VEŽBA 4 – 2D spektar i spektralana dekompozicija slike

Potrebno predznanje

- Poznavanje programskog jezika C
- Računske operacije nad 2d matricama
- 2D signali
- RGB i YUV prostori boja i konverzija izmedju ova dva prostora
- Kvantizacija

Šta će biti naučeno tokom izrade vežbe

- Predstava 2D signala u spektralnom domenu
- Na koji način se transformacije koje ste koristili nad 1D signalima vrše nad 2D signalima
- Upoznaćete se sa osobinama 2D DCT transformacije i na koji način se njene osobine primenjuju na rešavanje problema koji se tiču kompresije slike
- Kroz praktične primere upoznaćete se sa osnovnim blokovima enkodera 2D signala.

Motivacija

Kao i kod jednodimenzionalnih i dvodimenzionalne signale moguće je predstaviti i obrađivati u sprektralnom domenu. Analogno 1D signalima kod Spektar 2D diskretnih signala je takođe dvo-dimenzionalana funkcija dve prostorne frekvencije. Dve najčešće korišćene transformacije su: diskretna Furijeova transformacija (DFT) i diskretna kosinusna transformacija (DCT). 2D DCT predstavlja često korišćen alat u digitalnoj obradi slike, pogotovo u okviru algoritama koji se tiču kompresije signala slike sa gubicima.

Teorijske osnove

Spektar diskretnih 2D signala

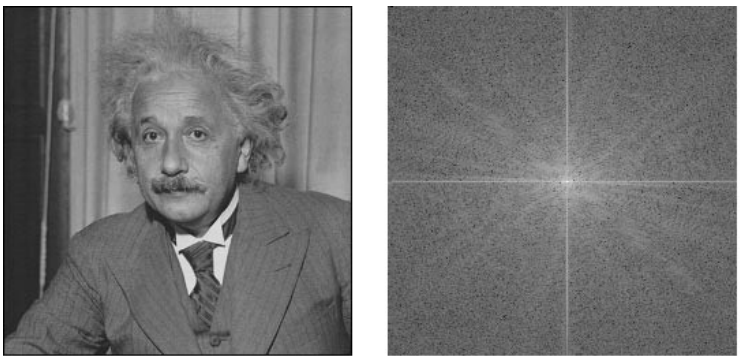
U toku izrade prethodne vežbe pokazano je na koji način su 2D signali predstavljeni. Za razliku od 1D signala koji su definisani kao funkcija vremena, 2D signali predstavaljaju funkciju dve prostorne koordinate. Dvodimenzione signale je poput jednodimenzionih moguće predstaviti i u frekventnom domenu. Spektar dvodimenzionih signala kao i spektar jednodimenzionih pokazuje od kojih se signale pored vremenskog moguće predstaviti i u frekventnom domenu, predstavlja od kojih se elementarnih signala sastoji posmatrani signal odnosno kako amplituda i faza tih signala učestvuju u njegovom formiranju. U slučaju 2D signala ti elementarni signali definisani su kao funkcije dve prostorne učestanosti.

Spektar diskretnih 2D signala izračunava se primenom jednodimenzionalne Furijeove transformacije na obe koordinate. Diskretna Furijeova transformacija dvodimentionalnih signala data je sledećom jednačinom:

$$S(n,k) = \sum_{v=0}^{N-1}\sum_{h=0}^{N-1} s(v,h)\cdot e^{-2j\pi\frac{n\cdot v+k\cdot h}{N}} \quad .$$

Kao što je kod jednodimenzionih signala DFT računata nad blokom od N odbiraka, u slučaju dvodimenzionalnih signala računa se nad blokom od N*N odbiraka i daje isto toliko spektralnih koeficijenata.

