Dále je k dispozici ruční nastavení závěrky a jasu. U těchto parametrů je možné zvolit automatické nastavení, kdy si kamera zvolí vhodné hodnoty sama.

6. Popis algoritmů pro rozpoznávání obrazu

Tyto filtry pro zpracování obrazu již nejsou součástí rozhraní DShow a byly naprogramovány ručně v Delphi, na rozdíl od vstupního filtru ovládajícího kameru.

6.1 Převod do stupňů šedi

Tento filtr zajišťuje převod obrazových dat do stupňů šedi. Před vstupem do tohoto filtru jsou obrazová data uložena po jednotlivých barevných složkách. Složky jsou podle barevného modelu RGB červená, zelená a modrá. Každá z uvedených barevných složek má rozlišení 8 bitů. Je tedy definováno 256 odstínů každé barvy a každý pixel obsahuje 24 bitů.

Aditivním skládáním jednotlivých barev vzniká výsledná barva pixelu, jak je uvedeno v následující tabulce.

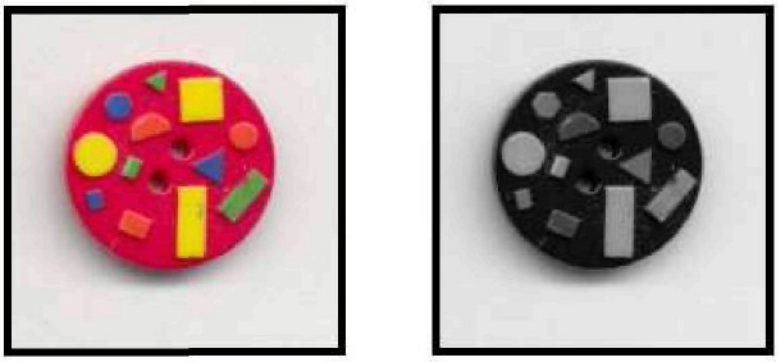
Tabulka 1 - Aditivní skládání barev v RGB prostoru

R	G	B	barva
0	0	0	černá
255	0	0	červená
0	255	0	zelená
0	0	255	modrá
255	255	0	žlutá
255	0	255	purpurová
0	255	255	azurová
255	255	255	černá

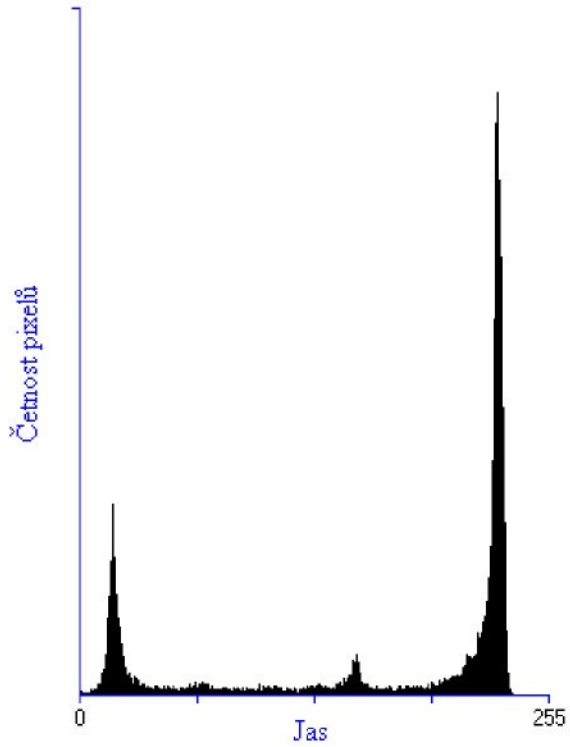
Převod z RGB prostoru do stupňů šedi ovšem není lineární. Provádí se pomocí vzorce:

$$Y = 0.3\ R + 0.59\ G + 0.11\ B \qquad (1)$$

kde Y je výsledný jas a R,G a B reprezentují jednotlivé barevné složky. Takto vypočítané hodnoty šedi můžeme pro názornost zobrazit v histogramu.



Obrázek 2 - Zdrojová data histogramu



Obrázek 3 - Histogram

Tento histogram znázorňuje výskyt jednotlivých stupňů šedi v obrázku. V levé části histogramu je vidět větší výskyt tmavších pixelů, které znázorňují objekt. V pravé části je výrazný počet světlých pixelů, představujících světlejší pozadí. S histogramem se můžeme setkat také například při fotografování, kde na něm lze vidět rozsah jasů scény.

6.2 Segmentace

6.2.1 Segmentace - úvod

Smyslem segmentace je oddělení hledaného objektu od pozadí. Je to jeden z nejdůležitějších kroků při zpracování obrazu. Segmentace komplexních scén je velmi složitá. Je nutno používat algoritmy vyšší úrovně, například sémantickou znalost segmentovaného obrazu. Obraz se segmentuje podle určitého kritéria stejnorodosti. V nejjednodušších algoritmech, které budou dále popsány, se jedná o segmentaci na základě analýzy jasu nebo barvy. Složitější algoritmy mohou segmentovat například na základě textury. V tom případě se musí analyzovat okolí daného bodu.

Je popsáno mnoho segmentačních algoritmů, žádný však není naprosto univerzální a použitelný pro všechny aplikace. Výběr vhodné metody je pro výsledek segmentace velmi důležitý.

Mezi dva základní jednoduché algoritmy pro segmentaci by se dala zařadit segmentace pomocí hledaní hran (Edge finding), a prahování (Thresholding). Metoda hledání hran hledá v obrázku hrany, kde se mění jas mezi pozadím a hledanými objekty. Pomocí této metody získáme obrys hledaného objektu. Druhou metodou je prahování, jejímž principem je určení prahu jasu mezi pozadím a hledanými objekty v histogramu a následné získání objektu z obrazu. Touto metodou získáme všechny pixely zahrnuté do objektu, ne jen jeho obrys.

6.2.2 Hrany

Hrany jsou místa v obraze, kde se prudce mění barva, v případě obrázku ve stupních šedi jas. V ideálním případě se tedy hrany nacházejí na pomezí odlišných objektů, mající různé barvy nebo jas. V praxi však hranové detektory mohou odhalit množství hran, které neodpovídají hranicím objektů. Tyto hrany mohou vznikat vlivem nestejnoměrného osvětlení,

stíny jiných objektů a podobně.

