DIPLOMAMUNKA

Pesti Kálmán

Debrecen
2008

Debreceni Egyetem
Informatika Kar

A H.264 eszközkészlete és előnyei a korábbi videó tömörítési eljárásokkal szemben

Témavezető:
Dr. Hajdú András
Egyetemi adjunktus

Készítette:
Pesti Kálmán
Programtervező matematikus

Debrecen
2008

1. Tartalomjegyzék

2. Bevezetés

A H.264 elnevezésű videó tömörítési technológia fejlesztése 2002. márciusában fejeződött be, viszont csak 2003. májusában vált elérhetővé az első végleges változat. Ez csak az első megjelenést jelenti, a fejlesztés azóta is gőzerővel folyik. Az eszközök tökéletesítésén, új elemek implementálásán, valamint a tömörítés elemeinek optimalizálásán, hatékonyabbá tételén napjainkban is számos cég dolgozik.

Az új videótömörítési technológiát a Joint Video Team (JVT) fejlesztette ki, amely az International Telecommunication Union (ITU) alá tartozó Video Coding Experts Group, valamint az International Organization for Standards/International Electrotechnical Commission (ISO/IEC) alá tartozó Moving Pictures Experts Group szervezetek közös csoportja.

Már a megjelenésekor az új videótömörítési eljárás valamivel 1 Mbit/s adatátviteli sebesség alatt képes volt DVD-közeli minőségű videó továbbítására. Bár ez a sávszélesség továbbra is elérhetetlen a világháló felhasználóinak döntő többsége számára, a technológia alacsonyabb bitráta esetén is jelentős minőségbeli javulást hoz az internetes videósugárzásban.

Az ITU által Recommendation H.264 elnevezéssel illetett tömörítési technológia az ISO köreiben az ISO/IEC 14496 10 Advanced Video Coding (AVC) nevet kapta. Legtöbbször MPEG-4 AVC vagy csak AVC néven hivatkoznak rá.

A diplomamunkám elsődleges célja a H.264 videó tömörítési eljárás eszközkészletének bemutatása. Először röviden ismertetem a korábbi technológiákat, mint például az MPEG1, MPEG2, MPEG4, H.261 és H.263, ezek előnyit az őket megelőző eljárásokkal szemben, és természetesen hátrányaikat is, amiért egy újabb technológiára volt szükség helyettük. Ezután általánosságban írok a H.264-ről, milyen célokkal jött létre, milyen újdonságokat hozott és miért is olyan népszerű. Majd az eszközkészletét mutatom be, különösen az új elemekről írok majd részletesebben. Szót ejtek továbbá még a H.264 eddig megjelent verzióról, és milyen változtatások, bővítések várhatóak a közeljövőben. Áttekintést nyújtok a széleskörű alkalmazási területeiről, és a további lehetőségekről, célkitűzésekről.

Választásom többek között azért esett erre a témára, mert amikor elkezdtem a diplomamunkám kidolgozását, a cégemnél (Infostyle Kft, Debrecen) éppen egy H.264 enkóder – dekóder páros implementációján kezdtünk el dolgozni. Az alkalmazás

megrendelésre készül, első körben általános felhasználási céllal. Ha ez elkészül majd, a megrendelőnk csak ezután dönt a továbbiakról, hogy milyen területre, konkrét felhasználásra specializáljuk a program működését.

A H.264 tömörítési eljárás mögött sok és összetett technológiák és algoritmusok állnak. Emellett pedig ez egy 5 éves, tehát viszonylag friss technológiának tekinthető, amely napjainkban is dinamikusan bővül, fejlődik.

A diplomamunkám elkészítésének során elsősorban az interneten megtalálható dokumentációkra, cikkekre és tanulmányokra támaszkodom, azok közül is túlnyomórészt angol nyelven íródottakra, ugyanis nyomtatott irodalomban igen csekély számú alkotás született még napjainkban.

A diplomamunka elkészítése során törekedtem a magyar terminológia használatára. Viszont előfordul néhány olyan angol szakkifejezés, amelynek nincs magyar megfelelője, vagy szinte egyáltalán nem használatos. Ilyen esetben az eredeti kifejezés használatát részesítettem előnyben. A **2.1** fejezet tartalmazza a terminológia gyakran előforduló eredeti, angol kifejezéseit és azok magyar megfelelőit valamint tömören a jelentésüket.

2.1 Szószedet

Bit-rate: bitráta, egységnyi idő alatt továbbítandó adatmennyiség.
Block: blokk, képpontok egy halmaza, makroblokk vagy annak felosztásával előállt egység.
Chroma: színkülönbségi jel.
Decoder: dekóder, a tömörített videó dekódolását, visszaállítását végzi.
Encoder: enkóder, a digitalizált videó anyag tömörítését végzi.
Field: mező, fél kép, egy képkocka páros vagy páratlan makroblokk sorainak halmaza.
Frame: képkocka, filmkocka.
Interlaced: váltott soros, egyfajta kódolási technika.
Inter Prediction: inter predikció, jóslás korábbi, esetleg következő képkockák alapján.
Intra Prediction: intra predikció, képen belüli jóslás.
Luma: világosság jel.
Makroblokk: 16x16 képpontból álló képrészlet.
Motion Compensation, MC: mozgás kompenzáció, mozgás vektor meghatározása.
Motion Estimation, ME: mozgásbecslés, mozgás vektor meghatározása.

Pixel: képpont.

Prediction: predikció, jóslás.

Slice: szelet, képszelet, a képkocka részhalmaza, makroblokkok halmaza.

Slice Group: képszelet csoport.

Stream: videó folyam, bitfolyam.

3. A videó tömörítés történetének áttekintése

Hogy minél jobban megérthessük a H.264 nyújtotta újdonságokat, technikai megoldásokat és ezek óriási előnyeit a korábbi videó tömörítési eljárásokkal szemben, ezért először tekintsük át vázlatosan a korábbi technológiákat!

A videó tömörítés nem a közismertebb RGB képpont ábrázoláson alapul, hanem az ún. YUV-n. Az YUV komponens jelek: *Y*: világosság jel, *U*, *V* vagy *Cr*, *Cb*: színkülönbségi jel, chroma jel. A szomszédos képpontok YUV komponenseinek értékei sokkal jobban hasonlítanak egymásra, így jobban tömöríthetőek, míg az RGB szomszédos értékek óriási eltéréseket mutatnak még közel homogén képrészletek esetén is.

Tömörítés során kihasznált legfontosabb tulajdonságok

A természetes mozgókép statisztikailag redundáns, azaz a képpont értékek adott környezeten belül, a képrészletek adott időintervallumon belül hasonlóak. Mozgóképben rejlő redundanciák:

- Térbeli: intra-frame (képen belüli), inrta-field (félképen belüli)
- Időbeli: inter-frame (képek közötti)

Az emberi látás tulajdonságai miatt, lényegtelen információk eltávolítása. Az emberi látás tulajdonságai, amiket a tömörítő eljárások ki is használnak:

- Világosságjelre a látásunk felbontó képessége 3 - 5-ször nagyobb, mint a színjelre.
- Álló és lassan változó képekre az emberi látás felbontás igénye lényegesen nagyobb, mint a gyorsan változó képszekvenciákra.
- Kevésbé érzékeny a kvantálási zajra és minden egyéb zajra, ha annak frekvenciája nagy.
- Hirtelen képváltás, vagy gyors mozgás esetén kevésbé zavaró a gyenge képminőség.
- Álló- és lassan változó képek esetén erősen érzékeny a képminőség változásra (térben és időben)
- Ellentétben a hanggal, a látásunk sokkal kisebb jel-zaj viszony esetén is jónak tartja a képet (szubjektív és objektív képminőség)
- Speciális mozgás fajták esetén drámai felbontás igény.

Most pedig nézzük át, a különböző videó kódoló eljárások hogyan használják ki ezeket a tömörítési lehetőségeket. Ebben a részben az MPEG (Motion Picture Expert Group, Mozgókép Szakértői Csoport) szabványokról lesz szó.

Az MPEG célja a videó és audió digitális, bitsebesség csökkentett formában történő hatékony ábrázolása. Ezek a szabványok nem specifikálják a kódolási algoritmusokat, csak a kódolás kimeneti adatfolyamainak bitszintaxisát.

Kódolási technikák, tulajdonságok

- Kifinomult, DCT (Discrate Cosinus Transformation, diszkrét koszinusz transzformáció) alapú mozgáskompenzációt használó hibrid kódolás.
- A kódolás és dekódolás számításigénye különböző, a rendszer tehát aszimmetrikus.
- MPEG videó kódolás egyik alapjellemzője a réteges szerkezete.
- A rétegszerkezet 6 egymásba ágyazott egységet tartalmaz, melyekben az alsóbb rétegek általában nem dekódolhatók a felsőbb szintek nélkül.
- A felső 4 egység bájthatáron kezdődő egyedi startkóddal rendelkezik, így kezdetük dekódolás nélkül megtalálható.

MPEG rétegek:

1. Szekvencia réteg: Magát a kódolt szekvenciát jelöli, fejléce tartalmazza a rendszeradatokat (képméret, bitsebesség, képfrekvencia, kvantálási mátrixok, stb.).
2. Képcsoport réteg (GOP=Group Of Pictures): Legalább 1 önmagában kódolt (intra) képet tartalmazó, bizonyos számú kép együttese, amely a véletlen hozzáférés egysége.
3. Képréteg: Egy kép kódolt adatait tartalmazza. A kódolatlan kép formátuma MPEG-1-ben 4:2:0, MPEG-2-ben 4:2:0, 4:2:2 vagy 4:4:4, struktúrájú YUV. Az MPEG-1 csak progresszív, míg az MPEG-2 alkalmas váltott-soros képek kódolására is.
4. Szelet (slice) réteg: Makroblokkok sorfolytonos csoportja, az újraszinkronizáció egység. Ez a legalsó szint, amelyen a dekóder még képes feléledni bithiba esetén.
5. Makroblokk réteg: Az *Y* 16x16-os, az *U*, *V* 8x8 (4:2:0), 8x16 (4:2:2), vagy 16x16-os (4:4:4) blokkjaiból áll, a mozgáskompenzáció egysége.
6. Blokk réteg: A makroblokk 8x8-as blokkjai, a DCT kódolás egysége.

Az MPEG 3 képtípust definiál (I: intra, P: predictive, B: bidirectionelly coded), ezekről majd a H.264-nél lesz szó részletesen. Itt csak azért említem meg, mert ezek a típusok minden MPEG szabványnál megtalálhatóak.

H.261 tulajdonságai (1990)

- Videókonferencia rendszerek kép és hangtömörítő eljárásai.
- 8x8-as DCT, együtthatók cikkcakk rendezése.
- Futamhossz kódolás.
- Mozgáskompenzáció: P típusú képek.
- Bitsebesség vezérlés k*64 kbit/s-ra.
- Csatornahiba elleni technikák.

MPEG-1 tulajdonságai (1993)

- Alacsony bitsebességű multimédiás alkalmazások (Video-CD, CD-I), kb. 1,5 Mb/s VHS képminőség, SIF felbontású kódolt videó. 4:2:0 YUV formátum.
- Csak progresszív képek kódolására alkalmas.
- Konstans bitsebességű kódolás (CBR).
- Elérhető tömörítés jó minőségű kódolás mellett 25-30.

MPEG-2 tulajdonságai (1995)

- Szintaktikailag az MPEG-1-re épül, s azzal felülről kompatibilis, tehát egy MPEG-2 dekódernek tudnia kell dekódolni egy MPEG-1 bitfolyamot.
- Az MPEG-2 kétféle képet különböztet meg: a teljes képet (frame) és a félképet (field). A frame kódolásakor a bennük lévő két félképet együtt, míg field esetén a félképeket egymástól függetlenül kezelik. Egy videó szekvencián belül a frame és field keverhető.
- A műsorszóró alkalmazások mind kettőt, a kis késleltetésű interaktív rendszerek általában a field-et használják.
- 4:2:2 és 4:4:4 YUV formátumok támogatása.
- Léptékelhető (skálázható) kódolási módok is. (SNR skálázás, térbeli skálázás)
- Profil és szint szerkezet. A profilok a szintaxist (alkalmazható képformátum, képtípus, kódolási eszközök, stb.) definiálják. Míg a szintek a paraméterek értékét (pl. képméret, bitsebesség, képsebesség, stb.) korlátozzák. A szintek a következő alkalmazásokat célozzák meg.
- Alkalmazási területek: műsorszórás (Digital Video Broadcasting, DVB), 2-8 Mb/s terjesztési minőség (distribution quality), stúdiótechnika, 18-50 Mb/s újrafeldolgozási minőség (contribution quality), Digital Versatile Disc (DVD), 3-7 Mb/s (VBR, Variable-bit-rate, változó bitsebesség) jobb, mint a PAL minőség.

H.263 tulajdonságai (1995)

- A H.261 kódolás optimalizálása, főleg az MPEG videó kódolás eszközkészletére és tapasztalataira támaszkodva.
- Fejlettebb mozgásbecslés: PB kép, félképpontos pontosság, predikció átlapolt területekkel, mozgásbecslés alapja MB helyett blokk is lehet, képről túlmutató mozgásvektor.
- Aritmetikai kódolás alkalmazása.

MPEG-4 tulajdonságai (2000)

- Alacsony bitsebességű kódolások (kiindulás x 64 kbit/s) szabványának indult (1994), de jelenleg az interaktív multimédia szolgáltatások objektum orientált szabványává vált. Népszerű implementációk: DivX, XviD.
- Szintetikus és természetes hang, beszéd, kép és videó adatok: média objektumok.
- A média objektumok kompozíciójából származik a jelenet.
- Médiaobjektumok összetartozása hierarchikus módon.
- Interaktivitás biztosítása.
- Alacsony bitsebesség.
- Erős hibatűrő képesség.
- Skálázhatóság, kódoló oldalon komplexitás skálázás (minőség rovására), dekódoló oldalon térbeli (kisebb képméret), időbeli (valóságosnál kevesebb kép lejátszása) és minőségi (többrétegű adatból kevesebb felhasználása) skálázás, illetve ezek kombinálása.
- Forma kódolás, arc és test animáció, 2D és 3D kódolás.
- Különböző profilok kialakítása.

Mivel a diplomamunkának nem célja a korábbi tömörítési technikák részletes bemutatása, ezért itt csak a legjelentősebb tömörítési eljárásokkal kerültek említésre vázlatosan. A továbbiakban a H.264 eszközkészlete és tulajdonságai kerülnek bemutatásra.

4. A H.264/MPEG-4 AVC

A H.264 egy eredetileg alacsony bitsebességre tervezett kódoló (x * 64) kbps-os videokonferencia alkalmazások kódolására, videó-telefon analóg vonalon, elektronikus újságok, interaktív multimédia adatbázisok átvitelének támogatására stb. Alkalmazható QCIF (176x144) 10 fps vagy kisebb képfelbontásokig, 4,8-64 Kbit/s között egészen 128Kbit/s bitrátáig. Bizonyos továbbiakban említett fejlesztéseknek köszönhetően ugyanolyan bitsebességnél, vagy akár 1/3 akkora sebességnél az MPEG-2-nél jobb képminőséget produkál, így alkalmazható videó szolgáltatások folyamainak tömörítésére is. Azért az eredeti célról sem kellett lemondani, hiszen a mobil hírközlésben a 3G szabvány videó-átviteli szabványának az MPEG-4-et választották. Az MPEG-4 következő formátumokat és adatátviteli sebességeket támogatja: tipikusan 5 Kbit/s és 10 Mbit/s közötti formátumok, progresszív és váltott soros (interlaced) videó-felbontás, tipikusan sub-QCIF (128x96) felbontástól az SDTV feletti felbontásokig, illetve nagyfelbontású (HD) videók legmagasabb minőségi igényű „intra frame” kódolása 600Mbit/s-ig. Alkalmas professzionális műsorszórás, HD stúdió, nagyfelbontású (HD) videók inter-frame (H-264) kódolására, tipikusan 10-20 Mbit/s sávszélességig; digitális földi, és műholdas HDTV adatátvitelre. A korábbi szabványokhoz képest a következő finomításokkal érték el a fejlesztők a jobb képminőséget a H.264 esetében:

A mozgás predikciónál a pontosabb leírók érdekében 16x16 - 4x4-ig változó méretű blokkokra végzik a mozgáskompenzációt. Ráadásul az MPEG-2-hez képest nemcsak vízszintesen, hanem függőlegesen is képesek jósolni a mozgásvektort. További fejlesztés, hogy a mozgásvektorokat nem csak egy referenciaképhez, hanem tetszőleges előző 32 képhez képest lehet megadni. Így a predikció jobb lesz, azaz kevesebb predikciós hibát kell átvinni a dekódoló oldalra, tehát alacsonyabb bitsebesség érhető el minőségromlás nélkül.

