Interpolacija slike

Ivanov, Andrea

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:188069

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-08-05**

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Informatics and Digital Technologies - INFORI Repository

zir.nsk.hr

DIGITALNA KNJIŽNICA

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

Sveučilište u Rijeci – Odjel za informatiku

Preddiplomski jednopredmetni studij informatike

Andrea Ivanov

Interpolacije slike

Završni rad

Mentor: doc. dr. sc. Bojan Crnković

Rijeka, srpanj 2018.

Sadržaj

1. Zadatak za završni rad ... 3
2. Sažetak ... 4
3. Uvod ... 5
4. Digitalna slika ... 7
4.1. Rezolucija slike ... 7
4.2. Intenzitet piksela ... 8
5. Interpolacijski polinom ... 10
5.1. Lagrangeov oblik interpolacijskog polinoma ... 11
5.1.1. Primjer Lagrangeovog interpolacijskog polinoma ... 12
6. Bilinearna interpolacija ... 14
6.1. Primjer ... 15
7. Bikubična interpolacija ... 17
7.1. Primjer ... 18
8. Directional Cubic Convolution Interpolation (DCCI) ... 19
8.1. Algoritam ... 19
8.1.1. Izračunavanje piksela u dijagonalnim razmacima ... 19
8.1.2. Računanje preostalih piksela ... 21
8.2. Primjer DCCI ... 25
9. Mean squared error (MSE) ... 26
10. Usporedba različitih načina interpolacije ... 27
10.1. Razlika između interpolacija ... 28
11. Zaključak ... 29
12. Popis slika ... 30
13. Popis tablica ... 31
14. Popis priloga ... 32
15. Literatura ... 40

1. Zadatak za završni rad

U ovom radu pojam interpolacija slike se odnosi na promjenu broja piksela raster fotografije upotrebom prostornih 2D interpolacijskih algoritama. Osnovni zadatak je usporediti kvalitetu aproksimacije standardnim bilinearnim i bikubičnim algoritmima s modernim prostorno adaptivnim algoritmom.

2. Sažetak

Svi smo se našli u situaciji kada na računalu želimo uvećati sliku, ili popraviti njezin oštećen dio, ili ju jednostavno želimo urediti kako bi ljepše izgledala. Za sve to potrebna nam je interpolacija slike o kojoj je i riječ u ovom radu. Prvenstveno je obraćena pažnja na promjene koje se događaju prilikom uvećanja slika. Kako se slika sastoji od piksela, a mijenjanjem njezinih dimenzija mijenja se i položaj piksela, tj. nastaju praznine koje treba popuniti novim pikselima, dolazi do potrebe za interpoliranjem. Interpolacija nam pomaže u izračunavanju tih novih vrijednosti, a neki od načina su ovdje i opisani.

Opisane su tri odabrane interpolacije, a to su: bilinearna interpolacija, bikubična interpolacija i DCCI (Directional Cubic Convolution Interpolation), te su na kraju uspoređene. Prije njih opisan je i Lagrangeov interpolacijski polinom koji se koristi prilikom bilinearne i bikubične interpolacije. Kako bi se bolje razumio sam pojam interpolacije slike, na početku rada opisana je digitalna slika i njezin način spremanja na računalu.

Ključne riječi: piksel, digitalna slika, rezolucija, interpolacijski polinom, Lagrangeov interpolacijski polinom, interpolacija, bilinearna interpolacija, bikubična interpolacija, DCCI, MSE.

3. Uvod

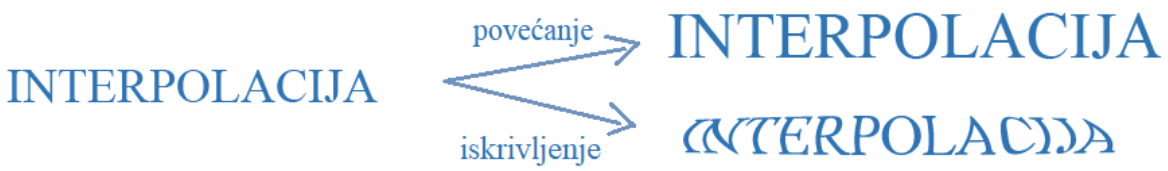
Interpolacija dolazi od riječi *inter* što znači između i riječi *polos* što znači os, tj. točka ili čvor. Svako izračunavanje nove točke između dviju ili više postojećih točaka podataka naziva se interpolacija. Niz od n različitih brojeva x_k nazivamo čvorovima. Ako za svaki broj x_k postoji drugi broj y_k, onda možemo odrediti funkciju f za koju vrijedi:

$$f(x_k) = y_k, \quad k = 1, 2, \dots, n$$

Par $(x_{k,}\, y_k)$ se naziva točka podataka, a f interpolant za te točke podataka.

Interpolacija je tehnika koja se nalazi u mnogih aplikacijama. Ona gotovo nikad nije cilj, ali utječe na željene rezultate i na načine dobivanja tih podataka. Njezini temeljni podaci kontinuirano se definiraju. S obzirom na uzorke podataka, moguće je izračunati vrijednost podataka temeljne kontinuirane funkcije za bilo koju apscisu[1].

Interpolacija slike se javlja na svim digitalnim fotografijama u nekoj fazi – bilo da se radi o rekonstrukciji boja ili povećanju fotografije. Događa se svaki put kod mijenjanja veličine slike ili preoblikovanja (iskrivljenja) slike iz jedne rešetke (koordinatne mreže) piksela u drugu. Promjena veličine slike nužna je kada je potrebno povećati ili smanjiti ukupan broj piksela, dok je preoblikovanje nužno prilikom promjene perspektive ili rotacije slike.



Slika 1. Interpolacija slike

Čak i ako se izvrši slična promjena veličine ili preoblikovanja, rezultati se mogu značajno razlikovati jer ovise o algoritmu interpolacije. To je samo aproksimacija[2] pa će slika svaki put, kad se izvrši interpolacija, gubiti na kvaliteti.

Algoritmi interpolacije se mogu podijeliti u dvije skupine: prilagodljivi i neprilagodljivi. Neprilagodljive metode tretiraju sve piksele jednako, dok se prilagodljive metode mijenjaju ovisno o tome što interpoliraju (oštri rubovi u odnosu na glatku teksturu).

Neprilagodljivi interpolacijski algoritmi su: najbliži susjed, bilinearna, bikubična, spline, Lanczos, … Što više susjednih piksela obuhvaćaju, njihovi rezultati je biti točniji, ali se time povećava i vrijeme obrade. Koriste se za iskrivljavanje i promjenu veličine slike.

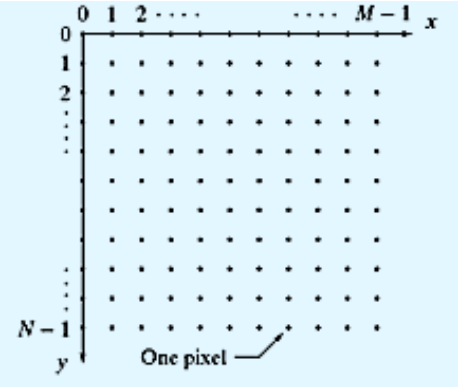
[1] Os x (vodoravni pravac)
[2] Matematički pojam koji označava postupak pronalaženja funkcije koja približno aproksimira (opisuje) zadani konačan skup točaka ili neku drugu funkciju

Prilagodljivi interpolacijski algoritmi uključuju mnoge vlasničke algoritme u licenciranom softveru kao što su: Qimage, PhotoZoom Pro, … Prvenstveno su dizajnirani kako bi smanjili pojavljivanje artefakata prilikom povećanja fotografija.

U nastavku će biti opisani neki od neprilagodljivih algoritama, kao što su: bilinearna i bikubična interpolacija te njihova usporedba s interpolacijom usmjerene kubične konvolucije (Directional Cubic Convolution Interpolation) koja pripada prilagodljivim algoritmima.

4. Digitalna slika

Digitalna slika je prikaz dvodimenzionalne slike s konačnim skupom digitalnih vrijednosti koje nazivamo pikselima. Pikseli su organizirani u pravokutnu matricu s M stupaca i N redaka. Širina slike je određena brojem stupaca, a visina brojem redaka u matrici. Kako bi mogli jedinstveno identificirati određeni piksel, za njega definiramo koordinate, x i y. Koordinatni sustav matrice slike je definiran tako da x raste od lijeva prema desno, a y od gore prema dolje te je prikazan na Slici 2. Piksel $(0,0)$ se smatra ishodištem ovog sustava. Za koordinate piksela se uvijek uzimaju cijeli brojevi.



Slika 2. Koordinatni sustav slike
(Preuzeto s: https://blogs.mathworks.com/steve/2011/08/26/digital-image-processing-using-matlab-digital-image-representation/)

4.1. Rezolucija slike

Rezolucija slike je veličina koja označava broj piksela koji tvore sliku, tj. govorimo o dimenziji matrice piksela $(M \times N)$. Dimenzije u pikselima obilježavaju ukupan broj piksela u jednom stupcu i jednom retku.

