ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

Katedra aplikované elektroniky a telekomunikací

DIPLOMOVÁ PRÁCE

Digitální zpracování obrazu a korekce jeho zkreslení

Autor práce: **Bc. Ondřej Virt**

Vedoucí práce: **Ing. Ivo Veřtát, Ph.D**

2017

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej VIRT**
Osobní číslo:	**E15N0063P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Telekomunikační a multimediální systémy**
Název tématu:	**Digitální zpracování obrazu a korekce jeho zkreslení**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Charakterizujte různé druhy zkreslení obrazu s důrazem zejména na rozostření obrazu.
2. Vytvořte funkce v Matlabu pro lokální potlačení vybraných druhů zkreslení, zejména zkreslení způsobená pohybem objektu během jeho snímání.
3. Vytvořte funkce v Matlabu pro lokální zvýšení rozlišení obrazu, pokud je k dispozici více snímků objektu.
4. Ilustrujte účinnost zpracování obrazu na ukázkových obrazových materiálech.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah kvalifikační práce: **40 - 60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí diplomové práce: **Ing. Ivo Veřtát, Ph.D.**
Katedra aplikované elektroniky a telekomunikací

Datum zadání diplomové práce: **14. října 2016**

Termín odevzdání diplomové práce: **19. května 2017**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 14. října 2016

Abstrakt

V této práci je řešeno pomocí programu Matlab zpracování obrazu a úprava lokálních částí obrazu pro eliminaci rozmazání, způsobeného pohybem snímaného předmětu nebo pohybem kamery. Dále je řešeno zvýšení lokálního rozlišení obrazu pomocí více snímků metodou tzv. super-resolution imaging.

Cílem je vytvořit knihovnu funkcí, které se dají na úpravu rozmazaných fotografií použít. Zkušená obsluha bude mít možnost zasahovat do těchto skriptů a měnit jednotlivé parametry, aby byl výsledek korekce co možná nejlepší. Toto u komerčních programů na úpravu fotografií nelze.

Jednotlivé skripty se zabývají úpravou pouze lokální části rozmazaných fotografií, ale ne fotografie jako celek. Jedná se například o zaostření textu na jedoucích vozidlech, zaostření obličeje člověka zachyceného na dopravní kameře apod.

Klíčová slova

Matlab, pohybová neostrost, roztřesení, super-resolution imaging, lokální rozmazání obrazu, zaostření textu, lokální zvýšení rozlišení, Wienerův filter, Blind deconvolution, PSF, metoda Lucy-Richardson.

Abstract

In this thesis is solved by using Matlab image processing and correction of locals parts of an image to eliminate motion blur caused by the movement of subjects or camera. Further, is solved increase resolution of local parts of an image by using more images. This method called super-resolution imaging.

The objective is to create a library of functions which can be use to correct blurry pictures. An experienced operator will be able to interfere with the script and change the parameters for the best result. This commercial programs for editing photos can not.

The scripts deal with correction only the local part of the blurry photos, but not the photo as a whole. These include text focusing on moving vehicles, the focus of the face of man captured on a traffic camera, etc.

Key words

Matlab, motion blur, blurring, super-resolution imaging, local blur of the image, text focusing, local increase resolution, Wiener filter, Blind deconvolution, PSF, Lucy-Richardson method.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

.....

podpis

V Plzni dne 17.5.2017 Bc. Ondřej Virt

Poděkování

Tímto bych rád poděkoval vedoucímu diplomové práce Ing. Ivu Veřtátovi, Ph.D. za cenné profesionální rady, připomínky a metodické vedení práce. A také do zapojení projektu SGS. Projekt SGS-2015-002.

Obsah

Seznam symbolů a zkratek

GUI ..... Graphical user interface (grafické uživatelské rozhraní)

NSR..... Noise-to-Signal Ratio (poměr šumu a signálu)

PSF..... Point spread function (funkce bodového rozpětí)

RGB ..... Red, green, blue (červená, zelená, modrá)

SR..... Super-resolution imaging

D..... matice diskrétního podvzorkování (decimace)

f(x,y)..... zdrojový obraz (nerozmazaný)

F(u,v)..... funkce Fourierovy transformace pro f(x,y)

$\hat{f}(x,y)$ ..... nejlepší aproximace zdrojového obrazu

$\bar{f}$ ..... obraz s vysokým rozlišením (super-resolution image)

g(x,y)..... výsledný rozmazaný obraz

g_n ..... n-tý pozorovaný obraz s nízkým rozlišením

h(x,y)..... funkce rozmazání (PSF funkce)

H(u,v)..... funkce Fourierovy transformace pro h(x,y)

H..... matice konvoluce z diskretizace h(x,y)

n(x,y)..... šum

S_f ..... spektrum zdrojového obrazu

S_η ..... spektrum šumu

$s\downarrow$..... podvzorkování o faktor s

T_n..... geometrická deformace vhodnou aproximací

α_n, β_n ..... skalární parametry osvětlení

η_n..... pozorovatelný šum

τ_n ..... geometrická transformace n-tého obrazu

1 Úvod

V současné době existuje celá řada grafických editorů (včetně bezplatných) pro práci s fotografiemi, které nabízí širokou škálu funkcí pro úpravu fotografií, avšak rozsah těchto funkcí bývá obdobný a využívají zpravidla základní metody zpracování obrazové informace. Předkládaná diplomová práce je zaměřena na úpravu lokálních částí obrazu pomocí programu Matlab. Jedná se o zaostření textů zachycených na jedoucích automobilech, nebo zvýšení rozlišení, pomocí více snímků získaných z videa nebo sekvenčním focením. Zkušená obsluha bude moci v průběhu úpravy obrazů měnit parametry a metody. To vede k lepším výsledkům zpracování pro vnímání člověkem.

Text je rozdělen do několika kapitol; v první kapitole je čtenář uveden do problematiky. Druhá kapitola obsahuje cíle práce, třetí kapitola je zaměřena na zkreslení obrazu jako takové. Ve čtvrté je nastíněna pohybová neostrost a to buď vlivem pohybu kamery nebo vlivem pohybu snímaného objektu tzv. pohybové neostrosti. Bude vysvětleno co je pohybová neostrost a jak vzniká a jak je možné se ji vyhnout. V angličtině je tento termín označován jako motion blur. V páté kapitole budou popsány jednotlivé metody na odstranění lokálního zkreslení, kde se budu zabývat především odstraněním zkreslení textů například na jedoucích automobilech apod. V následující kapitole budou ukázány metody pro zvýšení lokálního rozlišení, především super-resolution imaging. Předposlední kapitola bude zaměřena na zdrojový kód v programu Matlab a vysvětlení funkce samotných programů. Kapitola bude rozdělena na dvě části, první se bude zabývat algoritmem pro odstranění lokálního zkreslení a druhá na lokální zvýšení rozlišení. V poslední kapitole budou uvedeny jednotlivé ukázky použitých metod na odstranění lokálního zkreslení a na zvýšení lokální rozlišení. V závěru budou shrnuty výsledky práce a bude uvedeno, zda byly splněny určené cíle práce.

Na začátku práce byly zvoleny cíle práce, které byla snaha splnit. Ve třetí kapitole je pak stručně a srozumitelně vysvětleno co je zkreslení obrazu, jak vzniká a jak se mu vyvarovat, případně jak ho odstranit. Kapitola je rozdělena na dvě části. První se zabývá zkreslením obrazu vlivem pohybu snímaného objektu a druhá se zabývá zkreslením obrazu vlivem pohybu kamery. Je vysvětleno jak jednotlivá zkreslení vznikají a jak se jich při snímání obrazu vyvarovat.

Dále je popsáno jak taková zkreslení odstranit. Jsou uvedeny metody, které s tímto zkreslení pracují. Tyto metody jsou blíže popsány a jsou vysvětleny jejich vstupní parametry. Úpravami těchto vstupních parametrů se snažíme docílit co nejlepší výsledku a odstranění lokálního rozlišení. Důležitým parametrem pro dekonvoluci (odstranění rozmazání) je funkce PSF. Tato funkce je blíže vysvětlena v kapitole 4.1.1.

V následující kapitole je popsána metoda pro zvýšení lokálního rozlišení tzv. metoda super-resolution imaging. Tato metoda využívá sekvence snímků stejného místa. Tyto snímky lze získat pomocí sekvenčního focení. Druhou možností je danou scénu natočit na video kameru a ze získané videosekvence získat jednotlivé snímky (framy). Je uvedeno několik přístupů jak z těchto snímků získat tzv. snímek se super rozlišením.

Předposlední kapitole obsahuje zdrojový kód v programu Matlab. Přesněji řečeno, dva algoritmy, jeden pro odstranění lokálního zkreslení a druhý pro zvýšení lokálního rozlišení. Vytvořením jednoduchého grafického rozhraní bude umožněno obsluze zasáhnout do běhu programu a změnit metodu nebo jednotlivé parametry tak, aby byl výsledek co možná nejlepší. Obsluha bude muset vědět, kdy použít jednotlivé metody a pokusit se odhadnout předběžný výsledek.

Poslední kapitola je věnována ukázkám jednotlivých metod na konkrétních případech. Jsou uvedeny jednotlivé metody a parametry, která byly využity při odstraňování lokálního rozmazání, nebo při zvyšování rozlišení. V závěru je shrnuta celá diplomová práce. Jsou zhodnoceny jednotlivé výsledky obou metod, a uvedeny návrhy na vylepšení jednotlivých algoritmů.

2 Cíle práce

- Seznámení se základními problémy zkreslení obrazu vlivem pohybu. Co je pohybová neostrost (motion blur), kdy vzniká a jak se jí vyvarovat.

- Zjistit jednotlivé metody pro odstranění lokálního rozlišení, jaké jsou jejich vstupní parametry a jaký mají vliv na výsledný obraz.

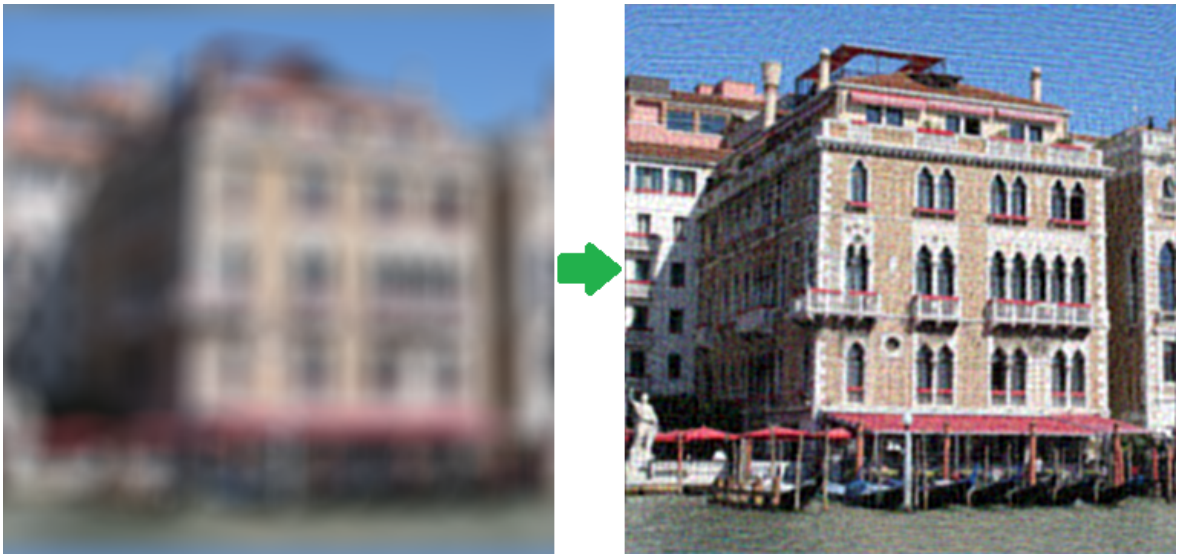
- Vysvětlit metodu super-resolution imaging. Jak funguje, kdy ji lze použít a aplikovat na některé zkušební snímky.

- Vytvoření algoritmů v programu Matlab, které budou provádět jednotlivé kroky pro odstranění zkreslení v lokálních částech obrazu a zvyšovat rozlišení pomocí metody super-resolution imaging. Vymyslet tento algoritmus tak, aby do něj mohla zasahovat zkušená obsluha. Změnou jednotlivých parametrů bude docíleno co možná nejlepšího výstupu.

3 Zkreslení obrazu

Restaurace zkreslených obrazů je jedním z nejzajímavějších a nejdůležitějších problémů zpracování obrazu jak z teoretického, tak i z praktického hlediska. Existují speciální případy rozmazání obrazu: rozmazání obrazu důsledkem nesprávného zaostření a rozmazání důsledkem pohybu kamery, nebo pohybu natáčeného (foceného) objektu. Tyto vady obrazu jsou pro každého dobře viditelné, ale velice těžko opravitelné. U jiných vad obrazu jako jsou: šum, nesprávná expozice atd., jsou v jednotlivých editačních programech příslušné nástroje. [1] Proč tedy nejsou v těchto programech příslušné nástroje pro korekci rozmazání a rozostření (s výjimkou masky pro zaostření)? Možná je to nemožné.

Ve skutečnosti je to možné, vývoj příslušné matematické teorie začal přibližně před 70 lety, ale stejně jako u jiných algoritmů pro zpracování obrazu, tak i algoritmy pro rozostřené obrazy se začaly široce používat až v posledních době. Níže je ukázán příklad rozostřeného obrazu a jeho následná korekce. [1]



***Obr. 3.1** - Ukázka korekce rozostřeného obrazu pomocí matematického algoritmu, převzato [1]*

3.1 Pohybová neostrost (motion blur)

Pohybová neostrost (roztřesení, rozmazání) je zachycení pohybujících se objektů ve fotografii, nebo na sekvenci snímků, například ve filmu nebo animaci. Neostrý záběr vzniká při záznamu jednoho snímku, nebo nahrávání obrazu, kdy dochází k pohybovým změnám.

Pohybové změny mohou být v důsledku rychlého pohybu snímaného objektu nebo fotoaparátu (kamery) během dostatečně dlouhé expozice. Může se jednat o nežádoucí efekt, nebo o autorův záměr.

Pojmy pohybová neostrost a roztřesení se významově liší. Roztřesení se používá v případě chvění fotoaparátu během snímání. Pojem pohybová neostrost značí pohyb objektů v záběru během snímání. Tyto dva pojmy jsou blíže vysvětleny v následujících kapitolách.

3.1.1 Zkreslení obrazu vlivem pohybu snímaného objektu

Jak už bylo výše zmíněno, toto zkreslení označujeme jako pohybovou neostrost. Vzniká při snímání pohybující se ho objektu. Veliký vliv má na tento problém rychlost závěrky.

***Obr. 3.2** – ukázka použití malé rychlosti závěrky a výsledný motion blur v obraze, převzato [http://www.photographymad.com/pages/view/shutter-speed-a-beginners-guide]*

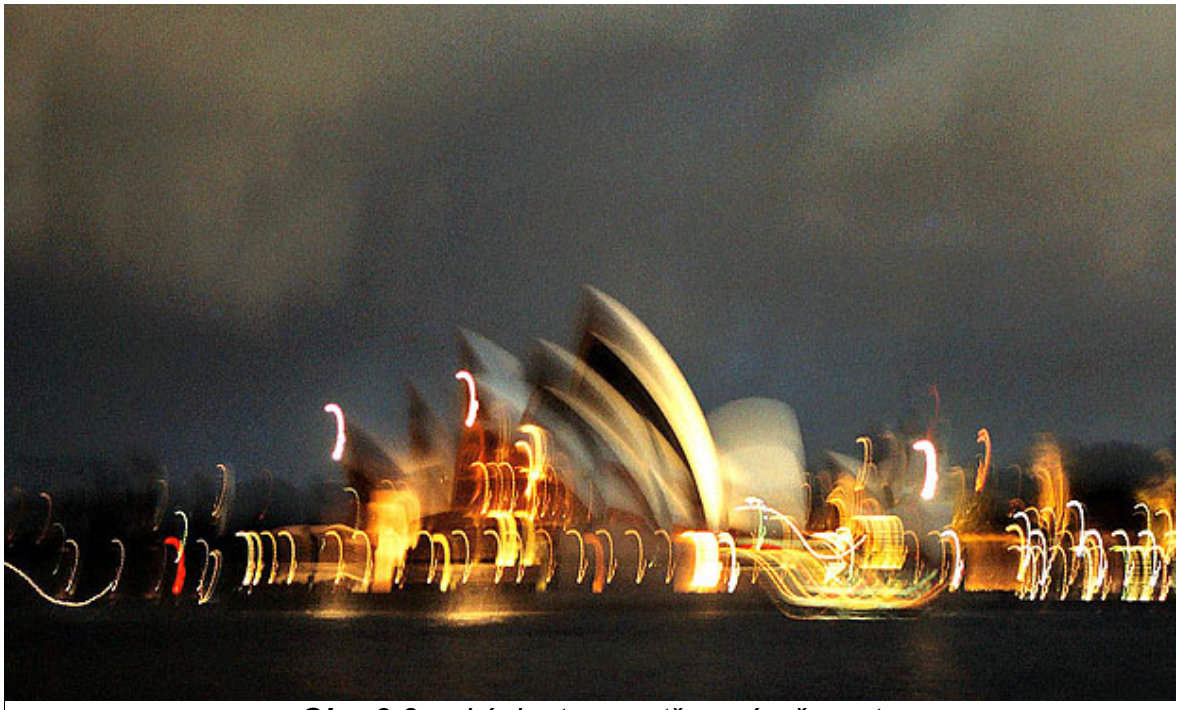
Uvnitř fotoaparátu přímo před snímačem je malá klapka, která se nazývá závěrka. Při pořizování fotografií, se závěrka otevírá a zavírá, aby se mohlo světlo dostat na snímač a tím se vytvořila fotografie. Rychlost závěrky popisuje, jak moc rychle (pomalu) se bude závěrka otevírat nebo zavírat. Při použití malé rychlosti závěrky, se bude snímaný objekt pohybovat

přes snímek, zatímco bude závěrka otevřena. To má za následek, že se snímaný objekt objeví jako rozmazaný pruh ve výsledném obraze. [2]

Tomuto se dá vyhnout pomocí vyšší rychlostí závěrky. To má za následek, že se snímaný objekt bude pohybovat méně, zatímco bude závěrka otevřena, čímž se snižuje efekt rozmazání. Dříve než se rozhodneme zvýšit rychlost závěrky jak jen to je možné, musíme zvážit, zda chceme úplně eliminovat rozmazání pohybem. Je to skvělý způsob, jak vyjádřit rychlost nebo pohyb ve scéně.

