VEŽBA 2 – 2D konvolucija

Potrebno predznanje

- Poznavanje programskog jezika C
- 2D signali
- RGB i YUV prostori boja i konverzija između ova dva prostora
- Diskretna konvolucija
- Filtriranje 1D signala u vremenskom domenu

Šta će biti naučeno tokom izrade vežbe

- Implementacija 2D diskretne konvolucije kao jedne od najčešće korišćenih operacija u obradi 2D signala
- Primena 2D diskretne konvolucije
- Osnovami filtriranja 2D signala
- Kakav je uticaj uklanjanja niskih i visokih frekvencija iz signala slike

Motivacija

Kod linearnih vremenski invarijantnih sistema vrednost izlaznih odbiraka signala računa se operacijom diskretne konvolucije ulaznog signala I impulsnog odziva sistema. Za 2D signale definisana je dvodimenziona diskretn konvolucija. Ova operacija predstavlja osnov za dalje izučavanje algoritama I sistema za obradu 2D signala.

Teorijske osnove

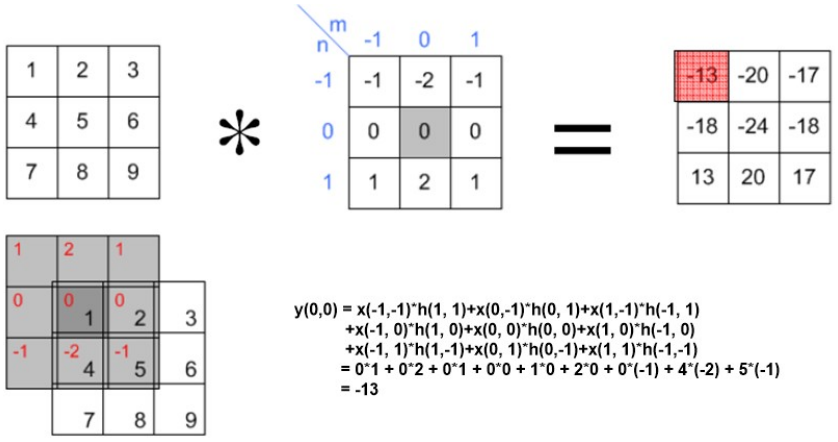
Diskretna 2D kovolucija

Kod 1D signala linearni vremenski invarijantni sistemi bili su predstavljeni impulsnim odzivom sistema *h(t)*. Obrada signala u okviru ovih sistema odgovara operaciji diskretne konvolucije između signala i impulsnog odziva sistema. Analogno tome u domenu 2D signala ovakvi sistemi predstavljeni su dvodimenzionalnim impulsnim odzivom *h(n,k)* veličine NxK. Obrada signala u okviru ovakvih sistema u prostornom domenu svodi se na operaciju dvodimenzionalne diskretne konvolucije. 2D konvolucija data je jednačinom:

$$y(v,h) = \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} h(n,k) \cdot x(v-n, h-k)$$

2D diskretna konvolucija je linearna kombinacija N x K okolnih tačaka pri čemu su težinski koeficijenti definisani sa N x K vrednosti impulsnog odziva h(n,k).

Prikaz računanja 2D diskretne konvolucije dat je na slici 1.



Matrica koja predstavlja impulsni odziv sistema se napiše unazad po X i Y osi, i pozicionira tako da se sredina matrice nalazi na koordinatama piksela čija se vrednost računa. Potom se vrši množenje svakog elementa matrice sa elementom slike koji se nalazi ispod njega i vrednosti se akumuliraju.

Tokom računanja kovolucije potrebno je voditi računa o prekoračenju opsega. Ukoliko rezultat konvolucije prelazi maksimalnu vrednost (npr. 255 za Y komponentu kod YUV formata) potrebno ga je ograničiti na tu vrednost, isti slučaj je i sa minimalnom vrednošću (u slučaju vrednosti manje od 0 ograničiti na 0).

Filtriranje slike

Filtriranje slike svodi se na operaciju diskretne konvolucije. Kernel filtra ujedno predstavlja i impulsni odziv. Veličina kernela mora biti neparna, odnosno kernel mora imati središnji element. Ukoliko je suma svih koeficijenata filtra jednaka 1, nakon filtriranja slika će sačuvati osvetljaj. Ukoliko je suma manja od 1, rezultujuća slika je tamnija od ulazne, a ukoliko je veća od 1 rezultujuća slika je svetlija.

