|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** |  |

Иван Лазаревић

**Имплементација алгоритама за интерполацију слике**

ИСПИТНИ РАД

- Основи алгоритама и структура ДСП 2 -

Ментор: Проф. др. Миодраг Темеринац

Нови Сад, 2016

**Sadržaj**

1. Uvod.........................................................................................................................................3

2. Teorijske osnove.....................................................................................................................3

3. Koncept rešenja.......................................................................................................................6

**Uvod**

Slike u digitalnom obliku imaju dosta ograničenja. Bez obzira na kvalitet uredjaja preko kojeg je došlo do digitalizacije, slike ima tačno ograničenu rezoluciju tj. veličinu ali i ograničenost prezentacije boja. Složenije slike tj. slike sa većom rezolucijom i većom količinom diskretnih elemenata kojima se predstavljaju boje sadrže velike količine podataka što može dovesti do problema kod same obrade ali i prenosa takvih podataka. Shodno potrebama ponekad je potrebno smanjiti ili povećati veličinu podataka koijma se neka slika reprezentuje. Da bi se došlo do kompromisa izmedju kvaliteta slike ali i količine podataka obradi slike se pridaje dosta značaja. Kod prenosa podataka je važno smanjiti količinu podataka koji se prenosi ali isto tako mora se imati u vidu da smanjenjem podataka dolaz i do gubitka kvaliteta slike. Isto tako ponekad postoji potreba da se slika manjeg kvaliteta prezentuje na uredjajima sa većom rezolucijom ili različitih proporcija te i tu interpolacija slike igra važnu ulogu.

Pored same reprezentacije slike preko količine podataka interpolacija slike se koristi i prilikom uveličavanja slike, raznih transformacija slike, popravljanja smetnje u slici, prepoznavanje oblika ili estimacija pokreta sa necelobrojnom tačnošću.

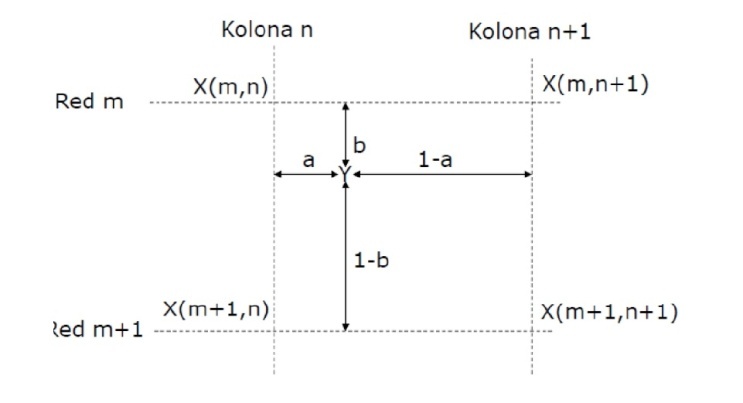
**Teorijske osnove**

Postoji potreba za obradom slike tako da se slika može uvećati ili smanjiti shodno potrebama uz očuvanje oštrine.

Interpolacija se svodi na odredjivanje nedostajućih podataka na osnovu već dostupnih. Postoje različite interpolacione tehnike koje se upotrebljavaju . Jedne od jednostavnijih tehnika su interpolacija ponavljanjem piksela (engl. Sample and Hold) i bilinearna interpolacija.

***-sample and hold algoritam***

Ovo je jednostavan algoritam koji za novu vrednost tacke uzima poznatu vrednost iz najbliže tacke u osnovnoj slici.

(Slika 1)

Kako se nova tačka nalazi unutar kvadrata koji je ograničen kolonama i redovima orignalne slike tako se shodno položaju izračunava koja je tačka osnovnog rastera najbliža novoj tački. Sve opcije možemo videti na sledećim formulama.