3.1.2 Zkreslení obrazu vlivem pohybu kamery

Zkreslení obrazu vlivem pohybu kamery nazýváme roztřesení. Rozostření vzniká držíme-li fotoaparát pouze v rukou. Nezáleží na tom, jak moc si myslíme, že dokážeme pevně udržet fotoaparát. Nikdy nemůžeme stát perfektně bez hnutí a i malý pohyb rukou se pak objeví na fotografii jako roztřesení nebo nedostatek ostrosti. [2]



***Obr. 3.3** - ukázka tzv. roztřesení, převzato [http://www.photographymad.com/pages/view/shutter-speed-a-beginners-guide]*

Chvění fotoaparátu se lze vyhnout použitím vyšší rychlosti závěrky. Toto opatření proti roztřesení je patrnější při použití objektivů s delší ohniskovou vzdáleností. Čím delší objektiv, tím více musíme zvýšit rychlost závěrky, aby se zabránilo chvění fotoaparátu. Při nastavování rychlosti závěrky platí pravidlo, že minimální rychlost závěrky je 1 / ohnisková

vzdálenost. Při výpočtu je důležité použít efektivní ohniskovou vzdálenost objektivu. Efektivní ohnisková vzdálenost je ohnisková vzdálenost objektivu vynásobená crop faktorem fotoaparátu. [2]

Nejjednodušší způsob jak zabránit zkreslení obrazu vlivem pohybu kamery je ke kameře využít stativ, monopod nebo jakoukoliv pevnou podložku a nedržet fotoaparát pouze v rukách. Tento jednoduchý způsob, ale nemusí být vždy užitečný, vždy je lepší fotoaparát správně nastavit. Moderní fotoaparáty a kamery využívají integrovanou stabilizaci na snímači, nebo stabilizaci integrovanou v objektivu. Tyto stabilizace začínají být v posledních letech velice účinné.

4 Metody pro odstranění lokálního zkreslení

Mnoho lidí si myslí, že rozostření obrazu je nevratná operace a obsažená informace je nadobro ztracena, protože se každý pixel promění v bod, všechno smíchá a v případě velkého kruhu rozostření získáme plochou barvu celého snímku. To ale není zcela pravda, všechny informace, které jsou šířeny v souladu s některými pravidly a mohou být jednoznačně obnovena za určitých předpokladů. Výjimkou jsou pouze hranice obrazu, jehož šířka je rovna poloměru rozostření, v tomto případě pak obnova není možná. [3]

Ukažme si to pomocí malého příkladu pro jednorozměrný případ. Předpokládejme, že máme řadu bodů s následujícími hodnotami: $x_1 \mid x_2 \mid x_3 \mid x_4 \mid \ldots$ – **zdrojový obraz**. [3] [4]

Po rozmazání je hodnota každého pixelu přidána k hodnotě pixelu nalevo: $x_i' = x_i + x_{i-1}$. Za normálních okolností, je také nutné tyto hodnoty vydělit dvěma, ale dělení vynecháme kvůli zjednodušení příkladu. Z důsledku toho máme rozmazaný obraz s následujícími hodnotami pixelu: $x_1 + x_0 \mid x_2 + x_1 \mid x_3 + x_2 \mid x_4 + x_3 \mid \ldots$ – **rozmazaný obraz**. [3] [4]

Nyní se budeme snažit rozmazaný obraz obnovit, budeme návazně odečítat hodnoty podle následujícího schématu. První pixel od druhého, výsledek druhého pixelu od třetího, výsledek třetího od čtvrtého a tak dále a dostaneme následující: $x_1 + x_0 \mid x_2 - x_0 \mid x_3 + x_0 \mid x_4 - x_0 \mid \ldots$ – **obnovený obraz**. [3] [4]

Výsledkem je, že namísto rozmazaného obrazu máme zdrojový obraz, kde se ke každému pixelu přidá neznámá konstanta x_0 s alternativním znakem. To je mnohem lepší, jelikož si tuto konstantu můžeme zvolit podle vizuálního cítění. Lze předpokládat, že tato konstanta je přibližně stejná jako hodnota x_1. Můžeme ji automaticky zvolit s takovým kritériem, kdy se hodnoty sousedních obrazových bodů mění co možná nejméně. Ale vše se změní jakmile se přidá šum (ten je u reálných obrazů vždy přítomen). V případě výše popsaného schématu se bude v každé iteraci akumulovat hodnota šumu do celkové hodnoty. Tato skutečnost může nakonec produkovat nepřijatelné výsledky, ale jak již víme, obnova je možná i s použitím takto jednoduché metody. [3] [4]

4.1 Model procesu rozmazání

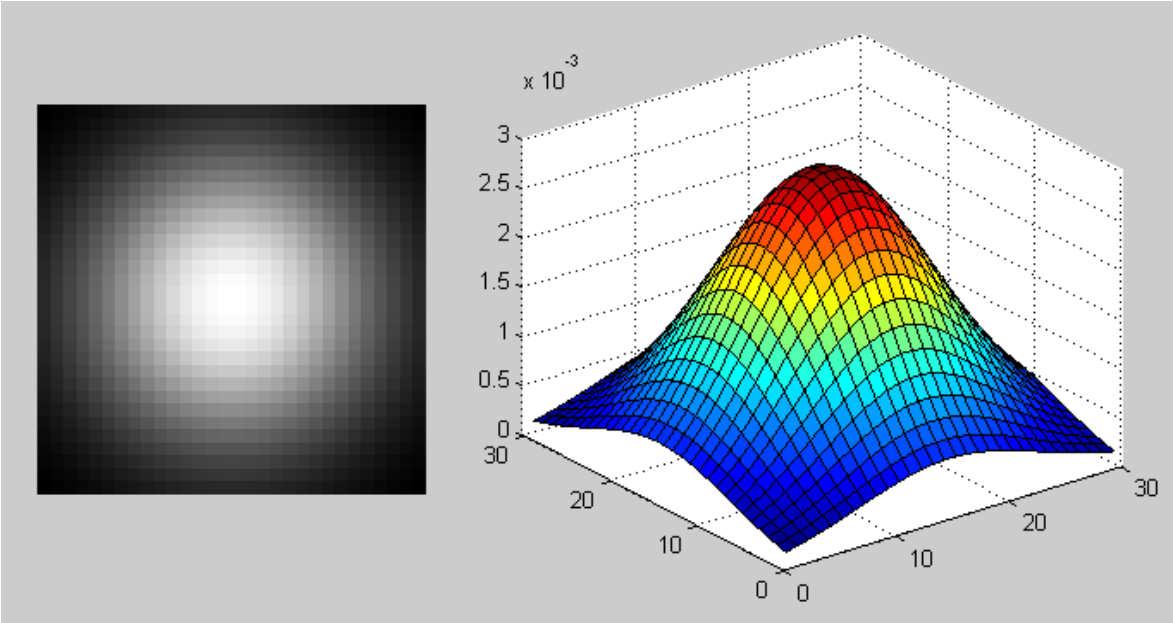
Nyní se podívejme na bližší popis tohoto rozmazání a obnovu procesu. Budeme se zabývat obrazy pouze ve stupních šedi, za předpokladu, že pro plný barevný rozsah, stačí opakovat příslušné kroky pro každý barevný kanál RGB. Ukažme si následující definice: *f (x,y)* – zdrojový obraz (nerozmazaný), *h (x,y)* – funkce rozmazání, *n (x,y)* – přidaný šum a *g (x,y)* – výsledný rozmazaný obraz. Následujícím způsobem můžeme vytvořit model procesu rozmazání:

$$g(x,y) = h(x,y) * f(x,y) + n(x,y) \tag{4.1}$$

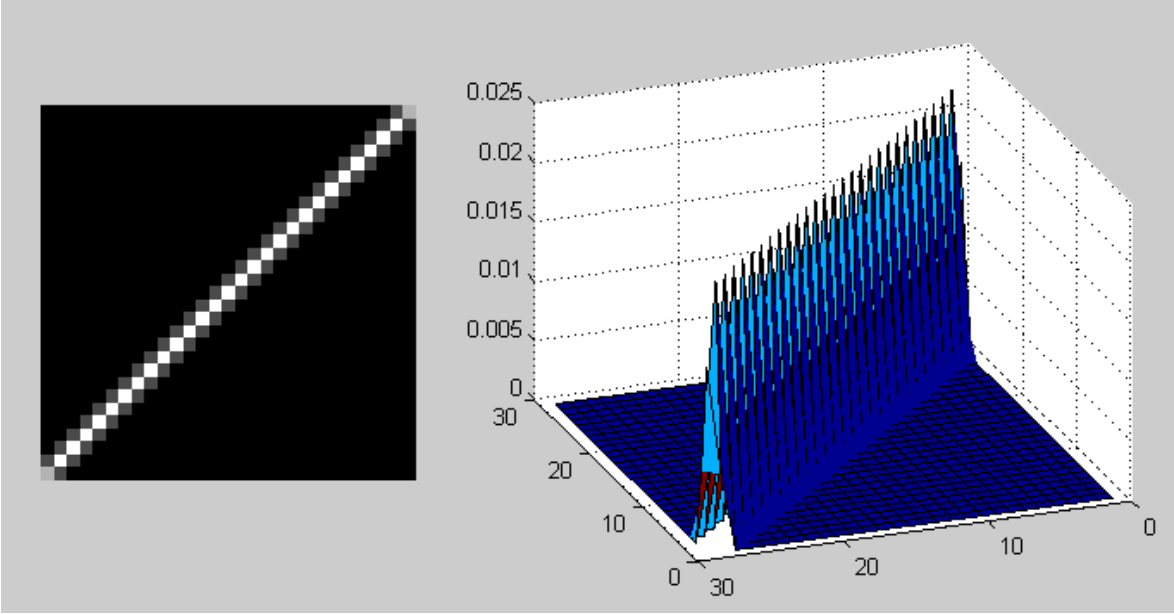
Hlavní úkol obnovy rozmazaného obrazu spočívá v nalezení nejlepší aproximace *f'(x,y)* do zdrojového obrazu. Podívejme se na jednotlivé komponenty podrobněji. Funkce *f (x,y)* a *g (x,y)* jsou zcela jasné a není potřeba je blíže vysvětlovat. Na funkci *h (x,y)* se podíváme podrobněji. V procesu rozmazání se každý pixel zdrojového obrazu změní v bod v případě rozostření a do úsečky v případě, že se jedná o rozostření v důsledku pohybu. Tento problém lze popsat i jinak. Každý pixel rozmazaného obrazu je sestaven z několika pixelů z okolí zdrojového obrazu. Všechny tyto pixely se vzájemně překrývají, což ve skutečnosti vede k rozmazání obrazu. Zásada, podle níž se šíří jeden pixel se nazývá funkce rozmazání neboli PSF (point spread function). Velikost této funkce je nižší než velikost samotného obrazu. Uvažujeme-li, že v prvním ukázkovém příkladu, je velikost funkce dva, protože každý výsledný pixel se skládá ze dvou pixelů. [3] [4]

4.1.1 Funkce rozmazání (PSF)

Nyní se podíváme, jak taková typická funkce rozmazání vypadá. Využijeme programu Matlab, který se stal již standardem pro takovéto účely. Obsahuje vše potřebné pro nejrůznější experimenty se zpracováním obrazu a umožňuje se soustředit na algoritmy, protože přesouvá všechnu rutinní práci do funkčních knihoven. Ale vraťme se k PSF, zde je několik příkladů.



***Obr. 4.1** – funkce PSF v případě využití Gaussovo funkce: fspecial(`gaussian`,30,8), převzato [3]*



***Obr. 4.2** - funkce PSF v případě využití motion blur funkce: fspecial(`motion`,40,45), převzato [3]*

Proces aplikování funkce rozmazání na jinou funkci (jako v případě na obrázku 4.1 a 4.2) se nazývá konvoluce. Některé oblasti zdrojového obrazu konvolují do jednoho pixelu rozmazaného obrazu. To je označováno pomocí operátoru ¨*¨, nezaměňovat s jednoduchým násobením. Matematicky, pro obraz *f* o rozměrech M x N a funkci rozmazání *h* o rozměrech m x n můžeme zapsat takto:

$$g(x,y) = h(x,y) * f(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} h(i,j) f(x+i, y+j) \qquad (4.2)$$

Kde *a = (m – 1) / 2 a b = (n – 1) / 2*. Tento proces, který je opakem konvoluce se nazývá dekonvoluce a řešení je zcela neobvyklé. [3] [4]

Máme několik způsobů, jak PSF funkci získat. Využijeme metody Wiener filter, která bude blíže popsána níže v kapitole 4.2.1. Existuje mnoho jiných přístupů, ale všechny dávají přibližně stejné výsledky. Níže uvedené popisy platí i pro jiné metody dekonvoluce. Hlavním úkolem, je získat PSF funkci, to lze provést několika způsoby: [5]

a) **Modelování.** Je velice časově náročné, protože moderní objektivy se skládají z desítek různých čoček a optických prvků, z nichž některé nemají sférický tvar, přičemž každý typ skla má své jedinečné vlastnosti lomu světla s různými vlnovými délkami. V důsledku toho, je správný výpočet šíření světla v tak složité optické soustavě s přihlédnutím k odrazům atd., téměř nemožné získat. Toto řešení se nabízí spíše pro vývojáře moderních čoček. [5]

b) **Přímé pozorování.** Pokud máme černé pozadí a přidáme na něj jeden bílý bod, poté obraz s požadovanou hodnotou rozostříme, budeme mít přímý výhled na PSF. Zdá se, že je to jednoduché, ale existuje spousta zvláštností a nuancí. [5]

c) **Výpočet nebo nepřímé pozorování.** Podíváme-li se na vzorec (4.4) procesu deformace, zjistíme, že potřebujeme dostat *H(u,v)*. Musíme mít zdrojový obraz *F(u,v)* a rozmazaný obraz *G(u,v)*. Poté když vydělíme Fourierův rozmazaný obraz Fourierovo zdrojovým obrazem, dostaneme požadovanou PSF. [5]

4.1.2 Model šumu

Bere se v úvahu pouze poslední sčítanec *n(x,y)* ve vzorci (4.1), který je zodpovědný za šum. U digitálních snímačů může být mnoho důvodů pro šum, ale mezi základní se řadí termální vibrace (tzv. Brownův pohyb) a temný proud. Úroveň šumu také závisí na řadě faktorů, jako je například hodnota ISO, typ snímače, velikost pixelu, teplota, atd. Ve většině

případů je zde Gaussův šum a ten je stanoven dvěma parametry, průměrem a disperzí. Gaussův šum je část, která nekoreluje s obrazem a není závislá na souřadnicích pixelu. Poslední tři předpoklady jsou velice důležité pro další práci. [3] [4]

4.1.3 Konvoluční teorém

Zpět k obnově. Musíme obrátit konvoluci s ohledem na šum. Ze vzorce (4.2) lze vidět, že není tak snadné dostat *f(x,y)* z *g(x,y)*, pokud budeme počítat přímočaře, vede tento problém na obrovské množství soustav rovnic. Zde si vypomůžeme Fourierovo transformací. Podle tzv. konvolučního teorému operace konvoluce v prostorové doméně odpovídá pravidelnému násobení ve frekvenční doméně (kde se násobí prvek po prvku). Tomu odpovídá, že inverzní operace ke konvoluci je ekvivalentní k dělení ve frekvenční doméně a může být vyjádřena následujícím způsobem: [3] [4]

$$h(x,y) * f(x,y) \Leftrightarrow H(u,v)F(u,v) \tag{4.3}$$

Kde *H(u,v)* a *F(u,v)* jsou funkce Fourierovy transformace pro *h(x,y)* a *f(x,y)*. Takže proces rozmazání z rovnice (4.1) může být přepsán ve frekvenční doméně jako: [3] [4]

$$G(u,v) = H(u,v)F(u,v) + N(u,v) \tag{4.4}$$

4.2 Existující přístupy k dekonvoluci

Existují postupy, které berou v úvahu přítomnost šumu v obraze.

4.2.1 Inverzní filtr

Vydělíme rovnici (4.4) *H(u,v)* a dostaneme toto:

$$\hat{F}(u,v) = F(u,v) + \frac{N(u,v)}{H(u,v)} \tag{4.5}$$

Tento jev se nazývá inverzní filtrování, ale v praxi příliš nefunguje. V případě, že funkce *H(u,v)* dává hodnoty, které se blíží nebo jsou rovny nule, tak celý zlomek bude v rovnici dominovat. Spektrum Fourierovy transformace má dvě složky, amplitudu a fázi.

