**Rīgas Tehniskā universitāte**

**Datorzinātnes un informācijas tehnoloģijas fakultāte**

Datorvadības, automātikas un datortehnikas institūts



**Atskaite par II. PRAKTISKO DARBu**

priekšmeta " Scēnu analīze"

**Izstrādāja: Igors Šemels**

**Pārbaudīja: O. Krutikova**

**Rīga, 2013.**

**Teorija**

Robertsa operators ir maska 2x2, kura tiek uzklāta uz katra attēla pikseļa. Izmantojot to, tiek meklēta gradienta vērtība.

Eksistē arī maskas 3x3 – Sobeļa operators, Previta operators un Frei-Čena operators. Visiem šiem operatoriem ir līdzīga darbība, tie atšķīrās tikai ar koeficientiem. kur Previta operatoram koeficients k=1, Sobeļa operatoram koeficients k=2, Frei-Čena operatoram koeficients ir k=√2.

Gradienta vērtība mainās robežās no 0 līdz 255. Izmantojot šo vērtību pie zīmēšanas var iegūt monohromatisko gradienta karti.

Uzliekot papildus slieksni, var normalizēt gradienta vērtības (piem., ja g(i,j)<128 tad g(i,j)=0, ja g(i,j)>128 tad g(i,j)=255). Tad var iegūt melnbalto gradienta karti, kurā tiek “atšķeltas” liekās gradienta vērtības.

**Programmas pirmkods ar komentāriem**

**Previta operators(melnbalts).**

procedure TMain\_menu.Previta1Click(Sender: TObject);

var i,j,k,gx,gy:integer;

begin k:=1;

for i:=1 to high(img)-1 do

for j:=1 to high(img[0]) do

begin

Gx:= round(

(1/(k+2))\*((img[i+1,j+1].I + k\*img[i+1,j].i + img[i+1,j-1].i )

- (img[i-1,j+1].i+k\*img[i-1,j].i+img[i-1,j-1].i)));

Gy:= round(

(1/(k+2))\*((img[i-1,j-1].I+ k\*img[i+1,j-1].i + img[i+1,j-1].i )

- (img[i-1,j+1].i+k\*img[i,j+1].i+img[i+1,j+1].i)));

img[i,j].edge:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

end;

imgToImage1(4);

end;

**Sobela operators(melnbalts)**

procedure TMain\_menu.Sobela1Click(Sender: TObject);

var i,j,k,gx,gy:integer;

begin

k:=2;

for i:=1 to high(img)-1 do

for j:=1 to high(img[0]) do

begin

Gx:= round(

(1/(k+2))\*((img[i+1,j+1].I + k\*img[i+1,j].i + img[i+1,j-1].i )

- (img[i-1,j+1].i+k\*img[i-1,j].i+img[i-1,j-1].i)));

Gy:= round(

(1/(k+2))\*((img[i-1,j-1].I+ k\*img[i+1,j-1].i + img[i+1,j-1].i )

- (img[i-1,j+1].i+k\*img[i,j+1].i+img[i+1,j+1].i)));

img[i,j].edge:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

end;

imgToImage1(5);

end;

**Frei-Čena operators(melnbalts).**

procedure TMain\_menu.FreiChena1Click(Sender: TObject);

var i,j,k,gx,gy:integer;

begin

k:=2;

for i:=1 to high(img)-1 do

for j:=1 to high(img[0]) do

begin

Gx:=round(1/((Power(k,1/2))+2))\*((img[i+1,j+1].I+(Power(k,1/2))\*img[i+1,j].i+ img[i+1,j-1].i ) - (img[i-1,j+1].i+(Power(k,1/2))\*img[i-1,j].i+img[i-1,j-1].i)));

Gy:=round((1/((Power(k,1/2))+2))\*((img[i-1,j-1].I+(Power(k,1/2))\*img[i+1,j-1].i+ img[i+1,j-1].i ) - (img[i-1,j+1].i+(Power(k,1/2))\*img[i,j+1].i+img[i+1,j+1].i))); img[i,j].edge:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

end; imgToImage1(6); end;

**Previta operators(krasains)**

procedure TMain\_menu.Previta2Click(Sender: TObject);

var i,j,k,gx,gy,tgx,tgy:integer;

begin

k:=1;

for i:=1 to high(img)-1 do

for j:=1 to high(img[0]) do

begin

tGx:= round((1/(k+2))\*((img[i+1,j+1].I + k\*img[i+1,j].i + img[i+1,j-1].i )- (img[i-1,j+1].i+k\*img[i-1,j].i+img[i-1,j-1].i)));

tGy:= round((1/(k+2))\*((img[i-1,j-1].I+ k\*img[i+1,j-1].i + img[i+1,j-1].i ) - (img[i-1,j+1].i+k\*img[i,j+1].i+img[i+1,j+1].i)));

