UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA RAČUNARSKU TEHNIKU I RAČUNARSKE KOMUNIKACIJE

Naziv predmeta:

Osnovi Algoritama i Struktura DSP 2

Projektni zadatak 2

Profesor:
Željko Lukač

Student:
Damjan Glamočić, RA65-2015

Novi Sad, Jun 2018.

Izveštaj

Zadatak 1):

U prvom zadatku bilo je potrebno implementirati funkciju: `void sampleAndHold(const uchar input[], int xSize, int ySize, uchar output[], int newXSize, int newYSize)`, koja vrši skaliranje slike upotrebom *sample and hold* algoritma. To je najjednostavniji algoritam, u kojem se za vrijenost piksela u skaliranoj slici uzima vrijednost najbližeg piksela u osnovnom rasteru. Jednostavna formula po kojoj je to moguće izvesti:

$$I_i(p,q) = I\left(\left[\frac{p-1}{F}+1\right],\left[\frac{q-1}{F}+1\right]\right)$$

u kojoj su ***p*** i ***q*** koordinate u skaliranoj slici, a ***F*** je faktor skaliranja (odnos dimenzija izlazne i ulazne slike).

Slika 1 : Rezlutat interpolacije sample and hold algoritmom za parametre 10, 10

Zadatak 2):

U ovom zadatku bilo je potrebno implementirati funkciju: `void bilinearInterpolate(const uchar input[], int xSize, int ySize, uchar output[], int newXSize, int newYSize)`, koja vrši *bilinearnu* interpolaciju slike. Algoritam je nešto složeniji u odnosu na prethodni, jer koristi 4 najbliža piksela u osnovnom rasteru. Uticaj svakog od ta 4 piksela zavisi od udaljenosti tog piksela od piksela u izlaznoj slici za koji se računa vrijenost. Formula za bilinearnu intepolaciju:

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

u kojoj su ***a*** i ***b*** udaljenosti, a ***m*** i ***n*** koordinate piksela u ulaznoj slici. Parametri ***a*** i ***b*** se računaju po formuli: $a = n_s / Sh - floor(n_s / Sh)$ $b = m_s / Sv - floor(m_s / Sv)$,

gdje su ***Sh*** i ***Sv*** horizontalni, odnosno veritkalni faktor skaliranja.

Slika 2 : Rezlutat bilinearne interpolacije za parametre 10, 10

Zadatak 3):

U ovom zadatku bilo je potrebno implementirati funkciju: `void bicubicInterpolate(const uchar input[], int xSize, int ySize, uchar output[], int newXSize, int newYSize)`, koja vrši *bikubičnu* interpolaciju slike. U ovom algoritmu se koristi 16 okolnih piksela iz originalne slike, i vrijeddnosti piksela koji su najbliži interpoliranom pikselu imaju najveći uticaj na interpoliranu vrijednost. Ideja je da se prvo uradi kubična interpolacija po jednoj dimenziji, a zatim po drugoj. Kubična interpolacija je opisana sljedećom jednačinom:

$$w(d)=\begin{cases}\frac{3}{2}|d|^3-\frac{5}{2}|d|^2+1 & |d|<1\\ -\frac{1}{2}|d|^3+\frac{5}{2}|d|^2-4|d|+2 & 1\le|d|<2\\ 0 & \text{inače}\end{cases}$$

$$Y(n)=\sum_{m=0}^{3}X(m)*w(d)$$

gdje je ***d*** udaljenost interpoliranog piksela i originalnog rastera, a ***X(0...3)*** izdvojene 4 vrijednosti oko vrijednosti ***Y(n).***

Slika 3 : Rezlutat bikubične interpolacije za parametre 10, 10

Zadatak 4):