Stvarna veličina slike prikazane digitalne slike na nekom uređaju, ovisi o gustoći piksela uređaja koji ju prikazuje. Stvarna veličina slike mjeri se u centimetrima ili inčima (1 inč = 2,54 cm), a gustoća piksela mjeri se u broju točaka po inču (dpi – dots per inch). Veličinu slike možemo odrediti prema formuli:

$$stvarna\ veličina = \frac{rezolucija\ slike}{gustoća\ piksela}$$

Na primjer, stvarnu veličinu slike od 1 600 x 1 200 piksela prikazanu na uređaju od 72 dpi možemo izračunati na sljedeći način:

$$širina = \frac{1600}{72} = 22{,}2\ inča \cdot 2{,}54 = 56{,}4\ cm$$

$$visina = \frac{1200}{72} = 16{,}7\ inča \cdot 2{,}54 = 42{,}3\ cm$$

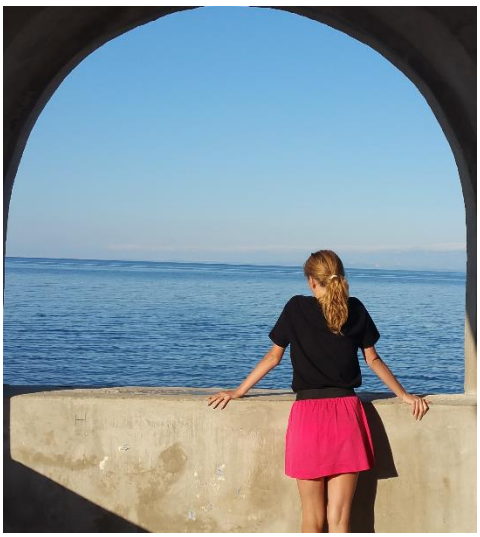
Dakle, širina te slike u stvarnosti iznosi 56,4 cm, a visina 42,3 cm.

4.2. Intenzitet piksela

Svaki piksel, osim već spomenutih koordinata x i y, ima vlastitu vrijednost intenziteta. Ako svi pikseli imaju istu vrijednost intenziteta, slika će biti u jednoj boji, npr. cijela crna, bijela, siva ili bilo koja druga boja.

Vrijednost intenziteta u digitalnim slikama određena je bitovima. Bit je binaran i ima samo dvije moguće vrijednosti, 0 ili 1. Monokromatska ili jednobojna slika je najjednostavnija vrsta slike kod koje se svaki piksel sprema kao jedan bit (0 ili 1) i kvaliteta slike ovisi samo o rezoluciji. Kod slika u sivim tonovima, svakom pikselu je dodijeljeno 8 bita ili 1 bajt pa može poprimiti bilo koju vrijednost od 0 do 255 ($2^8 = 256$). RGB sustav boja koristi aditivno miješanje osnovnih (primarnih) boja Red – Green – Blue (crveno – zeleno – plavo). Njihovim miješanjem možemo dobiti sve boje koje pripadaju našem vidljivom spektru. Svaki piksel je spremljen kao tri bajta (24 bita), po jedan za svaku RGB boju. Kod takvih slika je moguće $2^{24} = 16\,777\,216$ ($256 \times 256 \times 256$) kombinacija različitih boja. Npr. vrijednost (0, 0, 0) je crna boja, a (255, 255, 255) je bijela boja.

Slika 3. Jednobojna slika



Slika 4. RGB slika

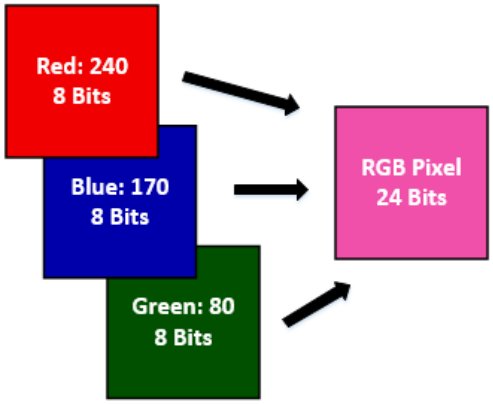
Matrica za jednobojnu sliku izgleda ovako:

$$\begin{matrix} f(1,1) & f(1,2) & \dots & f(1,M) \\ f(2,1) & f(2,2) & \dots & f(2,M) \\ \vdots & \vdots & \vdots & \vdots \\ f(N,1) & f(N,2) & \dots & f(N,M) \end{matrix}$$

pri čemu je M broj stupaca, N broj redaka, a $f(N,M)$ intenzitet slike u toj točki. Za sliku kažemo da je digitalna kada su sve te tri vrijednosti konačne i diskretne.

Kod RGB slika svaki piksel se dobije kombiniranjem crvene, zelene i plave boje, što je vidljivo na primjeru na Slici 5. Zapisuje se kao trojka (R, G, B) pa svaka vrijednost matrice RGB slike se sastoji od jedne trojke, npr. (255, 255, 0) što određuje žutu boju. Umjesto

dekadskog[3] zapisa boja često se koristi heksadekadski[4] zapis sa 6 znamenki (po dvije za svaku RGB boju). Prema njemu zapis crne boje je #000000, bijele #FFFFFF, žute #FFFF00, itd. Na Slici 6 prikazan je svaki sloj boja zasebno te njihov zajednički utjecaj na krajnji izgled slike.



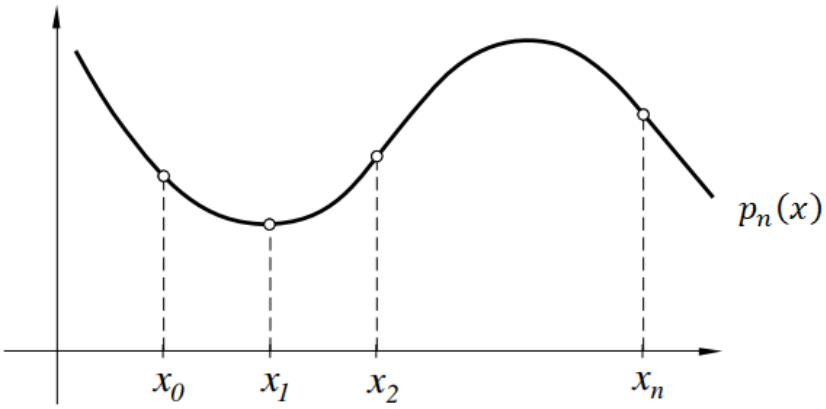
Slika 5. Primjer RGB piksela
(Preuzeto s: https://www.codeproject.com/Articles/1112774/Converting-an-Image-to-Grayscale-in-MatLab)

Slika 6. RGB slika razdvojena na R, G i B sloj
(Preuzeto s: https://www.wolmerfoto.it/Fototecnica/Photoshop/Guide/BitPlane/BitPlane.htm)

[3] Dekadski brojevni sustav je sustav s bazom 10 te su za prikaz svih brojeva dovoljne znamenke od 0 do 9.
[4] Heksadekadski brojevni sustav je sustav s bazom 16. Za prikazivanje znamenki koristi se 16 različitih znakova, a to su brojevi od 0 do 9 te slova A, B, C, D, E i F za brojeve od 10 do 15, redom.

5. Interpolacijski polinom

Zadane su vrijednosti funkcije f tako da $y_k = f(x_k), k = 0, 1, 2, \dots, n$ pri čemu vrijedi da su argumenti međusobno različiti, $x_0 < x_1 < x_2 < \cdots < x_n$. Ti podaci u ravnini predstavljaju točke koje čine graf Γ_f.



Slika 7. Interpolacijski polinom
(Preuzeto s: https://element.hr/artikli/file/1314)

Potrebno je izračunati aproksimirane vrijednosti funkcije f izvan točaka x_k, odnosno za vrijednosti argumenta x, $x \neq x_k$. Znamo vrijednosti funkcije u točkama $x_0, x_1, \dots, x_k$, dok nam je njezin oblik potpuno nepoznat. Zato moramo nepoznatu funkciju f zamijeniti s drugom funkcijom, koja je nama poznata i jednostavnija od f te ima iste vrijednosti u zadanim točkama. Najjednostavniji primjer takve funkcije je polinom, što nas dovodi do interpolacijskog polinoma:

$$P(x) = a_0 + a_1 x + \cdots + a_n x^n$$

pri čemu je stupanj polinoma manji ili jednak n. Vrijednosti tog polinoma u točkama x_k jednake su vrijednostima $f(x_k)$, odnosno vrijedi:

$$P(x_k) = f(x_k), \qquad k = 0, 1, 2, \dots, n$$

gdje točke x_k nazivamo čvorovima.

Ako je zadano $n + 1$ točaka, to znači da postoji točno jedan polinom manjeg ili jednakog stupnja n koji interpolira taj skup podataka. O tome govori sljedeći teorem:

Teorem 1. *Neka je* $n \in \mathbb{N}_0$. *Za zadane točke* $(x_k, y_k), k = 0, 1, 2, \dots, n$, *gdje je* $x_0 < x_1 < \cdots < x_n$, *postoji jedinstveni interpolacijski polinom* $p \in P_n$, *stupnja najviše n tako da vrijedi* $p(x_k) = y_k, k = 0, 1, 2, \dots, n$.

Taj interpolacijski polinom se može naći na više načina, a jedan od njih je Lagrangeov oblik interpolacijskog polinoma.

