LADÁNYI PÉTER (PATKÓ TAMÁS, DR. NAGY TAMÁS, MÁTHÉ JÓZSEF)

Digitális videorendszerek

II. A digitális képek tömörítése

Bevezetés

A digitális videorendszerek által készített képi információ tárolása és kezelése mindig komoly kihívást jelentett a témával foglalkozó szakemberek számára. Azonban a keletkező adatmennyiség nagysága miatt a digitális rendszerek, vitathatatlan előnyeik mellett még mindig csak rövid ideig képesek elérni az analóg rendszerek valós idejű képrögzítését. Megoldást jelenthet az információ megfelelő kódolása, tömörítése, mellyel a tárolt adatok mennyisége jelentősen csökkenthető.

1. A digitális képi információ nagysága

Tételezzük fel, hogy egy digitális videorendszer összes egysége és hardver eleme képes a képeket valós időben folyamatosan feldolgozni és tárolni. Egy PAL valós idejű videojel rögzítéséhez szükséges képek száma másodpercenként 25, percenként 25*60=1500, óránként 25*60*60=90000. Ezt beszorozva egyetlen kép tárolásához szükséges memóriaigénnyel, a különböző felbontások szerint, az 1. táblázatban szereplő eredményeket kapjuk.

Grafikai felbontás	***Inzenzitás (szín-) mélység***	***Egyetlen kép memóriaigénye***	***1 óra memóriaigénye***	***1 nap memóriaigénye***
384(H) X 288(V)	8 bites szürkeskálás	108 KB	9 492 MB	222 GB
384(H) X 288(V)	24 bites RGB	324 KB	28 477 MB	667 GB
768(H) X 576(V)	8 bites szürkeskálás	432 KB	37 969 MB	890 GB
768(H) X 576(V)	24 bites RGB	1 296 KB	85 430 MB	2 670 GB

1. táblázat

Látható, hogy a grafikai felbontás, a színmélység és a képformátum függvényében a tárolandó képek memóriaigénye között nagyságrendi különbségek lehetnek (példánkban 1:12).

A másik jelentős kérdés az, hogy a keletkező óriási adamennyiség hogyan kerül feldolgozásra. Hogyan történik az archíválás és hogyan kell a kezelőszemélyzetnek biztosítani a folyamatos rögzítést? Az analóg technikával kapcsolatos gyakorlat, hogy a keletkező videokazettákat egy meghatározott ideig biztos helyen tárolják. Ez hogyan valósítható meg a digitális eszközökkel? Ezekre a kérdésekre a legtöbb rendszer megoldást kínál, viszont egyik sem nélkülözi a képtömörítési eljárásokat. Ezen algoritmusok nélkül a digitális videorendszerek valószínűleg meg sem születtek volna.

2. A tömörítésről

A fentiekből látható, hogy a képet feldolgozó, tároló és kezelő eszközök az óriási adatmennyiség folyamatos valós idejű kezelésére ma még képtelenek, ezért az adatmennyiség csökkentése az egyedüli megoldás, viszont ennek számos hátránya van. A képi (mint bármilyen más digitális) információ bizonyos esetekben tömöríthető, ami azt jelenti, hogy egy alkalmas eljárással az információ memóriaigénye lecsökkenthető, majd amikor szükséges, egy inverz művelettel az eredeti információ visszaállítható. Látni fogjuk, hogy a visszaállított (kitömörített vagy kicsomagolt) adatok az eredetivel nem minden esetben egyeznek meg, bizonyos tömörítések használatakor csak "hasonlítanak" egymásra. Azonban a nagyfokú hasonlóság és az emberi érzékelés működése biztosítják a képi információ változatlan kiértékelhetőségét, és a nagyfokú tömörítés révén a tárolhatóságot.

A különböző tömörítési algoritmusok és módszerek célja éppen az, hogy a képi információt leíró adatmennyiséget a lehető legjobban lecsökkentsék, és az alkalmazott tömörítés hátrányait megszüntessék vagy arra lehetőséget, megoldást adjanak. A tömörítési algoritmusokban legtöbbször nem csak egyetlen módszert, hanem azok kombinációit alkalmazzák.

2.1 Az adatok számának csökkentése

Egy tömörítetlen adatcsoportban számos felesleges, megismételt (redundáns) adatminta található. Ezeket a mintákat az eredeti adatcsomagból eltávolítva és azokat ügyesen kódolva a keletkező adathalmaz mérete csökkenthető. Ha egy adathalmaz nem tartalmaz ilyen mintákat, teljesen véletlenszerű (totally random) és ezért ezzel a módszerrel nem tömöríthető.

