VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF BIOMEDICAL ENGINEERING

MIKROSKOPICKÉ SNÍMKY S VYSOKOU HLOUBKOU OSTROSTI

MICROSCOPIC IMAGES WITH HIGH DEPTH OF FOCUS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Mgr. MARTINA RŮŽIČKOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. VRATISLAV ČMIEL
SUPERVISOR

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav biomedicínského inženýrství

Diplomová práce

magisterský navazující studijní obor
Biomedicínské a ekologické inženýrství

Studentka:	Mgr. Martina Růžičková	***ID:***	114861
Ročník:	2	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Mikroskopické snímky s vysokou hloubkou ostrosti

POKYNY PRO VYPRACOVÁNÍ:

1) Seznamte se s optickým mikroskopem a s pořizováním mikroskopických fotografií. Prostudujte problematiku ostření a možnosti využití obrazového zpracování pro dosažení ostrého snímku s extrémně velkou hloubkou ostrosti ze skupiny snímků. Proveďte literární rešerši v této oblasti. 2) Vyberte vhodné algoritmy pro účely zvýšení hloubky ostrosti. 3) Vytvořte návrh metody a zabývejte se možností využití prostředí Matlab k tomuto účelu. 4) Realizujte programové řešení algoritmu pro zvýšení hloubky ostrosti. 5) Popište vlastní řešení. 6) Ověřte systém na reálných datech. 7) Zhodnoťte možnosti systému a možnosti praktického využití.

DOPORUČENÁ LITERATURA:

[1] ZAPLATÍLEK, K. MATLAB : tvorba uživatelských aplikací. Praha : BEN - technická literatura, 2004.
[2] SONKA, M., HLAVAC, V., BOYLE, R. Image Processing, Analysis, and Machine Vision. CL-Engineering, 2007.

Termín zadání:	6.2.2012	***Termín odevzdání:***	18.5.2012

Vedoucí práce: Ing. Vratislav Čmiel

prof. Ing. Ivo Provazník, Ph.D.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Cílem této práce je seznámení se s optickým mikroskopem a s možností pořizování mikroskopických fotografií. Dále se práce zabývá problematikou ostření a dosažení snímku s extrémní hloubkou ostrosti za předpokladu využití sady snímků s různými rovinami ostrosti. Pomocí vytvoření vhodného algoritmu pro zvýšení hloubky ostrosti pak provést obrazové zpracování této sady snímků a získat jeden výsledný obraz s vysokou hloubkou ostrosti.

Abstract

The aim of the paper is to introduce an optic microscope as well as microscopic photography producing. It further engages in sharpening issues and attaining a sharp image with extremely robust depth acuteness assuming using a series of images with different acuteness levels. The goal is to create a suitable algorithm to increase depth acuteness, to perform a visual processing of the series of images and to get one final image with high depth acuteness.

Klíčová slova

Světelný mikroskop
Mikroskopická fotografie
Rozlišovací schopnost
Hloubka ostrosti

Key words

Light microscope
Microscopic photography
Resolution
Depth acuteness

RŮŽIČKOVÁ, M. *Mikroskopické snímky s vysokou hloubkou ostrosti*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 77 s. Vedoucí diplomové práce Ing. Vratislav Čmiel.

Čestné prohlášení

Prohlašuji, že svou diplomovou práci na téma Mikroskopické snímky s vysokou hloubkou ostrosti jsem vypracovala samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně, dne 17. května 2012 ……

podpis autora

Poděkování

Ráda bych tímto poděkovala Ing. Vratislavu Čmielovi za jeho vědecké připomínky i odborné rady k dané problematice. Dále Ing. Ivanu Skalkovi za jeho pomoc při tvorbě algoritmu pro zvýšení hloubky ostrosti.

V Brně, dne 17. května 2012 ……

podpis autora

Obsah

Seznam obrázků

1 Úvod

Jedním z hojně rozšířenějších optických přístrojů je mikroskop, který nachází široké uplatnění v řadě odvětví vědy a techniky. Pomocí mikroskopu můžeme studovat rozličné vlastnosti předmětů, v závislosti na charakteru vyšetřovaného předmětu pak volíme různé mikroskopické metody.

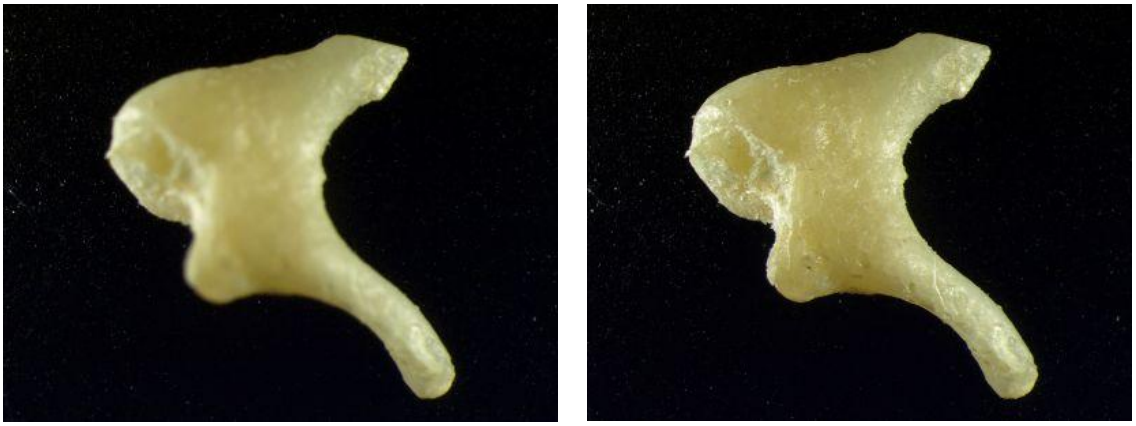
Jak je patrné z následujících obrázků, mikroskop nám odhaluje odlišný svět, který je fascinující svou jedinečností a rozmanitostí, ukazuje detaily, jež naše oko nedokáže v reálném životě zachytit. Díky této technice můžeme pozorovat různé látky a předměty, na kterých se učíme poznávat, z čeho a jak jsou věci složené, jaký mají význam.

Obr. č. 1: Mikroskopická fotografie příčné pruhovaného svalu, podélný řez [18], makroskopická fotografie deltového svalu

Světelná mikroskopie prodělala během minulého století ohromný rozmach. Zobrazení, jaké nám dnešní mikroskopy umožňují, je velmi dokonalé. Byly vynalezeny různé mikroskopické techniky, které nám umožňují odhalit a pozorovat struktury, jež jsou za normálních podmínek velmi těžce pozorovatelné. Dnešní světelný mikroskop dovoluje pozorovat objekty až při více než tisícinásobném zvětšení. Současná mikroskopická technika je nerozlučně spjata s počítači, digitálními kamerami a fotografickými zařízeními. Větší zvětšení než optický mikroskop nám poskytuje elektronový mikroskop. Jeho zvětšení může být stotisíckrát až milionkrát a s jeho pomocí byly objeveny například viry. Na rozdíl od světelného mikroskopu však jím většinou nelze pozorovat živé objekty.

Dnešní doba je typická překonáváním hranic, ať už se jedná o vývoj menších a výkonnějších počítačových čipů, sportovní výkony či nové technické objevy. Podobně je tomu i ve fotografování. Zde je však častou limitací hloubka ostrosti. Technika skládání obrazů s různými rovinami ostrosti umožňuje nový pohled na fotografované objekty a přináší téměř neomezenou hloubku ostrosti. Při fotografování drobných tvorů (například menších než 1 cm) je hloubka ostrosti velmi malá a při nárůstu zvětšení se nám stále zmenšuje. Tento problém lze řešit opakovaným fotografováním daného

objektu a postupně proostřovat jeho jednotlivé části, tj. pomocí vytvořeného programu (dnes je na trhu možnost zakoupení několika takových programů, např. Modul Deep Focus od Promicry) porovnat jednotlivé fotografie mezi sebou a z každé použít ke složení výsledného obrazu jen jeho ostrou část. Výsledný obraz pak vykazuje vysokou hloubku ostrosti.



Obr. č. 2: Snímky středoušní kůstky, vlevo běžný snímek, vpravo proostřený snímek, složen z 16 řezů programem Modul Deep Focus [16]

Obr. č. 3: Snímky klíštěte, vlevo běžný snímek, vpravo proostřený snímek, složen z 6 řezů programem Modul Deep Focus [16]

2 Mikroskopie

2.1 Historie mikroskopie

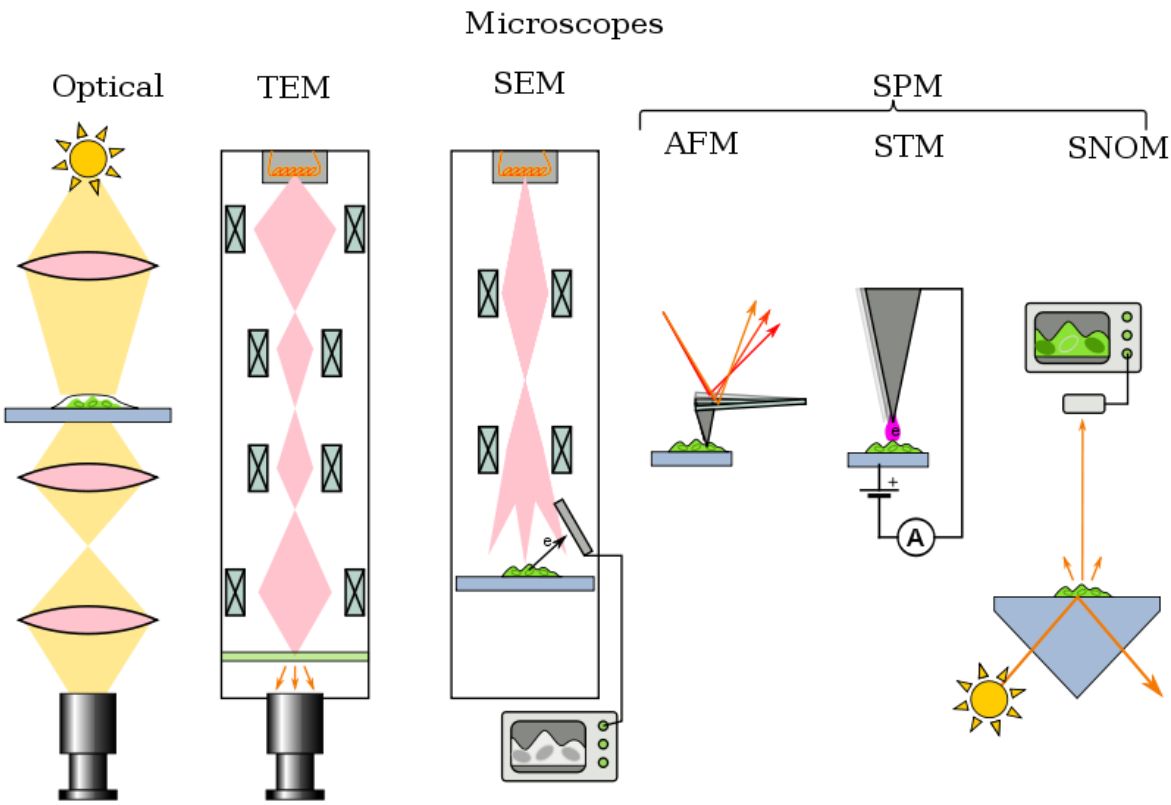
Mikroskop je přístroj, který lidstvu odhalil krásy mikrosvěta, a známe jej již více než 400 let. Tento vynález není připisován jedné osobě. Patrně jako první sestrojil mikroskop někdy kolem roku 1590 Zacharias Jansen, holandský brusič čoček a výrobce brýlí. Dále tento vynález vylepšil italský astronom a matematik Galileo Galilei.

Mikroskop používal také holandský obchodník Antonius van Leeuwenhoek, jenž přispěl významnou měrou ke zdokonalení dosud primitivního přístroje. Vymyslel, jak přesně vybrousit čočky a jak je sestavit a upevnit, aby vytvořily silný zvětšovací efekt. Také jako první na světě viděl a popsal krevní buňky. Velkou nevýhodou bylo, že mikroskop byl v této době jednočočkový. Tím byly jeho možnosti omezeny. O mikroskop složený z více čoček se v roce 1665 zasloužil anglický fyzik a chemik Robert Hooke. Ten v uvedeném roce vydal dílo Micrographia, v němž popsal konstrukci mikroskopu s odděleným objektivem, okulárem a osvětlovacím zařízením.

Dramatický vývoj pak zažívá mikroskopie v 19. století. K tomuto přispěl Carl Zeiss, který v roce 1847 zahájil výrobu mikroskopů, dále Ernst Abbe, jenž se zabýval studií optických principů, a Otto Schott, který se zajímal o výzkum optického skla.

Optický mikroskop však má své teoretické hranice, jež jsou dány rozlišovací mezí (viz níže) a vlnovou délkou viditelného světla. Bylo tedy nutné přemýšlet jiným směrem a zkonstruovat mikroskop na odlišném principu, a hlavně využít jiný druh „osvětlení“ o kratší vlnové délce. Byl vyvinut elektronový mikroskop, který místo světelného paprsku využívá elektronový svazek a místo skleněné čočky je čočka magnetická. Mikroskopie na tomto principu se poprvé objevila v Německu v roce 1931.

V 80. letech minulého století vznikla tzv. rastrovací sondová mikroskopie. Roku 1981 byl vytvořen první řádkovací tunelový mikroskop (STM) Gerdem Binningem a Heinrichem Rohrerem. O pět let později, v roce 1986, spatřil světlo světa mikroskop atomárních sil (AFM).



Obr. č. 4: Typy mikroskopů [3]

2.2 Světlo

Viditelné světlo je elektromagnetické záření o vlnové délce 400 – 750 nm. Vlnové délky světla leží mezi vlnovými délkami ultrafialového záření a infračerveného záření. Světlo má dualistickou povahu, což znamená, že ho můžeme chápat jako příčné vlnění spojitého elektromagnetického pole a současně jako proud částic (fotonů).

Mezi základní charakteristiky světla patří:

- Rychlost světla (c) – ve vakuu je $3 \cdot 10^8$ m.s^{-1}, v látkovém prostředí je vždy nižší.
- Frekvence (f) – počet kmitů za jednotku času, nezávisí na prostředí a udává barvu světla, s rostoucí frekvencí roste energie světla.
- Vlnová délka (λ) – mezi frekvencí, vlnovou délkou a rychlostí platí vztah $\lambda = c/f$.
- Amplituda (A) – udává největší odchylku sinusoidy od nulové hodnoty, na amplitudě závisí intenzita světla.
- Fáze (φ) – udává, v jaké části vlny (sinusoidy) se vlnění nachází v daném časovém okamžiku.

Světlo tvořené vlněním o jedné vlnové délce se nazývá monochromatické, o více vlnových délkách polychromatické. Světlo tvořené všemi viditelnými vlnovými délkami se jeví jako bílé.

Světlo se prostorem šíří podle tzv. Huygensova principu. Ten říká, že každý bod vlnoplochy v prostoru se stává zdrojem elementárního záření. V opticky homogenním prostředí se světelné paprsky šíří přímočaře a nezávisle na sobě. V nehomogenním prostředí může dojít k odrazu, lomu nebo ohybu světla, případně k jeho absorpci.

2.3 Světelná mikroskopie

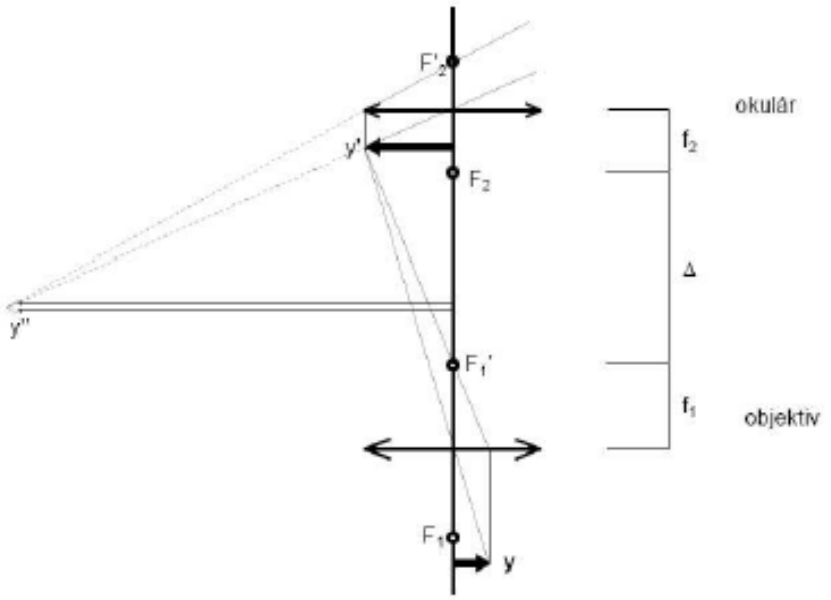
2.3.1 Světelný mikroskop

Mikroskop je nástroj, který nám umožňuje pozorovat předměty, jež jsou pro sledování pouhým okem příliš malé. Snažíme-li se dobře vidět nějaký drobný předmět, přibližujeme si jej k oku. Tím si zvětšujeme zorný úhel. Pokud ovšem u zdravého oka umístíme předmět do menší vzdálenosti, než je konvenční zraková vzdálenost (25 cm), vidíme ho nezřetelně a rozmazaně. Mikroskop umožňuje zvětšení zorného úhlu díky soustavě čoček, a tím dovoluje prohlédnutí malých předmětů.

Mikroskop se skládá ze tří základních částí:

- optické soustavy
- osvětlovací soustavy
- mechanického zařízení

Optická soustava se skládá ze tří částí: objektivu, okuláru a pohyblivého tubusu. Objektiv i okulár mají kladnou optickou mohutnost, tzn., že se chovají jako spojky. Objektiv vytváří zvětšený, skutečný a převrácený obraz. Obraz vytvořený objektivem se stává předmětem pro okulár. Obraz okuláru je zvětšený, zdánlivý a převrácený. V praxi se nejčastěji setkáme se dvěma druhy mikroskopů. Jedná se o mikroskop monokulární a binokulární.



Obr. č. 5: Schéma optické soustavy mikroskopu [7]

Osvětlovací soustava se skládá ze světelného zdroje, kondenzoru, zrcátka a irisové clony. U starších mikroskopů je použito zrcátka, aby odráželo umělé nebo přirozené světlo do optické soustavy mikroskopu. Moderní přístroje mají osvětlovací zařízení zabudováno přímo v těle mikroskopu. Kondenzor je tvořen soustavou čoček o krátké ohniskové vzdálenosti. Jeho hlavním úkolem je soustřeďovat světlo, které prochází pozorovaným preparátem. Kondenzor je uložen pod stolkem mikroskopu. Irisová clona je součástí kondenzoru a umožňuje regulovat kontrast obrazu analyzovaného preparátu.

Mechanická část zahrnuje stativ, nosič tubusu, stolek a revolverovou hlavici. Stativ vytváří velmi pevnou oporu mikroskopu, je spojen svým ramenem s nosičem

tubusu a se stolkem. V horní části tubusu se nachází okulár, ve spodní části je revolverová otáčecí hlavice, v níž jsou umístěny jednotlivé objektivy. Revolverová hlavice umožňuje snadné střídání objektivů. Na těle mikroskopu se nalézají dva šrouby – makrometrický a mikrometrický. Makrometrický šroub slouží k hrubšímu posunu a zaostření pozorovaného předmětu. Mikrometrický šroub umožňuje jemný posuv a přesné zaostření preparátu.

Celkové zvětšení Z mikroskopu se vypočítá jako součin zvětšení objektivu Z_{ob} a okuláru Z_{ok}:

$$Z = Z_{ob} \cdot Z_{ok} = \frac{\Delta \cdot d}{f_{ok} \cdot f_{ob}}, \qquad (1)$$

kde

d ... konvenční zraková vzdálenost,

Δ ... optický interval mikroskopu (vzdálenost mezi obrazovým ohniskem objektivu a předmětovým ohniskem okuláru),

f_{ob} *a* f_{ok} ... ohniskové vzdálenosti objektivu a okuláru.

V tuto chvíli by nás mohlo napadnout, že vhodnou volbou čoček v mikroskopu dosáhneme neomezeného zvětšení preparátu. Takové zvětšení by nám však neumožňovalo rozlišení detailů vzorku. Zvětšení je tedy omezeno rozlišovací mezí mikroskopu:

$$\delta = \frac{\lambda}{n \cdot \sin \alpha}, \qquad (2)$$

kde

δ … mřížková konstanta (vzdálenost dvou vrypů mřížky),

λ … vlnová délka použitého světla,

n … index lomu prostředí mezi čelem objektivu a krycím sklíčkem preparátu,

α … úhel svíraný optickou osou mikroskopu a pláštěm kuželu, v němž se nacházejí paprsky, které z daného místa preparátu mohou vstoupit do objektivu a podílet se na zobrazení.

Výraz ve jmenovateli se nazývá numerická apertura objektivu.

2.3.1.1 Podstata vzniku obrazu ve světelném mikroskopu

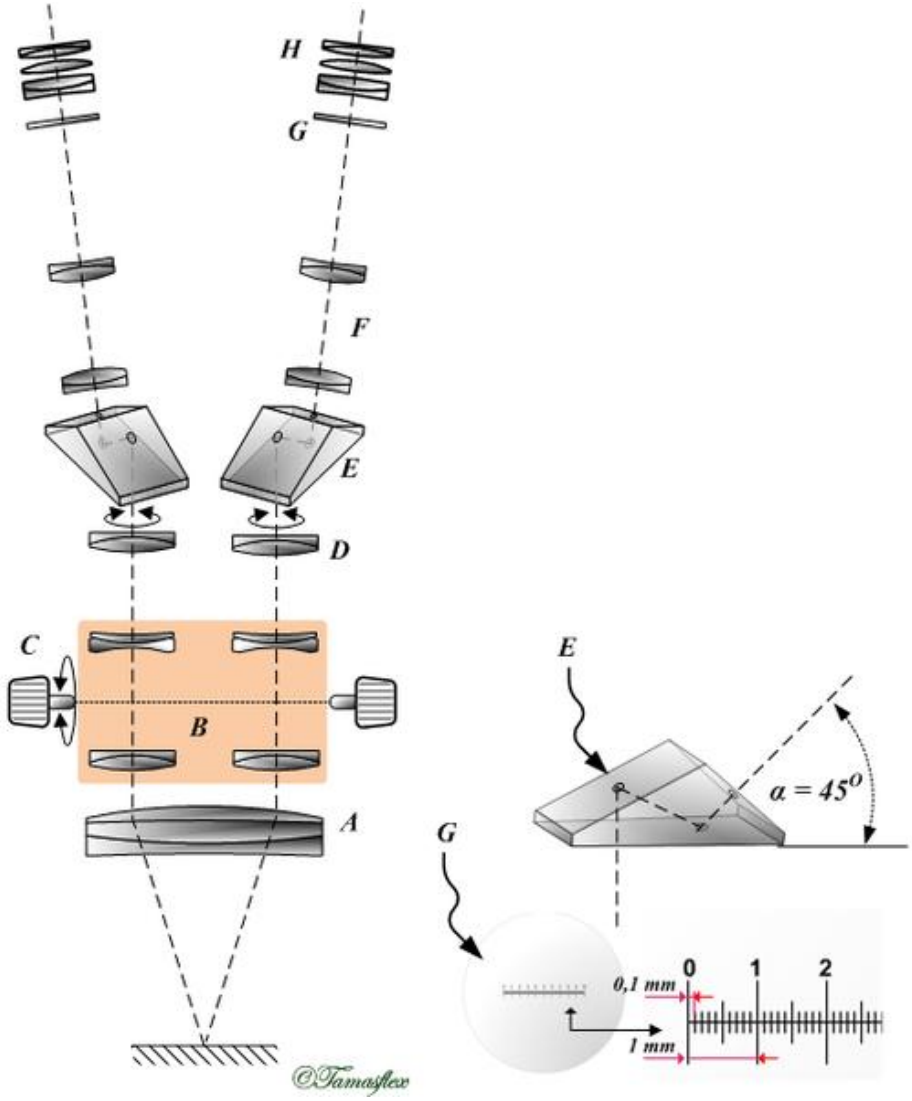
Vysvětlení vzniku obrazu v mikroskopu podal roku 1873 Ernst Abbe. Základní myšlenka se opírá o Huygensův princip – každý bod osvětleného objektu se stává zdrojem sekundárních sférických vln.