Képen belüli tömörítésnél a 8x8-as blokkok mentén balról – jobbra haladva a DC együtthatókat is próbálják megjósolni (MPEG-2). A képtartalomtól függően az MPEG-4 kódoló az AC koefficiensek közül képes a felső sort (a felette lévő blokkból), vagy a bal oldali oszlopot is (a tőle balra lévő blokkból) megjósolni. Például egy függőleges rudat ábrázoló kép esetében az MPEG-2 minden sorban óriási hibát vét a jóslással, hiszen éles határátmenet következik a DC együtthatókat tekintve, ezzel szemben az MPEG-4 képes

függőlegesen tömöríteni a rúd textúráját, és így gyakorlatilag hibátlanul becsli nemcsak a DC, hanem az említett AC együtthatókat is.

A kódoló a kvantálás után a 8x8-as blokkokból három különféle útvonalon olvassa ki a koefficienseket és ahol a legjobb a tömörítési lehetőség (a legtöbb az egymás utáni 0 érték), azt választja.

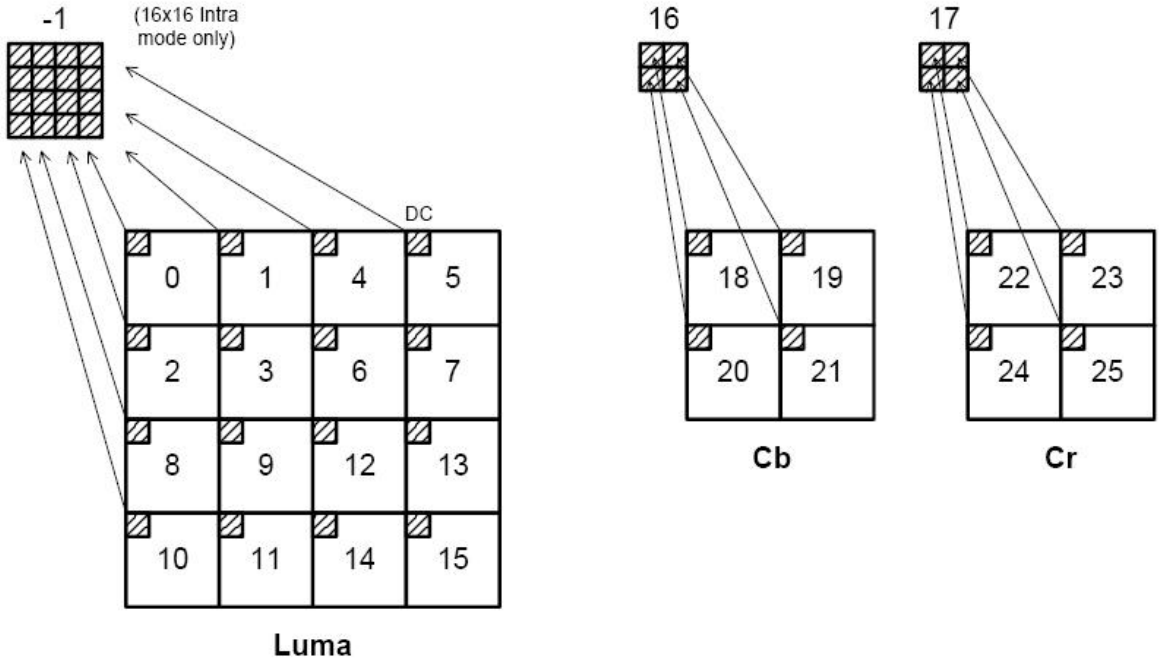
Globális mozgáskompenzáció. Egy nagyobb objektum globális mozgásvektoraiból interpolációval számítják egy adott 16x16-os makroblokk elmozdulás vektorát. Mindezen változtatások megnövelték a kódolás és a dekódolás hardverkövetelményeit (a kódoláshoz nyolcszoros, a dekódoláshoz négyszeres a teljesítmény-követelmény az MPEG-2-höz képest), de ennek a jelentősége napjainkra lecsökkent, hiszen ma egy átlagos asztali processzorral az MPEG-4 szaggatásmentes lejátszásához szükséges számítási kapacitás két-háromszorosát kapjuk, sőt elég olcsón valós idejű MPEG-4 kódoló kártyához is hozzájuthatunk. Más kérdés, hogy az előállított anyag minősége nem lesz az igazi, hiszen jó minőségű MPEG-4 állományt leginkább csak többmenetes kódolás során kaphatunk. Első menetben a szoftveres kódoló statisztikát készít a mozgások gyorsasága, objektumok bonyolultsága stb. alapján, és a második menetben ehhez igazítja a dinamikus módszereket, pl. a kvantálás paramétereit.

5. A H.264 eszközkészlete

5.1 *Blokk transzformáció és kvantálás*

A korábbi videó tömörítők, mint a MPEG-1, MPEG -2, MPEG-4 és a H.263 a diszkrét koszinusz transzformációt (DCT) alkalmaztak egy 8x8-as blokkokra. Ezzel szakítottak a H.264-ben. Bevezettek egy új transzformációt, az ún. Integer transzformációt [1-3]. A „Baseline" profilban 3 fajtája szerepel, attól függően, hogy a maradék blokk honnan származik. Egyiket a luma DC együtthatók 4x4-es tömbjére alkalmazzák, amikor egy makroblokkot 16x16-os intra módban dolgozunk fel. A második a chroma DC együtthatók 2x2-es tömbjének transzformálásánál használják, míg minden más esetben maradékok 4x4-es tömbjére egy harmadik transzformációt alkalmaznak.

Az **1. ábrán** látható a makroblokkban szereplő értékek továbbítási sorrendje.



1. ábra. A maradék blokkok bejárási sorrendje egy makroblokkon belül

Ha a makroblokk 16x16-os intra módban lett kódolva, akkor a -1-gyel jelölt, DC értékeket tartalmazó blokk kerül továbbításra először. Ezután a 0-15 blokkok, ahol a DC érték helyén már 0 szerepel, majd a 16-os és 17-es blokk, amely a chroma DC értékeket tartalmazza, végül pedig a chroma maradék blokkok (18-25), ahol szintén a DC értékek helyén 0 van.

Létezik 8x8-as blokkra is az Integer transzformáció, de ez lényegesen komplexebb, mint a 4x4-es esetében, ezért elég ritkán alkalmazzák.

Most tekintsük át az Integer transzformáció 3 fajtáját, majd pedig a kvantálás működését.

5.1.1 4x4 maradék transzformáció

Ezt a transzformációt alkalmazzuk a 0-15 és 18-25 maradék blokkokra a mozgás kompenzáció és az intra predikció után. Az Integer transzformáció visszavezethető a diszkrét koszinusz transzformációra (DCT), viszont van néhány igen jelentős különbég köztük:

1. Ez valóban egy Integer transzformáció, vagyis minden művelet végrehajtható egész aritmetikával. 16 bites egész aritmetika elégséges.
2. Az inverz transzformáció teljes mértékben specifikálva van a H.264 szabványban [1].
3. A transzformáció magja szorzás-mentes, vagyis miden művelet elvégezhető összeadással és biteltolással. Ez azért jelent nagy előnyt, mert így sokkal gyorsabb lesz az enkódolás. Ugyanis egy szorzás elvégzéséhez sokkal több processzor időre van szükség, mint egy összeadáshoz.

Integer transzformáció levezetése a 4x4-es DCT-ből

A 4x4-es DCT bemenete az *X* mátrix:

$$\mathbf{Y} = \mathbf{A}\mathbf{X}\mathbf{A}^T = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \quad \mathbf{X} \quad \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

Ahol:

$$a = \frac{1}{2}$$

$$b = \sqrt{\frac{1}{2}} \cos\left(\frac{\pi}{8}\right)$$

$$c = \sqrt{\frac{1}{2}} \cos\left(\frac{3\pi}{8}\right)$$

Ez a mátrixszorzás faktorizálható, így az előzővel egyenértékű a következő:

$$\mathbf{Y} = \left(\mathbf{C}\mathbf{X}\mathbf{C}^{T}\right) \otimes \mathbf{E} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} & & & \\ & \mathbf{X} & & \\ & & & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix}$$

CXC^T, a transzformáció magja 2 dimenziós. Az E a skálázó faktor mátrixa és a $\otimes$ szimbólum egy skalár szorzást jelöl a CXC^T és az E mátrixok megegyező pozícióin lévő elemek között. Az a és b konstansok értéke ugyanaz mint eddig, a d pedig c/b (megközelítőleg 0,414). A transzformáció implementálásának egyszerűsítése érdekében a d-t közelítsük 1/2-del. A transzformáció ortogonális voltának biztosítása miatt szükség van b értékének módosítására is:

$$a = \frac{1}{2}$$
$$b = \sqrt{\frac{2}{5}}$$
$$d = \frac{1}{2}$$

A C mátrix 2. és 4. sorát, a CT mátrix 2. és 4. oszlopát szorozzuk 2-vel, ezt pedig az E mátrixban kompenzálni kell osztással. (Azért kell ezt megtennünk, hogy elkerüljük a 1/2-del való szorzást a transzformáció magjában, így megmaradjon az egész aritmetika) Így kapjuk a transzformáció végső formáját:

$$\mathbf{Y} = \mathbf{C}_f\mathbf{X}\mathbf{C}_f^{T} \otimes \mathbf{E}_f = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} & & & \\ & \mathbf{X} & & \\ & & & \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix}$$

Ez a transzformáció egy közelítése a 4x4-es DCT-nek. Mivel a b és a d konstansok értékeit megváltoztattuk, ezért nem lesz teljesen ugyanaz a kimenet.

Az inverz transzformáció képlete a következő:

$$\mathbf{X}' = \mathbf{C}_i^{T}\left(\mathbf{Y} \otimes \mathbf{E}_i\right)\mathbf{C}_i = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \left(\begin{bmatrix} & & & \\ & \mathbf{Y} & & \\ & & & \end{bmatrix} \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$$

5.1.2 4x4-es luma DC együtthatók transzformációja

Ha a makroblokkot 16x16-os intra módban prediktáljuk, (ami azt jelenti, hogy egy 16x16-os luma komponenst prediktálunk a szomszédos képpontok értékekből) akkor először minden 4x4-es blokkra alkalmazzuk a fent leírt transzformáció magját ($C_f X C_f^T$). Majd az így keletkezett DC értékeket kigyűjtjük egy újabb mátrixba (-1-gyel jelöltük), utána pedig erre alkalmazzuk a 4x4-es Hadamard transzformációt:

$$\mathbf{Y}_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \mathbf{W}_D \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) / 2$$

Az inverz Hadamard transzformáció képlete a következő:

$$\mathbf{W}_{QD} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \mathbf{Z}_D \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right)$$

5.1.3 2x2-es chroma DC együtthatók transzformációja

Minden egyes chroma komponenst a makroblokkon belül 4 darab 4x4-es blokkra osztunk fel. Ezekre alkalmazzuk a 4x4-es maradék transzformációt, majd a DC együtthatókat kigyűjtjük egy 2x2-es mátrixba (W_D) és a következő transzformációt kell végrehajtani rajta:

$$\mathbf{Y}_D = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{W}_D \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

A transzformáció inverze:

$$\mathbf{W}_{QD} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{Z}_D \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

5.1.4 Kvantálás

A kvantálásra azért van szükség, hogy a kapott értékeket csökkenteni tudjuk, ezáltal jobb tömörítést tudjuk elérni minimális adatvesztéssel. Kódolás során majd döntenünk kell, hogy mi lesz a fontosabb: a jobb tömörítési arány vagy a minőség.

A H.264 skalár kvantálást használ. Az alap kvantáló művelet a következő:

$Z_{ij} = \text{round}(Y_{ij} / Qstep)$,

ahol Y_{ij} a transzformáció után kapott együttható, *Qstep* a kvantáló érték és Z_{ij} a kvantáló együttható.

A H.264 52 darab kvantáló értéket ajánl, amelyeket az ún. kvantáló paraméterrel (*QP*) indexel. A kvantáló paraméterekhez tartozó kvantáló értékeket tartalmazza az **1. táblázat**.

1. táblázat. A kvatáló értékek a H.264-ben

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	…
Qstep	0,625	0,6875	0,8125	0,875	1	1,125	1,25	1,375	1,625	1,75	2	2,25	2,5	
QP	…	18	…	24	…	30	…	36	…	42	…	48	…	51
Qstep		5		10		20		40		80		160		224

A *Qstep* minden 6. paraméternél megduplázódik, a paraméter eggyel történő növelése esetében pedig 12,5%-kal növelődik. A *Qstep* széles skálája lehetővé teszi a kódolónak, hogy megfelelően döntsön a bitráta és a minőség között. Lehetőség van a *QP* értékének makroblokkonkénti megválasztásához, valamint a makroblokkon belül különböző érték tartozhat a luma és a chroma blokkokhoz. A kvantáló paraméterre azért van szükség, mert ez minden esetben egész szám - míg a *Qstep* nem az -, így ezt könnyebben lehet majd tömöríteni és tovább küldeni.

A kvantálás során osztani kell a kvantáló értékkel, ami sok esetben lassú művelet. Ezt viszont a konkrét értékek ismeretében és mivel 16 bites egész aritmetikával dolgozunk (ami mindenképp eredményezni fog némi adatvesztést), vissza lehet vezetni néhány összeadásra és biteltolásra. Ez megint csak a kódolót fogja gyorsítani.

Összefoglalva, a kvantáló értékek megválasztásánál azt kell szem előtt tartani, hogy a lehető legjobb minőségre akarunk-e törekedni, vagy a korlátozott bitrátának kell megfelelnünk. Minél nagyobb *QP*-t választunk, annál jobb tömörítési arányt tudunk elérni, viszont ez mind a minőség romlására fog menni.

5.2 *Intra predikció*

A predikció jóslást, becslést jelent, és arra vonatkozik, hogy egy makroblokk vagy egy blokk képpontjainak értékét más képpontok értékeiből meg tudjuk becsülni. Ennek a folyamatnak az lesz a lényege, hogy jó becslés esetén a pontos értékektől való eltérés minimális lesz, és csak azt a minimális eltérést kell majd kódolnunk. A H.264-ben kétféle predikcióval találkozhatunk: intra és inter predikció. Tekintsük át előbb az intra predikciót [1], [4].

Az intra predikció lényege, hogy a környező blokkok bizonyos képpont értékeiből jósoljuk az aktuális blokk értékeit, majd a különbség mátrixot fogjuk transzformálni és továbbítani. A környező blokk az aktuális blokk felső és bal szomszédját jelöli, azoknak is csak 1-1 képpont sorát. A felső szomszéd alsó, a bal szomszéd jobb képpont sorát vesszük alapul.

Az intra predikciót 4x4-es, 8x8-as és 16x16-os luma blokkra, valamint 4x4-es chroma blokkokra alkalmazhatjuk.

5.2.1 4x4-es luma intra predikció

Intra predikció esetében a szomszédos a blokkot úgy kell érteni, hogy azt már előzőleg kódoltuk és rekonstruáltuk. Erre azért van szükség, hogy az estleges hibák ne vonuljanak végig a dekódolás során, ugyanis ekkor egy kép dekódolásának végén már hatalmas hibák keletkeznének.

A predikcióhoz felhasznált értékek **2. táblázat** szerint címkézzük:

2. táblázat. 4x4-es predikcióhoz a címkék

M	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
I	*A*	*b*	*c*	*d*				
J	*E*	*f*	*g*	*h*				
K	*I*	*j*	*k*	*l*				
L	*M*	*n*	*o*	*p*				

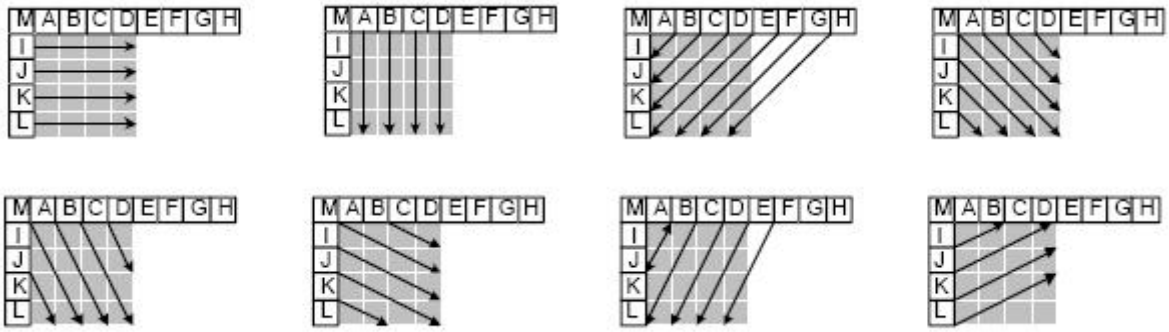
A nagybetűs értékek a szomszédos blokkok értékei. Viszont ezek nem mindig állnak rendelkezésünkre, mert például szelet határon helyezkedik el a blokk. Mindig csak olyan predikciós módot választhatunk, amely esetében a szükséges értékek rendelkezésünkre állnak. Ha az *E*, *F*, *G* és *H* értékek nem elérhetőek, akkor ezek a *D* értékét fogják megkapni.

