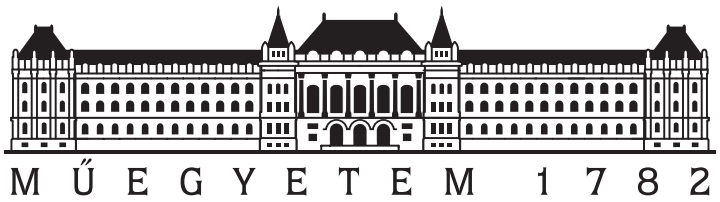


Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai kar
Híradástechnikai tanszék

Szinguláris értékek szerinti felbontáson alapuló képtömörítő eljárás

TDK dolgozat

Készítette: Wallner Ádám

Konzulens: Dr. Kovács Imre

Budapest, 2011

Kivonat

Napjainkban a technológia fejlődésével és egyre széleskörűbb alkalmazásával az amatőr, otthoni felhasználástól kezdve a profi stúdiótechnikáig bezárólag egyre jobb minőségű képek készítésére van lehetőség és igény, így a rögzítendő méret vagy a továbbítani kívánt adatsebesség is egyre nagyobb. Ezzel párhuzamosan a bitsebesség-csökkentő eljárások hatékonysága is növekszik, egyre jobb algoritmusokat fejlesztenek és építenek bele az újabb és újabb formátumokba (pl. JPEG 2000, MPEG4-AVC). A dolgozat témája egy olyan matematikai módszeren alapuló képtömörítő eljárás, amelyet eddig nem alkalmaztak széleskörűen e téren.

A szinguláris értékek szerinti felbontás előnye, hogy segítségével egy adott képblokknak egy optimális becslése állítható elő. A módszer hatékonysága természetesen nagyban függ az adott blokk méretétől és tartalmától, így az eljárás első lépése, a blokkokra bontás adaptív, azaz képtartalomfüggő. A kép részletgazdag, véletlenszerű részein kisebb, míg a homogén vagy jellegzetes struktúrájú területeken nagyobb blokkméret a hatékony.

A blokkokon elvégzett felbontás során kapott szinguláris vektorokat ezek után tovább kódolhatjuk: a következő lépésben egy szintén adaptívan választott transzformáció segítségével az adatsebesség (csekély minőségromlás mellett) nagymértékben tovább csökkenthető.

A dolgozat bemutatja az eljárás során használt matematikai módszereket, részletesen tárgyalja az eljárás teszteléséhez írt MATLAB programot és a hozzá tartozó felhasználói felületet, illetve kitér a megvalósított kódolási eljárás, valamint a már elterjedt és széleskörűen használt bitsebesség-csökkentő formátumok megfelelő kódolási lépéseivel való összevetésére is.

Abstract

Due to the technical development, we experience a growing tendency in the possibilities and demands of making better and better images nowadays - in consumer applications and in professional studios as well. Therefore, the amount of the recorded or transmitted data is also increasing. Simultaneously, the algorithms for bit rate reduction are getting more and more efficient as well, and are included in the new formats (e.g. JPEG 2000, MPEG4-AVC, etc.). This paper describes an image compression method, which is based on a technique that has not been applied widely in this area.

Using the singular value decomposition an optimal estimation of a given image-block can be made. The efficiency of the method highly depends on the size and content of this block. Therefore, the first step of the method is to split the frame into smaller blocks using an adaptive process. On the richly detailed or random-like parts of the picture the smaller blocks - while on the homogenous or characteristic structured areas the larger blocks are more effective.

The singular vectors obtained from the decomposition can be coded further. In the next step with a transformation (also adaptive chosen) the bit rate can be greatly reduced while little loss of quality.

The paper presents the mathematical tools used, describes the MATLAB program and it's user interface written for testing the method, and compares this method with the appropriate encoding of the current and widely used bit rate reduction formats.

Tartalomjegyzék

1. Bevezetés

Jelen TDK dolgozat célja egy olyan saját fejlesztésű, Matlab alapú veszteséges állókép-tömörítő eljárás bemutatása, mellyel – többek között a képen előforduló jellegzetes vízszintes és függőleges struktúrák megfelelő kódolásának köszönhetően – elfogadható minőségromlás mellett a kép mérete jelentősen csökken.

A dolgozatban először ismertetem a tömörítő eljárás során használt két legfontosabb matematikai eszközt: a szinguláris értékek szerinti felbontást illetve a transzformációs kódolást. Míg ez utóbbi módszer (illetve kétdimenziós változatának) alkalmazása széleskörűen elterjedt a bitsebességcsökkentő eljárásokban (például a JPEG és MPEG kódolókban), addig az előbbi eljárást a képtömörítésben – néhány demonstráción [1] illetve illusztráción [2, címlap] kívül – nem alkalmazzák.

Az azt következő fejezetben részletezem a tömörítés lépéseit. Ennek során bemutatom azokat a paramétereket, melyek segítségével a kimeneti képméret és képminőség szabályozható. A fejezet végén bemutatom az eljárás teszteléséhez készített grafikus felhasználói felületet is.

A tömörített képet leíró adatok mennyisége és fajtája a beállításoktól függően változik. Ahhoz, hogy ez az eljárás működőképes legyen, szükséges egy olyan fájlformátum, mellyel biztosítható, hogy a dekódoló helyesen értelmezi a letárolt adatokat, ugyanakkor minél kevesebb járulékos információt tartalmaz. A 4. fejezetben egy ilyan fájl-felépítést mutatok be.

Az utolsó fejezetben ismertetem a tömörítő eljárás tesztelésével kapott eredményeket, megvizsgálom, hogy a megadható paraméterek mely értékeinél lesz a tömörítés a leghatékonyabb, és bemutatom az eljárás továbbfejlesztésének lehetőségeit.

2. Matematikai alapok

E fejezetben bemutatom a tömörítő eljárásban alkalmazott matematikai módszereket: a szinguláris értékek szerinti felbontást és a transzformációs kódolást.

2.1. Szinguláris értékek

Egy négyzetes $\mathbf{A}$ mátrixnak λ egy sajátértéke és $\mathbf{v}$ egy sajátvektora, ha igaz a következő egyenlőség:

$$\mathbf{A} \cdot \mathbf{v} = \lambda \mathbf{v} \tag{1}$$

Ha a mátrix $n \times n$-es, akkor n sajátértéke van, de a (lineárisan) független sajátvektorok száma ennél kevesebb is lehet. Ha $m(\lambda_i)$-vel jelöljük egy λ_i sajátérték algebrai multiplicitását (azaz hogy hányszoros sajátérték), és $g(\lambda_i)$-vel a geometriai multiplicitását (azaz a hozzá tartozó sajátvektorok által kifeszített altér dimenzióját, vagyis hogy a sajátvektorok közül hány lineárisan független), akkor minden sajátértékre igaz a következő egyenlőtlenség:

$$g(\lambda_i) \leq m(\lambda_i), \tag{2}$$

és mivel $\sum m(\lambda_i) = n$, ezért $\sum g(\lambda_i) \leq n$. Ha viszont mindegyik sajátérték különböző, akkor az n db sajátvektor lineárisan független, így (1) alapján a következő egyenlet írható fel:

$$\mathbf{A} \cdot \mathbf{V} = \mathbf{V} \cdot \mathbf{\Lambda}, \tag{3}$$

ahol $\mathbf{V}$ a sajátvektorokból álló mátrix, $\mathbf{\Lambda}$ pedig a sajátértékekből álló diagonális mátrix. Az $\mathbf{A}$ mátrix sajátvektorok szerinti felbontása (4) ebből már egyszerűen kapható:

$$\mathbf{A} = \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V^{-1}} \tag{4}$$

Ennek a felbontásnak hátránya, hogy nem minden mátrixra létezik, a sajátvektorok nem feltétlenül ortogonálisak, és komplexek is lehetnek.

A szinguláris értékek szerinti felbontás (Singular Value Decomposition – SVD) [3] hasonlít a sajátvektorok szerinti felbontásra, ezzel szemben minden mátrixra (téglalap alakúakra) is értelmezett (5):

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \cdot \mathbf{\Sigma}_{m \times n} \cdot \mathbf{V}^T_{n \times n}. \tag{5}$$

Az $\mathbf{U}$ és $\mathbf{V}$ mátrixok oszlopai az ún. szinguláris vektorok, melyek ortogonálisak és tisztán valósak. A $\mathbf{\Sigma}$ mátrix pedig ($\mathbf{\Lambda}$-hoz hasonlóan) diagonális, a főátlójában a pozitív σ_i szinguláris értékek szerepelnek. Ezek száma megegyezik a mátrix rangjával (r), tehát ha $r < m$ és $r < n$, akkor $\mathbf{\Sigma}$ főátlójának alján 0-k is vannak (megegyezés szerint $\mathbf{\Sigma}$-ban σ_i-k csökkenő sorrendben szerepelnek).

A redukált SVD (5)-ből úgy kapható, hogy kihagyjuk a mátrixokból a felesleges részeket: $\mathbf{U}$-ból és $\mathbf{V}$-ből levágjuk az utolsó $(m - r)$ illetve $(n - r)$ oszlopot, $\mathbf{\Sigma}$-ból pedig csak a bal felső $r \times r$-es részt tartjuk meg. (Mivel $\mathbf{\Sigma}$-ban a főátlón kívül csak 0 elemek vannak, ezért $\mathbf{U}$ levágott oszlopai és $\mathbf{V}^T$ levágott sorai egymással és $\mathbf{\Sigma}$ főátlójának alsó 0 elemeivel szórzódtak volna össze). A redukált SVD tehát (6):

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times r} \cdot \mathbf{\Sigma}_{r \times r} \cdot \mathbf{V}^T_{r \times n}, \tag{6}$$

melyet a következő, diád-szorzatos alakban is írhatunk:

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \cdot \mathbf{u}_i \cdot \mathbf{v}_i^T \tag{7}$$

A csonkolt SVD (8) annyiban különbözik a redukált SVD-től (7), hogy csak az első (vagyis legnagyobb) t darab szorzatot adjuk össze:

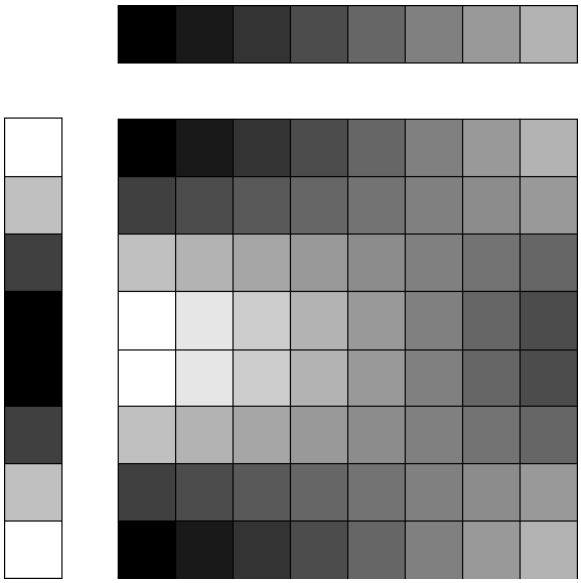
$$\begin{aligned} \widehat{\mathbf{A}}_t &= \mathbf{U}_{m\times t} \cdot \mathbf{\Sigma}_{t\times t} \cdot \mathbf{V}^T_{t\times n} \\ \widehat{\mathbf{A}}_t &= \sum_{i=1}^{t} \sigma_i \cdot \mathbf{u}_i \cdot \mathbf{v}_i^T \end{aligned} \tag{8}$$

Az így kapott $\widehat{\mathbf{A}}_\mathbf{t}$ mátrix rangja t, hiszen t darab diádmátrix összege. Ha $t \ll m, n$, akkor $\widehat{\mathbf{A}}_\mathbf{t}$ jóval kevesebb adattal írható le, mint $\mathbf{A}$, ugyanakkor az $\widehat{\mathbf{A}}_\mathbf{t}$ $\mathbf{A}$-nak egy ideális becslője. Az Eckart–Young tétel szerint ugyanis ha $\mathbf{B}$ egy legfeljebb t rangú $m \times n$-es mátrix, akkor az $||\mathbf{A} - \mathbf{B}||^2$ négyzetes hiba pontosan akkor minimális, ha $\mathbf{B} = \widehat{\mathbf{A}}_\mathbf{t}$.

