Cuvânt înainte

La ora actuală asistăm la o dezvoltare remarcabilă a sistemelor informatice, în majoritatea domeniilor economice şi sociale, şi implicit a proiectării de sisteme informaţionale capabile să preia şi automatizeze în mare parte sarcinile operatorului uman sau să-l asiste pe acesta în rezolvarea unor probleme. În acest context a crescut şi cererea de implementare a unor componente de viziune computerizată în sistemele informatice. Aceste componente trebuie să fie capabile să prelucreze şi îmbunătăţească imaginile digitale achiziţionate din scene ale realităţii fizice, precum şi să extragă direct anumite informaţii cantitative sau calitative din aceste imagini.

Lucrarea de faţă îşi propune să abordeze în special perspectiva practică a implementării diverselor componente care compun astfel de subsisteme de viziune computerizată, începând cu etapa de achiziţie, redare şi stocare a imaginilor şi secvenţelor video digitale, continuând cu componentele de preprocesare şi îmbunătăţire a imaginii achiziţionate şi filtrare a zgomotului posibil suprapus pe o astfel de imagine, urmând cu segmentarea şi analiza morfologică şi încheind în fine cu extragerea de informaţii cantitative. Ţinând cont de importanţa practică în aplicaţiile de comunicaţii a compresiei imaginilor şi secvenţelor video digitale, ultimele subcapitole ale cărţii sunt dedicate implementării practice a transformărilor de imagini cele mai folosite în standardele de compresie: transformata Wavelet şi transformata cosinus discretă, precum şi prelucrării imaginilor JPEG direct în domeniul comprimat – direcţie de interes la ora actuală pentru implementarea practică în timp real a diverşilor algoritmi de prelucrare a imaginilor digitală. Accentul este pus pe modul de proiectare şi implementare practică a câtorva algoritmi de bază adresând aspectele mai sus menţionate în mediul de dezvoltare LabView folosind biblioteca IMAQ Vision (bibliotecă dezvoltată în LabView pentru prelucrări de imagini). Acest mediu oferă avantajele uşurinţei de integrare a modulelor realizate în aplicaţii industriale (cerinţă de bază a componentelor de viziune computerizată) şi de simplitate a implementării unei mari varietăţi de algoritmi prin interconectarea grafică a modulelor existente. Pentru achiziţia imaginilor este prezentată o aplicaţie folosind placa de achiziţie IMAQ PCI 1411, care poate fi însă uşor adaptată şi la alte tipuri de plăci de achiziţie de imagini pentru care există drivere National Instruments.

Nivelul prezentării permite atât studenţilor cât şi altor categorii de cititori interesaţi să se familiarizeze cu câţiva algoritmi practici şi simpli de prelucrare a imaginilor digitale utilizaţi în sistemele de viziune computerizată, oferind o posibilă bază de pregătire practică pentru activităţi de dezvoltare şi cercetare în acest domeniu.

Dat fiind publicul-ţintă principal vizat – studenţii de la specializările *Electronică aplicată* şi *Telecomunicaţii* ai ***facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei,*** cartea este structurată pe 14 capitole, prezentate sub forma mai intuitivă a unor lucrări practice. Am încercat pe cât posibil să adoptăm, în prezentarea algoritmilor şi aplicaţiilor, un stil clar, direct şi concis, apropiat de cititor.

Cu speranţa că materialul elaborat va fi util în formarea viitorilor specialişti în domeniul dezvoltării de subsisteme şi aplicaţii de viziune computerizată (ingineri electronişti şi de telecomunicaţii precum şi altor cititori interesaţi), autorii vă mulţumesc pentru că v-aţi aplecat asupra lui şi vă incurajează să le comunicaţi observaţiile, sugestiile şi comentariile dumneavoastră.

Cluj-Napoca, 1 martie 2007

1 Formate de imagine

1.1 Noţiuni teoretice

Pentru a reuşi să stocăm pe suportul magnetic al calculatorului informaţia de imagine, aceasta trebuie să fie concepută în mod special şi adaptată resurselor hard ale acestuia. În primul rând trebuie ţinut cont că pentru a stoca un număr cât mai mare de imagini este neapărat nevoie să eliminăm informaţia redundantă. Aceasta înseamnă că nu toată informaţia furnizată de către placa de achiziţie este neapărat utilă. Acest lucru se realizează prin compresia imaginilor, dar aici conceptul de compresie nu înseamnă neapărat o pierdere de informaţie, ci posibilitatea de a reconstitui complet întreaga informaţie iniţială. În continuare, vom prezenta câteva dintre cele mai cunoscute formate de imagine, şi anume formatul BMP, GIF şi PNG. Tentaţia de a realiza un format de imagine propriu este foarte mare, mai ales în cazul firmelor de marcă care doresc să-şi impună standardele, ajungându-se astfel la peste 30 formate de imagine **diferite**. Acestea pot fi apreciate după mai multe criterii cum ar fi, ***gradul de compresie*** pe care îl realizează şi ***timpul necesar împachetării şi despachetării*** acestora. Cele mai importante formate din această categorie sunt: PCX, IMG, TARGA, IBM, TIF, WPG, JPG, PIC, CDR, PCD şi altele mai puţin importante şi mai ales recunoscute.

1.2 Formatul BitMap (BMP)

Deşi după cum am spus varietatea formatelor este foarte mare, formatul BMP s-a impus ca cel mai cunoscut, fiind cel mai simplu format. Numele sub care este cunoscut înseamnă „***hartă de biţi"*** şi în realitate seamănă cu o hartă foarte detaliată a ecranului.

Formatul conţine trei părţi mai importante:

- prima se numeşte **BitMapHeader**, şi ne furnizează importante informaţii privitoare la lungimea fişierului, numărul de pixeli dintr-o linie de imagine, numărul de linii folosite, numărul de culori şi altele.
- în a doua parte vom găsi informaţii privitoare la paleta de culori **RGB QUAD**, dacă aceasta există
- iar în ultima este prezentă informaţia de imagine **INFO IMG** fără terminator între linii.