S obzirom da slika ima konačne dimenzije, potrebno je pre filtriranja proširiti ivice slike za polovinu veličine kernela filtra sa svake strane, kako ne bi došlo do čitanja van opsega kada se vrši filtriranje ivičnih tačaka. Dodati pikseli se popunjavaju nulama ili se vrši kopiranje ivičnih vrednosti originalne slike. Drugi način rešenja ovog problema jeste da se u samoj implementaciji operacije konvolucije vodi računa o pristupanju vrednostima tačaka van slike. Kao i kod kopiranja vrednosti koje su logički van slike mogu da se zanemare (podrazumeva se da im je vrednost 0) ili da se vrši računanje sa ivičnim vrednostima.

Programski jezik C ne sadrži mogućnost proširenja već zauzetog memorijskog niza, te da bi se izvršilo proširenje slike neophodno je zauzeti novu sliku odgovarajućih dimenzija. Dimenzije nove slike predstavljaju zbir odgovarajuće dimenzije ulazne slike i vrednosti odgovarajuće dimenzije kernela umanjene za 1. Za ulaznu sliku dimenzija *xSize* i *ySize* i filtera veličine *NxN,* veličina proširene slike je:

```
int newXSize = xSize + (N - 1);
int newYSize = ySize + (N - 1);
```

Nakon zauzimanja memorije za proširenu sliku, neophodno je prekopirati sadržaj originalne slike u nju, na odgovarajuću poziciju:

```
int delta = (N - 1) / 2;
for (int i = 0; i <xSize; i++)
{
for (int j = 0; j <ySize; j++)
{
output[(j + delta) * newXSize + i + delta] = input[j * xSize + i];
}
}
```

Potom je neophodno popuniti preostale piksele u proširenoj slici. Dat je primer popunjavanja proširenih ivica kopiranjem ivičnih vrednosti originalne slike. Prvo se vrši kopiranje sadržaja redova koji odgovaraju prvom i poslednjem redu originalne slike u prvih i poslednjih *delta* redova koristeći funkciju *memcpy.*

```
for (int i = 0; i <delta; i++)
{
memcpy(&output[i * newXSize + delta], &input[0], xSize);
memcpy(&output[(ySize + delta + i)*newXSize + delta], &input[(ySize - 1) * xSize],
xSize);
}
```

Potom se vrši kopiranje kolona koje odgovraju prvoj i poslednjoj koloni originalne slike u prvih i poslednjih *delta*kolona. S obzirom da se pikseli koji pripadaju istoj koloni ne nalaze na uzastopnim lokacijama u memoriji nije moguće koristiti funkciju memcpy. Neophodno je proći kroz čitavu kolonu i

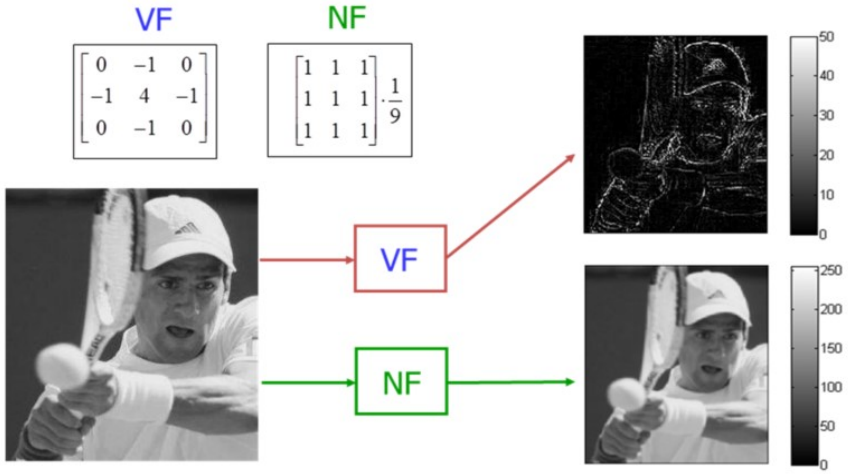
vrednost odgovarajućeg piksela prepisati na odgovarajućih *delta*lokacija, što se može uraditi upotrebom funkcije *memset*.

```
for (int i = 0; i < newYSize; i++)
{
memset(&output[i * newXSize], output[i * newXSize + delta], delta);
memset(&output[i * newXSize + delta + xSize], output[i * newXSize + xSize + delta - 1],
delta);
}
```

Nakon toga potrebno je izvršiti samo računanje dvodimenzione diskretne kovnolucije. Konvoluciju je potrebno računati samo za piksele slike koji s pripadali originalnoj slici. Rezultat je slika čije su dimenzije jednake originalnoj slici, pre proširenja ivica (u datom primeru xSize * ySize).