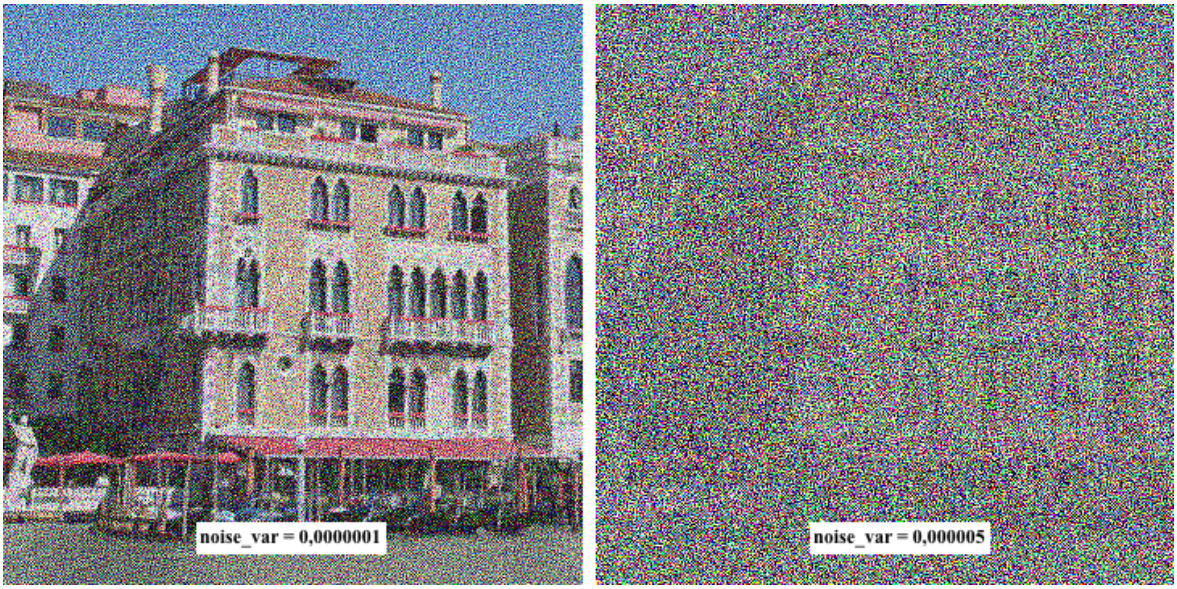
Amplitudové spektrum je znázorněno na logaritmické stupnici. Maximální hodnoty jsou v centru a směrem od centra se rychle snižují téměř k nule. Vzhledem k této skutečnosti funguje inverzní filtrování pouze v případě nulové nebo téměř nulové hodnoty šumu, jak je patrné z obrázku (4.3). Na zdrojový obrázek je použit následující skript v Matlabu: [3]

```
% Nahrani obrazu
I = im2double(imread('image_src.png'));
figure(1);
imshow(I);
title('Source image');

% Rozmazani obrazu
Blurred = imfilter(I,PSF,'circular','conv');
figure(2);
imshow(Blurred);
title('Blurred image');

% Pridani sumu
noise_mean = 0;
noise_var = 0.0;
Blurred = imnoise(Blurred,'gaussian',noise_mean,noise_var);

% Dekonvoluce
figure(3);
imshow(deconvwnr(Blurred, PSF, 0));
title('Result');
```



Obr. 4.3 *– ukázka rozdílu v hodnotách šumu v obrazu při inverzním filtrování, převzato [3]*

Jak je vidět na obrázku (4.3), tak i přidání velmi malého šumu způsobuje vážné rušení, které výrazně omezuje použití této metody. [3] [4]

4.2.2 Wiener filtr

Bere v úvahu, že obraz a zvuk jsou náhodné procesy a najde takovou hodnotu *f'* pro obraz bez zkreslení *f*, že kvadratická odchylka těchto dvou hodnot je minimální. Minimum této odchylky se dosáhne při funkci v kmitočtové doméně:

$$\hat{F}(u,v) = \left(\frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + \frac{S_\eta(u,v)}{S_f(u,v)}}\right) G(u,v) \tag{4.6}$$

Tento výsledek objevil Wiener v roce 1942. Funkce *S* označuje spektrum šumu (S_η) a zdrojového obrazu (S_f). Tyto hodnoty jsou velice zřídka známy, proto je jejich podíl nahrazen konstantou *K*, kterou lze přibližně označit jako poměr signálu k šumu. [3] [4]

4.2.3 Nejmenší počet čtverců (Tikhonova filtrace, Tikhonov regularization)

Myšlenka spočívá v tvorbě matice s následným řešením příslušného úkolu optimalizace. Výsledek rovnice může být zapsán takto: [3] [4]

$$\hat{F}(u,v) = \left(\frac{H^*(u,v)}{(H(u,v)|^2 + y|P(u,v)|^2}\right) G(u,v) \tag{4.7}$$

4.2.4 Metoda Lucy-Richardson

Další zajímavý přístup nabídli nezávisle na sobě Richardson (rok 1972) a Lucy (rok 1974). Největší rozdíl od předešlých přístupů je, že tento přístup je nelineární. Další vlastnost je, že je metoda iterativní. Hlavní myšlenka spočívá v použití metody maximální věrohodnosti, u níž se předpokládá, že je obraz vystaven Poissonovu rozdělení. Kalkulační vzorce jsou velice jednoduché, vše se děje v prostorové doméně a nevyužívá se Fourierovy transformace.

$$\hat{f}_{k+1}(x,y) = \hat{f}_k(x,y)\left[h(-x,-y) * \frac{g(x,y)}{h(x,y) * \hat{f}_k(x,y)}\right] \tag{4.8}$$

Symbol * opět značí operaci konvoluce. Tato metoda je široce využívána v programech pro zpracování astronomických snímků. Využívání dekonvoluce u těchto snímků je

standardem. Výpočetní náročnost této metody je poměrně vysoká, závisí na počtu iterací a může trvat mnoho hodin až dní. [3] [4]

4.2.5 Blind deconvolution

Ve všech předchozích metodách se předpokládalo, že funkce rozmazání PSF je známa. V praxi toto není pravda, většinou známe jen přibližnou PSF podle viditelných zkreslení. Blind deconvolution bere tuto skutečnost v úvahu. Princip je vcelku jednoduchý, je vybrána aproximace PSF, poté se provádí dekonvoluce podle některé z metod. Po dekonvoluci se podle zvoleného kritéria porovnává stupeň kvality. Na základě tohoto stupně je PSF vyladěna a iterace se opakuje, dokud není dosaženo požadovaného výsledku. [3] [4]

4.2.6 Ukázka jednotlivých metod

Do zdrojového obrazu byl přidán šum a následně byl obraz rozmazán dle zadaných parametrů. Na takto rozmazaný obraz byly jednotlivě aplikovány výše zmíněné metody pro odstranění rozmazání. Zdrojový obraz, rozmazaný obraz se šumem a výsledky jednotlivých metod jsou vidět na následujícím obrázku (4.4). Zdrojový kód z Matlabu je přiložen dále. [3] [4]

```
% Nahrani obrazu
I = im2double(imread('image_src.png'));
figure(a); imshow(I); title('Source image');

% Rozmazani obrazu
PSF = fspecial('disk', 15);
Blurred = imfilter(I, PSF,'circular','conv');

% Pridani sumu
noise_mean = 0;
noise_var = 0.00001;
Blurred = imnoise(Blurred,'gaussian',noise_mean,noise_var);
figure(b);
imshow(Blurred);
title('Blurred image');
estimated_nsr = noise_var / var(Blurred(:));

% Obnova obrazu
figure(c),
imshow(deconvwnr(Blurred,PSF,estimated_nsr)),
title('Wiener filter');

figure(d);
imshow(deconvreg(Blurred,PSF));
title('Tikhonov regularization');
```

```
figure(e);
imshow(deconvblind(Blurred,PSF,100));
title('Blind deconvolution');

figure(f);
imshow(deconvlucy(Blurred,PSF,100));
title('Lucy-Richardson filter');
```

Obr. 4.4 *– ukázka použití jednotlivých metod, (a) – zdrojový obraz, (b) – rozmazaný obraz se šumem, (c) – Wiener filtr, (d) – Tikhonov regularization, (e) – Blind deconvolution, (f) – Lucy-Richardson filtr, převzato [3]*

5 Zvýšení lokálního rozlišení (super-resolution imaging)

Tato kapitola se zabývá problémem fixační informace z několika pohledů na scénu, aby se vytvořil nový pohled na scénu s větším prostorovým rozlišením než je v jednotlivých zobrazeních. Světlo emitované ze scény se promítá na snímač kamery a produkuje vzor intenzity záření, který je časově integrovaný a je kvantován ke generování obrazu. Kromě toho je obraz často rozmazaný vzhledem ke špatnému zaostření nebo pohybu snímaného objektu. Snažíme se zvrátit tyto degradace k provedení odhadu s vysokým rozlišením reprezentujícího intenzity ve scéně. Za předpokladu Lambertianeho ploch (povrch, který se jeví jako rovnoměrně jasný ze všech směrů pohledu a odráží světlo kompletně do všech směrů) a rovnoměrného osvětlení, to je ekvivalentní k odhadnutí povrchového albeda (měřítko odrazivosti nebo optického jasu). Tyto techniky jsou často označovány jako super-resolution (super rozlišení). [6]

Vzhledem k typům obrazového snímání, na které se techniky super-resolution obecně aplikují, jsou dvě kategorie: [6]

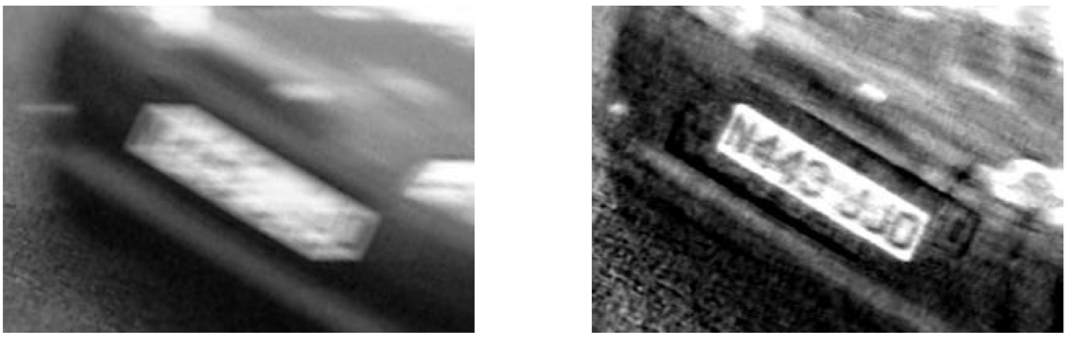
- **Hlavní degradací je aliasing:** Sekvence, ve kterých je optické nebo pohybové rozmazání zanedbatelné, ale požadované detaily textur nejsou k dispozici, z důvodu příliš nízké prostorové hustoty vzorkování kamery.

- **Hlavní degradací je rozostření:** Sekvence, ve kterých je hustota vzorkování naprosto dostačující pro zachycení požadovaného údaje, ale je v nich optické nebo pohybové rozmazání.

Ve většině literatury o super-resolution se pod pojmem rozlišení rozumí skutečný počet pixelů v obraze. Cílem je pak získat obraz vysokého rozlišení s vyšší hustotou pixelů, než má obraz nízkého rozlišení. Typický je tzv. zoom faktor, kde se zdvojnásobí pixely ve směrech X a Y. Vyprodukuje se obraz vysokého rozlišení se čtyř násobkem hustoty pixelů než v obrazu nízkého rozlišení. [6]

Tzv. pixel counting (počítání pixelů) je snadno použitelný pro zvýšení rozlišení u případů, kdy za degradaci může aliasing. U případů s rozostřením je zpravidla nutné zvýšit

hustotu pixelů. Požadované detaily textur jsou odhaleny odstraněním rozmazání. Příklad této myšlenky je znázorněn na obrázku 5.1, který ukazuje pohybově rozmazaný obraz auta s nečitelnou poznávací značkou. Nerozmazaná verze představuje dramatické zlepšení v čitelnosti, ačkoliv nebyl zvýšen počet pixelů. V takových případech se využívá vícero snímků ke snížení citlivosti šumu v rekonstrukci obrazu jako u rekonstrukce jednoho obrazu. [6]



***Obr. 5.1** – (vlevo) pohybově rozmazaná SPZ, (vpravo) po odstranění rozmazání, vnímané rozlišení je mnohem lepší, i když se počet pixelů nesměnil, převzato [5]*

5.1 Zobrazovací model z více pohledů

Chceme-li obnovit informaci z několika pohledů (snímků), musíme nejprve stanovit proces, při kterém jsou jednotlivé snímky generovány. Do použitého modelu musíme započítat tyto následující složky. [6]

- **Pohybový model** je rovinný. Jinými slovy, relativní pohyb mezi scénou a kamerou vyvolává projektivní transformace (homography) mezi pohledy.

- **Scéna** je omezena pohybovým modelem a to buď rovinným Lambertianovým povrchem, nebo statickou scénou snímanou kamerou, která se otáčí kolem své optické osy a/nebo při použití zoomu.

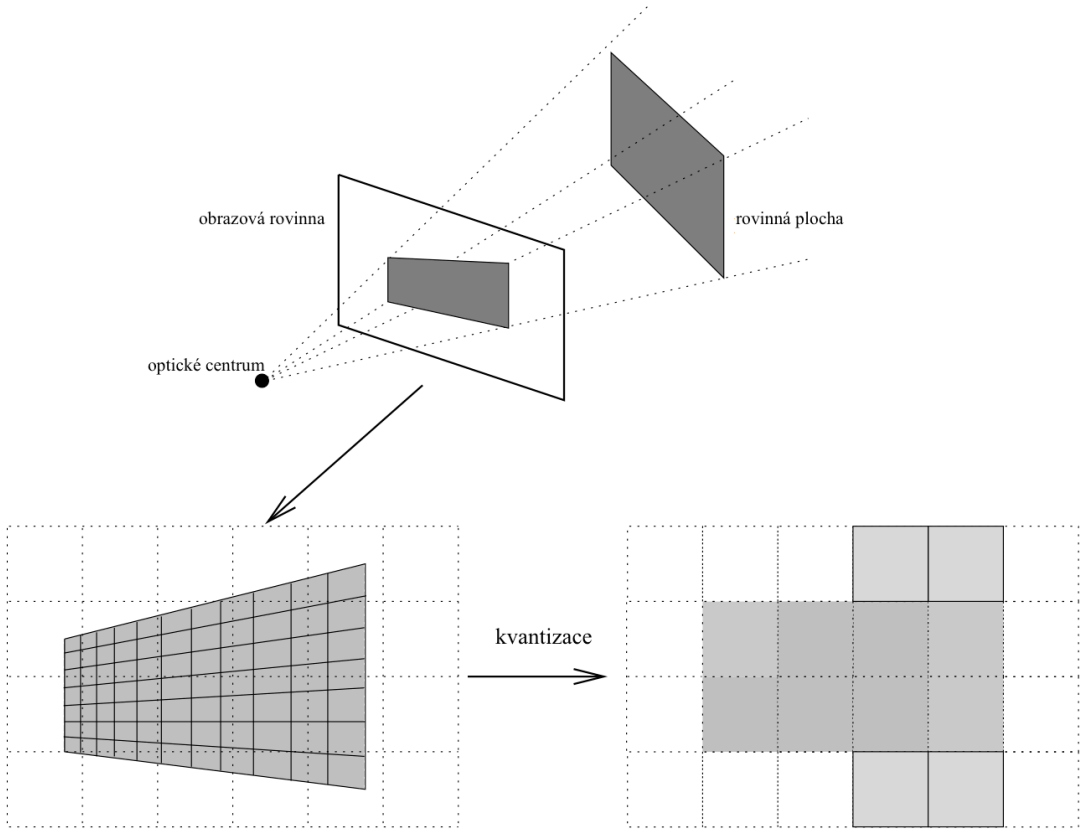
- **Světlo**. Předpokládá se, že je stejné po celé ploše. Může se konstantně měnit, zatím co se povrch a/nebo kamera pohybují.

- **Snímač kamery** je lineární zařízení, které obsahuje pole stejně vzdálených prvků. Každý z nich integruje zářivou energii dopadající na jeho povrch.

- **Geometrie kamery**. Uvažuje se, jako dírková komora. To znamená, že scéna je promítnuta v obrazové rovině kamery pomocí perspektivní projekce, což je zvláštní případ projektivní transformace.

- **Optika kamery** může zapříčinit rozmazání v důsledku špatného zaostření. Rozostření se předpokládá stejné po celém snímači kamery a proto je modelováno jako lineární konvoluce prostorově invariantní PSF funkce se zářením přeneseném na snímač kamery.

- **Šum obrazu** je prostorově nekorelované, izotropní normálové rozdělení se středem v nule a konstantním rozptylem.



