VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA


# BEZEZTRÁTOVÁ KOMPRESE PLENOPTICKÝCH FOTOGRAFIÍ

LOSSLESS LIGHT FIELD COMPRESSION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS


AUTOR PRÁCE — ROBERT NAVRÁTIL
AUTHOR

VEDOUCÍ PRÁCE — Ing. DAVID BAŘINA, Ph.D.
SUPERVISOR



BRNO 2019

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

# Zadání bakalářské práce

21823

Student: **Navrátil Robert**
Program: Informační technologie
Název: **Bezeztrátová komprese plenoptických fotografií**
**Lossless Light Field Compression**
Kategorie: Zpracování obrazu
Zadání:

1. Seznamte se s technologií plenoptických fotografií a formáty pro uložení těchto dat.
2. Seznamte se s technikami, které se používají k bezeztrátové kompresi obrazu, videa a volumetrických dat.
3. Vytvořte kompresní postup pro bezeztrátovou kompresi plenoptických obrazů. S tímto postupem experimentujte. Výstupem tohoto bodu je programová knihovna.
4. Srovnejte účinnost vytvořené metody s existujícími formáty. Kritériem je kompresní poměr a rychlost komprese/dekomprese.

Literatura:

- Touradj Ebrahimi, Siegfried Foessel, Fernando Pereira, Peter Schelkens, JPEG Pleno: Toward an Efficient Representation of Visual Reality, IEEE Multimedia, Oct-Dec 2016.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2 podrobně doložené technickou zprávou. Funkční kompresní řetězec k bodu 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Bařina David, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 6. listopadu 2018

## Abstrakt

Tato práce se věnuje bezeztrátové kompresi plenoptických fotografií, u kterých by mohlo být vhodné využít rozšíření běžných prediktorů do více rozměrů. První kapitola práce se zabývá potřebnou teorií ohledně plenoptických fotografií a s nimi spojeným 4D světelným polem. Dále následuje popis jednotlivých prediktorů a kodérů, se kterými je svázán popis použitého kompresního řetězce. Poslední kapitoly obsahují popis řešení a rozdíly výsledků jednotlivých typů prediktorů a jejich dimenzionálních variant. Nakonec jsou výsledky srovnány s PNG. Ukázalo se, že některé prediktory rozšířené do 4D jsou schopny mít lepší kompresní poměr než PNG.



## Abstract

This thesis is focused on lossless light-field images compression, where it could be viable to extend common predictors into more dimensions. First chapter is regarding to essential theory about light-field images and 4D light-field, which is followed by description of every predictor and coder. With this is connected description of used compression pipeline. Last chapters contains description of solution and differences between every predictor result and theirs dimensional variant. At the end the results are compared with PNG. The result is that some 4D variants of implemented predictors have better compression rate than PNG.



## Klíčová slova

bezeztrátová komprese, predikce, plenoptická fotografie, entropický kodér, barevný model, slovníkové metody, deflate

## Keywords

lossless compression, prediction, plenoptic photography, entropy coder, color model, dictionary methods, deflate


## Citace

NAVRÁTIL, Robert. *Bezeztrátová komprese plenoptických fotografií.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. David Bařina, Ph.D.

# Bezeztrátová komprese plenoptických fotografií

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doktora Davida Bařiny, a také že jsem uvedl všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Robert Navrátil
13. května 2019

## Poděkování

Děkuji vedoucímu této práce panu Ing. Davidu Bařinovi, Ph.D za pomoc, odborné rady a čas.

# Obsah

**1 Úvod** **2**

**2 Plenoptické fotografie** **3**
2.1 Světelné pole . . . . . 3
2.2 Plenoptický fotoaparát . . . . . 4

**3 Bezeztrátová komprese** **7**
3.1 Barevné modely . . . . . 7
3.2 Prediktory . . . . . 9
3.3 Kodéry . . . . . 12

**4 Implementace** **16**
4.1 Informace o knihovně . . . . . 16
4.2 Struktura knihovny . . . . . 16
4.3 Dimenze komprese . . . . . 17
4.4 Kodéry . . . . . 22

**5 Testování** **24**
5.1 Datasety . . . . . 24
5.2 Výsledky testování . . . . . 25

**6 Závěr** **31**

**Literatura** **32**

**A Obsah na DVD** **34**

# Kapitola 1

# Úvod

V dnešní době se velmi často setkáváme s digitálním obsahem jak ve formě fotek, tak videí. S rostoucím množstvím, ale také rostou problémy spojené s uchováním těchto dat.

Fotografie je určitým typem záznamu takzvaného světelného pole. To znamená, že zachytí světlo, které se v daný čas vyskytuje na scéně. Běžný fotoaparát zachytí scénu pouze z jednoho úhlu, takže je nejjednodušším záznamem světelného pole. Jedná se tedy o zachycení 2D světelného pole. Existuje ale i způsob, který je schopný zaznamenat více ze scény. Takovým fotografiím se říká plenoptické a jsou schopny zachytit objekt z více úhlů v jednom čase. Tyto snímky jsou jen mírně posunuté a jsou uloženy v matici fotografií. Problémem je, že velikost fotografie roste s počtem pohledů v matici snímků a může být velká jako stovky klasických fotek. Nutnost zmenšení je tedy nezbytná.

Samotnému zmenšování se říká komprese. Vzhledem k tomu, že lidské oko není schopné rozeznat tak širokou škálu barev jako zaznamená senzor fotoaparátu, je pro normální data běžné využívat kompresní metody, které spojí, či zanedbají, určité prvky. Díky tomu může být zmenšení co největší, avšak to nemusí být vždy přínosné. Používání kompresních metod, které jsou nevratné, není vhodné pro další práci s těmito daty. Pokud tedy v průběhu komprese nejsou ztracena žádná data, tak se bavíme o bezeztrátové kompresi a ta je hlavní částí této práce.

První dvě kapitoly se věnují teorii týkající se světelného pole a principu snímání běžného fotoaparátu oproti plenoptickému fotoaparátu. Dále také popisu bezeztrátové komprese a jejích komponent, což jsou barevné modely, které provádí dekorelaci barevných složek, prediktory, který snižují prostorovou redundanci a kodéry.

Zbytek práce je složen z popisu implementace kompresní knihovny, jejíž hlavní částí je rozšíření běžných prediktorů do více rozměrů a testování těchto prediktorů. Samotné testování potom určí, zda je vhodné pokoušet se o rozšíření prediktorů k dosažení vyšší komprese na plenoptických fotografiích.

# Kapitola 2

# Plenoptické fotografie

Jak bylo řečeno v úvodu, plenoptická fotografie je vlastně maticí mírně posunutých fotografií. Posun jednotlivých snímků je získán díky skutečnosti, že fotoaparát zachycuje i úhel pod kterým světlo dopadlo. V obrázku je tedy uložena informace o světelném poli. Abychom mohli blíže porozumět, jak je fotografie vytvořena, bude velmi stručně vysvětleno, co je to světelné pole a následně bude okrajově popsán princip plenoptického fotoaparátu. Poslední částí kapitoly je popis existujících formátů pro uložení plenoptické fotografie.

## 2.1 Světelné pole

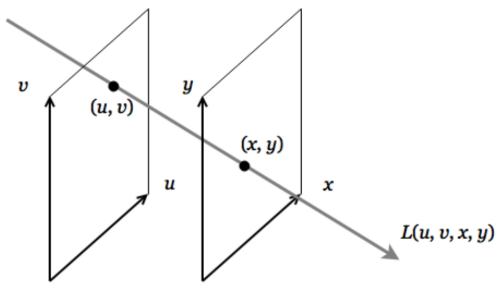
Světelné pole je definováno všemi světelnými paprsky (resp. množstvím světla), které putují ve všech možných směrech, v každém bodu prostoru[12]. Samotné pole je definováno 7-dimenzionální plenoptickou funkcí:

$$L(x, y, z, \theta, \phi, \lambda, t) \tag{2.1}$$

kde $x, y, z$ jsou prostorové souřadnice, $\theta, \phi$ určují všechny možné úhly, $\lambda$ je vlnová délka a $t$ je čas. Dohromady definují polohu paprsku a jeho intenzitu v daný čas v prostoru. Samotná funkce jde zjednodušit a k vyjádření postačí pouze 4-dimenzionální funkce světelného pole (*light-field function* nebo *lumigraph function*[5]). Ta je definována průsečíky paprsku skrze dvě roviny:

$$L(x, y, u, v) \tag{2.2}$$

kde $uv$ je v tomto případě pozice vstupu paprsku do fotoaparátu a $xy$ je pozice dopadu paprsku na senzor. Průběh jednoho paprsku je zobrazen na obrázku 2.1



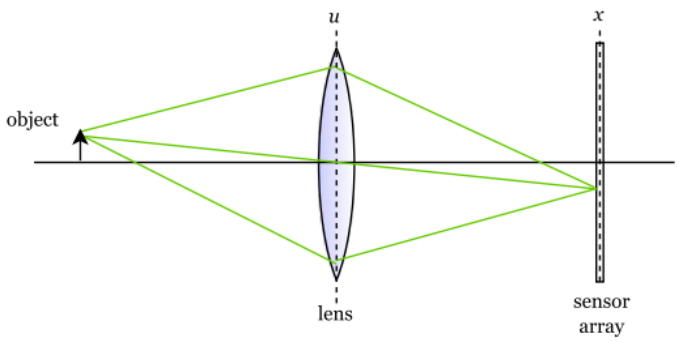

Obrázek 2.1: Zobrazení paprsku pomocí funkce světelného pole

## 2.2 Plenoptický fotoaparát

Za plenoptický fotoaparát je považováno zařízení, které vyfotí objekt z více úhlů ve stejném čase. Jednou z možností je soustava kamer (např. *Stanford multi-camera array*[11]). Tato varianta je velmi nákladná a závisí na přesné synchronizaci všech použitých kamer, avšak každý pohled má rozlišení jako každá z jednotlivých kamer. Nabízí se také použití klasického fotoaparátu, který je upevněn v konstrukci, která zajistí jeho postupný pohyb k získání matice snímků. Zde je potřeba zaručit stejné světelné podmínky v čase focení, aby byly rozdíly mezi jednotlivými snímky co nejmenší. Tímto způsobem se podle možností posuvného mechanismu dá přiblížit snímku z Lytro fotoaparátu, který bude následně popsán.

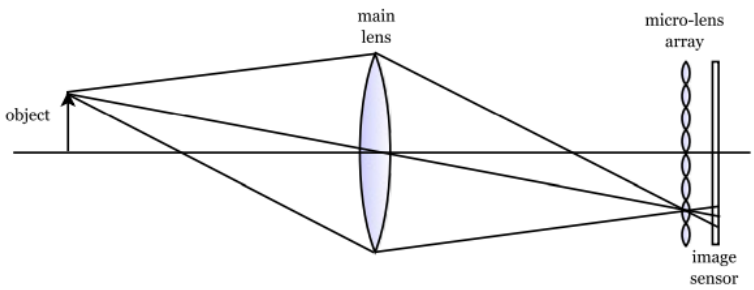
K pochopení funkce plenoptického fotoaparátu, či kamery, je nejprve nutné vysvětlit princip klasického snímání.

Na obrázku 2.2 je znázorněno k jaké ztrátě informace dochází při běžném focení. Je zřejmé, že informace o tom, jaký směr paprsek měl před vstupem do objektivu, je ztracena. Energie paprsků jsou tedy sečteny v dané buňce (dutině) senzoru a dříve zmíněná plenoptická funkce (2.1) je redukována pouze na 2 dimenze, resp. 2D pole energií v jednotlivých buňkách.



