Programe de aplicaţie

UN ALGORITM PENTRU OPERAŢII DE PANNING, ZOOM ŞI SCALARE PE O IMAGINE DE TIP RASTER

ing. Dragoş Dobran
ing. Marius Niţu,

Institutul de Cercetări în Informatică

Rezumat Acest articol prezintă un algoritm pentru realizarea funcţiilor de zoom, pan şi scalare pe o imagine raster stocată într-un fişier de tip TIFF. După o foarte scurtă prezentare a structurii fişierelor TIFF, se prezintă ideea care stă la baza efectuării operaţiilor de scalare. Faţă de alţi algoritmi cunoscuţi, prezentul algoritm oferă un control suplimentar al modului de afişare a imaginii prin variaţia unui parametru ce măsoară gradul de umplere al unei regiuni.

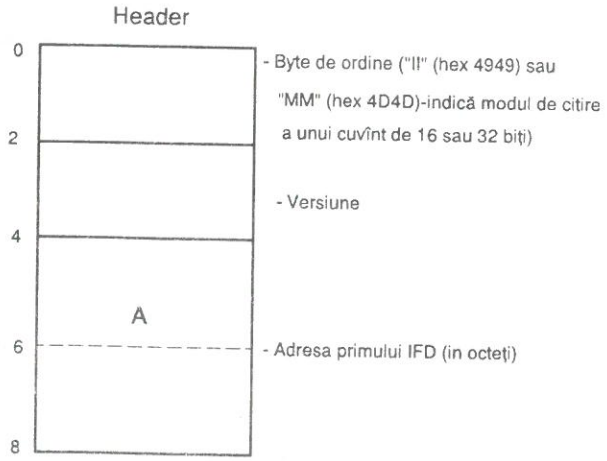
Cuvinte cheie: grafică, TIFF (Tag Image File Format), imagine raster, operaţii de vizualizare

1. Introducere

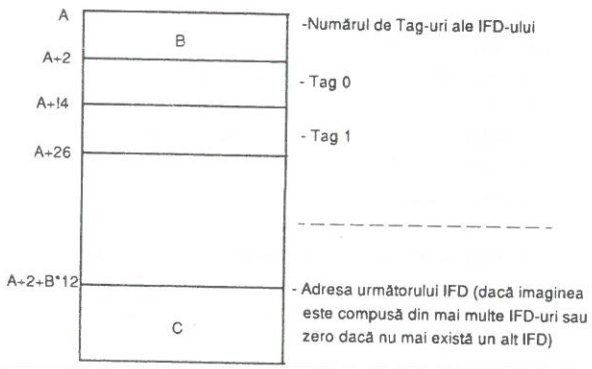
Acest articol reprezintă o posibilă soluţie de prelucrare a unor imagini de tip raster, rezultate în urma "scanării" unui desen. Ţinînd cont de faptul că imaginea raster, în cazul nostru un fişier TIFF, este independentă de sistemul de operare şi de tipul "maşinii de lucru", algoritmul a fost prezentat într-o formă secvenţială, uşor de înţeles, iar în final s-a oferit o posibilă implementare într-un limbaj de nivel înalt (C).

2. Prezentare

Algoritmul prezentat în continuare pleacă de la structura unui fişier de tip TIFF (Tag Image File Format), în care sînt înmagazinate toate informaţiile asupra unei imagini raster. Această structură este prezentată în schema următoare:

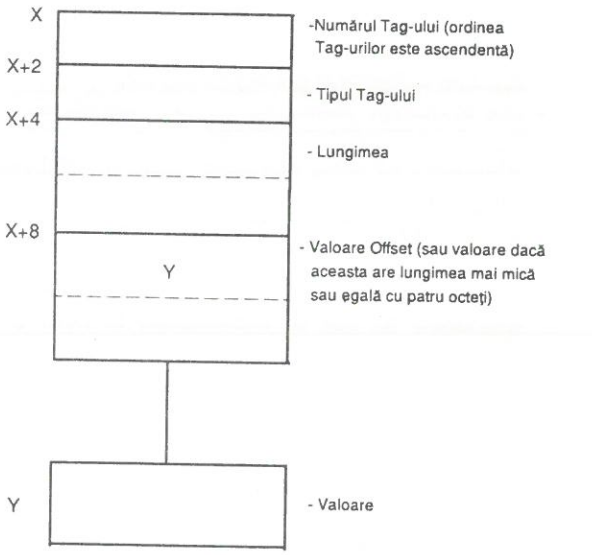


IFD (Image File Directory)