2.2 A pontosság csökkentése

Csökkenteni lehet a pontosságot, ha az egyes adatokat leíró bitek számát csökkentik. Ezáltal minden adat tárolásához kevesebb bitre van szükség, így a keletkező adathalmaz eredő mérete is kisebb lesz. Ezzel a módszerrel az adathalmaz véletlenszerűsége is csökkenthető, gondoljunk csak például a digitalizált adatban jelenlévő zajra.

2.3 Az adathalmazt leíró bitek számának csökkentése

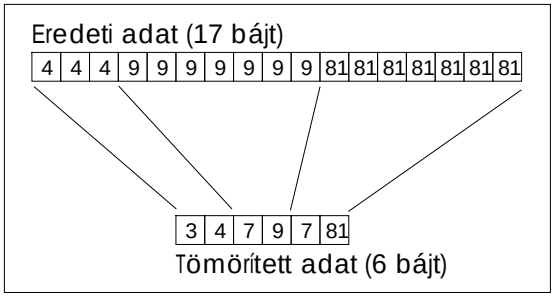
Számos algoritmus található a szakirodalomban erről a technikáról. A lényege az, hogy az adathalmaz egészét leíró bitek számát kell csökkenteni oly módon, hogy az információ megmaradjon, tehát a kicsomagolás után az eredeti adathalmazt lehessen visszanyerni. Két jelentős csoport létezik aszerint, hogy egyenlő (Fixed-length Codes), vagy változó hosszúságú kódokat (Variable-length Codes) alkalmaznak.

3. Adatvesztés nélküli tömörítések

A tömörítési algoritmusok egyik lényeges csoportja, mely tömörítések közös jellemzője, hogy a visszaállított digitális információ teljes mértékben megegyezik az eredetivel, ezért a képi információn kívül szinte tetszőleges adat tömörítésére alkalmazható. Az összes tömörítő-archiváló programcsomag alapja valamilyen adatvesztés nélküli tömörítés (ARC, ARJ, ZIP, RAR, stb.). A maximálisan elérhető tömörítési arány 1.5 - 3.0, így digitalizált képek tömörítéséhez nem adnak igazán jó megoldást.

3.1 Futamhossz kódolás (RLE)

A tömörítés lényege, hogy egy adathalmazban az egymás után következő azonos értékű bájtokat csupán kettővel helyettesíti: az ismétlés számával és az ismétlődő adat értékével (3.1 ábra).



3.1. ábra

Formátum	*Elnevezés*
CCITT Fax	Group 3
CGM	-
DDES-UEF01	
DDES-UEF03	
GEM/IMG	-
MacPaint	PackBits
HDF	DFTAG_RLE
PCX	-
PostScript	RunLengthEncode, RunLengthDeCode
RLE	RLE
SunRaster	-
TARGA	Type-9, 10, 11, 32, 33
TIFF	Type-2, 32773

3.1 táblázat

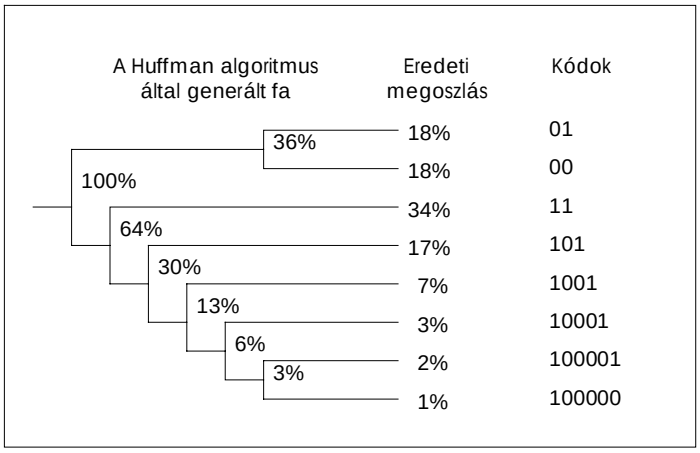
Ebből két dolog következik. Az egyik, hogy csak kettőnél több ismétlődés esetén van értelme a tömörítésnek. A másik, hogy biztosítani kell, hogy a nem ismétlődő bájtok leírásához ne kétszer annyi bájt kelljen. Ezt például egy ún. "Escape" karakterrel lehet megoldani, ami egy speciális kód, mely jelzi, hogy ismétlődő sorozat következik. Például a PackBits algoritmusnál, ha ismétlődő adatok következnek, akkor negatív, ha eltérőek, akkor pozitív hossz-adatot tárolnak el. Így a kitömörítő algoritmus a soron következő hossz-bájt előjele szerint fogja kicsomagolni az ismétlődő, vagy csak egyszerűen kimásolni az eltérő adatokat.

