CAPITOLUL 10

COMPRESIA DE IMAGINI

10. 1. Reprezentarea numerică a imaginilor

O imagine este o suprafață de obicei dreptunghiulară caracterizată, la nivelul oricărui punct al ei, de o anumită culoare. Ideal, culoarea variază continuu în oricare direcție. Din păcate, în sistemele numerice, nu se pot utiliza mărimi care variază continuu, ci doar forma discretizată a acestora.

Astfel, o imagine trebuie să fie discretizată înainte de a se pune problema reprezentării numerice. Discretizarea constă în împărțirea imaginii într-un caroiaj asemănător unei table de şah. Fiecare porțiune de imagine delimitată de acest caroiaj va fi considerată ca având o culoare uniformă - o medie a culorii existente pe această secțiune. Aceste secțiuni mai sunt denumite şi pixeli sau puncte, numărul acestora definind rezoluția imaginii. De exemplu, pentru o imagine oarecare care are o rezoluție de 640x480 pixeli, înseamnă că pe suprafața acesteia s-a definit un caroiaj care o împarte pe orizontală în 640 de secțiuni iar pe verticală, în 480.

Pasul următor îl constituie găsirea unei reprezentări pentru culoare. Orice culoare poate fi descompusă în trei culori primare (de exemplu roşu-R, verde-G şi albastru-B), cu alte cuvinte orice imagine poate fi obținută prin suprapunerea aditivă a trei radiații luminoase având aceste trei culori şi intensități diferite. Deci, pentru a reprezenta numeric o culoare, este suficient să se reprezinte intensitățile luminoase ale celor trei culori primare.

Dacă se alocă câte 8 biți pentru fiecare componentă, se pot coda 256 nivele de intensitate, astfel, absența culorii (intensitate zero) se codifică prin valoarea 00000000 în binar sau 00 în hexazecimal, iar intensitatea maximă, prin cea mai mare valoare ce poate fi reprezentată pe 8 biți, şi anume, 11111111 în binar sau FF în hexazecimal. Această reprezentare, însă, ține mai mult de modalitățile tehnice de captare şi reproducere a imaginii şi mai puțin de mecanismul fiziologic de percepere a culorii. Prin diferite experimente s-a constatat că din punct de vedere al capacității de percepere a detaliilor, ochiul este mai sensibil la intensitatea luminoasă a culorii decât la nuanță. Din acest motiv prezintă interes o altă modalitate de reprezentare a culorii care să țină cont de această observație, un exemplu fiind reprezentarea YUV utilizată în televiziunea în culori. În acest caz, în locul celor trei componente primare R,G,B se utilizează alte trei mărimi derivate din acestea, şi anume:

$$\begin{cases} Y = 0,3R + 0,59G + 0,11B \\ U = R - Y = 0,7T - 0,59G - 0,11B \\ V = B - Y = -0,3R - 0,59 + 0,89B \end{cases} \qquad (10.1)$$

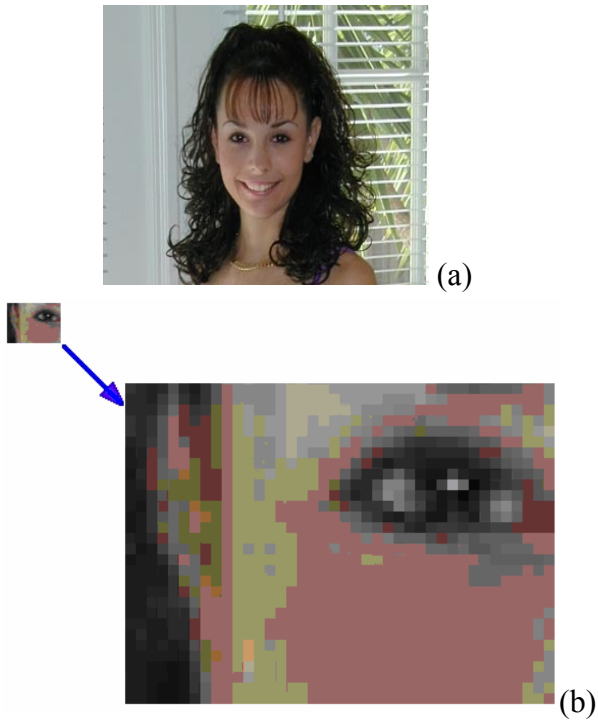
În cazul acestei reprezentări, componenta Y corespunde intensității luminoase percepute pentru respectiva culoare (coeficienții 0,30, 0,59 şi 0,11 reprezintă strălucirile relative la alb ale celor trei culori primare roşu, verde şi, respectiv, albastru). Această componentă mai este întâlnită şi sub numele de luminanță. Componentele U şi V sunt cele care definesc nuanța culorii, din acest motiv, sunt denumite componente de crominanță. Acestea se calculează ca diferența dintre componenta roşie, respectiv albastră, şi cea de luminanță.

Avantajul reprezentării YUV este acela că separă componenta de luminanță, pentru care ochiul este foarte sensibil la detalii, de componentele de nuanță pentru care sensibilitatea este mai redusă. Acest lucru face posibilă reducerea informației asociate unei imagini prin utilizarea unei rezoluții mai reduse pentru componentele de crominanță. În cazul

televiziunii în culori se realizează o "compresie" prin limitarea benzii de frecvență alocate semnalelor de crominanță (de exemplu în sistemul PAL semnalele U şi V au o bandă de 1,3MHz față de semnalul Y care are o bandă de 6MHz).

10.2. Reprezentarea imaginii în format necompresat

O imagine se reprezintă ca o matrice de puncte (de obicei de forma pătrată), fiecare punct fiind caracterizat de o culoare. De exemplu, pentru imaginea din Fig. 10.1(a) se poate evidenția acest lucru dacă se măreşte o secțiune a imaginii astfel încât matricea de puncte să devină vizibilă, ca în Fig. 10.1(b).



(a) (b)

Figura 10.1

Pentru a reprezenta o astfel de imagine trebuie să se utilizeze un mod de reprezentare numeric al culorii. Pentru aceasta se porneşte de la observaţia că orice culoare poate fi obţinută prin amestecul în diferite proporţii a trei culori de bază (culori primare). În practică se utilizează ROŞU (R), VERDE (G) şi ALBASTRU (B). Intensitatea luminoasă a unei culori primare poate fi reprezentată numeric sub forma unui întreg de 8 biţi, valoarea 0 corespunzând intensităţii nule iar cea maximă (255) intensităţii maxime. În acest fel, o culoare va fi reprezentată numeric printr-un triplet de întregi pe 8 biţi (R,G,B). De exemplu culoarea GALBEN va avea o reprezentare de forma (255,255,0). În aceste condiţii imaginea se reprezintă sub forma unei matrice IM(N_x,N_y) unde N_x reprezintă numărul de puncte pe orizontală şi N_y este nuămrul de puncte pe verticală, iar elementele matricei sunt tripleţi de întregi pe 8 biţi de tip (R,G,B).

10.3. Metode şi abordări ale compresiei imaginii

În continuare se reiau câteva metode folosite în compresie, evidenţiind aplicabilitatea lor în compresia de imagini.

1. Cuantizarea scalară poate fi folosită pentru a compresa imagini, dar performanţele ei sunt mediocre. De exemplu, o imagine cu 8 biţi/pixel poate fi compresată prin cuantizare scalară eliminând cei mai nesemnificativi patru biţi ai fiecărui pixel. Aceasta conduce la o rată de compresie de 0,5, care pe lângă faptul că nu este semnificativă, determină în acelaşi timp şi reducerea numărul de culori (sau nuanţe de gri) de la 256 la doar 16. O astfel de reducere nu numai ca descreşte pe ansamblu calitatea imaginii reconstruite, dar poate chiar crea benzi de diferite culori, un efect observabil şi deranjant care este ilustrat aici.

Exemplul 10.1.

Se consideră, de exemplu, un rând de 12 pixeli de culori similare, pornind de la 202 la 215. În notație binară aceste valori sunt:

11010111 11010110 11010101 11010011 11010010 11010001

11001111 11001110 11001101 11001100 11001011 11001010.

Cuantizarea va produce următoarele 12 valori de 4 biți:

1101 1101 1101 1101 1101 1101 1100 1100 1100 1100 1100 1100,

din care se vor reconstrui cei 12 pixeli, prin adăugarea a 4 zerouri, fiecărei valori cuantizate:

11010000 11010000 11010000 11010000 11010000 11010000

11000000 11000000 11000000 11000000 11000000 11000000.

Primii şase pixeli ai rândului acum au valoarea $11010000_2 = 208$, în timp ce următorii şase pixeli sunt $11000000_2 = 192$. Dacă rânduri adiacente au pixeli similari, primele şase coloane vor forma o bandă, clar diferită de banda formată de următoarele şase coloane. Acest fenomen de formare a benzilor, sau de conturare, este foarte evident pentru ochi, deoarece aceştia sunt sensibili la margini şi rupturi într-o imagine.

2. *Cuantizarea vectorială* poate fi folosită cu mai mult succes pentru a compresa imagini.

3. *Metodele statistice* funcționează mai bine când simbolurile ce trebuie compresate au probabilități diferite. O secvență de intrare în care mesajele au aceeaşi probabilitate nu se va compresa eficient. În acest sens, într-o imagine alb-negru sau color în tonuri continue, diferitele culori sau nuanțe de gri se dovedesc de multe ori a avea aproximativ aceleaşi probabilități. De aceea metodele statistice nu sunt o alegere bună pentru compresia unor astfel de imagini, şi sunt necesare noi abordări. Imaginile cu discontinuități de culoare, în care pixeli adiacenți au culori foarte diferite, se

compresează mai bine cu metodele statistice, dar în acest caz nu este uşoară predicția pixelilor.

4. *Metodele de compresie bazate pe dicționar* tind, de asemenea, să nu aibă succes în cazul imaginilor cu tonuri continue. O astfel de imagine conține de obicei pixeli adiacenți în culori similare, dar nu conține modele repetitive. Chiar şi o imagine care conține modele repetitive, cum sunt liniile verticale, le poate pierde când este digitizată. O linie verticală în imaginea originală poate deveni uşor oblică atunci când imaginea este digitizată. O linie verticală ideală este prezentată în Fig. 10.2a. În Fig. 10.2.b linia este presupusă a fi perfect digitizată în zece pixeli, aşezați vertical. Totuşi, dacă imaginea este plasată în digitizor uşor oblic, procesul de digitizare poate fi imperfect, şi pixelii rezultați pot arăta ca în Fig. 10.2.c.

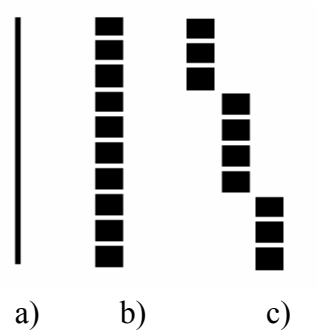
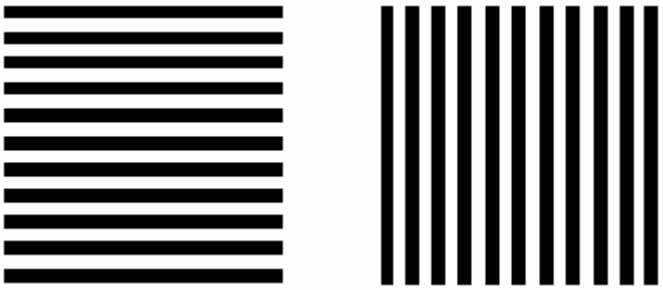


Fig. 10.2. Digitizare perfectă şi imperfectă.

O altă problemă a compresiei imaginilor bazate pe dicționar este aceea că astfel de metode scanează imaginea rând cu rând, şi pot pierde astfel corelații verticale între pixeli. Un exemplu sunt cele două imagini din Figura 10.3 a, b. Salvând ambele imagini în GIF89, un format de fişier grafic bazat pe dicționar, au rezultat fişiere de dimensiuni 1053, respectiv 1527 octeți.



(a) (b)

Figura 10.3. Compresia bazată pe dicționar a liniilor paralele

Metodele tradiționale sunt nesatisfăcătoare pentru compresia de imagini, astfel încât au fost necesare abordări noi, care, deşi diferite, se bazează pe eliminarea redundanței din imagine, folosind următorul principiu:

Principiul compresiei de imagine. Dacă se selectează aleator un pixel dintr-o imagine, există o probabilitate mare ca vecinii săi să aibă aceeaşi culoare sau culori foarte apropiate.

Compresia de imagine este, deci, bazată pe faptul că pixelii învecinați sunt puternic corelați. Această corelare se numeşte şi redundanță spațială.

Exemplul 10.2.

În continuare este prezentat un exemplu simplu care ilustrează cum poate fi eliminată redundanța dintr-un şir de pixeli corelați. Următoarea secvență de valori reprezintă intensitățile a 24 de pixeli adiacenți dintr-un rând al unei imagini în tonuri continue: 12, 17, 14, 19, 21, 26, 23, 29, 41, 38, 31, 44, 46, 57, 53, 50, 60, 58, 55, 54, 52, 51, 56, 60.

Doar doi din cei 24 de pixeli sunt identici. Valoarea lor medie este 40,3. Scăzând perechi de pixeli adiacenți rezultă secvența: 12, 5, -3, 5, 2, 4, -3, 6, 11, -3, -7, 13, 4, 11, -4, -3, 10, -2, -3, 1, -2, -1, 5, 4.
Cele două secvențe sunt ilustrate în Figura 10.4.

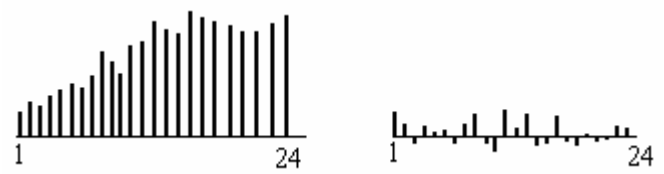


Fig. 10.4: Valorile şi diferențele celor 24 pixeli adiacenți.

Secvența de valori diferență are trei proprietăți care ilustrează potențialul ei de compresie:

1. Valorile secvenței diferență sunt mai mici decât valorile pixelilor originali. Media lor este de 2,58.
2. Există valori ale secvenței diferențelor care se repetă. Există doar 15 valori diferență distincte, deci, în principiu ele pot fi codate prin patru biți fiecare.
3. Valorile secvenței diferență sunt decorelate, adică valori diferență adiacente tind a fi diferite. Aceasta poate fi observată dacă se efectuează o nouă scădere a lor, rezultând secvența 12, -7, -8, 8, -3, 2, -7, 9, 5, -14, -4, 20, -11, 7, -15, 1, 13, -12, -1, 4, -3, 1, 6, 1. Valorile acesteia sunt mai mari decât diferențele anterioare.

În general, metodele de compresie pentru imagini sunt proiectate pentru un tip particular de imagine şi în continuare se prezintă câteva din aceste metode specifice. Imaginile particulare vizate sunt imagini cu două nivele, imagini cu tonuri de gri şi imagini color.

Abordarea 1. Aceasta este folosită pentru imagini cu două nivele. Un pixel dintr-o astfel de imagine este reprezentat printr-un bit. Aplicând

principiul compresiei de imagine asupra unei imagini cu două nivele, înseamnă că pixelii învecinați ai unui pixel *P* tind a fi identici cu *P*. Astfel, are sens folosirea unei codări RLC (Run length coding) pentru a compresa o astfel de imagine. O metodă de compresie pentru o astfel de imagine o poate scana în ordinea rastrului (rând cu rând), calculând lungimile şirurilor de pixeli albi şi negri. Aceste lungimi sunt codate prin coduri de lungime variabilă şi sunt înscrise în secvența compresată. Un exemplu de astfel de metodă este compresia facsimil.

Ar trebui accentuat faptul că aceasta este doar o abordare a imaginilor cu două nivele. În practică, detaliile metodelor particulare diferă, în funcție de aplicație. De exemplu, o metodă poate scana imaginea coloană cu coloană, sau în zig-zag, sau o poate scana regiune cu regiune.

Abordarea 2 se aplică, de asemenea, pentru imagini cu două nivele. Principiul compresiei de imagine spune că vecinii unui pixel tind a fi similari lui. Se poate extinde acest principiu şi concluziona că dacă pixelul curent are culoarea *c* (unde *c* este ori alb, ori negru), atunci pixelii de aceeaşi culoare anteriori şi următori tind să aibă aceiaşi vecini imediați.

