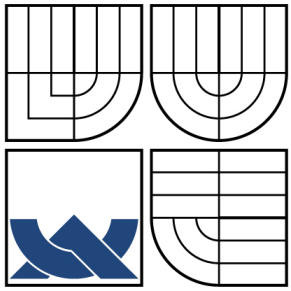
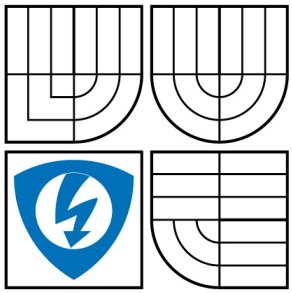


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

DOHLEDÁVÁNÍ OBJEKTŮ V OBRAZE

IMAGE OBJECT DETECTION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. RICHARD PLUSKAL

AUTHOR

VEDOUCÍ PRÁCE — Ing. MILOSLAV RICHTER, Ph.D.

SUPERVISOR

BRNO 2008

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ
Fakulta elektrotechniky
a komunikačních technologií
Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Pluskal Richard, Bc. *ID:* 88943
Ročník: 2 *Akademický rok:* 2007/08

NÁZEV TÉMATU:

Dohledávání objektů v obraze

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte prostředí pro zadávání obrazových dat (body, přímky ...) ve formátu vhodném pro následné zpracování. Zvolte vhodně typy zadávaných dat a jejich popis. Navrhněte a realizujte algoritmy usnadňující zadávání - zpřesňování polohy objektů, predikce umístění objektů při prvním a při následných zadáváních ... Ukažte použití algoritmů a zhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

Hlaváč V., Šonka M.: Počítačové vidění,Grada, Praha 1992, ISBN 80-85424-67-3
Faugeras O.: Three-Dimensional Computer Vision, The MIT Press 1993
Kraus K.: Photogrammetrie 1 und 2, Ummler / Bonn, 1996
Žára J., Beneš B., Sochor J., Felkel P.: Moderní počítačová grafika, Computer Press, 1998, ISBN 80-251-0454-0

Termín zadání: 3.12.2007 *Termín odevzdání:* 26.5.2008

Vedoucí projektu: Ing. Miloslav Richter, Ph.D.

prof. Ing. Pavel Jura, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta elektrotechniky a komunikačních technologií Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

A n o t a c e

Práce se zabývá návrhem prostředí usnadňujícího zadávání geometrických objektů v obraze za účelem jejich dalšího zpracování.

V první části je uveden stručný úvod do oblasti počítačového vidění a jeho základních metod použitých v práci a také popis použité knihovny pro zpracování obrazu. Další část popisuje základní geometrické útvary, které jsou použity ve výsledné aplikaci.

Poté jsou shrnuty některé používané metody pro hledání parametrického popisu objektů v obraze. Poslední kapitola popisuje navržený systém a uvádí příklad použití vytvořené aplikace při měření rozměrů .

Klíčová slova: zpracování obrazu, Houghova transformace, OpenCV, XML

Brno University of Technology

Faculty of Electrical Engineering and Communication

Department of Control, Measurement and Instrumentation

Image object detection

Thesis

Specialisation of study: Cybernetics, Control and Measurement
Student: Richard Pluskal
Supervisor: Ing Miloslav Richter, Ph.D.

Abstract :

The thesis deals with design of an application for entering various types of geometric objects in an image for the purpose of their further processing. The application should also contain algorithms to ease object entering (e.g. refining manually entered object position).

In the first part there is a brief description of the computer vision and its basic methods used in the work as well as introduction of the OpenCV image processing library, which was used in resulting application.

The following part describes types of geometric primitives that are implemented in the application for now.

Because the output of the program should be in the format suitable for subsequent processing there is short description of the XML, which satisfies this goal.

After that, there are summarized some methods for searching of parametric description of geometric primitives in an image.

The final chapter describes the proposed system and shows its application for optical measurement .

Keywords: image processing, Hough transform, OpenCV, XML

Bibliografická citace

PLUSKAL, R. *Dohledávání objektů v obraze.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 68 s. Vedoucí diplomové práce Ing. Miloslav Richter, Ph.D.

P r o h l á š e n í

„Prohlašuji, že svou diplomovou práci na téma Dohledávání objektů v obraze jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne : Podpis:

P o d ě k o v á n í

Děkuji tímto svému vedoucímu, ing. Miloslavu Richterovi, Ph.D. za rady při vypracování diplomové práce.

V Brně dne : Podpis:

SEZNAM OBRÁZKŮ

1. ÚVOD

Cílem této diplomové práce je návrh prostředí, které by usnadňovalo zadávání obrazových dat a poskytovalo výsledky ve formátu vhodném pro následné další zpracování.

Nejprve je třeba zvolit typy zadávaných dat a jejich reprezentaci. Jako vhodné se jeví použití jednoduchých geometrických útvarů, které vzájemnou kombinací mohou reprezentovat složitější útvary a přitom jejich zadávání je pro uživatele snadné.

Algoritmy usnadňující zadávání obrazových dat, patřící do oblasti počítačového vidění, mohou uživateli usnadnit a urychlit práci a také mohou pomoci zpřesnit výsledky.

Využití navrženého prostředí je možné například v oblasti optického měření tvaru a rozměrů nebo pro hledání společných bodů na různých obrazech pro tzv. registraci obrazů.

Práce je strukturována následovně:

V první kapitole jsou shrnuty potřebné základy oblasti počítačového vidění, předzpracování obrazu a jeho segmentace. Ve druhé kapitole jsou popsány základní geometrické útvary, které jsou v aplikaci využity při zadávání obrazových dat.

Vzhledem k požadavku na poskytování výsledků pro další zpracování je vhodné zvolit univerzální formát výstupu, který zajistí co nejlepší přenositelnost. Velmi univerzálním formátem pro přenos dat je stále více se prosazující XML, jehož základní vlastnosti jsou popsány ve třetí kapitole.

V páté kapitole jsou zmíněny některé z různých přístupů, které lze použít pro řešení úloh hledání parametrického popisu objektů v obraze.

Šestá kapitola popisuje vytvořenou aplikaci.

2. ZPRACOVÁNÍ OBRAZU

Zpracování obrazu je forma zpracování informace, jejímž vstupem je obraz. Výstupem nemusí být nutně obraz, ale například soubor příznaků, popisující daný obraz. Zpracováním obrazu zde nejsou myšleny pouze jeho úpravy, jako jsou například geometrické nebo barevné korekce, ale i interpretace jeho obsahu způsobem podobným člověku. Obor, který se takovýmto zpracováním a vyhodnocováním obrazu zabývá se označuje jako počítačové vidění *(computer vision)*.

2.1 POČÍTAČOVÉ VIDĚNÍ

Počítačové vidění je vědní obor, který se technickými prostředky snaží napodobit některé schopnosti lidského vidění. Při vyhodnocení obrazové informace člověkem hrají podstatnou roli jeho předchozí zkušenosti. Inteligence nám umožňuje reprezentovat nabyté znalosti či zkušenosti o okolním světě a využívat je pro řešení nových úloh. Počítačové vidění je v širším smyslu považováno za součást kybernetiky či umělé inteligence.

Některé současné aplikace počítačového vidění jsou například:

- medicína (3D rekonstrukce tomografických snímků)
- automatická vizuální kontrola výrobků v průmyslu
- sledování dopravní situace (evidence dopravních přestupků)
- vojenství (automatické navádění střel)
- rozpoznávání textu (*OCR*)

Aplikace počítačového vidění se s postupujícím výzkumem samozřejmě neustále rozšiřují do nových oblastí. Příkladem může být například vývoj autonomních vozidel nebo systémy pro rozpoznávání lidských gest, které budou moci být využity budoucími počítačovými systémy jako rozhraní pro interakci s člověkem.

2.1.1 Typické úlohy

Výše popisované aplikace počítačového vidění zahrnují řadu jednotlivých úloh, které mohou být řešeny pomocí různých přístupů mnoha různými metodami. Zde jsou některé příklady těchto úloh:

- ***Rozpoznávání objektu***

  Úkolem je zjistit, zda v obraze je nebo není přítomen určitý objekt nebo má obraz nějakou, předem určenou, vlastnost. (Takováto úloha je běžně řešena člověkem bez zvláštní pozornosti nebo úsilí. Počítačové vidění ji úspěšně řeší pro specifické objekty, například jednoduché geometrické objekty, za specifických podmínek, jako dobré osvětlení a daná pozice vůči kameře. Obecný případ – rozpoznání obecného objektu za obecných podmínek, ovšem vyřešen není.)

- ***Detekce pohybu***

  Jde o zpracování sekvence snímků s cílem zjistit případný pohyb v obraze (jeho směr a rychlost).

- ***Rekonstrukce scény***

  Cílem je vytvořit trojrozměrný model scény ze série snímků pořízených z různých úhlů.

- ***Restaurace obrazu***

  Úkolem je odstranit poškození obrazu (způsobené např. šumem nebo rozmazáním obrazu při snímání).

2.1.2 Hlavní problémy

Interpretace obrazových dat je základem porozumění obrazu v počítačovém vidění. Snahou je napodobit proces vnímání u člověka a jemu podobný způsob rozhodování na základě informace obsažené v obrazech. Některé z hlavních příčin obtíží v interpretaci obrazových dat v úlohách počítačového vidění jsou tyto: (podle [1])

- **Ztráta informace při perspektivním zobrazení** původně trojrozměrné scény do dvojrozměrné roviny snímače. Všechny body na polopřímce dané bodem [x,y] v obrazové rovině ve směru od středu promítání se zobrazí právě do bodu [x,y]. Zpětná úloha, která se snaží odvodit trojrozměrné vlastnosti objektů z obrazu jediné kamery, má nekonečně mnoho řešení. Ve skutečnosti ji lze řešit jen s využitím dodatečných informací.

- **Nejednoznačný vztah mezi jasem, měřeným kamerou a tvarem povrchu 3D objektů ve scéně.** Jas bodu závisí na více vlivech:

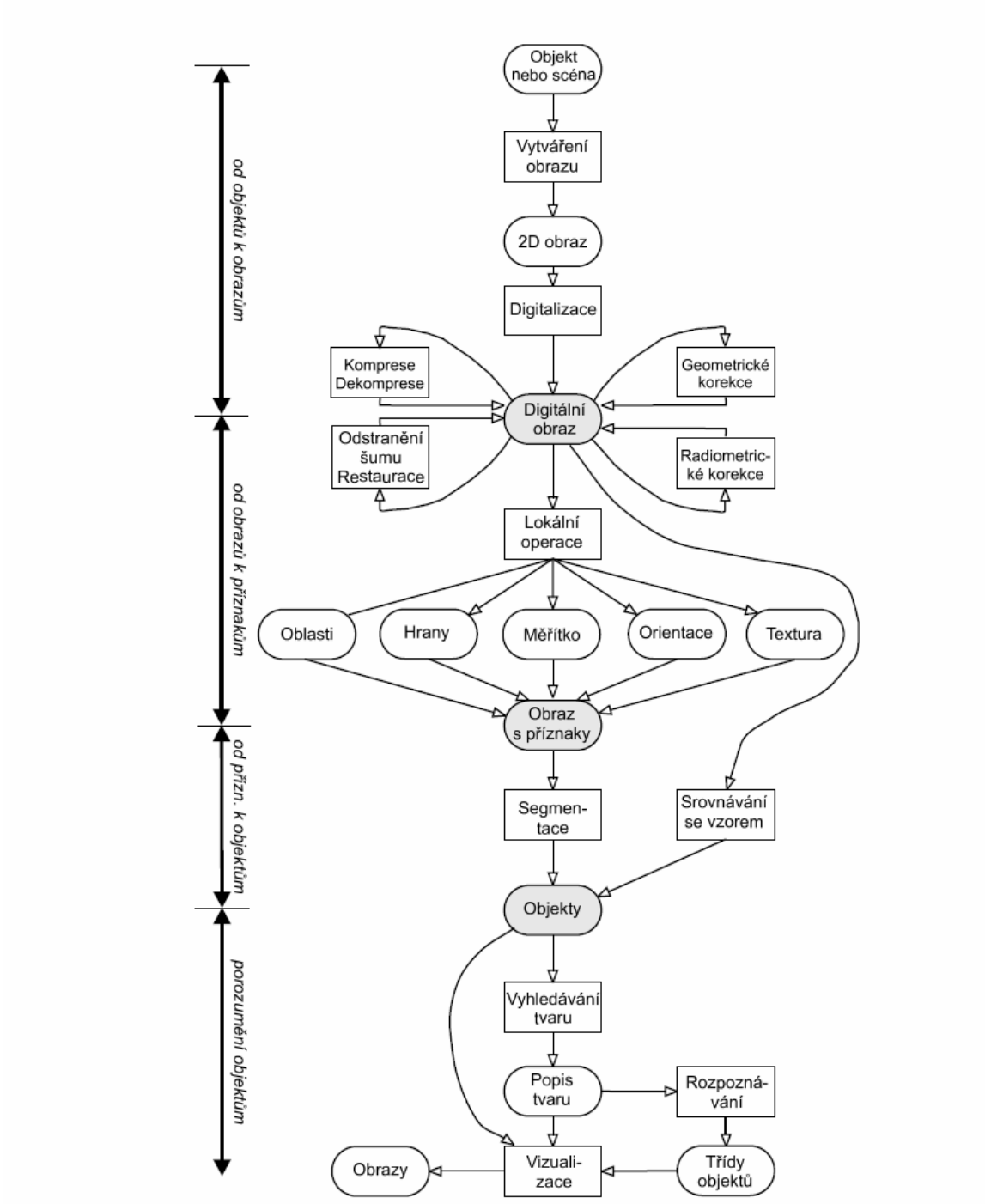
  - odrazivost povrchu pozorovaného předmětu
  - poloha a vlastnosti zdrojů světla
  - orientace povrchu vzhledem k pozorovateli

- **Velké množství obrazových dat**. Běžný černobílý televizní signál, chceme-li zachovat kvalitu původního analogového signálu, je nutné digitalizovat do obrazu o rozlišení 512x768 pixelů v 256 úrovních jasu. Takových obrazů je v televizním signálu 25 za sekundu. Zpracováváme tedy datový tok 9,4 MB/s. I statické obrazy obsahují velké množství dat. Jedna stránka formátu A4 digitalizovaná černobíle skenerem v rozlišení 300 dpi ve 256 jasových úrovních má datový objem zhruba 8,3 MB.

- **Šum v obraze reálné scény.** Je přítomen vždy a je hlavním důvodem, proč je při zpracování často nutné použít pravděpodobnostní techniky. Často ovšem není k dispozici tolik obrazů, aby bylo možné správně odhadnout statistické vlastnosti obrazových signálů.

- **Vztah mezi pozorovaným detailem a zjišťovaným celkem.** Algoritmy zpracování obrazu obvykle analyzují vlastnosti části obrazu prostřednictvím malého posuvného okna. Tímto způsobem se ovšem těžko zjišťují globální vlastnosti obrazu, o které většinou jde.

  Schéma zpracování obrazu metodami počítačového vidění je na Obr. 1.



Obr. 1 Postup zpracování obrazu metodami počítačového vidění

2.2 OBRAZ

Matematickým modelem obrazu je spojitá funkce dvou proměnných – obrazová funkce:

$$z = f(x, y) \tag{1}$$

Protože u obrazu předpokládáme omezené rozměry, můžeme definiční obor obrazové funkce zapsat jako kartézský součin dvou spojitých intervalů z oboru reálných čísel, které vymezují rozsah obrazu. Obrazová funkce realizuje zobrazení

$$f : (\langle x_{\min}, x_{\max} \rangle \times \langle y_{\min}, y_{\max} \rangle) \rightarrow H \tag{2}$$

Reálná čísla x, y , kde x: $x_{min} \leq x \leq x_{max}$ a y: $y_{min} \leq y \leq y_{max}$ jsou souřadnice bodu ve dvou rozměrech, v nichž funkce nabývá nějaké hodnoty z oboru hodnot H.

Hodnota obrazové funkce může být jediné reálné číslo (např. jas, intenzita modré barvy, atp.), v počítačové grafice je to ale obvykle více hodnot, typicky trojice červená, zelená a modrá složka obrazové informace v tzv. barevném modelu RGB. V některých případech může z reprezentovat celé spektrum hodnot (např. data z počítačového tomografu nebo spektrální měření svitu hvězd), jindy je výhodnější pracovat s komplexními čísly. Funkce tedy může nabývat hodnot uspořádaných n-tic údajů $z=[z_1,z_2,...,z_n]$

$$f : (\langle x_{\min}, x_{\max} \rangle \times \langle y_{\min}, y_{\max} \rangle) \rightarrow (H_1 \times H_2 \times ... \times H_n) \tag{3}$$

Obraz v počítači je ovšem uložen v diskrétní podobě, je proto třeba přejít od spojité funkce f(x,y) k diskrétní funkci $I_{i,j}$ s diskrétním oborem hodnot. Tento proces nazýváme digitalizace.

Digitalizace má dva nezávislé kroky: vzorkování a kvantování.

