Filtri di convoluzione (blur)

```
imgfx=(blurred7+double(lena))/2;
figure; imshow(imgfx,[0,255]);
```

Lena

“collant”

Filtri di convoluzione (blur)

```
motion9h=fspecial('motion');
moved9h=conv2(lena, motion9h, 'same');
figure; imshow(moved9h,[0,255]);
motion15v=fspecial('motion',15,90);
moved15v=conv2(lena, motion15v, 'same');
figure; imshow(moved15v,[0,255]);
```

motion: 9 pixels, 0°

motion: 15 pixels, 90°

Filtri di convoluzione (denoise)

```
noised=imnoise(lena,'salt & pepper',0.1);
figure; imshow(noised,[0,255]);
denoised=conv2(noised, mean3, 'same');
figure; imshow(denoised,[0,255]);
```

“salt & pepper” noise (10%)

denoised (mean 3×3)

Filtri di convoluzione (denoise)

noised=imnoise(lena,'salt & pepper',0.1);

è equivalente a

```
p=randperm(512*512);
t=round(5*512*512/100);
noised=lena;
noised(p(1:t))=0;
noised(p(t+1:2*t))=255;
```

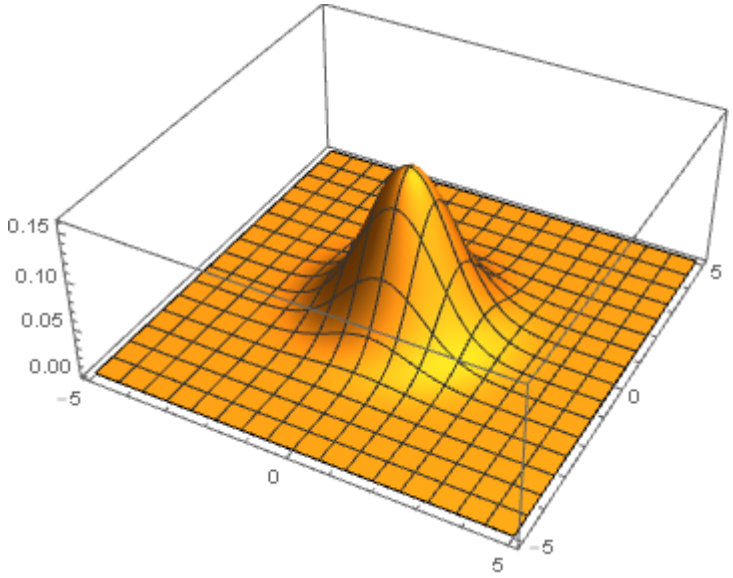
dove t=13107 corrisponde al 5% dei 512*512 pixel dell’immagine:

Filtri di convoluzione (Gaussian)

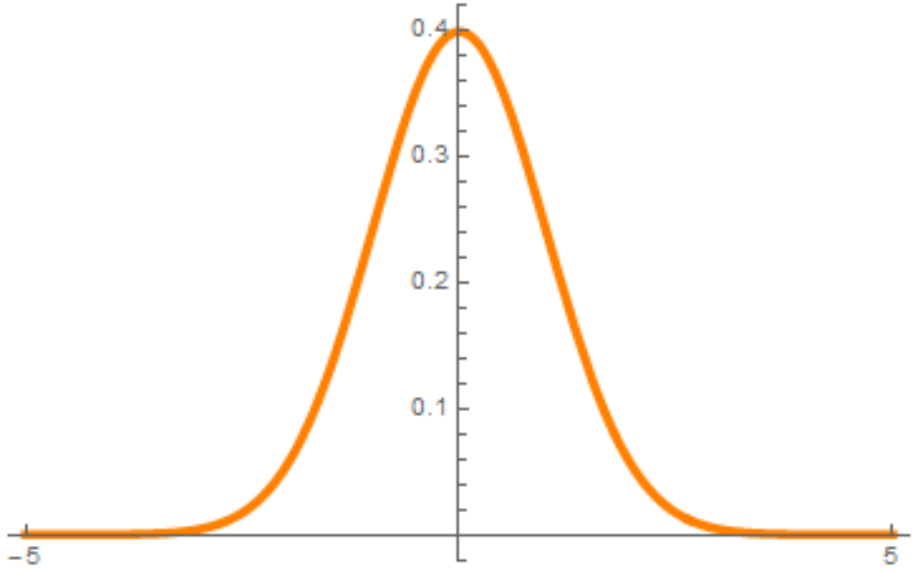
Il filtro di Gauss è a variabili separabili ($k_{ij}=k_i' \otimes k_j''$), dove $\otimes$ indica il prodotto tensoriale di Kronecker. Inoltre, grandi finestre possono essere approssimate con successivi filtraggi.

$$\text{Gauss}(x,y) = \frac{e^{-\frac{(x-\mu)^2+(y-\mu)^2}{2\sigma^2}}}{2\pi\sigma^2}$$

$$\text{Gauss}(x) = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}$$



μ=0, σ=1



μ=0, σ=1

Filtri di convoluzione (Gaussian)

Il filtro di Gauss è a variabili separabili ($k_{ij}=k_i' \otimes k_j''$), dove $\otimes$ indica il prodotto tensoriale di Kronecker. Inoltre, grandi finestre possono essere approssimate con successivi filtraggi.

$$\text{Gauss}(x,y)=\frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^2}$$

$$\text{Gauss}(x)=\frac{e^{-\frac{x^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}$$

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$\mu=0$, $\sigma=1$

$$\frac{1}{16}\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix}$$

$\mu=0$, $\sigma=1$

Filtri di convoluzione (Gaussian)

Il filtro di Gauss è a variabili separabili ($k_{ij}=k_i' \otimes k_j''$), dove $\otimes$ indica il prodotto tensoriale di Kronecker. Inoltre, grandi finestre possono essere approssimate con successivi filtraggi.

