KAPITOLA 6

BLOKOVÝ KÓD (BTC)

Tento spôsob kódovania vedie vždy ku kompresii so stratou, pri obraze často nazývanou degradáciou. Ide o veľmi jednoduchý princíp kódovania pomocou stredných hodnôt [71]. Ukážeme si ho na príklade dvojrozmerného poľa. Pole sa rozdelí na rovnaké, či nerovnako veľké bloky a každý blok sa spracuje takto:

1. vypočíta sa stredná hodnota bloku,
2. čísla v bloku sa rozdelia na tie, ktorých veľkosť je väčšia, nanajvýš sa rovná strednej hodnote bloku a na tie, ktorých veľkosť je menšia ako stredná hodnota bloku,
3. vytvorí sa bitová maska, v ktorej sa na miestach čísel väčších alebo rovnajúcich sa strednej hodnote bloku, sa umiestnia jednotky a na ostatných miestach nuly,
4. ku každému bloku sa predradia stredné hodnoty oboch podmnožín poľa, ktoré môžeme ešte pred tým rôzne kvantovať.

Lepšie je to azda vidno na príklade s jednoduchým unitárnym kvantovaním stredných hodnôt.

vstupný blok

107	100	12	10
3	150	127	144
86	55	23	102
122	33	78	112

$$P = (107+100+12+10+3+150+127+....+112)/16=1264/16=79,$$

ďalej bude označovaná aj ako prahová hodnota P,

bitová maska

1	1	0	0
0	1	1	1
1	0	0	1
1	0	0	1

stredná hodnota množiny čísel väčších alebo rovnajúcich sa strednej hodnote poľa

$$EV = (107+100+150+127+144+86+102+122+112)/9=1050/9=116,$$

ďalej bude označovaná aj ako rekonštrukčná úroveň A_2,

stredná hodnota množiny čísel menších ako stredná hodnota poľa

$$EM = (12+10+3+55+23+33+78)/7=214/7=30,$$

ďalej bude označovaná aj ako rekonštrukčná úroveň A_1.

Dekódovanie sa robí takto: miestam, kde je v maske jednotka, sa priradí EV a miestam, kde je v maske nula, sa priradí EM. Po dekódovaní teda pole bude vyzerať takto

116	116	30	30
116	116	116	116
116	30	30	116
116	30	30	116

Vidíme, že došlo k degradácii poľa, čo však v mnohých prípadoch, najmä pri kódovaní obrazu, nevedie k badateľnému zhoršeniu jeho kvality. Je možné tieto hodnoty ešte kvantovať iným kvantizačným krokom, než bol kvantizačný krok pôvodného poľa, čo pri vhodnej voľbe kvantizačných úrovní, v závislosti od štatistického výskytu v príslušnom poli, vedie k zvýšeniu kvality dekódovaného poľa pri zachovaní kompresného pomeru. V spolupráci s menením tvaru a veľkosti blokov je táto metóda využívaná i v ATM sieťach [69]. Rôzne modifikácie BTC je možné nájsť v literatúre pod názvom Adaptívne BTC [72], AMBTC [71] a pod. Ešte si ukážeme, aká je pre náš prípad pôvodná bitová náročnosť a bitová náročnosť po kódovaní len týmto postupom, bez využitia ďalších metód. Pôvodné pole, ako vidno z hodnôt, malo bitovú náročnosť 8 bitov na jeden prvok poľa. Kódované pole potrebuje 1 bit na prvok poľa pre bitovú mapu a (2.8)/16 = 1 bit na prvok poľa pre stredné hodnoty. Teda výsledná bitová náročnosť je 2 bity na prvok poľa.

V mnohých metódach sa pre ďalšie znižovanie bitovej náročnosti používa také usporiadanie, v ktorom výsledkom kódovania celého poľa sú tri polia:

1. bitová maska, zložená z bitových masiek všetkých blokov,
2. pole stredných hodnôt EV zo všetkých blokov,
3. pole stredných hodnôt EM zo všetkých blokov.

Robí sa to preto, lebo sa predpokladá, že tieto polia majú potom vysokú štatistickú závislosť svojich vlastných prvkov, prípadne prvky majú veľmi podobné hodnoty, čo je často predpokladom úspešnosti iných kódovacích postupov. Neskôr si ukážeme, ako sa tento spôsob kódovania dá použiť v hybridných štruktúrach v spolupráci s inými druhmi kódovania.