Vzorkováním rozumíme záznam hodnot – vzorků – v předem daných pravidelných intervalech. Jestliže je splněn vzorkovací teorém, nedochází při vzorkování ke ztrátě informace.

Kvantování je proces, který určitému intervalu hodnot přidělí jedinou zástupnou hodnotu. Tím dochází ke ztrátě informace.

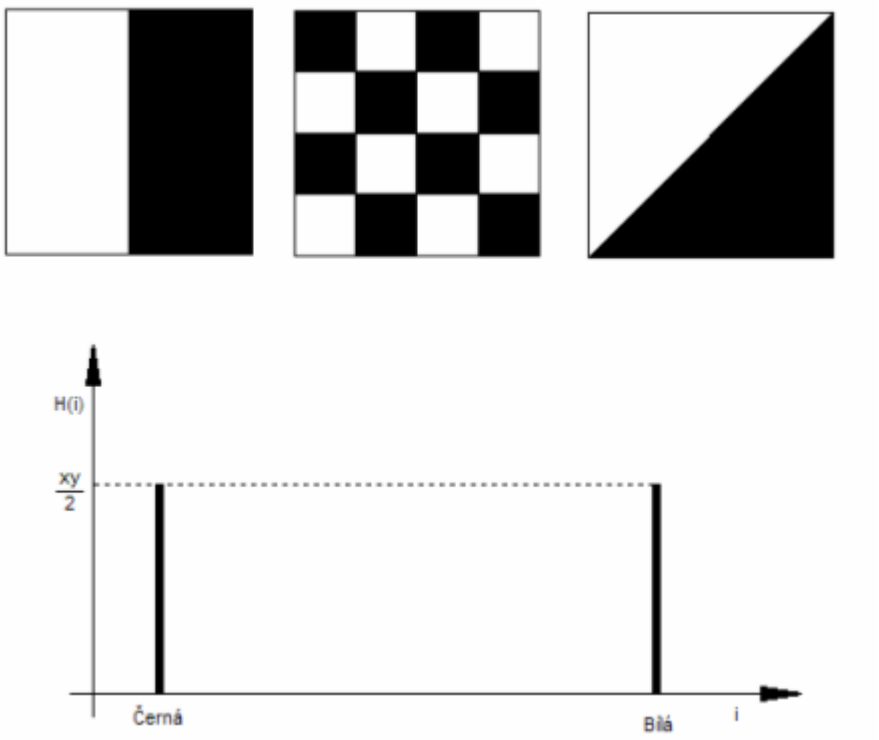
V praxi je obor hodnot obrazové funkce $I_{i,j}$ omezen použitou aplikací nebo parametry technického zařízení. V počítačové grafice se například pracuje nejčastěji s 16ti miliony barev, s 256ti stupni šedi nebo s 256ti odstíny barev.

2.2.1 Histogram

Histogram kvantifikuje množství a frekvenci výskytu barev zastoupených v obraze. Matematicky je histogram vektor absolutních četností zastoupených hodnot, tzn. hodnota H pro index i říká, kolik pixelů v obraze má intenzitu i. Histogram barevného obrazu RGB má tři složky, v případě černobílého obrazu má pouze jednu složku. Součet všech četností je počet pixelů v obrazu:

$$\sum_{i=0}^{\max} H(i) = xy \qquad (4)$$

Histogram je statistická veličina, která kvantifikuje jasové poměry v obraze, ale nenese žádnou informaci o jejich plošném rozložení, jak ilustrují níže uvedené obrazy a jejich společný histogram na Obr. 2.



Obr. 2 Různé obrazy a jejich společný histogram

2.3 SEGMENTACE OBRAZU

Segmentace obrazu je jedním z nejdůležitějších kroků při analýze obsahu zpracovávaných obrazových dat. Segmentace rozděluje obraz do částí, které mají úzkou souvislost s předměty nebo oblastmi reálného světa zachycenými na obraze. Výsledkem segmentace by měl být soubor vzájemně se nepřekrývajících oblastí, které buď jednoznačně korespondují s objekty vstupního obrazu, pak jde o kompletní segmentaci, nebo vytvořené segmenty nemusí přímo souhlasit s objekty obrazu a pak jde o částečnou segmentaci. Častá je situace, kdy je obraz tvořen kontrastními objekty na pozadí neměnného jasu - např. tištěný text. V tomto případě lze užít globální postupy a dosáhnout kompletní segmentace obrazu na objekty a pozadí.

Při částečné segmentaci je výsledkem rozdělení obrazu do samostatných částí, které jsou homogenní vzhledem k určitým zvoleným vlastnostem, jako jsou jas, barva, odrazivost, textura. Po zpracování je výsledkem seznam oblastí, které jsou homogenní v jistých zvolených rysech. Oblasti se obecně mohou překrývat. Na data popisující částečnou segmentaci je nezbytné aplikovat další postupy, které pomocí vyšší úrovně zpracování umožní získat výslednou segmentaci obrazu.

Dokonale správná segmentace složitějších obrazů není v této fázi zpracování obvykle dosažitelná. Dosažitelným cílem je zisk částečné segmentace, jejíž výsledky mohou být zpřesněny při následném zpracování operacemi vyšších úrovní. Přínosem segmentace je také výrazná redukce objemu zpracovávaných dat. Jedním z hlavních problémů ovlivňujících segmentaci je nejednoznačnost obrazových dat, často doprovázených šumem. Metody segmentace lze rozdělit do tří skupin:

1. Metody využívají globální znalosti obrazu (nebo jeho části) reprezentované obvykle histogramem určitých vlastností.
2. Postupy vycházející z určování hranic mezi oblastmi (částmi) obrazu
3. Postupy přímo vytvářející tyto oblasti (jejich části).

Není přitom podstatné, z jakých charakteristik (jas, barva, atd.) při hledání hranic nebo při tvorbě oblastí vycházíme. Každá oblast je jednoznačně

reprezentována svou hranicí a každá uzavřená hranice jednoznačně vypovídá o oblasti, kterou obklopuje. V důsledku odlišného charakteru obrazových dat i odlišných algoritmů tvorby hranic a oblastí přináší každá skupina metod rozdílné segmentační výsledky. Proto můžeme výsledky obou skupin postupů kombinovat a vytvořit jednu pro strukturu pro jejich popis.

2.3.1 Prahování

Prahování je nejjednodušší metoda segmentace. Je také nejstarší a kvůli výpočetní nenáročnosti také nejrychlejší (při použití specializovaného hardware ji lze provádět v reálném čase). Mnoho objektů nebo oblastí obrazu je charakterizováno konstantní odrazivostí či pohltivostí svého povrchu. Potom je možné využít určenou konstantu prahu k oddělení objektů od pozadí. Výsledek prahování je v ideálním případě kompletní segmentace do oblastí.

Kompletní segmentací obrazu R je nazývá konečná množina oblastí$\{R_1, R_2, ..., R_n\}$, pro kterou platí

$$R = \bigcup_{i-1}^{m} R_i, \qquad R_j \bigcap R_i = 0 \, pro \qquad i \neq j. \tag{5}$$

Prahování je transformace vstupního obrazu f na výstupní segmentovaný binární obraz g podle vztahu

$$g(i,j) = \begin{cases} 1 & pro \quad f(i,j) \geq T, \\ 0 & pro \quad f(i,j) < T, \end{cases} \tag{6}$$

kde T je předem určená konstanta nazývaná práh. Funkce *g(i,j)* je rovna 1 pro pixely náležející po segmentaci objektům a je rovna 0 pro pixely náležející k pozadí. Prahování prochází postupně všechny pixely obrazu *f*. Správná volba prahu je velmi důležitá pro úspěšný výsledek prahování. Hodnotu prahu lze určovat pokusně nebo pomocí některé z metod automatického určování prahu. Zpravidla není možné úspěšně použít stejný práh na celé ploše obrazu, dokonce ani v jednoduchých scénách nemusí dát globální prahování s jediným prahem požadované segmentační výsledky. To může být způsobeno změnami jasu objektů i pozadí zaviněnými nerovnoměrností osvětlení nebo nestejnými vlastnostmi snímacího zařízení v celé

ploše obrazu. V této situaci může pomoci prahování s proměnným prahem (adaptivní prahování), kdy se hodnota prahu určuje podle lokálních vlastností obrazu.

Globální práh se určuje z celého obrazu. Lokální práh je funkcí polohy, tzn. je určován pro část obrazu f_P. Části obrazu f_P lze získat například rozdělením obrazu na podobrazy a určit práh nezávisle v každém z nich. Pokud v některém z podobrazů postup určování lokálního prahu selže, použijeme práh získaný interpolací sousedních prahů. Každý podobraz prahujeme lokálním prahem.

Jednou z modifikací základního prahování je prahování podle vztahu

$$g(i,j)=\begin{cases}1 & pro \quad f(i,j)\in A,\\ 0 & jinak,\end{cases} \tag{7}$$

kde A je jistá množina úrovní jasu. Takto lze segmentovat obraz na oblasti, jejichž jas je z množiny A, a na ostatní oblasti. Toho je možné využít například pro segmentaci mikroskopických snímků krevních buněk, kde se cytoplazma jeví v určitém intervalu úrovní jasu, zatímco pozadí je světlejší a jádro naopak tmavší. Takto definované prahování můžeme použít i pro detekci hranic objektů.

Vhodnou volbou množiny A lze získat izočáry úrovní jasu.

Modifikací metody je prahování s více prahy, kdy výsledkem již není binární obraz, ale obraz s velmi omezeným počtem jasových úrovní

$$g(i,j)=\begin{cases}1 & pro\ f(i,j)\in A_1,\\ 2 & pro\ f(i,j)\in A_2,\\ \quad\vdots & \\ n & pro\ f(i,j)\in A_n,\\ 0 & jinak\end{cases} \tag{8}$$

kde A_i jsou podmnožiny jasových úrovní.

Zvláštním případem definice podmnožin A_i je poloprahování, které se používá pro vizuální hodnocení člověkem

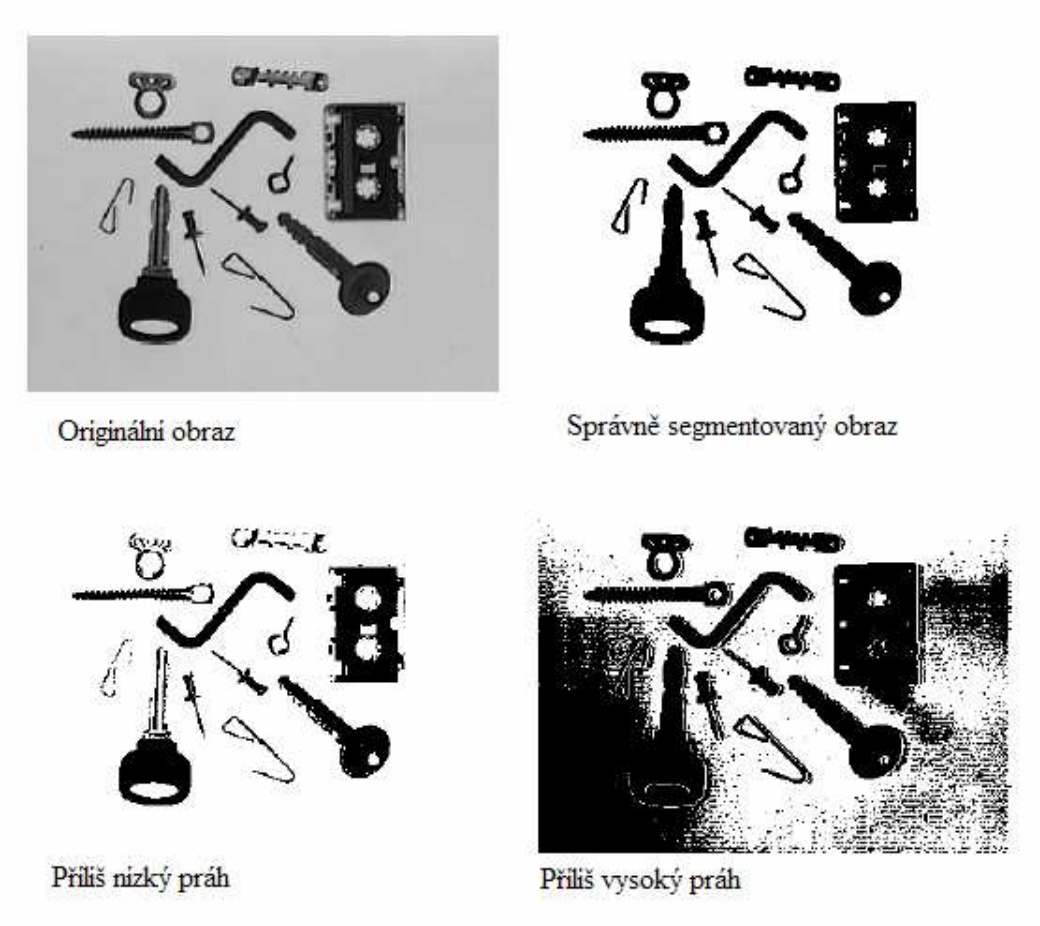
$$g(i,j)=\begin{cases}f(i,j) & pro\ f(i,j)\in A,\\ 0 & jinak.\end{cases} \tag{9}$$

Tímto způsobem je možno získat obraz, ve kterém bylo odstraněno pozadí, ale v objektech byla zachována informace o rozložení jasu.

Tento přístup můžeme použít vždy, když *f(i,j)* představuje hodnotu libovolného kritéria dekompozice obrazu.

2.3.2 Metody určování prahu

Hodnota prahu je naprosto klíčová pro správnou segmentaci obrazu (viz Obr. 3), ale ve většině případů je obtížné ji vhodně stanovit. Proto byly odvozeny některé metody, které se touto problematikou zabývají. Univerzální metoda pro určení prahu ovšem neexistuje.



Obr. 3 Vliv hodnoty prahu na výsledek segmentace

Nejjednodušší situace nastává, jestliže předem známe vlastnost, kterou má obraz po segmentaci mít. Potom práh určujeme tak, aby byla tato vlastnost segmentací splněna. S využitím znalosti poměru ploch objektů a pozadí můžeme na

základě histogramu určit takovou hodnotu prahu *T*, aby právě *1/p* plochy mělo úrovně jasu menší než *T*. Tato metoda se nazývá procentní prahování. V praxi však znalost procentuálního zastoupení není obvykle k dispozici, ale známe jinou veličinu, např. průměrnou šířku čar znaků. Práh *T* můžeme volit tak, aby šířka čar v segmentovaném obraze odpovídala předchozí znalosti.

Složitější metody analyzují tvar histogramu. Jestliže jsou v obrazu objekty přibližně stejného jasu a jasově odlišné pozadí, pak je histogram jasu bimodální (dvouvrcholový), jeden vrchol odpovídá četnosti pixelů pozadí, druhý četnosti pixelů objektů. Z tvaru histogramu vyplývá, že hodnoty jasu ležící mezi oběma vrcholy nejsou v obraze časté a odpovídají jasu hraničních pixelů mezi objekty a pozadím. Výsledný práh by měl splňovat požadavek minimální segmentační chyby. Práh pro oddělení objektů a pozadí se určí jako hodnota jasu, jejíž četnost je minimem ležícím mezi dvěma maximy. V případě, že je histogram multimodální (vícevrcholový), je možno určit více hodnot prahu, a to v minimech mezi dvěma maximy. Pro každý z určených prahů je výsledek segmentace jiný, nebo lze provést segmentaci s více prahy. Rozhodnutí, jestli je histogram dvouvrcholový nebo vícevrcholový, není jednoduché. Často nelze jednoznačně rozhodnout o významnosti lokálních maxim histogramu. Pokud zpracovaný obraz obsahuje objekty jasově odlišné od pozadí, pak je uvedený postup efektivní. Ze samotné existence dvou významných vrcholů v histogramu ale nelze soudit na správnost výsledku segmentace, protože obraz nemusí obsahovat objekty na jasově odlišném pozadí.

2.4 ODSTRANĚNÍ ŠUMU

V reálném obrazu je vždy přítomen šum, který je třeba odstranit. Nejjednodušší je odstranění náhodného šumu, když máme k dispozici několik obrazů stejné předlohy, které se liší právě šumem. Potom je možné například průměrovat hodnotu pixelu na stejných souřadnicích přes více obrázků nebo lze použít nejčastěji se vyskytující hodnotu, medián, apod. Podobnou techniku používají i digitální fotoaparáty v režimu „noční záběr“. První snímek je pořízen bez otevření závěrky, na snímku je jen šum snímacího prvku, poté je pořízen snímek běžným způsobem a první snímek je od druhého odečten. Tímto jednoduchým způsobem se potlačí jinak velmi výrazný šum snímacího prvku s nastavenou velkou citlivostí, nutnou pro dosažení přijatelného jasu při nízké úrovni osvětlení.