Cei 12 octeţi din care este format un Tag (Directory Entry) sînt structuraţi astfel:

Directory Entry (Tag)



Se poate determina, în funcţie de Tipul şi Lungimea Tag-ului, dacă "Valoare" este pe 4 octeţi sau mai mare (caz în care Valoare Offset conţine adresa unde se găseşte valoarea).

Lungimea este specificată în termeni de tipuri de Tag şi nu specifică numărul de octeţi a valorii (de exemplu, un cuvînt de 16 biţi (SHORT) are lungimea 1 şi nu 2).

Tipurile Tag–ului pot fi următoarele:

1 - BYTE (un octet fără semn);
2 - ASCII (secvenţa de octeţi care conţin coduri ASCII; ultimul octet trebuie să fie nul);
3 - SHORT (un cuvînt de 2 octeţi fără semn);
4 - LONG (un cuvînt de 4 octeţi fără semn);
5 -RATIONAL (2 LONG–uri primul reprezintă

numărătorul, şi al doilea numitorul, de exemplu la rezoluţia pe X şi Y).

În general, informaţiile conţinute de Tag-urile unui fişier TIFF se referă la:

- dimensiunile imaginii (date în pixeli);
- modul de compresie a imaginii (informaţia despre imaginea raster poate fi compresată cu una din cele 4 metode de compresare: arbori HOUFFMAN, LWZ,schema 30773 sau necompresată);
- adresa unde se găseşte informaţia despre imagine;
- modul de organizare a imaginii în fîşii (este posibil ca pentru uşurarea creării unui fişier TIFF cît şi pentru uşurarea decodificării acestuia, imaginea să fie compusă din mai multe fîşii; fiecare fîşie începe la un octet nou dar în cadrul unei fîşii o linie a imaginii nu începe în mod obligatoriu la un octet nou);
- numărul de biţi alocaţi pentru a descrie un pixel (în funcţie de tipul imaginii : alb-negru, grayscale, color etc.);
- paleta de culori dacă este cazul;
- rezoluţia pe X, Y sau unitară (dată în inch, centimetri sau fără unităţi absolute de măsură);
- la anumite fişiere TIFF pe lîngă imaginea normală mai este adăugată şi o imagine scalată;
- alte informaţii referitoare la tipul scanner-ului, la data şi autorul creării imaginii.

Algoritmul de zoom şi paning pentru imagini raster este implementat pentru imagini alb-negru în care un pixel este stocat în fişierul TIFF pe un bit. El primeşte ca parametri:

- handel–ul fişierului TIFF (la fiecare scalare sau operaţiune de pan se poziţionează în fişier şi se citesc datele corespunzătoare);
- dimensiunile ferestrei de desenare ("xmin", "ymin", "xmax", "ymax");
- structura în care s–au memorat informaţiile conţinute de Tag–urile fişierului TIFF ("ptrtf");
- poziţia pe X şi Y în pixeli de unde se afişează imaginea ("startx", "starty");
- gradul mediu de umplere a imaginii vizualizate ("fill");
- factorul de scale global ("zr").

Pentru realizarea scalării imaginii raster, dacă parametrul "zr" este zero, se calculează factorul de scalare pe X şi Y astfel încît toată imaginea să încapă în fereastra de desenare şi se alege drept factor de scalare global, cea mai mare valoare dintre acestea. Se foloseşte un factor de scalare unic deoarece imaginea se împarte în pătrate de dimensiune zr, fiecare pătrat reprezentînd un pixel în imaginea scalată. S-a luat decizia ca factorul de scalare să fie acelaşi şi pe X şi Y, deoarece, în caz contrar, la diferenţe prea mari între factorii de scalare, pe cele două dimensiuni este posibilă o distorsionare a imaginii.