Mivel az algoritmus binárisan teljesen egyező bájtokat képes tömöríteni, ezért az algoritmus a digitalizált képekre csak igen korlátozottan alkalmazható, mivel azok általában zajosabb és finomabb felbontású adatokat tartalmaznak (megfelelő bitmélység esetén). Számos esetben azonban (rajzolt képek, fax üzenetek, stb.) igen jó hatékonysággal alkalmazható módszer, éppen ezért igen sok változata megtalálható, melyeket a 3.1. táblázatban foglaltuk össze. A különböző megvalósítások a kódolásban ugyan eltérőek lehetnek, de a tömörítés alapelve ugyanaz.

3.2 Huffman kódolás

A szerzőjéről, David Albert Huffman-ról (első publikáció: 1952) elnevezett kódolás, amely a Shannon-Fano kódolás továbbfejlesztése. Mindkettőnek alapja, hogy változó hosszúságú kódokkal írják le az adathalmaz összes elemét: a gyakoribb adatokat rövidebb kóddal, a ritkábbakat hosszabb kóddal. Így a keletkező adathalmaz összességében rövidebb lesz, mint az eredeti.

A tömörítéshez először statisztikát kell készíteni az előforduló adatelemek gyakoriságáról, mely a tömörítés alapjául szolgál. A tömörítés során tulajdonképpen keletkezik egy fa, melynek végpontjain az egyes adatok kódjai szerepelnek, ez az ún. Huffman kódtáblázat. A tényleges adatok csak ezután következnek, a kódtáblázat alapján a többé-kevésbé gyakoribb mintákat a rövidebb-hosszabb kódokkal kell helyettesíteni. Tehát a Shannon-Fano és a Huffman kódnál is a gyakoriságok szerint kell a bit-kódokat kiosztani. A Shannon-Fano kódnál ez a következőképpen történik.



3.2. ábra

Formátum	*Elnevezés*
CCITT Fax	Group 3
JPEG	Huffman
MPEG	Huffman
TARGA	Type-32. Type-33

3.2 táblázat

A statisztikát (sorbarendezés után) ketté kell osztani oly módon, hogy a két csoport közelítőleg egyforma összgyakoriságú legyen (kb. 50-50%). Ezután mindkét csoport kap egy-egy bitet, a gyakoribb adatokat tartalmazó csoport egyest, a másik a nullát. Ezután az így keletkező két csoportot újra ketté kell osztani az előbb ismertetett módon addig, amíg minden különálló adat kódot kap.

A Huffman kódtáblázat elkészítése egy kicsit másként történik. Ugyancsak rendezett statisztikára van szükség, viszont a fa felépítése úgy történik, hogy a fa aljára egy új csomópontot hozva létre a legkisebb valószínűségű adat kapja az egyes bitet (a csomópont egyik ága), a második legkisebb a nullát (a másik ág). Ezt követően a két értéket a táblázatból törölve és összeadva, az összeget a táblába beillesztve a folyamatot itt is addig kell ismételni, amíg minden adat megkapja az őt meghatározó bitsorozatot (3.2. ábra). Ezt követően már csak a behelyettesítés van hátra, és a tömörítés elkészült.

A Huffman algoritmus igen elterjedt (3.2. táblázat), az általa létrehozott kódot "minimál kód"-nak is hívják. Ennek oka, hogy kategóriájában a legjobb eredményt nyújtja, hiszen ez az elméletileg legjobb kódolás ("best you can do"), ha minden eredeti adathoz egyedi kód tartozik.

A Huffman kódnak két fő változata létezik. Az egyiknél ("generic Huffman") általános kódtáblázatot használnak, ahol a táblázat egy általános, átlagosan jó táblázat, mely természetesen nem lehet optimális. A másik változatnál minden újabb adatcsoportra újabb kódtábla generálódik. Ennél a fajta kódolásnál jobb eredmény érhető el, hiszen mindig az adott adathalmazra optimális fa struktúra jön létre, viszont tárolni kell az új kódtáblázatot is, mely a tömörítési arányt rontja.

