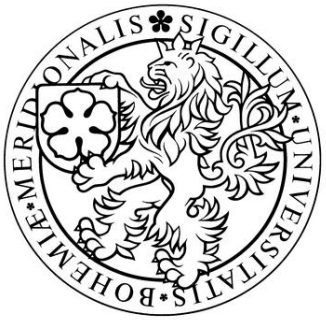
Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta



Knihovna pro automatickou detekci a měření objektů v obrazu z mikroskopu

Bakalářská práce

Lukáš Fessl

Školitel: Ing. Miroslav Skrbek, Ph.D.

České Budějovice 2012

Bibliografické údaje

Lukáš Fessl, 2012: Knihovna pro automatickou detekci a měření objektů v obrazu z mikroskopu.

[Library for automatic detection and measurement objects in the microscope pictures]
- 43 p. (počet stran), Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací programové knihovny v jazyce Java pro identifikaci a měření rozměrů živočichů na obrázcích z mikroskopu. Pro rozpoznávání a měření je využito Houghovo transformace. Dále jsou zde popsány dílčí kroky pro zpracování obrazu, jako je předzpracování, segmentace a rozpoznávání. Pro tyto jednotlivé kroky jsou zde vysvětleny některé metody, které se používají v těchto oblastech zpracování obrazu.

Abstract

The thesis deals with proposal and implementation of software library in programming language Java for identification and measurement of dimensions animals in the pictures from microscope. Hough transformation is used for identification and measurement of dimensions. Individual steps for image processing , like is preprocessing, segmentation and detection are described. Some methods, which are used in these areas at image processing are explained in individual steps.

Klíčová slova:

Zpracování obrazu, předzpracování, segmentace, Houghovo transformace

Keywords:

Image processing, preprocessing, segmentation, Hough transformation

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb., v platném znění, souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě, elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejich internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb., zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích, dne 27.4.2012 Podpis……

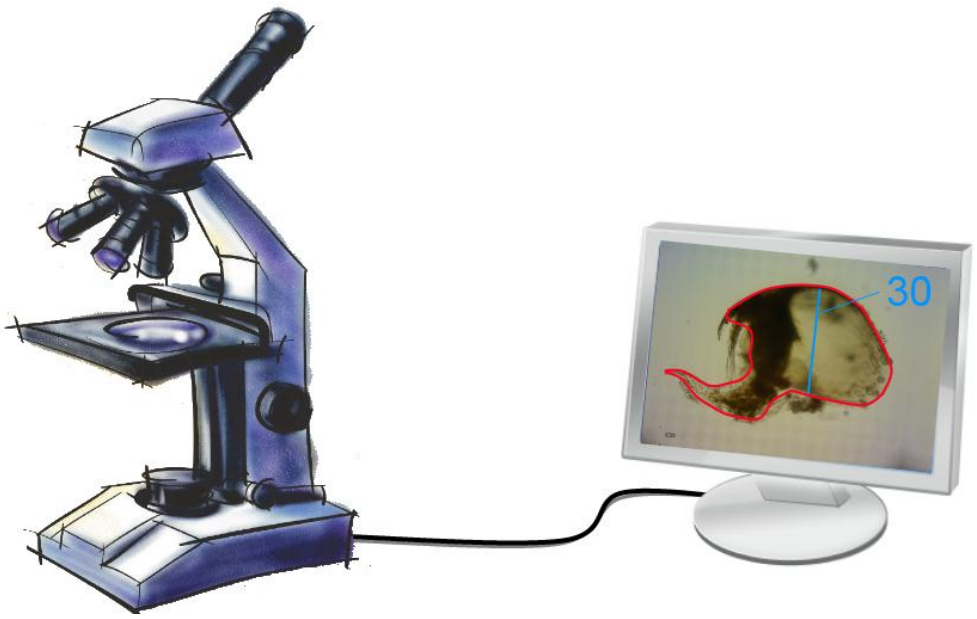
Poděkování

Děkuji panu Ing. Miroslavu Skrbkovi, Ph.D. za odborné vedení a pomoc při zpracování bakalářské práce a panu Mgr. Davidu Žaloudkovi za pomoc při překladu anglické části bakalářské práce.

Obsah

1 Úvod

Práce se zabývá návrhem a implementací aplikace v jazyce Java, pro identifikaci a měření rozměrů mikroskopických organismů na obrázcích z mikroskopu. Pro měření a rozpoznávání je využita Houghova transformace. Lze si také nadefinovat vlastní vzorové organismy a hledané vzdálenosti, podle potřeb uživatele. Tato aplikace je vytvářena pro potřeby biologického oddělení Přírodovědecké fakulty, která potřebuje aplikaci pro již zmíněné účely.



Obrázek 1. Schéma

Zpracování obrazu bylo vyvinuto v reakci na problémy spojené se zpracováním obrázků, které by mělo řešit digitalizaci obrázku a jeho kódování k usnadnění přenosu, tisku nebo uchování. V dnešní době zpracování obrazu pojednává především o zpracování digitálních obrázků, které své uplatnění nalézá v oblasti počítačového vidění robotiky [1].

Zpracování obrazových mikroskopických dat spadá až do poloviny 20. století, když si lidé uvědomili, že některá z technik zachytávání obrazu a jeho manipulace, poprvé vytvořeno pro televizi, mohla také být aplikována na obrázky zachycené pomocí mikroskopu.

Jeden z nejstarších přístupů pro zpracování mikroskopických obrázků byla metoda CYDAC (CYtophotometric DAta Conversation). Tato metoda funguje tak, že obraz byl

skenován digitalizován přímo z mikroskopu. Analýza těchto dat vzhledem k tehdejší době byla velice pomalá a složitá. Přístup k počítači byl velice omezený, jen na pár hodin v týdnu a samotný pokrok v této oblasti byl tedy velice pomalý. Až s příchodem programovatelných digitálních počítačů, byla odstraněna většina omezení. Nakonec tedy výzva neležela v samotném zpracování obrazu, ale v jeho správném a efektivním zpracování [2].

Dnes se zpracováním obrazu zabývá velké množství knihoven a frameworků, z nichž jsou některé volně dostupné včetně zdrojových kódů. Asi nejznámější volně dostupné knihovny jsou OpenCV, ITK a VTK. Knihovna ITK se zaměřuje na předzpracování a segmentaci obrazu, především v lékařství. Knihovna VTK pracuje s vizualizací dat a 3D grafikou. Obě knihovny se tak doplňují a lze je jednoduše propojit i díky tomu, že knihovny jsou vyvíjeny jednou firmou Kitware a obsahují velké množství společných prvků v architektuře. Obě knihovny jsou psány v jazyce C++ již více než 20 let [14]. OpenCV je knihovna zaměřená na zpracování obrazu a počítačové viděni. Knihovna obsahuje spoustu optimalizovaných algoritmů a je dostupná zdarma i pro komerční použití. Lze s ní pracovat v jazycích C, C++ a Python [15].

2 Cíl práce

Cílem práce je navrhnout a realizovat programovou knihovnu pro identifikaci a měření živočichů na obrázcích z mikroskopu. Provést rešerši metod pro zpracování obrazu, identifikaci objektů a měření v obraze. Po dohodě s vedoucím práce vybrat vhodné metody pro řešení problému, které budou implementovány ve formě knihovny a funkčnost knihovny demonstrovat na jednoduché aplikaci, která bude implementována v jazyce Java.

3 Zpracování obrazu

Zpracování obrazu je vědecko-technická disciplína, která se zabývá zpracováním digitálních obrazových dat, jejíž cílem je rozpoznání a porozumění obrazové informaci z reálného světa, která je nějakým způsobem zachycena. Postup zpracování a rozpoznávání obrazu lze obvykle rozložit do několika částí, z nichž nás bude zajím především předzpracování, segmentace a rozpoznávání obrazu [3].

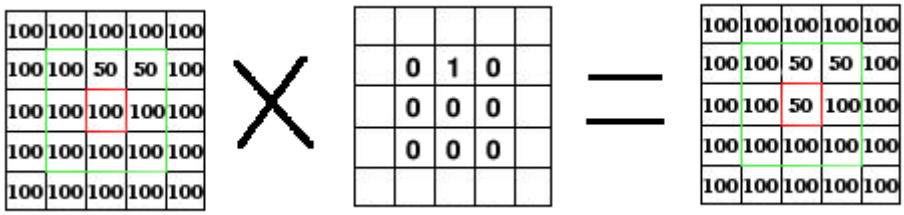
3.1 Předzpracování obrazu

Předzpracování je proces, při kterém dochází k úpravě obrazových dat. Při tomto procesu dochází k odstranění nežádoucích vlivů, jako je například šum a pomocí dalších metod může být obrázek upraven tak, aby byl snáze použitelný pro segmentaci. Bez provedení předzpracování by mohl i v krajním případě být obrázek s velkým šumem a dalšími nežádoucími vlivy nepoužitelný pro segmentaci [4].

3.1.1 Gaussův filtr

Gaussův filtr je jedním z nejúčinnějších filtrů, které se používají. Gaussův filtr lépe eliminuje šum pro větší použité masky, ale na druhou stranu při větších maskách Gaussova filtru více rozmazává obraz, což se stává hlavní nevýhodou, protože rozmazává i hrany, což může být problém pří následné detekci hran. Prudce také stoupá výpočetní náročnost rozmazání obrazu pomocí tohoto filtru. Rozmazání obrazu Gaussovým filtrem můžeme provést následujícími způsoby – konvolucí, pomocí Fourierovy transformace a rekurzivními filtry.

Konvoluce je operace s maticí pomocí jiné matice zvané jádro. Jako první matice se používá obrázek určený k úpravě. Obrázek je vlastně dvojrozměrná pravoúhlá síť souřadnicová síť pixelů a použité jádro závisí na požadovaném efektu. Obvykle se používá matice 3x3 dostačující pro všechny požadované efekty. Filtr postupně zpracovává všechny pixely obrázku. Pro každý z nich vynásobí hodnoty aktuálního pixelu a jeho osmi sousedních pixelů odpovídajícími hodnotami jádra. Výsledné hodnoty pak sečte a výsledek je hodnotou přiřazenou aktuálnímu pixelu [5]. Na následujícím obrázku je vidět jednoduchý příklad.



Obrázek 2. Aplikace Gaussova filtru [5]

Časová náročnost výpočtu konvoluce roste přímo úměrně s druhou mocninou poloměru gaussova filtru. Naštěstí konvoluční jádro Gaussova filtru je separabilní. To znamená, že jednu konvoluci s dvourozměrným jádrem lze rozdělit na dvě konvoluce s jednorozměrnými jádry. Dojde tak k redukci nárůstů časové složitosti z kvadratické na lineární. Konvoluce je nejjednodušší a pro dostatečně malé poloměry také nejrychlejší způsob filtrování obrazu pomocí Gaussova filtru. Podrobný popis konvoluce můžete nalézt zde [13].

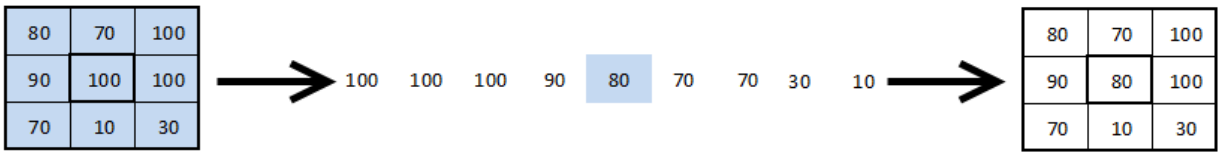
Jedním z možných způsobů, jak urychlit výpočet u větších poloměrů filtru, je Fourierovo transformace. Tato metoda je založena na faktu, že konvoluci lze převést na násobení a jelikož operace násobení má výpočetní složitost téměř stejnou pro malá i velká konvoluční jádra, neroste výpočetní doba tolik, jako v případě konvoluce.

Druhým, dnes patrně nejpoužívanějším způsobem aplikace Gaussova filtru, je využití rekurzivních filtrů. Základní myšlenka spočívá v tom, že se nepřepočítává celá konvoluce pro každý pixel, ale pro výpočet hodnoty nějakého bodu se využije již spočítaných hodnot z bodů předcházejících. V případě implementace rekurzivního filtru na výpočet Gaussova filtru je vše o dost složitější, neboť se nejedná homogení konvoluční jádro jako v případě průměrování. Při posunutí koncolučního jádra dojde k posunutí i všech jeho koeficientů a sčítance už nesedí, tak jako seděli u průměrovacího jádra. Právě z tohoto důvodu není návrh jakéhokoliv rekurzivího filtru jednoduchou záležitostí [6].

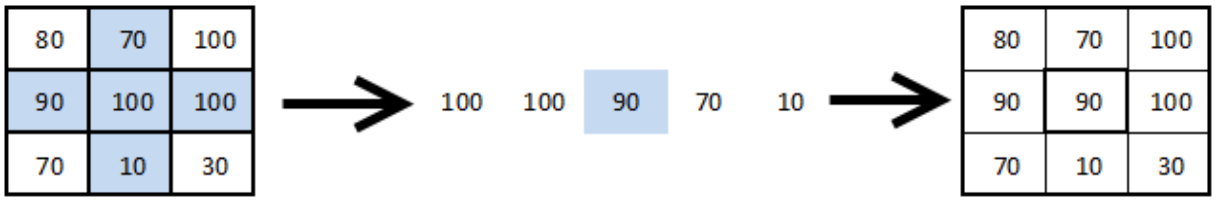
3.1.2 Mediánový filtr

Medián je hodnota, ležící uprostřed nějaké řady, která je seřazená vzestupně nebo sestupně. Pro řady, které mají sudý počet prvků, se medián vypočítá jako aritmetický průměr dvou prostředních prvků. Mediánový filtr určí jas bodu tak, že vypočítá medián jasu bodu v okolí vstupního obrazu. Nejčastější maska tohoto filtru bývá velikosti 3x3, 5x5 a 7x7.

Tento filtr nerozmazává tolik hrany jako Gaussův filtr, ale porušuje tenké čáry. Částečné řešení tohoto problému je nepoužívat standardní masky, které zahrnuje celé okolí, ale jen jeho části [7] [8]. Ukázka standardní a nestandardní masky o velikosti 3x3 je vidět na následujícím obrázku, kde modrou barvou je vyznačena maska.



