VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

http://fit.vut.cz

Projektová praxe 2019/2020

Komprese 4D světelných polí

Drahomír Dlabaja*

Abstrakt

Tato práce se zabývá zkoumáním pokročilých metod pro kompresi 4D světelných polí na bázi čtyřrozměrných bloků. V této práci byla navržena čtyřrozměrná směrová intra predikce, která byla implementována ve formě univerzálního modulu. Dále se tato práce věnuje zobecnění existující kompresní metody pro práci s hyperbloky obecných rozměrů. Práce také implementuje preprocessingovou operaci, která posunuje pohledy světelného pole tak, aby byla minimalizována vzájemná disparita.

Čtyřrozměrné intra predikce bylo dosaženo rozšířením a zobecněním metody intra predikce z doporučení H.265 do více dimenzí. Navržený algoritmus umožňuje libovolně označit dostupné (již zkomprimované) a nedostupné vzorky pro predikci a je tak invariantní vůči průchodu a dělení bloků. Implementovaný modul byl integrován do existujícího kodeku pro 4D světelná pole.

Zobecnění velikosti kompresních bloků umožnily tuto velikost přizpůsobit povaze dat jak v prostorové, tak v úhlové doméně, což mělo výrazný vliv na kompresní poměr. Tato úprava odemyká různé další možnosti, jak data komprimovat. Pomocí této úpravy lze implementovat dělení bloků do stromové struktury a velikost bloků tak bude možné přizpůsobit lokálním vlastnostem v obrázku, což by dále mělo zlepšit kompresi.

Vzájemný posun pohledů byl implementován v kombinaci s metodou, která minimalizuje chybu mezi pohledy. Zejména pro obrázky z pole kamer se ukázalo, že zaostření pohledů na nejobsáhlejší část scény má pozitivní vliv na kompresi.

Kodek s popsanými modifikacemi překonává state-of-the-art metody pro videokompresi pro obrázky s nižší disparitou (Lytro, Stanford). Pro obrázky s vyšší disparitou (Stanford, SAUCE) metoda překonává state-of-the-art metody při kompresi s kvalitou vyšší než 43 dB PSNR.

Klíčová slova: 4D světelné pole — Komprese — Predikce – Disparita

Přiložené materiály: Veřejný Repozitář

*xdlaba02@stud.fit.vut.cz, *Fakulta informačních technologií, Vysoké učení technické v Brně*

1. Úvod

Pro popis trojrozměrné scény z každého možného úhlu lze definovat plenoptickou funkci $P(x,y,z,\phi,\psi)$, kde (x,y,z) je pozice a (ϕ,ψ) je pozorovací úhel. Hodnota P určuje intenzitu světelného paprsku. Definici lze dále rozšířit o t (čas) pro popsání dynamické scény. Tato práce pracuje s fotografiemi pořízenými polem fotoaparátů nebo jedním senzorem, který je předcházen polem mikročoček. Oba případy lze reprezentovat jako pohledy na scénu uspořádané v pravidelné mřížce v rovině. Tato reprezentace je často nazývána 4D světelné pole a lze jej popsat funkcí $L(u,v,s,t)$, kde (u,v) jsou prostorové souřadnice (souřadnice pohledu) a (s,t) jsou původní úhlové souřadnice (souřadnice pixelu).

Světelné pole implicitně zachovává informace o prostorových i materiálových vlastnostech ve scéně. Jelikož jde světelné pole zaznamenat i zobrazit jak v reálném, tak v digitálním světě, lze jej považovat za prostředek komunikace vizuálních dat mezi těmito světy. Důležitým aspektem světelných polí, který je potřeba řešit, je jejich paměťová náročnost.

Na světelné pole se lze dívat jako na pole běžných, avšak silně korelovaných obrazových snímků. Je zřejmé, že tato korelace nemůže v případě komprese zůstat bez povšimnutí. Ukázalo se, že společně s ko-

relací v obrazové rovině lze korelaci mezi pohledy dobře využít při kompresi na bázi 4D bloků [2]. Tato bloková komprese byla srovnána s ostatními kompresními metodami [1], kde byla ve většině případů překonána. Této metodě uniká několik aspektů, které mají vliv na úroveň komprese.

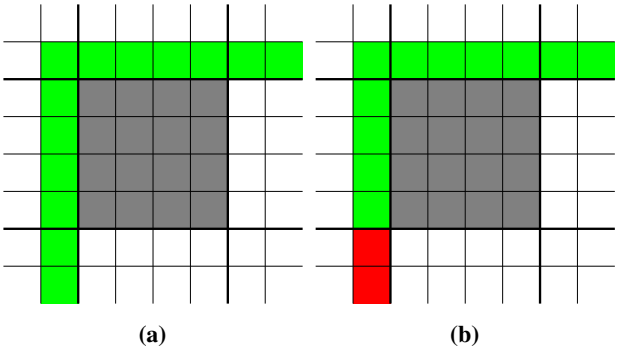
Jedna z věcí, kterou lze pro kompresi využít, je korelace mezi sousedními bloky. Stejně jako u 2D obrazových dat, i u světelných polí platí, že krajní vzorky sousedních bloků jsou s velkou pravděpodobností podobné obsahu aktuálně komprimovaného bloku. Obsah bloku lze proto zaznamenat jako metodu predikce právě komprimovaného bloku s využitím již zakódovaných vzorků a chyby, které se tato predikce dopustila (rezidua, rozdílu). Při správné volbě směrového vektoru je zakódovaná chyba predikce společně s metodou menší, než data kódovaná bez predikce, a dojde tak ke kompresi dat. Návrhu takového predikčního schématu pro 4D světelná pole se věnuje kapitola 2. Navržená metoda je zaměřená na univerzálnost použití, její přímou implementaci tak lze například použít v kombinaci s hyperbloky libovolných rozměrů.