Gx:= round((1/(k+2))\*((img[i+1,j+1].R + k\*img[i+1,j].R + img[i+1,j-1].R )- (img[i-1,j+1].R+k\*img[i-1,j].R+img[i-1,j-1].R)));

Gy:= round((1/(k+2))\*((img[i-1,j-1].R+ k\*img[i+1,j-1].R + img[i+1,j-1].R ) - (img[i-1,j+1].R+k\*img[i,j+1].R+img[i+1,j+1].R)));

***Gradienta vērtība mainās robežās no 0 līdz 255***

if tgx < 128 then gx := 0;

if tgy > 128 then gy := 255;

img[i,j].r:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

Gx:= round((1/(k+2))\*((img[i+1,j+1].G + k\*img[i+1,j].G + img[i+1,j-1].G )- (img[i-1,j+1].G+k\*img[i-1,j].G+img[i-1,j-1].G)));

Gy:= round((1/(k+2))\*((img[i-1,j-1].G+ k\*img[i+1,j-1].G + img[i+1,j-1].G ) - (img[i-1,j+1].G+k\*img[i,j+1].G+img[i+1,j+1].G)));

***Gradienta vērtība mainās robežās no 0 līdz 255***

if tgx < 128 then gx := 0;

if tgy > 128 then gy := 255;

img[i,j].g:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

Gx:= round((1/(k+2))\*((img[i+1,j+1].B + k\*img[i+1,j].B + img[i+1,j-1].B )- (img[i-1,j+1].B+k\*img[i-1,j].B+img[i-1,j-1].B)));

Gy:= round((1/(k+2))\*((img[i-1,j-1].B+ k\*img[i+1,j-1].B + img[i+1,j-1].B ) - (img[i-1,j+1].B+k\*img[i,j+1].B+img[i+1,j+1].B)));

***Gradienta vērtība mainās robežās no 0 līdz 255***

if tgx < 128 then gx := 0;

if tgy > 128 then gy := 255;

img[i,j].b:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

end;

imgToImage1(7);

end;

**Sobela operators(krasains)**

procedure TMain\_menu.Sobela2Click(Sender: TObject);

var i,j,k,gx,gy,tgx,tgy:integer;

begin // sobela kras

k:=2;

for i:=1 to high(img)-1 do

for j:=1 to high(img[0]) do

begin

tGx:= round((1/(k+2))\*((img[i+1,j+1].I + k\*img[i+1,j].i + img[i+1,j-1].i )- (img[i-1,j+1].i+k\*img[i-1,j].i+img[i-1,j-1].i)));

tGy:= round((1/(k+2))\*((img[i-1,j-1].I+ k\*img[i+1,j-1].i + img[i+1,j-1].i ) - (img[i-1,j+1].i+k\*img[i,j+1].i+img[i+1,j+1].i)));

Gx:= round((1/(k+2))\*((img[i+1,j+1].R + k\*img[i+1,j].R + img[i+1,j-1].R )- (img[i-1,j+1].R+k\*img[i-1,j].R+img[i-1,j-1].R)));

Gy:= round((1/(k+2))\*((img[i-1,j-1].R+ k\*img[i+1,j-1].R + img[i+1,j-1].R ) - (img[i-1,j+1].R+k\*img[i,j+1].R+img[i+1,j+1].R)));

if tgx < 128 then gx := 0;

if tgy > 128 then gy := 255;

img[i,j].r:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

Gx:= round((1/(k+2))\*((img[i+1,j+1].G + k\*img[i+1,j].G + img[i+1,j-1].G )- (img[i-1,j+1].G+k\*img[i-1,j].G+img[i-1,j-1].G)));

Gy:= round((1/(k+2))\*((img[i-1,j-1].G+ k\*img[i+1,j-1].G + img[i+1,j-1].G ) - (img[i-1,j+1].G+k\*img[i,j+1].G+img[i+1,j+1].G)));

if tgx < 128 then gx := 0;

if tgy > 128 then gy := 255;

img[i,j].g:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

Gx:= round((1/(k+2))\*((img[i+1,j+1].B + k\*img[i+1,j].B + img[i+1,j-1].B )- (img[i-1,j+1].B+k\*img[i-1,j].B+img[i-1,j-1].B)));