Ve většině případů ovšem nemáme více obrazů stejné předlohy. Odstranění šumu v jediném obrazu se také nazývá filtrace šumu. Protože nemáme k dispozici původní obrazovou funkci nezkreslenou šumem, nevíme, co je šum a co původní signál. V reálném obrazu jsou obvykle změny intenzity sousedních pixelů malé, výjimkou ostrých hran. Filtry proto vyhodnocují okolí daného pixelu a pro velké změny okolních pixelů označí pixel za šum. Tento přístup ovšem považuje za šum veškerou vysokofrekvenční informaci, tedy i hrany, které rozmazává.

Filtry šumu pracují buď na principu konvoluce (lineární filtry) nebo lokální statistiky okolí (nelineární filtry). Lineární filtry se dělí na filtry typu horní propust nebo dolní propust.

Filtry typu horní propust umožní zvýraznit detaily v obraze, ale zároveň také dojde obvykle ke zvýraznění šumu. Pro filtry typu horní propust je charakteristické, že součet váhových koeficientů v matici je roven 0.

Filtry typu dolní propust slouží především k odstranění vysokých prostorových frekvencí intenzit v obraze, čímž dojde k potlačení nežádoucího šumu, ale také k potlačení detailů v obraze. Pro filtry typu dolní propust je charakteristické, že součet váhových koeficientů v matici je roven 1. Dále budou uvedeny pouze filtry typu dolní propust.

2.4.1 Konvoluční metody (lineární filtry)

Konvoluce dvou funkcí je definována jako:

$$I(x) * h(x) = \int_{-\infty}^{\infty} I(x-\alpha)h(\alpha)d\alpha \qquad (10)$$

Při práci s digitálním obrazem se používá diskrétní konvoluce, která je diskrétní dvourozměrnou podobou konvolučního integrálu:

$$I'(i,j) = I * h|(i,j) = \sum_{m=-k}^{k}\sum_{n=-k}^{k} I(i-m, j-n)h(m,n). \qquad (11)$$

Konvoluční jádro *h(x,y)* je možné popsat tabulkou o rozměrech $\langle -k,k\rangle \times \langle -k,k\rangle$. Diskrétní konvoluce má tento význam: Výstupní obraz *I'(i,j)* se získá tak, že na každý pixel vstupního obrazu *I(i,j)* položíme konvoluční jádro *h(x,y)* a vypočítáme součet podle výše uvedeného vztahu (11). Jednou z vlastností diskrétní konvoluce je, že pro jakýkoliv obraz bez ohledu na jeho obsah se provede konstantní počet operací. To je výhoda při použití v paralelních výpočtech. Většina dnešních grafických karet podporuje rozšíření OpenGL verze 1.2, které provádí konvoluci přímo na grafické kartě.

Nejjednodušší metoda filtrace šumu je výpočet hodnoty pixelu jako průměru z jeho okolí. Tím se ze spektra odstraní vysoké frekvence – šum, ale i hrany, které se rozmažou. Tato metoda má název obyčejné průměrování a její konvoluční jádro (pro okolí 3x3) je následující:

$$h_{i,j} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Modifikací je filtr Gaussova šumu:

$$h_{i,j} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Pokud chceme zachovat původní jas obrazu, musí být součet hodnot v konvolučním jádru pro odstranění šumu jedna. Pokud by byl vyšší, výsledný obraz by byl světlejší, při součtu nižším než jedna by byl výsledek tmavší.

2.4.2 Metody pracující na základě lokální statistiky (nelineární filtry)

Nelineární filtry nepočítají intenzitu upravovaného bodu, ale vybírají z jeho okolí vhodnou hodnotu, kterou pak dosazují jeho hodnotu. Oproti lineárním filtrům mají tu výhodu, že nepřidávají do obrazu žádnou novou hodnotu intenzity.

Filtr medián vybírá z blízkého okolí bod se střední hodnotou intenzity, kterou pak dosadí do upravovaného bodu. Tento filtr je velmi účinný pro potlačení šumu. Jeho nevýhodou je, že ohlazuje hrany objektů, čímž mění jejich tvary.

Další popisované nelineární filtry patří do oblasti tzv. matematické morfologie.

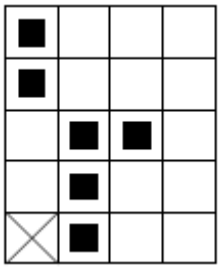
2.4.3 Matematická morfologie

Matematická morfologie se začala vyvíjet v šedesátých letech a svým matematickým aparátem, vycházejícím z algebry nelineárních operací, do značné míry při zpracování signálů či obrazů předstihuje tradiční lineární přístup, který využívá lineární kombinaci (konvoluci) bodových zdrojů představovaných Diracovými impulsy. Jde např. o předzpracování obrazu, o segmentaci s důrazem na tvar hledaných objektů, o kvantitativní popis nalezených objektů.

Matematická morfologie využívá vlastnosti bodových množin, výsledky z integrální geometrie a topologie. Výchozím předpokladem je představa, že reálné obrazy lze modelovat pomocí bodových množin libovolné dimenze (např. N-rozměrný euklidovský prostor).

V počítačovém vidění se používá digitální protějšek euklidovského prostoru. Pro binární matematickou morfologii, zpracovávající dvouúrovňový obraz, je základem množina dvojic celých čísel, pro šedotónovou matematickou morfologii, která zpracovává obraz s více úrovněmi, množina trojic celých čísel. Dále bude popsána binární matematická morfologie.

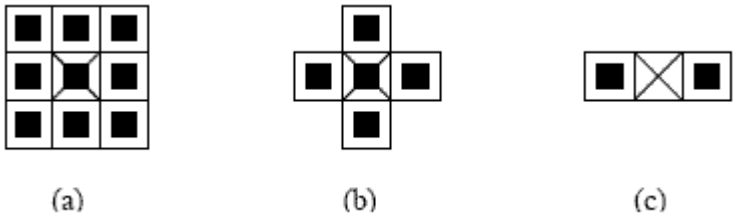
Na binární obraz lze nahlížet jako na podmnožinu 2D prostoru všech celých čísel Z^2. Pixel je reprezentován dvojicí celých čísel vzhledem ke dvěma souřadným osám diskrétní mřížky. Jednotková délka podél os odpovídá periodě vzorkování. Binární obraz lze vyjádřit jako 2D bodovou množinu, viz Obr. 4.



Obr. 4 Příklad bodové množiny (černé čtverečky reprezentují body objektu)

Body objektu v obraze reprezentují množinu X, což odpovídá pixelům s hodnotou jedna. Body doplňku X^c popisují pozadí a reprezentují pixely s hodnotou nula. Počátek (na Obr. 4 označen křížkem) má souřadnice [0,0] a souřadnice ostatních bodů [x,y] mají obvyklý význam.

Morfologická transformace Ψ je dána relací mezi obrazem (bodová množina X) a jinou, typicky menší bodovou množinou B, která se nazývá strukturní element. Strukturní element B je vztažen k lokálnímu počátku O, který se označuje jako reprezentativní bod. Příklady některých typických strukturních elementů jsou na Obr. 5.



Obr. 5 Příklad izotropních (a,b) a anizotropních (c) strukturních elementů

Aplikaci morfologické transformace si lze představit jako systematické posouvání strukturního elementu B po vstupním obrazu. Výsledek relace mezi obrazem X a strukturním elementem B se zapíše do výstupního obrazu v reprezentativním pixelu. Pro popisovanou binární morfologii je výsledek relace 0 nebo 1.

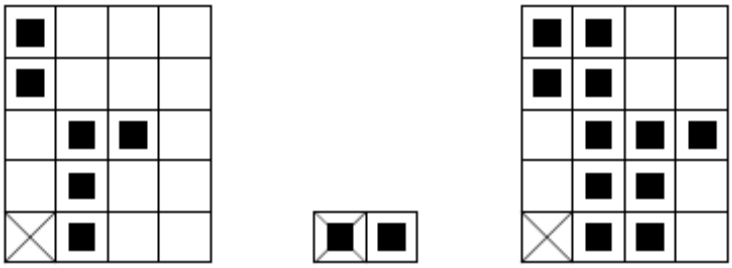
Mezi základní transformace binární matematické morfologie patří dilatace a eroze, jejich složením vznikne otevření a uzavření.

2.4.3.1 Dilatace

Dilatace $\oplus$ skládá body dvou množin pomocí vektorového součtu, např. *(a,b)+(c,d)=(a+c,b+d)*. Dilatace $X \oplus B$ je bodovou množinou všech možných vektorových součtů pro dvojice pixelů, vždy pro jeden z množiny X a jeden z množiny B:

$$X \oplus B = \left\{ p \in \varepsilon^2 : p = x + b, \quad x \in X, b \in B \right\}. \tag{12}$$

Příklad dilatace je na Obr. 6.



Obr. 6 Dilatace

Dilatace se používá samostatně k zaplnění malých děr, úzkých zálivů a jako základ složitějších operací. Dilatace zvětšuje objekty, pokud se má po použití dilatace zachovat původní velikost objektu, kombinuje se s erozí (viz dále).

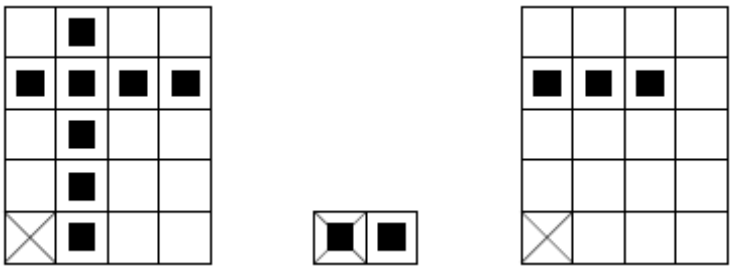
2.4.3.2 Eroze

Eroze je duální operace k dilataci, skládá dvě množiny podle předpisu:

$$X - B = \left\{ p \in \varepsilon^2 : p + b \in X \ \ pro\ každé\ b \in B \right\}. \tag{13}$$

Předchozí vztah znamená, že pro každý bod obrazu p se ověřuje, jestli pro všechna možná $p+b$ leží výsledek v X. Pokud ano, zapíše se v reprezentativním bodě do výsledného obrazu 1, v opačném případě 0. To lze také interpretovat jako posouvání strukturního elementu B po obrazu X. Pokud je B posunutý o vektor p obsažen v obrazu X, potom bod odpovídající reprezentativnímu bodu B patří do eroze X-B.

Příklad eroze je na Obr. 7.



Obr. 7 Eroze

Eroze se používá pro zjednodušení struktury objektů – objekty tloušťky 1 se ztratí, a tak se složitější objet může rozdělit na několik jednodušších.

2.4.3.3 Otevření a uzavření

Otevření a uzavření jsou tvořeny kombinací dilatace a eroze. Výsledkem obou transformací je zjednodušený obraz, obsahující méně detailů.

Eroze následovaná dilatací vytváří otevření. Otevření množiny *X* strukturním elementem *B* se označuje $X \circ B$ a je definováno jako:

$$X \circ B = (X - B) \oplus B. \qquad (14)$$

Dilatace následovaná erozí vytváří uzavření. Uzavření množiny *X* strukturním elementem *B* se označuje $X \bullet B$ a je definováno jako:

$$X \bullet B = (X \oplus B) - B. \qquad (15)$$

Pokud se obraz X nezmění po otevření strukturním elementem *B*, nazývá se obraz otevřený vzhledem k *B*, obdobně pokud se obraz nezmění po uzavření, nazývá se uzavřený vzhledem k *B*.

Otevření a uzavření izotropickým strukturním elementem se používá pro odstranění obrazových detailů, které jsou menší než strukturní element. Celkový tvar objektu tak zůstane po jejich použití neporušený. Otevření oddělí objekty spojené úzkou šíjí a zjednoduší tak strukturu objektů. Uzavření spojí objekty, které jsou blízko u sebe, zaplní malé díry a vyhladí obrys tím, že zaplní úzké zálivy. Pojmy „blízký“, „malý“ a „úzký“ jsou relativní vzhledem k velikosti strukturního elementu.

2.5 DETEKCE HRAN

Hranu v diskrétním obraze vnímáme tam, kde dochází k výrazné změně sousedních pixelů. Hrana je vysokofrekvenční informace, její zvýraznění je proto inverzní operací k odstranění šumu. Hrana je určena gradientem, tzn. velikostí a směrem. Směr lze popsat vektorovým operátorem nabla ∇:

$$\nabla f(x,y) = \left(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y} \right), \tag{16}$$

velikost je tedy určena jako délka vektoru:

$$\left|\nabla f(x,y)\right| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x} \right)^2 + \left(\frac{\partial f(x,y)}{\partial y} \right)^2}\ . \tag{17}$$

Výše uvedené vzorce platí pro spojité funkce. V diskrétním obraze gradient odhadujeme. Pro jeho určení se používají postupy založené na analýze okolí pixelu s použitím konvolučních nebo jiných operátorů.

Asi nejjednodušší metodou je použití tzv. Robertsova operátoru. Tento operátor není založen na konvoluci a jeho výhodou je velmi snadná implementace. K výpočtu se používá pixel a tří jeho sousedů. Jeho tvar je:

$$\left|\nabla f(i,j)\right| = \left| f(i,j) - f(i+1,j+1) \right| + \left| f(i,j+1) - f(i+1,j) \right|. \tag{18}$$

Výpočet určí velikost gradientu jako součet absolutních hodnot změn ve směru hlavní a vedlejší diagonály obrazu. Operátor detekuje především hrany se sklonem 45°.

Robertsův operátor je možné rozdělit na dvě složky, z nichž každá detekuje hrany v jednom ze dvou na sebe kolmých směrů.

Na podobném principu je založen i směrově orientovaný Sobelův operátor, který aproximuje první derivaci. Sobelův operátor je složen vždy z dvojice komplementárních konvolučních masek h a $\bar{h}$. Komplementární maska se získá z původní masky rotací o 90° kolem středu. Protože se v dalším výpočtu používá druhá mocnina nebo absolutní hodnota, nezáleží na tom, zda otáčíme doleva či doprava.

Příklad komplementárních konvolučních masek:

$$h = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \; \bar{h} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

Absolutní velikost gradientu se pak získá dvojnásobnou aplikací konvoluce, nejprve pro h a potom pro $\bar{h}$ a součtem:

$$|G| = \sqrt{h^2 + \bar{h}^2}\,. \qquad (19)$$

Součet se v praxi někdy zjednodušuje na:

$$|G| = |h| + |\bar{h}|\,. \qquad (20)$$

Na konvoluci je založen výpočet gradientu pomocí Laplaceova operátoru. Označuje se jako Δ a pro výpočet ze čtyřokolí pixelu ve směru kolmém na souřadnicové osy má jeho konvoluční jádro tvar:

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Varianta, která používá k výpočtu velikosti gradientu hodnot z osmiokolí, má tvar:

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Laplaceův operátor je invariantní k otáčení o násobky 45°.

Robertsův a Sobelův operátor aproximují výpočet první derivace, Laplaceův operátor je aproximací derivace druhé. Změny gradientu původní funkce se projeví jako lokální extrémy první derivace. Pokud provedeme druhou derivaci, extrémy odpovídají bodům, kde funkce prochází nulovou hodnotou. Proto se Laplaceův operátor označuje jako operátor průchodu nulou (zero-crossing operator). Hodnota, kterou tento operátor vypočítá, je aproximace druhé mocniny gradientu.

Gradient se odhaduje jako součet diferencí ve dvou na sebe kolmých směrech:

$$\Delta f(x,y) = \frac{\partial^2 f(x,y)}{\partial^2 x} + \frac{\partial^2 f(x,y)}{\partial^2 y} \qquad (21)$$

Laplaceův operátor obecně reaguje na hranu dvakrát, poprvé na její nástupné straně, podruhé na její straně sestupné. Výsledkem může být i negativní hodnota. V praxi se záporné hodnoty buď ořízne, celý rozsah se převede do kladných hodnot, a nebo, což je nejčastější, se použije absolutní hodnota.

Operátory zvýrazňující hrany zvýrazňují všechny vysoké frekvence, tedy i šum. Méně zvýrazňují šum operátory pracující s větší konvoluční maskou, tedy s větším okolím pixelu.

Zatímco konvoluční maska pro odstranění šumu měla vždy jako součet svých hodnot jedničku, pro detekci hran musí být součet hodnot roven nule. To zaručuje, že v oblastech s konstantní hodnotou bude i odezva konvoluce nulová.

2.5.1 Cannyho hranový detektor

Cannyho hranový detektor je vylepšený postup hledání hran v obraze. Tento detektor byl navržen podle následujících kritérií:

- maximalizace odstupu signál/šum
- co nejpřesnější nalezení hrany
- minimalizace vícenásobných odezev na jednu hranu

Jako první je vstupní obraz vyhlazen (potlačení šumu – čím větší je konvoluční maska pro vyhlazení, tím méně se uplatní šum ve vstupním obraze, na druhou stranu se zvětší odchylka nalezené hrany od hrany skutečné),např. Gaussovým filtrem. Dále je vypočítán gradient vyhlazeného obrazu ve směrech x a y, používá se např. Sobelův filtr. Z velikostí těchto dvou gradientů lze vypočítat směr hrany v bodě [x,y]jako:

$$\theta_{xy} = arctg\frac{Gy(x,y)}{Gx(x,y)}, \qquad (22)$$

kde θ_{xy} je směr gradientu v bodě [x,y],

Gx(x,y) je velikost gradientu v bodě [x,y] ve směru x,

Gy(x,y) je velikost gradientu v bodě [x,y] ve směru y.

