EECE\CS 253 Image Processing

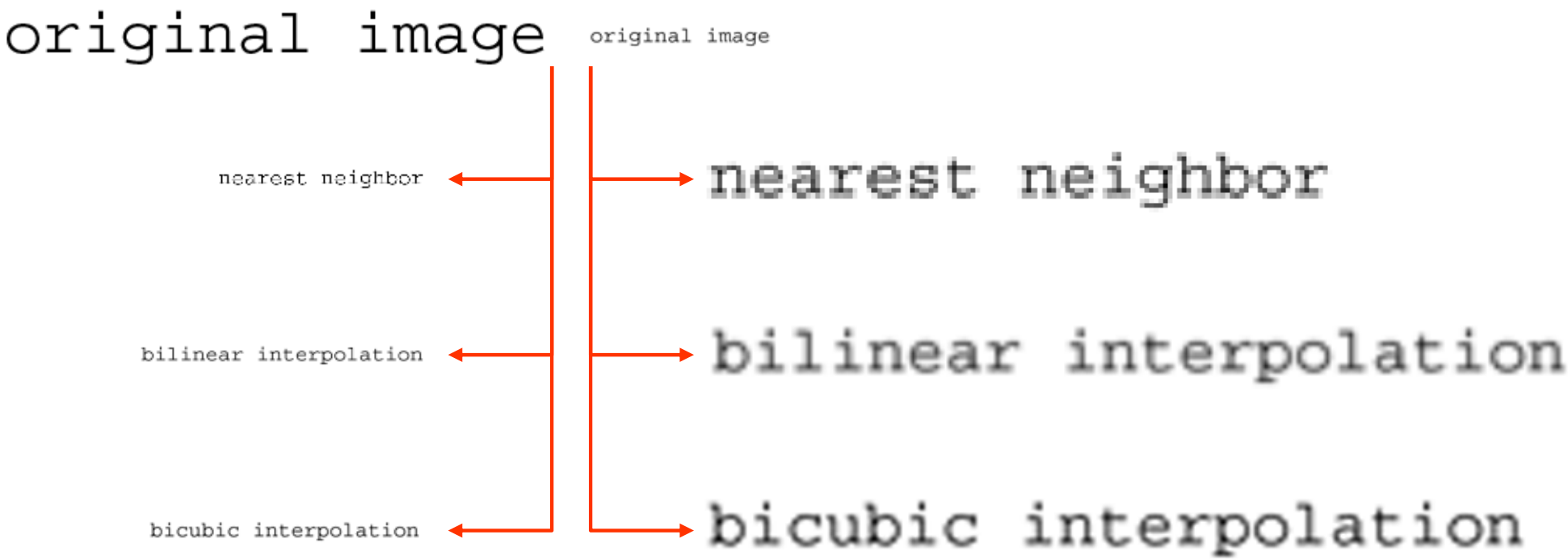
Lecture Notes: Resizing Images

Richard Alan Peters II

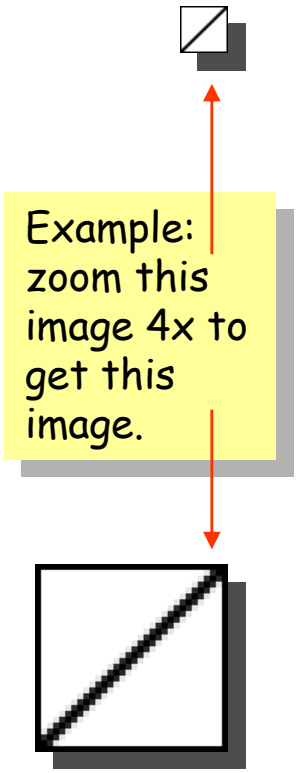
Department of Electrical Engineering and
Computer Science

Fall Semester 2007

Three Methods for Resizing Images

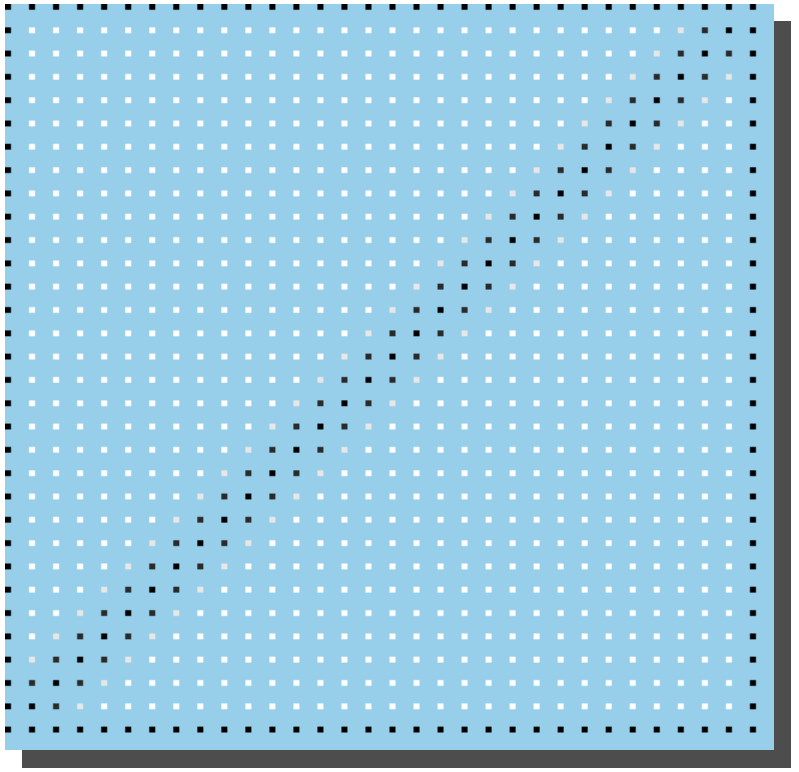


Enlarging Images Through Pixel Replication



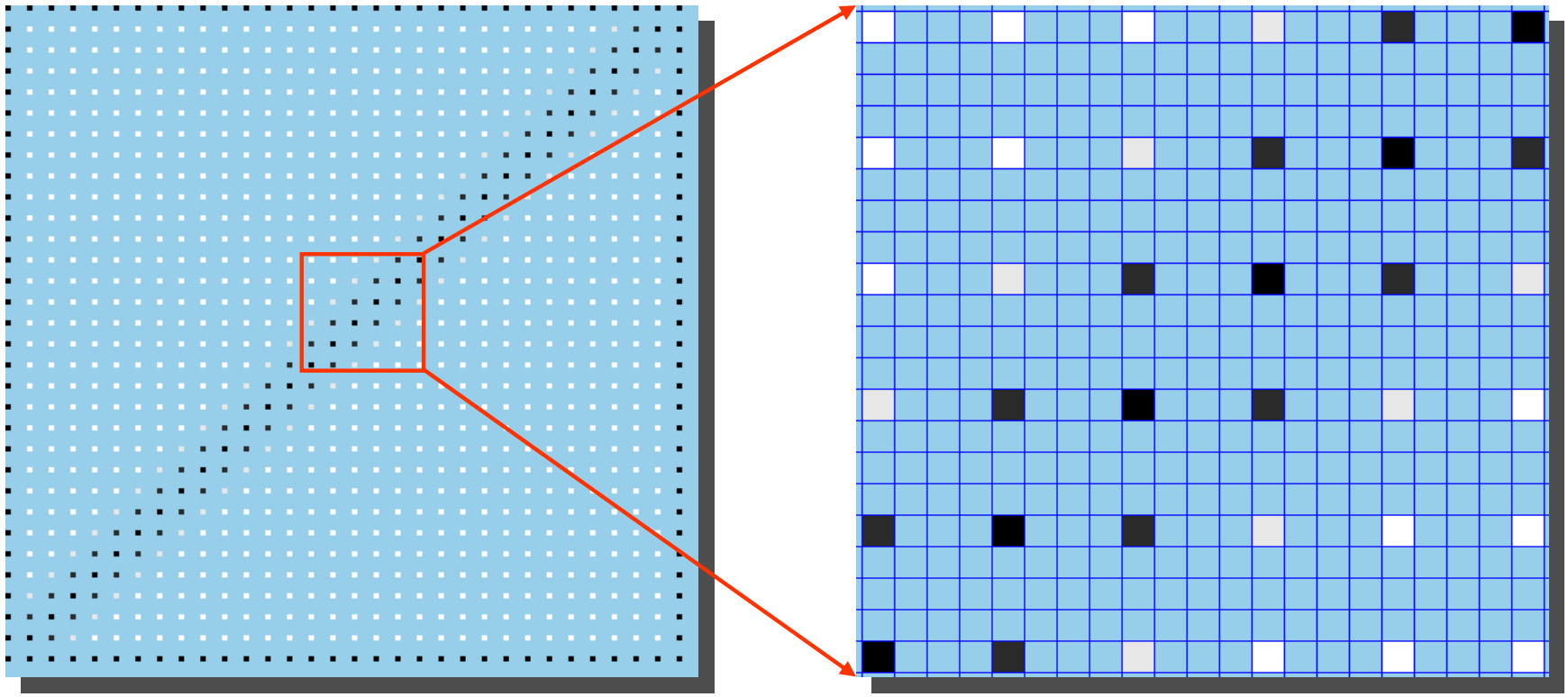
Example: zoom this image 4x to get this image.

Start with a blank image 4 times the linear dimensions of the original.



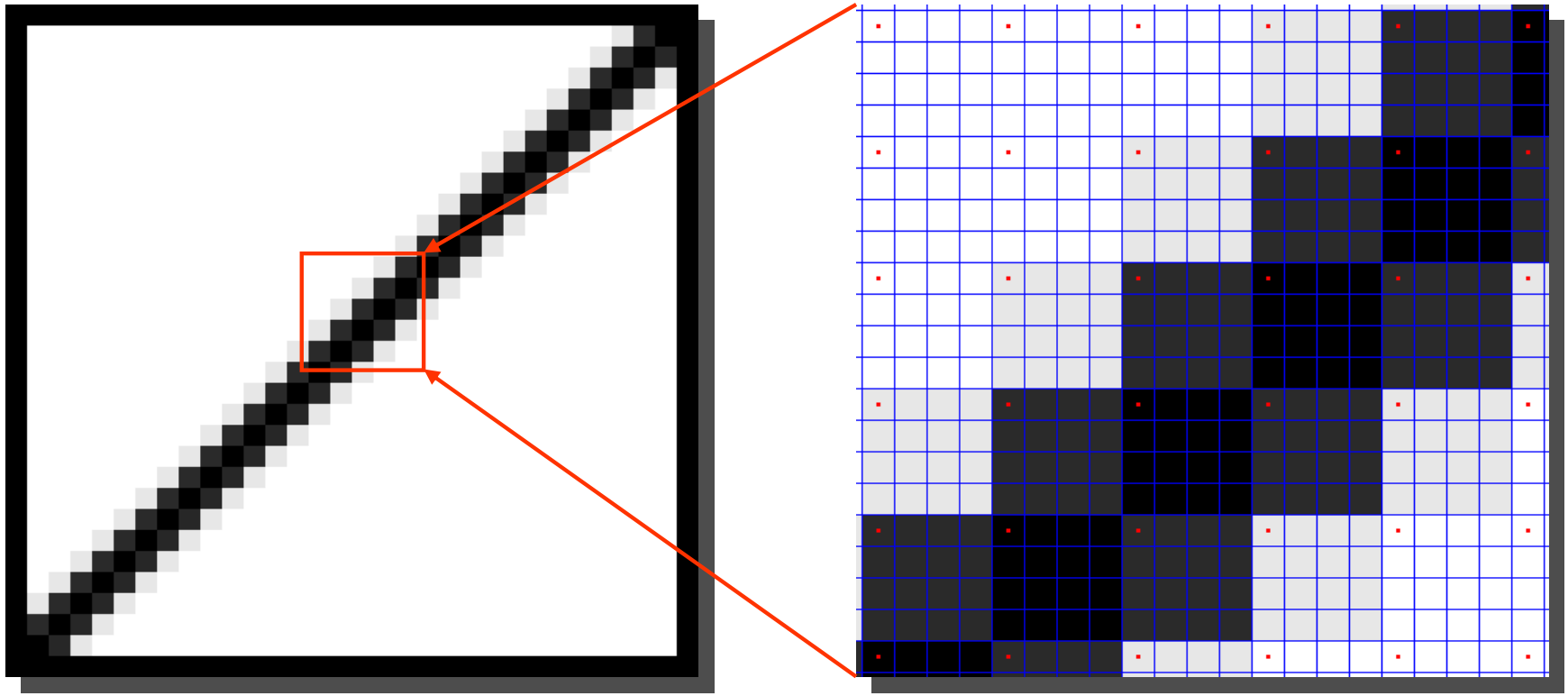
Fill in every 4th pixel in every 4th row with the original pixel values.

Enlarging Images Through Pixel Replication



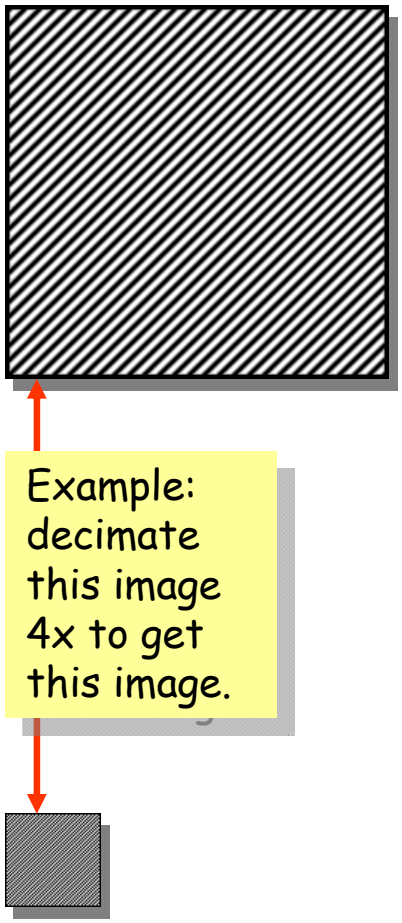
Detail showing every 4th pixel in every 4th row with the original pixel values.

Enlarging Images Through Pixel Replication



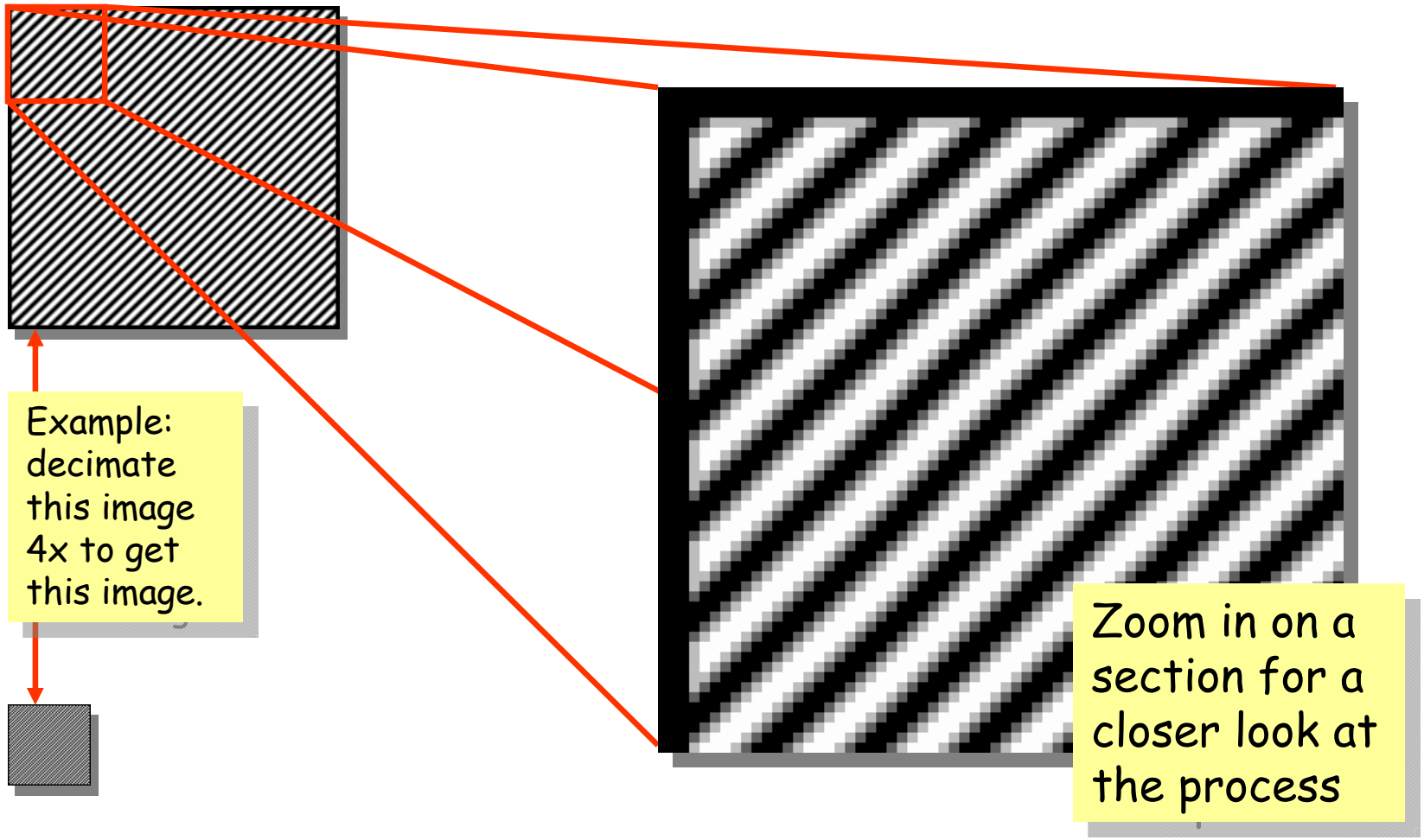
For each original value: replicate it 15 times to create a new, larger "pixel".

Reducing Images Through Pixel Decimation

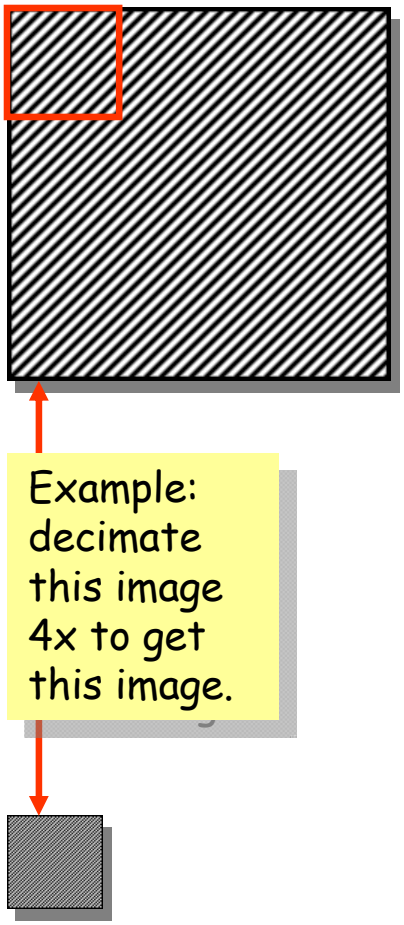


Decimation by a factor of *n*: take every *n*th pixel in every *n*th row

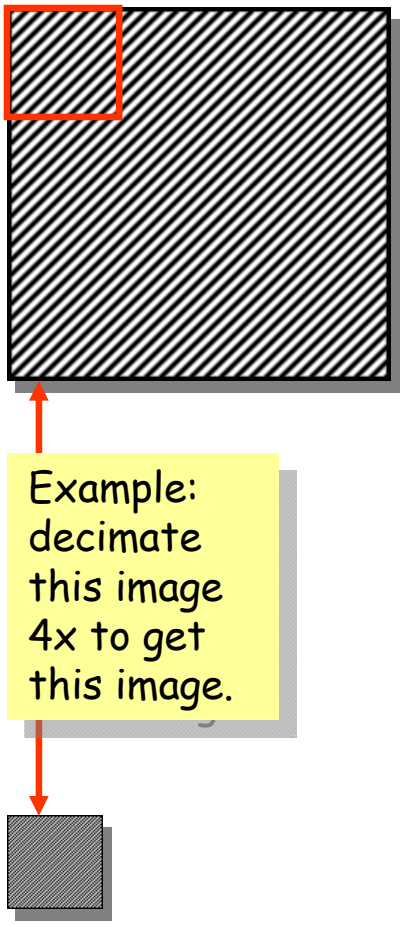
Reducing Images Through Pixel Decimation

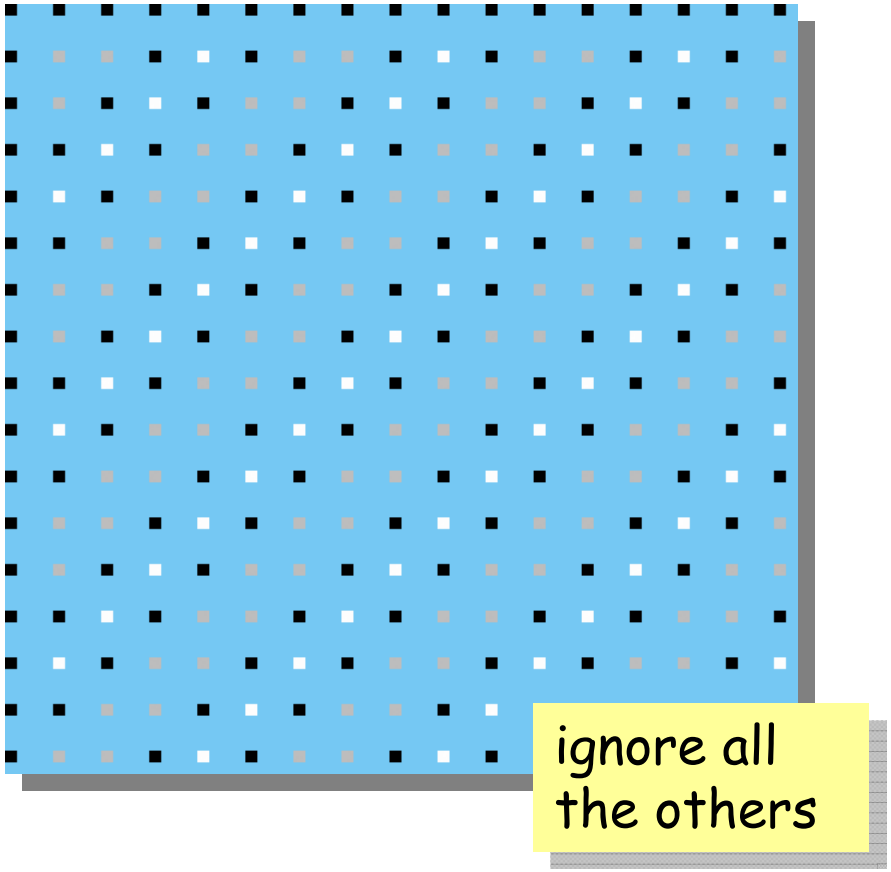


Reducing Images Through Pixel Decimation

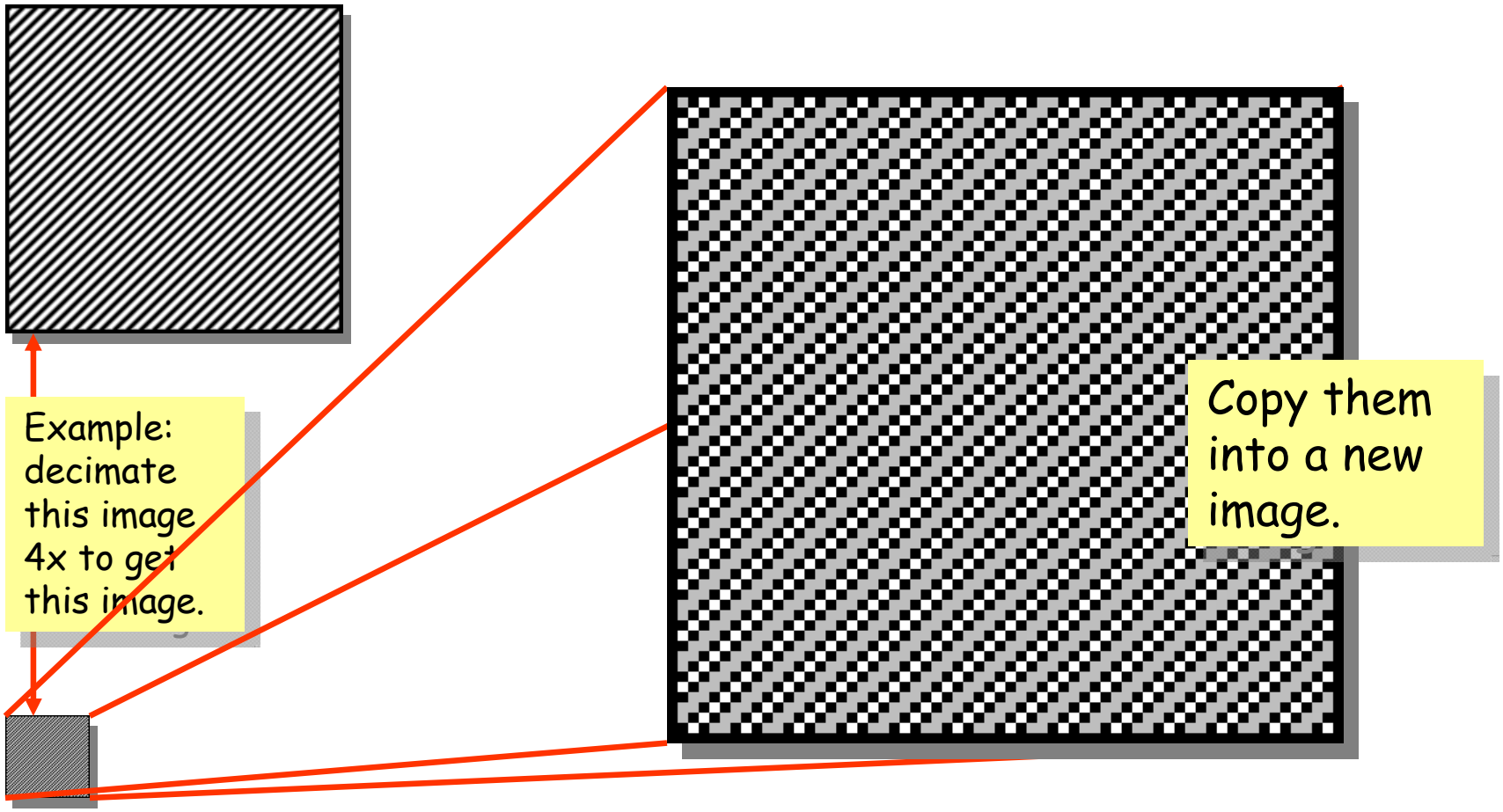


Reducing Images Through Pixel Decimation

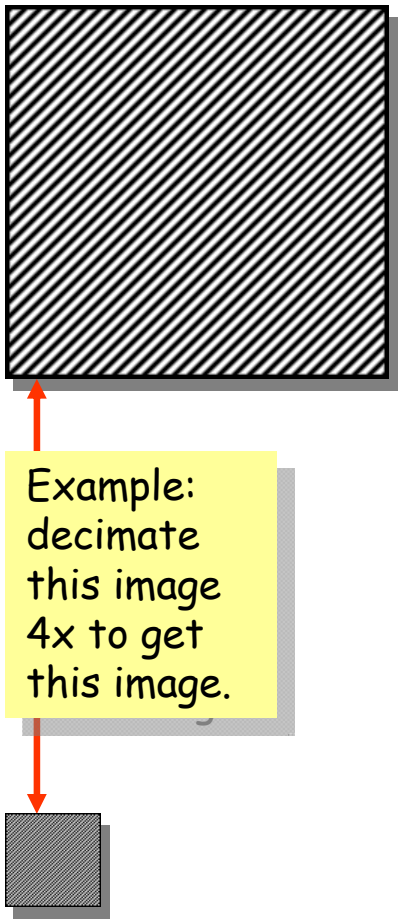


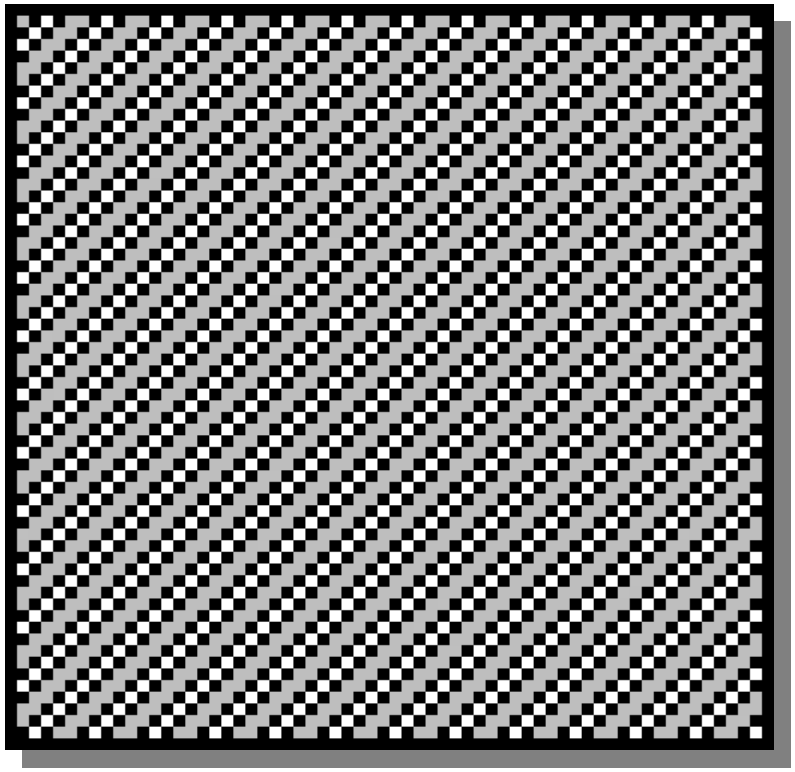


Reducing Images Through Pixel Decimation



Reducing Images Through Pixel Decimation





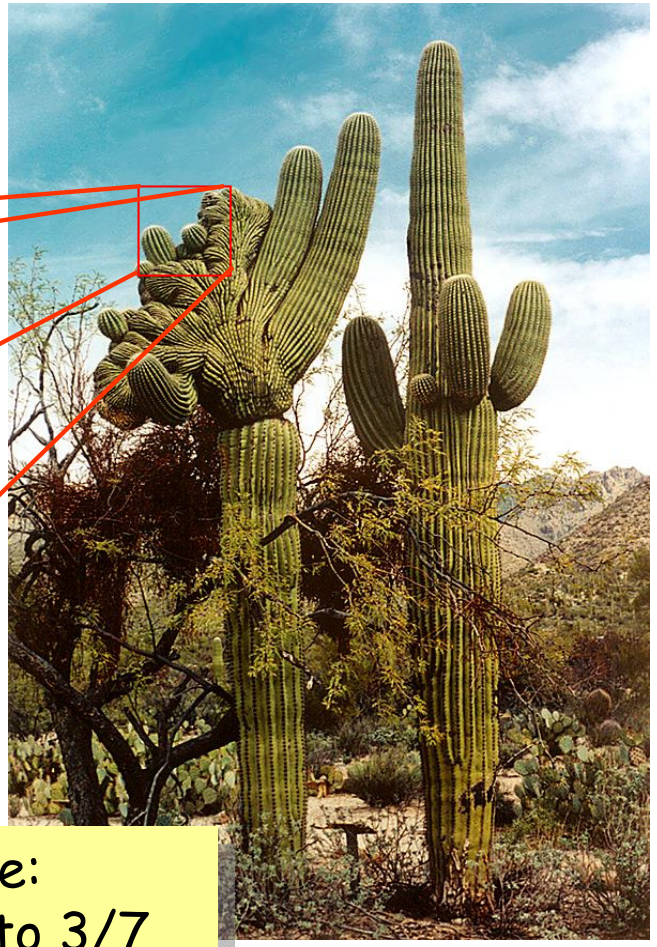
Nearest Neighbor Resampling

The "Nearest Neighbor" algorithm is a generalization of pixel replication and decimation.

It also includes fractional resizing, *i.e.* resizing an image so that it has p/q of the pixels per row and p/q of the rows in the original. (p and q are both integers.)

