Univerzitet u Beogradu, Elektrotehnički fakultet

Izveštaj projekta iz Računarske elektronike

Autori: Milica Barišić 2012/0304 i

Tatjana Toroman 2012/0282

Jun, 2017

Sadržaj

Projektni zadatak br. 6: Binarizacija slike iterativnim određivanjem praga binarizacije

Zadat je bio napisati program koji učitava sliku u PGMA formatu, zatim primenom iterativnog metoda određuje prag binarizacije T i zatim vrši binarizaciju slike tako što se svi pikseli sa vrednošću većom od T postavljaju na 255, a svi manji od T na 0.

Iterativno određivanje praga zasniva se na formiranju histograma slike (više informacija o histogramu može se pročitati u tekstu projekta 3) i definisanju početnog praga binarizacije T. Zatim se računaju srednje vrednosti osvetljaja piksela intenziteta manjeg od T (T1) i piksela intenziteta većeg od T (T2). Novi prag binarizacije dobija se kao srednja vrednost pragova T1 i T2. Ovaj postupak ponavlja se dok god je razlika između pragova dobijenih u dva susedna koraka veća od neke predefnisane konstante. Rezultat izvršavanja programa je slika u PGMA formatu binarizovana primenom dobijenog praga.

Procedura Čitaj

Procedura čitaj služi kako bi olakšala čitanje podataka iz fajla. Čitaju se karakteri dok se ne stigne do razmaka ili novog reda. Procedura čitaj poziva ParseDecimal32. Čitanje se poziva kasnij eu program od strane drugih procedura.

Procedura Nađi prag

Procedura Nađi prag za postojeću vrednost praga binarizacije T računa novi prag binarizacije iterativnom metodom.

To se postiže tako što se kreće kroz histogram i za sve vrednosti manje od praga T računa prosečnu vrednost intenziteta I smešta u pomoćnu promenljivu T1. Isto tako računa I prosečnu vrednost osvetljaja za piksele veće od T. Nakon toga računa aritmetičku sredinu T1 i T2 I to čuva u promenjljivoj T3. Ova vrednost predstavlja novi prag T.

Ova procedura se u Obradi slike poziva onoliko puta koliko je potrebno da razlika dve uzastopne vrednosti bude manja od unapred zadate konstante PRAG.

Procedura Učitavanje

Procedura Učitavanje učitava ime ulaznog fajla pozivajući procedure Čitaj. Procedura koristi bafer u kojem cuva vrednosti karaktera.

Procedura Formiranje histograma

Procedura Formiranje histograma paralelno učitava vrednosti piksela I skladišti ih u niz pixeli, I formira niz histogram u kojem se prebrojavaju ponavljanja piksela. Time je formiran histogram slike.

Procedura Obrada slike

Procedura Obrada slike poziva procedure Nađi prag sve dok je razlika dve uzastopne vrednosti praga veća od konstante PRAG. Ova procedura se poziva u glavnom programu.

Procedura Izlazni fajl

Procedura izlazni fajl služi da na osnovu praga T koji je izračunat iterativnim metodom u Obradi slike formira izlazni fajl. Svi pikseli koji imaju vrednost manju od praga T dobijaju vrednost 0, što predstavlja crnu boju, dok pikseli koji imaju veću vrednost od T dobijaju vrednost 255, što predstavlja belu boju. Ovim je postignuta binarizacija slike.

Naravno, podrazumeva se da se izlazni fajl u istom format kao I ulazni.

Glavni program

Glavni program poziva sledeće procedure: Ucitavanje, Formiranje histograma, Obrada slike i Izlazni fajl. Pre poziva ovih procedura obražuju se moguće greške do kojih može doći prilikom nepravilnog korišćenja programa.