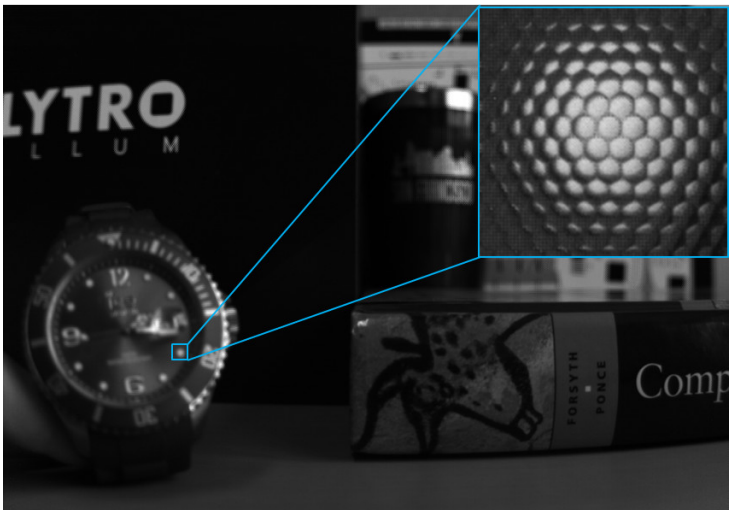

Obrázek 2.2: Snímání běžného fotoaparátu

Klasický fotoaparát má tedy čočku, která soustředí světelné paprsky na senzor. Naproti tomu je plenoptický fotoaparát, který je schopen rozlišit paprsky, které jsou u běžného fotoaparátu sečteny a soustředěny na jedno místo. Toho dosáhne díky poli mikročoček, které jsou mezi senzorem a klasickou čočkou. Chování jednotlivých paprsků uvnitř plenoptického fotoaparátu můžete vidět na obrázku 2.3.




Obrázek 2.3: Snímání plenoptického fotoaparátu

Je patrné, že jeden bod objektu už se nyní nezobrazí pouze na jeden bod senzoru. Následkem je tedy udržení informace o úhlu, pod kterým daný paprsek přišel[10]. Jestliže tedy každá mikročočka pokrývá plochu třeba $15 \times 15$ pixelů (buněk senzoru), tak uchová informace o 225 různých posunech. Je také zřejmé, že plochy pod mikročočkami zachycují pouze část celého obrázku. Výsledná fotografie v matici je tedy složena z bodů na stejných pozicích v jednotlivých blocích, které se nacházejí pod mikročočkami. Pokud bychom se podívali na fotografii v plném rozlišení (nebo spíše `raw` podobě), tak zjistíme, že obrázek je složen z malých bodů v podobě mikročoček, jak můžete vidět na obrázku 2.4.




Obrázek 2.4: Složení plenoptické fotografie

Ve zvětšené části jde vidět, jak jednotlivé mikročočky představují rozdílnou pozici. Pokud je například z každé plochy pod mikročočkou vybrán levý vrchní pixel, tak dostáváme fotografii s menším rozlišením, ale poloha je mírně posunutá doleva nahoru[14]. Tento princip jde použít pro každý pixel z 15x15 rozsahu, který byl zmíněn dříve a je tedy možné dosáhnout 225 pohledů, z čehož je vytvořený prostorový pohled typický pro plenoptické fotografie. Tato úvaha zanedbává okraje čoček, takže uvažujme, že čočky jsou ve tvaru čtverce. Tento typ zobrazení by byl největším možným úhlovým rozlišením, které je však na úkor kvality. Další možností je - pro plenoptické fotografie typická - změna zaostření po pořízení fotografie. K tomuto zobrazení je třeba určit novou přední ohniskovou rovinu a na základě úhlů vypočítat, které pixely jednotlivých mikročoček jsou součástí výsledné fotografie. Na obrázku 2.5 je zobrazena funkce přeostřování.

Obrázek 2.5: Roviny ostrosti

Ukládání a komprese plenoptických dat se liší na základě technologie pořízení. Pokud bychom mluvili o Stanfordském přístupu, tak jsou jednotlivé fotky uloženy zvlášť v libovolném dostupném formátu (PNG, JPEG, ...).

U fotoaparátu Lytro je přístup jiný. Vyfocená surová data (`raw`) jsou uložena v lfr formátu (light field raw). Součástí dat je nastavení fotoaparátu, typ senzoru a další technické věci spojené s daným fotoaparátem. Hlavní části jsou data ze senzoru, tedy samotná data fotografie, kde každý pixel zabírá 10 bitů. Výsledný obrázek je třeba ještě 'demozaikovat', aby byl barevný[6][9].

Aktuálně není definovaný standard, který by určoval nějaký způsob bezeztrátové komprese plenoptických fotografií. Jediný chystan je od společnosti JPEG a to standard JPEG Pleno.

# Kapitola 3

# Bezeztrátová komprese

Tato kapitola se věnuje bezeztrátové kompresi jako celku. Budou zde přiblíženy používané barevné modely, které je možno realizovat na počítači. Dále také jednotlivé typy nejpoužívanějších prediktorů a nakonec popis aritmetického kodéru. Kombinace zmíněných částí tvoří kompresní řetězec.

## 3.1 Barevné modely

„*Barevný model je matematický model, který popisuje barvu*“[4]. Nejběžnějším modelem je klasické RGB, které je složeno ze 3 barevných složek - červená, zelená, modrá. Každá složka má vymezený rozsah a číslo z tohoto rozsahu udává jas (nebo podíl) dané složky ve výsledné barvě. Rozsah všech tří komponent dohromady určuje barevnou hloubku pixelu (nebo jen hloubku). Při klasické reprezentaci RGB, kde každá složka má rozsah 1 byte, je možné zobrazit téměř 16,8 milionů barev.I když je RGB žádoucí k zobrazování dat na displejích, protože jejich pixely jsou složeny z těchto 3 složek, tak pro účel přenosu a komprimace dat se využívají modely závislé na jasu. Jedním z důvodů je vzájemná závislost těchto složek. Dále také samotné lidské oko je mnohem více náchylné na změny v jasu, než na barevnou složku jako takovou, čehož využívá mnoho ztrátových kompresních metod.

### 3.1.1 YUV

Tento model je tedy složen ze 2 barvonosných složek (U, V nebo C_B, C_R) a jedné jasové (Y). Modely závislé na jasu byly potřebné pro kompatibilitu s šedotónovým (nepřesně černobílým) vysíláním, kde ke složce jasu přidali také 2 chromatické složky. Běžně je model YUV definován analogově a převod z RGB je následující:

$$\begin{pmatrix} Y \\ C_B \\ C_R \end{pmatrix} = \begin{pmatrix} R \\ G \\ B \end{pmatrix} \cdot \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}1687 & -0{,}3313 & 0{,}5 \\ 0{,}5 & -0{,}4187 & -0{,}0813 \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} \tag{3.1}$$

a ještě opačným směrem:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} Y \\ C_B \\ C_R \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1{,}402 \\ 1 & -0{,}34414 & -0{,}71414 \\ 1 & 1{,}772 & 0 \end{pmatrix} + \begin{pmatrix} -179{,}456 \\ 135{,}45984 \\ -226{,}816 \end{pmatrix} \tag{3.2}$$

U rovnice 3.1, při výpočtu jasové složky (Y), je vidět citlivost lidského oka na jednotlivé barvy (vlnové délky). Oko tedy nejvíce vnímá zelenou barvu, následuje červená a poslední je modrá.

Jak už bylo poznamenáno, tak je tento převod analogový a pracuje tedy s reálnými čísly, což není vhodné pro počítačovou reprezentaci. Sám o sobě není YUV na počítači reverzibilní, takže je nutné tento algoritmus upravit.

### 3.1.2 LYUV

Jednou z reverzibilních metod převodu barevného modelu je LYUV. Jedná se o modifikaci YUV modelu, která zajišťuje plnou reverzibilitu.

$$\begin{aligned} Y_L &= G + \text{round}\left(\frac{0{,}299}{0{,}587} \cdot R + \frac{0{,}114}{0{,}587} \cdot B\right) \\ U_L &= B - \text{round}(0{,}587 \cdot Y_L) \\ V_L &= R - \text{round}(0{,}587 \cdot Y_L) \end{aligned} \tag{3.3}$$

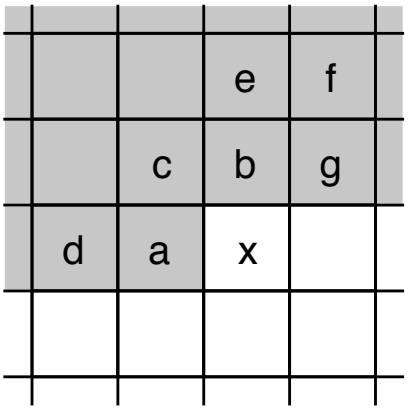
A ještě zpětný převod:

$$\begin{aligned} R &= V_L + \text{round}(0{,}587 \cdot Y_L) \\ G &= Y_L - \text{round}\left(\frac{0{,}299}{0{,}587} \cdot R + \frac{0{,}114}{0{,}587} \cdot B\right) \\ B &= U_L + \text{round}(0{,}587 \cdot Y_L) \end{aligned} \tag{3.4}$$

Samotné zaokrouhlování (round) zde znamená klasické zaokrouhlení na nejbližší celé číslo.

## 3.2 Prediktory

Princip prediktorů zakládá na skutečnosti, že okolní pixely jsou silně korelovány, takže je lze nějakým způsobem odhadnout z jejich okolí. Prediktor tedy předpoví hodnotu, které zpracovávaný pixel $x$ nabývá (každá složka se predikuje zvlášť) a následně tuto hodnotu odečte od opravdové hodnoty pixelu. Uložené hodnotě se říká „chyba predikce“ (pokud je 0, tak byla predikce úspěšná) a počet hodnot použitých k predikci určuje řád prediktoru. Tímto se snaží soustředit obor hodnot pixelů okolo nuly, aby následné kodéry mohly efektivněji komprimovat. Podstatou je, že prediktor musí vždy pracovat s pixely, které jsou při dekompresi známé i dekódující straně (už jsou rozkódovány), takže záleží na pořadí zpracování. Vizualizace predikce je na obrázku 3.1.




Obrázek 3.1: Ukázka predikce pixelu $x$ z okolních pixelů. Šedé pixely jsou známy dekódující straně a proto mohou být použity k predikci.

Práce s plenoptickými fotografiemi však rozšiřuje běžnou funkci prediktorů. Nyní nemusí pracovat pouze s jedním obrázkem a s tím spojeným odhadem z okolních hodnot, ale je možné použít další dimenze. Pokud by tedy klasický obrázek představoval 2D pole hodnot, tak řetězec obrázků (video) představuje možnost prediktor rozšířit do třetí dimenze, díky hodnotám z předchozího snímku. Samotné plenoptické fotografie jsou ale víc než video, protože jsou částečně schopny zachytit prostor a výsledná fotografie je matice fotografií, kde na každé je objekt zachycen z trochu jiného úhlu, jak je popsáno v kapitole 2.

Z této úvahy vychází možnost rozšíření prediktorů i do čtvrté dimenze, takže hodnota pixelu bude odhadována jak z jeho okolí, tak z různých příbuzných pixelů v okolních obrázcích. Jinými slovy řečeno budou pixely vybírány pomocí 4 souřadnic $P(u, v, x, y)$ kde $u$,$v$ jsou souřadnice obrázku v matici obrázků a $x$,$y$ jsou souřadnice pixelu v obrázku.