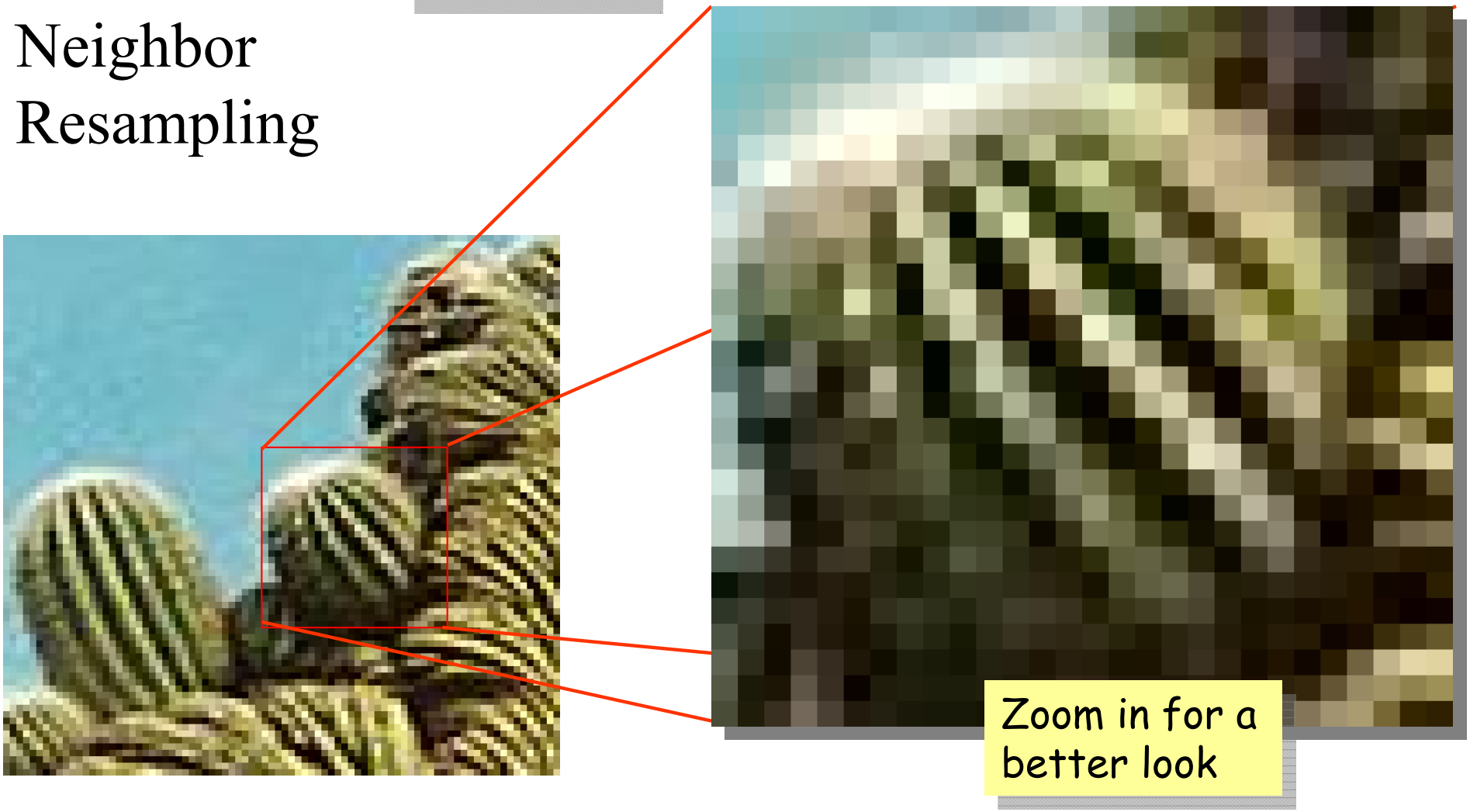
Nearest Neighbor Resampling

Zoom in on a section for a closer look at the process



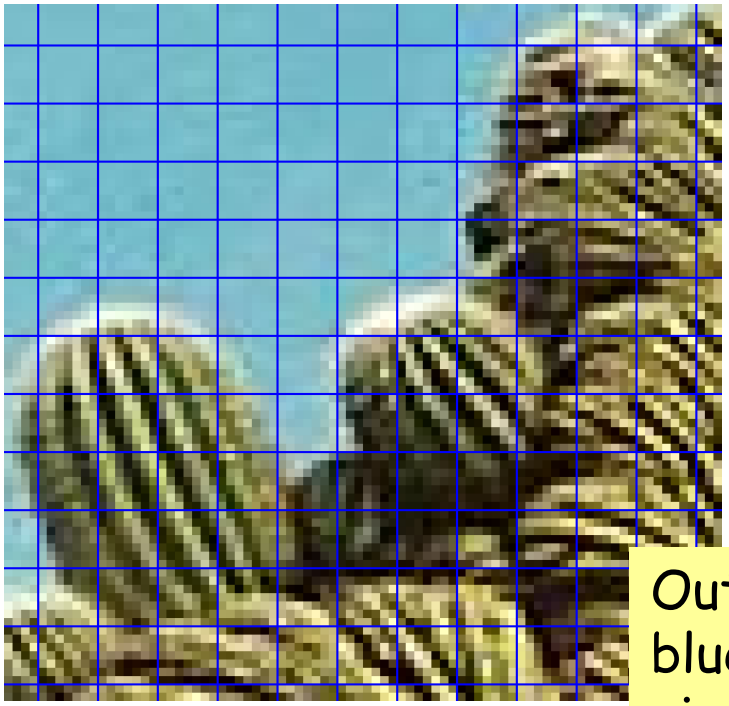
Example: resize to 3/7 of the original

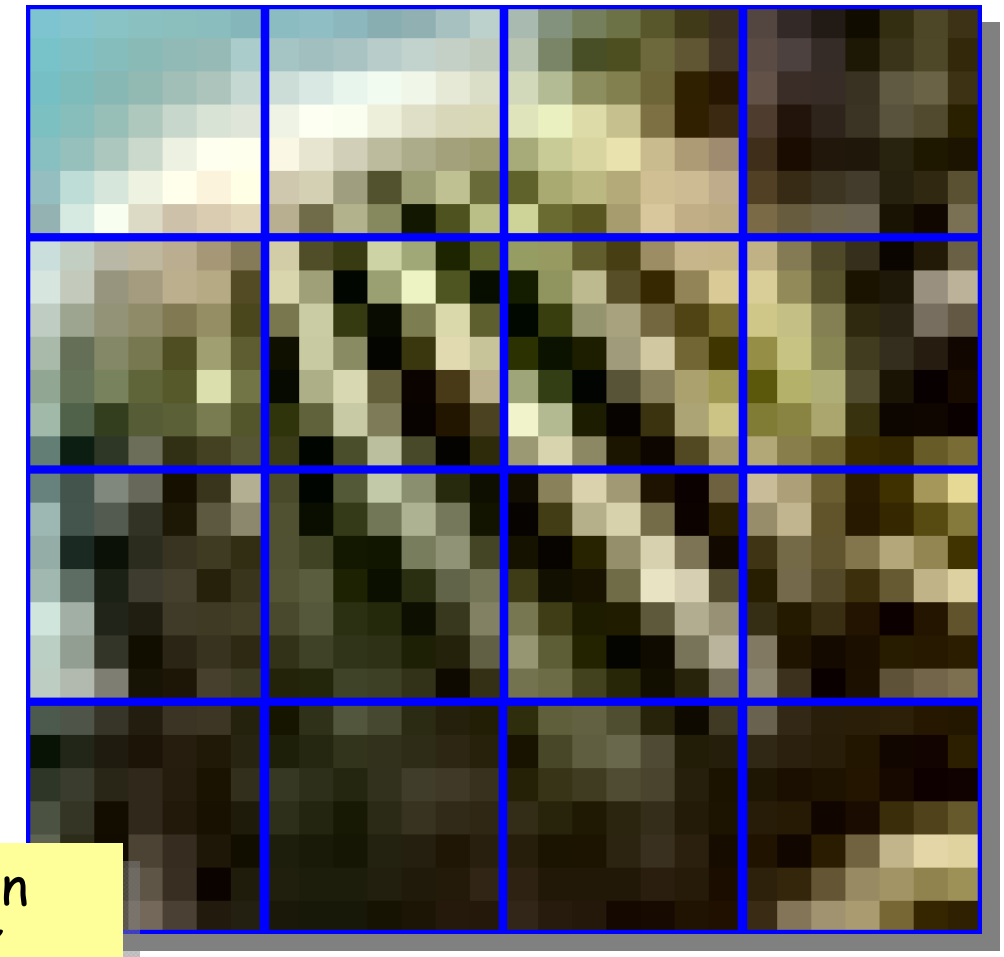
Nearest Neighbor Resampling



Nearest Neighbor Resampling

3/7 resize

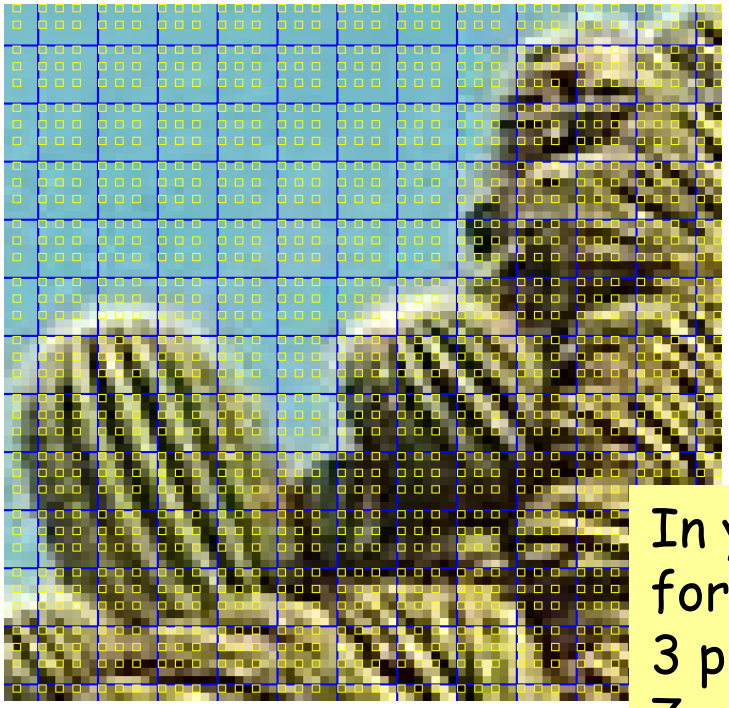


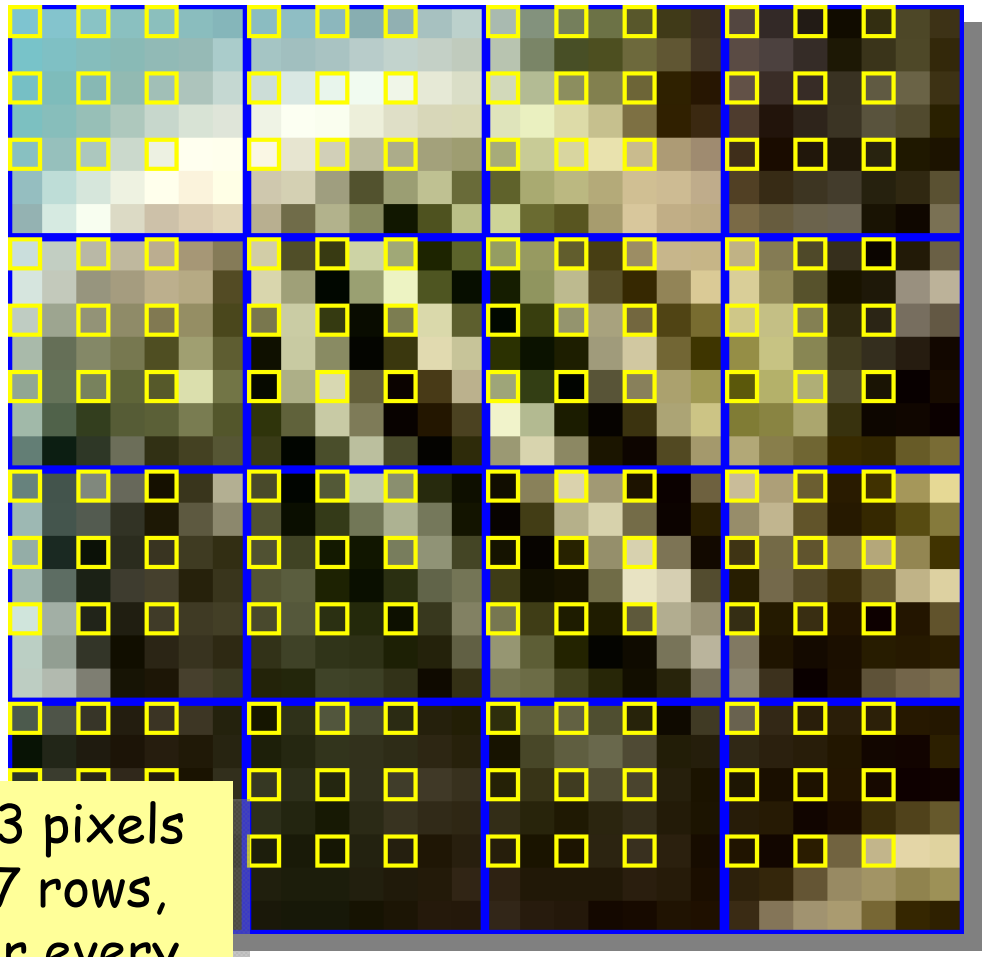


Outlined in blue: 7x7 pixel squares

Nearest Neighbor Resampling

3/7 resize





In yellow: 3 pixels for every 7 rows, 3 pixels for every 7 cols.

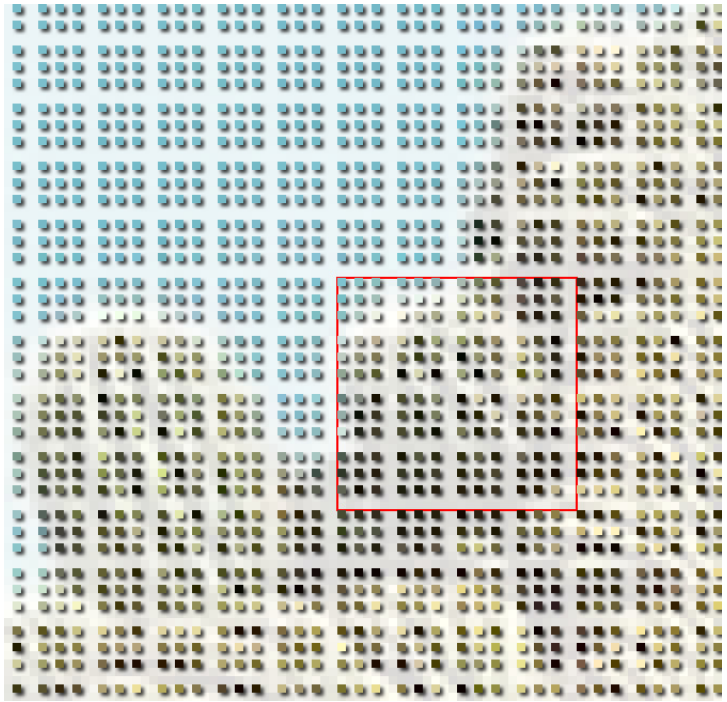
Nearest Neighbor Resampling

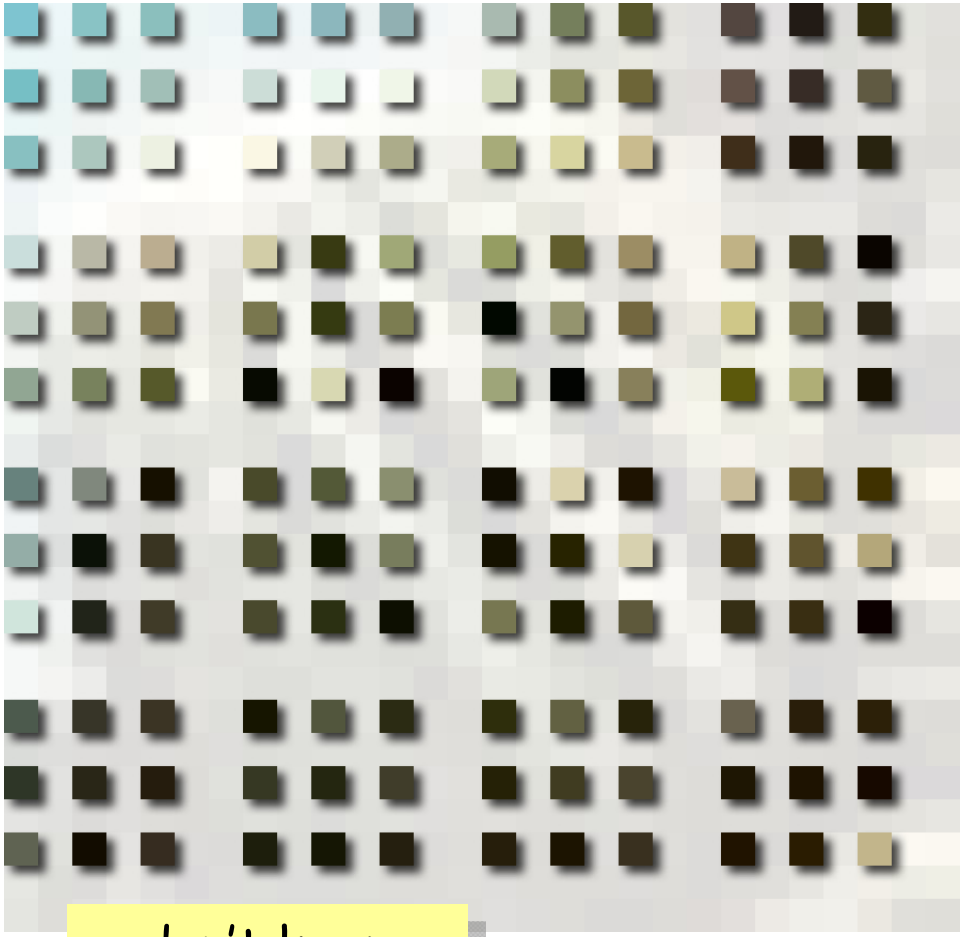
3/7 resize

Keep the highlighted pixels…

Nearest Neighbor Resampling

3/7 resize

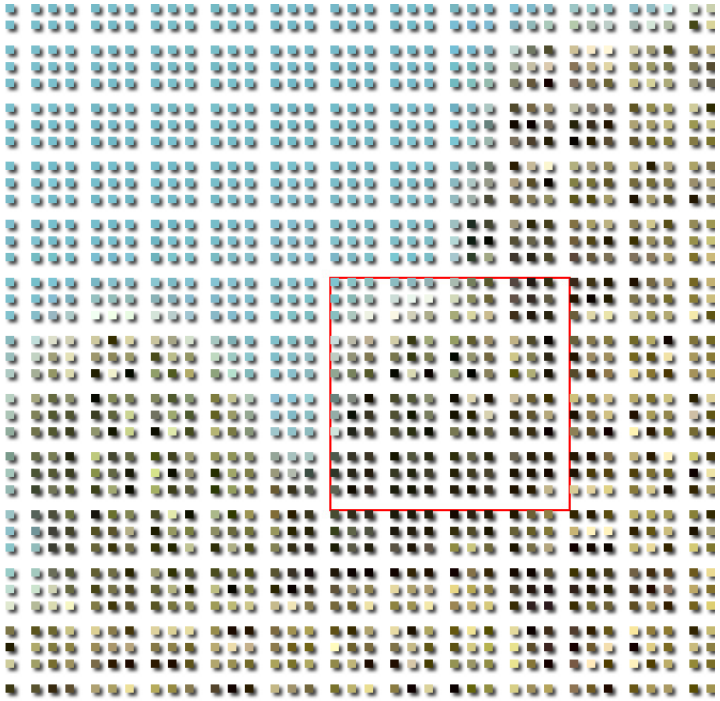


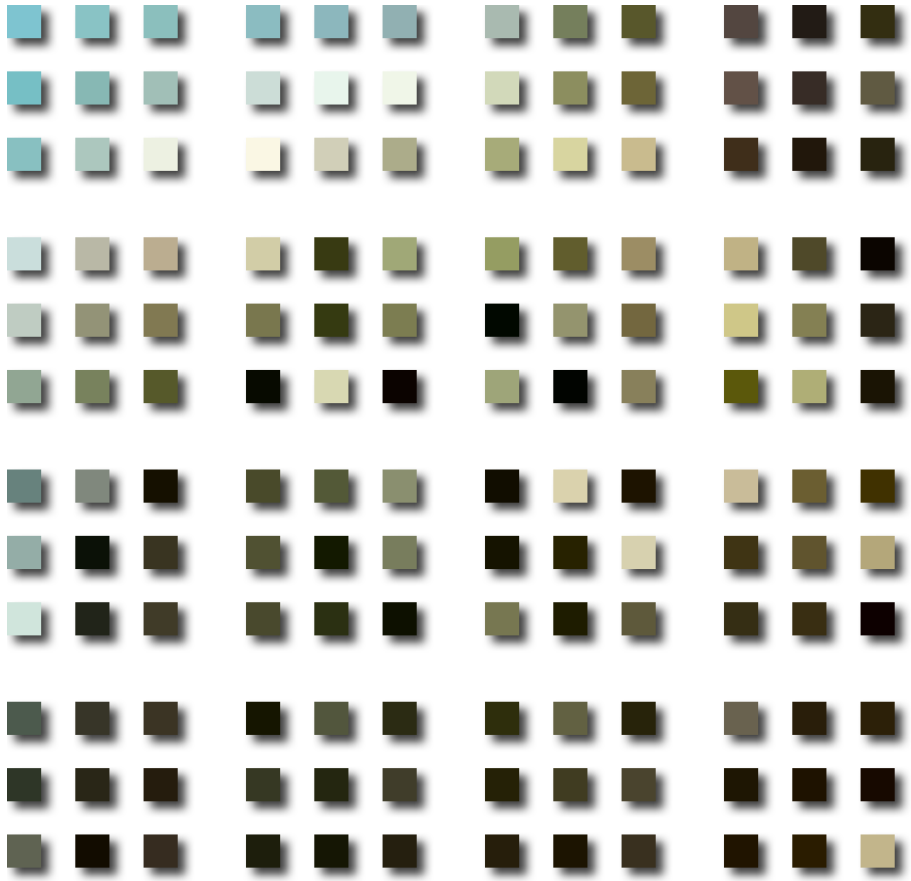


… don't keep the others.

Nearest Neighbor Resampling

3/7 resize

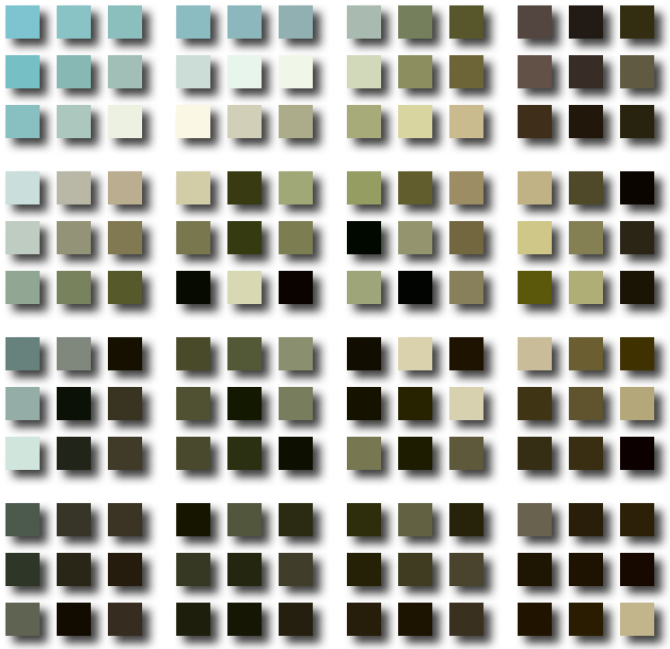




Copy them into a new image.

Nearest Neighbor Resampling

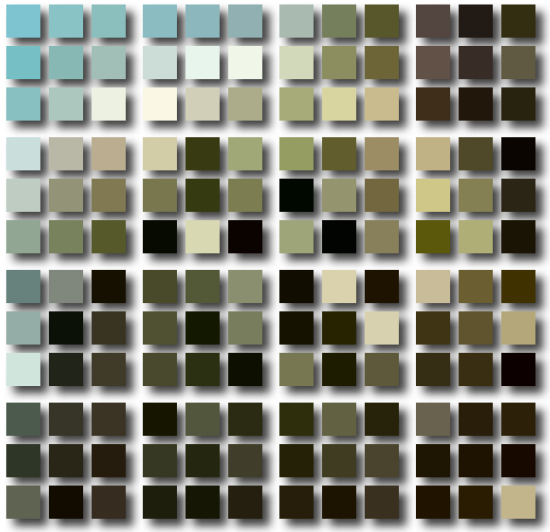
3/7 resize



Copy them into a new image.

Nearest Neighbor Resampling

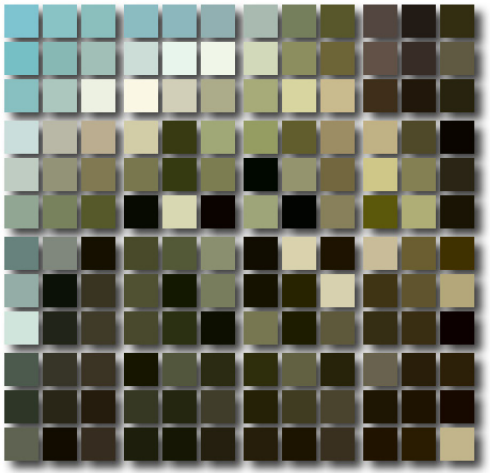
3/7 resize



Copy them into a new image.

Nearest Neighbor Resampling

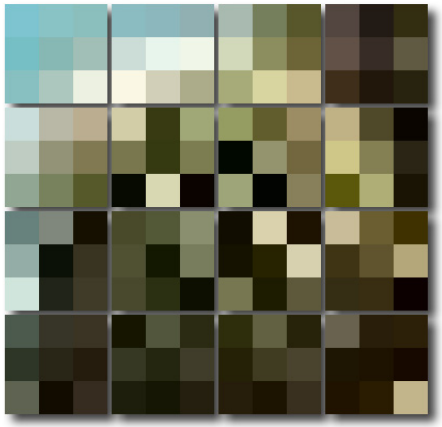
3/7 resize



Copy them into a new image.

Nearest Neighbor Resampling

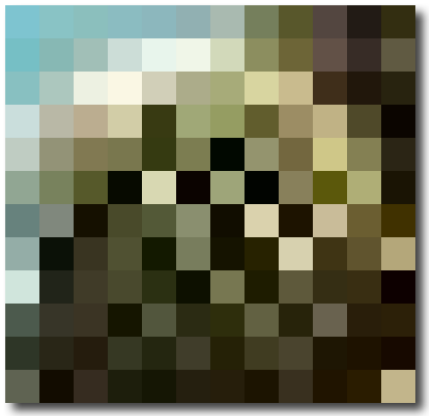
3/7 resize



Copy them into a new image.

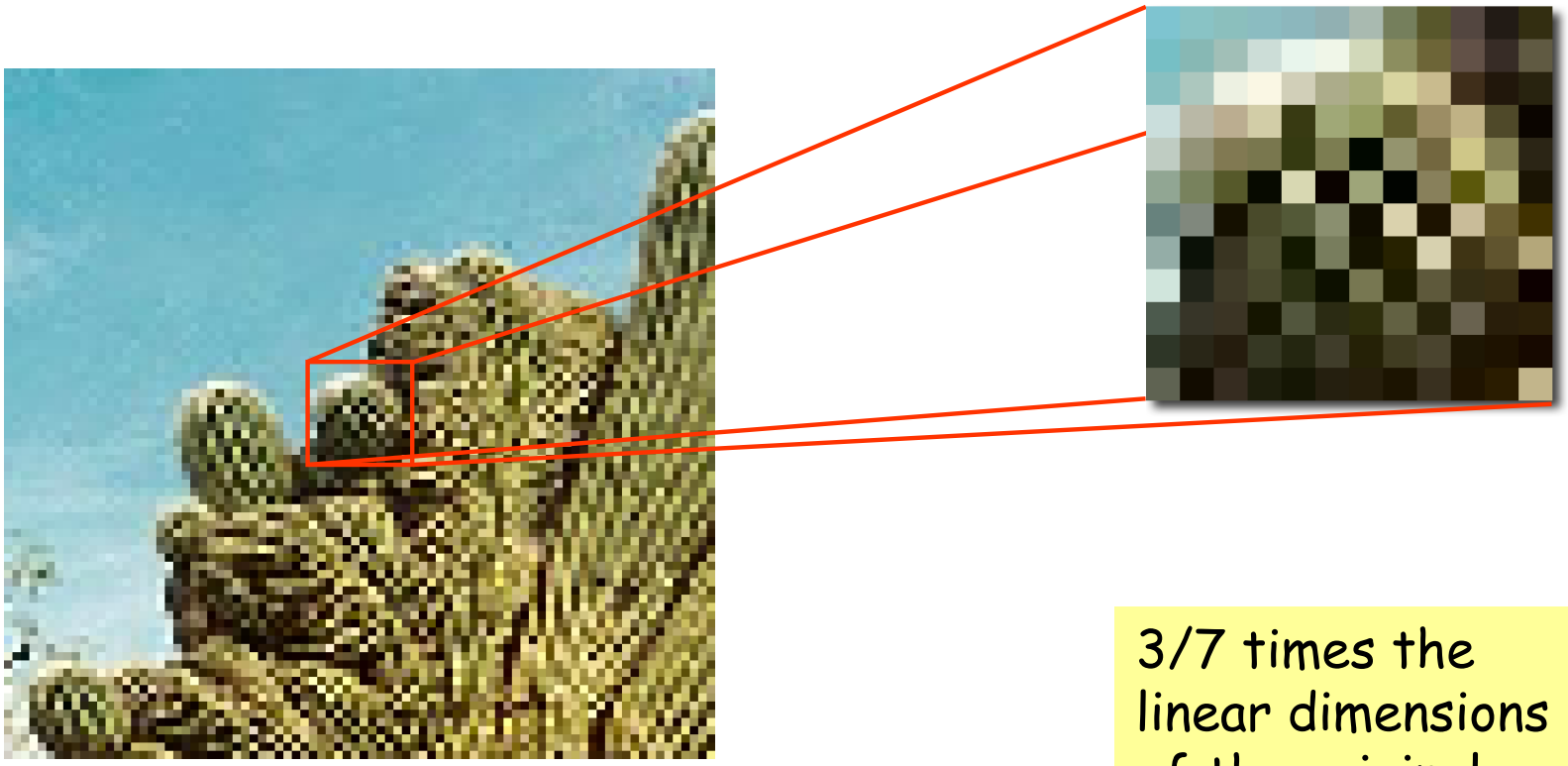
Nearest Neighbor Resampling

3/7 resize



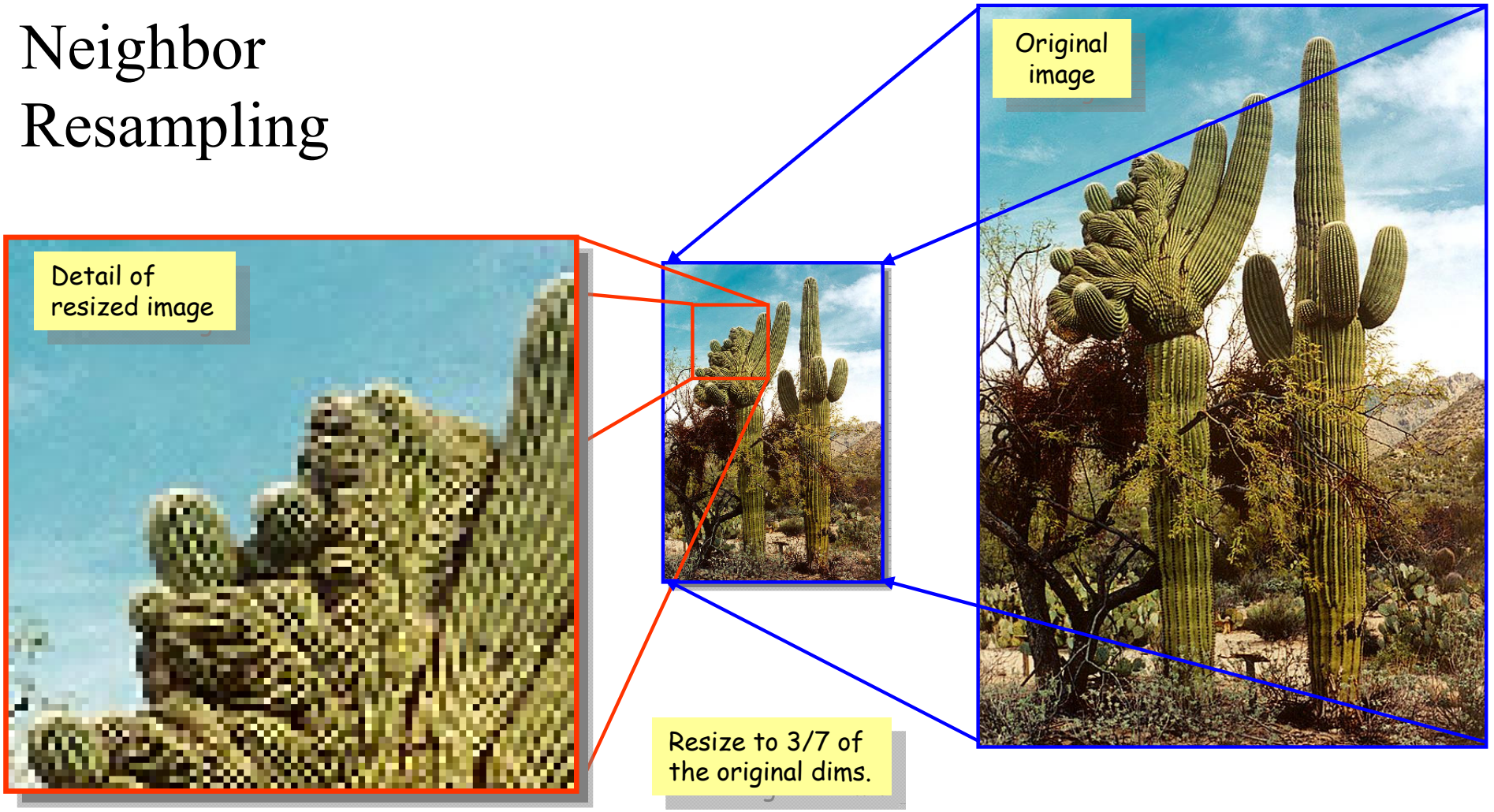
3/7 times the linear dimensions of the original

Nearest Neighbor Resampling

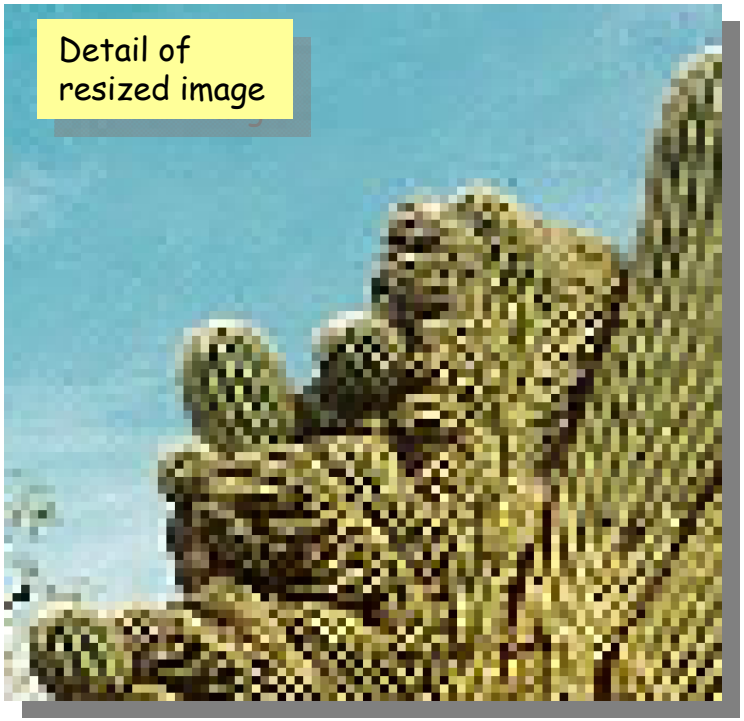


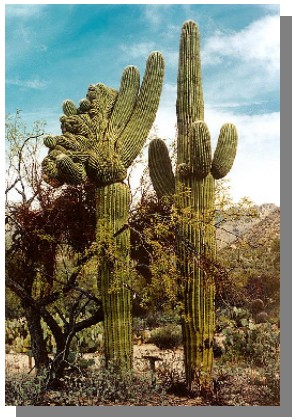
3/7 times the linear dimensions of the original

Nearest Neighbor Resampling



Nearest Neighbor Resampling

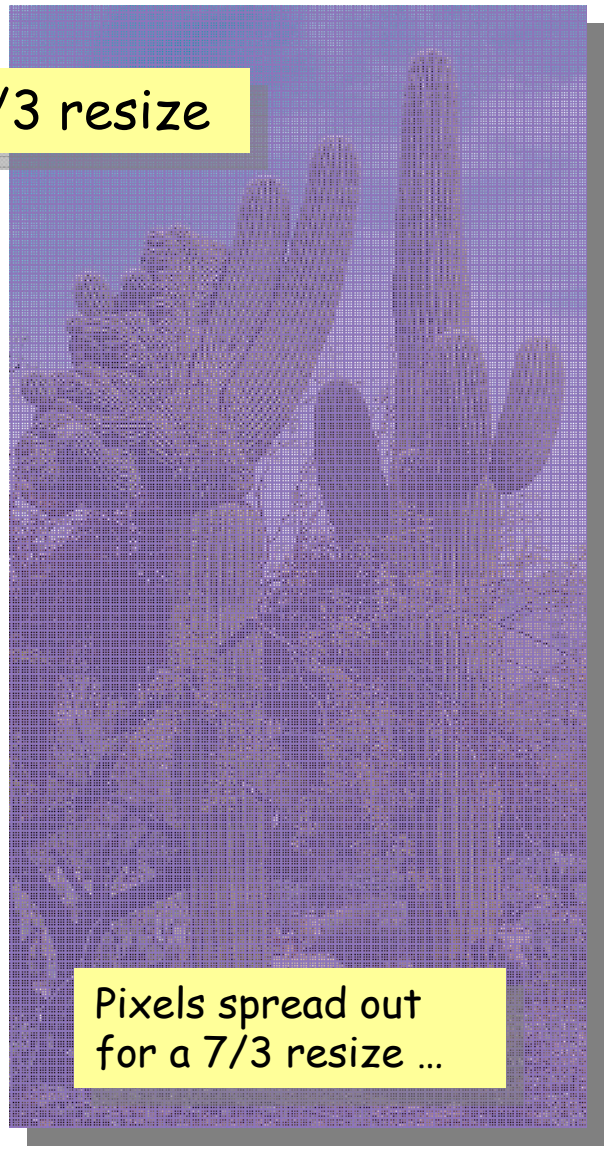




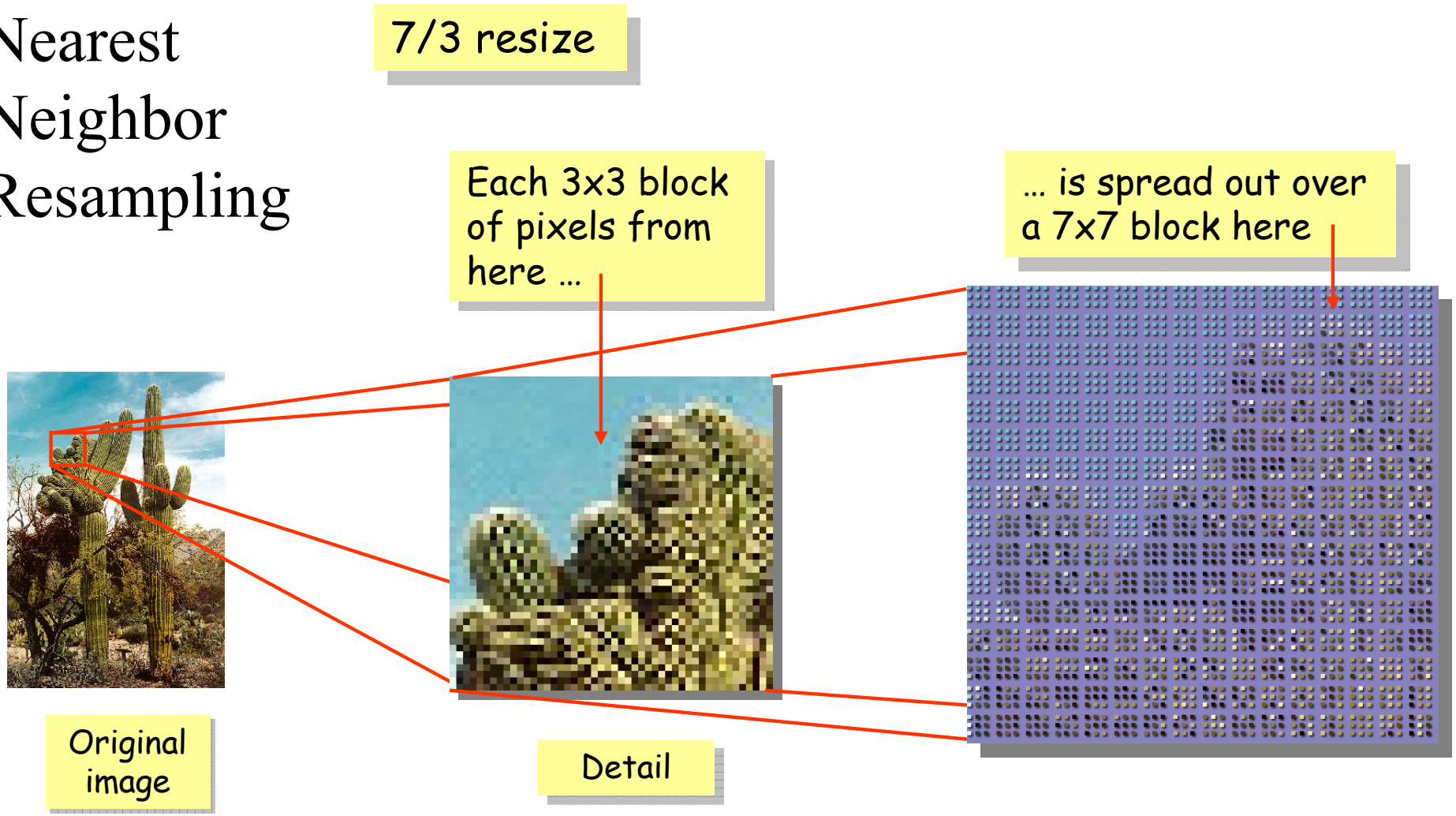
Resize to 3/7 of the original dims.

