

# Sisteme avansate de codare și compresie a datelor multimedia

## Curs 1 – Tematica disciplinei

Sl.Dr.Ing. Camelia FLOREA

Intelligent and multimodal image processing and analysis group (IMIPA),

Communications Departament, ETTI, TUCN,

E-mail: [Camelia.Florea@com.utcluj.ro](mailto:Camelia.Florea@com.utcluj.ro),

Address: C. Daicoviciu, 15, room 431, Cluj-Napoca, RO.

# SACDMM - Ce se va studia?

- Conceptelor de bază
  - specifice achiziției și modului de **reprezentare a datelor multimedia**.
- Tehnici de baza/algoritmi frecvent utilizați
  - pentru **codare și compresie CU sau FĂRĂ pierderi**
- **Standardele internaționale** pentru compresie
  - imaginilor statice și
  - secvențelor video
- **Aplicații practice** multimedia adaptate fișierelor comprimate

Se dorește înțelegerea atât a **bazelor teoretice**, precum și **aplicabilitatea** lor prin integrarea acestora în sisteme practice.

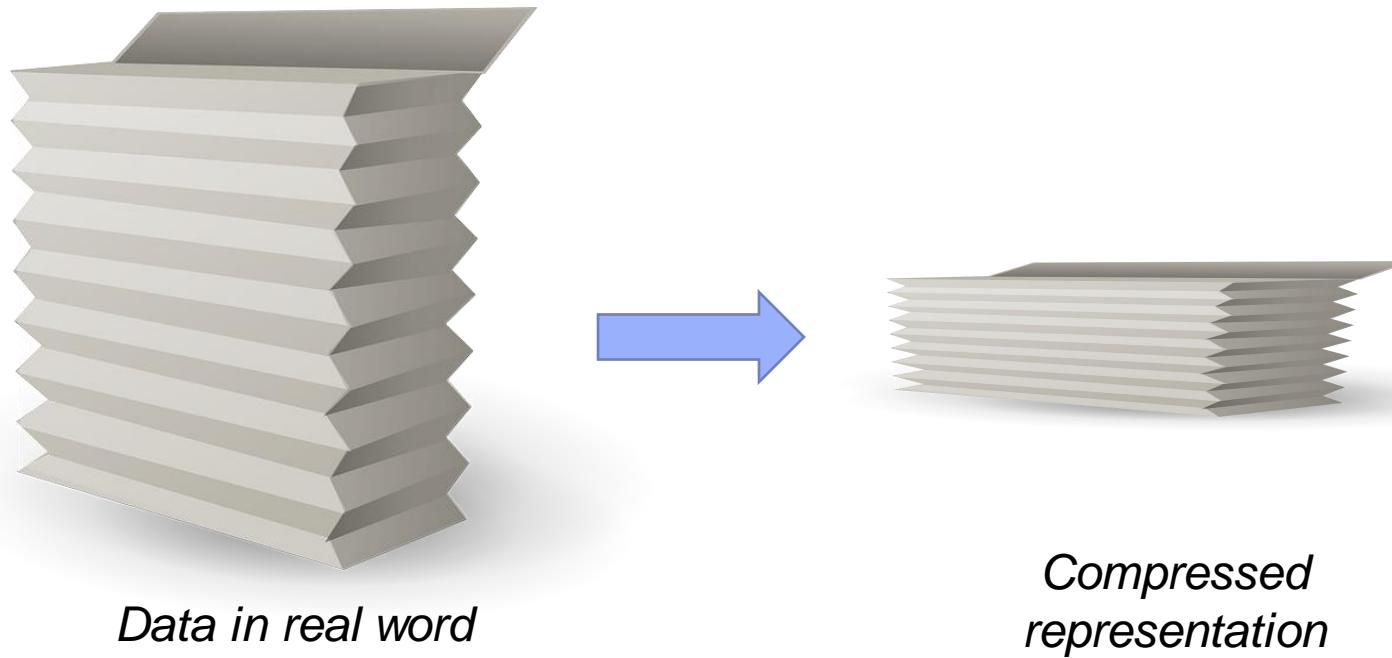
# Reprezentarea digitală a informațiilor

- Reprezentare digitală **compactă** a informației multimedia
  - cu aplicații în special:
    - în **transmisia** și
    - respectiv **stocarea** imaginilor
- Fără reprezentarea comprimată
  - multe dintre aplicațiile multimedia nu ar putea fi implementate practic.

# Data compression

= ***the art and science*** of reducing the amount of data required to represent a scene, object or information

[Gonzalez&Woods2008]



# Cum compresie?

- **Aspecte importante  
(ce urmărim)**

- să ocupe un spațiu redus (fizic sau digital)
  - rata de compresie mare
- la redare să avem obiectul/informația de calitate
  - păstrarea unei calități acceptabile a informației la reconstrucție
- cât de repede dorim reducerea/ redare informație
  - complexitate algoritm/ putere de calcul dispozitiv



***„I have made this letter longer than usual because I lack the time to make it shorter.” [Blaise Pascal]***

# De ce compresie?

---

- Exemplificări – de ce avem nevoie de compresie?

- 
- 
- 
- 
- 
- 
- 
- 
-

# De ce compresie?

- SmartPhone, Camere foto/video, retele de socializare, internet, DVD, etc.
  - suntem înconjurați de foarte multă informație multimedia!
- **De ce** avem nevoie de **compresie**?
  - să considerăm un video SD 720x480, 24 biți/pixel și 30 cadre/sec  
 $720 \times 480 \times 3 \text{ bytes} \times 30 = 31.104.000 \text{ bytes/sec}$

adică pentru un film de 2 ore

aprox. **224 GB**

=> adică 27 DVD dual-layer

pentru a putea stoca pe un DVD

=> o compresie de **1:27 pentru SD.**

**La HD compresia și mai mare!**



# Motivare compresie – camere/ SmartPhone

- Reducere spațiului de stocare
  - stocarea unui număr mare de imagini direct pe dispozitiv
- Reducerea timpului de transmisie/încărcare online
  - tipuri de rețele diferite – timp de transmisie diferit
    - o imagine 1920x1080, 24 biți/pixel se transmite
      - ~ 37 sec pe o rețea dial-up de 56Kbps
      - ~ 5 sec pe Broadband 10Mbps
      - ~ 1 sec pe Broadband 50Mbps
      - la peste 50Mbps - sub o secundă

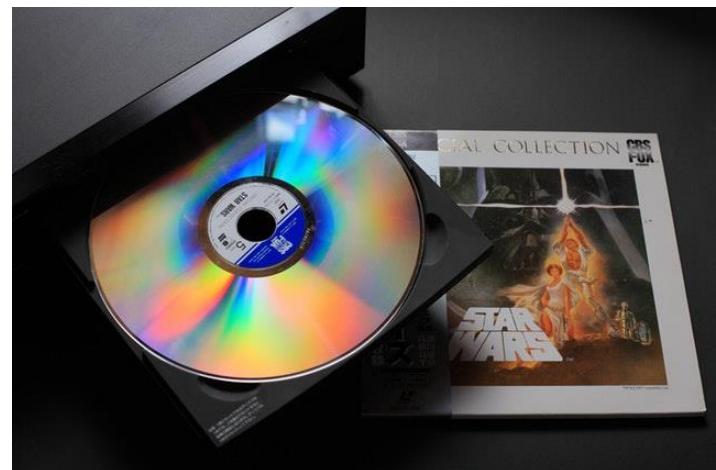


**Foarte rar dorim să transmită/ încărcăm doar o imagine – uzual ~200 imagini!**

- Încărcare poze online – GooglePhotos

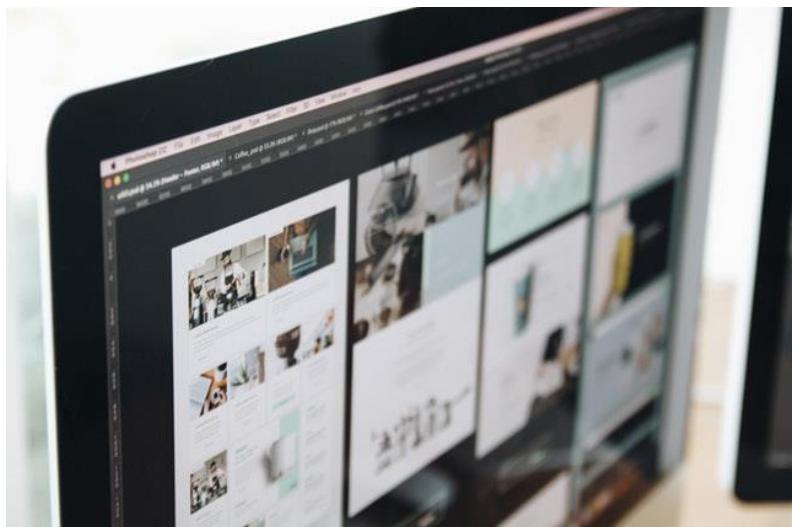
# Stocare pe dispozitive externe

- Timp copiere scriere pe dispozitive externe  
(CD, DVD, Memory Stick, HDD extern, etc)
- Dispozitiv lent: CD-ROM –  $2x = 300 \text{ kB/sec}$ 
  - Exemplu - o secvență video - 720x480 pixeli, 24 biti/pixel, 30 cadre/sec
    - rezulta aprox. 1Mb de memorie pentru fiecare cadru
    - 1 secundă - aproximativ 30MB
  - $\Rightarrow 30\text{MB/sec} \rightarrow 0.29\text{MB/sec} = \text{raport } 103$
- $x52 = 7.8 \text{ MB/sec}$   
 $\Rightarrow$  un raport de compresie de 3.8



# Ex. - Dictionar Enciclopedic Multimedia

- Poate conține:
  - **Text:** 500.000 pagini (2kB / pagina) - **în total ~ 1GB**,
  - **Imagini color:** 3.000 (rezolutie 640x480, 24biti => 0.87MB/imagină) - **total ~ 2.5 GB**,
  - **Hărți:** 500 (640x480x16biti=0,58 MB/harta) - în **total 0,28 GB**,
  - **Sunet stereo:** 60 minute (176kB / sec) - în **total 0,6 GB**,
  - **Animatie:** 30 secvențe cu durata de 2 minute în medie (640 x 320 x 16biti x 16 cadre/sec) - în **total 22 Gb**,
  - **Secvențe video:** 50 cu durata de 1 minut în medie (640 x 480 x 24biti x 25 cadre/sec= 23 MB/s.) - **total 64.3 GB**.



Capacitatea de stocare  
**TOTAL = 90.6 GB**

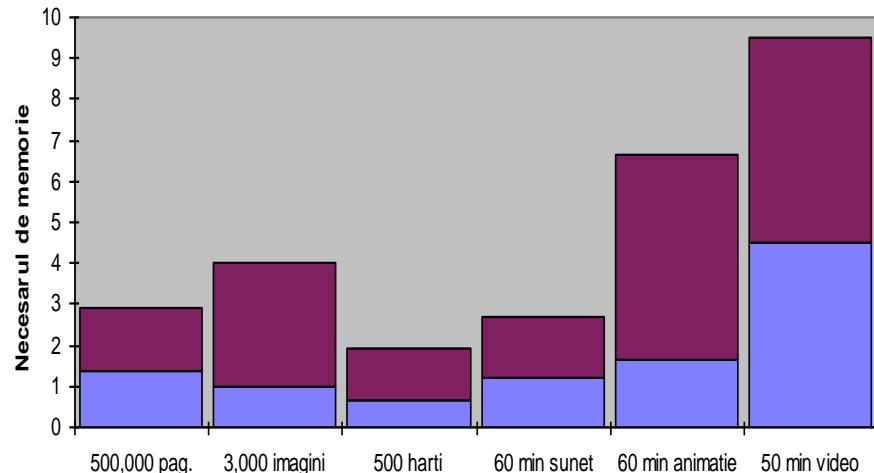
# Ex. - Dictionar Enciclopedic Multimedia

- Consideram ca se aplica compresie

cu **următoarele rate**:

- text 2:1,
- imagini color 15:1,
- harți 10:1,
- sunet stereo 6:1,
- animatie 50:1,
- video 50:1.

90.6 Gb  $\rightarrow$  dupa compresie  $\rightarrow$  2,94



=> Reducere capacitate stocare  
la **2.94 GB !**

# Convinși?

---

- Care este secretul acestei compresii?
- Este un algoritm general valabil?

# Convinși?

- Care este secretul compresiei?
  - eliminarea redundantei
- Este un algoritm general valabil?
  - NU, diferă în funcție de conținut (text, fotografie, schiță, grafică, sunet, video, etc.)



# Tematică curs și laborator

	Course (titles)	Applications (laboratory work)
1	Digital representation of multimedia information. Introduction in data compression. Motivation for data/ image/ video compression.	Digital data representations.
2	Fundamentals in information, entropy and redundancy theory. Performance measures. Image/video formats, containers and compression standards.	Image formats. Performance measures. Project assignment.
3	Lossless and lossy coding techniques (Basic techniques, statistical methods, dictionary methods)	Lossless and lossy predictive coding. PCM, DPCM, Delta modulation, JPEG-LS
4	Lossless compression. Binary image compression, JPEG-LS	
5	Block transform coding, DCT transform	Transform coding, JPEG standard
6	JPEG image compression standard	
7	Sub-band coding, wavelet	Sub-band coding, wavelet, JPEG 2000
8	JPEG 2000 image compression standard	
9	Motion estimation and compensation coding	Motion estimation and compensation coding, MPEG standard
10	MPEG video compression standard	
11	H.26x video compression standard	Compressed domain image and video manipulation/processing.
12	Compressed domain processing of digital images and videos	H.26x in real-time systems
13	Applications for H.26x standards. Windows Media Video (WMV) Standard.	Final evaluation, make-up missed lab sessions
14	Stereo Image Compression	

# Lucrări practice / Proiecte

---

- Lucrări practice
  - VC Demo/ MatLab/ Visual C#/
- Listă Proiecte (studiu si testare Demo)!

# Bibliografie curs

- **Suport online curs**
  - Teams o365 [20\_21]\_TM\_SACCDMM (prezentări curs, lucrări de laborator, probleme rezolvate, informații despre disciplină).
- **Suport aplicații practice**
  - VcDemo: „Image and Video Compression Learning Tool”, Developed at Delft University of Technology (<http://www-ict.its.tudelft.nl/~inald/vcdemo/>)
- **Cărți în limba română**
  - Bogdan Orza, „Codarea și compresia informațiilor multimedia”, Editura Albastra, 2007;
  - Aurel Vlaicu, „Prelucrarea imaginilor digitale”, Editura Albastra, 1997;
  - Bogdan Orza, Aurel Vlaicu, Camelia Popa, Mihaela Gordan, “Viziunea computerizată în exemple și aplicații practice”, UT Press, Cluj-Napoca, 2007.

# Bibliografie curs

- **Cărți în limba engleză**

- Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing (3rd Edition), Prentice Hall, 2008 (nr.inventar UTCN - 522.190)
- David Salomon, „Data Compression The Complete Reference”, Springer-Verlag, ISBN - 978-1-84628-602-5, 2007 (nr. Inv. UTCN – 522.269)
- Vasudev Bhaskaran, Konstantinos Konstantinides, „Image and Video Compression Standards Algorithms and Architectures”, Kluwer Academic Publishers, 1997, ISBN - 0-7923-9952-8
- Jerry D. Gibson, Toby Berger, Tom Lookabaugh, Dave Lindbergh, Richard L. Baker, „Digital Compression for Multimedia”, Morgan Kaufmann Publishers, 1998, ISBN- 1-55860-369-7
- Iain E. G. Richardson, „Video Codec Design”, John Wiley and Sons, 2007, ISBN-978-0-471-48553-7 (nr.inv. UTCN-522.193)
- I. Pitas, „Digital Image Processing Algorithms and Applications”, John Wiley & Sons, 2000, ISBN-0-471-37739-2, (nr.inv. UTCN-522.260)
- David S. Taubman, Michael W. Marcellin, „JPEG2000 Image Compression Fundamentals, Standards and Practice”, Kluwer Academic Publishers 2002, ISBN-0-7923-7519-X

# Modalitatea de evaluare

- 1) Examen scris: teorie & probleme = max 6 p
  - 2) Evaluare activitate laborator & proiect = max 3 p
  - 3) Activitate & prezenta curs = max 1 p
    - + 1p bonus pe activitate (curs + laborator + proiect)
- ⇒ Total: 11 p (nota max. = 10)

NU SE POATE LUA 10,

- daca nu exista activitate in cadrul orelor (interes ridicat pentru materie)

- NU doresc sa imi cereti sa scurtez orele
  - daca nu sunteti interesati de materie – cursurile nu sunt obligatorii/ iar la ora de laborator fa totul automat
  - daca aveti sugestii de imbunatatire tematica – astept sugestii in privat (fara sa va penalizez) cu ce anume doriti sa schimb
  - DORESC sa EXISTE COMUNICARE intre noi!
- MASTERUL este o forma de invatamant axata pe CERCETARE – descrierea matematica a unor concept **este necesara!!!**

# Formate de imagine, standarde de compresie

- Formatul fișierului de imagine
  - Organizarea și stocarea standardizată
- “Containere” de imagine
  - Fișier care include mai multe tipuri de imagini
- Standarde de compresie
  - Definesc proceduri de compresie și decompresie

# Organizațiile de standardizare

- ISO – International Standards Organization
- IEC – International Electrotechnical Commission
- ITU-T – International Telecommunications Union; fostă CCITT – Consultative Committee of the International Telephone and Telegraph
- Standarde de compresie video
  - VC-1 dată de SMPTE – Society of Motion Pictures and Television Engineers
  - AVS – audio-video standard – Ministerul Informațiilor din China
- Alte organizații

# Sisteme avansate de codare și compresie a datelor multimedia

## Curs 2 – Curs introductiv

Sl.Dr.Ing. Camelia FLOREA

Intelligent and multimodal image processing and analysis group (IMIPA),

Communications Departament, ETTI, TUCN,

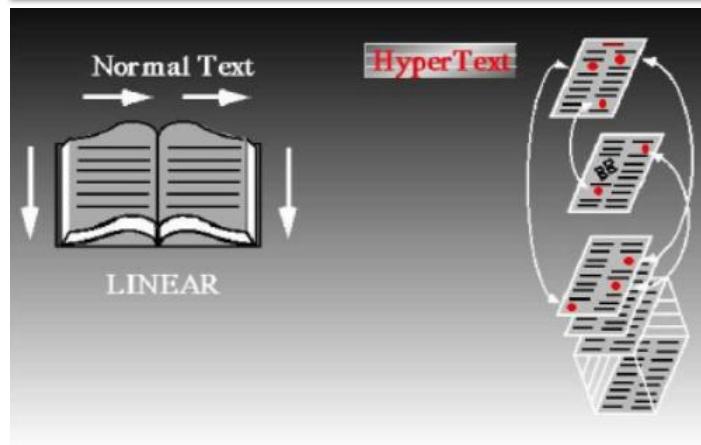
E-mail: [Camelia.Florea@com.utcluj.ro](mailto:Camelia.Florea@com.utcluj.ro),

Address: C. Daicoviciu, 15, room 431, Cluj-Napoca, RO.

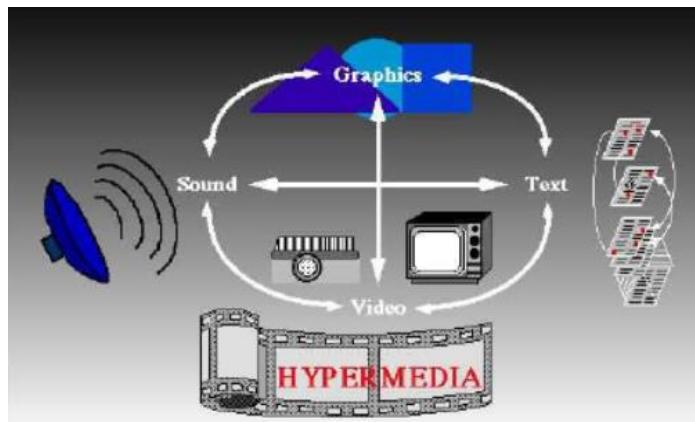
# Reprezentarea informației - Multimedia

- **multimedia** ⇔ reprezentarea informației sub diferite forme: text, imagini, audio, video etc.
- *Totuși,*
  - nu este suficientă reprezentarea informației sub diferite moduri
  - ci este necesar să fim capabili
    - să controlăm/ manipulăm/ transmitem aceste date în diferite formate
- *Astfel,* **multimedia** ⇔ domeniul care
  - se ocupă cu **integrarea comandată și controlată de calculator a diferitelor tipuri de informații:** text, grafică, imagini statice și în mișcare, animație, sunete, secvențe video
  - și în care fiecare tip de informație este reprezentată, memorată, procesată și transmisă sub formă digitală.

# HyperText și HyperMedia



- Aplicații HyperMedia
  - World Wide Web
  - Powerpoint
  - Adobe Acrobat (or other PDF software)
  - Adobe Flash
  - Video-on-Demand
  - Virtual Reality
  - Computer Games
  - Internet video streaming
  - Video conferencing
  - Digital Video Editing and Production Systems
  - Multimedia Database Systems



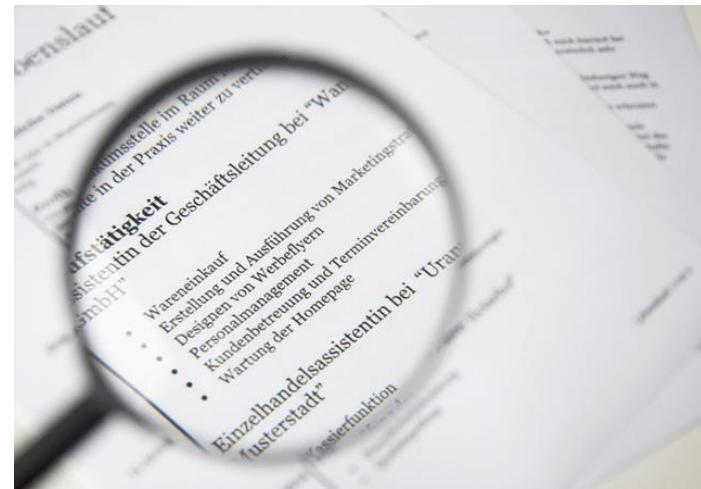
- HyperMedia – include imagini, grafică, video, sunet, ...

# Reprezentarea digitală a informațiilor multimedia

- producția și consumul de **materiale multimedia** => informațiile multimedia trebuie să aibă format digital -> pentru a putea fi gestionate de un sistem de calcul
  - Sistemul de calcul realizează digitizarea, manipularea și stocarea informațiilor multimedia
    - au la baza circuite cu două stări stabile (nivele de tensiune 0 și 1)  
= valori denumite **biți** -> aranjați în **octeți** -> aranjați în o succesiune liniară.
  - Modalități de **interpretare a modelelor de bit**
    - se poate face **prin asocierea cu numere** (cea mai simplă)
      - caracterele *textului* sunt reprezenta prin asocierea unui *număr unic fiecărei litere*
      - *luminozitatea* unei imagini într-un punct – valori între 0-255 (repräsentare pe 8 biti)
      - *amplitudinea instantanee* a undei de sunet, etc.
    - există **diverse modalități de implementare** a acestei asocieri,
      - singura problemă pe care trebuie să o rezolvăm este de a „programa” dispozitivele hardware/ software să fie capabile să interpreteze formatul digital pe care îl transmitem
- **STANDARDIZARE formate/ fișiere**

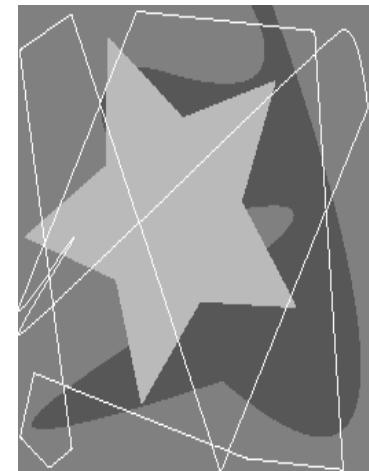
# Reprezentare: Text

- Input/generare: keyboard, speech input, optical character recognition (OCR), date stocate pe disc.
- Stocare/reprezentare caracter cu caracter:
  - 1 octet pe caracter (text), e.g. ASCII;
  - > 1 octeți pentru Unicode.
  - pentru alte tipuri de date (e.g. Spreadsheet files).
    - stocare ca text (cu formatare, e.g. CSV - Comma-Separated Values) sau stocare/codare binară
- Text formatat: Raw text sau text formatat - e.g HTML, Rich Text Format (RTF), Word or a program language source (Java, Python, MATLAB etc.)
- Compression: convenient to bundle files for archiving and transmission of larger files. E.g. Zip, RAR, 7-zip. General purpose compression programs may not work well for other media types: audio, image, video etc.



# Reprezentare: imagini statice versus grafică

- **grafică vectorială** = reține informația structurală → modelare (vectorială/ matematică) a formelor
  - sunt formate din **obiecte** (*linii, curbe, cercuri, etc.*) care au **attribute** (*grosime, scară de gri, culoare, tipar de umplere, etc.*)
  - sunt **revizuibile**, pot fi modificate, mutate, completate sau șterse.
  - obținute prin utilizarea unui **editor grafic**
  - Standarde grafice: OpenGL (Open Graphics Library), PHIGS, GKS
- **imaginile statice sau bitmap** = hartă de pixeli: o înregistrare a valorii fiecărui pixel din imagine (se atribuie o valoare pentru fiecare pixel)
  - NU sunt **revizuibile** – se poate doar ștergere și redesenare sau procesare/analiză conținut
- **dezavantajul major ale imaginilor bitmap** constă în mărimea relativ mare a fișierelor de imagine în comparație cu grafica vectorială.
- **imaginile și grafica** sunt adesea combinate
  - => o reprezentare **imagistică hibridă**.
- **conținutul semantic al desenului**
  - inclus în reprezentare



# Metode de producere a imaginilor BITMAP

- Captarea imaginilor din lumea reală (sub forma **bitmap** – imagini scanate)
  - cameră digitală
  - cameră analogică + placă de captură
  - scanare
- Create cu asistența calculatorului = ***imagini sintetizate***
  - prin generarea automata a imaginii de către un program
  - utilizarea unui program de editare specializat
  - captare ecran
  - conversia din format grafic la un format de tip imagine (ex. bitmap)
- ***Editoarele de imagine***
  - sunt programe similare cu ***editoarele grafice***, dar nu produc documente care rețin *conținutul semantic*.
  - oferă opțiuni mai sofisticate, în special pentru aplicații fotografice: procesare de imagine.
- „**A picture is worth a thousand words, but it uses up three thousand times the memory.**”

# Audio

---

- Input:
  - microfon
- CD Quality Audio necesita
  - 16-bit sampling la 44.1 KHz:
  - 24-bit, 96 KHz - audiophile rates
- fara compresie - 1 minut de audio
  - Mono CD quality = 5 MB.
  - Stereo CD quality = 10 MB.
- In general semnalul audio este comprimat
  - e.g. MP3, AAC, Flac, Ogg Vorbis



# Video

- Input: video camera
- O secventa video (raw) poate fi vazuta ca o serie de imagini statice
  - in general - 25, 30 sau 50 cadre pe secunda.
- Stocate comprimat:
  - Mpeg, H.26x, etc.



# Reprezentarea informației: Culoarea

- Culoarea este o senzație subiectivă produsă la nivelul creierului - ochiul uman conține două tipuri de celule receptoare:
  - **Bastonașele**
    - celulele folosite pentru vederea nocturnă - acestea *nu pot distinge culorile*
  - **Conurile**
    - folosite pentru vederea *cromatică* - sunt capabile să interpreteze culorile.
    - celulele de tip con sunt împărțite în trei subcategorii, fiecare din acestea fiind capabilă să recepționeze o anumită lungime de undă.
- => Faptul că ochiul reacționează la trei stimuli (lungime de undă diferite), conduce la ideea de **teoria vederii tricrome** adică orice culoare poate fi reprezentată prin combinația a trei **culori primare** (uzual: *roșu, verde și albastru*).
- Prin **amestecul aditiv a culorilor primare**, folosind factori de ponderare adecvați, se poate obține orice culoare din natură.
- **culorile primare** diferă în funcție de **echipament** (**RGB** afișare, **CMYK** imprimare, etc.)

# Adâncimea de culoare

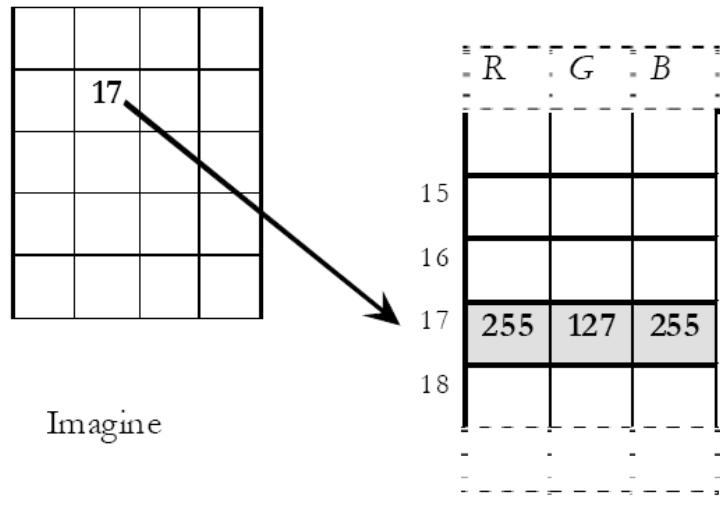
- 1bit/pixel
  - imagini binare (imaginile alb-negru, de fax, etc.)
- 4 biți/pixel (precizie mică)
  - imagini cu nivele de gri (16 nivele de gri)
  - grafică pentru calculator
- 8 biți/pixel
  - imagini atonale (“grayscale”- pe scara de gri)
  - Imagini color - 256 culori (uz general)
- 16 biți/pixel
  - utilizat de unele sisteme de calcul ( $r=5$ biți,  $g=6$ biți,  $b=5$ biți)
- 24 biți/pixel (true color)
  - sunt suficienți pentru a reprezenta toate culorile pe care ochiul uman le poate distinge
- 30, 36, 48 biți/pixel
  - aplicații profesionale

# Adâncimea de culoare

- Adâncimea de culoare
  - este factorul determinant în stabilirea **dimensiunii unui fișier**.
- Astfel, pentru stocarea informație de culoare
  - pentru 8 biți/pixel, spațiul necesar stocării este de 1 octet/pixel ( $W \times H \times 1$  octet)
  - 24 de biți/pixel este nevoie de 3 octeți/ pixel ( $W \times H \times 3$ ),
- Prin reducerea *adâncimii de culoare*
  - *putem obține o scădere substanțială a dimensiunii unui fișier*,
  - dar, se poate ajunge la o reducere a calitatii imaginii, însă nu în mod obligatoriu.
- Orice decizie în sensul reducerii adâncimii de culoare va trebui luată ținând cont
  - de imagine,
  - caracteristicile aplicației
  - și ale sistemului pe care acesta rulează.

# Indexarea culorilor - Colour Look-Up Tables (LUTs)

- Folosirea **colorilor indexate** - este utilă pe sisteme de calcul de performanță scăzută; și în cazul aplicațiilor de pe internet
  - În loc să folosim cate 8 biți pentru stocarea fiecărei componente R, G, B (= 3 octetăi)
    - => vom folosi un octet pentru stocarea unui index într-un tabel, cu 256 de poziții, fiecare dintre acestea conținând o valoare RGB pe 24 de biți.
  - fiecare imagine care utilizează culorile indexate va avea propria paletă de culori.
  - Paleta de culori se memorează în fișierul imaginii
  - Formate care permit reprezentarea/indexarea culorilor prin paleta
    - BMP, PNG, GIF, TIFF și altele.



## Utilizarea paletei de culori



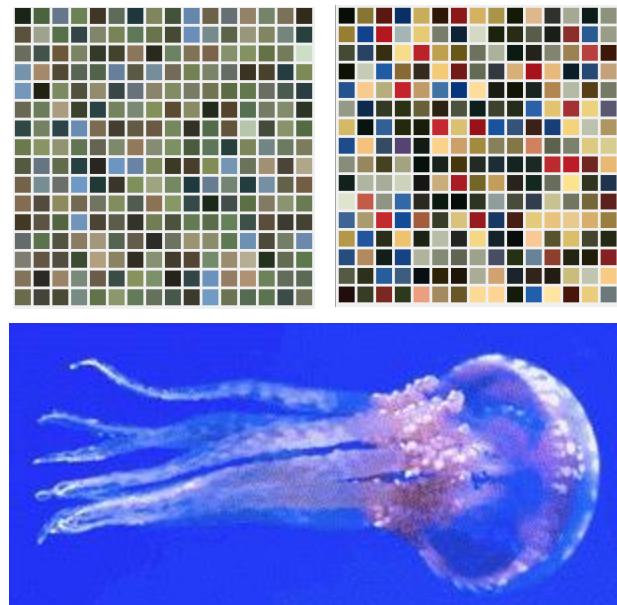
# Spațiul necesar stocării BMP

---

- Pentru o imagine de 640 x 480 pixel, este necesar:
  - Binar (1 bit) = 37.5 KB.
  - Pe nivale de gri (8 biți) = 300 KB.
  - Color (24-bit) = 921.6 KB
  - Color (8-bit, 256 culori) = 307.2 KB

# Indexarea culorilor

- Paleta de culori = culorile care sunt conținute în imagine => nu vom avea probleme deosebite legate de calitatea imaginii obținute
- fiecare imagine care utilizează culorile indexate va avea propria paletă de culori:
  - imagine subacvatică – un set de culori ce conține foarte multe nuanțe de albastru
  - imagine a unei păduri – un set de nuanțe de verde.
- chiar dacă fișierul se mărește prin adăugarea paletelor de culori, per ansamblu mărimea fișierului este de aproximativ 3 ori mai mică (practic se trece de la 24 de biți/pixel la 8 biți/pixel)



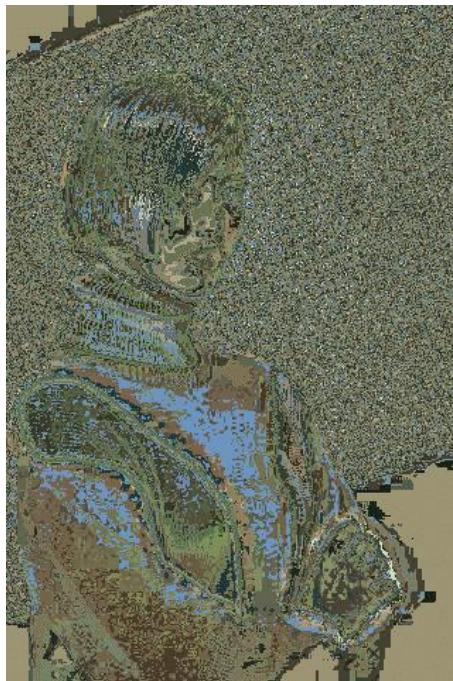


Imagine complexa (multe culori) apare efectul de contururi false.

Adăugare zgromot pseudo-aleator (dithering):



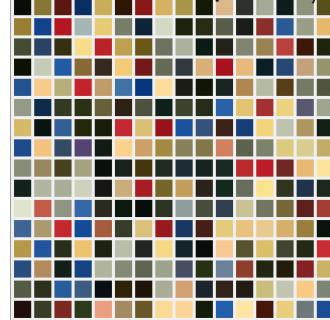
# Utilizând hărți de culoare/indexarea culorii



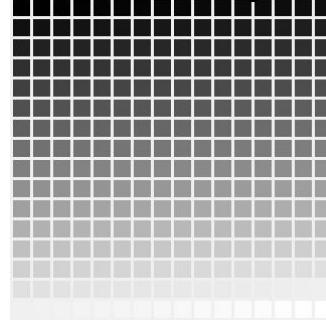
Paletă standard Web



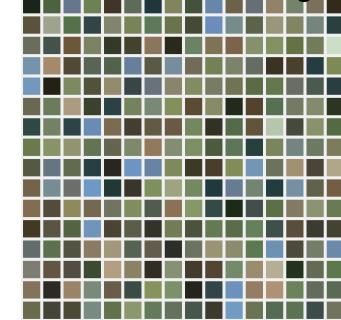
Paletă bazată pe conținut



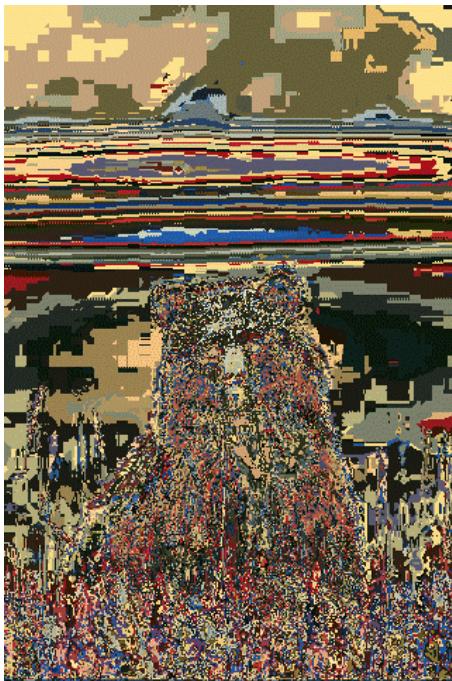
Paletă nivale de gri



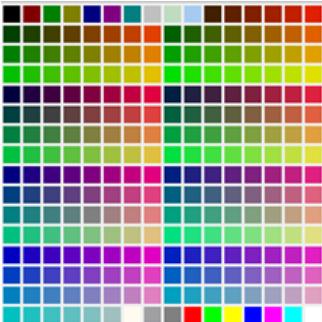
Paleta unei alte imagini



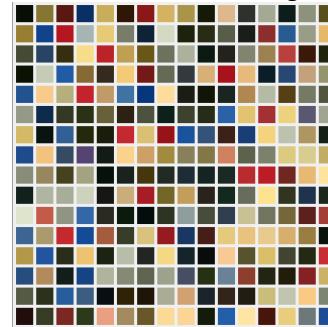
# Utilizând hărți de culoare/indexarea culorii



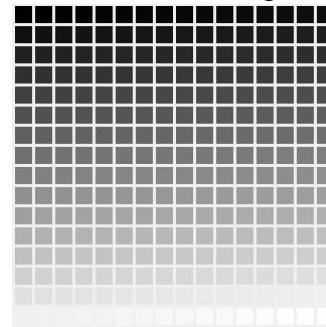
Paletă standard Web



Paleta unei alte imagini



Paletă nivele de gri

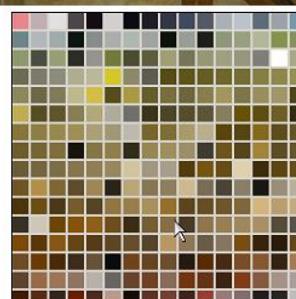


Paletă bazată pe conținut

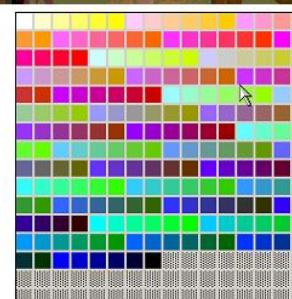


# Paleta de culori

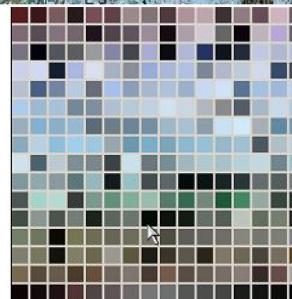
- imaginea originală cu 24 biți/pixel
- transformarea la o imagine cu 8 biți/pixel în care se folosește o paletă standard Web (figura 3.b) și se utilizează procesul de dithering
- reprezintă transformarea la nivele de gri (practic planul de luminanță al imaginii originale)
- reprezintă imaginea transformată la 8 biți/pixel utilizând o paletă (figura 3.a) cât mai apropiată de cea reală
- se utilizează din nou paleta Web din figura 3.b dar nu se folosește procesul de dithering
- prezintă imaginea transformată folosind paleta de culori 3.c a unei alte imagini



3.a



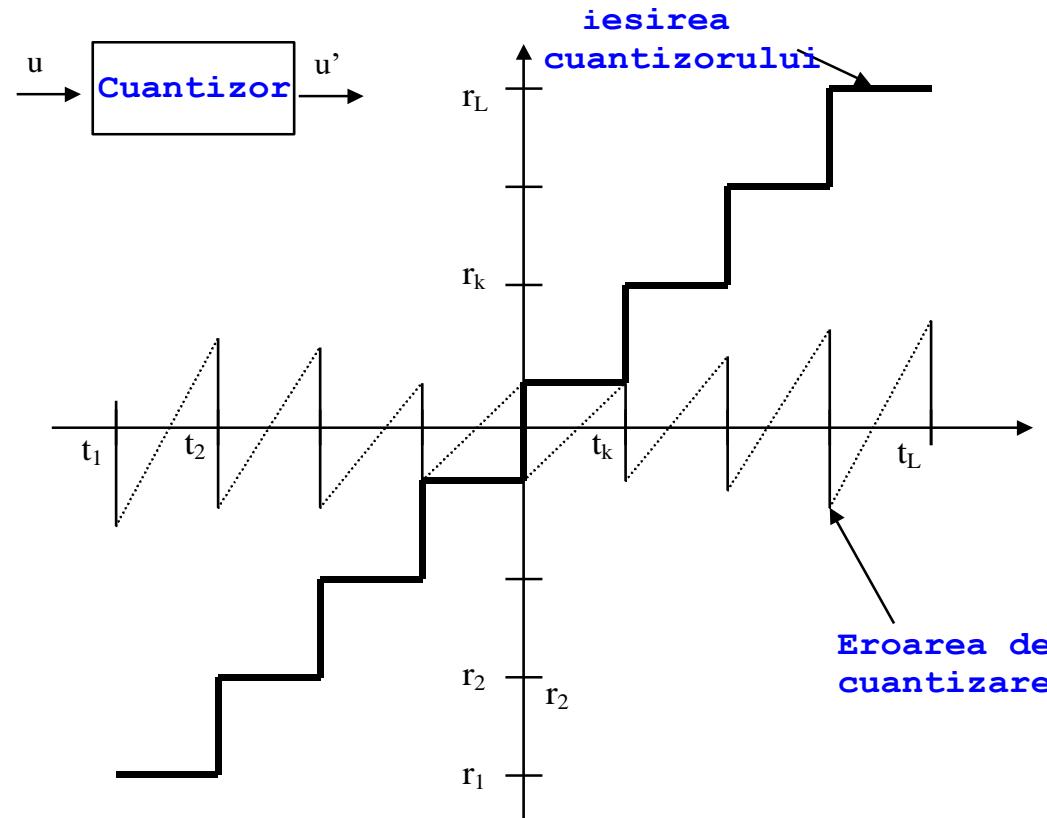
3.b



3.c

# Cuantizarea culorii/ Indexarea culorii

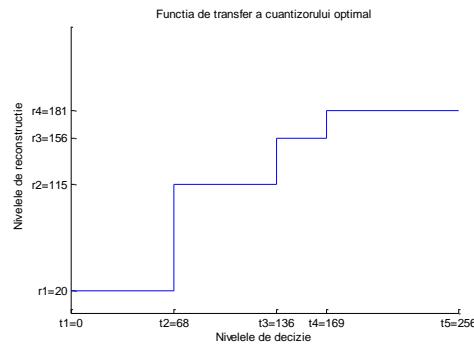
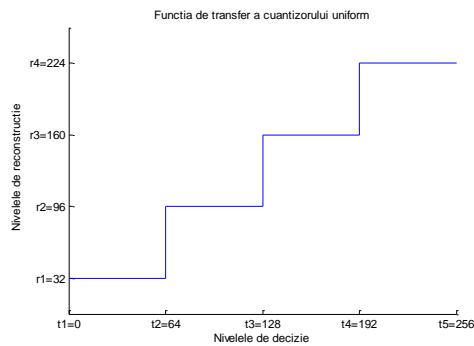
- Cuantizor uniform



# Cuantizare uniforma/ Cuantizare optimală



Imaginea necuantizata



Histograma imaginii necuantizate

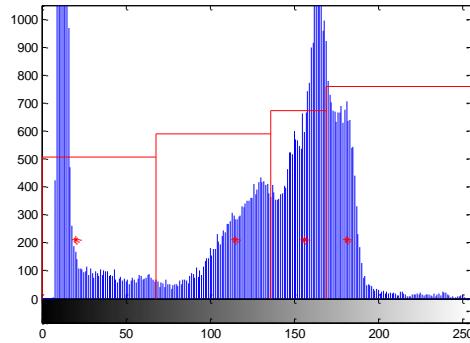
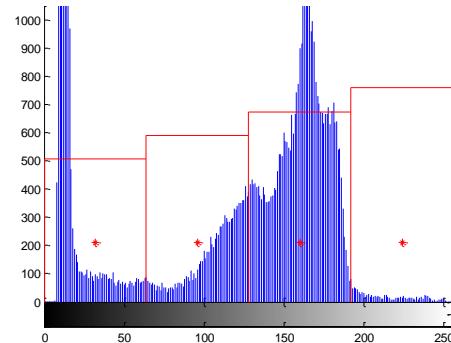
$$B=2 \Rightarrow L=4$$



Imaginea cuantizata uniform / optimal  
MSE=15



MSE=9.6



# Metode de cuantizare vizuala

- In general – daca  $B < 6$  (la cuantizarea uniforma) sau  $B < 5$  (la cuantizarea optimala) => apare fenomenul de "conturare" (apar contururi false) in imaginea cuantizata.
- "Conturarea" = grupuri de pixeli vecini sunt cuantizati la aceeasi valoare => apar regiuni de nivele de gri constante; contururile acestor regiuni sunt contururile false specifice "conturarii".
- Contururile false introduse de cuantizare nu contribuie semnificativ la MSE, dar sunt foarte deranjante vizual => este important sa reducem vizibilitatea erorii de cuantizare, nu doar MSE.
  - ⇒ Solutii: scheme de cuantizare vizuala, care sa mentina eroarea de cuantizare sub un nivel de vizibilitate acceptabil.
  - ⇒ O metoda foarte uzuală: cuantizarea cu zgomot pseudo-aleatoriu (dithering)



Cuantizare uniforma, B=4



Cuantizare optimala, B=4



Cuantizare uniforma, B=6

# Cuantizarea culorii/ Indexarea culorii



Img. Orig. B=24  
(16777216 culori)

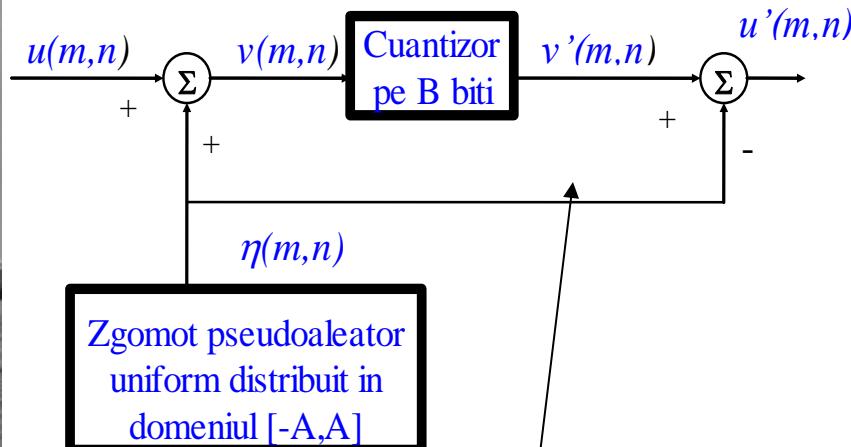
B=1 (2 culori)

B=2 (4 culori)

B=4 (16 culori)

B=8 (256 culori)

# Cuantizarea cu zgomot pseudo-aleator (“dither”)



Zgomot de amplitudine mare



Cuantizare uniforma,  $B=4$

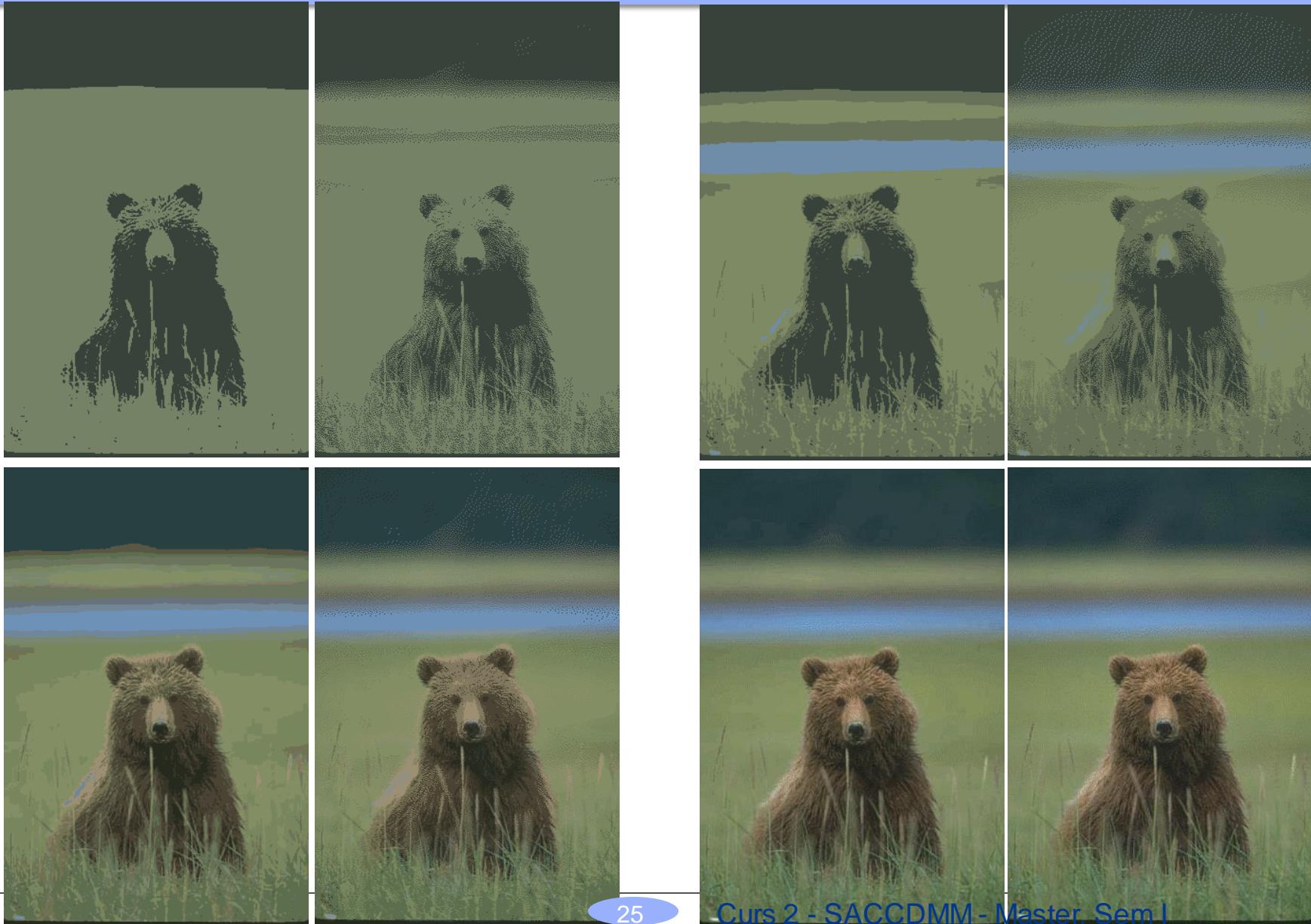


Inainte de scaderea zgomotului



Zgomot de amplitudine mica

# Cuantizare fără și cu zgomot pseudoaleator



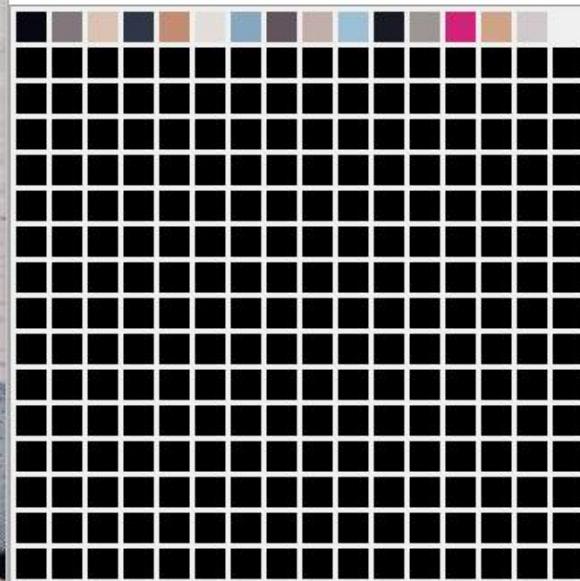
# Cuantizare fără și cu zgomot pseudoaleator





640 x 425 x 8 BPP | 4/12 | 102 % | 266.68 KB / 266.66 KB | 03.10.2017 / 14:47:15

Palette entries



Click on a color to select; double-click to edit

Index: Value:

Hint: click into the image (main window) to select the index here.

Refresh

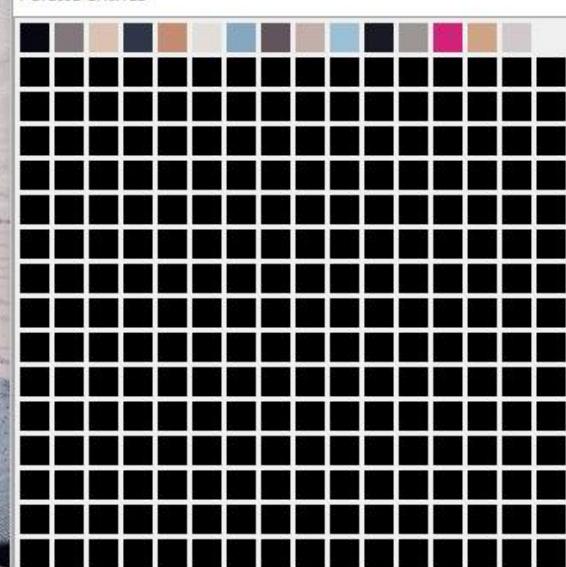
OK



640 x 425 x 8 BPP | 4/12



Palette entries



Click on a color to select; double-click to edit

Index: Value:

Hint: click into the image (main window) to select the index here.

Palette entries

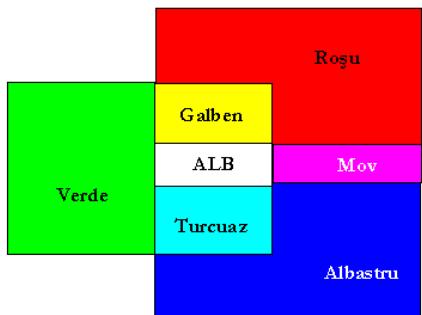


Click on a color to select; double-click to edit

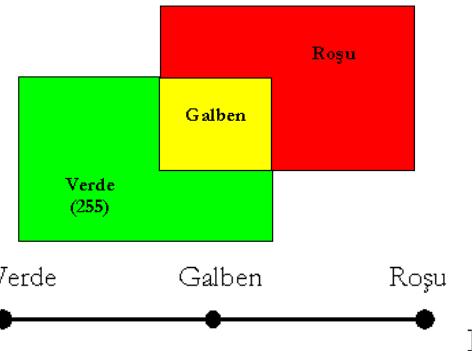
Index: Value:

Hint: click into the image (main window) to select the index here.

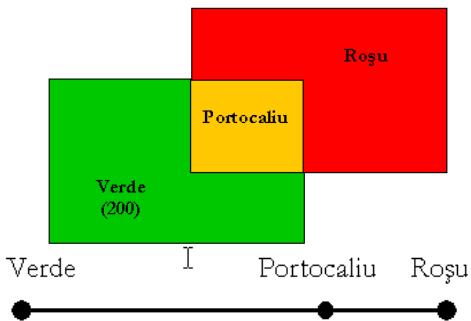
# Spațiul culorilor RGB



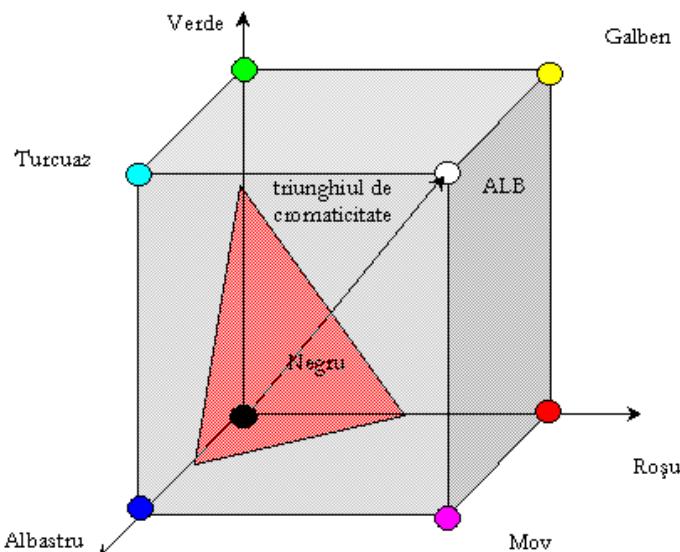
Amestec aditiv dintre **verde, roșu** și **albastru**



Amestec aditiv dintre **verde = 255** și **roșu = 255** (proporții egale – maxim)



Amestec aditiv dintre **verde = 200** și **roșu = 255**



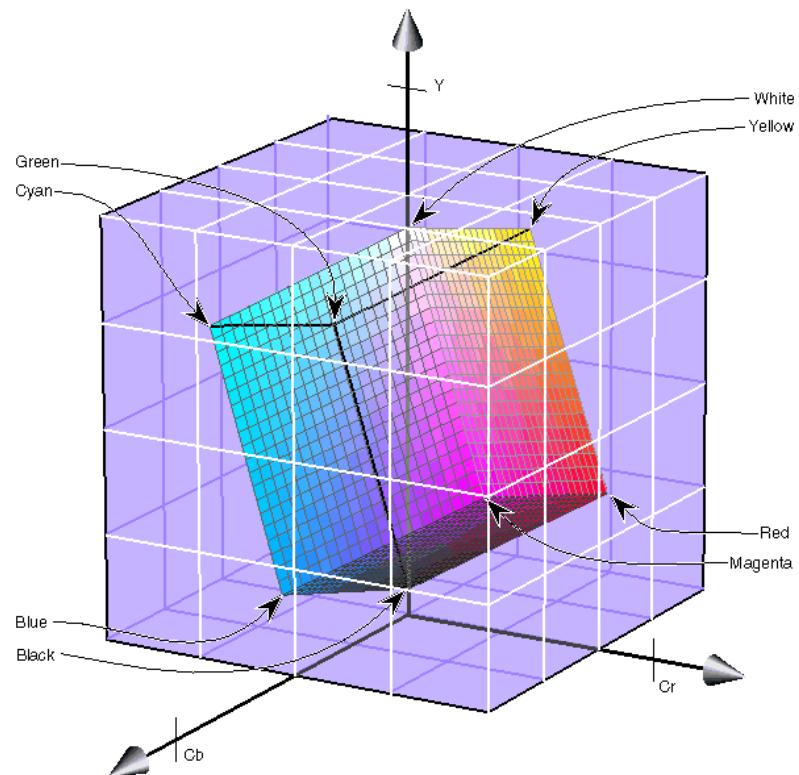
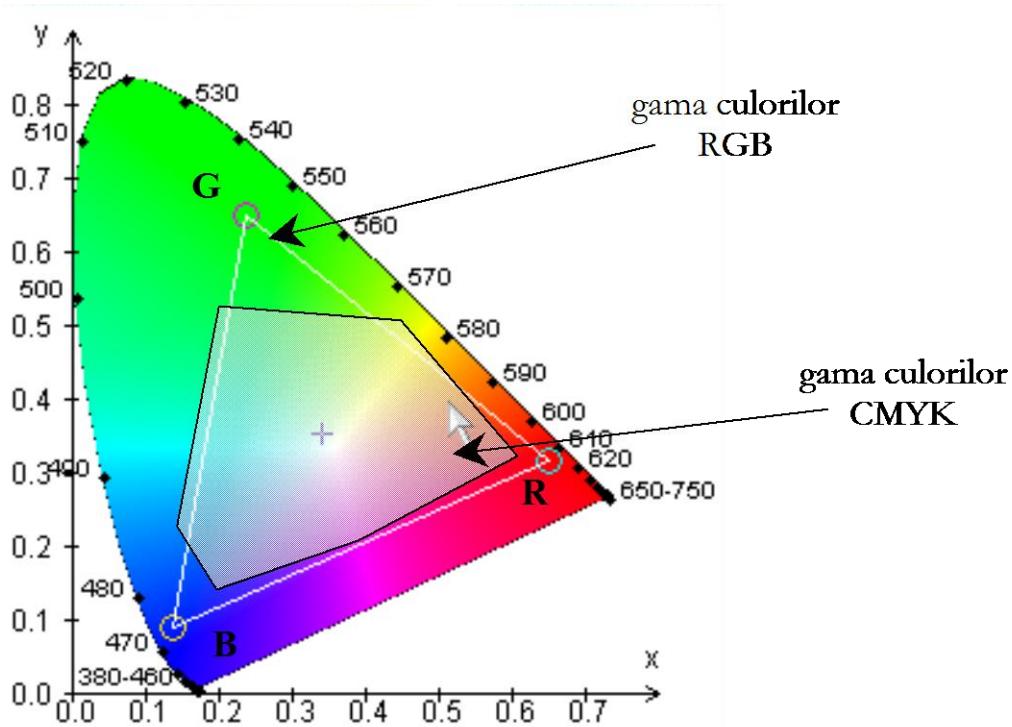
Componentele primare de culoare: R, G și B  
**sunt puternic corelate** ⇔ nu se pot analiza/  
procesa separat pentru descrierea culorii.

Imagine monocromă:

- toate punctele din imagine au  $R=G=B$   
⇒ cad pe diagonala cubului ( $R, G, B$ ).

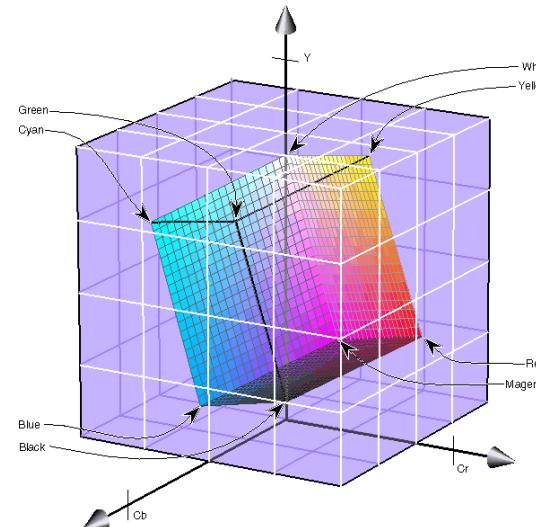
# Spații de culoare

- Tipuri de transformări:
  - liniare: YUV, YCbCr, YIQ, XYZ, etc.
  - neliniare: HSV, CIE L\*a\*b\*, CIE L\*u\*v\*, etc.

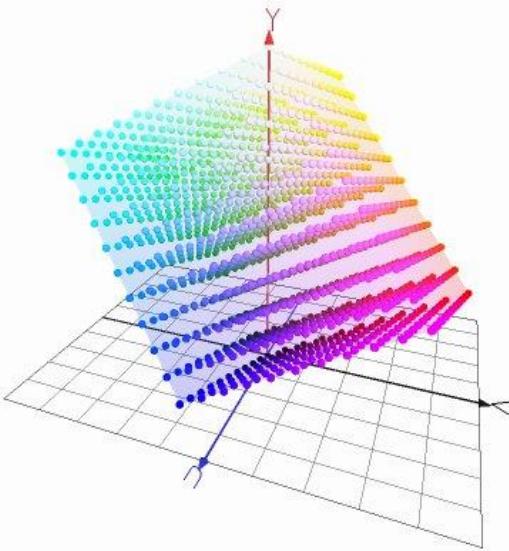


# Spații de culoare

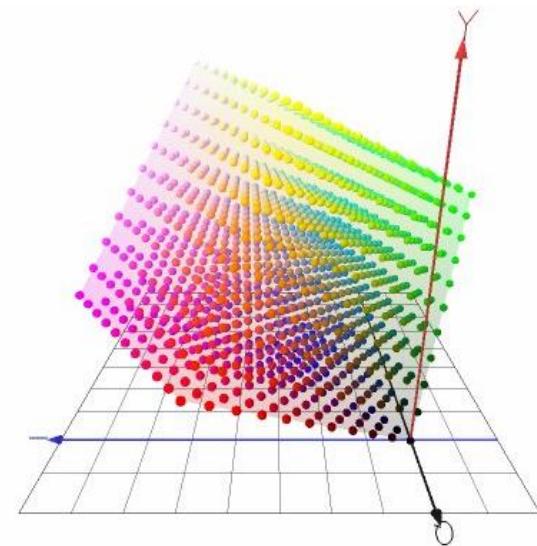
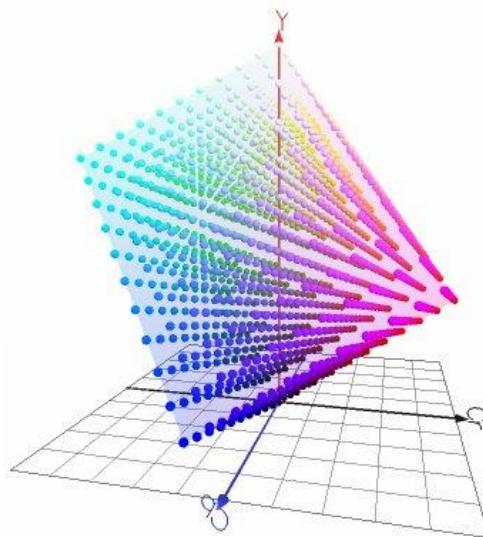
- **Tipuri de transformări:**
  - **liniare:** YUV, YCbCr, YIQ, XYZ, etc.
    - Besides the RGB representation, YIQ and YUV are the two commonly used in image/video.
  - **neliniare:** HSV, CIE L\*a\*b\*, CIE L\*u\*v\*, etc.
- Use Luminance and Chrominance to better encode how Humans 'see' colour.



# Transformari liniare ale spatiului RGB



$$\begin{cases} Y &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\ U &= -0.147 \times R - 0.289 \times G + 0.436 \times B \\ V &= 0.615 \times R - 0.515 \times G - 0.100 \times B \end{cases}$$

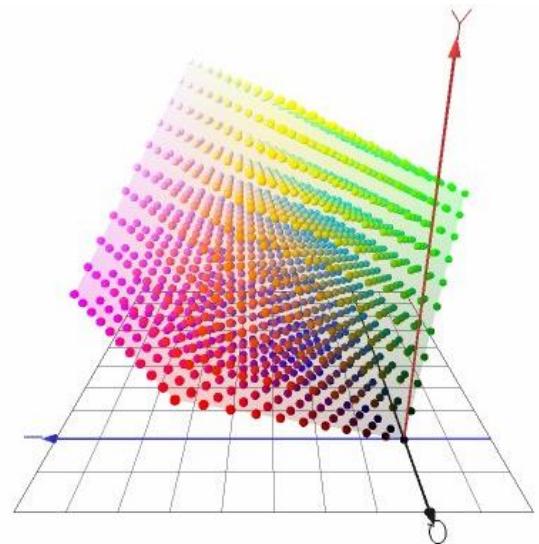


$$\begin{cases} Y &= 0.2989 \times R + 0.5866 \times G + 0.1145 \times B \\ Cb &= -0.1688 \times R - 0.3312 \times G + 0.5000 \times B \\ Cr &= 0.5000 \times R - 0.4184 \times G - 0.0816 \times B \end{cases}$$

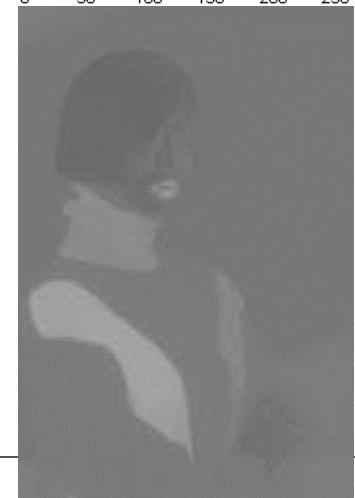
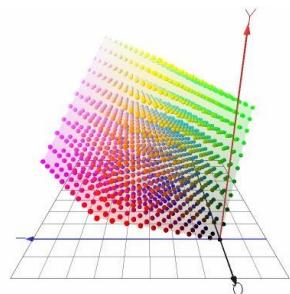
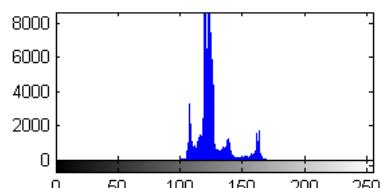
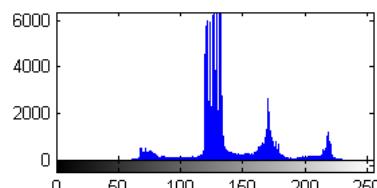
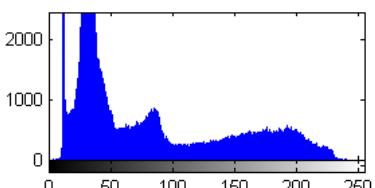
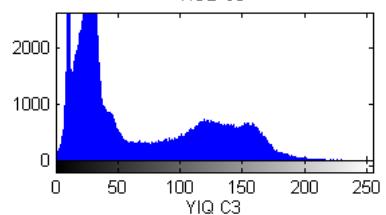
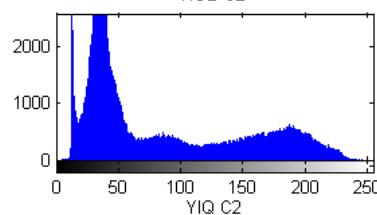
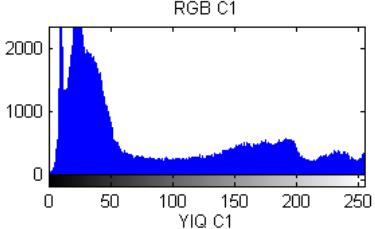
# YIQ Colour Model

- YIQ is used in colour TV broadcasting
- Y (luminance) is the CIE Y primary.
- I is red-orange axis, Q is roughly orthogonal to I.
- Eye is most sensitive to Y, next to I, next to Q.  
⇒ Analog Video Compression Scheme:
  - 4 MHz is allocated to Y,
  - 1.5 MHz to I,
  - 0.6 MHz to Q.

$$\begin{cases} Y &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\ I &= 0.596 \times R - 0.274 \times G - 0.322 \times B \\ Q &= 0.212 \times R - 0.523 \times G + 0.311 \times B \end{cases}$$



# RGB vs YIQ



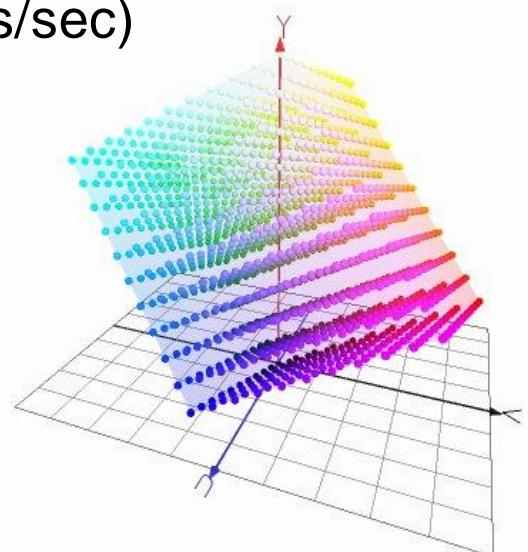
# YUV Color Model

- Digital video standard established in 1982
- Video is represented by a sequence of fields (odd and even lines). Two fields make a frame.
- Works in PAL (50 fields/sec) or NTSC (60 fields/sec)
- Uses the Y, U, V colour space

$$Y = 0.299R + 0.587G + 0.114B,$$

$$U = B - Y,$$

$$V = R - Y$$



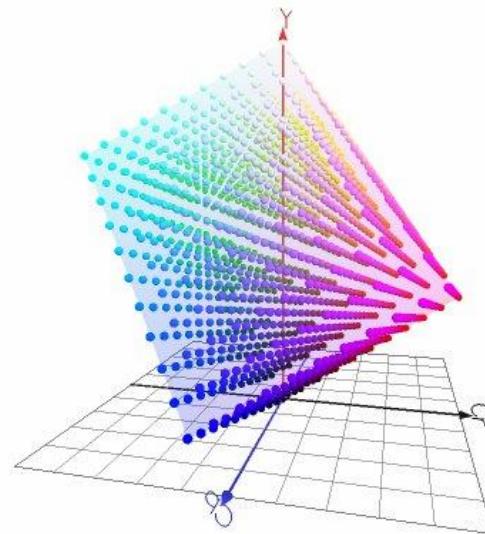
$$\begin{cases} Y &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\ U &= -0.147 \times R - 0.289 \times G + 0.436 \times B \\ V &= 0.615 \times R - 0.515 \times G - 0.100 \times B \end{cases}$$

# YCrCb Colour Model

- Similar to YUV
- YUV is normalised by a scaling

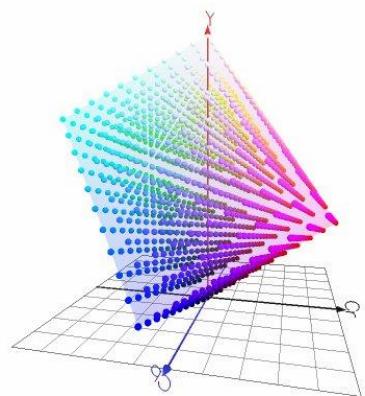
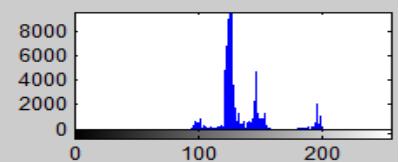
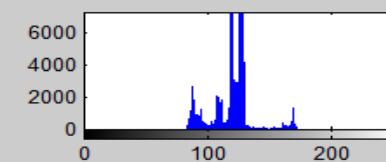
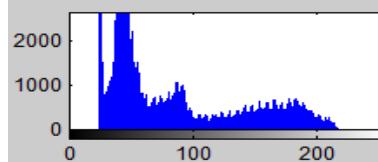
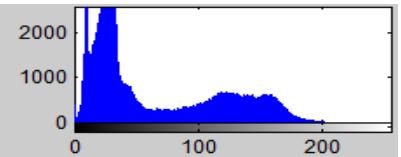
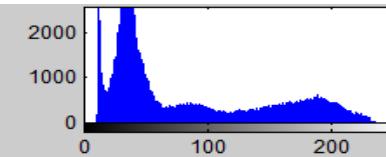
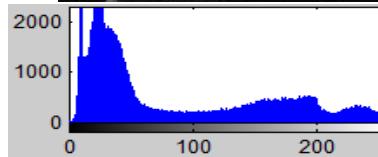
$$Cb = (B - Y)/1.772$$

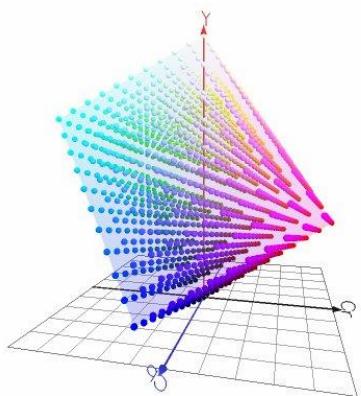
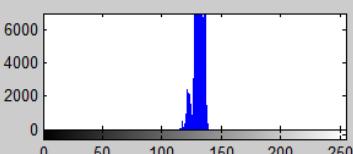
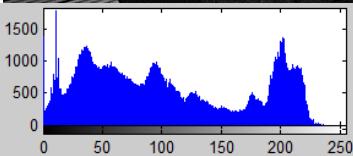
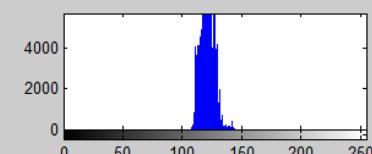
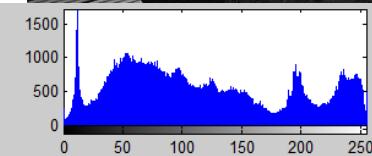
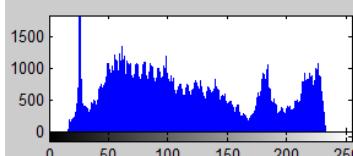
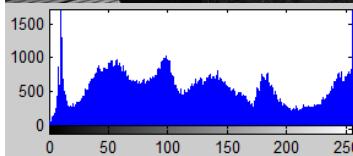
$$Cr = (R - Y)/1.402$$

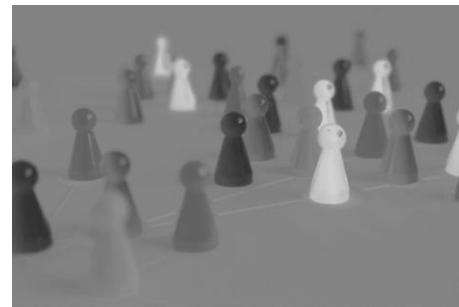
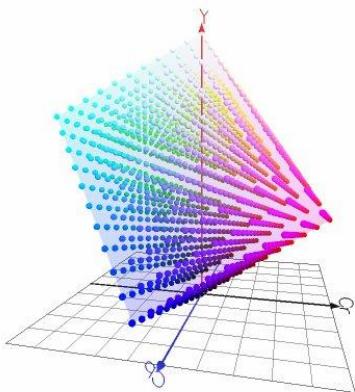
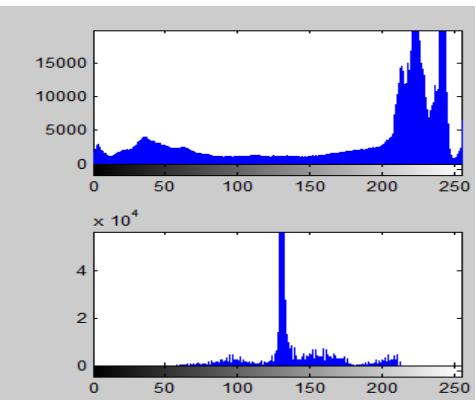
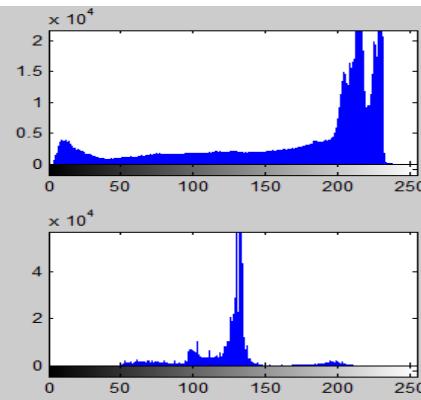
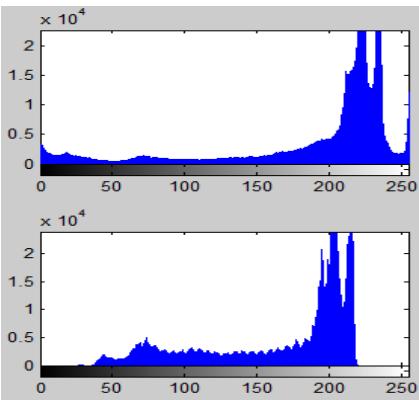


$$\begin{cases} Y &= 0.2989 \times R + 0.5866 \times G + 0.1145 \times B \\ Cb &= -0.1688 \times R - 0.3312 \times G + 0.5000 \times B \\ Cr &= 0.5000 \times R - 0.4184 \times G - 0.0816 \times B \end{cases}$$

# RGB vs YCbCr

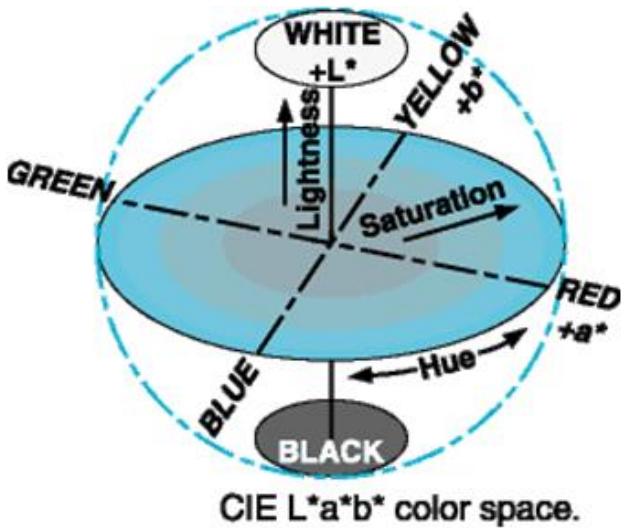






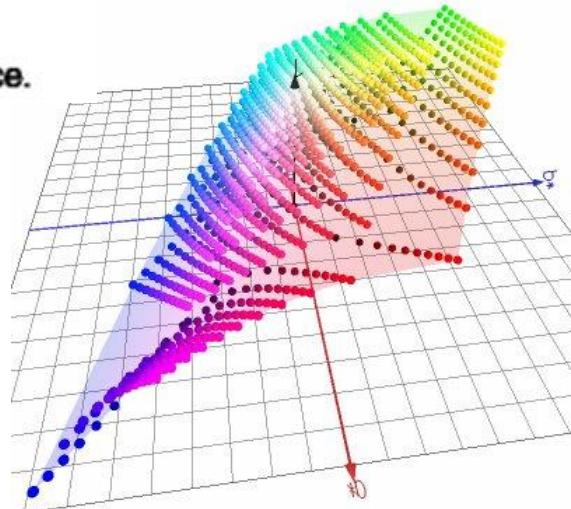
# Spatii de culoare - transformari neliniare

- Transformarea **CIE L\*a\*b\*** (similar – CIE L\*u\*v\*):
  - Conduce la un spatiu uniform perceptual

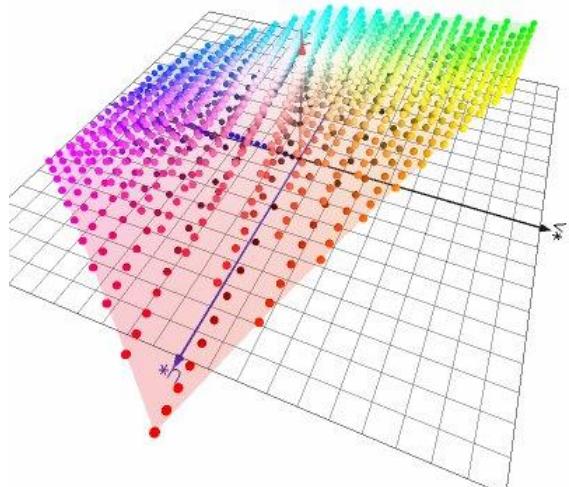


- Luminance: L; Chrominance: a,b
  - a - ranges from green to red,
  - b - ranges from blue to yellow

CIE L\*a\*b\*

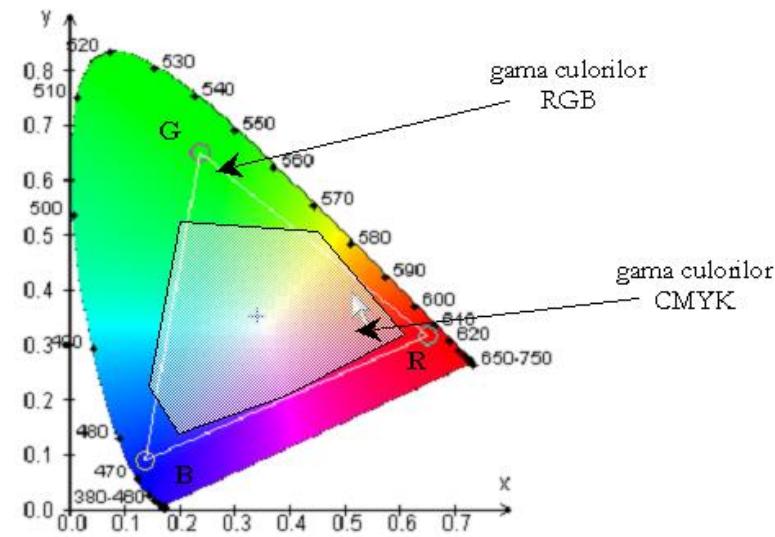
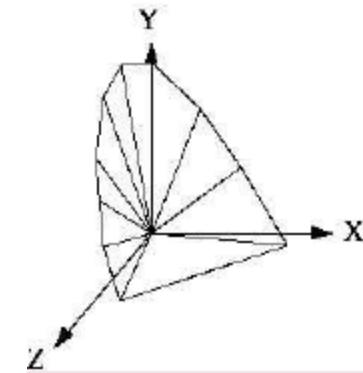


CIE L\*u\*v\*



# CIE Colour Model

- Human focussed Colour Models:
  - In 1931, the CIE defined three standard primaries ( $X$ ,  $Y$ ,  $Z$ )
    - The  $Y$  primary was intentionally chosen to be identical to the luminous-efficiency function of human eyes (Perceptual Model).
  - All visible colours are in a horseshoe shaped cone in the  $X$ - $Y$ - $Z$  space. Consider the plane  $X+Y+Z=1$  and project it onto the  $X$ - $Y$  plane, we get the CIE chromaticity diagram.
  - The edges represent the pure colours
  - White is at the dot
  - Vector-based colour model:
    - when added, any two colours (points on the CIE diagram) produce a point on the line between them.



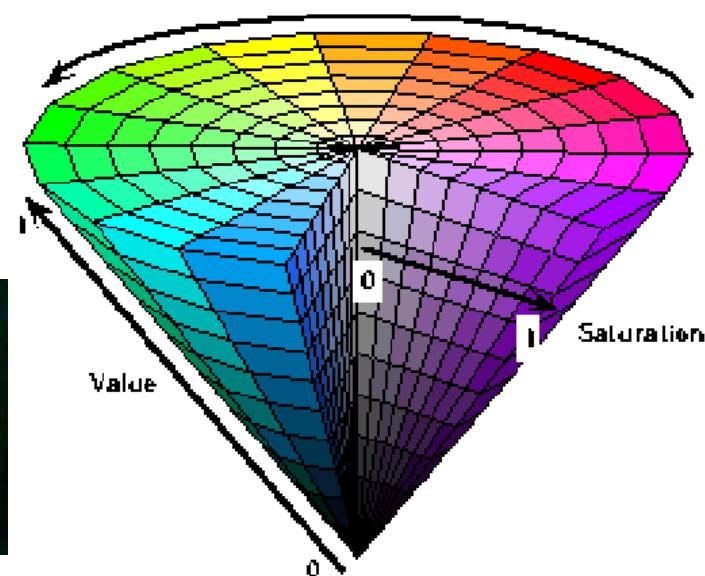
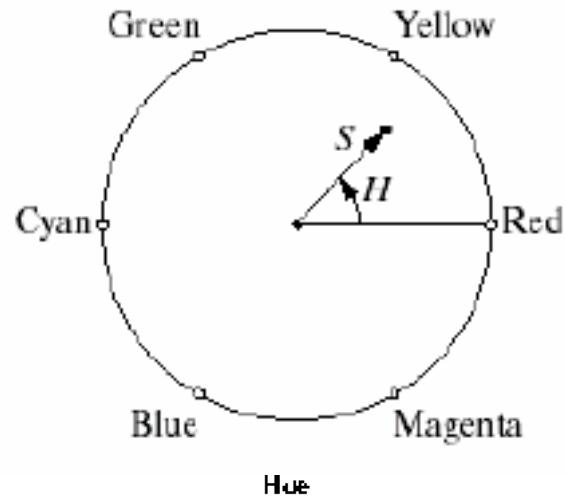
# Atributele perceptuale ale culorii

- **Atributele perceptuale ale culorii**

- atributele folosite de catre sistemul vizual uman pentru descrierea unei culori in procesul de perceptie si distinctie a culorii.

- **Sunt 3 atribute perceptuale ale culorii:**

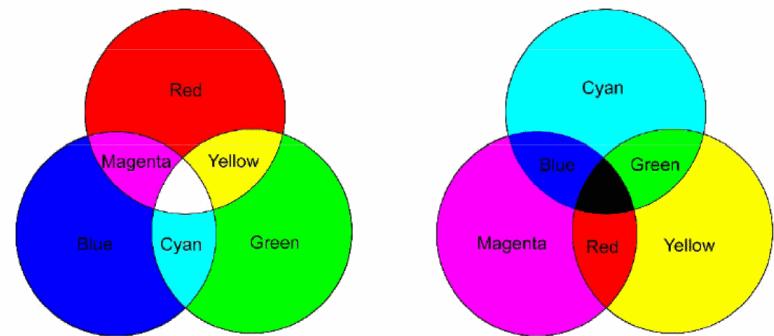
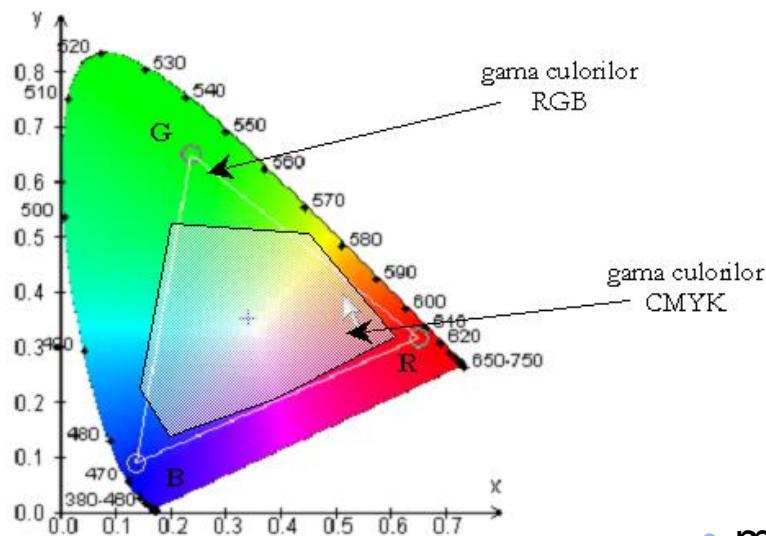
- Nuanța - culoarea de bază a unui obiect (familia de culori din care face parte) - primul criteriu prin care se deosebesc culorile
- Saturația (Puritatea)
- Luminanța (Strălucirea)



# Spațiul de culoare – CMYK

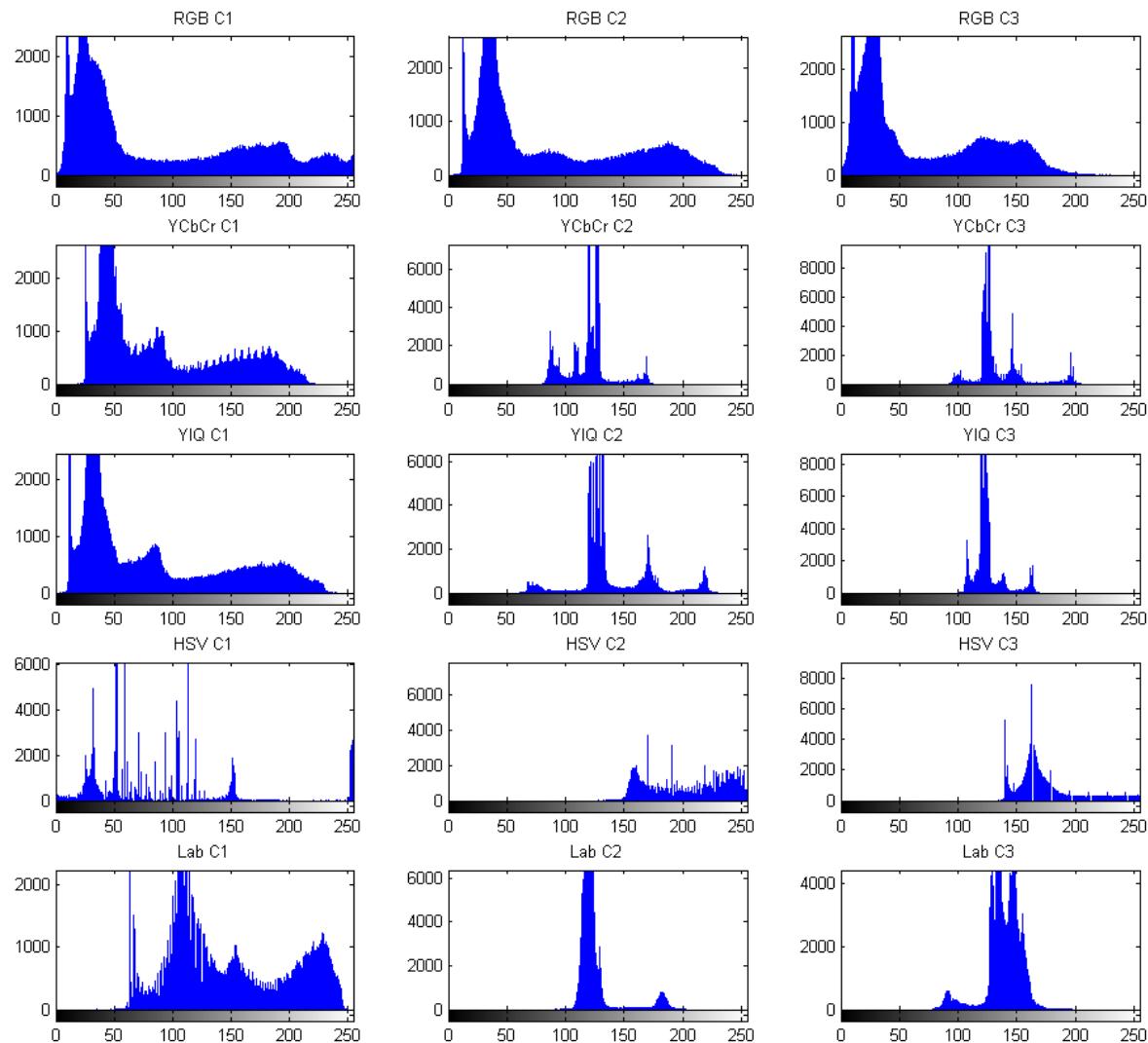
- culori complementare - descompunere din lumina albă
  - roșu – turcuaz (C – Cyan)
  - verde – mov (M – Magenta)
  - albastru – galben (Y – Yellow)
  - completare culoarea NEAGRĂ – BLACK

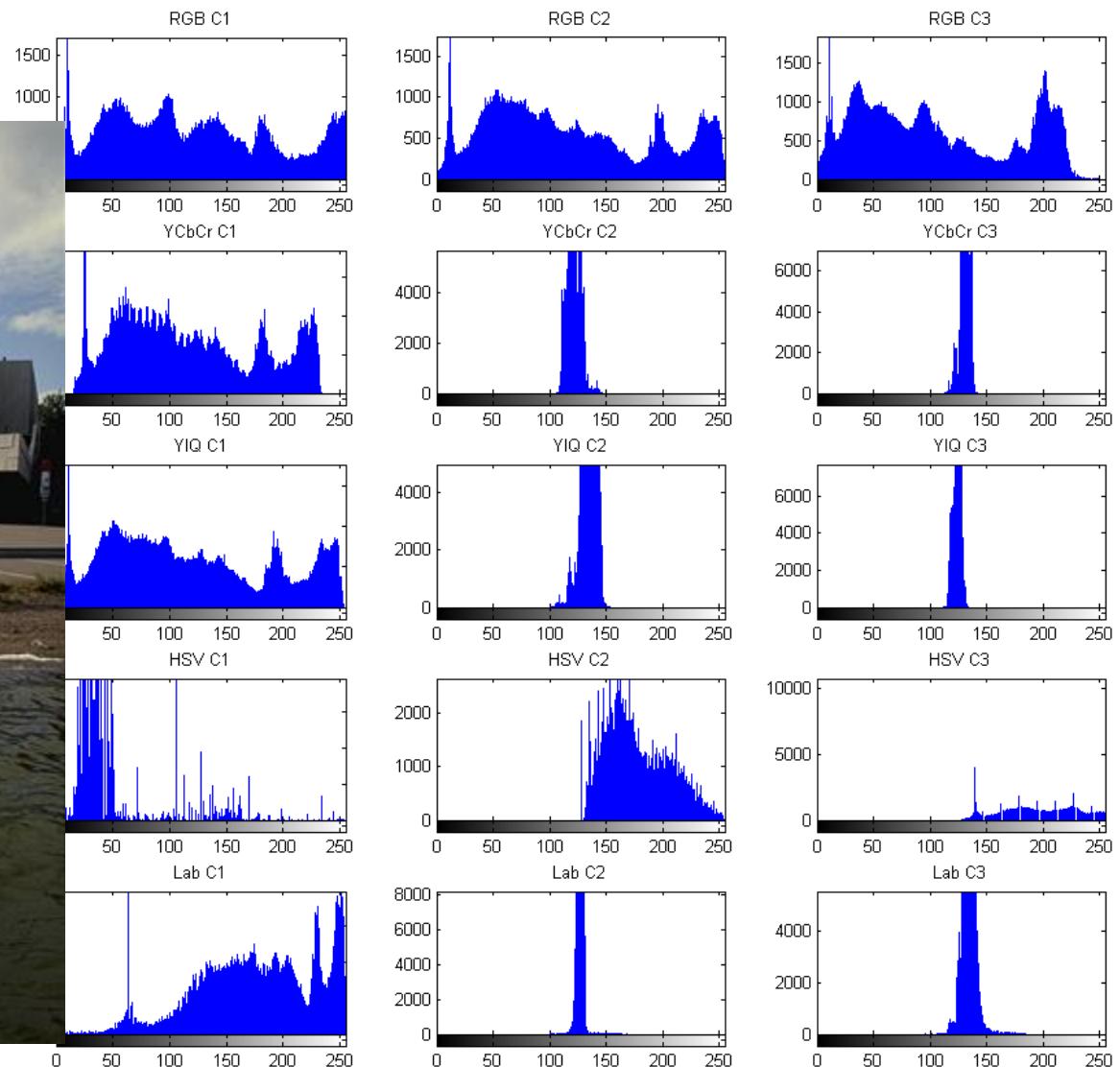
$$C = G + B = W - R$$
$$M = R + B = W - G$$
$$Y = G + R = W - B$$

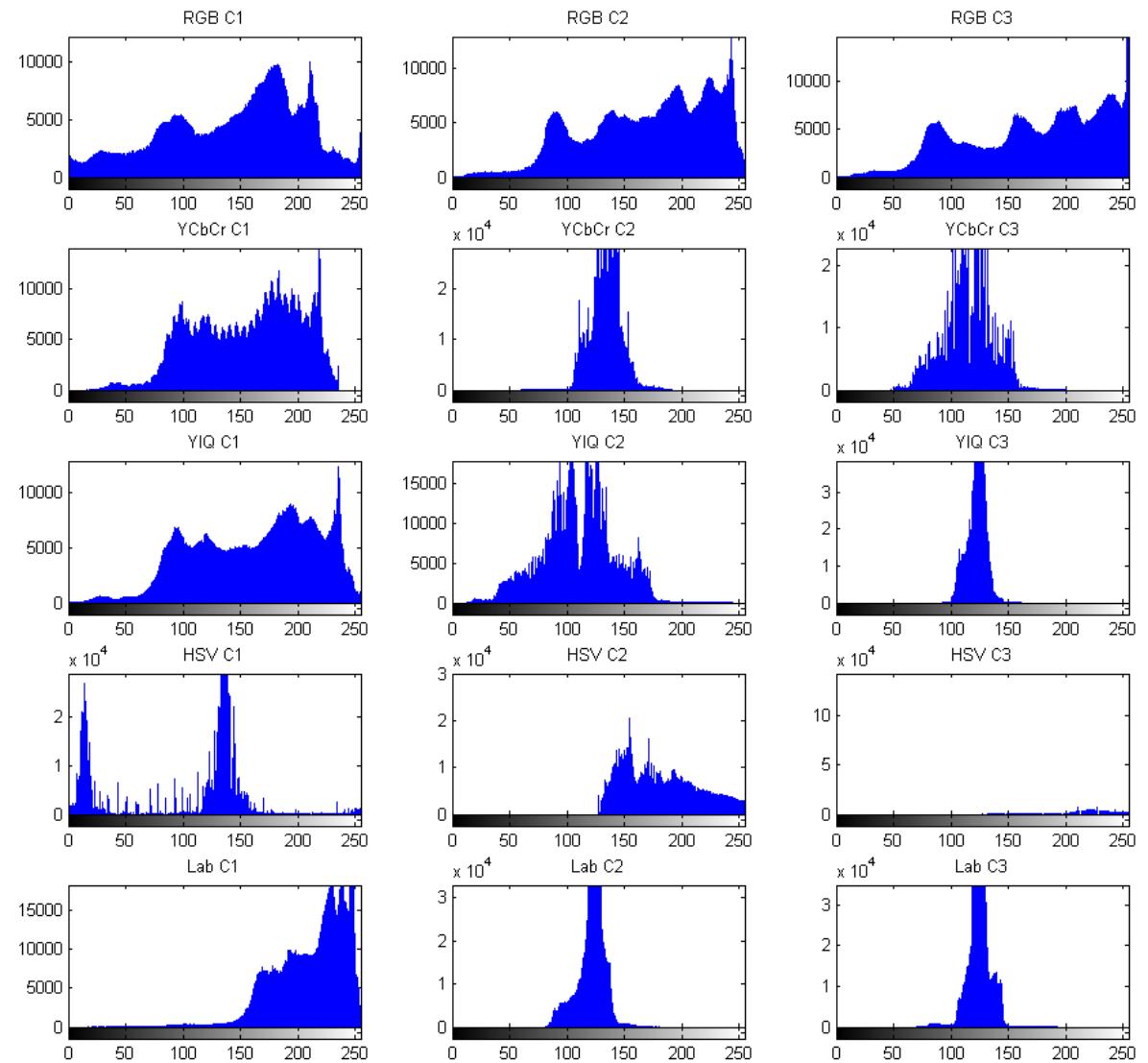


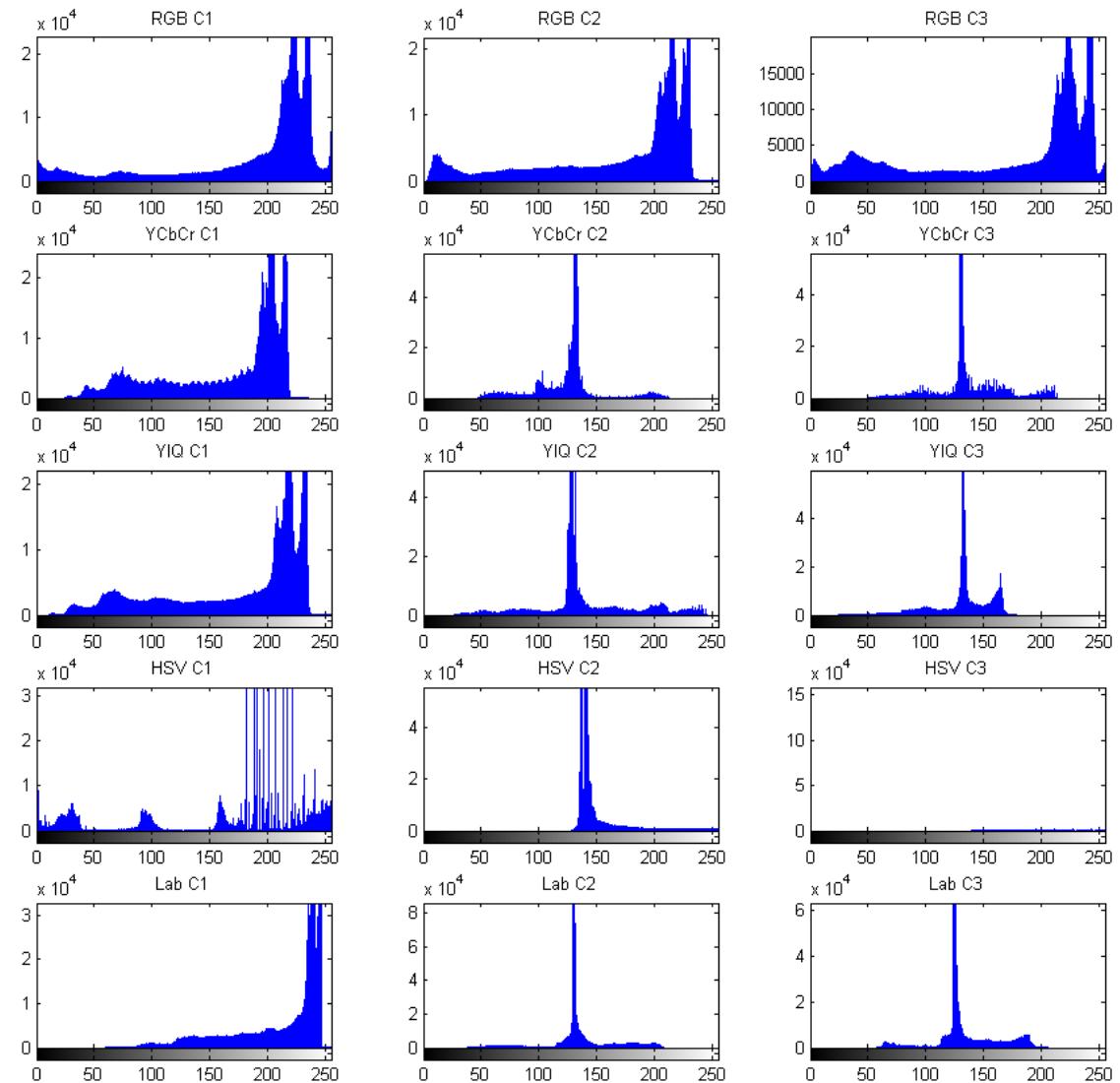
- mai apropiat de formarea culorii în realitate
- CMYK - culori de procesare pentru imprimare

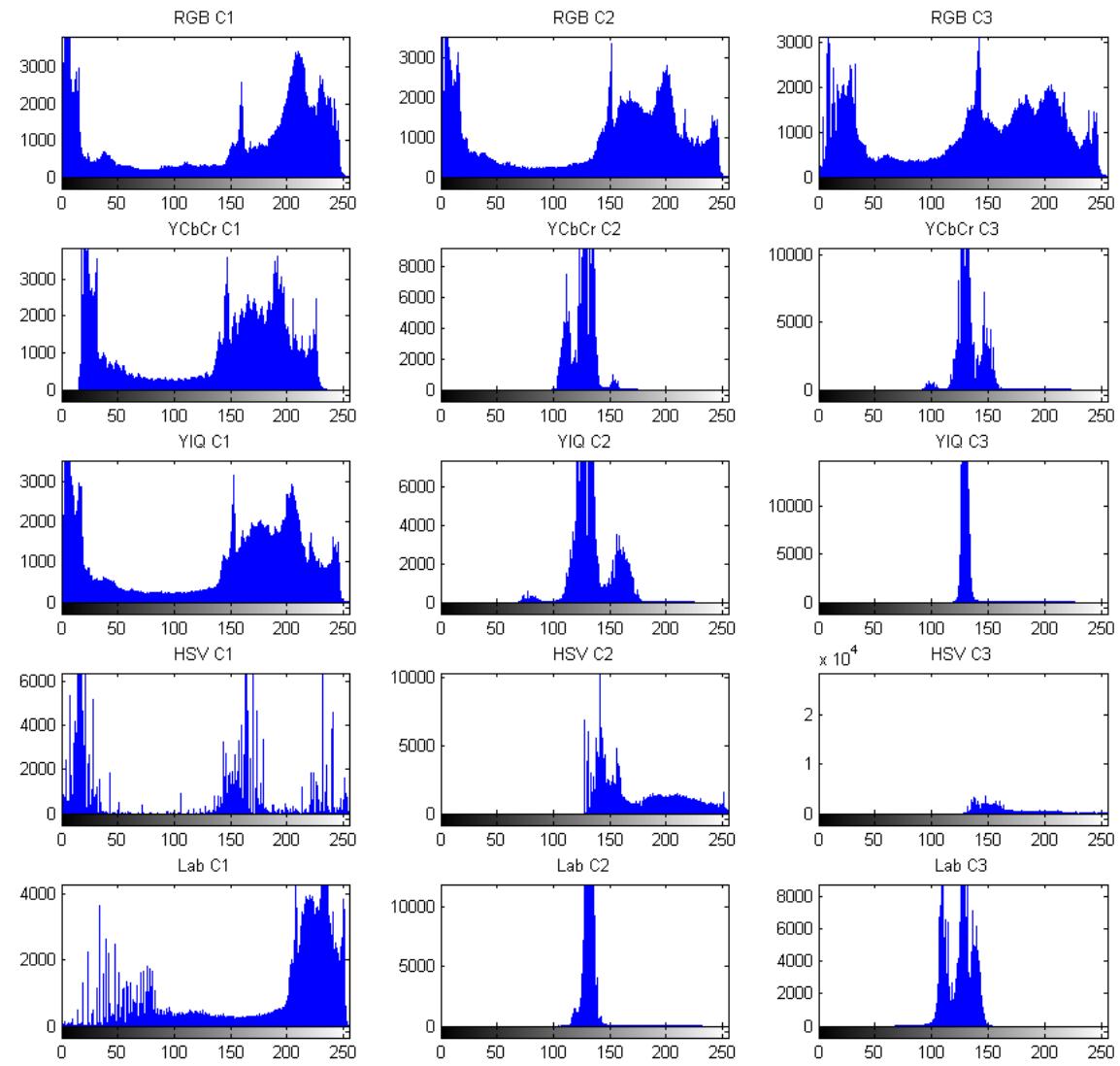
# Exemplificare Spații de Culoare











# Summary of Colour

- Colour images are encoded as triplets of values.
- Three common systems of encoding in video are
  - RGB, YIQ, YUV, and YCrCb.
  - besides the hardware-oriented colour models (i.e., RGB, CMY, YIQ, YUV),
    - HSB (Hue, Saturation, and Brightness, e.g., used in Photoshop) and
    - HLS (Hue, Lightness, and Saturation) are also commonly used.
- YUV/ YCrCb/ YIQ uses properties of the human eye to prioritise information
  - Y is the black and white (luminance) component,
  - U and V are the colour (chrominance) components
- YUV/YCrCb is a standard for digital video that specifies image size, and decimates the chrominance images (for 4:2:2 video)

# Types of Colour Video Signals

- **Component video** - each primary is sent as a separate video signal.
  - The primaries can either be RGB or a luminance-chrominance transformation of them (e.g., YIQ, YUV).
  - Best colour reproduction
  - Requires more bandwidth and good synchronisation of the three components
- **Composite video**
  - Colour (chrominance) and luminance signals are mixed into a single carrier wave.
  - Some interference between the two signals is inevitable.
- **S-Video** (Separated video, e.g., in S-VHS)
  - A compromise between component analog video and the composite video.
  - It uses two lines, one for luminance and another for composite chrominance signal.