Při segmentaci obrazu reálné scény je toto velký problém. Při návrhu jednoduché aplikace však můžeme počítat s tím, že scéna zachycená na obraze bude obsahovat homogenní objekty, které budou osvětleny homogenním světlem.

6.2.3 Hledání hran

Hrany v obraze se obvykle vyskytují na místech, kde se jas prudce mění. Identifikace těchto prudkých změn spočívá v hledání lokálních maxim u první derivace signálu. U druhé derivace signálu se hledají průchody nulou. Při počítačovém zpracování signálu se využívá konvoluce signálu s určitou maskou, která aproximuje jednotlivé derivace.

Při vyhledávání hran, kdy využíváme aproximaci první derivace diferencí, využíváme pro konvoluční jádro například Robertsův operátor, operátor Prewittové nebo Sobelův operátor. Robertsův operátor využívá okolí 2 * 2 body a je citlivý i na jednotlivé body, které představují šum. Konvoluční jádro Robertsova operátoru je pro derivaci v řádku:

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2}$$

a pro derivaci ve sloupci:

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \tag{3}$$

Oproti Robertsově operátoru je maska operátoru Prewittové, pomocí jíž se gradient aproximuje rozměru 3 * 3. Sobelův hranový operátor je podobný jako operátor Pewittové,

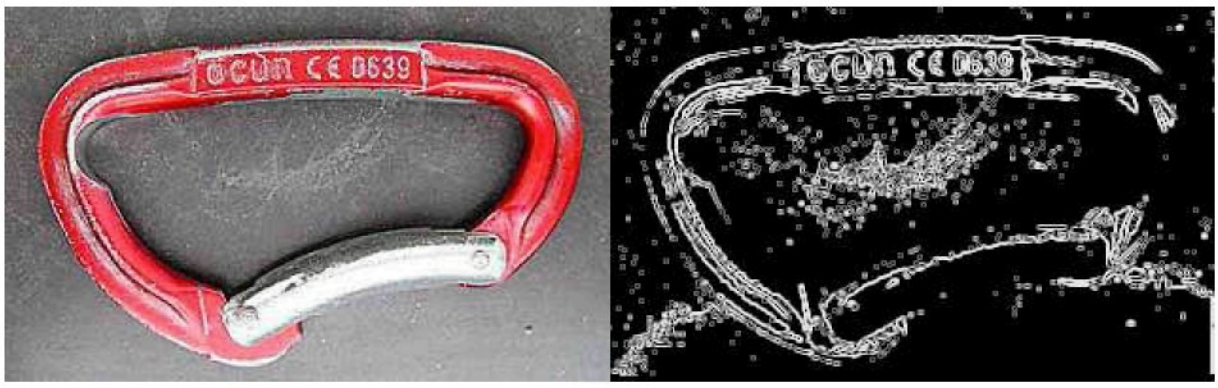
dává však větší váhu středu, čímž by mělo docházet k lepší detekci hran. Konvoluční jádro Sobelova operátoru vypadá pro derivaci v řádku následovně:

$$\frac{1}{4}\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \tag{4}$$

pro derivaci ve sloupci je pak Sobelův operátor:

$$\frac{1}{4}\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{5}$$

Dále existují metody, hledající průchody signálu druhé derivace nulou. Tyto metody, například Cannyho hranový operátor vykazují velmi dobré výsledky. Oproti tomu jednodušší metody založené na první derivaci nemusí vždy najít všechny hrany.



Obrázek 4 - Detekce hran Sobelovým operátorem

6.2.4 Prahování

Poněkud odlišnou metodou segmentace je prahování. Prahování je statistická metoda, jejímž základem je statistická analýza obrazových dat, nejčastěji hodnot jasu pixelů. Je to v podstatě jedna z nejstarších a nejjednodušších technik pro segmentaci obrazu. Při prahování určíme podle rozložení jednotlivých úrovní šedi v obraze hranici jasu, která bude odlišovat objekt od pozadí. Výsledkem prahování pak je obraz, kde bude hledaný objekt reprezentován například černou barvou a pozadí bílou barvou. Problémem je určení této hranice. K nalezení prahu existuje mnoho metod. Některé z nich jsou dále popsány.

Metoda pevného prahu

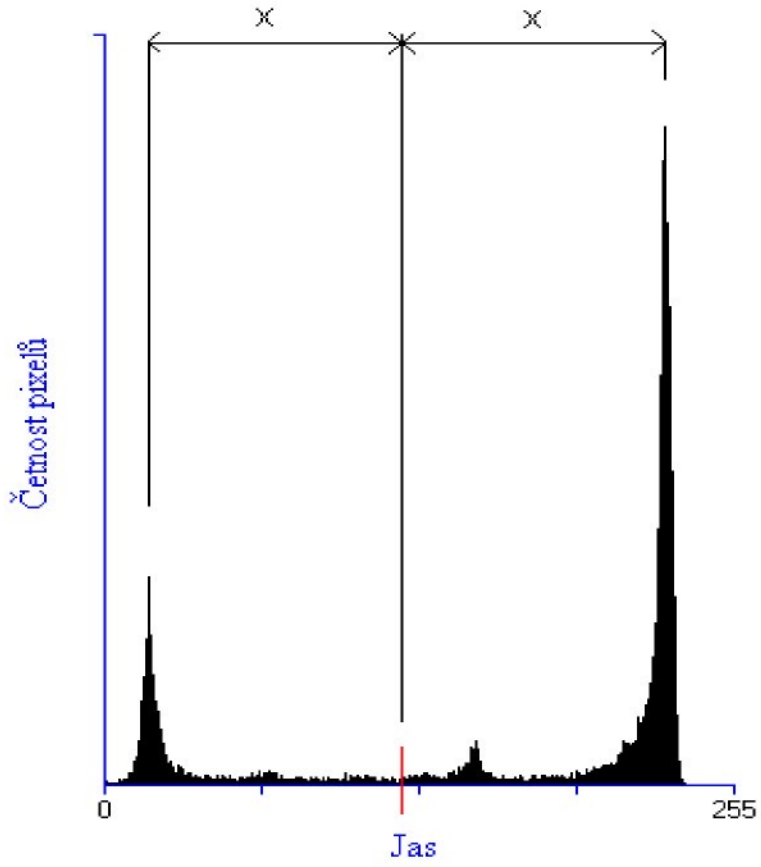
Nejjednodušší metodou je určení pevného prahu. Práh je nastaven trvale, nezávisle na obrazu, na který má být segmentace aplikována. Tato metoda by se dala použít v případě vyhodnocování obrazu, který by měl stále podobný jas pozadí i objektu. Algoritmus této metody je následující:

```
prah := 128;                          //pevné nastavení prahu
if A[x,y] > prah then A[x,y] := 255   //objekt
      else A[x,y] := 0;               //pozadí
```

