Primjena interpolacije u digitalnoj obradi slika

Josipović, Iwan

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Department of Mathematics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:126:999137

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Repository of School of Applied Mathematics and Computer Science

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike

Iwan Josipović

Primjena interpolacije u digitalnoj obradi slika

Završni rad

Osijek, 2018.

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike

Iwan Josipović

Primjena interpolacije u digitalnoj obradi slika

Završni rad

Mentor: izv. prof. dr. sc. Domagoj Matijević

Osijek, 2018.

Sažetak

U ovom ćemo radu definirati pojam interpolacije za neki dani skup podataka. Obraditi ćemo neke od 1-D metode interpolacije, a to su: metoda najbližeg susjedstva, linearna interpolacija i kubična interpolacija. Upoznati ćemo se s osnovama interpolacije kroz jednostavne primjere i stečeno znanje primjenjivati na sve kompleksnije primjere. Također ćemo obraditi i metode 2-D interpolacije, a to su: metoda najbližeg susjedstva, bilinearna interpolacija i bikubična interpolacija. Pretežno ćemo se baviti problemom uvećavanja slika, ali također ćemo spomenuti da se iste metode koriste kod rotacije slika, smanjivanja slika, iskrivljavanja slika i slično. Metode ćemo analizirati na zajedničkom primjeru, te usporedit dobivene rezultate kako bismo zaključili što u praksi dobivamo uporabom istih tih metoda. Ovisno o rezultatima koje nam daju pojedine metode, zaključiti ćemo gdje je najbolje primjenjivati te metode i zašto.

Ključne riječi

Interpolacija, obrada slike, metoda najbližeg susjedstva, linearna, kubična, bilinearna, bikubična

Summary

In this work we will define interpolation for a given data set. We will cover some of the 1-D interpolation methods such as: nearest neighbour interpolation, linear interpolation and cubic interpolation. We will learn about the basics of interpolation through simple examples, and use the acquired knowledge on more complex examples. Also we will cover some 2-D interpolation methods such as: neirest neighbour interpolation, bilinear interpolation and bicubic interpolation. Our main focus will be image scaling, but we will also mention that the same interpolation methods are used with image rotation, downsizing, distortion etc. We will analize our methods on a common example, and compare the results so we can deduce the benefits of those methods. Depending on the results, we will deduce why and where those methods should be used.

Key words

interpolation, image processing, nearest neighbour, linear, cubic, bilinear, bicubic

Sadržaj

Uvod **i**

1 Jednodimenzionalna interpolacija **1**

1.1 Pojam interpolacije i uvođenje pojmova 1

1.2 Lokalno konstantna interpolacija 2

1.3 Linearni interpolacijski spline 5

1.4 Kubični interpolacijski spline 8

2 Dvodimenzionalna interpolacija **10**

2.1 Metoda najbližeg susjedstva 10

2.2 Bilinearna interpolacija 12

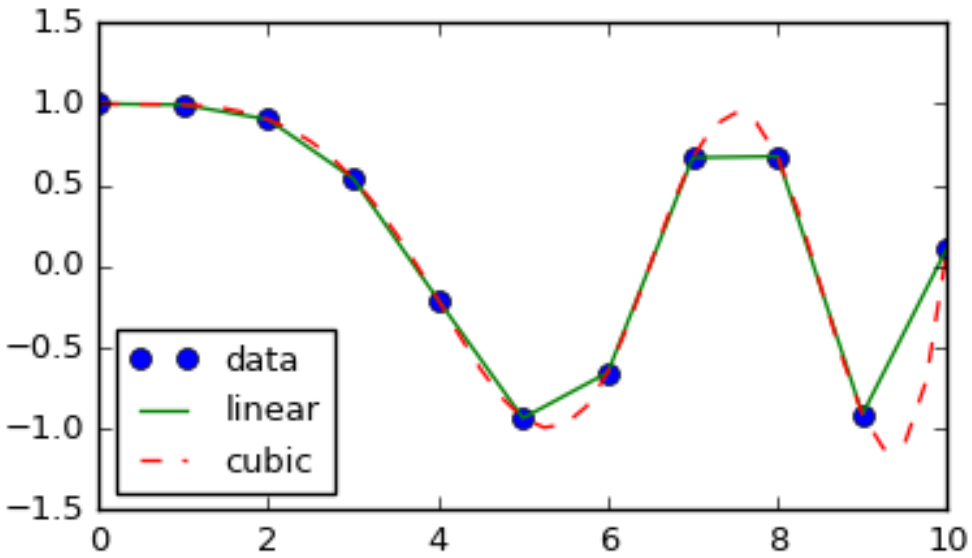
2.3 Bikubična interpolacija 15

2.4 Usporedba i primjene metoda interpolacije 18

Literatura **23**

Uvod

Interpolacija u matematici predstavlja određivanje nepoznatih vrijednosti neke veličine s pomoću poznatih vrijednosti u nekom intervalu u kojem su poznate zakonitosti njezinih promjena. Na primjer, poznato nam je koliko košta prirediti večeru za 10, 50 i 100 ljudi, ali na poziv za večeru odazvalo se 75 ljudi, ukoliko nas zanima cijena večere za tih 75 ljudi možemo iskoristiti podatke koje znamo i na temelju toga donijeti procjenu cijene. Već oko 300. godine prije Krista u antičkoj Grčkoj koristili su ne samo linearnu interpolaciju, nego i kompleksnije oblike interpolacije kako bi predvidjeli poziciju Sunca, Mjeseca i planeta za koje su znali.



Primjer interpolacije koristeći linearnu interpolaciju i kubičnu interpolaciju

Mi ćemo interpolaciju koristiti u digitalnoj obradi slika. Interpolacija nam je potrebna jer se npr. prilikom rotacije slike i uvećanja slike pojavljuju pikseli čije vrijednosti ne znamo, stoga ih moramo interpolirati. Obraditi ćemo tri metode interpolacije. U sve tri metode funkcija koju koristimo za interpolaciju je polinom. Kod metode najbližeg susjedstva to je konstantan polinom, kod bilinearne interpolacije je polinom prvog stupnja, a kod bikubične interpolacije koristimo polinom trećeg stupnja. Različite metode daju različite rezultate pa ćemo na zajedničkim primjerima uočiti kvalitete i nedostatke svake metode, te spomenuti gdje se koja metoda koristi.

1 Jednodimenzionalna interpolacija

1.1 Pojam interpolacije i uvođenje pojmova

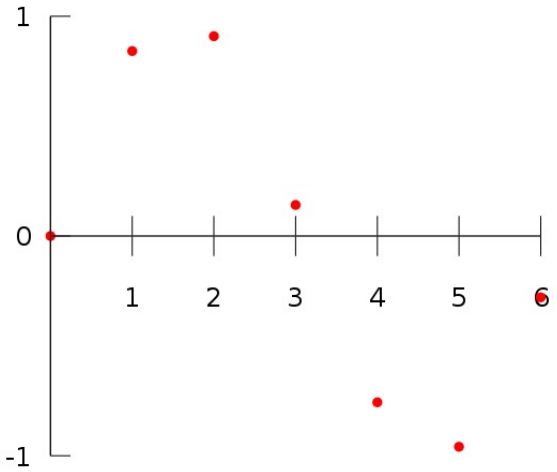
Problem interpolacije je proces kojim za konačan skup točaka: $x_0 < x_1 < \cdots < x_n$ i njihove pripadne funkcijske vrijednosti u nama nepoznatoj funkciji f: $f(x_0), f(x_1), \ldots, f(x_n)$, pronalazimo, po mogućnosti, što jednostavniju funkciju g definiranu na $[x_0, x_n]$, koja dobro aproksimira funkciju f. Funkcija g treba zadovoljavati sljedeći uvjet:

$$g(x_i) = f(x_i), \quad i = 0, 1, \ldots, n \tag{1}$$

Funkcija g često je polinom ili trigonometrijska, također može biti eksponencijalna, racionalna, konstantna itd. Tako dobivenu funkciju g koristiti ćemo za procjenjivanje vrijednosti funkcije f na $[x_0, x_n]$. U daljnjem tekstu koristit ćemo oznaku f za funkciju čiju vrijednost želimo interpolirati, s g funkciju dobivenu interpolacijom, a čvorovima interpolacije ćemo nazivati točke čije funkcijske vrijednosti znamo i označiti ih s $x_0, x_1, \ldots, x_n$, dok ćemo pripadne funkcijske vrijednosti označiti redom s $y_0 = f(x_0), y_1 = f(x_1), \ldots, y_n = f(x_n)$. Kako bi bili konzistentni, sve metode interpolacije analizirati ćemo na istom skupu podataka:

x	$f(x)$
0	0
1	0,8415
2	0,9093
3	0,1411
4	−0,7568
5	−0,9589
6	−0,2794

Tablica 1: *Čvorovi interpolacije i njihove pripadne funkcijske vrijednosti*



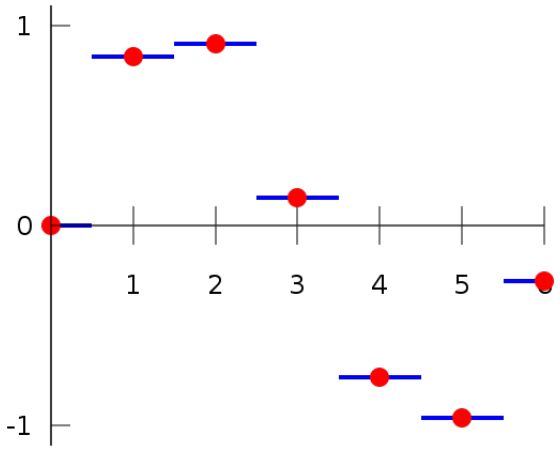
Slika 1: *Grafički prikaz podataka iz Tablice 1.*

1.2 Lokalno konstantna interpolacija

Jedna od najjednostavnijih i najbržih metoda interpolacije korištena u obradi slika je lokalno konstantna interpolacija koju zovemo još i metoda najbližeg susjedstva. U ovoj metodi funkcija g u točki x poprima vrijednost $f(x_i)$ gdje je x_i čvor interpolacije, koji je euklidski najmanje udaljen od x. Lako možemo zaključiti da će g biti oblika:

$$g(x) = \begin{cases} f(x_i), & \text{if } x \in \left[x_i, \frac{(x_i+x_{i+1})}{2}\right\rangle \\ f(x_{i+1}), & \text{if } x \in \left[\frac{(x_i+x_{i+1})}{2}, x_{i+1}\right], \quad i = 0, 1, \ldots, n-1 \end{cases}$$

Iz grafa funkcije g vidljivo je da je ona konstanta na segmentima oko čvorova interpolacija, te upravo zato ovu metodu nazivamo lokalno konstantnom interpolacijom.



Slika 2: *Graf funkcije g*

Na sljedećem ćemo primjeru ilustrirati korištenje lokalno konstantne interpolacije, na uvećanje slike. Slike su reprezentirane kao 2D-polja, pri čemu element tog polja na poziciji (i, j) sadrži informaciju o boji piksela na (i, j)-toj poziciji slike. Zbog jednostavnosti u ovom radu nećemo koristiti RGB boje, nego samo nijanse sive. Ovaj primjer, kao i sljedeći, generirani su programskom jeziku Python, koristeći biblioteke "numpy" i "PIL" (Python Imaging Library). Nijanse sive biramo brojem u intervalu $[0, 255]$, gdje je 0 vrijednost crne, a 255 vrijednost bijele. Za sada ćemo se baviti 1-D interpolacijom, stoga slika koju ćemo interpolirati, zapravo će biti 1-D polje piksela.

Primjer 1. *Generirajmo sliku dimenzija* 1×2*, s vrijednostima piksela redom* 100 *i* 200*. Uvećajmo dobiveno sliku 2 puta, zatim ju interpolirajmo koristeći lokalno konstantnu interpolaciju.*

Pripadni Python kod za generiranje slike:

```
#Potrebne biliioteke
import numpy as np
from PIL import Image

#Definiramo polje dimenzija 1x2, koje ce reprezentirati piksele
#uint8 je Unsigned integer (0 to 255)
array = np.zeros([1, 2], dtype=np.uint8)

#Postavljanje (i,j)-tog piksela na zadanu vrijednost
array[0][0] = 100
array[0][1] = 200

#Prebacivanje polja u sliku
img = Image.fromarray(array)

#Spremanje dobivene slike
img.save('slika3.png'))
```

Slika 3: *rezultat prethodne Python skripte*

Uvećajemo našu sliku 2 puta, znači da će slika dimenzija 1×2 *postati dimenzija* 2×4*, pri čemu se piksel sa* (i, j)*-te pozicije originala, aproksimativno* [1] *preslikao u poziciju* $(2*i, 2*j)$*, dok su svi ostali pikseli uvećane slike, u koje si nije preslikao niti jedan iz originala, bez ikakvih podataka, pa ćemo ih primjera radi postaviti na crnu. 2-D interpolacijom ćemo se baviti kasnije u radu, stoga ćemo interpolirati samo prvi redak naše* 2×4 *slike:*

Slika 4: *prvi redak uvećane slike prije interpolacije*

Ako redom piksele indeksiramo s lijeva na desno indeksima $0, 1, 2, 3$ *uz prethodno objašnjeni postupak lokalno konstantne interpolacije, vidljivo je da će piksel na indeksu* 1 *poprimiti kao onaj na indeksu* 0*, što će dati:*

[1] aproksimativno jer se kod lokalno konstantne i linearne interpolacije uvijek krajni pikseli preslikavaju u krajne, stoga će se u primjeru 1, piksel na indeksu (0,1) zapravo preslikati u (0,3), a ne u (0,2) kao što je očekivano. Demonstrirat ćemo ovaj princip u daljnim primjerima.

Slika 5: *rezultat nakon prvog koraka interpolacije*

analognim postukom će piksel na indeksu 2 *poprimit vrijednost kao piksel na indeksu* 3*, što daje konačan rezultat:*

Slika 6: *rezultat nakon interpolacije*

Konačni rezultat dobiven je sljedećim python kodom:

```
#Pozivamo resize() metodu, te prosljedujemo joj zeljenu dimenziju, te
#zeljenu metodu interpolacije (u nasem slucaju "nearest neighbour"
#metodu to jest metodu najblizeg susjedstva ili lokalno
#konstantnu interpolaciju.)
img = img.resize((4,1), Image.NEAREST)
img.save('slika6.png')

#Prebacit cemo dobivenu sliku u polje, kako bi vidjeli koje su
#tocno numericke vrijednosti poprimili interpolirani podatci
interpolated_data = np.array(img)

#Ispis navedenog polja
print(interpolated_data)
```

Navedeni kod ispisuje sljedeće polje: [[100 100 200 200]] *, što je u skladu s našim početnim poljem:* [[100 200]] *i opisanim postupkom.*

Uočimo da lokalno konstantna interpolacija ne zahtijeva nikakve izračune, nego naprosto kopira poznate podatke, te je s toga najbrža od svih interpolacija koje ćemo obraditi u ovom radu. Također uočimo da je rezulat u prethodnom primjeru dobra aproksimacija početne slike, ali neće uvijek biti tako, jer nikada nećemo dobiti nijednu drugu boju, osim onih koje su se pojavljivale. Stoga ćemo probati dobiti bolje aproksimacije, s neprekidnim funkcijama.