Směr hrany se použije pro trasování hrany a potlačení všech pixelů, které nejsou maximem. Výsledkem je, že hrana v obraze je reprezentována tenkou linií. Jako poslední krok se provádí prahování s hysterezí. Pokud je velikost gradientu v bodě nižší, než spodní práh, je vyřazen z množiny hranových bodů, pokud je vyšší, než horní práh, je ponechán. Jestliže je hodnota gradientu v daném bodě mezi oběma prahy, je bod označen jako hranový pouze v případě, že existuje cesta z tohoto bodu do bodu s gradientem vyšším, než vyšší práh, jinak je bod vyřazen.

2.6 OSTŘENÍ OBRAZU

Lidské vnímání je založeno na rozpoznávání hran. Podle obrysu postavy je člověk například schopen rozeznat konkrétní osobu. Jednou z možností jak zvýraznit nějaký obraz tak, abychom ho vnímali jako ostřejší, je právě zvýraznění hran použitím hranových operátorů.

Ostření obrazu je založeno na následujícím postupu:

$$g(i,j) = f(i,j) + c.s(i,j), \tag{23}$$

kde g(i,j) je výsledný obraz,

f(i,j) je původní obraz

s(i,j) je funkce reprezentující velikost gradientu obrazu f v bodě [i,j]

c je ostřicí koeficient

Tento postup může být dvoukrokový, pokud nepoužíváme konvoluci. Je třeba nejprve vypočíst meziobraz s a ten poté přičíst k původnímu obrazu. Pokud použijeme konvoluci, lze detekci hran a jejich připočtení k původnímu obrazu sloučit do jediného kroku. Stačí všechny hodnoty konvolučního jádra vynásobit koeficientem ostření.

Ve frekvenční oblasti spočívá ostření obrazu v potlačení nízkých frekvencí a zdůraznění vysokých. Ideální vysokofrekvenční filtr propouští pouze vysoké frekvence a je popsán následujícím vztahem:

$$H(u,v) = \begin{cases} 0 & pro\ \sqrt{(u^2+v^2)} < D_0 \\ 1 & pro\ \sqrt{(u^2+v^2)} \geq D_0 \end{cases}, \tag{24}$$

kde $\sqrt{(u^2+v^2)}$ označuje vzdálenost frekvence od počátku a D_0 je mezní frekvence. Slovo ideální označuje fakt, že filtr nepropouští frekvence nižší než D_0.

2.7 HOUGHOVA TRANSFORMACE

Houghova transformace je metoda pro nalezení parametrického popisu objektů v obraze. Při aplikaci této metody je třeba znát analytický popis hledaných objektů. Klasická varianta byla určena pro detekci čar, ale metoda byla také modifikována pro hledání jiných tvarů.

Hlavní výhodou Houghovy transformace je robustnost proti porušením a nepravidelnostem objektu v obraze.

K nevýhodám patří výpočetní náročnost, která se řeší různými modifikacemi, některé z nich jsou zmíněny v kapitole 2.7.4.

2.7.1 Houghova transformace pro přímku

Pro hledání přímky pomocí Houghovy transformace se používá rovnice v normálovém tvaru (viz kapitola 3.2):

$$x\cos\Theta + y\sin\Theta = r \qquad (25)$$

tato rovnice tedy popisuje přímku pomocí dvou parametrů: r (vzdálenost přímky od počátku soustavy souřadnic) a Θ (úhel, který svírá osa x s kolmicí vztyčenou z počátku k přímce). Intervaly těchto parametrů jsou omezené pro všechny přímky a lze je volit dvěma způsoby:

1.) $\Theta \in \langle 0, 2\pi \rangle, r \in R$

2.) $\Theta \in \langle 0, \pi \rangle, r \geq 0$

(v reálném obraze je vzdálenost r omezená - na velikost úhlopříčky obrazu)

Prostor parametrů (r, Θ) je označován jako Houghův prostor *(Hough space)*, nebo Houghův akumulátor *(Hough accumulator)*.

Pokud bodem o souřadnicích $[x_0,y_0]$ bude procházet libovolné množství přímek, potom budou všechny splňovat rovnici (25) pro tento konkrétní bod:

$$x_0\cos\Theta + y_0\sin\Theta = r(\Theta) \qquad (26)$$

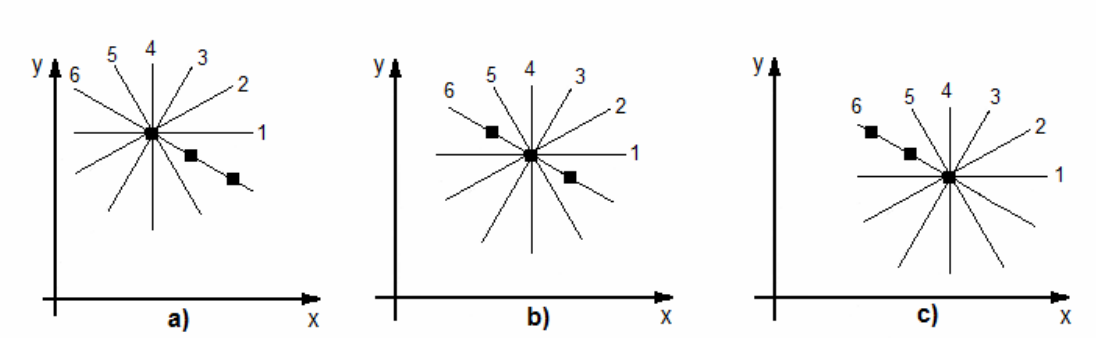
To v prostoru parametrů odpovídá sinusoidě, která je specifická právě pro tento bod. Množina bodů, ležící na přímce, potom v prostoru parametrů vytvoří množinu sinusoid, které se protnou v bodě, který odpovídá parametrům této přímky.

Houghův akumulátor má dimenzi rovnou počtu hledaných parametrů, v případě přímky je tedy dvourozměrný. Na začátku je na všech souřadnicích v akumulátoru stejná hodnota.

Vlastní transformace probíhá tak, že se systematicky prochází jednotlivé pixely vstupního obrazu a pokud je nalezen pixel náležející objektu, řeší se rovnice (26) pro všechny hodnoty parametru Θ, jejímž výsledkem je r. Poté se v akumulátoru na souřadnicích $[r, \Theta]$ zvýší hodnota. Po průchodu celým obrazem se v akumulátoru objeví maxima, která určují parametry nalezených objektů.

2.7.2 Příklad Houghovy transformace pro přímku

Pro jednoduchost budeme uvažovat pouze 3 body, znázorněné na Obr. 8 černými čtverečky.



Obr. 8 Příklad průběhu Houghovy transformace

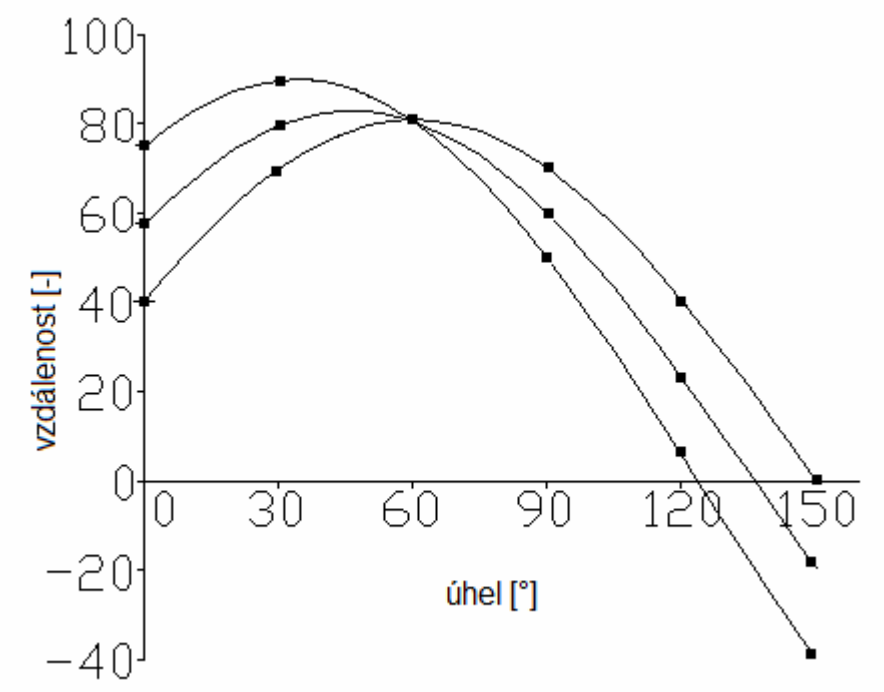
Postup transformace je následující:

- Každým uvažovaným bodem je proloženo několik přímek pod různými úhly. Počet přímek závisí na požadovaném úhlovém rozlišení.
- Pro každou z těchto přímek je určena kolmá vzdálenost od počátku soustavy souřadnic a úhel normály vzhledem k ose x. Výsledek této transformace je zaznamenán v Tab. 1.

přímka č.	úhel [°]	vzdálenost		
		a)	b)	c)
1	90	70,0	60,0	50,0
2	120	40,6	23,4	6,0
3	150	0,4	80,5	-39,6
4	0	40,0	57,1	74,6
5	30	69,6	79,5	89,6
6	60	81,2	80,5	80,6

Tab. 1 Výsledek transformace

- Grafické znázornění údajů – viz Graf 1.



Graf 1 Grafické znázornění výsledku HT

Z grafu je patrné, že Houghova transformace v tomto tvaru vytvoří v akumulátoru sinusoidy, z nichž každá reprezentuje průběh transformace jednoho bodu vstupního obrazu. Potom průsečík těchto sinusoid určuje parametry přímky, procházející zadanými body, v tomto případě se jedná o přímku označenou číslem 6. Jak je z řešení patrné, sinusoidy se neprotnou přesně v jediném bodě (důsledek omezené rozlišovací schopnosti v úhlu), ale vytvoří shluk, který parametry aproximuje.

Zde je jako výsledná vzdálenost použit průměr vzdáleností určených pro přímku označenou číslem 6:

$$x\cos 60° + y\sin 60° = 80{,}8\,,$$

což je výsledná rovnice nalezené přímky.

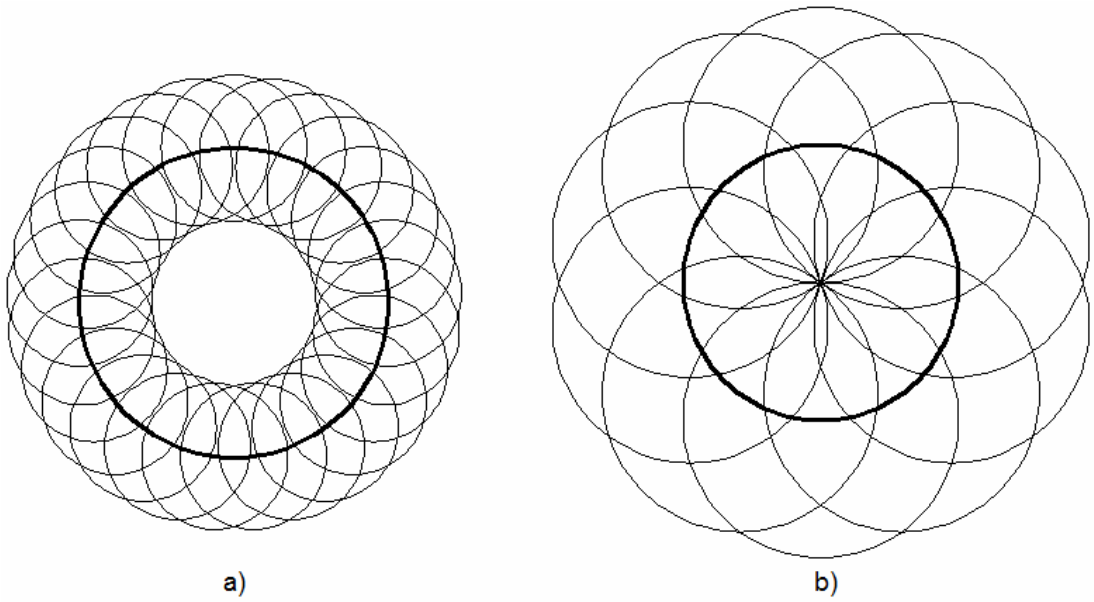
2.7.3 Houghova transformace pro kružnici

Pro hledání kružnice se používá její obecná rovnice (viz kap. 3.3):

$$(x-a)^2 + (y-b)^2 = r^2 \qquad (27)$$

Hledají se tedy tři parametry, souřadnice středu [a,b] a poloměr r, což znamená třírozměrný akumulátor a obecně vyšší výpočetní náročnost. Intervaly těchto parametrů se volí podle výchozích znalostí o hledaných objektech (velikost).

Příklad HT kružnice je uveden na Obr. 9. Zvýrazněná kružnice je objekt vstupního obrazu, tenké kružnice odpovídají řešením rovnice 27 pro jednotlivé body hledané kružnice. Obrázek znázorňuje pouze dva řezy trojrozměrným Houghovým akumulátorem (dva poloměry *r*).



Obr. 9 HT kružnice pro a) nesprávný poloměr, b) správný poloměr

2.7.4 Další varianty Houghovy transformace

Houghova transformace je často používána a proto má také mnoho variant, jejichž cílem je především snížit její výpočetní a časovou náročnost, zejména pro objekty, popsané více parametry (např. při hledání elipsy je třeba najít hodnoty pěti parametrů – střed $[x_0,y_0]$, délky hlavní a vedlejší poloosy a,b a úhel natočení θ).

- **Gradientní Houghova transformace (Gradient Hough transform)**

Využívá znalosti gradientu obrazu (může být k dispozici i jako výsledek předzpracování obrazu) k omezení počtu hlasů do Houghova akumulátoru pro každý bod. To má za následek snížení výpočetní i časové náročnosti a zvýraznění maxim v Houghově akumulátoru.

- **Pravděpodobnostní Houghova transformace (Probabilistic Hough transform)**

Snížení výpočetní náročnosti se dosahuje tak, že se neprovádí HT každého bodu vstupního obrazu, ale body jsou vybírány náhodným generátorem s rovnoměrným rozložením a jejich celkový počet je pouze určité procento z počtu bodů vstupního obrazu. Podle aplikace je doporučováno přibližně 5% – 15%. Tento postup odpovídá HT podvzorkovaného obrazu.

- **Náhodná Houghova transformace (Randomized Hough transform)**

Tato metoda náhodně vybírá n-tice bodů ze vstupního obrazu (kde n je určeno počtem hledaných parametrů) a provádí jejich HT. Počet opakování tohoto výběru je opět mnohem menší, než je počet bodů vstupního obrazu, čímž se sníží výpočetní náročnost.

- **Hierarchická Houghova transformace (Hierarchical Hough transform)**

Obraz je na počátku rozdělen na malé části. Na každou z těchto částí je aplikována HT pro přímku. Předpokládá se, že každá z částí obsahuje malý počet objektů. Tyto dílčí části jsou poté seskupovány v pyramidové struktuře tak, že každý uzel vyšší úrovně je tvořen například maticí 4x4 částí nižší úrovně. Uzel na nejvyšší úrovni pak reprezentuje celý vstupní obraz.

2.8 KNIHOVNA OPENCV

Knihovna Open Source Computer Vision Library je knihovna funkcí a několika tříd pro zpracování obrazu, založená na knihovně Image Processing Library firmy Intel. Je napsána jazyce C/C++ a není platformě závislá. Je poskytována zdarma pro komerční i nekomerční využití. Obsahuje několik stovek funkcí a tím vytváří rozhraní k programování aplikací pro zpracování obrazu.

Knihovna OpenCV se skládá ze čtyř částí:

- CV
  počítačové vidění (hranové operátory, matematická morfologie atd.)
- CVAUX
  experimentální vlastnosti
- CXCORE
  lineární algebra (maticové výpočty atd.)
- HIGHGUI
  podpora médií (načítání/ ukládání obrázků, jejich zobrazování, práce s videem atd.)

Pro ilustraci výkonnosti této knihovny lze uvést to , že byla použita i pro zpracování obrazu ve vítězném vozidle závodu Darpa Challenge 2005. V tomto závodě bylo úkolem plně autonomních vozidel projet předem neznámou trasu v těžkém pouštním terénu v co nejkratším čase. Jejich trasa byla zadána pouze orientačními body. Vítězné vozidlo Stanley Stanfordské univerzity projelo trasu dlouhou 132 mil (211 km) v čase 6 hodin 53 minut.

K výhodám patří i početná komunita (několik tisíc členů) na stránkách [12] kde lze najít mnoho rad a návodů pro použití této knihovny.

Momentálně je knihovna dostupná ve verzi 1.0, k dispozici jsou verze pro Windows a Linux.