Această abordare urmăreşte *n* vecini apropiați ai pixelului curent şi îi consideră ca un număr de *n* biți. Acest număr se numeşte *contextul* pixelului. În principiu pot exista 2^n contexte, dar datorită redundanței imaginii, distribuția lor este neuniformă. Unele contexte ar trebui să fie foarte frecvente, iar celelalte, rare. Codorul numără de câte ori a fost găsit deja fiecare context pentru un pixel de culoarea *c*, şi asignează corespunzător probabilități acestor contexte. Dacă pixelul curent are culoarea *c* şi contextul său are probabilitatea *p*, codificatorul poate folosi coduri aritmetice adaptive pentru a codifica pixelul cu acea probabilitate. Această abordare este folosită de standardul JBIG (Joint Bi-level Processing Group).

În continuare, se consideră imagini în nuanțe de gri. Un pixel dintr-o astfel de imagine este reprezentat prin n biți şi poate avea una din 2^n valori. Aplicarea principiul compresiei de imagine asupra unei imagini în nuanțe de gri implică faptul că vecinii imediați ai unui pixel P tind a fi similari cu P, însă nu în mod necesar identici cu el. Astfel, nu mai poate fi folosită codarea RLE (run length encoding) pentru compresia unei astfel de imagini, ci sunt folosite următoarele două abordări.

Abordarea 3. Se separă imaginea în tonuri de gri în n imagini pe două nivele şi apoi se compresează fiecare cu un cod RLE instantaneu. Principiul compresiei de imagini, în acest caz, s-ar formula prin afirmația că doi pixeli adiacenți care sunt similari în imaginea în tonuri de gri vor fi identici în cele mai multe imagini cu două nivele. Acest lucru însă nu este adevărat, aşa cum reiese din următorul exemplu.

Exemplul 10.3.

Se consideră o imagine în tonuri de gri cu $n = 4$ (adică 4 biți/pixel sau 16 nuanțe de gri). Imaginea poate fi separată în patru imagini bi-nivel. Dacă doi pixeli adiacenți din imaginea originală au valorile 0000 şi 0001, atunci sunt similari. De asemenea, sunt identici în trei din cele patru imagini bi-nivel. Totuşi, doi pixeli adiacenți cu valori 0111 şi 1000 sunt de asemenea similari în imaginea în tonuri de gri (valorile lor sunt 7, respectiv 8), dar diferă în toate cele patru imagini alb-negru.

Această problemă apare deoarece reprezentarea binară a numerelor întregi adiacente poate diferi prin mai mulți biți. Codurile binare pentru 0 şi 1 diferă printr-un bit, cele pentru 1 şi 2 diferă prin doi biți, şi cele pentru 7 şi 8 prin patru biți. Soluția este proiectarea unor coduri speciale, astfel încât codurile oricăror două numere întregi consecutive i şi i+1 să difere numai printr-un singur bit. Un exemplu de astfel de cod este codul Gray reflectat [Sal].

Abordarea 4. Se foloseşte contextului unui pixel pentru a prezice valoarea sa. Contextul unui pixel este dat de valorile câtorva dintre vecinii săi. Se examinează câțiva vecini ai unui pixel *P*, se calculează o medie A, a valorilor lor, şi se prezice că *P* va avea valoarea A. Principiul compresiei de imagini spune că predicția va fi corectă în cele mai multe cazuri, aproape corectă în multe cazuri, şi complet greşită în puține cazuri. Valoarea prezisă a pixelului *P* reprezintă informația redundantă în *P*, astfel încât se calculează diferența : $\Delta = P - A$, şi se asignează coduri instantanee de lungime variabilă pentru diferitele valori Δ. Dacă P poate lua valori de la 0 la *m*-1, atunci Δ va avea valori în intervalul [-(m-1),+(m-1)], şi numărul cuvintelor de cod necesare este 2m - 1.

Experimente cu un număr mare de imagini sugerează că valorile lui Δ tind să fie distribuite după distribuția Laplace. O metodă de compresie ar putea să folosească această distribuție pentru a asigna o probabilitate fiecărei valori a lui Δ, şi apoi să se folosească codarea aritmetică pentru a coda eficient valorile Δ. Acesta este principiul metodei progresive multinivel MLP [Sal].

Contextul unui pixel poate fi constituit din unul sau doi din vecinii săi imediați. Dacă, însă, se consideră mai mulți pixeli vecini în obținerea contextului, se pot obține rezultate mai bune. Media A într-un astfel de caz ar trebui ponderată cu vecinii apropiați, care au o pondere mai mare. Pentru ca decodorul să poată decoda o imagine, ar trebui să poată calcula contextul fiecărui pixel. Acest lucru înseamnă că în context ar trebui să fie incluşi doar pixelii care au fost deja codați. Dacă imaginea este scanată în ordinea rastrului, contextul ar trebui să conțină doar pixeli localizați deasupra pixelului curent sau pe acelaşi rând cu el, la stânga.

Abordarea 5. Se aplică o transformare valorilor pixelilor, şi se codează valorilor transformate. Se reaminteşte că pentru a realiza compresia, trebuie redusă sau eliminată redundanța. Redundanța unei imagini este cauzată de corelația dintre pixeli, deci transformând pixelii

într-o reprezentare în care aceştia sunt decorelați, se elimină redundanța. De asemenea este posibil ca o transformare să fie apreciată în funcție de entropia imaginii. Într-o imagine puternic corelată, pixelii tind a avea valori echiprobabile, ceea ce duce la o entropie maximă. Dacă pixelii transformați sunt decorelați, anumite valori de pixeli devin mai frecvente, având astfel probabilități mari, în timp ce alte valori sunt rare, fapt ce conduce la o entropie mică. Cuantizând valorile transformate, se poate produce o compresie cu pierdere de informație, eficientă, a imaginii. Se doreşte ca valorile transformate să fie independente, deoarece codarea valorilor independente face mai simplă construirea unui model statistic.

În cazul imaginilor în culori, un pixel este constituit din trei componente de culoare, roşu, verde şi albastru. Majoritatea imaginilor color sunt ori în tonuri continue, ori în tonuri discrete.

Abordarea 6. Principiul acestei abordări constă în separarea unei imagini color în tonuri continue în trei imagini în tonuri de gri şi compresia fiecăreia din ele separat, folosind abordările 2, 3 şi 4.

Pentru o imagine în tonuri continue, principiul compresiei de imagini implică faptul că pixelii adiacenți au culori similare, dacă nu chiar identice. Totuşi, culori similare nu înseamnă valori similare ale pixelilor. Se consideră, de exemplu, valori pe 12 biți ale pixelilor, în care fiecare componentă de culoare este exprimată în patru biți. Astfel, cei 12 biți 1000|0100|0000 reprezintă un pixel a cărui culoare este o mixtură de opt unități de roşu (aproape 50%, din valoarea maximă de 15 unități), patru unități de verde (circa 25%), şi deloc albastru. Se consideră doi pixeli adiacenți cu valorile 0011|0101|0011 şi 0010|0101|0011. Aceştia au culori similare, din moment ce doar componentele lor roşii diferă printr-o unitate. Cu toate acestea, când se consideră ca numere de 12 biți, cele două numere 001101010011 şi 001001010011 sunt diferite, pentru că diferă într-un bit cu pondere semnificativă.

O caracteristică importantă a acestei abordări este folosirea unei reprezentări tip luminanță – crominanță, YUV, în loc de reprezentarea comună RGB. Avantajul acestei reprezentări este că ochiul este sensibil la modificări mici ale luminanței, dar nu şi la ale crominanței. Aceasta permite pierderea unei cantități considerabile de date în componentele de crominanță, fără o pierdere vizibilă de calitate.

Abordarea 7. O abordare diferită este necesară pentru imaginile în tonuri discrete. Se reaminteşte că o astfel de imagine conține regiuni uniforme care pot apărea de mai multe ori într-o imagine. Un exemplu îl constituie o pagină scrisă la calculator care constă din text şi icoane. Fiecare caracter de text şi fiecare icoană este o regiune, şi fiecare regiune poate apărea de mai multe ori în imagine. O modalitate posibilă de compresie a unei astfel de imagini este scanarea sa, identificarea regiunilor, şi găsirea regiunilor care se repetă. Dacă o regiune B este identică cu o regiune A deja găsită, atunci B poate fi compresată prin înregistrarea unui pointer corespunzător lui A în secvența compresată. Metoda descompunerii în blocuri este un exemplu de implementare a acestei abordări.

Abordarea 8. Se împarte imaginea în regiuni (care se suprapun sau nu) şi se compresează prin procesarea părților una câte una. Se presupune că următoarea parte de imagine neprocesată este partea cu numărul *n.* Se încearcă regăsirea ei în părțile $1 \div n-1$, care au fost deja procesate. Dacă partea *n* poate fi exprimată, de exemplu, ca o combinație a unor părți anterioare scalate şi rotite, atunci doar cele câteva numere care specifică combinația trebuie salvate, şi partea *n* poate fi ignorată. Dacă partea *n* nu poate fi exprimată ca o combinație de părți deja procesate, aceasta este procesată şi salvată în secvența compresată.

Această abordare este baza diferitelor metode fractale pentru compresia de imagini. Se aplică principiul compresiei de imagine asupra părților de imagine, în loc de pixelii individuali. Aplicat în acest fel, principiul afirmă că imaginile ce urmează a fi compresate au un anumit

volum de auto-similaritate, adică părți de imagine sunt identice sau similare cu întreaga imagine sau cu alte părți.

10.4. Transformări folosite în compresia imaginilor

Conceptul matematic de transformare este important în multe domenii, printre care şi cel al compresiei de imagini. O imagine poate fi compresată prin transformarea pixelilor săi (care sunt corelați) într-o reprezentare unde aceştia sunt decorelați. Compresia este obținută dacă valorile noi sunt mai mici, în medie, decât cele originale. Compresia cu pierdere de informație poate fi obținută prin cuantizarea valorilor transformate. Decodorul primește valorile transformate din secvența compresată şi reconstruieşte datele originale (exacte sau aproximate), prin aplicarea transformării inverse. Transformările discutate în această secțiune sunt ortogonale.

Termenul de decorelare se referă la faptul că valorile transformate sunt independente unele de altele. Ca urmare, ele pot fi codate independent, ceea ce face mai simplă construirea unui model statistic. O imagine poate fi compresată, dacă reprezentarea sa are redundanță. Redundanța în imagini derivă din corelarea pixelilor. Dacă se transformă imaginea într-o reprezentare în care pixelii sunt decorelați, se elimină redundanța şi imaginea a devenit în totalitate compresată.

Se consideră cazul în care se scanează o imagine în ordinea rastrului şi se grupează perechile de pixeli adiacenți. Deoarece pixelii sunt corelați, cei doi pixeli ai unei perechi, în mod normal, au valori similare. În continuare se consideră perechile de pixeli ca puncte în spațiul bi-dimensional, şi se reprezintă grafic. Se ştie că toate punctele de forma (x,x) sunt localizate pe prima bisectoare, $y = x$, aşa că este de aşteptat ca punctele considerate să fie concentrate în jurul acesteia. Fig. 10.4a arată rezultatele reprezentării

pixelilor unei imagini oarecare, în care un pixel are valori de la 0 la 255. Cele mai multe puncte sunt grupate în jurul acestei linii, şi doar câteva puncte sunt localizate departe de ea. Acum se transformă imaginea prin rotirea tuturor punctelor cu 45° în sensul acelor de ceasornic în jurul originii, aşa încât prima bisectoare coincide acum cu axa *x*, ca în Fig.10.4 b.

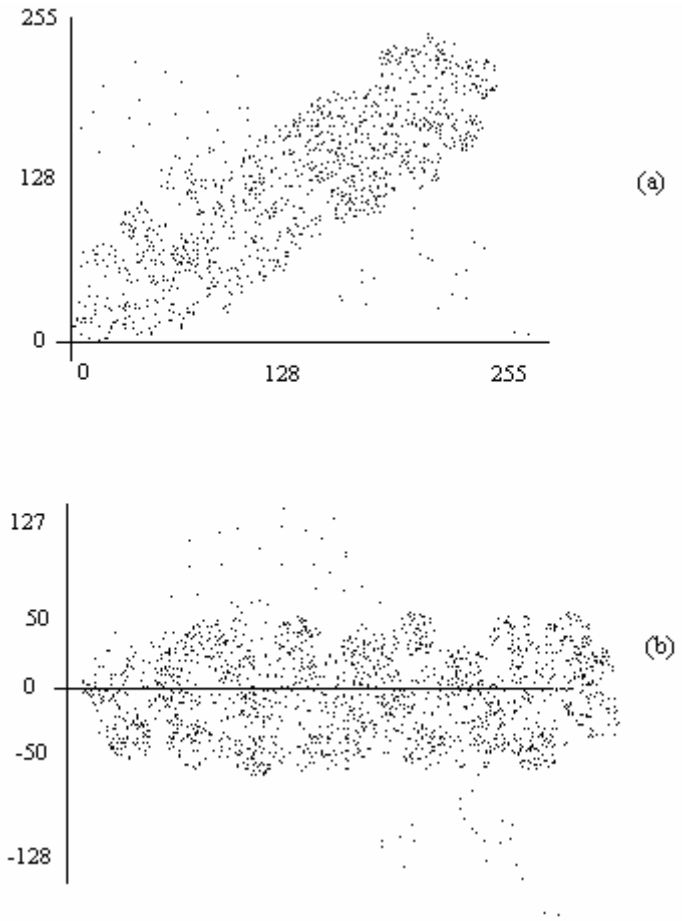


Fig. 10.4. Rotirea unui grup de puncte

Aceasta se face prin intermediul transformării simple

$$(x^*, y^*) = (x, y)\begin{pmatrix} \cos 45^0 & -\sin 45^0 \\ \sin 45^0 & \cos 45^0 \end{pmatrix} = (x, y)\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} = (x, y)\mathbf{R} \quad (10.2)$$

unde matricea de rotație **R** este ortonormală (adică produsul scalar al unui rând cu el însuşi este 1, produsul scalar al rândurilor diferite este 0, şi la fel pentru coloane). Transformarea inversă este

$$(x, y) = (x^*, y^*)\mathbf{R}^{-1} = (x^*, y^*)\mathbf{R}^T = (x^*, y^*)\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \quad (10.3)$$

(Inversa unei matrici ortonormale este transpusa sa).

Este evident că majoritatea punctelor ajung să aibă coordonatele *y* nule sau aproape nule, în timp ce coordonatele *x* nu se modifică foarte mult. Distribuțiile coordonatelor *x* şi *y* (adică pixelii cu număr impar şi par dintr-o imagine) înainte de rotație nu diferă cu mult, pe când, după rotație, distribuția coordonatelor *x* rămâne aproape la fel, dar cea a coordonatelor *y* este concentrată în jurul lui zero.

Cum coordonatele punctelor sunt cunoscute înainte şi după rotație, este simplu să se măsoare reducerea ce intervine în corelația punctelor, prin calculul funcției de corelație $\sum_i x_i y_i$ dintre puncte. În acest caz, compresia fără pierderi a imaginii poate fi efectuată prin simpla folosire a pixelilor transformați în secvența compresată. Dacă se acceptă compresie cu pierdere de informație, atunci toți pixelii pot fi cuantizați, obținându-se numere şi mai mici. De asemenea, se pot separa toți pixelii cu număr impar (cei care creează coordonatele *x* ale perechilor), urmați apoi de toți pixelii cu număr par. Aceste două secvențe sunt numite *vectorii coeficienților* transformării. A doua secvență constă din numere mici şi poate avea, după cuantizare, şiruri de zerouri, care pot conduce la o compresie şi mai bună.

Se poate arăta că dispersia totală a pixelilor nu se modifică prin rotație [Sal], din moment ce matricea de rotație este ortonormală. Totuşi, deoarece dispersia noilor coordonate *y* este mică, cea mai mare parte din

dispersie este acum concentrată în coordonatele *x*. Dispersia este uneori numită *energia* distribuției pixelilor, astfel încât se poate afirma că rotația a concentrat (sau compactat) energia în coordonata *x* şi a realizat compresia în acest fel.

Concentrarea energiei într-o coordonată prezintă avantajul că face posibilă cuantizarea acestei coordonate mai fin decât pentru celelalte coordonate. Următorul exemplu ilustrează eficiența acestei transformări fundamentale.

Exemplul 10.4.

