EECE 4353 Image Processing

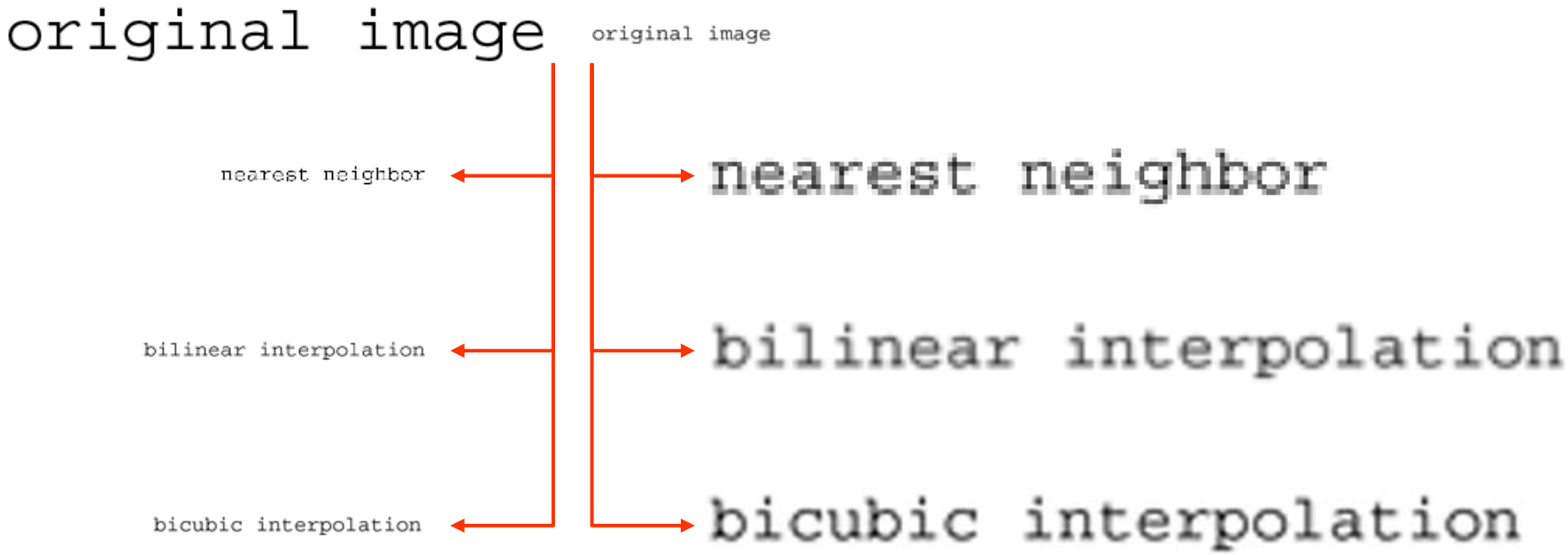
Lecture Notes: Resizing Images

Richard Alan Peters II

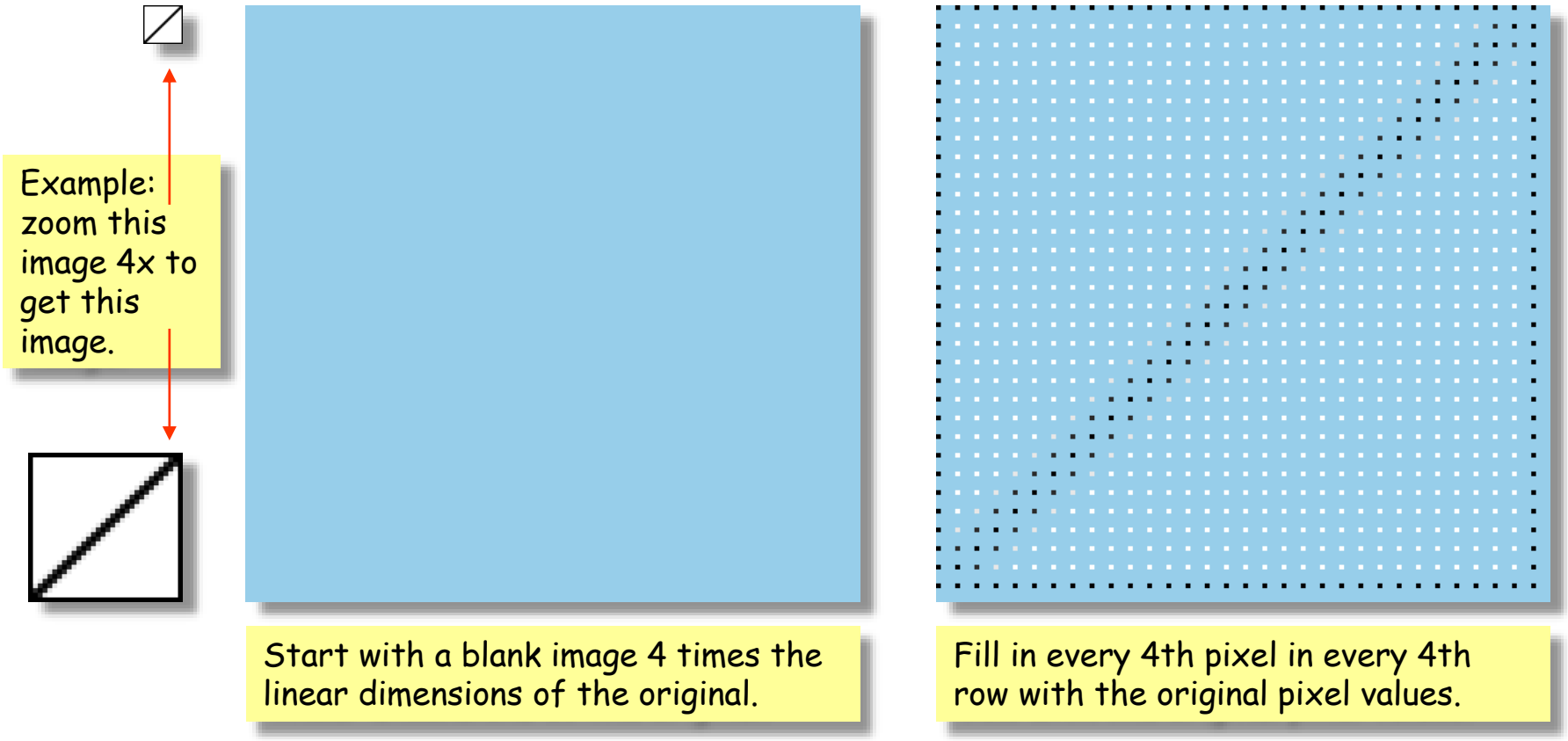
Department of Electrical Engineering and
Computer Science

Fall Semester 2016

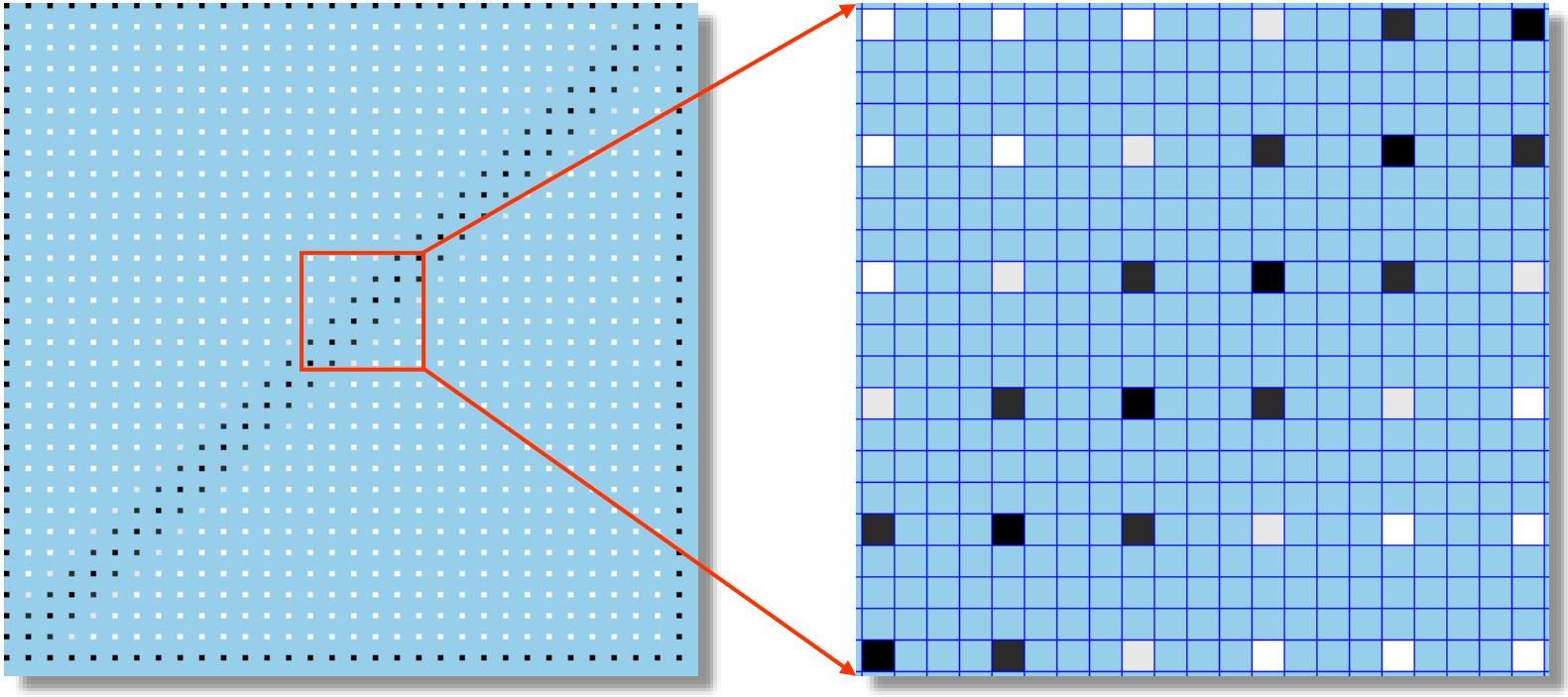
Three Methods for Resizing Images



Enlarging Images Through Pixel Replication

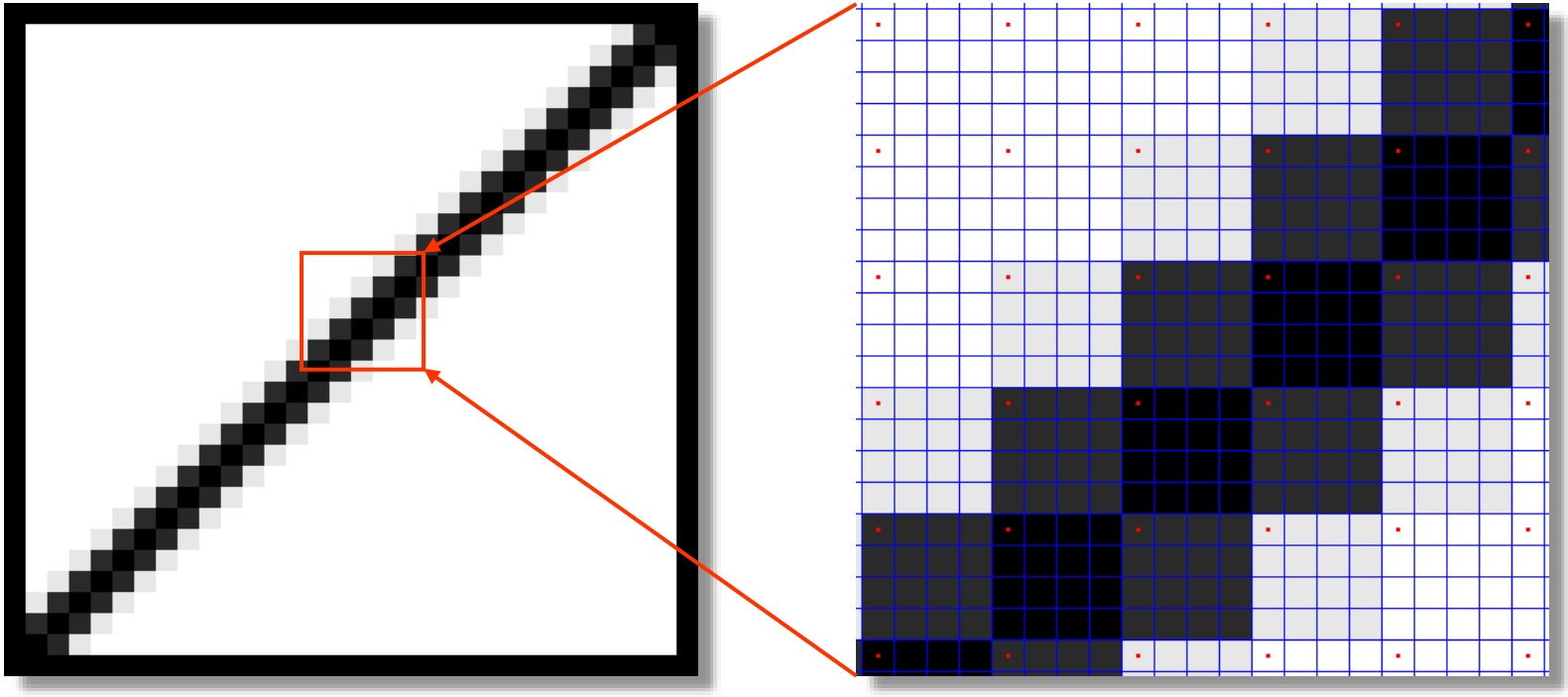


Enlarging Images Through Pixel Replication



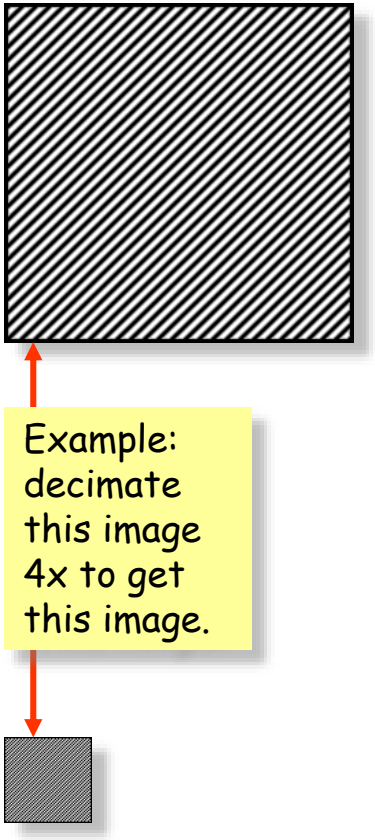
Detail showing every 4th pixel in every 4th row with the original pixel values.

Enlarging Images Through Pixel Replication



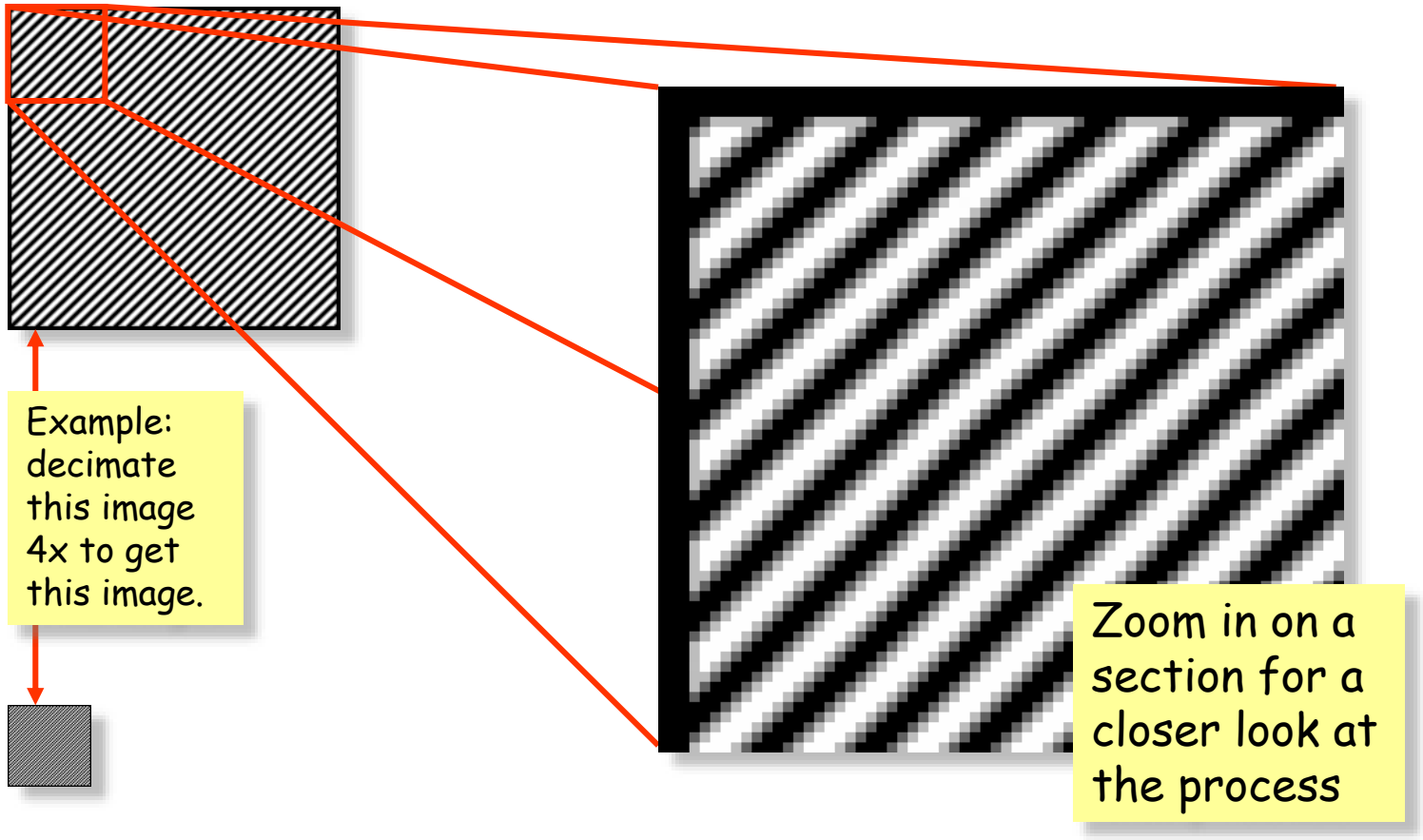
For each original value: replicate it 15 times to create a new, larger "pixel".