Nalezení hranice mezi dvěma vrcholy

Určení prahu by se dalo provést v případě, že histogram obsahuje dva vrcholy, které představují hledaný objekt a pozadí. Z toho je zřejmé, že hodnota prahu se bude nacházet

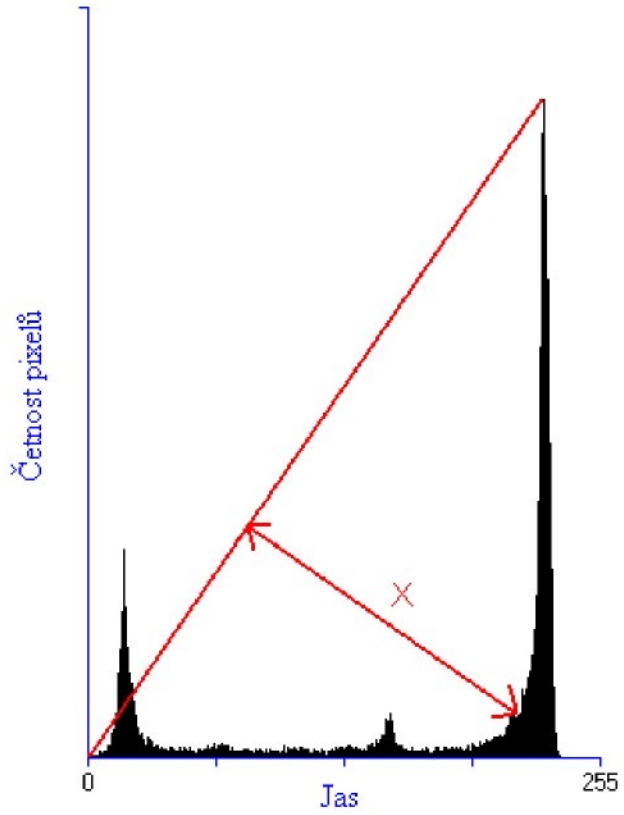
mezi těmito dvěma vrcholy.



Obrázek 5 – Nalezení prahu mezi dvěma vrcholy

Trojúhelníková metoda

Trojúhelníková metoda hledá práh také na základě histogramu. Principem metody je proložení úsečky mezi bodem na ose x, kde je poslední nulový počet pixelů odpovídající danému jasu a bodem znázorňujícím nejvyšší počet pixelů v celém histogramu. Poté se pro každou hodnotu na vodorovné ose pod touto přímkou spočítá její vzdálenost od přímky. V bodě, kde je vzdálenost od přímky největší, je práh. S takto určeným prahem se dále pracuje stejně jako v metodě s pevným prahem.

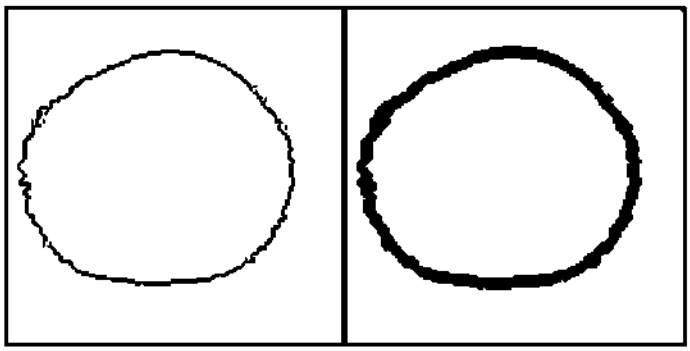


Obrázek 6 - Princip trojúhelníkové metody

6.3 Dilatace

Dilatace je proces, při kterém se hrany získané hranovou detekcí v obraze rozšiřují. To se provádí proto, aby se odstranily různé nespojitosti v hranách a aby došlo k zaplnění malých přerušení a zálivů. Tento proces se provádí tak, že se pomocí dilatační masky rozšíří všechny pixely, detekované jako hrany. Dilatační maska může mít různé rozměry i tvary. Tyto parametry pak určují vliv dilatace na obraz. V tomto konkrétním případě je dilatační maska zvolena rozměru 5 * 5 pixelů..

$$\begin{pmatrix} 0 & 0 & 255 & 0 & 0 \\ 0 & 255 & 255 & 255 & 0 \\ 255 & 255 & 255 & 255 & 255 \\ 0 & 255 & 255 & 255 & 0 \\ 0 & 0 & 255 & 0 & 0 \end{pmatrix} \quad (6)$$



