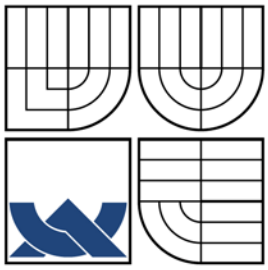


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

Zpracování obrazu mikroskopických vzorků

MICROSCOPE IMAGE PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. ONDŘEJ JANDA
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADOMIL MATOUŠEK, Ph.D.
SUPERVISOR

BRNO, 2009

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2008/2009

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Ondřej Janda

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Zpracování obrazu mikroskopických vzorků

v anglickém jazyce:

Microscope Image Processing

Stručná charakteristika problematiky úkolu:

Při biochemických testech, se uplatňuje metoda založená na počítání mikroorganismů, resp. jejich abnormalit v daném objemu média. Tento proces je prováděn manuálně, tj. v jistých případech se subjektivním vlivem pozorovatele. Výsledek DP by měl nejen urychlit daný proces, ale hlavně objektivizovat dané měření. DP se bude zabývat rozpoznáváním mikrobiologických vzorků, se zaměřením na identifikaci kvasinek v Burkerově komůrce. Vzorky (obrazy z CCD kamery) a případné konzultace z oboru biochemie jsou zajištěny pracovištěm ÚCHPBT FCH VUT v Brně.

Cíle diplomové práce:

- Seznámit se s problematikou zpracování obrazu prostřednictvím prostředí Matlab (Image Processing Toolbox) a stručně popsat použité metody.
- Vytvořit sadu funkcí využitelných k identifikaci objektů typu Burkerova komůrka, mikrobiologický vzorek, spojeni dvou obrazů.
- Navrhnout GUI pro danou aplikaci.
- Vytvořit adekvátní uživatelskou elektronickou dokumentaci.

Seznam odborné literatury:

• Gonzalez, R.C., Woods, R., Eddins, S.L.: Digital Image Processing Usány Matlab, Prentice Hall, ISBN-13 978-0130085191, 2004.

Vedoucí diplomové práce: Ing. Radomil Matoušek, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2008/2009.

V Brně, dne 28.11.2008

L.S.

_____	_____
doc. RNDr. Ing. Miloš Šeda, Ph.D.	doc. RNDr. Miroslav Doupovec, CSc.
Ředitel ústavu	Děkan fakulty

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:

Bytem:

Narozen/a (datum a místo):

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta strojního inženýrství

se sídlem Technická 2896/2, 616 69 Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP: _____

Vedoucí/ školitel VŠKP: _____

Ústav: _____

Datum obhajoby VŠKP: _____

VŠKP odevzdal autor nabyvateli v*:

- □ tištěné formě – počet exemplářů ……..
- □ elektronické formě – počet exemplářů ……..

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: …….

…….. ……

Nabyvatel Autor

Abstrakt

Práce se zabývá zpracováním obrazových dat mikrobiologických snímků. Jsou probrány základní a použité metody zpracování obrazu. Dále je uveden teoretický biologický úvod do zkoumaných dat. Nedílnou součástí je aplikace v prostředí Matlab aplikující tyto metody na skutečná data. Součástí zadání je i tvorba uživatelského prostředí a poskytnutí dokumentace.

Abstract

This diploma thesis deals with image processing of microbiological data. Firstly the brief introduction to biological basics is given. Secondly we take focus on basic and used image processing methods. Integral part of this thesis is working application witch uses image processing methods on input data. Creating user interface for this application is also part of this work. Documentation is provided.

Klíčová slova

Kvasinky, Bürkerova komůrka, zpracování obrazu, jasové transformace, prahování, segmentace, fázová korelace, Houghova transformace, Matlab

Keywords

Yeast, Bürker’s cell, image processing, brightness transformations, tresholding, segmentation, phase correlation, Hough transform, Matlab

Poděkování

Za účinnou podporu a obětavou pomoc, cenné připomínky a rady při zpracování diplomové práce tímto děkuji vedoucímu bakalářské práce, panu Ing. Radomilu Matouškovi Ph.D.

Dále bych velmi rád poděkoval mým rodičům za jejich stálou podporu, trpělivost a lásku. Bez nich by tato práce nemohla nikdy vzniknout.

Brno, Ondřej Janda
28.května.2009

Obsah:

1 Úvod

Cílem této práce je navržení softwaru pro obrazovou analýzu mikroskopických snímků kvasinek. K dispozici bylo poskytnuto několik sad snímků. Ty jsou použity jako vstupy do aplikace a postupně jsou spojeny do celku, rozloženy na jednotlivé zkoumané části a analyzovány.

Na navržený algoritmus není apriorně kladen žádný časový předpoklad, jeho účelem by mělo být přenesení procesu analýzy z člověka na software, respektive počítač a tím dosažení objektivizace výsledků. Vzhledem k nesčetné rozmanitosti vstupních dat je nutné zaveddení globálních pravidel pro všechny zkoušená data s cílem dosáhnout určité generalizace řešení problému, a tím pádem i možné opakovatelnosti při dosažení stejných výsledků. Dle zadání je tato práce vypracována v prostředí Matlab, které sice neprovádí všechny výpočty optimální rychlostí, ale je výhodné zvláště pro průzkum, návrh a ověření použitých algoritmů. Práce obsahuje jak část textovou, která zahrnuje podrobnější popis použitých technik, tak i část aplikační, kde jsou použity algoritmy obrazové analýzy na konkrétních mikroskopických snímcích.

Kapitola (2) naskýtá úvod do biologické části zkoumané problematiky. Popsány jsou vlastní zkoumané objekty, jejich skladba, rozmnožovací procesy a v neposlední řadě i konvenční metody jejich zkoumání. V kapitole (3) je uveden základní úvod do zpracování obrazu. Především zde jsou popsány základní typy zpracovávaných dat.

V dalších kapitolách se už zabývám vlastními algoritmy pro zpracování obrazu. Nejprve jsou v kapitole (4) popsány jasové transformace a problematika histogramu. Kapitola (5) je zaměřena na metody filtrace a hranování. Následující kapitola (6) se zaměřuje na segmentaci, detekci čar, fázovou korelaci a základní uvedení do Fouriérovy transformace. Probírány jsou především algoritmy použité k řešení problému, podrobný popis všech metod je mimo rozsah této práce. U každé uvedené metody je uveden základní matematický popis a následně implementace v pseudokódu v syntaxi Matlabu.

Poslední kapitola (7) pojednává o vlastní aplikaci, která je realizována pomocí GUI rozhraní. Tato kapitola rovněž popisuje uživatelské funkce, uvádí jejich syntaxi, použití a funkcionalitu v procesu zpracování obrazu daných vzorků.

Práce zahrnuje:

- Seznámit se s problematikou zpracování obrazu prostřednictvím prostředí Matlab (Image Processing Toolbox) a stručně popsat použité metody.
- Vytvořit sadu funkcí využitelných k identifikaci objektů typu Burkerova komůrka, mikrobiologický vzorek, spojení dvou obrazů.
- Navrhnout GUI pro danou aplikaci.
- Vytvořit adekvátní uživatelskou elektronickou dokumentaci.

2 Biologická podstata problematiky

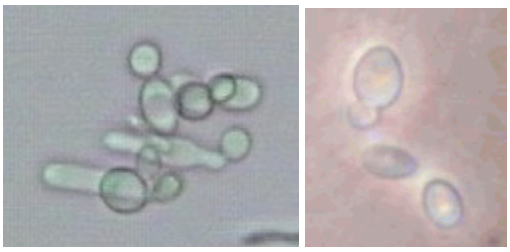
Tato kapitola poskytuje základní pohled do biologické části problematiky. Jsou uvedeny nezbytné informace o vlastních zkoumaných prvcích, jejich skladba a způsoby rozmnožování. Další důležitou částí je nastínění principu a typů metod počítání mikrobiologických entit ve zkušebních vzorcích.

Problematika počítání buněk je základem pro nesčetné množství biologických testů a procesů. Zajímá nás rozdíl v četnosti daných mikroorganismů před a po ukončení námi řízeného biologického procesu. Tento rozdíl se určuje manuálně počítáním buněk ve vstupních a výstupních vzorcích laborantem. Vzhledem k charakteru zkoumaných dat je náchylnost k nepřesnosti velice značná. Hlavním vlivem je neobjektivnost pozorovatele. V praxi dva lidé neprovedou analýzu stejného vzorku se shodnými výsledky. Dalším problémem je značná časová náročnost a monotónnost posuzovacího procesu. Tyto aspekty vedly k navržení diplomové práce, jejímž cílem bylo zmapovat danou problematiku a vytvořit nástroj pro objektivní posuzování dat.

2.2 Kvasinky

Kvasinky jsou heterotrofní eukaryotní mikroorganismy [2], náležící mezi houby. Název je odvozen ze schopnosti zkvašovat monosacharidy a některé disacharidy na ethanol a oxid uhličitý.

Tvar kvasinek souvisí se způsobem vegetativního rozmnožování, jenž se děje buď pučením nebo dělením. Nejčastější tvar je krátce elipsoidní [Obr.1], případně vejčitý až kulovitý. Některé rody tvoří pouze protáhlé buňky. Vyskytují se však i kvasinky tvaru citrónovitého, trojúhelníkového i válcovitého.

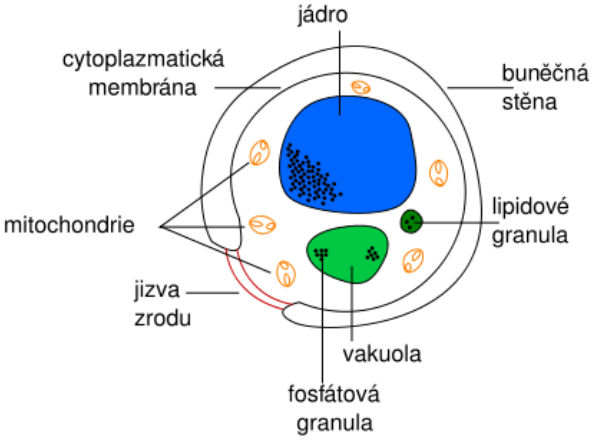


Obr.1 Protáhlé a elipsoidní kvasinky

Tvar buněk i jejich velikost jsou z větší části ovlivněny kultivačními podmínkami a stářím buněk. V dané kultuře buněk je však možná určitá variabilita tvaru a velikosti. Šířka většiny kvasinek je v rozmezí 3 – 6 μm.

2.3 Cytologie kvasinek

Vegetativní kvasinková buňka se skládá ze silné a pevné buněčné stěny, jemné cytoplazmatické membrány, cytoplazmy a jádra [Obr. 4]. Pohybové orgány vegetativní buňky kvasinek nemají.



Obr. 2 Průřez buňkou

2.3.1 Buněčná stěna

Tato stěna má silnou a pevnou strukturu, která dává buňce tvar a chrání ji před mechanickými vlivy a před osmotickým šokem. Velkými póry ve stěně mohou volně procházet všechny sloučeniny kromě sloučenin vysokomolekulárních, jako jsou polysacharidy a bílkoviny.

Hlavní složkou buněčné stěny kvasinek jsou polysacharidy představující asi 80 % sušiny stěny. Bílkoviny tvoří 6 – 10 % sušiny stěny. Zbylých 3 - 10 % tvoří lipidy, fosfolipidy nebo fosforečnany vázané na polysacharidy. Složení vnější stěny ovlivňuje sedimentační schopnosti kvasinek.

Na povrchu stěny kvasinek jsou patrné jizvy po pučení. Na jednom pólu každé buňky vidíme zvláštní jizvu, která zůstala v místě dřívějšího spojení buňky s buňkou mateřskou. Tato jizva se nazývá jizvou zrodu.

Obr. 3 Jizvy po pučení

2.3.2 Cytoplazmatická membrána

Cytoplazmatická membrána kvasinek má podobné složení jako membrána bakterií.
Je poměrně tenká, přibližně 7,5 – 8nm, složená z lipidů a proteinů. Vytváří vychlípeniny vybíhající do cytoplazmy. Je volně propustná pro malé molekuly bez náboje. Je centrem transportních mechanizmů pro příjem určitých látek buňkou nebo také pro vyloučení látek z těla buňky. Na rozdíl od bakterií neobsahuje dýchací enzymy a systém oxidační fosforylace.

2.3.3 Cytoplazma

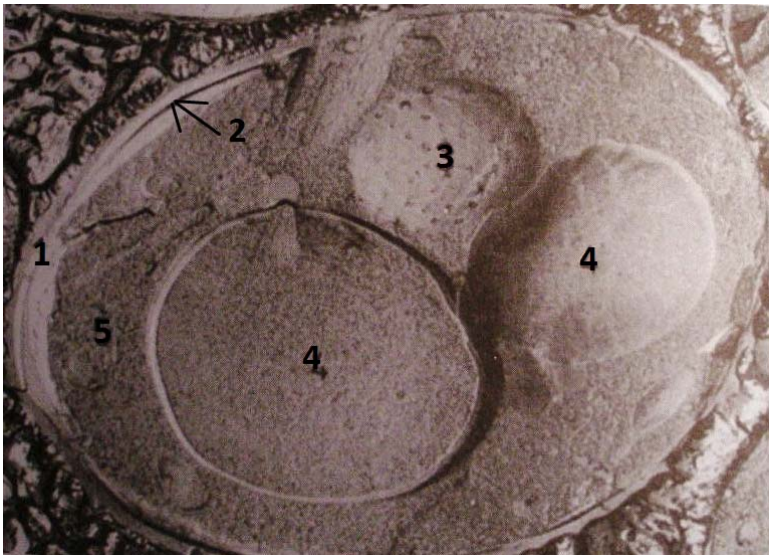
U mladých buněk kvasinek se ve světelném mikroskopu cytoplazma jeví jako průhledná, homogenní hmota, u starších buněk se objevují zrníčka a jemná nebo větší vakuolizace.

V cytoplazmě jsou také přítomny mitochondrie. Jsou to strukturované útvary velmi rozmanitého tvaru (kulovité, válcovité nebo laločnaté). Jsou 0,3 – 1 μm široké a až 3 μm dlouhé. Mitochondrie jsou složeny z bílkovin, lipidů a fosfolipidů. Obsahují RNA a složky DNA, která je nositelem mimojaderné dědičnosti kvasinek.

2.3.4 Vakuola

Patří k nejnápadnějším složkám cytoplazmy kvasinek. Většinou má kulovitý tvar obklopený jednoduchou membránou. U mladých nebo pučících kvasinek jsou často přítomny malé vakuoly a to ve velkém počtu. U zralých klidových buněk se vyskytuje převážně jedna vakuola. Starší buňky mají vakuolou vyplněný téměř celý vnitřek.

Vakuola obsahuje tekutinu, která propouští světlo, a proto má ve světelném mikroskopu stejný odstín jako pozadí zorného pole. Dále obsahují polyfosfáty, velkou zásobu draselných iontů, aminokyselin a purinů.



Obr. 4 Obsah buňky, zvětšeno 12000x, 1- buněčná stěna, 2- cytoplazmatická membrána, 3- jádro, 4-vakuola, 5- mitochondrie

2.3.4 Jádro

Jádro kvasinek je od cytoplazmy odděleno dvojitou jadernou membránou s velkými póry a je umístěno přibližně ve středu buňky. U nejlépe prostudovaných kvasinek bylo zjištěno 16 chromozomů v haploidním jádře. Diploidní jádro má dvojnásobný počet chromozomů neboť každý chromozom se v něm vyskytuje dvakrát.

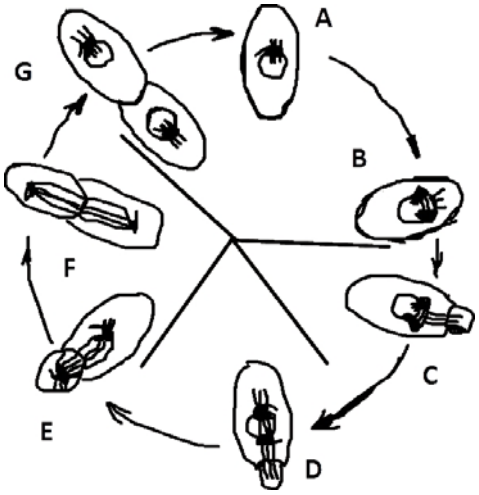
V jádře je také nízkomolekulární DNA, která má kruhovou strukturu a je obdobou plazmicidů bakterií. Dále je zde jadérko srpkovitého tvaru, uložené těsně pod jadernou membránou. Nachází se zde i pólové tělísko vřeténka. Má tvar disku a vychází z něj vlákna zvané mikrotubuly.

