УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
Трг Доситеја Обрадовића 6, 21000 Нови Сад, Република Србија
Деканат: 021 6350-413; 021 450-810; Централа: 021 485 2000
Рачуноводство: 021 458-220; Студентска служба: 021 6350-763
Телефакс: 021 458-133; e-mail: ftndean@uns.ac.rs
ИНТЕГРИСАНИ СИСТЕМ МЕНАЏМЕНТА СЕРТИФИКОВАН ОД:

PROJEKAT IZ RAČUNARSKE ELEKTRONIKE

NAZIV PROJEKTA:

Detekcija ivice preko algoritma “Canny”

TEKST ZADATKA:

Detekcija ivice na slici fotografisanoj preko Rasberi kamere v2, koristeći algoritam “Canny”, napisan u programskim jezicima C++ i Pajton i koji se izvršava na „Rasberi Paj“ uređaju.

MENTOR PROJEKTA:

dr Ivan Mezei, vanredni profesor

PROJEKAT IZRADIO:

Vladimir Vincan EE5-2015

DATUM ODBRANE PROJEKTA:

19.9.2019.

Sadržaj

1. Uvod

Detekcija ivice podrazumeva pronalaženje tačaka na slici u digitalnom formatu, u kojima se osvetljenje ili boja naglo menjaju. Pronađene tačke su uglavnom organizovane kao skup krivolinijskih segmenata koje nazivamo ivice. Detekcija ivice je bazičan algoritam prilikom obrade slike, jer pomoću njega detektujemo prisustvo, položaje i veličinu različitih kontura i oblika na slici, što nam značajno olakšava njenu dalju obradu. Postoji više algoritama za detekciju ivica, od čega je najpoznatiji algoritam „Canny“.

Cilj ovog projekta je da „Canny“ algoritam, implementiran u programskim jezicima C++ i Pajton, obradi sliku dobijenu pomoću Rasberi kamere v2. Algoritam bi se izvršavao na Rasberi Paj uređaju. Sam „Canny“ algoritam, kao i komunikacija sa Rasberi kamerom, bi bio implementiran u programskom jeziku Pajton, dok bi vremenski kritičan deo, konvolucija slike sa kernelom, bio implementiran u programskom jeziku C++. Kodovi napisani u programskim jezicima Pajton i C++ bi komunicirali putem deljenih objekata i razmenjivali podatke o dimenzijama slike, kao i samim vrednostima unutar matrica kernela i date slike.

Projekat je opisan kroz pet poglavlja od kojih je **prvo** poglavlje uvodno.

U **drugom** poglavlju je opisan cilj projekta

U **trećem** poglavlju je detaljno objašnjen svaki funkcionalni blok.

U **četvrtom** poglavlju je dat zaključak projekta.

Poslednje, **peto** poglavlje prikazuje korišćenu literaturu.

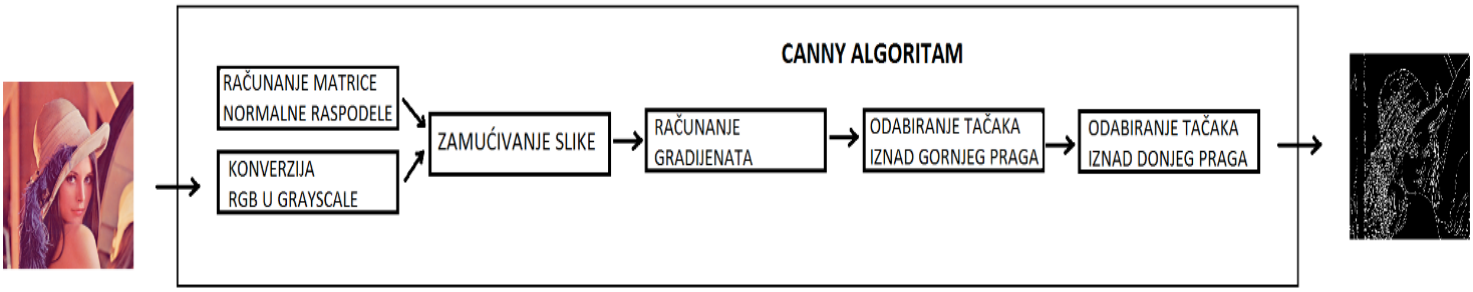
2. Analiza problema

Cilj ovog projekta je, kao što je već navedeno, implementirati algoritam za detekciju ivica na slici u digitalnom formatu. Principska šema algoritma, kojom se najbolje može objasniti šta se treba postići, prikazana je na slici ispod. Ulaz algoritma je slika u RGB formatu, a izlaz je crno-bela slika, gde su belim obeležene ivice originalne slike.