Ukázka kódu využívajícího knihovnu OpenCV:

```
...
//Načtení obrazu
IplImage* vstupni_obraz = cvLoadImage(“obraz.jpg“, -1);

//kontrola načtení
if( vstupni_obraz== 0 )
 {
    //Obraz nebyl načten
    ...
}

//konverze obrazu na šedotónový
IplImage* vystupni_obraz;
cvCvtColor(vstupni_obraz,vystupni_obraz,CV_RGB2GRAY);

// zobrazení původního obrazu
cvNamedWindow("Original", 0);
cvShowImage("Original", vstupni_obraz);

//zobrazení výsledku
cvNamedWindow("Sedotonovy", 0);
cvShowImage("Sedotonovy",vystupni_obraz );
...
```

3. GEOMETRICKÉ ÚTVARY V ROVINĚ

3.1 BOD

Bod je bezrozměrný základní geometrický útvar. Všechny ostatní geometrické útvary jsou definovány jako množina bodů. Bod v rovině je určen dvěma souřadnicemi.

Jako bod v digitálním obraze je možné chápat shluk pixelů s určitými vlastnostmi, co se týče velikosti, popř. tvaru.

3.2 PŘÍMKA

Přímku v rovině lze popsat několika způsoby:

- **obecná rovnice**

$$ax + by + c = 0\,, \tag{28}$$

kde a,b jsou složky normálového vektoru přímky a $c=bA_y+aA_x$ (A_x, A_y je x-ová, resp. y-ová souřadnice bodu, ležícího na přímce).

- **parametrické rovnice**

$$\begin{aligned} x &= x_0 + at \\ y &= y_0 + bt, \quad t \in R, \end{aligned} \tag{29}$$

kde $[x_0,y_0]$ je libovolný bod náležející přímce,
a,b jsou složky směrového vektoru přímky
t je parametr

- **rovnice ve směrnicovém tvaru**

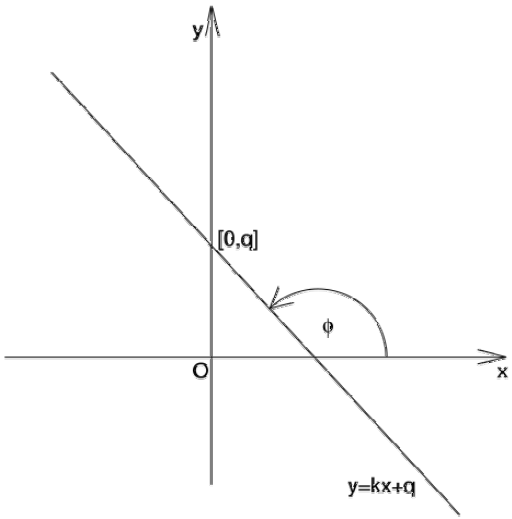
$$y = kx + q \tag{30}$$

kde k=tg φ je směrnice přímky

φ je orientovaný úhel s vrcholem v průsečíku přímky a osy x

q je úsek vyťatý přímkou na ose y (druhá souřadnice průsečíku s osou y).

Situace je znázorněna na Obr. 10. Rovnicí v tomto tvaru nelze vyjádřit přímku rovnoběžnou s osou y.



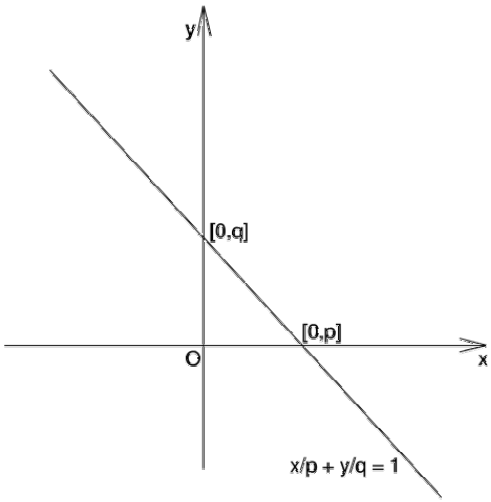
Obr. 10 Směrnicový tvar rovnice přímky

- **rovnice v úsekovém tvaru**

$$\frac{x}{p} + \frac{y}{q} = 1, \tag{31}$$

kde p≠0 je úsek vyťatý přímkou na ose x a q≠0 je úsek vyťatý přímkou na ose y.

Situace je znázorněna na Obr. 11. Rovnicí v úsekovém tvaru nelze vyjádřit přímku rovnoběžnou s některou ze souřadnicových os.



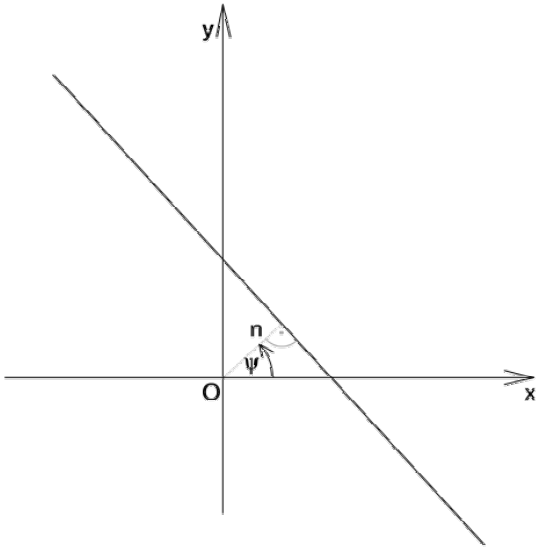
Obr. 11 Úsekový tvar rovnice přímky

- **rovnice v normálovém tvaru**

$$x\cos\Theta + y\sin\Theta = r \qquad (32)$$

kde r je vzdálenost přímky od počátku soustavy souřadnic

Θ je velikost orientovaného úhlu, jehož rameno je první kladná poloosa souřadné soustavy a druhé rameno je polopřímka s počátkem v *O* vedená kolmo k přímce. Situace je znázorněna na Obr. 12.



Obr. 12 Normálový tvar rovnice přímky

- **rovnice v polárních souřadnicích**

$$\rho = \frac{n}{\cos(\psi - \varphi)}, \tag{33}$$

kde n je vzdálenost přímky od počátku soustavy souřadnic

ψ je velikost orientovaného úhlu s vrcholem v počátku, jehož první rameno tvoří polární osa a druhé rameno polopřímka kolmá k přímce s počátkem v počátku soustavy souřadnic.

3.3 KRUŽNICE

Kružnici v rovině lze také popsat rovnicí ve více tvarech:

- **obecná rovnice**

$$(x-a)^2 + (y-b)^2 = r^2, \tag{34}$$

kde $[x_0,y_0]$ je střed kružnice

r je poloměr kružnice

- **vrcholová rovnice**

$$y^2 = 2rx - x^2, \tag{35}$$

kde $[r,0]$ je střed kružnice,

r je poloměr kružnice

- **parametrické rovnice**

$$\begin{aligned} x &= x_0 + r\cos\varphi \\ y &= y_0 + r\sin\varphi, \end{aligned} \tag{36}$$

kde $[x0,y0]$ je střed

φ je parametr, $\varphi \in \langle 0,2\pi \rangle$,

r je poloměr kružnice

- **rovnice v polárních souřadnicích**

$$\rho^2 - 2\rho\rho_0\cos(\varphi - \varphi_0) + \rho_0^2 = r^2, \tag{37}$$

kde $[\rho_0,\varphi_0]$ je střed kružnice,

r je poloměr kružnice

3.4 PRŮSEČÍKY KŘIVEK V ROVINĚ

- **Průnik dvou přímek v rovině**

Průsečík dvou přímek v rovině hledáme jako řešení soustavy dvou rovnic o dvou neznámých:

$$\begin{aligned} a_1 x + b_1 y + c_1 &= 0 \\ a_2 x + b_2 y + c_2 &= 0, \end{aligned} \tag{38}$$

Při řešení této soustavy mohou nastat tři případy:

- soustava nemá řešení (přímky jsou rovnoběžné)
- soustava má právě jedno řešení (hledaný průsečík)
- soustava má nekonečně mnoho řešení (přímky jsou totožné)

Analytické řešení této soustavy je:

$$\begin{aligned} x &= \frac{b_1 c_2 - c_1 b_2}{a_1 b_2 - a_2 b_1} \\ y &= \frac{a_2 c_1 - a_1 c_2}{a_1 b_2 - a_2 b_1} \end{aligned} \tag{39}$$

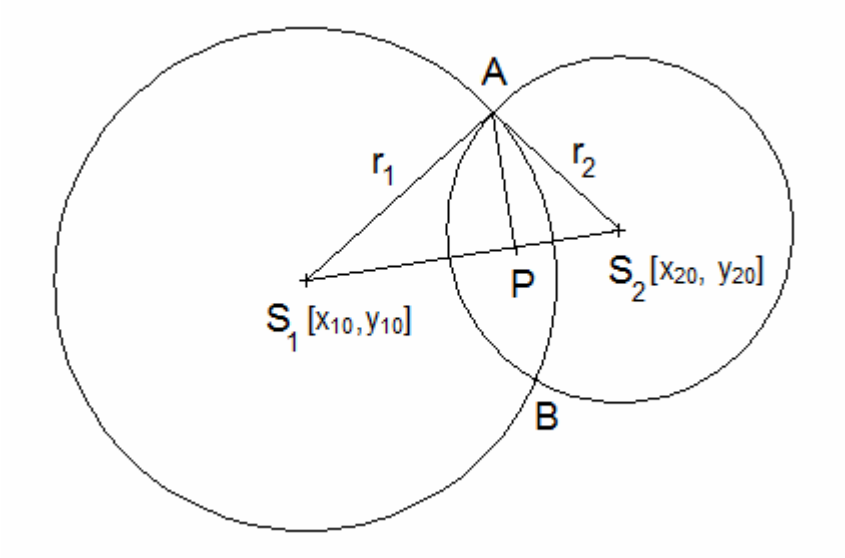
- **Průnik dvou kružnic v rovině**

Průsečíky dvou kružnic v rovině je možné vyjádřit obdobným způsobem jako v předešlém případě, tedy vyjádřením kružnic pomocí obecné rovnice kružnice a řešením soustavy dvou rovnic o dvou neznámých.:

$$\begin{aligned} (x - x_{10})^2 + (y - y_{10})^2 &= r_1^2 \\ (x - x_{20})^2 + (y - y_{20})^2 &= r_2^2 \end{aligned} \tag{40}$$

kde $[x_{10},y_{10}]$, $[x_{20},y_{20}]$ jsou souřadnice první, resp. druhé kružnice a r_1,r_2 je poloměr první, resp. druhé kružnice.

Analytické řešení této soustavy rovnic ovšem značně komplikuje fakt, že se jedná o rovnice kvadratické – tedy nelineární. Tuto soustavu rovnic lze ovšem snadno vyřešit graficky, situace je znázorněna na Obr. 13.



Obr. 13 Průnik dvou kružnic

Při řešení této soustavy rovnic mohou nastat čtyři případy:

($|S_1,S_2|$ značí Euklidovskou vzdálenost středů kružnic)

- soustava nemá řešení (pokud $|S_1,S_2|>r_1+r_2$ nebo $|S_1,S_2|<abs(r_1-r_2)$ kružnice se neprotínají)
- soustava má jedno (dvojnásobné) řešení (pokud $|S_1,S_2|=r_1+r_2$ nebo $|S_1,S_2|=abs(r_1-r_2)$, kružnice se dotýkají v jediném bodě)
- soustava má dvě různá řešení (pokud $|S_1,S_2|<r_1+r_2$ a zároveň $|S_1,S_2|=abs(r_1-r_2)$, kružnice se protínají ve dvou bodech),
- soustava má nekonečně mnoho řešení (pokud $|S_1,S_2|=0$ a zároveň $r1=r2$, kružnice jsou totožné)

Postup řešení:

Vypočítáme vzdálenost středů kružnic:

$$|S_1,S_2| = d = \sqrt{(x_{20}-x_{10})^2+(y_{20}-y_{10})^2}\ . \qquad (41)$$

Podle vypočtené vzdálenosti stanovíme počet průsečíků (podle výše uvedeného rozboru), pokud se kružnice neprotínají, výpočet končí. Jestliže mají kružnice pouze jeden společný bod, potom se postup dalšího výpočtu nemění, v závěru vypočtené dva body budou identické a stačí tedy uvažovat jako výsledek jeden z nich.

Bod P rozdělí úsečku S_1S_2 na dva úseky o délkách $m=|S_1P|$a $n=|PS_2|$ $(d=m+n)$, pro které platí:

$$\begin{aligned} r_1^2 &= |AP|^2 + m^2 \\ r_2^2 &= |AP|^2 + n^2 \end{aligned} \qquad (42)$$

po odečtení obou rovnic a úpravě dostáváme:

$$m = \frac{r_1^2 - r_2^2 + d^2}{2d}, \qquad (43)$$

souřadnice bodu P tedy jsou:

$$P = \left[x_{10} + \frac{m}{d}(x_{20} - x_{10}), y_{10} + \frac{m}{d}(y_{20} - y_{10}) \right]. \qquad (44)$$

Nyní vypočítáme délku úsečky AP:

$$|AP| = v = \sqrt{r_1^2 - m^2}\ , \qquad (45)$$

pro souřadnice průsečíků A,B potom platí:

$$\begin{aligned} A &= \left[P_x + \frac{v}{d}(y_{10} - y_{20}), P_y + \frac{v}{d}(x_{10} - x_{20}) \right] \\ B &= \left[P_x - \frac{v}{d}(y_{10} - y_{20}), P_y - \frac{v}{d}(x_{10} - x_{20}) \right] \end{aligned}, \qquad (46)$$

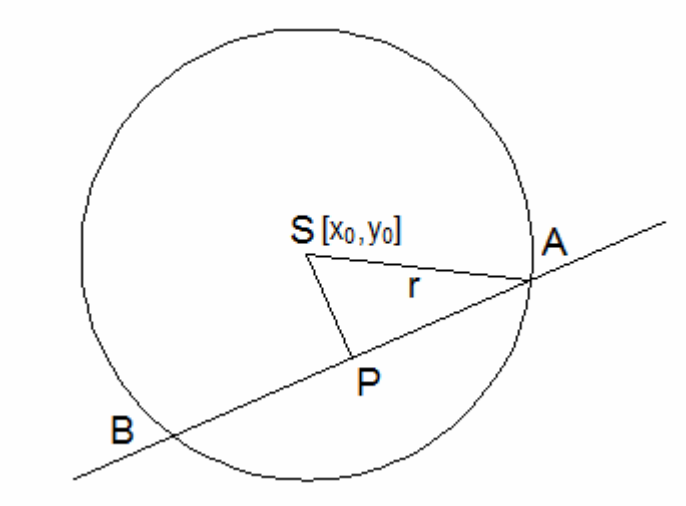
kde P_x, P_y značí x-ovou, resp. y-ovou souřadnici bodu P.

- **Průnik přímky a kružnice v rovině**

Hledání průsečíků přímky a kružnice lze opět vyjádřit soustavou dvou rovnic o dvou neznámých:

$$\begin{aligned} ax + by + c &= 0 \\ (x - x_0)^2 + (y - y_0)^2 &= r^2 \end{aligned} \qquad (47)$$

Analytické řešení této rovnice je opět komplikováno nelineární rovnicí kružnice a proto je lepší tuto soustavu řešit graficky, viz Obr. 14.



Obr. 14 Průnik přímky a kružnice

Při řešení této soustavy rovnic mohou nastat tři případy:

- soustava nemá řešení (pokud $|SP|>r$, přímka kružnici neprotíná)
- soustava má jedno (dvojité) řešení (pokud $|SP|=r$, přímka se kružnice dotýká v jediném bodě – tečna kružnice)
- soustava má dvě různá řešení (pokud $|SP|<r$, přímka kružnici protíná ve dvou bodech – sečna kružnice)

Postup řešeni:

Normálový vektor *u* zadané přímky je:

$u = [a, b]$.

Tento vektor je zároveň směrovým vektorem přímky *p*, na které leží úsečka *SP*. Rovnice přímky *p* procházející bodem $S[x_0,y_0]$ je:

$$bx - ay + y_0 a - bx_0 = 0 . \qquad (48)$$

Bod *P* určíme jako průsečík dvou přímek – přímky *p* a zadané přímky (viz výše).

Dále vypočítáme délku úsečky $|SP|$:

$$|SP| = d = \sqrt{(P_x - x_0)^2 + (P_y - y_0)^2} , \qquad (49)$$

kde P_x, P_y značí x-ovou, resp. y-ovou souřadnici bodu *P*.

Pro vzdálenost $|PA|$ potom platí:

$$|PA|^2 = v = r^2 - d^2, \tag{50}$$

a souřadnice průsečíků A,B jsou:

$$\begin{aligned} A &= \left[P_x + v\frac{-b}{|u|}, P_y + v\frac{a}{|u|} \right] \\ B &= \left[P_x - v\frac{-b}{|u|}, P_y - v\frac{a}{|u|} \right] \end{aligned}, \tag{51}$$

kde $|u|$ je velikost vektoru u, tedy

$$|u| = \sqrt{a^2 + b^2}, \tag{52}$$

a P_x, P_y značí x-ovou, resp. y-ovou souřadnici bodu P.