Nearest Neighbor Resampling

Original image

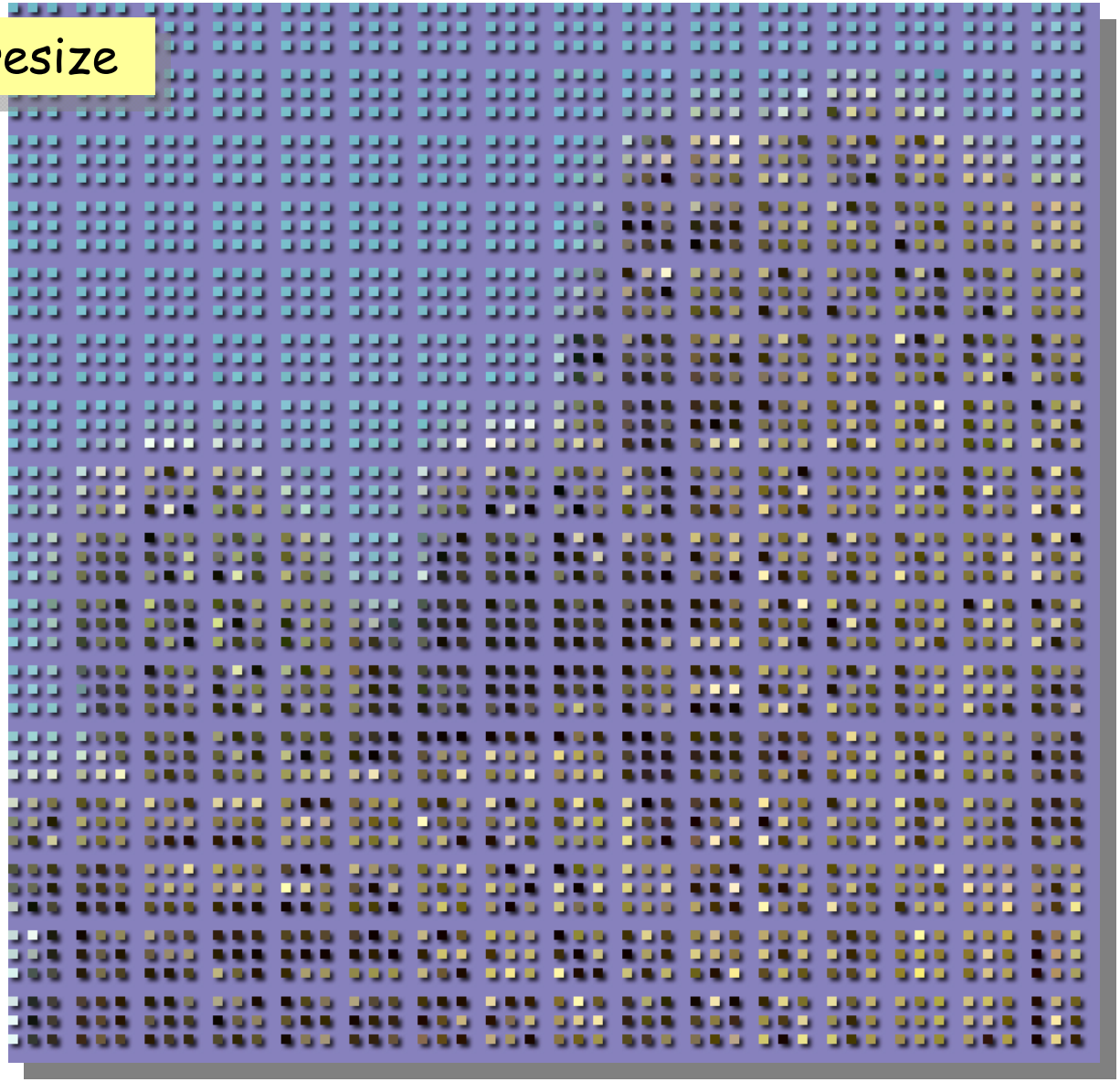


Nearest Neighbor Resampling



Nearest Neighbor Resampling

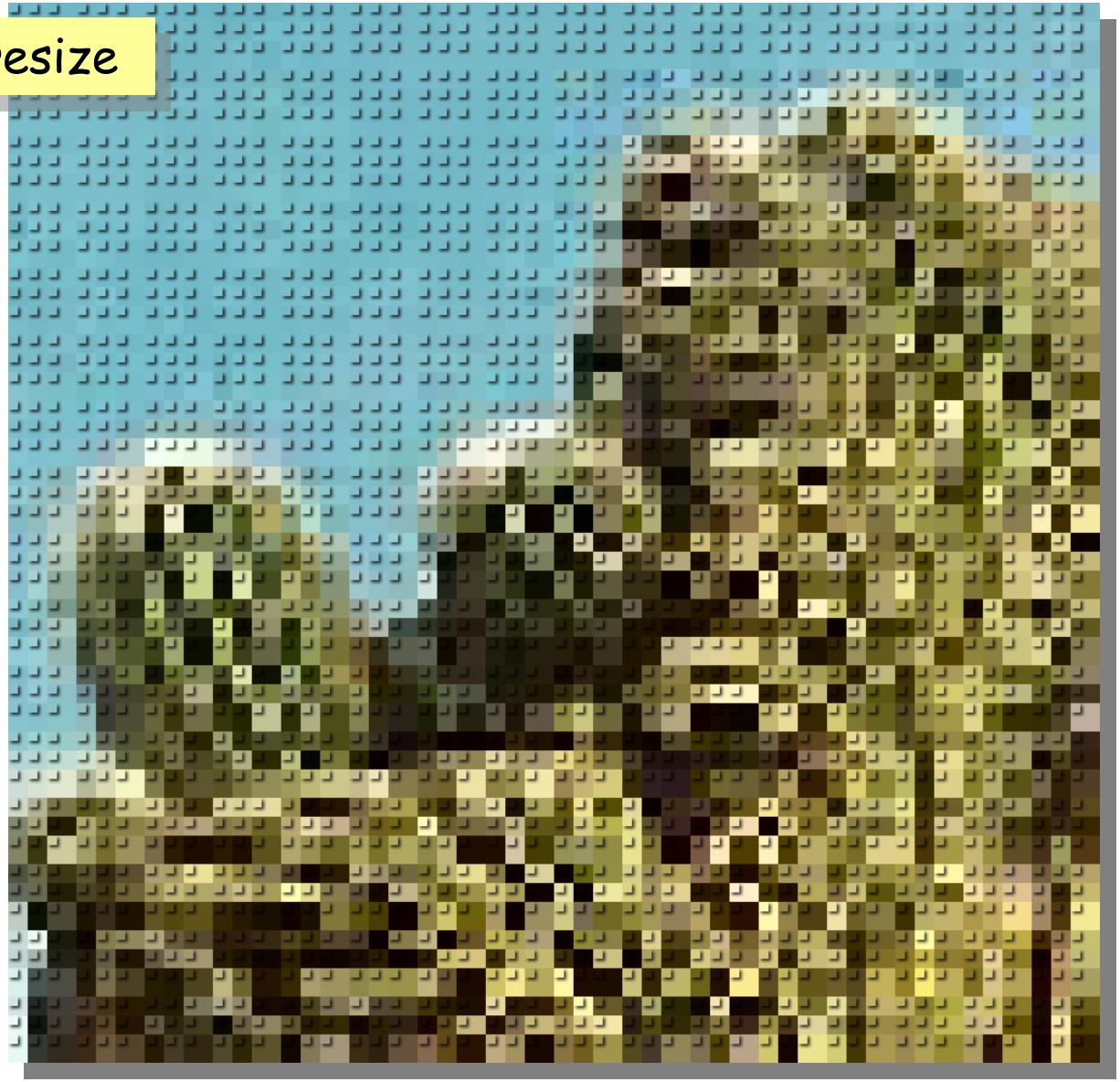
7/3 resize



3x3 blocks distributed over 7x7 blocks

Nearest Neighbor Resampling

7/3 resize



Empty pixels filled with color from ULH non-empty pixel

Nearest Neighbor Resampling

Empty pixels filled with color from ULH non-empty pixel

7/3 resize

Nearest Neighbor Resampling

Original image

Nearest Neighbor Resampling

Size of original image, I: $R' \times C'$

Size of scaled image, J: $R \times C$

Row scale factor (input to output):

$$S_r = \begin{cases} R'/R, & \text{if} \quad R' > R, \\ (R'-1)/R, & \text{if} \quad R' < R, \end{cases}$$

Column scale factor (input to output):

$$S_c = \begin{cases} C'/C, & \text{if} \quad C' > C, \\ (C'-1)/C, & \text{if} \quad C' < C \end{cases}$$

For each (r, c) in J, the corresponding fractional pixel location, (r_f, c_f), in I is:

$$(r_f, c_f) = (S_R \cdot r,\ S_C \cdot c)$$

if $S_r \geq 0.5$, and $S_c \geq 0.5$, for

$$r = 1, \ldots, R,$$

$$c = 1, \ldots, C,$$

If $S_r < 0.5$ or $S_c < 0.5$ then start at

$$r = \left\lfloor \tfrac{1}{S_r} \right\rfloor, \ldots, R,$$

$$c = \left\lfloor \tfrac{1}{S_c} \right\rfloor, \ldots, C,$$

accordingly. The closest integer pixel location (r', c'), in I is

$$(r', c') = \text{round}\,(r_f, c_f).$$

Then

$$J(r, c) = I(r', c').$$

Nearest Neighbor Resampling

Size
Size
Rov

Col

For each (r, c) in J, the corresponding fractional pixel location, (r_f, c_f), in I is:

The idea here is that the 4x4 neighborhood in I of the point $(r_f,c_f,)$ has $(r',c') = \lfloor(r_f,c_f,)\rfloor$ as its upper left corner and has $(r'+1,c'+1)$ as its lower right corner.

Thus for each $(r_f,c_f,)$: (1) neither r' nor c' can be less than one and (2) $r'+1$ cannot be greater than R' and $c'+1$ cannot be greater than C'.

If the set of all indices $\{(r',c')\}$ do not satisfy (1) or (2), you must adjust the indices so that they do.

$$(r_f, c_f) = (S_R \cdot r,\ S_C \cdot c)$$

if $S_r \geq 0.5$, and $S_c \geq 0.5$.

Otherwise, if $S_r < 0.5$ or $S_c < 0.5$ then

$$r = \left\lfloor \tfrac{1}{S_r} \right\rfloor, \ldots, R,$$
$$c = \left\lfloor \tfrac{1}{S_c} \right\rfloor, \ldots, C,$$

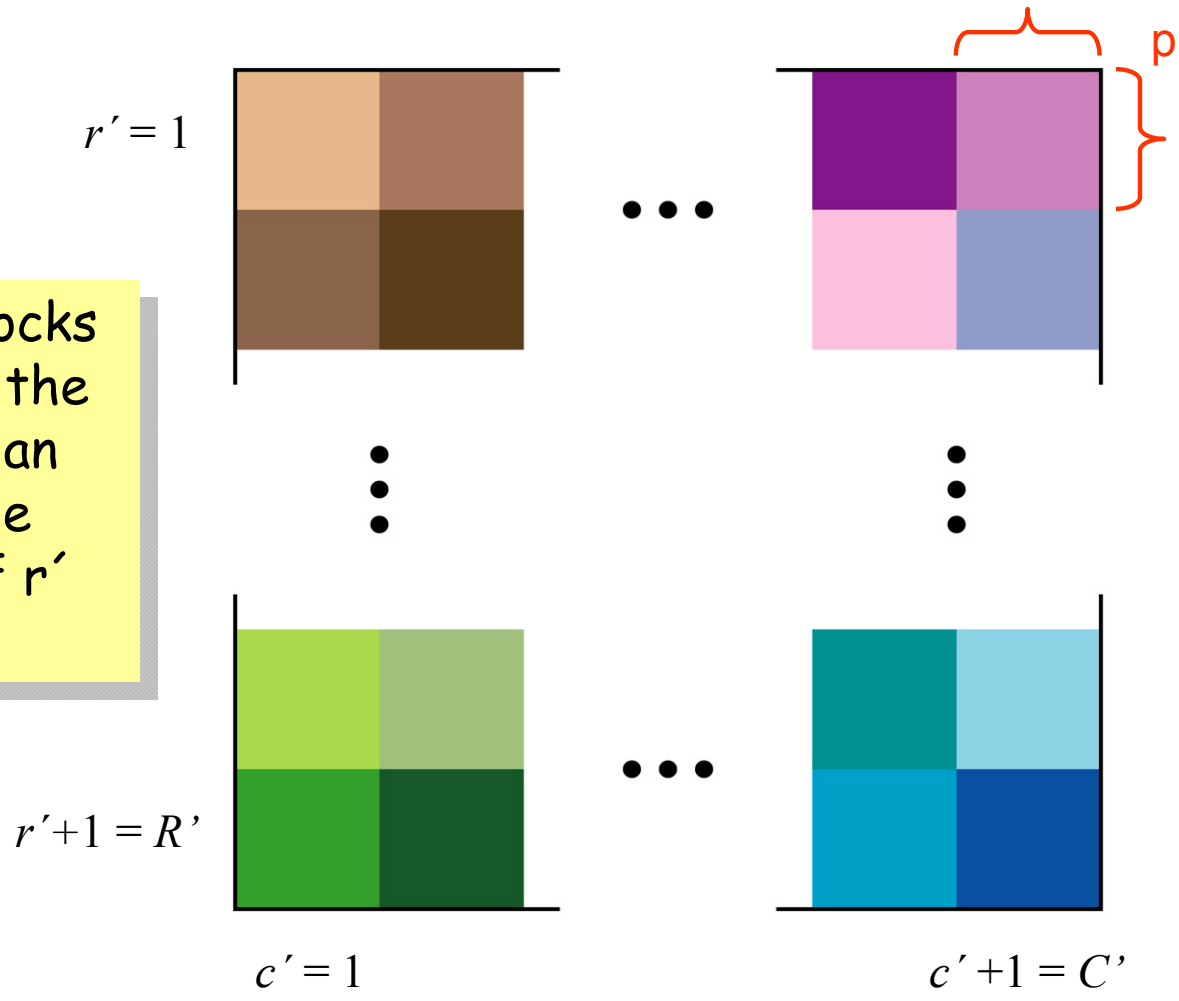
accordingly. The closest integer pixel location (r', c'), in I is

$$(r', c') = \text{round}\,(r_f, c_f).$$

Then

$$J(r, c) = I(r', c').$$

Nearest Neighbor Resampling

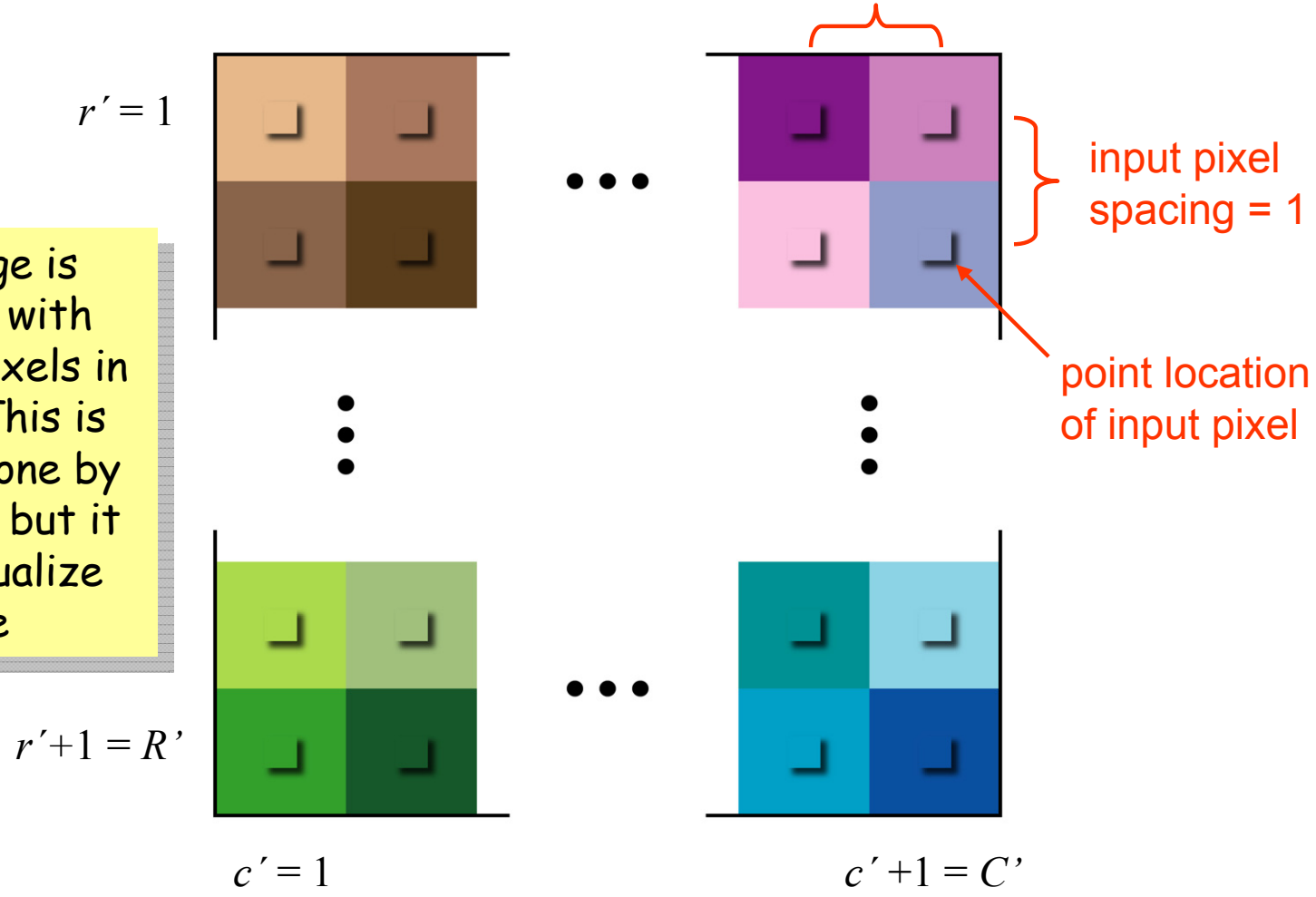


These 4 blocks represent the corners of an image — the extrema of r′ and c′.

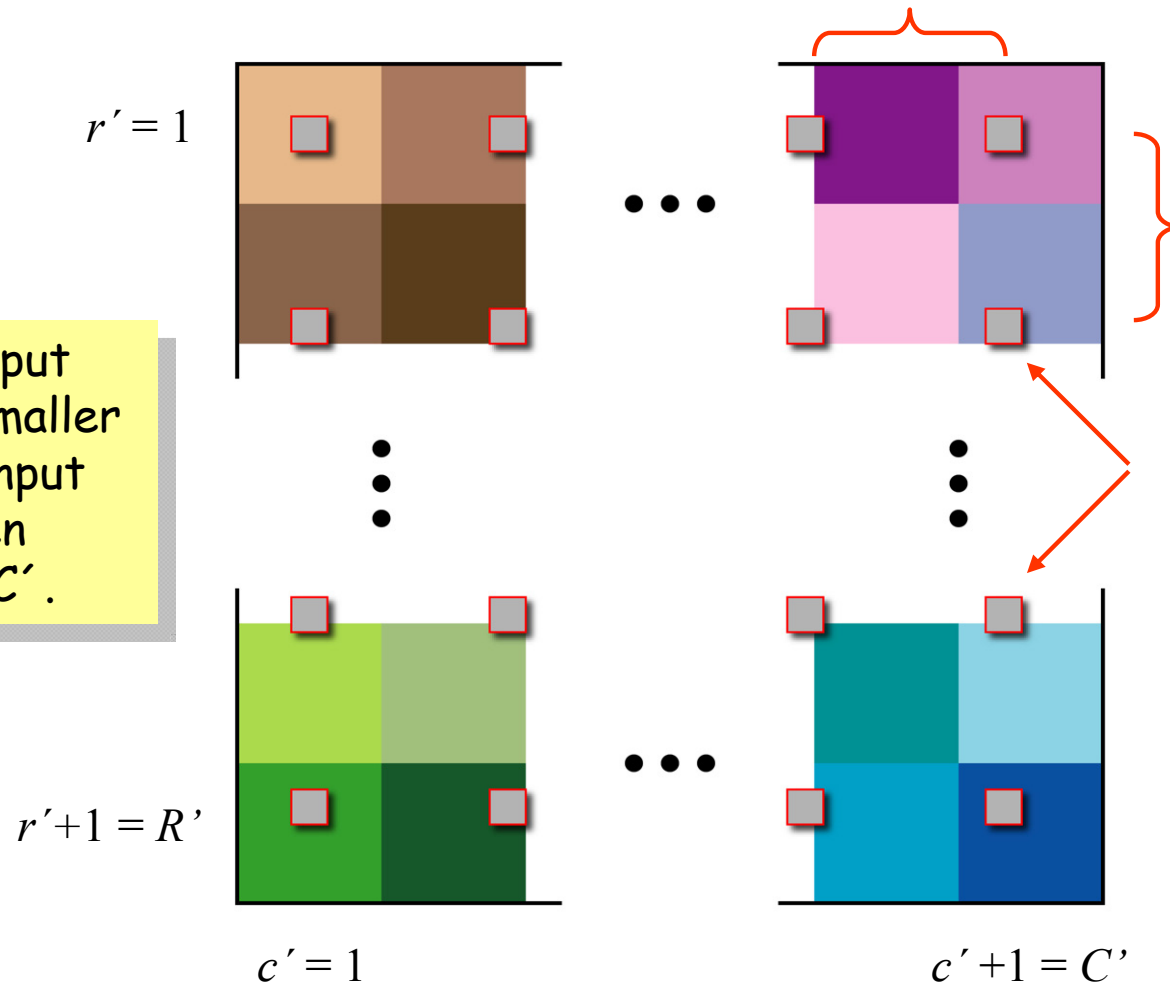
They are also the extrema of 4-pixel neighborhoods in the image.

Nearest Neighbor Resampling

Here the image is supersampled with the original pixels in the center. This is not actually none by the algorithm but it helps one visualize the procedure



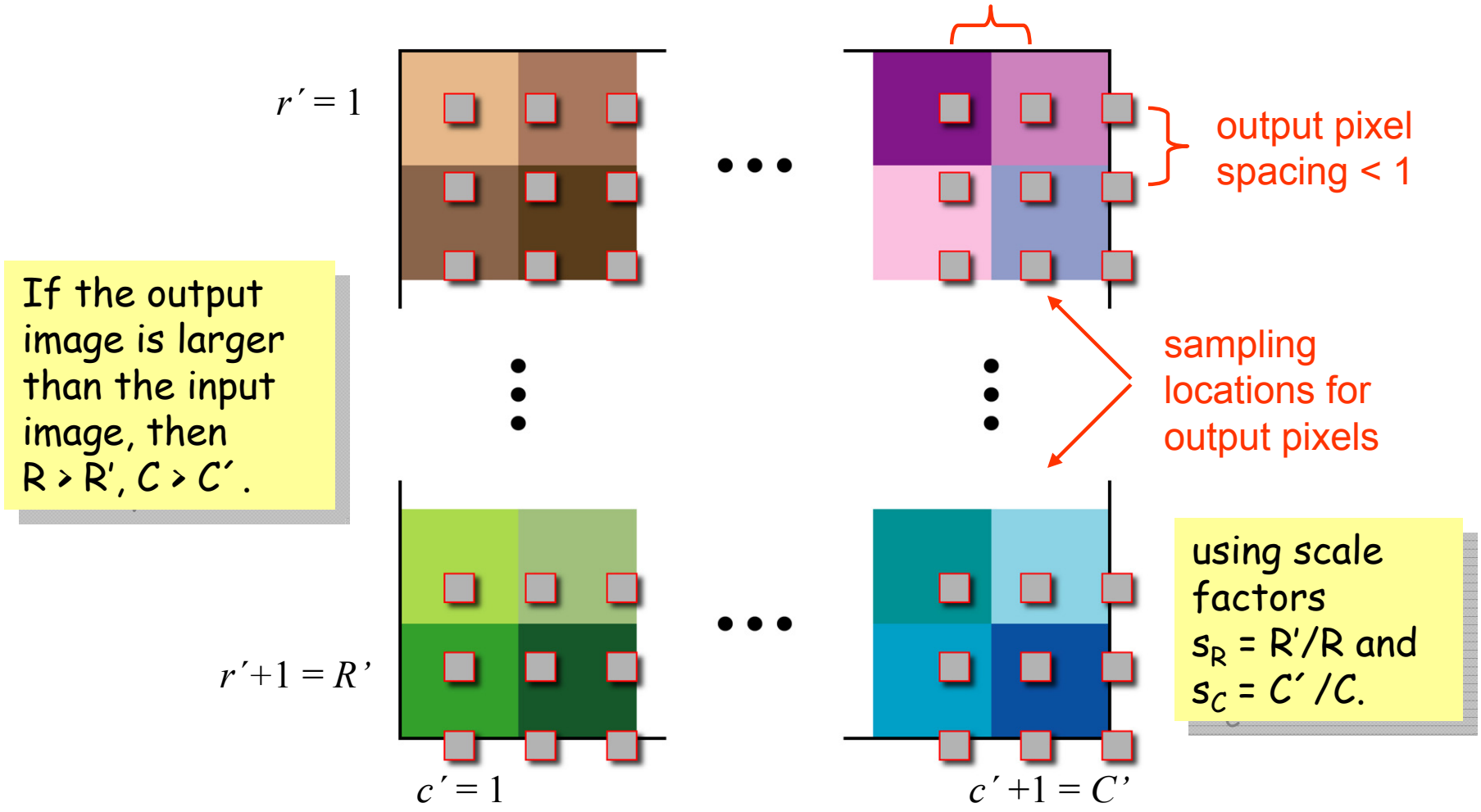
Nearest Neighbor Resampling



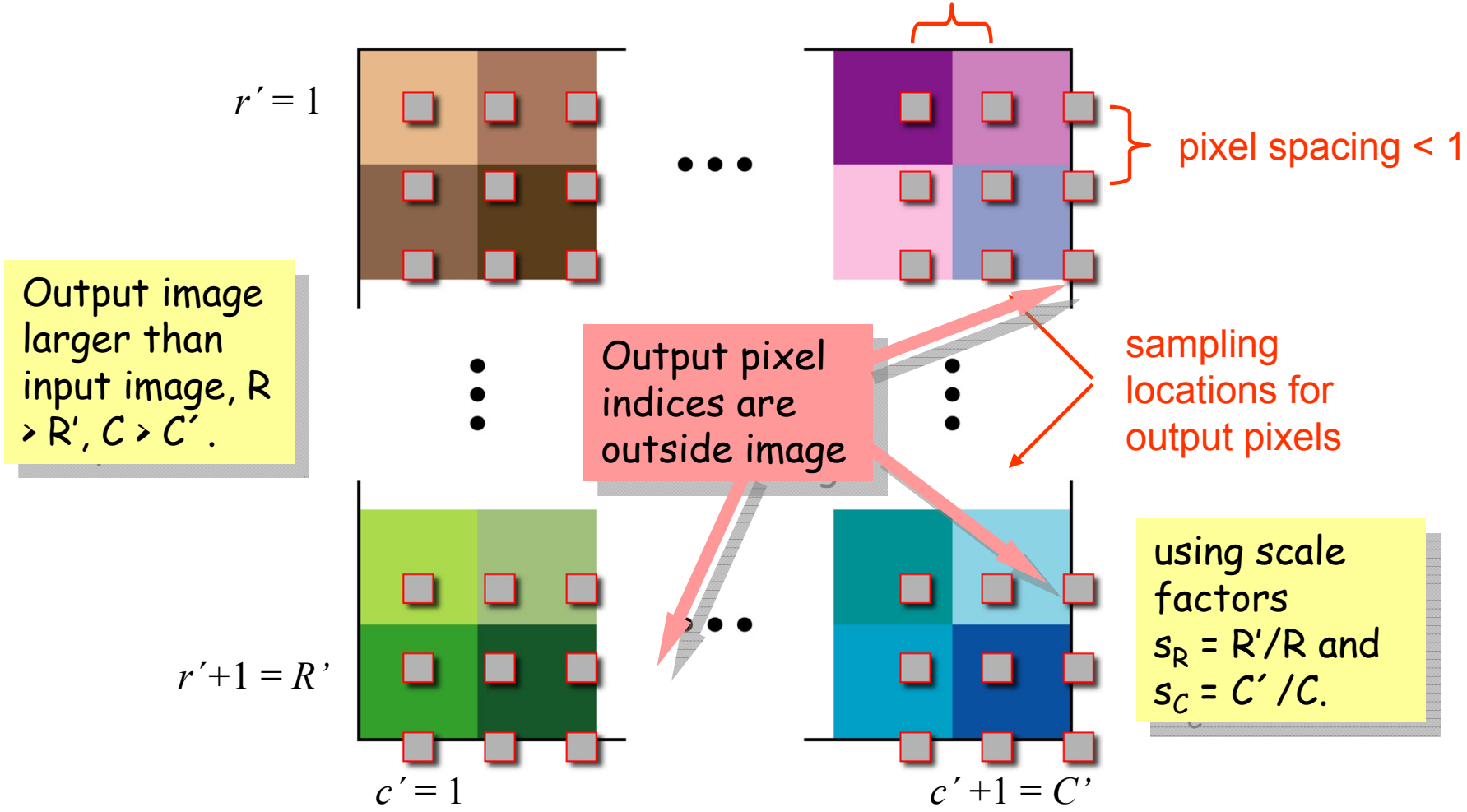
If the output image is smaller than the input image, then $R < R'$, $C < C'$.

using scale factors $s_R = R'/R$ and $s_C = C'/C$.

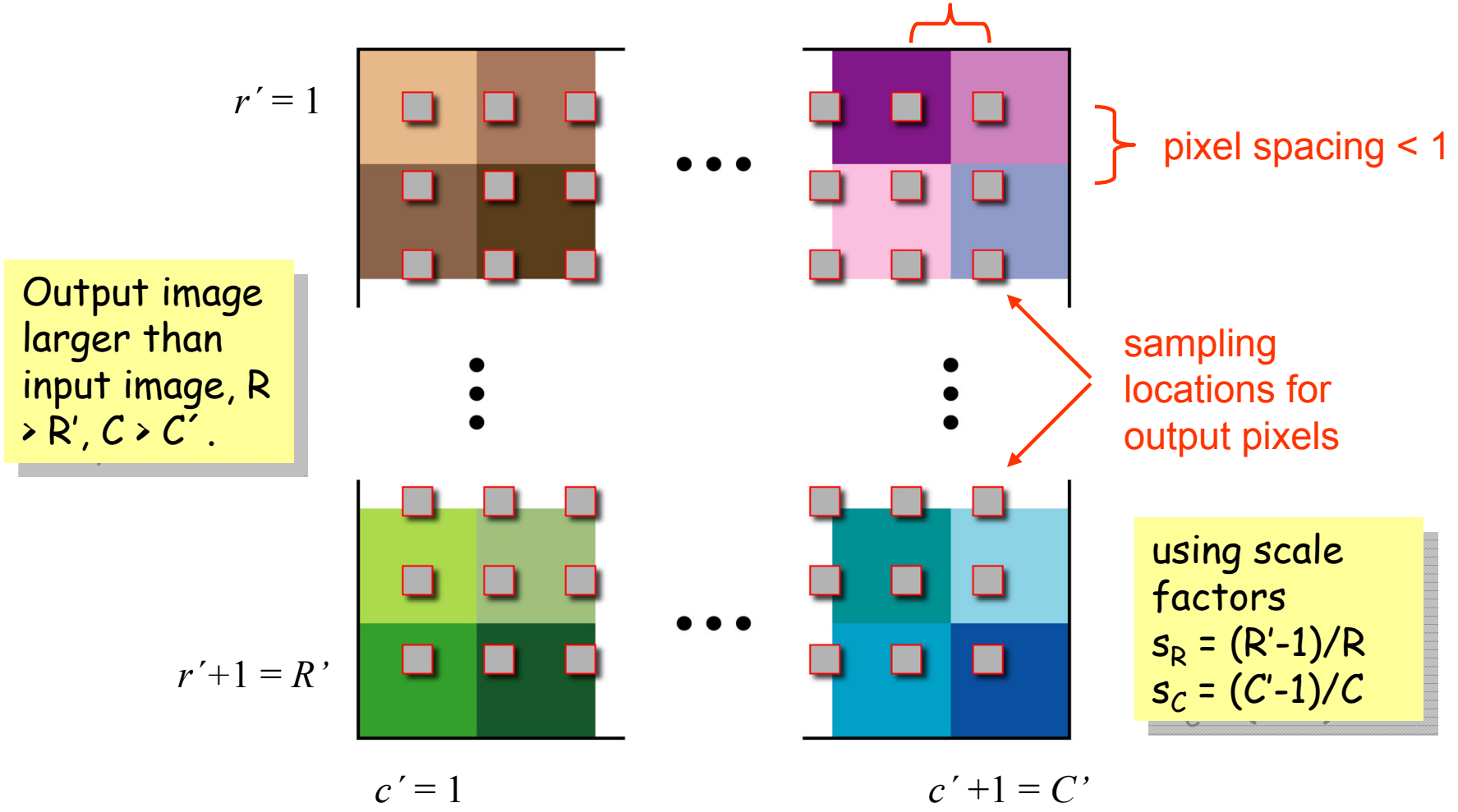
Nearest Neighbor Resampling



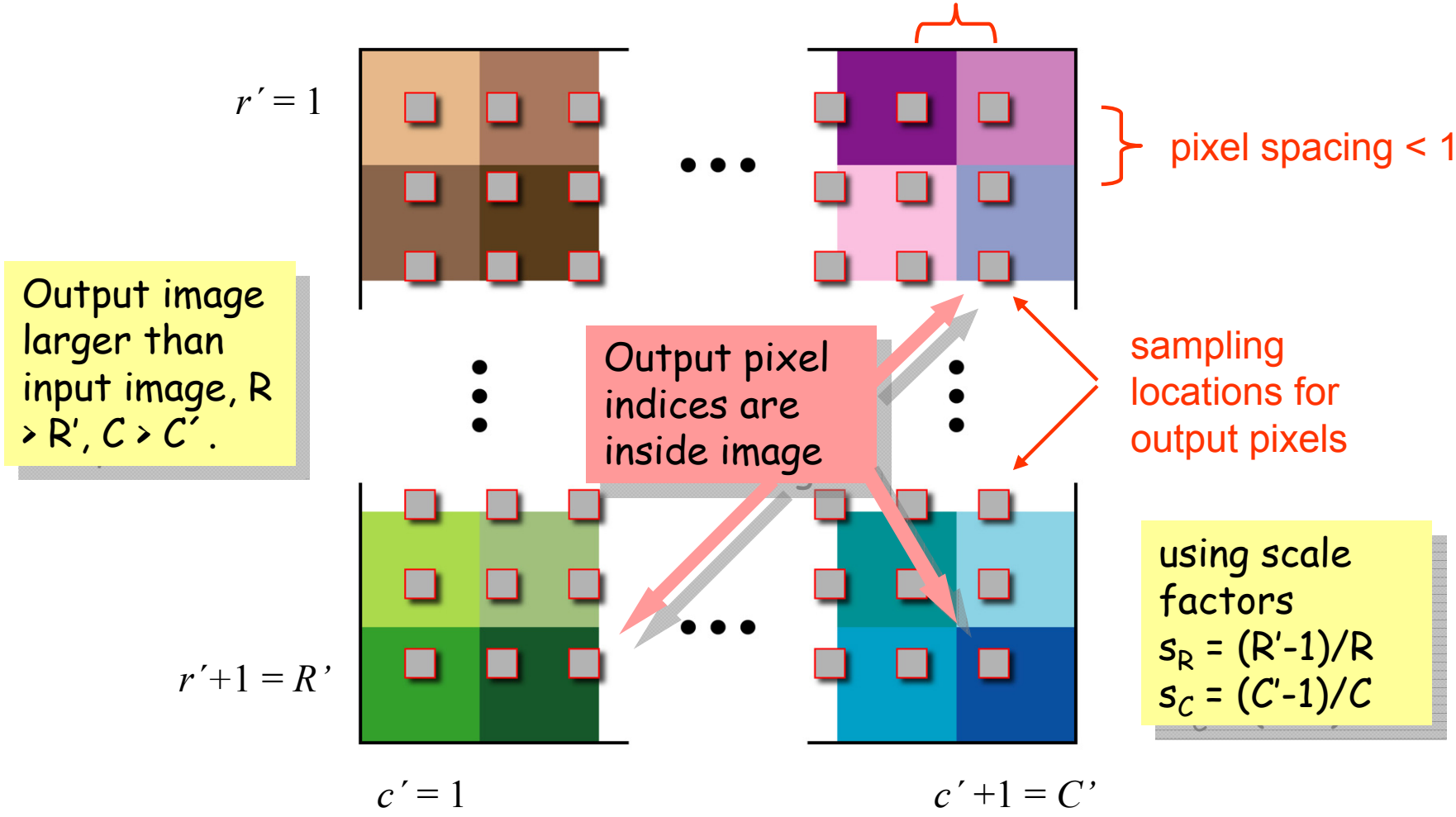
Nearest Neighbor Resampling



Nearest Neighbor Resampling



Nearest Neighbor Resampling



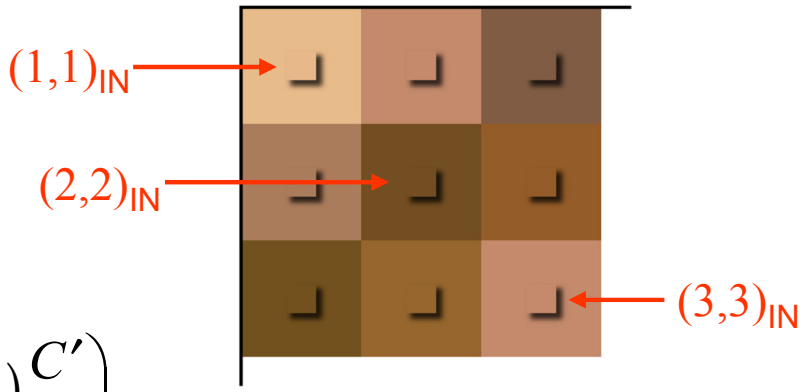
Nearest Neighbor Resampling

If the output is smaller than the input, $R < R'$ and $C < C'$.

$$(1,1)_{OUT} \leftrightarrow (1,1)_{IN}$$

$$(2,2)_{OUT} \leftrightarrow (1,1)_{IN} + \left(\frac{R'}{R}, \frac{C'}{C}\right)$$

$$(n,n)_{OUT} \leftrightarrow (1,1)_{IN} + \left((n-1)\frac{R'}{R}, (n-1)\frac{C'}{C}\right)$$



lies between $(\rho_n + 1, \chi_n + 1)$ and $(\rho_n + 2, \chi_n + 2)$

$$\rho_n = \left\lfloor (n-1)\frac{R'}{R} \right\rfloor \text{ and } \chi_n = \left\lfloor (n-1)\frac{C'}{C} \right\rfloor$$

This and the next 7 slides explain the scale factor selection algebraically

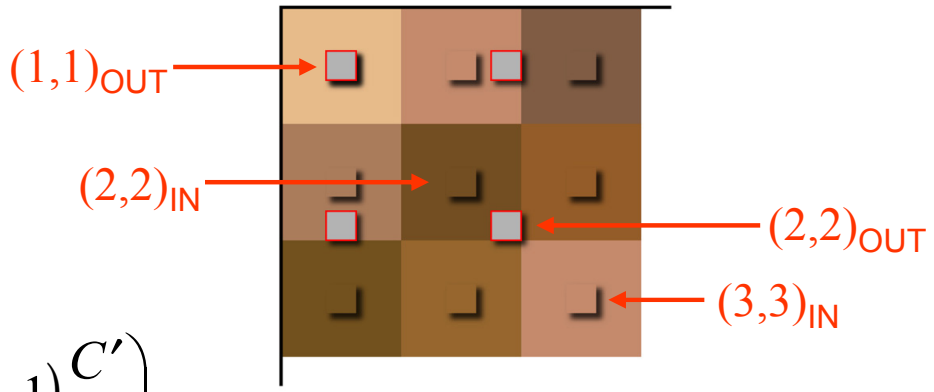
Nearest Neighbor Resampling

If the output is smaller than the input, $R < R'$ and $C < C'$.

$$(1,1)_{OUT} \leftrightarrow (1,1)_{IN}$$

$$(2,2)_{OUT} \leftrightarrow (1,1)_{IN} + \left(\frac{R'}{R}, \frac{C'}{C}\right)$$

$$(n,n)_{OUT} \leftrightarrow (1,1)_{IN} + \left((n-1)\frac{R'}{R}, (n-1)\frac{C'}{C}\right)$$



lies between $(\rho_n + 1, \chi_n + 1)$ and $(\rho_n + 2, \chi_n + 2)$

$$\rho_n = \left\lfloor (n-1)\frac{R'}{R} \right\rfloor \text{ and } \chi_n = \left\lfloor (n-1)\frac{C'}{C} \right\rfloor$$

$(2,2)_{OUT}$ lies between $(2,2)_{IN}$ and $(3,3)_{IN}$ since $R'/R > 1$ & $C'/C > 1$.