Dalším aspektem je tvar a velikost hyperbloků. U 2D obrázků lze bezpečně předpokládat, že data budou mít stejnou povahu ve vertikálním i horizontálním směru. Taková data lze komprimovat po čtvercových blocích, jiné rozměry by ve většině případů nepřinesly zlepšení. Pro 4D světelná pole je však situace jiná. Data mají podobnou povahu navzájem v úhlových souřadnicích a navzájem v prostorových souřadnicích, ale ne mezi úhlovými a prostorovými souřadnicemi. Dává proto smysl používat bloky, které mají v prostorových a úhlových souřadnicích rozdílnou velikost. Implementaci komprese s hyperbloky obecných rozměrů se věnuje kapitola 3.

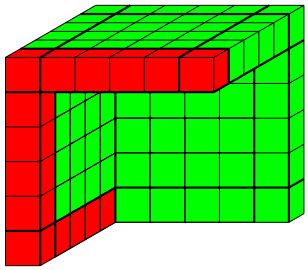
Třetím zkoumaným vlivem je disparita mezi pohledy. Experimenty ukázaly, že pro blokovou metodu je maximální dosažitelná úroveň komprese korelována s disparitou mezi pohledy. Platí, že světelná pole s menší disparitou lze komprimovat lépe než světelná pole s větší disparitou. Bylo proto otázkou, jestli nejde před samotnou kompresí minimalizovat disparitu v obrázku tak, aby došlo k lepší kompresi beze ztráty dat. Tato hypotéza se ukázala jako platná. Metoda, která disparitu minimalizuje, je popsána v kapitole 4.

Kodek s popsanými modifikacemi překonává state-of-the-art metody pro videokompresi pro obrázky s nižší disparitou (Lytro, Stanford). Pro obrázky s vyšší disparitou (Stanford, SAUCE) metoda překonává state-of-the-art metody při kompresi s kvalitou vyšší než 43 dB PSNR.

2. Intra predikce pomocí směrových vektorů



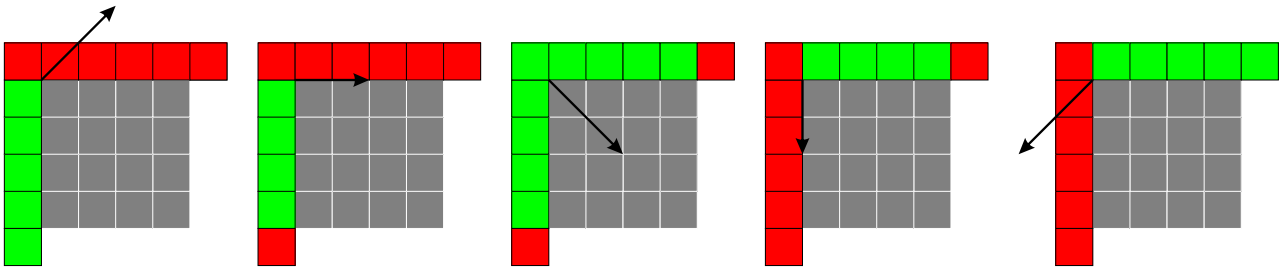
Obrázek 1. Obrázek 1a zeleně demonstruje vzorky, jejichž existenci prediktor předpokládá. Obrázek 1b červeně značí vzorky, které nejsou dostupné na straně dekodéru při scanline průchodu. Přístup k nedostupnému vzorku řeší uživatel, optimální strategie je vracet nejbližší dostupný vzorek.



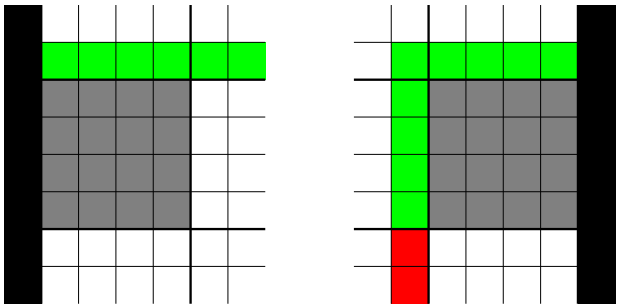
Obrázek 2. Ve třech rozměrech se scanline průchod komplikuje. Na první pohled není zřejmé, které vzorky jsou dostupné na straně dekodéru.