Decizia a se aprinde sau a se stinge pixelul corespunzător unui pătrat se ia în funcţie de rezultatul comparării dintre gradul real de umplere a pătratului (preponderenţa pixelilor stinşi faţa de cei aprinşi) şi gradul mediu de umplere a imaginii (parametrul "fill" hotărît de utilizator). În funcţie de parametrul "fill", imaginea apare mai întunecoasă sau mai luminoasă astfel: dacă "fill" este mai mare de 0,5, imaginea este mai întunecată (în acest caz textul imaginii este mai bine vizibil), iar dacă "fill" este mai mic de 0,5 imaginea este mai luminoasă (sînt vizibile mai bine contururile). S-a ales această soluţie (cu gradul de umplere mediu al imaginii) în locul soluţiei alegerii unui pixel reprezentativ pentru pătratul de scalare (stînga sus, centru) deoarece gradul de incertitudine şi de distorsionare a imaginii este mult mai mic pentru soluţia aleasă. Aceasta, chiar dacă viteza de lucru este sensibil micşorată.

Modul de lucru al algoritmului este următorul:

- se analizează o fîşie de pătrate de scalare, determinînd pentru fiecare pătrat gradul real de umplere al acestuia şi afişînd cîte o linie din imaginea scalată;
- pentru a realiza şi operaţiunea de pan pe Y la începutul algoritmului în funcţie de "starty" se poziţionează în fişierul TIFF la adresa potrivită;
- pentru fiecare linie de imagine se poziţionează la "startx" (dat în pixeli) şi se citesc numărul corespunzător de octeţi. Este posibil ca poziţionarea la "startx" să se realizeze în cadrul unui octet nu neapărat la început se va lua în calcul şi acest octet aplicîndu–i–se o mască corespunzătoare conform pasului următor;
- pentru fiecare octet în parte, se aplică masca corespunzătoare, şi rezultatul aplicării măştii se adaugă la celulele corespunzătoare dintr-un vector (iniţializat cu zero); se repetă acest pas pentru fiecare linie din fîşia de pătrate, la sfîrşitul operaţiunii fiecare celulă conţine gradul real de umplere a unui pătrat de scalare. Dimensiunea vectorului este dată de lăţimea ferestrei de desenare dacă imaginea scalată este mai mare lăţimea imaginii scalate sau lăţimea imaginii scalate în caz contrar. Prin parcurgerea vectorului şi în urma comparării valorii celulelor sale cu gradul mediu de umplere a imaginii ("fill"), se determină modul de afişare a liniei;
- se repetă paşii anteriori pentru fiecare fişie de pătrate de scalare.

Trebuie să se aibă în vedere că s-ar putea ca dimensiunile pătratului de scalare să fie mai mari decît un octet sau ca aceste dimensiuni să nu fie un multiplu sau divizor perfect a lui opt (dimensiunea unui octet-modul de citire a datelor din fişier este octet cu octet). Acest lucru s–a rezolvat deoarece masca aplicată este, pentru fiecare bit în parte din octet, şi nu pentru întreg

