VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

METODY PRO ZVÝŠENÍ BITOVÉ HLOUBKY FOTOGRAFIÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

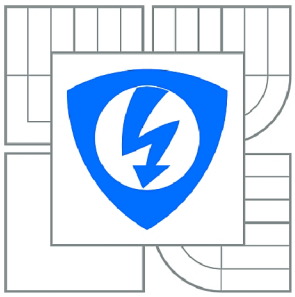
AUTOR PRÁCE
AUTHOR

PAVEL ZÁVIŠKA

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

METODY PRO ZVÝŠENÍ BITOVÉ HLOUBKY FOTOGRAFIÍ

METHODS FOR INCREASING BIT-DEPTH IN IMAGES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

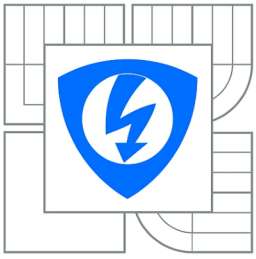
AUTOR PRÁCE — PAVEL ZÁVIŠKA

AUTHOR

VEDOUCÍ PRÁCE — Mgr. PAVEL RAJMIC, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Pavel Záviška ***ID:*** 154913
Ročník: 3 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Metody pro zvýšení bitové hloubky fotografií

POKYNY PRO VYPRACOVÁNÍ:

Bitová hloubka fotografií je obvykle 8 bitů na pixel a barvu. To však nezřídka bývá málo, nízká bitová hloubka typicky dělá potíže v případě, že snímek je silně podexponován.

Nastudujte z dostupných zdrojů přístupy pro řešení této problematiky. Důležité zástupce metod pro zvýšení bitové hloubky popište, porovnejte jejich vlastnosti teoreticky, implementujte a porovnejte prakticky, včetně hodnocení kvality výsledku.

Nastudujte problematiku řídkých reprezentací signálů a aplikujte ji na obrazy v kontextu zvyšování bitové hloubky. Porovnejte opět z různých hledisek se standardními přístupy.

DOPORUČENÁ LITERATURA:

[1] Elad, M. Sparse and redundant representations. Springer, 2010.

[2] Hou, L., Ji, H., Shen, Z. Recovering Over-/Underexposed Regions in Photographs. SIAM Journal on Imaging Sciences, Vol 6, No 4, p. 2213-2235, 2013.

[3] Jiří Žára, Bedřich Beneš, Jiří Sochor, Petr Felkel, Moderní počítačová grafika, druhé vydání, Computer Press, 2005, ISBN 80-251-0454-0

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 2.6.2015

Vedoucí práce: Mgr. Pavel Rajmic, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

ABSTRAKT

Cílem této bakalářské práce je aplikace problematiky řídkých reprezentací na úlohu zvyšování bitové hloubky fotografií. Jsou popsány základní metody pro zvyšování bitové hloubky a následně je představena metoda využívající řídké reprezentace obrazového signálu. Jednotlivé metody jsou naprogramovány v prostředí Matlab. Výsledky implementovaných metod jsou porovnány pomocí objektivních ukazatelů PSNR, SSIM i subjektivně pomocí online dotazníku.

KLÍČOVÁ SLOVA

Řídké reprezentace, Douglas-Rachfordův algoritmus, proximální dělení, zvýšení bitové hloubky, bitová hloubka, barevná hloubka, zpracování obrazu, falešné kontury

ABSTRACT

The goal of this bachelor thesis is the application of an issue of sparse representations on the task of increasing bit-depth in images. Standard methods for bit-depth expansion are described, subsequently a method using sparse representation of an image signal is introduced. Methods are programmed in the Matlab enviroment. The results of the methods implemented are compared by using PSNR, SSIM objective indexes and subjectively using an online questionnaire.

KEYWORDS

Sparse representations, Douglas-Rachford algorithm, proximal splitting, bit-depth expansion, bit-depth, color-depth, image processing, false contours

ZÁVIŠKA, Pavel *Metody pro zvýšení bitové hloubky fotografií*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 82 s. Vedoucí práce byl Mgr. Pavel Rajmic, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Metody pro zvýšení bitové hloubky fotografií“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Mgr. Pavlu Rajmicovi, Ph.D. za odborné vedení, věnovaný čas, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V této bakalářské práci se věnuji obecně metodám pro zvýšení bitové hloubky fotografií. V první kapitole je vysvětlena obecná teorie digitálního obrazu. Jsou objasněny pojmy jako histogram, barevná paleta a barevná (bitová) hloubka. Dále je detailněji popsána bitová hloubka, jsou uvedeny a pojmenovány různé používané bitové hloubky. Podrobně jsou také vysvětleny důvody zvyšování bitové hloubky. Stručně je zmíněna také technologie HDR fotografie. Dále je nastíněn pojem dithering, na jehož základě je postavena jedna z metody pro zvýšení bitové hloubky. Na závěr kapitoly jsou uvedeny metody pro objektivní hodnocení kvality obrazu, konkrétně ukazatele PSNR a SSIM.

Druhá kapitola shrnuje známé metody pro zvyšování bitové hloubky fotografií, které jsou v práci seřazeny od jednodušších až po komplexnější metody. Každá metoda je stručně přestavena a matematicky je popsán algoritmus následován závěrečným zhodnocením funkčnosti metody.

Protože cílem této práce je vytvořit metodu pro zvýšení bitové hloubky využívající řídkou reprezentaci signálu, je třetí kapitola věnována této problematice. V úvodu kapitoly jsou objasněny základní pojmy a značení, dále je představen aditivní model signálu a uvedeny podmínky nutné k nalezení řídkého řešení.

Kapitola Teoretické řešení matematicky specifikuje dekvantizační úlohu s využitím řídké reprezentace signálu. Dále tuto základní úlohu transformuje do tvarů uchopitelnějších pro reálné zpracování a představuje možnosti řešení tohoto problému – pomocí lineárního programování nebo pomocí algoritmů proximálního dělení.

Pátá kapitola pak navazuje na předchozí a popisuje samotnou realizaci jednotlivých algoritmů. Obsaženy jsou i popisy funkcí, které jsou součástí přiloženého CD. Strukturně je kapitola rozdělena na tři části – 1D signál, 2D signál a Reálný obraz.

Poslední, šestá kapitola pak shrnuje výsledky testovaných metod pro zvyšování bitové hloubky fotografií pomocí objektivních i subjektivních hodnocení. V úvodu každé části je stručně popsán princip testování a dále jsou výsledky zobrazeny pro lepší přehlednost ve sloupcových grafech.

1 TEORETICKÝ ÚVOD

V teoretickém úvodu se budu věnovat digitálnímu obrazu jako takovému. Popíšu druhy digitálních obrazů (vektorový a bitmapový). Zaměřím se hlavně na bitmapový obraz, který detailněji popíši a uvedu jeho výhody a nevýhody. Větší pozornost věnuji rozboru bitové hloubky fotografií a uvedu, jaké jsou důvody větší bitové hloubky, než standardních 8 bitů na pixel a barvu. Na závěr teoretického úvodu se krátce zmíním o ditheringu obrazu, popíšu histogram a uvedu zde nejčastěji používané jednotky pro objektivní měření kvality obrazu – PSNR a SSIM.

1.1 Digitální obraz

Definice obrazu není absolutně zavedena, ale liší se podle své aplikační oblasti. Obecně můžeme chápat obraz jako jakékoli grafické dvourozměrné vyjádření. Formálně můžeme použít matematický model, kterým je spojitá funkce dvou proměnných – obrazová funkce [35]:

$$z = f(x, y). \tag{1.1}$$

Digitálním obrazem pak rozumíme binární reprezentaci dvojrozměrného obrazu [9]. Obraz byl tedy navzorkován, nakvantován a posléze vyjádřen pouze jedničkami a nulami.

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

$$f : (\langle x_{\min}, x_{\max} \rangle \times \langle y_{\min}, y_{\max} \rangle) \to H. \tag{1.2}$$

Z principu můžeme digitální obraz rozdělit na dvě hlavní kategorie a to vektorový obraz [32] a rastrový obraz. V této práci se budu věnovat pouze obrazovým rastrovým, které v kapitole 1.1.1 podrobněji popíšu a uvedu jejich výhody a nevýhody.

1.1.1 Rastrový (bitmapový) obraz

Rastrový (též nazýván bitmapový) obraz je tvořen maticí jednotlivých bodů – tzv. pixelů (z anglického *picture element*). Této matici se také často říká bitová mapa nebo zkráceně bitmapa [35].

Důležitým parametrem bitmapového obrazu je jeho velikost, udávající počet pixelů, někdy také označována jako rozlišení obrazu. Obecně bývá uvedena ve formě součinu (např. 1920×1080) nebo v megapixelech (Mpix, Mpx, MP), tedy celkový

počet pixelů v milionech. Samotný počet pixelů ale nic nevypovídá o skutečných rozměrech obrazu. Proto byly zavedeny jedotky jako počet pixelů na palec (ppi – z anlického *pixels per inch*), počet bodů na palec (dpi – *dots per inch*), popř. počet bodů na centimetr (dpcm), která se ale v praxi prakticky nepoužívá.

Každý pixel v digitálním obraze má nejen svoji přesnou polohu, ale i údaj o barvě. Tento údaj závisí mimo jiné i na tom, v jakém barevném modelu je vyjádřen. Pokud jde o obrázek ve stupních šedi, číslo udává odstín šedi. Pokud je to barevný obrázek např. v RGB, každý jeho kanál nese údaj o tom, kolik je v celé barvě pixelu zastoupena červená, zelená a modrá složka. Takovýto mód, kdy jsou barvy uvedeny přímo v pixelu se nazývá Direct color. Typicky bývá tento mód použit například v obrazech formátu JPEG.

Existuje ale také mód, kdy barvy nejsou uloženy přímo, ale jsou uloženy v tzv. barevné paletě (*color palette*). Takovýto mód nazýváme indexový, protože pixely obrazu nesou pouze index barvy v paletě barev. Indexový mód je typický pro formáty PNG a GIF [35, 27]. Více o barevných paletách v kapitole 1.3.

Počet bitů, v kolika je barva pixelu vyjádřena, nazýváme bitová, popř. barevná hloubka. Bitová hloubka bude ještě detailněji probrána dále v kapitole 1.4.

Výhodou bitmapové grafiky je především snadnost pořízení (digitálním fotoaparátem, scanerem...) a relativní snadnost dodatečných úprav. Nevýhodou je pak nemožnost změny velikosti obrázku, aniž by zůstala zachována kvalita (při velkém zvětšení již je patrný rastr) a pro vysoká rozlišení také velká datová náročnost [5, 22].

1.2 Histogram

Jedním z možných interpretací rozložení barev v obraze je histogram. Je to statistický útvar, který zobrazuje četnost nebo také pravděpodobnost rozložení dané veličiny v grafu [22].

V histogramu máme několik možností zobrazení. Pokud je zkoumaný obraz šedotónový, veličina na ose x bude jas. Pokud ovšem budeme mít barevný obraz, histogram se bude skládat ze tří složek – RGB. Jednotlivé složky si v histogramu můžeme zobrazit každou zvlášť. Lze také zobrazit jejich prolnutí do jednoho histogramu, popř. zobrazit jejich průměr.

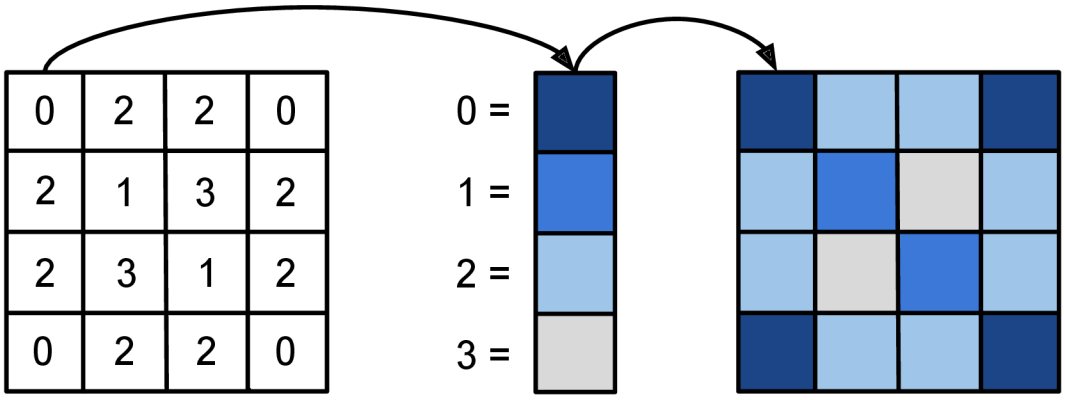
Počet hodnot na ose x závisí na bitové hloubce kanálu. Každá hodnota tedy udává jinou barvu, popř. jinou hodnotu jasu. Pro 8bitovou hloubku je to 256 hodnot na kanál.

Velmi často se v digitální fotografii používá histogram jasový. Z něj lze snadno poznat podexponovaná a přeexponovaná místa zkoumané fotografie [35].

Příklad histogramu lze vidět na obrázcích 1.4b, 1.6b a 1.5b.

1.3 Barevná paleta

Barevná paleta je používána v tzv. indexovém módu rastrového obrazu, kdy jednotlivý pixel nenese hodnotu barvy, ale udává pouze index, který ukazuje na hodnotu v paletě barev, viz obrázek 1.1. To s sebou nese určité výhody, ale zároveň i nevýhody. Výhodou je určitě nízká paměťová náročnost, pokud generujeme rastrový obraz o malém počtu barev. Nevýhodou potom bývá nízký počet barev u komplexnějších obrazů. Z toho vyplývá, že indexový mód se nehodí pro reálné fotografie, ale je ideální pro digitální grafiku, screenshoty, atp.



Obr. 1.1: Příklad obrázku v indexovém módu s 2bitovou barevnou paletou

Nabízí se otázka, jak vhodně nastavit barevnou paletu. Existují dvě varianty – přednastavené palety (též univerzální) a přízpůsobené palety. Přednastavené palety bývají zpravidla použitelné pro jakýkoli obraz. Z důvodu své univerzálnosti ale nesou barvy, které v daném obraze nemusí být zastoupeny vůbec a naopak barvy, které se v obraze objevují hodně naopak nemusí být obsaženy v paletě. Tento problém právě řeší palety přizpůsobené obrazu. K jejímu vytvoření je potřeba nejprve nový obraz analyzovat a paletu uložit spolu s obrazem [35].

1.3.1 Přednastavené palety

Mezi nejpoužívanější přednastavené palety patří barevná paleta 3–3–2, která je tzv. neuniformní. Znamená to, že každý pro každý kanál má vyhrazený jiný počet bitů. Paleta 3–3–2 má tedy pro červený a zelený kanál vyhrazeny 3 bity a pro modrý kanál pouze 2 bity, což ale nevadí, protože na modrou barvu je lidské oko nejméně citlivé. Výhodou je, že se tato paleta dá vyjádřit jedním bytem a přináší tak třetinovou úsporu velikosti oproti 24bitové paletě 8–8–8 [22, 27].

Další, v minulosti velmi používanou paletou, je tzv. Web-safe color (6×6×6 – používá tedy 216 barev). Tato paleta pochází ještě z doby, kdy některé zařízení,

které se připojovaly na web, uměly zobrazit pouze 256 barev. Proto vznikla tato paleta, aby všechna zařízení zobrazovala webový obsah ve stejných barvách. V dnešní době, kdy už prakticky všechna zařízení umí obrazit 16,7 milionů barev, tato paleta postrádá smysl [27].

1.3.2 Palety přizpůsobené obrazu

Z důvodů uvedených v 1.3 je vhodné použít pro obraz palety, které jsou obrazu „ušité na míru“. To znamená, že paleta maximálně respektuje barvy obsažené v obraze a z nich vytvoří speciální paletu pro konkrétní obraz.

V této kapitole nebudu uvádět příklady používaných přizpůsobených palet, ale spíše popíšu základní algoritmus pro výpočet takovéto palety.

Nejdříve se ze vstupního obrazu vypočítá trojrozměrný histogram. Při vstupním obrazu o bitové hloubce 24 bpp (cca 16,7 milionů barev) by měl histogram 16,7 milionů buněk pro uložení četnosti, což by bylo velice paměťově náročné. Proto se 3 nejméně významné bity vypouštějí a histogram má potom $32^3 = 32\,768$ buněk.

Poté se histogram rozdělí na zadaný počet oblastí. Ideálně tak, aby každá oblast obsahovala přibližně stený počet pixelů. Konkrétní hodnota barvy v paletě potom odpovídá průměru (těžišti) barev vstupních pixelů v dané oblasti [27].

1.4 Bitová hloubka

V této kapitole podrobněji vysvětlím, co je to bitová hloubka fotografie, uvedu příklady používaných bitových hloubek v počítačové grafice a pokusím se vysvětlit, proč někdy bitová hloubka 8 bitů na kanál není dostačující a je potřeba ji zvýšit.

1.4.1 Popis bitové hloubky

Jak už bylo zmíněno v kapitole 1.1.1, bitová hloubka udává, kolika bity je vyjádřena výsledná barva pixelu. Nejčastěji bývá uváděna jako počet bitů na pixel (bpp – z anglického *bits per pixel*). Může být udávána také jako počet bitů na barvu (bpc – *bits per color*), počet bitů na kanál (bpc – *bits per channel*) nebo počet bitů na vzorek (bps – *bits per sample*) [7].

Je tedy zřejmé, že čím vyšší bitová hloubka, tím větší škálu barev budeme moci zobrazit (tzv. tonální rozsah, popř. dynamický rozsah) a tím přesněji bude digitální obraz nakvantován. Zároveň bude ale jeho datová velikost větší.

1.4.2 Nejčastěji používané barevné hloubky

Pokud bychom vyjádřili barvu pixelu pouze jedním bitem, nabýval by hodnoty pouze 0 nebo 1, tedy bílá nebo černá. Takovýto obraz nazýváme monochromatický. Černobílé obrazy, neboli obrazy vyjádřené ve stupních šedi, bývají nejčastěji vyjádřeny 8 bity; to znamená, že mají 256 stupňů šedi. Barevné obrazy v prostoru RGB mívají typicky 8 bitů na kanál, tedy 24 bitů na pixel, což dává celkem 16 777 216 barevných kombinací.

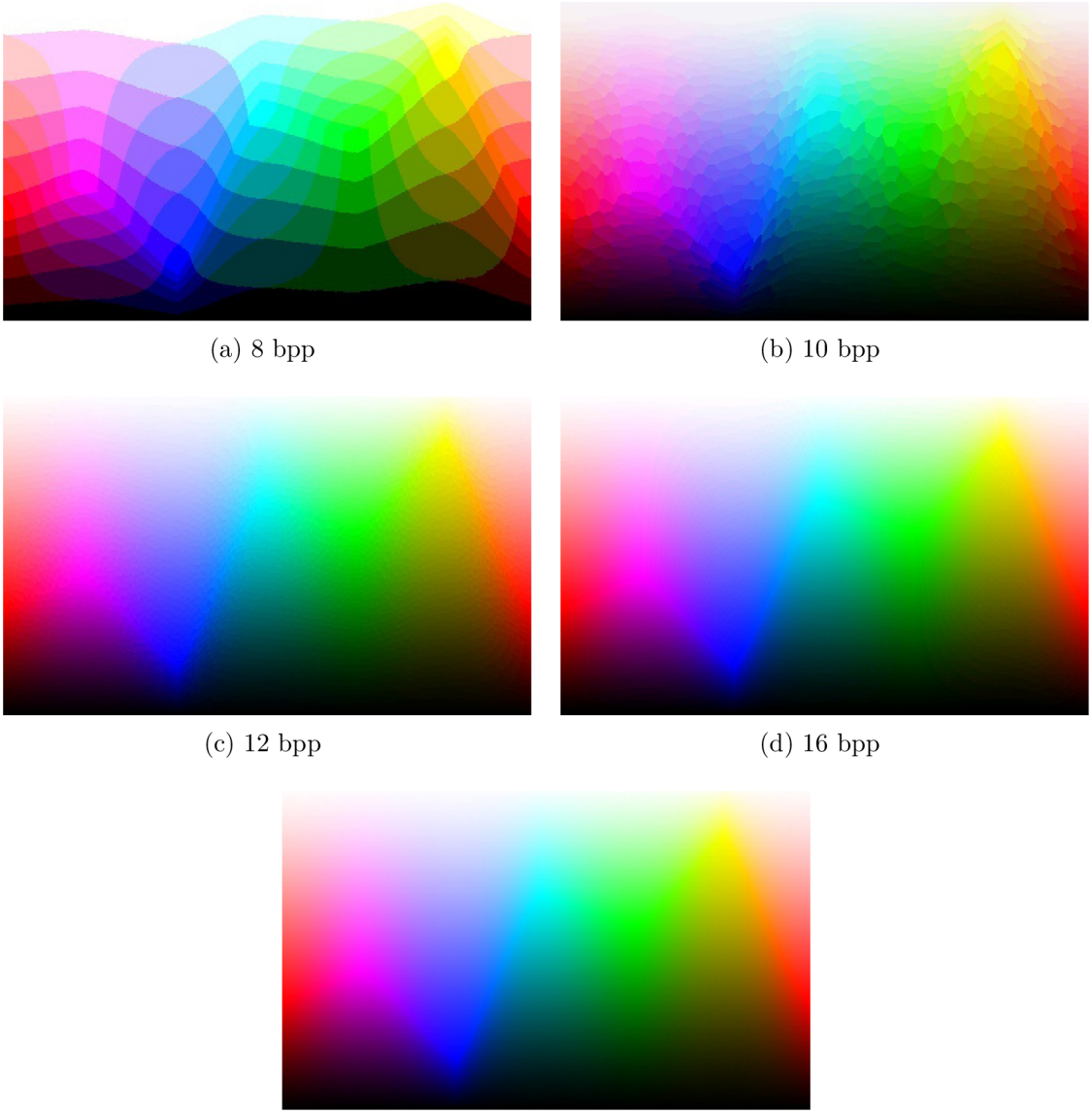
V tabulce 1.1 je uvedeno několik základních a nejčastěji používaných bitových hloubek při práci s fotografiemi, celkový počet barev a obecné pojmenování. Pokud bude pojmenování vyjádřeno zkratkou, všechny zkratky budou vysvětleny v seznamu zkratek na konci této práce.

Tab. 1.1: Nejčastěji používané barevné hloubky v počítačové grafice [22, 4]

Bitová hloubka [bpp]	počet barev	běžný název
1	$2^1 = 2$	monochrome
2	$2^2 = 4$	CGA
4	$2^4 = 16$	EGA
8	$2^8 = 256$	VGA
15	$2^{15} = 32\,768$	Low color
16	$2^{16} = 65\,536$	XGA, High color
24	$2^{24} = 16\,777\,216$	SVGA, True color
32	$2^{32} = 4\,294\,967\,296$	Super true color
48	$2^{48} = 281\,474\,976\,710\,656$	Deep color

Na sérii obrázků 1.2 můžeme vidět 5 různých případů bitové hloubky pro celý barevný model RGB. Zatímco 24bitový obrázek (e) má naprosto plynulé přechody a 16bitový obrázek (d) je od (e) odlišný jen minimálně, na 12bitovém (c) už při detailnějším pohledu poznáme známky posterizace. Na obrázcích (a) a (b) už je jev posterizace patrný i z letmého pohledu.

Na další sérii obrázků 1.3 je ukázán vliv bitové hloubky na reálnou fotografii. Na obrázku (a) můžeme vidět monochromatický obrázek – upravený tzv. ditheringem (viz kapitola 1.5). Obrázek 2bitový (b) je vyjádřen 4 barvami, objevuje se tam tedy bílá, světle šedá, tmavě šedá a černá. Na 4bitovém (c) jde stále vidět efekt posterizace a přechody nejsou plynulé. Na 8bitovém (d) je to o něco lepší, ale při detailnějším pohledu jsou stále patrné hrubé přechody. Obrázek (e) s 24 bpp už má přechody prakticky plynulé.



(a) 8 bpp

(b) 10 bpp

(c) 12 bpp

(d) 16 bpp

(e) 24 bpp

Obr. 1.2: Barevný model RGB v závislosti na barevné hloubce. Převzato z [4].

1.4.3 Důvody zvyšování bitové hloubky

Průměrné lidské oko je schopno rozpoznat cca 10 milionů barev. Proto by se mohlo zdát, že 24bitová hloubka, která nám poskytuje 16,7 milionů barev, bude naprosto dostatečná a jakékoliv vyšší hodnoty bitové hloubky budou zbytečné. Toto platí pouze v případě, že fotografii nebudeme dodatečně nijak upravovat. Pokud se ale rozhodneme výslednou fotografii dále upravovat, především měnit kontrast fotografie, s velkou pravděpodobností bude docházet k tzv. posterizaci – jevu, kdy se místo jemných přechodů mezi odstíny objeví hrubé a viditelné skoky [23]. Těmto skokům

říkáme kontury nebo též falešné kontury (anglicky *false contours*). Příklad falešných kontur můžeme vidět např. na obrázku 1.2a nebo na obrázku 1.6a.

(a) 1 bpp

(b) 2 bpp

(c) 4 bpp

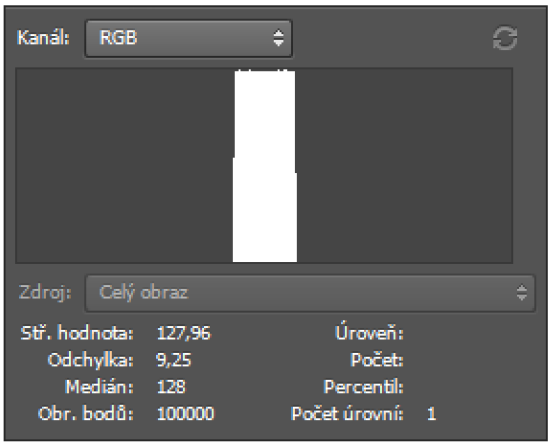
(d) 8 bpp

(e) 24 bpp

Obr. 1.3: Reálné fotografie listu v závislosti na bitové hloubce. Převzato z [7].

Představme si, že máme černobílou fotografii o nízkém kontrastu. V našem případě nám tuto fotografii bude demonstrovat plynulý přechod (tzv. gradient) od šedé (112, 112, 112) do (144, 144, 144), viz obrázek 1.4. Současně s obrázkem budeme sledovat i jeho histogram. Gradienty a histogramy byly vytvořeny v grafickém editoru Adobe Photoshop verze CS6.