Postoji više algoritama za određivanje ivice, kao što su „Canny“ i „Deriche“. U ovom projektu je implementiran „Canny“. Takođe, moguće je koristiti različite operatore (kernele), čime se postižu drugačija svojstva izlazne slike. U ovom projektu je korišćen „Sobel“ operator *(pogledati poglavlje 3.4)*.

Da bi se značajno povećala brzina obrade slike, neophodno je koristiti brzu Furijeovu transformaciju prilikom konvolucije slike i kernela, jer smanjuje vreme izvršavanja sa $\mathcal{O}(M^2 \cdot N^2)$ na $\mathcal{O}(M \cdot N \cdot (log(M) + log(N))$, gde m i n predstavljaju dimenzije slike (pod uslovom da su dimenzije kernela male, što jeste slučaj). U ovom projektu je brza Furijeova transformacija implementirana u C++ jeziku, jer je vreme izvršavanja koda napisanog u kompajlerskom (C++) jeziku značajno kraće u odnosu na isti kod napisan u interpreterskom (Pajton) jeziku.

3. Detaljan opis algoritma



„Canny“ algoritam se izvršava po principskoj šemi, prikazanoj na prethodnoj slici. U svakom bloku se izvršavaju operacije nad slikom dobijenom u prethodnom bloku *(ili ulazu)*. Rad svakog bloka je objašnjen u narednim poglavljima.

3.1. Konverzija RGB slike u Grayscale sliku

Piksel prikazan u RGB formatu se pretvara u Grayscale format na sledeći način:

$$gray = 0.2989 * r + 0.5870 * g + 0.1140 * b$$

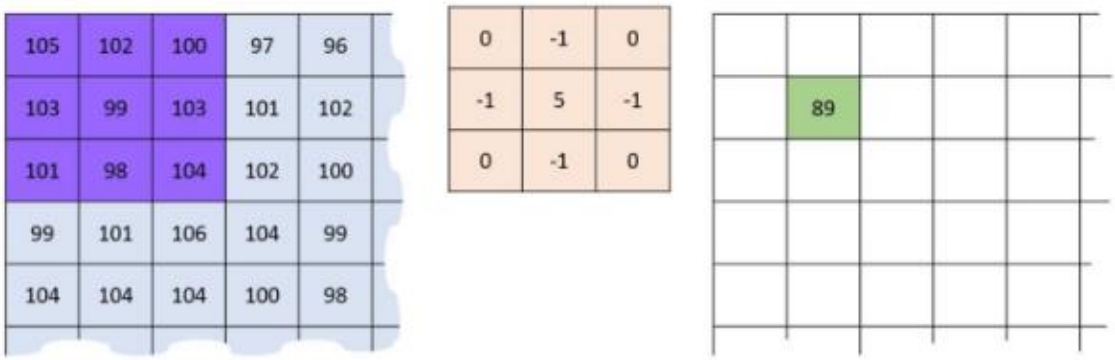
Gde su r, g i b realni brojevi, koji predstavljaju crvenu, zelenu, i plavu komponentu piksela, redom.

3.2. Konvolucija slike sa kernelom

Kernel predstavlja matricu neparnih dimenzija male veličine. Konvolucija je operacija dodavanja svakog elementa svojim susedima, pomnoženog sa odgovarajućim koeficijentom. Ako imamo dve 3x3 matrice, pri čemu je prva kernel, a druga parče originalne slike, iste dimenzije kao i kernel, onda konvolucija se definiše na sledeći način:

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)[2,2] = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9).$$

Rezultat konvolucije se smešta u novu matricu, na istu poziciju na kojoj se nalazi i centralno polje parčeta originalne slike *(pogledati sliku ispod, polje [2,2])*. Slično se definiše i za veće dimenzije matrica.



Algoritam za računanje konvolucije po definiciji je prespor, složenosti $\mathcal{O}(N^3)$. Zbog toga je neophodno ubrzati dati proces, što se postiže sa Furijeovim transformacijama, i složenost se smanjuje na $\mathcal{O}(N^2 \log N)$. U nastavku će biti data matematička definicija konvolucije, kao i detaljno izvođenje formula za konvoluciju pomoću Furijeove transformacije.

3.2.1. Konvolucija u jednoj dimenziji

Posmatrajmo dva niza, *f[n]* i *g[m]*, sa brojem elemenata *N* i *M* ($N \geq M$), redom. Pomoću njih će biti objašnjena dva osnovna tipa diskretne konvolucije – linearna i kružna.