2.4 Mechanismus rozmnožování kvasinek

2.4.1 Vegetativní rozmnožování

Většina rodů kvasinek se vegetativně rozmnožuje pučením [3]. Při pučení vzniká malá dceřiná buňka spojená kanálkem s mateřskou buňkou tzv. pupen. Při pučení dochází k splývání membrán endoplazmatického retikula a pak k jeho dělení, dále k opakovanému dělení vakuol a ke změně tvaru mitochondrií v dlouze protáhlé. Z počátku tvorby dceřiné buňky do ní vstupují drobné vakuoly a mitochondrie. Současně probíhá mitotické dělení jádra a jeho migrace. Spolu s jádrem migrují i jiné prvky cytoplazmy. Pak se cytoplazmatickou membránou uzavře kanálek mezi dceřinou a mateřskou buňkou. Po vytvoření buněčné stěny mezi původní a nově vzniklou buňkou je pučení ukončeno.

Většina dceřiných buněk se ihned oddělí, ale ojediněle zůstanou spojeny a vznikají tzv. buněčné svazky. Tento celý proces, tj. od vzniku pupenu až po případné oddělení, trvá za optimálních podmínek kolem dvou hodin.

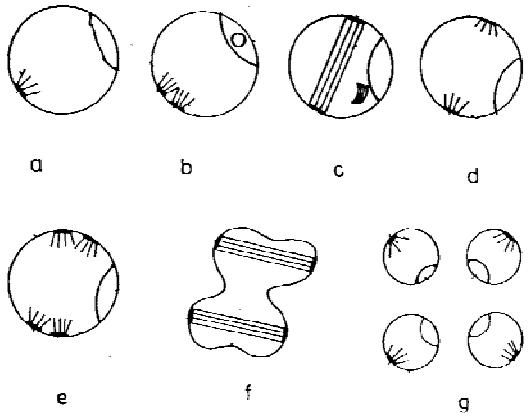


Obr. 5 Cyklus pučení. A-klidová buňka, B-zdvojení polárního tělíska, C-vznik malého pupenu, D-separace tělísek, E,F- migrace do pupenu, dělení jádra, G-vznik samostatných jader, dělení buněk

2.4.2 Pohlavní rozmnožování

Vedle vegetativního rozmnožování je u většiny kvasinek znám pohlavní způsob rozmnožování. Výsledkem tohoto procesu jsou pohlavní spory. Většina kvasinek tvoří jako pohlavní spory askospory, což jsou endospory umístěné ve vřeténku. Některé rody kvasinek však tvoří pohlavní exospory, tj. spory umístěné vně sporotvorných buněk.

Pohlavní rozmnožování je obecně charakterizováno spojením dvou haploidních buněk, čili konjugací a spojením jejich jader za vzniku diploidního jádra. Pak se toto jádro dělí mitózou, tj. redukčním dělením, ve čtyři haploidní jádra, která jsou buď základem pohlavních spor nebo se dělí další mitózou a pak teprve vznikají spory. V životním cyklu kvasinek se tedy střídá pravidelně fáze haploidní a diploidní.



Obr. 6 Jednotlivé fáze. a – jádro s jedním pólovým tělískem, b - rozdělení pólového tělíska, c – vytvoření vřeténka, d – rozpad vřeténka, e –rozdělení dvou pólových tělísek, f –vytvoření dvou vřetének, g – rozpad vřetének a vytvoření čtyř haploidních jader

2.5 Stanovení počtu buněk

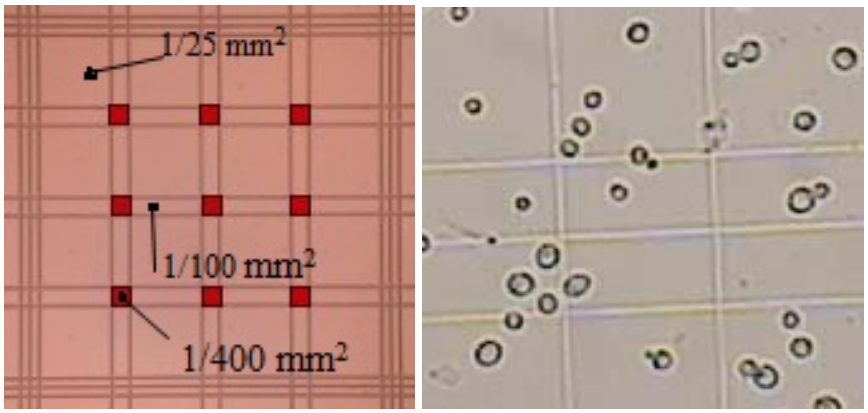
V mikrobiologické praxi jsme často postaveni před úkol měření růstu a množení mikroorganismů. Má to význam především v kvasném a chemickém průmyslu. V základním výzkumu slouží stanovení počtu buněk k posouzení kinetiky růstu, a ke stanovení specifické rychlosti růstu a množení v různých fázích jejich vývoje. V kontrolních laboratořích se využívá ke kontrole mikrobiálního znečištění surovin a výrobků a dokazují nám účinnost sterilizace.

Ke kvantitativnímu hodnocení počtu buněk se využívají dvě metody:

- přímé – mikroskopická
- nepřímá – kultivační

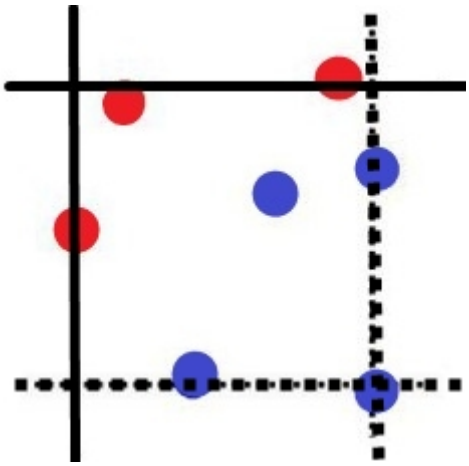
2.5.1 Přímá metoda stanovení počtu buněk

Pomoci přímé metody stanovujeme celkový počet buněk v jednotce objemu. Buňky se počítají pomocí speciálních komůrek – Bürkerovy nebo Thomovy. Bürkerova komůrka [Obr. 7] má tvar masivního podložního sklíčka, které je rozdělené dvěma rýhami na tři části. Celá mřížka je rozdělena na 25 menších čtverečků, které jsou dále děleny na 16 čtverečků. Rozměry jednotlivých čtverečků jsou 1/25mm^2 , 1/100 mm^2 a 1/400 mm^2. Při hloubce komůrky 0.1 mm je objem nad každým čtverečkem 1/250mm^3 , 1/1000 mm^3 a 1/4000 mm^3 . Po obou stranách vybroušených políček jsou odvodové kanálky.



Obr. 7 Plošné rozměry v jednom segmentu Bürkerovy komůrky (vlevo). Snímek části komůrky pod mikroskopem (vpravo).

Pro počítání buněk platí následující pravidlo: před začátkem počítání si vybereme dvě sousední hrany čtverce. Všechny buňky, které se pak těchto hran dotýkají, do celkového počtu zahrneme. Ty které se nedotýkají naopak nepočítáme. Buňka se také nepočítá, pokud leží na vybraných hranách, ale současně se dotýká hrany nevybrané. V této práci jsou vždy vybrané hrany spodní a pravé, tak jak je znázorněno na obrázku níže.



Obr. 8 Pravidlo výběru. Modrá – počítáme, červená - nepočítáme

Hodnocení

Počet buněk kvasinek v 1 mm^3 , největšího čtverce komůrky s objemem 1/250 mm^3

$$X = \frac{c * z * 250 * 1000}{P} \tag{2.1}$$

X – počet mikroorganismů v cm^3
c – celkový počet buněk kvasinek ve všech počítaných políčkách
z – číslo zředění
P – počet políček

2.5.2 Nepřímá metoda stanovení počtu buněk – kultivační

Kultivační stanovení počtu buněk se používá v kontrolních laboratořích potravinářského průmyslu, ve zdravotních zařízeních při sledování mikrobiologické čistoty vody, vzduchu, provozního prostředí atd..

Nejčastějším způsobem počítání buněk mikroorganizmů pomocí kultivace je počítání viditelných makroskopických kolonií vyrostlých na agarových plotnách. Metoda vychází z předpokladu, že z jedné životaschopné buňky vyroste jedna kolonie. Jednotlivé kolonie spočítáme a přepočteme na 1 ml původního vzorku.

Zanesení vzorku do agarového média je možné dvěma způsoby:

- očkováním vzorku a jeho rozetření
- přelitím potřebného objemu vzorku vytemperovaným agarem

Metodě rozetření dáváme přednost, jelikož umožňuje lepší rozptýlení buněk a lepší reprodukovatelnost výsledků než přelivem a nehrozí usmrcení buněk velmi horkým agarem. Nevýhodou těchto metod je delší čas potřebný k získání výsledku a to minimálně 24 hodin, a dále pak fakt, že v případě větších počtů typů mikroorganismů dostáváme většinou nižší počty buněk.

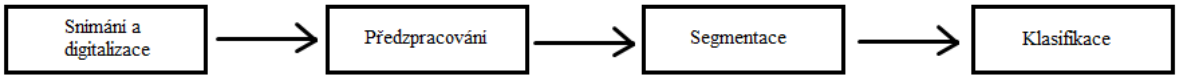
3 Zpracování obrazu

V oblasti počítačových technologií si pod pojmem zpracování obrazu můžeme představit jakousi formu zpracování signálu, kde vstupem je obraz. Výstupem tohoto procesu může být nový obraz nebo nějaké informace, které jsou ve vztahu k danému vstupu. Na většinu metod se můžeme dívat jako na techniky z oblasti zpracování signálu s tím rozdílem, že vstupem je obraz v podobě dvojdimenzionálního signálu.

Nejčastěji se mluví o tzv. digitálním zpracování obrazu, ale jsou možné i jiné formy např. optické nebo analogové zpracování obrazu.

Vlastní průběh zpracování obrazu se obvykle rozděluje do několika základních kroků. Rozdělení není jednoznačné a záleží na dané aplikaci problematiky, jaká posloupnost kroků bude zvolena. V našem případě by mohla být následující:

1. Snímání a digitalizace
2. Předzpracování
3. Segmentace
4. Klasifikace



Snímání a digitalizace není předmětem této práce. Do oblasti předzpracování obrazu můžeme zařadit: potlačení šumu, odstranění zkreslení a zvýraznění obrysů a hran. Segmentace se zabývá oddělením požadovaných dat od zbytku a redukcí nadbytečných dat. Klasifikace zpracovává výsledky z předchozích kroků.

V této kapitole si uvedeme základní poznatky pro práci z obrazem.

3.1 Digitální obraz

Digitální obraz je reprezentace dvojrozměrného obrazu jako konečný soubor digitálních hodnot - obrazové body.

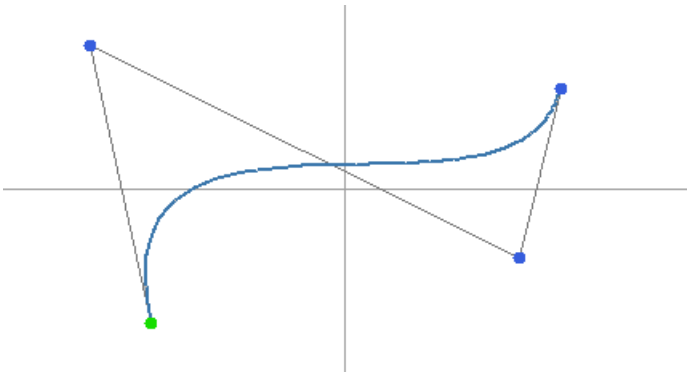
Obrazové body jsou typicky uloženy v paměti počítače jako rastrový obraz v podobě dvojrozměrné rastrové mapy obsahující množství malých celých čísel. Tyto hodnoty jsou často přenášeny nebo skladovány v komprimované formě.

Digitální obrazy mohou být vytvořeny různými vstupními zařízeními a technikou jako jsou digitální fotoaparáty, skenery, kamery atd.

Rozlišujeme dva typy obrazů:

Vektorový obraz

V případě vektorové grafiky je obraz reprezentován pomocí geometrických objektů (body, přímky, křivky, polygony).



Obr. 9 Vektorový obraz (Bezierova křivka)

Rastrový obraz

V bitmapové grafice je celý obrázek popsán pomocí jednotlivých barevných bodů (pixelů). Body jsou uspořádány do mřížky. Každý bod má určen svou přesnou polohu a barvu pomocí různých typů barevných modelů (např. RGB). Tento způsob popisu obrázků používá např. televize nebo digitální fotoaparát. Kvalitu záznamu obrázku ovlivňuje především rozlišení a barevná hloubka.

Některé typy rastrových obrazů :

- binární
- barevný
- šedotónový (monochromatický)
- multi-spektrální

Bez většího upřesnění se termínem digitální obraz rozumí obraz rastrový. V dalším budou popsány typy rastrových obrazů týkajících se této práce.

Reprezentace obrazu

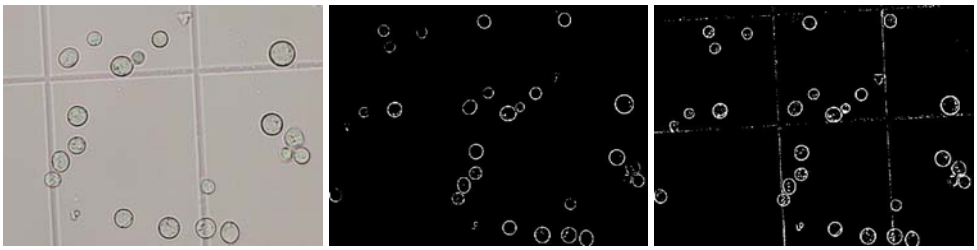
Při následném zpracování obrazu potřebujeme obraz a jasové hodnoty v něm vhodně reprezentovat. Obecně lze obraz definovat jako spojitou funkci dvou proměnných $f(x,y)$, kde f je hodnota jasu a x,y souřadnice pozice bodu v obrazu. Hodnota obrazové funkce může být jediné číslo, které reprezentuje hodnotu jasu na dané pozici. V případě barevného obrazu to mohou být hodnoty tři: $f_R(x,y), f_G(x,y), f_B(x,y)$, které reprezentují hodnotu jasu jednotlivých složek v modelu RGB.

3.1.1 Binární obraz

Binární obraz je digitální obraz, který má pouze dvě možné hodnoty pro každý pixel v rastrové mřížce. Zpravidla jsou tyto hodnoty bílá a černá, lze ale použít i jakékoliv jiné barvy.

Nejčastější využití je v technikách zpracování obrazu jako šablony pro další použití nebo jako výsledky operací s daným obrazem např. :

- prahování
- segmentace
- dithering



Obr. 10 Binární obrazy (zleva: originál, hodnota prahování 120, hodnota prahování 150)

Je patrné, že pro jeden digitální obraz je možné vytvořit více obrazů binárních. Nastavení funkčních hodnot je kritické pro výsledné zobrazení.

Reprezentace binárního obrazu

Binární obraz v této práci je reprezentován dvourozměrnou maticí - zde ji označme IMG

$$IMG = \begin{bmatrix} f(1,1) & f(1,2) & f(1,3) & \dots & f(1,N) \\ f(2,1) & f(2,2) & f(2,3) & \dots & f(2,N) \\ \dots & & & & \dots \\ f(M,1) & f(M,2) & f(M,3) & \dots & f(M,N) \end{bmatrix} \tag{3.1}$$

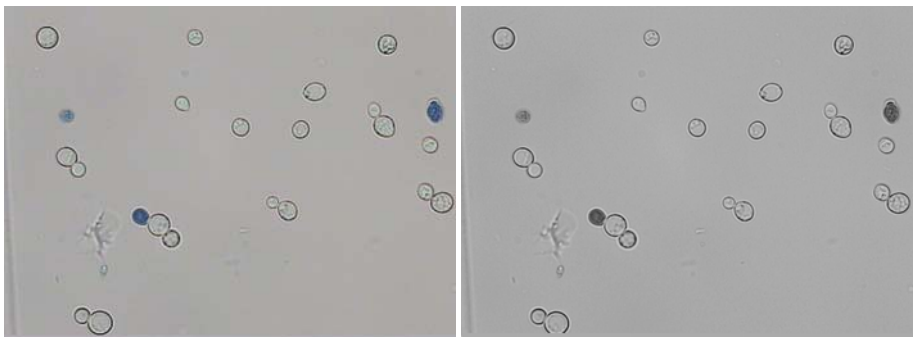
Kde x ∈ <1, …, M> a ∈ e <1, …, N>. Hodnota jasu *f* je buď 1 (bílá) nebo 0 (černá).