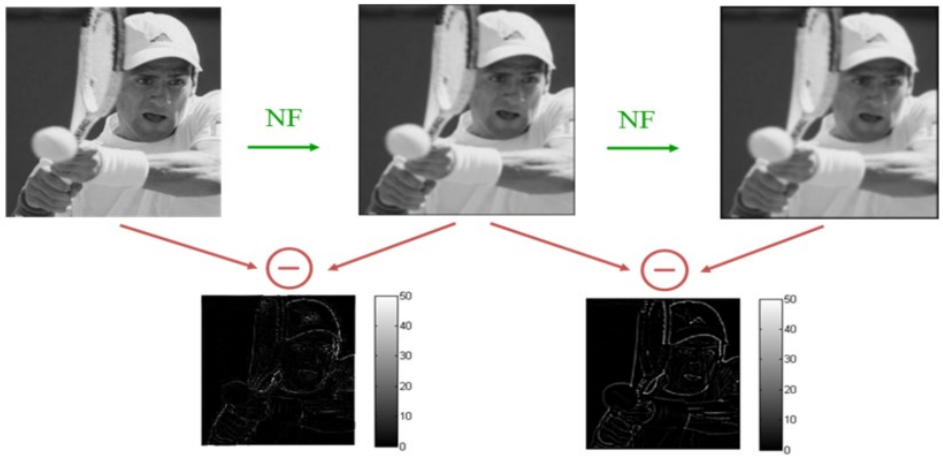
```
    for(int i = 0; i <xSize; i++)
{
        for (int j = 0; j <ySize; j++)
{
            double accum = 0;
            for (int m = 0; m <N; m++)
{
                for (int n = 0; n <N; n++)
{
accum += extendedImage[(j + n) * newXSize + i + m] * filterCoeff[(N - n) * N - m - 1];
}
}

            if (accum > 255.0)
image[j * xSize + i] = 255;
            elseif (accum < 0.0)
image[j * xSize + i] = 0;
            else
image[j * xSize + i] = floor(accum+0.5);
}
}
```

Na slici ispod dat je primer kernela filtera propusnika niskih frekvencija (NF) i visokih frekvencija (VF) i rezultat filtriranja slike ovim filterima.



Slika 1 - Primer filtriranja slike

Moguće je izvršiti I sukcesivno filtriranje koristeći isti filter. Dat je primer sukcesivnog filtriranja NF filtrom sa prethodne slike. Nakon svake iteracije dobija se zamućenija slika, odnosno više komponenti visokih frekvencija je potisnuto iz slike.



Slika 2 - Sukcesivno filtriranje NF filterom

Zadaci

U okviru ove vežbe vršiće se obrada nad Y komponentom slike. Na samom početku funkcije obrade izvršena je konverzija iz RGB prostora u YUV, a nakon obrade iz YUV u RGB upotrebom funkcija realizovanih u prošloj vežbi.

Zadatak 1

Implementirati funkciju:

- `void convolve2D (uchar Y_buff[], int xSize, int ySize, doublefilterCoeff, int N)`

koja vrši operaciju diskretne konvolucije nad ulaznom slikom koristeći prosleđeni filtar. Parametri funkcije su:

- `Y_buff` – ulazna slika
- `xSize` – horizontalna dimenzija slike u pikselima
- `ySize` – vertikalna dimenzija slike u pikselima
- `filterCoeff` – koeficijenti filtra
- `N` – veličina filtra (koeficijenti filtra su veličine NxN)

Pre računanja konvolucije potrebno je voditi računa o pristupanju vrednostima koje su logički van slike. Ovaj problem potrebno je rešiti proširivanjem slike za odgovrajući broj piksela sa svake strane.

Zadatak 2

Implementirati funkciju:

- `Void performNFFilter (uchar input[], int xSize, int ySize)`

koja vrši filtriranje slike upotrebom datog kernela NF filtra na slici 1 koristeći diskretnu konvoluciju realizovanu u prvom zadatku.

Zadatak 3

Implementirati funkciju:

- `void performVFFilter (uchar input[], int xSize, int ySize)`

koja vrši filtriranje slike upotrebom datog kernela VF filtra na slici 1 koristeći diskretnu konvoluciju realizovanu u prvom zadatku.

Zadatak 4

Implementirati funkciju:

- `void performSuccessiveNFFilter (uchar input[], int xSize, int ySize, intstages)`

koja vrši sukcesivno filtriranje slike upotrebom datog kernela NF filtra na slici 1 koristeći diskretnu konvoluciju realizovanu u prvom zadatku. Broj filtriranja zadat je parametrom *stages.* Prilikom poziva funkcije kao parametar *stages* proslediti vrednost grafičke kontrole *params[0].*