A 4x4-es intra predikciónak 9 módja áll rendelkezésünkre:

0. Függőleges

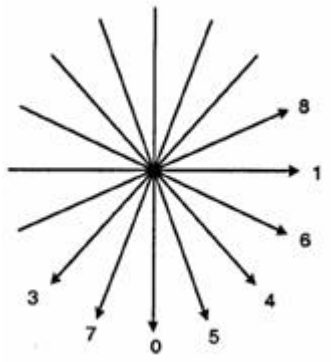
1. Vízszintes
2. DC
3. Átlósan balra lefelé
4. Átlósan jobbra lefelé
5. Függőlegesen jobbra
6. Vízszintesen lefelé
7. Függőlegesen balra
8. Vízszintesen felfelé

Ezen predikciós módok alkalmazásait mutatja be szemléletesen **2. ábra**. A vetítések egyértelműen mutatják, hogy hova milyen értéknek kell kerülnie. A DC mód pedig azt jelenti, hogy minden pozícióra az *A-D*, *I-L* értékek valamelyike fog kerülni.



2. ábra. 4x4-es intra predikciók szemléltetése a DC predikció kivételével

A **3. ábrán** az intra predikció irányai látszanak.



3. ábra. Predikciós irányok szemléltetése

Hogy mikor melyik módot kell választani, a következő dönti el: adott blokkra alkalmazzuk az összes predikciós módot, majd vegyük a kapott mátrix elemei és az eredeti

értékek különbségei abszolút értékeinek az összegét (Sum of Absolute Error, SAE). A legkisebb ilyen értékű predikciós módot kell választani, ugyanis ez fogja majd eredményezni a legjobb tömörítést. A különbség mátrix lesz a predikciós hiba, ezt kell majd transzformálni.

5.2.2 8x8-as luma intra predikció

A 8x8-as predikció a H.264 első változatában még nem szerepelt, később került csak bele. Teljes mértékben ugyanaz, mint a 4x4-es predikció, ugyanazok a predikciós módok alkalmazhatóak, csak a blokk méretében tér el.

5.2.3 16x16-os luma intra predikció

Lehetőség van makroblokk, azaz egy 16x16-os blokk prediktálására is. Ez esetben viszont mindössze 4-féle predikciós mód alkalmazható:

1. Függőleges
2. Vízszintes
3. DC
4. Plane

A függőleges és vízszintes predikció ugyanaz, mint korábban. A DC mód itt azt jelenti, hogy adott képpont pozíción a felső és bal szélső blokk szomszéd legközelebbi képpont értékeinek átlagát vesszük. A plane mód pedig a vízszintes és a függőleges predikció egy súlyozott kombinációjával dolgozik. A 4 lehetőség közül ugyanazzal a módszerrel kell választani, mint 4x4-es esetében.

Lehetőség van téglalap alakú blokkok prediktálására is. A 16x8-as és 8x4-es blokkokra általában a vízszintes, míg a 8x16-os és 4x8-as blokkokra a függőleges vetítést alkalmazzák.

5.2.4 Chroma intra predikció

A chroma blokkok (UV) esetében 4x4-es és 8x8-as blokkokra alkalmazható az intra predikció. Mindkét esetben ugyanazt a 4 predikciós módot használhatjuk, mint a 16x16-os luma intra predikciónál. Az eltérés csak annyi, hogy most a predikciós módok sorszáma más.

Egy makroblokkhoz tartozó mindkét chroma blokkra (*U* és *V*) ugyanazt a predikciós módot kell választani, illetve ha a luma blokkra intra predikciót használunk, akkor a chroma blokkokra is kötelező azt használni.

Az intra predikció azért hatékony tömörítési mód, mert csak a predikció sorszámát és a különbség mátrix transzformáltját kell továbbítani. Mivel a transzformáció kis értékeket és valószínűleg sok 0-át eredményez, ezért nagy hatásfokkal lehet majd tömöríteni. Ennél hatékonyabb tömörítést eredményezhet az inter predikció, amivel a következő fejezetben foglalkozunk.

Adott blokkra a legjobb predikciós mód kiválasztása történhet a már megismert módszerrel. A probléma ezzel csak az, hogy nagyon számítás- és időigényes. Ezért már sok gyors és hatékony algoritmus dolgoztak ki arra, hogy ne kelljen mindig kiszámolni az összes predikciós mód esetében az abszolút hibák összegét. Ezek az algoritmusok nem feltétlenül a legjobb tömörítést eredményezik, de nagyon közel vannak hozzá, és ami a legfontosabb, hogy nagyságrendekkel csökkentik a kereséshez szükséges időt és számításigényt.

5.3 *Inter predikció*

Az inter predikció [1] lényege, hogy egy blokkhoz nagyon hasonló blokkot keresünk egy korábbi, esetlegesen egy későbbi képen. Ez azért nagyon hasznos a tömörítés szempontjából, mert minél hasonlóbb blokkot sikerül találni, annál kisebb értékek lesznek a különbség mátrixban, ami viszont a transzformáció és az entrópia kódolás után nagyon kevés biten ábrázolható lesz.

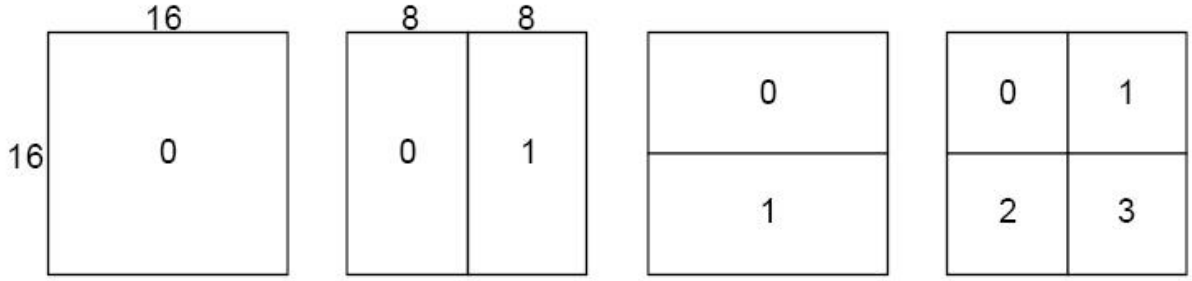
Inter predikcióval elsősorban a mozgásokat lehet jól leírni, ugyanis a mozgás úgy jelenik meg az egymást követő képeken, hogy (közel) azonos blokkok néhány képponttal eltolva jelennek meg. Ezt pedig elegendő egy képazonosítóval és mozgás vektorral (Motion Vector) leírni. Ha nem találunk teljesen azonos blokkot, akkor a különbség mátrixot kell transzformálni.

A mozgás vektor meghatározását nevezzük mozgás kompenzációnak (Motion Compensation, MC). Egyes irodalmak a mozgásbecslés (Motion Estimation) kifejezést használják. A H.264 blokk alapú, pontosabban változó blokkméretű mozgás kompenzációt

(Variable Block-Size Motion Compensation, VBSMC) használ. A blokk alapú azt jelenti, hogy blokkokra határozzuk meg a mozgás vektort és minden blokkra egyedileg. A változó blokkméret pedig azt jelenti, hogy kódolás közben a blokkok méretét változtathatjuk attól függően, hogy milyen paraméterek szerint szeretnénk tömöríteni.

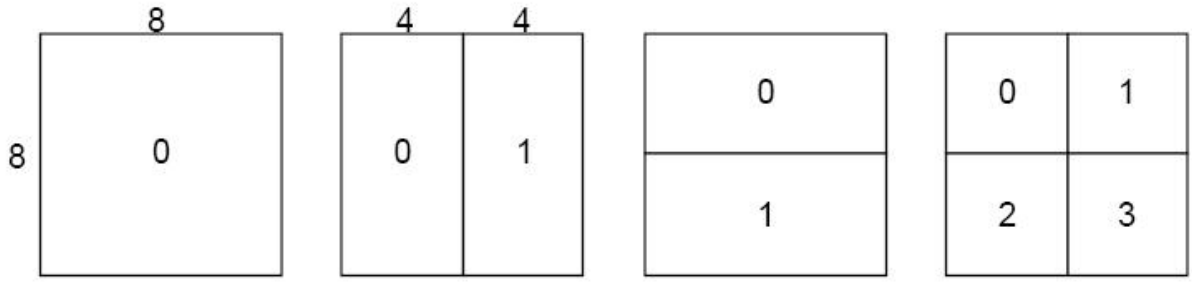
A H.264 hozott újdonságot az inter predikció terén is. Az egyik változtatás, hogy a legkisebb blokk méret 4x4-es, eddig ugyanis a 8x8-as volt. A másik, hogy az eddigi 1/2 képpont pontosságú mozgásbecslést felváltotta az 1/4 (luma) illetve 1/8 (chroma esetében) pontosság, valamint a frame határain túlra is mutathat a vektor.

A mozgás kompenzáció számára különböző blokk méretek állnak rendelkezésre a 16x16-ostól a 4x4-esig. A makroblokkot négyféle módon dolgozhatjuk fel: 16x16, 16x8, 8x16 és 8x8, ahogy azt a **4. ábra** is mutatja. Ezeket nevezzük makroblokk partícióknak.



4. ábra. Makroblokk partíció

Ha a 8x8-as méretet választjuk, akkor ezt további négyféle módon oszthatjuk tovább: 8x8, 8x4, 4x8 és 4x4 méretű blokkokra (makroblokk részpartíciók), ezt az **5. ábra** szemlélteti.



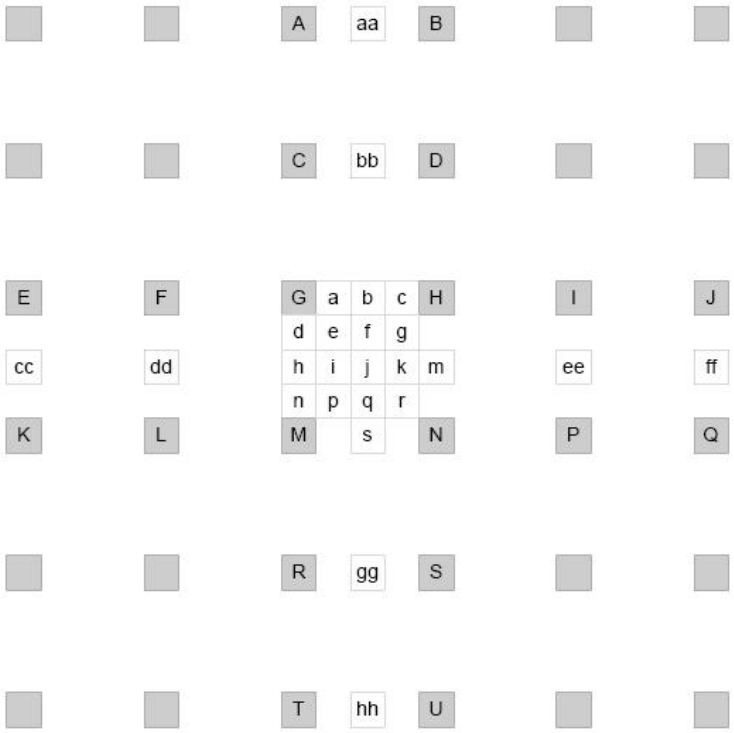
5. ábra. Makroblokk részpartíció

Ezek a felosztások elég nagyszámú lehetséges makroblokk darabolást tesznek lehetővé. A makroblokk felosztásának metódusát nevezzük fa struktúrás mozgás kompenzációnak.

Ha nagyméretű blokkokkal dolgozunk, akkor kevés bit szükséges a vektor és a felosztás típusának ábrázoláshoz, viszont feltehetőleg több helyre lesz szüksége a mozgás becslés maradék mátrixának. Ha viszont kisméretű blokkokkal számolunk, akkor több helyet

igényel a vektorok és a felosztás típusának ábrázolása, és viszonylag kevés helyen ábrázolhatóak a maradék mátrixok. Homogén területek esetében érdemes a nagy blokk méretek mellett dönteni, viszont a részletes, nagy eltéréseket mutató blokkok esetében célszerű a kisméretű felosztást választani.

A nem egész képpont értékeket és a frame határain kívülre eső értékeket interpolációval számítjuk ki. Először nézzük meg a luma komponensek esetében, hogyan kell kiszámítani a nem egész képpont értékeket.
A **6. ábra** szemlélteti a nem egész pozíciókon található képpontokat.



6. ábra. Nem egész pozíciókon található képpontok jelölése

A számításokhoz szükségünk lesz két segédfüggvényre:

$$\mathrm{Clip1}_Y(x) = \mathrm{Clip3}(0{,}255, x)$$

$$\mathrm{Clip3}(x, y, z) = \begin{cases} x; & z < x \\ y; & z > y \\ z; & \text{egyébként.} \end{cases}$$

Először a fél pozíción található *b* és *h* értékeket számoljuk ki egy ún. 6 részes filter - [1 -5 20 20 -5 1] - és a segédfüggvények segítségével:

$$b_1 = (E - 5 * F + 20 * G + 20 * H - 5 * I + J)$$

$$h_1 = (A - 5 * C + 20 * G + 20 * M - 5 * R + T)$$

$$b = \text{Clip1}_Y((b_1 + 16) >> 5)$$
$$h = \text{Clip1}_Y((h_1 + 16) >> 5)$$

Ezt követi a j értékének kiszámítása:

$$j_1 = cc - 5 * dd + 20 * h_1 + 20 * m_1 - 5 * ee + ff, \text{ vagy}$$
$$j_1 = aa - 5 * bb + 20 * b_1 + 20 * s_1 - 5 * gg + hh$$

$$j = \text{Clip1}_Y((j_1 + 512) >> 10)$$

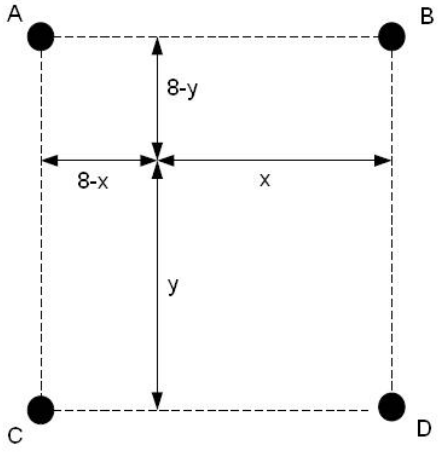
Az s_1 és m_1 értékeit ugyanúgy számoljuk, mint a b_1 és h_1 értékeit, amiből adódik a már ismert képlet:

$$s = \text{Clip1}_Y((s_1 + 16) >> 5)$$
$$m = \text{Clip1}_Y((h_1 + 16) >> 5)$$

A negyed pozíciókon található értékek meghatározása:

$$a = (G + b + 1) >> 1$$
$$c = (H + b + 1) >> 1$$
$$d = (G + h + 1) >> 1$$
$$n = (M + h + 1) >> 1$$
$$f = (b + j + 1) >> 1$$
$$i = (h + j + 1) >> 1$$
$$k = (i + m + 1) >> 1$$
$$g = (j + s + 1) >> 1$$
$$e = (b + h + 1) >> 1$$
$$g = (b + m + 1) >> 1$$
$$p = (h + s + 1) >> 1$$
$$r = (m + s + 1) >> 1$$

Most pedig nézzük meg, hogyan kell kiszámítani a nem egész képpont értékeket chroma esetben! Szemléletesen a **7. ábrán** látható.



7. ábra. Nem egész pozíciókon lévő chroma értékek számításának szemléltetése

Ez valamivel egyszerűbb, egy képlet alkalmazható minden egyes érték meghatározásához:
$E = ((8 - x) * (8 - y) * A + x * (8 - y) * B + (8 - x) * y * C + x * y * D + 32) >> 6.$

A mozgáskompenzáció már az MPEG-1 óta alapeleme a videó tömörítésnek. Egyben ez egyik kritikus pontja is, ugyanis a teljes kereséshez elfogadhatatlanul sok számításra van szükség. Ezért különböző algoritmusokat kell alkalmazni, hogy minél kevesebb számítással minél jobb vektort tudjunk találni. Tömörítés szempontjából nem feltétlenül a legjobb vektort fogjuk megkapni, de a számítások számának drasztikus csökkenése miatt beérhetjük egy közel legjobb vektorral.

Ezek az algoritmusok tehát az aktuális blokkunkhoz egy nagyon hasonló blokkot fognak szolgáltatni egy másik képről. Többféle kritérium szerint vizsgálhatunk egy blokkot, hogy az nekünk mennyire megfelelő: például a blokkok közötti képpont érték különbségek összege, vagy a különbségek négyzetösszege. De természetesen más feltételek is megadhatóak. Vannak könnyen alkalmazható feltételek, de vannak olyanok is, amelyek használata számításigényes.

Ezek a keresési algoritmusok általában úgy működnek, hogy első lépésben megvizsgálnak bizonyos blokkokat, majd ha nem találnak a feltételeknek megfelelőt, akkor következő lépésben blokkok egy másik csoportját vizsgálják át. Egyes algoritmusok a különböző lépésekben különböző kritériumokat támaszthatnak. Vannak olyan algoritmusok, amelyek az emberi szem érzékelésének határait kihasználva gyorsítanak a keresésen. Vannak, amelyek mindezeket vegyítik. Abban általában megegyeznek ezek az algoritmusok, hogy először az aktuális blokktól távol eső blokkok egy kisebb részét vizsgálják, majd ahogy haladnak középre, úgy a blokkok egyre nagyobb részét elemzik, ugyanis jó okkal feltételezhető a keresett blokk nem lesz nagyon távol az eredeti helytől. Most pedig nézzünk át néhány nagyon egyszerű blokk alapú keresési algoritmust [10]:

Háromlépéses keresés (Three Step Search, TSS)

Egyszerű, robosztus és közel optimális eredményt szolgáltat. Az algoritmus lépései:

1. Vegyünk fel egy kezdő lépésközt. A középponttól lépésköz távolságra lévő 8 (balra, jobbra, alatt, fölött és másik 4 úgy, hogy ezek egy négyzet csúcsait alkossák, az előbbiek pedig oldalfelezők) blokkot kell vizsgálni.
2. A lépésközt felezzük. Az új középpont az előbb vizsgált blokkok közül a legkedvezőbb eredményt nyújtó lesz.