Kod našeg programa

```
INCLUDE Irvine32.inc
INCLUDE macros.inc

.const
BUFFER_SIZE = 200000
MAX_PICTURE_SIZE = 65536 ;// 256*256
INITIAL_T = 256; ;// Pocetna vrednost praga binarizacije
PRAG = 1; ;// Prag izmedju dve susedne vrednosti T

.data
buffer BYTE BUFFER_SIZE DUP(?)
filename    BYTE 80 DUP(0)
fileHandle  HANDLE ?

outbuffer BYTE BUFFER_SIZE DUP(?)
outfilename BYTE 80 DUP(0)
outfileHandle HANDLE ?
outindex DWORD 0

;// Parametri slike:
P2 WORD 5032h
velicinaBafera DWORD ?
M WORD ? ;//Dimenzije slike su: M i N
N WORD ?
Lmax WORD ? ;//Maksimalna vrednost osvetljaja

;// Promenljive
pokazivacNiza DWORD 0;// Promenljivu koristimo kao index.
suma DWORD 0;// Promenljivu koristimo u proceduri ObradaSlike.
suma1 DWORD 0;// Promenljivu koristimo u proceduri ObradaSlike.
histogram WORD 256 DUP(0);// Vrednost clana niza zavisi od broja ponavljanja u nizu pixeli.
pixeli WORD MAX_PICTURE_SIZE DUP(?);// Niz koji koristimo za smestanje vrednosti iz buffera.
Kada naidjemo na EOL,
                ;// u niz upisujemo vrednost -1, zbog toga je niz tipa WORD.
               ;// U suprotnom, ako bismo imali BYTE, vrednost -1 bila bi ista kao vrednost 25
T WORD INITIAL_T;// Prag binarizacije
T1 WORD 0;// Pomocna promenljiva
T2 WORD 0;// Pomocna promenljiva
T3 WORD 0;// Pomocna promenljiva


broj BYTE 4 DUP(?)
```

```
cifra WORD 0;// Odredjuje da li je broj jednocifrena, dvocifrena ili trocifrena.
brojac DWORD 0

;// Ovde pocinje kod.

.code

close_file PROC
mov eax,fileHandle
call CloseFile
exit
close_file ENDP

;// PROCEDURA: Citaj
;// Pretvara ucitanu vrednost ciji je pocetni clan buffer[pokazivacNiza] u INT.
;// Po izlasku iz procedure vrednost je sacuvana u EAX, dok buffer[pokazivacNiza] pokazuje na
20h(space).

Citaj PROC STDCALL USES ebx esi ecx

mov edx,OFFSET buffer;
xor eax,eax
xor ecx,ecx
xor ebx,ebx

add edx,pokazivacNiza

;// Petlja ce se obradjivati sve dok ne naidjemo na razmak ili novi red.

Ucitavanje:
mov al,[edx]
mov broj[ebx],al
inc edx
inc pokazivacNiza
inc ebx
mov al,[edx]
cmp al,20h
je Pretvaranje
cmp al,0ah
jne Ucitavanje

Pretvaranje:
mov broj[ebx],3;// Kraj stringa u ASCII je 3h.

;// ParseDecimal trazi da EDX i ECX budu popunjeni ovako.
mov edx,OFFSET broj
mov ecx,ebx
call ParseDecimal32

;// Pre povratka iz rutine vracamo offset buffer-a u EDX.
mov edx,OFFSET buffer
ret
Citaj ENDP

;// PROCEDURA: NadjiPrag
;// Pretvara vrednost ciji je pocetni clan buffer[pokazivacNiza] u INT.
;// Po izlasku iz procedure vrednost je sacuvana u EAX, dok buffer[pokazivacNiza] pokazuje na
20h(space).
```

```
NadjiPrag PROC STDCALL USES ebx edi ecx eax

xor edi,edi
xor eax,eax
xor ebx,ebx
xor edx,edx

mov T1,0
mov T2,0
mov suma,0
mov suma1,0

Manji: ;// Racunamo srednju vrednost osvetljaja za one piksele koji su ispod praga T.