4. XML

XML (*eXtensible Markup Language – rozšiřitelný značkovací jazyk)* je v současné době asi nejvýznamnějším představitelem značkovacích jazyků. Tento jazyk je odvozen z obecného jazyka SGML.

Zdrojový text značkovacího jazyka obsahuje kromě vlastního textu i instrukce pro jeho zpracování, které mají formu tzv. značek *(tags)*. Na rozdíl například od HTML neobsahuje XML předdefinované značky.

XML je určen pro výměnu dat a publikování dokumentů. Popisuje dokument z hlediska jeho obsahu a sám o sobě nedefinuje formátování dokumentu. Vzhled výsledného dokumentu určuje až připojený styl. Pomocí stylů lze dokument i převést na jiný typ.

4.1 SYNTAXE

XML dokument je textový soubor, vždy v kódování Unicode, standardní je UTF-8, ale jsou možná i jiná kódování. Syntaxe XML kódu je na první pohled podobná HTML, je ovšem přísnější.

Dokument se skládá z *elementů*, které se v textu vyznačují *tagy*. Většina elementů označena dvěma tagy – počátečním a ukončovacím. Tagy se zapisují mezi znaky „<“ a „>“, např.:<tag>. Ukončovací tag má před svým názvem znak „/“, např.: </tag>. Elementy, které nemají žádný obsah je možné zapsat standardně jako
</br> nebo kratším zápisem
. Dokument musí obsahovat pro každý tag i ukončovací tag (pokud není element zapsán jako prázdný). Elementy v dokumentu se nesmí křížit.

U počátečního tagu je možné použít ještě atributy, které upřesňují význam elementu, atributy se uzavírají do uvozovek, např.: <tag barva=“červená“>.

Každý XML dokument musí být celý uzavřen v jednom elementu. Pokud XML dokument dodržuje všechny výše uvedené zásady, je tzv. správně strukturovaný *(well-formed)*. Takovýto dokument by měly být schopny zpracovat všechny aplikace s podporou XML.

4.2 PŘÍKLAD XML KÓDU

```
<?xml version="1.1" encoding="UTF-8"?>
<objekty>
     <kružnice>
          <střed>
               <x>10</x>
               <y>10</y>
          </střed>
          <poloměr>2</poloměr>
     </kružnice>
</objekty>
```

XML je velice vhodný prostředek pro strukturované ukládání informací, protože společně s informaci ukládá i její význam.

Specifikace jazyka XML (vytvořeného konsorciem W3C) je volně přístupná (viz [6]).

Podrobné informace v českém jazyce lze nalézt například v knize [3].

5. SOUČASNÝ STAV

Pro detekci nejrůznějších objektů v obraze existuje řada postupů, založených na různých principech. Pro následné stanovení parametrů objektů (rozměrů) je výhodné, pokud výstupem metody je analytický popis (parametry) tohoto objektu.

Dále jsou uvedeny odlišné přístupy pro detekci objektů, které využívají apriorní informaci o tvaru i velikosti objektů.

5.1 DETEKCE OBDÉLNÍKŮ POMOCÍ HOUGHOVY TRANSFORMACE [4]

Tato práce se zabývá detekcí objektů obdélníkového tvaru v obraze. Navržený postup pracuje tak, že na základě apriorní informace o rozměrech hledaných objektů se stanoví okolí, ve kterém se pomocí Houghovy transformace vyhledají úsečky, které mohou tvořit hrany hledaného obdélníku. Poté je zkoumáno, zda parametry nalezený úseček splňují jisté geometrické podmínky a formují tedy obdélník. Toto zkoumání se provádí přímo v Houghově prostoru. Pro vyloučení duplicit je používána chybová funkce, pracující s chybami určení pravých úhlů, vzdálenosti a rovnoběžnosti nalezených přímek. Jako výsledek hledání se pak bere obdélník s nejmenší hodnotou chyby.

5.2 SEGMENTACE BUNĚK POMOCÍ GENETICKÝCH ALGORITMŮ [5]

Základem práce je využití apriorní informace o tvaru hledaných objektů, které jsou elipsovitého tvaru. Po předzpracování obrazu – nalezení hranic objektů v obraze – je inicializován genetický algoritmus, jehož jedinci jsou tvořeni kruhy, případně elipsami. Tito jedinci jsou vyhodnocováni podle toho, jak dobře aproximují objekt v obraze. Výstupem jsou rovnice elips nebo kružnic, které se vyskytují v obraze. Pro zkvalitnění detekce je použita i apriorní informace o velikosti hledaných objektů.

5.3 ALGORITMUS UPWRITE [8]

V této práci je provedeno srovnání různých variant Houghovy transformace s novějším algoritmem, nazvaným UpWrite.

Tento algoritmus s skládá ze tří částí. Nejprve se obraz rozdělí na malé části a pro každou z nich se stanoví tzv. lokální model. Nalezené lokální modely se poté použijí k nalezení oblastí obrazu, které patří k jednomu objektu. Nakonec se pro takto získaný model objektu vypočítají geometrické momenty, které zařadí objekt do některé z předem definovaných tříd.

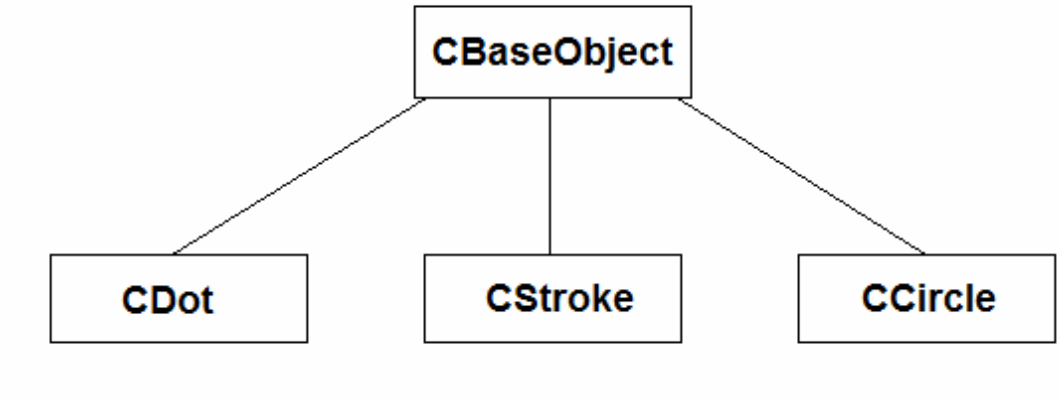
Dosažené výsledky ukazují, že algoritmus UpWrite je oproti Houghově transformaci více odolný proti šumu. Při použití vstupních obrazů bez šumu dosahovaly obě metody pro jednoduché objekty (úsečky, kružnice) srovnatelných výsledků. Pro složitější objekty (elipsy) dával UpWrite přesnější výsledky. Výhodou algoritmu UpWrite oproti Houghově transformaci je také to, že nepotřebuje apriorní analytický popis hledaného objektu.

6. NAVRŽENÉ PROSTŘEDÍ

6.1 OBJEKTOVÝ NÁVRH APLIKACE

V aplikaci je třeba spravovat více typů grafických objektů. V současné době jsou implementovány tři typy grafických objektů – body, úsečky a kružnice.

Aby tyto různé grafické objekty mohly být spravovány v jediné kolekci, byla vytvořena abstraktní základní třídu CBaseObject, která abstrahuje společné rozhraní těchto tří typů grafických objektů. Od této základní třídy jsou poté odvozeny třídy pro konkrétní typy grafických objektů, viz Obr. 15.



Obr. 15 Objektová hierarchie aplikace

Cílem návrhu těchto tříd bylo, aby byly co nejvíce samostatné, tj. nezávislé na zbylé části aplikace. Tohoto cíle bylo dosaženo tak, že třídy poskytují nejen metody pro nastavování svých parametrů, ale i metody pro vykreslení objektu, upřesnění objektu, modifikaci objektu nebo získání informací o objektu ve tvaru řetězce zformátovaného XML. Například metoda *Draw* vykreslí objekt na předaný kontext zařízení a hlavní aplikace tak při vykreslování vůbec nepotřebuje znát typy vykreslovaných objektů.

Implementace nejdůležitějších metod jsou podrobněji popsány v následujících kapitolách.

Díky této architektuře je možné do aplikace velice jednoduše začlenit další typy grafických objektů. Je potřebné pouze odvodit od základní třídy *CBaseObject* další třídu pro konkrétní typ grafického objektu, přidat kód pro hledání společných bodů nového typu objektu a ostatních typů objektů a přidat nabídku nového typu grafického objektu do rozhraní aplikace.

6.2 UŽIVATELSKÉ ROZHRANÍ GRAFICKÝCH OBJEKTŮ

Grafické objekty jsou zadávány pomocí myši. V případě úsečky *CStroke* určí uživatel její počáteční a koncový bod. V případě kružnice *CCircle* zadává uživatel dva body, které leží na průměru kružnice. Bod *CDot* je určen samozřejmě pouze jedním bodem.

Z hlediska větší budoucí rozšiřitelnosti by bylo možné metodu zadávání přepracovat tak, aby objekt pro nastavení svého tvaru nepřijímal pouze dva body, ale kolekci bodů. Bylo by tak například možné zadat trojúhelník třemi body, popř. určit více bodů a tyto body by poté objekt aproximoval hladkou křivkou apod.

Při výběru nakreslených objektů pomocí myši je nutné tolerovat určitou nepřesnost, se kterou uživatel objekt vybírá. Nelze se spolehnout na to, že uživatel označí vždy přímo bod daného objektu. Tento problém lze řešit například tak, že je stanoveno okolí kolem místa, kam uživatel kliknul a vyhledají se objekty, které do tohoto okolí zasahují. Další možností řešení je, že každý grafický objekt vymezuje sám oblast svého okolí a stačí tedy pro každý objekt určit, zda bod kliknutí leží v jeho oblasti.

Byla zvolena druhá možnost, která poskytuje větší možnosti, protože okolí každého může být určeno daným objektem individuálně. Ukázka oblastí, definovaných grafickými objekty je na Obr. 16.

Obr. 16 Oblasti definované jednotlivými typy objektů (šedé podbarvení)

V této aplikaci je výhodou zvoleného přístupu také to, že oblast vymezená objektem je využita i při upřesňování jeho tvaru (viz kap. 6.5).

Při výběru objektu může samozřejmě nastat i situace, kdy zvolený bod náleží do oblasti více objektů. Proto je výběr objektu řešen tak, že všechny objekty, do jejichž oblasti zvolený bod náleží jsou soustředěny do dočasné kolekce. Pokud tato kolekce po průchodu všemi objekty obsahuje pouze jeden prvek, je tento ihned vybrán. Jinak je první objekt v této kolekci zvýrazněn barvou odlišnou od barvy běžného výběru, což uživateli signalizuje, že existuje více možností pro výběr. Uživatel má poté možnost procházet jednotlivé objekty v dočasné kolekci pomocí pravého tlačítka myši, přičemž aktuální objekt se zvýrazní. Volbu konkrétního objektu uživatel potvrdí levým tlačítkem myši. Zvolený objekt je poté vybrán a dočasná kolekce objektů je vyprázdněna.

6.3 MODIFIKACE GRAFICKÝCH OBJEKTŮ

Aplikace umožňuje i základní modifikace umístěných grafických objektů jako jejich přesun, rotace a změna velikosti. Tyto operace jsou implementovány v metodách *Move*, *Rotate* a *ChangeSize* tříd, reprezentujících jednotlivé objekty.

Některé z těchto operací nemají pro určité typy grafických objektů smysl (například rotace nebo změna velikosti bodu *CDot*), ale jejich alespoň prázdná implementace je vyžadována společným rozhraním tříd grafických objektů.

6.4 SEGMENTACE OBRAZU

Segmentace obrazu má v tomto případě za cíl detekovat hrany objektů v obraze pro účely upřesňování tvaru objektů a díky použití knihovny OpenCV je její implementace poměrně jednoduchá.

Vstupní obraz je nejprve z barevného převeden na šedotónový.

Dále může být volitelně vyhlazen Gaussovým filtrem, popř. může být aplikováno morfologické otevření pro potlačení šumu.

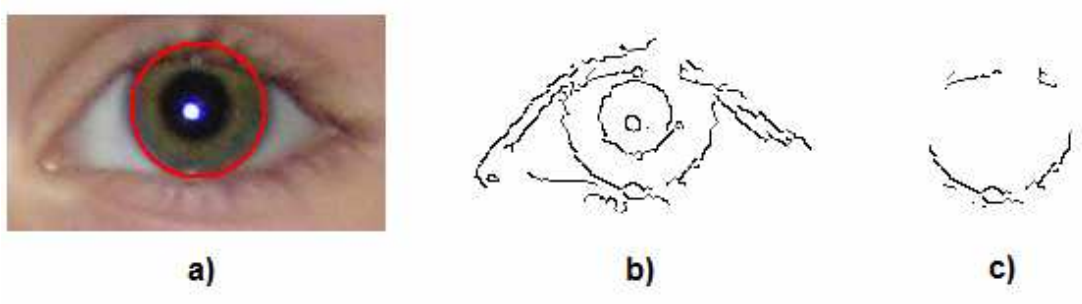
Pro detekci hran byl zvolen Cannyho hranový detektor, jehož parametry (prahy) je z důvodu variability vstupních obrazů možné uživatelsky nastavovat přímo na panelu nástrojů aplikace, viz kap. 6.9. Pro správný výsledek segmentace je důležité nutné vhodné nastavení Cannyho detektoru.

6.5 UPŘESŇOVÁNÍ ROZMĚRŮ A POZICE OBJEKTŮ

Upřesňování objektů je řešeno tak, že třída každého objektu implementuje metodu Refine, jejímiž vstupními argumenty jsou výřez segmentovaného obrazu, který je vytvořen na základě oblasti konkrétního objektu, zdrojový obraz a pozice výřezu v originálním obraze. Pro upřesnění přímek *CStroke* i kružnic *CCircle* byla zvolena Houghova transformace, bod *CDot* upřesňován není.

6.5.1 Vytvoření výřezu pro upřesnění

Pro upřesnění konkrétního objektu je funkcí *CObjektyView::Refin*e vytvořen na základě oblasti tohoto objektu výřez segmentovaného obrazu, který vstupuje do funkce *Refine* příslušného objektu. Výřez je vytvořen tak, že je stanoven ohraničující obdélník (*bounding rect*) oblasti objektu a do tohoto obdélníku jsou zkopírovány pouze ty body segmentovaného obrazu, které náleží do oblasti objektu, viz Obr. 17.



Obr. 17 Tvorba výřezu obrazu pro upřesnění objektu: a) vstupní obraz s uživatelem vyznačeným objektem, b) segmentovaný obraz, c) výřez ze segmentovaného obrazu s využitím oblasti objektu

6.5.2 Metoda CStroke::Refine

Úsečka je upřesněna pomocí standardní varianty Houghovy transformace pro přímku, která je dostupná funkcí cvHoughLines2 z knihovny OpenCV. Výstupem této funkce je sekvence dvojic parametrů r a Θ rovnic nalezených přímek ve tvaru (viz kapitola 2.7.1):

$$x\cos\Theta + y\sin\Theta = r \tag{53}$$

Parametry jsou seřazeny podle příslušné hodnoty Houghova akumulátoru a do dalšího zpracování vstupuje pouze první z nich, tzn. přímka s parametry příslušejícími nejvyšší hodnotě v Houghově akumulátoru.

Podle rovnice (53) lze odvodit parametrickou rovnici této přímky:

$$\begin{aligned} x &= x_0 + bt \\ y &= y_0 + at \end{aligned}, \tag{54}$$

kde

$$\begin{aligned} a &= \cos\Theta \\ b &= \sin\Theta \\ x_0 &= ar \\ y_0 &= br \end{aligned}.$$

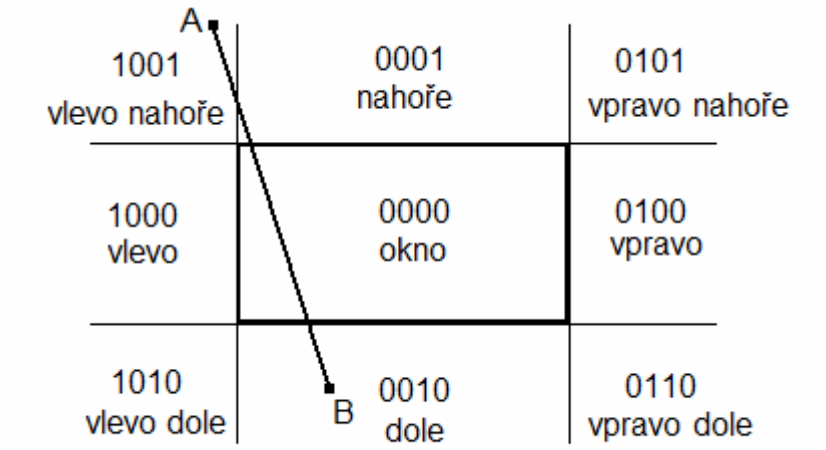
Nalezená přímka samozřejmě neobsahuje koncové body úsečky v obraze. Tyto body je nutné najít jiným způsobem. Jedna z možností je například okno sčítající hodnoty pixelů, posouvané po přímce. Koncové body se poté projeví jako nárůst resp. pokles načítané hodnoty, tato metoda je použita např. v [8].