Podle optických vlastností jednotlivých bodů objektu se dopadající světlo v každém z bodů transformuje (ohýbá se, láme, mění se jeho amplituda, fáze) a vznikají sekundární vlny. Ty spolu interferují jako po průchodu světla štěrbinou nebo optickou mřížkou. Výsledné vlnění, které obsahuje informaci o vzhledu objektu, vstupuje do objektivu. V jednotlivých bodech zadní ohniskové roviny objektivu se setkávají sekundární vlny, které opustily rovinu předmětu rovnoběžně. Dochází k jejich interferenci a v souladu s Huygensovým principem se stávají zdrojem nových vln, které v obrazové rovině mikroskopu skládají zvětšený a převrácený obraz. [5]

2.3.2 Stereomikroskop

Stereomikroskop se od světelného odlišuje svojí konstrukcí. Stereomikroskopie používá dvě oddělené optické dráhy, aby bylo dosaženo odlišných úhlů pro pravé a levé oko. Tímto způsobem se dosáhne trojrozměrné vizualizace vzorku. Tato mikroskopie je vhodná i pro větší objekty, může být použita např. pro studium povrchu pevných vzorků, při práci v blízkosti vzorku, jako je neurochirurgie, výroba hodinek apod.

Rozdíl oproti běžné mikroskopii je také v tom, že stereomikroskop používá vrchní osvětlení. Odrážející se světlo od povrchu objektu umožňuje vyšetřit části, které by byly pro běžnou mikroskopii příliš silné nebo neprůhledné. Pracovní vzdálenost a hloubka ostrosti jsou dvě vlastnosti, které jsou důležité pro práci se stereomikroskopem. Pro obě vlastnosti platí, že čím je vyšší rozlišení, tím je menší hloubka ostrosti a pracovní vzdálenost.



Obr. č. 6: Složení stereomikroskopu [19]

2.3.3 Hranice možností optických soustav

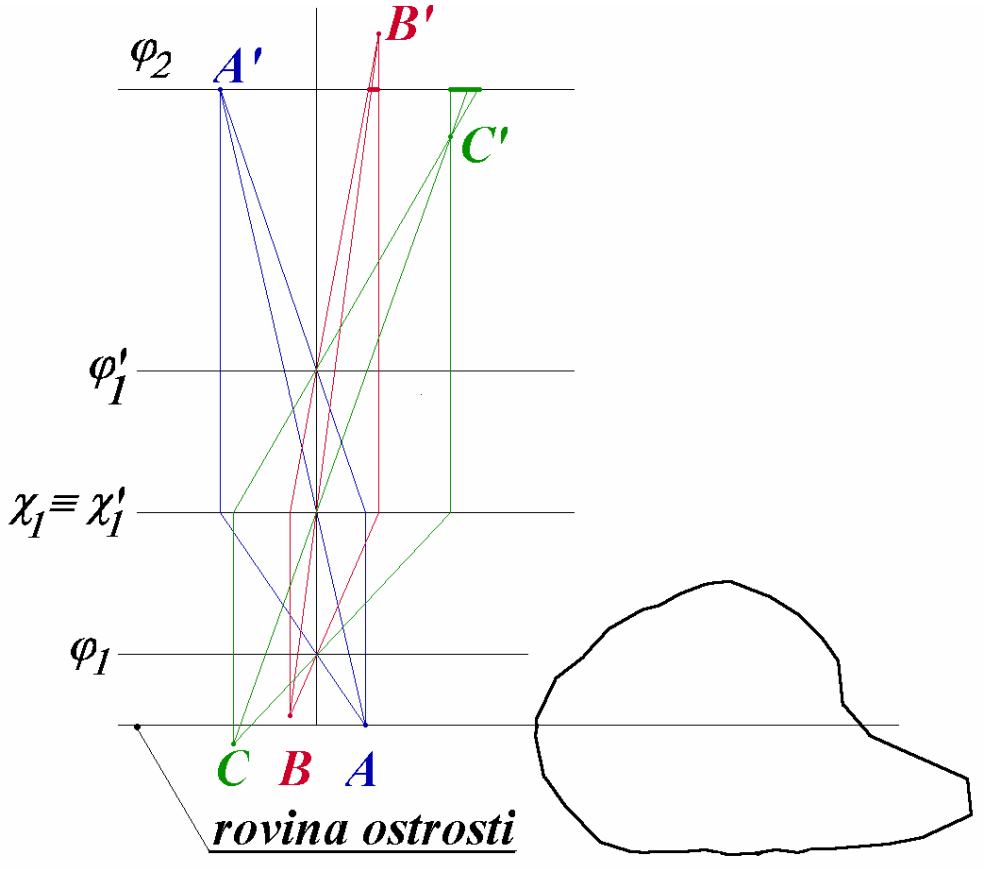
Hranice možností optických soustav jsou dány vlnovou podstatou světla, omezenou šířkou svazku paprsků, nekomplanárností pozorovaného objektu a rozlišovací schopností snímacího zařízení.

Světlo se šíří přímočaře, pokud je splněna podmínka homogenity a izotropního prostředí a pokud se šíří kolem překážek, které jsou mnohonásobně větší, než je délka použitého světla. Při pozorování mikroskopického preparátu je světlo vždy ohýbáno na preparátu samotném. K přesnějšímu popisu funkce optických soustav tedy musíme vzít v úvahu vlnovou podstatu světla. U optických soustav se zavádí tzv. rozlišovací schopnost, jež je dána vzdáleností dvou bodů či čar, které lze ještě od sebe rozlišit.

Rozlišovací schopnost mikroskopu dle Abbeho: Pozorujeme-li mikroskopem drobnou překážku, dochází k ohybu světla a změně světelného paprsku o úhel α. Označme 2*u* maximální úhel paprsků, které projdou mikroskopem. Kdyby bylo $\alpha_1 < u$, pozorovali bychom jen maximum 0. řádu, vytvořené jen paprsky procházejícími předmětovým nevlastním bodem. Protože chod těchto paprsků mřížka nezmění, pozorovali bychom stejnoměrně osvětlené pole. Přítomnost mřížky tedy pozorujeme díky paprskům, pro které je $\alpha_1 \geq u$. Zkušenost ukazuje, že ještě rozlišitelné jsou body, jejichž nulté maximum se kryje s prvním minimem dalšího bodu. [11]

Rozlišovací schopnost lze určit i experimentálně. Vezmeme lineární mřížky s postupně se zvětšujícím počtem čar na jednotku, pozorujeme daným mikroskopem a měříme kontrast. Za rozlišovací schopnost pak prohlásíme rozlišení mřížky, u které kontrast poklesl na předem stanovenou hodnotu α, přičemž za jednotkový kontrast považujeme kontrast obrazu mřížky, jejíž vrypy lze mikroskopem bezpečně rozeznat. Za prahovou hodnotu je nejčastěji považováno $\alpha = 0{,}5$ či $\alpha = 0{,}2$. [11]

Optickou soustavu nikdy nelze zaostřit tak, aby obrazem bodu byl bod. Pokud se má obraz zobrazit ostře, musí se nacházet v tzv. rovině ostrosti, tedy v rovině, na niž je soustava zaostřena. Pozorované předměty však často nelze za rovinné považovat a rovina ostrosti pak protne pozorovaný objekt ve vrstevnici. Body v této vrstevnici se zobrazí s maximální možnou ostrostí, body pozorovaného předmětu nacházející se mimo tuto rovinu se nebudou zobrazovat jako body a budou rozmazané.



Obr. č. 7: Nekomplanárnost pozorovaného předmětu [11]

Další limitací je rozlišovací schopnost snímacího zařízení. Snímací zařízení není schopné zobrazit euklidovský bod na euklidovský bod. Snímací zařízení má konečnou rozlišovací schopnost a zobrazuje bod na pixel s nenulovými rozměry. Při zobrazování se každý bod vlivem výše uvedených jevů zobrazí jako rozmazaná ploška, která se nazývá stopa bodu a má nenulové rozměry. Kvůli tomu, že při snímání není možné zobrazit bod jako bezrozměrný objekt, lze za ostře zobrazený považovat nejen bod, který se zobrazí na bod, ale i bod, jenž se zobrazí na stopu s rozměry menšími, než je pixel snímacího zařízení. Takový snímaný bod se pak nemusí nacházet přesně v rovině ostrosti, ale jen v tzv. pásmu ostrosti.

2.4 Pořizování mikroskopických fotografií

Mikroskopické fotografie se dají pořizovat digitálním fotoaparátem, CCD kamerou nebo digitálním mikroskopem. Pozorovaný a zaznamenaný mikroskopický obraz je dále zpracován pomocí speciálního softwaru na počítači.

První možností je pořizování mikroskopických fotografií digitálním fotoaparátem, který lze připojit na mikroskop. Digitální fotoaparáty jsou určené k připojení přímo do okulárového tubusu mikroskopu a umožňují sledování obrazu z mikroskopu v reálném čase nebo pořizování digitálních fotografií. Velkou výhodou digitální fotografie je možnost okamžité kontroly zhotoveného snímku. Pokud kvalita snímku není odpovídající, je snadné ihned zhotovit nový. Na dnešním trhu jsou dostupné kamery s širokým dynamickým rozsahem, velkou citlivostí a vysokým rozlišením. Na výběr jsou jak barevné, tak i černobílé systémy. Snímek se zaznamenává na paměťové

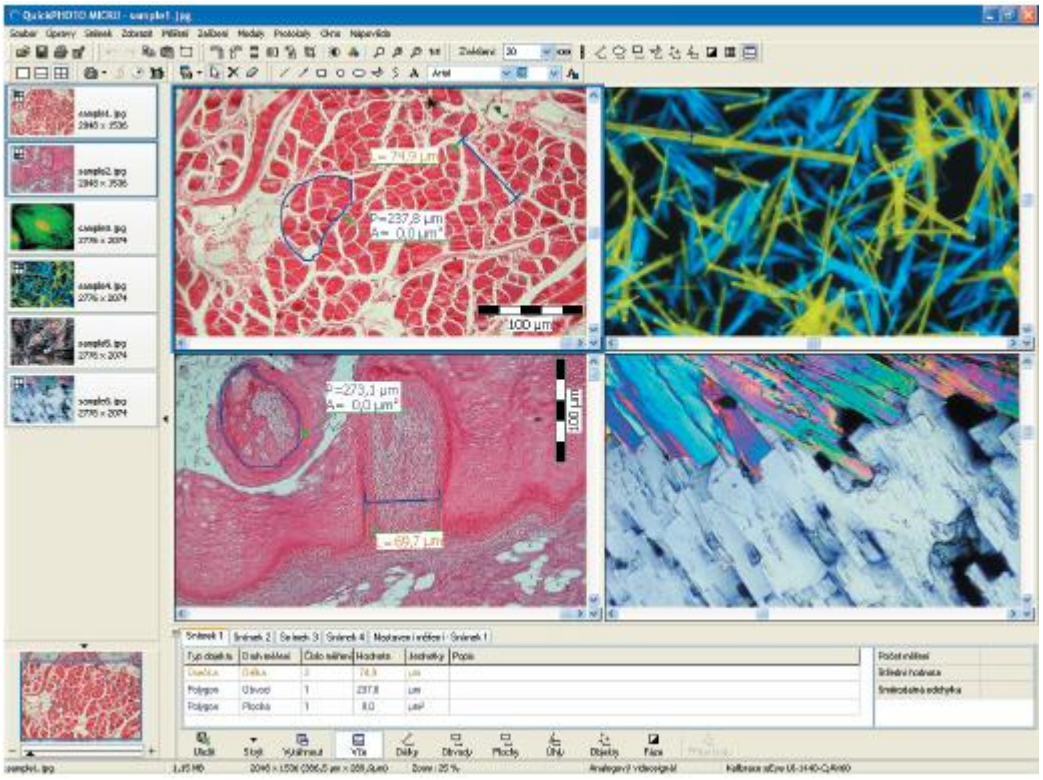
médium – kartu. Rozhodujícím faktorem ovlivňujícím kvalitu pořízených mikroskopických fotografií je kvalita snímacího systému. Tu můžeme zvýšit použitím jak kvalitní optiky mikroskopu, tak i digitálního fotoaparátu s rozlišením kolem 3 milionů pixelů a více.

Dále se nabízí možnost pořízení mikroskopických fotografií pomocí CCD kamery. Toto je spíše záležitost specializovaného mikroskopického pracoviště. Celý systém je poměrně nákladný, neboť vyžaduje kvalitní mikroskop, kameru a výkonný počítač. Kamery se připojují k mikroskopu pomocí speciálního závitu.

Je možné se setkat s tzv. digitálním mikroskopem, který umožňuje přenos sledovaných preparátů do počítače. Většina těchto mikroskopů vznikla rozšířením stávajících o integrovanou kameru.

Při nákupu mikroskopu, který umožňuje pořizování mikroskopických fotografií, bývá součástí balení i program na zpracování pořízených fotografií. Příkladem takového programu může být např. QuickPHOTO MICRO od Promicry, viz obr. 8.

Obecně programy slouží k vlastnímu snímání, archivování a zpracovávání získaných dat. Živý obraz na monitoru PC z mikroskopu dále usnadňuje ostření a správné exponování snímku, jako je úprava intenzity pozadí, jas, kontrast, sytost barev. Programy dále bývají vybaveny řadou funkcí, jako je měření délek, obvodů, ploch, úhlů, počítání objektů, analýza fází, měření intenzity fluorescence a barev, přidávání textové dokumentace k obrazům. Software disponují i funkcí pro časosběrné fotografování, což umožňuje pořizovat snímky v předem definovaném časovém intervalu. Programy rovněž umožňují pracovat s více snímky současně, pro snadné porovnávání se může zobrazovat vícero snímků vedle sebe.



Obr. č. 8: Ukázka software pro zpracování mikroskopických fotografií QuickPHOTO od Promicry [17]

3 Zpracování obrazu

Vlastní zpracování obrazu obvykle dělíme do několika kroků. Dělení není zcela jednoznačné a literatura se v tomto ohledu různí. Mezi základní kroky patří snímaní a digitalizace obrazu, předzpracování, segmentace obrazu, popis objektů, klasifikace. [4]

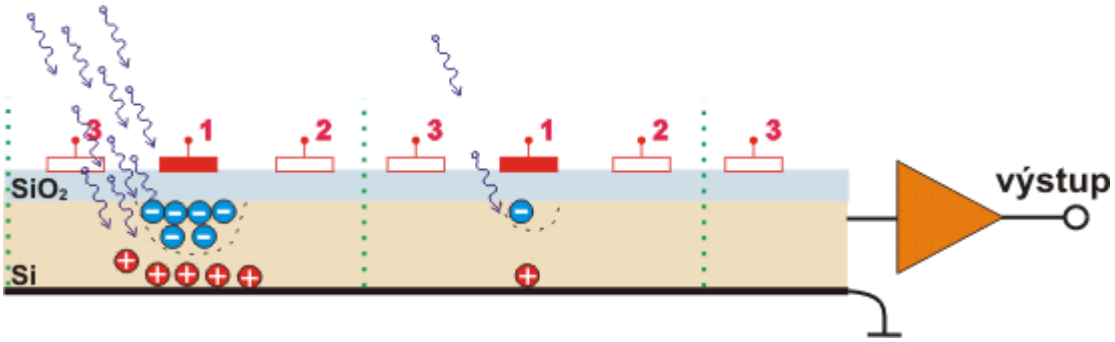
3.1 Snímání a digitalizace obrazu

Snímání obrazu je převod optické veličiny na elektrický signál. Při snímání obrazu je trojrozměrný objekt převeden na dvojrozměrný, což se provádí pomocí optické soustavy objektivu. Objekt je optickou soustavou promítán na obrazový snímač, který mění jas objektu v daném bodě na hodnotu elektrického napětí. V dnešní době se pro tento účel používají kamery s polovodičovými snímači, nejčastěji CCD prvky (charge coupled devices).

CCD prvky jsou tvořeny tzv. mozaikovou strukturou. V každém bodě mozaikové struktury je umístěn světločivný prvek, který dle počtu dopadajících fotonů vygeneruje množství elektrického náboje. Vygenerované množství elektrického náboje je úměrné výstupnímu elektrickému napětí. Mozaiková struktura je výhodná v tom, že obraz je touto strukturou rozdělen na pomyslnou matici obrazových bodů. Výstupní elektrický signál je tedy dán velikostí osvětlení daného obrazového bodu. Napětí na daném bodu se poté musí převést na číslicovou hodnotu, k čemuž slouží analogově-číslicový převodník.

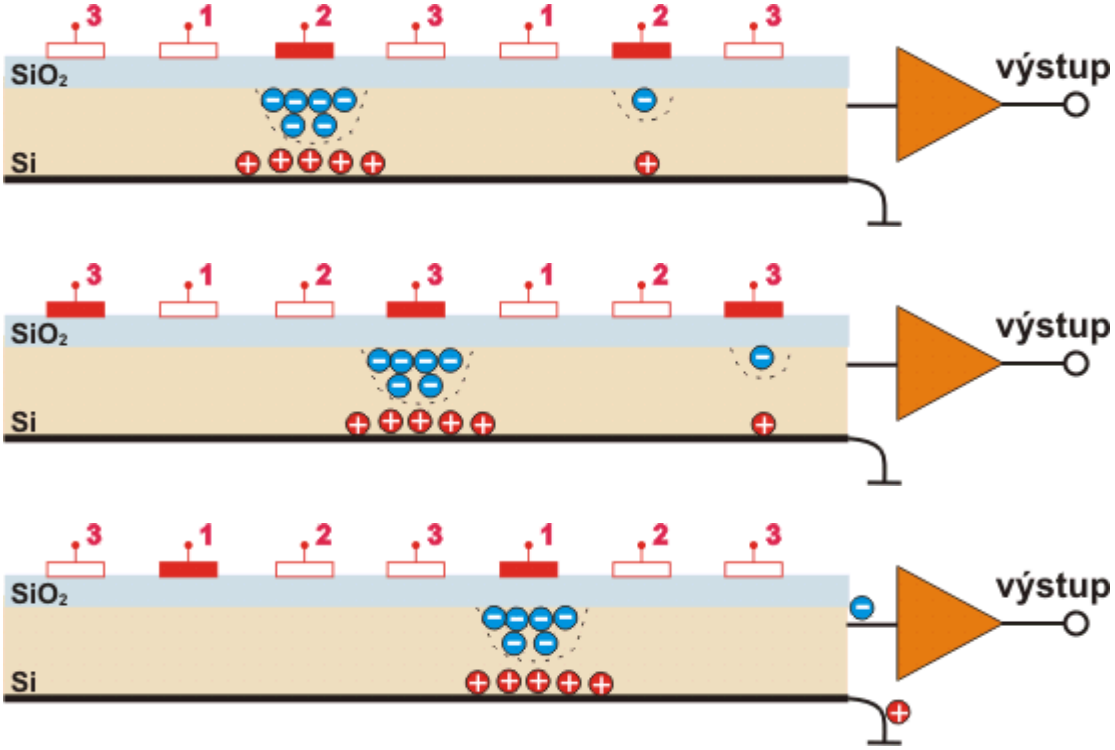
CCD využívá fyzikálního jevu známého jako fotoefekt. Fotoefekt spočívá v tom, že foton při nárazu do atomu dokáže převést některý z jeho elektronů ze základního do excitovaného stavu. V polovodiči se pak uvolněný elektron může podílet na elektrické vodivosti, tzn., že je ho možno z polovodiče odvést pomocí přiložených elektrod. U CCD je elektroda od polovodiče izolována tenkou vrstvou oxidu křemičitého SiO_2, který se chová jako dokonalý elektrický izolant, takže uvolněné elektrony nemohou být odvedeny pryč.

Činnost CCD lze rozdělit do 3 částí. První část je přípravná. Během této fáze jsou z CCD bez přístupu světla odebrány všechny volné elektrony, tzn., že je z čipu smazán jakýkoliv zbytek předchozího nasnímaného obrazu. Další fáze je fází expoziční. Na CCD prvek se nechá působit světlo. Dopadající fotony excitují v polovodiči elektrony a jsou přitahovány ke kladně nabitým elektrodám (na obr. 9 a 10 označeny číslem 1). Po elektronech zůstanou v polovodiči tzv. díry, které vykazují vůči svému okolí kladný náboj a jsou přitahovány elektrodou naspodu CCD. [10]



Obr. č. 9: Expozice obrazu [10]

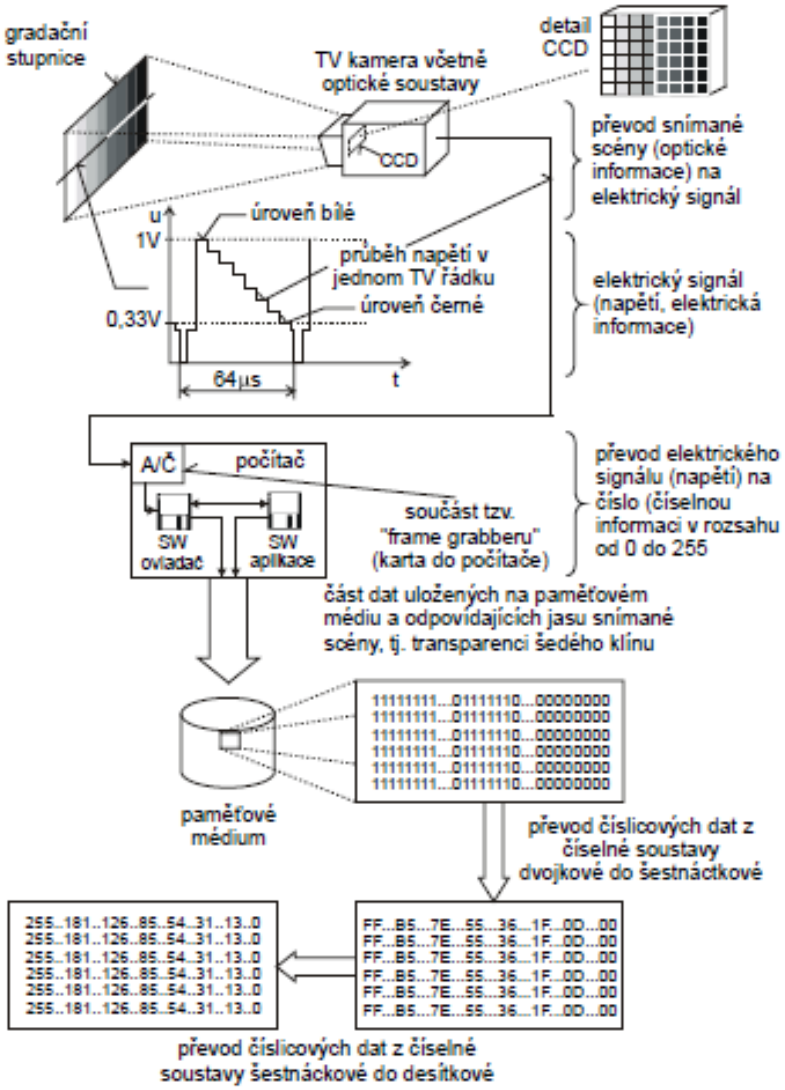
Poslední fází je snímání obrazu. Po uzavření závěrky (například u fotoaparátu) se na elektrody začne přivádět trojfázový hodinový signál (existují však i CCD prvky se čtyřfázovým nebo naopak dvoufázovým čtením). To znamená, že se na elektrodách 2 začne pozvolna zvyšovat napětí, zatímco na elektrodách 1 se snižuje. Tento celý děj poté opakovaně probíhá mezi elektrodami 2 a 3, dále mezi 3 a 1. Shluky elektronů z jednotlivých pixelů se posouvají přes sousední pixely směrem k výstupnímu zesilovači. Tento zesilovač pak už jen zesílí malý proud odpovídající počtu nachytaných elektronů v jednotlivých pixelech na napěťové úrovně vhodné pro další zpracování obrazu.



Obr. č. 10: Snímání obrazu [10]

Dalším nezbytným krokem ve zpracování obrazu je digitalizace, tedy už zmíněný převod analogového signálu na digitální pomocí analogově digitálního převodníku. Digitální obraz je pak ekvivalentem spojité obrazové funkce f(i,j), kde i, j jsou souřadnice v prostoru. Digitální obraz je získán pomocí vzorkování do matice bodů. Vzorkování se řídí Shanonovou větou, která říká, že nejmenší detail v digitálním obraze musí být minimálně dvojnásobkem vzorkovací frekvence. Volba vhodného vzorkování (rozlišení) v obraze je přitom zásadním krokem digitalizace. Při nízkém rozlišení by se

ztrácely informace o detailech v obraze, na druhé straně při velmi vysokém rozlišení by stoupala výpočetní složitost a náročnost dalšího zpracování obrazu. Velikost obrazu se udává v obrazových bodech – pixelech.