***Obr. 5.2** – (nahoře) kamera zachycující rovinnou scénu, (vlevo) scéna je promítána pomocí perspektivní projekce na CCD matici, (vpravo) prvky matice integrují energii záření na svém povrchu a vytváří prostorově kvantovaný obraz scény, převzato [6]*

Prostorové průměrování provedené každou buňkou snímače, může být modelováno jako konvoluce s obdélníkovým oknem. [7] Vliv optického rozmazání je obvykle modelován jako

konvoluce s kruhovou „top-hat“ funkcí. [7] Modelujeme kombinovaný účinek těchto dvou degradací, jako konvoluci s izotropním Gaussovo rozložením. [6]

Předpoklad lineárního mapování mezi zářením scény a hodnotou pixelu není v praxi zcela pravdivý, protože je často prováděna komprese dynamického rozsahu v systému pro zachycení videa. Podrobnosti nelineárního mapování nejsou k dispozici v tomto zařízení, přestože byly navrženy způsoby, jak je odhadnout pomocí kalibračních snímků.[8][9] Nicméně, mapování jsou obvykle poměrně velkou částí lineárního rozsahu střední intenzity, což je skutečnost, kterou pozoroval Robertson a spol. v jejich práci na zlepšení dynamického rozsahu s použitím vícenásobné expozice. [11][6]

5.1.1 Diskretizace obrazového modelu

Předpokládá se, že intenzity scény jsou spojité dvojrozměrné funkce. Z množiny pozorovaných obrazů s nízkým rozlišením, byl vyroben jediný obraz s vysokým rozlišením na základě následujícího modelu:

$$g_n(x,y) = \alpha_n s\downarrow\left(h(u,v) * \bar{f}\left(\tau_n(x,y)\right)\right) + \beta_n + \eta(x,y) \tag{5.1}$$

$\bar{f}$ - obraz s vysokým rozlišením

g_n - n-tý pozorovaný obraz s nízkým rozlišením

τ_n - geometrická transformace n-tého obrazu

h - PSF funkce

$s\downarrow$ - podvzorkování o faktor S

α_n, β_n - skalární parametry osvětlení

η_n - pozorovaný šum

Transformace τ je projektivní, funkce PSF h se předpokládá že je lineární a prostorově invariantní a šum η se s Gaussovo rozložením blíží nule. Po diskretizaci může být model vyjádřen pomocí matic, jako:

$$\boldsymbol{g}_n = \alpha_n \mathrm{M}_n \bar{\boldsymbol{f}} + \beta_n + \boldsymbol{\eta}_n \tag{5.2}$$

kde je vektor $\overline{f}$ lexikografická změna pořadí pixelů *f(x,y)* a lineární operátory τ_n *h* a $s\downarrow$ byly sloučeny do matice $\mathbf{M_n}$. Každý pixel nízkého rozlišení je dán váženým součtem super-resolution pixelů. Váha součtu je dána záznamovými parametry a tvarem PSF funkce. Tyto parametry kombinují optické a pohybové rozostření a prostorovou integraci. Modely o N snímcích jsou naskládány kolmo k vytvoření lineárního systému:

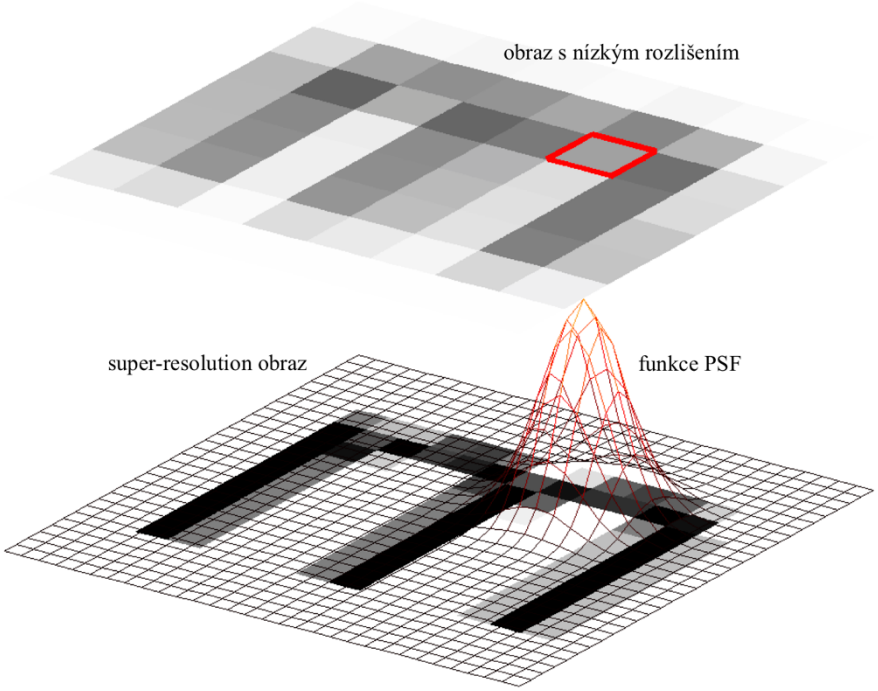
$$\begin{bmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_{N-1} \end{bmatrix} = \begin{bmatrix} \alpha_0 & & & \\ & \alpha_1 & & \\ & & \ddots & \\ & & & \alpha_{N-1} \end{bmatrix} \begin{bmatrix} \mathrm{M}_0 \\ \mathrm{M}_1 \\ \vdots \\ \mathrm{M}_{N-1} \end{bmatrix} \overline{f} + \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{N-1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\eta}_0 \\ \boldsymbol{\eta}_1 \\ \vdots \\ \boldsymbol{\eta}_{N-1} \end{bmatrix} \tag{5.3}$$

Z důvodu jednoduchosti se pixely *f* se super-resolution nazývají **super-pixels**, aby se odlišily od pixelů s nízkým rozlišením. [6]

5.1.2 Výpočet prvků matice $\mathbf{M_n}$

V této části se zabýváme otázkou, jak přesně vypočítat hodnoty v matici $\mathbf{M_n}$. Tento problém je ve většině literatury přehlížen.

Podle zobrazovacího modelu hodnota každého pixelu s nízkým rozlišením je integrovanou oblastí geometricky deformovaného super-resolution obrazu váženého podle PSF funkce viz. obrázek (5.3). Každý řádek matice $\mathbf{M_n}$ je tedy diskretizace integrální rovnice (5.1) pro jeden pixel. [6]



***Obr. 5.3** – každý pixel je vážený součet několika super-pixelů, váha je určena geometrickou transformací tvaru a velikosti PSF funkce a také kvadraturním pravidlem použitém k diskretizaci generativního modelu [6]*

5.2 Možné přístupy k diskretizaci generativního modelu

Existují tři možné postupy k diskretizaci generativního modelu.

5.2.1 Metoda A

Deformovat super-resolution obraz do mezilehlého snímku za použití interpolačního schématu (bilineární nebo bicubické), konvoluce diskretizované verze PSF funkce a konečného mezivzorku. Mezilehlý snímek je v souladu s nízkým rozlišením, ale hustota pixelů je stejná nebo vyšší než je super-resolution obraz. To odpovídá rozkladu $\mathbf{M_n}$, který využívali Zomet a Peleg. [12] [6]

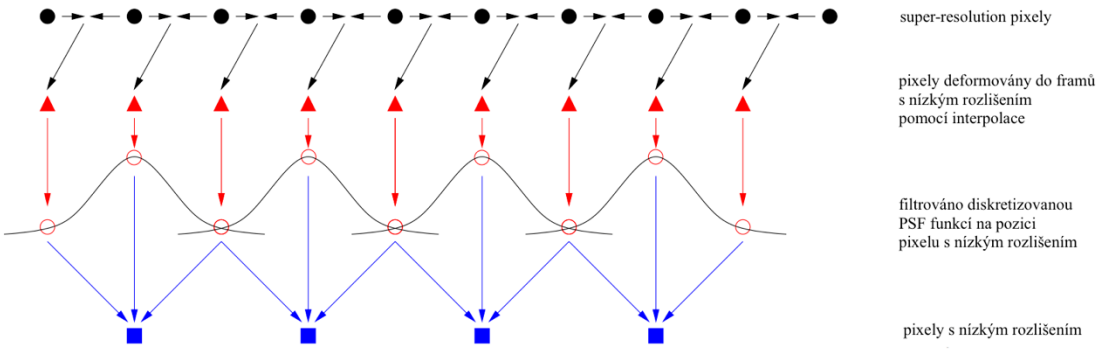
$$M_n = DHT_n \tag{5.4}$$

D - matice diskrétního podvzorkování (decimace)

H - matice konvoluce z diskretizace h(x,y)

T_n - geometrické deformace vhodnou interpolací

Operace matice-vektor $\boldsymbol{M_n} * f$ může být realizována za použití tří po sobě jdoucích lineárních transformací do super-resolution obrazu. Obrázek 5.4 ukazuje tento přístup na jednorozměrném signálu. Dvourozměrné schéma je znázorněno na obrázku 5.8. [6]



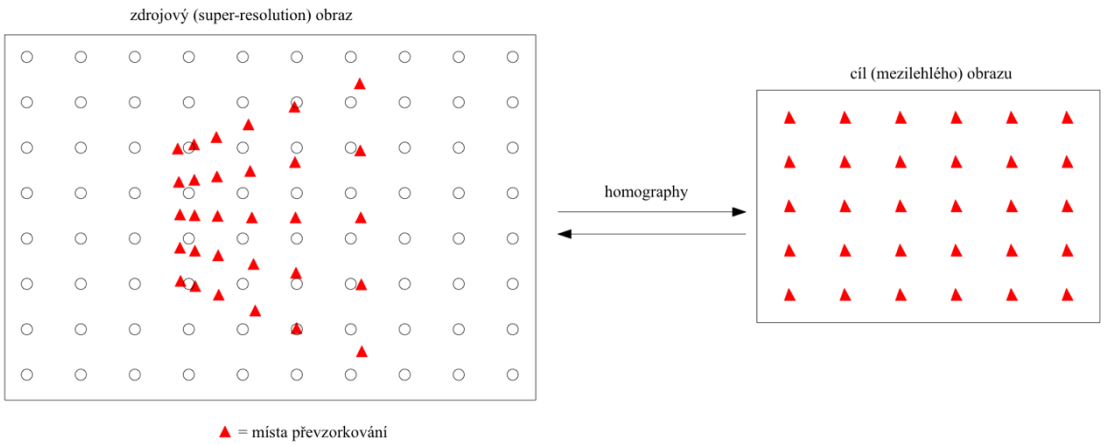
***Obr. 5.4** – metoda A [6]*

Super-resolution obraz se deformuje v závislosti na geometrické transformaci pohledu do mezilehlého snímku, pomocí schématu interpolace pro výpočet hodnoty v neceločíselných polohách. Mezilehlý snímek se filtruje přes diskretizovanou verzi PSF funkce v polohách, odpovídajících středu pixelu s nízkým rozlišením, pro vytvoření obrazu s nízkým rozlišením. Tento finální krok kombinuje konvoluci s podvzorkováním. [6]

Bohužel, toto schéma může dát špatnou aproximaci generativnímu modelu. Zejména v případech, kdy je perspektiva zdeformována, což má za následek velké rozdíly v celém obrazu. Abychom pochopili, proč tomu tak je, je dobré myslet na proces deformace obrazu jako na operaci převzorkování, ve které jsou převzorkovány stránky uvnitř obrazových bodů určených geometrickou transformací. Algoritmus deformace transformuje umístění pixelu z cílového snímku ke zdroji (super-resolution) snímku a stáhne zpět hodnotu podle schématu interpolace. Efekty deformace mohou způsobit převzorkování míst, které mají být řídce rozloženy v super-resolution obrazu, což vede k podvzorkování a frekvenčnímu aliasingu. V důsledku toho, mohou být jemné detaily v SR obrazu ztraceny nebo rozmazány ještě předtím, než byla použita PSF funkce. Tento problém je uveden na obrázku 5.5. Je možné tento problém zlepšit zvyšováním rozlišení meziobrazu, dokud není dosažena požadovaná hustota vzorkování v SR obrazu. [6]

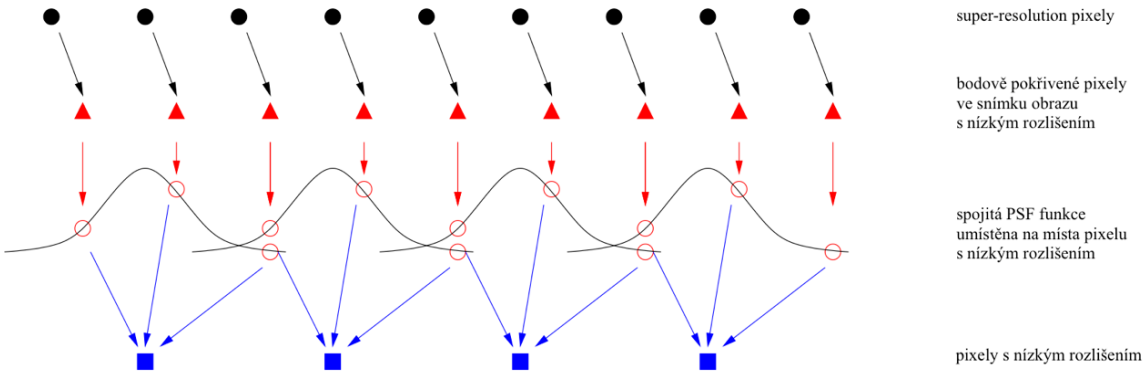
5.2.2 Metoda B

Super-pixely jako diskrétní body (bez interpolace) v souřadnicích snímku obrazu s nízkým rozlišením. Každý pixel s nízkým rozlišením je přidružený středu spojité hodnoty



***Obr. 5.5** – transformace bodů z cílového snímku do zdrojového snímku [6]*

PSF, která se používá k vážení podkladových deformovaných super-pixelů. Hodnota pixelu je dána jako vážený součet super-pixelů (součet hmotností normalizovaných na hodnotu 1). Obrázky 5.6 a 5.9 ilustrují tento systém v jednom a ve dvou rozměrech. [6]



***Obr. 5.6** – metoda B [6]*

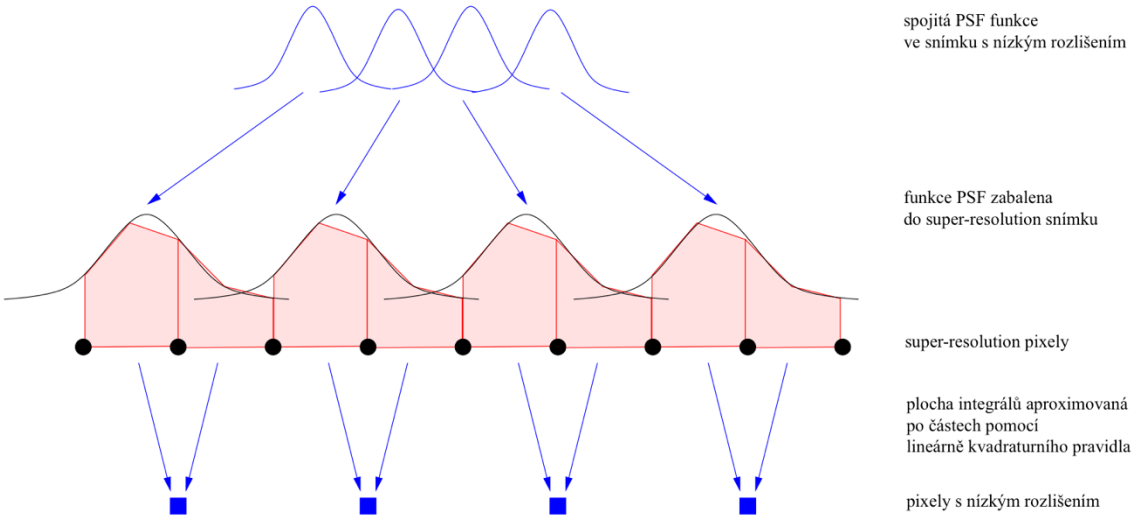
Super-pixely jsou bodově zdeformovány do snímku s nízkým rozlišením podle geometrické transformace pohledu, bez interpolace. Spojitá hodnota PSF je spojena s každým pixelem s nízkým rozlišením. PSF je zkrácena, řekněme třemi standartními odchylkami. Každý pixel je součet vážený PSF super-pixelů. [6]

Tento přístup má dva problémy. Za prvé, provádění bodové deformace super-pixelů do snímku s nízkým rozlišením, může být obtížné efektivně realizovat. Vymezujeme množinu těchto super-pixelů, které leží ve „vnímaném poli“ každé zkrácené PSF. Problém je v podstatě stejný jako určení počtu bodů, které leží v kružnici z několika tisíc nerovnoměrně rozložených bodů. Tato operace se musí opakovat v každém pixelu. [6]

Za druhé, je-li poměr hustoty pixelů *S* nízký (mezi 1 a 2), bude vnímané pole obsahovat pouze malé množství super-pixelů. V takovém případě není jednoduchý vážený součet hodnot pixelů dobrá aproximace požadovaného integrálu. [6]

5.2.3 Metoda C

Zabalená spojitá PSF funkce je spojena s každým pixelem uvnitř super-resolution snímku a vytváří rozumné předpoklady o spojitosti intra-pixelů v obrazu s vysokým rozlišením. Obrázky 5.7 a 5.10 objasňují tuto metodu.