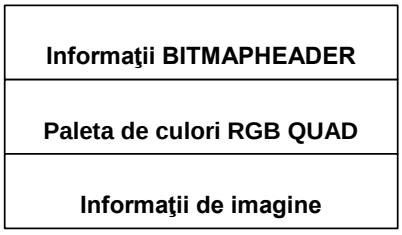


Figura 1.1 Structura formatului bitmap

Pe disc, fişierul imagine arată ca o înlănţuire de octeţi care pot fi vizualizaţi cu ajutorul programelor software de citire şi editare binară a fişierelor cum sunt cele oferite de Symantec sau Technologismiki (e vorba de Hackman cu care se va lucra în partea practică), şi care ne prezintă structura acestor fişiere în format hexazecimal. Astfel, se poate considera că primul octet de pe disc ce aparţine fişierului de imagine este octetul de referinţă în raport

cu care vor fi adresaţi ceilalţi. Acestui octet îi este atribuit un index, toţi octeţii fiind numerotaţi în raport cu acest prim octet. În acest caz ne putem imagina structura fişierului ca fiind asemănătoare modului de organizare a memoriei interne din calculator. În Figura 1.1 este reprezentată o hartă a structurii fişierului BMP. Intuitiv putem privi un fişier BitMap format din aceste trei blocuri prezentate anterior, primul bloc şi al doilea fiind foarte importante. Aici dacă apare o eroare în cele mai multe cazuri reprezentarea imaginii devine imposibilă sau în cel mai fericit caz conţinutul imaginii se strică apărând viraţii de culoare la acei pixeli care vor indexa regiştrii eronaţi. Ultimul bloc INFO IMG nu este atât de sensibil la erori şi acest lucru oferă structurii BitMap o mai mare siguranţă la stocarea informaţiei de imagine comparativ cu celelalte formate care folosesc tehnici de compresie.

Structura BitmapHeader este formată din două substructuri:

- BitmapFileHeader – primii 14 octeţi
- BitmapInfoHeader – următorii 40 octeţi

În continuare sunt prezentate structurile BitmapFileHeader şi BitmapInfoHeader.

BitmapFileHeader

- **Offset 00 număr octeţi 2 name "bftype":** reprezintă tipul fişierului (BMP), în Windows 3.x sau OS/2 aici se află depusă valoarea (4D42h).
- **Offset 02 număr octeţi 4 name "bfsize":** reprezintă mărimea fişierului în bytes (maxim ffffffffh)
- **Offset 06 număr octeţi 2 name "bfReserved1":** rezervat utilizatorului (0000h)
- **Offset 08 număr octeţi 2 name "bfReserved2":** rezervat utilizatorului (0000h)
- **Offset 10 număr octeţi 4 name "bfOffBits":** reprezintă offsetul de la începutul fişierului de unde începe structură de date imagine BITMAP în bytes.

BitmapInfoHeader

- **Offset 14 număr octeţi 4 name " biSize":** reprezintă dimensiunea BITMAPINFOHEADER, în octeţi.
- **Offset 18 număr octeţi 4 name "biWidth":** reprezintă lungimea imaginii în pixeli (numărul de pixeli dintr-o linie).
- **Offset 22 număr octeţi 4 name "biHeight":** reprezintă înălţimea imaginii în pixeli (numărul de linii).
- **Offset 26 număr octeţi 2 name "biPlanes":** reprezintă numărul de planuri de culoare (R,G,B); pentru 256 culori nu există planuri de culoare ci paletă de culori, deci această variabilă se află setată la 01h.
- **Offset 28 număr octeţi 2 name "biBitCount":** reprezintă numărul de biţi necesari pentru memorarea unui pixel, care specifică indirect câte culori conţine informaţia BitMap. Valorile uzuale sunt:
  - **1**: Bitmap monocrom 2 colori
  - **4**: BMP cu maxim 16 culori
  - **8**: BMP cu maxim 256 culori
  - **24**:BMP true color maxim 16,7 milioane culori . În acest caz nu există o tabelă de culori în structura BitMap, informaţia de imagine fiind organizată în trei plane de culoare (vezi paragraful 1.0.0).
- **Offset 30 număr octeţi 4 name "biCompression":** reprezintă tipul compresiei utilizate, valorile standard sunt:
  - **BIRGB** (valoare 00h) BitMap necomprimat (4 biţi/pixel).

- **BIRLE 4** (valoare 01h) se utilizează o codare cu pas variabil cu maxim 16 culori (4 biţi/pixel)
- **BIRLE 8** (valoare 02h) se utilizează o codare cu pas variabil cu maxim 256 culori (8 biţi/pixel).

- **Offset 34 număr octeţi 4 name "bisizeImage":** reprezintă mărimea structurii BMP în bytes.
- **Offset 38 număr octeţi 4 name "biXPelsPerMeter":** rezoluţia orizontală în pixeli/metru.
- **Offset 42 număr octeţi 4 name "biYPelsPerMeter":** rezoluţia verticală în pixeli/metru.
- **Offset 46 număr octeţi 4 name "biClvUsed":** numărul culorilor utilizate.
- **Offset 50 număr octeţi 4 name "biClvImportant":** numărul de culori semnificative în reprezentarea imaginii.