Obr. č. 11: Převod optické informace na číslicový obraz [6]

3.2 Předzpracování obrazu

Předzpracování je nezbytnou operací. Snímaný obraz může být kvůli vybranému způsobu snímání či nevhodným podmínkám při průběhu snímání zkreslen. Pokud máme informaci o charakteru zkreslení, dají se tyto chyby opravit korekcí, mezi něž může patřit úprava jasu, potlačení šumu, geometrické transformace, ostření. Předzpracování obrazu jsou základní postupy, které se musí provést, abychom mohli vyhodnocovat informaci, jež je obsažena v obraze.

3.2.1 Úprava jasu, kontrastu

K úpravě jasu, kontrastu či barevné stupnice se dají použít bodové operátory typu $g_{i,k} = \lambda(f_{i,k})$, kde λ je zvolená spojitá či diskrétní funkce. Bodové operátory jsou nejjednodušší prostředky na zpracování obrazu, i přesto se však jedná o účinný prostředek ke zvýrazňování obrazu.

Mezi základní typ operátoru patří globální transformace kontrastu, kde funkce λ nezávisí na poloze v obrazu. Existuje řada typických závislostí vyjadřujících tuto funkci, jejichž účelem zpravidla je zvýšení kontrastu v některé části stupnice šedi. Vzhledem k omezenému dynamickému rozsahu hodnot obrazových elementů taková úprava současně vede ke snížení kontrastu v jiné části stupnice, a proto bývá u obrazových procesorů možnost interaktivního upravování závislosti, které umožňuje prohlížet různé části obrazu postupně s různě nastaveným kontrastem. [8]

Zvýšení kontrastu šedotónového obrazu lze dosáhnout na základě následujícího principu. Všechny dostupné odstíny šedi ve výstupním obraze musí být zastoupeny se stejnými četnostmi. Jedná se o tzv. ekvalizaci histogramu, kdy je žádoucí, aby intenzity jasu byly zastoupeny v co nejširším rozmezí a se stejnou četností. Při této úpravě nedochází ke zvyšování informačního obsahu obrazu. Histogram takového obrazu pak má vyjadřovat co nejrovnoměrnější zastoupení.

3.2.2 Potlačování šumu

K omezování šumu lze využít např. lokálních operátorů. Tyto operátory se uplatňují tam, kde šum postihuje všechny elementy obrazu stejnou měrou. Pro omezení šumu můžeme použít model šumu, který má nulovou střední hodnotu a pro jehož hodnoty v sousedních obrazových prvcích musí platit, že jsou nezávislé. Lepší poměr signálu k šumu pak získáme váhovaným průměrováním hodnot sousedících prvků. Průměrující operátory mají obvykle charakter dolnopropustných filtrů.

Jiný typ rušení spočívá v tom, že postihuje jen jednotlivé body v obraze, často se označuje jako šum typu pepř a sůl. Pro odstranění tohoto typu šumu je třeba detekovat chybný prvek. Na odstranění tohoto šumu se často používá mediánový filtr, který setřídí dle velikosti prvky vstupního obrazu pokryté maskou zvolené velikosti a za výstupní prvek pak dosadí prostřední hodnotu setříděné posloupnosti.

3.2.3 Ostření obrazu

Ostření obrazu je do jisté míry protikladnou operací ve srovnání s předchozí částí o potlačení šumu. Při potlačování šumu bylo žádoucí potlačit složky s vyššími frekvencemi, protože často ty jsou právě šumové. Při ostření obrazu je naopak žádoucí zvýšit podíl složek s vyššími frekvencemi, protože se zde předpokládá, že právě ony nesou informaci o hranách a detailech obrazu. Při zpracování obrazu je tedy třeba hledat kompromis: zaostřování je možné jen do té míry, aby nedošlo k neúnosnému zhoršení signálu k šumu. [8]

Hrany jsou dány vlastnostmi obrazového elementu a jeho okolí, jsou určeny náhlou změnou obrazové funkce f(x,y). Změnu funkce popisuje její gradient. První složkou gradientu je jeho velikost, druhá složka pak udává úhel, pod kterým hrana běží.

Při zaostření je možno využít diferenčních operátorů, jež kompenzují průměrující funkce zobrazovacího či komunikačního systému.

3.3 Segmentace

Segmentace je dalším důležitým krokem ve zpracovávání obrazu. Tato analýza vede k nalezení objektů v obraze. Cílem segmentace je rozdělení obrazu do jednotlivých oblastí, které mohou být považovány za homogenní. Typickou segmentací je prahování, dále pak můžeme segmentovat na základě hran v obraze, hranic objektů apod.

Prahování je bodová operace, která segmentuje obraz na základě jasu. Při prostém prahování se zobrazuje vše černě pod jistou mezí vstupní zadaného jasu a vše bíle nad hodnotou meze jasu. Tedy pixely, které mají jas větší nebo roven prahu, jsou označeny algoritmem jako 1 a pixely, jejichž jas je menší než zvolený práh, jsou označeny jako 0. Často se pro zvolení prahu využívá histogram.

3.4 Popis objektů a klasifikace

Při popisu obrazu můžeme postupovat dvěma základními způsoby. První je založen na kvantitativním přístupu, jenž zahrnuje popis objektů v obraze pomocí číselných charakteristik jako např. velikost objektu. Jinou možností je kvalitativní přístup, kdy se popis soustřeďuje na jednotlivé objekty, tvarové vlastnosti apod.

Klasifikace výsledků je závěrečná operace ve zpracování obrazu. Při klasifikaci třídíme do skupin podle společných příznaků. Pro takovéto třídění lze použít nástrojů, jako jsou umělé neuronové sítě, statistické metody, ale třeba i metody fuzzy logiky.

4 Cíle práce

Tato práce má při pojetí tématu mikroskopické snímky s vysokou hloubkou ostrosti snahu nejenom popsat teoretické možnosti, ale také zabývat se hledáním možných způsobů dosáhnutí vytvoření snímku s vysokou hloubkou ostrosti.

Prvním cílem práce je seznámit se s optickým mikroskopem a s postupem při pořizování mikroskopických fotografií. Tento cíl rovněž zahrnuje prostudování problematiky ostření při pořizování konkrétních snímků s různými rovinami ostrosti a problematiky následného obrazového zpracování této sady snímků za účelem dosažení výsledného snímku s extrémní hloubkou ostrosti.

Dále je cílem této práce prostudování možných algoritmů pro obrazové zpracování sady snímků, poté zpracování návrhu metody pro zvýšení hloubky ostrosti a konečně provedení programového řešení algoritmu s využitím prostředí Matlab k tomuto účelu.

V neposlední řadě je cílem této práce ověření funkčnosti systému na reálních datech. Funkčnost vytvořeného algoritmu bude ověřena pořízením sady mikroskopických snímků konkrétního objektu s různými rovinami ostrosti a aplikací vytvořeného algoritmu, který provede propojení jednotlivých snímků a následně vygeneruje výsledný snímek s extrémní hloubkou ostrosti. Posledním cílem práce je zkoumat a zabývat se možnostmi aplikace systému pro další praktické využití.

5 Ostrost

5.1 Ostrost obrazu

Častým problémem v různých odvětvích počítačového vidění, mikroskopie aj. je výběr ostré části obrazu z celkového snímku. Cílem je vybrat „dobrou, ostrou“ část snímku, která bude vhodná pro další zpracování.

Mikroskop nám dává dvourozměrný obraz, kdežto vzorek, který snímáme, je vždy trojrozměrný. Obraz, který pozorujeme v mikroskopu, tedy obsahuje i rozmazané části vzorku, tvořené světlem přicházejícím k detektoru z oblasti, jež se nachází mimo zaostřenou rovinu. Tyto části navíc překrývají ostrý obraz. Celkově je získaný obraz tvořen rozmazanými částmi až z 90 %. Tato skutečnost je dána hloubkou ostrosti mikroskopu. Hloubka ostrosti je přímo úměrná vlnové délce a nepřímo úměrná numerické apertuře, přičemž numerickou aperturu (NA) můžeme definovat jako

$$NA = n \cdot sin\theta, \tag{3}$$

kde

θ… je úhlová apertura,
n… index lomu imerzního média.

Pro získání trojrozměrného obrazu je třeba snímat vzorek po jednotlivých rovinách, na které se postupně zaostřuje. Celkový obraz je tedy tvořen posloupností 2D snímků.

Dopadající světlo na detektor vzniklé v jednom bodě snímaného obrazu lze popsat funkcí, která se označuje jako rozptylová funkce PSF (Point Spread Function). Tato funkce určuje, jak bude vypadat každý zobrazovaný bod vzorku. Jednotlivé body vzorku jsou tedy charakterizovány funkcí PSF, přičemž platí, že čím větší je PSF funkce, tím je horší rozlišení obrazu.

Pokud budeme považovat PSF za prostorově nezávislou, můžeme vztah mezi snímaným vzorkem, funkcí PSF a získaným obrazem popsat konvolucí. Vztah mezi původním snímkem f(x,y) a získaným snímkem $g_1(x,y)$, …, $g_n(x,y)$ můžeme vyjádřit jako

$$g_i(x,y) = (f * h_i)(x,y), i = 1, \dots, n, \tag{4}$$

kde

$h_i(x,y)$ … PSF rozptylová funkce.

V idealizovaném případě $h_i(x,y) = \delta(x,y)$ bychom dostali $g_i(x,y) = f(x,y)$. Ve skutečnosti však $h_i(x,y)$ má charakter neznámého dolnopropustného filtru.

5.2 Hloubka ostrosti

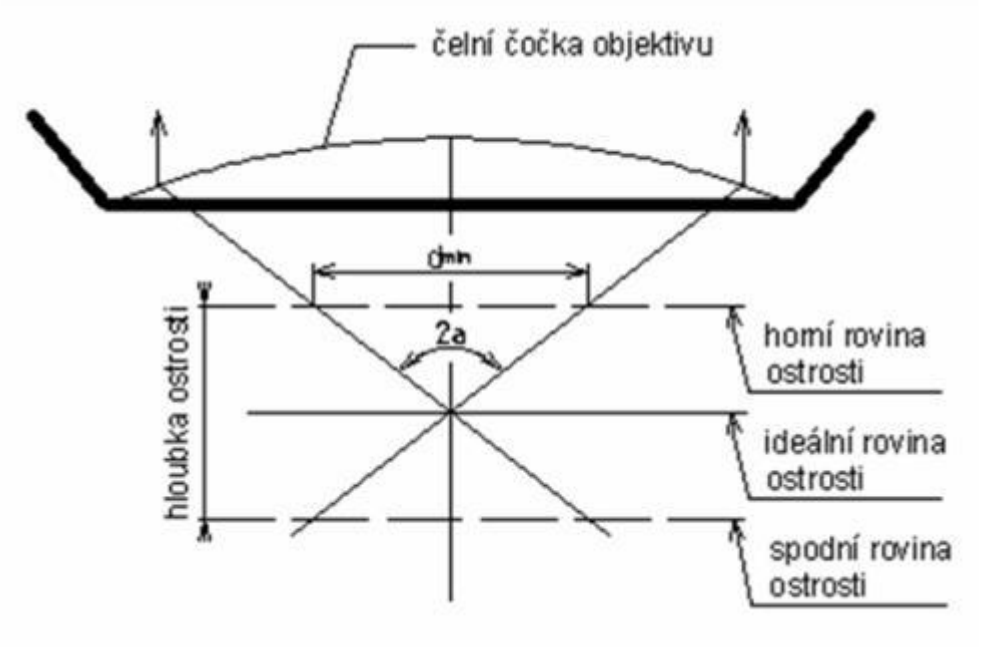
Hloubku ostrosti můžeme definovat jako hloubku prostoru v rovině předmětu, v níž lze obraz předmětu vidět jako ostrý. Je to vzdálenost mezi mezními rovinami, ve kterých je pozorovaný povrch ostře viditelný. Hloubka ostrosti závisí na rozlišitelné vzdálenosti d_{min} a na vstupním úhlu 2α použitého objektivu a je dána vztahem:

$$H = \frac{d_{\min}}{tg\alpha}, \tag{5}$$

kde

H … hloubka ostrosti.

Skutečně ostré objekty jsou pouze ty, které se nacházejí v rovině ostrosti. Čím je objekt blíže nebo dále od této roviny, tím jeho rozostření stoupá. Hloubka ostrosti je subjektivní rozsah, kde se nám zdají pozorované objekty ostré.

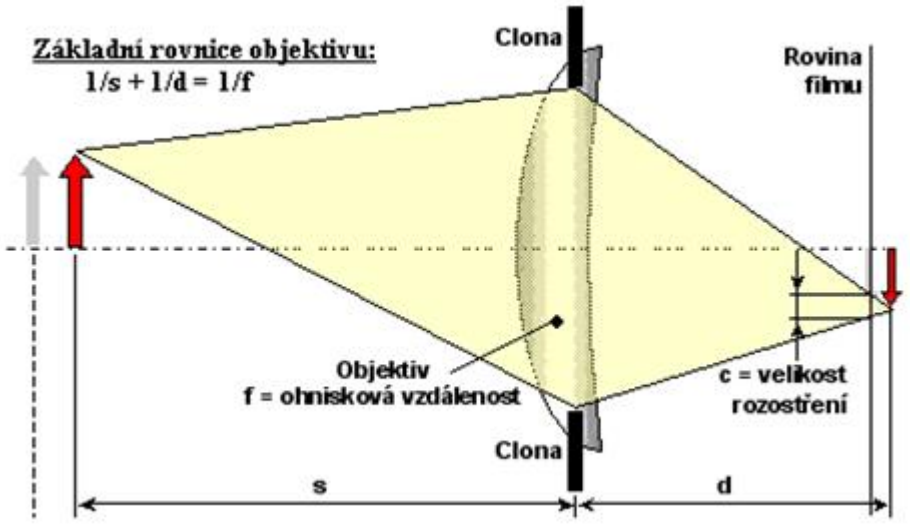


Obr. č. 12: Hloubka ostrosti [21]

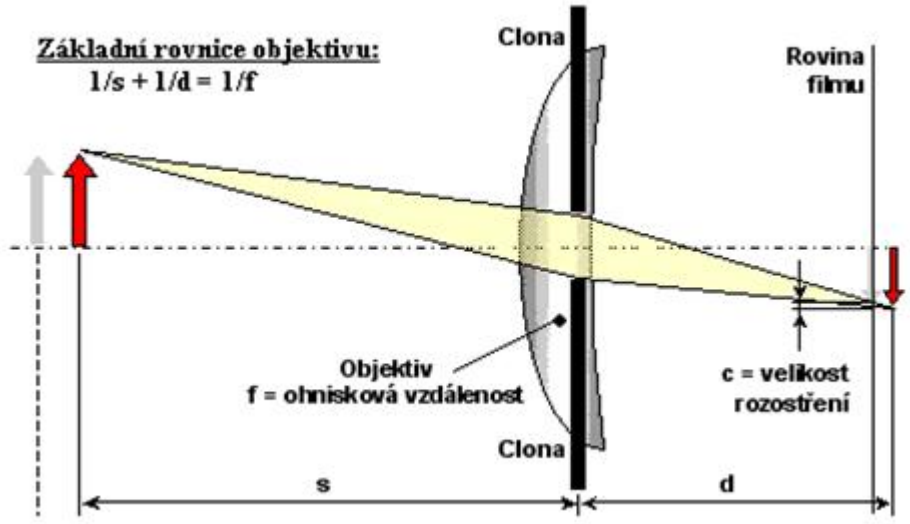
Hloubka ostrosti klesá s numerickou aperturou a se zvětšením objektivu. Méně zvětšující objektivy (4x, 10x) mají větší hloubku ostrosti oproti více zvětšujícím objektivům (40x, 60x, 100x). Hloubku ostrosti lze zvýšit cloněním nebo snížením postavení kondenzoru. Malá hloubka ostrosti tedy způsobuje, že v pozorovaném prostorovém útvaru (jakým je např. buňka) nelze zaostřit na všechny struktury najednou.

Hloubku ostrosti můžeme dále ovlivnit cloněním, přičemž při clonění nedochází k ovlivňování kompozice obrazu. Snížením clonového čísla (otevřením clony) se hloubka ostrosti snižuje, zatímco zvýšením clonového čísla (uzavřením clony) se hloubka ostrosti zvyšuje. Uzavřením clony se sice zvyšuje hloubka ostrosti, ale naopak se snižuje rozlišovací schopnost. Pokud je otevřená clona (viz obr. 13), dopadají

paprsky světla na detektor hodně rozevřené, čímž dochází k tomu, že i malá odchylka v zaostření vytvoří velké rozostření. Naopak při vyšším clonovém čísle (viz obr. 14) procházejí paprsky objektivem více sbíhavě, proto stejná odchylka v zaostření nezpůsobí v rovině detektoru tak velké rozostření.



Obr. č. 13: Ovlivnění hloubky ostrosti otevřenou clonou [14]



Obr. č. 14: Ovlivnění hloubky ostrosti uzavřenou clonou [14]

5.3 Metoda skládání obrazů

Mezi cíle této práce patří vysvětlení a uvedení metody, jak může vypadat realizace skládání dvou a více obrazů do sebe. Obrazy se od sebe liší rovinou ostrosti, tedy tím, že na každém obrazu je ostrá jiná část téhož pozorovaného objektu. Kvůli limitaci optických soustav není možné získat obraz, na kterém by byly všechny části objektu ostré. Proto vzniká technika skládání objektů, jejímž cílem je získat ze sady obrazů jeden výsledný obraz s vysokou hloubkou ostrosti. Tato technika se stává stále více

oblíbenou v mnoha odvětvích. Na trhu jsou k dispozici různé produkty zabývající se touto technikou, např. Modul Deep Focus od Promicry.

Základní myšlenkou techniky skládání objektů je výběr ostrých částí (pixelů) ze zdrojového obrazu a následně pak získání výsledného složeného obrazu. Tuto myšlenku lze realizovat vícero možnými principy. Centrem našeho zájmu může být např. prostorová frekvence, obrazový gradient či fázová koherence.

Základním bodem této techniky je skládání obrazů a v tom tkvící problém – jak vyhodnotit rozmazání obrazu a vybrat z něj tu část, která nese informaci o ostrosti.

Jednotlivé 2D obrazy označme jako $I_1(x,y)$, $I_2(x,y)$, $I_3(x,y)$, … $I_N(x,y)$ a tyto jednotlivé obrazy chceme spojit do jednoho výsledného f(x,y). Jednoduchou metodou by mohl být aritmetický součet vyjádřen jako:

$$f(x,y) = \frac{1}{N}\sum_{n=1}^{N} I_n(x,y). \tag{6}$$

Nevýhodou tohoto propojení je však to, že všechny informace obsažené v obraze jsou považovány za stejné (stejně důležité). Tímto způsobem by došlo k znehodnocení některých informací. K důležitým částem obrazu, které nesou „větší“ informaci, se musí přistupovat jinou cestou, např. váhovaným průměrem:

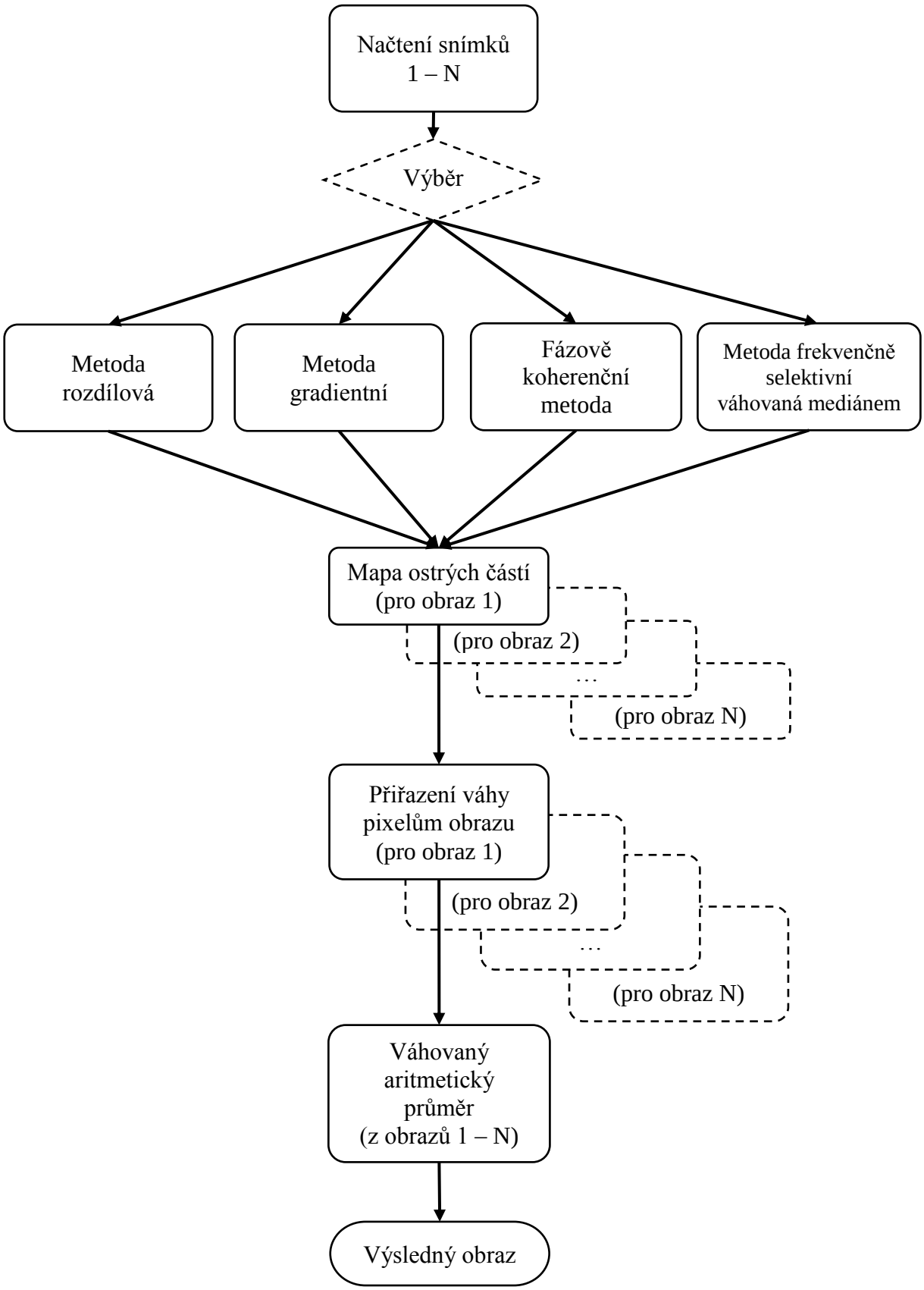
$$f(x,y) = \frac{\sum_{n=1}^{N} w_n(x,y) \cdot I_n(x,y)}{\sum_{n=1}^{N} w_n\,(x,y)}, \tag{7}$$

kde

$w_n(x,y)$ … váha obsahu informace na pozici (x,y) pro n-tý snímek.

A právě správná volba či správné vytvoření $w_n(x,y)$ bude důležité pro fúzi jednotlivých obrazů. Běžně se pro hodnocení ostré a rozostřené části obrazu používá prostorové frekvenční spektrum. Jak už bylo výše uvedeno, u složek s vyššími frekvencemi se předpokládá, že nesou informaci o hranách a detailech v obrazu.

5.3.1 Schéma metody



5.3.2 Možné metody při hledání míry ostrosti

Problém výběru lokálně nejlépe zaostřeného místa v obrazu ze série různě rozostřených (rozmazaných) obrazů jedné scény nachází četná uplatnění. Pozice ostrého místa v obrazu je pro každý pixel obrazu určena vyhledáním maximální míry ostrosti na okolích tohoto bodu. Pod pojmem míra ostrosti rozumíme funkcionál definovaný na prostoru obrazových funkcí, který kvantifikuje míru zaostření. Měla by být nezávislá na textuře objektu, jehož rozostřené obrazy popisuje, a monotónní vzhledem k rozostřování. Dále by měla být co nejvíce odolná vůči přítomnosti šumu. Většina měr ostrosti odpovídá filtrům zdůrazňujícím vysoké frekvence, protože rozostření má opačný efekt. [15]

Níže uvádím možné metody při hledání míry ostrosti (v následující části představuje I(x,y) obrazovou funkci):

- Rozdílová metoda – rozptyl jasových hodnot

$$S_{roz} = \frac{1}{M \times N} \sum_x \sum_y (I(x,y) - \mu)^2, \tag{8}$$

kde

μ … střední hodnota intenzity jasu pixelu je definována jako $\mu = \frac{1}{M \times N} \sum_x \sum_y I(x,y).$ (9)

Pro praktické použití je méně vhodná, protože při výpočtu míry ostrosti na překrývajících se obrazových oknech nelze využít již napočtené části, jelikož v každém okně je jiná střední hodnota.