***Obr. 5.7** – metoda C [6]*

Spojitá, zkrácená PSF funkce je spojena s každým pixelem s nízkým rozlišením. Funkce PSF je zabalena do super-resolution snímku. Předpokládá se, že super-resolution snímek a PSF funkce jsou bilineární a umožňují vypočtení integrálu velice přesně. [6]

Na tento přístup se podíváme blíže. Transformace Gaussovy PSF pod úroveň homography, která mapuje oblast mezi obrazem s nízkým rozlišením a SR obrazem, je

dosaženo tím, že počítá lokální afinní aproximaci ke každé pozici pixelu. Každý Gaussian je pak transformován do SR obrazu podle afinní transformace. [6]

Zapíšeme geometrickou transformaci bodů *(x,y)* v obrazu s nízkým rozlišením a bodů *(x',y')* v SR obrazu jako:

$$(\mathrm{x}', \mathrm{y}') = H(x, y) \tag{5.5}$$

požadovaná afinní transformace je dána aproximací Taylorovy řady prvního řádu kolem bodu (x_0,y_0):

$$H(x_0 + \delta x, y_0 + \delta y) \approx H(x_0, y_0) + \nabla H \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} \tag{5.6}$$

kde, Jacobian ∇H matice 2 x 2

$$\nabla H = \begin{bmatrix} \frac{\partial H_x}{\partial x} & \frac{\partial H_x}{\partial y} \\ \frac{\partial H_y}{\partial x} & \frac{\partial H_y}{\partial y} \end{bmatrix} \tag{5.7}$$

podle této transformace jsou parametry (μ, Σ), Gaussovy PSF funkce $G(\mu, \Sigma)$, s prostředkem (μ_x, μ_y) a kovarianční matice $\mathbf{\Sigma}$ transformovány do SR snímku jako

$$\mu' = H(\mu_x, \mu_y) \tag{5.8}$$

$$\Sigma' = (\nabla H)\Sigma(\nabla H)^{\mathrm{T}} \tag{5.9}$$

Aby bylo možné vypočítat integrál, budeme předpokládat, že SR obraz a PSF funkce lze aproximovat po částech bilineárního povrchu. Hodnota na pozici intra-pixelu je poté dána

$$f(x + \delta x, y + \delta y) = k_{00}f_{00} + k_{10}f_{10} + k_{01}f_{01} + k_{11}f_{11} \tag{5.10}$$

kde

$$f_{00} = f(x,y) \qquad f_{10} = f(x+1,y) \qquad f_{01} = f(x,y+1) \qquad f_{11} = f(x+1,y+1)$$

$$k_{00} = (1-\delta x)(1-\delta y) \quad k_{10} = \delta x(1-\delta y) \quad k_{01} = \delta y(1-\delta x) \quad k_{11} = \delta x \delta y$$

Integrál plochy f x h_{psf} přes čtvereční jednotku v SR obrazu je dán

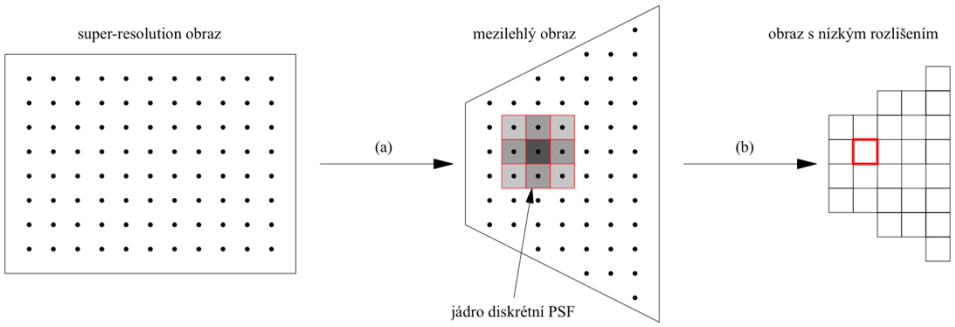
$$\int_0^1 \int_0^1 f_{xy} h_{xy} dxdy \tag{5.11}$$
$$= \frac{1}{9}(f_{00}h_{00} + f_{01}h_{01} + f_{10}h_{10} + f_{11}h_{11})$$
$$+ \frac{1}{18}\big(f_{00}(h_{10} + h_{01}) + f_{10}(h_{00} + h_{11})$$
$$+ f_{01}(h_{00} + h_{11}) + f_{11}(h_{10} + h_{01})\big)$$
$$+ \frac{1}{36}(f_{00}h_{11} + f_{10}h_{01} + f_{01}h_{10} + f_{11}h_{00})$$

V praxi je funkce PSF zkrácena na 3 standartní odchylky a normalizována tak, aby její součet byl roven 1. Integrál f x h_{psf} přes PSF jediného pixelu je vypočten pomocí rovnice 5.1. [6]

5.2.4 Dvourozměrná schémata

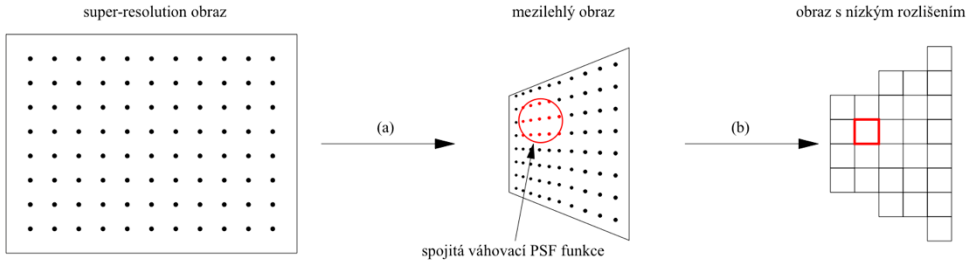
Následující tři obrázky (5.8, 5.9 a 5.10) ukazují přístup jednotlivých metod ve dvourozměrném schématu.

Metoda A. V kroku **(a)** je SR obraz zabalen, v závislosti na geometrické transformaci, do mezilehlého snímku. Pomocí interpolačního schématu jsou vypočteny hodnoty v neceločíselných polohách. V kroku **(b)** se mezilehlý snímek filtruje diskretizovanou verzí PSF v polohách, které odpovídají středům pixelu s nízkým rozlišením a vytváří obraz s nízkým rozlišením. [6]



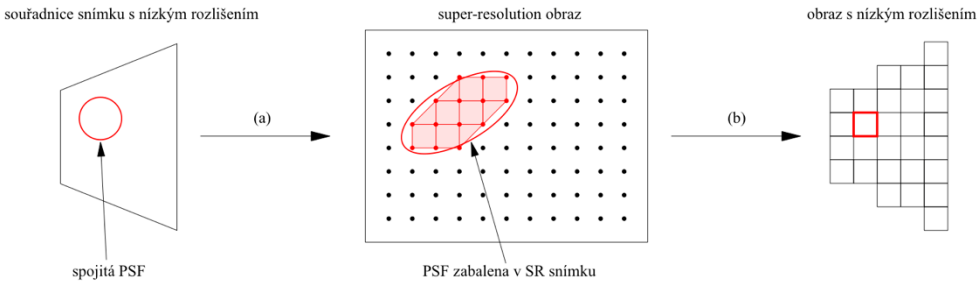
***Obr. 5.8** – metoda A [6]*

Metoda B. V kroku **(a)** jsou super-pixely bodově zabaleny do snímku s nízkým rozlišením podle geometrické transformace, bez interpolace. V kroku **(b)** je spojitá hodnota PSF funkce spojena s každým středem pixelu s nízkým rozlišením. Každý simulovaný pixel je PSF vážený součet super-pixelů. [6]



***Obr. 5.9** – metoda B [6]*

Metoda C. V kroku **(a)** je spojitá, zkrácená PSF s každým středem pixelu zabalena do SR snímku. V kroku **(b)** se předpokládá, že PSF a SR obraz jsou po částech bilineární a umožňují, aby byl integrál vypočten zcela přesně. [6]



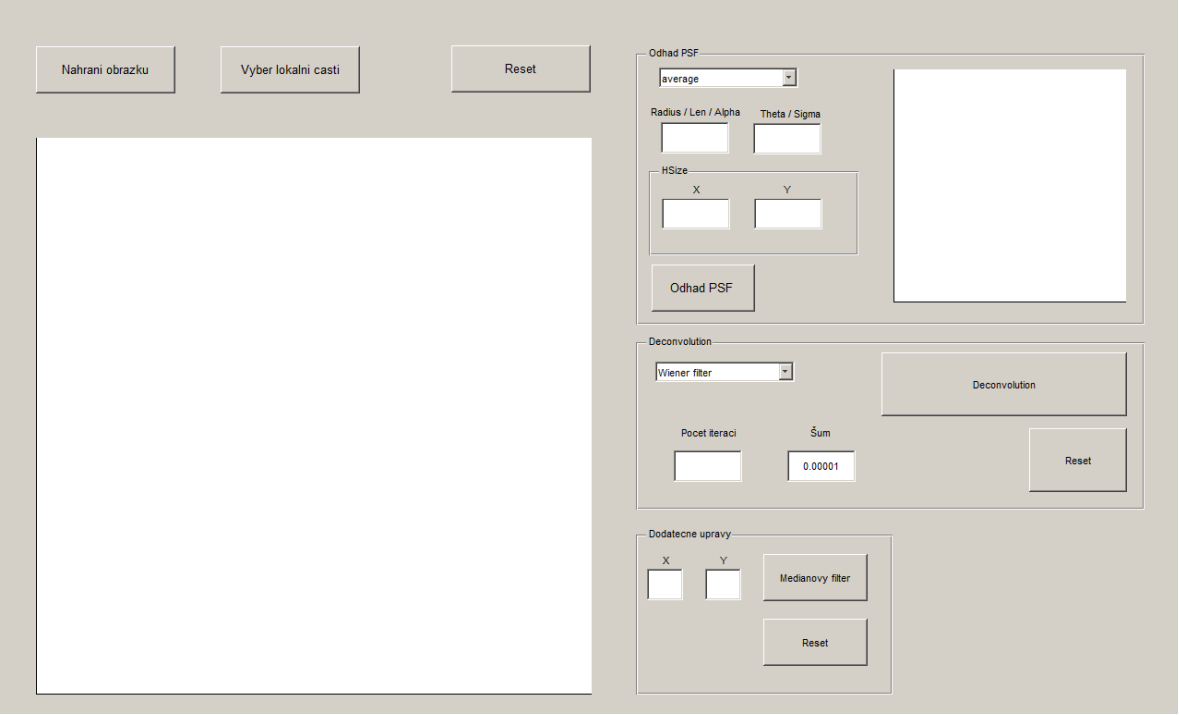
***Obr. 5.10** – metoda C [6]*

6 Popis algoritmů v programu Matlab

Cílem této práce bylo vytvořit algoritmy pro odstranění rozmazání v lokální části obrazu a zvýšení rozlišení lokální části pomocí metody super-resolution imaging. Jednotlivé algoritmy jsou přiloženy v přílohách. V následujících podkapitolách jsou tyto algoritmy popsány a vysvětleny. Návod k obsluze jednotlivých algoritmů se také nachází v přílohách.

6.1 Deconvolution algoritmus

Nejprve se podíváme na algoritmus k odstranění rozmazání. Na obrázku 6.1 je vidět náhled celkového GUI vytvořeného pomocí funkce *guide* v programu Matlab. Ovládání je velice jednoduché a intuitivní.



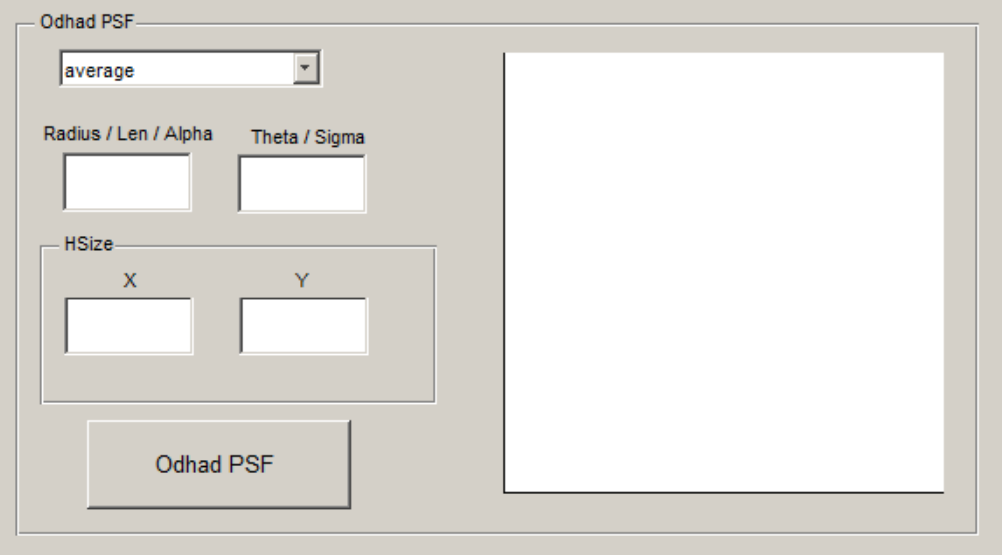
***Obr. 6.1** – GUI pro dekonvoluci*

Prostředí je rozděleno do čtyř hlavních částí. Nahrání obrázku, odhad PSF funkce, dekonvoluce a dodatečné úpravy. Po nahrání vybraného obrázku, se obrázek zobrazí v levém bílém poli. Po kliknutí na *„Výběr lokální části“* se uživateli zpřístupní vybrání lokální části obrázku pomocí myši. Tažením myši si uživatel vybere příslušnou lokální část, se kterou chce pracovat. Vybraná oblast se mu zobrazí místo nahraného obrazu. Tlačítkem *„Reset“* se

uživatel vždy vrátí jakkoli upravený obrázek do první varianty, která byla nahrána po spuštění programu. Zbylé tři bloky jsou popsány v podkapitolách níže.

6.1.1 Odhad PSF funkce

Při odhadu PSF funkce, která je vysvětlena v kapitole (4.1.1) musí uživatel nejprve vybrat a navolit několik vstupních parametrů. Tento odhad se provádí pomocí funkce *fspecial*. Pod obrázkem 6.2 jsou popsány jednotlivé vstupní parametry do této funkce.



***Obr. 6.2** – GUI pro odhad PSF funkce*

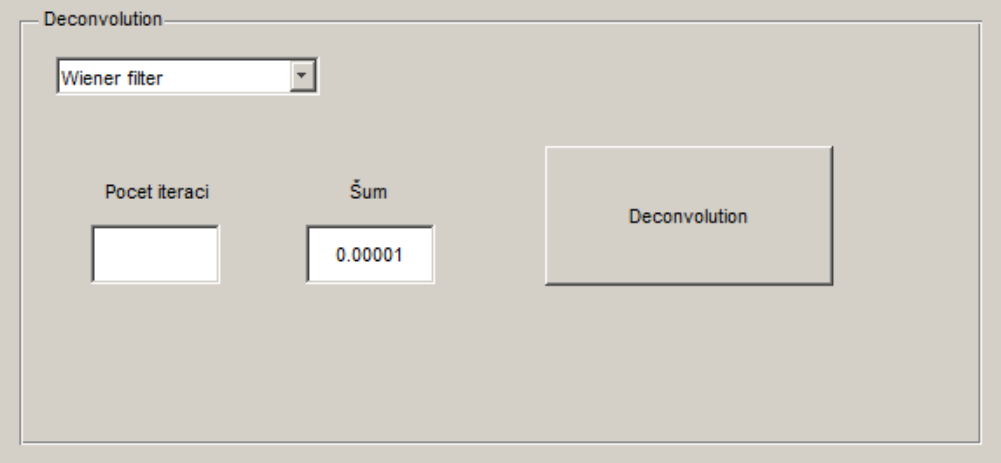
- **Average** – vytvoří průměrovací filtr o velikosti **HSize**, kde **HSize** je matice.
- **Disk** – vytvoří kruhový průměrovací filtr o velikosti 2***Radius**+1.
- **Laplacian** – vytvoří filtr 3x3, který aproximuje tvar 2D Laplacova operátoru. Parametr **Alpha** určuje tvar tohoto operátoru a musí být v rozmezí 0,0 – 0,1.
- **Gaussian** – vytvoří rotačně symetrický Gaussův filtr typu dolní propust o velikosti **HSize** se standartní deviací **Sigma**.
- **Log** – vytvoří rotačně symetrický Laplacian Gaussovo filtru o velikosti HSize a standartní deviaci **Sigma**.

- **Motion** – vytvoří filtr aproximující s lineárním pohybem kamery o **Len** pixely s úhlem **Theta** ve směru hodinových ručiček.

Po nastavení těchto parametrů se v bílém poli zobrazí náhled vytvořené PSF funkce, která se pak dále aplikuje na zvolený obrázek. Není-li uživatel s výběrem metody pro odhad PSF funkce spokojen, může ji změnit a přepsat tak stávající PSF funkci.

6.1.2 Deconvolution

Na obrázku 6.3 je zobrazen blok GUI pro dekonvoluci rozmazaného obrazu. Uživatel má na výběr ze čtyř metod pro dekonvoluci obrazu. Některé z těchto metod mají své vstupní parametry.



***Obr. 6.3** – GUI pro deconvolution*

- **Wiener filter** – vstupními parametry jsou nahraný obraz, PSF funkce a NSR. NSR je poměr šum / signál, kde šum je zadaný uživatelem a signál je vypočten z nahraného obrazu.

- **Blind deconvolution** - vstupními parametry jsou nahraný obraz, PSF funkce a Počet iterací. Počet iterací se opět zadává na hlavním panelu GUI.

- **Lucy-Richardson filter** – stejné parametry jako Blind deconvolution.

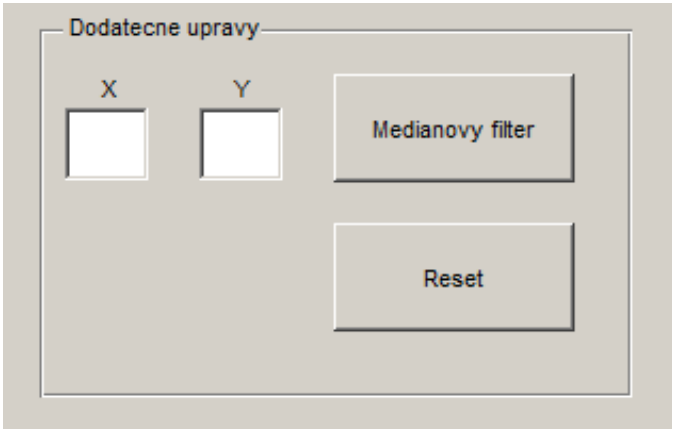
- **Tokhonov regularization** – vstupními parametry jsou pouze nahraný obraz a PSF funkce.

Stisknutím tlačítka *„Deconvolution“* spustí uživatel proces dekonvoluce dle vybrané metody. Po dokončení dekonvoluce se zobrazí upravený obraz. Není-li uživatel s výsledkem spokojen, může po stisknutí tlačítka *„Reset“* proces opakovat s jiným nastavením.