Gauss(x,y) Gauss(y)⊗Gauss(x) Gauss(x)⊗Gauss(y)

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix}$$

Filtri di convoluzione (denoise)

```
gaussian3=fspecial('gaussian',3);
blurred4=conv2(lena, gaussian3, 'same');
figure; imshow(blurred4,[0,255]);
gaussian9=fspecial('gaussian',9);
denoised2=conv2(noised, gaussian9, 'same');
figure; imshow(denoised2,[0,255]);
```

blurred (Gaussian 3×3, μ=0, σ=0.5)

denoised (Gaussian 9×9, μ=0, σ=0.5)

Filtri di non convoluzione (median)

```
median3=medfilt2(lena);
figure; imshow(median3,[0,255]);
denoised3=medfilt2(noised);
figure; imshow(denoised3,[0,255]);
```

blurred (median 3×3)

denoised (median 3×3)

Filtri di convoluzione (sharpen)

```
sharpen=fspecial('unsharp'); % [[-1 -4 -1]; [-4 26 -4]; [-1 -4 -1]]/6 high-pass filter
sharpened=conv2(lena, sharpen, 'same');
figure; imshow(sharpened,[0,255]);
```

Lena

sharpened

Filtri di convoluzione (sharpen)

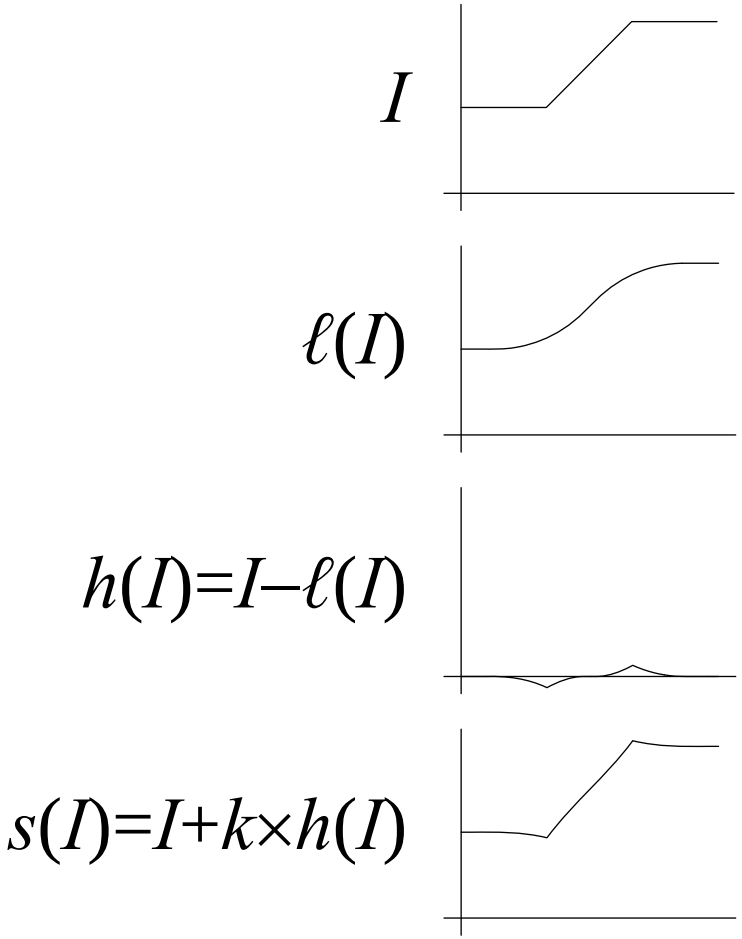
```
avg=fspecial('average'); % [[1 1 1]; [1 1 1]; [1 1 1]]/9 low-pass filter
sharpened2=lena+0.7*(lena-conv2(lena, avg, 'same'));
figure; imshow(sharpened2,[0,255]);
```

Lena

sharpened by averaging

Filtri di convoluzione (sharpen)

In pratica, questo filtro amplifica le alte frequenze del segnale. Si invita ad un confronto col filtro “collant”.



Filtro gradiente

Definiamo il gradiente di una funzione continua e derivabile, di due variabili reali:

$$\Delta I = \frac{\partial I}{\partial x} i + \frac{\partial I}{\partial y} j$$

Nel caso discreto le componenti diventano:

$$\frac{\partial D}{\partial x} \approx D(i, j+1) - D(i, j) = \Delta_x D = \begin{bmatrix} -1 & 1 \end{bmatrix}, \quad \frac{\partial D}{\partial y} \approx D(i+1, j) - D(i, j) = \Delta_y D = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Il modulo e la direzione del vettore gradiente sono rispettivamente:

$$|I| = \sqrt{\Delta_x^2 I + \Delta_y^2 I} \qquad \measuredangle I = \arctan \frac{\Delta_y}{\Delta x}$$

Filtro gradiente (Prewitt)

$$\Delta_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \Delta_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Sfruttando la separabilità delle variabili, la convoluzione diviene monodimensionale:

$$\Delta_x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \Delta_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