- Gradientní metoda – energie obrazového gradientu

$$S_{grad} = \sum_x \sum_y (I_x^2 + I_y^2), \tag{10}$$

kde

I_x a I_y … představují obrazový gradient ve vodorovné a svislé poloze a jsou definovány jako $I_x = I(x+1,y) - I(x,y)$ a $I_y = I(x,y+1) - I(x,y).$ (11)

- Fázově koherenční metoda – v určité pozici x, y může být stanoveno jako

$$S_{fkm}(x,y) = \frac{1}{2}\sum_{\theta}|(h(x,y,\theta)\sin(\theta))^2 + (h(x,y,\theta)\cos(\theta))^2| + \frac{1}{2}\sqrt{\begin{matrix}4\left(\sum_{\theta}(h(x,y,\theta)\sin(\theta)h(x,y,\theta)\cos(\theta))\right)^2 \\ + \\ \left(\sum_{\theta}[(h(x,y,\theta)\cos(\theta))^2 - (h(x,y,\theta)\sin(\theta))^2]\right)\end{matrix}},$$

(12)

$$h(x,y,\theta) = \frac{\sum_n W(x,y,\theta)|A_n(x,y,\theta)\Delta\varphi_n(x,y,\theta)|}{\sum_n A_n(x,y,\theta) + \xi},$$

(13)

$$\Delta\varphi_n\,(x,y,\theta) = cos\big(\varphi_n\,(x,y,\theta) - \overline{\varphi_n}(x,y,\theta)\big) - \big|sin\big(\varphi_n\,(x,y,\theta) - \overline{\varphi_n}(x,y,\theta)\big)\big|,$$

(14)

kde

W … představuje frekvenci,
A_n a φ_n … představují amplitudu a fázi,
ξ … je malá konstanta, která se používá, aby nedošlo k dělení nulou.

- Metoda frekvenčně selektivní váhovaná mediánem – kdy míra ostrosti bude vyjádřena jako

$$S_{fsvm} = \sum_x \sum_y (I_x^2 + I_y^2),$$

(15)

kde

$$\begin{aligned} I_x = med\{I(x-1,y), I(x,y), I(x+1,y)\} \\ -\tfrac{1}{2}med\{I(x-3,y), I(x-2,y), I(x-1,y)\} \\ -\tfrac{1}{2}med\{I(x+1,y), I(x+2,y), I(x+3,y)\} \end{aligned}$$

(16)

$$I_y = med\{I(x,y-1), I(x,y), I(x,y+1)\} - \frac{1}{2}med\{I(x,y-3), I(x,y-2), I(x,y-1)\} - \frac{1}{2}med\{I(x,y+1), I(x,y+2), I(x,y+3)\} \quad (17)$$

- Metoda energie Laplaceova operátoru obrazu – velmi často používanou mírou ostrosti, kterou můžeme vyjádřit

$$S_{el} = \sum_x \sum_y (\nabla^2 I(x,y)^2, \quad (18)$$

kde

$\nabla^2 I(x,y)$ … představuje obrazový gradient získaný Laplaceovým operátorem následovně

$$\begin{pmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{pmatrix}.$$

6 Realizace algoritmu pro zvýšení hloubky ostrosti

6.1 Popis použitého uživatelského rozhraní

Tvorba aplikace na skládání obrazů probíhala v grafickém uživatelském rozhraní GUI (Graphical User Interface) v Matlabu, což je rozhraní sestavené z grafických komponent, jako jsou tlačítka, textová pole, posuvné seznamy, nabídky apod. GUI poskytuje rozhraní mezi uživatelem a aplikaci podřízeným kódem. Soustavu nástrojů pro tvorbu GUI poskytuje GUIDE (Graphical User Interface Development Environment). Tyto nástroje proces návrhu a programování GUI velmi zjednodušují. GUIDE poskytuje nejen sadu návrhových nástrojů, ale také generuje m-file odpovídající použitým grafickým prvkům, který obsahuje kód pro ovládání, inicializaci a spouštění GUI. Tento m-file poskytuje kostru pro implementaci callback funkcí (funkce, které se vykonají, pokud je uživatelé aktivují objekty v GUI). Při tvorbě v GUI je třeba vykonat tyto dvě základní úlohy:

- vybrat komponenty a uspořádat je v Návrhovém editoru GUIDE
- naprogramovat callback funkce použitých komponent

Použijeme-li příkaz guide v příkazovém řádku Matlabu, otevře se nám dialog GUIDE Quick start. Pokud použijeme vytvoření nového GUI a ponecháme přednastavenou volbu, otevře se návrhový editor. Tento editor umožňuje snadné a rychlé vytvoření GUI použitím jeho součástí, jako jsou tlačítka (buttons), vysouvací nabídka (popdown menu) nebo osy (axes).

Při uložení či spuštění GUI, GUIDE dochází k automatickému vytvoření dvou souborů se stejným jménem lišících se pouze příponou:

- *Figure-file* – soubor s příponou *.fig. Tento soubor obsahuje grafickou část, popis grafické části, úpravy tohoto souboru lze provádět v již zmiňovaném Návrhovém editoru GUIDE.
- *M-file* – soubor s příponou *.m. Soubor obsahuje zdrojový kód, při prvním spuštění/uložení GUI z Návrhového editoru GUIDE vygeneruje m-file s možností doplnění vlastního kódu pro každou callback funkci. Jednotlivé callback funkce je nutno naprogramovat v m-file editoru.

6.2 Popis zdrojového kódu

Při spuštění nového GUI, GUIDE dochází k vytvoření *figure-file* a *m-file* (viz výše). M-file obsahuje zdrojový kód (viz příloha). Ve figure-file tvoříme jednotlivé komponenty, které potřebujeme. Jedná se např. o *push button* (symbolizuje spouštěcí tlačítko, používáme jej k zahájení určeného výpočtu, požadovaného kroku), *pop-down menu* (po rozbalení nám nabídne možnosti, které jsme si nastavili), *check box* (zaškrtnutím pole volíme danou možnost). Pomocí *property inspektor*, který se nám zobrazí po dvojitém klepnutí na komponentu, můžeme nastavovat i další parametry komponenty, např. minimální zobrazovanou hodnotu, maximální zobrazovanou hodnotu, barvy, velikost písma, název komponenty atd.

V okamžiku, kdy jsme hotoví s rozmístěním grafických prvků, můžeme přistoupit k vytvoření zdrojového kódu jednotlivých komponent. Při kliknutí na tlačítko dané komponenty se zobrazí m-file editor s předvytvořeným zdrojovým kódem. Zdrojový kód se skládá z jednotlivých funkcí. V zásadě lze tyto funkce dělit na dva druhy – *Callback* a *CreateFcn*. *Callback* je funkce, která obsluhuje akci jednotlivých komponent, a *CreateFcn* daný objekt vytváří.

Do zdrojového kódu pak přidáváme jednotlivé příkazy, které nám zajistí požadovanou funkci a předávání dat mezi jednotlivými funkcemi.

Úvodní část zdrojového kódu se tvoří automaticky, do níž se s výjimkou nastavení *Inicialization handles* nesmí zasahovat. Při vytváření GUI se všechna data shromažďují v proměnné *handles* a nahrávání do této proměnné se děje pomocí funkce *guidata*. *Tag* označuje objekt, jedná se o vlastnost, která definuje uživatelské jméno objektu.

Pro názornost a uvedení vztahů mezi *figure-file* a *m-file* uvádím následující tabulku:

Typ	**String Description**	**Tag**
Panel	Vstupní fotky	iconpanel
Panel	VSTUP	
Panel	Zobraz vstupní foto v grayscale	
Panel	VÝSTUP	
Panel	STATUS	status
Push Button	Procházej…	Browse
Push Button	VSTUP	Read_Input_Images
Push Button	SPOJENÍ	Execute_Fusion
Push Button	DEMO	Multifocusdemo
Listbox		Browse_window
Pop-up menu	Zobraz foto	Show_Input_Image
Pop-up menu	Algoritmus	Algorithms
Axes		picture

Tabulka č. 1: Vztahy mezi *figure-file* a *m-file*.

6.2.1 Algoritmus gradientní metody

Algoritmus pro skládání načtených snímků pomocí gradientní metody. Na začátku je zde možné upravit velikost masky m,n, která se posouvá, a u načtených snímků se tak provádí výpočet pro všechny jejich pixely. Dále je zjišťována velikost snímku *(size)*, je nutné, aby všechny snímky, s nimiž je pracováno, měly stejnou velikost. Prostřednictvím příkazu *tic, toc* je měřen čas, jak dlouho bude trvat cyklus zpracování načtených snímků.

Data jsou upravena do tvaru s plovoucí řádovou čárkou a s dvojnásobnou přesností (double precision) pomocí příkazu *im2double*. Je to z důvodu rychlejšího výpočtu v Matlabu. Příkazem *padarray* a následujícího cyklu jsou ošetřeny kraje obrazu.

Dále je v uvedeném algoritmu počítána obrazová funkce dle gradientní metody v rámci masky (viz rovnice 10, 11) a tím také určena váha pro každý pixel. Tyto váhy jsou uloženy do matice stejného rozměru jako vstupní snímek, normovány pomocí nalezení největší hodnoty v matici vah (příkazu *imdivide*). Suma jednotlivých vah je provedena pomocí příkazu *imadd*, která bude potřebná pro výpočet výsledného obrazu. Výsledný obraz je tvořen z částí jednotlivých dílčích obrazů podle přiřazených vah (dle rovnice 7). Výsledný obraz se ukáže v hlavním okně a také se ukládá do stejného adresáře, z něhož byly zpracovávané obrazy načteny.

```
%**************************************************************************
%**************************************************************************
%**                ALGORITHM EMERGY OF IMAGE GRADIENT                    **
%**************************************************************************
%**************************************************************************
if handles.algorithm_index == 2     %Index = 2 --> Energy of Image Gradient
    %Mask iteractively applied for the gradient calculation
    m = 3;  %Size of neighborhood is 2m+1 x 2n+1
    n = 3;
    %Size of the input picture
    [rows cols] = size(handles.IM(1).IM);
    %Displays window with the waitbar
    w = waitbar(0, 'Please wait.......');
    tic;
    for i = 1:1:handles.inputpictures
        %All calculation in double
        AA = im2double(handles.IM(i).IM);
        %Pad array with [m n], default is 'replicate' and 'both'
        A = padarray(AA, [m n]);
        for j = 1:1:rows
            for k = 1:1:cols
                tempweight = 0;
                %Watch the edges to make sure index stays within the array
                if j == rows
                    jj = j + m - 1;
                else
                    jj = j + m;
                end
                if k == cols
                    kk = k + n - 1;
                else
                    kk = k + n;
                end
                %Applying the mask - Algorithm Energy of Image Gradient
                for r = -m:1:m
                    for s = -n:1:n
                        ix = A((jj + r + 1), (kk + s)) - A((jj + r), (kk +
s));
```

```
                        iy = A((jj + r), (kk + s + 1)) - A((jj + r), (kk +
s));
                        tempweight = tempweight + ix^2 + iy^2;
                    end
                end
                %Set the weight for every pixel
                weight(i).grad(j, k) = tempweight;
            end
            %Update waitbar for each picture
            waitbar(j / rows, w, ['Please wait...Processing Picture ',
int2str(i)]);
        end
        %Convert matrix of weights to the nomalized form
        norm = max(max(weight(i).grad));
        weight(i).grad_n = imdivide(weight(i).grad, norm);
    end
    %Close the window with the waitbar
    close(w);

    tElapsed = toc

    %Double array preparation with rows x cols size
    suma_weight(rows, cols) = 0;
    suma_weight_d = double(suma_weight);
    %Sum of weight needed for the pciture fusion
    for i = 1:1:handles.inputpictures
         suma_weight_d = imadd(suma_weight_d, weight(i).grad_n);
    end
    %Double array preparation with rows x cols size
    output_picture_temp(rows, cols) = 0;
    output_picture = double(output_picture_temp);
    %Calculation of the output picture
    for i = 1:1:handles.inputpictures
        image(i).IM = double(handles.IM(i).IM);
        output_picture = output_picture + imdivide(immultiply(image(i).IM,
weight(i).grad_n), suma_weight_d);
    end
    %Convert to integer
    output_picture8 = uint8(output_picture);
    %Create the output file and save into the working directory
    file = [handles.pathname, 'output', handles.extension];
    imwrite(output_picture8, file);
    %Display result picture
    imshow(output_picture8, 'Parent', handles.picture);
    %Selected algorithm used for the Fusion
    selected_algorithm = handles.algorithm_name{handles.algorithm_index};
    %Send message to Status field
    set(handles.status, 'String', ['The picture shows result of the '
selected_algorithm]);
```

6.2.2 Algoritmus metody frekvenčně selektivní váhované mediánem

Algoritmus pro skládání načtených snímků pomocí metody frekvenčně selektivní váhované mediánem. Na začátku je zde možné upravit velikost masky m,n, která se posouvá, a u načtených snímků se tak provádí výpočet pro všechny jejich pixely. Dále je zjišťována velikost snímku *(size)*, je nutné, aby všechny snímky, s nimiž je pracováno, měly stejnou velikost. Prostřednictvím příkazu *tic*, *toc* je měřen čas, jak dlouho bude trvat cyklus zpracování načtených snímků.

Data jsou upravena do tvaru s plovoucí řádovou čárkou a s dvojnásobnou přesností (double precision) pomocí příkazu *im2double*. Je to z důvodu rychlejšího

výpočtu v Matlabu. Příkazem *padarray* a následujícího cyklu jsou ošetřeny kraje obrazu.

Na rozdíl od gradientní metody je zde počítána obrazová funkce pomocí okolí daného pixelu (viz rovnice 15, 16, 17), z nichž tvořím medián. Tedy z daného okolí pixelu je vybrán medián a ten nese informaci o ostrosti pixelu v obraze. Dále je opět takto vytvořená váha uložena do matice stejného rozměru jako vstupní snímek, normována pomocí nalezení největší hodnoty v matici vah (příkazu *imdivide*). Suma jednotlivých vah je provedena pomocí příkazu *imadd*, která bude potřebná pro výpočet výsledného obrazu. Výsledný obraz je tvořen z částí jednotlivých dílčích obrazů podle přiřazených vah (dle rovnice 7). Výsledný obraz se ukáže v hlavním okně a také se ukládá do stejného adresáře, z něhož byly zpracovávané obrazy načteny.

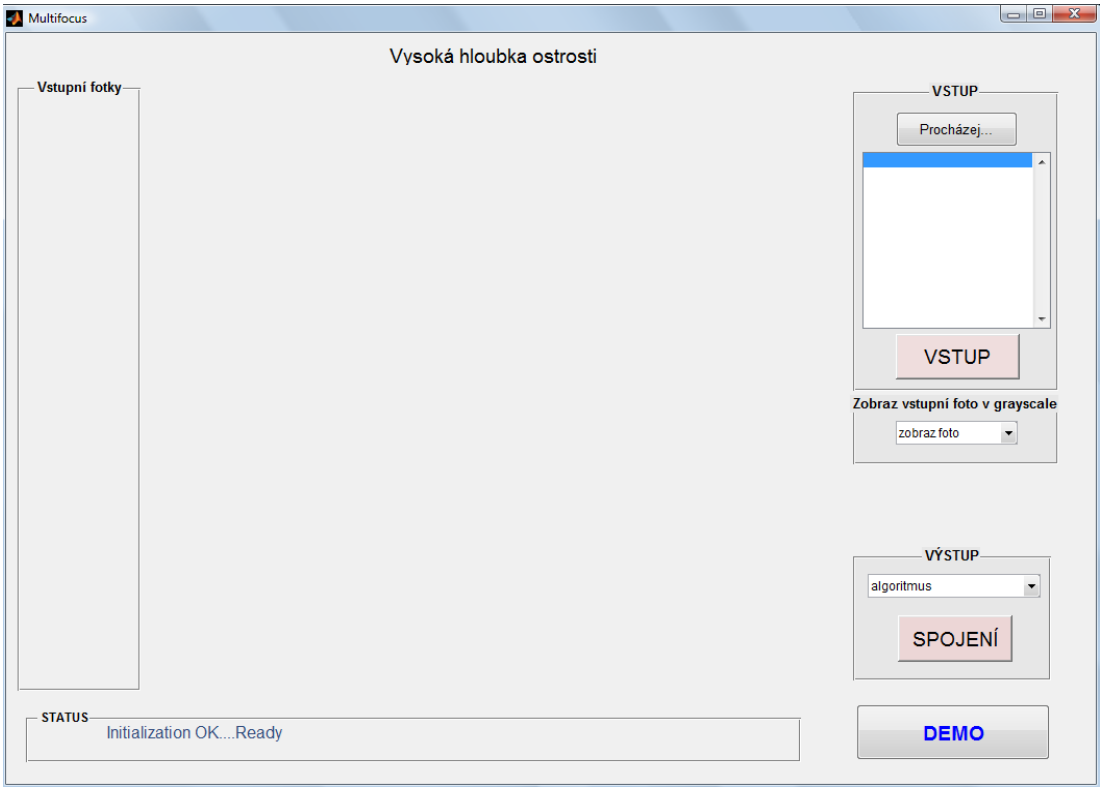
```
%************************************************************************
%************************************************************************
%**        ALGORITHM FREQUENCY SELECTIVE WEIGHTED MEDIAN FILTER          **
%************************************************************************
%************************************************************************
elseif handles.algorithm_index == 3     %Index = 3 --> Frequency selective
weighted median filter
    %%Mask iteractively applied for the gradient calculation
    m = 1;  %size of neighborhood is 2m+1 x 2n+1
    n = 1;  %
    %Size of the input picture
    [rows cols] = size(handles.IM(1).IM);
    %Displays window with the waitbar
    w = waitbar(0, 'Please wait....');
    tic;
    for i = 1:1:handles.inputpictures
        %All calculation in double
        AA = im2double(handles.IM(i).IM);
        %Pad array with [m n], default is 'replicate' and 'both'
        A = padarray(AA, [m+3 n+3]);
        for j = 1:1:rows
            for k = 1:1:cols
                %Watch the edges to make sure index stays within the array
                jj = j + m + 3;
                kk = k + n + 3;
                tempweight = 0;
                %Applying the mask - Algorithm Frequency selective weighted
median filter
                for r = -m:1:m
                    for s = -n:1:n
                        %Select the right pixels and sort them
                        r1 = [A((jj+r-1),(kk+s)) A((jj+r),(kk+s))
A((jj+r+1),(kk+s))];
                        r1_s = sort(r1);
                        r2 = [A((jj+r-3),(kk+s)) A((jj+r-2),(kk+s)) A((jj+r-
1),(kk+s))];
                        r2_s = sort(r2);
                        r3 = [A((jj+r+1),(kk+s)) A((jj+r+2),(kk+s))
A((jj+r+3),(kk+s))];
                        r3_s = sort(r3);
                        %Median is simpy the middle value of sorted array with
3 elements
                        ix = r1_s(m + 1) - r2_s(m + 1)/2 - r3_s(m + 1)/2;
                        c1 = [A((jj+r+1),(kk+s)) A((jj+r+2),(kk+s))
A((jj+r+3),(kk+s))];
                        c1_s = sort(c1);
                        c2 = [A((jj+r),(kk+s-3)) A((jj+r),(kk+s-2))
A((jj+r),(kk+s-1))];
                        c2_s = sort(c2);
                        c3 = [A((jj+r),(kk+s+1)) A((jj+r),(kk+s+2))
A((jj+r),(kk+s+3))];
```

```
                        c3_s = sort(c3);
                        %Median is simpy the middle value of sorted array with
3 elements
                        iy = c1_s(n + 1) - c2_s(n + 1)/2 - c3_s(n + 1)/2;
                        tempweight = tempweight + ix^2 + iy^2;
                    end
                end
                %Set the weight for every pixel
                weight(i).grad(j, k) = tempweight;
            end
            %Update waitbar for each picture
            waitbar(j / rows, w, ['Please wait...Processing Picture ',
int2str(i)]);
        end
        %Convert matrix of weights to the nomalized form
        norm = max(max(weight(i).grad));
        weight(i).grad_n = imdivide(weight(i).grad, norm);
    end
    %Close window with the waitbar
    close(w);

    tElapsed = toc

    %Double array preparation with rows x cols size
    suma_weight(rows, cols) = 0;
    suma_weight_d = double(suma_weight);
    %Sum of weight needed for the pciture fusion
    for i = 1:1:handles.inputpictures
         suma_weight_d = imadd(suma_weight_d, weight(i).grad_n);
    end
    %Double array preparation with rows x cols size
    output_picture_temp(rows, cols) = 0;
    output_picture = double(output_picture_temp);
    %Calculation of the output picture
    for i = 1:1:handles.inputpictures
        image(i).IM = double(handles.IM(i).IM);
        output_picture = output_picture + imdivide(immultiply(image(i).IM,
weight(i).grad_n), suma_weight_d);
    end
    %Convert to integer
    output_picture8 = uint8(output_picture);
    %create the output file and savs into the selected directory
    file = [handles.pathname, 'output', handles.extension];
    imwrite(output_picture8, file);
    %Display result picture
    imshow(output_picture8, 'Parent', handles.picture);
    %Selected algorithm used for the Fusion
    selected_algorithm = handles.algorithm_name{handles.algorithm_index};
    %Send message to Status field
    set(handles.status, 'String', ['The picture shows result of the '
selected_algorithm]);
```

6.3 Popis grafického prostředí

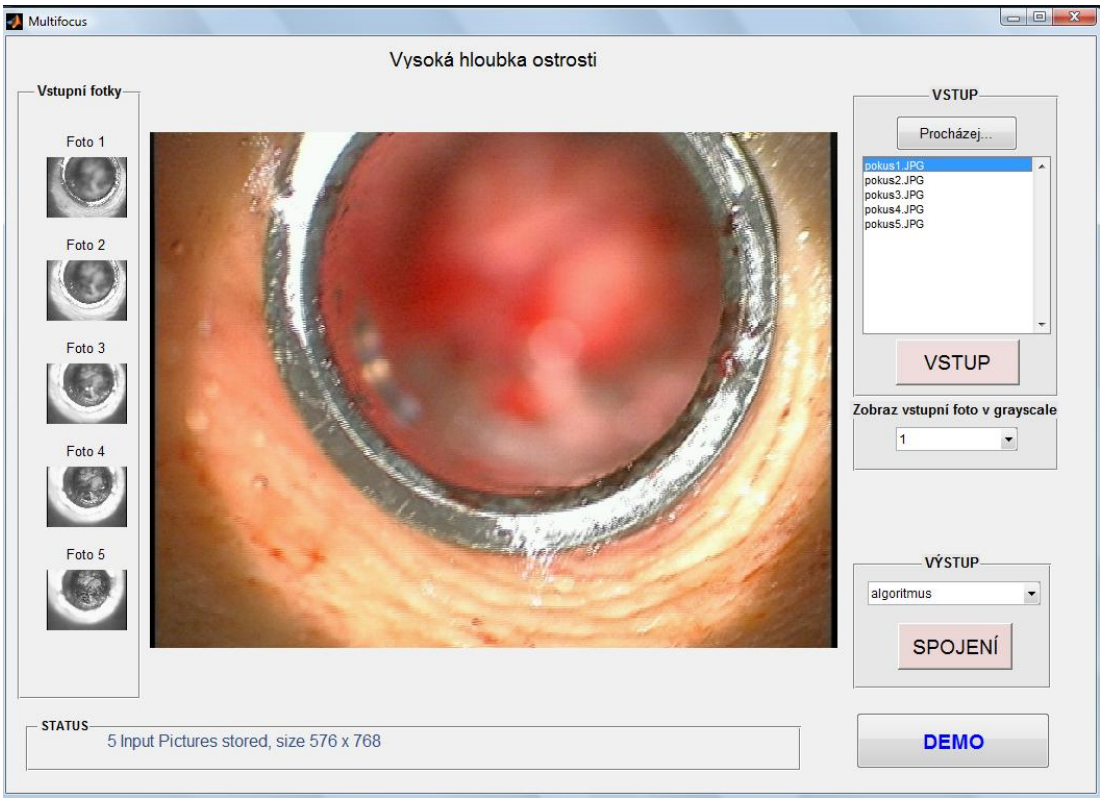
Spuštění figure-file s názvem Multifocus se dá provést následujícími způsoby. Do příkazového řádku v Matlabu lze napsat uvedený název souboru nebo se může do příkazového řádku napsat příkaz *guide*, kde je třeba vybrat záložku *Open existing GUI* a zde vybrat uvedený soubor.