Reducing Images Through Pixel Decimation

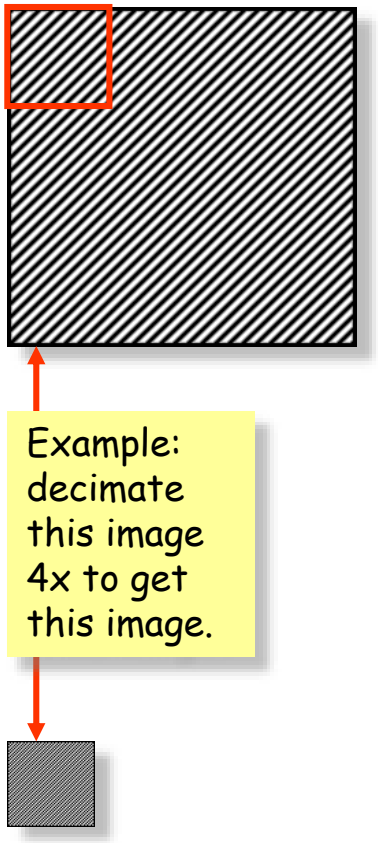


Decimation by a factor of *n*: take every *n*th pixel in every *n*th row

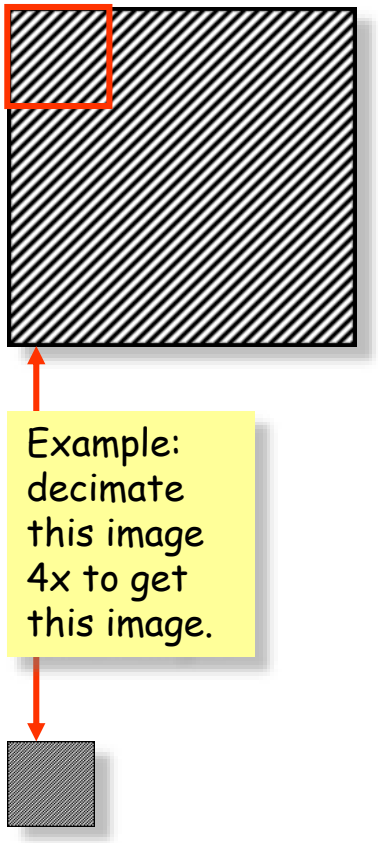
Reducing Images Through Pixel Decimation

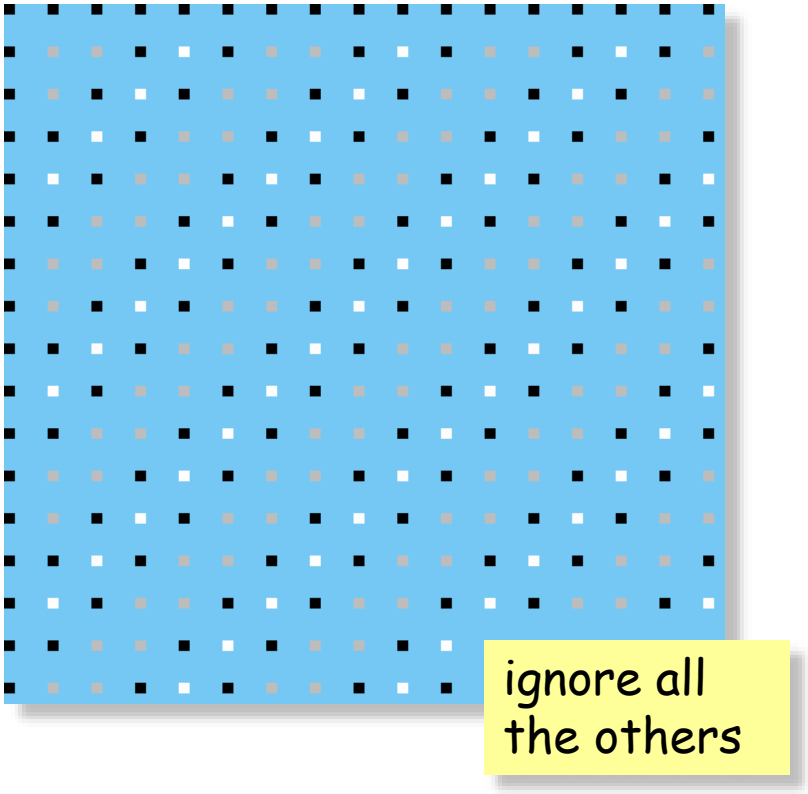


Reducing Images Through Pixel Decimation

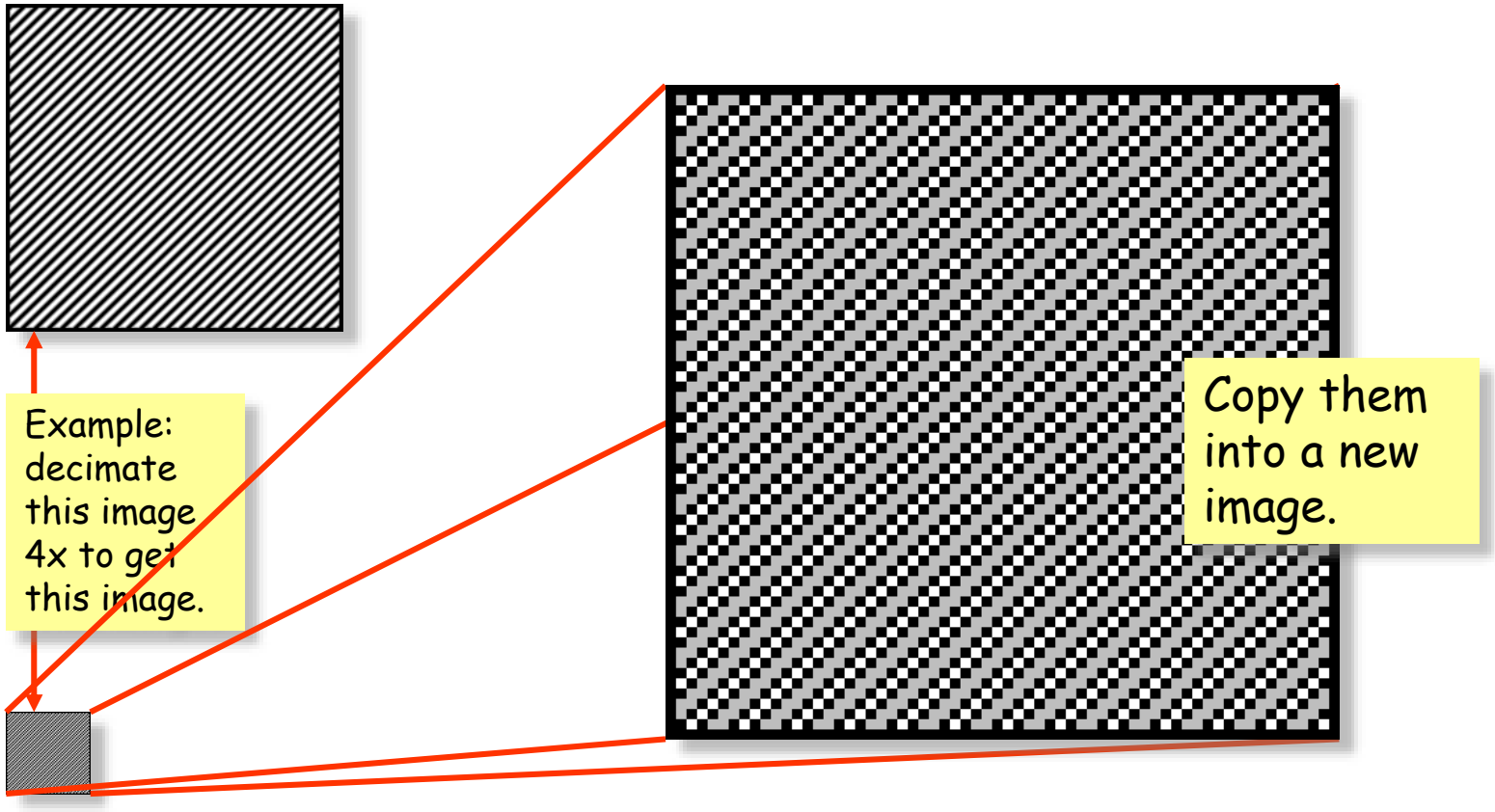


Reducing Images Through Pixel Decimation

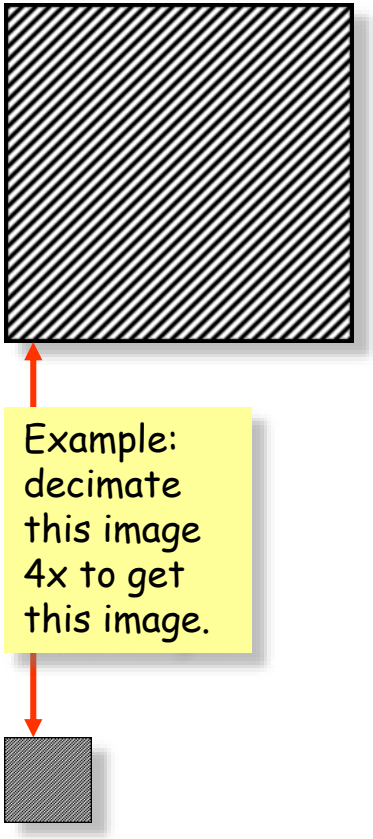


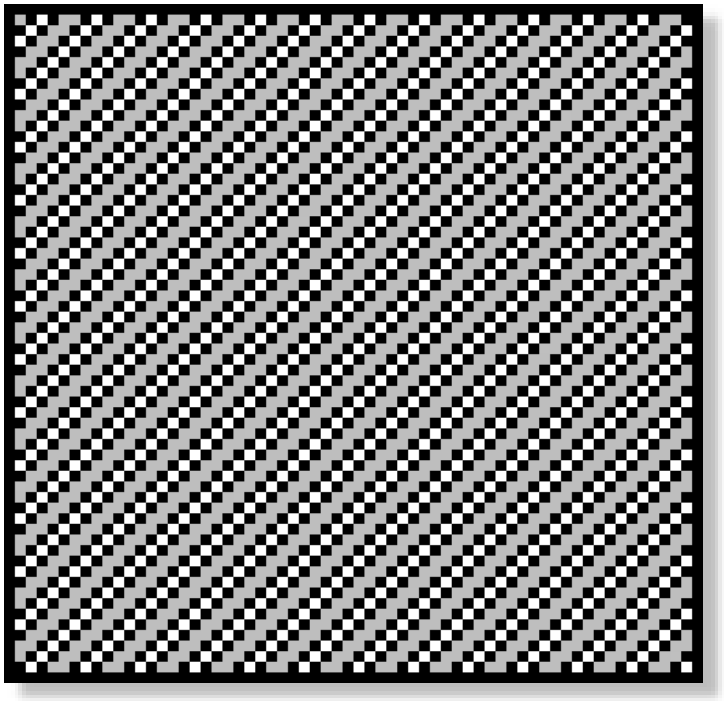


Reducing Images Through Pixel Decimation



Reducing Images Through Pixel Decimation





Backward Mapping

Resampling involves creating one image from another, each with a different spatial geometry. There are two strategies for doing this. Let **I** be the $R \times C$ input image and let **J** be the $R' \times C'$ output image.

1. For each pixel (r,c) in image **I** select a pixel (r',c') in **J** such that

$$\mathbf{J}(r',c') = \Phi\{\mathbf{I}(r,c);(r,c)\} \text{ for } r \in \{0,\ldots,R-1\} \text{ and } c \in \{0,\ldots,C-1\}.$$

2. For each pixel (r',c') in **J** select a pixel (r,c) in image **I** such that

$$\mathbf{J}(r',c') = \Phi^{-1}\{\mathbf{I}(r,c);(r',c')\} \text{ for } r' \in \{0,\ldots,R'-1\} \text{ and } c' \in \{0,\ldots,C'-1\}.$$

The first strategy, a forward mapping, can leave pixels in **J** empty – a problem. The second fills in every pixel in **J** by selecting a correct value from **I**. The second approach is a better strategy.

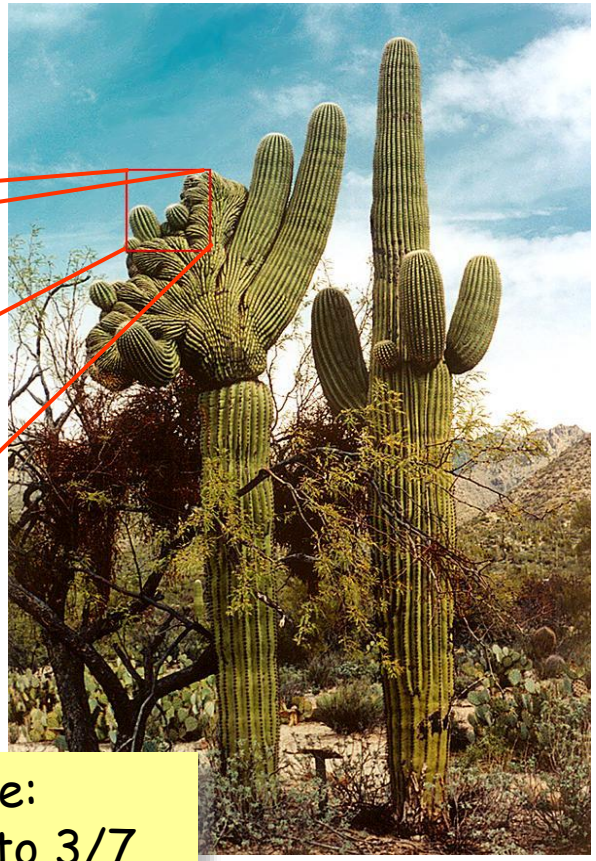
Nearest Neighbor Resampling

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.)