(a) původní gradient



(b) histogram původního gradientu

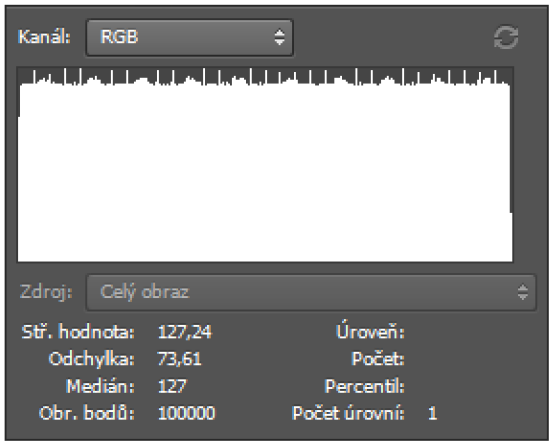
Obr. 1.4: Gradient od šedé (112, 112, 112) do (144, 144, 144) s histogramem

Na tomto gradientu následně provedu takové nastavení kontrastu, že se histogram „roztáhne“ do obou svých mezí a v ideálním případě nám vznikne gradient od černé (0, 0, 0) po bílou (255, 255, 255). Nejprve provedeme úpravu kontrastu pro obraz s bitovou hloubkou 16 bitů na kanál, což je 65 535 odstínů šedi.

Na obrázku 1.5 už vidíme upravený gradient. Na první pohled jsme dosáhli toho, čeho jsme chtěli. Tedy plynulý gradient od černé po bílou. Přes poměrně drastické zvýšení kontrastu zůstala plynulost gradientu zachována. Pouze roztřepené okraje histogramu poukazují na nepatrné narušení plynulosti přechodu.

Pokud ale tutéž změnu provedeme pro gradient v bitové hloubce 8 bitů (256 odstínů šedi), výsledek na obrázku 1.6 už bohužel není plynulý přechod, ale obraz, který má do plynulosti přechodů hodně daleko. Pokud se podíváme na histogram, vidíme charakteristiku připomínající hřeben, tedy nespojitou. Je to jednoduše proto, že pro plynulý přechod nemáme v 8bitové hloubce dostatečný počet barev.

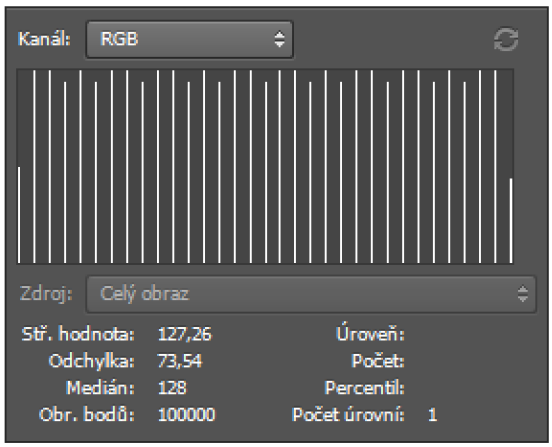
(a) 16bitový gradient se zvýšeným kontrastem.



(b) histogram 16bitového gradientu

Obr. 1.5: 16bitový gradient se zvýšeným kontrastem a jeho histogram

(a) 8bitový gradient se zvýšeným kontrastem.



(b) histogram 8bitového gradientu

Obr. 1.6: 8bitový gradient se zvýšeným kontrastem a jeho histogram

1.4.4 Kvantizace signálu

Při pořizování digitálního záznamu signálu (obrazového, zvukového, atd.) proces digitalizace probíhá ve třech krocích – vzorkování, kvantizace a kódování [28].

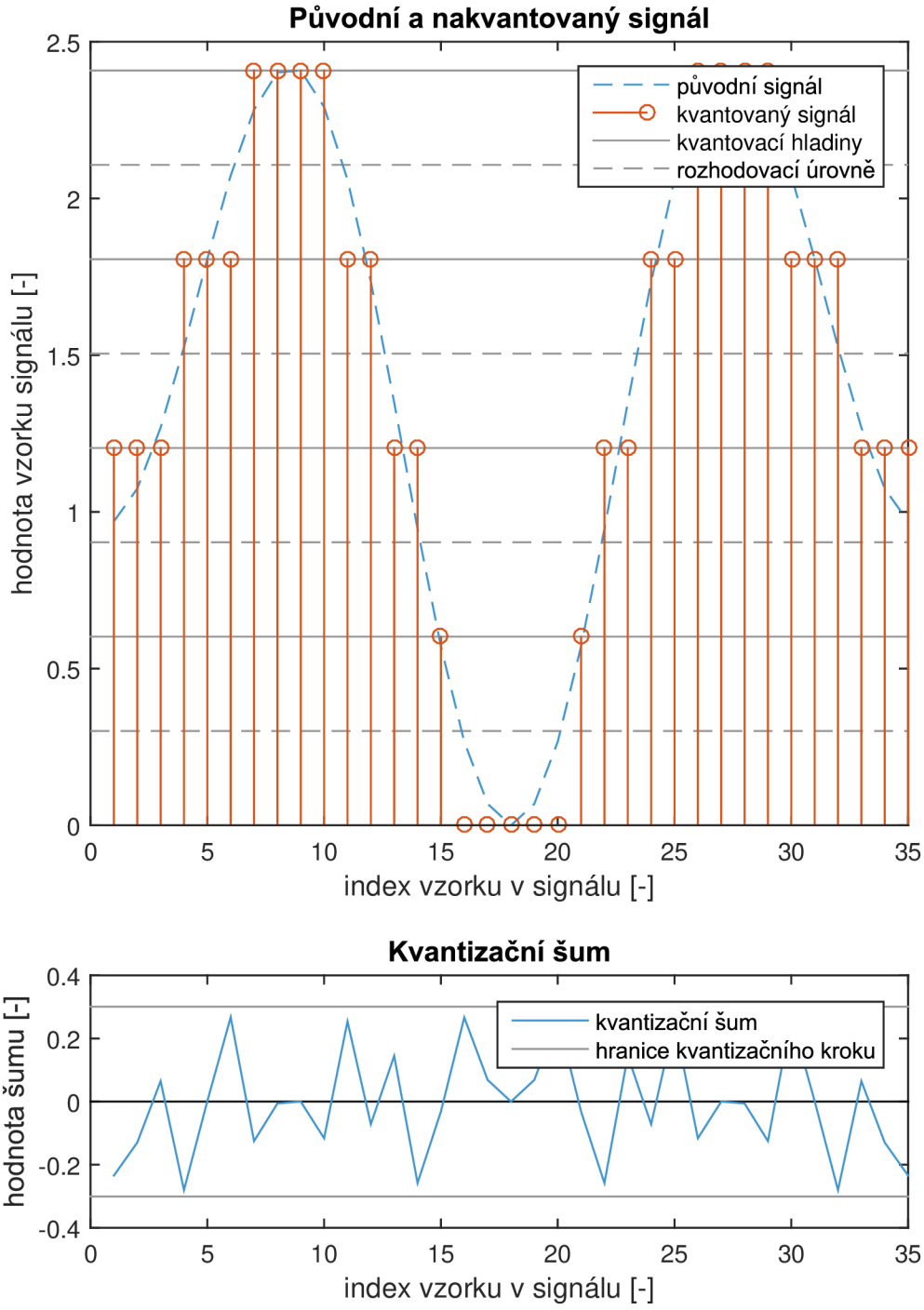
Vzorkování je proces, při kterém zachytíme jednotlivé vzorky signálu v konstantních časových intervalech a ze spojitého signálu tak získáme pouze diskrétní hodnoty. Převrácenou hodnotu časového intervalu mezi jednotlivými vzorky nazýváme vzorkovacím kmitočtem. Vzorkování můžeme rozdělit na uniformní a neuniformní. Uniformní používá konstantní vzorkovací interval, zatímco neuniformní používá proměnlivou velikost vzorkovacího intervalu, díky čemuž může lépe zohlednit rozložení hodnot původního signálu. V praxi se však častěji setkáváme se vzorkováním uniformním [28].

Protože se tato práce zabývá zvyšováním bitové hloubky, tedy dekvantizací, bylo by vhodné popsat zde podrobněji proces kvantizace, který je druhou fází při digitalizaci – předpokládá tedy vstupní signál už navzorkovaný. Při kvantizaci jsou přesné hodnoty signálu přiřazeny kvantizačním hladinám. Dochází tedy k určité ztrátě informace, která je v jistém smyslu nevratná. Mezi kvantizačními hladinami, ve většině případů uprostřed, leží rozhodovací úrovně. Vzdálenost mezi kvantizačními hladinami, tzv. kvantovací krok, označíme α. Rozdíl mezi původním a nakvantovaným signálem pak nazýváme jako kvantizační šum.

Podle způsobu kvantizace rozdělujeme kvantování na lineární a nelineární. Lineární používá konstantní velikost kvantovacího kroku α, naopak nelineární používá nelineární vzdálenost kvantizačních hladin. Na obrázku 1.7 lze vidět srovnání původního signálu a signálu nakvantovaného pomocí uniformní lineární kvantizace, která ale není souměrná podle nuly. Níže je pak zobrazen kvantovací šum.

Úlohou dekvantizace pak myslíme inverzní postup, kdy se z už nakvantovaného signálu snažíme získat zpět původní signál nebo alespoň takový signál, který by byl původnímu signálu co nejblíže. Takováto úloha by bez jakékoliv předchozí znalosti o původním signálu postrádala smysl. Pokud o původním signálu víme nějakou informaci navíc, například to, že je řídký v nějaké bázi, můžeme se pokusit tento signál na základě této informace dekvantizovat.

Posledním krokem při digitalizaci záznamu je pak kódování, což spočívá v přiřazení každé nakvantované hodnotě jistý kód (např. BCD), především z důvodu menší přenosové kapacity [28].



Obr. 1.7: Princip vzorkování a kvantování signálu

1.4.5 HDR fotografie

Dalším případem, kdy se s velkou výhodou používá vysoká bitová hloubka jsou tzv. HDR (z anglického *High Dynamic Range*) fotografie [11].

Může totiž nastat případ, kdy fotografovaná scéna má vyšší dynamický rozsah, než jaký je fotoaparát schopen vyfotit (např. fotografování uvnitř tunelu, fotografie proti slunci, atp.). Výsledkem tedy bude fotografie s podexponovanými po případě přeexponovanými místy či dokonce mohou nastat oba případy zároveň.

Pro takovéto situace vznikla technologie HDR, která se dnes velmi často implementuje do fotoaparátů ve smartphonech. HDR funguje na principu, kdy jednu scénu vyfotí vícekrát – pokaždé s různě dlouhou expozicí.

Na snímku s krátkým expozičním časem tedy budou světlá místa správně exponovaná, ale ostatní budou podexponovaná. Naopak na snímku s dlouhým expozičním časem budou správně exponovaná tmavá místa, ale světlá místa budou přepálená. Technologie HDR pak vybere z několika snímků pouze správně exponovaná místa a ty pak sloučí do jedné fotografie.

Vznikne tedy fotografie, která má vysokou bitovou hloubku. Ta by ale byla špatně zobrazitelná na klasických displejích, proto se pomocí tzv. Tone Mappingu barevná hloubka komprimuje na klasických 8 bitů na pixel a barvu [11, 6].

Podobného efektu bez nutnosti pořizovat několik snímků z různými expozičními časy (např. z důvodu fotografování pohybujícího se objektu) lze dosáhnout i fotografováním do formátu RAW a potom úpravou expozice přímo v grafickém editoru.

Příklad reálného použití technologie HDR můžeme vidět na obrázku 1.8.

(a) S technologií HDR (b) Bez technologie HDR

Obr. 1.8: Fotografie vyfocená s efektem HDR a bez něj. Převzato z [14].

1.5 Dithering

Tato metoda přímo nesouvisí se zvyšováním bitové hloubky, ale používá se v opačných případech, kdy snižujeme barevnou hloubku fotografie z důvodu především nižší datové náročnosti, ale chceme zabránit (alespoň částečně) posterizaci.

Dither je záměrně aplikovaná forma šumu použitá ke znáhodnění kvantizační chyby [10]. Používá se jak ve zpracování audio signálů, tak i u fotografií. U digitálního obrazu je výsledkem ditheringu nahrazení barvy, která se v dané barevné paletě nevyskytuje, kombinací barev z barevné palety menšího rozsahu.

Dithering tedy využívá nedokonalosti lidského oka, kdy lidské oko nevnímá každý pixel obrazu, ale výsledný vjem získává ze shluků jednotlivých bodů. Například vnímání šedé barvy lze nahradit vhodnou kombinací bílé a černé barvy.

Existuje několik metod ditheringu, jak obraz zpracovat. Od nejjednoduššího prahování, přes metody využívající náhodný i maticový rozptyl až po pokročilejší metody, kdy nejznámější a pravděpodobně nejpoužívanější je metoda Floyd-Steinbergova [22].

1.6 Objektivní měření kvality obrazu

Hodnotit kvalitu obrazu můžeme pomocí subjektivních i objektivních ukazatelů. Subjektivní kritéria vycházejí z relativního hodnocení jednotlivých osob. Tyto testy bývají náročné pro zpracování, především potřebujeme velký počet testovacích subjektů. Proto se používají i objektivní ukazatele, které lze poměrně snadno vypočítat pomocí matematických vztahů.

V této kapitole se krátce zmíním o možnostech objektivního měření kvality obrazu, především se zaměřím na dvě nejpoužívanější metody – PSNR a SSIM.

1.6.1 PSNR

PSNR neboli Peak Signal-to-Noise Ratio, česky tedy špičkový poměr signálu ku šumu nebo špičkový odstup signálu od šumu je v dnešní době stále jednou z nejpoužívanějších metod pro objektivní měření kvality obrazu, ačkoli nezohledňuje způsob, jakým lidské oko vnímá obraz. Nepracuje s psychologickými ani fyzickými vlastnostmi lidského zraku.

Pro definici PSNR musíme nejdříve definovat střední kvadratickou chybu (anglicky *Mean Squared Error*, dále jen MSE). Střední kvadratická chyba patří mezi nejstarší a nejjednodušší ukazatele hodnocení kvality obrazu. MSE můžeme defino-

vat jako:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \|I(i,j) - K(i,j)\|^2, \tag{1.3}$$

kde I a K jsou dva černobílé obrazy o rozměrech $m \times n$.

PSNR má výhodou oproti MSE v použití logaritmického měřítka. Vyjadřuje věrnost obrazu, tedy podobnost testovaného obrazu oproti referenčnímu obrazu. Můžeme jej vyjádřit jako:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) = 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right), \tag{1.4}$$

kde MAX_I je maximální hodnota pixelu v obrázku (pro 8 bitů na kanál je to 255). Pro RGB obrázky je MSE suma přes všechny složky dělená třemi.

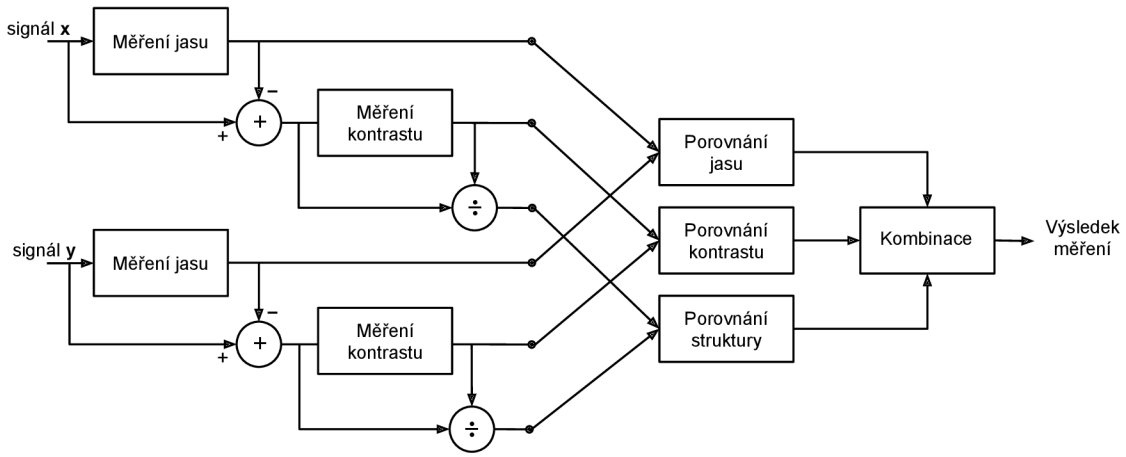
Platí, že čím vyšší hodnota PSNR, tím kvalitnější obrázek. Průměrná hodnota pro komprimované obrázky se pohybuje mezi 30 a 40 dB. Pro totožné obrazy je MSE nula, PSNR je tedy nedefinovatelné [21, 24].

1.6.2 SSIM

SSIM [34] neboli structural similarity vyjadřuje podobnost dvou obrazů. Tato metoda byla navržena, aby vylepšila stávající metody jako MSE a PSNR, které jak víme, nerespektují vlastnost lidského vnímání obrazu.

Obor hodnot funkce SSIM je interval $\langle 0, 1 \rangle$, kde 0 vyjadřuje nulový vztah obrazu s referenčním obrazem. Naopak 1 vyjadřuje maximální shodu – obrazy jsou totožné.

Tato metoda je založena na měření podobnosti podle tří složek obrazu, a to podobnost jasu, kontrasu a strukturální podobnost, viz obrázek 1.9. Pro barevné obrázky se ale obvykle počítá pouze na jasové složce.



Obr. 1.9: Diagram systému měření SSIM

Nejprve definujeme μ_x jako vážený průměr jasu:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{1.5}$$

Pro objektivní odhad kontrastu použijeme druhou odmocninu rozptylu

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}. \tag{1.6}$$

Pro srovnání sturktury stanovíme koeficient σ_{xy} který vypočteme podle vztahu

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y). \tag{1.7}$$

Pro porovnání jasu, kontrastu a sturktury pak definujeme rovnice

$$l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \tag{1.8}$$

$$c(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \tag{1.9}$$

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_1}{\sigma_x\sigma_y + C_3}, \tag{1.10}$$

kde $C_1 = (K_1 L)^2$ a $C_2 = (K_2 L)^2$. L je dynamický rozsah hodnot pixelů (255 pro 8bitovou šedotónovou fotografii) a konstanty $K_1 \ll 1$ a $K_2 \ll 1$. V definici byly použity $K_1 = 0{,}01$ a $K_2 = 0{,}03$.

SSIM můžeme vypočítat podle vztahu:

$$SSIM(x,y) = [l(x,y)]^\alpha \cdot [c(x,y)]^\beta \cdot [s(x,y)]^\gamma, \tag{1.11}$$

kde $\alpha > 0, \beta > 0$ a $\gamma > 0$ jsou parametry pro zvýšení relativní důležitosti z těchto tří komponent. Pro zjednodušení nastavíme $\alpha = \beta = \gamma = 1$ a $C_3 = C_2/2$. Po tom dostáváme finální tvar [34]

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}. \tag{1.12}$$

2 METODY

Tato část práce pojednává o metodách pro zvyšování bitové hloubky fotografií. Nejprve uvedu základní problém zvyšování bitové hloubky a potom jednotlivé metody rozeberu a popíšu jejich výhody a nevýhody.

Začnu nejjednoduššími metodami, fungujícími na principu tzv. one-to-one mappingu, tedy jednoduchým přepočtem hodnot jednotlivých pixelů. Tyto jednoduché metody patří do skupiny tzv. *pixel-based* metod, což znamená, že pro přepočet hodnoty pixelu nevyužívají prostorové informace okolí pixelu, ale pouze jeden pixel samotný. Poté popíšu také pokročilejší metody, které podávají lepší výsledky a nějakým způsobem řeší i odstranění falešných kontur, které vzniknou nízkou bitovou hloubkou fotografií.

2.1 Základní problém

Základní problém zvyšování bitové hloubky můžeme popsat následovně. Mějme p-bitový obraz X s rozsahem $[0, 2^p - 1]$ a q-bitový $q > p$ obraz Y s dynamickým rozsahem $[0, 2^q - 1]$. Našim cílem je tedy převést X na Y [25].

2.2 Zero padding

Nejjednodušší metoda pro zvýšení bitové hloubky se nazývá Zero padding (zkráceně ZP). Tato metoda spočívá v jednoduchém pronásobení hodnoty každého pixelu v obraze hodnotou 2^{q-p} [25]. Formálně tedy

$$[Y]_{i,j} = [X]_{i,j} \times 2^{(q-p)}. \tag{2.1}$$

Tato metoda, ač je prakticky nejjednodušší, není často používána. Problém je totiž v tom, že ZP si lze představit jako doplnění chybějících bitů nulami. Tím se způsobí, že hodnoty obrazu s vyšší bitovou hloubkou budou nejnižší možné a výsledný obraz pak bude mírně tmavší, než originál.

2.3 Multiplication by an ideal gain

Metoda Multiplication by an ideal gain, neboli vynásobení ideálním zesílením (dále jen MIG) je jakýmsi „vylepšením“ metody ZP. Opět se hodnoty jednotlivých pixelů pouze pronásobí, tentokrát koeficientem $\frac{2^q-1}{2^p-1}$ a výsledek se zaokrouhlí na nejbližší přirozené číslo [25], což způsobí rovnoměrné rozdělení hodnot v celém novém rozsahu.

Matematicky tedy můžeme zapsat:

$$[Y]_{i,j} = \text{Round}\left([X]_{i,j} \times \frac{2^q - 1}{2^p - 1}\right). \tag{2.2}$$

MIG, ač podává nepatrně lepší výsledky než ZP, není uspokojivá. Často se ale používá jako výchozí metoda pro pokročilejší metody, které nejprve touto metodou provedou samotné zvýšení bitové hloubky a následně různými způsoby odstraňují vzniklé kontury způsobené nízkou bitovou hloubkou.

2.4 Bit replication

Metoda Bit replication [31] (dále BR), neboli replikace bitů, vychází z matematické operace, kde pro $a \in \mathbb{R}$ a $b \in \mathbb{R}\backslash\{0, 1\}$ platí:

$$\frac{a-1}{b-1} = \frac{a}{b} + \frac{(a/b)-1}{b-1}. \tag{2.3}$$

Tento výraz můžeme pro N opakování zapsat obecně jako:

$$\frac{a-1}{b-1} = \sum_{i=1}^{N} \frac{a}{b^i} + \left(\frac{(a/b^N)-1}{b-1}\right). \tag{2.4}$$

V našem případě nahradíme $a = 2^p$ a $b = 2^q$ a ideální zesílení (ideal gain) můžeme vyjádřit jako:

$$G = \sum_{i=1}^{N} 2^{q-ip} + \left(\frac{2^{q-Np} - 1}{2^p - 1}\right). \tag{2.5}$$

Z tohohle vyjádření potom můžeme definovat přibližné zesílení (gain) jako

$$\hat{G} = \sum_{i=1}^{N} 2^{q-ip}, \tag{2.6}$$

a chybu zesílení pak jako

$$E = \frac{1 - 2^{q-Np}}{2^p - 1}. \tag{2.7}$$

Přibližný gain $\hat{G}$ pak může být jednoduše implementován jako replikace vstupní posloupnosti bitů, protože jakákoli hodnota $\hat{G}$ je mocnina dvou a násobení je tak ekvivalentní bitovému posunu.

Článek [31] se pak dále zabývá optimalizací opakování period. Je logické, že nejmenší chybu replikace dostaneme tehdy, když q a p budou dělitelné. V případě, že tato možnost nenastane (např. bychom chtěli zvýšit bitovou hloubku obrazu z 6 bpp na 8 bpp), pak při dvou opakováních ($N = 2$), dostaneme posloupnost 12 bitů, přitom jich požadujeme pouze 8. Z původního článku plyne, že jakékoli zaokrouhlování pouze zvýší průměrnou chybu, proto je lepší přebytečné bity prostě zahodit [31].

Tuto jednoduchou metodu pak můžeme matematicky zapsat jako:

$$[X]_{i,j} = x_{p-1}x_{p-2}\ldots x_1x_0; \quad [Y]_{i,j} = x_{p-1}x_{p-2}\ldots x_1x_0x_{p-1}x_{p-2}\ldots \tag{2.8}$$

Výhodou této metody např. oproti MIG je jednoduchost výpočtu, protože BR se dá hardwarově snadno implementovat jako posuvný registr. Reálné rozdíly metody BR oproti ZP nebo MIG jsou však lidským okem prakticky nepostřehnutelné, protože i když metoda vypočítá nepatrně odlišné hodnoty, stále funguje na principu one-to-one mappingu a tedy i přes vyšší bitovou hloubku je počet barev v obraze stále stejný a problém efektu posterizace přetrvává.

2.5 Gamma expansion

Další metoda pro zvýšení bitové hloubky byla představena v článku [1], který se zabývá především zobrazením LDR (*Low Dynamic Range*) obrázku na HDR displejích. Tato metoda byla v tomto článku použita pro zvýšení bitové hloubky LDR obrazu, aby se dynamický rozsah shodoval s rozsahem HDR monitoru. Jako Gamma expansion byla tato metoda nazvána až v [25], kde byla použita pro zvýšení bitové hloubky podexponovaného obrazu. Metodu můžeme formálně zapsat jako:

$$Y = k \left(\frac{X - X_{\min}}{X_{\max} - X_{\min}}\right)^{\gamma}, \tag{2.9}$$

kde k reprezentuje nejvyšší možnou hodnotu Y (pro výslednou 10bitovou hloubku by tato hodnota byla $2^{10} - 1 = 1023$), hodnoty $X_{\min}$ a $X_{\max}$ představují minimální a maximální hodnotu jasu původního LDR obrázku a konečně γ determinuje nelinearitu škálování.

Exponent γ udává, jak se bude střední hodnota jasu obrázku měnit oproti ostatním pixelům. Pro $\gamma = 1$ budou všechny pixely škálovány lineárně. Naopak pro $\gamma > 1$ bude výsledná hodnota jasu relativně tmavší a pro $\gamma < 1$ bude relativně světlejší, čehož se dá využít u podexponovaných obrázků.