3. Az első két lépést addig ismételjük, amíg találunk egy megfelelő blokkot vagy a lépésközt nem tudjuk tovább csökkenteni.

Az algoritmus problémája az első lépésben kiválasztott blokkok egységes mintája, amivel a kis mozgásokat nem lehet megfelelően meghatározni.

Kétdimenziós logaritmikus keresés (Two Dimensional Logarithmic Search, TDL)

A háromlépéses algoritmussal egy időben keletkezett. Több lépésből áll, viszont pontosabb eredményt szolgáltat. Az algoritmus lépései:

1. Vegyünk fel egy kezdő lépésközt. A középponttól lépésköz távolságra lévő 4 (balra, jobbra, lent és fent) blokkot és a középpontot vizsgáljuk.
2. Ha a megfelelő blokk a középponti, akkor a lépésközt felezzük, egyébként a 4 blokk közül a legjobb lesz a középpont, majd az 1-es lépést ismételjük.
3. Ha már a lépésközt nem tudjuk tovább csökkenteni, akkor megvizsgáljuk a középpontot és a körülötte lévő 8 blokkot és kiválasztjuk a legmegfelelőbbet.

Ennek az algoritmusnak sok változata létezik, amelyek a lépésköz változtatásában térnek el.

Ortogonális kereső algoritmus (Orthogonal Search Algorithm, OSA)

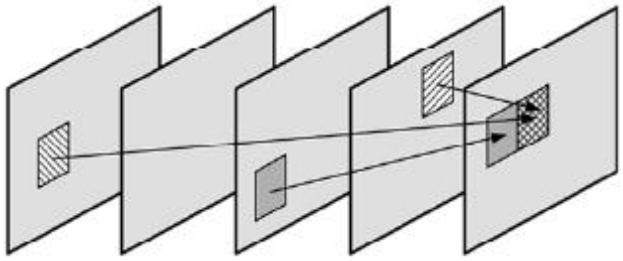
Ez az algoritmus az előző kettő keverékének is tekinthető. Lépéseiben függőleges és vízszintes állomásokat váltogat. Az algoritmus lépései:

1. Vegyünk fel egy kezdő lépésközt. Válasszuk ki a megfelelőbb blokkot a középső, balra és jobbra lépésközre lévők közül, és legyen ez a középpont.
2. Válasszuk ki a megfelelőbb blokkot a középső, balra és jobbra lépésközre lévők közül, és legyen ez a középpont.
3. Felezzük a lépésközt és ismételjük az első két lépést, amíg tovább nem csökkenthetjük a lépésközt.

Ezenkívül még az egyszerűbb algoritmusok közül említhetnénk még a bináris, a 4 lépéses, a keresztirányú, a spirális, a gyémánt vagy a hierarchikus blokkegyezést kereső algoritmusokat. Ma már ezeket önmagukban nem használják a nem elfogadható eredményeik miatt, hanem ezek valamilyen tovább fejlesztett és/vagy keverékeit alkalmazzák. Mára már rengetek jól használható algoritmust dolgoztak ki, de hely hiány és ezek bonyolultsága miatt erre most nem térünk ki.

Korábban egy képhez csak az őt megelőző kép lehetett referencia. A H.264-ben viszont az adott képhez az őt megelőző 16 kép közül választhatunk referencia képet. Ezt nevezzük több referenciaképes predikciónak (MRF, Multiple Reference Frames).

A H.264-ben egy képhez több referenciakép is tartozhat. Ez azt jelenti, hogy egy blokkot nem egy, hanem több blokkból, azok súlyozásával becsüljük meg. Ezt nevezzük több képes mozgás kompenzált predikciónak (Multi-Picture Motion-Compensated Prediction). Ezt a két tulajdonságot szemlélteti a **8. ábra**.



8. ábra. Több referenciaképes mozgás kompenzált predikció

Mint azt már a fejezet elején említettem, a mozgás kompenzáció nem csak korábbi képekkel dolgozhat, hanem jövőbeliekkel is. A predikció ezen és még néhány tulajdonságának jobb megértése érdekében először nézzük át a frame-ekkel kapcsolatos tudnivalókat.

5.4 *Szerkezeti elemek*

Ebben a fejezetben a H.264-es videó folyam szerkezeti felépítését és az egyes elemek típusait vizsgáljuk meg.

A H.264-es videó folyam felépítése [1] két részre osztható: a hálózati absztrakciós réteg (Network Abstraction Layer, NAL) és a videó kódolási réteg (Video Coding Layer, VCL). A NAL részei:

- NAL egység (NAL unit)
- VCL és nem VCL egység (Non-VCL Units)
- Paraméter halmazok
- Hozzáférési egységek (Access Units)

A VCL elemei:

- Képek és mezők

- Makroblokkok
- Színtér
- Szeletek és szelet csoportok (Slices, Slice Groups)
- Adaptív kép / mező kódolás
- Intra és inter predikció

A H.264 a videó folyamot diszkrét csomagokba szervezi, melyet NAL unit-nak (Network Abstraction Layer unit, hálózati absztrakciós réteg egység) nevezünk. Egy NAL unit általában egy szelet egy részét tartalmazza. De ezen kívül tartalmazhat még aláírásokat, fejrészeket és további adatokat.

Szeletnek (slice) nevezzük a frame egy részét, makroblokkok egy halmazát. Egy frame lehet egyetlen szelet is, de állhat több szeletből is. A szeletek szelet csoportokat is alkothatnak, így a frame szelet csoportokból fog állni. Egy frame maximálisan 8 szelet csoportból állhat. Lényeges még továbbá, hogy bármelyik makroblokk bármelyik szelethez tartozhat.

Egyik nagy előnye a H.264-nek, hogy lehetőség van videó folyam továbbítás során egyik folyamról egy másikra átváltani. Például adott nagyfelbontású videó folyam nem továbbítható a szűk sávszélesség miatt, ezért csak egy kisebb részét szeretnék a továbbiakban küldeni. Ezt úgy érhetjük a szeletek segítségével, hogy például a kép négy oldalát és a középső részét külön-külön szeletekként dolgozzuk fel, így a váltás után csak a középső szelet kerül továbbításra. A kép több szeletre osztása még azért is előnyös, mert a különböző szeleteken különböző tömörítési technikákat alkalmazhatunk. Pontosan ez alapján soroljuk kategóriákba a szeleteket:

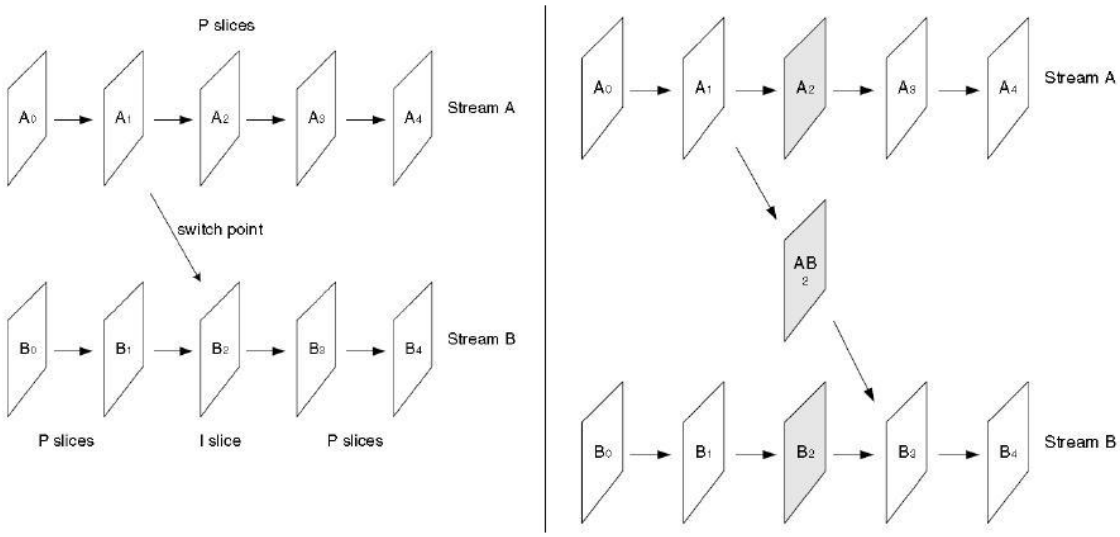
***I* slice** vagy intra slice csak *I*-blokkokat tartalmazhat, vagyis csak olyanokat, amelyek intra predikcióval kódoltunk. Előnye, hogy önmagában dekódolható, mert nem tartalmaz más szeletekre hivatkozást, viszont hátránya a viszonylag rossz tömörítési arány. Ha egy kép csak ilyen szeletekből áll, akkor I-frame-ről beszélünk. A videó folyam első képkockájának kötelező ilyennek lennie. Bizonyos frame szám után mindig el kell helyezni egy *I*-frame-et a folyamban a tekerés miatt. I-blokk referenciája lehet *P*- és *B*-blokknak egyaránt.

***P* slice** vagy prediktív (predicted) slice *P*- és *I*-blokkokat tartalmazhat. *P*-blokknak nevezzük az inter predikcióval kódolt blokkot. Ez a leggyakrabban használt szelet típus, ugyanis

könnyen kezelhető és nagyon jó tömörítési arányt lehet vele elérni. *P*-blokk referenciája lehet *P*- és *B*-blokknak egyaránt.

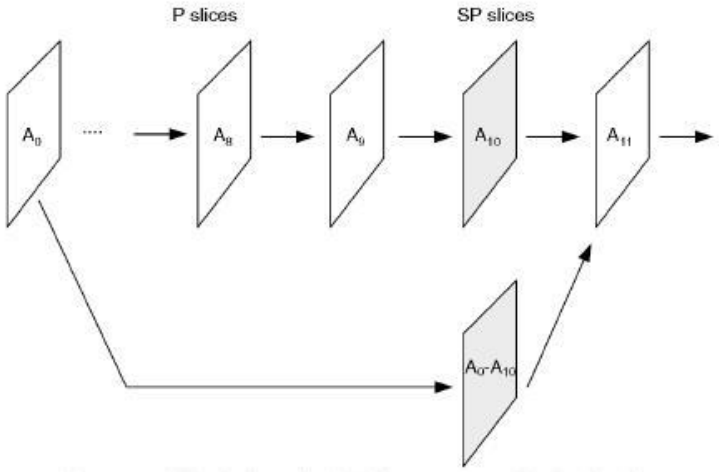
***B* slice** vagy bi-prediktív (bi-predicted, bi-directional predicted) slice *P*- és *B*-blokkokat tartalmazhat. *B*-blokknak nevezzük az olyan blokkot, amit inter predikcióval állítunk elő egy múltbéli és egy jövőbeni referenciakép súlyozásával. A H.264 újdonsága, hogy ez a súlyozás a korábbi 1/2:1/2 arány helyett most már *k*:(1-*k*) arányú is lehet, így a pontosabb interpoláció miatt kisebb lesz a különbségi adatmennyiség. *B*-blokk nem lehet referencia semmilyen más blokk számára sem. *B*-blokkok használat nem túl népszerű, ugyanis a képkockák időbeli tetszőleges feldolgozása nagyban bonyolítja a kódolást és sok memóriát is igényel.

***SI* és *SP* slice** vagy switching (kapcsoló) slice szintén a H.264 által hozott újdonságok közé tartozik. A csak ilyen szeleteket tartalmazó képeket nevezzük *SI*- és *SP*-frame-nek, amelyek átkapcsolást lehetővé tevő képek két bitfolyam között (pl. 1. stream-ből 2. stream-be, ahol az irány számít), vagy azonos bitfolyamon belül egy távoli pozícióra, ha két bitfolyam van ugyanazon tartalommal, de más reprezentációban (pl. más bitsebesség vagy képfrekvencia). Azoknál a képeknél, ahol az átkapcsolást engedélyezzük 1. stream-ből 2. stream bitfolyamra, beteszünk egy harmadik képet. Ennek referenciája az 1. stream-ben van, de paramétereit tekintve a 2. stream képeihez jelent átmenetet. Azonos bitfolyam távoli pozíciójára hasonlóan járunk el. A tároláskor redundancia lép fel, de az átvitel során nem. Természetesen a 2. stream bármelyik I képére át lehet kapcsolni betétkép nélkül, hiszen ezek önmagukban kódolódnak. Az **9. ábrán** minkét átmenetre láthatunk példát. Az ilyen típusú frame-ek használata elég népszerűtlen.



9. ábra. Átkapcsolás I és P frame-re

A „gyors előretekerés” funkciót megvalósító SP frame használatára láthatunk példát az **10. ábrán**.



10. ábra. Gyors előretekerés SP frame segítségével

Speciális frame típus az IDR (Instantaneous Decoding Refresh) kép, amely a szekvencia hozzáférési pontja. Hasonlít az *I*-frame-re, vagyis önmagában dekódolható, viszont minden más szekvencia paramétert is tartalmaz, amely szükséges ahhoz, hogy ettől a ponttól kezdve a videó folyam dekódolható legyen. Korábbi kép IDR után nem lehet referencia.

Arról már volt szó, hogy az egy szeletben található makroblokkok elhelyezkedése tetszőleges a képen belül, ezért ezt nevezik rugalmas makroblokk elrendezésnek is (Flexible Macroblock Ordering, FMO). A makroblokk allokációs térképben (MacroBlock Allocation map, MBAmap) tartjuk nyilván, hogy melyik makroblokk melyik szelethez tartozik. Az FMO és a tetszőleges szelet elrendezés (Arbitrary Slice Ordering, ASO) a frame-en a makroblokkok reprezentációjának újrastrukturálási technikái. Ezek szintén a videó tömörítés új elemei. Az FMO és az ASO tipikusan hibatűrési, robusztussági tulajdonságok, de egyéb célokra is felhasználhatóak. Az FMO alkalmazása más fejlett hibatűrési eszközökkel a vizuális minőséget meg lehet tartani még a csomagok 10%-ának elvesztése esetén is.

A szeleteknek tömörítés szempontjából is van előnyük, ugyanis adott szeleten azonos típusú blokkok szerepelnek, ezért bizonyos blokk tulajdonságokat elég egyszer tárolni a szelet fejrészében.

NAL unit tartalma a szeleteken kívül például különböző paraméter halmazok lehetnek:

Szekvencia paraméter halmaz (Sequence Parameter Sets, SPSs): az egész szekvenciára, videó folyamra vonatkozó információkat tartalmazza. Néhány paraméter ezek közül a teljesség igénye nélkül:

- profilazonosító
- szintazonosító
- szekvencia paraméter halmaz azonosító
- chroma formátum azonosító
- chroma és luma bitmélység
- referencia képek száma
- kép szélessége és magassága

Kép paraméter halmaz (Picture Parameter Sets,PPSs): egy képre vonatkozó paraméterek, információkat tartalmaz. Néhány paraméter ezek közül a teljesség igénye nélkül:

- kép paraméter halmazazonosító
- szekvencia paraméter halmaz azonosító
- entrópia kódolási mód azonosító
- kép megjelenítési sorrendjének azonosítója
- szelet csoportok száma
- súlyozott predikció jelző
- kezdő kvantáló paraméter

A kiegészítő megerősítő információ (Supplemental Enhancement Information, SEI) és videó használhatósági információ (Video Usability Information, VUI) olyan kiegészítő információkat tartalmaznak (pl. időzítési információk), amelyek fokozzák a dekódolt videó jel használhatóságát, de nem szükségesek a dekódolás folyamatához. Tetszőlegesen elhelyezhetőek a videó folyamba.

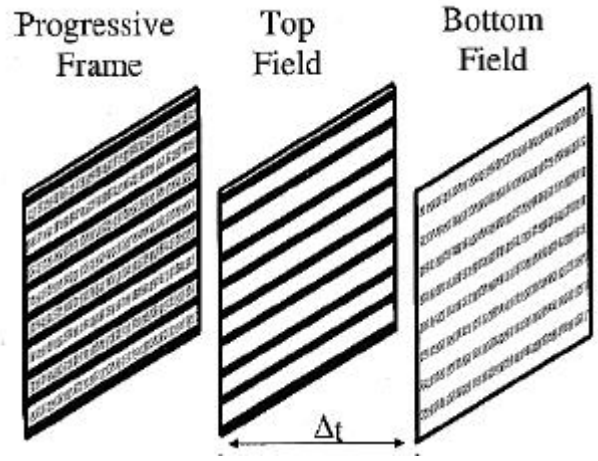
Színtér: A H.264 az YUV vagy YCbCr színtért használja, ahol *Y* a luma (világosság) az *U* és *V* (vagy *Cb* és *Cr*) a szín értékeket jelölik. A közismertebb RGB színtérrel szemben az a nagy előnye, hogy a szomszédos vagy közeli értékek kevésbé térnek el egymástól még heterogén képrészletnél is. Ez a tömörítésnél jelent óriási előnyt, mert például az RGB-nél egy heterogén részletnél óriási ugrások vannak az egymáshoz közeli képpont értékeknél, ezért ezt sokkal nehezebb lenne tömöríteni is.

