NEAREST NEIGHBOR INTERPOLATION

The basic idea behind nearest neighbor interpolation is to assign the pixel closest to the newly generated address as the output pixel. With this technique, the fractional address computed for the source pixel is rounded to the nearest valid pixel address. It is implemented in the code segment in Listing 4.1. The addition of 0.5 rounds the address to the nearest valid integer address.

Although fast, nearest neighborhood resampling yields results that can vary greatly. Since no new pixel values are computed, all output pixels are from the set of input pixels. The resulting output pixel can be in error by as much as √(2)/2 pixel units. Nearest neighbor interpolation cannot be used when subpixel values are of interest.

In general, the greater the number of output pixels tied to one input pixel, the worse the output looks. Scaling by a large number is an example of this. Visual blockiness, also known as the jaggies, can be seen in the output. If better results are required, use another interpolation function.

```
floatx = x_mapping_function(X_dest);
floaty = y_mapping_function(Y_dest);
X_source = (int) (floatx + 0.5);
Y_source = (int) (floaty + 0.5);
```

LISTING 4.1 Nearest neighbor interpolation.

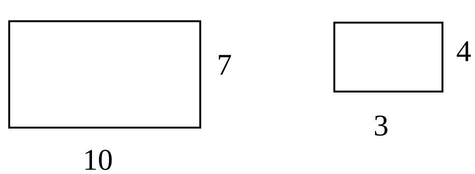
92)A is a 7x10 matrix as follows.

$$A=\begin{bmatrix} 50 & 97 & 16 & 45 & 78 & 23 & 22 & 15 & 10 & 77 \\ 61 & 62 & 56 & 39 & 71 & 83 & 57 & 2 & 91 & 85 \\ 82 & 70 & 69 & 78 & 11 & 2 & 12 & 44 & 11 & 92 \\ 53 & 72 & 43 & 73 & 39 & 86 & 67 & 83 & 52 & 99 \\ 20 & 35 & 84 & 43 & 59 & 8 & 60 & 62 & 14 & 51 \\ 45 & 52 & 73 & 69 & 46 & 67 & 6 & 52 & 56 & 27 \\ 43 & 56 & 36 & 95 & 5 & 50 & 6 & 86 & 0 & 10 \end{bmatrix}$$

A matrisini bir resim olarak dusunun biz bu resmi 4x3 olarak kucultmek istiyoruz.

B is a 4x3 matrix.

A(7x10), B(4,3)



yeni_Y=eski_Y*4/7, yeni_X= eski_X *3/10

eski_Y	yeni_Y	Round(yeni_Y)
1	0.571	1
2	1.14	1
3	1.71	2
4	2.29	2
5	2.86	3
6	3.43	3
7	4	4

eski_X	yeni_X	Round(yeni_X)
1	0.3	0
2	0.6	1
3	0.9	1
4	1.2	1
5	1.5	2
6	1.8	2
7	2.1	2
8	2.4	4
9	2.7	3
10	3	3

B(yeni_Y,yeni_X)= A(eski_Y,eski_X),

Matrislerin indexleri yuvarlatilmistir.

$$\begin{bmatrix} \text{gercek\_X} \\ \text{gercek\_Y} \end{bmatrix} = \begin{bmatrix} \text{round(yeni\_X)} \\ \text{round(yeni\_Y)} \end{bmatrix}$$