Intra predikce byla poprvé specifikována v doporučení ITU-T H.261, a to pro kompresi 2D snímků videa. Od té doby je v různých formách nedílnou součástí drtivé většiny používaných kodeků pro videokompresi. Novější specifikace se od starších liší zejména množstvím režimů intra komprese. Největší pokrok lze pozorovat mezi specifikacemi ITU-T H.264 a ITU-T H.265. Doporučení ITU-T H.264 používalo devět pevně určených modů predikce, které byly specifikovány pro určité velikosti bloků. Oproti tomu je v ITU-T H.265 zavedeno 35 módů predikce, které se musí vypořádat s dělením bloků do oktalových stromů (CTU). H.265 zároveň zavádí vyhlazení referenčních vzorků pomocí FIR filtru, což má za následek zlepšení predikce a lepší komprese. Existuje mnoho publikací zabývajících se metodou rychlé a efektivní volby vhodného režimu intra predikce pro real-time kompresi videa, ale žádná nezavádí predikci pro více-než-dvoudimenzionální bloky. Tato práce proto primárně vychází z poznatků již zmíněných doporučení ITU-T.

V této práci je představeno řešení pro intra predikci libovolně dimenzionálních bloků, které je výsledkem zobecnění metody používané v doporučení ITU-T



Obrázek 3. Zeleně jsou označeny vzorky, které bude prediktor pro daný směr číst. Prediktor nikdy nečte červeně označené vzorky. Uživatel se na toto chování může spolehnout a nemusí provádět dodatečné kontroly.



Obrázek 4. Obrázky 4a a 4b demonstrují případy, kdy je predikovaný blok okrajem obrázku. Začerněné vzorky nejsou dostupné ani na straně kodéru. Přístup k nedostupnému vzorku řeší uživatel, optimální strategie je vracet nejbližší dostupný vzorek. Vertikální okraje jsou řešeny obdobně.

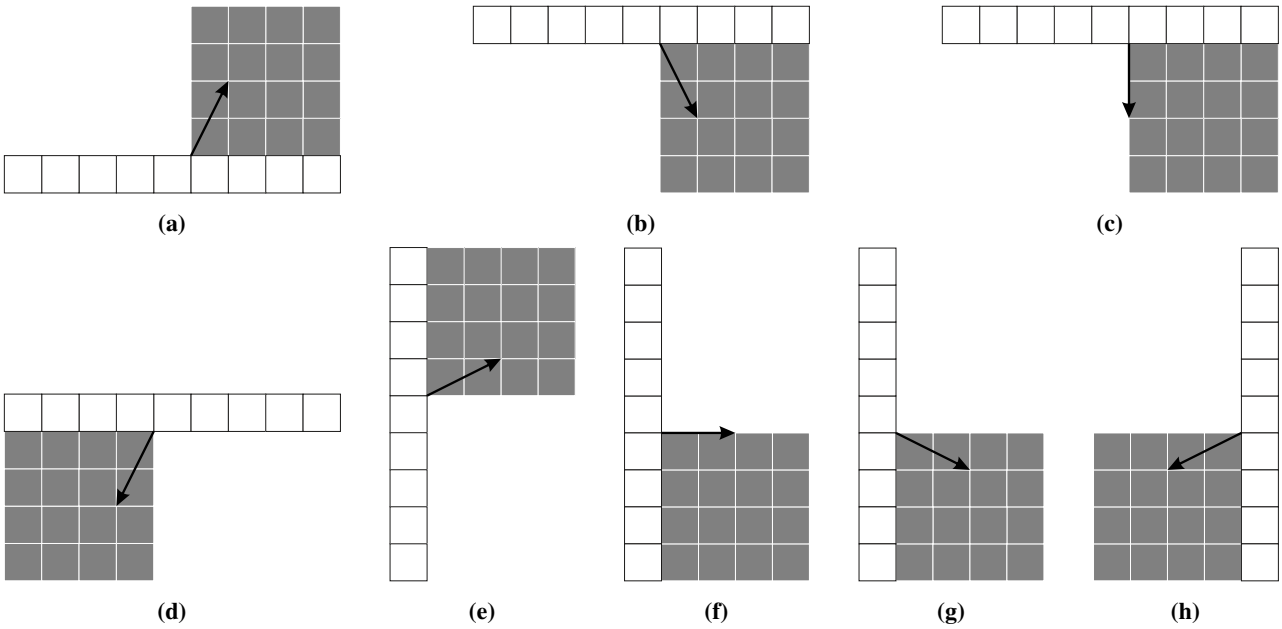
H.265. Řešení je rozděleno na dva kroky, přičemž každý krok řeší různé problémy, které vycházejí z multidimenzionality dat. V prvním kroku jsou referenční vzorky n-dimenzionálního okolí promítnuty do $n-1$-dimenzionální roviny podle n-dimenzionálního směrového vektoru. V tomto kroku je řešen problém nedostupného okolí, který se týká predikce okrajových bloků. Navržené řešení poskytuje rozhraní pro volbu strategie výběru náhradních vzorků. Lze tak například použít nejbližší dostupný vzorek, případně nějakou pokročilejší formu interpolace. V $n-1$-dimenzionální rovině poté probíhá vyhlazení vzorků pomocí FIR filtru, podobně jako v ITU-T H.265. V druhém kroku se vzorky z predikovaného n-dimenzionálního bloku promítnou z připravené $n-1$-dimenzionální roviny. V tomto kroku jsou již všechny vzorky vždy dostupné, takže tento krok je velmi rychlý. Tento dvoukrokový přístup neredukuje časovou ani paměťovou náročnost algoritmu, ale velmi výrazně zjednodušuje implementaci algoritmu a řešení okrajových případů.