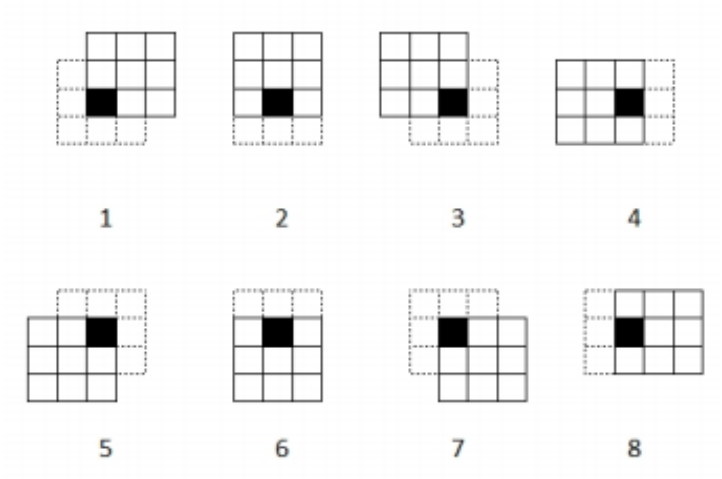
Obrázek 3. Standardní maska 3x3



Obrázek 4. Nestandardní maska 3x3

3.1.3 Filtr rotující maskou

Tento filtr funguje tak, že v okolí bodu 5x5 vyhledá takovou masku 3x3, která nejpravděpodobněji patří do stejného obrazce jako bod. Filtr tedy hledá masku, jejíž jasový průměr je nejblíže k hodnotě jasu hledaného bodu. Filtr tedy hledá masku, jejíž jasový průměr je nejblíže k hodnotě jasu hledaného bodu. Masek, které filtr prozkoumává, je celkem 9. Tato metoda částečně řeší problémy s rozmazáváním a porušováním tenkých čar a ostrých rohů a má mírně ostřící charakter [8].



Obrázek 5. Filtr rotující maskou v okolí 5x5 s maskou 3x3 [8]

3.2 Segmentace obrazu

Segmentace je skupina metod pro úpravu digitálního obrazu, jejichž cílem je rozčlenit obraz do částí a oddělit objekty od pozadí. Výsledkem by tedy měla být skupina vzájemně se nepřekrývajících oblastí. Výsledek je samozřejmě závislý na složitosti scény a použitých metodách zpracování. Výsledkem může být kompletní nebo částečná segmentace.

Při pořízení digitálního obrazu vznikají nežádoucí vlivy, které ztěžují rozpoznávání. Nejznámějším vlivem je šum, který byl již zmíněn výše. Dalšími problémy je nejednoznačnost obrazových dat, složitost scény, nebo se překrývající objekty. Při používání různých metod dostáváme různé výsledky a jedna metoda není bohužel vhodná pro všechny typy obrazových dat a proto je vždy důležité zvolit správnou metodu. Mezi nejznámější segmentační metody patří detekce hran, prahování, znalostní metody, hybridní metody a metody orientované na regiony [3].

3.2.1 Prahování

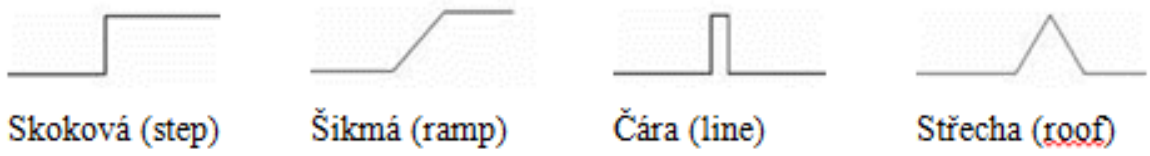
Prahování je nejstarší a nejjednodušší segmentační metoda. Navzdory své omezené použitelnosti jde o široce používanou a oblíbenou metodu. Používá se jak samostatně, tak je součástí jiných sofistikovanějších metod. Popularita prahování spočívá právě v jeho jednoduchosti, ze které vyplývá snadná implementace a velmi malá časová náročnost. Prahování je založeno na myšlence, že objekty a pozadí mají rozdílnou úroveň intenzity. Je

definovaný práh a každý pixel, který má menší hodnotu než tento práh, je určen jako pixel pozadí a všechny ostatní pixely jsou považovány za pixely objektu, či objektů. Výsledek prahování tedy dostáváme po jediném průchodu obrazu.

Způsobů jak využít principu prahování je několik. Nejznámější z nich je globální prahování, kde se určí úroveň šedi jako práh pro celý obraz. Tento způsob však trpí na nerovnoměrnost osvětlení. Tento nedostatek se dá v některých případech odstranit vhodným předzpracováním vstupního obrazu. Další technikou prahování je procentní prahovaní. U tohoto prahování se zadává jako práh procentní zastoupení bodů v obraze a může být vyšší, nižší nebo rovno požadovaného prahu. Procentní prahování je vhodné pro převod skenovaných dokumentů na text [6].

3.2.2 Detekce hran

Detekce hran je jednou z důležitých oblastí zpracování obrazu i přes to, že v reálných scénách je to složitý a dosud nevyřešený problém. Hranami se rozumí body obrazu, u kterých se hodnota jasu prudce mění. Hranu můžeme chápat jako vlastnost obrazového bodu započteného jako funkci obrazu v okolí tohoto bodu. Hrana je reprezentována velikostí a směrem. Změny či přerušení v jasu obrazu jsou jedny z nejzákladnějších charakteristik obrazu, protože naznačují fyzické rozmístění objektů v obraze. Lokální změny jasu obrazu z jedné úrovně na jinou se nazývají jasové hrany a globální změny pak jasové hraniční segmenty. Model ideální hrany může být skoková funkce. V reálných obrazech je změna jasu postupná, nikoli skoková, takže je vhodnější použít šikmou funkci. Pokud se obě definované funkce objeví v obraze vedle sebe, vznikají ještě dva typy hran, a to čára a střecha [3].



Obrázek 6. Typy hran [3]

Velké množství metod pro zvýraznění hran se dá realizovat pomocí konvoluce s vhodným jádrem. Princip konvoluce byl popsán výše. S různými typy matic, lze tedy realizovat různé filtry.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Obrázek 7. Obecný hranový detektor

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

Obrázek 8. Sobelův operátor [6]

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

Obrázek 9. Prewittův operátor [6]

$$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$$

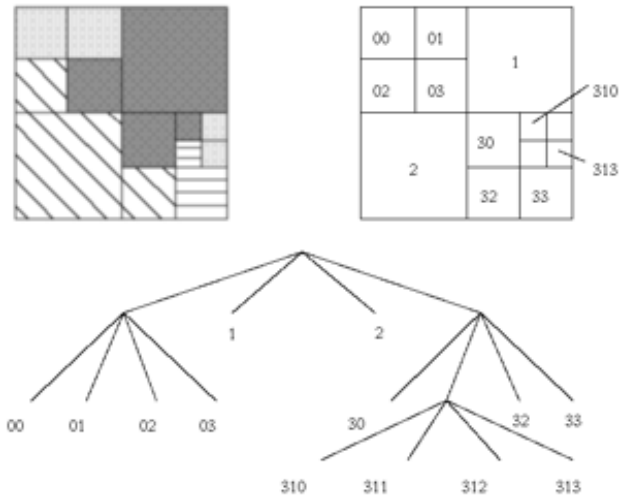
Obrázek 10. Robinsonův operátor [6]

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

Obrázek 11. Kirschův operátor [6]

3.2.3 Hledání oblastí

Hlavní výhodou této metody je mnohem větší odolnost proti šumu. Pro velice zašuměné obrazy vrací lepší výsledky než metody detekce hran. Hlavním segmentačním kritériem pro detekci oblastní v obraze je homogenita oblastí. Kritérium homogenity mohou být: úroveň šedí, barva, textura, tvar, apod [3]. Existují různé metody pro hledání oblastí jako: Region growig, Split and merge, Watershed.



Obrázek 12. Ukázka principu hierarchie segmentace Split and merge

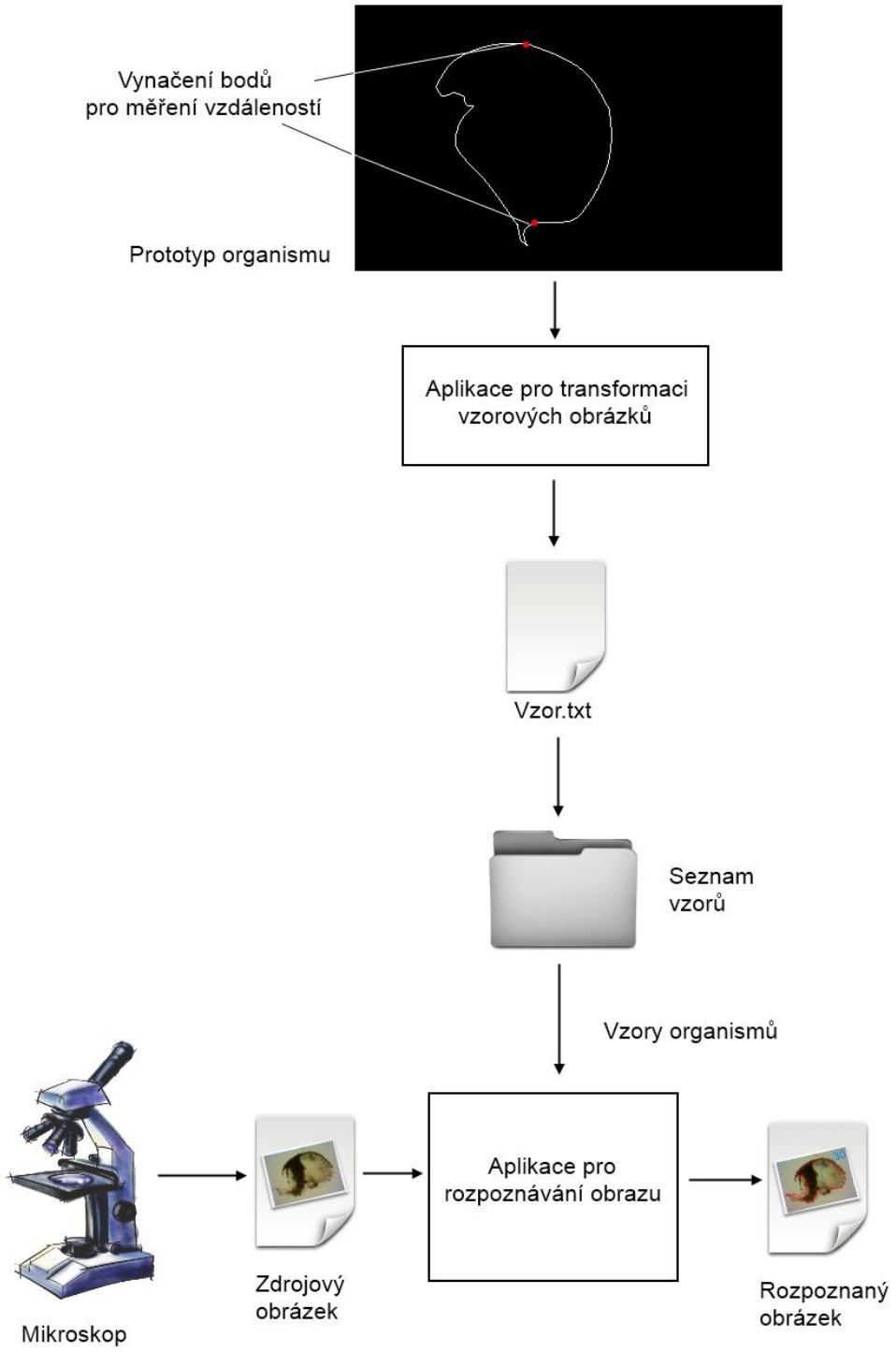
3.3 Rozpoznání obrazu

3.3.1 Houghova transformace

Pro rozpoznání objektů v obrázcích z mikroskopu použiji Houghovo transformaci. Houghova transformace je technika, sloužící k nalezení struktur v obraze. Výhodou je, že velice spolehlivě funguje i s nekvalitními daty, především pokud se jedná o šum. Existuje klasická a obecná Houghovo transformace. Klasická Houghova transformace se využívá pro detekci útvarů, jako jsou přímky, kružnice, elipsy apod., u kterých je parametrický popis dobře známý a snadno formulovatelný. Nás bude zajímat obecná Houghova transformace, protože umožňuje detekovat složitější útvary, u kterých nebudeme schopni nalézt jejich parametrické vyjádření. V takovém to případě se situace zjednodušuje tím, že se používá tabulka, která zachycuje jisté závislosti v hledaném útvaru. Tyto závislosti však nejsou definicí. Robustnost takovýchto charakteristik má zásadní vliv na to, zda detekce bude schopna nalézt požadované tvary. Takováto tabulka s charakteristikami se vytvoří ještě před samotnou detekcí. Za tímto účelem je třeba dodat prototyp tvaru, který chceme hledat. Z tohoto prototypu se vygeneruje tabulka charakteristik, která bude dále v detekci figurovat v podstatě stejně jako definice útvaru [16]. Podrobnější popis Houghovo transformace je v části Návrh řešení.