Az 1. ábrán látható egy példa, hogy $t = 1$ esetén hogy nézhet ki az $\widehat{\mathbf{A}}_\mathbf{t}$ mátrix. A felette és mellette lévő vektorok nem az $\mathbf{u}$ és $\mathbf{v}$ vektorok (hiszen azok normája 1), hanem azoknak konstansszorosai. Alul ezen vektorok és mátrix ábrázolása látható egy szürkeárnyalatos képen, ahol a –1-nek fekete, míg az 1-nek fehér szín felel meg.

$$\begin{bmatrix} -1 & -0,8 & -0,6 & -0,4 & -0,2 & 0 & 0,2 & 0,4 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0,5 \\ -0,5 \\ -1 \\ -1 \\ -0,5 \\ 0,5 \\ 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -0.8 & -0.6 & -0.4 & -0.2 & 0 & 0.2 & 0,4 \\ -0.5 & -0.4 & -0.3 & -0.2 & -0.1 & 0 & 0.1 & 0,2 \\ 0.5 & 0.4 & 0.3 & 0.2 & 0.1 & 0 & -0.1 & -0,2 \\ 1 & 0.8 & 0.6 & 0.4 & 0.2 & 0 & -0.2 & -0,4 \\ 1 & 0.8 & 0.6 & 0.4 & 0.2 & 0 & -0.2 & -0,4 \\ 0.5 & 0.4 & 0.3 & 0.2 & 0.1 & 0 & -0.1 & -0,2 \\ -0.5 & -0.4 & -0.3 & -0.2 & -0.1 & 0 & 0.1 & 0,2 \\ -1.0 & -0.8 & -0.6 & -0.4 & -0.2 & 0 & 0.2 & 0,4 \end{bmatrix}$$



1. ábra. Egy diádmátrix és ábrázolása

2.2. Transzformációs kódolás

A transzformációs kódolásnál egy $\mathbf{x}$ vektor helyett egy lineáris transzfolmáltját, $\mathbf{y}$-t tároljuk. A lineáris transzformáció leírható egy $\mathbf{T}$ mátrixszal (9):

$$\begin{aligned} \mathbf{y} &= \mathbf{T} \cdot \mathbf{x} \\ \mathbf{x} &= \mathbf{T}^{-1} \cdot \mathbf{y} = \mathbf{S} \cdot \mathbf{y} \end{aligned} \tag{9}$$

Az inverz transzformációnál tulajdonképpen az $\mathbf{S}$ mátrix oszlopait $\mathbf{y}$ megfelelő elemeivel szorozzuk és utána összeadjuk, vagyis ezek az oszlopok $\mathbf{x}$ bázisvektorai. Mivel $\mathbf{x}$ az $\mathbf{S}$ oszlopainak lineáris kombinációja, ezért ez utóbbi mátrixot a továbbiakban szintézis-, míg a $\mathbf{T}$-t transzformációs mátrix néven említem. A transzformációs kódolás lényege, hogy olyan bázisvektorokat alkalmazunk, melyek a lehető legjobban közelítik a lehetséges $\mathbf{x}$ vektorokat. Ekkor ugyanis kevés ilyen bázisvektor felhasználásával becsülhető $\mathbf{x}$, vagyis (a szintézismátrix ismeretében) $\mathbf{y}$-nak csak kevés értékes jegyét kell tárolnunk.

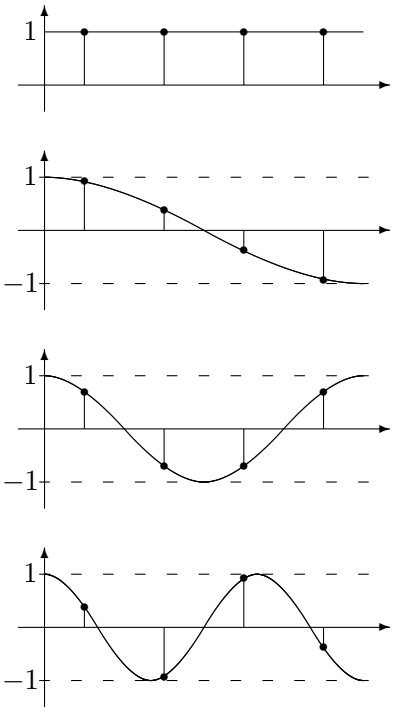
Ahhoz, hogy a transzformáció és az inverze is elvégezhető legyen, szükséges és elégséges feltétel, hogy $\mathbf{S}$ oszlopai lineárisan függetlenek legyenek (ekkor létezik $\mathbf{T}$). A különböző jelfeldolgozási algoritmusokban használt transzformációk a leggyakrabban ortogonálisak (ebben az esetben $\mathbf{S} = \mathbf{T}^T$), de ez nem feltétele a transzformáció elvégezhetőségének.

Mivel a transzformálandó $\mathbf{u}$ és $\mathbf{v}$ vektorok (a képtartalomtól függően) sokfélék lehetnek, ezért nehéz olyan transzformációt találni, mellyel minden esetben hatékonyan lehetne a vektorokat kódolni. A tömörítő eljárás tervezése során ezért nem volt cél egy optimális transzformáció megtalálása. Kétféle transzformációt választhatunk ki, ezek bázisvektoraival kellően jól le lehet írni az $\mathbf{u}$ és $\mathbf{v}$ vektorokat. Az egyik a JPEG és MPEG tömörítésben is használt diszkrét koszinusz transzformáció, a másik pedig jellegében hasonlít ehhez, de saját ötleten alapul. Ezek tetszőleges méretű vektorokra alkalmazhatók, itt most az eljárás során kiválasztható legkisebb, 4×4-es transzformációs mátrixokat mutatom be.

2.2.1. Diszkrét koszinusz transzformáció

A diszkrét koszinusz transzformáció (Discrete Cosine Transform – DCT) bázisvektorai egy-egy koszinuszfüggvény mintavételezett értékei (lásd 2. ábra), ezek frekvenciája pedig olyan, hogy a mátrix mérete a félhullámhossznak mindig egész számú többszöröse (kivéve természetesen a 0 frekvenciájú első vektort). Egy 4×4-es DCT szintézis- és transzformációs mátrixa például:

$$\begin{aligned} \mathbf{S}_{DCT,4} &= \cos \begin{bmatrix} 0 & \frac{\pi}{8} & \frac{2\pi}{8} & \frac{3\pi}{8} \\ 0 & \frac{3\pi}{8} & \frac{6\pi}{8} & \frac{9\pi}{8} \\ 0 & \frac{5\pi}{8} & \frac{10\pi}{8} & \frac{15\pi}{8} \\ 0 & \frac{7\pi}{8} & \frac{14\pi}{8} & \frac{21\pi}{8} \end{bmatrix} \cong \begin{bmatrix} 1 & 0{,}924 & 0{,}707 & 0{,}383 \\ 1 & 0{,}383 & -0{,}707 & -0{,}924 \\ 1 & -0{,}383 & -0{,}707 & 0{,}924 \\ 1 & -0{,}924 & 0{,}707 & -0{,}383 \end{bmatrix} \\ \mathbf{T}_{DCT,4} &= \operatorname{diag}\left[\frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right] \cdot \mathbf{S}_{\mathrm{DCT},4}^{\mathrm{T}} \cong \begin{bmatrix} 0{,}25 & 0{,}25 & 0{,}25 & 0{,}25 \\ 0{,}462 & 0{,}191 & -0{,}191 & -0{,}462 \\ 0{,}354 & -0{,}354 & -0{,}354 & 0{,}354 \\ 0{,}191 & -0{,}462 & 0{,}462 & -0{,}191 \end{bmatrix}. \end{aligned} \tag{10}$$



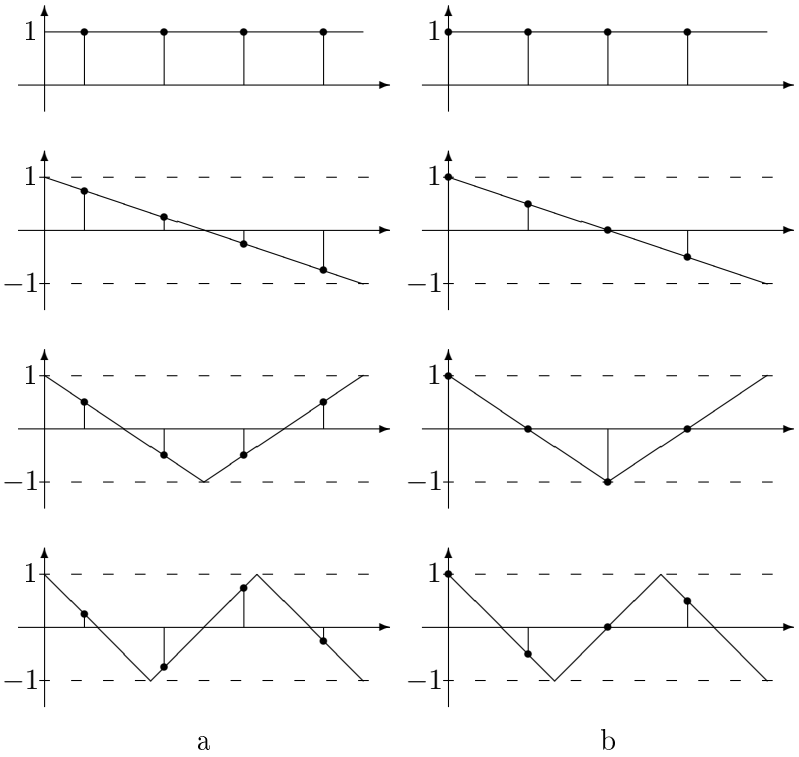
2. ábra. A 4×4-es DCT transzformáció együtthatói

2.2.2. Háromszög-transzformáció

E transzformáció hasonlít a diszkrét koszinusz transzformációhoz, azonban a bázisvektorok nem egy-egy koszinuszfüggvény, hanem egy azzal megegyező frekvenciájú háromszögjel mintavételezett értékei. Egy képen szereplő egyenletes változást ezzel a módszerrel hatékonyabban (kevesebb együtthatóval) tudunk leírni, mint DCT-vel. További előnye, hogy a transzformációs és a szintézismátrix is csak racionális elemeket tartalmaz.