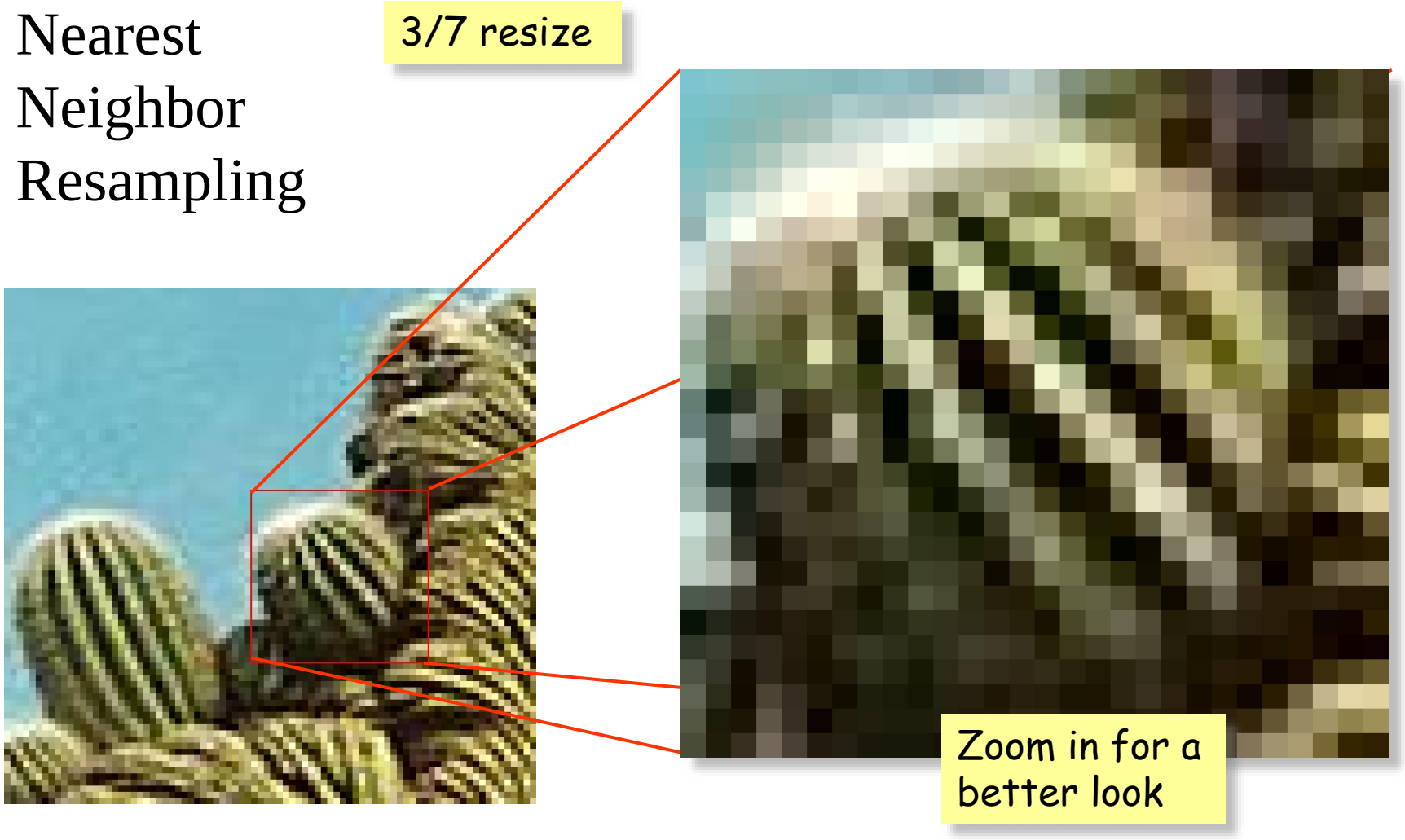
Nearest Neighbor Resampling

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



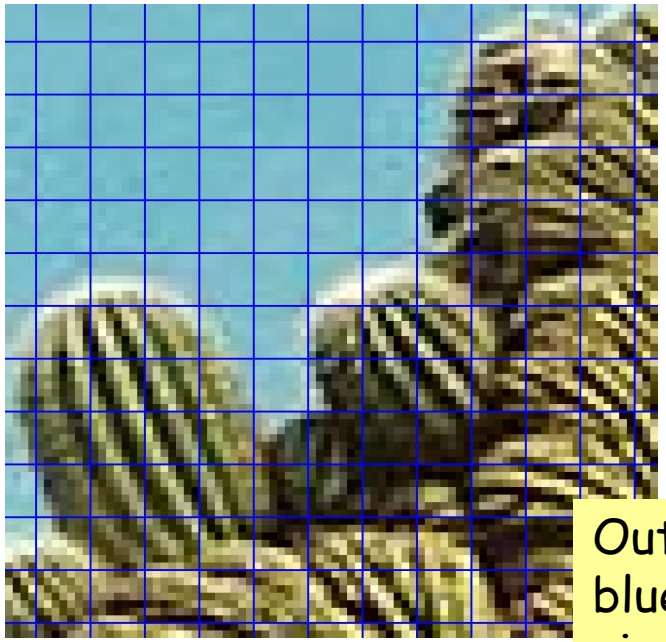
Example: resize to 3/7 of the original

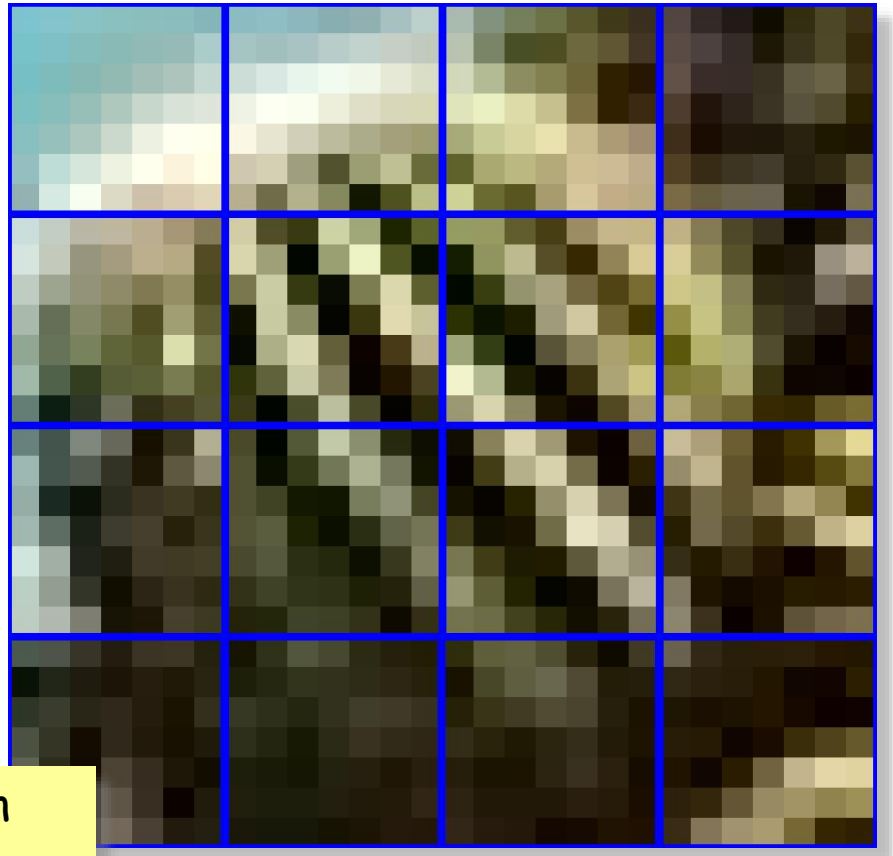
Nearest Neighbor Resampling



Nearest Neighbor Resampling

3/7 resize

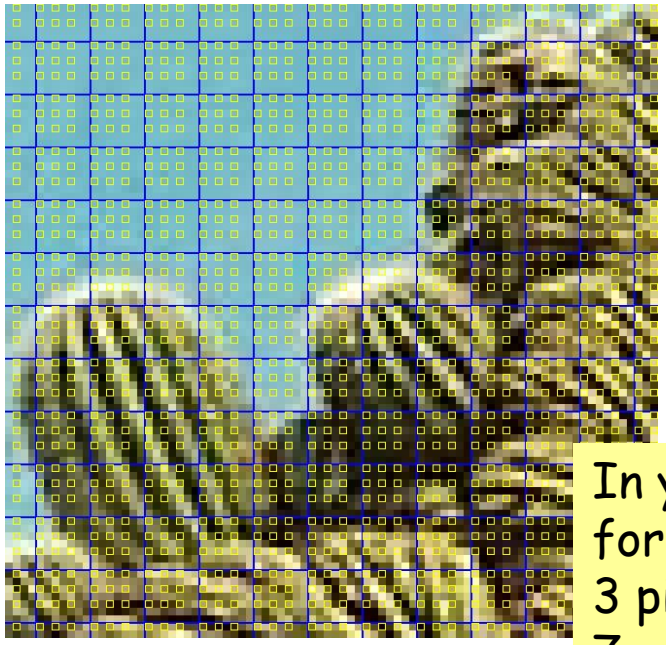


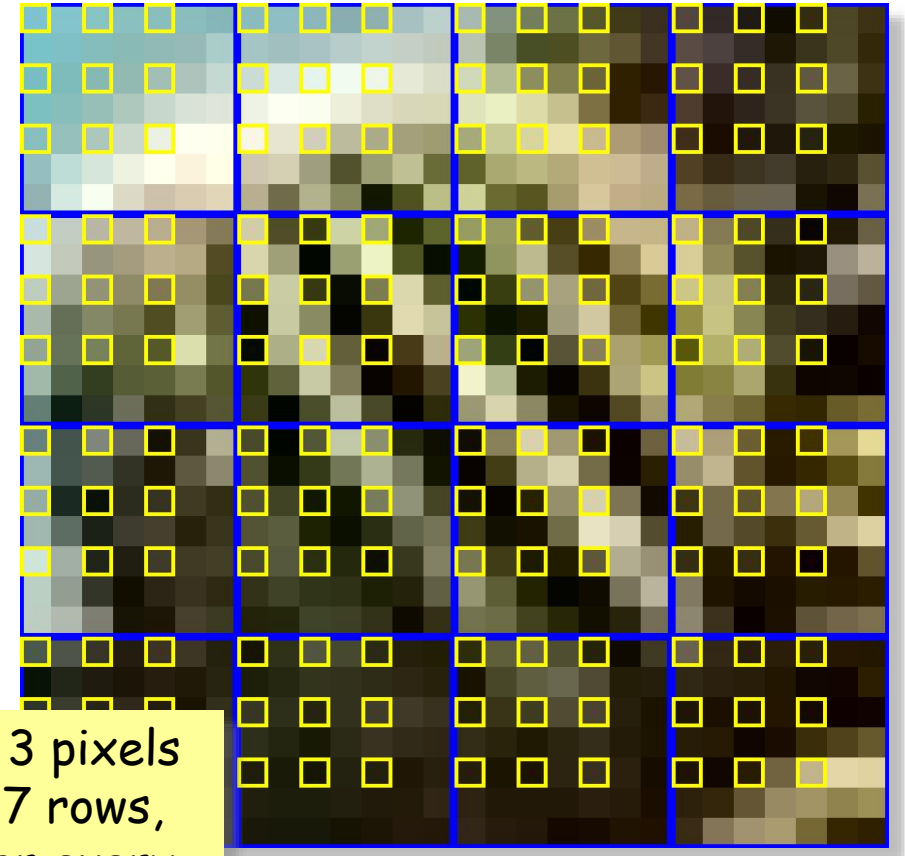


Outlined in blue: 7x7 pixel squares

Nearest Neighbor Resampling

3/7 resize

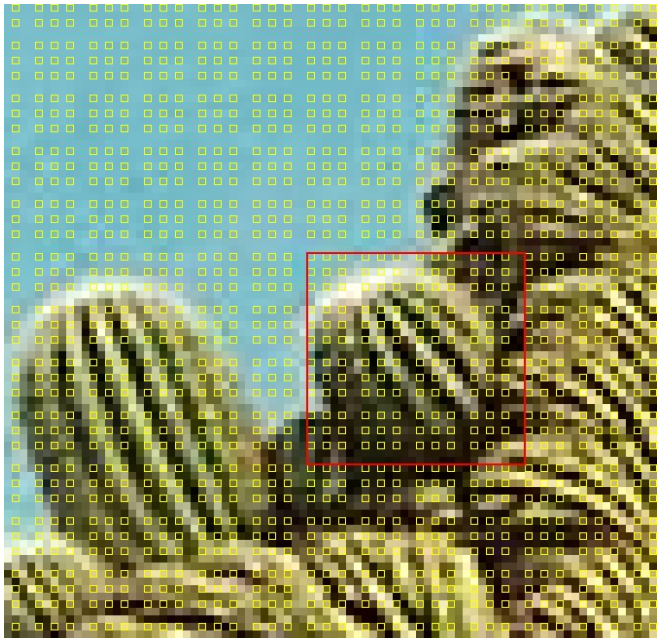




In yellow: 3 pixels for every 7 rows, 3 pixels for every 7 cols.

Nearest Neighbor Resampling

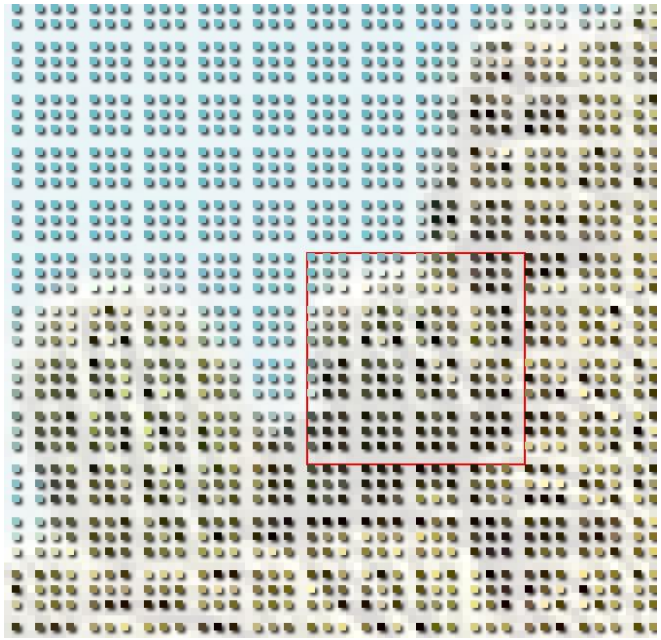
3/7 resize

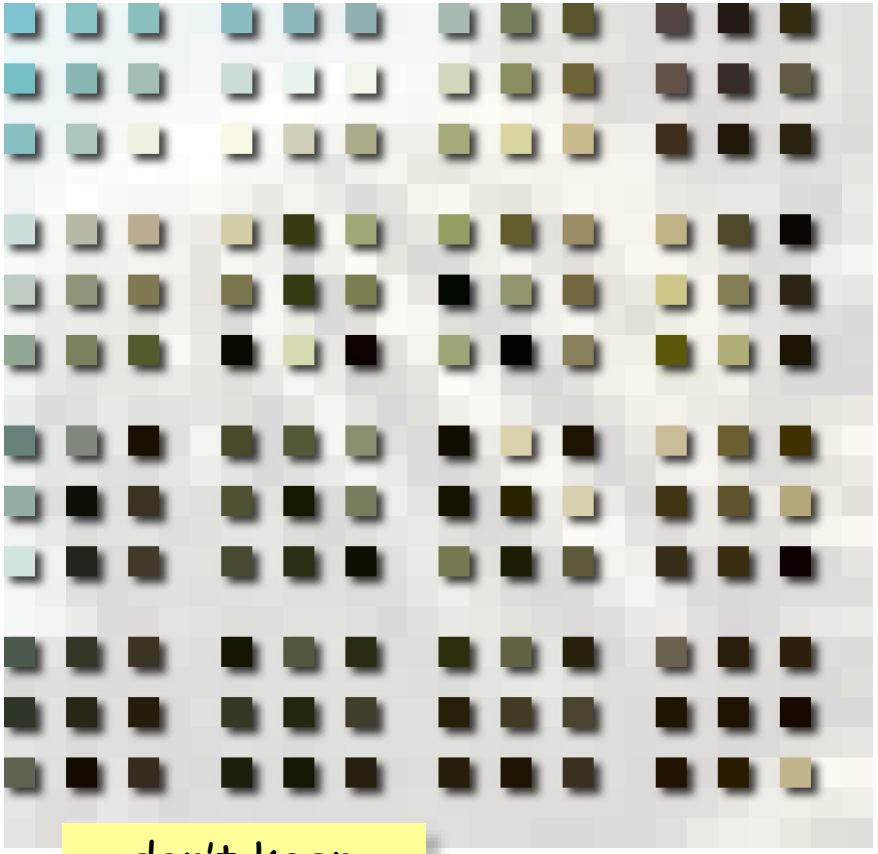


Keep the highlighted pixels…

Nearest Neighbor Resampling

3/7 resize

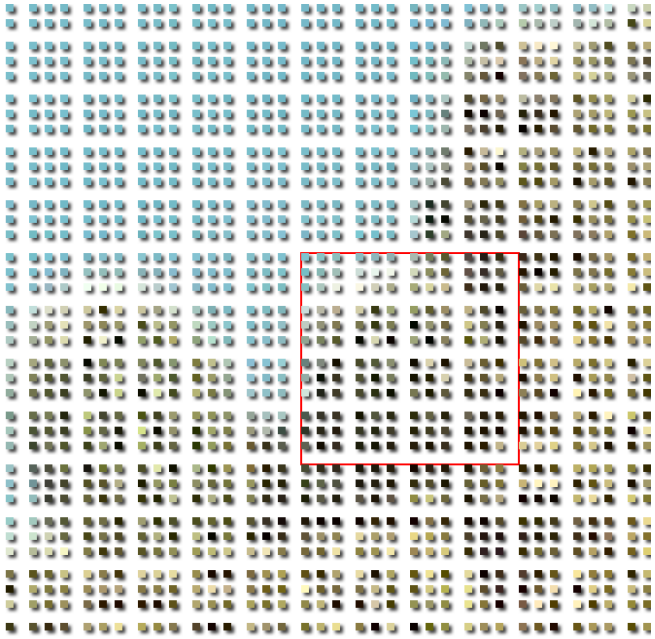


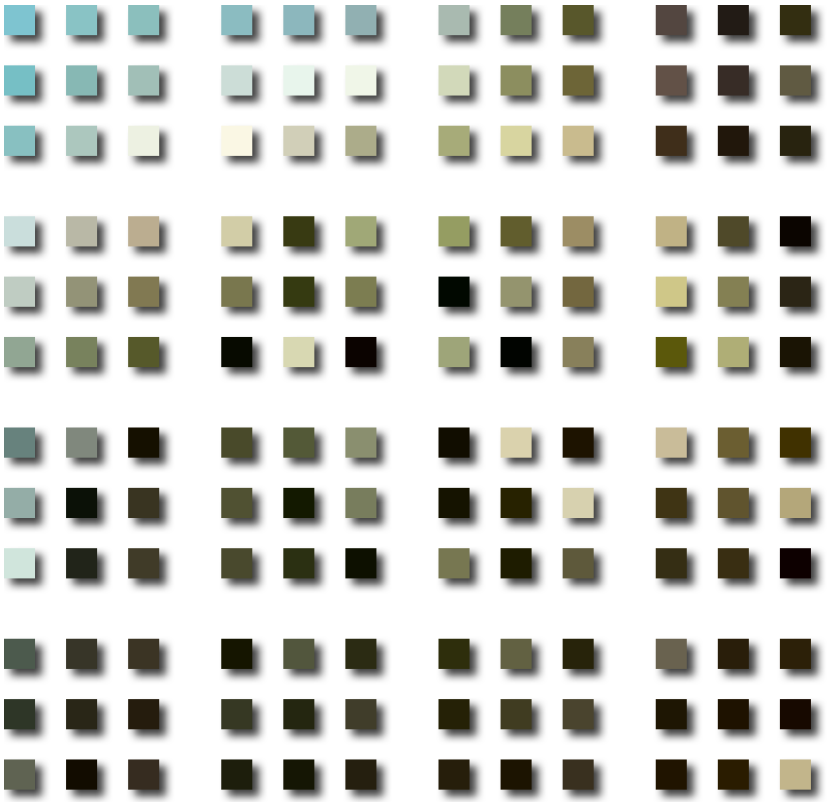


… don't keep the others.