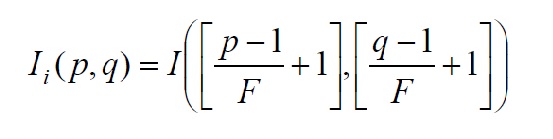
a < 0.5 b < 0 → Y = X(m, n)

a ≥ 0.5 b < 0 → Y = X(m, n+1)

a < 0.5 b ≥ 0 → Y = X(m+1, n)

a ≥ 0.5 b ≥ 0 → Y = X(m+1, n+1)

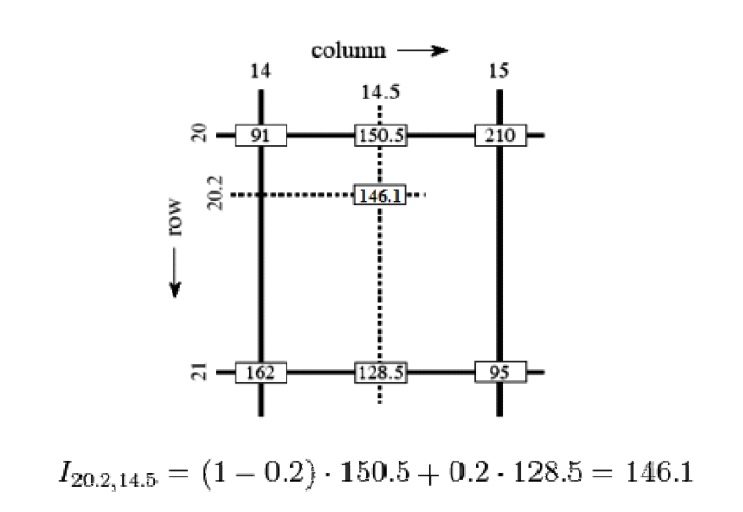
Algoritam se može jednostavnije izraziti sledećom formulom:



Glavna prednost ovog algoritma je u njegovoj jednostavnosti i brzini te time nezahteva velike procesorsku snagu. Nedostatak je blokovska tekstura slike.

***-bilinearna interpolacija***

Nešto složeniji algoritam od Sample and hold je Bilinearna interpolacija. Ovde se nove tačke računaju shodno udaljenosti od tačaka iz osnovnog rastera. Ovaj algoritam koristi 4 najbliže tačke iz osnovnog rastera slično Sample and hold algoritmu. Za razliku od tog algoritma svaka tačka ce shodno razdaljini od sve 4 obližne tačke imati drugačiju vrednost od svojih suseda čime se pegla blokovksi izgled koji smo dobili kao rezultat prethodnog algoritma. Možemo na slici 2. videti primer upotrebe ovog algoritma.



Slika 2. Primer bilinearne interpolacije

Bilinearna interpolacija se zasniva na sledećoj formuli:



Parametri m i n su horizontalni/vertikalni indeksi, dok se a i b mogu izračunati prema sledećim formulama:

a = ms / Sh – floor(ms / Sh) b = ns / Sv – floor(ns / Sv)

Gde su:

* ms/ns – horizontalni/vertikalni indks u skaliranoj slici
* Sh/Sv – horizontalni/vertikalni factor skaliranja

Složeniji algoritmi interpolacije postižu bolji kvalitet ali po ceni veće upotrebe procesorskih resursa.

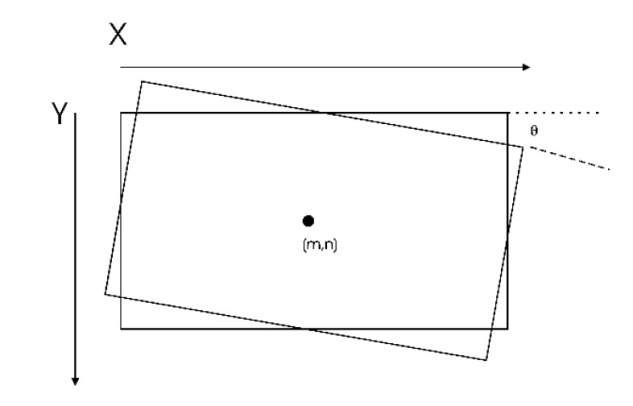
## *-rotacija slike*

Još jedna upotreba interpolacionih algoritama jeste kod rotacije slike. Rotacija slike sa centrom rotacije u gornjem levom uglu (kordinate x(0,0) ) sa željenim uglom θ se može realizovati preko sledećih formula:

*X’=X\**cos(θ) +*Y*\*sin*(*θ*)*

*Y’=Y\**cos(θ) -*X*\*sin*(*θ*)*

Nakon izračunavanja *X’* i *Y’,* u rezultujuću sliku na kordinate X i Y treba upisati vrednosti piksela ulazne slike sa koordinatama *X’* i *Y’.* S obzirom da dotične kordinate nisu celi brojevi potrebno je upotrebiti jedan od algoritama interpolacije.



