Vežba 7 - Implementacija algoritama za interpolaciju slike

Potrebnopredznanje

- Poznavanje programskog jezika C
- 2D signali
- RGB i YUV prostori boja
- Filtriranje 2D signala u vremenskom domenu

Šta će bitinaučenotokomizradezadatka

Tokom izrade ovog zadatka upoznaćete se sa problemom promene rezolucije slike. Naučićete na koji način je moguće povećati rezoluciju slike i na koji način različiti algoritmi utiču na kvalitet povećane slike. Realizovaćete dva prosta, ali u praksi često primenljiva algoritma za povećanje rezolucije slike: *Sample and hold* algoritam i algoritam bilinearne interpolacije. Naučićete i na koji način se ovi algoritmi primenjuju na gometrijske transformacije slike kao što je rotacija oko proizvoljne tačke.

Motivacija

Digitalne slike i video zapisi sadrže veliki broj podataka, čiji obim raste sa napretkom tehnike, što dovodi do prepreka prilikom prenosa multimedijalnog sadržaja, koji mora biti prenet u odgovarajućem roku, kao i bez gubitka kvaliteta. Kako bi se zadovoljili zahtevi kvaliteta, brzine prenosa i prilagođenja prenetih podataka ciljnim sistemima, obradi slike se pridaje sve veći značaj. Ušteda propusnog opsega mreže može se postići kodovanjem slike niske rezolucije na strani enkodera, koja se onda, na strani dekodera, pre samog prikazivanja krajnjem korisniku, uvećava do rezolucije modernih panela.

Povećanje slike do željene visoke rezolucije vrši se nekom od tehnika interpolacije. Jedan od važnih primera rastuće potrebe za interpolacijom je i prikaz TV signala standardne definicije (SD TV) na savremenim panelima koji su mahom veće rezolucije a dosta često i različitih proporcija (prikaz standardnog 4:3 SD signala na 16:9 HD panelu).

Pored navedenih primena interpolacija slike se koristi prilikom uveličavanja slike (*zoom*), izvršenja geometrijskih transformacija slike (kao što je rotiranje), popravljanje-smetnji u slici (engl. *imageinpainting*) ili estimacija pokreta sa ne-celobrojnom tačnošcu.

TEORIJSKEOSNOVE

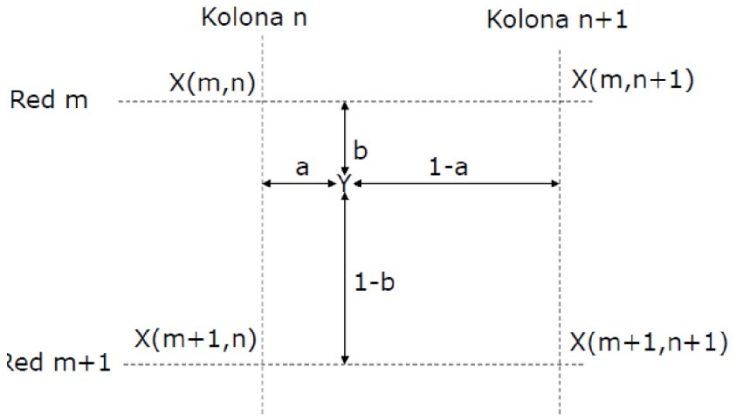
Cilj pojektnog zadatka je implementacija metoda za interpolaciju slike pri proizvoljnom faktoru uvećanja ili umanjenja, uz očuvanje oštrine. Ukratko , interpolacija se svodi na određivanje vrednosti i nedostajućih podataka, na osnovu onih već poznatih, i predstavlja vezu između diskretnog i kontinualnog domena. Postoje različite interpolacione tehnike. Neke od najprostijih tehnika interpolacije koje se vrlo često koriste u praksi zbog jednostavnosti računanja jesu metode koje koriste konstantne konvolucione kernele za celu sliku. U ove metode između ostalih ubrajaju se interpolacija ponavljanjem piksela (engl. *Sample and Hold*) i bilinearna interpolacija. Pored ovih tehnika postoje i različiti složeni algotitmikoji kombinuju pomenute metode sa dodatnim algoritmima obrade slike ili prikupljenim informacijama o slici (npr. ivicama).