3.1.2 Šedotónový obraz

Šedotónový (monochromatický) digitální obraz je obraz, kde hodnota každého pixelu nese jen informaci o intenzitě. Tato intenzita znamená, že hodnoty pixelů jsou nejčastěji vyjádřeny jako odstíny šedi. To znamená schopnost vyjádřit 256 rozdílných intenzit světla nebo také 8 bitů na každý pixel. Možné jsou i jiné hodnoty např. 10, 12 či 16 (65,535 intenzit) bitu na pixel. V našem případě se setkáme pouze s 8 bitovou škálou, což znamená hodnoty v rozsahu 0 (černá) – 255 (bílá). Často tento typ obrazu nazýváme černobílý nebo monochromatický obraz.

Existuje několik způsobů jak získat tento typ obrazu:

- jako výsledek měření intenzity světla v jednom pásmu elektromagnetického spektra
- převedením z barevného obrazu



Obr. 11 Konverze barevného obrazu na obraz monochromatický

Převod barevného obrazu na monochromatický

Odstíny šedi zde vyjadřují hodnoty jasu. Lidské oko je různě citlivé na jednotlivé složky RGB modelu, a tudíž nelze použít prosté sečtení jednotlivých hodnot složek barev RGB. Je nutné použít váhových koeficientů. V našem případě jsou váhy rozloženy následovně:

výsledná hodnota jasu = 0,299R + 0,587G + 0,114B

kde R (červená), G (zelená), B (modrá) reprezentují hodnotu dané barvy v aktuálním pixelu.

Reprezentace monochromatického obrazu

Monochromatický obraz je zde reprezentován dvourozměrnou maticí, zde ji označme IMG

$$IMG = \begin{bmatrix} f(1,1) & f(1,2) & f(1,3) & \dots & f(1,N) \\ f(2,1) & f(2,2) & f(2,3) & \dots & f(2,N) \\ \dots & & & & \dots \\ f(M,1) & f(M,2) & f(M,3) & \dots & f(M,N) \end{bmatrix} \tag{3.2}$$

Kde x ∈ <1, …, M> a y ∈ <1, …, N>. Hodnota jasu *f* je v rozsahu <0,255>.

3.1.2.1 Implementace

```
funkce : ToGreyScale
vstup : barevny obraz(PicArray)
výstup : GrayPicArray(monochromatický
         obraz)

GrayPicArray = ToGreyScale (PicArray)
{
 for x=1 to width(PicArray)
      for y=1 to height(PicArray)
            red = PicArray(x,y,1) * 0.299;
            green = PicArray(x,y,2) * 0.587;
            blue = PicArray(x,y,3) * 0.114;
            intenzita = red + green + blue;

            GrayPicArray(x,y) = intenzita;

      end
end
}
```

Seznam často použitých funkcí

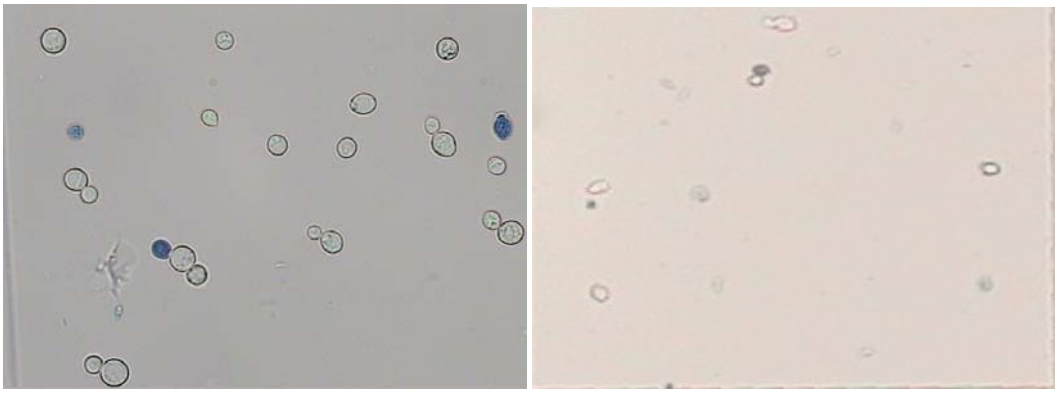
Tento seznam uvádí všechny použité funkce, které jsou implementovány ve vývojovém prostředí.

Název	Popis
width	vrací šířku vstupu
height	vrací výšku vstupu
length	vrací délku vstupu
pow	funkce počítá mocninu pow (základ, exponent)
max	vrací maximum ze vstupu
sqrt	funkce vrátí druhou odmocninu vstupu
sin	vrací sinus vstupu
cos	vrací kosinus vstupu
lines	kreslí úsečky do obrazu. Vstupem je obraz a sada souřadnic
fft2	2-D diskrétní Fourierova transformace
conj	součin F1 a komplexně sdružených čísel F2
abs	vrací absolutní hodnotu vstupu
ifft2	inverzní 2-D diskrétní Fourierova transformace
find	nalezení korespondujících x/y souřadnic globálního maxima
round	funkce zaokrouhluje vstup

Tab. 1 seznam často používaných funkcí

3.1.3 Barevný obraz

U barevného obrazu každý bod nese informace o hodnotách jasu všech tří RGB kanálů. To znamená, že hodnoty pixelů vyjadřují tři bajty, což znamená schopnost vyjádřit až 16 milionů rozdílných barev. Tato barevná hloubka se označuje jako 24 bitová. Možné jsou i jiné hodnoty např. 32, 48 bitu na pixel. Zde jsou použity pouze barevné obrazy o 24 bitové hloubce, což znamená hodnoty v rozsahu 0 – 255 pro každý kanál RGB.



Obr. 12 Barevné obrazy

Barevná hloubka	Počet bitů komponenty				
počet bitů	R	G	B	A	barev
8bit	3	3	2		256
16bit	5	6	5		65536
18bit	6	6	6		262144
24bit	8	8	8		16 777 216
32bit	8	8	8	8	4 294 967 296
48bit	12	12	12	12	281 474 976 710 656

Tab. 2 Barevné hloubky (poz. A – alfa kanál pro průhlednost v RGBA)

Reprezentace barevného obrazu

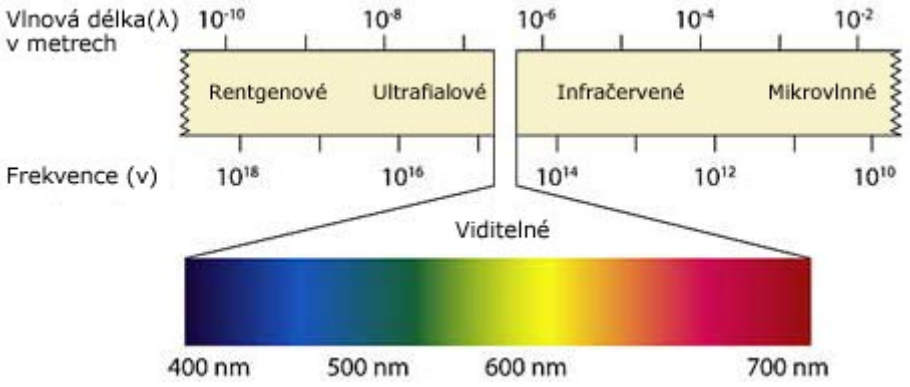
Barevný obraz v této práci je reprezentován trojrozměrnou maticí, zde ji označme IMG

$$IMG = \begin{bmatrix} f(1,1,1), f(1,1,2), f(1,1,3) & \dots & f(1,N,1), f(1,N,2), f(1,N,3) \\ \dots & & \dots \\ f(M,1,1), f(M,1,2), f(M,1,3) & \dots & f(M,N,1), f(M,N,2), f(M,N,3) \end{bmatrix} \quad (3.3)$$

Kde x ∈ <1, …, M> a y ∈ <1, …, N> a z (index RGB kanálu) ∈ <1,3>. Hodnota jasu *f* pro každý kanál je v rozsahu <0,255>.

3.2 Barevné modely

Barva skutečných předmětů je určena vlnovou délkou světla odraženého od jejich povrchu. Viditelné světlo má frekvenci cca 10 MHz. Vlnová délka sahá od 380 do 720 nm. V tomto rozsahu je průměrný člověk schopen rozlišit až 400 000 různých barev. Takovýto popis barev je ale pro počítačové zpracování nevhodný. Kvůli tomu byly zavedeny tzv. barevné modely popisující barvu poměrně jednoduchým způsobem.



Obr. 13 Světelné spektrum

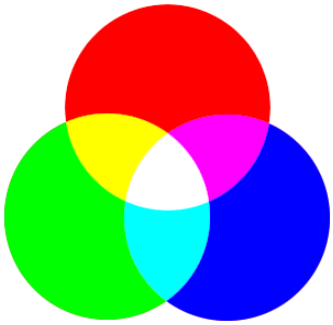
Jakýmsi původním modelem, zohledňujícím různou citlivost lidského oka na různé vlnové délky světla, je model stanovený již v roce 1931 Mezinárodní komisí pro osvětlení CIE *(Commision*

Internationale de l'Éclairage – fr.) a označený jako CIE 1931, někdy také Yxy [5]. Další barevné modely jsou jeho podmnožinou a tento model je používán více méně jen jako teoretický základ, z nějž se další barevné systémy odvozují (např. LAB, Luv, CIE 1976 USC ...).

V počítačovém zpracování se používá především dvou základních modelů:

- aditivního – RGB
- subtraktivního – CMY(K).

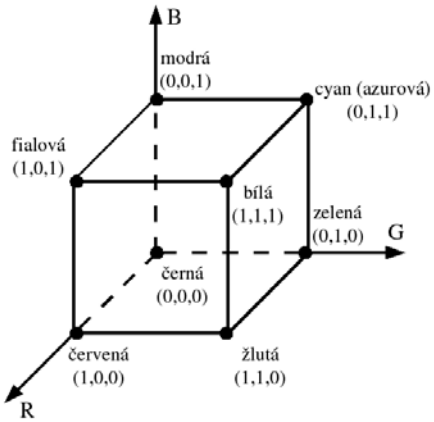
3.2.1 Model RGB



Obr. 14 aditivní model RGB

Model RGB se nejčastěji používá pro snímání barevného obrazu a zobrazování na monitoru. Byl standardizován již v roce 1931 komisí CIE [10]. Barevný model dle tohoto standartu je složen aditivně ze tří barev: červené, zelené a modré. Jednotlivým položkám byly přiřazeny vlnové délky: červená = 700 nm, zelená = 546,1 nm a modrá = 435,8 nm. Z technického hlediska je každá barva reprezentovaná určitým počtem bitů (běžně 8bit/barvu). Složením jednotlivých barevných složek RGB v patřičném poměru získáme výslednou barvu. Při 3 x 8 bitech tak lze vyjádřit asi 16,7 milionů barev (2^{24}).

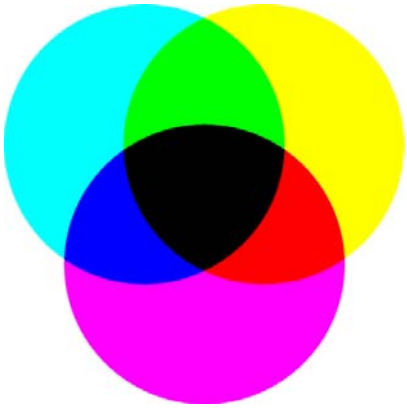
Barevný prostor RGB se označuje jako aditivní skládání barev (čím vyšší hodnoty, tím světlejší).



Obr. 15 RGB prostor

3.2.2 Model CMY a CMYK

Barevný model CMYK (modrozelená, purpurová, žlutá, černá) se používá primárně pro tisk. Označuje se jako subtraktivní.



Obr. 16 CMYK

V ideálním případě by byly postačující pouze první tři barvy (model CMY), jejichž subtraktivním složením dohromady by měla vzniknout černá barva. Ve skutečnosti však při použití běžných barviv vzniká barva tmavě šedivá, a zároveň je, na rozdíl od ostatních barev, černá výrazně levnější, a proto většina tiskových technik používá ještě čtvrtou, černou barvu.

Převod mezi RGB a CMY

U subtraktivního modelu (CMY) odpovídá počátek souřadnic bílé a vrchol [1, 1, 1] černé. Barvy C, M a Y leží na krychli v protilehlých rozích oproti RGB. CMY se označují jako doplňkové k RGB. Překrýváním těchto tří barev nevznikne dokonalá černá (proto zaveden CMYK).

Převod mezi RGB a CMY lze tedy provézt pomocí vztahu:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.4}$$

Předpokládáme zde, že jasové úrovně jsou normalizovány v intervalu <0,1>.

4 Jasové transformace

Jelikož ne vždy se podaří získat obraz takové kvality jakou bychom požadovali, máme k dispozici velkou škálu způsobů a metod jak si zlepšit výslednou kvalitu obrazu. Tyto metodiky se dají obecně nazvat transformace jasu. Změny jasu docílíme transformací jasové úrovně na úroveň jinou. Operace lze provádět buď v doméně *prostorové* nebo *frekvenční.*

V prostorové doméně se pracuje s obrazovou funkcí *f*(*x*,*y*), kde oborem hodnot jsou jednotlivé úrovně jasu a definičním oborem je rozměr daného obrazu. V doméně frekvenční se pracuje s obrazem transformovaným pomocí Fourierovy transformace.

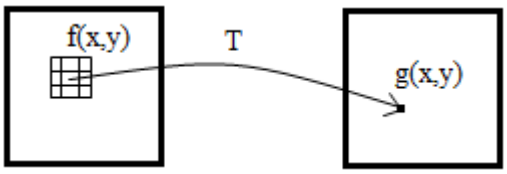
V této práci jsou aplikovány jasové transformace v oblasti prostorové, a nadále se budu zabývat pouze touto skupinou technik.

Obecně lze jasovou transformaci vyjádřit jako

$$g(x,y) = T\left[f(x,y)\right] \tag{4.1}$$

f(*x*,*y*) - vstupní obraz
g(*x*,*y*) - výstupní obraz
T- transformace

Nejjednodušší transformace neuvažují okolí daného bodu. Jde o jednoduché přiřazení bodu (x,y) o intenzitě f jiné úrovni jasu g. Některé transformace zpracovávají informaci z okolí daného bodu a tudíž výsledek je na tomto okolí závislý. Poslední skupina transformací je závislá na globálních vlastnostech obrazu např. minimum/maximum nebo na hodnotách z histogramu. Tato kapitola se bude zabývat transformacemi jasovými a transformacemi založenými na práci s histogramem.



Obr. 17 Transformace závislá na okolí

Použité jasové transformace

- Negativ
- Gama korekce
- Metoda vyrovnání histogramu
- Vyhledávací tabulka LUT
- Přímá specifikace histogramu

4.1 Negativ

Negativ je jedna z nejjednodušších jasových transformací. Nevyužívá hodnot z předchozí analýzy obrazu. Intenzita každého pixelu (x,y) je nahrazena hodnotou jinou.

Negativ monochromatického obrazu

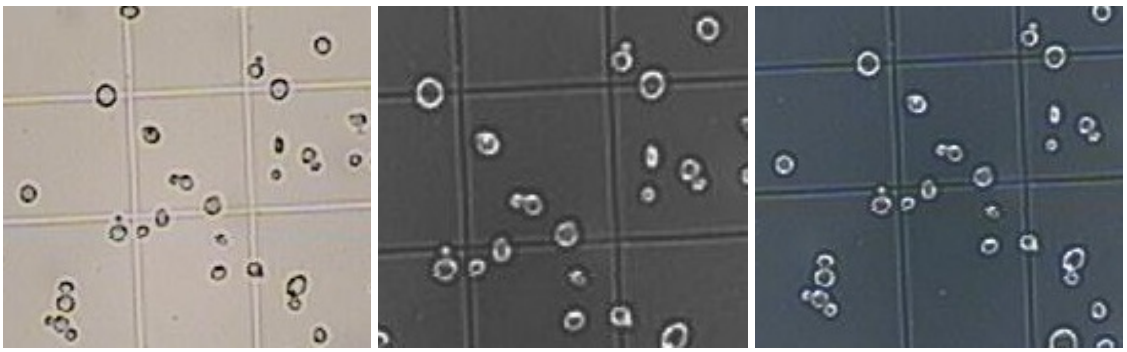
$$g(x,y) = 255 - f(x,y) \tag{4.2}$$

Přepis: $g = 255 - f$

Negativ barevného obrazu

$$g(x,y,z) = 255 - f(x,y,z) \tag{4.3}$$

Přepis: $g = 255 - f$



Obr.18 Zleva: Originál, negativ monochromatického obrazu, negativ barevného obrazu.