Az említett háromszögfüggvény mintavételezési helyeinek kiválasztására két lehetőségünk van. Vagy szimmetrikusan mintavételezzük, ekkor a bázisvektor első és utolsó elemeinek abszolútértéke megegyezik (lásd 3.a ábra), vagy pedig úgy, hogy a vektorok első elemeinek értéke mindig 1 (lásd 3.b ábra).

Szimmetrikus mintavételezés esetén az bázisvektorok elemeinek összege 0 (az elsőt kivéve), így a transzformációs mátrix első sora egyforma, $1/n$ értékű elemekből áll (így lesz a $\mathbf{T} \cdot \mathbf{S} = \mathbf{I}$ mátrix első sora $[1, 0, 0, \ldots 0]$), a transzformált vektor első eleme az eredeti vektor átlagával (vagyis DC komponensével) egyezik meg. A másik esetben ez nincs így, azonban a mátrixok elemei között előfordulnak 0-k, így a transzformáció számítása gyorsabb.



3. ábra. A háromszögfüggvények mintavételezési lehetőségei 4×4-es esetben

4×4-es esetben a mátrixok az alábbiak:

$$\mathbf{S}_{H1,4} = \begin{bmatrix} 1 & 0{,}75 & 0{,}5 & 0{,}25 \\ 1 & 0{,}25 & -0{,}5 & -0{,}75 \\ 1 & -0{,}25 & -0{,}5 & 0{,}75 \\ 1 & -0{,}75 & 0{,}5 & -0{,}25 \end{bmatrix}$$

$$\mathbf{T}_{H1,4} = \begin{bmatrix} 0{,}25 & 0{,}25 & 0{,}25 & 0{,}25 \\ 0{,}6 & 0{,}2 & -0{,}2 & -0{,}6 \\ 0{,}5 & -0{,}5 & -0{,}5 & 0{,}5 \\ 0{,}2 & -0{,}6 & 0{,}6 & -0{,}2 \end{bmatrix}. \tag{11}$$

$$\mathbf{S}_{H2,4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0,5 & 0 & -0,5 \\ 1 & 0 & -1 & 0 \\ 1 & -0,5 & 0 & 0,5 \end{bmatrix}$$

$$\mathbf{T}_{H2,4} = \begin{bmatrix} 0 & 0,5 & 0 & 0,5 \\ 0,5 & 0 & 0,5 & -1 \\ 0 & 0,5 & -1 & 0,5 \\ 0,5 & -1 & 0,5 & 0 \end{bmatrix}. \tag{12}$$

Az első esetben a 4×4-es szintézismátrix ortogonális, emiatt az oszlopainak konstans-szorosai lesznek a transzformációs mátrix sorai. A többi esetben ez azonban nem igaz, $\mathbf{T}$ csak inverz-számítással kapható meg $\mathbf{S}$-ből.

3. A tömörítő eljárás lépései

Ebben a fejezetben részletesen bemutatom a megvalósított tömörítő eljárás lépéseit. Maga a tömörítés a már említett szinguláris értékek szerinti felbontással illetve a transzformációs kódolással történik, de ezek hatékony működéséhez természetesen további lépések szükségesek. Az eljárás menete a következő:

- Színtér-transzformáció
- Padding
- Blokkokra bontás
- A színcsatornák blokkjainak alulmintavételezése
- SVD számítás és a kódolandó vektorok kiválasztása
- Transzformáció és a kódolandó együtthatók kiválasztása
- Kvantálás
- Inverz transzformáció, majd a blokkok és a teljes kép visszaszámolása
- A tömörítés objektív jellemzőinek meghatározása

3.1. Színtér-transzformáció

Színes képeknél minden pixelt három érték ír le, ezek leggyakrabban a vörös, zöld és kék csatorna értékei. Az emberi szem felbontóképessége a színes képelemekre kb. 3-8-szor rosszabb, mint a feketefehér képelemekre [4]. Emiatt hatékonyabb, ha ezen három adat helyett minden pixelt egy világosság- és két színinformációval (színkülönbségi jellel) kódolunk. A nagyfelbontású videótartalmakra illetve a számítógépes alkalmazásokra az ITU-R (International Telecommunication Union, Radiocommunication Sector) megfogalmazta a BT.709-es ajánlást [5], melynek színtér-transzformációja az alábbi:

$$\begin{aligned} Y &= 0,2126 \cdot R + 0,7152 \cdot G + 0,0722 \cdot B \\ P_R &= (R - Y) \cdot \frac{0,5}{0,7874} \\ P_G &= (G - Y) \cdot \frac{0,5}{0,2848} \\ P_B &= (B - Y) \cdot \frac{0,5}{0,9278}. \end{aligned} \tag{13}$$

A világosságjel dinamikatartománya 0..1, míg a három színkülönbségi jelé $\pm 0,5$. E három jel azonban nem független (hiszen $0.3348 \cdot P_R + 0.4074 \cdot P_G + 0.134 \cdot P_B = 0,2126 \cdot (R - Y) + 0,7152 \cdot (G - Y) + 0,0722 \cdot (B - Y) = 0$), vagyis közülük tetszőlegesen választott kettő, valamint a világosságjel egyértelműen meghatározza a színt.

Az analóg színes TV-technikában a továbbítandó jelnek (jel-zaj-viszony megfontolások miatt) a két legnagyobb értéktartományút, azaz a vörös és kék színkülönbségi jeleket[1] választották, és

[1] Mivel ott másak voltak az alapszínek, ezért a világosságjel számítása nem a (13) egyenletben szereplő együtthatókkal történt, valamint a színkülönbségi jeleket is máshogy skálázták.

ezeket a világosságjelhez képest kisebb (az amerikai NTSC rendszerben 1/3–1/8, míg az európai PAL rendszerben 1/5-szörös) sávszélességgel továbbították. A digitális kép- és videókódolásban is hasonló a helyzet: ott a leggyakrabban elterjedt formátumokban négy pixelre négy világosság-, de csak egy-egy vörös illetve kék színinformáció jut.

A tömörítő eljárás ezek miatt az RGB értékek helyett szintén a (13) alapján számolt Y-t, P_R-t és P_B-t használja, és a színinformációkat kevesebb adattal kódolja, mint a világosságét. Fekete-fehér kép esetén természetesen csak az Y csatorna hordoz információt, és $P_R = P_B = 0$ (mivel $R = G = B = Y$).

3.2. Padding

A tömörítendő kép mérete bármekkora lehet, de a következő lépést, a blokkokra bontást egyszerűbb úgy elvégezni, ha a kép méretei a nagyblokk méretének egész számú többszörösei. Ehhez a képet ebben a lépésben jobboldalt és alul kiegészítjük csupa 0 oszlopokkal és sorokkal.

3.3. Blokkokra bontás

Ha az egész képet egyben kódolnánk le, az akkor lenne hatékony, ha az egész képnek lenne egy egységes, jellegzetes struktúrája. Kisebb méretű blokkok esetén viszont szinte mindegyik blokknak van valamilyen jellegzetes mintázata, melyet hatékonyan lehet a szinguláris értékek szerinti felbontással tömöríteni. Azonban az, hogy mekkora blokkméret az ideális, nagyban függ a kép tartalmától. Nagyméretű egyszínű vagy viszonylag egyenletesen változó felületek esetén (például ég) egy nagyobb blokk is nagyon jól becsülhető mindössze néhány adattal. Ugyanakkor részletgazdag területeken (például fű) kisebb blokkméretet érdemes választani.

Mivel egy átlagos képen gyakran szerepelnek egyszínű és részletgazdag területek is, nehéz olyan blokkméretet találni, mely az egész képre optimális. A blokkokra bontás emiatt egy adaptív módszerrel is végezhetjük. Meg kell adnunk a maximális és a minimális blokkméretet, majd először felbontjuk a képet a maximális méretű blokkokra (a továbbiakban makroblokk[2]). Ezt követően ezeket a makroblokkokat egy rekurzív algoritmus darabolja tovább.

Ezen algoritmus bemeneti adata az aktuális blokk három csatornája (3 darab $b \times b$ méretű mátrix), valamint két paraméter, mely az eddigi blokkok méreteit és pozícióit azonosítja. A méret egy természetes szám: 0, ha az adott blokk megegyezik a makroblokkal, 1, ha feleakkora, stb. A pozíció pedig az adott blokk bal felső elemének koordinátája a makroblokkon belül. Az algoritmus minden blokkra eldönti, hogy felosztja-e négy további kisebb (bal felső, jobb felső, bal alsó és jobb alsó) blokkra. Ha igen, akkor mindegyikre meghívja ugyanezt az algoritmust, ha pedig nem, akkor a már említett két bemeneti paraméterhez hozzáfűzi ennek a blokknak a méretét és koordinátáját.

Ha az adott blokkméret megegyezik a minimális blokkmérettel, akkor ezt a blokkot az algoritmus értelemszerűen nem darabolja már tovább. Egyéb esetben a következő feltételek alapján dönt:

- Ha a bal alsó vagy a jobb felső blokk tiszta fekete, akkor valószínűleg a csupa 0-val kiegészített kép alján vagy jobb oldalán vagyunk. Ilyenkor érdemes a blokkot felosztani, mert a hasznos képtartalmat így rövidebb vektorokkal tudjuk leírni, a csupa 0-t tartalmazó blokkokat pedig 0 hasznos biten tároljuk.

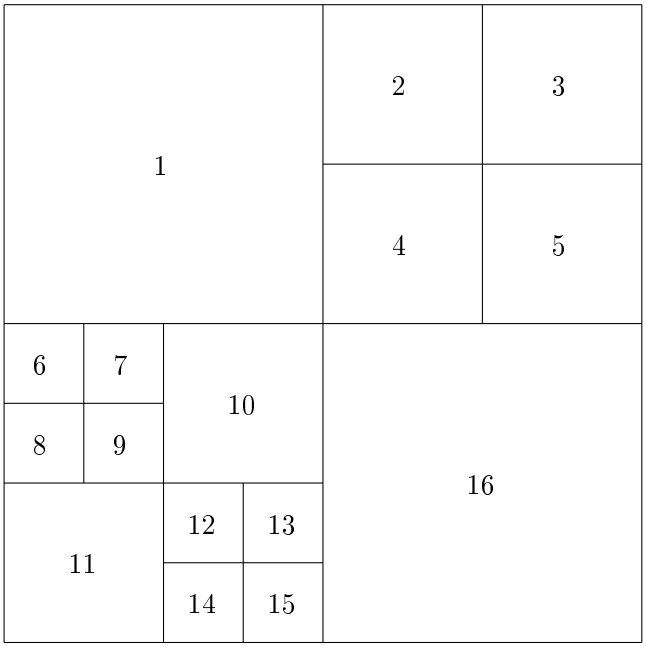
[2]Ezen makroblokkoknak természetesen semmi közük az MPEG kódolásban a mozgásbecslés alapegységéhez, melyeket szintén makroblokknak neveznek

- Ha a leendő négy kisebb blokkban a pixelek szórásának átlaga jelentősen eltér egymástól, az azt jelenti, hogy ezekben a blokkokban a képtartalom is eltérő lesz, így érdemes feldarabolni a blokkot.

