UNIVERZITET U BEOGRADU

ELEKTROTEHNIČKI FAKULTET

****

**PRIKAZ HISTOGRAMA, KARAKTERISTIKA I PERFORMANSI FILTERA NA ANDROID OPERATIVNOM SISTEMU**

Mentor: Studenti:

dr Irini Reljin, vanr. prof. Sofija Purić, dipl. ing.

Milan Mladenović, dipl. ing.

Beograd, jul 2014.

Sadržaj

[1. Uvod 3](#_Toc391810083)

[2. Postavka problema 3](#_Toc391810084)

[3. Karakteristike obrađenih filtera 3](#_Toc391810085)

[3.1. Primena unarnih filtera nad slikom 3](#_Toc391810086)

[3.1.1 Invert filter 4](#_Toc391810087)

[3.1.2 Black&White filter 5](#_Toc391810088)

[3.2. Primena binarnih filtera nad slikama 7](#_Toc391810089)

[3.3. Primena konvolucionih filtera nad slikom 7](#_Toc391810090)

[4. Primena histograma 7](#_Toc391810091)

[5. Pregled performansi filtera 7](#_Toc391810092)

[6. Zaključak 7](#_Toc391810093)

[7. Literatura 7](#_Toc391810094)

# Uvod

# Postavka problema

# Karakteristike obrađenih filtera

Primena filtera nad slikom predstavlja proces gde se nad informacijom koju nosi piksel, dakle *RGB* vrednosti, primenjuje matematička funkcija da bi se proizvela odgovarajuća alternativna verzija slike. Dakle, filteri predstavljaju programske operacije koje nam omogućuju primenu različitih efekata nad slikama.

Slika je funkcija  za neko N i M iz skupa prirodnih brojeva, pri čemu su N i M dimenzije slike u pikselima. Svaki piksel predstavljen je sa tri komponente: crvena (*R*), zelena (*G*) i plava (*B*), i to u opsegu od *0...255* za svaku komponentu. Neka je skup slika označen sa *I.* Filter predstavlja funkciju  koja preslikava postojeću sliku u novu sliku uz promenu (*RGB*) vrednosti svakog piksela po odgovarajućem algoritmu. Ako je skup svih filtera označen sa *F,* onda simbol *f(I)* predstavlja primenu nekog od filtera iz skupa *F* nad nekom od slika iz skupa slika *I.*

U ovom poglavlju biće dat pregled svih obrađenih filtera nad izabranom slikom, koji su podeljeni u tri kategorije: unarni, binarni i konvolucioni filteri. Biće opisan način rada samog filtera, kao i način na koji je filter implementiran u aplikaciji uz prikaz delova koda. Osim toga biće prikazan i rad tih filtera uz slikovit prikaz delova aplikacije.

## Primena unarnih filtera nad slikom

Pod pojmom unarnih filtera podrazumevamo one koji kao ulazni parametar zahtevaju samo jednu sliku nad kojom se primenjuje taj filter i kao rezultat vraćaju novu sliku nakon primene tog filtera. Pored toga unarni filteri mogu imati i argumente koji mogu predstavljati intenzitet primene ili jačinu neke od *RGB* komponenti koje su potrebne za primenu tog filtera nad slikom.

Filteri koji će biti opisani u nastavku rada su sledeći: invert, black&white, brightness, contrast, flip vertical, flip horizontal, grayscale, gamma correction, color filter, saturation, hue i shading filter.

### Invert filter

Invert filter, poznat još pod imenom negativ, filter je koji menja originalnu vrednost piksela u njegovu inverznu vrednost, odnosno invertuje vrednost svakog piksela. Svaki piksel na slici čuva informacije o četiri kanala: alpha, red, green i blue. Međutim, alpha ne odražava vizuelno boju slike, tako da se inverzija primenjuje na ostala tri kanala, po formuli:

*nova\_vrednost* (*R, G, B*) = 0xFF – *trenutna\_vrednost*(*R, G, B*) (3.1)

*nova\_vrednost* (*R, G, B*) = 0xFF – *trenutna\_vrednost*(*R, G, B*) (3.1)

Najvažniji deo koda koji obavlja invertovanje piksela prikazan je u tabeli 3.1:

Tabela 3.1: Invertovanje svih piksela na slici

**for** (**int** x=0; x<width; x++) {

**for** (**int** y=0; y<height; y++) {

r=255-Color.*red*(colorArray[y\*width+x]);

g=255-Color.*green*(colorArray[y\*width+x]);

b=255-Color.*blue*(colorArray[y\*width+x]);

colorArray[y\*width+x]=Color.*rgb*(r,g,b);

returnBitmap.setPixel(x,y,colorArray[y\*width+x]);