Gy:= round((1/(k+2))\*((img[i-1,j-1].B+ k\*img[i+1,j-1].B + img[i+1,j-1].B ) - (img[i-1,j+1].B+k\*img[i,j+1].B+img[i+1,j+1].B)));

if tgx < 128 then gx := 0;

if tgy > 128 then gy := 255;

img[i,j].b:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

end; imgToImage1(8); end;

**Frei-Čena operators(krasains).**

procedure TMain\_menu.FreiChena2Click(Sender: TObject);

var i,j,k,gx,gy,tgx,tgy:integer;

begin // Frei-Chena kras

k:=2;

for i:=1 to high(img)-1 do

for j:=1 to high(img[0]) do

begin

tGx:= round((1/((Power(k,1/2))+2))\*((img[i+1,j+1].I + (Power(k,1/2))\*img[i+1,j].i + img[i+1,j-1].i )- (img[i-1,j+1].i+(Power(k,1/2))\*img[i-1,j].i+img[i-1,j-1].i)));

tGy:= round((1/((Power(k,1/2))+2))\*((img[i-1,j-1].I+ (Power(k,1/2))\*img[i+1,j-1].i + img[i+1,j-1].i ) - (img[i-1,j+1].i+(Power(k,1/2))\*img[i,j+1].i+img[i+1,j+1].i)));

Gx:= round((1/((Power(k,1/2))+2))\*((img[i+1,j+1].R + (Power(k,1/2))\*img[i+1,j].R + img[i+1,j-1].R )- (img[i-1,j+1].R+(Power(k,1/2))\*img[i-1,j].R+img[i-1,j-1].R)));

Gy:= round((1/((Power(k,1/2))+2))\*((img[i-1,j-1].R+ (Power(k,1/2))\*img[i+1,j-1].R + img[i+1,j-1].R ) - (img[i-1,j+1].R+(Power(k,1/2))\*img[i,j+1].R+img[i+1,j+1].R)));

if tgx < 128 then gx := 0;

if tgy > 128 then gy := 255;

img[i,j].r:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

Gx:= round((1/((Power(k,1/2))+2))\*((img[i+1,j+1].G + (Power(k,1/2))\*img[i+1,j].G + img[i+1,j-1].G )- (img[i-1,j+1].G+(Power(k,1/2))\*img[i-1,j].G+img[i-1,j-1].G)));

Gy:= round((1/((Power(k,1/2))+2))\*((img[i-1,j-1].G+ (Power(k,1/2))\*img[i+1,j-1].G + img[i+1,j-1].G ) - (img[i-1,j+1].G+(Power(k,1/2))\*img[i,j+1].G+img[i+1,j+1].G)));

if tgx < 128 then gx := 0;

if tgy > 128 then gy := 255;

img[i,j].g:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

Gx:= round((1/((Power(k,1/2))+2))\*((img[i+1,j+1].B + (Power(k,1/2))\*img[i+1,j].B + img[i+1,j-1].B )- (img[i-1,j+1].B+(Power(k,1/2))\*img[i-1,j].B+img[i-1,j-1].B)));

Gy:= round((1/((Power(k,1/2))+2))\*((img[i-1,j-1].B+ (Power(k,1/2))\*img[i+1,j-1].B + img[i+1,j-1].B ) - (img[i-1,j+1].B+(Power(k,1/2))\*img[i,j+1].B+img[i+1,j+1].B)));

if tgx < 128 then gx := 0;

if tgy > 128 then gy := 255;

img[i,j].b:=round( Power(Power(Gx,2)+Power(Gy,2),1/2));

end; imgToImage1(9); end;

**Roberts operators(krasains)**

procedure TMain\_menu.Roberts2Click(Sender: TObject);

var i,j:integer;

begin // roberts kras

for i:=1 to high(img)-1 do

for j:=1 to high(img[0]) do

begin

img[i,j].R:=round(Power(Power(img[i,j].R-img[i+1,j+1].R,2)+Power(img[i,j+1].R-img[i+1,j].R,2),1/2));

img[i,j].G:=round(Power(Power(img[i,j].G-img[i+1,j+1].G,2)+Power(img[i,j+1].G-img[i+1,j].G,2),1/2));

img[i,j].B:=round(Power(Power(img[i,j].B-img[i+1,j+1].B,2)+Power(img[i,j+1].B-img[i+1,j].B,2),1/2));

end; imgToImage1(10);end;

**Attēla saglabašāna.**

var l:Integer;

begin

l:=1;

If dlgSave1.Execute then begin

if dlgSave1.FilterIndex = l then begin

JPG.Assign(Image1.Picture.Bitmap);