U ovom zadatku bilo je potrebno implementirati funkciju: `void imageTransform(const uchar input[], int xSize, int ySize, uchar output[], double k1, double k2)`, koja primijenjuje efekat *riblje oko* na izlaznu sliku. U obradi slike, efekat „riblje oko“ predstavlja oštećenje slike nastalo zbog nesavršenosti sočiva kojim je slika ili video snimljen. Ovaj efekat se manifestuje kao geometrijsko izobličenje na krajevima slika. Može se modelovati na sljedeći način:

$$X' - m = (X - m) * (1 + k1 * r2 + k2 * r4)$$

$$Y' - n = (Y - n) * (1 + k1 * r2 + k2 * r4) \qquad X_{norm} = (X - m) / X\_SIZE$$

$$r2 = X_{norm}^2 + Y_{norm}^2 \qquad Y_{norm} = (Y - n) / Y\_SIZE$$

U navedenim jednačinama ***X*** i ***Y*** predstavljaju koordinate piksela u izlaznoj (izobličenoj) slici, ***X'*** i ***Y'*** koordinate piksela u ulaznoj slici, dok su m i n koordinate koje odgovaraju sredini slike. ***Xnorm*** i ***Ynorm*** predstavljaju normalizovane vrednosti koordinata. Za određivanje vrijednosti piksela za koje se izračunaju koordinate u originalnoj slici, ne predstavljaju cio broj, potrebno je koristiti *sample and hold* algoritam interpolacije.

Slika 4 : Rezlutat efekta riblje oko za parametre 0.666, 0.6666 (lijevo) i -0.666, -0.666(desno)

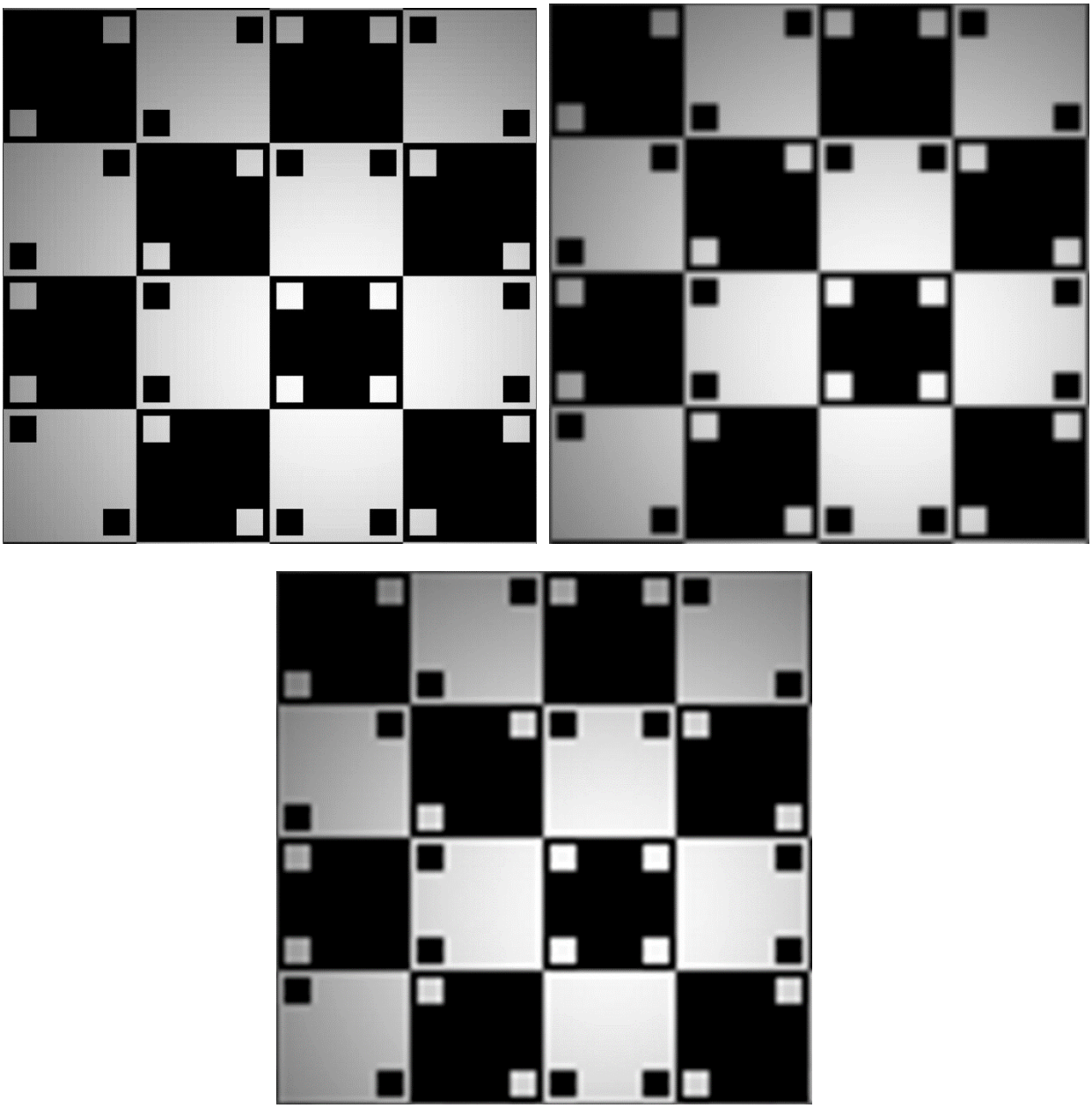
Zadatak 5):