Nearest Neighbor Resampling

3/7 resize

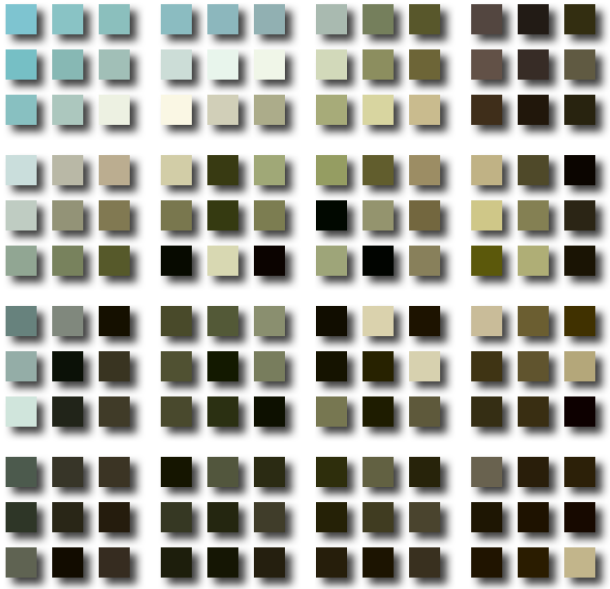




Copy them into a new image.

Nearest Neighbor Resampling

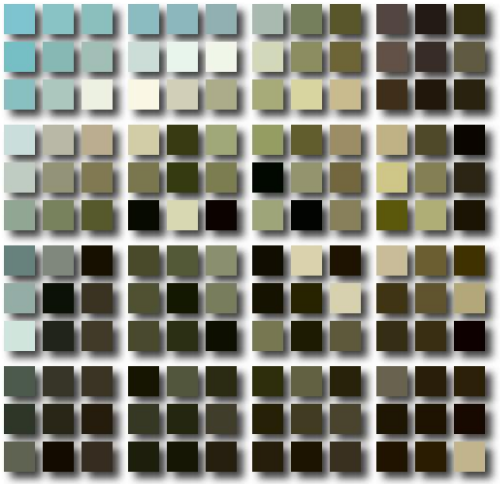
3/7 resize



Copy them into a new image.

Nearest Neighbor Resampling

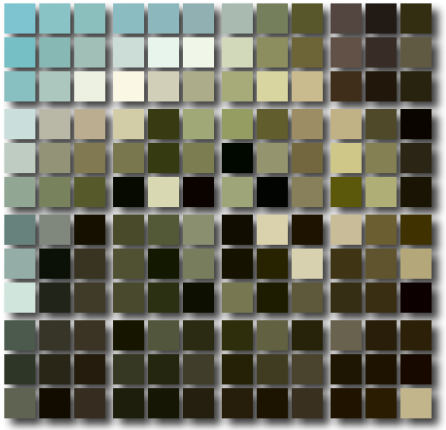
3/7 resize



Copy them into a new image.

Nearest Neighbor Resampling

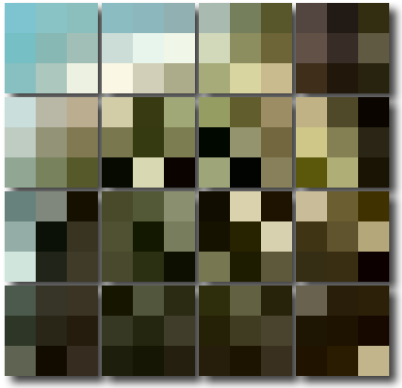
3/7 resize



Copy them into a new image.

Nearest Neighbor Resampling

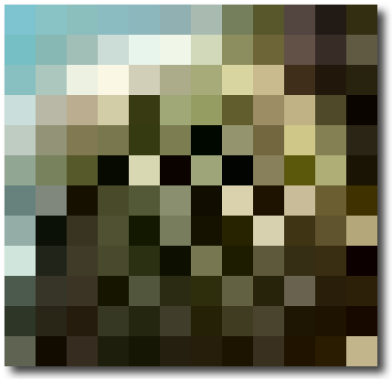
3/7 resize



Copy them into a new image.

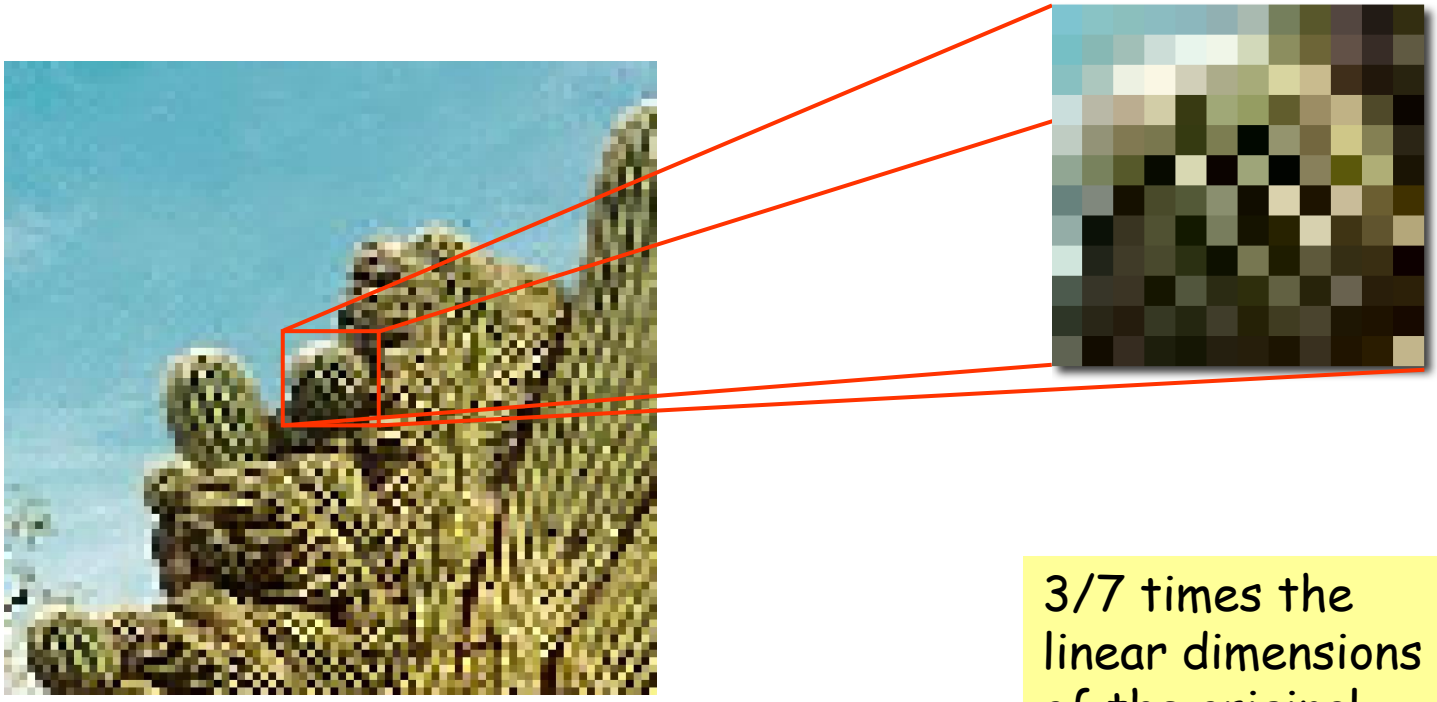
Nearest Neighbor Resampling

3/7 resize



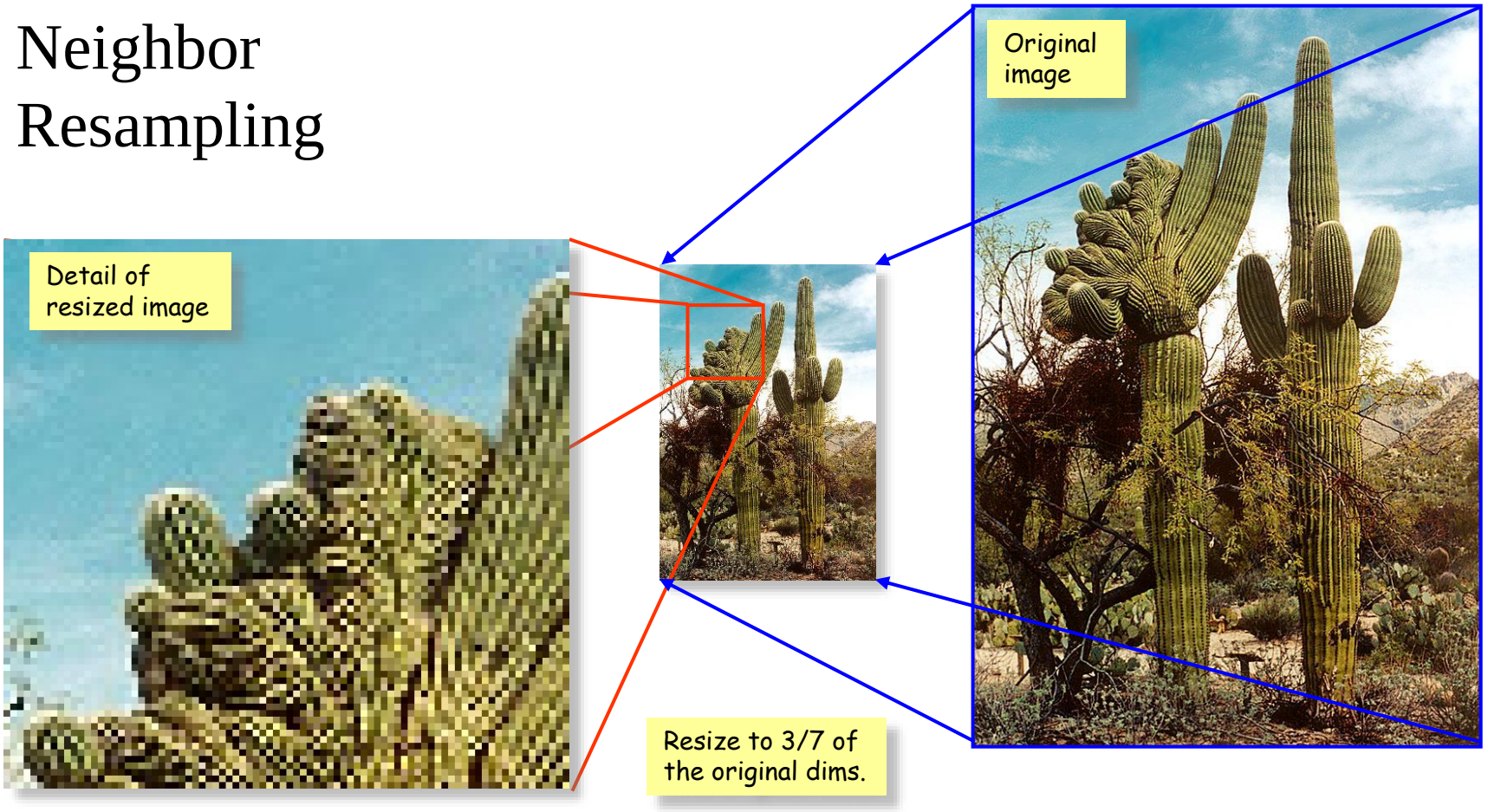
3/7 times the linear dimensions of the original

Nearest Neighbor Resampling



3/7 times the linear dimensions of the original

Nearest Neighbor Resampling

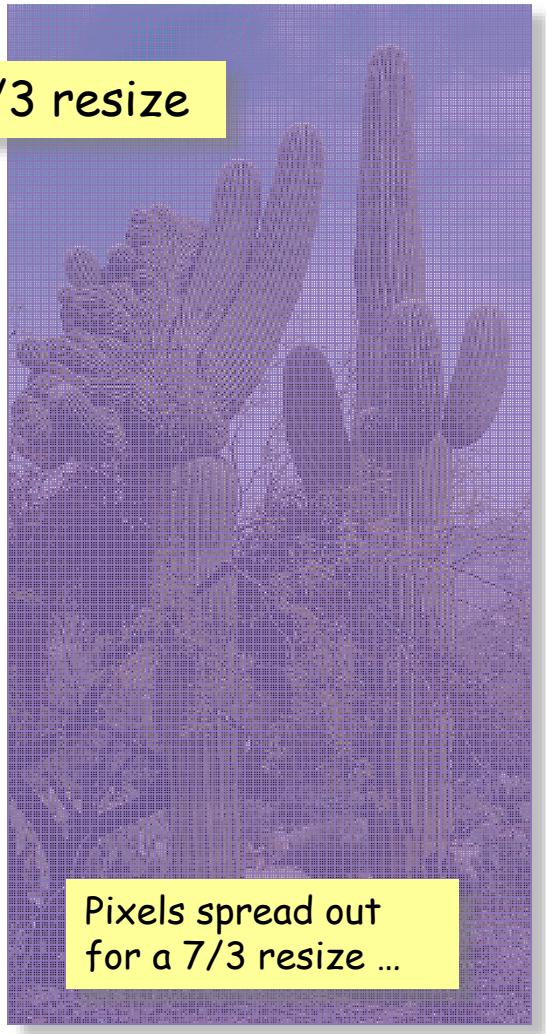


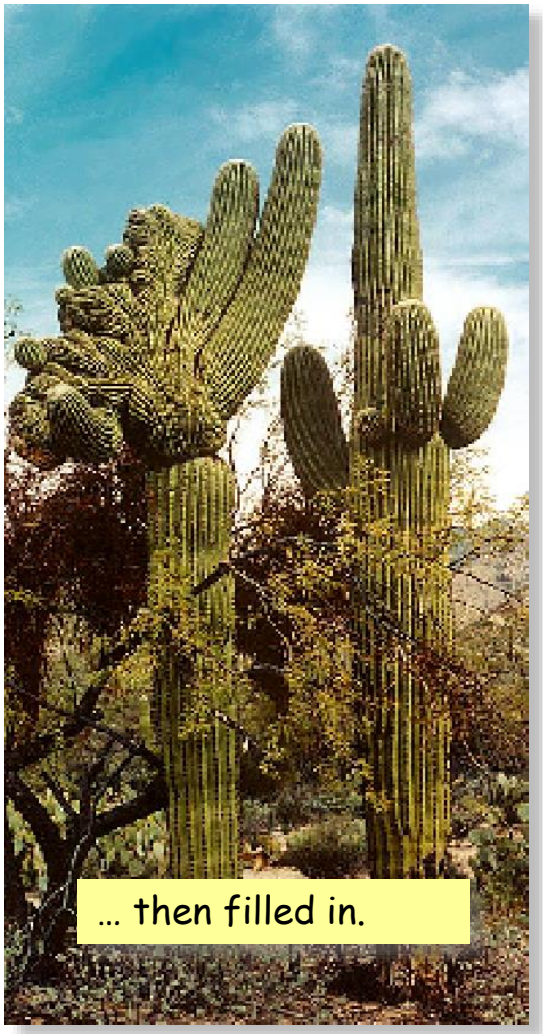
Nearest Neighbor Resampling

Resize to 3/7 of the original dims.