Přístup vůči rozdílným typům plenoptických dat se však může lišit. V sekci o plenoptickém fotoparátu 2.2 jsem zmínil Stanfordský přístup se synchronizovaným polem kamer. Rozdíl vzdálenosti jednotlivých snímků je mnohem větší, než u Lytro fotoaparátu, takže v případě predikce by bylo vhodné zjistit daný posun, který je stejný ve všech obrázcích a s ním pracovat. Možná řešení jsou blíže specifikovaná v kapitole 4.

Velmi častou operací je u prediktorů proložení rovinou, které běžně značí, že v okolí predikovaného pixelu nebyla nalezena žádná hrana. Pro další použití je proložení rovinou značeno $p$ a je rovno $p = a + b - c$. Prediktory lze rozdělit do dvou skupin a to jsou lineární a nelineární.

Lineární prediktor použije k odhadu hodnotu, která je výstupem lineární funkce nad pixely, které jsou součástí predikce. Takovou funkcí může být například aritmetický průměr, nebo již zmíněné proložení rovinou $p$. Nejjednodušším lineárním prediktorem je primitivní kódování rozdílů, což je vlastně pouze rozdíl sousedních hodnot. Princip tohoto prediktoru je znázorněn níže:

$$17 \quad 16 \quad 18 \quad 32 \quad 35 \quad 35 \quad 34 \quad 28 \quad 28 \quad \rightarrow \quad -1 \quad +2 \quad +14 \quad +3 \quad 0 \quad -1 \quad -6 \quad 0$$

Rovnice k tomuto prediktoru by tedy byla:

$$\hat{x} = a \tag{3.5}$$

Prediktor tedy pracuje od konce obrázku, přičemž obrázek je serializován po řádcích (první je predikován poslední pixel = vpravo dole).

Nelineární prediktor aplikuje - jak název napovídá - nějakou nelineární funkci na výběr predikční hodnoty. Typickými představiteli jsou například Paeth, MED (*Median Edge Detector*), GAP (*Gradient Adjusted Predictor*) nebo GED (*Gradient Edge Detector*)

### 3.2.1 Paeth

Řadí se mezi složitější prediktory a je použit ve formátu PNG[2]. Paeth je prediktor třetího řádu (jako MED) a používá levý, levý vrchní a vrchní pixel k odhadu hodnoty pixelu $x$. Známými pixely proloží rovinu a odhadne pravděpodobnou hodnotu $x$. Následně hledá nejbližší okolní hodnotu k tomuto odhadu a tu použije k predikci.

$$\hat{x} = \underset{y \in a,b,c}{arg\min} |p - y| \tag{3.6}$$

### 3.2.2 MED

Resp. *Median Edge Detector* je prediktor používaný v JPEG-LS[8]. Rozhodování prediktoru:

$$\hat{x} = \begin{cases} \min(a,b) & : c \geq \max(a,b) \\ \max(a,b) & : c \leq \min(a,b) \\ p & \end{cases} \tag{3.7}$$

Prediktor nejprve zjistí, zda se okolo predikovaného pixelu nenachází vertikální, nebo horizontální hrana (princip je vysvětlen v závislosti na obrázku 3.1). Pokud ano, tak je pravděpodobně pixel $x$ už za hranou, tedy například pokud je $c \geq \max(a,b)$ a maximum by tedy bylo $a$, tak se s velkou pravděpodobností jedná o vertikální hranu $(c,a)$. Vzhledem k tomu, že $b$ bylo rozdílné, tak právě jeho hodnota je predikovanou hodnotou $x$. Podobným způsobem je možné vysvětlit i druhou část této funkce. Pokud však neplatí ani jedna z těchto variant, tak se nejspíš jedná o hladké okolí a postačí tedy $x$ predikovat jako bod roviny, která je proložena ostatními body ($p = a + b - c$).

Daná funkce se dá také vyjádřit jako medián těchto hodnot, a proto název *Median Edge Detector*:

$$\hat{x} = \text{median}(a, b, a + b - c) \tag{3.8}$$

### 3.2.3 GAP

*Gradient Adjusted Predictor* je další z řady nelineárních prediktorů. Podobně jako MED predikuje pixel $x$ na základě hran v okolí. Na rozdíl od něj ale rozpozná 3 úrovně hran a díky tomu je schopný lépe odhadnout hodnotu pixelu. Vertikální i horizontální hrany tedy rozpoznává ve 3 stupních (ostrá, normální a slabá)[3].
K detekci hran používá lokální odhad gradientu:

$$\begin{aligned} g_v &= |a-c| + |b-e| + |g-f| \\ g_h &= |a-d| + |b-c| + |b-g| \end{aligned} \tag{3.9}$$

a 3 heuristické prahy (8, 32, 80), které definují, o jak silnou hranu se jedná. Problémem tohoto prediktoru je však vysoká složitost procesu predikce. Pokud bychom rozhodování rozdělili do dvou rovnic, tak by část pro ostré hrany byla:

$$\hat{x} = \begin{cases} a & \text{pro } (g_v - g_h) > 80 \\ b & \text{pro } (g_v - g_h) < -80 \end{cases} \tag{3.10}$$

pokud by neplatila ani jedna z těchto variant, tak je vypočítaná váhovaná hodnota z okolních pixelů:

$$P = \frac{a+b}{2} + \frac{g-c}{4} \tag{3.11}$$

a rozhodování bude následující:

$$\hat{x} = \begin{cases} \frac{P+a}{2} & \text{pro } (g_v - g_h) > 32 \\ \frac{3P+a}{4} & \text{pro } (g_v - g_h) > 8 \\ \frac{P+b}{2} & \text{pro } (g_v - g_h) < -32 \\ \frac{3P+b}{4} & \text{pro } (g_v - g_h) < -8 \\ P & \text{jinak} \end{cases} \tag{3.12}$$

### 3.2.4 GED

Podobně jako předešlý GAP používá i GED lokální odhad gradientu k detekci hran, avšak k odhadu používá pouze 5 okolních pixelů na rozdíl od 7, které jsou potřeba u prediktoru GAP.

Samotný prediktor rozpoznává pouze 1 typ vertikální a horizontální hrany, protože práh je nastavitelný a je vhodné ho volit v závislosti na povaze zpracovávaného obrázku. Pokud není práh fixně určený, tak je ale potřeba ho při použití prediktoru uschovat do komprimovaného souboru. Lokální odhad gradientu je tedy zjednodušen pouze na:

$$\begin{aligned} g_v &= |c-a| + |e-b| \\ g_h &= |d-a| + |c-b| \end{aligned} \tag{3.13}$$

a rozhodování tohoto prediktoru by bylo následující:

$$\hat{x} = \begin{cases} a & \text{pro } (g_v - g_h) > T \\ b & \text{pro } (g_v - g_h) < -T \\ \frac{3(a+b)}{8} + \frac{d+c+e}{12} & \text{jinak} \end{cases} \tag{3.14}$$

kde $g_v$ a $g_h$ jsou lokální odhady gradientu z předchozí rovnice (3.13) a $T$ je volitelný práh. GED je tedy kompromisem mezi MED a GAP.

## 3.3 Kodéry

Poslední částí jednoduchého kompresního řetězce jsou kodéry. Ty můžeme rozdělit na entropické a slovníkové. Entropický kodér přiřazuje nejčastěji vyskytujícímu se znaku nejkratší kód. Naproti tomu slovníkové metody mají uložené sekvence po sobě jdoucích znaků a celá sekvence je nahrazena jedním symbolem, takže při kódování se hledá nejdelší shoda se sekvencemi ve slovníku.

### 3.3.1 Entropické kodéry

Entropické kódování prochází vstupní řetězec a příchozím symbolům přiřazuje kód, přičemž délka přiřazeného kódu může být i delší než délka samotného symbolu. To však neznamená, že nedojde ke kompresi, protože často se vyskytující symboly mají kratší kódy.

#### Aritmetický kodér

Jednou z možností entropického kódování je aritmetický kodér, jehož kompresní poměr se velmi blíží heuristické funkci udávající největší možnou kompresi s daným výskytem znaků. Tudíž tedy nezáleží na pravděpodobnostních výskytech jednotlivých symbolů (například Huffmanovo kódování totiž dosahuje nejlepších výsledků pro pravděpodobnosti, které jsou rovny záporné mocnině 2).

Samotné kódování je založené na jednoduchém principu. Výsledný kód je jedno dlouhé desetinné číslo, které je vytvořené zužováním intervalu $\langle 0; 1)$ pomocí pravděpodobnosti výskytu příchozích symbolů. Pokud je zanedbána adaptivní varianta, která si tabulku výskytů upravuje postupně, tak je proveden jeden průchod souborem a do tabulky se uloží jednotlivé počty symbolů. Tato tabulka se jako první zapíše do výstupního souboru. Ke každému znaku je vypočítána pravděpodobnost výskytu:

$$p(x) = \frac{\text{výskyt symbolu}}{\text{počet symbolů}} \tag{3.15}$$

Aritmetický kodér pracuje tedy s intervalem $\langle 0; 1)$. Tento interval je následně rozdělen podle četnosti jednotlivých znaků.

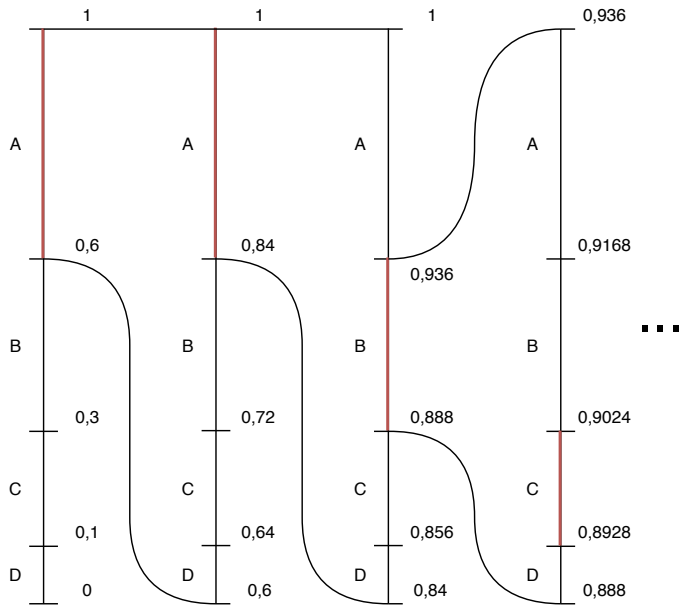
U jednoduchého příkladu, kde je vstupní řetězec `AABCBADACB` jsou výskyty:

| Symbol | Počet výskytů | Pravděpodobnost výskytu |
|---|---|---|
| A | 4 | 40% |
| B | 3 | 30% |
| C | 2 | 20% |
| D | 1 | 10% |

Tabulka 3.1: Tabulka aritmetického kodéru

Výstupní intervaly tedy budou $A \rightarrow \langle 0{,}6; 1), B \rightarrow \langle 0{,}3; 0{,}6), C \rightarrow \langle 0{,}1; 0{,}3), D \rightarrow \langle 0; 0{,}1)$. Tyto intervaly následně slouží k postupnému kódování vstupního řetězce pomocí zužování intervalu $\langle 0; 1)$.

Z daného řetězce `AABCBADACB` by bylo na vstupu nejprve `A`, což by mělo za následek zužení intervalu $\langle 0; 1)$ na $\langle 0{,}6; 1)$. Dalším znakem je znovu `A` a interval by byl zúžen na $\langle 0{,}84; 1)$. Tento postup je znázorněn na následujícím obrázku.




Obrázek 3.2: Ukázka práce aritmetického kodéru

Pokud by kódování probíhalo tímto způsobem, tak by řetězec v zakódované formě bylo číslo z intervalu $\langle 0{,}8974844928; 0{,}8974872576)$, takže jakékoliv číslo z daného intervalu přesně definuje daný řetězec `AABCBADACB`.

