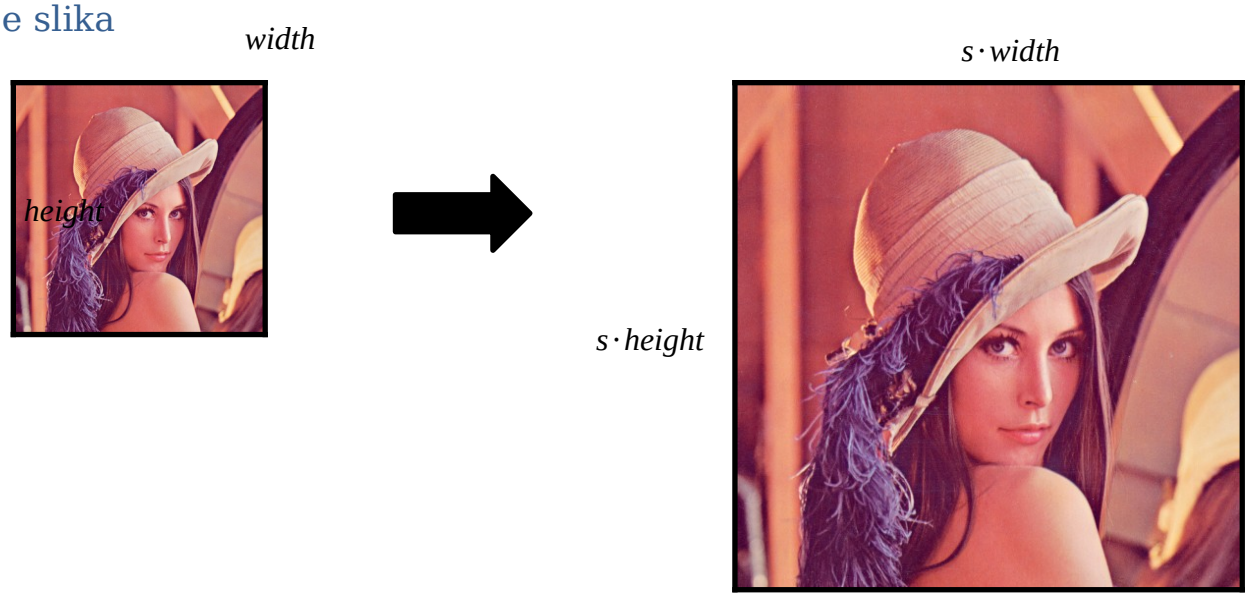
5. Interpolacija

Skaliranje slika



Slika 1. Skaliranje slika

Potrebno je uvećati sliku za poznati faktor skaliranja s.

1. *Nearest-neighbor* interpolacija

Nearest-neighbor interpolacijom se svaki *pixel* originalne slike ponavlja s puta po visini i po širini, gde je s faktor skaliranja. Za fakor skaliranja npr. $s=3$:

	1	2
1	1	2
2	3	4

	1	2	3	4	5	6
1	1	1	1	2	2	2
2	1	1	1	2	2	2
3	1	1	1	2	2	2
4	3	3	3	4	4	4
5	3	3	3	4	4	4
6	3	3	3	4	4	4

, vrednosti ulazne matrice na indeksima (y_i, x_i) se preslikavaju na vrednosti izlazne matrice na indeksima (y_o, x_o) na sledeći način:

(y_o, x_o)	(y_i, x_i)
$(1,1)$	$(1,1)$
$(1,2)$	$(1,1)$
$(1,3)$	$(1,1)$
$(1,4)$	$(1,2)$
$(1,5)$	$(1,2)$
$(1,6)$	$(1,2)$
. . .	. . .