5.1. Lagrangeov oblik interpolacijskog polinoma

Kako bi odredili koeficijente interpolacijskog polinoma ne moramo rješavati linearni sustav jednadžbi. Interpolacijski polinom p_n možemo odmah zapisati u tzv. Lagrangeovoj bazi prostora P_n:

$$p_n(x) = \sum_{k=0}^{n} f(x_k) l_k(x)$$

pri čemu je:

$$l_k(x_i) = \begin{cases} 0, & i \neq k \\ 1, & i = k \end{cases}$$

Interpolacijski polinom l_k možemo kako pronaći jer su sva čvorišta, osim x_k, njegove nultočke. Slijedi:

$$l_k(x) = C_k(x - x_0)(x - x_1) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n)$$

gdje je $C_k$konstanta, koju možemo odrediti tako da umjesto x uvrstimo x_k. Kako je $l_k(x_k) = 1$, dobijemo:

$$1 = C_k(x_k - x_0)(x_k - x_1) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_n)$$

odnosno:

$$C_k = \frac{1}{(x_k - x_0)(x_k - x_1) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_n)}$$

Uvrstivši C_k u gornju jednadžbu za l_k dobijemo:

$$l_k(x) = \frac{(x - x_0)(x - x_1) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n)}{(x_k - x_0)(x_k - x_1) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_n)}$$

što jednostavnije možemo zapisati:

$$l_k(x) = \prod_{i=0, i \neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

Iz toga slijedi da je početna formula za $p_n(x)$ jednaka:

$$p_n(x) = \sum_{k=0}^{n} f(x_k) \prod_{i=0, i \neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

te se taj zapis zove Lagrangeov oblik interpolacijskog polinoma, a l_k Lagrangeovi bazni polinomi.

Lagrangeov polinom kroz dvije točke, $(x_0, f(x_0))$ i $(x_1, f(x_1))$, daje polinom prvog stupnja ($n = 1$), tj. jednadžbu pravca:

$$p_1(x) = f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0}$$

Kada su zadane tri točke, $(x_0, f(x_0))$, $(x_1, f(x_1))$ i $(x_2, f(x_2))$, pomoću Lagrangeovog polinoma dobijemo polinom drugog stupnja ($n = 2$), tj. jednadžbu parabole:

$$p_2(x) = f(x_0)\frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + f(x_1)\frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + f(x_2)\frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

Za četiri zadane točke, $(x_0, f(x_0))$, $(x_1, f(x_1))$, $(x_2, f(x_2))$ i $(x_3, f(x_3))$, dobijemo polinom trećeg stupnja ($n = 3$), tj. kubičnu jednadžbu:

$$p_3(x) = f(x_0)\frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)} + f(x_1)\frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)}$$
$$+ f(x_2)\frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)} + f(x_3)\frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)}$$

Na sličan način možemo dobiti Lagrangeov polinom za bilo koji broj točaka, s time da za n točaka dobijemo polinom n-1 stupnja. Rješavanjem linearnog sustava jednadžbi ili primjenom Lagrangeovog polinoma dođemo do istog rezultata jer je interpolacijski polinom jedinstven.

5.1.1. Primjer Lagrangeovog interpolacijskog polinoma

Potrebno je odrediti Lagrangeov interpolacijski polinom za četiri točke zadane tablicom:

Tablica 1. Točke i njihove funkcijske vrijednosti

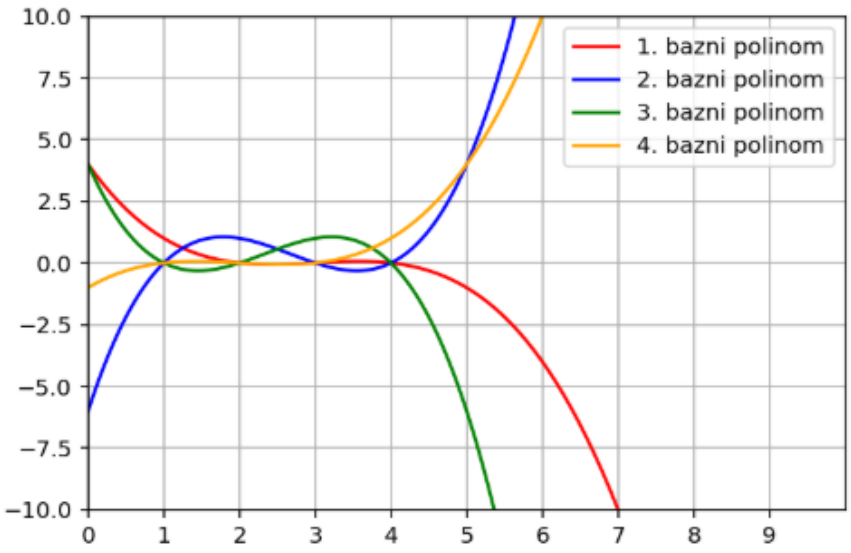
x	1	2	3	4
y	4	1	3	5

Rješenje:

Prvo izračunamo sve bazne polinome prema formuli:

$$l_k(x) = \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

te dobijemo:



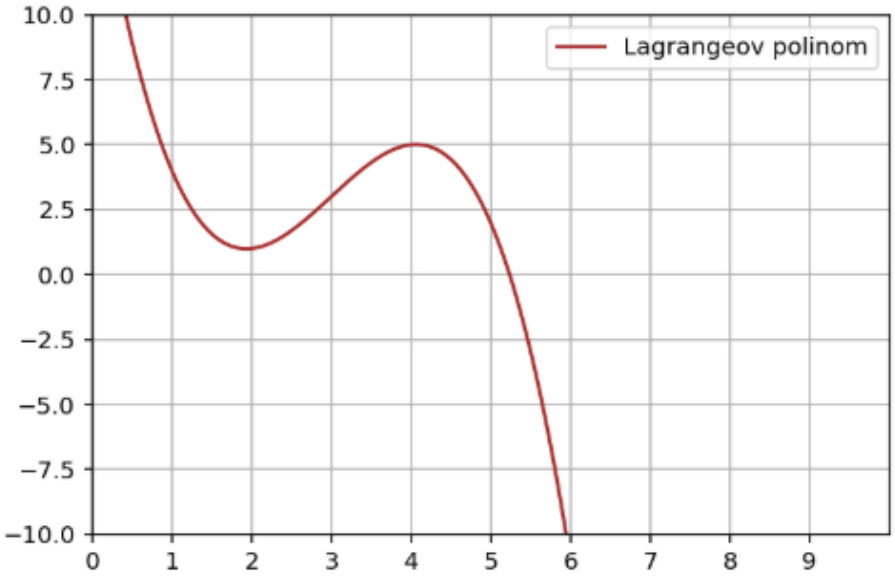
Slika 8. Bazni polinomi

Svaki od njih, $l_k, k = 1, \dots, 4$, siječe x os u $x[i] = 1, \dots 4$ pri čemu je $i \neq x$. Sada lako izračunamo Lagrangeov polinom prema formuli:

$$p_n(x) = \sum_{k=0}^{n} f(x_k) l_k(x)$$

te dobijemo:

$$p_n(x) = -0.833x^3 + 7.5x^2 - 19.667x + 17$$



Slika 9. Lagrangeov polinom

6. Bilinearna interpolacija

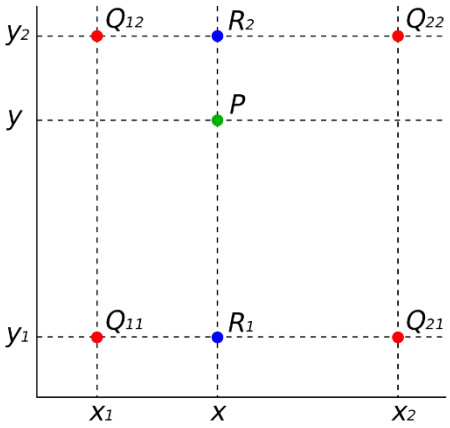
Bilinearna interpolacija je nastavak linearne interpolacije[5] za interpoliranje funkcija dviju varijabli (npr. x i y) u dvije dimenzije. Njezina temeljna ideja je da se prvo linearna interpolacija provodi u jednom smjeru (smjeru osi x), a zatim u drugom smjeru (smjeru osi y). Iako je svaki korak bilinearne interpolacije u pronalasku novih točaka linearan, ukupan rezultat je kvadratan.

Neka je $f(x,y)$ funkcija dvije varijable koja ima poznate vrijednosti u kutovima kvadrata. Pretpostavimo da želimo izračunati vrijednost te funkcije u nekoj točki unutar kvadrata. To možemo odrediti pomoću izraza:

$$f(x,y) = \sum_{i=0}^{1}\sum_{j=0}^{1} a_{ij}x^i y^j$$

$$f(x,y) = a_{00} + a_{10}x + a_{01}y + a_{11}xy$$

kroz četiri poznate vrijednosti.