- Jelöljük $\mathbf{y}_u$-val és $\mathbf{y}_v^T$-vel egy blokk világosságjelének ($\mathbf{Y}$) sor- illetve oszlopösszegét. E mátrix nemnegatív, így hatékonyan becsülhető a sor- és oszlopösszegek szorzatával kapott diádmátrix konstans-szorosával ($\widehat{\mathbf{Y}} = k \cdot \mathbf{y}_u \cdot \mathbf{y}_v$), mely nemnegatív mátrixokra nagyon jól közelíti az első szinguláris érték által meghatározott becslést ($\widehat{\mathbf{Y}}_1 = \sigma_1 \cdot \mathbf{u}_1 \cdot \mathbf{v}_1^T$, lásd (8), 9. oldal), ugyanakkor egyszerűbb kiszámolni.

  Ha tehát az $\mathbf{Y} - \widehat{\mathbf{Y}}$ mátrix átlagos négyzetes hibája egy előre megadott korlát alattt van, akkor ezt a blokkot kevés szinguláris értékkel tudjuk leírni, míg ha nem, akkor érdemes a blokkot tovább osztani.

Az adaptív felbontásra egy példa a 4. ábrán látható.



sorszám	méret	koordináta
1	1	(1, 1)
2	2	(1,17)
3	2	(1,25)
4	2	(9,17)
5	2	(9,25)
6	3	(17, 1)
7	3	(17, 5)
8	3	(21, 1)
9	3	(21, 5)
10	2	(17, 9)
11	2	(25, 1)
12	3	(25, 9)
13	3	(25,13)
14	3	(29, 9)
15	3	(29,13)
16	1	(17,17)

4. ábra. Példa az adaptív blokkokra bontásra egy 32×32-es makroblokk esetén

A változó méretű blokkokra bontást az állókép-tömörítő eljárások nem alkalmazzák, a H.264/AVC kódolás azonban igen [6], [7], ahol a mozgásbecslés során kapott különbségi képet bontják fel ilyen módon.

A fent ismertetett adaptív felbontást természetesen lehetne javítani további feltételek bevezetésével, esetleg egy tanulóalgoritmus segítségével. Ennek vizsgálata azonban meghaladja e dolgozat kereteit.

3.4. A színcsatornák blokkjainak alulmintavételezése

Ebben a lépésben kihasználjuk az emberi látás már említett (lásd 3.1. fejezet) tulajdonságát, hogy a színes képelemeket kevésbé tudjuk megkülönböztetni. Miután elvégeztük a blokkokra bontást, a

színcsatornák blokkjait alulmintavételezzük: minden 2×2-es rész helyett ezen négy érték összegének a felét (azaz átlaguk kétszeresét) tároljuk le, így a mátrixok mérete negyedakkora lesz, és első néhány szinguláris értékük csak kis mértékben változnak.

Ha ugyanis a négy érték átlagát tárolnánk le, akkor a szinguláris értékek feleakkorára változnának. Tekintsünk egy $\mathbf{A} = \mathbf{u} \cdot \mathbf{v}^T$ diád-mátrixot és legyen $\mathbf{A}'$ feleakkora, mint $\mathbf{A}$, és minden eleme $\mathbf{A}$ megfelelő 2×2-es részében lévő elemeinek átlaga. Hasonlóan $\mathbf{u}'$ és $\mathbf{v}'$ feleakkora vektorok, mint $\mathbf{u}$ és $\mathbf{v}$, elemeik $\mathbf{u}$ és $\mathbf{v}$ két egymás melletti elemének átlagai. Ekkor igaz az $\mathbf{A}' = \mathbf{u}' \cdot \mathbf{v}'^T$ egyenlőség is. Ugyanis $\mathbf{A}'$-nak egy tetszőleges (például az $a'_{1,1}$) elemére:

$$\begin{aligned} a'_{1,1} &= \frac{a'_{1,1} + a'_{1,2} + a'_{2,1} + a'_{2,2}}{4} = \frac{u_1v_1 + u_2v_1 + u_1v_2 + u_2v_2}{4} = \frac{(u_1 + u_2)(v_1 + v_2)}{4} = \\ &= \frac{u_1 + u_2}{2} \cdot \frac{v_1 + v_2}{2} = u'_1 \cdot v'_1. \end{aligned} \tag{14}$$

Ha azonban $\mathbf{u}$ és $\mathbf{v}$ ortogonális volt, akkor $\mathbf{u}'$ és $\mathbf{v}'$ elemeinek négyzetösszege nagyjából 0,5 lesz (mivel feleannyi darab, de átlagosan ugyanakkora elemeik vannaik, mint $\mathbf{u}$-nak és $\mathbf{v}$-nek). Ha feltesszük, hogy a négyzetösszeg pontosan 0,5, akkor $\sqrt{2}$-vel kell $\mathbf{u}'$-t és $\mathbf{v}'$-t megszorozni, hogy ortogonálisak legyenek. Ha felírjuk a szinguláris értékek szerinti felbontást $\mathbf{A}$-ra és $\mathbf{A}'$-re, akkor látszik, hogy ez utóbbinak a szinguláris értéke feleakkora:

$$\begin{aligned} \mathbf{A} &= 1 \cdot \mathbf{u} \cdot \mathbf{v} \\ \mathbf{A}' &= \frac{1}{2} \cdot \sqrt{2}\mathbf{u}' \cdot \sqrt{2}\mathbf{v}'^T. \end{aligned} \tag{15}$$

A (15) egyenlet természetesen bármelyik szinguláris érték–szinguláris vektor csoportra felírható, vagyis ahhoz, hogy egy kétszeresen alulmintavételezett mátrix szinguláris értékei az eredetihez képest ne nagyon változzanak meg, az kell, hogy az átlagszámítás során ne 4-gyel, hanem csak 2-vel osszunk.

Az alulmintavételezés során a jelet egy 2×2-es méretű téglalap-ablakfüggvénnyel súlyoztuk. Ez egy olyan aluláteresztő, melynek frekvenciatartománybeli képe $\frac{\sin x}{x}$ jellegű, vagyis nem ideális, a magasabb frekvenciás komponensek átlapolódhatnak az alulmintavételezés során. Ezt úgy lehetne elkerülni, hogy az átlagszámításnál nem csak a már említett 4 pixelt, hanem távolabbiakat is figyelembe veszünk. Tapasztalataim szerint azonban önmagában ez az egyszerű átlagolás szinte semmilyen észrevehető minőségromlást sem okoz, így a tömörítőeljárásban is ezt alkalmaztam.

3.5. SVD számítás és a kódolandó vektorok kiválasztása

A szinguláris értékek szerinti felbontást a Matlab `svd` utasítása végzi el, a felbontás eredménye a $(\sigma_i,\ \mathbf{u}_i,\ \mathbf{v}_i)$ hármasok, blokkonként $3b$ (fekete-fehér képnél b) darab.

A tömörítés lényege, hogy ezen b darab érték helyett csak t darabot tárolunk le. Annak a meghatározása, hogy egy adott blokkban hány ilyen érték legyen, háromféle eljárással történhet:

- A legegyszerűbb, ha t értéke minden blokkra egy előre megadott állandó. Ennek a kiválasztásnak az az előnye, hogy a képméret könnyen szabályozható, viszont ha a blokkokra bontás nem adaptív, akkor egy kép különböző részein lévő azonos méretű blokkokra ez a fix érték néha túl sok, néha túl kevés. Ha viszont a blokkokra bontás adaptív, akkor ez a hátrány nem jelentkezik. Színes képeknél a különböző csatornák szinguláris értékeit összegyűjtve és csökkenő sorrendbe rendezve választjuk ki az első t darabot.

- Abszolút korlát: egy értéket akkor és csak akkor veszünk be, ha ennél a korlátnál nagyobb. Előnye, hogy a képen mindenhol nagyjából egyenletes lesz a torzítás (a szinguláris érték nagysága arányos azzal, amekkora mértékben a hozzá tartozó ($\sigma \cdot \mathbf{u} \cdot \mathbf{v}$) diád-szorzat a kapott képet módosítja), hátránya viszont, hogy a képméret nehezebben szabályozható. Az elnevezés abból a szempontból nem teljesen pontos, hogy változó blokkméret esetén ezen abszolút korlát is változik (lásd (15)), értéke minden blokkra $a \cdot b$, ahol a az előre megadott korlát, b pedig a blokkméret.

- Relatív korlát: hasonló az előzőhöz, csak itt a korlát minden blokkra egyedi: a maximális érték adott hányada. Akkor érdemes használni, hogyha egy képen nagyon világos és nagyon sötét részek is vannak (abszolút korlát alkalmazásával az előbbi helyekre túl sok, utóbbiakra túl kevés érték jutna).

Ezen lépés végeredménye az $\mathbf{U}_{b\times t}$, $\mathbf{\Sigma}_{t\times t}$ és $\mathbf{V}_{b\times t}$ mátrixok (a továbbiakban $\mathbf{U}$, $\mathbf{\Sigma}$ és $\mathbf{V}$), ahol t értéke nem feltétlenül állandó az egyes blokkoknál és csatornáknál.

3.6. Transzformáció és a kódolandó együtthatók kiválasztása

Amennyiben van transzformáció, úgy azt (a (9) egyenlethez hasonlóan) a következőképpen számoljuk:

$$\begin{aligned} \mathbf{U}_{tr} &= \mathbf{T} \cdot \mathbf{U} \\ \mathbf{V}_{tr} &= \mathbf{T} \cdot \mathbf{V}. \end{aligned} \tag{16}$$

A kódolandó együtthatók kiválasztásánál kihasználhatjuk azt, hogy az emberi szem kevésbé érzékeny a magasabb frekvenciás képtartalomra. A JPEG tömörítés során ezt úgy veszik figyelembe, hogy a transzformáció (mely ott egy 2 dimenziós DCT) után a kapott együtthatókat egy együtthatómátrix megfelelő elemeivel osztják el, és utána kvantálják. Ezen együtthatómátrixok gyakran használt értékei például [8], [4]:

$$Q_1 = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$Q_2 = \begin{bmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 37 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix} \tag{17}$$

E kvantálómátrixokat az emberi látórendszer vizsgálata és szubjektív tapasztalatok alapján határozták meg. Mindkét mátrixra igaz, hogy a legkisebb és legnagyobb értékek közti arány kb. tízszeres, és az együtthatók alapvetően balról jobbra illetve fentről lefele növekednek (a bal felső a DC komponenshez, míg a jobb alsó a legmagasabb vízszintes és függőleges frekvenciájú komponenshez tartozik).

