

## (curs 12 - S4)

## 9. Compresia

- cunoscutul format JPEG, de exemplu, aplică TCD bi-dimensională pentru blocuri de  $8 \times 8$  pixeli dintr-o imagine, și stochează rezultatele folosind codificarea Huffman
- detaliiile compresiei JPEG sunt investigate ca un studiu de caz în Secțiunile 9.2–9.3
- o versiune modificată a Transformantei Cosinus Discrete, numită Transformata Cosinus Discretă Modificată (TCDM), este baza majorității formelor de compresie audio moderne
- TCDM este formatul standard curent pentru compresia fișierelor de sunet
- vom introduce TCDM și investigăm aplicarea ei pentru codificare și decodificare, care oferă tehnologia de bază pentru formatele de fișiere cum ar fi MP3 și AAC (Advanced Audio Coding)

### 9.1. Transformata cosinus disretă

#### 9.1.1. TCD uni-dimensională

- fie  $n$  un întreg pozitiv
- Transformata Cosinus Discretă uni-dimensională de ordinul  $n$  este definită de matricea  $n \times n C$  ale cărei intrări sunt

$$C_{ij} = \sqrt{\frac{2}{n}} a_i \cos \frac{i(2j+1)\pi}{2n}, \quad (1)$$

pentru  $i, j = 0, \dots, n-1$ , unde

$$a_i \equiv \begin{cases} 1/\sqrt{2} & \text{dacă } i = 0, \\ 1 & \text{dacă } i = 1, \dots, n-1, \end{cases}$$

sau

$$C = \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \cos \frac{\pi}{2n} & \cos \frac{3\pi}{2n} & \cdots & \cos \frac{(2n-1)\pi}{2n} \\ \cos \frac{2\pi}{2n} & \cos \frac{6\pi}{2n} & \cdots & \cos \frac{2(2n-1)\pi}{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \cos \frac{(n-1)\pi}{2n} & \cos \frac{(n-1)3\pi}{2n} & \cdots & \cos \frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix}. \quad (2)$$

- în cazul imaginilor bi-dimensionale, convenția este de a începe cu 0 în loc de 1
- notația va fi mai ușoară dacă extindem această convenție la numerotarea matricilor, după cum am făcut în (1)
- în acest capitol, indicii pentru matricile  $n \times n$  vor fi între 0 și  $n-1$
- pentru simplitate, vom trata doar cazul în care  $n$  este par în următoarea discuție

#### Definiția 1

Fie  $C$  matricea definită în (2). **Transformata Cosinus Discretă** (TCD) a lui  $x = [x_0, \dots, x_{n-1}]^T$  este vectorul  $n$ -dimensional  $y = [y_0, \dots, y_{n-1}]^T$ , unde

$$y = Cx. \quad (3)$$

- observăm că  $C$  este o matrice reală ortogonală, ceea ce înseamnă că transpusa este egală cu inversa ei:

$$C^{-1} = C^T = \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \cos \frac{\pi}{2n} & \cdots & \cos \frac{(n-1)\pi}{2n} \\ \frac{1}{\sqrt{2}} & \cos \frac{3\pi}{2n} & \cdots & \cos \frac{(n-1)3\pi}{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & \cos \frac{(2n-1)\pi}{2n} & \cdots & \cos \frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix}. \quad (4)$$

- rândurile unei matrici ortogonale sunt vectori unitari ortogonali doi către doi
- ortogonalitatea lui  $C$  rezultă din faptul că vectorii care reprezintă coloanele lui  $C^T$  sunt vectori proprii unitari ai matricii reale simetrice  $n \times n$

$$\begin{bmatrix} 1 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}. \quad (5)$$

- faptul că  $C$  este o matrice reală ortogonală este ceea ce face TCD să fie utilă
- Teorema interpolării prin funcții ortogonale aplicată matricii  $C$  implică Teorema 1

### Teorema 1 (Teorema de interpolare a TCD)

Fie  $x = [x_0, \dots, x_{n-1}]^T$  un vector de  $n$  numere reale. Definim  $y = [y_0, \dots, y_{n-1}]^T = Cx$ , unde  $C$  este matricea Transformatei Cosinus Discrete de ordinul  $n$ . Atunci funcția reală

$$P_n(t) = \frac{1}{\sqrt{n}} y_0 + \sqrt{\frac{2}{n}} \sum_{k=1}^{n-1} y_k \cos \frac{k(2t+1)\pi}{2n}$$

satisfacă  $P_n(j) = x_j$  pentru  $j = 0, \dots, n-1$ .

- Teorema 1 arată că matricea  $n \times n$   $C$  transformă  $n$  puncte în  $n$  coeficienți de interpolare
- Ia fel ca Transformata Fourier Discretă, Transformata Cosinus Discretă ne dă coeficienții pentru o funcție de interpolare trigonometrică
- Spre deosebire de TFD, TCD folosește doar termenii cu cosinus și este definită doar în termeni de numere reale

### Exemplu 1

- folosiți TCD pentru a interpola punctele  $(0, 1), (1, 0), (2, -1), (3, 0)$
- este folositor să observăm, folosind trigonometria elementară, că matricea  $4 \times 4$  a TCD poate fi privită în forma

$$C = \frac{1}{\sqrt{2}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos \frac{\pi}{8} & \cos \frac{3\pi}{8} & \cos \frac{5\pi}{8} & \cos \frac{7\pi}{8} \\ \cos \frac{2\pi}{8} & \cos \frac{6\pi}{8} & \cos \frac{10\pi}{8} & \cos \frac{14\pi}{8} \\ \cos \frac{3\pi}{8} & \cos \frac{9\pi}{8} & \cos \frac{15\pi}{8} & \cos \frac{21\pi}{8} \end{bmatrix} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}, \quad (6)$$

unde

$$a = \frac{1}{2}, \quad b = \frac{1}{\sqrt{2}} \cos \frac{\pi}{8} = \frac{\sqrt{2+\sqrt{2}}}{2\sqrt{2}}, \quad c = \frac{1}{\sqrt{2}} \cos \frac{3\pi}{8} = \frac{\sqrt{2-\sqrt{2}}}{2\sqrt{2}}. \quad (7)$$

- TCD de ordinul 4 înmulțită cu datele  $x = [1, 0, -1, 0]^T$  este

$$\begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ c+b \\ 2a \\ c-b \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\sqrt{2-\sqrt{2}}+\sqrt{2+\sqrt{2}}}{2\sqrt{2}} \\ 2 \\ \frac{\sqrt{2-\sqrt{2}}-\sqrt{2+\sqrt{2}}}{2\sqrt{2}} \end{bmatrix} \approx \begin{bmatrix} 0.0000 \\ 0.9239 \\ 1.0000 \\ -0.3827 \end{bmatrix}.$$

Potrivit Teoremei 1 cu  $n = 4$ , funcția

$$P_4(t) = \frac{1}{\sqrt{2}} \left[ 0.9239 \cos \frac{(2t+1)\pi}{8} + \cos \frac{2(2t+1)\pi}{8} - 0.3827 \cos \frac{3(2t+1)\pi}{8} \right] \quad (8)$$

interpolează cele patru puncte

funcția  $P_4(t)$  este reprezentată grafic sub forma curbei continue în Figura 1

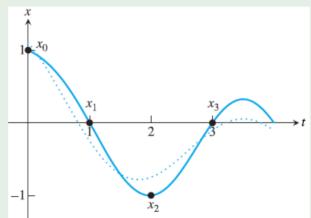


Figura 1: Interpolarea TCD și aproximarea de tip cele mai mici pătrate. Punctele sunt  $(j, x_j)$ , unde  $x = [1, 0, -1, 0]^T$ . Funcția de interpolare folosind TCD  $P_4(t)$  din (8) este prezentată ca o curbă continuă, împreună cu funcția de aproximare de tip cele mai mici pătrate folosind TCD  $P_3(t)$  din (9) ca o curbă punctată.