4.1.1 Implementace

```
funkce : Negativ
vstup : barevny nebo monochromatický obraz (PicArray)
výstup : negativ obrazu (NegArray)

NegArray = Negativ (PicArray)
{
 for x=1 to width(PicArray)
      for y=1 to height(PicArray)
            z =  length(PicArray)
            if z =2
                  NegArray(x,y,1) = 255 - PicArray(x,y,1);
                  NegArray(x,y,2) = 255 - PicArray(x,y,2);
                  NegArray(x,y,3) = 255 - PicArray(x,y,3);
            elseif z=1
                  NegArray(x,y) = 255 - PicArray(x,y);

            end
      end
end
}
```

4.2 Gama korekce

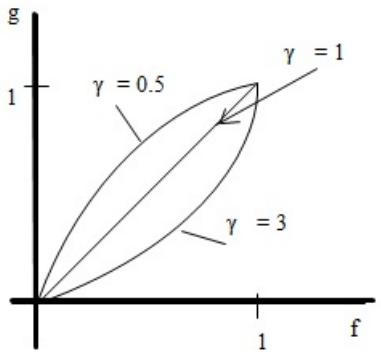
Tato nelineární operace se využívá pro opravu jasu, popřípadě špatné expozice, kdy je požadovaný objekt ukryt v příliš tmavé oblasti nebo naopak v příliš světlé oblasti.

Základem je křivka gama korekce a výpočtový vztah:

Výpočtový vztah (v nejjednodušším případě)

$$g(x,y) = f(x,y)^{\gamma} \tag{4.4}$$

Křivka gama korekce



Obr.19 Křivka gama korekce

Rovnice gama křivky

$$y = y_{\min} + (y_{\max} - y_{\min})\left(\frac{x}{x_{\max}}\right)^{\gamma+1} \tag{4.5}$$

4.2.1 Implementace

```
funkce : GammaCorrection
vstup : monochromatický obraz(PicArray), gama
výstup : upravený monochromatický obraz(GammaArray)

GammaArray = GammaCorrection (PicArray, gama)
{
 for x=1 to width(PicArray)
      for y=1 to height(PicArray)

            GammaArray (x,y) = pow(PicArray(x,y),gama);

      end
end
}
```

4.3 Vyhledávací tabulka LUT

Zde vycházíme z faktu, že při operaci kdy přiřazujeme hodnotě jasu f_k hodnotu jasu g a tu následně ukládáme na patřičné místo ve výsledném obraze, není nutné vypočítávat transformaci g_k=T(f_k) znovu pro každý bod v obraze. Transformace se vypočte jednou a její hodnoty se uloží do LUT tabulky. Jedná se o jednorozměrné pole obsahující vypočtené hodnoty jasu g_k pro jednotlivé úrovně jasu f_k = 0, 1, … , 255, což lze zapsat následovně: LUT(f_k) = g_k.

Do výsledného obrazu pak přiřazujeme LUT(f_k) = g_k , kde f_k je hodnota jasu v originálním obraze a $f(x,y)$ je index v LUT tabulce.

Alternativní zápis

$$g(x,y) = LUT[f(x,y)] \tag{4.6}$$

4.3.1 Implementace LUT

```
funkce : LUTHIsto
vstup : monochromatický obraz(PicArray)
výstup : upravený monochromatický obraz(GammaArray)

GammaArray = LUTHIsto (PicArray)
{
h = histogram(PicArray); viz. 3.3.2

celkemPixelu = width(PicArray)* height(PicArray);
for i = 0 to 255
      if i== 0
                  LUT[i] = h/celkemPixelu
            else
                  LUT[i] = LUT[i-1] + h/celkemPixelu;
            end
end

for x=1 to width(PicArray)
      for y=1 to height(PicArray)

          zmena = LUT(PicArray(x,y)*255)/ PicArray(x,y);
                intenzita = PicArray(x,y) * změna;

                vystupArray(x,y) = intenzita;

      end
end
}
```

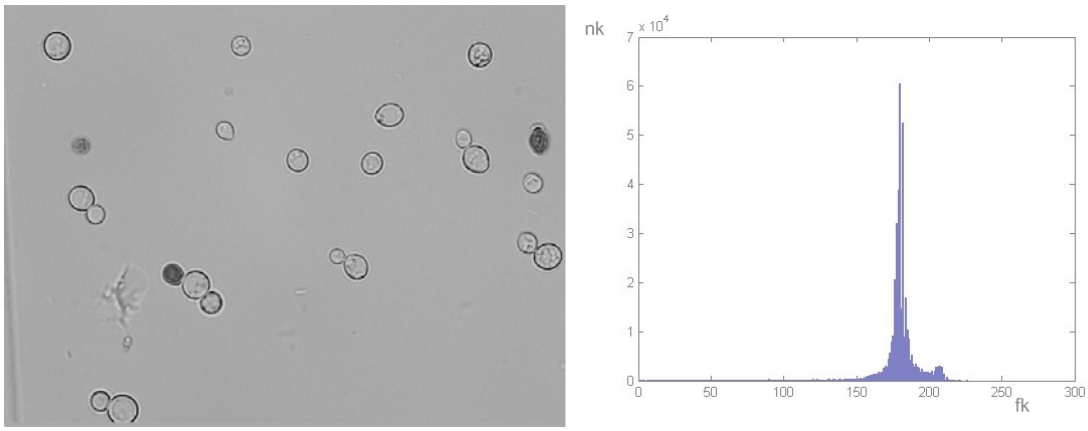
4.3.2 Histogram

Histogram představuje rozložení jasových úrovní v obraze. U digitálního obrazu se jedná o diskrétní funkci $h(f_k) = n_k$, kde f_k je k-tá úroveň jasu a n_k je počet pixelů s touto jasovou hodnotou. Pro monochromatický obraz vystačíme s jedním histogramem, ale pro obraz barevný potřebujeme histogram pro každou RGB složku obrazu. Histogram se zobrazuje jako graf, kde horizontální osa představuje hodnoty jasu f_k a vertikální osa počet pixelů daného jasu.

Programově je histogram jednorozměrné pole např. h, kde každá položka odpovídá jasové hodnotě. V našem případě tedy máme pole o délce 255. Následně projdeme každý bod obrazu, zjistíme jeho jasovou hodnotu a implementujeme příslušný index v poli tedy: $h(fk) = h(fk) + 1$.

Implementace (pro monochromatický obraz)

```
h = array(255);
for x=1 to width(PicArray)
      for y=1 to height(PicArray)
           h(PicArray(x,y)) +=1;
      end
end
```

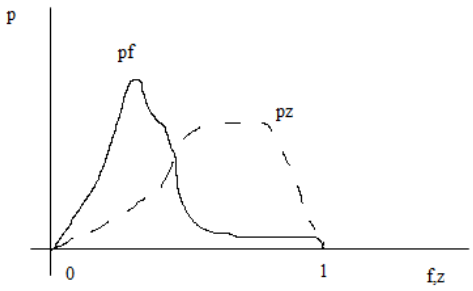


Obr.20 Obrázek a korespondující histogram

4.4 Přímá specifikace histogramu

Tato technika nám umožní specifikovat histogram výstupního obrazu tak, aby určité jasové úrovně byly zastoupeny s vyšší četností a jiné s nižší. Je to pro případ kdy očekáváme, že v dané jasové úrovni je pro nás podstatná informace.

Princip je následující: hodnoty jasu v originálním obraze označíme f a ve výstupním z. Funkce hustoty pravděpodobnosti jasu $p_z(z)$ pro výstupní a $p_f(f)$ pro vstupní obraz. Hledáme tedy vztah mezi f a z.



Obr.21 Specifikace histogramu

Vztah mezi původním a transformovaným obrazem můžeme zapsat jako

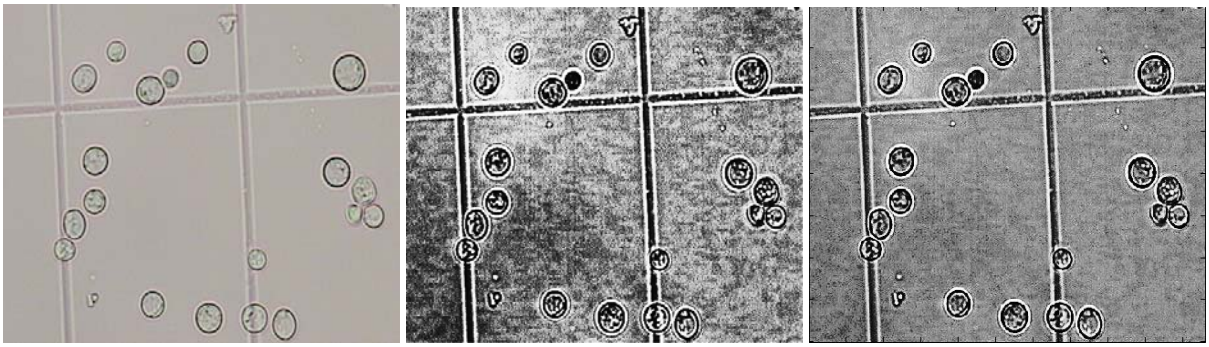
$$g = \int_0^f pf(w)dw \tag{4.7}$$

respektive pro výstupní obraz

$$v = \int_0^z p_z(w)dw \tag{4.7}$$

Transformované hodnoty by jsme chtěli v = g. Hledáme tedy *f* -> *g* <- *z*.

Řešení této inverzní transformační funkce je v diskrétním případě možné. Hledání vztahu spočívá ve vytvoření dvou funkcí, které dávají uniformní hustotu pravděpodobnosti, zapisování mezivýsledků a porovnávání transformovaných hodnot.



Obr.22 Vstupní obraz (vlevo), vyrovnání pomocí LUT (uprostřed), vyrovnání pomocí specifikace (vpravo)

5 Filtrace

Metody filtrace zahrnují vyhlazování obrazu, ostření obrazu a hledání hran nespojitostí. Jedná se o procesy, které zpracovávají informace z okolí daného bodu v obraze. Velice často jsou založeny na principu konvoluce nebo korelace. Jsou to lineární operace a tedy platí: $T(af + bg) = aT(f) + bT(g)$.

Kromě konvoluce a korelace se při filtraci používají i nelineární metody např. mediánová filtrace. Dále se zaměřím spíše na metody použité v této práci a to:

- Vyhlazování průměrováním.
- Laplaceův operátor.
- LoG.
- Sobelův operátor.
- Prewitové operátor.

5.1 Konvoluce

V počítačové grafice se o konvoluci mluví jako o diskrétní dvourozměrné konvoluci. Konvoluce zde využívá masku m obsahující číselné hodnoty. Maska se někdy nazývá konvoluční jádro. Maska se postupně posunuje po obraze a v každém kroku posunutí se provede součin každého koeficientu v masce s hodnotou v obraze pod maskou a vypočítá se suma těchto hodnot. Tato suma se uloží do nového obrazu na danou pozici. Pozice je často volena jako středový bod masky. Při korelaci se postupuje stejně s tím rozdílem, že maska je vůči konvoluční masce otočená o 180°.

Konvoluce pro masku 3x3 lze zapsat:

$$g(x,y) = \sum_{s=-1}^{1}\sum_{t=-1}^{1} m(s,t) f(x-s, y-t) \tag{5.1}$$

Při aplikací masky na obraz je nutné ošetřit hranice obrazu. Toho je možné dosáhnout několika způsoby: zvětšením obrazu o polovinu masky a doplnění nulami, pohyb masky je ohraničen jen tak, aby nepřesahoval přes hranice obrazu, nebo upravíme rozsah masky tak, aby nepřesáhla hranice. V našem případě je upraven pohyb masky.

5.1.1 Implementace aplikace masky

```
funkce : maska3x3
vstup : souřadnice x,y, monochromatický obraz(PicArray), maska
        3x3
výstup : výsledná hodnota na pozici x,y po aplikaci masky

suma = maska3x3 (x,y,PicArray,maska)
{
 for i= -1 to 1
      for j= -1 to 1
            suma +=  PicArray(x+i,y+j) * maska(i+2,j+2);
      end
end
}
```

5.2 Vyhlazování průměrováním

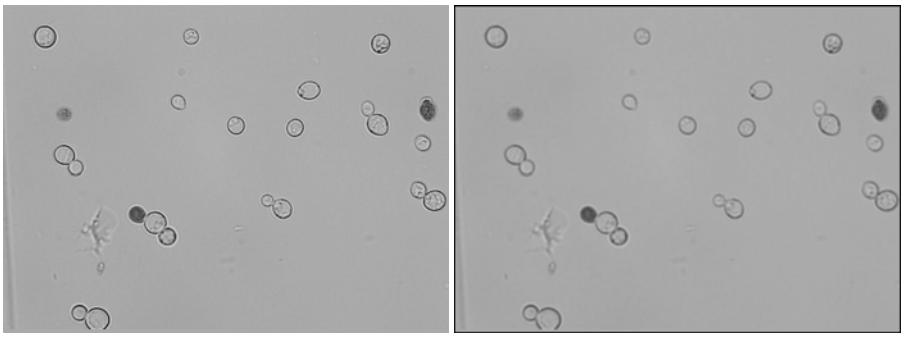
Vyhlazování obrazu použijeme v případech, kdy je obraz znehodnocen jevy jako např. šum, zrnění., či jiné artefakty. Vždy je ale nutné zvolit vhodnou metodu vyhlazování. Výsledky se mohou lišit. V této práci je použito vyhlazování průměrováním.

Jedná se o jedno z nejjednodušších vyhlazování obrazu pomocí konvoluce. Principielně jde o nanesení masky, která zajistí, že výsledkem konvoluce je průměr hodnot jasu v okolí středu masky. Pro masku o velikosti 3x3 vyjde konvoluce ve tvaru:

$$g(x,y)=\frac{1}{9}\sum_{s=-1}^{1}\sum_{t=1}^{1}f(x-s,y-t) \tag{5.2}$$

tedy maska:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



Obr.23 Průměrové vyhlazování. Vlevo originální obrázek.

5.2.1 Implementace

Jedná se o pouhé použití algoritmu z 4.1.1 Implementace aplikace masky a použití dané masky.

5.3 Laplaceův operátor

Jedná se o problematiku detekce hran a nespojitostí. Hrany v obraze odpovídají prudkým změnám hodnot jasu. Takže hledáme místa v obraze kde se jas významně mění. Toho docilujeme první nebo druhou derivací intenzity jasu. Kritériem je velikost první či druhé derivace jasu nebo také změna znaménka derivace.

Laplaceův operátor využívá druhé derivace jasu. Ta představuje rychlost změny hodnoty jasu a to zejména na strmých a izolovaných hranách.

V diskrétních obrazech lze druhou derivaci spočítat jako rozdíl hodnot jasu ležících vedle sebe [5].

Pro osu x:

$$\frac{\partial^2 f(x,y)}{\partial x^2}\approx\left[f(x+1,y)-f(x,y)\right]-\left[f(x,y)-f(x-1,y)\right] \tag{5.3}$$

Pro osu y:

$$\frac{\partial^2 f(x,y)}{\partial y^2} \approx [f(x,y+1)-f(x,y)]-[f(x,y)-f(x,y-1)] \tag{5.4}$$

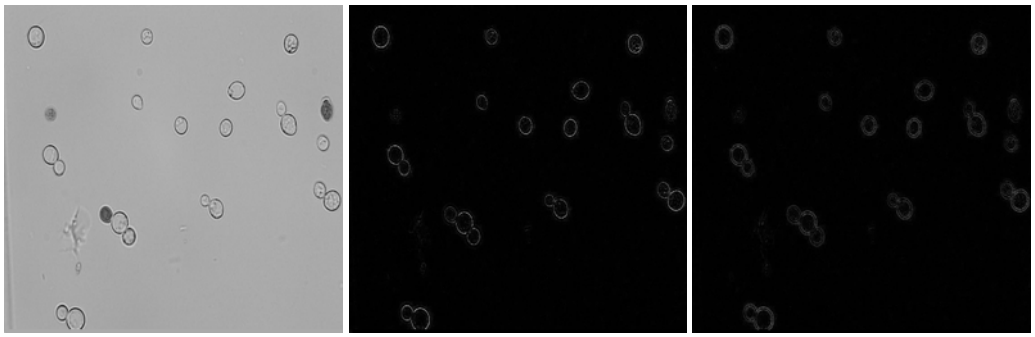
Též můžeme použít Laplaceův operátor

$$\nabla^2 g(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \tag{5.5}$$

Pro Laplaceův operátor se nejčastěji používá 5x5 masky a to buď v pozitivní nebo negativní variantě:

0	1	0
1	-4	1
0	1	0

0	-1	0
-1	4	-1
0	-1	0



Obr.24 Použití Laplaceho operátoru. Originál (vlevo), pozitivní (uprostřed) a negativní varianta masky (vpravo).