6.1 IBTC - n

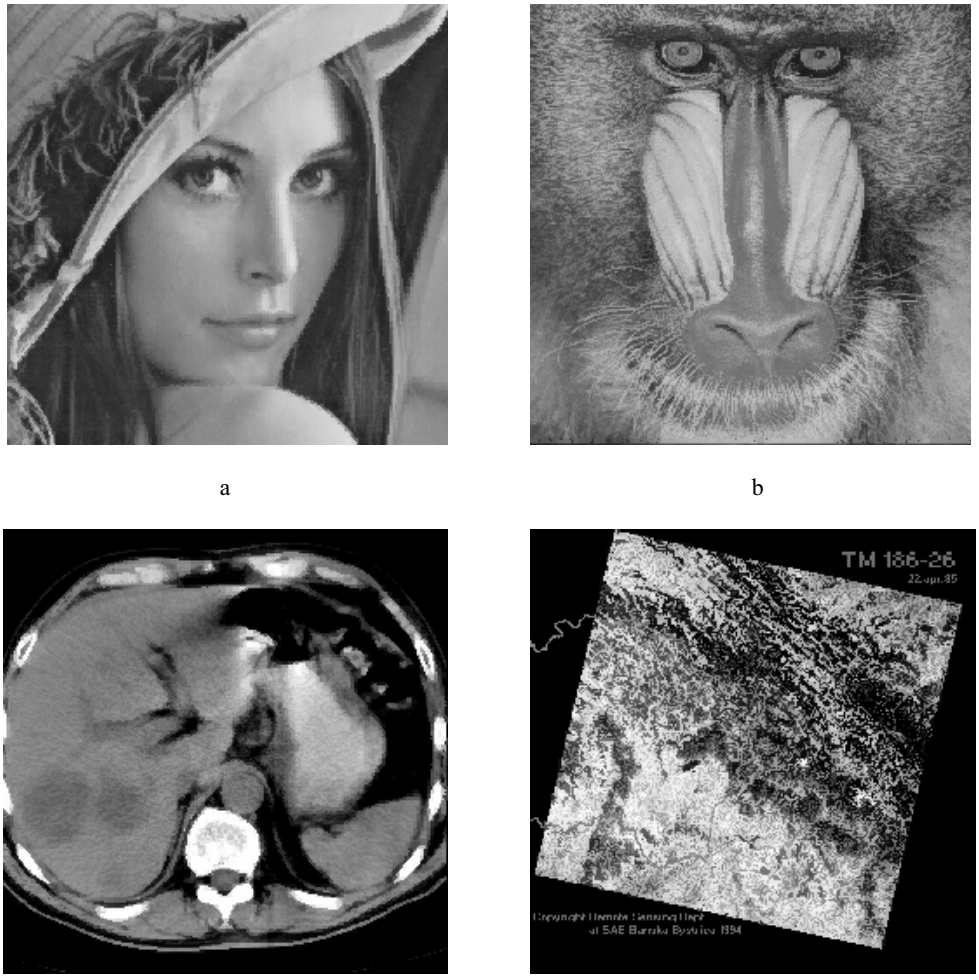
Ďalším vylepšením metódy blokového kódovania je interpolatívne kódovanie, známe pod skratkou IBTC-n [72], [73] kde n znamená stupeň interpolácie. V najjednoduchšom prípade sa pole predspracuje ako pri najjednoduchšom BTC, t.j., zistí sa EV a EM (alebo A_1 a A_2) a urobí sa maska. Podľa stupňa iterácie sa však maska neprenáša celá. Obyčajne sa to rieši nasledovne. Miesta v maske označme pomocou dvoch symbolov

x	y	x	y
y	x	y	x
x	y	x	y
y	x	y	x

Pomocou tohoto označenia môžeme zaviesť IBTC-1, keď sa z masky prenášajú len miesta x podľa predchádzajúcej masky. Iným označením môžeme interpoláciu postupne zovšeobecňovať.

$$
\begin{array}{ccccc}
x & y & x & y & x \\
y & z & y & z & y \\
x & y & x & y & x \\
y & z & y & z & y \\
x & y & x & y & x
\end{array}
$$