Dekódování probíhá opačným způsobem. Nejdříve je rekonstruována tabulka výskytů. Následně je načten uložený vstup, což je libovolné číslo z předchozího intervalu, takže například spodní hranice 0,8974844928. Potom dekodér zjistí, do kterého intervalu z tabulky aktuální hodnota náleží. Hodnota spadá do intervalu symbolu `A` $\rightarrow \langle 0{,}6; 1)$. Je tedy vygenerován znak `A`. Potom je od vstupní hodnoty odečtena dolní hranice intervalu `A` (0,6) a hodnota je podělena délkou intervalu ($\frac{0{,}8974844928-0{,}6}{0{,}4}$). Takto je vstupní hodnota postupně dekódována.

| Hodnota | Symbol | Interval | Nová hodnota |
|---|---|---|---|
| 0,8974872576 | A | ⟨0,6; 1) | $\frac{0{,}8974872576-0{,}6}{0{,}4}$ |
| 0,743711232 | A | ⟨0,6; 1) | $\frac{0{,}743711232-0{,}6}{0{,}4}$ |
| 0,35927808 | B | ⟨0,3; 0,6) | $\frac{0{,}35927808-0{,}3}{0{,}3}$ |
| 0,1975936 | C | ⟨0,1; 0,3) | $\frac{0{,}1975936-0{,}1}{0{,}2}$ |
| 0,487968 | B | ⟨0,3; 0,6) | $\frac{0{,}487968-0{,}3}{0{,}3}$ |
| 0,62656 | A | ⟨0,6; 1) | $\frac{0{,}62656-0{,}6}{0{,}4}$ |
| 0,0664 | D | ⟨0; 0,1) | $\frac{0{,}0664-0}{0{,}1}$ |
| 0,664 | A | ⟨0,6; 1) | $\frac{0{,}664-0{,}6}{0{,}4}$ |
| 0,16 | C | ⟨0,1; 0,3) | $\frac{0{,}16-0{,}1}{0{,}2}$ |
| 0,3 | B | ⟨0,3; 0,6) | - |

Tabulka 3.2: Tabulka aritmetického dekodéru

Pokud vybereme spodní hranici, tak je zřejmé, že výsledná hodnota bude spodní hranice intervalu posledního dekódovaného symbolu (v tomto případě `B` → 0,3). Toto rozložení platí pro celý interval a z toho důvodu je také interval z vrchu uzavřený, protože jinak by vyšlo 0,6.

Problémem však je, že aritmetický kodér pracuje s reálnými čísly a pro funkčnost na počítači je potřeba vše převést do celočíselné formy, čehož jde docílit pomocí postupného upravování výstupu během kódování. Dochází tedy ke škálování a posunům[7](snímek 20 a dál).

### 3.3.2 Slovníkové metody

Tento způsob využívá ke kódování slovník. Ten může být reprezentovaný mnoha způsoby (trie, tabulka nebo posuvné okno, které obsahuje vždy část vstupu). Hlavním rozdílem od entropického kódování je snaha zakódovat více po sobě jdoucích symbolů jako jednu značku místo snížení velikosti jednotlivých vstupních symbolů.

#### Deflate

Tato kódovací metoda je spojením slovníkové metody LZ77 a entropického Huffmanova kódování. Je použita ve formátech ZIP, PDF, PNG, aj. Výstup se skládá z literálů (symbolů), offsetů (vzdáleností shody) a délek (délka shody). Tyto 3 prvky jsou uloženy ve dvou tabulkách/stromech a kódovány Huffmanovým kódováním. Jedna tabulka uchovává literály/délky a druhá offset. Pro literály je vyhrazeno místo pro každý znak $(0-255)$, následuje EOB symbol, který značí konec bloku, a $257-285$ je vyhrazeno pro délky shody. Deflate pracuje ve 3 základních stupních komprese.

- první možností je uložit data nekomprimovaná, což se vyplatí pokud chceme velký soubor rozdělit do menších bloků, avšak maximální nekomprimovaný blok, který Deflate umožňuje uložit je $2^{15}-1$B $(65\,535\,B)$
- další možností je komprese s fixními tabulkami, takže není třeba tabulky komprimovat a ukládat do výsledného souboru. Tato metoda je vhodná, pokud komprimujeme

známá data (textový soubor v angličtině), pro která jsou vytvořené tabulky výskytů s vcelku přesnými odhady výskytů

- poslední možností je úplná komprese, která projde výskyty jednotlivých symbolů v souboru a uloží si jejich četnosti. U této metody je potřeba uložit také tabulky do výstupního toku

Zkomprimovaný soubor je vnitřně uložen po blocích, přičemž každý blok obsahuje vlastní tabulky. Záleží už na samotném kodéru, kdy uzná za vhodné blok ukončit a znovu obnovit tabulky.

Deflate tedy prochází vstupní data a 32 kB posledních dat si drží v paměti. Nejprve se pokusí najít shodu s nějakým již dříve nalezeným řetězcem, a zde záleží na stupni komprese, jak daleko ve vyhledávacím bufferu bude hledat. Pokud je shoda nalezena, tak do výstupu uloží dvojici (délka, offset), která značí jak daleko a jak dlouhá byla shoda od aktuálního symbolu. Offset je omezený již zmíněnou velikostí vyhledávacího bufferu (32 kB) a maximální délka shody je 258. Také je důležité vzít v potaz to, že symboly i délky shod jsou uloženy v jednom stromě/tabulce, takže délka shody je vypsána jako Huffmanův kód. Symboly a délky shody jsou vlastně znaky jedné abecedy, která má 286 symbolů. Pokud není shoda nalezena, tak je vypsán zakódovaný literál pomocí Huffmanova kódu.

```
..string#str#pokus#|ring#st..
```

Pro tento vstupní řetězec, kde `|` značí aktuální pozici kompresoru (posun je prováděn zprava doleva) by shoda vypadala následovně:

```
[délka 7][offset 15]..
```

Délka bude Huffmanův kód v prvním stromě pro shodu 7 znaků a offset Huffmanův kód ze stromu pro offsety. Pokud by se jednalo o délku větší než 32 nebo offset přesahující 30, tak je potřeba přidat pomocné bity, které zařídí rozšíření do specifikovaných velikostí (offset $\rightarrow 1-32\,768$, délka $\rightarrow 3-258$). V případě kompresoru, který si tabulky vytváří sám, mohou být tabulky i menší vzhledem ke skutečnosti, že nějaké znaky se ve vstupním souboru vůbec nemusí vyskytovat.

# Kapitola 4

# Implementace

Cílem práce je tedy implementace knihovny, který zprostředkovává bezeztrátovou kompresi plenoptických fotografií. V této kapitole budou ukázány hlavní funkce a princip samotné knihovny.

## 4.1 Informace o knihovně

Knihovna je implementována v jazyce C++14 a pro vývoj bylo použito vývojové prostředí CLion od JetBrains. Také je nutné podotknout, že knihovna pracuje pouze s obrázky ve formátu `.ppm` a barevným formátem RGB24.

Jak už bylo zmíněno v kapitole 3, tak kompresní řetězec je složen ze 3 částí: změna barevného modelu, predikce a kódování. Typ barevného modelu, prediktoru nebo kodéru ale nemusí být fixní. Knihovna obsahuje více typů barevných modelů(`-m`) i prediktorů(`-p`) a uživateli jsou přístupné přes přepínače v příkazové řádce. Na výběr má také mezi kodérem Deflate a nebo aritmetickým kodérem pomocí přepínače `-c`.

Výstupem kompilace jsou dva spustitelné soubory (`pleno_compress` a `pleno_decompress`) a statická a dynamická knihovna (`plenocompress` a `static_plenocompress`) s rozhraním `plenocompress.h`.

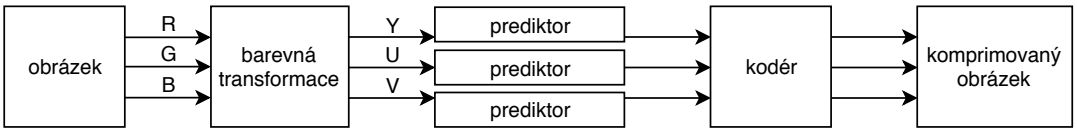
Program, který zprostředkovává kompresi má 3 povinné argumenty (masku vstupních souborů, výstupní soubor a dimenze komprese) a 4 nepovinné (prediktor, kodér, barevný model a posun). Dekompresní program má pouze 2 povinné argumenty (masku výstupních souborů a vstupní soubor).

## 4.2 Struktura knihovny

Jelikož knihovna pracuje ve více dimenzích, tak bude nejprve rozebrána klasická 2D varianta, kde se komprimuje každý obrázek zvlášť. Bude ukázána pouze komprese, protože dekomprese je ten stejný přístup akorát v opačném pořadí.

Knihovna tedy načte příchozí obrázek, zkontroluje hlavičku souboru a následně čte jednotlivé pixely a jejich RGB složky. Jednotlivé pixely jsou serializované zleva doprava po řádcích z vrchu dolů. Obrázky jsou uloženy ve třídě `FileLoad_PPM`, kde jsou pixely uloženy ve vektoru struktur obsahujících RGB hodnoty. Následně jsou jednotlivé barevné složky dekorelovány pomocí barevné transformace. Knihovna obsahuje 2 typy barevných transformací a je na uživateli, kterou použije. Implementované je LOCO-I a modifikace YUV 3.1.2. Výstupem barevné transformace jsou 3 složky a na každou z nich je použit

prediktor. Po průchodu prediktoru je nad daty zavolán kodér a výsledný komprimovaný obrázek následně uložen.




Obrázek 4.1: Obecné schéma kompresního řetězce

V paměti je tedy uložený jeden obrázek, který je postupně modifikován a v daných krocích zmenšován.

Pokud bychom počítali s kompresí videa, tak prediktor pracuje s aktuálním a předchozím obrázkem v jednom čase, protože nějaké části snímků jsou podobné nebo stejné, a tak může dosáhnout lepší predikce. U plenoptické fotografie je možné hodnotu pixelu predikovat z více okolních obrázků, protože zachycuje část prostoru foceného objektu. Z těchto důvodů je při zpracování a kompresi plenoptické fotografie v paměti celá matice obrázků a postupně na povaze prediktoru jsou vybírány příslušné obrázky z matice. O dimenzi komprese tedy rozhoduje to, jaké pixely používá prediktor k odhadu

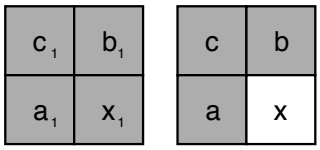
## 4.3 Dimenze komprese

K více možným kompresím zakládajících na dimenzích vedla reprezentace plenoptických fotografií, a to ta skutečnost, že jsou uloženy v matici a mírným posunem. Jsou to tedy „videa“ v řádcích i sloupcích matice. Vzhledem k této skutečnosti je při získávání pixelů z okolních obrázků implicitně vždy posun o jeden pixel. Ke každé dimenzi komprese je v knihovně vytvořená funkce s prefixem `PlenoCompress_` následovaným dimenzí (2D, 3D, 4D) použitého prediktoru.

Klasická 2D varianta byla vysvětlena v předchozí sekci 4.2. Program prochází jednotlivé obrázky v matici, které jsou postupně komprimovány. V knihovně jsou implementovány prediktory, které jsou zmíněny v sekci 3.2. U každého z těchto prediktorů byla snaha o rozšíření do více dimenzí.