B(1,1)=A(1,2)=97 B(2,2)=A(3,6)=2 B(3,3)=A(5,10)=51
B(1,1)=A(1,3)=16 B(2,2)=A(3,7)=12 B(3,1)=A(6,2)=52
B(1,1)=A(1,4)=45 B(2,2)=A(3,8)=44 B(3,1)=A(6,3)=73
B(1,2)=A(1,5)=78 B(2,3)=A(3,9)=11 B(3,1)=A(6,4)=69
B(1,2)=A(1,6)=23 B(2,3)=A(3,10)=92 B(3,2)=A(6,5)=46
B(1,2)=A(1,7)=22 B(2,1)=A(4,2)=72 B(3,2)=A(6,6)=67
B(1,2)=A(1,8)=15 B(2,1)=A(4,3)=43 B(3,2)=A(6,7)=6
B(1,3)=A(1,9)=10 B(2,1)=A(4,4)=73 B(3,2)=A(6,8)=52
B(1,3)=A(1,10)=77 B(2,2)=A(4,5)=39 B(3,3)=A(6,9)=56
B(1,1)=A(2,2)=62 B(2,2)=A(4,6)=86 B(3,3)=A(6,10)=27
B(1,1)=A(2,3)=56 B(2,2)=A(4,7)=67 B(4,1)=A(7,2)=56
B(1,1)=A(2,4)=39 B(2,2)=A(4,8)=83 B(4,1)=A(7,3)=36
B(1,2)=A(2,5)=71 B(2,3)=A(4,9)=52 B(4,1)=A(7,4)=95
B(1,2)=A(2,6)=83 B(2,3)=A(4,10)=99 B(4,2)=A(7,5)=5
B(1,2)=A(2,7)=57 B(3,1)=A(5,2)=35 B(4,2)=A(7,6)=50
B(1,2)=A(2,8)=2 B(3,1)=A(5,3)=84 B(4,2)=A(7,7)=6
B(1,3)=A(2,9)=91 B(3,1)=A(5,4)=43 B(4,2)=A(7,8)=86
B(1,3)=A(2,10)=85 B(3,2)=A(5,5)=59 B(4,3)=A(7,9)=0
B(2,1)=A(3,2)=70 B(3,2)=A(5,6)=8 B(4,3)=A(7,10)=10
B(2,1)=A(3,3)=69 B(3,2)=A(5,7)=60
B(2,1)=A(3,4)=78 B(3,2)=A(5,8)=62

Eski resme ait pek cok nokta yeni resimde ayni noktaya tekabul edecektir. Mesela B(1,1) noktasina A matrisinden 6 nokta karsilik gelmektedir.
B(1,1)=A(1,2)=97 B(1,1)=A(1,3)=16 B(1,1)=A(1,4)=45
B(1,1)=A(2,2)=62 B(1,1)=A(2,3)=56 B(1,1)=A(2,4)=39

B(1,1) degeri olarak
a) 6 noktadan rasgele birini alabiliriz.
b)6 noktanin yaklasik geometrik ortasindaki rakami alabiliriz.
c)6 noktanin ortalamasini alabiliriz.
d)Bu6 noktanin pixellerini ve koordinat degerlerini hesaba katarak hesaplayabiliriz.

6 noktanin yaklasik geometrik ortasindaki rakami aldigimizda “nearest neighbour” yontemi uygulamis oluruz. Geometrik ortalama kavrami asagida aciklanmistir.

71	**17**	37

.

41	6	86
90	**66**	90
6	2	35

28
53
52

6 noktanin pixellerini ve koordinat degerlerini hesaba katarak hesapladigimizda bilinear, veya cubic yaklasim yapmis oluruz.

Nearest neighbour yontemiyle elde ettigimiz B matrisi asagidaki gibi olacaktir.

$$B=\begin{bmatrix} 97 & 23 & 10 \\ 70 & 2 & 11 \\ 35 & 8 & 14 \\ 56 & 50 & 0 \end{bmatrix}$$

Simdi de elimizdeki 4x3 boyutundaki B matrisini tekrar 7x10 haline getirelim.