Se consideră punctul (4,5), ale cărui coordonate sunt apropiate. Folosind ecuația (10.2), punctul este transformat în $(4,5)\mathbf{R} = (9,1)/\sqrt{2} \approx (6,36396; 0,7071)$. Energiile punctului şi transformatului său sunt $4^2 + 5^2 = 41 = (9^2 + 1^2)/2$. Dacă se neglijează coordonata mai mică, 4, a punctului, se ajunge la o eroare de $4^2/41 = 0,39$. Dacă, însă, se neglijează cel mai mic din cei doi coeficienți transformați (0,7071), eroarea rezultată este doar de $0,7071^2/41 = 0,012$. Un alt mod de a obține aceeaşi eroare este considerarea punctului reconstruit. Aplicând punctului transformat $\frac{1}{\sqrt{2}}(9,1)$ transformarea inversă dată de relația 10.3, se ajunge la punctul original (4,5). Făcând acelaşi lucru cu $\frac{1}{\sqrt{2}}(9,0)$ rezultă punctul reconstruit aproximat (4,5, 4,5). Diferența de energie dintre punctul original şi cel reconstruit este aceeaşi cantitate

$$\frac{[(4^2 + 5^2) - (4,5^2 + 4,5^2)]}{4^2 + 5^2} = \frac{41 - 40,5}{41} = 0,012 \qquad (10.4)$$

Această transformare simplă poate fi extinsă cu uşurință la orice număr de dimensiuni. În loc de a selecta perechi de pixeli adiacenți, se pot selecta triplete. Fiecare triplet devine un punct în spațiul tri-dimensional, şi aceste puncte formează o regiune concentrată în jurul liniei care formează

unghiuri de 45° cu cele trei axe de coordonate. Când această linie este rotită astfel încât să coincidă cu axa *x*, coordonatele *y* şi *z* ale punctelor transformate devin numere mici. Transformarea este făcută prin multiplicarea fiecărui punct cu o matrice de rotație de dimensiune 3 × 3, ortonormală. Punctele transformate sunt apoi separate în trei vectori de coeficienți, din care ultimii doi sunt alcătuiți din numere mici. Pentru compresie maximă, fiecare vector de coeficienți trebuie cuantizat separat.

Această idee se poate extinde la mai mult de trei dimensiuni, cu singura diferență că nu se pot vizualiza spații de dimensiuni mai mari de trei. Unele metode de compresie, cum ar fi JPEG, împart o imagine în blocuri de 8 × 8 pixeli fiecare, şi rotesc fiecare bloc de două ori. Această rotație dublă produce un set de 64 de valori transformate, din care prima, numită „coeficient DC" sau de curent continuu, este mare, şi celelalte 63, numite „coeficienții AC" sau de curent alternativ, sunt, de obicei, mici. Astfel, această transformare concentrează energia în prima din cele 64 de dimensiuni.

10.4.1. Transformări ortogonale

Transformările de imagine folosite în practică trebuie să fie rapide şi, de preferință, simplu de implementat. Aceasta sugerează folosirea transformărilor liniare. Într-o astfel de transformare, fiecare valoare transformată c_i este o sumă ponderată a pixelilor d_j, unde fiecare este multiplicat cu un factor (sau coeficient de transformare) w_{ij}. Astfel,

$$c_i = \sum_j d_j w_{ij} \text{ , pentru } i,j=1,2, ...,n. \qquad (10.4)$$

Pentru n=4, relația precedentă se scrie matriceal

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix}$$

În general

$$\mathbf{C = W \cdot D} \qquad (10.4')$$

Fiecare rând al lui **W** este numit *vector al bazei*.

Se doreşte determinarea valorilor ponderilor w_{ij}, astfel încât prima valoare transformată c_1 să fie mare, şi restul valorilor $c_2, c_2, ...$ să fie mici. Din relația $c_i = \sum_j d_j w_{ij}$ se observă că c_i va fi mare, când fiecare pondere w_{ij} consolidează contribuția lui d_j la c_i. Aceasta are loc, de exemplu, când vectorii w_{ij} şi d_j au valori şi semne similare. Invers, c_i va fi mic, dacă toate ponderile w_{ij} sunt mici şi jumătate din ele au semnul opus lui d_j. Astfel, când se obține un c_i mare, înseamnă că vectorul w_{ij} al bazei este similar cu vectorul de date d_j. Un c_i mic, pe de altă parte, înseamnă că w_{ij} şi d_j au forme diferite. În concluzie, vectorii bazei w_{ij} pot fi interpretați ca mijloace de a extrage caracteristici din vectorul de date.

În practică, ponderile ar trebui să fie independente de date. Altfel, ponderile ar trebui incluse în secvența compresată, pentru a fi folosite de decodor. Aceast lucru, combinat cu faptul că datele sunt valorile pixelilor, care sunt nenegative, sugerează o modalitate de a alege vectorii bazei. Primul vector, cel care produce c_1, ar trebui să fie alcătuit din valori pozitive, poate chiar identice. Aceasta va întări valorile nenegative ale pixelilor. Fiecare din ceilalți vectori ar trebui să aibă jumătate din elemente pozitive, cealaltă jumătate, negative. Când sunt multiplicați cu valorile nenegative ale datelor, astfel de vectori tind să producă o valoare mică. O

alegere bună a vectorilor bazei ar fi dacă aceştia ar fi foarte diferiţi unul de altul, şi astfel pot extrage mai multe caracteristici. Aceasta duce la ideea ca vectorii bazei să fie ortogonali. Dacă matricea de transformare **W** este ortogonală, transformarea în sine se numeşte ortogonală. O altă observaţie care ajută la selectarea vectorilor bazei este că aceştia ar trebui să aibă frecvenţe din ce în ce mai mari, extrăgând astfel caracteristici de frecvenţă mai înaltă din date pe parcursul calculului valorilor transformate.

Aceste consideraţii sunt satisfăcute de matricea ortogonală:

$$\mathrm{W} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \tag{10.5}$$

Primul vector al bazei (rândul de sus al lui **W**) constă numai din valori 1, deci frecvenţa sa este zero. Fiecare din vectorii următori are doi de +1 şi doi de -1, deci produc valori transformate mici, şi frecvenţele lor (măsurate ca numărul de schimbări de semn de-a lungul vectorului bazei) devin mai înalte. Această matrice este similară cu transformarea Walsh-Hadamard [ref].

Exemplul 10.5.

Se transformă vectorul de date (4, 6, 5, 2) cu ajutorul matricei 10.5 şi se obţine

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 6 \\ 5 \\ 2 \end{pmatrix} = \begin{pmatrix} 17 \\ 3 \\ -5 \\ 1 \end{pmatrix}$$

Rezultatele sunt încurajatoare, din moment ce c_1 este mare (în comparaţie cu datele originale) şi două din valorile rămase c_i sunt mici.

Totuşi, energia datelor originale este $4^2+6^2+5^2+2^2=81$, în timp ce energia valorilor transformate este $17^2+3^2+(-5)^2+1^2=324$, de patru ori mai mult. Este posibil a se conserva energia, prin multiplicarea matricei de transformare **W** prin factorul de scalare 1/2. Noul produs $\mathbf{W}\cdot(4,6,5,2)^T$ generează acum valorile transformate (17/2, 3/2, -5/2, 1/2). Energia este conservată, dar este concentrată în prima componentă, care conţine $8,5^2/81=89\%$ din energia totală, în comparaţie cu datele originale, în care prima componentă conţinea $4^2/81=20\%$ din energie.

Un alt avantaj al lui **W** este că efectuează şi transformarea inversă. Produsul $\mathbf{W}\cdot(17/2,\ 3/2,\ -5/2,\ 1/2)^T$ reconstruieşte datele originale (4, 6, 5, 2).

Pentru a aprecia eficienţa transformării, se cuantizează vectorul transformat (8,5; 1,5; -2,5; 0,5) în întregii (9, 1, -3, 0) şi se efectuează transformarea inversă, obţinându-se vectorul (3,5; 6,5; 5,5; 2,5). Un alt experiment este de a renunţa complet la cele două elemente mai mici ale vectorului şi de a face transformarea inversă pentru vectorul (8,5; 0; -2,5; 0). Aceasta produce datele reconstruite (3; 5,5; 5,5; 3), încă foarte apropiate de valorile originale. În concluzie şi această transformare simplă este utilă în compresie.

10.4.2. Transformări bi-dimensionale

Exemplul 10.6.

Având datele bi-dimensionale sub forma de matrice 4×4

$$D=\begin{pmatrix}4 & 7 & 6 & 9\\ 6 & 8 & 3 & 6\\ 5 & 4 & 7 & 6\\ 2 & 4 & 5 & 9\end{pmatrix}$$

(în care prima coloană este identică cu exemplul anterior), se poate aplica transformare uni-dimensională anterioară. Rezultatul este:

$$\mathbf{C'} = \mathbf{W}\times\mathbf{D} = \frac{1}{2}\cdot\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \qquad \mathbf{D} = \begin{pmatrix} 8,5 & 11,5 & 10,5 & 15 \\ 1,5 & 3,5 & -1,5 & 0 \\ -2,5 & -0,5 & 0,5 & 3 \\ 0,5 & -0,5 & 2,5 & 0 \end{pmatrix}$$

Fiecare coloană din **C'** este transformarea unei coloane din **D**. Se observă primul element al fiecărei coloane a lui **C'** este dominant. De asemenea, toate coloanele au aceeaşi energie. Se poate considera că **C'** este prima etapă dintr-un proces în doi paşi care produce transformarea bi-dimensională a matricei **D**. Pasul doi ar trebui să transforme fiecare linie a lui **C'**, şi aceasta se face prin înmulţirea lui **C'** cu transpusa $\mathbf{W}^{\mathbf{T}}$. Pentru acest caz particular **W** este simetrică, deci se ajunge la $C=C'\times W^T=WCW^T$ sau

$$\mathbf{C} = \begin{pmatrix} 8,5 & 11,5 & 10,5 & 15 \\ 1,5 & 3,5 & -1,5 & 0 \\ -2,5 & -0,5 & 0,5 & 3 \\ 0,5 & -0,5 & 2,5 & 0 \end{pmatrix}\cdot\frac{1}{2}\cdot\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} =$$

$$= \begin{pmatrix} 22,75 & -2,75 & 0,75 & -3,75 \\ 1,75 & 3,25 & -0,25 & -1,75 \\ 0,25 & -3,25 & 0,25 & -2,25 \\ 1,25 & -1,25 & -0,75 & 1,75 \end{pmatrix}$$

Elementul din stânga sus este dominant, conţinând 89% din energia totală de 579 din matricea de date **D** originală. Transformarea în doi paşi, bi-dimensională, a redus corelarea atât în dimensiunea verticală cât şi în cea orizontală.

În continuarea se vor prezenta pe scurt câteva transformări uzuale.

10.4.3. Transformarea Karhunen-Loève (KLT)

Transformarea Karhunen-Loève (KLT) are cea mai bună eficienţă din punct de vedere al compactării energiei (sau, altfel spus, decorelare a pixelilor), dar are mai mult o valoare teoretică decât una practică.

Se consideră o imagine care se împarte în *k* blocuri de câte *n* pixeli fiecare, unde *n* este de obicei 64, dar poate avea şi alte valori, şi *k* depinde de mărimea imaginii. Se consideră blocurile de vectori $\mathbf{b}^{(i)}$, unde *i*=1, 2, ..., *k*. Pe baza acestora se calculează vectorul medie $\bar{\mathbf{b}} = \left(\sum_i \mathbf{b}^{(i)}\right)/k$. Se defineşte un nou set de vectori $\mathbf{v}^{(i)} = \mathbf{b}^{(i)} - \bar{\mathbf{b}}$, care vor avea media zero. Matricea transformării KLT are dimensiunea $n \times n$ şi se notează cu **A**. Rezultatul transformării vectorului $\mathbf{v}^{(i)}$ este vectorul pondere $\mathbf{w}^{(i)}=\mathbf{A}\mathbf{v}^{(i)}$. Valoarea medie a lui $\mathbf{w}^{(i)}$ este de asemenea zero. Se construieşte o matrice **V** ale cărei coloane sunt vectorii $\mathbf{v}^{(i)}$ şi o altă matrice ale cărei coloane sunt vectorii pondere $\mathbf{w}^{(i)}$.

$$\mathbf{V}=(\mathbf{v}^{(1)},\mathbf{v}^{(2)},.....,\mathbf{v}^{(k)}), \quad \mathbf{W}=(\mathbf{w}^{(1)},\mathbf{w}^{(2)},....,\mathbf{w}^{(k)}) \tag{10.6}$$

Matricele **V** şi **W** au fiecare *n* linii şi *k* coloane. Din definiţia lui $\mathbf{w}^{(i)}$ rezultă că $\mathbf{W}=\mathbf{A}\cdot\mathbf{V}$.

Cei *n* vectori de coeficienţi $\mathbf{c}^{(j)}$ din transformarea Karhunen-Loève sunt daţi de relaţia:

$$\mathbf{c}^{(j)} = (w_j^{(1)}, w_j^{(2)}, ..., w_j^{(k)}), \quad j = 1, 2, ..., n \tag{10.7}$$

Astfel, vectorul $\mathbf{c}^{(j)}$ este format din elementele "j" ale tuturor vectorilor pondere $\mathbf{w}^{(i)}$ cu *i*=1,2,...,*k* ($\mathbf{c}^{(j)}$ este coordonata *j* a vectorilor $\mathbf{w}^{(i)}$).

Se examinează elementele matricei produs ($\mathbf{W}\cdot\mathbf{W}^T$) (aceasta este o matrice de dimensiuni $n \times n$). Un element oarecare, din linia *a* şi coloana *b*, al acestei matrice este o sumă de produse:

$$(\mathbf{W} \cdot \mathbf{W}^T)_{ab} = \sum_{i=1}^{k} w_a^{(i)} w_b^{(i)} = \sum_{i=1}^{k} c_i^{(a)} c_i^{(b)} = \mathbf{c}^{(a)}\mathbf{c}^{(b)}, \text{ pentru } a, b \in [1, n] \tag{10.8}$$

Deoarece valoarea medie a fiecărui vector $\mathbf{w}^{(i)}$ este zero, un element $(\mathbf{W}\cdot\mathbf{W}^{\mathrm{T}})_{jj}$, de pe diagonala principală a matricei produs, este varianța sau dispersia elementului j (sau a coordonatei j) a vectorului $\mathbf{w}^{(i)}$.

Elementele din afara diagonalei sunt covariațele vectorilor $\mathbf{w}^{(i)}$, adică un element $(\mathbf{W}\times\mathbf{W}^{T})_{ab}$ este covarianța coordonatelor a şi b ale vectorilor $\mathbf{w}^{(i)}$, care este egală cu produsul scalar $\mathbf{c}^{(a)}\mathbf{c}^{(b)}$. Un deziderat major al transformărilor aplicate imaginii este de a decorela coordonatele vectorilor. Din teoria probabilităților se ştie că două coordonate sunt decorelate, dacă covarianța lor este zero. Un alt deziderat este acela de compactare a energiei, care, de fapt este în strânsă legătură cu primul. Având în vedere aceste lucruri, se urmăreşte găsirea unei matrice de transformare $\mathbf{A}$, astfel încât produsul $(\mathbf{W}\cdot\mathbf{W}^{\mathrm{T}})$ să fie o matrice diagonală.

Din definiția matricei $\mathbf{W}$ se obține:

$$\mathbf{W}\cdot\mathbf{W}^{\mathrm{T}}=(\mathbf{AV})\cdot(\mathbf{AV})^{\mathrm{T}}=\mathbf{A}(\mathbf{V}\cdot\mathbf{V}^{\mathrm{T}})\mathbf{A}^{\mathrm{T}} \tag{10.9}$$

Matricea $\mathbf{V}\cdot\mathbf{V}^{\mathrm{T}}$ este simetrică, şi elementele ei sunt covariațele coordonatelor vectorilor $\mathbf{v}^{(i)}$:

$$(\mathbf{V}\times\mathbf{V}^{T})_{ab}=\sum_{i=1}^{k}v_{a}^{(i)}v_{b}^{(i)},\ \text{pentru}\ a,b\in[1,n] \tag{10.10}$$

Deoarece matricea $(\mathbf{V}\times\mathbf{V}^{T})$ este simetrică, vectorii săi proprii sunt ortogonali. Se normalizează aceşti vectori, pentru a fi ortonormali şi se aleg ca linii ale matricei $\mathbf{A}$. Rezultatul obținut este:

$$\mathbf{W}\times\mathbf{W}^{T}=A(\mathbf{V}\times\mathbf{V}^{T})A^{T}=\begin{pmatrix}\lambda_1 & 0 & 0 & \cdots & 0\\ 0 & \lambda_2 & 0 & \cdots & 0\\ 0 & 0 & \lambda_3 & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & 0 & \cdots & \lambda_n\end{pmatrix} \tag{10.11}$$

Această alegere a matricei $\mathbf{A}$ conduce la o matrice $\mathbf{W}\times\mathbf{W}^{T}$ diagonală, ale cărei elemente de pe diagonala principală sunt valorile proprii ale matricei

$\mathbf{V}\times\mathbf{V}^T$. Matricea **A** este matricea transformării Karhunen – Loeve, liniile sale fiind vectorii bazei KLT şi energiile (varianțele) vectorilor transformați sunt valorile proprii $\lambda_1, \lambda_2, ..., \lambda_n$ ai matricei $(\mathbf{V}\times\mathbf{V}^T)$. Vectorii bazei transformării KL nu sunt ficşi, ei sunt dependenți de date, fiind calculați pe baza pixelilor imaginii originale. Din acest motiv, ei trebuie să incluşi în secvența de date compresate, fapt ce scade eficiența acestei transformate.