- **Linearna konvolucija** se zasniva na proširivanju signala *f* i *g* sa nulama i sa leve i sa desne strane, do beskonačnosti, i potom izvršavanje sledeće operacije:

$$h_\infty[k] = (f * g)[k] \triangleq \sum_{i=-\infty}^{\infty} f_\infty[i] \cdot g_\infty[k-i] = (g * f)[k]$$

  Pri čemu važi:

$$f_\infty[n] = \begin{cases} f[n], 0 \leq n \leq N \\ 0, inače \end{cases}$$

$$g_\infty[m] = \begin{cases} g[n], 0 \leq m \leq M \\ 0, inače \end{cases}$$

  Rezultujući signal h_∞ može imati maksimalno *N+M+1* nenultih vrednosti. U zavisnosti od toga da li konvolucioni proizvod uključuje sve nenulte članove dobijenog niza h_∞, razlikujemo tri tipa konvolucionog množenja:

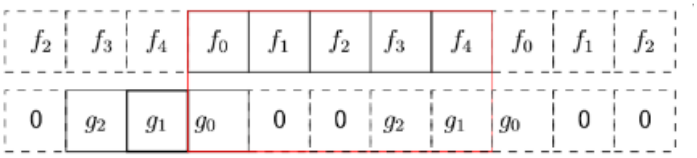
  - **Potpuno množenje** – rezultujući niz *h* će uključivati sve elemente niza h_∞, koji mogu imati nenultu vrednost. Niz *h* će posedovati *N+M+1* elemenata.
  - **Množenje iste veličine** – rezultujući niz *h* će uključivati samo članove niza h_∞, kod kojih je centralni element niza *g* pomnožen sa sa nekim članom neproširenog niza *f*. Niz *h* će posedovati *N* elemenata.
  - **Validno množenje** – rezultujući niz *h* će uključivati samo članove niza h_∞, kod kojih su svi neproširenі elementi niza *g* pomnoženi sa sa neproširenim članovima niza *f*. Niz *h* će posedovati *N-M+1* elemenata.

  Dalje u radu će se pod linearnim konvolucionim množenjem podrazumevati potpuno množenje, osim ukoliko ne bude drugačije naznačeno, pošto je algoritam za konvoluciju implementiran pomoću potpunog množenja. Matematički, formula za potpuno konvoluciono množenje se definiše na sledeći način:

$$h[k] \triangleq \sum_{i=\max(0,k-M+1)}^{\min(N-1,k)} f[i] \cdot g[k-i], \qquad \forall k \in [0, N+M-2]$$

- **Kružna konvolucija** se zasniva na „obmotavanju“ nizova *f* i *g* sa sopstvenim članovima, umesto što se proširuju sa nulom do beskonačnosti. Time se postiže da novodobijeni nizovi

f_P i g_P budu periodični sa istim periodom P. Konačni nizovi f i g su prošireni sa nulama tek toliko, da bi se zadovoljio uslov da nizovi budu iste periodičnosti *($P = N_P = M_P \geq N \geq M$).* Drugim rečima, za bilo koji ceo broj k, važi *f(k) = f(k mod P)* i *g(k) = g(k mod P).* Na slici ispod se nalazi primer za *P=N=5* i *M=3.*



Matematički, formula za potpuno konvoluciono množenje sa modulom P se definiše na sledeći način:

$$h_P[k] = (f *_P g)[k] \triangleq \sum_{i=0}^{P-1} f_P[i] \cdot g_P[k-i] = (g *_P f)[k], \qquad \forall k \in \mathbb{Z}$$

Pri čemu važi:

$$f_P[i] = \sum_{p=-\infty}^{\infty} f_\infty[i + p \cdot P] = f_\infty[i \bmod P], \ \forall i \in \mathbb{Z}$$

$$g_P[i] = \sum_{p=-\infty}^{\infty} g_\infty[i + p \cdot P] = g_\infty[i \bmod P], \ \forall i \in \mathbb{Z}$$

U slučaju da je moduo kružne konvolucije P isti kao veličina niza f *($P = N$)*, jednostavno se može pokazati da linearna i kružna konvolucija predstavljaju identičnu operaciju:

$$(f * g)[k] = (f *_P g)[k], \quad ako\ P = N$$

Uloga i značaj pojma kružne konvolucije je u tome što ona predstavlja spregu između diskretne Furijeove transformacije i linearne konvolucije signala.