Slika 10. Bilinearna interpolacija (Preuzeto s: https://en.wikipedia.org/wiki/Bilinear_interpolation)

Prvo izvršavamo linearnu interpolaciju između točaka $f(x_1,y_1)$ i $f(x_2,y_1)$ te dobijemo:
$f(x,y_1) = f(x_1,y_1) + \big(f(x_2,y_1) - f(x_1,y_1)\big)x.$

Na sličan način dobijemo i za točke $f(x_1,y_2)$ i $f(x_2,y_2)$:
$f(x,y_2) = f(x_1,y_2) + \big(f(x_2,y_2) - f(x_1,y_2)\big)x.$

Zatim vršimo linearnu interpolaciju u drugom smjeru:
$f(x,y) = f(x,y_1) + \big(f(x,y_2) - f(x,y_1)\big)x.$

Uvrstimo $f(x,y_1)$ i $f(x,y_2)$ u zadnju jednakost te dobijemo oblik bilinearne interpolacije:

$$f(x,y) = f(x_1,y_1) + \big(f(x_2,y_1) - f(x_1,y_1)\big)x + \big(f(x_1,y_2) - f(x_1,y_1)\big)y + \big(f(x_2,y_2) + f(x_1,y_1) - f(x_1,y_2) + f(x_2,y_1)\big)xy.$$

Drugi način za pronalaženje bilinearne interpolacije je korištenjem Lagrangeovog polinoma kroz dvije točke koji je opisan ranije. Pretpostavimo da želimo naći vrijednost nepoznate funkcije f u točki (x,y). Ako znamo vrijednost funkcije u četiri točke, $Q_{11} = (x_1,y_1)$, $Q_{12} = (x_1,y_2)$, $Q_{21} = (x_2,y_1)$ i $Q_{22} = (x_2,y_2)$, onda možemo izvršiti interpolaciju u smjeru osi x pomoću Lagrangeovog polinoma:

$$f(x,y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$f(x,y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

[5] Linearna interpolacija je metoda koja koristi linearne polinome (polinome prvog stupnja) za pronalazak novih točaka podataka između dvije poznate točke

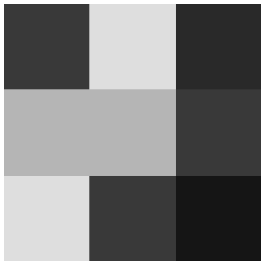
Nakon toga interpoliramo u smjeru osi y kako bi dobili traženu vrijednost:

$$
\begin{aligned}
f(x,y) &\approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\
&= \frac{y_2 - y}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})\right) \\
&\quad + \frac{y - y_1}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})\right) \\
&= \frac{1}{(x_2 - x_1)(y_2 - y_1)}\big(f(Q_{11})(x_2 - x)(y_2 - y) + f(Q_{21})(x - x_1)(y_2 - y) \\
&\quad + f(Q_{12})(x_2 - x)(y - y_1) + Q_{22}(x - x_1)(y - y_1)\big) \\
&= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 - x & x - x_1 \end{bmatrix} \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix}
\end{aligned}
$$

Rezultat će biti isti i ako prvo interpoliramo u smjeru osi y, a onda u smjeru osi x, tj. rezultat ne ovisi o tome u kojem ćemo smjeru prvo interpolirati.

6.1. Primjer

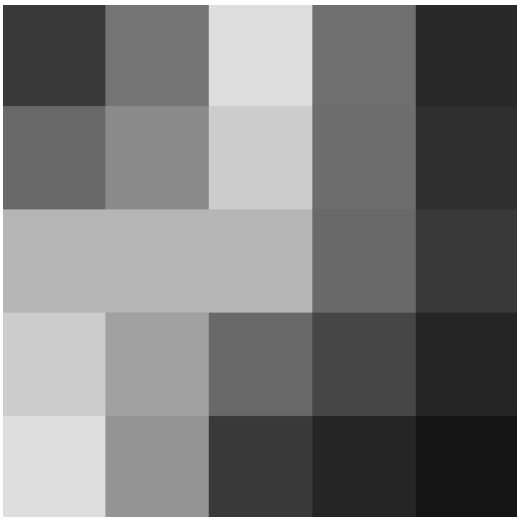
Imamo dvije originalne slike:



Slika 11. 3x3 piksela

Slika 12. Smješko

koje uvećavamo 2 puta bilinearnom interpolacijom kojom dobijemo:



Slika 13. 5x5 piksela (bilinearna interpolacija)

Slika 14. Uvećani smješko (bilinearna interpolacija)

Dimenzije uvećanih slika su uvijek 2 puta veće od originalnih dimenzija slike minus 1, npr. dimenzije slike od 3x3 piksela se duplim povećanjem promijene na 5x5 piksela, jer se između svaka dva piksela u originalnoj slici dodaje po jedan piksel.

7. Bikubična interpolacija

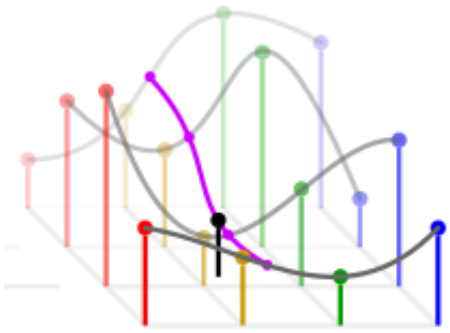
Bikubična interpolacija je nastavak kubične interpolacije[6] za interpoliranje točaka podataka u dvije dimenzije. Interpolirana površina je glađa u odnosu na površinu interpoliranu bilinearnom interpolacijom. Bikubična interpolacija se može dobiti pomoću Lagrangeovih polinoma, kubičnog splajna ili algoritma kubične konvolucije.

Pri obradi slike, odabire se bikubična interpolacija ako brzina nije najbitnija. Za razliku od bilinearne interpolacije, koja koristi 4 piksela (2×2), bikubična koristi 16 piksela (4×4) zbog čega i daje oštrije slike. Budući da se poznati pikseli nalaze na različitim udaljenostima od nepoznatog piksela, pikseli koji su bliži više utječu na rezultat.

Pretpostavimo da znamo vrijednosti funkcije f i derivacija f_x, f_y i f_{xy} u kutovima kvadrata. Interpoliranu površinu možemo zapisati s:

$$f(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$$

te se problem interpolacije sastoji od određivanja 16 koeficijenata.



Slika 15. Bikubična interpolacija (Preuzeto s: https://en.wikipedia.org/wiki/Bicubic_interpolation)

Za četiri najbliže točke oko tražene nepoznate točke, x_1, x_2, x_3 i x_4, prvo interpoliramo kubičnom interpolacijom u jednom smjeru, prema formuli:

$$f(x) = \frac{1}{2}\Big(\big((-x_1 + 3x_2 - 3x_3 + x_4)d + (2x_1 - 5x_2 + 4x_3 - x_4)\big)d + (-x_1 + x_3)\Big)\,d + x_2$$

pri čemu je d decimalni dio od realnog x interpolirane funkcije $f(x)$.

Zatim, za te dobivene vrijednosti, interpolirano kubičnom interpolacijom u drugom smjeru te kao rezultat dobijemo bikubičnu interpolaciju.

Primijenimo li Lagrangeov polinom u pronalaženju bikubične interpolacije, u smjeru osi x dobivamo:

$$\begin{aligned} f(x,y_i) \approx f(x_0,y_i)\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} + f(x_1,y_i)\frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} \\ + f(x_2,y_i)\frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} \\ + f(x_3,y_i)\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} \end{aligned}$$

[6] Kubična interpolacija je metoda koja koristi polinome trećeg stupnja pri interpolaciji između x_1 i x_2 ako su poznate vrijednosti funkcije f(x) i njezine derivacije u x_1 i x_2

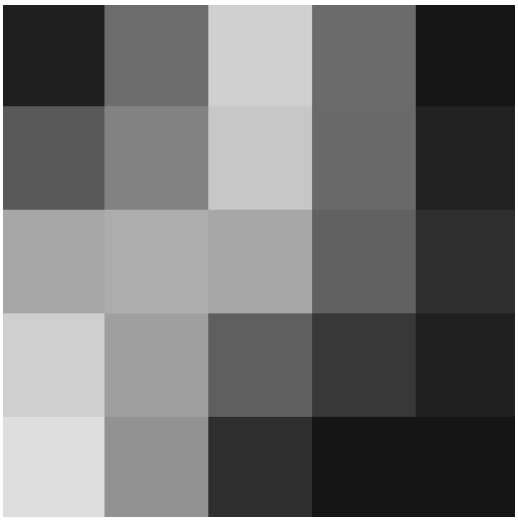
pri čemu je $i = 0{,}1\ 2,$ i 3, dok njegovom primjenom u smjeru osi y dobijemo:

$$f(x,y) \approx f(x,y_0)\frac{(y-y_1)(y-y_2)(y-y_3)}{(y_0-y_1)(y_0-y_2)(y_0-y_3)} + f(x,y_1)\frac{(y-y_0)(y-y_2)(y-y_3)}{(y_1-y_0)(y_1-y_2)(y_1-y_3)}$$
$$+ f(x,y_2)\frac{(y-y_0)(y-y_1)(y-y_3)}{(y_2-y_0)(y_2-y_1)(y_2-y_3)}$$
$$+ f(x,y_3)\frac{(y-y_0)(y-y_1)(y-y_2)}{(y_3-y_0)(y_3-y_1)(y_3-y_2)}$$

Kao i kod bilinearne, nije bitno u kojem smjeru prvo interpoliramo.