Az emberi szem sokkal érzékenyebb a világosságra, mint a színre, ezért elegendő kevesebb szín információ használatára. A H.264 először a 4:2:0 szín felbontást alkalmazta,

ami azt jelenti, hogy egy luma makroblokkhoz két 8x8-as chroma blokk tartozik. Később bevezetésre került a 4:2:2 típus, ahol egy makroblokkhoz két fele akkora chroma blokk, majd a 4:4:4 típus, ahol egy luma blokkhoz két ugyanakkora chroma blokk tartozik.

A H.264 a progresszív kódolás mellett az interlaced (váltott soros) kódolást is támogatja. A progresszív kódolás azt jelenti, hogy a teljes képet dolgozzuk fel, míg az interlacing kódolás felváltva csak a kép páros illetve páratlan sorait dolgozza fel. Ez utóbbi technika régebben volt népszerű a kisebb sávszélesség és megjelenítési problémák (régi katódsugárcsövek hosszú kialvási ideje) miatt. Ma már ezek nem jelentenek gondot, ezért nem is nagyon használatos már ez a kódolási mód. Ennek ellenére a H.264 még támogatja.

A H.264-ben rugalmas interlaced letapogatás videó kódolási eszköz áll rendelkezésünkre, mely a következőket tartalmazza: a frame-et két részre osztjuk, felső és alsó mezőre (top és bottom field). Ez úgy néz ki, hogy a felső mezőhöz tartoznak a páratlan, az alsóhoz pedig a páros makroblokk sorok (lásd **11. ábra**). Makroblokk párnak nevezzük egy felső mezőben elhelyezkedő és az alatta lévő szomszédos makroblokkot.



11. ábra. Progresszív és interlaced frame

Kódolási módok:

Frame mód: a két mező alkotta frame-et egy képként kódoljuk.

Field mód: a két mezőt külön-külön kódoljuk, mintha külön képek lennének.

Makroblokk-adaptív kép-mező kódolás (Macroblock-adaptive frame-field coding, MBAFF): a két mező alkotta frame-et egy képnek tekintjük, de megengedjük, hogy a függőlegesen szomszédos makroblokk párokat egyénileg mezőkre osszuk a predikció számára.

Ha az első két eset valamelyikét választjuk, akkor kép-adaptív kép-mező kódolásról (Picture-Adaptive Frame-Field coding, PAFF vagy PicAFF) beszélünk.

Az interlace technikák képenként meghatározhatóak és a kép tulajdonságaitól függően ki- és bekapcsolhatóak.

5.5 *Blokkosodás elleni szűrő*

A blokkosodás a kvantálási hiba következménye. A blokkhatáron nagyobb az eltérés a különböző blokkok szomszédos képpontjai között, mint blokkon belül (ez többnyire csak szubjektív hiba). A hibaeloszlás nagyobb energiájú a blokk határán, mint a blokkon belül (objektív tény). Ez a hiba a blokk alapú transzformáció miatt jön létre, és úgy látjuk, hogy a blokk határain megszűnik a kép folytonossága. (lásd **12. ábra**)

12. ábra. Blokkosodás elleni szűrő használata előtt és után

A blokkosodás a legjelentősebb torzulás, ami a tömörítés során a képet éri, ezért a H.264-ben is nagy hangsúlyt fektetnek a hatásának csökkentésére. Ezt szűrők segítségével lehet elérni. A H.264 által használt szűrőt In-Loop Deblocking Filter-nek nevezzük [1].
A szűrés erőssége különböző szinteken változtatható:

- Szelet szinten dinamikusan elküldve.
- Blokkhatár szinten (2 szomszédos blokk érintett) az alábbiak függvénye: inter / intra predikciót használtunk, mozgásvektorok értéke közötti eltérés és becslési hiba jelenléte.
- Képpont szinten úgy, hogy a szűrés a képpont érték és kvantálási szint alapján ki-be kapcsolható.

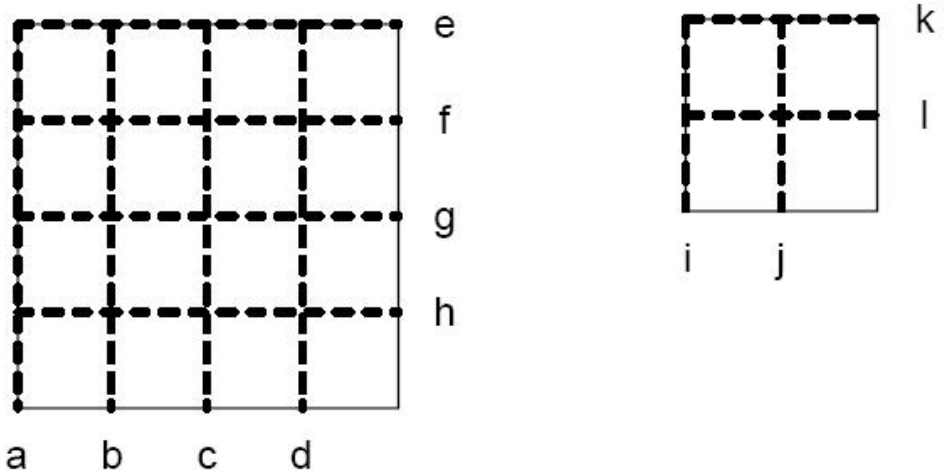
A blokkosodás elleni szűrő már a kódoló oldalon megjelenik (erre utal az in-loop kifejezés), mivel az enkóder tartalmaz dekódert is, ugyanis egy kép nem az eredeti formájában lesz referenciakép, hanem dekódolt alakban. Ezáltal tovább javul a kódolt kép minősége. Különösen alacsony bitrátánál jelentkezik jótékony hatása, viszont mind a kódolás, mind a dekódolás folyamatát lassítja.

A szűrő működése

A szűrőt minden dekódolt makroblokkra alkalmazzuk a torzulás csökkentésének érdekében az inverz transzformáció után. A szűrőnek két előnye van: kisimítja a blokk éleit, javítva ezzel a dekódolt kép megjelenését (különösen magasabb tömörítési aránynál), a szűrt makroblokk pedig hasznos a mozgás kompenzált predikcióra a további képek számára az enkóderben, ami kisebb maradékot eredményez a predikció után. A kép éleit nem szűrjük.

A szűrést alkalmazzuk a makroblokkban a 4x4-es blokkok függőleges és vízszintes éleire a következő sorrendben:

1. Szűrűnk a luma komponens 4 függőleges határára (sorrendben *a*, *b*, *c*, *d*, lásd **13. ábra**)
2. Szűrűnk a luma komponens 4 vízszintes határára (sorrendben *e*, *f*, *g*, *h*)
3. Szűrűnk a chroma komponens 2 függőleges határára (sorrendben *i*, *j*)
4. Szűrűnk a chroma komponens 2 vízszintes határára (sorrendben *k*, *l*)



13. ábra. Él filterezési sorrend makroblokknál (16x16-os luma és 8x8-as chroma blokk esetében)

Mindegyik szűrő operátor maximum 3 képpontra van hatással a blokkhatár mindkét oldalán. Az aktuális kvantáló paramétertől, a szomszédos blokkok kódolási módjától és

képrészlet határon túlra való esésétől sokféle kimenetele lehet a szűrésnek: lehet, hogy egyetlen képpontot sem, de lehet, hogy mind a 6 szóba jöhető képpontot érinti a szűrés.

5.6 *Entrópia kódolás*

Az átviendő adatok mennyiségét a statisztikai törvényszerűségek kihasználásával, különböző speciális eljárások segítségével is hatékonyan csökkenteni lehet.

Az entrópia kódolás lényege, hogy a leggyakrabban előforduló mintákhoz a legkevesebb számú bitet rendeli. Dekódoláskor természetesen visszakapjuk az eredeti mintát. Az eljárás veszteségmentes, és a videó folyam statisztikai jellemzőin alapul.

Entrópia

Minden előforduló jel (p) információval bír:

$I_i = -\log_2 p_i$

A jelforrás entrópiája az átlagosan egy jelre jutó információ mennyisége,

$H = \Sigma^n_{i=1} p_i I_i = -\Sigma^n_{i=1} p_i \log_2 p_i$

ami akkor veszi fel legnagyobb értékét, ha a jelek egyenletes eloszlással rendelkeznek.

Zajmentes információforrás kódolásának tétele

A tétel a forrásszimbólumok blokkjainak bináris kódszavakká való átkódolásának határait adja meg. Legyen S ergodikus (végtelen számú realizáció) jelforrás n méretű ABC-vel és $H(S)$ entrópiával. N db szimbólumú blokkokat kódolunk bináris kódszavakká, a veszteségmentes kódolás lehetséges tömörségét a következő állítás adja meg:

Bármilyen δ>0-hoz létezik (elég nagy) N, hogy az átlagos, egy szimbólumra eső bitek számára fennáll a következő egyenlőtlenség:

$$H(S) \leq L < H(S) + \delta$$

Változó kódhosszúságú kódolók

A cél, úgy kódolni az információforrás jeleit, hogy minimális legyen a kód hossza, miközben nem történik információvesztés.

Entrópia: $\Sigma^n_{i=1} \mathrm{p}(s_i)\, I\,(s_i)$

Átlagos kódhossz: $L = \Sigma^n_{i=1} p(s_i)\, \mathrm{l}\,(s_i)$

Optimális esetben: $l(s_i) = I(s_i) = -\log_2 p(s_i)$

Tehát $l(s)$ legyen: $-\,[\log_2 p(s_i)]$

Kódolás hatékonysága veszteségmentes tömörítés esetén: $\eta = H(S) / L$

Általános entrópia kódolók

- Shannon-Fano kódolás
- Huffmann kódolás
- Futáshossz kódolás
- Aritmetikus kódolás
- LZW algoritmus

Ezek részletezése nem tartozik szorosan a témához, inkább rátérnék a H.264 által használt entrópia kódolásokra.

A H.264 szabványa kétfajta entrópia kódolást használ: a környezet adaptív bináris aritmetikai kódolást (Context-Adaptive Binary Arithmetic Coding, CABAC), és a környezet adaptív változó hosszúságú kódolást (Context-Adaptive Variable Length Coding, CAVLC).

Aritmetikai kódolás

Az aritmetikai kódolás ötlete az, hogy N forrásszimbólumra a [0,1) intervallumot N rész-intervallumokra bontjuk úgy, hogy minden szimbólum kap egy részintervallumot, amely rész-intervallum hossza a szimbólum valószínűsége. Egy forrásszimbólum kódolásának eredménye a szimbólum részintervalluma, illetve egy tetszőleges szám ezen részintervallumon belül. A szimbólumok sorozatának kódolási eredménye az utolsó részintervallumból egy tetszőleges szám.

A [0,1) intervallum felosztása formálisan a következő:

Forrásszimbólum	Intervallum hossza	Intervallum
x_1	$P(x_1)$	$[0, P(x_1))$
X_2	$P(x_2)$	$[P(x_1), P(x_1)+P(x_2))$
X_3	$P(x_3)$	$[P(x_1)+P(x_2), P(x_1)+P(x_2)+P(x_3))$
…	…	
x_i	$P(x_i)$	$[P(x_1)+...+P(x_{i-1}), P(x_1)+...+P(x_i))$
…	…	
x_N	$P(x_N)$	$[P(x_1)+...+P(x_{N-1}), 1)$

A kumulált valószínűségeket bevezetve a lehetséges részintervallumok megadása egyszerűbb lesz. Jelölje az x_k-hoz tartozó kumulált valószínűséget c_k, vagyis

$c_k = \Sigma^k_{i=1} p(x_k)$, ha $k = 1..N$ és legyen $c_0=0$.

Ekkor az x_k-hoz kezdetben a [0,1) intervallum $[c_{k-1}, c_k)$ rész-intervalluma tartozik. Fontos, hogy 0 valószínűséget nem engedünk meg, vagyis $c_i<c_j$, ha $i<j$.

A következő forrásszimbólumokat úgy kódoljuk, hogy a [0,1) intervallum szerepét az előzőleg kapott részintervallum veszi át. Így a forrásszimbólumok sorozatának kódolásával

egymásba skatulyázott részintervallum-sorozatot kapunk, a kódolás végeredménye pedig az utolsó, legkisebb részintervallumon belül egy tetszőleges szám lesz - ezt a számot tehát minden részintervallum tartalmazza, így lépésről lépésre az összes szimbólum dekódolható.

Intervallumok egymásba skatulyázása

Jelölje a $t+1$-dik forrásszimbólum, $x(t+1)$ kódolásakor a kiindulási intervallumot $[l(t),h(t))$, és legyen az intervallum hossza kezdetben $r(t)=h(t)-l(t)$. Az algoritmus alapján, ha $x(t+1) = x_k$, akkor az intervallumok egymásba skatulyázásának algoritmusa:

- az intervallum hosszára: $r(t+1) = r(t) \cdot p(x_k)$
- az intervallum alsó határára: $l(t+1) = l(t) + r(t) \cdot c_{k-1}$
- az intervallum felső határára: $h(t+1) = l(t) + r(t) \cdot c_k$

Ebből látható, hogy ha a 0 valószínűséget nem engedjük meg, akkor a skatulyázás helyes, vagyis: $l(t+1) \geq l(t)$, $h(t+1) \leq h(t)$ és $l(t) < h(t)$.

A fenti összefüggés alapján már látható, hogy mitől végez gyakoriságfüggő kódolást az aritmetikai kódoló: t darab szimbólum kódolása után a kapott részintervallum hossza:

$$r(t+1) = \prod_{i=1}^{t+1} p(x(i)),$$

az intervallum két végpontjának távolsága pedig bitekben kifejezve

$$B = [-\log_2(r(t+1))] = [-\log_2(\prod_{i=1}^{t+1} p(x(i)))] = [-\prod_{i=1}^{t+1} \log_2 (p(x(i))],$$

tehát legfeljebb ennyi bit kell az intervallum egy pontjának kódolására.

Fix pontos megvalósítás

A fix pontos megvalósítás fő problémája a következő: a fentiek alapját tehát meg kell határozni (a forrásszimbólumok sorozatát végigolvasva), hogy B mennyi, és egy B pontosságú aritmetikával ezután végigszámolni az összes szimbólumra a fenti műveleteket. Ez például járható út akkor, amikor pl. 10 szimbólumból álló vektorokat kell egyszerre kódolni, ahol a szimbólumok száma minden koordinátában max. 8 bittel leírható. De egy több száz kByte-os szöveg esetén ez lehetetlen.

A fix pontos implementáció lényege az, hogy három részre osztjuk az aktuális részintervallum határát megadó két számot, vagyis $l(t)$-t és $h(t)$-t.

1. rész: $l(t)$ és $h(t)$ bitjei azonosak: Mivel ezek a bitek már nem változhatnak meg $l(t)<h(t)$ miatt, ezért ezeket a biteket kiírjuk a kimenetre. A kódolás végeredménye ez a rész.

2. rész: csúszó ablak: Adott számábrázolási pontossággal (pl. A bittel, ahol A állandó) ábrázoljuk $l(t)$ és $h(t)$ ide eső bitjeit.

3. rész: túl a számábrázolási pontosságon: Itt feltételezzük, hogy $l(t)$ bitjei nullák, $h(t)$ bitjei pedig egyesek.

	1 rész	2. rész (fix biten)	3. rész
$l(t)$	0.10010011110100000111010100 11	0x003E	000000…
$h(t)$	0.10010011111010000111010100 11	0xC38F	111111…

A fix bites rész léptetésének az a szabálya, hogy az intervallumot lehetőleg minél pontosabban ábrázoljuk, hiszen a 3. részben már csak közelítünk.

Az a megoldáshoz vezető ötlet, hogy ha a csúszó ablakon belül h és l már olyan közel van egymáshoz, hogy legalább 1 biten közös a prefixumuk, akkor 1 bitet a kimenetre írva a csúszó ablakon belül a bitek balra lépnek egyet, és h és l értékeit 2-vel szorozva a köztük lévő távolság nő.

Kódolás során a kimeneten lévő bináris szám az l és h prefixuma. Az utolsó szimbólum kódolása után azonban még kell biteket kiírni, ezeket mindig úgy kell megválasztani, hogy az így kapott szám az l felett, vagy azzal egyenlő legyen, de a h alatt.

Mivel bináris számábrázoláson alapul az aritmetikánk, ezért az 1/4, 1/2 és 3/4 a kitüntetett pont (vagyis a felső 2 bit vizsgálata), és ezeket vizsgálva hasonlítjuk össze a csúszó ablakon belül az l alsó és a h felső korlátot.