2.6 Spatial Varying Filter

Jedny z prvních metod, kterými se řešily kontury vzniklé nízkou bitovou hloubkou, byly kombinace ZP, MIG, popř. BR a nějakého filtru. Z prvních tří možností bylo experimentálně zjištěno, že nejlepší výsledky podává metoda MIG, která hodnoty nové bitové hloubky rovnoměrně rozdělí do celého nového rozsahu hodnot. Vzniklé kontury se potom rozmazaly filtrem typu dolní propust. Tahle možnost ale nepřináší dobré výsledky, protože spolu s konturami se rozmaže celý obrázek, což rozhodně

není požadovaný efekt. Druhou a přijatelnější možností je prostorově proměnný filtr (z anglického *Spatial Varying Filter* – SVF) [2].

Metoda pro zvýšení bitové hloubky používající právě filtr SVF funguje následovně. Nejprve se zvýší bitová hloubka obrázku pomocí metody MIG 2.3. Na tento obraz se pak aplikuje filtr podle rovnice 2.10.

$$X_{\text{filtered}}(i,j) = \sum_{k=i-1}^{k=i+1} \sum_{l=j-1}^{l=j+1} a_{ij}(k,l)x(k,l), \tag{2.10}$$

kde $x(i,j)$ je hodnota pixelu na pozici (i,j) a $a_{ij}(k,l)$ je váha přidělená pixelu podle:

$$a_{ij}(k,l) = f(|x(k,l) - \hat{x}(i,j)|), \tag{2.11}$$

kde $\hat{x}(i,j)$ je průměrná hodnota pixelu a jeho sousedních pixelů a

$$f(z) = \left(1 - \frac{z}{2^q - 1}\right) \text{ pro } z = 0, 1, \ldots, 2^q - 1, \tag{2.12}$$

kde q je cílový počet bitů obrazu [18].

Metoda používající MIG a SVF je pravděpodobně nejlepší kombinací z jednoduchých metod pro zvýšení bitové hloubky a různých filtrů. Přesto je výsledek metody rozmazaný (sice méně, než při použití dolní propusti, ale i tak je rozmazání viditelné) a kontury jsou stále patrné. Na této metodě ovšem staví metoda ABDE, kterou si v další kapitole představíme.

2.7 Adaptive Bit-Depth Expansion method

Nyní se už dostáváme ke komplexnějším metodám, které se věnují vzniklým konturám v důsledku nízké barevné hloubky a dokáží je relativně uspokojivě odstranit. Metoda Adaptive Bit-Depth Expansion (dále jako ABDE) byla publikována v [18] v roce 2008.

Tato metoda se skládá ze dvou částí. V první části se provede zvýšení bitové hloubky pomocí metody Zero-padding (viz část 2.2), ke které se následně přidá adaptivně stanovený nezáporný posun (v originále *offset*) pro využití všech možných hodnot z nového dynamického rozsahu. Tato část bývá v literatuře označována také jako ABDE-P1. Druhá část se potom hlavně soustředí na dodatečné odstranění kontur pomocí detekce kontur v původním obraze a aplikování filtru typu dolní propust pro odstranění kontur.

Část 1 (ABDE-P1) – zvýšení bitové hloubky
Při zvyšování bitové hloubky například z 4bitové fotografie X na 6bitovou fotografii Y, všechny možné hodnoty výsledku jsou $4X$ plus hodnota posunu D, která v tomto

případě bude mezi 0 a 3. Můžeme jednoduše dokázat že výsledky metod ZP, BR a MIG jsou prakticky totožné, mění se pouze hodnota posunu D.

V metodě ABDE-P1 budeme uvažovat hodnoty posunu pouze jako celá čísla. V našem případě tedy hodnota D může nabývat hodnot 0, 1, 2 a 3. Nyní označíme X_{zero} jako obrázek po aplikování metody ZP. Dále stanovíme hodnotu X_{expected}, která při stejné pravděpodobnosti všech možností bude nabývat hodnoty offsetu 1,5. Nakonec jako X_{filtered} označíme obrázek, na který byla aplikována metoda MIG a následně SVF filtrace (viz část 2.6).

Výsledný obraz X_{temp}, který je produktem ABDE-P1 pak vznikne následovně:

$$\begin{aligned}
&\text{I.} \quad X_D = X_{\text{filtered}}(i,j) - X_{\text{expected}}(i,j) \\
&\text{II.} \quad X'_D(i,j) = \begin{cases} 0, & \text{pro} \quad -T > X_D(i,j); \\ 1, & \text{pro} \quad -T \leq X_D(i,j) < 0; \\ 2, & \text{pro} \quad 0 \leq X_D(i,j) < T; \\ 3, & \text{pro} \quad T \leq X_D(i,j). \end{cases} \\
&\text{III.} \quad X_{\text{temp}}(i,j) = X_{\text{zero}}(i,j) + X'_D(i,j),
\end{aligned} \tag{2.13}$$

kde T je stanovený práh (*threshold*) [18].

Aplikováním metody ABDE-P1 jsme dosáhli zvýšení bitové hloubky v celém novém dynamickém rozsahu fotografie. Toho je dosaženo přičtením hodnoty posunu X'_{D} ke všem pixelům (i,j) obrazu X_{zero}. Hodnota posunu X'_{D} byla stanovena prahováním hodnoty X_{D}, která vznikne odečtením obrazu s předpokládaným posunem X_{expected} od obrazu X_{filtered} filtrovaného SVF – viz rovnice (2.13).

Po aplikování ABDE-P1 jsou kontury v X_{temp} redukovány, stále jsou ale viditelné. Proto je potřeba provést další úpravy k ostranění kontur – viz druhá část.

Část 2 – odstranění kontur

Tato část se zabývá odstraněním kontur z obrazu X_{temp}, který byl spočítán v první části. K odstranění kontur je nezbytné tyto kontury v obraze nejprve identifikovat. Pokud bychom tak neučinili, zbytečně bychom celý obraz rozmazali.

Vzniklé kontury se detekují z původního LDR obrazu a jsou to všechny pixely (i,j), které splňují podmínky:

$$\begin{aligned}
&\text{I.} \quad |x(k,l) - x(i,j)| \leq T_1 \ \forall (k,l) \in 3 \times 3 \text{ sousedství} \\
&\text{II.} \quad \text{rozptyl pixelů ze sousedství } 3 \times 3 \leq T_2,
\end{aligned} \tag{2.14}$$

kde T_1 a T_2 jsou opět uživatelem stanovené hranice. Použité, ani navrhované hodnoty T, T_1 ani T_2 článek [18] bohužel neuvádí.

Pokud obě podmínky platí pro pixel $x(i,j)$ a všechny sousední pixely v okolí 5×5, pak je pixel $x(i,j)$ prohlášen za pixel z oblasti kontur. Posledním krokem je

aplikace filtru typu dolní propust o velikosti 7×7 pixelů na všechny pixely splňující podmínky (viz výše). Filtrování může proběhnout i vícekrát. Obecně platí, že čím větší jsou oblasti kontur, tím vícekrát je potřeba dolní propust aplikovat.

Tato metoda podává relativně uspokojivé výsledky a využívá celý rozsah nové bitové hloubky. Má však i své nedostatky. Jedny ze základních bych považoval stále drobné pozůstatky po konturách a v některých případech chybnou detekci kontur. Další možnou nevýhodou metody by se dala považovat nutnost nastavení koeficientů T, T_1 a T_2, se kterými se musí „experimentovat“, než se dojde k cílenému výsledku.

2.8 Contour-Region Reconstruction

Metoda Contour-Region Reconstruction [16] (dále používána zkratka CRR), též nazývána Chengova metoda (podle hlavního autora), byla představena v roce 2009 a do dnešní doby je jednou z nejpoužívanějších metod pro zvýšení bitové hloubky fotografií.

Metoda CRR se skládá z několika kroků. Prvním a zároveň stěžejním bodem této metody je sestavení tzv. distanční mapy, které probíhá následovně. Nejprve se identifikují hrany kontur v obraze. Ty jsou definovány jako rozdíl jednoho bitu LDR obrazu u dvou sousedních pixelů. Pokud je tento rozdíl větší, jak jeden bit, pak pixel pravděpodobně netvoří hranu kontury. Vytvoříme tedy dvě matice *down_map* (představuje hranu z nižší hodnoty na vyšší) a *up_map* (představuje hranu z vyšší hodnoty na nižší), které jsou stejného rozměru, jako obraz. Mapy pak mají hodnotu 1, kde je v obraze hrana kontury a zbytek hodnoty blíké nekonečnu. Tyto hodnoty jsou potom postupně nahrazovány vzdálenostmi od jedničky (viz obrázek 2.1a). Následně se z distančních map vypočítá matice s plovoucí desetinnou čárkou nazvaná *step_ratio* – viz rovnice (2.15), která rovnoměrně rozdělí přechod mezi hranami v rozsahu $\langle 0, 1 \rangle$.

$$step\_ratio(i,j) = \frac{down\_map(i,j)}{down\_map(i,j) + up\_map(i,j)}, \tag{2.15}$$

kde i a j značí pozici daného pixelu v obraze.

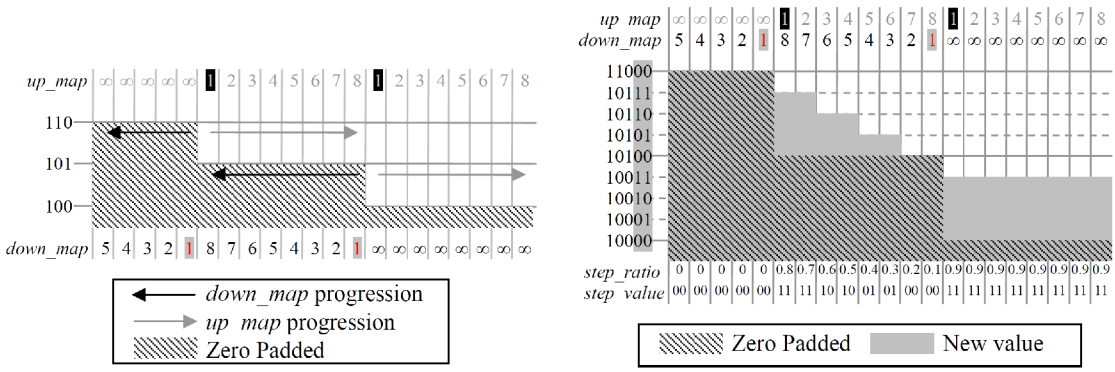
Poté jsou hodnoty *step_ratio* přepočítány na hodnoty v binárním vyjádření, kdy počet bitů je shodný s počtem navýšených bitů v novém obraze – viz rovnice (2.16). Rovnice (2.17) pak představuje výpočet obrazu s vyšší bitovou hloubkou, který vznikne součtem původního obrazu po aplikování metody ZP a matice *step_value*.

$$step\_value(i,j) = step\_ratio(i,j) \times 2^{q-p} \tag{2.16}$$

$$Y(i,j) = \text{ZeroPad}(X(i,j)) + step\_value(i,j) \tag{2.17}$$

Obrázek 2.1b pak v 1D ilustruje algoritmus prakticky celé metody pro přepočítání 3bitového signálu na 5bitový. V horní části můžeme vidět hodnoty v distančních mapách *up_map* a *down_map*. Ve spodní části jsou zobrazeny hodnoty matic *step_ratio* a *step_value*. Graf na obrázku 2.1b pak představuje srovnání původních hodnot s hodnotami po aplikování metody CRR.

Aplikováním tohoto algoritmu jsou všechny hrany kontur transformovány na plynulý gradient. Algoritmus je však aplikován pouze na hrany kontur a díky tomu nezpůsobuje rozmazání celého obrazu a ztrátu detailů v obraze. Celkově jsou výsledky této metody uspokojivé jak z objektivního pohledu (hodnoty PSNR V [16]) tak i ze subjektivního hlediska – např. v porovnání s ABDE podává metoda znatelně lepší výsledky.



(a) Algoritmus tvoření distančních map *up_map* a *down_map*

(b) Srovnání původního obrazu s obrazem s novým po aplikaci CRR algoritmu

Obr. 2.1: Ilustrace v 1D metody Contour-Region Reconstruction. Převzato z [16].

2.9 Contour-Region Dithering

Metoda Contour-Region Dithering (dále jen CRR) [15], neboli dithering oblastí kontur, nesouvisí se zvýšením bitové hloubky obrazu přímo, ale spíše k vylepšení LDR fotografie. Lokální výpočet pro zvýšení bitové hloubky však tato metoda obsahuje, proto jsem ji do své práce zahrnul také. Výsledkem metody je fotografie o stejné bitové hloubce, pouze kontury jsou odstraněny ditheringem, v tomhle případě metoda používá Floyd-Steinbergův algoritmus [22].

Tento algoritmus patří do kategorie metod s distribucí zaokrouhlovací chyby. To znamená, že algoritmus prochází daný obraz po řádcích od shora dolů a chyba, která vznikne při zaokrouhlení se rozdistribuje do ještě nezaokrouhlených míst. V případě

Floyd-Steinbergova algoritmu je distribuce chyby dána maticí

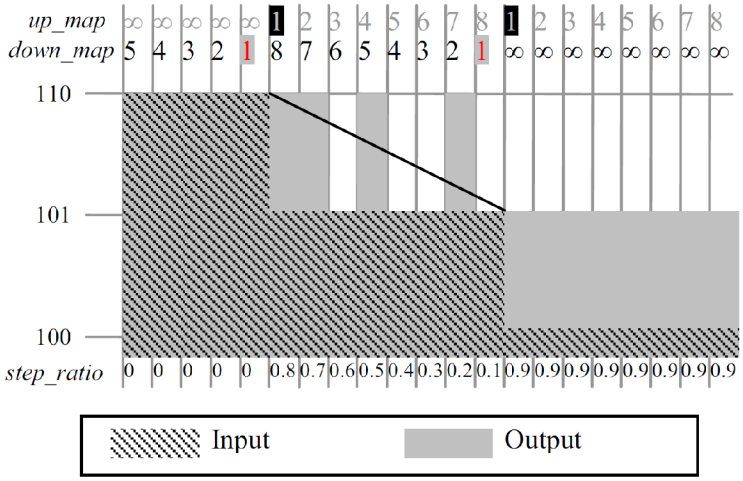
$$\frac{1}{16} \times \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 7 \\ 3 & 5 & 1 \end{bmatrix}. \tag{2.18}$$

Metoda funguje na obdobném principu jako CRR (viz kapitola 2.8). Opět se detekují hrany kontur, sestaví se distanční mapy *down_map* a *up_map*, z nich se pak podle rovnice (2.15) spočítá matice *step_ratio*.

Nově se ale matice *step_ratio* přičte k vstupnímu obrázku podle rovnice (2.19) a vznikne matice vyjádřená v plovoucí desetinné čárce Y' [15].

$$Y'(i,j) = X(i,j) + step\_ratio(i,j) \tag{2.19}$$

Posledním krokem je aplikace Floyd-Steinbergova algoritmu na obraz s plovoucí čárkou. Nejprve je pixel $Y'(i,j)$ zaokrouhlen na nejbližší celou hodnotu $Y(i,j)$ v nižší bitové hloubce. Rozdíl $Y'(i,j) - Y(i,j)$ je pak distribuován podle matice (2.18) do ostatních pixelů. Porovnání vstupu a výstupu v 1D je ilustrováno na obrázku 2.2.



Obr. 2.2: Ilustrace v 1D metody Contour-Region Dithering. Převzato z [15].

Tato metoda „bojuje“ s konturami vzniklými nízkou bitovou hloubkou ditheringem. To způsobuje, že obrázky se při pohledu z dálky zdají v pořádku, ale při detailním pohledu jsou patrné tečky způsobené ditheringem. Pro určitá použití, kde nechceme přímo zvýšit bitovou hloubku fotografie, ale při zachování barevné hloubky alespoň potlačit nepříjemné kontury, by tato metoda mohla být užitečná.

2.10 Content Adaptive Image Bit-Depth Expansion

Chengova metoda (CRR – viz kapitola 2.8) podává velmi dobré výsledky v rekonstrukci oblastí kontur, kde ale selhává jsou oblasti lokálních maxim a minim (LMM – z anglického *local maxima/minima*). Tato metoda představená v červenci 2012 vylepšuje Chengovu metodu právě o detekci LMM oblastí prahováním v matici *step_ratio* a následnou rekonstrukcí těchto oblastí pomocí tvoření tzv. virtuální kostry.

Tento problém je zobrazen na obrázku 2.3. Nejprve můžeme vidět původní 4bitový signál, který je posléze redukován pouze na dva bity. Je zřejmé, že Zero-padding metoda vytvoří několik falešných kontur. Chengova metoda kontury sice odstraní, bohužel ale informace lokálních maxim, popř. minim je ztracená, což právě tato metoda opravuje.

Tato metoda funguje na principu rekonstrukce ztracených LSB bitů 1D interpolací. Nejprve se obdobně jako u Chengovy metody stanoví mapa *step_ratio* (*SR*) algoritmem nazvaným jako *neighborhood flooding*, neboli zaplavováním sousedních pixelů. Na *SR* mapě se poté prahováním detekují LMM oblasti. Pro tyto oblasti jsou následně vyznačeny virtuální kostry, pro konverzi problému extrapolace na interpolaci. Na konec se vypočítají LSB bity pro různé oblasti v *SR* mapě.

1. Neighborhood Flooding

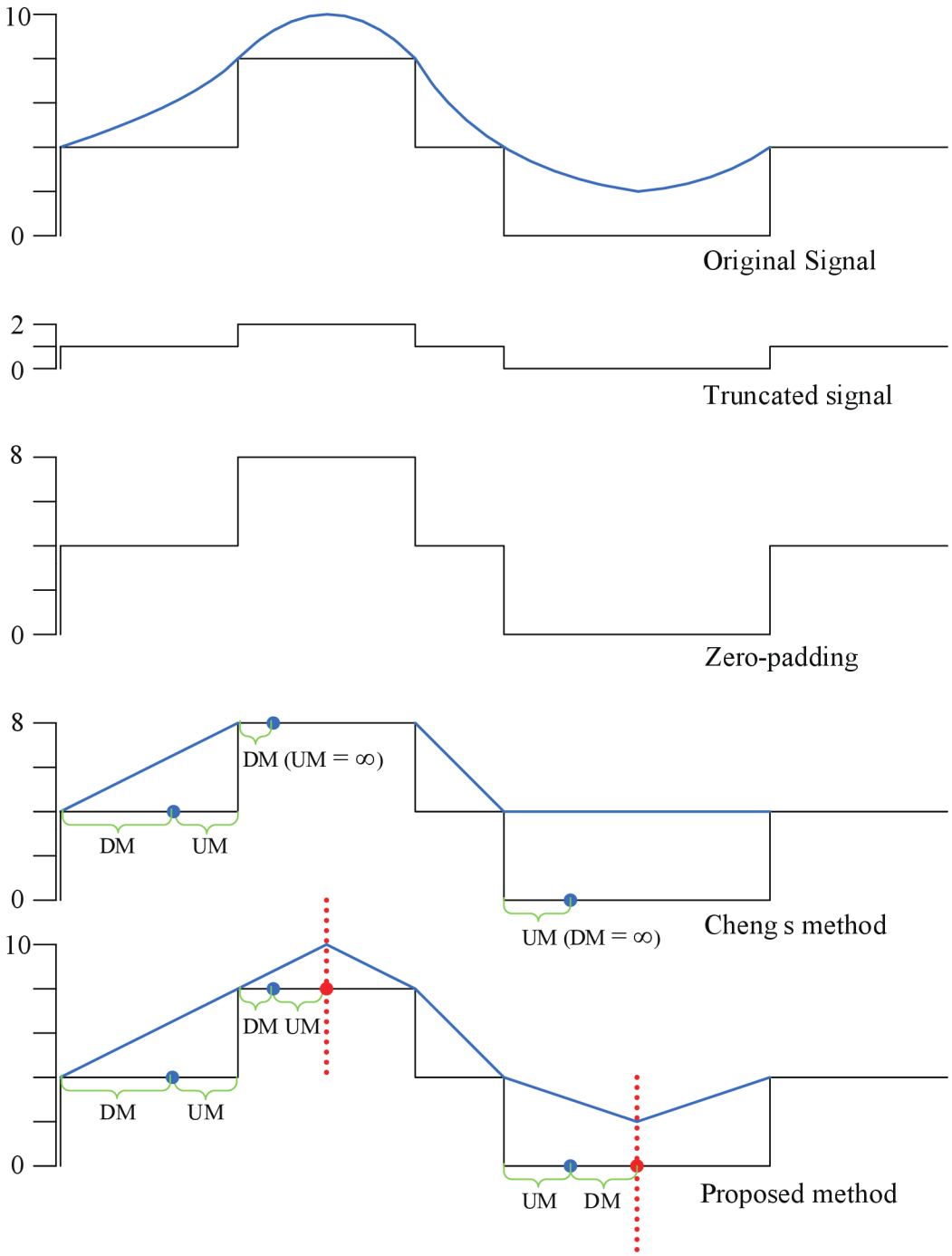
První krok v [33] je spočívá podobně jako u Chengovy metody ve vytvoření distančních map *up_map* (*DM*) a *down_map* (*UM*). Nyní ale používáme k vytvoření těchto map osm sousedních pixelů (oproti Chengově metodě, která používala pouze čtyři).

Hodnoty v mapách si můžeme představit následovně. Pro kruh se středem v pixelu k v obrazu I_l můžeme stanovit malý poloměr $r(r > 0)$. Tento poloměr budeme zvětšovat, dokud nenarazíme na alespoň jeden pixel s intenzitou menší než $I_l(k)$, ale zároveň větší než $I_l(k) - \mathcal{T}_e$, pak poloměr r je hodnota *DM* na pozici k. Hranice $\mathcal{T}_e$ umožňuje odlišení falešných kontur od ostatních hran v obraze. Obdobně se určí i mapa *UM*.

Definujeme ještě absolutní index pixelu pro sousedství j pixelu k jako $\mathcal{N}_k(j)$ a relativní vzdálenost jako $\mathcal{D}(j) \in \{1, \sqrt{2}\}$, kde $j = 0, \ldots, 7$, pro model s osmi sousedními pixely. Model 8 sousedních pixelů zobrazující jejich číslování a relativní vzdálenosti můžeme vidět na obrázku 2.4

Algoritmus zaplavování pak můžeme popsat následovně:

1. Inicializace
   (a) Nastavíme $DM = \infty$, $UM = \infty$ pro všechny pixely



Obr. 2.3: Ilustrace v 1D srovnávající různé metody při rekonstrukci signálu. Převzato z [33].

5	6	7
4	k	0
3	2	1

(a) Model 8 sousedních pixelů

$\sqrt{2}$	1	$\sqrt{2}$
1	k	1
$\sqrt{2}$	1	$\sqrt{2}$

(b) Relativní vzdálenost $\mathcal{D}(j)$ od pixelu k

Obr. 2.4: Model 8 sousedních pixelů a jejich relativních vzdáleností k pixelu k

(b) Pro každý pixel k najdeme hodnoty sousedních pixelů pro $j = 0, \ldots, 7$:

$$\mathcal{S}_k^{dn} = \{j | I_l(k) - I_l(\mathcal{N}_k(j)) \in (0, \mathcal{T}_e)\}$$
$$\mathcal{S}_k^{up} = \{j | I_l(\mathcal{N}_k(j)) - I_l(k) \in (0, \mathcal{T}_e)\}$$
$$\mathcal{S}_k^{eq} = \{j | I_l(k) = I_l(\mathcal{N}_k(j))\}$$

(c) Nastavíme hodnoty do distančních map:

$$DM(k) = \arg\min_{j \in \mathcal{S}_k^{dn}} \mathcal{D}(j)$$
$$UM(k) = \arg\min_{j \in \mathcal{S}_k^{up}} \mathcal{D}(j)$$

Po inicializaci všech pixelů mají mapy DM a UM pouze tři možné hodnoty: $1, \sqrt{2}$ a ∞.

2. Zaplavování

(a) Pro každý pixel k se aktualizují distanční mapy podle následujících pravidel:

$$DM(k) = \min(DM(k), \overline{DM}(k))$$
$$UM(k) = \min(UM(k), \overline{UM}(k)),$$

kde

$$\overline{DM}(k) = \arg\min_{j \in \mathcal{S}_k^{eq}} DM(\mathcal{N}_k(j)) + \mathcal{D}(j)$$
$$\overline{UM}(k) = \arg\min_{j \in \mathcal{S}_k^{eq}} UM(\mathcal{N}_k(j)) + \mathcal{D}(j)$$

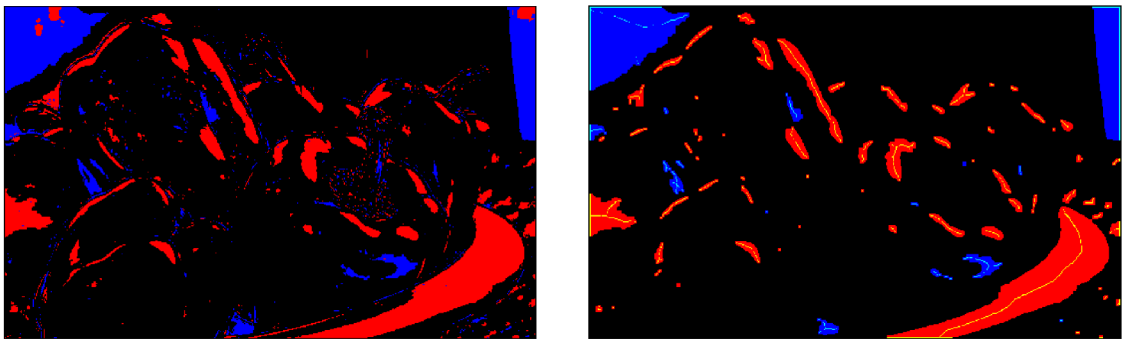
(b) Tento krok opakujeme tak dlouho, dokud se DM a UM neustálí, tzn. nebudou probíhat žádné změny.

V konečné fázi této části metody spočítáme SR stejně jako u předešlé metody – viz rovnice (2.15).

2. Detekce a následná úprava LMM oblastí

Detekce LMM oblastí probíhá relativně jednoduše díky faktu, že pro lokální maxima je $UM = \infty$, tedy $SR = 0$. Naopak pro lokální minima je $DM = \infty$, tedy $SR = 1$.