7.1. Primjer

Primjer originalnih slika jednak je kao i kod primjera za bilinearnu interpolaciju te duplim uvećanjem pomoću bikubične interpolacije dobivamo slike veće glatkoće:



Slika 16. 5x5 piksela (bikubična interpolacija)

Slika 17. Uvećani smješko (bikubična interpolacija)

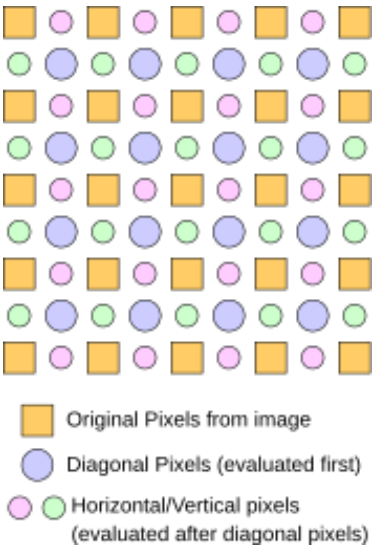
8. Directional Cubic Convolution Interpolation (DCCI)

DCCI je rubno usmjeren algoritam za skaliranje slike. Uzimanjem u obzir rubove u slici, ovaj algoritam smanjuje artefakte koji se pojavljuju primjenom drugih algoritama za skaliranje slike, poput prethodna dva opisana. Dimenzije skalirane slike su dva puta veće od originalnih dimenzija minus 1.

8.1. Algoritam

Sastoji se od tri glavna koraka:

1) Kopiranje originalnih piksela u završnu sliku s razmacima među njima
2) Izračunavanje piksela za dijagonalne razmake
3) Izračunavanje piksela za preostale horizontalne i vertikalne razmake



Slika 18. Algoritam DCCI
(Preuzeto s: https://en.wikipedia.org/wiki/Directional_Cubic_Convolution_Interpolation)

8.1.1. Izračunavanje piksela u dijagonalnim razmacima

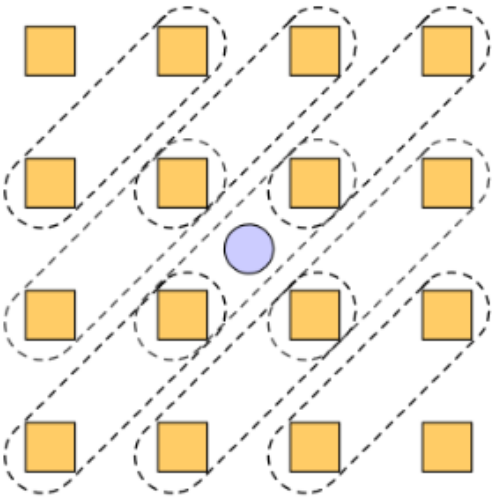
Evaluacija dijagonalnih piksela se vrši na podacima iz originalne slike u 4×4 regiji, s novim pikselom u sredini, u razmaku između originalnih piksela. Ovo se može protumačiti i kao 7×7 regija na uvećanoj slici centriranoj na poziciji novog izračunatog piksela, a originalni pikseli su već kopirani.
Algoritam bira jedan od tri slučaja:

1. Rubno u smjeru gore-desno → interpolira u smjeru dolje-desno
2. Rubno u smjeru dolje-desno → interpolira u smjeru gore-desno
3. Zaglađivanje površine → interpolira u oba smjera, a zatim množi vrijednosti s težinama

Računanje jačine dijagonalnog ruba

Neka je $d1$ suma svih rubova u smjeru gore-desno, a $d2$ suma u smjeru dolje-desno. Kako bi izračunali $d1$ trebamo sumirati sve $|(P(x,y) - P(x-1, y+1))|$, u regiji od $x = 1$

do 3 i $y = 0$ do 2, a za $d2$ treba sumirati $|(P(x,y) - P(x+1,y+1))|$ u regiji od $x = 0$ do 2 i $y = 0$ do 2.



Slika 20. Smjer dolje-desno (Preuzeto s: https://en.wikipedia.org/wiki/Directional_Cubic_Convolution_Interpolation)

Interpoliranje piksela

Ako $\frac{1+d1}{1+d2} > 1.15$ onda je rub u smjeru gore-desno, a ako je $\frac{1+d2}{1+d1} > 1.15$ onda je u smjeru dolje-desno. Inače, se nalazimo u glatkom području. Za zaobilaženje dijeljenja i operacija decimalnim brojevima, ovo se može zapisati kao $100 \cdot (1 + d1) > 115 \cdot (1 + d2)$ i $100 \cdot (1 + d2) > 115 \cdot (1 + d1)$.

Rub gore-desno

Za izračunavanje piksela u smjeru gore-desno trebamo interpolirati u smjeru dolje-desno prema formuli:

$$\frac{-1 \cdot P(0,0) + 9 \cdot P(1,1) + 9 \cdot P(2,2) - 1 \cdot P(3,3)}{16}$$

Vrijednosti piksela je potrebno staviti u granice validnih vrijednosti piksela (uobičajeno od 0 do 255).

Rub dolje-desno

Za izračunavanje piksela u smjeru dolje-desno trebamo interpolirati u smjeru gore-desno prema formuli:

$$\frac{-1 \cdot P(3,0) + 9 \cdot P(2,1) + 9 \cdot P(1,2) - 1 \cdot P(0,3)}{16}$$

Vrijednosti piksela je potrebno staviti u granice validnih vrijednosti piksela (uobičajeno od 0 do 255).

Glatko područje

U glatkom području, jačina ruba od gore-desno utječe na vrijednost dolje-desno piksela te jačina ruba od dolje-desno utječe na vrijednost gore-desno piksela.

$$w1 = \frac{1}{1 + d1^5}$$

$$w2 = \frac{1}{1 + d2^5}$$

$$težina1 = \frac{w1}{w1 + w2}$$

$$težina2 = \frac{w2}{w1 + w2}$$

$$DoljeDesnoPiksel = \frac{-1 \cdot P(0,0) + 9 \cdot P(1,1) + 9 \cdot P(2,2) - 1 \cdot P(3,3)}{16}$$

$$GoreDesnoPiksel = \frac{-1 \cdot P(3,0) + 9 \cdot P(2,1) + 9 \cdot P(1,2) - 1 \cdot P(0,3)}{16}$$

$$izračunatiPiksel = DoljeDesnoPiksel \cdot težina1 + GoreDesnoPiksel \cdot težina2$$

Vrijednosti piksela je potrebno staviti u granice validnih vrijednosti piksela (uobičajeno od 0 do 255).

8.1.2. Računanje preostalih piksela

Evaluacija preostalih piksela se odvija na skaliranoj slici u 7×7 regiji, s novim pikselom koji se računa u sredini. Računanje se oslanja na originalne piksele slike ili na prethodno izračunate dijagonalne piksele.
Algoritam bira jedan od tri slučaja:

1. Horizontalni rub → interpolira u vertikalnom smjeru
2. Vertikalni rub → interpolira u horizontalnom smjeru
3. Glatko područje → interpolira u oba smjera te množi s vrijednostima težina

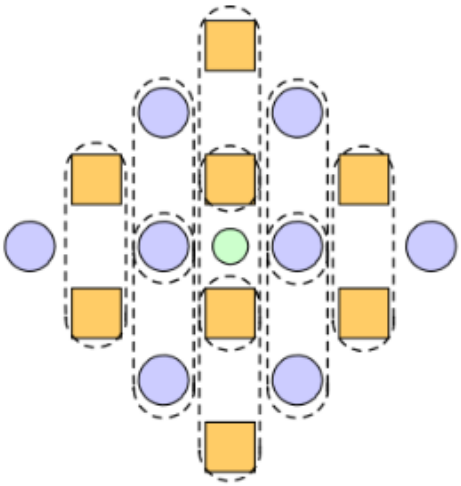
Računanje jačine horizontalnog/vertikalnog ruba

Neka je $d1$ suma svih rubova u horizontalnom smjeru, a $d2$ u vertikalnom smjeru. Uzmemo 7×7 regiju dijamantnog oblika centriranog oko piksela kojeg računamo. Regija se sastoji od vrijednosti piksela iz originalne slike te dobivenih vrijednosti dijagonalnih piksela. Da izračunamo $d1$, uzmimo sumu apsolutne vrijednosti razlika između horizontalnih rubova uzimanjem ovih vrijednosti:

$$|P(x+1,y-2) - P(x-1,y-2)| +$$
$$|P(x+2,y-1) - P(x,y-1)| + |P(x,y-1) - P(x-2,y-1)| +$$
$$|P(x+3,y) - P(x+1,y)| + |P(x+1,y) - P(x-1,y)| + |P(x-1,y) - P(x-3,y)| +$$
$$|P(x+2,y+1) - P(x,y+1)| + |P(x,y+1) - P(x-2,y+1)| +$$
$$|P(x+1,y+2) - P(x-1,y+2)|$$

Da izračunamo $d2$, uzmimo sumu apsolutne vrijednosti razlika između vertikalnih rubova uzimanjem ovih vrijednosti:

$$
\begin{aligned}
&|P(x-2,y+1)-P(x-2,y-1)|+\\
&|P(x-1,y+2)-P(x-1,y)|\\
&\qquad +|P(x-1,y)\\
&\qquad -P(x-1,y-2)|+\\
&|P(x,y+3)-P(x,y+1)|\\
&\qquad +|P(x,y+1)-P(x,y-1)|\\
&\qquad +|P(x,y-1)-P(x,y-3)|\\
&\qquad +\\
&|P(x+1,y+2)-P(x+1,y)|\\
&\qquad +|P(x+1,y)\\
&\qquad -P(x+1,y-2)|+\\
&|P(x+2,y+1)-P(x+2,y-1)|
\end{aligned}
$$



Interpoliranje piksela

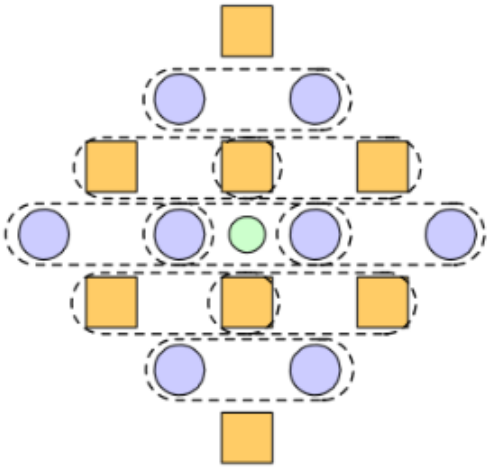
Ako je $\frac{1+d1}{1+d2} > 1.15$ onda je rub u horizontalnom smjeru, a ako je $\frac{1+d2}{1+d1} > 1.15$ onda je u vertikalnom smjeru. Inače, se nalazimo u glatkom području. Za zaobilaženje dijeljenja i operacija decimalnim brojevima, ovo se može zapisati kao $100 \cdot (1+d1) > 115 \cdot (1+d2)$ i $100 \cdot (1+d2) > 115 \cdot (1+d1)$.