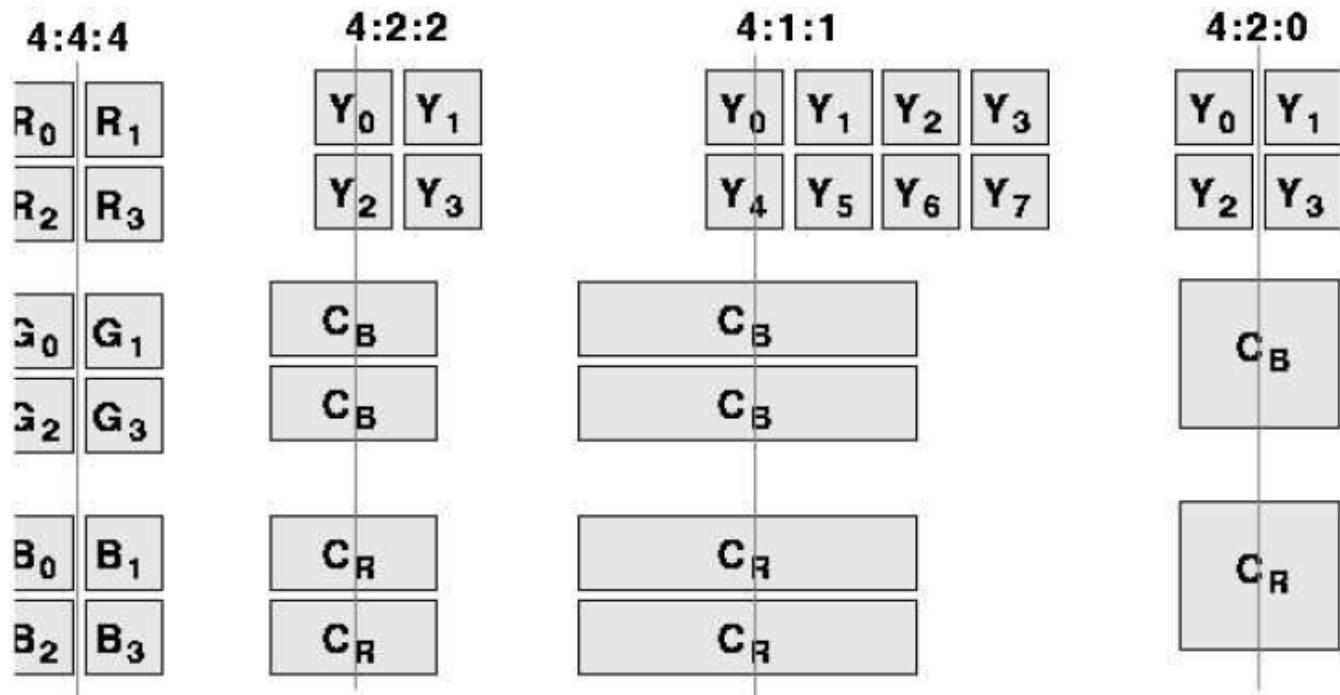
# Chroma Subsampling

- A method that stores an image's (or video frame's) colour information at lower resolution than its intensity information.
  - Main Application: COMPRESSION
  - Used in JPEG Image and MPEG Video Compression
    - One (of two) primary Lossy sources.
- Exploit traits of Human visual system (HVS)
  - HVS more sensitive to variations in brightness than colour.
  - So devote more bandwidth to Y than the color difference components Cr/I and Cb/Q.
    - HVS is less sensitive to the position and motion of color than luminance
    - Bandwidth can be optimised by storing more luminance detail than color detail.
  - Reduction results in almost no perceivable visual difference.

# How to Chroma Subsample?

- Operate on color difference components - the signal is divided into:
  - Luma (Y): the intensity component and
  - Chroma: two color difference components which we subsample in some way to reduce its bandwidth
- How to subsample for chrominance?
  - The subsampling scheme is commonly expressed as a three part ratio (e.g. 4:2:2)
- Chroma Subsample 3 Part Ratio - Each part is respectively:
  1. Luma (Y) or Red (R): Horizontal sampling reference (originally, as a multiple of 3.579 MHz in the NTSC analog television system | rounded to 4)
  2. Cr/I/G: Horizontal factor (relative to first digit)
  3. Cb/Q/B: Horizontal factor (relative to first digit), except when zero.
    - Zero indicates that Cb (Q/B) horizontal factor is equal to second digit, and,
      - Both Cr (I/G) and Cb (Qb) are subsampled 2:1 vertically.

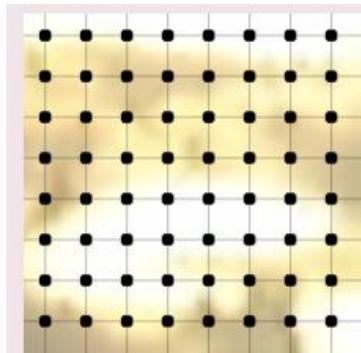
# Chroma Subsampling Examples



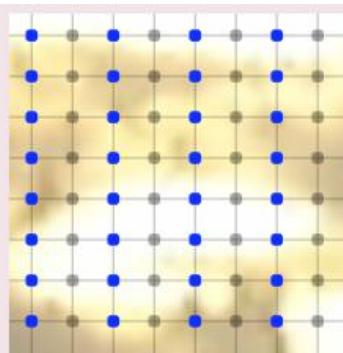
- 4:4:4 — no subsampling in any band — equal ratios.
- 4:2:2 → Two chroma components are sampled at half the sample rate of luma, horizontal chroma resolution halved.
- 4:1:1 → Horizontally subsampled by a factor of 4.
- 4:2:0 → Subsampled by a factor of 2 in both the horizontal and vertical axes

# Chroma Subsampling: How to Compute?

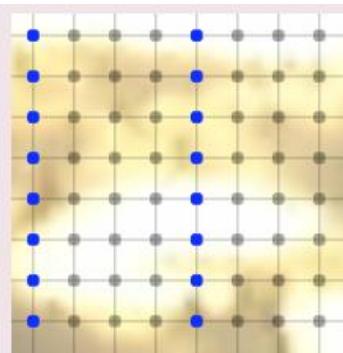
- Simple Image sub-sampling:
    - Simply different frequency sampling of digitised signal
    - Digital Subsampling: For 4:4:4, 4:2:2 and 4:1:1
- Perform 2x2 (or 1x2, or 1x4) chroma subsampling
- Subsample horizontal and, where applicable, vertical directions
  - i.e. Choose every second, fourth pixel value.



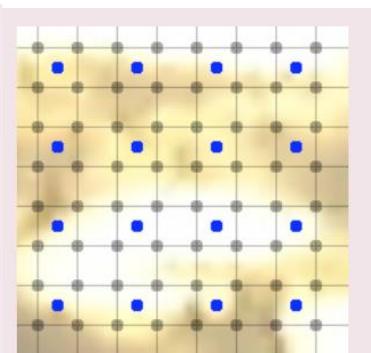
4:4:4



4:2:2



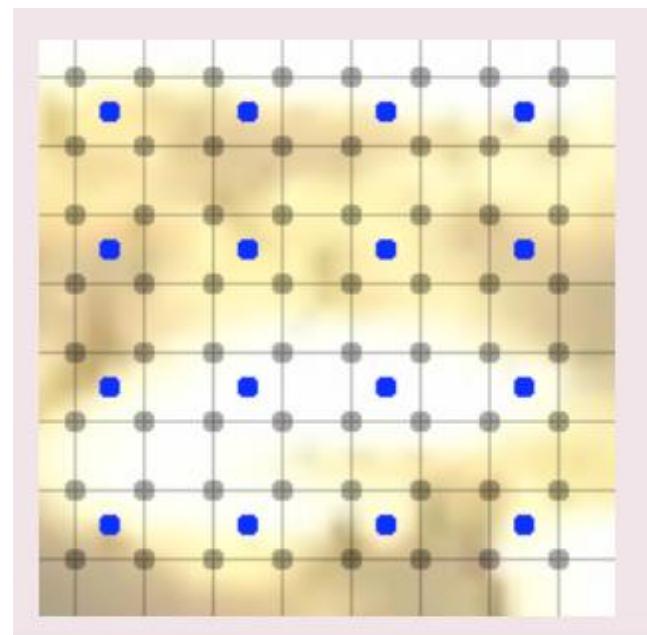
4:1:1



4:2:0

# Chroma Subsampling: How to Compute?

- 4:2:0 Subsampling:
  - For 4:2:0, Cr and Cb are effectively centred vertically halfway between image rows.:
  - Break the image into 2x2 pixel blocks and
  - Stores the average color information for each 2x2 pixel group.



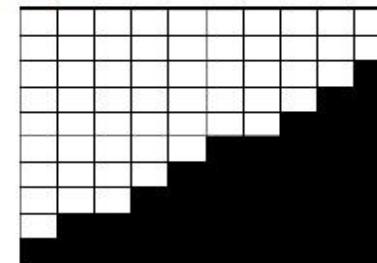
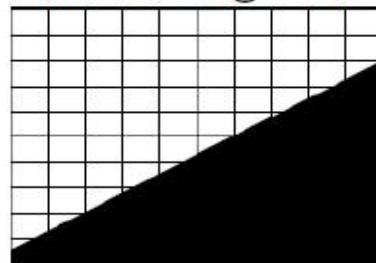
- This sampling process introduces two kinds of errors:
  - A minor problem is that colour is typically stored at only half the horizontal and vertical resolution as the original image - subsampling.
    - This is not a real problem:
      - Recall: The human eye has lower resolving power for colour than for intensity.
      - Nearly all digital cameras have lower resolution for colour than for intensity, so there is no high resolution colour information present in digital camera images.

# Chroma Subsampling Errors - Integer Rounding Errors

- Another issue: The subsampling process demands two conversions of the image:
  - From the original RGB representation to an intensity+colour (YIQ/YUV) representation , and
  - Then back again (YIQ/YUV {>} RGB) when the image is displayed.
  - Conversion is done in integer arithmetic – some round-off error is introduced.
  - This is a much smaller effect,
  - But (slightly) affects the colour of (typically) one or two percent of the pixels in an image.
- Do not compress/recompress videos too often:
  - edit the original!

# Aliasing in Images

- **Stair-stepping:** Stepped or jagged edges of angled lines, e.g., at the slanted edges of letters.



- **Image Zooming:** changing resolution or not acquiring image in adequate resolution, e.g. digital zoom on cameras, digital scanning.  
(see [zoom\\_alias.m](#))



**Explanation:** Simply Application of Nyquist's Sampling Theorem:  
Zooming in by a factor  $n$  divides the sample resolution by  $n$

# Factorul de scalare

---

- de obicei - afişarea pe un dispozitiv cu factorul de scalare 1
- Factor de scalare supraunitar
  - rezultă supraeşantionare
  - pierderea calităţii
- Factor de scalare subunitar
  - subeşantionare
  - unii pixeli lipsesc/sunt eliminati
  - pot dispara detalii
- Trebuie gasite tehnici adecvate
  - pentru supraeşantionare / subeşantionare

# Sisteme avansate de codare și compresie a datelor multimedia

## Curs 3 – Introducere în compresia informațiilor

Sl.Dr.Ing. Camelia FLOREA

Intelligent and multimodal image processing and analysis group (IMIPA),

Communications Departament, ETTI, TUCN,

E-mail: [Camelia.Florea@com.utcluj.ro](mailto:Camelia.Florea@com.utcluj.ro); Phone: +4 0264 401285;

Address: C. Daicoviciu, 15, room 433, Cluj-Napoca, RO.

# Introducere

- **Compresia**
  - reprezentare digitală compactă a informației multimedia
  - cu aplicații în special:
    - în transmisia informației
    - stocarea datelor/imaginilor/secvențelor video
    - aplicații multimedia
- Fără reprezentarea comprimată
  - multe dintre aplicațiile multimedia nu ar putea fi implementate practic.

# Introducere – concepte de bază

## • Compresia

- reducerea cantității de date necesare pentru reprezentarea unei informații

$$C = \frac{b}{b'}$$

$$R = 1 - \frac{1}{C}$$

- C – factorul de compresie

•  $b$  – număr biți reprezentare informație în forma originală

•  $b'$  – număr biți reprezentare informație după compresie

- R - redundanța relativă

## • Exemplificare:

- $C=10$  rezultă  $R=0.9$  adică 90% din b sunt date redundante

- **Simbol, si**
  - o sursă  $S$  de simboluri aleatoare  $s_1, s_2, \dots, s_N$ .
    - $S$  – semnalul - poate fi o imagine digitală
    - $si$  reprezintă una din cele  $N$  valori posibile pe care le poate lua un pixel din imagine

- **Informația,  $I(si)$** 
  - = numărul de biți necesari pentru codarea simbolului  $si$ 
    - dacă baza logaritmului = 2 => informația se măsoară în *biți*.
    - dacă baza logaritmului = 10 => informația se măsoară în *digiti*.

$$I(s_i) = \log \frac{1}{p_i} = -\log p_i$$

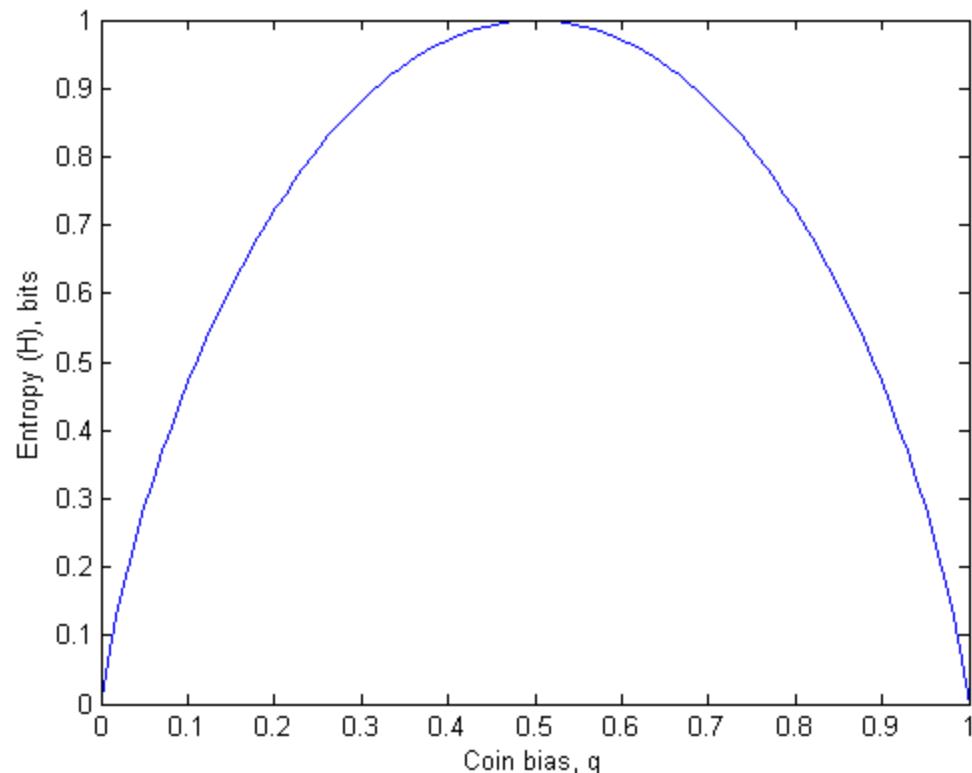
- **Entropia,  $H(S)$** 
  - = informația medie pe simbol ( $\Leftrightarrow$  numărul mediu de biți pe simbol)

$$H(S) = \sum_i p_i \log \frac{1}{p_i}$$

## Example: Entropy of a fair coin.

The coin emits symbols  $s_1 = \text{heads}$  and  $s_2 = \text{tails}$  with  $p_1 = p_2 = 1/2$ . Therefore, the entropy if this source is:

$$\begin{aligned} H(\text{coin}) &= -(1/2 \times \log_2 1/2 + 1/2 \times \log_2 1/2) = \\ &= -(1/2 \times -1 + 1/2 \times -1) = -(-1/2 - 1/2) = 1 \text{ bit.} \end{aligned}$$



## Exemplu

- 8 simboluri -  $s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8$
- $p_i = 1/8$
- entropia = 3
- nu putem găsi un algoritm de codare cu mai puțin de 3 biți/eșantion
- $p_1=p_2=3/12; p_3=p_4=p_5=p_6=p_7=p_8=1/12$
- entropia = 2.79

## Exemplu

- O imagine de 1024x1024 pixeli și 8 biți/pixel are entropia 5.3 biți/pixel.

- Care este raportul de compresie maxim care se poate obține?

$$C = \frac{b}{b'}$$

$$C = 8 / 5.3 = 1.5$$

- Care este redundanța relativă?

$$R = 1 - \frac{1}{C}$$

$$R = 1 - 1 / 1.5 = 0.33$$

=> 33% date redundante

# Redundanțe

## • **Redundanță codării**

- simbolurile de codare sunt reprezentate pe mai mulți biți decât este necesar

## • **Redundanță spațială, temporală și spectrală**

- *Informația se repetă în timp/spațiu*

## • **Redundanță spectrale**

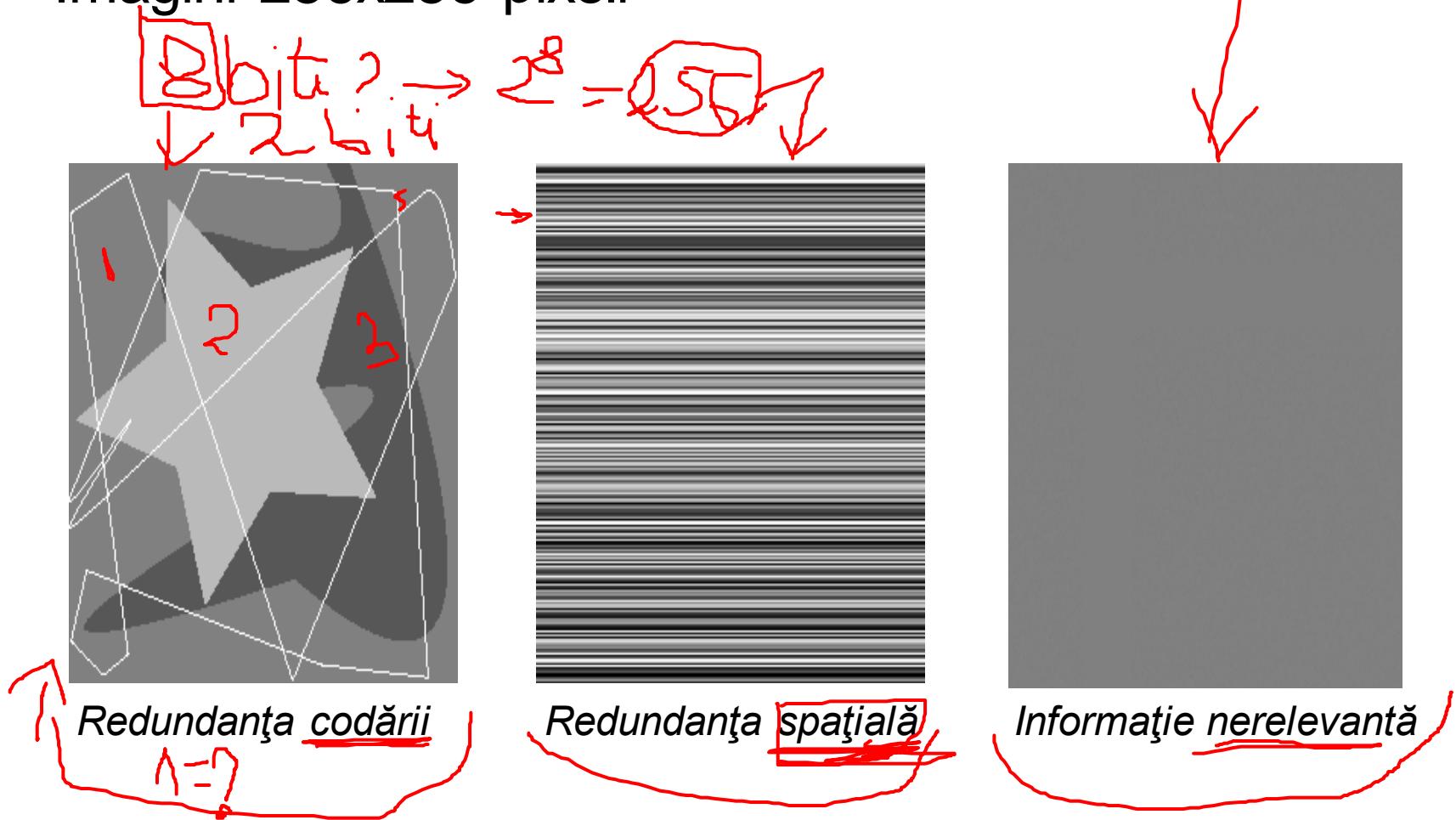
- Exploatază reprezentarea semnalului *în domeniu frecvență*

## • **Informație nerelevantă**

- din punct de vedere al *sistemului vizual uman*

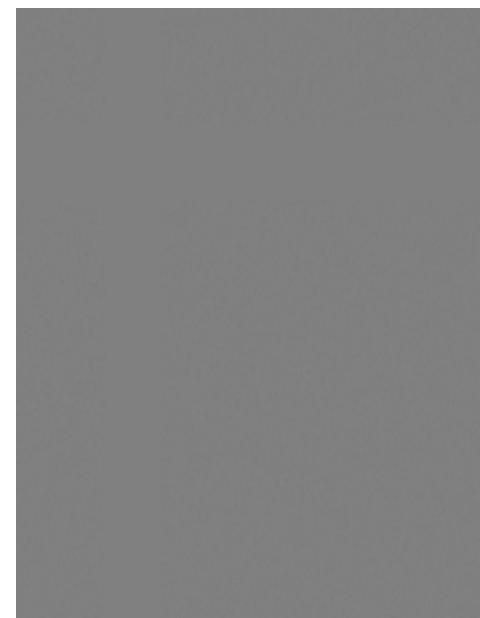
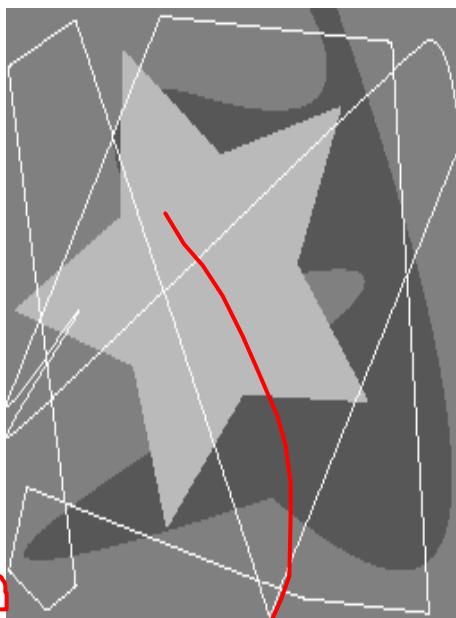
# Introducere – concepte de bază

- Imagini 256x256 pixeli



# Imaginiile de test ...

Imagini 256x256 pixeli



- Imaginea a – entropia = 1.6614 biți/pixel
- Imaginea b – entropia = 8 biți /pixel
- Imaginea c – entropia = 1.566 biți/ pixel
- Concluzie => Entropia și informația **NU sunt intuitive**

# Redundanța codării

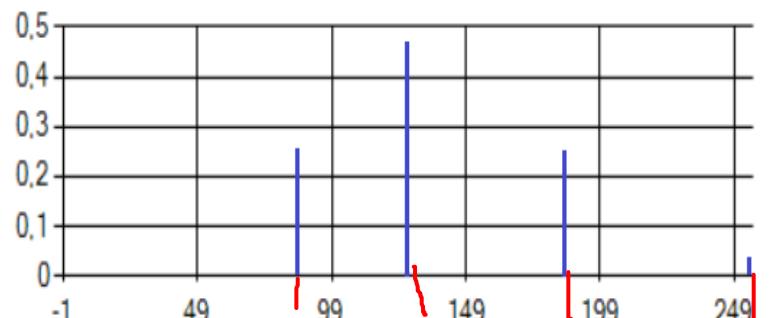
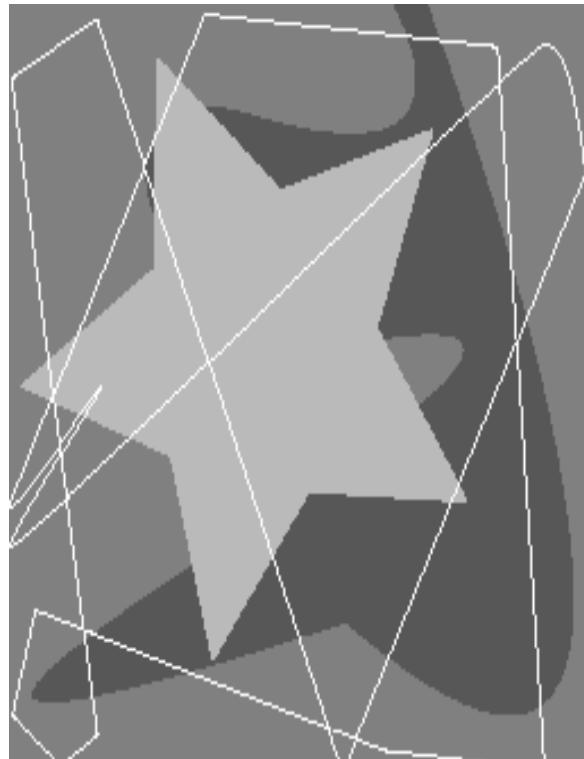
- Codarea optimă a informației pe un număr cât mai mic de biți
- Numărul mediu de biți pe care se poate reprezenta informația:

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

$r_k$  - nivelul de gri k în intervalul [0, L-1]

$p_r(r_k)$  probabilitatea de apariție a nivelului de gri  $r_k$

$l(r_k)$  - numărul de biți necesari pentru reprezentarea nivelul  $r_k$



# Redundanța codării - exemplu

Nivel gri $r_k$	Probabilitatea de apariție $p_r(r_k)$	Code 1	Număr biți reprezentare code 1 $l_1(r_k)$	Code 2	Număr biți reprezentare code 2 $l_2(r_k)$
87	0.25	01010111	8	01	2
128	0.47	10000000	8	1	1
186	0.25	11000100	8	000	3
255	0.03	11111111	8	001	3

- să calculăm  $L_{avg}$  pentru codul 1 și 2
  - asignarea codurilor scurte la simbolurile cu probabilitate mare de apariție
- => **coduri de lungime variabilă**

$$L_{avg} = 8 \cdot 0,25 + 8 \cdot 0,47 + 8 \cdot 0,25 + 8 \cdot 0,03$$

$$= 8$$

# Redundanță codării

Numărul total de biți necesari pentru reprezentarea unei imagini

$$b' = M \times N \times L_{avg} = 256 \times 256 \times 1.81$$

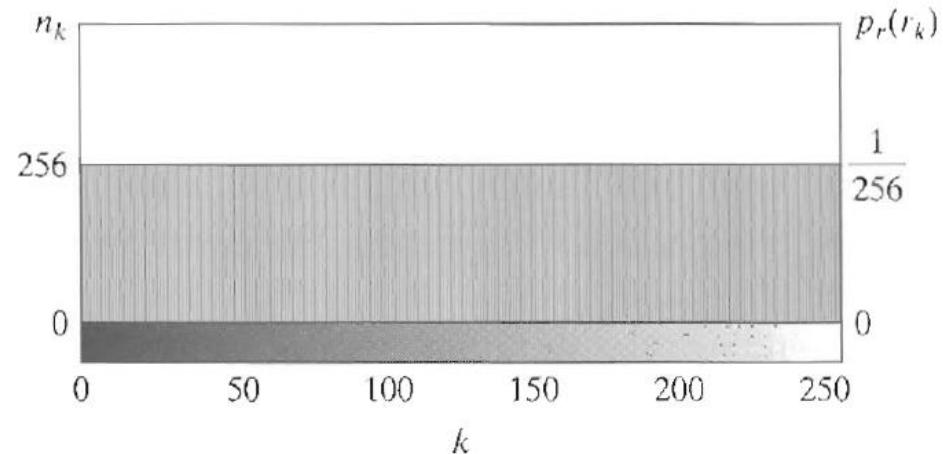
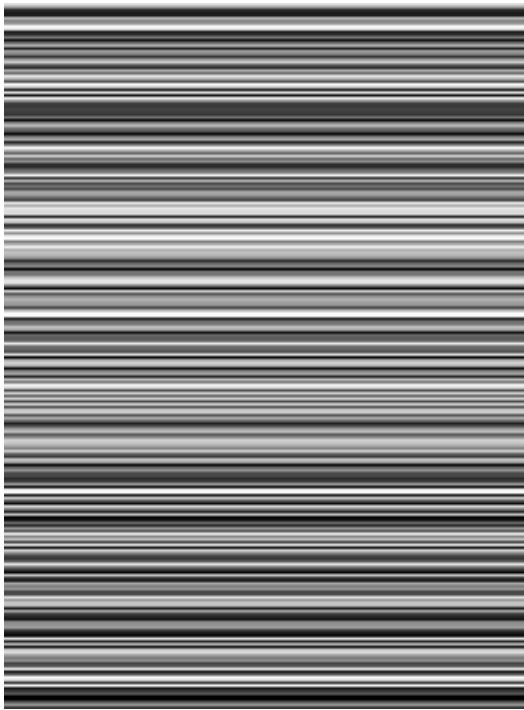
$$b = M \times N \times L_{avg} = 256 \times 256 \times 8$$

$$C = \frac{b}{b'} = 4.42$$

$$R = 1 - \frac{1}{C} = 1 - \frac{1}{4.42} = 0.774$$
 adică 77.4% este informatie redundantă

# Redundanță spațială, temporală, spectrală

- Toate intensitățile au probabilități egale



- Intensitatea fiecărei linii este selectată aleator – pixelii sunt independenți pe verticală
- Pixelii dintr-o linie sunt identici – maxim corelați, dependenți unul de altul pe orizontală

# Redundanță spațială, temporală

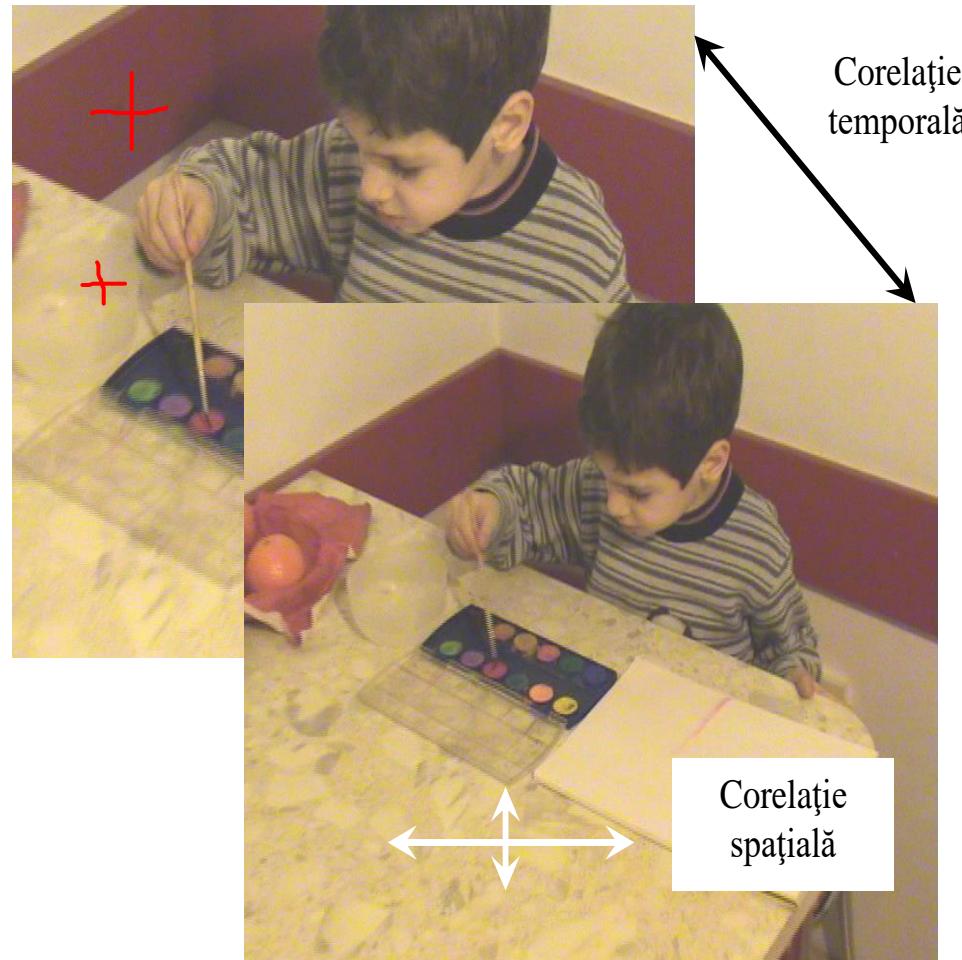
- Observații

- NU se pot folosi codurile de lungime variabilă – cele 256 simboluri rk au aceeași probabilitate
- Se poate coda lungimea curselor
- Pereche de codare
  - Start nouă intensitate
  - Lungimea intensității
- Să vedem un exemplu.....
  - Fiecare linie se înlocuiește cu
    - Valoare intensitate
    - Lungimea – care pentru exemplul nostru este 256

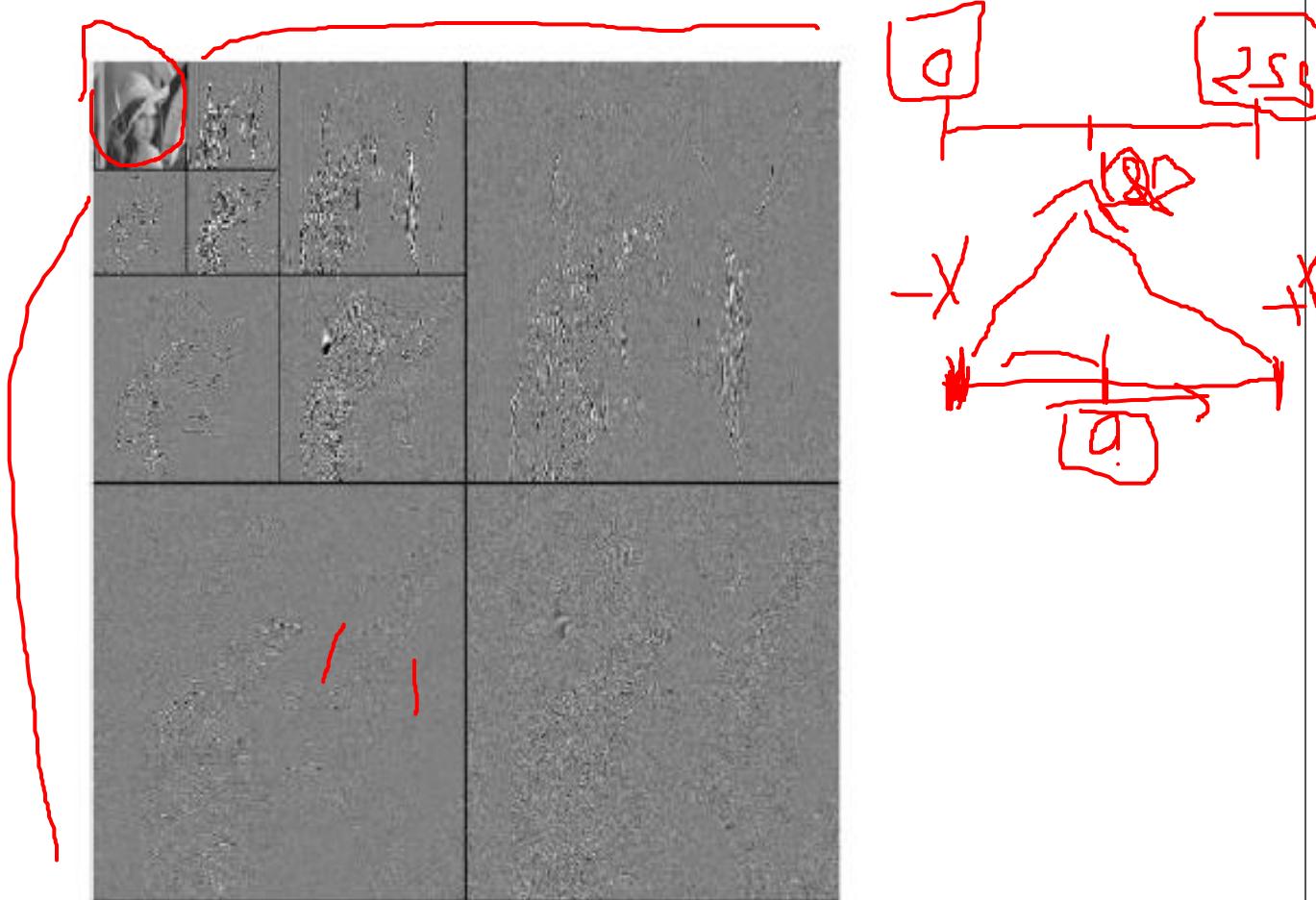
(10256), (00256) 256

# Redundanță spațială, temporală, spectrală

- Corelația **spațială** – ambele direcții
- Corelație **temporală**
  - Cadre succesive



# Redundanță spectrală



# Informație irelevantă

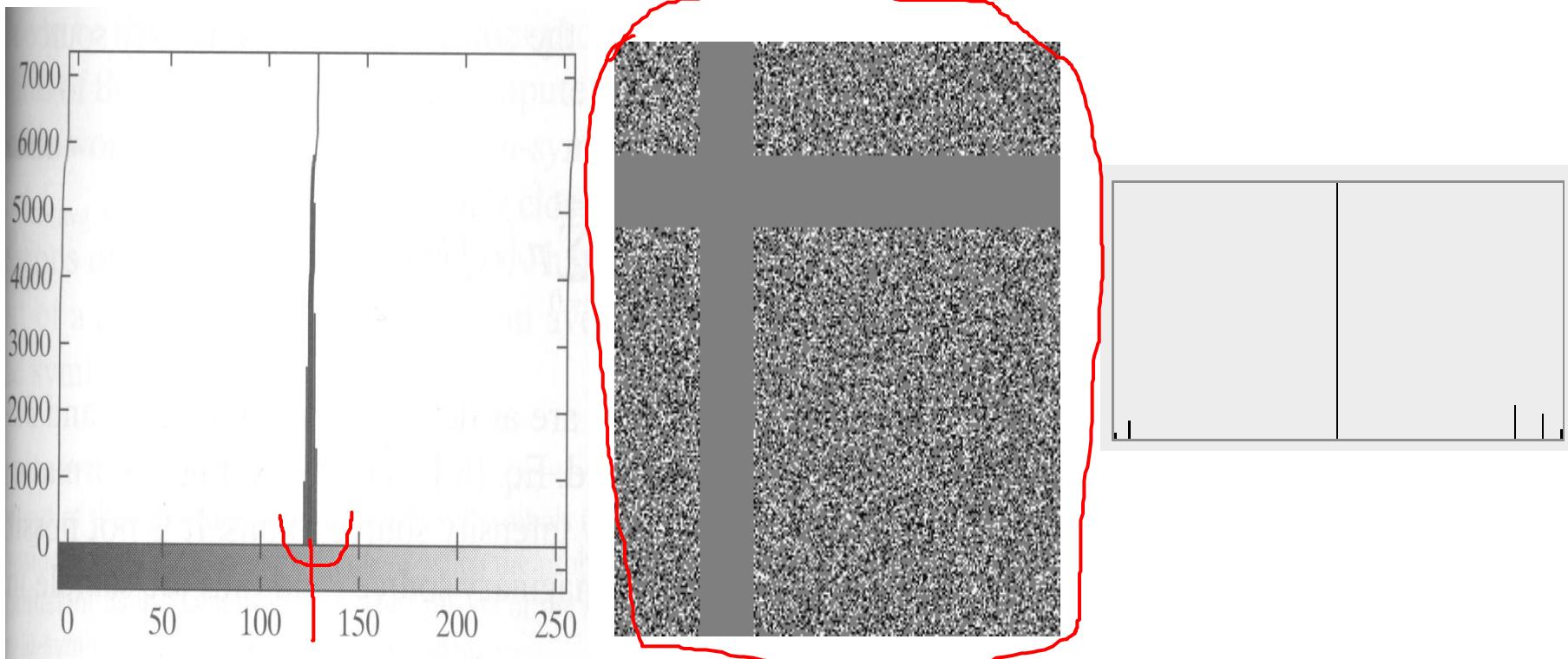
- Reprezentarea intensității medie (intensitățile sunt foarte apropiate și ochiul nu le poate distinge)
- Un singur octet
- Compresie

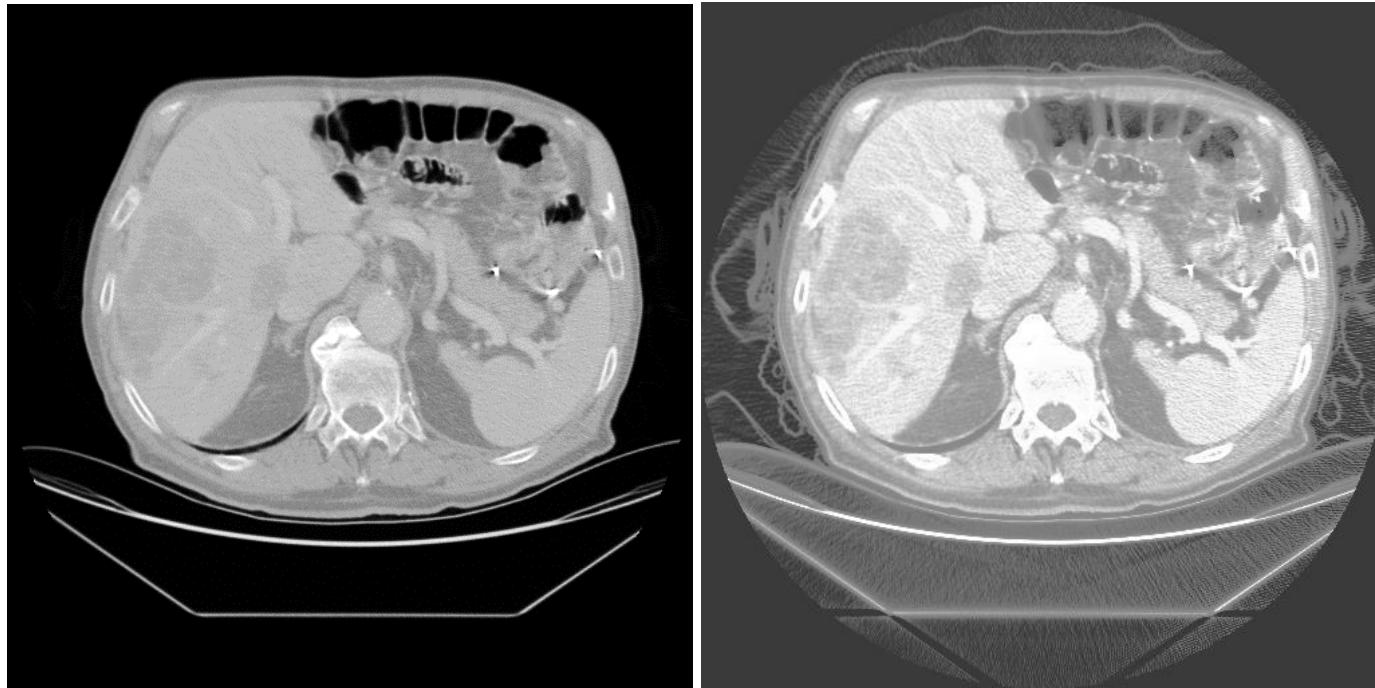
$$C = \frac{b}{b'} = \frac{256 \times 256 \times 8}{8} = 65536$$

- La decodare imaginea refăcută nu va avea de suferit foarte mult

# Informație irelevantă

- Totuși imaginea dacă se face egalizare nu are intensitate uniformă
- Imaginea initială nivele de gri – 125-131, după egalizare apar informații relevante și pentru sistemul vizual
- Depinde de tipul aplicației... de ex. **imaginile medicale!!!**





# Criterii de calitate

- **Obiective**

- Funcții matematice
- Evaluare simplă

$$MAD(i,j) = \frac{1}{mn} \sum_i^n \sum_j^m |F(i,j) - G(i,j)|$$

MAD - medie absolută a diferenței

- **Subiective**

- Scală de evaluare:

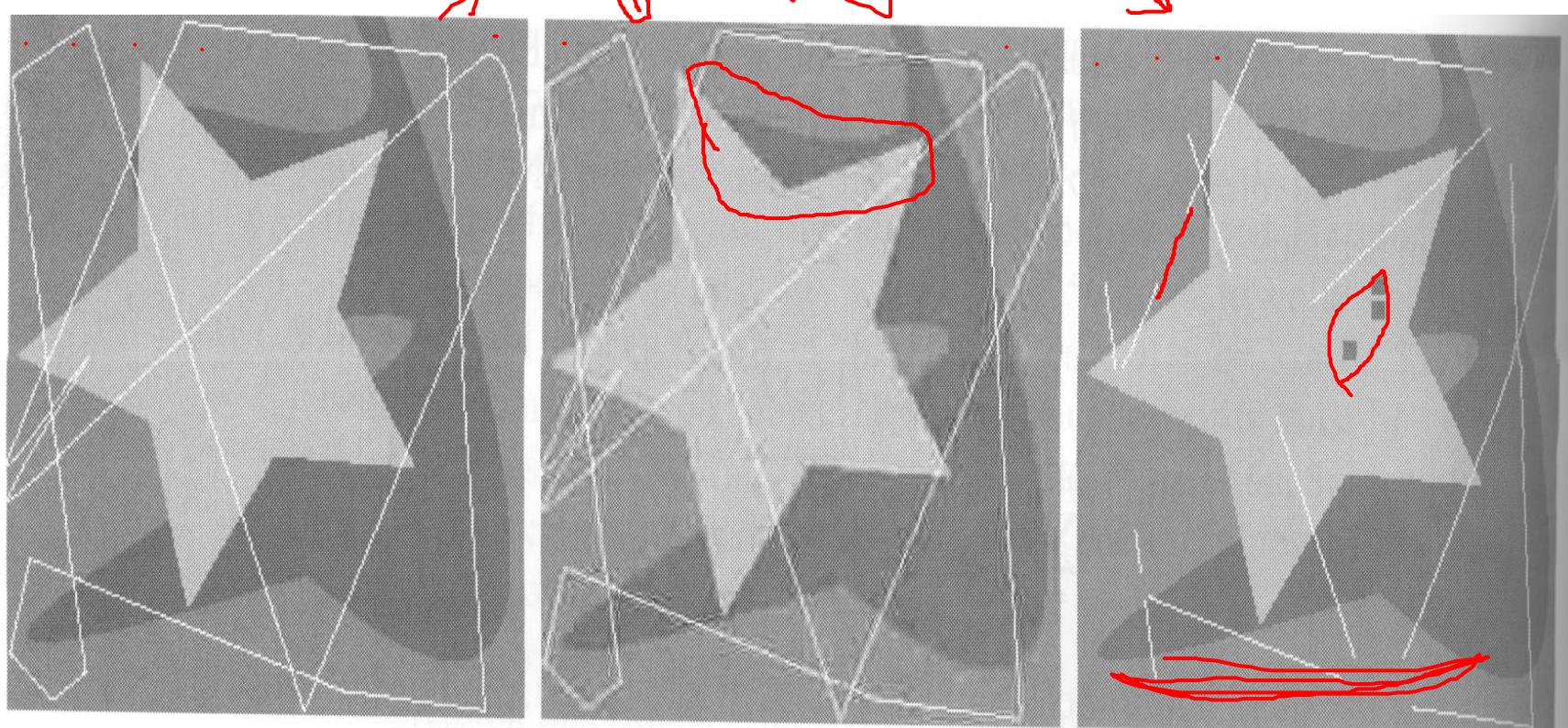
- excelent, bună, acceptabilă, satisfăcătoare, inferioară, inacceptabilă

PSNR

$$MSE(U, \hat{U}) = \frac{1}{M \cdot N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (u(m,n) - \hat{u}(m,n))^2.$$

# Criterii de calitate - exemplu

- 1-original, 2-decomprimată, 3-generată artif.
- ATENTIE – 1, 3 prin criteriul obiectiv, RSZ da o valoare f.bună, DAR prin criteriul subiectiv NU este aceeași imagine!!! ↗ 1 ↘ 3



# Sisteme avansate de codare și compresie a datelor multimedia (SACCDMM)

**- Curs 4 -**

## **Algoritmi de compresie cu/fără pierderi**

# Compresie CU sau FĂRĂ pierderi

Compression can be categorised in two broad ways:

**Lossless Compression:** after decompression gives an exact copy of the original data.

**Example:** Entropy encoding schemes (Shannon-Fano, Huffman coding), arithmetic coding, LZW algorithm (used in GIF image file format).

**Lossy Compression:** after decompression gives ideally a “close” approximation of the original data, ideally perceptually lossless.

**Example:** Transform coding — FFT/DCT based quantisation used in JPEG/MPEG differential encoding, vector quantisation.

# De ce Compresie cu PIERDERI

- Lossy methods are **typically** applied to high resolution audio, image compression.
- Have to be employed in video compression (apart from special cases).

Basic reason:

- **Compression ratio** of lossless methods (e.g. Huffman coding, arithmetic coding, LZW) is not high enough for audio/video.
- By cleverly making a **small** sacrifice in terms of fidelity of data, we can often achieve **very high** compression ratios.
  - Cleverly = sacrifice information that is psycho-physically unimportant.

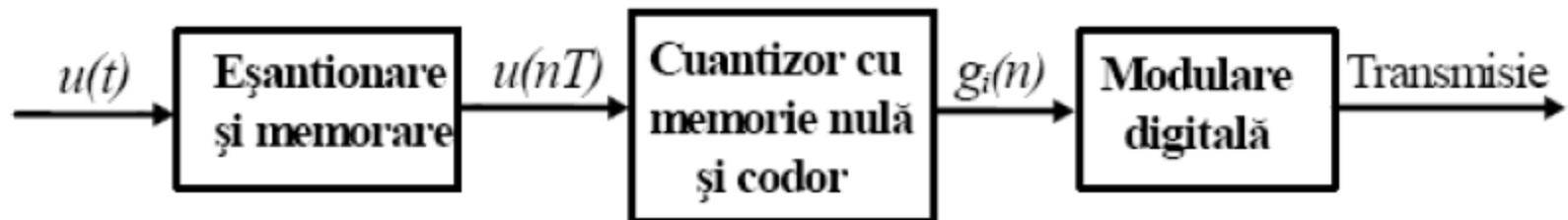
# Algoritmi de compresie fără pierderi

- Semantic-based coding/source coding
  - Repetitive Sequence Suppression.
  - Run-Length Encoding (RLE).
  - Pattern Substitution.
  - Entropy Encoding:
    - Shannon-Fano Algorithm
    - Human Coding
    - Arithmetic Coding
  - Lempel-Ziv-Welch (LZW) Algorithm.
- Some broad methods that exist:
  - Differential Encoding
  - Vector Quantisation
  - Transform Coding

# Algoritmi de compresie fără pierderi

- setul original de date poate fi refăcut complet
- există aplicații unde NU se acceptă pierderi
  - Imagini medicale
  - Imagini **bitonale**
    - desene, scrisori, ziar, hărți și alte documente

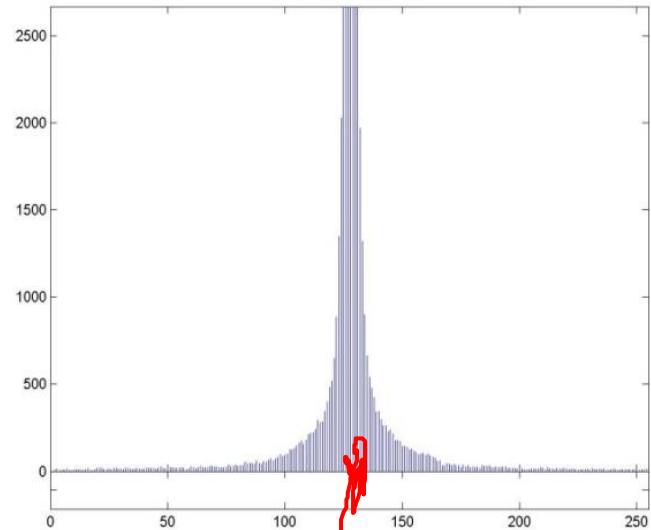
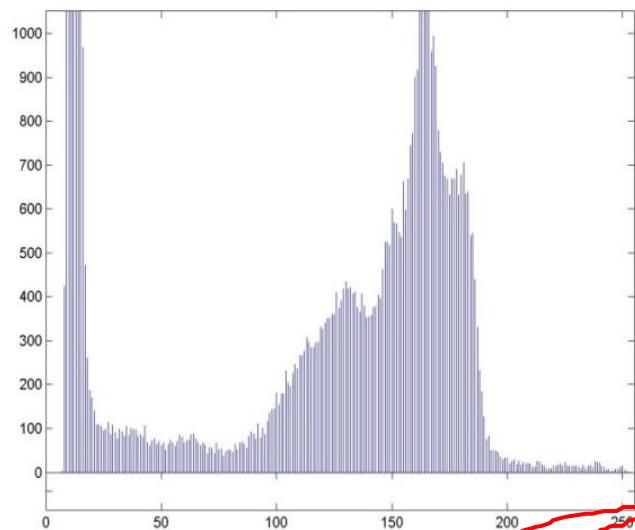
# Modularea impulsului în cod (PCM)



- Semnalul este eșantionat, cuantizat și codat
- Ieșirea cuantizorului este codată cu cuvinte binare de lungime fixă (B biți, tipic 8 biți/cuvant)

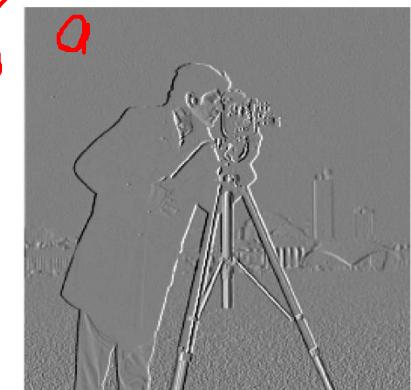
# Codarea diferențială

OPCM



- codarea diferențială
  - îmbunătățirea ratei de compresie
  - algoritm de preprocesare

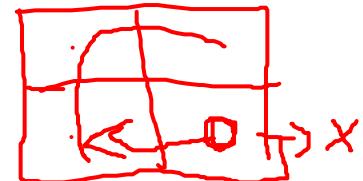
= modificarea ratei de apariție simbolurilor astfel încât să se obțină distribuție mai eficientă pentru codare



# Codarea predictivă fără pierderi

- Codare

$$e(n) = f(n) - \hat{f}(n)$$



- Decodare

$$f(n) = e(n) + \hat{f}(n)$$

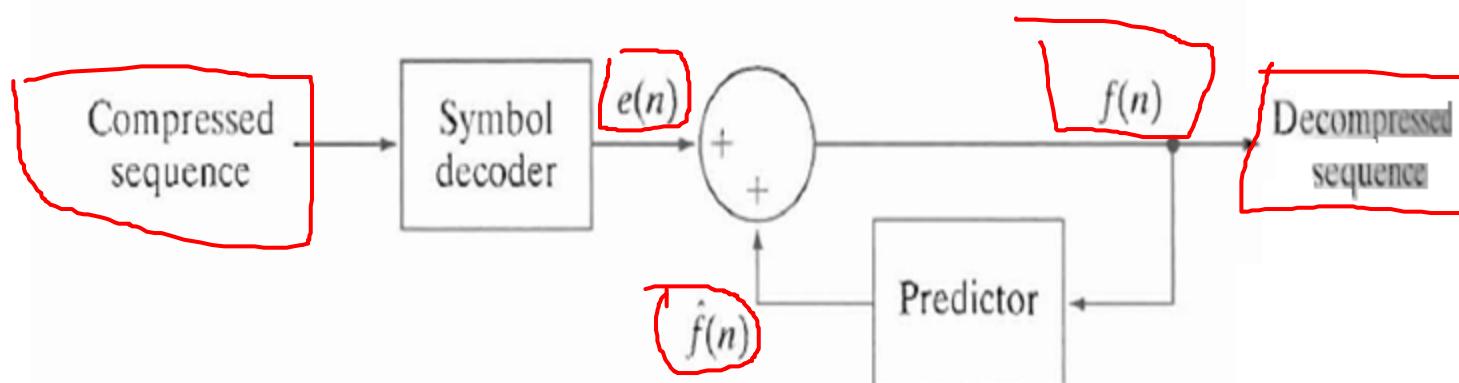
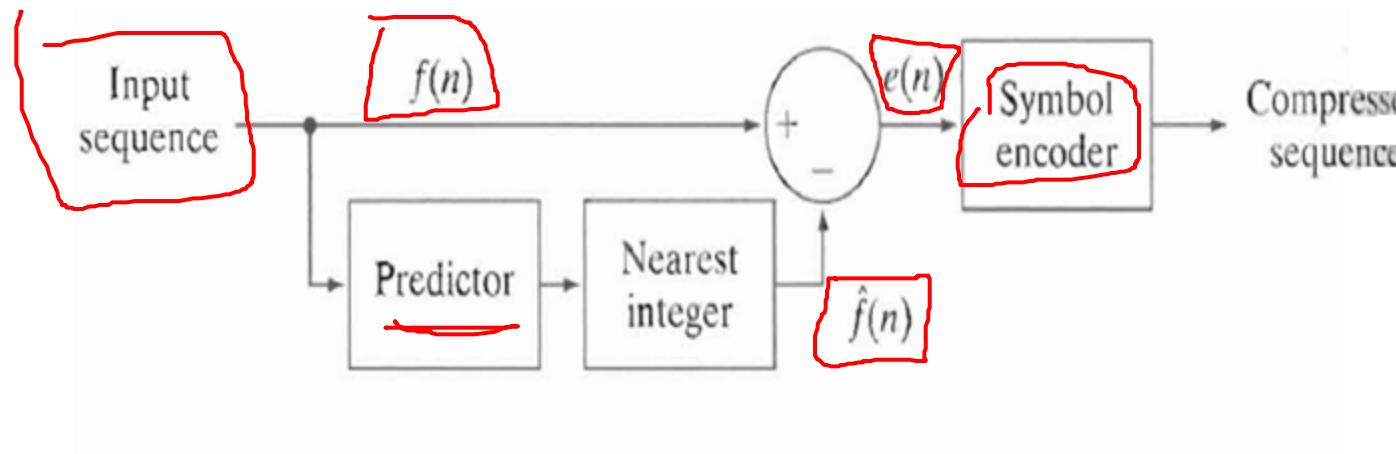
- Predictorul – combinație liniară a  $m$  eșantioane anterioare

- $m$  – ordinul predictorului
- ROUND – cel mai apropiat întreg
- $\alpha_i$  – coeficienții de predicție

$$\hat{f}(n) = \text{round} \left[ \sum_{i=1}^m \alpha_i f(n-i) \right]$$

# Codarea predictivă fără pierderi

- Codorul și decodorul – același *predictor*



# Codarea predictivă fără pierderi

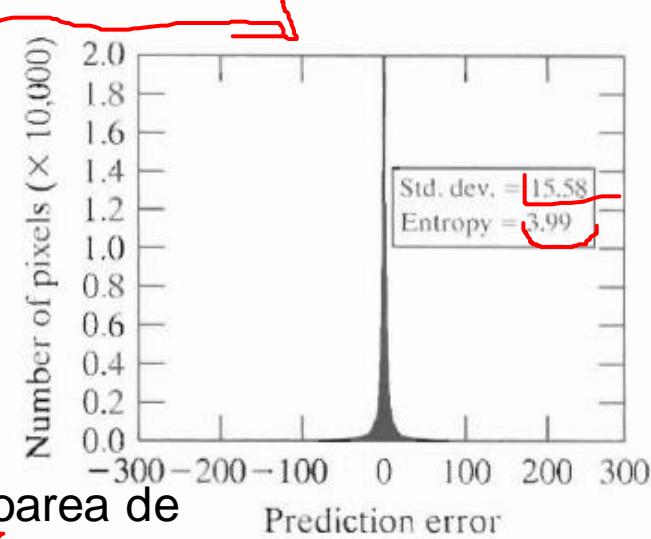
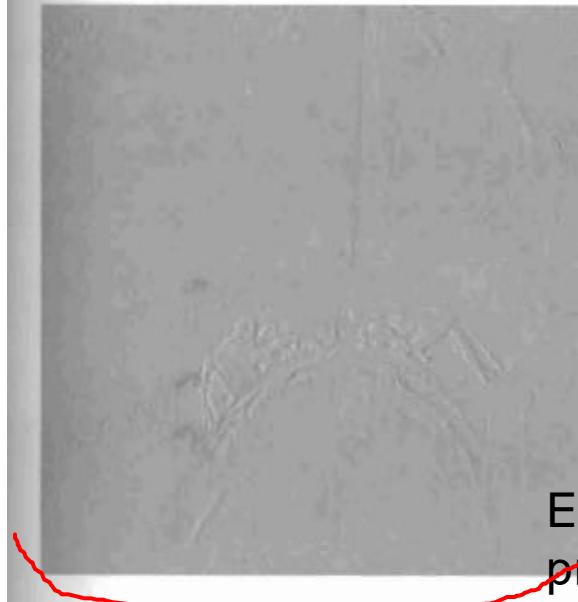
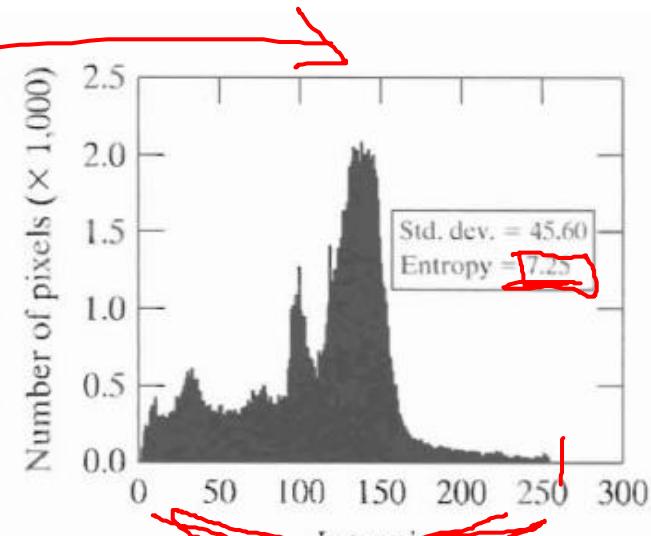
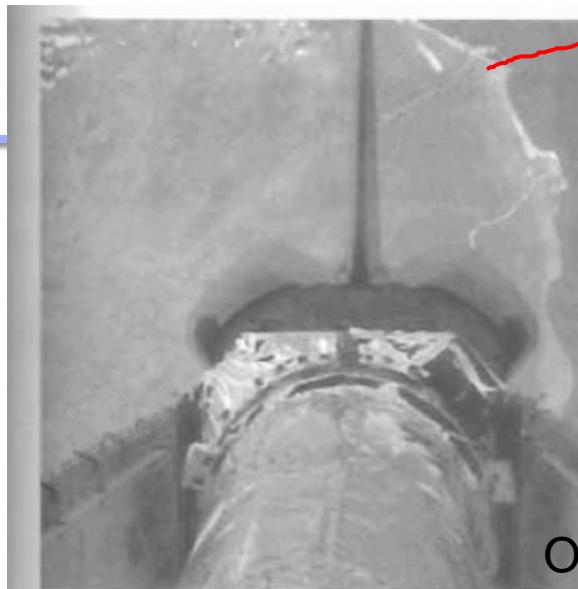
- 1-D – linia curentă
- 2-D – linia curentă și anterioară
- 3-D – imaginea curentă și imaginea anterioară
- Pentru imagini  $(x, y)$ 
  - Dacă
    - $m=1, \alpha=1$
    - Codarea diferențială

$$\hat{f}(x, y) = \text{round} \left[ \sum_{i=1}^m \alpha_i f(x, y - i) \right]$$

$$\hat{f}(x, y) = \text{round} \left[ \alpha f(x, y - 1) \right]$$

# 1-D

- Imaginea eroarea de predictie
  - scalare pe 128
- Se reduce redundanta spatiala
- Std. dev.
  - 45.6 la original
  - 15.58 la eroarea de predictie
- Se poate obtine un  $C = 8/3.99 = \text{aprox. } 2$



$$m^i = 1 \rightarrow d_i = 1$$

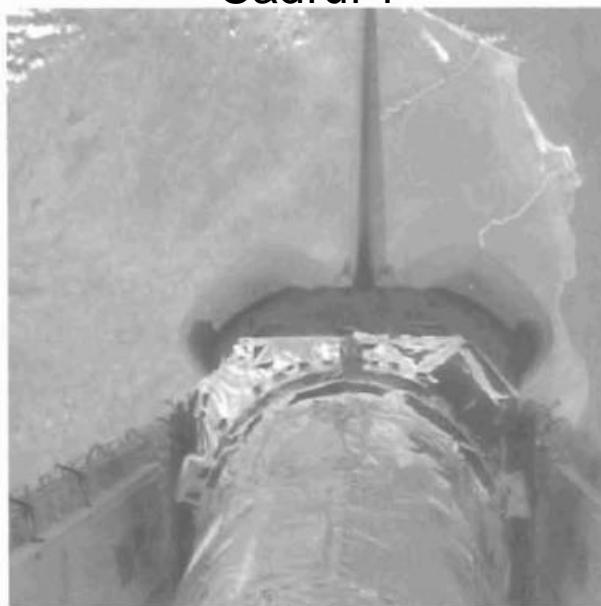
# 3-D

- Scalare pe 128
- Std. dev. – 3.76
- Entropia 2.59
- Se poate obține un  $C=8/2.59 = \text{aprox. } 3.1:1$

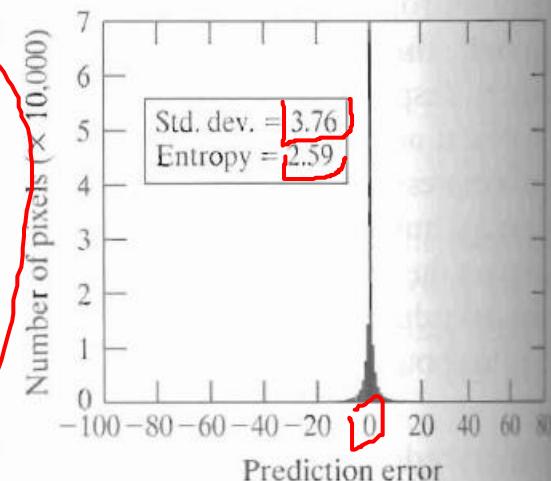
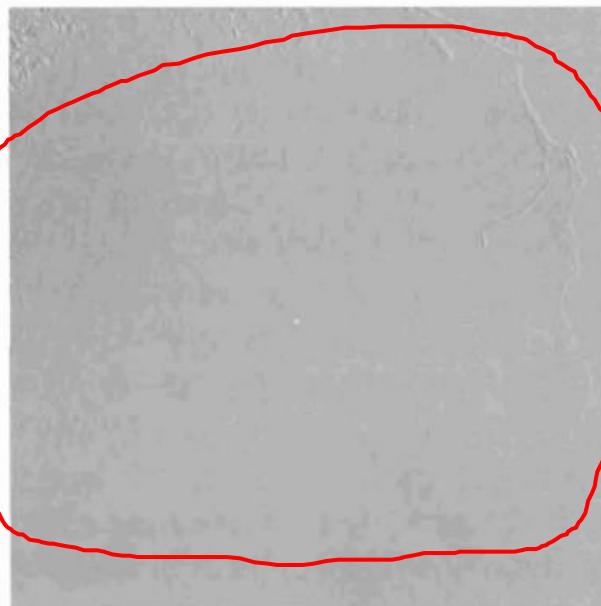
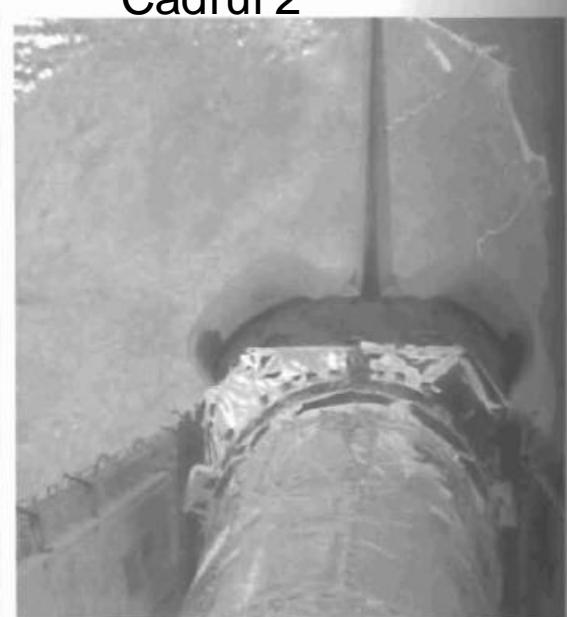
$$\hat{f}(x, y, t) = \text{round}[\alpha f(x, y, t - 1)]$$

$$e(x, y, \tilde{t}) = |f(x, y, \tilde{t}) - \hat{f}(x, y, t)|$$

Cadrul 1



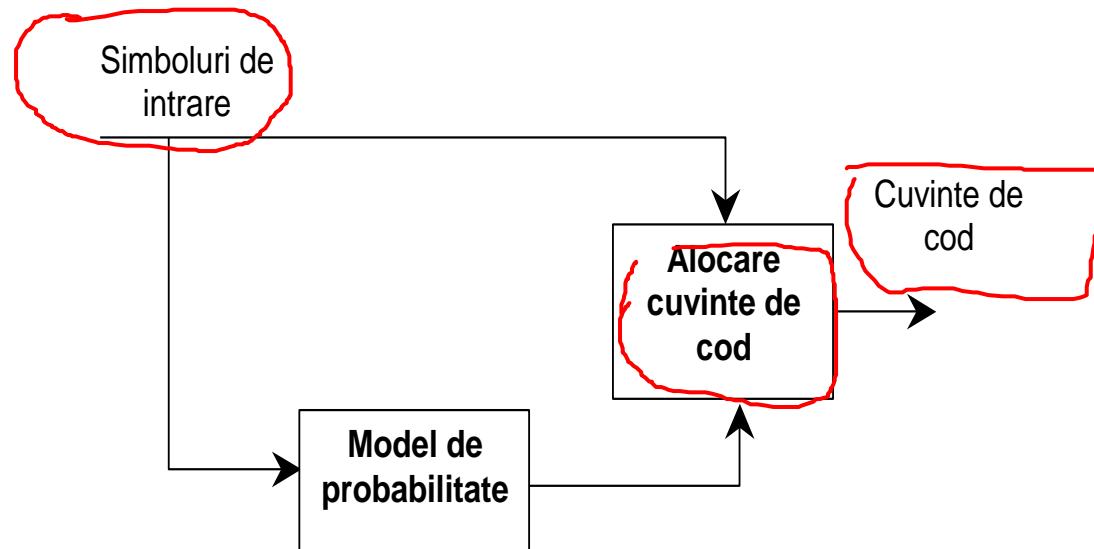
Cadrul 2



Eroare de predicție

# Arhitectura codorului entropic

- Împărțirea mesajului în simboluri
  - $256 \times 256$  pixeli = 1 simbol lungime 64000 pixeli
  - $256 \times 256$  pixeli =  $256 \times 256$  simboluri între 0 – 255



# Codarea Huffman

- **simboluri**

- probabilitate mare – cod lungime mica
- probabilitate mica – cod de lungime mai mare
- tabel coduri Huffman – lungime  $L$

Original source		Source reduction			
Symbol	Probability	1	2	3	4
$a_2$	0.4	0.4	0.4	0.4	0.6
$a_6$	0.3	0.3	0.3	0.3	0.4
$a_1$	0.1	0.1	0.2	0.3	0.1
$a_4$	0.1	0.1	0.1	0.1	0.1
$a_3$	0.06	0.1	0.1	0.1	0.1
$a_5$	0.04	0.1	0.1	0.1	0.1

# Codarea Huffman

Symbol	Original source		Source reduction			
	Probability	Code	1	2	3	4
$a_1$	0.4	1	0.4	1	0.4	0.6
$a_6$	0.3	00	0.3	00	0.3	0.4
$a_1$	0.1	011	0.1	011	0.3	1
$a_4$	0.1	0100	0.1	0100	0.2	0.1
$a_3$	0.06	01010	0.1	0101	0.1	0.1
$a_5$	0.04	01011			0.3	0.1

- Entropia sursei este 2.14 biți/pixel

$$H(S) = -0.4 \cdot \log_2(0.4) - 0.3 \cdot \log_2(0.3) - 0.1 \cdot \log_2(0.1) - 0.1 \cdot \log_2(0.1) - 0.06 \cdot \log_2(0.06) - 0.04 \cdot \log_2(0.04) = 2.14$$

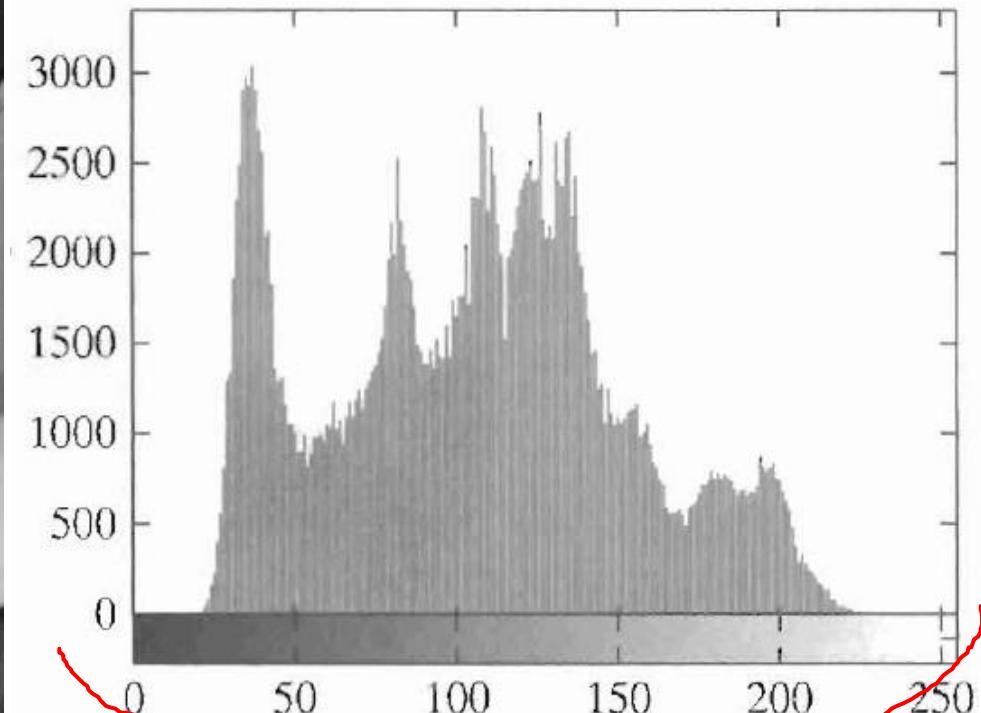
- Evident **cod unic**: 010100111100 –  $a_3\ a_1\ a_2\ a_2\ a_6$

- Lungimea medie pe simbol = 2.2 bits/pixel

$$\begin{aligned}
 L_{\text{avg}} &= (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5) \\
 &= 2.2 \text{ bits/pixel}
 \end{aligned}$$

# Codarea Huffman

- 515x515x8 biți/pixel



# Codarea Huffman

- Entropia = 7.3838 biți/pixel
- Lavg = 7.428 biți/pixel (Huffman - Matlab)
- Diferența Lavg-Entropia
  - este de  $512 \times 512 \times (7.428 - 7.3838)$ ,  
=> adică 11587 biți - 0.6%
- $C = 8 / 7.428 = 1.077$
- $R = 1 - 1 / 1.077 = 0.0715$  (7.15% se reduce prin codare)

# Codarea Huffman

## • cod Huffman trunchiat

- se alege  $L_1 < L$ ,  $L_1 = 56$
- primele  $L_1$  simboluri se codeaza Huffman
- celelalte simboluri – cod\_prefix + cod lungime fixa

## • cod Huffman modificat

$$L_1 = 50$$

- se alege  $L_1 < L$  ;
- primele  $L_1$  simboluri se codeaza Huffman
- celelalte simboluri – cod\_prefix (catul  $q$ ) + cod terminator (codul Huffman al restului  $j$ )

$$i = qL_1 + j, \quad 0 \leq q \leq \text{Int}\left[\frac{(L-1)}{L_1}\right],$$

$$0 \leq j \leq L_1 - 1$$

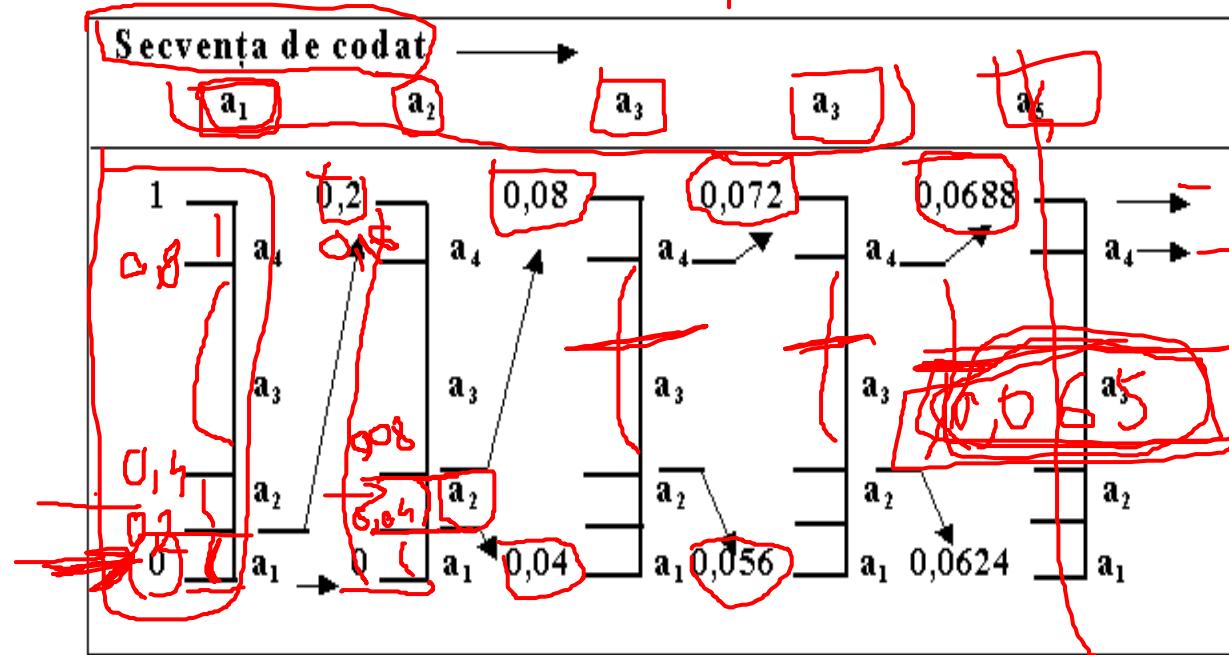
$$i = 52 = 1 \cdot 50 + 2$$

# Codarea aritmetică

- nu există corespondență între simbolurile sursei și cuvintele de cod
- cuvant de cod = interval de numere reale din  $(0,1)$
- cu cat numarul de simboluri din mesaj creste -> scade intervalul de reprezentare
- fiecare simbol reduce intervalul conform probabilitatii de aparitie

# Codarea aritmetică

Simbol sursă	Probabilitate	Interval inițial
$a_1$	0,2	$[0,0 \div 0,2)$
$a_2$	0,2	$[0,2 \div 0,4)$
$a_3$	0,4	$[0,4 \div 0,8)$
$a_4$	0,2	$[0,8 \div 1,0)$



# Codarea RLC

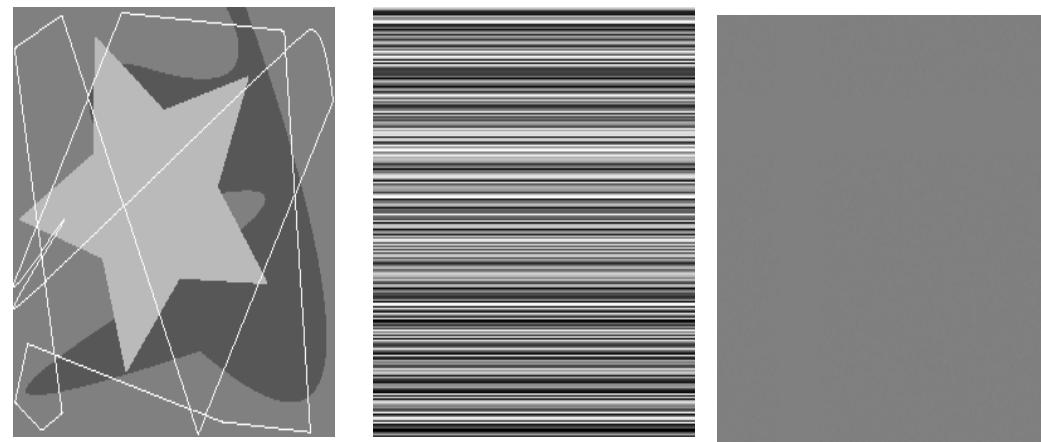
- Imagini în care se repetă intensitatea pe rânduri/coloane
- surse binare de simboluri cu 0 sau 1
- perechi de lungimi de curse – start nouă intensitate, lungime
- grafice, documente tiparite, imagini binare ( $p(0)$  – mare aproape de 1)
- 1000000100000000111110000000
- (11)60(11)80(51)70

# Codarea RLC

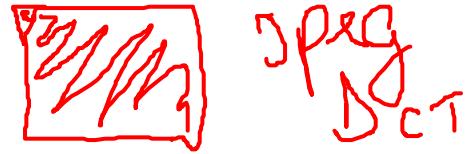
- BMP – fără codare și cu codarea RLC

- Figura a
  - BMP-fără codare- 263.244 bytes
  - BMP-cu codare- 267.706 bytes
  - Mai mare cu codare!!! ~~C=0.38~~

- Figura c
  - $C=1.35$



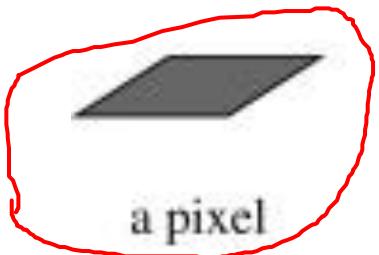
# Codarea RLC fingerprint



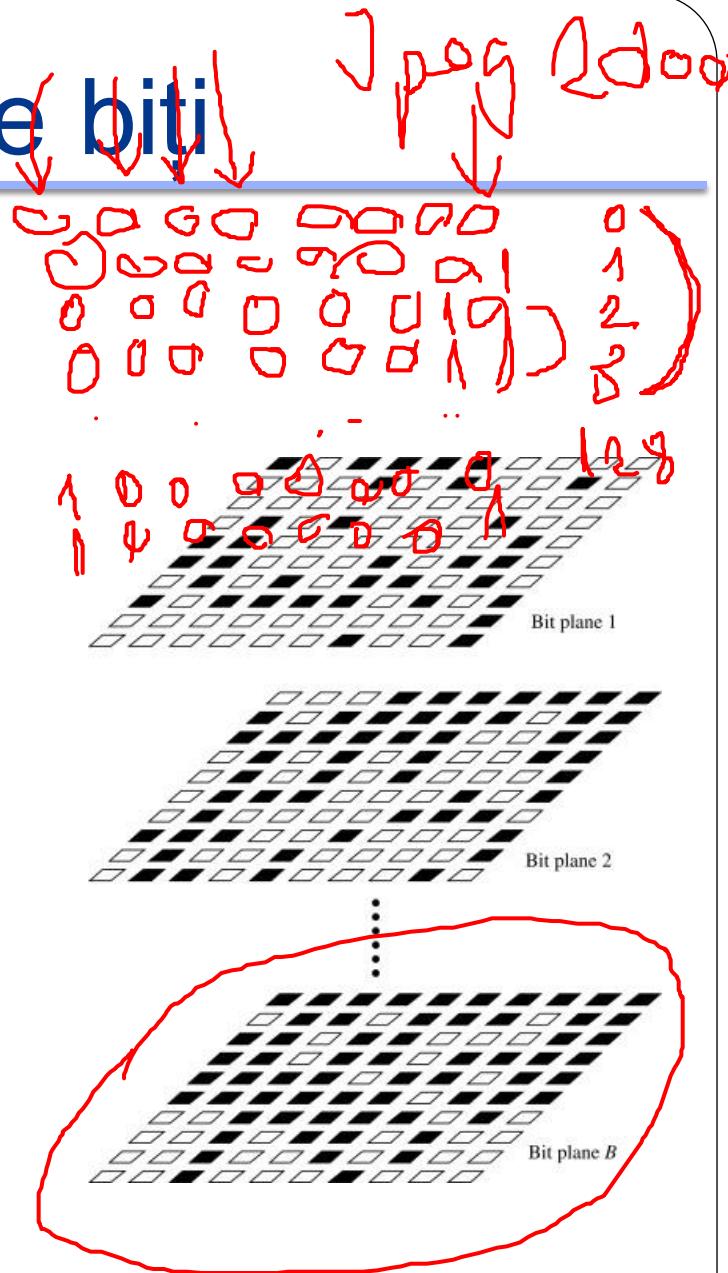
- numar mic de coeficienti
- dupa DCT, Wavelet etc.
- 107-254 – coef. Nenuli
- daca  $|coef| > 73$  se transmite cod escape urmat de valoarea coef.
- 1-100 curse de zero
- daca cursa de zero  $> 100$ , se transmit 105, 106 si valoarea cursei de zero

Simbol	Valoare
1	cursă de zero de lungime 1
2	cursă de zero de lungime 2
...	...
100	cursă de zero de lungime 100
101	simbol escape pentru index pozitiv 8 biți
102	simbol escape pentru index negativ 8 biți
103	simbol escape pentru index pozitiv 16 biți
104	simbol escape pentru index negativ 16 biți
105	simbol escape pentru cursă de zero 8 biți
106	simbol escape pentru cursă de zero 16 biți
107	valoare coeficient -73
108	valoare coeficient -72
...	...
180	neutilizat; se utilizează simbolul 1
...	...
253	valoare coeficient 72
254	valoare coeficient 73

# Codarea planurilor de biți



- O imagine cu 256 nivele de gri, codată cu 8 biți/pixel, poate fi considerată ca un set de 8 plane de 1 bit (planul MSB(0)... planul LSB (7)) și fiecare dintre acestea este codată RLC
- Metoda poate atinge compresii de până la 1.5-2, dar este sensibilă la erorile canalului, dacă cele mai semnificative plane de biți nu sunt protejate corespunzător



# Algoritmi de codare cu pierderi

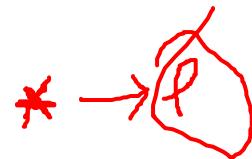
- **codarea predictivă**

- codarea predictivă cu pierderi
- modulația Delta

- **codarea pe blocuri de pixeli**

- alg. de codare spatială pe blocuri
- codarea prin transformări

# Codarea predictivă

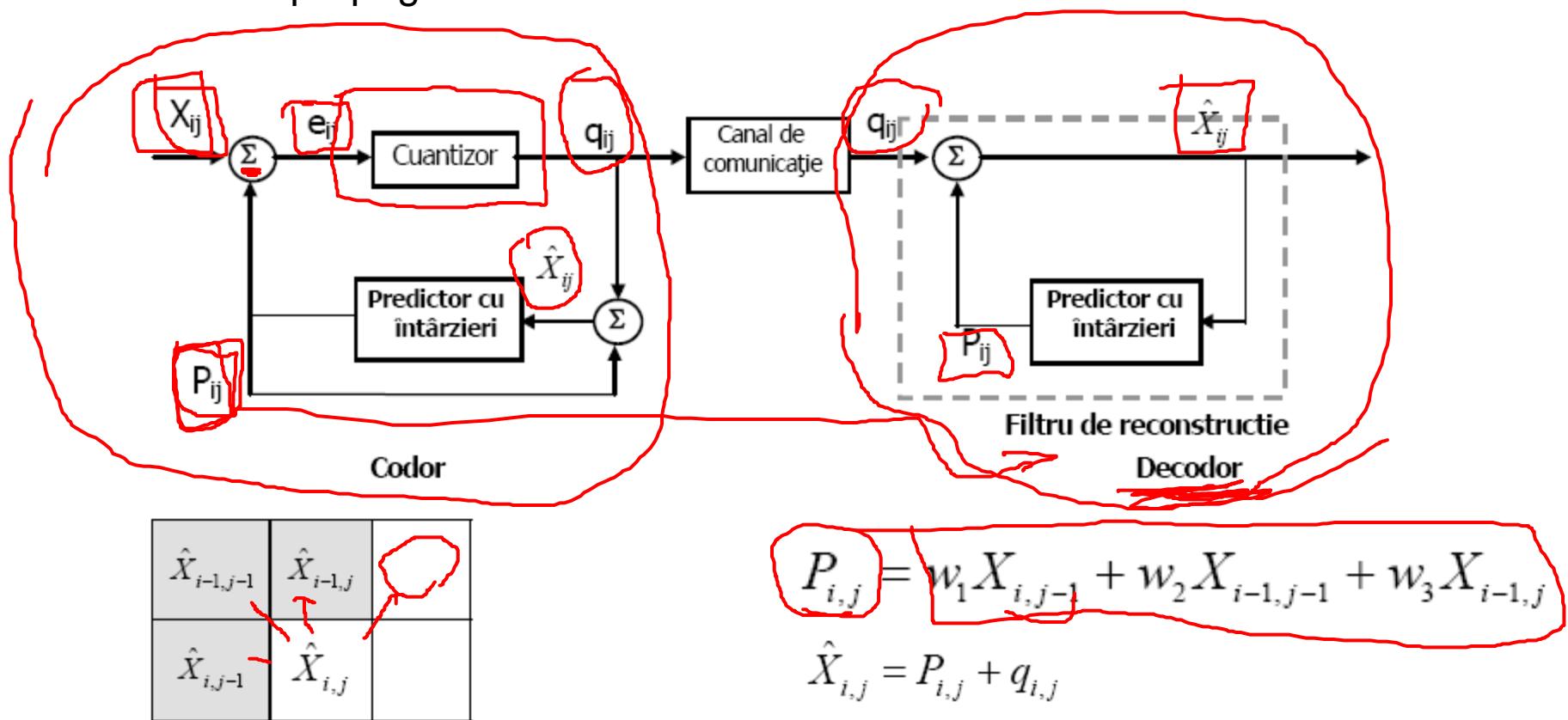


- Eșantioanele care sunt luate în considerare în cazul codării predictive pot fi fie
  - din domeniul spațial
  - fie din domeniu frecvență.
- Unul dintre cei mai simplii algoritmi de codare predictivă este:
  - codarea DPCM – Differential Pulse Code Modulation***



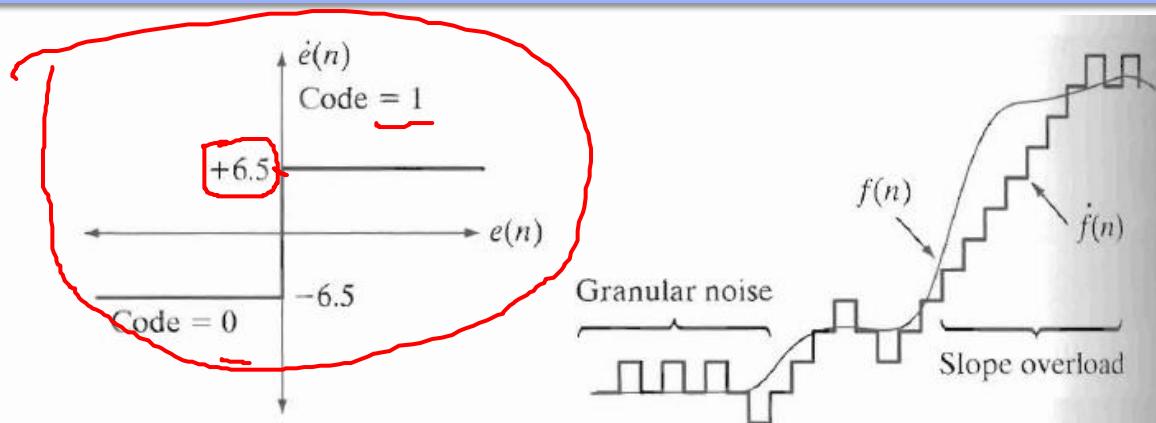
# Codarea predictivă cu pierderi

- Se adaugă un cuantizor
- Eroarea de predicție va fi mapată într-un domeniu limitat
- Cuantizorul este cel care implică pierderi
- Previne propagarea erorilor la decodor



# Modulația Delta

- Caz particular al codării predictive cu pierderi
- Probleme
  - Depășirea de pantă
  - Zgomotul de granularitate
- Variante de rezolvare
  - modulația adaptivă pe trei stări ( $\pm 1, 0$ ) datorită faptului că 65-85% sunt pixeli de nivel



Input		Encoder			Decoder			Error
$n$	$f(n)$	$\hat{f}(n)$	$e(n)$	$\dot{e}(n)$	$f(n)$	$\hat{f}(n)$	$\dot{f}(n)$	$f(n) - \hat{f}(n)$
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	-5.5
2	14	20.5	-6.5	-6.5	14.0	20.5	14.0	0.0
3	15	14.0	1.0	6.5	20.5	14.0	20.5	-5.5
.	.	.	.	.	.	.	.	.
14	29	20.5	8.5	6.5	27.0	20.5	27.0	2.0
15	37	27.0	10.0	6.5	33.5	27.0	33.5	3.5
16	47	33.5	13.5	6.5	40.0	33.5	40.0	7.0
17	62	40.0	22.0	6.5	46.5	40.0	46.5	15.5
18	75	46.5	28.5	6.5	53.0	46.5	53.0	22.0
19	77	53.0	24.0	6.5	59.6	53.0	59.6	17.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

$$\hat{f}(n) = \alpha \hat{f}(n-1)$$

# Codarea pe blocuri de pixeli

- **algoritmi de codare spațială pe blocuri**
  - gruparea pixelilor în blocuri
  - metode clasice de compresie spațială
  - compresie mai bună decât varianta clasică
- **algoritmi de codare prin transformări**
  - gruparea pixelilor pe blocuri
  - transformarea într-un alt domeniu (frecvență)
  - DFT, DCT, DST, Hadamard

# Lempel-Ziv-Welch (LZW)

- Este o tehnica de compresie foarte populara
- Folosita in fisiere: GIF (LZW), Adobe PDF (LZW), compresie UNIX (LZ Only)
- Exemplu: codare - Oxford Concise English Dictionary
  - Contine 159 000 cuvinte,  $\lceil \log_2 159\ 000 \rceil = 18\ Bits$   
=> putem reprezenta fiecare cuvant printr-un numar reprezentat pe 18 biti dar,
    - sunt prea multi biti pe cuvant
    - toata lumea are nevoie de un dictionar pentru decodare
    - merge doar pentru limba engleza
  - Solutii:
    - Crearea adaptiva a dictionarului
    - LZW – introduce ideea ca doar un dictionar initial trebuie transmis
      - atat codorul cat si decodorul sunt capabili sa creeze restul dictionarului
  - LZW original,
    - folosea dictionare de 4K (valori/entries), primele 256 (0-255) fiind ASCII codes.