Primer jedne 2D slike i njenog spektra itračunatog upotrebom diskretne Furijeove transformacije date je na slici:



Slika 1 - Primer slike i njenog spektra

Inverzna transformacija definisana je sa

$$s(v,h) = \frac{1}{N\cdot N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1} S(n,k)\cdot e^{2j\pi\frac{n\cdot v+k\cdot h}{N}}$$

Direktna i inverzna 2D DFT se takođe mogu predstaviti u matričnoj formi. Ulazni signal s(v,h) je kvadratna matrica S dimenzija NxN a takođe i spektralni koeficijenti S(n,k) definišu jednu matricu dimenzije NxN. Koristeći već poznatu DFT matricu T takođe dimenzija NxN, direktna i inverzna transformacija se mogu predstaviti matrično:

$$S = T \cdot s \cdot T^T \qquad s = \frac{1}{N^2} T^{-1} \cdot S \cdot \left(T^{-1}\right)^T \qquad T(n,k) = e^{-2j\pi\frac{nk}{N}}$$

2D DCT (diskretna kosinusna transformacija)

Analogno DFT transformaciji i diskretna kosinusna transformacija je definisana za dvodimenzione signale. Kao i kod jednodimenzionih signala vrlo često se koristi u obradi zbog njene osobine da daje realne spektralne koeficijente u osnovnom opsegu učestanosti. Za definiciju 2D DCT je uopštena već poznata 1D DCT:

$$S(n) = \sum_{k=0}^{N-1} t(n,k) \cdot s(k) \qquad t(n,k) = \alpha(n) \cdot \cos\left(\pi \frac{n}{N}(k+1/2)\right) \qquad \alpha(n) = \begin{cases} \sqrt{1/N} & n = 0 \\ \sqrt{2/N} & n = 1,\ldots,N-1 \end{cases}$$

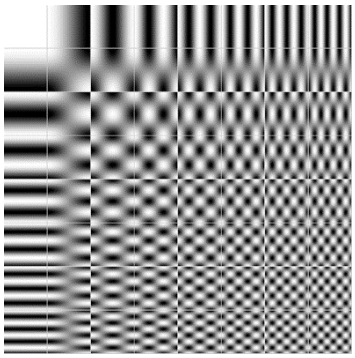
Kao i kod DFT 2D transformacija je realizovana primenom jednodimenzionalne transformacije na obe prostorne koordinate:

$$S(n,k) = \sum_{v=0}^{N=1} \sum_{h=0}^{N-1} t(n,v) \cdot t(k,h) \cdot s(v,h)$$

2D DCT za blok signala s(v,h) dimenzija NxN daje NxN spektralnih koeficijenata S(n,k) koji odgovaraju pod-opsezima 1/(2N) x 1/(2N) oko centralnih učestanosti. Inverzna 2D DCT je data sa:

$$s(v,h) = \sum_{n=0}^{N=1} \sum_{k=0}^{N-1} t(n,v) \cdot t(k,h) \cdot S(n,k)$$

Prikaz elemenetarnih signala na koje se signal razlaže prilikom računanja 2D DCT transformacije dat je na slici ispod.



Slika 2 - Elementarni signali 8x8 2D DCT transformacije

Kao i DFT direktna i inverzna 2D DCT se takođe mogu izraziti i u matričnoj formi:

$$S = T \cdot s \cdot T^T$$
$$s = T^{-1} \cdot S \cdot \left(T^{-1}\right)^T \qquad T(n,k) = t(n,k)$$

.

Bitna osobina 2D DCT transformacije jeste ortogonalnost iz koje sledi da je:

$$T^{-1} = T^T$$

Radi postizanja većeg ubrzanja obrade DCT se u praksi retko primenjuje nad čitavom slikom. Umesto toga vrši pre primene 2D DCT transformacije vrši se podela slike na blokove veličine NxN i računa se DCT transformacija nad tim blokova. Ovakav pristu omogućava i da matrica transformacije (T u prethodnoj jednačini) ne zavisi od veličine slike, pa je moguće unapred je izračunati i izračunate vrednosti koristiti prilikom računanja transformacije. Prilikom primene DCT po blokovima može doći do problema ukoliko veličina slike nije deljiva sa veličinom bloka (npr. rezolucija slike je 54*54 a veličina bloka 8x8). Ovaj problem se rešava proširirvanjem dimenzija slike do prve vrednosti deljive sa veličinom bloka. Proširenje se popunjava crnom bojom ili kopiranjem poslednjeg piksela slike pre proširenja.