În continuare vom prezenta structura paletei de culori. Această informaţie începe de la **offset 54** şi este cunoscută sub numele de **RGB QUAD**, fiind organizată în partiţii de patru octeţi, câte o partiţie pentru fiecare culoare folosită, şi un octet este rezervat utilizatorului. Să presupunem astfel că avem un număr de 236 culori folosite, (indicat la poziţia 46 în BitMap Header) vor rezulta în **RGB QUAD** 236 partiţii a câte patru octeţi fiecare, aceasta fiind de fapt şi lungimea **RGB QUAD** pentru imaginea citită. Această lungime este diferită de la o imagine la alta, iar în cazul în care avem o imagine cu o singură culoare, **RGB QUAD** are lungimea de patru octeţi. Între aceştia se va găsi informaţia care face referire la modul de programare a paletei de culori (doar în cazul în care avem o imagine cu maxim 256 culori) şi arată astfel:

Structura unei partiţii **RGB QUAD**:

- **Offset 00h număr octeţi 1 name "rgbBlue":** intensitatea componentei de bază pentru albastru.
- **Offset 01h număr octeţi 1 name "rgbGreen":** intensitatea componentei de bază pentru verde.
- **Offset 02 număr octeţi 1 name "rgbRed":** intensitatea componentei de bază pentru roşu.
- **Offset 03h număr octeţi 1 name "rgbReserved":** nu este folosit, (00h).

1.2.1 Formatul BitMap necomprimat

Cu valoare zero pentru "**biCompression**", datele sunt ordonate de jos în sus primul pixel care reprezintă un pixel din imagine fiind cel indicat de "**bfOffBits**" şi care este pixelul din stânga jos de coordonată (0, MAXy). Astfel, primul pixel citit de aici va fi pus în partea inferioară a ecranului. Rezultă deci, că în mod normal, imaginea va apare pe ecran de jos în sus.

1.2.2 Formatul BitMap de 24 biţi

Are o structură cu totul specială care se pretează unor imagini în format "**TRUE COLOR**" de o calitate superioară. În acest caz nu mai există o informaţie referitoare la paleta de culori deoarece ea practic nu mai este necesară, calculatorului lipsindu-i o paletă de culori. În acest caz informaţia de culoare este memorată pentru fiecare pixel în parte. Se poate practic reprezenta orice imagine fără a programa paleta, acest lucru fiind posibil datorită numărului foarte mare de combinaţii de culori (16,7 milioane - 2^{24}).

Informaţia citită din fişierul BMP format 24 biţi reprezintă direct cele trei componente de culoare **R G B** care pot lua valori între 0 şi 255, putând în acest caz avea un

număr de 256 nivele diferite de gri, pentru fiecare plan de culoare. Avantajul folosirii acestui mod grafic iese în evidenţă la vizualizarea formatelor AVI (secvenţe video) care memorează o secvenţă de imagine, fără a mai sintetiza o paletă de culori care ar creşte ca mărime o dată cu creşterea duratei secvenţei. Aceste moduri grafice sunt cele folosite în High Digital Television având posibilitatea redării oricărui conţinut de culoare dintr-o imagine.

1.2.3 Calculul dimensiunii unui fişier BMP

După cum am arătat mai sus un fişier BMP este format din trei părţi: bitmap header, paleta de culori şi informaţia de imagine. De aici rezultă că dimensiunea unui fişier BMP va fi egală cu suma dimensiunilor celor trei câmpuri. Astfel, spaţiul de memorie ocupat de bitmap header este de 54 de octeti (bytes). În cazul în care se foloseşte pentru reprezentarea imaginii o paletă de culori, pentru fiecare culoare se vor aloca câte patru octeţi (câte unul pentru fiecare componenta de culoare roşu, albastru şi verde, urmaţi de încă unul cu valoarea 0, octet rezervat de standard).

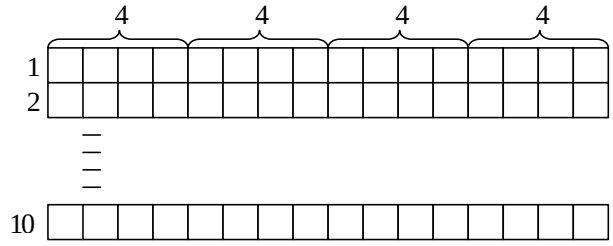
Mărimea informaţiei de imagine este direct proporţională cu dimensiunea imaginii. Astfel, pentru fiecare pixel al imaginii se vor aloca câte b-biti (unde b reprezintă poziţia culorii în paletă, b este reprezentat de un număr de biţi egal cu numărul de biţi pe care este reprezentată fiecare culoare din paleta de culori împărţit la trei) respective 24 biţi (pentru imaginile true color, cele care nu folosesc paleta de culori).

Între calculul teoretic şi cel practic (dimensiunea reală a unui fişier BMP) pot să existe diferenţe. Acestea apar atunci când numărul de pixeli pe orizontală nu este divizibil cu patru (multiplu de patru). Prin urmare avem două cazuri.

1. *Dimensiunea liniei unei imagini **este multiplu de patru**, caz în care rezultatul calculului teoretic va coincide cu dimensiunea fişierului:*

Ex. O imagine de 16 pixeli lăţime şi 10 înălţime, având o paletă de 256 de culori.

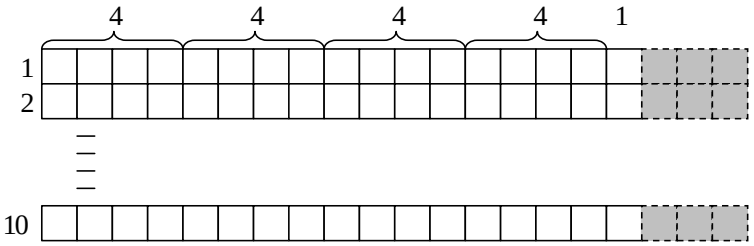
$$D_{teoretic} = D_{header} + D_{paleta} + D_{\text{inf}.img.} = 54octeti + 4*256octeti + 16*10octeti = 1238octeti$$
$$D_{practic} = D_{header} + D_{paleta} + D_{\text{inf}.img.} = 54octeti + 4*256octeti + 16*10octeti = 1238octeti$$



2. *Lăţimea imaginii **nu este multiplu de patru**.*

Ex: Dimensiunea liniei este de 17 pixeli lăţime şi 10 înălţime, având aceeaşi paletă de culori:

$$D_{teoretic} = D_{header} + D_{paleta} + D_{\text{inf}.img.} = 54octeti + 4*256octeti + 17*10octeti = 1248octeti$$
$$D_{practic} = D_{header} + D_{paleta} + D_{\text{inf}.img.} = 54octeti + 4*256octeti + 20*10octeti = 1278octeti$$



În acest caz se observă că, se va lua în calcul nu dimensiunea liniei, ci a primului număr divizibil cu 4 mai mare decât lăţimea liniei. Conform standardului, cei 1, 2 sau 3 pixeli adăugaţi în fiecare linie se completează cu 0, ei nefiind interpretaţi la afişare. De remarcat că în headerul BMP se reţine dimensiune reală a imaginii, nu cea la care au fost adăugaţi pixelii de completare (în cazul nostru 17 nu 20 cât este multiplu de 4).