Nicméně originální LMM mapa může být velmi zašuměná. Proto se na LMM mapu aplikují metody, které odstraní velmi malé LMM oblasti, odstraní regiony nesprávně označené za LMM a vyplní malé díry v LMM regionech způsobené odstraněním velmi malých oblastí [33]. Na obrázcích 2.5 pak můžeme vidět originální LMM mapu a upravenou odšuměnou LMM mapu s vyznačenými virtuálními kostrami (viz níže). Modře jsou vyznačeny oblasti lokálních minim a červeně oblasti lokálních maxim. Azurovou a žlutou jsou pak vyznačeny jejich virtuální kostry.



(a) Původní LMM mapa

(b) Odšuměná LMM mapa s vyznačenými virtuálními kostrami

Obr. 2.5: Porovnání LMM map. Převzato z [33].

3. Vyznačení virtuálních koster v LMM regionech

Předtím, než definujeme algoritmus pro určování virtuálních koster, stanovíme si pro LMM oblasti následující předpoklady:

1. Pixely na virtuálních kostrách mají nejvyšší, resp. nejnižší intenzitu v lokálních maximech, resp. minimech.
2. Pro LMM oblasti, které nejsou na kraji obrázku je jejich virtuální kostra symetricky rozložená podle jejich tvaru (tzv. topologická kostra).
3. Pro LMM oblasti, které jsou na kraji obrázku je jejich virtuální kostra po celé hraně a zároveň obsahují i topologickou kostru.

Tato metoda tedy tvoří virtuální kostry z pixelů, které jsou buď na kraji obrázku nebo splňují podmínku:

$$\sum_{j=0}^{3} f(M(k) - M(\mathcal{N}_k(j)), M(k) - M(\mathcal{N}_k(j+4))) \leq \lambda, \tag{2.20}$$

kde $f(x_1, x_2) = 1$ tehdy, pokud $x_1 > 0$ a $x_2 > 0$. Hodnota λ je opět stanovená hranice. Podle [33] byla nejuspokojivější hodnota $\lambda = 2$ nebo 3.

4. Rekonstrukce LSB závislá na obsahu

Poslední krok této metody spočívá v rekonstrukci LSB jednotlivých pixelů. Tato rekonstrukce je pak závislá na funkci $g(k)$, která závisí na oblasti, ve které je pixel na pozici k přiřazen. Závorky $\lfloor x \rfloor$ značí dolní celou část čísla x.

$$\hat{I}_h(k) = \hat{I}_h^{zp}(k) + \lfloor g(k) \times (2^{q-p} - 1) \rfloor \tag{2.21}$$

$$g(k) = \begin{cases} SR(k) & \text{pro } k \in \mathcal{R}_{\text{nl}} \\ \cos(1 - SR(k))^{\alpha} & \text{pro } k \in \mathcal{R}_{\text{ss}} \\ 0{,}5 & \text{pro } k \in \mathcal{R}_{\text{ad}} \\ 0{,}5 \times SR(k) & \text{pro } k \in \mathcal{R}_{\text{max}} \\ 0{,}5 + 0{,}5 \times SR(k) & \text{pro } k \in \mathcal{R}_{\text{min}} \end{cases} \tag{2.22}$$

Kde

- $\mathcal{R}_{\text{nl}}$ je množina pixelů mimo LMM regiony.
- $\mathcal{R}_{\text{ss}}$ je množina tzv. super-saturovaných oblastí, které jsou definovány, jako množiny takových pixelů k, které vyhovují podmínce $I_l(k) = 2^p - 1$. Pro rekonstrukci takovýchto pixelů používáme funkci kosinus s mocninou α, kterou můžeme upravit výsledek.
- $\mathcal{R}_{\text{ad}}$ je množina pixelů z absolutně tmavých (*absolute-dark*) oblastí. Ty jsou definovány jako všechny pixely k, které splňují $I_l(k) = 0$.
- $\mathcal{R}_{\text{max}}/\mathcal{R}_{\text{min}}$ je množina pixelů patřící do zbývajících oblastí lokálních maxim, respektive minim.

Na závěr je ještě aplikován bilaterální filtr na pixely virtuálních koster, jejichž SR hodnoty jsou chybně reprezentovány jako 0 nebo 1 [33]. Tento filtr patří do kategorie nelineárních hrany-zachovávajících filtrů. Jeho principem je nahrazení hodnoty intenzity pixelu váženým průměrem intenzit okolních pixelů. Váhování může být založeno např. podle Normálního (Gaussova) rozdělení [3].

Ze všech dosud představených metod v této práci právě tato metoda s adaptivním přístupem podává nejlepší výsledky. Vylepšuje totiž kvalitní Chengovu metodu ještě o detekci a úpravu přeexponovaných a podexponovaných oblastí.

Podle [33] při zvýšení bitové hloubky z 6 na 8 bitů dosahuje hodnota SSIM téměř 1. Dokonce i v případě zvýšení bitové hloubky z 2 na 8 bitů je SSIM ukazatel roven hodnotě 0,8. Pro srovnání metoda ZP dosahuje SSIM pouze cca 0,52.

2.11 Zvýšení bitové hloubky procesem inverzní kvantizace

Metoda pro zvýšení bitové hloubky procesem inverzní kvantizace, která byla představena v srpnu 2012, tedy krátce po předchozí metodě CAIBDE, se dívá na problém zvyšování opět z trochu jiného úhlu. V této metodě je představen algoritmus zvýšení bitové hloubky využívající „hrany-zachovávající“ filtr typu dolní propust s adaptivní velikostí okna (v originálním znění *edge-preserving low-pass filter with adaptive window size*) [29].

V této metodě budeme značit LDR obraz s p-bitovou barevnou hloubkou jako $x_2^p(i,j)$, kde dolní index značí číselnou soustavu (v tomto případě binární soustava) a (i,j) značí pozici pixelu. Výslednou bitovou hloubku označíme písmenem q. Nejprve zde popíšu celkový algoritmus a následně pak detailněji popíši proces filtrování.

1. Celkový algoritmus

Tento algoritmus je například oproti předchozí metodě relativně jednoduchý. V prvním kroku metody převedeme binární vyjádření $x_2^p(i,j)$ na dekadické hodnoty $x_{10}^p(i,j)$ a aplikujeme na obraz metodu MIG (viz 2.3). Výsledkem bude obraz $\widetilde{x}_{10}^q(i,j)$. Na tento obraz je pak aplikován filtrovací proces $F[\cdot]$:

$$\hat{x}_{10}(i,j) = F\left[\widetilde{x}_{10}^q(i,j)\right]. \tag{2.23}$$

Výsledek filtrace následně zaokrouhlíme na nejbližší celé číslo:

$$\hat{x}_{10}^q(i,j) = \mathrm{Round}\{\hat{x}_{10}(i,j)\}. \tag{2.24}$$

Konečné dekadické hodnoty pak můžeme zpátky převést na binární. Tím získáme $\hat{x}_2^q(i,j)$ [29].

2. Proces filtrace

V této sekci bude konkrétně vysvětlen proces filtrace $F[\cdot]$. Metoda používá tzv. ε-filtr jehož výstup pro okno $(2n+1)\times(2n+1)$ je:

$$\widetilde{x}_{10}^{(n)}(i,j) = \frac{\sum_{k=-n}^{n}\sum_{l=-n}^{n}\left[f\{\widetilde{x}_{10}^q(i+k,j+l) - \widetilde{x}_{10}^q(i,j)\} + \widetilde{x}_{10}^q(i,j)\right]}{(2n+1)^2}, \tag{2.25}$$

kde $f\{\cdot\}$ je nelineární funkce definována jako:

$$f\{\alpha\} = \begin{cases} \alpha & \text{pro } |\alpha| \le \varepsilon \\ 0 & \text{pro } |\alpha| > \varepsilon, \end{cases} \tag{2.26}$$

kde ε je kladná konstanta. Tento filtr dokáže realizovat filtraci zachovávající hrany, pokud hodnoty jsou z intervalu od $\widetilde{x}^q_{10}(i,j) - \varepsilon$ do $\widetilde{x}^q_{10}(i,j) + \varepsilon$.

Výsledek filtrovacího procesu $F[\cdot]$ je tedy dán:

$$\hat{x}_{10}(i,j) = \begin{cases} \hat{x}^{(n_S)}_{10}(i,j) \text{ pokud } V_m > \mu \\ \hat{x}^{(n_L)}_{10}(i,j) \text{ jinak,} \end{cases} \tag{2.27}$$

kde $n_S < n_L$ a $\hat{x}^{(*)}_{10}(i,j)$ (* zastupuje n_S nebo n_L) je odvozena podle:

$$\hat{x}^{(n)}_{10}(i,j) = \begin{cases} \widetilde{x}^q_{10}(i,j) + \frac{2^q-1}{2(2^p-1)} & \text{pro } \widetilde{x}^{(*)}_{10}(i,j) > \widetilde{x}^q_{10}(i,j) + \frac{2^q-1}{2(2^p-1)} \\ \widetilde{x}^{(n)}_{10}(i,j) & \text{pro } \widetilde{x}^q_{10}(i,j) - \frac{2^q-1}{2(2^p-1)} \leq \widetilde{x}^{(*)}_{10}(i,j) \leq \widetilde{x}^q_{10}(i,j) + \frac{2^q-1}{2(2^p-1)} \\ \widetilde{x}^q_{10}(i,j) - \frac{2^q-1}{2(2^p-1)} & \text{pro } \widetilde{x}^{(*)}_{10}(i,j) < \widetilde{x}^q_{10}(i,j) - \frac{2^q-1}{2(2^p-1)}. \end{cases} \tag{2.28}$$

V_m v rovnici (2.27) znamená průměr hodnot pixelů okna $(2m+1) \times (2m+1)$ se středem v (i,j). Hodnota μ je pak stanovený práh. Pokud je hodnota V_m větší, než μ, znamená to, že se pixely nachází v oblasti hran a měly by být filtrovány menším oknem (n_S). Naopak pokud je tato hodnota větší než průměr V_m, oblast by měla být filtrována větším oknem (n_L) pro lepší odstranění falešných kontur [29].

Tato metoda umožňuje nastavit dokonce 4 parametry a to n_S, n_L, m a μ. V závislosti na těchto parametrech se pak nepatrně liší výsledky metody. Přesné výsledky hodnoty PSNR pro čtyři různé obrázky a hodnoty parametrů jsou uvedeny v [29].

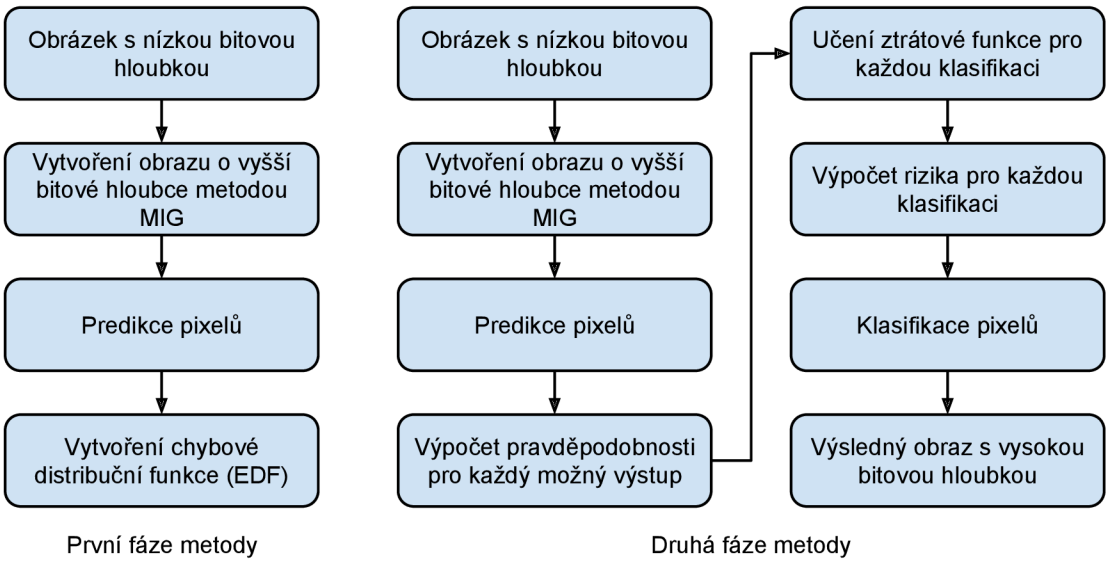
Celkově tato metoda nepodává špatné výsledky. Přímé srovnání však uvádí pouze s prvním krokem metody s adaptivním filtrem (ABDE-P1), kterou sice poráží, ale dle mého názoru na základě srovnání obrázků metody ABDE, Chengova metoda a především metoda Content Adaptive Image podávají kvalitnější výsledky.

2.12 Zvýšení bitové hloubky na základě klasifikace podle minimálního rizika chyby

Další zajímavá metoda (v originále *Bit-Depth Expansion Using Minimum Risk Based Classification*) byla představena v listopadu roku 2012 v článku [19] a k výpočtu bitové hloubky používá především pravděpodobnost výskytu pixelů, které klasifikuje do několika kategorií podle rizika chyby. Tuto metodu přehledně popisuje obrázek 2.6.

První fáze

První fázi můžeme podle obrázku 2.6 rozdělit na 3 kroky. V prvním kroku na obrázek s nízkou bitovou hloubkou aplikujeme metodu MIG (část 2.3).



Obr. 2.6: Přehled metody klasifikace minimálního rizika

V druhém kroku, nazvaném predikce pixelů, předpovíme pravděpodobnou hodnotu každého pixelů ze čtyř okolních pixelů a ze znalosti těchto hodnot vytvoříme chybový obraz (v originále *error image*) E:

$$\begin{aligned} P(i,j) &= [I_{\text{mig}}(i,j-1) + I_{\text{mig}}(i-1,j) + I_{\text{mig}}(i,j+1) + I_{\text{mig}}(i+1,j)]/4 \\ E(i,j) &= I_{\text{mig}}(i,j) - P(i,j), \end{aligned} \tag{2.29}$$

kde $I_{\text{mig}}(i,j)$ je hodnota pixelu na pozici (i,j) obrazu s vyšší bitovou hloubkou vytvořeném metodou MIG a $P(i,j)$ je pak předpokládaná hodnota pixelu.

Ve třetím kroku vytvoříme distribuční funkci chyb (*Error Distribution Function* – EDF). Pokud EDF má v bodě X hodnotu N, pak existuje N pixelů takových, že splňují

$$X = I_{\text{mig}}(i,j) - P(i,j). \tag{2.30}$$

Tato distribuční funkce je pak normalizována vydělením každé hodnoty v histogramu celkovým počtem pixelů [19].

Druhá fáze

V druhé fázi budeme opět postupovat podle obrázku 2.6. Opět nejprve vytvoříme nový obraz pomocí MIG a aplikujeme predikci pixelů pro každý pixel z hodnoty průměru čtyř sousedních pixelů.

Můžeme snadno určit, že celkový počet možností, které může pixel nabývat při zvýšení bitové hloubky je $2^{(H-L)}$, kde H a L je bitová hloubka HBDI, resp. LBDI. Hodnoty, které výstupní pixel může nabývat tedy leží v rozsahu od $I_L(i,j) \times 2^{H-L}$

do $I_L(i,j) \times 2^{H-L} + 2^{(H-L)} - 1$. Pak pro každou hodnotu z tohoto rozsahu můžeme spočítat rozdíl od předpokládané hodnoty $P(i,j)$.

$$\begin{aligned} Diff_1 &= P(i,j) - (I_L \times 2^{(H-L)}) \\ Diff_2 &= P(i,j) - (I_L \times 2^{(H-L)} + 1) \\ &\vdots \\ Diff_{2^{(H-L)}} &= P(i,j) - (I_L \times 2^{(H-L)} + 2^{(H-L)} - 1). \end{aligned} \tag{2.31}$$

Nyní využijeme spočítanou chybovou distribuční funkci EDF, ze které spočteme podmíněné pravděpodobnosti:

$$\begin{aligned} P(O_1) &= EDF(Diff_1) \\ P(O_2) &= EDF(Diff_2) \\ &\vdots \\ P(O_{2^{(H-L)}}) &= EDF(Diff_{2^{(H-L)}}). \end{aligned} \tag{2.32}$$

Po nalezení všech pravděpodobností $[P(O_1); P(O_{2^{(H-L)}})]$ musíme tyto pravděpodobnosti normalizovat. Normalizace hodnot je dána podle:

$$P(O_j) = \frac{P(O_j)}{\sum_{k=1}^{2^{(H-L)}} P(O_k)}. \tag{2.33}$$

Dále definujeme tzv. ztrátovou funkci. Každý pixel původní hodnoty od $I_L(i,j) \times 2^{H-L}$ do $I_L(i,j) \times 2^{H-L} + 2^{(H-L)} - 1$ může být klasifikován do jakékoli jiné kategorie. Ztrátová funkce pak tedy bude mít $2^{2\times(H-L)}$ hodnot. Hodnota λ_{ij} pak udává kvadratickou chybu pixelu z kategorie i přiřazeného do kategorie j.

$$\lambda_{ij} = (i-j)^2 \tag{2.34}$$

Pokud tedy bude hodnota přiřazena do původní kategorie ($i = j$), ztrátová hodnota je nulová. V opačném případě se ztráta rovná kvadrátu rozdílu.

Po získání podmíněných pravděpodobností všech pixelů a ztrátových funkcí určíme hladinu rizika zařazení pixelu do každé kategorie. Hodnota rizika spojená se zařazením pixelu z j-té kategorie je dána jako:

$$R(O_j) = \sum_{i=1}^{2^{(H-L)}} P(O_i)\lambda_{ji}. \tag{2.35}$$

Nakonec po nalezení hodnot rizik pro každou možnou výstupní kategorii obrazu s vysokou bitovou hloubkou, přiřadíme každý pixel do kategorie, kde bude mít nejnižší hodnotu rizika [19].

Tato metoda přináší dobré výsledky výsledných obrazů bez nutnosti následné filtrace, což je velká výhoda. Bohužel se v [19] nachází pouze jediný obrázek, na který

byla aplikována metoda, což neumožňuje řádné srovnání výsledků. Článek udává pouze hodnoty PSNR a výsledky metody porovnává podobně jako předchozí metoda pouze s metodami ZP, MIG, BR a ABDE-P1. Tato metoda podává z uvedených metod nejlepší hodnoty PSNR. Následně článek [19] dodává, že výsledky metody můžeme ještě dále upravit algoritmem pro odstranění kontur, jaký byl představen v druhé části metody ABDE.

3 ŘÍDKÉ REPREZENTACE

Jak už bylo zmíněno v kapitolách výše, jakýkoli digitální obraz můžeme vnímat jako diskrétní 2D signál. Bitová hloubka tohoto obrazu není nic jiného, než počet hladin, na které je tento signál nakvantován. Zvýšení bitové hloubky si tedy můžeme představit jako dekvantizaci signálu.

Tuto úlohu můžeme implementovat několika způsoby. Několik základních metod řešících tuto úlohu je představeno v kapitole 2. Jelikož ale víme, že obrazový signál je v určitých bázích řídký (například DCT nebo DWT), jeví se jako velmi výhodné použít dekvantizaci právě na základě řídké reprezentace signálu.

V této kapitole budou nejprve uvedeny základní pojmy a vztahy pro pochopení problematiky řídké reprezentace. Dále budou zmíněny podmínky nutné k nalezení řídkého řešení nedourčeného systému rovnic a ke konci kapitoly budou uvedeny i některé algoritmy pro nalezení řídkého řešení.

3.1 Základní pojmy a značení

V následujících částech práce budeme skalární veličiny značit kurzívou, např. *m, N*. Vektory budou značeny tučně malými písmeny, např. **x, y**, matice pak tučně velkými písmeny, tedy jako **A, B**. Vektory budeme uvažovat jako sloupcové s konečným počtem prvků, indexované od jedničky. Tedy první prvek vektoru $\mathbf{x}$ bude značen jako x_1, n-tý prvek jako x_n, atd. Kardinalita množiny, neboli počet prvků množiny, bude značena obdobně jako absolutní hodnota, tj. např. $|\{-5, 2, 7, 1, -8\}| = 5$.

Nejprve zde definujeme pojem nosič vektoru. Tímto označením máme na mysli množinu indexů, v nichž má vektor nenulové hodnoty. Tuto množinu pak značíme $\mathrm{supp}(\mathbf{x})$. Obecně tedy $\mathrm{supp}(\mathbf{x}) = \{i \,|\, x_i \neq 0\}$ [13].

Mějme tedy řídký signál $\mathbf{x} = [0, 0, 3, 0, 0, 4, 5, 0, 0, 8, 0, 1]^\top$. Podle výše uvedených definic bude platit $\mathrm{supp}(\mathbf{x}) = \{3,4,5,8,1\}$ a $|\mathrm{supp}(\mathbf{x})| = 5$.

Nyní si ještě definujeme ℓ_p-normu vektoru. Ta je pro libovolný vektor $\mathbf{x} \in \mathbb{C}^N$ definována jako

$$\begin{aligned} \|\mathbf{x}\|_p &:= \left(\sum_{i=1}^{N} \|x_i\|^p \right)^{1/p} \quad \text{pro} \quad 1 \leq p < \infty, \\ \|\mathbf{x}\|_p &:= \sum_{i=1}^{N} |x_i|^p \quad \text{pro} \quad 0 < p < 1, \\ \|\mathbf{x}\|_\infty &:= \max_i |x_i|, \\ \|\mathbf{x}\|_0 &:= |\mathrm{supp}(\mathbf{x})|. \end{aligned} \tag{3.1}$$

Z matematického hlediska se o normu jedná pouze v případě $1 \leq p < \infty$. Pro zjednodušení však pro pro všechna p budeme používat označení ℓ_p-norma [13].

Speciálními případy ℓ_p-norem jsou ℓ_1-norma, tedy $\|\cdot\|_1$, která představuje součet absolutních hodnot prvků vektoru a norma $\|\cdot\|_0$ reprezentující počet nenulových složek vektoru. Často používána je také norma $\|\cdot\|_\infty$, která vrací složku vektoru s největší hodnotou [13].

Pokud se zde zabýváme řídkými reprezentacemi signálu, měli bychom si také definovat pojem řídkost. Vektor $\mathbf{x}$ nazveme k-řídkým, pokud platí

$$\|\mathbf{x}\|_0 \leq k. \tag{3.2}$$

Jinými slovy k-řídký vektor je takový vektor, který má nejvýše k složek nenulových. Relativní řídkostí potom rozumíme počet nenulových složek vektoru k ku délce vektoru N, tedy $\frac{k}{N}$ [13].

3.2 Aditivní (syntezující) model signálu

Pro další pochopení je důležité si uvést jednu důležitou věc. Jakýkoli signál (ať už spojitý nebo diskrétní) můžeme vyjádřit pomocí součinu matice a vektoru souřadnic. Tento proces lze také nazvat transformací. Použitou matici pak většinou označujeme jako transformační matici. Např. v případě Fourierovy transformace můžeme signál vyjádřit pomocí součtu sinusovek a kosinusovek. V tomto případě by transformační matice obsahovala jednotlivé sinusovky a kosinusovky a vektor souřadnic pak jednotlivé koeficienty transformace.

Mějme tedy libovolný signál $\mathbf{y}$. Ten tedy můžeme vyjádřit pomocí součinu transformační matice, kterou označíme např. $\mathbf{A}$ a vektoru souřadnic $\mathbf{x}$. Takovýto model signálu nazveme aditivní, popř. synzující, protože celý signál skládáme z více částí. Platí tedy, že $\mathbf{y} = \mathbf{A}\mathbf{x}$.

3.3 Hledání řídkého řešení

Při hledání řídkého řešení vlastně hledáme řešení soustavy lineárních rovnic $\mathbf{A}\mathbf{x} = \mathbf{y}$ s tím rozdílem, že možných řešení této soustavy je nekonečně mnoho a my hledáme právě to nejřidší, tedy hledáme takové řešení, které má co největší počet nulových složek. Jedná se o následující úlohu:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{vzhledem k} \quad \mathbf{A}\mathbf{x} = \mathbf{y} \tag{3.3}$$

kde známe vektor $\mathbf{y} \in \mathbb{C}^m$ a matici $\mathbf{A} \in \mathbb{C}^{m \times N}$. Budeme uvažovat pouze případy, kdy $m < N$ popř. $m \ll N$ a $\mathbf{A}$ je plné řádkové hodnosti. Matici $\mathbf{A}$ obvykle označujeme jako slovník (*dictionary*) a její sloupce jako atomy (*atoms*) [13].

V praxi se často můžeme setkat s tím, že požadovaný signál bude zašuměn a právě tento šum způsobí odchylku od přesného řešení. Proto se při výpočtu $\mathbf{Ax} = \mathbf{y}$ povolí malá odchylka:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{vzhledem k} \quad \|\mathbf{Ax} - \mathbf{y}\|_p \leq \delta, \tag{3.4}$$

kde většinou uvažujeme $p = 2$ [13].

Nalezení řídkého řešení není jednoznačná záležitost. Slovník $\mathbf{A}$ musí splňovat určité podmínky, aby bylo možno zjistit, zda vůbec řídké řešení existuje, popř. jestli nalezené řešení je nejřidší možné a jedinečné.

Důležitou vlastností pro rozhodování o existenci a vlastnostech řešení je číslo *spark*, udávající nejmenší počet sloupců matice, které jsou lineárně závislé. Formálně tedy můžeme zapsat [13]:

$$\text{spark}(\mathbf{A}) = \min_{\mathbf{z} \in \ker \mathbf{A}, \mathbf{z} \neq 0} \|\mathbf{z}\|_0 \tag{3.5}$$

Z definice lze usoudit, že pro nenulovou matici $\mathbf{A} \in \mathbb{C}^{m \times N}$, kde $m < N$, platí $\text{spark}(\mathbf{A}) \in \{2, \ldots, m+1\}$. Pro řešení našeho problému $\mathbf{Ax} = \mathbf{y}$ můžeme číslo spark využít takto: pokud by nalezené řešení $\mathbf{x}$ splňovalo podmínku

$$\|\mathbf{x}\|_0 < \frac{\text{spark}(\mathbf{A})}{2}, \tag{3.6}$$

pak toto řešení je nutně nejřidší možné a žádné jiné řešení se stejnou kardinalitou neexistuje [13].