## 9.1.2. TCD și aproximarea de tip cele mai mici patrate

### Teorema 2 (Teorema de aproximare de tip cele mai mici pătrate a TCD)

Fie  $x = [x_0, \dots, x_{n-1}]^T$  un vector de  $n$  numere reale. Definim  $y = [y_0, \dots, y_{n-1}]^T = Cx$ , unde  $C$  este matricea Transformatei Cosinus Discrete. Atunci, pentru orice întreg pozitiv  $m \leq n$ , alegerea coeficienților  $y_0, \dots, y_{m-1}$  din

$$P_m(t) = \sqrt{\frac{1}{n}}y_0 + \sqrt{\frac{2}{n}} \sum_{k=1}^{m-1} y_k \cos \frac{k(2t+1)\pi}{2n}$$

minimizează eroarea pătratică de aproximare  $\sum_{j=0}^{n-1} (P_m(j) - x_j)^2$  a celor  $n$  puncte.

- referindu-ne la Exemplul 1, dacă dorim să găsim cea mai bună aproximare de tip cele mai mici pătrate pentru aceleași patru puncte, dar folosind cele trei funcții de bază de mai jos

$$1, \cos \frac{(2t+1)\pi}{8}, \cos \frac{2(2t+1)\pi}{8}$$

atunci soluția este

$$P_3(t) = \frac{1}{2} \cdot 0 + \frac{1}{\sqrt{2}} \left[ 0.9239 \cos \frac{(2t+1)\pi}{8} + \cos \frac{2(2t+1)\pi}{8} \right]. \quad (9)$$

- Figura 1 compară soluția de tip cele mai mici pătrate  $P_3$  cu funcția de interpolare  $P_4$ .

### Exemplul 2

- folosind TCD și Teorema 2, găsiți interpolările de tip cele mai mici pătrate pentru punctele  $t = 0, \dots, 7$  și  $x = [-2.2, -2.8, -6.1, -3.9, 0.0, 1.1, -0.6, -1.1]^T$  pentru  $m = 4, 6$ , și 8
- luând  $m = 8$ , găsim că TCD a datelor este

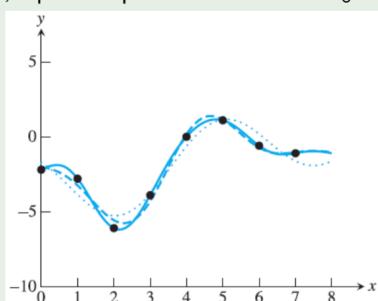
$$y = Cx = \begin{bmatrix} -5.5154 \\ -3.8345 \\ 0.5833 \\ 4.3715 \\ 0.4243 \\ -1.5504 \\ -0.6243 \\ -0.5769 \end{bmatrix}.$$

potrivit Teoremei 1, interpolantul cosinus discret al celor opt puncte este

$$\begin{aligned} P_8(t) = & \frac{1}{\sqrt{8}}(-5.5154) + \frac{1}{2} \left[ -3.8345 \cos \frac{(2t+1)\pi}{16} + 0.5833 \cos \frac{2(2t+1)\pi}{16} \right. \\ & + 4.3715 \cos \frac{3(2t+1)\pi}{16} + 0.4243 \cos \frac{4(2t+1)\pi}{16} \\ & - 1.5504 \cos \frac{5(2t+1)\pi}{16} - 0.6243 \cos \frac{6(2t+1)\pi}{16} \\ & \left. - 0.5769 \cos \frac{7(2t+1)\pi}{16} \right]. \end{aligned}$$

interpolantul  $P_8$  este reprezentat grafic în Figura 2, împreună cu interpolările de tip cele mai mici pătrate  $P_6$  și  $P_4$ .

ultimele două sunt obținute, potrivit Teoremei 2, prin păstrarea primilor șase, și, respectiv, a primilor patru termeni ai lui  $P_8$ .



**Figura 2: Interpolarea TCD și aproximarea de tip cele mai mici pătrate.**  
Curba continuă este interpolantul TCD al punctelor din Exemplul 2. Curba întreruptă este interpolarea de tip cele mai mici pătrate folosind doar primii șase termeni, și curba punctată este aproximarea folosind patru termeni.

## 9.2. TCD bi-dimensională și compresia imaginilor

- Transformata Cosinus Discretă este adesea folosită pentru compresia unor blocuri mici dintr-o imagine, de exemplu de  $8 \times 8$  pixeli
- compresia este cu pierdere de informații, ceea ce înseamnă că anumite informații din bloc sunt ignorate
- caracteristica principală a TCD este că ajută la organizarea informației astfel încât partea care este ignorată este exact partea la care ochiul uman este cel mai puțin sensibil
- mai precis, TCD ne va arăta cum să interpolăm datele cu o mulțime de funcții de bază care sunt în ordinea descrescătoare a importanței din punctul de vedere al sistemului vizual uman; termenii mai puțin importanți ai interpolării pot fi ignorati, dacă se dorește acest lucru
- apoi, vom aplica ceea ce am dedus despre TCD pentru a efectua compresia imaginilor
- folosind împreună quantizarea și codificarea Huffman, fiecare bloc de dimensiune  $8 \times 8$  al unei imagini poate fi redus la un flux de biți care poate fi stocat cu fluxuri de biți din celelalte blocuri ale imaginii
- fluxul de biți complet este decodat atunci când o imagine trebuie decompresată și afișată, prin inversarea procesului de codificare
- vom descrie această abordare, numită Baseline JPEG, metoda standard pentru stocarea imaginilor JPEG

### 9.2.1 TCD bi-dimensională

- TCD bi-dimensională este pur și simplu TCD uni-dimensională aplicată în două dimensiuni, una după cealaltă
- poate fi folosită pentru a interpola sau aproxima date care sunt prezentate sub formă unei grile bi-dimensionale, într-o analogie simplă cu cazul uni-dimensional
- în contextul procesării de imagini, grila bi-dimensională reprezintă un bloc de valori ale pixelilor—de exemplu, intensități în tonuri de gri sau intensități color
- doar în acest capitol, vom lista coordonata verticală prima dată și apoi coordonata orizontală când ne vom referi la un punct bi-dimensional, după cum se arată în Figura 3
- scopul este de a fi consistentă cu convenția uzualeă pentru matrici, în care indicele  $i$  întrările  $x_{ij}$  se modifică de-a lungul direcției verticale, și indicele  $j$  de-a lungul direcției orizontale
- o aplicație importantă a acestei secțiuni este pentru fișiere de pixeli care reprezintă imagini, care sunt privite cel mai natural ca fiind matrice de numere
- Figura 3 prezintă o grilă de puncte  $(s, t)$  în planul bi-dimensional cu valorile asignate  $x_{ij}$  fiecarui punct  $(s_i, t_j)$  din grila rectangulară

- TCD-2D realizează aceasta în mod optim din punctul de vedere al celor mai mici pătrate, ceea ce înseamnă că interpolarea se degradează cât de puțin posibil pe măsură ce se renunță la funcții de bază din funcția de interpolare
- TCD-2D este TCD uni-dimensională aplicată succesiv atât direcției orizontale cât și direcției verticale
- considerăm matricea  $X$  constând din valorile  $x_{ij}$ , ca în Figura 3
- pentru a aplica TCD-1D în direcția verticală  $s$ , trebuie să luăm mai întâi transpusa lui  $X$ , și apoi să înmulțim cu  $C$
- coloanele rezultante sunt TCD-1D-urile rândurilor lui  $X$
- fiecare coloană a lui  $CX^T$  corespunde unui  $t_j$  fixat
- a face o TCD-1D în direcția  $t$  înseamnă a ne mișca de-a lungul rândurilor; astfel, luând transpusa și înmulțind cu  $C$  din nou, obținem

$$C(CX^T)^T = CXCT. \quad (10)$$

#### Definiția 2

**Transformata Cosinus Discretă bi-dimensională (TCD-2D)** a matricii  $n \times n X$  este matricea  $Y = CXCT$ , unde  $C$  este definită în (1).

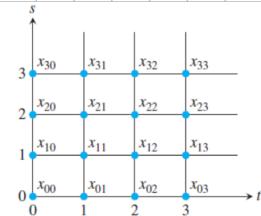
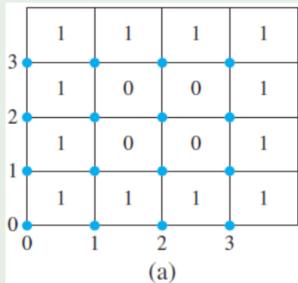


Figura 3: Grilă bi-dimensională de puncte. TCD-2D poate fi folosită pentru a interpola valori de funcții pe o grilă pătratică, cum ar fi valorile pixelilor dintr-o imagine.