4 Návrh řešení

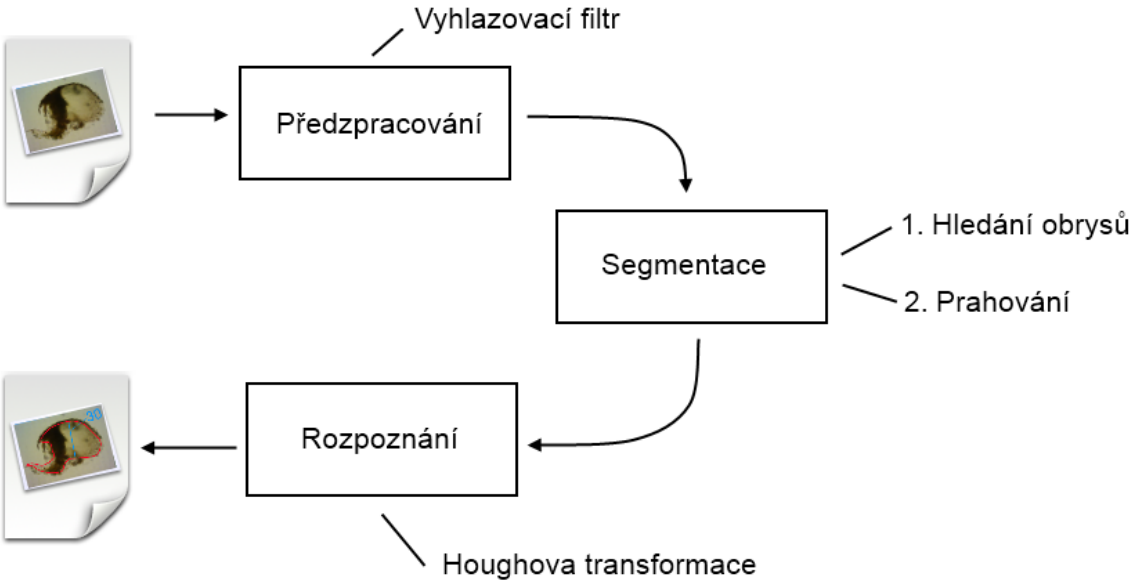
V návrhu řešení jsou popsány postupy a metody, které byly zvoleny pro rozpoznávání objeků na obrázcích a návrh aplikace



Obrázek 13. Funkční logika

4.1 Postup při rozpoznávání obrazu

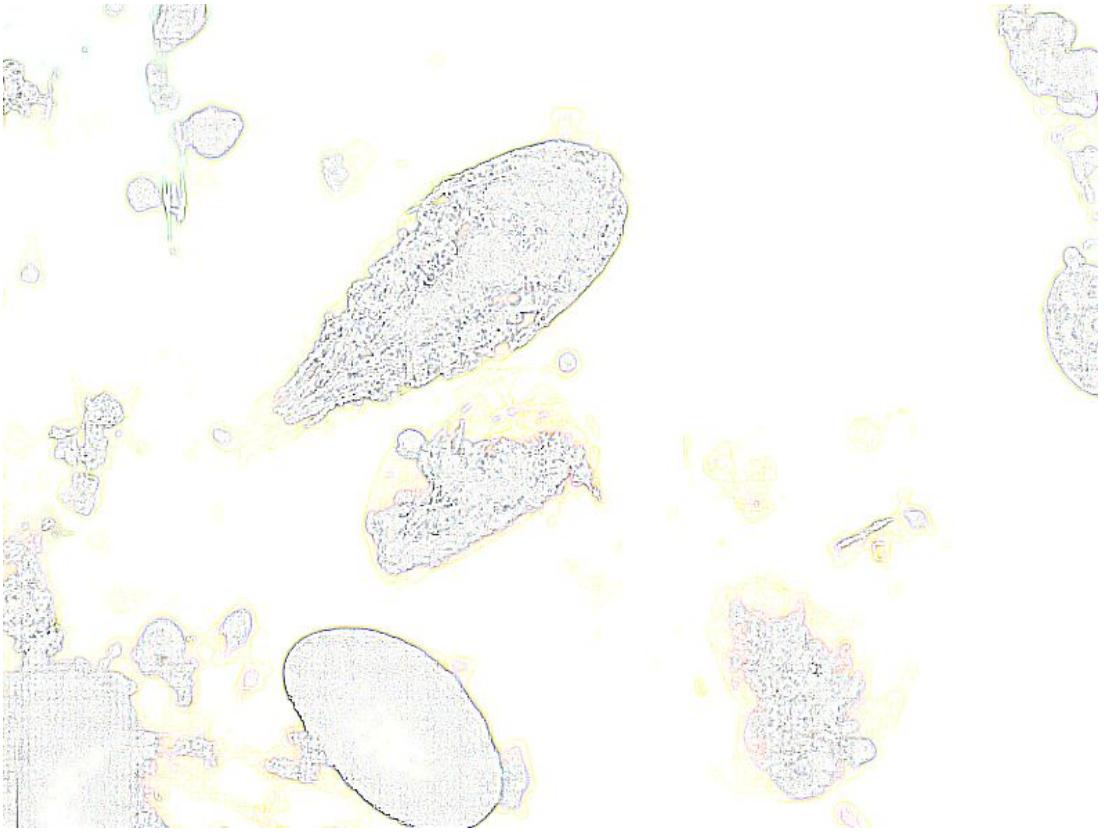
Zpracování obrazu je komplexnější úloha, která se skládá z několika částí. Pro úspěšné zpracování obrazu, kdy výsledkem bude rozpoznaný objekt, musí být implementovány všechny jeho dílčí metody.



Obrázek 14. Postup pro rozpoznávání obrazu

První částí zpracování obrazu je předzpracování. Tato metoda odstraní z obrazu nežádoucí vlivy, jako je například šum. Podle prostudované literatury bylo zjištěno, že nejběžnějším filtrem a v poměru rychlost/kvalita nejlepším filtrem je Gaussův filtr. Proto byl implementován právě tento filtr.

Segmentace se skládá ze dvou částí. První částí je hledání hran. Jak již bylo zmíněno, existuje více metod. Byla zvolena metoda, která se nazývá hranový detektor pro svou jednoduchou možnost implementace a dobrému poměru rychlost/kvalita. Druhým krokem je prahování. Tato metoda převede obrázek na černobílý podle nastavení prahu. Díky předchozí aplikaci hranového detektoru, by měl být dostupný obrázek, kdy bílou barvou budou vyznačeny obrysy objektů. V následujících obrázkách byly barvy invertovány pro lepší viditelnost.



Obrázek 15. Obrázek po aplikaci hranového detektoru

Obrázek 16. Obrázek po aplikaci prahování s prahem 64

Poslední část je rozpoznávání objektů na obraze. Pro nalezení hledaného objektu byla zvolena Houghova transformace. Tato transformace je vhodná i pro využití na měření vzdáleností. Aby ovšem mohla být použita Houghova transformace, je potřeba mít k dispozici ještě takzvaný prototyp hledaného objektu. Postup na získání prototypu je popsán dále. Nejprve je třeba projít celý vzorový obrázek a nalézt všechny bílé body a jejich souřadnice, které se umístí do listu. Dále se vytvoří dvourozměrné pole, jehož rozměry jsou dány velikostí zdrojového obrázku. Poté se zvolí vzorový objekt, jehož střed se umístí do levého horního rohu zdrojového obrázku. Začne se procházet list bílých bodů a ke každému bílému bodu se spočte vzdálenost od středu vzorového objektu podle vzorce (1)

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2} \qquad (1)$$

Proměnné x a y určují souřadnice bílého bodu a x_c, y_c určují souřadnice středu vzorového obrázku. Pokud je tato vzdálenost větší, než největší vzdálenost vzorového objektu, pak je bílý bod prohlášen za příliš vzdálený a nepočítá se s ním. Pokud ovšem je jeho vzdálenost menší nebo rovná, spočte se úhel, který svírá s osou x podle vzorce (2).

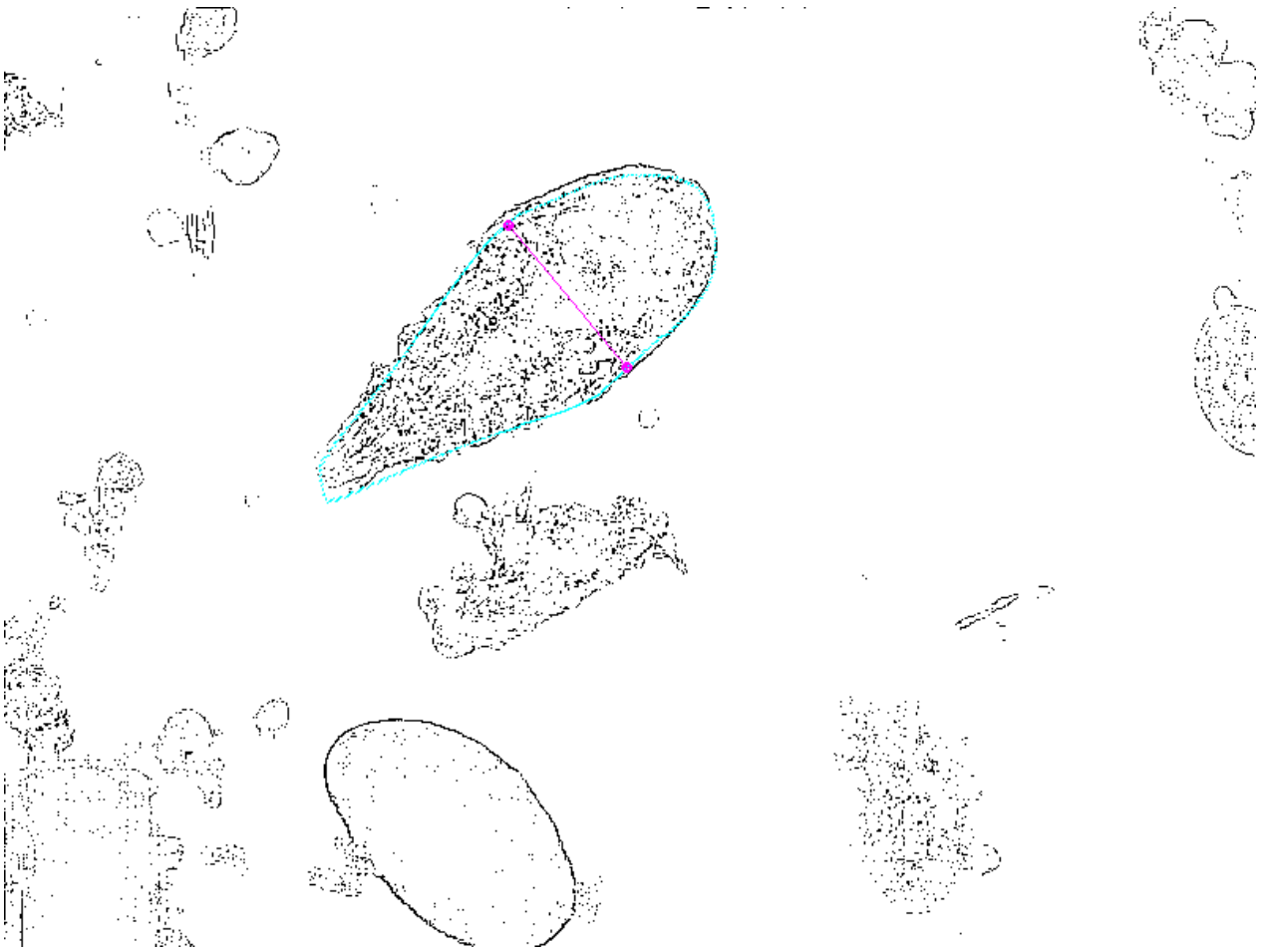
$$\varphi = arctg\frac{y - y_c}{x - x_c} \qquad (2)$$

Tento spočtený úhel je pak hledán ve vzorovém objektu a vzdálenost, která tomuto úhlu odpovídá. Pomocí vzorce (3) se spočítá hodnota, která je od 0 do 1 a určuje přesnost.

$$p = e^{-\frac{(r - r_d)^2}{2\sigma^2}} \qquad (3)$$

Proměnná r je vzdálenost u vzorového objektu, proměnná r_d znázorňuje vzdálenost mezi středem vzorového objektu a bílým bodem na zdrojovém obrázku. Proměnná σ určuje toleranci shody. Tento postup se opakuje pro všechny bílé body na zdrojovém obrázku.

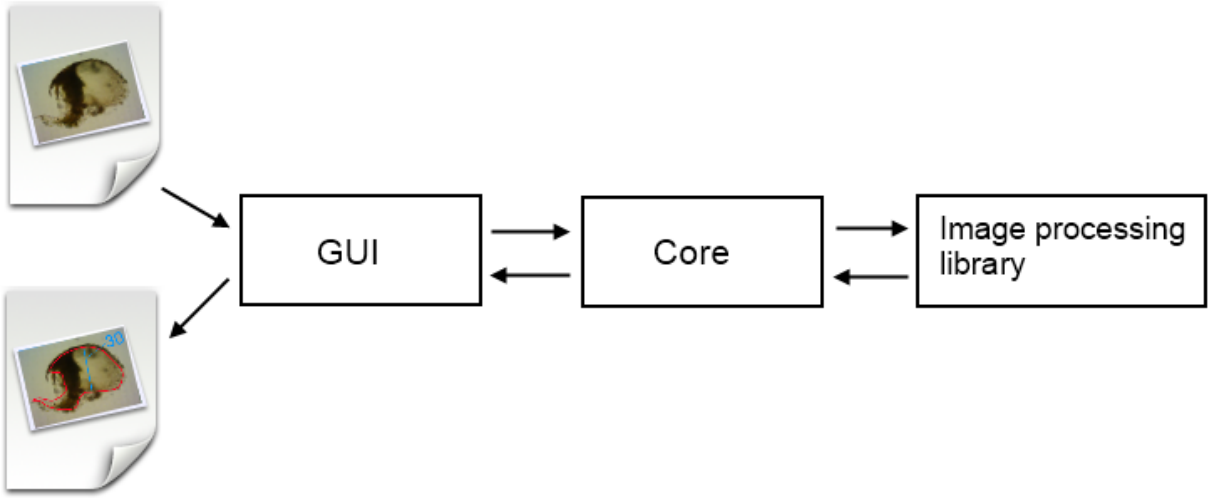
Jednotlivé hodnoty, které jsou vypočteny podle vzorce (3) se sčítají a ukládají do dvourozměrného pole, na místo souřadnice středu vzorového objektu. Poté je střed posunut a celý cyklus se opakuje. Tímto postupem se prochází celý zdrojový obrázek. Výsledkem je dvourozměrné pole naplněné takovými hodnotami, jejichž velikost odpovídá míře shody se vzorovým objektem. Z toho dvourozměrného pole jsou vybrány ty největší hodnoty, které odpovídají největší shodě, jejichž hranice pro vybrání se dá měnit, a jejich souřadnice jsou uloženy do listu. Tyto souřadnice jsou zatím jen podezřelé, že obsahují souřadnice středu nalezeného objektu. Pak dochází k pootočení vzorového objektu a celý proces začíná znovu. Po proběhnutí výpočtů pro všechny úhly se vzorový obrázek o určitou hodnotu zvětší a opět probíhá porovnávání. Protože vzorový objekt obsahuje informace o otočení a vzdálenostech nemusí se hodnoty, o které se má vzorový objekt pootočit nebo zvětšit složitě vypočítávat, ale pouze se přičtou. Protože v této metodě pro hledání se vyskytuje velké množství výpočtů, což vede k velké době hledání, je třeba množství výpočtů co nejvíce snížit. Proto je před začátkem vyhledávání stanovena mřížka, po které se střed vzorového objektů posunuje. Posun tedy není 1 pixel, ale podle toho, jak si ho stanoví uživatel pomocí GUI. Aplikace dovoluje nastavit velikost mřížky na 2,4,8,16 a 32. Dále není třeba porovnávat každý jednotlivý úhel, ale pouze některé úhly. Tyto úhly se mění po 30°. Velikost objektu je měněná v rozmezí 0.8 až 1.4 po hodnotách 0.2. Těmito změnami dochází k velkému zrychlení. Po skončení prohledávání pro jeden objekt dochází k tomu, že v listu, ve kterém jsou nashromážděny středy, se seřadí podle velikosti shody. Z toho listu se vyberou pouze ty hodnoty, které jsou největší pro dané okolí, ostatní se zahodí. Pro několik takto získaných středů se provede zpřesnění velikosti a úhlu natočení. Protože je těchto středů jen pár, nepředstavuje to takovou výpočetní zátěž. Zpřesnění v úhlu se provádí po 5° a velikost se zpřesňuje po velikost 0.05. Takto upravené středy se uloží do nového listu, který obsahuje všechny středy. Tento celý proces se provádí stejným způsobem pro další vzorový objekt.