5.3.1 Implementace

Opět se jedná o použití algoritmu z 4.1.1 Implementace aplikace masky a použití dané masky.

```
funkce : laplacexy
vstup :  monochromatický obraz(PicArray)
výstup : upravený obraz(PicOut)

PicOut = laplacexy (PicArray)
{
maska = [[0,1,0],[1,-4,1],[0,1,0]];
for x=2 to width(PicArray)-1
      for y=2 to height(PicArray)-1
          PicOut (x,y) maska3x3 (x,y,PicArray,maska);
      end
end
}
```

5.4 Laplacian of Gaussian

Laplaceův operátor má nevýhodu v citlivosti na šum. To může vést až k detekci dvojitých hran, dále není schopen určit směr hrany. Tyto nevýhody lze zmírnit následným použitím jiné

techniky. Jako vhodná technika se osvědčil Gaussův operátor využívaný např. pro vyhlazování obrazu. Použitím Gaussova operátoru by došlo k rozmazání obrazu, které by omezilo následnou citlivost Laplaceova operátoru na šum. Operace není nutné provádět zvlášť, lze je spojit do jedné. Tato dvojrozměrná funkce se nazývá: Laplacian of Gaussian(LoG) [5].

Výsledný tvar LoG:

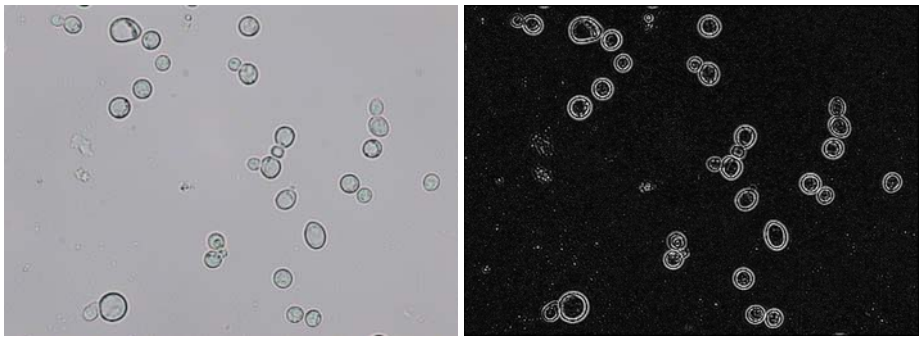
$$h(x,y)=\frac{r^2-\sigma^2}{2\pi\sigma^6}e^{-\frac{r^2}{2\sigma^2}} \tag{5.6}$$

kde $r=x^2+y^2$.

LoG nedetekuje ostré rohy a má tendenci k vyhlazování ostrých hran a spojování do uzavřených křivek. Výsledky jsou ovlivněny hodnotou šířky Gaussova filtru σ. Pro σ=0,4

Máme masku 5 x 5 ve tvaru:

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0



Obr.25 Aplikace LoG: originál (vlevo), ohraněný obraz

5.4.1 Implementace

Nastavíme masku dle potřeby a použijeme ji ve funkci **maska3x3** viz. 4.1.1 Implementace aplikace masky.

5.5 Sobelův a Prewittové operátor

Tyto operátory využívají první derivaci obrazu ve směrech x nebo y. Velikost změny funkce určuje gradient $\nabla f(x, y) = [Gx,Gy]$ a směr gradientu dává informaci o směru největšího růstu obrazové funkce. Body s velkým gradientem se nazývají hrany. Směr určuje úhel:

$$\varphi(x,y)=\arctan\frac{Gx}{Gy} \tag{5.7}$$

Odhad derivací Gx a Gy lze v diskrétním případě spočítat jako rozdíl dvou sousedních hodnot:

$$Gx \approx f(x+1,y)-f(x,y) \tag{5.8}$$

respektive

$$Gy \approx f(x, y+1) - f(x, y) \tag{5.9}$$

Vlastní operátory pak bývají nejčastěji ve velikosti 3x3 a ve tvaru:

- Sobelův operator, směr x respektive y

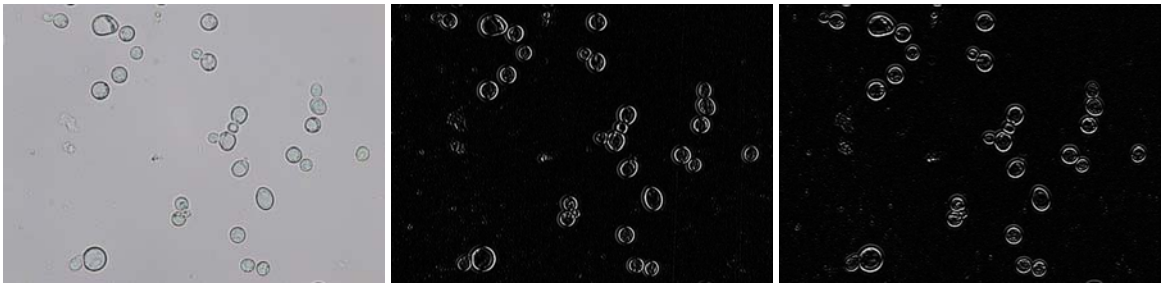
-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

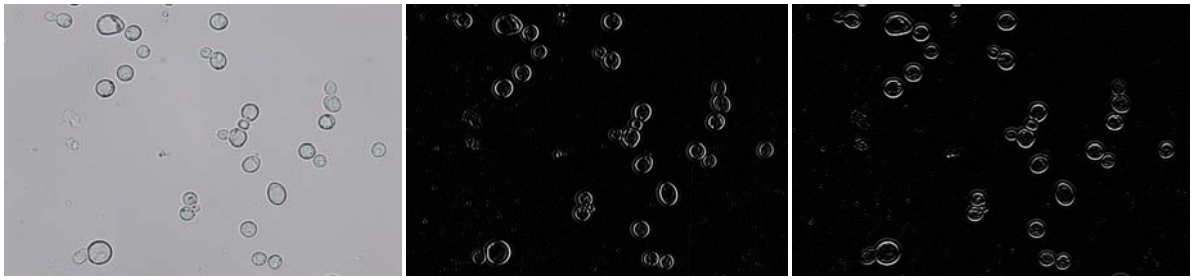
- Prewittové operator, směr x respektive y

-1	0	-1
-1	0	-1
-1	0	-1

-1	-1	-1
0	0	0
1	1	1



Obr.26 Sobelův operátor: originál (vlevo), směr x a směr y.



Obr.27 Prewittové operátor. Zleva: originál, směr x a směr y.

5.5.1 Implementace

```
funkce : EdgeDetect
vstup :  monochromatický obraz(PicArray), identifikátor
         operátoru(operator)
výstup : upravený obraz(PicOut)

PicOut = EdgeDetect (PicArray, operator)
{
If operator == „sobel"
     maska = [[-1,0,1],[-2,0,2],[ -1,0,1]];
     maska2 = [[-1,-2,-1],[0,0,0],[ -1,-2,-1]];
elseif operator == „prewitt"
     maska = [[-1,0,-1],[ -1,0,-1],[ -1,0,-1]];
     maska2 = [[-1,-1,-1],[0,0,0],[ -1,-1,-1]];
end

for x=2 to width(PicArray)-1
      for y=2 to height(PicArray)-1
                    PicOutX (x,y) maska3x3
(x,y,PicArray,maska);
                    PicOutY (x,y) maska3x3
(x,y,PicArray,maska2);
             end
end


for x=1 to width(PicArray)
      for y=1 to height(PicArray)
                 if PicOutY (x,y)  > 0
                    PicOutX (x,y)= PicOutY (x,y);
                 end
             end
end

}
```

6. Segmentace, detekce čar, fázová korelace

Pod segmentací si můžeme představit proces, kdy rozdělíme obsah obrazu na části pro nás zajímavé a zbytek, který je pro nás zpravidla nezajímavý. Je to velice složitý proces vzhledem ke všem možným zachyceným událostem. Algoritmy segmentace můžeme rozdělit na algoritmy hledající podobnosti nebo hledající nespojitosti [5].

Algoritmy hledající podobnosti:

- Region growing
- Region splitting
- Merging

Algoritmy hledající nespojitosti:

- Hledání hran (4.5 Sobelův a Prewittové operátor)
- Detekce čar - Houghova transformace

Statistické metody

- Prahování
- Clustering

Hybridní metody

- Watershed transform
- Neuronové sítě

V této práci byly použity jen některé techniky, proto se budeme zabývat jen jimi.

6.1 Prahování

Vstupem prahování je monochromatický obraz a výstupem obraz binární. Důležitou hodnotou je zde tzv. práh. Je to určitá mez, která určuje kdy je ve výsledném obraze přiřazena hodnota 1 a kdy 0. Pro hodnoty větší než práh odpovídá 1, pro hodnoty menší 0. Prahování můžeme rozdělit na globální, kdy máme jednu hodnotu prahu a ta platí pro celý obraz, nebo lokální, kdy je hodnota prahu měněna v závislosti na části obrazu. To lze zapsat následovně:

$$f(x,y) \geq T \begin{cases} g(x,y)=1 \\ g(x,y)=0 \end{cases} \tag{6.1}$$

$f(x,y)$ - vstupní obraz
$g(x,y)$ - výstupní obraz
T- prahová hodnota

Nejzásadnější úkolem je nalezení vhodného prahu. Rozdílné prahové hodnoty mohou vést k diametrálně odlišným výsledkům. Možné metody stanovení prahu:

- Manuálně uživatelem.
- Metoda automatického nalezení prahu.
- Nalezení optimálního prahu.
- Lokální prahování.
- Prahování s distribucí chyby.

Implementace

Použití prahování je triviální záležitost, v podstatě stačí otestovat každý bod obrazu v jakém je vztahu k prahové hodnotě. Např. následovně:

```
for x=1 to width(PicArray)
      for y=1 to height(PicArray)
                  if PicArray (x,y)  >= T
                     PicOutX (x,y)= 1;
                   else
                     PicOutX (x,y)= 0;
                  end
              end
end
```

6.1.1 Metoda automatického nalezení prahu

Aplikovatelné u obrazů, které mají histogram uspořádaný tak, že má dva výrazné vrcholy. Jeden vrchol pak odpovídá jasovým hodnotám objektu zájmu a vrchol druhý hodnotám pozadí.

Metoda funguje na postupném hledání prahu T jako průměru středních hodnot oblastí odpovídajících popředí a pozadí. Algoritmus je následující:

1. Zvolíme T počáteční hodnotu prahu (náhodně, střed mezi maximem a minimem jasu)
2. Vytvoříme dvě množiny objektů. Jedna např. A pro hodnoty jasu menší než T. Druhá množina hodnot B je pro body, které mají hodnotu jasu menší než T.
3. Určíme průměrnou hodnotu jasu v každé z množin, označíme m_A a m_B.
4. Určíme nový práh dle $T_{new} = (m_A + m_B)/2$.
5. Pokračujeme od kroku 2, pokud není splněna podmínka ukončení. Ta může být např. $|T - T_{new}| < 0.5$ nebo $T - T_{new} = 0$.

6.1.2 Lokální prahování

Tento postup je výhodné použít v případě nerovnoměrného rozložení jasu, kdy by globální práh nedával dobré výsledky. Princip spočívá v postupném zkoumání okolí každého bodu a stanovení prahovací hodnoty T dle získaných informací z okolí. V našem případě je okolí maska, která se pohybuje po obraze.

Pro určení hodnoty prahu v masce je možné použít už jakýkoliv algoritmus pro stanovení prahu.

6.2 Detekce čar - Houghova transformace

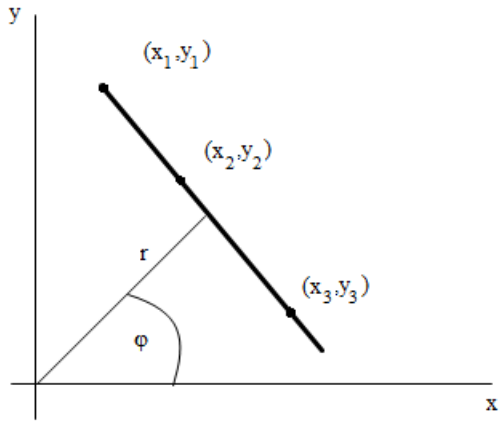
Houghova transformace je metoda pro nalezení parametrického popisu objektu v obraze [7]. Při implementaci je třeba znát analytický popis tvaru hledaného objektu. Proto je tato metoda používána pro detekci jednoduchých objektů v obraze jakou jsou přímky, kružnice, elipsy atd.

Obrovskou výhodou této metody je její robustnost, kdy segmentace není příliš citlivá na porušená data nebo šum. Jde o transformaci z Kartézského souřadnicového systému do polárního.

Úkolem je tedy nalezení přímek v rovině. Pro přímku použijeme:

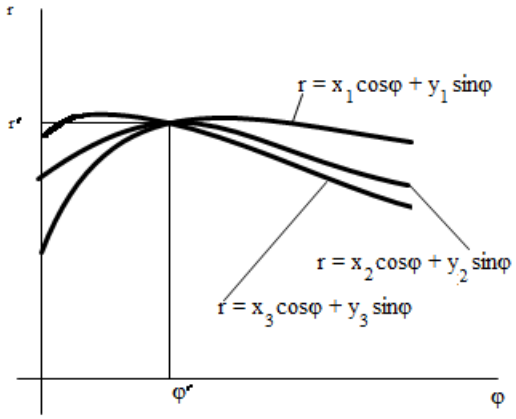
$$r = x\cos\varphi + y\sin\varphi$$

kde r je kolmá vzdálenost do počátku a φ úhel mezi normálou a osou x.



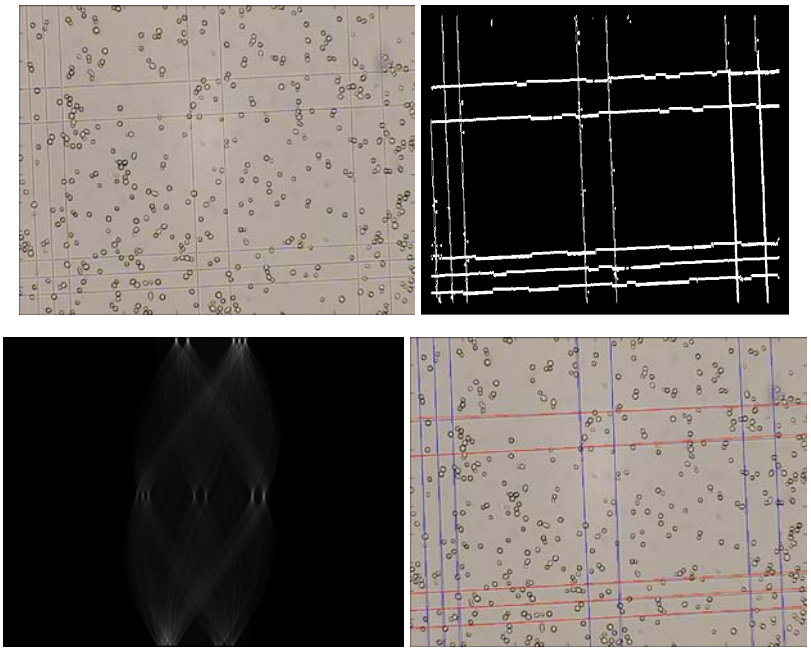
Obr.28 Parametrický model přímky

Vstupem jsou souřadnice bodu (např. x_1,y_1) a hledáme r a φ. Když naše vstupní hodnoty vložíme do parametrického modelu přímky, tak množina všech možných řešení (r, φ) vytvoří v Houghově prostoru spojitou křivku. Jestliže budeme dále vkládat body na naší hledané přímce, docílíme protnutí křivek v Houghově prostoru na jednom místě. Toto místo respektive souřadnice jsou námi hledané parametry přímky r a φ.