- LZW – înlocuiește siruri de caractere printr-un cod
  - Nu face nici o analiză a textului/datelor de la intrare
  - doar adaugă fiecare sir nou de caractere de la intrare într-un tabel/dictionar
  - Coduri de 12 biti 0-256

STRING = get input character

WHILE there are still input characters DO

    CHARACTER = get input character

    IF STRING+CHARACTER is in the string table then

        STRING = STRING+character

    ELSE

        output the code for STRING

        add STRING+CHARACTER to the string table

        STRING = CHARACTER

    END of IF

END of WHILE

output the code for STRING

# Problema 1

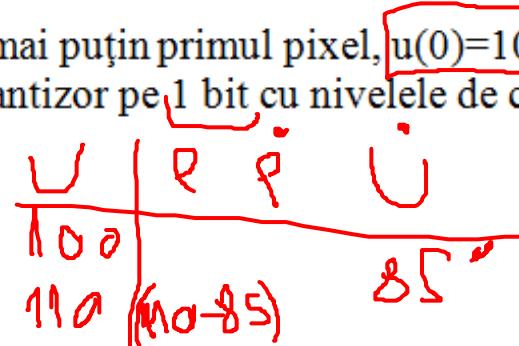
DPCM

Fie următoarea linie dintr-o imagine digitală pe nivele de gri:

$$\mathbf{u} = [100 \quad 110 \quad 120 \quad 128 \quad 64 \quad 64 \quad 64 \quad 64]$$

Dorim să codăm această linie de imagine (mai puțin primul pixel,  $u(0)=100$ ) folosind modulația delta, având la dispoziție un cuantizor pe 1 bit cu nivelele de cuantizare:  $\{-15;15\}$  și funcția de transfer:

$$e = \begin{cases} -15, & \text{daca } e < 0 \\ 15, & \text{daca } e \geq 0 \end{cases}$$



Se consideră reprezentarea valorilor de codat prin sirul de luminanțe  $\{u(1), u(2), \dots, u(7)\} = \{110, 120, \dots, 64\}$ . Dacă valoarea decodată reconstruită a primului

eșantion,  $u(0) = 85$  calculați:

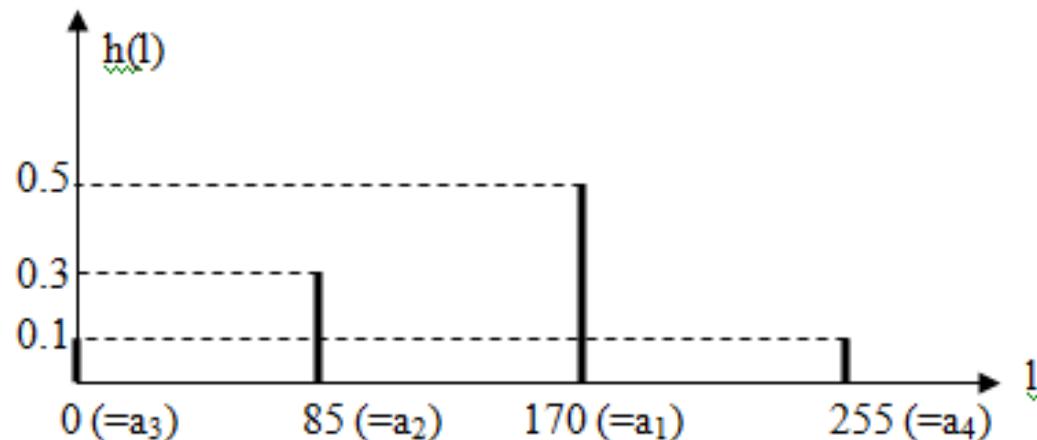
a) sirul erorilor de predicție  $\{e(n)\}$ ,  $n=1,2,\dots,7$ ;

b) sirul erorilor de predicție cuantizate  $\left\{\dot{e}(n)\right\}$ ,  $n=1,2,\dots,7$ ;

c) linia din imagine reconstruită la decodor.

# Problema 2

Pentru o clasă de imagini, se aplică cuantizarea uniformă a nivelor de gri pe 2 biți, astfel încât în orice imagine din această clasă pot fi prezente doar nivelele de gri: 0; 85; 170; 255. Probabilitățile de apariție ale celor 4 nivele de gri în imaginile din această clasă sunt date în histograma liniară normalizată a nivelor de gri ale imaginilor din clasă din figură. Se va considera că cele 4 nivele de gri reprezintă simboluri posibile a fi emise de către o sursă S, în ordinea descrescătoare a probabilităților lor de apariție:  $S=\{a_1, a_2, a_3, a_4\}=\{170, 85, 0, 255\}$ . Dacă pentru un bloc de imagine  $U[8 \times 4]$  din clasa menționată, prima linie din bloc este:  $u(0) = [170 \ 170 \ 0 \ 0]$ , să se codeze această linie ca secvență emisă de sursa S folosind codarea aritmetică.

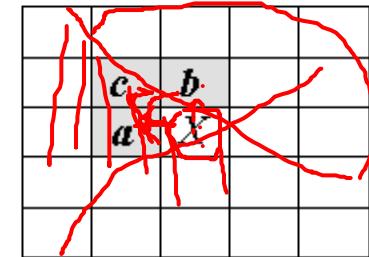
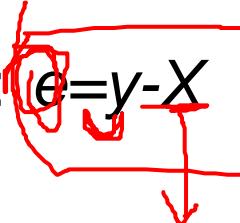


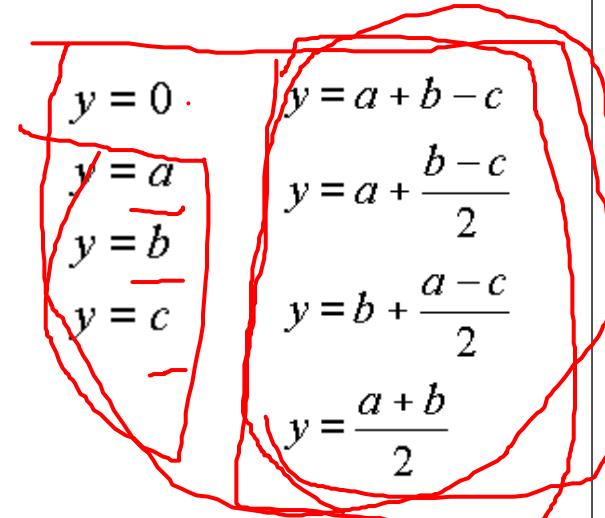
# **JPEG**- Cele două metode de compresie

- **Metoda de compresie cu pierderi bazată pe transformata cosinus discretă**
  - este vorba de formatul **JPEG „clasic”**, care permite rate de compresie importante (de **10:1** până la **20:1**) păstrând în același timp o foarte **bună calitate a imaginii**; această metodă de compresie este ireversibilă.
- **Metoda de compresie predictivă fără pierderi**
  - nu au loc pierderi de informație și este în consecință posibilă reproduserea exactă a imaginii originale, dar rata compresiei posibilă cu această metodă este mult mai modestă (aproximativ **2:1**); această metodă de compresie este reversibilă.

# JPEG/fara pierderi

- NU foloseste algoritmii de compresie prin transformari –  
~~DCT~~
- Factor de compresie slab ✓
- algoritmul JPEG fara pierderi:
  - codare diferențială
  - codor Huffman sau aritmetic
- predictorii:
  - Eroarea de predicție:  $e = y - X$




$$y = 0$$
$$y = a$$
$$y = b$$
$$y = c$$
$$y = a + b - c$$
$$y = a + \frac{b - c}{2}$$
$$y = b + \frac{a - c}{2}$$
$$y = \frac{a + b}{2}$$

# JPEG fara pierderi – codarea

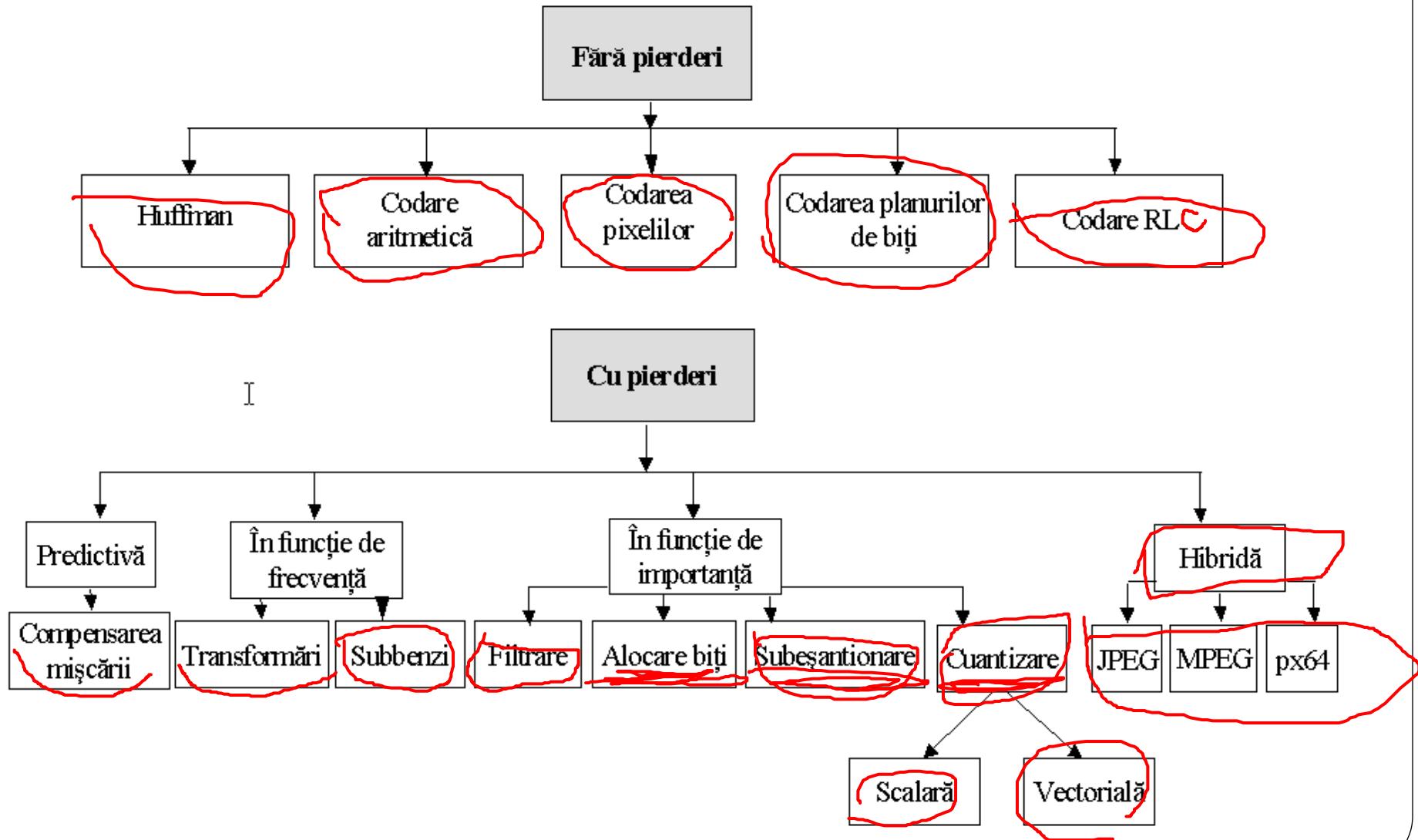
- perechi de cuvinte – *categorie, amplitudine*
- *categoria* = numarul de biti necesari pentru codarea amplitudinii
- categoria se codeaza Huffman
- daca amplitudinea este pozitiva – *valoarea amplitudinii*
- daca amplitudinea este negativa – complementul valorii absolute

# Exemplu de codare

- $a=100, b=156, c=76, X=173$
- $e = \frac{a+b}{2} - X$
- $e = -45$
- categoria 6
- rep. binara a lui 45 este 101101
- complementul lui este 010010
- $-45$  se reprezinta ca (6, 010010)
- daca codul Huffman a lui 6 este 1110
- $-45$  se codeaza cu 1110010010 (10 biti)

Categorie	Eroare de predictie
0	0
1	-1, 1
2	-3, -2, 2, 3
3	-7, ..., -4, 4, ..., 7
4	-15, ..., -8, 8, ..., 15
5	-31, ..., -16, 16, ..., 31
6	-63, ..., -32, 32, ..., 63
7	-127, ..., -64, 64, ..., 127
8	-255, ..., -128, 128, ..., 255
9	-511, ..., -256, 256, ..., 511
10	-1023, ..., -512, 512, ..., 1023
11	-2047, ..., -1024, 1024, ..., 2047
12	-4095, ..., -2048, 2048, ..., 4095
13	-8191, ..., -4096, 4096, ..., 8191
14	-16383, ..., -8192, 8192, ..., 16383
15	-32767, ..., -16384, 16384, ..., 32767
16	32768

# Compresie cu/fără pierderi



# SACCDMM - Curs 05

## Sisteme de codare bazate pe transformari

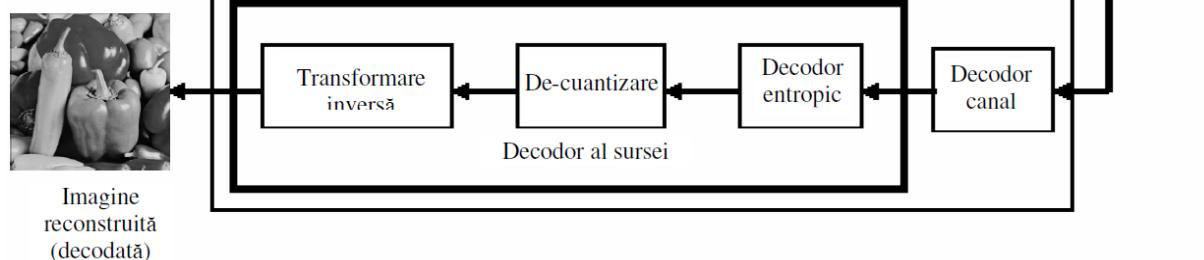
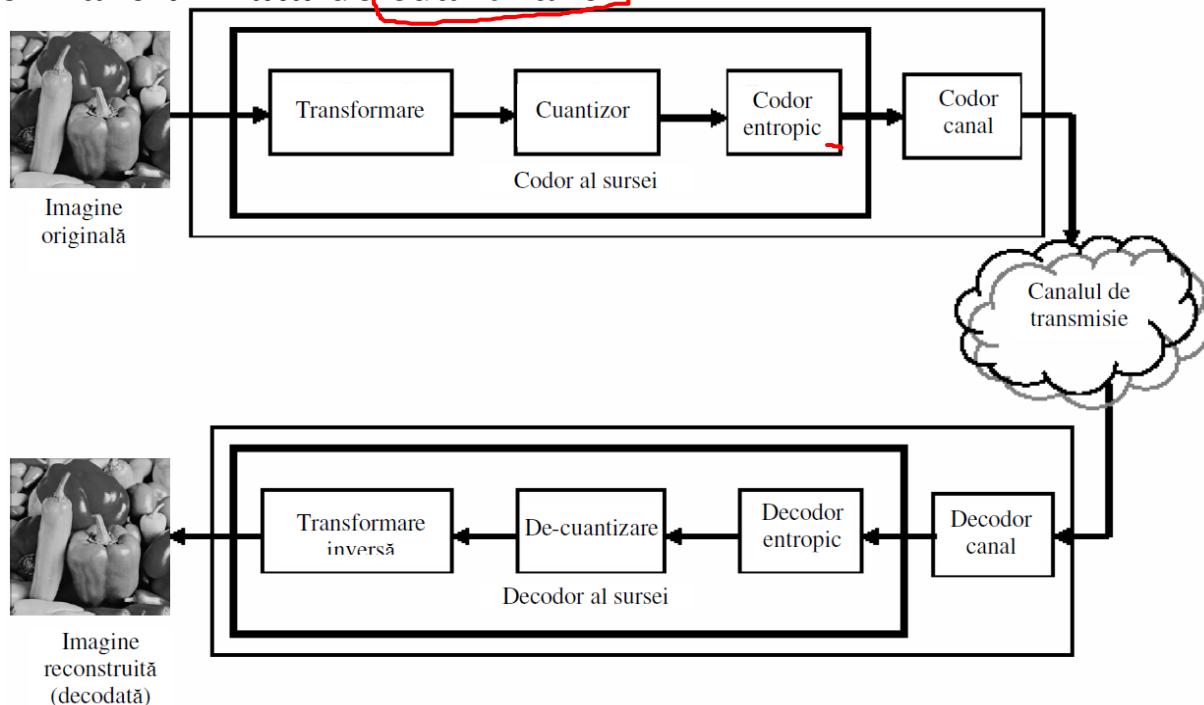
JPEG

# Topicul cursului

- Transformări de imagini
- Transformări ortogonale
- Transformări 2D
- DCT, IDCT

# Schema-bloc tipica a unui sistem de compresie a imaginilor bazat pe transformări

- **Compresie** - o imagine poate fi comprimată dacă prezintă redundanță
  - utilizare transformări pentru a decorela intensitățile pixelilor
- **Compresia cu pierderi**
  - se realizează prin transformare urmată de cuantizare



Dar, compresia realizata de blocul de codare a sursei (datorata în principal codării prin transformări) este semnificativa fata de redundanta introdusa de blocul de codare a canalului, încât efectul ultimului în micșorarea ratei de compresie poate fi neglijata.

# Compresia prin transformări de imagini

- Compresia imaginilor prin transformări:
  - exploatează principiul **redundanței spatiale**
    - adică: valoarea luminantei fiecărui pixel aduce o cantitate mica de informație vizuală suplimentară fata de vecinii săi
  - NU vom pierde din informație - transformarea este reversibilă (prin aplicarea inversei)
    - vom reprezenta aceeași informație (completă) într-un mod mai compact (prin mai puține valori)

=> transformarea intensităților pixelilor - care sunt valori corelate - într-o reprezentare în care acestea sunt decorelate

## Decorelare

- valorile după transformare sunt (ideal) independente una de alta
- media noilor valori este mai mică decât a valorilor inițiale
- entropia noilor valori este redusă (mai mică)
  - ⇒ reducerea numărului de biți necesari pentru reprezentarea

# Compresia prin transformări de imagini

- Există mai multe transformări de imagini
  - cu cât o transformare de imagine decoreleză mai mult coeficientii în imaginea transformată și
  - compactează mai bine energia într-un număr mai mic de coeficienți (date-imagine),  
=> transformare mai bună din punctul de vedere al ratei de compresie a imaginii rezultate
- **IMPORTANT:** complexitatea de calcul a transformării
  - în implementările practice - trebuie să fie cât mai mică
  - altfel, există riscul imposibilității transmisiiei și receptiei imaginilor/sevențelor video în timp real.
- Transformările tratate
  - Ortogonale (Hadamard, DCT)
  - Sub-banda (Wavelet)

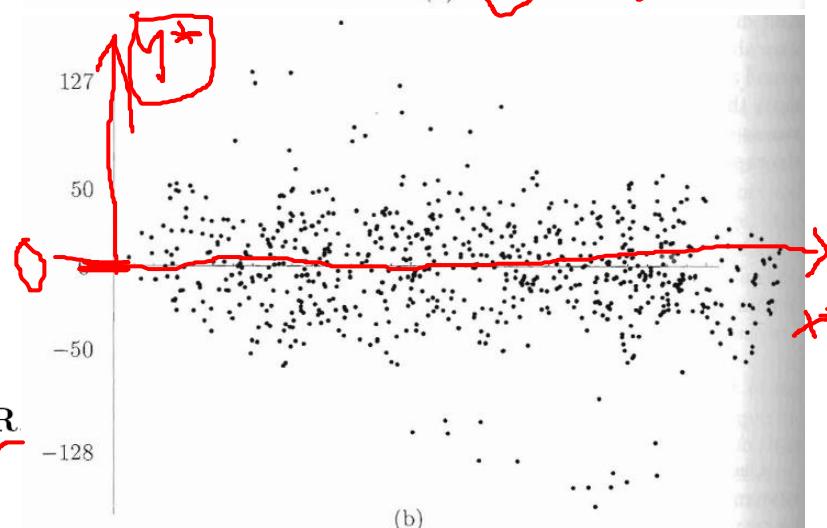
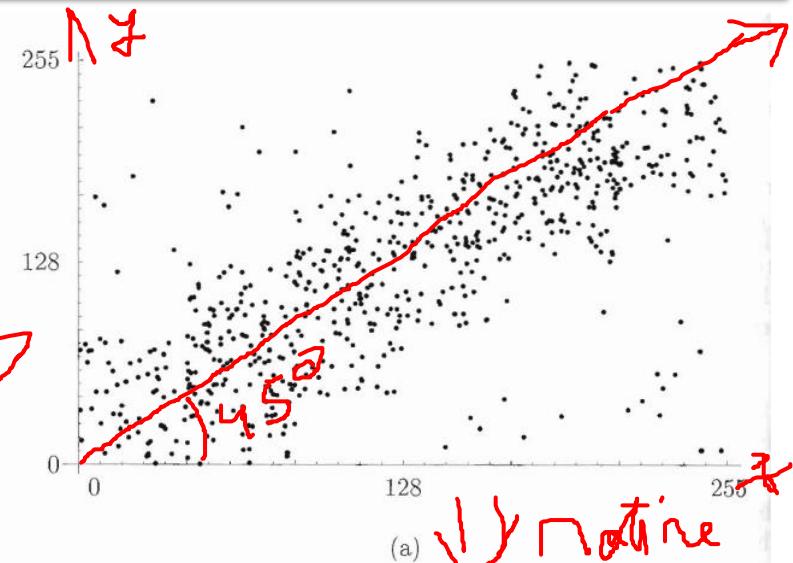
# Transformări de imagini – exemplu [Salomon, 2007]

## Scanare imagine

– extragere perechi de intensități adiacente  $(x, y)$

x	y	x	y	...	x	y
.	.	.	.	...	.	.
■	■	■	■	...	■	■
■	■	■	■	...	■	■
...	...	...	...	...	...	...
■	■	■	■	...	■	■
...	...	...	...	...	...	...
...	...	...	...	...	...	...

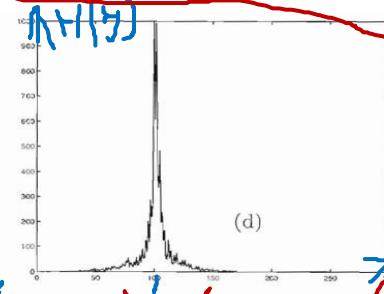
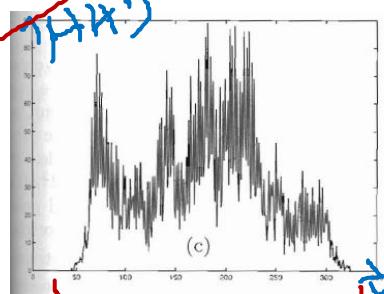
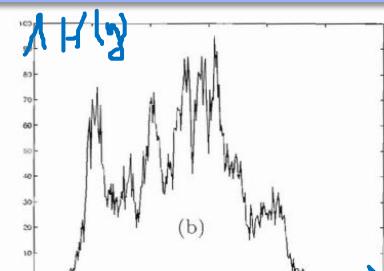
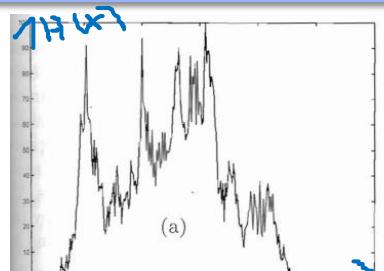
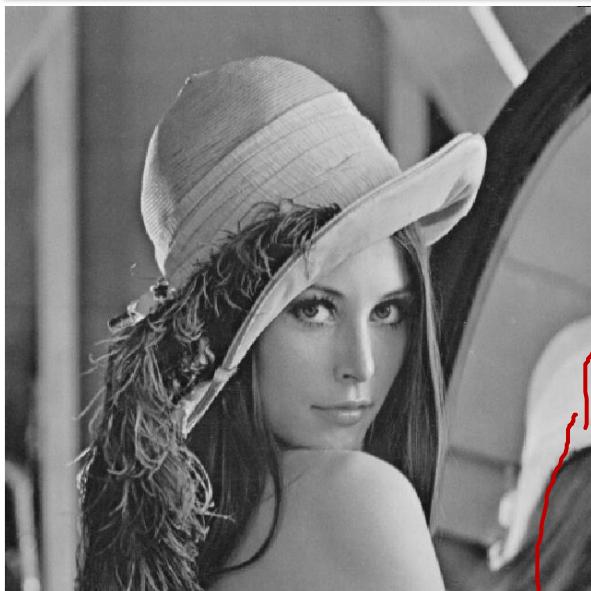
- o pereche va fi un punct în reprezentarea grafică  $x$  vs.  $y$  (vezi fig. a)
- în general, valorile  $x, y$ , sunt aproximativ egale pentru pixelii adiacenți
  - $y=x \rightarrow$  punctul e pe axa de  $45^\circ$
  - $y \approx x \rightarrow$  punctul e în jurul axei de  $45^\circ$
- aplicarea unei transformări
  - rotire cu  $45^\circ$  (vezi fig. b)
    - $y$  devine aproape 0,
    - $x$  nu se schimbă mult



$$(x^*, y^*) = (x, y) \begin{pmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{pmatrix} = (x, y) \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} = (x, y) R$$

$$(x, y) = (x^*, y^*) R^{-1} = (x^*, y^*) R^T = (x^*, y^*) \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

# Transformări de imagini – exemplificare



Distribuțiile pe x, y sunt aproximativ asemănătoare înainte de transformare

Distribuțiile pe x, y sunt DIFERITE după transformare – y aproximativ 0, scalare pe 101; x nu se modifică实质ial

## Compresie:

- se transmit valorile obținute după transformare = coeficienții transformării
  - se pot transmite sub forma a doi vectori X, Y → rezultă siruri lungi de zero
    - X – vectorul transformării intensităților de pe pozițiile impare
    - Y – vectorul transformării intensităților de pe pozițiile pare
  - după transformare - se pot adăuga alte metode de compresie
  - dacă se acceptă compresie cu pierderi – transformarea este urmata de cuantizare => valori mai mici
- Concentrarea energie într-o componentă
  - Face posibila cuantizarea mai fină a vectorului X și mai puternica a lui Y

# Transformări de imagini 1D/2D

- Majoritatea transformărilor de imagini = “generalizări” ale transformatei Fourier  
=> reprezintă imaginea printr-o componentă de c.c. și mai multe componente de c.a.
- Semnal 1D → poate fi reprezentat prin serii de funcții ortogonale,
- O imagine → poate fi reprezentată printr-o serie de funcții de bază ortogonale,
  - numite imagine de bază - set generat cu ajutorul unor matrici unitare.
- O transformare unitară a imaginii  $\mathbf{U}[M \times N]$  = o rotație a spațiului MN-dimensional,
  - definită de o matrice unitară de rotație  $\mathbf{A}$  (de dimensiune  $MN \times MN$ )  
$$\mathbf{A}^{-1} = \mathbf{A}^T$$

1D

$$v = \mathbf{A} u \quad \text{sau} \quad v(k) = \sum_{n=0}^{N-1} a(k, n) u(n), \quad 0 \leq k \leq N-1$$

2D

$$u = \mathbf{A}^T v \quad \text{sau} \quad u(n) = \sum_{k=0}^{N-1} a^*(n, k) v(k), \quad 0 \leq n \leq N-1$$

1D

$$V = \mathbf{A} U \mathbf{A}^T \quad \text{sau} \quad v(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a_{k,l}(m, n) \cdot u(m, n), \quad 0 \leq k, l \leq N-1$$

2D

$$U = \mathbf{A}^T V \mathbf{A}^* \quad \text{sau} \quad u(m, n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{k,l}^*(m, n) \cdot v(k, l), \quad 0 \leq k, l \leq N-1$$

# Transformări de imagini 1D/2D

## • Conservarea energiei

- **1-D**  $\|v\|^2 = \sum_{k=0}^{N-1} |v(k)|^2 = \sum_{n=0}^{N-1} |u(n)|^2 = \|u\|^2$

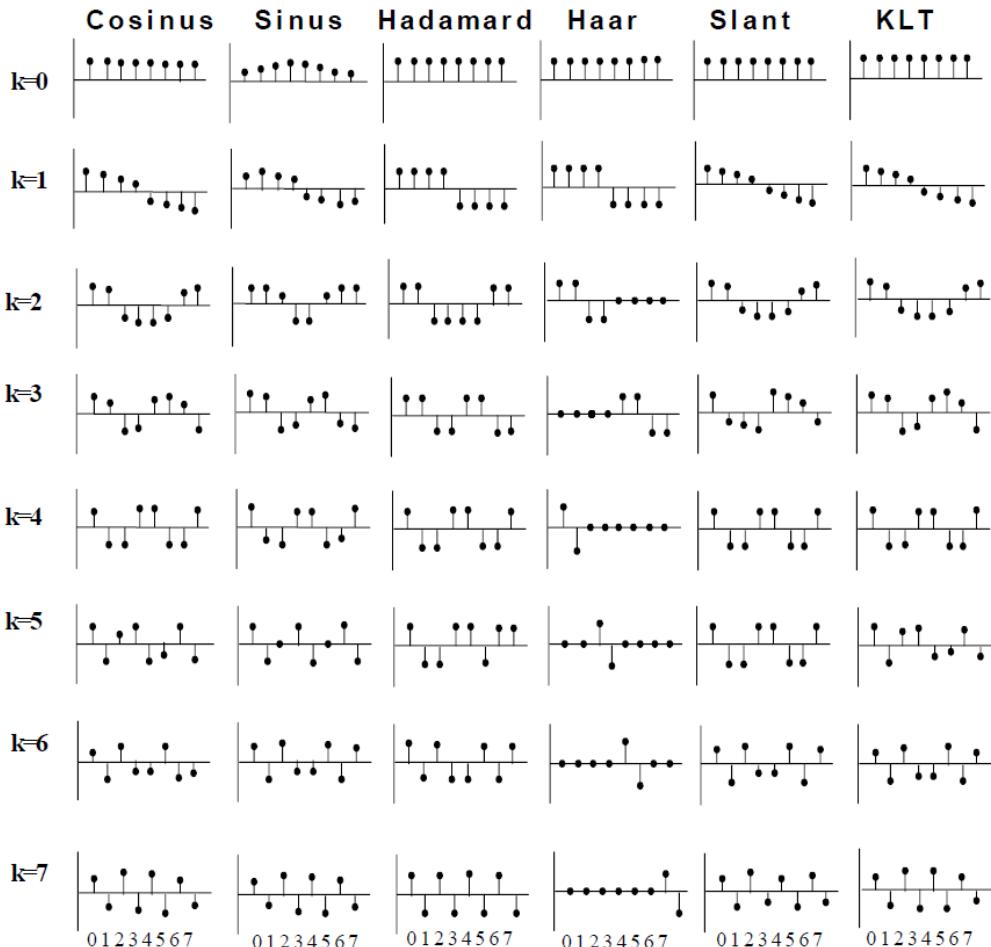
- **2-D**  $\|V\|^2 = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} |v(k,l)|^2 = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} |u(m,n)|^2 = \|U\|^2$

## • Compactarea energiei și varianța coeficienților

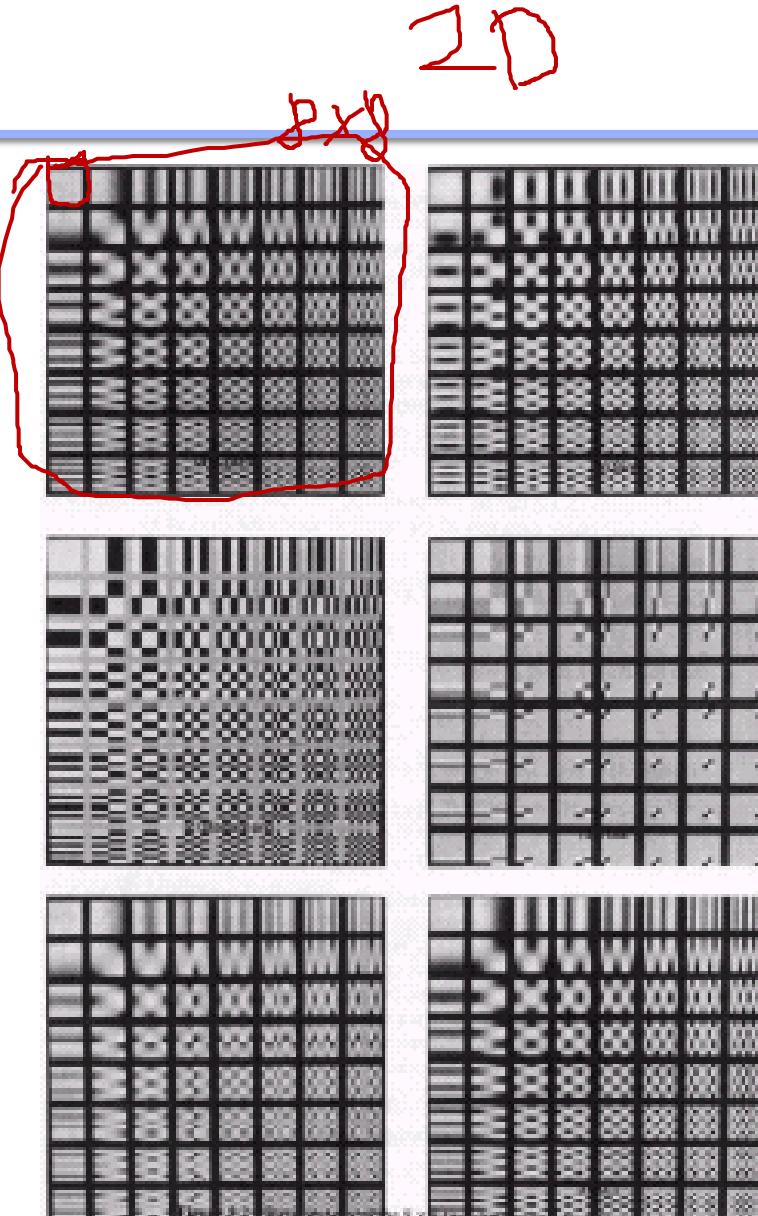
- Transformările unitare au tendința de împachetare (compactare) a unei importante părți a energiei imaginii într-un număr relativ mic de coeficienți ai transformatei imaginii.
- Energia secvenței de intrare **U** este egal distribuită - energia coeficienților **V** trebuie să fie inegal distribuită
- Energia totală se conservă,
  - rezultă că o mare parte a coeficienților vor conține o cantitate foarte mică de energie.

# Funcții și imagini de bază

10



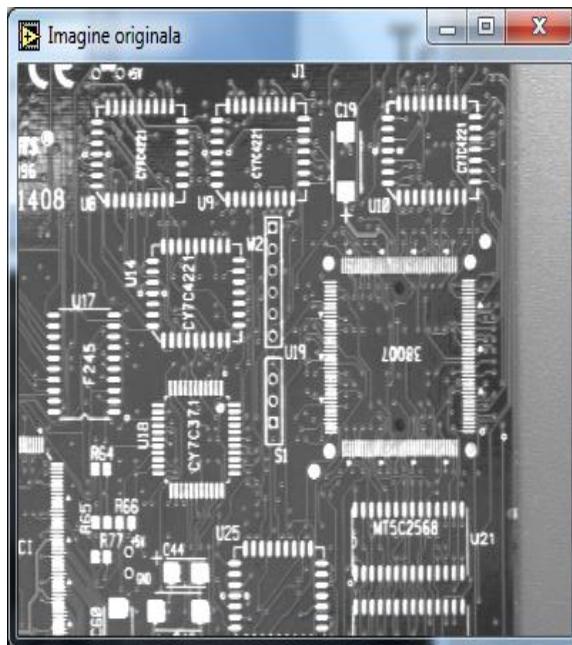
Functii de baza (vectori de baza)



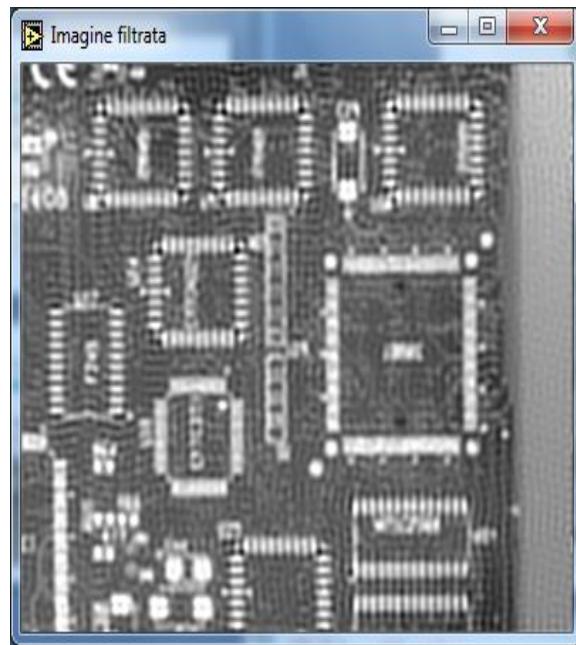
Imagini de baza (ex.): DCT, Haar, ....

# Transformări ortogonale

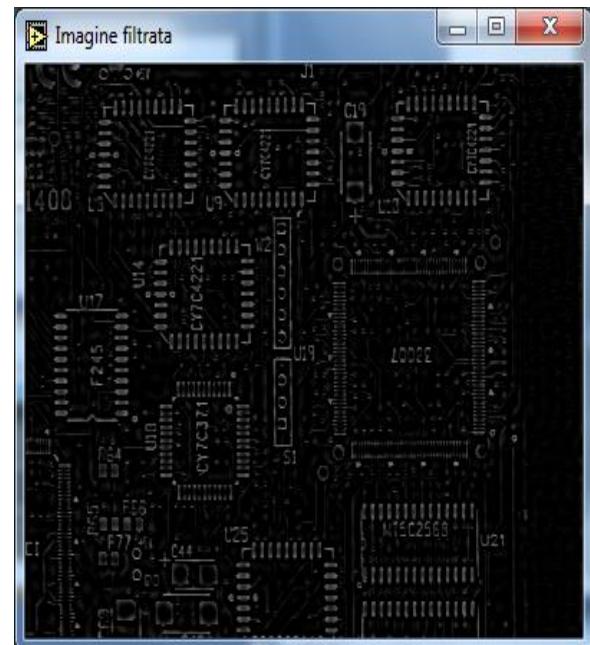
- Frecvență
  - Joase – caracteristicile importante din imagine
  - Înalte – detalii mai puțin importante



Original



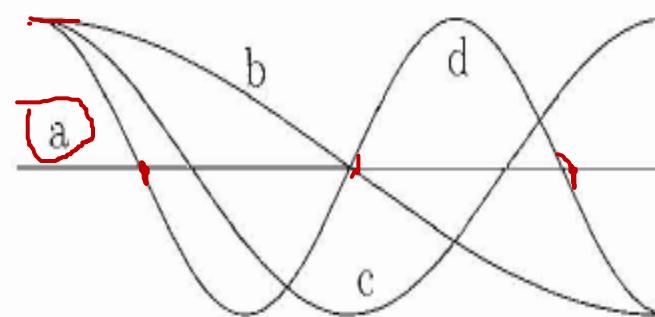
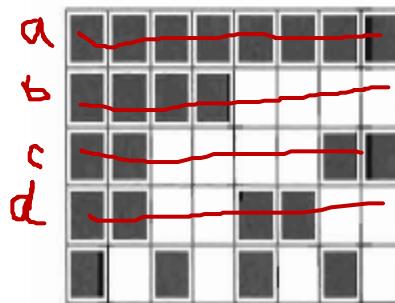
FTJ



FTS

# Transformări ortogonale

- Reducerea redundanței imaginii
- Izolarea frecvențelor
  - Identificarea părților mai puțin importante din imagine prin izolarea diferitelor frecvențe (benzi de frecvență) din imagine
  - Se pot aplica grade de cuantizare diferită pentru diversele benzi de frecvență
- Transformările implementate practic
  - Rapide
  - Ușor de implementat
  - În general se folosesc transformările liniare



Frecvența orizontală

# Transformări ortogonale

- Transformări liniare:

$$\begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix}$$

- Forma matricială:

$$v = Au, \text{ sau } v(k) = \sum_{n=0}^{N-1} a(k, n)u(n), \quad 0 \leq k \leq N - 1$$

- $a_{kn}$  (fiecare rând-vector de bază) se calculează în funcție de tipul transformației și trebuie să:
  - reducă redundanța
  - izoleze frecvențele
  - $u_n$  sunt valori pozitive și corelate (intensități)
  - $v_k$  = suma ponderată a tuturor intensităților  $u_n$  (care sunt transformate)
  - primul  $v_k$  va avea o valoare mai mare,  $a_{kn}$  având doar valori pozitive
  - ceilalți coeficienți  $v_k$  vor avea valori mai mici dacă  $a_{kn}$  vor fi pozitivi și negativi

# Transformări ortogonale

- Pentru a putea izola diferențele frecvențe
  - vectorii de bază trebuie să fie ortogonali adică matricea A este ortogonală (Ortogonală  $\leftrightarrow$  inversa ei = cu transpusa)

$$A^{-1} = A^T$$

- Exemplu de A: număr egal de 1, -1
  - Primul rând va fi numai de 1
  - Rândul 2 va avea o singură schimbare de semn
  - Rândul 3 va avea 2 schimbări de semn și.a.m.d.

=> Transformarea Walsh-Hadamard:

- Pentru conservarea energiei
  - necesar factorul de scală  $\frac{1}{2}$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix}$$

$$A^{-1} = A$$

# Transformări ortogonale – conservarea energiei

- Conservarea energiei

$$E(u) = 5^2 + 6^2 + 7^2 + 8^2 = 174$$

$$E(v) = 13^2 + (-2)^2 + 0^2 + (-1)^2 = 174$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 2 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix}$$

- Exemplu 1:  $\mathbf{u} = [5 \ 6 \ 7 \ 8]^T \Rightarrow \mathbf{v} = [13 \ -2 \ 0 \ -1]^T$

$$E(v) = E(u) = 174$$

compactare energie în  $\mathbf{v}$ , prima valoare 97% din energie !!!

- în  $\mathbf{u}$  primul element 14% din energie

$$5 + 6 + 7 + 8 = 26$$

$$\Rightarrow v_{(0)} = 13$$

- Exemplu 2:

$$3 \cdot 187^2 + 2 \cdot (189)^2 + \dots = 281280$$

$$\mathbf{u} = [186 \ 184 \ 187 \ 187 \ 189 \ 187 \ 190 \ 190]^T$$

$$\mathbf{v} = [530,33 \ -4,24 \ 0 \ -2,83 \ 1,41 \ 0 \ 0 \ 1,41]^T$$

$$E(v) = E(u) = 281280$$

# Transformări ortogonale

- Doar prin transformare NU obținem compresie!!!
  - ⇒ este necesară **cuantizarea** pentru a avea **compresie**
    - rezultatul transformării inverse va fi **diferit** de original
    - chiar dacă sunt **diferite**, matricile sunt foarte apropiate de original  
=> **compresie cu pierderi**
- Pentru a obține un raport de compresie ridicat
  - algoritmi **succesivi**, transformări complexe

# Cuantizare exemplificare

$$\mathbf{u} = [186 \quad 184 \quad 187 \quad 187 \quad 189 \quad 187 \quad 190 \quad 190]^T$$

- Fie rotunjirea la valori întregi

$$\mathbf{v} = [530 \quad -4 \quad 0 \quad -3 \quad 1 \quad 0 \quad 0 \quad 1]^T$$

$$\Rightarrow \mathbf{u}' = [185,62 \quad 184,2 \quad 187,03 \quad 187,03 \quad 188,44 \quad 187,03 \quad 189,86 \quad 189,86]^T$$

$$\Rightarrow \mathbf{u}' = [186 \quad 184 \quad 187 \quad 187 \quad 188 \quad 187 \quad 190 \quad 190]^T$$

- Fie trunchierea ( $|v_i| < 2$  atunci zero)

$$\mathbf{v} = [530,33 \quad -4,24 \quad 0 \quad -2,83 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

$$\Rightarrow \mathbf{u}' = [185 \quad 185 \quad 187 \quad 187 \quad 187,99 \quad 187,99 \quad 189,99 \quad 189,99]^T$$

$$\Rightarrow \mathbf{u}' = [185 \quad 185 \quad 187 \quad 187 \quad 188 \quad 188 \quad 190 \quad 190]^T$$

# Transformări bidimensionale



- Se aplică 1D pe linii

$$4 \times 4 \quad U = \begin{pmatrix} 5 & 6 & 7 & 4 \\ 6 & 5 & 7 & 5 \\ 7 & 7 & 6 & 6 \\ 8 & 8 & 8 & 8 \end{pmatrix};$$

$$V_1 = A \cdot U = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 5 & 6 & 7 & 4 \\ 6 & 5 & 7 & 5 \\ 7 & 7 & 6 & 6 \\ 8 & 8 & 8 & 8 \end{pmatrix} = \begin{pmatrix} 26 & 26 & 28 & 23 \\ -4 & -4 & 0 & -5 \\ 0 & 2 & 2 & 1 \\ -2 & 0 & -2 & -3 \end{pmatrix}$$

- Mai avem corelație între elementele de pe linii
  - se aplică 1D pe coloane, W- simetrică deci

$$A^T = A$$

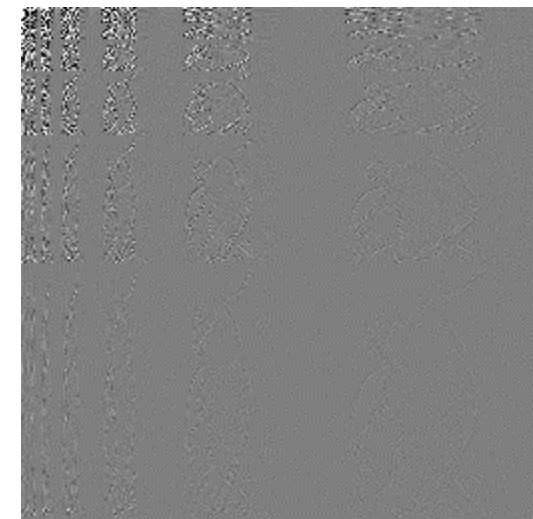
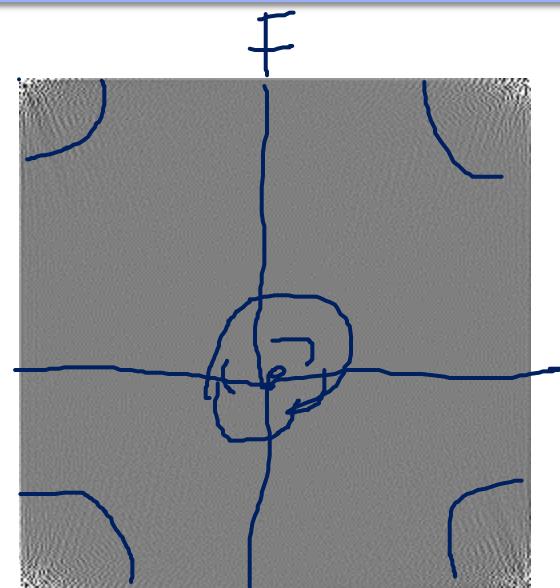
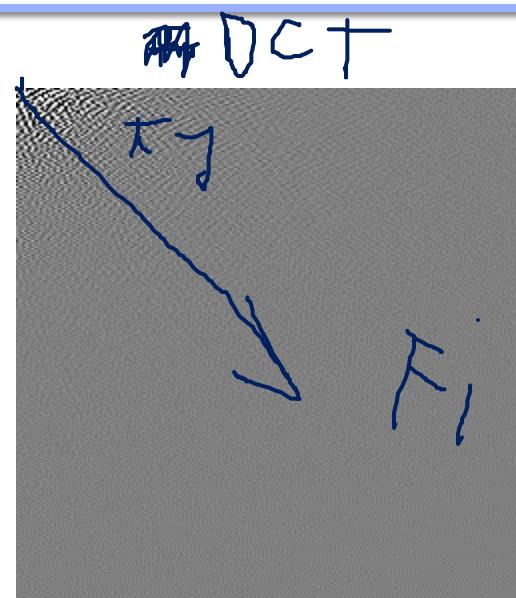
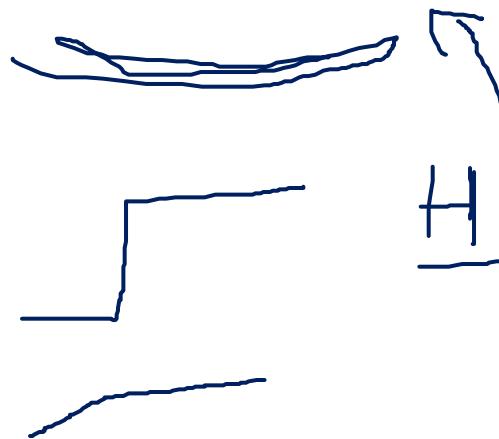
$$V = V_1 \cdot A^T = A \cdot U \cdot A^T = A \cdot U \cdot A$$

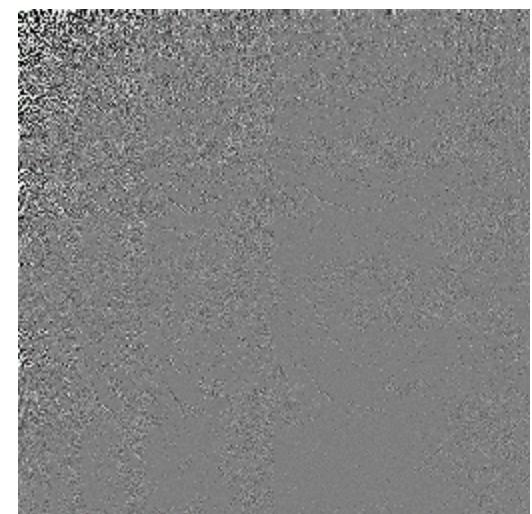
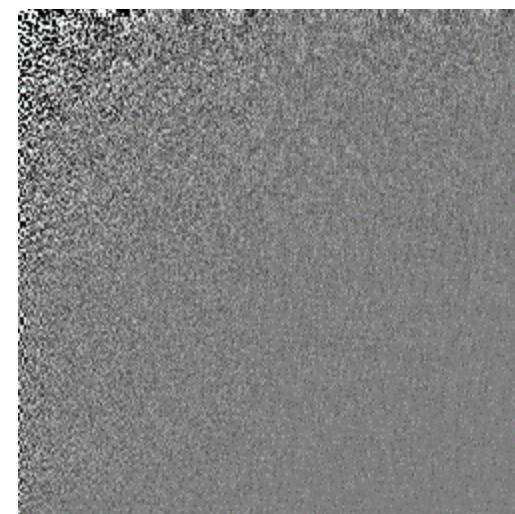
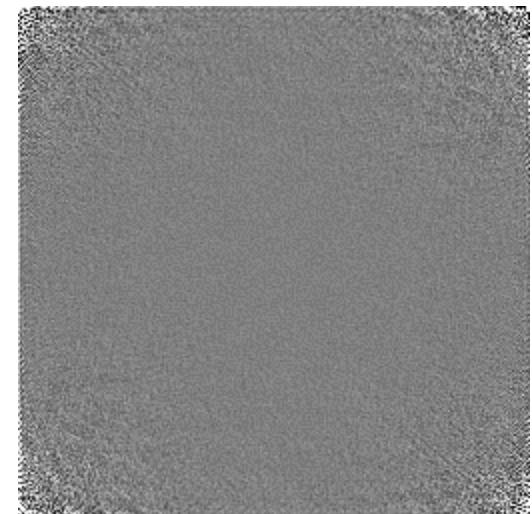
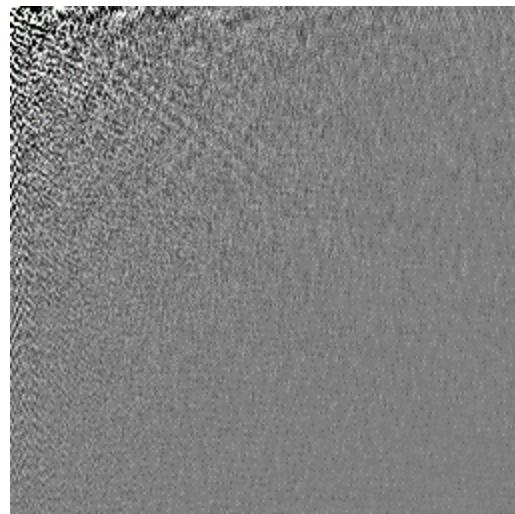
$$V = V_1 \cdot A^T = \begin{pmatrix} 26 & 26 & 28 & 23 \\ -4 & -4 & 0 & -5 \\ 0 & 2 & 2 & 1 \\ -2 & 0 & -2 & -3 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 103 & 1 & -5 & 5 \\ -13 & -3 & -5 & 5 \\ 5 & -1 & -3 & -1 \\ -7 & 3 & -3 & -1 \end{pmatrix}$$

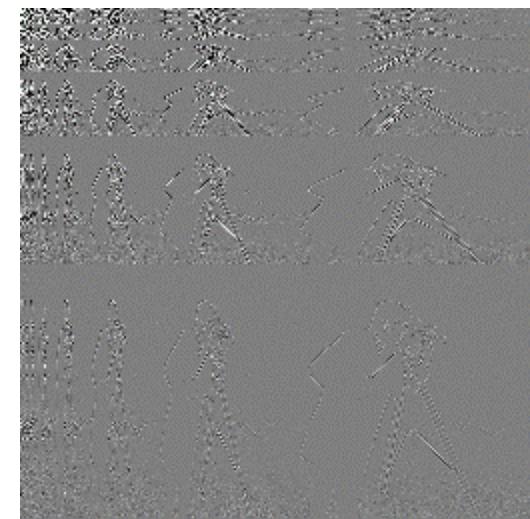
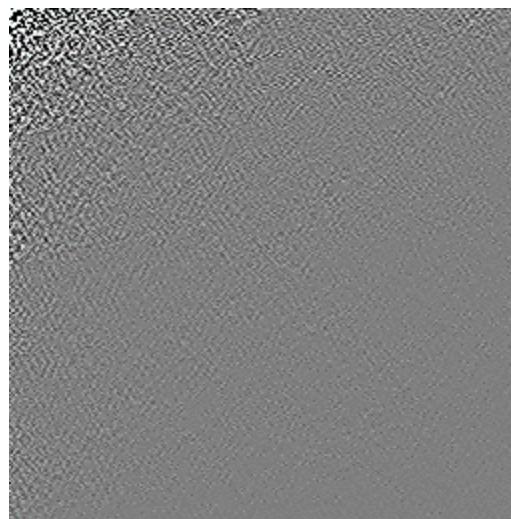
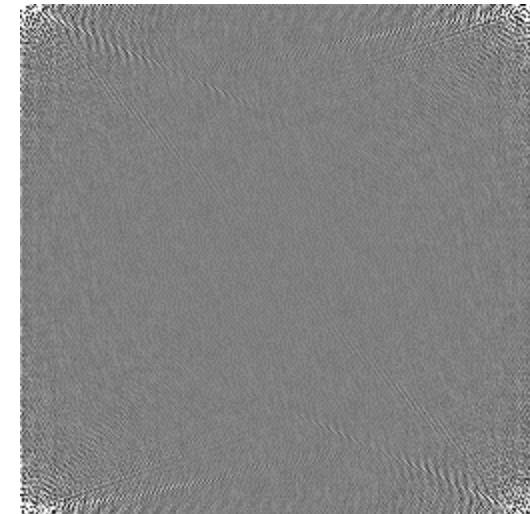
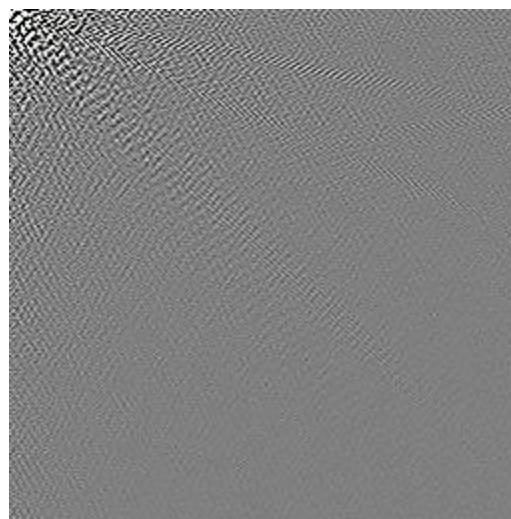
- V este decorrelată și elementul din stânga sus – este dominant
- Colțul din stânga sus conțin elemente importante
- Compresie – dacă se aplică cuantizare
- Se pot elimina în special elementele din dreapta-jos – elementele de înaltă frecvență

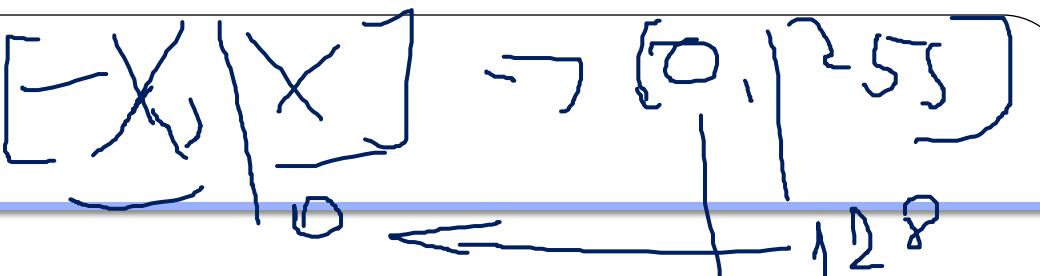
# Transformări bidimensionale

- Transformarea **DFT**
  - rapida, foarte utilizata in prelucrarea digitala a semnalelor, convolutie, filtrare digitala, analiza
  - foarte buna împachetare a energiei, dar necesita calcule cu valori complexe
- Transformata **Walsh-Hadamard**
  - este mai rapida decat transformarile sinusoidale (nu necesita inmultiri)
  - se foloseste pentru implementarea hard a algoritmilor de PDI
  - usor de simulat dar dificil de implementat
  - se aplica in compresia de imagini, filtrare si proiectarea de codoare
  - prezinta o bună compactare de energie
- Transformata **Haar**
  - transformare foarte rapida (cea mai simplă dintre transformatele Wavelet)
  - se foloseste in extragerea de caracteristici, codarea de imagini si in aplicatii de analiza a imaginilor
  - compactarea de energie este medie.
- Transformata **Karhunen-Loeve**
  - Cea mai bună variantă teoretică – compactarea energiei
  - Coeficientii nu sunt ficsi – depind de datele de la intrare; calcule complicate; coeficientii trebuie inclusi în şirul codat (necesari la decodare)
  - nu prezinta un algoritm rapid
  - se foloseste mai ales in cazul vectorilor de dimensiuni mici si pentru evaluarea performantelor altor transformari
- Transformata **Cosinus Discretă (DCT)**
  - Aproape la fel de eficientă ca și KLT
  - Folosește vectori de bază ficsi
  - transformare rapida, necesita operatiuni reale
  - alternativa optimala a transformarii K-L pentru imagini inalt corelate
  - utilizata in proiectarea codoarelor prin transformari si a filtrorelor Wiener.
  - o compactare a energiei excelenta



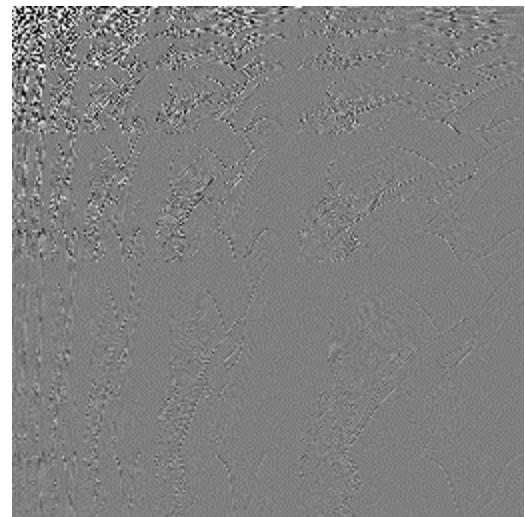
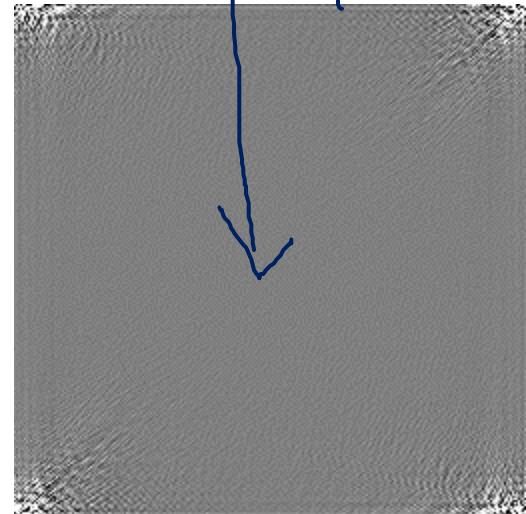
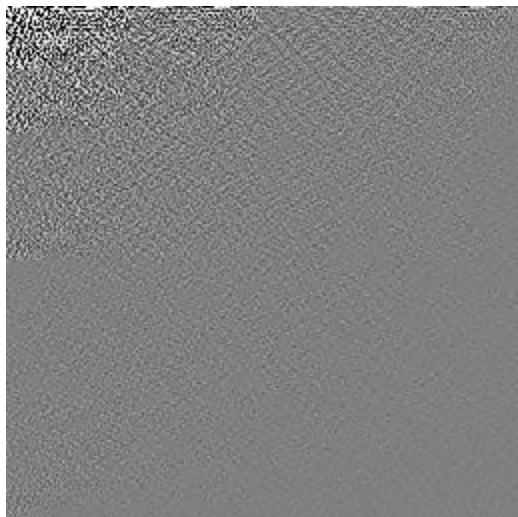
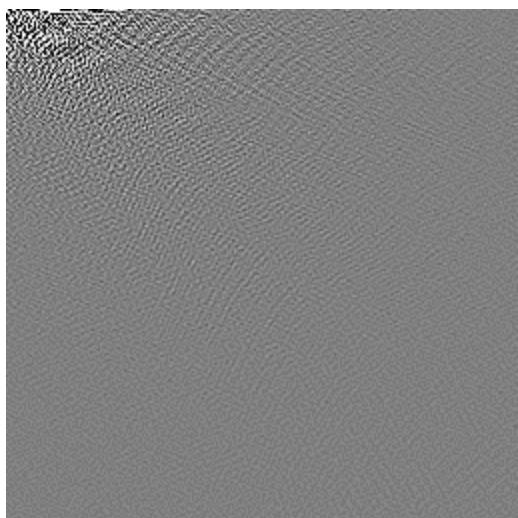




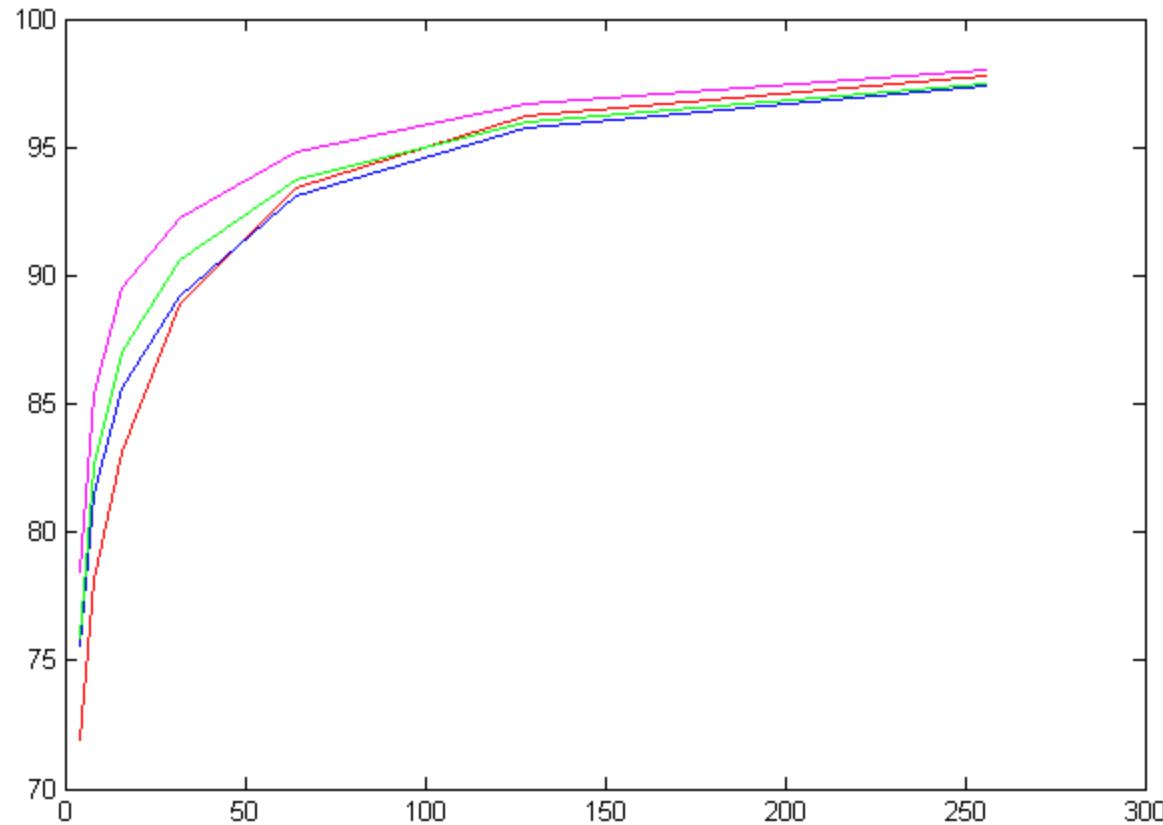


$(0, 255)$   
 $\downarrow$

$[-X, +X]$

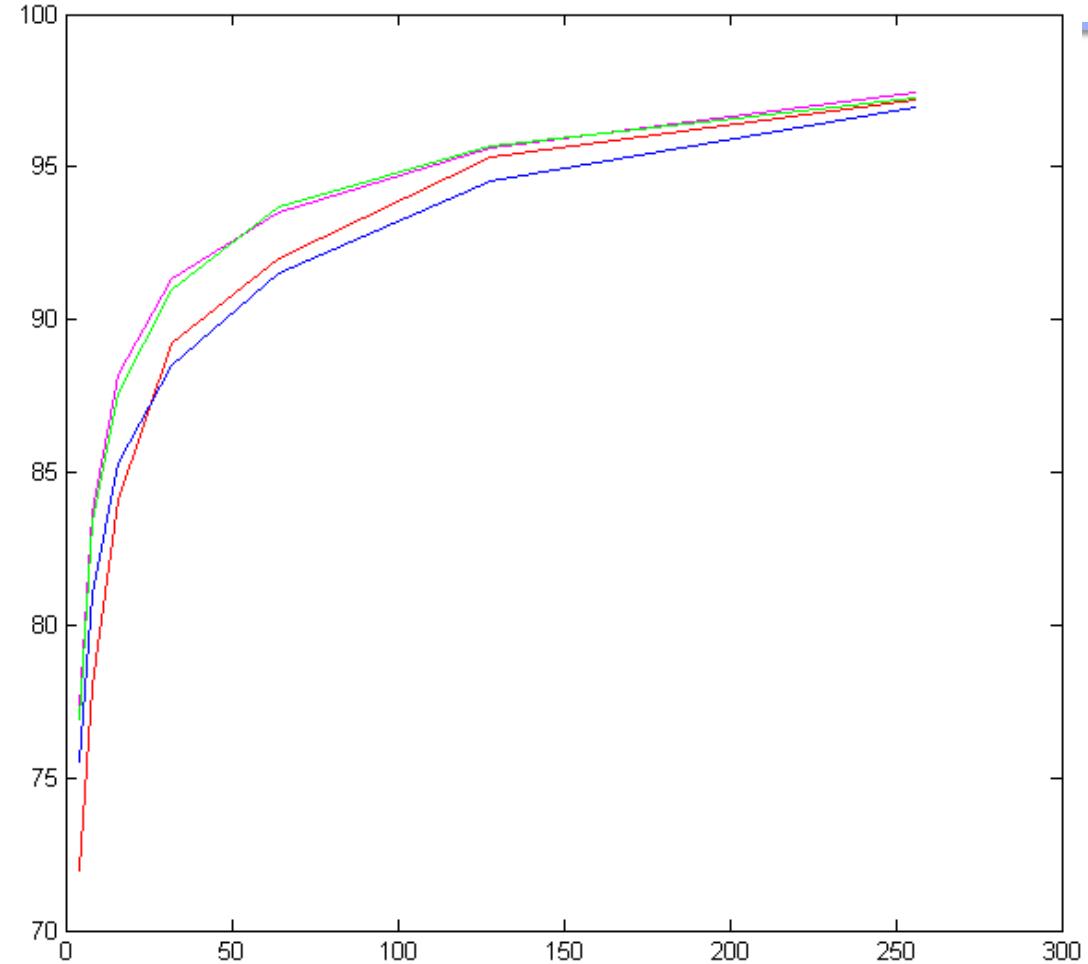


# Lena - dimensiune bloc/ compactare energie



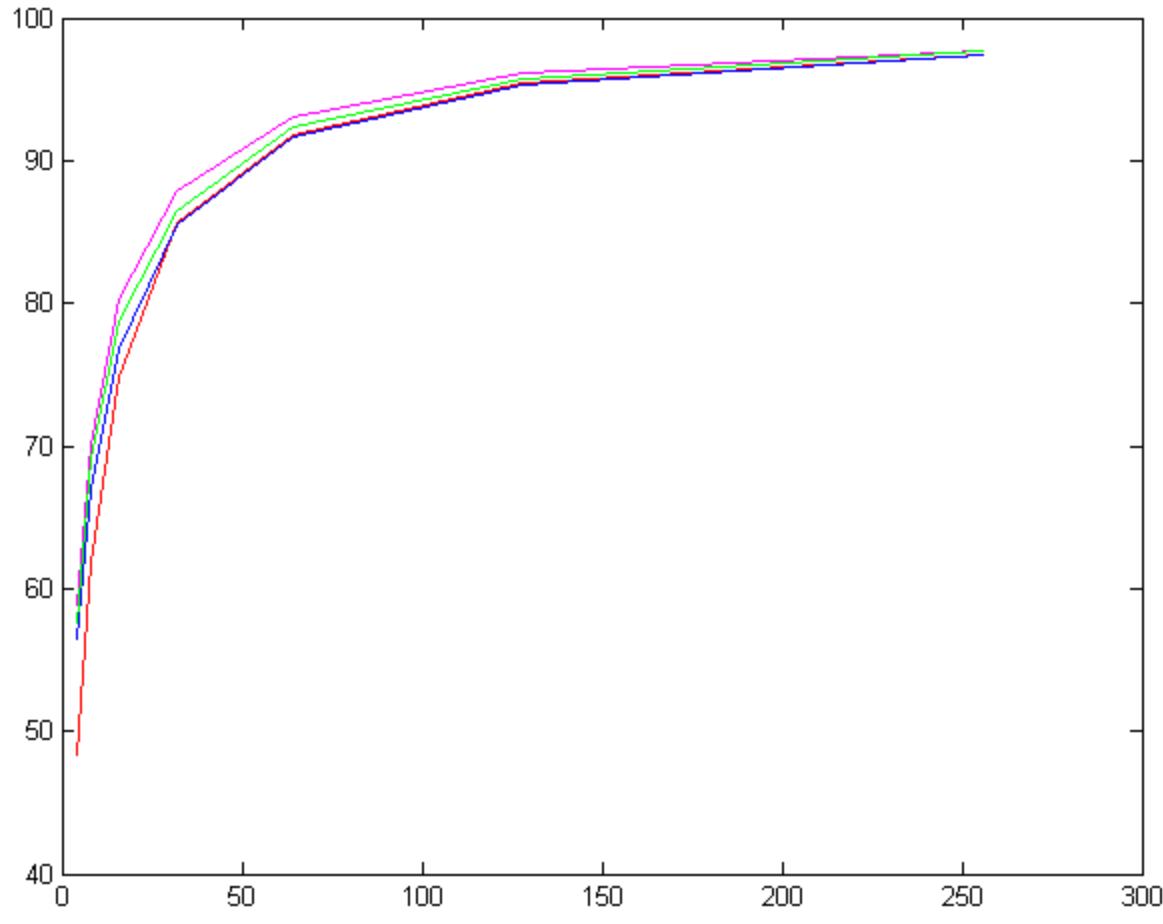
- BISz 256:24 dB BISz 128:25 dB BISz 64:27 dB BISz 32:29 dB BISz 16:32 dB BISz 8:34 dB BISz 4:37 dB DFT
- BISz 256:23 dB BISz 128:25 dB BISz 64:27 dB BISz 32:29 dB BISz 16:32 dB BISz 8:35 dB BISz 4:38 dB Hadamard
- BISz 256:25 dB BISz 128:26 dB BISz 64:28 dB BISz 32:31 dB BISz 16:33 dB BISz 8:36 dB BISz 4:38 dB DCT
- BISz 256:25 dB BISz 128:27 dB BISz 64:28 dB BISz 32:31 dB BISz 16:33 dB BISz 8:35 dB BISz 4:38 dB Haar

# Cameraman



- BISz 256:23 dB BISz 128:24 dB BISz 64:27 dB BISz 32:29 dB BISz 16:32 dB BISz 8:35 dB BISz 4:39 dB DFT
- BISz 256:22 dB BISz 128:24 dB BISz 64:26 dB BISz 32:29 dB BISz 16:32 dB BISz 8:36 dB BISz 4:39 dB Hadamard
- BISz 256:23 dB BISz 128:25 dB BISz 64:27 dB BISz 32:30 dB BISz 16:33 dB BISz 8:36 dB BISz 4:39 dB DCT
- BISz 256:25 dB BISz 128:27 dB BISz 64:29 dB BISz 32:31 dB BISz 16:34 dB BISz 8:36 dB BISz 4:39 dB Haar

# Bridge



- BISz 256:22 dB BISz 128:24 dB BISz 64:25 dB BISz 32:27 dB BISz 16:29 dB BISz 8:32 dB BISz 4:36 dB DFT
- BISz 256:22 dB BISz 128:24 dB BISz 64:25 dB BISz 32:27 dB BISz 16:30 dB BISz 8:33 dB BISz 4:36 dB Hadamard
- BISz 256:23 dB BISz 128:24 dB BISz 64:26 dB BISz 32:28 dB BISz 16:30 dB BISz 8:33 dB BISz 4:36 dB DCT
- BISz 256:23 dB BISz 128:24 dB BISz 64:26 dB BISz 32:28 dB BISz 16:30 dB BISz 8:33 dB BISz 4:36 dB Haar

# Transformata Cosinus Discreta (DCT)

- DCT – Discrete Cosine Transform

- intrările** – pixeli, eșantioane audio, etc.,
- valori întregi

- ieșirile** - sunt coeficienții DCT

- primul coeficientul DC +
- restul sunt coeficienții AC
- *valori reale pozitive și negative*

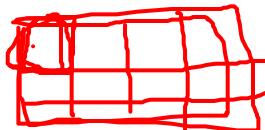
- Calcule greoaie** – timpi de calcul mari –
- necesitatea alegerii DCT rapide

- Avantaj** / - concentrează energia într-un număr mic de coeficienți
  - cei mai mulți coeficienți sunt zero sau foarte apropiati de zero

- În aplicații practice datele sunt împărtășite în seturi de  $n$  valori
  - **în general**  $n=8$

$$v(k,l) = \alpha(k) \cdot \alpha(l) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m,n) \cdot \cos\left[\frac{(2m+1)k\pi}{2N}\right] \cos\left[\frac{(2n+1)l\pi}{2N}\right]$$

$$u(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \alpha(k) \cdot \alpha(l) \cdot v(k,l) \cdot \cos\left[\frac{(2m+1)k\pi}{2N}\right] \cos\left[\frac{(2n+1)l\pi}{2N}\right]$$



$$\mathbf{V} = \mathbf{C} \mathbf{U} \mathbf{C}^T$$

$$\mathbf{U} = \mathbf{C}^T \mathbf{V} \mathbf{C}$$

$$c_{k,m} = \alpha(k) \cdot \cos\left[\frac{(2m+1)k\pi}{2N}\right]$$

$$\alpha(0) = \sqrt{\frac{1}{N}} \quad \text{si}$$

$$\alpha(k) = \sqrt{\frac{2}{N}} \quad \text{pentru } 1 \leq k \leq N$$

# Obținerea matricei DCT

10 → C

- Se selectează  $N$  unghiuri

$$\theta_m = \frac{(2m + 1)\pi}{2N}$$

$N=8$ :

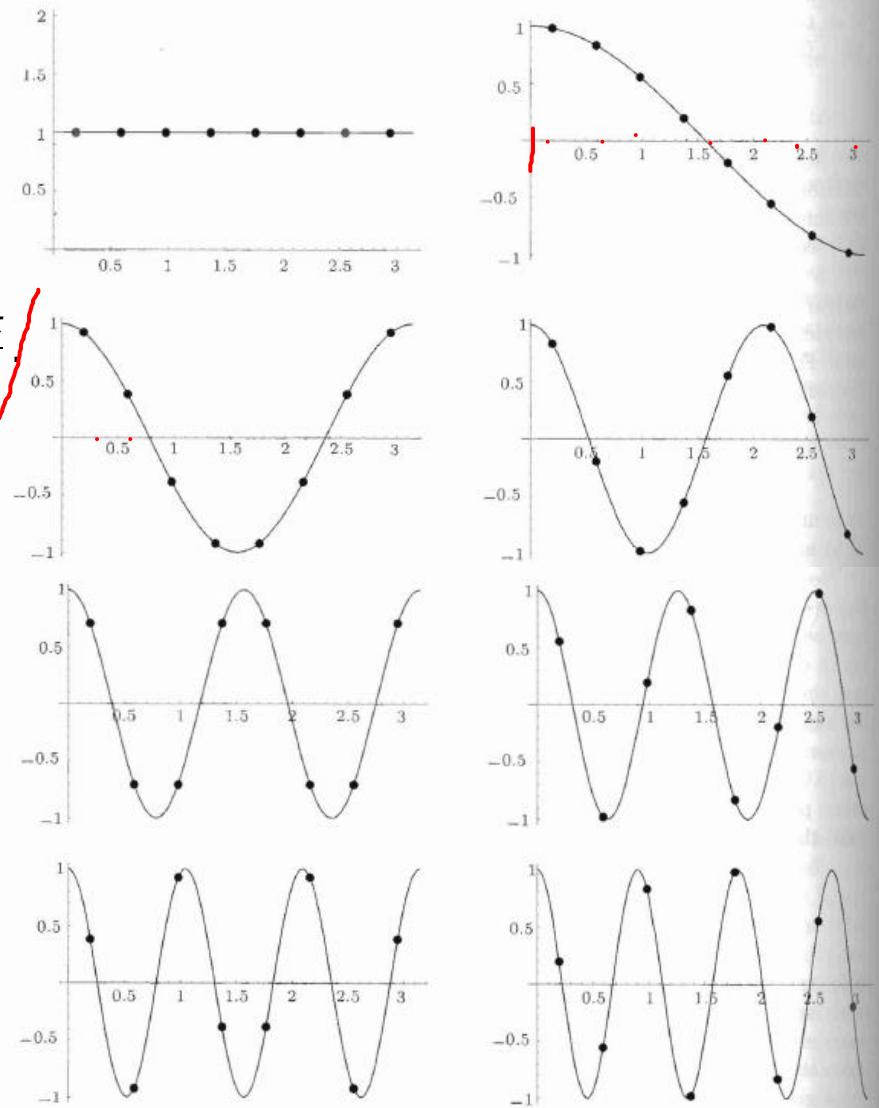
$$\theta_m = \left[ \frac{\pi}{16}, \frac{3\pi}{16}, \frac{5\pi}{16}, \frac{7\pi}{16}, \frac{9\pi}{16}, \frac{11\pi}{16}, \frac{13\pi}{16}, \frac{15\pi}{16} \right]$$

- Se calculează vectorii  $\cos(k\theta_m)$

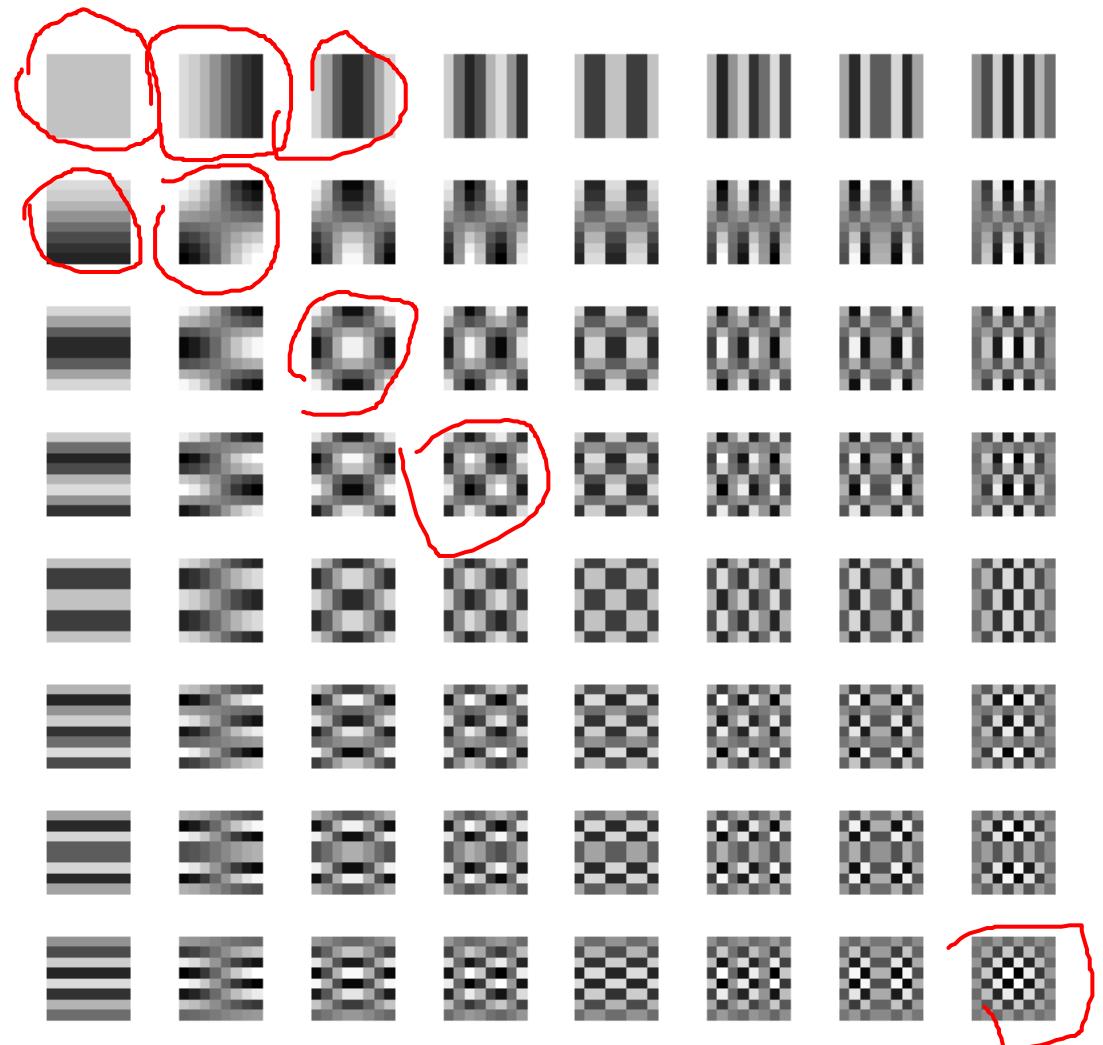
- Se normalizează fiecare vector și se creează **C** matricea coeficientilor DCT

$$c_{k,m} = \alpha(k) \cdot \cos(k\theta_m)$$

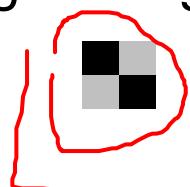
$$\alpha(0) = \sqrt{\frac{1}{N}} \quad \text{si} \quad \alpha(k) = \sqrt{\frac{2}{N}} \quad \text{pentru} \quad 1 \leq k \leq N$$



# Imagini de baza DCT



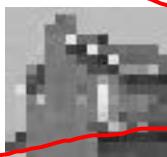
Imagine originală



$$= V(1,1)$$

$$+ V(9,9)$$

Imagine originală



$$= V(1,3)$$

$$+ V(1,5)$$

$$+ V(1,7)$$

$$+ V(1,9)$$

$$V(1,13)$$

+

$$+ V(1,15)$$

+

$$V(2,1)$$

+

$$V(2,9)$$

+

$$V(3,1)$$

+

$$V(3,5)$$

$$+ V(5,1)$$

+

$$+ V(5,2)$$

+

$$V(5,6)$$

+

$$V(5,8)$$

...

$$+ V(16,15)$$

+

Imagine aproximata

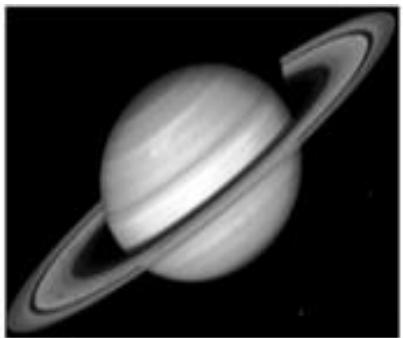
Dacă se rețin doar 50% din coeficienți

Curs 5 - SACCDMM - an I Master, Semestrul I

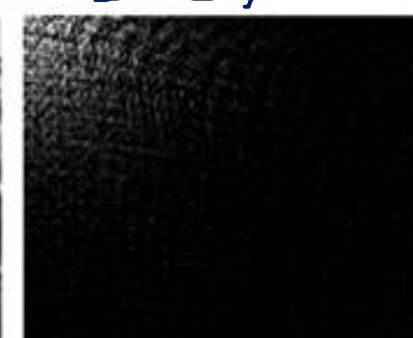


## DCT } Exemple cu privire la proprietatea de compactare a energiei pentru imagini standard

a)



d)



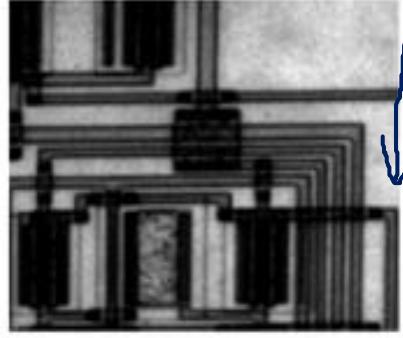
b)



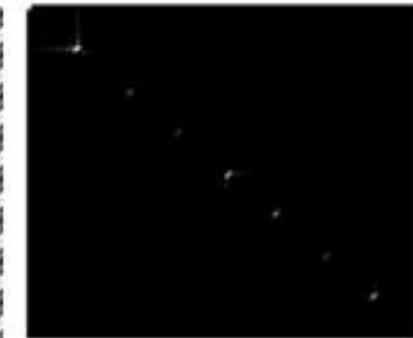
e)



c)



f)



# Exemplu DCT și IDCT ... cuantizarea!

Imagine:

$$\mathbf{u} = [186 \quad 184 \quad 187 \quad 187 \quad 189 \quad 187 \quad 190 \quad 190]^T$$

- Coeficientii DCT:

$$\mathbf{v} = [530.33 \quad -4.65 \quad 0 \quad -0.52 \quad 1.14 \quad 0.99 \quad 0 \quad 2.25]^T$$

- Dacă se păstrează toți coeficientii se poate reface perfect setul de date inițial (fără erori)

Coeficientii Hadamard	530,33	-4,24	0	-2,83	1,41	0	0	1,41
Coeficientii DCT	530,33	-4,65	0	-0,52	1,41	1	0	2,26

Hadamard Cuantizare 1	530	-4	0	-3	1	0	0	1
Hadamard Cuantizare 2	530	-4	0	-3	0	0	0	0
DCT Cuantizare 1	530	-5	0	-1	1	1	0	2
DCT Cuantizare 2	530	-5	0	0	0	0	0	2

Imagine refăcută								
186	184	187	187	189	187	190	190	
185	185	187	187	188	188	190	190	
185	184	187	187	189	187	190	190	
185	185	187	186	189	188	190	190	

# Exemple DCT-2D corelat



$U =$

159	163	161	161	162	159	161	161
159	161	161	159	161	159	161	161
162	162	159	158	159	162	162	158
159	161	158	156	161	162	162	161
156	159	161	158	161	162	162	161
161	159	162	158	158	161	161	159
162	159	161	158	158	158	161	158
158	158	159	159	158	159	161	158

$V =$

1279	-1,88	1,915	1,244	-4,87	-0,15	-0,93	2,757
4,271	-0,07	-0,35	-0,44	0,638	-3,74	-2,33	-0,6
-0,76	2,946	-0,87	-2,71	2,146	1,652	-1,67	-0,05
1,737	0,208	-1,23	-2,83	0,383	-0,21	1,213	1,05
-0,12	-3,86	-2,08	-0,93	0,125	-2,44	-1,44	-0,5
0,559	2,261	2,14	1,882	0,381	0,093	-0,74	-0,02
0,45	1,282	0,08	1,485	-1,02	1,141	0,369	-1,65
0,34	0,486	-0,89	-0,03	-0,49	0,75	-0,4	-0,2

1279	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$Ur =$

160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160

$D =$

1	0	0	0	0	0	1	0	0
1	0	0	1	0	1	0	1	0
0	0	1	2	1	0	0	0	2
1	0	2	4	0	0	0	0	0
4	1	0	2	0	0	0	0	0
0	1	0	2	2	0	0	1	0
0	1	0	2	2	2	0	2	0
2	2	1	1	2	1	0	2	0

1279	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$Ur' =$

160	161	161	160	160	161	161	160
160	161	161	160	160	161	161	160
160	161	161	160	160	161	161	160
159	161	161	159	159	161	161	159
159	160	160	159	159	160	160	159
159	160	160	159	159	160	160	159
159	160	160	159	159	160	160	159
159	160	160	159	159	160	160	159

$D' =$

1	0	0	1	0	2	0	0	0
0	0	2	2	1	0	0	2	0
0	0	3	3	0	0	0	0	0
3	1	0	1	0	0	0	0	0
0	1	0	1	1	0	0	0	0
0	1	0	1	1	2	0	0	1
1	2	1	0	1	1	0	1	0

# Bloc cu energie mare

~~U = V = D =~~

~~→~~

116	121	120	133	159	172	182	186
114	107	103	130	156	168	182	187
106	99	93	114	141	158	175	176
106	99	91	98	126	144	162	161
103	98	91	88	109	131	142	142
110	107	99	81	88	119	120	113
137	137	124	98	81	100	107	99
142	126	109	103	91	92	109	109

986,3	-112	61,55	29,15	-10,5	1,60	-4,93	0,79
117,2	-127	-23,8	14,67	16,93	-8,82	-5,84	2,52
38,1	43,9	-5,61	-12,8	6,40	3,07	-3,19	-2,78
-11	10,01	-3,91	-1,98	-17	-4,77	8,36	-1,08
4	-7,37	2,10	-0,48	6,25	-2,32	-2,76	4,05
4,44	17,39	1,17	-5,06	-10,4	-0,44	0,87	0,17
-9,29	-9,73	-0,69	3,92	-0,22	-0,71	1,10	0,19
3,375	4,23	-0,74	-3,84	-1,25	-0,13	-0,42	-0,52

U=

V=

986	-112	62	29	-10	0	0	0
117	-127	-24	15	17	0	0	0
38	44	0	-13	0	0	0	0
-11	10	0	0	-17	0	0	0
0	0	0	0	0	0	0	0
0	17	0	0	-10	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

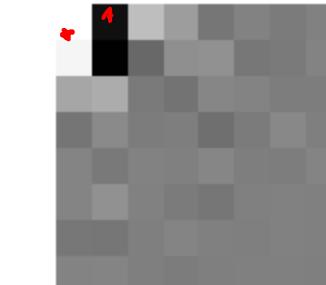
Ur=

120	121	122	131	152	174	183	182
113	102	105	128	154	170	182	192
109	96	96	117	143	160	172	180
105	96	91	101	124	146	156	157
102	96	88	90	108	131	142	140
114	113	100	85	92	114	123	116
133	133	116	90	85	100	108	101
141	132	116	99	90	94	103	110

4	0	2	0	0	2	0	5
0	0	2	0	0	2	0	4
3	0	3	3	2	2	0	0
0	0	0	3	0	2	0	0
0	0	0	2	0	0	0	0
4	6	1	4	4	0	3	3
0	0	0	0	4	0	1	2
0	6	7	0	0	2	0	1



Imagine originală



Imagine DCT

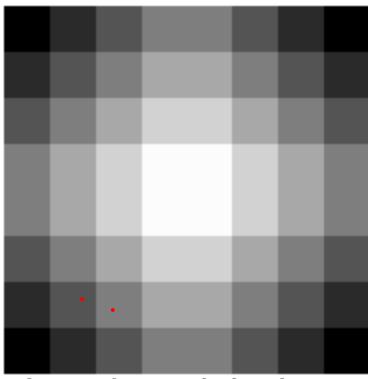


Imagine refăcută



DCT Cuantizat

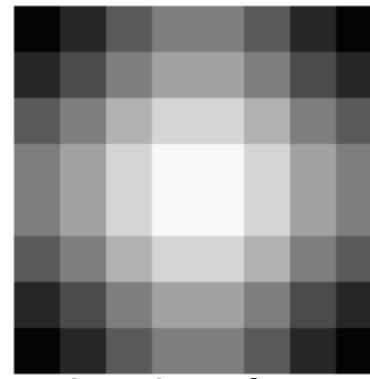
# Exemple DCT-2D nivele de gri-binare



Imagine originală



Imagine DCT



Imagine refăcută



DCT Cuantizați

240	0	-89,22	0	0	0	-6,34	0
0	0	0	0	0	0	0	0
-89,22	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-6,34	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

V=

2400	0	-89	0	0	0	0	0
0	0	0	0	0	0	0	0
-89	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Vc=

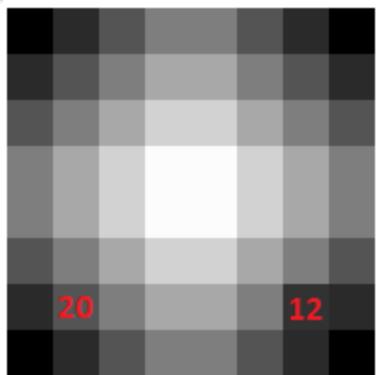
0	10	20	30	30	20	10	0
10	20	30	40	40	30	20	10
20	30	40	50	50	40	30	20
30	40	50	60	60	50	40	30
30	40	50	60	60	50	40	30
20	30	40	50	50	40	30	20
10	20	30	40	40	30	20	10
0	10	20	30	30	20	10	0

Ur=

1	9	21	30	30	21	9	1
9	18	30	39	39	30	18	9
21	30	42	51	51	42	30	21
30	39	51	59	59	51	39	30
30	39	51	59	59	51	39	30
21	30	42	51	51	42	30	21
9	18	30	39	39	30	18	9
1	9	21	30	30	21	9	1

D=

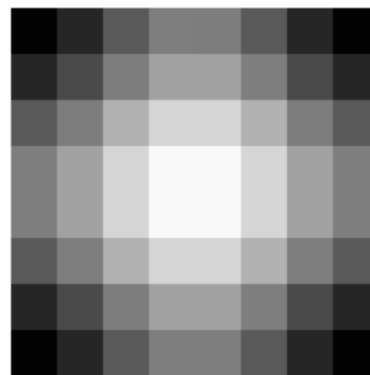
1	0	1	0	0	1	0	1
0	0	0	0	0	0	0	0
1	0	2	1	1	2	0	1
0	0	1	0	0	1	0	0
1	0	2	1	1	2	0	1
0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	1
1	0	1	0	0	1	0	1