mov eax, edi
cmp ax, T
jge Veci1
mov ax, histogram[edi]
add suma1, eax
mov ax, histogram[edi]
mov ebx, edi
imul bx
add suma, eax
add edi, 2

jmp Manji

Veci1: ;// Racunamo srednju vrednost osvetljaja za one piksele koji su ispod praga T.

mov eax, suma
mov ebx, suma1
cmp bx, 0
je SveNule
idiv bx
xor edx,edx
mov bx,2
idiv bx
mov T1, ax
jmp Petlja

SveNule: ;// U slucaju da su svi pikseli jednaki 0 ne sme da se deli sa 0.
xor edx,edx
mov ax, T
mov bx, 2
idiv bx
mov T1, ax

Petlja:
mov suma1,0
mov suma, 0

Veci:

mov ax, histogram[edi]
add suma1, eax
mov ax, histogram[edi]
mov ebx, edi
imul bx
add suma, eax
```

```
add edi, 2



cmp edi,512
jl Veci

Dalje: ;// Nastavljamo.
xor edx,edx
mov eax, suma
mov edx, suma
shr edx,16
mov ebx, suma1
div ebx
mov bx,2
xor edx,edx
div bx
mov T2, ax


xor edx,edx
add ax, T1
mov bx, 2
div bx
mov T3, ax

ret
NadjiPrag ENDP

;// PROCEDURA: Ucitavanje
;// Otvaramo sliku i formiramo izlaznu sliku uz provere ispravnosti.
;// Ucitavamo parametre M,N,Lmax i komentar.
;// Prepisujemo prva cetiri reda u outbuffer.
;// Po izlasku iz procedure vrednost buffer[pokazivacNiza] pokazuje na prvi pixel,
;// dok outindex pokazuje na index outbuffer-a od kojeg upisujemo promenjene pixele u proceduri
IzlazniFajl.

Ucitavanje PROC
mov edx, OFFSET buffer
xor ebx,ebx
mov ebx,edx
add ebx,pokazivacNiza
mov ah,[ebx]
inc ebx
mov al,[ebx]
cmp ax,P2
je DrugiRed
mWrite <"Format slike je pogresan.">
call WriteWindowsMsg
call close_file

DrugiRed:
add pokazivacNiza,3; // Sada pokazivacNiza pokazuje na #.
add ebx,2
;// Prelazimo preko komentara.
Komentar:
inc pokazivacNiza
inc ebx
mov dl,[ebx]
```

```
cmp dl,0ah
jne Komentar

TreciRed:;//Treci red
inc pokazivacNiza
call Citaj;// Ucitali smo M.
mov M,ax
inc pokazivacNiza
call Citaj
mov N,ax;// Ucitali smo N.
inc pokazivacNiza
call Citaj
mov Lmax,ax;// Ucitali smo Lmax.

mWrite "Unesite zeljeno ime izlaznog fajla: "
mov edx,OFFSET outfilename
mov ecx,SIZEOF outfilename
call ReadString

call CreateOutputFile
mov  outfileHandle,eax

;// Prepisujemo prva cetiri reda u izlazni fajl.
;// Po izlasku iz petlje rucno upisujemo buffer[0], jer za vrednost ECX=0 nismo prosli kroz petlju.
mov ecx,pokazivacNiza
Prepisivanje:
mov al,buffer[ecx]
mov outbuffer[ecx],al
loop Prepisivanje

mov al,buffer[0]
mov outbuffer[ecx],al

inc pokazivacNiza;// pokazivacNiza pokazuje na prvi pixel.
mov eax,pokazivacNiza
mov outindex,eax

ret
Ucitavanje ENDP

;// PROCEDURA: FormiranjeHistograma
;// Koristimo proceduru Citaj da bismo dobili INT vrednosti, koje upisujemo u niz pixeli[].
;// Kada naidjemo na EOL, upisujemo vrednost -1.
;// Po izlasku iz procedure niz pixeli[] je popunjen vrednostima pixel-a originalne slike.

FormiranjeHistograma PROC

xor ebx,ebx;// Koristimo ga kao brojac