Filtro gradiente (Sobel)

$$\Delta_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \Delta_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sfruttando la separabilità delle variabili, la convoluzione diviene monodimensionale:

$$\Delta_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \Delta_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Filtri di convoluzione (Sobel)

```
sobelk=[-1,-2,-1; 0,0,0; 1,2,1];
sobelx=conv2(lena, sobelk, 'same');
figure; imshow(sobelx,[-1020,1020]);
sobely=conv2(lena, sobelk', 'same');
figure; imshow(sobely,[-1020,1020]);
```

sobelx (horizontal component)

sobely (vertical component)

Filtri di convoluzione (Sobel)

```
sobeleuclidean=sqrt(sobelx.^2 + sobely.^2)./sqrt(32);
figure; imshow(sobeleuclidean,[0,255]);
sobelcityblock=(abs(sobelx) + abs(sobely))./8;
figure; imshow(sobelcityblock,[0,255]);
```

Euclidean magnitude

“city block” magnitude

Filtri di convoluzione (Sobel)

```
sobelangle=atan2(sobely, sobelx);
figure; imshow(sobelangle,[-pi,pi]);
```

Lena

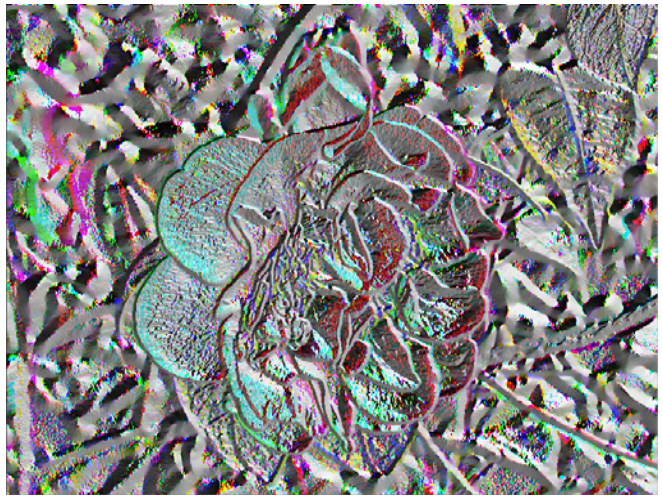
orientation

Filtri di convoluzione (Sobel)

Gli algoritmi fin qui visti possono essere applicati alle componenti colore e i risultati possono essere composti in immagini a colori.

input image

“city block” magnitude



orientation

Filtro laplaciano

Definiamo il laplaciano di una funzione continua e derivabile due volte, di due variabili reali:

$$\Delta^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Nel caso discreto le componenti diventano:

$$\begin{aligned}
\frac{\partial^2 D}{\partial x^2} &\approx \frac{\partial (D(i,j+1) - D(i,j))}{\partial x} = \frac{\partial D(i,j+1)}{\partial x} - \frac{\partial D(i,j)}{\partial x} = \\
&= (D(i,j+2) - D(i,j+1)) - (D(i,j+1) - D(i,j)) = \\
&= D(i,j+2) - 2D(i,j+1) + D(i,j) \\
\frac{\partial^2 D}{\partial y^2} &\approx D(i+2,j) - 2D(i+1,j) + D(i,j)
\end{aligned}$$

Filtro laplaciano

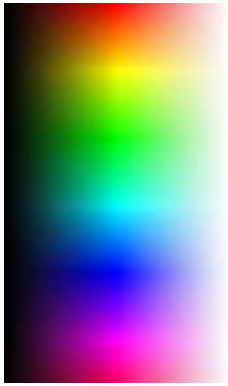
Effettuando la traslazione in (i, j):

$$\frac{\partial^2 D}{\partial x^2} \approx D(i, j+1) - 2D(i, j) + D(i, j-1) = \Delta_x^2 D$$
$$\frac{\partial^2 D}{\partial y^2} \approx D(i+1, j) - 2D(i, j) + D(i-1, j) = \Delta_y^2 D$$

Da cui:

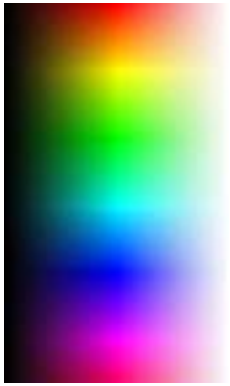
$$\Delta_x^2 = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \qquad \Delta_y^2 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Quantizzazione



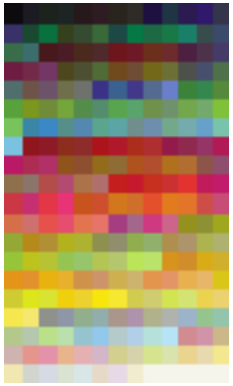
True colors (24 bpp)

Quantizzazione



True colors (16 bpp)

Quantizzazione



Indexed colors (8 bpp)

Quantizzazione

Indexed colors (4 bpp)