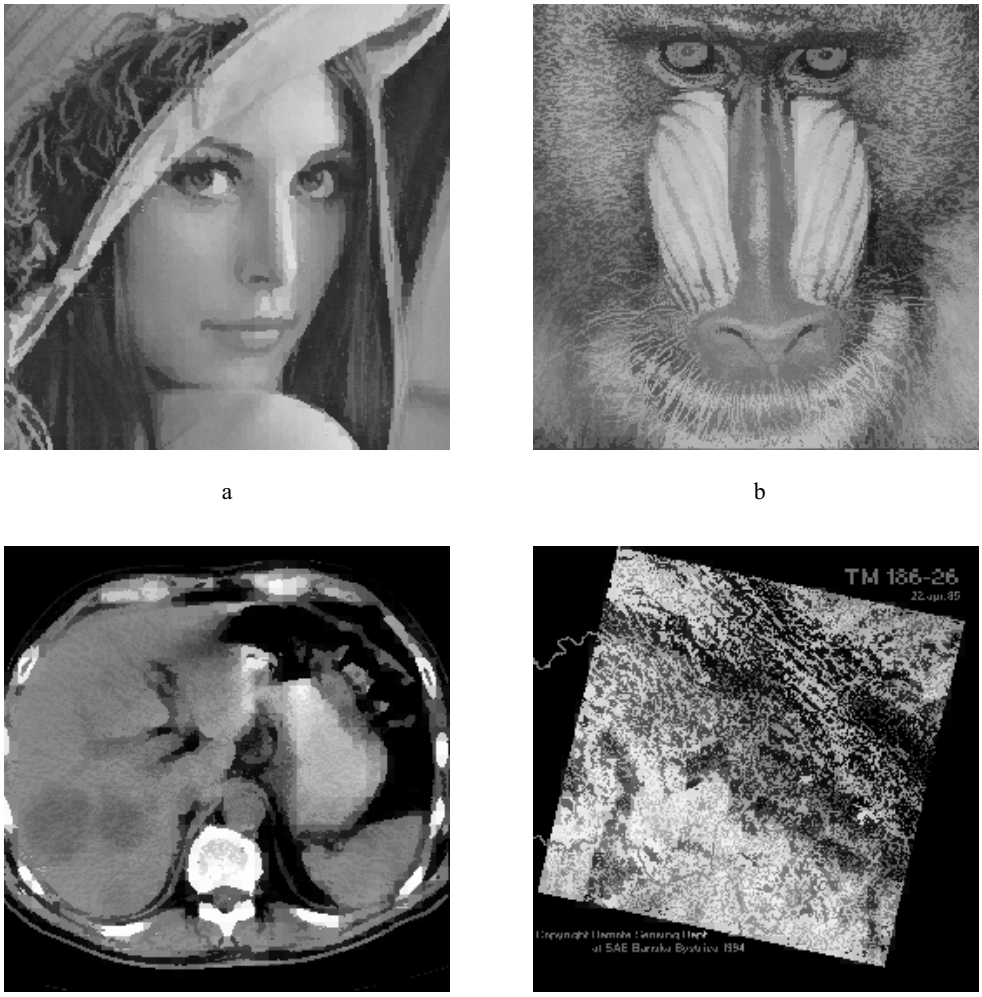
Ide o IBTC-2, keď sa z masky podľa tohoto návodu prenášajú len x.



a b

c d

Obr. 6.1 Rekonštruované obrazy kódované BTC po blokoch 4 x 4: výrez obrazu Lena (a), baboon (b), pečeň (c), aerokozmická snímka (d)



a b

c d

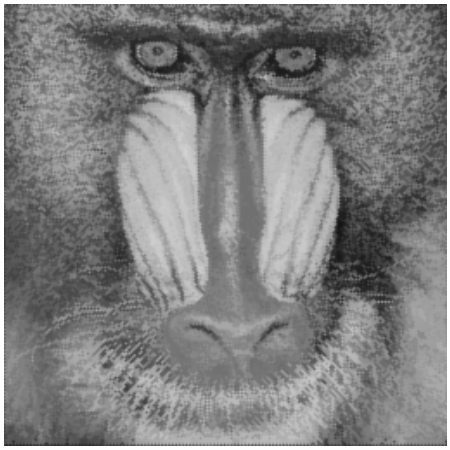
Obr. 6.2 Rekonštruované obrazy kódované BTC po blokoch 4 x 4: výrez obrazu Lena (a), baboon (b), pečeň (c), aerokozmická snímka (d)

Schéma sa dá veľmi jednoducho rozšíriť na IBTC - n. Záleží len na dohode. Tu je dôležitejší spôsob rekonštrukcie. Preneseným nulám a jednotkám masky sa v dekodéri priradia EV a EM. Ostatné prázdne miesta sa vypočítajú niektorou aproximačnou alebo interpolačnou metódou. Najjednoduchšie je urobiť strednú hodnotu zo susedných vyplnených miest [40]. Oveľa korektnejšie je však použitie zložitejšej interpolácie, či už polynomickej [40], goniometrickej [40], splajnovej [76], exponenciálnej, bilineárnej [40], mediánovej [72], [73] a pod. Záleží aj na stupni interpolácie, čo sa týka počtu použitých susedných koeficientov. Je jasné, že táto metóda je účinnejšia, pokiaľ ide o stupeň kompresie, ale náročnejšia na dekódovanie. Dá sa zaradiť, podobne ako BTC, aj do zložitejších štruktúr typu AMBTC a pod..