Horizontalni rub

Slika 21. Horizontalni smjer (Preuzeto s: https://en.wikipedia.org/wiki/Directional_Cubic_Convolution_Interpolation)

Slika 22. Vertikalni smjer (Preuzeto s: https://en.wikipedia.org/wiki/Directional_Cubic_Convolution_Interpolation)

Za izračunavanje piksela u horizontalnom rubu trebamo interpolirati u vertikalnom smjeru, uzimanjem samo stupca centriranog na pikselu.



$$izračunati\ piksel = \frac{-1 \cdot P(x, y-3) + 9 \cdot P(x, y-1) + 9 \cdot P(x, y+1) - 1 \cdot P(x, y+3)}{16}$$

Vrijednosti piksela je potrebno staviti u granice validnih vrijednosti piksela (uobičajeno od 0 do 255).

Vertikalni rub

Za izračunavanje piksela u vertikalnom rubu trebamo interpolirati u horizontalnom smjeru, uzimanjem samo retka centriranog na pikselu.

$$izračunati\ piksel = \frac{-1 \cdot P(x-3, y) + 9 \cdot P(x-1, y) + 9 \cdot P(x+1, y) - 1 \cdot P(x+3, y)}{16}$$

Vrijednosti piksela je potrebno staviti u granice validnih vrijednosti piksela (uobičajeno od 0 do 255).

Glatko područje

U glatkom području, jačina horizontalnog ruba utječe na težinu piksela dobivenog vertikalnim rubom, a jačina vertikalnog ruba utječe na težinu piksela dobivenog horizontalnim rubom.

$$w1 = \frac{1}{1 + d1^5}$$

$$w2 = \frac{1}{1 + d2^5}$$

$$težina1 = \frac{w1}{w1 + w2}$$

$$težina2 = \frac{w2}{w1 + w2}$$

$$HorizontalniPixel = \frac{-1 \cdot P(x-3,y) + 9 \cdot P(x-1,y) + 9 \cdot P(x+1,y) - 1 \cdot P(x+3,y)}{16}$$

$$VertikalniPixel = \frac{-1 \cdot P(x,y-3) + 9 \cdot P(x,y-1) + 9 \cdot P(x,y+1) - 1 \cdot P(x,y+3)}{16}$$

$$izračunatiPixel = VertikalniPixel \cdot težina1 + HorizontalniPixel \cdot težina2$$

Vrijednosti piksela je potrebno staviti u granice validnih vrijednosti piksela (uobičajeno od 0 do 255).

8.2. Primjer DCCI

Originalna slika je jednaka kao u prethodna dva primjera, a dvostruko uvećana slika pomoću DCCI izgleda ovako:

Slika 23. Uvećani smješko (DCCI)

Njezina glatkoća je još veća nego što je to kod bikubične interpolacije.

9. Mean squared error (MSE)

Jedan od načina na koji možemo mjeriti razlike između slika je Mean squared error. On mjeri srednju vrijednost pogreške, tj. srednju vrijednost kvadriranih razlika između procijenjenih vrijednosti i onoga sto se procjenjuje. MSE je uvijek pozitivan i stogo veći od 0 jer se ne očekuju informacije koje bi mogle dovesti do bolje procjene. Vrijednosti bliže nuli su bolje.

Kako bi odredili MSE kvadriramo razliku između originalnih i procijenjenih piksela, zatim sumiramo po svim pikselima i na kraju podijelimo s ukupnim brojem piksela, odnosno računamo prema formuli:

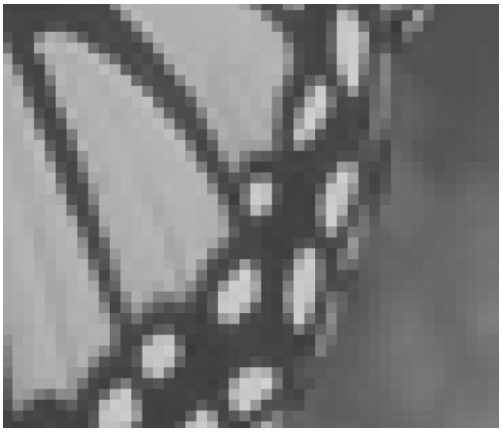
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - {Y'}_i)^2$$

pri čemu je Y_i vrijednost originalnih piksela, ${Y'}_i$ vrijednost procijenjenih piksela, a n ukupan broj piksela.

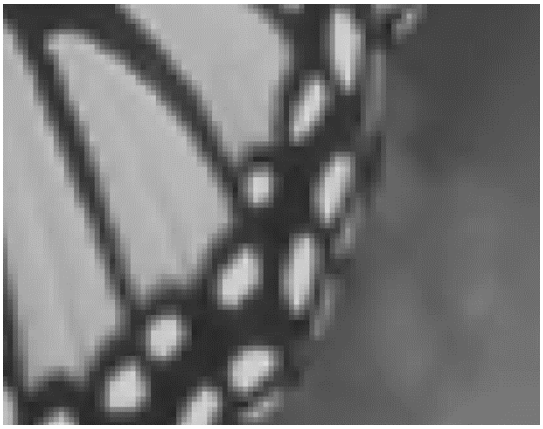
Osim za mjerenje razlike među slikama, MSE se može koristiti i za razna druga mjerenja, poput mjerenja grešaka u statistici. Dva ili više statističkih modela se mogu uspoređivati koristeći njihove MSE kao mjeru koliko dobro objašnjavaju zadani skup opažanja. Najbolji od njih je onaj čiji je rezultat najbliži nuli.

10. Usporedba različitih načina interpolacije

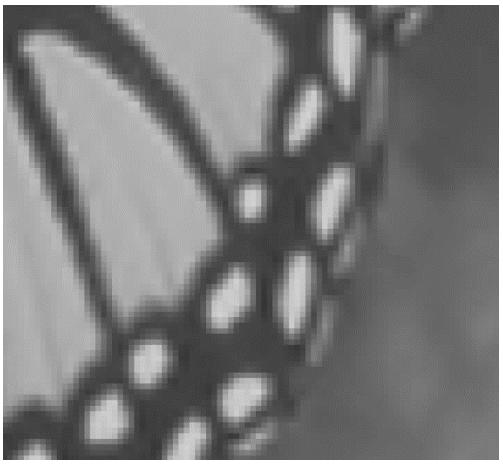
Za usporedbu prethodno tri opisana algoritma interpolacije korišten je detalj slike leptira. Uvećanjem slika pikseli dolaze više do izražaja pa se razlika između interpolacija može najbolje uočiti. Odabrana slika je duplo uvećana bez i sa korištenjem opisanih interpolacija te je dobiven sljedeći rezultat:



Slika 24. Uvećanje bez interpolacija



Slika 25. Uvećanje bilinearnom interpolacijom



Slika 26. Uvećanje bikubičnom interpolacijom

Slika 27. Uvećanje pomoću DCCI

Možemo zaključiti da je slika koja je uvećana bez primjene interpolacija najlošije kvalitete. Korištenjem interpolacija dobivamo glađe slike, pri čemu je bilinearna najjednostavnija od ovdje navedenih i ima najmanju glatkoću. Ona uzima u obzir 4 najbliža piksela, a bikubična 16 piksela, zbog čega je bikubična sporija od bilinearne, ali daje bolju kvalitetu slike pa se odabire u slučajevima kad brzina nije glavni kriterij. Pomoću DCCI dobivamo najglađu sliku jer, osim u smjeru osi x i y, interpolira i u dijagonalnom smjeru, pri čemu se odabire onaj smjer koji bolje odgovara u izračunavanju novog piksela, a način odabira opisan je ranije.