Slika 3. Rotacija

Rotacija slike oko proizvoljne ose se vrši upotrebom sledećih formula:

*X’ = X \** cos(θ) – *Y \** sin(θ) – m \* cos(θ) + n \* sin(θ) + m

*Y’ = Y \** cos(θ) + *X \** sin(θ) – m \* sin(θ) - n \* cos(θ) + n

Gde m i n predstavljaju kordinate tačke oko koje će se slika rotirati.

**Koncept rešenja**

Potrebno je realizovati sistem za promenu rezolucije i rotiranje slike uz upotrebu dva osnovna algoritma interplacije sample and hold i bilinearne interpolacije.

## *-realizacija sample and hold agoritma*

Realizacija sample and hold algoritma ce se raditi preko funkcije:

**void** *sampleAndHold*(**uchar** input[], **int** xSize, **int** ySize, **uchar** output[], **int** newXSize, **int** newYSize);

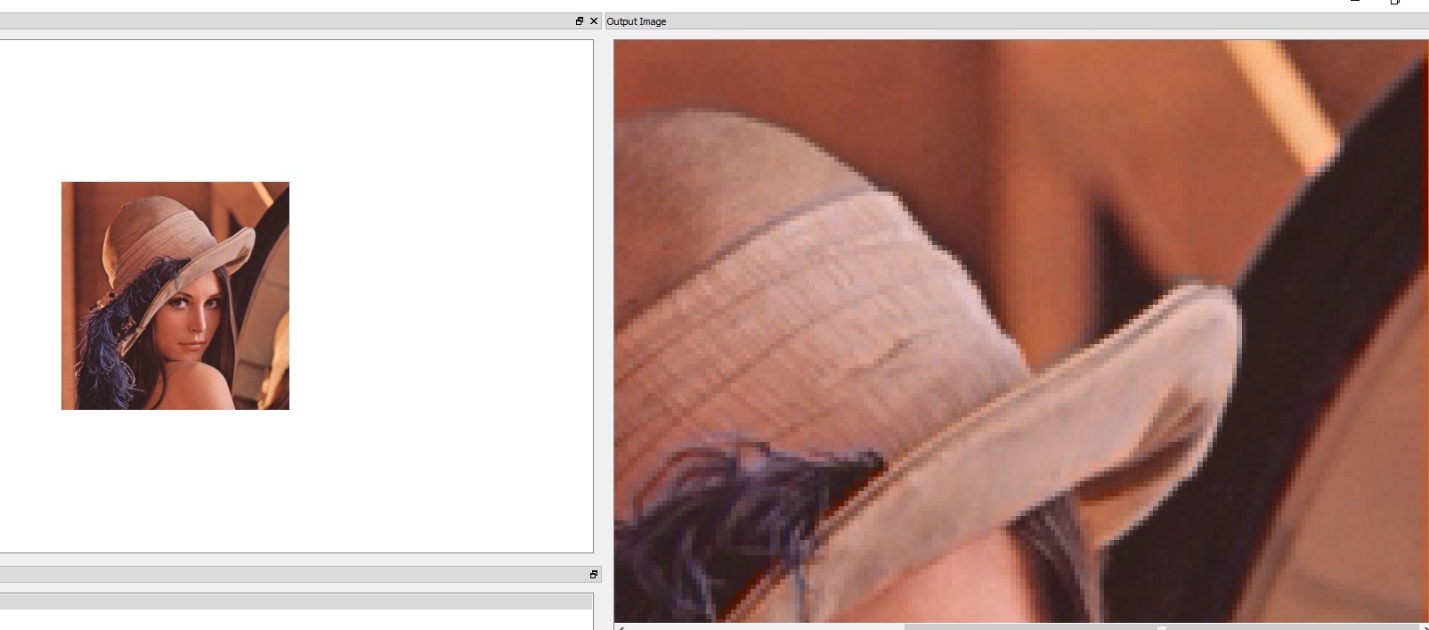
Parametri funkcije su:

* Input – ulazna slika u RGB formatu
* xSize – horizontalna dimenzija ulazne slike u pikselima
* zSize – vertikalna dimenzija ulazne slike u pikselima
* output – izlazna slika u RGB formatu
* newXSize – horizontalna dimenzija izlazne slike u pikselima
* newYSize – vertikalna dimenzija izlazne slike u pikselima

Pre nego što se pozove sama funkcija treba izračunati rezoluciju izlazne slike preko faktora skaliranja i pri tome obezbediti da je nova rezolucija deljiva sa 4 zbog upotrebe YUV420 formata. Faktor skaliranja po vertikali je u params[0] a po horizontali u params[1]. Potrebno je alocirati memoriju za izlaznu sliku.