Na obr. 6.3 a obr. 6.4 sú uvedené výsledky z kódovania BTC pre rôzne typy obrazov a kódovania IBTC-1 a IBTC -2 s mediánovou aproximáciou s maskou 3 x 3.

a

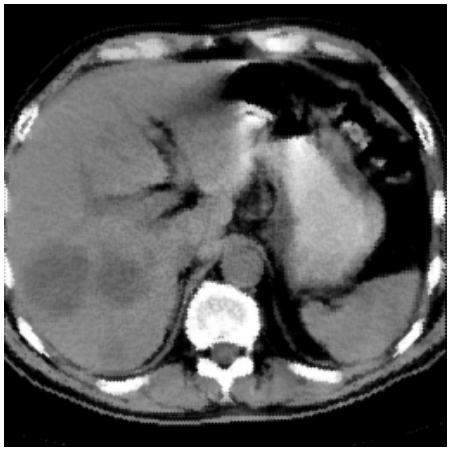
b



c

d

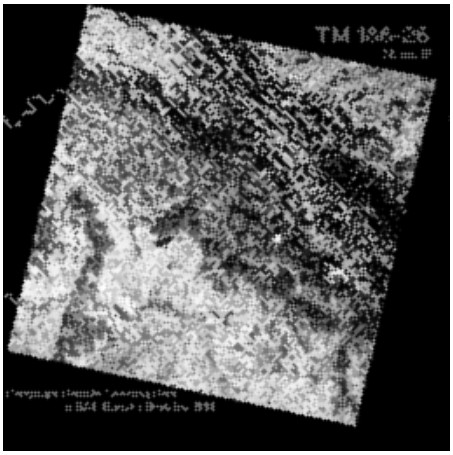
Obr. 6.3 Rekonštruované obrazy kódované IBTC po blokoch 4 x 4: výrez obrazu Lena IBTC - 1 (a), výrez obrazu Lena IBTC - 2 (b), baboon IBTC - 1 (c), baboon IBTC - 2 (d)



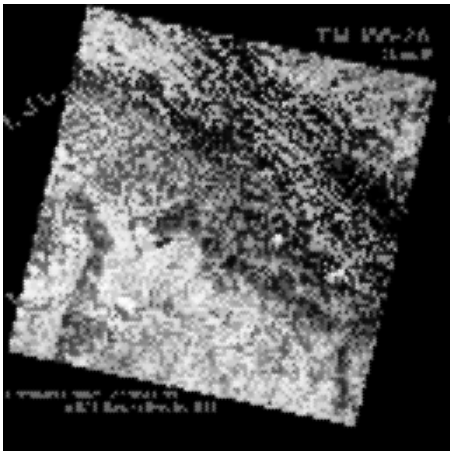
a

b

Obr. 6.4 a, b



c



d

Obr. 6.4 Rekonštruované obrazy kódované IBTC po blokoch 4 x 4: pečeň IBTC - 1 (a), pečeň IBTC - 2 (b), aerokozmická snímka IBTC - 1 (c), aerokozmická snímka IBTC - 2 (d)

6.1.1 Používané metódy interpolácie

Mediánový filter

Mediánová filtrácia je jednoduchá a pomerne účinná metóda pre filtrovanie zašumenných obrazov, používaná najmä pre filtráciu impulzného šumu zo signálov. Je tiež vhodná ako interpolačná metóda pre IBTC-1 alebo IBTC-2. Mediánové filtre majú naviac veľmi jednoduché algoritmy a dobre zachovávajú hrany. Pre bloky veľkostí 2 x 2 a 4 x 4 používame filter navrhnutý v [69]. Ďalej používame mediánové filter typu max and maxmin medián podľa [20]. Pre bloky väčšie ako 4x4 sa výrazne zväčšuje chyba interpolácie, preto sa nepožívajú na inerpoláciu obrazu. V prípade IBTC-2, medián sa používa v dvoch krokoch pre postupný výpočet rekonštruovaného obrazu.

Pre IBTC-1 je mediánový filter 2 x 2 definovaný [69]

$$y = med\left(x_1, x_2, x_3, x_4, \tfrac{1}{4}\cdot(x_1 + x_2 + x_3 + x_4)\right), \tag{6.1}$$

kde

$$\begin{array}{ccc} & x_1 & \\ x_2 & y & x_3 \\ & x_4 & \end{array} \tag{6.2}$$

a y je úroveň šedej interpolovaných pixelov a x_i sú susedné pixely, ktorých úroveň šedej poznáme. Medián je 0,5-kvantil náhodnej premennej a mediánový filter sa definuje ako stredné číslo z postupnosti hodnôt úrovní šedej a ich strednej hodnoty po usporiadaní podľa veľkosti.

Mediánový filter 4 x 4 sa definuje [69]

$$y = med\left(\{x_i\}_{i=1}^{16}, \tfrac{1}{16}\cdot\sum_{i=1}^{16} x_i\right), \tag{6.3}$$

kde

$$\begin{array}{ccccccc} & & & x_1 & & & \\ & & x_2 & . & x_5 & & \\ & x_3 & . & x_6 & . & x_9 & \\ x_4 & . & x_7 & y & x_{10} & . & x_{13} \\ & x_8 & . & x_{11} & . & x_{14} & \\ & & x_{12} & . & x_{15} & & \\ & & & x_{16} & & & \end{array} \tag{6.4}$$

y je úroveň šedej interpolovaných pixelov a x_i sú susedné pixely, ktorých úroveň šedej poznáme