Nearest Neighbor Resampling

If the output is smaller than the input, $R < R'$ and $C < C'$.

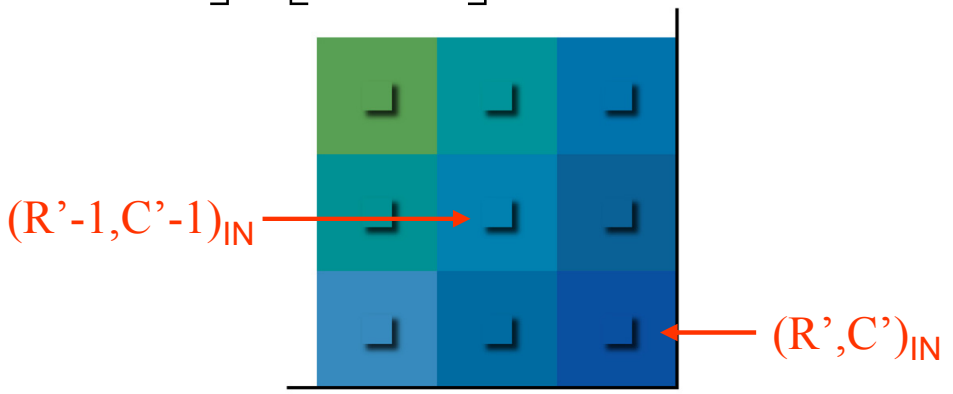
$$(R,C)_{\text{OUT}} \leftrightarrow (1,1)_{\text{IN}} + \left((R-1)\frac{R'}{R},(C-1)\frac{C'}{C}\right)$$

but $R = \alpha R'$ and $C = \beta C'$ where $\alpha < 1$ and $\beta < 1$.

$$\rho_R = \left\lfloor (R-1)\frac{R'}{R} \right\rfloor = \left\lfloor (\alpha R'-1)\frac{R'}{\alpha R'} \right\rfloor = \left\lfloor \frac{1}{\alpha}(\alpha R'-1) \right\rfloor = \left\lfloor R' - \frac{1}{\alpha} \right\rfloor = R'-2.$$

Similarly, $\chi_C = C'-2$.

Thus $(R,C)_{\text{OUT}}$ lies between $(R'-1, C'-1)_{\text{IN}}$ and $(R', C')_{\text{IN}}$.



Nearest Neighbor Resampling

If the output is smaller than the input, $R < R'$ and $C < C'$.

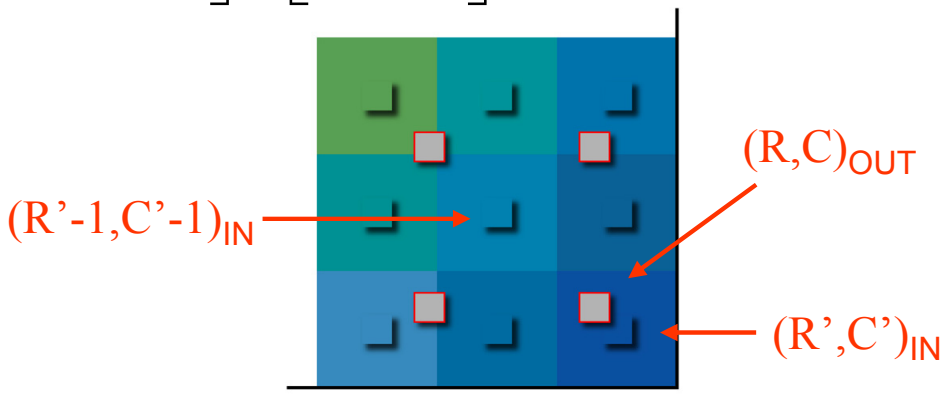
$$(R,C)_{\text{OUT}} \leftrightarrow (1,1)_{\text{IN}} + \left((R-1)\frac{R'}{R},(C-1)\frac{C'}{C}\right)$$

but $R = \alpha R'$ and $C = \beta C'$ where $\alpha < 1$ and $\beta < 1$.

$$\rho_R = \left\lfloor (R-1)\frac{R'}{R} \right\rfloor = \left\lfloor (\alpha R'-1)\frac{R'}{\alpha R'} \right\rfloor = \left\lfloor \frac{1}{\alpha}(\alpha R'-1) \right\rfloor = \left\lfloor R' - \frac{1}{\alpha} \right\rfloor = R'-2.$$

Similarly, $\chi_C = C'-2.$

Thus $(R,C)_{\text{OUT}}$ lies between $(R'-1, C'-1)_{\text{IN}}$ and $(R', C')_{\text{IN}}$.



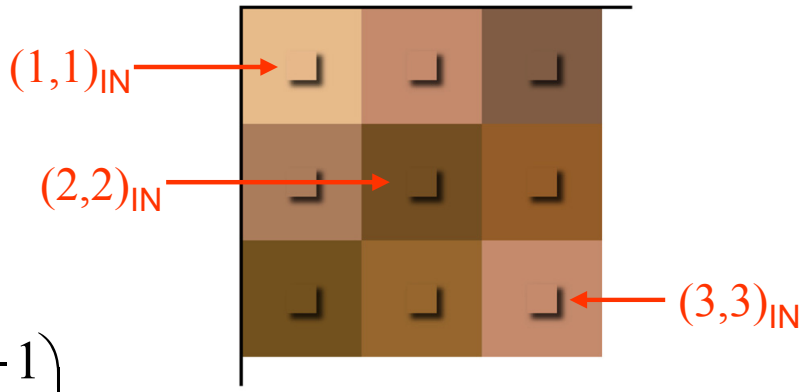
Nearest Neighbor Resampling

If the output is larger than the input, $R' < R$ and $C' < C$.

$$(1,1)_{OUT} \leftrightarrow (1,1)_{IN}$$

$$(2,2)_{OUT} \leftrightarrow (1,1)_{IN} + \left(\frac{R'-1}{R}, \frac{C'-1}{C}\right)$$

$$(n,n)_{OUT} \leftrightarrow (1,1)_{IN} + \left((n-1)\frac{R'-1}{R}, (n-1)\frac{C'-1}{C}\right)$$



lies between (ρ_n, χ_n) and $(\rho_n + 1, \chi_n + 1)$

$$\rho_n = \left\lfloor (n-1)\frac{R'-1}{R} \right\rfloor \text{ and } \chi_n = \left\lfloor (n-1)\frac{C'-1}{C} \right\rfloor$$

Nearest Neighbor Resampling

If the output is larger than the input, $R' < R$ and $C' < C$.

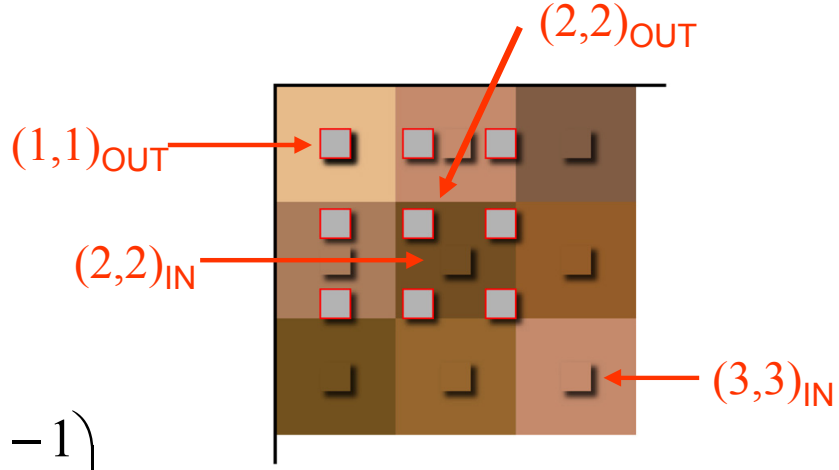
$$(1,1)_{OUT} \leftrightarrow (1,1)_{IN}$$

$$(2,2)_{OUT} \leftrightarrow (1,1)_{IN} + \left(\frac{R'-1}{R}, \frac{C'-1}{C}\right)$$

$$(n,n)_{OUT} \leftrightarrow (1,1)_{IN} + \left((n-1)\frac{R'-1}{R}, (n-1)\frac{C'-1}{C}\right)$$

lies between (ρ_n, χ_n) and $(\rho_n + 1, \chi_n + 1)$

$$\rho_n = \left\lfloor (n-1)\frac{R'-1}{R} \right\rfloor \text{ and } \chi_n = \left\lfloor (n-1)\frac{C'-1}{C} \right\rfloor$$



$(2,2)_{OUT}$ lies between $(1,1)_{IN}$ and $(2,2)_{IN}$ since $(R'-1)/R < 1$.

Nearest Neighbor Resampling

If the output is larger than the input, $R' < R$ and $C' < C$.

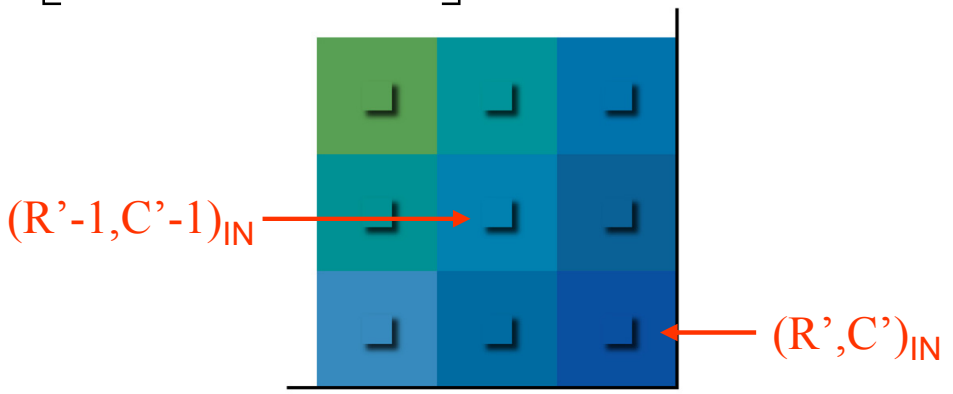
$$(R,C)_{\text{OUT}} \leftrightarrow (1,1)_{\text{IN}} + \left((R-1)\frac{R'-1}{R},(C-1)\frac{C'-1}{C}\right)$$

but $R = \alpha R'$ and $C = \beta C'$ where $\alpha > 1$ and $\beta > 1$.

$$\rho_R = \left\lfloor (R-1)\frac{R'-1}{R} \right\rfloor = \left\lfloor (\alpha R'-1)\frac{R'-1}{\alpha R'} \right\rfloor = \left\lfloor R'-1-\frac{1}{\alpha}+\frac{1}{\alpha R'} \right\rfloor = R'-2.$$

Similarly, $\chi_C = C'-2.$

Thus $(R,C)_{\text{OUT}}$ lies between $(R'-1, C'-1)_{\text{IN}}$ and $(R', C')_{\text{IN}}$.



Nearest Neighbor Resampling

If the output is larger than the input, $R' < R$ and $C' < C$.

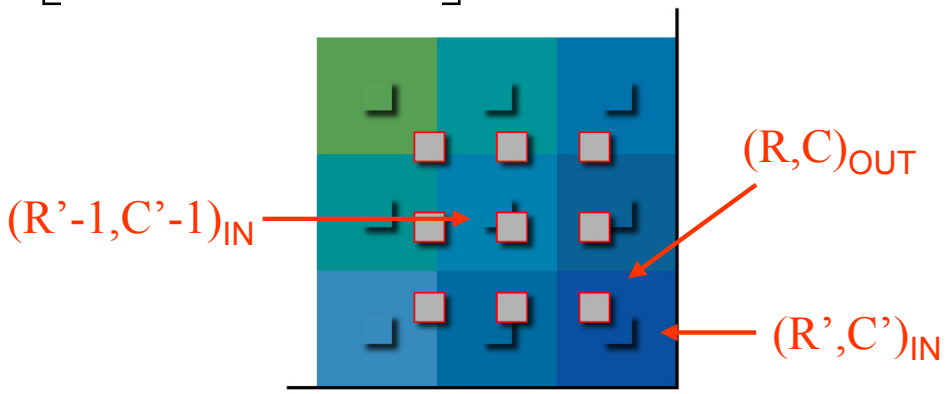
$$(R,C)_{\text{OUT}} \leftrightarrow (1,1)_{\text{IN}} + \left((R-1)\frac{R'-1}{R},(C-1)\frac{C'-1}{C}\right)$$

but $R = \alpha R'$ and $C = \beta C'$ where $\alpha > 1$ and $\beta > 1$.

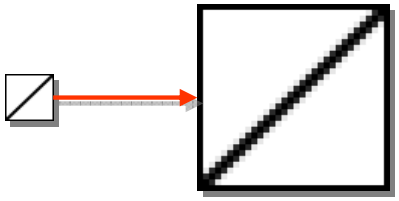
$$\rho_R = \left\lfloor (R-1)\frac{R'-1}{R} \right\rfloor = \left\lfloor (\alpha R'-1)\frac{R'-1}{\alpha R'} \right\rfloor = \left\lfloor R'-1-\frac{1}{\alpha}+\frac{1}{\alpha R'} \right\rfloor = R'-2.$$

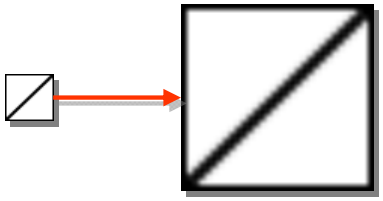
Similarly, $\chi_C = C'-2.$

Thus $(R,C)_{\text{OUT}}$ lies between $(R'-1, C'-1)_{\text{IN}}$ and $(R', C')_{\text{IN}}$.



Enlarging Images: Replication vs. Interpolation



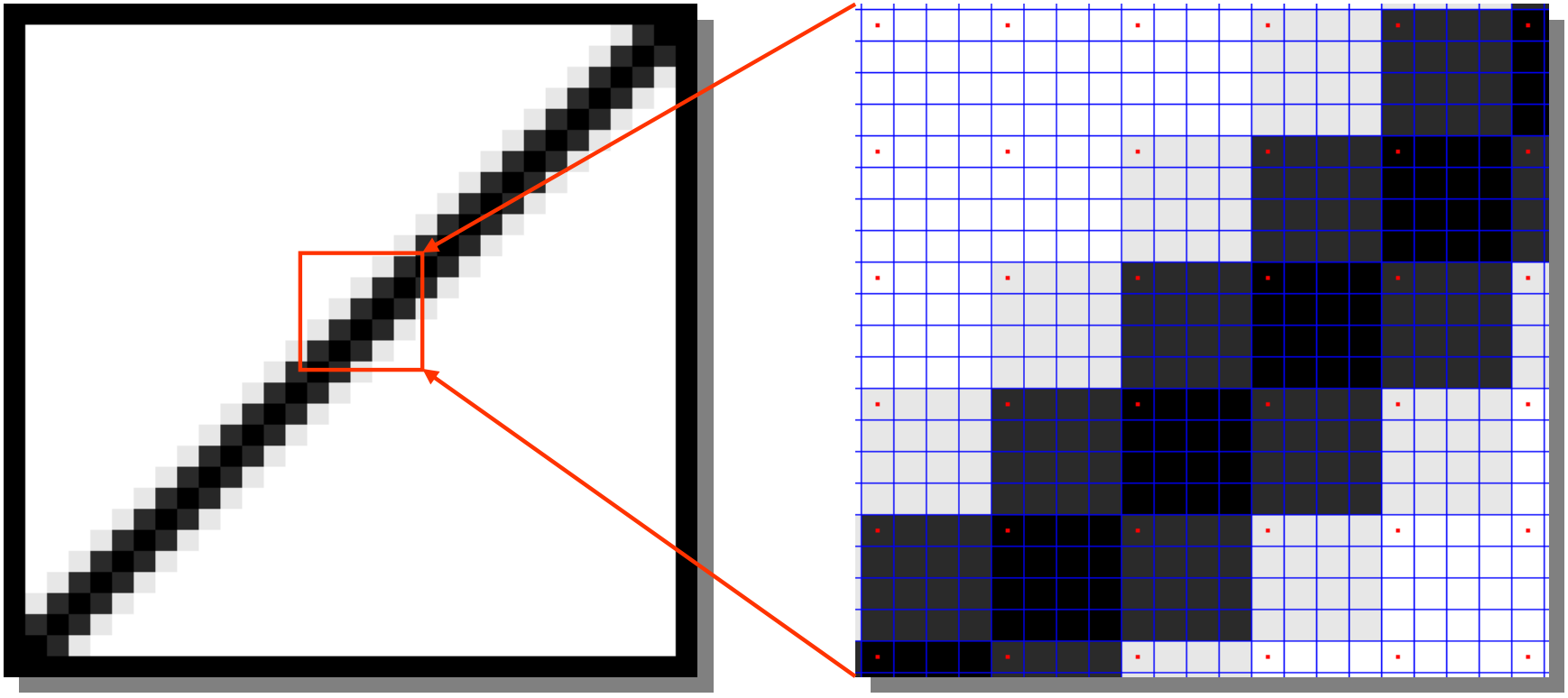


Pixel replication produces a "jagged" result since each individual square pixel is made larger.

Bilinear interpolation creates new pixels that have values intermediate between the originals. The result is smoother but blurry.

Pixel Replication

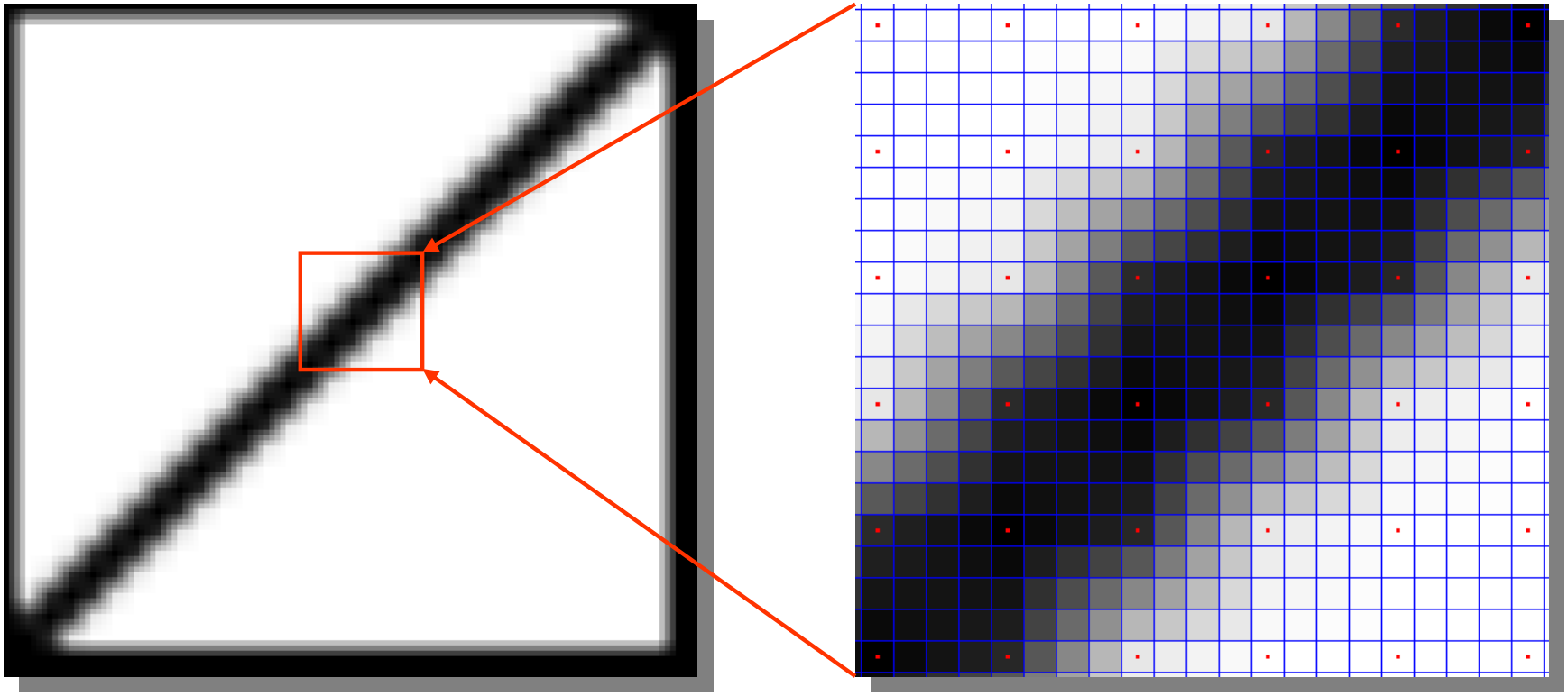
Red dots mark original pixel values.



Small square pixels become large square pixels.

Bilinear Interpolation

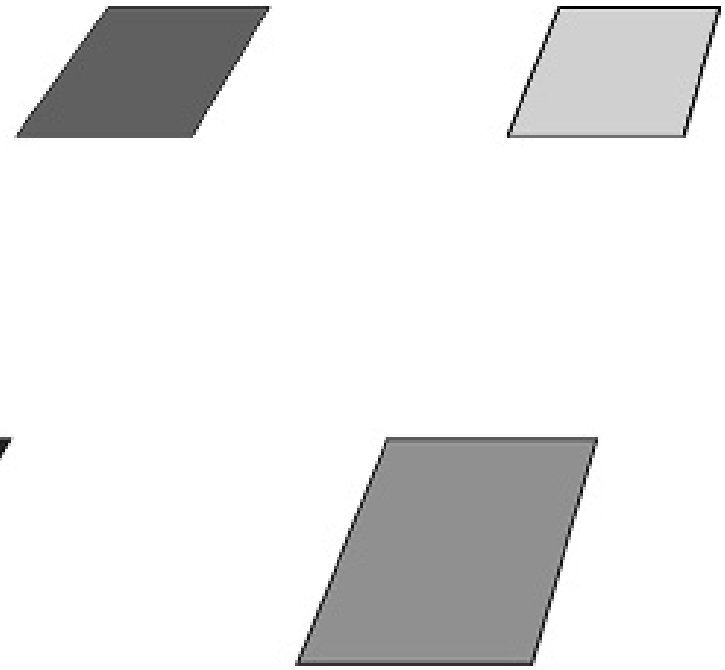
Red dots mark original pixel values.



Intermediate locations are filled with intermediate values.

Resampling Through Bilinear Interpolation

want to upsample this image by a factor of two

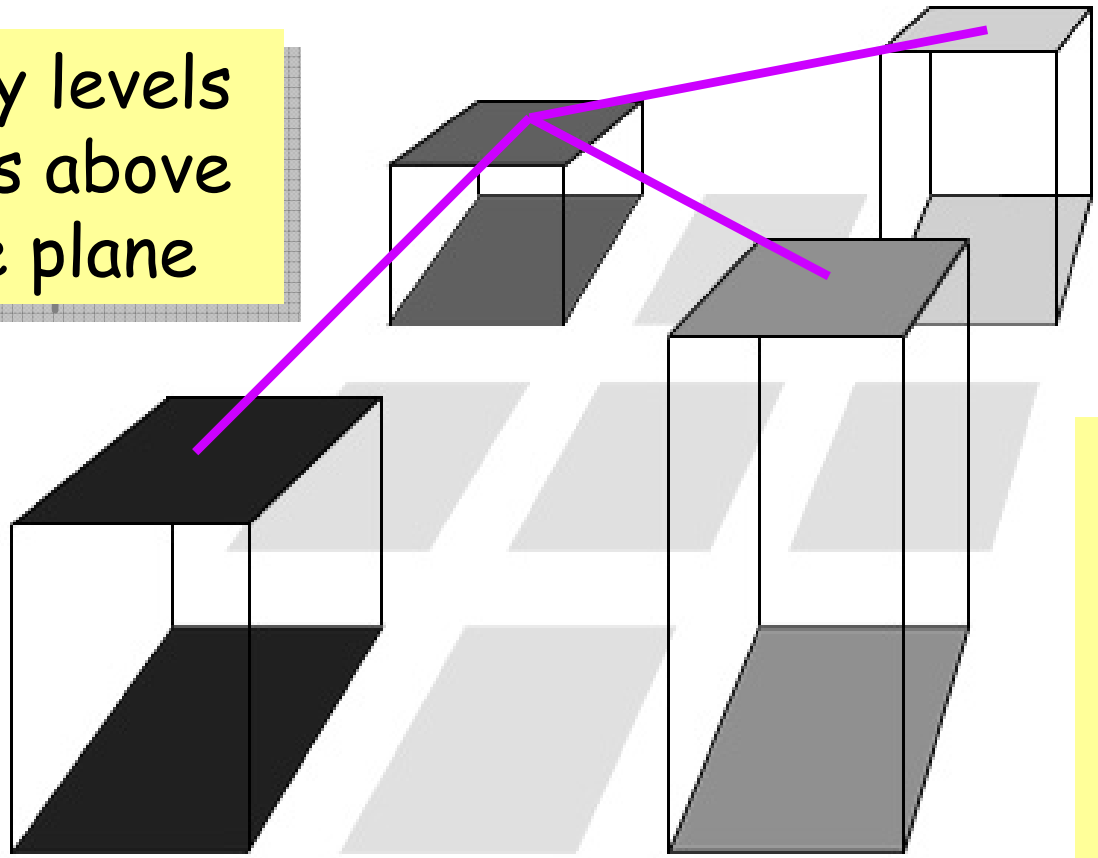


Resampling Through Bilinear Interpolation

new samples to be added at light gray locations.

Resampling Through Bilinear Interpolation

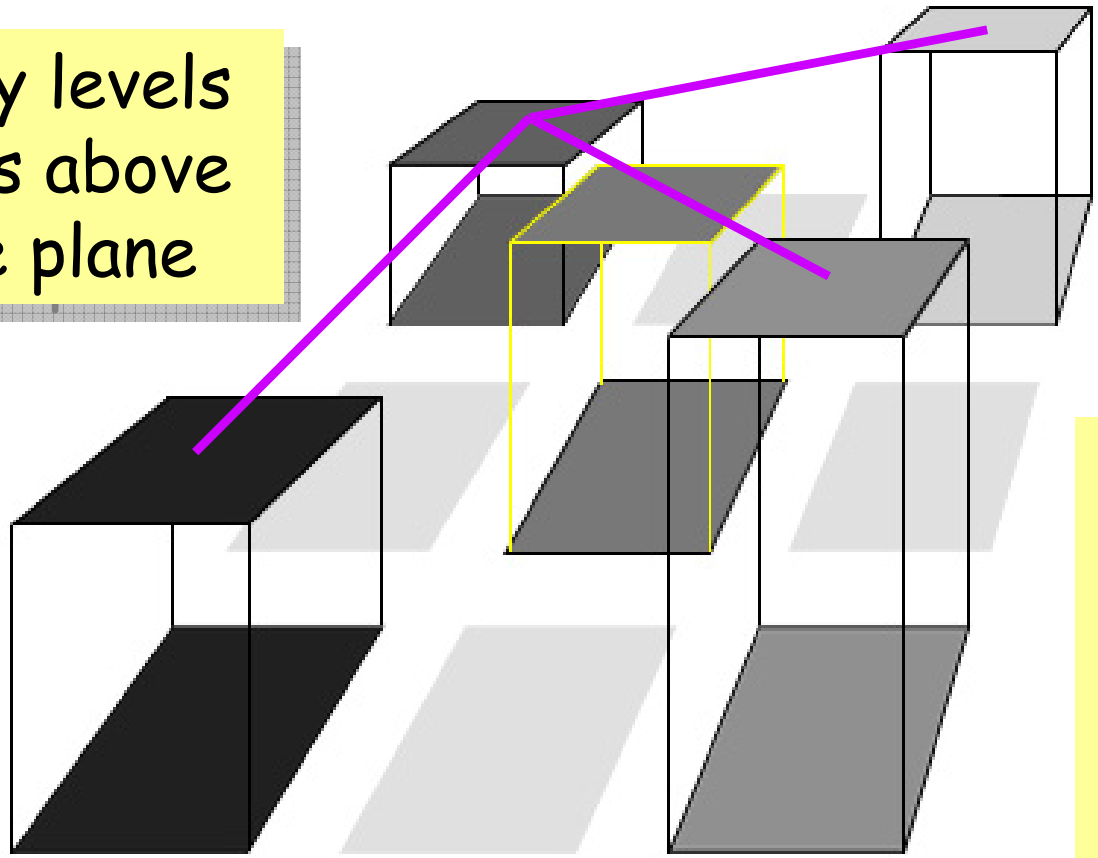
treat gray levels as heights above the image plane



center = weighted average of four corners

Resampling Through Bilinear Interpolation

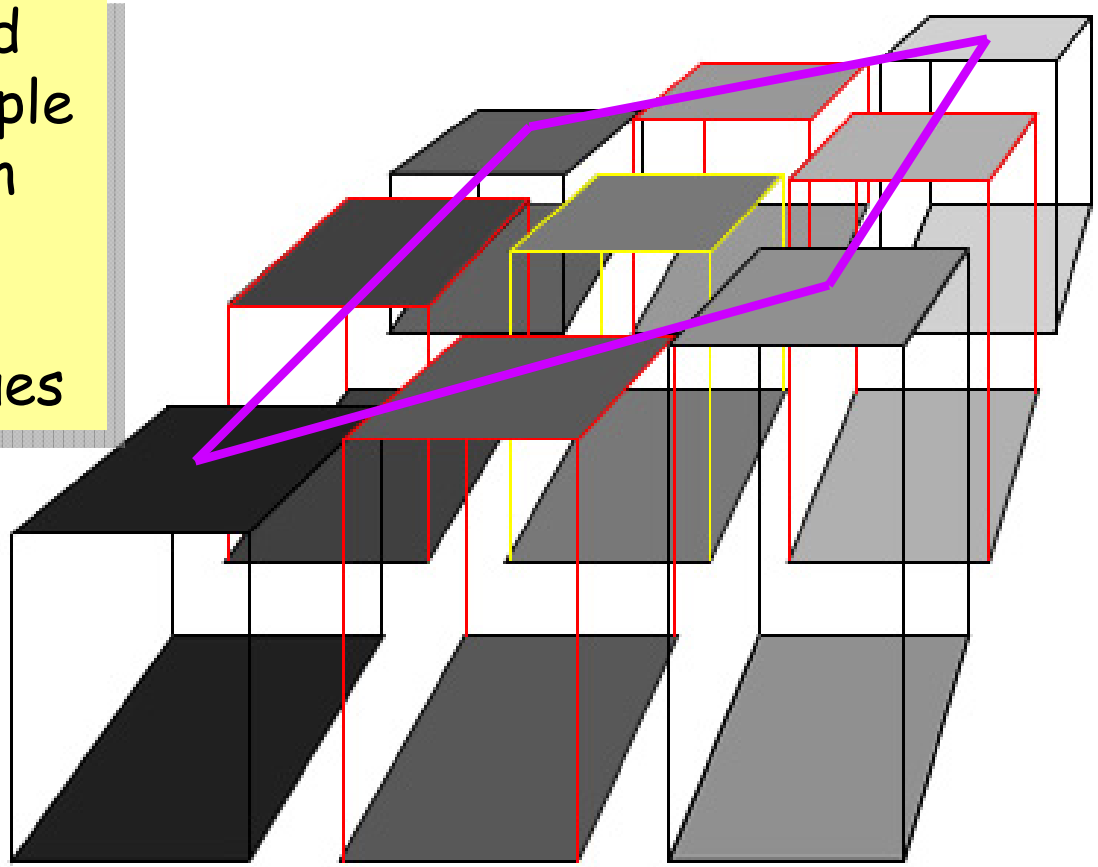
treat gray levels as heights above the image plane



center = weighted average of four corners

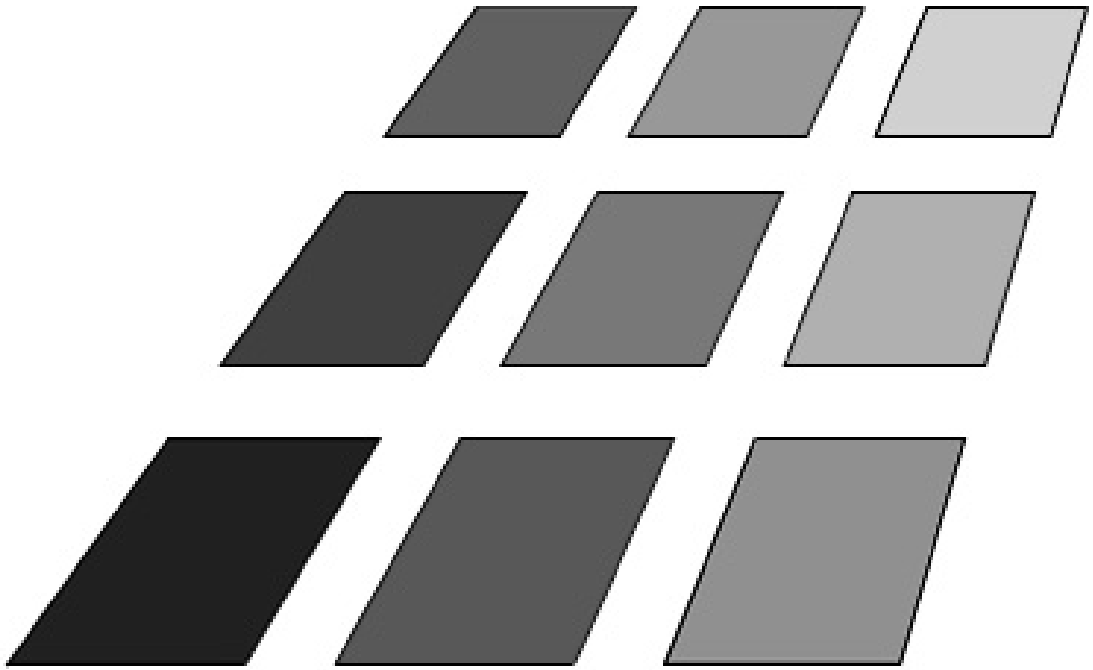
Resampling Through Bilinear Interpolation

new row and column sample values lie on the lines connecting the old values



Resampling Through Bilinear Interpolation

the results



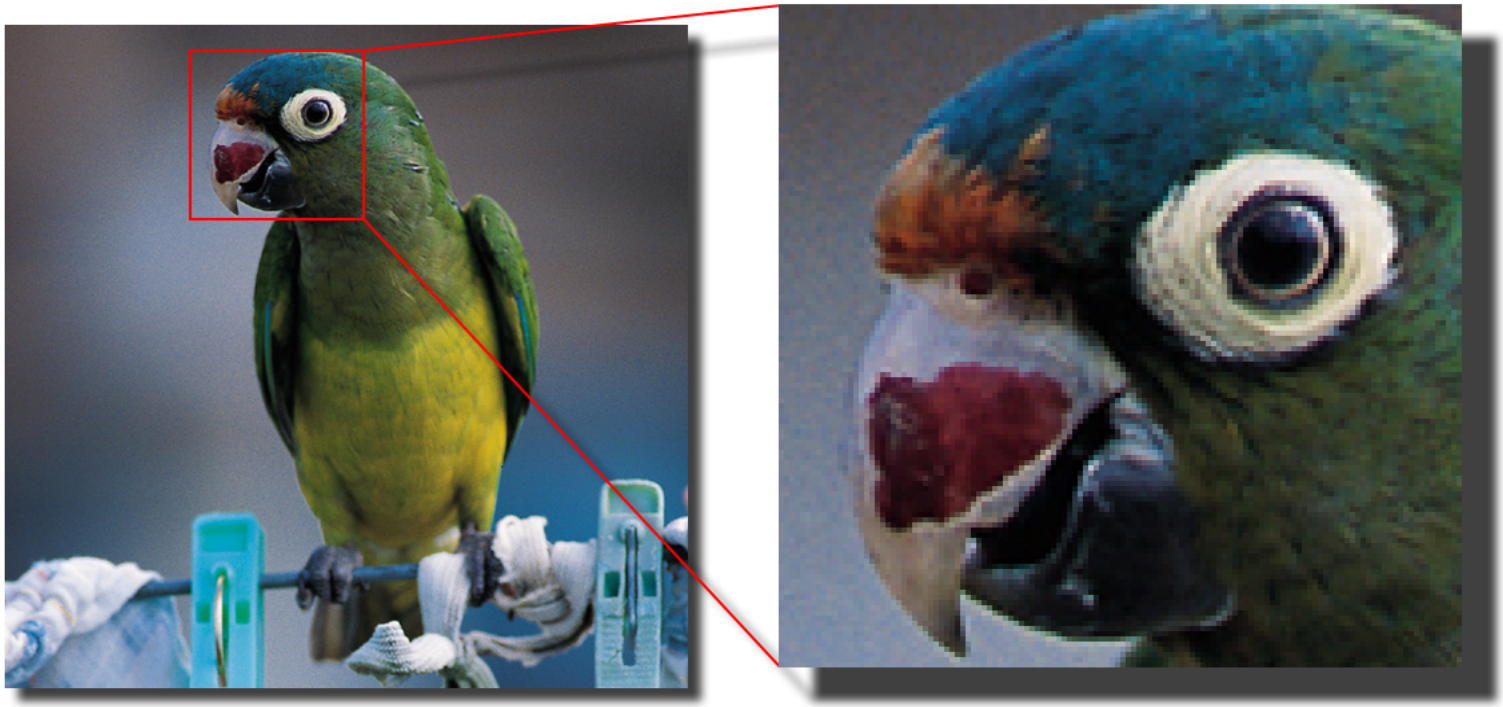
Bilinear Interpolation Example

We'll enlarge this image by a factor of 4 …

… via bilinear interpolation and compare it to a nearest neighbor enlargement.