Prikazan je primer računanja matrice transformacije T za zadatu večičinu bloka NxN, u programskom jeziku C, koristeći datu jednačinu za DCT. Za računanje vrednosti kvadratnog korena i kosinusa korišćene su funkcije iz standardnog zaglavlja *math.h.*

```
void GenerateDCTmatrix(doubleDCTKernel[], intN)
{
int i, j;
double alpha;
double denominator;
for (j = 0; j < N; j++)
{
DCTKernel[0, j] = sqrt(1 / (double)N);
}
alpha = sqrt(2 / (double)N);
denominator = (double)2 * N;
for (j = 0; j < N; j++)
{
for (i = 1; i < N; i++)
{
DCTKernel[i*N + j] = alpha * cos(((2 * j + 1) * i * PI) / denominator);
}
    }
}
```

Funkcija za računanje koeficijenata DCT transformacije se poziva jednom na početku izvršenja programa. Naknadno je potrebno pozivati samo u slučaju promene veličine bloka transformacije. Naredni korak kod računanja transformacije slike proizvoljne veličine vrši se podelom slike na blokove veličine NxN.Potrebno je iterirati kroz čitavu ulaznu sliku, po obe ose, sa korakom N i u svakoj iteraciji

blok veličine NxN počev od trenutne pozicije izdvojiti u privremenu matricu. Potom se nad izdvojenom podmatricom vrši poziv funkcije za računanje DCT.

```
    for (int y = 0; y < ySize; y+=N)
{
        for (int x = 0; x < xSize; x+=N)
        {
            for (int j=0; j<N; j++)
{
                for (int i=0; i<N; i++)
{
inBlock[j*N+i] = Y_buff2[(N*y+j)*(xSize2)+N*x+i];
}
}

DCT(inBlock, dctCoeffs, N, DCTKernel);
}
}
```

Nakon što su izračunati koeficijenti transformacije (*DCTKernel*) računanje same transformacije se svodi na množenje matrica. Ulazna matrica se množi sa koeficijentima transformacije, nakon čega se rezultat množi sa transponovanom matricom sa koeficijentima. Pre samog množenja, potrebno je ulazni blok (ukoliko je u pitanju komponenta za čiju predstavu se koriste neoznačeni brojevi) modifikovati tako da vrednosti budu centrirane oko 0, oduzimanjem broja koji odgovara sredini opsega od svake vrednosti.

```
void DCT(constucharinput[], shortoutput[], intN, doubleDCTKernel[])
{
double* temp = newdouble[N*N];
double sum;

for (int i = 0; i <N; i++)
    {
for (int j = 0; j <N; j++)
        {
            sum = 0;
for (int k = 0; k <N - 1; k++)
            {
                sum = sum + DCTKernel[i*N+k] * (input[k*N+j]-128.0);
            }
            temp[i*N + j] = sum;
        }
    }

for (int i = 0; i <= N - 1; i++)
    {
for (int j = 0; j <= N - 1; j++)
        {
            sum = 0;
for (int k = 0; k <= N - 1; k++)
            {
                sum = sum + temp[i*N+k] * DCTKernel[j*N+k];
            }
```

```
output[i*N+j] = floor(sum+0.5);
        }
}
delete[] temp;
}
```

2D DCT u kompresiji slike

Važna osobina 2D DCT, značajna u obradi 2D signala slike, a naročito iz aspekta kompresije jeste kompakcija energije. Naime, spektralna analiza 2D signala slike pokazuje da je najveći deo energije skoncentrisan na nižim učestanostima. Ustvari, sporadične manje komponente na višim učestanostima se javljaju samo zbog retkih skokova na ivicama objekata u slici. Sa obzirom na finu frekvencijsku rezoluciju 2D DCT, veliki deo energije 2D signala će biti samo uz manji broj koeficijenata BxB koji odgovaraju niskim učestanostima (n=0,1,...,B i k=0,1,...,B) imaće značajne vrednosti dok će drugi koeficijenti biti značajno manji.

Ova osobina i njena primenjivost u oblasti kompresije slike se jednostavno ilustruje na sledeći način. Nad ulaznom slikom izračuna se 2D DCT, zatim seanuliraju svi spektralni koeficijenti sem BxB najvećih spektralnih koeficijenata koji odgovaraju nisko-frekvencijskom opsegu (0 – B/2N) X (0 – B/2N).