Quantizzazione

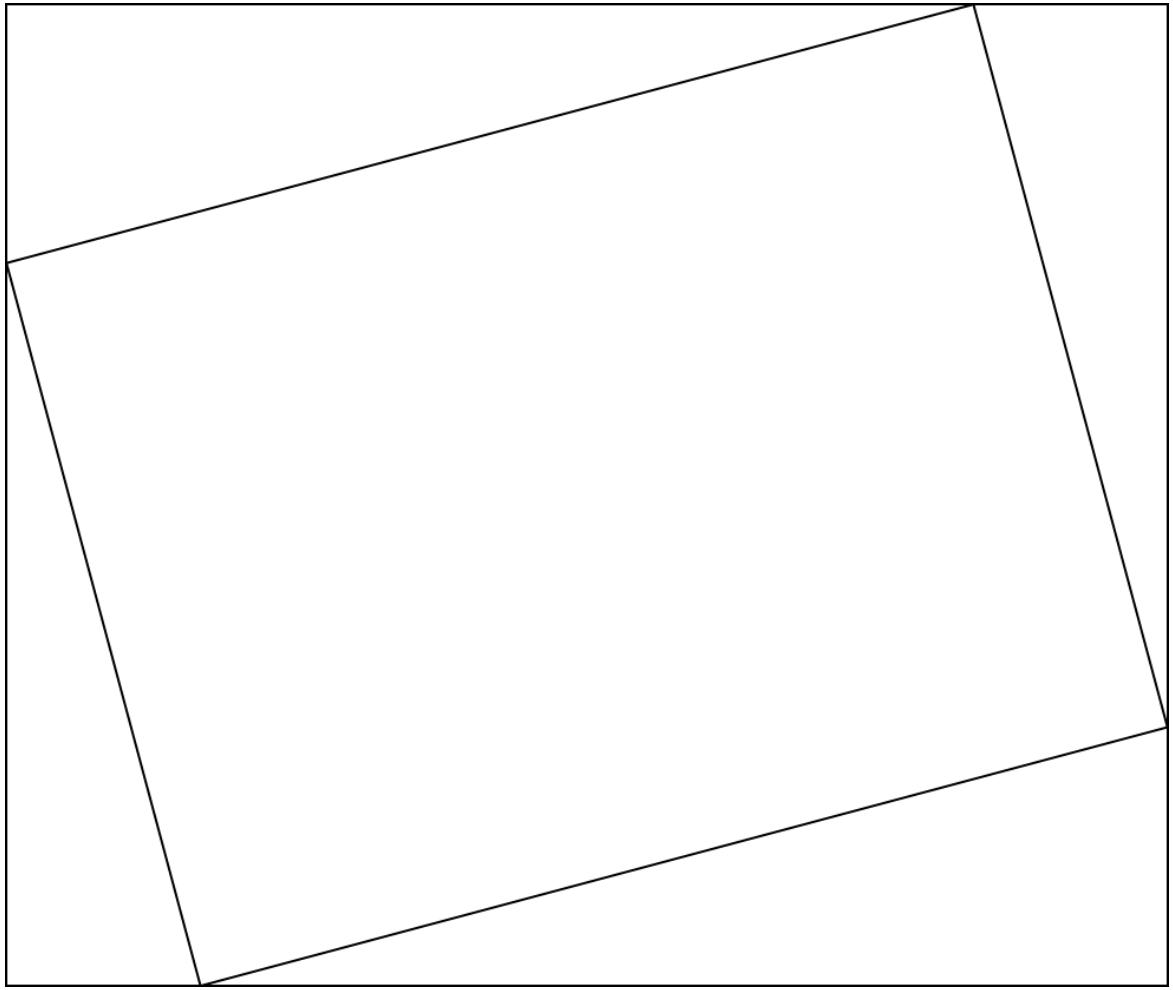
Indexed colors (2 bpp)

Quantizzazione

Indexed colors (1 bpp)

Rotazione

$$\begin{cases} X = \left[x\cos\alpha - y\sin\alpha \right] \\ Y = \left[x\sin\alpha + y\cos\alpha \right] \end{cases}$$



Rotazione

$$\begin{cases} X = \lfloor x\cos\alpha - y\sin\alpha \rfloor \\ Y = \lfloor x\sin\alpha + y\cos\alpha \rfloor \end{cases}$$

Rotazione

$$\begin{cases} x = \left[X\cos\alpha + Y\sin\alpha \right] \\ y = \left[Y\cos\alpha - X\sin\alpha \right] \end{cases}$$