U samoj funkciji se alociraju po dva bufera za Y, U i V komponentu koji će služiti za konverziju iz RGB u YUV420 format, obradu/interpolaciju u YUV420 formatu i konverziju nazad u RGB format kao izlazni rezultat.

Na osnovu već objašnjenog postupka za sample and hold algoritam nalazi se najbliža tačka iz starog rastera i ta vrednost se upisuje u novu tačku u novom rasteru. To se ponavlja sa sva tri bufera Y, U i V imajući u vidu da su U i V buferi 4 puta manji od Y komponente.



***-realizacija bilinearne interpolacije***

Realizacija bilinearne interpolacije će se raditi preko funkcije:

**void** *bilinearInterpolate*(**uchar** input[], **int** xSize, **int** ySize, **uchar** output[],**int** newXSize,**int** newYSize);

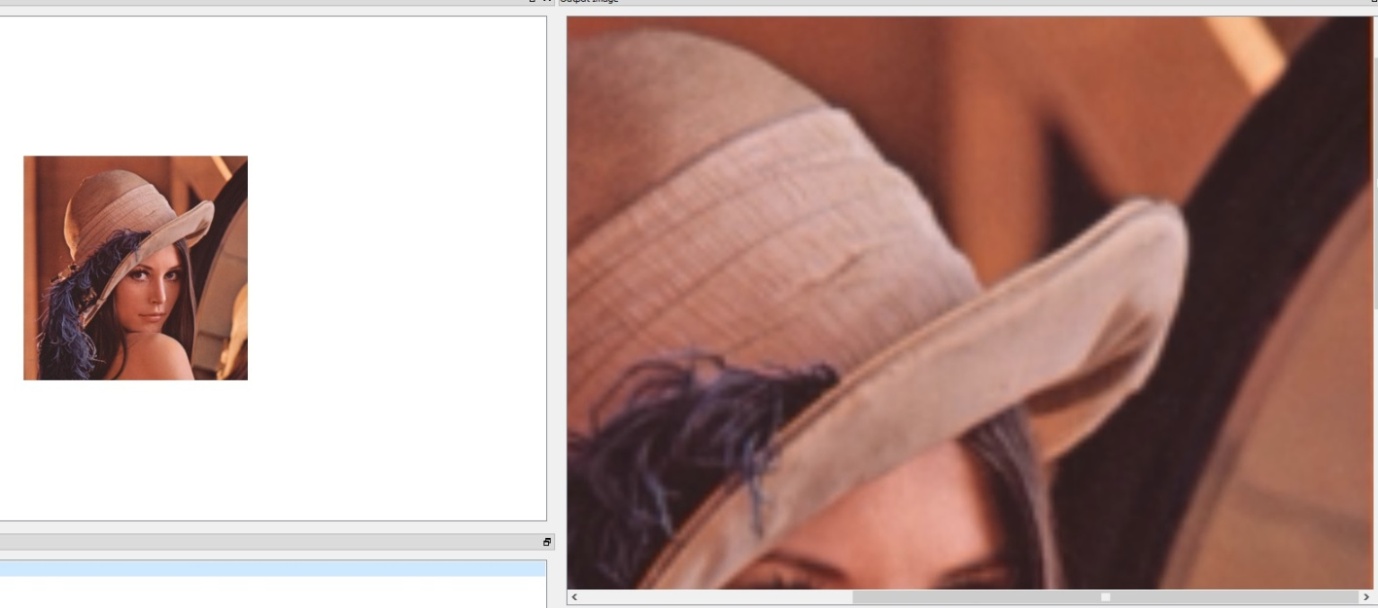
Parametri funkcije su:

* Input – ulazna slika u RGB formatu
* xSize – horizontalna dimenzija ulazne slike u pikselima
* zSize – vertikalna dimenzija ulazne slike u pikselima
* output – izlazna slika u RGB formatu
* newXSize – horizontalna dimenzija izlazne slike u pikselima
* newYSize – vertikalna dimenzija izlazne slike u pikselima

Pre nego što se pozove sama funkcija treba izračunati rezoluciju izlazne slike preko faktora skaliranja i pri tome obezbediti da je nova rezolucija deljiva sa 4 zbog upotrebe YUV420 formata. Faktor skaliranja po vertikali je u params[0] a po horizontali u params[1]. Potrebno je alocirati memoriju za izlaznu sliku.