6.1.3 Dodatečné úpravy

Dodatečné úpravy se provádí, není-li uživatel úplně spokojen s výsledkem dekonvoluce. Nebo je-li potřeba vylepšit ostrost hran. Může zde využít metody mediánového filtru. Vstupními parametry tohoto filtru jsou upravený obraz pomocí dekonvoluce a dva parametry (X a Y), které se zadávají na hlavním panelu GUI.

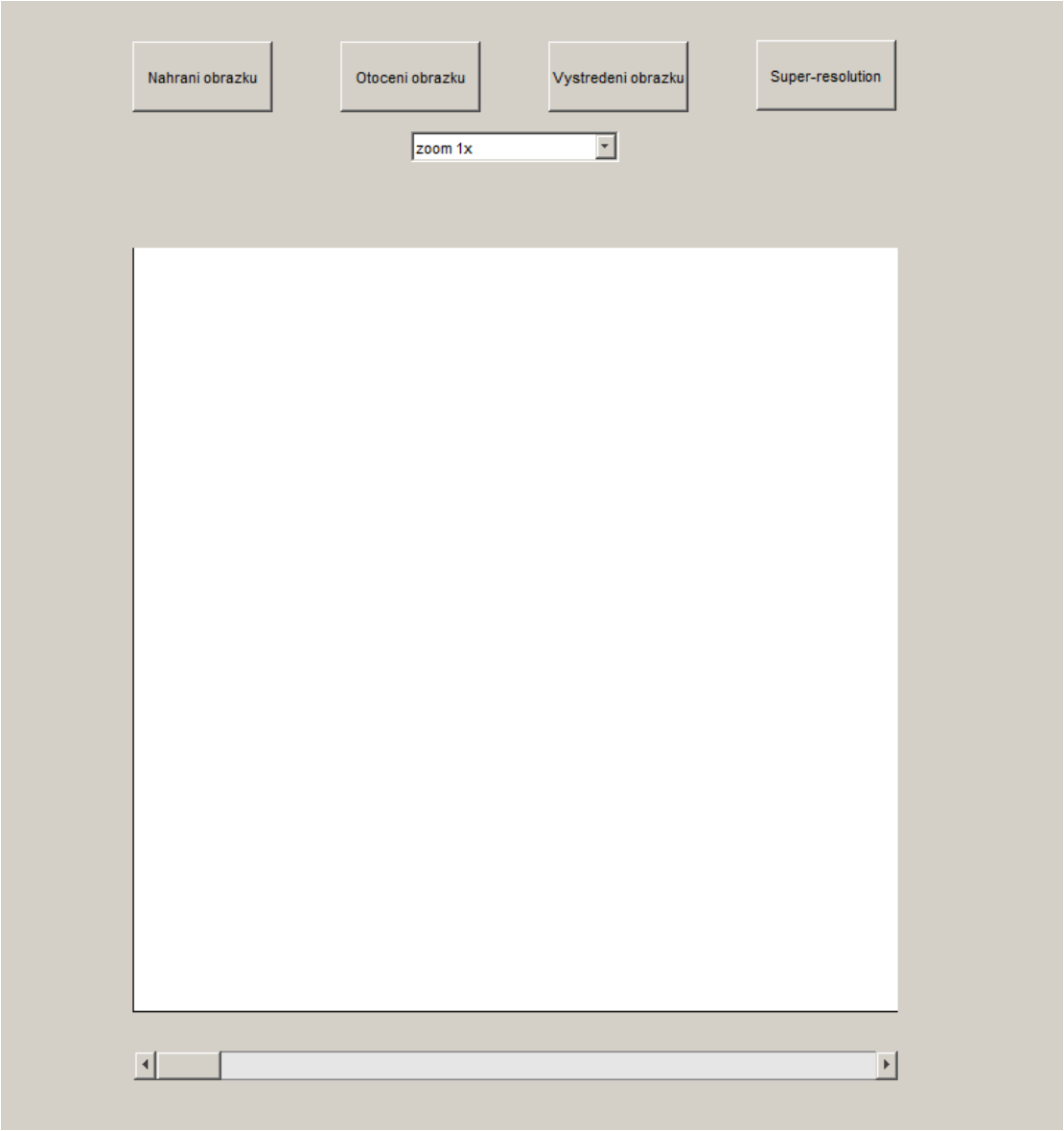
Mediánový filtr filtruje vstupní obraz. Každý výchozí pixel je dán mediánem o velikosti X - Y okolních pixelů okolo pixelu ve vstupním obraze. Výsledek lze opět vrátit tlačítkem *„Reset“*.



***Obr. 6.4** – GUI pro dodatečné úpravy*

6.2 Super-resolution imaging algoritmus

Po spuštění programu pro metodu Super-resolution imaging se zobrazí okno s příslušným GUI, je vidět na obrázku 6.4. Metoda SR spočívá ve zvýšení rozlišení obrazu pomocí více snímků stejného snímaného objektu. Nejprve, než se začnou nahrávat jednotlivé snímky, musí se všechny snímky přejmenovat na obrazek1.jpg až obrazekn.jpg, aby došlo ke správnému načtení všech snímků. Po načtení obrázků se v *„command“* okně v Matlabu zobrazí počet nahraných snímků. Uživatel si tak může zkontrolovat, zdali se nahráli všechny snímky správně.



***Obr. 6.5** - GUI pro metodu Super-resolution imaging*

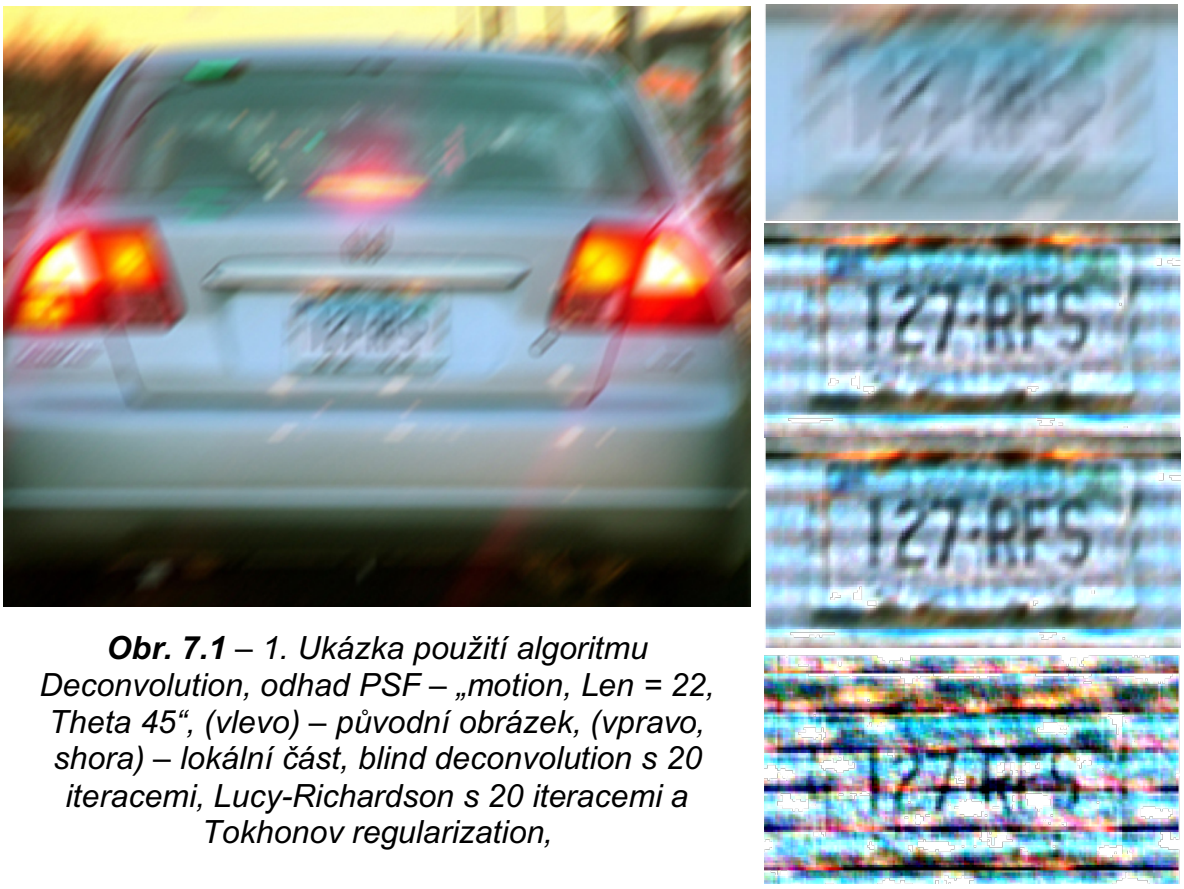
Nahrají-li se všechny snímky správně, zobrazí se v bílém poli pod tlačítky. Pomocí posuvníku, se lze mezi jednotlivými snímky posouvat. Je-li potřeba, je možné všechny snímky otáčet o 90° proti směru hodinových ručiček. Poté následuje jen vystředění všech snímků. Jsou-li snímky vystředěny, následuje výběr lokální části pomocí myši. Když je lokální část snímků vybrána, může ji uživatel přiblížit pomocí pop-up menu. Zoom může být dvojnásobný a čtyřnásobný. Přiblížený detail se zobrazí v novém okně. Algoritmus pro super-resolution imaging se spustí tlačítkem „Super resolution“. Po dokončení algoritmu, se snímek se super rozlišením zobrazí v novém okně. [13]

7 Ukázky použití jednotlivých metod

V této kapitole je uvedeno několik ukázek použití jednotlivých algoritmů. Ukázek není příliš mnoho, kvůli přehlednosti práce. Všechny zdrojové obrázky, snímky a videosekvence jsou přiloženy na CD v diplomové práci.

7.1 Deconvolution

Jak už bylo zmíněno, pro dekonvoluci rozostřených nebo rozmazaných obrazů byly použity čtyři metody. Jednotlivé metody jsou popsány v samostatné kapitole 4.2. Bohužel, ne všechny metody se hodí na tzv. pohybovou neostrost. Stejně tak vstupní parametry pro výpočet PSF funkce, tedy typ filtru, nejsou všechny užitečné. Nejvíce se využilo vstupního parametru *„motion“* a jeho parametrů. Ukázky jsou uvedeny níže i s příslušnými parametry.



***Obr. 7.1** – 1. Ukázka použití algoritmu Deconvolution, odhad PSF – „motion, Len = 22, Theta 45“, (vlevo) – původní obrázek, (vpravo, shora) – lokální část, blind deconvolution s 20 iteracemi, Lucy-Richardson s 20 iteracemi a Tokhonov regularization,*

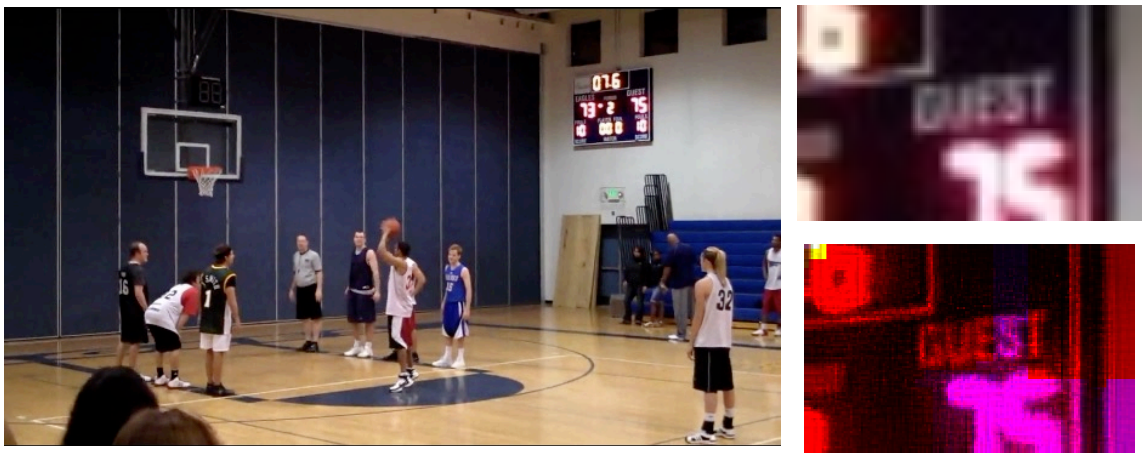
Obr. 7.2 *- 2. Ukázka použití algoritmu Deconvolution, odhad PSF – „motion, Len = 10, Theta 90“, (vlevo) – původní obrázek, (vpravo, shora) – lokální část, blind deconvolution s 15 iteracemi, Lucy-Richardson s 15 iteracemi*

Obr. 7.3 *- 3. Ukázka použití algoritmu Deconvolution, odhad PSF – „motion, Len = 10, Theta 60“, (vlevo) – původní obrázek, (vpravo, shora) – lokální část, blind deconvolution s 20 iteracemi, Lucy-Richardson s 20 iteracemi*

7.2 Super-resolution imaging

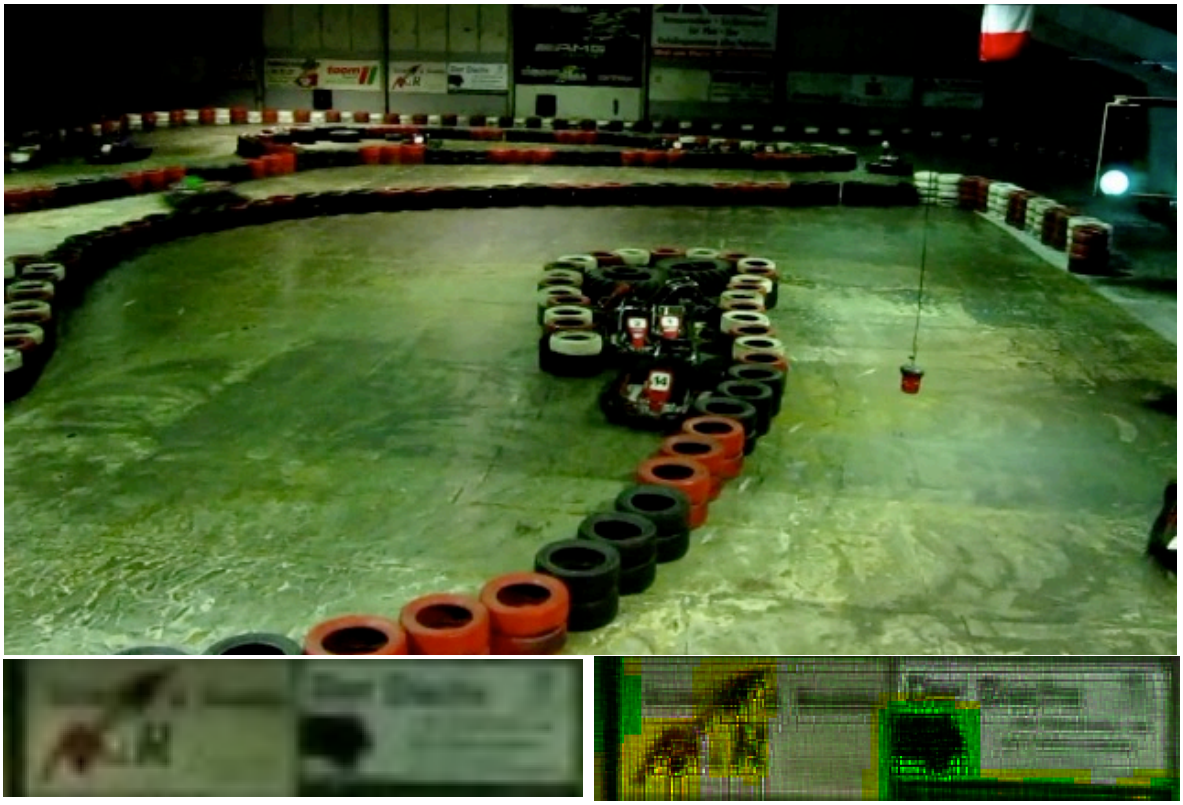
Níže jsou uvedeny čtyři ukázky použití metody super.resolution imaging. Všechny tyto ukázky jsou vytvořeny z videosekvencí, které byly rozděleny na jednotlivé snímky, na které pak byl aplikován super-resolution. Jak je možné vidět na příkladech, při použití algoritmu pro super-resolution imaging vznikají barevné artefakty.

***Obr. 7.4** – 1. ukázka použití alogoritmu pro super-resolution imaging, (vlevo) – původní obraz, (vpravo, shora) – zvětšení 4x, super-resolution*



***Obr. 7.5** – 2. ukázka použití alogoritmu pro super-resolution imaging, (vlevo) – původní obraz, (vpravo, shora) – zvětšení 4x, super-resolution*

Obr. 7.6 *– 3. ukázka použití alogoritmu pro super-resolution imaging, (vlevo) – původní obraz, (vpravo, shora) – zvětšení 4x, super-resolution*



Obr. 7.7 *– 4. ukázka použití alogoritmu pro super-resolution imaging, (vlevo) – původní obraz, (vpravo, shora) – zvětšení 4x, super-resolution*

Závěr

Cílem této diplomové práce bylo seznámit čtenáře se základním rozdělením rozmazání obrazu. Vysvětlit vznik rozmazaných nebo neostrých obrazů a jak těmto neduhům zabránit. Poté byly nastíněny některé metody, které tyto problémy odstraňují. Jednotlivé metody byly vysvětleny a ukázány na příkladech. Podle těchto metod byl vytvořen algoritmus v programu Matlab. Jak bylo zmíněno výše, velký vliv na výsledný nerozmazaný obraz má funkce PSF. Tuto funkci je potřeba v algoritmu odhadnout za pomoci několika parametrů. Na testovacích obrazech byly metody vyzkoušeny při různých nastaveních. Výsledné obrazy i s originálním rozmazaným obrazem jsou k vidění v kapitole 7.1.