Taj ekseriment se može opisati sledećom obradom:

$$
\begin{aligned}
&slika: && s \quad \text{dim}enzija \quad N \times N \\
&1.\quad 2D\ DCT: && S = T \cdot s \cdot T^T \\
&2.\quad NF: && S_{NF}(n,k) = \begin{cases} S(n,k) & n = 0,...,B\ k = 0,...,B \\ 0 & drugde \end{cases} \\
&3.\quad 2D\ IDCT: && s_{NF} = T^{-1} \cdot S_{NF} \cdot \left(T^{-1}\right)^T \\
&NF\ slika: && s_{NF} \quad \text{dim}enzija \quad N \times N
\end{aligned}
$$

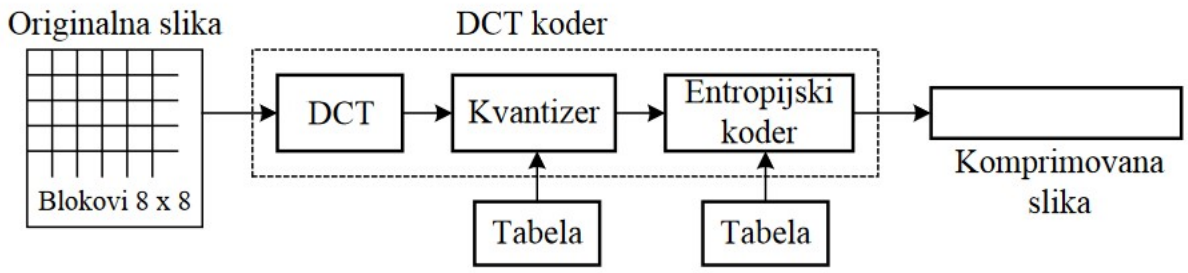
Rezultat eksperimenta dat je na slici:

Slika3 - Ulaznaslika, njen DCT spektar, slikanakonodbacivanja DCT koeficijenata

Originalna slika je dimenzije 512*512 piksela. U prvom slučaju (B=25) uzeto je 0.25 % NF spektralnih koeficijenata koji nose 97 % ukupne energije a u drugom slučaju (B=100) uzeto je 3.8 % NF koeficijenata koji nose 99 % ukupne energije. Iz prvog slučaja se vidi da je kvalitet slike nizak iako sadrži veliki deo ukupne energije.

Zbog ove osobine 2D DCT se koristi u mnogim algoritmima kompresije slike i video signala (JPEG, MJPEG, MPEG, DV i Theora). JPEG (*Joint Photographic Experts Group*) je standard koji se široko koristi za kompresiju mirne slike. Veoma je pogodan za kompresiju slika koje ne sadrže visoke frekvencije već sadrže blage prelaze. Koristi se za kompresiju monohromatskih slika ili slika u boji kod kojih nema mnogo naglih promena boje kao što su fotografije ili skenirane slike. JPEG nije pogodan za ikone, grafove, bitonalne slike i sl. JPEG omogućava stepen kompresije do 24:1 bez vidljivog gubitka kvaliteta.

Blok dijagram JPEG enkodera dat je na slici 4:



Slika4 - Koracikodkompresije 2D signalasagubicima

Dakle JPEG enkoder vrši DCT transformaciju, zatim kvantizaciju DCT koeficijenata i entropijsko kodovanje. Prilikom kvantizacije koristi se prethodno opisana osobina DCT transformacije koja se odnosi na kopaktovanje energije. Ova osobina se koristi tako što se primenjuje nelinearna kvantizacija, pri kojoj se NF koeficijenti (u kojima se nalazi koncentracija energije) kvantizuju sa većim brojem bita, a VF sa znazno manjim. JPEG standard propisuje tzv. kvantizacione tabele, koje sadrže faktore kvantizacije za različite veličine blokova. Data je kvantizaciona tabela za luminantnu komponentu slike i blok 8x8.