3.2.2. Konvolucija u dve dimenzije

Celokupna teorija diskutovana za konvoluciju u jednoj dimenziji može biti primenjena i na konvoluciju u više dimenzija. Ovaj rad će biti ograničen na dve dimenzije, odnosno ograničen na konvolucije nad matricama. Linearna konvolucija nad matricama f i g se definiše na sledeći način:

$$h_\infty[k_1, k_2] \triangleq (f * g)[k_1, k_2] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f_\infty[i, j] \cdot g_\infty[k_1 - i, k_2 - j] = (g * f)[k_1, k_2]$$

Pri čemu važi:

$$f_\infty[i, j] = \begin{cases} f[i, j], 0 \leq i \leq H_1, 0 \leq j \leq W_1, \\ 0, inače \end{cases}$$

$$g_\infty[i, j] = \begin{cases} g[i, j], 0 \leq i \leq H_2, 0 \leq j \leq W_2, \\ 0, inače \end{cases}$$

Potpuna linearna konvolucija će biti matrica veličine *(H_1+H_2-1, W_1+W_2-1)*. Kružno konvoluciono množenje nad matricama *f* i *g* se definiše na sledeći način:

$$h_P[k_1,k_2] \triangleq \left(f *_{P_1,P_2} g\right)[k_1,k_2] = \sum_{i=0}^{P_1-1}\sum_{j=0}^{P_2-1} f[i,j]\cdot g[k_1-i,k_2-j] = \left(g *_{P_1,P_2} f\right)[k_1,k_2],$$

$$\forall k_1,k_2 \in \mathbb{Z}$$

Pri čemu važi:

$$f_{P_H,P_W}[i,j] = \sum_{p_1=-\infty}^{\infty}\sum_{p_2=-\infty}^{\infty} f_\infty[i+p_1\cdot P_H, j+p_2\cdot P_W] = f_\infty[i \bmod P_H, j \bmod P_W], \quad \forall i,j \in \mathbb{Z}$$

$$g_{P_H,P_W}[i,j] = \sum_{p_1=-\infty}^{\infty}\sum_{p_2=-\infty}^{\infty} g_\infty[i+p_1\cdot P_H, j+p_2\cdot P_W] = g_\infty[i \bmod P_H, j \bmod P_W], \ \forall i,j \in \mathbb{Z}$$

I u dvodimenzionalnom slučaju se može pokazati da kružna i linearna konvolucija predstavljaju identičnu operaciju ukoliko važi da je $P_H = H_1 \geq H_2$ i $P_W = W_1 \geq W_2$

3.2.3. Diskretna Furijeova transformacija u jednoj dimenziji

Diskretna Furijeova transformacija nad diskretnim signalom *f[n]* dužine *N* se definiše na sledeći način:

$$F[k] = \mathcal{F}\{f\}[k] \triangleq \sum_{n=0}^{N-1} f[n] e^{-\frac{2i\pi kn}{N}} = \sum_{n=0}^{N-1} f[n] W_N^{kn}$$

Vrednosti kompleksne eksponencijalne funkcije sa kojima se množe odbirci diskretnog signala *f[n]* nazivaju se rotacioni „twiddle“ faktori. Rotacioni faktori imaju sledeće osobine, koji će biti značajni za izvođenje brze Furijeove transformacije:

- Kompleksno konjugovana simetričnost

$$W_N^{k(N-n)} = W_N^{-kn} = \left(W_N^{kn}\right)^C$$

- Periodičnost

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$$

- Redundantnost

$$W_N^{2kn} = W_{N/2}^{kn}$$

Inverzna diskretna Furijeova transformacija nad istim signalom se definiše na sledeći način:

$$f[n,m] = \mathcal{F}^{-1}\{F\}[n] \triangleq \frac{1}{N}\sum_{k=0}^{N-1} F[k] e^{\frac{2i\pi kn}{N}} = \frac{1}{N}\left(\sum_{k=0}^{N-1} F[k]^C e^{-\frac{2i\pi kn}{N}}\right)^C$$

Iz formule za inverznu Furijeovu transformaciju se može zaključiti da Furijeova transformacija nad konjugovanim signalom u frekvencijskom domenu predstavlja inverznu Furijeovu transformaciju. Konjugacija nad rezultujućim signalom nije neophodna, pošto za dati algoritam su značajne samo realne vrednosti inverzne Furijeove transformacije. Korišćenjem date osobine se povećava konciznost koda, i omogućava se korišćenje već realizovane funkcije za direktnu Furijeovu transformaciju, sa neznatnim usporavanjem vremena izvršavanja. U embeded aplikacijama, sa memorijskim ograničenjima, ova osobina se može pokazati veoma korisnom.