A kiegyenlítő algoritmus léptetési stratégiája a következő:

h	l	mit jelent	mit kell tenni	magyarázat
$\mathbf{0}h_{1..A}$	(ekkor biztosan ez áll itt: $\mathbf{0}l_{1..A}$)	$h < 1/2$	$BitKi(0,1^{követőkSzáma})$ $h \leftarrow 1$, $l \leftarrow 0$ követőkSzáma = 0	h-ból és l-ből is 0 lép ki
(ekkor biztosan: $\mathbf{1}h_{1..A}$)	$\mathbf{1}l_{1..A}$	$l \geq 1/2$	$BitKi(1,0^{követőkSzáma})$ $h \leftarrow 1$, $l \leftarrow 0$ *követőkSzáma* = 0	h-ból és l-ből is 1 lép ki
$\mathbf{10}h_{2..A}$	$\mathbf{01}l_{1..A}$	$h < 1/2$ $l \geq 1/2$	++ *követőkSzáma* $h \leftarrow 1$ és $h_0 = 1$ $l \leftarrow 0$ és $l_0 = 0$	Ugyanaz, mintha a beléptetés előtt h -= 1/4 és l -= 1/4

A fenti algoritmusban csak a *követőkSzáma* szorul magyarázatra. Tekintsük ehhez a következő esetet:

Kezdeti állapot	9 lépés után
$h = 10^9 1 = 10000000001xx…$	$11xx…$
$l = 01^9 1 = 01111111111yy…$	$01yy…$

Ekkor tíz darab bitre előre meg tudjuk mondani, hogy az eredmény vagy 01^9, vagy pedig 10^9 lesz. A 11 bit alapján pedig fenti algoritmus még nem tud dönteni, a következő lépésben azonban új szimbólumot kódolunk le, ezért l nőni, h pedig csökkenni fog, tehát vagy (h esetében) az 10^9 fog alulcsordulni, és 01^9 lesz belőle, vagy pedig (l esetében) az 01^9 fog túlcsordulni, és 10^9 lesz belőle (a kettő együtt nem lehet, mert $l<h$). Az alulcsordulást pedig úgy vehetjük észre, hogy h esetében $\mathbf{0}h_{1..A}$ alakul ki, a túlcsordulást l esetében lehet detektálni azzal, hogy $\mathbf{1}l_{1..A}$ alakul ki.

Az itt bemutatott kiegyenlítő algoritmust az új rész-intervallum kiszámítása után kell lefuttatni, és amikor ez leállt (látható, hogy néhány lépésen belül ez megtörténik), akkor lehet a következő szimbólumot kódolni.

Az intervallumhatárokat A biten ábrázoló aritmetikai kódoló algoritmusa tehát a következő:

(1) Kezdetben legyen

$követőkSzáma$:=0

$l := 0^A$ vagyis az intervallum alsó határa: 0.00…0

$h := 1^A$ vagyis az intervallum felső határa: 0.11...1

(2) A következő elküldendő forrásszimbólum az x_k. Végrehajtjuk az intervallum szűkítését:

$r := h - l + 1$

$l := l + r \cdot c_{k-1}$

$h := l + r \cdot c_k$

(3) Ismételjük a kiegyenlítő eljárást addig, amíg az ír ki bitet.

(4) Ha van további szimbólum, akkor →(2)

(5) Befejezésül jelezzük a dekódernek, hogy az 1/4 vagy a 3/4 közelében álltunk-e le a következő módon:

$követőkSzáma := követőkSzáma + 1$

Ha $l < 1/4$ Akkor $BitKi(0,1^{követőkSzáma})$

Különben $BitKi(1,0^{követőkSzáma})$

Röviden összefoglalva az aritmetikai kódolás lépéseit:

1. Az üzenetekben előforduló szimbólumok előfordulási gyakoriságának meghatározása.
2. Minden szimbólumhoz hozzárendelünk egy 0 – 1 közé eső számtartományt. A számtartomány nagysága arányos a szimbólum relatív gyakoriságával.

3. A teljes karaktersorozatot egy számmá alakítjuk. Az átalakítást úgy végezzük, hogy az üzenetben egymás után következő karakterek által kijelölt számtartományt lépésről lépésre beszűkítjük, míg végül egy számhoz jutunk.

Ezzel a módszerrel 5-10%-kal jobb eredményt lehet elérni, mint a Huffmann kódolással.

CABAC

Statisztikai kódolóról lévén szó, először jó becslést kell adni az előforduló szimbólumok valószínűségi eloszlásáról. Ezt nevezzük a forrás modellezésének. Az adaptív jelző utal arra, hogy a modell dinamikusan változik, az előző szimbólumok figyelembe vételével határozza meg az adott szimbólum valószínűségét. Hátrány annyi lehet az adaptív kódolásnak, hogy a vevő oldalon is létre kell hozni a statisztikát. A kódolási modell és az adatok reprezentációja független egymástól. Az implementációnál fontos szempont az adott hardver fizikai tulajdonsága is, mint például a pontosság és a regiszterek mérete.

A CABAC [1],[5] jó tömörítési teljesítményt ér el, mert minden egyes szintaktikai elemhez az elem környezetének megfelelően választ a lehetséges modellek közül, a valószínűségi becslések a helyi statisztikákon alapulnak, és aritmetika kódolást használ.

Egy adatszimbólum kódolása a következő lépésekből áll:

1. Binarizáció: A CABAC bináris aritmetikát használ, ami azt jelenti, hogy csak bináris döntéseket (0 vagy 1) kódol. A nem bináris értékű szimbólumokat (pl. transzformációs együtthatók vagy mozgás vektor) először bináris kóddá kell konvertálni az aritmetikai kódoló részére.

A 2-es, 3-as és 4-es lépést a binarizált szimbólum minden egyes bitjére (binjére, bin: a binarizált szintaktikai elem egy bitje) ismételjük.

2. Környezeti modell kiválasztása: A környezeti modell egy lehetséges modell a binarizált szimbólum egy vagy több binje számára. Ezt a modellt választhatjuk az elérhető modellek közül a most kódolt szimbólum statisztikájától függően. A környezeti modell tárolja minden bin 0-val vagy 1-gyel való kezdődésének valószínűségét.
3. Aritmetikai kódolás: Az aritmetikai kódoló minden bin-t kódol a kiválasztott valószínűségi modell alapján. Megjegyzendő, hogy minden bin-re csak két altartomány van.
4. Valószínűségi frissítés: A kiválasztott környezeti modellt frissíteni kell az aktuálisan kódolt érték alapján.

Környezeti modell

A H.264 szabványban definiálva vannak a környezeti modellek és a binarizációs sémák minden szintaktikai elemre. Összesen 267 különböző modell létezik a változatos szintaktikai elemekre. Néhány modellnek a szelet típusától függően különböző felhasználása van.

A szelet kódolásának elején a kvantáló paraméter kezdeti értékétől függően inicializáljuk a környezeti modellt. Erre azért van szükség, mert a különböző adat szimbólumok előfordulásának valószínűségére alapvető hatása van a kvantáló paraméternek.

Az aritmetikai kódoló motor

A H.264 szabvány leír néhány részletet az aritmetikai kódolóról. Ennek 3 különböző tulajdonsága van:

- A valószínűség becslést egy átmeneti folyamat hajtja végre 64 különböző valószínűségi állapot között a legkevésbé valószínű szimbólum részére.
- Az aritmetikai kódoló aktuális állapotát reprezentáló R tartományt kvantáljuk elő-halmaz egy kis tartományával mielőtt minden lépésben kiszámoljuk az új tartományt, lehetővé téve az új tartomány kiszámítását egy kereső táblázatot használva.
- A közel egyforma valószínűségi eloszlású adatszimbólumok részére egyszerűsített kódoló és dekódoló folyamatot definiál.

CAVLC

A változó szóhosszúságú kódolás (Variable Length Coding, VLC) lényege, hogy a kevésbé valószínű forrásszimbólumokhoz hosszabb, a valószínűbbekhez rövidebb szimbólumsorozatot (kódszót) rendelünk. Legismertebb példa rá a már említett Huffmann-kód. Egy másik technika a futamhossz kódolás (Run-Length Coding, RLC), ami nagyon egyszerű eljárás. Számpárok sorozata lesz a kimenet, az első szám az értéket jelöli, a második pedig, hogy az előbbi értékből hány darab követi egymást.

A CAVLC-t [1],[6],[7] használjuk a maradékok kódolására, amit transzformációs együtthatók 4x4-es és 2x2-es blokkjainak cikk-cakk bejárásával (**1. ábra**) kapunk. A CAVLC-t úgy tervezték meg, hogy kihasználja a kvantált 4x4-es blokkok számos jellegzetességét:

1. Predikció, transzformáció és a kvantálás után a blokkok tipikusan ritkák, azaz többnyire nullákat tartalmaznak. A CAVLC futamhossz kódolást használ a nullák tömörebb ábrázolására.
2. A cikk-cakk bejárás után a legmagasabb nem-nulla együtthatók gyakran +/-1 szekvenciák. A CAVLC tömör módon jelzi a magas frekvenciás +/-1 együtthatók számát (vezető 1-es, Trailing 1s vagy T1).
3. A nem-nulla együtthatók száma a szomszédos blokkokban összefüggésben vannak. Az együtthatók számának kódolására kereső táblát használ. A kereső táblából való választás a szomszédos blokkokban található nem-nulla együtthatók számától függ.
4. A nem-nulla együtthatók szintje (nagysága) az újrarendezett tömb elején (a DC együtthatóhoz közel) magasabb, míg a magasabb frekvenciák irányában alacsonyabb. A CAVLC kihasználja ezt a nemrég kódolt szint nagyságától függő szint paraméterhez a VLC kereső tábla kiválasztásánál.

Most pedig nézzük meg, mi a transzformációs együtthatók blokkjának kódolási folyamata a CAVLC-ben.

Az együtthatók száma és a vezető egyesek (token) kódolása

AZ első VLC kódolta az összes nem-nulla együtthatók számát (ÖEH = összes együttható) és a vezető +/-1 értékek számát (T1). Az ÖEH bármi lehet 0-tól (nincs együttható a 4x4-es blokkban)[1] 16-ig (16 darab nem-null együttható). T1 bármi lehet 0-tól 3-ig. Ha több, mint 3 vezető +/-1-es van, speciális esetként csak az utolsó 3 kerül feldolgozásra és a többit normális együtthatóként kódoljuk.

A kereső táblának 4 választási lehetősége van a token kódolására, jelölje ezeket VLC0, VLC1, VLC2 és FLC (3 változó hosszúságú kódtábla és egy fix hosszúságú kód). A tábla választása a felső és bal oldali, előzőleg kódolt N_U és N_L blokkokban található nem-nulla együtthatók számától függ. Az N paraméter kiszámítása:

Ha az U és L blokkok elérhetőek (ugyanazon szeleten): $N = (N_U + N_L)/2$, ha csak az U blokk elérhető: $N = N_U$, ha csak az L blokk elérhető: $N = N_L$, ha egyik sem: $N = 0$.

N választ egy kereső táblát (**3. táblázat**) és így a VLC választása a szomszédos blokkokban kódolt együtthatók számától függ (környezet adaptív). A VLC0 az együtthatók

[1] Ez úgy lehetséges, hogy a kódolt 8x8-as blokknak lehet olyan 4x4-es részblokkja, amelyik nem tartalmaz együtthatót.

kis számának irányában elfogult. Az ÖEH alacsony értékeit (0 és 1) különösen rövid és a magas értékeit különösen hosszú kódok jelölik. A VLC1 az együtthatók közepes számának irányában elfogult (az ÖEH 2-4 körüli értékeit relatív rövid kódok jelölik), a VLC2 az együtthatók magasabb számának irányában elfogult és az FLC esetében az EÖH minden érétkét fix 6 bites kódszavak jelölik.

3. táblázat. Kereső táblázat választás token részére

N	token táblázat
0,1	VLC0
2,3	VLC1
4,5,6,7	VLC2
8 vagy afölött	FLC

T1 jelek kódolása

Minden T1-gyel (vezető +/-1-esek) jelölt token-re egy bit kódolja a jelet (0=+, 1=-). Ezek fordított sorrendben kerülnek kódolásra, a legmagasabb frekvenciájú T1-gyel kezdve.

A hátralévő nem-nulla együtthatók szintjeinek kódolása

A blokkban miden hátralévő nem-nulla együttható szintje (jel és nagyság) fordított sorrendben kerül kódolásra, kezdve a legmagasabb frekvenciával és haladva a DC együttható felé. A VLC tábla választása a szintek kódolására az egymást követő kódolt szintek nagyságától függ (környezet adaptív). 7 VLC tábla közül lehet választani VLC0-tól VLC6-ig. A VLC0 a kis, a VLC1 a némileg nagyobb nagyság felé elfogult és így tovább. A tábla választása a következők szerint történik:

1. Inicializáljuk a táblát a VLC0-ra (kivéve, ha több mint 10 nem-nulla együttható van és kevesebb, mint 3 vezető egyes, ebben az esetben VLC1 lesz a kezdő tábla).
2. Kódoljuk a legmagasabb frekvenciájú nem-nulla együtthatót.
3. Ha ennek az együtthatónak a nagysága magasabb, mint az előre definiált küszöbérték, lépjünk a következő VLC táblára.

Ebben az esetben a szint választása megegyezik a nemrég kódolt együtthatók nagyságával. A küszöbértékek listája megtalálható a **4. táblázatban**. Az első küszöb 0, ami azt jelenti, hogy a tábla mindig növekszik az első kódolt együttható szint után.

4. táblázat. Küszöbértékek, amelyek megmutatják, hogy a tábla számát növelni kell-e

Aktuális VLC tábla	Küszöb a táblaléptetéshez
VLC0	0
VLC1	3
VLC2	6
VLC3	12

VLC4	24
VLC5	48
VLC6	Nincs (legmagasabb tábla)

Nullák számának kódolása az utolsó együttható előtt

Az összes nullák számát, amelyik megelőzi a legmagasabb nem-nulla együtthatót az újrarendezett tömbben, VLC-vel kódoljuk. Annak az oka, hogy ezt a számot független VLC-vel küldjük el az, hogy sok blokk tartalmaz nem-nulla együtthatókat a tömb elején és ez a megközelítés jelenti azt, hogy tömb elején lévő 0 sorozatot nem szükséges kódolni.

A nulla sorozatok kódolása

Minden nem-nulla együtthatót megelőző nullák számát kódoljuk fordított sorrendben. Ezt a számot minden nem-nulla együtthatóra kódoljuk, kezdve a legmagasabb frekvenciával, két kivétellel:

1. Ha már nincs több 0 hátra, nem szükséges több ilyen paramétert kódolni.
2. Szintén nem szükséges ezt a számot kódolni az utolsó (legkisebb frekvenciájú) nem-nulla együttható esetében.

VLC választása a nulla sorozatokhoz a még nem kódolt és a megelőző nullák számától függ.

Mindkét eljárás (CABAC és CAVLC) jó hatásfokkal dolgozó, veszteségmentes tömörítést eredményez. Leglényegesebb különbség köztük, hogy a CABAC implementációja bonyolultabb, de cserébe 10% hatékonyabb tömörítés elérésére képes.

5.7 *További újdonságok és tulajdonságok*

Ebben a fejezetben azokat az eszközöket tekintjük át, amelyekről még nem volt szó. Ezek többsége olyan lehetőséget ad, amit nem tartalmaz minden profil. Tipikusan speciális igények kielégítését célozzák meg.

Egyik ilyen újítás, amikor egy képpont értékét nem 8 biten ábrázolják, hanem 10 vagy akár 14 biten. Ezzel is javítva a kódolt kép minőségét.

5.7.1 SVC (Scalable Video Coding, Skálázható videó kódolás)

Másik ilyen újítás az SVC, amely a H.264 egy kiegészítése. Az SVC szabvány [8] célja, hogy a magas minőségű videó bitfolyam egy vagy több rész bitfolyamból áll, amelyek

önmagukban is dekódolhatóak. Így lehetőség nyílik arra, hogy különböző számítási képességű és sávszélességű eszközökön is lejátszható ugyanaz a videó. A rész bitfolyam a nagyobb bitfolyamból származik csomagok eldobásával.

A rész bitfolyam alacsonyabb térbeli vagy időbeli felbontással vagy alacsonyabb minőségű videó jellel reprezentálható. A skálázásra a következő lehetőségek vannak:

- Időbeli skálázás: A mozgás kompenzáció miatt az egész frame eldobható a bitfolyamból. Ez a lehetőség már a H.264-ben benne van, az SVC csak kiegészítő információt ad a használatának tökéletesítéséhez.
- Térbeli skálázás: A videót többrétegű térbeli felbontással kódolják. A dekódolt alacsonyabb felbontású mintából jósolják meg a magasabb felbontású mintát, ezzel is csökkentve a bitrátát.
- SNR (Signal-to-Noise Ratio, jel/zaj viszony) / Minőségi / Pontosság skálázás: A videót egyetlen térbeli felbontással kódolják, de különböző minőségben. Az alacsonyabb minőséggel kódolt és dekódolt mintából jósolják meg a magasabb minőségű mintát ezáltal csökkentve a bitrátát.
- Kombinált skálázás: Az előző 3 skálázási lehetőség kombinálása.