B(4,3) → C(7x10),

yeni_Y=eski_Y*7/4, yeni_X= eski_X *10/3

C(yeni_Y,yeni_X)= B(eski_Y,eski_X),

C(1,1)=B(1,1)=97	C(2,9)=B(1,3)=10	C(4,7)=B(2,2)=2	C(6,5)=B(3,2)=8
C(1,2)=B(1,1)=97	C(2,10)=B(1,3)=10	C(4,8)=B(2,2)=2	C(6,6)=B(3,2)=8
C(1,3)=B(1,1)=97	C(3,1)=B(2,1)=70	C(4,9)=B(2,3)=11	C(6,7)=B(3,2)=8
C(1,4)=B(1,1)=97	C(3,2)=B(2,1)=70	C(4,10)=B(2,3)=11	C(6,8)=B(3,2)=8
C(1,5)=B(1,2)=23	C(3,3)=B(2,1)=70	C(5,1)=B(3,1)=35	C(6,9)=B(3,3)=14
C(1,6)=B(1,2)=23	C(3,4)=B(2,1)=70	C(5,2)=B(3,1)=35	C(6,10)=B(3,3)=14
C(1,7)=B(1,2)=23	C(3,5)=B(2,2)=2	C(5,3)=B(3,1)=35	C(7,1)=B(4,1)=56
C(1,8)=B(1,2)=23	C(3,6)=B(2,2)=2	C(5,4)=B(3,1)=35	C(7,2)=B(4,1)=56
C(1,9)=B(1,3)=10	C(3,7)=B(2,2)=2	C(5,5)=B(3,2)=8	C(7,3)=B(4,1)=56
C(1,10)=B(1,3)=10	C(3,8)=B(2,2)=2	C(5,6)=B(3,2)=8	C(7,4)=B(4,1)=56
C(2,1)=B(1,1)=97	C(3,9)=B(2,3)=11	C(5,7)=B(3,2)=8	C(7,5)=B(4,2)=50
C(2,2)=B(1,1)=97	C(3,10)=B(2,3)=11	C(5,8)=B(3,2)=8	C(7,6)=B(4,2)=50
C(2,3)=B(1,1)=97	C(4,1)=B(2,1)=70	C(5,9)=B(3,3)=14	C(7,7)=B(4,2)=50
C(2,4)=B(1,1)=97	C(4,2)=B(2,1)=70	C(5,10)=B(3,3)=14	C(7,8)=B(4,2)=50
C(2,5)=B(1,2)=23	C(4,3)=B(2,1)=70	C(6,1)=B(3,1)=35	C(7,9)=B(4,3)=0
C(2,6)=B(1,2)=23	C(4,4)=B(2,1)=70	C(6,2)=B(3,1)=35	C(7,10)=B(4,3)=0
C(2,7)=B(1,2)=23	C(4,5)=B(2,2)=2	C(6,3)=B(3,1)=35	
C(2,8)=B(1,2)=23	C(4,6)=B(2,2)=2	C(6,4)=B(3,1)=35	

$$\text{yeniA}=\begin{bmatrix} 97 & 97 & 97 & 97 & 23 & 23 & 23 & 23 & 10 & 10 \\ 97 & 97 & 97 & 97 & 23 & 23 & 23 & 23 & 10 & 10 \\ 70 & 70 & 70 & 70 & 2 & 2 & 2 & 2 & 11 & 11 \\ 70 & 70 & 70 & 70 & 2 & 2 & 2 & 2 & 11 & 11 \\ 35 & 35 & 35 & 35 & 8 & 8 & 8 & 8 & 14 & 14 \\ 35 & 35 & 35 & 35 & 8 & 8 & 8 & 8 & 14 & 14 \\ 56 & 56 & 56 & 56 & 50 & 50 & 50 & 50 & 0 & 0 \end{bmatrix} \quad \text{eskiA}=\begin{bmatrix} 50 & 97 & 16 & 45 & 78 & 23 & 22 & 15 & 10 & 77 \\ 61 & 62 & 56 & 39 & 71 & 83 & 57 & 2 & 91 & 85 \\ 82 & 70 & 69 & 78 & 11 & 2 & 12 & 44 & 11 & 92 \\ 53 & 72 & 43 & 73 & 39 & 86 & 67 & 83 & 52 & 99 \\ 20 & 35 & 84 & 43 & 59 & 8 & 60 & 62 & 14 & 51 \\ 45 & 52 & 73 & 69 & 46 & 67 & 6 & 52 & 56 & 27 \\ 43 & 56 & 36 & 95 & 5 & 50 & 6 & 86 & 0 & 10 \end{bmatrix}$$

goruldugu gibi resmi kucultup tekrar buyultmek kayiplara neden olmustur.
.

Bilinear Interpolation

Bilinear interpolation produces a smoother interpolation than does the nearest-neighbor approach. Given four neighboring image coordinates $f(n_{10}, n_{20})$, $f(n_{11}, n_{21})$, $f(n_{12}, n_{22})$, and $f(n_{13}, n_{23})$—these can be the four nearest neighbors of $f[\mathbf{a}(\mathbf{n})]$—then the geometrically transformed image $g(n_1, n_2)$ is computed as

$$g(n_1, n_2) = A_0 + A_1 n_1 + A_2 n_2 + A_3 n_1 n_2, \qquad (47)$$

which is a bilinear function in the coordinates (n_1, n_2). The bilinear weights A_0, A_1, A_2, and A_3 are found by solving

$$\begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} 1 & n_{10} & n_{20} & n_{10}n_{20} \\ 1 & n_{11} & n_{21} & n_{11}n_{21} \\ 1 & n_{12} & n_{22} & n_{12}n_{22} \\ 1 & n_{13} & n_{23} & n_{13}n_{23} \end{bmatrix}^{-1} \begin{bmatrix} f(n_{10}, n_{20}) \\ f(n_{11}, n_{21}) \\ f(n_{12}, n_{22}) \\ f(n_{13}, n_{23}) \end{bmatrix}. \qquad (48)$$