Obrázek 7 - Dilatace obrysu

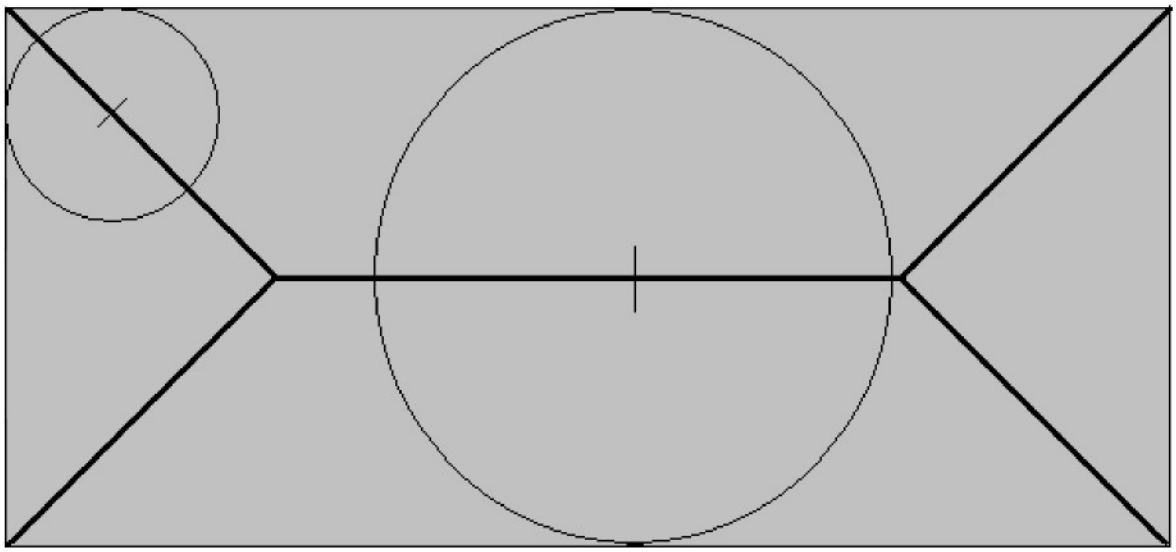
6.4 Skelet

Skelet, neboli kostra, představuje popis objektu nebo jeho obrysu pomocí křivek. Křivky tvořící skelet musí mít následující vlastnosti:

- jejich šířka je jeden pixel
- jsou umístěny uprostřed objektu
- reprezentují tvar objektu

Druhá podmínka pro nás není tak důležitá, jelikož dilatace obrysu, která proběhla v předchozím kroku není tak výrazná, aby obrys přestal reprezentovat tvar objektu.

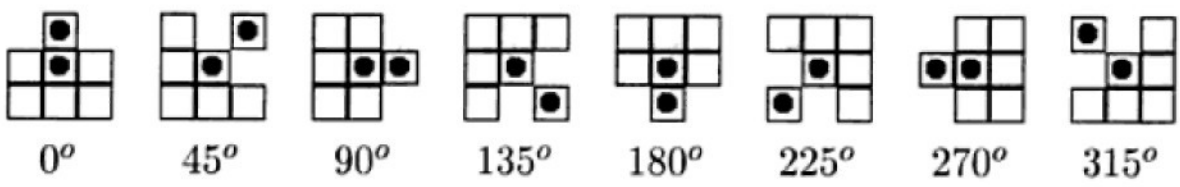
Při hledání skeletu se vychází z toho, že skelet objektu je sjednocení bodů, které odpovídají středům kružnic, jež jsou obsaženy v objektech a dotýkají se jeho hranice nejméně dvěma body.



Obrázek 8 - Princip tvorby skeletu

6.5 Pruning

Název tohoto filtru by se do češtiny dal přeložit jako prořezávání nebo odvětvování. To je v podstatě i slovní popis jeho funkce. Skelet, který je výstupem předchozího filtru, má totiž slepé výběžky a jejich odstranění je smyslem tohoto filtru. Odstraňování těchto slepých výběžků se provádí pomocí osmi šablon, které jsou na následujícím obrázku.

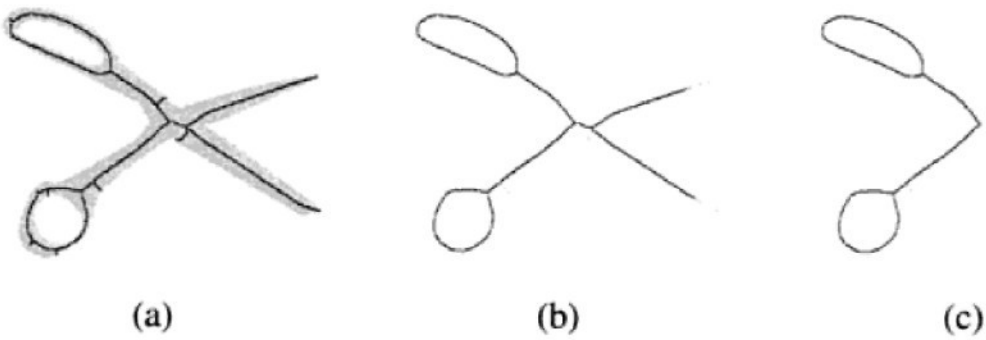


Obrázek 9 - Šablony pro pruning

Pomocí těchto šablon se detekují koncové body slepých výběžků, které jsou poté

odmazány.

Toto se děje v cyklech, přičemž nastavení jejich počtu může být poněkud obtížné. Pokud bychom použili malý počet cyklů, zůstanou na skeletu zbytky slepých výběžků, jejichž délka byla delší než počet cyklů filtru. Pokud bychom nastavili počet cyklů moc vysoký, ze skeletu zůstanou jen uzavřené oblasti, které již nebudou vypovídat o původním objektu. V našem případě není nastavení počtu cyklů příliš důležité, jelikož hledáme uzavřené obrysy. Na následujícím obrázku je zřejmá závislost výsledku prořezávání na nastavení počtu cyklů filtru.



Obrázek 10 - Prořezávání: (a) původní skelet, (b) menší počet cyklů, (c) větší počet cyklů

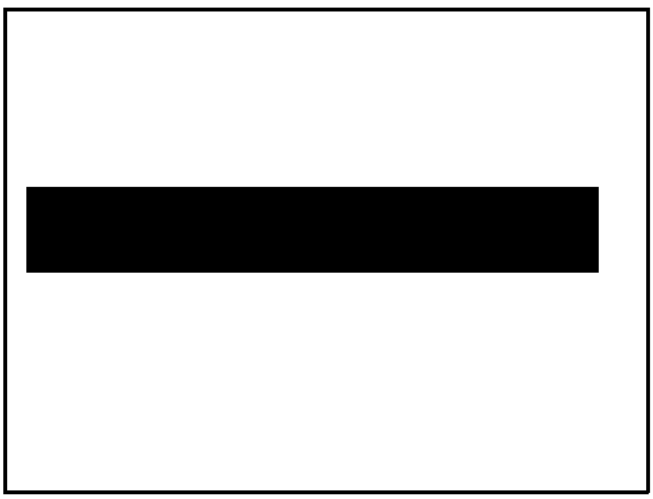
Na obrázku je vidět, že v případě (b) je nastaveni počtu cyklů optimální. Naopak v případě (c) je nastavení příliš vysoké a došlo ke ztrátě informace o původním tvaru objektu.