### 4.3.1 3D prediktory

Rozšíření prediktorů do třetí dimenze tedy umožňuje při predikci využít i pixely z předchozího obrázku. Snímky pro 3D predikci jsou serializovány po řádcích z vrchu dolů ze vstupní matice obrázků a plenoptický snímek se poté chová jako několik za sebou jdoucích videí. Tento způsob je na začátcích řádků neefektivní, protože pixely, které jsou použité k predikci, jsou z posledního obrázku minulého řádku. Je možné problém zmírnit způsobem serializace matice, ale to by ústilo v další komplikace spojené s implicitním posunem predikovaných pixelů. Součástí implementace 3D predikce je tedy pouze jeden typ serializace matice snímků a to po řádcích.




Obrázek 4.2: Pixely používané k predikci u jednoduchých 3D prediktorů. Šedé pixely jsou známé dekódující straně.

### 3D kódování rozdílů

Tento typ prediktoru byl přítomný hlavně pro testovací účely a srovnání s ostatními prediktory. Od klasické 2D varianty, která odečítá předchozí pixel na řádku od aktuálního je nyní odečítána hodnota pixelu na stejné pozici, ale předchozího obrázku. Tedy pro příklad na obrázku 4.2 je hodnota $x$ po predikci $x - x_1$.

Nabízí se i možnost, kde bude použit průměr $a$ a $x_1$ namísto samotného $x_1$.

### 3D MED

Na rozdíl od ostatních prediktorů je 3D varianta MED prediktoru vytvořená a použitá ve standartu pro 3D JPEG-LS. Tato varianta jen mírně rozšiřuje klasický MED prediktor popsaný v sekci 3.2.2.

Pro demonstraci je zde rozhodování tohoto prediktoru:

$$\hat{x} = \begin{cases} \min(b, a, x_1) & \text{pro } \min(c, b_1, a_1) \geq \max(b, a, x_1) \\ \max(b, a, x_1) & \text{pro } \max(c, b_1, a_1) \leq \min(b, a, x_1) \\ \frac{2(b+a+x_1)}{3} - \frac{c+b_1+a_1}{3} & \text{jinak} \end{cases} \tag{4.1}$$

### 3D Paeth

Paeth, který je sám o sobě popsaný v sekci 3.2.1, je podivné rozšiřovat. Je důležité se rozhodnout které pixely by měly být součástí predikce, protože možností je několik. Implementované jsou 4 varianty, které budou následně testovány.

První možností je zanechat původní proložení rovinou ($p = a + b - c$) a k predikci využít pixely z aktuálního obrázku a navíc k nim $x_1$, tedy $x$ z předchozího obrázku. Toto $x$ by teoreticky mohlo být na skoro stejné pozici jako predikovaný pixel:

$$\hat{x} = \underset{y \in M}{arg\,\min} |p - y|, \qquad M = \{b, a, c, x_1\} \tag{4.2}$$

U této varianty je pravděpodobně nejnižší dopad, pokud by byly obrázky příliš rozdílné.

Jinou možností této varianty by mohlo být rozšíření množiny $M$ o pixely z předešlého obrázku a k množině $M$ by se tedy přidaly pixely $c_1, b_1, a_1$.

Další implementovanou variantou je použití všech dostupných pixelů jako v rovnici (4.2), ale proložení rovinou $p$ bude váhovaným průměrem proložení v aktuálním a předešlém snímku:

$$p = \frac{2(a + b - c) + (a_1 + b_1 - c_1)}{3} \tag{4.3}$$

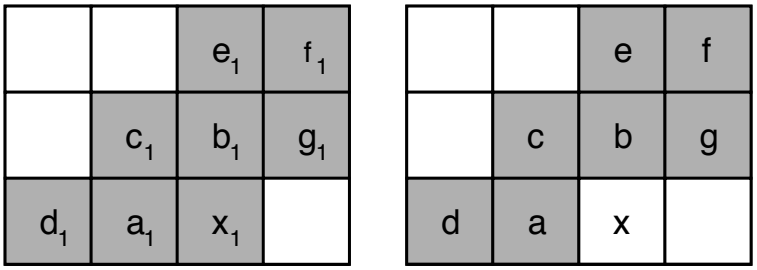
Poslední možností je využít předešlého obrázku jiným způsobem a místo klasického proložení rovinou provést proložení skrze oba obrázky, tedy proložení hyperrovinou:

$$p = a + b - x_1 \tag{4.4}$$

### 3D GAP

I když je GAP podobný prediktoru MED (oba prediktory jsou popsány v sekci 3.2), tak rozšíření do třetí dimenze se liší. Důvodem je už tak vysoká náročnost GAP prediktoru.

První kontrolou je přítomnost ostré hrany v aktuálním snímku. Pokud je přítomna ostrá hrana, tak je vrácena náležející hodnota (pixel $a$ nebo $b$). Když v aktuálním snímku nebyla přítomná ostrá hrana, tak prediktor pokračuje hledáním ostrých hran v předchozím snímku. Pokud předešlý obrázek obsahuje vertikální nebo horizontální ostrou hranu, tak prediktor usoudí, že v závislosti na drobném posunu při plenoptickém snímku, nebude v rámci rozdílu pár pixelů další ostrá hrana. K predikci jsou použity všechny okolní pixely jako u klasického prediktoru GAP, ale s rozšířením o předchozí snímek:




Obrázek 4.3: Pixely používané k predikci GAP 3D

Pokud je v předchozím snímku přítomná ostrá vertikální hrana, tak odhad hodnoty predikovaného pixelu je rozdělen do tří částí:

$$\hat{x} = \begin{cases} \frac{g_1+b+v}{3} & \text{pokud je normální vertikální hrana v aktuálním snímku} \\ \frac{g_1+b+3v}{5} & \text{pokud je slabá vertikální hrana v aktuálním snímku} \\ a & \text{jinak} \end{cases} \tag{4.5}$$

kde $v$ je váhovaná hodnota z rovnice (3.11). Prediktor se snaží zjistit zda hrana, která se nachází v předchozím snímku, není hranou, která byla nalezena v aktuálním snímku a nebyla tedy protažena.

Další možností je přítomnost ostré horizontální hrany v předchozím snímku:

$$\hat{x} = \frac{x_1 + 2a}{3} \qquad \text{pokud je ostrá horizontální hrana v aktuálním snímku} \tag{4.6}$$

Pokud neplatí ani jedna z těchto variant, tak je z důvodu jednoduchosti zbytek predikce proveden pouze pro aktuální snímek, protože existuje mnoho kombinací a variant, jak jednotlivé hodnoty spojit a zda je hrana stále ještě přítomná v právě zpracovávaném obrázku. Algoritmus GAP pro normální nebo slabé hrany, či hladké okolí, je v rovnici (3.12).

### 3D GED

Rozšíření prediktoru GED je vcelku jednoduché a zakládá na stejném principu jako 3D GAP. Pokud je nalezena hrana v předchozím snímku, tak je provedena kontrola zda je hrana i v aktuálním snímku. Pro vertikální hranu to znamená, že je přechod tak dlouhý, že ho lze najít v obou snímcích a je tedy možné použít $x_1$ a $b_1$ jako část predikce.

$$\hat{x} = \begin{cases} \frac{x_1+b_1+b}{3} & \text{pokud aktuální obrázek obsahuje vertikální hranu} \\ p & \text{jinak} \end{cases} \tag{4.7}$$

Pro horizontální je význam pouze to, že hrana pokračuje dál a predikce je podobná jako u 3D GAP (4.6).

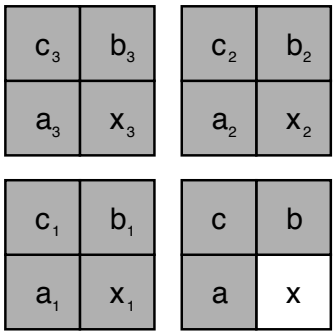
Pokud není v předchozím snímku nalezena hrana, tak je otestován aktuální snímek na přítomnosti hran a v případě nalezení vertikální nebo horizontální hrany proběhne odhad jako pro 2D variantu prediktoru GED.

Poslední možností je hladké okolí v obou snímcích. Pro tuto možnost je odhad následující:

$$\hat{x} = \frac{2(a+b-c)+x_1}{3} \tag{4.8}$$

## 4.3.2 4D prediktory

Rozšíření prediktorů do čtvrté dimenze je poslední logickou variantou pro plenoptická data. Smyslem je tedy nyní použít k predikci část vstupní matice obrázků a pomocí nich provést co nejlepší odhad aktuálního pixelu.




Obrázek 4.4: Pixely používané k predikci u jednoduchých 4D prediktorů. Šedé pixely jsou známé dekódující straně

### 4D kódování rozdílů

Rozšíření tohoto prediktoru do čtvrté dimenze je podobné jako její 3D varianta. Výsledný odhad je vypočítán jako průměr hodnot pixelů na stejné pozici z okolních snímků.

$$\hat{x} = \frac{x_1+x_2+x_3}{3} \tag{4.9}$$

K teoretickému vylepšení výsledku by se do odhadu mohl přidat také pixel $a$ a případně mu také dát vyšší váhu a pixelu $x_3$ dát nejnižší váhu, protože se nachází na vcelku vzdálené pozici.

### 4D MED

U rozšíření MED prediktoru se vycházelo z jeho již vytvořené 3D varianty. Základem jsou pořád 3 větve, ze kterých se vybírá predikovaná hodnota:

$$\hat{x} = \begin{cases} \min(a, b, X) & \text{pro } \min(M) \geq \max(a, b, X) \\ \max(a, b, X) & \text{pro } \max(M) \leq \min(a, b, X) \\ \frac{3(a+b+X)}{3} - \frac{a_1+b_1+X}{3} - \frac{a_2+b_2+X}{3} & \text{jinak} \end{cases} \tag{4.10}$$

kde $M = \{a_1, b_1, a_2, b_2, a_3, b_3, c\}$ a $X$ je průměrem hodnot pixelů $x$ z okolních obrázků.

### 4D Paeth

Už ve 3D variantě v předešlé sekci 4.3.1 byly vysvětleny možnosti, jak Paeth modifikovat, takže rošíření do čtvrté dimenze je vcelku jednoduché. Do množiny pixelů použitých k predikci se přidají pixely ze 3 okolních snímků namísto jednoho předchozího. Stejně jako u 3D varianty je více možností, jak Paeth rozšířit a jaké pixely budou součástí predikce.

Celkem bylo tedy navrženo pět typů prediktoru Paeth pro čtyři dimenze. Všechny verze pracují na stejném základu jako klasický Paeth, a to hledání nejbližší hodnoty k proložení rovinou $p$.

První možností je původní Paeth, jehož množina pixelů, ze kterých predikuje, je rozšířena pouze o pixely na pozicích $x$ z okolních obrázků matice.

Další možná varianta je přidat do této množiny také všechny okolní pixely $(a, b, c)$ z okolních obrázků, přičemž proložení rovinou $p$ je stále stejné a je vypočítáno pouze z aktuálního obrázku ($p = a + b - c$).

Třetí možností je použít průměr proložení rovin z okolních obrázků na stejných pozicích. Je provedeno také váhování, $p$ aktuálního obrázku mělo nejvyšší váhu pro případ, že by byly obrázky až příliš odlišné.

Poslední 2 rozšíření jsou si velmi podobné a jako u 3D varianty v sekci 4.3.1 jde o proložení hyperrovinou. Nyní je rovina proložena přes všechny okolní snímky a nemá nic společného s aktuálním snímkem. Tedy kromě toho, že je vysoce pravděpodobná korelace okolních snímků s aktuálním. Proložení rovinou $p$ bude nyní tedy:

$$p = x_1 + x_2 - x_3 \tag{4.11}$$

Po získání tohoto proložení se poslední varianty liší pouze množinou, která je použita k odhadu. Je možné zahrnout veškeré pixely ze všech okolních snímků i z aktuálního včetně okolních $x$, nebo použít pouze hodnoty, pomocí kterých bylo vytvořeno proložení hyperrovinou, takže $x$ z okolních snímků.