Obrázek 17. Rozpoznaný objekt po Houghovo transformaci

4.2 Návrh aplikace pro rozpoznávání obrazu

Tato aplikace má za úkol hledat objekty pomocí Houghovo transformace na obrázcích. Aplikace se skládá ze 3 částí – GUI, Core a Image processing library.

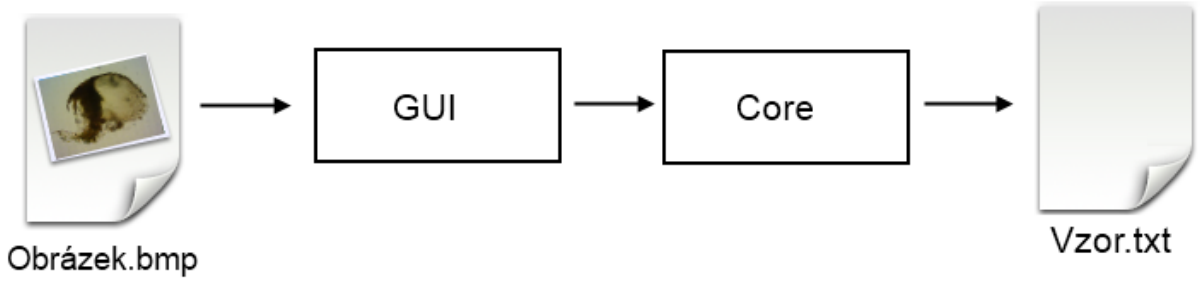


Obrázek 18. Návrh aplikace pro zpracování obrazu

GUI je grafické uživatelské rozhraní, které umožňuje jednoduchou práci s aplikací. Obsahuje různé nastavovací prvky pro metody zpracovávající obraz. Přes toto rozhraní si může uživatel navolit obrázky, které se mají zpracovávat. Tyto zvolené obrázky se zobrazují v jednoduchém seznamu. GUI dále obsahuje seznam jednotlivých metod pro zpracován obrazu. Tyto metody lze deaktivovat, což umožnuje, aby se dali jednotlivé metody pohodlně nastavovat a bylo také možno shlédnout výsledek tohoto nastavení. Core obsahuje seznamy obrázků a filtrů, a stará se hlavně o to, aby se metody pro zpracování obrazu vykonávaly. Image processing library je knihovna, která obsahuje zpracované metody pro zpracování obrazu. Skládá se z vyhlazovacího filtru, hranového detektoru, prahování a Houghovo transformace.

4.3 Návrh aplikace pro transformaci vzorových obrázků

Tato aplikace převádí černobílý obrázek na tabulku, která obsahuje informace o úhlech a vzdálenostech. Tyto informace jsou uloženy do textového souboru. Tyto soubory využívá Houghova transformace jako zdroj vzorových dat, které na obrázcích vyhledává.



Obrázek 19. Návrh aplikace pro transformaci vzorových obrázků

Jak již bylo řečeno, je potřeba prototyp obrázku. Tento prototyp je potřeba převést na černobílý, kde se vyznačí obrysy objektu bílou barvou. Tento obrázek musí být ve formátu bmp kvůli zachování kvality obrázku. Hledaná vzdálenost se vyznačí pomocí jakékoliv dvojice barev. Je důležité dodržet pravidlo, kdy vyznačené body pro měření vzdálenosti musí mít stejnou barvu. Pro měření další vzdálenosti se musí zvolit další jakákoliv nepoužitá dvojice barev. Tyto barvy nesmějí být samozřejmě černé ani bílé, protože těmito barvami je vyznačeno pozadí a obrys objektu. Takto upravený obrázek se použije na vstup aplikace. Z tohoto obrázku se dopočítá těžiště objektu zprůměrováním souřadnic bílých bodů. Z tohoto těžiště se spočítá vzdálenost ke každému bílému bodu a úhlu, který svírá s osou x. Vzdálenost na ose x se spočítá odečtením souřadnice těžiště x od souřadnice x bílého bodu. Tento postup opakuji pro souřadnice y. Z vypočtených vzdáleností na ose x a y se spočte pomocí Pythagorovy věty přepona, což je hledaná vzdálenost. Následně se vypočte úhel pomocí goniometrické funkce cotg. Vznikne tety tabulka obsahující informace o úhlu a vzdálenosti k bílým bodům. Pro zrychlení hledání úhlů, jsou úhly zaokrouhleny a vzniká pole o 360 prvcích, kdy index pole odpovídá úhlu a do pole jsou zaneseny pouze vzdálenosti k bílým bodům. Díky zaokrouhlování může nastat případ, kdy pod úhlem není žádná vzdálenost, nebo naopak je k úhlu přiřazeno více vzdáleností. Z těchto informací lze snadno dopočítat opět souřadnice každého bílého bodu. Díky znalosti pouze úhlu a vzdálenosti se dá snadno manipulovat s velikostí a pootočením při rekonstrukci vzorového obrázku. Obdobným způsobem se zpracovávají body, pomocí kterých se měří vzdálenost. Tato aplikace urychlí zpracování obrazu, protože Houghova transformace použije již vypočtená data.

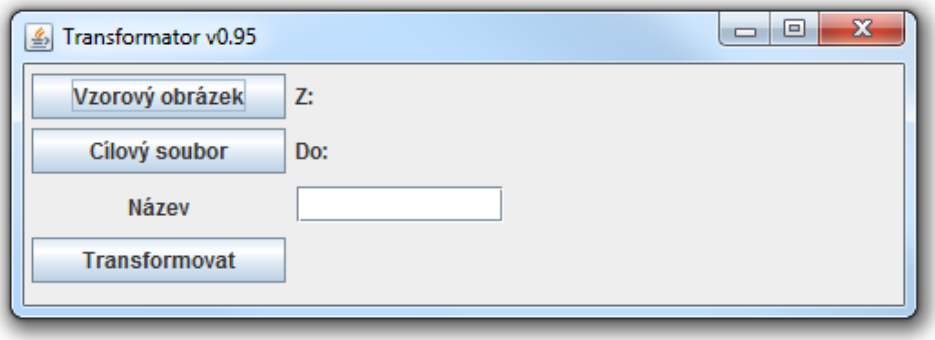
5 Implementace

V této části jsou popsány implementace metod aplikace pro transformaci a aplikace pro rozpoznávání obrázků.

5.1 Implementace aplikace pro transformaci vzorových obrázků

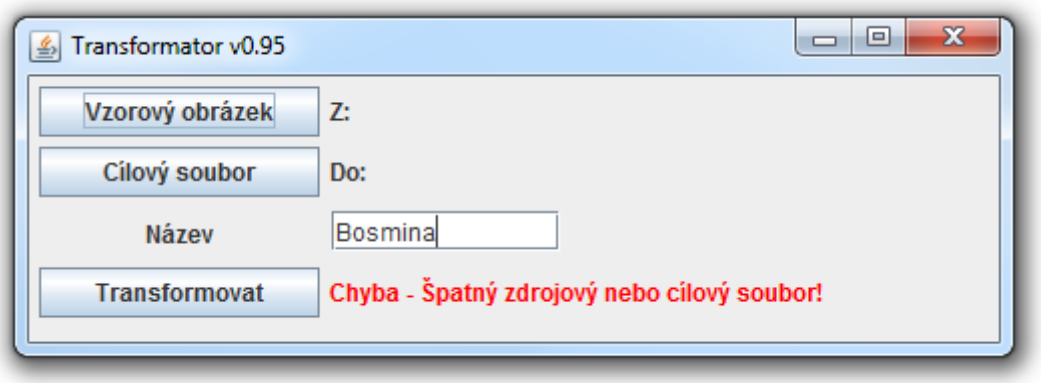
Pro převod černobílých obrázků na tabulku o vzdálenostech a úhlech slouží třída Core. Druhou třídou je třída TransformtorWindow, která implementuje jednoduché grafické rozhraní.

Třída TransformatorWindow má konstruktor bez parametrů. Po vytvoření této třídy se zobrazí jednoduché grafické prostředí.

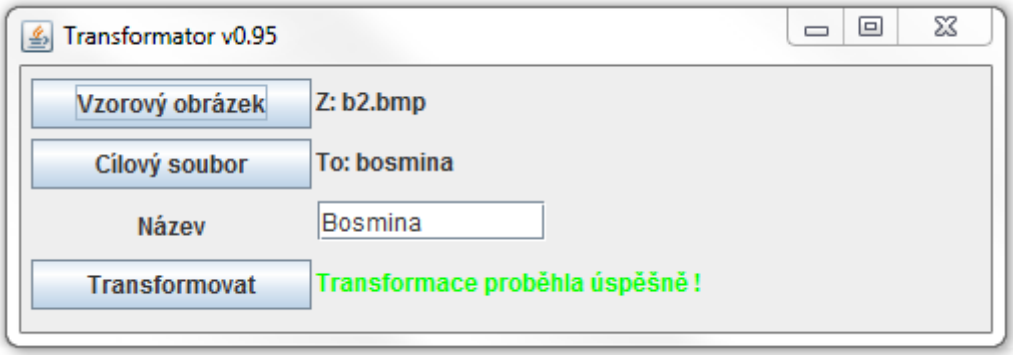


Obrázek 20. Grafické rozhraní aplikace pro transformaci

Toto prostředí obsahuje dvě tlačítka. Prvním tlačítkem „Vzorový obrázek“ se volí černobílý obrázek vzorového objektu. Druhým tlačítkem „Cílový soubor“ se volí výsledné místo, do kterého se uloží textový soubor. Možnost výběru souborů u tlačítek je realizována pomocí objektu JFileChooser, kterým lze díky přednastavenému filtru vybrat pouze správné formáty souborů. Do textového pole „Název“ se vyplňuje název převáděného obrázku. Tlačítkem „Transformovat“ se vytvoří třída Core, která se stará o převod. V případě nevybraných souborů, nebo nevyplněným názvem, skončí výsledek transformace neúspěšně. O neúspěšném převodu informuje chybová hláška. V opačném případě informuje aplikace o úspěšném převodu.



Obrázek 21. Neúspěšný převod



Obrázek 22. Úspěšný převod

Konstruktor třídy Core má tři parametry. První parametr je typu File. Tento parametr udává vstupní soubor. Tento soubor by měl být typu bmp. Druhý parametr je také typu File, který udává výstupní soubor. Typ souboru by měl být txt. Poslední parametr je typu String. Tento parametr nese název objektu na obrázku, který se používá jako vstupní soubor. Dále konstruktor volá pětici privátních metod – calculationCenter(), calculationFiR(), sort(), rounding(), savingToFile(). Tyto metody musí být volány přesně v uvedeném pořadí.

Metoda calculationCenter slouží ke spočtení těžiště, nalezení bílých bodů a bodů mezi kterými se měří vzdálenost. Metoda prochází obraz a hledá barvu bodů, které odpovídají 0xFFFFFFFF, což značí bílou barvu. Po nalezení se do pole o dvou prvcích zapíší souřadnice x a y bílého bodu. Toto pole se následně uloží do listu „listxy“. V případě nalezení jiné barvy, která ovšem není černá, se souřadnice a barva uloží do jiného listu. Po nalezení všech bílých bodů se spočítá na základě souřadnic bílých bodů těžiště. Tato metoda dále volá privátní metodu sortMeasurementPoint s jedním parametrem. Tento parametr je typu list. List by měl obsahovat informace o barevných bodech. Tato metoda seřadí barevné body podle barvy v

listu. Tento list je následně seřazen do takové podoby, že osahuje pole o 4 prvcích. Každé pole obsahuje souřadnice dvou bodů mezi kterými se bude měřit vzdálenost. Takto setříděný list se uloží do listu „cp“ (color points).

Metoda calculationFiR slouží k výpočtu úhlů a vzdáleností. Prochází list „listxy“, který nese souřadnice. Odečtením souřadnice těžiště od souřadnic bílých bodů se získají vzdálenosti po ose x a y. Z těchto vzdáleností se spočítá přes Pythagorovu větu vzdálenost od těžiště k bílému bodu. Pro výpočet úhlu je použita z třídy Math metoda atan2. Výsledný úhel a vzdálenost se uloží do pole o dvou prvcích, kdy první prvek je úhel a druhý vzdálenost. Toto pole se přidá do listu „listfir“. Postup se opakuje, dokud se neprojde celý list „listxy“. Dále se provádí obdobný proces s barevnými body.

Další volanou metodou je metoda sort. Tato metoda setřídí list „listfir“ podle úhlu.