Po otevření se nám ukáže následující okno (viz obr. 15). Vpravo se nacházejí jednotlivé komponenty, s jejichž pomocí se dá program ovládat.



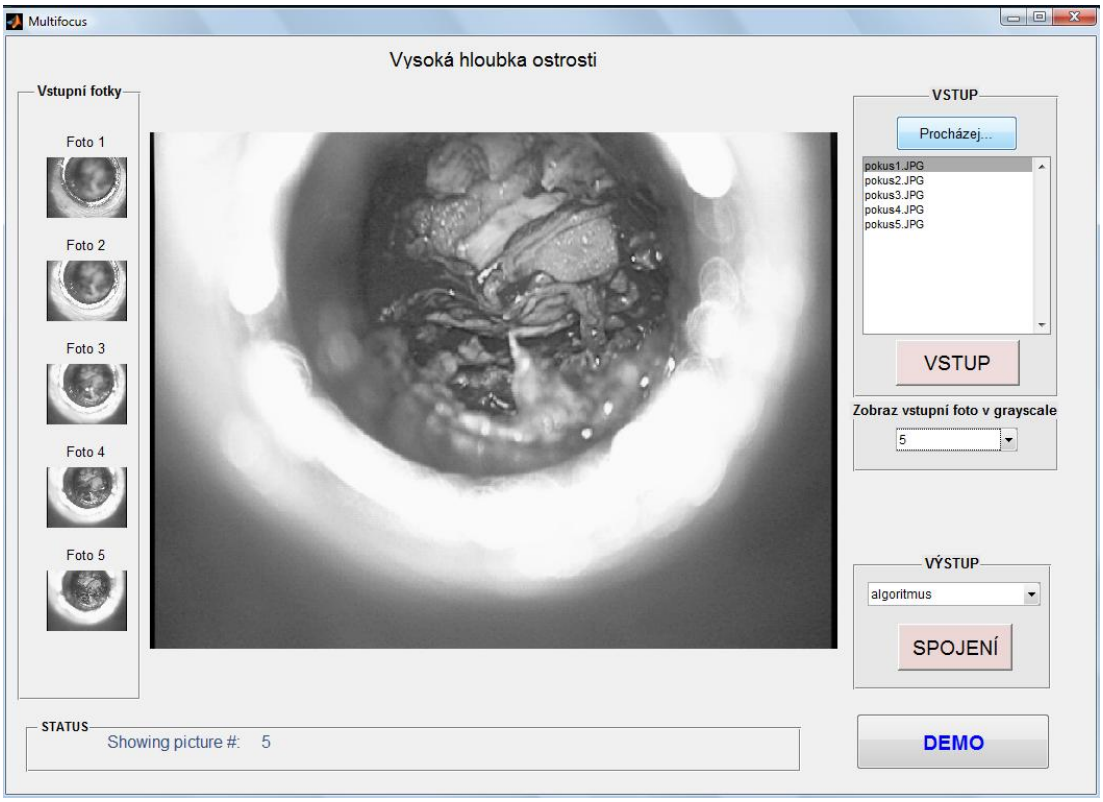
Obr. č. 15: Ukázka spuštěného figure-file Multifocus

Kliknutím na tlačítko *Procházej…* lze vybrat složku se snímky, které chceme dále zpracovávat. Snímky mohou mít libovolný název, ale názvy musí být indexovány. Syntaxe názvu je <jméno><x>, kde x je přirozené číslo od 1 do n. Dále snímky musí být v některém z následujících formátů: jpg, bmp, tif. Po stisku tlačítka *VSTUP* dochází k načtení snímků a jejich zobrazení v levém panelu *Vstupní fotky* ve formě malých ikon (viz obr. 16). Počet ikon je omezen na 5 z důvodu přehlednosti. Poklepnutím na seznam snímků v *listboxu* se zobrazí vybraný vstupní snímek v hlavním okně v originálním barevném provedení. Dole v poli *STATUS* se zobrazí informace o počtu načetných snímků a o jejich velikosti. V tomto poli se vždy objevuje informace o aktuální akci programu.



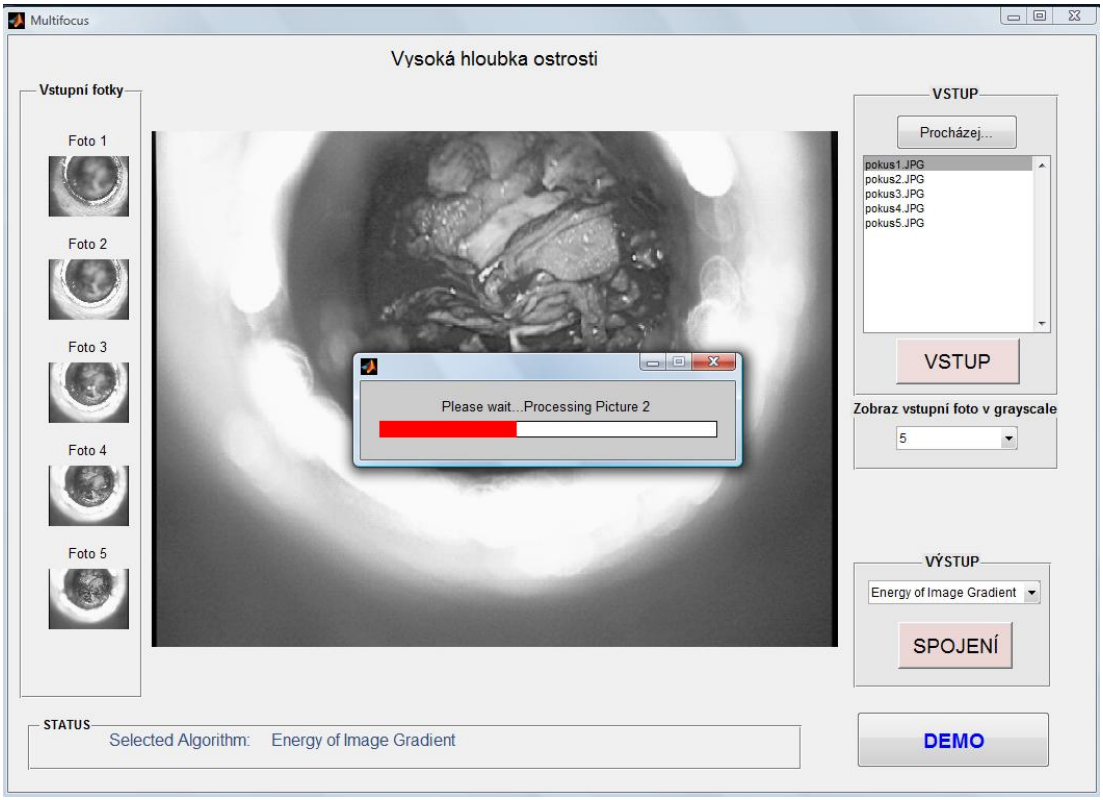
Obr. č. 16: Ukázka vložených snímků ve vytvořeném programu

Dále lze také zobrazit vybraný snímek ve stupních šedi. Rozkliknutím rozbalovacího menu *Zobraz vstupní foto v grayscale* se objeví čísla dle počtu načtených snímků. Pokud vybereme číslo námi požadovaného snímku, v hlavním okně se objeví vybraný snímek převedený na šedotónový (viz obr. 17).



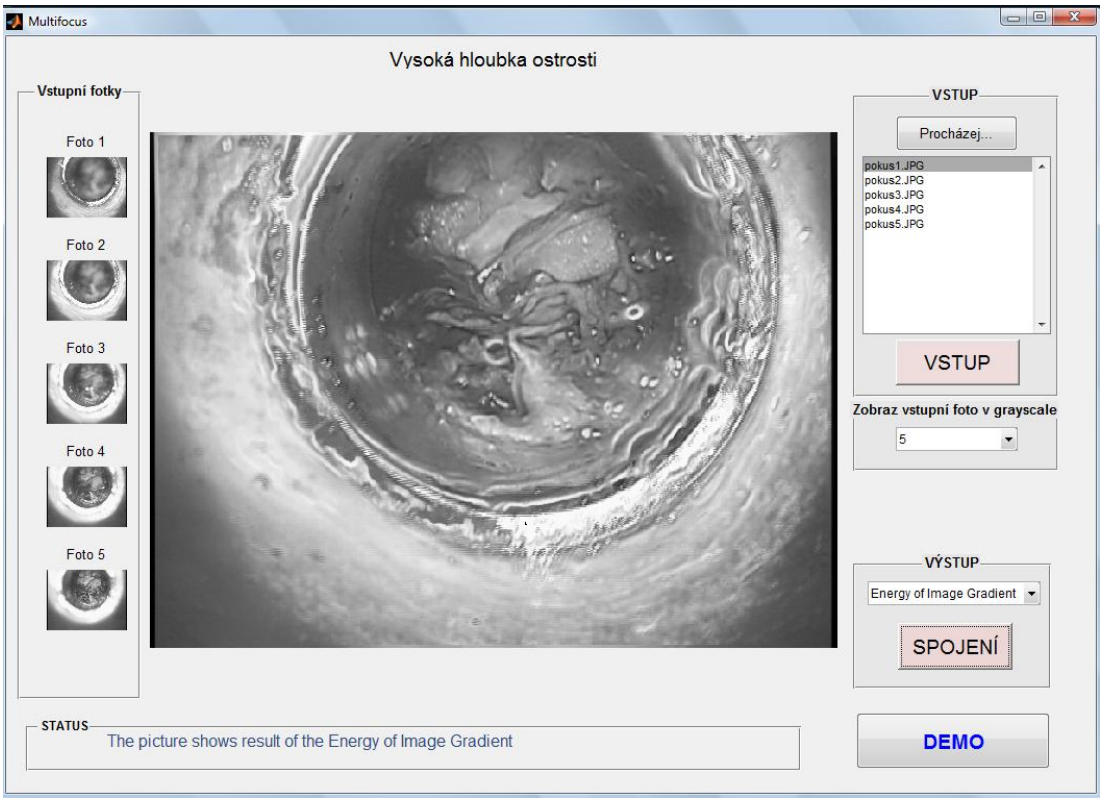
Obr. č. 17: Ukázka převedených snímků na šedotónové ve vytvořeném programu

Po rozkliknutí algoritmu k sekci *VÝSTUP* se objeví výběr dvou možností zpracování načtených snímků. Zde je nutné vybrat algoritmus, kterým chceme zpracovat snímky. Na výběr je metoda gradientní a metoda frekvenčně selektivní váhovaná mediánem. Po výběru metody dochází k postupnému zpracování snímků. Metoda první je rychlejší než druhá. O době, která bude nutná pro zpracování jednotlivých snímků, nás informuje *waitbar*. Z něj lze vyčíst, který obrázek se právě zpracovává a kolik z něj je již zpracováno (viz obr. 18). V příkazovém okně se ukáže celkový čas, který byl potřeba pro zpracování sady snímků.



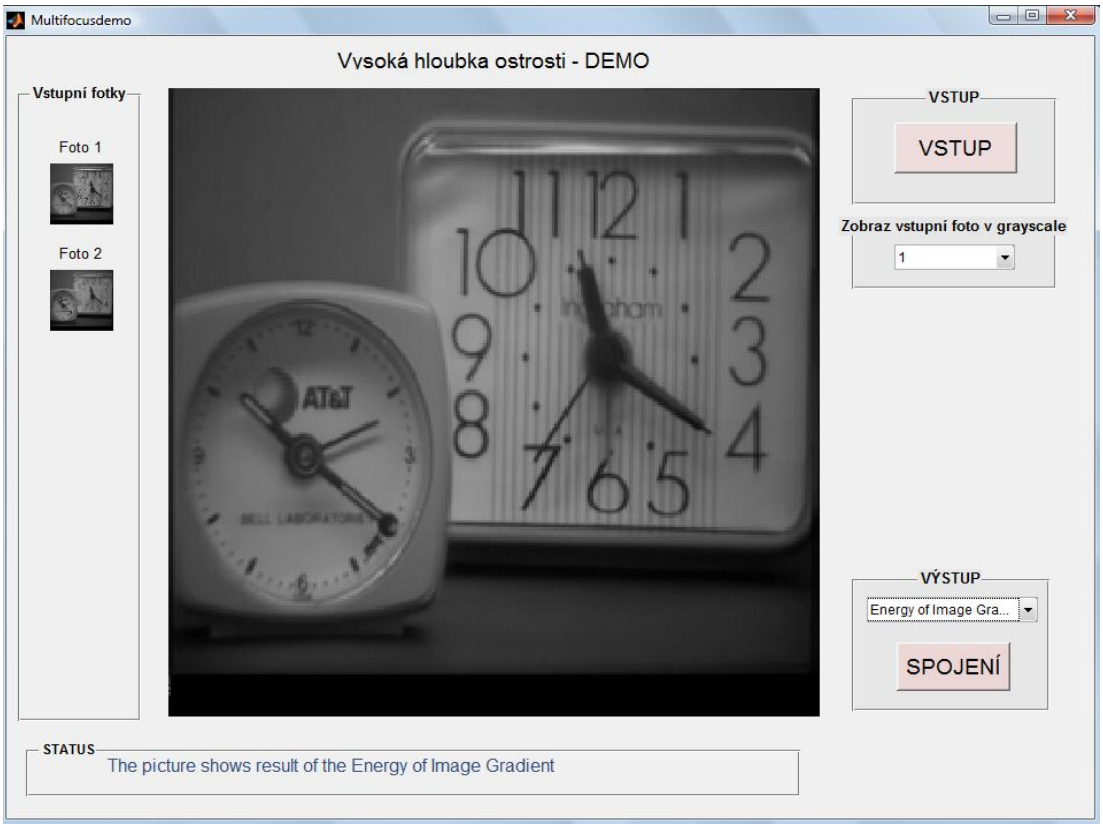
Obr. č. 18: Ukázka waitbaru ve vytvořeném programu

Výsledek se po ukončení zpracování ukáže v hlavním okně (viz obr. 19). Dále se také výsledek pod názvem *output* ukládá do stejného adresáře, z něhož byly zpracovávané snímky načteny.



Obr. č. 19: Ukázka výsledného obrazu ve vytvořeném programu

Poslední tlačítko DEMO po spuštění otevírá podobné, ale jednodušší okno, do něhož nelze snímky načítat (viz obr. 20). Jsou zde již předdefinovány dva snímky, které po jejich načtení tlačítkem *VSTUP* a složení vybraným algoritmem, demonstrují ve zjednodušené podobě funkci vytvořeného programu.



Obr. č. 20: Ukázka demoverze ve vytvořeném programu

7 Ověření systému

7.1 Snímky

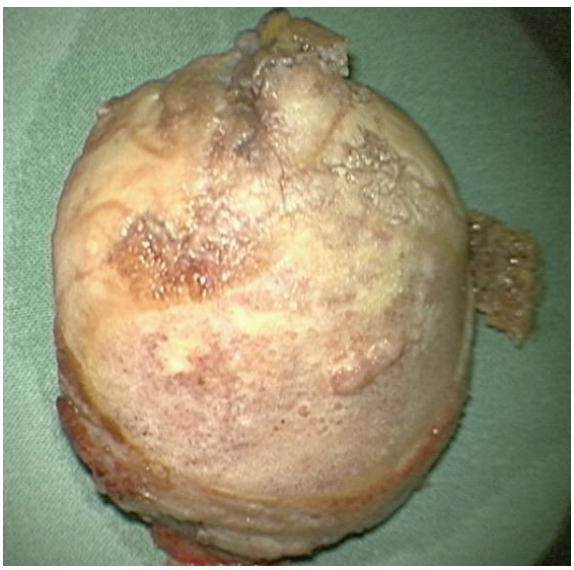
Cílem této práce bylo mimo jiné získat sady biologických snímků. Složením ostrých částí z jednotlivých snímků pak vytvořit jeden ostrý snímek s vysokou hloubkou ostrosti. Nejdříve jsou nasnímány jednotlivé snímky se standardní hloubkou ostrosti s různou rovinou zaostření. Na každém snímku je tedy ostrá jiná část objektu.

Následující snímky byly pořízeny operačním mikroskopem, vhodným pro neurochirurgické výkony, Opmi Neuro NC4 se záznamovým příslušenstvím Medilive Mindstream a Medilive Trio. Stereomikroskopem Nikon SMZ 1500, který je ve školní laboratoři, byly vyfotografovány vzorky čištěné kostní tkáně a hmyzu.

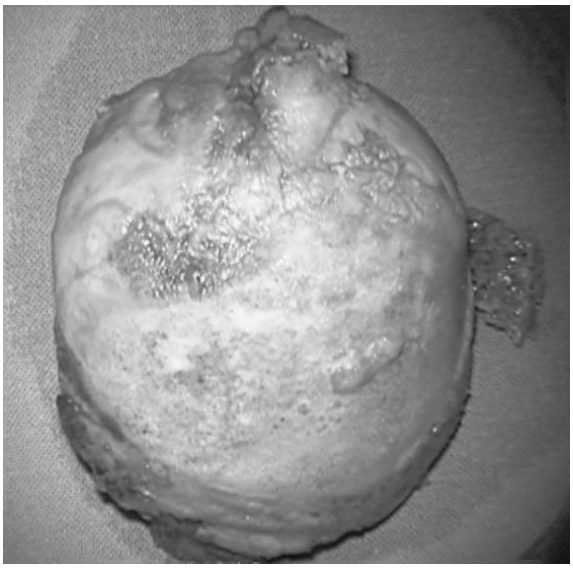
Po vyfotografování jednotlivých snímků je třeba snímky upravit tak, aby snímaný objekt byl na každém snímku přesně na stejné pozici. Při proostřování objektu (změna zoomu) dochází z fyzikálního hlediska ke změně ohniskové vzdálenosti objektivu. Současně se změnou ohniskové vzdálenosti se mění tři důležité a okamžitě viditelné veličiny - úhel záběru, zvětšení a hloubka ostrosti. Při snímání objektu tedy dochází ke změně úhlu záběru (a naší požadované změně hloubce ostrosti). V důsledku toho se bude snímaný objekt na jednotlivých snímcích lišících se rovinou ostrosti nacházet na jiném místě. Nejedná se sice o dramatické změny, přesto však je třeba objekt zachycený na snímcích centrovat, aby měl na každém snímku stejné umístění.

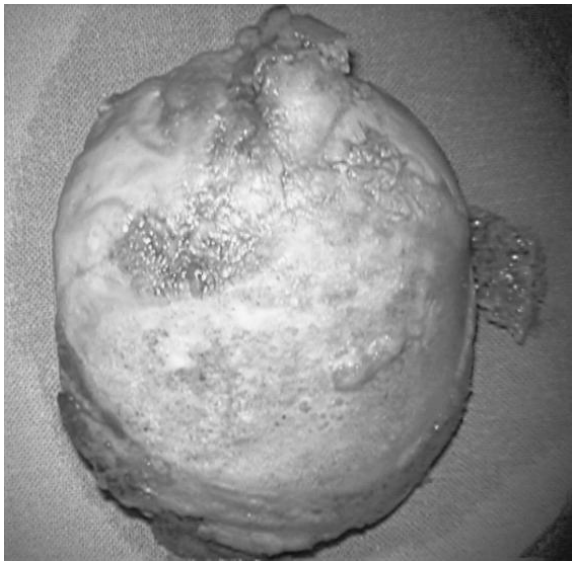
Nutností pro další zpracování tedy je, aby objekt zájmu byl na každém snímku na stejné pozici a aby všechny snímky měly stejné rozměry.

Následující snímky byly centrovány a ořezány ve volně dostupném programu GIMP 2.

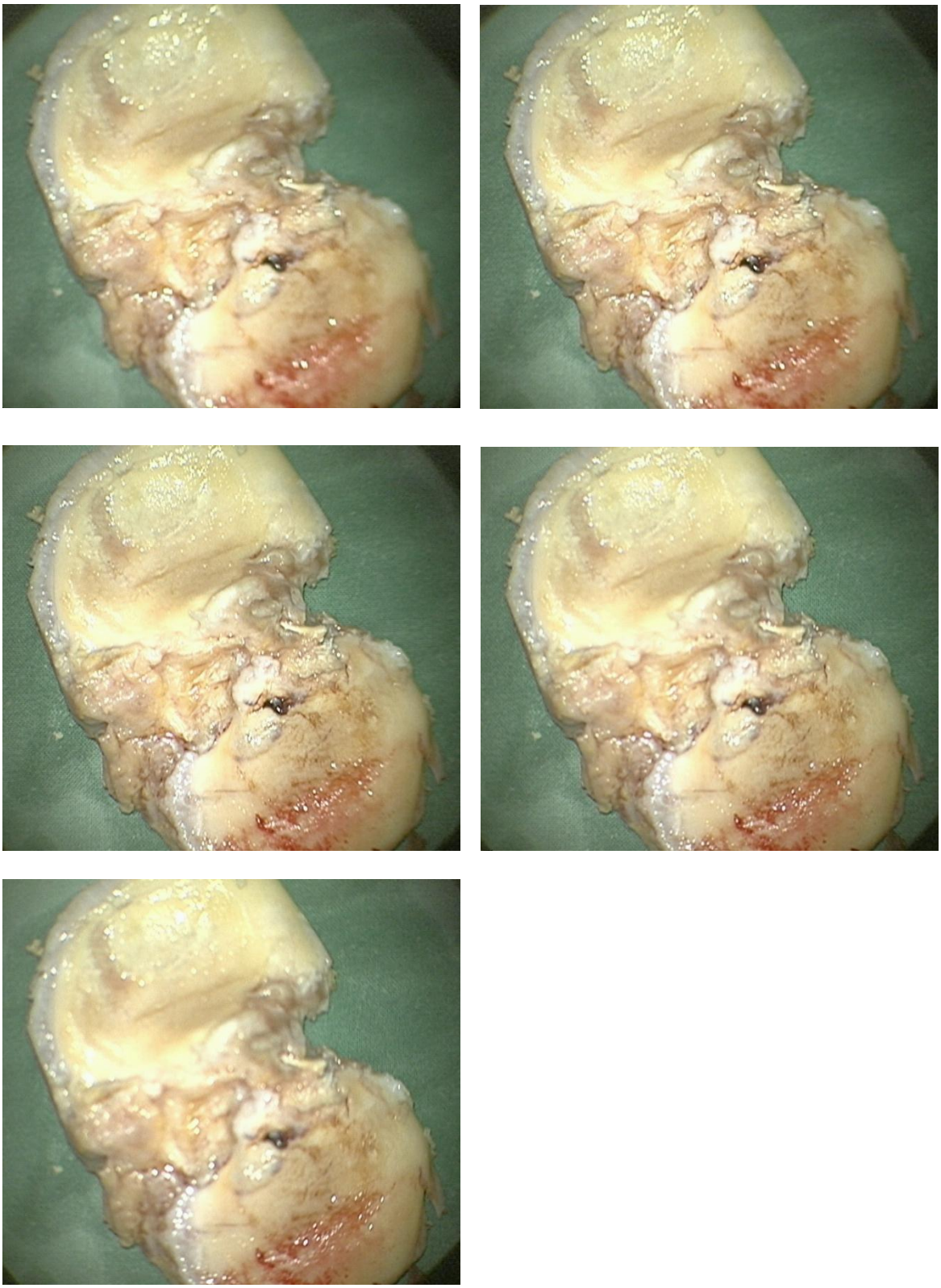


Obr. č. 21a, b, c: Snímky kloubní hlavice femuru, na snímcích lze pozorovat jednotlivé roviny ostrosti