Vstupem algoritmu je velikost predikovaného bloku, jeho dostupné okolí a směrový vektor, podle kterého se okolí rozšíří do bloku. Platí, že každý n-dimenzionální blok má $2n$ $n-1$-dimenzionálních sousedů. Algoritmus beze ztráty obecnosti předpokládá, že existuje n sousedů, ze kterých je možné predikovat, a jsou to vzorky s příslušející souřadnicí nižší než souřadnice vzorků v predikovaném bloku. Pro dva rozměry je předpokládané okolí zaznačeno na obrázku 1a. V případě potřeby lze algoritmus vždy zavolat s invertovaným přístupem k okolí a invertovaným směrovým vektorem tak, aby platil zmíněný předpoklad. Z tohoto předpokladu plyne, že alespoň jedna komponenta směrového vektoru musí být kladná.

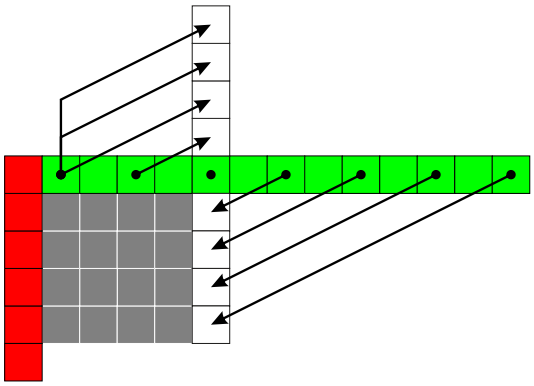
K okolí příslušejícímu ne-kladné komponentě směrového vektoru algoritmus vůbec nepřistupuje, pro korektní predikci nemusí vůbec existovat. Obrázek 3 ukazuje, ke kterým vzorkům prediktor přistupuje v závislosti na kvadrantu směrového vektoru. Uživatel se na tento přístup může spolehnout, může například volat predikci jen pro takové vektory, pro které má dostupné okolí. Alternativní možností uživatele je volat algoritmus nezávisle na dostupnosti vzorků a poté pouze provádět korekce přístupu k nim, pokud se o to algoritmus pokusí.

Reálně se nedá předpokládat, že všechny vzorky z předpokládaného okolí budou dostupné na straně dekodéru. Pro průchod po řádcích je okolí dostupné na straně dekodéru zaznačeho na obrázku 1b. Obrázek 2 zobrazuje vzorky dostupné na straně dekodéru pro kompresi ve 3D. Už ve třech rozměrech není na první pohled zřejmé, které vzorky jsou při průchodu po řádcích dostupné na straně dekodéru. Pro průchod po řádcích v libovolném počtu dimenzí platí, že kodér může v daném rozměru přistupovat k vzorkům z následujících bloků pouze pokud v jednom z vyšších rozměrů zároveň přistupuje k předcházejícím blokům. Toto si lze snadno představit ve dvou rozměrech s pomocí obrázku 1b. Prediktor může přistoupit k vzorkům přesahujícím velikost bloku na horizontální ose pouze pokud na vertikální ose přistupuje *do minulosti*. Prediktor nemůže přistoupit k vzorkům přesahujícím velikost bloku na vertikální ose, protože vertikální rozměr je nejvyšší.

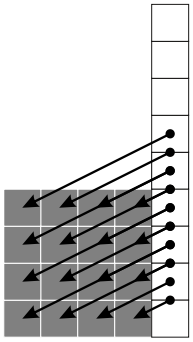
Algoritmus se také musí vypořádat s okrajovými případy, ve kterých nejsou potřebné vzorky dostupné ani na straně kodéru. Takové případy přísluší kra-



Obrázek 5. Poloha a orientace primárního souseda v závislosti na oktantu směrového vektoru.



Obrázek 6. Prvním krokem prediktoru je promítnutí předpokládaných dostupných vzorků z okolí na primárního souseda.



Obrázek 7. Druhým krokem prediktoru je promítnutí vzorků z primárního souseda do bloku. Provádí se interpolace.

jním blokům komprimovaného obrázku, jak je demonstrováno na obrázku 4. Při kompresi světelných polí se ukázalo, že takto okrajovým případem je alespoň v jedné dimenzi každý blok. Tyto případy je proto nutné řešit více důkladně než při kompresi ve 2D. Tyto případy jsou opět řešeny na straně uživatele, který může volit vlastní strategii. Podobně jako při problému dostupnosti při průchodu po řádcích může uživatel algoritmus volat jen s takovými vektory, které nebudou přistupovat k nedostupným vzorkům podle obrázku 3. Alternativní možností uživatele je volat algoritmus nezávisle na dostupnosti vzorků a poté pouze provádět korekce přístupu k nim. Za korekci se v tomto případě považuje přesměrování na nejbližší dostupný vzorek.

Algoritmus začíná s volbou primární dimenze. Za tu je prohlášena dimenze s největší absolutní hodnotou prvku ve směrovém vektoru. Primární dimenze určuje tvar primárního souseda. Primární soused je $n-1$-dimenzionální blok, do kterého je promítnuto veškeré předpokládané okolí. Volba s největší absolutní hodnotou zajistí, že všech n $n-1$-dimenzionálních sousedů lze podle zvoleného směrového vektoru promítnout do jednoho nejmenšího možného primárního souseda. Například pro blok velikosti $4\times4\times8\times8$) a směrový vektor $(1,-3,2,-2)$ bude mít primární soused velikost $8\times12\times12$.