1 *Sample and hold* algoritam

Ovo je najjednostavniji algoritam u kojem se za interpoliranu vrednost uzima poznata vrednost iz najbliže tačke u osnovnom rasteru:

$$a < 0.5 \;\; b < 0.5 \rightarrow Y = X(m, n)$$
$$a \geq 0.5 \;\; b < 0.5 \rightarrow Y = X(m, n+1)$$
$$a < 0.5 \;\; b \geq 0.5 \rightarrow Y = X(m+1, n)$$
$$a \geq 0.5 \;\; b \geq 0.5 \rightarrow Y = X(m+1, n+1)$$

Vrednosti parametara m, n, a i b u jednačini odgovaraju indeksima prikazanim na slici 1.



Slika 1 - Prikaz interpolacije

Algoritam se može izraziti i jednostavnije:

$$I_i(p,q) = I\left(\left\lfloor \frac{p - F/2}{F} + 1 \right\rfloor, \left\lfloor \frac{q - F/2}{F} + 1 \right\rfloor\right)$$

Na slici 2 je prikazana interpolacija sa SH algoritmom. Velika prednost ovog algoritma je jednostavnost (praktično nepotrebni procesorski resursi) a nedostatak je stepeničasta struktura ivica i neprirodno uniformisana tekstura slike (blokovska mustra).

Slika 2 - Rezultat interpolacije sa faktorom povećanja 4 koristeći SAH metodu

2 Bilinearna interpolacija

Bilinearna interpolacija je nešto kompleksniji algoritam gde se koeficijenti interpolacije računaju na osnovu udaljenosti tačaka iz osnovnog rastera od interpolacione tačke. Osnovna ideja bilinearne interpolacije je da se prvo izvede linearna interpolacija po jednoj dimenziji slike, a potom po drugoj. Za razliku od prethodno opisane tehnike bilinearna interpolacija koristi 4 najbliže vrednosti tačaka, locirane u dijagonalnim pravcima od trenutnog piksela. Bilinearna interpolacija koristi oblast 2x2 poznatih vrednosti piksela koji okružuju nepoznati piksel. Interpolacija se zasniva na usrednjavanju te 4 vrednosti po formuli sledećoj formuli:

$$Y = (1-a)(1-b)X(m,n)+(1-a)bX(m+1,n)+a(1-b)X(m,n+1)+abX(m+1,n+1)$$

Vrednosti parametara m, n, a i b u jednačini odgovaraju indeksima prikazanim na slici 1.

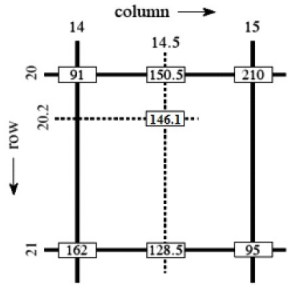
Vrednosti *a* i *b* se mogu izračunati po formuli:

$$a = \frac{n_s}{F_h} - \left\lfloor \frac{n_s}{F_h} \right\rfloor \qquad b = \frac{m_s}{F_v} - \left\lfloor \frac{m_s}{F_v} \right\rfloor$$

Gde su:

- n_S/m_S– horizontalni/vertikalni indeks piksela u skaliranoj slici (pozicija)
- *Fh/Fv* – horizontalni/vertikalni faktor skaliranja

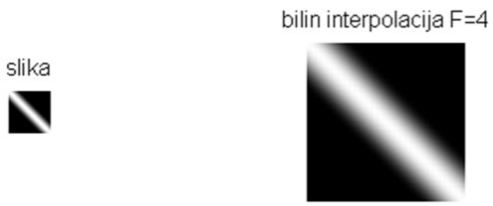
Primer izračunavanja jedne vrednosti piksela korišćenjem bilinearne interpolacije dat je na slici 3.



$$I_{20.2,14.5} = (1-0.2)\cdot 150.5 + 0.2\cdot 128.5 = 146.1$$

Slika 3 - Primer izračunavanja bilinearne interpolacije

Na slici 4 je prikazana interpolacija sa korišćenjem ovog algoritma. Uz veće zahteve u procesorskim resursima, postiže se bolja interpolacija ivica i prirodnija tekstura objekata.



Slika 4 - Rezultat bilinearne interpolacije sa faktorom povećanja 4

Složeniji algoritmi interpolacije postižu bolji kvalitet sa sve većim procesorskim resursima. To se postiže povećanjem kvadrata interpolacije (N=4,8), to jest susednih tačaka na osnovu kojih se interpolira vrednost u interpolacionoj tački, i složenijim određivanjem koeficijenata interpolacije uzimajući u obzir strukturu objekata u slici.