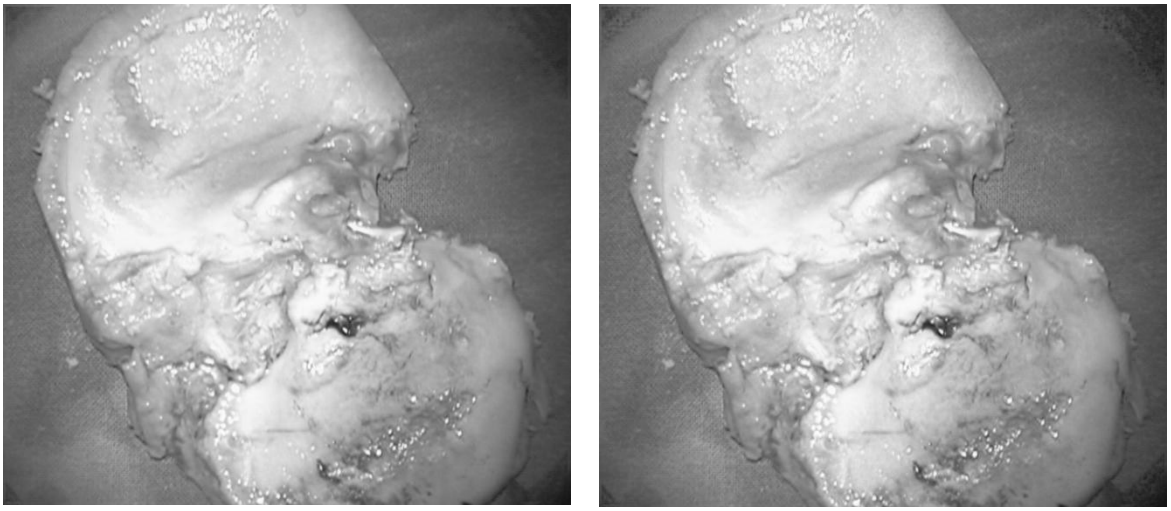




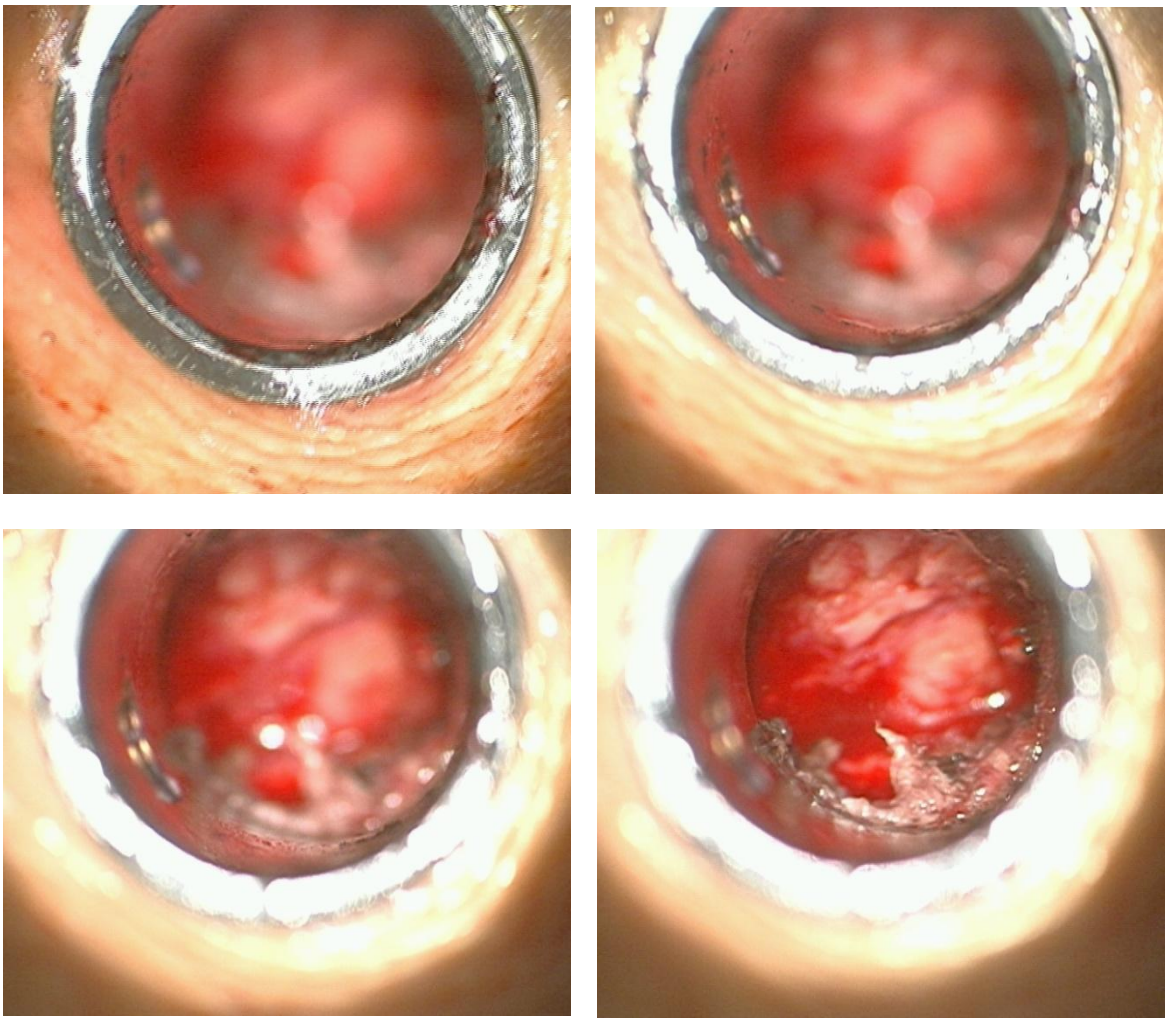
Obr. č. 22a, b: Výsledný snímek kloubní hlavice femuru, složen ze 3 řezů, vlevo složen metodou gradientní, vpravo složen metodou frekvenčně selektivní váhovanou mediánem

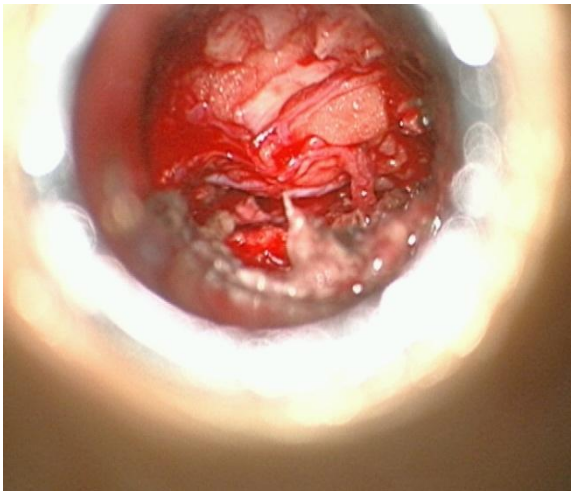


Obr. č. 23a, b, c, d, e: Snímky distálního konce femuru v řezu, na snímcích lze pozorovat jednotlivé roviny ostrosti

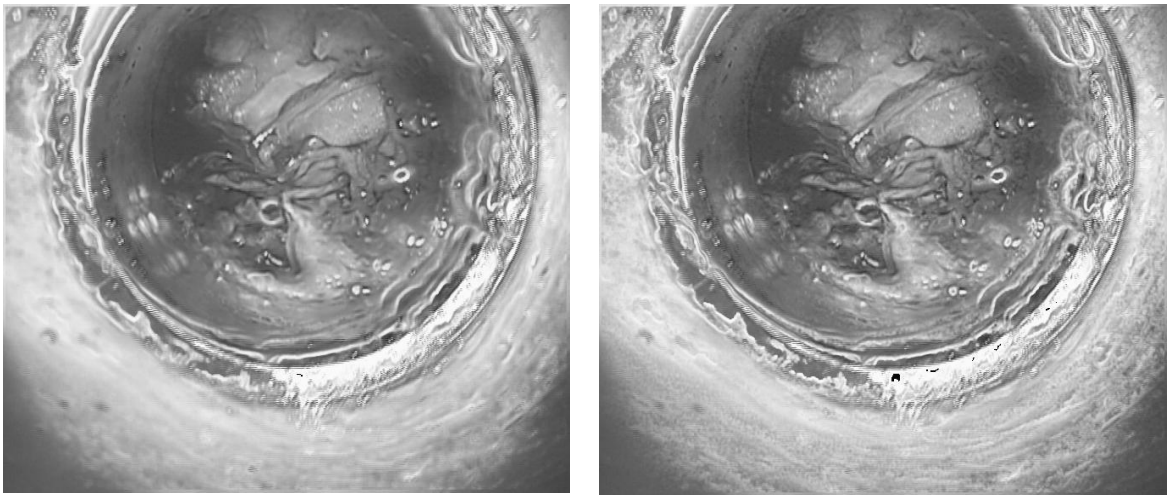


Obr. č. 24a, b: Výsledný snímek distálního konce femuru, složen z 5 řezů, vlevo složen metodou gradientní, vpravo složen metodou frekvenčně selektivní váhovanou mediánem

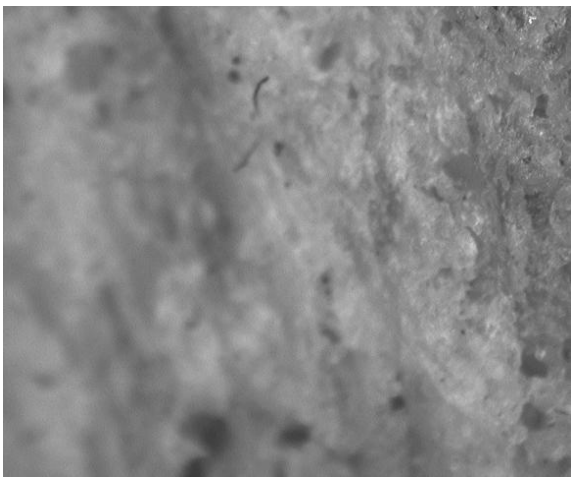
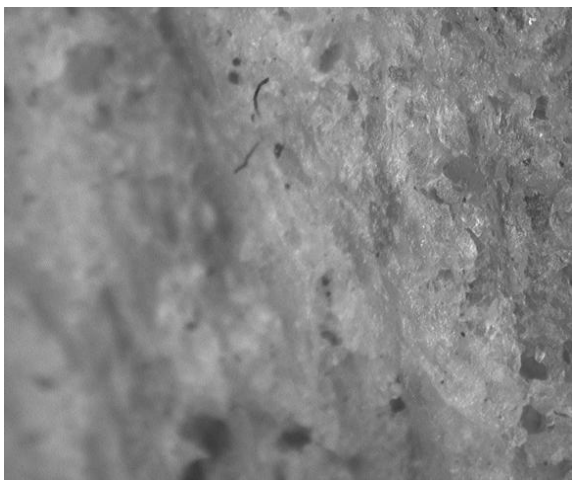
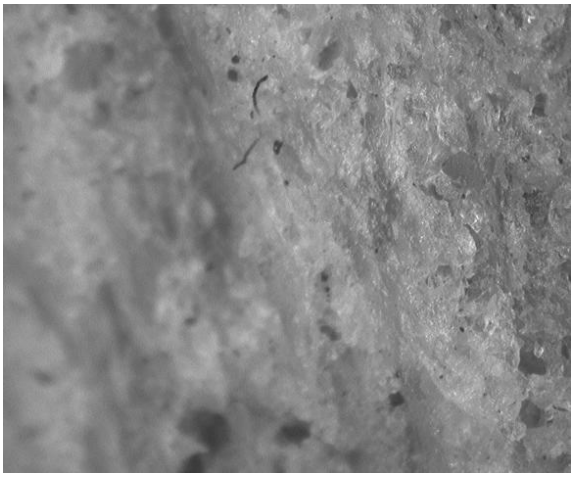
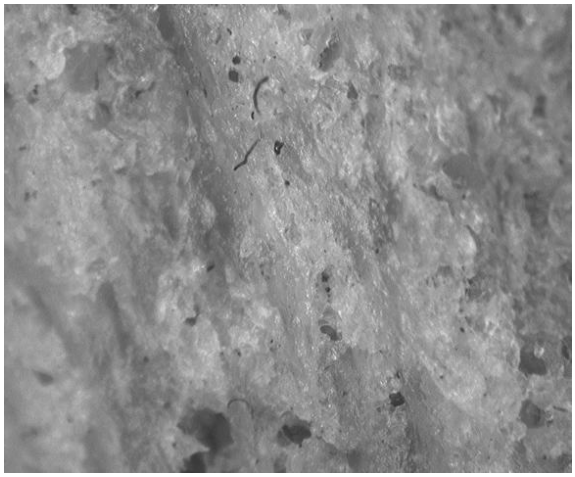
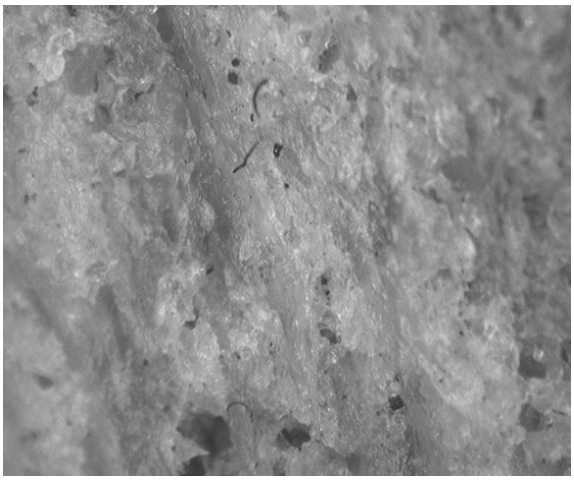
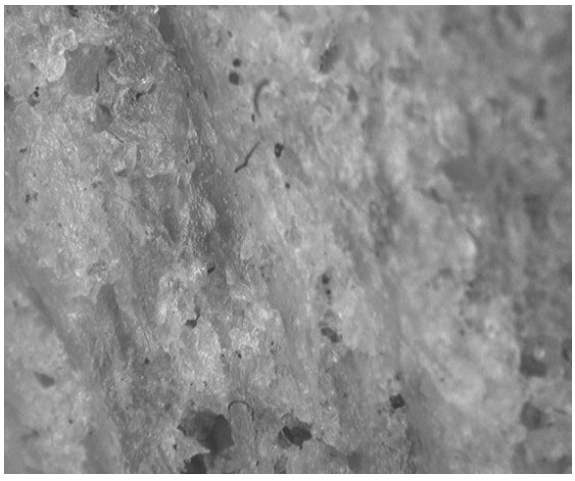


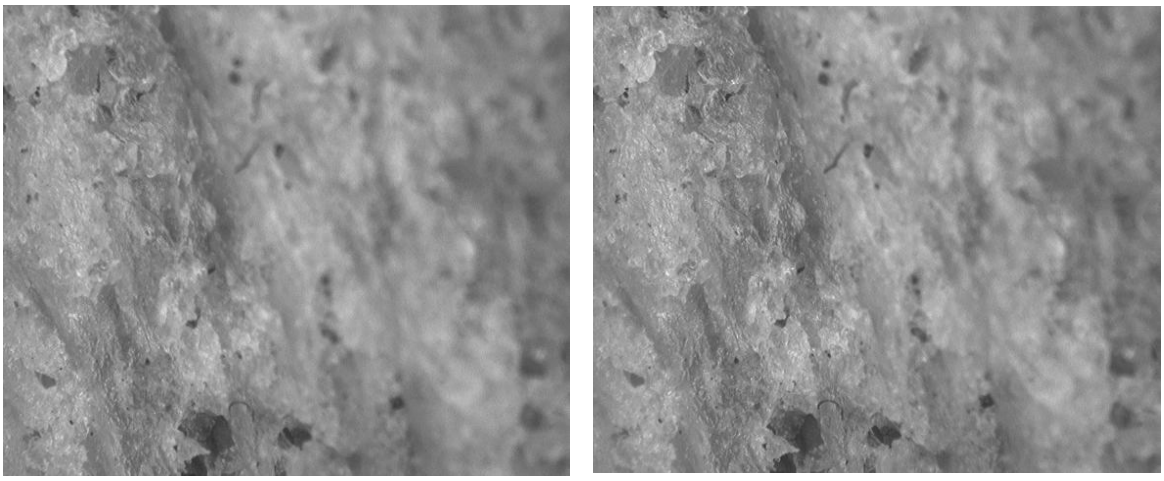


Obr. č. 25a, b, c, d, e: Snímky pořízené při mikroextirpaci hernie disci, na snímcích lze pozorovat jednotlivé roviny ostrosti

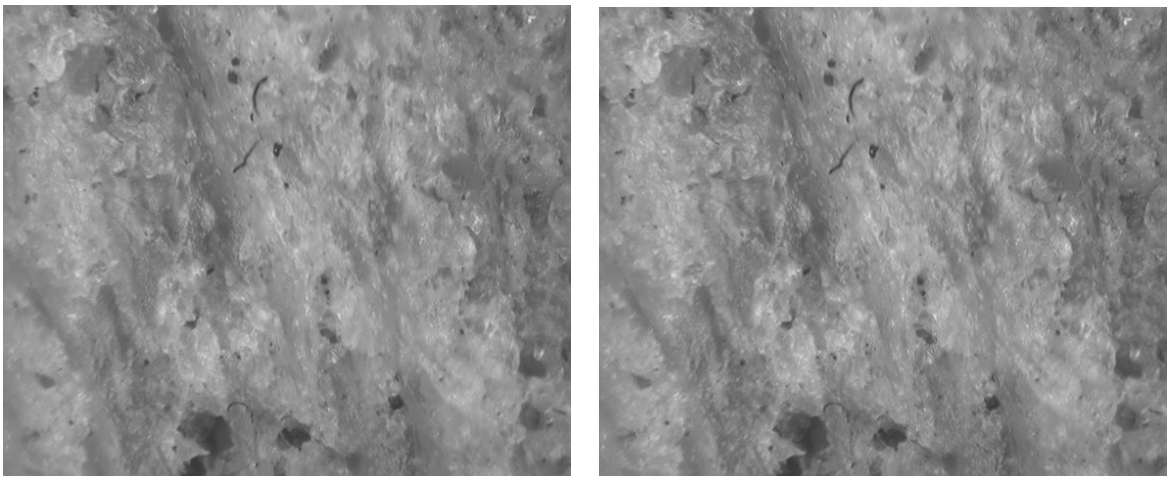


Obr. č. 26a, b: Výsledný snímek mikroextirpace hernie disci, složen z 5 řezů, vlevo složen metodou gradientní, vpravo složen metodou frekvenčně selektivní váhovanou mediánem

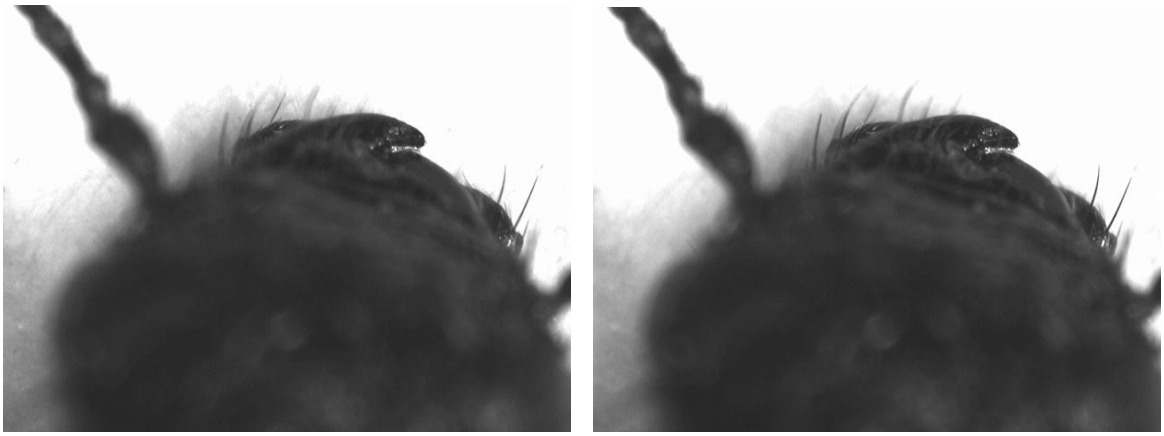







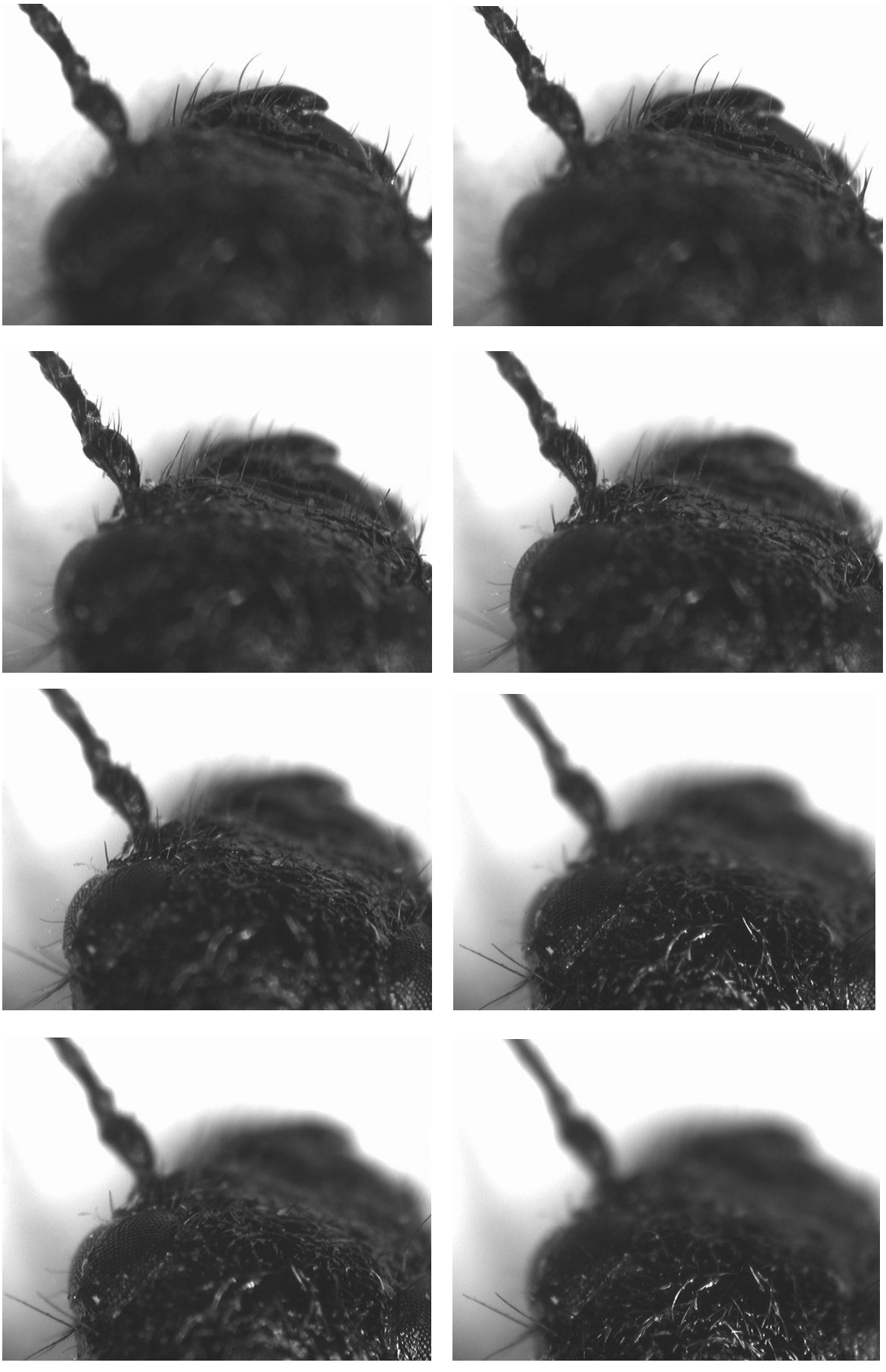


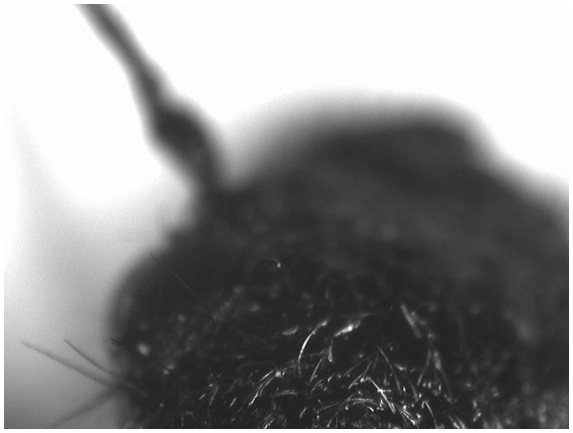
Obr. č. 27a, b, c, d, e, f, g, h: Snímky čištěné kostní tkáně, na snímcích lze pozorovat jednotlivé roviny ostrosti



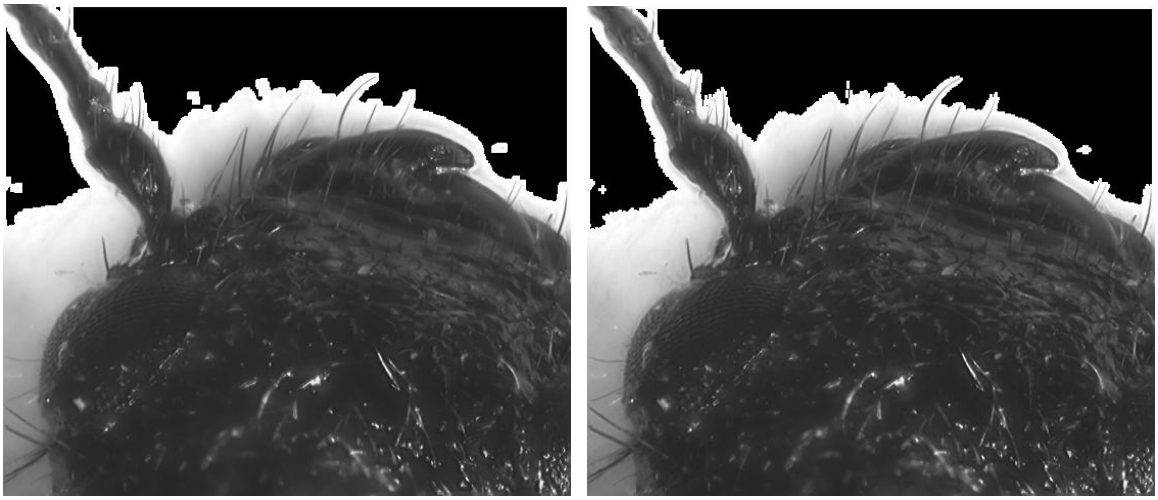
Obr. č. 28a, b: Výsledný snímek čištěné kostní tkáně, složen z 8 řezů, vlevo složen metodou gradientní, vpravo složen metodou frekvenčně selektivní váhovanou mediánem







Obr. č. 29a, b, c, d, e, f, g, h, i, j, k: Snímky hlavy tesaříka, na snímcích lze pozorovat jednotlivé roviny ostrosti



Obr. č. 30a, b: Výsledný snímek hlavy tesaříka, složen z 11 řezů, vlevo složen metodou gradientní, vpravo složen metodou frekvenčně selektivní váhovanou mediánem

Na příkladu hlavy tesaříka je vidět vliv oblasti se stejnou barvou ve všech vstupních obrazech (v našem případě bílá barva pozadí). Ve výsledném složeném obrazu se tato oblast zobrazí jako černá. Důvodem je skutečnost, že gradient takovéto oblasti se stejnou barvou je blízký nebo roven nule, a tedy jas ve výsledném obraze se rovněž rovná nule. Ve výsledku se zobrazí jako černá barva.