Poloha a orientace primárního souseda ve dvou rozměrech je pro všechny oktanty směrového vektoru demonstrována na obrázku 5. Ze všech případů polohy primárních sousedů jsou nejvíce zvláštní případy 5a a 5h. V těchto případech primární soused neleží v rovině žádného existujícího souseda. Pro správné pochopení kroků algoritmu budou tyto kroky vizualizovány právě na tomto případě.

Samotné promítnutí se provede rychle se zaokrouhlením na nejbližší celý vzorek. Již v ITU-T H.265 bylo zjištěno, že preciznější promítání přináší nemalé výpočetní nároky a nemá velký vliv na kompresi. Tento krok je pro směrový vektor z obrázku 5h demonstro-

ván obrázkem 6. Zelené vzorky představují předpokládané okolí, bílou barvou jsou zaznamenány vzorky primárního souseda. Červenou barvou jsou zaznamenány vzorky, ke kterým prediktor nemusí a nesmí přistupovat, jelikož vektor spadá do případu 3e.

Podobně jako v ITU-T H.265 je primární soused je před průmětem do bloku podroben 3^n Gaussově filtru. Toto filtrování předejde případnému vzniku vysokých frekvencí v reziduu a má proto pozitivní vliv na kompresi. Toto chování bylo experimentálně potvrzeno.

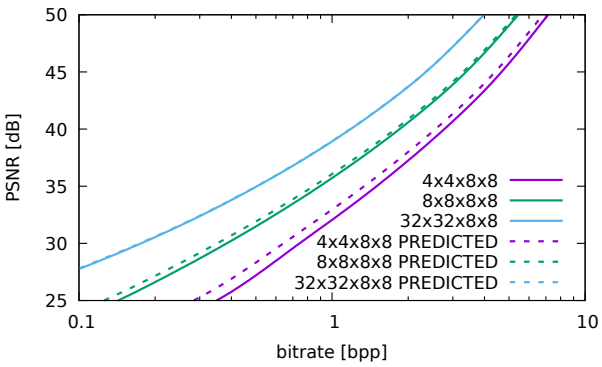
Pomocí takto zpracovaného primárního souseda poté probíhá samotné promítnutí do bloku. Tento krok je demonstrován obrázkem 7. Každému vzorku v bloku je pomocí inverzního směrového vektoru nalezena pozice na primárním sousedovi, ze kterého je poté lineárně interpolována predikovaná hodnota vzorku. V tomto kroku je již zajištěno, že všechny potřebné vzorky jsou dostupné.

Navržená metoda je implementována jako modul pro využití v kompresních řetězcích, které data kódují po blocích v libovolném počtu dimenzí. Očekávané využití metody je pro kompresi 4D světelných polí, avšak řešení je dostatečně obecné na to, aby dokázalo pracovat se 3D volumetrickými (lékařskými) daty či s běžnými 2D fotografiemi. Navržené řešení je určeno pro kombinaci s algoritmem, který dokáže volit módy predikce tak, aby bylo dosaženo požadovaného poměru rychlosti a zkreslení.

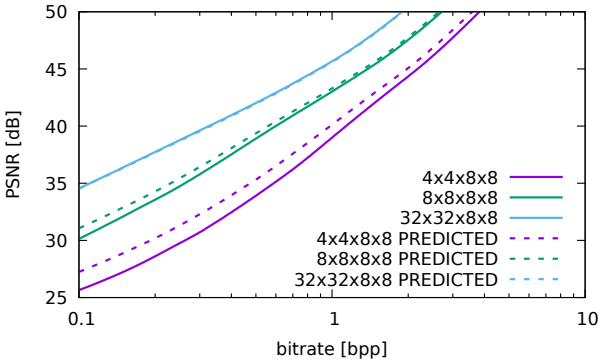
Modul byl integrován do existujícího kodéru světelných polí, který pracuje se čtyřrozměrnými hyperbloky obecných rozměrů (kapitola 3). Pro každý blok je systematicky testováno 529 různých směrů, přičemž je zvolena predikce s nejmenší absolutní chybou oproti originálnímu bloku. Ke směrové predikci je přidána také DC predikce a planární predikce, která vzniká jako průměr dílčích směrových predikcí s bázovým vektorem. Grafy 8 demonstrují, že takto integrovaná predikce redukuje bitrate jen pro menší bloky. Testování hrubou silou zároveň navyšuje časovou náročnost o několik řádů. Autor proto navrhuje použít predikci v kombinaci s CTU. Zároveň by se dalo diskutovat o použití chytrého algoritmu, který by dokázal volit směrové vektory na základě heuristiky.

3. Hyperbloky obecných rozměrů

Jedním z největších omezení, kterými disponovalo přímé rozšíření kodéru JPEG do čtyř rozměrů, bylo použití hyperbloků pouze ve tvaru hyperkrychle. Při správné velikosti bloku sice kódování dosahovalo dobré komprese, ale bylo zřejmé, že komprimovaná data mají v úhlových dimenzích jinou povahu než v prostorových dimenzích. Jedním z logických kroků proto



(a) Světelné pole HaToy.