7. Vyhodnocování filtrovaného obrazu

Z původního obrazu po projití všemi výše uvedenými filtry zbude jen obrys, který reprezentuje objekt v původním obrazu. Tento obrys musí být uzavřený a každý pixel tvořící obrys musí mít pouze dva sousední pixely. Pokud jsou tyto podmínky splněny, můžeme objekt vhodným způsobem klasifikovat. Možností je několik. Například pomocí popisu obrysu, plochy objektu, nebo barvy objektu v případě, že bychom pracovali s jednotlivými složkami barevného obrazu zvlášť a neprováděli jako první úpravu převod do stupňů šedi.

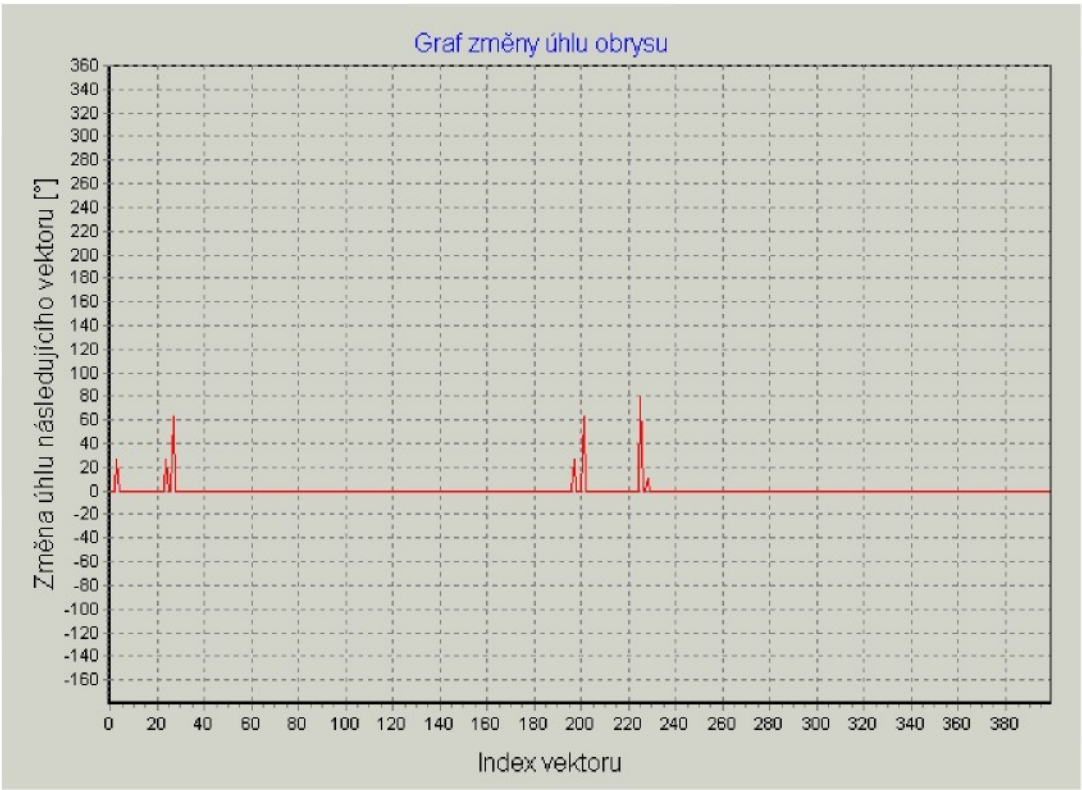
7.1 Klasifikace na základě tvaru obrysu

Pro klasifikaci obrysu jsem zvolil dvě různé metody. První je algoritmus, který projde obrys a spočítá odchylku úhlu mezi vektorem a referenčním směrem. Jednotlivé vektory jsou mezi každým pátým pixelem obrysu. Tyto úhly se od sebe postupně odečtou, čímž získáme diferenční popis obrysu. Z tohoto popisu jsou vidět změny směru obrysu.



Obrázek 11 - Příklad objektu

Obrázek 12 - Obrys objektu



Obrázek 13 - Graf změny úhlů obrysu

Tento signál je poté normován na jednotnou délku a je provedena konvoluce s maskou, která zajistí jeho mírné rozostření. Konvoluce je matematický vztah, popisující interakci signálu se systémem. V našem případě je signálem obrys objektu a systémem

trojúhelníková maska, která zajišťuje rozostření obrysu. V následujícím příkladě můžeme vidět výsledek konvoluce signálu *x* s maskou *m*. Výsledkem je pole *y*.

```
x = [0 5 0 0 5 5]
m =  [1 2 3 2 1]
y = conv(x,m)
y = [0 5 10 15 15 20 25 25 15 5]
```

Posledním krokem je spočítání Pearsonova korelačního koeficientu mezi objektem z kamery a objekty, se kterými vstupní obraz porovnáváme. Pro výpočet použijeme obrys již upravený konvolucí.

$$P = \frac{\Sigma(x_i - \bar{x}) * \Sigma(y_i - \bar{y})}{\sqrt{(x_i - \bar{x})} + \sqrt{(y_i - \bar{y})}} \qquad (7)$$

Pearsonův korelační koeficient je statistický test, počítající jak těsný nebo volný je vztah dvou proměnných. Pokud je koeficient do 0,2, jedná se o zanedbatelný vztah, mezi 0,2 a 0,4 je vztah nepříliš těsný, mezi 0,4 a 0,7 je středně těsný vztah, 0,7 až 0,9 je velmi těsný vztah a od 0,9 do 1 se jedná o extrémně těsný vztah.

Na základě tohoto testu tedy zjistíme, jakému objektu z databáze je snímaný objekt nejpodobnější a jeho název vypíšeme jako výsledek detekce.