### 4D GAP

Stejně jako u 3D varianty je princip rozšířeného GAP prediktoru detekce hrany v okolních obrázcích. Je vhodné poznamenat, že k odhadu jsou použity pouze 4 snímky matice, jak je ukázáno na začátku kapitoly, takže fakt, že využívá více pixelů k odhadu, zde neovlivňuje počet okolních snímků použitých k odhadu.

Počátek predikce je závislý na tom, zda se v aktuálním obrázku nachází vertikální ostrá hrana a pokud ano, tak je pravděpodobné, že odhad normální verze prediktoru bude

velice přesný. Pokud se v aktuálním snímku nenachází ostrá hrana, tak prediktor zjišťuje přítomnost ostrých hran v okolních snímcích.

Nejprve zjistí zda není ostrá vertikální hrana v levém obrázku. Když je nalezena, tak se rozhoduje na základě toho, jakou hranu nalezl v aktuálním snímku:

$$\hat{x} = \begin{cases} \frac{a_2+b+x_2+v}{4} & \text{pokud je normální vertikální hrana v aktuálním snímku} \\ \frac{g_1+b+x_2+3v}{6} & \text{pokud je slabá vertikální hrana v aktuálním snímku} \end{cases} \tag{4.12}$$

kde $v$ je váhovaná hodnota z rovnice (3.11). Pokud nenalezl vertikální hranu v aktuálním snímku, tak prošetřuje další možnost a tou je ostrá horizontální hrana ve vrchním snímku, která by mohla znamenat delší přechod do aktuálního obrázku:

$$\hat{x} = \begin{cases} \frac{a_2+x_2+2v+2a}{6} & \text{pokud je normální horizontální hrana v aktuálním snímku} \\ \frac{a_2+x_2+3v+a}{6} & \text{pokud je slabá horizontální hrana v aktuálním snímku} \end{cases} \tag{4.13}$$

Posledním pokusem o odhad pomocí okolních snímků je ostrá horizontální hrana v levém snímku, která by mohla zasahovat i do aktuálního, a tím omezit pixely k odhadu. Když je tedy nalezena ostrá horizontální hrana i v aktuálním snímku, tak je odhad stejný jako v 3D variantě.

Pokud nebyla splněna ani jedna z těchto podmínek, tak je proveden normální GAP predikce pro normální a slabé hrany a hladké okolí.

#### 4D GED

Podobně jako u 3D GED prediktoru je rozšíření do čtvrté dimenze velmi podobné prediktoru GAP. Postupně se snaží zjistit zda není vertikální hrana z levého snímku přítomna i v aktuálním, popřípadě zda horizontální hrana z vrchního snímku nezasahuje až do aktuálního. Nakonec zjistí zda není vedena horizontální hrana z levého obrázku do aktuálního a jinak je provedeno proložení rovinou zprůměrováno pixely na pozici $x$ z vrchního a levého snímku.

## 4.4 Kodéry

Součástí implementace je Aritmetický kodér s adaptivní i klasickou variantou a Deflate, které jsou popsány v sekci 3.3. Deflate není součástí implementace, ale je použita knihovna, které jsou předány data. Hlavičkový soubor, který zprostředkovává rozhraní kodéru obsahuje odkaz na Github ke zdrojovým kódům knihovny.

### 4.4.1 Aritmetický kodér

Jak bylo nastíněno v sekci o kodérech 3.3, tak při implementaci aritmetického kodéru je problém s normalizací intervalu, aby samotné kódování, které ukládá vstupní řetězec jako jedno dlouhé reálné číslo, mohlo fungovat. K implementaci aritmetického kodéru se využívají dva přístupy. Jeden využívá krok (`step`) a druhý rozsah (`range`). Implementovaná je varianta využívající rozsah, která dokáže lépe využít hranice daného intervalu, ale při použití 32 bitových čísel je maximální kumulativní frekvence omezena na 16383, což určitým způsobem snižuje kvalitu komprese.

Základem aritmetického kodéru jsou výskyty jednotlivých symbolů (frekvence) a kumulativní frekvence. Ta udává postupný součet jednotlivých výskytů a také intervaly pro jednotlivé symboly. Tyto frekvence a s nimi spojené kumulativní frekvence jsou v tabulce kodéru. Pokud se jedná o adaptivní variantu, tak po zakódování každého symbolu jsou tyto proměnné aktualizovány, jinak je proveden průchod souborem a z těchto proměnných se stávají konstanty po zbytek komprese.

Hlavními proměnnými při kódování jsou `high` a `low`, které určují aktuální pozici v intervalu a na základě kterých se provádí škálování.

Z počátku komprese je `low` nastavený na 0 a `high` na nejvyšší možnou hodnotu. Poté začíná samotný cyklus, který prochází vstupní řetězec a postupně komprimuje příchozí symboly. Nejprve se vypočítá `range = (high - low) + 1` a poté se aktualizují proměnné `high` a `low`:

```
high = low + (range * cumFreq[sym]) / cumFreq[0] - 1
low = low + (range * cumFreq[sym+1]) / cumFreq[0]
```

kde `cumFreq[0]` je součet všech frekvencí symbolů. U `high` je odečtená jedna z důvodu vrchní otevřenosti intervalu.

Jakmile jsou vypočítané hranice, tak je potřeba zjistit, zda není interval ve stavu, kdy by se `low` a `high` po pár cyklech rovnalo a nebylo by tedy možné rozlišit jednotlivé symboly. Z tohoto důvodu existuje E1, E2 a E3 škálování, které bylo zmíněno v podsekci o aritmetickém kodéru. Můžou nastat tedy 3 špatné stavy, ze kterých už se `high` a `low` nedostanou.

E1 škálování nastane když je `high` menší, než polovina maximální hodnoty. V takovém případě je interval zaseknutý ve spodní polovině. Do výstupního bufferu je vložen bit s hodnotou 0 a provede se rozšíření intervalu:

```
low = low * 2
high = high * 2 + 1
```

E2 škálování nastává v opačném případě jako E1, takže pokud `low` překročí polovinu maximální hodnoty. To by znamenalo, že `high` a `low` už navždy zůstanou v horní polovině intervalu. Do výstupního bufferu je tedy vložen bit s hodnotou 1 a od obou proměnných se odečte polovina maximální hodnoty. Následně se provede stejné rozšíření jako u E1 škálování.

Pomocí těchto dvou metod je možné zajistit, že proměnné `high` a `low` nikdy neuvíznou v horní nebo spodní polovině intervalu, ale může nastat ještě jedna možnost. Škálování je také potřebné v případě, kdy jsou obě hodnoty blízko středu intervalu, protože E1 ani E2 škálování by v takovém případě nezajistilo rozšíření hranic intervalu. Když je tedy `low >= FirstQtr and high < ThirdQtr`, tak je potřeba provést E3 škálování, u kterého je od obou proměnných odečtena čtvrtka maximální hodnoty. Problémem je, že v tomto případě není známo, do jaké části intervalu bude hodnota patřit, takže je uloženo pouze to, že bylo provedeno E3 škálování a při dalším E1 a E2 škálování se zjistí, jaký bit má být poslán na výstup.

# Kapitola 5

# Testování

Tato kapitola obsahuje tabulky a grafy dosažených výsledků jednotlivých prediktorů a srovnání vice-dimenzionálního přístupu. Podstatnou částí je hlavně graf výskytu jednotlivých symbolů.

## 5.1 Datasety

Nejdříve je potřeba ukázat nad jakými daty byly pokusy prováděny. Knihovna byla testována nad 2 datasety: fotografie z Lytro fotoaparátu[1] a matice fotografií ze Stanfordské matice kamer[11](odkaz *Light Fields*).

### 5.1.1 Lytro dataset

Z Lytro datasetu byly vybrány 2 fotografie z každé z jednotlivých sekcí (budovy, příroda, atd.). Cělkově je dataset složen z 16 fotografií. Všechny snímky mají stejné rozměry a byly foceny stejným fotoaparátem (*Lytro Illum B01 (10-bit) camera*). Data byla získána ze souborů typu `.mat`, kde byl snímek ve formátu `.LFR` (*light field raw* formát) upraven pomocí programu Matlab a byla vygenerována matice jednotlivých snímků. Tato matice má rozměry $15 \times 15 \times 434 \times 625 \times 4$. Matice je tedy složena z $15 \times 15$ fotek s rozlišením $434 \times 625$ a RGB hodnot [13]. Fotografie musely být upraveny, protože jsou pořízeny s RGB48 a knihovna pracuje pouze s RGB24.

Obrázek 5.1: Část datasetu z fotoaparátu Lytro

### 5.1.2 Stanfordský dataset

Tento dataset se skládá z šesti fotografií, přičemž jednotlivé fotografie mohou mít odlišné rozměry. Snímky byly pořízeny buď přes Stanfordskou matici kamer, a nebo přes "Light Field Gantry" tedy kameru, která putuje na posuvném mechanismu a výslednou matici fotí průběžně.

Obrázek 5.2: Část datasetu z archivu na stránkách univerzity ve Stanfordu

## 5.2 Výsledky testování

Jelikož hlavním výstupem práce bylo pokusit se navrhnout co nejvhodnější rozšíření známých prediktorů do více dimenzí pro bezeztrátovou kompresi plenoptických fotografií, tak jsou hlavním výstupem grafy, které ukazují kvalitu predikce. Konečným výstupem je také kompresní poměr.

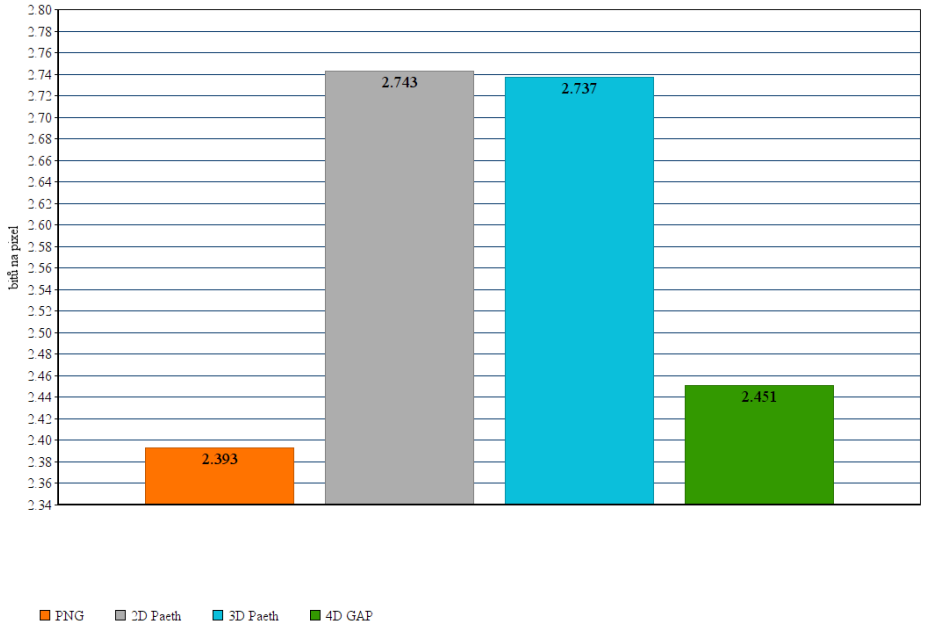
Program byl spouštěn s barevným modelem LYUV 3.1.2 a kodérem Deflate 3.3. Výsledky jsou rozděleny po datasetech, protože snímky z Lytro fotoaparátu a Stafordské matice jsou velice odlišné a výstup by mohl být zkreslený. Také je vhodné zjistit, jaký přístup je vhodnější na jednotlivá data. Výsledky v jednotlivých datasetech byly zprůměrovány přes všechny obrázky datasetu, takže kategorie uvnitř datasetu nejsou brány v potaz.