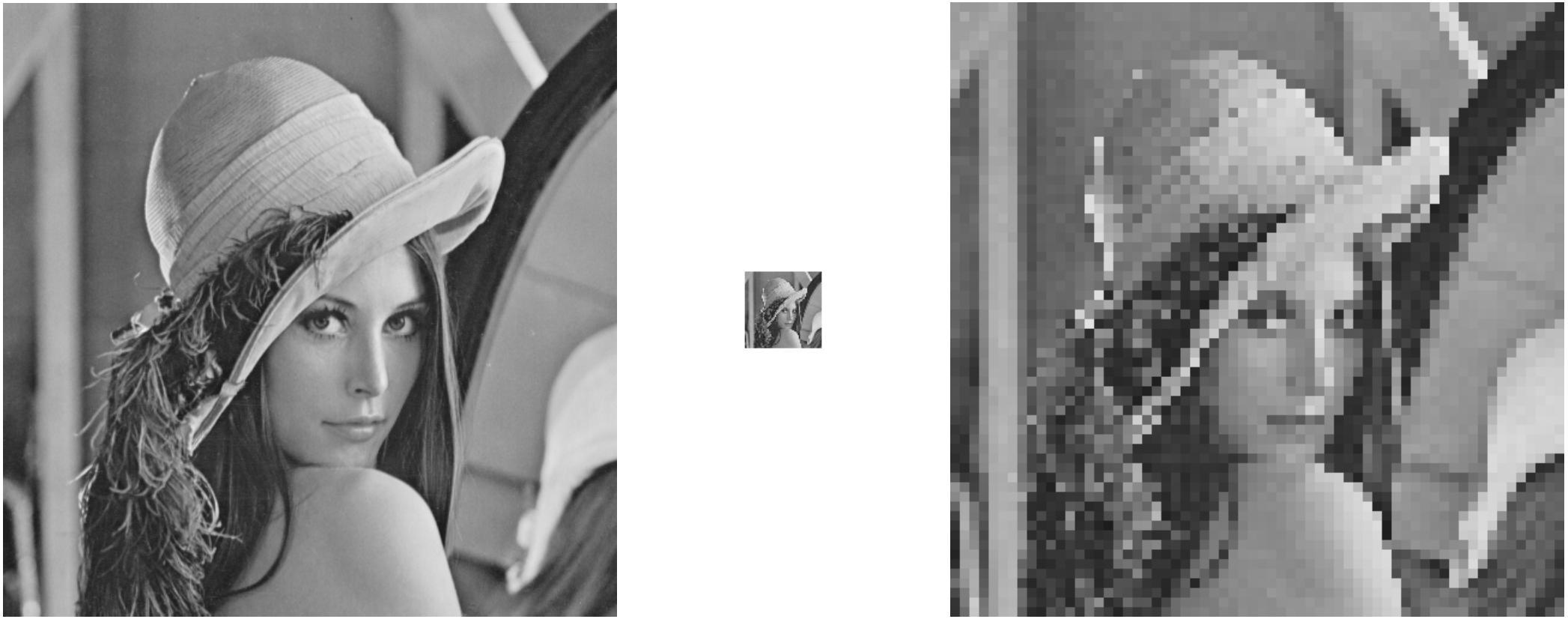
Ridimensionamento

512×512 → 256×256 (nearest) → 512×512 (nearest)

Ridimensionamento

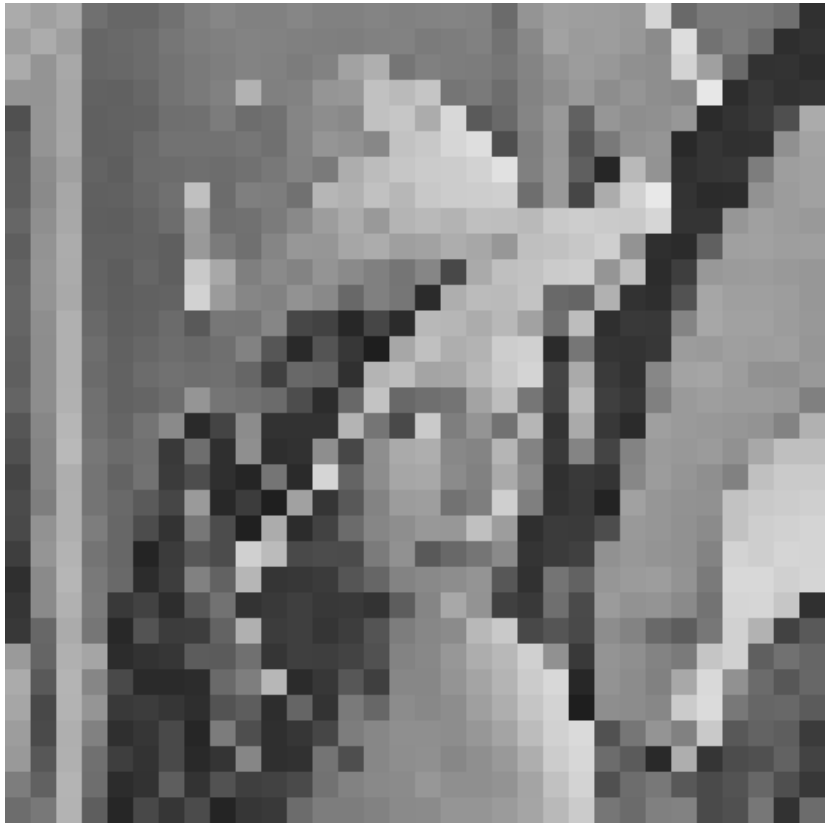
512×512 → 128×128 (nearest) → 512×512 (nearest)