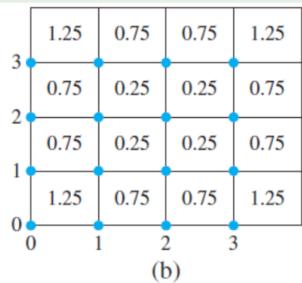
- pentru concretețe, vom folosi grila cu numere întregi  $s_i \in \{0, 1, \dots, n - 1\}$  (de-a lungul axei verticale) și  $t_j \in \{0, 1, \dots, n - 1\}$  de-a lungul axei orizontale
- scopul TCD bi-dimensional este de a construi o funcție de interpolare  $F(s, t)$  care interpolează cele  $n^2$  puncte  $(s_i, t_j, x_{ij})$ , pentru  $i, j = 0, \dots, n - 1$

### Exemplul 3

găsiți Transformata Cosinus Discretă 2D a datelor din Figura 4(a)



(a)



(b)

**Figura 4:** Date bi-dimensionale pentru Exemplul 3. (a) Cele 16 puncte  $(i, j, x_{ij})$ . (b) Valorile aproximării de tip cele mai mici pătrate (14) în punctele grilei.

- din definiție și din (6), TCD-2D este matricea

$$\begin{aligned}
 Y = CXC^T &= \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix} \\
 &= \begin{bmatrix} 3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{11}
 \end{aligned}$$

- inversa TCD-2D este ușor de exprimat în funcție de matricea TCD  $C$
- deoarece  $Y = CXC^T$  și  $C$  este ortogonală,  $X$  este recuperată în forma  $X = C^T Y C$

### Definiția 3

**Transformata Cosinus Discretă bi-dimensională inversă** a matricii  $n \times n$   $Y$  este matricea  $X = C^T Y C$ .

- după cum am văzut, există o legătură strânsă între inversarea unei transformări ortogonale (cum ar fi TCD-2D) și interpolare
- scopul interpolării este de a recupera datele initiale din funcții care sunt construite folosind coeficienții de interpolare dați de către transformare
- deoarece  $C$  este o matrice ortogonală,  $C^{-1} = C^T$
- inversarea TCD-2D poate fi scrisă ca un fapt despre interpolare,  $X = C^T Y C$ , deoarece în această ecuație, valorile  $x_{ij}$  sunt exprimate în termeni de produse de cosinuși
- pentru a scrie o expresie folositoare pentru funcția de interpolare, ne reamintim definiția lui  $C$  din (1),

$$C_{ij} = \sqrt{\frac{2}{n}} a_i \cos \frac{i(2j+1)\pi}{2n}, \tag{12}$$

pentru  $i, j = 0, \dots, n-1$ , unde

$$a_i \equiv \begin{cases} 1/\sqrt{2} & \text{dacă } i = 0 \\ 1 & \text{dacă } i = 1, \dots, n-1. \end{cases}$$

- potrivit regulilor înmulțirii matricilor, ecuația  $X = C^T Y C$  se traduce în

$$\begin{aligned} x_{ij} &= \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} C_{ik}^T y_{kl} C_{lj} \\ &= \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} C_{ki} y_{kl} C_{lj} \\ &= \frac{2}{n} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} y_{kl} a_k a_l \cos \frac{k(2i+1)\pi}{2n} \cos \frac{l(2j+1)\pi}{2n}. \end{aligned} \quad (13)$$

- aceasta este exact formularea interpolării pe care o căutăm

### Teorema 3 (Teorema de interpolare a TCD-2D)

Fie  $X = (x_{ij})$  o matrice de  $n^2$  numere reale. Fie  $Y = (y_{kl})$  Transformata Cosinus Discretă bi-dimensională a lui  $X$ . Definim  $a_0 = 1/\sqrt{2}$  și  $a_k = 1$  pentru  $k > 0$ . Atunci funcția reală

$$P_n(s, t) = \frac{2}{n} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} y_{kl} a_k a_l \cos \frac{k(2s+1)\pi}{2n} \cos \frac{l(2t+1)\pi}{2n}$$

satisfacă  $P_n(i, j) = x_{ij}$ , pentru  $i, j = 0, \dots, n-1$ .

- întorcându-ne la Exemplul 3, singurii coeficienți de interpolare nenuli sunt  $y_{00} = 3$ ,  $y_{02} = y_{20} = 1$ , și  $y_{22} = -1$
- scriind funcția de interpolare din Teorema 3, obținem

$$\begin{aligned} P_4(s, t) &= \frac{2}{4} \left[ \frac{1}{2} y_{00} + \frac{1}{\sqrt{2}} y_{02} \cos \frac{2(2t+1)\pi}{8} + \frac{1}{\sqrt{2}} y_{20} \cos \frac{2(2s+1)\pi}{8} \right. \\ &\quad \left. + y_{22} \cos \frac{2(2s+1)\pi}{8} \cos \frac{2(2t+1)\pi}{8} \right] \\ &= \frac{1}{2} \left[ \frac{1}{2}(3) + \frac{1}{\sqrt{2}}(1) \cos \frac{2(2t+1)\pi}{8} + \frac{1}{\sqrt{2}}(1) \cos \frac{2(2s+1)\pi}{8} \right. \\ &\quad \left. + (-1) \cos \frac{2(2s+1)\pi}{8} \cos \frac{2(2t+1)\pi}{8} \right] \\ &= \frac{3}{4} + \frac{1}{2\sqrt{2}} \cos \frac{(2t+1)\pi}{4} + \frac{1}{2\sqrt{2}} \cos \frac{(2s+1)\pi}{4} \\ &\quad - \frac{1}{2} \cos \frac{(2s+1)\pi}{4} \cos \frac{(2t+1)\pi}{4}. \end{aligned}$$

- de exemplu, implementarea unui filtru trece-jos va însemna pur și simplu renunțarea la componente cu frecvențe înalte, cele ale căror coeficienți au indicei cei mai mari, din funcția de interpolare
- în Exemplul 3, cea mai bună interpolare de tip cele mai mici pătrate pentru funcțiile de bază

$$\cos \frac{i(2s+1)\pi}{8} \cos \frac{j(2t+1)\pi}{8},$$

pentru  $i+j \leq 2$ , este dată prin renunțarea la toți termenii care nu satisfac  $i+j \leq 2$

- în acest caz, singurul termen nenul de frecvență înaltă este termenul cu  $i=j=2$ , ceea ce dă

$$P_2(s, t) = \frac{3}{4} + \frac{1}{2\sqrt{2}} \cos \frac{(2t+1)\pi}{4} + \frac{1}{2\sqrt{2}} \cos \frac{(2s+1)\pi}{4}. \quad (14)$$

- această aproximare de tip cele mai mici pătrate este prezentată în Figura 4(b)

## 9.2.2. Compresia imaginilor

- conceptul de ortogonalitate, după cum este reprezentat în Transformata Cosinus Discretă, este crucial pentru realizarea compresiei imaginilor
- imaginile constau din pixeli, fiecare reprezentat printr-un număr (sau trei numere, pentru imagini color)
- modul convenabil în care metode cum ar fi TCD pot efectua aproximarea de tip cele mai mici pătrate, face ușoră reducerea numărului de biți necesari pentru a reprezenta valorile pixelilor, degradând doar puțin imaginea, probabil imperceptibil pentru ochiul uman
- Figura 5(a) prezintă o redare în tonuri de gri a unei matrice de pixeli de dimensiune  $256 \times 256$
- tonul de gri al fiecărui pixel este reprezentat printr-un octet, adică un sir de 8 biți reprezentând tonurile de la 0 = 00000000 (negru) la 255 = 11111111 (alb)
- putem să ne gândim la informația prezentată în figură ca la o matrice de dimensiune  $256 \times 256$  de numere întregi
- reprezentată în acest fel, imaginea conține  $(256)^2 = 2^{16} = 64K$  octeți de informație

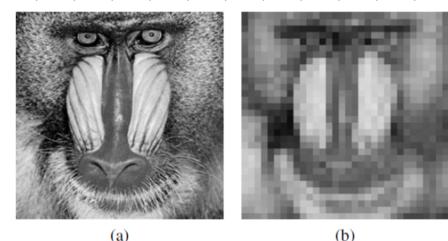
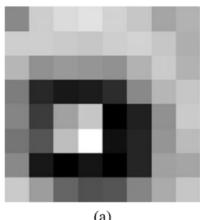


Figura 5: Imagine în tonuri de gri. (a) Fiecare pixel din grila  $256 \times 256$  este reprezentat printr-un întreg între 0 și 255. (b) Compresia brută—fiecare bloc de pixeli de dimensiune  $8 \times 8$  este înlocuit cu valoarea sa medie în tonuri de gri.