1.3 Linearni interpolacijski spline

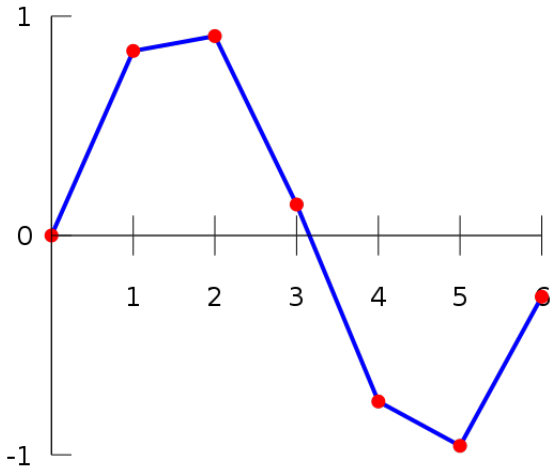
Prethodno smo zaključili da lokalno konstantnom interpolacijom nećemo uvijek dobiti željeni rezultat, npr. nije moguće dobiti nove nijanse boja, stoga ćemo se korisiti linearnim interpolacijskim splineom. Linearni jer će dobivena funkcija biti polinom prvog stupnja, a spline jer interpoliramo samo između svaka dva susjedna čvora. Intuitivno vidimo da će to zapravo biti pravci koji spajaju susjedne čvorove. Funkcija g biti će sljedećeg oblika:

$$g|_{[x_{i-1},x_i]} = g_i, \quad \text{za } i = 1, \ldots, n \tag{2}$$

pri čemu je svaki g_i oblika:

$$g_i(x) = y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x - x_{i-1}), \quad x \in [x_{i-1}, x_i]. \tag{3}$$

Uočimo da će tako konstruirana funkcija uvijek postojati i biti jedinstvena, zbog toga što je pravac jedinstveno određen prolaskom kroz dvije točke. Pogledajmo kako izgleda linearni interpolacijski spline podataka iz Tablice 1.



Slika 7: *linearni interpolacijski spline podataka iz Tablice 1.*

Jasno je vidljivo da će linearni interpolacijski spline dati puno realističniju aproksimaciju podataka koje se nalaze između čvorova. Za razliku od lokalno konstantne u ovoj vrsti interpolacije zapravo izračunavamo nove podatke, pogledajmo to na sljedećem primjeru.

Primjer 2. *Koristeći linearni interpolacijski spline, uvećaj* 3×1 *sliku s pripadnim podatcima:* [[100 150 200]] *, u slike dimenzija:* 4×1 *,* 5×1 *i* 6×1 *i* 7×1*, te ispiši pripadne podatke.*

Slika 8: *Početna slika*

(a) 4x1

(b) 5x1

(c) 6x1

(d) 7x1

Slika 9: Slike dobivene koristeći lineranu interpolaciju

Slike i podaci su rezultat sljedeće python skripte:

```
import numpy as np
from PIL import Image

array = np.zeros([1, 3], dtype=np.uint8)
array[0][0] = 100
array[0][1] = 150
array[0][2] = 200
img = Image.fromarray(array)
img.save('slika8.png')

img4x1 = img.resize((4,1), Image.BILINEAR)
img5x1 = img.resize((5,1), Image.BILINEAR)
img6x1 = img.resize((6,1), Image.BILINEAR)
img7x1 = img.resize((7,1), Image.BILINEAR)
img4x1.save('4x1.png')
img5x1.save('5x1.png')
img6x1.save('6x1.png')
img7x1.save('7x1.png')

pix = np.array(img)
print(pix)
pix4x1 = np.array(img4x1)
print(pix4x1)
pix5x1 = np.array(img5x1)
```

```
print(pix5x1)
pix6x1 = np.array(img6x1)
print(pix6x1)
pix7x1 = np.array(img7x1)
print(pix7x1)
```

Uočimo da samo ovaj put kao argument resize metode odabrali "BILINEAR", što je u stvari linearna interpolacija za dvodimenzionalan slučaj kojim ćemo se baviti kasnije.
Iz dobivenih slika na oko je vidljivo da smo ovim postupkom izračunali nove podatke, to jest, dobili smo boje (nijanse sive) koje se u originalu nisu pojavljivale. Pogledajmo numeričke vrijednosti dobivenih slika, i usporedimo ih s originalom:
Original : [[100 150 200]]
4×1 : [[100 131 169 200]]
5×1 : [[100 120 150 180 200]]
6×1 : [[100 113 138 163 188 200]]
7×1 : [[100 107 129 150 171 193 200]]

Uočimo da su u svim slučajevima, prva i posljednja vrijednost jednake prvoj i posljednjoj iz originala, to je upravo ono što smo spomenuli u fusnoti 1. Također uočimo da su početne vrijednosti bile redom 100 150 200 *, ali da se te vrijednosti pojavljuju samo u slučaju* 5×1 *i* 7×1*, dok se u svim ostalim slučajevima pojavljuju samo* 100 *i* 200*. To je zato što u slučajevima* 5×1 *i* 7×1*, između svaka dva susjedna piksela iz originala, dolazi jednak broj piksela koje moramo izracunati. Npr. u slučaju* 7×1 *dodali smo dva piksela između prvog i srednjeg, i dva između srednjeg i zadnjeg iz originala. U slučajevima* 4×1 *i* 6×1 *, vrijednost* 150 *nije se prikazala, ali stoga je vrijednosti* 150 *pridodana veća težina u izračunu novonastalih susjednih piksela. Uočimo to na slučaju* 4×1 *, novonastale vrijednosti su* 131 *i* 169*, s obzirom da je korištena linearna interpolacija, to jest vrijednost npr.* 131 *je točka na pravcu koji spaja* 100 *i* 150*, pa vidimo da je* 131 *bliži vrijednosti* 150 *jer je veći od polovišta, koji iznosi* 125*. Analogno vrijedi za sve ostale takve slučajeve.*

Prednosti ove interpolacije su velika brzina, malo računskih operacija, te neprekidnost dobivene funkcije aproksimacije g. Nedostatak je to što u čvorovima interpolacije, dobivena funkcija g nije glatka, a to može smanjiti kvalitetu dobivene slike.

1.4 Kubični interpolacijski spline

Linearni interpolacijski spline bio je dobar pokušaj interpolacije, ali on vodi računa samo o dva susjedna piksela, stoga izračunati podaci mogu odstupati od očekivanih. U nastojanju da to popravimo, koristiti ćemo kubični interpolacijski spline, koji u obzir uzima čak četiri susjedna piksela za izračune, čije postojanje znamo iz sljedećeg teorema iz [1].

Teorem 1. *Zadani su podaci* $(x_i, y_i), \quad i = 0, 1, \ldots, n$, *koji zadovoljavaju uvjet* $x_0 < x_1 < \cdots < x_n$. *Tada postoji jedinstveni prirodni kubični interpolacijski spline* C, *pri čemu su polinomi* $C_i, \quad i = 1, \ldots, n$, *zadani s:*

$$C_i(x) = \left(y_{i-1} - s_{i-1}\frac{h_i^2}{6}\right) + b_i(x - x_{i-1}) + \frac{s_{i-1}}{6h_i}(x_i - x)^3 + \frac{s_i}{6h_i}(x - x_{i-1})^3,$$

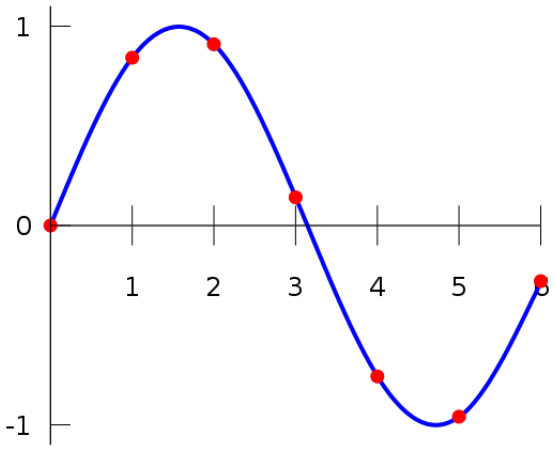
gdje je

$$b_i = d_i - (s_i - s_{i-1})\frac{h_i}{6},$$
$$d_i = \frac{y_i - y_{i-1}}{h_i}, \quad h_i = x_i - x_{i-1},$$

a brojevi $s_i, i = 0, \ldots, n$, *zadovoljavaju sustav jednadžbi*

$$s_{i-1}h_i + 2s_i(h_i + h_{i+1}) + s_{i+1}h_{i+1} = 6(d_{i+1} - d_i), \quad i = 1, \ldots, n-1$$
$$s_0 = s_n = 0.$$

Na sljedećoj slici uočimo da je za razliku od linearnog splinea, kubični ne samo neprekidan, nego i gladak. Neprekidnost i glatkoća vrijede za sve moguće podatke te slijede iz konstrukcije kubičnog interpolacijskog. Više o tome u [1].