Tabela1 – Kvantizacionamatricakod JPEG kodera

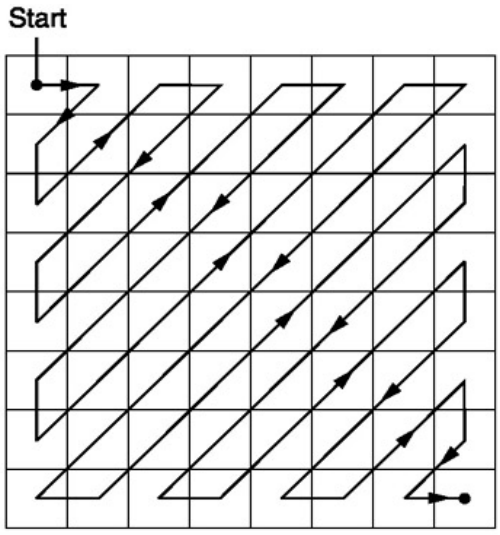
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Kvantizacija bloka koeficijenata veličine NxN vrši se tako što se svaki koeficijent podeli sa odgovarajućim faktorom kvantizacije, i potom se izvrši zaokruživanje rezultata kako bi se dobio ceo broj. Da bi se očuvao nivo osvetljaja u slici neophodno je kvantizovanu vrednost ponovo pomnožiti sa faktorom kvantizacije.

```
for (int j = 0; j < N; j++)
    {
for (int i = 0; i < N; i++)
        {
input[j*N + i] = round(input[j*N + i] / quantizationMatrix[j*N +
i])*quantizationMatrix[j*N + i];
        }
    }
```

Nakon kvantizacije koeficijenata, a pre primene entropijskog kodovanja potrebno je izvršiti konverziju 2D matrice u 1D niz. Prilikom konverzije moguće je iskoristiti dobro kompakciju energije koju imaju koeficijenti 2D DCT transformacije, kako bi se postigao što veći stepen kompresije. Primenom Zig-Zag transformacije prilikom transformacije matrice u niz, očuvava se koncentracija koeficijeanta koji sadrže najviše energije.

Zig-zag transformacija pordrazumeva razmotavanje 2D matrice u 1D niz tako što se čitanje matrice započne iz gornjeg levog ugla bloka (koeficijent sa koordinatama 0,0). Potom se vrši čitanje prvog koeficijenta sa desne strane i kreće dijagonalno u smeru dole levo. Kada se stigne do ivice bloka, pomera se za jedan piksel na dole i potom kreće dijagnoalno u smeru gore desno. Primer zig-zag transformacije za matricu veličine 8x8 dat je na slici 5.



Slika5 - Zig zag transformacijazablok 8x8

Zig-zag transformacija se može implementirati na više različitih načina. Jedan primer implementacije dat je u nastavku. Prikazana implementacija se svodi na iteraciju po dijagonalama kvadrata. Promenljiva *currDiagonalWidth* predstavlja dužinu dijagonale kroz koju se trenutno prolazi, i ima početnu vrednost 1 (dužina dijagonale u gornjem levom uglu). Prva petlja predstavlja prolazak kroz prvih N-1 dijagonala (odnosno dokle god je dužina dijagonale manja od N). U svakoj iteraciji petlje se iterira kroz elemente na dijagonali u smeru naviše ili naniže. Smer se određuje na osnovu parnosti indeksa dijagonale. Na kraju svake iteracije petlje, veličina dijagonale se povećava. Druga petlja predstavlja prolazak kroz preostalih N dijagonala, počevši od najduže. Na sličan način se iterira kroz

svaku od dijagonala u smeru naviše ili naniže. Na kraju svake iteracije petlje, dužina dijagonale se smanjuje.

```
        while (currDiagonalWidth<N)
        {
                for (i = 0; i<currDiagonalWidth; i++)
                {
                        if (currDiagonalWidth % 2 == 1)
                        {
                                row = currDiagonalWidth - i - 1;
                                col = i;
                        }
                        else
                        {
                                row = i;
                                col = currDiagonalWidth - i - 1;
                        }

// OBRADA PIKSELA SA KOORDINATAMA row I col
                }

                currDiagonalWidth++;
        }

        while (currDiagonalWidth> 0)
        {
                for (i = currDiagonalWidth; i> 0; i--)
                {
                        if (currDiagonalWidth % 2 == 1)                 {
                                row = N - currDiagonalWidth + i - 1;
                                col = N - i;
                        }
                        Else
                        {
                                row = N - i;
                                col = N - currDiagonalWidth + i - 1;
                        }

        // OBRADA PIKSELA SA KOORDINATAMA row I col
        }

                currDiagonalWidth--;
        }
```