Figura 5(b) prezintă o metodă brută de compresie, în care fiecare bloc de pixeli de dimensiune  $8 \times 8$  este înlocuit cu valoarea medie a pixelilor din acel bloc

- cantitatea de date compresate este considerabilă—există doar  $(32)^2 = 2^{10}$  blocuri, fiecare reprezentat acum de un singur număr întreg—dar calitatea imaginii rezultată este proastă
- scopul nostru este să facem compresia mai puțin brută, prin înlocuirea fiecărui bloc de dimensiune  $8 \times 8$  cu câteva numere întregi care transmit mai bine informația imaginii inițiale



(a)

110	168	178	182	170	159	134	145
166	168	164	161	165	171	159	141
146	118	124	122	119	145	162	144
102	34	22	25	38	111	146	159
107	49	130	159	2	29	117	164
95	71	153	207	15	30	122	150
112	21	0	19	0	30	132	136
163	129	83	67	69	107	139	159

(b)

-18	40	48	54	42	31	6	17
38	40	36	33	37	43	31	13
18	-10	-4	-6	-9	17	34	16
-26	-84	-106	-103	-90	-17	18	31
-21	-79	2	31	-126	-99	-11	36
-33	-57	25	79	-113	-98	-6	22
-16	107	-128	-109	-128	-98	4	7
35	1	-45	-61	-59	-21	11	31

(c)

Figura 6: Exemplu de bloc de dimensiune  $8 \times 8$ . (a) Redare în tonuri de gri (b) Valorile pixelilor în tonuri de gri (c) Valorile pixelilor minus 128.

- și ne vom baza pe capacitatea TCD-2D de a sorta informația potrivit importanței ei pentru sistemul vizual uman
- calculăm TCD-2D a lui  $X$  ca fiind

$$Y = C_8 X C_8^T = \begin{bmatrix} -121 & -66 & 127 & -65 & 27 & 98 & 7 & -25 \\ 200 & 22 & -124 & 34 & -36 & -62 & 5 & 6 \\ 113 & 43 & -32 & 55 & -25 & -75 & -21 & 12 \\ -10 & 35 & -69 & -131 & 28 & 54 & -4 & -24 \\ -14 & -18 & 16 & 1 & -5 & -27 & 14 & -6 \\ -124 & -74 & 47 & 60 & -1 & -16 & -8 & 13 \\ 81 & 35 & -57 & -54 & -7 & 6 & 1 & -16 \\ -16 & 11 & 5 & -15 & 11 & 12 & -1 & 9 \end{bmatrix}, \quad (16)$$

după rotunjirea la cel mai apropiat întreg, pentru simplificare

- această rotunjire adaugă o mică eroare suplimentară și nu este strict necesară, dar va ajuta la compresie
- observăm că, datorită amplitudinilor mai mari, există tendință ca mai multe informații să fie stocate în partea stângă sus a matricii de transformare  $Y$ , prin comparație cu partea dreaptă jos
- partea dreaptă jos reprezintă funcțiile de bază cu frecvență spațială relativ mare prin luarea tuturor  $y_{kl} = 0$ , pentru  $k + l \geq 7$  (reamintim că intrările unei matrice sunt numerotate astfel încât  $0 \leq k, l \leq 7$ )
- după filtrarea trece-jos, coeficienții transformatei sunt

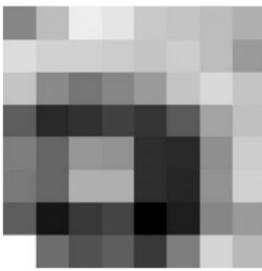
$$X = \begin{bmatrix} -18 & 40 & 48 & 54 & 42 & 31 & 6 & 17 \\ 38 & 40 & 36 & 33 & 37 & 43 & 31 & 13 \\ 18 & -10 & -4 & -6 & -9 & 17 & 34 & 16 \\ -26 & -94 & -106 & -103 & -90 & -17 & 18 & 31 \\ -21 & -79 & 2 & 31 & -126 & -99 & -11 & 36 \\ -33 & -57 & 25 & 79 & -113 & -98 & -6 & 22 \\ -16 & 107 & -128 & -109 & -128 & -98 & 4 & 7 \\ 35 & 1 & -45 & -61 & -59 & -21 & 11 & 31 \end{bmatrix}, \quad (15)$$

- totuși, datorită faptului că TCD-2D este o transformare inversabilă, informația din  $Y$  poate fi folosită pentru a reconstrui complet imaginea inițială, cu excepția erorilor de rotunjire

- prima strategie de compresie pe care o vom încerca este o formă a filtrării trece-jos
- după cum am discutat în subsecțiunea anterioară, aproximarea de tip cele mai mici pătrate folosind TCD-2D se reduce la renunțarea la anumiti termeni din funcția de interpolare  $P_8(s, t)$
- de exemplu, putem renunța la contribuția funcțiilor cu frecvență spațială relativ mare prin luarea tuturor  $y_{kl} = 0$ , pentru  $k + l \geq 7$  (reamintim că intrările unei matrice sunt numerotate astfel încât  $0 \leq k, l \leq 7$ )
- după filtrarea trece-jos, coeficienții transformatei sunt

$$Y_{\text{jos}} = \begin{bmatrix} -121 & -66 & 127 & -65 & 27 & 98 & 7 & 0 \\ 200 & 22 & -124 & 34 & -36 & -62 & 0 & 0 \\ 113 & 43 & -32 & 55 & -25 & 0 & 0 & 0 \\ -10 & 35 & -69 & -131 & 0 & 0 & 0 & 0 \\ -14 & -18 & 16 & 1 & 0 & 0 & 0 & 0 \\ -124 & -74 & 0 & 0 & 0 & 0 & 0 & 0 \\ 81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (17)$$

- pentru a reconstrui imaginea, aplicăm TCD-2D inversă sub forma  $C_8^T Y_{\text{jos}} C_8$  și obținem valorile pixelilor în tonuri de gri prezentate în Figura 7
- imaginea din partea (a) este asemănătoare cu cea inițială din Figura 6(a), dar diferă ca nivel de detaliu
- cât de mult am compresat informația din blocul de dimensiune  $8 \times 8$ ?
- imaginea inițială poate fi reconstruită (fără pierdere de informație, cu excepția rotunjirii la numere întregi) prin transformarea inversă a TCD-2D (16) și prin adunarea înapoi a lui 128
- făcând filtrarea trece-jos cu matricea (17), am redus cerințele de spațiu de stocare aproximativ la jumătate, reținând în același timp mare parte din aspectele vizuale calitative ale blocului



(a)

109	151	191	185	162	158	152	141
177	169	170	165	159	164	152	127
160	113	98	110	126	158	174	160
78	34	41	55	43	75	133	156
103	83	123	119	39	35	115	164
100	84	143	141	39	31	120	167
77	18	48	59	-3	26	111	126
206	89	68	76	47	103	173	150

(b)

-19	23	63	57	34	30	24	13
49	41	42	37	31	36	24	-1
32	-15	-30	-18	-2	30	46	32
-50	-94	-87	-73	-85	-53	5	28
-25	-45	-5	-9	-89	-93	-13	36
-28	-44	15	13	-89	-97	-8	39
-51	-110	-80	-69	-131	-102	-17	-2
78	-39	-60	-52	-81	-25	45	22

(c)

Figura 7: Rezultatul unui filtru trece-jos. (a) Imaginea filtrată (b) Valorile pixelilor în tonuri de gri, după transformare și adunând 128 (c) Datele invers transformate.