1.3 Formatul GIF – noţiuni teoretice

1.3.1 Noţiuni introductive

Formatul de imagine GIF este un standard definit de ***CompuServe's*** pentru imaginile color. Acest format (GIF – **G**raphics **I**nterchange **F**ormat) permite afişarea imaginilor, la o calitate şi rezoluţie înaltă, pe o serie de echipamente hardware. Formatul GIF este realizat cu scopul de a suporta dezvoltările actuale şi viitoare în domeniu. În continuare este prezentat formatul unui fişier ***GIF***.

1.3.2 Formatul fişierului GIF

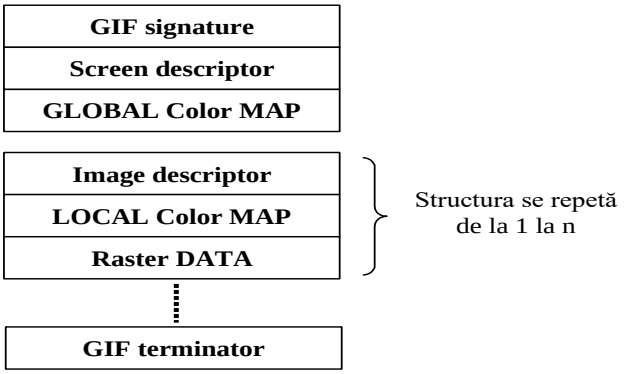


Figura 1.2 Formatul fişierului GIF

1.3.3 GIF Signature

Este folosit pentru a identifica tipul fişierului – fişier **GIF**. Conţine 6 caractere:

- **GIF87a**
- **GIF89a**

1.3.4 Screen descriptor

Descrie toţi parametrii ce vor fi folosiţi în continuare. Aici este definită dimensiunea spaţiului de imagine sau a ecranului logic necesar afişării, informaţia despre paleta de culori, culoarea fundalului (background) ecranului de afişare. Structura **screen descriptor** este prezentată în Figura 1.3.

Dimensiunea ecranului logic poate fi mai mare decât dimensiunea fizică a monitorului. Afişarea imaginilor mai mari decât dimensiunea ecranului se face folosind, de exemplu, barele de defilare oferite de sistemul de operare Windows.

Valoarea biţilor **pixel** – reprezintă numărul de culori dintr-o imagine. Domeniul de variaţie este de 0-7 adică 1 până la 8 biţi. Deci numărul de culori poate varia de la 2 (alb/negru) până la 256 culori. Bitul 3 al octetului 5 este rezervat dezvoltărilor viitoare şi este setat implicit la valoarea 0.

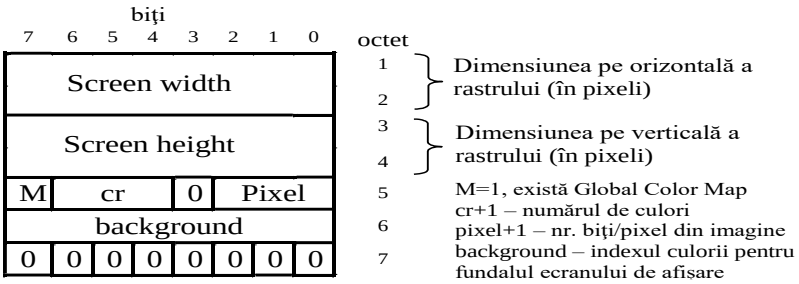


Figura 1.3 Structura screen descriptor-ului

1.3.5 Global Color Map (GCM)

Structura GCM – este opţională în structura fişierului GIF, dar se recomandă în cazul imaginilor unde se doreşte o redare cât mai exactă a culorilor. Existenţa GCM este indicată de bitul 7 din octetul 5 a structurii screen descriptor.

GCM este format din $2^{(nr.biţi/pixel)}$ blocuri de bază **CMB (Color Map Block)**. Structura unui bloc CMB este prezentată în Figura 1.4.

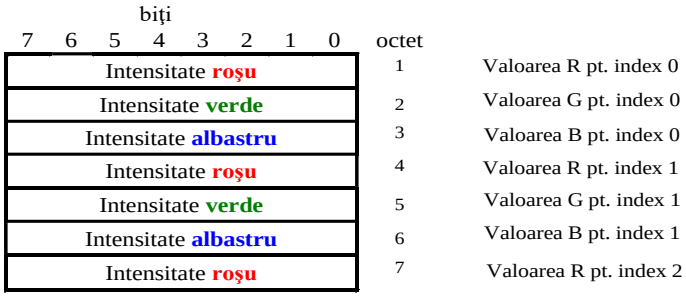


Figura 1.4 Structura unui CMB

Valoarea fiecărui pixel de imagine recepţionată va fi afişat folosind culoarea corespunzătoare din paleta de culori. Fiecare componentă de culoare R,G,B va avea valori cuprinse între 0÷255. Culoarea albă va corespunde reprezentării 255, 255, 255, iar negrul va corespunde reprezentării 0, 0, 0.

Dacă dispozitivul de afişare nu suportă rezoluţia imaginii, pentru afişare se vor folosi cei mai semnificativi biţi din fiecare componentă R, G, B.