Imagine originală



Imagine DCT



Imagine refăcută



DCT Cuantizată

239,00	1,18	-89,76	-0,28	1,00	-1,39	-5,03	-0,79
1,18	-1,38	0,64	0,32	-1,18	1,63	-1,54	0,92
-89,76	0,64	-0,29	-0,15	0,54	-0,75	0,71	-0,43
-0,28	0,32	-0,15	-0,08	0,28	-0,38	0,36	-0,22
1,00	-1,18	0,54	0,28	-1,00	1,39	-1,31	0,79
-1,39	1,63	-0,75	-0,38	1,39	-1,92	1,81	-1,09
-5,03	-1,54	0,71	0,36	-1,31	1,81	-1,71	1,03
-0,79	0,92	-0,43	-0,22	0,79	-1,09	1,03	-0,62

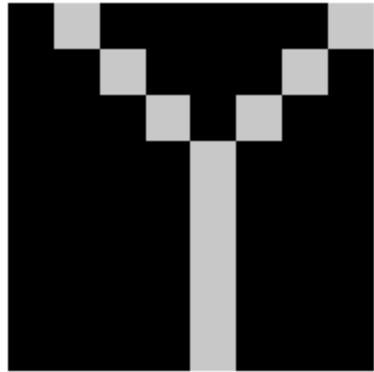
V=	239	0	-90	0	0	0	0	0
	0	0	0	0	0	0	0	0
	-90	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

U=	0	10	20	30	30	20	10	0
	10	20	30	40	40	30	20	10
	20	30	40	50	50	40	30	20
	30	40	50	60	60	50	40	30
	30	40	50	60	60	50	40	30
	20	30	40	50	50	40	30	20
	10	20	30	40	40	30	12	10
	0	10	20	30	30	20	10	0

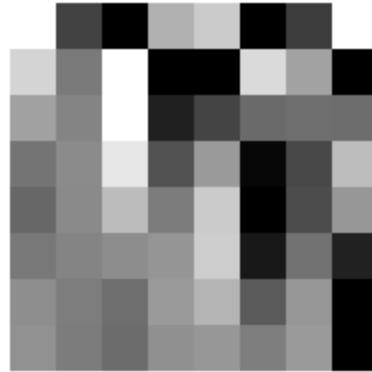
Ur=	0	9	21	30	30	21	9	0
	9	18	30	38	38	30	18	9
	21	30	42	51	51	42	30	21
	30	38	51	59	59	51	38	30
	30	38	51	59	59	51	38	30
	21	30	42	51	51	42	30	21
	9	18	30	38	38	30	18	9
	0	9	21	30	30	21	9	0

Vc=	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	-90	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

D=	0	0	1	0	0	1	0	0
	0	0	0	0	0	0	0	0
	1	0	2	1	1	2	0	1
	0	0	1	0	0	1	0	0
	0	0	1	0	0	1	0	0
	1	0	2	1	1	2	0	1
	0	0	0	0	0	0	6	0
	0	0	1	0	0	1	0	0



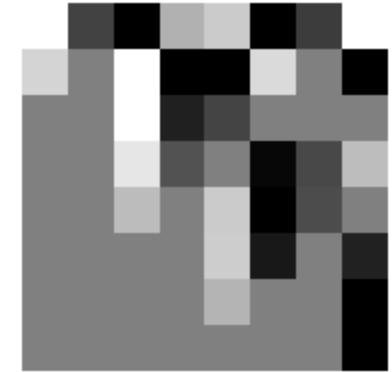
Imagine originală



Imagine DCT



Imagine refăcută



DCT Cuantizată

$V =$

275	-62,27	-163,3	49,173	75	-137,2	-67,65	131,81
83,715	-5,709	137,16	-137	-142,5	89,66	33,8	-145,6
32,664	3,7329	128,03	-96,25	-59,72	-22,24	-17,68	-18,77
-12,18	10,802	102,48	-46,3	25,972	-120,9	-55,63	61,024
-25	10,323	59,724	-3,907	75	-147,8	-51,8	23,17
-8,136	3,6211	12,919	20,554	77,488	-103,7	-14,16	-95,26
13,53	-3,031	-17,68	25,664	51,798	-38,41	21,967	-180,9
16,652	-4,185	-19,82	16,406	22,633	-1,613	26,232	-144,3

$V_C =$

275	-62	-163	49	75	-137	-68	132
84	0	137	-137	-143	90	0	-146
0	0	128	-96	-60	0	0	0
0	0	102	-46	0	-121	-56	61
0	0	60	0	75	-148	-52	0
0	0	0	0	77	-104	0	-95
0	0	0	0	52	0	0	-181
0	0	0	0	0	0	0	-144

$U =$

0	200	0	0	0	0	0	200
0	0	200	0	0	0	200	0
0	0	0	200	0	200	0	0
0	0	0	0	200	0	0	0
0	0	0	0	0	200	0	0
0	0	0	0	0	0	200	0
0	0	0	0	0	0	0	200
0	0	0	0	0	0	0	0

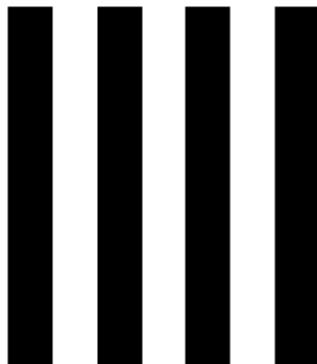
$U_r =$

0	199	3	5	0	0	20	196
7	0	188	0	21	2	189	0
0	9	0	204	0	155	10	0
7	9	2	24	226	27	0	18
0	35	0	0	199	14	13	6
3	0	9	3	195	0	8	1
0	0	20	0	203	9	0	0
11	0	0	0	188	0	0	0

$D =$

0	0	3	5	0	0	20	0
7	0	0	0	21	2	0	0
0	9	0	4	0	0	10	0
7	9	2	24	26	27	0	18
0	35	0	0	0	14	13	6
3	0	9	3	0	0	8	1
0	0	20	0	3	9	0	0
11	0	0	0	0	0	0	0

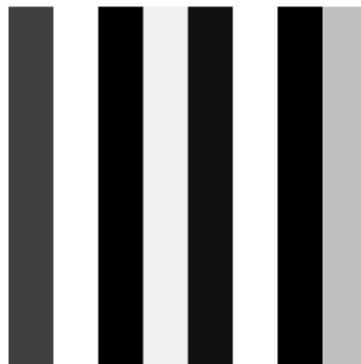
# Coeficienții DCT-2D



Imagine originală



Imagine DCT



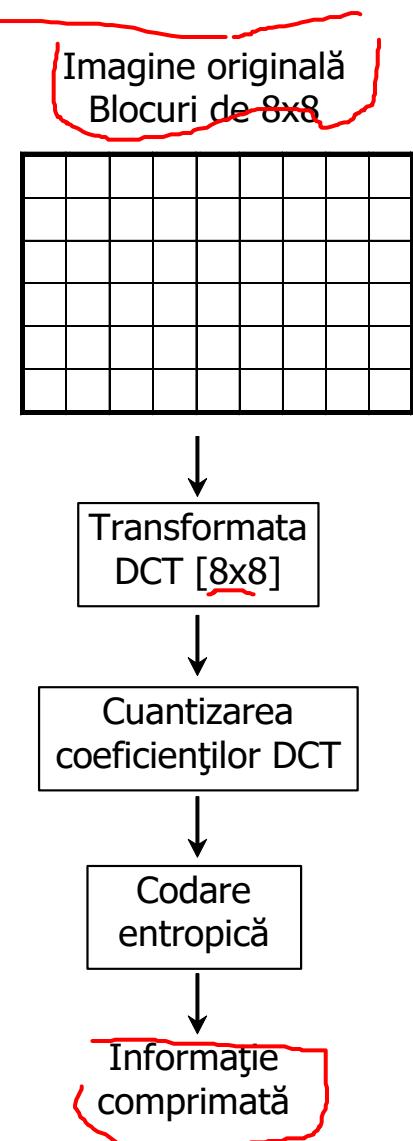
Imagine refăcută



DCT Cuantizați

# Sistem pentru compresia imaginilor prin transformări - DCT

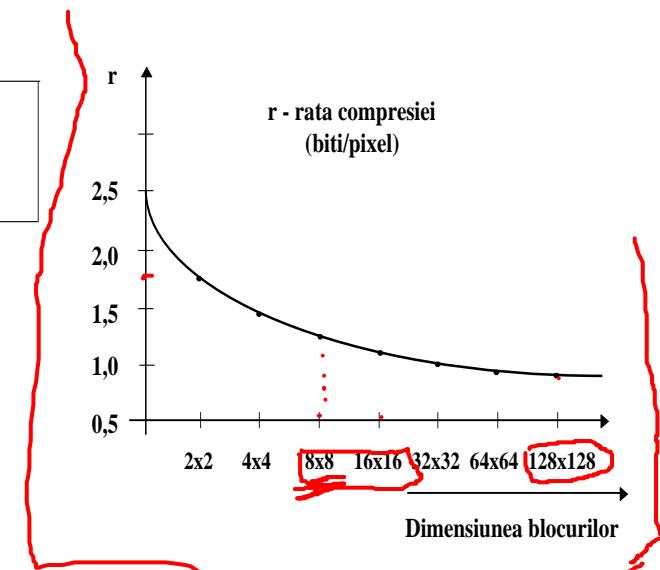
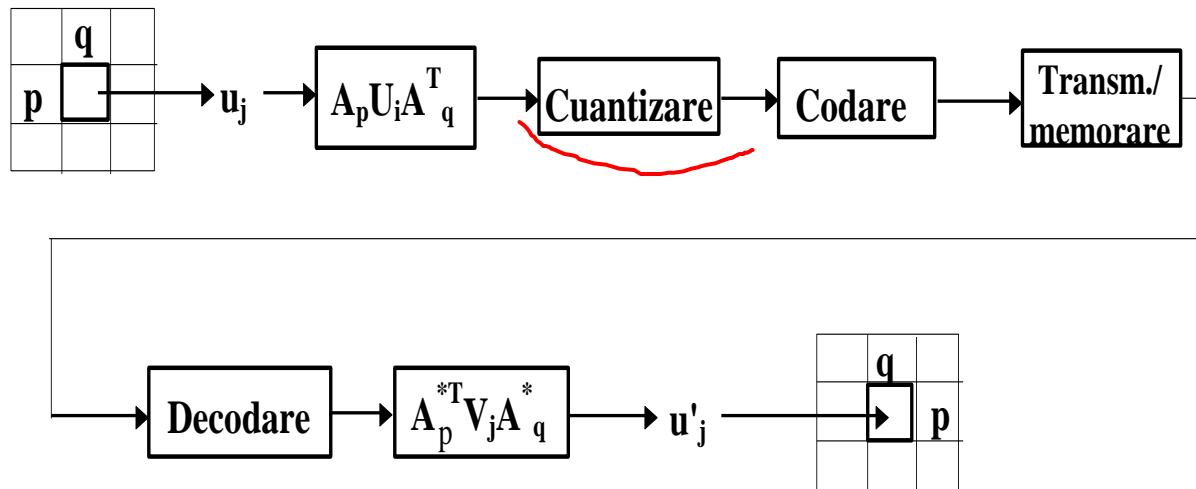
- exploatează capacitatea de compactare a energiei într-un număr mic de coeficienți semnificativi (neneglijabili)
  - **DCT 2-D** este varianta optimala pentru imagini naturale
    - ca timp de calcul și eficiența compactării energiei
- **Sistem de compresie prin transformări**,
  - aplicare transformare asupra întregii imagini
    - dezavantaj - timp de calcul și memorii ocupate mari (dată de dimensiunea imaginilor)
  - divizare imagine în blocuri de  $N \times N$  pixeli (tipic,  $8 \times 8$  sau  $16 \times 16$  pixeli) și **aplica transformata asupra fiecărui bloc**
    - necesar mic de memorie și viteza de calcul mare
    - un oarecare dezavantaj este o diminuare a compactării energiei per ansamblu față de cazul aplicării transformatei asupra întregii imagini



# Codarea prin transformare bidimensională

- divizarea imaginii de intrare

- $H \times W$  → rezulta  $HW/pq$  blocuri de dimensiune  $pxq$
- se reduce numărul de operații necesare pentru calculul transformatei (inclusiv prin implementarea algoritmilor rapizi de calcul)



# Codarea prin transformare bidimensională

- alocarea bitilor
- alegerea cuantizorului (coef. CC, CA)
- codarea ieșirii cuantizorului
- reproducerea coeficienților (canal fără zgomot)

Tabelul 3.1

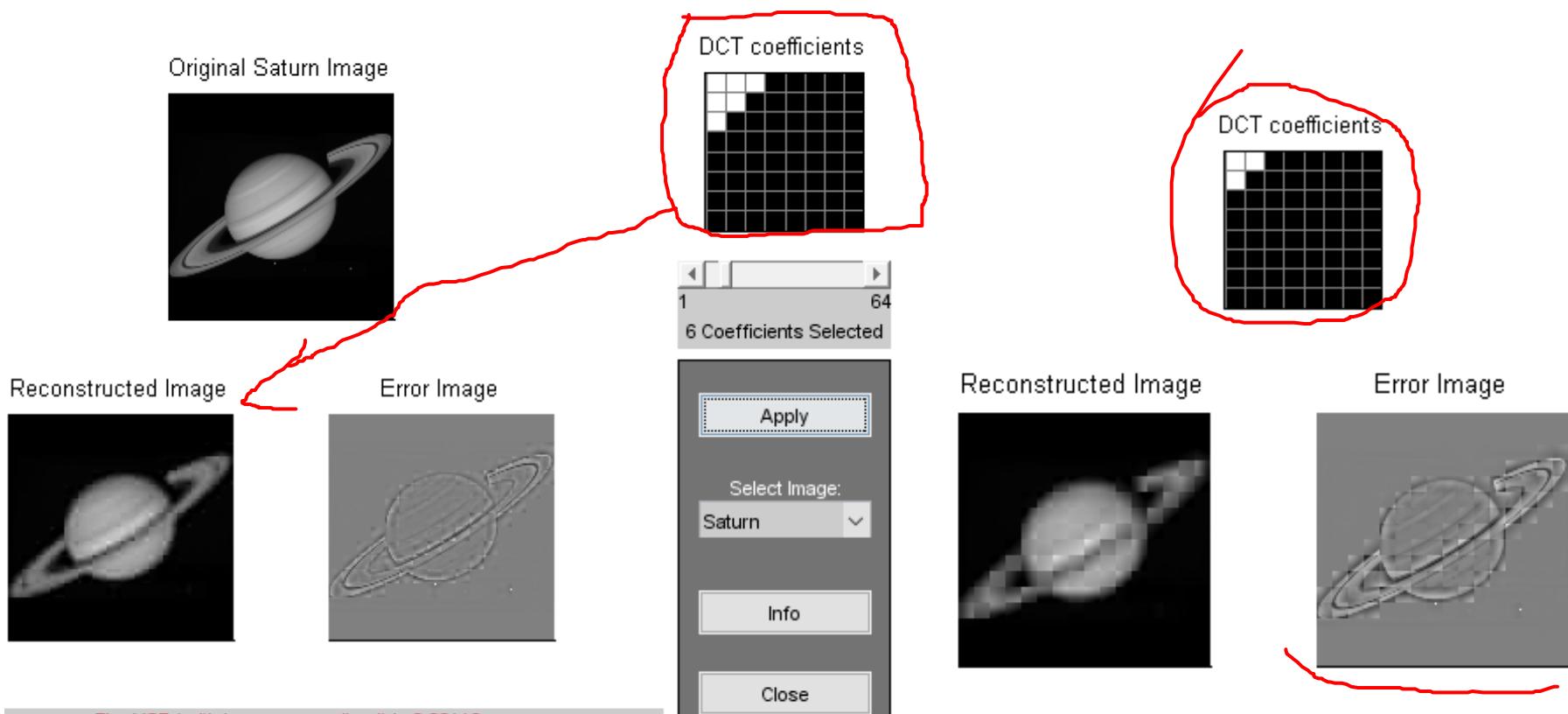
Mărimea blocului	Rata de compresie Biti/pixel	Raport Semnal-zgomot				
		KL	Cosinus	Sinus	Fourier	Hadamard
8x8	0.5	13.82	13.76	11.69	12.27	12.65
	1	16.24	16.19	14.82	14.99	15.19
	2	20.95	20.89	19.53	19.73	19.86
16x16	0.5	-	14.25	12.82	12.87	12.78
	1	-	16.58	15.65	15.52	15.27
	2	-	21.26	20.37	20.24	20.01

# Cuantizarea coeficientilor transformatei

- Cuantizarea coeficientilor transformatei
  - proces implicit cu **pierdere de informatie**
- trebuie ales cu atentie numarul de biti pe care se reprezinta/cuantizeaza în special coeficientii de înalta frecventa din domeniul transformat:
  - daca acest numar este prea mic, atunci eroarea introdusa poate cauza aparitia efectului "de bloc",
- efectului "de bloc"
  - la reconstructia blocurilor de  $8 \times 8$  sau  $16 \times 16$  pixeli prin invers cuantizare si IDCT 2-D, lipsesc detaliile din aceste blocuri si se observa "patratele" pe nivele de gri în locul unei zone continue de nivele de gri din imagine.

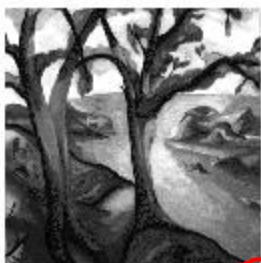
# Efectului "de bloc,,

- Efectului "de bloc "
  - se poate observa cu aplicatia DCTDemo, din Matlab, pentru diferiti coeficienti ai transformatiei setati la 0 (echivalent cu cuantizarea grosiera a diferitelor frecvente din domeniul transformat).

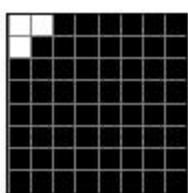


The MSE (with images normalized) is 0.00112 .

Original Trees Image



DCT coefficients

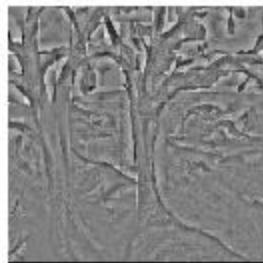


1 64  
3 Coefficients Selected

Reconstructed Image



Error Image

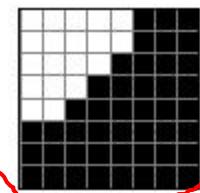


The MSE (with images normalized) is 0.0167 .

Original Trees Image

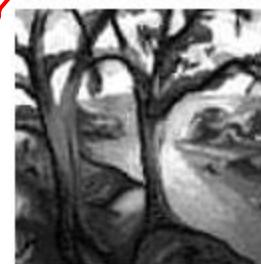


DCT coefficients

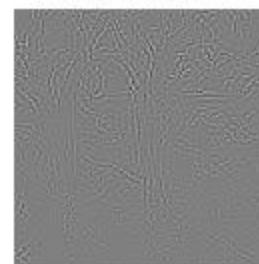


1 64  
19 Coefficients Selected

Reconstructed Image



Error Image



The MSE (with images normalized) is 0.00381 .

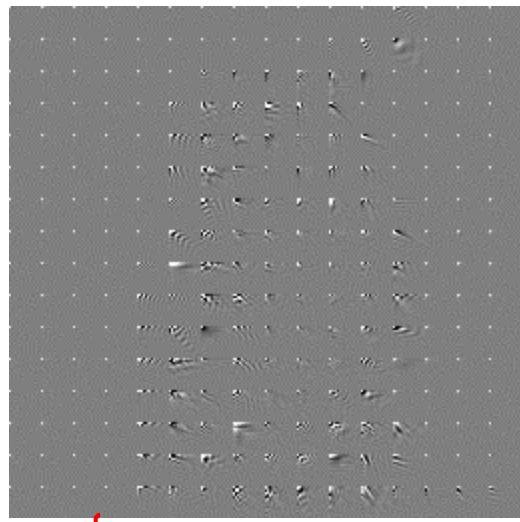
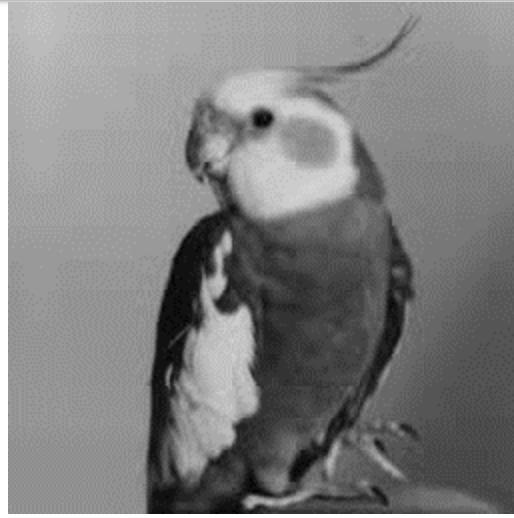
Apply

Select Image:

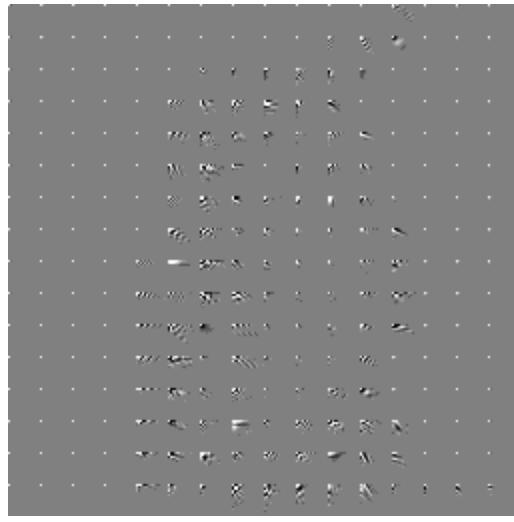
Trees

Info

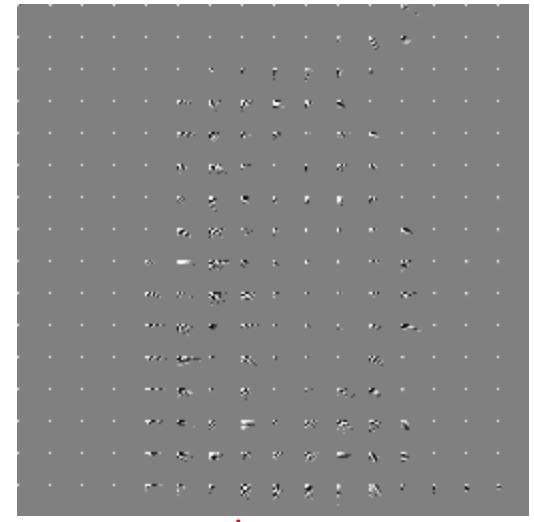
Close



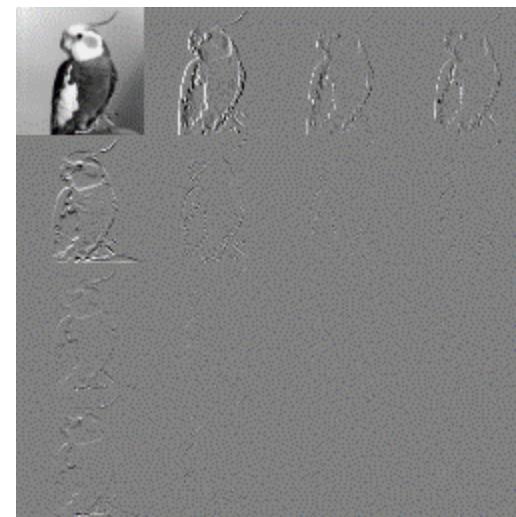
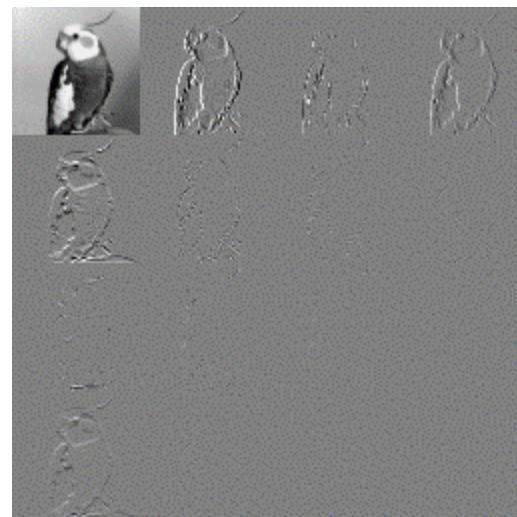
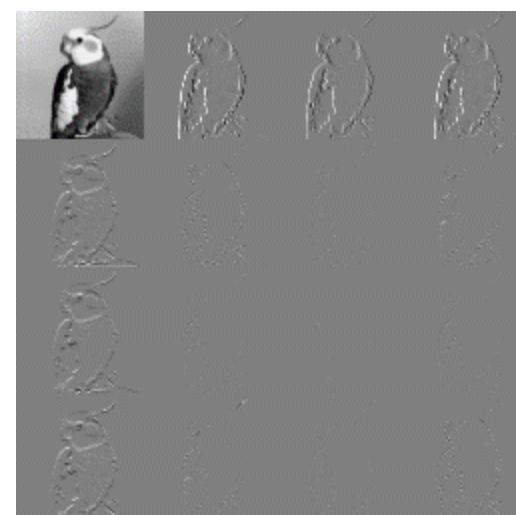
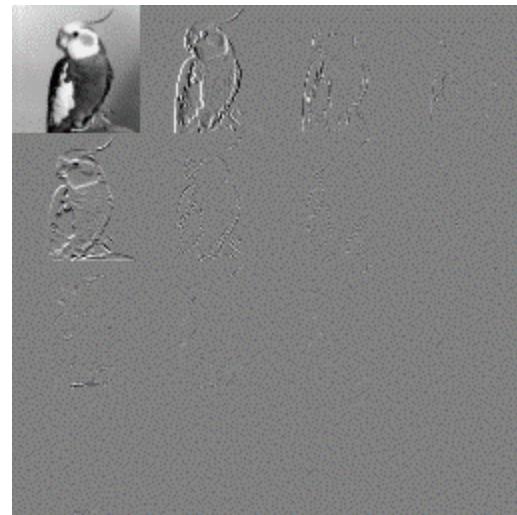
85% zero-uri



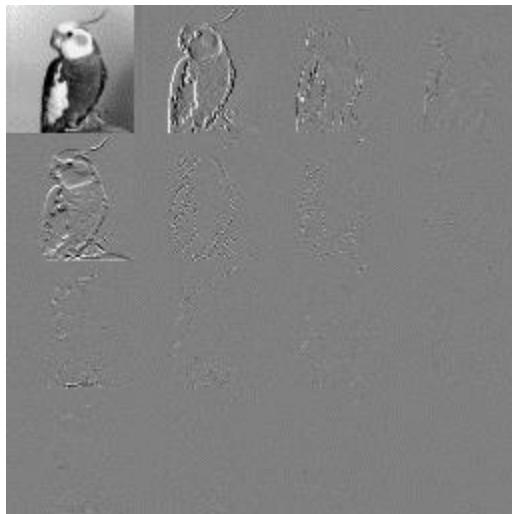
96% zero-uri



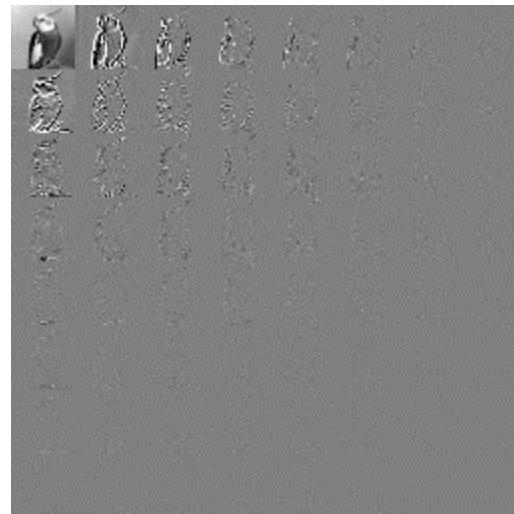
98%



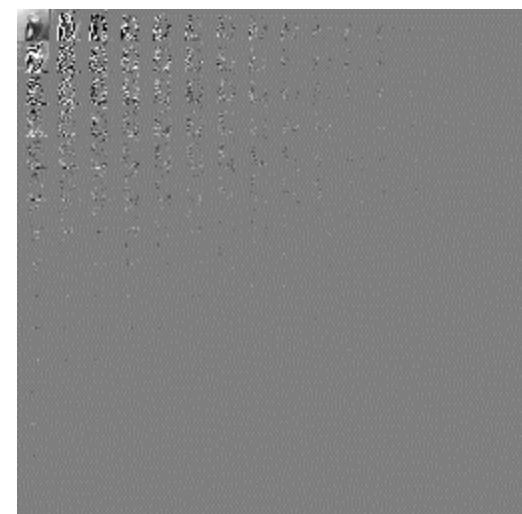
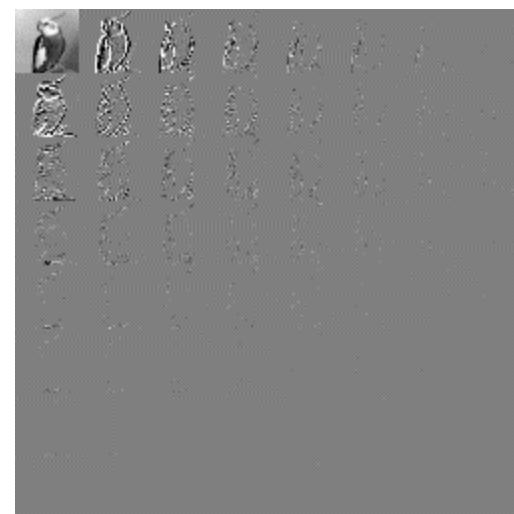
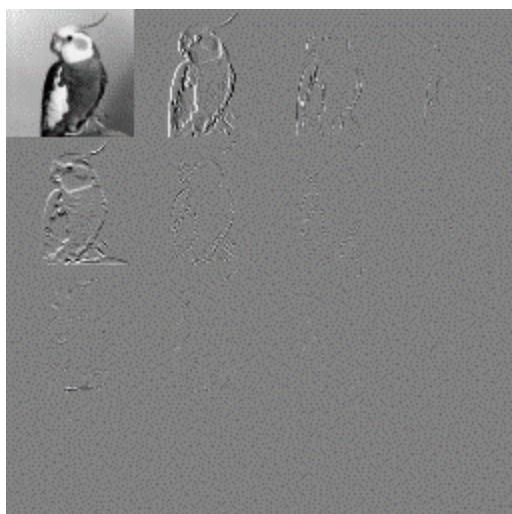
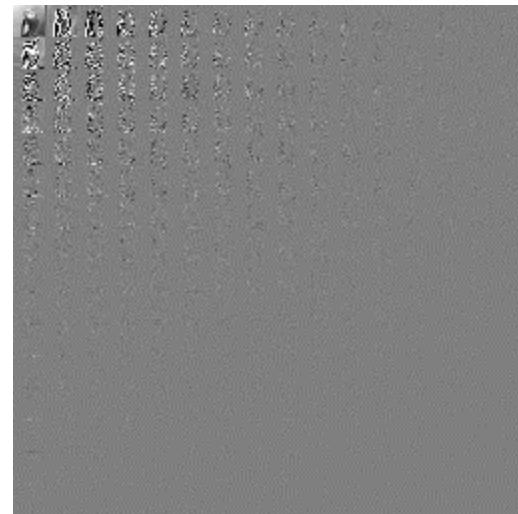
$4 \times 4$

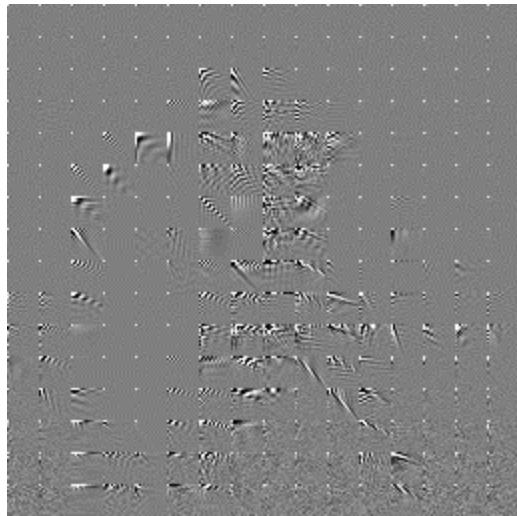


$\downarrow$   
 $8 \times 8$

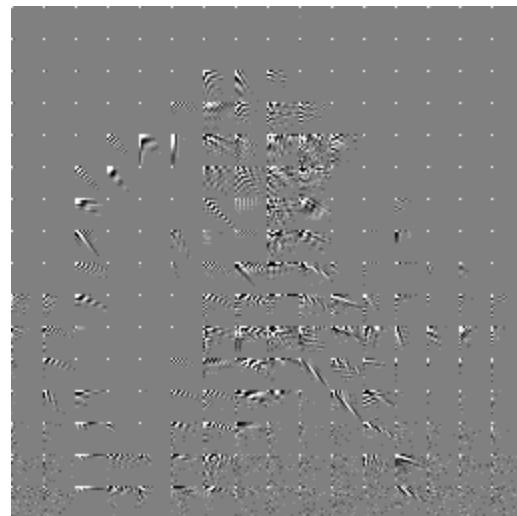


$16 \times 16$

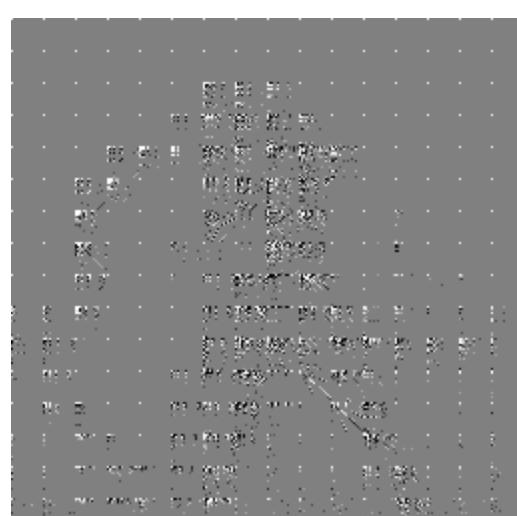




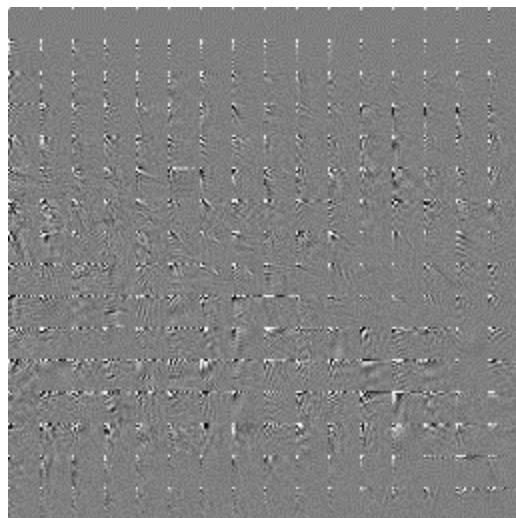
63% zero-uri



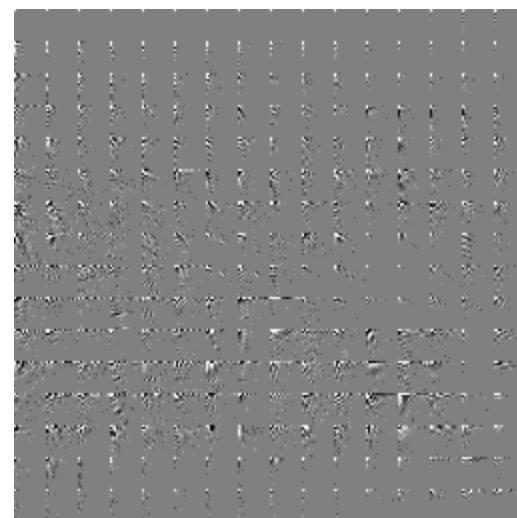
88% zero-uri



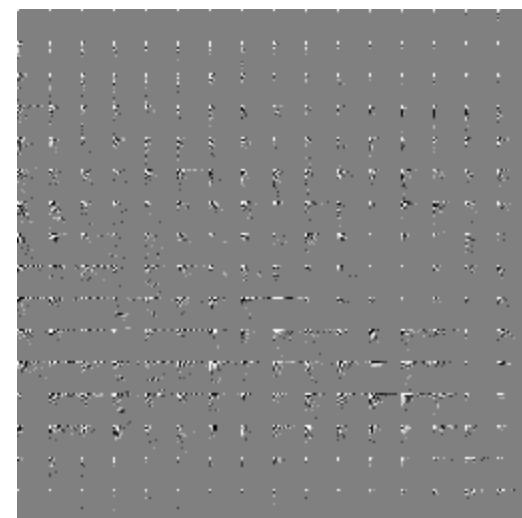
95%



44% zero-uri



87% zero-uri



95%

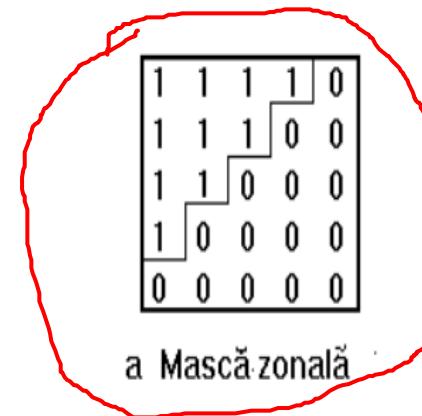
# Codarea zonală și codarea cu prag

- doar o mica zonă din imaginea transformată se transmite (allocarea biților)

- codarea zonală**

- masca zonală

$$m(k,l) = \begin{cases} 1, & k,l \in I_t \\ 0, & \text{in rest} \end{cases}$$

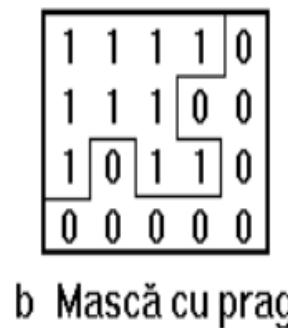


- codarea cu prag

- pragul
- masca cu prag

$$I_t = \{k,l : |v(k,l)| > \eta\}$$

$$m_\eta(k,l) = \begin{cases} 1, & k,l \in I_t \\ 0, & \text{in rest} \end{cases}$$



- Cuantizoare uniforme**

- ordonarea "zig-zag" a coeficientilor

# Cuantizoarele din schemele de compresie bazate pe transformari

- Cuantizoare uniforme
    - definite în mod unic prin numarul de biti alocat fiecarui coeficient din domeniul transformat
    - pentru un bloc de dimensiune  $p \times p$ , cu  $p$  – oarecare (în general 8 sau 16), cuantizoarele pot fi specificate printr-o **matrice de cuantizare**  $\mathbf{Q}$  de aceeasi dimensiune  $p \times p$ , unde elementele din matrice reprezinta **intervalul de cuantizare** alocat cuantizorului coeficientului respectiv al transformantei (coeficientul de c.c. sau unul din coeficientii de c.a.).
    - procesul de cuantizare constă pur și simplu în divizarea valorilor coeficientilor transformatei imaginii la valoarea corespunzatoare din matricea de cuantizare (adica, la intervalul de cuantizare = pasul de cuantizare)
  - Alegerea numărului de nivele de cuantizare (intervalului de cuantizare pentru fiecare cuantizor în parte),
    - se realizează în funcție de varianța fiecarui coeficient al transformantei imaginii, estimată pentru/dintr-un set de imagini tipice pentru aplicatia de codare considerata:
      - cu cât varianța unui coeficient este mai mare, cu atât numarul de nivele de cuantizare necesar pentru reprezentarea fidela a coeficientului și ca urmare și a imaginii reconstruite (decodate) este mai mare.
- => procedura poartă numele de determinarea alocării bitilor.

- În general, există obținute, din date experimentale, seturi de matrici de cuantizare recomandate pentru diferite categorii de imagini, pentru dimensiuni ale blocurilor de imagine de  $8 \times 8$  sau  $16 \times 16$  pixeli.
- Valorile coeficientilor acestor matrici se pot însă ajusta în schemele de compresie pentru aplicații practice la specificul imaginilor comprimate în aplicația respectiva.

$$Q_1 = \begin{bmatrix} 8 & 36 & 36 & 36 & 39 & 45 & 52 & 65 \\ 36 & 36 & 36 & 37 & 41 & 47 & 56 & 58 \\ 36 & 36 & 38 & 42 & 47 & 54 & 64 & 78 \\ 36 & 37 & 42 & 50 & 59 & 69 & 81 & 98 \\ 39 & 41 & 47 & 54 & 73 & 89 & 108 & 130 \\ 45 & 47 & 54 & 69 & 89 & 115 & 144 & 178 \\ 53 & 56 & 64 & 81 & 108 & 144 & 190 & 243 \\ 65 & 68 & 78 & 98 & 130 & 178 & 243 & 255 \end{bmatrix}, Q_2 = \begin{bmatrix} 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \\ 17 & 19 & 21 & 23 & 25 & 27 & 29 & 31 \end{bmatrix}$$

$Q_3(niv.calit.50\%) =$

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

$Q_4(niv.calit.10\%) =$

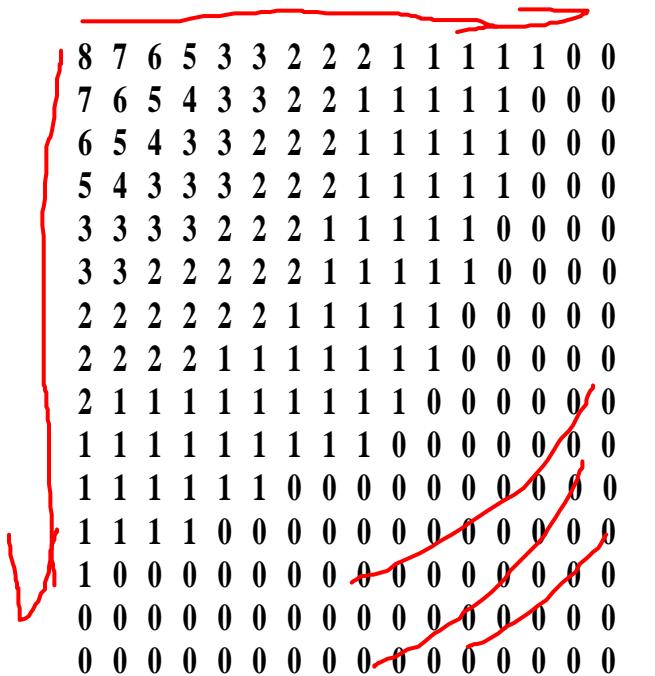
80	60	50	80	120	200	255	255
55	60	70	95	130	255	255	255
70	65	80	120	200	255	255	255
70	85	110	145	255	255	255	255
90	110	185	255	255	255	255	255
120	175	255	255	255	255	255	255
245	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

$Q_5(niv.calit.90\%) =$

3	2	2	3	5	8	10	12
2	2	3	4	5	12	12	11
3	3	3	5	8	11	14	11
3	3	4	6	10	17	16	12
4	4	7	11	14	22	21	15
5	7	11	13	16	12	23	18
10	13	16	17	21	24	24	21
14	18	19	20	22	20	20	20

# Alocarea bitilor

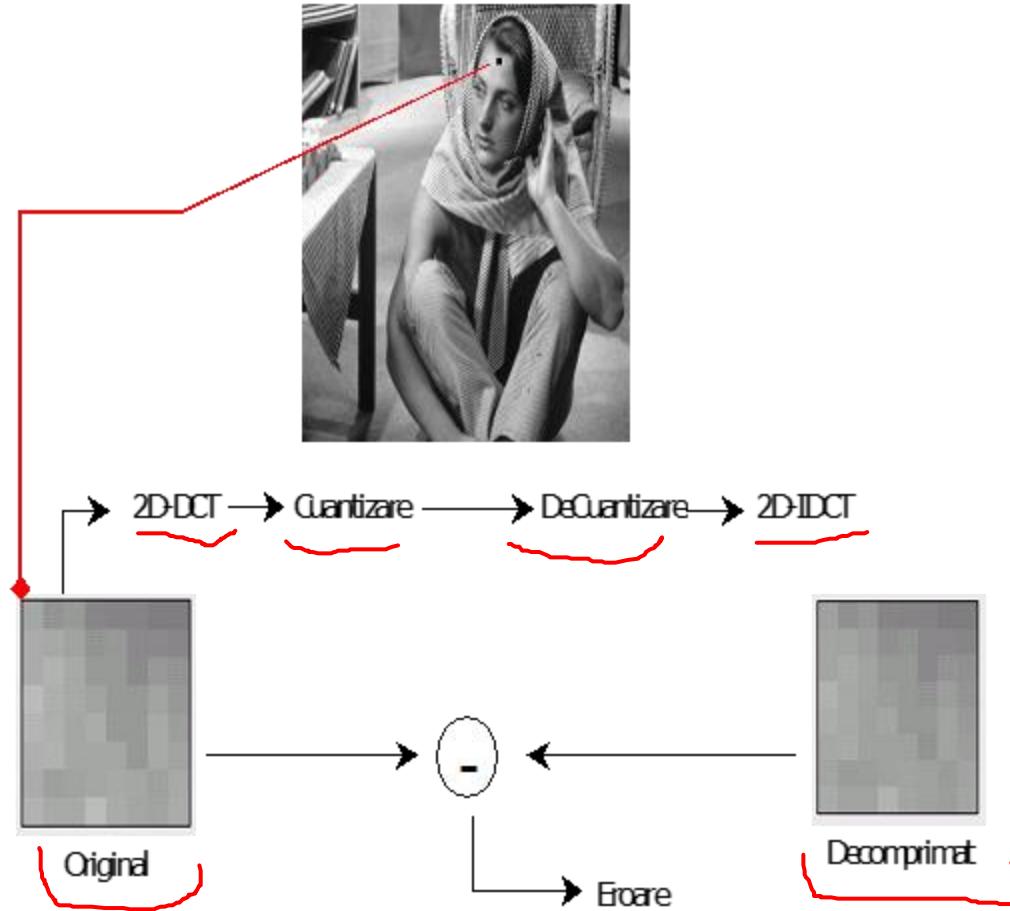
- dispersia coeficienților nu este uniformă
- cuantizare cu număr diferit de biți
- numărul biților de cuantizare este  $n_k$  întreg



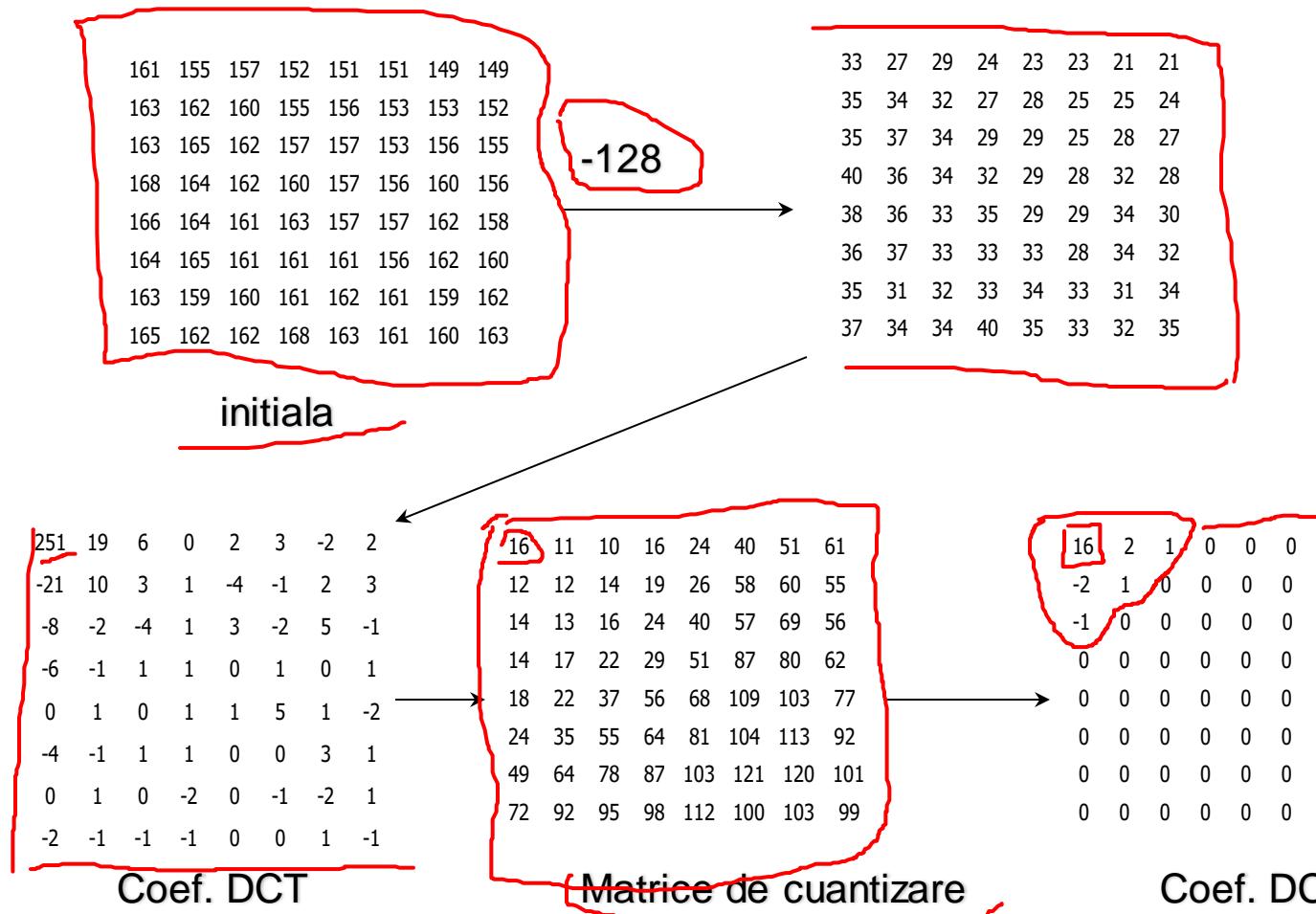
alocarea bitilor pentru DCT pe fereastra de 16x16 pixeli pentru o rata medie de 1bit/pixel

- Ultima etapa din schemele de codare prin transformari constă în:
  - (1) ordonarea coeficientilor cuantizati ai transformatei după o regula prestabilită (scopul fiind în general gruparea coeficientilor cuantizati care au valoarea 0 într-un sir cât mai compact, deoarece acesta poate fi omis la transmisie/stocare) (ex. pt. transformata DCT 2-D, se foloseste ordonarea în zig-zag);
  - (2) codarea entropica (Huffman, codare aritmetica) a coeficientilor cuantizati nenuli, care urmează a fi transmisi/stocati – realizată în standardul JPEG pe baza unor tabele de codare prestabilite în urma experimentelor pe un numar mare de imagini – tabele standardizate de codare.

# Schemă codare DCT



# Codarea DCT



# Codarea DCT

256	22	10	0	0	0	0	0
-24	12	0	0	0	0	0	0
-14	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

IDCT; +128

162	160	157	153	151	149	149	148
163	162	158	155	153	151	151	151
166	164	161	158	156	155	155	155
167	166	163	161	159	158	158	159
168	167	164	162	161	161	161	162
167	166	164	162	161	161	162	163
166	164	163	161	161	161	162	163
164	163	162	160	160	161	162	163

1	5	0	1	0	-2	0	-1
0	0	-2	0	-3	-2	-2	-1
3	-1	-1	1	-1	2	-1	0
-1	2	1	1	2	2	-2	3
2	3	3	-1	4	4	-1	4
3	1	3	1	0	5	0	3
3	5	3	0	-1	0	3	1

$$erroarea = 20 \log_{10} \left( \frac{255}{\sqrt{\frac{1}{MN} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} [X(x,y) - \hat{X}(x,y)]^2}} \right)$$

40.67 dB

# Algoritmi de calcul rapid al DCT

Metoda DCT sau IDCT	Înmulțiri		Adunări	
	1D	2D	1D	2D
1D Chen	16	256	26	416
1D Lee	12	192	29	464
1D Loeffler	11	176	29	464
2D Kamangar		128		430
2D Cho, Lee		96		466

- DCT 2D 8x8 pixeli clasic avem 1024 înmulțiri și 896 adunări

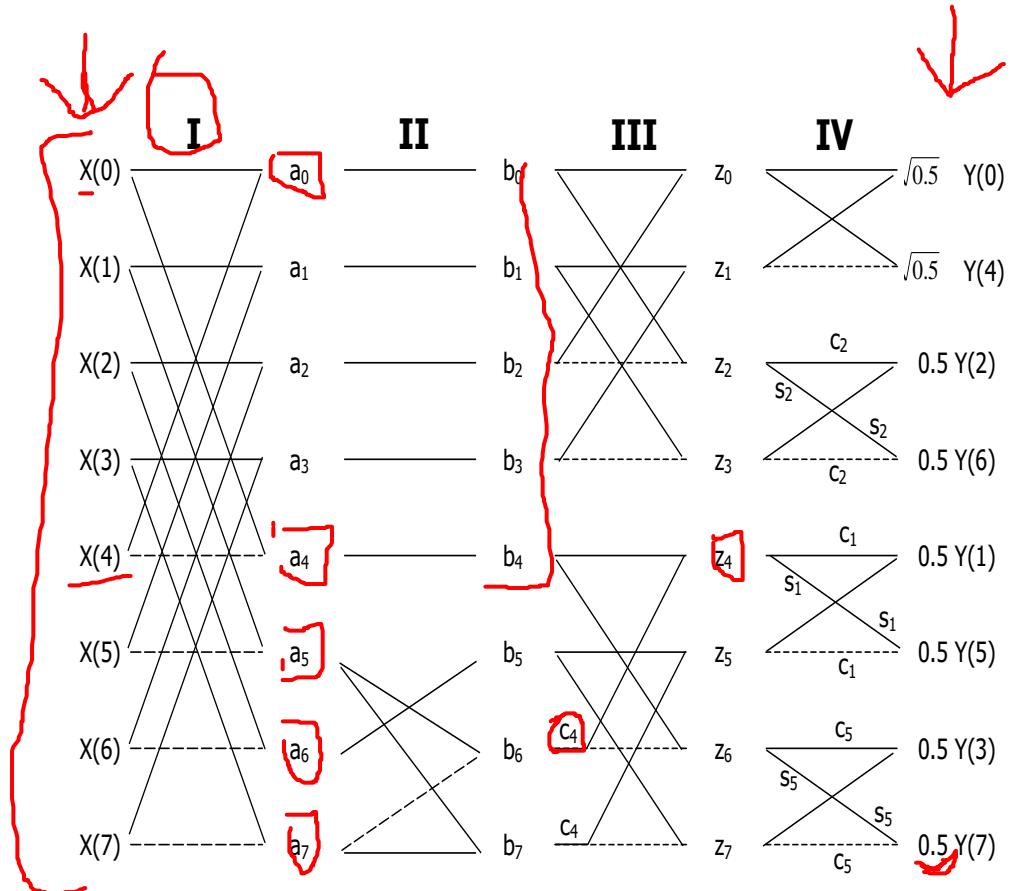
# Algoritmi de calcul rapid al DCT

- Algoritmul II – CHEN
- Pasul 1

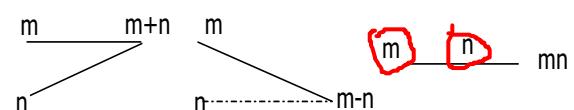
$$\begin{array}{ll}
 a_0 = X(0) + X(4) & a_4 = X(0) - X(4) \\
 a_1 = X(1) + X(5) & a_5 = X(1) - X(5) \\
 a_2 = X(2) + X(6) & a_6 = X(2) - X(6) \\
 a_3 = X(3) + X(7) & a_7 = X(3) - X(7)
 \end{array}$$

- Pasul 2

$$\begin{array}{ll}
 b_0 = a_0 & b_4 = a_4 \\
 b_1 = a_1 & b_5 = a_6 \\
 b_2 = a_2 & b_6 = a_5 - a_7 \\
 b_3 = a_3 & b_7 = a_5 + a_7
 \end{array}$$



Unde avem următoarele notății:



# Algoritmi de calcul rapid al DCT

- Algoritmul II – CHEN ... rezultă **16 înmulțiri și 26 adunări**
- Pasul 3

$$\begin{array}{ll} z_0 = b_0 + b_2 & z_4 = b_4 + c_4 b_6 \\ z_1 = b_1 + b_3 & z_5 = b_5 + c_4 b_7 \\ z_2 = b_0 - b_2 & z_6 = b_4 - c_4 b_6 \\ z_3 = b_1 - b_3 & z_7 = b_5 - c_4 b_7 \end{array}$$

- Pasul 4

$$\begin{array}{ll} \sqrt{0.5} \cdot Y(0) = z_0 + z_1 & 0.5 \cdot Y(1) = z_4 c_1 + z_5 s_1 \\ \sqrt{0.5} \cdot Y(4) = z_0 - z_1 & 0.5 \cdot Y(5) = z_4 s_1 - z_5 c_1 \\ 0.5 \cdot Y(2) = z_2 c_2 + z_3 s_2 & 0.5 \cdot Y(3) = z_6 c_5 + z_7 s_5 \\ 0.5 \cdot Y(6) = z_2 s_2 - z_3 c_2 & 0.5 \cdot Y(7) = z_6 s_5 - z_7 c_5 \end{array}$$

# Algoritmi de calcul rapid al DCT

- 1D DCT pe 8 puncte – 64 înmulțiri; 56 adunări
- Algoritmul I – simetric

$$s_k = \sin \frac{k\pi}{16}; \quad \text{respectiv} \quad c_k = \cos \frac{k\pi}{16}$$

$$c_1 = s_7 = 0.9808$$

$$c_2 = s_6 = 0.9239$$

$$c_3 = s_5 = 0.8315$$

$$c_4 = s_4 = 0.7071$$

$$c_5 = s_3 = 0.5556$$

$$c_6 = s_2 = 0.3827$$

$$c_7 = s_1 = 0.1951$$

$$\boxed{s_{jk}} = \underline{s(j) + s(k)}$$

$$s_{07} = \underline{s(0) + s(7)}$$

$$s_{16} = \underline{s(1) + s(6)}$$

$$s_{25} = \underline{s(2) + s(5)}$$

$$s_{34} = \underline{s(3) + s(4)}$$

$$s_{0734} = s_{07} + s_{34}$$

$$s_{1625} = s_{16} + s_{25}$$

# Algoritmi de calcul rapid al DCT

- Algoritmul I – simetric

$$d_{jk} = s(j) - s(k)$$

$$d_{07} = s(0) - s(7)$$

$$d_{16} = s(1) - s(6)$$

$$d_{25} = s(2) - s(5)$$

$$d_{34} = s(3) - s(4)$$

$$d_{0734} = s_{07} - s_{34}$$

$$d_{1625} = s_{16} - s_{25}$$

$$2S(0) = c_4(s_{0734} + s_{1625})$$

$$2S(1) = c_1d_{07} + c_3d_{16} + c_5d_{25} + c_7d_{34}$$

$$2S(2) = c_2d_{0734} + c_6d_{1625}$$

$$2S(3) = c_3d_{07} - c_7d_{16} - c_1d_{25} - c_5d_{34}$$

$$2S(4) = c_4(s_{0734} - s_{1625})$$

$$2S(5) = c_5d_{07} - c_1d_{16} + c_7d_{25} + c_3d_{34}$$

$$2S(6) = c_6d_{0734} - c_2d_{1625}$$

$$2S(7) = c_7d_{07} - c_5d_{16} + c_3d_{25} - c_1d_{34}$$

- 22 înmulțiri și 28 (6+6+16) adunări

# Problema

$$V = A \cdot U \cdot A^T \quad V_H = H \cdot U \cdot H$$

Fie blocul de imagine  $U$  de dimensiune  $4 \times 4$  pixeli de mai jos, care reprezintă o porțiune dintr-o imagine tip tablă de șah, în care fiecare pixel este reprezentat prin luminanță sa. Dorim să aplicăm o transformare unitară asupra blocului  $U$  astfel încât să se conserve, în urma transformării, în totalitate energia blocului, dar în același timp, să se compacteze într-un număr mult mai mic de coeficienți nenuli decât numărul coeficienților nenuli din blocul  $U$  reprezentat în domeniul spațial.

- Dacă matricile transformărilor pe care le avem la dispoziție sunt: matricea transformării Hadamard unitară ordonată,  $H_4[4 \times 4]$ ; matricea transformării cosinus discrete,  $C_4[4 \times 4]$ ; matricea transformării Fourier discrete unitare,  $F_4[4 \times 4]$ , pe care dintre ele ați alege-o pentru a asigura atât conservarea cât și compactarea energiei? Justificați răspunsul.
- Cum arată blocul de imagine  $V[4 \times 4]$  în domeniul transformat, folosind matricea transformării aleasă? (Se va realiza o transformare bidimensională a imaginii). Demonstrați conservarea energiei blocului în urma transformării și estimăți compactarea energiei blocului obținută prin transformare.

$$U = \begin{bmatrix} 0 & 0 & 100 & 100 \\ 100 & 100 & 0 & 0 \\ 0 & 0 & 100 & 100 \\ 100 & 100 & 0 & 0 \end{bmatrix}.$$

$$H_4 = \frac{1}{2} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}; C_4 = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.65 & 0.27 & -0.27 & -0.65 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.27 & -0.65 & 0.65 & -0.27 \end{bmatrix}; F_4 = \frac{1}{2} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$H^T = H$$

$$C^T \neq C$$

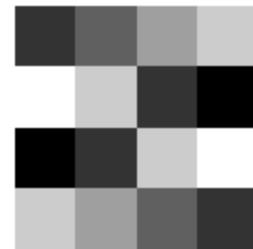
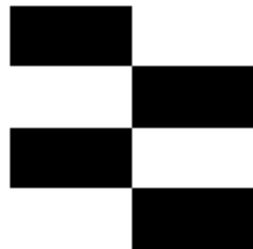
$$F^T = F$$

$$E_U = 8 \cdot 100^2 = 8000$$

$$E_V = 2 \cdot 200^2 = 8000$$

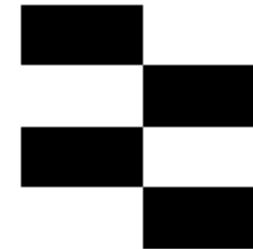
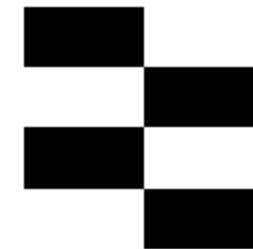
DCT

510.00	0.00	0.00	0.00
0.00	-180.31	0.00	74.69
0.00	0.00	0.00	0.00
0.00	-435.31	0.00	180.31



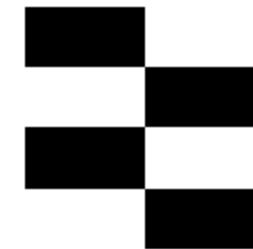
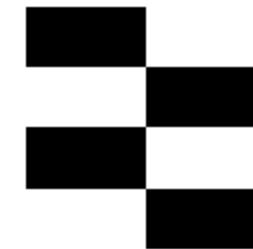
Hadamard

510.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	-510.00	0.00	0.00



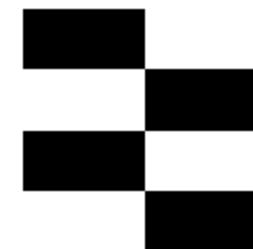
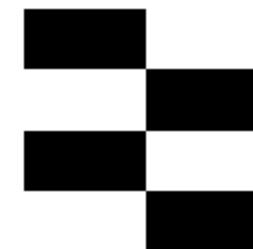
Fourier

510	0	0	0
0	0	0	0
0	-255 + 255i	0	-255 + 255i
0	0	0	0



Haar

510.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	-360.62	0.00	0.00
0.00	-360.62	0.00	0.00



# SACCDMM - Curs 06

## Standardul de compresie JPEG

Sl.Dr.Ing. Camelia FLOREA

# JPEG

- JPEG - acronim pentru grupul de experți fotografi care a elaborat standardul „Joint Photographic Experts Group”;
  - grup de experti provenind din mai multe organizații de standardizare
    - ISO -International Standards Organization
    - IEC -International Electrotechnical Commission
    - ITU -International Telecommunications Union

## JPEG clasic

ISO/ IEC 10928-1

ITU-T Recommendation T-81

draft standard: 1991

international standard: 1992

## JPEG 2000

ISO/ IEC 15444-1

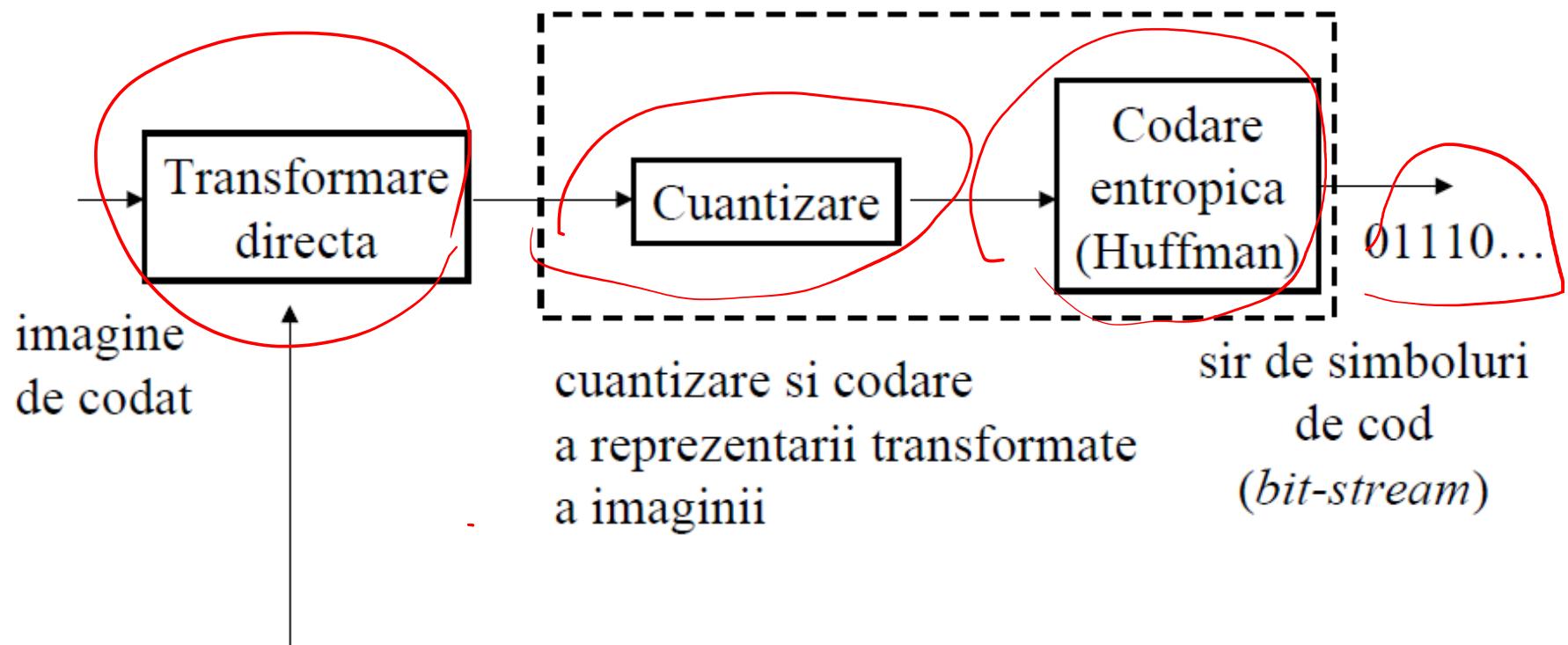
ITU-T Recommendation T-800

draft standard: 2000

international standard: Dec. 2000  
(partea 1)

# Schema bloc de codare

JPEG



prelucrari in suportul spatial al imaginii  
prelucrari in domeniul valorilor pixelilor  
sunt diferite la JPEG / JPEG2000

# JPEG clasic

- JPEG este un mecanism standardizat de **compresie a imaginilor**.
  - un algoritm pentru compresia imaginilor color (cu 24-biți pe pixel) sau pe nivele de gri **ce descriu scene din lumea reală** (de exemplu, fotografii).
- astfel, JPEG operează bine pe fotografii, pe opere de artă naturaliste, sau orice scene similare;
  - însă **rezintă pierderi** în cazul imaginilor de tip **inscripții, desene sau schițe** (datorită algoritmului de compresie).

# JPEG

- **Compresie „cu pierderi” bazata pe DCT** (Discrete Cosine Transform)
  - imaginea obtinuta prin decompresie NU este identica cu cea initiala,
  - insa, vizual diferențele sunt imperceptibile!
- JPEG este dedicat *compresiei imaginilor care vor fi vizualizate de oameni*
  - pierderile detaliilor imperceptibile vizual, sunt implementate in vederea maximizarii ratei de compresie!
- ⇒ JPEG exploateaza limitarile ochiului uman,
  - anume faptul ca schimbările mici de culoare sunt percepute cu o acuratețe mai mică decât schimbările mici de strălucire.
- Principalul câștig
  - compresii puternice, cu o rată de compresie de până la 100:1 (de obicei de 10:1 până la 20:1 fără degradare majoră).

# JPEG

- Obiectivele JPEG
  - să poată fi ușor de implementat în aplicații
  - să permită alegerea gradului de compresie și/sau calitatea
  - independent de tipul imaginii
  - grad de complexitate redus
  - permite codarea sevențială
  - permite codarea progresivă
  - opțiuni de codare ierarhica

# Introducere

- 1988 – utilizarea DCT ca transf.
  - 1988-1990 teste, simulari, evaluari
  - 1991 – propunere de standard
  - 1992 – standard international
- 
- **două metode de compresie**
    - metoda de compresie *cu pierderi* – DCT
    - metoda de compresie *fara pierderi*

# Metode de compresie JPEG

- Metoda de compresie **cu pierderi** bazată pe transformata cosinus discretă
  - este vorba de formatul **JPEG „clasic”**, care permite rate de compresie importante (de 10:1 până la 20:1) păstrând în același timp o foarte bună calitate a imaginii; această metodă de compresie este ireversibilă.
- Metoda de compresie **predictivă fără pierderi**
  - nu au loc pierderi de informație și este în consecință posibilă reproducerea exactă a imaginii originale, dar rata compresiei posibilă cu această metodă este mult mai modestă (aproximativ 2:1); această metodă de compresie este reversibilă.

# JPEG fara pierderi

- NU folosește algoritmii de compresie prin transformări – DCT
- Factor de compresie slab
- Algoritmul **JPEG fara pierderi**:
  - codare differentiala DPCM
  - codor Huffman sau arithmetic
- Predictorii:
  - eroarea de predicție:  $e = y - \underline{\underline{X}}$

		<i>c</i>	<i>b</i>	
	<i>a</i>		<i>X</i>	

$$y = 0$$

$$y = a$$

$$y = b$$

$$y = c$$

$$y = a + b - c$$

$$y = a + \frac{b - c}{2}$$

$$y = b + \frac{a - c}{2}$$

$$y = \frac{a + b}{2}$$

# Modul de operare compresie pe baza de DCT

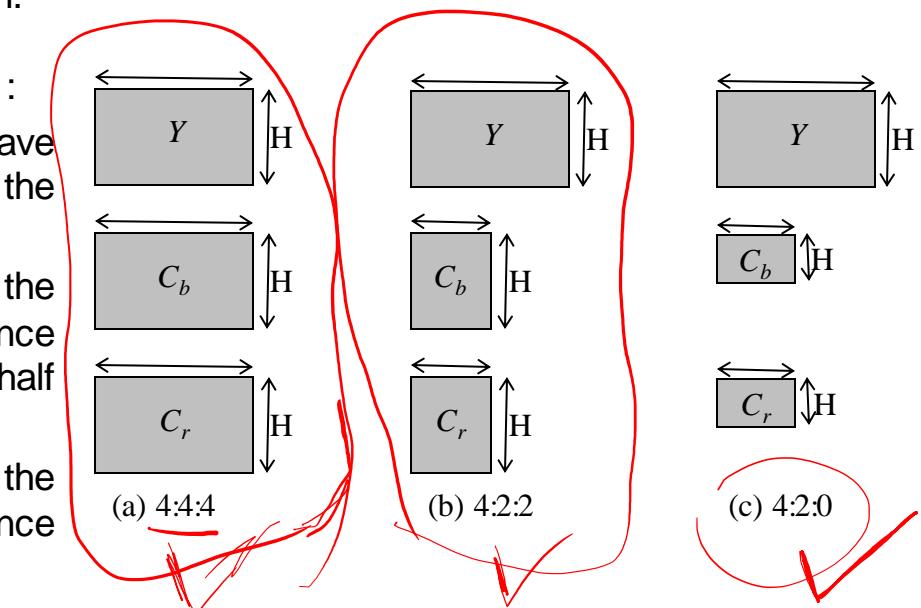
- Standardul JPEG specifică trei moduri de operare pentru compresia imaginilor:
  - **modul sequential bazat pe DCT**
    - modul de bază de operare JPEG - cel mai popular
    - suportă doar codare cu pierderi
  - **modul progresiv bazat pe DCT**
    - se realizează un set de subimagini, fiecare subimagine fiind codată cu un set specific de coeficienți DCT.
    - codorul DCT va trebui să aibă un buffer în care datele (coeficienții DCT ai subimaginilor) să fie memorate înainte de codarea entropică.
  - **modul ierarhic**
    - pentru aplicații – când e nevoie de imagini la rezoluții multiple
    - fiecare plan este codat ca o secvență de cadre (primul cadru – versiune subzesantionată a imaginii originale; cadrele următoare – cadre diferență)
    - se poate folosi: JPEG cu pierderi; JPEG fără pierderi; combinat (ultima etapă JPEG fără pierderi).

# Spațiul de culoare în JPEG

- Imaginile pot fi reprezentate pe mai multe planuri de culoare
  - cele mai multe dispozitive de scanare generează imagini pe planuri de culoare R,G,B.
  - standardul JPEG nu prezinta restrictii pentru tipul imaginii de intrare
    - uzual compresia se realizează pe un format care realizează o decorelare intre luminanta și crominanță: YUV, YCbCr, etc.
- Imaginea este interpretată ca o colecție de planuri de imagine/ componente
  - maxim 255 planuri de imagine (matrice de pixeli)
  - se comprimă fiecare componentă de culoare separat
- pixeli reprezentati:
  - 8 sau 12 biti/pixel – codarea cu DCT;
  - între 2 – 16 biti/pixel – codarea fără pierderi

# Subeșantionarea

- JPEG acceptă ca cele trei componente ale tripletului YUV/ YCbCr să fie **comprimeate folosind rate de compresie diferite**.
- Compresia JPEG este **mai eficientă** atunci culoarea este reprezintă sub forma decorrelată: **luminanță (strălucire) și crominanță**.
  - creșterea eficienței codării se explică prin faptul că ochiul este mai puțin sensibil la schimbările de culoare decât la schimbările de luminozitate  
=> canalele de crominanță pot fi **codate cu pierderi mai mari** decât canalul de luminanță.
- Utilizare în mod curent: câte o valoare U și V pentru fiecare 4 valori ale componentei Y (4:2:0)
  - **se va salva 50% din spațiul utilizat** (6 valori în loc de 12) fără a afecta calitatea imaginii din punctul de vedere a perceptiei ochiului uman.
- There are three color format in the baseline system :
  - a) **4:4:4** The chrominance components have identical vertical and horizontal resolution as the luminance component.
  - b) **4:2:2** The chrominance components have the same vertical resolution as the luminance component, but the horizontal resolution is half one.
  - c) **4:2:0** Both vertical and horizontal resolution of the chrominance component is half of the luminance component.



# Dimensiunea componentelor

- Componentele pot avea dimensiuni diferite:
  - $x_i, y_i$  dimensiunile pentru componenta  $i$
  - $x_j, y_j$  dimensiunile pentru componenta  $j$
  - factorii de esantionare relativa ( $H_i, V_i$ ) iau valori intre 1 - 4
- In format/ fișier se specifică
  - dimensiunile maxime ( $X, Y$ )
  - factorii de esantionare relativi maximi ( $H_{max}, V_{max}$ )
  - factorii de esantionare componente ( $H_i, V_i$ )
- Dacă avem  $X, H_{max}$ , și  $H_i$ 
  - putem determina  $x_i$  pentru fiecare plan
- Dacă avem  $Y, V_{max}$ , și  $V_i$ 
  - putem determina  $y_i$  pentru fiecare plan

$$\frac{x_i}{x_j} = \frac{H_i}{H_j}$$

$$\frac{y_i}{y_j} = \frac{V_i}{V_j}$$

$$X = \max(x_i);$$
$$Y = \max(y_i).$$

~~$$y = \left[ X \cdot \frac{H_i}{H_{max}} \right]$$~~
$$x_i = \left[ X \cdot \frac{H_i}{H_{max}} \right]$$

$$c_6 \rightarrow 1$$
$$c_7 \rightarrow 1$$

$$y_i = \left[ Y \cdot \frac{V_i}{V_{max}} \right]$$

# Intrețeserea planurilor de culoare

- moduri de prelucrare:
  - fara intretesere
    - fiecare plan de culoare/componentă este prelucrată separat
    - ex. RGB - > 3 planuri de culoare de 8 biti (R,G,B)
  - cu intretesere
    - definim un **element unitar**
      - pixel – compresia fara pierderi
      - bloc de 8x8 – compresia cu pierderi
    - unitatea de codare minima (MCU)
      - planul de culoare *i* este impartit in blocuri de dim.  $H_i \times V_i$
      - selectarea subblocurilor corespondente din fiecare plan
      - cea mai mica grupare de date intretesute

DCT



# Exemplu de intretesere

- Spatiu de culoare YUV sau YCbCr
  - ~~Y – 2x2 blocuri de cate 8x8 pixeli;~~
  - ~~C<sub>b</sub>, C<sub>r</sub> – câte un bloc elementar de 8x8 pixeli~~
- **MCU** – **4** blocuri de Y; **1** bloc de C<sub>b</sub>; **1** bloc de C<sub>r</sub>

	0	1	2	3	4
0	● ●	● ●			
1	● ●	● ●			
2	● ●	● ●			
3	● ●	● ●			
4					

Planul Y

	0	1	2	3
0	●	●	●	●
1	●	●	●	●
2	●	●	●	●
3	●	●	●	●

Planul C<sub>b</sub>

	0	1	2	3	4
0	●	●	●	●	
1	●	●	●	●	
2	●	●	●	●	
3	●	●	●	●	
4					

Planul C<sub>r</sub>

$$\text{MCU}_1 = Y_{00}Y_{01}Y_{10}Y_{11}C_{b00}C_{r00}; \quad \text{MCU}_2 = Y_{02}Y_{03}Y_{12}Y_{13}C_{b01}C_{r01}$$

- numarul maxim de planuri intretesute este 4
- numarul maxim de elemente dintr-un MCU este 10
  - daca nu se satisface aceasta conditie => codare ne-intretesuta

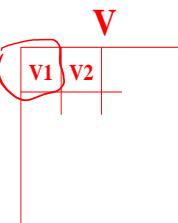
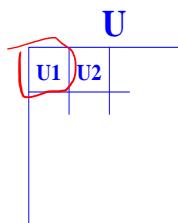
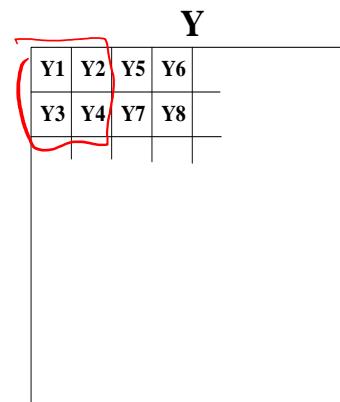
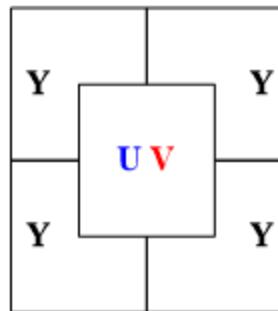
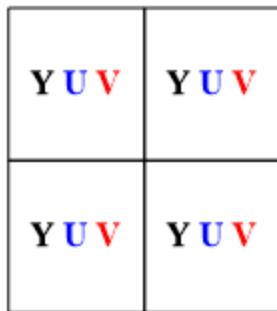
# Modul secvențial bazat pe DCT

- Presupune transformarea unui bloc de imagine de dimensiune  $N \times N$  din **domeniul spațial** în **domeniul coeficienților DCT** (spațiul frecvențelor) și aplicarea a catorva etape de prelucrare pentru compresie.
  - În general, valoarea lui  $N$  este egală cu 8.
- Alegerea dimensiunii de  $8 \times 8$  pixeli pentru blocul de bază de este motivată de mai multe considerente:
  - **implementare hardware și software mai ușoară** datorită necesarului relativ scăzut de memorie
    - complexitatea de calcul la un bloc de  $8 \times 8$  pixeli este relativ redusă
    - din punct de vedere al eficienței, alegerea unui bloc mai mare de  $8 \times 8$  pixeli **nu oferă un grad de compresie semnificativ mai mare**

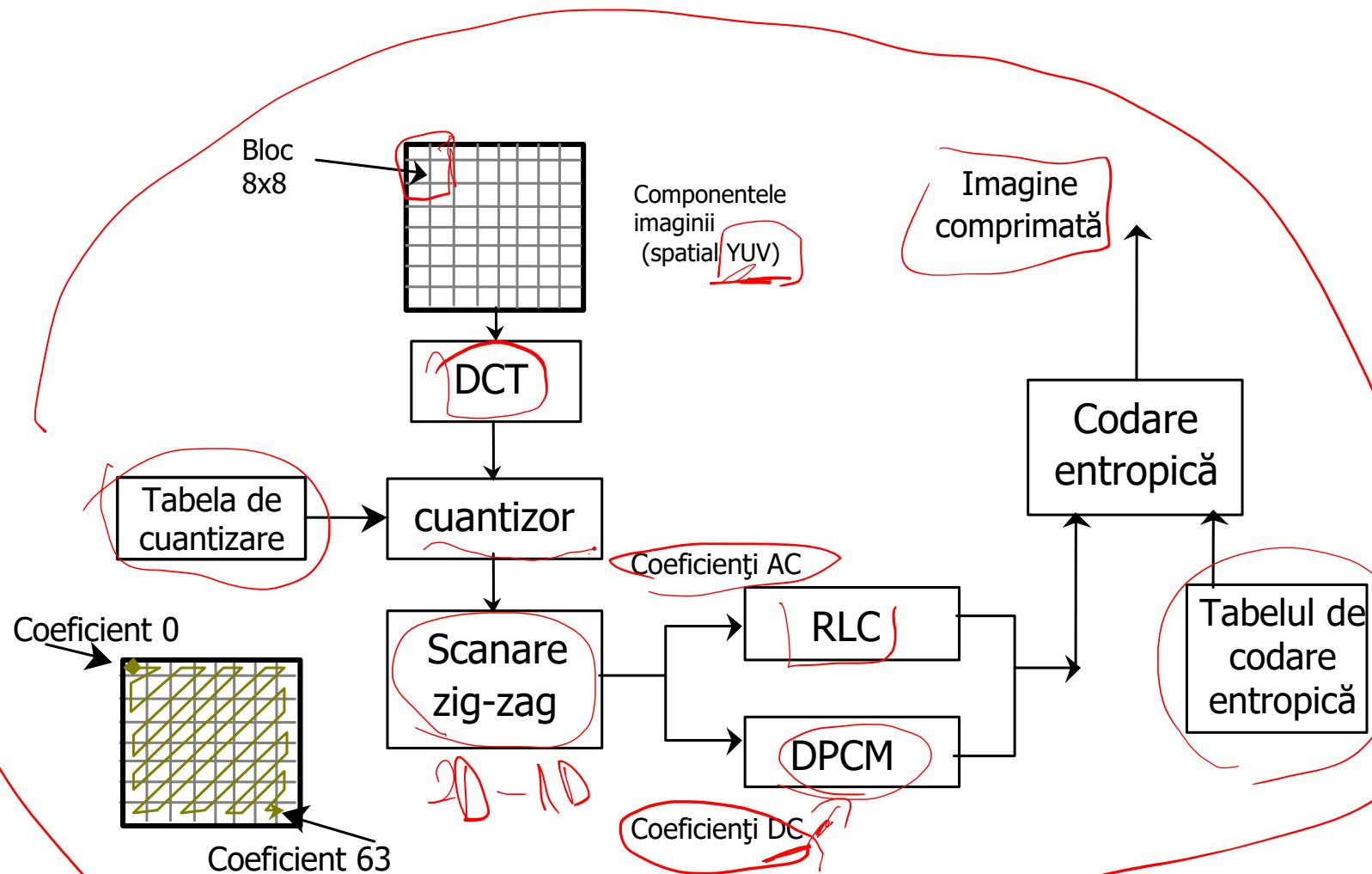
# Descompunerea în blocuri

- Procedura de compresie se aplică pe rând, pe fiecare bloc de  $8 \times 8$  pixeli din imagine – astfel, cele trei componente Y, U și V sunt descompuse în blocuri de această dimensiune.
- Datorită reducerii rezoluției componentelor de crominanță U și V prin subeșantionare rezultă că la 4 sub-blocuri de  $2 \times 2$  pixeli ale componentei Y le corespunde câte un singur bloc  $2 \times 2$  valori de ale componentelor U, respectiv V.
- Blocurile de imagine din cele trei componente sunt stocate întrețesut.
- Ordinea stocării acestora va fi:

Y1, Y2, Y3, Y4, U1, V1, Y5, Y6, Y7, Y8, U2, V2, ...



# Codorul JPEG



# Procesare bloc spre compresie

- Fiecare bloc de  $8 \times 8$  pixeli este prelucrat (comprimat):
  - axarea componentelor pe zero
  - aplicarea transformatiei cosinus discrete,
  - cuantizarea matricii transformate,
  - calculul coeficientului diferență DC,
  - aranjarea coeficientilor DCT în zigzag,
  - codare RLE, reprezentarea sirului RLE în format binar,
  - codare entropică (codarea coeficientului DC și codarea coeficientilor AC).

# Axarea componentelor pe zero

- Valorile originale ale eșantioanelor semnalelor Y respectiv U și V sunt cuprinse în domeniul  $[0, 2^b - 1]$ , unde b reprezintă numărul de biți/eșantion.
- Aceste valori sunt deplasate în domeniul  $[-2^{b-1}, 2^{b-1} - 1]$ , **centrate față de zero** pentru a putea realiza o precizie de calcul mai mare la aplicare DCT.
- În cazul compresiei JPEG, pentru fiecare componentă în parte, vom avea b=8 astfel încât valorile intensităților din imaginea originală cuprinse în intervalul  $[0, 255]$  sunt deplasate în intervalul  **$[-128, 127]$** .

# Aplicarea transformatei cosinus discrete (DCT)

- prin intermediul acestei transformări, un vector conținând date despre intensitate este transformat într-un vector care conține date despre modul de variație spațial al intensității  
    ⇒ matricea coeficienților DCT
- Elementele matricii coeficienților DCT au valorile mari concentrate în colțul din stânga-sus, iar în colțul din dreapta jos valorile sunt foarte mici - aproape nule.
  - DCT realizează o transformare a datelor în domeniul frecvență – astfel:
    - coeficienții din colțul din stânga-sus corespund frecvențelor joase - variații lente de intensitate între pixeli;
    - pe măsură ce se avansează către colțul din dreapta-jos, coeficienții corespund frecvențelor înalte - variații rapide de intensitate date de detaliile fine din imagine.
- În general într-o imagine reală frecvențele înalte au o pondere mai redusă decât cele joase, ceea ce explică valorile obținute în urma transformării.

# Cuantizarea coeficienților DCT ai blocurilor

- Operația de **cuantizare** este singura în care **se pierde informație** → efecte de blocking
- Rol cuantizare – a se obține cât mai multe valori nule sau mici → avantajul unei compresii eficiente realizate ulterior
- Aplicare pe DCT - oferă posibilitatea de a realiza această operație astfel incat pierderile de informație să fie cât mai redus
- Există o matrice de cuantizare pentru fiecare plan de culoare
- Trebuie realizat un echilibru rata de compresie – calitate (tehnici de proiectare psihovizuale)
  - controlul ratei de bit - mai mulți biți pentru coeficienții cu varianță mare
- Dacă valorile din tabelele standard de cuantizare sunt divizate cu 2, imaginea reconstruită este de obicei identică vizual cu imaginea sursă

$$U_{dct\_Q}(i,j) = [U_{dct}(i,j) / Q(i,j)], \text{ unde } i, j = \{0 \dots 7\}$$

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

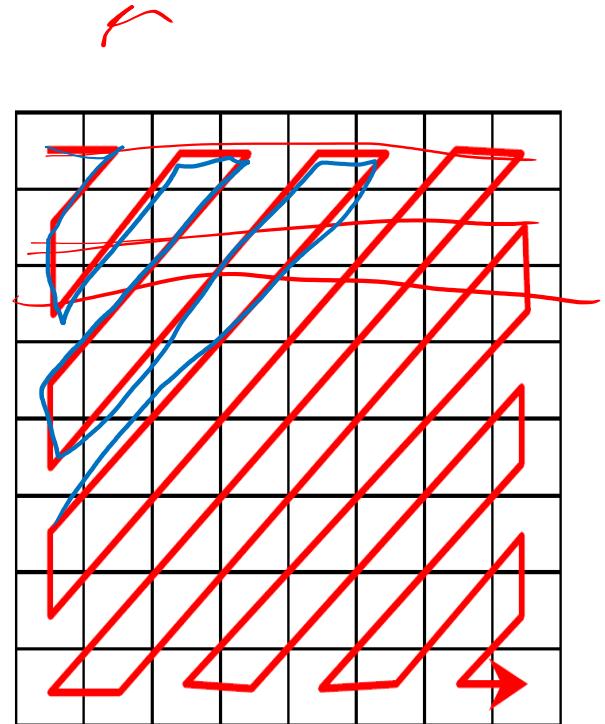
Tabelele de cuantizare Y luminanță

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

Tabelele de cuantizare crominanță

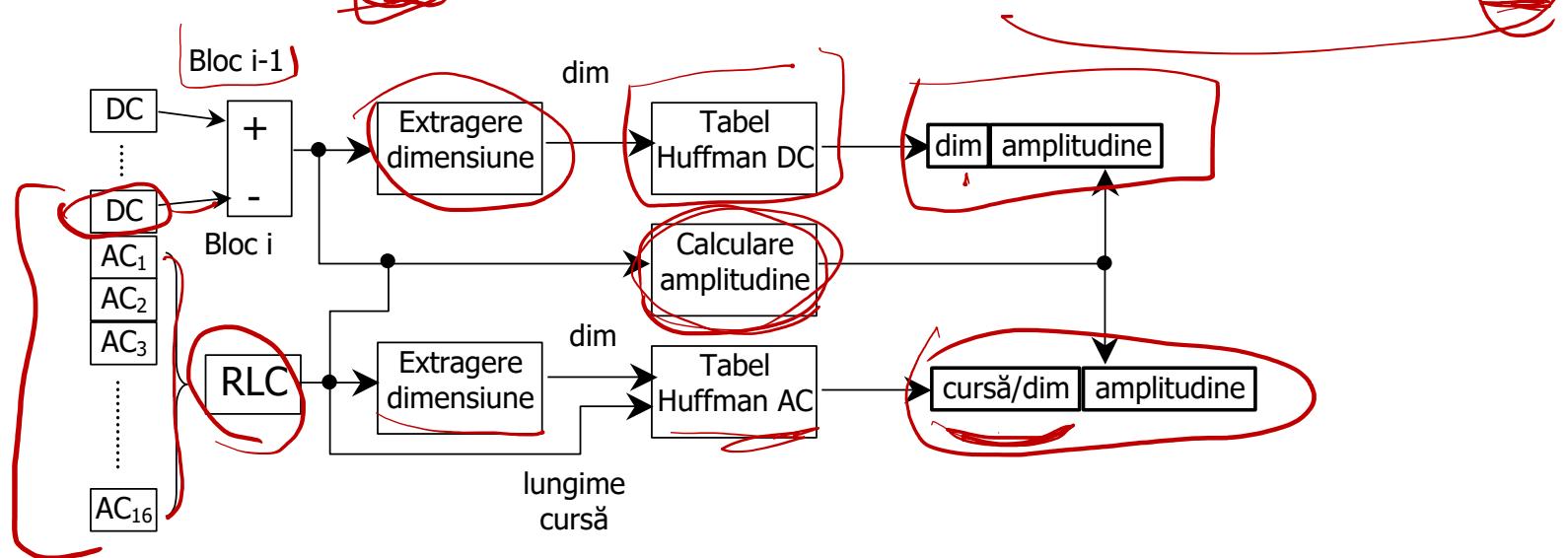
# Aranjarea blocului în zigzag

- Ordinea în care este parcursă matricea în vederea codării coeficienților de tip AC se alege în aşa fel încât să se profite cât mai bine de distribuția valorilor coeficienților.
- Se urmărește gruparea valorilor nule în şiruri cât mai lungi deoarece acest fapt permite o codare cât mai eficientă (compresie maximă).
- Deoarece valorile nenule sunt concentrate într-un colț al matricii, o parcurgere normală de tipul linie cu linie nu este eficientă, de aceea se preferă o parcurgere în zigzag.



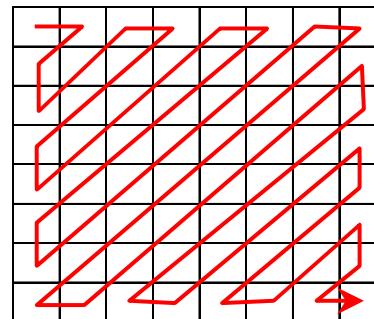
# Codarea entropică

- variante Huffman, aritmetică +
- DPCM (DC), RLC (AC)
- Huffman - se pot folosi specificațiile standardului
- limitări:
  - cuvinte de cod < 16 biți
  - cuvinte de cod FF trebuie urmate de un cuvant de cod 00



# Codarea Huffman

- coeficientii DC – (dim, amplitudine) 16 categorii
- Coeficientii AC [-1023, 1023]
  - 10 categorii de amplitudine
  - scanarea zig-zag
  - cuvantul de cod – (lungime\_cursa/dim, amp)
    - Lungime cursa > 15 codul (15/0) – partajeaza de cate ori este nevoie
    - Daca dupa un AC nenul toti sunt nuli – EOB (0/0)



# Calcul coeficient diferență DC (DPCM – Differential Pulse Code Modulation)

- Coeficientul diferență DC este obținut prin scăderea coeficientului DC al blocului anterior procesat din **coeficientul DC curent**.
- Această operație este echivalentă cu o codare diferențială a coeficientilor DC și se face pentru a se obține coeficienți de codat cât mai mici posibil, fiind aplicabilă tuturor coeficientilor DC cu excepția primului coeficient DC din imagine.
- Codul unei valori se obține prin concatenarea unui cod pentru clasa din care face parte și un cod corespunzător valorii acesteia.
- Exemplu - cod de clasă 5 (SSSS=0101) citind în tabelă se găsește codul de 3 biți 110.
  - după codul de clasă trebuie adăugată valoarea coeficientului diferență DC - cei mai puțin semnificativi SSSS biți din valoarea în cazul valorilor pozitive sau din valoarea în complement fata de 2 minus 1 dacă aceasta este negativă.

Clasa (SSSS) - 4 biți	Valoare diferență
0 (0000)	0
1 (0001)	-1, 1
2 (0010)	-3, -2, 2, 3
3 (0011)	-7,...,-4, 4,...7
4 (0100)	-15,...,-8, 8,...15
5 (0101)	-31,...,-16, 16,...,31
6 (0110)	-63,...,-32, 32,...,63
7 (0111)	-127,...,-64, 64,...,127
8 (1000)	-255,...,-128, 128,...,255
9 (1001)	-511,...,-256, 256,...,511
10 (1010)	-1023,...,-512, 512,...,1023
11 (1011)	-2047,...,-1024, 1024,...,2047

# Codare coeficient diferență DC

- Codare valoare 21 ce aparține componentei Y
  - clasa 5 - pentru care codul este 110.
  - valoare pozitivă – se adaugă cei mai puțin semnificativi 5 biți ai acesteia: 10101  
⇒ reprezentare binara 11010101 cod complet (cod clasa + cod valoare)
- Codare valoare -21 ce aparține componentei Y
  - aceeași clasa clasa, clasa 5 codul este 110
  - valoare negativă - se reprezintă în complement față de 2 minus 1 (21 complement față de 2 va fi 111111111101011 se scade 1 și se păstrează cei mai puțin semnificativi 5 biți) - adică 01010.
  - Codul complet în acest caz: 11001010.

Tabel 1. Tabel pt diferența coeficienților DC a luminantei

Category	Code length	Code word
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	<u>110</u>
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

Tabel 2. Tabel pt diferența coeficienților DC a crominanței

Category	Code length	Code word
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

# Codare RLE (Run Length Encoded)

- Dupa scanare in zigzag coeficienții DCT cuantizați - unui vector cu multe zerouri
  - coeficientul diferența DC nu se codeaza RLC
  - coeficienții AC cuantizati se codeaza RLE considerand numărul de zerouri consecutive
- **Vectorul RLE** contine perechi - ***numărul de zerouri*** care preced valoarea coeficientului AC nenul, urmată de valoarea ***coeficientului AC nenul***
- Fiecare coeficient - număr întreg în domeniul -1023 și +1023 (valori obținute analizând cazul cel mai defavorabil în modul de operare DCT).
  - în cazul codării "normale" reprezentare pe 11 biți (primul bit pentru semn, următorii 10 biți - valoarea)
    - duce practic la o reprezentare pe mai mulți biți decât în cazul imaginii necomprimeate (se folosesc 11 biți pentru un coeficient în loc de 8 biți pentru un pixel).
- Coeficienții din vectorul **RLE** sunt reprezentați prin **șiruri binare de lungime variabilă**
  - reprezentare sub forma: ***lungime cursă/identificator clasa, valoare coeficient***
    - Lungime cursă - numărul de zerouri care preced valoarea coeficientului AC nenul
    - Identificator clasa ne arată lungimea în biți a șirului binar care dă valoarea coeficientului
    - Valoare coeficientului nenul
      - Valorile pozitive - încep cu un 1 codate în maniera binară obișnuită.
      - Numerele negative - încep cu un 0, fiind reprezentate în complement față de 2 minus 1.

# Codare coeficienti AC

(lungime cură, ValCoefAC)

- **Identifier clasa** (SSSS) - clasa valorii coeficientului diferit de zero obținută similar ca în cazul coeficienților DC
- **Lungime cursa**(RRRR) - numărul de zerouri - valoare între 0 și 15 care poate fi reprezentată pe 4 biți
- Tabele Huffman continand coduri pentru **lungime cursa/identifier clasa**
  - pentru componenta de luminanta (Tabel 3)
  - pentru componentelete de crominata (Tabel 4)
- Poziția codului în tabela
  - $INDEX=11xRRRR+SSSS$ .
- Se rezerva codurile
  - cu RRRR=0 și SSSS=0 pentru simbolul special EOB
  - cu RRRR=15 și SSSS=0 pentru simbolul special ZRL.
- **Exemplu**, codificare (1,-4) din componente Y
  - SSSS = 3 și RRRR = 1, valoarea aparține componentei Y - se va utiliza Tabel 3 din anexe => deci codul clasei este 1111001 reprezentat pe 7 biți.
  - Valoare coefficient nul este -4 (111111111111100 in complement fata de 2), valoarea minus 1 este 11111111111011, deci se vor adăuga cei mai puțin semnificativi 3 biți adică 011.
  - Codul complet al simbolului (1,-4) este 1111001011.