xor eax,eax
mov ax, M
mul N
mov ecx,eax

Petlja:
call Citaj
mov pixeli[ebx],ax;// Upisujemo svaki pixel slike.
imul eax,2;// EAX sada postaje indeks histograma. Mnozimo sa 2,jer je histogram 16 bita.
mov esi,eax
inc histogram[esi]
```

```
inc pokazivacNiza
add edx,pokazivacNiza
mov edi,[edx]
mov edx,edi
cmp dl,0ah;// Proveravamo da li nam pokazivacNiza trenutno pokazuje na EOL.
jnz Nastavi
inc ebx;// Povecavamo ebx za 2 jer je on indeks pixel-a, koji su WORD.
inc ebx
mov pixeli[ebx],-1
inc pokazivacNiza;// Pomeramo pokazivacNiza na prvu cifru narednog pixela
Nastavi:
inc ebx
inc ebx

loop Petlja

ret
FormiranjeHistograma ENDP

;// ObradaSlike
;// Koristimo formulu za promenu vrednosti pixel-a.
;// Vrednost histogram[edi] jednaka je broju ponavljanja pixel-a EDI. Njega prvo dodajemo sumi,
;// a zatim ga menjamo.
;// Po izlasku iz procedure u nizu histogram[] se nalaze nove vrednosti pixel-a.

ObradaSlike PROC



xor edi,edi
xor ecx,ecx
xor eax,eax
xor ebx,ebx
xor edx,edx

Obrada:

call NadjiPrag

mov bx, T3
cmp bx, T
jge Raste
Opada:
sub T, bx
mov bx,T
mov ax, T3
mov T, ax
cmp bx, PRAG
jge Obrada
jmp Kraj
Raste:
sub bx, T
mov ax, T3
mov T, ax
cmp bx, PRAG
jge Obrada

Kraj:
```

```
ret
ObradaSlike ENDP



;// IzlazniFajl
;// Menjamo niz pixeli[] novim vrednostima iz histogram-a.
;// Popunjavamo outbuffer[] pocevsi od vrednosti outindex, a zatim outbuffer[] saljemo
;// na izlazni fajl.

IzlazniFajl PROC
mov edx,OFFSET pixeli
xor ebx,ebx
xor edx,edx
xor edi, edi

mov ax,M
mul N
mov brojac,eax;// Koristimo brojac,jer zelimo da se osiguramo da necemo obraditi vrednosti koje
ukazuju na EOL.

;// Menjamo niz pixeli[] novim vrednostima iz histogram-a.
NoviPixeli:
cmp edi, 131072
jg Gotovo
mov ax,pixeli[edi]
cmp ax,-1
je VrednostNovogReda;// Ovim smo sigurni da smo obradili samo pixele, a ne i vrednosti koje ukazuju
na EOL.
imul eax,2
mov cx,T
cmp pixeli[edi], cx
jg  Belo
mov dx,0
mov pixeli[edi],dx
jmp SledeciPix
Belo:
mov dx,255
mov pixeli[edi],dx
jmp SledeciPix


VrednostNovogReda:
inc brojac
SledeciPix:
inc edi
inc edi
dec brojac
cmp brojac, 0
jne NoviPixeli

;// Ponovo podesavamo vrednosti brojaca i cistimo registre.
;// Promenljive M i N koristimo za smestanje vrednosti 100 i 10, jer nam dimenzije slike vise nisu
potrebne.

Gotovo:
mov ax,M
imul N
mov brojac,eax
```

```
xor eax,eax
xor ebx,ebx;// EBX je indeks od outbuffer BYTE, dok je EDI indeks od pixeli WORD.
xor edi,edi
mov ebx,outindex
mov M,100;//Ne moze neposredno div 100,i div 10 pa smestamo u M i N.
mov N,10

;// Petlja NoviIzlazni popunjava outbuffer novim vrednostima niza pixeli[].
;// Da bismo popunili outbuffer moramo pretvoriti INT u CHAR, tako da vrsimo proveru broja cifara INT,
;// a zatim na tu vrednost dodajemo 30h i tako dobijamo ascii vrednost broja.