Druhým cílem této diplomové práce bylo vysvětlit co je super-resolution imaging. Tato problematika je obsažena v kapitole 5. Jsou vysvětleny jednotlivé přístupy a metody jak za pomoci více snímků získaných z fotoaparátu nebo z videosekvence získat obraz se super rozlišením. Jednotlivé metody jsou velice komplikované a náročné na výpočet. Ukázky použití metody super-resolution imaging jsou uvedeny v kapitole 7.2. Jsou zde sekvence jednotlivých snímků a výsledný obraz se super rozlišením.

Kapitola 7.1 obsahuje ukázkové příklady použití algoritmu pro odstranění lokálního rozmazání. Byly vyzkoušeny všechny metody, které algoritmus nabízí. Uvedené metody nevykazovaly stejné výsledky při odstranění lokálního rozmazání. Nejlepších výsledků dosahovaly metody Blind deconvolution a Lucy-Richardson. Důležitým aspektem při odstranění lokálního rozmazání byl odhad PSF funkce. Jak můžeme vidět v ukázkách, byl použit pouze parametr *„motion“*. Ten se jevil jako nejlepší při pohybové neostrosti tzv. motion blur.

Jak je možné vidět v kapitole 7.2, algoritmus pro super-resoluton imaging vytváří nechtěné barevné artefakty. Tyto artefakty mohou být zapříčiněny pohybem jiného objektu přes uživatelem vybranou část, na kterou je následně aplikováno SR. Jelikož jednotlivé snímky byly vytvořeny z videosekvencí a tyto videosekvence nebyly správně stabilizované, došlo k posunutí jednotlivých snímků. I když algoritmus provádí vystředění snímků, není toto vystředění úplně bezchybné. Také kvalita těchto vstupních videosekvencí není příliš vysoká. Všechny tyto aspekty by se daly vyřešit po delším zkoumání jednotlivých parametrů v algoritmu pro super-resolution imaging.

V budoucnu by bylo dobré obohatit algoritmy o další funkce, které by proškolené obsluze umožňovaly dosahovat lepších výsledků. Není cílem, aby všechny úpravy provedl program automaticky po stisknutí tlačítka uživatelem, ale aby program fungoval v interakci s uživatelem a tím se dosahovalo co možná nejlepších výsledků ve vnímání obrazu člověkem.

Seznam literatury a informačních zdrojů

[1] YUZHIKOV V. Restoration of defocused and blurred images. *yuzhikov.com* [online]. Cit. [2017-03-27]. Dostupné z: http://yuzhikov.com/index.html

[2] PETE. Shutter speed: A beginner's guide. *photographymad.com* [online]. Cit. [2017-03-27]. Dostupné z: http://www.photographymad.com/pages/view/shutter-speed-a-beginners-guide

[3] YUZHIKOV V. Restoration of defocused and blurred images. *yuzhikov.com* [online]. Cit. [2017-03-27]. Dostupné z: http://yuzhikov.com/articles/BlurredImagesRestoration1.htm

[4] GONZALES R., WOODS R. a EDDINS E. *Digital image processing using Matlab second edition*. Vydavatelství Gatesmark Publishing, USA 2009

[5] YUZHIKOV V. Restoration of defocused and blurred images. *yuzhikov.com* [online]. Cit. [2017-03-27]. Dostupné z: http://yuzhikov.com/articles/BlurredImagesRestoration2.htm

[6] CAPEL D. *Image mocaicing and Super-resolution.* Oxford: University of Oxford, Robotics Research Group, Department of Engineering Science, 2001.

[7] CAPEL D., ZISSERMAN A. *Automated mosaicing with super-resolution zoom.* IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, Červen 1998

[8] AGGARWAL M., HUA H., AHUJA N. *On cosine-fourth and vignetting effects in real lens.* 8. International Conference on Computer Vision, Vancouver, 2001

[9] REICHENBACH S. E., PARK S. K., NARAYANSWAMY R. *Characterizing digital image quisition devices.* Optical Engineering, 1991

[10] TSIN Y., RAMESH V., KANADE T. *Statistical calibration of the ccd imaging process.* 8. International Conference on Computer Vision, Vancouver, 2001

[11] ROBERTSON M. A., BORMAN S., STEVENSON R. L. *Estimation theoretic approach to dynamic range improvement through multiple exposures.* IEEE International Conference on Image Processing, 1999

[12] ZOMET A., PELEG S. *Efficient super.resolution and application to mosaics.* International Conference on Pattern Recognition, 2000

[13] CHENG S. Superresolution demo. *mathworks.com* [online]. Cit. [2017-04-01] Dostupné z: https://www.mathworks.com/matlabcentral/fileexchange/30488-superresolution-demo

[14] THE MATHWORKS, Inc. Help. *mathworks.com* [online]. Cit. [2017-04-20]. Dostupné z: https://www.mathworks.com/help

[15] HRADIŠ M. CNN for License Plate Motion Deblurring. fit.vutbr.cz [online]. Cit. [2017-4-25]. Dostupné z: http://www.fit.vutbr.cz/~ihradis/CNN-Deblur/

[16] DEE M. Unique Solutions. *infognition.com* [online]. Cit. [2017-4-25]. Dostupné z: http://www.infognition.com/videoenhancer/

[17] KALYANS. TVSG Submission.eecs.harvard.edu [online]. Cit. [2017-4-25]. Dostupné z: http://www.eecs.harvard.edu/~kalyans/research/snapshots/supplementary/results.html

Přílohy

Příloha A – Zdrojový kód pro Deconvolution program

```
function varargout = Deconvolution(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Deconvolution_OpeningFcn, ...
                   'gui_OutputFcn',  @Deconvolution_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 deconvolution is made visible.
function Deconvolution_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Deconvolution_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in ButtonLoadImage.
function ButtonLoadImage_Callback(hObject, eventdata, handles)

% nahrani obrazku
global im im2
[path,user_cance] = imgetfile();
if user_cance
    msgbox(sprintf('Chyba'),'Chyba','Chyba');
    return
end
im = imread(path);
im = im2double(im);      % prevod na double
im2 = im;                % zaloha, kdyby nahodou

axes(handles.axes1);
imshow(im);
im;
% --- Executes on button press in ButtonReset.
function ButtonReset_Callback(hObject, eventdata, handles)

% funkce pro tlacitko reset
global im2
axes(handles.axes1);
imshow(im2);
```

```
% --- Executes on button press in ButtonChooseLocalPart.
function ButtonChooseLocalPart_Callback(hObject, eventdata, handles)

% vyber lokalni casti
global im local local2
sp = im;
sp = getrect;       % vyber lokalni casti pomoci mysi
% ziskani x a y koordinatu z funkce getrect
sp(1) = max(floor(sp(1)), 1);       % xmin
sp(2) = max(floor(sp(2)), 1);       % ymin
sp(3) = min(ceil(sp(1) + sp(3)));   % xmax
sp(4) = min(ceil(sp(2) + sp(4)));   % ymax

local = im(sp(2):sp(4), sp(1): sp(3),:);    % indexov·nÌ koordinatu zpet do
puvodniho obrazku

axes(handles.axes1);
imshow(local);

% --- Executes on button press in ButtonPSF.
function ButtonPSF_Callback(hObject, eventdata, handles)

% odhad PSF funkce
global PSF popChoosePSFVal inputpar1Val pocetpar inputpar2Val inputpar3Val
inputpar4Val
if (pocetpar == 1)
    PSF = fspecial(popChoosePSFVal,inputpar1Val);   % PSF pro metody s jednim
parametrem (disk, laplacian)
elseif (pocetpar == 2)
    PSF = fspecial(popChoosePSFVal,inputpar1Val,inputpar2Val);  % PSF pro metody
s dvema parametry (motion)
elseif (pocetpar == 3)
    PSF = fspecial(popChoosePSFVal,[inputpar3Val inputpar4Val]);    % PSF pro
metody s vektorovym parametrem (average)
elseif (pocetpar == 4)
    PSF = fspecial(popChoosePSFVal,[inputpar3Val inputpar4Val],inputpar2Val);   %
PSF pro metody s vektorovym parametrem a parametrem (log, gaussian)
end
axes(handles.axes2);
imshow(PSF,[]);     % zobrazeni PSF funkce

% --- Executes during object creation, after setting all properties.
function ChoosePSF_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% popupmenu - vyber PSF metody
% --- Executes on selection change in ChoosePSF.
function ChoosePSF_Callback(hObject, eventdata, handles)
global popChoosePSFVal pocetpar
contents = cellstr(get(hObject,'String'));
popChoosePSF = contents{get(hObject,'Value')};
if (strcmp(popChoosePSF,'disk'))
    popChoosePSFVal = 'disk';   % je-li vybrana metoda "disk"
    pocetpar = 1;       % pocet parametru
elseif (strcmp(popChoosePSF,'average'))
    popChoosePSFVal = 'average';
    pocetpar = 3;
elseif (strcmp(popChoosePSF,'laplacian'))
    popChoosePSFVal = 'laplacian';
    pocetpar = 1;
elseif (strcmp(popChoosePSF,'gaussian'))
    popChoosePSFVal = 'gaussian';
    pocetpar = 4;
```

```
elseif (strcmp(popChoosePSF,'log'))
    popChoosePSFVal = 'log';
    pocetpar = 4;
elseif (strcmp(popChoosePSF,'motion'))
    popChoosePSFVal = 'motion';
    pocetpar = 2;
end

function InputPar_1_Callback(hObject, eventdata, handles)
global inputpar1Val

inputpar1Val = str2double(get(handles.InputPar_1,'string'));    % cteni z pole
zadaneho parametru

% --- Executes during object creation, after setting all properties.
function InputPar_1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function InputPar_2_Callback(hObject, eventdata, handles)
global inputpar2Val

inputpar2Val = str2double(get(handles.InputPar_2,'string'));    % cteni z pole
zadaneho parametru

% --- Executes during object creation, after setting all properties.
function InputPar_2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function InputPar_3_Callback(hObject, eventdata, handles)
global inputpar3Val

inputpar3Val = str2double(get(handles.InputPar_3,'string'));    % cteni z pole
zadaneho parametru

% --- Executes during object creation, after setting all properties.
function InputPar_3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function InputPar_4_Callback(hObject, eventdata, handles)
global inputpar4Val

inputpar4Val = str2double(get(handles.InputPar_4,'string'));    % cteni z pole
zadaneho parametru

% --- Executes during object creation, after setting all properties.
function InputPar_4_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% popupmenu - vyber deconvolution metody
% --- Executes on selection change in ChooseDeconv.
function ChooseDeconv_Callback(hObject, eventdata, handles)
global popChooseDeconv popChooseDeconvVal metoda
```

```
contents = cellstr(get(hObject,'String'));
popChooseDeconv = contents{get(hObject,'Value')};
if (strcmp(popChooseDeconv,'Wiener filter'))    % je-li vybrana metoda "winer
filter"
    popChooseDeconvVal = 'Wiener filter';
    metoda = 1;     % cislo metody
elseif (strcmp(popChooseDeconv,'Blind deconvolution'))
    popChooseDeconvVal = 'Blind deconvolution';
    metoda = 2;
elseif (strcmp(popChooseDeconv,'Tikhonov regularization'))
    popChooseDeconvVal = 'Tikhonov regularization';
    metoda = 3;
elseif (strcmp(popChooseDeconv,'Lucy-Richardson filter'))
    popChooseDeconvVal = 'Lucy-Richardson filter';
    metoda = 4;
end

% --- Executes during object creation, after setting all properties.
function ChooseDeconv_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% nastaveni tlacitka Deconvolution
% --- Executes on button press in ButtonDeconv.
function ButtonDeconv_Callback(hObject, eventdata, handles)
global metoda PSF DeBlurrImage NumberOfIter nsr IputNoiseVal h local

if (metoda == 1)    % cislo metody
    nsr = IputNoiseVal / var(local(:));    % vypocet noise/signal ratio
    h = msgbox('Deconvolution...','»ekejte...');
    DeBlurrImage = (deconvwnr(local,PSF,nsr)); % provedeni deconvoluce
    delete(h);
elseif (metoda == 2)
    h = msgbox('Deconvolution...','»ekejte...');
    DeBlurrImage = (deconvblind(local,PSF,NumberOfIter));
    delete(h);
elseif (metoda == 3)
    h = msgbox('Deconvolution...','»ekejte...');
    DeBlurrImage = (deconvreg(local,PSF));
    delete(h);
elseif (metoda == 4)
    h = msgbox('Deconvolution...','»ekejte...');
    DeBlurrImage = (deconvlucy(local,PSF,NumberOfIter));
    delete(h);
end

axes(handles.axes1);
imshow(DeBlurrImage);   % zobrazeni vysledneho obrazu

function InputParDeconvIter_Callback(hObject, eventdata, handles)
global NumberOfIter

NumberOfIter = str2double(get(handles.InputParDeconvIter,'string'));    % cteni z
pole zadaneho parametru

% --- Executes during object creation, after setting all properties.
function InputParDeconvIter_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function InputNoise_Callback(hObject, eventdata, handles)
```

```
global IputNoiseVal

IputNoiseVal = str2double(get(handles.InputNoise,'string'));    % cteni z pole
zadaneho parametru

% --- Executes during object creation, after setting all properties.
function InputNoise_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in ButtonMedian.
function ButtonMedian_Callback(hObject, eventdata, handles)
global DeBlurrImage final ParMedian1 ParMedian2
final = DeBlurrImage;
for c = 1:3
    final(:,:,c) = medfilt2(DeBlurrImage(:,:,c), [ParMedian1, ParMedian2]);
end

axes(handles.axes1);
imshow(final);

% --- Executes on button press in ButtonResetMedian.
function ButtonResetMedian_Callback(hObject, eventdata, handles)
global DeBlurrImage
axes(handles.axes1);
imshow(DeBlurrImage);

function ParMedian1_Callback(hObject, eventdata, handles)
global ParMedian1

ParMedian1 = str2double(get(handles.ParMedian1,'string'));

% --- Executes during object creation, after setting all properties.
function ParMedian1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function ParMedian2_Callback(hObject, eventdata, handles)
global ParMedian2

ParMedian2 = str2double(get(handles.ParMedian2,'string'));

% --- Executes during object creation, after setting all properties.
function ParMedian2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in ButtonResetDeconv.
function ButtonResetDeconv_Callback(hObject, eventdata, handles)
global local
axes(handles.axes1);
imshow(local);
```

Příloha B – Návod k obsluze Deconvolution programu

1. **Nahrání obrázku**
   - po spuštění programu (Deconvolution.m) se zobrazí hlavní panel GUI. Tlačítkem *Nahrát obrázek* se nahraje zvolený obrázek, který se zobrazí v bílém poli.
2. **Výběr lokální části obrazu pomocí myši**
   - myší uživatel vybere lokální část. Vybraná lokální část se zobrazí místo nahraného obrázku. Tlačítkem *Reset* se opět zobrazí nahraný obrázek.
3. **Odhad PSF funkce**
   - uživatel si vybere z rolovacího menu metodu a zadá příslušné parametry. Stisknutí tlačítka *Odhad PSF* se zobrazí náhled funkce.

Metoda	**Parametry (forma)**
motion	Len (skalár), Theta (skalár)
average	HSize (matice)
disk	Radius (skalár)
laplacian	Alpha (skalár)
gausian	HSize (matice), Sigma (skalár)
log	HSize (matice), Sigma (skalár)

4. **Dekonvoluce**
   - uživatel si vybere metodu pro dekonvoluci a zadá příslušné parametry. Před opakováním dekonvoluce, musí uživatel resetovat poslední uloženou dekonvoluci.

Metoda	**Parametry**
Blind deconvolution	Počet iterací
Lucy-Richarson filter	Počet iterací
Tikhonov regularization	-
Wiener filter	šum

5. **Dodatečné úpravy**
   - po dokončení dekonvoluce, má uživatel možnost aplikovat na výsledný obraz mediánový filtr na vyhlazení hran.