Obr.29 Houghův prostor, protnutí křivek je hledaný bod

Houghova transformace je implementována tak, že v prvním kroku je Houghův prostor diskretizován. Každý element tohoto prostoru, který nazýváme akumulační buňka je vynulován. Každý bod (x,y) je transformován do diskretizované křivky (r, φ), přičemž hodnota akumulačních buněk podél této křivky je inkrementována. Souřadnice maxima v akumulační rovině (r, φ) jsou parametry hledané přímky v obraze.



Obr.30 Aplikace Hougovy transformace. Originální obraz (vlevo nahoře), vstupní obraz do algoritmu (vpravo nahoře) ohraněn a filtrován. Hugův prostor (vlevo dole). Výsledný obrázek (vpravo dole).

6.2.1 Implementace

Tuto implementaci můžeme rozdělit do dvou kroků. Nejdříve vytvoříme Hougovu akumulační matici (**HoughLineDetect**) a poté najdeme lokální maxima a pomocí nich vykreslíme přímky (**Vykresli**).

```
funkce : HoughLineDetect
vstup :  binární obraz(PicArray)
výstup : Hougova akumulační matice(houghAcumulator)

houghAcumulator = HoughLineDetect(PicArray)
{
temp = max(width(PicArray), height(PicArray));
houghH=temp*sqrt(2);
houghW=180;
houghAcumulator = 0;
step = 3.1416 / 180;
```

```
for x=1 to width(PicArray)
   for y=1 to height(PicArray)
      if PicOutY (x,y)  ~= 0
         for k=1 to houghW-1
            temp  = (((x-width(PicArray)/2)*cos(k*3.1416/
            180)) + ((y- height(PicArray)/2)*sin(k*3.1416/
            180)));
            temp = temp + houghH;
            if temp >0 AND temp <= 2* houghH
              houghAcumulator(k,temp) =
               houghAcumulator(k,temp) +1;
            end
          end
      end
   end
end
}
```

```
funkce : Vykresli
vstup :  originální obraz, hougova akumulační matice, citlivost
výstup : originální obraz s vykreslenými přímkami

PicOut = HoughLineDetect(OrigPic, houghAcumulator, citlivost)
{
maxAC = max(houghAcumulator);
hranice = maxAC * citlivost;
temp = max(width(PicArray), height(PicArray));
houghH=temp*sqrt(2);
houghW=180;
okoli=7;
step = 3.1416 / 180;

for x=1 to houghW
   for y=1 to 2* houghH
      if houghAcumulator (x,y)  > hranice
        if lokalniVrchol(houghAcumulator, x,y,okoli)
          tsin = sin(x*step);
          tcos = cos(x*step);
          if( x <= houghW/4 OR x>= 3* houghW/4)
             y1=1;y2= height(PicArray)-1;
             x1=(((y – houghH) – ((y1 - height(PicArray)/2 ) *
             tsin)) / tcos) + width(PicArray)/2;
             x2=(((y – houghH) – ((y2 - height(PicArray)/2 ) *
             tsin)) / tcos) + width(PicArray)/2;
```

```
            else
               x1=1;x2= width (PicArray)-1;
               y1=(((x - houghH) - ((x1 - width (PicArray)/2 ) *
               tcos)) / tsin) + heigth(PicArray)/2;
               x2=(((y - houghH) - ((x2 - width (PicArray)/2 ) *
               tcos)) / tsin) + heigth (PicArray)/2;
            end
            OrigPic = lines(OrigPic,x1,x2,y1,y2);
         end
      end
   end
end
}
```

6.3 Fázová korelace

Fázová korelace patří mezi metody registrace obrazu založené na Fourierově transformaci (6.3.2). Cílem metody je získat velikost posuvu mezi dvěma obrazy, které se částečně překrývají. Myšlenka fázová korelace je založená na tzv. Fourierově posuvném teorému a také na skutečnosti, že dva obrázky s jistým stupněm podobnosti tvoří v jejich křížovém výkonovém spektru souvislý ostrý vrchol právě v místě registrace, narozdíl od šumu, který je rozložený náhodně v nesouvislých vrcholcích [10]. Algoritmus si popíšeme následovně:

1. Spočteme Fourierovu transformaci F(A), F(B) vstupních obrazu A a B:

$$FT\{f1(x,y)\} = F1(\omega_x,\omega_y) \tag{6.2}$$

$$FT\{f2(x,y)\} = F2(\omega_x,\omega_y) \tag{6.3}$$

   kde FT značí Fourierovu transformaci.

2. Provedeme fázovou korelaci F1 a F2

   2.1 Předpokládáme vzájemný posun A a B.

$$f2(x,y) = f1(x+\Delta x, y+\Delta y) \tag{6.4}$$

   2.2 Aplikací Fourierova posuvného teorému získáme:

$$F2(\omega_x,\omega_y) = F1(\omega_x,\omega_y)e^{j(\omega_x\Delta x+\omega_y\Delta y)} \tag{6.5}$$

   2.3 Výsledné křížové výkonové spektrum:

$$e^{j(\omega_x\Delta x+\omega_y\Delta y)} = \frac{F2(\omega_x,\omega_y)F1^*(\omega_x,\omega_y)}{\left|F2(\omega_x,\omega_y)F1(\omega_x,\omega_y)\right|} \tag{6.6}$$

   ,kde * znamená komplexně sdružená čísla.

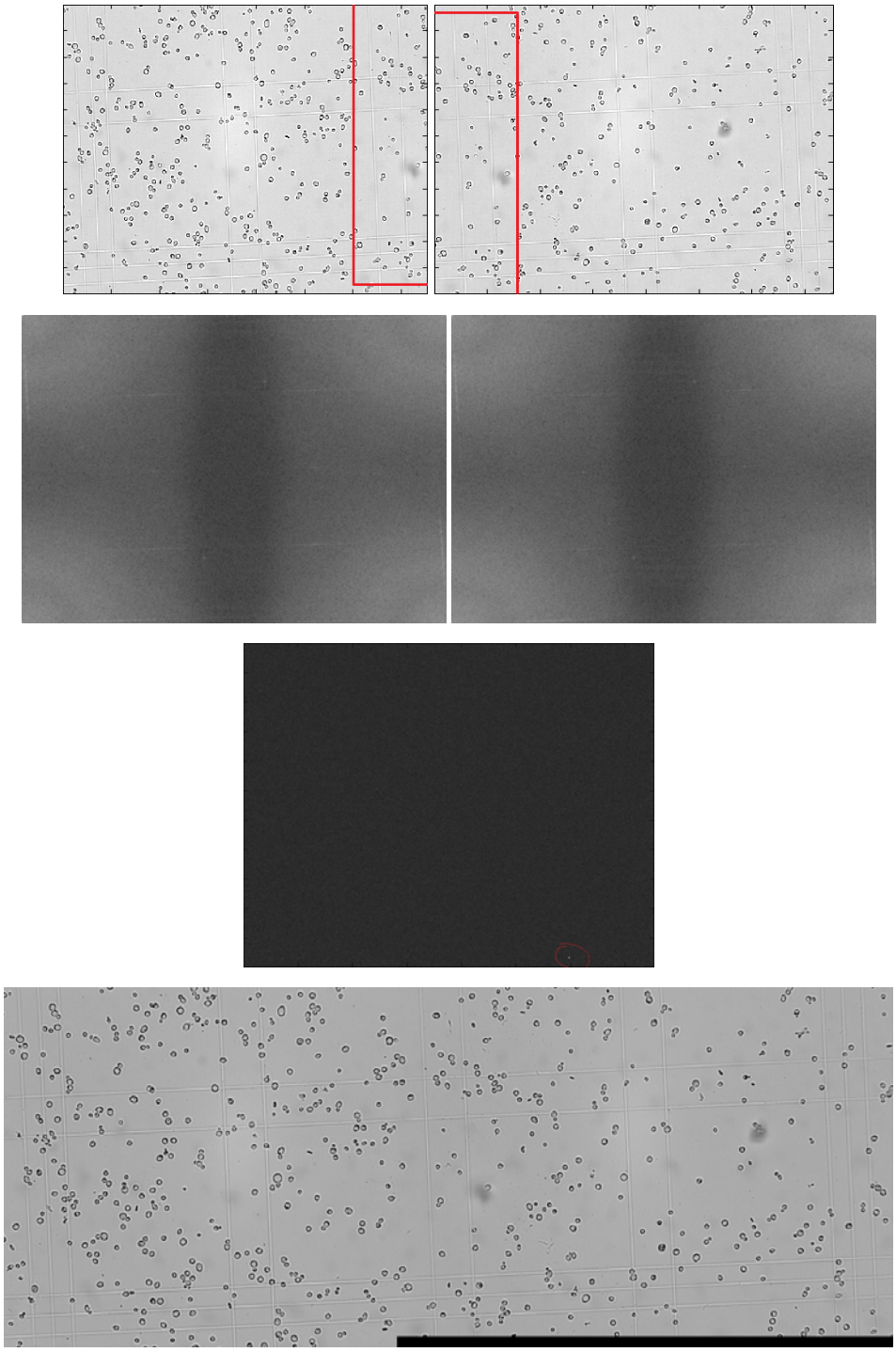
3. Nyní provedeme inverzní Fourierovu transformaci a získáme Dirakovu delta funkci se středem v (△x , △y):

$$FT^{-1}e^{j(\omega_x\Delta x+\omega_y\Delta y)} = FT^{-1}\left(\frac{F2(\omega_x,\omega_y)F1^*(\omega_x,\omega_y)}{\left|F2(\omega_x,\omega_y)F1(\omega_x,\omega_y)\right|}\right) = \delta(\Delta x,\Delta y) \tag{6.7}$$

   kde FT^{-1} značí inverzní FT.

Tento postup předpokládá reálné funkce s neomezeným definičním oborem. Zde ale pracujeme s diskrétními obrazovými funkcemi konečné velikosti. Z tohoto důvodu musíme použít

diskrétní verzi Fourierovy transformace. Dirakova funkce je na konci nahrazena jednotkovým impulsem. Pro nalezení posunu teď stačí jen určení maxima v prostorové oblasti křížového výkonového spektra.



Obr.31 Aplikace fázové korelace. Postupně: vstupní data, fouriérovy obrazy vstupních dat, nalezený vrchol, výsledný obraz.

6.3.1 Implementace

Vzhledem k složitosti implementace Fourierovy transformace je následující ukázka založena na funkcích v prostředí Matlab.

```
funkce : ComputeDelta
vstup :  dva monochromatické obrazy(im1,im2)
výstup : souřadnice společného bodu obrazů(y,x)

[y,x]  = ComputeDelta (im1,im2)
{
F1 = fft2(im1);  %% 2-D diskrétní Fourierova transformace
F2 = fft2(im2);

Fz = F1. * conj(F2); %%  součin F1 a komplexně združených čísel
F2
Fm = abs(F1.*F2);

div = Fz./Fm;%% maticový podíl
delta = ifft2(div); %% inverzní 2-D diskrétní Fourierova
transformace
maxvalue=max(max(delta));  %% nalezení globálního maxima

[y,x] = find(delta==maxvalue); %% nalezení korespondujících x/y
                                  souřadnic globálního maxima
}
```

6.3.2 Fourierova transformace

Fourierova transformace je názornou metodou teorie signálů, která reprezentuje signál pomocí jeho frekvenční charakteristiky. Stejnou metodu lze také využít při zpracování obrazu, který je také možno vyjádřit jako superpozici sinusových funkcí různých fází a amplitud. V případě zpracování obrazu se nepracuje se spojitým, ale s diskrétním obrazem a tím se tedy některé postupy zjednodušují. Pro reprezentaci funkce její frekvenční charakteristikou máme k dispozici dvě reprezentace. V prvním případě, kdy pracujeme s obrazovými prvky, se pohybujeme v časové oblasti, v druhém případě hovoříme o oblasti frekvenční. Přechod mezi těmito oblastmi nám umožňuje právě Fourierova transformace [13].

Definice spojité Fourierovy transformace

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j\pi ft} dt \tag{6.8}$$

Definice inverzní Fourierovy transformace

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{2j\pi ft} dt \tag{6.9}$$

kde:

$x(t)$ je originální funkce

$X(f)$ je Fourierova transformace(obraz k originálu $x(t)$)

6.3.3 Diskrétní Fourierova transformace

Při zpracování signálů pomocí počítačové techniky se pracuje s konečným počtem vzorků a pro spojité funkce to tedy znamená, že lze pracovat pouze s určitým počtem vzorků těchto funkcí. Pracuje se tedy s diskrétními průběhy i ve frekvenční oblasti. Signály v časové i frekvenční oblasti mají konečný počet hodnot N a při výpočtech se považují za periodické. Transformace, která umožňuje přechody mezi časovou oblastí, kde nezávisle proměnnou budeme značit n, a frekvenční oblastí, kde nezávisle proměnnou budeme značit k, je tzv. finitní Fourierova transformace. Nazývá se diskrétní Fourierova transformace (DFT). Analyzuje signál nalezením jeho amplitudy a fázového spektra.

Pro výpočet této transformace byly vypracovány algoritmy, kterým se říká rychlá Fourierova transformace (Fast Fourier Transform - FFT). Tímto se DFT stala výkonným nástrojem pro frekvenční analýzu a číslicovou filtraci. Předpokládáme konečnou posloupnost N konečných komplexních čísel

Definice diskrétní přímé Fourierovy transformace

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} \quad \text{k=0,1,2,…,N-1} \tag{6.10}$$

Definice inverzní Fourierovy transformace

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn} \quad \text{k=0,1,2,…,N-1} \tag{6.11}$$

kde:

$x(k)$ je originální funkce

$X(n)$ je Fourierova transformace(obraz k originálu $x(t)$)

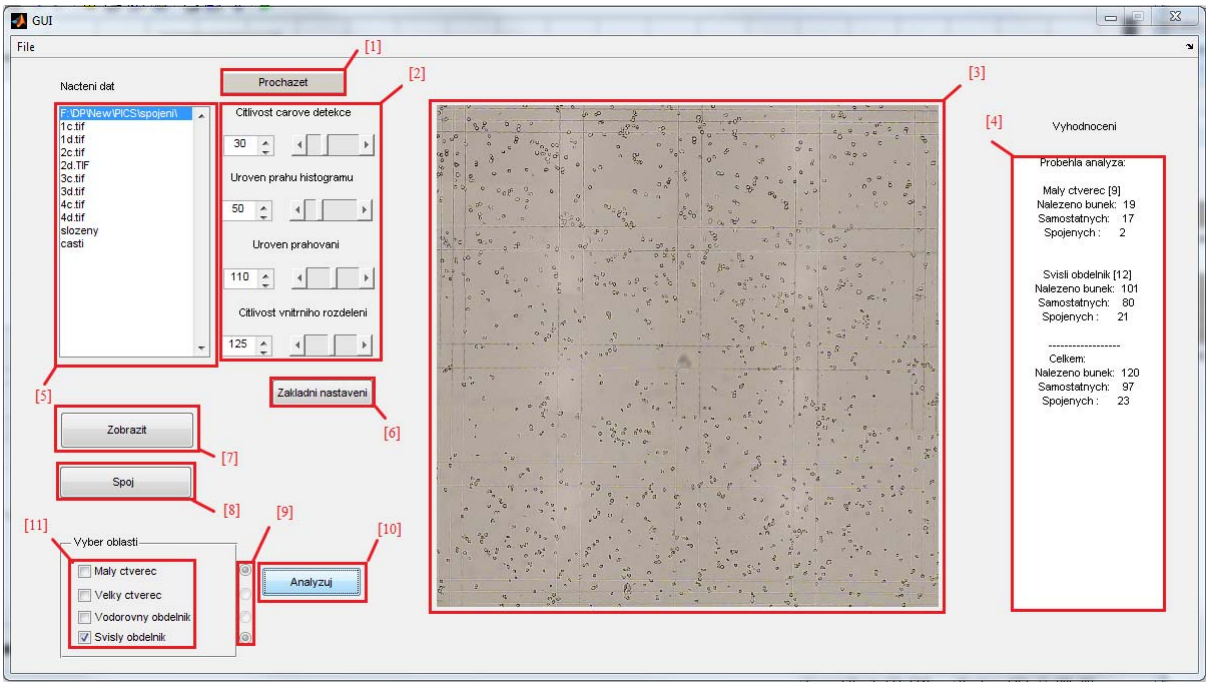
7 Rozbor aplikace

Nedílnou součástí výsledné aplikace je návrh pracovního a uživatelského prostředí, které by usnadňovalo použití a přístup k dříve uvedeným metodám. Vzhledem k použití vývojového prostředí Matlab by volání jednotlivých funkcí nebylo vhodné. Cílem tedy je vytvoření interaktivního prostředí ve kterém by byl uživatel schopen dosáhnout patřičných výsledků bez větší znalosti problematiky obrazové analýzy, prostředí Matlab a použité struktury a posloupnosti běhu funkcí.