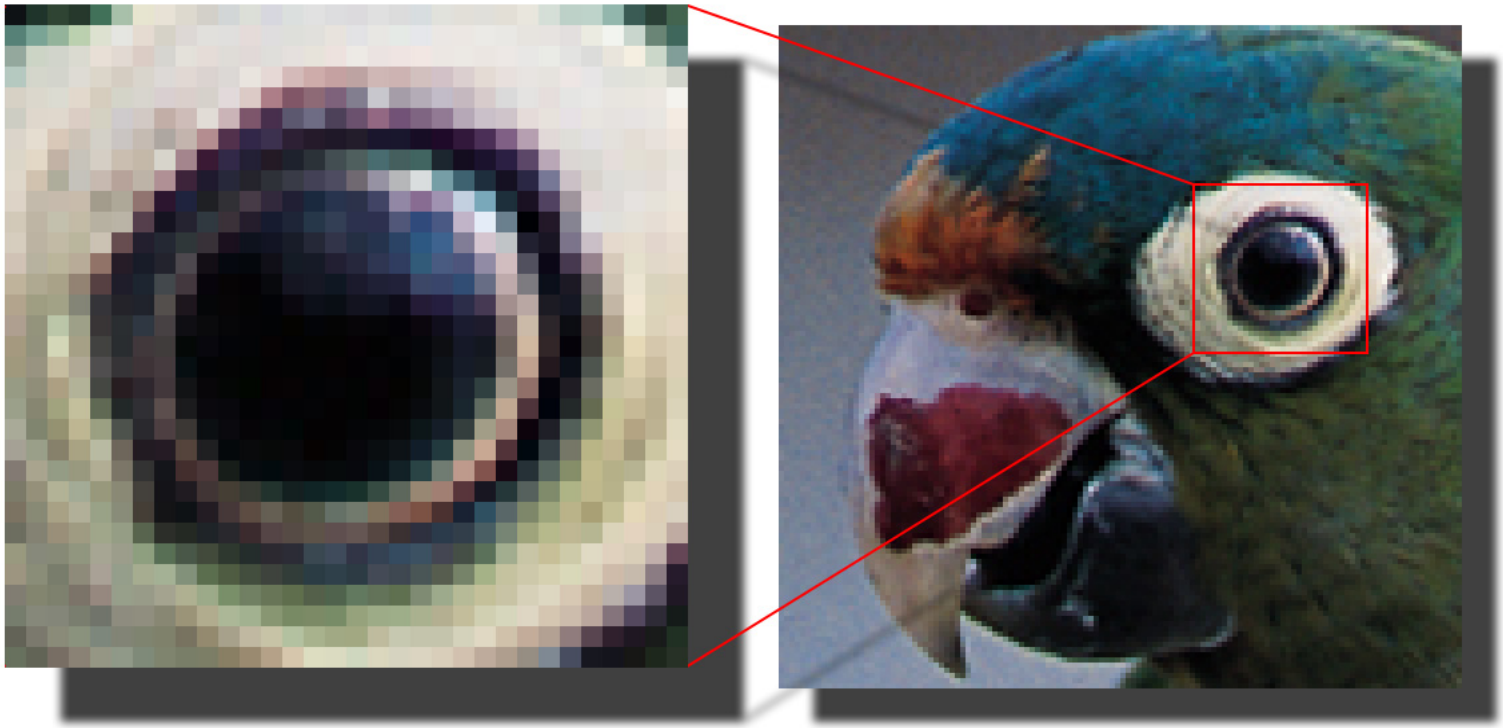
Example: Bilinear Interpolation – 4× Zoom

Original Image



To better see what happens, we'll look at the parrot's eye.

Example: Bilinear Interpolation – 4× Zoom



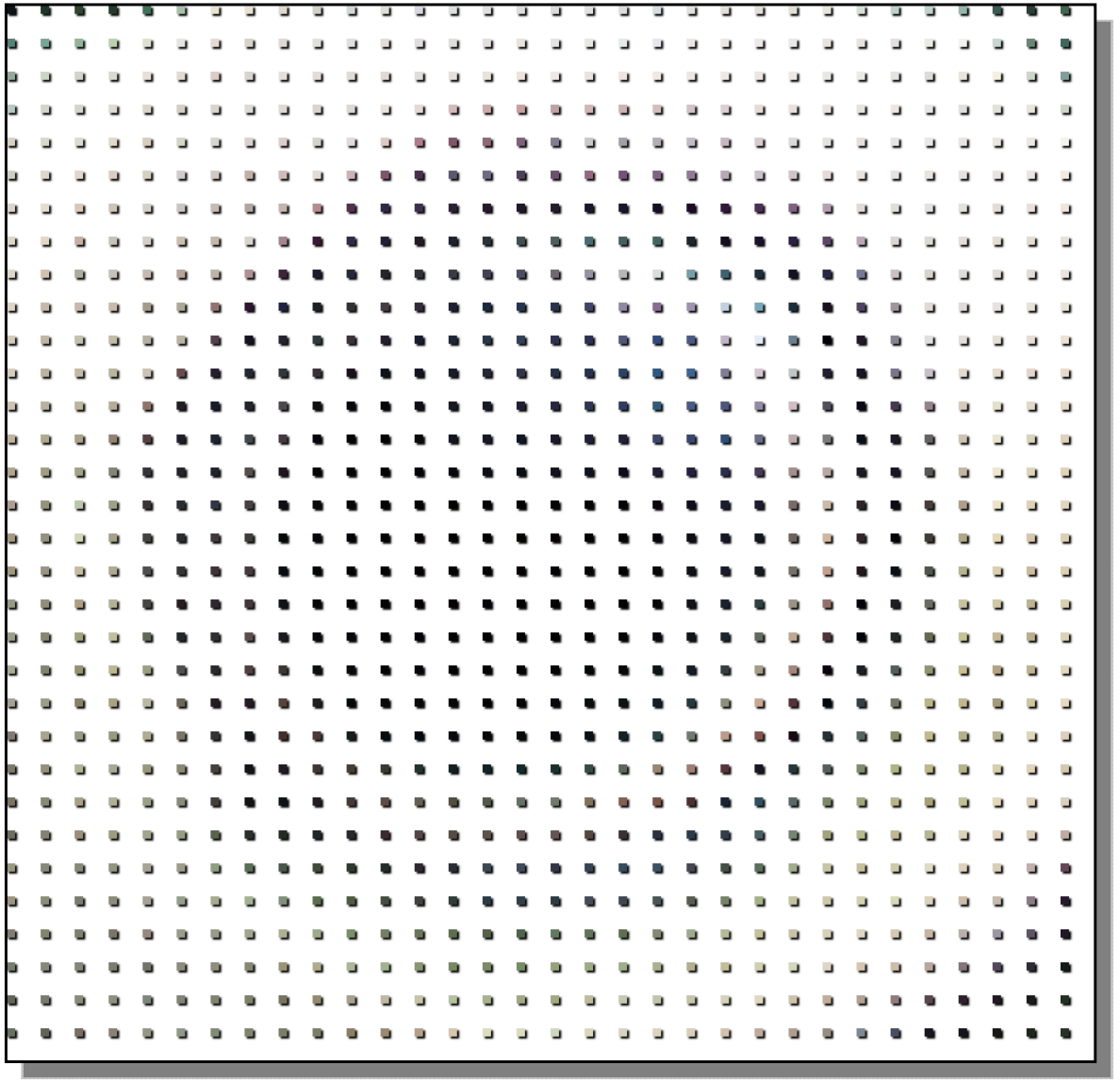
To better see what happens, we'll look at the parrot's eye.

Bilinear Interpolation

For a 4x zoom, create a blank image, four times the size of the original.

Bilinear Interpolation

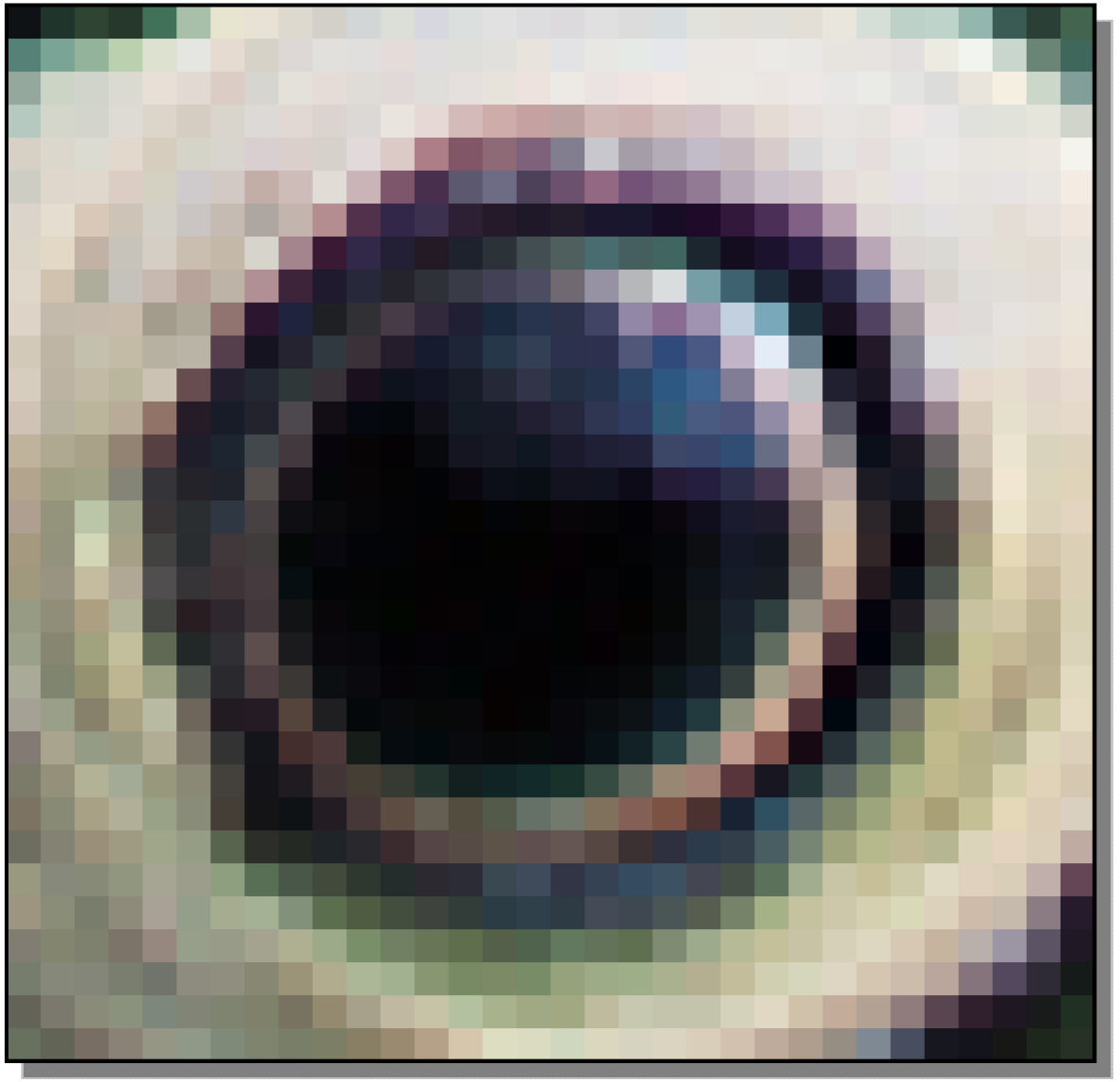
Then fill in every 4th pixel in every 4th row with the original values.



Bilinear Interpolation

Then fill in every 4th pixel in every 4th row with the original values.

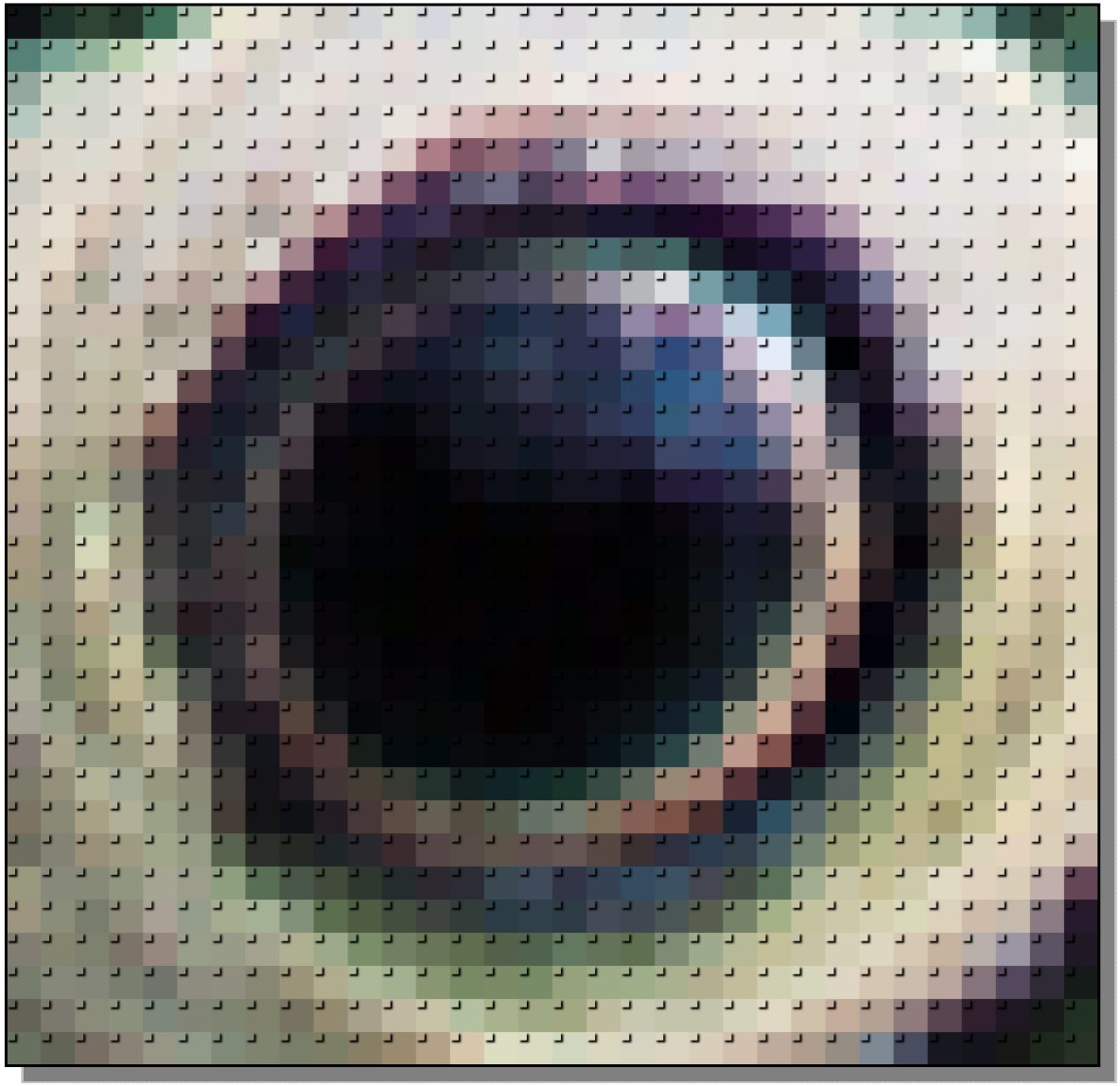
4x replication



Bilinear Interpolation

Then fill in every 4th pixel in every 4th row with the original values.

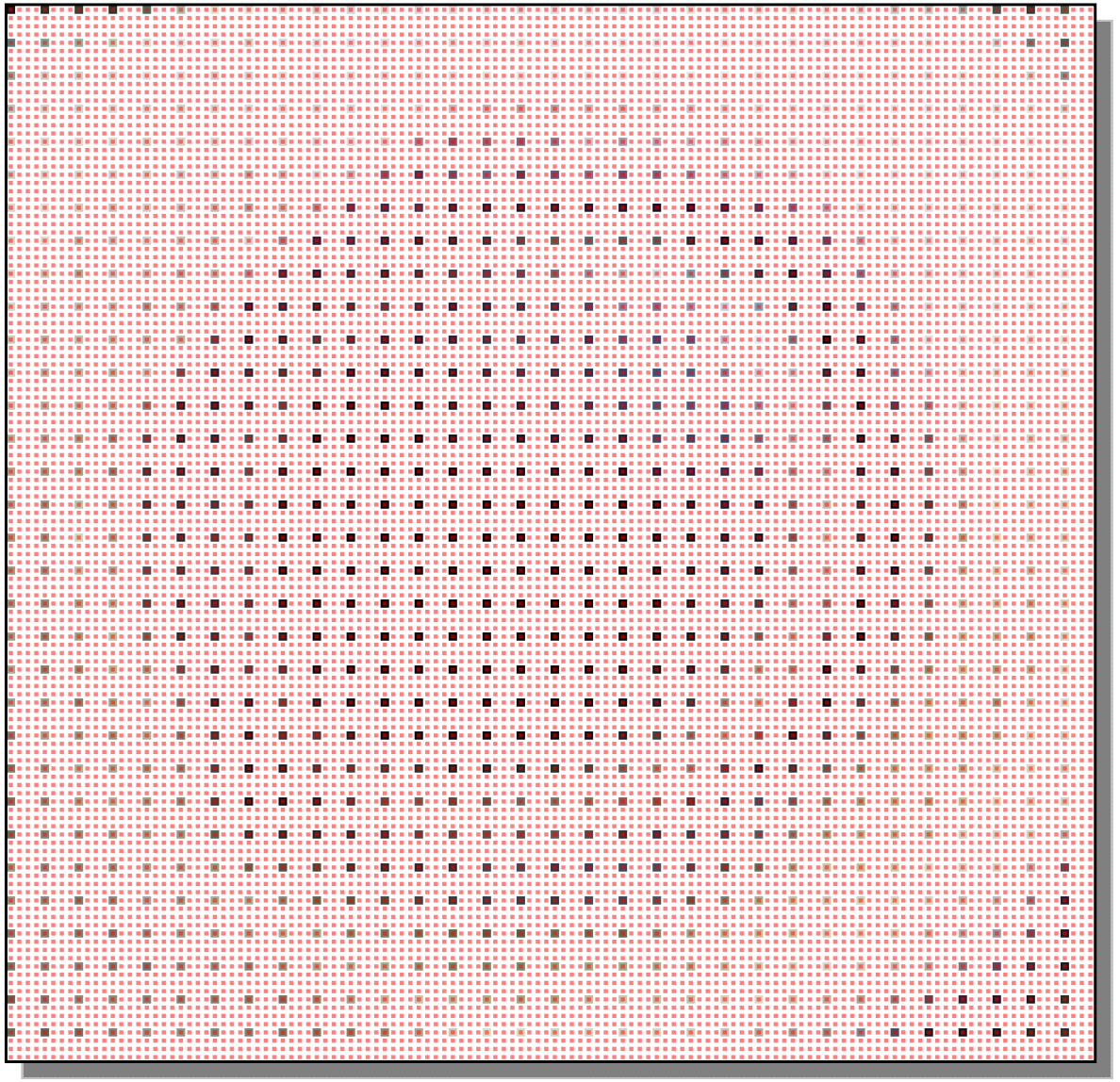
4x replication with the original pixels overlaid.



Bilinear Interpolation

Next you want to fill in the other 15 pixels in each block with the intermediate values.

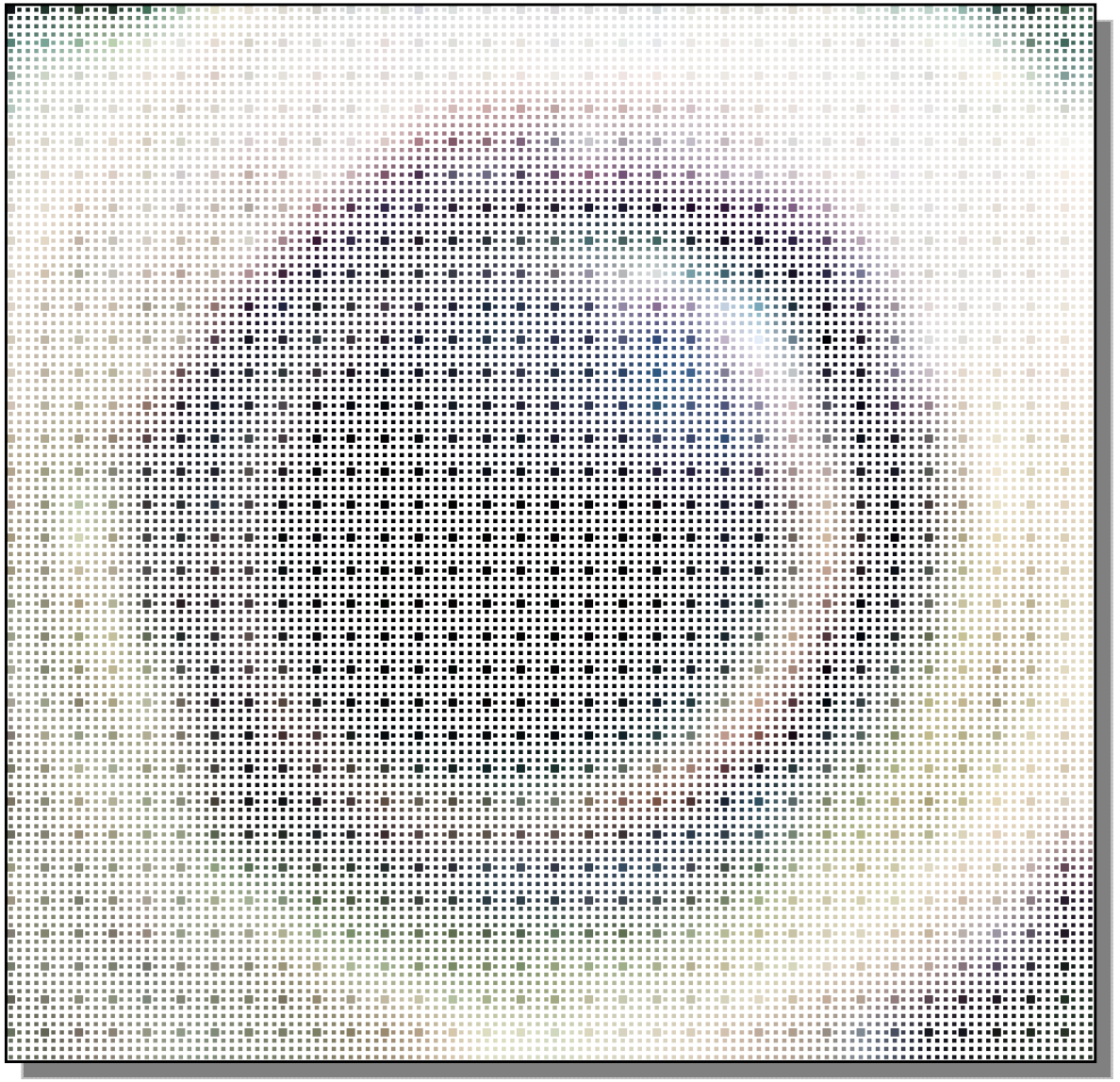
Each pink dot is a pixel location.



Bilinear Interpolation

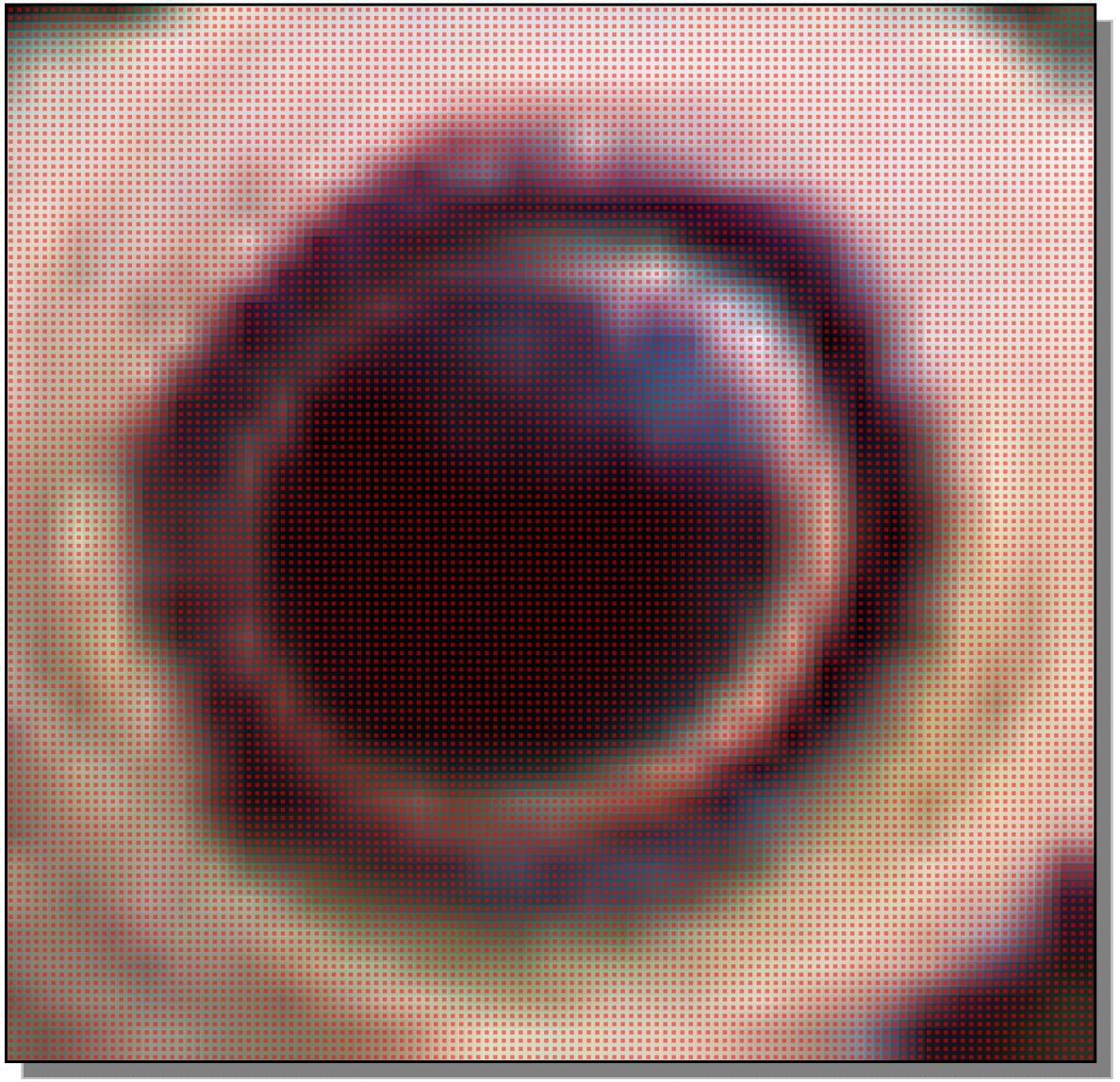
Next you want to fill in the other 15 pixels in each block with the intermediate values.

Intermediate values filled in. Spaces left so individual pixels can be seen.



Bilinear Interpolation

Next you want to fill in the other 15 pixels in each block with the intermediate values.



Intermediate values filled in. Red dots mark individual pixels.

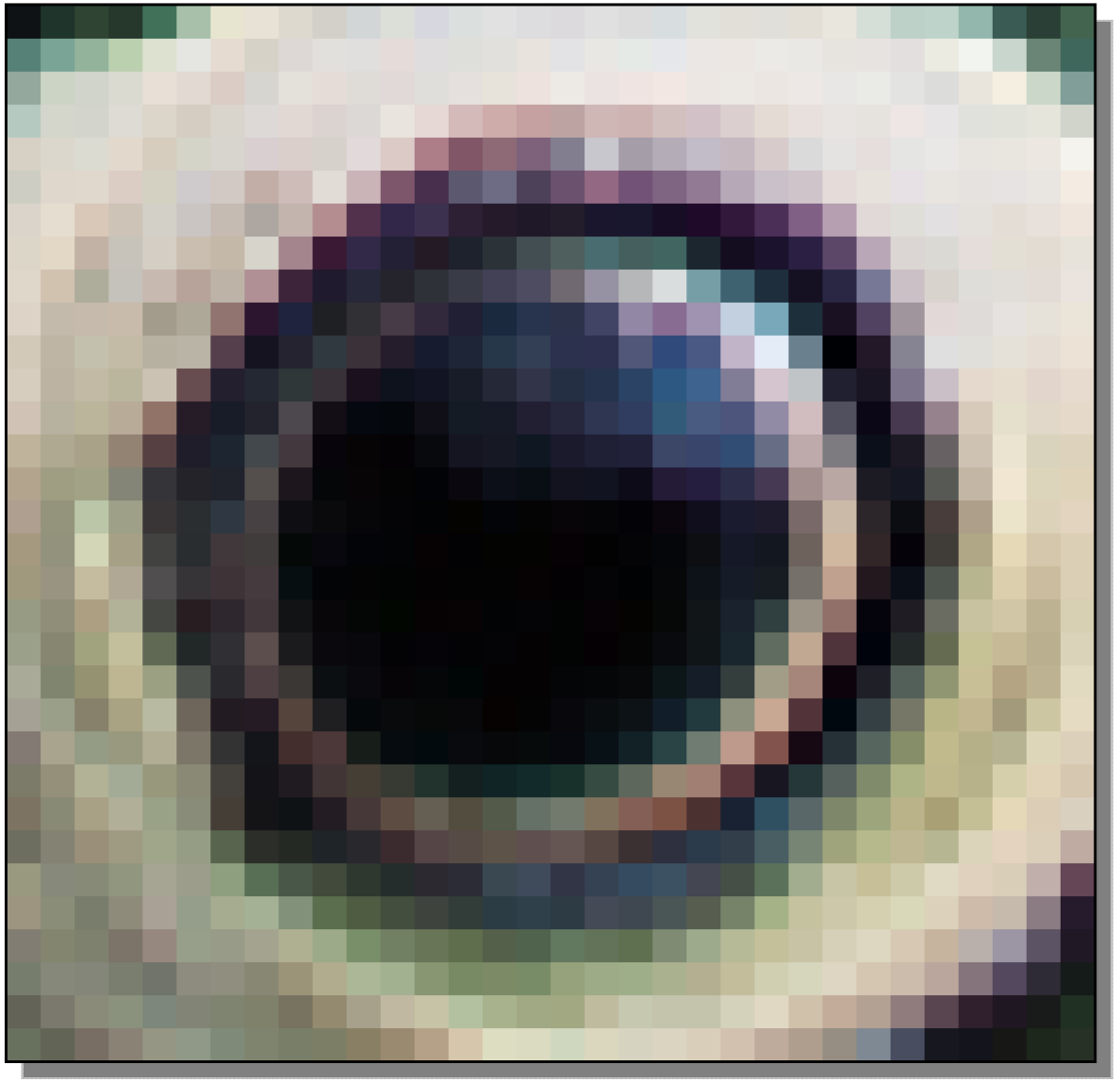
Bilinear Interpolation

The result:

Compare to the next slide which contains a 4x pixel zoom via pixel replication.

Pixel Replication

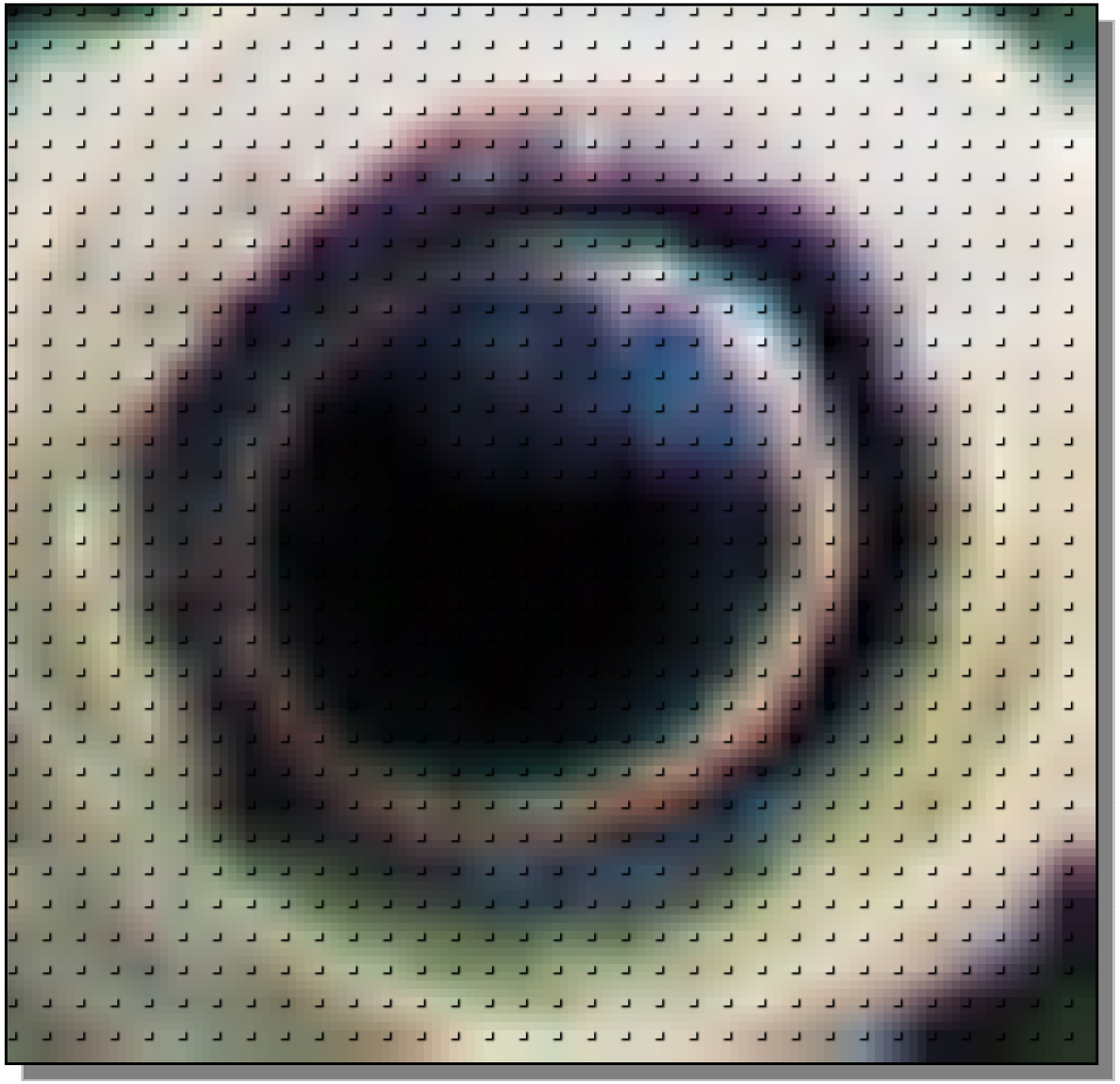
The result:



Compare to the prev. slide which contains a 4x pixel zoom via bilinear interp.