Slika 10: *Kubični interpolacijski spline podataka iz Tablice 1.*

Primjer 3. *Koristeći linearni interpolacijski spline, uvećaj* 3×1 *sliku s pripadnim podatcima:* [[100 150 200]] , *u sliku dimenzije:* 7×1, *te ispiši pripadne podatke i usporedi podatke i novonastalu sliku s rezultatima iz prošlog primjera.*

Slika 11: *7x1 slika iz prethodnog primjera*

Slika 12: *novonastala 7x1 slika*

Uočimo da nema gotovo nikakve vidljive razlike među slikama. Pogledajmo stoga podatke:

Stari podaci: $7 \times 1 : [[100 \quad 107 \quad 129 \quad 150 \quad 171 \quad 193 \quad 200]]$

Novi podaci: $7 \times 1 : [[95 \quad 104 \quad 125 \quad 150 \quad 175 \quad 196 \quad 205]]$

Jasno je da na malim primjerima nema velikih razlika u podacima, stoga ne možemo golim okom uočiti razliku. No razlika između linearnog i kubičnog splinea jasno će biti vidljiva u nastavku ovog rada, kada budemo interpolirali 2-D slike većih dimenzija, tada će sve što smo dosada obradili dobiti smisao.

2 Dvodimenzionalna interpolacija

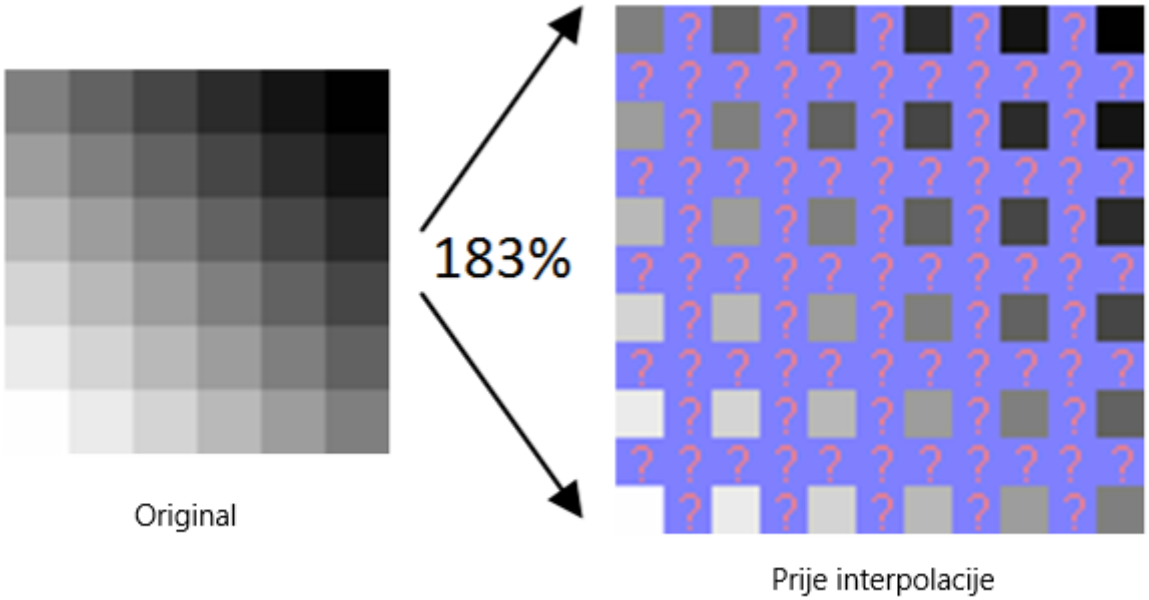
Sada kada znamo interpolirati jedan red slike imamo gotovo sve što trebamo da bi interpolirali cijelu sliku. Kao i dosad u ovom radu krenit ćemo sa najjednostavnijom metodom, a to je metoda najbližeg susjedstva ili metoda lokalno konstantne interpolacije.
Slično kao u prvom poglavlju, definirati ćemo skup podataka koji ćemo obraditi svim metodama, radi usporedbe. Ovaj put je to tablica dimenzija 6x6 koja reprezentira vrijednosti piksela 6x6 grayscale slike:

125	100	75	50	25	0
155	130	105	80	55	3
180	155	130	105	80	55
205	180	155	130	105	80
230	205	180	155	130	105
255	230	205	180	155	130

Tablica 2: Vrijednosti piksela 6x6 slike

2.1 Metoda najbližeg susjedstva

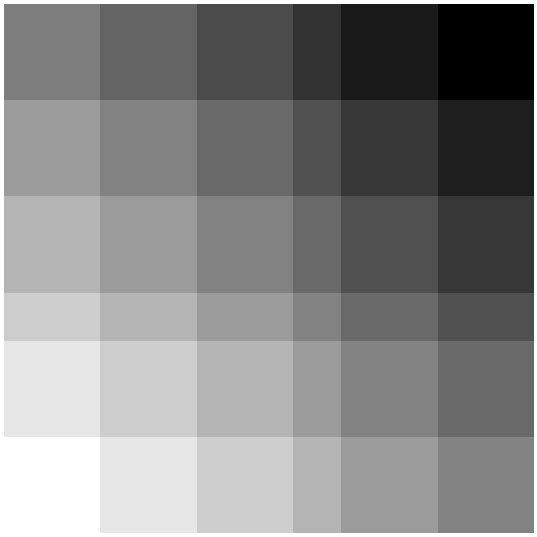
Prije nego interpoliramo podatke iz Tablice 2, idemo vidjeti što se zapravo dogodi kada povećamo 2-D sliku. Pa povećajmo sliku s podacima iz Tablice 2 iz 6x6 slike u 11x11 sliku. To je uvećanje od otprilike 183% .