10.1. Razlika između interpolacija

Kako je slika na računalu spremljena kao matrica piksela, da bi dobili razliku među slikama trebamo oduzeti vrijednosti piksela koji se nalaze na istim pozicijama. Uzmemo apsolutnu vrijednost dobivene razlike piksela, jer pikseli ne smiju biti negativni brojevi. Kada napravimo razliku između bilinearne interpolacije i DCCI dobijemo:

Slika 28. Razlika bilinearne interpolacije i DCCI

dok razlika bikubične interpolacije i DCCI izgleda ovako:

Slika 29. Razlika bikubične interpolacije i DCCI

Što su pikseli tamniji, razlika između slika je manja, dok svjetliji pikseli predstvaljaju veću razliku. Možemo primijetiti da je razlika između bilinearne interpolacije i DCCI veća od razlike između bikubične interpolacije i DCCI, a tome je tako jer je bilinearna interpolacija lošija od bikubične interpolacije. Do istog zaključka možemo doći i uspoređujući njihove MSE: MSE za bilinearnu interpolaciju i DCCI iznosi 204.84, dok za bikubičnu interpolaciju i DCCI iznosi 200.52.

11. Zaključak

Razvoj računalne tehnologije svakodnevno mijenja svijet kakav poznajemo pa se tako promijenio i način razvoja slika. Umjesto fotografiranja starim fotoaparatima i nestrpljivog iščekivanje izrade slika, danas ih na digitalnim fotoaparatima i mobitelima možemo pregledati odmah nakon njihovog nastanka. Zbog toga se slike sve češće pohranjuju u digitalnom obliku, a foto albumi odlaze u povijest. Samim time dolazimo do potrebe skaliranja ili uljepšavanja slika na što utječe interpolacija.

Kroz ovaj rad opisane su tri metode interpolacije, pri čemu bilinearna i bikubična interpolacija pripada neprilagodljivim algoritmima, a DCCI pripada prilagodljivim algoritmima interpolacije. Možemo zaključiti da je DCCI najbolji, ali i najsloženiji, način interpolacije od svih ovdje navedenih. Pošto pripada prilagodljivim načina interpoliranja, bira smjer u kojem će interpolirati ovisno o intenzitetu okolnih piksela kako bi dobili što bolju sliku.

Radeći na ovom radu zaključila sam da je interpolacija slike vrlo korisna za uvećavanje slike uz zadržavanje visoke kvalitete. Naučila sam kako računalo radi sa slikama i na koji način ih pohranjuje te saznala da postoje razni efikasni algoritmi za interpolaciju slike koji sačuvaju njezin originalni izgled.

12. Popis slika

Slika 1. Interpolacija slike 5
Slika 2. Koordinatni sustav slike 7
Slika 3. Jednobojna slika 8
Slika 4. RGB slika 8
Slika 5. Primjer RGB piksela 9
Slika 6. RGB slika razdvojena na R, G i B sloj 9
Slika 7. Interpolacijski polinom 10
Slika 8. Bazni polinomi 13
Slika 9. Lagrangeov polinom 13
Slika 10. Bilinearna interpolacija 14
Slika 11. 3x3 piksela 15
Slika 12. Smješko 15
Slika 13. 5x5 piksela (bilinearna interpolacija) 16
Slika 14. Uvećani smješko (bilinearna interpolacija) 16
Slika 15. Bikubična interpolacija 17
Slika 16. 5x5 piksela (bikubična interpolacija) 18
Slika 17. Uvećani smješko (bikubična interpolacija) 18
Slika 18. Algoritam DCCI 19
Slika 19. Smjer gore-desno 20
Slika 20. Smjer dolje-desno 20
Slika 21. Horizontalni smjer 22
Slika 22. Vertikalni smjer 22
Slika 23. Uvećani smješko (DCCI) 25
Slika 24. Uvećanje bez interpolacija 27
Slika 25. Uvećanje bilinearnom interpolacijom 27
Slika 26. Uvećanje bikubičnom interpolacijom 26
Slika 27. Uvećanje pomoći DCCI 26
Slika 28. Razlika bilinearne interpolacije i DCCI 28
Slika 29. Razlika bikubične interpolacije i DCCI 28

13. Popis tablica

Tablica 1. Točke i njihove funkcijske vrijednosti *12*

14. Popis priloga

Prilog priložen uz ovaj rad je CD na kojem se, osim ovog rada, nalaze tri python datoteke: *BilinearnaInterpolacija.py*, *BikubicnaInterpolacija.py* te *DCCI.py*, i slika leptir.jpg koja je potrebna za *DCCI.py*. Python datoteke su isprogramirane za potrebe ovog rada te u njemu i korištene za primjere interpolacija i njihovu usporedbu.

BilinearnaInterpolacija.py

```
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy import misc

def bilinear_interpolation(img1D,size):
    #originalna velicina slike
    orig_size=img1D.shape[0]
    #pripremi prazan niz nove izabrane velicine u kojima se nalaze (r,g,b)[pixeli]
    out_img=np.zeros((size), dtype=(np.int32))
    #idi po svakom pizelu

    ratio=(orig_size-1)/(size-1)#omjer sirine intervala

    for i in range(size-1):
        Coord=math.floor(i*ratio)

        alpha=i*ratio-Coord

        if alpha>1: print('alpha',alpha)
        out_img[i]=img1D[Coord]*(1.-alpha)+alpha*img1D[Coord+1]
        out_img[-1]=img1D[-1]
    return out_img

def MSE(slika1, slika2):
    error=np.sum((slika1.astype("float")-slika2.astype("float"))**2)
    error/=float(slika1.shape[0]*slika1.shape[1])
    return error

#velicina slike
nx=5
ny=5

#priprema originalne slike
mat=np.array([[100,200,230],[230,200,100],[90,100,80]],dtype='B')

img=np.zeros((3,3),dtype='B')
img[:,0]=mat[0]
img[:,1]=mat[1]
img[:,2]=mat[2]

#priprema privremene matrice koja cuva sliku interpoliranu po jednoj osi
img2=np.zeros((ny,img.shape[1]),dtype='B')

#interpoliraj po jednoj osi, tako da ides jedan po jedan
for i in range(img.shape[1]):
    img2[:,i]=bilinear_interpolation(img[:,i],ny)
#pripremi krajnji array
img3=np.zeros((ny,nx),dtype='B')
#interpoliraj po drugoj osi vec interpolirane vrijednosti iz privremene matrice
for i in range(ny):
    img3[i]=bilinear_interpolation(img2[i],nx)
#prikazi
plt.figure(figsize=(10,4))
f,ax=plt.subplots(3)
#prikaz prve slike
ax[0].imshow(img3, cmap='gray', vmin=0, vmax=255)
#prikaz druge slike
```

```
img_i=misc.imresize(img, (ny, nx), interp='bilinear')# resize(img, (ny, nx))
ax[1].imshow(img_i, cmap='gray', vmin=0, vmax=255)
#prikaz razlike
img_r=np.abs(img3.astype(np.int)-img_i.astype(np.int)).astype(np.uint8)
ax[2].imshow(img_r, cmap='gray', vmin=0, vmax=255)

print(MSE(img3, img_i))
plt.imsave("slika_orig.png",img)
plt.imsave("slika_inter.png",img3)
```

BikubičnaInterpolacija.py

```
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy import misc

def CRspline(x1, x2, x3, x4, d):
    return np.uint8(0.5*(((-int(x1)+3*int(x2)-3*int(x3)+int(x4))*d+(2*int(x1)-5*int(x2)+4*int(x3)-int(x4)))*d+(-
int(x1)+int(x3)))*d+int(x2))

def bicubic_interpolation(img1D,size):
    #originalna velicina slike
    orig_size=img1D.shape[0]
    #pripremi prazan niz nove izabrane velicine u kojima se nalaze (r,g,b)[pixeli]
    out_img=np.zeros((size), dtype=(np.int32))

    ratio=(orig_size-1)/(size-1)#omjer sirine intervala

    #idi po svakom pizelu
    for i in range(size-1):
        #dobij aproksimiranu koordinatu pixela ovisno o skaliranju slike
        Coord=math.floor(i*ratio)

        #racunanje 4 tocke za redom
        #prve dvije oko aproksimirane tocke
        x1=Coord
        x2=Coord+1
        #druga tocka ispred
        x0=x1-1
        #druga tocka iza
        x3=x2+1

        #racunanje decimalnog dijela aproksimirane tocke
        decCoord = i*ratio - Coord

        #ogranicavanje tocaka na interval od 0 do velicina niza-1
        x0=max(0,min(x0,orig_size-1))
        x1=max(0,min(x1,orig_size-1))
        x2=max(0,min(x2,orig_size-1))
        x3=max(0,min(x3,orig_size-1))

        out_img[i]=CRspline(img1D[x0], img1D[x1], img1D[x2], img1D[x3], decCoord)
        out_img[-1]=img1D[-1]
    return out_img

def MSE(slika1, slika2):
    error=np.sum((slika1.astype("float")-slika2.astype("float"))**2)
    error/=float(slika1.shape[0]*slika1.shape[1])
    return error

#velicina slike
nx=5
ny=5

#priprema originalne slike
mat=np.array([[100,200,230],[230,200,100],[90,100,80]],dtype='B')

img=np.zeros((3,3),dtype='B')
img[:,0]=mat[0]
```

```
img[:,1]=mat[1]
img[:,2]=mat[2]

#priprema privremene matrice koja cuva sliku interpoliranu po jednoj osi
img2=np.zeros((ny,img.shape[1]),dtype='B')

#interpoliraj po jednoj osi, tako da ides jedan po jedan
for i in range(img.shape[1]):
    img2[:,i]=bicubic_interpolation(img[:,i],ny)
#pripremi krajnji array
img3=np.zeros((ny,nx),dtype='B')
#interpoliraj po drugoj osi vec interpolirane vrijednosti iz privremene matrice
for i in range(ny):
    img3[i]=bicubic_interpolation(img2[i],nx)
#prikazi
plt.figure(figsize=(10,4))
f,ax=plt.subplots(3)
#prikaz prve slike
ax[0].imshow(img3, cmap='gray', vmin=0, vmax=255)
#prikaz druge slike
img_i=misc.imresize(img, (ny, nx), interp='bicubic')# resize(img, (ny, nx))
ax[1].imshow(img_i, cmap='gray', vmin=0, vmax=255)
#prikaz razlike
img_r=np.abs(img3.astype(np.int)-img_i.astype(np.int)).astype(np.uint8)
ax[2].imshow(img_r, cmap='gray', vmin=0, vmax=255)

print(MSE(img3, img_i))
plt.imsave("slika_orig.png",img)
plt.imsave("slika_inter.png",img_i)
```