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

U okviru funkcije:

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

implementirano je vršenje DCT transformacije nad ulaznom slikom. Parametri funkcije su:

- `intput` – ulazna slika
- `xSize` – horizontalna dimenzija slike u pikselima
- `ySize` – vertikalna dimenzija slike u pikselima
- `N` – veličina bloka DCT transformacije

U okviru funkcije izvršeno je prišširivanje slike, generisanje transformacione matrice za zadatu veličinu bloka, podela slike na blokove i poziv DCT transformacije nad blokovima. Vrednosti DCT koeficijenata su skalirani i smešteni u Y_buff[] kako bi bili iscrtani nakon obrade.

U okviru funkcije:

- `voidDCT(constucharinput[], shortoutput[], intN, double* DCTKernel)`

Implementirano je računanje DCT transformacije nad blokom NxN koristeći prosleđenu matricu sa transformacionim koeficijentima *DCTKernel*. Računanje se svodi na množenje ulazne matrice sa transformacionom matricom, i potom sa transponovanom transformacionom matricom kao što je opisano u teorijskim osnovama.

1. Pokrenuti program.
2. Postaviti vrednost veličine bloka 256 i prikazati rezultat obrade nad ulaznom slikom *lena.jpg.* Dimenzije ove slike su 256*256 piksela, pa ovakav poziv odgovara izvršenju DCT transformacije nad čitavom slikom.
3. Proveriti rezultate jedne i druge funkcije za veličine bloka 8, 16 i 32, za različite ulazne slike.

Zadatak 2

U okviru funkcije:

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

implementirano je vršenje DCT transformacije nad ulaznom slikom kao u prethodnoj zadatku a potom odmah i inverzna IDCT transformacija. Parametri funkcije su:

- `input` – ulazna slika
- `xSize` – horizontalna dimenzija slike u pikselima
- `ySize` – vertikalna dimenzija slike u pikselima
- `N` – veličina bloka DCT transformacije

Implementirati funkciju:

- `voidIDCT(constshortinput[], ucharoutput[], intN, double* DCTKernel)`

Koja vrši inverznu diskretnu kosinusnu transformaciju nad ulaznim vrednostima. Parametri funkcije su:

- `input` – ulazni blok DCT koeficijenata dimenzije NxN
- `output` – izlazni blok u koji je potrebno upisati vrednosti piksela slike, dimenzije NxN
- `N` – veličina bloka DCT transformacije
- `DCTKernel` – koeficijenti transformacije

Pokrenuti program. Očekivani rezultat funkcije `performDCTandIDCT`je da ulazna slika ostane nepromenjena.

Zadatak 3

Implementirati funkciju:

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

koja vrši poziv DCT i IDCT kao u prvom zadatku. Između poziva DCT i IDCT izvršiti kvantizaciju DCT koeficijanata upotrebom kvantizacione tabele date u okviru teorijskog uvoda.

Pre poziva funkcije zapamtiti kopiju ulazne slike (*Y_buff*), i nakon izvršene funkcije izračunati srednju kvadratnu grešku (MSE) po formuli:

$$MSE = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M * N}$$

Ispis vrednosti srednje kvadratne greške realizovati kao štampanje u QT konzolu za informacije (qInfo).

```
qInfo() << "MSE: " << MSE;
```

Zadatak 4

Implementirati funkciju:

- `void performMaskDCTCoeffs(char input[], int xSize, int ySize, int N, int B)`

koja vrši poziv DCT i IDCT transformacijepoputfunkcije`performDCTandIDCT`, ali između poziva ove dve transformacije nad blokom DCT koeficijenata veličine NxN izvršiti potrebno je odbaciti deo koeficijenata.

Odbacivanje koeficijenata izvršiti tako što ćete izvršiti konverziju matrice u niz primenom Zig-Zag transformacije i potom maskirati prvih**B** koeficijenata u dobijenom jednodimenzionom nizu (ostale koeficijente nulirati).