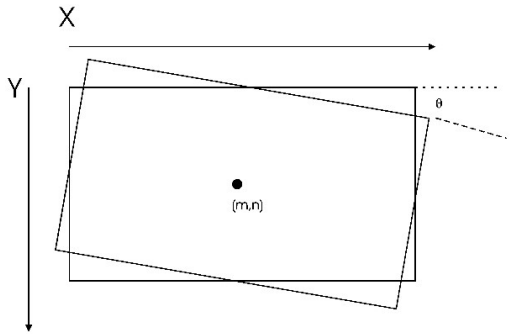
3 Rotacija slike

Još jedna operacija nad slikama pored promene veličine koja podrazumeva primenu interpolacionih tehnika jeste rotacija slike. Rotacija slike oko piksela sa koordinatama (0, 0) (gornji levi ugao) se na prost način može izvršiti primenom sledeće jednačine:

$$X' = X * cos(\theta) + Y * sin(\theta)$$

$$Y' = Y * cos(\theta) - X * sin(\theta)$$

gde je θ željeni ugao rotacije. Nakon izračunavanja *X'* i *Y'* potrebno je u rezultujuću sliku na koordinate *X* i *Y* upisati vrednosti piksela ulazne slike sa koordinatama *X'* i *Y'*. S obzirom da *X'* i *Y'*najčešće nisu celi brojevi prilikom određivanja vrednosti piksela primenjuje se jedna od tehnika interpolacije.



Slika 5 - Rotacija slike oko proizvoljne tačke

Rotacija slike oko proizvoljne tačke vrši se upotrebom sledeće jednačine:

$$X' = X * cos(\theta) - Y * sin(\theta) - m * cos(\theta) + n * sin(\theta) + m$$

$$Y' = Y * cos(\theta) + X * sin(\theta) - m * sin(\theta) - n * cos(\theta) + n$$

gde m i n predstavljaju koordinate tačke oko koje se slika rotira.

ZADATAK

3.1 Zadatak 1

Realizovati funkciju

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

Koja vrši skaliranje slike upotrebom *Sample and hold* algoritma opisanog u prethodnom odeljku.

Parametri funkcije su:

- `input` – ulazna slika u RGB formatu
- `xSize` – horizontalna dimenzija ulazne slike u pikselima
- `ySize` – 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

Prilikom poziva funkcije neophodno je inicijalizovati memoriju za izlaznu sliku. Veličinu izlazne slike potrebno je izračunati na osnovu dimenzija ulazne slike i faktora skaliranja. Faktor skaliranja po vertikalnoj dimenziji dat je parametrom params[0], a po horizontalnoj sa params[1].

Obradu vršiti nad slikom u RGB formatu.

3.2 Zadatak 2

Realizovati funkciju

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

Koja vrši bilinearnu interpolaciju slike na osnovu algoritma opisanog u prethodnom odeljku.

Parametri funkcije su:

- `input` – ulazna slika u RGB formatu
- `xSize` – horizontalna dimenzija ulazne slike u pikselima
- `ySize` – 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

Prilikom poziva funkcije neophodno je inicijalizovati memoriju za izlaznu sliku. Veličinu izlazne slike potrebno je izračunati na osnovu dimenzija ulazne slike i faktora skaliranja. Faktor skaliranja po vertikalnoj dimenziji dat je parametrom params[0], a po horizontalnoj sa params[1].

Obradu vršiti nad slikom u RGB formatu.

3.3 Zadatak 3

Realizovati funkciju

- `voidimageRotate(uchar input[], int xSize, int ySize, uchar output[], int m, int n, double angle);`

Koja vrši rotaciju slike oko tačke sa koordinatama (*m, n)* za ugao *angle*. Izlazna slika je istih dimenzija kao i ulazna. Piksele u okviru izlazne slike za koje su izračunate koordinate van opsega ulazne slike popuniti crnom bojom. Za određivanje vrednosti piksela za koje izračunate koordinate u originalnoj slici ne predstavljaju ceo broj, koristiti *Sample and hold* algoritam interpolacije.

Prilikom poziva funkcije neophodno je inicijalizovati memoriju za izlaznu sliku. Parametre m i n postaviti tako da se nalaze na sredini slike. Ugao rotacije dat je parametrom params[0].

3.4 Zadatak 4

Realizovati funkciju

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

Koja vrši rotaciju slike oko tačke sa koordinatama (*m, n)* za ugao *angle*. Izlazna slika je istih dimenzija kao i ulazna. Piksele u okviru izlazne slike za koje su izračunate koordinate van opsega ulazne slike popuniti crnom bojom. Za određivanje vrednosti piksela koristiti bilinearnu interpolaciju.

Prilikom poziva funkcije neophodno je inicijalizovati memoriju za izlaznu sliku. Parametre m i n postaviti tako da se nalaze na sredini slike. Ugao rotacije dat je parametrom params[0].