Az SVC-t nagyobb sok videó alkalmazás átvette már. Ennek a technológiának az ötlete 2003-ban vetődött fel először, végleges verzió pedig 2007- ben jelent meg.

Az SVC szabvány 3 profilt tartalmaz:

- Alap profil: Elsősorban mobil és felügyeleti alkalmazások használják.
- High profil: Műsorszórási és tárolási alkalmazásoknál használatos.
- High intra profil: Professzionális alkalmazásoknál jelenik meg.

5.7.2 MVC (Multi-View Video Coding, Több nézőpontos videó kódolás)

A JVT (Joint Video Team) 2006 júliusa óta dolgozik ezen az új kiterjesztésen, ami az FTV (Free Viewpoint Television, szabad nézőpontú televízió) technológián alapul. Ennek az a lényege, hogy megengedné a felhasználónak, hogy interaktív módon változtathasson nézőpontot, ezáltal dinamikusan új képet kapva egy tetszőleges 3 dimenziós pozícióból. Ennek a kidolgozása még folyamatban van, nagyon sok problémát meg kell még oldani, mielőtt ilyen technológiával készült műsort nézhetnénk.

5.7.3 RDO (Rate Distortion Optimisation, Aránytorzításos optimalizálás)

Ezzel a technikával meg tudja keresni a videó tömörítő eljárás az intra és inter kódolás esetén a legjobb beállításokat az adott blokkhoz, kerethez. Nem egy H.264 által definiált technikáról van szó, csupán arról, hogy először a H.264 szabványába került be ez a döntési mechanizmus.

5.8 *Profilok*

A H.264 profilokat definiál különböző eszközkészlettel a különböző alkalmazási területekre. 2003-as megjelenésekor 3 profilt tartalmazott, azóta viszont újabbakkal bővült. Tekintsük át először a 3 eredeti (alap, fő és kiterjesztet), majd az azóta megjelent profilokat:

Alap (Baseline, BP) profil

Alkalmazási terület: Tipikusan alacsony számítási kapacitással rendelkező eszközökhöz, mobil és alacsony bitsebességen üzemelő videó-konferencia alkalmazásokhoz fejlesztették ki.
Eszközkészlet:

- *I* és *P* szeletek (*B* nem, pontosan a gyors számításigény miatt)
- blokkosodás elleni szűrő
- váltott soros és progresszív kódolás
- 1/4 képpont pontosságú mozgásvektorok
- makroblokk továbbosztás, fa struktúrájú szegmentáció 16x16-os blokkmérettől a 4x4-es blokkméretig
- CAVLC
- kép-félkép kódolás képenként meghatározható
- A hibavédelem elemei:
    - rugalmas makroblokk-sorrend (Flexible Macroblock Ordering, FMO)
    - képszelet csoportok (slice group), maximális száma 8
    - tetszőleges szelet-sorrend
    - redundáns szeletek

Fő (Main, MP) profil

Alkalmazási terület: Eredetileg médiatároláshoz és műsorszóráshoz fejlesztették, de funkcióját mára átvette a High profil.
Eszközkészlet:

- Az alap profil minden funkcióját tartalmazza a hibavédelem célú eszközök kivételével
- *B* szeletek
- *P* és *B* típusú mozgásbecslésnél súlyozott átlagolás

- CABAC
- kép-félkép kódolás makroblokkonként meghatározható

Kiterjesztett (Extended, XP) profil

Alkalmazási terület: Videó folyamok számára ajánlott profil, amely az alacsony bitsebesség mellett jól ellenáll a hálózati hibáknak és a szerver oldali folyam átkapcsolásnak.

Eszközkészlet:

- Az alap profil minden funkcióját tartalmazza
- *B* szeletek
- Átváltás másik bitfolyamra/gyorstekerés: *SP* / *SI* típusú képek
- További hibavédelmi funkció: adat particionálás

A H.264 továbbfejlesztése során kialakult profilok:

High (HiP) profil

Elsődlegesen tárolási és műsorszórási célokra, HDTV alkalmazásokhoz ajánlott profil. Ezt használja a HD DVD és a Blu-ray is. A fő profilt egészíti ki a HD tartalom hatékonyabb kódolása érdekében. Adaptív 4x4-es és 8x8-as transzformációt használ észleléses kvantálási mátrix engedélyezésével.

High 10 (Hi10P) profil

A High profilhoz képest sokkal nagyobb pontossággal (10 bit) tárolt együtthatókat használ, azáltal jobb kódolt képet eredményez.

High 4:2:2 (Hi422P) profil

Professzionális alkalmazások használják, mint például kamera és szerkesztő rendszerek. 10 bites mintavételezést használ és a 4:2:2 chroma formátumot támogatja.

High 4:4:4 (Hi444P) profil

A szintér transzformációból adódó hibák kiküszöbölésére lehetővé teszi a direkt RGB mintavételezést. 14 bites mintavételezést használ és a 4:4:4 chroma formátumot támogatja.

Ezeken felül a szabvány további 4 profilt definiál, amely a már ismertetett profilok részhalmaza, és csak intra kódolási mód (all-intra) megengedett bennük. Többnyire professzionális alkalmazások használják.

High 10 Intra profil: A High 10 profil leszűkítése all-intra mód használatára

High 4:2:2 Intra profil: A High 4:2:2 profil leszűkítése all-intra mód használatára

High 4:4:4 Intra profil: A High 4:4:4 profil leszűkítése all-intra mód használatára

CAVLC 4:4:4 Intra profil: A High 4:4:4 profil leszűkítése all-intra mód használatára és a CAVLC entrópia kódolás engedélyezése (de CABAC nem).

A skálázható videó kódolás (Scalable Video Coding, SVC) kiterjesztés eredményeként a szabvány 3 újabb profillal (skálázható profil) bővült. Ezek valamelyik korábbi profil és a skálázási eszközök kombinációját definiálják.

Skálázható alap profil: Elsődleges célja a videó-konferencia, mobil és felügyeleti alkalmazásokban való felhasználás. Az alap profilra épül, amelynek alkalmazkodónak kell lennie. Az elérhető skálázási eszközök egy részhalmazát használja.

Skálázható high profil: Elsődleges célja műsorszórási és videó streamelési alkalmazásokban való felhasználás. A high profilra épül, amelynek alkalmazkodónak kell lennie.

Skálázható high intra profil: Elsődleges célja gyártási alkalmazásokban való felhasználás. A Skálázható high profil leszűkítése all-intra mód használatára.

Az **1. függelékben** egy táblázatban összefoglalva megtalálható, hogy melyik profilban melyik eszköz található meg és melyik nem.

6. A H.264 teljesítménye

6.1 *Hibatűrés*

Az H.264 jó hibatűrése több fokozatban valósul meg, melyre a következő eszközöket használhatjuk – ezek egy részéről már volt szó korábbi fejezetekben:

ASO (Arbitrary Slice Ordering, tetszőleges képszelet sorrend): Mivel a képszeletek egymástól függetlenül dekódolhatóak, ezért tetszőleges sorrendben továbbíthatóak. Ez biztosítja, hogy nem egy teljes kép adata vész el egy burst-ös hiba (csomós hiba, szomszédos bitek vesznek el) esetén.

FMO (Flexible Macroblock Order, [9]): A makroblokkokat is tetszőleges sorrendben tárolhatjuk vagy továbbíthatjuk. Szerepe hasonló az ASO-hoz.

RS (Redunadant Slices): A redundáns keretek (*SP* / *SI*) segítségével lehet szinkronizálni a dekódert. Nem csak a teljes frame-et, hanem annak egy részét is lehet redundánsan tárolni (több módon kódolva). Tipikusan alacsony felbontás esetében használják.

DP (Data Partitioning): Az adatpartícionálás egy képszeleten belül úgy valósul meg, hogy a kódolt adat a szeleten belül három helyre tárolódik: *A*, *B* és *C* helyre. Az *A* helyen a szelet fejlécét, a szelet makroblokkjainak fejléceit tároljuk, a *B* helyen az *I* és *SI* képek különbségi adatait, a *C* helyen pedig a *P* és *B* makroblokkok adatait. Az *A* partíció feltétlenül szükséges a dekódoláshoz, de a *B* és *C* egyikének elvesztése már nem okoz leállást, mert csak egyikük dekódolásával már kaphatunk értékelhető képet.

A hibatűrésben mutatott robusztussága miatt jól használható a szabvány hálózati környezetben, ami egyik célkitűzés is volt a H.264 megtervezésekor.

6.2 *A H.264 videó tömörítéssel elérhető eredmények*

A veszteséges kódolást többféle módon is minősíthetjük, mérhetjük eredményességét.
Objektív úton: MSE (Mean Square Error, négyzetes középhiba) és PSNR (Peak Signal to Noise Ratio, jel-zaj arány).
Szubjektív úton: MOS (Mean Opinion Score, legfontosabb vélemény eredménye).

Olyan egzakt pszichofizikai mérőszám, amely a HVS (Human Visual System, emberi látórendszer) összes tulajdonságát figyelembe veszi jelenleg nincs.

Objektív képminőség mutató dekódolt képre:

PSNR n bites felbontásra:

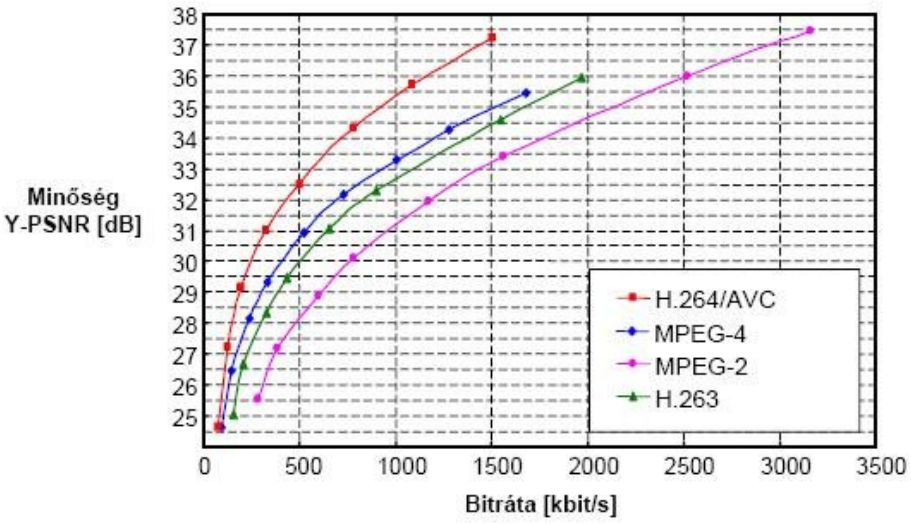
$$PSNR = 10 * \log (2^{2n} / MSE)$$

MSE a kódolás és dekódolás átlagolt négyzetes hibája:

$$MSE = 1 / XY \sum^{X}_{i=1} \sum^{Y}_{i=1} (x_{ij} - y_{ij})^2,$$

ahol X és Y a kép méretei (pl. képpontban), x_{ij} az eredeti, y_{ij} a rekonstruált képpont, n a komponensenkénti bitszám. A PSNR értékek mértékegysége a deciBel (dB). A nagyobb PSNR értékek nagyobb hasonlóságot jelentenek az egyes képek között.

Az **14. ábrán** látható PSNR mutatóval a H.264 által elért eredményt, minőség javulást.



14. ábra. A minőség változás PSNR mutatóval látványosnak mondható

A H.264-ben különböző szinteket különböztetünk meg felbontás és képátviteli sebesség szerint. A **2. függelékben** található táblázat mutatja az egyes szintekhez rendelt átviteli sávszélesség értékeket.

7. A H.264 verziói

A H.264 2003-as megjelenése óta sokat változott. Újabb és újabb verziók jelentek meg, amelyek a videó tömörítés hatékonyságát, skálázhatóságát hivatottak javítani, fejleszteni.

A H.264 szabvány verziói a következő átdolgozásokat, javításokat és kiegészítéseket tartalmazzák (zárójelben az ITU-T végleges jóváhagyásának dátuma szerepel, természetesen minden egyes verzió tartalmazza a megelőző verziók újdonságait):

1. (2003. május) A H.264/AVC első elfogadott verziója, amely a Baseline, Extended és Main profilokat tartalmazza.
2. (2004. május) Helyesbítés, amely különféle kisebb korrekciókat tartalmaz.
3. (2005. március) Az első jelentősebb kiegészítést tartalmazó verzió. Itt jelent meg a FRExt (Fidelity Range Extensions) kiterjesztés, amely a High, High 10, High 4:2:2, és a High 4:4:4 profilokat tartalmazza.
4. (2005. szeptember) Különböző, kevésbé jelentős javításokat tartalmaz, valamint 3 szempont aránymutatóval bővült a szabvány.
5. (2006. június) A korábbi High 4:4:4 profil került eltávolításra ebben a verzióban (az ISO/IEC helyesbítését feldolgozva)
6. (2006. június) Ez a verzió kevésbe jelentős kiterjesztéseket tartalmaz, mint például a kiterjesztett skála színtér támogatást (összekötve a fent említett szempont aránymutatókkal az ISO/IEC-ben)
7. (2007. április) Ez a verzió már tartalmazza a High 4:4:4 Predictive és a 4 Intra-only profilt (High 10 Intra, High 4:2:2 Intra, High 4:4:4 Intra, és CAVLC 4:4:4 Intra)
8. (2007. november) Jelentős bővítést hajtottak végre ebben a verzióban a skálázható videókódolás (Scalable Video Coding, SVC) és a Scalable Baseline, Scalable High és a Scalable High Intra profilok bevezetésével.

<u>További tervezett bővítések:</u>

- Különböző kevésbé jelentős javítások
- Több nézetes videó kódolás (Multi-View Coding, MVC): Lényege, hogy amikor egy videó anyagot több kamera váltogatásával készítenek, akkor ha a különböző kamerák képeit lehetne egymás után kódolni, lényegesen csökkenthető lenne a bitráta.

Ezek a tervezett kiterjesztések fejlesztése természetesen folyamatban van, a közeljövőben számíthatunk ezek megjelenésére. Valószínűleg még itt sem áll majd meg a H.264 bővítése, fejlesztése, hogy minél kisebb sávszélesség igény mellett minél jobb minőséget lehessen elérni majd a jövőben.

8. Alkalmazási területek

Mint már említettem, a H.264 egy eredetileg alacsony bitsebességre tervezett kódoló, olyan területekre, mint például videokonferencia alkalmazások kódolására, videó-telefon analóg vonalon, elektronikus újságok stb. Megjelenése óta azonban minden multimédiával foglalkozó területen megjelent, és egyre nagyobb előszeretettel alkalmazzák. A teljesség igénye nélkül néhány főbb alkalmazási terület:

- CATV (Cable TV): kábel TV optikai hálózaton, rézvezetéken, stb.
- DBS (Direct Broadcast Satellite): direkt műsorszórás műholdon keresztül.
- DSL (Digital Subscriber Line): digitális előfizetői hurok.
- DTTB (Digital Terrestrial Television Broadcasting): digitális földi televízió sugárzás.
- ISM (Interactive Storage Media): interaktív médiatárolás például optikai lemezen.
- MMM (Multimedia Mailing): multimédialevelezés.
- MSPN (Multimedia Services over Packet Networks): multimédiaszolgáltatások csomagkapcsolt hálózaton.
- RTC (Real-time Conversational): valós idejű szolgáltatások, pl. videokonferencia, videotelefonálás, stb.
- RVS (Remote Video Surveillance): távoli videó felügyelet.
- SSM (Serial Storage Media): soros médiatárolás, pl. digitális VTR (video tape recorder, képmagnó, olyan funkciók támogatása, mint pl. a pillanat-állj, gyors-előre, gyors-hátra).
- IP kamera: 2008-ban az Axis kifejlesztette az első olyan IP kameráját, amely már H.264-et használ a videó tömörítésre.

A DVD utódja, a Blu-Ray Disc a H.264 High profilt alkalmazza és a lejátszókba kötelező funkció lesz a H.264 formátumú videó lejátszása. A DVB (Digital Video Broadcast, Digitális Videó Műsorszórás) szabványok többsége elfogadta a H.264-et műsorszórási formátumként Európában. A francia miniszterelnök Jean-Pierre Raffarin 2004-ben követelményként jelölte meg a H.264 támogatását HDTV eszközökben, Franciaországban. Japánban, a mobil szegmensben a legnagyobb szolgáltatók (NHK, Tokyo Broadcasting System (TBS), Nippon Television (NTV), TV Asahi, Fuji Television, TV Tokyo) a H.264-et fogják használni. A direkt műholdas TV szolgáltatók (DirecTV (Egyesült Államok), Dish

Network (Egyesült Államok), Euro1080 (Európa), Premiere (Németország), ProSieben HD & Sat1 HD (Németország), BSkyB (Egyesült Királyság Írország)) szintén a H.264 mellett rakták le voksukat. A NATO belső katonai használatra szintén bevezeti a H.264-et. Az Internet Engineering Task Force (IETF) kidolgozta a H.264 videók átvitelét Real-time Transport Protocol (RTP) segítségével. Az Apple termékek (iTunes videó letöltő, iPod videó, MacOS) szintén alkalmazzák a H.264-et.