Zjišťování vlastností řešení pomocí čísla spark má však obrovskou nevýhodu v tom, že nalezení $\text{spark}(\mathbf{A})$ je srovnatelně výpočetně náročné, jako samotné řešení problému (3.3). Proto se v praxi nepoužívá.

Zavedeme si tedy další vlastnost matice $\mathbf{A}$, a tou je vzájemná koherence (*mutual coherence*). Je definována jako největší absolutní normalizovaný skalární součin dvou různých sloupců matice $\mathbf{A}$,

$$\mu(\mathbf{A}) = \max_{1 \leq j,k \leq N, j \neq k} \frac{|\mathbf{a}_j^\top \mathbf{a}_k|}{\|\mathbf{a}_j\|_2 \cdot \|\mathbf{a}_k\|_2}, \tag{3.7}$$

kde $\mathbf{a}_j$ označuje j-tý sloupec matice $\mathbf{A}$. Pro libovolnou matici $\mathbf{A}$ platí vztah mezi číslem spark a vzájemnou koherencí matice následující:

$$\text{spark}(\mathbf{A}) \leq 1 + \frac{1}{\mu(\mathbf{A})}. \tag{3.8}$$

Pokud tedy pro soustavu rovnic $\mathbf{Ax} = \mathbf{y}$ najdeme řešení $\mathbf{x}$ splňující

$$\|\mathbf{x}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{A})}\right), \tag{3.9}$$

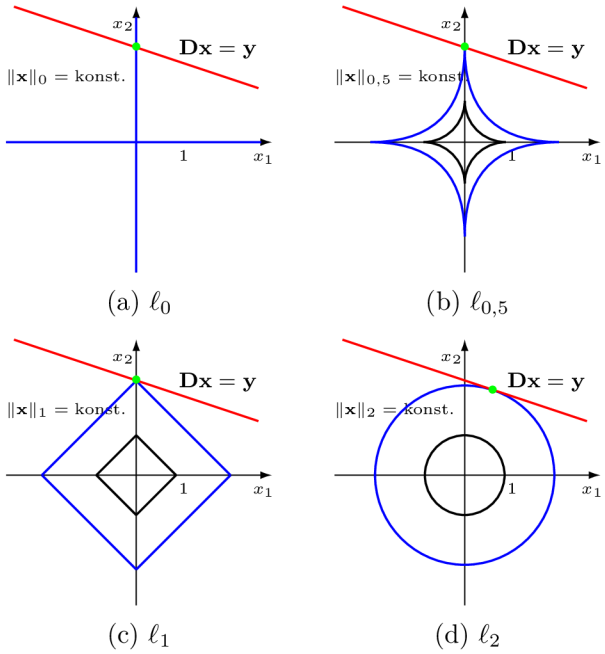
pak $\mathbf{x}$ je nutně nejřidší možné a je jediné takové. Navíc lze takovéto řešení nalézt ℓ_1-minimalizací (viz níže). Z toho vyplývá, že pro co nejřidší řešení potřebujeme použít maximálně nekoherentní slovníky [13].

Objevuje se zde ale jiný problém. Pokud bychom totiž trvali na nalezení přesného řešení (tedy takové $\mathbf{x}$ s minimální ℓ_0-normou), museli bychom, při existenci k_0-řídkého řešení dané soustavy, projít všech $\binom{N}{k_0}$ kombinací podmnožin sloupců matice. Takovéto řešení odpovídá NP-složitosti (*NP-hard*), což v praxi, kdy N nabývá vysokých hodnot, není přijatelné. Nezbývá tedy než použít aproximační metodu a nekonvexní ℓ_0-normu nahradit ℓ_1-normou, která už konvexní je. Tuto aproximační úlohu tedy můžeme zapsat jako

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{vzhledem k} \quad \mathbf{Ax} = \mathbf{y}. \tag{3.10}$$

Cenou za rychlejší výpočet je tedy mírná nepřesnost [13].

Nalezení řešení úlohy (3.10) nám nezaručí, že nalezené řešení bude nejřidší. Lze ale dokázat, že ve většině případů se řešení ℓ_1-relaxace shoduje s řešením ℓ_0-minimalizace. V některých případech se ale může stát, že nalezneme pomocí ℓ_1- minimalizace prakticky nekonečně mnoho řešení, avšak ty „nejřidší“ budou pouze ta krajní. Pro ilustraci tohoto případu slouží obrázek 3.1 , který zobrazuje „nafukující se“ koule v normách (a) ℓ_0,(b) $\ell_{0,5}$, (c) ℓ_1 a (d) ℓ_2 a jejich dotyk s nadrovinou určenou soustavou $\mathbf{Ax} = \mathbf{y}$. Z obrázků je patrné, že pokud by nadrovina procházela právě stranou jednotkové koule ℓ_1, pak by řešení ℓ_1-relaxace bylo nekonečně mnoho, nejřidší by však byla pouze dvě krajní řešení (na obrázku průsečíky s osami x_1 a x_2).



Obr. 3.1: Ilustrace jednotkových koulí

Existuje několik podmínek, které právě zajišťují ekvivalenci řešení ℓ_0- a ℓ_1- minimalizace. V praxi se však příliš často nepoužívají a proto zde nebudu zacházet do přílišných detailů a konkrétních definic, ale pouze zjednodušeně naznačím, na jakém principu tyto podmínky fungují.

První podmínkou je tzv. vlastnost nulového prostoru (*NSP – Null Space Property*). Splnění podmínky NSP pro libovolný vektor z jádra $\ker(\mathbf{A})$ zajišťuje, že v něm bude norma koncentrována v „malém počtu“ prvků. NSP tedy obstará, že k-řídké řešení je jednoznačné a lze jej nalézt ℓ_1-minimalizací. Zároveň platí i opačně. Tedy pokud je možné ze soustavy $\mathbf{Ax} = \mathbf{y}$ rekonstruovat všechny k-řídké vektory $\mathbf{x}$ pomocí ℓ_1-minimalizace, pak matice $\mathbf{A}$ splňuje podmínku NSP [13].

Druhou podmínkou, kterou zde uvedu, je tzv. vlastnost zeslabené izometrie (*RIP* – Restricted Isometry Property), která nabízí výpočetně přijatelnější řešení, než výše uvedená NSP, navíc je stabilní i pod vlivem šumu. Nevýhodou je, že se dosud nikomu nepodařilo sestavit deterministickou matici, která by splňovala RIP s předem definovanými parametry. S vysokou pravděpodobností ale RIP splňují matice, které vznikly za přispění náhody, např. náhodným vybráním m řádků. Zde se v praxi používají především Gaussovské matice (její prvky jsou nezávisle generovány z normálního rozdělení) a Bernoulliovské matice (každý prvek je náhodnou veličinou nabývající $\pm 1/\sqrt{m}$ se stejnou pravděpodobností) [13]. Tyto matice se s výhodou používají při tzv. komprimovaném snímání [12].

4 TEORETICKÉ ŘEŠENÍ

4.1 Matematická formulace úlohy dekvantizace

Na základě znalosti faktu, že obrazové signály jsou v určitých reprezentacích řídké, jsme se rozhodli pro dekvantizační úlohu využít postup založený na řídké reprezentaci signálů. Nejdříve bychom si měli matematicky formulovat náš problém. Pro jednoduchost budeme zatím uvažovat jedno-dimenzionální obecný signál. Jak uvidíme později, přechod k 2D obrazovému signálu bude už relativně snadný.

Mějme vektor hodnot $\mathbf{y}$, který reprezentuje původní signál. Tento signál nakvantujeme. Protože se zabýváme obrazovým signálem, budeme předpokládat, že signál je kvantován s konstantním krokem a rozhodovací úroveň kvantizace bude umístěna přesně v půli mezi kvantizačními hladinami. Velikost kvantizačního kroku, tedy vzdálenost mezi dvěma kvantizačními hladinami, si označme jako α. Nakvantovaný signál označme například $\mathbf{y}_\mathrm{q}$. Předpokládáme, že náš signál je řídký v určité bázi $\mathbf{A}$. Z poznatků o kvantizaci víme, že maximální odchylka kvantovaného signálu od původního je $\alpha/2$. Proto hranice pro dekvantizaci nastavíme $\alpha/2$ od kvantovaného signálu. Nemůže tedy nastat situace, že bychom odhadli hodnotu signálu spadající do jiné kvantovací hladiny, než původní signál $\mathbf{y}$. Matematicky můžeme úlohu formulovat jako:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{vzhledem k} \quad \|\mathbf{y}_\mathrm{q} - \mathbf{A}\mathbf{x}\|_\infty \leq \alpha/2. \tag{4.1}$$

Za povšimnutí stojí, že v tomto případě je použita norma $\|\cdot\|_\infty$, která vrací prvek s největší velikostí. Můžeme s jistotou tvrdit, že pokud maximální rozdíl nepřekročí hodnotu $\alpha/2$, pak logicky bude takto nerovnost platit pro všechny ostatní prvky.

Problém (4.1) však můžeme transformovat na jiný typ úlohy, který náš problém vyřeší efektivněji a rychleji. Jednou z možností je přeformulovat jej do úlohy lineárního programování [20]. Další možností je použití algoritmu z rodiny metod tzv. proximálního dělení (*proximal splitting*) [8]. Oba přístupy jsou uvedeny níže v částech 4.3 a 4.4.

4.2 Rozbor omezující podmínky

Pro další účely řešení dekvantizační úlohy by bylo vhodné přesněji specifikovat omezující podmínku $\|\mathbf{y}_\mathrm{q} - \mathbf{A}\mathbf{x}\|_\infty \leq \alpha/2$ a definovat C jako množinu všech $\mathbf{x}$, která podmínce vyhovují, tedy $C = \{\mathbf{x} \mid \|\mathbf{y}_\mathrm{q} - \mathbf{A}\mathbf{x}\|_\infty \leq \alpha/2\}$. Podmínku lze také zapsat ve formě, že hledáme takové $\mathbf{x}$, kde platí:

$$\forall i : -\alpha/2 \leq y_{\mathrm{q}i} - (Ax)_i \leq \alpha/2. \tag{4.2}$$

V případě, že nerovnosti budeme brát po složkách pak můžeme vektorově zapsat, že

$$-\frac{\alpha}{2}\mathbf{1} \leq \mathbf{y}_{\mathrm{q}} - \mathbf{A}\mathbf{x} \leq \frac{\alpha}{2}\mathbf{1}. \tag{4.3}$$

Jinými slovy hledáme takové $\mathbf{x}$, kde každý jeho prvek splňuje, že absolutní hodnota rozdílu kvantovaného signálu a rekonstruovaného signálu nebude větší jak polovina kvantovacího kroku.

Pro programové zpracování je však nutné zapsat tyto dvě nerovnosti do jedné. Tento krok lze učinit následovně: napíšeme si zvlášť levou a pravou nerovnost, po drobné matematické úpravě dostáváme

$$\begin{aligned} \mathbf{A}\mathbf{x} &\leq \frac{\alpha}{2}\mathbf{1} + \mathbf{y}_{\mathrm{q}}, \\ -\mathbf{A}\mathbf{x} &\leq \frac{\alpha}{2}\mathbf{1} - \mathbf{y}_{\mathrm{q}}. \end{aligned} \tag{4.4}$$

Z těchto dvou nerovností pak můžeme jednoduše pomocí maticového zápisu vytvořit jedinou nerovnost, která bude zabezpečovat ohraničení požadovaných hodnot.

$$\begin{bmatrix} \mathbf{A} \\ -\mathbf{A} \end{bmatrix} \cdot \mathbf{x} \leq \begin{bmatrix} \alpha/2 + \mathbf{y}_{\mathrm{q}} \\ \alpha/2 - \mathbf{y}_{\mathrm{q}} \end{bmatrix} \tag{4.5}$$

4.3 Řešení úlohy pomocí lineárního programování

4.3.1 Lineární programování

Lineární programování je odvětví optimalizace, které řeší nalezení minima (popř. maxima) lineární funkce N proměnných na tzv. přípustné množině, popsané soustavou lineárních nerovností [17]. Obecná úloha lineárního programování má tvar:

$$\min_{\mathbf{x}} \mathbf{c}^{\top}\mathbf{x} \quad \text{vzhledem k} \quad \begin{cases} \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \geq 0, \end{cases} \tag{4.6}$$

kde matice $\mathbf{A}$ je rozměru $m \times N$, $\mathbf{x}$ reprezentuje N-rozměrný vektor neznámých. Dále m-rozměrný vektor $\mathbf{b}$ a N-rozměrný vektor $\mathbf{c}$ jsou vektory známých koeficientů [20].

4.3.2 Dekvantizace pomocí lineárního programování

Při použití v dekvantizační úloze budeme vycházet z tvaru pro lineární programování uvedeného v (4.6). Protože z úlohy (4.1) chceme minimalizovat ℓ_1-normu vektoru $\mathbf{x}$, tedy součet absolutních hodnot jednotlivých prvků vektoru, vektor $\mathbf{c}$ můžeme nahradit jednotkovým vektorem.

Po začlenění omezovací podmínky $\|\mathbf{y}_\mathrm{q} - \mathbf{A}\mathbf{x}\|_\infty \leq \alpha/2$ pak řešíme následující úlohu:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{vzhledem k} \quad \begin{bmatrix} \mathbf{A} \\ -\mathbf{A} \end{bmatrix} \cdot \mathbf{x} \leq \begin{bmatrix} \alpha/2 + \mathbf{y}_\mathrm{q} \\ \alpha/2 - \mathbf{y}_\mathrm{q} \end{bmatrix} \tag{4.7}$$

Pro použití lineárního programování nás však stále omezuje podmínka $\mathbf{x} \geq 0$, protože obecně vektor $\mathbf{x}$ může nabývat jakýchkoli hodnot z oboru reálných čísel. Tento problém ale můžeme elegantně obejít následujícím způsobem. Jakýkoli vektor můžeme totiž nahradit rozdílem dvou nezáporných vektorů. Proto vektor $\mathbf{x}$ nahradíme dvěma nezápornými vektory $\mathbf{u}$ a $\mathbf{v}$, pro které platí, že $\mathbf{x} = \mathbf{u} - \mathbf{v}$. Celou úlohu pak můžeme zapsat jako [20]

$$\min_{\mathbf{u},\mathbf{v}} \mathbf{1}^\top \mathbf{u} + \mathbf{1}^\top \mathbf{v} \quad \text{vzhledem k} \quad \begin{cases} \mathbf{u} \geq 0, \;\; \mathbf{v} \geq 0 \\ -\dfrac{\alpha}{2}\mathbf{1} \leq (\mathbf{y}_\mathrm{q} - \mathbf{A}\mathbf{u} + \mathbf{A}\mathbf{v}) \leq \dfrac{\alpha}{2}\mathbf{1}. \end{cases} \tag{4.8}$$

Na konci optimalizace pak bude platit, že $\forall i : u_i \cdot v_i = 0$. Tímto postupem můžeme pomocí lineárního programování zjistit vektory $\mathbf{u}$ a $\mathbf{v}$, které jsou nezáporné a jejich jednoduchým odečtením získáme požadovaný vektor souřadnic $\mathbf{x}$.

4.4 Řešení úlohy pomocí proximálního dělení

Další možností, jak vyřešit minimalizační problém dekvantizační úlohy, je použít algoritmus z rodiny algoritmů proximálního dělení, konkrétně Douglas-Rachfordův algoritmus. Protože tento algoritmus budeme brát pouze jako prostředek pro vyřešení minimalizačního problému při dekvantizaci, nebudeme zde zacházet do přílišných detailů.

4.4.1 Proximální operátory

V algoritmech proximálního dělení se můžeme často setkat s tzv. proximálními operátory. Proximální operátor funkce f je definován pro každé $\mathbf{x} \in \mathbb{R}^N$ jako řešení minimalizačního problému [8]:

$$\min_{\mathbf{y} \in \mathbb{R}^N} f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2. \tag{4.9}$$

Proximální operátory nám umožňují pracovat i s nediferencovatelnými funkcemi, díky čemuž můžeme právě tyto algoritmy použít.

Jak vyplyne z části 4.4.3, budou pro nás důležité především proximální operátory pro ℓ_1-normu a pro indikátorovou funkci. Proximálním operátorem pro $\|\mathbf{x}\|_1$ je tzv.

měkké prahování (*soft tresholding*), které lze definovat následovně [8]:

$$\mathrm{soft}_{\langle \underline{\omega}, \overline{\omega} \rangle}(\mathbf{x}) = \begin{cases} \mathbf{x} - \underline{\omega} & \text{pokud } \mathbf{x} < \underline{\omega}, \\ 0 & \text{pokud } \mathbf{x} \in \langle \underline{\omega}, \overline{\omega} \rangle, \\ \mathbf{x} - \overline{\omega} & \text{pokud } \mathbf{x} > \overline{\omega}. \end{cases} \tag{4.10}$$

Indikátorovou funkci ι_C lze definovat jako

$$\iota_C : \mathbf{x} \mapsto \begin{cases} 0 & \text{pokud } \mathbf{x} \in C, \\ +\infty & \text{pokud } \mathbf{x} \notin C, \end{cases} \tag{4.11}$$

kde množina $C \subset \mathbb{R}^N$ [8]. Jinými slovy indikátorová funkce je taková funkce, která vrátí hodnotu 0, pokud prvek $\mathbf{x}$ náleží do množiny C, a hodnotu $+\infty$ pokud nikoliv. Proximálním operátorem indikátorové funkce ι_C je pak projekce na množinu C, což budeme zapisovat $\mathrm{P}_C\mathbf{x}$. Tuto projekci si můžeme představit jako funkci, která ponechá $\mathbf{x}$ spadající do množiny C a posune $\mathbf{x} \notin C$, na jemu nejbližší bod z C.

4.4.2 Algoritmy proximálního dělení

Algoritmy z rodiny proximálního dělení jsou založeny na myšlence, že se mnoho optimalizačních problémů dá zapsat v tzv. neomezeném tvaru (*unconstrained form*), kdy minimalizujeme součet m konvexních funkcí. V základním tvaru tedy neomezený tvar vypadá následovně [8]:

$$\min_{\mathbf{x} \in \mathbb{R}^N} f_1(\mathbf{x}) + \cdots + f_m(\mathbf{x}), \tag{4.12}$$

kde $f_1, \ldots, f_m$ jsou konvexní funkce.

Douglas-Rachfordův algoritmus je obdobný jako dopředně-zpětný algoritmus, který řešil úlohu [8]:

$$\min_{x \in \mathbb{R}^N} f_1(\mathbf{x}) + f_2(\mathbf{x}), \tag{4.13}$$

kdy obě funkce f_1 a f_2 konvexní. Navíc platí, že jedna z těchto funkcí je diferencovatelná pomocí tzv. Lipschitzova spojitého gradientu ∇f_2. Pokud funkce splňují výše uvedené podmínky, pak existuje alespoň jedno přípustné řešení, které pro $\gamma \in (0, +\infty)$ můžeme charakterizovat rovnicí [8]

$$\mathbf{x} = \mathrm{prox}_{\gamma f_1}(\mathbf{x} - \gamma \nabla f_2(\mathbf{x})). \tag{4.14}$$

Z této rovnice je patrné, že pokud budeme výše uvedený postup opakovat, budeme postupně konvergovat k přesnému řešení. Tento iterativní postup můžeme zapsat jako [8]

$$\mathbf{x}_{n+1} = \underbrace{\mathrm{prox}_{\gamma_n f_1}}_{\text{zpětný krok}} \underbrace{(\mathbf{x}_n - \gamma_n \nabla f_2(\mathbf{x}_n))}_{\text{dopředný krok}}. \tag{4.15}$$

Celý dopředně-zpětný algoritmus pak můžeme shrnout následovně [8]:

$$\begin{aligned}
&\text{Zvolíme } \epsilon \in \left(0, \min\left\{1, \tfrac{1}{\beta}\right\}\right), \mathbf{x}_0 \in \mathbb{R}^N \\
&n = 0, 1, 2, \dots \\
&\gamma_n \in \langle \epsilon, \tfrac{2}{\beta} - \epsilon \rangle \\
&\mathbf{y}_n = x_n - \gamma_n \nabla f_2(\mathbf{x}_n) \\
&\lambda_n \in \langle \epsilon, 1 \rangle \\
&\mathbf{x}_{n+1} = \mathbf{x}_n + \lambda_n \left(\text{prox}_{\gamma_n f_1} \mathbf{y}_n - \mathbf{x}_n\right).
\end{aligned} \tag{4.16}$$

Tento algoritmus se ale pro naše použití nehodí, především z důvodu své omezenosti, kdy předpokládá jednu funkci diferencovatelnou podle Lipschitzova spojitého gradientu. V dekvantizační úloze ale takovouto funkci nepředpokládáme. Můžeme ale podmínku diferencovatelnosti nahradit opět pomocí proximálního operátoru. Takovýto algoritmus je nazýván Douglas-Rachfordův a jeho řešení je pro $\gamma \in (0, +\infty)$ charakterizováno pomocí následujících dvou podmínek [8]:

$$\begin{aligned}
&\mathbf{x} = \text{prox}_{\gamma f_2} \mathbf{y} \\
&\text{prox}_{\gamma f_2} \mathbf{y} = \text{prox}_{\gamma f_1}(2\text{prox}_{\gamma f_2} \mathbf{y} - \mathbf{y}).
\end{aligned} \tag{4.17}$$

Vlastní Douglas-Rachfordův algoritmus lze pak obecně zapsat takto [8]:

$$\begin{aligned}
&\text{Zvolíme } \epsilon \in (0, 1), \gamma > 0, \mathbf{y}_0 \in \mathbb{R}^N \\
&n = 0, 1, 2, \dots \\
&\mathbf{x}_n = \text{prox}_{\gamma f_2} \mathbf{y}_n \\
&\lambda_n \in \langle \epsilon, 2 - \epsilon \rangle \\
&\mathbf{y}_{n+1} = \mathbf{y}_n + \lambda_n \left(\text{prox}_{\gamma_n f_1}(2\mathbf{x}_n - \mathbf{y}_n) - \mathbf{x}_n\right).
\end{aligned} \tag{4.18}$$

Douglas-Rachfordův algoritmus stejně jako dopředně-zpětný algoritmus využívá rozdělení tak, že každou funkci z úlohy (4.13) využívá odděleně. Na rozdíl od dopředně-zpětného algoritmu však nevynucuje podmínku pro diferencovatelnost jedné funkce podle Lipschitzova spojitého gradientu a je tak univerzálnější, ale je to za cenu pomalejších výpočtů.

4.4.3 Dekvantizace pomocí Douglas-Rachfordova algoritmu

Prvním krokem pro vyřešení základní úlohy dekvantizace (4.1) pomocí Douglas-Rachfordova algoritmu je převést úlohu na neomezený tvar. Funkce f_1 pak bude mít tvar

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1. \tag{4.19}$$

Jak ale do tohoto tvaru aplikovat omezení, že hledáme taková $\mathbf{x}$, která splňují podmínku $\|\mathbf{y}_q - \mathbf{A}\mathbf{x}\|_\infty \leq \alpha/2$? Odpověď na tuto otázku je poměrně jednoduchá. Použijeme výše zmíněnou indikátorovou funkci ι_C, definovanou podle (4.11).

Kompletní dekvantizační úlohu (4.1) v neomezeném tvaru pak můžeme formulovat jako [8]

$$\min_{\mathbf{x}} \underbrace{\|\mathbf{x}\|_1}_{f_1} + \underbrace{\iota_C}_{f_2}. \tag{4.20}$$

Toto řešení je v praxi velmi elegantní. Pokud totiž najdeme řešení, které spadá do množiny přípustných řešení C, indikátorová funkce bude rovna nule a budeme řešit pouze minimalizaci tvaru $\|\mathbf{x}\|_1$. Pokud naopak řešení bude ležet mimo množinu C, ι_C bude rovno $+\infty$ a tvar $\|\mathbf{x}\|_1$ bude zanedbatelný. Pak toto řešení můžeme vyloučit, protože je jasné, že zaručeně nebude nejřidší.

Proximální operátory neomezeného tvaru (4.20) pak budou vycházet z definic pro měkké prahování (4.10) a projekci indikátorové funkce (4.11).

Po stanovení proximálních operátorů už můžeme sestavit konkrétní znění Douglas-Rachfordova algoritmu, který použijeme při řešení dekvantizační úlohy [30]:

$$\begin{aligned}
&\text{Zvolíme } \epsilon \in (0,1), \gamma > 0, \mathbf{y}_0 \in \mathbb{R}^N \\
&n = 0, 1, 2 \ldots \\
&\mathbf{x}_n = P_{\gamma C}\mathbf{y}_n \\
&\lambda_n \in \langle \epsilon, 2-\epsilon \rangle \\
&\mathbf{y}_{n+1} = \mathbf{y}_n + \lambda_n \left(\gamma \mathrm{soft}_{\langle \underline{\omega}, \overline{\omega} \rangle}(2\mathbf{x}_n - \mathbf{y}_n) - \mathbf{x}_n \right).
\end{aligned} \tag{4.21}$$

4.4.4 Kvadratické programování

Kvadratické programování je obdobně jako lineární programování 4.3 odvětví optimalizace, které řeší úlohu:

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} + \mathbf{f}^\top \mathbf{x} \quad \text{vzhledem k} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}, \tag{4.22}$$

kde $\mathbf{H}$ je symetrická matice velikosti $N \times N$, $\mathbf{f}$ je N-rozměrný vektor, $\mathbf{A}$ je matice rozměru $m \times N$, $\mathbf{b}$ je m-rozměrný vektor a konečně $\mathbf{x}$ reprezentuje N-rozměrný vektor neznámých [26].

4.4.5 Projekce na přípustnou množinu

V této části ještě bude popsána samotná projekce na množinu C a její možné způsoby řešení při použití v dekvantizační úloze. Jak již bylo uvedeno dříve, projekce na množinu C je proximální operátor indikátorové funkce.