Clasa (SSSS) - 4 biți	Valoare
1 (0001)	-1, 1
2 (0010)	-3, -2, 2, 3
3 (0011)	-7,...,-4, 4,...7
4 (0100)	-15,...,-8, 8,...,15
5 (0101)	-31,...,-16, 16,...,31
6 (0110)	-63,...,-32, 32,...,63
7 (0111)	-127,...,-64, 64,...,127
8 (1000)	-255,...,-128, 128,...,255
9 (1001)	-511,...,-256, 256,...,511
10 (1010)	-1023,...,-512, 512,...,1023

Tabel 3. Tabel pt coeficienții AC a luminanței (part 1/4)

R / S (Run/Size)	Lungime cod	Cuvant cod
0/0 (EOB)	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	111111110000010
0/A	16	111111110000011
1/1	4	1100
1/2	5	11011
1/3	7	1111001
1/4	9	111110110
1/5	11	1111110110
1/6	16	111111110000100
1/7	16	111111110000101
1/8	16	111111110000110
1/9	16	111111110000111
1/A	16	111111110001000
2/1	5	11100
2/2	8	11111001
2/3	10	1111110111
2/4	12	111111110100
2/5	16	111111110001001
2/6	16	111111110001010
2/7	16	111111110001011
2/8	16	111111110001100
2/9	16	111111110001101
2/A	16	111111110001110
3/1	6	111010
3/2	9	111110111
3/3	12	111111110101
3/4	16	111111110001111
3/5	16	111111110010000
3/6	16	111111110010001
3/7	16	111111110010010
3/8	16	111111110010011
3/9	16	111111110010100
3/A	16	111111110010101

Tabel 3. (part 2/4)

R / S (Run/Size)	Lungime cod	Cuvant cod
4/1	6	111011
4/2	10	1111111000
4/3	16	111111110010110
4/4	16	1111111110010111
4/5	16	1111111110011000
4/6	16	1111111110011001
4/7	16	1111111110011010
4/8	16	1111111110011011
4/9	16	1111111110011100
4/A	16	1111111110011101
5/1	7	1111010
5/2	11	11111110111
5/3	16	111111110011110
5/4	16	1111111110011111
5/5	16	1111111110100000
5/6	16	1111111110100001
5/7	16	1111111110100010
5/8	16	1111111110100011
5/9	16	1111111110100100
5/A	16	1111111110100101
6/1	7	1111011
6/2	12	111111110110
6/3	16	1111111110100110
6/4	16	11111111110100111
6/5	16	11111111110101000
6/6	16	11111111110101001
6/7	16	11111111110101010
6/8	16	11111111110101011
6/9	16	11111111110101100
6/A	16	11111111110101101
7/1	8	11111010
7/2	12	111111110111
7/3	16	1111111110101110
7/4	16	11111111110101111
7/5	16	11111111110110000
7/6	16	11111111110110001
7/7	16	11111111110110010
7/8	16	11111111110110011
7/9	16	11111111110110100
7/A	16	11111111110110101
8/1	9	111111000
8/2	15	111111111000000

Tabel 3. (part 3/4)

R / S (Run/Size)	Lungime cod	Cuvant cod
8/3	16	11111111110110110
8/4	16	11111111110110111
8/5	16	11111111111011100
8/6	16	11111111111011101
8/7	16	111111111110111010
8/8	16	111111111110111011
8/9	16	111111111110111100
8/A	16	111111111110111101
9/1	9	111111001
9/2	16	11111111110111110
9/3	16	111111111110111111
9/4	16	111111111111000000
9/5	16	111111111111000001
9/6	16	111111111111000010
9/7	16	111111111111000011
9/8	16	111111111111000100
9/9	16	111111111111000101
9/A	16	111111111111000110
A/1	9	111111010
A/2	16	111111111111000111
A/3	16	111111111111001000
A/4	16	111111111111001001
A/5	16	111111111111001010
A/6	16	111111111111001011
A/7	16	111111111111001100
A/8	16	111111111111001101
A/9	16	111111111111001110
A/A	16	111111111111001111
B/1	10	1111111001
B/2	16	111111111111010000
B/3	16	111111111111010001
B/4	16	111111111111010010
B/5	16	111111111111010011
B/6	16	111111111111010100
B/7	16	111111111111010101
B/8	16	111111111111010110
B/9	16	111111111111010111
B/A	16	111111111111011000
C/1	10	1111111010
C/2	16	111111111111011001
C/3	16	111111111111011010
C/4	16	111111111111011011

Tabel 3. (part 4/4)

R / S (Run/Size)	Lungime cod	Cuvant cod
C/5	16	111111111111011100
C/6	16	111111111111011101
C/7	16	111111111111011110
C/8	16	111111111111011111
C/9	16	111111111111100000
C/A	16	111111111111100001
D/1	11	111111111111110000
D/2	16	111111111111111000
D/3	16	11111111111111110001
D/4	16	11111111111111111000
D/5	16	11111111111111111001
D/6	16	11111111111111111010
D/7	16	11111111111111111011
D/8	16	11111111111111111100
D/9	16	11111111111111111110
D/A	16	11111111111111111111
E/1	16	111111111111111111110
E/2	16	1111111111111111111101
E/3	16	11111111111111111111011
E/4	16	1111111111111111111110
E/5	16	1111111111111111111111
E/6	16	11111111111111111111110
E/7	16	111111111111111111111100
E/8	16	1111111111111111111111001
E/9	16	1111111111111111111111101
E/A	16	1111111111111111111111110
F/0 (ZRL)	11	11111111111111001
F/1	16	11111111111111110101
F/2	16	11111111111111111111010
F/3	16	1111111111111111111111111
F/4	16	1111111111111111111111110
F/5	16	11111111111111111111111100
F/6	16	11111111111111111111111101
F/7	16	111111111111111111111111011
F/8	16	11111111111111111111111110
F/9	16	111111111111111111111111101
F/A	16	111111111111111111111111110

Tabel 4. Tabel pt coeficientii AC a crominantei (part 1/4)

R / S (Run/Size)	Lungime cod	Cuvant cod
0/0 (EOB)	2	00
0/1	2	01
0/2	3	100
0/3	4	1010
0/4	5	11000
0/5	5	11001
0/6	6	111000
0/7	7	1111000
0/8	9	111110100
0/9	10	1111110110
0/A	12	11111110100
1/1	4	1011
1/2	6	111001
1/3	8	11110110
1/4	9	111110101
1/5	11	1111110110
1/6	12	11111110101
1/7	16	111111110001000
1/8	16	111111110001001
1/9	16	111111110001010
1/A	16	111111110001011
2/1	5	11010
2/2	8	11110111
2/3	10	1111110111
2/4	12	11111110110
2/5	15	11111111000010
2/6	16	111111110001100
2/7	16	111111110001101
2/8	16	111111110001110
2/9	16	111111110001111
2/A	16	1111111110010000
3/1	5	11011
3/2	8	11111000
3/3	10	111111000
3/4	12	11111110111
3/5	16	111111110010001
3/6	16	111111110010010
3/7	16	111111110010011
3/8	16	111111110010100
3/9	16	111111110010101
3/A	16	111111110010110
4/1	6	111010

Tabel 4. (part 2/4)

R / S (Run/Size)	Lungime cod	Cuvant cod
4/2	9	1111110110
4/3	16	1111111110010111
4/4	16	1111111110011000
4/5	16	1111111110011001
4/6	16	1111111110011010
4/7	16	1111111110011011
4/8	16	1111111110011100
4/9	16	1111111110011101
4/A	16	1111111110011110
5/1	6	111011
5/2	10	11111111001
5/3	16	1111111110011111
5/4	16	1111111110100000
5/5	16	1111111110100001
5/6	16	1111111110100010
5/7	16	1111111110100011
5/8	16	1111111110100100
5/9	16	1111111110100101
5/A	16	1111111110100110
6/1	7	1111001
6/2	11	11111110111
6/3	16	1111111110100111
6/4	16	1111111110101000
6/5	16	1111111110101001
6/6	16	1111111110101010
6/7	16	1111111110101011
6/8	16	1111111110101100
6/9	16	1111111110101101
6/A	16	1111111110101110
7/1	7	1111010
7/2	11	11111111000
7/3	16	1111111110101111
7/4	16	1111111110101000
7/5	16	1111111110101001
7/6	16	1111111110101000
7/7	16	1111111110101001
7/8	16	1111111110101000
7/9	16	1111111110101011
7/A	16	1111111110101010
8/1	8	11111001
8/2	16	1111111110101111
8/3	16	1111111110101100

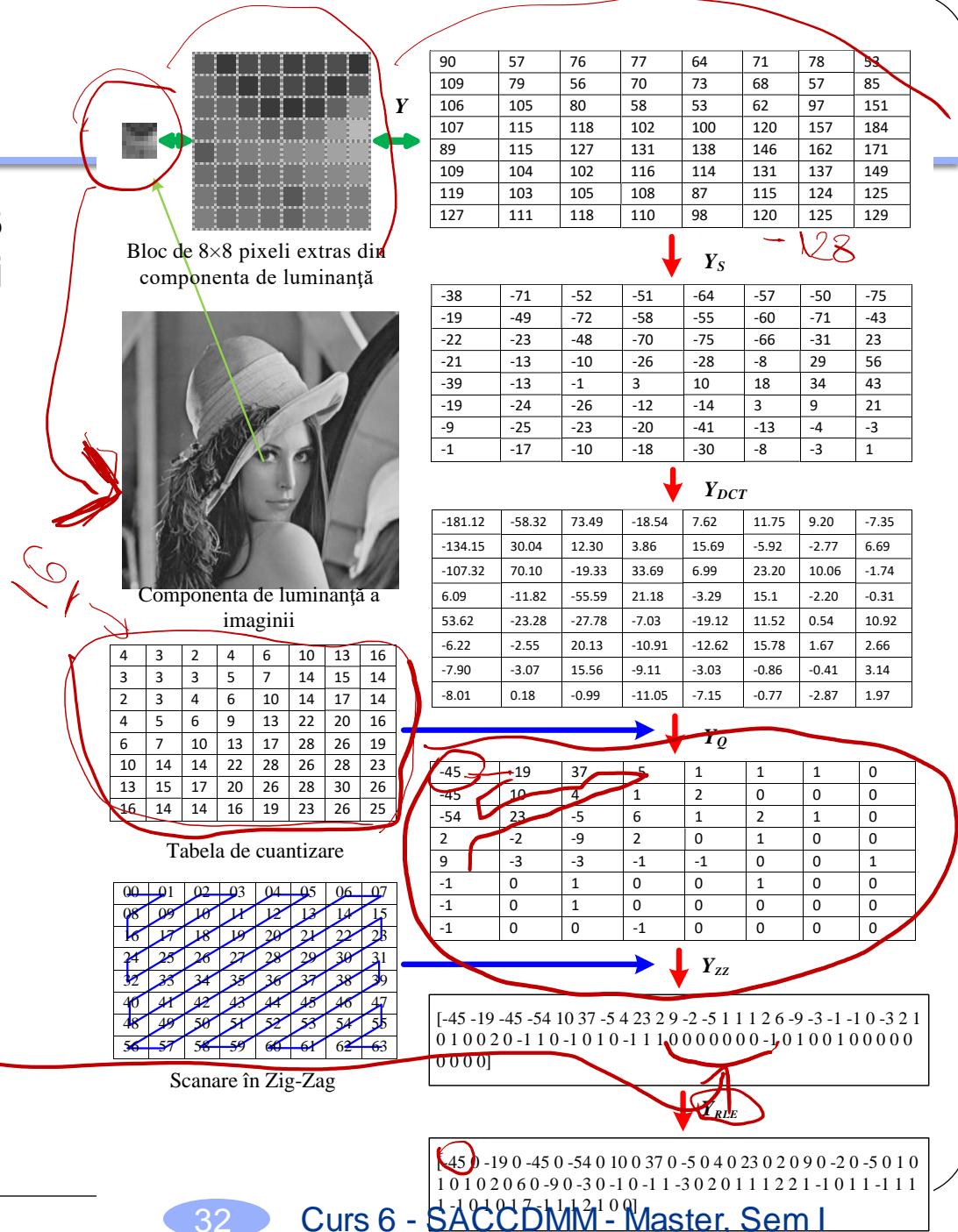
Tabel 4. (part 3/4)

R / S (Run/Size)	Lungime cod	Cuvant cod
8/4	16	1111111110111001
8/5	16	1111111110111010
8/6	16	1111111110111011
8/7	16	1111111110111100
8/8	16	1111111110111101
8/9	16	1111111110111110
8/A	16	1111111110111111
9/1	9	111110111
9/2	16	1111111111000000
9/3	16	1111111111000001
9/4	16	1111111111000010
9/5	16	1111111111000011
9/6	16	1111111111000100
9/7	16	1111111111000101
9/8	16	1111111111000110
9/9	16	1111111111000111
9/A	16	1111111111001000
A/1	9	111111000
A/2	16	1111111111001001
A/3	16	1111111111001010
A/4	16	1111111111001011
A/5	16	1111111111001100
A/6	16	1111111111001101
A/7	16	1111111111001110
A/8	16	1111111111001111
A/9	16	1111111111010000
A/A	16	1111111111010001
B/1	9	111111001
B/2	16	1111111111010010
B/3	16	1111111111010011
B/4	16	1111111111010100
B/5	16	1111111111010101
B/6	16	1111111111010110
B/7	16	1111111111010111
B/8	16	1111111111010100
B/9	16	1111111111010101
B/A	16	1111111111010100
C/1	9	111111010
C/2	16	1111111111010101
C/3	16	1111111111010110
C/4	16	1111111111010111
C/5	16	1111111111010110

Tabel 4. (part 4/4)

R / S (Run/Size)	Lungime cod	Cuvant cod
C/6	16	1111111111011111
C/7	16	1111111111100000
C/8	16	1111111111100001
C/9	16	1111111111100010
C/A	16	1111111111100011
D/1	11	111111111001
D/2	16	1111111111100100
D/3	16	11111111111100101
D/4	16	11111111111110010
D/5	16	11111111111111001
D/6	16	11111111111111100
D/7	16	11111111111111101
D/8	16	111111111111111010
D/9	16	111111111111111011
D/A	16	1111111111111110100
E/1	14	1111111111100000
E/2	16	111111111111101101
E/3	16	111111111111110110
E/4	16	1111111111111110111
E/5	16	1111111111111110000
E/6	16	1111111111111110001
E/7	16	1111111111111110010
E/8	16	1111111111111110011
E/9	16	1111111111111110100
E/A	16	1111111111111110101
F/0 (ZRL)	10	1111111010
F/1	15	1111111111000011
F/2	16	1111111111110110
F/3	16	1111111111111011
F/4	16	1111111111111100
F/5	16	111111111111111001
F/6	16	111111111111111010
F/7	16	111111111111111011
F/8	16	111111111111111100
F/9	16	1111111111111111101
F/A	16	1111111111111111110

- Extragerea unui bloc de  $8 \times 8$  pixeli din componenta Y și aplicarea pașilor pentru compresie



# Decodorul JPEG

- 1) Scanarea fișierului JPEG, decodarea antetului/ delimitatori/ flux de date.
  - 2) Decodarea Huffman a coeficientilor (se folosesc tabelele din segmentul DHT din antet);
  - 3) Reordonarea zigzag a elementelor din fiecare bloc de  $8 \times 8$ ;
  - 4) Decuantizarea elementelor din fiecare bloc de  $8 \times 8$  (se folosesc tabelele din segmentul DQT);
  - 5) Transformarea inversă a DCT (IDCT) pentru fiecare bloc de  $8 \times 8$
  - 6) Reșantionarea. La acest punct o imagine de pixeli de dimensiunea originală a fost reconstruită.
  - 7) Refacerea datelor în format RGB din YUV. Acest pas nu se aplică pentru imaginile pe nivale de gri.
- 
- Pașii 2 – 7 trebuie repetați pentru toate blocurile de  $8 \times 8$ , până când toate blocurile sunt prelucrate.

# Fișierele JPEG

- JPEG a dezvoltat un **format de fișier** care descrie modul de organizare a conținutului/șirului de biți codat – pentru a avea atât **parametrii de codare**, cât și **imaginea**
  - Astfel, un fișier JPEG este împărțit în mai multe segmente
    - cu ajutorul unor cuvinte de cod - delimitatori ( fiecare delimitator este precedat de 1 octet 0xFF).
    - aceste segmente sunt utilizate pentru păstrarea informațiilor
      - generale despre imagine,
      - despre modul de compresie a imaginii (rata de compresie, tabele de cuantizare, tabele Huffman, etc.),
      - precum și datele ce descriu conținutul imaginii.
- acestea permit implementarea JPEG pe diferite platforme, iar decompresia poate fi realizată fără alte probleme.
- au de obicei extensia „*jpeg*” sau „*jpg*” dar se mai utilizează ocazional și extensia „*jif*” (JPEG File Interchange Format).

# Delimitatori de segmente – JPEG

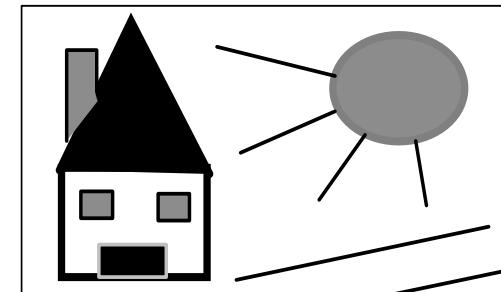
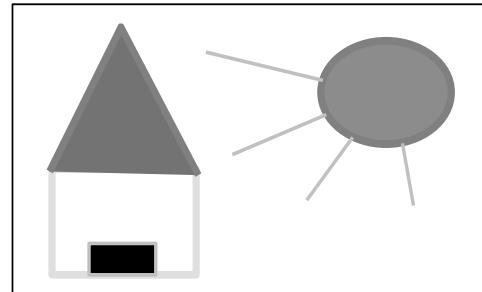
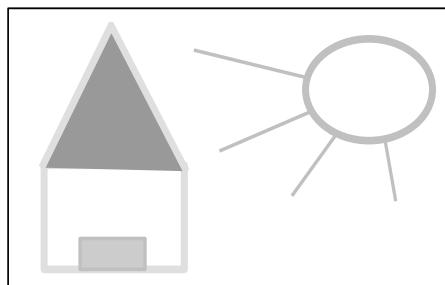
Nume delimitator	Identifier delimitator	Descriere
SOI	0xD8	Start imagine
APP0	0xE0	Segmentul aplicației JFIF
APPn	0xE1 – 0xEF	Alte segmente APP
COM	0xFE	Comentariu text
DQT	0xDB	Tabelul de cuantizare
SOF0	0xC0	Start cadru
DHT	0xC4	Tabelul Huffman
SOS	0xDA	Start scanare
EOI	0xD9	Sfârșitul imaginii

## Segmente principale:

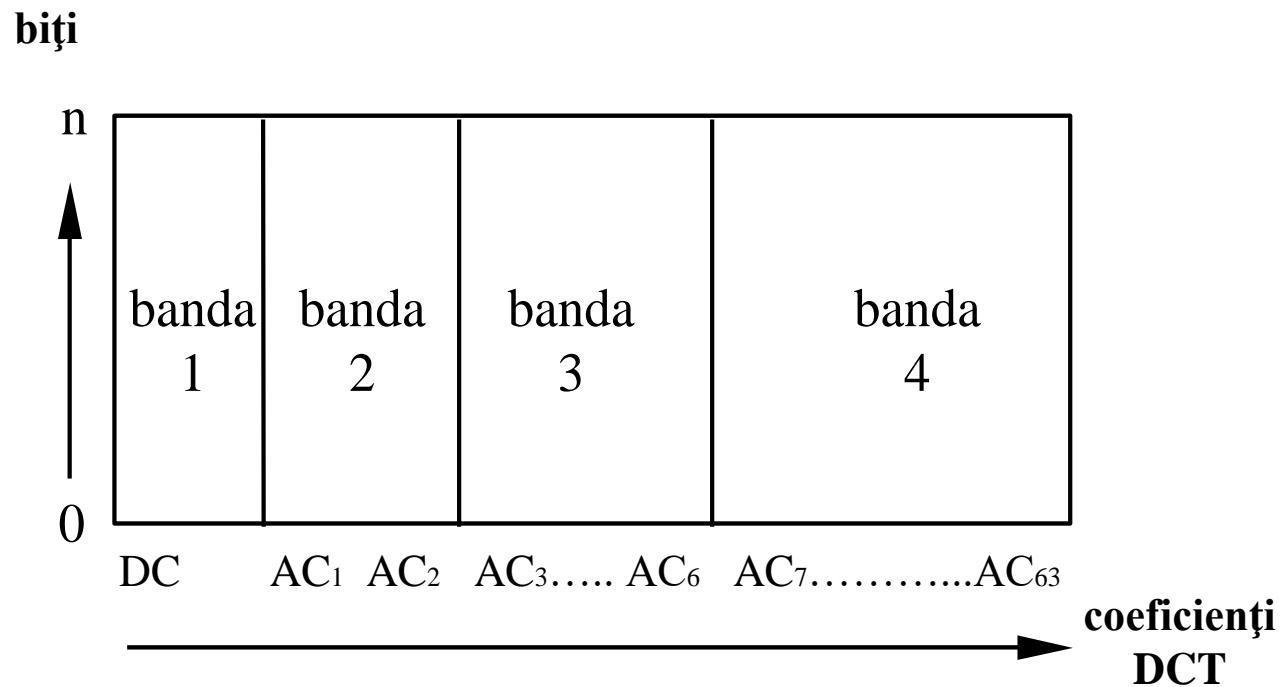
- Delimitatorul **SOI** - apare la începutul imaginii
- **APP0** - conține date pentru identificarea fișierului
- **APPn** (unde *n* poate lua valori de la 1 la 15) - conțin parametrii de achiziție ai imaginii (metadata) – acest segment este optional.
- **DQT** - unul sau mai multe tabele de cuantizare
- **SOF0** - pentru start cadru – indică faptul că modul de operare pentru compresie este secvențial bazat pe DCT (pentru celelalte moduri există alte delimitatoare) și specifică dimensiunea imaginii, numărul de componente și rata de sub eșantionare a componentelor
- **DHT** - Unul sau mai multe tabele Huffman
- **SOS** - specifică începutul scanării datelor ce descriu conținutul imaginii
- **EOI** - pentru sfârșitul imaginii

# Modul de codare progresiv

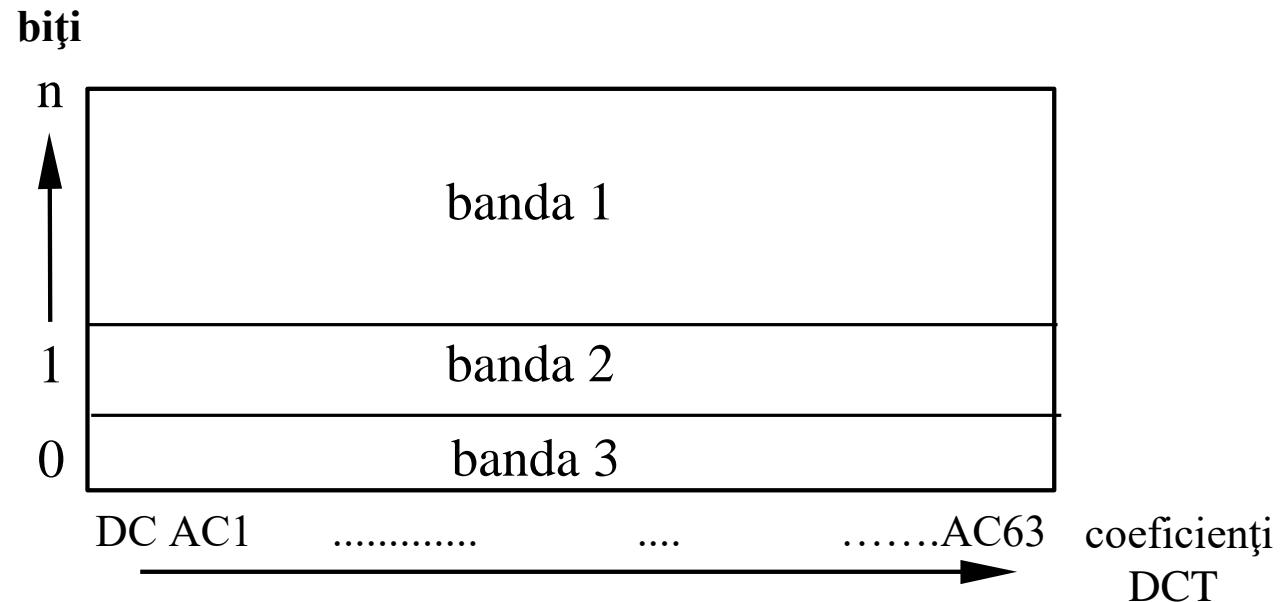
- diferente fata de modul de baza:
  - 8-12 biti pentru reprezentarea planurilor de culoare
  - codare Huffman sau aritmetica
- tipuri de algoritmi
  - algoritm progresiv cu selectie spectrala
  - algoritm progresiv cu aproximari succesive
  - algoritm progresiv combinat



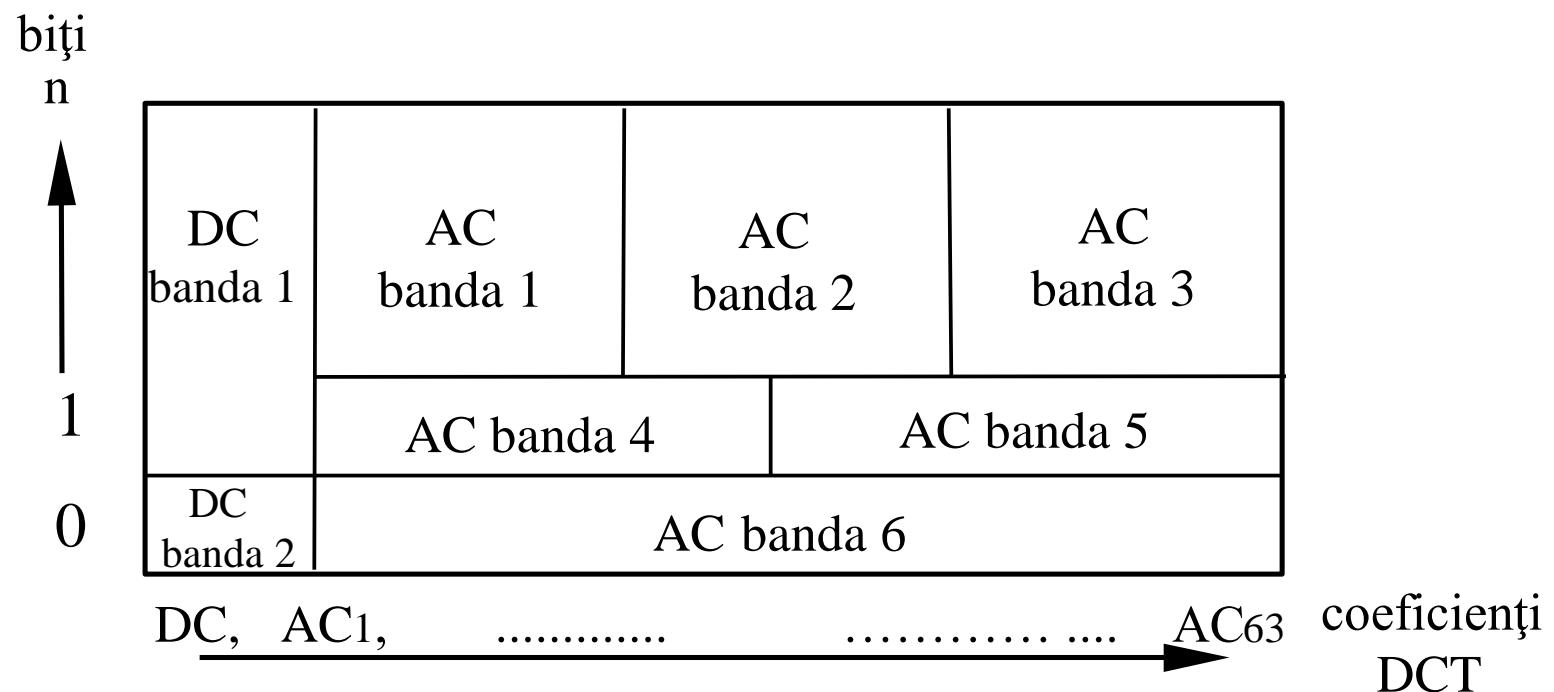
# Algoritmul progresiv cu selectie spectrala



# Algoritmul progresiv cu aproximari succesive



# Algoritmul progresiv combinat



## Etapele de decompresie progresiva



a) imagine la momentul  $t_1$



b) imagine la momentul  $t_2 > t_1$



c) imagine la momentul  $t_3 > t_2$



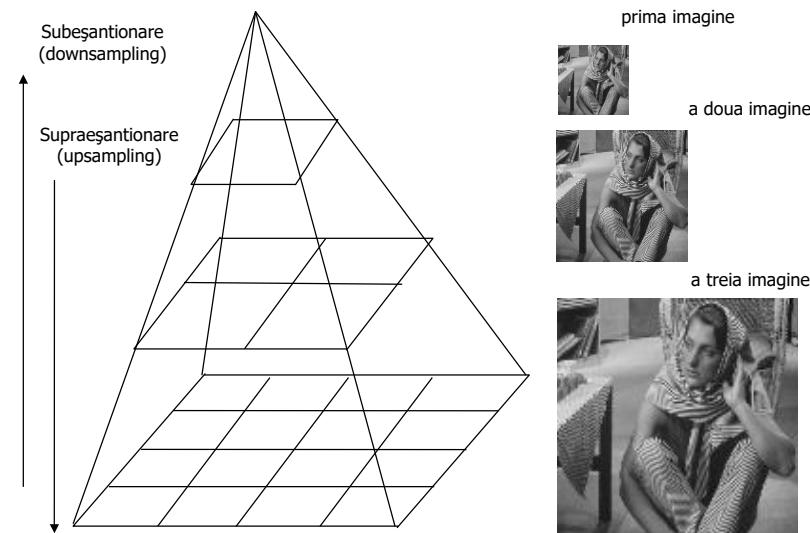
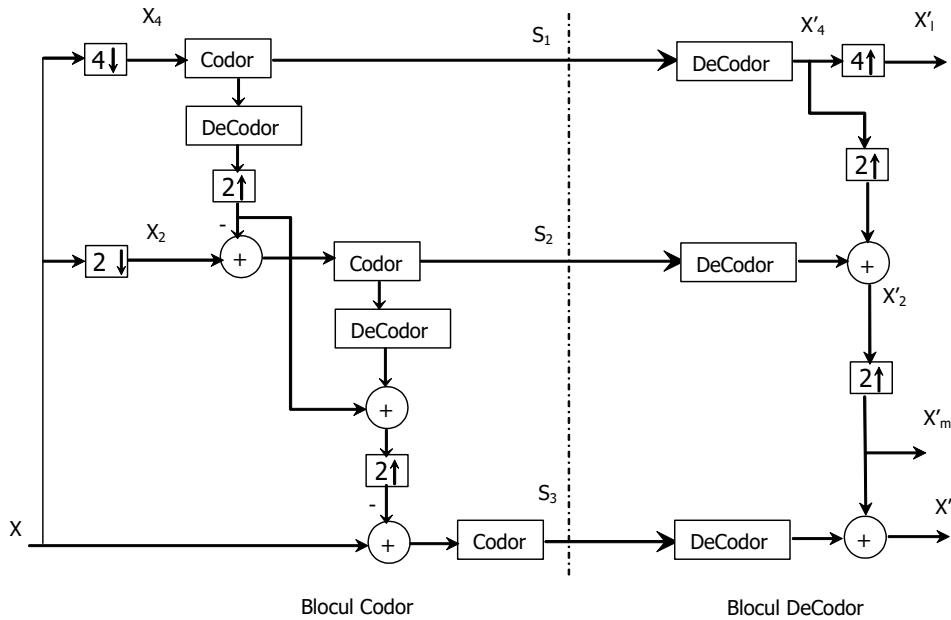
d) imagine originală ( $t=t_4$ )

# Modul ierarhic de operare compresie

- **modul ierarhic**
  - fiecare componentă a imaginii este codat ca o secvență de cadre:
    - primul cadru este o versiune de rezoluție joasă a imaginii originale (subeșantionată).
    - cadrele care urmează sunt cadre diferență între planurile imaginii sursă subeșantionate și planurile de referință reconstruite (supraeșantionate).
  - Cadrele pot fi codate folosind fie
    - JPEG cu pierderi
    - JPEG fără pierderi
    - combinat (ultima etapă JPEG fără pierderi)
  - Codarea ierarhică este folositoare atunci când avem nevoie de imagini la rezoluții multiple.
  - Exemplu: o aplicație poate afișa o imagine la rezoluție înaltă pe o stație grafică performantă și de asemenea poate afișa o imagine la rezoluție joasă pentru un simplu PC.

# Codarea JPEG ierarhica

- Fiecare componentă a imaginii este codat ca o secvență de cadre:
  - primul cadru este o versiune de rezoluție joasă a imaginii originale (subeșantionată).
  - cadrele care urmează sunt cadre diferență între planurile imaginii sursă subeșantionate și planurile de referință reconstruite (supraeșantionate).



# Quantization aspects

- An image at 100% quality has (almost) no loss, and 1% quality is a very low quality image. In general, quality levels of 90% or higher are considered "high quality", 80%-90% is "medium quality", and 70%-80% is low quality. Anything below 70% is typically a very low quality image.
- **Standard Quantization Tables**
  - These tables are referred to as "50%". The JPEG Standard also defines a scaling algorithm that can be used to alter these values to approximate the quality range from 1% to 100%. (This algorithm makes it easy for an application to provide 100 quality levels without hard-coding 100 pairs of quantization tables.) Many applications follow the JPEG Standard and use the sample quantization tables and scaling algorithm to quickly apply a selected quality level.
- **Non-Standard Quantization Tables**
  - The JPEG Standard's quantization tables are explicitly provided as an *example*; compliance is neither required nor essential. They are called the "Standard Quantization Tables" because they are described as examples in the JPEG Standard document, and not because of some standard requirement to use them.
  - Applications do not need to follow the JPEG Standard when determining quantization tables. Many devices and applications use their own, custom quantization tables. These are referred to as "non-standard" since they do not follow the examples in the JPEG Standard. Adobe, for example, defaults to using non-standard quantization tables and scaling algorithms. Similarly, if your digital camera has quality settings for "High", "Medium", and "Low", then it is actually referring to three hard-coded sets of quantization tables that are often non-standard.
- **JPEG %**
  - The JPEG % algorithm evaluates the values in the quantization tables. If the tables align with the JPEG Standard, then it identifies the required scaling factor. For non-standard tables, the algorithm estimates the percentage needed to achieve the same quality using the JPEG Standard values.
  - The results from JPEG % include:
    - The percentage needed to achieve the same quality using the JPEG Standard values.
    - Whether the percent value matches the JPEG Standard or is an estimate based on non-standard quantization tables.
    - The raw quantization tables extracted from the JPEG.
    - If the picture is not a JPEG, then JPEG % identifies the file as lossless and equivalent to 100% quality.

- The JPEG Standard ([CCITT/ITU T.81 Annex K](#) and [RFC 2435 section 4.2](#)) defines an approach that uses a scalar to adjust a set of well-defined quantization tables. However, the JPEG Standard uses two separate algorithms; one computation is used when scaling between 50% and 100% quality, and a different algorithm scales between 1% and 50% quality.
- For JPEG types 0 and 1 (and their corresponding types 64 and 65), Q values between 1 and 99 inclusive are defined as follows. Other values less than 128 are reserved. Additional types are encouraged to use this definition if applicable.
- Both type 0 and type 1 JPEG require two quantization tables. These tables are calculated as follows. For  $1 \leq Q \leq 99$ , the Independent JPEG Group's formula [5] is used to produce a scale factor S as:

$$S = 5000 / Q \quad \text{for } 1 \leq Q \leq 50$$

$$S = 200 - 2 * Q \quad \text{for } 51 \leq Q \leq 99$$

$$Lq[i,j] = (\text{jpeg\_luma\_quantizer}[i,j] * s + 50) / 100;$$

- This value is then used to scale Tables K.1 and K.2 from [1] (saturating each value to 8 bits) to give quantization table numbers 0 and 1, respectively.

- **Adobe**
- The JPEG Standard defines one approach for determining quantization tables, but it is not the only approach. For example, Adobe Photoshop offers multiple scaling methods:
- **Save As.** One method used by Photoshop is seen when using "Save As" and allows the user to select one of 12 quality levels with names like "Maximum", "High", and "Low".
- **Save for Web.** Another method appears when using "Save for Web" and permits the user to select a quality value from 0 to 100. However, saving an image with Photoshop at "75" is not the same as saving a JPEG using the JPEG Standard algorithm at 75% quality.
- **Advanced.** Photoshop includes an advanced option to save using the JPEG Standard algorithm rather than its own quantization tables. However, this option is buried in the menus and the location varies by software version.

# SACCDMM - Curs 12

## Procesarea imaginilor JPEG direct in domeniul comprimat

Sl.Dr.Ing. Camelia FLOREA

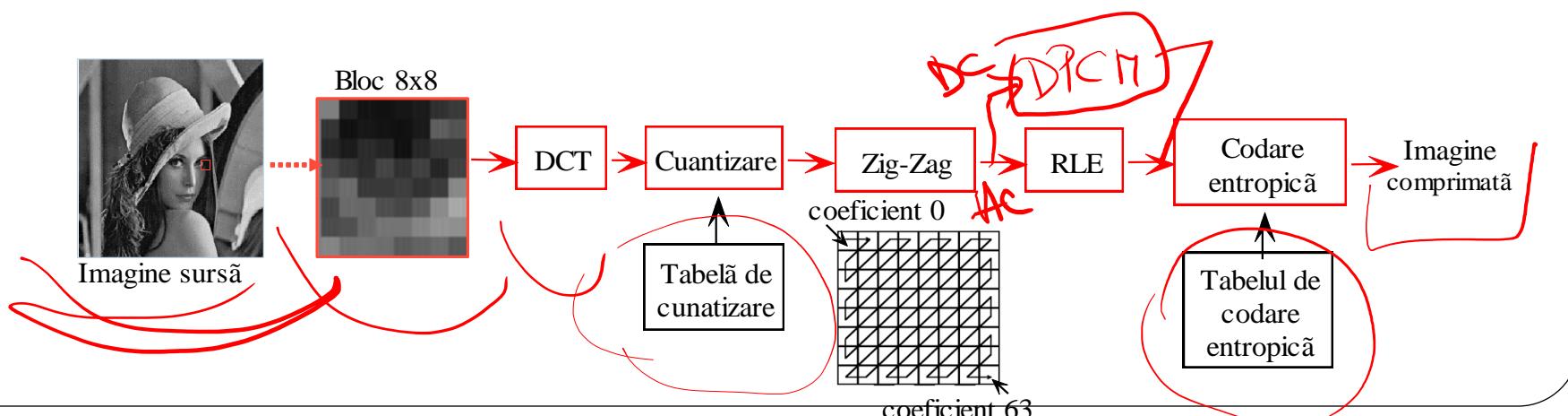
# Introducere

- Odată cu **dezvoltarea exponențială a tehnologiilor care implică informație digitală**, pornind de la simplul format text și ajungând la imagini digitale de înaltă rezoluție, apare și necesitatea manevrării într-un mod optim a acestora. Astfel, problemele deschise de necesitatea **transmiterii** imaginilor la distanță, **stocarea** lor pe compact discuri cu păstrarea calității la o dimensiune binară minimă, **integrarea** datelor în diferite contexte și medii, precum și **procesarea acestora în timp real**, trebuie abordate cu atenție.
- Trebuie menționată în acest context **eficiența formatului JPEG**, un format utilizat pe scară largă pentru stocarea digitală a imaginilor. Pe lângă o rată înaltă de compresie, apare necesitatea de păstrare a calității informațiilor originale, formatul JPEG îmbinând cele două deziderate.
- Astfel, devine naturală tendința de prelucrare a imaginilor JPEG direct în domeniul comprimat. Operațiile de prelucrare a imaginilor JPEG în domeniul comprimat pot opera adesea mult mai repede decât corespondentele din domeniul spațial fapt dat de evitarea decompresia/recompresia înainte/după prelucrare și de faptul că în domeniul comprimat se procesează mai puține date. Rămân însă câteva probleme de rezolvat: nu există tehnici adecvate în domeniul comprimat de efectuare a tuturor operațiilor neliniare.

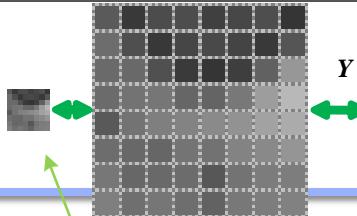
- În ultima perioadă, a existat un interes deosebit în dezvoltarea unor tehnici de căutare a imaginilor după conținutul lor în baze de date, pe web, etc...
- Librariile digitale manipulează foarte multe imagini – datorită limitelor date de spațiul de stocare și timpului necesar manipулării lor – informația este reprezentată în format comprimat => tehniciile utilizate pentru segmentarea, clasificarea sau indexarea imaginilor direct în domeniul comprimat au devenit un subiect important în domeniul librariilor digitale.
- Dintre formatele de compresie, formatul JPEG a fost utilizat mai mult decât celelalte (mai mult de 95 % din imaginile disponibile pe internet sunt stocate ca JPEG-uri)
- Chiar și majoritatea imaginilor medicale sunt stocate în formatul JPEG, datorită avantajelor oferite de acest format:
  - Spațiul redus necesar pentru ~~stocarea~~ imaginilor, și
  - Performanțe superioare la transmiterea imaginilor (via internet).

# Compresia datelor utilizând formatul JPEG

- Spatiul de culoare (RGB  $\rightarrow$  YUV)  $YCbCr$
- Imaginea este impartita in blocuri de 8x8 pixeli
- Valorile sunt scalate simetric fata de 0 (din [0, 255] in [-128, 127])  $\leftarrow 128$
- Fiecare bloc de 8x8 pixeli este procesat pentru compresie
  - DCT este aplicata pe fiecare bloc  $\Rightarrow$  se obtin coeficientii DCT (DC si AC)
  - Coeficientii DCT sunt cuantizati – coeficientii care au valori mici sunt cuantizati la zero
  - Ordonarea in zig-zag a blocului DCT
  - RLE (Run Length Encoding)
  - Codarea entropica



**Extragerea unui bloc de  $8 \times 8$  pixeli din componenta Y și aplicarea pașilor pentru compresie în format JPEG.**



Bloc de  $8 \times 8$  pixeli extras din componenta de luminanță



Componenta de luminanță a imaginii

4	3	2	4	6	10	13	16
3	3	3	5	7	14	15	14
2	3	4	6	10	14	17	14
4	5	6	9	13	22	20	16
6	7	10	13	17	28	26	19
10	14	14	22	28	26	28	23
13	15	17	20	26	28	30	26
16	14	14	16	19	23	26	25

Tabela de cuantizare

00	01	02	03	04	05	06	07
08	09	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Scanare în Zig-Zag

90	57	76	77	64	71	78	53
109	79	56	70	73	68	57	85
106	105	80	58	53	62	97	151
107	115	118	102	100	120	157	184
89	115	127	131	138	146	162	171
109	104	102	116	114	131	137	149
119	103	105	108	87	115	124	125
127	111	118	110	98	120	125	129

$Y_S$

-38	-71	-52	-51	-64	-57	-50	-75
-19	-49	-72	-58	-55	-60	-71	-43
-22	-23	-48	-70	-75	-66	-31	23
-21	-13	-10	-26	-28	-8	29	56
-39	-13	-1	3	10	18	34	43
-19	-24	-26	-12	-14	3	9	21
-9	-25	-23	-20	-41	-13	-4	-3
-1	-17	-10	-18	-30	-8	-3	1

$Y_{DCT}$

-181.12	-58.32	73.49	-18.54	7.62	11.75	9.20	-7.35
-134.15	30.04	12.30	3.86	15.69	-5.92	-2.77	6.69
-107.32	70.10	-19.33	33.69	6.99	23.20	10.06	-1.74
6.09	-11.82	-55.59	21.18	-3.29	15.1	-2.20	-0.31
53.62	-23.28	-27.78	-7.03	-19.12	11.52	0.54	10.92
-6.22	-2.55	20.13	-10.91	-12.62	15.78	1.67	2.66
-7.90	-3.07	15.56	-9.11	-3.03	-0.86	-0.41	3.14
8.01	0.18	-0.99	-11.05	-7.15	-0.77	-2.87	1.97

$Y_Q$

-45	-19	37	-5	1	1	1	0
-45	10	4	1	2	0	0	0
-54	23	-5	6	1	2	1	0
2	-2	-9	2	0	1	0	0
9	-3	-3	-1	-1	0	0	1
-1	0	1	0	0	1	0	0
-1	0	1	0	0	0	0	0
-1	0	0	-1	0	0	0	0

$Y_{ZZ}$

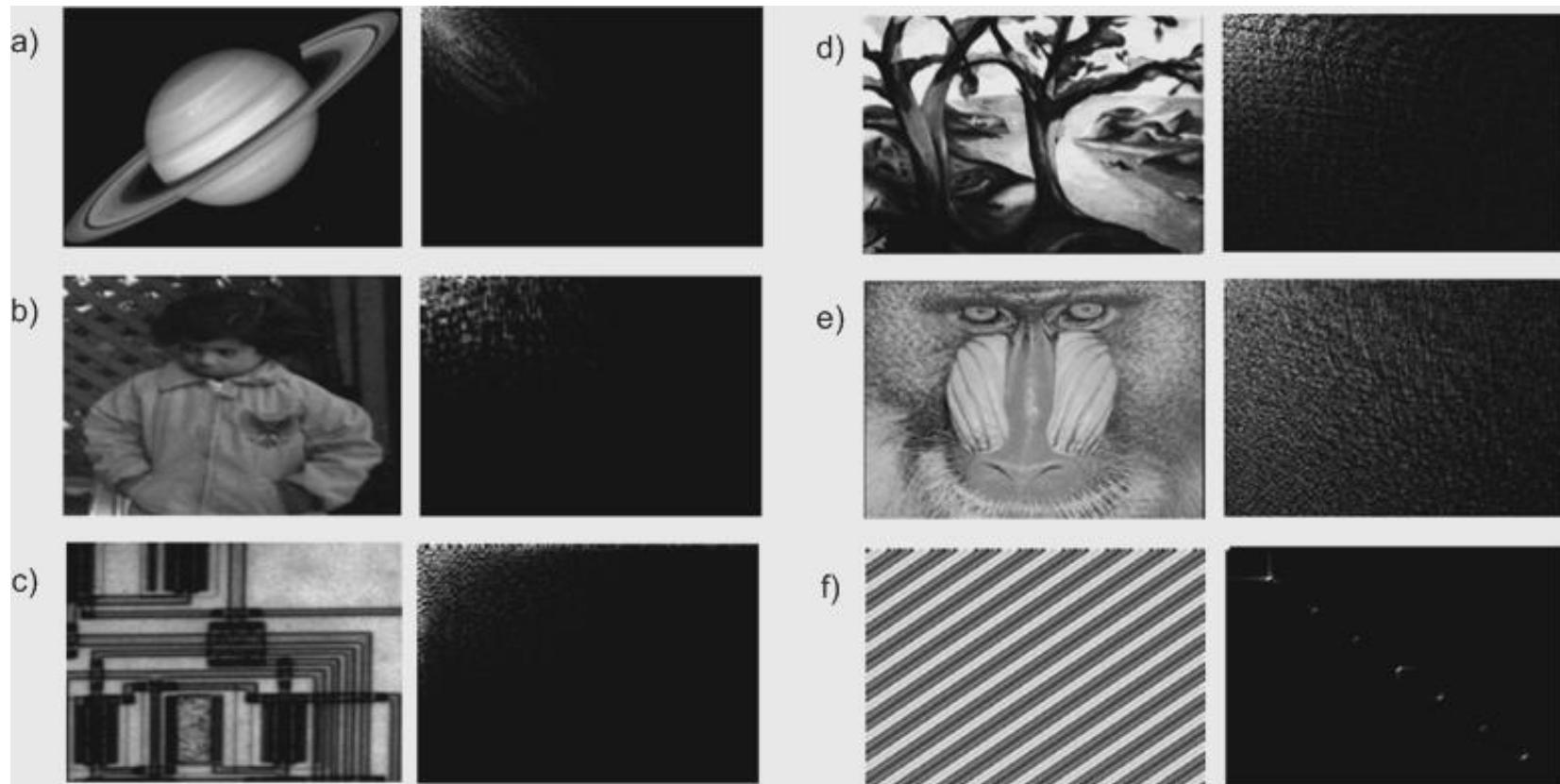
[ -45 -19 -45 -54 10 37 -5 4 23 2 9 -2 -5 1 1 1 2 6 -9 -3 -1 -1 0 -3 2 1  
0 1 0 0 2 0 -1 1 0 -1 0 1 0 -1 1 1 0 0 0 0 0 0 0 -1 0 1 0 0 1 0 0 0 0 0  
0 0 0 0 ]

$Y_{RLE}$

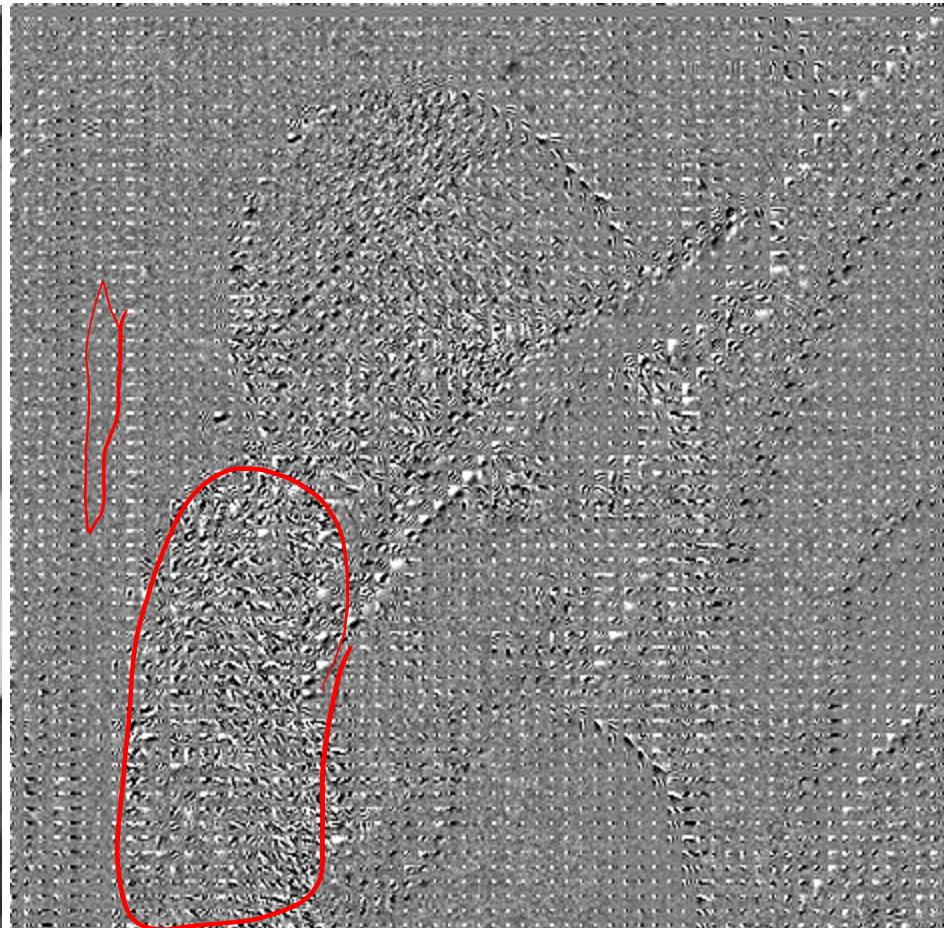
[ -45 0 -19 0 -45 0 -54 0 10 0 37 0 -5 0 4 0 23 0 2 0 9 0 -2 0 -5 0 1 0  
1 0 1 0 2 0 6 0 -9 -3 0 -1 0 -1 1 -3 0 2 0 1 1 1 2 2 1 -1 0 1 1 -1 1 1  
1 -1 0 1 0 1 7 -1 1 1 2 1 0 0 ]

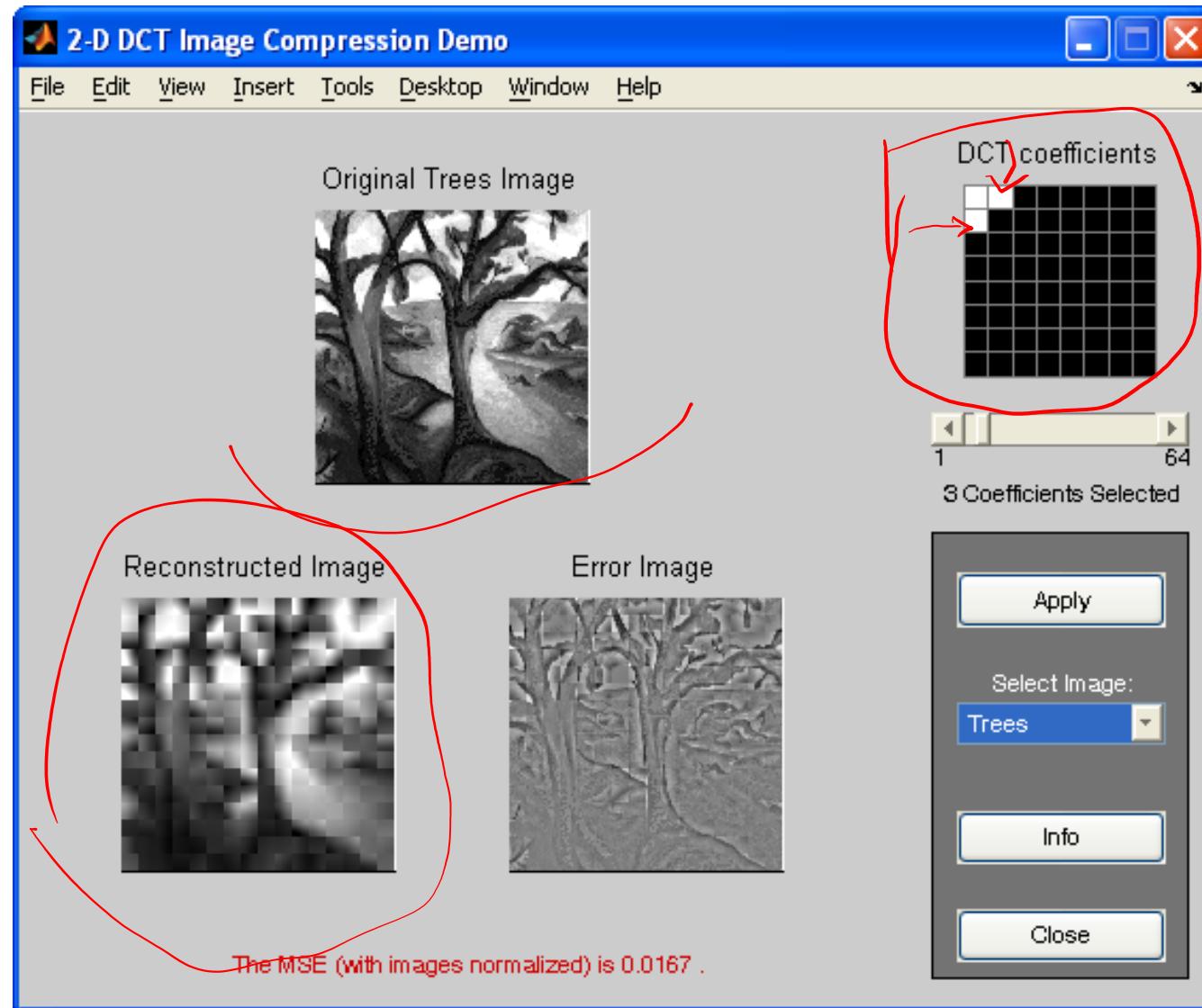
# Coeficienții DCT

- În coeficienții DCT este prezentă toată informația din blocul spațial de imagine, într-o altă formă de reprezentare:
  - Coeficientul DC – luminanță/crominanță medie dintr-un bloc de pixeli
  - Coeficienții AC – modul de variație a luminanței/crominanței din acel bloc (față de coeficientul DC)



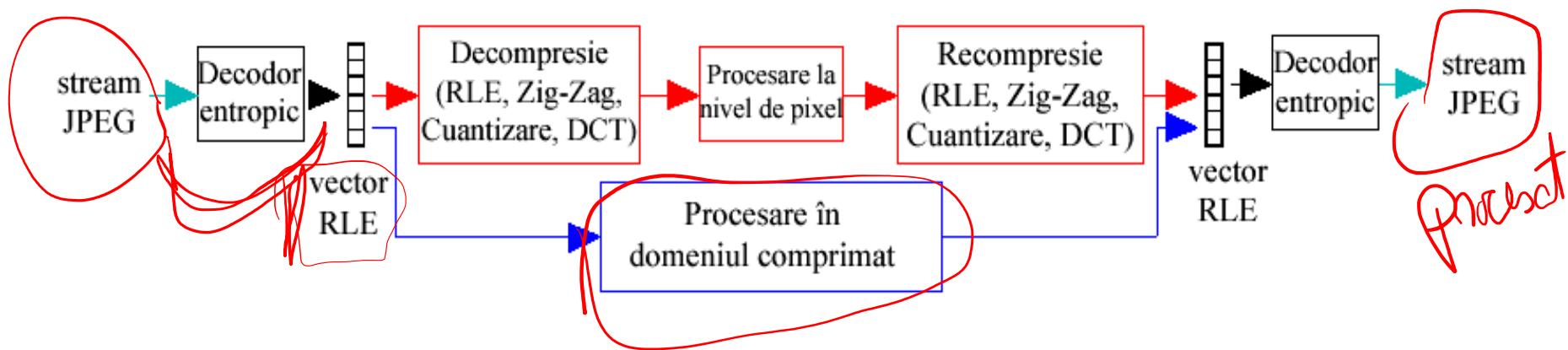
# Imaginea in domeniul comprimat JPEG





# Procesarea imaginilor comprimate JPEG

- Există două moduri de procesare:



- Avand imaginea stocata in formatul comprimat JPEG este mult mai eficient sa fie procesata direct, fara:
  - realizarea decompresiei, procesarea la nivel de pixel, si recompresia.
- Procesarea in domeniul comprimat este realizata pe vectorul RLE.
- Vectorul RLE contine informatie despre valoarea medie si modul de variatie a luminantei/ crominantei din blocul de 8x8

- Reprezentarea imaginii în domeniul transformatei cosinus discrete (DCT) este mai eficientă din perspectiva:
  - decorelării datelor comparativ cu reprezentarea spațială
  - compactării energiei totale a imaginii într-un număr relativ mic de coeficienți;
- Coeficienții DCT din blocuri de  $8 \times 8$  pixeli disponibili în formatul JPEG furnizează direct informații de culoare și textură locală, în formă compactă, fără transformări suplimentare
  - => putem extrage ușor discriminatori de textura (ex. energia de curent alternativ (AC) din fiecare bloc);
  - => putem clasifica blocurile de pixeli în domeniul comprimat  $\Leftrightarrow$  segmentare a imaginilor JPEG în domeniul comprimat

# Operații liniare de prelucrare a imaginilor în domeniul comprimat

- Prelucrarea imaginilor în domeniul comprimat  $\Leftrightarrow$  transpunerea algoritmilor de prelucrare spatială a pixelilor în domeniul comprimat;
- Transpunerea algoritmilor liniari în domeniul comprimat:
  - foarte simplă – datorită liniarității DCT
    - => algoritmi mult mai rapizi prin:
      - evitarea decompresiei și recompresiei;
      - reducerea numărului de date de prelucrat fata de domeniul spațial (după cuantizare majoritatea coeficienților AC sunt zero)
    - => categoria principala de operații: “*aritmetică pixelilor*”: adunare cu o constantă, înmulțire cu o constantă, adunarea pixelilor, înmulțirea pixelilor

# Operații neliniare de prelucrare a imaginilor în domeniul comprimat

- Transpunerea algoritmilor neliniari în domeniul comprimat:
  - nu este directă; unii algoritmi sunt dificil de reformulat;
  - în ipoteza determinării relației corecte pentru implementare în domeniul comprimat, ofera avantajul unei complexitati numerice reduse
  - reprezinta o tendinta actuala in domeniu

# Algoritmii liniari de prelucrare a imaginilor în domeniul comprimat

- Aritmetica pixelilor
  - adunare scalară (se adună cu o constată valoarea luminanței din fiecare pixel);
  - înmulțire scalară (înmulțește valoarea luminanței din fiecare pixel din imagine cu o constată);
  - adunare a pixelilor (adună valorile luminanțelor pixelilor din două imagini);
  - înmulțire a pixelilor (înmulțește valorile luminanțelor pixelilor din două imagini);

$$h[i, j] = \alpha \cdot f[i, j] \Leftrightarrow H_Q[u, v] = \frac{\alpha \cdot q_F(u, v)}{q_H(u, v)} \cdot F_Q[u, v].$$

$\mathcal{F} = \text{DCT}(f)$

Coef. DC

	Spatial domain signal - x	Transform domain signal - X
Scalar addition	$[f] + \alpha$	$[F] + \begin{bmatrix} 8\alpha/Q_{00} & 0 \\ 0 & 0 \end{bmatrix}$
Scalar Multiplication	$\alpha[f]$	$\alpha[F]$
Pixel Addition	$[f] + [g]$	$[F] + [G]$
Pixel Multiplication	$[f] \bullet [g]$	$[F] \otimes [G]$

# Aplicații a aritmeticii pixelilor

- Negativare

Componenta Y  $\rightarrow f$

$$g(x, y) = 256 - f(x, y) \mid -128$$

$$g(x, y) - 128 = 256 - f(x, y) - 128$$

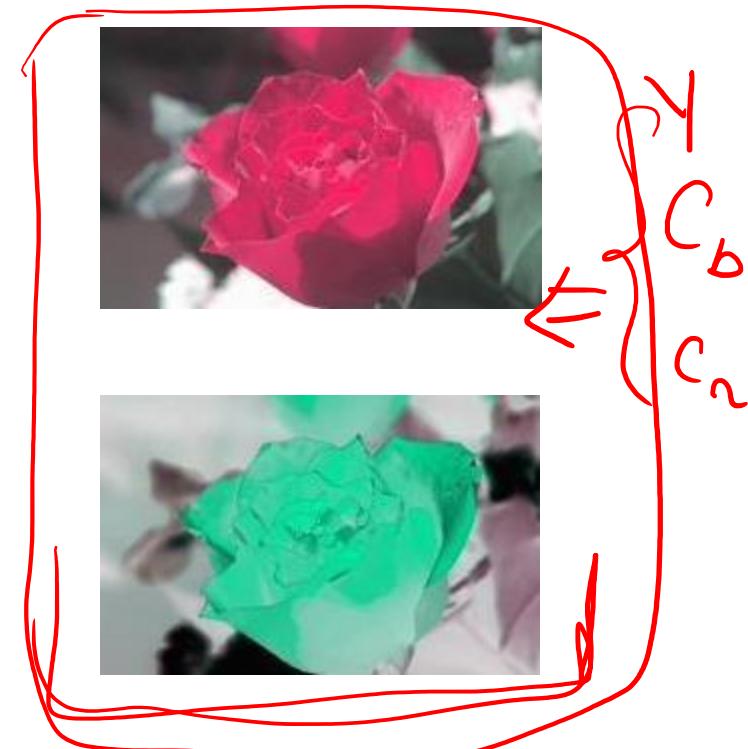
$$g(x, y) - 128 = 128 - f(x, y)$$

$$g(x, y) - 128 = -(f(x, y) - 128)$$

$$DCT(g - 128) = -DCT(f - 128)$$

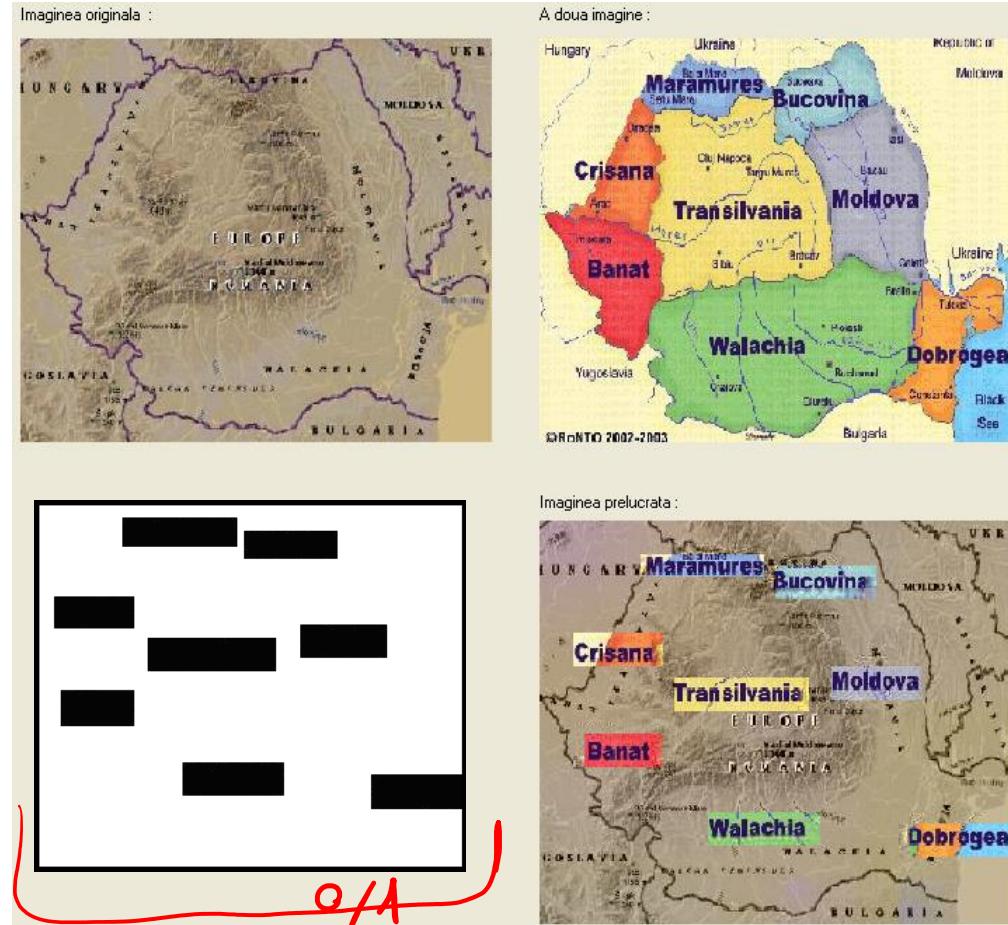
$$\text{Val}_{DCT_{pq}} = -\text{Val}_{DCT_{1pq}}$$

Pas 1 - axene valori tezoare  
 $L = 128$



# Aplicații a aritmeticii pixelilor

- Compunere de imagini  
(ex. suprapunerea  
buletinului meteo pe  
hartă)



$$g(x, y) = \underbrace{m(x, y)}_0 \cdot \underbrace{f(x, y)}_c + (1 - m(x, y)) \cdot b(x, y)$$

$$= \frac{b}{f}$$

# Obturare progresivă

$$g(x, y) = \alpha f_2(x, y) + (1 - \alpha) f_1(x, y)$$

$\underbrace{\alpha f_2(x, y)}_{=0} + \underbrace{(1 - \alpha) f_1(x, y)}_{=0}$



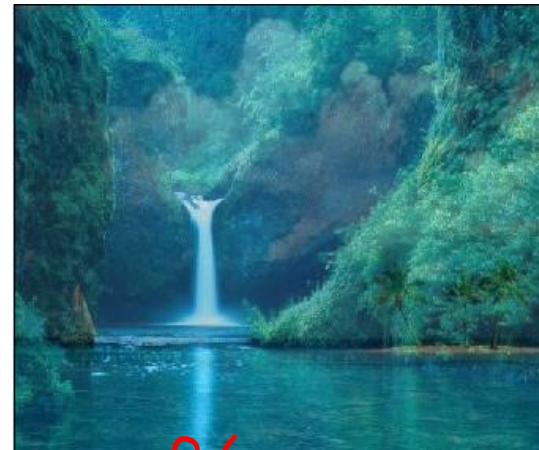
$\alpha=1$



~~$0.8 f_2 + 0.2 f_1$~~



$0.1\alpha$



$0.16$



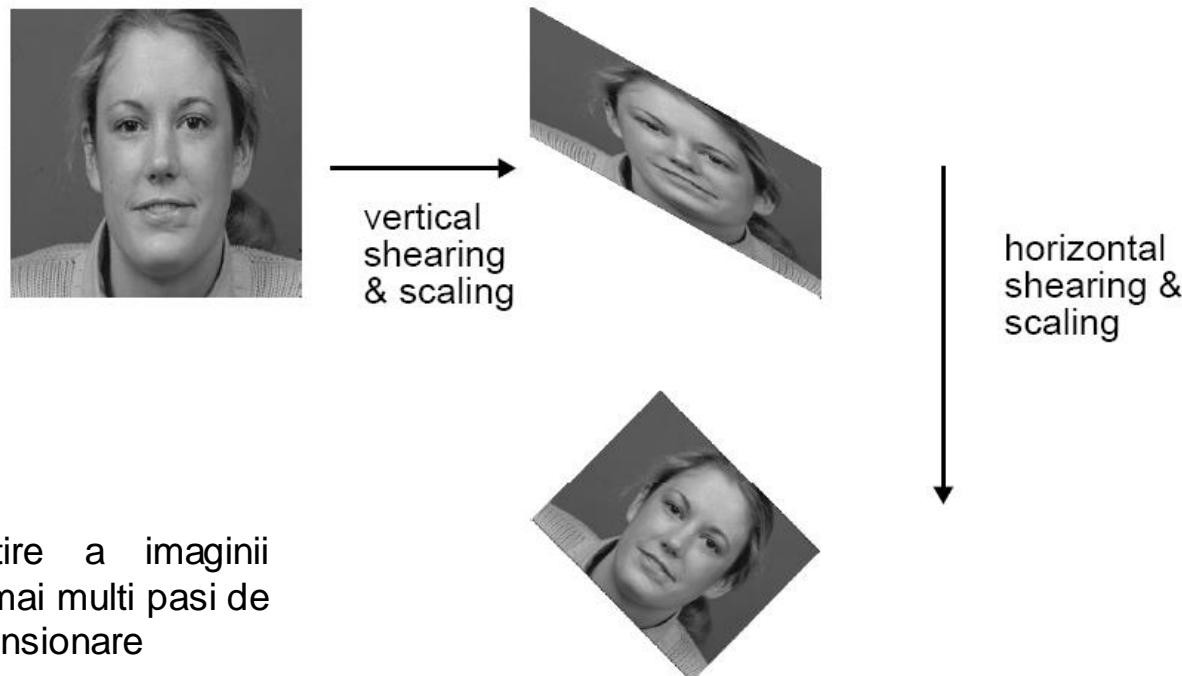
$0.2$



$\alpha=0 \Rightarrow f_1$

# Transformari Geometrice

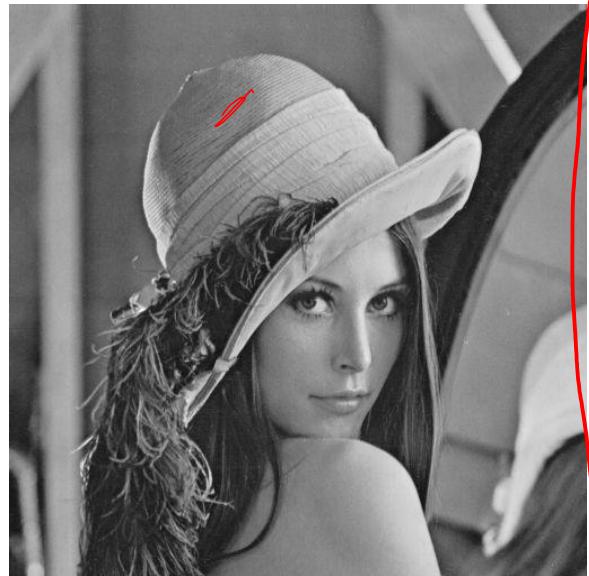
- Tehnici pentru manipularea imaginilor in domeniu comprimat pentru efecte speciale, corectia deformarilor: redimensionarea imaginilor, translatia, rotatia, warping (deformarea).
- Functioneaza prin simpla rearanjare a datelor in domeniu comprimat (coeficientilor DCT), fara a se decompresa intreaga imagine.
- Aceste algoritmi sunt aplicabili pe orice transformata ortogonală cum ar fi: DCT, DFT, ...



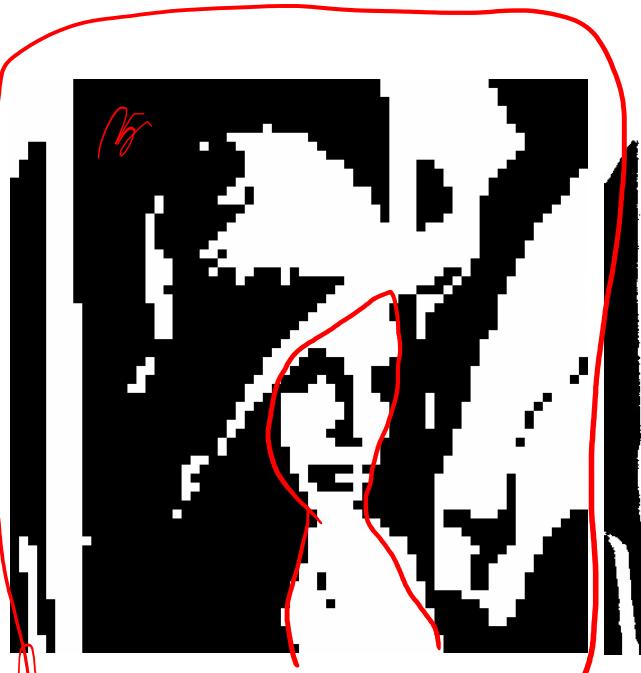
# Implementarea operației neliniare de tip comparare cu prag în domeniul comprimat

- Problema adresată
  - Implementarea direct în domeniul comprimat a unor algoritmi care necesită una sau mai multe comparații cu prag a luminanței fiecărui pixel
  - Coeficienții DCT nu furnizează direct luminanțele pixelilor decât în cazul blocurilor perfect uniforme => pentru implementarea comparației cu prag a fiecarei luminanțe ar fi necesară decompresia
- Abordare
  - Realizarea unui algoritm adaptiv
    - blocurile cu luminanță uniformă se procesează direct în domeniul comprimat
    - dar blocurile care concentrează o cantitate mare de energie de curent alternativ (blocuri neuniforme) vor fi decomprime și se procesate individual - fiecare luminantă.
- Avantaje
  - Decompresia nu mai este necesară pentru toată imaginea
  - Pentru blocurile procesate în domeniul comprimat comparația cu pragul se face o singură dată pentru tot blocul

# Binarizarea unei imaginii



Imaginea originală  
“Lena.jpg”;



Imaginea procesată în  
domeniul comprimat;



Imaginea procesată  
la nivel de pixel

b7 componenti:

→ 1 componentă  
a coef DC

# Utilizarea algoritmului cu decompresie adaptivă

- Pentru fiecare bloc de  $8 \times 8$  pixeli din imaginea digitală comprimată JPEG:

- Calculează energia medie a coeficientilor AC din blocul DCT,  $E_{AC}$ .

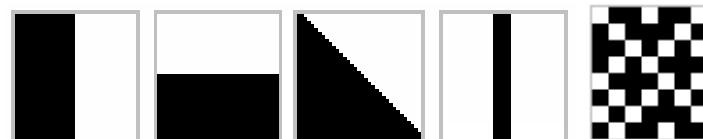
$$E_{AC} = \frac{\sum_{i=1}^{(N_{RLE}-1)/2} (u_{RLE}[2i])^2}{(N_{RLE}-1)/2}$$

- Dacă  $E_{AC} < e_{\text{thd}}$

=> blocul este suficient de uniform pentru a fi procesat direct în domeniul comprimat.

Dacă  $E_{AC} \geq e_{\text{thd}}$

=> blocul are un număr semnificativ de detalii => se va decomprima; fiecare pixel din bloc este comparat individual cu pragul de luminanta.



- Valoarea  $e_{\text{thd}}$  = valoarea parametrului de selecție a blocurilor uniforme fata de blocurile cu număr semnificativ de detalii.

# Selectia parametrului $e_{thd}$

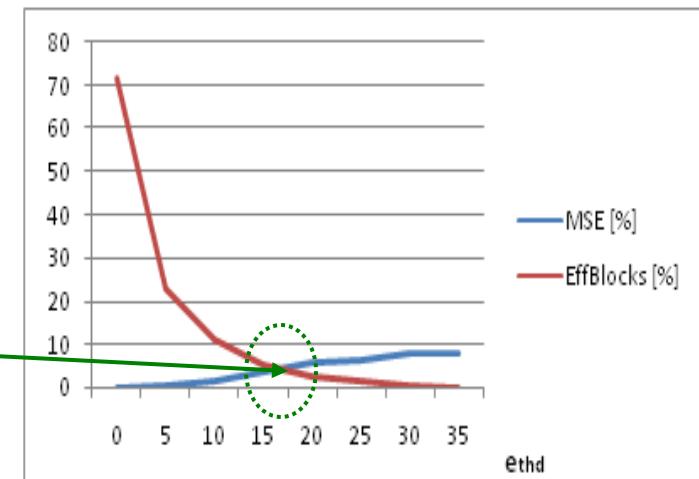
- Pentru o clasă de imagini dată, valoarea  $e_{thd}$  optimă = valoarea care oferă cel mai bun compromis între:
  - complexitatea de calcul și
  - calitatea imaginii procesate.

Determinarea  $e_{thd}$ :

- experimental, prin examinarea variatiilor
  - graficului erorii medii pătrate (MSE - Mean Square Error)  
și
  - graficului EffBlocks = procentul numărului de blocuri decomprime din numărul total de blocuri din imagine (la aplicarea algoritmului propus),

în funcție de valorile parametrului  $e_{thd}$

Regiunea  
corespunzătoare  
valorii optime a  
parametrului  $e_{thd}$



# Binarizarea imaginilor în domeniul comprimat folosind algoritmul adaptiv

- Formularea algoritmului adaptiv de binarizare a imaginilor comprimate JPEG propus:
  1. Alegerea pragului T de binarizare (implicit 128) =>  $DC_{thd}$  (implicit 0).
  2. Calculul conținutului mediu de energie  $E_{AC}$  pentru fiecare bloc de  $8 \times 8$  pixeli din imaginea comprimata JPEG
    - Dacă  $E_{AC} < e_{thd}$ , atunci blocul se poate considera de luminanță uniformă și se binarizează direct în domeniul comprimat:

$$u_{RLE}[0] = \begin{cases} -127, & \text{dacă } u_{RLE}[0] < DC_{thd} \\ 127, & \text{dacă } u_{RLE}[0] \geq DC_{thd} \end{cases}$$

$RLE[0] \rightarrow \text{coef. DC}$

$$u_{RLE}[1] = 0, \quad u_{RLE}[2] = 0, \quad N_{RLE} = 3$$

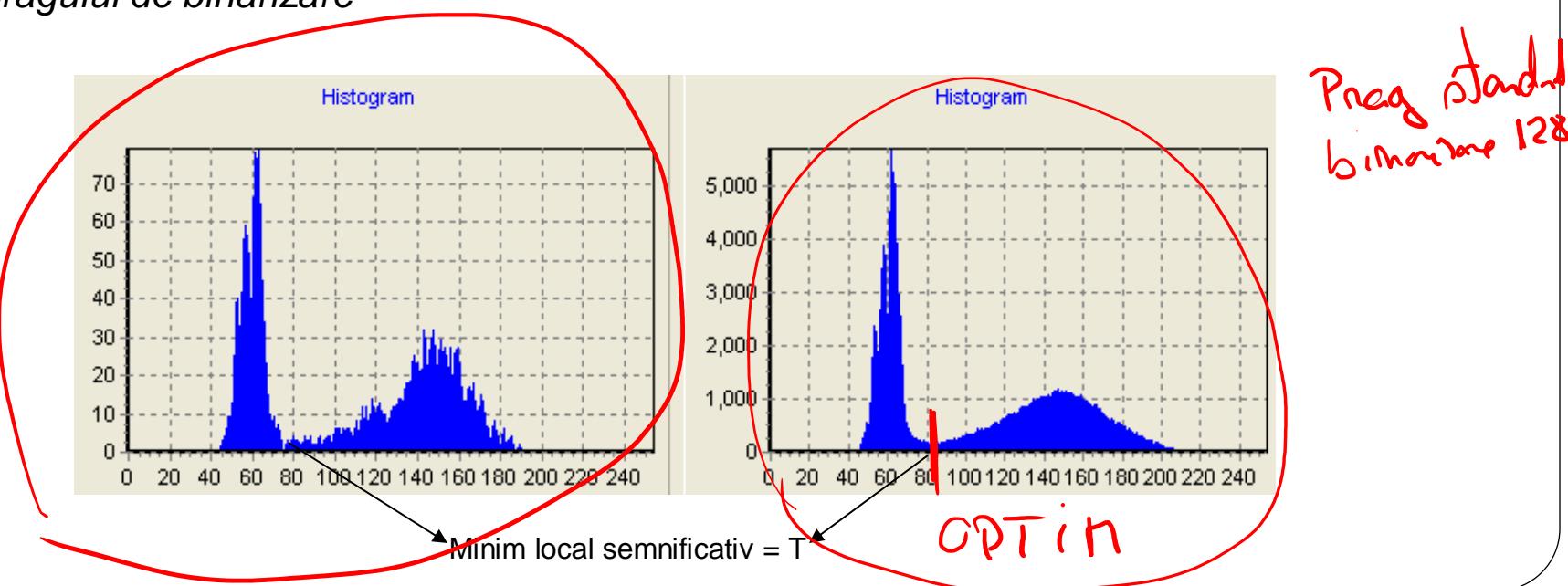
$EOB(0,0)$   
 $\text{coeficienți AC} = 0$

- Dacă  $E_{AC} > e_{thd}$ , blocul are un conținut semnificativ de detaliu și trebuie decomprimat - binarizarea fiecarui pixel în parte:

$$b(i, j) = \begin{cases} 0, & \text{dacă } u(i, j) < T \\ 255, & \text{dacă } u(i, j) \geq T \end{cases} \quad i = 0, 1, \dots, 7, \quad j = 0, 1, \dots, 7$$

# Selectia pragului de binarizare fara decompresie:

- Procedura curenta (la prelucrarea in domeniul spatial): examinarea histogramei; pragul de binarizare  $\Leftrightarrow$  un minim local in histograma
- În domeniul comprimat, nu avem acces la valorile luminantelor pixelilor pentru a realiza histograma imaginii  
 $\Rightarrow$  putem construi o histograma aproximativa utilizand coeficientii DC,  
 $\Leftrightarrow$  o aproximare a histogramei nivelerelor de gri, cu pierderea detalilor, dar *pastrarea minimelor si maximelor locale semnificative*  
 $\Rightarrow$  *histograma coeficientilor DC ai blocurilor ofera un bun indicator vizual pentru selectia pragului de binarizare*



Imagine binarizată în domeniul comprimat

blocuri cu  
energie mare sunt dezcomprimate

## Rezultate experimentale

Rezultatele pentru o valoare  $e_{thd} = 0.1$

Imagine	EffBlock s [%]	Numărul de comparații cu prag pe imagine		MSE [%]
		Binarizare clasică	Binarizare cu algoritmul propus	
Img1.jpg	13.5	325.071	414.720	1.45
Img2.jpg	16.07	86.016	40.593	0.48
Img3.jpg	24	96.768	46.179	0.012
Img4.jpg	18.24	262.144	173.881	1.45

Imagine binarizată în domeniul spațial

Diferența dintre cele două rezultate



Imagine binarizată adaptiv  
în domeniul comprimat



Imagine binarizată în  
domeniul spațial

# Algoritmii neliniari de prelucrare a imaginilor în domeniul comprimat

- Pt. majoritatea algoritmilor neliniari e dificilă transpunerea lor în domeniul coeficienților DCT, dar odată ce combinația corectă este găsită (pentru reformulare în domeniul comprimat) algoritmul este mult mai rapid.
- Chiar și dacă forma transpusă în domeniul comprimat este de multe ori mai complexă, este totuși mult mai rapidă:
  - Unu, se evită decompresia la nivel de pixel, procesarea pixelilor, recompresia.
  - Doi, se procesează mult mai puține date, deoarece după cuantizare majoritatea coeficienților (cei care aparțin frecvențelor înalte) devin zero

- Ex: Calculul pătratelor pixelilor dintr-o imagine:  
(se folosește formula de convoluție în domeniul comprimat)

$$\begin{aligned} U_{dct,sq}(x_1, x_2) &= \frac{1}{4 \cdot Q(x_1, x_2)} \sum_i \sum_j C(i, x_1) \cdot C(j, x_2) \cdot [u(i, j)]^2 = \\ &= \sum_{y_1, y_2, w_1, w_2} U_{dct}(y_1, y_2) \cdot U_{dct}(w_1, w_2) \cdot W_Q(y_1, y_2, w_1, w_2, x_1, x_2) \end{aligned}$$

unde :

$$W_Q(y_1, y_2, w_1, w_2, x_1, x_2) = \frac{Q(y_1, y_2) \cdot Q(w_1, w_2)}{256 \cdot 64 \cdot Q(x_1, x_2)} W(x_1, y_1, w_1) \cdot W(x_2, y_2, w_2)$$

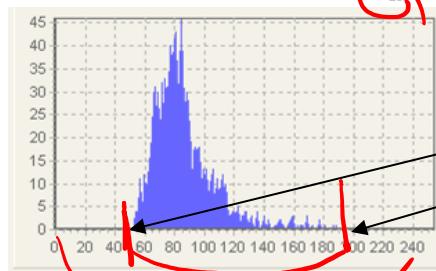
cu :  $W(x, y, w) = \sum_i C(i, x) \cdot C(i, y) \cdot C(i, w)$

$$C(i, x) = A(x) \cos \frac{(2 \cdot i + 1) \cdot x \cdot \pi}{16}; \quad A(x) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } x = 0 \\ 1, & \text{for } x \neq 0 \end{cases}$$

# Îmbunătățirea imaginilor folosind operatorul fuzzy de intensificare a unei funcții de apartenență.

- Pentru a reformula acest algoritm în domeniul comprimat avem nevoie de următoarele operații:
  - liniare: adunare, înmulțire cu o constantă, pătratul fiecărui pixel din imagine (calculat cu formula de conoluție în domeniul comprimat prezentată pe slide-ul anterior)
  - neliniare: comparare cu prag

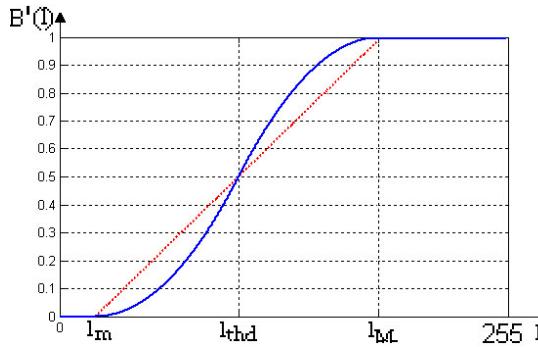
a. Funcția de apartenență pentru reprezentarea dinamicii nivelelor de gri din imagine prin conceptul "Luminos"



$$[l_{\min}, l_{\max}] \subseteq [l_m, l_M],$$

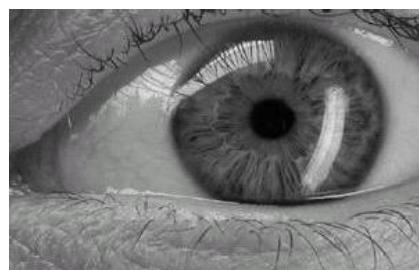
*l - nivel de gri*

$$B'(l) = INT(B(l)) = \begin{cases} 2 \cdot (\alpha \cdot l + b)^2, & \text{dacă } 0 \leq \alpha \cdot l + b \leq 0.5 \\ 1 - 2 \cdot (-\alpha \cdot l + 1 - b)^2, & \text{dacă } 0.5 \leq \alpha \cdot l + b \leq 1 \end{cases}$$



b. Rezultatul aplicării operatorului fuzzy de intensificare asupra funcției de apartenență "Luminos" – reprezentând noua dinamică a nivelelor de gri, în imaginea prelucrată

Rezultatul aplicării operatorului fuzzy de intensificare asupra  $B(l)$  este funcția de apartență:



# Algoritm de îmbunătățire a imaginilor bazat pe reguli fuzzy

Algoritmul de îmbunătățire a imaginilor bazat pe reguli fuzzy, Takagi-Sugeno, este un algoritm cu efect vizual foarte bun, dar ca orice algoritm fuzzy este neliniar, reformularea lui în domeniul comprimat nefiind directă.

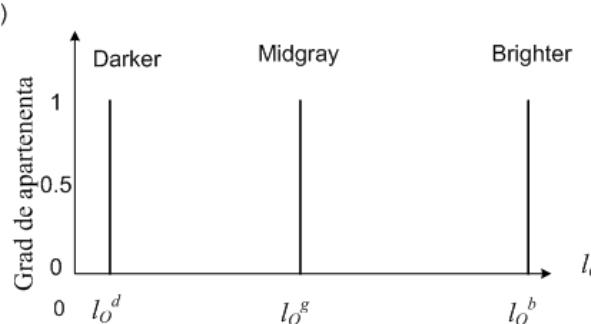
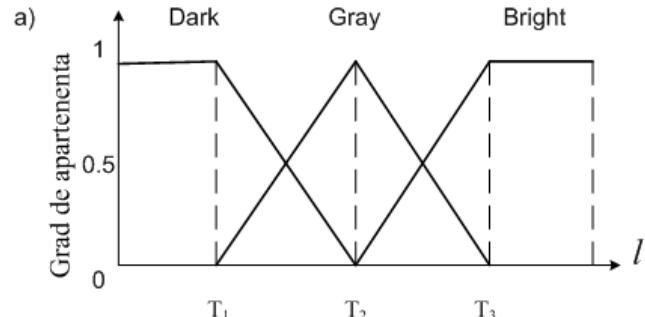
Sistemul de inferenta fuzzy Takagi-Sugeno cuprinde următoarele 3 reguli :

R1: DACĂ  $I_u$  este Dark ATUNCII  $I_v$  este Darker

R2: DACĂ  $I_u$  este Gray ATUNCII  $I_v$  este Midgray

R3: DACĂ  $I_u$  este Bright ATUNCII  $I_v$  este Brighter,

Funcțiile de apartenență de la intrarea și ieșirea sistemului Takagi-Sugeno sunt:



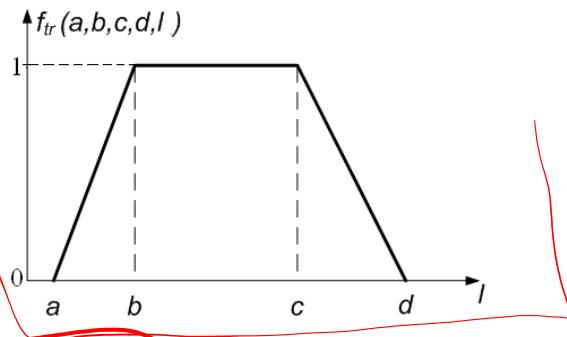
Pentru fiecare intensitate  $l_u^*$  de la intrarea sistemului fuzzy, în imaginea de ieșire, intensitatea corespunzătoare  $l_o^*$  se obține prin aplicarea mecanismului de inferență Takagi-Sugeno:

$$l_v^* = \frac{\mu_{Dark}(l_u^*) \cdot l_o^d + \mu_{Gray}(l_u^*) \cdot l_o^g + \mu_{Bright}(l_u^*) \cdot l_o^b}{\mu_{Dark}(l_u^*) + \mu_{Gray}(l_u^*) + \mu_{Bright}(l_u^*)}$$

unde:  $\mu_{Dark}(l_u^*)$ ,  $\mu_{Gray}(l_u^*)$ ,  $\mu_{Bright}(l_u^*)$  sunt gradele de apartenență ale intensității curent procesate la multimile fuzzy definite peste intrare: *Dark, Gray și Bright*

# Reformularea inferenței fuzzy în domeniul comprimat

Forma grafică și analitică a funcției pentru calculul gradelor de apartenență:



Algoritm clasic

$$f_{tr}(a, b, c, d, l) = \begin{cases} 0, & \text{dacă } l \in [0, a] \\ \frac{l-a}{b-a}, & \text{dacă } l \in [a, b] \\ 1, & \text{dacă } l \in [b, c] \\ \frac{-l+d}{d-c}, & \text{dacă } l \in (c, d] \\ 0, & \text{dacă } l \in (d, 255] \end{cases}.$$

Algoritm propus în domeniul comprimat

$$\begin{aligned} K_s &= [k_1^s, k_2^s] = f_{tr}^{DCT}(a, b, c, d, u_Q(0,0)) = \\ &= \begin{cases} [0 \ 0], & \text{dacă } u_Q(0,0) \in [-128, a-128) \\ \left[ \frac{1}{b-a} \ \frac{128-a}{b-a} \right], & \text{dacă } u_Q(0,0) \in [a-128, b-128) \\ [1 \ 1], & \text{dacă } u_Q(0,0) \in [b-128, c-128) \\ \left[ \frac{-1}{d-c} \ \frac{-128+d}{d-c} \right], & \text{dacă } u_Q(0,0) \in [c-128, d-128) \\ [0 \ 0], & \text{dacă } u_Q(0,0) \in [d-128, 128) \end{cases}. \end{aligned}$$

Blocul obținut prin aplicarea mecanismului de inferență Takagi-Sugeno în domeniul comprimat:

$$U_{Q,Int} = D_Q^{Dark} \cdot (l_o^d - 128) + D_Q^{Gray} \cdot (l_o^g - 128) + D_Q^{Bright} \cdot (l_o^b - 128).$$

unde:  $D_q^s = c_m \cdot U_q + C_a^{dct}$ ,  $s \in \{Dark, Gray, Bright\}$

$$C_a^Q = [k_2^s \ 0 \ 0 \ \dots \ 0], \quad c_m = k_1^s$$

## Rezultate experimentale

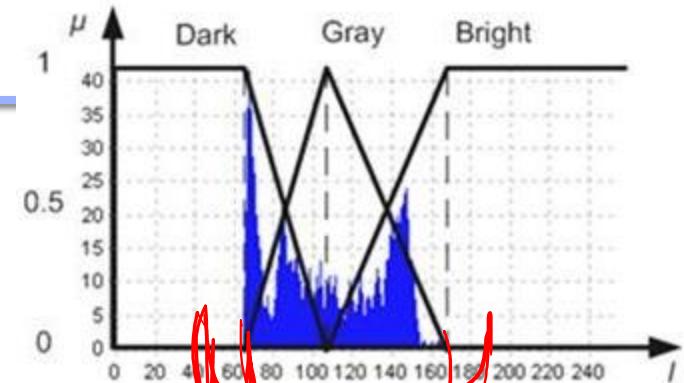
- Algoritmul este aplicat doar pe componenta de luminanță, dar poate fi folosit și pentru accentuarea imaginilor color, cu păstrarea componentelor de crominanță nemodificate



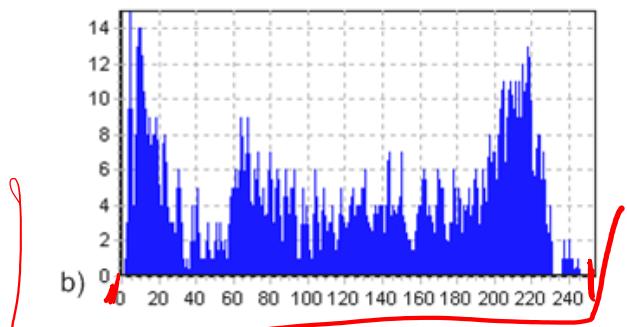
Imaginea originală



Imaginea după aplicarea algoritmului propus



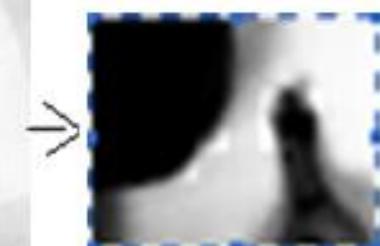
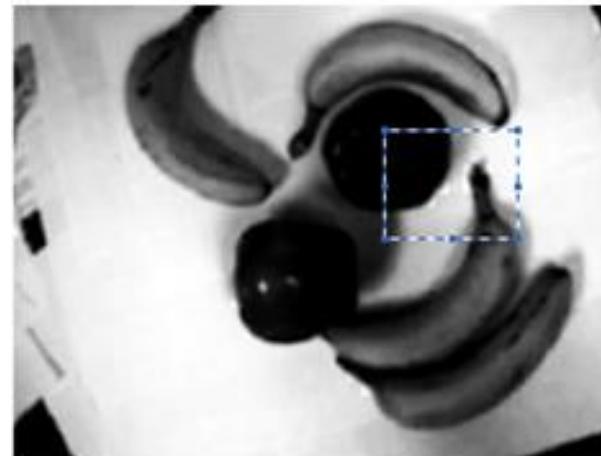
Funcțiile de apartenență de la intrarea sistemului fuzzy și histograma DC a componentei Y pentru imaginea originală



Histograma DC a componentei Y după accentuarea contrastului

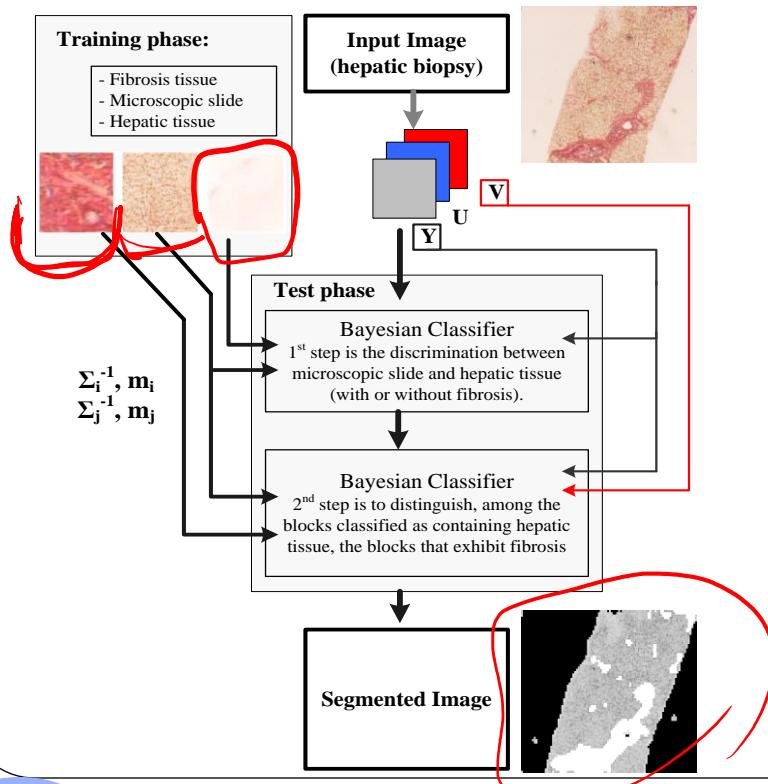
# Probleme care apar la clasificarea tuturor pixelilor din bloc cu coeficientul DC

- Ambele algoritme fuzzy prezentate mai sus implică o (operatorul fuzzy INT) sau mai multe (alg. bazat pe reguli fuzzy) comparații cu prag
- Dacă se clasifică toți pixelii dintr-un bloc, pentru procesare, luând în considerare doar valoarea DC a acelui bloc, este foarte probabil că va apărea efectul de “blocking”
- Acest efect apare în cazul blocurilor care conțin muchii, există variații mari între valorile intensităților conținute
- Pentru evitarea acestui efect dar păstrarea avantajului de procesare a datelor direct în domeniul comprimat, se recomandă decompresia la nivel de pixel doar a blocurilor cu energie mare (variații mari) concentrată în coeficienții DCT, procesarea acestor blocuri la nivel de pixel și recompresia.
- Totuși, numărul acestor blocuri care vor fi decompriimate va fi sub 20%



# Segmentarea imaginilor folosind clasificatorul Bayes și modelul Gaussian direct în domeniul comprimat

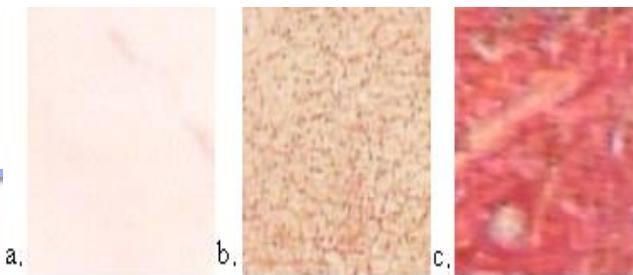
- Un tasc mai dificil este implementarea unor algoritmi care necesită învățare, sau implementarea unor clasificatoare mai complexe, direct în domeniul comprimat.
- Aceasta abordare își propune să fie o rapidă și eficace perspectiva pentru identificarea și cuantificarea fibrozei hepatice din biopsiile de ficat folosind metoda de clasificare a țesuturilor, clasificatorul Bayesian.



Aceasta abordare se bazează pe informația din culori la nivel de pixel, dar și din informația texturilor locale în blocurile vecine de 8x8 pixeli, făcând uz de informațiile existente deja în imaginile microscopice luând în considerare reprezentarea lor în formatul JPEG.

Formatul de compresie a imaginii în sine oferă informațiile necesare unei identificări și segmentări de o acuratețe ridicată a biopsiilor hepatice în țesut vs. lama și mai departe, țesut în țesut sănătos vs. fibros hepatica.