Po setřídění se volá metoda rounding. Metoda se stará o převod z radiánů na úhly. Použití metody atan2 v metodě calculationFiR způsobilo to, že úhly jsou v radiánech. Dále metoda atan2 vrací úhly od - π do π. Proto se tato metoda stará o to, že převádí radiány na úhly. Tyto úhly jsou následně zaokrouhlovány. Pro odstranění rozsahu -180° až 180° je přičteno k úhlům menším než 0° hodnota 360°. Metoda dále sloučí duplicitní úhly pod jeden. Protože nyní odpovídá jeden index jednomu úhlu, vznikne pole o 360 prvcích. Každému indexu jsou přiřazeny pouze vzdálenosti. Při zaokrouhlování úhlů mohlo dojít k tomu, že k úhlu není přiřazena žádná vzdálenost. V tomto případě obsahuje daný index vzdálenost -1.

index	úhel	vzdálenost
0	0	17.2
1	0	17.3
2	0	17
3	1	17.5
4	1	17.6
5	2	17.9
6	2	18
7	2	18.1
8	2	18
9	4	18.3
10	4	18.4

index	vzdálenost
0	17, 17.2, 17.3
1	17.5, 17,6
2	17,9, 18 ,18, 18.1
3	-1
4	18.3, 18.4

Obrázek 23. Příklad transformace úhlů a vzdáleností

Poslední metodou je savingToFile. Tato metoda, jak název napovídá, ukládá všechny získané informace do souboru. Pro usnadnění práce se soubory jsou k dispozici další dvě třídy Length a Data. Třída Length je třída, do které se ukládají informace o bodech, mezi kterými se měří vzdálenost. Pro každou vzdálenost se vytvoří nový objekt ze třídy Length. Konstruktor třídy Length má pětici parametrů. První parametr je typu String a nese pojmenování měřené vzdálenosti. Druhý parametr je typu int. Tento parametr udává uhel k prvnímu bodu. Třetí parametr je typu double a udává vzdálenost k prvnímu bodu. Čtvrtý parametr je typu int a nese úhel druhého bodu. Poslední parametr je typu double a nese vzdálenost k druhému bodu. Všechny objekty typu Length se ukládají do listu. Do třídy Data se vkládají všechny další vypočítané informace. Konstruktor třídy Data má 6 parametrů. Prvním parametrem je dvourozměrné pole typu double. První rozměr je 360. Druhý rozměr je volitelný, protože není dopředu známo počet vzdáleností, které připadají k indexům. Druhý parametr je typu int a nese počet bílých bodů vzorového obrázku. Třetí parametr je typu List. Tento List je typu Length a obsahuje všechny objekty, které představují vzdálenosti mezi body. Čtvrtý a pátý parametr je typu int. Tyto parametry nesou nejkratší a nejdelší vzdálenost k bílému bodu. Poslední parametr je typu String a udává název hledaného objektu. Objekt typu Data je pak uložen do souboru. Aby bylo možno uložit objekt Data do souboru, musí implementovat rozhraní Serializable. Třída Core obsahuje několik veřejných metod. Tyto metody nejsou k výpočtům potřebné, ale umožňují kontrolovat správnost výpočtů. První je veřejná metoda je „vykresleniObrazku“. Tato metoda vrací vzorový obrázek. Druhou metodou je „VykresleníBodu“. Tato metoda vykreslí do objektu typu BufferedImage obrázek. Tento obrázek je zakreslen na základě spočtených vzdáleností a úhlů. Po zakreslení, vrací metoda tento obrázek.

5.2 Implementace aplikace pro rozpoznávání obrazu

Jak již bylo řečeno, tato aplikace se skládá ze 3 samostatných projektů. První projekt je GUI, které umožňuje uživateli jednoduché ovládání aplikace. Druhý projekt je Core, které se stará o správný chod aplikace a posledním projektem je IPLibrary (Image Processing Library), který obsahuje implementované metody pro zpracování obrazu.

Projekt IPLibrary obsahuje následující třídy SmoothingFilter, EdgeDetection, Thresholding, BlackWhiteFilter, HoughTransform a pomocnou třídu pro Houghovo transformaci HTCenter.

Třída SmoothingFilter reprezentuje vyhlazovací filtr, jehož konstruktor je bez parametrů. Obsahuje metodu setImage s parametrem typu BufferedImge. Pomocí této metody se nastavuje zdrojový obrázek, který se má upravit. Dále obsahuje metodu edtiImage. Zavoláním této metody se spustí úprava obrázku. Poslední metodou je metoda getImage, která vrací upravený obrázek.

Další třída je edgeDetection, která obsahuje stejné metody se stejnými funkcemi a proto ji nebudu dále rozepisovat.

Třída Thresholding je třída, která provádí prahování. Po jejím vytvoření konstruktor inicializuje horní a dolní prah. Obsahuje dvojici setBottomThresh a setUpperThresh. Obě metody mají jeden parametr typu int, kterým se nastavuje úroveň prahů. Dále obsahuje druhou dvojici metod getBottomThresh a getUpperThresh, které vrací úroveň prahů. Další metoda je setImage s jedním parametrem typu BufferedImage. Tato metoda nastaví zdrojový obrázek a metoda getImage vrátí upravený obrázek. Poslední metoda je editImage, která oprahuje zvolený obrázek a změní ho na černobílý.

Další třídou je třída BlackWhiteFilter. Tato třída je spíše doplňková a jejím účelem je procházení černobílého obrázku a hledání osamocených bílých bodů, které se přebarví na černou barvu. Použití tohoto filtru je vhodné po použití prahovaní. Vytvořením této třídy se inicializují proměnné, které udávají velikost hledaného místa počet bílých bodů, které obsahuje. Obsahuje několik metod. Metoda setImage má jeden parametr typu BufferedImage pro nastavení zdrojového obrázku a metodu getImage pro vrácení upraveného obrázku. Metody setLength_x a setLenth_y mají parametr typu int, kterým se udává velikost procházené oblasti. GetLength_x a getLenth_y vrací velikost procházené oblasti. Metoda setLimit s parametrem typu int, udává maximální počet bílých bodů které se v hledané oblasti

může nacházet. Pokud je bílých bodů nalezeno méně než hranice, tak se obarví na černou barvu. Metoda getLimit vrací maximální počet bílých bodů. Poslední metodou je metoda editImage, jejíž zavoláním se spustí upravování.

Třída HTCenter je pomocná třída pro Houghovo transformaci, která nese informace o předpokládaném středu nalezeného objektu, typu nalezeného objektu, množství bílých bodů z kterých se vzorový objekt skládá, jeho velikosti a natočení oproti vzorovému objektu. Pro každý nalezený útvar se vytvoří nový objekt. Konstruktor této třídy má 6 parametrů. První 3 parametry jsou typu int. První udává množství bílých bodů, druhý souřadnice středu na ose x a třetí souřadnice středu na ose y. Čtvrtý parametr je typu double a udává poměř zvětšení nebo zmenšení oproti vzorového objektu. Pátý parametr je typu int a udává pootočení oproti vzorového objetu. Poslední parametr je typu Data, to je objet, který je popsán výše, a udává typ nalezeného objektu. Třída dále obsahuje pětici metod. GetCenter_x vrací souřadnice středu na ose x, getCenter_y vrací souřadnice středu na ose y, getAngle vrací pootočení oproti vzorového obrázku, getAmount vrací množství bílých bodů, které obsahuje vzorový obraz, a getAnimal vrací objekt Data.

Nejdůležitější třídou je HoughtTransform, jejíž cílem je rozpoznávat objekty. Konstruktor této třídy nemá žádné parametry, inicializuje pouze řadu proměnných. Třída obsahuje několik veřejných metod. První metoda setImage slouží k vybrání obrázku, na kterém se mají vyhledávat objekty. Metoda má jeden parametr typu BufferedImage. Další metodou je addPattern, pomocí které se vybírá složka, která obsahuje zpracované vzorové obrázky. Má jeden parametr typu String, který nese cestu ke složce. Další metodou je clearAll, která slouží pro mazaní proměnných. Tuto metodu je vhodné volat před každým začátkem vyhledávání. Další veřejnou metodou je metoda searchingPattern, která prohledává obrázek a hledá shodu se vzorovými obrázky. Dále obsahuje metodu setMoveOnCenter. Tato metoda má jeden parametr typu int a udává velikost kroků, po kterých se obrázek vyhledává. Metoda getMoveOnCenter vrací velikost těchto kroků. Další metodou je metoda setTheta s jedním parametrem typu double. Tato metoda nastavuje toleranci vzhledem k rozdílu vzdálenosti bílému bodu na vzorovém obrázku a obrázku, na kterém se vyhledávají objekty. Metoda getTheta vrací hodnotu, na kterou je nastavena tolerance. Dále obsahuje metodu setAnimalPointsPercest s jedním parametrem typu int. Tato metoda nastavuje hodnotu v procentech., která určuje míru shody pro uznání nalezeného obrázku. Metoda getAnimalPointsPercent vrací tuto hodnotu. Poslední metodou je metoda getTrueCenter, která vrací List typu HTCenter, což představuje seznam nalezených objektů. Třída dále obsahuje

čtveřici privátních metod, které jsou volány z metody searchingPattern. Metoda whitePointImage uloži do listu všechny bílé body, které se vyskytují v obraze. Metoda createMatrix vytvoří matici o velikosti obrázku, na kterém se vyhledávají tvary. Do této matice se poté zaznamenávají vypočtené hodnoty pro každé těžiště, které se vypočítávají v metodě searchingPattern. Další metodou je metoda setIntenzity. Tato metoda prohledává matici, a pokud obsahuje na nějakém místě hodnotu větší než je zvolená hranice, tak se do listu vloží nový objekt HTCenter, který je popsán výše, s příslušnými parametry. Hranice je volitelná, ale její největší velikost odpovídá počtům bílých bodů u vzorového obrázku. Metoda findTrueCenter vyfiltruje falešné nalezené objekty. Takovéto falešné nalezené objekty jsou zpravidla vedle sebe, a proto se ze shluku těchto objektů vybírá pouze ten, který měl v matici největší hodnotu. Po vyfiltrování objektů je volána metoda focus. Tato metoda se snaží vypočítat přesnou velikost a úhel natočení nalezeného objektu.

Druhý projekt Core obsahuje čtveřici tříd: Core, Image, Marking a Save.

Třída Save, jak už název napovídá, slouží ukládání. Tato třída implementuje rozhraní Serializable, aby mohl být následně celý objekt uložen do souboru. Tato třída má konstruktor bez parametrů a vytvořením objektu se jen inicializuje trojice listů. Třída dále obsahuje řadu metod. Metoda setName má jeden parametr typu String a louží k vložení jména obrázku do listu, kde parametr nese název. Třída getName má také jeden parametr typu int, který svým číslem vybírá podle indexu název z listu. Další metodou je setPath, která má parametr typu File a ukládá se opět do listu. Metoda getPath má jeden parametr typu int, pomocí kterého se opět vybírá prvek z listu. Další metodou je metoda setHTCenter s jedním parametrem typu list, který je typu HTCenter. Třída HTCenter je popsána výše. Tento list typu HTCenter se vkládá do listu. Výsledkem je tedy pro upřesnění list listů typu HTCenter. Metoda getHTCenter má jeden parametr typu int, pomocí kterého se vybírá list z listu. Poslední metodou je metoda size, která vrací počet prvků v listu. Výsledkem jsou 3 listy které mají stejné množství prvků.

Další třídou je třída Marking. Každý objekt z této třídy označuje jeden filtr. Třída má tři parametry. První je typu String a označuje jméno filtru. Druhý je typu int a pomocí něj se identifikuje filtr. Identifikační čísla všech filtrů jsou přiřazena ve třídě Core. Poslední parametr je typu boolean a označuje hodnotou true a false, zda-li je filtr aktivní či nikoli. Skládá se z řady metod. Metoda getName vrací název filtru. Metoda getIdFilter vrací id filtru. Metoda getExecute vrací true nebo false. Podle toho, zda se má filtr vykonávat nebo ne.

Poslední metodou je metoda setExecute, která má jeden parametr typu boolean. Nastavuje se tedy pravdivostní hodnota true nebo false.

Třída Image obsahuje informace o obrázku, na kterém se vyhledávají objekty. Uchovává informace o vzhledu obrázku před a po aplikování filtrů a počet nalezených objektů přes třídu HTCenter, která je popsána výše. Konstruktor třídy má jeden parametr typu File, který nese informaci o zdrojovém obrázku. Třída obsahuje metody refresModifiImage, pomocí které se smaže upravený obrázek a načte se znovu čistý, bez aplikovaných filtrů. Metodu setTrueCenter má jeden parametr typu List, který je typu HTCenter. Tato metoda ukládá nalezené objekty. Metoda getTrueCenter vrací list všech nalezených objektů. Metody setImage a setModifiImage mají jeden parametr typu BufferedImage, který nese informaci o zdrojovém obrázku. A metody getImage a getModifiImage vrací zdrojový obrázek buď v neupraveném nebo upraveném stavu. Metoda getPath vrací cestu k obrázku a metoda getName vrací název obrázku.