Hasonló módszert alkalmaztam ennél a tömörítő eljárásnál: a transzformációval kapott együttható-vektorokat ($\mathbf{U}_{tr}$ és $\mathbf{V}_{tr}$ b méretű oszlopait) először egy $\mathbf{w}$ vektorral osztottam el. E vektor elemeinek meghatározását a (17)-beli mátrixok alapján végeztem. Ha csak az első transzformációs együtthatót vesszük figyelembe $\mathbf{u}_{tr}$-ből és $\mathbf{v}_{tr}$-ből, akkor a visszatranszformált $\mathbf{u}$ és $\mathbf{v}$ vektorok konstansok így a diádszorzatuk is konstans lesz, mint a bal felső DCT együtthatónak. Ha pedig csak az utolsó transzformációs együtthatót tároljuk, akkor az $\mathbf{u}$ és $\mathbf{v}$ vektorok diádszorzata a jobb alsó DCT-együtthatónak felel meg. Így ha a $\mathbf{w}$ vektor első eleme 1, az utolsó $\sqrt{b}$, a közbülsőket pedig lineárisan interpoláljuk, akkor az ezzel való szorzás nagyjából megfelel a DCT-nél alkalmazott kvantálómátrixnak (a legnagyobb frekvenciájú $\mathbf{u}$ és $\mathbf{v}$ vektorok szorzatának az együtthatója $\sqrt{b}\cdot\sqrt{b}=b$-szerese a DC együtthatónak). A $\mathbf{w}$ vektor nagyságrendjének meghatározásához írjuk fel a számtani és a négyzetes közép közti egyenlőtlenséget egy b méretű szinguláris vektor ($\mathbf{u}$) elemeire:

$$\begin{aligned}\frac{u_1+u_2+\dots u_b}{b} &\leq \sqrt{\frac{u_1^2+u_2^2+\dots u_b^2}{b}}\\ \sqrt{b}\cdot\frac{u_1+u_2+\dots u_b}{b} &\leq \sqrt{u_1^2+u_2^2+\dots u_b^2}\\ \sqrt{b}\cdot ut_1 &\leq 1\end{aligned} \tag{18}$$

A fenti egyenlőtlenségben kihasználtuk, hogy az első transzformációs együttható (ut_1) megegyezik az eredeti vektor elemeinek átlagával, illetve hogy az $\mathbf{u}$ vektor normált. A $\mathbf{w}$ vektor első elemét tehát $\frac{1}{\sqrt{b}}$-re érdemes választani, így a súlyozás után sem fogunk 1-nél nagyobb értéket kapni. A vektor utolsó eleme ezek alapján $\frac{1}{\sqrt{b}}\cdot\sqrt{b}=1$, a többi érték pedig ezek között lineárisan helyezkedik el.

A súlyozás után az együtthatók nagyjából csökkenő sorrendben lesznek. Közülük az első néhányat választjuk ki úgy, hogy a nem kiválasztottak között már ne legyen egy előre megadott korlátnál nagyobb abszolútértékű. Így adott esetben feleslegesen tárolunk le kis értékű együtthatókat, viszont elég a tárolt együtthatók darabszámát tudnunk, nem kell megmondani, hogy egy adott tárolt együttható pontosan melyik bázisvektorhoz tartozik.

3.7. Kvantálás

A szinguláris értékek szerinti felbontás illetve a transzformáció során kiszámolt $\mathbf{\Sigma}$, $\mathbf{U}'_{tr}$ és $\mathbf{V}'_{tr}$ (vagy $\mathbf{\Sigma}$, $\mathbf{U}$ és $\mathbf{V}$) mátrixok elemeit a Matlab double típusú, azaz 8 byte méretű számokként tárolja. Az eredeti kép minden pixelét csatornánként 8 biten írtuk le, célszerűnek látszik ezeket a mátrixokat is 8 bitre kvantálni. A programban egy-egy vektorként megadható, hogy a szinguláris értékeket illetve a vektorok elemeit hány biten tároljuk. Az 5.4 fejezetben ismertetett eredmények alapján e két vektor alapértelmezésben $\mathbf{q}_\Sigma=\mathbf{q}_{U,V}=[9,\ 7,\ 7]$ (vagyis 9 bit jut minden Y, és 7-7 bit minden P_R és P_B csatornát leíró értékhez).

A kvantáláshoz természetesen azt is kell tudni, hogy a kvantálandó értékeknek mi a dinamikatartománya. Ha m-mel jelöljük egy $b\times b$ méretű mátrixban a megengedett maximális értéket, akkor

a szinguláris érték akkor lesz maximális, ha a mátrix csupa m elemből áll. Az $\mathbf{u}$ és $\mathbf{v}$ elemei ekkor egyformák, és értékük $\frac{1}{\sqrt{b}}$, mivel a vektorok normáltak. Az $\mathbf{u}\cdot\mathbf{v}^T$ mátrix elemei tehát $\frac{1}{b}$ nagyságúak, vagyis a keresett maximális szinguláris érték $m \cdot b$.

A világosságjel maximális értéke 8 bites ábrázolás esetén 255 lehet (ha $R = G = B = 255$), a színkülönbségi jeleké pedig $\pm 127,5$ (lásd (13), 13. oldal), így a szinguláris értékek maximuma $255 \cdot b$ illetve $127,5 \cdot b$. Ha tehát n bitre kvantálunk, akkor a legnagyobb helyiértéket 2^{7+b}-re illetve 2^{6+b}-re választva (ez utóbbi esetben egy előjelbittel) le tudjuk kódolni a lehetséges értékeket.

Az $\mathbf{u}$ és $\mathbf{v}$ vektorok normáltak, vagyis elemeik a ± 1 tartományba esnek, és ez a korlát érvényes a transzformált, majd a $\mathbf{w}$-vel súlyozott $\mathbf{u}_{tr}$ és $\mathbf{v}_{tr}$ vektorokra is. Ha tehát a vektorokat n biten tároljuk (melyből 1 bit az előjel), akkor kvantáláskor a legnagyobb helyiértéket 2^{-1}-re, a legalacsonyabbat pedig 2^{-n+1}-re érdemes választani.

3.8. Inverz transzformáció, majd a blokkok és a teljes kép visszaszámolása

Egy blokk egy csatornájának (például a világosságjelnek) visszaszámolása az alábbiak szerint történik:

$$\begin{aligned}\left(\mathbf{U}' &= \mathbf{S}\cdot\mathbf{U}'_{tr}\right)\\ \left(\mathbf{V}' &= \mathbf{S}\cdot\mathbf{V}'_{tr}\right)\\ \mathbf{Y}' &= \mathbf{U}'\cdot\mathbf{\Sigma}'\cdot\mathbf{V}'^T\end{aligned} \tag{19}$$

A színjelek esetén természetesen szükséges egy kétszeres interpoláció is. Az alulmintavételezéshez hasonlóan itt is a legegyszerűbb módszert alkalmaztam: az interpolált mátrix egy 2×2-es részének mindegyik eleme a régi mátrix megfelelő elemének fele (mivel ez az elem a 4 eredeti elem átlagának kétszerese volt, lásd (15)). Ha megvannak az Y, P_R és P_B értékek egy blokkra, akkor először kiszámoljuk a harmadik színkülönbségi jelet, majd az eredeti értékeket is (20) alapján:

$$\begin{aligned}P_G &= -\frac{0,2126\cdot P_R + 0,0772\cdot P_B}{0,7152}\\ R &= Y + P_R\\ G &= Y + P_G\\ B &= Y + P_B\end{aligned} \tag{20}$$

3.9. A tömörítés objektív jellemzőinek meghatározása

A kapott kép szubjektív értékelésén kívül a tömörítést néhány objektív adattal is jellemezhetjük. Az egyik legfontosabb adat a tömörítés hatásfoka, vagyis a tömörített kép és az eredeti kép méretének hányadosa.

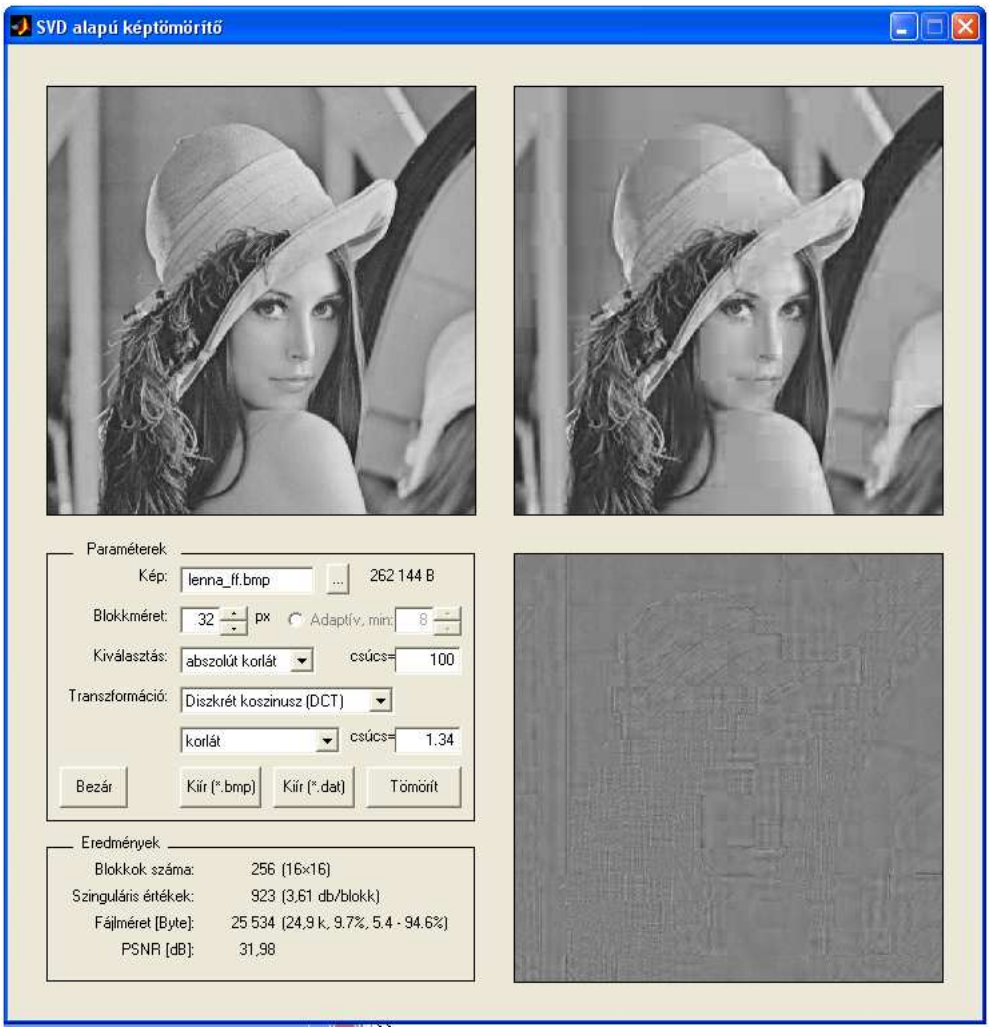
Egy tömörített kép minőségének a leggyakrabban használt objektív mérőszáma a PSNR érték (Peak Signal to Noise Ratio – Csúcsjel–zaj viszony) [9]. A peak szó itt arra utal, hogy nem a kép egy adott pixelértékét, hanem a lehetséges maximumot osztjuk el a hibával. A meghatározásához először kiszámoljuk az egész képre az átlagos négyzetes hibát (MSE – Mean Squared Error), majd a legnagyobb lehetséges pixelérték négyzetét elosztjuk az MSE-vel. Ha az eredeti $n\times m$ méretű, c csatornás kép pixeleit $x_k[i,j]$-vel, a tömörítettét pedig $y_k[i,j]$-vel jelöljük, akkor csatornánként 8

bites képekre a PSNR érték (21) szerint alakul:

$$\begin{aligned} MSE &= \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{c}\left(x_k[i,j]-y_k[i,j]\right)^2}{n\cdot m\cdot c} \\ PSNR_{\mathrm{dB}} &= 10\cdot \lg\left(\frac{255^2}{MSE}\right) \end{aligned} \tag{21}$$

Ezen mérőszám nem veszi figyelembe az emberi látórendszer sajátosságait, így adott esetben két, eltérő jellegű hibákat tartalmazó kép közül a nagyobb PSNR értékű tűnhet rosszabb minőségűnek.