Maxminmediánový filter sa definuje [20]

$$y = y_1 \text{ ak } |y_1 - y_0| \le |y_2 - y_0| \,, \qquad (6.5)$$

$$y_2 \text{ ak } |y_1 - y_0| > |y_2 - y_0| \,.$$

kde $y_0 = med\left(\{x_i\}_{i=1}^{16}, \frac{1}{16} \cdot \sum_{i=1}^{16} x_i \right)$, $y_2 = \max(z_1, z_2, z_3, z_4)$, $y_1 = \min(z_1, z_2, z_3, z_4)$

a

$$z_1 = med\left(x_1, x_2, x_3, x_4, \tfrac{1}{4} \cdot (x_1 + x_2 + x_3 + x_4) \right) ,$$

$$z_2 = med\left(x_5, x_6, x_7, x_8, \tfrac{1}{4} \cdot (x_5 + x_6 + x_7 + x_8) \right) ,$$

$$z_3 = med\left(x_9, x_{10}, x_{11}, x_{12}, \tfrac{1}{4} \cdot (x_9 + x_{10} + x_{11} + x_{12}) \right) ,$$

$$z_4 = med\left(x_{13}, x_{14}, x_{15}, x_{16}, \tfrac{1}{4} \cdot (x_{13} + x_{14} + x_{15} + x_{16}) \right) .$$

kde y je úroveň jasu interpolovaných pixelov.

Tento zložitejší typ mediánu sa používa pre svoju štatisticky vyššiu efektivitu pre zachovanie hrán v istých smeroch oproti klasickému mediánu (dôkaz [192]), a preto ako filter vykazuje lepšia výsledky pre hrany a rohy. Cenou je výrazné spomalenie algoritmu (rozhodovanie v každom kroku) a potreba počítať päť mediánov namiesto jedného na výpočet každého pixela. V prípade mediánovej filtrácie pomocou algoritmu "running median" [188] to nie je veľké spomalenie, ale pre interpolačnú modifikáciu mediánového filtra sa časová náročnosť stáva prekážkou použitia takéhoto spôsobu interpolácie v praxi. Podobný problém nastáva aj pri použití maxmediánového filtra [192],definovaného

$$y = \max(z_1, z_2, z_3, z_4) \,, \qquad (6.6)$$

kde

$$z_1 = med\left(x_1, x_2, x_3, x_4, \tfrac{1}{4} \cdot (x_1 + x_2 + x_3 + x_4) \right) ,$$

$$z_2 = med\left(x_5, x_6, x_7, x_8, \tfrac{1}{4} \cdot (x_5 + x_6 + x_7 + x_8) \right) ,$$

$$z_3 = med\left(x_9, x_{10}, x_{11}, x_{12}, \tfrac{1}{4} \cdot (x_9 + x_{10} + x_{11} + x_{12}) \right) ,$$

$$z_4 = med\left(x_{13}, x_{14}, x_{15}, x_{16}, \tfrac{1}{4} \cdot (x_{13} + x_{14} + x_{15} + x_{16}) \right) .$$

Meanový filter

Algoritmy pre meanovú interpoláciu sú podobné ako pre medián, ale y sa počíta ako aritmetický priemer susedných pixelov (6.2), resp.(6.4).

Interpolácia Lagrangeovými polynómami

Tento spôsob interpolácie je aplikáciou všeobecne známeho matematického postupu interpolácie pomocou Lagrangeových polynómov. Ak máme n hodnôt, vieme nájsť polynóm stupňa n-1, ktorý prechádza týmito hodnotami a interpoluje danú funkciu. Vo všeobecnosti je interpolačný operátor

$$L[f(x)] = f(x) + \sum_{j=1}^{n} \sum_{i=0}^{m} A_{ij}(x) \cdot f^{(i)}(a_{ij}) \,, \qquad (6.7)$$

kde A_{ij} sú polymómy a a_{ij} sú body, v ktorých poznáme hodnotu interpolovanej funkcie. Pre Lagrangeovu interpoláciu je m=0 (nepoužíva deriváciu ani diferenciu, len funkčné hodnoty). N je počet vzoriek (bodov), z ktorých interpolujeme. Obvyklé je v prípade m=0 označovať A_{0j} = - l_j . Pre body a_j požadujeme

$$L[f(a_j)] = 0 \,. \tag{6.8}$$

Z toho

$$y(x) = \sum_{j=1}^{n} l_j(x) \cdot f(a_j) \,, \tag{6.9}$$

je výsledná funkcia a

$$f(x) = y(x) + E(x) \,. \tag{6.10}$$

Pre body $x = a_i$ chceme, aby $E(x) = 0$. Preto musí byť

$$l_j(a_k) = \delta_{jk} \tag{6.11}$$

a Lagrangeov interpolačný vzorec môžeme vyjadriť v tvare

$$l_j(x) = \frac{(x-a_1)(x-a_2)\cdot .....\cdot (x-a_{j-1})(x-a_{j+1})\cdot .....\cdot (x-a_n)}{(a_j-a_1)(a_j-a_2)\cdot .....\cdot (a_j-a_{j-1})(a_j-a_{j+1})\cdot .....\cdot (a_j-a_n)} \,. \tag{6.12}$$