Při vlastní projekci $P_C\mathbf{x}$ vlastně hledáme

$$\arg\min_{\mathbf{z}} \|\mathbf{x} - \mathbf{z}\|_2 \quad \text{vzhledem k} \quad \mathbf{z} \in C. \tag{4.23}$$

Omezovací podmínka pro projekci, neboli množina možných výsledků C, zde vychází z rovnic definovaných v části 4.2. Pro vyřešení této úlohy pak můžeme použít kvadratické programování. Musíme ale dále specifikovat matici $\mathbf{H}$ a vektory $\mathbf{f}$ a $\mathbf{b}$, které

určíme následovně: pokud víme, že při projekci zmenšujeme rozdíl dvou vektorů, dokud projektovaný vektor nespadne do množiny přípustných řešení C, můžeme rozdíl těchto dvou vektorů z ℓ_2-normy přepsat na tvar, který lépe odpovídá potřebám kvadratického programování [30]:

$$\|\mathbf{x}-\mathbf{z}\|_2^2 = (\mathbf{x}-\mathbf{z})^\top(\mathbf{x}-\mathbf{z}) = \mathbf{x}^\top\mathbf{x} - 2\mathbf{z}^\top\mathbf{x} + \mathbf{z}^\top\mathbf{z}. \tag{4.24}$$

Z porovnání rovnic (4.22) a (4.24) pak můžeme jednoduše odvodit, že matice $\mathbf{H}$ bude jednotková, pouze kvůli zlomku v definici pro kvadratické programování ji vynásobíme dvěma. Vektor $\mathbf{f}$ pak bude reprezentovat člen $-2\mathbf{z}$, kde $\mathbf{z}$ představuje vektor bodů, které budeme projektovat. Vektor $\mathbf{b}$ pak můžeme obecně stanovit z omezovací podmínky jako pravou stranu nerovnice (4.5). Absolutní člen $\mathbf{z}^\top\mathbf{z}$ v případě hledání řešení úlohy argmin nemá vliv na výsledek a můžeme ho tedy zanedbat.

Výsledné řešení projekce pomocí kvadratického programování má velkou výhodu v možnosti realizovat takovýmto způsobem projekce prakticky pro jakékoli slovníky $\mathbf{A}$. Nevýhodou při programovém řešení však je, že se celá tato matice zavádí do operační paměti, což znemožňuje použít tuto metodu pro rozměrnější signály, např. pro reálné obrazy.

Pokud ovšem zpřísníme podmínky pro matici $\mathbf{A}$ a budeme uvažovat pouze ortonormální báze, můžeme projekci provést v oblasti signálu (na rozdíl od kvadratického programování, kde projekce probíhala v oblasti souřadnic) a vyhnout se tak počítání s explicitně vyjádřenou maticí $\mathbf{A}$.

Zde je důkaz, že pro ortonormální báze můžeme projekci v oblasti signálu použít. Budeme vycházet z původní definice pro projekci $P_C\mathbf{x}$, tedy z úlohy (4.23). Dále víme, že pokud je matice $\mathbf{A}$ ortonormální, je současně také unitární, což znamená, že platí $\|\mathbf{Ay}\|_2 = \|\mathbf{y}\|_2$, čehož s výhodou využijeme při následující úvaze. Původní tvar úlohy (4.23) pak můžeme zapsat jako

$$\arg\min_{\mathbf{x}} \|\mathbf{Ax}-\mathbf{Az}\|_2 \quad \text{vzhledem k} \quad \mathbf{z} \in C. \tag{4.25}$$

Pokud součin $\mathbf{Az}$ nahradíme $\mathbf{u}$, pak platí, že $\mathbf{z} = \mathbf{A}^{-1}\mathbf{u}$ a rovnice přechází na tvar

$$\mathbf{A}^{-1} \cdot \arg\min_{\mathbf{u}} \|\mathbf{Ax}-\mathbf{u}\|_2 \quad \text{vzhledem k} \quad \mathbf{u} \in \mathbf{A}\cdot C, \tag{4.26}$$

což není nic jiného než projekce bodu $\mathbf{Ax}$ na množinu $\mathbf{A}\cdot C$ vynásobená inverzní maticí $\mathbf{A}$. Pak tedy platí, že

$$P_C(\mathbf{x}) = \mathbf{A}^{-1}\cdot P_{\mathbf{A}\cdot C}(\mathbf{Ax}) \tag{4.27}$$

a my můžeme s jistotou tvrdit, že pokud provedeme projekci v oblasti signálu, tedy projekci $P_{\mathbf{A}\cdot C}(\mathbf{Ax})$, a následně se vynásobením inverzní maticí vrátíme zpět do oblasti souřadnic, je výsledek identický, jako při projekci přímo v oblasti souřadnic. Musíme si však uvědomit, že toto zjednodušení platí pouze pro ortonormální báze.

5 PROGRAMOVÉ ŘEŠENÍ

V této kapitole se zaměřím už na samotnou realizaci jednotlivých algoritmů pro dekvantizaci. Všechny algoritmy byly napsány ve výpočetním software Matlab ve verzi 2014b, který se pro podobné účely hodí hned z několika důvodů. Prvním z nich je relativní jednoduchost programování, na rozdíl například od jazyka C. Při programování je pak možné se lépe soustředit na vlastní implementaci algoritmu, než na samotné programování. Další výhodou tohoto programu je fakt, že Matlab standardně pracuje se všemi proměnnými jako s maticemi, což velmi usnadní práci při zpracovávání obrázků. V neposlední řadě bych zmínil ještě velké množství volně dostupných funkcí (např. výpočet hodnot PSNR, SSIM, DCT a DWT transformace, funkce pro lineární, popř. kvadratické programování, atp.) a velmi kvalitní dokumentaci jednotlivých funkcí přímo na webu Mathworks, autorů Matlabu.

Tato kapitola naváže na předchozí kapitolu Teoretické řešení a popíše realizaci dekvantizačního algoritmu nejprve pro 1D signál, poté pro 2D signál. Nakonec bude uvedeno i řešení aplikace pro reálný obraz, což je hlavním cílem této bakalářské práce. V poslední části, která se zabývá zpracováním reálného obrazu, budou zobrazeny jak ukázky dekvantovaných obrázků z 2 bpp, tak i ze 4 bpp, rekonstruovaných pomocí DCT i DWT a budou popsány výhody a nevýhody jednotlivých bází.

5.1 1D signál

Při vývoji metody pro zvýšení bitové hloubky na základě řídké reprezentace jsme se pro jednoduchost nejprve zaměřili na 1D signál. Takový signál bylo nejprve nutno vytvořit a nakvantovat, abychom poté mohli provést dekvantizaci a srovnat její výsledky.

Hlavním programem balíku 1D je dávkový soubor `main_1D`. Skládá se ze dvou hlavních částí – nastavení a implementace. V části nastavení si může uživatel zvolit velikost signálu N, řídkost signálu k, počet hladin pro kvantizaci d. Dále pak typ použitého slovníku (DCT nebo DWT), v případě DWT pak také typ použitého waveletu a hloubku dekompozice J. V neposlední řadě je pak možné zvolit algoritmus pro dekvantizaci. Možnosti jsou dekvantizace pomocí Lineárního programování (viz část 4.3.2) nebo dekvantizace pomocí Douglas-Rachfordova algoritmu s projekcí v oblasti signálu nebo v oblasti souřadnic (viz část 4.4.3).

Ke generování náhodného signálu slouží funkce `generator`, která ze zadaného slovníku $\mathbf{A}$ a řídkosti k vygeneruje náhodný signál. Pro generaci slovníku je použita funkce `slovnik`, která podle zadaných vstupních parametrů generuje čtvercovou matici $\mathbf{A}$ o velikosti N koeficientů DCT, popř. DWT. Pro waveletové koeficienty je

pak možné ještě specifikovat konkrétní wavelet a hloubku dekompozice.

Vygenerovaný signál je následně kvantován pomocí funkce `kvantizace`. Vstupními parametry této funkce je kromě samotného vstupního signálu také počet kvantovacích hladin d. Proces kvantizace je zde uniformní a lineární, výsledný signál však není souměrný podle nuly. Pro naše potřeby však takováto kvantizace plně dostačuje.

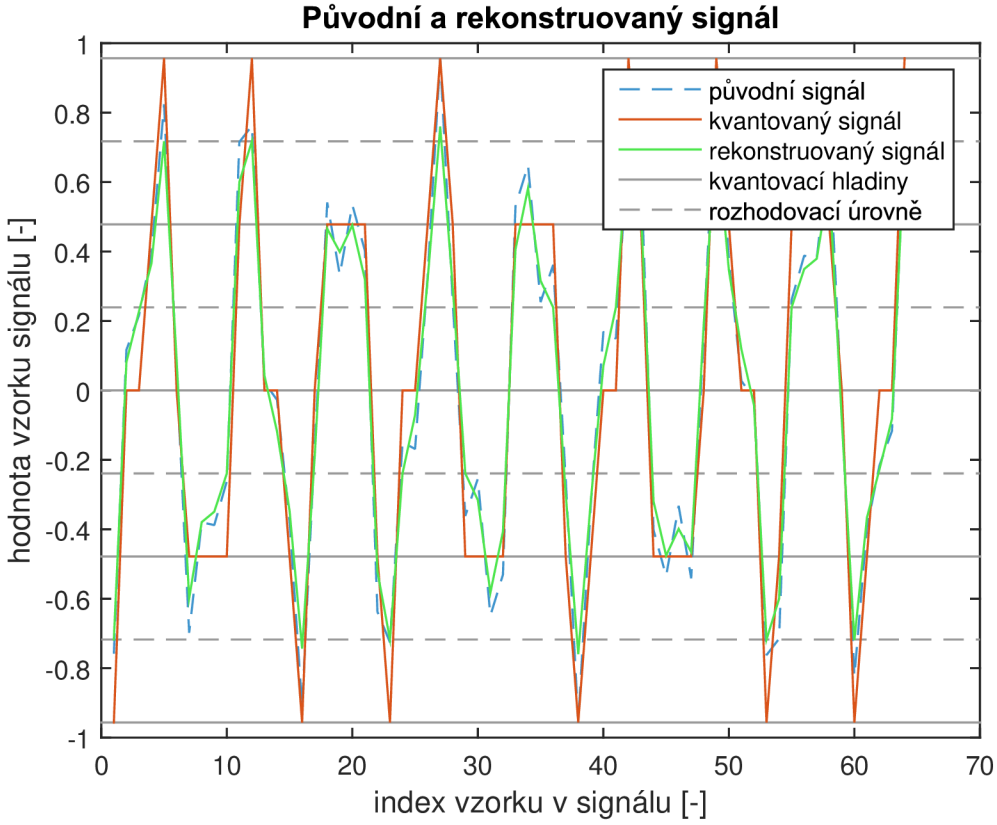
Jak již bylo napsáno výše, dekvantizaci lze v tomto programu provést třemi způsoby. Pomocí lineárního programování (funkce `dekvantizace_LP`) se nejprve stanoví vektor $\mathbf{b}$, který představuje hranice rekonstruovaného signálu, což odpovídá pravé straně nerovnosti v podmínce úlohy (4.7). Samotné lineární programování je pak realizováno funkcí `linprog`. Výsledkem je vektor $\mathbf{w}$, který obsahuje vektory $\mathbf{u}$ a $\mathbf{v}$ (viz část 4.3.2). Pokud oba vektory separujeme a odečteme, získáme rekonstruovaný vektor souřadnic. Po vynásobení slovníkem $\mathbf{A}$ pak dostáváme výsledný dekvantovaný signál. V případě dekvantizace podle Douglas-Rachfordova algoritmu se také nejprve stanoví rozšířená matice $\mathbf{A}$ a vektor $\mathbf{b}$ podle (4.7). Dále je definována funkce měkkého prahování `soft` s prahem $\omega = 1$. Po stanovení uvolňovacího parametru $\lambda = 1$ pak následuje hlavní část Douglas-Rachfordova algoritmu, viz (4.21). Tento algoritmus může pro svůj účel provádět projekci na přípustnou množinu v oblasti souřadnic nebo pro ortonormální báze v oblasti signálu (viz část 4.4.5). Pro dekvantizaci s projekcí v oblasti souřadnic slouží funkce `dekvantizace_DRA_souradnice`, kde projekce v oblasti souřadnic se nachází ve funkci `projekce_souradnice` a je realizována pomocí kvadratického programování (viz část 4.4.4) funkcí `quadprog`. V případě použití dekvantizace s projekcí v oblasti signálu se volá funkce `dekvantizace_DRA_signal`. Tato funkce je velmi obdobná dekvantizaci s projekcí v oblasti souřadnic, pouze se pro projekci používá funkce `projekce_signal_dct`, resp. `projekce_signal_dwt` v závislosti na použité bázi. Do projekce vstupuje signál ve formě souřadnic, na začátku se transformuje do oblasti signálu, zde proběhne projekce na přípustnou množinu a výsledek se opět transformuje do oblasti souřadnic pro další iteraci algoritmu.

Pro všechny funkce používající Douglas-Rachfordův algoritmus platí, že iterace probíhají tak dlouho, dokud relativní odchylka n-tého a $(n-1)$ řešení nebude menší než stanovená hodnota. Pro vektory používáme pro tento případ ℓ_2-normu vektorů. Pro všechny výpočty dekvantizace pro 1D signál byla hodnota relativní odchylky stanovena na 0,0005.

Protože hranice pro dekvantizaci jsou přesně velikost kvantovacího kroku, může se stát, že rekonstruovaný signál dopadne přesně doprostřed mezi kvantovací kroky, tedy na rozhodovací hladinu. Když je pak rekonstruovaný signál znovu kvantován, může nastat situace, že kvantizační funkce nakvantuje tento bod signálu na vedlejší kvantovací hladinu, než byl původní signál. Proto vznikla funkce `fix_dekvantizace`,

která detekuje, zda úroveň rekonstruovaného signálu leží na rozhodovací hladině. Pokud ano, nepatrně (o hodnotu 0,0001) ho posune směrem ke kvantovací úrovni.

Funkce s předponou `demo` byly použity především v průběhu programování metod a sloužily pro odhalení chyb v kódu a snadnější debugování. Ve většině případů tyto funkce zobrazují grafy aktuálního signálu, kvantovací šum, kontrolují, zda se rekonstruovaný nakvantovaný signál a původní nakvantovaný signál shodují, atp...

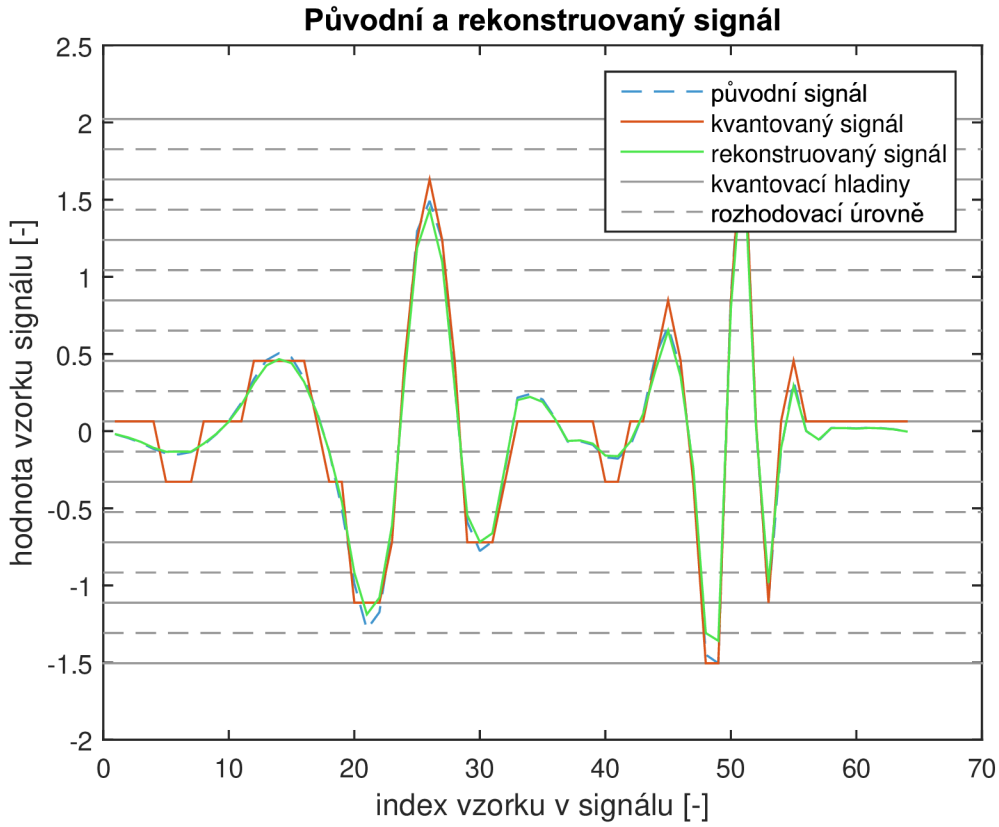


Obr. 5.1: Výsledek dekvantizace pro signál řídký v DCT

Na obrázku 5.1 můžeme vidět výsledek dekvantizace 1D signálu, který je řídký v DCT. Relativní řídkost je zde $\frac{2}{64}$. Výsledku je dosaženo dekvantizací pomocí Douglas-Rachfordova algoritmu s projekcí v oblasti signálu.

Na obrázku 5.2 pak lze vidět výsledek dekvantizace pro signál řídký ve waveletové transformaci. Relativní řídkost je zde opět $\frac{2}{64}$. Použitý wavelet je Daubechies 5 s hloubkou dekompozice 5. Výsledku bylo dosaženo pomocí lineárního programování.

Ve většině případů algoritmy lineárního programování a Douglas-Rachfordův algotitmus najdou totožné řešení minimalizační úlohy. V některých případech, kdy existuje nekonečně mnoho řešení (viz část 3.3), oba algoritmy najdou řešení s minimální ℓ_1-normou, avšak tato řešení nejsou stejná. Experimentálně bylo zjištěno, že v těchto případech se algoritmus lineárního programování přiblíží lépe originálnímu



Obr. 5.2: Výsledek dekvantizace pro signál řídký v DWT, wavelet db5, hloubka dekompozice 5

signálu u signálů řídkých ve waveletové transformaci s menší relativní řídkostí. Naopak Douglas-Rachfordův algoritmu podává mírně lepší výsledky pro signály řídké v DCT a pro vyšší řídkosti.

5.2 2D signál

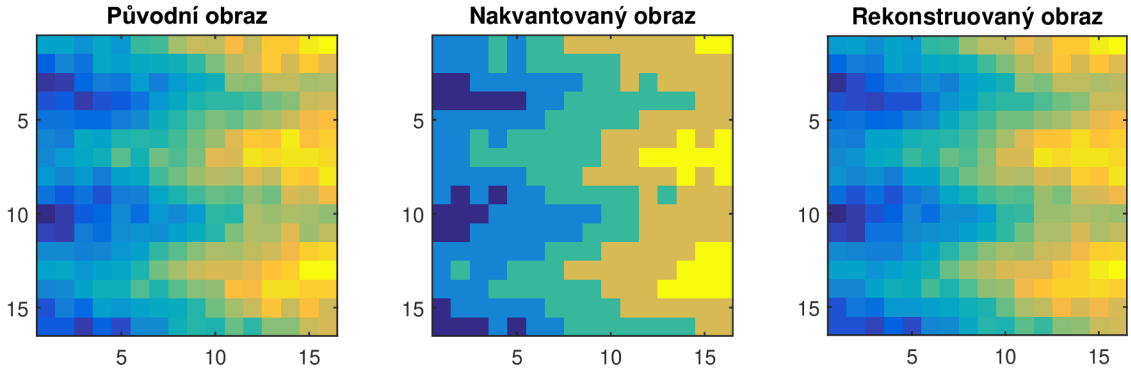
Balík funkcí 2D pro vytvoření, nakvantování a následnou dekvantizaci 2D signálu je obsahově velmi podobný balíku pro 1D. Hlavním programem balíku je dávkový soubor `main_2D`, který se obdobně jako hlavní program pro 1D signál skládá ze dvou částí – nastavení a implementace. Možnosti nastavení jsou obdobné jako u 1D signálu, pouze odpadla možnost nastavení dekvantizačního algoritmu. Pro 2D signál už byl zachován pouze Douglas-Rachforův algoritmus, především z důvodu nízké hardwarové náročnosti a rychlosti zpracování. Nízká hardwarová náročnost metody spočívá především v použité projekci, díky které se přímo nepracuje se slovníkem $\mathbf{A}$, což umožňuje použití této metody i pro reálné obrazy s vysokým rozlišením.

Pro generaci náhodného 2D signálu slouží funkce `generator_2D`, která vytvoří

čtvercovou matici $\mathbf{X}$ s k nenulovými prvky. Výsledný 2D signál se pak vytvoří jako součin matice $\mathbf{X}$ se slovníkem $\mathbf{A}$ a s transponovaným slovníkem.

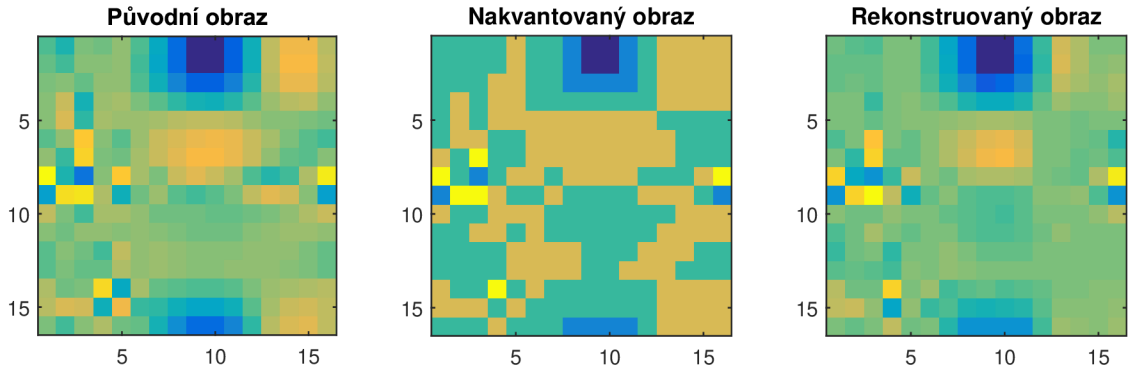
Tento signál se opět nakvantuje pomocí funkce `kvantizace_2D`. Dekvantizace probíhá obdobně, jako v případě 1D signálu. Hlavním rozdílem je použití transformačních funkcí pro 2D signál (`dct2, idct2, waverec2` a `wavedec2`).

Na obrázku 5.3 můžeme vidět vlevo původní obraz, uprostřed obraz nakvantovaný na 5 hladin a vpravo pak výsledný rekonstruovaný obraz. Koeficient řídkosti původního obrazu je $k = 5$. Pro zobrazení 2D signálu je použito pseudobarevné schéma.



Obr. 5.3: Výsledek dekvantizace pro 2D signál řídký v DCT

Na obrázku 5.4 je pak srovnání dekvantizace pro waveletovou transformaci. Koeficient řídkosti je $k = 5$. Jako mateřský wavelet je opět použit Daubechies 5 s hloubkou dekompozice 2. Signál je kvantován opět na 5 hladin.



Obr. 5.4: Výsledek dekvantizace pro 2D signál řídký v DWT, wavelet db5, hloubka dekompozice 2

5.3 Reálný obraz

Pro aplikaci dekvantizace na základě řídkých reprezentací pro reálné obrazy vznikly dvě velmi podobné funkce – `Sparse_DCT` a `Sparse_DWT`. Jak již název napovídá, liší se pouze použitými bázemi. Vstupními parametry obou funkcí jsou obraz s nízkou bitovou hloubkou (LBDI), bitová hloubka LBDI obrazu a nová bitová hloubka. V případě použití waveletové báze jsou pak dalšími parametry ještě použitý wavelet a hloubka dekompozice.

Při importu jakékoli šedotónové fotografie do Matlabu je datový typ importovaného obrazu uint8. To znamená, že každé číslo matice (každý pixel) je vyjádřeno 8 bity. Může tedy nabývat hodnot 0 až 255. Pokud ale importujeme např. obrázek s 2bitovou hloubkou, nebudou hodnoty pixelů 0, 1, 2 a 3, jak bychom čekali, ale 0, 85, 170 a 255. Pro další výpočet je potřeba znát kvantovací krok. Na základě předchozích znalostí ho můžeme určit jako $255/(2^p - 1)$, kde p je bitová hloubka obrazu s nízkou bitovou hloubkou.

Dalšími kroky obou metod jsou, stejně jako v předchozích případech, definování funkce `soft` pro měkké prahování, inicializace uvolňovacího parametru λ a relativní odchylky. Pro urychlení celého algoritmu jako počáteční hodnotu pro iterace nevolíme nuly, ale hodnotu kvantovaného obrazu, od kterého po složkách odečteme hodnotu kvantovacího kroku. Přiblížíme se tak rychleji k řešení, zároveň se ale v počátečním odhadu nedostaneme do oblasti přípustných řešení. Pokud je počáteční odhad zvolen přímo v oblasti přípustných řešení, algoritmus má problémy s konvergencí. Algoritmus samozřejmě pracuje v oblasti souřadnic, proto je výchozí odhad transformován do DCT funkcí `dct2` nebo do DWT funkcí `wavedec2`.

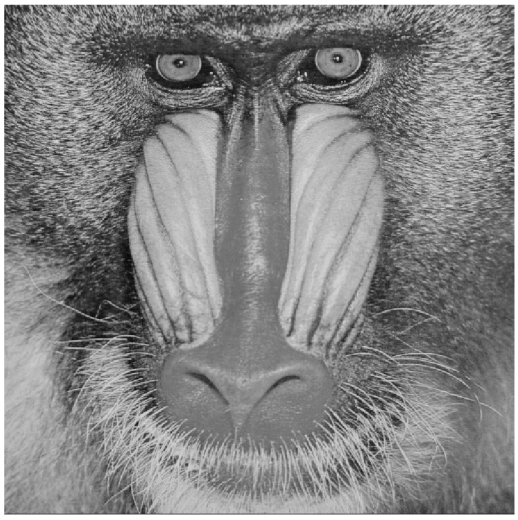
Samotný Douglas-Rachfordův algoritmus a projekce jsou stejné jako v případu pro 2D signál, proto je zde nebudu již dále popisovat.