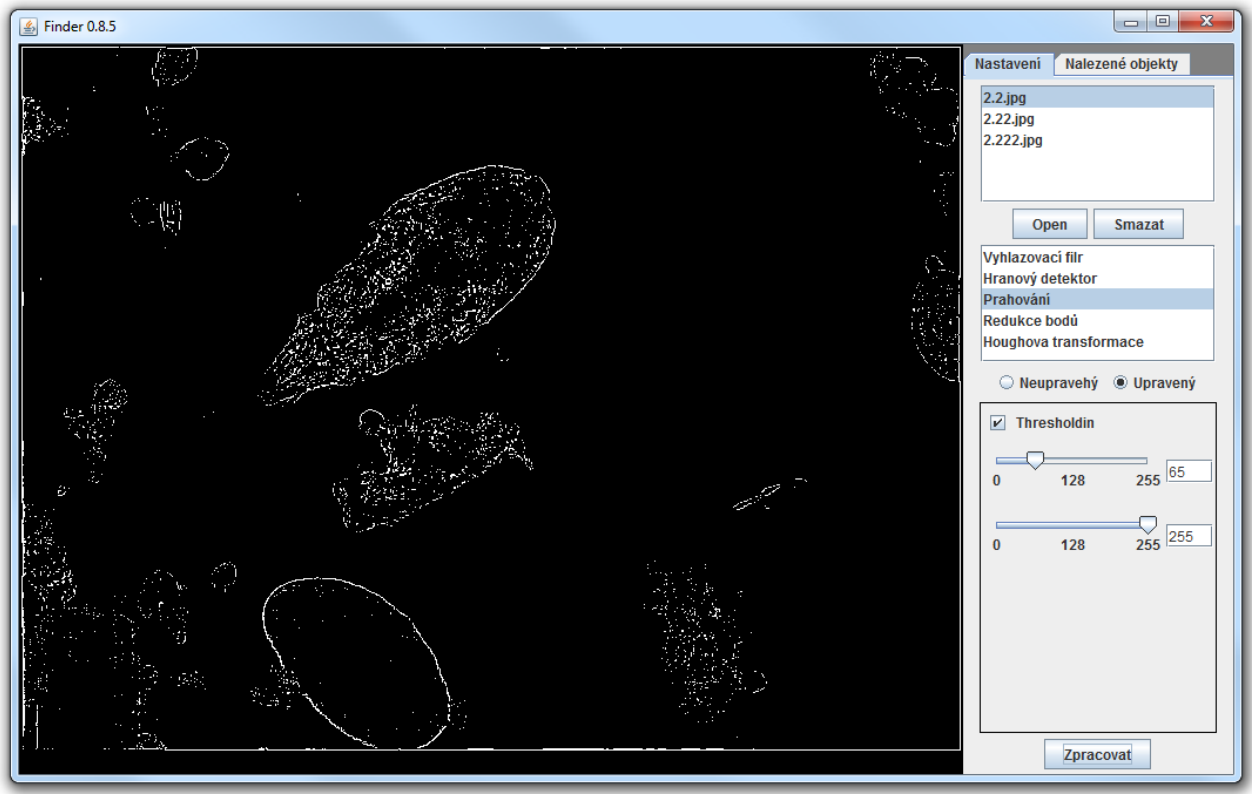
Vytvořením třídy Core se zavolá konstruktor, který inicializuje proměnné, vytvoří seznam filtrů, zkontroluje zda-li existují potřebné adresáře a soubory. Pokud neexistují, tak je vytvoří, pokud existují, tak zajistí jejich načtení a získání posledního pracovního nastavení. Skládá se z metod addPicture, která má jeden parametr typu File[]. Tento parametr nese pole obrázků, které se přidají do seznamu a jsou určeny pro zpracování. Další metoda getListofImages vrací list objektů typu Image. Metoda getImage vrací konkrétní objekt typu Image a má jeden parametr typu int, pomocí kterého se nastavuje index hledaného objektu v listu. GetListMarking vrací list typu Makring. Třída dále obsahuje dvojici metod setLibPath a getLibPath. SetLibPath má jeden parametr typu String a nastavuje cestu ke vzorovým obrázkům pro Houghovo transformaci. Metoda getLibPath vrací cestu k vzorovým obrázkům. Dálé třída obsahuje metody getThresholding, getHoughTransform, getEdgeDetection, getSmoothingFilter a getBlackWhiteFilter. Tyto metody vrací jednotlivé objekty. Pro ukládání a načítání slouží medoty saveImage a loadImage. Poslední metodou je metoda starProcess, po jejímž volání se na obrázek aplikují jednotlivé filtry a začne vyhledávání obrázku. Tato metoda má jeden parametr typu int, který udává index obrázku v listu.

Posledním projektem je Projekt s názvem GUI, který vytváří grafické rozhraní pro uživatele. Skládá se z tříd Window, PanelAnimal a hlavní třídy Main.

Třída PanelAnimal slouží ke grafickému znázornění nalezených objektů. Každý objekt z této třídy se zakreslí do předem připraveného seznamu. Konstruktor má dva parametry.

První je typu Window, který nese referenci na hlavní grafické rozhraní. Druhý je typu String, který obsahuje jméno nalezeného objektu. Třída má dvě metody. První je metoda getChekAnimal, která vrací true nebo false, podle toho, zda-li je checkBox zaškrtnutý nebo ne. Druhou metodou je addLenght, která vypisuje vzdálenost. Má pět parametrů. První čtyři jsou typu int a určují souřadnice bodů mezi kterými se měří vzdálenost. Pátý parametr je typu String a nese pojmenování vzdálenosti pro přehled.

Druhu třídou je třída Window. Tato třída se stará o vytvoření grafického rozhraní, které se inicializuje pomocí konstruktoru. Obsahuje metodu paint, pomocí které se vykreslují objekty a volá privátní metodu draw, která se stará o vykreslování objektů. Další metoda checkResolution kontroluje velikost obrázku. Pokud je zdrojový obrázek větší než 800x600, tak se obrázek vizuálně zmenší tuto velikost. Dále obsahuje metody setLayoutThesholding, setLayoutHough a setLayoutBlackWhite. Tyto metody nastavují vykreslení jednotlivých nastvení fitrů.



Obrázek 24. Grafické rozhraní

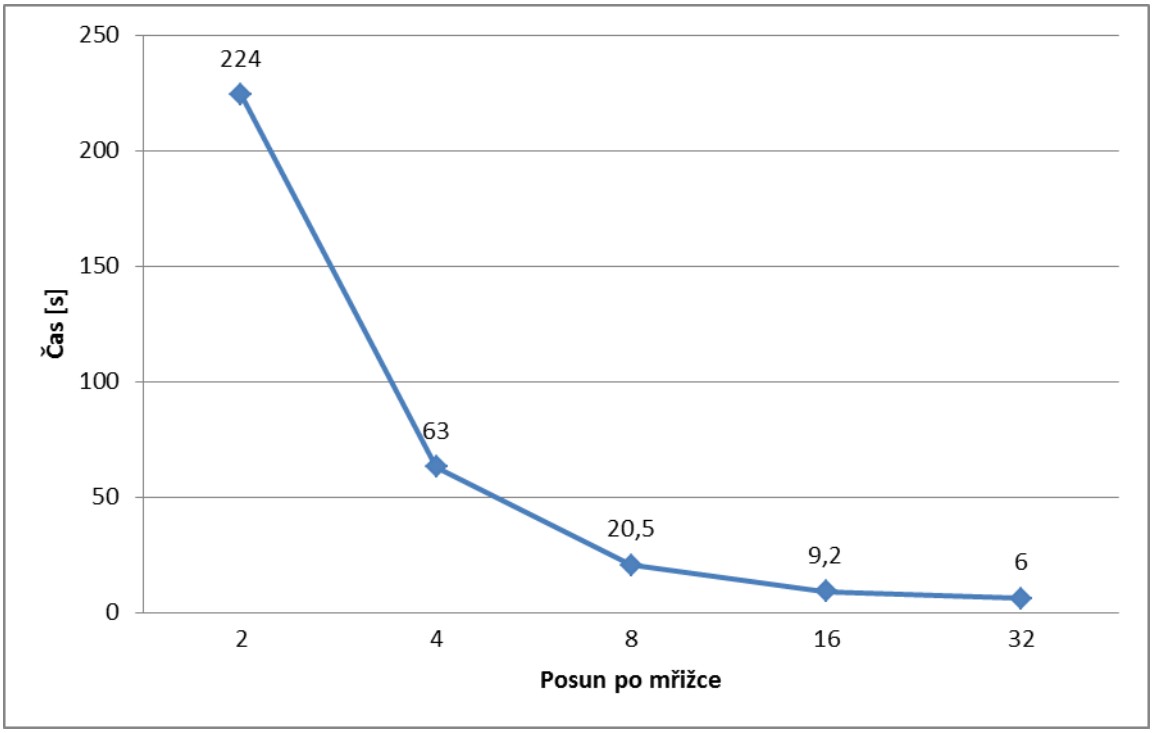
Poslední třídou je třída Main, která pouze vytváří objekt ze třídy Window.

6 Testování

Testování se bude skládat ze dvou částí. V první části se budou rozeznávat předpřipravené černobílé obrázky s jasnými obrysy. Ve druhé části se bude testovat rozpoznávání na pravých snímcích z mikroskopu.

6.1 Testování na připravených obrázcích

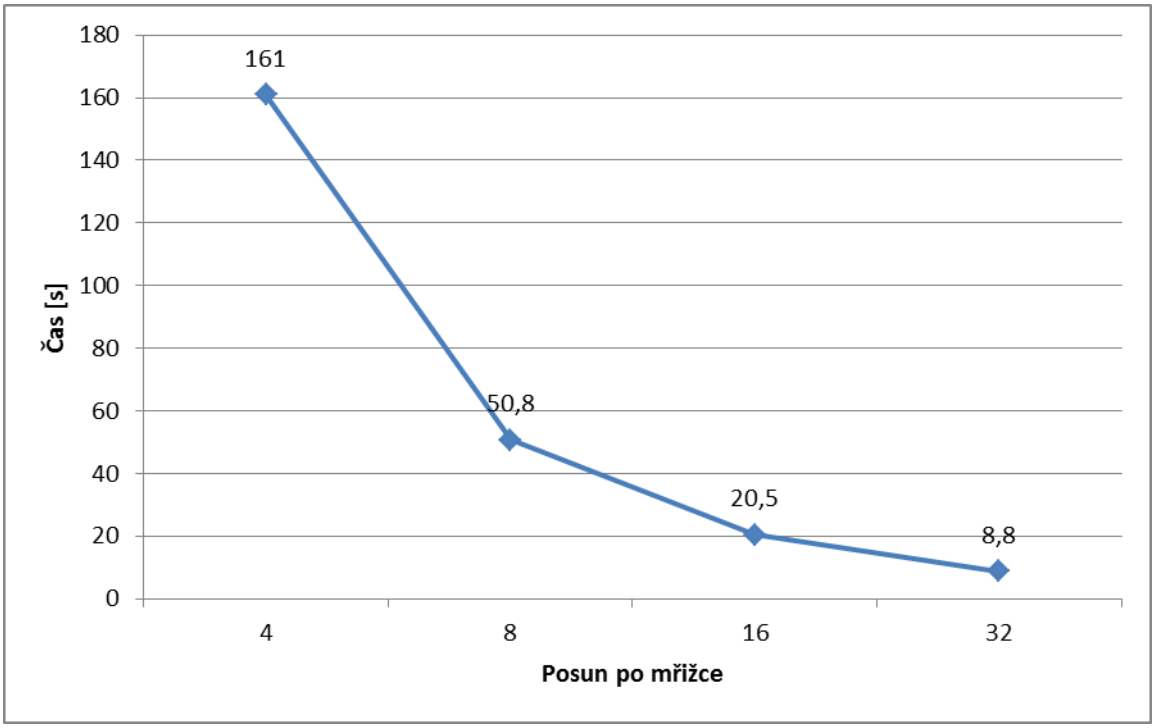
Pro první test byl zvolen obrázek o velikosti 500x500 pixelů. Objekt byl náhodně pootočen a byla mu náhodně změněna velikost. Počet bílých bodů na obrázku byl 1962. Knihovna obsahovala jeden vzor hledaného objektu.



Obrázek 25. Graf znázorňující rychlost hledání v závislosti na posunu po mřížce

Nejvyšší rychlost byla 6 vteřin při posunu po mřížce o velikosti 32. Při tomto posunu nebyl objekt nalezen zcela ideálně, byl o několik stupňů pootočen, ale stále se z něj dalo správně vyčíst potřebné údaje. U ostatních měření byl hledaný obraz nalezen bez problémů.

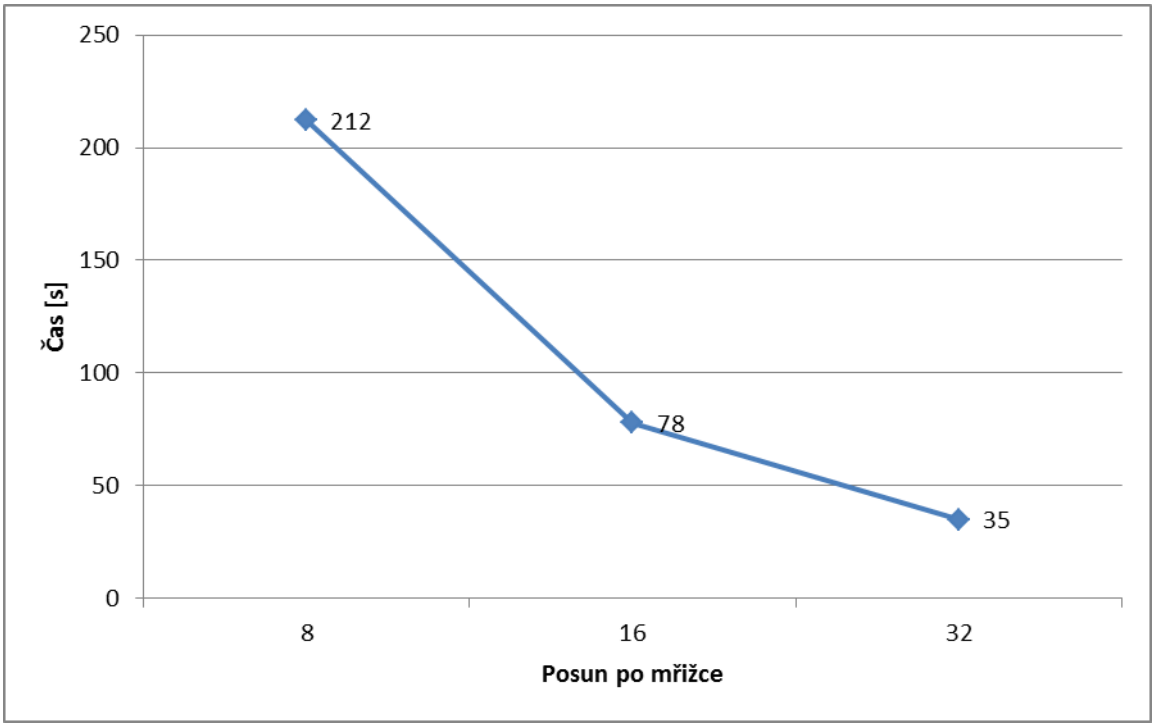
Pro druhý test byl zvolen obrázek o velikosti 500x500 pixelů. Na obrázek byly umístěny dva objekty stejného typu, oběma byla náhodně změněna velikost a úhel natočení. Počet bílých bodů vzrostl na 2990. Knihovna obsahuje stále jeden vzor hledaného objektu.



Obrázek 26. Graf znázorňující rychlost hledání v závislosti na posunu po mřížce

Ve druhém testu se doba hledání prodloužila. Při posunu po mřížce o 32 se vyskytl opět problém, že objekt byl identifikován pod trochu jiným úhlem. Při posunu o 16 a 8 byl objekt přesně nalezen. Při posunu o 4 byl objekt špatně identifikován. Byl mu přiřazen o 30 stupňů jiný úhel natočení, velikost ale byl správná a dalo se z ní získat správné vzdálenosti.

Pro třetí test byl zvolen obrázek o velikosti 500x500 pixelů. Na obrázek byly umístěny dva objekty různého typu, oběma byla náhodně změněna velikost a úhel natočení. Počet bílých bodů opět vzrostl, a to na 4462. Knihovna obsahuje dva vzory hledaných objektů.