Bilinear Interpolation

Result with original pixels marked:

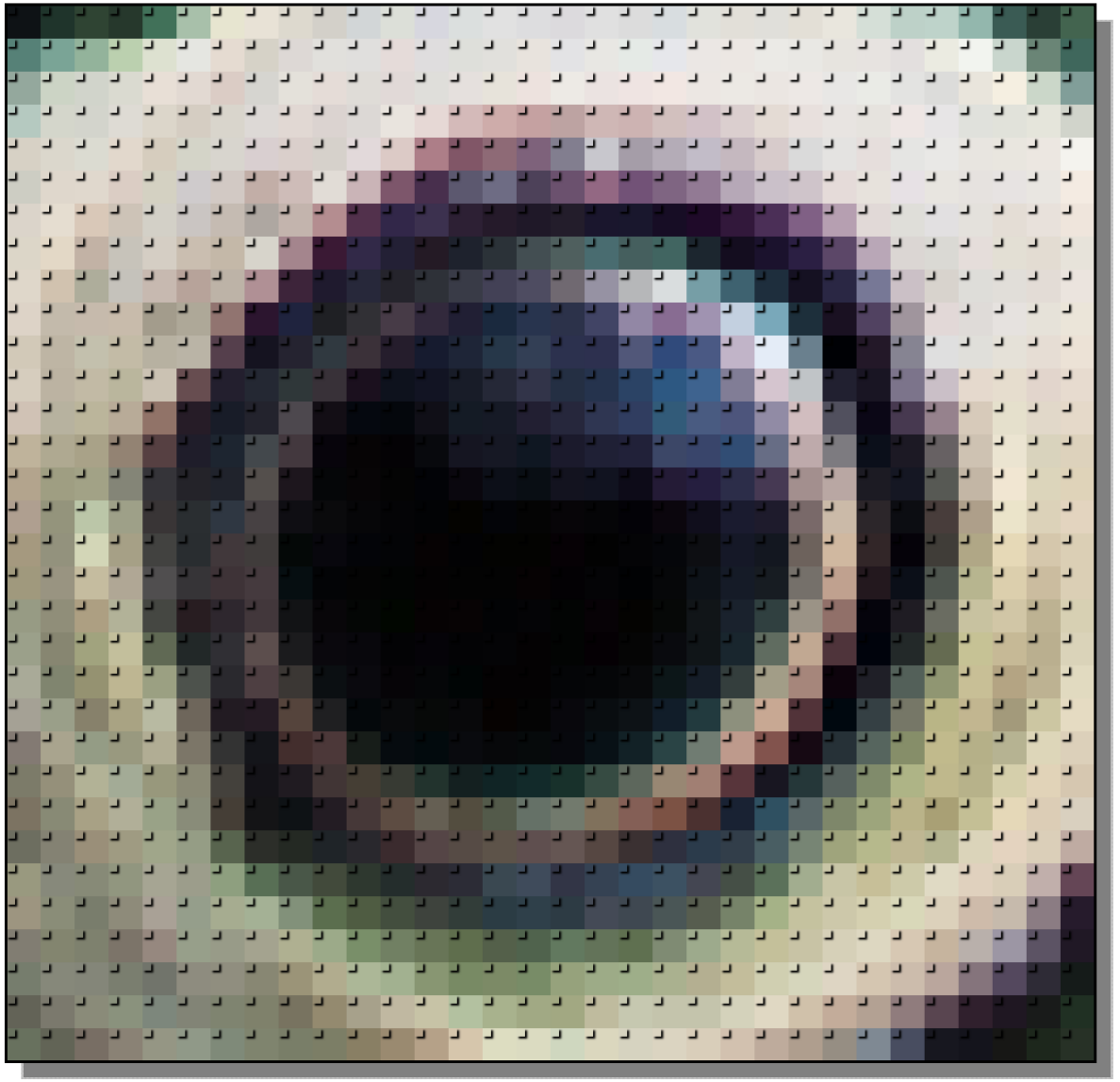


Compare to the next slide which contains a 4x pixel zoom via pixel replication.

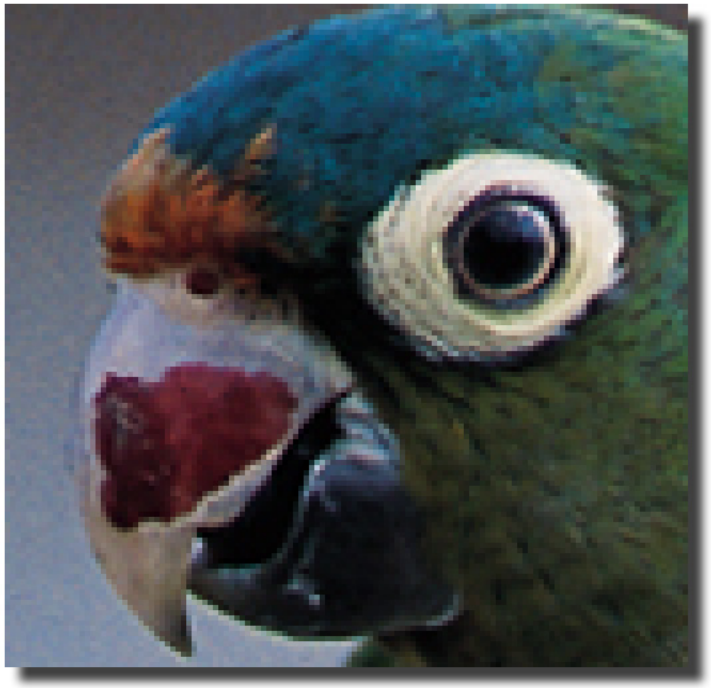
Pixel Replication

Result with original pixels marked:

Compare to the prev. slide which contains a 4x pixel zoom via bilinear interp.



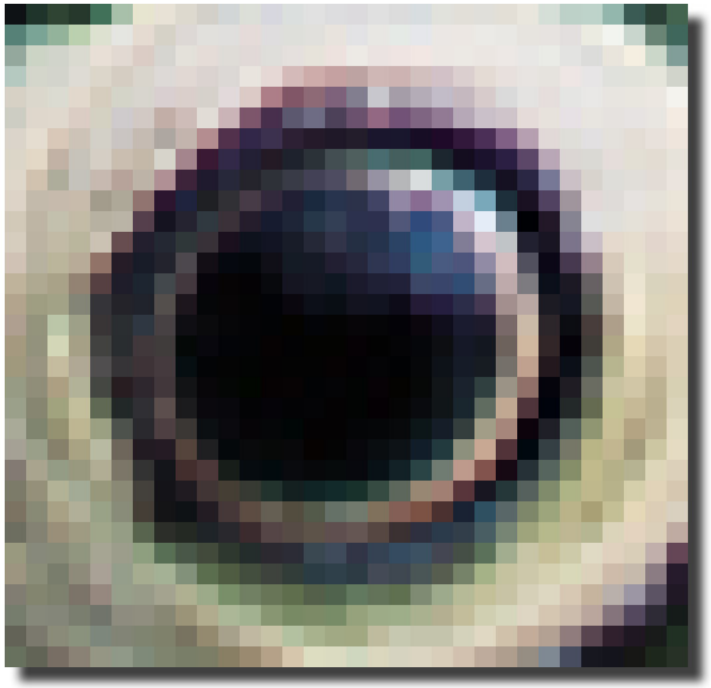
Pixel Replication vs. Bilinear Interpolation



Pixel replication

Bilinear interpolation

Pixel Replication vs. Bilinear Interpolation



Pixel replication

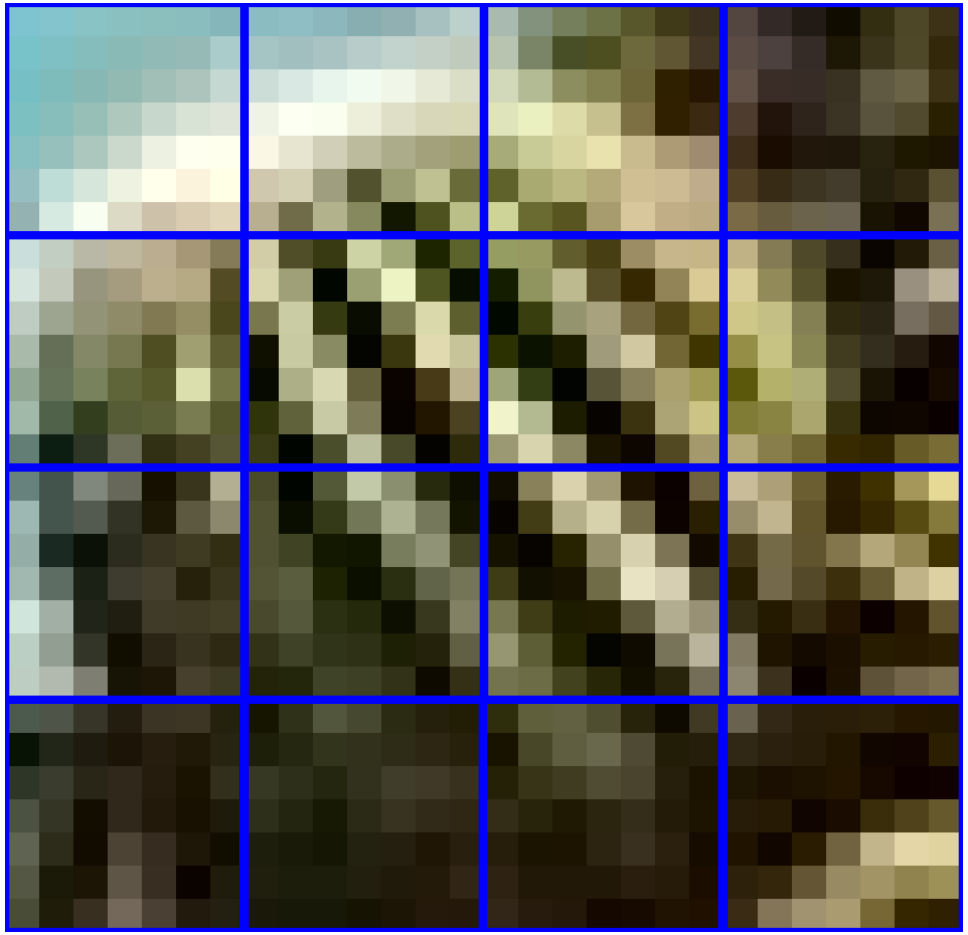
Bilinear interpolation

Resampling Through Bilinear Interpolation

Example: reduce the cactus image to 3/7 its original size using bilinear interpolation

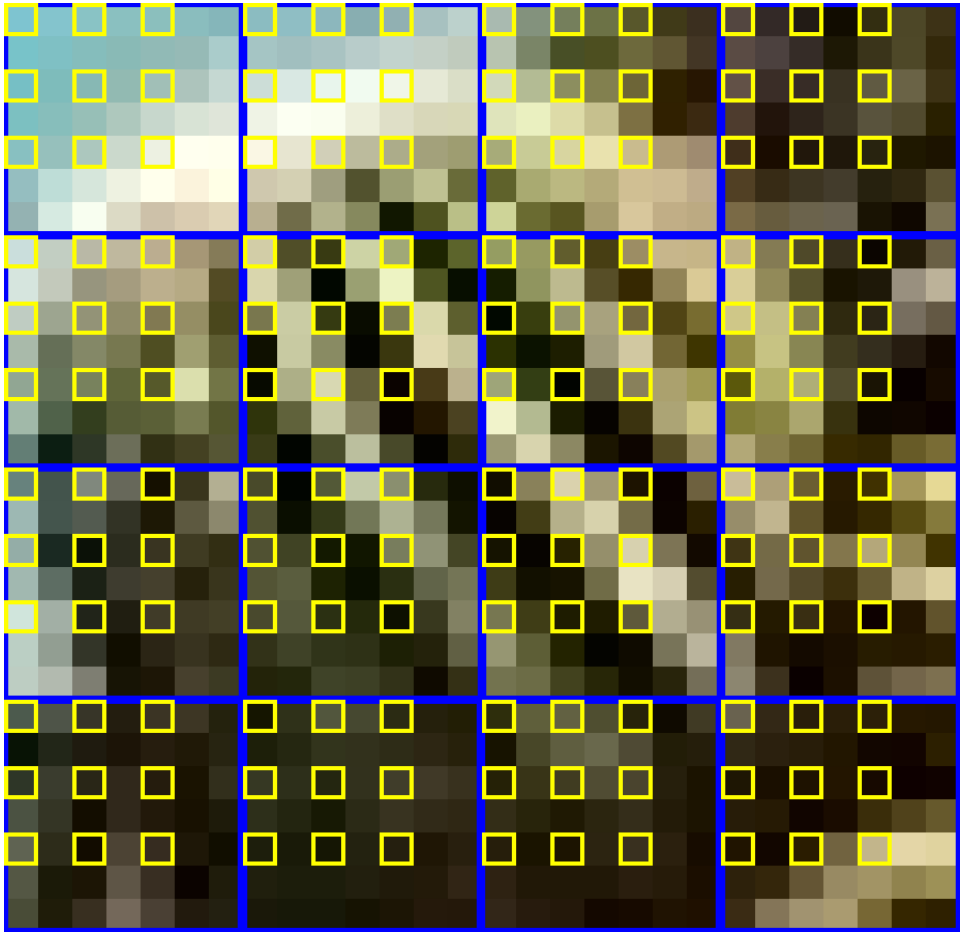
Resampling Through Bilinear Interpolation

For each 7x7 block of pixels select 3x3 = 9 pixel locations.



Resampling Through Bilinear Interpolation

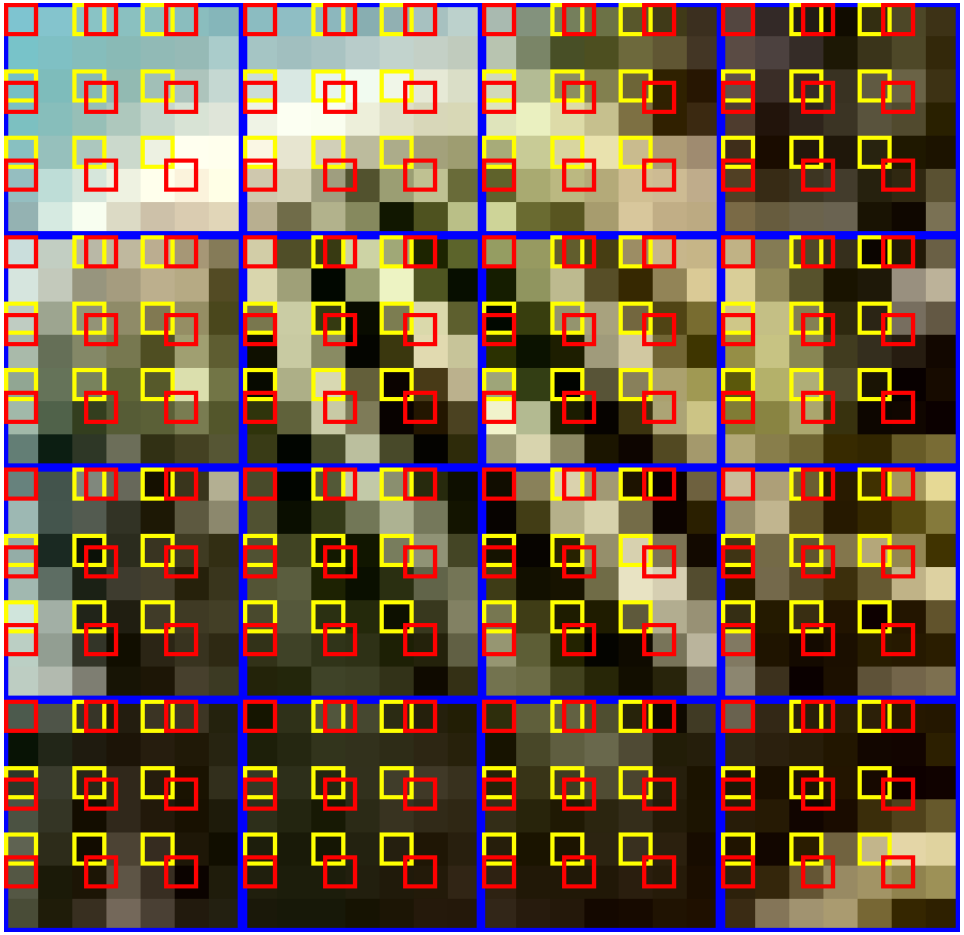
For nearest neighbor sampling the 9 pixel locations correspond to pixel locations in the original image.



Nearest neighbor selected pixels outlined in yellow.

Resampling Through Bilinear Interpolation

In bilinear interpolation the 9 pixel locations are distributed evenly.

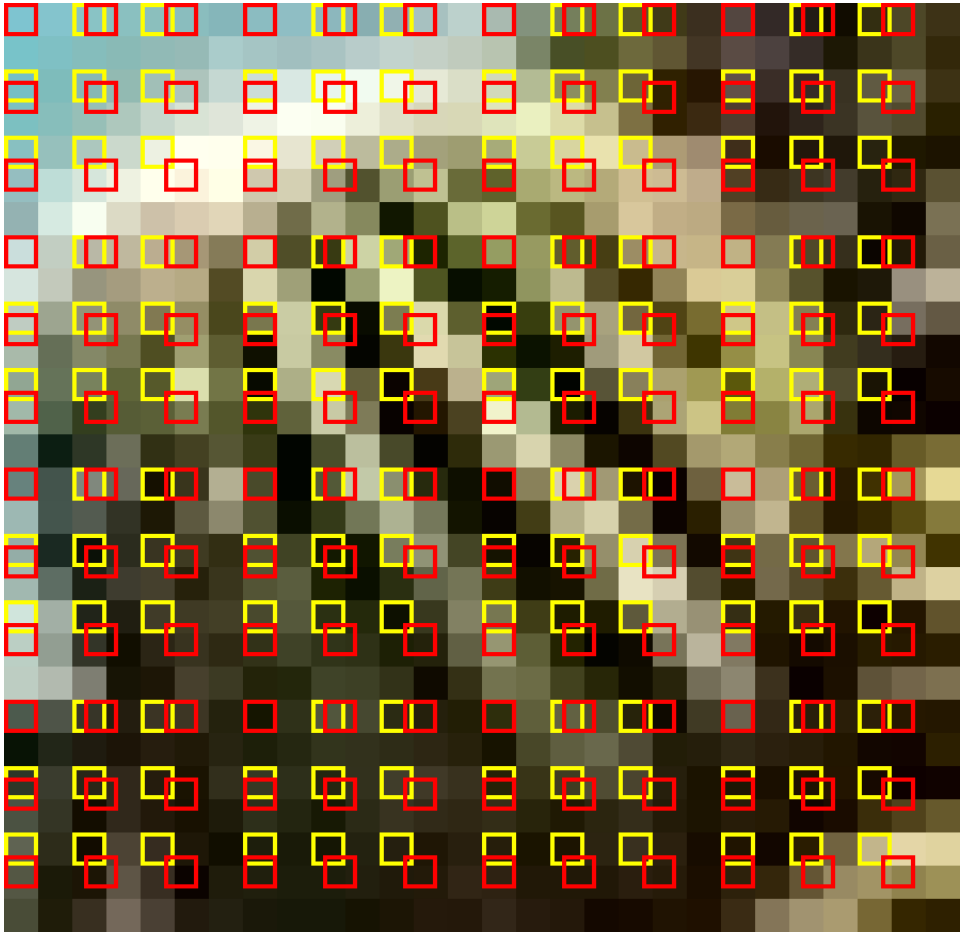


Nearest neighbor selected pixels outlined in yellow.

Bilinear interp. pixels locations outlined in red.

Resampling Through Bilinear Interpolation

In bilinear interpolation the 9 pixel locations are distributed evenly.

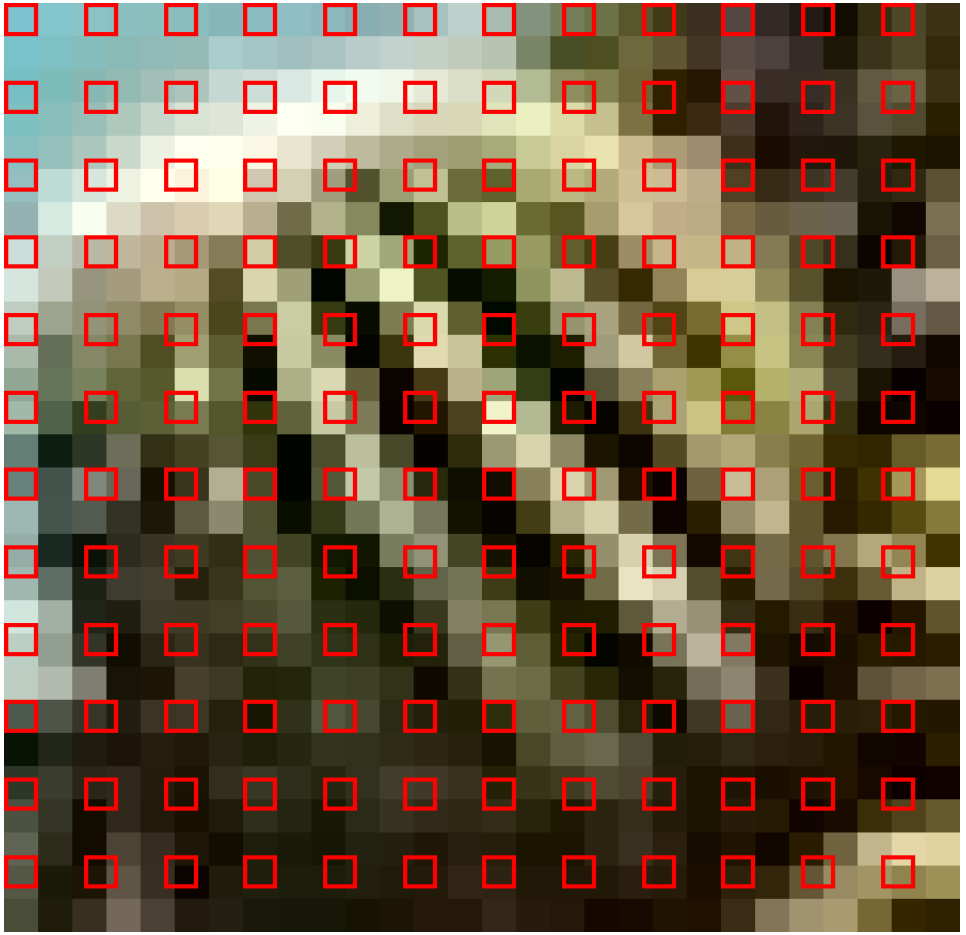


Nearest neighbor selected pixels outlined in yellow.

Bilinear interp. pixels locations outlined in red.

Resampling Through Bilinear Interpolation

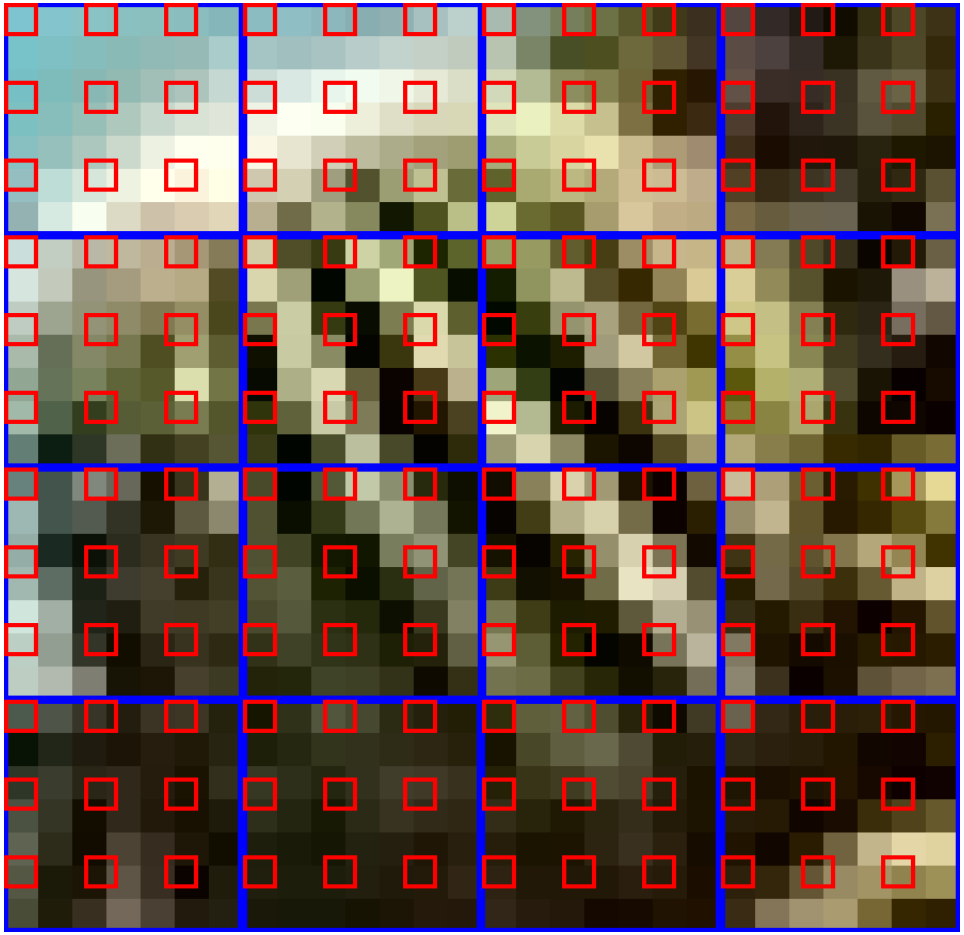
In bilinear interpolation the 9 pixel locations are distributed evenly.



Notice that the locations overlap pixels in the original image.

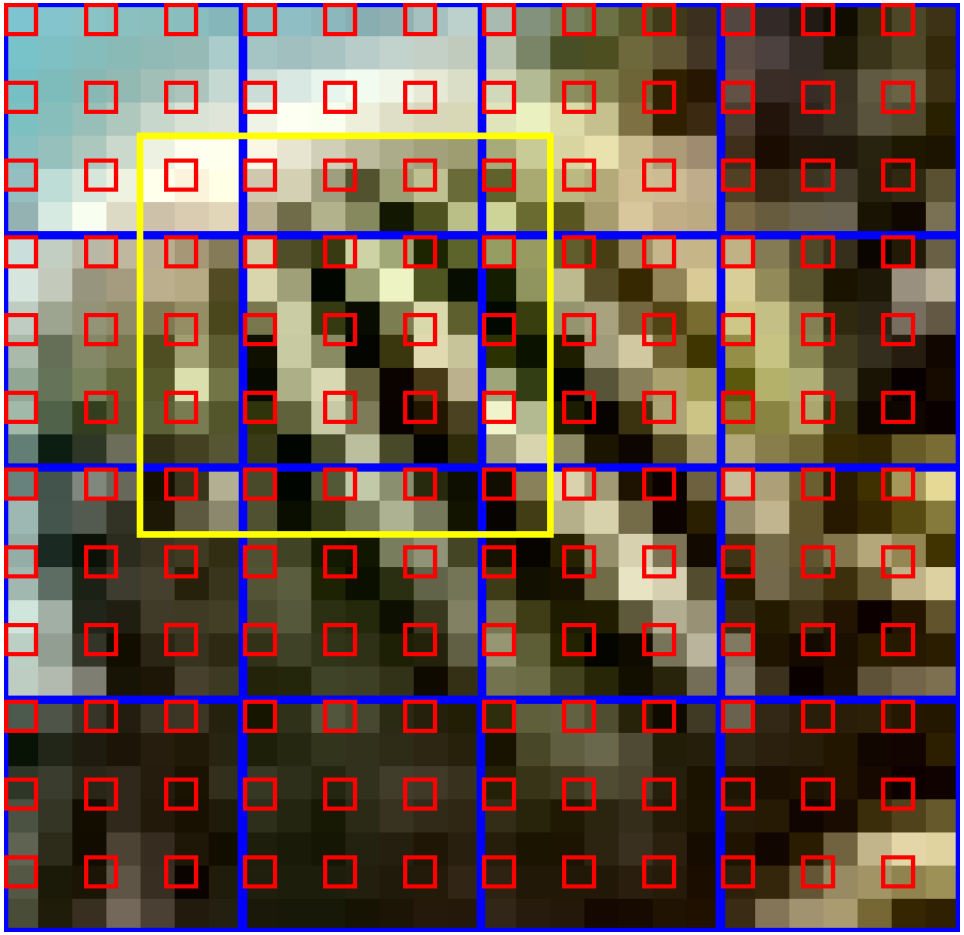
Resampling Through Bilinear Interpolation

For each 7x7 block of pixels select 3x3 = 9 pixel locations.

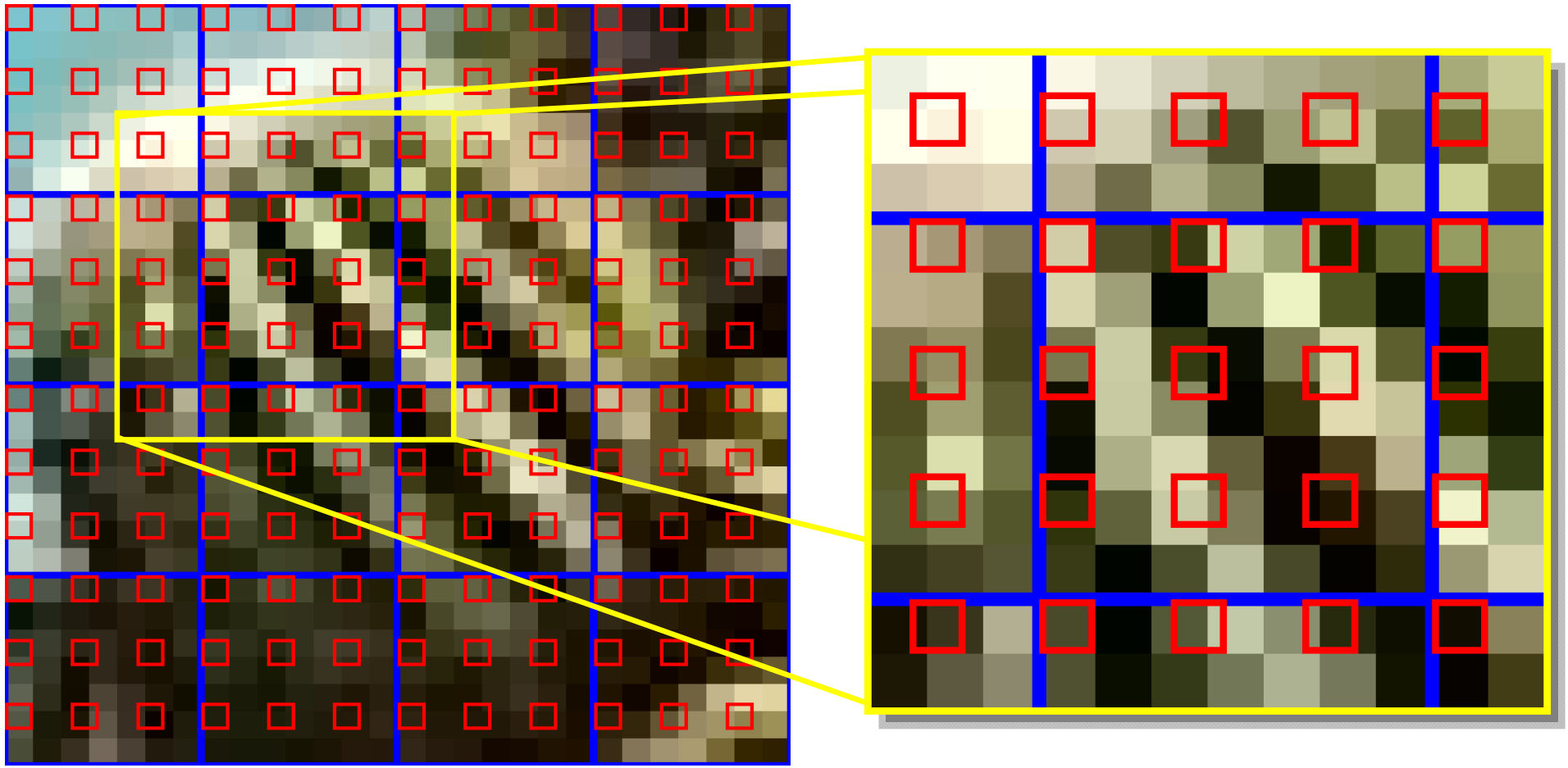


Resampling Through Bilinear Interpolation

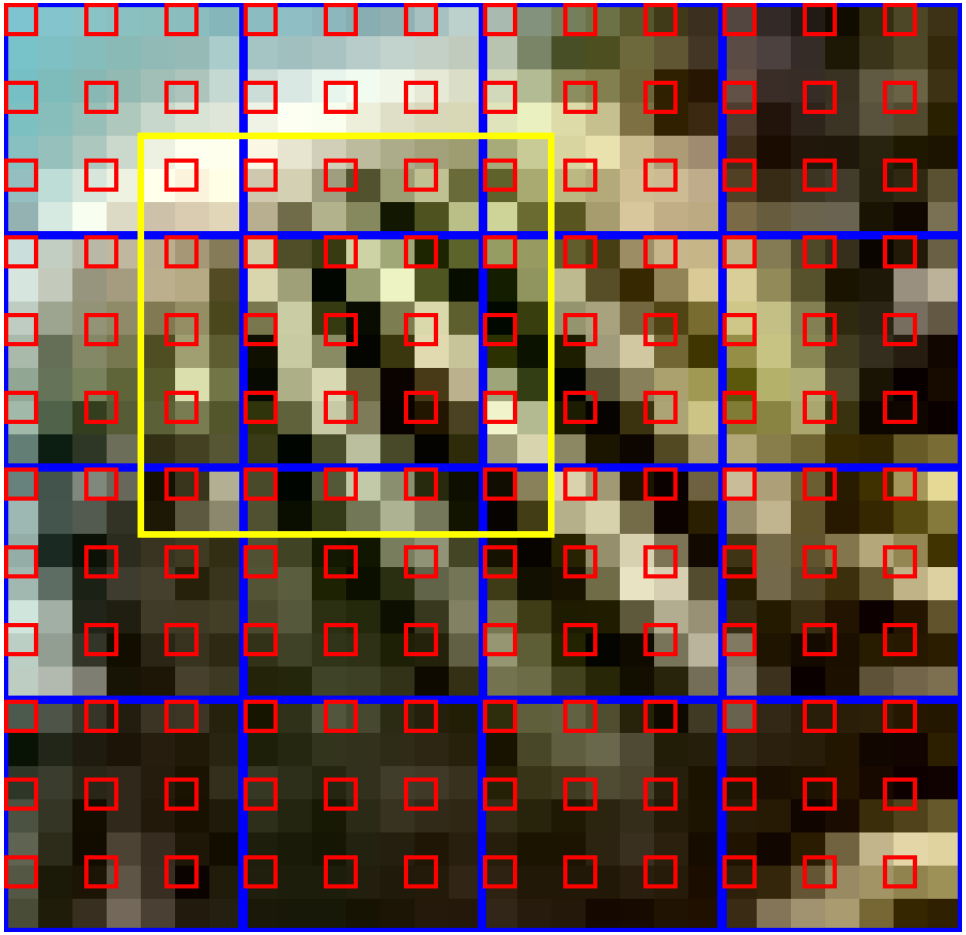
Examine one section in detail.