Exemplul 10.7.

Se urmăreşte obținerea transformatei Karhunen Loeve de ordinul 2 pentru o secvență de intrare arbitrară. Matricea de autocorelație a unui proces știonar este

$$\mathbf{R} = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) \\ R_{xx}(1) & R_{xx}(0) \end{bmatrix} \tag{10.12}$$

unde $R_{xx}(n)$, n=0,1 sunt valorile funcției de autocorelație.

Rezolvând ecuația $|\lambda I - R| = 0$, se obțin cele două valori proprii $\lambda_1 = R_{xx}(0) + R_{xx}(1), \lambda_2 = R_{xx}(0) - R_{xx}(1)$. Vectorii proprii corespunzători sunt

$$V_1 = \begin{bmatrix} \alpha \\ \alpha \end{bmatrix} \qquad V_2 = \begin{bmatrix} \beta \\ -\beta \end{bmatrix} \tag{10.13}$$

unde α, β sunt constante arbitrare. Dacă se impune condiția de ortonormalitate, care presupune ca amplitudinea vectorului să fie unu, se obține

$$\alpha = \beta = \frac{1}{\sqrt{2}}$$

şi matricea K este:

$$K = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{10.14}$$

Se observă că această matrice nu depinde de valoarile lui $R_{xx}(0)$ şi $R_{xx}(1)$. Acest lucru este adevărat doar pentru transformată de dimensiune 2×2. Matricile transformate de ordin mai mare sunt funcție de valorile funcției de autocorelație şi, implicit, de date.

10.4.4. Transformarea Walsh-Hadamard (WHT)

Această transformare are eficiență de compresie scăzută, nefiind folosită mult în practică. Cu toate acestea, este rapidă, deoarece poate fi calculată doar prin adunări, scăderi, şi uneori, câte o deplasare la dreapta (pentru a împărți eficient printr-o putere a lui 2).

Dat fiind un bloc de $N \times N$ de pixeli P_{xy} (unde N trebuie să fie o putere a lui 2, $N = 2^n$), WHT bi-dimensională corespunzătoare şi inversa sa sunt definite prin ecuațiile 10.15 şi, respectiv 10.16:

$$\begin{aligned} H(u,v) &= \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} P_{xy} g(x,y,u,v) = \\ &= \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} P_{xy}(-1)^{\sum_{i=0}^{n-1}[b_i(x)p_i(x)+b_i(y)p_i(y)]} \end{aligned} \tag{10.15}$$

$$\begin{aligned} P_{xy} &= \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} H(u,v) h(x,y,u,v) = \\ &= \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} H(u,v)(-1)^{\sum_{i=0}^{n-1}[b_i(x)p_i(x)+b_i(y)p_i(y)]} \end{aligned} \tag{10.16}$$

unde $H(u,v)$ sunt rezultatele transformării (adică, coeficienții WHT), cantitatea $b_i(u)$ este bitul i al reprezentării binare a numărului întreg u, iar $p_i(u)$ este definit în funcție de $b_j(u)$ prin ecuația 10.17.

$$
\begin{aligned}
&p_0(u) = b_{n-1}(u),\\
&p_1(u) = b_{n-1}(u) + b_{n-2}(u),\\
&p_2(u) = b_{n-2}(u) + b_{n-3}(u),\\
&\text{.....}\\
&p_{n-1}(u) = b_1(u) + b_0(u).
\end{aligned}
\tag{10.17}
$$

De exemplu, se consideră u=6=110_2. Biții zero, unu, şi doi ai lui 6 sunt, respectiv, 0,1 şi 1, deci $b_0(6)=0$, $b_1(6)=1$, şi $b_2(6)=1$.

Mărimile g(x, y, u, v) şi h(x, y, u, v) sunt numite nuclee (kernels) (sau imagini de bază) ale WHT. Aceste matrice sunt identice. Elementele lor sunt doar +1 şi -1, şi sunt multiplicate prin factorul 1/N. Ca urmare, transformarea WHT constă în multiplicarea fiecărui pixel din imagine cu +1 sau -1, adunare, şi împărţirea sumei prin N. Deoarece $N=2^n$, împărţirea prin 2^n poate fi făcută prin deplasarea cu *n* poziţii spre dreapta.

Nucleele WHT sunt prezentate, în formă grafică, pentru N=4, în figura 10.5, unde albul reprezintă +1 şi negrul -1 (factorul 1/N este ignorat).

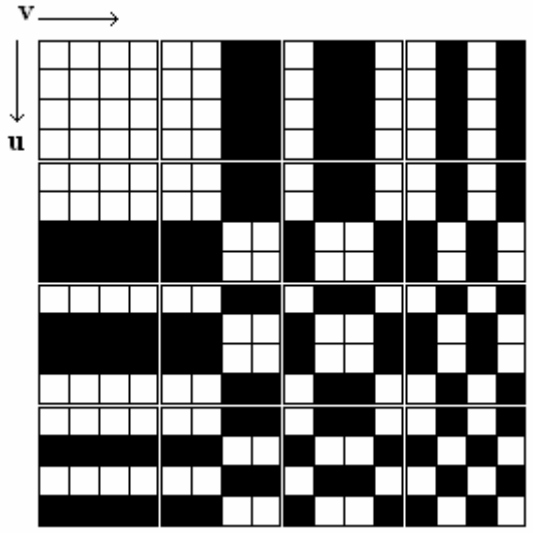


Fig. 10.5. Nucleul ordonat al WHT pentru N = 4.

Rândurile şi coloanele de blocuri din această figură corespund unor valori ale lui u şi respectiv, v, de la 0 la 3. Numărul de schimbări de semn pe parcursul unui rând sau al unei coloane a unei matrice se numeşte *secvențierea* rândului sau coloanei.

Matricele corespunzătoare transformatei DWHT pot fi obținute recursiv şi ca rearanjări ale matricelor discrete Hadamard care sunt de o importanță particulară în teoria codării [ref]. O matrice Hadamard de dimensiune N are proprietatea că $\mathbf{HH}^{\mathbf{T}}=\mathbf{NI}$ unde **I** este matricea unitate. Matricele Hadamard ale căror dimensiuni sunt puteri ale lui doi pot fi construite astfel:

$$H_{2N}=\begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix} \tag{10.18}$$

cu $H_1=[1]$. Astfel se obțin

$$H_2=\begin{bmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{bmatrix}=\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{10.19}$$

$$H_4=\begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix}=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{10.20}$$

$$H_8=\begin{bmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{bmatrix}=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{10.21}$$

Matricea **H** a transformării DWHT poate fi obținută din matricea Hadamard prin multiplicarea acesteia cu un factor de normalizare, astfel încât $\mathbf{HH}^{\mathrm{T}}=\mathbf{I}$, şi prin reordonarea liniilor în ordine descrescătoare a frecvenței. Normalizarea implică multiplicarea matricei cu $\frac{1}{\sqrt{N}}$. Reordonând H_8, se obține

$$H=\frac{1}{\sqrt{8}}\begin{bmatrix}1&1&1&1&1&1&1&1\\1&1&1&1&-1&-1&-1&-1\\1&1&-1&-1&-1&-1&1&1\\1&1&-1&-1&1&1&-1&-1\\1&-1&-1&1&1&-1&-1&1\\1&-1&-1&1&-1&1&1&-1\\1&-1&1&-1&-1&1&-1&1\\1&-1&1&-1&1&-1&1&-1\end{bmatrix} \qquad (10.22)$$

Deoarece matricea fără factorii de scalare constă din ±1, operația de transformare constă doar în adunare şi scădere. Din acest motiv transformata este utilă în situații în care minimizarea câştigului de calcul este foarte importantă.

10.4.5. Transformarea Haar

Transformarea Haar [ref] este bazată pe funcția Haar $h_k(x)$, care este definită pentru $x \in [0,1]$ şi pentru k=0, 1, ..., N-1, unde $N=2^n$.

Orice întreg *k* poate fi exprimat sub formă de sumă astfel k=2p+q-1, unde $0 \le p \le n-1$, q=0 sau 1 pentru p=0, şi $1 \le q \le 2^p$ pentru $p \ne 0$. De exemplu, pentru $N=4=2^2$, se obține $0=2^0+0-1$, $1=2^0+1-1$, $2=2^1+1-1$, şi $3=2^1+2-1$.

Funcţia de bază a transformatei Haar este definită astfel:

$$h_0(x) = h_{00}(x) = \frac{1}{\sqrt{N}}, \qquad pentru \quad 0 \le x \le 1 \tag{10.23}$$

şi

$$h_k(x) \overset{def}{=} h_{pq}(x) = \frac{1}{\sqrt{N}} \begin{cases} 2^{p/2}, \frac{q-1}{2^p} \le x < \frac{(q-1)/2}{2^p} \\ -2^{p/2}, \frac{(q-1)/2}{2^p} \le x < \frac{q}{2^p} \\ 0, \text{ în rest} \end{cases} \tag{10.24}$$

Matricea transformatei Haar, A_N de ordinul N×N poate fi construită astfel: un element oarecare al matricei (i,j) este funcţia de bază $h_i(j)$, unde i=0, 1,.., N-1 şi j=0/N, 1/N, ..., (N-1)/N (i=1 implică p=0 şi q=1). De exemplu:

$$A_2 = \begin{pmatrix} h_0(0/2) & h_0(1/2) \\ h_1(0/2) & h_1(1/2) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{10.25}$$

10.4.6. Transformata Discretă Cosinus (DCT)

Transformările cosinus şi sinus discrete, (DCT, DST) aparţin familiei transformărilor trigonometrice cu aplicaţii în compresia/decompresia datelor. Dintre acestea, DCT este de departe mai folosită în practică, datorită proprietăţii de compactare a energiei.

- **DCT unidimensională**

În practică se foloseşte DCT bi-dimensională, dar pentru uşurinţa înţelegerii se consideră mai întâi DCT uni-dimensională. Se consideră formele de undă w(f)=cos(fθ), pentru $0 \le \theta \le \pi$, cu frecvenţele f=0, 1, ...,7, şi

$$\theta = \frac{\pi}{16}, \frac{3\pi}{16}, \frac{5\pi}{16}, \frac{7\pi}{16}, \frac{9\pi}{16}, \frac{11\pi}{16}, \frac{13\pi}{16}, \frac{15\pi}{16} \qquad (10.26)$$

Fiecare formă de undă w(f) este eşantionată în opt puncte, pentru a forma un vector al bazei $\mathbf{v}_f$. Cei opt vectori rezultați $\mathbf{v}_f$, f=0, 1,...,7 (un total de 64 de numere) sunt prezentați în Tabelul 10.1. Aceştia reprezintă baza pentru DCT uni-dimensională. Se observă similaritatea dintre acest tabel şi matricea **W** din ecuația (10.5).

Tabelul 10.1

θ	0,196	0,589	0,982	1,374	1,767	2,160	2,553	2,945
cos 0θ	1	1	1	1	1	1	1	1
cos 1θ	0,981	0,831	0,556	0,195	-0,195	-0,556	-0,831	-0,981
cos 2θ	0,924	0,383	-0,383	-0,924	-0,924	-0,383	0,383	0,924
cos 3θ	0,831	-0,195	-0,981	-0,556	0,556	0,981	0,195	-0,831
cos 4θ	0,707	-0,707	-0,707	0,707	0,707	-0,707	-0,707	0,707
cos 5θ	0,556	-0,981	0,195	0,831	-0,831	-0,195	0,981	-0,556
cos 6θ	0,383	-0,924	0,924	0,383	-0,383	0,924	-0,924	0,383
cos 7θ	0,195	-0,556	0,831	0,981	0,981	-0,831	0,556	-0,195

Aceşti opt vectori $\mathbf{v}_i$ sunt ortonormali (datorită alegerii particulare a celor opt puncte de eşantionare) şi pot fi organizați într-o matrice de transformare 8×8. Pentru că această matrice este ortonormală, ea este o matrice de rotație, deci, DCT uni-dimensională poate fi interpretă ca o rotație în opt dimensiuni.

O altă interpretare a DCT uni-dimensională este aceea că se pot considera cei opt vectori ortonormali $\mathbf{v}_i$ ca bază a unui spațiu vectorial, şi orice alt vector **p** poate fi exprimat în acest spațiu ca o combinație liniară a acestor $\mathbf{v}_i$. De exemplu, se aleg ca date de test 8 numere corelate, **p**=(0,6; 0,5; 0,4; 0,5; 0,6; 0,5; 0,4; 0,55). Se exprimăm vectorul **p** ca o combinație

liniară a celor opt vectori ai bazei, $\mathbf{p} = \sum w_i \mathbf{v}_i$. Rezolvând acest sistem de opt ecuații se obțin cele opt ponderi:

$$w_0 = 0,506, \quad w_1 = 0,0143, \quad w_2 = 0,0115, \quad w_3 = 0,0439,$$
$$w_4 = 0,0795, \quad w_5 = -0,0432, \quad w_6 = 0,00478, \quad w_7 = -0,0077$$

Ponderea w_0 nu este cu mult diferită de elementele vectorului **p**, dar celelalte şapte ponderi sunt mult mai mici. Acest fapt indică modul în care DCT (sau orice altă transformare ortogonală) produce compresie. Cele 8 ponderi vor reprezenta pur şi simplu elementele compresate ale vectorului **p**. Cuantizând cele opt ponderi, se poate creşte considerabil compresia, în timp ce se pierde doar o cantitate mică de date.

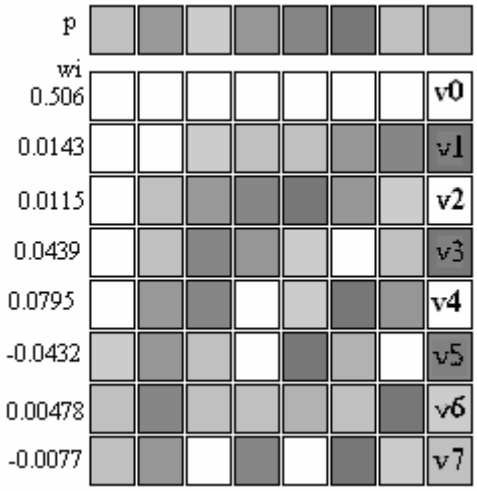


Fig. 10.6. Reprezentarea grafică a DCT uni-dimensional

Figura 10.6 ilustrează grafic această combinație liniară. Fiecare din cei opt vectori $\mathbf{v}_i$ este prezentat ca un rând de opt dreptunghiuri mici, gri, unde o valoare de +1 este colorată în alb, şi -1 în negru. Fiecare din cele opt elemente ale vectorului **p** este exprimat ca suma ponderată a unei scări de gri cu opt nivele.

Cel mai simplu mod de a calcula DCT uni-dimensională, în practică, este cu relația

$$G_f = \frac{1}{2} C_f \sum_{t=0}^{7} p_t \cos\left(\frac{(2t+1) f \pi}{16} \right) \qquad (10.27)$$

unde

$$C_f = \begin{cases} \frac{1}{\sqrt{2}}, & f = 0, \\ 1, & f > 0, \end{cases} \quad \text{pentru } f = 0,1,...,7 . \qquad (10.28)$$

Se începe cu un set de opt valori de date p_t (pixeli, eşantioane de sunet, sau alte date) şi se obține un set de opt coeficienți DCT, G_f. Decodorul primeşte coeficienții DCT în seturi de opt, şi aplică transformarea inversă DCT (IDCT) pentru a reconstrui valorile de date originale (tot în grupuri de câte opt). IDCT se calculează cu relația

$$p_t = \frac{1}{2} \sum_{j=0}^{7} C_j G_j \cos\left(\frac{(2t+1) j \pi}{16} \right), \quad \text{pentru t=0, 1,...., 7} \qquad (10.29)$$

Exemplul 10. 7.

Următorul experiment ilustrează eficiența DCT. Se începe cu setul de opt date **p**=(12, 10, 8, 10, 12, 10, 8, 11), se aplică acestora DCT uni-dimensională şi se obțin cei opt coeficienți: 28,6375; 0,571202; 0,46194; 1,757; 3,18198; -1,72956; 0,191342; -0,308709.

Aceştia pot fi folosiți pentru a reconstrui cu precizie datele originale (cu excepția unor mici erori cauzate de precizia limitată a maşinilor). Scopul este, însă, de a compresa datele şi mai mult, astfel încât se recurge la cuantizarea coeficienților obținuți.

Mai întâi aceştia se cuantizează la 28,6; 0,6; 0,5; 1,8; 3,2; -1,8; 0,2; -0,3; şi apoi se aplică IDCT, obținându-se coeficienții 12,0254; 10,0233; 7,96054; 9,93097; 12,0164; 9,99321; 7,94354; 10,9989.