U ovom zadatku bilo je potrebno implementirati funkciju:
`void imageTransformBilinear(const uchar input[], int xSize, int ySize, uchar output[], double k1, double k2),` koja primijenjuje efekat *riblje oko* na izlaznu sliku. Jedina razlika u odnosu na prethodni zadatak je što se za određivanje vrijednosti piksela za koje se izračunaju koordinate u originalnoj slici, ne predstavljaju cio broj, ne koristi *sample and hold* algoritam interpolacije, nego bilinearna interpolacija.

Slika 5 : Rezlutat efekta riblje oko za parametre 0.666, 0.6666 (lijevo) i -0.666, -0.666(desno)

Zaključak:

Na osnovu svega do sad odrađenog, zaključujem da je sample and hold algoritam najjednostavniji, i na polju kvaliteta slike daje najlošije kvalitete slike, ali je najbrži i to je njegova prednost (sliku 256x256 uveća 10 puta za 192 ms). Algoritam bilinearne interpolacije daje kvalitetnije rezultate, ali na račun procesorskog vremena (istu sliku uveća 10 puta za 638 ms). Bikubična interpolacija daje najkvalitetniju sliku, ali je i daleko najsporija (sliku 256x256 uveća 10 puta za 9.335 s). Takođe treba napomenuti da je bolje koristiti sample and hold algoritam, ukoliko su ivice u slici ravne i oštre, što se jasno vidi na slikama ispod.



Slika 6 : Rezlutat sample and hold algoritma(gore lijevo), bilinearne(gore desno) i bikubične interpolacije(dole), za sliku sa ravnim i oštrim ivicama

Bonus:

Kao bonus, trebalo je implementirati funkcije koje vrše inverznu transformaciju na slikama na kojima je već primijenjen efekat ribljeg oka, pomoću sample and hold alogirtma, kao i bilinearne interpolacije. Piksele koji izlaze iz opsega slike, potrebno je popuniti crnom bojom.

Slika 7 : Rezlutat inverzne transformacije ribljeg oka za parametre 0.666, 0.6666. Originalna slika (gore lijevo), primijenjen efekat riblje oko za navedene parametre (gore desno), korekcija primjenom sample and hold algoritma (dole lijevo), korekcija primjenom bilinearne interpolacije(dole desno)

Slika 8 : Rezlutat inverzne transformacije ribljeg oka za parametre -0.666, -0.6666. Originalna slika (gore lijevo), primijenjen efekat riblje oko za navedene parametre (gore desno), korekcija primjenom sample and hold algoritma (dole lijevo), korekcija primjenom bilinearne interpolacije(dole desno)

Zaključak:

Nakon transformacije ribljeg oka, moguće je vratiti originalnu sliku, ali uz određene gubitke. Gubici se ne mogu izbjeći zbog toga što se za pozitivne parameter k1 i k2 (barrel distortion) slika dobija bez učešća piksela koji predstavljaju krive linije na slici 7. Oni ne postoje u transformisanoj slici, pa se ne mogu vratiti u original. Isti je slučaj kod pozitvnih parametara (pincushion distortion), samo se gube pikseli izvan centralnog kruga na slici 8, koji učestvuje u transformaciji slike. Takođe, jasno se vidi da bolje rezultate daje sample and hold algoritam, zato što nema povećanja dimenzija slike, pa je bespotrebno koristiti bilinearnu interpolaciju, koja koristi 4 okolna piksela iz osnovnog rastera u formiranju izlaznog piksela. Bolje je koristiti jednostavan algoritam kopiranja najbližeg piksela iz ulazne slike