}

}

Pri tome, width označava širinu slike, height visinu, a colorArray je niz celobrojnih vrednosti u kome pamtimo vrednosti piksela (ima width\*height elemenata). Vrednosti komponenata piksela (alpha, red, green, blue) su uskladištene u vrednost piksela na Android operativnom sistemu na sledeći način:

*pixel* = (*alpha* << 24) | (*red* << 16) | (*green* << 8) | *blue* (3.2)

*pixel* = (*alpha* << 24) | (*red* << 16) | (*green* << 8) | *blue* (3.2)

Metode Color.red, Color.green i Color.blue vraćaju crvenu, zelenu i plavu komponentu odgovarajuće vrednosti piksela koja se prosleđuje kao argument metode, respektivno. Zbog poznatog načina skladištenja pojedinačnih komponenti piksela, te vrednosti smo mogli dobiti i na sledeći način:

*red* = (*color* >> 16) & 0xFF (3.3)

*green* = (*color* >> 8) & 0xFF (3.4)

*blue* = *color* & 0xFF (3.5)

*red* = (*color* >> 16) & 0xFF (3.3)

*green* = (*color* >> 8) & 0xFF (3.4)

*blue* = *color* & 0xFF (3.5)

Metoda Color.rgb(r, g, b) vraća celobrojnu vrednost piksela na osnovu vrednosti pojedinačnih komponenti koje se prosleđuju kao argumenti poziva metode, a koje treba da budu u opsegu [0..255]. Vrednost alpha komponente je implicitno 255, što označava potpunu neprozračnost (neprovidnost). Metoda returnBitmap.setPixel(x, y, colorArray[y\*width+x]) upisuje odgovarajuću vrednost piksela na odgovarajuće mesto u slici (na specificiranu x i y koordinatu piksela). Slika je na Android operativnom sistemu predstavljena preko objekta Bitmap.

Na slikama 3.1 i 3.2 prikazane su originalna slika i slika nakon primene invert filtera:

### Black&White filter

Black and white predstavlja filter koji za postojeću sliku koja se zadaje, kao i *scale* parametar koji predstavlja intenzitet, vraća sliku čiji pikseli mogu imati samo crnu ili belu boju. Tj vrednosti komponenti *RGB* za sve piksele mogu biti ili *0,0,0* ili *255,255,255*. Filter prihvata sliku kao prvi argument, i parametar *scale* koji predstavlja realan broj od *0* do *100.* Filter zatim pravi lokalnu kopiju prosleđene slike, i računa prag na osnovu *scale* parametra po formuli:



Zatim se vrši obilazak matrice i za svaki piksel određuje se njegovo osvetljenje po formuli.



Pri čemu je *x[i][j]* piksel na poziciji *i,j,* a anotacija *.red, .green, .blue,* označava da se uzima crvena (*R*), zelena (*G*), plava (*B*) komponenta boje tog piksela. Nakon toga se na osnovu osvetljenosti piksela i praga računa njegova nova vrednost i to ako je *newValue > t* uzima se 255, ako je *newValue < t* uzima se *0*, ako je *t > 127.5* uzima se *255*, ako je *t < 127.5* uzima se *0* i na kraju ako nije ni jedan od prethodnih uslova uzima se *255.* Zatim se nova vrednost postavlja kao parametar za sve tri komponente (*RGB*) piksela.

scale = (scale/100)\*255;

for(int i = 0; i < width; i++) {

for(int j = 0; j < height; j++) { r = Color.red(colorArray[j \* width + i]);

g = Color.green(colorArray[j \* width + i]);

b = Color.blue(colorArray[j \* width + i]);

newValue=0.2126\*r+0.7152\*g+0.0722\*b; if (newValue>scale) newValue=255; else if (newValue<scale) newValue=0; else if (scale>127.5) newValue=255; else if (scale<127.5) newValue=0;

else newValue=255;

colorArray[j \* width + i] = Color.rgb((int)newValue, (int)newValue, (int)newValue);

returnBitmap.setPixel(i, j, colorArray[j \* width + i]);

}

}

## Primena binarnih filtera nad slikama

## Primena konvolucionih filtera nad slikom

Konvolucioni filteri su naziv dobili na osnovu matematičke konvolucije, koja predstavlja primenu operacije nad dve funkcije *f* i *g* čime se proizvodi nova fukcina.



Kod procesiranja slika, termin kernel, konvoluciona matrica ili maska predstavljaju matricu koja se primenom konvolucije nad slikom za odgovarajuće vrednosti matrice koje definiše filter dobijaju efekti nad slikama. U samom projektu...

# Primena histograma

# Pregled performansi filtera

# Zaključak

# Literatura