Se cuantizează apoi coeficienţii şi mai mult, la 28; 1; 1; 2; 3; -2; 0; 0, şi se aplică IDCT, obţinându-se valorile 12,1883; 10,2315; 7,74931; 9,20863; 11,7876; 9,54549; 7,82865; 10,6557.

În sfârşit, se cuantizează coeficienţii la 28; 0; 0; 2; 3; -2; 0; 0, şi prin aplicarea IDCT se obţine secvenţa 11,236; 9,62443; 7,66286; 9,57302; 12,3471; 10,0146; 8,05304; 10,6842; unde cea mai mare diferenţă dintre o valoare originală (12) şi o valoare reconstruită (11,236) este 0,764 (sau 6,4% din 12).

- **DCT bi-dimensională**

Din experienţă se ştie că pixelii unei imagini sunt corelaţi pe două dimensiuni, nu doar pe una (un pixel este corelat cu vecinii săi de la stânga şi de la dreapta, deasupra şi dedesubt). De aceea metodele de compresie a imaginii folosesc DCT bi-dimensională, dată de relaţia

$$G_{ij} = \frac{1}{\sqrt{2n}} C_i C_j \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} p_{xy} \cos\left(\frac{(2y+1)j\pi}{2n}\right) \cos\left(\frac{(2x+1)i\pi}{2n}\right), \quad (10.30)$$

pentru $0 \le i, j \le n$-1. Imaginea este împărţită în blocuri de $n \times n$ pixeli p_{xy} (de obicei se foloseşte n=8), şi ecuaţia (10.30) este folosită pentru a obţine un bloc de 8×8 coeficienţi DCT, G_{ij}, pentru fiecare bloc de pixeli. Dacă compresia este cu pierdere de informaţie, coeficienţii sunt cuantizaţi. Decodorul reconstruieşte un bloc de valori de date (aproximate sau precise) prin calculul IDCT.

$$p_{xy} = \frac{1}{\sqrt{2n}} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} C_i C_j G_{ij} \cos\left(\frac{(2y+1)j\pi}{2n}\right) \cos\left(\frac{(2x+1)i\pi}{2n}\right) \quad (10.31)$$

unde

$$C_f = \begin{cases} \frac{1}{\sqrt{2}}, & f = 0, \\ 1, & f > 0, \end{cases} \quad \text{pentru } f = 0,1,...,7$$

DCT bi-dimensională poate fi interpretată în două moduri diferite, ca o rotație (de fapt, două rotații separate), şi ca bază a unui spațiu vectorial *n*-dimensional. În prima interpretare se consideră un bloc de *n*×*n* pixeli. Mai întâi se consideră fiecare rând al acestui bloc ca un punct $(p_{x,0}; p_{x,1}; ...; p_{x,n-1})$ în spațiul *n*-dimensional, şi se roteşte punctul cu ajutorul transformării date de suma din interior

$$G1_{x,j} = C_j \sum_{y=0}^{n-1} p_{xy} \cos\left(\frac{(2y+1)j\pi}{2n}\right) \tag{10.32}$$

a ecuației (10.30). Aceasta transformare are ca rezultat un bloc $G1_{x,j}$ de *n*×*n* coeficienți, unde primul element al fiecărui rând este dominant şi restul elementelor sunt mici. Suma exterioară a ecuației (10.30) este

$$G_{ij} = \frac{1}{\sqrt{2n}} C_i \sum_{x=0}^{n-1} G1_{x,j} \cos\left(\frac{(2x+1)i\pi}{2n}\right) \tag{10.33}$$

Aici, coloanele lui $G1_{x,j}$ sunt considerate puncte în spațiul *n*-dimensional, şi sunt rotite. Rezultatul este un coeficient mare în colțul stânga-sus al blocului şi n^2 - 1 coeficienți mici în rest. Această interpretare consideră DCT bi-dimensional ca două rotații separate, fiecare în *n* dimensiuni. Este interesant de observat că două rotații în *n* dimensiuni sunt mai rapide decât una în n^2 dimensiuni, deoarece în al doilea caz este necesară o matrice de rotație de dimensiune $n^2 \times n^2$.

A două interpretare (presupunând *n* = 8) foloseşte ecuația (10.30) pentru a crea 64 blocuri de 8×8 valori fiecare. Cele 64 de blocuri sunt apoi folosite ca bază a unui spațiu de vectori 64-dimensionali (sunt imagini de bază). Imaginile de bază folosite în DCT bi-dimensională sunt date în Fig.10.7. Orice bloc *B* de 8×8 pixeli poate fi exprimat ca o combinație liniară a imaginilor de bază, şi cele 64 de ponderi ale acestei combinații liniare sunt coeficienții DCT ai blocului *B*.

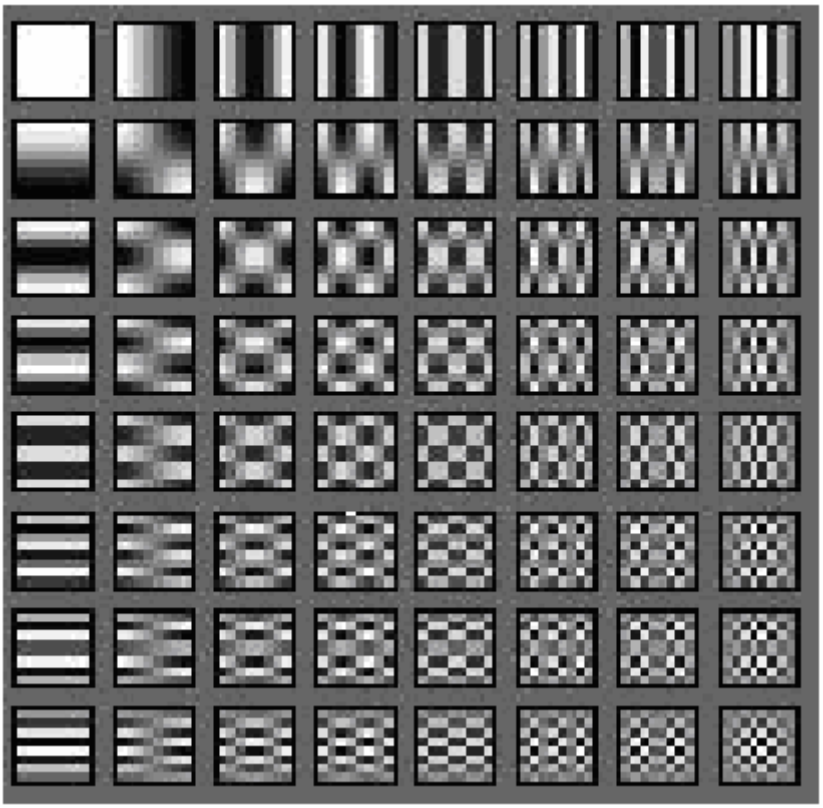


Fig. 10.7. Imaginile de bază pentru transformata discretă cosinus bi-dimensinală

Exemplul 10.8.

În continuare, se prezintă rezultatele aplicării transformatei DCT bi-dimensionale la a două blocuri de 8×8 valori. Primul bloc, cu valorile din Tabelul 10.2, are valori întregi puternic corelate în intervalul [8,12] şi cel de-al doilea, valori aleatoare în acelaşi interval. Primul bloc conduce la un coeficient DC mare, urmat de coeficienţi AC mici (incluzând 20 de zerouri). Coeficienţii celui de-al doilea bloc, prezentaţi în Tabelul 10.3, includ doar un zero.

Tabelul 10.2. DCT bi-dimensională a unui bloc de valori corelate

12 10 8 10 12 10 8 11	81 0 0 0 0 0 0 0
11 12 10 8 10 12 10 8	0 1,57 0,61 1,19 0,38 -1,81 0,20 -0,32
8 11 12 10 8 10 12 10	0 -0,61 0,71 0,35 0 0,07 0 0,02
10 8 11 12 10 8 10 12	0 1,90 -0,35 4,76 0,77 -3,39 0,25 -0,54
12 10 8 11 12 10 8 10	0 -0,38 0 -0,77 8,00 0,51 0 0,07
10 12 10 8 11 12 10 8	0 -1,81 -0,07 -3,39 -0,51 1,57 0,56 0,25
8 10 12 10 8 11 12 10	0 -0,20 0 -0,25 0 -0,56 -0,71 0,29
10 8 10 12 10 8 11 12	0 -0,32 -0,02 -0,54 -0,07 0,25 -0,29 -0,90

Tabelul 10.3. DCT bi-dimensională a unui bloc de valori aleatoare

8 10 9 11 11 9 9 12	79,12 0,98 0,64 -1,51 -0,62 -0,86 1,22 0,32
11 8 12 8 11 10 11 10	0,15 -1,64 -0,09 1,23 0,10 3,29 1,08 -2,97
9 11 9 10 12 9 9 8	-1,26 -0,29 -3,27 1,69 -0,51 1,13 1,52 1,33
9 12 10 8 8 9 8 9	-1,27 -0,25 -0,67 -0,15 1,63 -1,94 0,47 -1,30
12 8 9 9 12 10 8 11	-2,12 -0,67 -0,07 -0,79 0,13 -1,40 0,16 -0,15
8 11 10 12 9 12 12 10	-2,68 1,08 -1,99 -1,93 -1,77 -0,35 0 -0,80
10 10 12 10 12 10 10 12	1,20 2,10 -0,98 0,87 -1,55 -0,59 -0,98 2,76
12 9 11 11 9 8 8 12	-2,24 0,55 0,29 0,75 -2,40 -2,40 0,06 1,14

Compresia unei imagini cu DCT presupune parcurgerea următorilor paşi:

- Se împarte imaginea în k blocuri B_i , i=1,2,…,k, de $n \times n$ (obişnuit, 8×8) pixeli fiecare.
- Se aplică DCT bi-dimensională fiecărui bloc B_i. Aceasta transformare exprimă blocul ca o combinație liniară a celor 64 de imagini de bază din Fig. 10.7. Rezultatul este un bloc, numit vector $\mathbf{W}^{(i)}$ de 64 de ponderi $w_j^{(i)}$, unde j=0, 1, ...,.63.

- Cei k vectori $\mathbf{W}^{(i)}$ (i=1, 2,...., k) sunt împărțiți în 64 de vectori de coeficienți $\mathbf{C}^{(j)}$, unde cele k elemente ale vectorului $\mathbf{C}^{(j)}$ sunt $(w_j^{(1)}, w_j^{(2)}, ..., w_j^{(k)})$. Primul vector de coeficienți $\mathbf{C}^{(0)}$ este format din cei k coeficienți DC.
- Fiecare vector de coeficienți $\mathbf{C}^{(j)}$ este cuantizat separat pentru a produce un vector cuantizat $\mathbf{Q}^{(j)}$, care reprezintă datele compresate.

Decodorul citeşte cei 64 de vectori de coeficienți cuantizați $\mathbf{Q}^{(j)}$, îi foloseşte pentru a construi k vectori de ponderi $\mathbf{W}^{(i)}$, şi aplică IDCT fiecărui vector de ponderi, pentru a reconstrui cei 64 de pixeli ai blocului B_i.

10.4.7. Transformarea Discretă Sinus

Transformarea discretă sinus, DST, este complementara DCT. DCT asigură performanțe apropiate transformatei K-L optime, în ceea ce priveşte compactarea, când corelația coeficienților ρ este mare, iar DST asigură performanțe apropiate transformatei K-L optime, când ρ este mic. Datorită acestei proprietăți, este adesea folosită ca transformată complementară a DCT în codarea de imagini şi audio.

Elementele matricei transformate pentru o DST de dimensiune $n \times n$ sunt date de

$$[S]_{i,j} = \sqrt{\frac{2}{n}} \sin \frac{\pi (i+1)(j+1)}{2n}; \; i, j = 0, 1, ..., n-1 \qquad (10.34)$$

Pentru a justifica folosirea mult mai frecventă a DCT în defavoarea DST, în continuare se prezintă diferențele dintre funcțiile sinus şi cosinus şi de ce aceste diferențe duc la o transformare sinus discretă ineficientă.

Funcția sinus este o funcție impară, iar funcția cosinus, pară. Deşi singura diferență dintre cele două funcții este faza (adică funcția cosinus este o versiune defazată a sinusului), această diferență este suficientă pentru

a le inversa paritatea. Pentru a înțelege diferența dintre DCT şi DST se examinează cazul uni-dimensional. DCT uni-dimensională, dată de ecuația (10.27), foloseşte funcția cos((2t+1)fπ/16) pentru f=0, 1.... 7. Pentru primul termen, unde f=0, această funcție devine cos(0), care este 1. Acest termen este coeficientul DC, care produce media celor opt valori de date supuse transformării.

DST este bazată în mod similar pe funcția sin((2t+1)fπ/16), având ca rezultat un prim termen nul (din moment ce sin(0)=0), care nu contribuie cu nimic la transformare, deci DST nu are un coeficient DC.

Dezavantajul acestui lucru poate fi observat când se consideră exemplul a opt valori de date identice ce trebuie transformate. Astfel de valori sunt, desigur, perfect corelate. Când sunt reprezentate grafic ele devin o linie orizontală. Aplicând DCT acestor valori, se produce doar un coeficient DC; toți coeficienții AC fiind nuli. DCT compactează toată energia datelor într-un unic coeficient DC, a cărui valoare este identică cu a datelor. IDCT poate reconstrui exact cele opt valori (cu excepția unor modificări minore date de precizia limitată de calcul). Aplicarea DST asupra aceloraşi opt valori, pe de altă parte, conduce la şapte coeficienți AC a căror sumă este o formă de undă care trece prin cele opt puncte corespunzătoare datelor, dar oscilează între aceste puncte. Acest comportament, are trei dezavantaje, în principal:

1. Energia datelor originale nu este compactată;

2. Cei şapte coeficienți nu sunt decorelați (pe când datele sunt perfect corelate);

3. Cuantizând cei şapte coeficienți se poate ajunge la o puternică scădere a calității reconstrucției realizate de DST inversă.

Exemplul 10.9.

Se urm[re;te eficiența DST, când se aplică unor secvențe corelate. Aplicând DST unei secvențe de opt valori identice, de 100, rezultă următorii

opt coeficienți (0; 256,3; 0; 90; 0; 60,1; 0; 51). Folosind aceşti coeficienți, IDST poate reconstrui valorile originale, dar este uşor de observat că în acest caz coeficienții AC nu se comportă ca cei ai DCT. Aceştia nu devin din ce în ce mai mici şi nu există şiruri de zerouri între ei.

Aplicând DST asupra următoarelor opt valori puternic corelate: 11; 22; 33; 44; 55; 66; 77; 88, rezultă un set de coeficienți şi mai puțin convenabili (0; 126,9; -57,5; 44,5; -31,1; 29,8; -23,8; 25,2), neexistând absolut deloc compactare de energie.

Aceste argumente şi exemple, împreună cu faptul că DCT produce coeficienți puternic decorelați, justifică folosirea DCT în defavoarea DST în compresia de date.

10.5. Compresia JPEG (Joint Photographers Experts Group)

Domeniul compresiei (codării) de imagini este legat de minimizarea numărului de biți necesari pentru a reface o imagine, cu aplicații în special în transmisia şi stocarea imaginilor. Aplicațiile din domeniul transmisiilor de imagini se întâlnesc în televiziunea radiodifuzată, comunicațiile spațiale, radar şi sonar, rețele de telecomunicații, transmisii fax, teleconferințe etc. Compresia imaginilor este esențială din punct de vedere al memorării (stocării) imaginilor în aplicații de imagistică medicală, în tehnica video digitală, pentru realizarea documentelor multimedia etc. Noile tehnologii de compresie a imaginilor oferă o soluție posibilă pentru integrarea aplicațiilor de imagini şi video digitale. Ratele de compresie au ajuns în prezent până la 1:100, depinzând de calitatea imaginii refăcute. Tehnica de compresie nu este suficientă pentru a putea rezolva problemele care apar în aplicațiile multimedia. Pentru a putea realiza portabilitatea aplicațiilor de imagini şi secvențe video digitale pe mai multe sisteme, este necesară implementarea

unor standarde pentru compresia datelor multimedia. Aceste standarde stabilesc modalitățile de stocare şi transmisie a datelor compresate în vederea posibilității utilizării lor. Cel mai utilizat standard de compresie a imaginilor statice este standardul JPEG, creat de Joint Photographics Experts Group. Metoda de compresie este de tip "cu pierdere", fiind concepută astfel încât să se profite de limitările în percepția video a ochiului uman. Acest standard permite setarea raportului calitate/compresie şi lucrează cu aceleaşi nivele de culoare, în număr de 24 (16,7 milioane de culori), indiferent de numărul total de culori din imagine. În momentul de față este unul dintre cele mai frecvent întâlnite formate de fişiere grafice.