7.2 Klasifikace na základě délky obrysu

Jako druhou metodu klasifikace objektů jsem zvolil klasifikaci podle délky obrysu. Délku obrysu zjistíme jednoduše tak, že v cyklu projdeme obrázek s obrysem a při detekci pixelu, jehož barva odpovídá obrysu, inkrementujeme počítadlo délky obrysu. Poté porovnáváme zjištěnou délku obrysu aktuálního objektu s délkou obrysů v databázi. Pokud je délka obrysu objektu snímaného kamerou větší než 80% a kratší než 120% délky objektu v databázi, pak považujeme objekt před kamerou za odpovídající objektu v databázi. V případě, že tato kritéria splní více objektů, program z databáze vybere ten s nejmenší odchylkou délky obrysu od délky obrysu snímaného objektu.

8. Práce s databází objektů

Pro ukládání objektů jsem využil složený datový typ record, tedy záznam. Tento datový typ může obsahovat více proměnných různých typů. Přístup k nim je přes tečkový operátor, kterým se oddělují jednotlivé proměnné obsažené v záznamu. Záznam v tomto případě obsahuje tři proměnné pro ukládání názvu objektu, délky obrysu a vlastního tvaru obrysu. Objekty se ukládají do souboru složeného z takto definovaných záznamů, jehož přípona je .obj.

Definice typu TobjectRec:

```
type TObjectRec = record
    name: string[30];
    OutlineLength: integer;
    Outline: TPole;
  end;
```

Po spuštění programu se automaticky otevře soubor „default.obj" v adresáři, kde je program spuštěn. V případě že soubor tohoto jména v daném adresář neexistuje, dojde k jeho vytvoření. Ukládání objektů do souboru probíhá pomocí tlačítka na záložce „Kamera", kde se do Editu zadá také jméno objektu.

Na záložce „Databáze" je k dispozici tlačítko vytvoření nového souboru, po jehož stisknutí se otevře SaveDialog s přednastavenou příponou souboru. Pomocí tohoto dialogu nastavíme jméno souboru a adresář, kam dojde k jeho uložení.

Pokud nechceme používat standartní soubor „default.obj" můžeme na záložce „Databáze" načíst jiný soubor. V OpenDialogu otevíraném na kliknutí na tlačítko je pomocí filtru nastaveno otevírání pouze souborů s příponou .obj. Po načtení souboru se provede nastavení velikosti pole záznamů TobjectRec na délku souboru a dojde k načtení dat o objektech ze souboru do tohoto pole.

Další možností je mazání posledního objektu z aktuálně otevřeného souboru. Tato operace se realizuje pomocí procedury Seek, která nastaví ukazatel na konec souboru a procedury Truncate, která odřízne poslední záznam v souboru. Poté se zavolá procedura, která vypíše seznam objektů v souboru do ListBoxu a procedura, která načte takto upravený soubor do pole záznamů.

Procedura pro zkracování databázového souboru:

```
procedure TfrmMain.Button20Click(Sender: TObject);
begin
if FileSize(soubor) >= 1 then begin
   Seek(soubor,FileSize(soubor)-1);
   Truncate(soubor);
   end;
VypisSouboru;
NacteniZeSouboru();
end;
```

9. Ovládání aplikace a testování

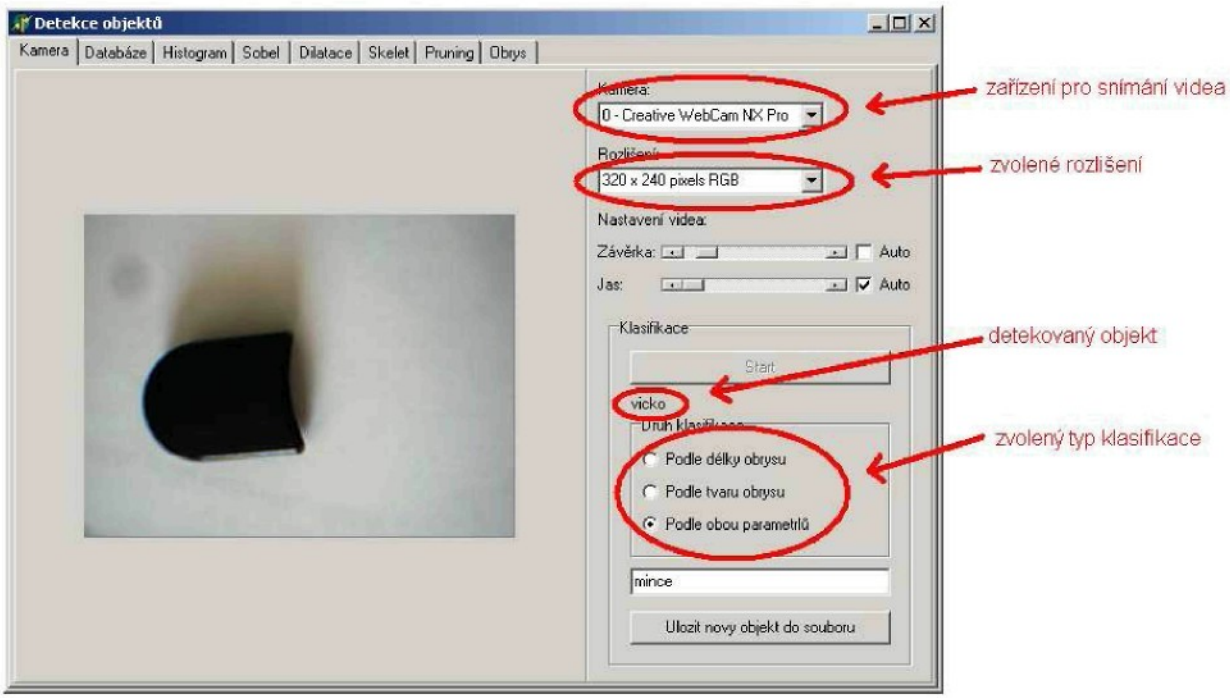
Po startu aplikace je nutno vybrat kameru ze seznamu připojených zařízení pro snímání obrazu. Dalším krokem je zvolení vhodného rozlišení, dostačující je 320*240 pixelů. Je nutné vybrat výstup z kamery ve formátu RGB. V jiném případě se jedná o komprimovaný formát, se kterým neumí aplikace pracovat. Po vybrání kamery se zpřístupní tlačítko „Start". Po kliknutí na něj začne probíhat detekce objektů z databáze v obraze.