## 9.2.3. Cuantizarea

- ideea cuantizării va permite ca efectele filtrării trece-jos să fie obținute într-un mod mai selectiv
- în locul ignorării complete a coeficientilor, vom reține versiuni cu acuratețe redusă ale anumitor coeficienți la un cost de stocare mai mic
- această idee exploatează aceleași aspecte ale sistemului vizual uman—și anume că acesta este mai puțin sensibil la frecvențe spațiale mai înalte
- ideea de bază este de a atribui mai puțini biți pentru a stoca informații despre colțul din dreapta jos al matricii de transformare  $Y$ , în locul renunțării complete la acesta

### Algoritmul 1 (Cuantizarea modulo $q$ )

Cuantizarea:

$$z = \text{rotunjire} \left( \frac{y}{q} \right)$$

Decuantizarea:

$$\bar{y} = qz \quad (18)$$

- aici, „rotunjire” înseamnă „rotunjire la cel mai apropiat întreg”
- eroarea de cuantizare este diferența dintre intrarea  $y$  și ieșirea  $\bar{y}$  după cuantizare și decuantizare
- eroarea maximă a cuantizării modulo  $q$  este  $q/2$

### Exemplu 4

- cuantizați numerele  $-10, 3, \text{ și } 65$  modulo 8
- valorile cuantizate sunt  $-1, 0, \text{ și } 8$
- la decuantizare, rezultatele sunt  $-8, 0, \text{ și } 64$
- erorile sunt  $| -2 |, | 3 |, \text{ și, respectiv, } | 1 |$ , fiecare mai mică decât  $q/2 = 4$

$\rightarrow$  eroare = huit

- întorcându-ne la exemplul cu imaginea, numărul de biți permisi pentru fiecare frecvență poate fi ales arbitrar
- fie  $Q$  o matrice  $8 \times 8$  numită **matricea de cuantizare**
- intrările  $q_{kl}, 0 \leq k, l \leq 7$  ne vor spune câți biți trebuie să atribuim fiecărei intrări a matricii de transformare  $Y$
- înlocuim  $Y$  prin matricea compresată

$$Y_Q = \left[ \text{rotunjire} \left( \frac{y_{kl}}{q_{kl}} \right) \right], \quad 0 \leq k, l \leq 7. \quad (19)$$

- matricea  $Y$  este împărțită element cu element prin matricea de cuantizare
- rotunjirea care urmează este punctul în care are loc pierderea de informație, făcând această metodă o formă a compresiei cu pierdere de informații
- observăm că, cu cât este mai mare o intrare a lui  $Q$ , cu atât se pierde mai multă informație datorită cuantizării
- ca un prim exemplu, **cuantizarea liniară** este definită prin matricea

$$q_{kl} = 8p(k + l + 1) \text{ pentru } 0 \leq k, l \leq 7, \quad (20)$$

- pentru o anumită constantă  $p$ , numită **parametrul de pierdere**
- prin urmare,

$$Q = p \begin{bmatrix} 8 & 16 & 24 & 32 & 40 & 48 & 56 & 64 \\ 16 & 24 & 32 & 40 & 48 & 56 & 64 & 72 \\ 24 & 32 & 40 & 48 & 56 & 64 & 72 & 80 \\ 32 & 40 & 48 & 56 & 64 & 72 & 80 & 88 \\ 40 & 48 & 56 & 64 & 72 & 80 & 88 & 96 \\ 48 & 56 & 64 & 72 & 80 & 88 & 96 & 104 \\ 56 & 64 & 72 & 80 & 88 & 96 & 104 & 112 \\ 64 & 72 & 80 & 88 & 96 & 104 & 112 & 120 \end{bmatrix}.$$

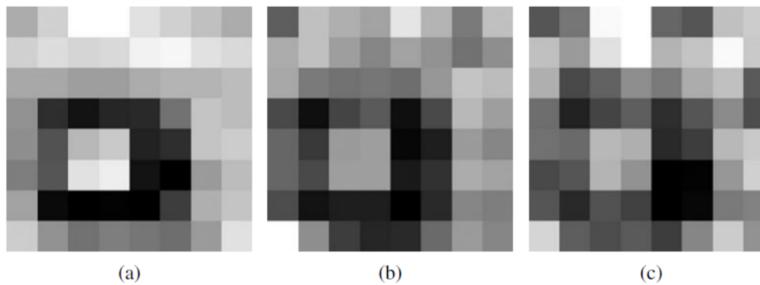
- parametrul de pierdere  $p$  poate fi modificat pentru a transforma biți în acuratețe vizuală

- cu cât este mai mic parametrul de pierdere, cu atât mai bună va fi reconstrucția
- mulțimea de numere rezultată din matricea  $Y_Q$  reprezintă noua versiune cuantizată a imaginii
- pentru a decompresa fișierul, matricea  $Y_Q$  este decuantizată prin inversarea procesului, care este o înmulțire element cu element cu matricea  $Q$
- aceasta este partea în care se pierde informație în codificarea imaginilor
- înlocuind intrările  $y_{kl}$  prin împărțirea lor la  $q_{kl}$  și rotunjire, și apoi reconstruind prin înmulțire cu  $q_{kl}$ , are o eroare potențială adăugată de  $q_{kl}/2$  pentru  $y_{kl}$
- aceasta este eroarea de cuantizare
- cu cât este mai mare  $q_{kl}$ , cu atât este mai mare eroarea potențială de reconstrucție a imaginii
- pe de altă parte, cu cât este mai mare  $q_{kl}$ , cu atât sunt mai mici intrările întregi ale lui  $Y_Q$ , și mai puțini biți sunt necesari pentru a le stoca
- acesta este compromisul între acuratețea imaginii și dimensiunea fisierului

- de fapt, cuantizarea reușește două lucruri: multe contribuții mici ale frecvențelor înalte sunt imediat puse pe zero de (19), și contribuții care rămân diferite de zero sunt reduse în dimensiune, astfel încât să poată fi transmise sau stocate folosind mai puțini biți
- mulțimea de numere rezultată este convertită într-un flux de biți folosind codificarea Huffman, discutată în secțiunea următoare
- când cuantizarea liniară este aplicată pentru (16) cu  $p = 1$ , coeficienții rezultați sunt

$$Y_Q = \begin{bmatrix} -15 & -4 & 5 & -2 & 1 & 2 & 0 & 0 \\ 13 & 1 & -4 & 1 & -1 & -1 & 0 & 0 \\ 5 & 1 & -1 & 1 & 0 & -1 & 0 & 0 \\ 1 & -1 & -2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (21)$$

- blocul de imagine reconstruit, format prin decuantizarea și transformarea inversă a lui  $Y_Q$ , este prezentat în Figura 8(a)
- mici diferențe pot fi observate în comparație cu blocul inițial, dar este mai precisă decât reconstrucția din cazul filtrării trece-îos



**Figura 8: Rezultatul cuantizării liniare.** Parametrul de pierdere este (a)  $p = 1$  (b)  $p = 2$  (c)  $p = 4$ .

- după cuantizarea liniară cu  $p = 2$ , coeficienții transformării cuantizate sunt

$$Y_Q = \begin{bmatrix} -8 & -2 & 3 & -1 & 0 & 1 & 0 & 0 \\ 6 & 0 & -2 & 0 & 0 & -1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (22)$$

iar după o cuantizare liniară cu  $p = 4$ , coeficienții transformării cuantizate sunt

$$Y_Q = \begin{bmatrix} -4 & -1 & 1 & -1 & 0 & 1 & 0 & 0 \\ 3 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (23)$$

- Figura 8 prezintă rezultatul cuantizării liniare pentru trei valori diferite ale parametrului de pierdere  $p$

- observăm că, cu cât este mai mare valoarea parametrului de pierdere  $p$ , cu atât mai multe intrări ale matricii  $Y_Q$  sunt făcute zero de către procedura de cuantizare, cu atât sunt mai mici cerințele de spațiu de stocare pentru reprezentarea pixelilor, și cu atât mai puțin precisă este reconstrucția imaginii inițiale
- în continuare, vom cuantiza toate cele  $32 \times 32 = 1024$  blocuri ale imaginii din Figura 5
- și anume, vom efectua 1024 de versiuni independente ale exemplului precedent
- rezultatele pentru parametrul de pierdere  $p = 1, 2$ , și 4 sunt prezentate în Figura 9
- imaginea a început să se deterioreze semnificativ pentru  $p = 4$
- putem face un calcul aproximativ pentru a găsi cât de mult a fost compresată imaginea datorită cuantizării
- imaginea inițială folosește o valoare a pixelilor de la 0 la 255, ceea ce reprezintă un octet, sau 8 biți
- pentru fiecare bloc  $8 \times 8$ , numărul total de biți necesari fără compresie este de  $8(8)^2 = 512$  biți
- acum, să presupunem că folosim cuantizarea liniară cu parametrul de pierdere  $p = 1$

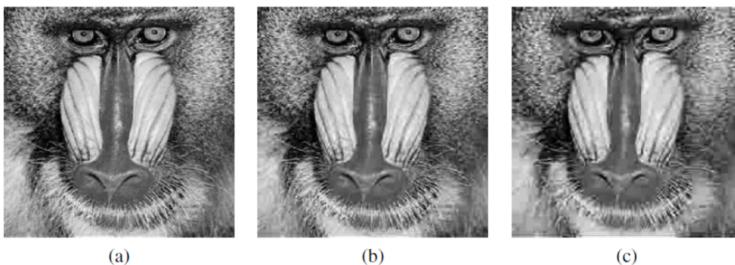


Figura 9: Rezultatul cuantizării liniare a tuturor celor 1024 de blocuri de dimensiune  $8 \times 8$ . Parametrii de pierdere sunt (a)  $p = 1$  (b)  $p = 2$  (c)  $p = 4$ .