Formatul JPEG este recomandat pentru afişarea de imagini redate cu o foarte mare varietate de culori sau pentru imagini de precizie fotografică. JPEG foloseşte o tehnică de compresie variabilă, care are drept rezultat obținerea de fişiere foarte mici în comparație cu alte formate.

Standardul JPEG se bazează pe transformarea informației primare din domeniul timp în domeniul frecvență. Este cunoscut faptul că imaginile sunt puternic corelate spațial, adică un pixel de imagine conține informații şi despre pixelii vecini. Corelația spațială ce caracterizează imaginile reprezintă redundanță din punct de vedere informațional şi se diminuează prin transformări matematice care au rolul de a concentra energia imaginii în cât mai puține elemente. Transformările matematice din domeniul timp în domeniul frecvență nu reprezintă în sine compresie de date. Abia operațiunile ce urmează, şi anume, cuantizarea şi codarea entropică reprezintă compresie de date. Reducerea redundanței spațiale se face atât pentru imaginea sursă originală, cât şi pentru eroarea reziduală, aşa cum se va vedea în cele ce urmează. La refacerea imaginilor, după ce acestea au fost compresate JPEG, cantitatea de informație este mai mică decât cea inițială, fără o afectare vizibilă a calității. Prin transformarea imaginii din domeniul timp, (pixeli), în domeniul frecvență se rețin doar componentele de joasă frecvență ale imaginii. Componentele de frecvență înaltă pot fi

reduse, fără o afectare deranjantă a percepţiei vizuale a imaginii. Evident că acest lucru este determinat de gradul de compresie acceptat.

JPEG este o metodă sofisticată de compresie cu pierdere pentru imagini color sau alb-negru (cu scară de gri). Un avantaj al JPEG este faptul că foloseşte mulţi parametri, permiţând astfel utilizatorului să regleze cantitatea de date pierdute şi, de asemenea, rata de compresie. Momentan reprezintă cel mai bun standard existent în materie de compresie a imaginilor statice. Standardul este implementat atât în format software cât şi hardware pentru a satisface necesităţile de prelucrare în timp real a aplicaţiilor multimedia. Creat iniţial pentru compresia imaginilor statice, standardul a fost extins şi pentru secvenţele video. Standardul realizat pentru secvenţe video se numeşte M-JPEG (Motion JPEG). Practic în cazul secvenţelor video digitale fiecare cadru este considerat ca o imagine fixă şi compresat cu standardul JPEG. Metoda nu este cea mai eficientă din punctul de vedere al mărimii ratei de compresie, dar oferă o alternativă pentru compresia video digitală. Adesea, ochiul uman nu distinge nici o degradare a imaginii chiar la o rată de compresie de 10:1 sau 20:1.

Există patru moduri principale de operare specificate de standardul JPEG :

- modul de bază, în care fiecare componentă a imaginii este codată printr-o singură scanare stânga-dreapta, respectiv sus-jos;
- codarea expandată DCT cu pierderi, în care se realizează o codare progresivă a spectrelor imaginii de intrare;
- codarea fără pierderi, în care imaginea este codată astfel încât se garantează reproducerea exactă la decodare;
- codarea ierarhică, în care imaginea este codată la rezoluţii multiple.

În continuare, se va prezenta detaliat numai primul dintre acestea, celelalte numai principial.

10.5.1. Modul de bază (baseline)

Schema bloc a algoritmului de codare JPEG este dată în Fig. 10.8.

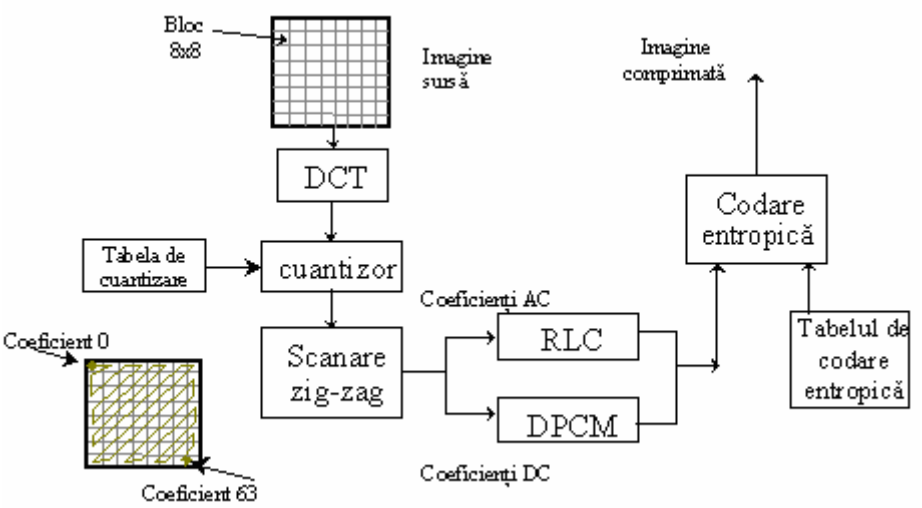


Fig. 10.8 Algoritmul de codare JPEG

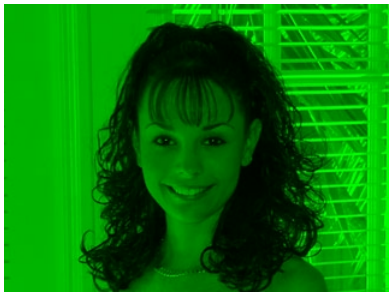
Principiul compresiei prin metode de tip DCT

Compresia cu pierderi presupune câteva etape de prelucrare, şi anume:

- **Transformarea din reprezentarea (R,G,B) în reprezentarea (Y,U,V)**

În procedurile de compresie a imaginii se preferă o reprezentare a culorii diferită de cea normală (R,G,B), şi anume, reprezentarea (Y,U,V), obținută cu relațiile 10.1. Valorile componentelor Y, U şi V sunt cuprinse între -128 şi 127. Utilitatea acestei reprezentări echivalente se poate evidenția în Fig. 10.9, în care sunt prezentate descompunerile în forma (R,G,B) respectiv (Y,U,V) ale imaginii din Fig. 10.1a.

Componenta R



Componenta G

Componenta B

Componenta Y

Componenta U

Componenta V

Fig. 10.9. Componentele R, G, B, Y, U, V ale imaginii 10.1.a

Analizând acest exemplu, se pot face câteva observații importante şi anume:

- componenta Y corespunde unei imagini alb-negru;
- componentele U şi V conțin mult mai puține detalii şi prezintă un contrast mult mai redus.

Datorită absenței detaliilor şi contrastului scăzut al componentelor U şi V, acestora li se aplică o subeşantionare cu factorul 2 pe ambele direcții, verticală şi orizontală, ținându-se cont de faptul că percepția ochiului este mai mică la semnalele de crominanță, față de cele de luminanță. Modul de realizare a subeşantionării constă în înlocuirea blocurilor de 2x2 puncte cu un singur punct care are intensitatea egală cu media celor 4. În aceste condiții imaginea va fi descrisă de componentele U' şi V', din Fig. 10.10.

Componenta U' (subesantionată)

Componenta V' (subesantionată)

Fig. 10.10 Componentele subeşantionate U' şi V'

Prin aceste operații se realizează o primă compresie, cu factorul 2:1. Astfel, reprezentarea (R,G,B) pentru imaginea din exemplu cu rezoluția de 320x240 puncte necesită 3 componente a câte 320x240=76800 elemente, adică un total de 230400 elemente (octeți). Reprezentarea (Y,U',V') necesită 320x240=76800 elemente pentru Y şi 160*120=19200 elemente pentru U' şi V' adică un total de 115200 elemente (octeți).

- **Descompunerea în blocuri**

Procedura de compresie se aplică unor blocuri de imagine de 8x8 puncte. Dacă nici una din dimensiunile imaginii nu este multiplu de 8, codorul copie ultima coloană sau linie până când lungimea finală este

multiplu de 8. Aceste linii sau coloane suplimentare sunt îndepărtate în timpul procesului de decodare.

Cele trei componente Y, U' şi V' sunt descompuse în blocuri de dimensiune 8x8. Datorită rezoluției reduse, în urma subeşantionării, rezultă că la 4 (2x2) blocuri ale componentei Y corespunde câte un singur bloc al componentelor U', respectiv V'.

În cazul formatului JPEG cele trei componente ale blocurilor de imagine sunt prelucrate întreţesut. Pentru o numerotare a blocurilor, conform Fig. 10.11, ordinea prelucrării acestora va fi Y1, Y2, Y3, Y4, U1, V1, Y5, Y6, Y7, Y8, U2, V2,....:

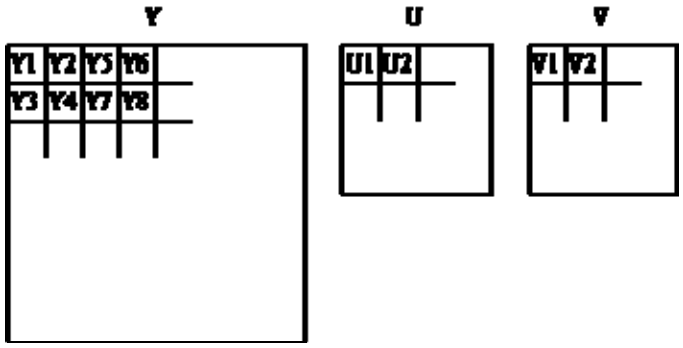


Figura 10.11. Ordinea prelucrării blocurilor

Fiecare bloc este prelucrat utilizând aceeaşi procedură.

- **aplicarea trasformatei cosinus discrete**

Valorile originale ale componentelor Y, U', V' sunt cuprinse în domeniul $[0, 2^{b-1}]$, unde *b* reprezintă numărul de biți/eşantion. 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 aplicarea DCT (Discrete Cosine Transform – Transformata Cosinus Discretă). Pentru primul nivel de codare JPEG, b=8, astfel încât valorile originale cuprinse în intervalul [0, 255] sunt deplasate în intervalul [-128, 127]. Fiecare componentă este apoi divizată

în blocuri de 8x8 pixeli, aşa cum se poate observa şi în Fig.10.8. Fiecărui bloc astfel obţinut i se aplică transformata cosinus discretă bi-dimensională, folosind ecuaţiile:

$$\text{DCT: } G_{ij} = \frac{1}{4} C_i C_j \sum_{x=0}^{7} \sum_{y=0}^{7} p_{xy} \cos\left(\frac{(2y+1)j\pi}{16}\right) \cos\left(\frac{(2x+1)i\pi}{16}\right) \quad (10.35)$$

$$\text{IDCT: } p_{xy} = \frac{1}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} C_i C_j G_{ij} \cos\left(\frac{(2y+1)j\pi}{16}\right) \cos\left(\frac{(2x+1)i\pi}{16}\right) \quad (10.36)$$

unde

$$\begin{aligned} C_i = C_j &= \frac{1}{\sqrt{2}}, \quad \text{dacă } i = j = 0 \\ C_i = C_j &= 1, \quad \text{în rest} \end{aligned} \quad (10.37)$$

Pentru a ilustra cum lucrează algoritmul, se va folosi un bloc de dimensiune 8×8 din componenta de luminanţă a imaginii Lena, aşa cum este prezentat în Tabelul 10.4.

Tabelul 10.4. un bloc de dimensiune 8×8 al imaginii "Lena"

124	**125**	**122**	**120**	**122**	**119**	**117**	**118**
121	**121**	**120**	**119**	**119**	**120**	**120**	**118**
126	**124**	**123**	**122**	**121**	**121**	**120**	**120**
124	**124**	**125**	**125**	**126**	**125**	**124**	**124**
127	**127**	**128**	**129**	**130**	**128**	**127**	**125**
143	**142**	**143**	**142**	**140**	**139**	**139**	**139**
150	**148**	**152**	**152**	**152**	**152**	**150**	**151**
156	**159**	**158**	**155**	**158**	**158**	**157**	**156**

Considerând blocul de 8×8 pixeli din Tabelul 10.4, se scade 128 din fiecare element şi aplicând DCT, se obţin coeficienţii DCT prezentaţi în Tabelul 10.5.

Tabelul 10.5. Matricea **G** a coeficienții DCT corespunzători blocului de date al imaginii Lena după deplasare

39. 88	**6. 56**	**-2. 24**	**1. 22**	**-0. 37**	**-1. 08**	**0. 79**	**1. 13**
-102. 43	**4. 56**	**2. 26**	**1. 12**	**0. 35**	**-0. 63**	**-1. 05**	**-0. 48**
37. 77	**1. 31**	**1. 77**	**0. 25**	**-1. 50**	**-2. 21**	**-0. 10**	**0. 23**
-5. 67	**2. 24**	**-1. 32**	**-0. 81**	**1. 41**	**0. 22**	**-0. 13**	**0. 17**
-3. 37	**-0. 74**	**-1. 75**	**0. 77**	**-0. 62**	**-2. 65**	**-1. 30**	**0. 76**
5. 98	**-0. 13**	**-0. 45**	**-0. 77**	**1. 99**	**-0. 26**	**1. 46**	**0. 00**
3. 97	**5. 52**	**2. 39**	**-0. 55**	**-0. 051**	**-0. 84**	**-0. 52**	**-0. 13**
-3. 43	**0. 51**	**-1. 07**	**0. 87**	**0. 96**	**0. 09**	**0. 33**	**0. 01**

Se poate observa că elementele acestei matrice au valorile mari concentrate în colțul din stânga sus, restul fiind valori mici, aproape nule. Explicația acestui fenomen este dată de faptul că transformata cosinus discretă realizează o descompunere "în frecvență". Astfel, coeficienții din colțul din stânga sus corespund frecvențelor joase - variații lente de intensitate între pixeli, iar 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. Această situație este ilustrată în Fig. 10.12.

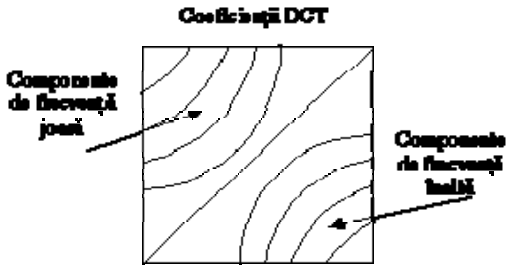


Fig. 10.12. Distribuția componentelor de frecvență într-o matrice de coeficienți DCT

- **cuantizarea matricei transformate**

Operația de cuantizare este singura în care se pierde informație. Algoritmul JPEG utilizează coeficienți de cuantizare pentru a cuantiza diferiții coeficienți de intrare. Mărimea pasului de cuantizare este organizată într-un tabel, numit tabel de cuantizare. Un exemplu de tabel de cuantizare din recomandările JPEG este prezentat în Tabelul 10.6. Fiecare valoare cuantizată este reprezentată de o etichetă. Prin cuantizare se înțelege împărțirea element cu element a matricii **G** cu o matrice de cuantizare **Q,** cu reținerea doar a părții întregi, rezultând o matrice **I**.

Tabelul 10.6. Coeficienții de cuantizare pentru luminanta (a) si pentru crominanta (b)

(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**

(b)

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**

Eticheta corespunzătoare valorii cuantizate a coeficienților G_{ij} ai transformatei este

$$I_{i,j} = \left\lfloor \frac{G_{ij}}{Q_{ij}^t} + 0,5 \right\rfloor \tag{10.36}$$

unde Q_{ij}^t este elementul (i,j) din tabelul de cuantizare şi $\lfloor x \rfloor$ este cel mai mare întreg mai mic decât x. Se consideră coeficientul G_{00} din Tabelul 10.5, care este 39, 88. Din tabelul 10.6a, Q_{00}^t este 16, deci,

$$I_{00} = \left\lfloor \frac{39,88}{16} + 0,5 \right\rfloor = \lfloor 2,9925 \rfloor = 2 \tag{10.37}$$

Valoarea reconstruită este obținută din etichetă, prin multiplicarea acesteia cu intrarea corespunzătoare în tabelul de cuantizare. Deci, valoarea reconstruită a θ_{00} va fi $I_{00} \times Q_{00}^t$ adică 2*16=32. Eroarea de cuantizare în acest caz este 39, 88-32=7, 88. Similar, din Tabelele 10.5 şi 10.6, G_{01} este 6, 56 şi Q_{01}^t este 11, deci

$$I_{01} = \left\lfloor \frac{6,56}{11} + 0,5 \right\rfloor = \lfloor 1,096 \rfloor = 1 \tag{10.38}$$

Valoarea reconstruită este 11 şi eroarea de cuantizare este 11-6,56 = 4,44. Continuând în acest mod, se obțin eşantioanele din Tabelul 10.7.