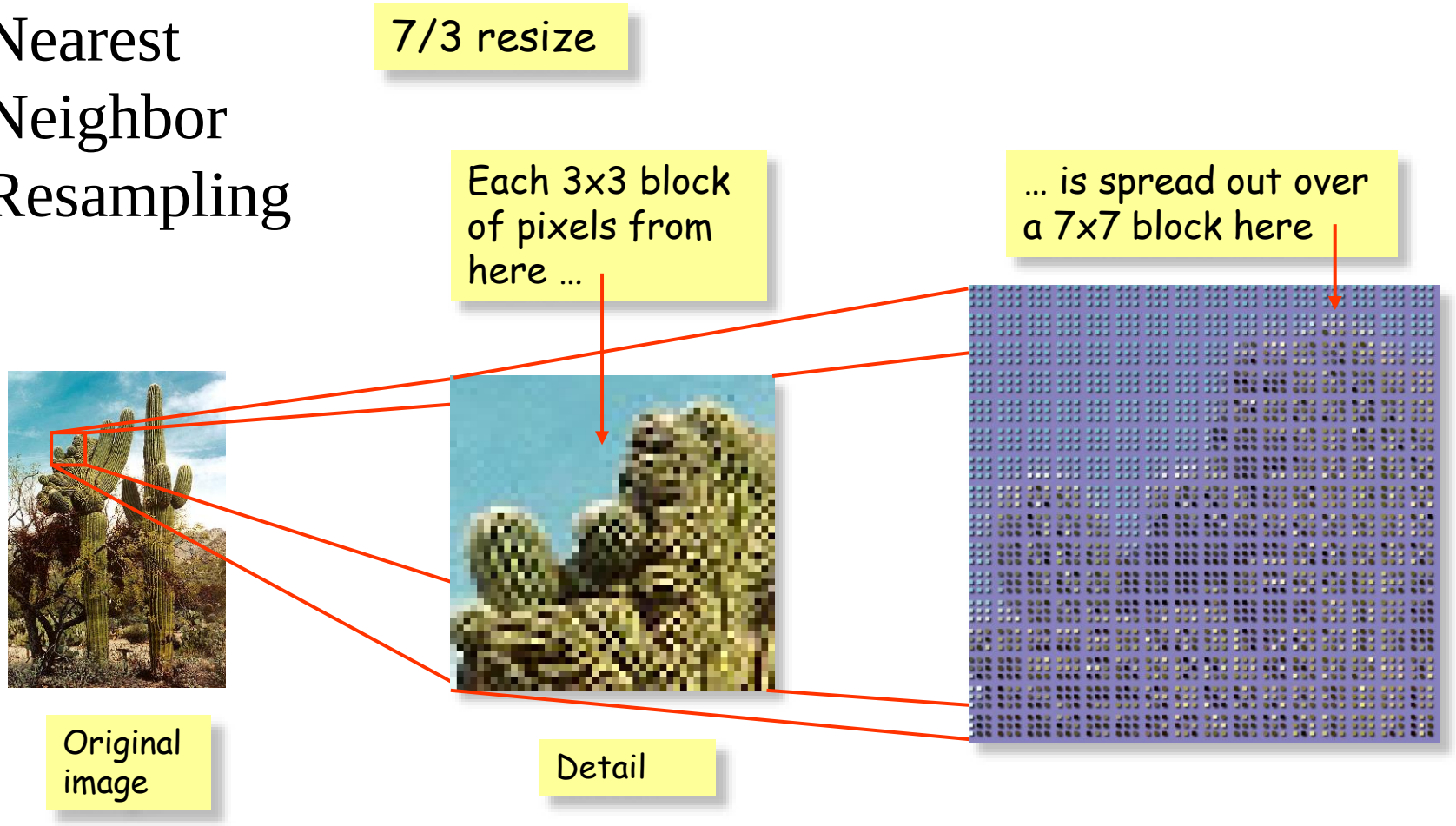
Nearest Neighbor Resampling

Original image





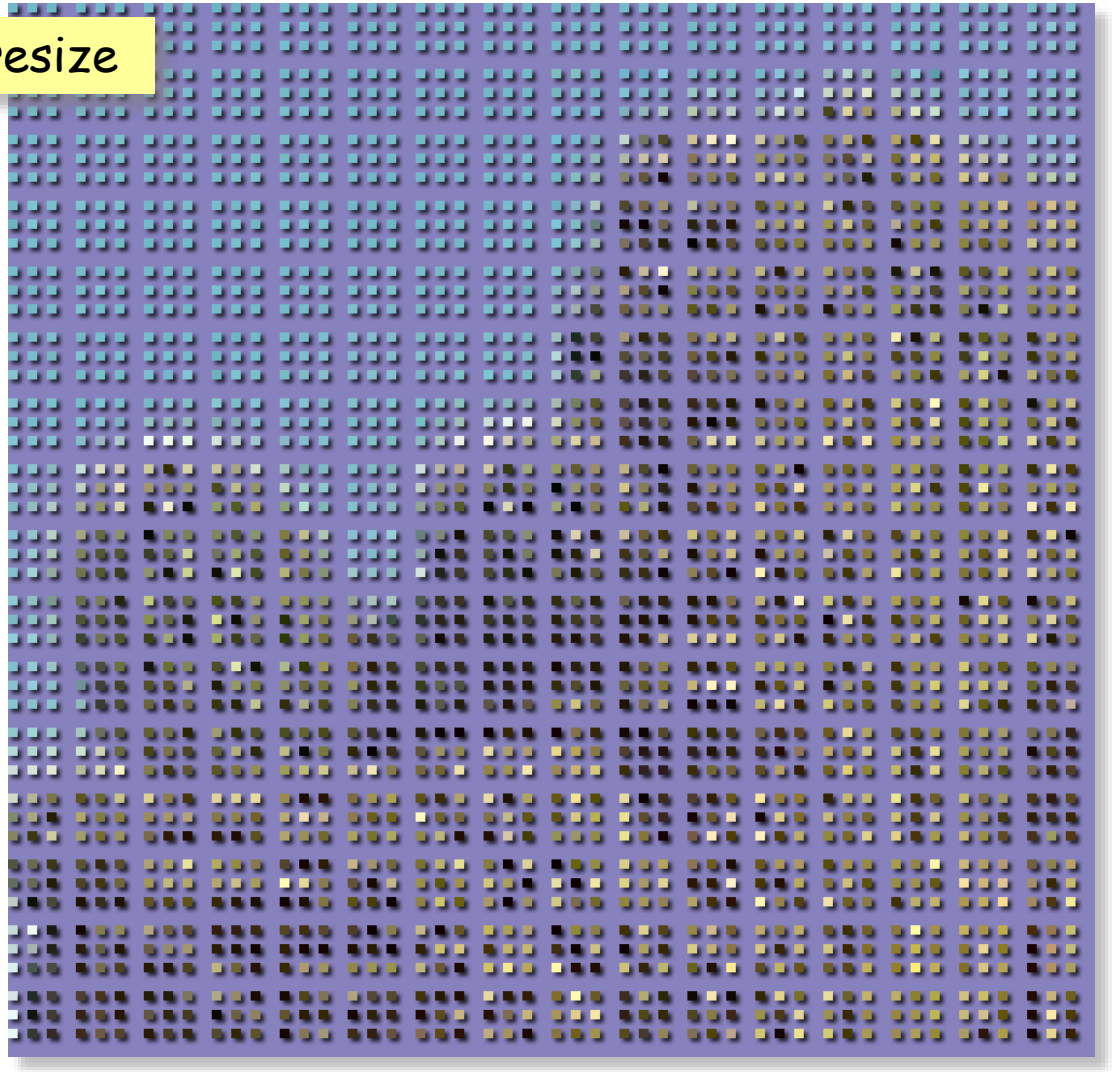
Nearest Neighbor Resampling



Nearest Neighbor Resampling

7/3 resize

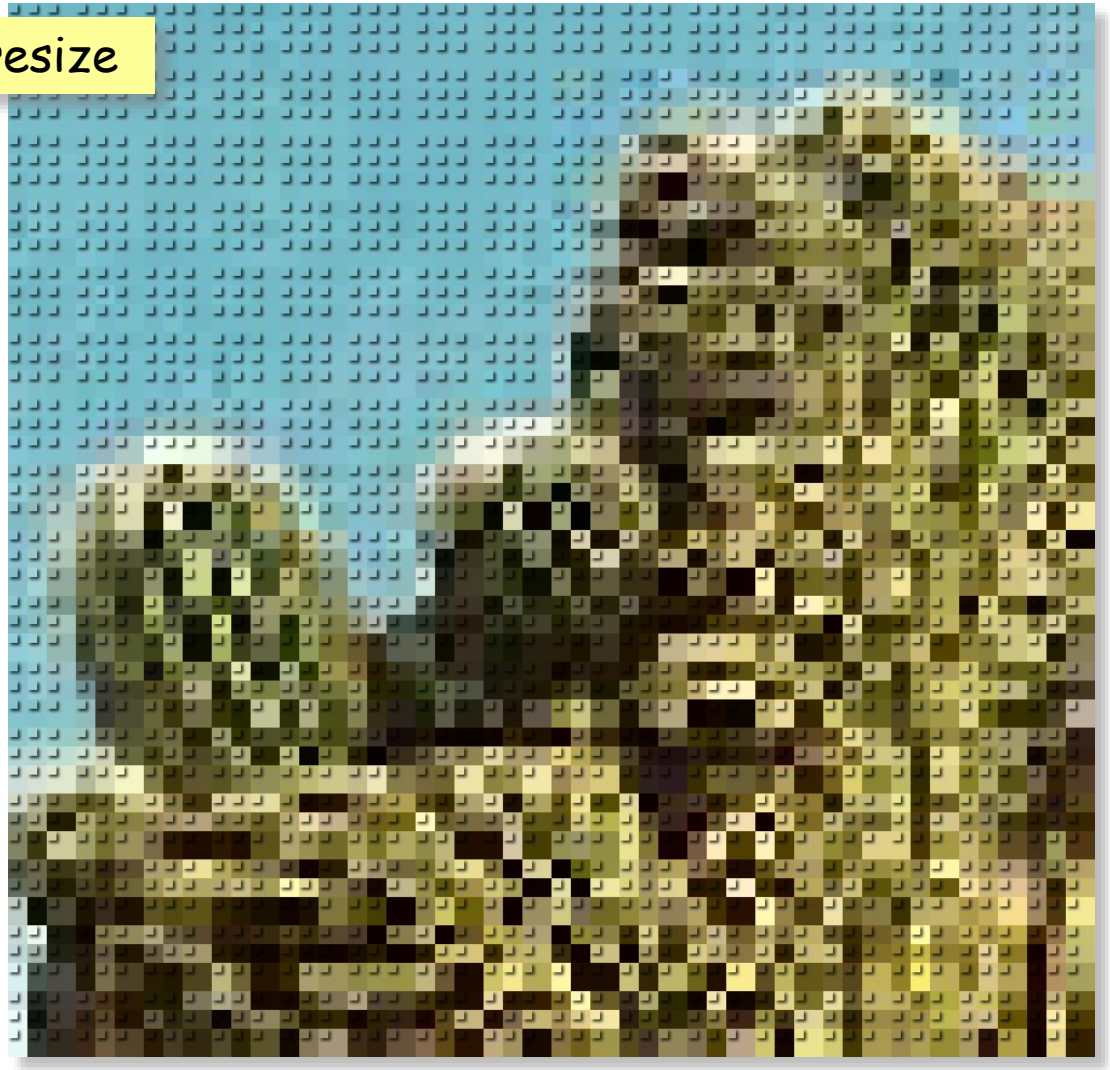
3x3 blocks distributed over 7x7 blocks



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:

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

for $r' = \alpha, \ldots, R'$,

$c' = \beta, \ldots, C'$.

If $S_r \geq 0.5$ then $\alpha = 1$.

If $S_c \geq 0.5$ then $\beta = 1$.

If $S_r < 0.5$ then $\alpha = \left\lfloor \frac{1}{S_r} \right\rfloor$.

If $S_c < 0.5$ then $\beta = \left\lfloor \frac{1}{S_c} \right\rfloor$.

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

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

Then

$$\mathbf{J}(r', c') = \mathbf{I}(\bar{r}, \bar{c}).$$

Nearest Neighbor Resampling

If the output image is larger than the input image then the scale factor is less than 1.

Size of

Size of

Row sc

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

If the output image is smaller than the input image then the scale factor is greater than 1.

Colum

S_c

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

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

Which pixel to start with?

for $r' = \alpha, \ldots, R'$,

$c' = \beta, \ldots, C'$.

If $S_r \geq 0.5$ then $\alpha = 1$.

If $S_c \geq 0.5$ then $\beta = 1$.

If $S_r < 0.5$ then $\alpha = \left\lfloor \frac{1}{S_r} \right\rfloor$.

If $S_c < 0.5$ then $\beta = \left\lfloor \frac{1}{S_c} \right\rfloor$.

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

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

Then

$$\mathbf{J}(r', c') = \mathbf{I}(\bar{r}, \bar{c}).$$

Nearest Neighbor Resampling

Size

Size

Row

Colu

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

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

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

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

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

for $r' = \alpha, \ldots, R',$

$c' = \beta, \ldots, C'.$

If $S_r \geq 0.5$ then $\alpha = 1$.

If $S_c \geq 0.5$ then $\beta = 1$.

If $S_r < 0.5$ then $\alpha = \left\lfloor \frac{1}{S_r} \right\rfloor$.

If $S_c < 0.5$ then $\beta = \left\lfloor \frac{1}{S_c} \right\rfloor$.

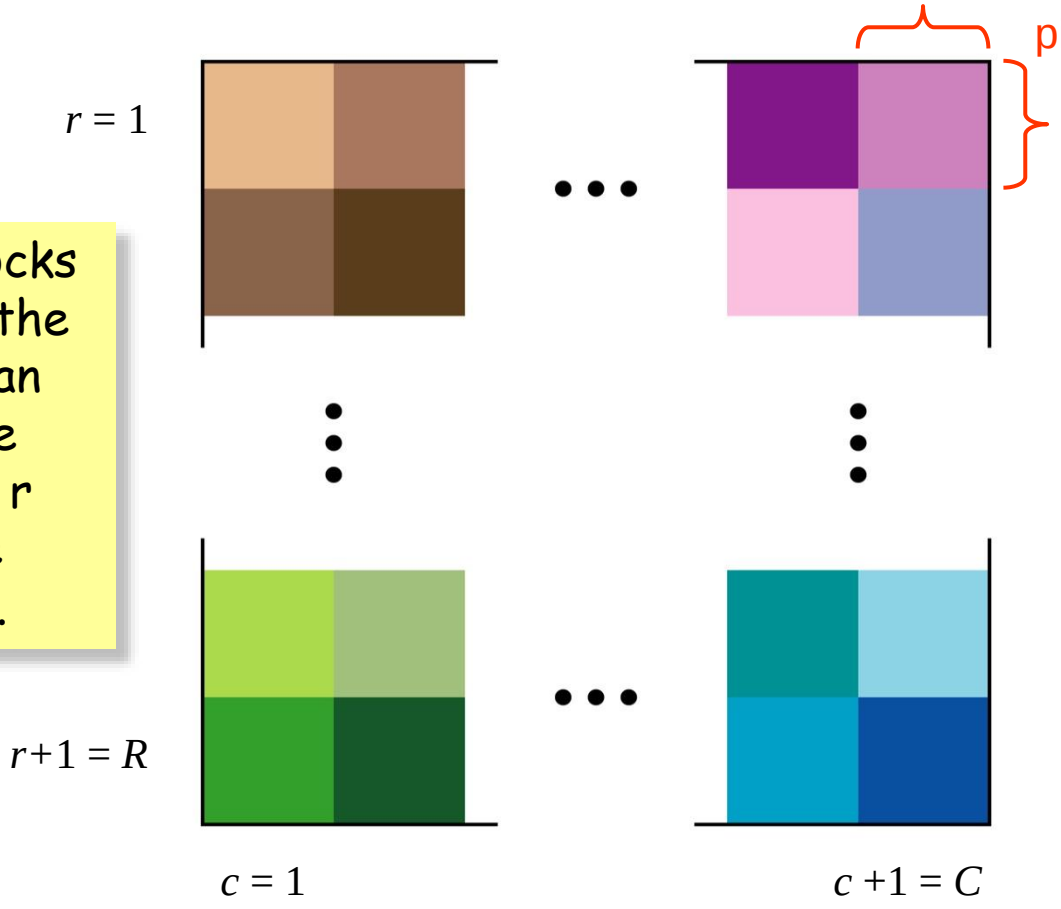
The closest integer pixel location (r, c), in I is

$$(\bar{r}, \bar{c}) = \text{round}\left(r_f, c_f\right).$$

Then

$$\mathbf{J}(r', c') = \mathbf{I}(\bar{r}, \bar{c}).$$

Nearest Neighbor Resampling

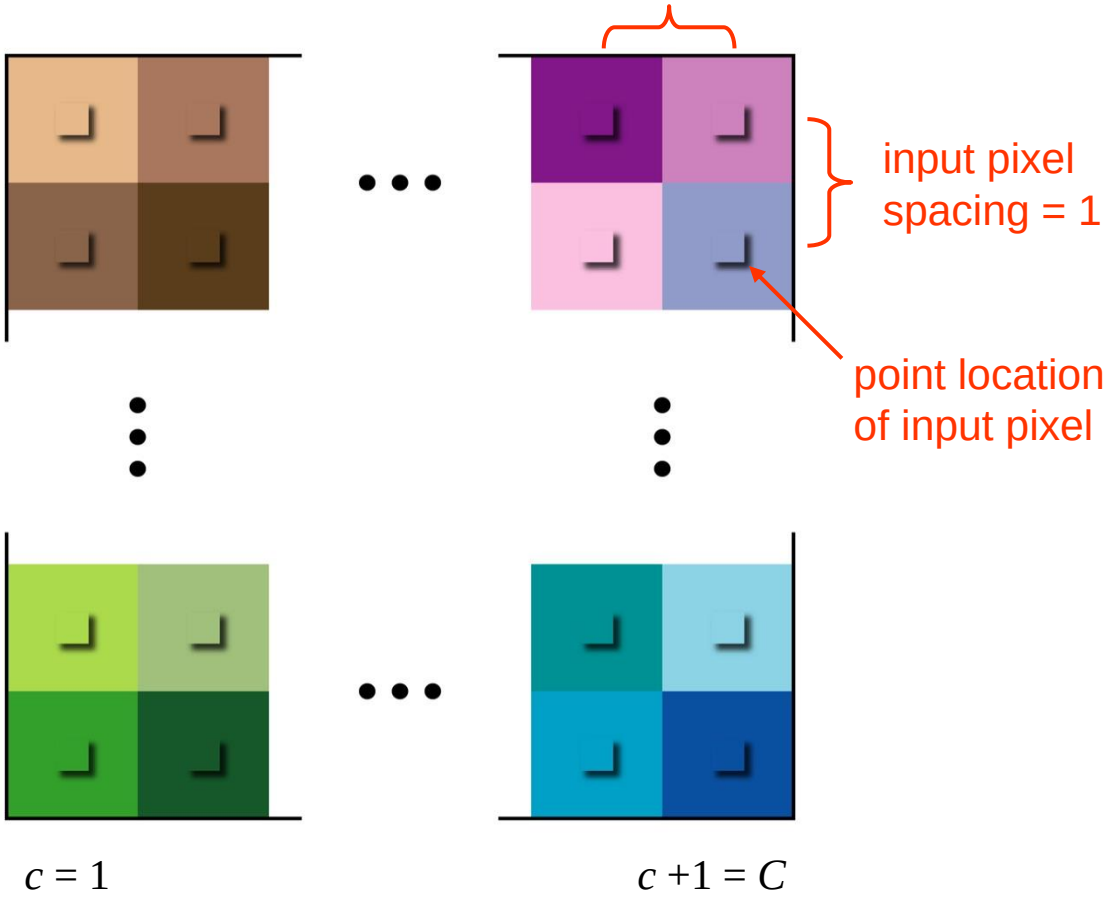


These 4 blocks represent the corners of an image — the extrema of r and c in the input image.