3.10. A grafikus felhasználói felület



5. ábra. A grafikus felhasználói felület

A felhasználói felület négy részre tagolódik: bal felül az eredeti kép, jobb felül a tömörített kép, valamint jobb alul ezek különbsége[3] látható. A tömörítési paraméterek megadása és az eredmények

[3]az elnevezés nem teljesen pontos, mivel ezt a képet úgy kapjuk, hogy az egyszínű 50 %-os szürke képhez adjuk hozzá az eredeti és a tömörített különbségét, így negatív különbségnek sötétebb pixel, pozitív különbségnek pedig világosabb pixel felel meg.

kijelzése a bal alsó részben történik (lásd 5. ábra). A kép megnyitása a [...] gombra kattintással történhet, de be is írhatjuk a kép nevét a megfelelő mezőbe. Ezután felül megjelenik a kép, és a program kiírja az eredeti méretet byte-okban.

Ezután a paramétereket kell megadnunk, először a blokkméretet, valamint ha adaptív blokkra bontást választunk, akkor a minimális blokkméretet is. Ezek alatt egy legördülő menüben szerepelnek a szinguláris értékek kiválasztásának módjai. Az ennek megfelelő paramétert (vagyis a blokkonkénti fix értéket, az abszolút vagy a relatív korlát nagyságát) a mellette lévő mezőben adhatjuk meg. Ezek alatt pedig a transzformáció jellemzői választhatók ki. A tömörítést értelemszerűen a [Tömörít] gombra kattintva végezhetjük el, ekkor megjelenik a kapott kép a jobb felső ábrán, jobb alul pedig a különbségi kép.

A bal alsó részben jelennek meg az eredmények: a blokkok száma (adaptív felbontás esetén az is, hogy melyik méretű blokkból hány darab van), az összesen kiválasztott szinguláris értékek száma, és ennek átlagos blokkonkénti értéke. A fájlméretet byte-ban adja meg, ezek után zárójelben szerepel ez az érték kB-ban és az eredeti mérethez viszonyítva százalékban is. A két utolsó érték a tömörített fájlban lévő járulékos illetve hasznos információ arányát adja meg. A legalsó paraméter pedig a PSNR.

A [Kiír (*.bmp)] gombra kattintva a visszaszámolt képet, a [Kiír (*.dat)] gombbal pedig a tömörített képet menthetjük el a következő fejezetben ismertetett szintaxissal.

4. A tömörített fájl felépítése

A legtöbb bitsebesség-csökkentő eljárásnál csak a kimeneti bitszintaxist szabványosították, a kódoló felépítését nem. A dekódolónak így „nem kell tudnia", hogy a kódoló milyen módon, mekkora számításigénnyel állította elő a tömörített állományt, feltéve hogy helyes szintaxisú a kimenet. Így a kódolási algoritmusok fejlődésével egyre hatékonyabb tömörítés vált elérhetővé úgy, hogy ezeket a régebbi verziójú dekódolók is képesek voltak értelmezni.

Minél hatékonyabb egy tömörítés, annál kevesebb adattal írja le a képet, viszont annál több úgynevezett „overhead" információra van szükség, melyek azt mondják meg, hogy egy konkrét adatot hogyan értelmezzünk. A megvalósított tömörítő eljárás során a beállításoktól függően a letárolandó adatok mennyisége és fajtája is változik, így az itt ismertetett fájlfelépítés első, állandó hosszú részében azok az adatok szereplnek, melyek segítségével a fájl további, változó hosszú részei azonosíthatók.

Ebben az első részben csak néhány byte-nyi információt tárolunk, ugyanakkor ezen adatok nagy része kevesebb biten is leírható lenne. Mivel azonban ezek az adatok csak egyszer szerepelnek, ezért nem sokat számít, hogy például a kép csatornáinak számát (fekete-fehér képnél 1, színesnél 3) nem 1, hanem 8 biten írjuk le.

A további részekben a különböző adatok típusuk szerint és nem pedig helyük szerint vannak csoportosítva. A kódolás és a dekódolás így ugyan egy kicsit bonyolultabb, de a sok ugyanolyan típusú egymás melletti adatot hatékonyabban lehet tárolni, például kevesebb bittel leírva (lásd 4.2.) vagy előre definiált kódolást alkalmazva (lásd 4.4.). A dekódoláshoz emiatt szükség van arra, hogy az egész fájl egy időben a rendelkezésre álljon. Ez egyszerű adattárolás esetén nem probléma, viszont adattovábbításkor a dekódolással meg kell várnunk, míg a teljes adatmennyiség megérkezik.

A fájlfelépítésben próbáltam a redundanciákat minimálisra csökkenteni, így nem foglalkoztam például a dekódolást (és egyben az esetleges hibák detektálását) segítő offszet- és hosszinformációk, markerek kódolásával. Ugyanakkor szem előtt tartottam, hogy az algoritmus továbbfejlesztése, kiegészítése esetén az esetleges új információkat (például az adaptívan választott transzformációk azonosítását az egyes blokkokban vagy új transzformációs mátrixok definiálását) is el lehessen helyezni ebben a szintaxisban.

A fájl felépítése az alábbi:

1. Fejléc, paraméterek
2. A blokkokra bontás azonosítása
3. A transzformációk típusa
4. A szinguláris értékek száma az egyes blokkokban
5. Szinguláris értékek
6. Szinguláris vektorok
7. A transzformációs együtthatók mennyisége
8. Transzformációs együtthatók

4.1. Fejléc, paraméterek

A már említett, fix méretű adatok az alábbiak:

- A formátum azonosítása, hogy egy másmilyen típusú fájl esetén a dekódoló ne értelmezze azt tömörített képként.
- A kép vízszintes és függőleges méretét leíró 2-2 byte, a maximális méret így elvileg $2^{16} \cdot 2^{16} = 65536 \times 65536$ pixel lehet.
- A maximális és a minimális blokkméret (1-1 byte)
- A transzformáció típusának azonosítása (2 byte). Az első 8 biten megadhatjuk a transzformáció típusát, amennyiben az egész képen ezt alkalmaztuk. Ha nem, akkor egy-egy előre definiált kóddal azonosítható az, hogy a transzformáció kiválasztása adaptívan, nagyblokkonként vagy blokkonként történt. A másik byte pedig azt mondja meg, hogy a transzformációk típusánál (4.3) hány új transzformációt definiálunk.
- A kép csatornáinak száma (1 byte)
- A színjelek alulmintavételezésének mértéke (1 byte)
- Egy-egy háromelemű vektor (összesen 6 byte), melyek megadják, hogy a szinguláris értékeket illetve vektorokat hány bitre kvantáljuk
- A fájl teljes mérete (34 byte), mely értelemszerűen nem lesz nagyobb, mint a tömörítetlen fájl maximális mérete, azaz $65536 \cdot 65536 \cdot 3 = 3 \cdot 2^{32}$ B.

Ha a formátum-azonosítás mérete például 10 B, akkor a fejléc mérete összesen 60 B.

4.2. A blokkokra bontás azonosítása

A kép mérete és a maximális blokkméret (azaz a makroblokk méretének) ismeretében könnyen kiszámolható, hogy a képet hány makroblokkra bontottuk fel. Egy ilyenen belül ahhoz, hogy a kisebb blokkokat azonosítsuk, a 3.3. fejezetben említett két paraméter közül elég, ha csak a méreteket tudjuk. Sőt, mivel a legkisebb méretű blokkok a felbontás során mindig négyesével keletkeztek, elég, ha minden négy egymást követő egyforma, a legkisebb blokkot azonosító szám közül csak egyet tartunk meg. Így a 4. ábrán (lásd 15. oldal) szereplő felbontás esetén `1222233332233331` helyett elég, ha az `122232231` sorozatot tároljuk le.

Mivel ezt a számsort minden makroblokknál le kell tárolni, ezért itt érdemes csökkenteni a szükséges méretet. Mivel a kiválasztható blokkméretek mind 2-nek a hatványai, valamint legkisebb és legnagyobb blokkméret ($4 = 2^2$ és $512 = 2^9$) között 7 nagyságrend az eltérés, ezért a számsorban szereplő legnagyobb szám a 7 lehet (ez a minimális, 4 méretű blokk azonosítója, ha a makroblokkok 512 pixel méretűek). Vagyis a számok kódolásához elég 3 bit.

A dekódolás során először kiegészítjük a kapott számsort úgy, hogy minden, a minimális blokkméretnek megfelelő szám (vagyis a példában a 3-as) után beillesztjük ezt a számot még háromszor. Ezután a teljes sorozaton végigmenve egyértelműen azonosítható, hogy melyik blokk mekkora, és hol van.

4.3. A transzformációk típusa

Amennyiben nincs transzformáció, vagy típusa az egész képen állandó, úgy a szintaxisból ez a rész kimarad. Ha definiálunk új transzformációt, akkor először mindegyikre letárolunk egy-egy azonosítót. Ebben a mátrixok minimális és maximális méretei, valamint a kódolási pontosság szerepel. Ezek után következnek maguk a transzformációs és szintézismátrixok.

A fejléc és a blokkokra bontást azonosító számsorok dekódolása után már tudható, hogy makroblokkonként vagy blokkonként választottuk ki a transzformációt, és az is, hogy összesen a képen hány makroblokk és blokk szerepel. Ezeket elég néhány (például 3) biten tárolni (a dekódoló tudni fogja, hogy pl a `001` a DCT-nek felel meg). Ha tudjuk, hogy hány transzformáció-azonosítót kell dekódolnunk, illetve hogy ezek hány bit méretűek, akkor ezek alapján már egyértelműen meghatározható ezen rész hossza is.