V nynější práci byl zvolen odlišný přístup – úsečka je pouze ořezána rozměry zdrojového obrazu. Uživatel poté může koncové body úsečky najít jako průsečíky s jinými geometrickými útvary a přímku podle nich ořezat. Tento přístup se jevil vhodnější například pro hledání rohů.

Ořezání přímky rozměry obrazu je prováděno tak, že na základě parametrické rovnice přímky jsou vypočteny počáteční a koncový bod úsečky pro velkou hodnotu parametru *t* a oba body tak leží mimo obraz. Dále popsaný algoritmus pak zajistí ořezání přímky.

Algoritmus Cohen-Sutherland

Tento algoritmus slouží k ořezávání úseček oknem, jehož hrany jsou rovnoběžné s osami souřadného systému. Nejprve jsou koncové body úsečky ohodnoceny tzv. *hraničními kódy*. Jednotlivé bity těchto kódů popisují polohu bodu vzhledem k oknu, viz Obr. 18.



Obr. 18 Kódy oblastí vymezených oknem

V zobrazeném příkladu je kód bodu A kód(A)=1001 a kód bodu B kód(B)= 0010.

Pro jakoukoliv úsečku AB mohou nastat tři možnosti

a) $kód(A) \cup kód(B) = 0$

- celá úsečka leží v okně, ořezání není potřeba.

b) $kód(A) \cap kód(B) \neq 0$

- celá úsečka leží mimo okno a lze ji vyřadit z dalšího zpracování.

c) $kód(A) \cap kód(B) = 0$

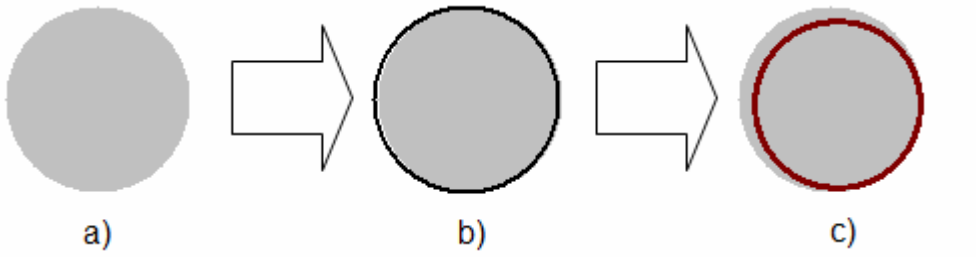
- úsečka prochází více oblastmi a bude ořezána. To je i případ úsečky AB na Obr. 18. Vybere se koncový bod s neprázdným hraničním kódem a podle jednoho nastaveného bitu je koncový bod upraven – jedna souřadnice se nastaví na příslušnou souřadnici hranice okna, druhou souřadnici je nutné dopočítat jako průsečík úsečky a příslušné hranice okna. S upravenými body s novými hraničními kódy se opakují předchozí testy.

Další algoritmy pro ořezávání grafických objektů jsou popsány např. v [2].

Na závěr procesu upřesňování je vypočítán vzájemný úhel zadané a nalezené úsečky a pokud je menší než tolerance (nyní 30°), jsou počáteční a koncový bod úsečky nastaveny na hodnoty upřesněné úsečky.

6.5.3 Metoda CCircle::Refine

Pro upřesnění kružnice byla původně použita funkce cvHoughCircles z knihovny OpenCV, která provádí Houghovu transformaci (HT) pro kružnici přímo z šedotónového obrazu za použití gradientní metody. Výsledky použití této funkce ovšem nebyly v některých případech uspokojivé viz Obr. 19.



Obr. 19 Nepřesnost při použití funkce cvHoughCircles: a) vstupní obraz, b) obrys vyznačený uživatelem, c) chybně upřesněný obrys

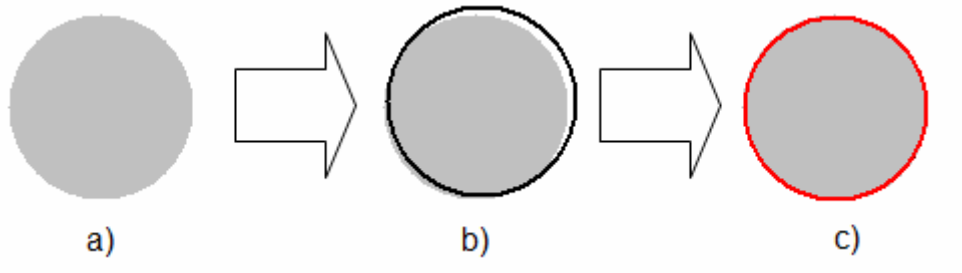
Nahlédnutím do zdrojového kódu této funkce se ukázalo, že tento je vcelku nepřehledný a málo srozumitelný a že odhalení, a eventuální odstranění příčiny této nepřesnosti by bylo velmi náročné.

Z těchto důvodů byla pro HT kružnice vytvořena vlastní implementace.

Pro implementaci byla vybrána prostá metoda HT pro kružnici, zejména z důvodu jednoduchosti a průhlednosti její implementace.

Pro snížení paměťových nároků je HT implementována tak, že se cyklicky provádí HT pro konkrétní poloměr (jeden řez 3D Houghovým akumulátorem) a vyhodnocuje se hodnota maxima (*peak*) akumulátoru. Pokud je toto maximum větší než současně uložené maximum, je jeho hodnota uložena společně s jeho souřadnicemi a aktuálním poloměrem. Na konci cyklu tak máme uloženy souřadnice, na kterých se v akumulátoru vyskytlo maximum, tzn. nalezené parametry kružnice. Pokud maximum v akumulátoru přesáhlo nastavený práh, jsou parametry objektu poté nastaveny na nalezené parametry.

Ukázka výsledku aplikace tohoto algoritmu je na Obr. 20.

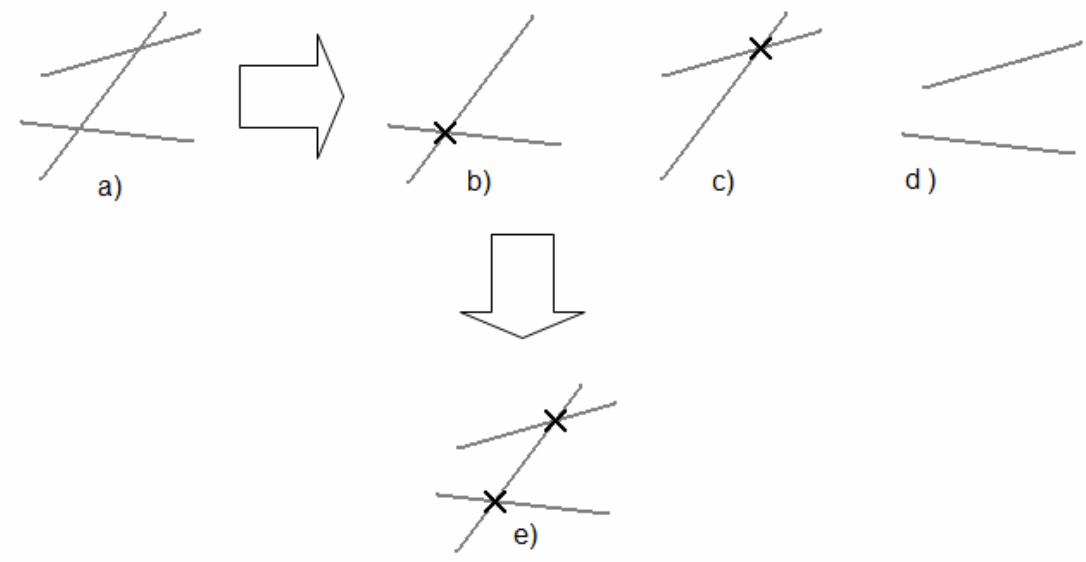


Obr. 20 Ukázka použití vlastní implementace HT: a) vstupní obraz, b) obrys vyznačený uživatelem, c) upřesněný obrys

6.6 HLEDÁNÍ PRŮSEČÍKŮ GRAFICKÝCH OBJEKTŮ

Protože uživatel může pro hledání průsečíků zadat více objektů než dva, jsou z vybraných objektů vytvořeny všechny možné dvojice bez opakování a na tyto dvojice jsou poté aplikovány algoritmy hledání společných bodů podle typu grafických objektů obsažených ve dvojici, viz Obr. 21.

Pro nyní použité typy grafických objektů mohou nastat tři případy, jejichž rozbor je proveden v kapitole 3.4.



Obr. 21 Hledání průsečíků grafických objektů: a) objekty, jejichž průsečíky mají být nalezeny, b), c), d) vytvořené dvojice objektů s označenými průsečíky, e) výsledek hledání

Protože v matematickém rozboru jsou uvažovány přímky, u kterých průsečík existuje vždy, když jsou různoběžné a vytvořená aplikace pracuje s úsečkami, u kterých toto neplatí, probíhá v případě úseček hledání průsečíků tak, že nejprve jsou stanoveny průsečíky pro přímky a výsledné body jsou otestovány, zda leží na obou úsečkách (v případě průsečíku úsečky a kružnice na jediné úsečce). Body jsou onačeny za průsečíky a přidány do seznamu nalezených objektů pouze tehdy, pokud toto kritérium splní.

Funkce pro hledání průsečíků úseček je využita i při konstrukci kolmice k vybrané přímce – u uživatelem zvolené přímky je určen její normálový vektor a ten společně s aktuální pozicí kurzoru myši určuje polopřímku kolmou k zadané úsečce. Tato polopřímka je poté zkrácena na úsečku v bodě průsečíku těchto dvou grafických objektů.

6.7 OŘEZÁNÍ GRAFICKÉHO OBJEKTU JINÝM OBJEKTEM

Pro ořezání objektu jiným objektem je výhodné použít již navrženou metodu pro hledání průsečíků grafických objektů.

Výše uvedená implementace hledání průsečíků objektů byla tedy modifikována tak, že pokud je jí předán jako parametr ukazatel na pole bodů, jsou nalezené průsečíky uloženy do tohoto pole a nikoliv do kolekce nalezených objektů. Nalezené průsečíky potom vstupují do metody *Trim* objektu, který má být ořezán.

6.7.1 CStroke::Trim

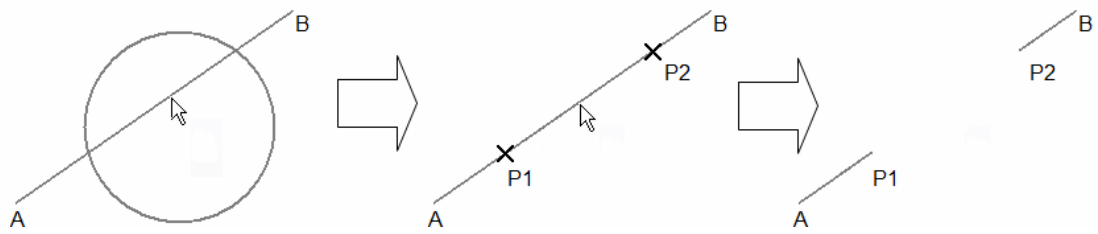
Do metody vstupuje pole nalezených průsečíků a uživatelem zvolený bod, určující, která část objektu se bude ořezávat.

Nejprve je třeba určit, kterou část úsečky je třeba odstranit. Proto je vypočítána vzdálenost mezi počátkem úsečky a bodem pro ořezání. Pokud je tato vzdálenost větší, než vzdálenost mezi počátkem úsečky a bode, který zvolil uživatel, potom je počáteční bod úsečky nastaven na souřadnice ořezávajícího průsečíku, jinak je na tyto souřadnice nastaven koncový bod úsečky.

Pokud jsou ořezávající průsečíky dva (např. ořezávání úsečky kružnicí) a uživatel zvolil k ořezání část mezi jedním z nich a koncovým bodem, je situace obdobná jako v předchozím případě, přičemž je uvažován pouze jeden průsečík (bližší zvolenému bodu).

Pokud uživatel zvolil bod mezi oběma průsečíky, je zřejmé, že původní úsečka se rozdělí na dvě. Toho je dosaženo tak, že souřadnice současné úsečky se nastaví tak, že jeden z jejích koncových bodů zůstane původní a druhý je nastaven na souřadnice průsečíku. Poté je vytvořen nový objekt typu *CStroke*, jehož koncové body jsou nastaveny na druhý koncový bod původní úsečky a druhý z průsečíků. Ukazatel na nově vytvořený objekt je funkcí *Trim* vrácen a volající funkce jej zařadí do kolekce nalezených objektů.

Postup ořezání úsečky dvěma body je znázorněn na Obr. 22 (šipka označuje místo kliknutí uživatelem).



Obr. 22 Ořezání úsečky dvěma body: a) úsečka AB a ořezávající kružnice, b) úsečka AB s vyznačenými ořezávajícími průsečíky, c) nově vzniklé úsečky

6.7.2 CCircle::Trim

Vstupní parametry jsou samozřejmě shodné s předchozím případem. U kružnice je uvažována pouze situace, kdy ořezávající průsečíky jsou dva.

Třída CCircle obsahuje členské proměnné ArcStart a ArcEnd, které označují počátek a konec kruhového oblouku. Při vytváření objektu jsou obě tyto proměnné nastaveny na stejnou hodnotu a kružnice je tedy úplná.

Při prvním ořezání jsou tyto hodnoty nastaveny na hodnoty ořezávajících průsečíků.

Při následném ořezávání takto vytvořeného kruhového oblouku se objekt rozdělí na dva. Počáteční a koncové body oblouků jsou nastaveny podle umístění ořezávajících průsečíků a bodu zvoleného uživatelem.

Ukazatel na nově vytvořený objekt je opět funkcí *Trim* vrácen a volající funkce jej zařadí do kolekce nalezených objektů.

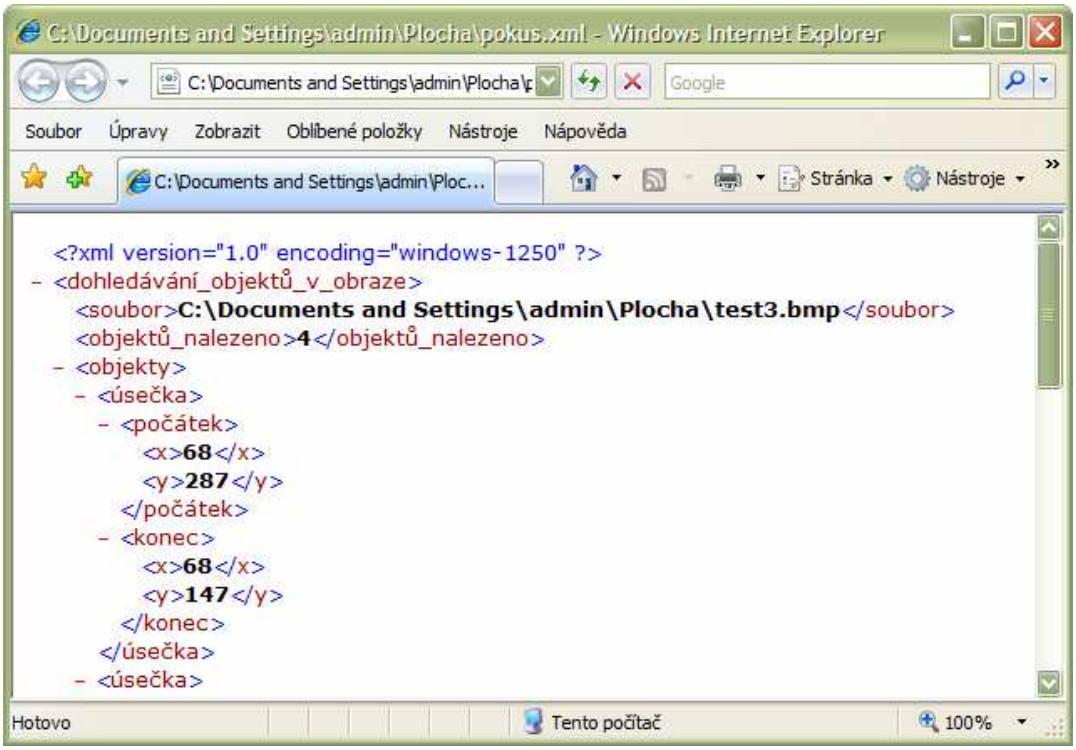
6.8 UKLÁDÁNÍ DAT

Pro ukládání výstupních dat programu jsem zvolil univerzální formát XML, který umožňuje snadnou přenositelnost dat, viz kap. 4. Uložení dat v tomto formátu také umožňuje bezproblémové další zpracování v jakékoli jiné aplikaci s podporou XML (pro případnou navazující vyšší úroveň zpracování).