;// Ako naidjemo na -1, signaliziramo kraj reda i upisujemo 0Ah u outbuffer. U tom slucaju brojac se ne
smanjuje.
NoviIzlazni:

mov ax,pixeli[edi]
cmp ax,-1
je KrajReda

xor edx,edx
div M
cmp ax,0;// Proveravamo da li je broj dvocifren.
je Dvocifren
Trocifren:;// Trocifreni broj
mov cifra,3
add al,30h
mov broj[0],al;// Stotine
mov ax,dx
xor edx,edx
div N
add al,30h
add dl,30h
mov broj[1],al;// Desetice
mov broj[2],dl;// Jedinice
jmp Ispis

Dvocifren:;// Dvocifreni broj
mov cifra,2
mov al,dl
xor edx,edx
div N
cmp al,0
je Jednocifren
add al,30h
add dl,30h
mov broj[0],al;// Desetice
mov broj[1],dl;// Jedinice
jmp Ispis

Jednocifren:;// Jednocifreni broj
mov cifra,1
add dl,30h
mov broj[0],dl;// Jedinice

;// Petlja kojom popunjavamo outbuffer.
Ispis:
xor edx,edx
xor eax,eax
mov eax,edi;// EAX sada cuva index od pixeli[] dok koristimo EDI za index niza broj.
xor edi,edi
```

```
Dodavanje:
mov dl,broj[edi]
mov outbuffer[ebx],dl
inc ebx
inc edi
dec cifra
cmp cifra,0
jnz Dodavanje
mov outbuffer[ebx],20h;// Ne povecavamo ebx, zato sto se to radi u labeli Sledeci.
jmp Sledeci

KrajReda:
mov outbuffer[ebx],0ah
mov eax,edi;// U slucaju kada nije kraj reda, EAX cuva vrednost EDI, koju vracamo u labeli Sledeci.
 ;// Ako udjemo u labelu KrajReda smestanje se nece desiti pa to radimo ovde zbog kasnije zamene.
inc brojac;// Jer ne treba da se smanjuje za kraj reda.

Sledeci:
inc ebx
mov edi,eax
inc edi
inc edi

dec brojac
cmp brojac,0
jne NoviIzlazni

;// Upisujemo vrednosti binarizovane slike outbuffer-om u izlazni fajl.
mov  eax, outfileHandle
mov  edx, OFFSET outbuffer
mov  ecx,BUFFER_SIZE
call WriteToFile

ret
IzlazniFajl ENDP

;//        MAIN
;// U glavnom programu se citava ulazna sliku i obradjuju se greske. Nako toga se pozivaju procedure
pripreme i obrade slike.


main PROC

mWrite <"Unesite naziv slike u formatu naziv.pgm: ">
mov edx,OFFSET filename
mov ecx,SIZEOF filename
call ReadString

mov edx,OFFSET filename
call OpenInputFile
mov fileHandle,eax

cmp eax,INVALID_HANDLE_VALUE
jne file_ok
mWrite <"Greska pri otvaranju fajla.",0dh,0ah>
call WriteWindowsMsg
jmp quit

file_ok:
```

```
mov edx,OFFSET buffer
mov ecx,BUFFER_SIZE
call ReadFromFile
jnc check_buffer_size

mWrite <"Greska pri citanju fajla. ",0dh,0ah>
call WriteWindowsMsg
mov eax, fileHandle
call close_file
jmp quit

check_buffer_size:
cmp eax,BUFFER_SIZE
jbe buf_size_ok

mWrite <"Greska: Dimenzije slike su neodgovarajuce.",0dh,0ah>
call WriteWindowsMsg
mov eax, fileHandle
call close_file
jmp quit

buf_size_ok:
mov velicinaBafera,eax

call Ucitavanje
call FormiranjeHistograma
call ObradaSlike
call IzlazniFajl


mov  eax,outfileHandle
call CloseFile

mov eax,fileHandle
call CloseFile

quit:
exit
main ENDP
END main
```