Ridimensionamento



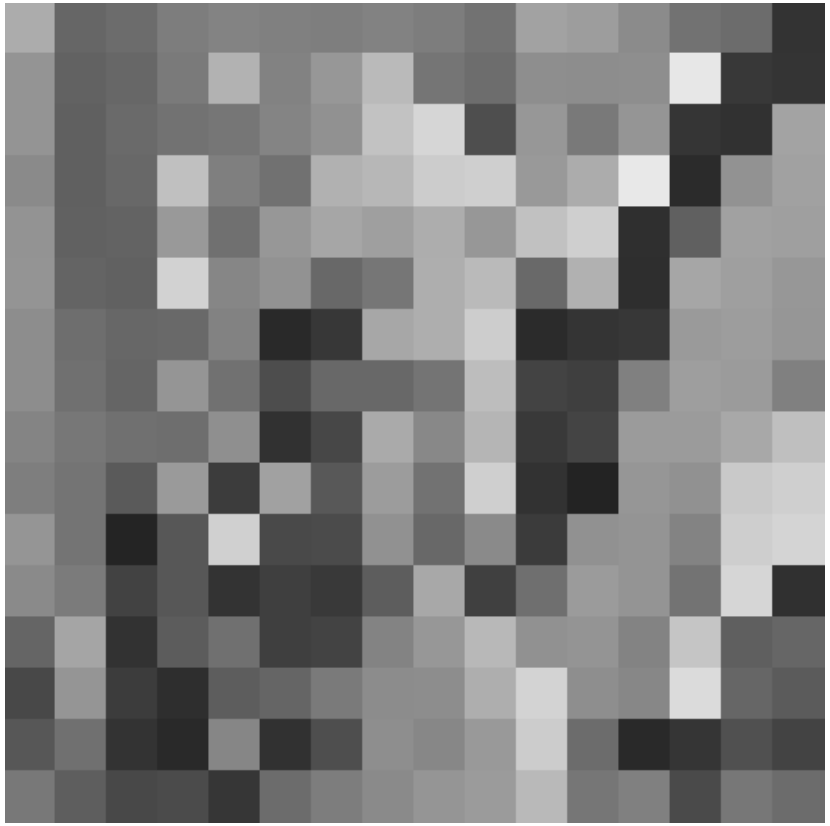
512×512 → 64×64 (nearest) → 512×512 (nearest)

Ridimensionamento



512×512 → 32×32 (nearest) → 512×512 (nearest)

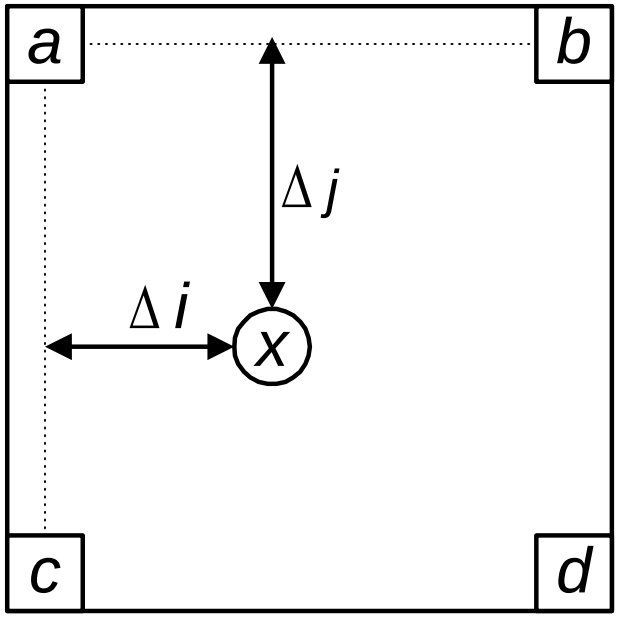
Ridimensionamento



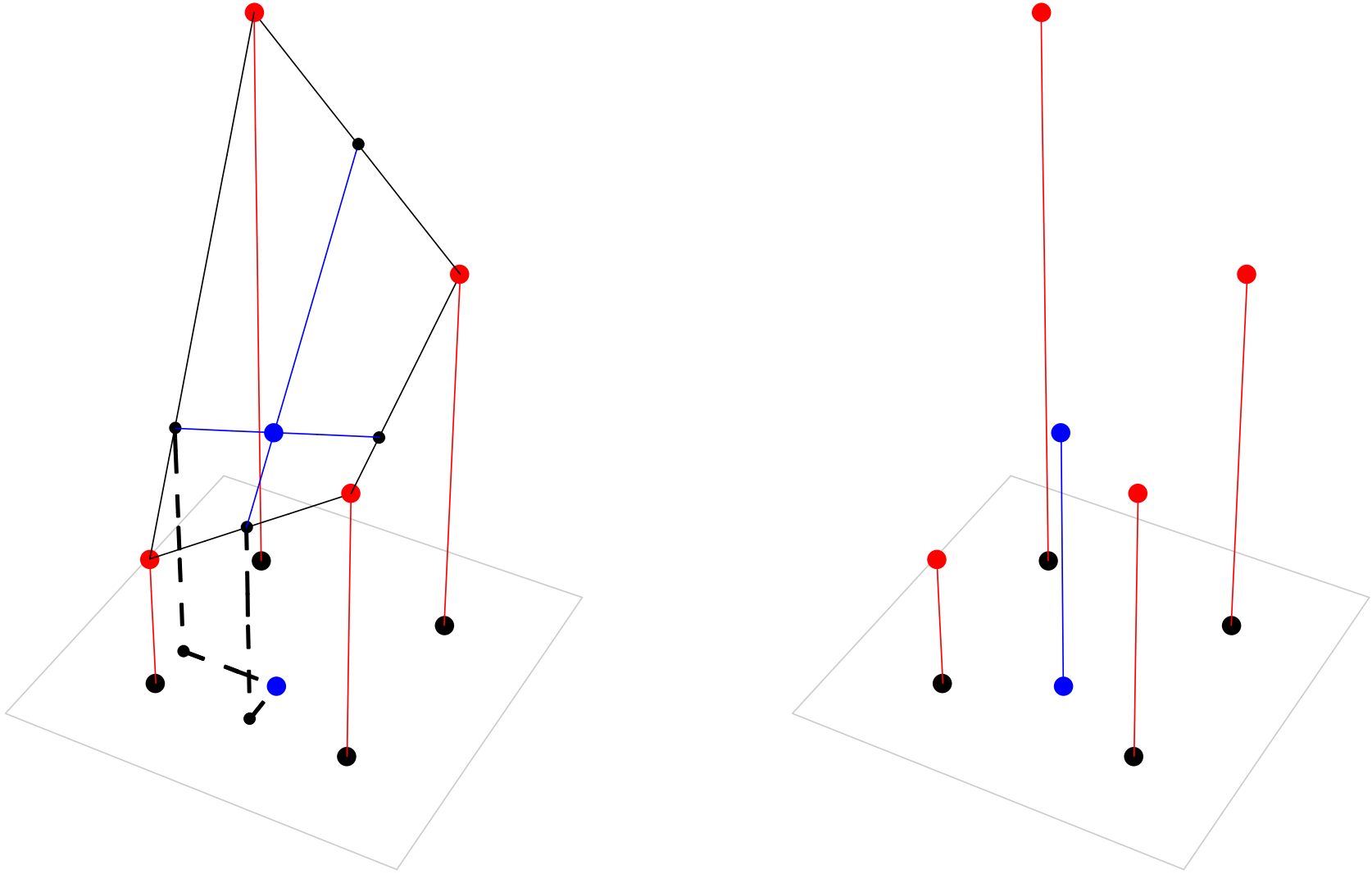
512×512 → 16×16 (nearest) → 512×512 (nearest)