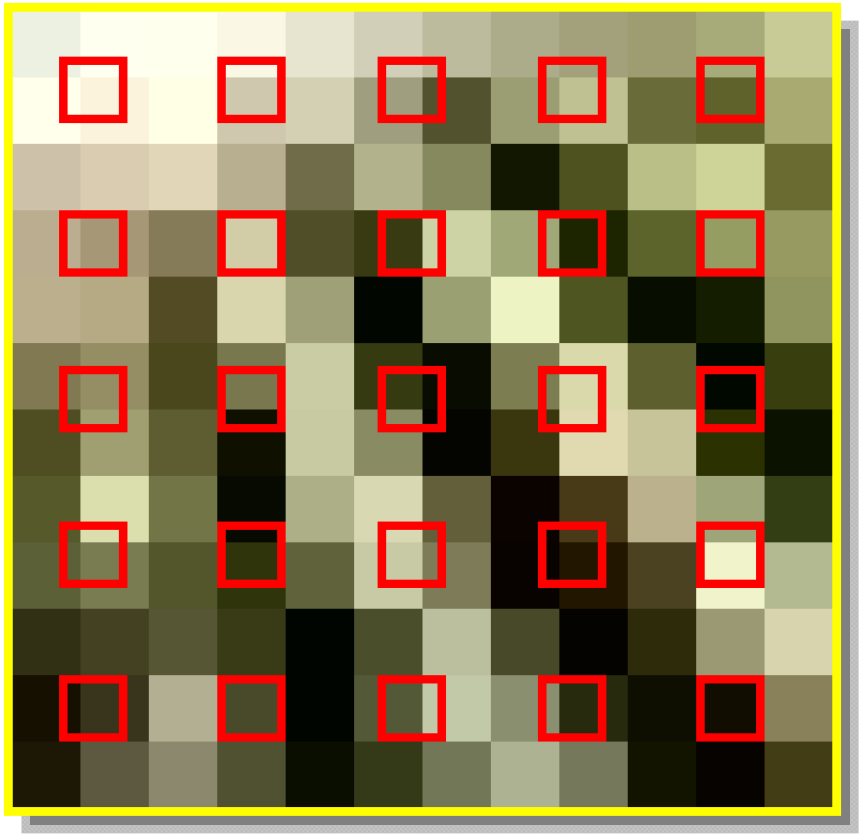


Resampling Through Bilinear Interpolation

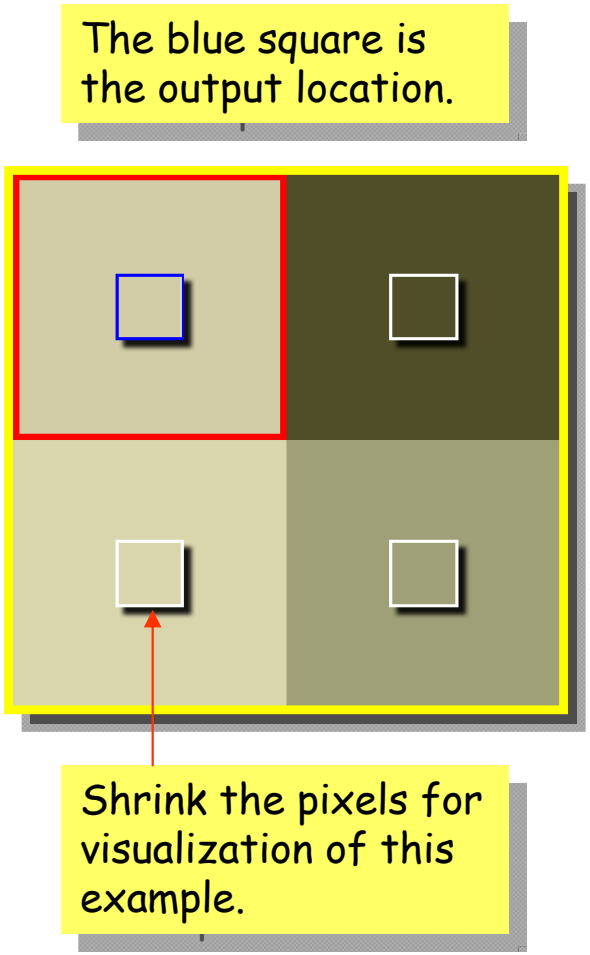


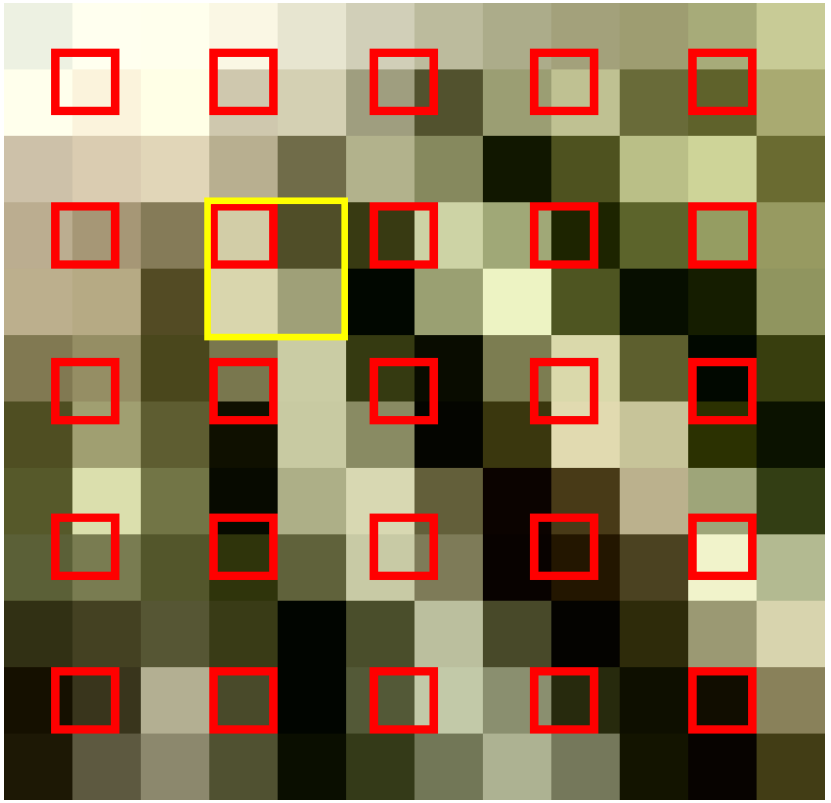
Resampling Through Bilinear Interpolation



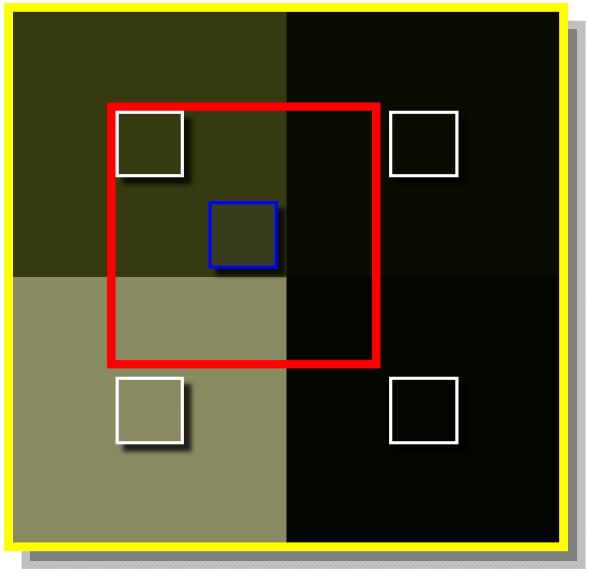


Resampling Through Bilinear Interpolation

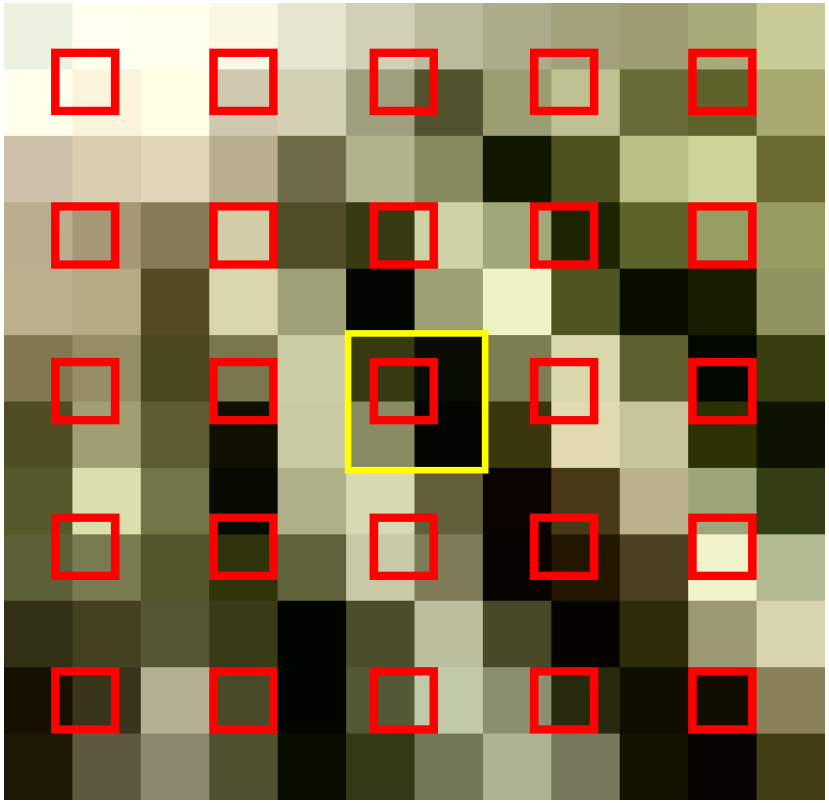




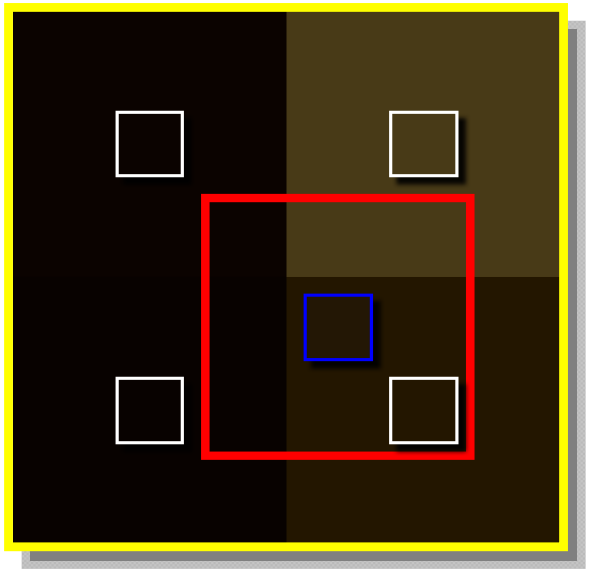
Resampling Through Bilinear Interpolation



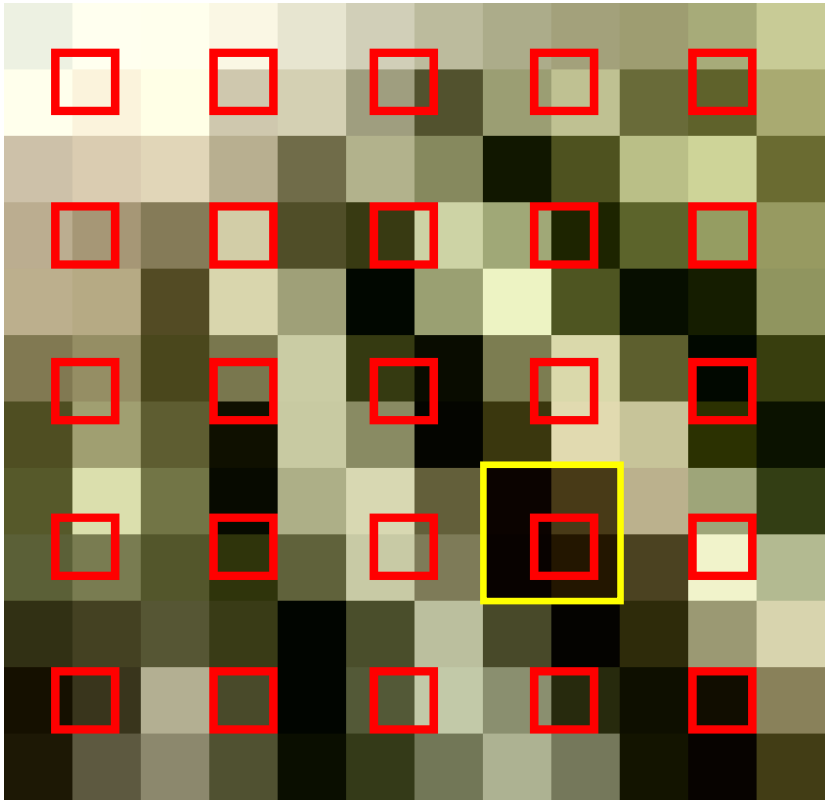
The blue square is the output location.



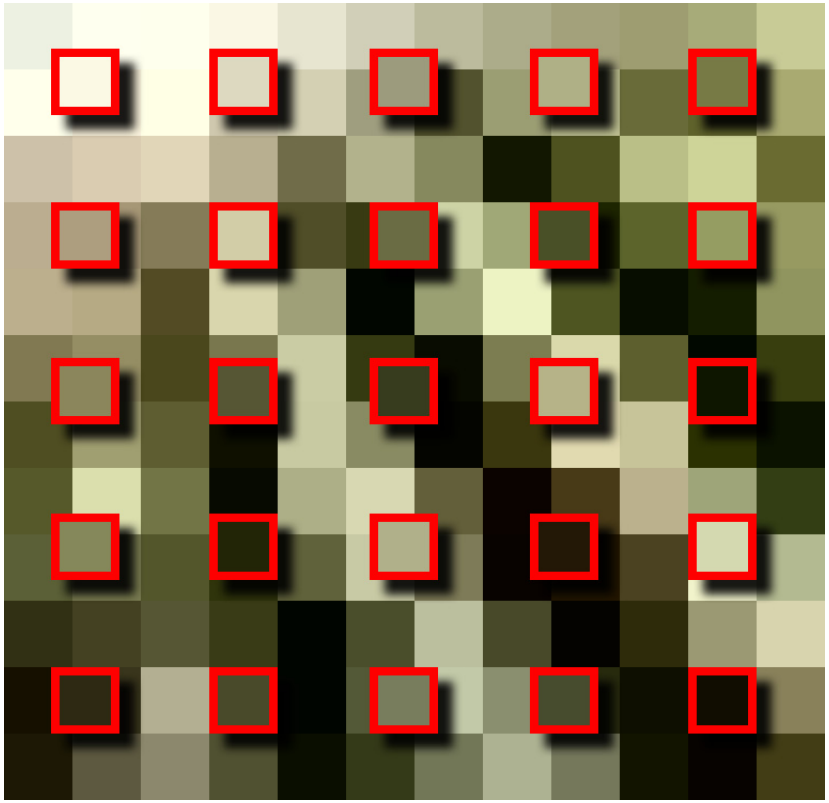
Resampling Through Bilinear Interpolation



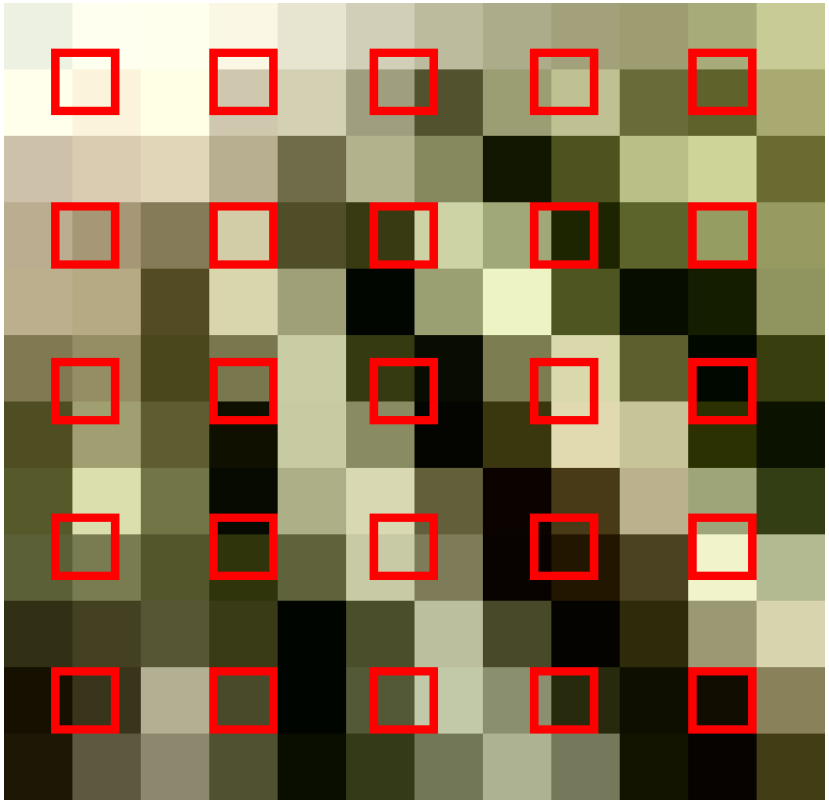
The blue square is the output location.



Resampling Through Bilinear Interpolation

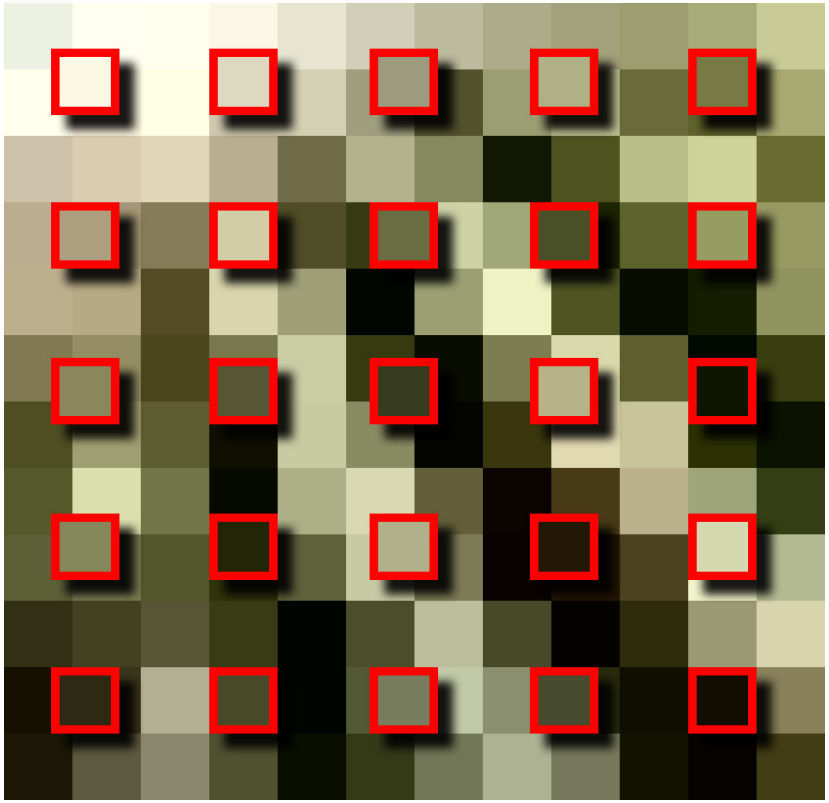


locs & colors of new pixels



locations of new pixels

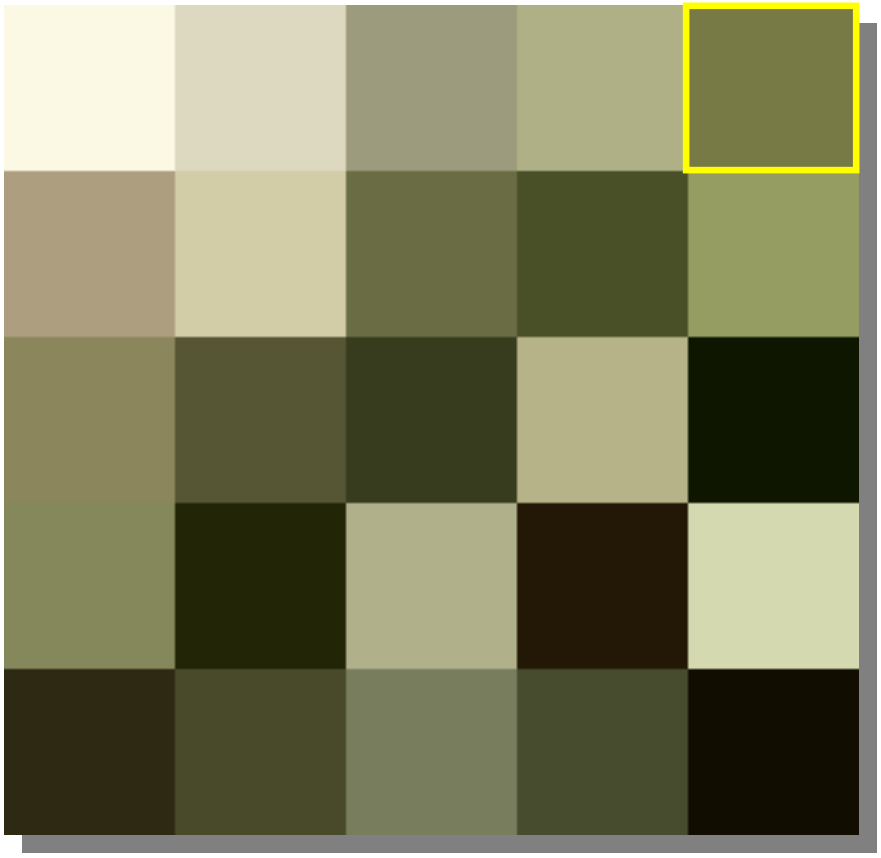
Resampling Through Bilinear Interpolation



locs & colors of new pixels

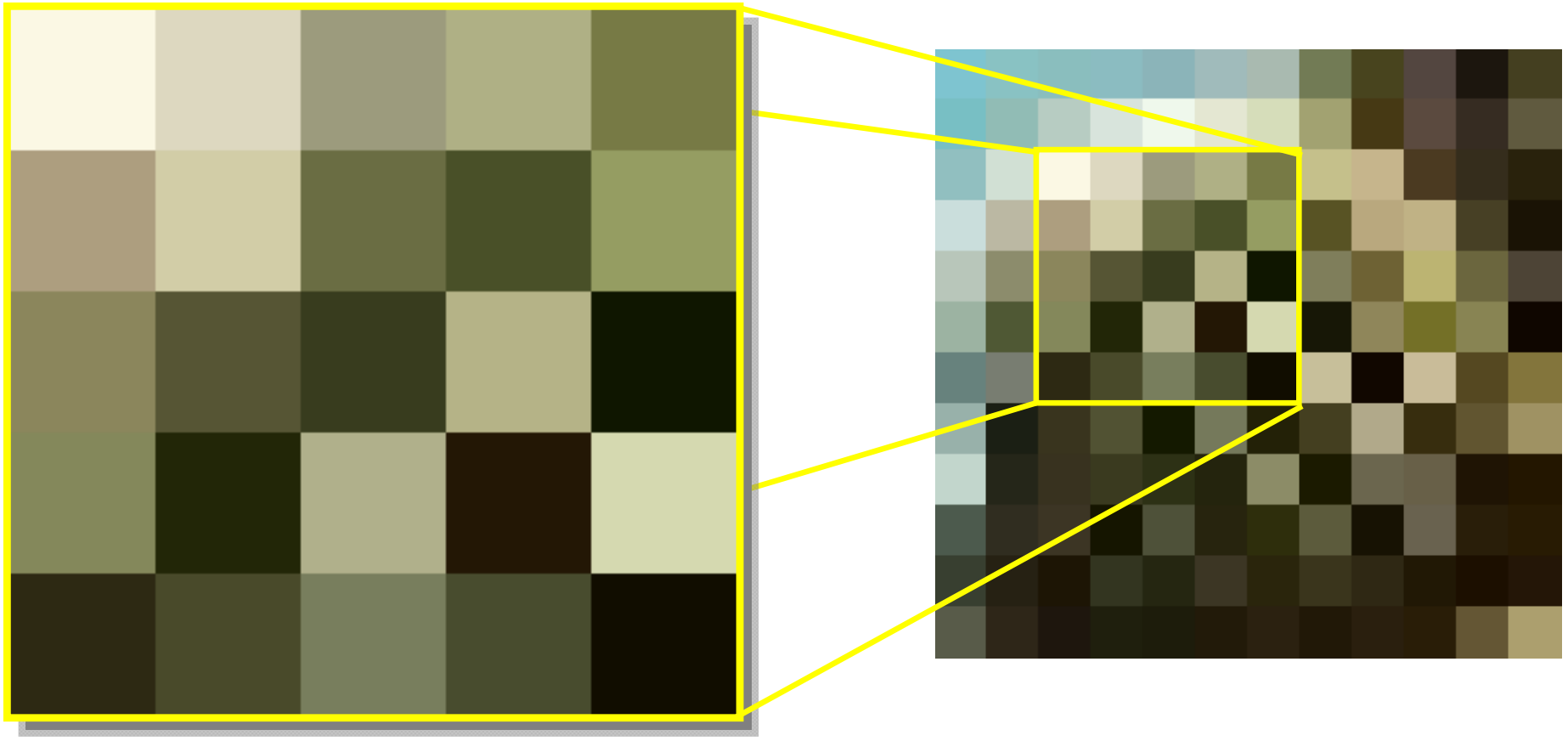
locs & colors of new pixels

Resampling Through Bilinear Interpolation

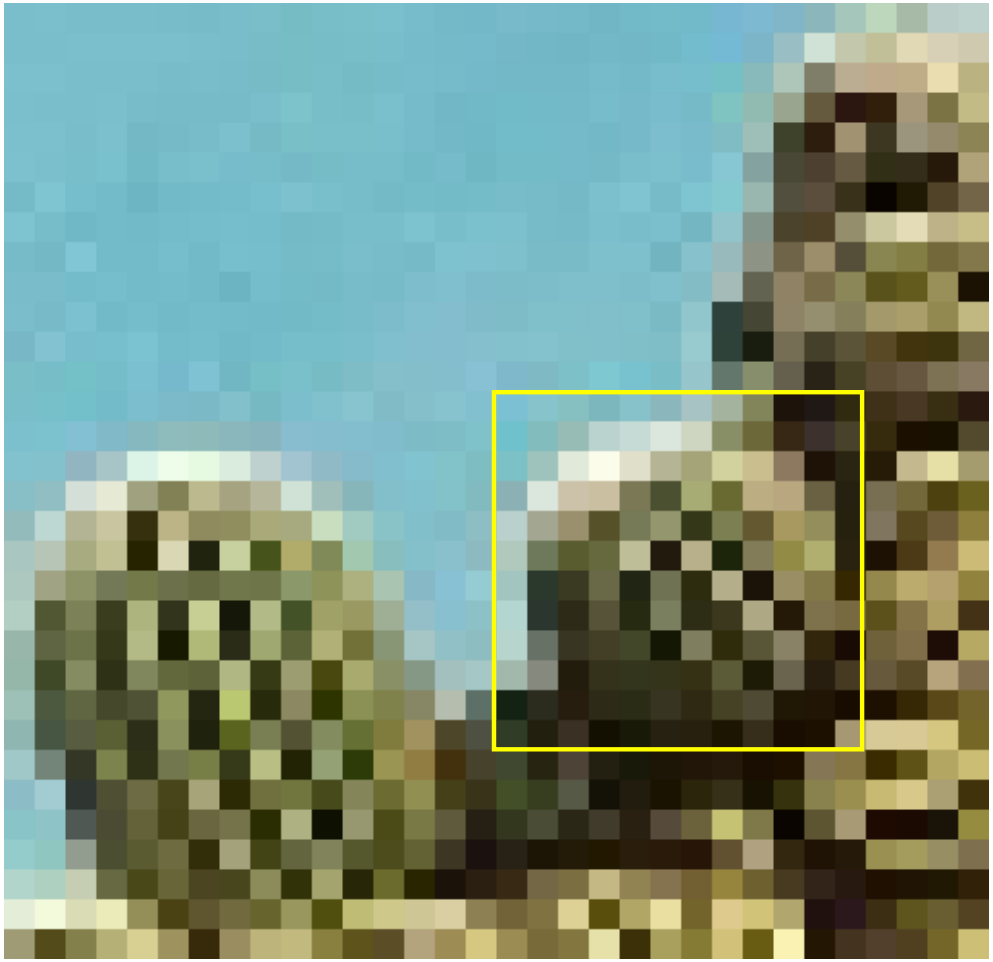


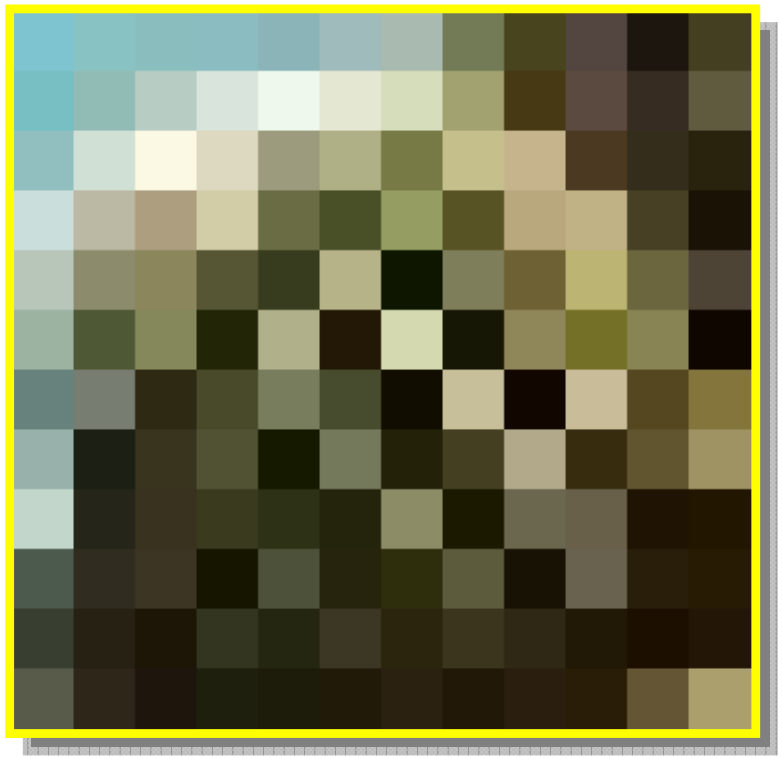
New image from new pixels

Resampling Through Bilinear Interpolation

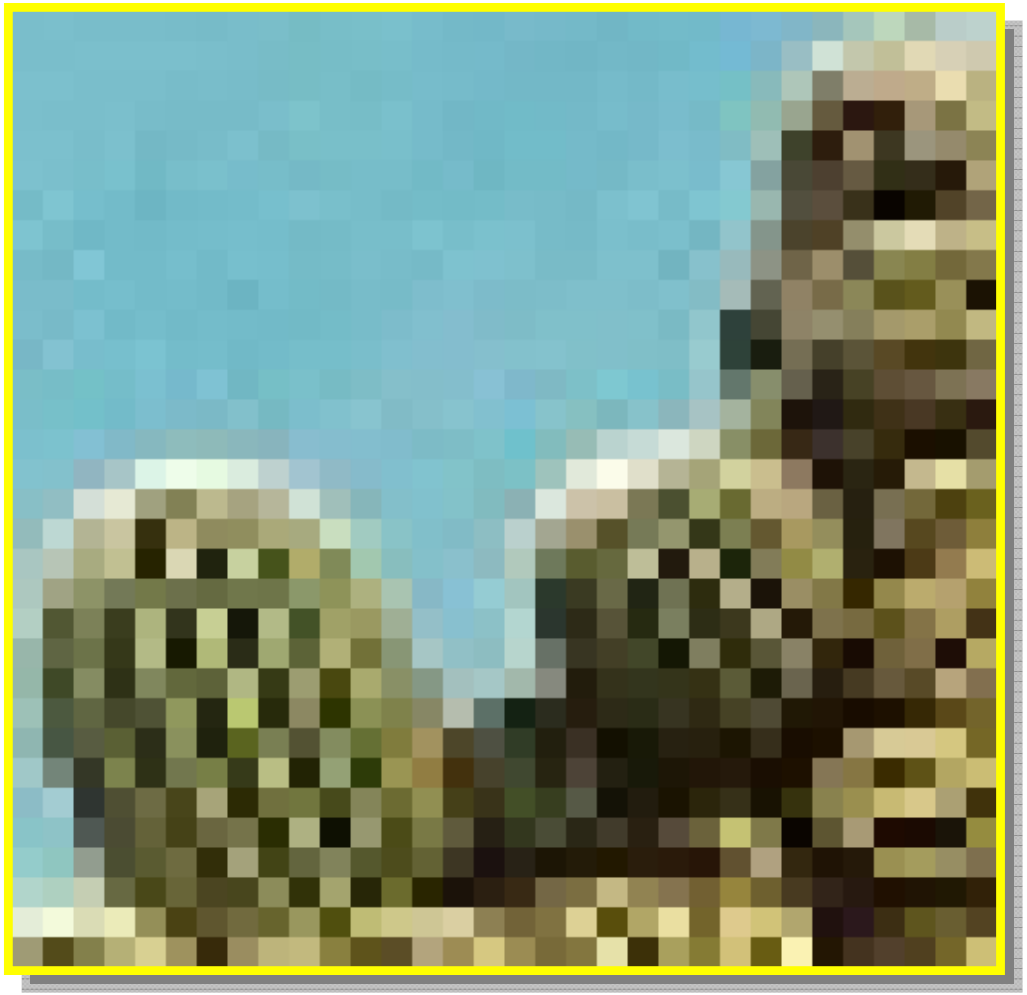


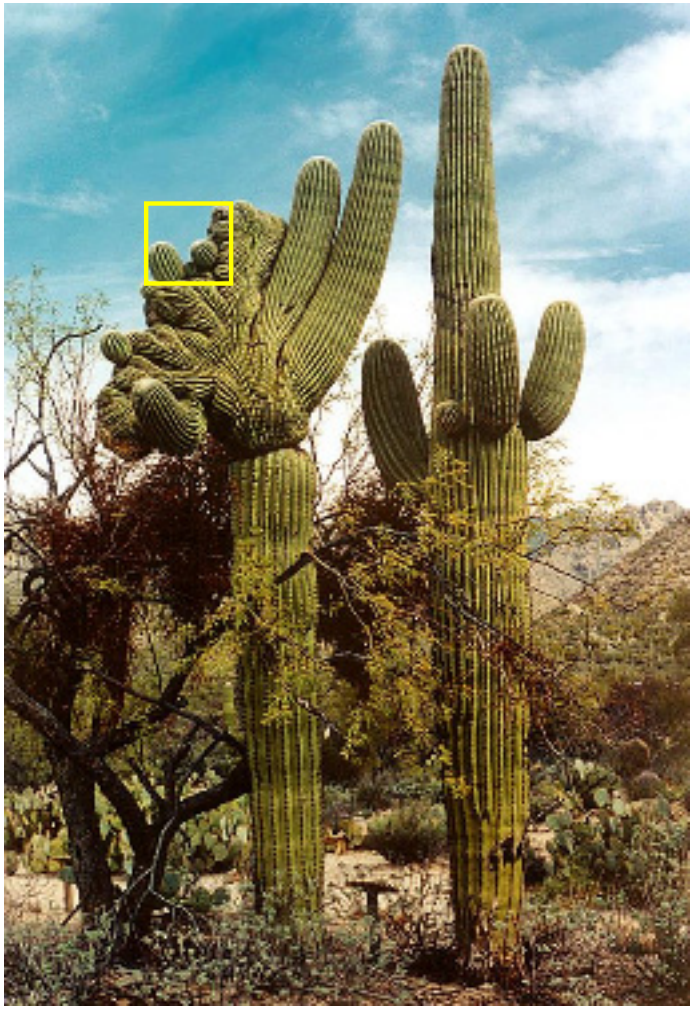
Resampling Through Bilinear Interpolation





Resampling Through Bilinear Interpolation





Resampling Through Bilinear Interpolation

Let I be an $R' \times C'$ image.
We want to resize I to $R \times C$.
Call the new image J.

Let $s_R = R'/R$ and $s_C = C'/C$.

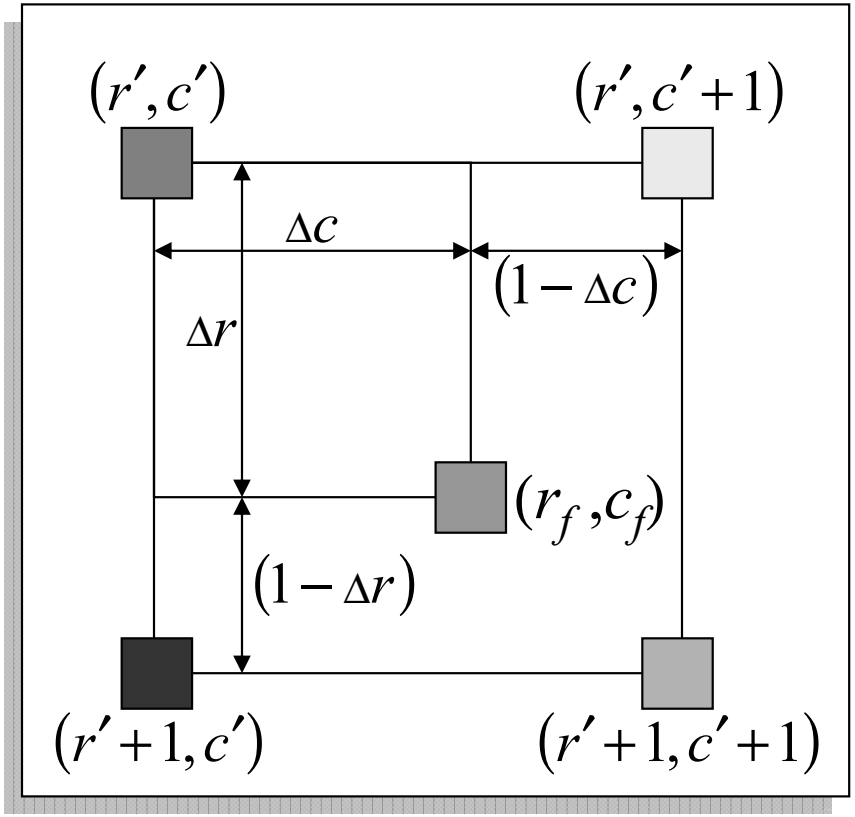
Let $r_f = r \cdot s_R$ for $r = 1, \ldots, R$

and $c_f = c \cdot s_C$ for $c = 1, \ldots, C$.

Let $r' = \lfloor r_f \rfloor$ and $c' = \lfloor c_f \rfloor$.

Let $\Delta r = r_f - r'$ and $\Delta c = c_f - c'$.

$$\begin{aligned} \text{Then} \quad J(r,c) = {} & I(r',c') \cdot (1-\Delta r) \cdot (1-\Delta c) \\ & + I(r'+1,c') \cdot \Delta r \cdot (1-\Delta c) \\ & + I(r',c'+1) \cdot (1-\Delta r) \cdot \Delta c \\ & + I(r'+1,c'+1) \cdot \Delta r \cdot \Delta c. \end{aligned}$$



Resampling Through Bilinear In

For each pixel (r,c) in output image, J, compute the fractional location (r_f,c_f) in I. Use (r′,c′) the integer part of (r_f,c_f) to find the 4 neighboring locations in I. Compute J(r,c) from a weighted sum of I at each of the locations. The weights are computed from Δr and Δc.

Let I [illegible] ge.
We w[illegible] nt to resize I to [illegible] $\times C$.
Call t[illegible] e new image J.

Scale factors

Let $s_R = R'/R$ and $s_C = C'/C$.