7.2 Aplikace vytvořeného systému

Mezi cíle této práce patřilo vytvoření systému na zvýšení hloubky ostrosti a dosažení tak snímku s extrémní hloubkou ostrosti. Tato technika umožňuje nový pohled na fotografované objekty a přináší téměř neomezenou hloubku ostrosti. Při fotografování mikroskopických předmětů je hloubka ostrosti velmi malá a při nárůstu zvětšení se ještě zmenšuje. Proto je problematické zaostřit na část objektu, kterou si přejeme pozorovat, fotografovat apod.

Z tohoto důvodu je velmi zajímavá možnost skládáných fotografií, kdy je možné tento problém s hloubkou ostrosti řešit. Výsledný obraz skládáné fotografie je tedy složen z dílčího počtu fotografií s různou hloubkou ostrosti.

Pro tuto práci jsem vytvořila různé sady mikroskopických biologických fotografií. Fotografie jsou uvedeny v předchozí kapitole. Problémem při pořizování snímků je, že se při přeostřování mění ohnisková vzdálenost, a je tudíž nutné snímky předupravit. Upravování snímků je náročné zvláště na čas a na přesnost. Objekt, který fotografujeme, musí být na všech snímcích ve stejné pozici, což stanovuji jako velkou nevýhodu tohoto řešení malé hloubky ostrosti. Nejedná se však o nevýhodu jen vytvořeného algoritmu, jde o společný problém všech komerčně dostupných programů na zvýšení hloubky ostrosti.

Máme-li však fotografie upravené, použití vytvořeného algoritmu je jednoduché a univerzální, dá se použít pro sady snímků různých formátů, velikostí. Dále může být systém využit jak pro fotografie mikroskopické, tak i makroskopické. Dílčí snímky pro další zpracování v uvedeném programu mohou být v následujících běžných formátech jpg, bmp, tif. Velikost snímků může být libovolná, podmínkou však je, že všechny snímky v sadě musí mít velikost stejnou. Nezáleží ani na tom, zda se jedná o fotografii barevnou či černobílou. V případě fotografií barevných program provede úpravu snímků na šedotónové a pak s takto upraveným snímkem pracuje.

Zajímavá aplikace systému je zvlášť pro mikrokopiské fotografie či makro fotografie nejrůznějších biologických objektů pro vědecké či publikační účely. Na upravených fotografiích lze pozorovat velmi jemné detaily fotografovaného objektu (viz fotografie hlavy tesaříka). Vytvořený systém však dokáže stejně dobře zpracovat i fotografie pořízené běžným fotoaparátem. Nemusí se tedy jednat jen o fotografie mikroskopické. Stále však zůstává požadavek na nutnost úpravy fotografií a posunutí fotografovaného objektu na fotografiích na stejné místo.

8 Závěr

Cíle, které byly stanoveny pro tuto práci, byly úspěšně dosaženy. V úvodu práce se zabývám optickým mikroskopem, možností pořizování mikroskopických fotografií, čtenáře seznamuji s možnostmi zpracování obrazu. Dále také rozebírám problematiku ostření, definici hloubky ostrosti a možnosti dosažení snímku s extrémní hloubkou ostrosti pomocí obrazového zpracování sady snímků s různými rovinami ostrosti.

V další části se zabývám možnými algoritmy pro obrazové zpracování sady snímků s různými rovinami ostrosti. Zde je rovněž uvedeneno schéma návrhu metody na zvýšení hloubky ostrosti. K programovému řešení algoritmu na zvýšení hloubky ostrosti jsem využila prostředí Matlab.

V kapitole 6 je popsána realizace výše uvedeného algoritmu. Pro řešení bylo využito grafické uživatelské prostředí GUI. Ukázky zdrojového kódu jsou uvedeny v popisované kapitole 6. Pro realizaci jsem si vybrala následující metody na hledání míry ostrosti – gradientní metoda a metoda frekvenčně selektivní váhovaná mediánem.

Posledním cílem této práce bylo ověření systému na reálných datech. K tomuto účelu jsem pořídila sadu mikroskopických snímků různých objektů s různými rovinami ostrosti. Tyto snímky je možné si prohlédnout v kapitole 7. Zde jsou uvedeny jak jednotlivé sady snímků s různými rovinami ostrosti, tak výsledný snímek složený z pořízené sady snímků. U každé jednotlivé série snímků uvádím získaný výsledek medotou gradientní i metodou frekvenčně selektivní váhovanou mediánem. Obě tyto metody dosahují téměř shodného výsledku, rozdíl mezi metodami je hlavně v čase, neboť metoda gradientní zpracovává jeden snímek přibližně 2 – 3 s (s ohledem na velikost snímku), kdežto metoda frekvenčně selektivní vahovaná mediánem 30 – 40 s.

Rychlost u obou metod závisí jednak na velikosti jednotlivých snímků, jednak samozřejmě na počtu zpracovávaných snímků. U snímků kloubní hlavice femuru, kdy je výsledek složen ze 3 řezů, jsem dosáhla celkového času potřebného na složení snímků t_{grad} = 8,06 s, t_{fsvm} = 89,89 s. Uvedené časy představují průměr ze 3 naměřených hodnot. Pro distální femur (výsledek složen z 5 snímků) se jedná o časy t_{grad} = 15,85 s, t_{fsvm} = 178,28 s. Pro snímky pořízené při mikroextirpaci hernie disci (výsledek složen z 5 snímků) jsem dosáhla časů t_{grad} = 16,32 s, t_{fsvm} = 212,71 s. Smínky čištěné kostní tkáně byly programen složeny za čas t_{grad} = 18,8 s, t_{fsvm} = 210,41 s, přičemž výsledek byl složen z 8 snímků. Konečně u hlavy tesaříka, kdy se jednalo o 11 snímků, bylo složení dosaženo za časový úsek t_{grad} = 20,91 s, t_{fsvm} = 218,15 s.

Rychlost algoritmu je dále ovlivněna velikostí masky m,n, která je postupně aplikována na všechny pixely, posouvá se, a tak se provádí zpracování načtených snímků. Pro algoritmus metodou gradientní je maska přednastavená na hodnotu m = 3, n = 3, kdy je okolí dáno jako (2m + 1) x (2n + 1). Metoda frekvenčně selektivní váhovaná mediánem pak má velikost masky m = 1, n = 1, okolí je dáno stejně jako v předchozí metodě. Zvětšení masky prodlužuje dobu výpočtu, přičemž výsledný obraz pro m = 1, 3, 5; n = 1, 3, 5 je shodný.

Závěrem lze tedy konstantovat, že vytvořený systém je univerzální, dá se aplikovat na různé snímky, jak mikroskopické, tak makroskopické, není nutné upravovat žádné parametry. Podmínkou užívání vytvořeného algoritmu je disponovat programem Matlab, v němž lze vytvořenou aplikaci otevřít a dále využívat.

Literatura

[1] BLAŠKA, J., KRUMPHOLC, M., SEDLÁČEK, M.: Využití grafického uživatelského rozhraní Matlabu ve výzkumu a výuce měření. [on-line]. 2003 [cit. 2011-3-14]. Dostupné na: <http://dsp.vscht.cz/konference_matlab/matlab03/blaska.pdf>

[2] BÜRGER, T.: Grafical User Interface v programu MATLAB. [on-line]. 2009 [cit. 2011-4-7]. Dostupné na: <http://theses.cz/id/64e10v/downloadPraceContent_adipIdno_13843>

[3] DOMINEC – Microscope [on-line]. 2008 [cit. 2011-3-5]. Dostupné na: <http://en.wikipedia.org/wiki/Microscope>

[4] FIŘT, J., HOLOTA, R.: Digitalizace a zpracování obrazu [on-line]. 2009 [cit. 2011-3-15]. Dostupné na: <http://home.zcu.cz/~holota5/publ/DigZprO.pdf>

[5] HAMPL – Světelná mikroskopie [on-line]. 2010 [cit. 2011-3-5]. Dostupné na: <http://web.natur.cuni.cz/parasitology/parpages/mikroskopickatechnika/>

[6] HOZMAN, J.: Základní metody předzpracování obrazu [on-line]. 2003 [cit. 2011-3-15]. Dostupné na: <http://webzam.fbmi.cvut.cz/hozman/Zprac_obr_prisp_kurz_UEM_3_2003.pdf>

[7] HRAZDIRA, I., MORNSTEIN, V., ŠKORPÍKOVÁ, J.: Základy biofyziky a zdravotnické techniky. Neptun, Brno 2006, 312 s. ISBN 80-86850-01-3

[8] JAN, J.: Číslicová filtrace, analýza a restaurace signálů. VUTIUM, Brno 2002, 427 s. ISBN 80-214-1558-4

[9] KAUTSKÝ, J., FLUSSER, J., ZITOVÁ, B., ŠIMBEROVÁ, S.: A new zavelet-based measure of image focus. [on-line]. 2002 [cit. 2011-3-5]. Dostupné na: <http://www.elsevier.com/locate/patrec>

[10] KOZÁK, M.: CCD [on-line]. 2007 [cit. 2011-3-28]. Dostupné na: <http://cs.wikipedia.org/wiki/CCD>

[11] MARTIŠEK, D.: Matematické metody modelování mikroskopických objektů [on-line]. 2002 [cit. 2011-3-28]. Dostupné na: <http://www.ancor.cz/no/dr_martisek.pdf>

[12] Mikroskop-mikroskopy: Mikroskopy. [on-line]. 2008 [cit. 2011-3-14]. Dostupné na: <http://www.mikroskop-mikroskopy.cz/>

[13] Optic Indutries – O mikroskopu [on-line]. 2008 [cit. 2011-3-5]. Dostupné na: <http://www.optic.cz/technicke_informace.html>

[14] PIHAN, R.: Fenomén hloubky ostrosti[on-line]. 2005 [cit. 2011-4-14]. Dostupné na: <http://fotoroman.cz/techniques2/focus_dof.htm>

[15] PONEC, M.: Rekonstrukce reliéfu lomu ze série neostrých optických snímků [on-line]. 2005 [cit. 2012-4-9]. Dostupné na: <http://dp.ponec.org/dp-ponec-lowres.pdf>

[16] Promicra – Modul Deep Focus [on-line]. 2011 [cit. 2011-5-26]. Dostupné na: <http://www.promicra.cz/produkty-deepfocusmodul.php#zalozka-galerie>

[17] Promicra – QuickPHOTO MICRO 2.3 [on-line]. 2011 [cit. 2012-3-21]. Dostupné na: <http://www.promicra.cz/produkty-quickphoto-micro.php>

[18] SLÁDEK, Z.: Seznam mikroskopických preparátů [on-line]. 2009 [cit. 2012-3-10]. Dostupné na: <http://user.mendelu.cz/sladek/mikprep/seznam.html>

[19] TAMAS – Stereo microscope [on-line]. 2010 [cit. 2011-3-5]. Dostupné na: <http://en.wikipedia.org/wiki/Stereo_microscope>

[20] TIAN, J., CHEN, L., MA, L., YU, W.: Multi-focus image fusion using a bilateral gradient-based sharpness criterion. 2011 [cit. 2011-4-25]. Dostupné na: <http://www.elsevier.com/locate/optcom>

[21] VOJTKULÁKOVÁ – Světelná mikroskopie [on-line]. 2008 [cit. 2011-3-5]. Dostupné na: <http://ime.fme.vutbr.cz/files/Studijni%20opory/sm/Index.html>

[22] VYMĚTALOVÁ, V., SAVICKÁ, D., NAMYSLO, L.: Digitální mikrofotografie [on-line]. 2004 [cit. 2011-3-22]. Dostupné na: <http://www.vscht.cz/obsah/fakulty/fpbt/ostatni/digimikro/pokus.htm>

[23] Wikipedia – Mikroskop [on-line]. 2011 [cit. 2011-3-5]. Dostupné na: <http://cs.wikipedia.org/wiki/Mikroskop>

Seznam použitých ztratek a symbolů

2D – Dvourozměrný

AFM – Mikroskopie atomárních sil (Atomic force microscopy)

CCD – Charge-Coupled Device

GUI – Graphical User Interface

GUIDE – Graphical User Interface Development Environment

PSF – Rozptylová funkce (Point Spread Function)

STM – Řádkovací tunelový mikroskop (Scanning Tunneling Microscope)

Seznam příloh

Zdrojový kód Multifocus

Zdrojový kód Multifocusdemo

CD se zdrojovým kódem vytvořeného programu, všechny pořízené snímky i s výslednými obrazy