$(4,1)$	$(2,1)$
$(4,2)$	$(2,1)$
$(4,3)$	$(2,1)$
$(4,4)$	$(2,2)$
$(4,5)$	$(2,2)$
$(4,6)$	$(2,2)$

Funkcija koja opisuje ovo preslikavanje je ((y_i, x_i) se izračunavaju na osnovu (y_o, x_o)):

$$f(y_o, x_o) = \left(\lfloor \frac{y_o - 1}{s} \rfloor + 1, \lfloor \frac{x_o - 1}{s} \rfloor + 1 \right) = (y_i, x_i)$$

, gde su x i y indeksi izlazne matrice, s je faktor skaliranja, a $\lfloor \rfloor$ predstavlja zaokruživanje vrednosti na manju.

2. Bilinearna interpolacija

Bilinearnom interpolacijom se izlazna matrica deli na $n \times m$ blokova:
$(n, m) = (h - 1, w - 1)$

, gde su w i h, širina, odnosno visina ulazne matrice.

Za faktor skaliranja npr. $s = 3$:

blok		1		2	
		1	2	3	
1	1	1.00	2.00	3.00	
2	2	4.00	5.00	6.00	
	3	7.00	8.00	9.00	

blok		1				2				
		1	2	3	4	5	6	7	8	9
1	1	1.00	1.25	1.50	1.75	2.00	2.25	2.50	2.75	3.00
	2	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	3.75
	3	2.50	2.75	3.00	3.25	3.50	3.75	4.00	4.25	4.50
	4	3.25	3.50	3.75	4.00	4.25	4.50	4.75	5.00	5.25
	5	4.00	4.25	4.50	4.75	5.00	5.25	5.50	5.75	6.00
2	6	4.75	5.00	5.25	5.50	5.75	6.00	6.25	6.50	6.75
	7	5.50	5.75	6.00	6.25	6.50	6.75	7.00	7.25	7.50
	8	6.25	6.50	6.75	7.00	7.25	7.50	7.75	8.00	8.25
	9	7.0	7.2	7.5	7.7	8.0	8.2	8.5	8.7	9.0

0	5	0	5	0	5	0	5	0

, za blok $(1,1)$, vrednosti ulazne matrice se preslikavaju na vrednosti izlazne matrice na sledeći način:

$$o_1=\frac{(5-2)}{(5-1)}i_{1,1}+\frac{(2-1)}{(5-1)}i_{1,2}o_2=\frac{(5-2)}{(5-1)}i_{2,1}+\frac{(2-1)}{(5-1)}i_{2,2}o_{3,2}=\frac{(5-3)}{(5-1)}o_1+\frac{(3-1)}{(5-1)}o_2$$

, gde su i vrednosti ulazne matrice, a o vrednosti izlazne matrice.

Za blok (y_b, x_b), vrednosti ulazne matrice se preslikavaju na vrednosti izlazne matrice na sledeći način:

$$o_1 = \frac{(x_2 - x)}{(x_2 - x_1)} i_{y_b, x_b} + \frac{(x - x_1)}{(x_2 - x_1)} i_{y_b, x_b+1} o_2 = \frac{(x_2 - x)}{(x_2 - x_1)} i_{y_b+1, x_b} + \frac{(x - x_1)}{(x_2 - x_1)} i_{y_b+1, x_b+1} o_{y,x} = \frac{(y_2 - y)}{(y_2 - y_1)} o_1 + \frac{(y - y_1)}{(y_2 - y_1)} o_2$$

, gde su x_1 i x_2, početni, odnosno krajnji indeksi elemenata bloka (y_b, x_b) po širini izlazne matrice, y_1 i y_2, su početni, odnosno krajnji indeksi bloka (y_b, x_b) po visini izlazne matrice, a x i y su indeksi traženih elemenata bloka, pri čemu važi:
$x_1 \leq x \leq x_2$
$y_1 \leq y \leq y_2$

x_1, x_2, y_1 i y_2 se za svaki blok mogu izračunati unapred, na osnovu dimenzija izlazne matrice i širine, odosno visine bloka. Primetiti da za svaki blok važi:

$x_1 = x_2^{prethodnog\ bloka}$, osim za 1. blok
$y_1 = y_2^{prethodnog\ bloka}$, osim za 1. blok

Zadatak 1
Napisati funkcije koje realizuju *nearest-neighbor* i bilinearnu interpolaciju nad fotografijom proizvoljnih dimenzija i uporediti rezultate.