JPG.SaveToFile(dlgSave1.FileName + '.jpg')

end else begin

BMP.Assign(Image1.Picture.Bitmap);

BMP.SaveToFile(dlgSave1.FileName + '.bmp');

end; end;

**Ekrānšāviņi ar algoritma darbības rezultātu**

Roberts melnbalts. Previta melnbalts

Sobela melnbalts Frei-Čena melnbalts

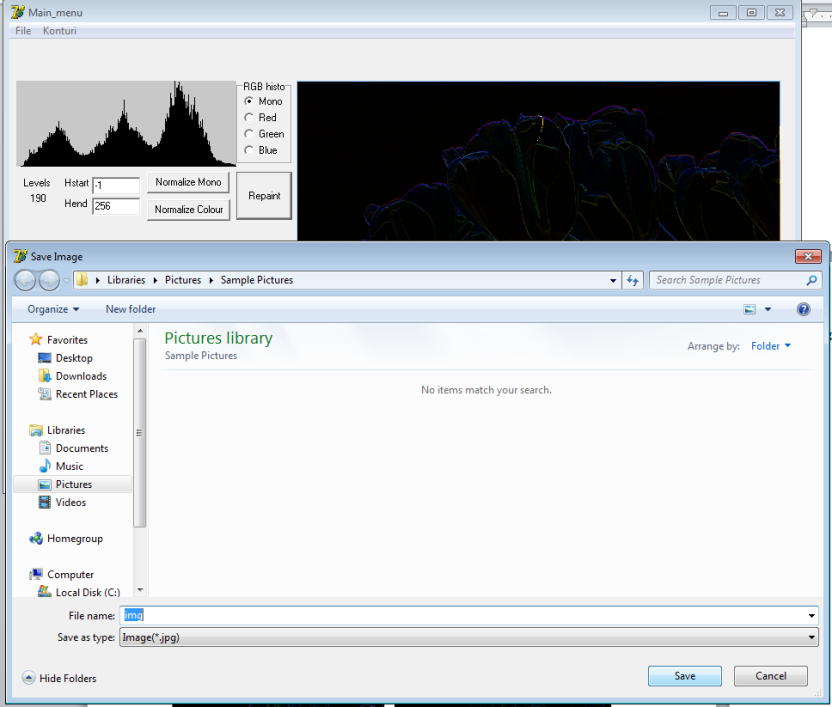
 

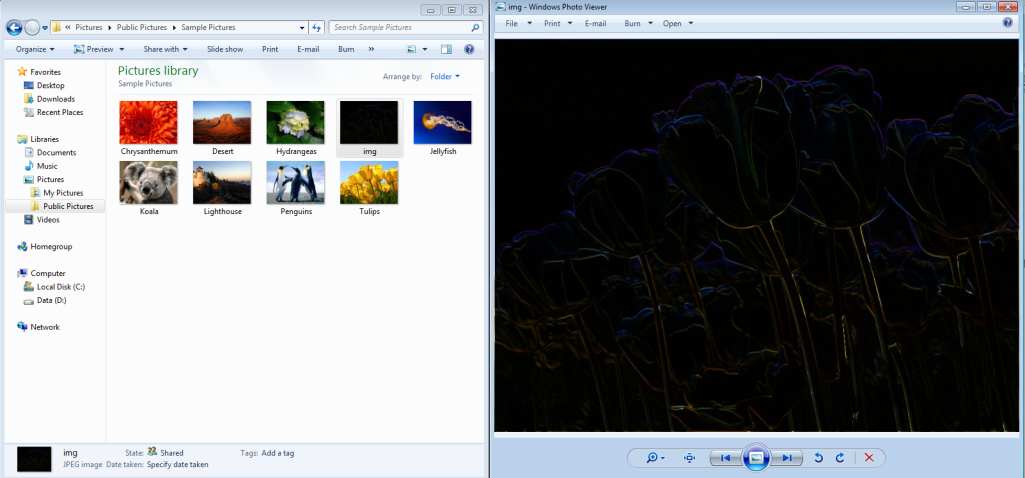
Roberts krasains Previta krasains

Sobela krasains Frei-Čena krasains

**Saglabašana.**





**Secinājūmi:**

Tiek izstradats otrais laboratorijas darbs, kura laika tika realizeta kontura darbibas ar atteliem. Konturu var izdalit ar melnbaltu un krasainu krasu un attēlu pēc kontura izdališanas var zaglabat divos formatos (JPG,BMP). Laboratorijas darbs tika izstradats sekmigi.