Jako databázi můžeme využít standardně otevíraný soubor „default.obj" nebo nahrát některý, z již dříve vytvořených souborů. Pokud chceme do načteného souboru přidat aktuální objekt, vyplníme jeho název a klikneme na tlačítko „Ulož nový objekt do souboru". Tyto operace se provádějí na záložce „Databáze".

Pro porovnání objektů můžeme pomocí RadioButtonů vybrat tři způsoby. První metoda využívá pouze délky obrysu, takže i objekty s jiným obrysem, ale stejným obvodem budou vyhodnoceny jako stejné.

Druhý způsob využívá tvar obrysu, a na základě výpočtu Pearsonova korelačního koeficientu zjišťuje podobnost snímaného objektu s objekty v databázi.

Třetí metoda porovnávání využívá obě vlastnosti obrysu, jeho délku i tvar. Těmto dvěma parametrům je nastavena váha s jakou ovlivňují výsledný koeficient podobnosti. Jako váha pro srovnání délky obrysu bylo zvoleno 0,2 a pro porovnání tvaru obrysu pomocí pearsonova korelačního koeficientu 0,8.



Obrázek 14 - Screenshot programu

9.1 Testování aplikace

Pro testování aplikace byly zvoleny tři objekty: mince, víčko flashdisku a černý čtverec z papíru. Čtverec měl oproti ostatním objektům podstatně menší délku obrysu. Všechny tyto objekty byly uloženy do databáze a bylo zkoumáno, jak úspěšně budou detekovány v obraze. Výsledky testování jsou v následujících tabulkách. Testování probíhalo vždy jednu minutu. K realizaci testování byla do programu dočasně přidána počítadla, která ukládala počet detekcí jednotlivých objektů. Výsledky detekce trvající jednu minutu jsou zobrazeny v následujících tabulkách.

Detekce podle délky obrysu

Tabulka 2 - Výsledky detekce podle délky obrysu

	Detekované objekty					
Snímaný objekt	mince	čtverec	víčko	nedetekováno	Počet detekcí	Úspěšnost
mince	86	0	0	88	174	49%
čtverec	0	103	0	30	133	77%
víčko	3	0	48	77	128	38%

Z této tabulky jsou vidět výsledky detekce objektů v obraze podle délky jejich obrysu. Ve všech případech jsou výsledky relativně dobré. Pokud došlo k detekci objektu, tato detekce byla ve velké většině případů správná. Je to dáno také tím, že všechny objekty měly délku obrysu dostatečně odlišnou. V případě potřeby detekce objektů s podobnou délkou obrysu by bylo třeba použít některou jinou metodu.

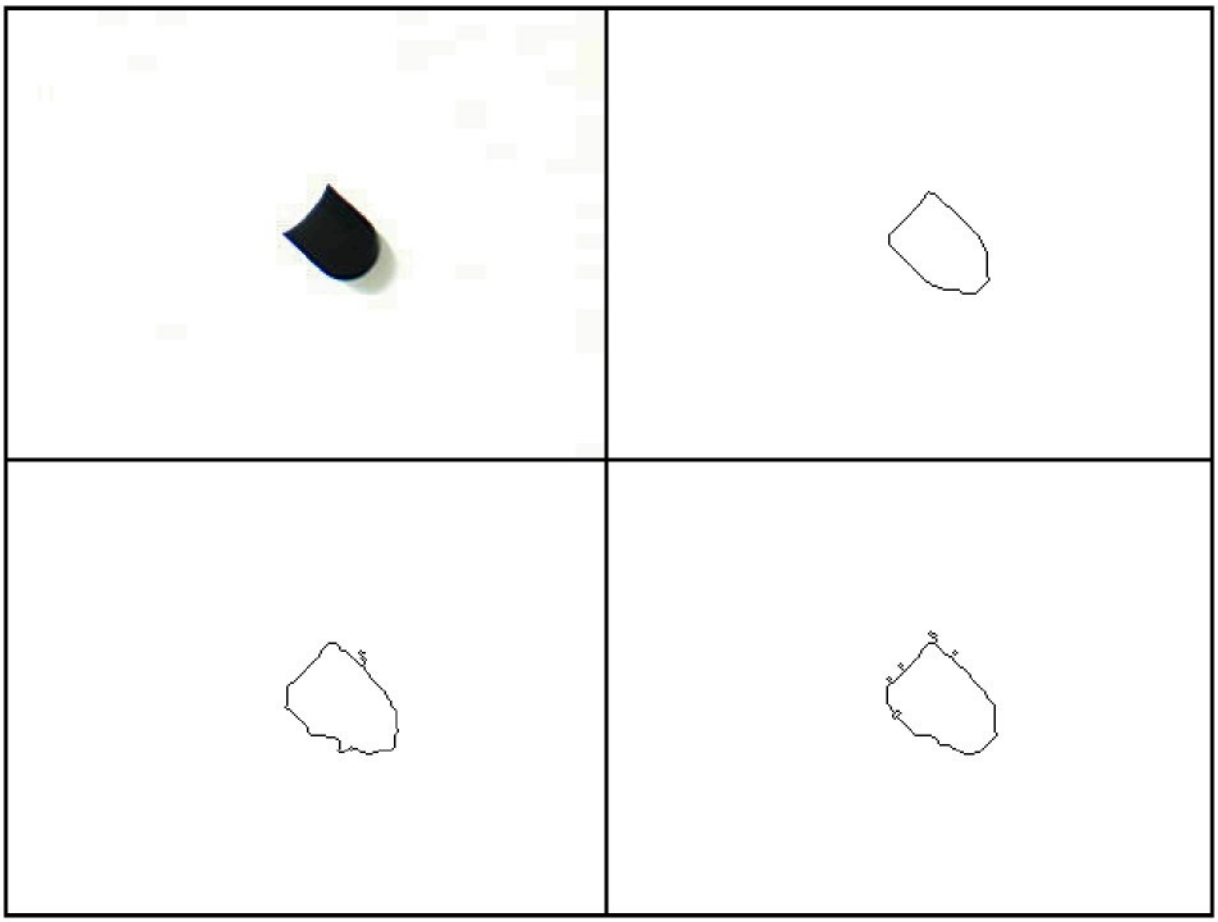
Detekce podle tvaru obrysu

Tabulka 3 - Výsledky detekce podle tvaru obrysu

	Detekované objekty					
Snímaný objekt	mince	čtverec	víčko	nedetekováno	Počet detekcí	Úspěšnost
mince	13	3	6	22	44	30%
čtverec	20	2	78	27	127	2%
víčko	9	12	46	8	75	61%