3.2.4. Diskretna Furijeova transformacija u dve dimenzije

Diskretna Furijeova transformacija nad diskretnim višedimenzionim signalom *f[n]* se definiše analogno kao u jednodimenzionom slučaju. Formula za Furijeovu transformaciju u dve dimenzije nad matricom *f* dimenzija *(N, M)* glasi:

$$F[k,l] = \mathcal{F}\{f\}[k,l] \triangleq$$

$$\triangleq \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m] e^{-\frac{2i\pi lm}{M}} e^{-\frac{2i\pi kn}{N}}$$

$$= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m] W_M^{lm} W_N^{kn}$$

$$= \sum_{n=0}^{N-1}\left(\sum_{m=0}^{M-1} f[n,m] W_M^{lm}\right) W_N^{kn}$$

Data jednačina pokazuje kako se pomoću jednodimenzione diskretne Furijeove transformacije može dobiti dvodimenziona Furijeova transformacija. Prvo se izvrši Furijeova transformacija nad svim kolonama, a potom po svim vrstama, ili obratno, da bi se dobila željena dvodimenziona transformacija. Ova osobina je značajna, jer pojednostavljuje implementaciju algoritma, i može se primeniti nad beskonačno dimenzija.

Inverzna Furijeova u dve dimenzije se može realizovati izvšavanjem dvodimenzione Furijeove transformacije nad konjugovanom matricom:

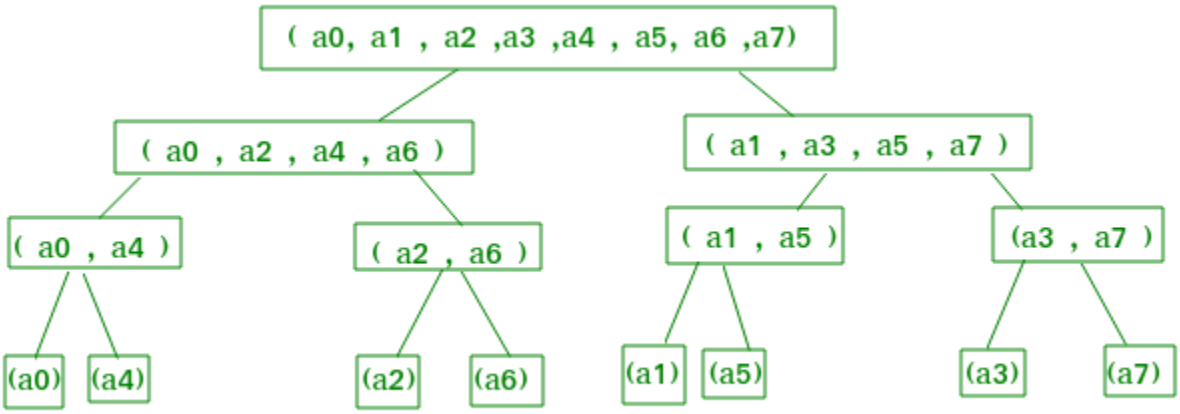
$$f[n,m] = \mathcal{F}^{-1}\{F\}[n,m] = (\mathcal{F}\{F^C\}[n,m])^C$$

3.2.5. Brza Furijeova transformacija

Postoji veliki broj algoritama koji smanjuju vreme izvršavanja jednodimenzione Furijeove transformacije sa $\mathcal{O}(N^2)$ na $\mathcal{O}(N \log N)$, kao što su Kuli-Tukijev algoritam („Radix 2“), Prajm faktor algoritam, Raderov algoritam, Vinogradov algoritam, i drugi. U ovom radu će biti implementiran „Radix 2“ algoritam u vremenskom domenu, pošto predstavlja najstariji i najpoznatiji algoritam za brzu Furijeovu transformaciju.

„Radix 2“ postiže ubrzanje u odnosu na diskretnu Furijeovu transformaciju izbegavanjem ponovnog računanja određenih izraza. Pripada klasi „zavadi pa vladaj“ algoritama, i kao preduslov neophodno

je da broj članova niza bude stepen dvojke. Tokom preprocesiranja, izmenimo redosled članova niza pomoću algoritma „bit reversal“. Potom. rekurzivnim ponavljanjem, podelimo članove na parne i neparne, izračunamo brzu Furijeovu transformaciju za oba novodobijena niza i spojimo rezultate, koristeći osobine korena jedinice (rotacionih faktora) u polju kompleksnih brojeva.