(b) Světelné pole Take2_1.

Obrázek 8. Grafy demonstrují klesající efektivitu predikce s rostoucí velikostí bloku.

bylo existující kodér přepsat tak, aby dokázal využít hyperbloky libovolných rozměrů, které by poté bylo možné přizpůsobit komprimovaným datům.

Tato kapitola se zabývá popisem toho, co všechno změny zahrnovaly. Nejdříve je popsáno, jak s bloky funguje původní framework, jeho myšlenka, výhody a nevýhody. Jsou zhodnoceny návrhy, jak bylo možné problém řešit. Dále je vysvětlno jakými změnami kodér prošel a jaká úskalí to přineslo. Je zhodnocen vliv jak na kompresi, tak i na časovou náročnost. Kapitola končí výčtem možností, jak na tuto práci navázat a co by mohlo řešit některé existující problémy.

Kodér byl v základní variantě navržen s myšlenkou statických bloků o velikosti 8^4. To vycházelo z kodéru JPEG, který používal bloky 8^2. Nebylo předokládáno, že by kodér mohl používat bloky jiných rozměrů, proto byl blok reprezentován datovou strukturou, která poskytovala čtyřrozměrné rozhraní k jednorozměrnému statickému poli o velikosti 4096 prvků. Tento přístup měl tu výhodu, že již při kompilaci šly do jisté míry zjistit paměťové nároky, kompilátor tak měl možnost v případě potřeby provádět optimalizace. Zároveň se tak dalo vyhnout dynamickým alokacím, které by v prvních fázích vývoje mohly vést na větší chybovost.

Později se ukázalo, že bloky 8^4 jsou nedostatečné, bylo proto rozhodnuto velikost bloku definovat po-

mocí šablonového parametru v jazyce C++. Tímto způsobem bylo možné stále využívat statické pole pro data, ale velikost bloku bylo možné volit při kompilaci. To vedlo k tomu, že bylo potřeba přepsat do šablon také veškerý kód, který takové bloky používá, zejména DCT a cik-cak průchod. Šablonový kód vedl na časově náročenější kompilaci, ale na druhou stranu umožňoval inlinovat funkce, u kterých to dává smysl.

Ortogonálně k tomu byl na základě častého experimentování s jiným počtem dimenzí přidán šablonový parametr pro počet dimenzí. S tím bylo spojené přepsání celého frameworku tak, aby byl schopný pracovat univerzálně, tedy s libovolně dimenzionálními bloky libovolných rozměrů. Byla to netriviální a časově náročná úloha, která však vedla na minimalistický, dobře pochopitelný a často rekurzivní kód, který bylo možné instancovat pro všechny potřebné experimenty s blokovou kompresí. Zároveň tím extrémně odlehčila způsobu přemýšlení ve čtyřech rozměrech, které vedlo na snažší řešení nových problémů. Autorovi stačilo nový problém vyřešit ve dvou a třech rozměrech, abstrahovat rekurzivní pattern a otestovat, jestli platí i pro 4D.

Taková implementace je sice silně univerzální, ale disponuje řadou nevýhod. Tvar bloku je daný statickým parametrem, uživatel si tak musí při kompilaci zvolit, jaké bloky bude používat. Další nevýhodou je, že v každé instanci se předpokládá komprese pomocí hyperkrychlí, což často nerespektuje povahu dat. Tyto nevýhody například vylučují použití CTU, které ze své podstaty pracuje s bloky různých velikostí.

První idea řešení byla z jednoho šablonového parametru udělat pole parametrů, které určují velikost bloku. Tento přístup by zachoval výhody statického bloku, ale zároveň i jeho nevýhody, jeho rozměry by musely být známé při kompilaci. Druhou možností bylo zavedení dynamické alokace. Takové řešení by umožňovalo vytvoření libovolných bloků na požádání a řádné zapouzdření by řešilo korektní správu paměti. Toto řešení však vyvolávalo nedůvěru. Nedalo se spoléhat, že dynamická alokace bude stejně rychlá jako zásobníková alokace. Vzhledem k častému vytváření nových bloků a rušení starých, oproti znovupoužití, by to mohlo vést na nepříjemné zpomalení běhu, což by brzdilo už tak časově náročný proces validace kódu po každé změně.

Po zhodnocení byla zvolena dynamická alokace. Dynamicky alokované pole bylo zapouzdřeno do šablonové struktury, kde šablonové parametry specifikují datový typ vzorku a počet dimenzí. Velikost bloku je zadávána v konstruktoru. Změna zahrnovala refaktorizaci drtivé většiny kódu, která pracuje s bloky. Z funkcí byl odstraněn šablonový parametr pro velikost bloku, rozměry bloku jsou poskytovány v jeho rozhraní nebo v poli jako běžný parametr. Přepsány byly jednotlivé kroky komprese. Vzhledem k jejich existující rekurzivní implementaci to většinou nepředstavovalo problém.