Teda pre m bodov je interpolovaná funkcia

$$y_L(x) = \sum_{j=0}^{m-1} l_{j,m}(x) \cdot f_j(x) \,, \tag{6.13}$$

kde

$$l_{j,m} = \prod_{i=0, i\neq j}^{m-1} \frac{(x-x_j)}{(x_i-x_j)} \,.$$

Interpolácia trigonometrickými polynómami

Je tiež často používanou interpolačnou metódou, no jej algoritmus je implementovaný pomocou diskrétnych ortogonálnych transformácií z triedy trigonometrických transformácií s interpolačnými vlastnosťami, preto ho v tejto časti nebudeme opisovať. Čitateľa odkazujeme na kapitolu 2, časť o interpolácii pomocou DOT. Výhodou tohto spôsobu je možnosť rýchlej realizácie interpolácie pomocou dobre prepracovaných algoritmov DOT.

Interpolácia pomocou konvolúcie

V prípade interpolácie na IBTC-2, môžeme použiť interpoláciu metódou konvolúcie podľa [191]. Obraz po podvzorkovaní obrazu doplníme striedavo nulami a po konvolúcii s interpolačným jadrom získame obraz. v ktorom na miestach núl budú interpolované hodnoty podľa metódy v závislosti od zvoleného konvolučného jadra. Používané interpolačné konvolučné jadrá sú:

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \text{- peg} \qquad \frac{1}{4} \cdot \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \text{- pyramidálne} \tag{6.14}$$

$$\frac{1}{9}\cdot\begin{pmatrix}1&3&3&1\\3&9&9&3\\3&9&9&3\\1&3&3&1\end{pmatrix}$$ - zvon $$\frac{1}{64}\cdot\begin{pmatrix}1&4&16&4&1\\4&10&24&10&4\\16&24&64&24&16\\4&10&24&10&4\\1&4&16&4&1\end{pmatrix}$$ - splajnové

Výsledkom konvolúcie s jadrom "peg" je interpolácia typu "zopakuj hodnotu suseda", pyramidálne jadro má za výsledok bilineárnu interpoláciu, teda v podstate meanový interpolačný filter, zvonové jadro rezultuje interpoláciou pomocou zvonovej funkcie, s ktorou váhujeme hodnoty susedných pixelov, a splajnové jadro vedie na interpoláciu B-splajnovými krivkami [18], pričom algoritmus interpolácie je rýchlejší ako pre všeobecne počítané splajnové funkcie pri zachovaní takmer identických hodnôt chýb interpolácie.

6.2 NÁVRH KVANTIZÁTORA

V uvedenom príklade bol kvantizátor navrhnutý na základe lokálnych priemerných hodnôt. Ako uvidíme ďalej, kvantizátor zachováva prvý aj druhý moment bloku, a tiež minimalizuje strednú kvadratickú odchýlku, ak obmedzenie jasu je dané priemerom bloku [71]. Keďže kvantizátor je definovaný s určitým obmedzením (napr. priemerom bloku) a s dvoma rekonštrukčnými úrovňami, máme tri možnosti ako modifikovať základný návrh kvantizátora.. Môžeme však využiť aj ďalšie kritériá a potom obmedzenie (prahová hodnota) nemusí byť striktne dané priemerom úrovní bloku.

6.2.1 Kvantizátory zachovávajúce momenty

Ako už vyplýva z názvu, ide o kvantizátory, ktoré zachovávajú limitovaný počet momentov údajov v bloku (môže to byť napr. stredný aritmetický priemer bloku alebo rozptyl) [71]. Zachovanie týchto momentov nám vlastne umožní zachovať dôležité vizuálne informácie o obraze.

Jeden z najjednoduchších kvantizátorov bol navrhnutý tak, že zachováva priemernú hodnotu v bloku a rozptyl. Obraz sa najprv rozdelí na neprekrývajúce sa bloky N x N. Nech $M = N.N$ a nech x_1, x_2, ..., x_M sú hodnoty jasu jednotlivých bodov v danom bloku pôvodného obrazu. Kvantizátor má zachovať prvý a druhý štatistický

Prvý štatistický moment

$$\bar{x} = \frac{1}{M}\sum_{i=1}^{M} x_i \ . \tag{6.15}$$

Druhý štatistický moment

$$^2\bar{x} \ = \frac{1}{M}\sum_{i=1}^{M} x_i^2 \ . \tag{6.16}$$

Na základe týchto dvoch hodnôt vieme vypočítať rozptyl σ^2 [71], [19]:

$$\sigma^2 = ^2\bar{x} - \bar{x}^2 \ . \tag{6.17}$$

Takže tým, že dokážeme zachovať hodnotu rozptylu, dokážeme zároveň za určitách podmienok zachovať prvý a druhý vzorkovací moment.