Bilinear interpolation

$$x = a + \Delta i\left(b - a\right) + \Delta j\left(c - a\right) + \Delta i \Delta j\left(a - b - c + d\right)$$



Bilinear interpolation



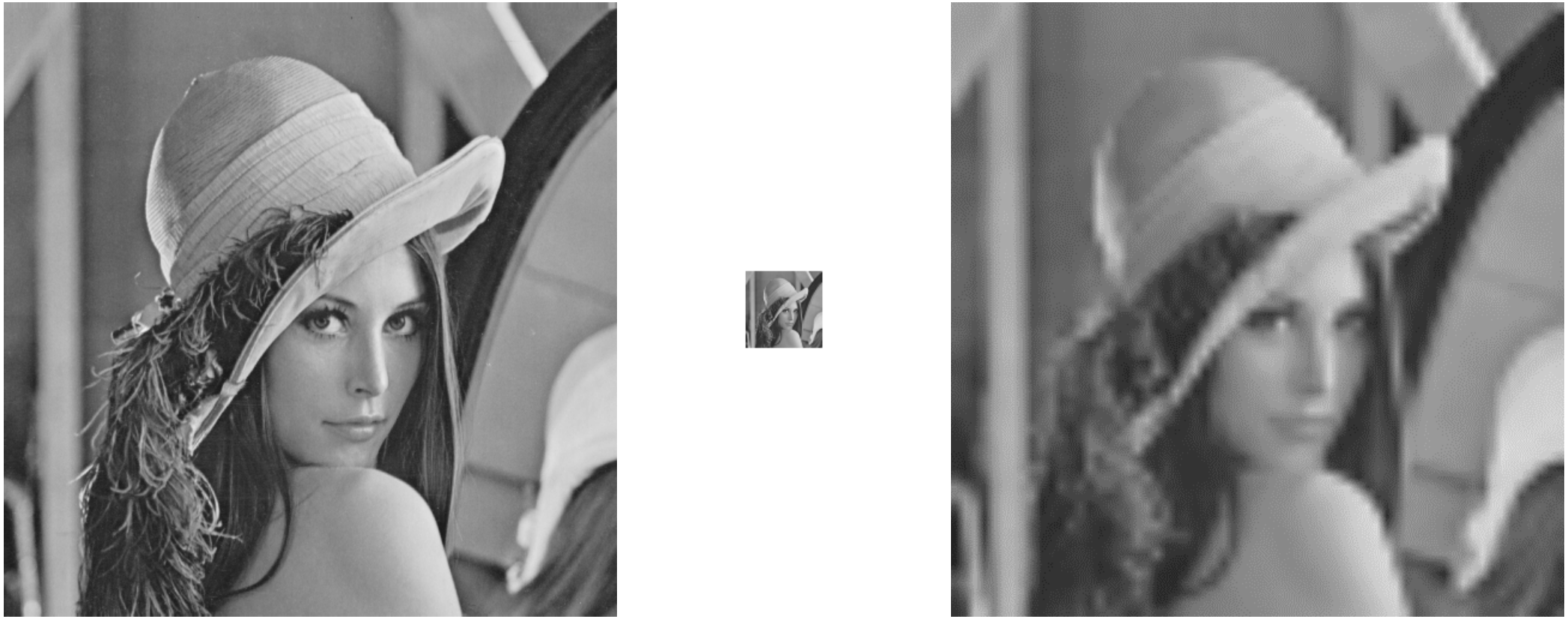
Ridimensionamento

512×512 → 256×256 (bilinear) → 512×512 (bilinear)