Výstupem jsou dvě sady grafů. První typ grafu je prostý sloupcový graf, který bude znázorňovat kompresní poměr PNG vůči nejlepšímu 2D, 3D a 4D prediktoru v příslušném datasetu, kde 1 znamená žádná komprese. Druhé jsou histogramy, které budou ukazovat kvalitu predikce, tedy procentuální zastoupení jednotlivých hodnot $-128 - 127$ po průchodu prediktory, přičemž vhodné je aby největší procento připadalo hodnotám okolo nuly (nejlépe nula). Každý nejlepší prediktor ve své skupině (2D, 3D a 4D) bude mít svůj vlastní histogram, přičemž srovnávací je průměrem hodnot ve všech `.ppm` obrázcích daného datasetu.

### 5.2.1 Výsledky – matice kamer / gantry

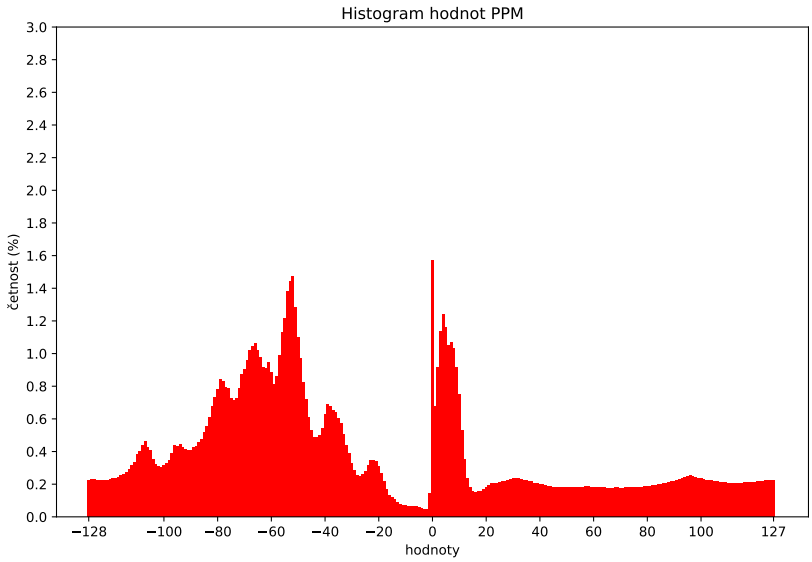
Pro Stanfordský dataset se s implementovanými rozšířeními prediktorů jako nejlepší ukázal klasický 2D přístup, kde dominoval prediktor Paeth. Fotografie byly až příliš rozdílné a výsledný pokus o predikci z více snímků byl díky tomu nepřesný. Ve 3D rozšíření byl nejlepší také Paeth, ale serializace matice po řádcích byla nevhodná pro kompresi prvních obrázků v řádku matice, protože posun snímků byl už příliš velký. Nejlepším 4D prediktorem se u tohoto datasetu ukázal GAP.

Srovnání kompresních poměrů je v následujícím grafu.




Obrázek 5.3: Průměrný kompresní poměr Stanfordského datasetu (více je lépe)

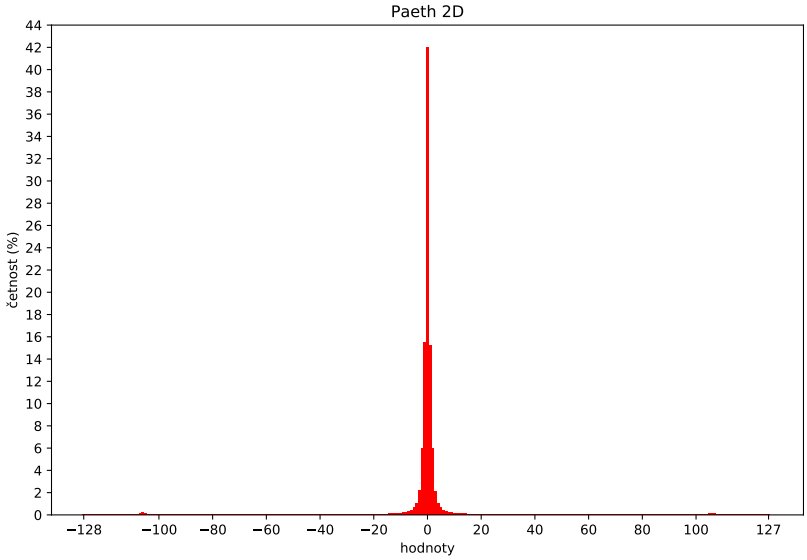
Nyní k samotným histogramům pro Stanfordský dataset. Nejprve histogram hodnot u nezmenšených snímků (matice ve formátu `.ppm`).




Obrázek 5.4: Histogram původních hodnot

A dále histogramy nejlepších prediktorů pro dané rozšíření:

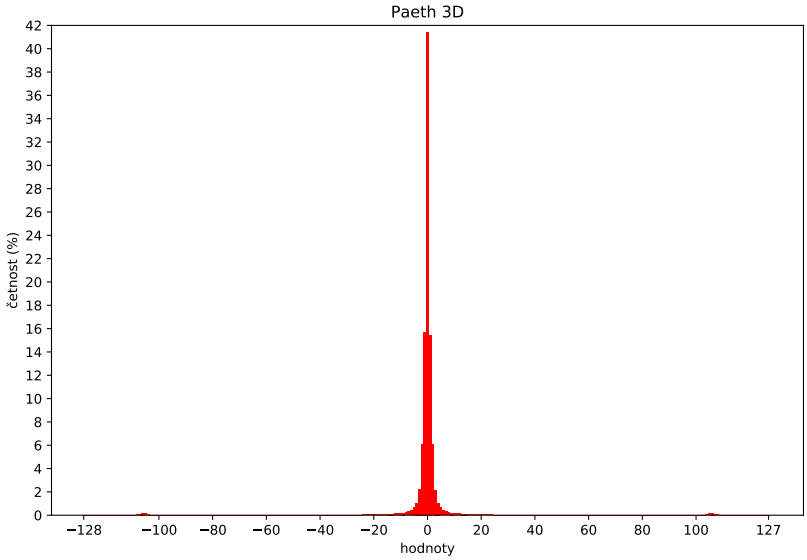
### 2D




Obrázek 5.5: Histogram hodnoty po průchodu Paeth 2D

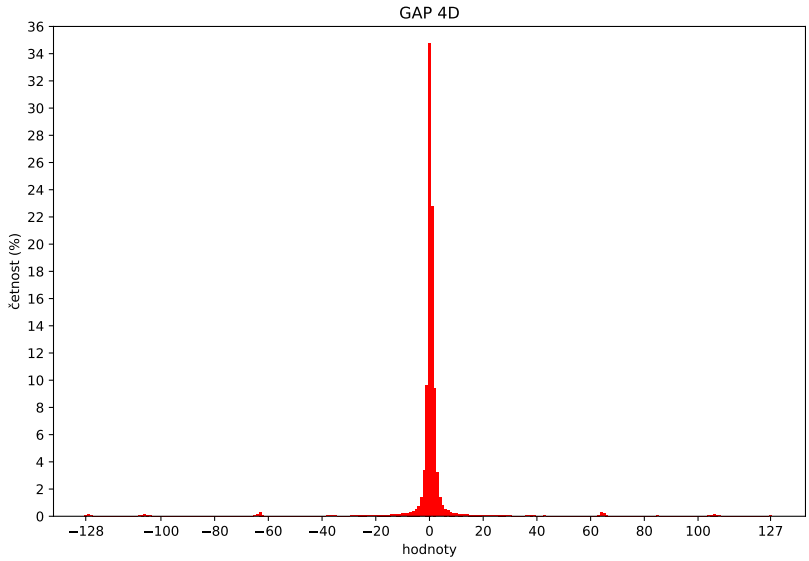
### 3D

Zde je vhodné zmínit, že nejlépe fungovala varianta, která k odhadu přidala pouze $x$ z předchozího snímku.




Obrázek 5.6: Histogram hodnoty po průchodu Paeth 3D

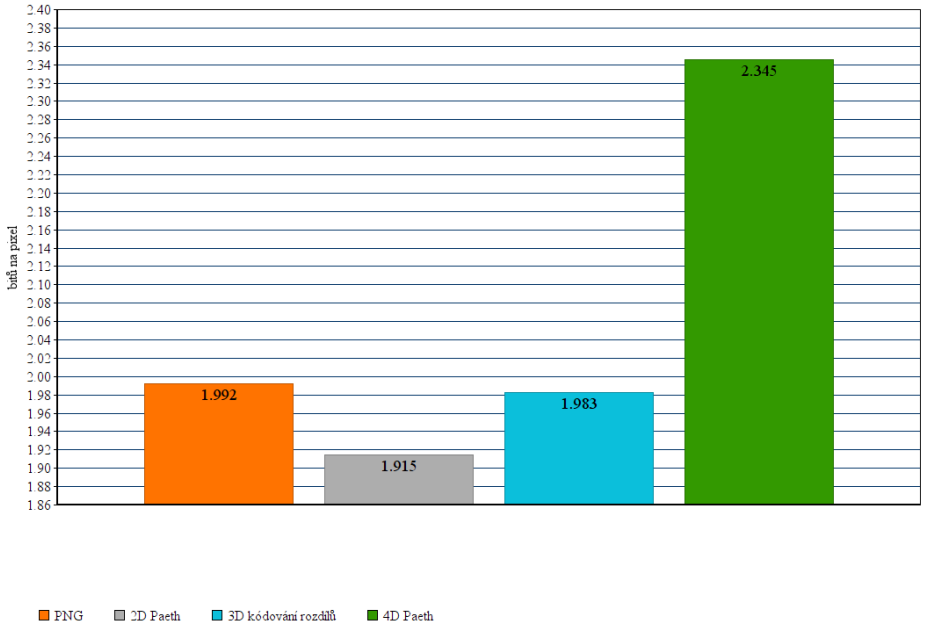
**4D**




Obrázek 5.7: Histogram hodnoty po průchodu GAP 4D

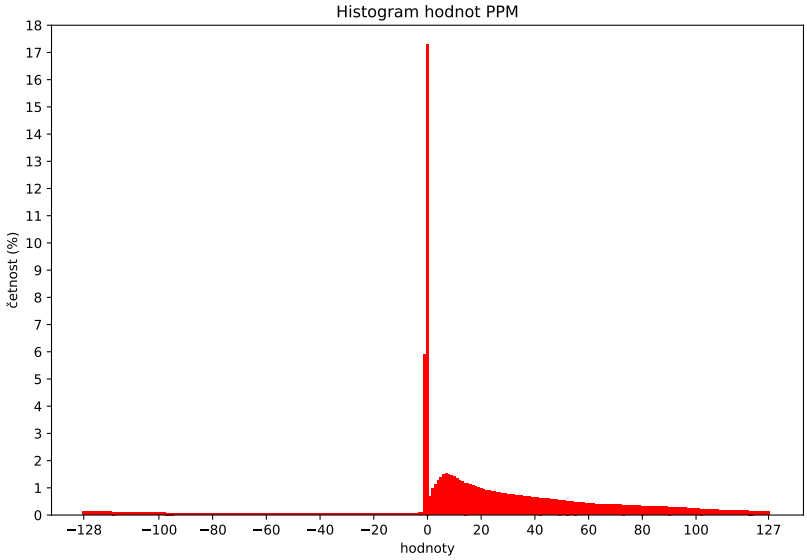
### 5.2.2 Výsledky - lytro fotoaparát

Výsledky pro lytro dataset se v mnohém lišily od matice kamer. Hlavním rozdílem bylo, že více-dimenzionální přístup byl lepší než běžné prediktory. Pro 2D prediktory byl, jak se dalo očekávat, nejlepší Paeth. Ve 3D variantě však nejlepších výsledků dosáhlo jednoduché kódování rozdílů a pro prediktory rozšířené do čtvrté dimenze byl nejlepší Paeth.