Příloha C – Zdrojový kód pro Super-resolution imaging

```
function varargout = super(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @super_OpeningFcn, ...
                   'gui_OutputFcn',  @super_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 super is made visible.
function super_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = super_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% skript pro tlacitko nahrani obrazku
% --- Executes on button press in ButtonLoadImages.
function ButtonLoadImages_Callback(hObject, eventdata, handles)

folder = uigetdir;                                  % prochazeni slozky
files = dir(sprintf('%s/*.JPG',folder));            % hledani souboru .JPG
pocet_obrazku = (length(files))                     % zjisteni poctu obrazku

handles.images=[];
h=waitbar(0,'Nahravani...');                        % loading bar
for id=1:pocet_obrazku                              % postupne nacitani obrazku ve
slozce
    jpgFileName = sprintf('obrazek (%d).JPG',id);
    fullFileName = fullfile(folder,jpgFileName);
    if exist(fullFileName,'file')
    handles.images{id}=double(imread(fullFileName))/255;    % nacitani obrazku
    else                                            % chybova hlaska, kdyz nejsou
nalezeny obrazky ve slozce
    warningMessage = sprintf('Varovani, soubor neexistuje:\n%s', fullFileName);
    uiwait(warndlg(warningMessage));
    end

    if length(size(handles.images{id})) == 2    % rozsireni grey scale image to
color image format
        t=handles.images{id};
        t=zeros([size(handles.images{id}) 3]);
        t(:,:,1)=handles.images{id};
        t(:,:,2)=handles.images{id};
```

```
        t(:,:,3)=handles.images{id};
        handles.images{id}=t;
    end
    waitbar(id/pocet_obrazku,h);                % loading bar
end
delete(h);
% zobrazeni nahranych obrazku
axes(handles.ImageAxes);
hold off;
imshow(handles.images{1});
hold on;
% nastaveni pro posuvny slider na prepinani mezi jednotlivymi snimky
set(handles.FrameSlider,'min',1);
set(handles.FrameSlider,'max',length(handles.images));
set(handles.FrameSlider,'value',1);
set(handles.frame_title,'string','Frame 1');
title('Snimek 1');
guidata(hObject,handles);

% skript pro tlacitko otoceni obrazku
% --- Executes on button press in ButtonRotateImages.
function ButtonRotateImages_Callback(hObject, eventdata, handles)

h=waitbar(0,'Otoceni obrazku...');                  % loading bar
for id=1:length(handles.images)
    clear timage;
    for color=1:3                                   % otoceni obrazku
        timage(:,:,color)=double(flipud(handles.images{id}(:,:,color)'));
    end
    handles.images{id}=timage;
    waitbar(id/length(handles.images),h);
end
delete(h);
% zobrazeni otocenych obrazku
axes(handles.ImageAxes);
hold off;
imshow(handles.images{1});
hold on;
% nastaveni pro posuvny slider na prepinani mezi jednotlivymi obrazky
set(handles.FrameSlider,'value',1);
set(handles.frame_title,'string','Frame 1');
title('Frame 1');
guidata(hObject,handles);

% nastaveni slideru
% --- Executes on slider movement.
function FrameSlider_Callback(hObject, eventdata, handles)

% zobrazeni obrazku
axes(handles.ImageAxes);
hold off;
imshow(handles.images{round(get(hObject,'value'))});
hold on;
% zobrazeni poradi obrazku
title(sprintf('Frame %d',round(get(hObject,'value'))));

% --- Executes during object creation, after setting all properties.
function FrameSlider_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% nastaveni tlacitka na vystredeni obrazku
% --- Executes on button press in ButtonRegisterImage.
function ButtonRegisterImage_Callback(hObject, eventdata, handles)
```

```
h=waitbar(0,'Vystredovani snimku...');         % loading bar
for id=2:length(handles.images)
    clear timage;
    % vyuziti funkce register_color_image
    timage=vystredeni_snimku(double(handles.images{1}),
double(handles.images{id}));
    handles.images{id}=timage;
    waitbar(id/length(handles.images),h);
end
delete(h);
guidata(hObject,handles);
FrameSlider_Callback(hObject, [], handles)

% funkce pro oznacovani lokalniho mista pomoci mysi
% --- Executes on mouse motion over figure - except title and menu.
function figure1_WindowButtonMotionFcn(hObject, eventdata, handles)

pos=get(handles.ImageAxes,'currentpoint');
pos=pos(1,1:2);
if inrange(pos,handles.ImageAxes)
    if isfield(handles,'mouse_pressed')
        if handles.mouse_pressed
           if isfield(handles,'last_box')
                if ishandle(handles.last_box)
                    delete(handles.last_box);
                end
           end
           axes(handles.ImageAxes);
           handles.last_box=drawbox(handles.start_pos,pos);
           guidata(hObject,handles);
        end
    end

end

function h=drawbox(p1,p2)
x=[p1(1) p2(1) p2(1) p1(1) p1(1)];
y=[p1(2) p1(2) p2(2) p2(2) p1(2)];
h=plot(x,y);

% funkce pro oznacovani lokalniho mista pomoci mysi
% --- Executes on mouse press over figure background, over a disabled or
% --- inactive control, or over an axes background.
function figure1_WindowButtonDownFcn(hObject, eventdata, handles)

pos=get(handles.ImageAxes,'currentpoint');
pos=pos(1,1:2);
if inrange(pos,handles.ImageAxes)
    handles.start_pos=pos;
    handles.mouse_pressed=1;
end
guidata(hObject,handles);

function flag=inrange(pos,h)
s=size(get(findobj(h,'type','image'),'cdata'));

if (pos(1)>=1) && (pos(2) >=1) && (pos(2) <= s(1)) && (pos(1) <= s(2))
    flag=1;
else
    flag=0;
end

% funkce pro oznacovani lokalniho mista pomoci mysi
% --- Executes on mouse press over figure background, over a disabled or
% --- inactive control, or over an axes background.
function figure1_WindowButtonUpFcn(hObject, eventdata, handles)

pos=get(handles.ImageAxes,'currentpoint');
```

```
pos=pos(1,1:2);

if inrange(pos,handles.ImageAxes)
    handles.mouse_pressed=0;
    handles.end_pos=pos;
    guidata(hObject,handles);
end

% zoom menu
% --- Executes on selection change in PopupMenuZoom.
function PopupMenuZoom_Callback(hObject, eventdata, handles)

p1(1)=min([handles.start_pos(1),handles.end_pos(1)]);
p1(2)=min([handles.start_pos(2),handles.end_pos(2)]);
p2(1)=max([handles.start_pos(1),handles.end_pos(1)]);
p2(2)=max([handles.start_pos(2),handles.end_pos(2)]);
p1=round(p1); p2=round(p2);
imobj=findobj(handles.ImageAxes,'type','image');
cdata=get(imobj,'cdata');
fid=round(get(handles.FrameSlider,'value'));
cdata=handles.images{fid};
imclip=cdata(p1(2):p2(2),p1(1):p2(1),:);

contents=get(hObject,'String');
figure;
% rozbalovaci menu Zoom
switch contents{get(hObject,'value')}
    case 'zoom 1x'
        imshow(imclip);
        title('zoom 1x');
    case 'zoom 2x'
        imclip=interpolace1(double(imclip));    % volani pomocne funkce pro
bilinearni interpolace
        imshow(imclip);
        title('zoom 2x');
    case 'zoom 4x'
        imclip=interpolace1(double(imclip));    % volani pomocne funkce pro
bilinearni interpolace
        imclip=interpolace1(imclip);
        imshow(imclip);
        title('zoom 4x');
end

% --- Executes during object creation, after setting all properties.
function PopupMenuZoom_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function [p1,p2]=get_clip_range(handles)
p1(1)=min([handles.start_pos(1),handles.end_pos(1)]);
p1(2)=min([handles.start_pos(2),handles.end_pos(2)]);
p2(1)=max([handles.start_pos(1),handles.end_pos(1)]);
p2(2)=max([handles.start_pos(2),handles.end_pos(2)]);
p1=round(p1); p2=round(p2);

% funkce pro tlacitko super-resolution
% --- Executes on button press in ButtonSuperResolution.
function ButtonSuperResolution_Callback(hObject, eventdata, handles)

[p1,p2]=get_clip_range(handles);
h=waitbar(0,'Super-resolution imaging...');              % loading bar
for color=1:3
    for id=1:length(handles.images)
        t2{id}=handles.images{id}(p1(2):p2(2),p1(1):p2(1),color)*255;
    end
```

```
    image{color}=sr_bw(t2,h,(color-1)/3,1/3)/255;
end
delete(h);
tmpimg=zeros([size(image{color}) 3]);
for color=1:3
    tmpimg(:,:,color)=image{color};
end
figure; imshow(tmpimg); title('Super-resolution obraz');
```

Příloha D – Pomocné algoritmy pro Super-resolution imaging

gen_shift_downsample_image.m

```
function imout=gen_shift_downsample_image(image,ds,sh)

image=image(1:floor(size(image,1)/ds)*ds,1:floor(size(image,2)/ds)*ds);

image=posun_snimku(image,sh);

imouts=floor(size(image)/ds);

imout=zeros(imouts);
for id=1:ds
    for jd=1:ds
        imout=imout+image(id:ds:size(image,1),jd:ds:size(image,2));
    end
end
imout=imout/ds/ds;
```

Interpolace1.m

```
% funkce bilinearni interpolace1
function imgout=interpolace1(img)

if length(size(img))==3
    for id=1:size(img,3)
        imgout(:,:,id)=interpolace2(img(:,:,id));
    end
else
    imgout=interpolace2(img);
end
```

interpolace2.m

```
% funkce bilinearni interpolace2
function imgout=interpolace2(img)

imgout=interpolace_1d(img);
imgout=interpolace_1d(imgout');
imgout=imgout';

% funkce bilinearni interpolace pro 1D pole
function imgout=interpolace_1d(img)

interp=(img(:,1:size(img,2)-1)+img(:,2:size(img,2)))/2;

imgout=zeros(size(img,1),size(img,2)*2-1);
imgout(:,1:size(imgout,2)-1)=kron(img(:,1:size(img,2)-1),[1 0])+kron(interp,[0
1]);
imgout(:,size(imgout,2))=img(:,size(img,2));
```

pocs.m

```
function image=pocs(image,lr_image,ds,sh)

sh_image=gen_shift_downsample_image(image,ds,sh);
scale=lr_image./sh_image;

scale=posun_snimku(kron(scale,ones(ds)),-sh);
image=image.*scale;
```

posun_snimku.m

```
% pomocna funkce pro vystredeni snimku
function image=posun_snimku(image,sh)
```

```
if sh(1)>0 % nad
    image=[repmat(image(1,:),[sh(1),1]); image];
    image(size(image,1)-sh(1)+1:size(image,1),:)=[];
elseif sh(1)<0 % pod
    image=[image; repmat(image(size(image,1),:),[-sh(1),1])];
    image(1:-sh(1),:)=[];
end

if sh(2)>0 % vlevo
    image=[repmat(image(:,1),[1,sh(2)]), image];
    image(:,size(image,2)-sh(2)+1:size(image,2))=[];
elseif sh(2)<0 % vpravo
    image=[image, repmat(image(:,size(image,2)),[1,-sh(2)])];
    image(:,1:-sh(2))=[];
end
```

sr_bw.m

```
function
image=sr_bw(t2,handle_wb,offset_wb,scale_wb,ds,max_shift,th_prob1,th_prob2,no_ite
r)

if ~exist('ds','var')
    ds=4; % four time super resolution
end
if ~exist('max_shift','var')
    max_shift=8;
end
if ~exist('th_prob1','var')
    th_prob1=0.9;
end
if ~exist('th_prob2','var')
    th_prob2=0.85;
end
if ~exist('no_iter','var')
    no_iter=2;
end

fprintf('start SR\n...');
search_range=[-max_shift max_shift -max_shift max_shift];
image=kron(t2{1},ones(ds));

sigma=-1; % use the default sigma to estimate prob (see also subpixel_register.m)
fprintf('start super-resolution 1st phase...\n');
tic
[image,probs,shs,scores]=sr_one_step_wb(image,t2,'average',ds,search_range,sigma,
th_prob1,handle_wb,offset_wb,scale_wb/2); %,...
toc

fprintf('start super-resolution 2nd phase...\n');
for iter=1:no_iter
    tic

[image,probs,shs,scores]=sr_one_step_wb(image,t2,'sort_pocs',ds,search_range,sigm
a,th_prob2,handle_wb,offset_wb+(iter-
1)*scale_wb/2/no_iter+scale_wb/2,scale_wb/2/no_iter);
    toc
    iter
    if length(find(probs>th_prob2))==length(probs)
          break;
    end
end
```

sr_one_step_wb.m

```
function
[image,probs,shs,scores]=sr_one_step_wb(image,t,method,ds,sr,sigma,th_prob,wb_han
dle,offset,scale,orig_sh)
```

```
if ~exist('th_prob','var')
    th_prob=0.9;
end

shs=zeros(length(t),2);

orig_image=image;
if exist('orig_sh','var') % cheat mode
   scores=[];
   for tid=1:length(t)
     shs(tid,:)=orig_sh(tid,:);
     probs(tid)=1;
   end
else
  for tid=1:length(t)
      waitbar(tid/2/length(t)*scale+offset,wb_handle);
        [tmp_sh,tmp_prob,tmp_scores]=subpixel_register(image,t{tid},ds,sr,sigma);
       probs(tid)=tmp_prob;
       shs(tid,:)=tmp_sh;
       scores{tid}=tmp_scores;
  end
end

if strcmp(method,'max_pocs') % only project to the one with highest score
    [~,max_id]=max(probs);
    image=pocs(image,t{max_id},ds,shs(max_id,:));
elseif strcmp(method,'pocs')
  for tid=1:length(t)
      waitbar((length(t)+tid)/2/length(t)*scale+offset,wb_handle);
     if (probs(tid) > th_prob)
            image=pocs(image,t{tid},ds,shs(tid,:));
     end
  end
elseif strcmp(method,'sort_pocs')
    [~,sort_ids]=sort(probs(:),1,'descend');
    for tid=1:length(t)
      waitbar((length(t)+tid)/2/length(t)*scale+offset,wb_handle);
      if (probs(sort_ids(tid)) > th_prob)
            image=pocs(image,t{tid},ds,shs(tid,:));
      end
    end
elseif strcmp(method,'average')
  image=zeros(size(image));
  for tid=1:length(t)
      waitbar((length(t)+tid)/2/length(t)*scale+offset,wb_handle);
     if (probs(tid) > th_prob)
            sh_image=posun_snimku(kron(t{tid},ones(ds)),-shs(tid,:));
            image=image+sh_image;
     end
  end
  if length(find(probs>th_prob))>0
      image=image/length(find(probs>th_prob));
  else
      image=orig_image; % can't find suitable one
  end
else
  error('method is not recognized');
end
```

sr_one_step.m

```
function
[image,probs,shs,scores]=sr_one_step(image,t,method,ds,sr,sigma,th_prob,orig_sh)

if ~exist('th_prob','var')
    th_prob=0.9;
end
```

```
shs=zeros(length(t),2);

orig_image=image;
if exist('orig_sh','var') % cheat mode
   scores=[];
   for tid=1:length(t)
     shs(tid,:)=orig_sh(tid,:);
     probs(tid)=1;
   end
else
  for tid=1:length(t)
        [tmp_sh,tmp_prob,tmp_scores]=subpixel_register(image,t{tid},ds,sr,sigma);
        probs(tid)=tmp_prob;
        shs(tid,:)=tmp_sh;
        scores{tid}=tmp_scores;
  end
end

if strcmp(method,'max_pocs') % only project to the one with highest score
    [dummy,max_id]=max(probs);
    image=pocs(image,t{max_id},ds,shs(max_id,:));
elseif strcmp(method,'pocs')
  for tid=1:length(t)
     if (probs(tid) > th_prob)
            image=pocs(image,t{tid},ds,shs(tid,:));
     end
  end
elseif strcmp(method,'average')
  image=zeros(size(image));
  for tid=1:length(t)
     if (probs(tid) > th_prob)
            sh_image=shift_image(kron(t{tid},ones(ds)),-shs(tid,:));
            image=image+sh_image;
     end
  end
  if length(find(probs>th_prob))>0
      image=image/length(find(probs>th_prob));
  else
      image=orig_image; % can't find suitable one
  end
else
  error('method is not recognized');
end
```

subpixel_register.m

```
function [sh,prob,score]=subpixel_register(hr_image,lr_image,ds,sr,sigma)

if ~exist('sigma','var')
    sigma=40;
end
if sigma < 0
    sigma=40;
end
sid=1;
sranges={};
sigma2=sigma*sigma;
for id=sr(1):sr(2) % search range
    for jd=sr(3):sr(4)
        sranges{sid}=[id jd];
        t=gen_shift_downsample_image(hr_image,ds,[id jd]);
        score(sid)=prod(prod(exp(-(t-lr_image).^2/(2*sigma*sigma))));

        sid=sid+1;
    end
end
```

```
[dummy,mid]=max(score);
prob=dummy/sum(score);
sh=sranges{mid};
```

vystredeni_snimku.m

```
% pomocna funkce pro vystredeni snimku
function [im2_reg,sh,f]=vystredeni_snimku(cim1,cim2)

f=zeros([size(cim1,1) size(cim1,2)]);       % matice naplnena nulami
for c=1:3
    im1=double(cim1(:,:,c));
    im2=double(cim2(:,:,c));
    tmp=fft2(im1).*fft2(flipud(fliplr(im2)));
    f=f+fftshift(abs(ifft2(tmp)));
end
[~,shid]=max(f(:));
sh(1)=(mod(shid-1,size(im1,1))+1)-size(im1,1)/2;
sh(2)=ceil((shid-1)/size(im1,1))-size(im1,2)/2;
im2_reg1=posun_snimku(cim2(:,:,1),sh);
im2_reg2=posun_snimku(cim2(:,:,2),sh);
im2_reg3=posun_snimku(cim2(:,:,3),sh);

im2_reg=zeros(size(cim1));
im2_reg(:,:,1)=im2_reg1;
im2_reg(:,:,2)=im2_reg2;
im2_reg(:,:,3)=im2_reg3;
```

Příloha E – Návod k obsluze Super-resolution imaging programu

1. **Přejmenování snímků**
   - před spuštěním samotného programu super.m je nutné přejmenovat vstupní snímky do příslušného tvaru – obrazek1 až obrazekN.

2. **Nahrání sekvence snímků**
   - pro nahrání snímků do programu stačí vybrat pouze složku, která obsahuje příslušné snímky. Snímky musí být přejmenovány viz bod 1.

3. **Otočení snímků (je-li potřeba)**
   - neotočil si uživatel snímky v jiném programu, může tak učinit zde.

4. **Vystředění jednotlivých snímku**
   - slouží k vystředění jednotlivých snímků pokud došlo při jejich snímání k posunu mezi jednotlivými snímky.

5. **Vybrání lokální části snímků pomocí myši**
   - kurzorem myši si uživatel vybere konkrétní část, na kterou chce aplikovat metodu Super-resolution imaging.

6. **Přiblížení lokální části snímku**
   - po vybrání lokální části (je ohraničena barevným obdélníkem) musí uživatel tuto část přiblížit. Je zde na výběr přiblížení 2x a 4x. Přiblížená část snímku se zobrazí v novém okně.

7. **Spuštění metody super-resolution imaging**
   - po spuštění může uživatel sledovat průběh na bar grafu. Po dokončení SR se výsledný obraz se super rozlišením objeví v novém okně.