a) Definisati funkciju `nnInter`:

```
def nnInter(input, factor):
```

b) Implementirati *nearest-neighbor* interpolaciju ulazne matrice `in` na izlaznu matircu `out` za prosleđeni faktor skaliranja `factor`.

c) Testirati funkciju na primeru:

```
A = np.array([[1, 2]
              [3, 4]])

nnInter(A, 3.0)
```

Rezultat:

```
1    1    1    2    2    2
1    1    1    2    2    2
1    1    1    2    2    2
3    3    3    4    4    4
3    3    3    4    4    4
3    3    3    4    4    4
```

d) Definisati funkciju `scaleNN`:

```
def scaleNN(in, factor):
    # rastavljanje 3D matrice na 3 2D matrice
    R = in[:,:,0]
    G = in[:,:,1]
    B = in[:,:,2]
    # skaliranje
    R = nnInter(R, factor)
    G = nnInter(G, factor)
    B = nnInter(B, factor)

    # kombinovanje 3D matrice od 3 2D matrice
    # zaokruživanje realnih vrednosti na celobrojne
    return (np.dstack((R, G, B)) * 255.999).astype(np.uint8)
```

e) Testirati funkciju na primeru:

```
factor = 3
image = imread('Lenna.png') # učitavanje slike

nnScaled = scaleNN(image, factor)
imsave('Lenna (nearest-neighbor).png', nnScaled) # čuvanje slike

plt.imshow(nnScaled)
plt.show() # prikaz slike
```

f) Definisati funkciju `bilinearInter`:

```
def bilinearInter(in, factor):
```

g) Implementirati bilinearnu interpolaciju ulazne matrice `in` na izlaznu matircu `out` za prosleđeni faktor skaliranja `factor`.

h) Testirati funkciju na primeru:

```
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

bilinearInter(A, 3.0)
```

Rezultat:

```
1.0000    1.2500    1.5000    1.7500    2.0000    2.2500    2.5000    2.7500    3.0000
1.7500    2.0000    2.2500    2.5000    2.7500    3.0000    3.2500    3.5000    3.7500
2.5000    2.7500    3.0000    3.2500    3.5000    3.7500    4.0000    4.2500    4.5000
3.2500    3.5000    3.7500    4.0000    4.2500    4.5000    4.7500    5.0000    5.2500
4.0000    4.2500    4.5000    4.7500    5.0000    5.2500    5.5000    5.7500    6.0000
4.7500    5.0000    5.2500    5.5000    5.7500    6.0000    6.2500    6.5000    6.7500
5.5000    5.7500    6.0000    6.2500    6.5000    6.7500    7.0000    7.2500    7.5000
6.2500    6.5000    6.7500    7.0000    7.2500    7.5000    7.7500    8.0000    8.2500
7.0000    7.2500    7.5000    7.7500    8.0000    8.2500    8.5000    8.7500    9.0000
```

i) Definisati funkciju `scaleBilinear`:

```
def scaleBilinear(in, factor):
    # rastavljanje 3D matrice na 3 2D matrice
    R = in[:,:,0]
    G = in[:,:,1]
    B = in[:,:,2]

    R = bilinearInter(R, factor)
    G = bilinearInter(G, factor)
    B = bilinearInter(B, factor)

    # kombinovanje 3D matrice od 3 2D matrice
    # zaokruživanje realnih vrednosti na celobrojne
    return (np.dstack((R, G, B)) * 255.999).astype(np.uint8)
```

j) Dopuniti test pod e) i uporediti rezultate:

```
factor = 3.0
image = imread('Lenna.png') # učitavanje slike

nnScaled = scaleNN(image, factor)
imsave('Lenna (nearest-neighbor).png', nnScaled) # čuvanje slike
bilinearScaled = scaleBilinear(image, factor)
imsave('Lenna (bilinear).png', bilinearScaled)

plt.imshow(nnScaled) # prikaz slike
plt.imshow(bilinearScaled)
plt.show()
```