U samoj funkciji se alociraju po dva bufera za Y, U i V komponentu koji će služiti za konverziju iz RGB u YUV420 format, obradu/interpolaciju u YUV420 formatu i konverziju nazad u RGB format kao izlazni rezultat.

Na osnovu već objašnjenog postupka za algoritam bilinearne interpolacije nalazi se najbliže tačke iz starog rastera i uzimajući vrednosti sve četri tačke i rastojanja tačke u novom rasteru od te četri tačke nalazi se srednja vrednost koja se upisuje u tačku novog rastera. To se ponavlja sa sva tri bufera Y, U i V imajući u vidu da su U i V buferi 4 puta manji od Y komponente.



## *-rotacija slike uz upotrebu sample and hold algoritma*

Realizacija rotacije slike uz upotrebu sample and hold algoritma interpolacije će se raditi preko funkcije:

**void** imageRotate(**uchar** input[], **int** xSize, **int** ySize, **uchar** output[], **int** m, **int** n, **double** angle);

Parametri funkcije su:

* Input – ulazna slika u RGB formatu
* xSize – horizontalna dimenzija ulazne slike u pikselima
* zSize – vertikalna dimenzija ulazne slike u pikselima
* output – izlazna slika u RGB formatu
* m – horizontalna kordinata oko koje će se vršiti rotacija slike
* n – vertikalna koordinata oko koje će se vršiti rotacija slike
* angle – ugao za koji će se slika rotirati

Pre nego što se pozove sama funkcija treba obezbediti da je rezolucija deljiva sa 4 zbog upotrebe YUV420 formata. Ugao rotacije je u dat u params[0]. Potrebno je alocirati memoriju za izlaznu sliku.

U samoj funkciji se alociraju po dva bufera za Y, U i V komponentu koji će služiti za konverziju iz RGB u YUV420 format, obradu/interpolaciju u YUV420 formatu i konverziju nazad u RGB format kao izlazni rezultat.

Na osnovu već objašnjenog postupka za rotaciju slike za svaku tačku se preko formule odredi nova pozicija u novom rasteru i u nju se iz originalne slike, uzimajući u obzir sample and hold algoritam, kopira vrednost tačke iz starog rastera. To se ponavlja sa sva tri bufera Y, U i V imajući u vidu da su U i V buferi 4 puta manji od Y komponente.



## -rotacija slike uz upotrebu bilinear interpolacije

Realizacija rotacije slike uz upotrebu bilinearne interpolacije će se raditi preko funkcije:

**void** *imageRotateBilinear*(**uchar** input[], **int** xSize, **int** ySize, **uchar** output[], **int** m, **int** n, **double** angle);

Parametri funkcije su:

* Input – ulazna slika u RGB formatu
* xSize – horizontalna dimenzija ulazne slike u pikselima
* zSize – vertikalna dimenzija ulazne slike u pikselima
* output – izlazna slika u RGB formatu
* m – horizontalna kordinata oko koje će se vršiti rotacija slike
* n – vertikalna koordinata oko koje će se vršiti rotacija slike
* angle – ugao za koji će se slika rotirati

Pre nego što se pozove sama funkcija treba obezbediti da je rezolucija deljiva sa 4 zbog upotrebe YUV420 formata. Ugao rotacije je u dat u params[0]. Potrebno je alocirati memoriju za izlaznu sliku.

U samoj funkciji se alociraju po dva bufera za Y, U i V komponentu koji će služiti za konverziju iz RGB u YUV420 format, obradu/interpolaciju u YUV420 formatu i konverziju nazad u RGB format kao izlazni rezultat.

Na osnovu već objašnjenog postupka za rotaciju slike za svaku tačku se preko formule odredi nova pozicija u novom rasteru i u nju se iz originalne slike, uzimajući u obzir bilinearnu interpolaciju, uzeti vrednosti 4 tačke starog rastera preko kojih će se uz upotrebu formule za bilinearnu interpolaciju izračunati vradnost i upisati u tačku novog rastera. To se ponavlja sa sva tri bufera Y, U i V imajući u vidu da su U i V buferi 4 puta manji od Y komponente.