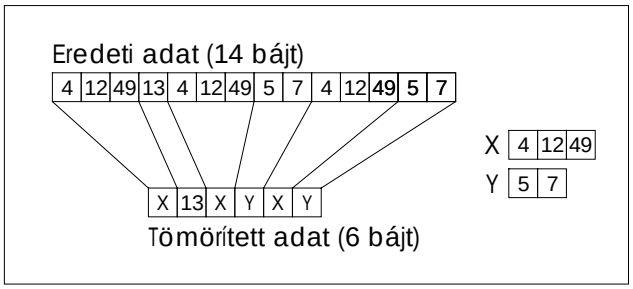
3.3 LZW kódolás[1]

Az LZW kódolás, mely szintén szerzőiről kapta nevét (Lempel-Ziv és Welch) jellemzője, hogy a többször ismételt adatoknak nem szükséges azonosnak lennie, az algoritmus képes az eltérő bájtokból álló szekvenciák tömörítésére is. Ráadásul a tömörítésnél használt kódtáblázat (CodeBook) külön tárolása sem szükséges, mert a tömörített adatfolyam azt tartalmazza. Ez úgy lehetséges, hogy betömörítéskor a szekvenciák első előfordulásakor nem a neki megfelelő kód, hanem maga a szekvencia és egy jelzés kerül a kimenetre. A kicsomagoló a szekvenciát és a jelzést olvasva automatikusan felépíti a saját kódtáblázatát. Külön előny, hogy az algoritmus teljesen szekvenciális adatfolyamra is alkalmazható, ami azt jelenti, hogy az algoritmus a működése folyamán csak a bemenetén érkező adatokat és a kódtábla információkat használja fel, így a bementi adatra a későbbiekben nem lesz már szüksége.

Formátum	*Elnevezés*
GIF	LZW compression
PostScript	LZWDecoder, LZWEncoder
TIFF	Sceme 5

3.3 táblázat

[1] A legtöbb ismertetett tömörítési algoritmustól eltérően, az LZW kódolás kereskedelmi használatra védettséget élvez. A liszensz természetesen megvásárolható a Unisys-től, további információt az [1] irodalomban találhatnak (US patent number 4,558,302 LZW compression in commercial application).