- presupunem că valoarea maximă a unei intrări a matricii de transformare  $Y$  este 255

- atunci cele mai mari valori posibile pentru intrările lui  $Y_Q$ , după cuantizarea prin  $Q$ , sunt

$$\begin{bmatrix} 32 & 16 & 11 & 8 & 6 & 5 & 5 & 4 \\ 16 & 11 & 8 & 6 & 5 & 5 & 4 & 4 \\ 11 & 8 & 6 & 5 & 5 & 4 & 4 & 3 \\ 8 & 6 & 5 & 5 & 4 & 4 & 3 & 3 \\ 6 & 5 & 5 & 4 & 4 & 3 & 3 & 3 \\ 5 & 5 & 4 & 4 & 3 & 3 & 3 & 2 \\ 5 & 4 & 4 & 3 & 3 & 3 & 2 & 2 \\ 4 & 4 & 3 & 3 & 2 & 2 & 2 & 2 \end{bmatrix}$$

- deoarece atât intrări pozitive cât și negative sunt posibile, numărul de biți necesari pentru a stoca fiecare intrare este

$$\begin{bmatrix} 7 & 6 & 5 & 5 & 4 & 4 & 4 & 4 \\ 6 & 5 & 5 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 4 & 4 & 4 & 4 & 4 & 3 \\ 5 & 4 & 4 & 4 & 4 & 4 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 3 & 3 & 3 & 3 & 3 & 3 \end{bmatrix}$$

- cuantizarea liniară cu  $p = 1$  este apropiată de cantizarea JPEG standard
- matricea de cantizare care oferă cea mai bună compresie cu cea mai mică degradare a imaginii a fost subiectul multor cercetări și discuții
- standardul JPEG include un apendix numit „Anexa K: Exemple și Instrucțiuni” care conține o matrice  $Q$  bazată pe experimente făcute cu sistemul vizual uman
- matricea

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

este foarte folosită de codificatoarele JPEG distribuite în prezent

- luând parametrul de pierdere  $p = 1$  ar trebui să dea o reconstrucție practic perfectă din punctul de vedere al sistemului vizual uman, în timp ce  $p = 4$  introduce de obicei defecte vizibile
- într-o anumită măsură, calitatea vizuală depinde de valoarea pixelului: dacă pixelii sunt mici, anumite erori ar putea trece neobservate
- până acum, am vorbit doar despre imagini în tonuri de gri
- este destul de ușor să extindem aplicația la imagini color, care pot fi exprimate în sistemul color RGB
- fiecarui pixel îi sunt atribuți trei întregi, câte unul pentru intensitățile roșu, verde, și albastru
- o abordare pentru a realiza compresia acestor imagini este de a repeta procesarea precedentă independent pentru fiecare dintre cele trei culori, tratând-o pe fiecare ca și cum ar fi în tonuri de gri, iar apoi de a reconstituia imaginea din cele trei culori la sfârșit
- deși standardul JPEG nu menționează cum trebuie tratată culoarea, metoda numită Baseline JPEG folosește o abordare mai delicată
- definim **luminanță**  $Y = 0.299R + 0.587G + 0.114B$  și diferențele de culoare  $U = B - Y$  și  $V = R - Y$
- aceasta transformă datele de culoare RGB în sistemul YUV
- aceasta este o transformare complet reversibilă, deoarece valorile RGB pot fi găsite folosind formulele:  $B = U + Y$ ,  $R = V + Y$ , și  $G = (Y - 0.299R - 0.114B)/(0.587)$

- suma acestor 64 de numere este 249, sau  $249/64 \approx 3.89$  biți/pixel, care este mai puțin de jumătate din numărul de biți (512, sau 8 biți/pixel) necesari pentru a stoca valorile inițiale ale pixelilor din matricea de imagine de dimensiune  $8 \times 8$
- statisticile corespunzătoare pentru alte valori ale lui  $p$  sunt prezentate în următorul tabel:

$p$	total biți	biți/pixel
1	249	3.89
2	191	2.98
4	147	2.30

- după cum se vede în tabel, numărul de biți necesari pentru a reprezenta imaginea este redus printre-un factor de 2 când  $p = 1$ , cu puține schimbări vizibile în imagine
- această compresie este datorată cuantizării
- pentru a compresa și mai mult, putem profita de faptul că mulți dintre termenii de frecvențe înalte din transformată sunt zero după cuantizare
- acest lucru se face cel mai eficient prin folosirea codificărilor Huffman și run-length, introduse în secțiunea următoare

- metoda Baseline JPEG aplică filtrarea TCD discutată anterior, independent pentru  $Y$ ,  $U$ , și  $V$ , folosind matricea de cuantizare  $Q_Y$  din Anexa K pentru variabila de luminanță  $Y$  și matricea de cuantizare

$$Q_C = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix} \quad (25)$$

pentru diferențele de culoare  $U$  și  $V$

- după reconstruirea lui  $Y$ ,  $U$ , și  $V$ , ele sunt puse înapoi împreună și transformate înapoi în sistemul RGB pentru a reconstituи imaginea
- datorită rolurilor mai puțin importante ale lui  $U$  și  $V$  în sistemul vizual uman, o cuantizare mai agresivă este permisă pentru ele, după cum se poate vedea în (25)
- o compresie mai bună poate fi dedusă dintr-un sir de trucuri ad-hoc adiționale—de exemplu, prin medierea diferențelor de culoare și tratarea lor pe o grilă cu mai puține puncte

## 9.3. Codificarea Huffman

### 9.3.1. Teoria informației și codificarea

- considerăm un mesaj care constă dintr-un sir de simboluri
- simbolurile sunt arbitrale; să presupunem că ele aparțin unei mulțimi finite
- în această subsecțiune, vom discuta modalități eficiente de a codifica un astfel de sir folosind cifre binare, sau biți
- cu cât este mai scurt sirul de biți, cu atât este mai ușor și mai ieftin să stocăm sau să transmitem mesajul

#### Exemplu 5

- codificați mesajul ABAACDAB sub forma unui sir binar
- deoarece sunt patru simboluri, o codificare binară convenabilă ar putea să asocieze doi biți fiecărei litere
- de exemplu, am putea să alegem corespondența

A	00
B	01
C	10
D	11

- atunci, mesajul va fi codificat în forma

(00)(01)(00)(00)(10)(11)(00)(01).