Tabelul 10.7. Coeficientii cuantizați obținuți folosind tabelul de cuantizare al coeficienților

2	**1**	**0**	**0**	**0**	**0**	**0**	**0**

-9	**0**	**0**	**0**	**0**	**0**	**0**	**0**
3	**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**

Se observă că, în urma cuantizării, în matricea coeficienților cuantizați, fenomenul de concentrare a valorilor mari în colțul din stânga sus, şi predominața valorilor mici (chiar zero) în rest, este mult mai accentuată.

Pierderea de informație se datorează realizării împărțirii cu reținerea doar a parții întregi a rezultatelor. În acest fel valorile pierd din precizie (cele mici transformându-se în zero). Efectul acestei pierderi, însă, nu este sesizabil deoarece, după cum se observă, pierderile cele mai mari sunt concentrate la nivelul coeficienților de înaltă frecvență, care au pondere redusă în imagine şi care, corespunzând detaliilor fine, sunt mai puțin observabile de către ochiul uman.

Rolul operației de cuantizare este acela de a obține cât mai multe valori nule sau mici, acestea având avantajul unei codări eficiente realizată ulterior. Transformata cosinus discretă oferă posibilitatea de a realiza această operație astfel încât efectul pierderii de informație să fie cât mai redus.

Din Tabelul 10.7 care conține eşantioanele cuantizate se poate observa că mărimea pasului creşte pe măsura îndepărtării de coeficientul DC. Deoarece eroarea de cuantizare este o funcție crescătoare de mărimea pasului, coeficienții de înaltă frecvență vor fi afectați de o eroare de cuantizare mai mare decât cei de joasă frecvență. Decizia asupra mărimii

relative a pasului de cuantizare se bazează pe modul de percepere a erorilor de sistemul vizual uman. Diferiţi coeficienţi ai transformării au importanţă perceptuală diferită. Erorile de cuantizare din coeficienţi DC şi AC de joasă frecvenţă sunt mai uşor de detectat decât erorile de cuantizare pentru coeficienţii AC de înaltă frecvenţă. De aceea se foloseşte un pas mai mare pentru coeficienţii mai puţin importanţi perceptual.

Deoarece cuantizoarele au nivelul 0 ca nivel de reconstrucţie, procesul de cuantizare funcţionează, de asemenea, şi ca operaţie de codare de prag. Toţi coeficienţii cu amplitudinea mai mică decât jumătate din mărimea pasului vor fi zero. Deoarece mărimea pasului la sfârşitul scanarii în zig-zag este mare, probabilitatea găsirii unei secvenţe lungi de zero creşte. Acest efect poate reprezenta o modalitate de modificare a ratei. Prin mărirea pasului, se poate reduce numărul de valori diferite de zero necesare pentru a fi transmise, ceea ce înseamnă o reducere a numărului de biţi necesari.

- **codarea elementelor din matricea coeficienţilor cuantizaţi**

Coeficienţii cuantizaţi sunt scanaţi zig-zag, în scopul obţinerii unei secvenţe unidimensionale, ce va fi aplicată codorului entropic. Aşa cum s-a mai precizat, primul coeficient se numeşte coeficient de curent continuu, DC, şi reprezintă media intensităţii blocului. Ceilalţi 63 de coeficienţi se numesc coeficienţi AC (coeficienţi de curent alternativ). Scanarea în zig-zag se face în ideea ordonării după spectrul de frecvenţă. Deoarece componentele de frecvenţă înaltă au valori aproximativ nule, în urma scanării zig-zag rezultă un şir de zerouri la sfârşitul secvenţei, dând posibilitatea realizării unei codări eficiente RLC (Run Length Coding) şi Huffman.

În algoritmul de compresie JPEG sunt utilizate două proceduri de codare diferite. Prima procedură este utilizată pentru codificarea elementului I_{00}, care este coeficientul DC, a doua procedură utilizându-se

pentru codificarea celorlalți 63 de coeficienți AC. Coeficientul DC este codat diferențial față de coeficientul DC din blocul anterior, folosind algoritmul DPCM (Differential Pulse Code Modulation). Coeficienții AC sunt codați RLC. Coeficienții DC şi AC astfel codați vor fi codați apoi entropic utilizând codarea Huffman.

Codarea coeficienților DC

Din Fig. 10.7 se observă că matricea de bază corespunzătoare coeficienților DC este o matrice constantă, astfel încât coeficientul DC este media (sau un pultiplu al acesteia) pixelilor din blocul de dimensiune 8×8. Valoarea medie a pixelilor din orice bloc 8×8 nu va diferi substanțial de valoarea medie din blocul 8×8 vecin; de aceea valorile coeficienților DC vor fi relativ apropiate, motiv pentru care este eficient a coda diferența între acesta şi valoarea coeficientului DC corespunzătoare blocului de 8x8 puncte anterior din aceeaşi categorie (Y, U sau V), decât etichetele în sine.

În funcție de numărul de biți folosiți la codarea valorii pixelului, numărul de valori pe care le pot lua etichetele şi, deci, diferențele dintre coeficienți, poate deveni destul de mare. Un cod Huffman pentru un alfabet aşa mare ar fi greu de implementat. Recomandarea JPEG rezolvă această problemă prin partiționarea valorilor pe care le pot lua diferențele, în clase. Mărimea acestor clase creşte în puteri ale lui doi. Astfel, clasa zero are un singur membru, clasa unu are doi membri, clasa doi are patru membri şi aşa mai departe. Numărul clasei este codat Huffman. Clasele şi cuvintele de cod Huffman corespunzătoare acestora sunt prezentate în Tabelul 10.8. Fiecare linie a Tabelul 10.8 conține numere mai mari şi mai multe decât ale liniei precedentei, neconținând numerele din linia precedentă. Linia *i* conține întregi din domeniul $[-(2^i-1),+ (2^i-1)]$, din care lipseşte intervalul din mijloc $[-(2^{i-1}-1),+ (2^{i-1}-1)]$. În codarea coeficienților DC se transmite codul corespunzător clasei *i* (liniei din tabel) în care se încadrează numărul, urmat

de *i* biți care reprezintă numărul coloanei din Tabelul 10.8 în care se află numărul ce trebuie codat.

Numărul coloanei, C, este reprezentarea pe *i* biți a valorii diferenței, în cazul valorilor pozitive sau de cei mai puțin semnificativi *i* biți ai reprezentării valorii diferenței minus 1, în complement față de 2, dacă aceasta este negativă.

În cazul valorilor din Tabelul 10.7, considerând că blocul nu este primul, eticheta corespunzătoare coeficientului DC este codată diferențial, adică se codează diferența dintre valoarea cuantizată a etichetei din acest bloc şi valoarea cuantizată a esantionului din blocul anterior. Dacă blocul de procesat este primul, se transmite în clar valoarea coeficientului DC.

De exemplu, dacă valoarea diferenței care trebuie codificată este 21 şi aparține componentei Y, se transmite întâi codul pentru clasa 5, care este 111110. Valoarea este pozitivă şi se găseşte în a 21-a coloană a liniei 5 (numărarea coloanelor începe cu 0), deci la acest cod trebuie adăugată reprezentarea pe 5 biți a numărului 21 care este 10101, codul complet fiind 11111010101.

Dacă, însă, valoarea care trebuie codată este -21, clasa este aceeaşi, diferența fiind valoarea negativă a coeficientului. Această valoare se găseşte în coloana a 10-a a clasei 5. Aşadar, fie se transmite reprezentarea pe 5 biți a numărului coloanei, 10, care este 01010, fie se transmit ultimii 5 biți ai reprezentării în complement față de 2 a numărului (-21), din care se scade 1, adică $(-22)_{2C}$= 101010, ai cărui ultimi 5 biți sunt identici cu ai numărului 10. Codul complet va fi în acest caz 11111001010.

În cazul coeficientului DC din Tabelul 10.7, dacă se presupune că eticheta corespunzătoare din blocul anterior a fost -1, atunci diferența va fi 3. Din Tabelul 10.9 se observă că această valoare aparține clasei 2. Deci, se va transmite codul Huffman pentru clasa 2, 110, urmat de o secvență de doi biți, 11, pentru a indica numărul coloanei în care se află numărul 3 sau valoarea din această clasă care a fost codată, adică, se va transmite 11011.

Tabelul 10.9. Codarea diferențelor etichetelor DC

Clasa								Codul Huffman corespunzător clasei
0				0				0
1			-1		1			10
2		-3	-2		2	3		110
3	-7	.	-4		4	.	7	1110
4	-15	.	-8		8	.	15	11110
5	-31	.	-16		16	.	31	111110
6	-63	.	-32		32	.	63	1111110
7	-127	.	-64		64	.	127	11111110
8	-255	.	-128		128	.	255	111111110
9	-511	.	-256		256	.	511	1111111110
10	-1023	.	-512		512	.	1023	11111111110
11	-2047	.	-1024		1024	.	2047	111111111110
12	-4095	.	-2048		2048	.	4095	1111111111110
13	-8191	.	-4096		4096	.	8191	11111111111110
14	-16383	.	-8192		8192	.	16383	111111111111110
15	-32767	.	-16384		16384	.	32768	111111111111111
16				32768				

Numărul cuvintelor de cod Huffman este egal cu logaritmul în baza doi al numărului de valori posibile pe care le pot lua diferențele dintre etichete. Dacă diferențele pot lua 4096 de valori posibile, lungimea codului Huffman este $\log_2 4096 = 12$, număr folosit de obicei în codare.

Codarea coeficienților AC

Ordinea în care este parcursă matricea în vederea codificării coeficienților de tip AC se alege în aşa fel încât să se profite cât mai bine de distribuția valorilor coeficienților. Se urmăreşte gruparea valorilor nule în şiruri cât mai lungi, deoarece acest fapt permite o codare mai eficientă (compresie maximă).

Deoarece valorile diferite de zero sunt concentrate într-un colț al matricei, o parcurgere de tipul linie cu linie nu este eficientă. De aceea se preferă o parcurgere în zig-zag, ordinea de extragere a elementelor din matrice fiind prezentată în Fig. 10.12.

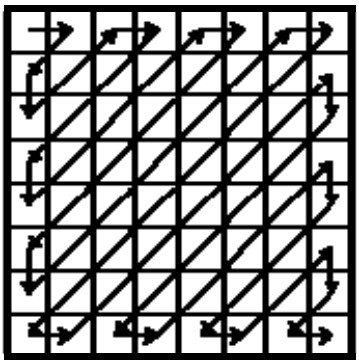


Figura 10.12. Parcurgerea în zig zag a matricei coeficienților cuantizați

Pentru matricea coeficienților din Tabelul 10.7 , coeficienții extraşi în această ordine sunt: 1, -9, 3, 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, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0. Se observă modul în care se grupează valorile nule către sfârşitul şirului. De acest fapt se va ține cont în următoarea etapă, în care se realizează codarea acestor elemente. Fiecare coeficient este un număr întreg care poate varia, potrivit standardului JPEG, între -1023 şi +1023 (valorile sunt obținute analizând cazul cel mai defavorabil şi modul în care lucrează transformata cosinus discretă).

În cazul codificării "normale" fiecare din aceşti coeficienţi poate fi reprezentat pe 11 biţi (1 bit de semn + 10 biţi valoare). Acest mod de codificare duce practic la o reprezentare pe mai mulţi biţi decât în cazul imaginii necomprimate (se folosesc 11 biţi pentru un coeficient, în loc de 8 biţi pentru un pixel). Din acest motiv, în cazul algoritmului de compresie JPEG se recurge la o altă modalitate de codificare.

În acest caz se asociază un cod combinaţiei dintre numărul de valori nule (dacă există) care precede un element diferit de zero şi valoarea acestuia din urmă. Practic se codifică perechi (Număr de zerouri, Z – Valoare, x) în locul fiecărui coeficient în parte. În realitate, din considerente de reducere a numărului de combinaţii posibile, numărul de zerouri se limitează la 16. În cazul în care există mai mult de 16 elemente nule se emit coduri speciale (ZRL) care semnifică 16 zerouri care nu sunt urmate de un element diferit de zero. De exemplu, 18 zerouri urmate de un element cu valoarea -21 se vor coda printr-un ZRL urmat de codul corespunzător perechii (2,-21). De asemenea, în cazul în care dintr-un anumit punct al şirului până la sfârşitul acestuia nu mai există nici un element diferit de zero, se emite un alt cod special (EOB) în locul tuturor valorilor nule rămase.

În concluzie, pentru fiecare număr x, precedat de Z zerouri, care formează perechea (Z,x), codorul trebuie

- să găsească numărul de zerouri consecutive Z, care îl preced;
- să determine linia *i* şi coloana C din Tabelul 10.9 corespunzătoare numărului;
- să identifice din Tabelul 10.10, după numărul Z şi clasa *i*, codul Huffman corespunzător perechii;
- la cuvântul de cod Huffman găsit se concatenează reprezentarea pe *i* biţi a coloanei C.

Tabelul 10.10. Codarea HUFFMAN pentru coeficienţii AC

Z/*i*	Cuv. de cod	Z/*i*	Cuv. de cod	Z/*i*	Cuv. de cod
0/0(EOB)	1010			F/0(ZRL)	11111111001
0/1	00	1/1	1100	F/1	1111111111110101
0/2	01	1/2	11011	F/2	1111111111110110
0/3	100	1/3	1111001	F/3	1111111111110111
0/4	1011	1/4	111110110	F/4	1111111111111000
0/5	11010	1/5	11111110110	F/5	1111111111111001
⋮					

Ţinând cont de aceste observaţii, pentru coeficienţii din Tabelul 10.7, rezultă următorul set de coduri care trebuie generate:

(0,1)

(0,-9)

(0,-3)

(0,0,...0) = EOB

De exemplu, se doreşte codificarea simbolului (0,1) din şirul de mai sus. Prima valoare, 1, aparţine categoriei 1. Deoarece nu sunt zerouri care să preceadă această valoare, se transmite codul Huffman corespunzător lui 0/1, care, din Tabelul 10.10, este 00. Se concatenează cu acest cod un singur bit de 1 pentru a indica faptul că valoarea transmisă este 1. Cuvântul de cod pentru perechea (0,1) este deci 001. Analog, -9 este al şaselea element din clasa 4. De aceea se transmite şirul binar 1011, care este codul Huffman pentru 0/4, urmat de 0110, pentru a arăta că -9 este cel de-al şaselea element din clasă. Următoarea valoare este 3, care aparţine clasei 2, deci se transmite codul Huffman 01, corespunzător lui 0/2, urmat de 11. Toate esantioanele după acest punct sunt zero, astfel încât se transmite codul Huffman EOB, care în acest caz este 1010.

Pentru exemplul considerat, datele sunt 11011 001 10110110 0111 1010, adică se foloseşte un număr de 24 de biți pentru reprezentarea blocului de dimensiune 8x8, adică o medie de 3/8 biți/pixel.

Se precizează că tabele prezentate pentru cuantizarea şi codarea coeficienților DC şi AC nu sunt singurele recomandate de standardul JPEG, acesta limitând însă la 4, tabelele de coduri Huffman pentru codarea coeficienților AC pentru luminanță şi crominanță. Modul JPEG de bază foloseşte numai două asemenea tabele.

Lanțul de decodare JPEG este parcurs în ordine inversă codării. Astfel, imaginea compresată JPEG este supusă în primul pas unui decodor entropic. După decodarea entropică, se aplică decuantizarea, folosind aceiaşi coeficienți care au fost folosiți şi la cuantizare, prezentați în Tabelele 10.6 a,b. În urma decuantizării se obțin coeficienții transformatei cosinus discrete din Tabelul 10.11. Aceşti coeficienți sunt "trimişi" blocului de transformare cosinus discretă inversă, IDCT, care aplică transformarea dată de relația 10.36. Acestora li se adună 128 şi se obține blocul reconstruit prezentat în Tabelul 10.12. Calitatea acestei imagini depinde de numărul de coeficienți păstrați la codare. Dacă se vor păstra toți coeficienții nenuli, atunci imaginea reconstruită va fi foarte asemănătoare cu originalul.

Tabelul 10.11. Valorile coeficienților după decuantizare

32	**11**	**0**	**0**	**0**	**0**	**0**	**0**
-108	**0**	**0**	**0**	**0**	**0**	**0**	**0**
42	**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**

Tabelul 10.12. Blocul reconstruit

123	**122**	**122**	**121**	**120**	**120**	**119**	**119**
121	**121**	**121**	**120**	**119**	**118**	**118**	**118**
121	**121**	**120**	**119**	**119**	**118**	**117**	**117**
124	**124**	**123**	**122**	**122**	**121**	**120**	**120**
130	**130**	**129**	**129**	**128**	**128**	**128**	**127**
141	**141**	**140**	**140**	**139**	**138**	**138**	**137**
152	**152**	**151**	**151**	**150**	**149**	**149**	**148**
159	**159**	**158**	**157**	**157**	**156**	**155**	**155**

Deşi reducerea este substanțială, de la 8 biți pe pixel la 3/8 biți pe pixel, reproducerea este remarcabil de apropiată de original.