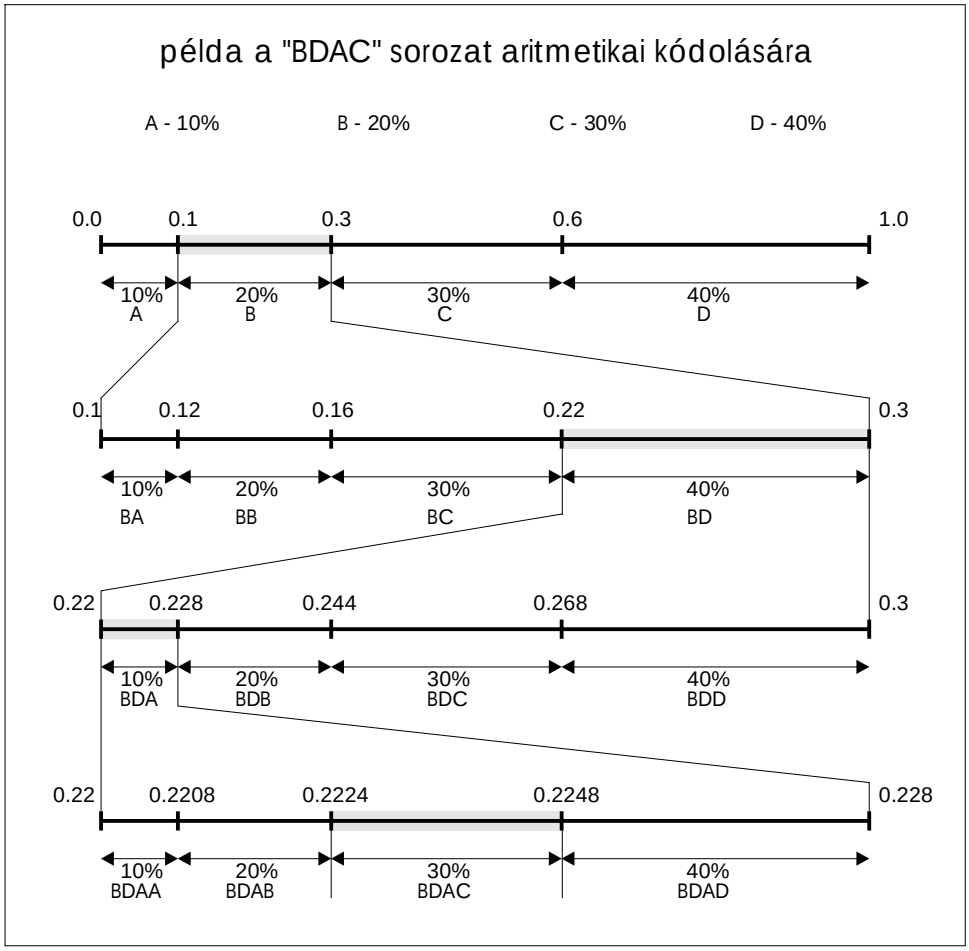
3.3. ábra

Tehát a kódtáblázat felépítése automatikus, és ez (előnyei mellett) számos problémát vet fel. A 3.3. ábrán látható egy bájtcsoport, mellyel a működés nyomon követhető. Látható, hogy ha a tömörítetlen adat sorrendisége szabja meg azt, hogy a kódtáblázat milyen módon jön létre, akkor egyes hosszabb adatcsoportokat (melyek kódolásával jobb tömörítési arányt lehetne elérni) nehéz észrevenni, mert a kódtáblázatban lévő rövidebb szekvenciák hamarabb kerülnek a kimenetre. Természetesen a legjobb tömörítési arány akkor érhető el, ha az összes rész-adatcsoportot vizsgálnánk, de ez nagyon számítás- és időigényes feladat lenne.

Az LZW tömörítés során a kódtáblázatban már megtalálható szekvenciák bővítése történik meg az új bájttal, így a hosszabb ismétlődő szekvenciák kódolása is megtörténik.

3.4 Aritmetikai kódolás[2]

A Huffman kódoláshoz képest 5-10%-kal jobb eredmény érhető el ezzel a módszerrel. Az alapelv az, hogy a bájt-sorozatot egyetlen számmal, egy valós számmal kell ábrázolni. A valós szám a 3.4. ábrán látható módon keletkezik. Először fel kell osztani a nulla és az egy közötti részt az adatelemek megoszlása szerint. Ezután a kódolandó szekvencia szerint az egyes rész-tartományokat tovább kell osztani, ugyancsak a statisztikai előfordulás szerint. Az adatelemek sorrendje tulajdonképpen mindegy. A lényeg az, hogy a résztartományok felosztása is mindig az eredeti sorrendben történjen. Így egy adott bájt-szekvencia egyetlen valós számmal valóban ábrázolható.



3.4. ábra

Azonban ezzel még nem értünk el jelentős eredményt, a valós számot is tárolni kell, ráadásul a szabványos fix- vagy lebegőpontos számábrázolások is több bájtot igényelnek. Ezért a valós számot nem a szokásos fix- vagy lebegőpontos módszerrel, és nem is decimális számjegyekkel, hanem binárisan kell tárolni. Például úgy, hogy a biteknek megfelelő tartományokat jelölünk ki, az egyes bithez a 0.5-től 1.0-ig terjedőt, a nullás bithez a 0.0-tól 0.5-ig terjedő résztartományt. A bájt-sorozaton egyre beljebb lépkedve mindig normálni kell az adott tartományt 0-tól 1-ig, így a bitek előbbi szereposztása szerint a szükséges szekvencia a megfelelő tartományok kijelölésével egyszerűen kódolható.

[2] Az aritmetikai kódolást az Egyesült Államokban 1990-ben szabadalmaztatták, így használata védettséget élvez (US patent number: 4,905,297 "Arithmetic Coding Encoder and Decoder System").

3.5 Különbségi kódolás

A tömörítés lényege, hogy az egymást követő, egymással fizikailag valamelyest összefüggő adatok különbségeinek tárolásával az újabb bájtok (a különbségek) kevesebb bitszámmal ábrázolhatók, mint az abszolút értékük. A digitalizált kép pont ilyen, az egymást követő minták a mintavétel sebességéhez képest viszonylag lassú jelváltozásai nagyszerűen tömöríthetők ezzel a módszerrel. Sajnos a kép bitmélységének növekedésével a tömörítés hatékonysága is jelentősen csökken, mert az egymást követő bájtok közötti különbség szintén növekszik.