Dále si v této kapitole uvedeme stručný seznam základních pomocných a sekundárních funkcí popřípadě skriptů. Popis veškerých metod a jejich rozbor je mimo rozsah této práce.

Pracovní postup GUI:

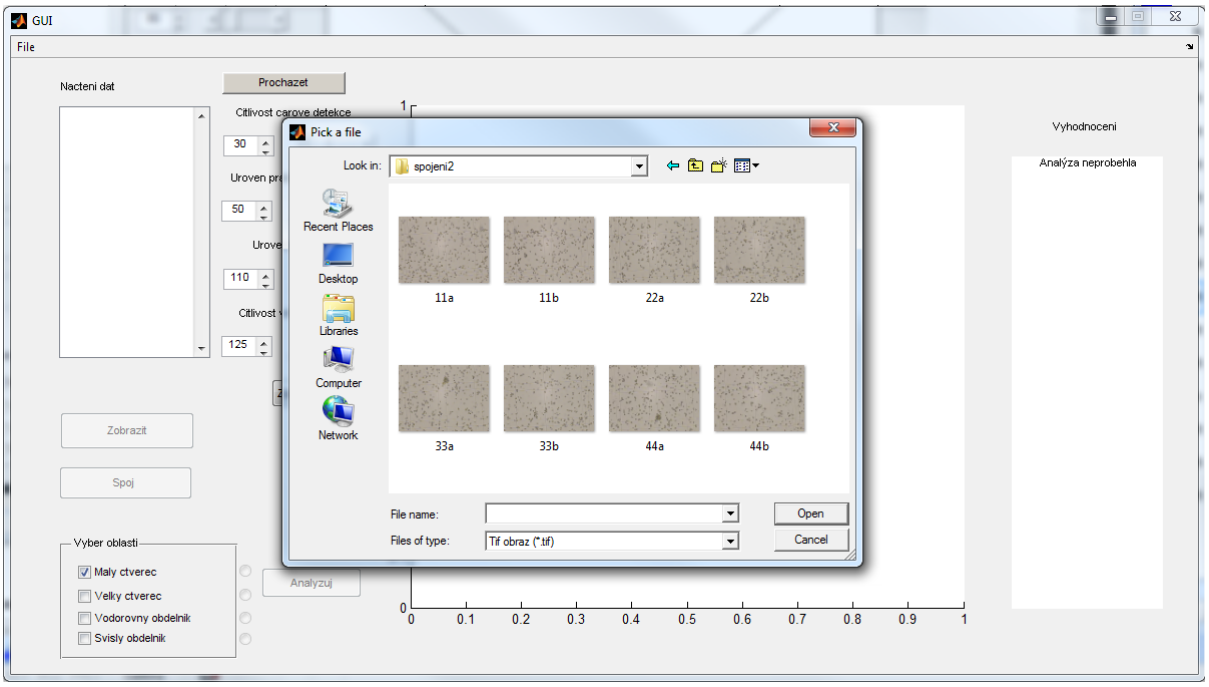
1. Načtení dat k analýze a jejich zobrazení.
2. Detekce mřížek a vyrovnání obrazu spojení obrazu.
3. Rozdělení mřížky na zkoumané oblasti.
4. Analýza zvolených částí.
5. Zobrazení výsledků a hodnocení.



Obr.32 Pohled na základní GUI. 1- tlačítko pro načtení dat, 2- ovládací prvky běhu analýzy, 3-hlavní zobrazovací okno, 4 –pole s vyhodnocením analýzy, 5-seznam načtených a dílčích dat, 6-reset ovládacích prvků, 7- tlačítko pro zobrazení vybraného prvku, 8-tlačítko pro započetí spojení nahraných dat, 9-indikace již proběhlých analýz, 10-tlačítko pro spuštění analýzy, 11- výběr části pro analýzu.

7.1 Načtení a zobrazení dat

Vstupní data jsou předpokládána typu TIFF. Jedná se o obrazová data s bezeztrátovou popřípadě žádnou datovou kompresí. Uživatelské prostředí by mělo nabídnout možnost načtení dat pomocí standardního navigačního dialogového okna. Výběr vlastních souborů je v základu omezen na soubory typu TIFF, je však možné zobrazit veškeré soubory v dané složce.



Obr.33 Selekce vstupních dat

Zde se dá využít vestavěných funkcí prostředí Matlab. Při kliknutí na zvolené tlačítko dochází k zavolání událostní funkce. Až v této funkci řešíme vyvolání příslušného dialogového okna, načtení dat a následnou práci s daty a jejich výpis. Postup je rozdělen do několika kroků:

Načtení dat

Použitá je vestavěná funkce Matlabu : *uigetfile*

```
[filename, pathname] = uigetfile({  '*.tif','Tif obraz
(*.tif)'; '*.*',  'All Files (*.*)'}, 'Pick a file',
MultiSelect', 'on');
```

Vstupem jsou parametry specifikující možnosti dialogového okna.
Výstupem je struktura obsahující názvy souborů, cesta k souborům.

Vypsání a uložení dat

Pomocí funkce : *nacteniZGUI*

```
[vse] = nacteniZGUI(filename, pathname,handles);
```

Vstupem jsou data načtená v předcházejícím kroku a struktura ukazatelů na jednotlivá rozhraní GUI.

Výstupem je uspořádaná struktura, která obsahuje vlastní data a další základní informace. Je vstupem pro další metody použité dále.

Vlastní metoda data načte, setřídí, vytvoří jejich monochromatický obraz (pomocí 3.1.2.1 Implementace) a zobrazí jejich seznam do Listboxu na hlavní ploše GUI.

Zobrazení vybraných dat

Tato volba je možná až po načtení dat. Princip je následující: vybráním konkrétního řádku v Listboxu a kliknutím na příslušné tlačítko je zavolána obsluhující funkce, kde je určen index vybraného řádku Listboxu a pomocí něj určen vybraný soubor. Ten je pak zobrazen. K přístupu k položkám GUI jsou použity metody Matlabu *get* a *set*.

Pomocí funkce : *zobrazNahraneGUI*

```
[] = zobrazNahraneGUI(vse, ListBoxIndex, handles,soubor);
```

Vstupem jsou načtená data, index řádku, struktura ukazatelů na jednotlivá rozhraní GUI a jméno souboru.
Tato metoda nemá žádný výstup.

7.2 Detekce mřížek, spojení a vyrovnání obrazu

Tato oblast programu zastupuje základní část předzpracování obrazových dat v této práci. Úkolem je rozpoznat mřížky Bürkerovy komůrky, spojit je do jednoho celku a vyrovnat natočení obrazu vzniklé při pořizování obrazových dat.

K detekci mřížky je použita Houghova transformace. Tato metoda je aplikována na dva rohové snímky a jeden sousední. Poté následuje další část programu: fázová korelace, kde je vytvořen celkový hrubý obraz mřížky. Následně se určí natočení obrazu a provede se korekce. Pomocí dříve určených mřížek ve třech snímcích je možné určit rozložení mřížek ve zbývajících obrazech, respektive v celkovém spojeném obraze.

V horní části GUI je možné nastavit citlivost Houghovy transformace v rozmezí 10 až 100. Základní hodnota je 30.

Před počátkem detekce a spojování je nutné potvrdit volbu v dialogovém okně, aby nedošlo ke zbytečnému spuštění procesu. To je vhodné vzhledem k délce procesu. Celý postup je sledován progress barem.

Detekce mřížek

Použitá je funkce *HoughLineDetect* na detekci čar a funkce *Vykresli* na následné vykreslení čar do původního obrazu.

Implementace těchto funkcí a definování vstupů a výstupů koresponduje s popisem v kapitole 6.2.1 Implementace

Spojení obrazu

Spojením obrazů použitím fázové korelace se zabývá kapitola 6.3 Fázová korelace. Zde je pouze aplikována na posloupnost vstupních dat a výsledek je poskytnut pro další zpracování.

Rozpoznání zbylých mřížek

Pomocí funkce : *lineCop*

```
[PicArray] = lineCop(PicArray);
```

Vstupem je složený vyrovnaný obraz, kde na třech snímcích byla provedena čárová detekce.
Výstupem je obraz s označenou celou mřížkou.

Metoda pomocí již detekovaných rohových snímků určí rozložení čar v dané ose. Spolu s obrazem třetím a s faktem, že rozměry segmentů mřížky jsou stejné pro obě osy, určí rozložení ve zbylých částech celkového obrazu.

Natočení obrazu

Pomocí funkce : *natoc*

```
[PicArray] = natoc(PicArray, uhel);
```

Vstupem je složený obraz a úhel, o který se má natočit.
Výstupem je obraz natočený o daný úhel.

Jedná se o jednu ze základních metod geometrické transformace. Transformace otočení okolo počátku souřadnic o úhel φ je dán:

$$b_x = a_x \cos\varphi - a_y \sin\varphi$$
$$b_y = a_x \sin\varphi + a_y \cos\varphi \qquad (7.1)$$

kde

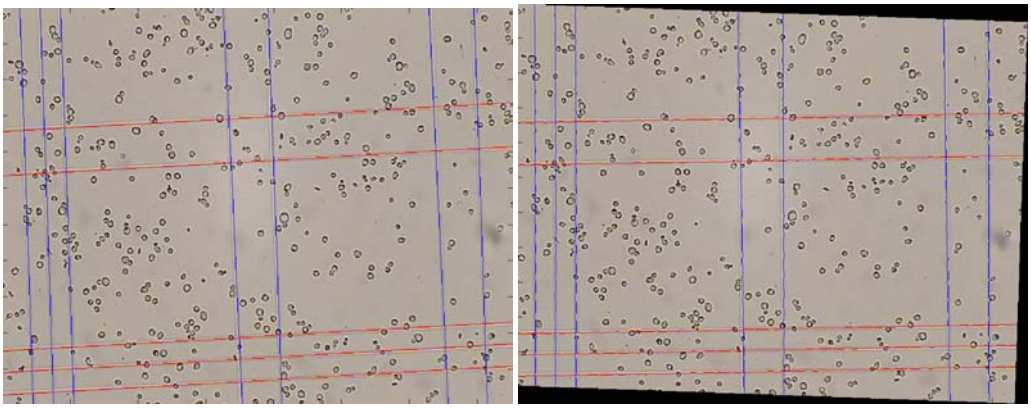
a,b jsou souřadnice bodu.

7.2.1 Implementace natočení

```
funkce : natoc
vstup :  obraz, úhel natočení
výstup : natočený obraz

PicOut = natoc(PicArray,uhel)
{
radian = 2* pi * (uhel /360);
for x=1 to width(PicArray)
      for y=1 to height(PicArray)
            bx  =  round( i * cos(radian) - j*sin(radian));
            by  =  round( i * sin(radian) + j*cos(radian));

            PicOut (bx,by,1) = PicArray(i,j,1);
            PicOut (bx,by,2) = PicArray(i,j,2);
            PicOut (bx,by,3) = PicArray(i,j,3);
      end
end
}
```



Obr.34 Vyrovnání obrazů

7.3 Rozdělení mřížky na zkoumané oblasti

Jak je patrné z kapitoly 2.3.1 Vegetativní rozmnožování, zkoumaná komůrka se skládá z oblastí různých velikostí. Toto dělení respektive následná volba zkoumané části mřížky má zásadní vliv na časovou náročnost analýzy. Standardní mřížka je složena ze 49 částí. Z toho 16 je tvarem čtverec o ploše 1/25 mm^2, 24 má tvar svislých nebo vodorovných obdélníků o ploše 1/100 mm^2 a 9 jsou malé čtverce o ploše 1/400 mm^2.

Obr.35 Typy zkoumaných tvarů

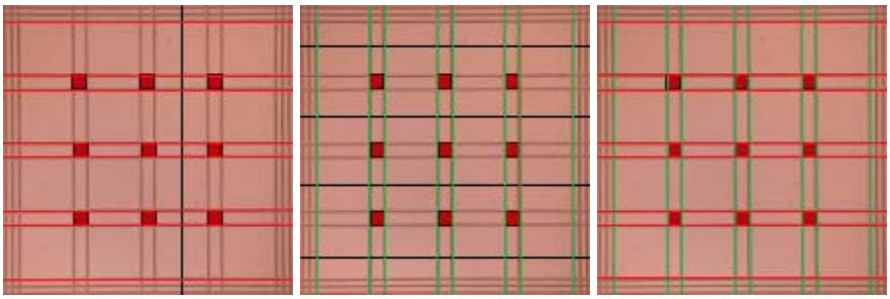
Rozdělení

O rozdělení se stará funkce : *rozdeleniNaCasti*

```
[casti] = rozdeleniNaCasti(PicArray,casti,PicArrayOrig)
```

Vstupem je celkový obraz mřížky s vyznačenými čarami, struktura pro uchování dat a celkový obraz mřížky bez vyznačených čar.
Výstupem je struktura naplněná daty.

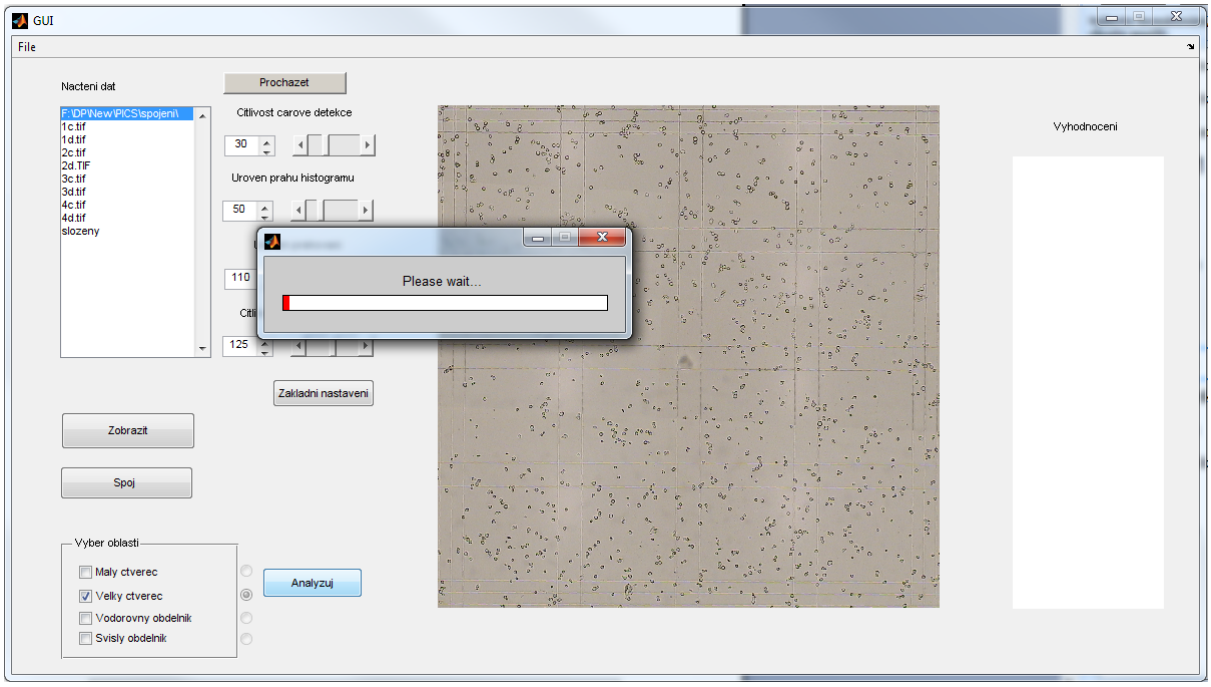
Funkce pomocí detekovaných čar vyhledává příslušné oblasti. Nejdříve jsou proložením ve vertikálním směru nalezeny hranice mřížek a jejich korespondující středy. Pak je pro každý tento střed provedena detekce horizontálních hranic. Složením těchto údajů získáme hrubou představu o souřadnicích hranic každé části mřížky. Následně proběhne přesnější vyhledání hranic pro každou část mřížky zvlášť, otypování, uložení souřadnic a vložení výřezu mřížky do struktury.