Obrázek 5.8: Průměrný kompresní poměr lytro datasetu (více je lépe)

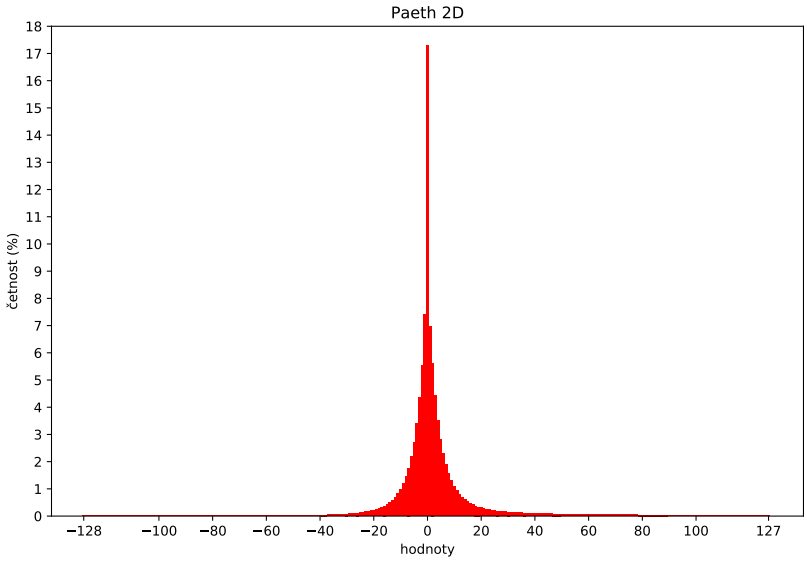
Nakonec srovnávací histogram hodnot ze všech snímků lytro datasetu a histogramy pro jednotlivé prediktory:




Obrázek 5.9: Histogram původních hodnot

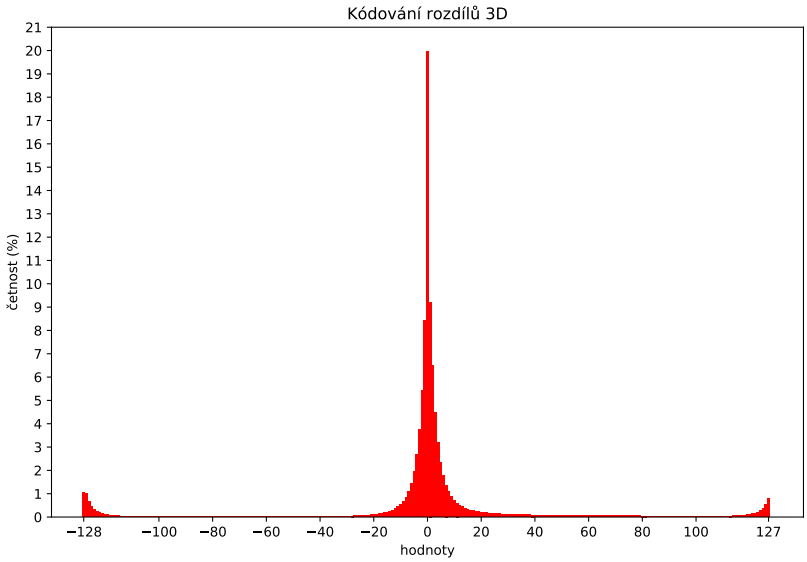
Vysoké procento nuly (černých pixelů) ovlivňuje fakt, že 14.2% všech obrázků je tvořeno prakticky pouze jen černými pixely.

### 2D




Obrázek 5.10: Histogram hodnoty po průchodu Paeth 2D

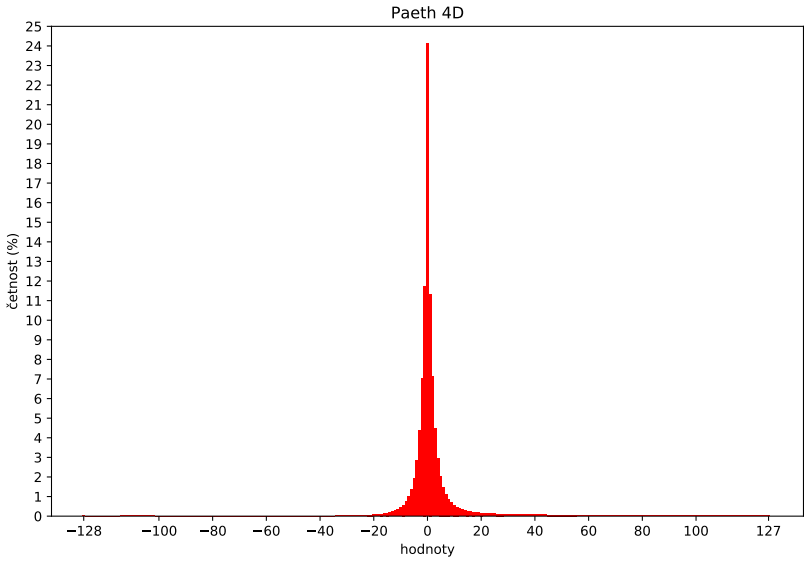
### 3D




Obrázek 5.11: Histogram hodnoty po průchodu Kódování rozdílů 2D

### 4D

Z možných variant prediktoru Paeth po rozšíření do čtvrté dimenze bylo vybráno proložení rovinou skrze $x$ z okolních snímků (4.11) a byl vyhledáván pixel nejblíže tomuto odhadu ze všech okolních pixelů ze všech snímků.




Obrázek 5.12: Histogram hodnoty po průchodu Paeth 4D

# Kapitola 6

# Závěr

Předmětem této práce bylo vytvořit knihovnu zprostředkovávající bezeztrátovou kompresi plenoptických fotografií. Počátek se věnoval teorii popisující způsob jakým plenoptický fotoaparát zachycuje světelné pole a dále také principu snímání běžného digitálního fotoaparátu. Následující kapitola obsahovala teorii týkající se bezeztrátové komprese a také strukturu kompresního řetězce, který je složen z barevného modelu, prediktoru a kodéru. Hlavní částí bylo rozšíření běžných prediktorů, které pracují v kontextu jednoho snímku, do více dimenzí. Bylo rozšířeno celkem 5 různých typů prediktorů.

K testování knihovny byly vytvořeny 2 datasety, které se lišily principem záznamu, a na nich byla knihovna nezávisle testována, a poté srovnána s PNG. Sledovaným parametrem byl kompresní poměr, který určí, kolikrát menší je výstup. Testy byly spouštěny s barevným modelem LYUV a kodérem Deflate, přičemž cílem bylo najít nejúčinější prediktor pro každé dimenzionální rozšíření.

Pro dataset, kde byly snímky pořízeny pomocí stanfordské matice kamer (nebo gantry) měly nejlepší prediktory v každé dimenzi větší kompresní poměr než PNG. Díky velké rozdílnosti snímků ale nebylo možné využít podobnosti sousedních obrázků k menší chybě predikce, takže s růstem dimenze prediktoru klesal kompresní poměr.

Druhý dataset, ve kterém byly fotografie zachyceny pomocí Lytro fotoaparátu, měl rozdílné výsledky. Lepšího kompresního poměru než PNG dosáhly pouze některé prediktory ve 4D rozšíření. Zde se tedy uplatnilo predikovat pixel z okolních snímků, protože jejich korelace byla velmi znatelná.

Z těchto výsledků je patrné, že i když je více dimenzionální přístup u lytro fotografií lepší, tak je zde ještě hodně místa ke zlepšování a případným návrhům, jak který prediktor rozšířit, aby využil plného potenciálu, který se skrývá v podobnosti těchto snímků. Dále také to, že pro implementované rozšíření prediktorů není varianta, která by dokázala vhodně využít korelace mezi snímky ve stanfordském datasetu.

Není tedy vhodné jednoznačně určit, která metoda je obecně nejlepší pro všechny typy snímání 4D světelného pole, protože korelace snímků hraje velkou roli v kvalitě komprese při zvyšování dimenze prediktoru. Z výsledků ale vyplývá, že pro plenoptické fotografie pořízené pomocí Lytro fotoaparátu, je nejlepší prediktor Paeth 4D a pro matici kamer, či gantry je nejlepší Paeth 2D.

# Literatura


[1] *JPEG Pleno Database: EPFL Light-field data set* . [Online; navštíveno 13.05.2019]. URL https://jpeg.org/plenodb/lf/epfl/

[2] *PNG (Portable Network Graphics) Specification* . [Online; navštíveno 13.05.2019]. URL https://www.w3.org/TR/PNG-Filters.html

[3] Avramović, A.; Reljin, B.: *Gradient Edge Detection Predictor for Image Lossless Compression. Proceedings Elmar - International Symposium Electronics in Marine*, 1 2010.

[4] Bařina, D.; Zemčík, P.: *Multimédia* . [Online; navštíveno 13.05.2019]. URL http://www.fit.vutbr.cz/~ibarina/pub/mul-opora-2017-03-28.pdf

[5] Gortler, S. J.; Grzeszczuk, R.; Szeliski, R.; aj.: *The Lumigraph. SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques Pages 43-54*, 1996, doi:10.1145/237170.237200.

[6] Kučera, J.: *LYTRO Meltdown* . [Online; navštíveno 13.05.2019]. URL http://optics.miloush.net/lytro/TheFileFormat.aspx

[7] Liu, C.: *Arithmetic Coding* . [Online; navštíveno 13.05.2019]. URL https://people.cs.nctu.edu.tw/~cmliu/Courses/Compression/chap4.pdf

[8] Rane, S. D.; Sapiro, G.: Evaluation of JPEG-LS, the new lossless and controlled-lossy still image compression standard, for compression of high-resolution elevation data. Říjen 2001, doi:10.1109/36.957293.

[9] Tao, M.: *Unified Multi-Cue Depth Estimation from Light-Field Images: Correspondence, Defocus, Shading, and Specularity.* Disertační práce, Electrical Engineering and Computer Sciences University of California at Berkeley, 2015.

[10] Thurow, B. S.: *Plenoptic Cameras: The Future of Imaging.* [Online; navštíveno 13.05.2019]. URL https://www.youtube.com/watch?v=nyRgcD4VxE8

[11] Vaish, V.; Adams, A.: *The (New) Stanford Light Field Archive.* [Online; navštíveno 13.05.2019]. URL http://lightfield.stanford.edu/

[12] Viola, I.; Řeřábek, M.; Ebrahimi, T.: Comparison and Evaluation of Light Field Image Coding Approaches. *IEEE Journal of Selected Topics in Signal Processing*, ročník 11, č. 7, Říjen 2017: s. 1092–1106, ISSN 1932-4553, doi:10.1109/JSTSP.2017.2740167.

[13] Řeřábek, M.; Ebrahimi, T.: *New Light Field Image Dataset . 8th International Workshop on Quality of Multimedia Experience (QoMEX), Lisbon, Portugal*, 2016.

[14] Švihálek, J.: *Metody zpracování plenoptických obrazových dat.* Magisterská práce, České vysoké učení technické v Praze, 2016.

# Příloha A

# Obsah na DVD