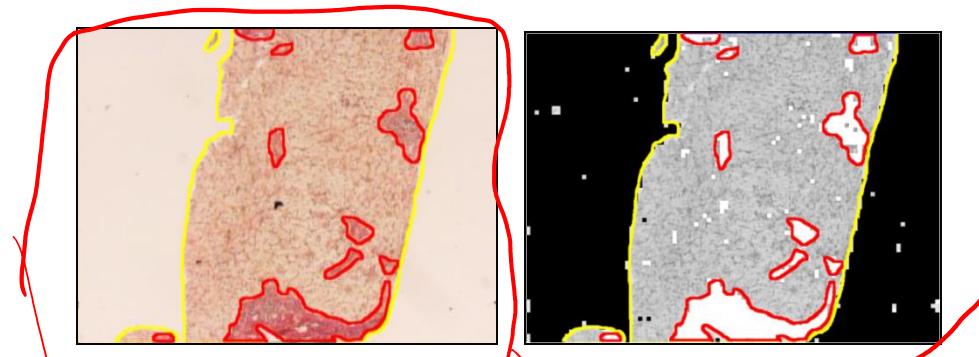
- Rez. Exp. ale metodei demonstrează nu doar o ridicată acuratețe, dar și o clasificare rapidă, datorată domeniului în care avem datele, lucru care permite o clasificare la nivel de bloc în detrimentul unei clasificări la nivel de pixel
  - ⇒ fazele de învățare și de clasificare sau test sunt semnificativ mai rapide.
  - ⇒ acuratețea oferita de folosirea informațiilor din textura este superioara uneia bazata doar pe informația din culorile la nivel de pixel.



Vedere microscopică a biopsiei ficatului: a. Lamă; b. Ţesut sănătos; c. Ţesut fibros



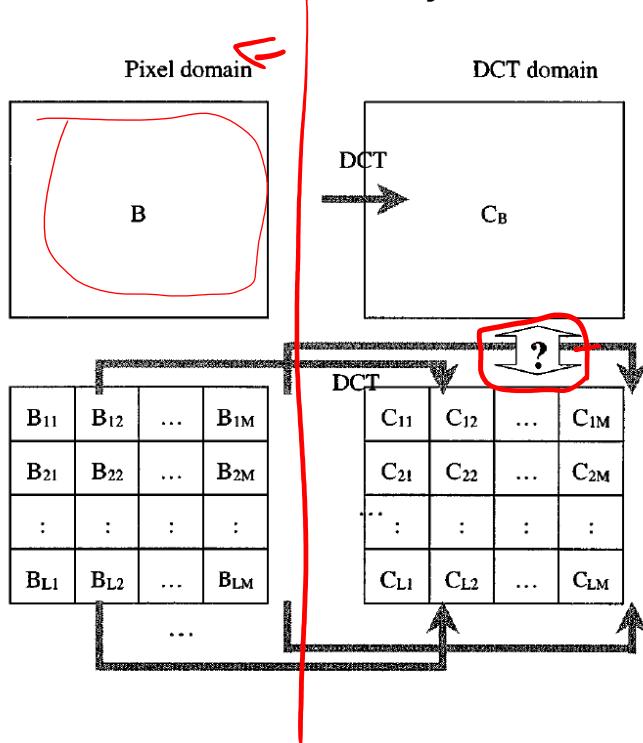
Imaginea hepatică: a) Originală; b) Segmentată la nivel de pixel în spațiul color RGB; c) Segmentată în domeniul comprimat la nivel de bloc DCT;



Diferențele dintre rezultatul segmentării cu algoritmul propus și „ground truth”

# Relația spațială a coeficienților DCT dintre un bloc și sub-blocurile acestuia

- Fiecare element  $C_{i,j}$  reprezintă setul de coeficienți DCT pentru sub-blocul  $SB_{i,j}$  și implicit este o matrice cu  $N \times N$  elemente.
- Tot mai mulți algoritmi de procesare de imagini sunt dezvoltăți în domeniul comprimat pentru a reduce calculele și pentru îmbunătățirea vitezei de procesare, o nouă problemă poate apărea din faptul că trebuie folosite blocuri DCT de dimensiuni diferite pentru a asigura performanțe optimizate:
  - extragerea trăsăturilor globale (blocuri mari), sau
  - extragerea trăsăturilor locale (blocuri mai mici)
- Derivarea directă a coeficienților DCT în subblocuri cu dimensiuni diferite este posibilă:

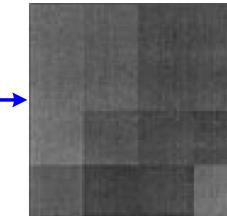


$$\boxed{C_B} = \frac{1}{M} \boxed{A^*} \begin{pmatrix} C_{0,0} & \cdots & C_{0,M-1} \\ \vdots & \ddots & \vdots \\ C_{L-1,0} & \cdots & C_{L-1,M-1} \end{pmatrix} \boxed{A^{*T}}$$
$$\begin{pmatrix} C_{0,0} & \cdots & C_{0,M-1} \\ \vdots & \ddots & \vdots \\ C_{L-1,0} & \cdots & C_{L-1,M-1} \end{pmatrix} = M A^{*-1} \boxed{C_B} A^{*T-1}$$

# Exemplu de obținere a coeficienților DCT pe baza relațiilor între coeficienții DCT ai unui bloc de $4 \times 4$ pixeli și coeficienții DCT ai celor 4 sub-blocuri de $2 \times 2$ pixeli din bloc



Imaginea sursă



Blocul de  $4 \times 4$  pixeli decupat

Transformarea din 4 sub-blocuri adiacente de  $2 \times 2$  pixeli în blocul corespunzător de  $4 \times 4$  pixeli:

106	97	83	85
106	95	84	85
105	84	74	69
77	60	57	89

Blocul cu luminanțele pixelilor

202	10	168	-1
1	-1	84	85
163	19	144	-13
26	2	-1	19

Coef. DCT a 4 blocuri de dimensiune  $2 \times 2$

1	0	1	0
0.9239	0.3827	-0.9239	0.3827
0	1	0	-1
-0.3827	0.9239	0.3827	0.9239

Matricea  $A^*$

339	23	22	-3
34	13	-13	0
-12	-16	8	-5
-1	13	-4	4

Transformarea dintr-un bloc de  $4 \times 4$  pixeli în 4 sub-blocuri adiacente de  $2 \times 2$  pixeli:

106	97	83	85
106	95	84	85
105	84	74	69
77	60	57	89

Blocul cu luminanțele pixelilor

339	23	22	-3
34	13	-13	0
-12	-16	8	-5
-1	13	-4	4

Coef. DCT a blocului de dimensiune  $4 \times 4$

0.5	0.4619	0.5	-0.1913
0	0.1913	0	0.4619
0.5	-0.4619	0.5	0.1913
0	0.1913	0	0.4619

Matricea inversă  $A^*$

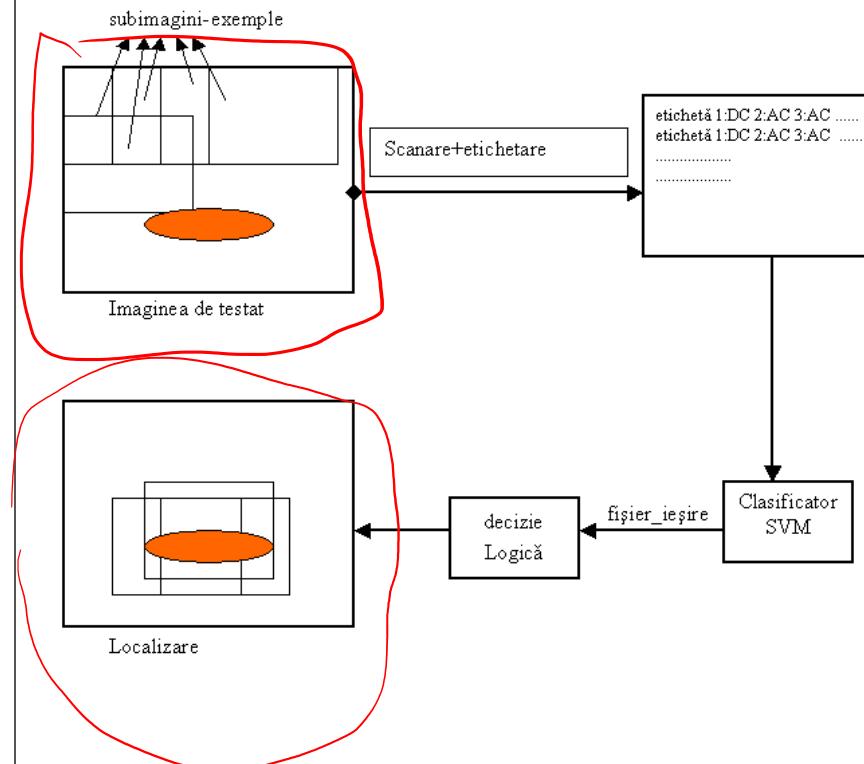
202	10	168	-1
1	-1	84	85
163	19	144	-13
26	2	-1	19

# Aplicarea mașinilor cu vectori suport în recunoașterea vizuală a obiectelor în domeniul comprimat

- Problema adresată
  - Implementarea unui clasificator instruibil direct în domeniul comprimat (mașini cu vectori suport)
  - probleme apărute:
    - în domeniul comprimat avem acces direct doar la blocuri de  $8 \times 8$  pixeli – dimensiune prea restrictiva pentru regiunile de interes de analizat
- Abordarea
  - Utilizarea unui algoritm care oferă o relație directă pentru obținerea unui bloc de coeficienți DCT din sub-blocurile lui (și invers) – pentru setarea regiunii de interes direct în domeniul comprimat
- Avantaje
  - Se obțin regiunile de interes direct în domeniul comprimat fără necesitatea extragerii suplimentare a trasaturilor
  - O regiune de interes în domeniul comprimat va avea un număr redus de valori nenele – crește viteza de clasificare

# MVS direct în domeniul comprimat

- Mașinile cu vectori-suport sunt clasificatoare binare bazate pe învățarea statistică din exemple, care se bucură de un succes deosebit la ora actuală datorită performanțelor lor foarte bune în multe aplicații de analiză a imaginilor și recunoaștere a formelor.

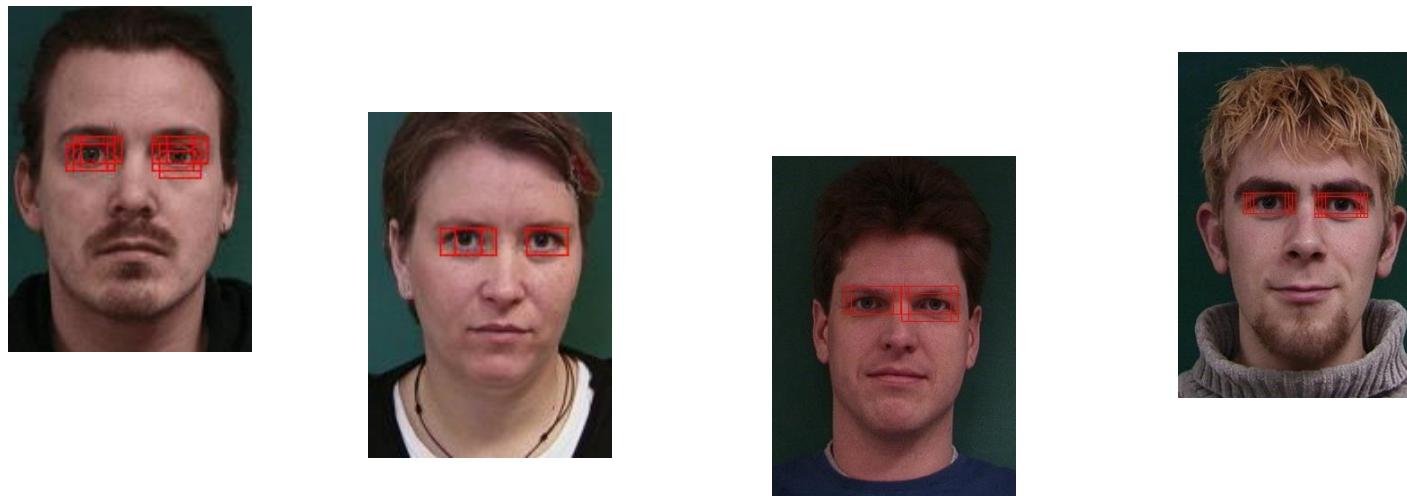


- Obiectul de recunoscut este o zonă de imagine (fereastră de dimensiuni cunoscute din faza de antrenare), descris prin coeficienții DCT (în cazul nostru) din fereastră scanați în ordine rând cu rând.
- În cazul procesării imaginilor JPEG direct în domeniul comprimat avem la dispoziție doar blocuri de  $8 \times 8$  - de multe ori un ochi ocupă o zonă din imagine mai mare decât atât.
- După o analiză a relațiilor dintre un bloc de coeficienți DCT și sub-blocurile sale, s-a putut implementa o aplicație care ne oferă o relație directă pentru formarea din mai multe blocuri de coeficienții DCT un singur bloc conținând coeficienții DCT, fără decompresia imaginilor și aplicarea transformatei DCT pe întregul bloc.

# Aplicarea mașinilor cu vectori suport în recunoașterea vizuală a obiectelor în domeniul comprimat



- pentru a reduce costul de procesare și de memorie
  - operațiile implicate în analiza imaginilor: extragerea trăsăturilor, selecția regiunii de interes, clasificarea și localizarea obiectelor de interes – realizate direct în domeniul comprimat
- Aplicație: localizare exactă a regiunii ochilor - sistemele de urmărire a traiectoriei privirii
- Rezultatele sunt usor superioare clasificarii în domeniul spatial



# Problema JPEG

Fie  $U_{dct}[8 \times 8]$  transformata DCT a blocului de  $8 \times 8$  pixeli curent de codat folosind standardul JPEG și  $Q[8 \times 8]$  matricea de cuantizare folosită, iar tabela de coduri pt. coeficientul de curent continuu cea din Fig. 1.

- a) Cum arată matricea coeficienților DCT cuantizați (valorile rezultate în urma cuantizării se vor rotunji la cel mai apropiat întreg)?
- b) Reprezentați șirul coeficienților DCT cuantizați obținut în urma unei ordonări în zig-zag. Care este rolul ordonării în zig-zag?
- c) Să se codeze RLC șirul coeficienților DCT cuantizați și ordonați în zig-zag.
- d) Dacă valoarea cuantizată a coeficientului de c.c. (coeficientul DC) din blocul anterior codat este -16, care este valoarea care va fi codată entropic pentru coeficientul de c.c. din blocul curent?
- e) Care este codarea utilizată în standardul JPEG pentru codarea entropică a datelor? Care este codul coeficientului de c.c. din blocul curent codat entropic?
- f) Care este forma analitică a operației de negativare a imaginilor direct în domeniul comprimat? Aplicați operația de negativare pe matricea coeficienților DCT cuantizați obținută la punctul (a).

-227,00	40,71	66,32	-0,21	5,75	-0,60	2,02	-0,94
-164,88	64,14	70,93	6,57	5,45	-1,96	-1,59	1,07
-37,63	20,70	19,44	4,69	3,06	3,36	0,91	2,22
4,28	-2,34	-1,03	-0,38	-3,50	-3,65	-6,25	2,13
2,25	-0,11	-5,07	-5,63	-4,50	-2,16	-1,15	-0,11
8,49	-7,23	-0,67	-7,00	-1,88	-0,44	4,34	-0,62
9,09	2,35	1,91	-4,28	-0,46	-0,79	1,06	1,51
6,02	3,72	2,34	-1,27	-3,17	2,18	1,18	-1,32

$U_{dct} =$

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

$Q =$

Categorie	Codul de bază	Lungimea	Categorie	Codul de bază	Lungimea
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

*round*

$$U_{dct\_Q}(i,j) = \lfloor U_{dct}(i,j) / Q(i,j) \rfloor, \text{ unde } i, j = \{0 \dots 7\}$$

$U_{dct}$  =

-227,00	40,71	66,32	-0,21	5,75	-0,60	2,02	-0,94
-164,88	64,14	70,93	6,57	5,45	-1,96	-1,59	1,07
-37,63	20,70	19,44	4,69	3,06	3,36	0,91	2,22
4,28	-2,34	-1,03	-0,38	-3,50	-3,65	-6,25	2,13
2,25	-0,11	-5,07	-5,63	-4,50	-2,16	-1,15	-0,11
8,49	-7,23	-0,67	-7,00	-1,88	-0,44	4,34	-0,62
9,09	2,35	1,91	-4,28	-0,46	-0,79	1,06	1,51
6,02	3,72	2,34	-1,27	-3,17	2,18	1,18	-1,32

$Q =$

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

a)

$$U_{dct\_Q}(0,0) = \left\lfloor \frac{-227}{16} \right\rfloor = -14$$

a)

-14	4	2	0	0	0	0	0
-14	5	1	0	0	0	0	0
-3	2	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

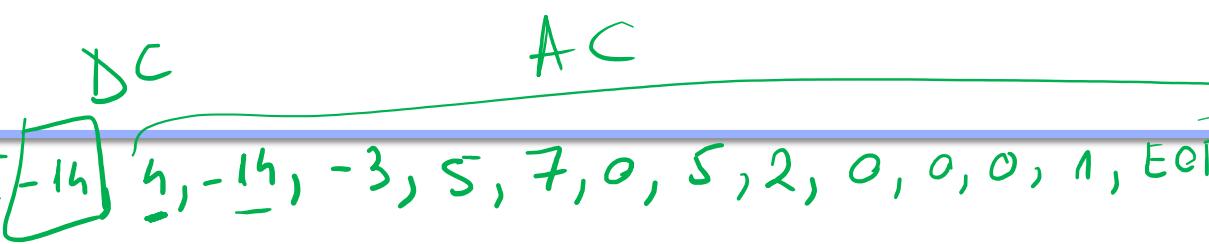
Concentrate values  
in storage bins

b)  $2D \rightarrow 1D$   
Scans in zig-zag

[14, 4, -14, -3, 5, 7, 0,  
5, 2, 0, 0, 0, 1, 0, 0, ... ]  
EOB

{ Coeff DC  $\rightarrow$  DPCM }  
{ Coeff AC  $\rightarrow$  RLE }

f)  $U_{dct\_Q,N} = (-1) \cdot U_{dct,Q}$   
 $\rightarrow$  facilitates calculations

c) Codice RLE 

$-14, (0, 4), (0, -14), (0, -3), (0, 5), (0, 7), (1, 5), (0, 2), (3, 1)$  (0, 0)

$\hookrightarrow$  codice Huffman

$\hookrightarrow$  DC  $\rightarrow$  codice dif. -DPCM

d)  $DC_{i-1} = -16 \quad ; \quad DC_i = -14$

$$e = DC_i - DC_{i-1} = -14 - (-16) = 2$$

$\hookrightarrow$  codice Huffman

e) Codice Huffman cof. DC [cod. loss - ampl. val]

$$2_4 = \boxed{10010}$$

6. Transformata DCT a blocului curent de codat în standardul JPEG este cea din Fig. 6.a. Matricea de cuantizare folosită este cea din Fig. 6.b, iar tabela de coduri pt. coeficientul de current continuu – în Fig. 6.c.

- Cum arată sirul coeficienților DCT cuantizați și ordonați în zig-zag? Care este rolul ordonării în zig-zag?
- Dacă valoarea cuantizată a coeficientului de c.c. din blocul anterior codat este 16, care este codul coeficientului de c.c. din blocul curent?

200	69	-50	-24	0	16	21	14
147	-63	-45	-22	0	15	19	12
0	0	0	0	0	0	0	0
-52	22	16	8	0	-5	-7	-4
0	0	0	0	0	0	0	0
34	-15	-11	5	0	3	4	3
0	4	9	0	0	0	0	0
-29	12	-3	0	0	-2	0	0

Fig. 6.a

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

Fig. 6.b

Categorie	Codul de bază	Lungimea	Categorie	Codul de bază	Lungimea
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

Fig. 6.c

# **SACCDMM - Curs 08**

## **Standardul JPEG 2000**

# Subband coding: motivation

- Coding with block-wise transform introduces visible blocking artifacts, as bit-rate decreases.
- Can we, somehow, overlap adjacent blocks,
  - thereby smoothing block boundaries,
  - but without increasing the number of transform coefficients?
- Solution: subband transform.

# Subbands vs. block-wise transform

- Blockwise transforms are a special case of subband decompositions with:
  - Number of bands  $m = \text{order of transform } N$
  - Length of impulse responses of analysis/synthesis filters  $\leq m$
- Filters used in subband coders are **not** in general orthogonal.
- Linear phase is desirable for images.

# Subbands vs. block-wise transform (cont.)

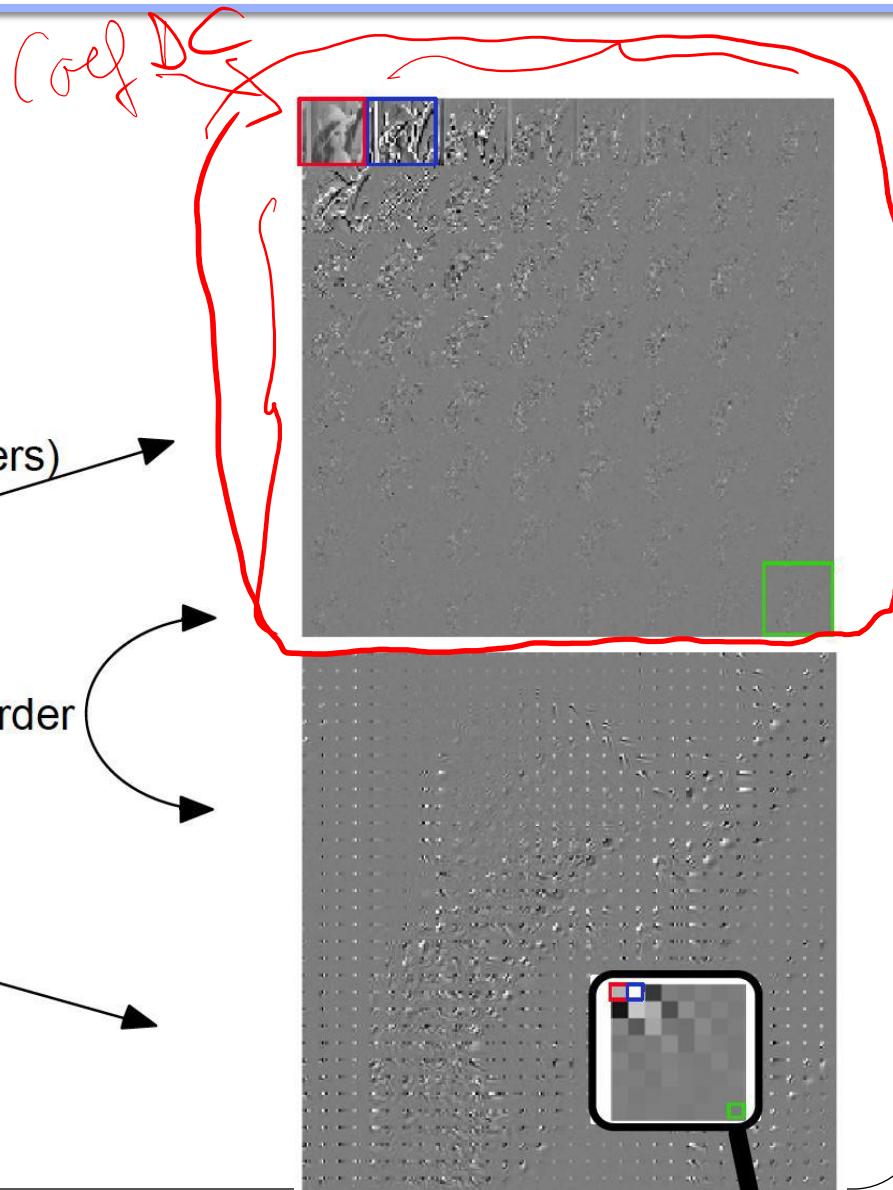
Original image



8-channel  
subband  
decomposition  
(using DCT filters)

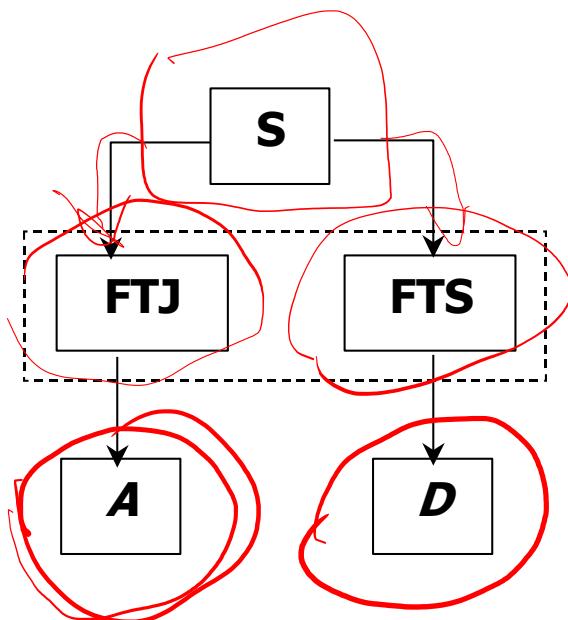
8x8 DCT  
64

re-order



# Transformata Wavelet discrete (DWT - Discrete Wavelet Transform)

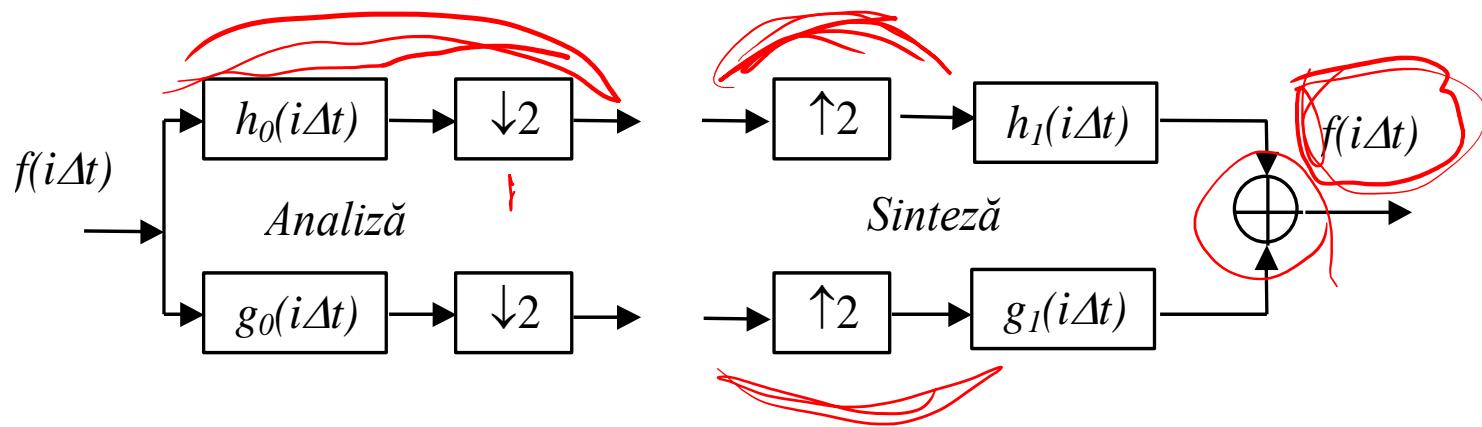
- calculul coeficientilor Wavelet pentru orice scala -> costisitor
- aplica DWT – set de scale si pozitii
- metoda diadica  $\Leftrightarrow$  factor de scala si pozitie = putere a lui 2
  - permite realizarea unei analize mult mai eficiente și mai corecte
- putem implementa DWT folosind filtrele
  - prin descompunerea in subbenzi folosind două canale
    - conținutul de joasă frecvență – definește „identitatea” semnalului.
    - conținutul de frecvență înaltă aduce îmbunătățiri în descrierea semnalului.
  - Ex. - vocea umană
    - eliminăm frecvențele înalte, aceasta se va auzi diferit dar va fi inteligibilă
    - eliminăm frecvențele joase, vocea este posibil să fie distorsionată.



- reprezentarea cu filtre a DWT
  - avem două componente:
    - Aproximările (A) – sunt componente cu factor de scală mare, adică componente de frecvență joasă
    - Detalii (D) – componente cu factor de scală mic, adică componente de frecvență înaltă

# DWT

- semnal original trecut prin DWT (trecut prin două filtre complementare)
  - => 2 ori numarul de esantioane (două componente: A – aproximări, D – detalii)
  - trebuie facuta o subesantionare => la ieșirea bancului de filtre se obtine aceeași cantitate de informație ca la intrare
- la reconstructie se aplica operatiunea inversă



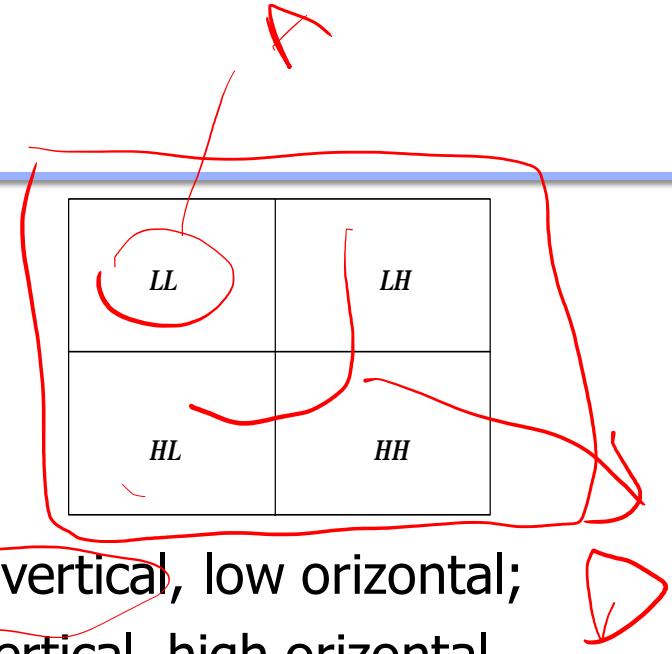
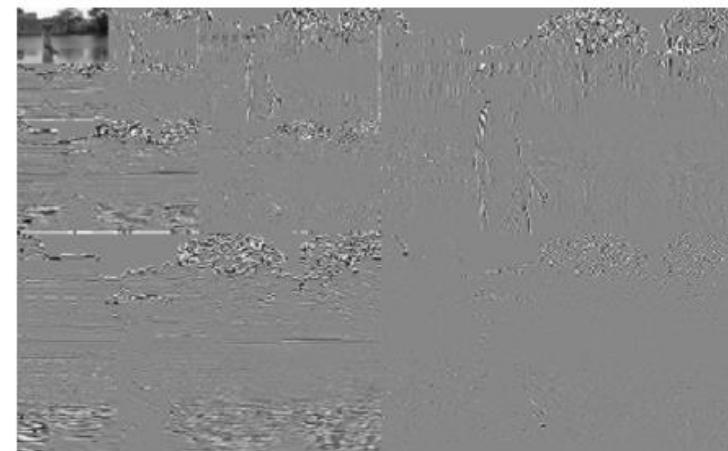
# DWT bidimensională

- Aplicarea transformatei wavelet unidimensionale discrete de două ori:
  - pe linii
  - pe coloane

LL – informatii de joasa frecventa; HL – high vertical, low orizontal;

LH – low vertical, high ~~orizonttal~~; HH – high vertical, high orizontal

1	2		5						
3	4								
		6	7						
				8					
					9				10



a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

pair(1)+pair(2) pair(1)-pair(2)

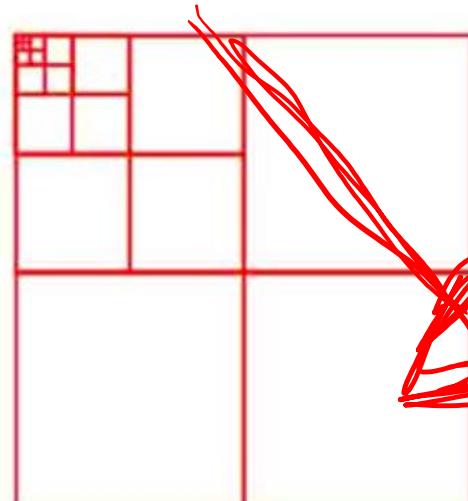
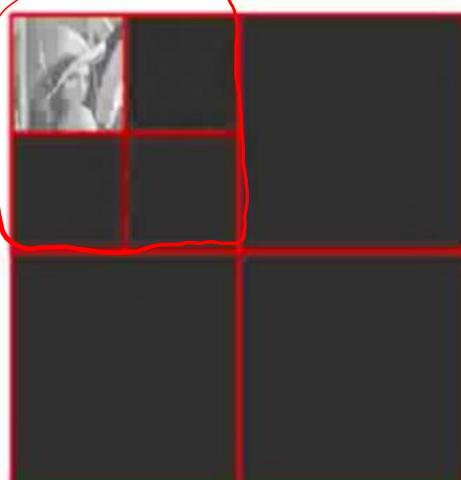
a+b	c+d	a-b	c-d
e+f	g+h	e-f	g-h
i+j	k+l	i-j	k-l
m+n	o+p	m-n	o-p

pair(1)+pair(2)  
pair(1)-pair(2)

x	x	x	x
o	o	o	o

LL HL  
LH HH

LL	HL
LH	HH



# JPEG 2000

- Noiembrie 1997 – evaluare a 20 algoritmi
- alegerea transformatei Wavelet – transformare de baza
- JPEG 2000 – 6 parti
  - partea 1 – modul de baza
    - complexitate minima
    - acopera 80% din aplicatii
    - defineste un format de fisier
    - devine standard international in Decembrie 2000
  - partea 2-6 –
    - extensii de algoritm
    - extensii de format de fisier
    - diferite stadii de dezvoltare

# JPEG 2000 “parts”

- partea 2
  - imbunatatirea performantelor
  - complexitate mare
  - IPR (intellectual property rights)
- partea 3
  - Motion JPEG 2000 - MJP2
- partea 4
  - testare
- partea 5
  - implementare software
    - implementare JAVA – grup JJ2000 (format din Canon, EPFL, Ericsson)
    - implementare C – Image Power, University of British Columbia
- partea 6
  - fisier complex – aplicatii de scanare de documente, fax

# JPEG 2000 – de ce?

- nu numai pentru a imbunatati compresia
- o noua reprezentare a imaginii
- extinderea domeniului de aplicabilitate
- facilitati:
  - imbunatatirea eficientei codarii
  - compresie ~~cu~~ fara pierderi
  - reprezentari la ~~rezolutii~~ multiple
  - sisteme de tip “embedded bit-stream”
    - decodare progresiva
    - scalabilitate SNR
  - impartirea in blocuri (tile component - tiling)
  - codarea ROI
  - rezistenta la erori
  - accesul si prelucrarea aleatoare a sirului codat
  - fisier mai flexibil

aplică pe img.  
într-o sing.

JPEG  $\rightarrow$  JPEG 2000

# JPEG 2000 – Cum?

- DCT -> DWT
  - compactare energetica mai buna
  - decorelare
  - reprezentare imaginii la mai multe rezolutii
  - compresie cu fara pierderi – acelasi sir codat
- codorul Huffman - > codare aritmetica adaptiva (codorul MQ)
- planurile de biti sunt codate independent (blocurile de codare)
- introducerea sistemului de coordonate – faciliteaza:
  - operatii de rotatii
  - inversare

# Pașii codării JPEG 2000 :

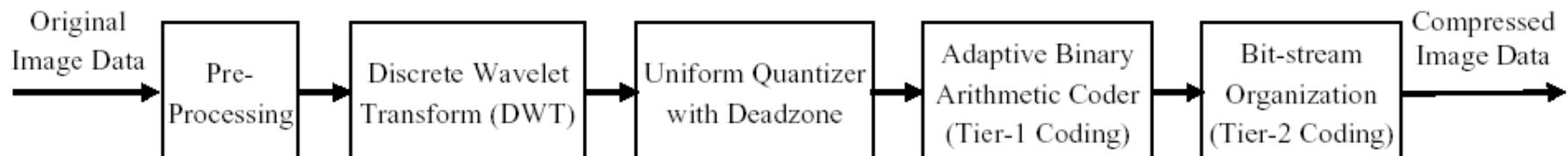
- descompunerea imaginii din spatiul de reprezentare RGB într-un alt spatiu de reprezentare care realizeaza o decorelatie intre componenta de luminanta si cele de crominanta (ex. formatele YUV, YC<sub>r</sub>C<sub>b</sub>, etc.)
- Componentele pentru reprezentarea imaginii sunt împărțite în blocuri de imagine elementare (tiling) - acest proces nu este obligatoriu
- se aplică transformata pe fiecare bloc elementar de imagine - transformata folosită în standardul JPEG2000 este transformata Wavelet discretă 2D
- subbenzile de coeficienți sunt cuantizate și grupate în „blocuri de codare”
- planurile de biți ale coeficienților dintr-un bloc de codare sunt codate entropic
- codarea poate fi implementată astfel încât să luăm în considerare ROI
- adăugarea elementelor de marcare suplimentare pentru reducerea erorilor la transmisie și salvare
- obținerea sirului codat JPEG 2000

# Acești pași pot fi grupați în 3 etape mari:

- etapa de **pre-procesare**
- etapa de **procesare de bază**
- etapa de **generare a sirului codat**

## II. Arhitectura JPEG 2000

- pre-procesare
- transformata Wavelet
- cuantizorul uniform
- codorul aritmetic adaptiv – nivelul 1 de codare
- organizarea sirului codat – nivelul 2 de codare



# Pre-procesarea

- imagini - 3 (RGB, YC<sub>r</sub>C<sub>b</sub>) -> 2<sup>14</sup> (16384) componente
- valorile esantioanelor: intregi cu/fara semn - esantion reprezentat cu B biti
  - reprezentarea fara semn: (0, 2<sup>B-1</sup>) - sunt translatate simetric fata de zero (prin scaderea valorii 2<sup>B-1</sup>)  $\Rightarrow [-128, 127]$
  - reprezentarea cu semn: (-2<sup>B-1</sup>, 2<sup>B-1</sup>-1) - nu sunt translatate
- impartirea imaginii in zone drept. de dim. egale (exceptie marginile)
  - dimensiunea zonelor – arbitrara (poate fi chiar toata imaginea)
  - fiecare zona este comprimata independent de celelalte (putand folosi un setul propriu de parametrii)  $\rightarrow$
  - avantaj – la codare cand memoria disponibila este redusa

# Pre-procesarea

- valoare esantion (fara semn) –  $2^{B-1}$  (simetrie fata de zero)
- esantioanele cu semn nu sunt “deplasate”
- transformarea spatiului de culoare
  - ICT – irreversible color transform RGB – ~~YC<sub>b</sub>C<sub>r</sub>~~ (lossy compression)

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.500 \\ 0.500 & -0.41869 & -0.08131 \end{pmatrix} \times \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.0 & 0 & 1.402 \\ 1.0 & -0.34413 & 0.71414 \\ 1.0 & 1.772 & 0 \end{pmatrix} \times \begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix}$$

- RCT – reversible color transform RGB (lossless and lossy coding)

$$Y = \left[ \frac{R + 2G + B}{4} \right], \quad U = R - G, \\ V = B - G,$$

$$G = Y - \left[ \frac{U + V}{4} \right], \quad R = \overbrace{U + G}, \quad B = \overbrace{V + G}.$$

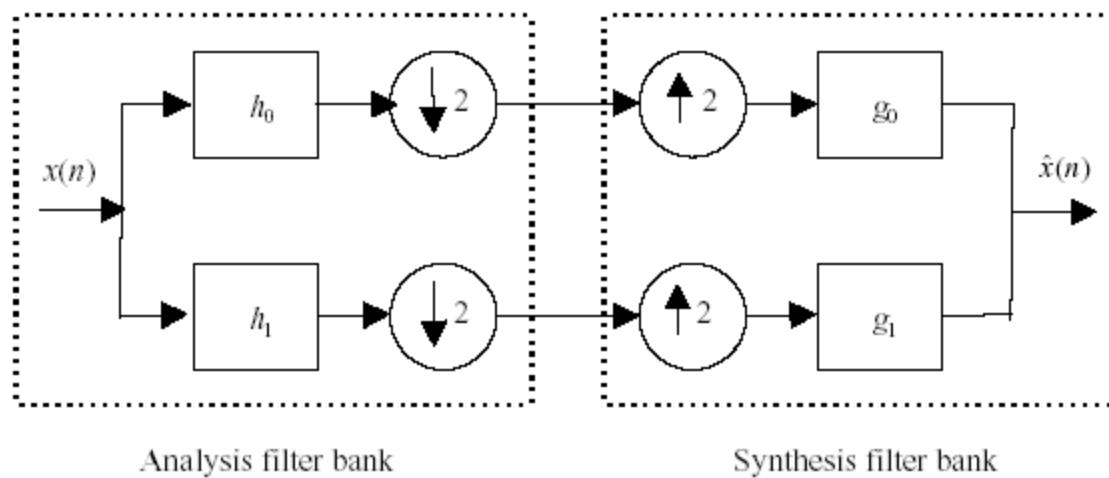
- Dacă pentru standardul JPEG se procedă la o operație de subeșantionare a componentelor de crominanță  $C_r$ ,  $C_b$  acest lucru nu este recomandabil pentru JPEG2000.
- Practic în JPEG2000 se realizează un prim pas al DWT pe componentele de crominanță.  
• Componentele LH, HL și HH se elimină și practic se obține o subeșantionare cu 2 atât în plan vertical cât și în plan orizontal.

# Transformata Wavelet Discreta (DWT)

- DCT pe blocuri -> DWT pe fiecare “tile” (sau pe intreaga imagine)
- caracteristici:
  - reprezentarea multi-rezolutie
  - eliminarea artefactelor (efect blocking) la comp. mare
  - folosirea filtrelor DWT intregi -> putem implementa:
    - codarea cu pierderi
    - codarea fara pierderi

# 1D –DWT

- succesiune de perechi de filtre FTJ, FTS + subiesantionare cu 2



- perechea FTS, FTJ – analiza bancurilor de filtre
- FTJ – pastreaza frecvențele joase (imagine incetosată)
- FTS – pastreaza frecvențele înalte (contururi, texturi, detalii etc.)

## 1D-DWT multiplu

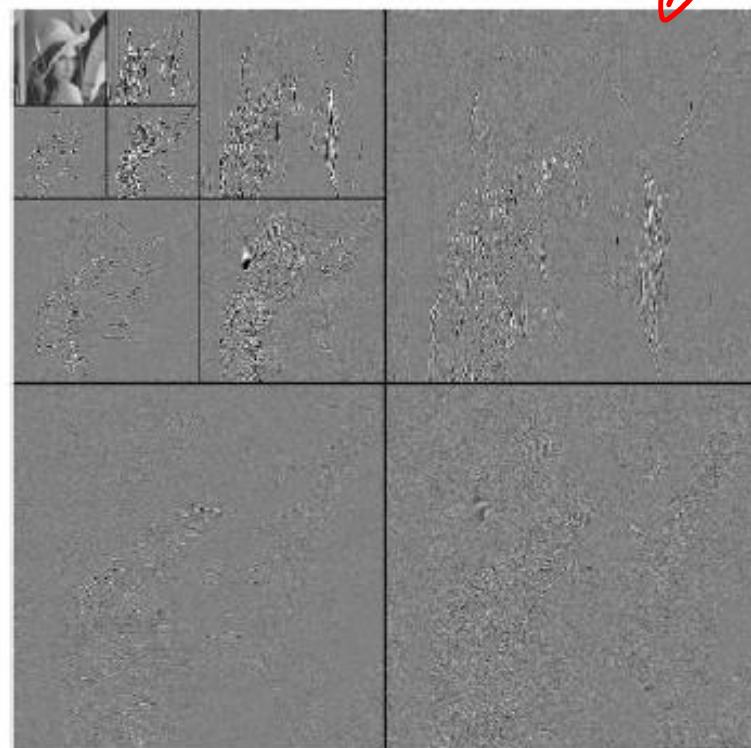
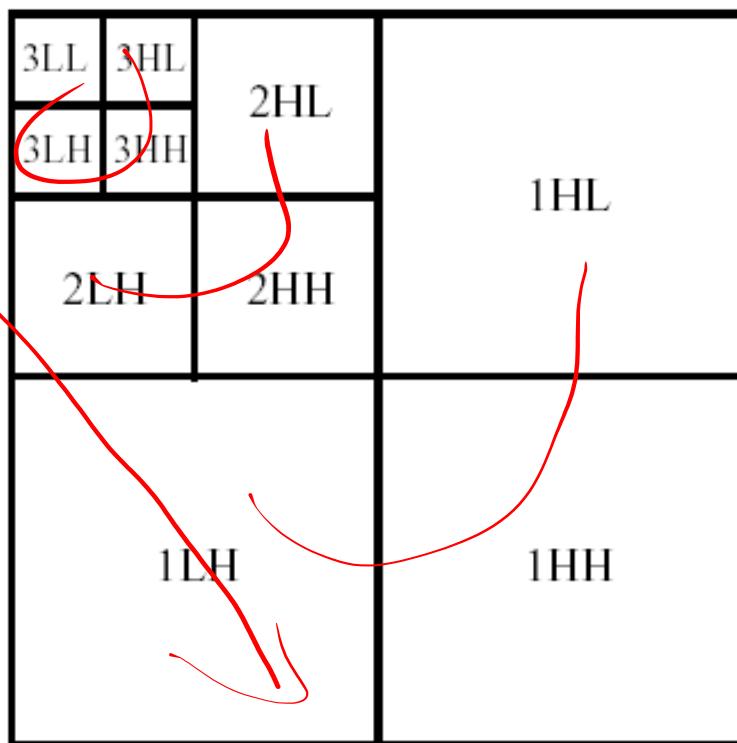
- primul pas DWT – secventa *low-pass* este inca corelata
- se aplica DWT pe secventa *low-pass* - *dyadic*
- DWT pe secventa *high-pass* – nu aduce castig!
- JPEG 2000 (part 1) – descompunerea dyadic
- JPEG 2000 (part 2) – + descompunerea pe secventa high-pass

## DWT – 2D

- 1D – DWT pe randuri ( $h_0, h_1$ )
- 1D – DWT pe coloane ( $h_0, h_1$ )
- dupa 1 pas 2D – DWT -> 4 subbenzi
  - subbanda LL
  - subbanda HL (H – pe orizontala; L verticala)
  - subbanda LH (L – orizontala; H – verticala)
  - subbanda HH
- subbenzile AC – adauga 128 pentru vizualizare mai buna

# Exemplu 2D – DWT

- bancuri de filtre DWT



+128  
0

# Implementarea DWT

- buffer pentru intreaga imagine
- impartirea imaginii in blocuri (tiles) – reduce necesarul de memorie
- folosirea schemei de lift-are – *lifting scheme*
  - reducerea cantitatii de memorie
  - reducerea complexitatii de calcul
  - rapiditate de calcul
  - adoptata in JPEG 2000

# Schema de *lift-are*

- impartirea semnalului de intrare
  - sevenete de esantioane **pare**  $d_i^0$
  - sevenete de esantioane **impare**  $s_i^0$
- alternarea pasilor:
  - predictie
  - updatare

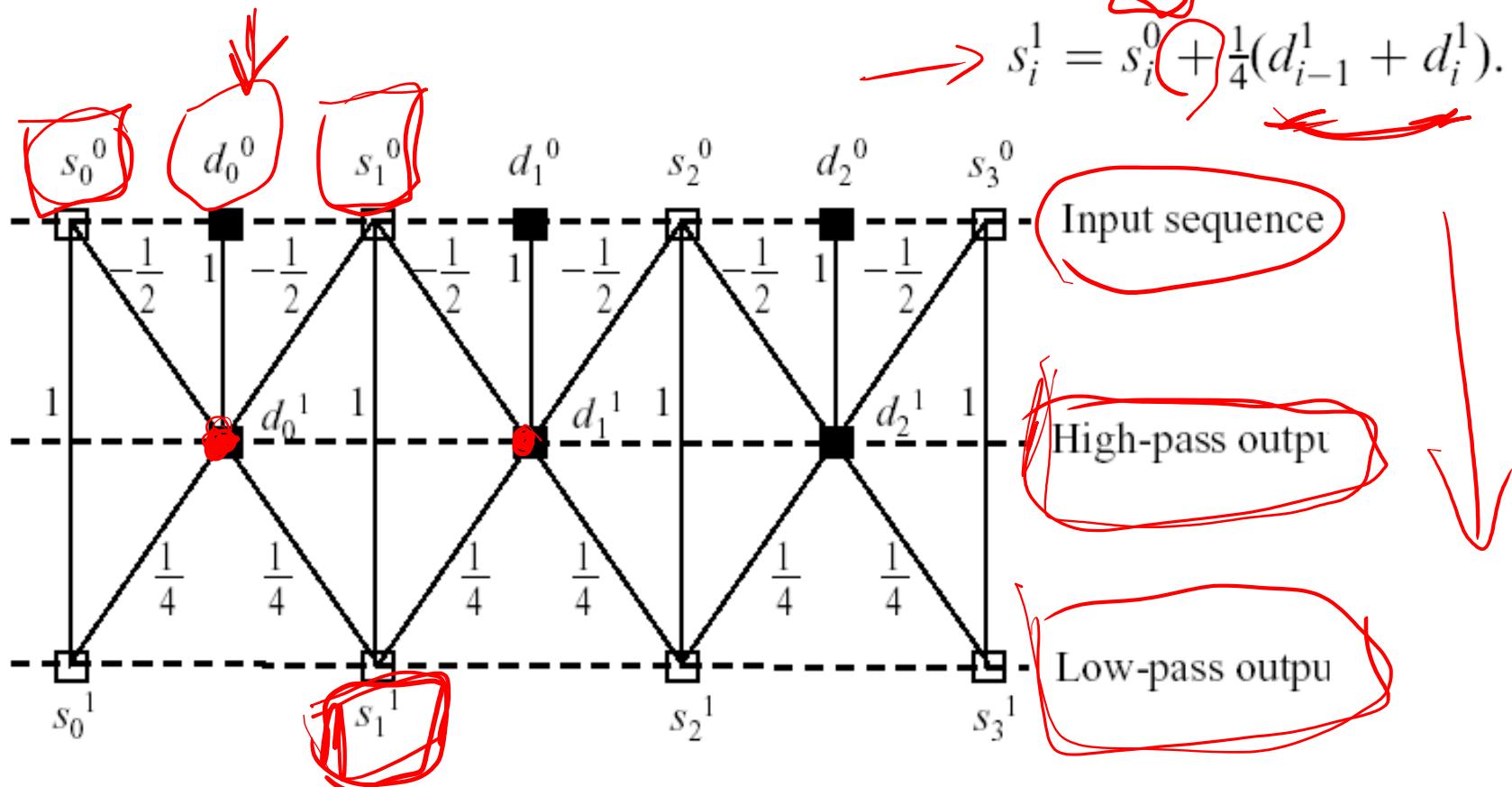
# Schema de liftare pentru filtru (5,3)

- predictia – fiecare esantion par se exprima
- update – fiecare esantion impar se exprima
- suficient un singur pas

$$I = I_{TJ} + I_{FS}$$

$$\rightarrow d_i^1 = d_i^0 - \frac{1}{2}(s_i^0 + s_{i+1}^0).$$

$$\rightarrow s_i^1 = s_i^0 + \frac{1}{4}(d_{i-1}^1 + d_i^1).$$



# Cuantizarea

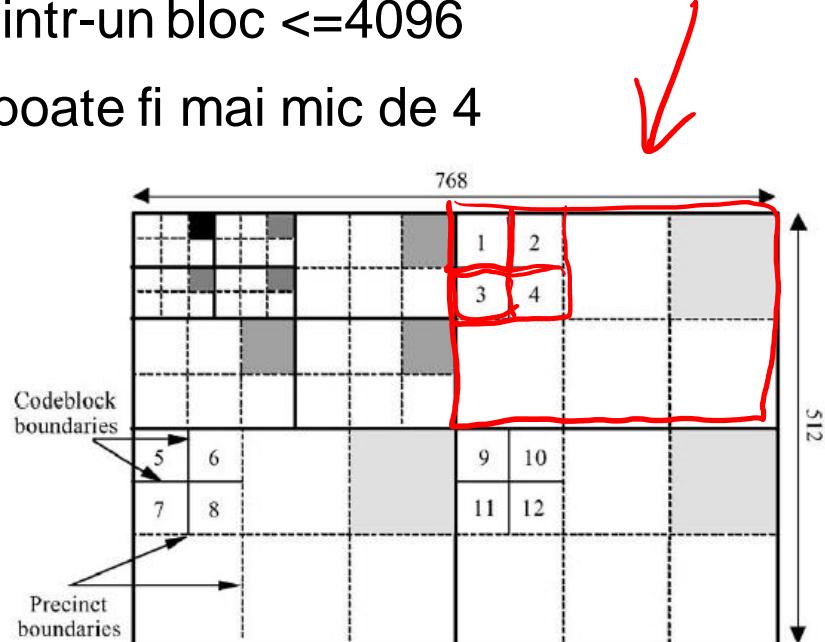
- Se folosește o cuantizare uniformă pentru fiecare bandă de coeficienți DWT.
- Această etapă induce pierderi în imaginea reconstruită.
- Fiecare coeficient  $a_b(u,v)$  al subbenzii  $b$  este cuantizat la valoarea  $q_b(u,v)$  conform formulei:

$$q_b(u,v) = \text{semn}[a_b(u,v)] \cdot \left\lfloor \frac{|a_b(u,v)|}{\Delta b} \right\rfloor$$

unde  $\Delta b$ , reprezintă factorul de cuantizare pentru subbanda  $b$

# Codarea entropica

- fiecare subbanda este codata separat
- fiecare subbanda este impartita in *blocuri de codare*
- fiecare bloc de codare este codat independent
- dimensiunea blocului de codare:
  - intreg, putere a lui 2
  - numarul maxim de coeficienti dintr-un bloc  $\leq 4096$
  - nivelul blocurilor de codare nu poate fi mai mic de 4



# Avantaje ale codarii independente

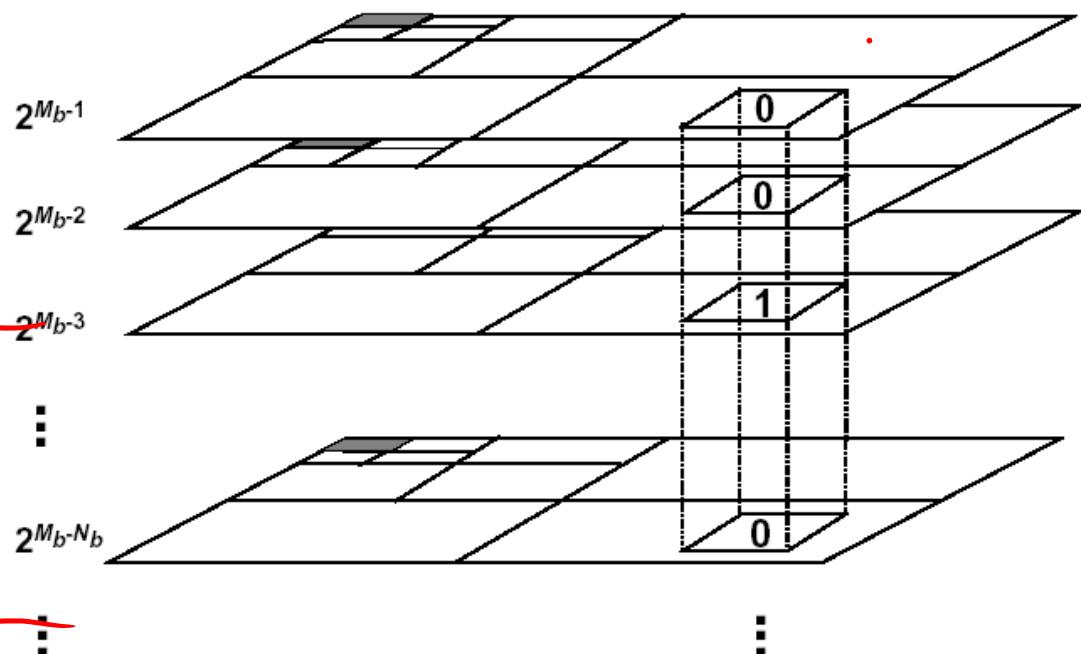
- accesul aleator la imagine
- implementarea pe arhitecturi paralele
- functionalitati de prelucrare a imaginii
- rezistenta la erori
- eficienta controlului ratei de bit
- flexibilitate maxima pentru aranjarea ordinii de aparitie

# Codarea pe plane de biti

- coeficienti
  - nesemnificativi – bitul este zero
  - semnificativi – bitul este 1 – incepe codarea coeficientului

Diagram illustrating the representation of coefficients on bit planes:

0	0	0	0	0	0	0	-0
0	0	0	0	0	0	1	-1
0	0	0	0	0	0	1	-2
0	0	0	0	0	0	1	-3
⋮							⋮
1	0	0	0	0	0	0	-128
⋮							⋮
i	i	i	i	i	i	i	-255



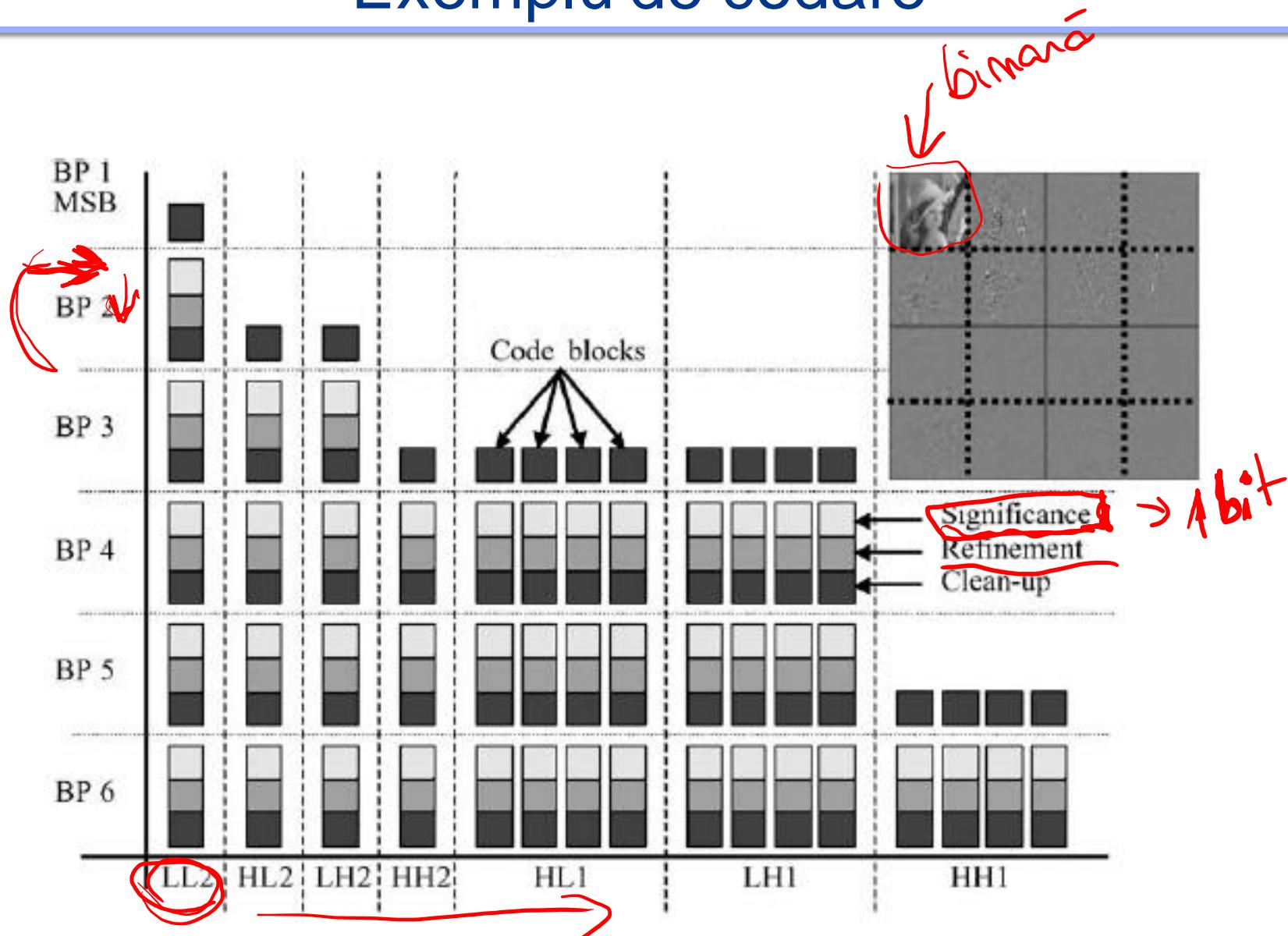
# Codarea aritmetica si MQ

- codeaza o intreaga secventa de simboluri
- cuvantul de cod – prin impartirea recursiva a  $(0,1)$  dupa probabilitati
- codare adaptiva

# Etapele codarii pe planuri de biti

- fiecare plan de biti este codat independent folosind 3 etape (sub-bitplanes) cu “intreruperea” sirului codat dupa fiecare etapa
- avantaje:
  - pentru optimizarea afisarii imaginilor la diferite rezolutii
  - minimizarea distorsiunilor care pot aparea in sirul codat
- pasul I – “significance propagation”
  - codarea coeficientilor care au probabilitatea sa devina semnificativi
- pasul II – “refinement”
  - imbunatatirea codarii coeficientilor semnificativi
- pasul III – “cleanup”
  - codarea restului coeficientilor din plan care nu vor deveni semnificativi

# Exemplu de codare



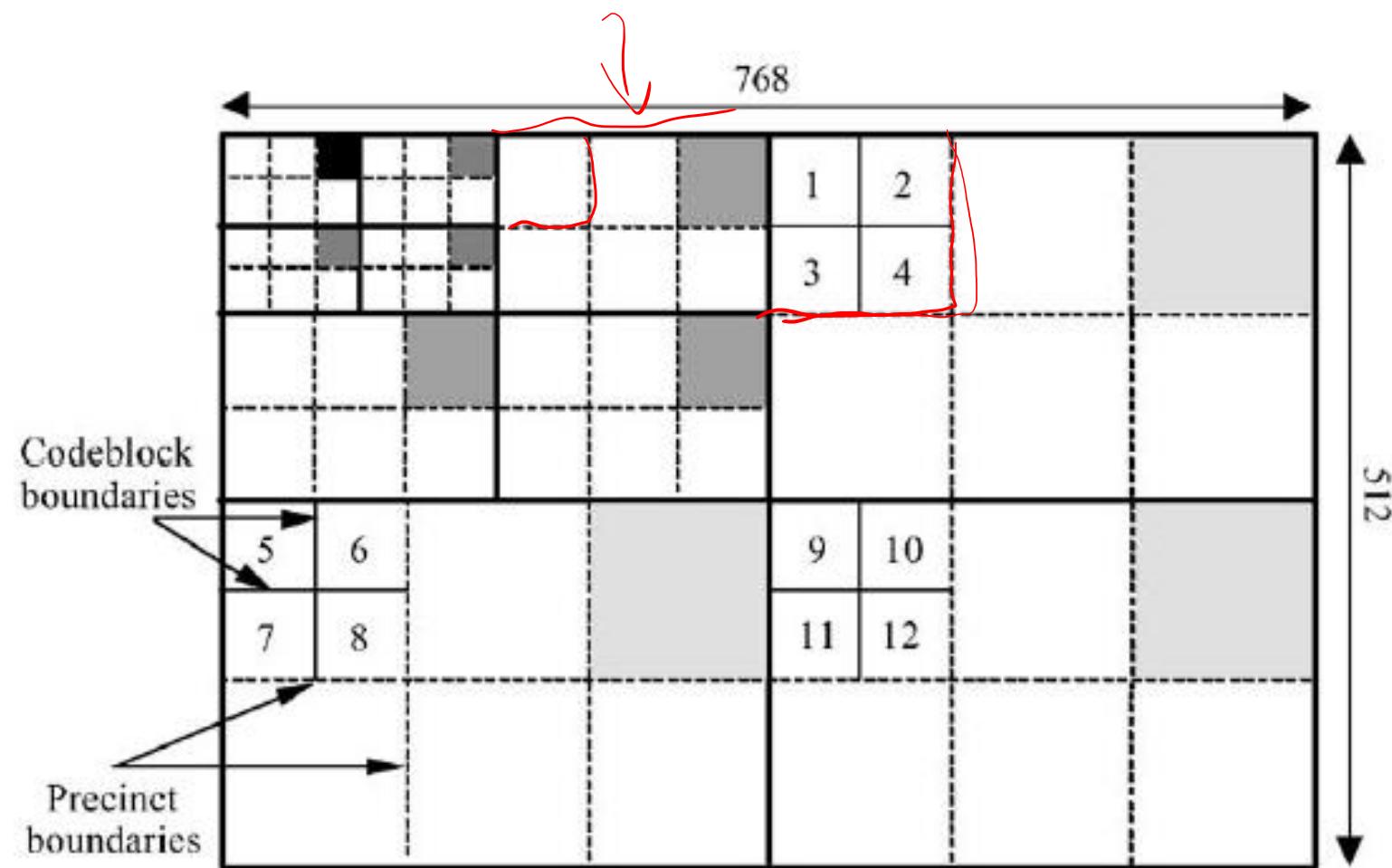
# Organizarea sirului codat in JPEG2000

- de ce?
  - acces aleator la imagine
  - ROI
  - scalabilitate
- impartirea imaginii
  - canale de culoare (componente)  $Y, G_b, C_n$  / YUV
  - regiuni spatiale (tiles)  $\rightarrow$  unul intregă imaginie
  - regiuni de frecvență (subbenzi ale DWT)
  - regiuni spatiu-frecvență (blocurile de codare)

# Alte structuri de date utilizate

- **precinct** – colectie de blocuri de codare, (aproximativ uniforme) din toate subbenzile cu acelasi nivel de descompunere DWT
- Fiecare precinct este împărțit în blocuri elementare de codare
- **pachete** – este un segment continuu din sirul codat care contine un numar din nivelele de codare pe planuri de biti pentru fiecare bloc de codare din precinct
- pachetele din fiecare precinct la toate rezolutiile sunt grupate in **layer-e**

# Exemplu de structuri “precinct” si blocuri de codare



# Ordinea de aparitie a pachetelor din sirul codat

- progression order
- pentru o componenta “tile” 4 componente pentru identificarea pachetului:
  - componenta
  - rezolutia
  - nivelul (layer)
  - pozitia (precinct)
- pachetele pentru o componenta specifica sunt generate prin scanarea precinct-ului intr-o ordine bine stabilită

# Algoritmi de citire ordonata

- Layer-resolution-component-position – LRCP
  - aplicatii de BD de imagini
- Resolution-Layer-Component-Position – RLCP
  - aplicatii client-server – clientii acceseaza imagini la diferite rezolutii
- Resolution-Position-Component-Layer – RPCL
  - aplicatii unde este necesara accesarea imaginilor la diferite rez. si poz.
- Position-Component-Resolution-Layer – PCRL
  - imbunatatirea calitatii imaginii pentru o pozitie specifica
- Component-Position-Resolution-Layer – CPRL
  - obtinerea celei mai bune calitati a unei imagini pentru o pozitie particulara

# Performantele diferitilor algoritmi JPEG 2000

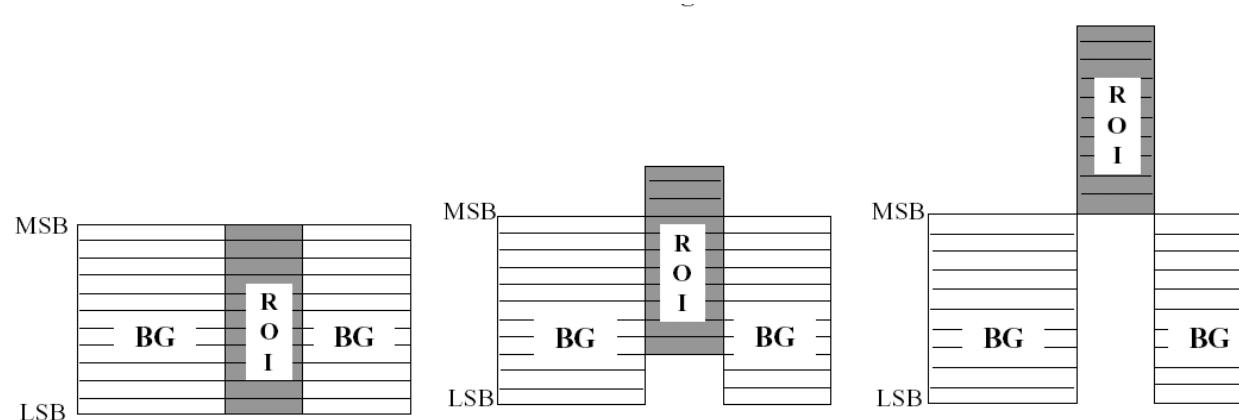
- **parametrii:**
  - eficiența codării
  - viteza
  - complexitatea implementării
- **compresia cu pierderi:**
  - dimensiunea componentei "tile"
  - dimensiunea blocului de codare
  - filtre DWT
  - nivelor de descompunere DWT
  - opțiuni de codare entropică
  - efectul ciclurilor multiple de codare
- **compresia fără pierderi:**
  - transformări de culoare reversibile
  - opțiuni ale codorului fără pierderi
  - opțiuni de codare entropică

## Imbunatatiri ale standardului JPEG 2000

- definirea ROI
- scalabilitatea spatiala si dupa RSZ
- imunitate la erori
- posibilitatea de protejare IPR

# ROI

- se pot defini diferite zone din imagine
- importanta mai mare sau mai mica a zonelor
- principiu:
  - translatarea coeficientilor din ROI
  - se codeaza mai intai planurile de biti MSB
  - apoi se codeaza planurile BG



# Scalabilitatea

- după RSZ
  - pentru aplicații unde este utilă accesul la imagini cu multiple nivele de calitate
  - aceeași rezoluție spatială la RSZ diferiti
- spatială
  - aplicații unde avem nevoie de imagini la rezoluții multiple dar la aceeași calitate
  - codare imagine de rezoluție mică + transmitere versiune de rezoluție superioară pentru imbunatatirea primei versiuni

## Imunitate la zgomot

- pentru aplicatii pe sisteme mobile si Internet
- doua nivele de imunitate la zgomot:
  - la nivelul codorului entropic
    - folosirea blocurilor de codare
    - folosirea terminatoarelor in procesul de codare aritmetica
  - la nivelul blocului de formare a sirului de biti
    - folosirea pachetelor scurte
    - folosirea marker-lor de resincronizare

# Comparatie JPEG <-> JPEG 2000

↗ factor de comp.  
mare

Imagine	Tip	Rezoluție	Observații	Compresie	RSZ(dB)
<b>Barb.bmp</b>	8biți/pixel	512x512	-		-
Barb_1_27_2000.jpc	8biți/pixel	512x512	1:27	JPEG2000	29.22
Barb_1_4_2000.jpc	8biți/pixel	512x512	1:4	JPEG2000	38.55
Barb_1_27.jpg	8biți/pixel	512x512	1:27	JPEG	23.28
Barb_1_4.jpg	8biți/pixel	512x512	1:4	JPEG	36.68
<b>Bogdan.bmp</b>	24biți/pixel	256x256	-		-
Bogdan_41_2000.jpc	24biți/pixel	256x256	1:41	JPEG2000	26.62
Bogdan_7_2000.jpc	24biți/pixel	256x256	1:7	JPEG2000	27.67
Bogdan_41.jpg	24biți/pixel	256x256	1:41	JPEG	23.05
Bogdan_7.jpg	24biți/pixel	256x256	1:7	JPEG	34.8*

## JPEG <-> JPEG 2000



barb.bmp



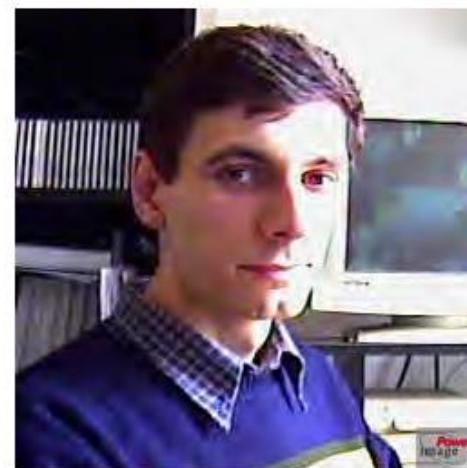
barb\_1\_27\_2000.jpgc



barb\_1\_27.jpg



bogdan.bmp

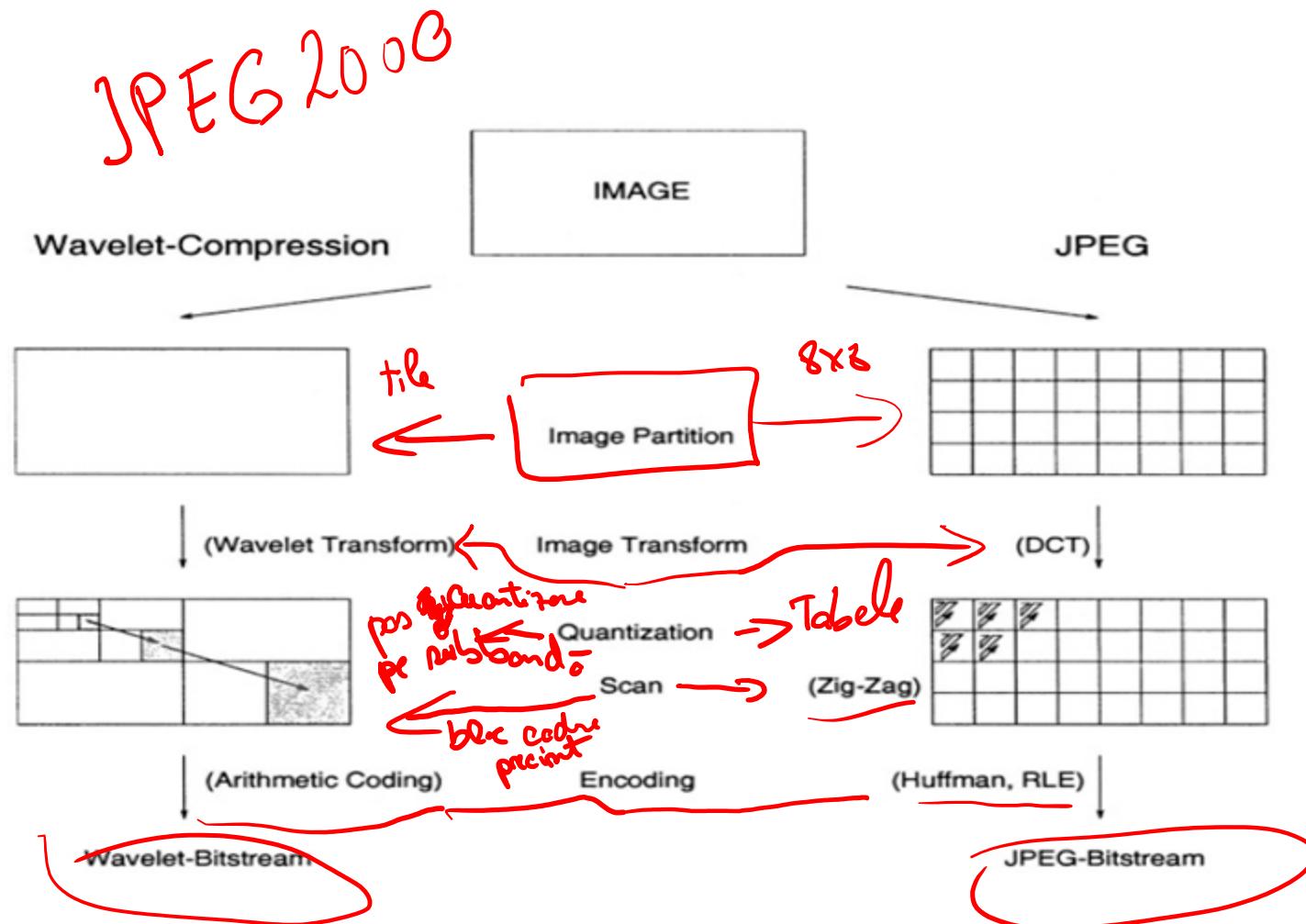


bogdan\_1\_41\_2000.jpgc



bogdan\_1\_41.jpg

# Comparație între schema de compresie bazată pe DCT și cea bazată pe wavelet



# SACCDMM - Curs 09

## Codarea intercadre; Estimarea mișcării intercadre

Sl.Dr.Ing. Camelia FLOREA

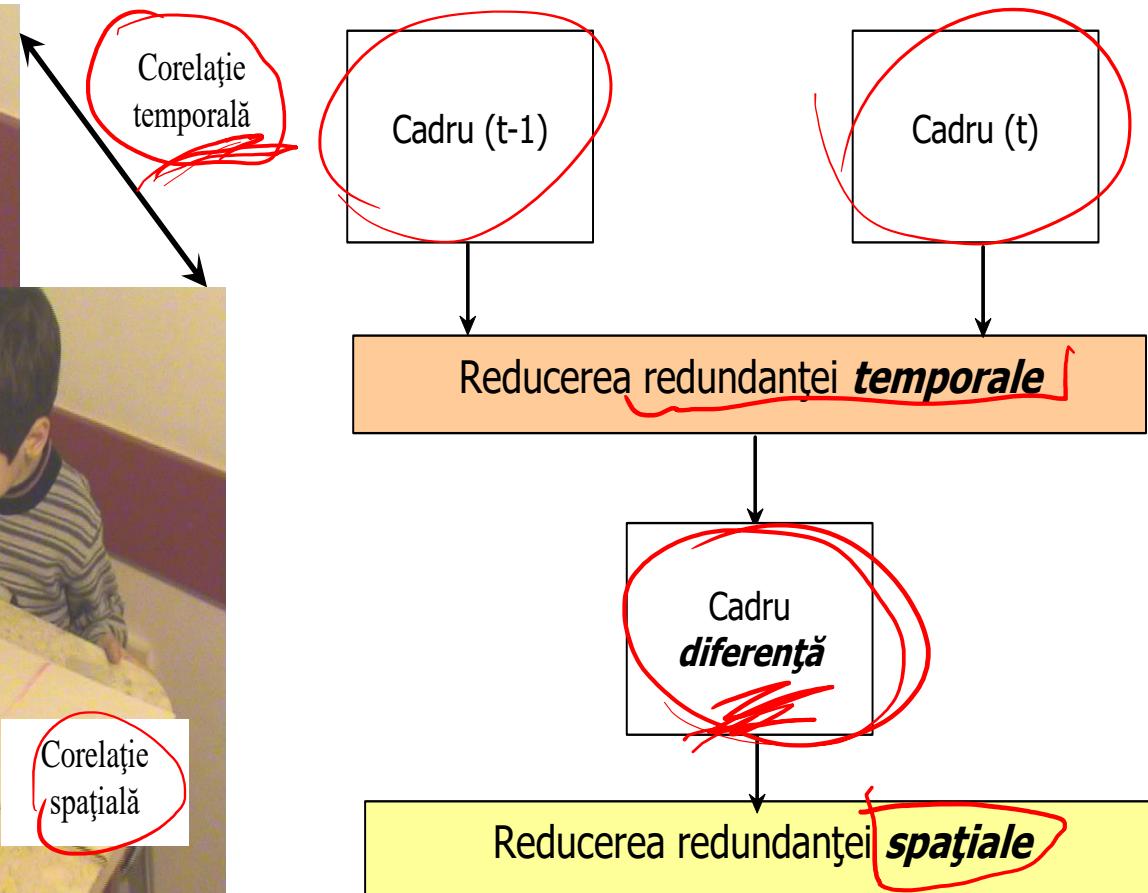
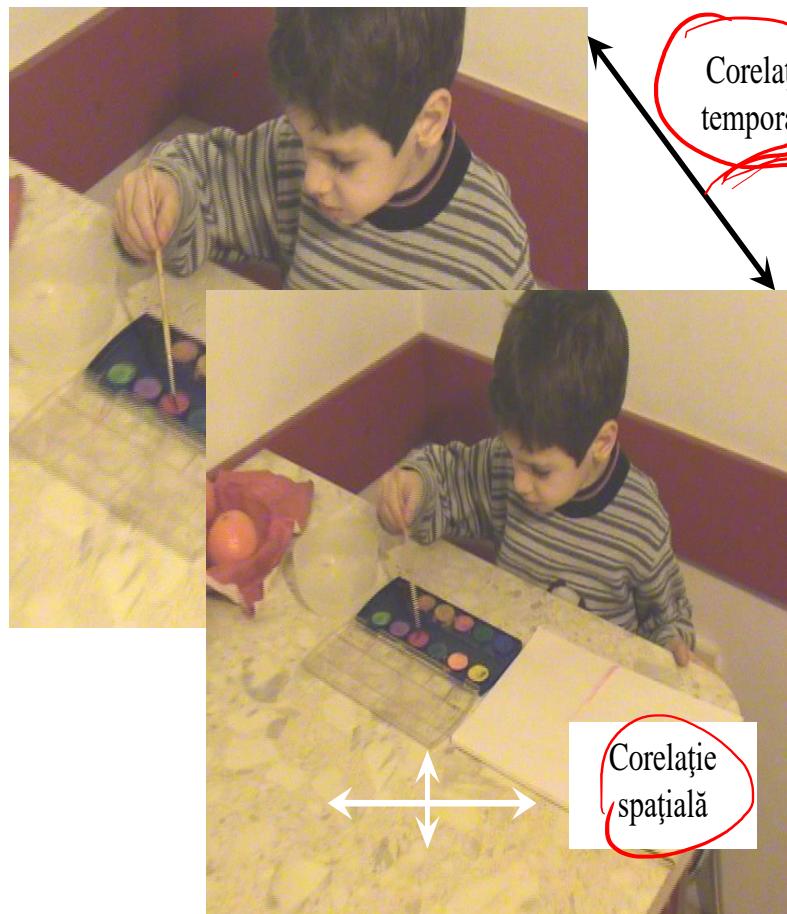
# Compresia secvențelor video

- Necesitatea măririi raportului de compresie pentru video
- Secvență:  $512 \times 515 \times 8 \text{ biți} \times 3 \text{ planex} 25 \text{ imagini/sec}$  rezultă  $150 \text{ Mbiți/sec}$ 
  - raport compresie imagini în general 12:1
  - rezultă  $12,5 \text{ Mbiți/sec}$  PREA MARE
  - este necesar un raport de compresie 100:1

=> NECESAR exploatarea ***codării intracadrul si intercadru***

# Corelația temporală și spațială

- 2 etape: redundanța intercadru, redundanța intracadru



# Reducerea redundanței

- Spațiale
  - Cele utilizate și la imaginile statice
- Temporale
  - exploatează redundanța dintre cadrele succesive
    - Se codează doar modificările – “diferențele”, datorate:
      - mișcării obiectelor în imagine
      - mișcării camerei
      - zoomingului, etc
    - urmărirea fiecărui pixel  
-> compromis eficiență-calcule -> Împărțirea în **macro-blocuri** 16x16 pixeli -> Urmărirea macroblocurilor de la un cadru la altul

# Segmentarea imaginii în zone staționare și “în mișcare”

- sistemului vizual uman este foarte slab în cazul imaginilor care conțin simultan frecvențe spațiale și temporale înalte.  
=> zonele cu modificări rapide dintr-o imagine pot fi reprezentate cu o amplitudine și o rezoluție spațială mai redusă în comparație cu zonele staționare.
- Acest lucru permite înlocuirea rezoluției spațiale cu rezoluția temporală și se folosește în scopul obținerii unor secvențe de imagini de calitate bună, la o rată de transmisie de  $2\div 2,5$  biți/pixel.
- O asemenea metodă presupune segmentarea imaginii în zone staționare și “în mișcare”.
- Segmentarea se poate realiza prin evaluarea semnalului-diferența între două cadre succesive.
  - În zonele staționare se transmite diferența între cadre pentru fiecare al doilea pixel, iar pixelii rămași sunt repetați din cadrul precedent.
  - În zonele “în mișcare” se folosește o subeșantionare orizontală de 2:1, elementele netransmise fiind reconstituite prin interpolare de-a lungul liniilor.
- Cele mai importante distorsiuni pot apărea la muchiile ascuțite ce se deplasează în imagine cu viteză moderată.

# Codarea “cu înlocuire condiționată”

- se bazează pe detecția și codarea zonelor în mișcare care se înlocuiesc de la un cadru la altul:

$$e(m,n,i) = u(m,n,i) - u \cdot (m,n,i-1)$$

- $u(m,n,i)$ , pixelul de coordonate  $(m,n)$  în cadrul I.
- $u \cdot (m,n,i-1)$  este valoarea reprodusă a pixelului de coordonate  $(m,n)$  din cadrul I-1.
- În cazul în care  $e(m,n,i)$  depășește o valoare de prag  $\eta$   
=> ea este cuantizată și codată pentru transmisie.

# Codarea “cu înlocuire condiționată”, la receptie:

- valoarea unui pixel se obține:
  - dacă acesta provine dintr-o zonă staționară
    - prin repetarea valorii pixelului din cadrul precedent,
  - dacă pixelul provine dintr-o zonă “în mișcare”
    - prin înlocuirea cu semnalul-diferență decodat

$$u(m,n,i) = \begin{cases} u^*(m,n,i-1) + e(m,n,i), & \text{daca } |e(m,n,i)| > \eta \\ u^*(m,n,i-1), & \text{in rest} \end{cases}$$

- Pentru a obține o rată de transmisie stabilă este necesară folosirea unui buffer de mărime rezonabilă și o strategie de control a bufferului adecvată.
- În aceste condiții se poate obține o rată de transmisie de 1 bit/pixel pentru nivele ale raportului semnal/zgomot mediu de 34 dB (39 dB în zonele staționare și 30 dB în zonele în mișcare).

# Problema 1:

- În figura următoare sunt prezentate: un bloc de 8x8 pixeli codat și reconstruit dintr-un cadru de referință, și respectiv blocul de pe aceeași poziție spațială de 8x8 pixeli din cadrul curent (următor cadrului de referință) care urmează a fi codat. Dacă asupra secvenței video se aplică o codare cu înlocuire condiționată cu pragul =11,
  - Cum arată matricea de eroare  $E[8x8]$  transmisă decodorului? Se va presupune o cuantizare fără pierderi a lui E.
  - Cum arată blocul de 8x8 pixeli din cadrul curent codat, reconstruit la decodor? Calculați MSE (eroarea medie pătratică) dintre blocul decodat (reconstruit) din cadrul curent și blocul original din cadrul curent.

Cadrul de referinta =	$\begin{bmatrix} 120 & 128 & 120 & 140 & 160 & 200 & 210 & 192 \\ 120 & 120 & 130 & 164 & 164 & 200 & 200 & 180 \\ 120 & 100 & 80 & 80 & 80 & 90 & 120 & 164 \\ 100 & 70 & 60 & 64 & 64 & 60 & 60 & 100 \\ 70 & 60 & 32 & 32 & 32 & 32 & 60 & 70 \\ 100 & 80 & 80 & 70 & 70 & 100 & 132 & 132 \\ 100 & 90 & 90 & 100 & 100 & 100 & 100 & 132 \\ 100 & 100 & 90 & 80 & 80 & 90 & 100 & 132 \end{bmatrix}$
-----------------------	--

Cadrul curent, de codat =	$\begin{bmatrix} 114 & 120 & 120 & 120 & 140 & 180 & 200 & 180 \\ 114 & 120 & 120 & 160 & 160 & 200 & 210 & 180 \\ 114 & 100 & 80 & 80 & 92 & 92 & 120 & 160 \\ 90 & 64 & 70 & 90 & 80 & 80 & 60 & 70 \\ 50 & 40 & 48 & 70 & 64 & 50 & 60 & 60 \\ 90 & 80 & 80 & 70 & 80 & 120 & 140 & 140 \\ 110 & 80 & 90 & 114 & 114 & 110 & 100 & 140 \\ 104 & 100 & 80 & 80 & 80 & 90 & 108 & 128 \end{bmatrix}$
---------------------------	---

120	128	120	140	160	200	210	192
120	120	130	164	164	200	200	180
120	100	80	80	80	90	120	164
100	70	60	64	64	60	60	100
70	60	32	32	32	32	60	70
100	80	80	70	70	100	132	132
100	90	90	100	100	100	100	132
100	100	90	80	80	90	100	132

Cadrul de referinta =

114	120	120	120	140	180	200	180
114	120	120	160	160	200	210	180
114	100	80	80	92	92	120	160
90	64	70	90	80	80	60	70
50	40	48	70	64	50	60	60
90	80	80	70	80	120	140	140
110	80	90	114	114	110	100	140
104	100	80	80	80	90	108	128

Cadrul curent, de codat =

$$e = \text{Cadrul Curent} - \text{Cadrul Referinta}$$

$m_e = 11$

$$e_{+}(m, m) = \begin{cases} e(m, m), & \text{dacă } e(m, m) > 0 \\ 0, & \text{altele} \end{cases}$$

-6	8	0	-20	-20	-20	-10	-12
-6	0	-10	-6	-6	0	10	0
-6	0	0	0	12	0	0	-6
-6	0	0	0	0	0	0	-6
-10	-6	14	26	16	20	0	-30
-20	-20	16	38	32	18	0	-10
-10	0	0	0	16	20	8	8
-10	0	0	0	0	16	10	0
10	-10	0	14	14	10	0	8
5	0	-10	0	0	0	8	-6

$E =$

$E_t =$

0	0	0	-20	-20	-20	0	-12
0	0	0	0	0	0	0	0
0	0	0	0	12	0	0	0
0	0	0	26	16	20	0	-30
-20	-20	16	38	32	18	0	0
0	0	0	0	0	20	0	0
0	0	0	14	14	0	0	0
0	0	0	0	0	0	0	0

$$MSE = \frac{1}{64} ((-6)^2 \cdot 4 + 8^2 \cdot 5 + 12 \cdot 10^2 + 4^2 \cdot 5 + 2^2) = 27,31$$

$$\text{Cadrul reconstituit} = \text{Cadrul Referintă} + E_t$$

120	128	120	120	140	180	210	180
120	120	130	160	160	200	200	180
120	100	80	80	68	90	120	160
100	70	60	38	48	40	60	70
50	40	48	70	64	50	60	70
100	80	80	70	70	120	132	132
160	90	90	114	114	100	100	132
100	100	90	80	80	90	100	132

$$m = ?$$

$$E_t = 0$$

$$m = 39$$

$\Rightarrow \text{MSE - Min}$

$$\text{MSE} = \frac{\sum_{i=1}^8 (C_{\text{rec.}} - C_{\text{real codat}})^2}{(8 \times 8)} \quad m = ?$$

$$\text{MSE} = 0$$

$$\Rightarrow m = 1$$

$$E_t = E$$

# Problema 2:

- În figura urmatoare sunt reprezentate cadrele 10 și 11 dintr-o secvență video. Ce valoare trebuie să aibă valoarea de prag de codare a erorii,  $\eta$ , pentru a coda cadrul 11 fără a transmite nici un semnal de eroare, folosind tehnica de codare intercadre cu înlocuire condiționată, considerând ca referință cadrul 10? Calculați în acest caz MSE la redarea cadrului 11.

12	8	7	9	5	6	6	3
4	5	3	0	2	1	4	5
1	2	3	5	3	2	3	2
4	0	0	0	2	3	0	4
8	0	2	2	2	3	1	2
4	2	3	4	5	6	3	2
1	7	4	8	9	6	3	9
9	8	7	6	8	9	8	9

Cadrul 10

12	8	7	9	5	6	6	3
4	5	3	0	2	1	4	5
1	3	0	0	2	2	3	2
4	3	2	3	5	3	2	4
8	4	0	0	0	2	3	2
4	3	0	2	2	2	3	2
1	7	2	3	4	5	6	9
9	8	7	6	8	9	8	9

Cadrul 11

# Codarea predictivă adaptivă

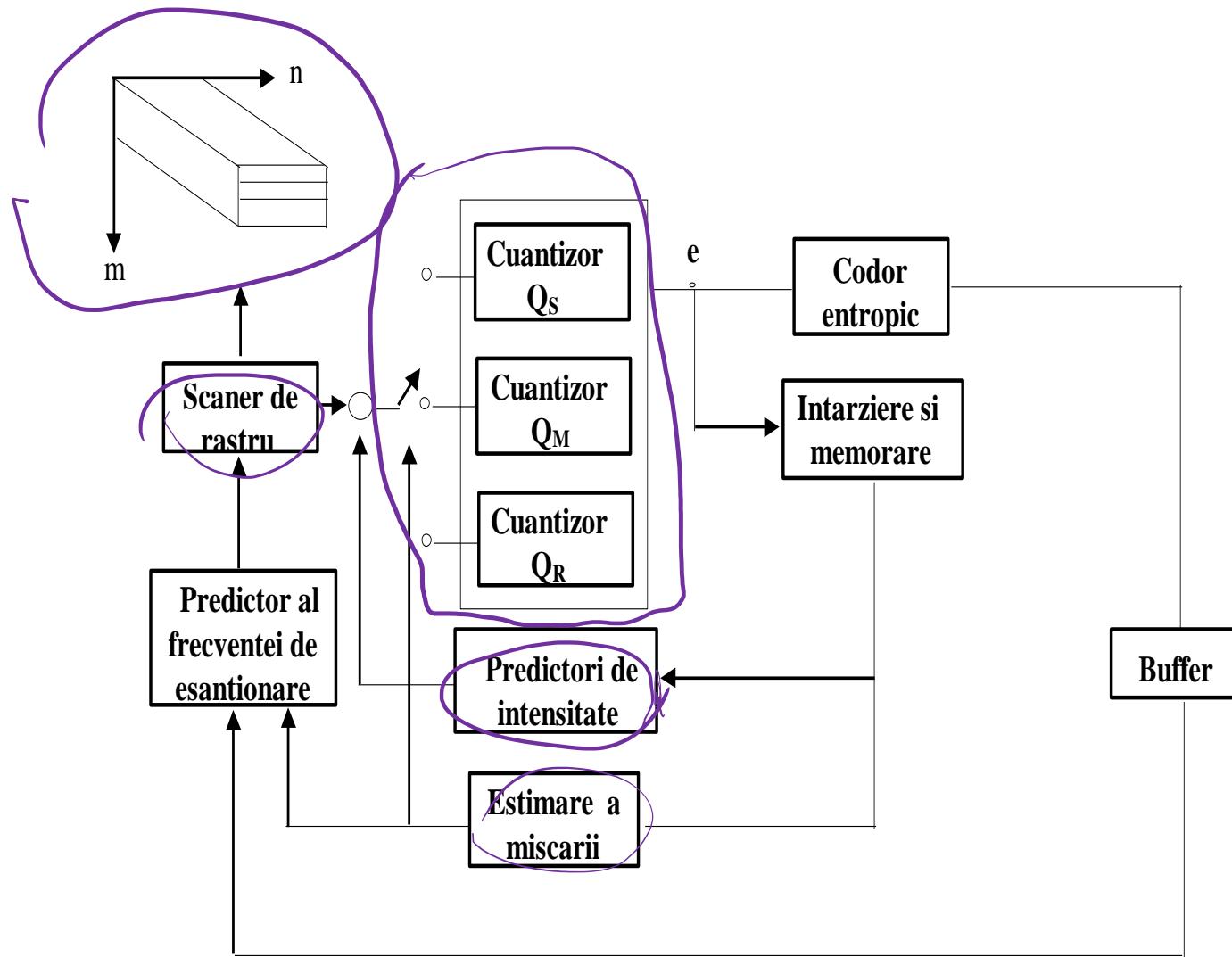
- se bazează pe adaptarea codării la caracteristicile mișcării din imagine.
- pixelii sunt clasificați ca aparținând:
  - unei zone staționare (CS),
  - în mișcare înceată/moderată (CM), sau
  - în mișcare rapidă (CR).
- Clasificarea se bazează pe un indice de activitate  $\alpha(m,n,i)$ , care este suma absolută a diferențelor intercadre dintr-o vecinătate N a pixelilor codați anterior, adică:
- O valoare mare a lui  $\alpha(m,n,i)$  indică mișcare în vecinătatea pixelului.

$$\alpha(m,n,i) = \sum_{(x,y) \in N} |u^*(m+x, n+y, i) - u^*(m+x, n+y, i-1)|$$

$N = \{(0,-s), (-1,-1), (-1,0), (-1,1)\}$

(m-1, n-1)  
(-1, 1) (-1, 0) (-1, -1)  
m, n

# Codarea adaptiva predictiva intercadre - schema bloc



- Valoarea prezisă a pixelului  $u(m,n,i)$  este:

$$\bar{u}^*(m,n,i) = \begin{cases} u^*(m,n,i-1), & (m,n) \in C_s \\ u^*(m-p,n-q,i-1), & (m,n) \in C_M \\ \rho_1 u^*(m-1,n-1,i) + \rho_2 u^*(m-1,n,i) - \rho_1 \rho_2 u^*(m-1,n-1,i), & (m,n) \in C_R \end{cases}$$

- unde  $\rho_1$  și  $\rho_2$  sunt coeficienții de corelație cu pixelii vecini pe direcțiile  $m$  și  $n$ .
- valorile  $p$  și  $q$  se aleg după estimarea deplasării vecinătății  $N$  care dă activitatea minimă

- De exemplu, daca se ignora cadrele de ordin par,  $u(m,n,2i)$ ,  $i=1,2,\dots$ , prin folosirea compensarii miscarii se obtine:
  - prin repetarea cadrelor:
$$u^\bullet(m,n,2i) = u^\bullet(\underline{m-p},\underline{n-q},2i-1)$$
  - prin interpolarea cadrelor:
$$u^\bullet(m,n,2i) = \frac{1}{2} [u^\bullet(\underline{m-p},\underline{n-q},2i-1) + u^\bullet(\underline{m-p'},\underline{n-q'},2i+1)]$$
- unde  $(p,q)$  si  $(p',q')$  sunt vectorii de deplasare relativ la cadrul precedent si, respectiv, urmator.

# Codare predictivă cu compensarea mișcării

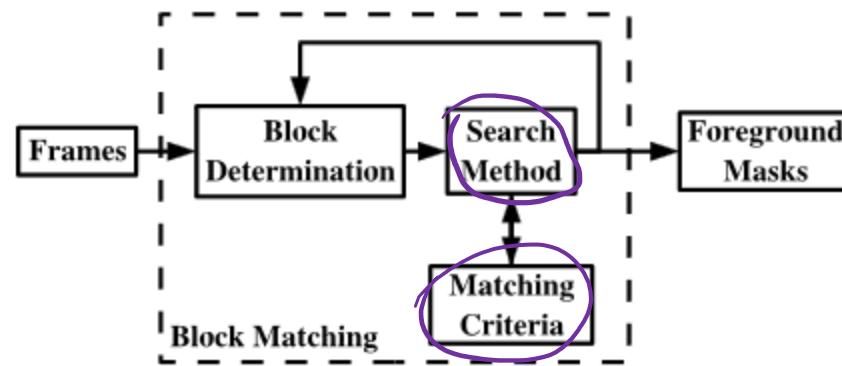
- În cazul imaginilor conținând elemente în mișcare, dacă traectoria mișcării ar putea fi măsurată s-ar putea coda cadrul inițial și informația referitoare la traectoria mișcării.
- La reproducerea imaginii ar trebui pur și simplu să “deplasăm” pixelii de-a lungul traectoriilor lor.
- În practică, mișcarea obiectelor dintr-o scenă poate fi aproximată prin deplasarea treptată de la cadru la cadru.
- Din acest motiv, în codarea cu compensarea mișcării se folosește vectorul de deplasare în vederea obținerii predictorului.
- Succesul codării depinde de acuratețea, viteza și imunitatea la perturbații a metodei de estimare a mișcării.

# Estimarea și compensarea mișcării

- **estimarea mișcării** - constă în determinarea blocurilor din imagine care au suferit o deplasare de la un cadru la altul, precum și a direcției și vitezei de deplasare a acestora.
- **compensarea mișcării** - procesul de reconstrucție a unei imagini, folosind secvențe de imagine din cadre anterioare, împreună cu informații referitoare la mișcare
- Algoritmii de estimare și compensare a mișcării sunt
  - relativ simpli, dar
  - necesită timpi de calcul relativ importanți.