Let $r_f = r \cdot s_R$ for $r = 1,\ldots,R$

and $c_f = c \cdot s_C$ for $c = 1,\ldots,C$.

Fractional location (r_f,c_f) in I of pixel (r,c) in J.

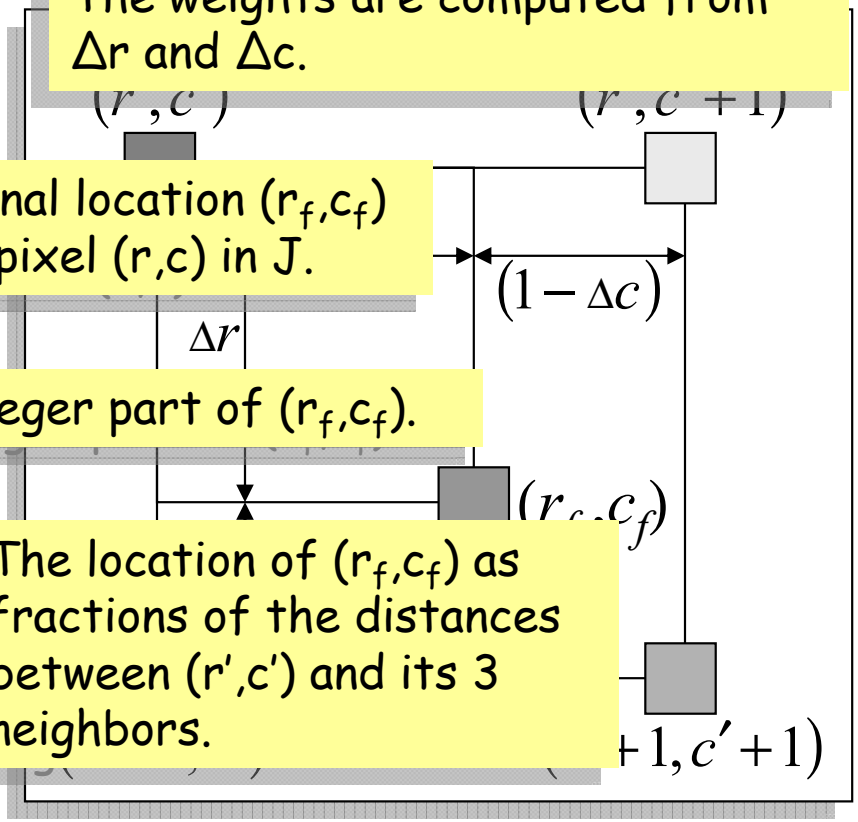
Let $r' = \lfloor r_f \rfloor$ and $c' = \lfloor c_f \rfloor$.

Integer part of (r_f,c_f).

Let $\Delta r = r_f - r'$ and $\Delta c = c_f - c'$.

The location of (r_f,c_f) as fractions of the distances between (r′,c′) and its 3 neighbors.

Then
$$J(r,c) = I(r',c') \cdot (1-\Delta r) \cdot (1-\Delta c) + I(r'+1,c') \cdot \Delta r \cdot (1-\Delta c) + I(r',c'+1) \cdot (1-\Delta r) \cdot \Delta c + I(r'+1,c'+1) \cdot \Delta r \cdot \Delta c.$$



Resampling Through Bilinear Interpolation

Size of original image: $R' \times C'$

Size of scaled image: $R \times C$

Row scale factor:

$$S_r = \begin{cases} R'/R, & \text{if} \quad R' > R, \\ (R'-1)/R, & \text{if} \quad R' < R. \end{cases}$$

Column scale factor:

$$S_c = \begin{cases} C'/C, & \text{if} \quad C' > C, \\ (C'-1)/C, & \text{if} \quad C' < C. \end{cases}$$

(r_f, c_f) is the fractional location in the input image from which to sample the output pixel (r, c).

$$r_f = [(1,\ldots,R)\cdot S_r], c_f = [(1,\ldots,C)\cdot S_c]$$

(r', c') are the row and column indices of the pixels in the input image to use in the algorithm.

$$(r', c') = (\lfloor r_f \rfloor, \lfloor c_f \rfloor)$$

$(\Delta r, \Delta c)$ are the fractional parts of the row and column locations, (r_f, c_f).

$$(\Delta r, \Delta c) = (r_f - r', c_f - c')$$

Then the value of each output pixel is given by

$$\begin{aligned} J(r,c) = {} & I(r',c')\cdot(1-\Delta r)\cdot(1-\Delta c) \\ & + I(r'+1,c')\cdot\Delta r\cdot(1-\Delta c) \\ & + I(r',c'+1)\cdot(1-\Delta r)\cdot\Delta c \\ & + I(r'+1,c'+1)\cdot\Delta r\cdot\Delta c. \end{aligned}$$

Resampling Through Bilinear Interpolation

Size of original image: $R' \times C'$

Like nearest neighbor resampling the 4x4 neighborhood in I of the point $(r_f,c_f,)$ has $(r',c') = \lfloor (r_f,c_f,) \rfloor$ as its upper left corner and has $(r'+1,c'+1)$ as its lower right corner.

Thus for each $(r_f,c_f,)$: (1) neither r' nor c' can be less than one and (2) $r'+1$ cannot be greater than R' and $c'+1$ cannot be greater than C'.

If the set of all indices $\{(r',c')\}$ do not satisfy (1) or (2), you must adjust the indices so that they do.

pixel (r, c).

$$r_f = [(1,\ldots,R)\cdot S_r], c_f = [(1,\ldots,C)\cdot S_c]$$

(r', c') are the row and column indices of the pixels in the input image to use in the algorithm.

$$(r', c') = (\lfloor r_f \rfloor, \lfloor c_f \rfloor)$$

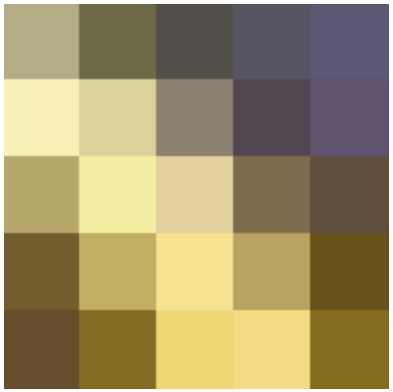
$(\Delta r, \Delta c)$ are the fractional parts of the row and column locations, (r_f, c_f).

$$(\Delta r, \Delta c) = (r_f - r', c_f - c')$$

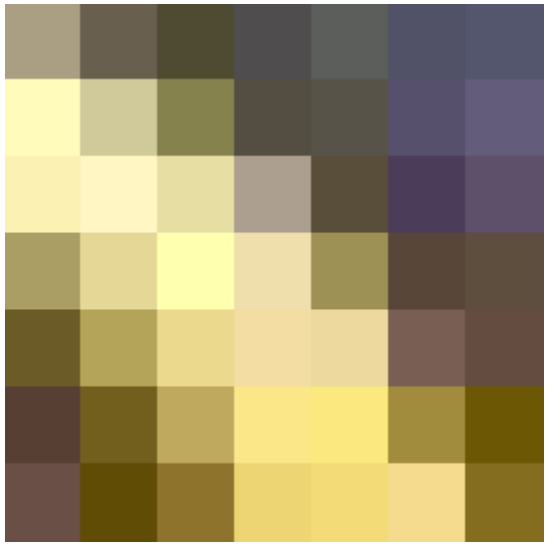
Then the value of each output pixel is given by

$$\begin{aligned} J(r,c) = {} & I(r',c')\cdot(1-\Delta r)\cdot(1-\Delta c) \\ & + I(r'+1,c')\cdot\Delta r\cdot(1-\Delta c) \\ & + I(r',c'+1)\cdot(1-\Delta r)\cdot\Delta c \\ & + I(r'+1,c'+1)\cdot\Delta r\cdot\Delta c. \end{aligned}$$

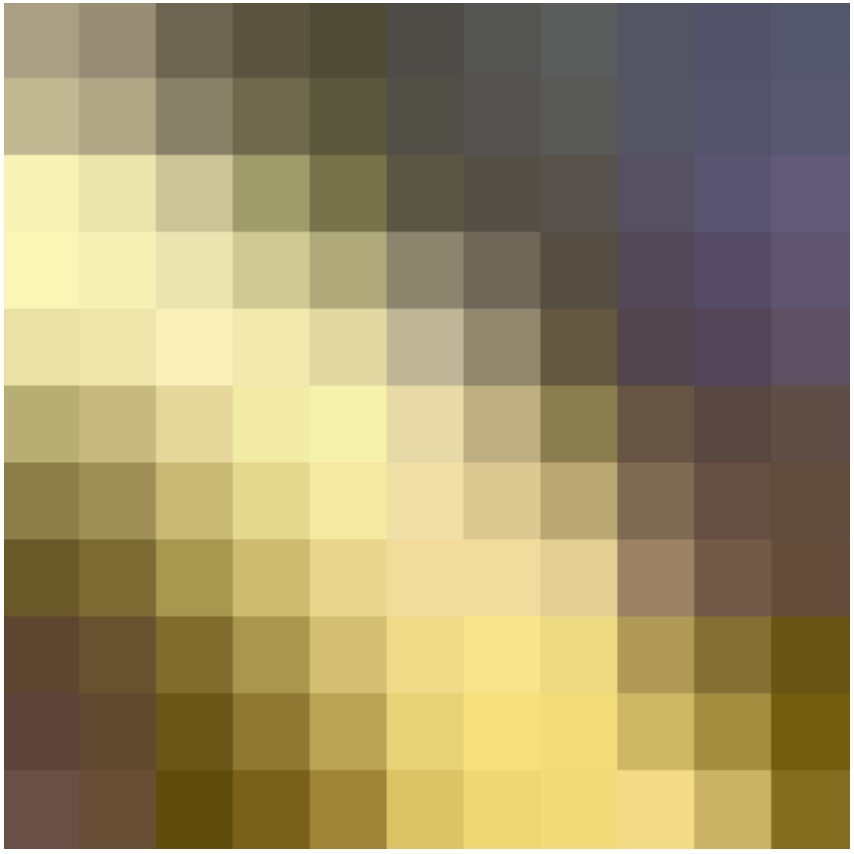
Bilinear Interpolation



5:7

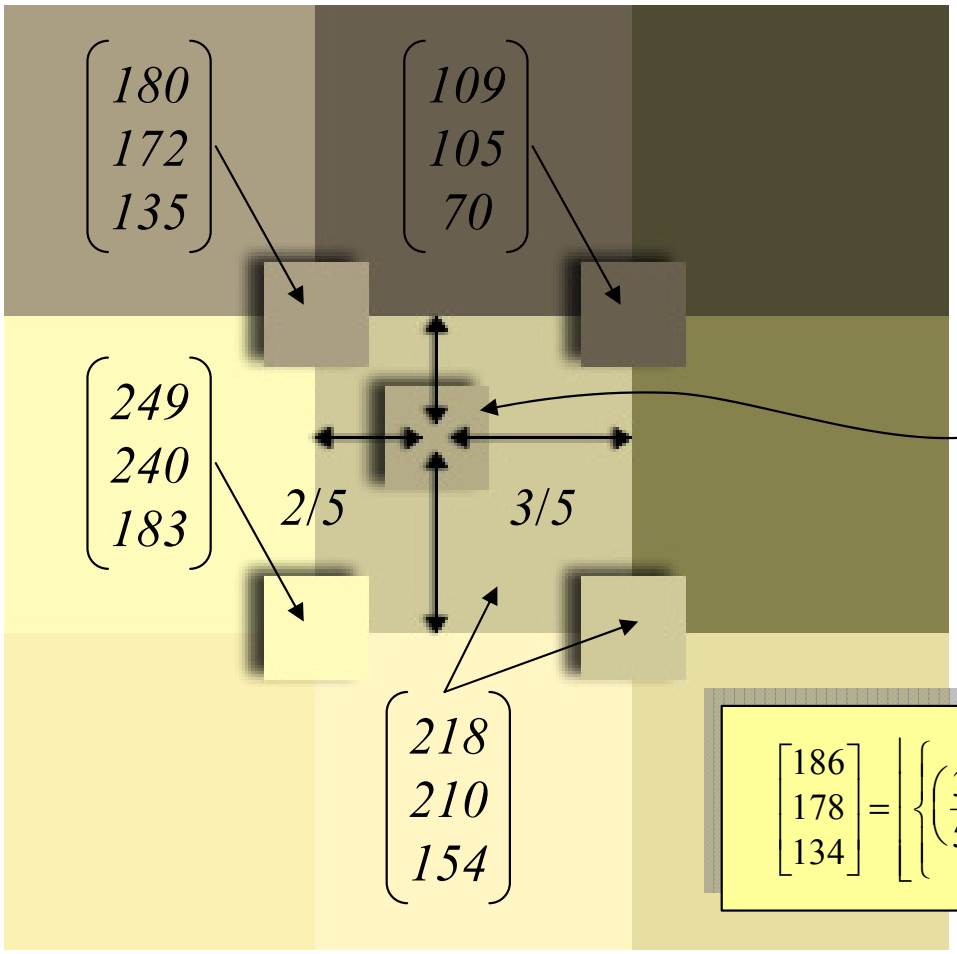


1:1



11:7

Bi-Interp Example: resize to 5/7 of original dimensions.



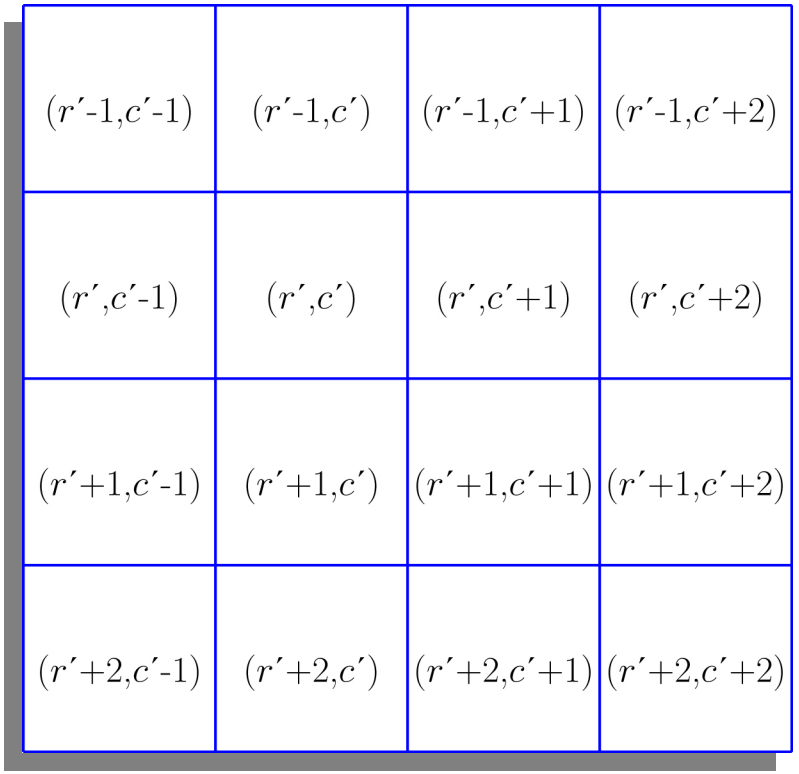
$$\begin{bmatrix}186\\178\\134\end{bmatrix} = \left\lfloor \left\{ \left(\frac{3}{5}\right)^2 \begin{bmatrix}180\\172\\135\end{bmatrix} + \left(\frac{2}{5}\right)\left(\frac{3}{5}\right)\begin{bmatrix}249\\240\\183\end{bmatrix} + \left(\frac{3}{5}\right)\left(\frac{2}{5}\right)\begin{bmatrix}109\\105\\70\end{bmatrix} + \left(\frac{2}{5}\right)^2 \begin{bmatrix}218\\210\\154\end{bmatrix} \right\} + 0.5 \right\rfloor$$

Resampling Through Bicubic Interpolation

Bilinear interpolation computes a value for J(r,c) as the weighted combination of I(r',c'), I(r'+1,c'), I(r'+1,c'+1), and I(r',c'+1).

Bicubic interpolation uses not only those 4 input image pixels, but also their partial derivatives:

$$\left\{\left\{\frac{\partial}{\partial c'}\mathrm{I}(r'+i,c'+j),\right.\right.$$
$$\left.\frac{\partial}{\partial r'}\mathrm{I}(r'+i,c'+j),\right.$$
$$\left.\left.\frac{\partial}{\partial c'r'}\mathrm{I}(r'+i,c'+j)\right\}_{j=0}^{1}\right\}_{i=0}^{1}$$

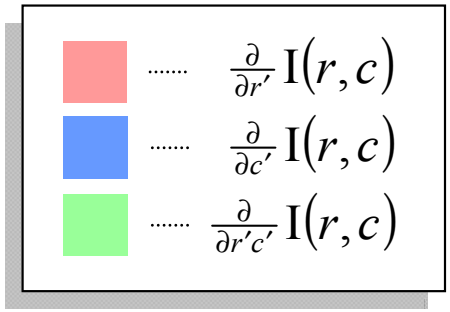


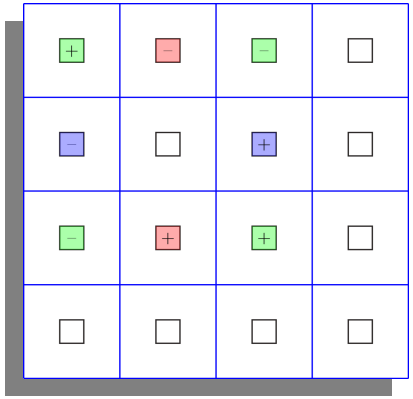
Since the derivatives are are computed digitally on the 8 neighborhoods of the 4 pixel locations, a 4×4 neighborhood of the input image is needed for each output value.

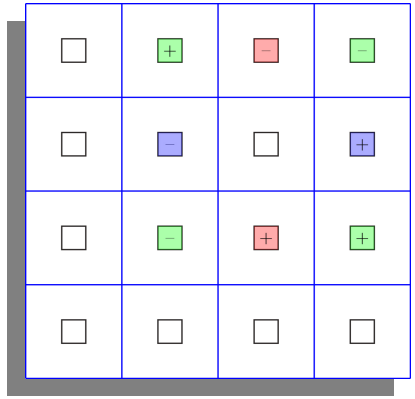
Resampling Through Bicubic Interpolation

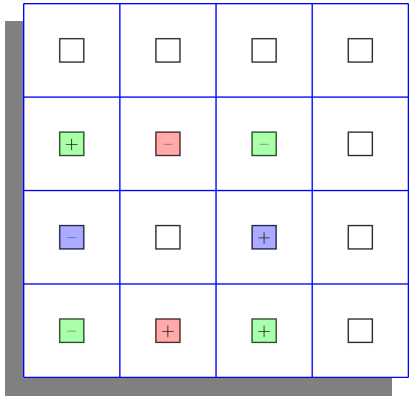
$\mathcal{N}(r',c')$ is the 8-pixel neighborhood of (r',c').

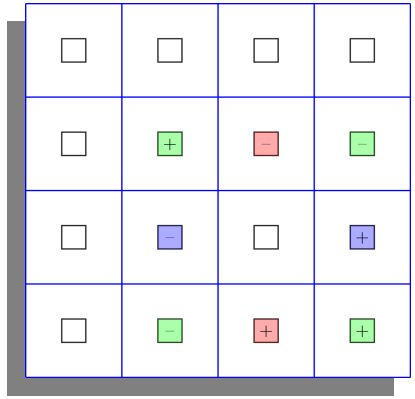
$$\frac{\partial}{\partial r'} I(r',c') = \text{avg. of forward diff and backward diff @ } (r',c')$$
$$= \tfrac{1}{2}\left[\left(I(r'+1,c') - I(r',c')\right) + \left(I(r',c') - I(r'-1,c')\right)\right]$$
$$= \tfrac{1}{2}\left[I(r'+1,c') - I(r'-1,c')\right]$$











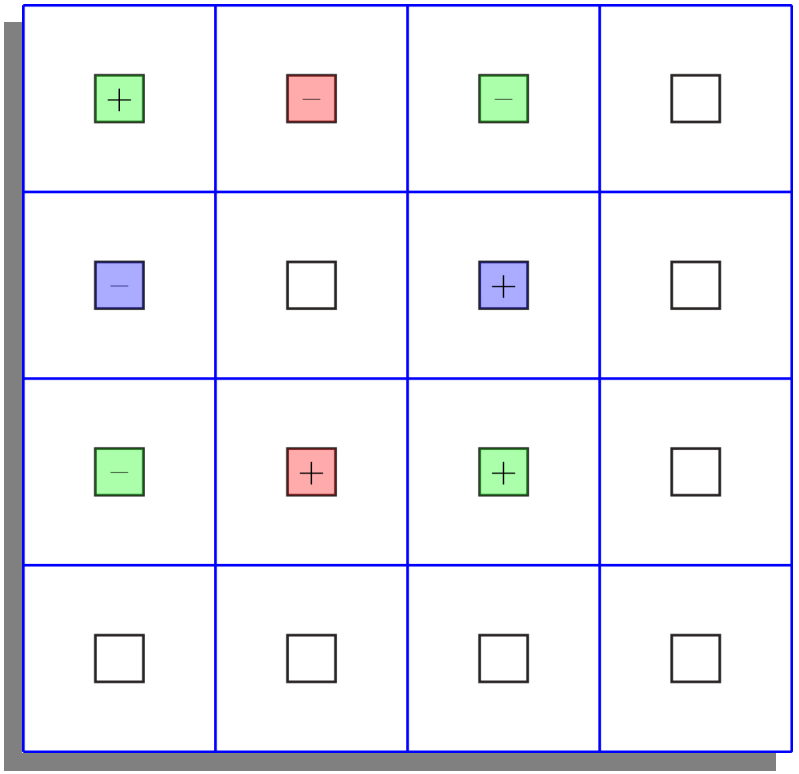
Resampling Through Bicubic Interpolation

$$\mathcal{N}(r', c')$$

$$\frac{\partial}{\partial r'} \mathrm{I}(r', c') = \frac{1}{2}\left[\mathrm{I}(r'+1, c') - \mathrm{I}(r'-1, c')\right],$$

$$\frac{\partial}{\partial c'} \mathrm{I}(r', c') = \frac{1}{2}\left[\mathrm{I}(r', c'+1) - \mathrm{I}(r', c'-1)\right],$$

$$\frac{\partial}{\partial c' r'} \mathrm{I}(r', c') = \frac{1}{4}\left[\mathrm{I}(r'+1, c'+1) - \mathrm{I}(r'+1, c'-1) + \mathrm{I}(r'-1, c'-1) - \mathrm{I}(r'-1, c'+1)\right]$$



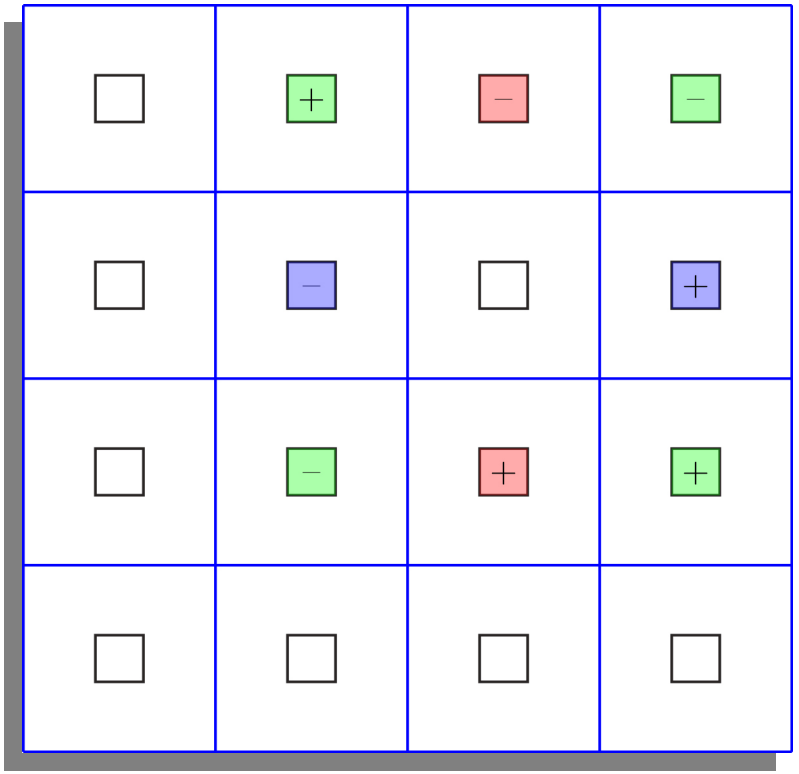
Resampling Through Bicubic Interpolation

$$\mathcal{N}(r', c'+1)$$

$$\frac{\partial}{\partial r'} \mathrm{I}(r', c'+1) = \frac{1}{2}\left[\mathrm{I}(r'+1, c'+1) - \mathrm{I}(r'-1, c'+1)\right],$$

$$\frac{\partial}{\partial c'} \mathrm{I}(r', c'+1) = \frac{1}{2}\left[\mathrm{I}(r', c'+2) - \mathrm{I}(r', c')\right],$$

$$\frac{\partial}{\partial c' r'} \mathrm{I}(r', c'+1) = \frac{1}{4}\left[\mathrm{I}(r'+1, c'+2) - \mathrm{I}(r'+1, c') + \mathrm{I}(r'-1, c') - \mathrm{I}(r'-1, c'+2)\right]$$



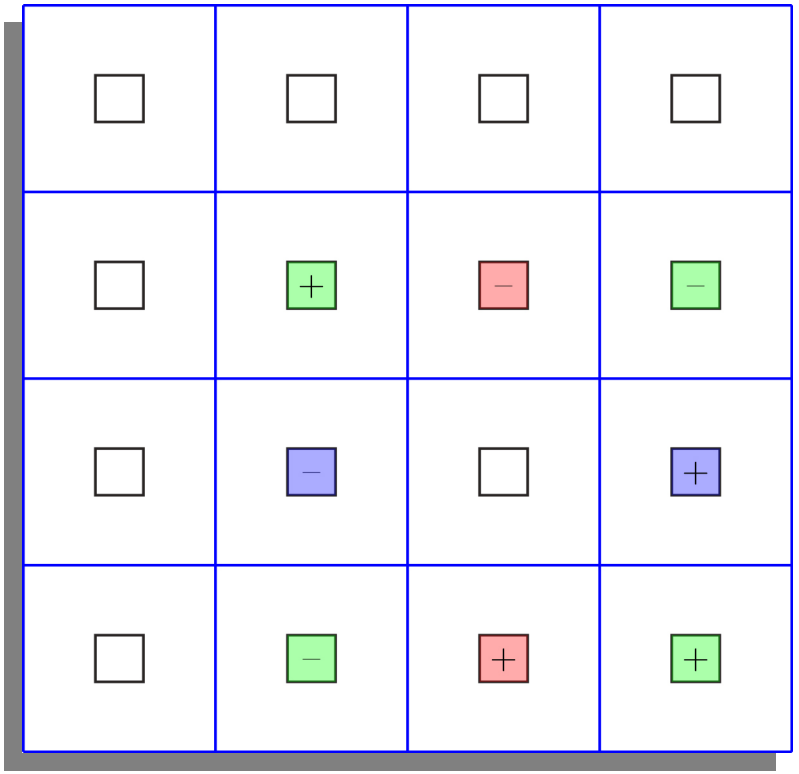
Resampling Through Bicubic Interpolation

$$\mathcal{N}(r'+1, c'+1)$$

$$\frac{\partial}{\partial r'} \mathrm{I}(r'+1, c'+1) = \frac{1}{2}\left[\mathrm{I}(r'+2, c'+1) - \mathrm{I}(r', c'+1)\right],$$

$$\frac{\partial}{\partial c'} \mathrm{I}(r'+1, c'+1) = \frac{1}{2}\left[\mathrm{I}(r'+1, c'+2) - \mathrm{I}(r'+1, c')\right],$$

$$\frac{\partial}{\partial c' r'} \mathrm{I}(r'+1, c'+1) = \frac{1}{4}\left[\mathrm{I}(r'+2, c'+2) - \mathrm{I}(r'+2, c') + \mathrm{I}(r', c') - \mathrm{I}(r', c'+2)\right]$$



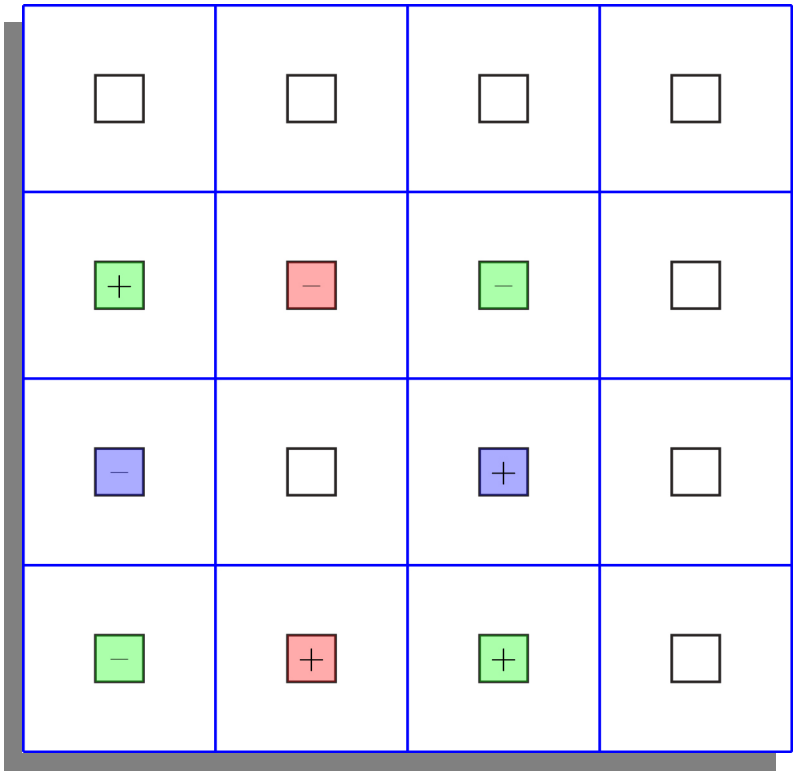
Resampling Through Bicubic Interpolation

$$\mathcal{N}(r'+1, c')$$

$$\frac{\partial}{\partial r'} \mathrm{I}(r'+1, c') = \frac{1}{2}\left[\mathrm{I}(r'+2, c') - \mathrm{I}(r', c')\right],$$

$$\frac{\partial}{\partial c'} \mathrm{I}(r'+1, c') = \frac{1}{2}\left[\mathrm{I}(r'+1, c'+1) - \mathrm{I}(r'+1, c'-1)\right],$$

$$\frac{\partial}{\partial c' r'} \mathrm{I}(r'+1, c') = \frac{1}{4}\left[\mathrm{I}(r'+2, c'+1) - \mathrm{I}(r'+2, c'-1) + \mathrm{I}(r', c'-1) - \mathrm{I}(r', c'+1)\right]$$



Resampling Through Bicubic Interpolation

$$S_r = \begin{cases} R'/R, & \text{if} \quad R' > R, \\ (R'-1)/R, & \text{if} \quad R' < R. \end{cases}$$

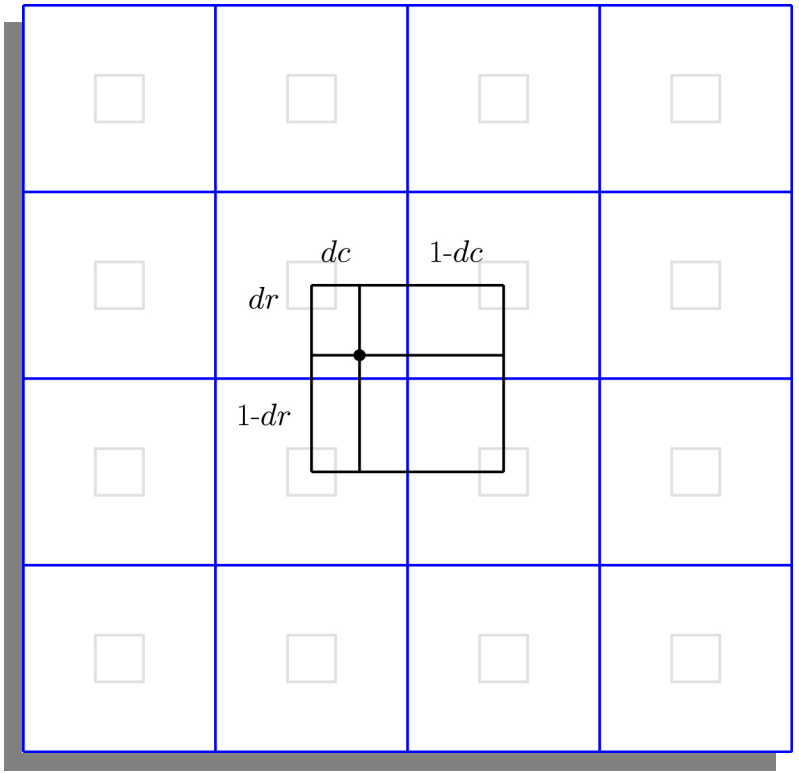
$$S_c = \begin{cases} C'/C, & \text{if} \quad C' > C, \\ (C'-1)/C, & \text{if} \quad C' < C. \end{cases}$$

$$r_f = [(1,\ldots,R)\cdot S_r], c_f = [(1,\ldots,C)\cdot S_c]$$

$$(r', c') = (\lfloor r_f \rfloor, \lfloor c_f \rfloor)$$

$$(dr, dc) = (r_f - r', c_f - c')$$

This part is the same as NN interpolation.



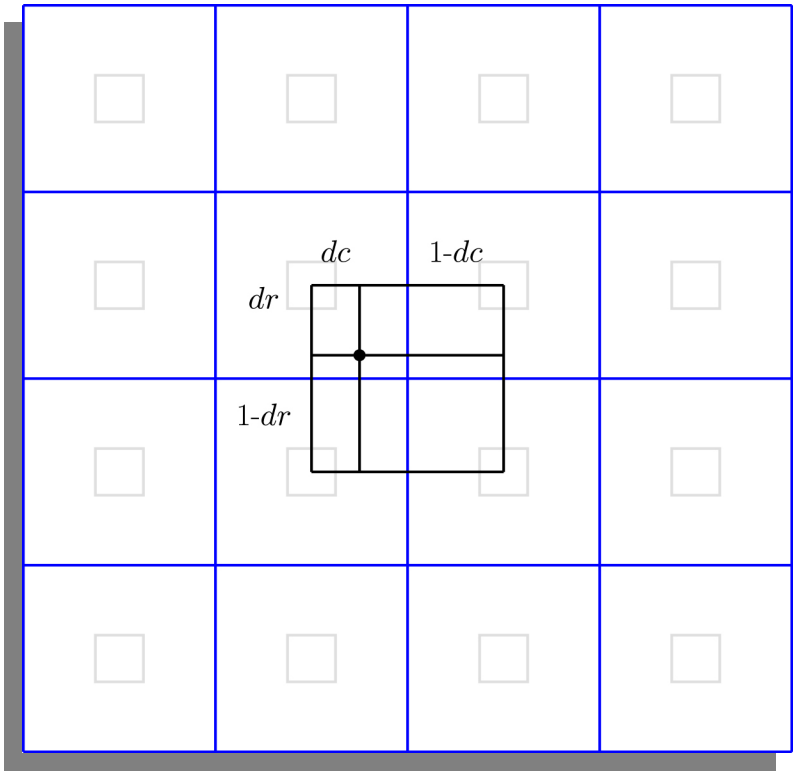
Resampling Through Bicubic Interpolation

$$J(r,c) = \sum_{m=-1}^{2} \sum_{n=-1}^{2} I(r'+m, c'+n) P(dr-m) P(n-dc)$$

$$P(x) = \tfrac{1}{6}\left[Q(x+2)^3 - 4Q(x+1)^3 - 6Q(x)^3 - 4Q(x-1)^3\right]$$

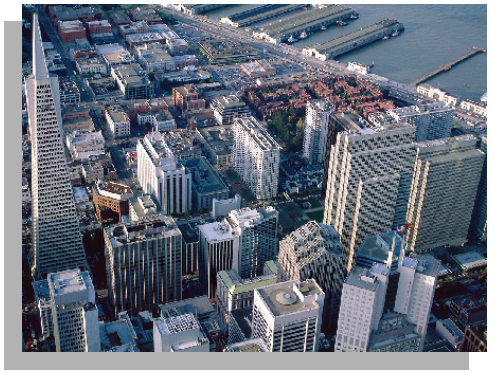
$$Q(x) = \begin{cases} x & \text{for } x > 0 \\ 0 & \text{for } x \le 0 \end{cases}$$

The differentials on pp 101-105 are combined in this sum of products of polynomials.



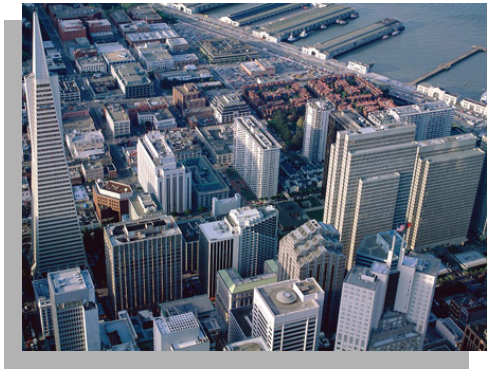
Size Reduction 3/7

original



nearest neighbor

bilinear



bicubic

Size Reduction 3/7

original

Size Reduction 3/7 (zoomed)

nearest neighbor

Size Reduction 3/7 (zoomed)

bilinear

Size Reduction 3/7 (zoomed)

bicubic

Size Reduction 3/7

original

Enlargement

nearest neighbor
7/3

Enlargement

bilinear 7/3

Enlargement

bicubic 7/3