Ďalším krokom je nájdenie prahovej hodnoty P a rekonštrukčné úrovne tak, aby

$$x_i' = \begin{cases} A_1, \text{ ak } x_i < P \\ A_2, \text{ ak } x_i \geq P \end{cases} \quad \text{pre } i = 1, 2, ..., M . \tag{6.18}$$

Celá analýza sa zjednoduší, keď zachovanie vzorkovacích momentov zabezpečíme dvoma rekonštrukčnými úrovňami (prahová úroveň bude nastavená na $\bar{x}$). Nech q je počet čísel z x_i, ktoré sú väčšie alebo rovné prahovej hodnote $\bar{x}$. Aby sme dokázali zachovať prvý a druhý štatistický moment, musia platiť tieto rovnice:

$$M\bar{x} = (M-q)A_1 + qA_2 , \tag{6.19}$$

$$M \cdot^2 \bar{x} = (M-q)A_1^2 + qA_2^2 . \tag{6.20}$$

Úpravou týchto rovníc dostávame [71], [19]:

$$A_1 = \bar{x} - \sigma\sqrt{\frac{q}{M-q}} , \tag{6.21}$$

$$A_2 = \bar{x} + \sigma\sqrt{\frac{M-q}{q}} . \tag{6.22}$$

Prenášame obe rekonštrukčné úrovne A_1 a A_2 a bitovú masku. Je však možné vyslať namiesto A_1 a A_2 hodnoty $\bar{x}$ a σ^2 a rekonštrukčné úrovne sa vypočítajú. Výhoda tohto kroku spočíva v tom, že hodnoty $\bar{x}$ a σ^2 môžu byť zakódované použitím menšieho počtu bitov ako A_1 a A_2. Rekonštrukcia v prijímači sa uskutoční indikovaním, ktoré body bitovej masky sú rekonštruované ako A_1, a ktoré ako A_2. Nevýhodou tohto kvantizátora je ale skutočnosť, že pri výpočtoch používa matematické operácie umocňovania a odmocňovania, čo spôsobuje problémy pri hardvérovej realizácii. Preto boli vyvinuté náhrady tohto kvantizátora, ktoré nepoužívajú tieto matematické operácie, a ktoré zachovávajú absolútne momenty.

Priemer a prvý absolútny centrálnymoment bloku N x N sú dané [71]:

$$\bar{x} = \frac{1}{M}\sum_{i=1}^{M} x_i , \tag{6.23}$$

$$\alpha = \frac{1}{M}\sum_{i=1}^{M} |x_i - \bar{x}| . \tag{6.24}$$

Ak opäť obmedzíme prahovú hodnotu kvantizátora na $\bar{x}$, tak rekonštrukčné úrovne, potrebné na zachovanie $\bar{x}$ a α sú [71]:

$$A_1 = \bar{x} - \frac{M\alpha}{2(M-q)} , \tag{6.25}$$

$$A_2 = \bar{x} + \frac{M\alpha}{2q} . \tag{6.26}$$

kde q má ten istý význam ako predtým.

Rekonštrukčné úrovne A_1 a A_2 môžu byť opäť zakódované priamo alebo budeme kódovať $\bar{x}$ a α.

6.2.2 Kvantizátory minimalizujúce chyby

Keď prahová hodnota kvantizátorov, ktoré uchovávajú absolútny moment, je obmedzená na priemer, očakáva sa, že MSE kvantizátora bude minimálna [71].

Pre MSE jedného bloku platí

$$e_{MSE} = \frac{1}{M}\left[\sum_{x_i<\bar{x}} (x_i - A_1)^2 + \sum_{x_i\geq\bar{x}} (x_i - A_2)^2\right]. \tag{6.27}$$

Ak budeme derivovať tento výraz podľa A_1, a potom ho dáme rovnajúci sa nule, pre rekonštrukčnú úroveň A_1 [71] dostaneme

$$A_1 = \frac{1}{M-q}\sum_{x_i<\bar{x}} x_i\,. \tag{6.28}$$

Analogickým postupom dostaneme výraz pre výpočet hodnoty rekonštrukčnej úrovne A_2

$$A_2 = \frac{1}{q}\sum_{x_i\geq\bar{x}} x_i\,. \tag{6.29}$$

Takto máme navrhnuté dve rekonštrukčné úrovne, ktoré budeme prenášať spolu s bitovou mapou (maskou).