Obrázek 27. Graf znázorňující rychlost hledání v závislosti na posunu po mřížce

Jak je vidět z grafu, doba hledání se značně prodloužila. Při posunu po mřížce 32 byl jeden objekt identifikován opět pod špatným úhlem. Druhý objekt nebyl nalezen. Při posunu o 16 byly úspěšně nalezeny oba hedané objekty, dále ovšem byly identifikovány další dva objekty, které se v obraze nevyskytovaly. Při posunu po přížce o 8 byly opět správně identifikovány dva objekty, dále byly nalezeny ještě 3 další objekty, které se v obraze nevyskytovaly.

6.2 Testování na obrázcích z mikroskopu

Pro testování byly vybrány dva obrázky, které mají velikost 800x600. Knihovna obsahuje pouze jeden vzorový organismus. Na prvním snímku je organismus s minimálním šumem a jinými nežádoucími vlivy. Na druhém je organismus, kolem kterého se objevují další jiné částečky, které zhoršuj možnost rozpoznávání.

První snímek po prahování obsahuje 10498 bílých bodů. Díky tomu, že obrys částečně splývá s pozadím, je těžké oddělit hrany, a proto je získáno tolik bílých bodů. Proto byl

zvolen posun po mřížce 32. Hledání trvalo 801 vteřin. Byl nalezen správný obrys, ale také díky šumu dalších 5 nežádoucích nálezů.

Druhý snímek po prahování obsahoval 5931 bílých bodů. Proto mohl být zvolen pohyb po mřížce o velikosti 16. Hledání bylo dokončeno za 98 vteřin. Na tomto obrázku se nepodařilo objekt přesně najít. Byly nalezeny dva objekty odpovídající tvaru organismu ve správné části obrázku, ale jejich velikost a úhel natočení nesouhlasily.

6.3 Závěry z testování

Z testování vyplývá, že rychlost a přesnost nalezení objektu na obrázku je závislá na kvalitě prahování, které určuje množství bílých bodů. Možnost správného prahování je závislá na kvalitě pořízeného snímku a na schopnosti detekce hran. Vyhledávání dále urychluje správné nastavení Houghovo transformace. Při nastavení pohybu po mřížce 32 je vyhledávání velice rychlé, ale dochází k chybovosti. Tato velikost krokování se dá využít pro předběžné prohledávání. Jako ideální pro rychlost vyhledávání a správné nalezení objektu se zdá být správné nastavení pohybu po mřížce o velikosti 16. Dále bylo zjištěno, že neexistující objekty, které byly rozeznány, byly nalezeny na základě blízkosti jeden druhého a při hledání se použila část z jednoho objektu a část z druhého. Chybějící část mezi jednotlivými objekty byly dostatečně malé na to, aby splňovala požadavky pro rozeznání. Bylo také zjištěno, že naimplementovaný hranový detektor vytahuje hrany málo. Toto způsobuje, že nejsou dostatečně kvalitní obrysy objektů a může docházet k problémům při rozeznávání.

7 Návrh pro budoucí řešení

Knihovna pro zpracování obrazu Image processing library má implementovány jen základní metody pro zpracování obrazu. Do této knihovny lze naimplementovat další metody pro zpracování obrazu, které by urychlily a zpřesnily proces zpracování. Důležitou metodou by mohl být jiný způsob pro nalezení hran objektů, který hraje důležitou roli před procesem prahování.

Dále je možné provést řadu optimalizací na Houghovo trasformaci, pro urychlení a zpřesnění hledání, a to v místech kde se aplikace prohledává místo na obrázku, kde se hledané objekty nemůžou vyskytovat. Další možností je také vytvořit jinou techniku pro nalezení hledaného objektu.

Pro urychlení hledání lze také kromě optimalizace využít možnost grafické akcelerace, která pro výpočet využívá grafického procesoru na grafické kartě. Tyto grafické karty mají ve specifických výpočtech výkonnější procesor než je hlavní procesor počítače.

Lze také implementovat možnost zobrazení obrazu z mikroskopu v reálném čase pro snadnější a pohodlnější manipulaci. Po připojení mikroskopu k počítači by se zobrazil obraz přímo v aplikaci, ve které by zaznamenávala snímky, a na tyto pořízené by se pak opět aplikovaly metody pro zpracování obrazu.

8 Závěr

Byla navržena programová knihovna pro měření a rozpoznáván mikroskopických živočichů na obrázcích z mikroskopu. Do této knihovny bylo implementováno několik metod pro zpracování obrazu. Tyto metody jsou Gaussův filtr, obecný hranový detektor, prahování a Houghovo transformace. Jednotlivé metody byly úspěšně otestovány na jednotlivých obrázcích. Tato knihovna byla použita v jednoduché aplikaci. Výsledná aplikace je schopna rozpoznávat mikroskopické organismy. Přesnost a rychlost hledání mikroskopických organismů závisí na nastavení parametrů u prahování a Houghovo transformace. Čím větší je práh u prahování, tím se zobrazí méně bílých bodů. Tím se zvyšuje rychlost vyhledávání, protože Houghova transformace nemusí porovnávat tolik bílých bodů. Jejich nízký počet ovšem může způsobit to, že hledaný objekt nemusí být nalezen. U Houghovi transformace závisí rychlost na nastavení rozptylu a na velikosti kroků, pomocí kterých se prochází obrázek. S většími kroky dojte k rychlejšímu hledání, ale může se stát, že hledaný objekt nebude nalezen. Použití Houghovo transformace má za následek delší dobu hledání, jak ukazují testy. Tato doba lze ovšem zkrátit pomocí různých optimalizací, které jsou uvedeny v Návrhu pro budoucí řešitele.

Byl vytvořen manuál k aplikaci, který popisuje jak aplikaci pro transformaci tak aplikaci pro rozpoznávání. Jsou zde dále popsány postupy pro práci s aplikacemi. Manuál je přiložen na CD v elektronické podobě.

Seznámil jsem se s metodami pro zpracování obrazu a zpracoval jsem rešeršní část zadání. Následně jsem vytvořil aplikaci a programovou knihovnu pro měření a rozpoznávání mikroskopických organismů.

9 Použitá literatura

[1] *Image Processing*: *The Fundamentals*. 2. Chichester, West Sussex (United Kingdom): Wiley, 2010. 2. ISBN 978-0-470-74586-1. Dostupné z: http://books.google.cz/books?id=w3BpSIxN9ZYC

[2] *Microscope Image Processing*. Oxford: Academic press, 2008. ISBN 978-0-12-372578-3. Dostupné z: http://books.google.cz/books?id=uGWmR0f_350C

[3] *Obrazové segmentační techniky*: *Přehled existujících metod*. Brno, 2005. Dostupné z: http://www.fit.vutbr.cz/~spanel/segmentace/

[4] *Základní metody předzpracování obrazu*. Praha, 2003. Dostupné z: http://webzam.fbmi.cvut.cz/hozman/Zprac_obr_prisp_kurz_UEM_3_2003.pdf

[5] GNU Image Manipulation Program. *Http://docs.gimp.org* [online]. [cit. 2012-01-06]. Dostupné z: http://docs.gimp.org/2.2/cs/plug-in-convmatrix.html

[6] *Segmentace obrazu* [online]. Brno, 2009 [cit. 2011-12-31]. Dostupné z: http://is.muni.cz/th/72784/fi_m/dp.pdf. Diplomová práce. Masarykova univerzita.

[7] *Mediánové filtry a jejich použití při filtraci impulsních šumů*. Plzeň, 2003. Dostupné z: http://webs.zcu.cz/fel/kae/+eln/Median02.pdf

[8] *Převod černobílých obrázků* [online]. Brno, 2010 [cit. 2012-01-08]. Dostupné z: http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=28724. Bakalářská práce. Vysoké učení technické v Brně.

[9] *Digitální filtry pro obrazová data* [online]. Brno, 2010 [cit. 2012-02-19]. Dostupné z: http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=31349. Bakalářská práce. Vysoké učení technické v Brně.

[10] Sobel Edge Detector. *Http://homepages.inf.ed.ac.uk/* [online]. 2004 [cit. 2012-02-19]. Dostupné z: http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

[11] *Rapid-i* [online]. 18.10.2011 [cit. 2011-02-02]. Integrating RapidMiner into your application. Dostupné z WWW: <http://rapidi.com/wiki/index.php?title=Integrating_RapidMiner_into_your_application>

[12] *Laral.istc.cnr.it* [online]. 2003 [cit. 2011-02-02]. Neural Networks Library In Java. Dostupné z WWW: http://laral.istc.cnr.it/daniele/software/NNLibManual.pdf

[13] Počítačová grafika. *Http://cg.tucna.net/* [online]. 26.11.2011 [cit. 2012-02-22]. Dostupné z: http://cg.tucna.net/2011/11/diskretni-dvourozmerna-konvoluce/

[14] Open-source projects for image processing. In: *Zpracování signálů, obrazu dat* [online]. 2011 [cit. 2012-02-24]. Dostupné z: http://www.feec.vutbr.cz/EEICT/2011/sbornik/01-Bakalarske%20projekty/02-Zpracovani%20signalu,%20obrazu%20a%20dat/05-xslint01.pdf

[15] OpenCV Wiki. *OpenCV* [online]. 15.2.2012 [cit. 2012-02-24]. Dostupné z: http://opencv.willowgarage.com/wiki/

[16] *Houghova transformace pro detekci čar* [online]. Brno, 2009 [cit. 2012-02-24]. Dostupné z: http://www.fit.vutbr.cz/study/DP/rpfile.php?id=8869. Bakalářská práce. Vysoké učení technické v Brně.

[17] Culver, D. A., Boucherle, M. M., Bean, D. J. & Fletcher, J. W., 1985. Biomass of freshwater crustacean zooplankton from length-weight regressions. Canadian Journal of Fisheries and Aquatic sciences 42: 1380-1390

10 Přílohy

[A] Manuál k aplikacím

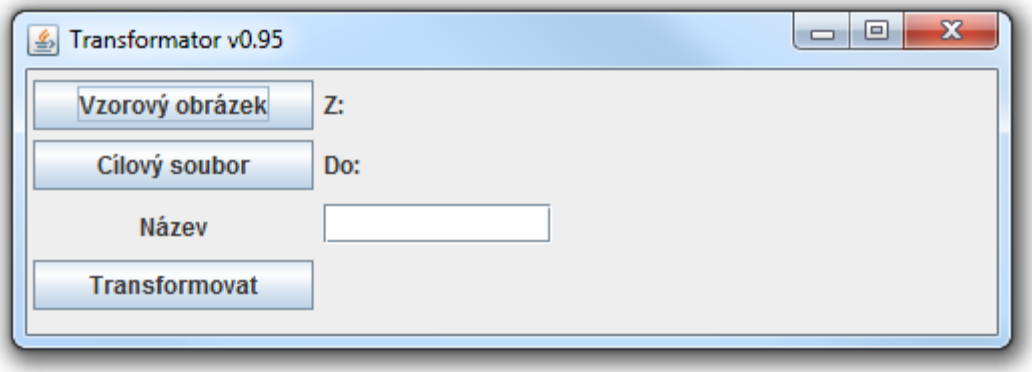
[B] CD s aplikací a jejími zdrojovými kódy, s dokumentací k vytvořené aplikaci, s textem bakalářské práce a s manuálem aplikací v elektronické podobě.

11 Manuál k aplikacím

11.1 Aplikace pro transformaci obrázků

11.1.1 Popis aplikace

Aby bylo možné rozpoznávat objekty na obrázcích, je třeba nejprve získat prototyp objektu (vzorový objekt) a ten upravit, aby se dal použít jako vzor. K tomuto slouží aplikace Transformátor.



Obrázek 28. GUI aplikace pro transformaci

Jak můžete vidět, aplikace je velmi jednoduchá a skládá se jen z pár prvků.

1. Tlačítko „Vzorový obrázek“ – Pomocí tohoto tlačítka si volíte prototyp obrázku, který se má transformovat. Vzorový obrázek musí být typu bmp, kvůli zachování kvality.
2. Tlačítko „Cílový soubor“ – Pomocí tohoto tlačítka zvolíte název souboru a adresář, do kterého se má obrázek uložit.
3. Do kolonky název napíšete název objektu na vrozovém obrázku.
4. Tačítko „Transformovat“ – Pomocí tohoto tlačítka se spustí transformace obrázku. Aplikace oznámí, zda proběhla transformac v pořádku.

Důležité:

Vzorová obrázek musí být typu bmp.

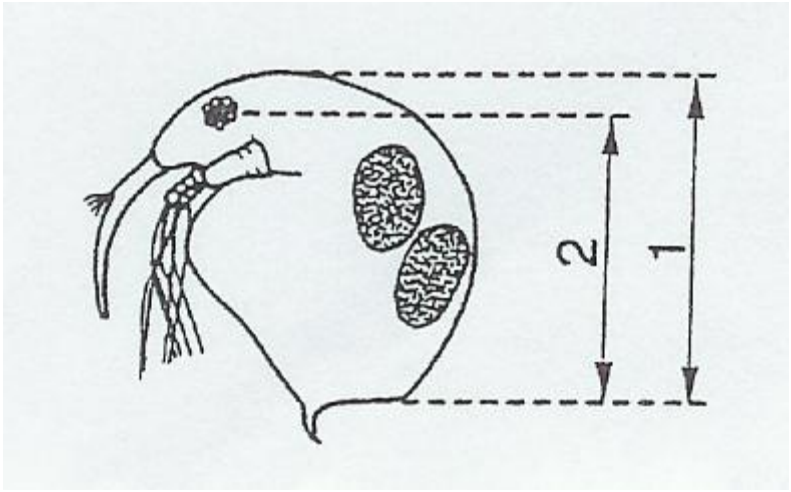
Cílový soubor by mě mít koncoku txt (není nutností).

Složka, ve které jsou upravené prototypy obrázků, nesmí obsahovat nic jiného.