- Dacă o zona de imagine reprezentată de un bloc de pixeli suferă o mișcare relativ lentă (mai mică decât lățimea blocului) și nu este distorsionată semnificativ datorită mișcării, prin schimbarea formei sau orientării, va exista o destul de bună potrivire între acest bloc și blocul de pixeli echivalent din imaginea următoare.
- Prin suprapunerea blocului în diferite poziții ale noii imagini și prin calcularea diferenței matematice între blocuri pentru fiecare poziție, se poate obține gradul maxim de “potrivire” a blocurilor de pixeli, pentru o anumită poziție a acestuia (caz în care, evident, eroarea matematică determinată are valoarea minimă și se situează sub un anumit “prag de potrivire”).
- Dacă s-a găsit o soluție acceptabilă de “potrivire”, noul bloc poate fi reprezentat ca o deplasare a celui anterior fără să mai fie necesară transmisia lui ca atare

# Block Matching Techniques



**Figure 1: Block Matching Flowchart**

# Tehnici de potrivire a blocurilor

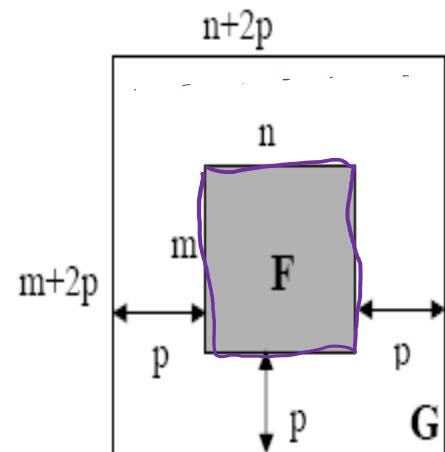
- Valoarea medie absolută a diferenței (MAD)

$$MAD(dx, dy) = \frac{1}{mn} \sum_{i=-n/2}^{n/2} \sum_{j=-m/2}^{m/2} |F(i, j) - G(i + dx, j + dy)|$$

$(dx, dy)$

- Valoarea medie pătratică a diferenței (MPD)

$$MPD(dx, dy) = \frac{1}{256} \sum_{i=-8}^8 \sum_{j=-8}^8 [F(i, j) - G(i + dx, j + dy)]^2$$



- Funcția de cross-corelație (CCF)

$$CCF(dx, dy) = \frac{\sum_i \sum_j F(i, j) G(i + dx, j + dy)}{\left( \sum_i \sum_j F^2(i, j) \right)^{1/2} \left( \sum_i \sum_j G^2(i + dx, j + dy) \right)^{1/2}}$$

$$RSZ = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x, y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2}$$

# Algoritmi de estimare a mișcării

- **Tehnici de căutare** a vectorului de deplasare care asigură minimizarea distorsiunilor între un cadru de referință și cadrul curent.
  - Aceste tehnici sunt utile mai ales în cazul secvențelor de imagini în care deplasarea este constantă pentru blocuri întregi de pixeli. Pentru estimarea mișcării intercadre, căutarea se face de obicei pe ferestre de 5x5 pixeli.
- **Tehnicile diferențiale** se bazează pe faptul că valoarea nivelului de gri a pixelului rămîne constantă de-a lungul traectoriei de mișcare

# Tehnicile diferențiale

$$u((x(t), y(t), t)) = \text{const.}$$

$$\Rightarrow: \frac{\partial u}{\partial t} + v_1 \frac{\partial u}{\partial x} + v_2 \frac{\partial u}{\partial y} = 0, \quad (x, y) \in R$$

unde:

$$v_1 = \frac{\partial x(t)}{\partial t},$$

$$v_2 = \frac{\partial y(t)}{\partial t}$$

= cele două componente ale vitezei;

$R$  = multimea pixelilor în miscare care au aceeași traекторie

- **Vectorul de deplasare  $d$  poate fi estimat ca:**

$$d = \int_{t_0}^{t_0 + \tau} v dt \approx v\tau$$

unde vectorul viteza  $v$  se obține prin minimizarea:  $J = \int \int_R \left[ \frac{\partial u}{\partial t} + v_1 \frac{\partial u}{\partial x} + v_2 \frac{\partial u}{\partial y} \right]^2 dx dy$

# Problema

În tabelul de mai jos este reprezentat cadrul 12 dintr-o secvență video, în care pixelii sunt reprezentați prin luminanțele lor, în domeniul  $\{0,1,\dots,255\}$ . Rata cadrelor secvenței video este de 100 cadre/s. Blocul colorat cu gri din cadrul 12 reprezintă un obiect rigid, care se deplasează cu viteză constantă de-a lungul secvenței, vectorul de deplasare al acestui bloc fiind specificat prin vitezele sale orizontală și verticală: 0.2 pixeli/ms pe orizontală și -0.1 pixeli/ms pe verticală.

- Dacă se consideră că variația luminanței obiectului între cadrele 12 și 20 ale secvenței video este nulă, reprezentați obiectul rigid (conținut de blocul hașurat cu gri în cadrul 12) în cadrul 14 al secvenței video.
- Cum s-ar modifica apariția acestui obiect rigid în cadrul 14 dacă luminozitatea medie a scenei se reduce liniar cu un factor de 0.5 la fiecare 10 cadre (la 10 cadre, luminozitatea în fiecare punct al scenei scade la jumătate)?

0	5	5	5	5	5	5	50	50
1	5	5	5	23	21	20	52	50
2	50	5	5	22	24	65	53	72
3	150	25	35	35	50	75	55	70
4	150	50	25	75	70	85	130	122
5	152	175	175	75	80	90	125	140
6	250	170	170	180	124	95	124	110
7	250	170	170	185	110	120	123	120

a) Cadru 12 → cadru 14 = 2 cadre

100 cadre / s

1s . . . 100 cadre

X . . . 2 cadre

$$X = \frac{1.2}{100} = 0,02 \text{ s} = 20 \text{ ms}$$

deplasare pe x →  $d_x = 0,2 \frac{\text{Pixeli}}{\text{ms}} \cdot 20 \text{ ms} = 4 \text{ pixeli}$

pe y →  $d_y = -0,1 \frac{\text{Pixeli}}{\text{ms}} \cdot 20 \text{ ms} = -2 \text{ pixeli}$



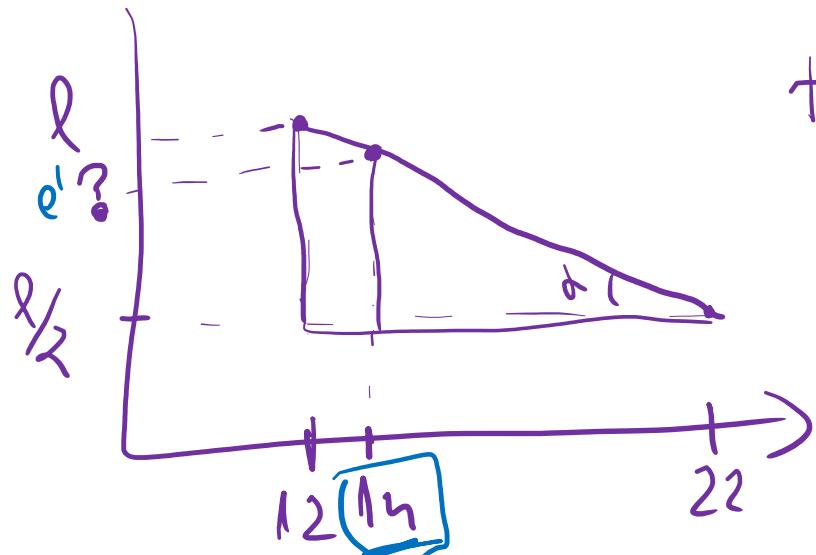
$(0,3) \xrightarrow{(d_x, d_y)} (4,1)$   
 $(4,-2)$

b)

10 cadre . . . 0,5 · l

2 cadre . . . x

$$x = \frac{0,5 \cdot 2}{10} = 0,1 \Rightarrow \text{Scade cu } 0,1 \Rightarrow 0,9l.$$



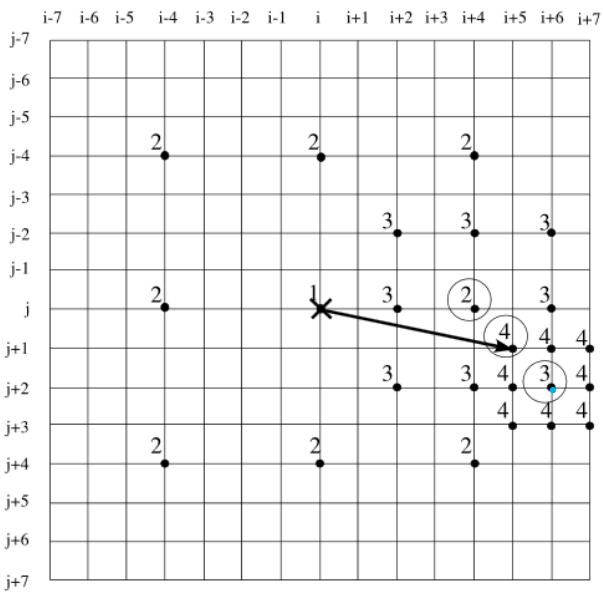
$$l' = (l - l/2) \cdot 0,8 + c_{sj} \cdot l = 0,9l$$

135	23	32	32
135	45	23	68
137	158	158	68
225	153	153	162

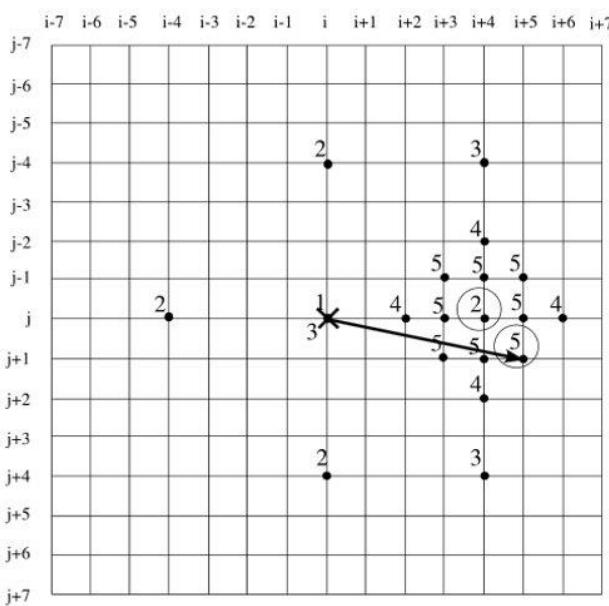
$$\operatorname{tg} \alpha = \frac{l - l/2}{22 - 14} = \frac{l' - l/2}{22 - 14}$$

$$\Rightarrow l' - l/2 = \frac{(l - l/2)(22 - 14)}{22 - 14}$$

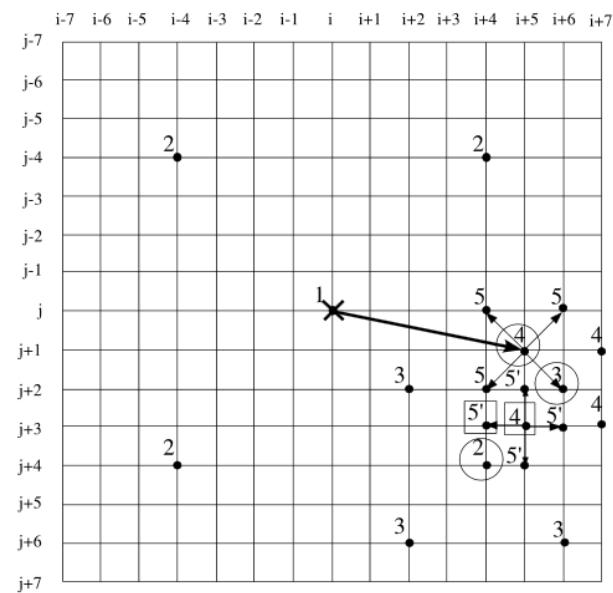
$$l' = \frac{(l - l/2) \cdot 8}{10} + \frac{l}{2} =$$



3: Example of TSS with step size = 4; each numl

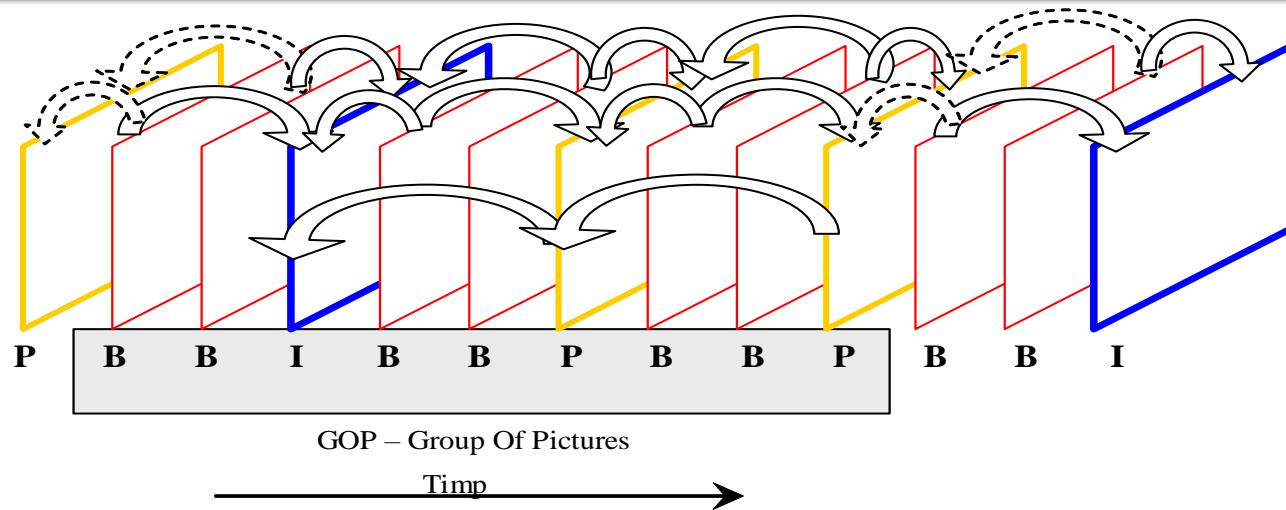


5: Example of 2D-logarithmic search with step size = 4; each number represents



4: Example of cross search with step size = 4; each number represents

# Tipuri de cadre și predicația folosită



- **Cadre I**
  - cadre ce se codează **fără a folosi tehnica de predicție**.
  - se codează asemănător cu o imagine statică JPEG
- **Cadre P**
  - care sunt **imagini prezise** și folosesc predicția cu salt înainte față de un cadru de referință ce poate fi un cadru I, sau P
- **Cadre B**
  - cadre interpolate, folosind două cadre de referință de tip I sau P

## 2.3 Matching Criterion

Given an  $n \times n$  block, a matching criteria,  $M(p, q)$ , measures the dissimilarity of a block in the current frame,  $I_c$ , and a block in the reference frame,  $I_r$ , shifted by  $(p, q)$ . These criteria can be characterized by  $M(p, q) = \sum_{i=1}^n \sum_{j=1}^n \phi(e)$ , where  $e = I_c(i, j) - I_r(i + p, j + q)$  and  $\phi(e)$  is the criteria function. Figure 6 shows the criteria functions for a given  $e$ . We examine four matching criteria which are also known as error or matching functions.

**SAD** The sum of the absolute values of the differences in the two blocks:

$$M(p, q) = \sum_{i=1}^n \sum_{j=1}^n |I_c(i, j) - I_r(i + p, j + q)|$$

**MAD** The mean of the absolute values of the differences in the two blocks:

$$M(p, q) = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n |I_c(i, j) - I_r(i + p, j + q)|$$

**MSD** The mean of the square of the differences in the two blocks:

$$M(p, q) = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n (I_c(i, j) - I_r(i + p, j + q))^2$$

**MPC** The sum of the non-matching pixels in the two blocks, a match is determined by the absolute value of the difference being less than a threshold,  $t_{MPC}$ .

$$M(p, q) = \sum_{i=1}^n \sum_{j=1}^n D(I_c(i, j), I_r(i + p, j + q))$$

$$D(a, b) = \begin{cases} 0 & \text{if } |a - b| \leq t_{MPC} \\ 1 & \text{otherwise} \end{cases}$$

SAD and MAD only differ by a constant in the case of fixed size blocks and can be used interchangeably in our comparison. Practically, SAD is faster due to the removal of the divide operation. While MAD incorporates large differences, MSD penalizes more a block with one or more large differences. MPC on the other hand equally weights any difference above a threshold.

# SACCDMM - Curs 10

## Standardul de compresie MPEG

Sl.Dr.Ing. Camelia FLOREA

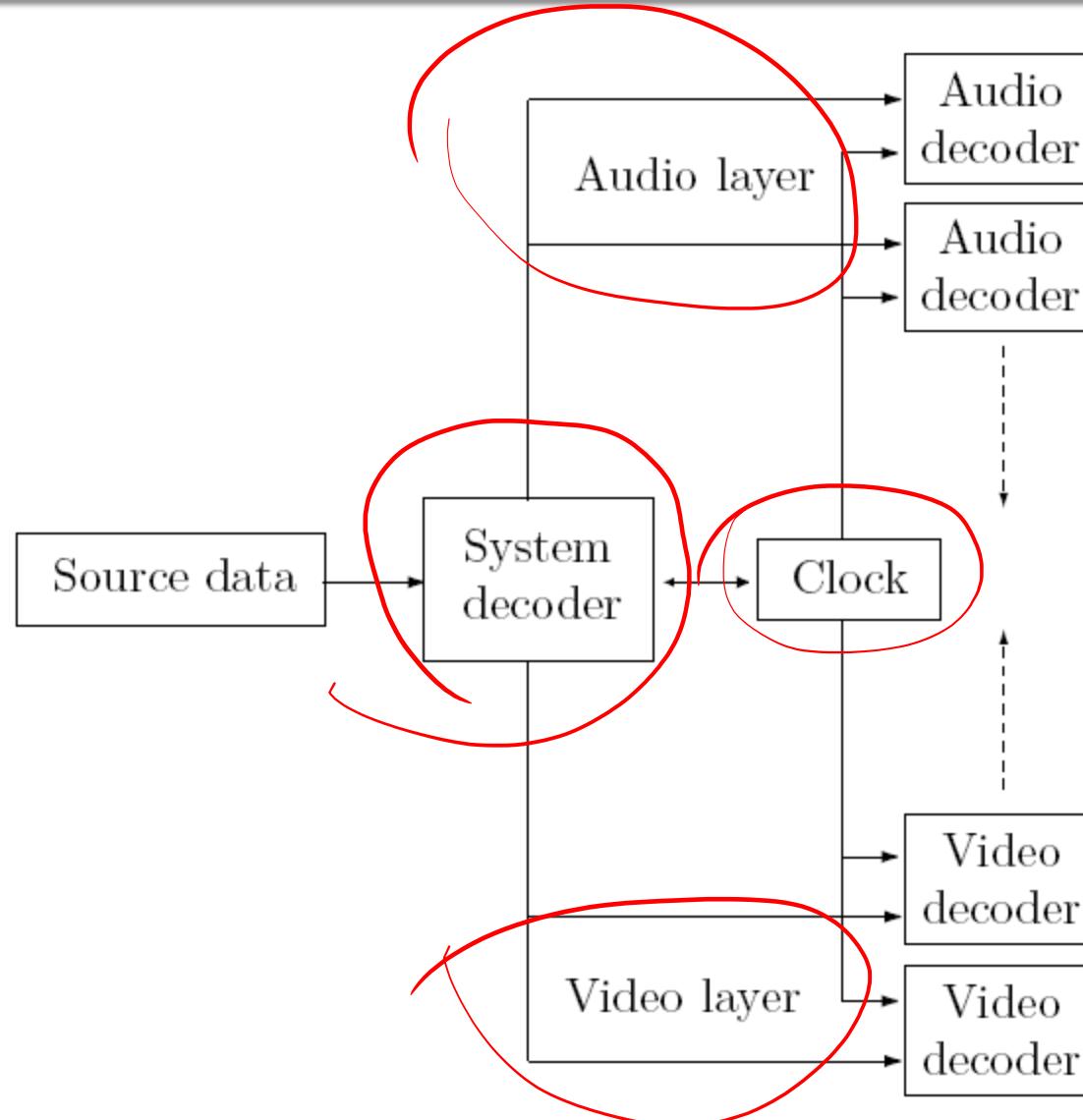
# Standardul de compresie MPEG

- MPEG (Moving Pictures Experts Group)
  - ISO (International Standardization Organization), si
  - IEC (International Electrotechnical Committee).
- Necesitatea:
  - **Capacitatea** de stocare **pe disk** – capacitate limitată de stocare
  - **Transmisia** semnalului video digital – spectru de frecvență limitat
  - **HDTV** – folosirea imaginilor de **calitate superioară**
  - **Aplicații** multimedia – aceeași codare diferiți utilizatori, rețele

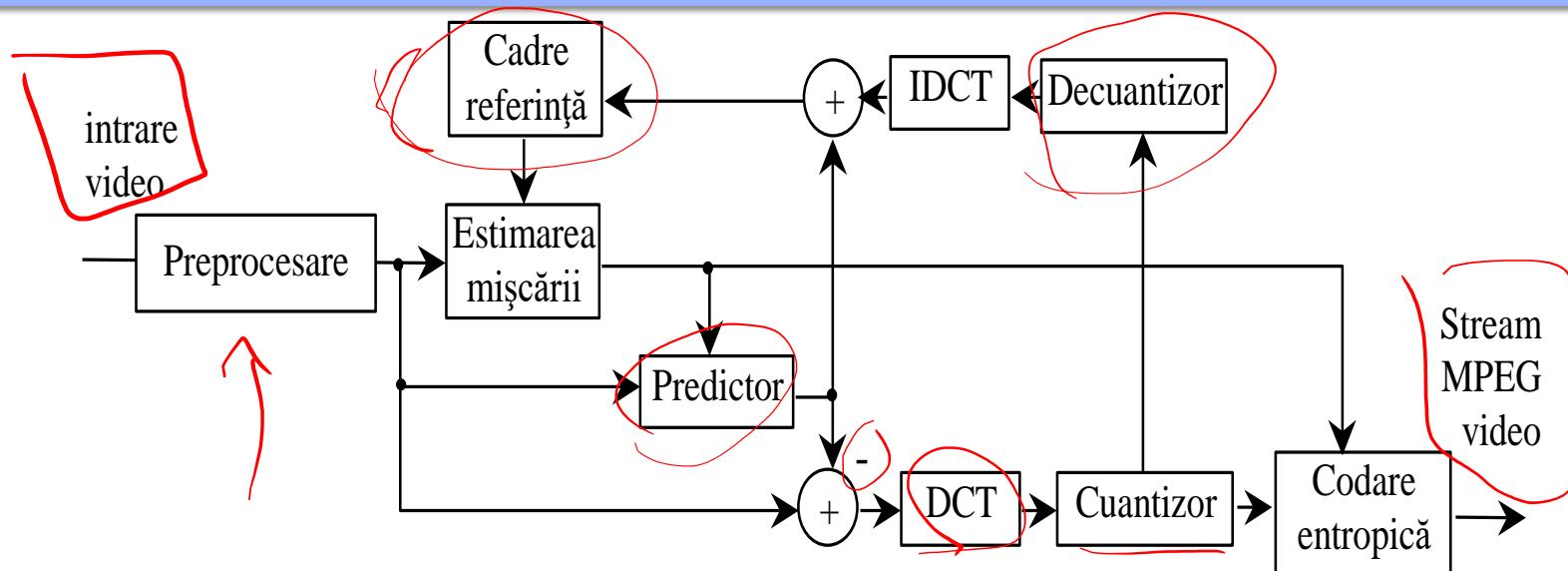
- **MPEG-1**
  - rate intermediare: apox. 1.5 Mbit/sec
- **MPEG-2**
  - rate ridicate de compresie: aprox. 10 Mbit/sec
- **MPEG-3**
  - pentru compresie HDTV, dar s-a ajuns la concluzia ca este redundant si s-a unit cu MPEG-2
- **MPEG-4**
  - Rate foarte mici de compresie: < 64 Kbit/sec.

# Componentele MPEG

- **Video**
  - Compresia imaginilor în mișcare
- **Audio**
  - Compresia semnalului audio
- **System**
  - descrie modalitățile în care diferitele tipuri de stream-uri (audio, video, sau date generice) sunt **multiplexate și sincronizate**

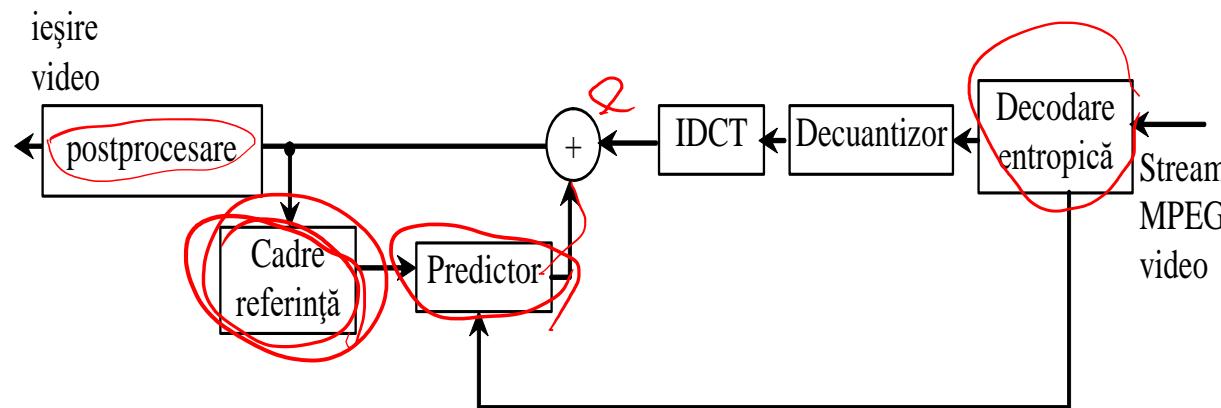


# Componenta MPEG video



- Blocul de procesare (interpolare și filtrare)
- Blocul de estimare a mișcării
  - determină predictorul cadrului curent, din cadrele transmise anterior
  - vectorii de mișcare - sunt transmiși blocului de codare entropică
- Predictorul este scăzut din fiecare bloc în parte,
  - => eroarea de predicție -> trimisă blocului DCT.
- Coeficienții DCT sunt cuantizați și apoi codăți entropic
- Coeficienții DCT cuantizați
  - sunt folosiți pentru a reconstrui cadrele de referință

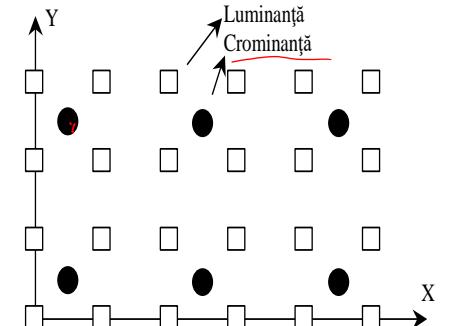
# Decodorul MPEG video



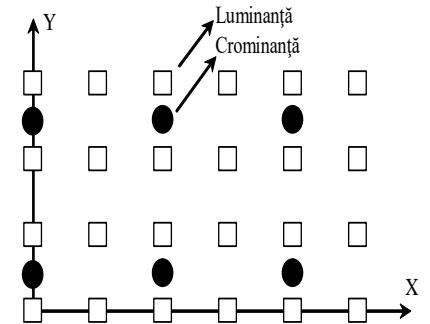
- decodarea entropică -> cauantizare inversă (decuantizare) -> IDCT.  
-> Informația obținută este sumată cu cadrul de referință => cadrul curent refăcut.
- cadrul curent refăcut
  - este trecut printr-un bloc de postprocesare (interpolare și filtrare).

# Reprezentarea secvențelor

- Pe cadre
  - MPEG-1
    - Y, Cr, Cb (croma eșantionată cu factor 2)
    - MPEG-1 nu specifică modul de subeșantionare - se poate considera metoda din figura alăturată
  - MPEG-2, Y, Cr, Cb
    - MPEG-2 specifică subeșantionarea orizontală



a) standardul MPEG-1



b) standardul MPEG-2

# Reprezentaera secvențelor pe campuri

- MPEG 2
  - divizarea fiecăruia cadru în două câmpuri (semicadre) întrețesute
    - liniile (rândurile) pare respectiv
    - liniile impare dintr-un cadru.

# Tipuri de cadre

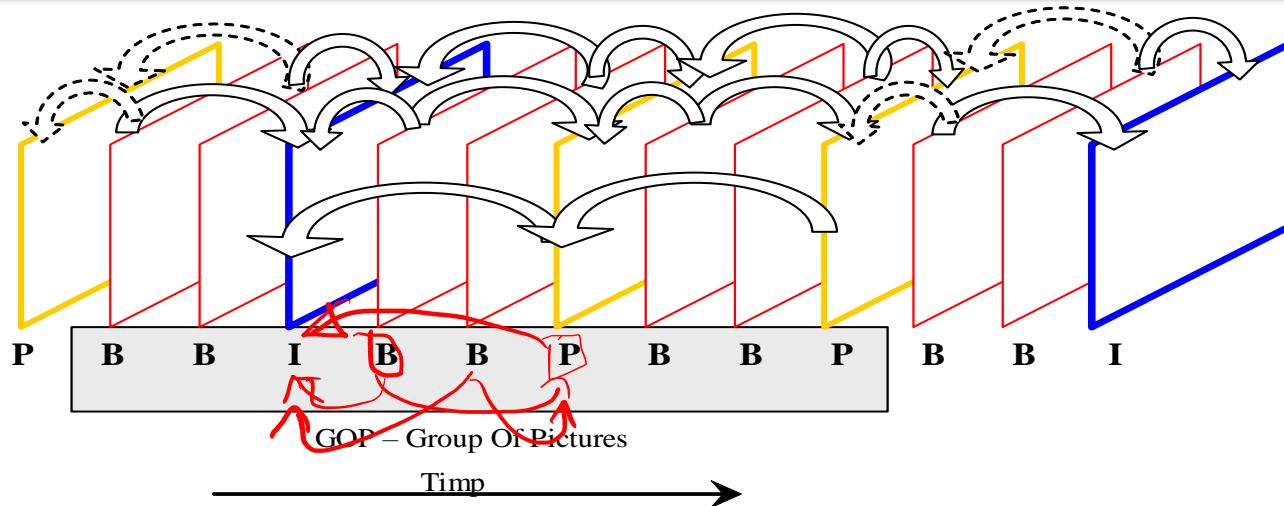
- Probleme:
    - Acces aleator la cadre
    - Trebuie decodată întreaga secvență
    - Apariția unei erori SE PROPAGĂ
  - Soluții
    - Procesul de refresh – din loc în loc câte un cadru NU se codează temporal
    - Predicția cu îintreruperi – eliminarea erorilor de predicție
- întra - lâne  
predicție (I)*  
*înter - B, P*



■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■  
I B B B P B B B P B B B P

- I → JPEG
- P → predicted from previous I / P  
→ JPEG
- B → Backward prediction from previous I / P  
→ Forward prediction from next I / P  
→ Average between previous B and next B  
→ JPEG

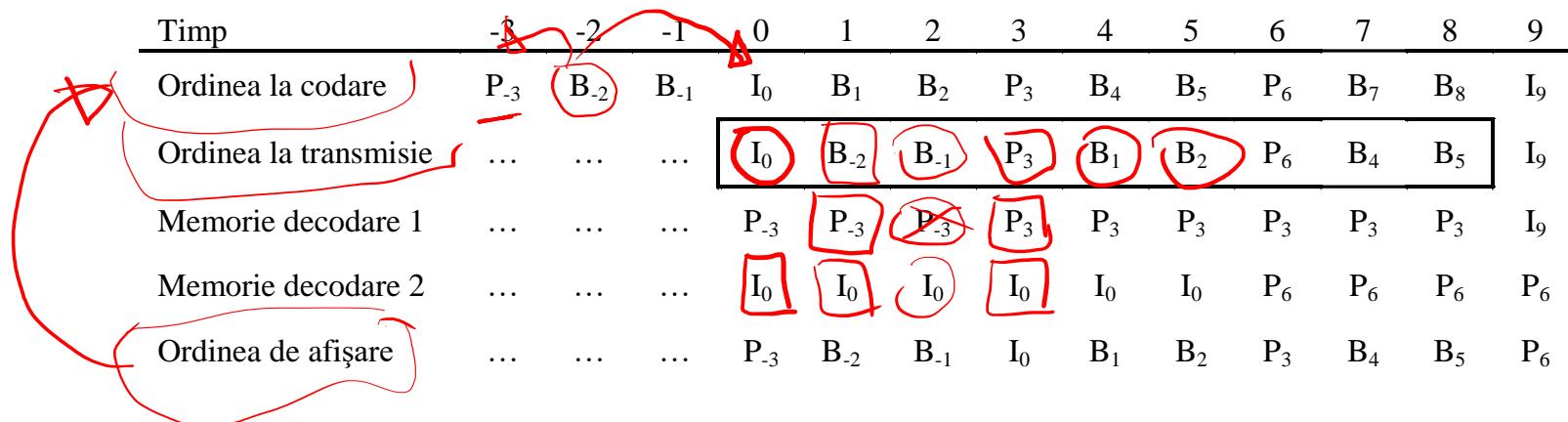
# Tipuri de cadre și predicația folosită



- **Cadre I**
  - cadre ce se codează **fără a folosi tehnica de predicție**.
  - se codează asemănător cu o imagine statică JPEG
- **Cadre P**
  - care sunt **imagini prezise** și folosesc predicția cu salt înainte față de un cadru de referință ce poate fi un cadru I, sau P
- **Cadre B**
  - cadre interpolate, folosind două cadre de referință de tip I sau P

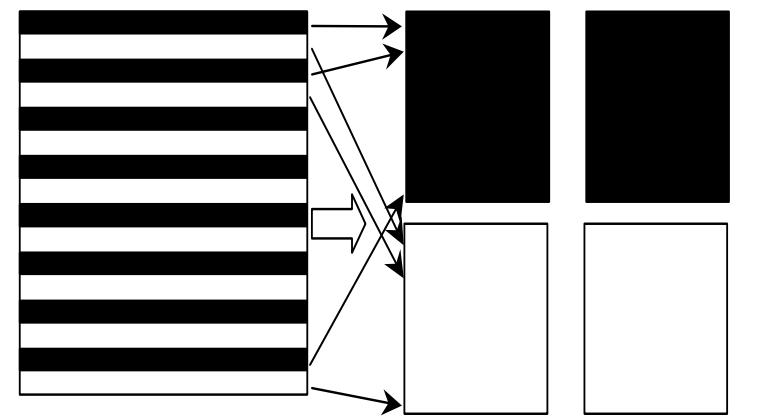
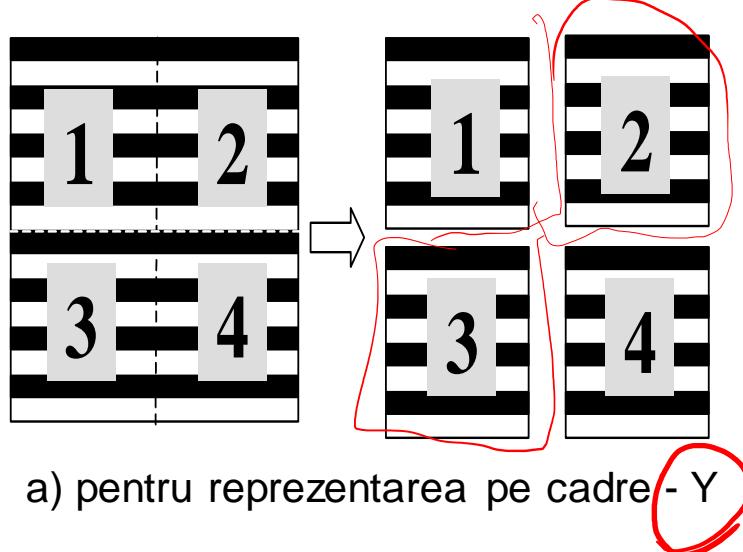
# Afișarea și ordinea la transmisie

- GOP – Group of Pictures
  - o secvență continuă de cadre, care începe cu un cadru de tip I (inclusiv) și se termină cu următorul cadru de tip I (exclusiv).
- pentru a reface cadrele de tip P și B avem nevoie de cadrele de referință corespunzătoare.
- Evitarea stocării informația în mod neeficient la decodor  
⇒ ordinea de transmisie a cadrelor MPEG, va fi diferită:
  - orice cadru recepționat trebuie să poată fi imediat decodat și afișat.



# Transformata DCT

- Împărțire în 6 blocuri: 4 Y, 2 Croma



# Cunatizarea

- vechiul de dupăcare  
- ~~Q~~ + v.d.  
→ DCT + cuant

- \* este coeficientul DC – nu se cuantizează

- intra - DCT + cuant

*	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
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

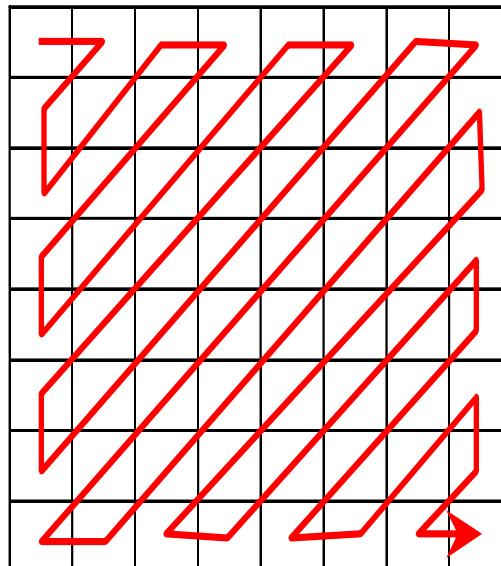
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

a) pentru cadre fără predicție

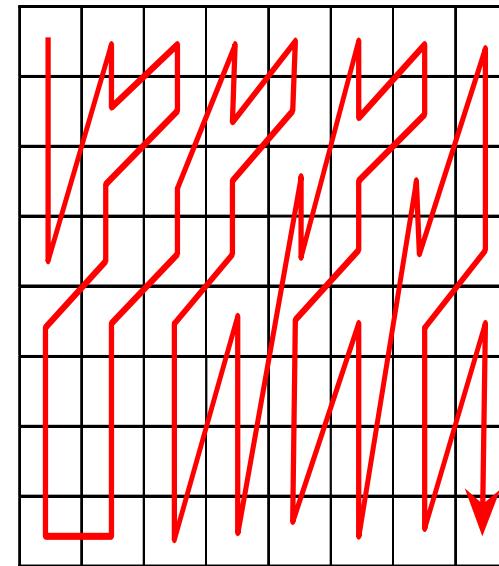
b) pentru cadre cu predicție

# CODAREA ENTROPICĂ

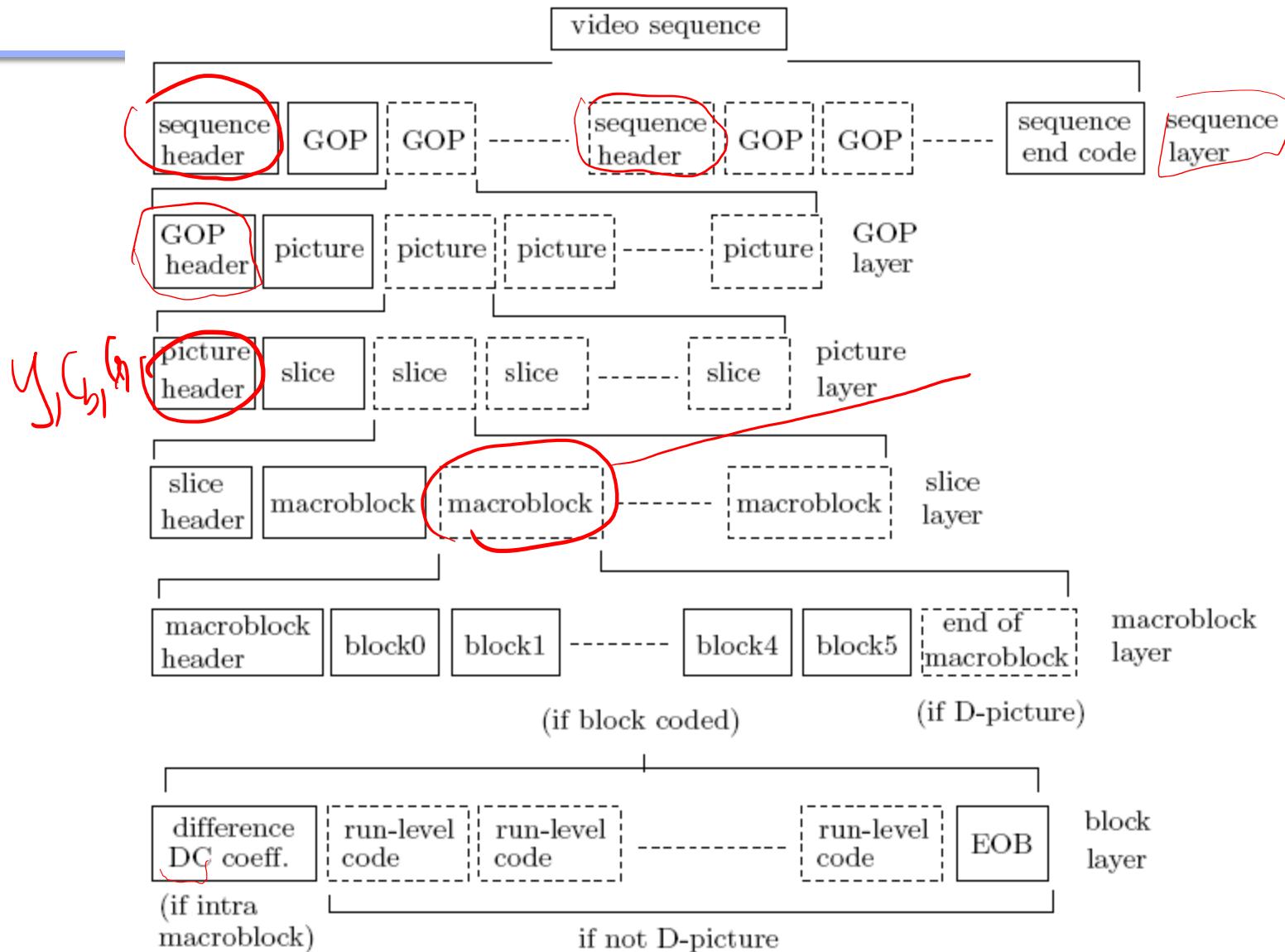
- Conversia 2D->1D
  - scanare zig-zag – MPEG 1, 2
  - scanare verticală – MPEG 2
- Codarea RLC
- Codarea Huffman modificată – codarea perechilor



Zig-zag



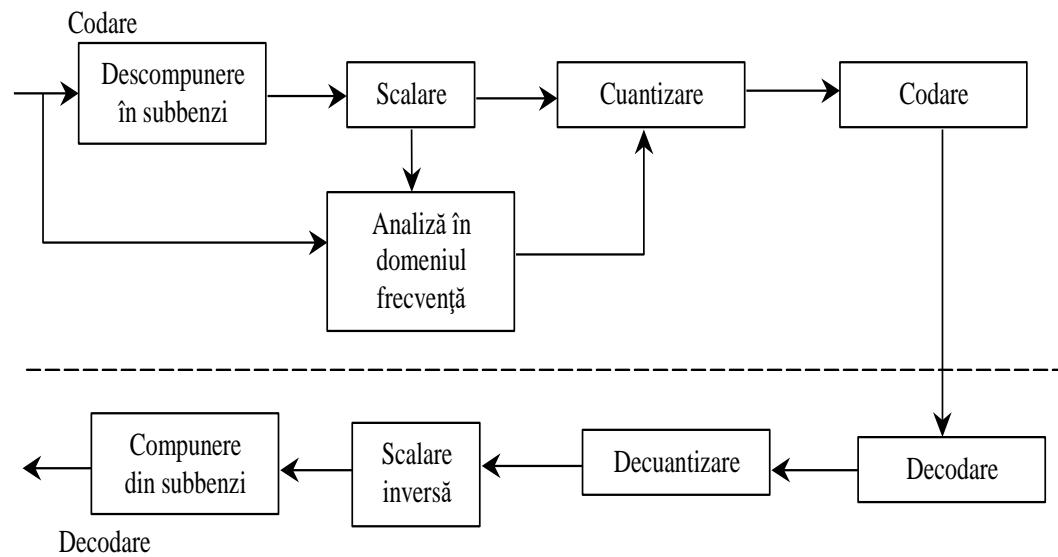
verticală



# Componenta MPEG audio

- Standardul de compresie audio MPEG
  - definește o serie de algoritmi (pentru diferitele tipuri de materiale audio – vorbire, muzică, efecte speciale) bazați pe compresia pe subbenzi folosind tehnici ce exploatează proprietățile sistemului auditiv uman.

# Componenta AUDIO



- La codare
  - Descompunerea în subbenzi separă semnalul audio de intrare în benzi multiple de frecvență.
  - Fiecare subbandă este scalată și cuantizată (factorul de scalare diferă de la o subbandă la alta).
  - În paralel se face o analiză în domeniul frecvență pentru a determina pasul de cuantizare specific fiecărei subbenzi în parte.
  - Eșantioanele sunt apoi codate împreună cu informații suplimentare folosind codarea Huffman și apoi transmise.

# ESTIMAREA MIŞCĂRII

$I \rightarrow P \rightarrow B$

- Cadrele de tip **B** pot folosi fie:

- predicția cu salt înainte:

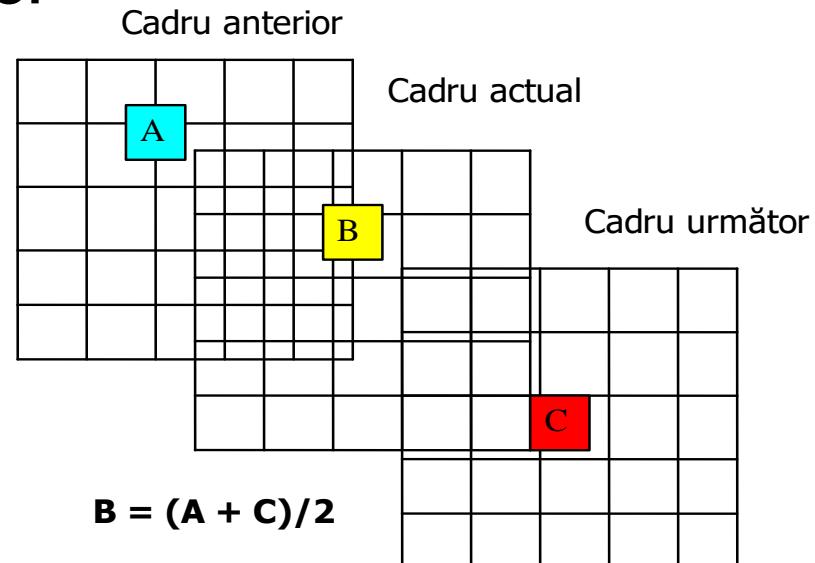
$$B=A$$

- fie predicția cu salt înapoi:

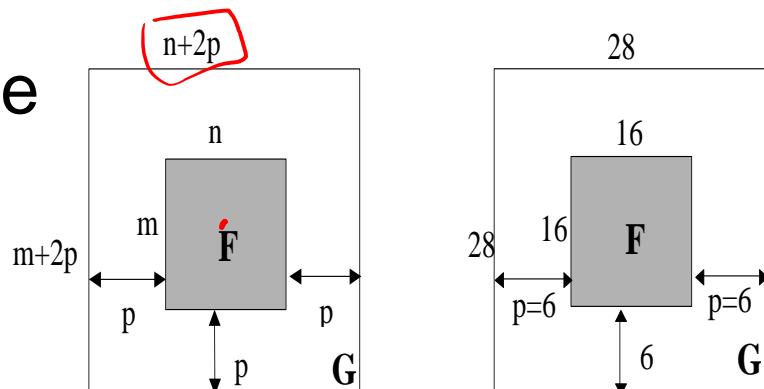
$$B=C$$

- fie interpolarea:

$$B=(A+C)/2$$

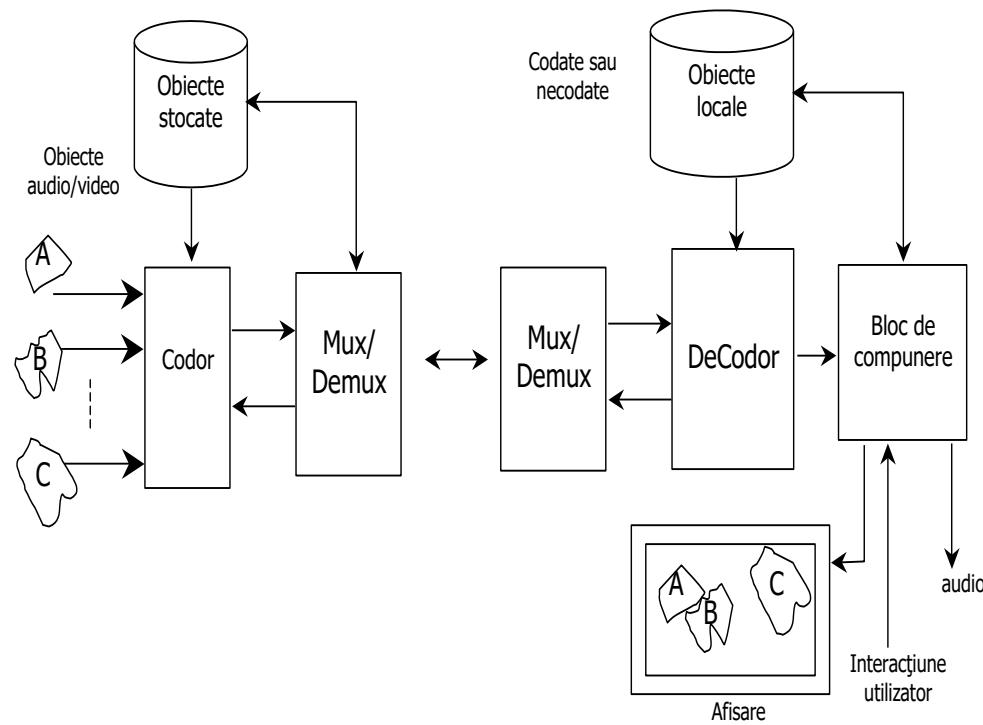


- Aria de căutare a vectorului de estimare:

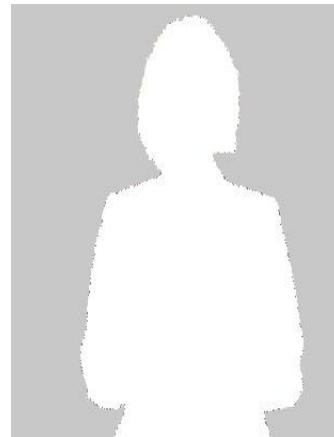


# MPEG-4

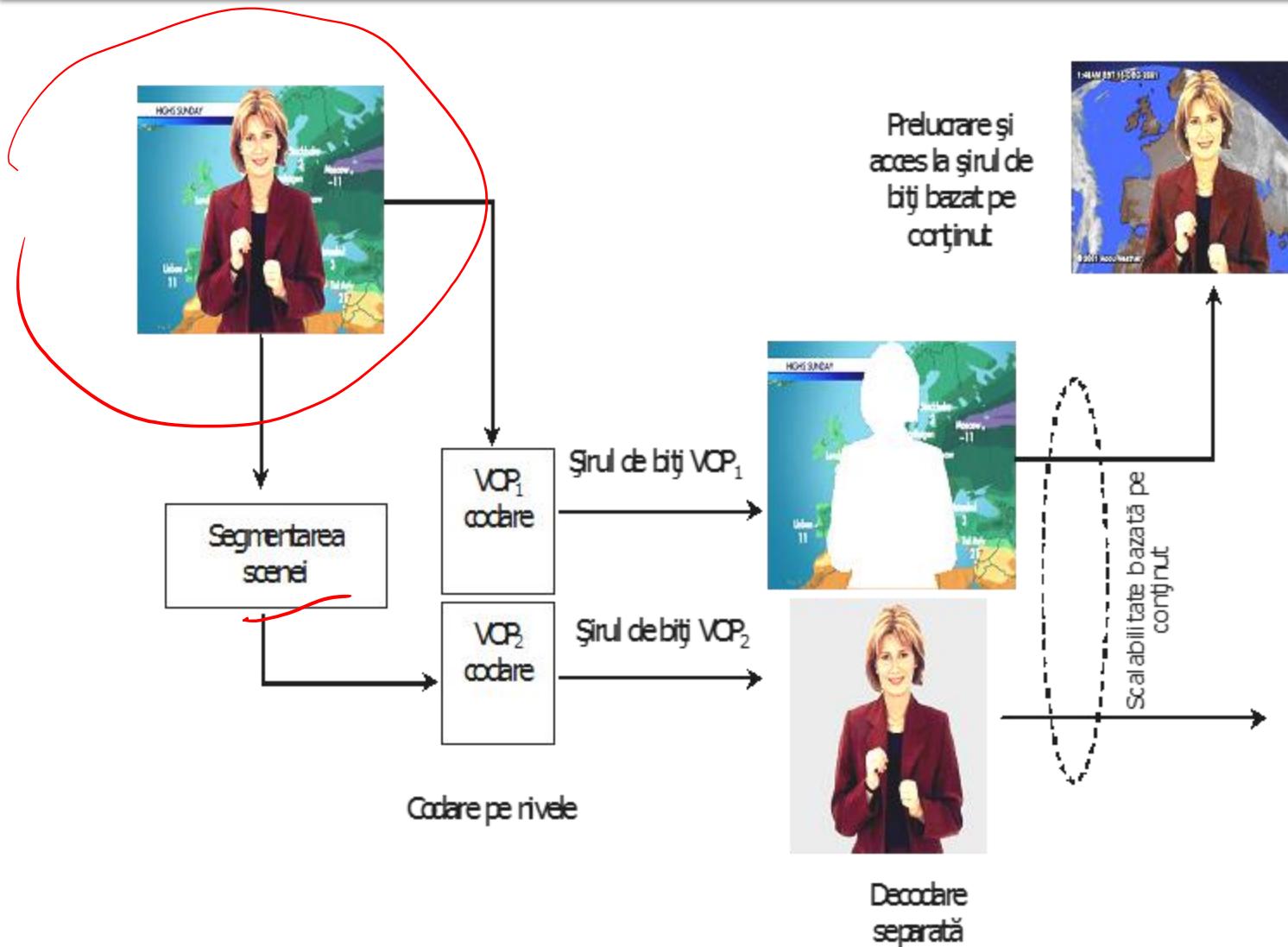
- Scenă audiovizuală
- **Obiecte** audio, video între care există relații temporale
- Obiect video: fundal, prezentator, text
- Obiect audio: un singur instrument,



# VOP – Video Object Planes



# Codarea VOP



# MPEG-7, MPEG-21

- MPEG-7
  - extinde posibilitățile limitate de căutare
  - pot fi incluse imagini staționare, grafică, secvențe de imagini, informații de compoziție precum și câteva cazuri speciale cum ar fi expresii faciale
- MPEG-21
  - Crearea unui cadru multimedia pentru cooperarea:
    - Obiectele digitale
    - Identificare obiecte
    - Utilizare-manipulare conținut
    - Terminale și rețele
    - Reprezentarea conținutului
    - Raportare de evenimente

# SACCDMM - Curs 11

## Standardele de compresie H.26x

Sl.Dr.Ing. Camelia FLOREA

# Topicul cursului

- Formate de imagine – rezoluția
  - H.261
  - H.262
  - H.263
  - H.264
- 
- [http://iphome.hhi.de/wiegand/assets/pdfs/2012\\_12 IE EE-HEVC-Overview.pdf](http://iphome.hhi.de/wiegand/assets/pdfs/2012_12_IE_EE-HEVC-Overview.pdf)

# Formate de imagine

Resolution	Dimensions	Pixel/s at 30 frames/s	Applications
Sub-QCIF	128 × 96	0.37 M	Handheld mobile video and
QCIF	176 × 144	0.76 M	videoconferencing via public
CIF	352 × 288	3.04 M	phone networks Videotape recorder quality
CCIR 601	720 × 480	10.40 M	TV
4CIF	704 × 576	12.17 M	NTSC – for PAL – 720x576
HDTV 1440	1440 × 960	47.00 M	Consumer HDTV
16CIF	1408 × 1152	48.66 M	Video surveillance - DVR
HDTV	1920 × 1080	62.70 M	Studio HDTV
Video surveillance – DVD applications			

# H.261 [Solomon07, p.703]

- 1984 – CCITT (actualmente ITU-T)
  - a organizat un grup de experti pentru dezvoltarea unui **standard de videotelefonie pe ISDN**:
    - *transmitere de imagini*, și
    - *sunet* pe terminale speciale=> utilizatorii pot să se și vadă în timpul unei conversații telefonice
- Au fost propuse o serie de standarde de compresie:
  - H.xxx – pentru componenta video
  - G.xxx – pentru componenta audio
- toate operează la viteza de  $p \times 64$  Kbit/sec,  $1 \leq p \leq 30$  (64-1920Kbps)=> numite - **Standardele  $p \times 64$**

# H.261

Standard	Purpose
H.261	Video Codecul video – poate fi folosit la H.320, H.323
H.221	Communications Structura transmisiei
H.230	Initial handshake Semnalizare – inclus în H.320
H.320	Terminal systems Aplicații audio-video-data peste ISDN
H.242	Control protocol
G.711	Companded audio (64 Kbits/s)
G.722	High quality audio (64 Kbits/s)
G.728	Speech (LD-CELP @16kbits/s)

Table 6.37: The  $p \times 64$  Standards.

H.323 – aplicații audio-video-data pe rețele comutare de pachete

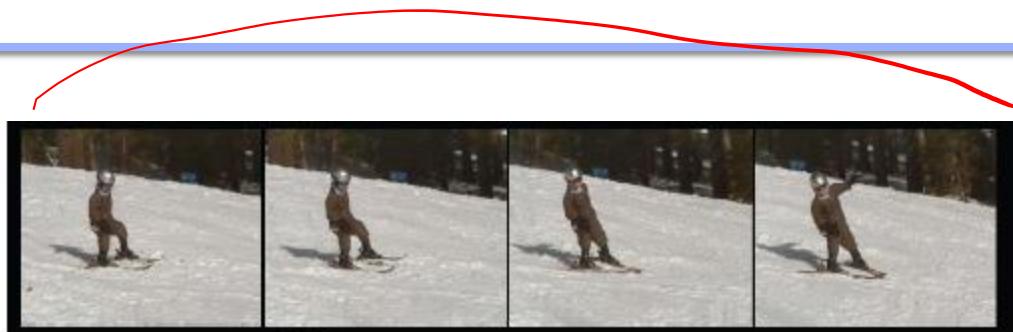
# H.261

- Membrii a grupului de experți px64 au participat și la dezvoltarea MPEG  
=> multe elemente comune ale algoritmului de codare
- Diferențe
  - MPEG
    - codorul poate fi complex, lent
    - decodorul trebuie să fie rapid, pentru operarea în timp real
    - => compresie asimetrică
  - H.261
    - codorul și decodorul sunt rapide – operare în timp real
    - Standardul definește:
      - structura stream-ului de date și
      - arhitectura decodorului
    - Pentru codor orice metodă care obține stream-ul de date...

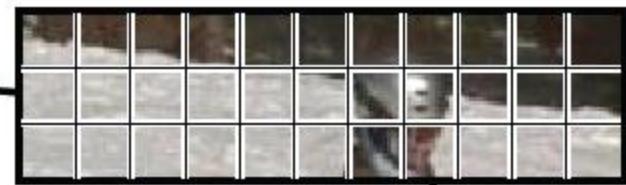
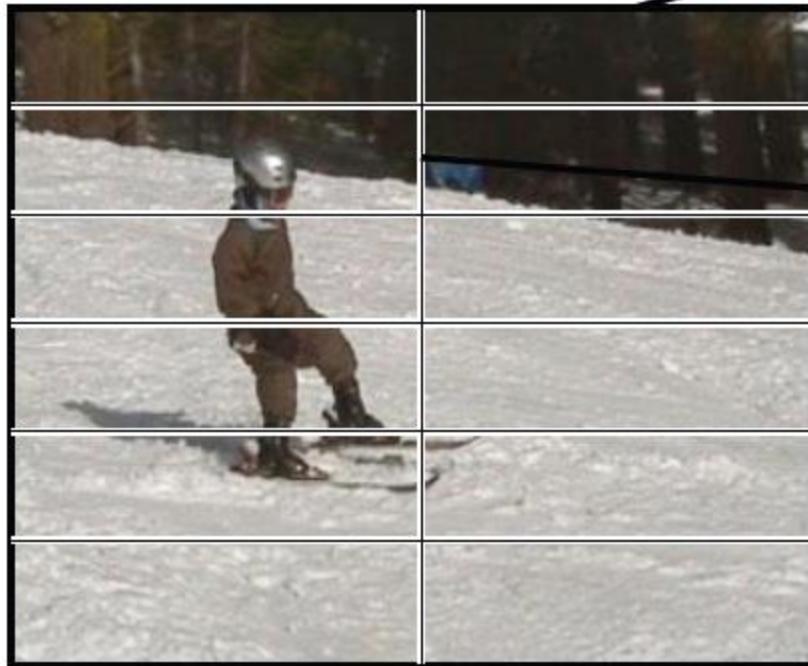
# H.261

- CIF

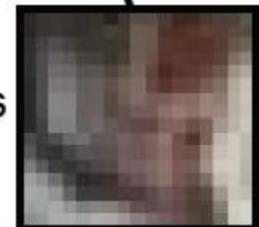
## H.261 structure



Video composed of frames



Each GOB is composed of  
11x3 MacroBlocks



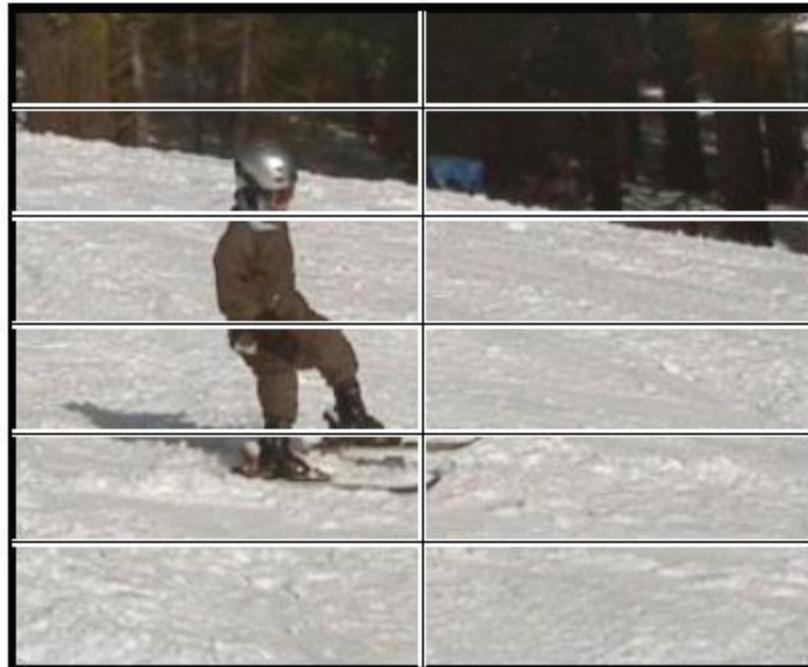
Each MB is  
16x16 pixels

Each CIF frame composed of 12  
Groups of Blocks (GOBs)

# H.261

- QCIF

## CIF and QCIF Frame Formats



Each CIF frame (352x288 pixels) is composed of 12 Groups of Blocks (GOBs)

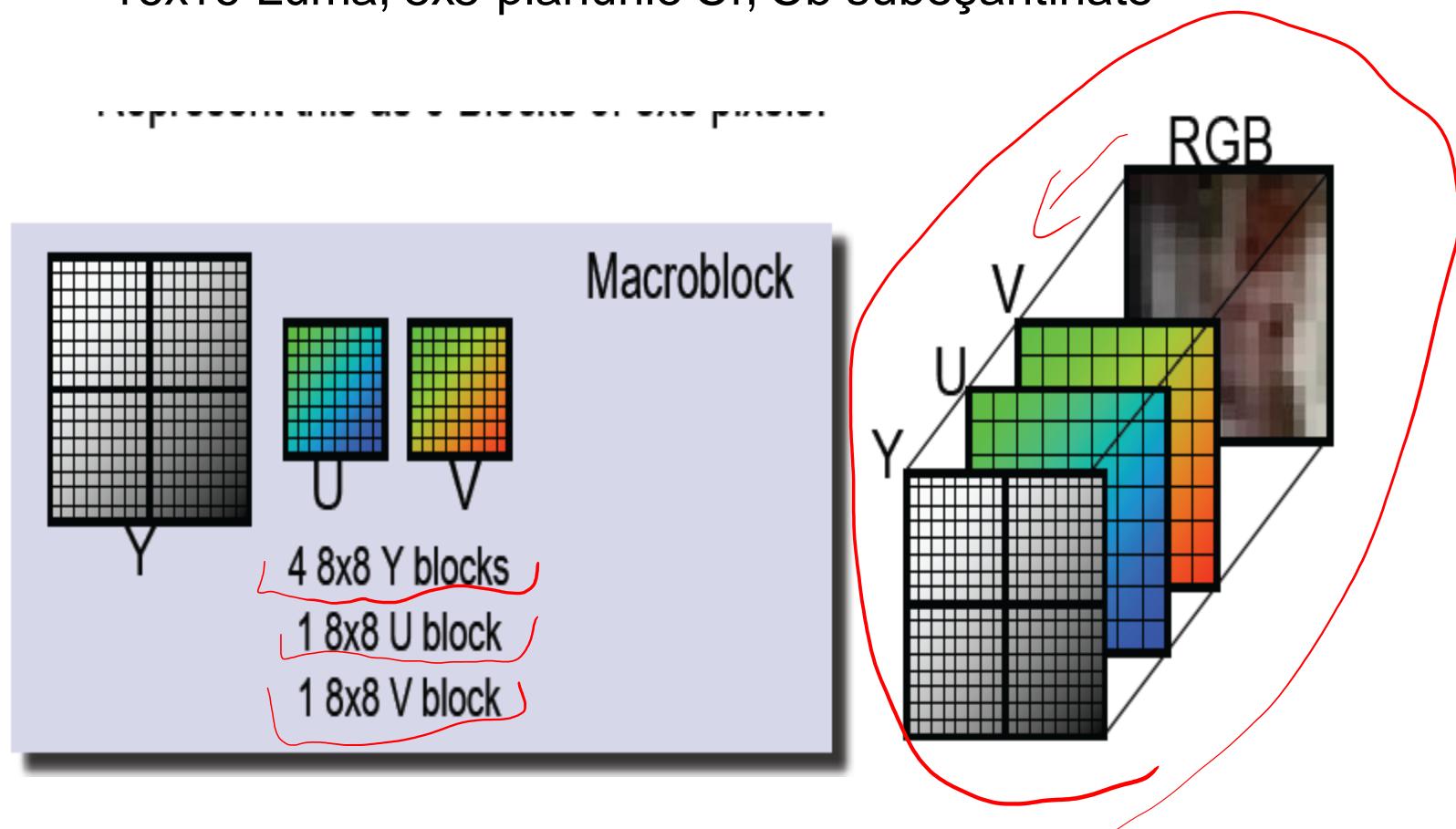


Each QCIF frame (176x144 pixels) is composed of 3 Groups of Blocks (GOBs)

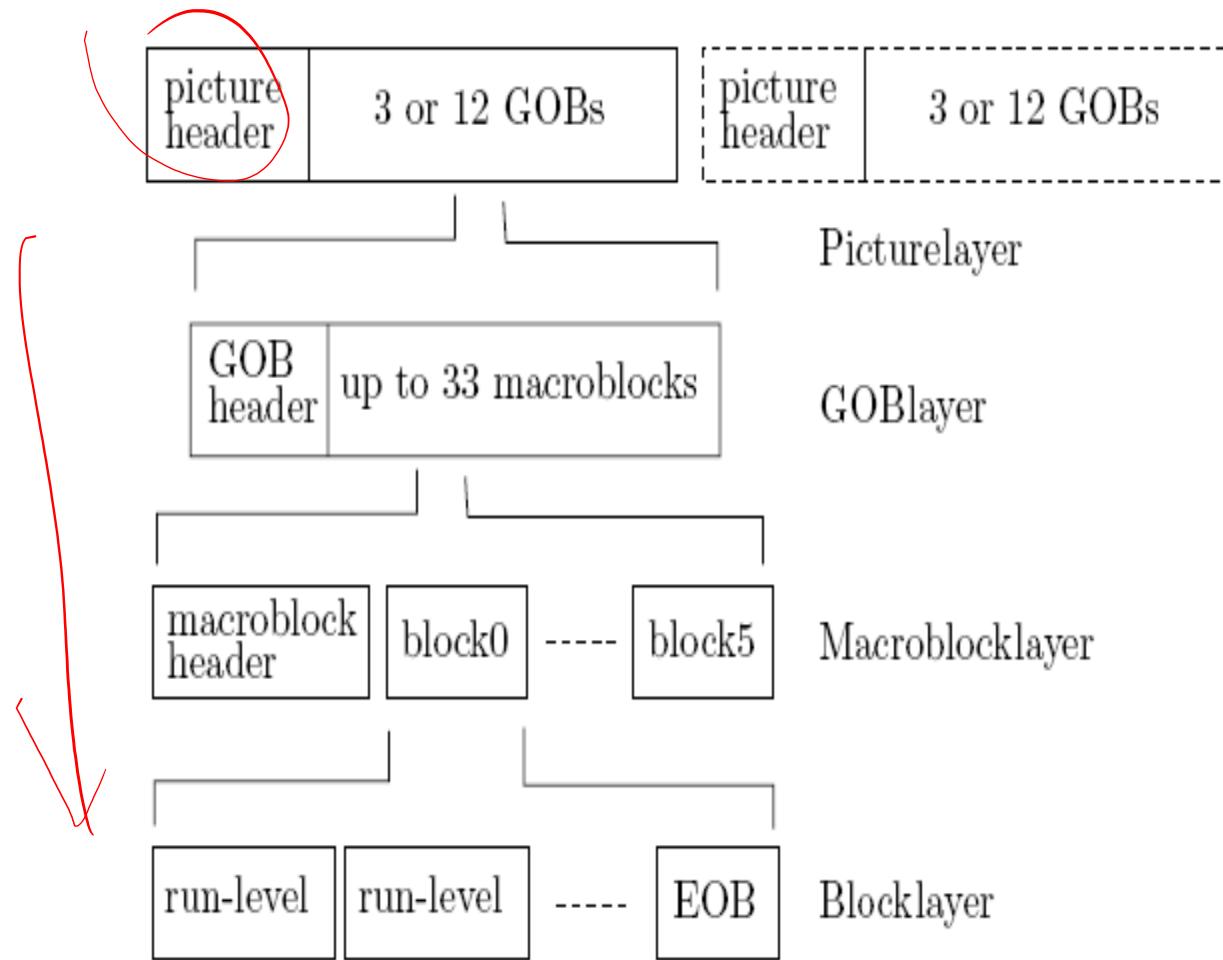
GOB and MacroBlock format is identical in both frame formats.

# H.261

- Fiecare macrobloc **16x16 pixeli** (YUV, 4:2:0)
  - 16x16 Luma, 8x8 planurile Cr, Cb subeșantionate



# H.261



# H.261- codarea macroblocurilor

- Trei moduri de compresie
  - Nu se codează – dacă cadrul este identic cu precedentul  
nu se transmite
  - Compresie intra-cadru
    - DCT, cuantizare, zigzag, RLC, Huffman – JPEG
  - Compresie inter-cadru
    - Diferența dintre cadre
    - Se poate folosi estimarea mișcării – codarea vectorilor de mișcare
    - Compresie intra-cadru DAR pe diferență

# H.261 – codarea intra-cadru

- Codarea intra-cadru – similar JPEG
  - DCT
  - Cuantizarea coeficienților DCT
    - Se folosește o singură valoare pentru cuantizare în loc de matrice de 8x8 la JPEG
    - Buclă pentru modificarea dinamică a nivelului de cuantizare pentru a menține rata de bit
  - Ordonarea zigzag
  - RLC
  - Huffman

# H.261 – codarea inter-cadre

- Proces similar cu codarea intra-cadru
- Nu avem cadre I, P, B... ca la MPEG
- Datele sunt – diferența dintre cadre succesive

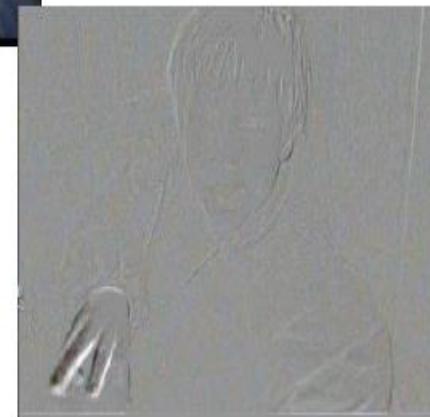


Frame 1



Frame 2

Difference:  
Frame 2 - 1



# H.261 – mișcarea

- Mișcarea în scenă duce la creșterea diferenței intercadru
- Se configurează un algoritm de estimare al mișcării
  - Trimite vectorul de mișcare – doi întregi  $(x, y)$  care dau mărimea deplasării pe orizontală și verticală
  - Se codează diferența față de blocul deplasat (DCT, cuantizare, ... etc.)

# H.261 – căutarea vectorilor de mișcare

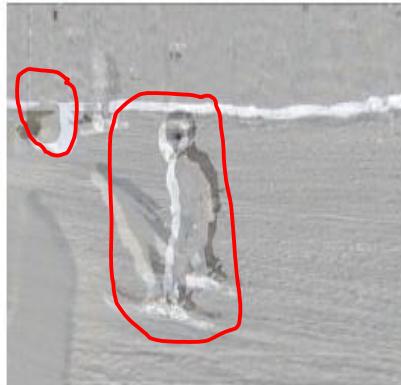


Frame 1

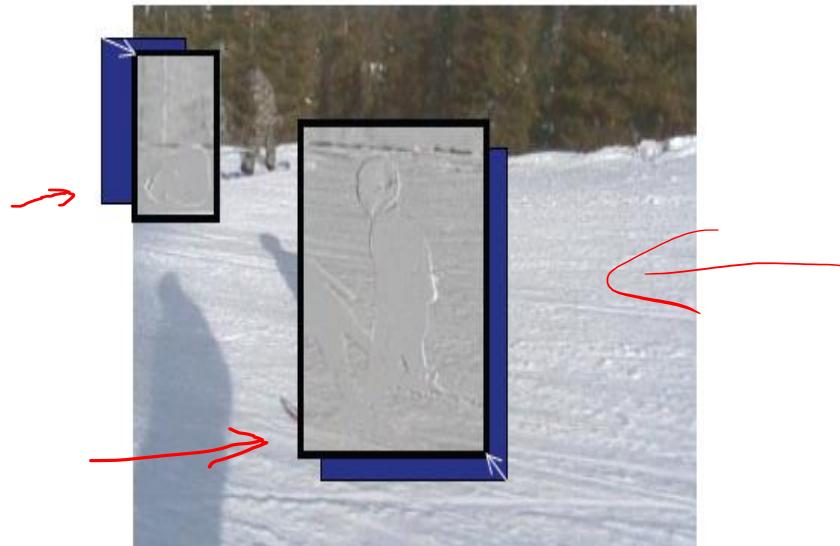


Frame 2

Coding from moved part of previous image can reduce the differences

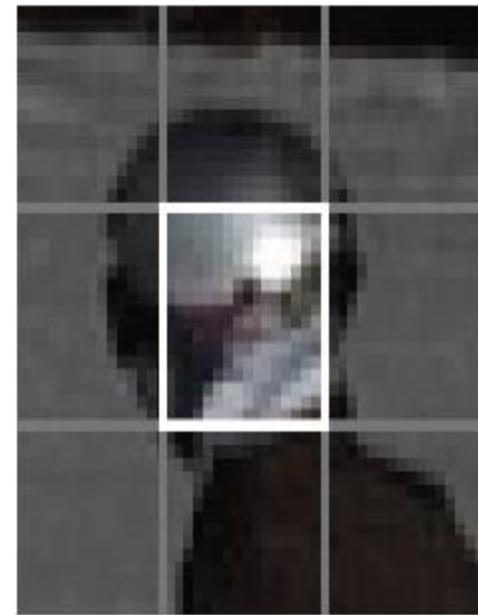


Frame 2 - 1  
(lots of motion)



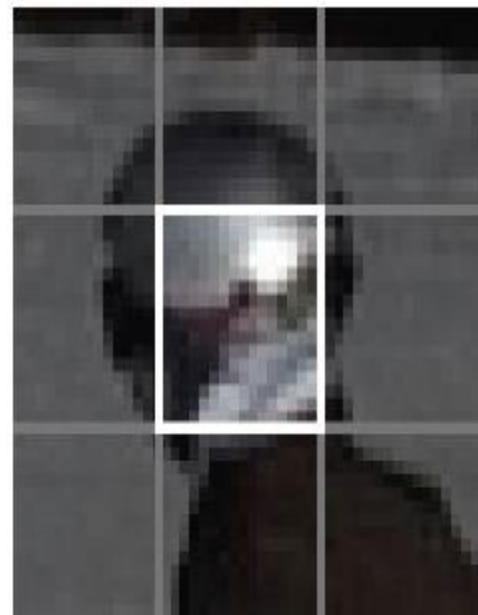
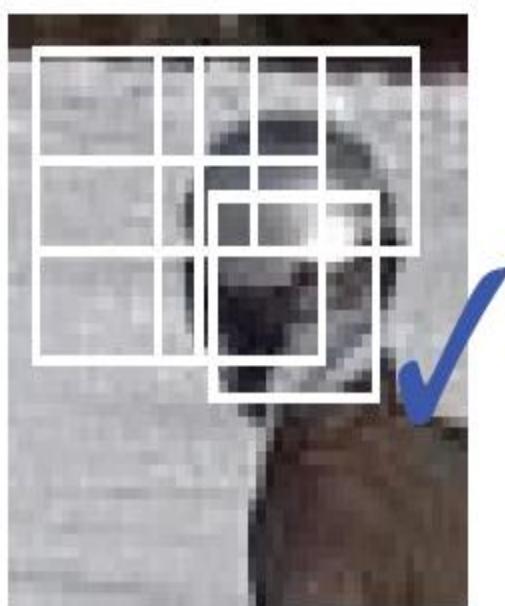
# H.261 – căutarea vectorilor de mișcare

- De unde “vine” macroblocul din cadrul anterior

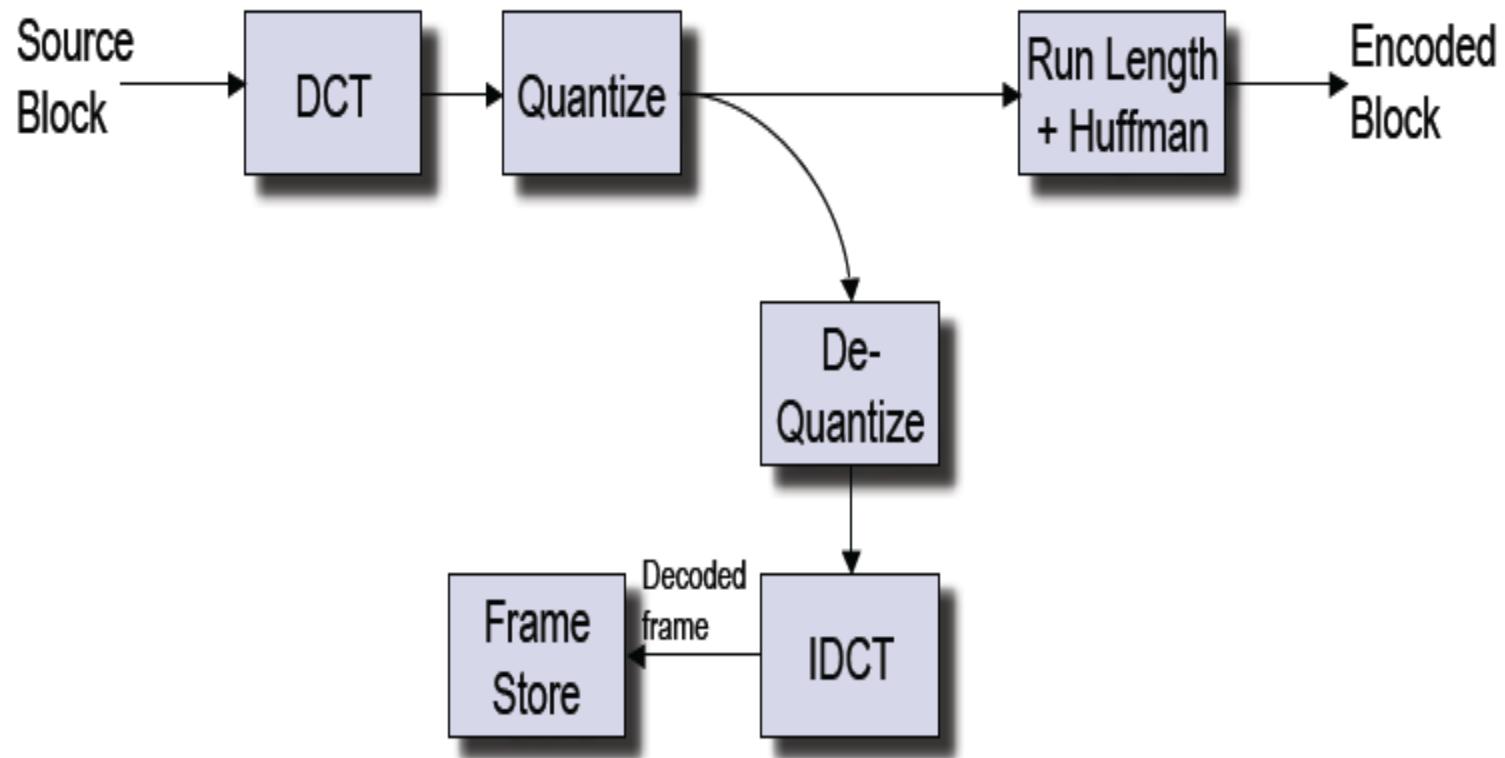


# H.261 – căutarea vectorilor de mișcare

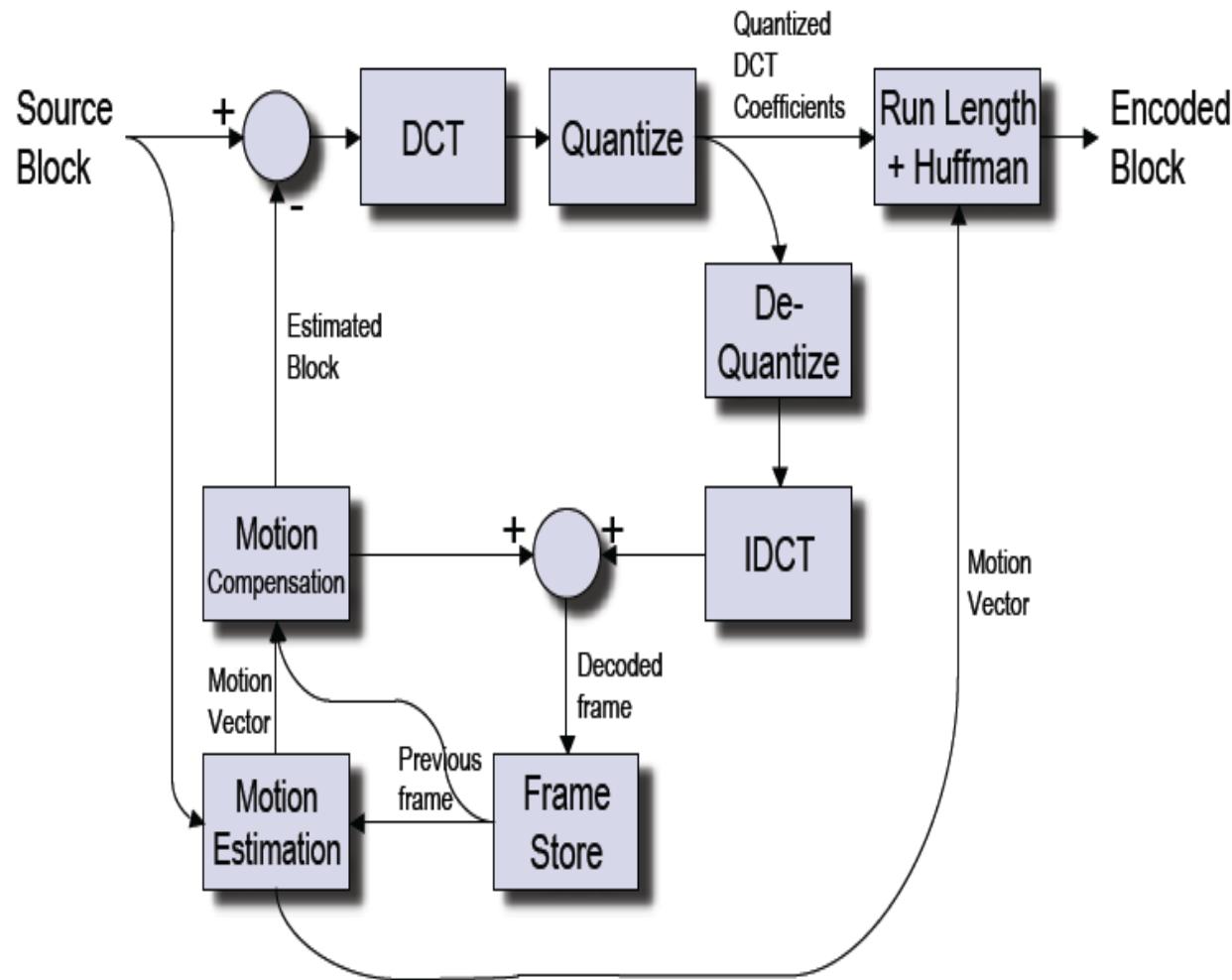
- Se caută  $\pm 15$  pixeli pe cele două direcții
- Procesul cel mai complex – standardul nu specifică algoritmul



# H.261 – codorul intra-cadru



# H.261 – codorul inter-cadru



# H.263

- Asemănător cu H.261
- Performanțe în compresie ridicate – 30kbps
- Flexibilitate în utilizare – înlocuit în multe aplicații

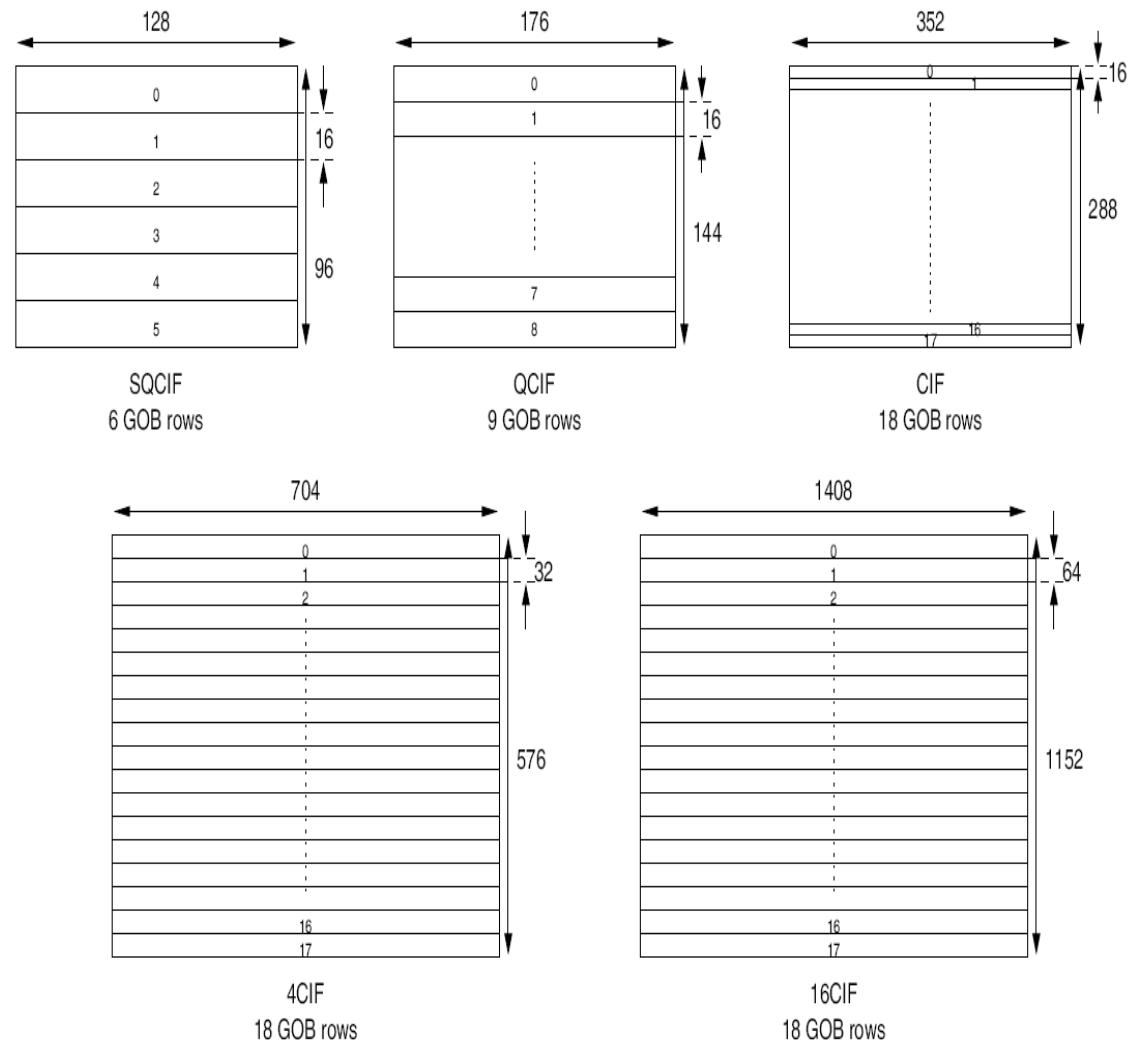
Picture Format	Lumin. Cols.	Lumin. Lines	H.261 Support	H.263 Support	Uncompressed Bitrate (Mbit/s)			
					10 Frame/s		30 Frame/s	
					Gray	Color	Gray	Color
SQCIF	128	96	No	Yes	1.0	1.5	3.0	4.4
QCIF	176	144	Yes	Yes	2.0	3.0	6.1	9.1
CIF	352	288	Optional	Optional	8.1	12.2	24.3	36.5
4CIF	704	576	No	Optional	32.4	48.7	97.3	146.0
16CIF	1408	1152	No	Optional	129.8	194.6	389.3	583.9

# H.263

- Prima versiune H.263 – 1995 – 4 modele de codare opționale
- Vectorii de mișcare se calculează pentru precizie  $\frac{1}{2}$  pixel față de 1 pixel la H.261
- Versiunea v2 – 1998 / H.263+ sau H.263.++
  - Se adaugă alte modele de codare la cele 4
  - + suportă anumite codecuri
  - ++ suportă toate codecurile

# H.263

- Un macrobloc format din 4 blocuri de Y și 2 de croma – 8x8 pixeli
- GOB –
  - tot rândul de macroblocuri pentru SQCIF, QCIF, CIF
  - 2 rânduri de macroblocuri pentru 4CIF – 32 pixeli
  - 4 rânduri de macroblocuri pentru 16CIF – 64 pixeli
- Structura asemănătoare H.261
  - Imagine
  - GOB
  - Macroblock
  - Bloc – 4 luminanță, 2 croma



# H.264

- Dezvoltat de ISO/IEC MPEG + ITU-VCEG
- Scop
  - Îmbunătățirea eficienței codării
  - Suport pentru aplicații speciale: videoconferință, stocare DVD, video broadcasting, video streaming
  - Fiabilitate

# H.264

- 2001 – ITU demarează 2 proiecte
  - Noul H.263 (versiunea a 2-a a H.263)
  - Un nou standard H.26L
    - Aprobat în 2003 cu modificări în 2004
    - Mai multe nume
      - H.264 – ITU (numele oficial este AVC – Advanced Video Coding)
      - MPEG-4 part 10 – ISO
- H.264 are o rată de 1.5 Mbps față de 3.5 Mbps la MPEG2

# **RETELELOR NEURONALE ARTIFICIALE**

-

# **RETELELOR NEURONALE CONVOLUTIONALE**

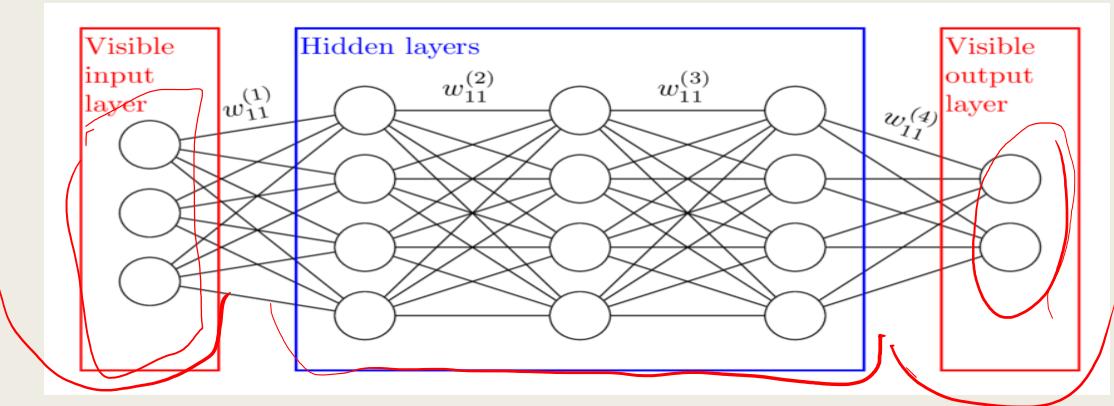
**Sl. Dr. Ing. Camelia FLOREA**  
(Camelia.Florea@com.utcluj.ro)

# Retelelor Neuronale Artificiale

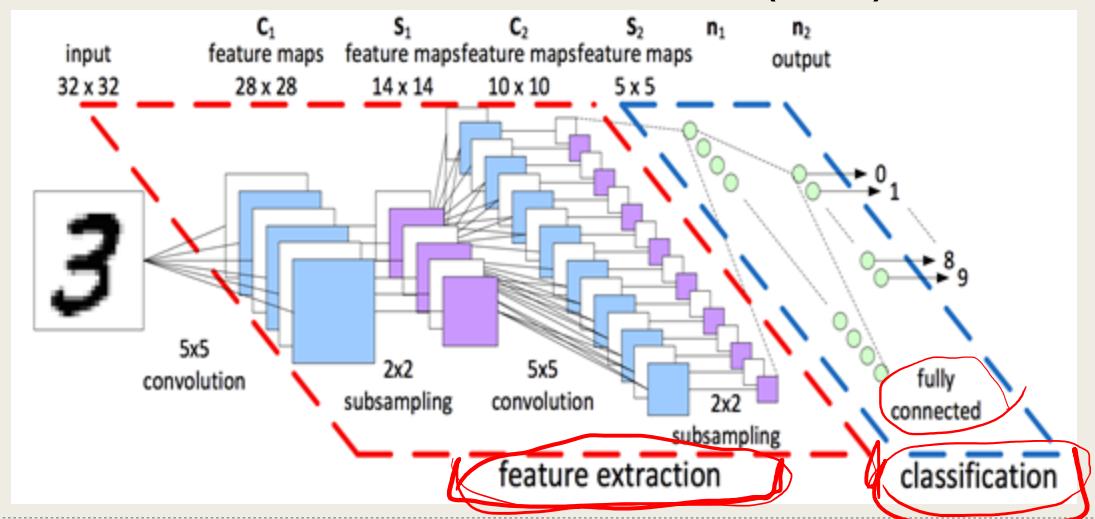
## ■ Caracteristici "împrumutate" de la creier:

- **capacitatea de a învăța**
  - învățare din exemple (antrenare cu seturi mari de date)
- **capacitatea de a generaliza**
  - pot da răspunsuri corecte pentru intrări ușor diferite de cele cu care au fost antrenate
- **capacitatea de a sintetiza**
  - pot da răspunsuri corecte pentru intrări afectate de zgomot/ imprecise/ partiale

Artificial Neural Networks (ANN)



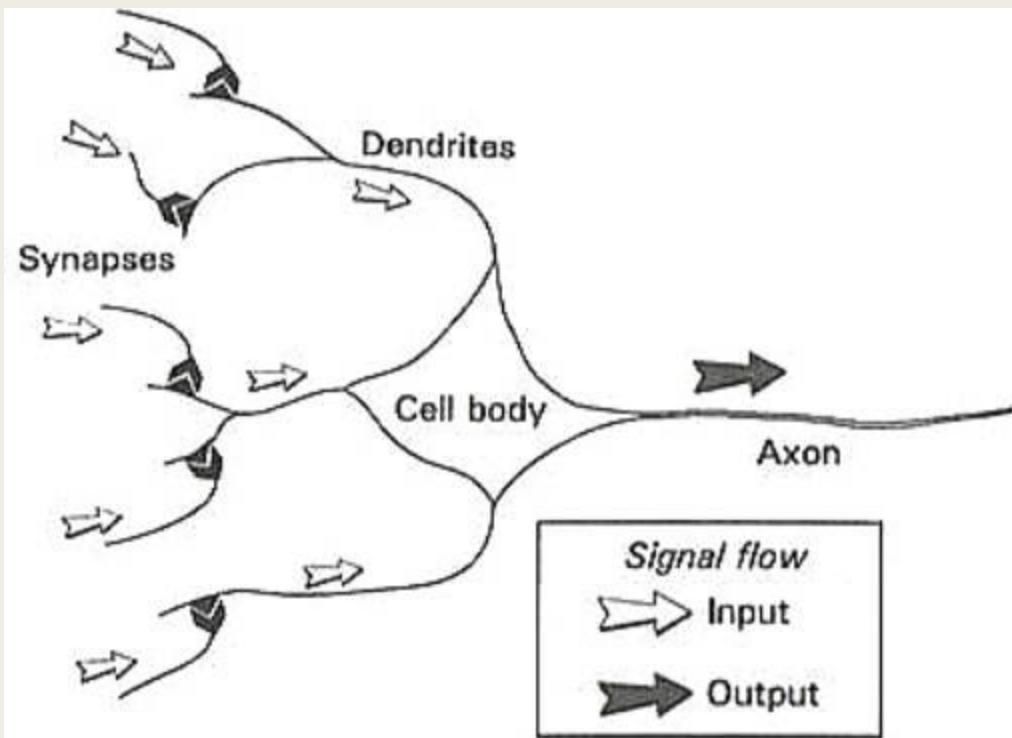
Convolutional Neural Networks (CNN)



# Artificial Neuron

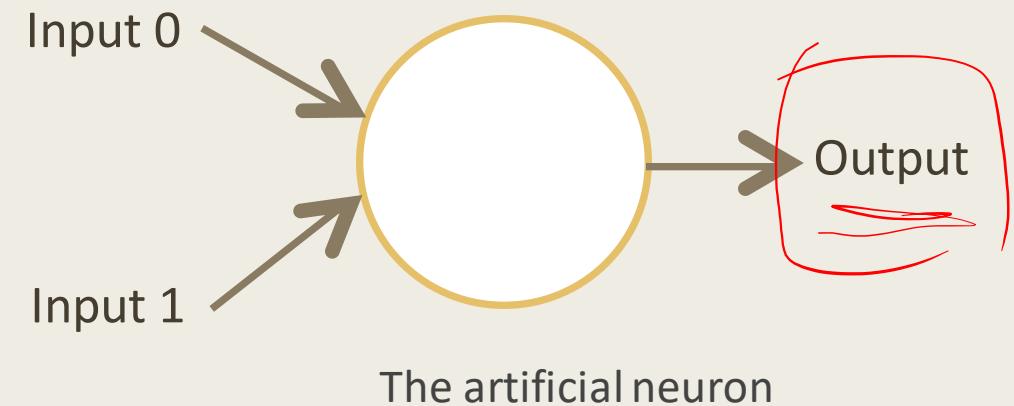
Retelele Neuronale Artificiale – au bazele in biologie

Un neuron biologic:



The biological neuron

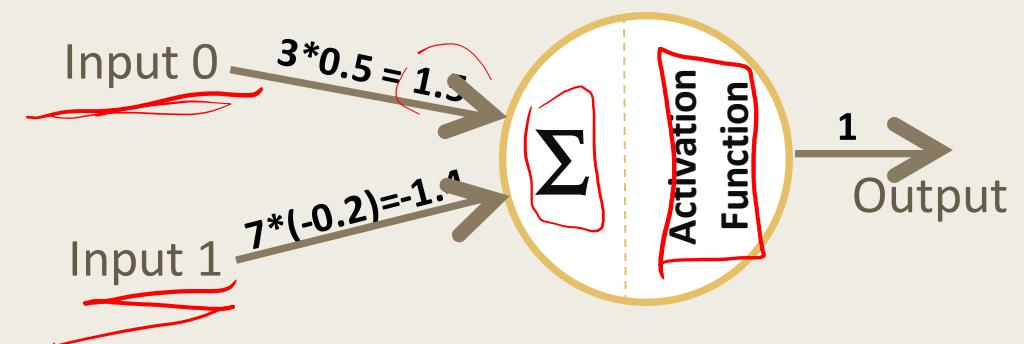
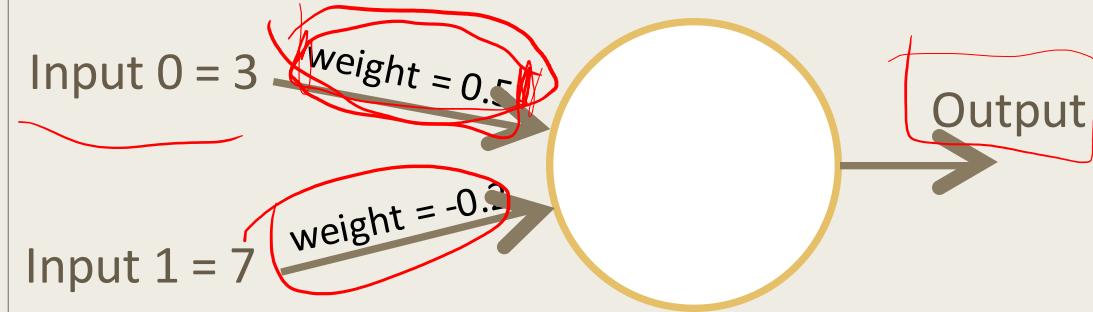
Un neuron artificial:



- Modelul simplu este cunoscut ca si perceptron.
  - are intrari si iesiri similar unui neuron biologic!