V balících 1D a 2D bylo nutné vyřešit případ, kdy by se rekonstruovaný signál v průběhu dekvantizace dostal nad horní rozhodovací úroveň nejvyšší kvantovací hladiny, popř. pod dolní úroveň nejnižší kvantovací hladiny. Tento signál by pak byl projektován na hranici $\alpha/2$ od kvantovacích hladin. Původní signál však předpokládáme pouze mezi kvantovacími hladinami. V předchozích balících byl tento problém vyřešen přidáním dalších proměnných, detekcí a opravou takovéto situace. Ve funkcích pro zpracování reálného obrazu je vyřešení těchto případů ještě jednodušší. Předpokládáme, že signál pro dekvantizaci se pohybuje mezi hodnotami 0 až 255. Pak pro proměnné dolní mez a horní mez zvolíme úmyslně datový typ uint8. Výsledkem bude stav, že horní i dolní mez budou mít maximum 255 a minimum 0. Nemůže tak nastat situace, že bychom projektovali signál do oblasti, ve které nepředpokládáme, že se vyskytoval.

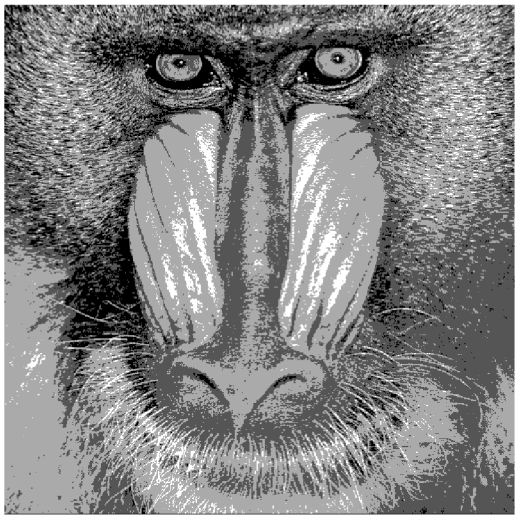
V posledním kroku metody pak signál převedeme z oblasti souřadnic do oblasti

signálu a hodnoty nakvantujeme pro novou bitovou hloubku. Díky tomu je možné zvýšit bitovou hloubku obrazu až na 16 bitů, kde je použit datový typ uint16.

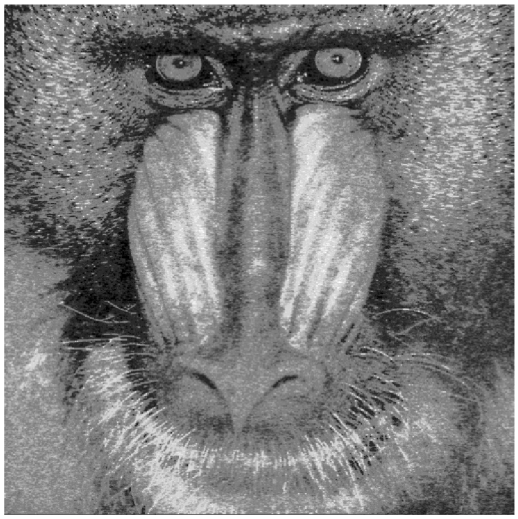
Na sadě obrázků 5.5 můžeme vidět výsledky dosažené pomocí dekvantizace na základě řídké reprezentace signálů. Vlevo nahoře (5.5a) je původní 8bitový obrázek. Vpravo nahoře (5.5b) je obrázek degradovaný pouze na 2 bity. Dole jsou pak výsledky dekvantizace pomocí bází DCT (5.5c) a DWT (5.5d). Pro DWT byl použit wavelet Daubechies 5 (db5) s hloubkou dekompozice 8.



(a) Původní 8bitový obrázek



(b) Kvantovaný 2bitový obrázek



(c) Rekonstruovaný obrázek pomocí DCT

(d) Rekonstruovaný obrázek pomocí DWT

Obr. 5.5: Srovnání výsledků dekvantizace ze 2 bitů zpět na 8 bitů, obrázek Baboon

Na další sadě obrázků 5.6 pak můžeme výsledky pro dekvantizaci ze 4 bitů. Pořadí obrázků je stejné jako v předchozím případě. Pro DWT byl opět použit wavelet db5 s hloubkou dekompozice 8.

(a) Původní 8bitový obrázek

(b) Kvantovaný 4bitový obrázek

(c) Rekonstruovaný obrázek pomocí DCT

(d) Rekonstruovaný obrázek pomocí DWT

Obr. 5.6: Srovnání výsledků dekvantizace ze 4 bitů zpět na 8 bitů, obrázek Lenna

Při srovnání výsledků dekvantizace u obrázků pomocí DCT a DWT je patrné, že při použití DCT je rekonstruovaný obraz značně zašuměný. Při vysoké degradaci původního obrázku (velmi hrubého nakvantování) jsou však i přes vzniklý šum lépe zachovány detaily obrazu, konkrétně například špička nosu u obrázku Baboon (viz 5.5). Naopak DWT se hodí pro obrazy, které jsou nakvantované jemněji a podává dle mého subjektivního názoru lepší výsledky s minimem šumu.

6 VÝSLEDKY

Pro porovnání kvality výsledků metod pro zvýšení bitové hloubky na základě řídké reprezentace bylo vybráno pět testovacích obrázků. Konkrétně to jsou obrázky Baboon, Boat, Lenna, Peppers a Sunset. Tyto obrázky se standardně používají pro srovnávání výsledků různých metod pracujících s obrázky. Těchto pět obrázků bylo postupně ze standardní 8bitové hloubky degradováno na 2, 4 a 6 bpp a následně pomocí několika metod dekvantováno zpět na 8 bpp. Z metod řídkých reprezentací byla zahrnuta jak metoda používající DCT, tak i DWT. Pro metodu používající DWT byl pro všechny obrázky použit mateřský wavelet Daubechies 5 (db5) s hloubkou dekompozice 8. Původním plánem bylo i srovnat výsledky jednotlivých mateřských waveletů. Bylo však zjištěno, že pro 2D signál má volba mateřského waveletu pouze nepatrný vliv.

Přestože bylo naprogramováno více metod popsaných v kapitole 2, porovnávám výsledky dekvantizace pomocí řídké reprezentace signálů pouze s „vyspělejšími“ metodami, které nějakým způsobem řeší odstranění falešných kontur. Konkrétně jsou pro srovnání použity metody Spatial Varying Filter (viz část 2.6), Adaptive Bit-Depth Expansion včetně pouze první fáze ABDE-P1 (viz část 2.7) a Contour Region Reconstruction (viz část 2.8).

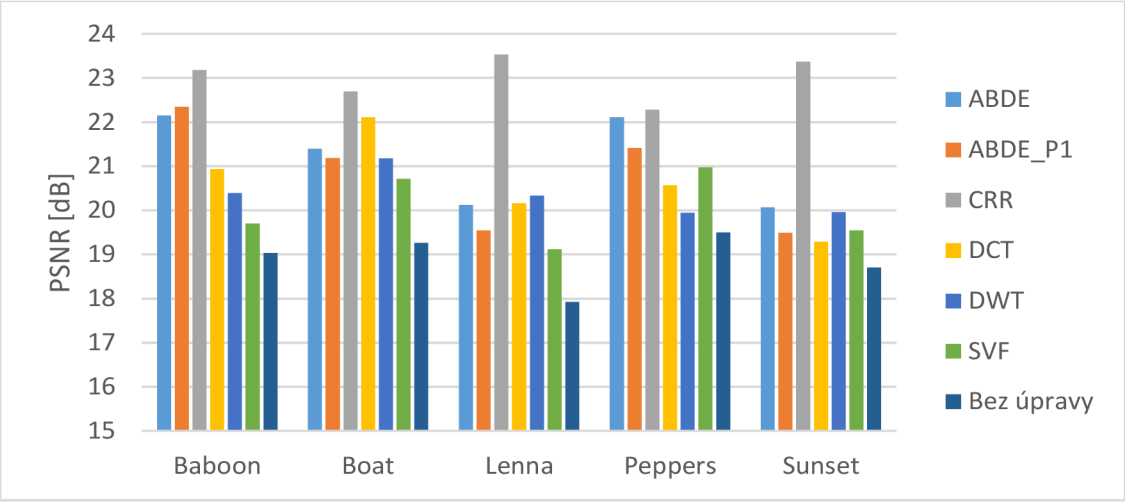
Porovnání jednotlivých metod zahrnuje objektivní srovnání pomocí ukazatelů PSNR a SSIM, subjektivní srovnání pomocí online dotazníku a v neposlední řadě je porovnána také výpočetní náročnost jednotlivých metod.

6.1 Objektivní srovnání

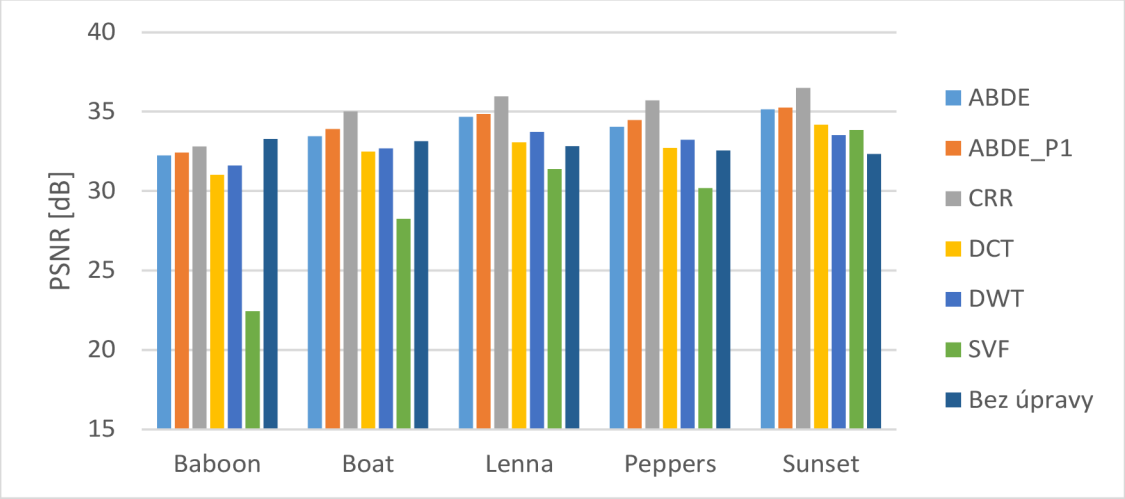
6.1.1 PSNR

Na následujících grafech je uvedeno srovnání PSNR hodnot dekvantovaných obrázků pro různé metody. Poslední sloupec je vždy PSNR hodnota přímo LBDI obrazu, který nebyl upraven žádnou metodou. Lze tak dobře poznat, o kolik dB pomohla metoda obraz vylepšit.

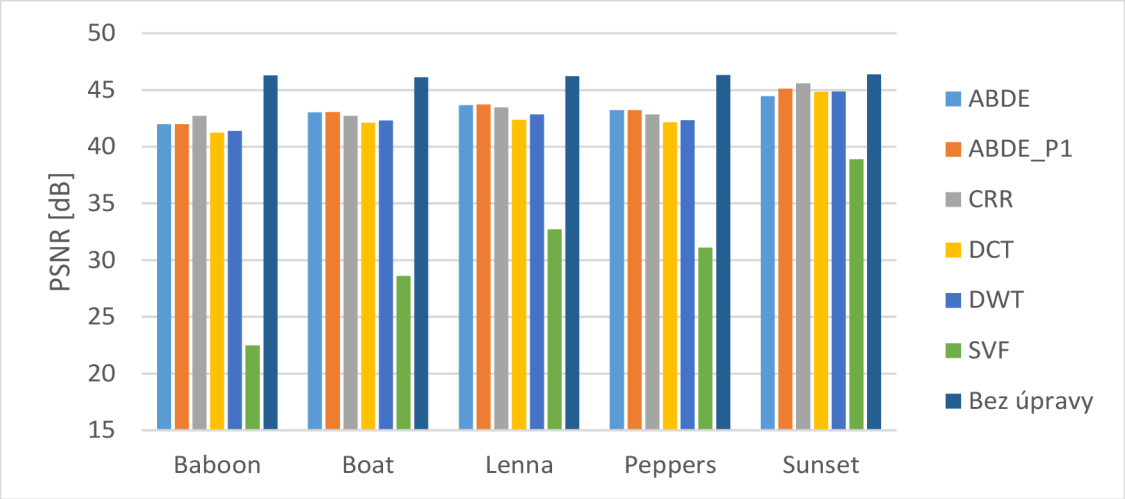
Z níže uvedených grafů je patrné, že pro zvýšení bitové hloubky ze 2 bitů zpět na 8 dominovala metoda Contour Region Reconstruction. Dále si lze všimnout, že obrázek bez úpravy má ve všech případech nejhorší PSNR. Lze tedy říct, že každá metoda obrázek alespoň nějak vylepšila. Toto tvrzení už neplatí pro dekvantizaci ze 4 bpp, kde je vidět, že metoda Spatial Varying Filter značně zaostává. Výsledky dosažené ostatními metodami jsou, oproti dekvantizaci z 2 bpp vyrovnanější. Pro obrázky zvyšované z 6 bpp pak platí, že nejvyšších hodnot PSNR dosahuje původí 6bitový obraz. Výsledky ostatních metod jsou pak, s výjimkou SVF, velmi vyrovnané.



Obr. 6.1: Grafy hodnot PSNR pro dekvantizaci ze 2 bpp na 8 bpp



Obr. 6.2: Grafy hodnot PSNR pro dekvantizaci ze 4 bpp na 8 bpp

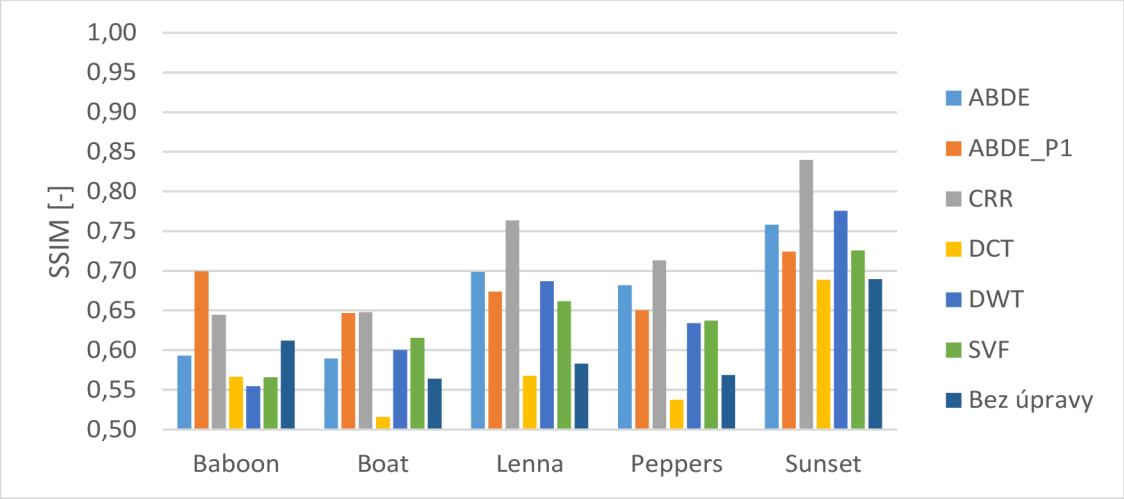


Obr. 6.3: Grafy hodnot PSNR pro dekvantizaci ze 6 bpp na 8 bpp

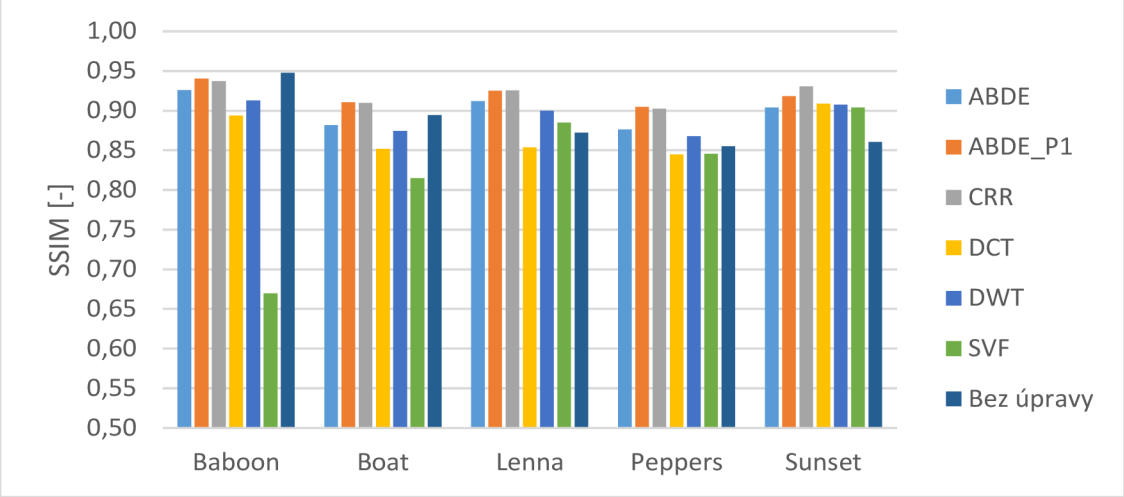
6.1.2 SSIM

Hodnocení podle identifikátoru SSIM (viz část 1.6.2) by teoreticky mělo odpovídat lépe lidskému vnímání obrazu.

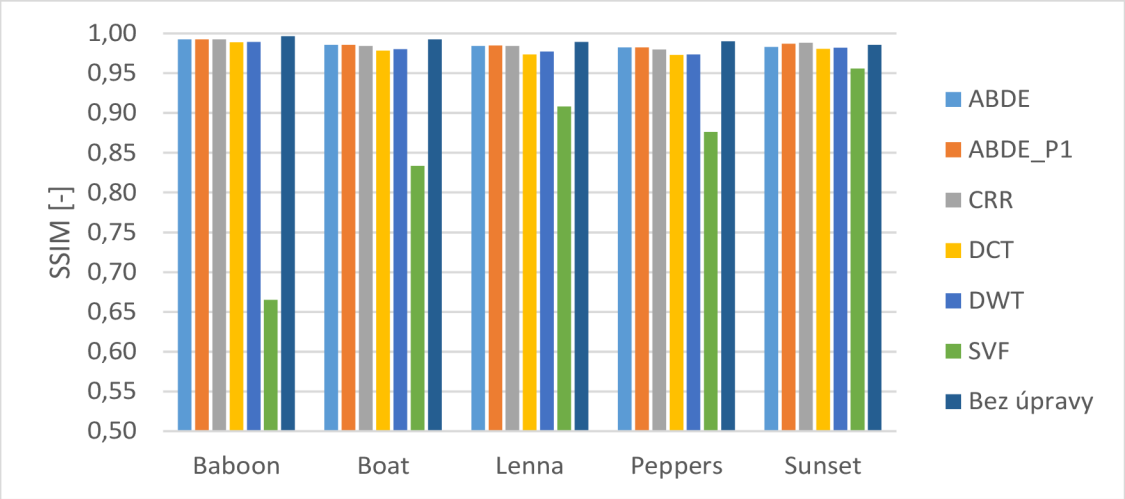
Při srovnání výsledků je patrné, že si stále velmi dobře vede metoda CRR, naopak metoda SVF podává nejhorší výsledky z testovaných metod. Relativně špatně si podle identifikátou SSIM vede i dekvantizační metoda využívající DCT. Je to patrně proto, že identifikátor SSIM je náchylnější na zašuměné obrazy, které tato metoda produkuje. Při pohledu na graf 6.6 je patrné, že s výjimkou SVF už je prakticky nemožné rozeznat rozdíly mezi jednotlivými metodami.



Obr. 6.4: Grafy hodnot SSIM pro dekvantizaci ze 2 bpp na 8 bpp



Obr. 6.5: Grafy hodnot SSIM pro dekvantizaci ze 4 bpp na 8 bpp



Obr. 6.6: Grafy hodnot SSIM pro dekvantizaci ze 6 bpp na 8 bpp

6.2 Subjektivní srovnání

Pro subjektivní srovnání byl pomocí technologie Google Forms vytvořen webový dotazník, který měl za cíl srovnat výsledky jednotlivých metod pro zvýšení bitové hloubky pomocí subjektivního vnímání jednotlivých respondentů. Pro potřeby tohoto dotazníku byly vybrány tři obrázky (Baboon, Lena a Sunset). Obrázek Baboon je složen z velké části z malých detailů, naopak Sunset je tvořen především pomalými gradienty. Tyto obrázky byly degradovány na bitovou hloubku 2 bpp a 4 bpp a následně metodami (stejnými jako v případě objektivního srovnání) dekvantovány zpět na 8 bpp. Pro dekvantizaci ze 6 bpp nebylo subjektivní hodnocení uskutečněno, protože rozdíly mezi jednotlivými obrázky už jsou minimální. U každého obrázku pak byla škála od 1 do 10, kde uživatel vyplnil, jak dobře se metodě povedlo přiblížit se originálnímu 8bitovému obrázku.

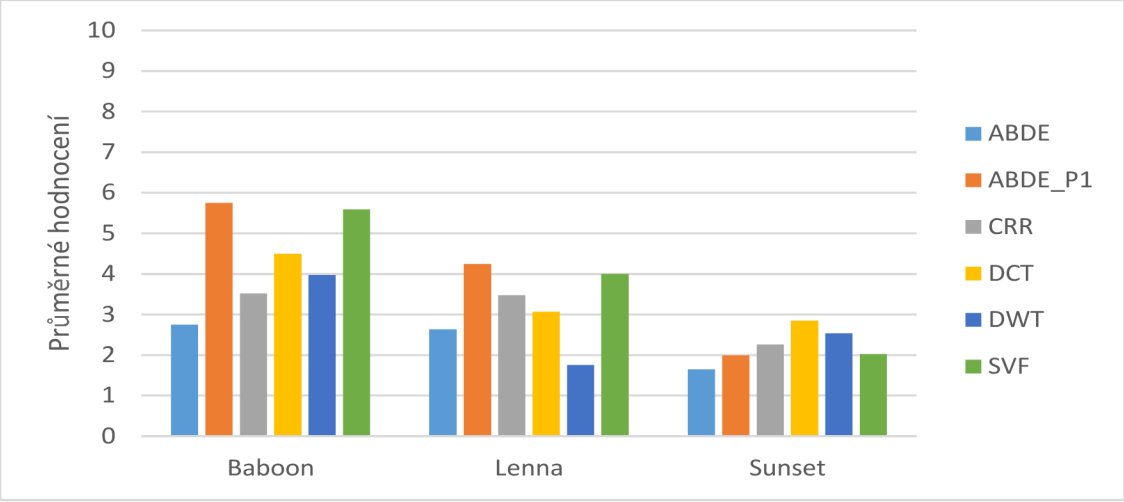
Dotazník vyplnilo celkem 121 respondentů. Výsledky dotazníku byly podrobeny korelační analýze pro odhalení irelevantních výsledků (například převrácení škály hodnot). Nevyhovující výsledky byly smazány a výsledná data pak vycházejí ze 107 hodnocení.

Výsledné hodnoty hodnocení uživatelů byly zprůměrovány. Graf 6.7 obsahuje data z hodnocení dekvantizace 2bitových obrázků a graf 6.8 pak data z dekvantizace 4bitových obrázků. Už na první pohled je patrné, že subjektivní vnímání respondentů plně neodpovídá objektivnímu hodnocení podle PSNR či SSIM. Při velké degradaci původního obrázku uživatelé preferují jednodušší metody (ABDE_P1 a SVF), které sice nepotlačí falešné kontury, ale zachovávají detaily obrázků. U obrázku Sunset s velmi pomalým gradientem pak respondenti vyhodnotili jako nejlepší

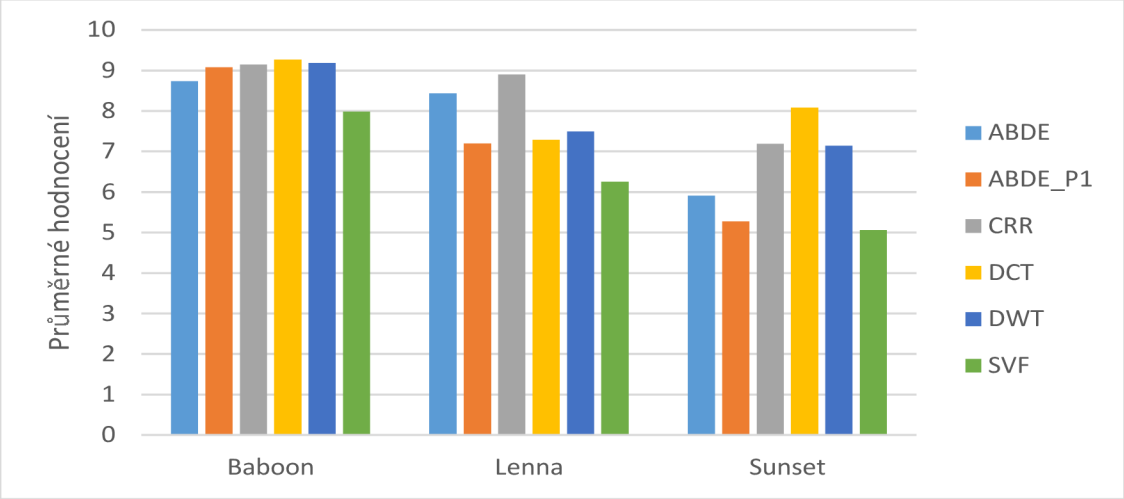
metodu řídké reprezentace s DCT bází, která i přes vyskytující se šum v obrázku podává pravděpodobně nejlepší výsledek z testovaných metod.

U subjektivního hodnocení dekvantizace 4bitových obrázků už jsou rozdíly mezi jednotlivými metodami menší než v předchozím případě. Jednoduché metody typu ABDE_P1 a SVF pomalu zaostávají, naopak metody založené na řídké reprezentaci podávají v tomto testu velmi dobré výsledky i přes to, že v objektivním hodnocení ztrácely.

Z těchto grafů se potvrzuje i myšlenka, že nízká bitová hloubka ovlivní více kvalitu fotografií s pomalými gradienty než obrázky s mnoha detaily. Obrázek Baboon má přibližně o 20% lepší hodnocení než obrázek Sunset.