4.4. A szinguláris értékek száma az egyes blokkokban

A kódolt szinguláris értékek mennyiségét blokkonként egy három elemű vektorral tudjuk leírni a színes képeknél és egy eleművel a fekete-fehéreknél. Azonban ahelyett, hogy az összes (n darab) blokkra letárolnánk ezt a vektort ($8n \cdot c$ bit, ahol c a színcsatornák száma), érdemes először az összes előforduló vektort (k darab) definiálni (vektoronként c byte-on), majd ezután sorban megmondani, hogy az adott blokkot ezek közül melyik írja le. Ehhez először persze (1 byte-on) meg kell mondanunk k értékét, az egyes blokkoknál pedig a megfelelő vektort a lehető legkevesebb bittel ($\lceil \log_2 k \rceil$) érdemes azonosítani. Ha tehát a (22) egyenlőtlenség teljesül, akkor érdemes ezt a kódolást alkalmazni.

$$8n \cdot c > 8 + 8k \cdot c + n \cdot \lceil \log_2 k \rceil \tag{22}$$

4.5. Szinguláris értékek

Az előző rész dekódolása után már tudható, hogy összesen hány szinguláris érték van, azok melyik színcsatornához tartoznak, és hány bittel vannak kódolva. Így ezen számokat bitfolytonosan tárolhatjuk.

4.6. Szinguláris vektorok

A szinguláris értékek után (ha nem volt transzformáció) a szinguláris vektorokat tároljuk, szintén bitfolytonosan. A kódolási sorrend (azaz hogy blokkonként vagy csatornánként tároljuk őket) nem befolyásolja a kimeneti fájl méretét.

4.7. A transzformációs együtthatók mennyisége

Ha volt transzformáció, akkor a szinguláris értékek kódolása után a transzformációs együtthatók következnek. Egy b méretű vektornál $\log_2 b$ biten tudjuk megmondani, hogy az első hány ilyen együtthatót választottuk ki.

4.8. A transzformációs együtthatók

A transzformációs együtthatók dekódolásakor már tudjuk, hogy a blokkok milyen méretűek, blokkonként hány vektort illetve vektoronként hány együtthatót kódoltunk. Így ezen együtthatókat is egyszerűen, bitfolytonosan tárolhatjuk.

5. Tesztelés, eredmények

Ebben a fejezetben bemutatom a tömörítőalgoritmust néhány képen. Mivel a tömörítés hatása minden képre más, érdemes olyan tesztképeket választani, melyeken sokféle képtartalom előfordul. A legismertebb ilyen kép a „Lena” [10], forrás: [11], melyet 1973 óta alkalmaznak a különféle bitsebesség-csökkentő eljárások tesztelésére, illusztrálására. Ezen kívül két, a Kodak által kibocsátott tesztképet („Flowers” és „Parrots”, forrás: [12]) választottam, melyeken nagy, egyszínű felületek, részletgazdag területek és jellegzetes átlós mintázatok is szerepelnek. E három tesztképet a 6. ábrán láthatjuk.

Lena

Flowers

Parrots

6. ábra. A használt tesztképek

Ahhoz, hogy az eredmények jól összehasonlíthatók legyenek, érdemes vagy a képméretet vagy a PSNR-t állandó értéken tartani. Én ez utóbbit választottam, vagyis a tömörítés hatákonyságát a kimeneti fájlméret határozza meg. E méretet kilobyte-ban illetve az eredeti (bmp) fájl méretéhez viszonyítva, százalékban is megadom. A tesztelés során a különböző paraméterek megfelelő beállításaival biztosítottam, hogy a PSNR értéke nagyjából állandó, 35 dB illetve 38 dB volt. Tapasztalataim szerint JPEG esetén 38 dB-es PSNR értékű tömörítés még nem okoz jelentős képminőség-romlást, a 35 dB-es viszont már igen. E kétféle minőségű JPEG képek méretét az 1. táblázat tartalmazza.

1. táblázat. A JPEG tömörítés eredményei

Kép	Eredeti méret	38 dB		35 dB	
Lena	257 kB	30 kB	11,7 %	14,9 kB	5,8 %
Flowers	1152 kB	68,4 kB	5,9 %	42,6 kB	3,7 %
Parrots	1152 kB	50,2 kB	4,4 %	25,5 kB	2,2 %

5.1. A blokkok mérete

A különböző blokkméretek vizsgálatánál a szinguláris értékek kiválasztása az abszolút korlát szerint történt, e korlát értékét mindig úgy állítottam be, hogy a PSNR érték 38 dB illetve 35 dB körül legyen.

2. táblázat. A blokkok mérete

Kép	Blokkméret	38 dB		35 dB	
Lena	64	141,3 kB	55,2 %	71,2 kB	27,8 %
	32	97,9 kB	38,2 %	66,1 kB	25,8 %
	16	95,3 kB	37,2 %	69,1 kB	27 %
	8	114,6 kB	44,8 %	93,8 kB	36,6 %
	4	175,3 kB	68,5 %	165,5 kB	64,6 %
	adaptív, 64 – 16	96,5 kB	37,7 %	63,8 kB	24,9 %
	adaptív, 64 – 8	103,3 kB	40,4 %	68,6 kB	26,8 %
	adaptív, 64 – 4	154,1 kB	60,2 %	81,5 kB	31,9 %
Flowers	64	274,1 kB	23,8 %	184,1 kB	16 %
	32	222,8 kB	19,3 %	160,7 kB	14 %
	16	214,7 kB	18,6 %	170,7 kB	14,8 %
	8	269,9 kB	23,4 %	227,8 kB	19,8 %
	adaptív, 64 – 16	205,7 kB	17,9 %	147,8 kB	12,8 %
	adaptív, 64 – 8	232,5 kB	20,2 %	184,1 kB	16 %
Parrots	64	384,5 kB	33,4 %	133,3 kB	11,6 %
	32	191,6 kB	16,6 %	112,6 kB	9,8 %
	16	173,1 kB	15,1 %	129,1 kB	11,2 %
	8	248,1 kB	21,5 %	208,9 kB	18,1 %
	adaptív, 64 – 16	200,2 kB	17,4 %	113,4 kB	9,8 %
	adaptív, 64 – 8	222,1 kB	19,3 %	124,1 kB	10,8 %

A táblázatból látszik, hogy adaptív blokkméret esetén az eljárás nagyjából ugyanannyira hatékony, mint az optimális, 16-os illetve a 32-es állandó blokkméret esetében. A 7. ábrán összehasonlíthatjuk az állandó, 32, illetve az adaptívan, 64 és 16 pixel között választott blokkméret hatását két egyforma minőségű (PSNR=30 dB) képen. Látható, hogy az adatpív algoritmus a részletgazdag területek, átlós mintázatok környékén kisebb, míg a nagyjából egyszínű háttérnél pedig nagyobb blokkméretet alkalmazott.

Blokkméret: 32 pixel Blokkméret: 64 – 16 pixel

7. ábra. Az állandó és változó méretű blokkokra bontás összehasonlítása

5.2. A szinguláris értékek kiválasztásának módja

A kiválasztás módját mindhárom képnél az állandó 32-es illetve a 64–16 pixel között adaptívan változó blokkméret esetén teszteltem. A kapott eredmények a 3. táblázatban láthatók.

Megfigyelhető, hogy mindegyik esetben az abszolút korlát alkalmazásával kapjuk a legjobb eredményt, relatív korlát esetén ennél pár százalékkal lesz nagyobb a kép, míg állandó számú érték és statikus blokkméret esetén a tömörítés utáni méret ennek szinte kétszerese is lehet. Adaptív blokkméretnél nincs ekkora különbség, hiszen a blokkokra bontást úgy végeztük, hogy mindegyik blokkot nagyjából ugyanannyi értékkel tudjunk leírni.

A következő két összehasonlítás során az itt optimálisnak talált 32-es blokkméretet és abszolút korlát szerinti kiválasztást alkalmaztam.

5.3. A transzformáció típusa

Ahhoz, hogy a transzformációk hatása összehasonlítható legyen, úgy kell beállítani a paramétereket, hogy a transzformáció utáni PSNR értékek megegyezzenek, és érdemes ezeket több kiindulási képminőségre is megvizsgálni. Három minőségi tartományban, 41 dB → 38 dB, 38 dB → 35 dB és 35 dB → 32 dB között hasonlítottam össze az eljárásban kiválasztható háromféle transzformációt.

3. táblázat. A kiválasztás módja

Kép	Blokkméret	Kiválasztás	38 dB		35 dB	
Lena	32	adott n	164,6 kB	64,3 %	109,8 kB	42,9 %
		absz. korlát	97,9 kB	38,2 %	66,1 kB	25,8 %
		rel. korlát	107,9 kB	42,1 %	69,1 kB	27 %
	64 – 16	adott n	143,9 kB	56,2 %	86,4 kB	33,8 %
		absz. korlát	96,5 kB	37,7 %	63,8 kB	24,9 %
		rel. korlát	109,2 kB	42,6 %	70,1 kB	27,4 %
Flowers	32	adott n	340,7 kB	29,6 %	265,9 kB	23,1 %
		absz. korlát	222,8 kB	19,3 %	160,7 kB	14 %
		rel. korlát	232,8 kB	20,2 %	164,8 kB	14,3 %
	64 – 16	adott n	300,6 kB	26,1 %	228,2 kB	19,8 %
		absz. korlát	205,7 kB	17,9 %	147,8 kB	12,8 %
		rel. korlát	230,8 kB	20 %	169,4 kB	14,7 %
Parrots	32	adott n	370,9 kB	32,2 %	207,1 kB	18 %
		absz. korlát	191,6 kB	16,6 %	112,6 kB	9,8 %
		rel. korlát	209,9 kB	18,2 %	121,8 kB	10,6 %
	64 – 16	adott n	285,3 kB	24,8 %	157,9 kB	13,7 %
		absz. korlát	200,2 kB	17,4 %	113,4 kB	9,8 %
		rel. korlát	210,9 kB	18,3 %	121,8 kB	10,6 %

Az eredményeket a 4. táblázat mutatja. A 38 dB, 35 dB és 32 dB feliratú oszlopokban a transzformáció nélkül kapott képméretek (hely hiányában csak százalékban megadva), a másik három (dupla) oszlopban pedig az ugyanilyen PSNR értékű, de transzformációval kapott képek méretei szerepelnek.

4. táblázat. A transzformáció hatása

Kép	Transzf.	41 dB → 38 dB		38 dB	38 dB → 35 dB		35 dB	35 dB → 32 dB		32 dB
Lena	DCT	135 kB	52,6 %	38,2%	55,7 kB	21,8 %	25,8%	24,9 kB	9,7 %	18,2%
	Hsz. 1	140 kB	54,5 %		61,2 kB	23,9 %		27,8 kB	10,9 %	
	Hsz. 2	143 kB	55,8 %		64,9 kB	25,3 %		30,1 kB	11,8 %	
Flowers	DCT	250 kB	21,7 %	19,3%	159,9 kB	13,9 %	14%	87 kB	7,6 %	9,8%
	Hsz. 1	260 kB	22,6 %		171,7 kB	14,9 %		93,3 kB	8,1 %	
	Hsz. 2	266 kB	23,1 %		176,8 kB	15,3 %		96,8 kB	8,4 %	
Parrots	DCT	421 kB	36,6 %	16,6%	124,9 kB	10,8 %	9,8%	44,4 kB	3,9 %	6,7%
	Hsz. 1	429 kB	37,2 %		131,5 kB	11,4 %		46,8 kB	4,1 %	
	Hsz. 2	434 kB	37,6 %		135,3 kB	11,7 %		48,5 kB	4,2 %	

A táblázatból látszik, hogy a transzformációk hatékonysága azonos mértékű képminőség-romlás mellett döntően függ a kiindulási kép minőségétől. 38 dB-es PSNR érték felett érdemes inkább a szinguláris értékek számát csökkenteni, míg alatta a transzformációk alkalmazása tűnik hatékonyabbnak. A transzformáció-fajták hatékonysága között nincs jelentős eltérés, de a DCT kis mér-

tékben jobb a háromszög-transzformációknál. A transzformáció képminőségre gyakorolt hatását a 8. ábrán láthatjuk. A képek PSNR értéke itt is egyforma (30 dB), a baloldali a transzformáció nélkül kapott kép, a jobboldalinál pedig DCT-t alkalmaztam, a kiindulási kép a 35 dB-es volt.