Dacă se doreşte o reproducere cu acuratețe mai mare, se poate face acest lucru cu prețul creşterii ratei, micşorând mărimea pasului de cuantizare din tabelul de cuantizare. Acest lucru va determina o creştere a numărului de biți transmişi. Similar, se poate scădea rata, prin creşterea pasului de cuantizare, cu prețul creşterii distorsiunilor.

10.5.2. Modul de codare cu pierderi expandat

Există trei diferențe principale între modul de bază şi cel expandat, şi anume:

- Modul expandat poate folosi 8-12 biți;
- Modul expandat poate folosi ca şi codare entropică atât codarea Huffman, cât şi codarea aritmetică;
- Modul expandat poate folosi atât codarea progresivă cât şi cea secvențială.

În multe dintre aplicațiile de compresie a imaginilor apare dezavantajul că, datorită rezoluțiilor mari la care sunt reprezentate imaginile, timpul de codare, de transmisie şi, respectiv, de decompresie este de ordinul minutelor. În astfel de aplicații se poate aplica un algoritm de compresie progresiv, care inițial produce o imagine brută, după care, prin scanări succesive, se îmbunătăţeşte calitatea imaginii.

Dacă se presupune că valoarea unui pixel este apropiată de cea a vecinului său, se poate folosi valoarea pixelului învecinat pentru a predicta valoarea pixelului de codat. Ideea este aceea de a îndepărta orice redundanță care poate exista. Diferența dintre valoarea actuală şi cea prezisă este codată şi transmisă. Această diferență se numeşte *eroare de predicție* sau *reziduu*. Receptorul foloseşte acelaşi pixel învecinat pentru a face predicția pentru acel pixel şi acelaşi algoritm de predicție ca şi emițătorul. Din moment ce se foloseşte acelaşi pixel şi acelaşi algoritm pentru a face predicția, receptorul ar trebui să genereze aceeaşi valoare de predicție ca şi emițătorul. Această valoare, când este adăugată la eroarea de predicție dată de emițător, ar trebui să conducă exact la pixelul original recuperat. Dacă algoritmul folosit pentru predicție constă în alegerea unei combinații liniare a pixelilor învecinați, atunci această abordare se numeşte *predicție liniară*. Pentru ca, atât emițătorul cât şi receptorul să folosească acelaşi pixel pentru generarea predicției, trebuie să se impună o ordine a pixelilor. În general, se presupune că pixelii unei imagini sunt generați linie cu linie, de la stânga la dreapta, şi de sus în jos. Acest procedeu se numeşte ordine de scanare de tip rastru.

De exemplu, se consideră că " imaginea" din Fig. 10.13a este imaginea originală. Dacă se foloseşte vecinul stâng al fiecărui pixel ca predicție a acelui pixel, eroarea de predicție poate fi reprezentată ca o imagine reziduală, după cum se arată în Fig. 10.13b.

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

(a)

4	4	-4	4	-7	0	0	0
1	1	2	2	-1	-4	0	0
8	-4	1	0	0	0	0	0
2	2	4	-3	2	2	-4	0
2	2	2	1	0	0	2	0
2	0	0	1	1	5	-2	-4
3	0	3	0	0	0	1	0
7	0	0	0	-1	1	0	0

(b)

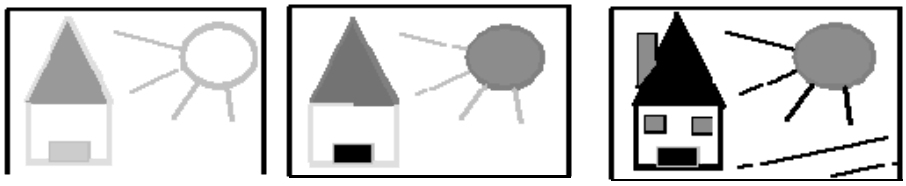
Fig. 10.13. (a) Imaginea originală şi (b) eroarea ei de predicție

Se observă numărul mare de zerouri din imaginea reziduală. Într-o imagine în care există cu precădere acest tip de redundanță sau structură, adică pixeli vecini au valori apropiate una de alta, această abordare va duce la o imagine reziduală care constă în principal din zerouri şi numere mici. Imaginea reziduală poate fi codificată în general cu mult mai puțini biți decât imaginea originală. În acest exemplu se foloseşte vecinul din stânga pentru predicție. Dar se poate folosi, de asemenea, vecinul de sus sau o combinație avantajoasă a lor. Schemele de predicție pentru pixelul curent care se bazează pe vecini aflați pe direcții diferite sunt cunoscute sub denumirea de scheme de predicție bi-dimensionale. În ciuda simplității lor, tehnicile de predicție liniară sunt eficiente, iar performanțele lor sunt surprinzător de apropiate de performanțele obținute cu ajutorul altor tehnici mai sofisticate.

Modul de compresie progresiv JPEG, se obține prin realizarea unui set de subimagini şi fiecare subimagine va fi codată cu un set specific de coeficienți DCT. În consecință, codorul DCT va trebui să aibă un buffer în care datele (coeficienții DCT ai subimaginilor) să fie memorate înainte de codarea entropică. În Fig. 10. 14 este prezentat modul de formare a imaginilor codate progresiv JPEG.

Compresia progresivă JPEG poate fi obținută folosind trei tipuri de algoritmi:

- un algoritm progresiv de selecție spectrală;
- un algoritm progresiv de aproximări succesive;
- un algoritm progresiv combinat.



a) b) c)

Fig. 10.14. Compresia progresivă JPEG

Algoritmul progresiv de selecție spectrală grupează coeficienții DCT în câteva benzi de frecvență. De obicei, întâi sunt transmişi coeficienții de joasă frecvență, după care urmează coeficienții de înaltă frecvență. În Fig. 10.15 este prezentat un exemplu în care coeficienții DCT sunt împărțiți în patru benzi de frecvență. În banda 1 se găsesc numai coeficienții de curent continuu (DC), banda 2 cuprinde primii coeficienți AC, AC1, AC2, banda 3 conține coeficienții AC3, AC4, AC5, AC6, iar banda 4, coeficienții AC7,... , AC63.

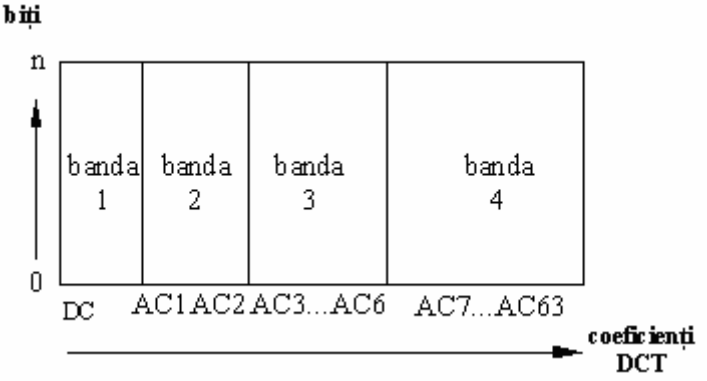


Fig. 10.15. Algoritmul progresiv cu selecție spectrală

Algoritmul progresiv cu aproximări succesive se bazează pe transmisia inițială, cu precizie scăzută, a tuturor coeficienților DCT, după care se transmite restul informației pentru a se obține o imagine de calitate superioară. În Fig. 10.16 este prezentat un exemplu în care coeficienții DCT sunt împărțiți în trei benzi de aproximări succesive: banda 1, care cuprinde toți coeficienții DCT împărțiți la 4, banda 2, cuprinzând coeficienții împărțiți la 2 şi banda 3 conținând toți coeficienții DCT la precizia nominală.

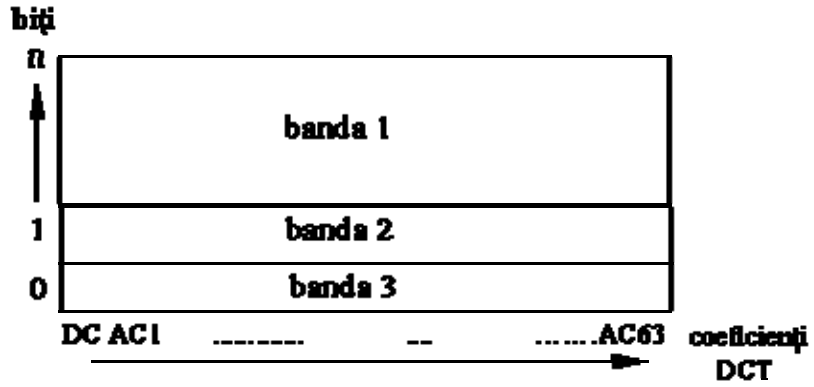


Fig. 10.16. Algoritmul progresiv cu aproximări succesive

Algoritmul progresiv combinat este o combinație a primilor doi algoritmi. În Fig. 10.17 este prezentat planul coeficienților DCT ai unei imagini, după aplicarea algoritmului combinat. S-au obținut 8 subsecvențe de coeficienți DCT, care vor fi transmişi în ordinea numerotării, şi anume, coeficienții DC1 apoi AC1 şi aşa mai departe.

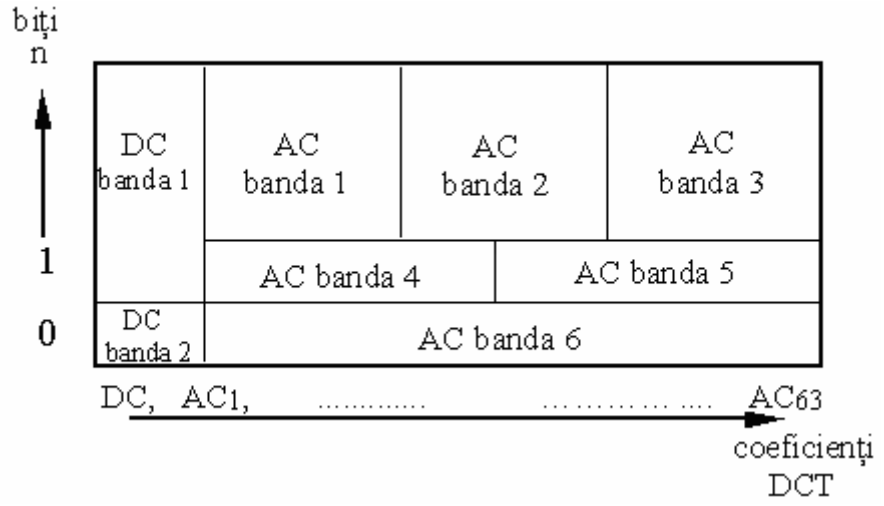


Fig. 10.17. Algoritmul progresiv combinat

În Fig. 10.18 este prezentată imaginea decompresată progresiv JPEG pentru diferite momente de timp.

a) imagine la momentul t1

b) imagine la momentul t2>t1

c) imagine la momentul t3>t2

d) imagine originală (t=t4)

Fig. 10.18. Etapele de decompresie ale imaginii folosind algoritmul progresiv

10.5.3. Compresia secvențială JPEG fără pierderi

Standardul JPEG permite şi folosirea unui algoritm de compresie fără pierderi, respectiv un algoritm de compresie predictivă în locul transformării DCT. În Fig. 10.19 este prezentată schema bloc a unui codor JPEG fără pierderi. Blocurile de cuantizare neuniformă din schema standard de compresie JPEG sunt înlocuite în cazul de față cu un bloc de codare predictivă.

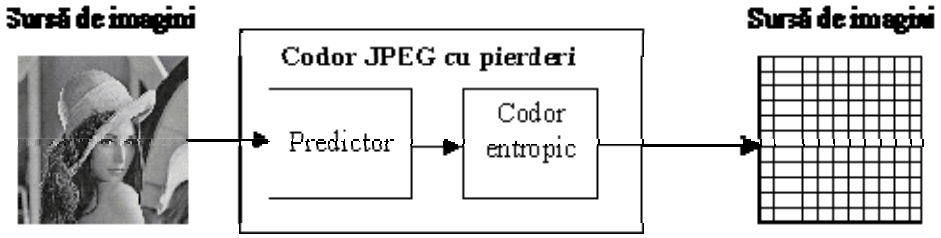


Fig. 10.19. Schema generalizată o codorului JPEG fără pierderi

Blocul predictor lucrează astfel: se calculează o predicție a eşantionului X' pe baza eşantioanelor precedente A, B şi C, figurate]n Fig. 10.20, după care se calculează diferența dintre eşantionul original X şi cel prezis: DX=X-X'.

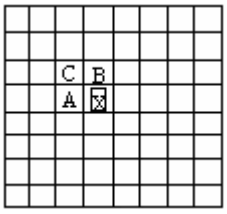


Fig. 10.20. Poziționarea eşantioanelor pe baza cărora se calculează predicția

Această diferență este codată entropic cu ajutorul unui codor Huffman. În Fig. 10.21 sunt prezentați paşii parcurşi în algoritmul de codare.

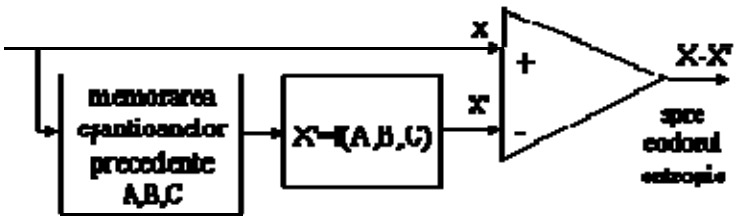


Fig. 10.21. Schema de codare cu pierderi JPEG

În Tabelul 10.13 sunt date câteva formule folosite în predicție.

Tabelul 10.13. Predictori folosiți
pentru compresia cu pierderi

Index	formula de predicție
0	fără predictor
1	X=A
2	X=B
3	X=C
4	X=A+B-C
5	X=A+(B-C)/2
6	X=B+(A-C)/2
7	X=(A+B)/2

10.5.4. Compresia JPEG ierarhică

Compresia ierarhică JPEG permite o reprezentare progresivă a imaginii decodate, într-un mod similar cu algoritmul progresiv, dar, în plus față de acesta, permite imagini codate cu rezoluții multiple. Se creează astfel, un set de imagini compresate, începând cu imagini de rezoluție mică

şi continuând cu imagini a căror rezoluţie creşte către rezoluţia imaginii originale. Acest proces se numeşte subeşantionare sau codare piramidală. În Fig. 10.22 este figurat un astfel de algoritm.

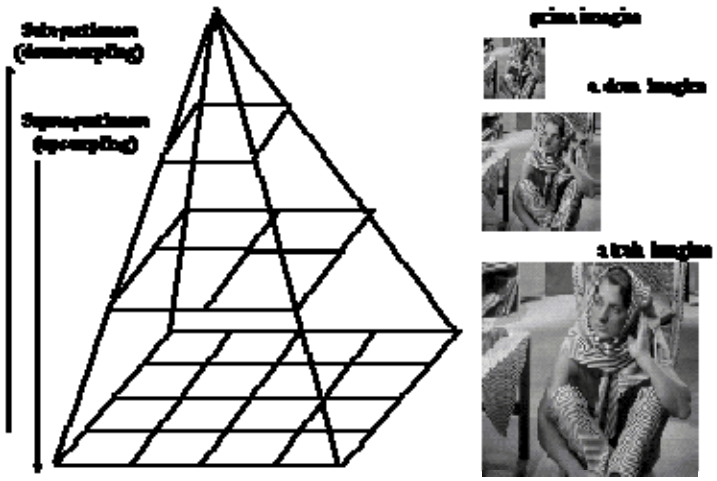


Fig. 10.22. Codare ierarhică JPEG

După procesul de subeşantionare, fiecare imagine de joasă rezoluţie se scalează la imaginea cu rezoluţie imediat superioară prin metoda inversă, numită supraeşantionare, fiind folosită ca predictor pentru următoarea etapă. Algoritmul de compresie ierarhică constă din următorii paşi:

- filtrarea şi subeşantionarea imaginii originale cu factorul dorit (multiplu al lui 2), de-a lungul fiecărei dimensiuni (vertical, orizontal);
- codarea imaginii de rezoluţie mică folosind tehnicile de codare DCT secvenţială, DCT progresivă sau codarea fără pierderi;
- decodarea imaginii de rezoluţie redusă, interpolarea şi supraeşantionarea cu un factor de 2, pe direcţia orizontală şi/sau verticală, folosind un filtru de interpolare identic cu cel de la decodor;

- folosirea imaginii supraeşantionate ca predictor al imaginii originale, şi codarea diferenţelor dintre cele două imagini, folosind una din tehnicile amintite;
- se repetă algoritmul până când se codează imaginea la rezoluția inițială.

Codarea ierarhică necesită o zonă de memorie destul de mare pentru a putea implementa algoritmul, dar avantajele sunt acelea că imaginea codată este imediat disponibilă la diferite rezoluții.