1.3.6 Image Descriptor

Defineşte poziţia actuală şi extensia pentru următoarea imagine din cadrul screen descriptor. De asemenea este definit flag-ul ce indică existenţa paletei locale de culori (Local Color Map). Fiecare descriptor de imagine este introdus printr-un **caracter separator de imagine**. Rolul separatorului este de a stabili care sunt secvenţele de imagine, în cazul în care o structură GIF conţine mai mult decât o imagine. Separatorul este "**,**" (sau **2C**h). Când se întâlneşte acest caracter înseamnă că urmează imediat o structură **Image Descriptor**. Orice caractere care apar între indicatorul de sfârşit al imaginii precedente şi caracterul separator al imaginii curente sunt ignorate. Această modalitate de citire permite dezvoltarea formatului GIF pe viitor cu alte informaţii ce vor putea fi introduse în aceste poziţii. Structura unui **image descriptor** este prezentată în Figura 1.5.

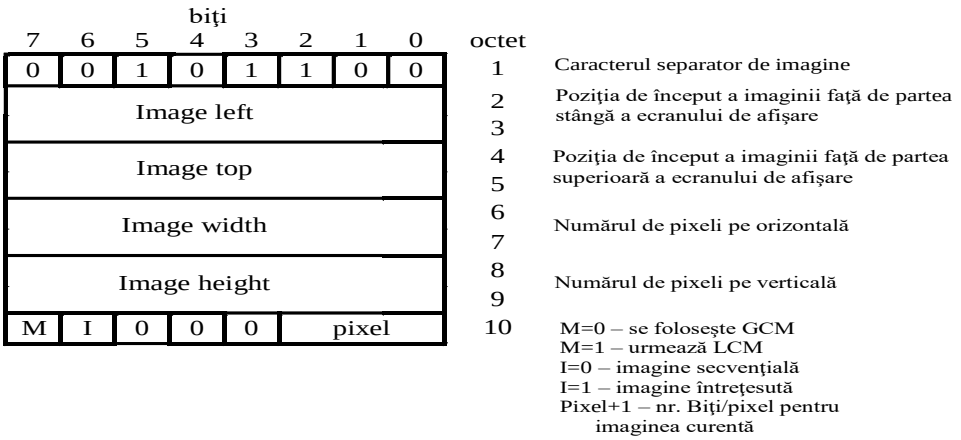


Figura 1.5 Structura unui image descriptor

1.3.7 Local Color Map (LCM)

Este o structură opţională. Dacă M=1 din structura **image descriptor**, avem LCM. LCM este folosită pentru afişarea imaginii curente. La sfârşitul structurii imaginii curente, paleta de culori va reveni la cea descrisă după **screen descriptor**. De notat faptul că valoarea câmpului **pixel** din image descriptor este folosit numai dacă se foloseşte LCM. Acest câmp defineşte rezoluţia culorii (biţi/pixel), determinând în felul acesta numărul de culori care compun paleta LCM ce urmează. După procesarea imaginii curente numărul de biţi/pixel va reveni la valoarea specificată în screen descriptor.

1.3.8 Raster Data

Imaginea este definită prin *serii de valori de indecşi* în ***paleta de culori***. Pixelii sunt baleiaţi de la stânga la dreapta pe fiecare rând, începând cu primul rând din imagine (rândul superior) şi terminându-se cu ultimul rând din imagine (rândul inferior).

Dacă bitul **I** (interlaced) este setat , pentru afişarea rândurilor se parcurg 4 paşi:

- ***În primul pas***, se afişează fiecare al 8-lea rând din imagine începând cu primul rând (rândul superior).
- ***În pasul doi***, se afişează fiecare al 8-lea rând începând cu rândul al 4-lea (relativ la primul rând).
- ***În pasul trei***, se afişează fiecare al 4-lea rând începând cu rândul al 2-lea (relativ la primul rând).

- ***În pasul patru***, se completează imaginea începând cu al doilea rând (relativ la primul rând).

Rând de imagine	Pasul1	Pasul 2	Pasul 3	Pasul 4	*rezultat*
0	1a				*1a*
1				4a	*4a*
2			3a		*3a*
3				4b	*4b*
4		2a			*2a*
5				4c	*4c*
6			3b		*3b*
7				4d	*4d*
8	1b				*1b*
9				4e	*4e*
10			3c		*3c*
11				4f	*4f*
12		2b			*2b*
13				4g	*4g*

Figura 1.6 Citirea unei imagini GIF Interlaced

O descriere grafică a procesului de afişare a unei imagini Interlaced este prezentată în Figura 1.6. Valorile pixelilor sunt indecşi într-o paletă de culori predefinită. Valorile citite din paleta de culori sunt afişate apoi pe ecran. Şirul de pixeli (de dimensiune lxL) este apoi comprimat folosind un algoritm LZW (Lempel-Ziv Welch).

1.3.9 GIF Terminator

Pentru a identifica sfârşitul fişierului GIF, un decodor GIF va cunoaşte acest lucru dacă întâlneşte caracterul "**;**" (**3B**h).

1.4 Formatul PNG

Fişierul PNG (Portable Network Graphics) reprezintă un format portabil pentru imaginile comprimate fără pierderi. Motivul iniţial pentru dezvoltarea PNG a fost de a înlocui formatul GIF cu un nou format care să conţină o serie de facilităţi care nu există în formatul GIF, şi care să fie relativ asemănător cu acesta, astfel încât costurile de conversie să fie relativ minime. Formatul PNG include următoarele caracteristici preluate din formatul GIF:

- imagini cu paletă până la 256 culori
- afişare progresivă
- transparenţa
- informaţii suplimentare: comentarii textuale sau alte informaţii ce pot fi incluse în fişier
- independenţa de platformă şi sistemul hardware
- compresia 100% fără pierderi

Formatul PNG conţine o serie de caracteristici noi care nu sunt incluse în GIF, dintre care cele mai importante sunt:

- utilizarea imaginilor true color până la imagini cu 48 biţi/pixel
- imagini cu nivele de gri până la 16 biţi/pixel
- utilizarea unui canal complet *alfa* de transparenţă (măştile de transparenţă)
- informaţia de luminozitate şi contrast

- fiabilitate în detectarea fişierelor eronate
- prezentare iniţială rapidă pentru modul de afişare progresiv

Formatul PNG se pronunţă de fapt „*ping*”.