Vliv velikosti bloku na datový tok lze vyčíst z obrázku 11. Dynamická alokace se projevila jen $< 10\%$ nárůstem doby komprese a dekomprese. Za tuto cenu bylo při správné volbě velikosti bloku u některých obrázků dosaženo až 30% redukce bitrate při stejné hodnotě PSNR. Této redukce však bylo dosaženo pouze při prohledávání velikostí bloku hrubou silou. Aktuálně proto problém spočívá v tom, jak rychle a efektivně najít optimální velikost bloku. Zároveň se předpokládá, že se vyplatí navázat implementací CTU, která bude velikosti bloku volit podle lokálních změn v obrázku.

Pro vyvážení zvýšených nároků dynamické alokace autor navrhuje implementaci zásobníkového alokátoru. Taková implementace by však musela zajistit, že k dealokaci bloků bude vždy docházet v opačném pořadí oproti alokaci. Alternativním řešením by mohl být předem alokovaný pool bloků různých typů. V obou případech by šlo využít faktu, že při spuštění programu jsou známé všechny informace o počtu a typu blokových bufferů, které bude kodér i dekodér potřebovat.

4. Vzájemný posun pohledů

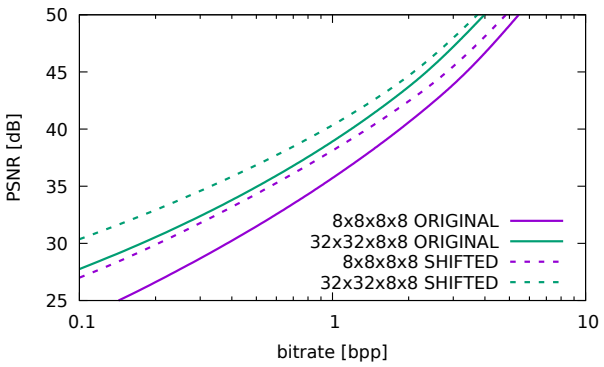
(a) Před vzájemným posunutím. **(b)** Po vzájemném posunutí.

Obrázek 9. Průměr ze všech pohledů ve světelném poli Take2_1 před (9a) a po (9b) vzájemném posunutí.

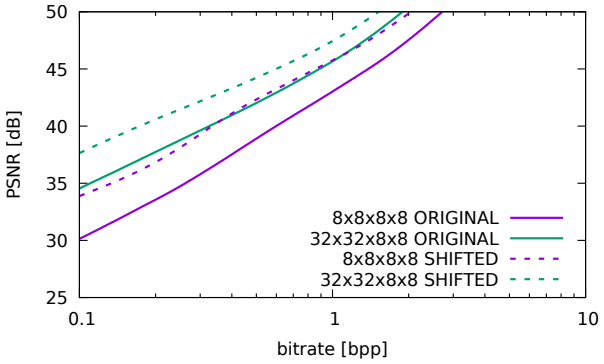
(a) Před vzájemným posunutím. **(b)** Po vzájemném posunutí.

Obrázek 10. Průměr ze všech pohledů ve světelném poli HaToy před (10a) a po (10b) vzájemném posunutí.

Čtyřrozměrné světelné pole reprezentuje paprsky světla procházející dvourozměrným rozhraním ve trojrozměrném prostoru. Definice ovšem nic neříká o tom,



(a) Světelné pole HaToy.



(b) Světelné pole Take2_1.

Obrázek 11. Grafy demonstrující redukci datového toku po nalezení optimální velikosti bloku a optimálního vzáhemného posunu.

jaké má rozhraní tvar, velikost, jaký je rozsah a hustota vzorkování v prostorové a úhlové doméně. Často se stává, že záznam světelného pole je rozostřený v prostorové doméně, což jde vidět na obrázku 9a a 10a. Jednotlivé pohledy tak sice můžou být vzájemně velmi podobné, ale o několik pixelů posunuté. To může vést až k tomu, že hyperblok obsahuje z každého pohledu jinou logickou část scény. Vzorky tak v rámci bloku nejsou korelované a to vede na špatnou kompresi.

Logickým krokem k řešení tohoto problému je posunutí pohledů tak, aby se mezi nimi minimalizovala vzájemná chyba. Jelikož se dá předpokládat, že pohledy byly zaznamenány pravidelným polem kamer, lze také předpokládat, že je posun stejný ve vertikálním i horizontálním směru. Zároveň lze předpokládat, že posun mezi libovolnými dvěma sousedními pohledy je vždy stejný. Celý posun proto lze reprezentovat jednou hodnotou.

Středový pohled se prohlásí za fixní. Každý pohled je v horizontálním, resp. vertikálním směru posunut o počet pixelů, který odpovídá hodnotě posunu vynásobené horizontální, resp. vertikální vzdáleností od středového pohledu. Platí, že snímky horizontálně nalevo a vertikálně nahoře mají zápornou vzdálenost.

Posunuté vzorky polohou přesahující rozměry pohledu jsou rolovány do pomyslného torusu pomocí operace modulo. Tato operace ve výsledném obrázku tvoří vysoké frekvence kolem okrajů a v závislosti na velikosti posunu obsahují okraje pohledů vzájemně nekorelovaná data. Z pohledu blokového kodéru to však nevadí, pokud v rámci jednoho bloku nejsou korelovaná ani původní data. V takovém případě dojde z pohledu kodéru pouze ke zvýšení průměrné korelace a tím i zlepšení komprese. Nevýhodou je, že po dekompresi můžou mít okraje horší kvalitu než střed obrázku. Vliv posunutí na ostrost v prostorové doméně lze pozorovat na obrázcích 9 a 10. Oba tyto obrázky jsou v základu prostorově rozostřené. U obrázku 10 lze navíc pozorovat, že pro účely komprese bylo v tomto případě lepší zaostřit na rozsáhlou část scény (tapetu) oproti části podstatné.