Formátum	*Elnevezés*
JPEG	Predictive
LANDSAT	-
MPEG	-
CCITT H.261	-

3.5 táblázat

Egydimenziós, ill. többdimenziós változata is létezik a feldolgozandó adatok típusa szerint. A kép két- vagy háromdimenziós adathalmaz aszerint, hogy álló vagy mozgóképről van szó. Az állóképeknél a képpontok a közvetlen környezetüktől, a mozgóképek képpontjai viszont az előző kép ugyanazon képpontjaitól is függnek. Tulajdonképpen ezt használja ki ez a fajta kódolás, viszont a gyengéje is azonnal megmutatkozik, ha egy kontrasztosabb képet vizsgálunk. Az erős kontraszt éles, erőteljes sötét-világos átmeneteket jelent. Ez viszont nem kódolható optimálisan ezzel a módszerrel, hiszen a sötét-világos átmenet tárolásához szükséges bitszám akár a képpontok ábrázolásához szükséges bitszámot is elérheti. Sőt, ha igen nagy az átmenet, akkor még egy bittel több szükséges az előjel tárolásához, ami már nehezen nevezhető adattömörítésnek. Egyszerűsége miatt azonban kisebb kompromisszumokkal gyakran alkalmazzák.

4. Adatvesztéses tömörítések

Jelentős áttörést jelentettek az adatvesztéses tömörítések, hiszen ezekkel a módszerekkel jelentős adatcsökkenés érhető el: 50..100, vagy akár 1000 is lehet a tömörítési arány. Az emberi tökéletlenséget és tökéletességet használják ki egyidőben, hiszen az algoritmusokat az emberi érzékelés korlátai, ill. kreativitása szerint is optimalizálták. Jelenleg ez a terület még dinamikusan fejlődik, de már ipari körülmények között is számos helyen alkalmazzák, folyamatosan újabb és újabb felhasználási területeken bukkannak elő.

Természetesen a digitális videorendszerek a biztonságtechnikában sem léteznének nélkülük, az általuk biztosított nagyfokú adatcsökkenés teszi lehetővé a nagyfelbontású videojel digitális feldolgozását, tárolását, továbbítását. A következőkben két algoritmus működését vázoljuk, a részletesebb ismertetés sajnos meghaladja jelen cikk kereteit. További információk a témával kapcsolatos szakirodalomban találhatók.

4.1 Fraktál tömörítés[3]

A fraktál tömörítés lényege, hogy az eredeti képben kisebb és/vagy hasonló alakzatokat, képrészleteket lehet találni, és ezen redundancia megfelelő kódolásával igen nagy tömörítés érhető el. A tömörítés egy igen hosszadalmas iterációs algoritmus során megkeresi az ismétlődő vagy hasonló képrészleteket. A hasonlóság mellett egy olyan transzformációt is keres, mellyel az eredeti képrészlet a lehető legjobban megközelíthető. A tömörített adathalmaz csak kevés eredeti képinformációt tartalmaz, viszont a hivatkozásokra és a transzformációkra vonatkozó adatok adják a legnagyobb részét. Az eddigi legnagyobb, 1000:1-es tömörítés is elérhető vele, bár gyakorlati felhasználása jelenleg még nem jelentős, hiszen óriási számítási kapacitás szükséges a tömörítéshez. Nem szimmetrikus tömörítés abból a szempontból, hogy a kicsomagolás már sokkal gyorsabban elvégezhető.

[3] A tömörítési eljárást Michael Barnsley fejlesztette ki, cégével az Iterated Systems-szel. Michael Barnsley birtokolja a fraktál tömörítésre vonatkozó szabadalmat is (US patent number: 5,065,447 Fractal-based compression).

a) kicsomagolás 1 iterációval

b) kicsomagolás 2 iterációval

c) kicsomagolás 4 iterációval

d) a tömörítés előtti (eredeti) kép

4.1.1. ábra

A fraktál tömörített szekvencia kicsomagolása éppen ellenkezőleg történik, a kevés számú képi információból indulva, a hivatkozásokkal és a transzformációkkal "közelíti" a képet, több iteráción keresztül (4.1.1. ábra). Néhány iteráció után az eredmény: rendkívül jó minőségű kitömörített kép. Az eredmény annál is inkább figyelemreméltó, hogy az egyéb ismert adatvesztéses tömörítési algoritmusok a nagyobb mértékű tömörítéseknél már szemmel is látható romlást eredményeznek, amely a vizuális információ drasztikus romlását jelenti. Ezt a "digitális hatás"-t az emberi szem már nem szereti, zavaró számára. A fraktál transzformáción alapuló képtömörítés mentes ezektől a hátrányoktól, egy statisztikai felmérés alapján mind színárnyalat, mind felismerhetőség szempontjából jobb minőségű képet ad, mint egy hasonló tulajdonságokkal rendelkező JPEG tömörítésű kép (4.1.2. ábra).