11.1.2 Postup úpravy obrázku

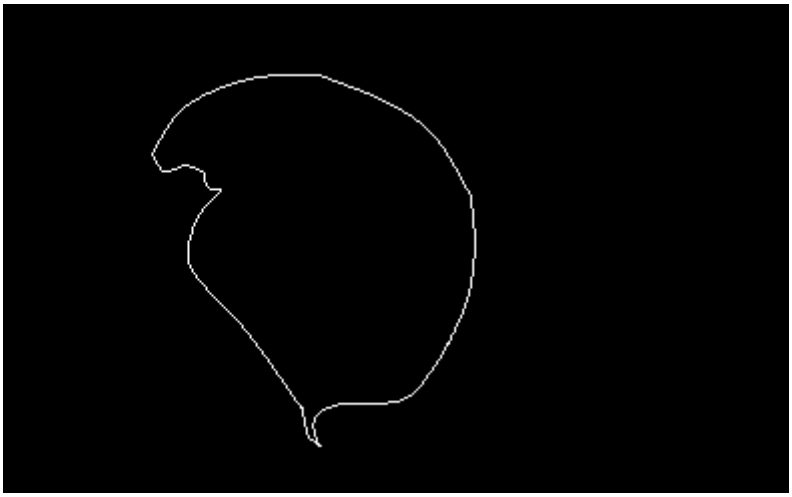
Postup úpravy obrázku bude demonstrován na jednoduchém příkladu

1) Nejprve je třeba zvolit prototyp obrázku



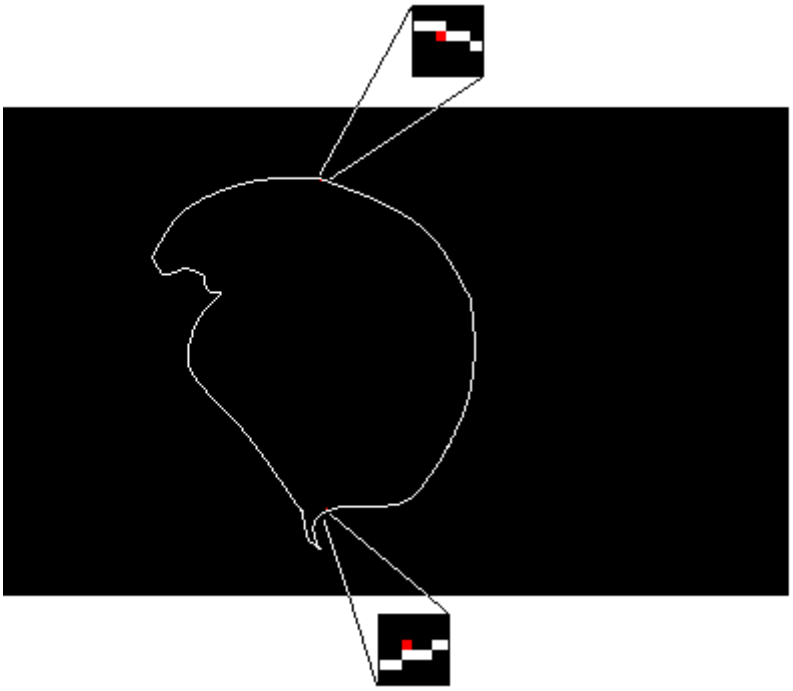
Obrázek 29. Organismus Bosmina [1]

2) Tento obrázek je třeba pomocí libovolného grafického editoru upravit, a to tím způsobem, že pozadí je černé a obrys organismu je bílou barvou. Měl by se vyznačit takový obrys, který není variabilní. Neměly by se zakreslovat například „štětiny“ a jiné podobné výčnělky, které má každý exemplář organismu individuální.



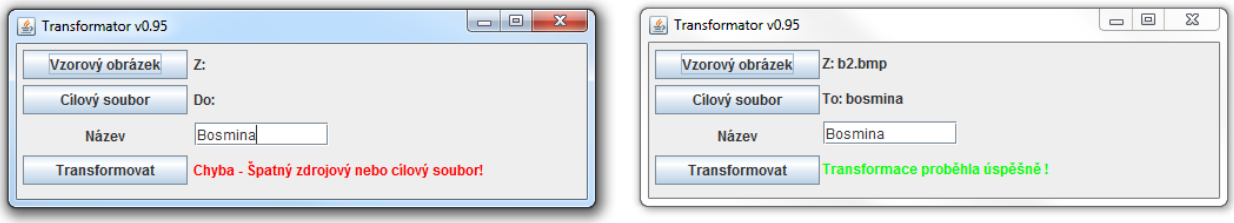
Obrázek 30. Obrys organismu

3) Dalším krokem je v grafickém editoru vyznačit vzdálenosti, které se mají měřit. To se provádí tak, že se vzdálenost vyznačí dvojicí barevných bodů. Pro vyznačení jiné vzdálenosti se musí zvolit jiná barva dvojice bodů. Velikost barevných bodů musí být 1px.



Obrázek 31. Vyznačené body pro měření

4) Tento obrázek musíte uložit jako bmp. Uložený obrázek použijte jako vzorový v aplikaci Transformátor.
5) Poté navolte cílový soubor
6) Vyplňte název organismu (v tomto případě Bosmina)
7) Použijte tlačítko transformovat

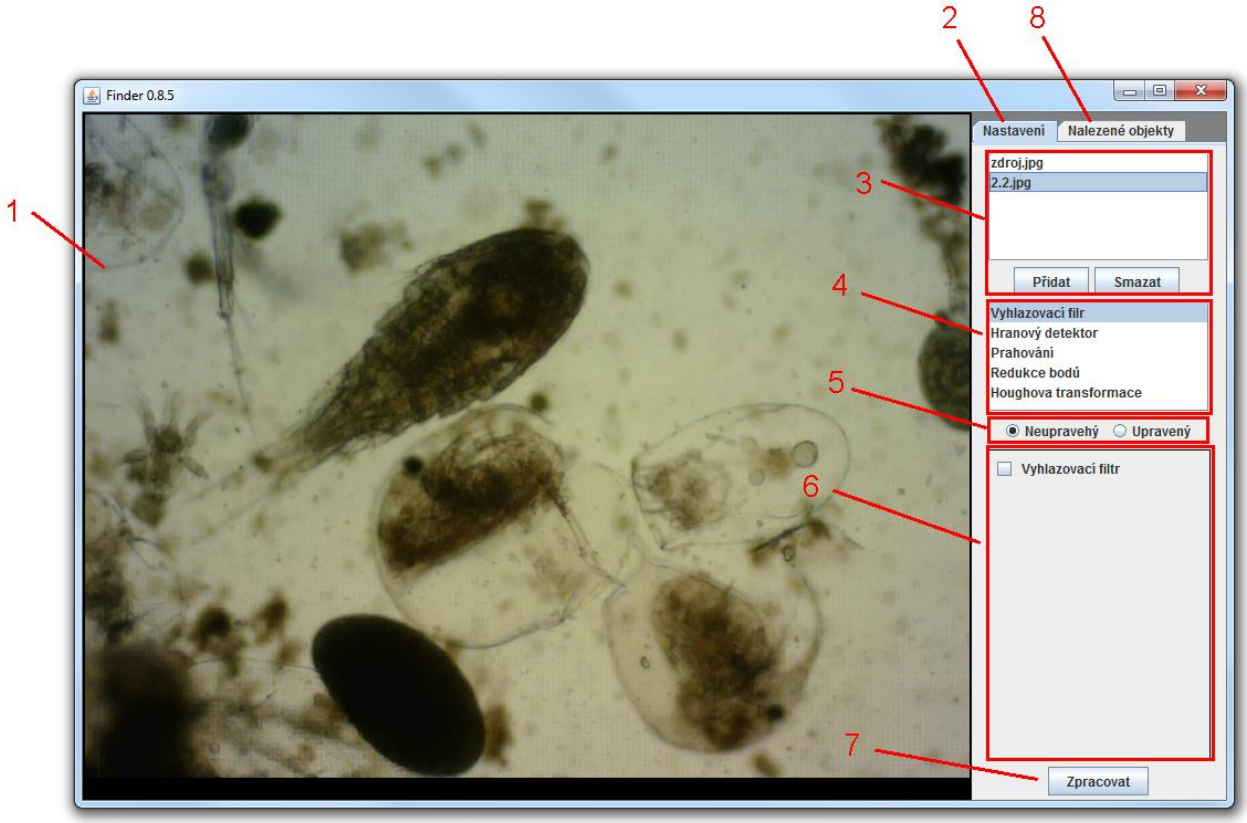


Obrázek 32. Neúspěšný převod a úspěšný převod

11.2 Aplikace pro rozpoznávání mikroskopických organismů

11.2.1 Popis aplikace

Tato aplikace vyhledává objekty, které byly vytvořeny pomocí aplikace Transformátor.



Obrázek 33. GUI aplikace pro rozpoznávání

1) Zobrazovací prostor, který zobrazí obrázek ve velikosti 800x600. Pokud by se měl zobrazit větší obrázek, tak je zmenšen na tuto velikost, ale jen pouze pro zobrazení. Pro výpočty zůstává obrrázk ve stejném rozlišení.
2) Záložka nastavení
3) V tomto seznamu se vybírají obrázky, na kterých se hledají objekty. Pomocí tlačítka „Přidat“ lze přidat další obrázky. Tlačítkem „Smazat“ se odebere vybraný obrázek.
4) Zde je seznam filtrů, které se musí aplikovat, aby byl zdrojový obrázek vhodně upravený pro rozpoznávání.
5) Dvojice tlačítek, které umožnuje zobrazení upraveného a neupraveného zdrojového obrázku.
6) Oblast, kde se objevuje nastavení jednotlivých filtrů. Každý filtr obsahuje zaškrtávací tlačítko, pomocí kterého se dá aktivovat a deaktivovat aplikace filtru. Toto je vhodné,

když se snažíte vhodně nastavit parametry. (Redukce bodů je jediný nepovinný filtr, není defaultně aktivní)

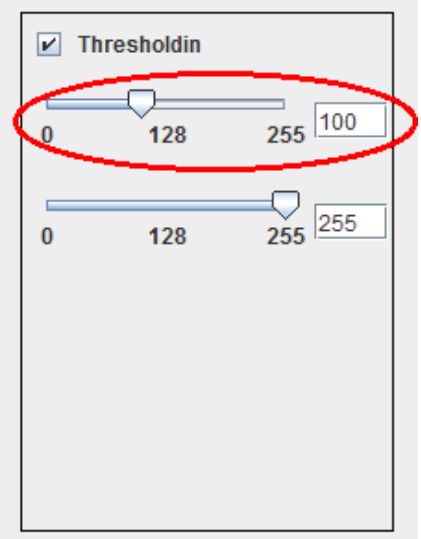
7) Tlačítko „Zpracovat“ spustí po kliknutí proces postupné aplikace všech filtrů na obrázek.
8) V záložce „Nalezené objekt“ se zobrazí seznam nalezených objektů.

11.2.2 Popis filtrů

1) Vyhlazovací filtr a Hranový detektor mají jen tlačítko na aktivaci a deaktivaci.

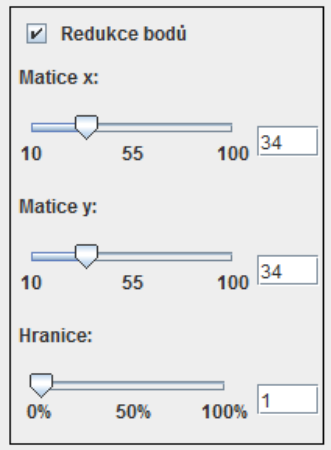
Obrázek 34. Tlačítka pro aktivaci

2) Prahování je metoda, která převádí obrázek na černobílý. Jeho nastavení má dva slidery. První nastavuje dolní práh a druhý nastavuje horní práh. Pro využití hledání organismů je doporučeno pohybovat jen s prvním sliderem, který poskytuje uspokojivé výsledky.



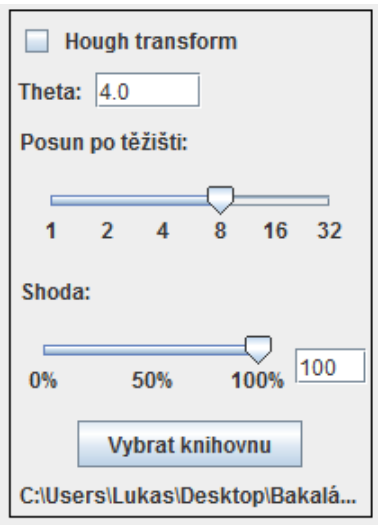
Obrázek 35. Nastavení prahování

3) Redukce bodů je metoda, která prochází zdrojový obrázek po určitých velikostech a pokud v dané oblasti je počet bílých bodů menší než zvolená hranice, pak se zbarví na černou.



Obrázek 36. Nastavení Redukce bodů

4) Houghova transformace je metoda, která vyhledává objekty. Theta je parametr, kterým se nastavuje tolerance shody nalezeného objektu se vzorovým objektem. Posunem po těžišti se nastavuj kroky, po kterých se prochází zdrojový obraz. Čím větší je tato hodnota, tím kratší dobu trvá hledaní, ale může nastat, že objekt nebude nalezen. Toto se dá do určité míry redukovat pomocí parametru Théta. Parametr Shoda určuje shodu pro nalezený objekt se vzorovým obrázkem. Tato hodnota by se měla pohybovat kolem 80 %, záleží ovšem na kvalitě prahování. Tlačítkem „Vybrat knihovnu“ se vybírá složka, která obsahuje vzorové soubory. Tyto soubory byly vytvořeny pomocí aplikace Transformátor.



Obrázek 37. Nastavení Houghovo transformace

11.2.3 Popis postupu nastavení aplikace pro vyhledávání

1) Tlačítkem „Přidat“ přidejte obrázek, na kterém chcete vyhledávat objekty.
2) Ze seznamu vyberte obrázek.
3) Přepněte tlačítko na upravený, aby jste mohl vidět vzhled obrázku po aplikaci filtrů.
4) V seznamu filtrů vyhlazovací filtr a hranový detektor jsou implicitně aktivovány a nemají žádné parametry, proto se o ně nemusíte zajímat. Vyberte tedy Prahování a měňte dolní práh pro co nejlepší výsledek. Aby byla vidět změna nastavení prahu, je nutné použít tlačítko „Zpracovat“.
5) Vyberte redukce bodů a aktivujte ho. Jeho popis je výše. Jak již bylo zmíněno, filtr není povinný.
6) Pokud jste spokojeni s výsledkem aktivujte Houghovo transformaci. Po nastavení parametrů klikněte na „Zpracovat“ a počkejte na dokončení vyhedávání. Tato činnost může chvíli v závislosti na nastevení parametrů a velikosti knihovny.

Pokud objekt nebyl nalezen:

Zkontrolujte, zda-li je vybrána správná knihovna.

Upravte dolní práh u prahování.

Upravte nastavení Houghovo transformace.

11.3 Použitá literatura

[1] Culver, D. A., Boucherle, M. M., Bean, D. J. & Fletcher, J. W., 1985. Biomass of freshwater crustacean zooplankton from length-weight regressions. Canadian Journal of Fisheries and Aquatic sciences 42: 1380-1390