Obr. 6.7: Výsledky subjektivního hodnocení pro dekvantizaci ze 2 bpp na 8 bpp

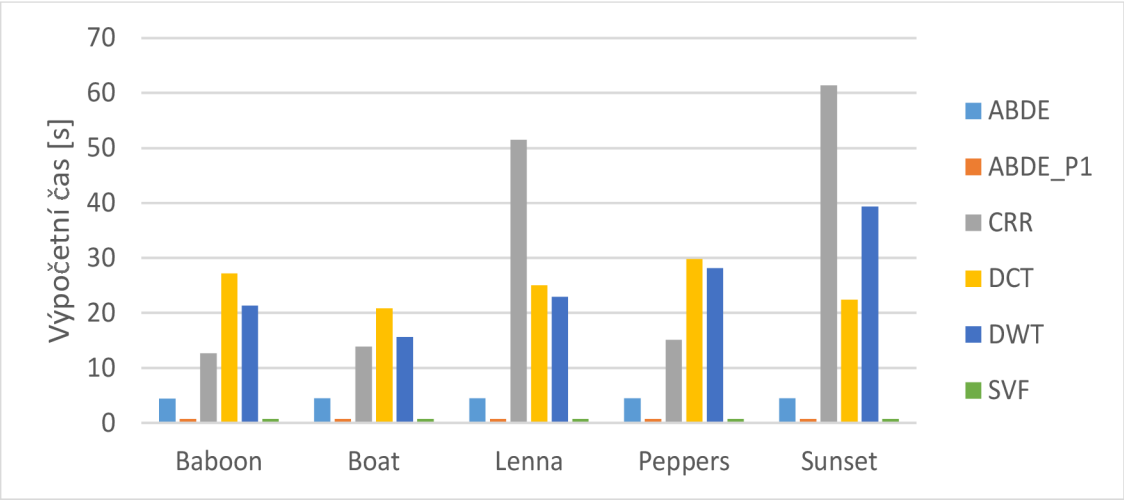


Obr. 6.8: Výsledky subjektivního hodnocení pro dekvantizaci ze 4 bpp na 8 bpp

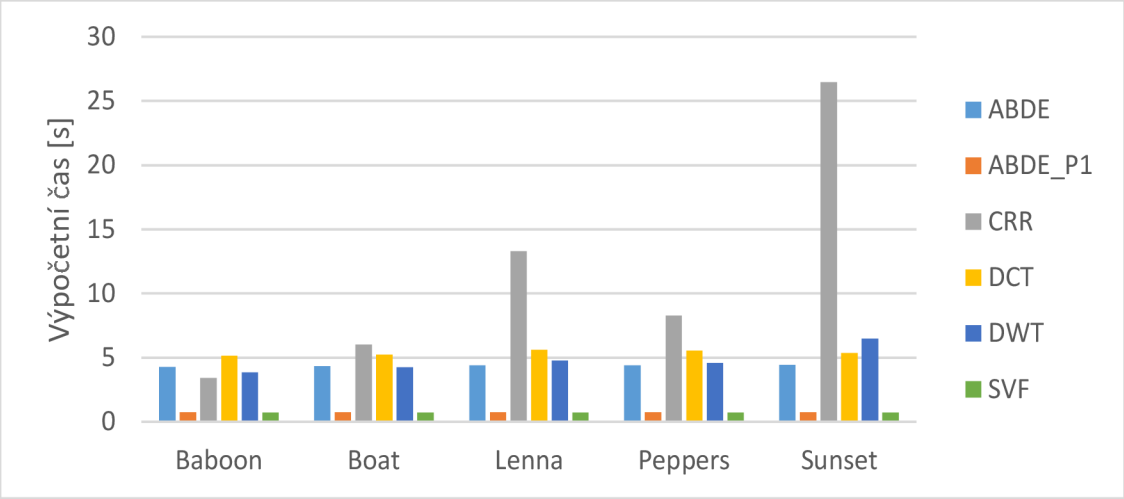
6.3 Srovnání časové náročnosti

Posledním faktorem porovnání jednotlivých metod je srovnání jejich výpočetní náročnosti. Výpočty byly prováděny v prostředí Matlab ve verzi 2014b na PC s OS Windows 7 Professional, CPU Intel Core i5-4690K o taktu 3,5 GHz a 8 GB RAM DDR3 1600 MHz.

Testy výpočetní náročnosti byly provedeny pro 2bitové i 4bitové obrázky. Z grafů se dá odvodit, že metody ABDE, ABDE_P1 i SVF nezávisí na míře degradace obrázku a pro všechny varianty jsou prakticky stejně časově náročné. Jako nejnáročnější se ukázala metoda CRR, jejíž náročnost závisí především na počtu iterací zaplavovacího algoritmu.



Obr. 6.9: Výpočetní časová náročnost metod pro dekvantizaci ze 2 bpp na 8 bpp



Obr. 6.10: Výpočetní časová náročnost metod pro dekvantizaci ze 4 bpp na 8 bpp

7 ZÁVĚR

Tato bakalářská práce objasňuje problematiku nízké bitové hloubky a popisuje celkem jedenáct metod řešících tento problém. Jednotlivé metody jsou vždy stručně představeny, matematicky je popsán jejich algoritmus a na závěr je uvedeno závěrečné zhodnocení funkčnosti metody.

Bakalářská práce dále popisuje problematiku řídké reprezentace signálů a její aplikaci na úlohu dekvantizace. Ve třetí kapitole byly objasněny základní pojmy a značení, dále byl představen aditivní model signálu a podmínky nutné k nalezení řídkého řešení. V kapitole Teoretické řešení bylo detailně rozebráno řešení dekvantizační úlohy a byly představeny dva přístupy, kterými lze tuto úlohu řešit – pomocí lineárního programování a pomocí proximálního dělení.

Pro implementaci algoritmů bylo zvoleno prostředí Matlab. Při programování metody pro zvýšení bitové hloubky na základě řídké reprezentace jsme se nejprve zaměřili na 1D signál, který jsme s definovanou řídkostí vygenerovali, nakvantovali a následně dekvantovali. V tomto bodě proběhlo srovnání přístupu pomocí lineárního programování a pomocí proximálního dělení. Oba algoritmy podávaly dobré výsledky, velkým problémem ale byla vysoká časová i hardwarová náročnost, která by neumožňovala aplikaci takovýchto metod na reálné obrazy. Provedli jsme několik pokusů s rozdělením obrazu na bloky, které byly jednotlivě dekvantovány a následně spojeny ve výsledný obraz. Při rozdělení však logicky vzniká blokový artefakt, který působí velmi rušivě. Vysoká výpočetní náročnost byla vyřešena až s upravenou projekcí pro Douglas-Rachfordův algoritmus, která neprobíhala v oblasti souřadnic, ale v oblasti signálu. Obdobně jako pro 1D signál jsme postupovali i v případě 2D signálu a tuto metodu jsme aplikovali i pro reálné obrazy.

Kromě výše popsaných metod bylo naprogramováno i sedm standardních metod popsaných ve druhé kapitole a se čtyřmi z nich pak byly metody srovnány. Pro porovnání metod na základě řídké reprezentace byla použita jak báze DCT, tak i DWT, konkrétně wavelet db5 a pro dané rozlišení obrázku maximální hloubka dekompozice, tedy 8. Objektivní srovnávání probíhalo pro pět testovacích obrázků degradovaných na 2, 4 a 6 bpp pomocí ukazatelů PSNR a SSIM. Podle těchto ukazatelů metody Sparse_DCT ani Sparse_DWT nedosahovaly kvalit metody Contour Region Reconstruction. Subjektivní hodnocení však pro tyto metody dopadlo příznivěji. Drobným překvapením byl fakt, že metoda využívající DCT bázi i přes vysokou úroveň šumu ve výsledném obrázku dopadla v hodnocení lépe, než metoda využívající DWT.

Ač výsledky této bakalářské práce podávají relativně uspokojivé výsledky, dalo by se v této práci dál pokračovat a uvedené metody ještě vylepšit, např. nahrazením použitých bází framy, které nejsou tak omezené a podle teoretických předpokladů slibují přesnější rekonstrukci.

LITERATURA

[1] AKYÜZ, A., O.; FLEMING, R.; RIECKE, B., E.; REINHARD, E,; BÜLTHOFF H., H. *Do HDR Displays Support LDR Content? A Psychophysical Evaluation* [online]. 2007, [cit. 25. 11. 2014]. Dostupné z URL: <http://www.sfu.ca/~ber1/web/conferencePapers/AkyuzFlemingRieckeReinhardBulthoff_07_SiggraphPaper__Do_HDR_Displays_Support_LDR_Content_-_A_Psychophysical_Evaluation.pdf>.

[2] AU, Oscar C.; FU, Ming Sun; WONG, Peter H. W.; WONG Justy W. C.; GUO, Zihua. Hybrid Inverse Halftoning using Adaptive Filtering. *Circuits and Systems* [online]. 1999. s 259–262. Print ISBN: 0-7803-5471-0. Dostupné z URL: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=779991&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F6311%2F16886%2F00779991.pdf%3Farnumber%3D779991>.

[3] *Bilateral filter*, Wikipedia [online]. 2014, poslední aktualizace 26. 8. 2014 [cit. 11. 12. 2014]. Dostupné z URL: <http://en.wikipedia.org/wiki/Bilateral_filter>.

[4] *Bit Depth Tutorial*, Cambridge in color [online]. © 2014, [cit. 6. 11. 2014]. Dostupné z URL: <http://www.cambridgeincolour.com/tutorials/bit-depth.htm>.

[5] *Bitmapová grafika*, Adaptic [online]. © 2005–2014 [cit. 6. 11. 2014]. Dostupné z URL: <http://www.adaptic.cz/znalosti/slovnicek/bitmapova-grafika/>.

[6] *Co je fotografie ve formátu HDR?*, Nikon [online]. 2009, poslední aktualizace 3. 11. 2014 [cit. 22. 11. 2014]. Dostupné z URL: <https://nikoneurope-cz.custhelp.com/app/answers/detail/a_id/27522>.

[7] *Color Depth*, Wikipedia [online]. 2014, poslední aktualizace 1. 11. 2014 [cit. 6. 11. 2014]. Dostupné z URL: <http://en.wikipedia.org/wiki/Color_depth>.

[8] COMBETTES, L.P.; PESQUET, J-C.: *Proximal splitting methods in signal processing*. Fixed-Point Algorithms for Inverse Problems in Science and Engineering, 2011: 28 s. Dostupné z URL: <http://arxiv.org/abs/0912.3522v4<.

[9] *Digitální obraz*, Wikipedia [online]. 2013, poslední aktualizace 20. 3. 2013 [cit. 6. 11. 2014]. Dostupné z URL: <http://cs.wikipedia.org/wiki/Digit%C3%A1ln%C3%AD_obraz>.

[10] *Dither*, Wikipedia [online]. 2014, poslední aktualizace 10. 11. 2014 [cit. 23. 11. 2014]. Dostupné z URL: <http://en.wikipedia.org/wiki/Dither>.

[11] *High Dynamic Range*, Wikipedia [online]. 2014, poslední aktualizace 27. 8. 2014 [cit. 22. 11. 2014]. Dostupné z URL: <http://cs.wikipedia.org/wiki/High_Dynamic_Range>.

[12] HRBÁČEK, R.; RAJMIC, P.; VESELÝ, V.; ŠPIŘÍK, J. *Rídké reprezentace signálu: komprimované snímání* [online]. Elektrorevue, 2011. ISSN 1213-1539. Dostupné z URL: <http://elektrorevue.cz/cz/download/ridke-reprezentace-signalu--komprimovane-snimani/>.

[13] HRBÁČEK, R.; RAJMIC, P.; VESELÝ, V.; ŠPIŘÍK, J. *Řídká reprezentace signálů: úvod do problematiky* [online]. Elektrorevue, 2011. ISSN 1213-1539. Dostupné z URL: <http://www.elektrorevue.cz/cz/download/ridke-reprezentace-signalu--uvod-do-problematiky/.e.cz/>.

[14] HUBER, Richard. *File: München, St. Benedikt vom Pfarrhof (HDR vs NORMAL).jpg* [online]. 2008, poslední aktualizace 16. 2. 2012 [cit 22. 11. 2014]. Dostupné z URL: <http://commons.wikimedia.org/wiki/File:München,_St._Benedikt_vom_Pfarrhof_(HDR_vs_NORMAL).jpg>.

[15] CHENG, Cheuk-Hong; AU, Oscar C.; CHEUNG, Ngai-Man; LIU, Chun-Hung; YIP, Ka-Yue. Low Color Bit-depth Image Enhancement by Contour-Region Dithering. *Communications, Computers and Signal Processing* [online]. 2009. s 666–670. E-ISBN: 978-1-4244-4561-5. Dostupné z URL: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5291290&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5291290>.

[16] CHENG, Cheuk-Hong; AU, Oscar C.; LIU, Chun-Hung; YIP, Ka-Yue. Bit-depth Expansion By Contour Region Reconstruction. *Circuits and Systems* [online]. 2009. s 944–947. E-ISBN: 978-1-4244-3828-0. Dostupné z URL: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5117913&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5117913>.

[17] *Linear Programming*, Wikipedia [online]. 2015, poslední aktualizace 12. 4. 2015 [cit. 19. 4. 2015]. Dostupné z URL: <http://en.wikipedia.org/wiki/Linear_programming>.

[18] LIU, Chun Hung; AU, Oscar C.; WONG Peter H. W.; KUNG, M. C.; CHAO, Shen Chang. Bit-Depth Expansion By Adaptive Filter. *Circuits and Systems* [online]. 2008. s 496–499. E-ISBN: 978-1-4244-1684-4. Dostupné z URL: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4541463&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4541463>.

[19] MITTAL, Gaurav; JAKHETIYA, Vinit; JAISWAL, Sunil Prasad; AU, Oscar C.; TIWARI, Anil Kumar; WEI, Dai. Bit-Depth Expansion Using Minimum Risk Based Classification. *Visual Communications and Image Processing (VCIP)* [online]. 2012. s 1-5. E-ISBN: 978-1-4673-4406-7. Dostupné z URL: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6410837&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6410837>.

[20] NAVRÁTILOVÁ, Barbora: *Aplikace řídkých reprezentací dat: diplomová práce.* Brno, FSI VUT v Brně, 2014. 62 stran, [cit. 12. 4. 2015].

[21] NETUŠIL, Kamil. *Subjektivní a objektivní hodnocení kvality obrazu a videa* [online]. 2011 [cit. 10. 11. 2014]. Bakalářská práce. University of West Bohemia, Fakulty of Electrical Engineering. Vedoucí práce Ivo Veřtát. Dostupné z URL: <http://theses.cz/id/wuibb5>

[22] PELC, Lukáš. *Metody ditheringu obrazu: diplomová práce.* Brno, FEKT VUT v Brně, 2014. 75 stran, [cit. 6. 11. 2014].

[23] PIHAN, Roman. *Barevná Hloubka (Bit Depth)*, Fotoroman [online]. © 2011, [cit. 6. 11. 2014]. Dostupné z URL: <http://www.fotoroman.cz/glossary2/3_barevna_hloubka.htm>.

[24] *PSNR*, Wikipedia [online]. 2014, poslední aktualizace 16. 1. 2014 [cit. 10. 11. 2014]. Dostupné z URL: <http://cs.wikipedia.org/wiki/PSNR>.

[25] QIAO, Motong; WANG, Wei; NG, Michael Kwok-Po. A Variational Method for Expanding the Bit-Depth of Low Contrast Image. *Energy minimization methods in computer vision and pattern recognition: 9th International Conference, EMMCVPR 2013, Lund, Sweden, August 19-21, 2013. Proceedings.* 1st edition. 2013, s. 54–65. ISBN: 3642403948.

[26] *Quadratic programming*, Wikipedia [online]. 2015, poslední aktualizace 20. 3. 2015 [cit. 1. 5. 2015]. Dostupné z URL: <http://en.wikipedia.org/wiki/Quadratic_programming>.

[27] RAJMIC, Pavel. *Základy počítačové sazby a grafiky*, první vydání. 2012, 161 s. ISBN: 978-80-214-4451-5.

[28] STEIBAUER, Miloslav: *Digitalizace, číslicové zpracování a rekonstrukce signálu.* V publikaci: *Měření v elektrotechnice: skripta k předmětu* . Brno, 2010. 210 stran, [cit. 1. 5. 2015].

[29] TAGUCHI, Akira; NISHIYAMA, Johji. Bit-Length Expansion By Inverse Quantization Process. *Signal Processing Conference (EUSIPCO)* [online]. 2012. s 1543–1547. ISSN: 2219-5491. Dostpné z URL: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6333954&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6333954>.

[30] TALÁR, Ondřej: *Metody pořízení a zpracování obrazů založené na řídkých reprezentacích: bakalářská práce.* Brno, FEKT VUT v Brně, 2014. 40 stran, [cit. 19. 4. 2015].

[31] ULICHNEY, R.; CHEUNG, S. *Pixel Bit-Depth Increase by Bit Replication* [online]. 1998, [cit. 27. 11. 2014]. Dostupné z URL: <http://home.comcast.net/~ulichney/CV/papers/1998-bit-replication.pdf>.

[32] *Vektorová grafika*, Adaptic [online]. © 2005–2014 [cit. 3. 11. 2014]. Dostupné z URL: <http://www.adaptic.cz/znalosti/slovnicek/vektorova-grafika/>.

[33] WAN, Pengfei; AU, Oscar C.; TANG, Ketan; GUO, Yuanfang; FANG, Lu. From 2D Extrapolation to 1D Interpolation: Content Adaptive Bit-Depth Expansion. *Multimedia and Expo (ICME)* [online]. 2012. s 170–175. ISSN: 1945-7871. Dostupné z URL: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6298393>.

[34] WANG, Z.; BOVIK, A. C.; SHEIKH, H. R.; SIMONCELLI, E. P. *Image Quality Assessment: From Error Visibility to Structural Similarity* [online]. 2004 [cit. 13. 11. 2014]. Dostupné z URL: <http://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf>.

[35] ŽÁRA, Jiří; BENEŠ, Bedřich; SOCHOR, Jiří; FELKER, Petr. [cit. 3. 11. 2014]. *Moderní počítačová grafika.* Vyd 2. Brno: Computer Press, 2004, 609 s. ISBN: 80-251-0454-0.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ABDE	Adaptive Bit-Depth Expansion – adaptivní zvýšení bitové hloubky
ABDE-P1	Adaptive Bit-Depth Expansion Part 1 – první část metody ABDE
bpc	bits per color/channel – počet bitů na barvu resp. kanál
bpp	bits per pixel – počet bitů na pixel
BR	Bit replication – replikace bitů
CGA	Color Graphics Adapter
CRD	Contour-Region Dithering – dithering oblastí kontur
CRR	Contour-Region Reconstruction – rekonstrukce oblastí kontur
DCT	Diskrétní kosinová transformace
dpcm	Dots per centimeter – počet bodů na centimetr
dpi	Dots per inch – počet bodů na palec
DWT	Diskrétní waveletová (vlnková) transformace
EGA	Enhanced Graphics Adapter
GE	Gamma expansion – gama expanze
HBDI	High Bit-Depth Image – obraz s vysokou bitovou hloubkou
HDR	High Dynamic Range – vysoký dynamický rozsah
LBDI	Low Bit-Depth Image – obraz s nízkou bitovou hloubkou
LDR	Low Dynamic Range – nízký dynamický rozsah
MIG	Multiplication by ideal gain – vynásobení ideálním zesílením
MSE	Mean Square Error – střední kvadratická chyba
NSP	Null Space Property – vlastnost nulového prostoru
ppi	Pixels per inch – počet pixelů na palec
PSNR	Peak Signal-to-Noise Ratio – špičkový odstup signálu od šumu
RIP	Restricted Isometry Property – vlastnost zeslabené izometrie

SSIM	Structural Similarity – index vyjadřující podobnost dvou obrazů
SVF	Spatial Varying Filter – prostorově proměnný filtr
SVGA	Super Video Graphics Array
VGA	Video Graphics Array
XGA	Extended Graphics Array
ZP	Zero-padding – metoda vyplnění nulami

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

Na přiloženém CD lze najít zdrojové kódy naprogramovaných metod pro zvýšení bitové hloubky a zdrojové kódy k řešení úlohy dekvantizace signálu na základě řídké reprezentace signálů pro 1D signál, 2D signál i jako funkce pro dekvantizaci reálného obrazu. Všechny zdrojové kódy byly vytvořeny v prostředí Matlab ve verzi 2014b. Zpětná kompatibilita s nižšími verzemi software Matlab není zajištěna. Dále přiložené CD obsahuje testovací obrázky pro jednotlivé metody a tabulky hodnot ke grafům z kapitoly 6. Pro přehlednost je obsah CD rozdělen do následujících adresářů.

A.1 Metody

V tomto adresáři se nacházejí zdrojové kódy k vybraným naprogramovaným metodám pro zvýšení bitové hloubky z kapitoly 2.

1. `ABDE.m` – Adaptive-Bit Depth Expansion (viz část 2.7)
2. `ABDE_P1.m` – Adaptive-Bit Depth Expansion Part 1 (viz část 2.7)
3. `BR.m` – Bit Replication (viz část 2.4)
4. `CRR.m` – Contour Region Reconstruction (viz část 2.8)
5. `GE.m` – Gamma Expansion (viz část 2.5)
6. `LPF.m` – MIG v kombinaci s filtrem typu dolní propust
7. `MIG.m` – Multiplication by an ideal gain (viz část 2.3)
8. `SVF.m` – Spatial Varying Filter (viz část 2.6)
9. `ZP.m` – Zero Padding (viz část 2.2)

A.2 Sparse

Adresář obsahující zdrojové kódy k dekvantizačním metodám na základě řídké reprezentace signálů.

A.2.1 1D

Obsahuje zdrojové kódy k dekvantizační úloze pro 1D signál popsané v části 5.1.

1. `dekvantizace_DRA_signal.m` – funkce pro dekvantizaci podle Douhlas-Rachfordova algoritmu s projekcí v oblasti signálu
2. `dekvantizace_DRA_souradnice.m` – funkce pro dekvantizaci podle Douhlas-Rachfordova algoritmu s projekcí v oblasti souřadnic
3. `dekvantizace_LP.m` – funkce pro dekvantizaci pomocí lineárního programování

4. `demo_dekvantizace.m` – funkce demonstrující výsledky dekvantizace
5. `demo_generator.m` – funkce demonstrující výsledky generace signálu
6. `demo_kvantizace.m` – funkce demonstrující výsledky kvantizace signálu
7. `dwtmatrix.m` – funkce autorů Zdeněk Průša a Pavel Rajmic pro výpočet transformační matice waveletové transformace
8. `fix_dekvantizace.m` – funkce pro detekci případu, kdy rekonstruovaný signál leží přesně na rozhodovací hladině, a její následnou opravu
9. `generator.m` – funkce pro generování náhodného řídkého signálu
10. `kvantizace.m` – funkce pro nakvantování signálu
11. `main_1D.m` – hlavní dávkový soubor balíku 1D
12. `projekce_signal_dct.m` – funkce pro projekci na přípustnou množinu v oblasti signálu pro bázi DCT
13. `projekce_signal_dwt.m` – funkce pro projekci na přípustnou množinu v oblasti signálu pro bázi DWT
14. `dekvantizace_souradnice.m` – funkce pro projekci na přípustnou množinu v oblasti souřadnic pro bázi DWT
15. `slovnik.m` – funkce pro generování slovníku

A.2.2 2D

Obsahuje zdrojové kódy k dekvantizační úloze pro 2D signál popsané v části 5.2.

1. `dekvantizace_2D.m` – funkce pro dekvantizaci 2D signálu podle Douhlas-Rachfordova algoritmu s projekcí v oblasti signálu
2. `demo_dekvantizace_2D.m` – funkce demonstrující výsledky dekvantizace 2D signálu
3. `demo_generator_2D.m` – funkce demonstrující výsledky generace 2D signálu
4. `dwtmatrix.m` – funkce autorů Zdeněk Průša a Pavel Rajmic pro výpočet transformační matice waveletové transformace
5. `fix_dekvantizace_2D.m` – funkce pro detekci případu, kdy rekonstruovaný 2D signál leží přesně na rozhodovací hladině, a její následnou opravu
6. `generator_2D.m` – funkce pro generování náhodného řídkého 2D signálu
7. `kvantizace_2D.m` – funkce pro nakvantování 2D signálu
8. `main_2D.m` – hlavní dávkový soubor balíku 2D
9. `projekce_2D_dct.m` – funkce pro projekci 2D signálu na přípustnou množinu v oblasti signálu pro bázi DCT
10. `projekce_2D_dwt.m` – funkce pro projekci 2D signálu na přípustnou množinu v oblasti signálu pro bázi DWT
11. `slovnik.m` – funkce pro generování slovníku

A.2.3 Real picture

Obsahuje funkce určené pro zvýšení bitové hloubky reálných obrazů (viz část 5.3).

1. `Sparse_DCT.m` – funkce pro zvýšení bitové hloubky na základě řídké reprezentace signálu v DCT bázi
2. `Sparse_DWT.m` – funkce pro zvýšení bitové hloubky na základě řídké reprezentace signálu v DWT bázi

A.3 Testing

Tento adresář obsahuje pomocné funkce, které byly vytvořeny pro snažší a rychlejší získání výsledků metod, např. hodnot PSNR, SSIM nebo měření časové náročnosti jednotlivých metod.

1. `importfile.m` – funkce generovaná Matlabem pro automatický import obrázku při zachování jména obrázku jako názvu proměnné
2. `PSNR_SSIM.m` – funkce pro výpočet hodnot PSNR a SSIM pro všechny testované obrázky
3. `Time_test.m` – funkce pro výpočet časové náročnosti jednotlivých metod

A.4 Testing_images

Tento adresář obsahuje všechny testované obrázky ve formátu *.png. Nacházejí se zde originální obrázky, obrázky degradované na 2, 4 a 6 bitů a také obrázky dekvantované jednotlivými metodami.

A.5 Výsledky

Adresář výsledky obsahuje několik pdf souborů s tabulkami výsledků hodnocení jednotlivých metod.

1. `PSNR.pdf` – výsledky hodnocení PSNR ukazatele
2. `SSIM.pdf` – výsledky hodnocení SSIM ukazatele
3. `Subjektivní srovnání.pdf` – výsledky hodnocení dotazníku, obsahuje průměr a medián všech platných odpovědí
4. `Výpočetní náročnost.pdf` – výsledky hodnocení výpočetní časové náročnosti metod