Az internetet elárasztó videók trendje már ma is olyan erős, hogy szakmai becslések szerint 2011-ig a mozgóképek mennyisége legalább ötször nagyobb lesz, mint ma. A televíziós szakemberek a webvideót csupán a tömegeket vonzó, szórakoztató eszköznek tekintették, amely azonban semmiképp nem adja vissza a TV teljes képernyős varázsát, elsősorban műszaki okok miatt. Ez igaz is volt 2008-ig. Az USA-ban és Európában hadrendbe állították egy új Adobe szervert (FMS3) amely támogatja a H.264-es kódolású .MP4 fájltípust, amelyet videóként a világszerte legjobban elterjedt médialejátszó, az Adobe Flash Player 9 jelenít meg.. Ezek a szerverek nem csupán a honlapok videónak színvonalát hivatott javítani, de jelentős változást hozhat az online reklámipar számára is.

8.1 *Az MPEG-4 AVC támogatottsága*

Elmondható, hogy számítógépes környezetben az AVC támogatottsága szinte teljes. Kezdve a legnépszerűbb lejátszó programoktól, a széles körben használt videó szerkesztő programokon át a kereskedelemben kapható videó-digitalizáló kártyákig szinte mindegyik támogatja az MPEG-4 AVC-t. A legfontosabb lépés az volt, hogy 2002-től a számítógépes berkekben messze legnépszerűbb médiakezelő program, az Apple Quicktime Player 7. verziójától tartalmaz AVC dekódert is, ezzel meglódult az AVC népszerűsége. Ugyanekkor, mikor már látszott, hogy az MPEG-4 meg fog jelenni számos egyéb médián, mi több, az interneten és egyéb műsorterjesztő hálózaton is életképes, számos médiában és az informatikában érdekelt cég – Apple, Cisco, IBM, Kasenna, Phillips és a Sun - összeállt, és megalakította az ISMA-t (Internet Streaming Media Alliance, internetes folyamatos átvitelű multimédiás anyagok), hogy kifejlessze a streaming média egyes szereplői közötti szabványos együttműködés profiljait. Később még 20-30 vezető informatikai és médiacég csatlakozásával biztosították, hogy bármely MPEG-4-es média-stream megjeleníthető legyen a többiek lejátszóin. A támogatottságot növelte az is, hogy a vezeték nélküli multimédiás szabványok, a

3GPP (3rd Generation Partnership Project) és a 3GPP2 már az MPEG-4-es szabványra épültek. Végül a műholdas digitális műsorszolgáltatók is elkezdtek felsorakozni a szabvány mögé, és a DirecTV és a DVB is adaptálta a tömörítési eljárást. Legutoljára a HD-DVD, a nagyfelbontású DVD és a Blue Ray specifikációi közé került be az AVC. Ezzel vált teljessé a paletta, a kis bitsebességű mobil eszközöktől a legnagyobb bitsebességű HD-DVD megoldásokig mindenütt az MPEG-4-et használják.

A digitális jogkezelés

A fent említett ISMA szervezet volt az, amely kidolgozta a digitális tartalomszórás szerzői jogainak védelmét szolgáló DRM (Digital Rights Management) nyílt szabványt, és nyilvánossá tette a Encryption and Authentication specification V1.0 című kiadványában. Itt részletesen tárgyalja a titkosító eljárásokat, a titkosítás és a feloldás algoritmusait, az átviteli csatorna jelzésrendszerét és az MPEG-4 fájl struktúráját. Mivel az összefogás is igen széleskörű mely az ISMA-t létrehozta, az általuk kidolgozott tartalomvédelmi eljárások is elterjedtek. Ma a V2.0-ás verzió az aktuális, és nyílt ajánlásként használják a tartalomszolgáltatók

9. Eredmények, tapasztalatok

A diplomamunka elején említettem, hogy a cégünknél jelenleg egy H.264 enkóder – dekóder páros implementációján dolgozunk. Ebben a fejezetben erre térnék ki, hogy milyen eredményeket értünk el, milyen tapasztalatokat szereztünk a fejlesztések során.

Az alkalmazás külföldi megrendelésre készül és első körben általános felhasználási céllal, az alap profil kerül elkészítésre. Ha ezzel elkészültünk, a megrendelőnk csak ezután dönt a továbbiakról, hogy milyen területre, konkrét felhasználásra specializáljuk a program működését. A legvalószínűbb, hogy a DVB-S technológiájába szeretné majd adoptálni, vagyis a digitális műholdas műsorszórásba.

Az implementáció ANSI C-ben készül. Később majd a kritikus, számításigényes műveleteket assembly modulokra fogjuk cserélni a hatékonyabb működés érdekében. Az enkóder és dekóder fejlesztése párhuzamosan zajlik, így a program tesztelése, a visszacsatolás folyamatosan történik.

Eddig elkészült az enkóder és a dekóder keretrendszere, előálltak a szükséges struktúrák, paraméter halmazok. Az intra predikció és inverzének összes fajtájának kódolása készen van, de a döntési mechanizmuson még finomítani kell, mert sok esetben még nem az optimális választ adja. Az Integer transzformációt és inverzét szintén elkészítettük már. A CAVLC és inverz folyamata közel hiba nélkül működik már.

Jelenleg az inter predikción dolgozunk, ami elég bonyolult és összetett feladat. Referenciaként elkészítettünk egy megadott sugarú környezetben, egy képen belül dolgozó teljes keresőt. Ez értelemszerűen egy időigényes művelet. Egy teljes kép ilyen eljárással történő feldolgozása a kép méretétől függően több percig, akár 10 percig is eltarthat. Ez elfogadhatatlan, mivel egy enkódernek másodpercenként akár 50-60 képet is fel kell tudnia dolgozni, valamint egyszerre nem csak egy, hanem maximálisan 32 korábbi képen keresnie a leghasonlóbb képrészletet. Az persze egy külön döntési mechanizmus, hogy mekkora sugarú környezetben és milyen méretű blokkhoz keresünk. Az algoritmus assembly nyelvre történő átírása során figyelni kell még arra is, hogy a szoftvert futtató platform hány bites aritmetikával dolgozik. Szem előtt kell még tartani a memória méretét, mekkora struktúrákat és mennyi adatot tarthatunk a gyorsabb feldolgozás érdekében a memóriában.

A teljes keresőn kívül már több egyszerű algoritmust is implementáltunk, mint pl. a kétdimenziós logaritmikus, az ortogonális vagy a gyémánt kereső algoritmusokat. Most olyan

megoldáson dolgozunk, ami az ilyen algoritmusok előnyeit ötvözni tudja. Ez nagyon sok fejlesztést és tesztelést igényel. Napjainkban már számos ilyen algoritmus létezik, nagy részük nyilvános, bárki által elérhető. Ezek többsége nagyon hasonlít egymásra, kis részletekben térnek el. Mi is merítünk ezekből ötleteket, de egy saját fejlesztésű algoritmuson dolgozunk.

Többek között az Integer transzformáció fejlesztése is az én feladatom volt. Először elkészítettem egy diszkrét koszinusz transzformációt az ellenőrzés miatt, mivel ennek ugyanazt a kimenetet kell produkálnia. Ezután az Integer transzformációt, melyben még szorzás és osztás műveletek is szerepelnek, végül pedig a H.264 által ajánlott transzformációt. Ez utóbbi kettőnél nagyon látványos volt a gyorsulás, sok esetben több mint tízszeres, amellett hogy a pontosság megmaradt. Ez annak volt köszönhető, hogy szorzás és osztás műveletek helyett a bit eltolás műveletét alkalmaztuk. Az inverz transzformáció segítségével, a kvantáló paramétertől függően átlagosan 95-99%-ban sikerült visszaállítani az eredeti bemeneti adatokat. Viszont ha nagyon magas kvantáló paramétert használtunk, akkor magas tömörítési arányt tudtunk elérni, ami a visszaállíthatóság rovására ment.

Implementációs kódrészleteket sajnos nem áll módomban mellékelni, ugyanis az a cég kizárólagos tulajdonát képezik, és nem járult hozzá a közzétételhez.

Az inter predikció kidolgozásával egy időben készülnek olyan kisebb-nagyobb modulok, mint pl. a blokkosodás elleni szűrő, a szeletek kezelése vagy a váltott soros és progresszív kódolás szétválasztása. Ezután már csak a modulok integrálását kell megoldani.

10. Összefoglalás

Ezennel végére is értünk a H.264 tárgyalásnak. Most tekintsük át röviden, milyen új eszközökkel, újításokkal is találkozhattunk a legfrissebb videó tömörítési eljárás megismerés során:

- Fejlettebb intra becslési megoldás
- Erősebb mozgásbecslési képesség:
    - dinamikus blokkra bontás / fa struktúrájú makroblokk szegmentálás: 16x16-tól lefelé egészen 4x4-es blokkméretig
    - 1/4 képpont pontosságú mozgásbecslés
- Több referenciakép használata
- Súlyozott képirányú becslés
- Kontextus adaptív VLC és aritmetikai kódolás
- Egész értékű transzformáció
- Beépített blokkosodás elleni adaptív szűrés
- Hálózati környezetre való adaptálhatóság

A fenti eszközöknek köszönhetően egyrészt jelentős bitsebesség megtakarítás (akár 50%) tapasztalható a többi szabványhoz képest. Ugyanakkor ez a komplexitás növekedésével is jár a korábbi módszerekhez képest, a kódolás 3-szor, a dekódolás 2-szer nagyobb erőforrás igényű a többi szabványhoz képest.

Az elért eredmények tükrében érthető, hogy miért annyira népszerű a H.264. Megfigyelhető, hogy minden területen, ahol videó tömörítést használnak, sorra megjelenik az új tömörítési eljárás, és szép lassan átveszi a vezető szerepet. Sőt a tömörítési arány növekedésével és a nagyobb teljesítményű architektúráknak köszönhetően új területek és lehetőségek is megnyílnak.

11. Irodalomjegyzék

[1] ITU-T Rec. H.264 / ISO/IEC 11496-10, „Advanced video coding for generic audiovisual services”, 03/2005.

[2] A. Hallapuro and M. Karczewicz, “Low complexity transform and quantization – Part 1: Basic Implementation”, JVT document JVT-B038, February 2001

[3] JVT Reference Software version

[4] Iain E G Richardson, “H.264 and MPEG-4 Video Compression”, John Wiley & Sons, to be published late 2003

[5] D. Marpe, G Blättermann and T Wiegand, “Adaptive Codes for H.26L”, ITU-T SG16/6 document VCEG-L13, Eibsee, Germany, January 2001

[6] JVT Document JVT-C028, G.Bjontegaard and K. Lillevold, “Context-Adaptive VLC Coding of Coefficients”, Fairfax, VA, May 2002

[7] JVT Document JVT-L13, D. Marpe, G. Blattermann and T. Wiegand, “Adaptive Codes for H.26L”, Eibsee, January 2001

[8] Overview of the Scalable Video Coding Extension of the H.264/AVC Standard Heiko Schwarz, Detlev Marpe, Member, IEEE, and Thomas Wiegand, Member, IEEE, VOL. 17, NO. 9, September 2007

[9] Y.Dhondt, P.Lambert: Flexible Macroblock Ordering, an error resilience tool in H.264/AVC. Fifth FTW PhD Symposium, Faculty of Engineering, Ghent University, Desembre 2004, Paper No. 106

[10] Search Algorithms for Block-Matching in Motion Estimation, Deepak Turaga, Mohamed Alkanhal, Mid-Term project, 18-899, Spring, 1998

12. Függelék

1. függelék – Eszközök a különböző profilokban[2]

	Baseline	Extended	Main	High	High 10	High 4:2:2	High 4:4:4 Predective
I és P szeletek	I	I	I	I	I	I	I
B szeletek	N	I	I	I	I	I	I
SI és SP szeletek	N	I	N	N	N	N	N
Több referenciaképes predikció	I	I	I	I	I	I	I
In-Loop Deblocking Filter	I	I	I	I	I	I	I
CAVLC	I	I	I	I	I	I	I
CABAC	N	N	I	I	I	I	I
FMO	I	I	N	N	N	N	N
ASO	I	I	N	N	N	N	N
RS	I	I	N	N	N	N	N
DP	N	I	N	N	N	N	N
Interlaced Coding	N	I	I	I	I	I	I
4:2:0 chroma fomrátum	I	I	I	I	I	I	I
Monokróm videó formátum (4:0:0)	N	N	N	I	I	I	I
4:2:2 chroma formátum	N	N	N	N	N	I	I
4:4:4 chroma formátum	N	N	N	N	N	N	I
8 bites kódolás	I	I	I	I	I	I	I
9 és 10 bites kódolás	N	N	N	N	I	I	I
11-14 bites kódolás	N	N	N	N	N	N	I
Adaptív 4x4-es és 8x8-as transzformáció	N	N	N	I	I	I	I
Külön luma és chroma QP vezérlés	N	N	N	I	I	I	I
Külön színtér kódolás	N	N	N	N	N	N	I
Predictive veszteségmentes kódolás	N	N	N	N	N	N	I

[2] I: Igen, N: Nem

2. függelék – Áviteli sávszélesség az egyes szinteken

Sor-szám	Max. makroblokk másod-percenként	Max. frame méret makro-blokkban	Max. bitsebesség Baseline, Extended és Main profil esetén	Max. bitsebesség High profil esetén	Max. bitsebesség High10 profil esetén	Max. bitsebesség High 4:2:2 és High 4:4:4 profil esetén	Felbontás / képváltási sebesség (max. tárolt frame)
1	1 485	99	64 kbit/s	80 kbit/s	192 kbit/s	256 kbit/s	128x96 / 30.9 (8)
							176x144 / 15.0 (4)
1b	1 485	99	128 kbit/s	160 kbit/s	384 kbit/s	512 kbit/s	128x96 / 30.9 (8)
							176x144 / 15.0 (4)
1.1	3 000	396	192 kbit/s	240 kbit/s	576 kbit/s	768 kbit/s	176x144 / 30.3 (9)
							320x240 / 10.0 (3)
							352x288 / 7.5 (2)
1.2	6 000	396	384 kbit/s	480 kbit/s	1152 kbit/s	1536 kbit/s	320x240 / 20.0 (7)
							352x288 / 15.2 (6)
1.3	11 880	396	768 kbit/s	960 kbit/s	2304 kbit/s	3072 kbit/s	320x240 / 36.0 (7)
							352x288 / 30.0 (6)
2	11 880	396	2 Mbit/s	2.5 Mbit/s	6 Mbit/s	8 Mbit/s	320x240 / 36.0 (7)
							352x288 / 30.0 (6)
2.1	19 800	792	4 Mbit/s	5 Mbit/s	12 Mbit/s	16 Mbit/s	352x480 / 30.0 (7)
							352x576 / 25.0 (6)
2.2	20 250	1 620	4 Mbit/s	5 Mbit/s	12 Mbit/s	16 Mbit/s	352x480 / 30.7(10)
							352x576 / 25.6 (7)
							720x480 / 15.0 (6)
							720x576 / 12.5 (5)
3	40 500	1 620	10 Mbit/s	12.5 Mbit/s	30 Mbit/s	40 Mbit/s	352x480 / 61.4 (12)
							352x576 / 51.1 (10)
							720x480 / 30.0 (6)
							720x576 / 25.0 (5)

Sor-szám	Max. makroblokk másod-percenként	Max. frame méret makro-blokkban	Max. bitsebesség Baseline, Extended és Main profil esetén	Max. bitsebesség High profil esetén	Max. bitsebesség High10 profil esetén	Max. bitsebesség High 4:2:2 és High 4:4:4 profil esetén	Felbontás / képváltási sebesség (max. tárolt frame)
3.1	108 000	3 600	14 Mbit/s	17.5 Mbit/s	42 Mbit/s	56 Mbit/s	720x480 / 80.0 (13)
							720x576 / 66.7 (11)
							1280x720 / 30.0 (5)
3.2	216 000	5 120	20 Mbit/s	25 Mbit/s	60 Mbit/s	80 Mbit/s	1280x720 / 60.0 (5)
							1280x1024 / 42.2 (4)
4	245 760	8 192	20 Mbit/s	25 Mbit/s	60 Mbit/s	80 Mbit/s	1280x720 / 68.3 (9)
							1920x1088 / 30.1 (4)
							2048x1024 / 30.0 (4)
4.1	245 760	8 192	50 Mbit/s	50 Mbit/s	150 Mbit/s	200 Mbit/s	1280x720 / 68.3 (9)
							1920x1088 / 30.1 (4)
							2048x1024 / 30.0 (4)
4.2	522 240	8 704	50 Mbit/s	50 Mbit/s	150 Mbit/s	200 Mbit/s	1920x1088 / 64.0 (4)
							2048x1088 / 60.0 (4)
5	589 824	22 080	135 Mbit/s	168.75 Mbit/s	405 Mbit/s	540 Mbit/s	1920x1088 / 72.3 (13)
							2048x1024 / 72.0 (13)
							2048x1088 / 67.8 (12)
							2560x1920 / 30.7 (5)
							3680x1536/26.7 (5)
5.1	983 040	36 864	240 Mbit/s	300 Mbit/s	720 Mbit/s	960 Mbit/s	1920x1088 / 120.5 (16)
							4096x2048 / 30.0 (5)
							4096x2304 / 26.7 (5)