Přílohy

Zdrojový kód Multifocus

```
function varargout = Multifocus(varargin)
% MULTIFOCUS M-file for Multifocus.fig
%      MULTIFOCUS, by itself, creates a new MULTIFOCUS or raises the existing
%      singleton*.
%
%      H = MULTIFOCUS returns the handle to a new MULTIFOCUS or the handle to
%      the existing singleton*.
%
%      MULTIFOCUS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MULTIFOCUS.M with the given input arguments.
%
%      MULTIFOCUS('Property','Value',...) creates a new MULTIFOCUS or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Multifocus_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Multifocus_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Multifocus

% Last Modified by GUIDE v2.5 15-Apr-2012 14:37:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Multifocus_OpeningFcn, ...
                   'gui_OutputFcn',  @Multifocus_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before Multifocus is made visible.
function Multifocus_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Multifocus (see VARARGIN)
clc;  %Clear Command window
global dname %Define global variable
dname='C:\'; %Path set up

%Initialization of handles.* variables
handles.inputpictures = 1;
handles.algorithm_index = 1;
showpicture = 1;
handles.validname = 0;
handles.picturefound = 0;
handles.show_picture_init = 0;
%Send message to the Status field
set(handles.status, 'String', 'Initialization OK....Ready');

% Choose default command line output for Multifocus
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Multifocus wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Multifocus_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


%**************************************************************************
%**                     LIST OF handles.* VARIABLES                      **
%**************************************************************************
%handles.inputpictures      - number of input pictures found and read
%handles.algorithm_index    - index of selected algorithm from the list
%handles.algorithm_name     - list of algorithms
%handles.validname          - indicates valid name found in the List box
%handles.picturefound       - at least one picture found while browsing
%handles.show_picture_init  - show pictures only when READ has been executed
%handles.IM(x).IM           - structure in which all input pictures are stored
%handles.nameforreading     - full path without index
%handles.extension          - file extension
%handles.dir_struct         - content of selected directory
%handles.pathname           - path
%handles.dname              - used in browse and list box
%**************************************************************************



%**************************************************************************
%**               POP-DOWN MENU ALGORITMUS IN THE PANEL VYSTUP           **
%**************************************************************************
% --- Executes on selection change in Algorithms.
function Algorithms_Callback(hObject, eventdata, handles)
% hObject    handle to Algorithms (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: contents = get(hObject,'String') returns Algorithms contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from Algorithms

%Get index of selected algorithm
handles.algorithm_index = get(hObject,'Value');
%Get list of algorithms as cell array
handles.algorithm_name = get(hObject,'String');
%Get name of the selected algorithm
selected_algorithm = handles.algorithm_name{handles.algorithm_index};

%Algorithm selection is valid only if index > 1
if handles.algorithm_index > 1
    %Send message to Status field
    set(handles.status, 'String', ['Selected Algorithm:     ',
selected_algorithm]);
end
%Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function Algorithms_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Algorithms (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%**************************************************************************
%**      POP-DOWN MENU ZOBRAZ FOTO IN THE PANEL ZOBRAZ VSTUPNI FOTO      **
%**************************************************************************
% --- Executes on selection change in Show_Input_Image.
function Show_Input_Image_Callback(hObject, eventdata, handles)
% hObject    handle to Show_Input_Image (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns Show_Input_Image contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from
Show_Input_Image

%Get index for displying a picture
showpicture = get(hObject, 'Value');
%Executes only when READ was executed
if handles.show_picture_init
    %Set axes "picture" as the current
    imshow (handles.IM(showpicture).IM, 'Parent', handles.picture);
    %Send message to Status field
    set(handles.status, 'String', ['Showing picture #:     ',
int2str(showpicture)]);
end

% --- Executes during object creation, after setting all properties.
function Show_Input_Image_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Show_Input_Image (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%**************************************************************************
%**            PUSH BUTTON VSTUP IN THE PANEL VSTUP                      **
%**************************************************************************
% --- Executes on button press in Read_Input_Images.
function Read_Input_Images_Callback(hObject, eventdata, handles)
% hObject    handle to Read_Input_Images (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Set the variable for Showe_Input_Image
handles.show_picture_init = 1;
%At least one picture found in the current directory
if handles.validname
    %Prepare column based on # of pictures found
    colvector = [1:handles.inputpictures]';
    %Set the right menu for the pop-down menu
    set(handles.Show_Input_Image, 'String', colvector);
    %Set Icon panel as the Current axes for displaying the pictures and clears
"iconpanel"
    icon = subplot(1, 1, 1, 'Parent', handles.iconpanel);
    %Reads and stores input pictures into the handles.IM(i).IM structure
    for i = 1:1:handles.inputpictures
        %Get the full path and file name
        file = [handles.nameforreading, int2str(i), handles.extension];
        %Read into handles.IMx.IM
        tempim = imread(file);
        %Get info about the picture
        info = imfinfo(file);
        %'truecolor' picture convert to gray scale
        if strcmpi(info.ColorType, 'truecolor')
            handles.IM(i).IM = rgb2gray(tempim);
        else
            handles.IM(i).IM = tempim;
        end
%**************************************************************************
%**************************************************************************
%**       Reduce size of input pictures if computation takes too long    **
%**                                                                      **
%**               %handles.IM(i).IM = imresize(tempim, 0.5);             **
%**************************************************************************
%**************************************************************************
        %Display max 5 icons in the "Vstupni fotky" panel
        if i <= 5
            %Set Icon panel as the Current axes for displaying the pictures
            icon = subplot(5, 1, i, 'Parent', handles.iconpanel);
            set(handles.figure1, 'CurrentAxes', icon);
            imshow(handles.IM(i).IM);
            title(['Foto ',int2str(i)]);
        end
    end
    %Find out the size of the first picture
    [rows cols] = size(handles.IM(1).IM);
    %Send message to Status field
    set(handles.status, 'String', [int2str(handles.inputpictures),' Input
Pictures stored, size ',int2str(rows),' x ',int2str(cols)]);
    %Update hadles structure
    guidata(hObject, handles);
else
    %Send message to Status field
    set(handles.status, 'String', 'File name is not in the format "namexx",
where x are numbers 1-9');
```

```
end

%**************************************************************************
%**                  PUSH BUTTON SPOJENI IN THE PANEL VYSTUP               **
%**************************************************************************
% --- Executes on button press in Execute_Fusion.
function Execute_Fusion_Callback(hObject, eventdata, handles)
% hObject    handle to Execute_Fusion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%**************************************************************************
%**************************************************************************
%**                  ALGORITHM EMERGY OF IMAGE GRADIENT                    **
%**************************************************************************
%**************************************************************************
if handles.algorithm_index == 2     %Index = 2 --> Energy of Image Gradient
    %Mask iteractively applied for the gradient calculation
    m = 3;  %Size of neighborhood is 2m+1 x 2n+1
    n = 3;
    %Size of the input picture
    [rows cols] = size(handles.IM(1).IM);
    %Displays window with the waitbar
    w = waitbar(0, 'Please wait.......');
    tic;
    for i = 1:1:handles.inputpictures
        %All calculation in double
        AA = im2double(handles.IM(i).IM);
        %Pad array with [m n], default is 'replicate' and 'both'
        A = padarray(AA, [m n]);
        for j = 1:1:rows
            for k = 1:1:cols
                tempweight = 0;
                %Watch the edges to make sure index stays within the array
                if j == rows
                    jj = j + m - 1;
                else
                    jj = j + m;
                end
                if k == cols
                    kk = k + n - 1;
                else
                    kk = k + n;
                end
                %Applying the mask - Algorithm Energy of Image Gradient
                for r = -m:1:m
                    for s = -n:1:n
                        ix = A((jj + r + 1), (kk + s)) - A((jj + r), (kk +
s));
                        iy = A((jj + r), (kk + s + 1)) - A((jj + r), (kk +
s));
                        tempweight = tempweight + ix^2 + iy^2;
                    end
                end
                %Set the weight for every pixel
                weight(i).grad(j, k) = tempweight;
            end
            %Update waitbar for each picture
            waitbar(j / rows, w, ['Please wait...Processing Picture ',
int2str(i)]);
        end
        %Convert matrix of weights to the nomalized form
        norm = max(max(weight(i).grad));
        weight(i).grad_n = imdivide(weight(i).grad, norm);
    end
    %Close the window with the waitbar
    close(w);
```

```
    tElapsed = toc

    %Double array preparation with rows x cols size
    suma_weight(rows, cols) = 0;
    suma_weight_d = double(suma_weight);
    %Sum of weight needed for the pciture fusion
    for i = 1:1:handles.inputpictures
         suma_weight_d = imadd(suma_weight_d, weight(i).grad_n);
    end
    %Double array preparation with rows x cols size
    output_picture_temp(rows, cols) = 0;
    output_picture = double(output_picture_temp);
    %Calculation of the output picture
    for i = 1:1:handles.inputpictures
        image(i).IM = double(handles.IM(i).IM);
        output_picture = output_picture + imdivide(immultiply(image(i).IM,
weight(i).grad_n), suma_weight_d);
    end
    %Convert to integer
    output_picture8 = uint8(output_picture);
    %Create the output file and save into the working directory
    file = [handles.pathname, 'output', handles.extension];
    imwrite(output_picture8, file);
    %Display result picture
    imshow(output_picture8, 'Parent', handles.picture);
    %Selected algorithm used for the Fusion
    selected_algorithm = handles.algorithm_name{handles.algorithm_index};
    %Send message to Status field
    set(handles.status, 'String', ['The picture shows result of the '
selected_algorithm]);

%**************************************************************************
%**************************************************************************
%**        ALGORITHM FREQUENCY SELECTIVE WEIGHTED MEDIAN FILTER          **
%**************************************************************************
%**************************************************************************
elseif handles.algorithm_index == 3     %Index = 3 --> Frequency selective
weighted median filter
    %%Mask iteractively applied for the gradient calculation
    m = 1;  %size of neighborhood is 2m+1 x 2n+1
    n = 1;  %
    %Size of the input picture
    [rows cols] = size(handles.IM(1).IM);
    %Displays window with the waitbar
    w = waitbar(0, 'Please wait....');
    tic;
    for i = 1:1:handles.inputpictures
        %All calculation in double
        AA = im2double(handles.IM(i).IM);
        %Pad array with [m n], default is 'replicate' and 'both'
        A = padarray(AA, [m+3 n+3]);
        for j = 1:1:rows
            for k = 1:1:cols
                %Watch the edges to make sure index stays within the array
                jj = j + m + 3;
                kk = k + n + 3;
                tempweight = 0;
                %Applying the mask - Algorithm Frequency selective weighted
median filter
                for r = -m:1:m
                    for s = -n:1:n
                        %Select the right pixels and sort them
                        r1 = [A((jj+r-1),(kk+s)) A((jj+r),(kk+s))
A((jj+r+1),(kk+s))];
                        r1_s = sort(r1);
```

```
                        r2 = [A((jj+r-3),(kk+s)) A((jj+r-2),(kk+s)) A((jj+r-
1),(kk+s))];
                        r2_s = sort(r2);
                        r3 = [A((jj+r+1),(kk+s)) A((jj+r+2),(kk+s))
A((jj+r+3),(kk+s))];
                        r3_s = sort(r3);
                        %Median is simpy the middle value of sorted array with
3 elements
                        ix = r1_s(m + 1) - r2_s(m + 1)/2 - r3_s(m + 1)/2;
                        c1 = [A((jj+r+1),(kk+s)) A((jj+r+2),(kk+s))
A((jj+r+3),(kk+s))];
                        c1_s = sort(c1);
                        c2 = [A((jj+r),(kk+s-3)) A((jj+r),(kk+s-2))
A((jj+r),(kk+s-1))];
                        c2_s = sort(c2);
                        c3 = [A((jj+r),(kk+s+1)) A((jj+r),(kk+s+2))
A((jj+r),(kk+s+3))];
                        c3_s = sort(c3);
                        %Median is simpy the middle value of sorted array with
3 elements
                        iy = c1_s(n + 1) - c2_s(n + 1)/2 - c3_s(n + 1)/2;
                        tempweight = tempweight + ix^2 + iy^2;
                    end
                end
                %Set the weight for every pixel
                weight(i).grad(j, k) = tempweight;
            end
            %Update waitbar for each picture
            waitbar(j / rows, w, ['Please wait...Processing Picture ',
int2str(i)]);
        end
        %Convert matrix of weights to the nomalized form
        norm = max(max(weight(i).grad));
        weight(i).grad_n = imdivide(weight(i).grad, norm);
    end
    %Close window with the waitbar
    close(w);

    tElapsed = toc

    %Double array preparation with rows x cols size
    suma_weight(rows, cols) = 0;
    suma_weight_d = double(suma_weight);
    %Sum of weight needed for the pciture fusion
    for i = 1:1:handles.inputpictures
         suma_weight_d = imadd(suma_weight_d, weight(i).grad_n);
    end
    %Double array preparation with rows x cols size
    output_picture_temp(rows, cols) = 0;
    output_picture = double(output_picture_temp);
    %Calculation of the output picture
    for i = 1:1:handles.inputpictures
        image(i).IM = double(handles.IM(i).IM);
        output_picture = output_picture + imdivide(immultiply(image(i).IM,
weight(i).grad_n), suma_weight_d);
    end
    %Convert to integer
    output_picture8 = uint8(output_picture);
    %create the output file and savs into the selected directory
    file = [handles.pathname, 'output', handles.extension];
    imwrite(output_picture8, file);
    %Display result picture
    imshow(output_picture8, 'Parent', handles.picture);
    %Selected algorithm used for the Fusion
    selected_algorithm = handles.algorithm_name{handles.algorithm_index};
    %Send message to Status field
```

```
    set(handles.status, 'String', ['The picture shows result of the '
selected_algorithm]);

%**************************************************************************
%**************************************************************************
%**    Here put "elseif handles.algorithm_index == 4" and an additional  **
%**    algorithm. You need also to modify "algorithms" pop-down menu in  **
%**    the VYSTUP panel by adding new algorithm name onto the list.      **
%**************************************************************************
%**************************************************************************
else
    %Non algorithm chosen
    set(handles.status, 'String', 'Please choose an algorithm');
end



%**************************************************************************
%**                           PUSH BUTTON DEMO                           **
%**************************************************************************
% --- Executes on button press in Multifocusdemo.
function Multifocusdemo_Callback(hObject, eventdata, handles)
% hObject    handle to Multifocusdemo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Invoke multifocusdemo.fig file to run the demo
multifocusdemo



%**************************************************************************
%**            PUSH BUTTON PROCHAZEJ... IN THE PANEL VSTUP               **
%**************************************************************************
% --- Executes on button press in Browse.
function Browse_Callback(hObject, eventdata, handles)
% hObject    handle to Browse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Opens Dialog box, gets content of selected directory into dname
handles.dname = uigetdir('C:\');
%List box filled only if at least one picture found
handles.picturefound = 0;
%Avoid Cancel problem with '0' in dname
if handles.dname ~= 0
    %handles.dir_struct contains info about all directories and files in the
selected directory
    handles.dir_struct = dir(handles.dname);
    %Size of handles.dir_struct
    [x ,y, z] = size(handles.dir_struct);
    %Index for the list_of_files
    moveindex = 1;
    %Look for pictures in the current directory
    for i = 1:1:x
        %Get info about the file
        [path, name, ext] = fileparts(handles.dir_struct(i).name);
        %Look for JPG, BMP or TIF pictures and fill in list which is shown in
th List box
        switch ext
            %jpg or JPG picture will be added
            case '.jpg'
                %Add valid file name onto the list
                list_of_files{moveindex} = [name, ext];
                moveindex = moveindex + 1;
                handles.picturefound = 1;
            case '.JPG'
                %Add valid file name onto the list
```

```
                list_of_files{moveindex} = [name, ext];
                moveindex = moveindex + 1;
                handles.picturefound = 1;
            %bmp or BMP picture will be added
            case '.bmp'
                %Add valid file name onto the list
                list_of_files{moveindex} = [name, ext];
                moveindex = moveindex + 1;
                handles.picturefound = 1;
            case '.BMP'
                %Add valid file name onto the list
                list_of_files{moveindex} = [name, ext];
                moveindex = moveindex + 1;
                handles.picturefound = 1;
            %tif or TIF picture will be added
            case '.tif'
                %Add valid file name onto the list
                list_of_files{moveindex} = [name, ext];
                moveindex = moveindex + 1;
                handles.picturefound = 1;
            case '.TIF'
                %Add valid file name onto the list
                list_of_files{moveindex} = [name, ext];
                moveindex = moveindex + 1;
                handles.picturefound = 1;
        end
    end
end
%List of files found is displayed in the List box
if handles.picturefound
    set(handles.Browse_window, 'String', list_of_files);
end
%Update handles structure
guidata(hObject, handles);



%**************************************************************************
%**                LIST BOX IN THE PANEL VSTUP                           **
%**************************************************************************
% --- Executes on selection change in Browse_window.
function Browse_window_Callback(hObject, eventdata, handles)
% hObject    handle to Browse_window (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Browse_window
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
Browse_window

%Get index of the picture selected in the List box
index_selected = get(handles.Browse_window, 'Value');
%Get list of all pictures from the List box
file_list = get(handles.Browse_window, 'String');
%Get name of the picture selected in the List box
filename = file_list{index_selected};
%Get extension of the selected file
[temp, temp1, handles.extension] = fileparts(filename);
%Path to the selected picture
handles.pathname = strcat(handles.dname, '\');
%Get the full path including the file name for imread and imshow
name2 = strcat(handles.pathname, filename);
%Show picture in the current axes picture
imshow(name2, 'Parent', handles.picture);
%Indicates valid name for Read_Input_Images
handles.validname = 0;
%Get filename and path without extension
```

```
filename_reduced = strtok(filename, '.');
name2_reduced = strtok(name2, '.');
%Transform file_list to one row
file_list_row = (file_list)';
%Get number of characters in the file name, temp = 1 means one row
[temp lenght_of_filename] = size(filename_reduced);

%Get number of files with name in "filename" and in the form
<filename><X>.<ext>, where X is 1 to n
%Check that last character is number
if (filename_reduced(lenght_of_filename) >= '1') &&
(filename_reduced(lenght_of_filename) <= '9')
    %Get row vector where "1" indicate positive comparison of file names
without index
    samenames = strncmpi(file_list_row, filename_reduced, (lenght_of_filename
- 1));
    %Sum of "1"
    samenamesnumber = find(samenames);
    %handles.inputpictures contains number of pictures found and ready for
reading
    [temp handles.inputpictures] = size(samenamesnumber);
    handles.validname = 1;
end
%Get lenght (# of chracters) of the full path including the file name without
extension
[temp lenght_of_fullname] = size(name2_reduced);
%Clear variable
handles.nameforreading = '';
%Cut the last digit/index
for i = 1:1:(lenght_of_fullname - 1)
    %Get the full path without index
    handles.nameforreading(i) = name2_reduced(i);
end
%Update handles structure
guidata(hObject, handles);



% --- Executes during object creation, after setting all properties.
function Browse_window_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Browse_window (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Zdrojový kód Multifocusdemo

```
function varargout = Multifocusdemo(varargin)
% MULTIFOCUSDEMO M-file for Multifocusdemo.fig
%      MULTIFOCUSDEMO, by itself, creates a new MULTIFOCUSDEMO or raises the
existing
%      singleton*.
%
%      H = MULTIFOCUSDEMO returns the handle to a new MULTIFOCUSDEMO or the
handle to
%      the existing singleton*.
```

```
%
%      MULTIFOCUSDEMO('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in MULTIFOCUSDEMO.M with the given input
arguments.
%
%      MULTIFOCUSDEMO('Property','Value',...) creates a new MULTIFOCUSDEMO or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Multifocusdemo_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Multifocusdemo_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Multifocusdemo

% Last Modified by GUIDE v2.5 18-Apr-2012 00:18:02

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Multifocusdemo_OpeningFcn, ...
                   'gui_OutputFcn',  @Multifocusdemo_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before Multifocusdemo is made visible.
function Multifocusdemo_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Multifocusdemo (see VARARGIN)
clc;  %% vymažeme konzoli
global dname %% definování globální proměnné
dname='C:\'; %% nastavení cesty

%Initialization of handles.* variables
handles.inputpictures = 2;
handles.algorithm_index = 1;
showpicture = 1;
handles.validname = 1;
handles.picturefound = 0;
handles.show_picture_init = 0;
%Demo specific set up
handles.nameforreading = 'clock_';
handles.extension = '.bmp';
handles.pathname = '';
```

```
%Send message to the Status field
set(handles.status, 'String', 'Initialization OK....Ready');

% Choose default command line output for Multifocusdemo
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Multifocusdemo wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Multifocusdemo_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



%**************************************************************************
%**                      LIST OF handles.* VARIABLES                      **
%**************************************************************************
%handles.inputpictures      - number of input pictures found and read
%handles.algorithm_index    - index of selected algorithm from the list
%handles.algorithm_name     - list of algorithms
%handles.validname          - indicates valid name found in the List box
%handles.picturefound       - at least one picture found while browsing
%handles.show_picture_init  - show pictures only when READ has been executed
%handles.IM(x).IM           - structure in which all input pictures are stored
%handles.nameforreading     - full path without index
%handles.extension          - file extension
%handles.dir_struct         - content of selected directory
%handles.pathname           - path
%**************************************************************************



%**************************************************************************
%**               POP-DOWN MENU ALGORITMUS IN THE PANEL VYSTUP           **
%**************************************************************************
% --- Executes on selection change in Algorithms.
function Algorithms_Callback(hObject, eventdata, handles)
% hObject    handle to Algorithms (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns Algorithms contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from Algorithms

%Get index of selected algorithm
handles.algorithm_index = get(hObject,'Value');
%Get list of algorithms as cell array
handles.algorithm_name = get(hObject,'String');
%Get name of the selected algorithm
selected_algorithm = handles.algorithm_name{handles.algorithm_index};

%Algorithm selection is valid only if index > 1
if handles.algorithm_index > 1
    %Send message to Status field
    set(handles.status, 'String', ['Selected Algorithm:     ',
selected_algorithm]);
```

```
end
%Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function Algorithms_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Algorithms (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%**************************************************************************
%**      POP-DOWN MENU ZOBRAZ FOTO IN THE PANEL ZOBRAZ VSTUPNI FOTO      **
%**************************************************************************
% --- Executes on selection change in Show_Input_Image.
function Show_Input_Image_Callback(hObject, eventdata, handles)
% hObject    handle to Show_Input_Image (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns Show_Input_Image contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from
Show_Input_Image

%Get index for displying a picture
showpicture = get(hObject, 'Value');
%Executes only when READ was executed
if handles.show_picture_init
    %Set axes "picture" as the current
    imshow (handles.IM(showpicture).IM, 'Parent', handles.picture);
    %Send message to Status field
    set(handles.status, 'String', ['Showing picture #:     ',
int2str(showpicture)]);
end

% --- Executes during object creation, after setting all properties.
function Show_Input_Image_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Show_Input_Image (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%**************************************************************************
%**            PUSH BUTTON VSTUP IN THE PANEL VSTUP                      **
%**************************************************************************
% --- Executes on button press in Read_Input_Images.
function Read_Input_Images_Callback(hObject, eventdata, handles)
% hObject    handle to Read_Input_Images (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Set the variable for Showe_Input_Image
```

```
handles.show_picture_init = 1;
%At least one picture found in the current directory
if handles.validname
    %Prepare column based on # of pictures found
    colvector = [1:handles.inputpictures]';
    %Set the right menu for the pop-down menu
    set(handles.Show_Input_Image, 'String', colvector);
    %Set Icon panel as the Current axes for displaying the pictures and clears
"iconpanel"
    icon = subplot(1, 1, 1, 'Parent', handles.iconpanel);
    %Reads and stores input pictures into the handles.IM(i).IM structure
    for i = 1:1:handles.inputpictures
        %Get the full path and file name
        file = [handles.nameforreading, int2str(i), handles.extension];
        %Read into handles.IMx.IM
        tempim = imread(file);
        %Get info about the picture
        info = imfinfo(file);
        %'truecolor' picture convert to gray scale
        if strcmpi(info.ColorType, 'truecolor')
            handles.IM(i).IM = rgb2gray(tempim);
        else
            handles.IM(i).IM = tempim;
        end
%************************************************************************
%************************************************************************
%**       Reduce size of input pictures if computation takes too long    **
%**                                                                      **
%**               %handles.IM(i).IM = imresize(tempim, 0.5);             **
%************************************************************************
%************************************************************************
        %Display max 5 icons in the "Vstupni fotky" panel
        if i <= 5
            %Set Icon panel as the Current axes for displaying the pictures
            icon = subplot(5, 1, i, 'Parent', handles.iconpanel);
            set(handles.figure1, 'CurrentAxes', icon);
            imshow(handles.IM(i).IM);
            title(['Foto ',int2str(i)]);
        end
    end
    %Find out the size of the first picture
    [rows cols] = size(handles.IM(1).IM);
    %Send message to Status field
    set(handles.status, 'String', [int2str(handles.inputpictures),' Input
Pictures stored, size ',int2str(rows),' x ',int2str(cols)]);
    %Update hadles structure
    guidata(hObject, handles);
else
    %Send message to Status field
    set(handles.status, 'String', 'File name is not in the format "namexx",
where x are numbers 1-9');
end

%************************************************************************
%**                PUSH BUTTON SPOJENI IN THE PANEL VYSTUP               **
%************************************************************************
% --- Executes on button press in Execute_Fusion.
function Execute_Fusion_Callback(hObject, eventdata, handles)
% hObject    handle to Execute_Fusion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%************************************************************************
%************************************************************************
%**                ALGORITHM EMERGY OF IMAGE GRADIENT                    **
%************************************************************************
%************************************************************************
```

```
if handles.algorithm_index == 2     %Index = 2 --> Energy of Image Gradient
    %Mask iteractively applied for the gradient calculation
    m = 3;  %Size of neighborhood is 2m+1 x 2n+1
    n = 3;
    %Size of the input picture
    [rows cols] = size(handles.IM(1).IM);
    %Displays window with the waitbar
    w = waitbar(0, 'Please wait.......');
    for i = 1:1:handles.inputpictures
        %All calculation in double
        AA = im2double(handles.IM(i).IM);
        %Pad array with [m n], default is 'replicate' and 'both'
        A = padarray(AA, [m n]);
        for j = 1:1:rows
            for k = 1:1:cols
                tempweight = 0;
                %Watch the edges to make sure index stays within the array
                if j == rows
                    jj = j + m - 1;
                else
                    jj = j + m;
                end
                if k == cols
                    kk = k + n - 1;
                else
                    kk = k + n;
                end
                %Applying the mask - Algorithm Energy of Image Gradient
                for r = -m:1:m
                    for s = -n:1:n
                        ix = A((jj + r + 1), (kk + s)) - A((jj + r), (kk +
s));
                        iy = A((jj + r), (kk + s + 1)) - A((jj + r), (kk +
s));
                        tempweight = tempweight + ix^2 + iy^2;
                    end
                end
                %Set the weight for every pixel
                weight(i).grad(j, k) = tempweight;
            end
            %Update waitbar for each picture
            waitbar(j / rows, w, ['Please wait...Processing Picture ',
int2str(i)]);
        end
        %Convert matrix of weights to the nomalized form
        norm = max(max(weight(i).grad));
        weight(i).grad_n = imdivide(weight(i).grad, norm);
    end
    %Close the window with the waitbar
    close(w);
    %Double array preparation with rows x cols size
    suma_weight(rows, cols) = 0;
    suma_weight_d = double(suma_weight);
    %Sum of weight needed for the pciture fusion
    for i = 1:1:handles.inputpictures
         suma_weight_d = imadd(suma_weight_d, weight(i).grad_n);
    end
    %Double array preparation with rows x cols size
    output_picture_temp(rows, cols) = 0;
    output_picture = double(output_picture_temp);
    %Calculation of the output picture
    for i = 1:1:handles.inputpictures
        image(i).IM = double(handles.IM(i).IM);
        output_picture = output_picture + imdivide(immultiply(image(i).IM,
weight(i).grad_n), suma_weight_d);
    end
    %Convert to integer
    output_picture8 = uint8(output_picture);
```

```
    %Create the output file and save into the working directory
    file = [handles.pathname, 'output', handles.extension];
    imwrite(output_picture8, file);
    %Display result picture
    imshow(output_picture8, 'Parent', handles.picture);
    %Selected algorithm used for the Fusion
    selected_algorithm = handles.algorithm_name{handles.algorithm_index};
    %Send message to Status field
    set(handles.status, 'String', ['The picture shows result of the '
selected_algorithm]);

%**************************************************************************
%**************************************************************************
%**        ALGORITHM FREQUENCY SELECTIVE WEIGHTED MEDIAN FILTER          **
%**************************************************************************
%**************************************************************************
elseif handles.algorithm_index == 3     %Index = 3 --> Frequency selective
weighted median filter
    %%Mask iteractively applied for the gradient calculation
    m = 1;  %size of neighborhood is 2m+1 x 2n+1
    n = 1;  %
    %Size of the input picture
    [rows cols] = size(handles.IM(1).IM);
    %Displays window with the waitbar
    w = waitbar(0, 'Please wait....');
    for i = 1:1:handles.inputpictures
        %All calculation in double
        AA = im2double(handles.IM(i).IM);
        %Pad array with [m n], default is 'replicate' and 'both'
        A = padarray(AA, [m+3 n+3]);
        for j = 1:1:rows
            for k = 1:1:cols
                %Watch the edges to make sure index stays within the array
                jj = j + m + 3;
                kk = k + n + 3;
                tempweight = 0;
                %Applying the mask - Algorithm Frequency selective weighted
median filter
                for r = -m:1:m
                    for s = -n:1:n
                        %Select the right pixels and sort them
                        r1 = [A((jj+r-1),(kk+s)) A((jj+r),(kk+s))
A((jj+r+1),(kk+s))];
                        r1_s = sort(r1);
                        r2 = [A((jj+r-3),(kk+s)) A((jj+r-2),(kk+s)) A((jj+r-
1),(kk+s))];
                        r2_s = sort(r2);
                        r3 = [A((jj+r+1),(kk+s)) A((jj+r+2),(kk+s))
A((jj+r+3),(kk+s))];
                        r3_s = sort(r3);
                        %Median is simpy the middle value of sorted array with
3 elements
                        ix = r1_s(m + 1) - r2_s(m + 1)/2 - r3_s(m + 1)/2;
                        c1 = [A((jj+r+1),(kk+s)) A((jj+r+2),(kk+s))
A((jj+r+3),(kk+s))];
                        c1_s = sort(c1);
                        c2 = [A((jj+r),(kk+s-3)) A((jj+r),(kk+s-2))
A((jj+r),(kk+s-1))];
                        c2_s = sort(c2);
                        c3 = [A((jj+r),(kk+s+1)) A((jj+r),(kk+s+2))
A((jj+r),(kk+s+3))];
                        c3_s = sort(c3);
                        %Median is simpy the middle value of sorted array with
3 elements
                        iy = c1_s(n + 1) - c2_s(n + 1)/2 - c3_s(n + 1)/2;
                        tempweight = tempweight + ix^2 + iy^2;
                    end
                end
```

```
                %Set the weight for every pixel
                weight(i).grad(j, k) = tempweight;
            end
            %Update waitbar for each picture
            waitbar(j / rows, w, ['Please wait...Processing Picture ',
int2str(i)]);
        end
        %Convert matrix of weights to the nomalized form
        norm = max(max(weight(i).grad));
        weight(i).grad_n = imdivide(weight(i).grad, norm);
    end
    %Close window with the waitbar
    close(w);
    %Double array preparation with rows x cols size
    suma_weight(rows, cols) = 0;
    suma_weight_d = double(suma_weight);
    %Sum of weight needed for the pciture fusion
    for i = 1:1:handles.inputpictures
         suma_weight_d = imadd(suma_weight_d, weight(i).grad_n);
    end
    %Double array preparation with rows x cols size
    output_picture_temp(rows, cols) = 0;
    output_picture = double(output_picture_temp);
    %Calculation of the output picture
    for i = 1:1:handles.inputpictures
        image(i).IM = double(handles.IM(i).IM);
        output_picture = output_picture + imdivide(immultiply(image(i).IM,
weight(i).grad_n), suma_weight_d);
    end
    %Convert to integer
    output_picture8 = uint8(output_picture);
    %create the output file and savs into the selected directory
    file = [handles.pathname, 'output', handles.extension];
    imwrite(output_picture8, file);
    %Display result picture
    imshow(output_picture8, 'Parent', handles.picture);
    %Selected algorithm used for the Fusion
    selected_algorithm = handles.algorithm_name{handles.algorithm_index};
    %Send message to Status field
    set(handles.status, 'String', ['The picture shows result of the '
selected_algorithm]);

%**************************************************************************
%**************************************************************************
%**    Here put "elseif handles.algorithm_index == 4" and an additional  **
%**    algorithm. You need also to modify "algorithms" pop-down menu in  **
%**    the VYSTUP panel by adding new algorithm name onto the list.      **
%**************************************************************************
%**************************************************************************
else
    %Non algorithm chosen
    set(handles.status, 'String', 'Please choose an algorithm');
end
```