1.4.1 Reprezentarea datelor

Toţi întregii care necesită mai mult de un octet vor fi ordonaţi după cum urmează: cei mai semnificativi octeţi se vor transmite primii urmând apoi să transmitem cei mai puţini semnificativi octeţi. De exemplu, pentru 2 octeţi se transmite întâi octetul MSB apoi octetul LSB, în timp ce pentru 4 octeţi se transmite octetul B_3, B_2, B_1, şi B_0. Cel mai mare bit al unui octet (valoarea 128) este identificat ca bitul 7, în timp ce pentru cel mai mic bit (valoare 1) acesta se identifică cu bitul 0. Valorile sunt fără semn dacă nu se specifică altfel. În cazul în care se specifică ca este vorba de valori cu semn atunci acestea sunt reprezentate în complement faţă de doi.

Culorile pot fi reprezentate fie sub formă de nivele de gri fie sub formă RGB. Datele sub formă de nivele de gri reprezintă de fapt luminanţa. Valorile culorilor variază de la zero (negru) până la $2^{\text{adâncimea eşantionului}}-1$.

Imaginea PNG poate fi privită ca un tablou dreptunghiular de pixeli, care apar de la stânga la dreapta şi de sus în jos (mai puţin în cazul afişării progresive – întreţesute când este vorba de o cu totul altă ordine de afişare). Dimensiunea fiecărui pixel este determinată de *numărul de biţi per pixel* cu care este reprezentată imaginea.

Există trei ***tipuri de pixeli***:

- pixel *indexat* – atunci când folosim paleta de culori
- pixel *cu nivel de gri* – atunci când pentru zero avem negru şi pentru alb avem corespunzătoare cea mai mare valoare a eşantionului
- pixel *true color* – în care fiecare pixel este reprezentat pe baza a trei eşantioane: roşu , verde şi albastru.

Canalul ***alfa*** reprezintă informaţia de transparenţă care poate fi inclusă în imaginile cu nivele de gri sau cele true color. Valoarea *alfa* egală cu zero reprezintă grad de transparenţă maxim, în timp ce $2^{\text{adâncime de bit}}-1$ reprezintă opacitate maximă pentru pixelul respectiv. Evident valorile intermediare reprezintă transparenţă parţială a pixelului curent.

Formatul PNG permite ca imaginea să fie filtrată înainte de a fi comprimată. Filtrarea ar putea îmbunătăţi gradul de compresie a imaginii. Toate filtrele PNG utilizate sunt fără pierderi. Cele mai importante filtre utilizate de formatul PNG sunt [1]: None – adică fără filtrare, Sub , Up, Average, Paeth

Am amintit faptul că PNG permite afişarea întreţesută. Scopul afişării întreţesute este de a permite utilizatorului să recepţioneze o parte din imagine, urmând ca restul să fie recepţionată după o anumită perioadă de timp. Metoda de întreţesere numită Adam7 după numele autorului acesteia Adam M. Costello, constă din parcurgerea succesivă a imaginii în 7 paşi. Fiecare pas transmite un subset de pixeli din imagine. Imaginea este împărţită în blocuri de 8x8 pixeli iar poziţia pixelilor transmişi în fiecare etapă se poate observa în Figura 1.7:

```
1 6 4 6 2 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
3 6 4 6 3 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
```

Figura 1.7 Poziţiile de întreţesere pentru cei 7 paşi

În fiecare etapă, pixelii selectaţi de acelaşi ordin sunt transmişi de la stânga la dreapta, şi respectiv de sus în jos. De exemplu, în pasul 2 sunt transmişi pixelii 4, 12, 20 etc. din liniile 0, 8, 16 etc. Formatul PNG poate specifica prin intermediul structurii **gAMA**, caracteristicile gamma ale imaginii astfel încât imaginea să fie afişată asemănător cu originalul.

Fişierul PNG poate stoca informaţii textuale asociate cu imaginea, cum ar fi o scurtă descriere a imaginii sau informaţii legate de copyright.

1.4.2 Structura fişierului PNG

Un fişier PNG conţine *semnătura PNG* şi o serie de structuri (chunk). În continuare vor fi prezentate structura semnăturii PNG şi a celor mai importante structuri ale fişierului.

1.4.3 Semnătura

Primii opt octeţi ai fişierului PNG reprezintă semnătura care conţine valorile zecimale:

137 80 78 71 13 10 26 10

Semnătura indică că ceea ce a mai rămas din fişier reprezintă o singură imagine PNG, care conţine o serie de structuri incluse între structurile de început **IHDR** şi de sfârşit **IEND**.

1.4.4 Formatul structurii (chunk)

Fiecare structură este formată din patru câmpuri:

- **lungime** – întreg fără semn pe 4 octeţi care specifică numărul de octeţi din câmpul data al structurii
- **tip structură** – codul tipului structurii pe 4 octeţi. Se folosesc numai litere ASCII (adică caractere A-Z, a-z sau valorile zecimale corespunzătoare 65-90, respectiv 97-122). Pentru mai multe detalii legate de tipul structurii şi convenţiile de nume se poate consulta standardul [1].
- **câmpul date** – octeţii de date corespunzători tipului de structură. Acest câmp poate avea lungime zero
- **CRC** – cod CRC (Cyclic Redundancy Check) calculat pentru câmpurile precedente *tip structură* şi *câmp date* (câmpul *lungime* nu intervine în calculul CRC)

1.4.5 Structuri critice

O imagine PNG validă trebuie să conţină următoarele structuri:

- **IHDR**
- unul sau mai multe structuri **IDAT**
- **IEND**

IHDR – apare ca primă structură şi conţine:	
Width:	4 bytes
Height:	4 bytes
Bit depth:	1 byte
Color type:	1 byte
Compression method:	1 byte
Filter method:	1 byte