U narednih nekoliko koraka će biti izvedene formule za „Radix 2“ algoritam brze Furijeove transformacije:

$$F[k] = \mathcal{F}\{f\}[k] = \sum_{n=0}^{N-1} f[n] W_N^{nk} =$$

$$\sum_{n=0}^{\frac{N}{2}-1} f[2n] W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} f[2n+1] W_N^{(2n+1)k} =$$

$$\sum_{n=0}^{\frac{N}{2}-1} f[2n] W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} f[2n+1] W_N^{2nk}$$

$$k \in [0, \dots, N-1]$$

Na osnovu osobine redundantnosti, a potom i periodičnosti rotacionih faktora, formula za Furijeovu transformaciju postaje:

$$F[k] = \sum_{n=0}^{\frac{N}{2}-1} f[2n] W_{N/2}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} f[2n+1] W_{N/2}^{nk} = F_p[k] + W_N^k F_n[k]$$

$$k \in [0, \dots, \frac{N}{2} - 1]$$

Na osnovu osobine kompleksno konjugovane simetričnosti, redundantnosti i periodičnosti, može se jednostavno, na osnovu već izračunatih vrednost, dobiti vrednost člana F[k+N/2]:

$$F\left[k+\frac{N}{2}\right] = F_p\left[k+\frac{N}{2}\right] + W_N^{k+\frac{N}{2}} F_n\left[k+\frac{N}{2}\right] = F_p[k] - W_N^k F_n[k]$$

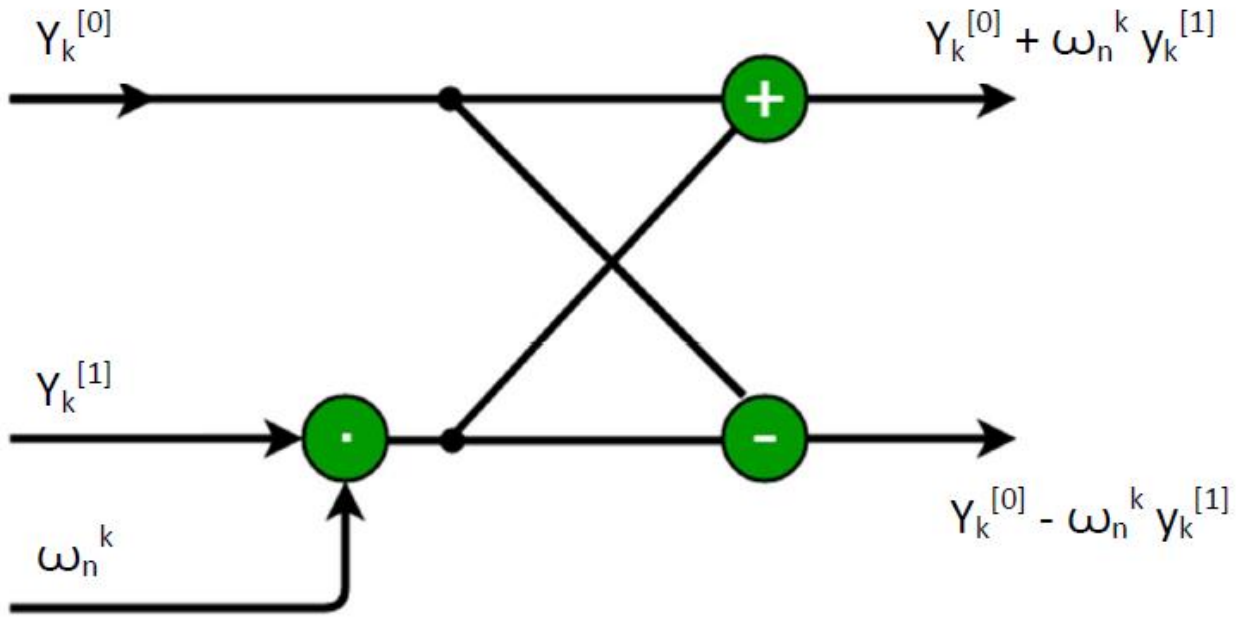
$$k \in [0, \dots, \frac{N}{2} - 1]$$

Iz poslednjih jednačina možemo primetiti rekurzivnu prirodu Furijeove transformacije i konačan izgled „Radix 2“ algoritma:

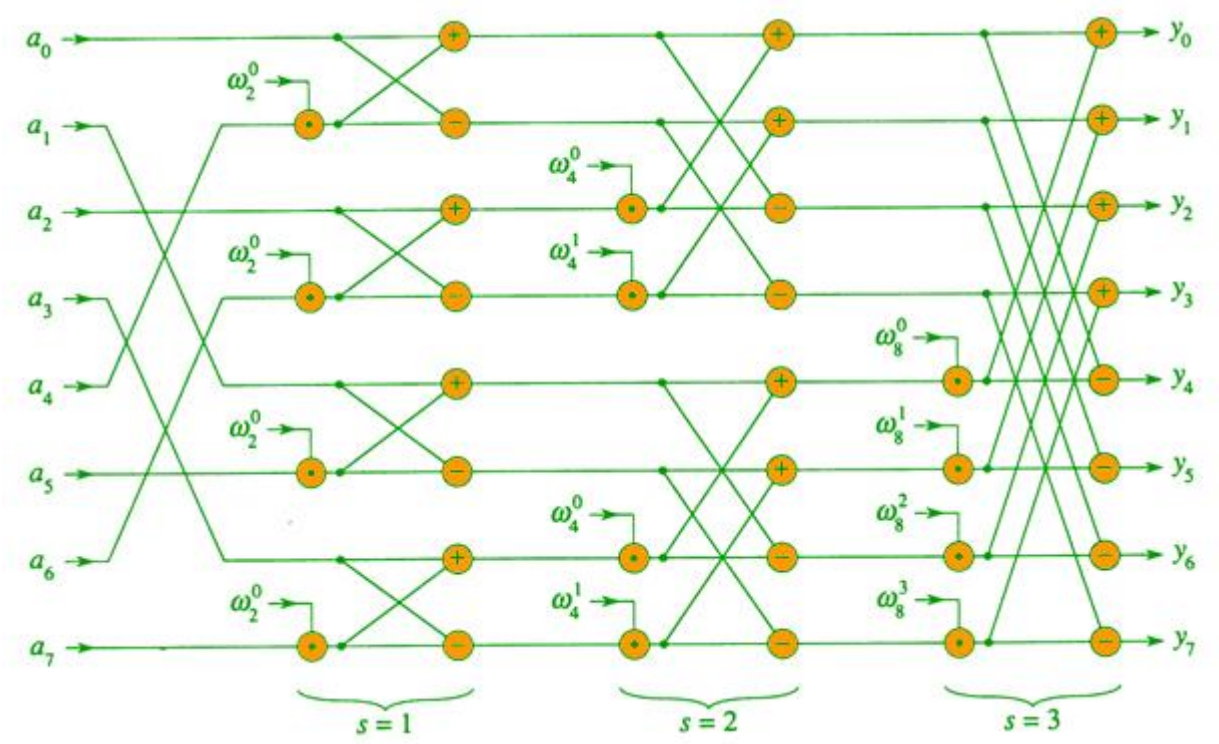
$$F[k] = F_p[k] + W_N^k F_n[k]$$

$$F[k+N/2] = F_p[k] - W_N^k F_n[k]$$

One nam omogućavaju da značajno uštedimo broj upotrebljenih operacija. Operacije računanja vrednosti F[k] i F[k+N/2], odnosno množenje neparnog člana sa rotacionim faktorom, sabiranje i oduzimanje sa parnim članom se naziva leptir (eng. „butterfly“) operacija.



Na slici ispod je prikazan postupak dobijanja Furijeove transformacije niza sa osam članova pomoću „Radix 2“ algoritma.



3.2.6. Odnos konvolucije i Furijeove transformacije

Neka su f i g dva niza veličine N i M, redom, i neka je $P \geq N \geq M$ moduo kružne konvolucije. Tada formula koja spaja kružnu konvoluciju sa diskretnom Furijeovom transformacijom glasi:

$$(f *_P g)[k] = \mathcal{F}^{-1}\{\mathcal{F}\{f_P\} \cdot \mathcal{F}\{g_P\}\}[k]$$

Formula se može dokazati računanjem Furijeove transformacije kružne konvolucije signala f i g.

$$\mathcal{F}\{(f *_P g)\}[k] = \sum_{i=0}^{P-1}\left((f *_P g)[i]W_P^{ki}\right) = \sum_{i=0}^{P-1}\sum_{j=0}^{P-1} f_P[j] \cdot g_P[i-j]W_P^{ki}$$

$$= \sum_{j=0}^{P-1} f_P[j]W_P^{kj} \sum_{i=0}^{P-1} g_P[i-j]W_P^{k(i-j)}$$

$$= \sum_{j=0}^{P-1} f_P[j]W_P^{kj} \sum_{i=0}^{P-1} g_P[i]W_P^{k(i)}, \qquad (zbog\ periodičnosti\ g_P)$$

$$= \mathcal{F}\{f_P\}[k] \cdot \mathcal{F}\{g_P\}[k]$$

Povezivanjem date formule sa odnosom između potpune linearne i kružne konvolucije, dobijemo formulu za računanje potpune linearne konvolucije korišćenjem Furijeovih transformacija:

$$h[k] = \sum_{i=\max(0,k-M+1)}^{\min(N-1,k)} f[i] \cdot g[k-i] = (f *_P g)[k] = \mathcal{F}^{-1}\{\mathcal{F}\{f_P\}\mathcal{F}\{g_P\}\}[k],$$

$$\forall k \in [0, N+M-2]$$

3.3. Računanje matrice normalne raspodele i zamućivanje slike

Matrica normalne raspodele, ili Gausova matrica zamućenja *(Gaussian blur matrix)*, je kernel koji koji konvolucijom zamućuje sliku. Služi da ublaži šum prisutan na slici. Koeficijenti matrice se računaju na osnovu dvodimenzionalne raspodele Gausove statistike:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Matrica normalne raspodele dimenzija 5x5 je prikazana ispod.