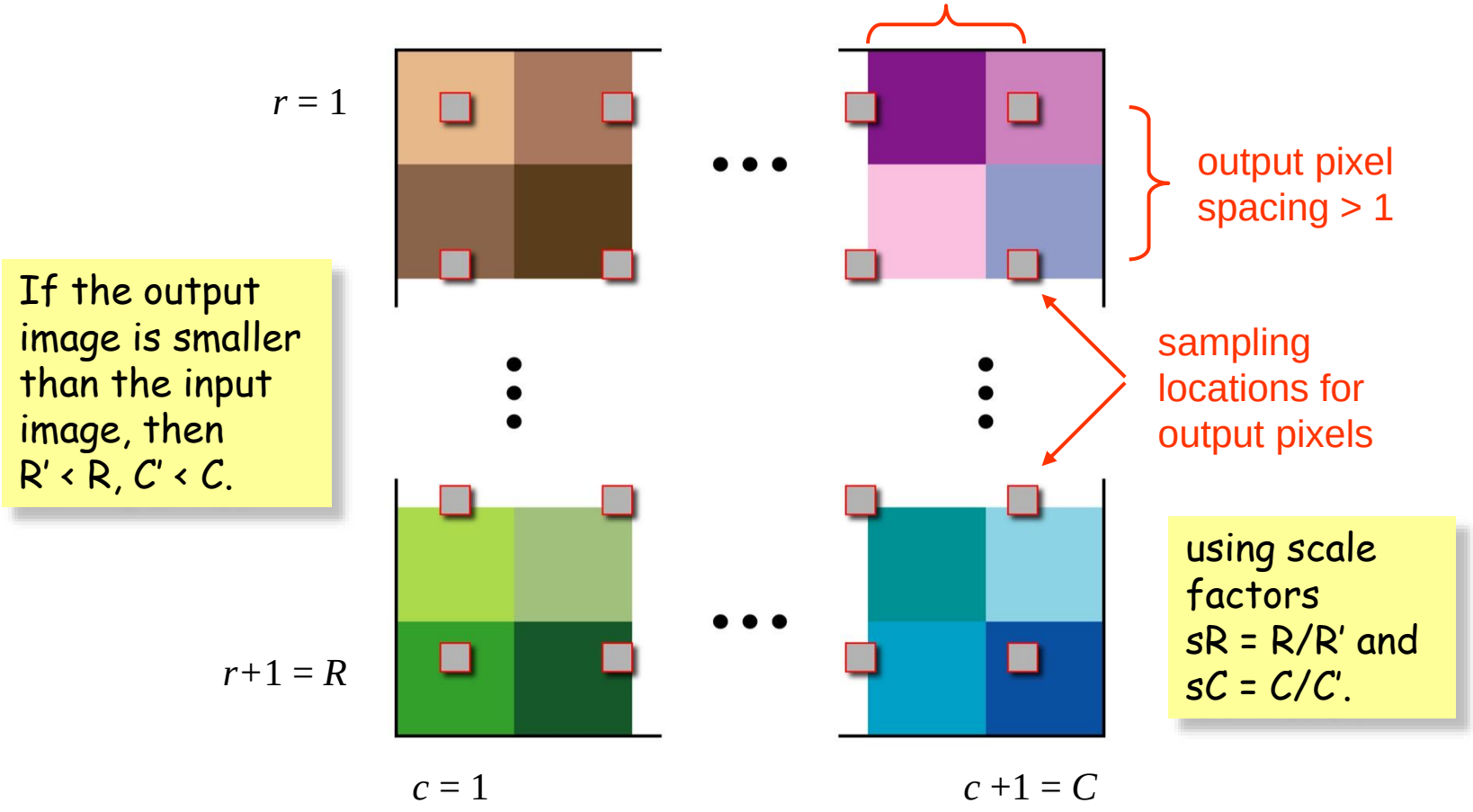
They are also the extrema of 4-pixel neighborhoods in the image.

Nearest Neighbor Resampling

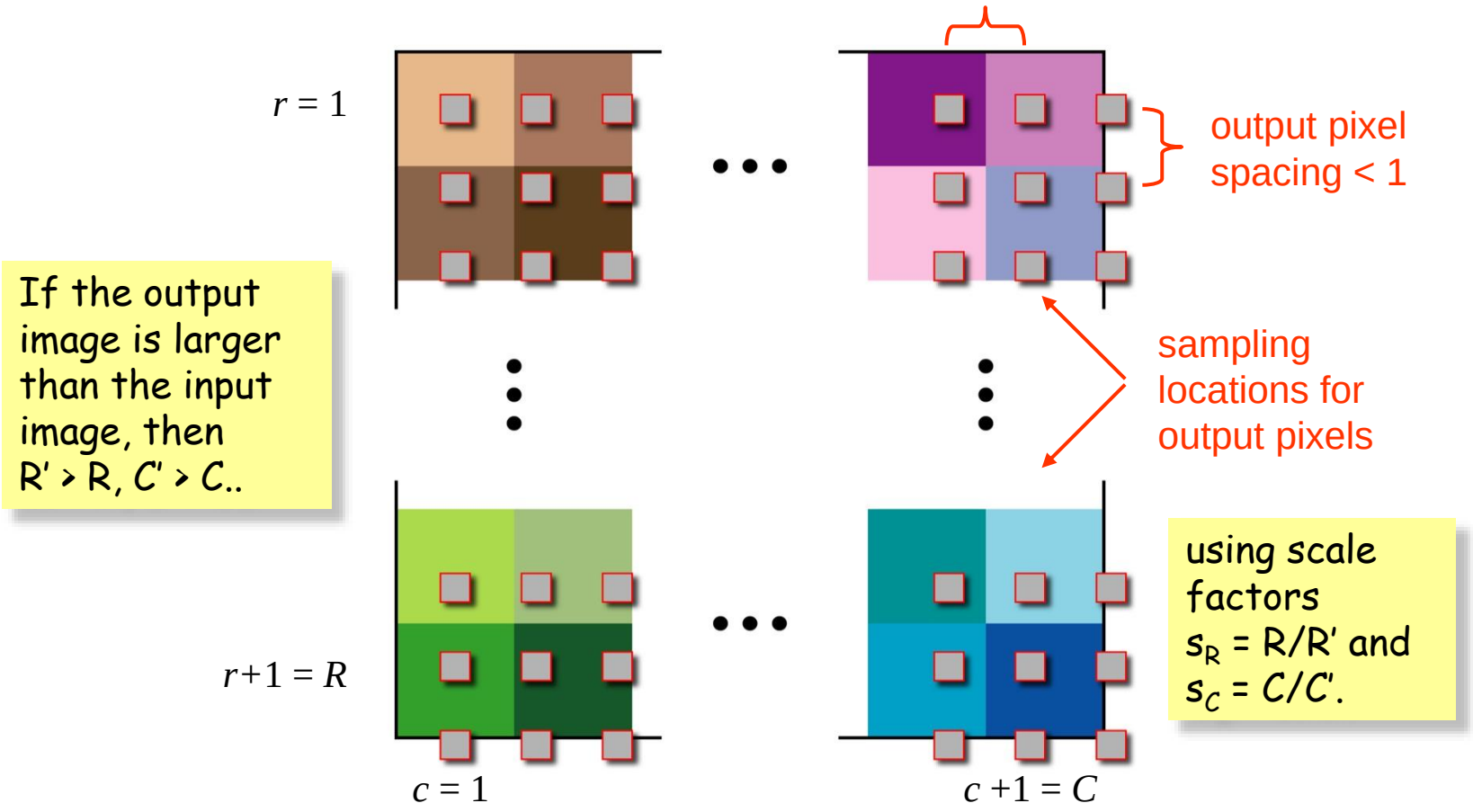
Here the image is supersampled with the original pixels in the center. This is not actually none by the algorithm but it helps one visualize the procedure