Transzformáció nélkül DCT transzformációval

8. ábra. A transzformáció hatása

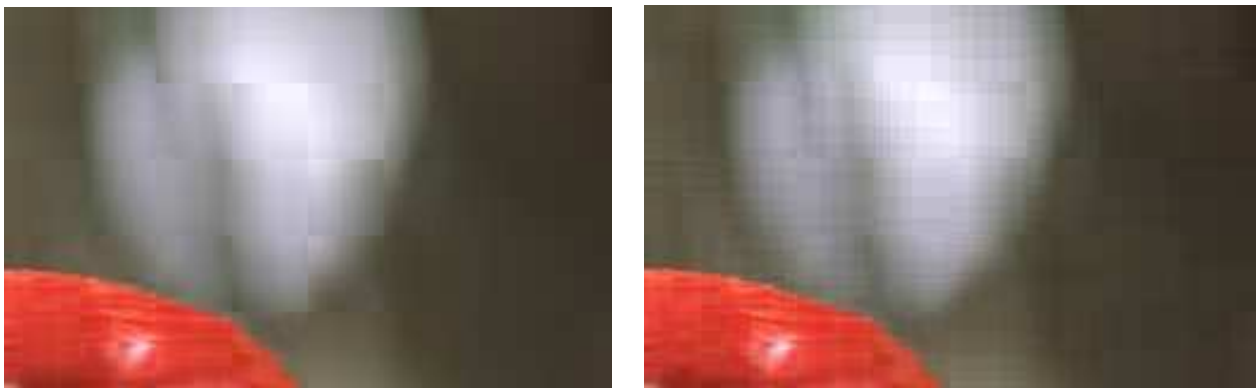
5.4. Kvantálás

Az eljárásban megadható, hogy a különböző színcsatornákban a szinguláris értékeket és a vektorokat hány bites számként tároljuk le. A különböző bitmélységek tesztelésénél a szinguláris értékekre és vektorokra is az 5. táblázat második oszlopában megadott együtthatókat alkalmaztam. Ahol nincsenek megadva a méretek, az arra utal, hogy azokkal a bitszámokkal nem lehet elérni a 38 dB-es PSNR értéket. A táblázatból látszik, hogy a tömörítés hatékonyságát döntően a világosságjel bitmélysége határozza meg.

5. táblázat. A kvantálás okozta minőségromlás

Kép	A bitek száma	38 dB		35 dB	
Lena	[10]	109,5 kB	42,8 %	74,9 kB	29,3 %
	[9]	107,9 kB	42,1 %	69,1 kB	27 %
	[8]	233,2 kB	91,1 %	75,1 kB	29,3 %
Flowers	[10, 8, 8]	239,1 kB	20,8 %	178,9 kB	15,5 %
	[9, 8, 8]	229 kB	19,9 %	170,1 kB	14,8 %
	[9, 7, 7]	233,9 kB	20,3 %	165,6 kB	14,4 %
	[8, 8, 8]	317,7 kB	27,6 %	169,8 kB	14,7 %
	[8, 7, 7]	309,6 kB	26,9 %	168,5 kB	14,6 %
	[8, 6, 6]	– kB	– %	168,1 kB	14,6 %
Parrots	[10, 8, 8]	199,8 kB	17,3 %	127,8 kB	11,1 %
	[9, 8, 8]	202,7 kB	17,6 %	121,7 kB	10,6 %
	[9, 7, 7]	205,2 kB	17,8 %	120,7 kB	10,5 %
	[8, 8, 8]	– kB	– %	128,5 kB	11,2 %
	[8, 7, 7]	– kB	– %	129 kB	11,2 %
	[8, 6, 6]	– kB	– %	132,4 kB	11,5 %

A PSNR érték ismertetésénél említettem, hogy ezen mérőszám értékéből nem feltétlenül lehet a szubjektíven megítélt minőségre következtetni. Erre mutat egy példát a 9. ábra, ahol a Parrots kép jobb felső részlete látható különböző kvantálások esetén.



kvantálás: [10, 8, 8] kvantálás: [8, 7, 7]

9. ábra. Azonos PSNR értékű képek részleteinek összehasonlítása

A képek PSNR értéke megegyezik, és a teljes méretük is majdnem ugyanakkora (lásd 5. táblázat). Mivel a jobboldali képen minden számot kevesebb bittel írunk le, több szinguláris értéket tudunk tárolni, és emiatt a blokkok határa kevésbé észrevehető. Ugyanakkor a képen zavaró a kvantálás hatása. A baloldali képen ezzel szemben a kvantálás nem vehető észre, de kevesebb szinguláris értéket tudtunk tárolni, így a blokkok becslése pontatlanabb lett.

5.5. A járulékos információk mértéke

Ebben a részben összehasonlítok néhány, az előzőekben optimálisnak talált beállítást abból a szempontból, hogy a kimeneti képméret hány százaléka a hasznos adat, és hány százaléka segédinformáció. Az előbbiek közé a szinguláris értékek (4.5.) és vagy a szinguláris vektorok (4.6.), vagy a transzformációs együtthatók (4.8.) tartoznak. Az utóbbiak közé pedig a fejléc (4.1.), a blokkokra bontás azonosítása (4.2.), a transzformációk típusa (4.3.), valamint a szinguláris értékek és a transzformációs együtthatók mennyisége (4.4.) és (4.7.).

6. táblázat. A járulékos információk mértéke

Blokkméret	Kép	38 dB		35 dB		38 dB → 35 dB, DCT	
32	Lena	38,2 %	0,2 – 99,8	25,8 %	0,3 – 99,7	21,8 %	3,4 – 96,6
	Flowers	20 %	0,4 – 99,6	14,3 %	0,4 – 99,6	13,9 %	3,7 – 96,3
	Parrots	17,2 %	0,4 – 99,6	9,5 %	0,6 – 99,4	10,8 %	4,3 – 95,7
64-16	Lena	37,7 %	0,5 – 99,5	24,9 %	0,6 – 99,4	22,14 %	5,3 – 94,7
	Flowers	18,3 %	0,6 – 99,4	13,2 %	0,7 – 99,3	12,6 %	5,3 – 94,7
	Parrots	18 %	0,6 – 99,4	10,2 %	0,8 – 99,2	12,3 %	5,1 – 94,9

A 6. táblázatban szereplő számpáreok első tagja a járulékos információnak, a második tagja pedig a hasznos információnak a százalékos arányát jelzi a tömörített fájlon belül. Látszik, hogy ha nem alkalmazunk transzformációt, akkor a járulékos információ mennyisége nem éri el az 1 %-ot a tömörített fájlban, és transzformáció esetén is csak néhány százalék.

6. Összefoglalás

Dolgozatomban részletesen bemutattam a saját fejlesztésű tömörítőeljárást, és összehasonlítottam a különböző paraméterekkel kapott eredményeket. Ezek alapján elmondható, hogy megfelelő beállításokkal a kép mérete akár 10–20 %-ára is csökkenthető, elfogadható mértékű minőségromlás (azaz 35–38 dB-es PSNR érték) mellett.

Idő hiányában az algoritmus több része még nem lett kidolgozva tökéletesen, terveim között szerepel az eljárás továbbfejlesztése, vizsgálata több helyen.

- Az adaptív felbontást végző algoritmus egy adott blokkra mindig eldönti, hogy hatékonyan becsülhető-e a blokk az SVD-vel. E döntés akkor tud optimális lenni, hogyha ez az algoritmus valóban elvégzi a szinguláris érték szerinti felbontást mindegyik blokkra, és a kapott értékek alapján dönti el, hogy hatékony lehet-e a becslés, vagy ossza-e fel inkább a blokkot négy kisebbre, és ezeket vizsgálja. Megfelelő feltételek beállításával biztosítható, hogy a blokkokra bontás közel optimális legyen, ugyanakkor az algoritmus számításigénye így jelentősen növekszik.

- Az eljárásban egyelőre csak kétféle transzformáció választható ki, szeretném ezt továbbiakkal bővíteni. Amennyiben ezek mátrixai jelentősen eltérnek egymástól, úgy egy adott vektor transzformáltja is alapvetően másmilyen lehet. Ekkor érdemes a transzformációt is adaptívan végezni, minden egyes blokkra kiválasztjuk az előre definiáltak közül, hogy melyik a leghatékonyabb.

- Mivel a kódolt fájlban lévő információ jelentős része hasznos adat, ennek egy része (például a gyakran ismétlődő vektorok) valószínűleg hatékonyan kódolható valamilyen változó-hosszú kódolással.

A megvalósított tömörítő eljárás jelenlegi állapotában egy jó kiindulási alap, bízom benne, hogy a további fejlesztések során fel tudja majd venni a versenyt a JPEG tömörítéssel.

Ábrák jegyzéke

Táblázatok jegyzéke

Hivatkozások

[1] On-line képtömörítési demonstráció
`http://demonstrations.wolfram.com/ImageCompressionViaTheSingularValueDecomposition`

[2] Howard Anton, Robert C. Busby, *Contemporary Linear Algebra*, Anton Textbooks Inc, 2003

[3] Carl Meyer, *Matrix Analysis and Applied Linear Algebra*, Society of Industrial and Applied Mathematics, 2000

[4] Kovács Imre, *Videó-stúdiótechnika*, órai jegyzet, BME, 2010

[5] Az ITU-R BT.709-es ajánlása, `http://www.itu.int/rec/R-REC-BT.709/en`

[6] Mathias Wien, *Variable Block-Size Transforms for H.264/AVC*, IEEE Transactions On Circuits And Systems For Video Technology, Vol. 13, No. 7, pp. 604–613, July 2003

[7] Yu-Kuang Tua, Jar-Ferr Yanga, Ming-Ting Sunb, Yuesheng T. Tsaic, *Fast variable-size block motion estimation for efficient H.264/AVC encoding*, Signal Processing: Image Communication pp.595–623, 2005

[8] Ken Cabeen and Peter Gent, *Image Compression and the Discrete Cosine Transform*, College of Redwoods

[9] Peak signal-to-noise ratio, `http://en.wikipedia.org/wiki/PSNR`

[10] Lenna, `http://en.wikipedia.org/wiki/Lenna`

[11] University of Southern California, Signal and Image Processing Institute Image Database, `http://sipi.usc.edu/database/database.php?volume=misc`

[12] True Color Kodak Images, `http://r0k.us/graphics/kodak`