Obr.36 Rozdělení na části. Nalezení horizontálních hranic (vlevo), vertikální hranice (uprostřed), složení hranic (vpravo).

7.4 Analýza zvolených částí

V tomto kroku dochází k vlastní analýze. Ta je prováděna na dříve připravených datech. Volba jaká data zpracovat se nabízí v levé spodní oblasti GUI. Vybráním příslušných checkboxů dáme najevo, které oblasti komůrky se mají analyzovat. Analýza se spouští tlačítkem vedle *Analyzuj*. Z důsledku značné časové náročnosti větších částí komůrky, jsou již analyzované části v důsledku prevence nechtěného znovu vybrání indikovány příslušnými radiobutony. Vlastní proces analýzy je mapován progress barem.

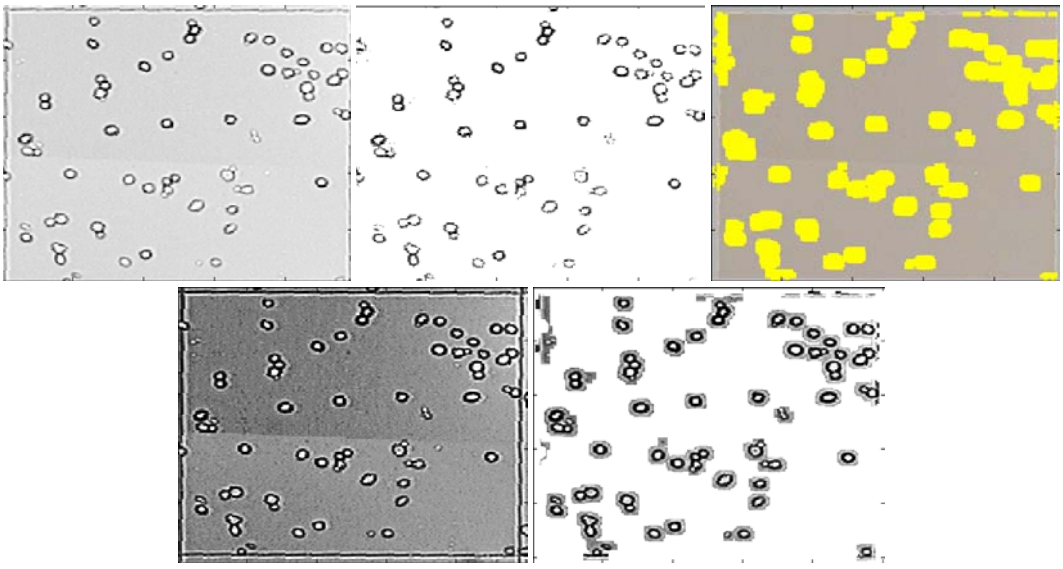


Obr.37 Proces analýzy

Vlastní postup analýzy se dá rozdělit do několika kroků, kde v každém dochází k zjemnění filtrování. Vstupem je obraz části mřížky.

První část

Základem je vytvoření monochromatického obrazu pomocí algoritmu popsaného zde. Následně vytvoříme histogram tohoto obrazu a vybereme jen ty hodnoty jasu které jsou větší než zvolený práh. Volba prahu je možná v horní části GUI. K tomuto účelu slouží funkce redukceHisto. Označíme blízké okolí vybraných bodů. Vytvoříme obraz vyrovnáním histogramu vstupního obrazu části mřížky. Proložíme jej s okolím získaným v předchozím kroku. Tímto získáme hrubou představu o objektech v obrazu.



Obr.38 První část analýzy

Část druhá

V této části dochází k prahování [6.1.1 Metoda automatického nalezení prahu], kde hodnota prahu je nastavitelná v GUI. Následně se odfiltrují malé shluky osamocených pixelů. Pak dochází k základnímu ohraničení zbylých objektů na obraze a odstranění objektů nestandardních tvarů. Tuto činnost zajišťuje funkce *ohraničObjek*.

```
[G,H]= ohranicObjekt(PicArray,PIC);
```

Vstupem je filtrovaný a prahovaný obraz části mřížky a původní vstupní obraz části mřížky.
Výstupem je původní obraz z ohraničenými objekty a matice zaznamenávající přesné pozice objektů v obraze

Funkce postupně prochází objekty v obraze vytváří okolo nich hranici. V případě nutnosti dopočítává malé mezery(řádově dva až tři pixely) vzniklé při pořizování snímku ,např. špatným osvětlením nebo zaostřením, a následným prahováním.

Část třetí

Zde dochází k rozlišení vlastních (funkce *findCells*) objektů (kvasinek). Zejména rozdělení na osamělé a pučící [2.3.1 Vegetativní rozmnožování] nebo kolonie kvasinek. Ke každému případu se pak přistupuje zvlášť (funkce *oneColorCell* a *MultiColorCell*). Nalezené rozlišené objekty jsou zaznačeny do původního obrázku.

```
[G,H] = findCells(G,H,PIC );
```

Vstupem je původní obraz z ohraničenými objekty, matice pozic objektů a původní vstupní obraz části mřížky.
Výstupem je obraz s rozčleněnými objekty a matice zaznamenávající přesné pozice objektů v obraze.

Každý objekt v obraze je postupně vyhodnocen dle počtu oblastí, které by mohly tvořit tělo buňky.

```
[G,H] = oneColorCell(G,H,PIC);
```

Vstupem je původní obraz z ohraničenými a do skupin rozdělenými objekty, matice pozic objektů a původní vstupní obraz části mřížky.
Výstupem je obraz s rozčleněnými objekty, kde jsou již zpracované jednoduché buňky. Matice zaznamenávající přesné pozice objektů v obraze.

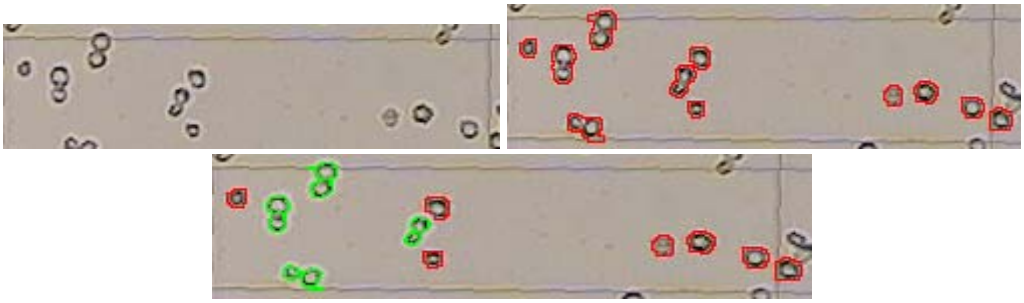
Metoda se zaměřuje na kandidáty pro osamocené buňky. Těchto buněk bývá zpravidla nejvíce. Ve výsledných obrazech jsou zaznačeny červenou barvou.

Nejdříve je dle matice pozic určeno místo buňky v obraze a do něj je vložen obrázek s vyrovnaným histogramem, který vzhledem k již dříve vymezenému prostoru pro buňku, velice slušně popisuje skutečný obvod buňky. Tento obvod je následně obtažen červenou linií.

```
[G,H] = MultiColorCell(G,H,PIC);
```

Vstupem je původní obraz z ohraničenými a do skupin rozdělenými objekty. Zde již jsou zpracované jednoduché buňky. Dalším vstupem je matice pozic objektů a původní vstupní obraz části mřížky.
Výstupem je obraz s rozčleněnými objekty. Matice zaznamenávající přesné pozice objektů v obraze.

Tato metoda zpracuje zbylé buňky. Jedná se nejčastěji o seskupení více buněk pohromadě. Množství těchto typů buněk je různorodé, je také možné, že není nalezená jediná taková buňka. Ve výsledných obrazech jsou zaznačeny zelenou barvou.



Obr.39 Příklad rozlišování buněk.

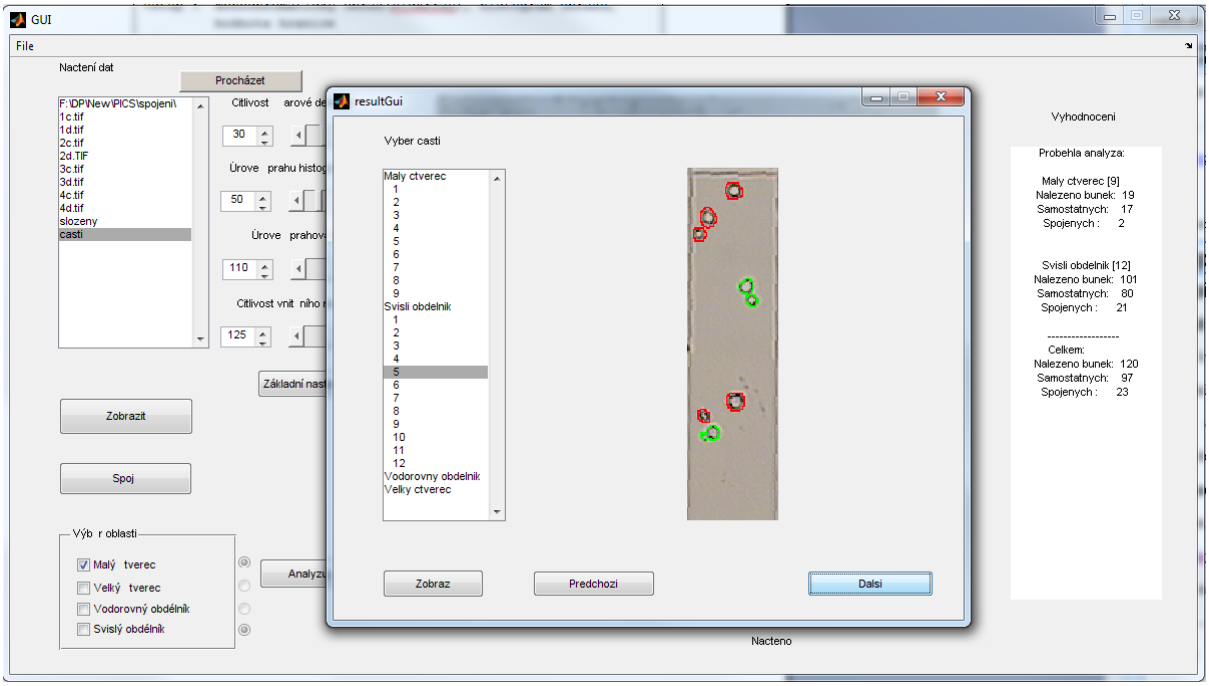
7.4.1 Implementace použitých funkcí

```
funkce : redukceHisto
vstup :  monochromatický obraz(PicArray), histogram obrazu,
         hodnota hranice
výstup : upravený obraz(PicOut)

[PicOut ] = redukceHisto(PicArray,histo,hranice)
{
radian = 2* pi * (uhel /360);
for x=1 to width(PicArray)
      for y=1 to height(PicArray)
            if histo(PicArray(x,y))
                  (PicArray(x,y) = 255;
            end
      end
end
}
```

7.5 Zobrazení výsledků a hodnocení

Jedná se o poslední krok ve zpracování vstupních snímků. Po provedení analýzy minimálně jednoho typu výřezu mřížky je umožněno zobrazit výsledky analýzy. Postup je podobný jako při zobrazování vstupních dat (7.1 Načtení a zobrazení dat), ale vzhledem k četnosti a velikosti dat, která by se měla zobrazit, je použito pomocné GUI. Toto GUI slouží pouze k účelu zobrazení dat. Navigace probíhá pomocí tlačítek *další* a *předchozí*. Tlačítko *Zobraz* zobrazí aktuální výběr v příslušném ListBoxu.



Obr.40 Sekundární GUI pro procházení obrázků

Vlastní vyhodnocení probíhá automaticky po analýze zvolené části celkové mřížky. Vyhodnocení jsou zobrazena v pravé části základního GUI. Obsahuje informace jednak o typu a počtu analyzovaných výřezů, tak o počtu nalezených buněk a hlavně o počtu jednotlivých druhů buněk.

Hodnocení

Jedná se o finální část procesu. Metoda *vyhodnoceniAnalyza* zpracovává nalezené výsledky v předchozích krocích běhu programu.

```
vyhodnoceniAnalyza(stav,handles,casti)
```

Vstupem je informace o provedených analýzách, struktura ukazatelů na jednotlivá rozhraní GUI a struktura obsahující jednotlivé zpracované části mřížky.
Metoda nemá výstup.

Dle informace o stavu analyzovaných dat jsou postupně procházena data v datové struktuře. Pro každý typ mřížky je spočítán výskyt daných typů buněk. Výsledky jsou zobrazeny v textovém poli v pravé části GUI.

8. Závěrečné shrnutí

V této práci je vyzkoušeno a implementováno několik základních metod zpracování obrazu s cílem detekce mikrobiologických entit v analyzovaném snímku. Vzhledem k rozmanitosti dat, na kterých jsou v práci použité numerické metody aplikovány, bylo dosaženo celkem pozitivních výsledků. Základním cílem práce bylo navržení postupu, který by dokázal postupným zpracováváním vstupních dat získat hledané výsledky. Houghova transformace použitá pro čárovou detekci se jeví jako robustní řešení tohoto problému. Stejně tak aplikace fázové korelace využitím Fourierovy transformace přináší výborné výsledky. Největší problém představuje následné zpracování vlastních dat na snímku. Rozlišení hledaných objektů od nečistot popřípadě jiných organických entit je provedeno vícenásobným filtrováním, kdy pokaždé zjemňujeme naše hledání. Tento postup dává subjektivně odpovídající výsledky. Navržené uživatelské prostředí je směřováno na jednoduché a intuitivní ovládání, přístup k výsledkům je taktéž snadný.

Vzhledem k blíže nedefinovaným časovým požadavkům je vývojové prostředí Matlab vhodným základem pro stavbu výpočtových aplikací. V případě potřeby zrychlení výpočtu by bylo možné provedení optimalizace kódu. Pro případné širší nasazení by bylo vhodné zvážit použití jiného vývojového prostředí, které by se více orientovalo na výkonovou stránku výpočtu.

Seznam použité literatury

[1] Hrubiško M., *Hematologie a krevní transfúze I : Hematologie*. [s.l.], Avicenum, 1981, 274 s.

[2] Šilhánková Ludmila, *Mikrobiologie pro potraviná*ře *a biotechniky* 3. vyd. [s.l.], Academia, 2002, ISBN 80-200-1024-6, s. 57-84.

[3] Vesela Marián, Drdák Milan, *Praktikum z obecné Mikrologie*, 2. vyd., Brno, VUT-CHEM 1999, 125 s., ISBN 80-214-1305-0.

[4] WU Qiang, Merchant Fatia , Castleman Kenneth R., *Microscope Image Processing,* UK: Elsorver inc. - Academic Press, 2008, 830 s., ISBN 978-0-12-372578-3.

[5] Dobeš Michal, Zpracování obrazu a algoritmy v C#. 1. vyd., Praha BEN, 2008, 143 s., ISBN 978-80-7300-233-6.

[6] Gonzalez R.C., Woods R., Eddins S.L., Digital Image Processing using Matlab, Prentice Hall, ISBN-13 978-0130085191, 2004.

[7] *Houghova transformace* [online]. 2000 [cit. 2008-10-24]. Dostupný z WWW: <http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv5/hough.htm>.

[8] *Obrazové segmentační techniky : Přehled existujících metod* [online]. 2005 [cit. 2006-01-19]. Dostupný z WWW: <http://www.fit.vutbr.cz/~spanel/segmentace/.cs.iso-8859-2#_Toc125769322>.

[9] *Documentation for MathWorks Products, R2009a* [online]. 1994 [cit. 2009-04-12]. Dostupný z WWW: <http://www.mathworks.com/access/helpdesk/help/helpdesk.html>.

[10] *RGB,* [online]. 2009 [cit. 2009-05-12]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/RGB>.

[11] Phase correlation*,* [online]. 2009 [cit. 2009-05-12]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Phase_correlation>.

[12] Fast Normalized Cross-Correlation*,* [online]. 2009 [cit. 2009-05-12]. Dostupný z WWW: <http://www.idiom.com/~zilla/Papers/nvisionInterface/nip.html>.

[13] Fourierovy řady*,* [online]. 2009 [cit. 2009-05-12]. Dostupný z WWW: <http://mathonline.fme.vutbr.cz/Fourierovy-rady/sc-73-sr-1-a-60/default.aspx>.

Přílohy

Přílohou je CD s elektronickou verzi diplomové práce a aplikace.