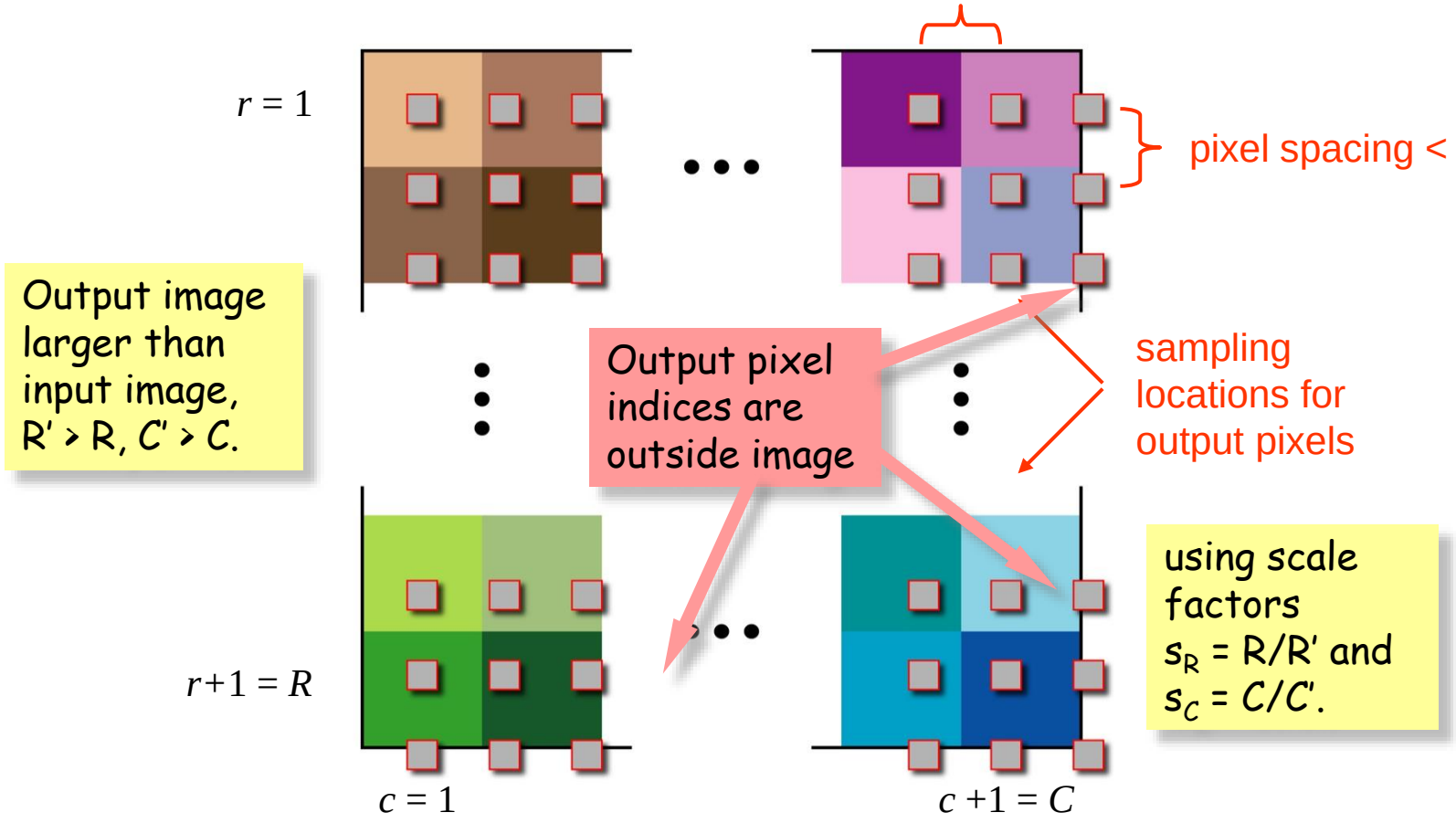
Nearest Neighbor Resampling



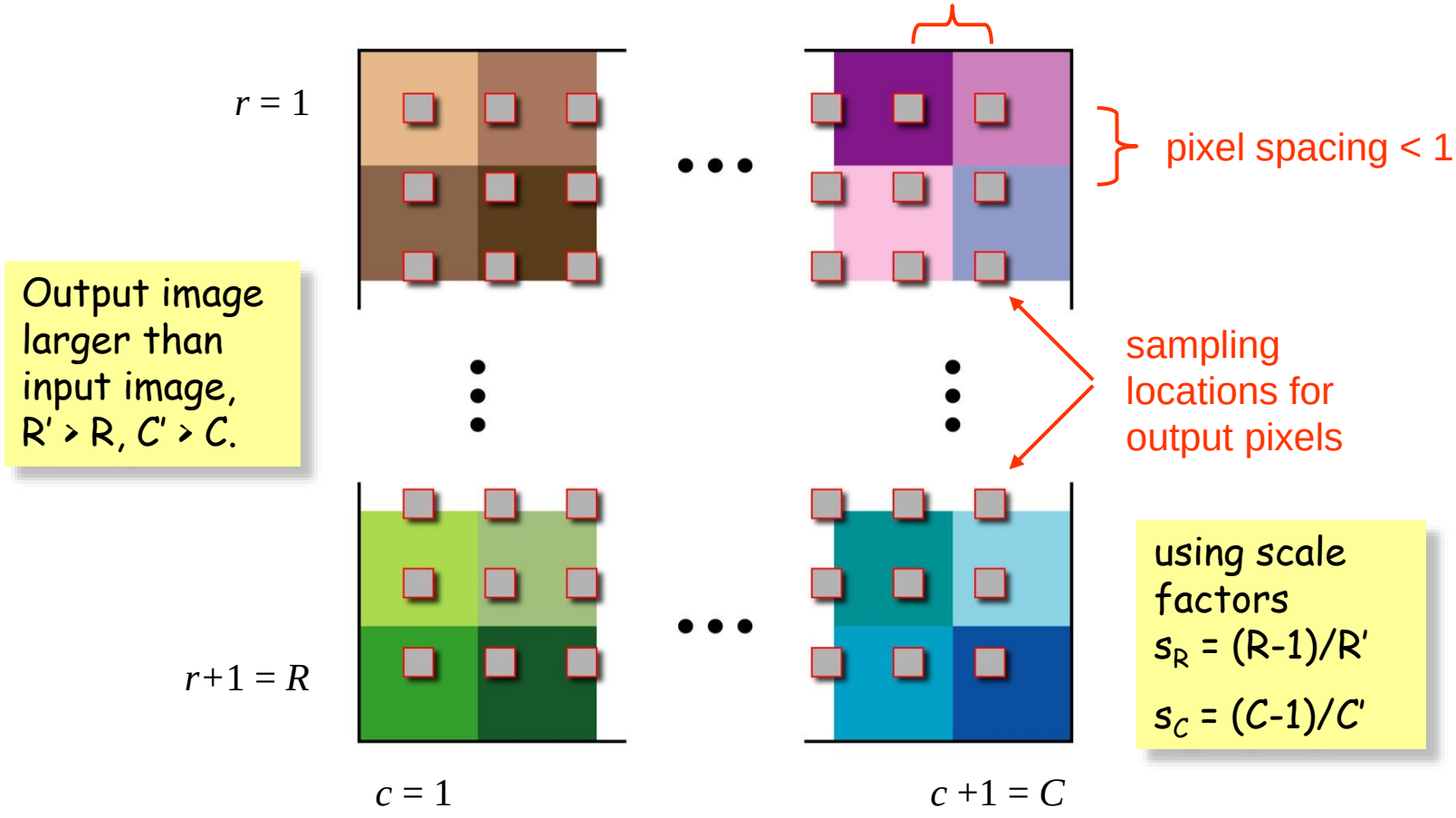
Nearest Neighbor Resampling



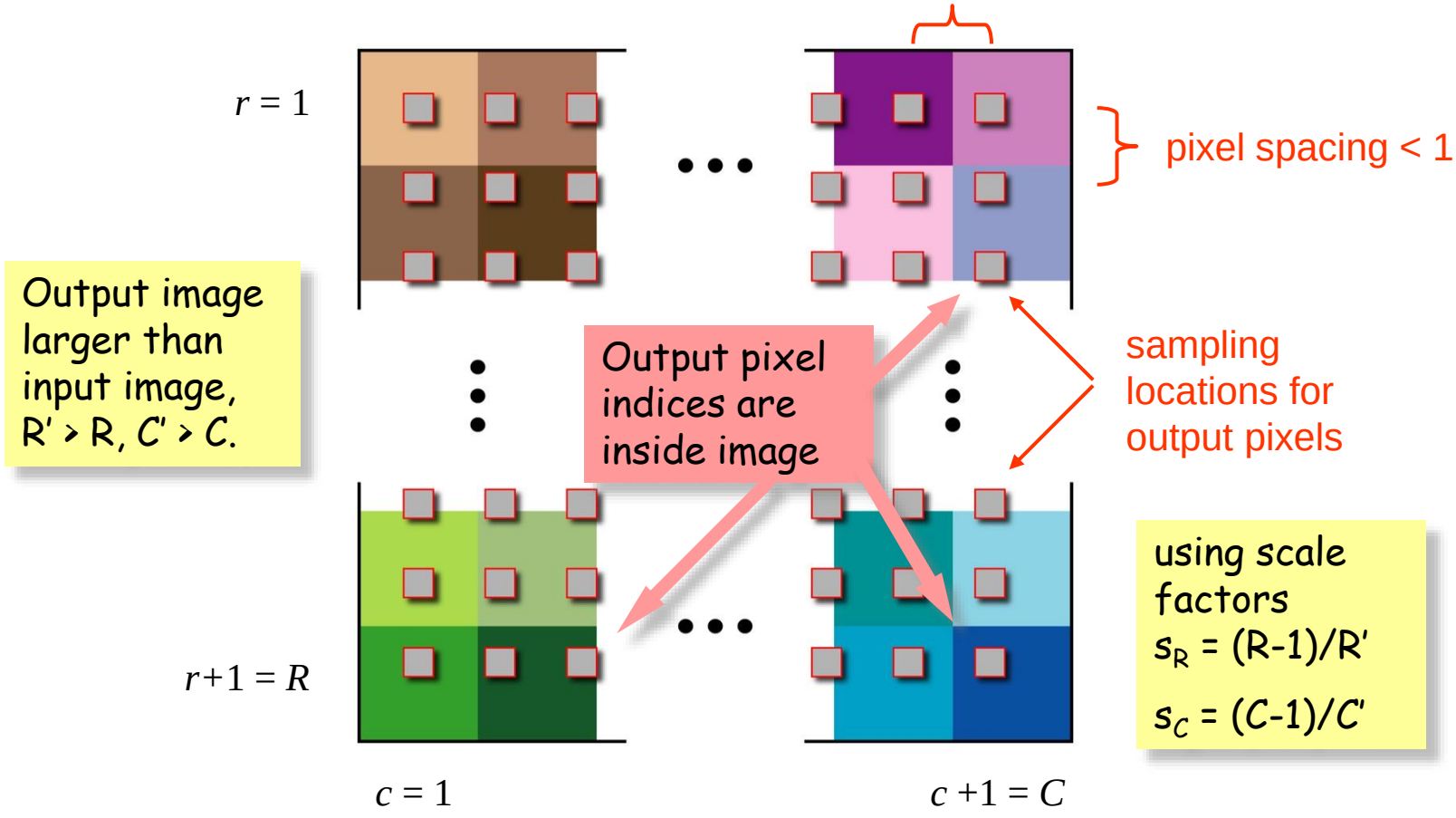
Nearest Neighbor Resampling



Nearest Neighbor Resampling



Nearest Neighbor Resampling



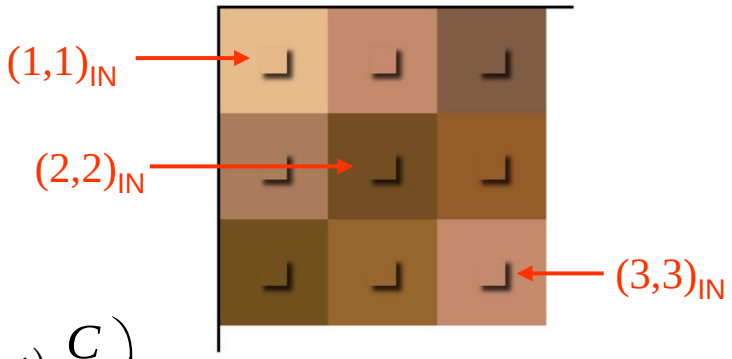
Nearest Neighbor Resampling

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

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

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

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



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

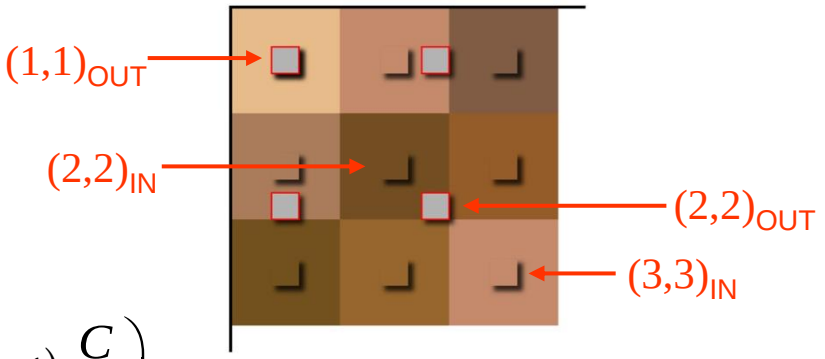
Nearest Neighbor Resampling

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

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

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

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



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

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

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

Nearest Neighbor Resampling

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

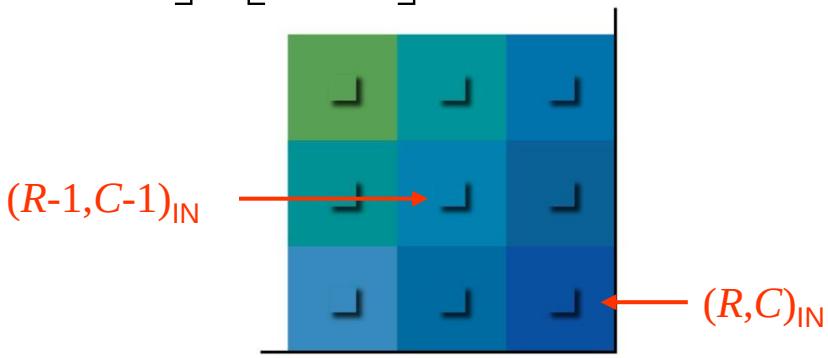
$$(R',C')_{\text{OUT}} \leftrightarrow (1,1)_{\text{IN}} + \left((R'-1)\frac{R}{R'}, (C'-1)\frac{C}{C'} \right)$$

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 $C' < C$.

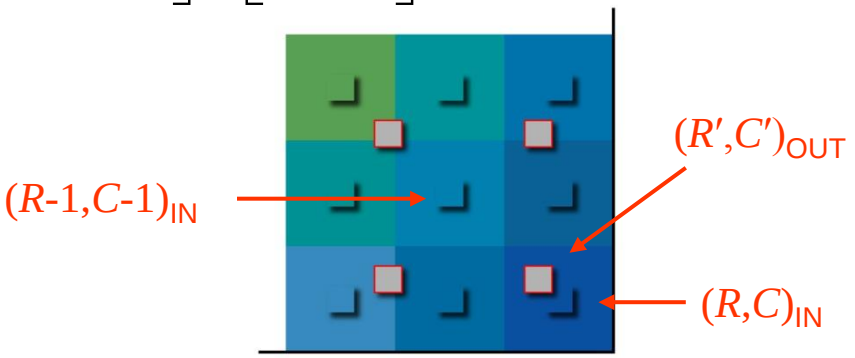
$$(R',C')_{\text{OUT}} \leftrightarrow (1,1)_{\text{IN}} + \left((R'-1)\frac{R}{R'}, (C'-1)\frac{C}{C'} \right)$$

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 $(R, C)_{\text{IN}}$.



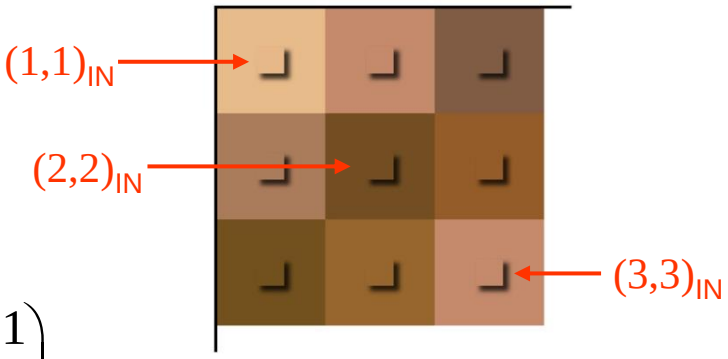
Nearest Neighbor Resampling

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

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

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

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



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 $C < C'$.

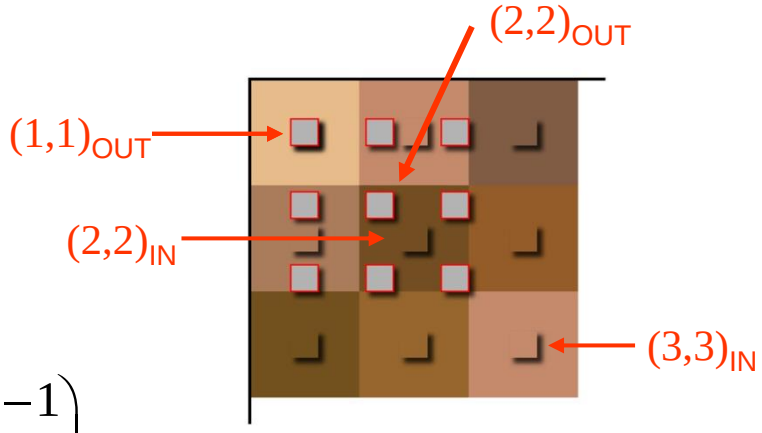
$$(1,1)_{\text{OUT}} \leftrightarrow (1,1)_{\text{IN}}$$

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

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

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

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



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

Nearest Neighbor Resampling

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

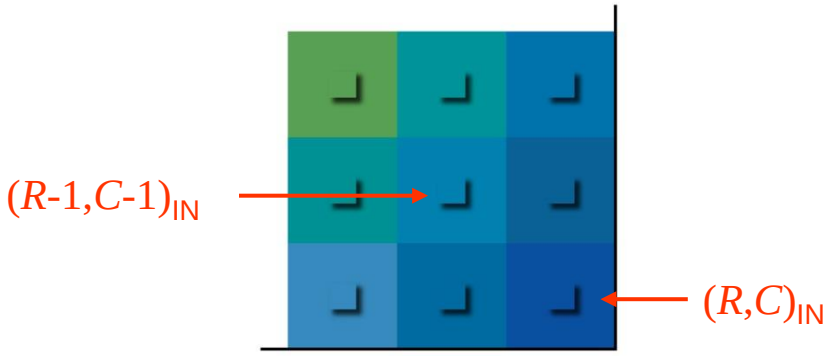
$$(R',C')_{\text{OUT}} \leftrightarrow (1,1)_{\text{IN}} + \left((R'-1)\frac{R-1}{R'}, (C-1)\frac{R-1}{C'} \right)$$

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 $C < C'$.

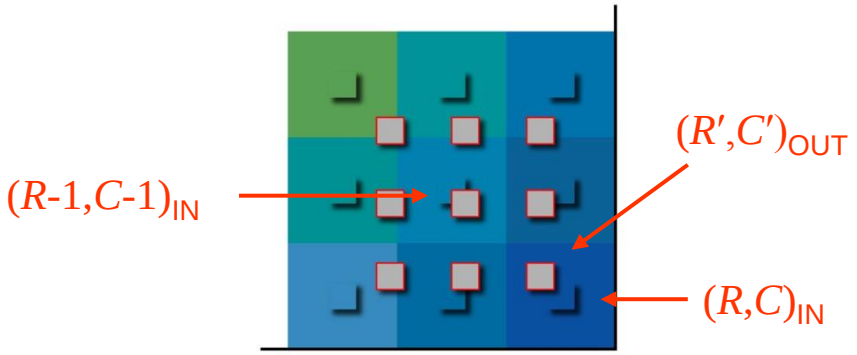
$$(R',C')_{\text{OUT}} \leftrightarrow (1,1)_{\text{IN}} + \left((R'-1)\frac{R-1}{R'}, (C-1)\frac{R-1}{C'} \right)$$

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 $(R, C)_{\text{IN}}$.



Interpolation

Assume that an image, **I**, is defined on a rectangle in a continuous domain and takes on a continuous range of non-negative values on that domain. That is,

$$\mathbf{I}:\left[r_1,r_R\right]\times\left[c_1,c_C\right]\rightarrow\left[g_{\min},g_{\max}\right] \text{ where } g_{\min}\geq 0 \text{ and } g_{\max}<\infty.$$

Also assume that we know the values of **I** only on an evenly spaced grid of discrete points in the domain. That is, we know

$$\mathbf{I}(r,c) \text{ for } r\in\{1,\ldots,R\} \text{ and } c\in\{1,\ldots,C\}.$$

Unknown values can be estimated through *interpolation*. Most methods of interpolation fit a 2-D polynomial to the known points surrounding the unknown point, then use its value at the unknown point. For example, if we want to know **I** at subpixel location, (4.5,3.75) we would find a polynomial, *p*, such that

$$p(4,3)=\mathbf{I}(4,3),\ p(4,4)=\mathbf{I}(4,4),\ p(5,4)=\mathbf{I}(5,4),\ \text{and}\ p(5,3)=\mathbf{I}(5,3).$$

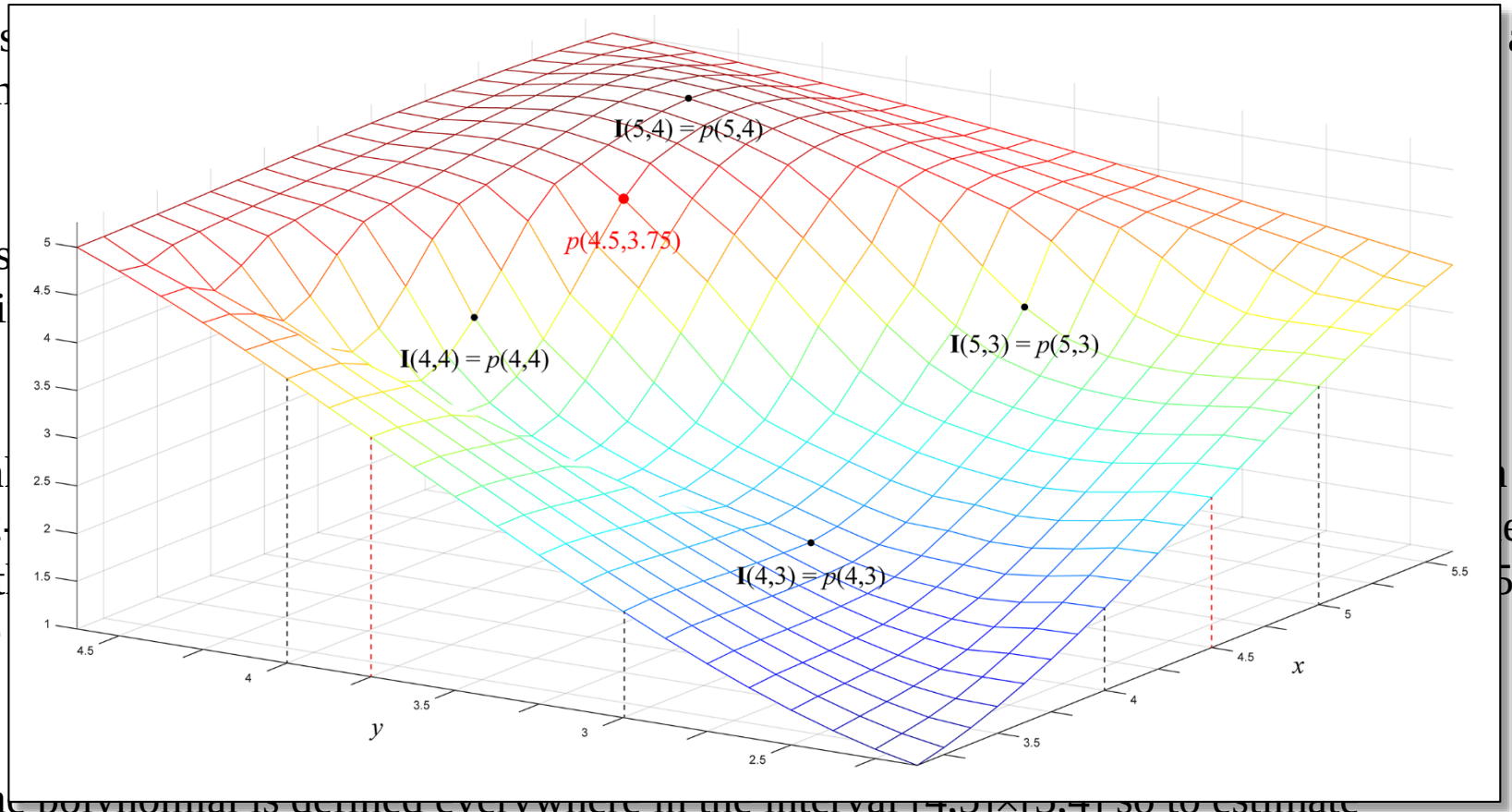
The polynomial is defined everywhere in the interval [4,5]×[3,4] so to estimate **I**(4.5,3.75), we compute *p*(4.5,3.75).