a) Fraktál tömörítés (1:24, 2711 byte)

b) JPEG tömörítés (1:23, 2848 byte)

c) Fraktál tömörítés (1:11, 5963 byte)

d) JPEG tömörítés (1:11, 5723 byte)

e) Fraktál tömörítés (1:6, 11069 byte)

f) JPEG tömörítés (1:6, 11155 byte)

4.1.2. ábra

4.2 Diszkrét koszinusz transzformáción alapuló tömörítések (DCT)

Számos tömörített képformátum alapja a Diszkrét Koszinusz Transzformáció (DCT). A matematikai részletektől most eltekintve számos transzformációhoz hasonlóan a DCT transzformációnak is az a lényege, hogy a kép egy részletét (a DCT esetében a 8x8-as méret terjedt el) az őt létrehozó frekvencia komponensek segítségével adnak meg. Ezzel még nem lehetne jelentős tömörítést elérni, azonban a jelentéktelen frekvenciakomponensek elnyomásával (kvantálás) és célszerű sorrendbe állítással (zigzag) olyan adatsorozat hozható létre, amely már rendkívül jól tömöríthető az ismert adatvesztés nélküli tömörítési algoritmusok segítségével. A JPEG formátum például a DCT transzformációt követően RLE, majd Huffman vagy aritmetikai kódolást alkalmaz.

Formátum	*Elnevezés*
ACATS	-, (HDTV-hez)
JPEG	DCT
MPEG	DCT
CCITT H.261	DCT

4.2.1. táblázat

Transzformáció	*Rövidítés*
Diszkrét Fourier Transzformáció	DFT
Hadamard-Haar Transzformációk	HHT
Karhunen-Loeve Transzformációk	KLT
Slant-Haar Transzformációk	SHT
Walsh-Hadamard Transzformációk	WHT

4.2.2. táblázat

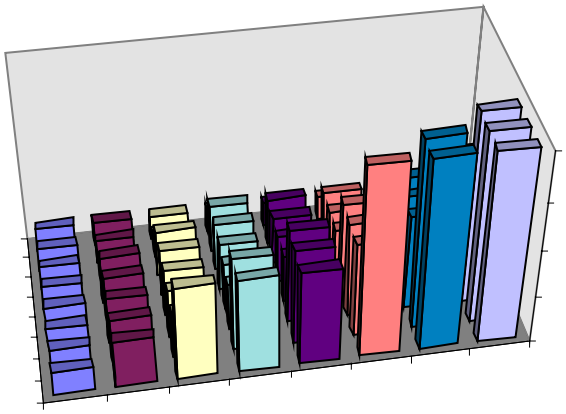
A DCT gyakori alkalmazásának többek között az az oka, hogy egyszerű és gyors algoritmusok léteznek a transzformáció megvalósítására, ráadásul jobb vizuális eredményt ad magas tömörítési arány mellett is az egyéb transzformációs eljárásokhoz képest. Néhány, a DCT-hez hasonló transzformációs eljárás a 4.2.2. táblázatban található (a [2.] irodalomban bővebb információ található ezekről a transzformációkról).

További előny, hogy nem csak tömörítésre használható, hanem a segítségével számos ügyes képfeldolgozási funkció, trükk is megvalósítható. Például remekül alkalmazható a kép paramétereinek javítására, módosítására (kép élesítése és homályosítása; fényerő, kontraszt beállítás; stb.), sőt néhány biztonságtechnikai digitális videorendszer ezeken a szolgáltatásokon túlmenően speciálisabb lehetőségeket is felkínál. Lehetőség van például egy telepített rendszer jól beállított kameráiról referencia képet készíteni, mely egy későbbi

időpontban a kiértékeléskor az elállítódott, rossz képet adó kamera képének javítására használható fel. Nem kis jelentőségű ez a funkció, ha belegondolunk, hogy a manapság hazánkban alkalmazott videorendszerek alkalmatlansága és a gondatlan kezelés, vagy éppen a karbantartás hiánya következtében rendre az éles helyzetekben derül ki, hogy a rögzített anyag csapnivaló.