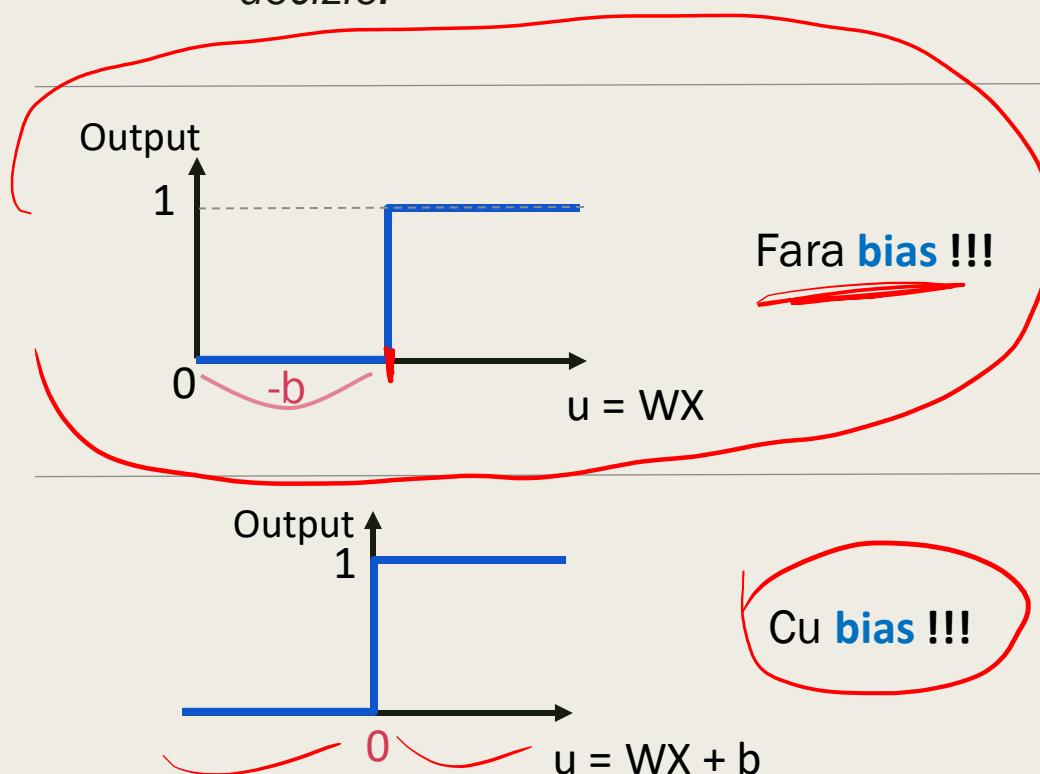
# Exemplu simplu de functionare

- Consideram un perceptron simplu
  - cu două intrari și o ieșire
  - fiecare intrare are o pondere
- Intrările sunt valorile trasaturilor
  - Se vor înmulți cu ponderile aferente
    - ponderile initial vor lua valori aleatoare
- Rezultatul înmulțirilor insumate vor trece printr-o funcție de activare
  - Există mai multe variante a funcției de activare (vom dicuta mai încolo în curs)! – aici considerăm o funcție de activare simplă:
    - Dacă suma intrărilor este pozitivă - se returnează 1
    - Dacă suma intrărilor este negativă - se returnează 0
  - În cazul acesta  $1.5 + (-1.4) = 0.1$ 
    - $\Rightarrow$  la ieșire avem 1

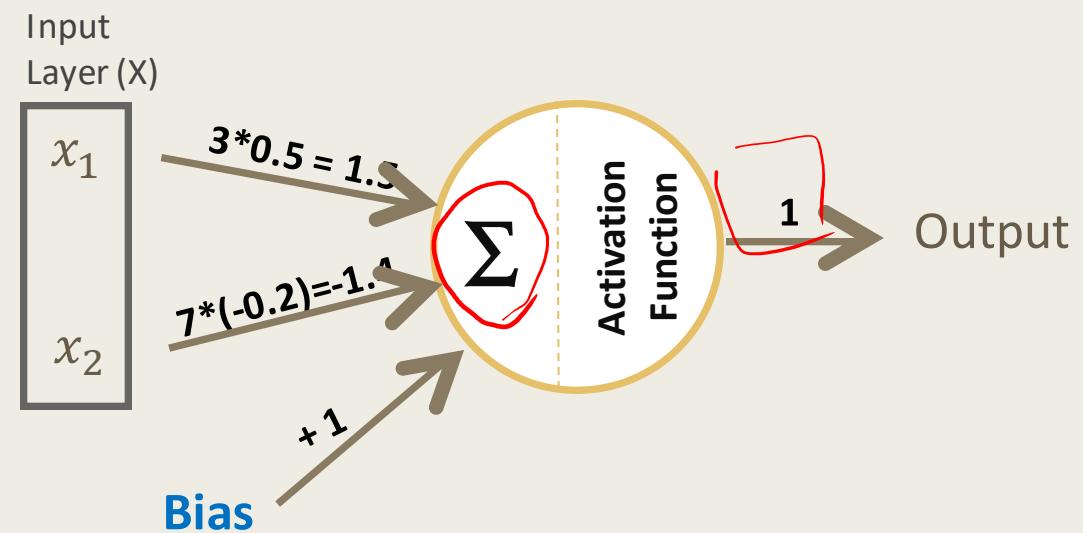


# Termenul bias

- Termenul **bias** (în engleză) este cunoscut drept:
  - *deplasare sau*
  - *distanța față de origine a suprafeței de decizie.*



- În acest caz (exemplu dat),
  - conderam  $bias = 1$



# Formulare matematica

- Formulare/descriere matematica perceptron

- 1 perceptron (neuron)
- $M$  intrari (generalizare, cu mai mult de doua intrari)
- 1 ieșire

$$u = \sum_{i=0}^M w_i x_i + b$$

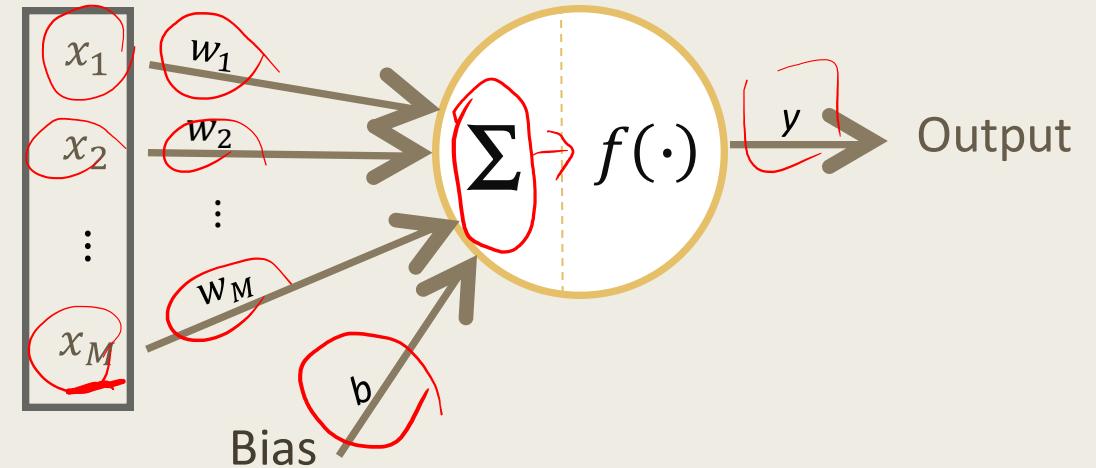
$$y = f(u)$$

Forma matricială:

$$y = f(WX + b)$$

$$W = [w_1 \quad w_2 \quad \dots \quad w_M]$$
$$X = [x_1 \quad x_2 \quad \dots \quad x_M]^T$$

Input Layer (X)



$x_i$  - intrările, ( $X$  vectorul de trasaturi,  
 $x_i$  trasatura  $i$ ,

$M$  - nr. trasaturi/ intrari)

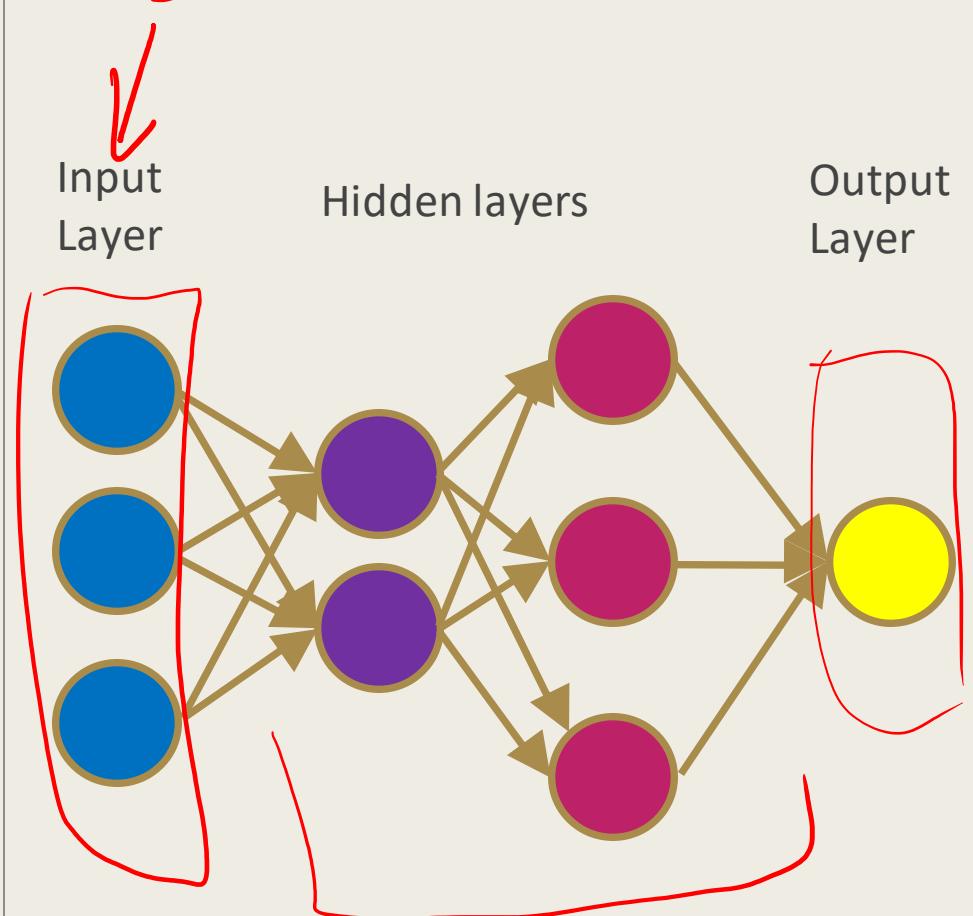
$w_i$  - ponderile sinaptice (weights),  
 $W$  vectorul ponderilor;

$b$  - bias;  $f(\cdot)$  - funcția de activare;  $y$  - ieșirea

# Neural Networks

*Setul de date*

- Multiple Perceptrons Network
  - *conectarea a mai multi perceptroni impreuna*
- Ex.: Input Layer; 2 hidden layers; Output Layer.
- Input Layers
  - *Valori reale – vectorul de trasaturi a setului de date*
- Hidden Layers
  - *Nivele/Layers intre input si output*
  - *3 sau mai multe nivele in “deep network”*
- Output Layer
  - *Estimarea finala a rezultatului*



# Formulare matematica MPN (Multiple Perceptrons Network)

- Formulare/descriere matematica perceptron

- N neuroni (multi perceptroni)
- M intrari
- C iesiri (dat de numarul de clase)

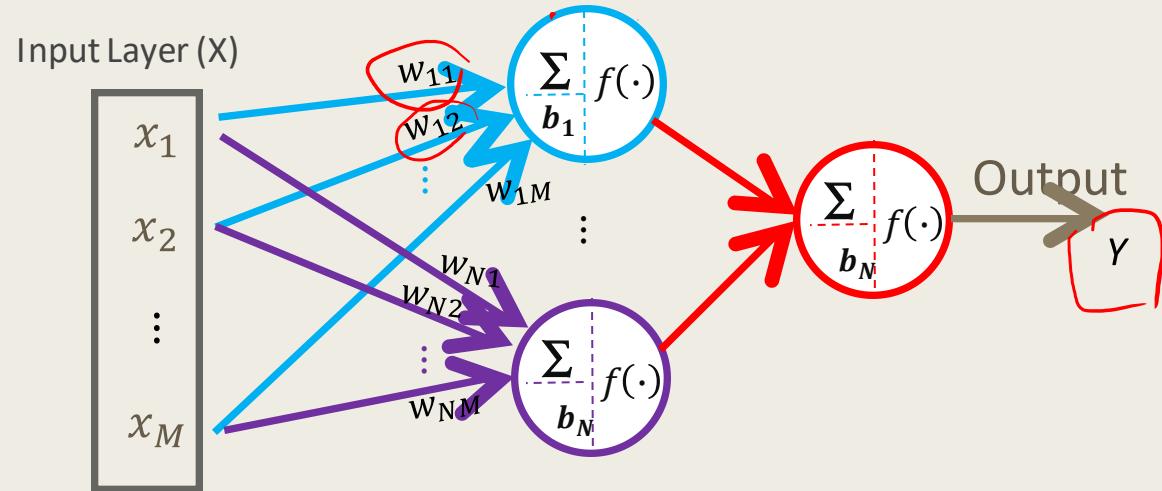
Pentru un perceptron:  $y = f(WX + b)$

$$W = [w_1 \quad w_2 \quad \dots \quad w_M]$$

$$X = [x_1 \quad x_2 \quad \dots \quad x_M]^T$$

$x_i$  - intrarile, (X vectorul de trasaturi,  
 $x_i$  trasatura  $i$ ,  
 $M$  - nr. trasaturi/ intrari)

$w_i$  - ponderile sinaptice (weights),  
 W vectorul ponderilor;  
 $b$  - bias;  $f(\cdot)$  - functia de activare;  $y$  - iesirea



$$Y = f(WX + B)$$

Mai multi perceptroni, relatia se extinde:  
 -  $b$  valoare  $\rightarrow$  B vector  
 -  $W$  1D  $\rightarrow$  W 2D  
 -  $y$  valoare  $\rightarrow$  Y vector  
**unde  $N$  - nr. de neuroni**

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1M} \\ \dots & \dots & \dots & \dots \\ w_{N1} & w_{N2} & \dots & w_{NM} \end{bmatrix}$$

$$X = [x_1 \quad x_2 \quad \dots \quad x_M]^T$$

$$B = [b_1 \quad b_2 \quad \dots \quad b_N]$$

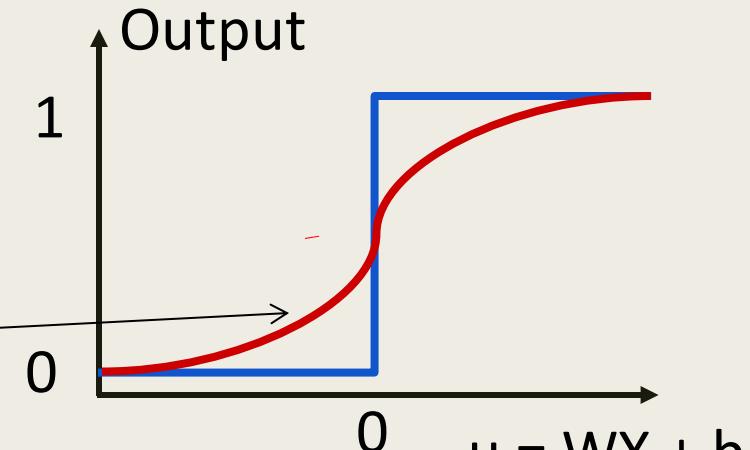
# Functii de activare (Activation function)

- Functii de activare

- O functie simpla care are la iesire 0 sau 1  
(reprezentata in albastru in graphic)
    - acest tip de functie nu reflecta schimbarile usoare

- Functia sigmoid – este o functie mai dinamica  
(reprezentata in rosu in graphic)

$$f(u) = \frac{1}{1 + e^{-u}}$$

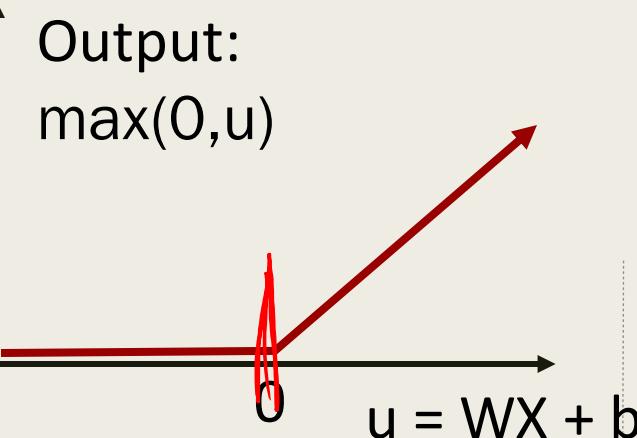


- Hyperbolic Tangent:  $\tanh(u)$

$$\begin{aligned}\cosh x &= \frac{e^x + e^{-x}}{2} \\ \sinh x &= \frac{e^x - e^{-x}}{2} \\ \tanh x &= \frac{\sinh x}{\cosh x}\end{aligned}$$

- Rectified Linear Unit (ReLU)

- este o functie relativ simpla:  $\max(0, u)$



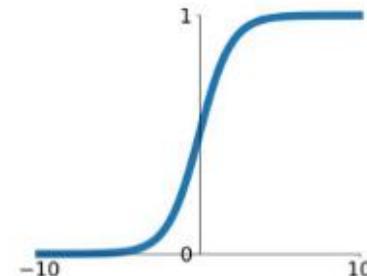
- Modificarea functiei de activare poate fi benefica (depinde mult de task)

- ReLu si Tanh – au cele mai bune performante

# Functii de activare

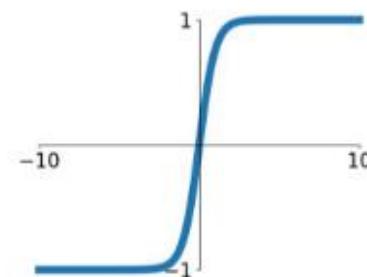
**Sigmoid**

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



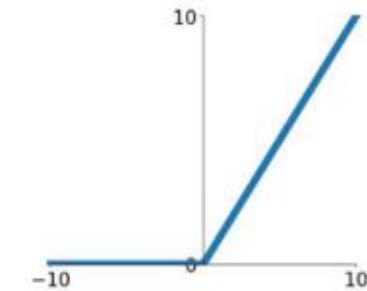
**tanh**

$$\tanh(x)$$



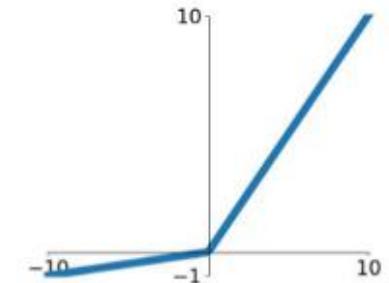
**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

$$\max(0.1x, x)$$

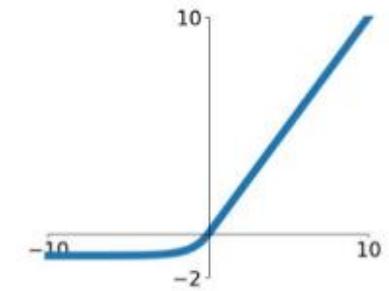


**Maxout**

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



# Instruirea Retea Neuronala Artificiala

- Instruirea/antrenarea are loc prin
  - *ajustarea succesiva a ponderilor pentru a reduce diferența dintre iesirile reale (dorite) și cele prezise pentru toate datele de antrenare.*
- Cum putem evalua performanta unui neuron?
  - Se utilizeaza *functia Cost*
    - pentru a masura cat de departe suntem de valoarea estimată/preconizată
- Se vor utiliza urmatoarele variabile:
  - *d* pentru a reprezenta valoarea reală/dorită
  - *y* pentru a reprezenta predictia neuronului
- In termini cu ponderi si bias:  $w^*x + b = u$   
*u este trecut prin functia de activare  $f(u) = y$*
- Scopul este de a minimiza *funcția de cost*
  - *prin modificarea ponderilor și a bias-ului rețelei.*
- **Quadratic Cost:**  $C = \sum(d-y)^2 / N$ 
  - se poate observa ca erorile mai mari sunt mai evidente datorita ridicarii la patrat
  - dar, din pacare acest calcul poate duce la o incetinire destul de mare a procesului de invatare
- **Cross Entropy:**
$$C = (-1/N) \sum (d \cdot \ln(y) + (1-d) \cdot \ln(1-y))$$
  - aceasta functie Cost permite invatare mai rapida
  - cu cat este mai mare diferența, cu atat mai repede poate invata neuronal

# Instruirea Retea Neuronala Artificiala

- Algoritm de instruire = modul în care se modifică ponderile, proces iterativ,

- la momentul  $t+1$ , actualizare pondere:

$$w_{ij}(t + 1) = w_{ij}(t) + \Delta w_{ij}(t)$$

$$\Delta w_{ij}(t) = -\mu \nabla C(t) = -\mu \frac{\partial C(t)}{\partial w_{ij}(t)} = -(d - y)f'(u)x_i$$

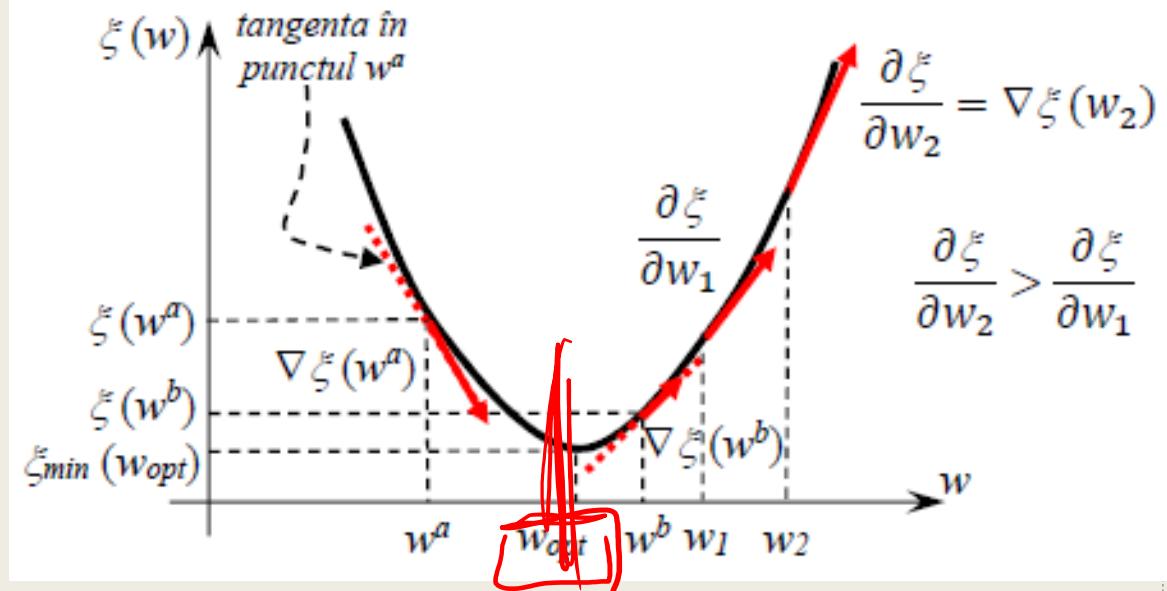
- unde:

- $\Delta w_{ij}(t)$  - algoritm de instruire
    - i – neuron de la care “pleacă” ponderea,
    - j – neuronul spre care “vine” ponderea
    - t – momentul de timp
    - $\mu$  – pas de instruire (subunitar)

- daca  $\nabla C(t) > 0$ ,  $w_{ij}$  trebuie sa scada
    - daca  $\nabla C(t) < 0$ ,  $w_{ij}$  trebuie sa creasca

- Optimizare prin gasirea minimului

- cea mai mică eroare posibilă – la vectorul optim de ponderi
  - metoda: gradientului descendente (derivata erorii in raport cu ponderea)



# Backpropagation

- Este utilizat pentru a calcula contributia fiecarui neuron la eroare - dupa ce un lot (batch) de date este procesat
- **Backpropagation** – ajustarea parametrilor/ ponderilor in intreaga retea
  - eroarea se propaga inapoi (de aici si denumirea algoritmului), de la ultimul strat inspre primul strat.
- Algoritmul Backpropagation implica 2 faze:
  - *faza **Forward**, in care parametrii retelei sunt fixati si semnalul de intrare este propagat prin retea , strat cu strat.*
    - la sfarsitul acestei faze se calculeaza eroarea e
  - *faza **Backward**, in care eroarea e se propaga prin retea inapoi.*
    - in aceasta faza, se modifica parametrii pondere si bias, in scopul minimizarii erorii e

# Pasi Antrenare / Instruire

Setul de date  
→ antrenare  
→ testat  
→ Validare

- Mod de antrenare/ optimizare :

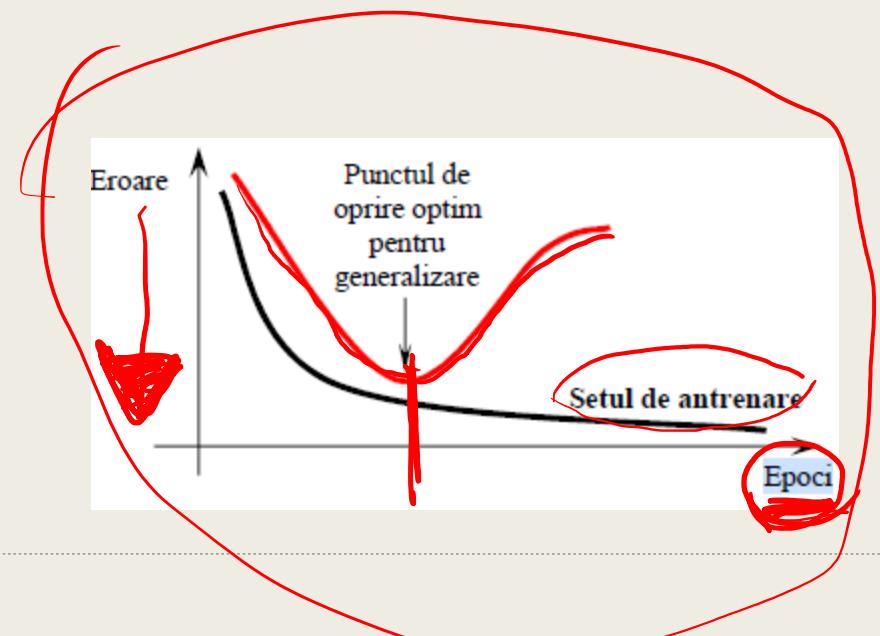
- Se generează un set de ponderi și valori bias, în mod aleator;
  - 1) Se prezintă structurii neuronale intrările X și ieșirile dorite Y;
  - 2) Se calculează ieșirile fiecărui neuron plasat pe fiecare strat neuronal;
  - 3) Se calculează eroarea la nivelul de ieșire
  - 4) Se propagă eroarea înapoi în rețea și se reține factorul delta corespunzător fiecărei ponderi
  - 5) Se aplică ajustarea corespunzătoare fiecărei ponderi
  - 6) Dacă s-a îndeplinit condiția de oprire a algoritmul de antrenare acesta se oprește; dacă nu se reia algoritmul de la pasul 1.

- Condiția de oprire depinde

- de valoarea erorii (a scăzut sub un anumit prag acceptabil)
- și/sau numărul de epoci de antrenare

- De ce poate diferenția rezultatul la fiecare rulare?

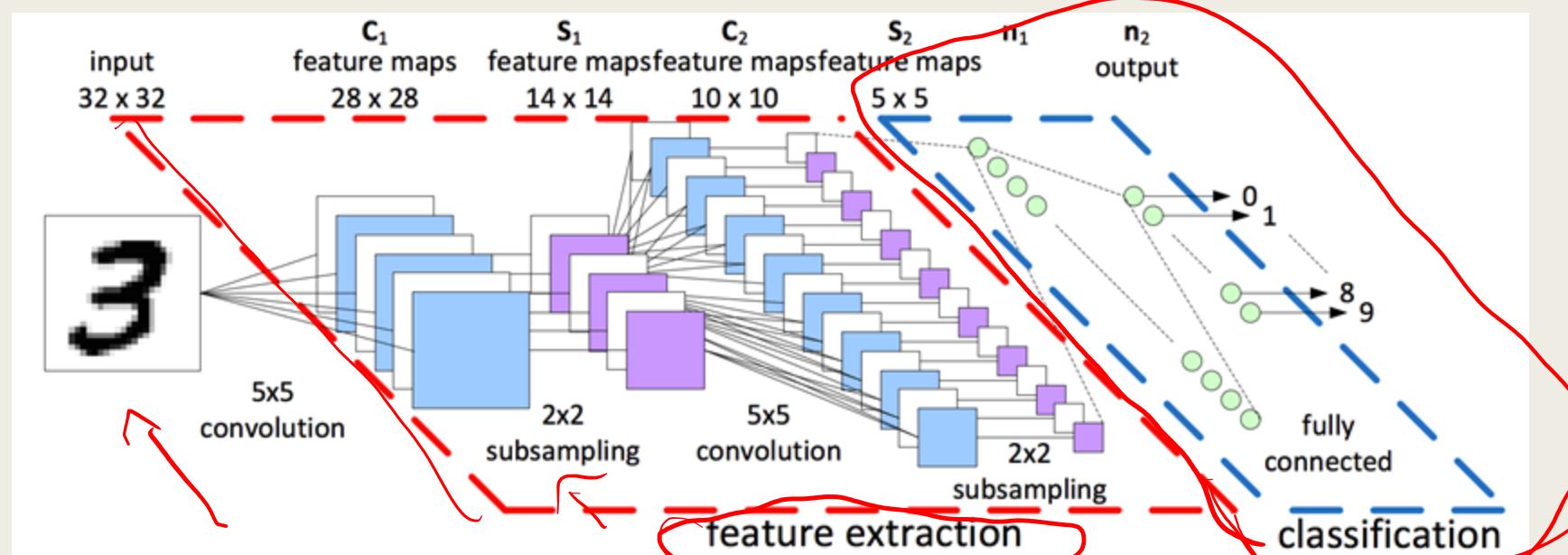
- initial avem o asociere aleatoare a ponderilor  
→ rezultatul poate să difere (in general, usor)



# Convolutional Neural Networks (CNN)

# Convolutional Neural Networks (CNN)

- **Neural Network** - optim daca s-au **extras/definit trasaturile** pentru descrierea datelor
- Direct pe imagini - se recomanda **Convolutional Neural Networks (CNN)**
  - pentru a rezolva *in mod eficient* problemele care datele de tip **imaginile** le pot avea
- In general, cand dicutam de CNN, vom intalni diagrame de genul:



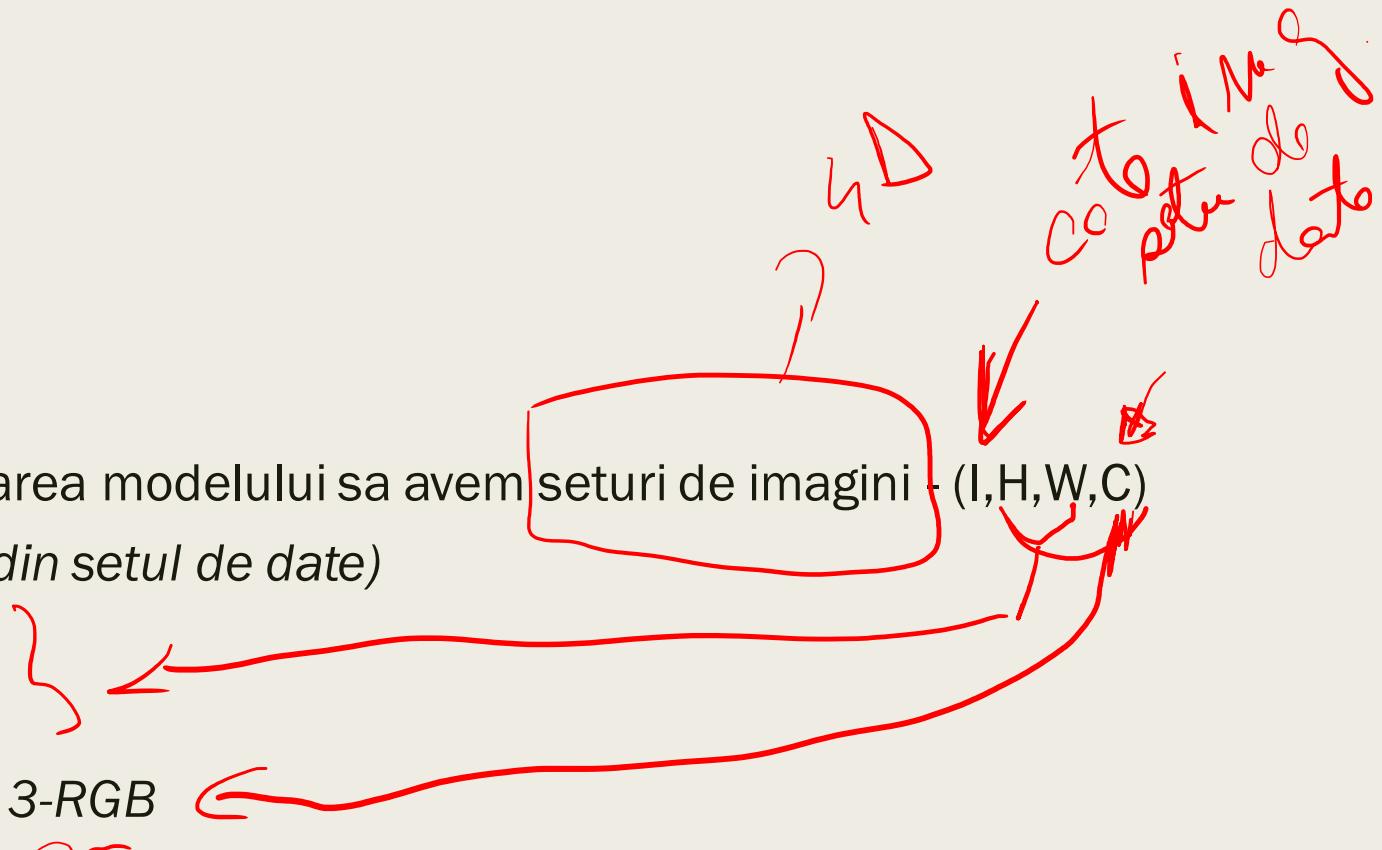
# Tensors

- Tensori - N-Dimensional Arrays:

- Scalar - 3
- Vector -  $[3, 4, 5]$
- Matrix -  $[[3, 4], [5, 6], [7, 8]]$
- Tensor -  $[[[1, 2], [3, 4]], [[5, 6], [7, 8]]]$

- Tensori – convenint pentru ca la intrarea modelului sa avem seturi de imagini -  $(I, H, W, C)$

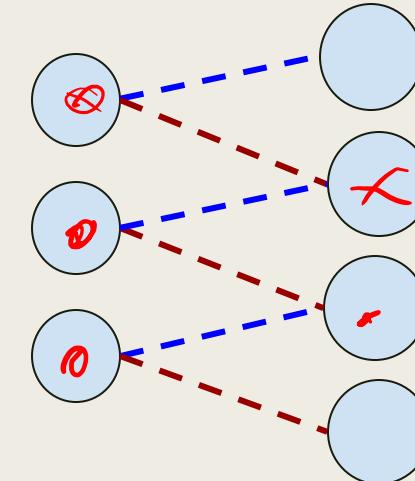
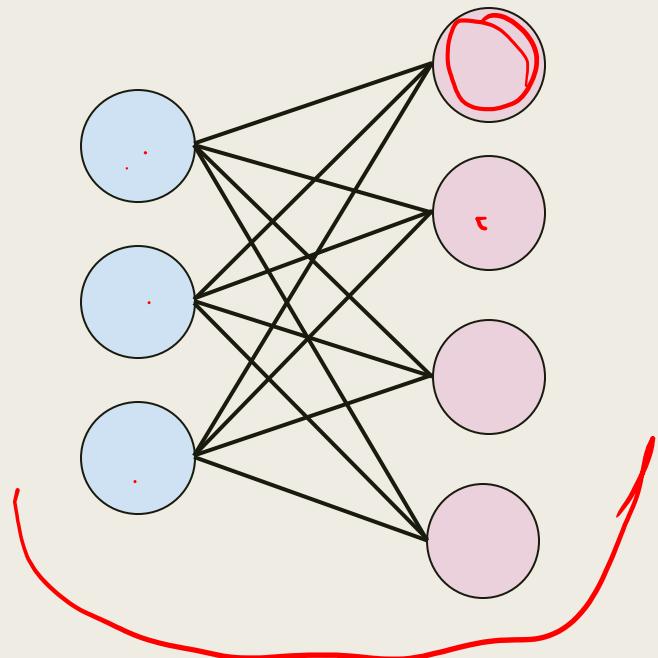
- $I$ : *Images (numarul de imagini din setul de date)*
- $H$ : *Height of Image in Pixels*
- $W$ : *Width of Image in Pixels*
- $C$ : *Color Channels: 1-Grayscale, 3-RGB*



# Densely Connected layer/ Convolutional Layer

## ■ Densely Connected layer/ Convolutional Layer

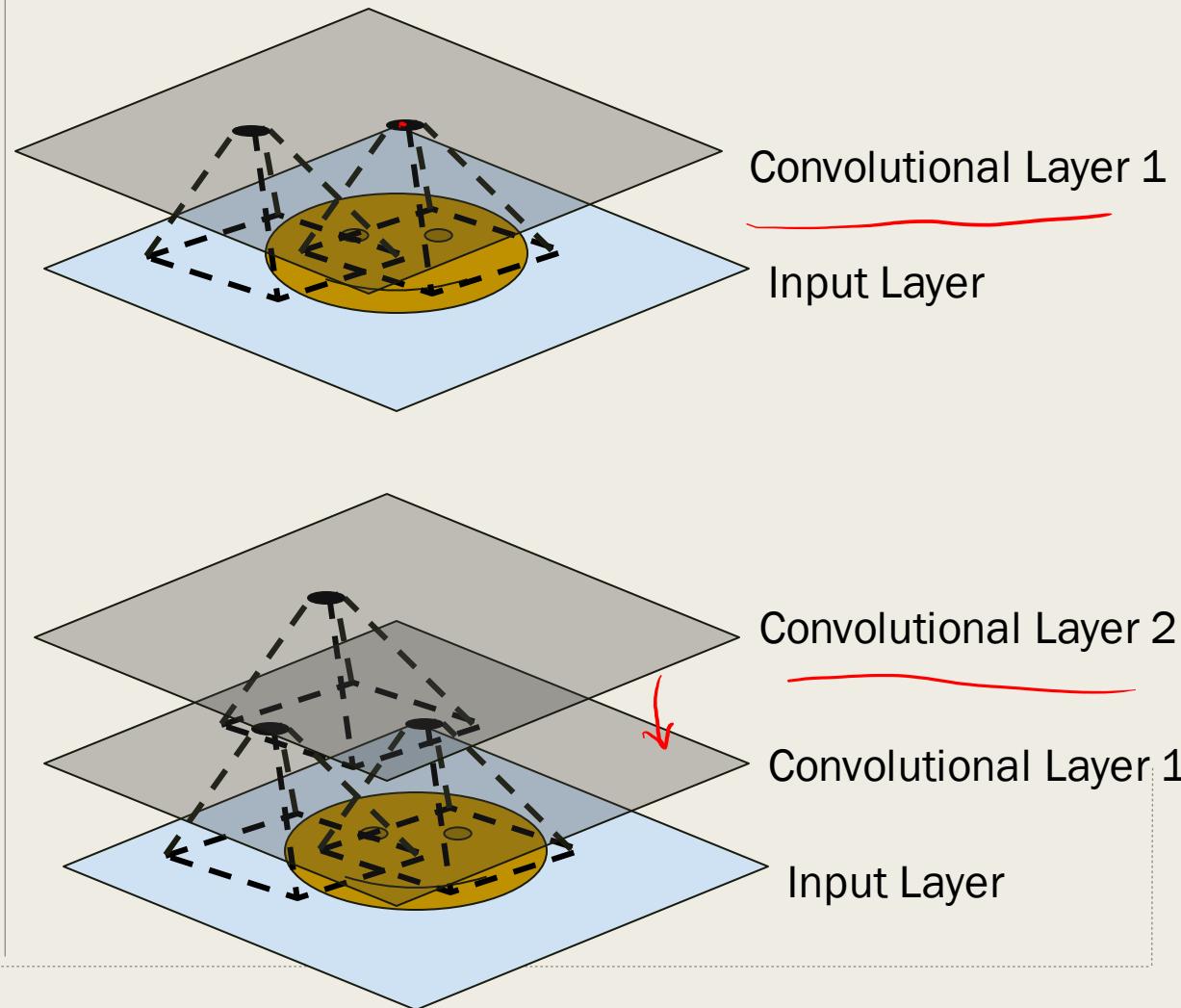
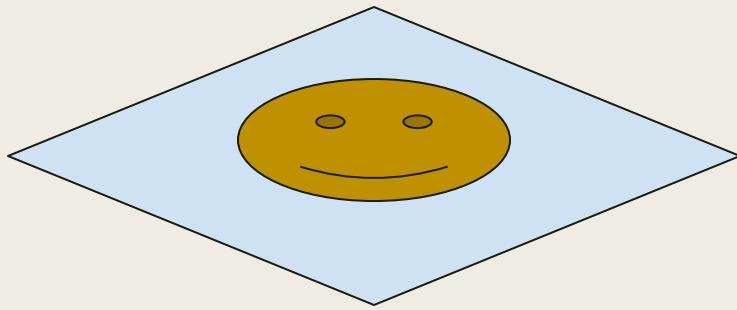
Fiecare neuron este conectat cu un numar mai mic de neuroni vecini.



Care este avantajul CNN versus DNN? Majoritatea imaginilor au o rezolutie de cel putin  $256 \times 256$   
→ prea multi parametrii/ prea multe calcule implicate

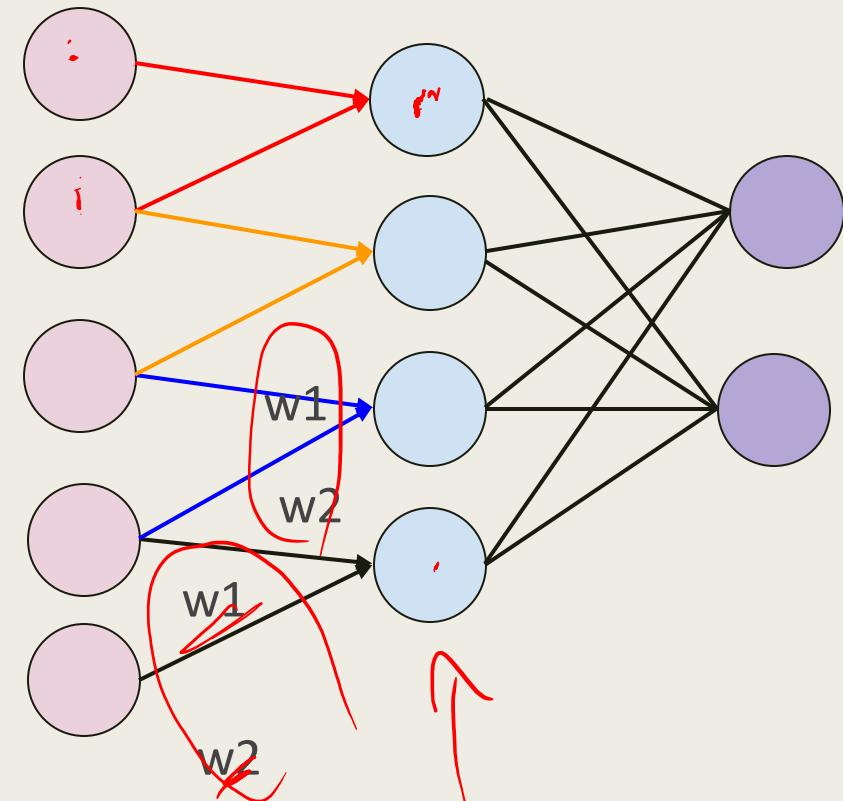
# CNN

- CNN – aplicat pe imagini pentru clasificare
- Input layer – imaginea in sine
- Nivelele convolutionale (Convolutional layers)
  - Sunt conectate doar catre valorile din campul lor/respectiv

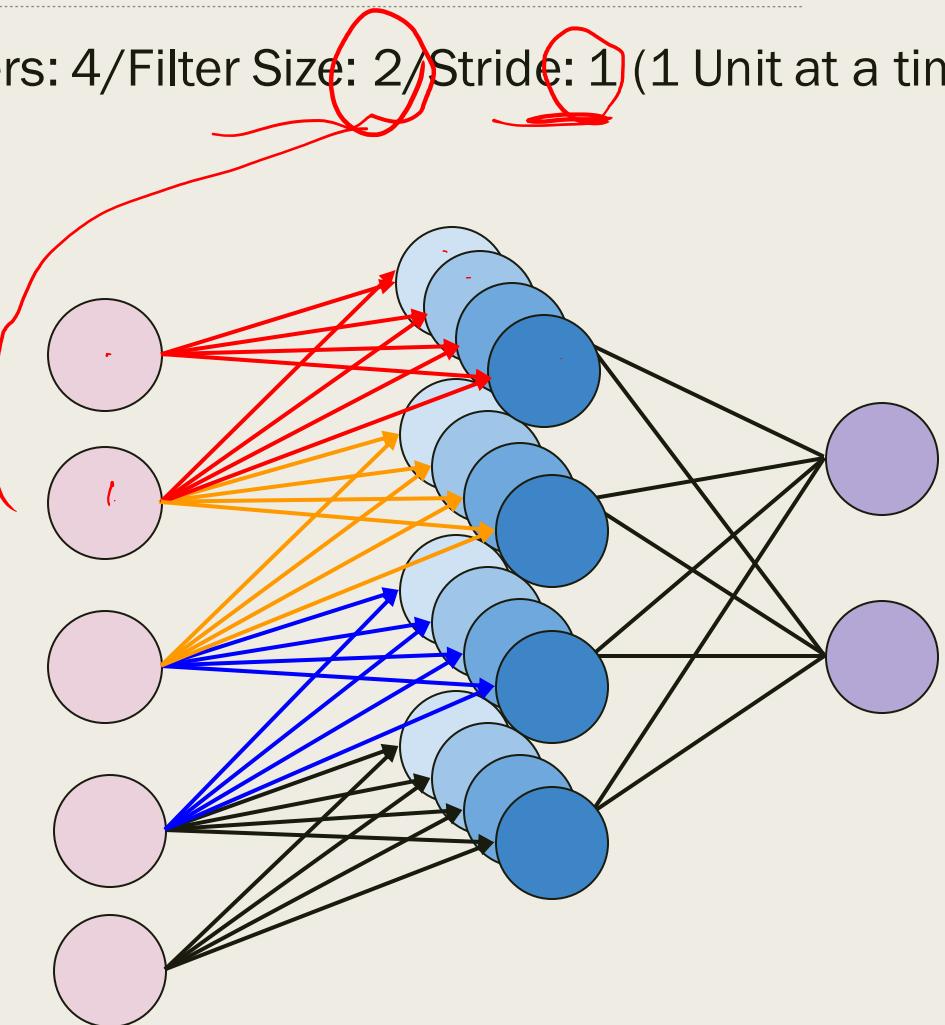


# CNN, 1-D Convolution

Filters: 1/Filter Size: 2/ Stride: 1

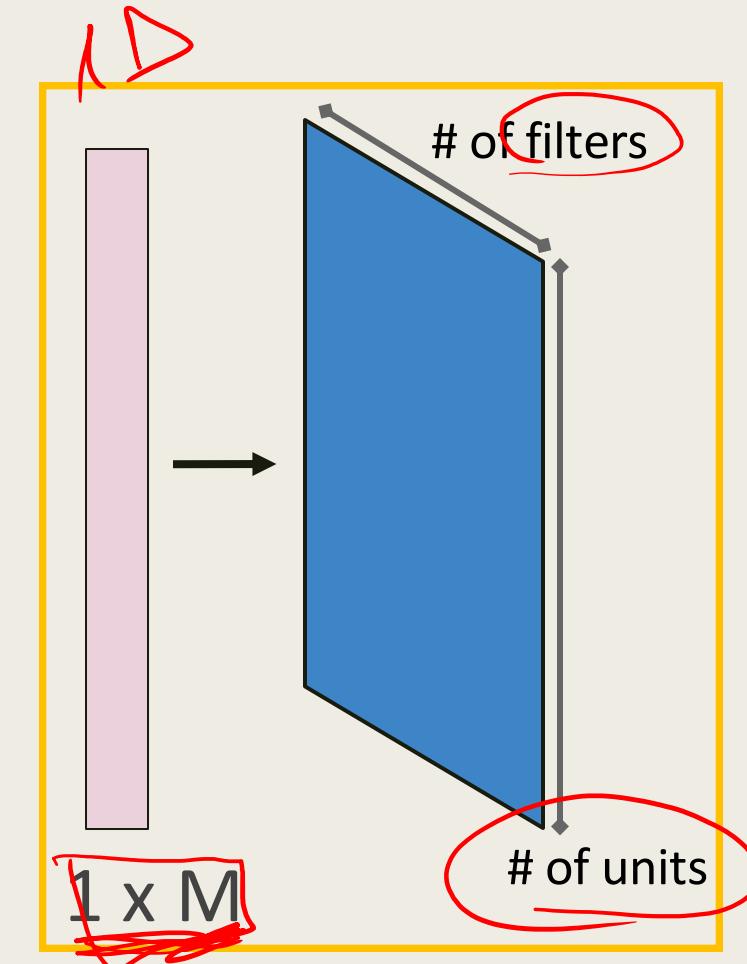
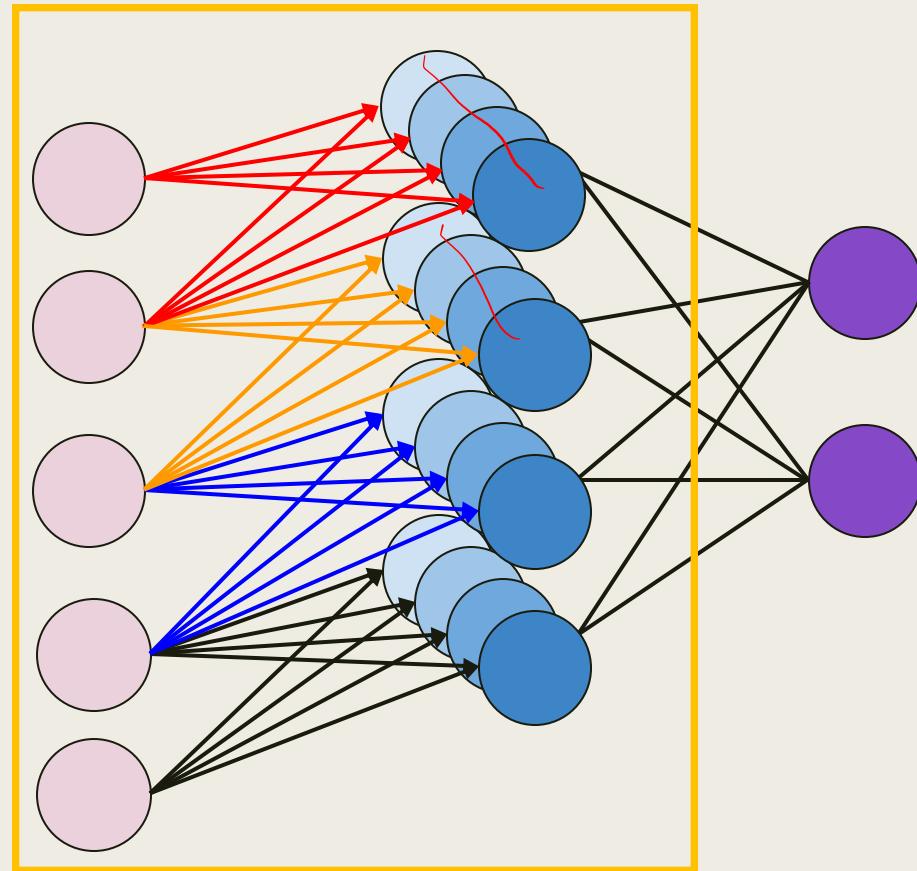


Filters: 4/Filter Size: 2/Stride: 1 (1 Unit at a time)



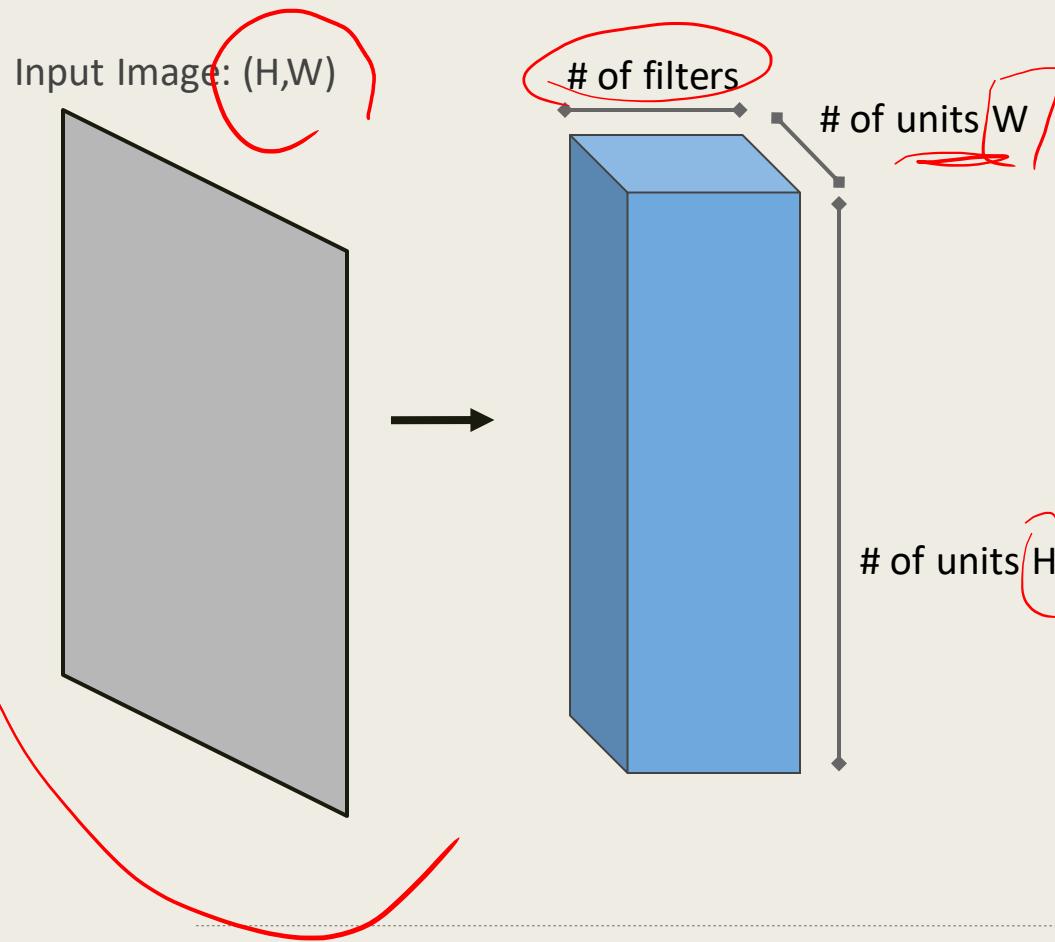
# CNN, 1D Convolution

- Pentru simplificare – vizualizare ca blocuri

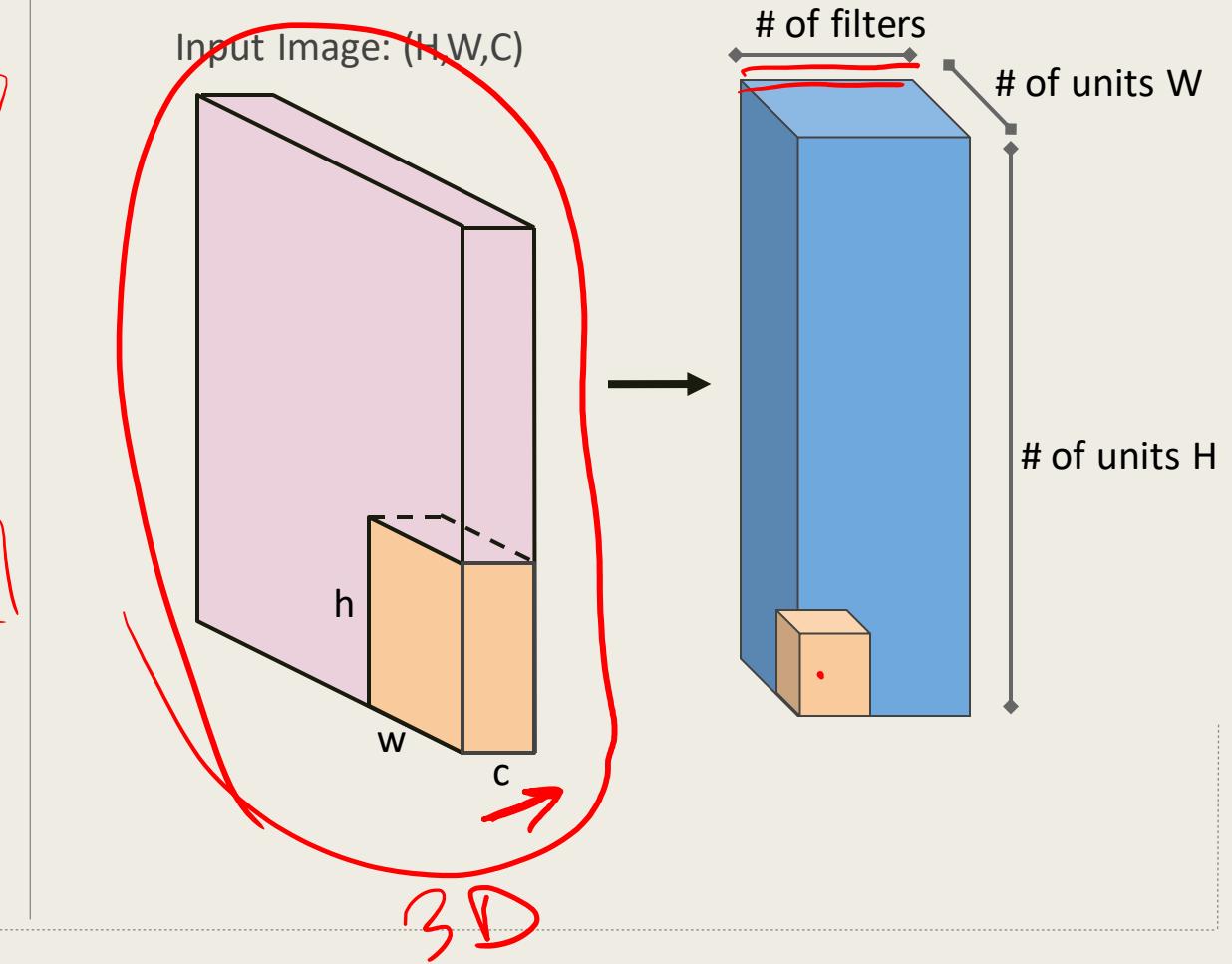


# CNN, 2D Convolution

Imagini pe nivale de gri:



Imagini color:



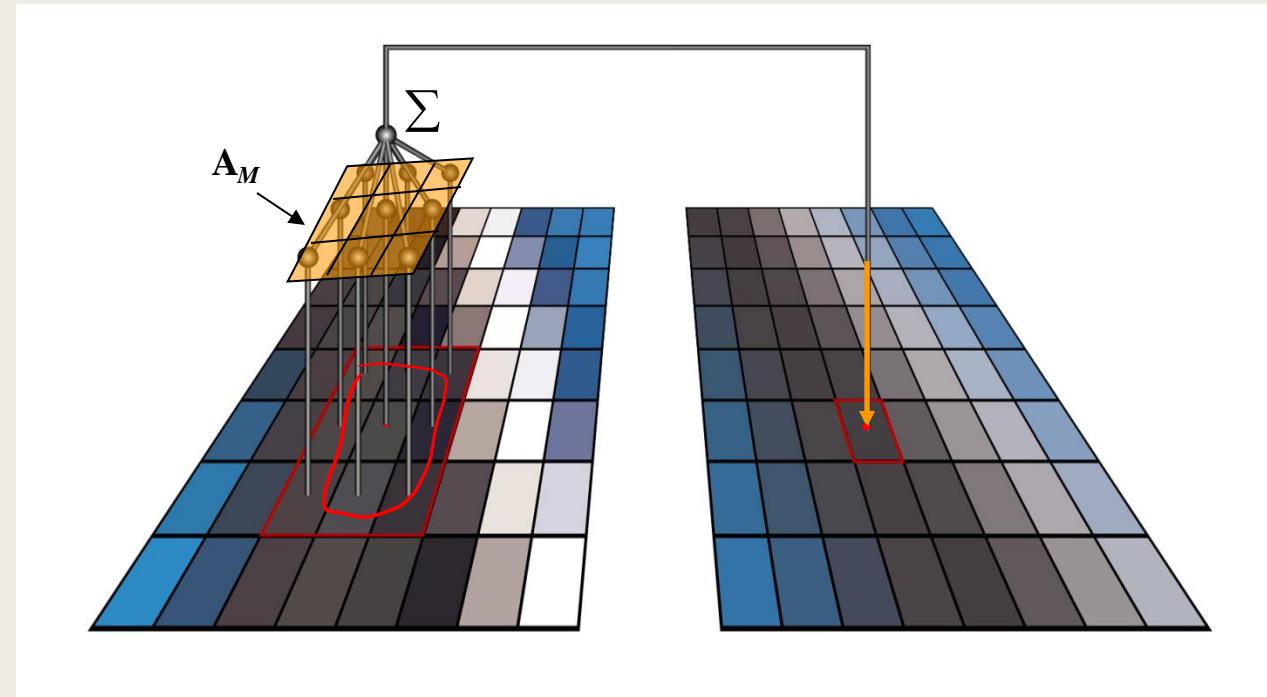
- Convolutie – are un mare avantaj pentru procesarea/analiza imaginilor
  - *pixelii vecini sunt mai corelati intre ei* – efficient in descrierea locala a informatiei/ a variatiei locale
- Fiecare nivel CNN – se poate ‘uita’ la o alta rezolutie a imaginii.
- Avand neuroni legati doar in ‘vecinatate’
  - este *limitat numarul de ponderi (weights) la dimensiunea ferestrei de convolutie*

# Operatia de convolutie

## ■ Formula de convolutie:

$$v(m,n) = \sum \sum_{(k,l) \in W} a(k,l)u(m-k, n-l)$$

$\mathbf{A}[K \times L] = \{a(k,l)\}$  - Masca de convolutie



$$\mathbf{A} = \begin{bmatrix} a(-1,-1) & a(-1,0) & a(-1,1) \\ a(0,-1) & a(0,0) & a(0,1) \\ a(1,-1) & a(1,0) & a(1,1) \end{bmatrix}$$

## ■ Filtrul (de 3x3 în acest exemplu)

- este tensor de ponderi invatat prin backpropagation

0	0	0	0	0	0
0	1	1	1	1	0
0	1	1	-1	1	0
0	1	-1	-1	-1	0
0	1	1	1	1	0
0	0	0	0	0	0

3 by 3 Filter

0	0	0
0	-1	1
0	1	-1

Multiply by filter weights

0x0	0x0	0x0
0x0	1x-1	1x1
0x0	1x1	1x-1

Sum the Result

0	0	0
0	-1	1
0	1	-1

5x5

3x3	x1
-----	----

6x6

5x5

$$(+1) \cdot (+1) + (-1) \cdot (-1) + (+1) \cdot (+1) = 0$$

-2

$$(-1) \cdot (-1) + 1 \cdot 1 + 1 \cdot (-1) + (-1) \cdot (-1) = 2$$

0	2	0
0	2	0
0	2	0

# Stride Distance

Exemplu, stride = 1

0	0	0	0	0	0
0	1	1	1	1	0
0	1	-1	-1	1	0
0	1	-1	-1	1	0
0	1	1	1	1	0
0	0	0	0	0	0

3 by 3 Filter

0	0	0
0	-1	1
0	1	-1

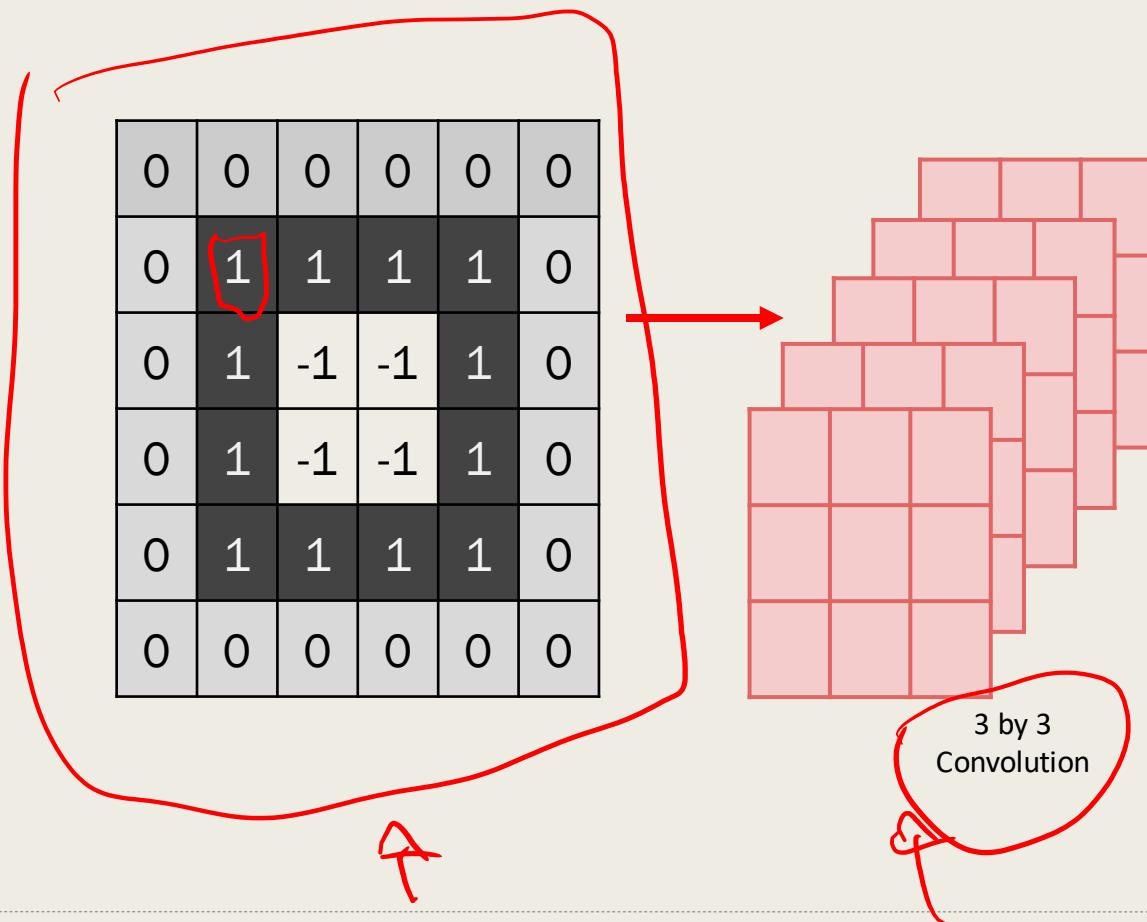
Exemplu, stride = 2

0	0	0	0	0	0
0	1	1	1	1	0
0	1	-1	-1	1	0
0	1	-1	-1	1	0
0	1	1	1	1	0
0	0	0	0	0	0

3 by 3 Filter

0	0	0
0	-1	1
0	1	-1

- Reprezentare – filtre multiple (Multiple Filters)





blur

0,0625	0,125	0,0625
0,125	0,25	0,125
0,0625	0,125	0,0625

outline

-1	-1	-1
-1	8	-1
-1	-1	-1



top sobel

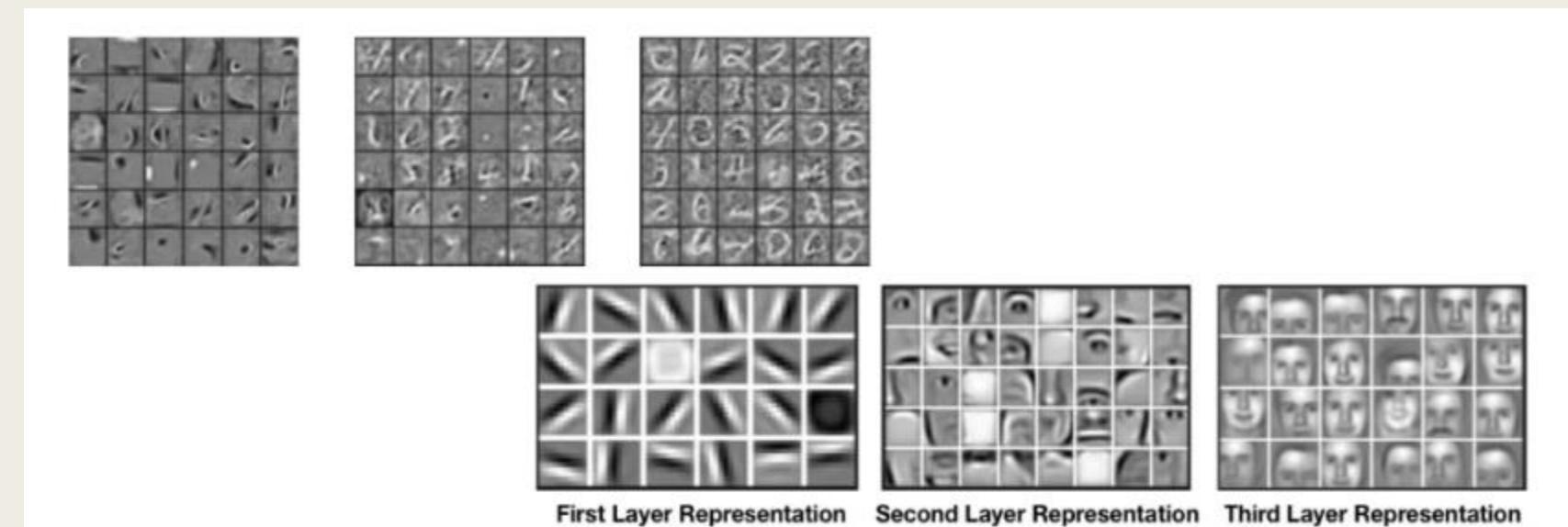
1	2	1
0	0	0
-1	-2	-1

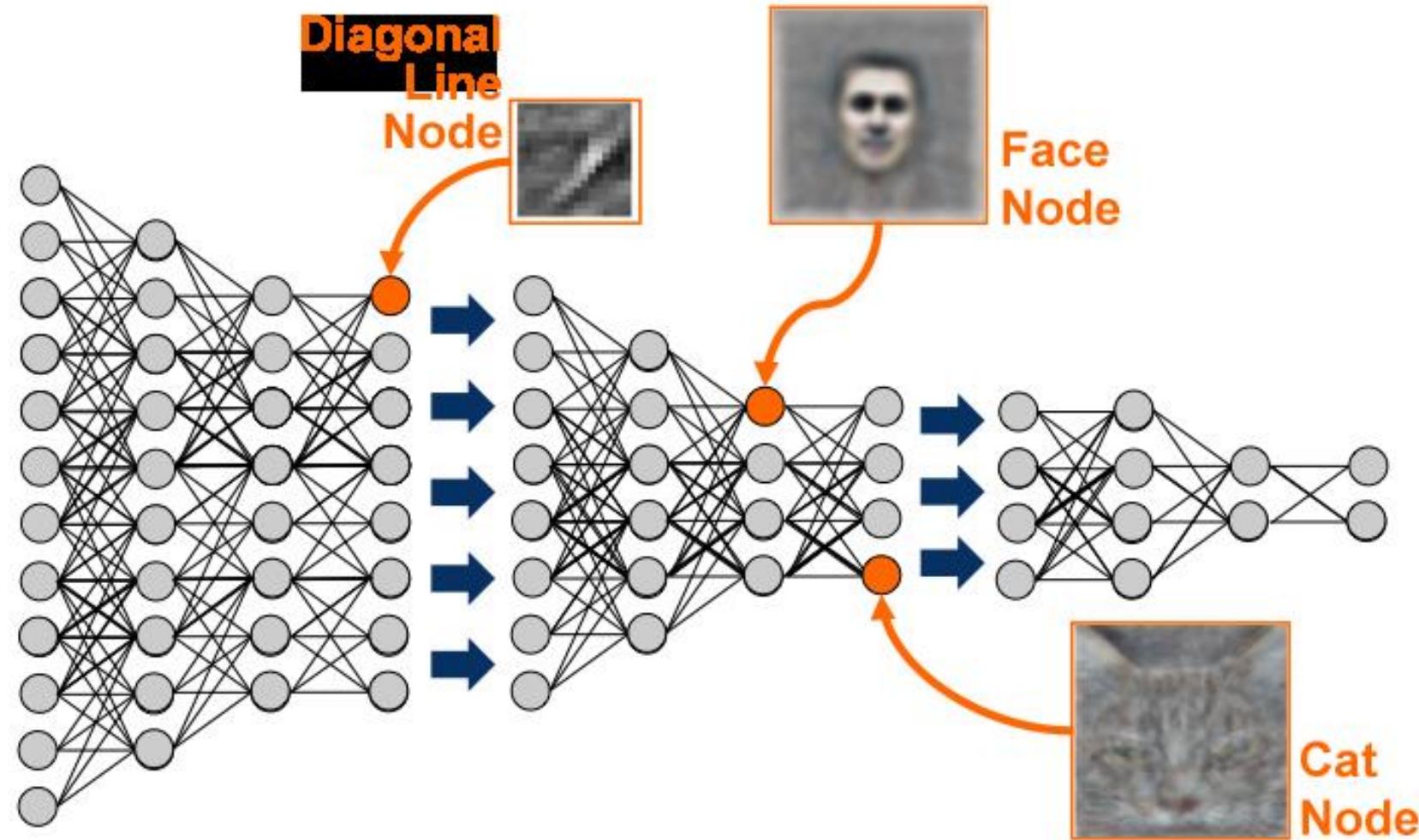
emboss

-2	-1	0
-1	1	1
0	1	2



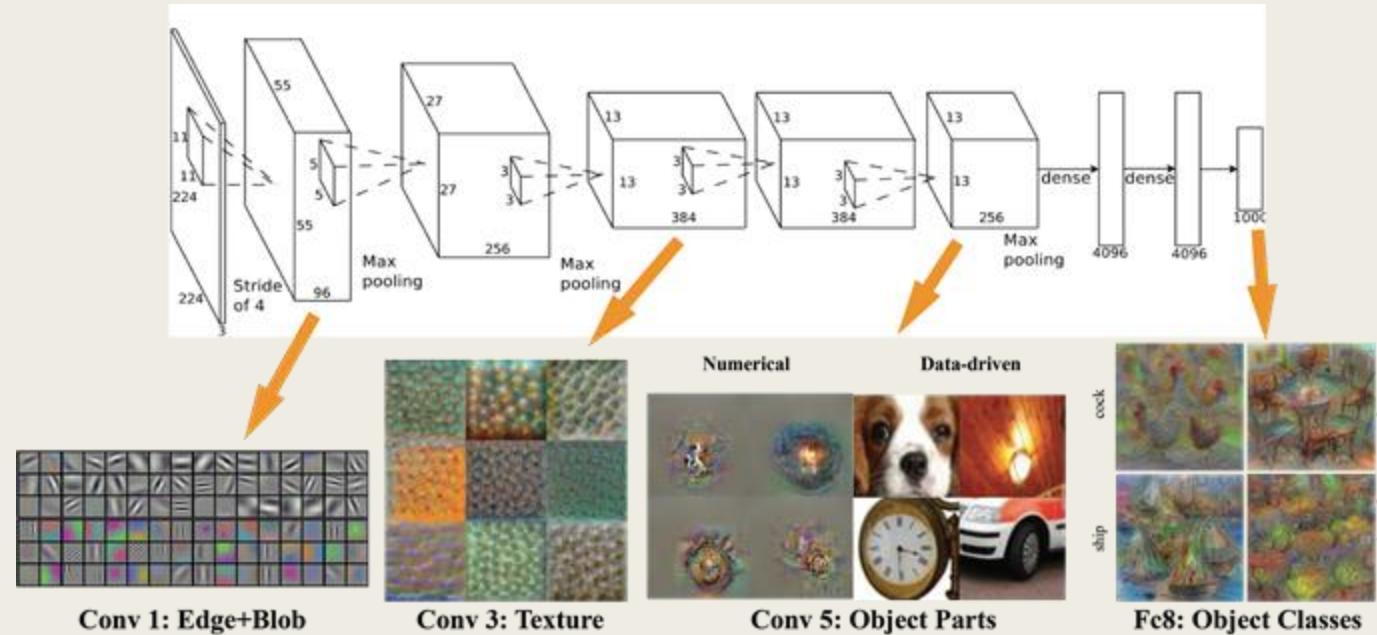
- Filtrele de conoluție odată „învățate” (prin ajustarea ponderilor) asigură extragerea automată a caracteristicilor specifice, începând cu unele elementare, extrem de simple în primul strat conoluțional și ajungând, prin agregarea acestora strat după strat, la caracteristici complexe reprezentând părți din, sau chiar obiecte întregi.
- <http://setosa.io/ev/image-kernels/>
- <http://cs231n.stanford.edu/>





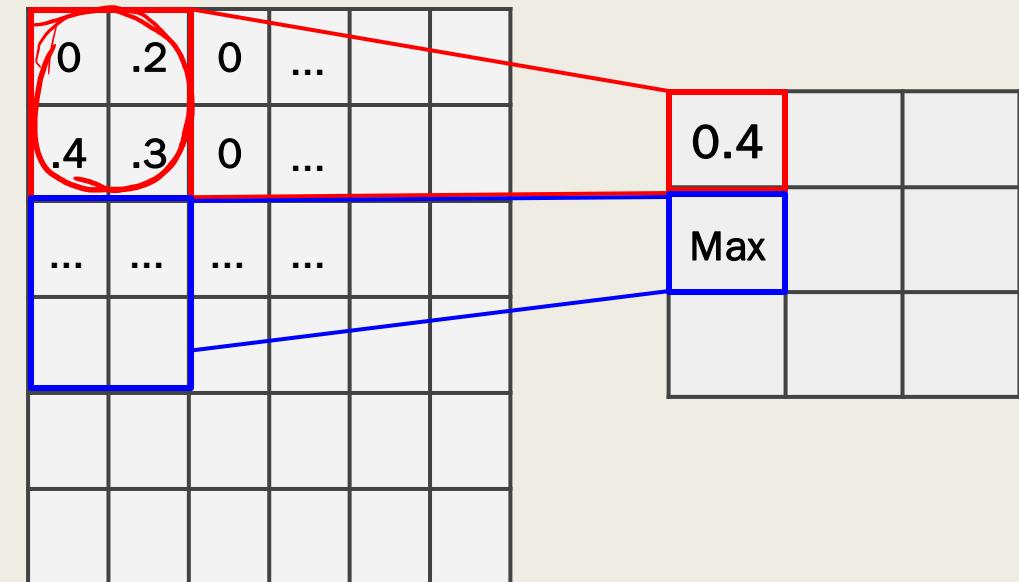
# Diagrama CNN - AlexNet

## ■ Diagrama CNN - AlexNet



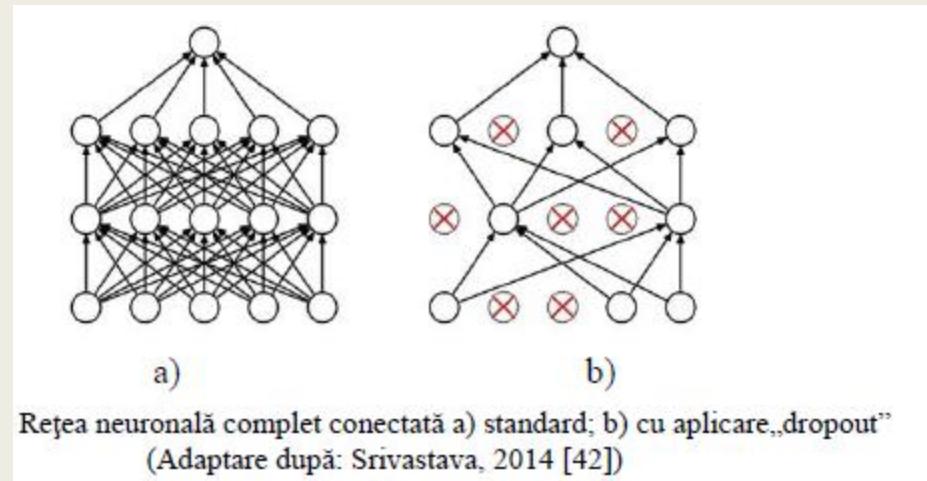
# Pooling Layers

- Pooling layers
  - aceasta operatie inlocuieste valoarea unei zone din imagine/harta de trasaturi cu o statistica a acelei zone.
- Functia de max-pooling este cea mai folosita in aplicatii si inlocuieste valoarea dintr-o zona bine definita cu maximul acelei zone, rezultand intr-o harta de trasaturi mai mica, **dar care pastreaza cea mai relevanta statistica**.
  - In acest mod, pe langa reducerea dimensionalitatii, se obtine si o invarianta la translatii mici.
- Ex. 2x2 max-pooling
  - pastram valoare maxima dintr-o regiune de 2x2
  - si ne mutam cu pasul dat de 'stride' (in acest caz 2)



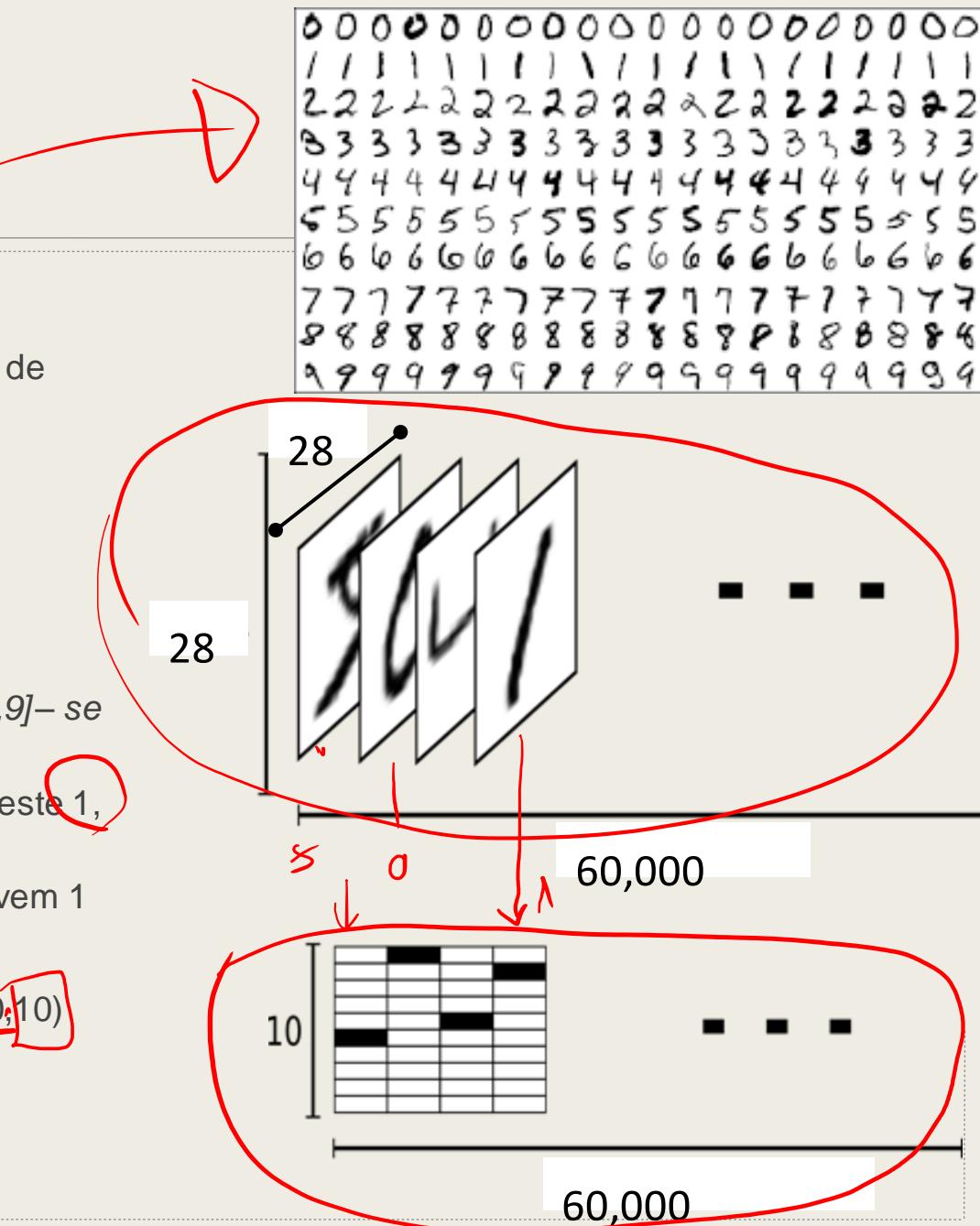
# Dropout

- Tehnica “dropout” - constă în setarea (aleatoare) la zero a ieșirilor unor neuroni de pe straturile ascunse cu o probabilitate de 0,5 (respectiv jumătate dintre aceștia).
- Neuronii care sunt astfel “decuplați” nu contribuie la propagarea înainte a semnalului și nici nu participă la ajustarea celorlalte ponderi la antrenarea cu *backpropagation*.
- Această tehnică reduce co-adaptările complexe ale neuronilor (ajuta la prevenire overfitting),
  - *din moment ce un neuron nu se poate baza pe prezența altor neuroni particulari, fiind astfel forțat să învețe caracteristici mai robuste care sunt utile în conjuncție cu multe subseturi aleatoare de alți neuroni.*

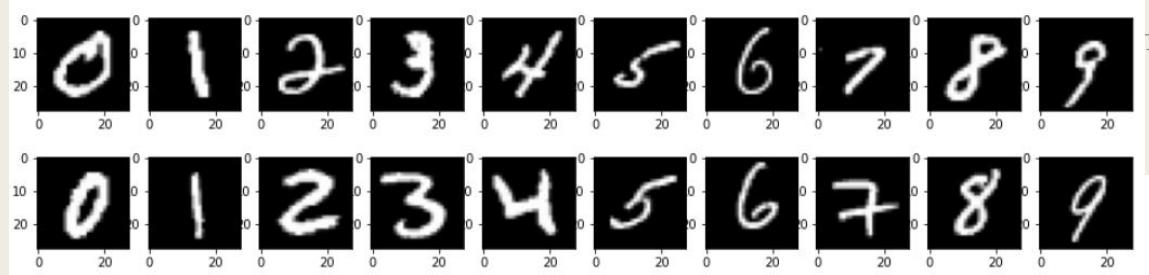


# MNIST DataSet

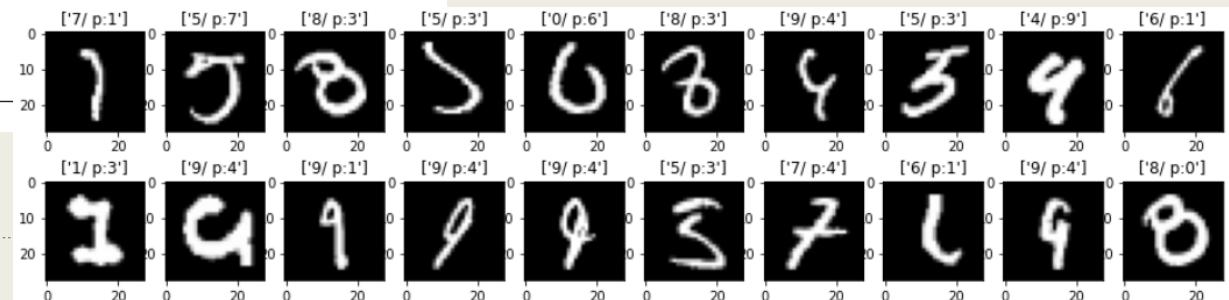
- Contine cei 10 digitii de la 0 la 9, scrisi de mână
  - Contine 60,000 imagini de o rezolutie 28x28, pe nivale de gri (1 canal de culoare)
  - Reprezentat ca un tensor - 4 dimensiuni (Samples, H, W, channels)
    - $(60000, 28, 28, 1)$
    - pentru imagini color, ultima dimensiune = 3 (RGB)
  - Etichetarea se realizează sub forma One-Hot Encoding.
    - Label-urile - in loc de o valoare pe imagine din  $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$  - se vor reprezenta sub forma unui vector pentru fiecare imagine
      - Vector de 10 valori (numarul claselor), unde doar un index este 1, restul zero
      - Valoarea label-ului este data de indexul pozitiei pe care avem 1
- Ex. pentru 4 vom avea label-ul [0,0,0,1,0,0,0,0,0]  $\leftarrow Y$
- => Etichetele pentru setul de date devin 2D, de dimensiune  $(60000, 10)$



# MNIST DataSet – Model Simplu

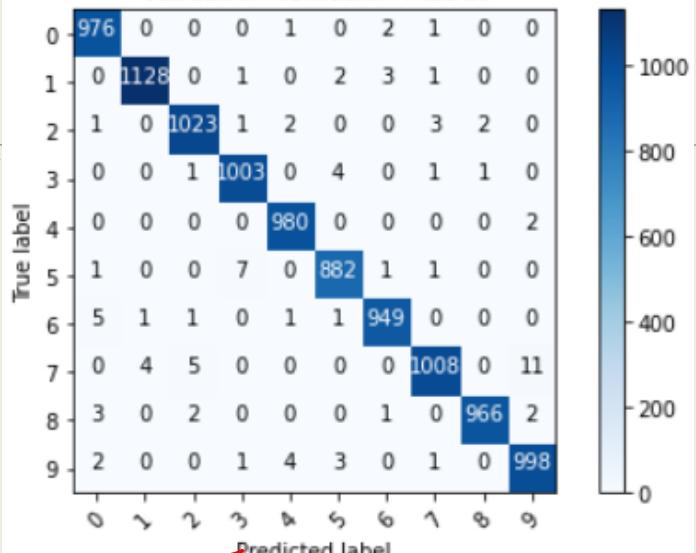


Layer (type)	Output Shape	Param #
<hr/>		
conv2d_2 (Conv2D)	(None, 25, 25, 32)	544
conv2d_3 (Conv2D)	(None, 22, 22, 32)	16416
max_pooling2d_2 (MaxPooling2D)	(None, 11, 11, 32)	0
conv2d_4 (Conv2D)	(None, 8, 8, 64)	32832
conv2d_5 (Conv2D)	(None, 5, 5, 64)	65600
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 64)	0
flatten_2 (Flatten)	(None, 256)	0
dense_3 (Dense)	(None, 512)	131584
dense_4 (Dense)	(None, 10)	5130
<hr/>		
Total params:	252,106	
Trainable params:	252,106	
Non-trainable params:	0	
<hr/>		
None		



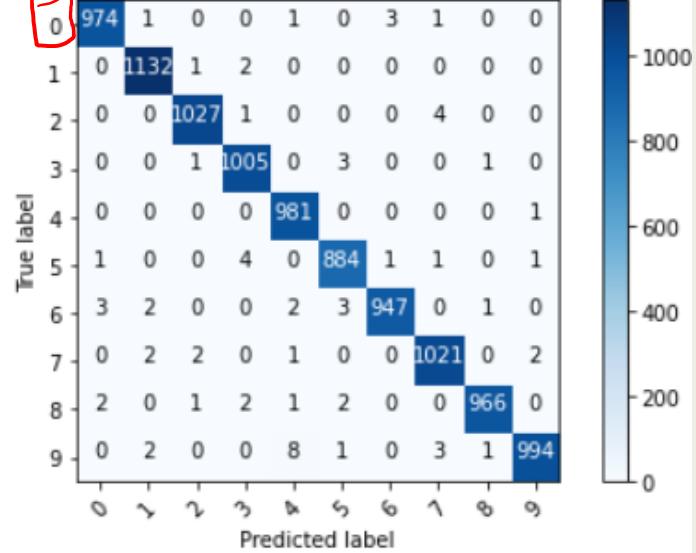
20 epoci - 87 immagini eronate

Validation confusion matrix



5 epoci - 69 immagini eronate

Validation confusion matrix



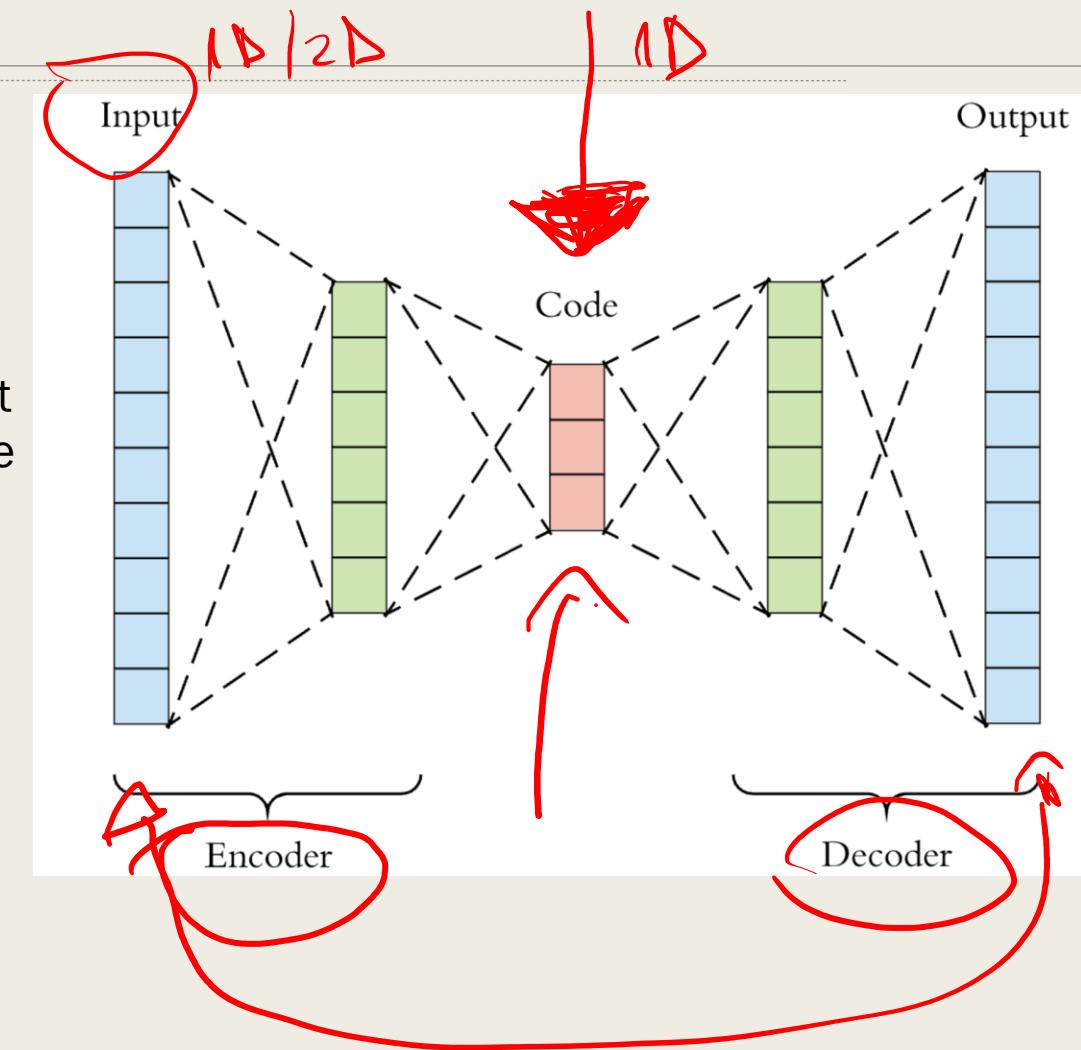
- <https://www.youtube.com/watch?v=vT1JzLTH4G4&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv>
- <https://www.pieriandata.com/>
- <https://www.pyimagesearch.com/2018/12/10/keras-save-and-load-your-deep-learning-models/>
- <https://pylessons.com/CNN-tutorial-introduction/>
  
- <https://towardsdatascience.com/architecture-comparison-of-alexnet-vggnet-resnet-inception-densenet-beb8b116866d>

# Autoencoders

Autoencoders are a deep learning model for transforming data from a high-dimensional space to a lower-dimensional space. They work by encoding the data, whatever its size, to a 1-D vector. This vector can then be decoded to reconstruct the original data (in this case, an image). The more accurate the autoencoder, the closer the generated data is to the original.

In contrast with **discriminative models**, there is another group called **generative models** which can create new images. For a given input image, the output of a discriminative model is a class label; the output of a generative model is an image of the same size and similar appearance as the input image.

One of the simplest generative models is the **autoencoder** (AE for short), which is the focus of this tutorial.



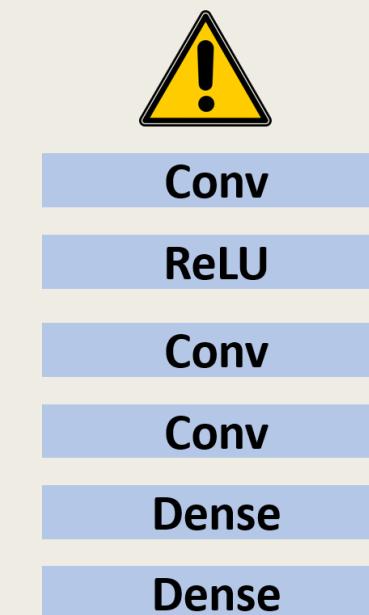
# Autoencoders

Autoencoders are a deep neural network model that can take in data, propagate it through a number of layers to condense and understand its structure, and finally generate that data again.

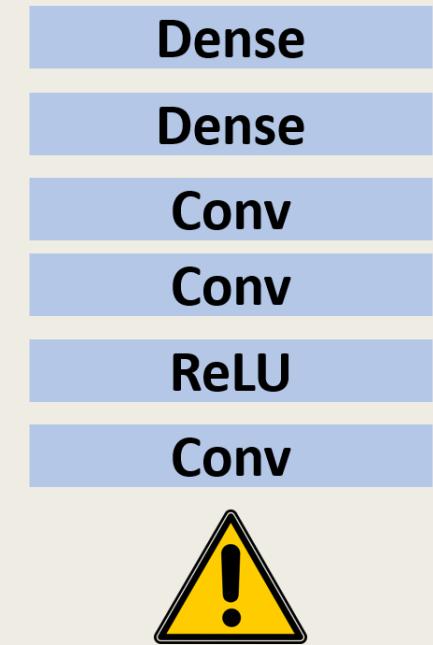
To accomplish this task an autoencoder uses two different types of networks. The first is called an **encoder**, and the other is the **decoder**. The decoder is just a reflection of the layers inside the encoder. Let's clarify how this works.

The job of the encoder is to accept the original data (e.g. an image) that could have two or more dimensions and generate a single 1-D vector that represents the entire image. The number of elements in the 1-D vector varies based on the task being solved. It could have 1 or more elements. The fewer elements in the vector, the more complexity in reproducing the original image accurately.

## Encoder Network



## Decoder Network



The loss is calculated by comparing the original and reconstructed images, i.e. by calculating the difference between the pixels in the 2 images.