Slika 13: *Postupak uvećavanja*

Pikseli iz originala raširе se koliko je potrebno, te između njih ostaju pikseli čije vrijednosti ne znamo, to jest moramo ih izračunati. Kod metode najbližeg susjedstva taj "izračun" je

u stvari pronalazak najbližeg piksela, te kopiranje njegove vrijednosti. Tako će svi pikseli iz ljubičastog područja sa slike, zapravo biti kopije svojeg najbližeg susjeda, od tuda i ime ove metode. Pa će rezultat nakon interpolacije biti:



Slika 14: *Uvećana 11x11*

Uočimo da je jedan redak i jedan stupac uži od svih ostalih, to nije greška metode, nego jednostavno stvar dimenzija. Originalno je slika bila 6x6, a nova je 11x11. A 11 nije djeljivo s 6, iz tog razloga, Iz tog razloga ne mogu svi retci/stupci biti iste širine. Odabir retka/stupca koji je uži je stvar implementacije resize() funkcije. Pogledajmo vrijednosti piksela 11x11 slike:

```
[[125 125 100 100  75  75  50  25  25   0   0]
 [125 125 100 100  75  75  50  25  25   0   0]
 [155 155 130 130 105 105  80  55  55  30  30]
 [155 155 130 130 105 105  80  55  55  30  30]
 [180 180 155 155 130 130 105  80  80  55  55]
 [180 180 155 155 130 130 105  80  80  55  55]
 [205 205 180 180 155 155 130 105 105  80  80]
 [230 230 205 205 180 180 155 130 130 105 105]
 [230 230 205 205 180 180 155 130 130 105 105]
 [255 255 230 230 205 205 180 155 155 130 130]
 [255 255 230 230 205 205 180 155 155 130 130]]
```

Tablica 3: Vrijednosti piksela 11x11 slike

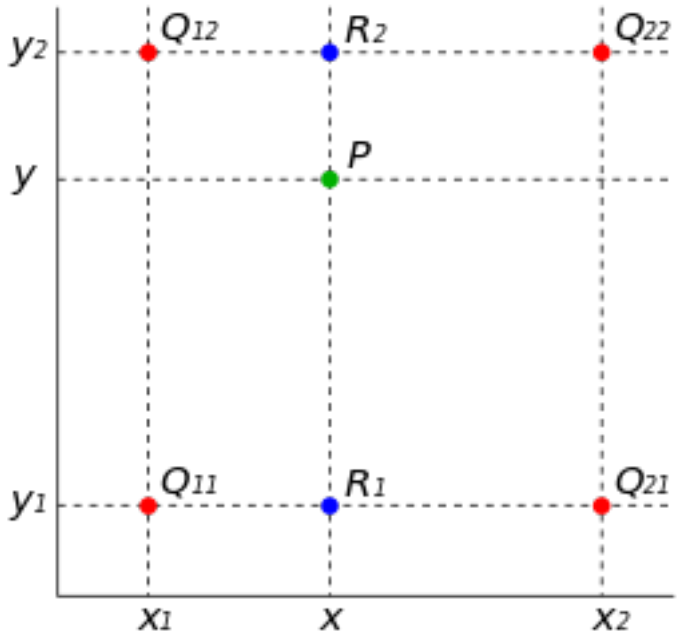
Vidimo da su uistinu svi podaci, kopije originalnih kao što je i očekivano uz opisani postupak. Originalna i uvećana slika, ispis podataka rezultat su sljedeće python skripte:

```
import numpy as np
from PIL import Image

array = np.zeros([6, 6], dtype=np.uint8)
for i in range(6):
    array[0][i] = 127-25*i-2
for i in range(1,6):
    for j in range(6):
        array[i][j] = array[0][j] + 25*i+5

img = Image.fromarray(array)
img.save('Original.png')

img = img.resize((11,11))
img.save('11x11.png')
pix = np.array(img)
print(pix)
```

Uočimo da resize() metodi nismo kao argument proslijedili koju vrstu interpolacije želimo koristiti, jer ona defaultno koristi metodu najbližeg susjedstva, pa to nije bilo potrebno. Isto kao u 1-D slučaju, slike dobivene metodom najbližeg susjedstva nemaju nikakve prijelazne boje. Više mogućnosti nam pruža linearna interpolacija, koja se u 2-D slučaju zove bilinearna interpolacija.

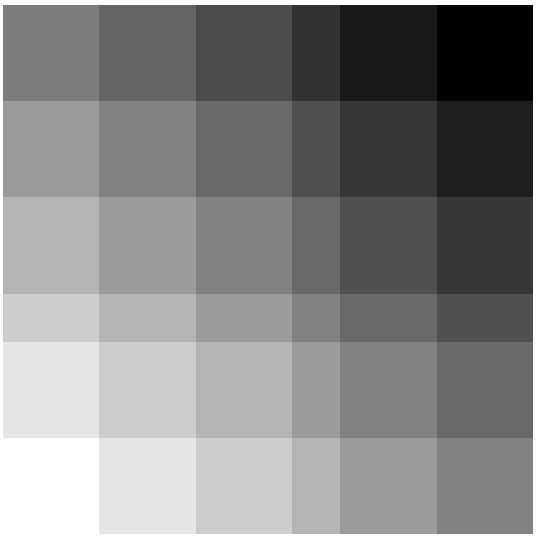
2.2 Bilinearna interpolacija

U 1-D slučaju bilo je jednostavno linearno interpolirati, jer smo imali dvije točke kroz koje treba provući pravac, te odabrati vrijednost na tom pravcu koju ćemo pridružiti pikselu. U 2-D slučaju stvari su malo kompliciranije, jer se novonastali piksel može nalaziti između dva ili četiri piksela. Dva ako je točno poravnat sa svoja dva susjeda, to jest nalazi se u istom redu kao i poznati podaci koji ga okružuju (npr. prvi red u slici prije interpolacije, Slika 13). Četiri ako se nalazi u redu u kojem nema poznatih podataka (npr. drugi red u slici prije interpolacije, Slika 13). Objasnimo to na idućoj slici:

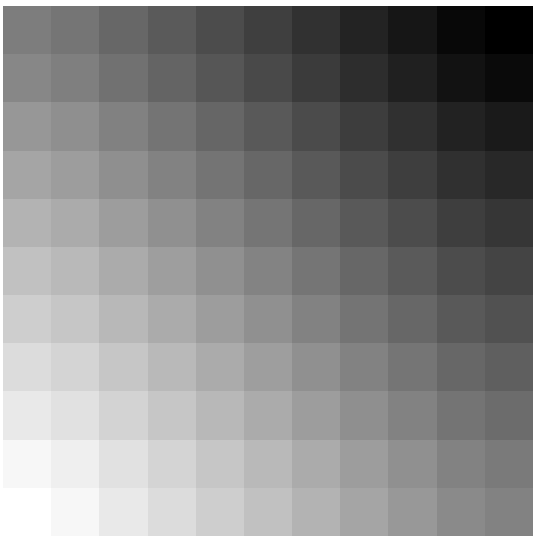


Slika 15: *Crveno - poznate vrijednosti, Zeleno - tražena vrijednost, Plava - pomoćne vrijednosti*

Uz notaciju sa slike i opisa, objasnimo kako fukcionira 2-D linearna interpolacija. Tražena vrijedenost je u točki P s koordinatama (x, y) koja se nalazi između 4 poznate vrijednosti $Q_{11}, Q_{12}, Q_{21}, Q_{22}$. Vrijednost u točki P dobijemo tako da prvo linearno interpoliramo vrijednost točaka R_1 i R_2. Vrijednost točke R_1 dobijemo provlačenjem pravca kroz točke Q_{21} i Q_{11} te uvrštavanjem x koordinate. Analogno dobijemo R_2 provlačenjem pravca kroz točke Q_{12} i Q_{22} te uvrštavanjem x koordinate. Sada kada imamo dva poznata podatka R_1 i R_2 koji su kolinearna s točkom P, provlačimo pravac kroz točke R_1 i R_2, te određujemo vrijednost točke P uvrštavanjem koordinate y na taj pravac. Zaključujemo da se bilinearna interpolacija zapravo svodi na tri linearne interpolacije.
Usporedimo sad rezultat metode bilinearne interpolacije i metode najbližeg susjedstva, pri uvećanju slike 6x6 s podacima iz tablice 2. u sliku 11x11:



(a) Metoda najbližeg susjedstva



(b) Bilinearna interpolacija

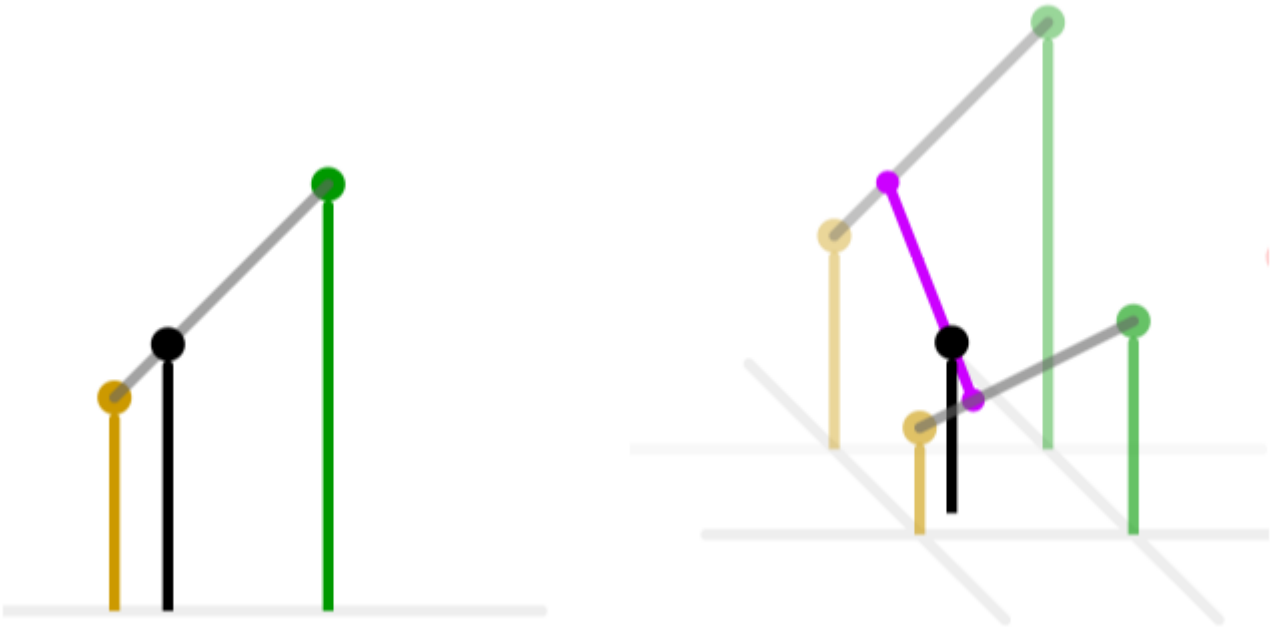
```
[[125 117 103  90  76  63  49  35  22   8   0]
 [135 127 113 100  86  73  59  45  32  18  10]
 [151 143 129 116 102  89  75  61  48  34  26]
 [165 157 143 130 116 103  89  75  62  48  40]
 [179 171 157 144 130 117 103  89  76  62  54]
 [193 185 171 158 144 131 117 103  90  76  68]
 [206 198 184 171 157 144 130 116 103  89  81]
 [220 212 198 185 171 158 144 130 117 103  95]
 [233 225 211 198 184 171 157 143 130 116 108]
 [247 239 225 212 198 185 171 157 144 130 122]
 [255 247 233 220 206 193 179 165 152 138 130]]
```

Tablica 4: Vrijednosti piksela slike dobivene bilinearnom interpolacijom

Odmah je jasno da bilinearna interpolacija nudi puno ljepše prijelaze boja već na ovako malom primjeru, ali je ona računski zahtjevnija od metode najbližeg susjedstva, stoga i vremenski zahtjevnija. Preostalo nam je još obraditi 2-D slučaj kubične interpolacije, takozvanu bikubičnu interpolaciju, koja se u praksi najčešće i koristi od svih koje smo dosada obradili.

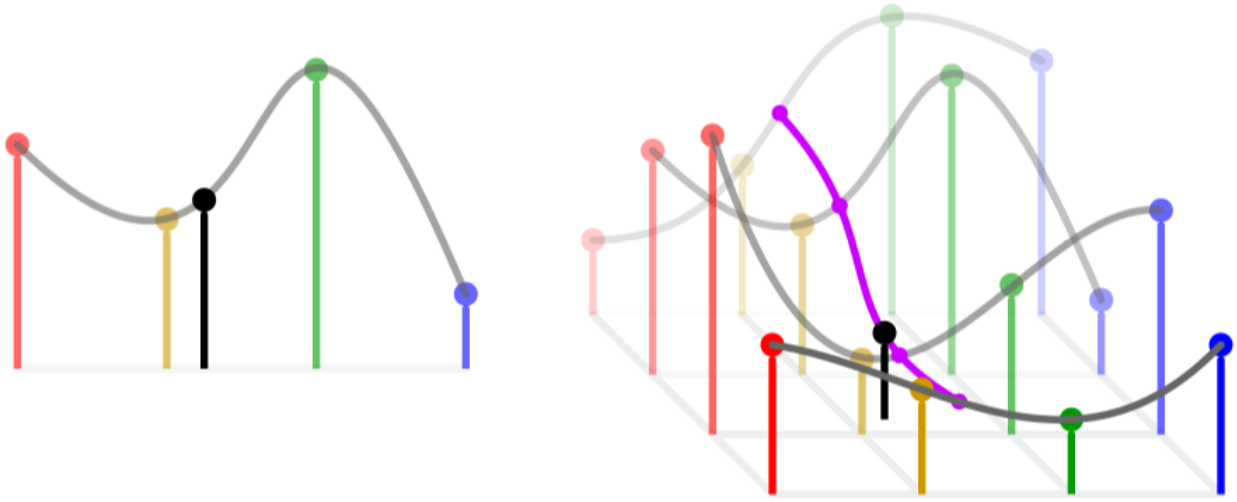
2.3 Bikubična interpolacija

Bikubična interpolacija posljednja je interpolacija koju ćemo obraditi u ovom radu. Već smo spomenuli da se ona najčešće koristi u praksi u alatima za obradu slika. Koristi se upravo iz razloga što je slika uvećana bikubičnom iterpolacijom oku najugodnija, što ćemo i vidjeti na primjerima. Za razliku od bilinearne interpolacije koja je za izračun vrijednosti nepoznatog piksela koristila 2x2 područje piksela koje ga okružuje, to jest 4 piksela. Bikubična interpolacija koristi 4x4 područje, to jest, ukupno 16 piksela. Pogledajmo proces bikubične interpolacije:
Slično kao što smo kod bilinearne interpolacije tri puta linearno interpolirali



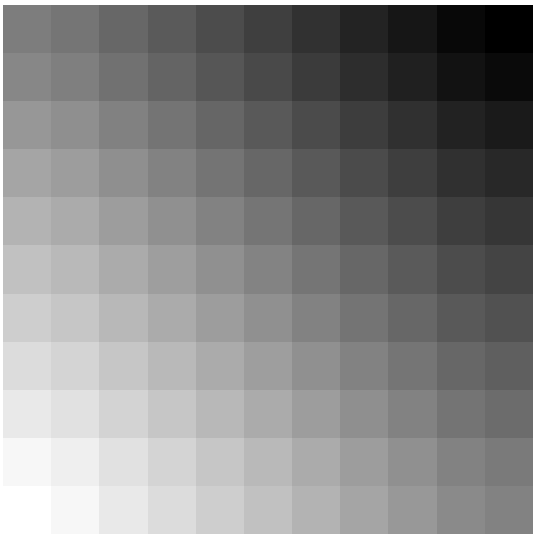
Slika 17: *Usporedba linearne i bilinearne interpolacije*

Kod bikubične interpolacije provodimo postupak kubične interpolacije pet puta:

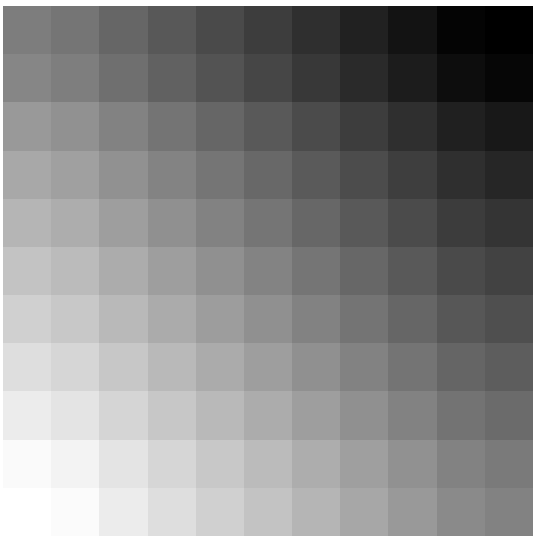


Slika 18: *Usporedba kubične i bikubične interpolacije*

Crvena, žuta, zelena i plava boja predstavljaju poznate podatke. Crna boja predstavlja traženi podatak. Prvi korak je kubična interpolacija susjednih poznatih podataka, sive krivulje predstavljaju upravo taj korak. Potom odabiremo pomočne podatke na dobivenim krivuljama, to jest na slici, ljubičaste čvorove koje ćemo također kubično interpolirati, te na dobivenoj krivulji, odabrati naš traženi podatak.
Usporedimo sad rezultat metode bilinearne interpolacije i bikubične interpolacije, pri uvećanju slike 6x6 s podacima iz tablice 2. u sliku 11x11:



(a) Bilinearna interpolacija



(b) Bikubična interpolacija

Razlika na slikama gotovo je neprimjetna, ali ako pogledamo podatke slike dobivene bikubičnom interpolacijom i usporedimo s podacima iz tablice 4., vidimo da razlika postoji.

```
[[125 117 102  88  74  61  47  33  19   4   0]
 [134 126 111  97  83  70  56  42  28  13   6]
 [153 145 130 116 102  89  75  61  47  32  24]
 [168 160 145 131 117 104  90  76  62  47  38]
 [181 173 158 144 130 117 103  89  75  60  52]
 [195 187 172 158 144 131 117 103  89  74  66]
 [208 200 185 171 157 144 130 116 102  87  79]
 [222 214 199 185 171 158 144 130 116 101  93]
 [236 228 213 199 185 172 158 144 130 115 107]
 [250 243 228 214 200 187 173 159 145 130 122]
 [255 251 236 222 208 195 181 167 153 138 130]]
```

Tablica 5: Vrijednosti piksela slike dobivene bikubičnom interpolacijom

Dosada nismo vidjeli pravu razliku između bikubične i linearne interpolacije, iako je bikubična računski najzahtjevnija od metoda koje smo obradili, nije nam ponudila ništa bolje rezultate nego bilinearna. Iz tog razloga što na dosadašnjim primjerima, koji su bili dosta trivijalni, nismo mogli uočiti značajnu razliku između odabira bilinearne interpolacije i bikubične interpolacije, u sljedećem podpoglavlju obradit ćemo slike većih dimenzija na kojima ćemo uistinu moći vidjeti razliku.

2.4 Usporedba i primjene metoda interpolacije

Sada kada smo utvrdili kako koja metoda funkcionira, primijeniti ćemo ih na primjeru iz stvarnog života.

Primjer 4. *Uzmimo zadanu fotografiju dimenzija 100x100, i uvećajmo ju na 400% te interpolirajmo svim obrađenim metodama, usporedimo rezultate i prokomentirajmo nedostatke i kvalitete svake metode.*

Slika 20: *Originalna fotografija dimezija 100x100*

Slika 21: Metoda najbližeg susjedstva

Metoda najbližeg susjedstva daje čistu sliku, ali jako zrnatu. Ovu metodu dobro je koristiti za jednostavne slike, na primjer slike koje možemo nacrtati u Microsoftovom alatu Paint, slike koje sadrže slova jer ih neće zamutiti nego će samo replicirati ono što se nalazilo u originalu, i uvećati to.

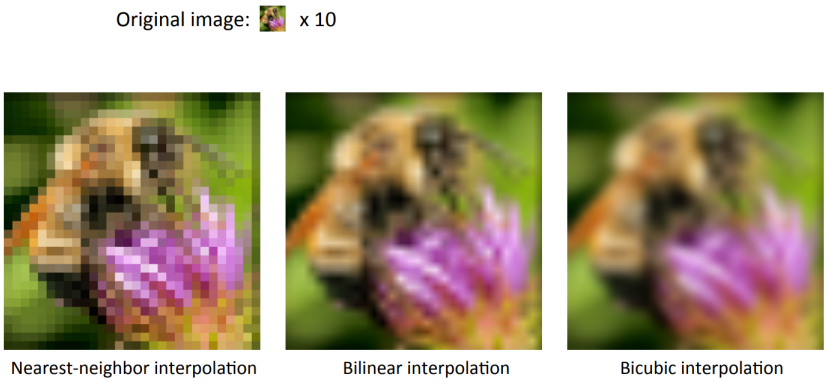
Slika 22: Bilinearna interpolacija

Bilinearna interpolacija nam daje sliku koja se čini mutna, ali znatno manje zrnata nego slika koju nam daje metoda najbližeg susjedstva. Ova metoda se čini kao dobar izbor jer je za razliku od metode najbližeg susjedstva koja je jako zrnata, ali nije mutna, ova metoda podjednako mutna i zrnata.

Slika 23: Bikubična interpolacija

Slika koju nam daje metoda bikubične interpolacije čini se podjednako mutna kao i ona koju nam daje bilinearna inerpolacija, ali ako pogledamo na primjer područje oko očiju mačke sa slike, vidimo da je slika 22. puno zrnatija nego slika 23 (ukoliko razlika nije vidljiva pogledaj [16]). Tu dolazi do izražaja bikubična interpolacija koja za izračun u obzir uzima 16 okolnih piksela, za razliku od 4 piksela koliko ih uzima bilinearna.

Slične rezultate možemo vidjeti i na sljedećem primjeru:

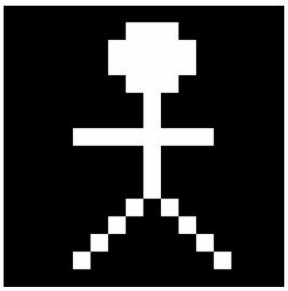


Slika 24: Usporedba metoda

Iz čega zaključujemo da ćemo birati metodu interpolacije ovisno o tome koliko rezultantna slika treba biti oštra.

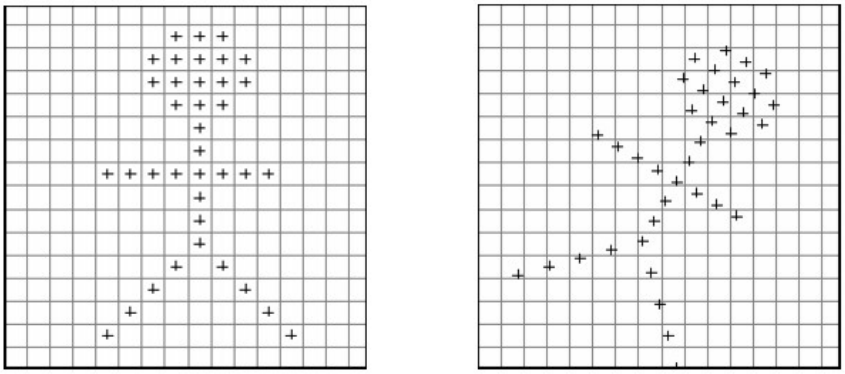
Osim kod uvećavanja slika, obrađene metode interpolacije koriste se kod gotovo svih transformacija slika kao što su na primjer: rotacija, smanjenje slika, iskrivljenje slika i slično. Pogledajmo na sljedećim primjerima:

Primjer 5. *Neka je zadana slika:*



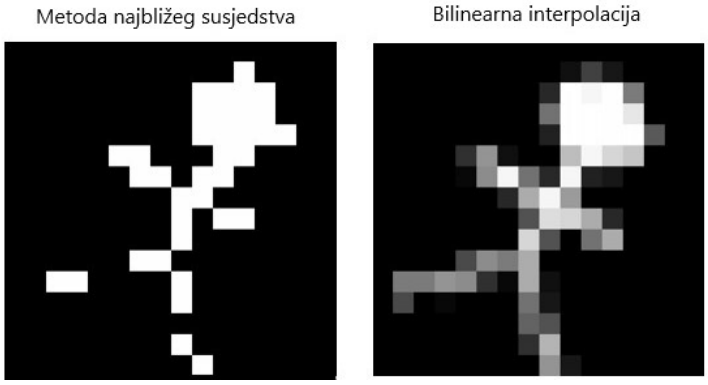
Slika 25: Zadana slika

Uslijed rotacije originalne slike događa se slijedeće: Kada se centar transformiranog pi-



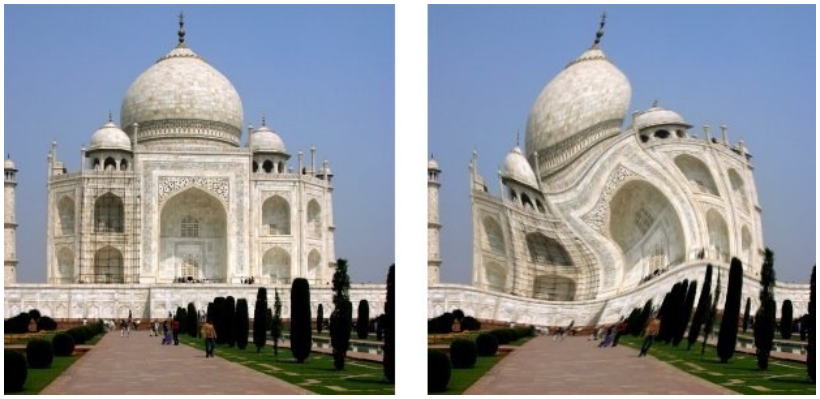
Slika 26: centri piksela prije rotacije - centri piksela poslje rotacije

skela ne podudara s nijednim pikselom u novoj slici, moramo koristiti nekakvu metodu kojom ćemo determinirati što će se nalaziti na pikselu nove slike. To su najčešće neke od metoda interpolacije koje smo obradili.



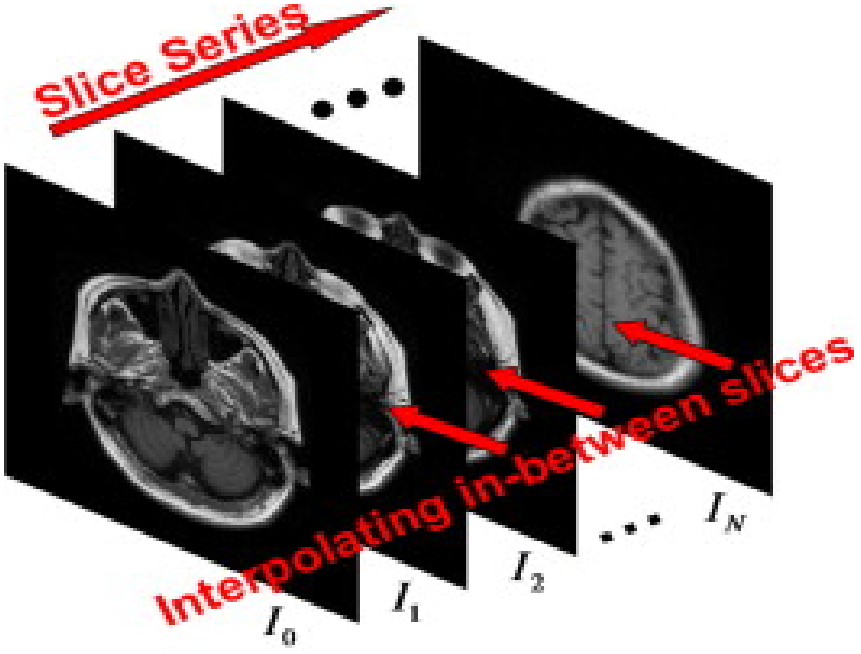
Slika 27: Rezultati interpolacije

Jasno je da pri rotaciji već na ovako malim i jednostavnim primjerima kvaliteta slike opada jako brzo. Slično je i kod iskrivljavanja slika:



Slika 28: Primjer iskrivljavanja slike

Osim primjene na jednoj slici, metode interpolacije također se primjenjuju za interpolaciju podataka između više slika. Linearna interpolacija se u medicini koristi pri izračunu nepoznatih režnjeva prilikom snimanja CT strojom. Također su se prije, zbog njihove brzine i jednostavnosti, koristile linearna i kubična interpolacija za izračun podataka između režnjeva dobivene magnetskom rezonancom.



Slika 29: Primjer režnjeva magnetske rezonance

Naravno danas postoje složenije metode interpolacije kao što su Lanczos interpolacija, Krigingova metoda interpolacije, fraktalna interpolacija, hibridne metode itd., ali njihova analiza nadilazi opseg ovog rada.

Literatura

[1] R. Scitovski, *Numerička Matematika*, Odjel za matematiku, Sveučilište Josipa Jurja Strossmayera u Osijeku, udžbenik. https://www.mathos.unios.hr/images/homepages/scitowsk/Num.pdf

[2] *Bicubic interpolation*, https://en.wikipedia.org/wiki/Bicubic_interpolation

[3] Cambridge in colour, *Digital image interpolation*, Članak https://www.cambridgeincolour.com/tutorials/image-interpolation.htm

[4] *Interpolacija*, https://hr.wikipedia.org/wiki/Interpolacija

[5] Vision Systems, *Understanding image-interpolation techniques*, Članak https://www.vision-systems.com/articles/print/volume-12/issue-10/departments/wilsons-websites/understanding-image-interpolation-techniques.html

[6] *Nearest-neighbor interpolation*, https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation

[7] *Matplotlib*, https://matplotlib.org/users/image_tutorial.html

[8] *GIMP*, https://docs.gimp.org/en/plug-in-whirl-pinch.htmll

[9] Python Informer, *Image processing with numpy*, Članak http://pythoninformer.com/python-libraries/numpy/numpy-and-images/

[10] *Bilinear interpolation*, https://en.wikipedia.org/wiki/Bilinear_interpolation

[11] *Interpolation Methods*, http://northstar-www.dartmouth.edu/doc/idl/html_6.2/Interpolation_Methods.html

[12] Alex Clark, *Pillow (PIL Fork) Documentation*, https://media.readthedocs.org/pdf/pillow/latest/pillow.pdf

[13] Autar Kaw, *History of Interpolation*, https://www.saylor.org/site/wp-content/uploads/2011/11/ME205-5.1-TEXT2.pdf

[14] Francesca Pizzorni Ferrarese , *IMAGE INTERPOLATION*, Odjel za računalnu znanost, Sveučilište u Veroni, http://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid358544.pdf

[15] M. Ohnesorge, Thomas G. Flohr, Christoph R. Becker, Andreas Knez, Maximilian F Reiseri, *Multi-slice and Dual-source CT in Cardiac Imaging*, https://books.google.hr/books?id=ywdP9WHoXaUC&pg=PA81&dq=slice+interpolation&hl=hr&sa=X&ved=0ahUKEwiS993R5ZLdAhUC-aQKHaoMBK8Q6AEIJjAA#v=onepage&q&f=false

[16] *GIF*, https://media.giphy.com/media/3Fb4MojvN2MDPLgxcM/giphy.gif