Výsledky tohoto způsobu porovnávání jsou horší než u předchozí metody. Často nedošlo k detekci vůbec. U provedených detekcí jsou výsledky neuspokojivé. To je způsobeno pravděpodobně tím, že obrys objektu není příliš hladký a obsahuje mnoho různých zakřivení. Tato zakřivení pak mají ve výsledku pro výpočet korelačního koeficientu větší váhu než skutečné změny směru obrysu. Výsledky uvedené metody v podstatě nepřinášejí žádné relevantní informace o objektech ve snímaném obraze.

Pro názornost je zobrazeno několik možností výsledku filtrování u téhož objektu. Tato nestálost velmi ztěžuje klasifikaci podle tvaru obrysu.



Obrázek 15 - Možnosti výsledku filtrování stejného objektu

Detekce podle obou parametrů

Tabulka 4 - Výsledky detekce podle obou parametrů

	Detekované objekty					
Snímaný objekt	mince	čtverec	víčko	nedetekováno	Počet detekcí	Úspěšnost
mince	103	15	55	0	173	60%
čtverec	11	84	57	0	152	55%
víčko	133	0	130	68	331	39%

Metoda využívající pro porovnání snímaných objektů s databází obě jejich vlastnosti přináší lepší výsledky než porovnávání tvaru obrysu. Vzhledem k výsledkům předchozích metod je však zřejmé, že rozhodující vliv na kvalitu detekce má porovnání podle délky obrysu. V případě víčka však docházelo vlivem stínů ke špatné detekci a bylo zaměňováno za minci.

Tyto naměřené výsledky jsou však ovlivněny kvalitou konkrétní kamery i osvětlení. V případě použití jiné kamery nebo jiného osvětlení je možné očekávat, že by se výsledky detekce poměrně lišily. Je také vidět že záleží na typech detekovaných objektů. Například při detekci objektů s rozdílnou velikostí je možno řídit se výsledky první metody s poměrně velkou jistotou.

10. Závěr

V této práci jsou stručně popsány základní algoritmy pro zpracování obrazu. Postupnou aplikací těchto jednotlivých filtrů lze z původního obrazu reálné scény získat obrys objektů v této scéně. Dále jsou zde základní informace o funkci rozhraní DirectShow, kterého se využívá při práci s kamerou a snímání obrazu.

Součástí této práce byla realizace jednoduchého programu pro klasifikaci objektů. Výsledná aplikace snímá obraz z kamery připojené k počítači, pomocí filtrů jej upravuje a klasifikuje nalezené objekty. Ty pak porovnává s objekty v databázovém souboru. V předchozí kapitole jsou výsledky testování aplikace, z nichž je vidět, že některé metody klasifikace jsou dobře použitelné, zatímco jiné prináší nepříliš uspokojivé výsledky. Funkčnost aplikace je ovšem velmi ovlivněna vnějšími podmínkami, hlavně rovnoměrností osvětlení scény. Rovnoměrným osvětlením se zabrání vzniku stínů, které mohou působit v obraze rušivě.

Ke zlepšení funkčnosti aplikace by bylo možno naprogramovat některé další filtry určené k úpravě obrazu, nebo provést další optimalizaci současných. Optimalizace by se mohla provést podle konkrétních podmínek použití programu. Další možností, která nesouvisí přímo s aplikací, by byla konstrukce speciálního držáku pro kameru. Pokud by na tomto držáku bylo umístěno i vhodné osvětlovací zařízení, předešlo by se tím vzniku různých stínů, a došlo by ke zlepšení funkčnosti aplikace.

Seznam použité literatury

[1] Steve Teixeira, Xavier Pacheco. *mistrovství v Delphi 6*. Vydání první. Praha, Computer Press, 2002. ISBN 80-7226-627-6.

[2] PÍSEK S. *Delphi – praktické príklady.* 1. vyd. Praha: Grada Publishing, 2002. ISBN 80-247-0323-8.

[3] Bc. Tomáš Selucký. *Delphi – možnosti grafu.* [online]. 17.3. 2009. URL: <http://www.selucky.com/docs/art_delphi/mendelnet_2004_delphi_moznosti_grafu.pdf>.

[4] Jan Šindelář. *Tipy a triky v Delphi, díl 37*. 10. 4. 2002. URL: <http://www.zive.cz/Clanky/Tipy-a-triky-v-Delphi-dil-37/sc-3-a-106118/default.aspx>.

[5] The Delphi Corner. Working with Files of Records. [online]. 13. 4. 2009. URL: <http://www.delphicorner.f9.co.uk/articles/apps6.htm>.

[6] About.com. *Create your own Database using Delphi's "File Of" Typed Files*. [online]. URL: <http://delphi.about.com/od/fileio/a/fileof_delphi.htm>.

[7] Ing. Michal Španěl, Ing. Vítězslav Beran. *Obrazové segmentační techniky*. [online]. URL: <http://www.fit.vutbr.cz/~spanel/segmentace/.cs.iso-8859-2#_Toc125769322>.

[8] I.T.Young, J.J. Gerbrands, L.J. van Vliet. *Image Processing Fundamentals.* [online]. URL: <http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip.html>.

[9] Karel Horák, Morfologie – *Přednáška kurzu Počítačové vidění.* [online]. URL: <http://www.uamt.feec.vutbr.cz/vision/TEACHING/MPOV/06%20-%20Morfologie.pdf>.

[10] Bohumil Kovář. *Identifikace zóny zájmu v obraze*. Praha, 1998. Diplomová práce. Fakulta dopravní. ČVUT v Praze.