- cu acest cod, un total de 16 biți sunt necesari pentru a stoca sau a transmite acest mesaj
- există însă și metode de codificare mai eficiente
- pentru a le înțelege, va trebui mai întâi să introducem ideea de informație
- presupunem că există  $k$  simboluri diferite, și notăm cu  $p_i$  probabilitatea apariției simbolului  $i$  în orice punct din sir
- probabilitatea ar putea fi cunoscută de la început, sau ar putea fi estimată empiric prin împărțirea numărului de apariții ale simbolului  $i$  în sir la lungimea sirului

- compresia cu pierdere de informații pentru imagini necesită un compromis între acuratețe și dimensiunea fișierului
- dacă reducerile în acuratețe sunt suficient de mici pentru a nu fi observate pentru scopul propus al imaginii, acest compromis s-ar putea să merită să fie făcut
- pierderea acurateții are loc în pasul de cuantizare, după transformarea care separă imaginea în frecvențele ei spațiale
- compresia fără pierdere de informații se referă la compresia în continuare, care ar putea fi aplicată fără a mai pierde din acuratețe, pur și simplu datorită unei codificări eficiente a imaginii transformate folosind TCD și cuantizate
- în această secțiune, vom discuta compresia fără pierdere de informații
- ca o aplicație relevantă, există metode simple și eficiente pentru transformarea matricii transformate folosind TCD și cuantizate din secțiunea anterioară într-un flux de biți JPEG
- pentru a afla cum să facem acest lucru, va trebui să facem un scurt tur al bazelor teoriei informației

#### Definiția 4

**Informația Shannon**, sau **entropia Shannon** a unui șir este

$$I = - \sum_{i=1}^k p_i \log_2 p_i.$$

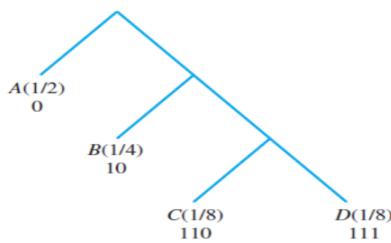
- definiția își are numele de la C. Shannon de la Bell Laboratories, care a făcut cercetări fundamentale despre teoria informației la mijlocul secolului al 20-lea
- informația Shannon a unui șir este considerată o medie a numărului minim de biți per simbol necesari pentru a codifica mesajul
- logica este după cum urmează: în medie, dacă un simbol apare  $p_i$  din timp, atunci ne așteptăm să avem nevoie de  $-\log_2 p_i$  biți pentru a-l reprezenta
- de exemplu, un simbol care apare 1/8 din timp ar putea fi reprezentat de unul dintre simbolurile cu  $-\log_2(1/8) = 3$  biți 000, 001, ..., 111, care sunt în număr de 8
- pentru a găsi media de biți per simbol pentru toate simbolurile, ar trebui să ponderăm numărul de biți per simbol  $i$  cu probabilitatea lui  $p_i$
- aceasta înseamnă că numărul mediu de biți/simbol pentru tot mesajul este suma  $I$  din definiție

#### Exemplul 6

- găsiți informația Shannon a șirului ABAACDAB
- probabilitățile empirice de apariție a simbolurilor A, B, C, D sunt, respectiv,  $p_1 = 4/8 = 2^{-1}$ ,  $p_2 = 2/8 = 2^{-2}$ ,  $p_3 = 1/8 = 2^{-3}$ ,  $p_4 = 2^{-3}$
- informația Shannon este

$$-\sum_{i=1}^4 p_i \log_2 p_i = \frac{1}{2}1 + \frac{1}{4}2 + \frac{1}{8}3 + \frac{1}{8}3 = \frac{7}{4}.$$

- prin urmare, informația Shannon estimează că cel puțin 1.75 biți/simbol sunt necesari pentru a codifica șirul
- deoarece șirul are lungimea 8, numărul total optim de biți ar trebui să fie  $(1.75)(8) = 14$ , și nu 16, cum am codificat șirul mai înainte
- de fapt, mesajul poate fi trimis în cei 14 biți preziși, folosind metoda cunoscută sub numele de **codificarea Huffman**
- scopul este de a atribui un cod binar unic fiecărui simbol, care reflectă probabilitatea apariției simbolului, unde simbolurile mai comune primesc coduri mai scurte
- algoritmul funcționează prin construirea unui arbore din care poate fi citit codul binar
- începem cu două simboluri cu cea mai mică probabilitate, și considerăm simbolul „combinat”, căruia îi atribuim probabilitatea combinată
- cele două simboluri formează o ramificație a arborelui
- apoi repetăm acest pas, combinând simbolurile pentru a construi ramificațiile arborelui, până când există doar un singur grup de simboluri rămas, care corespunde rădăcinii arborelui
- în cazul nostru, am combinat mai întâi simbolurile cele mai puțin probabile C și D într-un simbol CD cu probabilitatea 1/4
- probabilitățile rămase sunt A(1/2), B(1/4), și CD(1/4)
- din nou, combinăm cele mai puțin probabile două simboluri, pentru a obține A(1/2), BCD(1/2)
- în final, combinând cele două simboluri rămase, obținem ABCD(1)
- fiecare combinație formează o ramificație a arborelui Huffman:



- odată ce arborele este complet, codul Huffman pentru fiecare simbol poate fi citit prin traversarea arborelui de sus în jos, scriind un 0 pentru o ramificație la stânga și un 1 pentru o ramificație la dreapta, după cum se arată mai sus
- de exemplu, A este reprezentat prin 0, și C este reprezentat prin două ramificații la dreapta și una la stânga, și anume 110

A - apărare de 4 ori  
 B - apărare de 2 ori  
 C - apărare 3 dată  
 D - apărare 0 dată  
 total : 8 litere

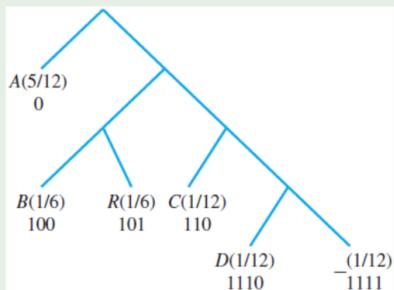
- acum, sirul de litere ABAACDAB poate fi tradus intr-un flux de biti de lungime 14:  
(0)(10)(0)(0)(110)(111)(0)(10).
- informația Shannon a mesajului oferă o limită de jos pentru numărul de biti/simbol al codificării binare
- în acest caz, codul Huffman a realizat limita de informație Shannon de  $14/8 = 1.75$  biti/simbol
- din nefericire, acest lucru nu este întotdeauna posibil, după cum arată următorul exemplu

### Exemplul 7

- găsiți informația Shannon și o codificare Huffman a mesajului ABRA CADABRA
- probabilitățile empirice ale celor șase simboluri sunt

A	5/12
B	2/12
R	2/12
C	1/12
D	1/12
_	1/12

- observăm că spațiul a fost tratat ca un simbol
  - informația Shannon este
- $$-\sum_{i=1}^6 p_i \log_2 p_i = -\frac{5}{12} \log_2 \frac{5}{12} - 2 \frac{1}{6} \log_2 \frac{1}{6} - 3 \frac{1}{12} \log_2 \frac{1}{12} \approx 2.28 \text{ biti/simbol.}$$
- acesta este minimul teoretic pentru media de biti/simbol pentru codificarea mesajului ABRA CADABRA
  - pentru a găsi codificarea Huffman, procedăm după cum deja am descris
  - începem prin a combina simbolurile D și \_, deși oricare două dintre cele trei cu probabilitatea  $1/12$  ar fi putut fi alese pentru ramificația cea mai de jos
  - simbolul A apare ultimul, deoarece are cea mai mare probabilitate
  - o codificare Huffman este prezentată în diagrama de mai jos



- observăm că A are un cod scurt, datorită faptului că este un simbol popular în mesaj
- secvența binară codificată pentru ABRA CADABRA este

(0)(100)(101)(0)(1111)(110)(0)(1110)(0)(100)(101)(0),

care are lungimea de 28 de biti

- media pentru această codificare este de  $28/12 = 2\frac{1}{3}$  biti/simbol, ușor mai mare decât minimul teoretic calculat anterior
- codurile Huffman nu pot întotdeauna să se potrivească exact cu informația Shannon, dar adesea se apropiu foarte mult de aceasta
- secretul unui cod Huffman este următorul: deoarece fiecare simbol apare doar la capătul unei ramificații a arborelui, niciun cod complet al unui simbol nu poate fi începutul unui alt cod de simbol
- prin urmare, nu există nicio ambiguitate când traducem codul înapoi în simboluri

### 9.3.2. Codificarea Huffman pentru formatul JPEG

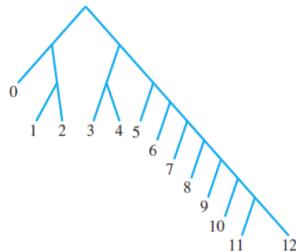
- această subsecțiune este dedicată unui exemplu extins al codificării Huffman în practică
- formatul de compresie a imaginilor JPEG este ubicuu în fotografie digitală modernă
- reprezintă un studiu de caz fascinant datorită juxtapunerii matematicii teoretice și a considerațiilor inginerești
- codificarea binară a coeficientilor transformatei pentru un fișier de imagine JPEG folosește codificarea Huffman în două moduri diferite, unul pentru componenta CD (intrarea  $(0, 0)$  din matricea de transformare) și un altul pentru celelalte 63 de intrări ale matricii  $8 \times 8$ , așa-numitele componente CA (curent alternativ)