4.2.1 JPEG, EJPEG formátumok

A DCT algoritmust alkalmazó számos alkalmazás közül talán a JPEG és az EJPEG formátumokat érdemes kiemelni [4.], [8.]. Ezeknél a formátumoknál a színes képek tárolása YUV szerint történik (bővebben előző számunkban). A DCT-t követően a kvantálás következik, mellyel a lényegi adatvesztés is megtörténik, ugyanis ezzel a művelettel a DCT értékek jelentős része megsemmisül (a kvantálás lényegében osztási műveletet jelent, csak a véges számábrázolás miatt a kitömörítéskor használt szorzás nem mindig az eredeti adatot fogja visszaadni). Természetesen a DCT értékeket (a 8x8 képpontos részletet leíró frekvencia komponenseket) súlyozva kell leosztani, így biztosítható a képrészlet vizuális hatásának megőrzése. Ennek megfelelően a kvantálást egy ún. kvantáló táblával valósítják meg, mely mind a 64 DCT értékre megadja a kvantálás mértékét. A kvantáló értékek elhelyezkedése illeszkedik a DCT után keletkező adatsorrendhez. A bal felső sarokban az egyenkomponens (DC), majd a táblázatban átlósan ide-oda haladva az egyre magasabb frekvenciák következnek (AC).



4.2.1. ábra

Egy tipikus kvantáló tábla látható a 4.2.1. ábrán, a oszlopok magassága az adatvesztéssel (a kvantálással) arányos. Az alacsony oszlopok az alacsony-frekvenciás összetevőket (bal felső sarok környéke) megőrzendő kis értéket, a magasak a nagyobb frekvenciás értéket (jobb alsó sarok felé) jobban elnyomó nagyobb értéket jelentenek. Ezt követően a "zigzag" kódolás következik, ami azt jelenti, hogy a DC-től az egyre magasabb frekvencia komponensek felé a kvantált adatokat sorbarendezik. Ha a példánkban alkalmazott kvantálótáblát használták, akkor a keletkező 64 érték első részében nagyobb, a vége felé közeledve egyre kisebb értékű, majd egyre több nulla adódik. Ez a csökkenő elrendezés kedvező az adatvesztés nélküli tömörítéseknek. A sok azonos bájtot kedvezően tömörítő RLE algoritmus után a JPEG / EJPEG formátum utolsó lépése, a Huffman kódolás vagy az aritmetikai kódolás következik. A keletkezett adatfolyam egy 5-30 tömörítési arányú, összecsomagolt képi információ.

Az EJPEG formátum a JPEG egyik továbbfejlesztése. Lehetővé teszi, hogy a DCT által kezelt 8x8-as képrészletekhez kellemesen kiválasztható, akár cellánként eltérő kvantálótábla tartozhat. Így a tömörítés paraméterei még szélesebb körben állíthatók be, mellyel biztosítható, hogy a kép eredő tömörítési aránya a lehető legkedvezőbb lehessen.

5. Irodalomjegyzék, hivatkozások

[1.] C. Wazne Brown and Barrz J. Shepherd: Graphics File Formats
Manning ISBN 1-884777-00-7
Prentice Hall ISBN 0-13-303405-4

[2.] Gonzalez, Rafael C., Digital Image Processing, Second Edition, Addison-Wesley, Reading, 1987.

[3.] Mark Nelson, Jean-Loup Gailly: The Data Compression Book
M&T Books ISBN 1-55851-434-1

[4.] JPEG standard;
Digital Compression and Coding of Continuous-Tone Still Images, ISO 10918 (ISO/IEC comittee JTC1/SC29/WG10)

[5.] MPEG standard;
Information Technology, Coding of moving pictures and associated audio, For digital storage media at up to about 1.5 MBit/s: ISO CD 11172

[6.] IEEE-P754 Floating Point Standard, Lebegőpontos nemzetközi szabvány

[7.] Magyar Képfeldolgozók és Alakfelismerők Országos konferenciája (KÉPAF), Keszthely 1997. október 9-11.
DCT és Fraktál tömörítő eljárások pszichovizuális összehasonlítása: "http://www.georgikon.pate.hu/visual.htm"

[8.] Magyar Képfeldolgozók és Alakfelismerők Országos konferenciája (KÉPAF), Keszthely 1997. október 9-11. Képfeldolgozás a biztonságtechnikában