DCCI.py

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import misc
from skimage import io

#### DCCI ######
def detectDirect(A, tip, k, T):
    if (tip==1):
        #45 degree diagonal direction
        t1=np.abs(A[2][0]-A[0][2])
        t2=np.abs(A[4][0]-A[2][2])+np.abs(A[2][2]-A[0][4])
        t3=np.abs(A[6][0]-A[4][2])+np.abs(A[4][2]-A[2][4])+np.abs(A[2][4]-A[0][6])
        t4=np.abs(A[6][2]-A[4][4])+np.abs(A[4][4]-A[2][6])
        t5=np.abs(A[6][4]-A[4][6])
        d1=t1+t2+t3+t4+t5
        #135 degree diagonal direction
        t1=np.abs(A[0][4]-A[2][6])
        t2=np.abs(A[0][2]-A[2][4])+np.abs(A[2][4]-A[4][6])
        t3=np.abs(A[0][0]-A[2][2])+np.abs(A[2][2]-A[4][4])+np.abs(A[4][4]-A[6][6])
        t4=np.abs(A[2][0]-A[4][2])+np.abs(A[4][2]-A[6][4])
        t5=np.abs(A[4][0]-A[6][2])
        d2=t1+t2+t3+t4+t5
    else:
        #horizontal direction
        t1=np.abs(A[0][1]-A[0][3])+np.abs(A[2][1]-A[2][3])+np.abs(A[4][1]-A[4][3])
        t2=np.abs(A[1][0]-A[1][2])+np.abs(A[1][2]-A[1][4])
        t3=np.abs(A[3][0]-A[3][2])+np.abs(A[3][2]-A[3][4])

        d1=t1+t2+t3
        #vertical direction
        t1=np.abs(A[1][0]-A[3][0])+np.abs(A[1][2]-A[3][2])+np.abs(A[1][4]-A[3][4])
        t2=np.abs(A[0][1]-A[2][1])+np.abs(A[2][1]-A[4][1])
        t3=np.abs(A[0][3]-A[2][3])+np.abs(A[2][3]-A[4][3])
        d2=t1+t2+t3

    #compute the weight vector
    w1=1+d1**k
    w2=1+d2**k
    w=(1/w1,1/w2)
    #compute the directional index
    n=3
    if (1+d1)/(1+d2)>T:
        n=1
    elif (1+d2)/(1+d1)>T:
        n=2
    return (w, n)

def pixelValue(A, tip, w, n):
    f=np.divide((-1, 9, 9, -1), 16)
    if tip==1:
        v1=(A[6][0], A[4][2], A[2][4], A[0][6])
        v2=(A[0][0], A[2][2], A[4][4], A[6][6])
    else:
        v1=(A[3][0], A[3][2], A[3][4], A[3][6])
        v2=(A[0][3], A[2][3], A[4][3], A[6][3])
    if n==1:
        return np.sum(v2*f)
    elif n==2:
```

```
        return np.sum(v1*f)
    else:
        p1=np.sum(v2*f)
        p2=np.sum(v1*f)
        return (w[0]*p1+w[1]*p2)/(w[0]+w[1])


#k=5, T=1.15
def DCCI(img, k, T):
    sizex=img.shape[0]*2-1
    sizey=img.shape[1]*2-1
    #Initialize the output image
    out_img=np.zeros((sizex, sizey), dtype=(np.float64))
    for i in range (0,img.shape[0]-1):
        for j in range (0,img.shape[1]-1):
            out_img[i*2][j*2]=img[i][j]
    #Do the cubic convolution interpolation
    for i in range (3, sizex-3, 2):
        for j in range (3, sizey-3, 2):
            temp=np.zeros((7, 7), dtype=(np.float64))
            for k in range(-3, 4):
                for l in range(-3, 4):
                    temp[k+3][l+3]=out_img[i+k][j+l]
            w, n=detectDirect(temp, 1, k, T)
            out_img[i][j]=pixelValue(temp, 1, w, n)
    for i in range(4, sizex-4, 2):
        for j in range(3, sizey-3, 2):
            temp=np.zeros((5, 5), dtype=(np.float64))
            for k in range(-2, 3):
                for l in range(-2, 3):
                    temp[k+2][l+2]=out_img[i+k][j+l]
            w, n=detectDirect(temp, 2, k, T)
            temp=np.zeros((7, 7), dtype=(np.float64))
            for k in range(-3, 4):
                for l in range(-3, 4):
                    temp[k+3][l+3]=out_img[i+k][j+l]
            out_img[i][j]=pixelValue(temp, 2, w, n)
    for i in range(3, sizex-3, 2):
        for j in range(4, sizey-4, 2):
            temp=np.zeros((5, 5), dtype=(np.float64))
            for k in range(-2, 3):
                for l in range(-2, 3):
                    temp[k+2][l+2]=out_img[i+k][j+l]
            w, n=detectDirect(temp, 3, k, T)
            temp=np.zeros((7, 7), dtype=(np.float64))
            for k in range(-3, 4):
                for l in range(-3, 4):
                    temp[k+3][l+3]=out_img[i+k][j+l]
            out_img[i][j]=pixelValue(temp, 3, w, n)
    return out_img

#### DCCI zavrsava #####

img=io.imread("leptir.jpg", True)
img_i=misc.imresize(img, (256, 256), interp='bicubic')

imgr=DCCI(img_i, 5, 1.15)

f,ax=plt.subplots(2)
ax[0].imshow(img_i, cmap='gray', vmin=0, vmax=255)
```

```
ax[1].imshow(imgr, cmap='gray', vmin=0, vmax=255)

plt.imsave("dcci.png",imgr, vmin=0, vmax=255)
plt.imsave("orig.png",img_i, vmin=0, vmax=255)
```

15. Literatura

2014. *Aproksimacija.* 17. 4. https://hr.wikipedia.org/wiki/Aproksimacija.

n.d. »Aproksimacija i interpolacija.« Pokušaj pristupa 1. 7 2018. https://www.fsb.unizg.hr/mat-4/OldWeb/4.pdf.

2017. *Bicubic interpolation.* 10. 10. https://en.wikipedia.org/wiki/Bicubic_interpolation.

2018. *Bilinear interpolation.* 9. 6. https://en.wikipedia.org/wiki/Bilinear_interpolation.

2013. »Grafika.« U *Multimedijski sustavi*, autor Nataša Hoić Božić. Rijeka: Sveučilište u Rijeci, Odjel za informatiku.

Breeuwsma, Paul. n.d. *Cubic interpolation.* Pokušaj pristupa 28. 6 2018. https://www.paulinternet.nl/?page=bicubic.

2017. *Dekadski brojevni sustav.* 10. 10. https://hr.wikipedia.org/wiki/Dekadski_brojevni_sustav.

n.d. *Digital image interpolation.* Pokušaj pristupa 25. 6 2018. https://www.cambridgeincolour.com/tutorials/image-interpolation.htm.

2014. *Digitalna slika.* 30. 11. https://bs.wikipedia.org/wiki/Digitalna_slika.

2017. *Directional Cubic Convolution Interpolation.* 29. 8. https://en.wikipedia.org/wiki/Directional_Cubic_Convolution_Interpolation.

Eddins, Steve. 2011. *Digital image representation.* 26. 8. https://blogs.mathworks.com/steve/2011/08/26/digital-image-processing-using-matlab-digital-image-representation/.

2013. *Heksadekadski brojevni sustav.* 4. 10. https://hr.wikipedia.org/wiki/Heksadekadski_brojevni_sustav.

2013. *Interpolacija.* 3. 11. https://hr.wikipedia.org/wiki/Interpolacija.

n.d. »Interpolacija i aproksimacija funkcija.« Pokušaj pristupa 1. 7 2018. https://element.hr/artikli/file/1314.

2018. *Linear interpolation.* 9. 3. https://en.wikipedia.org/wiki/Linear_interpolation.

2018. *Mean squared error.* 15. 6. https://en.wikipedia.org/wiki/Mean_squared_error#Mean.

Philippe Thévenaz, Thierry Blu and Michael Unser. n.d. »Image Interpolation and Resampling.« Pokušaj pristupa 25. 6 2018. http://bigwww.epfl.ch/publications/thevenaz9901.pdf.

Sheets, Kristopher. n.d. *What is a digital Image.* Pokušaj pristupa 30. 6 2018. https://sites.google.com/site/learnimagej/image-processing/what-is-a-digital-image.