#### Definiția 5

Fie  $y$  un întreg. **Dimensiunea** lui  $y$  este definită ca fiind

$$L = \begin{cases} \text{floor}(\log_2 |y|) + 1 & \text{dacă } y \neq 0 \\ 0 & \text{dacă } y = 0. \end{cases}$$

- codificarea Huffman pentru JPEG are trei ingrediente: un arbore Huffman pentru componente CD, un alt arbore Huffman pentru componente CA, și un tabel de identificatori întregi
- prima parte a codificării pentru intrarea  $y = y_{00}$  este codul binar pentru dimensiunea lui  $y$ , din următorul arbore Huffman pentru componente CD, numit **arborele DPCM**, de la Differential Pulse Code Modulation



- din nou, arborele trebuie interpretat prin codificarea unui 0 sau unui 1 când mergem pe o ramificație stângă, respectiv dreaptă
- prima parte este urmată de un sir binar din următorul tabel de identificatori întregi:

$L$	intrare	binar
0	0	--
1	-1, 1	0, 1
2	-3, -2, 2, 3	00, 01, 10, 11
3	-7, -6, -5, -4, 4, 5, 6, 7	000, 001, 010, 011, 100, 101, 110, 111
4	-15, -14, ..., -8, 8, ..., 14, 15	0000, 0001, ..., 0111, 1000, ..., 1110, 1111
5	-31, -30, ..., -16, 16, ..., 30, 31	00000, 00001, ..., 01111, 10000, ..., 11110, 11111
6	-63, -62, ..., -32, 32, ..., 62, 63	000000, 000001, ..., 011111, 100000, ..., 111110, 111111
:	:	:

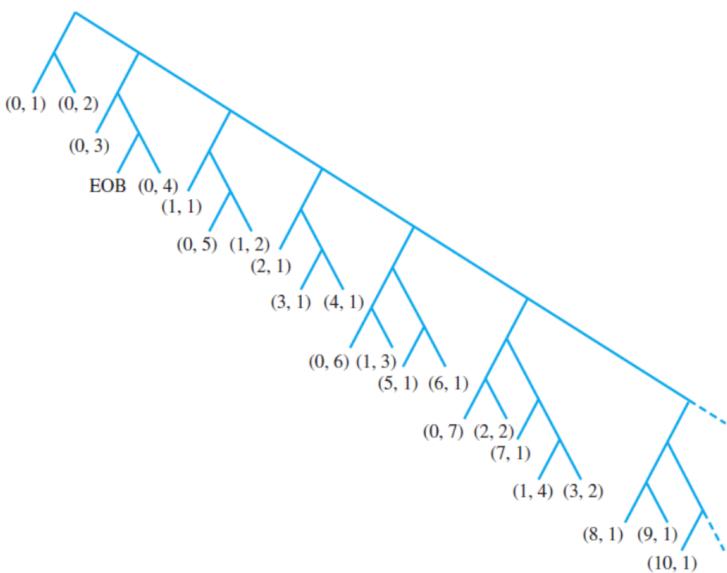
- de exemplu, intrarea  $y_{00} = 13$  va avea dimensiunea  $L = 4$
- potrivit arborelui DPCM, codul Huffman pentru 4 este (101)
- tabelul ne arată că cifrele în plus pentru 13 sunt (1101), deci concatenarea celor două părți, 1011101, va fi stocată pentru componenta CD
- deoarece există adesea corelații între componente CD ale unor blocuri  $8 \times 8$  apropiate, doar diferențele de la bloc la bloc sunt stocate după primul bloc

- diferențele sunt stocate, de la stânga la dreapta, folosind arborele DPCM
- pentru cele 63 de componente CA rămase ale blocului de dimensiune  $8 \times 8$ , **Codificarea Run Length** este folosită ca o modalitate eficientă de a stoca siruri lungi de zerouri
- ordinea convențională pentru stocarea celor 63 de componente este tiparul zigzag

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

(26)

- în loc să codificăm însesă cele 63 de numere, o pereche run-length zero  $(n, L)$  este codificată, unde  $n$  denotă lungimea unui sir de zerouri, și  $L$  reprezintă dimensiunea următoarei intrări care nu este zero
- cele mai comune coduri întâlnite în imagini JPEG tipice, și codificările lor implicate potrivit standardului JPEG, sunt prezentate în arborele Huffman pentru componentele CA



- în fluxul de biți, codul Huffman din arbore (care identifică doar dimensiunea intrării) este urmat imediat de un cod binar care identifică numărul întreg, din tabelul anterior
- de exemplu, secvența de intrări  $-5, 0, 0, 0, 2$  va fi reprezentată sub formă  $(0, 3) -5 (3, 2) 2$ , unde  $(0, 3)$  înseamnă niciun zero, urmat de un număr de dimensiune 3, și  $(3, 2)$  reprezintă 3 zerouri urmate de un număr de dimensiune 2
- din arborele Huffman, găsim că  $(0, 3)$  este codat ca  $(100)$ , și  $(3, 2)$  ca  $(111110111)$
- identifierul pentru  $-5$  este  $(010)$  și pentru  $2$  este  $(10)$ , din tabelul de identifieri întregi
- prin urmare, fluxul de biți folosit pentru a codifica  $-5, 0, 0, 0, 2$  este  $(100)(010)(111110111)(10)$
- arborele Huffman precedent arată doar codurile run-length JPEG care apar cel mai des
- alte coduri folosite sunt  $(11, 1) = 1111111001$ ,  $(12, 1) = 1111111010$ , și  $(13, 1) = 11111111000$

### Exemplul 8

- codificați matricea transformării TCD cuantizate din (23) pentru un fișier de imagine JPEG
- intrarea CD  $y_{00} = -4$  are dimensiunea 3, codificată ca (100) de către arborele DPCM, și, în plus, biții (011) din tabelul de identificatori întregi
- apoi, considerăm sirul de coeficienți CA
- conform cu (26), coeficienții CA sunt ordonați ca  $-1, 3, 1, 0, 1, -1, -1$ , șapte zerouri, 1, patru zerouri,  $-1$ , trei zerouri,  $-1$ , și restul toate zerouri
- codificarea run-length începe cu  $-1$ , care are dimensiunea 1 și astfel contribuie (0, 1) din codul run-length
- următorul număr 3 are dimensiunea 2 și contribuie cu (0, 2)
- perechile run-length zero sunt

$(0, 1) - 1 (0, 2) 3 (0, 1) 1 (1, 1) 1 (0, 1) - 1 (0, 1) - 1$   
 $(7, 1) 1 (4, 1) - 1 (3, 1) - 1$  EOB.

- aici, EOB înseamnă „end-of-block” și se traduce prin faptul că intrările rămase constau din zerouri
- în continuare, citim reprezentările biților din arborele Huffman de mai sus și tabelul de identificatori întregi
- fluxul de biți care stochează blocul de dimensiune  $8 \times 8$  al fotografiei din Figura 8(c) este listat mai jos, în care parantezele sunt incluse doar pentru a spori lizibilitatea:

$(100)(011)$   
 $(00)(0)(01)(11)(00)(1)(1100)(1)(00)(0)(00)(0)$   
 $(11111010)(1)(111011)(0)(111010)(0)(1010).$

- blocul de pixeli din Figura 8(c), care este o aproximare rezonabilă a Figurii 6(a) initiale, este exact reprezentat de acești 54 de biți
- numărul de biți per pixel poate fi calculat ca  $54/64 \approx 0.84$  biți/pixel
- observăm superioritatea acestei codificări față de biți/pixel obținuți de către filtrarea trece-jos și cuantizare
- dat fiind faptul că pixelii au fost inițial întregi pe 8 biți, imaginea  $8 \times 8$  a fost compresată cu mai mult de un factor de 9 : 1

- decompresia unui fișier JPEG constă din inversarea pașilor de compresie
- cititorul JPEG decodează fluxul de biți în simboluri run-length, care formează blocurile transformate TCD de dimensiune  $8 \times 8$  care sunt convertite în final înapoi în blocuri de pixeli folosind TCD inversă