Interlace method: 1 byte

- **Width** şi **Height**, ne dau dimensiunea imaginii în pixeli. Valorile sunt întregi pe 4 octeţi, deci valoarea maximă este 2^{31}-1.
- **Bit depth**, ne dă numărul de biţi per pixel. Valorile pot fi 1, 2, 4, 8 sau 16.
- **Compression method** – indică tipul de compresie folosită. În prezent (versiunea 1.0) există un singur algoritm de compresie care va fi folosit de toate tipurile de imagini.
- **Filter method** – indică tipul de prelucrare folosită înaintea compresiei. Prelucrarea utilizată în prezent este filtrarea adaptivă cu cele 5 categorii de filtre.
- **Interlaced method** – indică modalitatea de transmisie a datelor de imagine. Valorile posibile sunt:
  - 0 – neîntreţesută
  - 1 – întreţeserea Adam7 descrisă mai sus

- **Color type**, descrie modalitatea de interpretare a datelor de imagine. Valorile posibile sunt 0, 2, 3, 4, şi 6.

Tabelul 1.1 Combinaţiile posibile pentru parametrul Color type

Color type	Valorile permise pentru **Bit depth**	Interpretare
0	1, 2, 4, 8, 16	fiecare pixel reprezintă un nivel de gri
2	8, 16	fiecare pixel reprezintă un triplet R, G, B
3	1, 2, 4, 8	fiecare pixel reprezintă un index în paleta de culori. Trebuie să avem o structură PLTE
4	8, 16	fiecare pixel reprezintă un nivel de gri urmat de nivelul *alfa*
6	8, 16	fiecare pixel este un triplet R, G, B, urmat de nivelul *alfa*

PLTE – conţine paleta de culori 1-256, pentru fiecare intrare câte o triadă RGB

Red: 1 byte (0 = black, 255 = red)
Green: 1 byte (0 = black, 255 = green)
Blue: 1 byte (0 = black, 255 = blue)

IDAT – image data – conţine datele de imagine.

Acestea se obţin prin parcurgerea etapelor următoare:

- scanarea imaginii
- aplicarea filtrării
- compresia datelor de imagine

Dacă avem structuri de date IDAT multiple acestea vor fi ordonate una după alta fără a insera alte tipuri de structuri.

IEND – apare pe ultima poziţie a fişierului semnalând sfârşitul şirului PNG.

Câmpul structurii este gol.

1.4.6 Structuri auxiliare

Toate structurile auxiliare sunt opţionale astfel că arhitectura de codare şi decodare poate să le ignore. În continuare vor fi prezentate cele mai importante structuri auxiliare:

bKGD – specifică culoarea implicită a fondului imaginii

Pentru ***color type* = *3*** (indexed color), bKGD conţine:

Palette index: 1 byte

Valoarea este indexul din paletă a culorii corespunzătoare fundalului.

Pentru ***color type 0* sau *4*** (imagine cu nivele de gri cu sau fără parametrul *alfa*), structura bKGD conţine:

Gray: 2 bytes, *domeniul 0 .. (2^bitdepth)-1*

Valoarea reprezintă nivelul de gri utilizat.

Pentru *color types 2* şi *6* (truecolor, cu sau fără parametrul *alfa*), bKGD conţine:

Red: 2 bytes, *domeniul* 0 .. (2^bitdepth)-1
Green: 2 bytes, *domeniul* 0 .. (2^bitdepth)-1
Blue: 2 bytes, *domeniul* 0 .. (2^bitdepth)-1

Valoarea reprezintă conţinutul RGB a culorii fundalului.

Atunci când este prezent în structura fişierului, structura bKGD trebuie să preceadă prima structură IDAT, şi trebuie să fie urmată de structura PLTE dacă există.

cHRM – pentru specificarea parametrilor independenţi de dispozitiv

Structura cHRM conţine:

White Point x:	4 bytes
White Point y:	4 bytes
Red x:	4 bytes
Red y:	4 bytes
Green x:	4 bytes
Green y:	4 bytes
Blue x:	4 bytes
Blue y:	4 bytes

Fiecare valoare este codată ca întreg fără semn pe 4 octeţi şi reprezintă valorile *x* sau *y* înmulţite cu 100000. De exemplu, valoarea 0,5432 va fi codată 54320. Atunci când este prezent în structura fişierului trebuie să preceadă prima structură IDAT şi de asemenea să preceadă structura PLTE dacă există.

gAMA – specifică parametrul gamma al camerei (virtuală) care produce imaginea

Structura gAMA conţine:

Image gamma: 4 bytes

Valoare este codată ca întreg fără semn pe 4 octeţi şi reprezintă valoarea *gamma* înmulţită cu 100000. Atunci când este prezent în structura fişierului, gAMA trebuie să preceadă prima structură IDAT şi de asemenea să preceadă structura PLTE dacă există.

hIST – specifică frecvenţa de apariţie a fiecărei culori din paleta de culori

Structura hIST apare numai dacă există şi structura PLTE.

pHYs – specifică dimensiunea pixelului

Structura pHYs conţine:

Pixels per unit, X axis:	4 bytes (unsigned integer)
Pixels per unit, Y axis:	4 bytes (unsigned integer)
Unit specifier:	1 byte

Valoarea câmpului *unit specifier* poate fi:

- 0; unitatea nu este cunoscută
- 1; unitatea este *metrul*

Dacă este prezentă, structura pHYs trebuie să preceadă prima structură IDAT.

tEXt – permite inserarea de informaţii textuale

Structura tEXt conţine:

Keyword:	1-79 bytes (şir de caractere)
Null separator:	1 byte
Text:	n bytes (şir de caractere)

Câmpul *keyword* şi *text* sunt separate printr-un caracter separator *null separator* (caracterul NULL). Câmpul *text* nu are terminator de şir, lungimea structurii ne va spune cât de mare este acest şir de caractere. Există o serie de cuvinte cheie *(keyword)* predefinite:

Title:	Short (one line) title or caption for image
Author:	Name of image's creator
Description:	Description of image (possibly long)
Copyright:	Copyright notice
Creation Time:	Time of original image creation
Software:	Software used to create the image
Disclaimer:	Legal disclaimer
Warning:	Warning of nature of content
Source:	Device used to create the image
Comment:	Miscellaneous comment; conversion from GIF comment

tIME – specifică data la care s-a efectuat ultima modificare

Structura tIME conţine câmpurile:

Year:	2 bytes (complet; de exemplu, 1995, şi nu 95)
Month:	1 byte (1-12)
Day:	1 byte (1-31)
Hour:	1 byte (0-23)
Minute:	1 byte (0-59)
Second:	1 byte (0-60)

tRNS – specifică că imaginea foloseşte efectul de transparenţă

Structura poate specifica dacă imaginea foloseşte efectul de transparenţă prin parametrul *alfa* sau foloseşte o singură culoare ca fiind transparentă. Atunci când este prezentă, structura tRNS trebuie să preceadă prima structură IDAT.

zTXt – conţine informaţii textuale comprimate

Structura zTXt conţine câmpurile:

Keyword:	1-79 bytes (character string)
Null separator:	1 byte
Compression method:	1 byte
Compressed text:	n bytes

Descrierea câmpurilor este asemănătoare cu a structurii tEXt.

1.5 Desfăşurarea lucrării

1.5.1 Activitate

a) Folosind programul ***PaintShopPro*** se va construi un fişier BMP corespunzător unei imagini de dimensiune 20x20 pixeli, în care se află un pătrat centrat de culoare roşie pe fond albastru de dimensiune 5x5. Fişierul bitmap se va numi ***mybitmap_256.bmp***, pentru fişier în format 8 biţi/pixel. Se repetă operaţiunea de creare a imaginilor pentru fişiere cu 24 biţi/pixel. Folosind submeniul ***view*** şi submeniul ***Image information*** se citesc parametri imaginilor realizate şi se compară aceşti parametrii pentru fiecare dintre cele două fişiere.

b) Folosind ***PaintShopPro*** salvaţi imaginea de tip BMP în fişier GIF, PNG şi faceţi comparaţii privitor la dimensiunea fişierelor (atât pentru imaginile cu 8 biţi/pixel cât şi pentru cele cu 24 biţi/pixel).

c) Încărcaţi imaginea **lena.bmp**. şi refaceţi punctele *a)* şi *b)*. Ce se poate observa legat de dimensiunea fişierelor?

d) Folosind utilitarul ***hackman***, deschideţi fişierele **lena.bmp,** şi **lena.png**. Baleiaţi fişierele şi vedeţi care sunt parametrii utilizaţi. Extrageţi din fişierul **lena.png** toate numele de structuri folosite de formatul ales.

e) Cu ajutorul programului ***PaintShopPro***, deschideţi fişierul creat la punctul a) (mybitmap_256.bmp). Deschideţi paleta de culori (Edit Pallete) aferentă fişierului şi definiţi culorile de la index 13 şi respectiv 200 după cum urmează:

- index 13; R=255; G=255; B=128;
- index 200; R=200; G=20; B=255;

După ce aţi modificat paleta de culori salvaţi fişierul mybitmap_256.bmp. Deschideţi apoi fişierul cu ajutorul programului ***hackman*** şi editaţi-l astfel încât, să avem ca fundal culoarea de la indexul 13 din paleta de culori şi respectiv pătratul să aibă culoarea de la indexul 200 din paleta de culori aferentă. Deschideţi fişierul cu PaintShopPro. Ce se poate observa?

f) Cu ajutorul programului ***hackman***, deschideţi fişierul creat la punctul a) (mybitmap_256.bmp). Editaţi paleta de culori aferentă fişierului pentru culorile de la index 56 (din paleta de culori) şi respectiv 210 după cum urmează:

- index 56; R=200; G=200; B=100;
- index 210; R=255; G=0; B=255;

Editaţi în continuare fişierul astfel încât să avem ca fundal culoarea 210 şi pătratul să aibă culoarea 56 din paleta de culori aferentă. Deschideţi fişierul cu PaintShopPro. Ce se poate observa?

g) Cu ajutorul programului ***hackman***, deschideţi fişierul creat la punctul a) (mybitmap_256.bmp). Editaţi imaginea astfel încât să măriţi dimensiunea pătratului de la 5x5 pixeli la 7x7 pixeli.

1.6 Întrebări şi probleme

1. Care sunt dezavantajele majore ale formatului BMP?
2. Ce mărime va avea un fişier de imagine BMP 8 biţi/pixel pe disc, dacă se definesc 200 culori, şi are dimensiunea 320 pixeli/linie şi 200 linii? Completaţi cele trei structuri ale formatului BMP, astfel încât fişierul obţinut să fie valid.
3. Care este numărul maxim de nuanţe de culoare pe care le putem realiza cu trei regiştrii de culoare de câte un octet fiecare?
4. Ce se întâmplă dacă vreţi să salvaţi o imagine BMP true color în format GIF? Explicaţi. Dar în cazul în care se doreşte conversia din BMP true color în format PNG?
5. Creaţi un fişier PNG cu ajutorul unui program de procesare a imaginilor. Introduceţi cu ajutorul editorului ***hackman***, structuri tEXt, pentru cuvintele cheie Title, Author, Description, Copyright, Comment. Textul aferent cuvintelor cheie nu este condiţionat.
6. Enumeraţi avantajele formatului PNG faţă de formatul GIF.

1.7 Bibliografie suplimentară

[1]. BMP format http://www.fortunecity.com/ skyscraper/windows/ 364/ bmpffrmt.html

[2]. GIF format http://astronomy.swin.edu.au/~pbourke/dataformats/gif/

[3]. PNG Specification Version 1.0; ftp://ftp.uu.net/graphics/png/