6.3 VYNECHANIE BITOVEJ MASKY

Ide o veľmi jednoduchú techniku [71] redukcie bitovej náročnosti. Používa sa vtedy, ak zmena blokov obrazu je malá alebo stále rovnaká, a ak rekonštrukčné úrovne A a B sa líšia len malou hodnotou. V takomto prípade sa prenáša len hodnota $\bar{x}$, ktorá bude rekonštrukčnou úrovňou pre celý blok. Priemerná bitová náročnosť, ktorú dosahujeme touto metódou, je veľmi závislá od kódovaného obrazu. Ak kódujeme obraz s veľkými plochami s približne rovnakou úrovňou, dosiahneme významnú redukciu bitovej náročnosti pri veľmi malej strate kvality rekonštruovaného obrazu oproti základnej metóde BTC.

6.4 NEZÁVISLÉ A ZÁVISLÉ BITY

Princíp tejto modifikácie BTC spočíva v tom, že celé lokality v bitovej maske označíme ako nezávislé bity a iné ako závislé [71]. Prenášať sa budú len nezávislé bity a závislé bity sa určia v prijímači na základe nezávislých bitov, použitím preddefinovaného postupu. Najdôležitejšou časťou celého kódovania je výber závislých a nezávislých bitov. Výsledná kvalita závisí aj od konkrétneho obrazu.

6.5 ADAPTÍVNA VEĽKOSŤ BLOKU

Stanovenie rovnakej veľkosti blokov pre celý obraz nie je jednoznačným vyjadrením jeho charakteru. Aby sme dosiahli vysokú presnosť rekonštruovaného obrazu, mala by byť veľkosť spracovávaných blokov čo najmenšia. Na druhej strane však dochádza k veľmi malej redukcii bitovej

náročnosti. Preto sa zdá byť zaujímavým rozdelenie obrazu na nerovnako veľké bloky, ktoré by boli vyjadrením charakteru obrazu. Napr. veľké plochy s približne rovnakým jasom a malými zmenami spracujeme vo veľkých blokoch a plochy s množstvom jasových zmien spracujeme v malých blokoch.

Aj keď tento spôsob často nevedie k významnému zvýšeniu redukcie dát, ale kvalita rekonštruovaného obrazu môže byť výrazne lepšia.

6.6 BTC S BEZSTRATOVÝM KÓDOM

Výstupom rôznych modifikácií BTC sú: bitová maska a rekonštrukčné úrovne. Je samozrejmé, že tieto nemusia byť ešte úplne redukované, ak sa použijú kódy s konštantnou dĺžkou kódového slova. Preto sa odporúča tieto dva súbory dát kódovať bezstratovými kompresnými kódmi. Pre kódovanie masky sa zdá najvýhodnejšie použitie niektorého z triedy Zivovych-Lempelovych kódov a pre kódovanie rekonštrukčných úrovní niektorý z entropických kódov. Nie je to však všeobecne platné, preto je výhodnejšie urobiť výber metódy pre určitú triedu obrazov, a potom ho pre obrazy z tejto triedy v praxi používať.

Ako príklad uvádzame, že výrez Leny 256 x 256, kódovaný BTC po blokoch 4 x 4, ktorý má 2 bity/bod, má po ďalšom kódovaní komerčným Zivovym-Lempelovym kódom gzip už len 1,85 bitu/bod. Pri kódovaní IBTC-1 po blokoch 4 x 4 BTC sa zníži bitová náročnosť z 1,5 bitu/bod na 1,4 bitu/bod. Oveľa účinnejšie je toto kódovanie pre tomografický snímok pečene 256 x 256: BTC 4 x 4 - z 2 bitov/bod na 1,5 bitu/bod a IBTC-1 4 x 4 - z 1,5 bitu/bod na 1,2 bitu/bod.

Príklady

6.1 Zvoľte maticu 32 x 32 8-bitových čísel, rozdeľte ju na bloky

1. 4 x 4 a spracujte pomocou BTC,
2. 8 x 8 a spracujte pomocou BTC,
3. 16 x 16 a spracujte pomocou BTC,
4. 32 x 32 a spracujte pomocou BTC.
5. 4 x 4 a spracujte pomocou IBTC - 1, pri rekonštrukcii použite výpočet chýbajúcich čísel pomocou strednej hodnoty prvých susedných prenesených čísel. Vyhodnoťte bitové náročnosti a strednú kvadratickú odchýlku rekonštruovaných polí od originálu v závislosti od veľkosti spracovaného bloku.

6.2 Pre matice z príkladu 6.1 meňte počet kvantizačných úrovní pred kódovaním BTC a pri kódovaní BTC. Pre obe zmeny vyhodnoťte bitovú náročnosť a strednú kvadratickú odchýlku dekódovaného pľa od originálu. Tento istý príklad riešte pre návrh kvantizátora, zachovávajúceho moment a kvantizátora, minimalizujúceho chyby.

6.3 Pre bloky 4 x 4 z príkladu 6.1 vytvorte 3 súbory: bitovú masku, EV a EM a posledné dve podrobte dvojrozmernej DCT. Opíšte výsledok.