Implementovaný posun v aktuální verzi pracuje s celými pixely. Při návrhu bylo zvažováno, jestli má být umožněn posun o necelý počet pixelů v kombinaci s interpolací. Od tohoto návrhu bylo upuštěno. Experimenty ukázaly, že posun na sub-pixelové úrovni má zanedbatelný vliv na kompresi, interpolace je časově náročná a navíc ztrátová.

Vliv posunutí na datový tok lze vypozorovat z obrázku 11. Z vyhodnocení plyne, že pro obrázky, které jsou originálně rozostřené v prostorové doméně dokáže posun redukovat bitrate až o 45% při stejné kvalitě. Nejvíce se posun projevuje na obrázcích z datasetu SAUCE, které jsou zachyceny polem kamer. Oproti tomu pro obrázky z fotoaparátu Lytro, které mají disparitu mezi snímky menší než jeden pixel, je optimální hodnota posunu nula, tím pádem je pro ně tento krok irelevantní. Je zřejmé, že tato pre-processingová operace nedokáže spasit každý špatně komprimovatelný snímek. Pouze upraví existující data do podoby, která je lépe komprimovatelná blokovou metodou.

Posun neřeší volbu jeho hodnoty. Tu je potřeba před kompresí zjistit, liší se pro každý obrázek. Bylo proto navrženo použití golden-section search pro vyhledání minima absolutní chyby mezi sousedními pohledy. Experimentálně bylo zjištěno, že absolutní chyba mezi sousedními snímky koreluje s výsledným datovým tokem. Platí tak, že menší chyba vede na menší datový tok. Golden-section search nalezne pouze jedno lokální minimum. Experimenty ukázaly, že pokud má obrázek více lokálních minim, tak je s nimi dosaženo srovnatelných datových toků. Vyhledávání je implementováno na rozsahu posunů od záporné do kladné maximální zvolené hodnoty velikosti bloku v úhlové doméně, například $\langle -8; 8 \rangle$ pro bloky o velikosti $4 \times 8 \times 16 \times 16$. Experimenty také ukázaly, že měření chyby mezi pohledy metodou Monte Carlo

dokáže zásadně redukovat čas vyhledávání optimálního posunu bez zásadního vlivu na výsledek.

5. Závěr

Tato práce se věnovala zkoumání různých metod, pomocí kterých by bylo možné vylepšit kompresi 4D světelných polí.

Byl navržen a implementován robustní algoritmus pro směrovou intra predikci pro libovolně dimenzionální bloky. Tento algoritmus se dokáže vypořádat s řadou okrajových případů. Reálnou dostupnost dat algoritmus neřeší, ale dává možnost uživateli volit strategii doplnění dat nedostupných. Tento algoritmus lze zavolat nad libovolným n-dimenzionálním směrovým vektorem pro predikci n-dimenzionálního bloku.

Navržený predikční algoritmus neřeší volbu optimálního směrového vektoru. S rostoucím počtem dimenzí exponenciálně roste i počet možných vektorů. Na tuto práci je proto možné navázat návrhem optimalizační metody, která bude směrový vektor volit v rozumném čase. Experimenty ukázaly, že směrová predikce redukuje datový tok pro bloky menších rozměrů.

Byl modifikován existující kodek světelných polí, který nyní umí zpracovávat obrázek po hyperblocích obecných rozměrů. Tato modifikace umožňuje jemnější volbu velikosti bloku a tím pomáhá dosáhnout lepšího kompresního poměru. Na tuto modifikaci je možné navázat implementací dělení bloků do stromové struktury (CTU). Volba vhodných rozměrů bloků dokáže redukovat bitrate až o 30%.

Byl zkoumán vliv vzájemného posunu mezi pohledy na datový tok. Ukázalo se, že u některých obrázků lze pohledy zaostřit na nejvíce obsáhlý objekt ve scéně a dosáhnout tak redukce datového toku. Hledání optimální hodnoty posunu je prováděno pomocí golden-section vyhledávání tak, aby byla minimalizována absolutní chyba mezi pohledy. Vzájemný posun pohledů v obrázku dokáže pro jisté případy redukovat datový tok až o 45%.

Plánem pro navazující projektovou praxi je implementace dělení bloků do stromové struktury (CTU) a s tím související návrh nového entropického kodéru. V kombinaci s intra predikcí to povede na lepší dynamické přizpůsobení kodéru na kódovaná data a tím i na zvýšení kompresního výkonu.

Poděkování

Rád bych poděloval svému vedoucímu Ing. Davidu Bařinovi, Ph.D. za jeho trpělivost v dobách, kdy produktivita práce na této projektové praxi klesala na nulu.

Literatura

[1] BARINA, D., CHLUBNA, T., SOLONY, M., DLABAJA, D. a ZEMCIK, P. Evaluation of 4D Light Field Compression Methods. In: *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*. 2019.

[2] DLABAJA, D. *Ztrátová komprese plenoptických fotografií*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně. Fakulta informačních technologií.