octetul, iar adăugarea rezultatului aplicării măştii la celula corespunzătoare din vector se face cît timp ne aflăm în cadrul unui pătrat de scalare.

```
//     tiff header file
struct tiff_h{          /* header structure */
  int byteo,vers;
  unsigned long int IFD1;
};
struct dir_entry{    /* directory entry structure */
  int tag,type;
  unsigned long int length,value;
};
struct ifd{/* IFD structure */
  int entry_count;
  struct dir_entry *ptrdire_l[45]; /* Maximum number
of
                                    directory entrys */
  long ifd_offset;
};
struct tag_fields{
  unsigned int BitsPerSample;
  unsigned int ColorMap;
  unsigned int ColorResponseCurves;
  unsigned int Compression;
  unsigned int GrayResponseCurve;
  unsigned int GrayResponseUnit;
  unsigned long int ImageLength;
  unsigned long int ImageWidth;
  unsigned long int NewSubfileType;
  unsigned int PhotometricInterpretation;
  unsigned int PlanarConfiguration;
  unsigned int Predictor;
  unsigned int ResolutionUnit;
  unsigned long int RowsPerStrip;
  unsigned int SamplesPerPixel;
  unsigned long int StripByteCounts;
  unsigned long int StripOffset;
  unsigned long int XResolution;
  unsigned long int YResolution;
  unsigned long int Group3Options;
  unsigned long int Group4Options;
  unsigned int SubfileType;
};

typedef struct tiff_h *PTRHEADER;
typedef struct dir_entry *PTRDIRENTRY;
typedef struct ifd *PTRIFD;
typedef struct tag_fields *PTRTAGFIELDS;

void putlinie(xmin,xmax, i,b[],n,gru)
{
  int x;
  for (x=0;x<n;x++)
    if (x<(xmax-xmin))
      if (b[x]>gru)
          putpixel((xmin+x),i,WHITE);
      else
          putpixel((xmin+x),i,BLACK);
}
unsigned int comp_tiff_image(hnd,xmin,ymin,xmax,ymax,
ptrtf,zr,startx,starty,fill)
{
    unsigned int dx,dy,j,k,i,l,m,n,p,ii,dpy;
    unsigned char buf1,buf;
    unsigned char a[128],b[640];
    unsigned long pozfile;
    float gru;

  if ((starty<ptrtf->ImageLength) &&
     (startx<ptrtf->ImageWidth)) {
   if (ptrtf->ImageLength>(ymax-ymin))
    dy=(unsigned
int)ptrtf->ImageLength/(ymax-ymin)+1;
    else
     dy=1;
    if (ptrtf->ImageWidth>(xmax-xmin))
    dx=(unsigned int)ptrtf -> ImageWidth/(xmax-xmin)+1;
    else
     dx=1;
   if (zr>=dy)
     starty=0;
    if (zr>=dx)
     startx=0;
    dy=max(dx,dy);
    if (zr)    /* zr <> 0 zoom cu valoarea impus\ */
     dy=zr;
    if (fabs(fill)>1.0)
     fill=0.9;
    gru=dy*dy*fill;
    dpy=min((unsigned  int)(ptrtf->ImageLength/dy  - -
starty/dy),(ymax-ymin));
    for(p=0;p<dpy;p++) {
          for (ii=0;ii<640;ii++)
                b[ii]=0;
        for(m=0;m<dy;m++) {
   pozfile=ptrtf->StripOffset + (ptrtf ->
                   ImageWidth/8)*(starty+p*dy+m);
                lseek(hnd,pozfile,SEEK_SET);
                j=0;
                l=0;

   for(n=0;n<ptrtf->ImageWidth/128+1;n++) {
                        if ((read(hnd, &a, 128)) == -1)
                         printerror(hnd,1);
                        if (startx/8<=128*(n+1))
                         for(i=0;i<128;i++) {
                               buf=a[i];

   if(i+n*128>=startx/8)
```

```
                    for(k=0;k<8;k++) {
                            buf1=buf;
                            buf1=buf1<<k;
                            buf1=buf1>>7;
                            b[j]=b[j]+buf1;
                                    l++;
                                    if (l==dy) {
                                    l=0;
                                    j++;
                                    }
                            }
                            if (j>((ptrtf->ImageWidth -
startx-startx % 8)/dy) || (j>xmax))
                                    break;
                            }
                            if
(j>((ptrtf->ImageWidth-startx - startx % 8)/dy) || (j>xmax))
                                    break;
                    }
            }
            if (p<(ymax-ymin))

      putlinie(xmin,xmax,ymin+p,b,j-1,gru);
      }
   }
   return (dy);
}
```

De observat că, acest algoritm face parte din cadr unui program de vectorizare manuală, prin trecere fişierelor TIFF rezultate în urma scanării unui desen î formatul DXF, format ce poate fi importat de produs AutoCad sau de alte produse din clasa editoarelc grafice, imaginea rezultată putînd fi prelucrată mult ma uşor cu aceste produse.

3. Concluzii

În acest articol s-a prezentat un algoritm pentr realizarea operaţiunilor de vizualizare a unor imagini d tip raster : scalare, pan, zoom. Algoritmul este descri în paşi, iar la sfîrşit se prezintă şi implementarea lui î limbajul C. De observat compromisul făcut între vitez de prelucrare şi controlul suplimentar al afişări imaginii, control realizat prin intermediul parametrulu "fill". Prezentul algoritm, aflat la prima versiune, poat fi îmbunătăţit, în special, în domeniul vitezei d execuţie.

Bibliografie

***: Tag Image File Format-Specification Revision 5.0 Aldus Corporation, 1989