Ridimensionamento

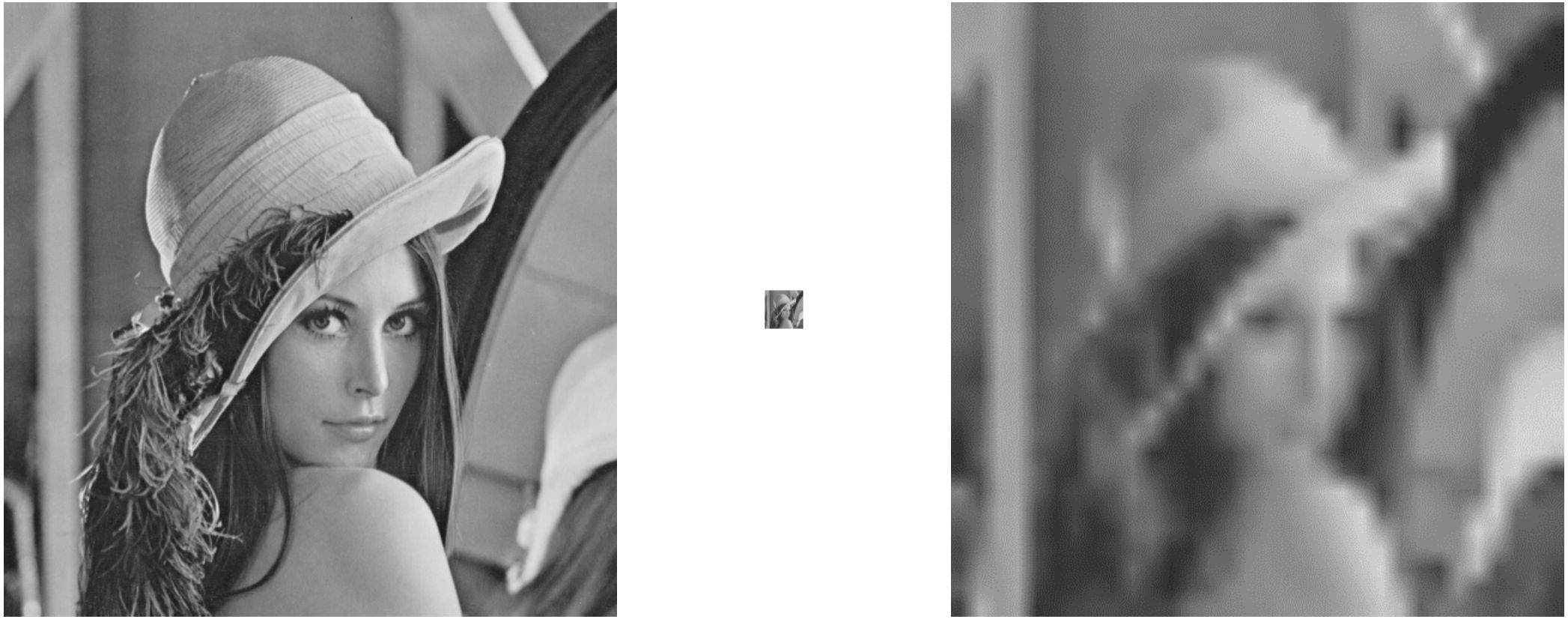
512×512 → 128×128 (bilinear) → 512×512 (bilinear)

Ridimensionamento



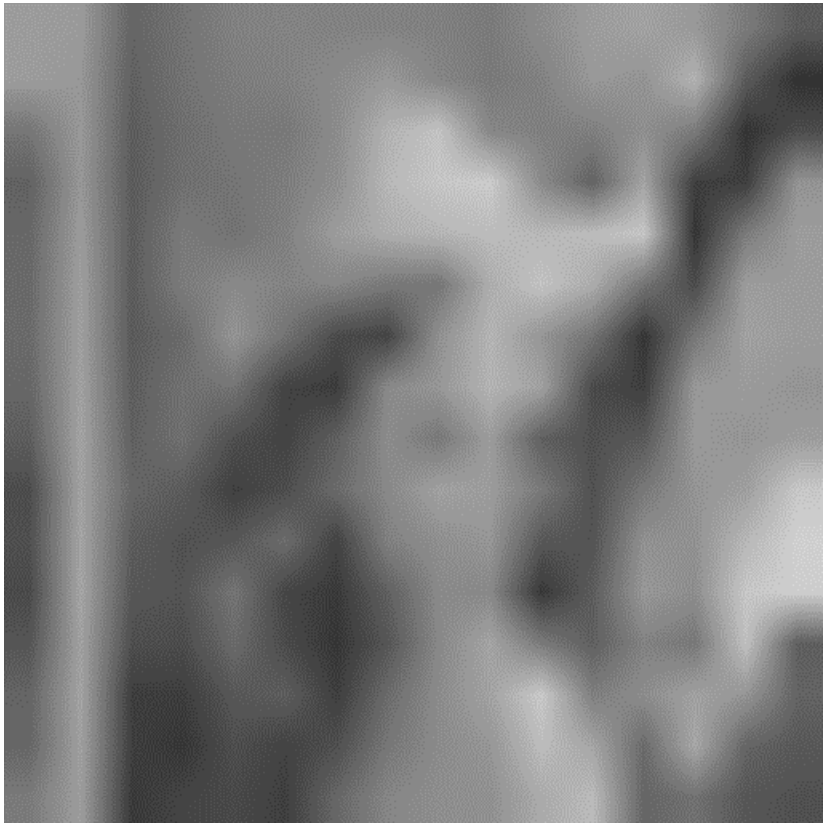
512×512 → 64×64 (bilinear) → 512×512 (bilinear)

Ridimensionamento



512×512 → 32×32 (bilinear) → 512×512 (bilinear)

Ridimensionamento



512×512 → 16×16 (bilinear) → 512×512 (bilinear)