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

3.4. Računanje gradijenata

Sobel operatori *(matrice prikazane na slici ispod)*, pri konvoluciji sa slikom, služe za aproksimativno računanje diferencijala po horizontalnoj osi G_x, i vertikalnoj osi G_y.

$$\mathbf{Sx} = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad \mathbf{Sy} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Umesto Sobel operatora, mogu se koristiti Previtov kompas ili Šarov operator. Amplituda G i orijentacija Θ gradijenata se računaju na osnovu sledećih formula:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}} \qquad \mathbf{\Theta} = \operatorname{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

3.5. Odabiranje tačaka iznad gornjeg praga

Ulazni parametri algoritma su dva realna broja, koji predstavljaju gornji i donji prag amplituda gradijenata. Svi pikseli čija je amplituda gradijenta veća od vrednosti gornjeg praga su sigurno delovi ivica, i stoga ih automatski menjamo u maksimalnu vrednost 255, dok pikseli čija se amplituda gradijenta nalazi između gonjeg i donjeg praga mogu, ali ne moraju da budu deo ivica, za sada ih

menjamo u bilo koju predefinisanu vrednost. Odabir piksela iz ovog opsega se vrši u sledećem bloku. Ostale vrednosti nam nisu od interesa, pošto ne mogu biti delovi ivica, i svodimo ih na nulu.

3.6. Odabiranje tačaka iznad donjeg praga

Razlog za postojanje dva praga je uzrokovan pojavom kraćih prekida na ivicama. Rešenje sa dva praga značajno olakšava izbor vrednost da bi se problem otklonio, pošto je teško savršeno odrediti vrednost jedinstvenog praga u opštem slučaju, a često i nije moguće ako su delovi slike različito osvetljeni.

Za svaku tačku, odnosno piksel koji se nalazi iznad donjeg praga, proveravamo da li postoji piksel iznad gornjeg praga u neposrednoj blizini *(dodiruju se)*, i da li su im orijentacije iste. Ukoliko su prethodni uslovi zadovoljeni, vrednost te tačke menjamo u 255. Ovaj postupak se ponavlja sve dok ne bude bude ni jednog piksela promenjenog u iteraciji. Posle toga, sve vrednosti svih preostalih piksela, koji ne pripadaju ivicama, menjamo u nulu.

4. Zaključak

Uspešno je realizovan algoritam definisan projektnim zadatkom. Na slici ispod se nalazi rezultat obrade slike Lene sa napisanim algoritmom. Vreme obrade slike je i dalje veoma dugo, odnosno napisani algoritam se ne može koristiti za obradu snimka u realnom vremenu. Dodatno ubrzanje se može postići optimizovanjem algoritma za odabiranje tačaka iznad gornjeg i donjeg praga, korišćenjem struktura podataka kao što je graf i algoritma za pretragu u širinu, čime će se izbeći višestruko proveravanje vrednosti datog piksela.

Trenutna verzija operativnog sistema „Rasbian Stretch“, koji je specifično razvijen za Rasberi Paj, ne podržava Pajton verziju veću od 3.4, što predstavlja veliki problem za implementirani algoritam, jer se ne može uspostaviti komunikacija između Pajton i C++ kodova pomoću deljenih biblioteka. Rešenje prestavlja korišćenje „Arch“ operativnog sistema, dok „Rasbian Stretch“ ne podrži novije verzije Pajtona.

5. Literatura

[1] https://en.wikipedia.org/wiki/Edge_detection

[2] https://en.wikipedia.org/wiki/Canny_edge_detector

[3] Efficient convolution using the Fast Fourier Transform,
Jeremy Fix, 2011

[4] https://www.geeksforgeeks.org/converting-a-real-number-between-0-and-1-to-binary-string/

[5] http://www.aishack.in/tutorials/implementing-canny-edges-scratch/

[6] https://www.geeksforgeeks.org/iterative-fast-fourier-transformation-polynomial-multiplication/, pogledano dana 19.9.2019.

[7] http://alwayslearn.com/DFT%20and%20FFT%20Tutorial/DFTandFFT_FFT_TwiddleFactor.html, pogledano dana 19.9.2019.