Pro ukládání dat ve formátu XML byla vytvořena třída *CXmlData*. Tato třída obsahuje pouze jedinou metodu – *CXmlData::WriteToFile*, jejímž vstupním parametrem je ukazatel na objekt dokumentu. Pomocí tohoto ukazatele je získán přístup ke kolekci nalezených objektů. Pro každý objekt je poté volána metoda *FormatXMLString*, která vrací textový řetězec, obsahující parametry objektu,

naformátovaný pomocí XML. Všechny tyto řetězce jsou spolu s XML záhlavím a zápatím uloženy do souboru. Výhodou tohoto přístupu je úplná nezávislost ukládání informací na typech implementovaných objektů. Do aplikace lze přidat další typ grafického objektu bez jakéhokoliv zásahu do kódu pro ukládání parametrů.

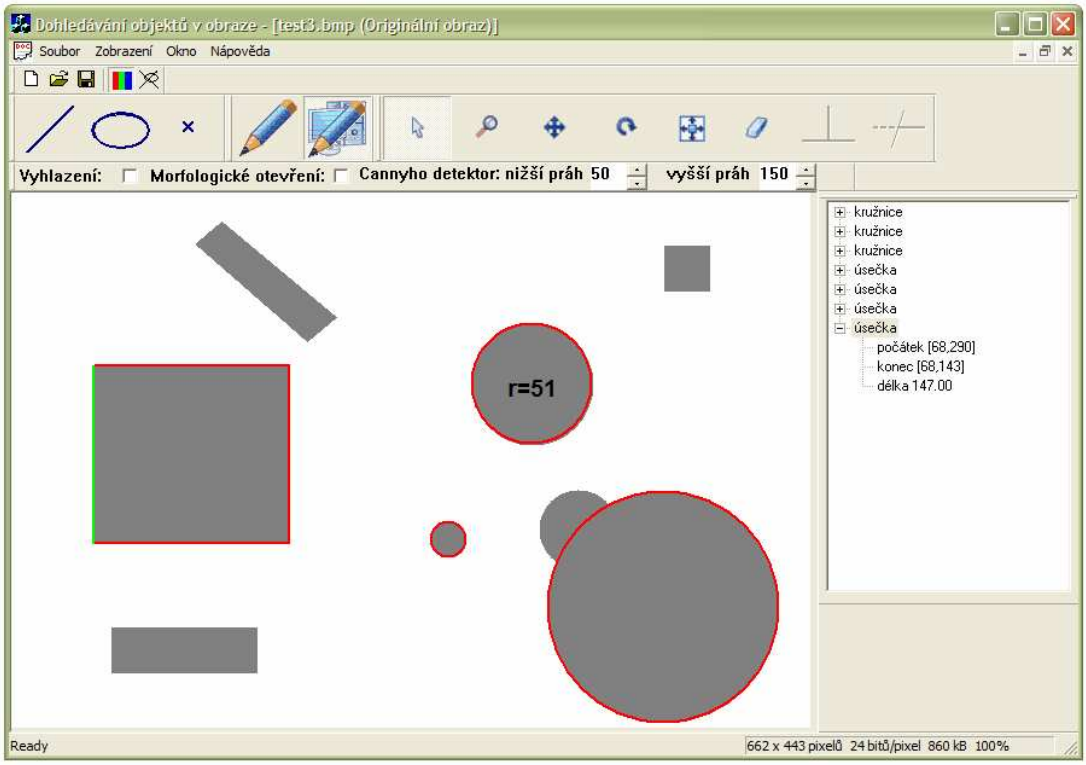
Výstup uložený programem lze zobrazit pomocí jakéhokoliv textového prohlížeče nebo editoru, popř. v hierarchickém znázornění například pomocí Windows Exploreru, viz Obr. 23.



Obr. 23 Výstup aplikace zobrazený ve Windows Internet Exploreru

6.9 POPIS OVLÁDÁNÍ APLIKACE

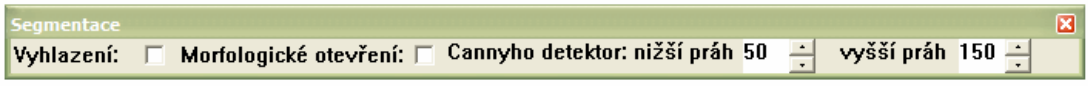
Uživatelské rozhraní aplikace je na Obr. 24.



Obr. 24 Uživatelské rozhraní vytvořené aplikace

Načtení vstupního obrazu se provede volbou Otevřít z nabídky Soubor, popř. tlačítkem na hlavním panelu nástrojů. Formát vstupního obrazu může být BMP, JPEG nebo TIFF.Načtený obraz je možné zobrazit jak původní podobě tlačítkem , tak i jako segmentovaný tlačítkem . Měřítko zobrazení lze měnit kolečkem myši v rozsahu 10% až 1000%.

Parametry segmentace se nastavují na panelu nástrojů Segmentace. Po změně některého z těchto parametrů je obraz znovu segmentován s novým nastavením.



Volba režimu, ve kterém program pracuje se provádí na panelu nástrojů Režim. Změna režimu mění vybraný nástroj na nástroj Výběr.

 manuální režim (bez upřesňování)

 režim s upřesňováním – po nakreslení objektu je volána jeho metoda Refine a objekt je tedy automaticky upřesněn

Výběr typu objektu se provádí na panelu nástrojů Kreslení. Kreslení objektu se zahajuje kliknutím levého tlačítka myši, druhým kliknutím se ukončí. Kreslení je možné v jeho průběhu přerušit pravým tlačítkem myši.

 úsečka

 kružnice

 bod – kreslí se jediným kliknutím myši

Na panelu nástrojů Nástroje jsou dostupné volby pro výběr a modifikace objektů. Nevybraný objekt je zobrazen červenou barvou, vybraný zelenou. Pokud je při výběru objektu více možností (v blízkosti místa kliknutí leží více objektů), je první možný objekt zvýrazněn (fialová barva). Mezi jednotlivými objekty je poté možné přepínat pravým tlačítkem myši. Vybraný objekt se potvrdí levým tlačítkem myši.

 výběr – jednoduchý výběr (vybrán může být pouze jeden objekt)

 hledání průsečíků – levým tlačítkem myši se provede výběr všech potřebných objektů a poté pravým tlačítkem se výběr potvrdí a jsou vyhledány jejich průsečíky

Přesun, rotace a změna velikosti objektu se zahájí kliknutím levého tlačítka myši na požadovaný objekt a druhým kliknutím se ukončí. Operaci je v jejím průběhu možné přerušit pravým tlačítkem myši.

přesun objektu – referenční bod je určen místem kliknutí na objekt

rotace objektu – střed otáčení je určen místem kliknutí na objekt

změna velikosti objektu – v případě úsečky je třeba pro změnu velikosti kliknout v blízkosti koncového bodu úsečky

vymazání objektu

hledání kolmice k úsečce – volba je dostupná pouze tehdy, jestliže je jednoduchým výběrem vybrána úsečka, ke které se má kolmice konstruovat. Tuto volbu je možné použít i v režimu s upřesňováním, kdy se úsečka při přemísťování průběžně upřesňuje podle svého okolí. Operaci je možné ukončit kliknutím pravého tlačítka myši nebo výběrem jiného nástroje.

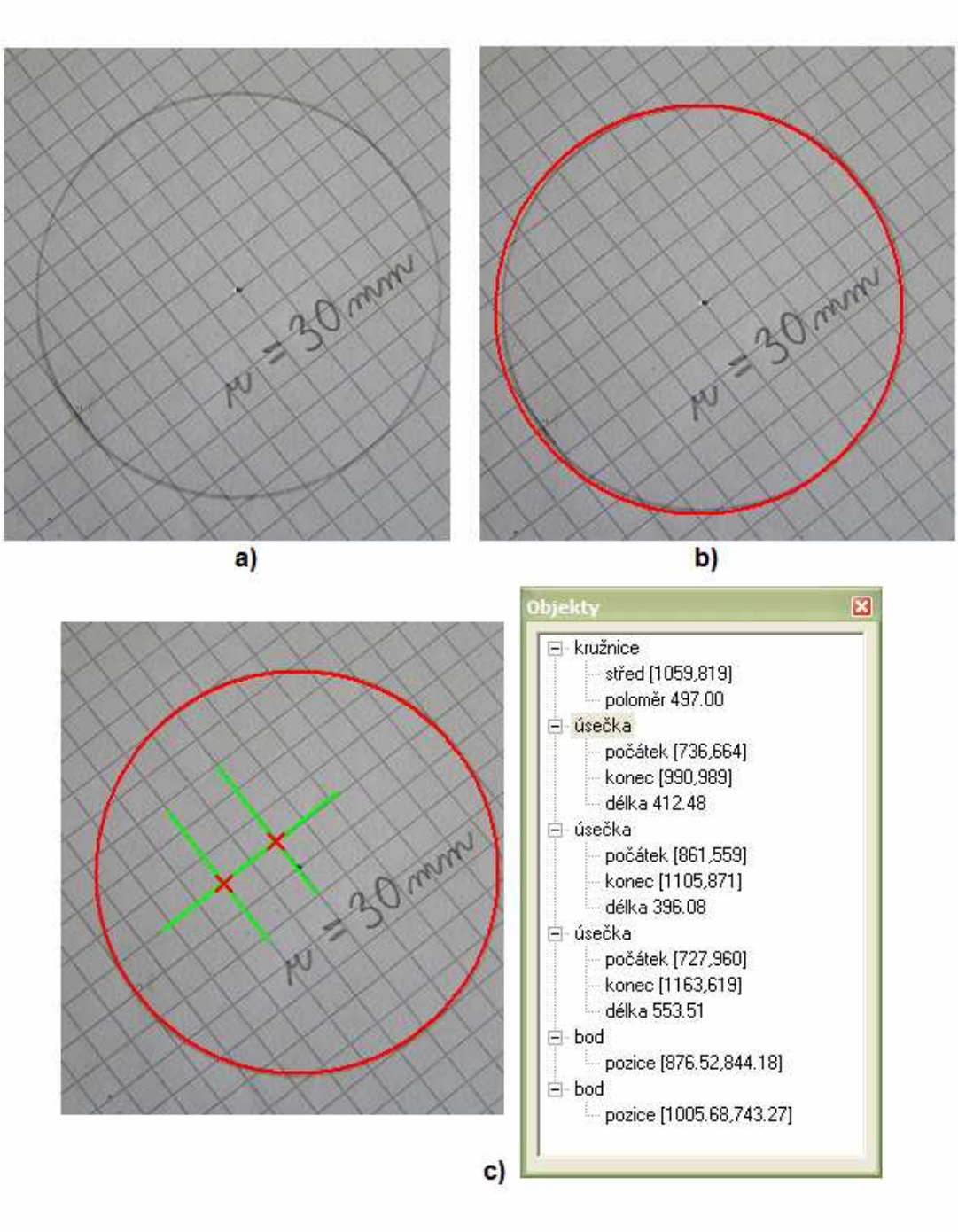
ořezání objektu – volba je dostupná pouze tehdy, jestliže je vybrán objekt, podle kterého se bude ořezávat. Ořezává se ta část objektu, na kterou uživatel klikne levým tlačítkem myši.

Parametry objektů se zobrazují na panelu Objekty. Po vybrání objektu v seznamu je tento objekt vybrán i v obraze.

Výstup programu je možné uložit ve formátu XML volbou Uložit z nabídky Soubor.

6.10 UKÁZKA POUŽITÍ APLIKACE

Aplikaci je možné využít například pro měření rozměrů objektů, viz Obr. 25.



Obr. 25 Ukázka měření rozměrů poměrovou metodou: a) vstupní obraz, b) obrys vyznačený uživatelem, c) upřesněný obrys a stanovení dvou referenčních bodů

Poloměr kružnice v obraze byl upřesněn na:

$r_{OBRAZ} = 497\,pixelů$.

Pro stanovení skutečných rozměrů využijeme referenční body, které byly nalezeny jako průsečíky vybraných úseček upřesněných podle mřížky v obraze (úsečky byly pro lepší přehlednost zkráceny).

U referenčních bodů známe vzdálenost v obraze d_{OBRAZ}, i skutečnou vzdálenost d_{REAL}:

$$d_{OBRAZ} = \sqrt{(1005{,}68 - 876{,}52)^2 + (743{,}27 - 844{,}18)^2} = 163{,}9\,pixelů$$
$$d_{REAL} = 10mm$$

Poměr těchto vzdáleností tedy je:

$n = 16{,}39\,pixelů / mm$

Tímto poměrem vydělíme nalezený poloměr kružnice a získáme tak její skutečné rozměry:

$$r_{REAL} = \frac{r_{OBRAZ}}{n} = 30{,}3mm.$$

Výhodou je usnadnění zadávání měřených objektů jejich automatickým upřesňováním, tzn. uživatel nemusí objekty označovat příliš pečlivě, viz Obr. 25.

6.11 OMEZENÍ

Se zvolenou koncepcí souvisí i některá omezení. Uživatel musí objekt zadat tak, aby alespoň přibližně odpovídal objektu v obraze, resp. aby se dostatečně velká část obrazových bodů dostala do výřezu obrazu, ve kterém se objekt upřesňuje. Tento problém by bylo možné obejít zvětšením oblasti objektu, ovšem to by přineslo komplikace jako například častější mnohoznačnost při výběru konkrétního objektu a také by se prodloužila doba potřebná pro upřesnění objektu, zvláště patrné by to bylo například u kružnice.

Obrazová data reprezentovaná zvolenými tvary mohou vcelku dobře popisovat jednodušší objekty v obraze, jako například siluetu domu, jeho okna, dveře apod. Tyto jednoduché objekty nejsou vhodné pro popis složitějších objektů (přírodní scény).

7. ZÁVĚR

V této práci byl proveden návrh prostředí pro zadávání obrazových dat. Jsou zde shrnuty základní teoretické poznatky a popsány navržené algoritmy.

Navržená aplikace pracuje v současné době se třemi typy grafických objektů – body, přímkami a kružnicemi. Objektový návrh aplikace umožňuje snadné přidávání dalších typů grafických objektů. Grafické objekty podporují základní možnosti editace, jako přesun, změnu velikosti, rotaci a ořezání pomocí jiného objektu.

Aplikace může pracovat ve dvou režimech – v manuálním režimu a v režimu s upřesňováním, kdy je objekt zadaný uživatelem automaticky upřesněn podle obsahu obrazu.

Výstup aplikace – parametry grafických objektů – je v univerzálním formátu XML.

Byly navrženy algoritmy, které upřesňují uživatelem zadávané objekty. Úspěšnost upřesnění objektu závisí na charakteru obrazu a nastavení parametrů segmentace. Některé z těchto parametrů může uživatel ovlivňovat, toto řešení bylo zvoleno z důvodu variability možných vstupních obrazů. Uživatel může tyto parametry nastavovat na základě vizuálního zhodnocení při zobrazení segmentovaného obrazu.

Aplikace může být využita například při kontrole tvaru a rozměrů objektů v obraze, kdy uživateli automatickým upřesňováním objektů usnadňuje práci

8. POUŽITÉ INFORMAČNÍ ZDROJE

[1] Hlaváč, V., Hlaváček, M. Zpracování signálu a obrazu. Skriptum FEL ČVUT, Praha 2000

[2] Žára, J., Beneš, B.,Sochor, J., Felkel, P. Moderní počítačová grafika. Computer Press, Praha 2004. ISBN 80-251-0454-0

[3] Kosek, J. XML pro každého. Grada Publishing, Praha 2000. ISBN 80-7169-860-1

[4] Jung, C. R., Schramm, R. Rectangle Detection based on a Windowed Hough Transform [on-line], UNISINOS - Universidade do Vale do Rio dos Sinos. Dostupné na Internetu: <ieeexplore.ieee.org/iel5/9360/29722/01352951.pdf>

[5] Křivánková, S. Segmentace buněk v biomedicínských obrazech pomocí genetických algoritmů [on-line], Fakulta informatiky Masarykovy univerzity v Brně. Dotupné na Internetu: <hilbert.chtf.stuba.sk/KUZV/download/kuzv-krivankova.pdf>

[6] World Wide Web consortium [on-line]. Dostupné na Internetu: <www.w3c.org>

[7] Houghova transformace. Wikipedia, The Free Encyclopedia [on-line]. Dostupné na Internetu: < http://en.wikipedia.org/wiki/Hough_transform>

[8] McLaughlin, R. A., Alder, M. D. The Hough Transform Versus the UpWrite, IEEE Transactions on pattern analysis ans machine intelligence, vol. 20, no. 4, april 1998 [on-line]. Dostupné na Internetu: <historical.ncstrl.org/litesite-data/uwa_ee/jp98-02.ps.gz>

[9] Prata, S. Mistrovství v C++. Computer Press, Praha 2001. ISBN 80-7226-339-0

[10] Prosise, J. Programování ve Windows pomocí MFC. Computer Press, Praha 2000. ISBN 80-7226-309-9

[11] Dokumentace ke knihovně OpenCV [on-line]. Dostupné na Internetu: <opencvlibrary.sourceforge.net>

[12] Open Source Computer Vision Library Community [on-line]. Dostupné na Internetu: <tech.groups.yahoo.com/group/OpenCV>