Interpolation

Ass a
cor

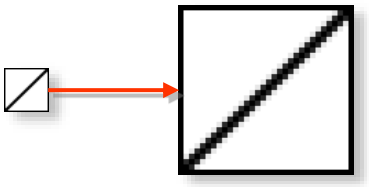
Als
poi

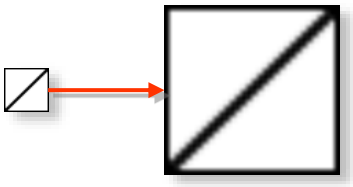
Un fit
a 2- e
at t 5)
we



The polynomial is defined everywhere in the interval [4,5]×[3,4] so to estimate **I**(4.5,3.75), we compute $p(4.5,3.75)$.

Enlarging Images: Replication vs. Interpolation



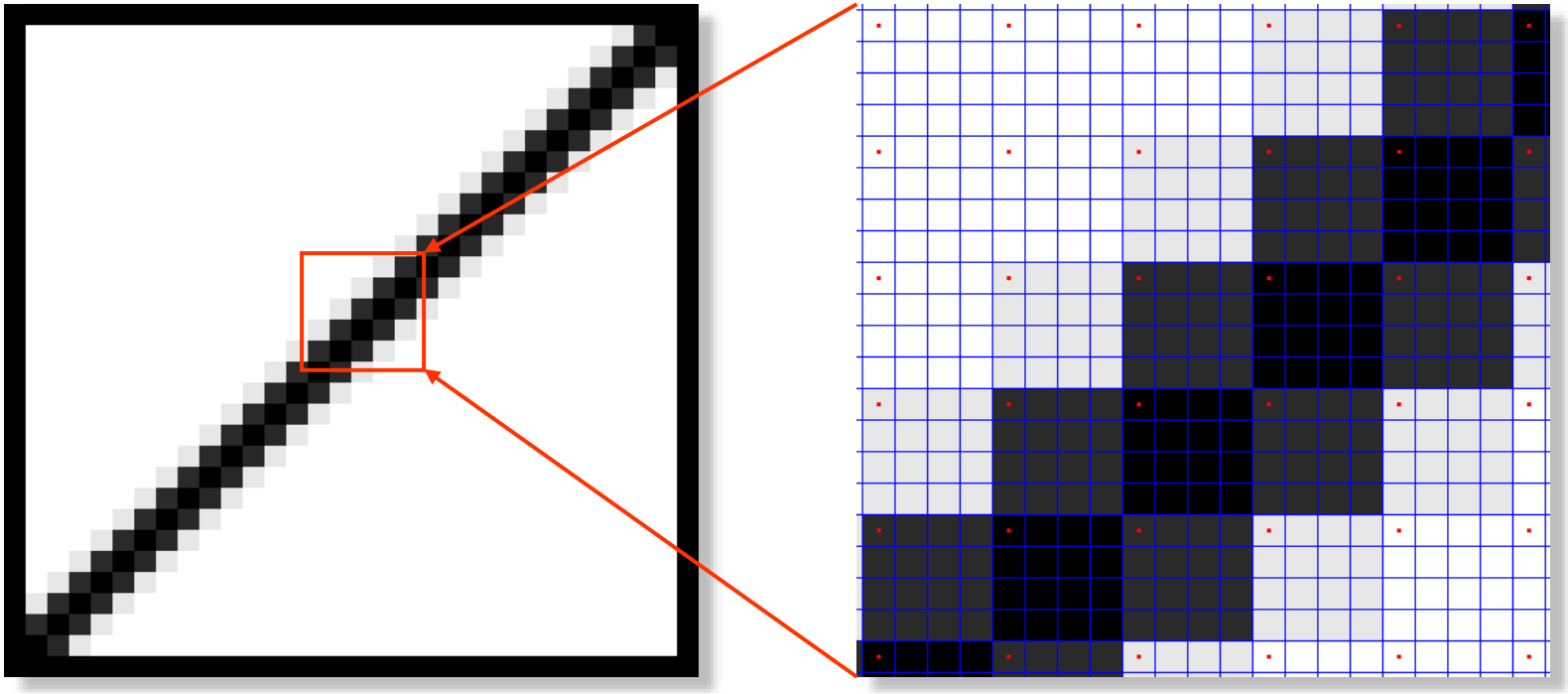


Pixel replication produces a "jagged" result since each individual square pixel is made larger.

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

Pixel Replication

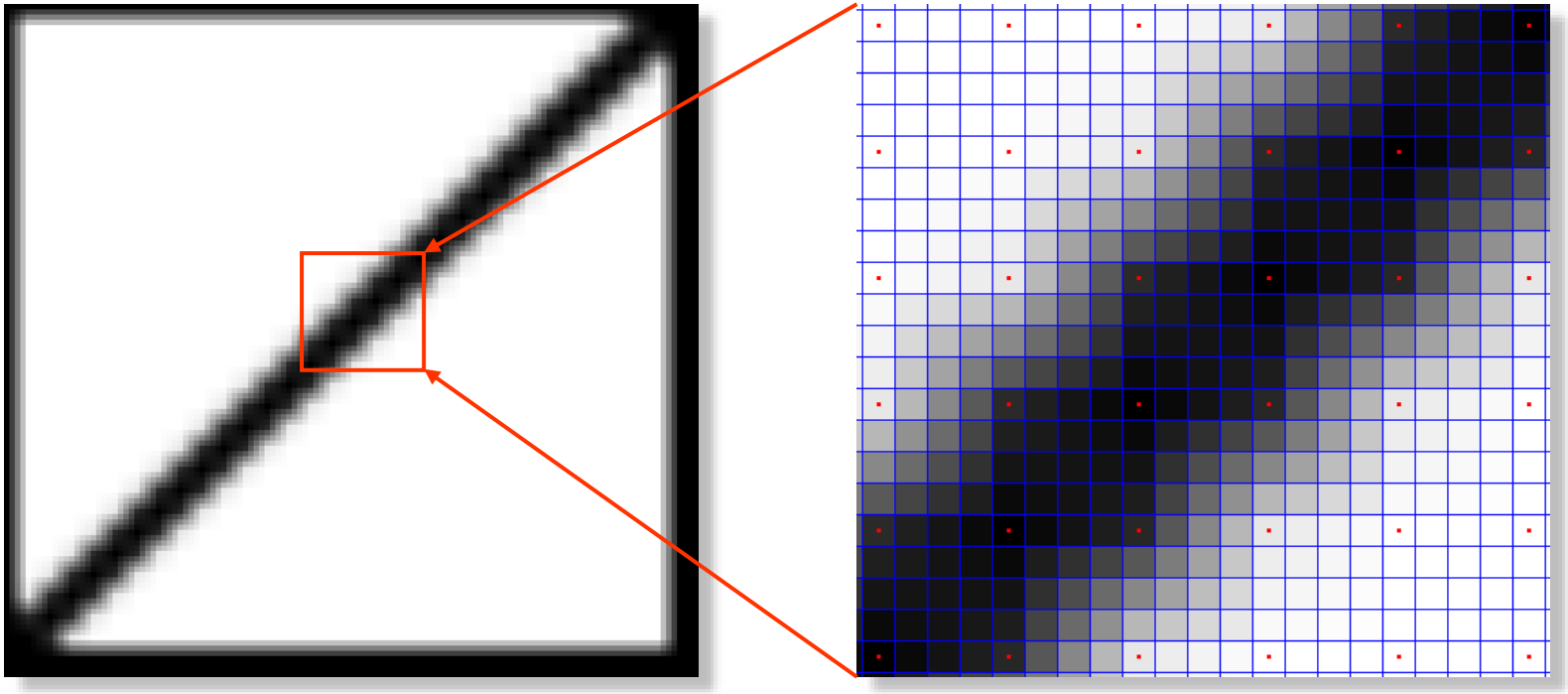
Red dots mark original pixel values.



Small square pixels become large square pixels.

Bilinear Interpolation

Red dots mark original pixel values.



Intermediate locations are filled with intermediate values.

Bilinear Interpolation

Bilinear interpolation uses a first-order polynomial – a plane – to estimate a subpixel value. A bilinear polynomial is defined by

$$p_{\text{bilin}}(r,c) = \alpha r + \beta c + \gamma rc + \delta.$$

If r_f and c_f are non integers and we know **I** at integer locations, we estimate $\mathbf{I}(r_f, c_f)$ as $p_{\text{bilin}}(\Delta r, \Delta c)$ with $\Delta r = r_f - r_0$ and $\Delta c = c_f - c_0$ where $r_0 = \lfloor r_f \rfloor$ and $c_0 = \lfloor c_f \rfloor$. Then

$$\begin{aligned} p_{\text{bilin}}(\Delta r, \Delta c) = & \left[\mathbf{I}(r_0+1, c_0) - \mathbf{I}(r_0, c_0)\right]\Delta r \\ & + \left[\mathbf{I}(r_0, c_0+1) - \mathbf{I}(r_0, c_0)\right]\Delta c \\ & + \left[\mathbf{I}(r_0+1, c_0+1) - \mathbf{I}(r_0+1, c_0) - \mathbf{I}(r_0, c_0+1) + \mathbf{I}(r_0, c_0)\right]\Delta r \Delta c \\ & + \mathbf{I}(r_0, c_0). \end{aligned}$$

Therefore, the coefficients are

$$\alpha = \mathbf{I}(r_0+1, c_0) - \mathbf{I}(r_0, c_0),\ \beta = \mathbf{I}(r_0, c_0+1) - \mathbf{I}(r_0, c_0),\ \delta = \mathbf{I}(r_0, c_0),\ \text{and}$$

$$\gamma = \mathbf{I}(r_0+1, c_0+1) - \mathbf{I}(r_0+1, c_0) - \mathbf{I}(r_0, c_0+1) + \mathbf{I}(r_0, c_0).$$

Resampling Through Bilinear Interpolation

Let **I** be an $R \times C$ image.
We want to resize **I** to $R' \times C'$ to make a new image **J** with pixel locs (r',c').

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

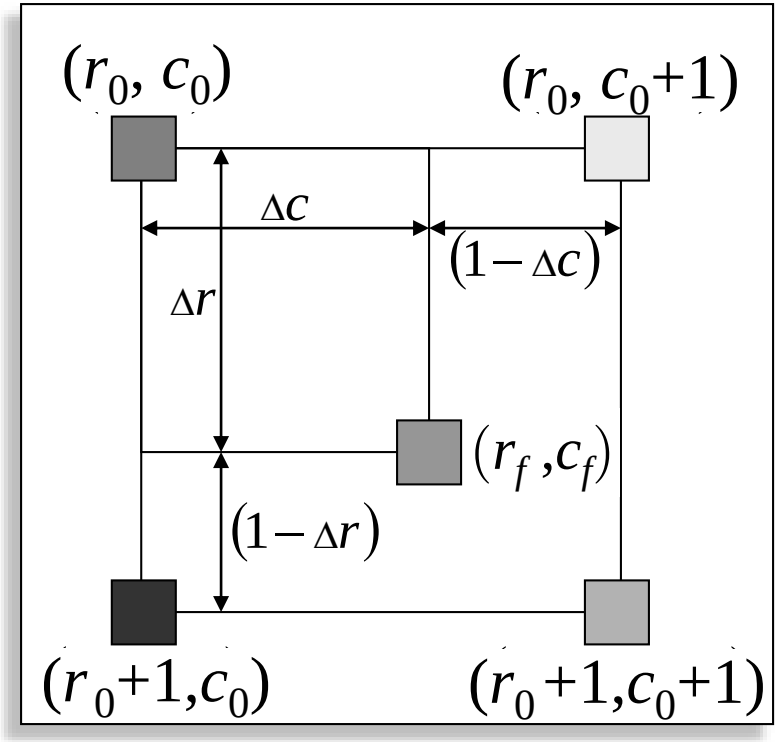
Let $r_f = r' \cdot s_R$ for $r' = 1,\ldots,R'$

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

Let $r_0 = \lfloor r_f \rfloor$ and $c_0 = \lfloor c_f \rfloor$.

Let $\Delta r = r_f - r_0$ and $\Delta c = c_f - c_0$.

Then $\mathbf{J}(r',c') = \mathbf{I}(r_0,c_0)\cdot(1-\Delta r)\cdot(1-\Delta c)$
$+\mathbf{I}(r_0+1,c_0)\cdot\Delta r\cdot(1-\Delta c)$
$+\mathbf{I}(r_0,c_0+1)\cdot(1-\Delta r)\cdot\Delta c$
$+\mathbf{I}(r_0+1,c_0+1)\cdot\Delta r\cdot\Delta c.$



Resampling Through Bilinear In

Let **I** b
We wa t to resize **I** to $R \times C'$ to make
a new mage **J** with pix l locs (r',c').

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

Let $r_0 = \lfloor r_f \rfloor$ and $c_0 = \lfloor c_f \rfloor$.

Let $\Delta r = r_f - r_0$ and $\Delta c = c_f - c_0$.

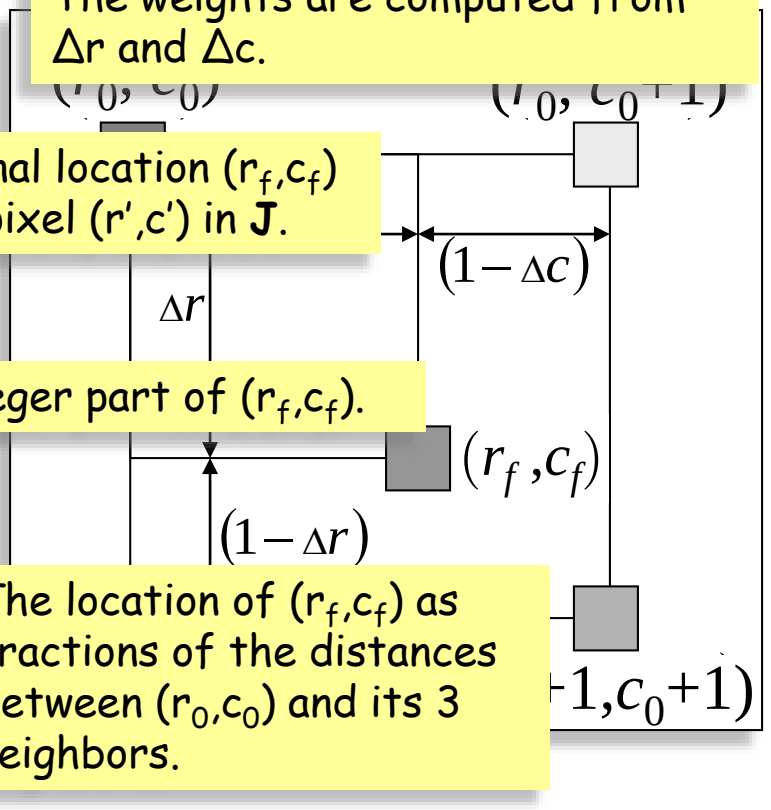
Then $\mathbf{J}(r',c') = \mathbf{I}(r_0,c_0)\cdot(1-\Delta r)\cdot(1-\Delta c)$
$+\mathbf{I}(r_0+1,c_0)\cdot\Delta r\cdot(1-\Delta c)$
$+\mathbf{I}(r_0,c_0+1)\cdot(1-\Delta r)\cdot\Delta c$
$+\mathbf{I}(r_0+1,c_0+1)\cdot\Delta r\cdot\Delta c.$

For each pixel (r′,c′) in output image, **J**, compute the fractional location (r_f,c_f) in **I**. Use (r_0,c_0), 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.

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

Integer part of (r_f,c_f).

The location of (r_f,c_f) as fractions of the distances between (r_0,c_0) and its 3 neighbors.



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:

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

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

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

(r_0, c_0) are the row and column indices of the pixels in the input image to use in the algorithm.

$$(r_0, c_0) = \left(\lfloor r_f \rfloor, \lfloor c_f \rfloor \right)$$

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

$$(\Delta r, \Delta c) = (r_f - r_0, c_f - c_0)$$

Then the value of each output pixel is given by

$$\begin{aligned} \mathbf{J}(r', c') = {} & \mathbf{I}(r_0, c_0) \cdot (1 - \Delta r) \cdot (1 - \Delta c) \\ & + \mathbf{I}(r_0 + 1, c_0) \cdot \Delta r \cdot (1 - \Delta c) \\ & + \mathbf{I}(r_0, c_0 + 1) \cdot (1 - \Delta r) \cdot \Delta c \\ & + \mathbf{I}(r_0 + 1, c_0 + 1) \cdot \Delta r \cdot \Delta c. \end{aligned}$$

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 $(r_0,c_0) = \lfloor (r_f,c_f,) \rfloor$ as its upper left corner and has (r_0+1,c_0+1) as its lower right corner.

Thus for each $(r_f,c_f,)$: (1) neither r_0 nor c_0 can be less than one and (2) r_0+1 cannot be greater than R and c_0+1 cannot be greater than C.

If the set of all indices $\{(r_0,c_0)\}$ do not satisfy (1) or (2), you must adjust the indices so that they do.

pixel $(r\,,\,c\,)$.

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

$(r_0,\, c_0\,)$ are the row and column indices of the pixels in the input image to use in the algorithm.

$$(r_0,\, c_0\,) = \left(\lfloor r_f \rfloor, \lfloor c_f \rfloor \right)$$

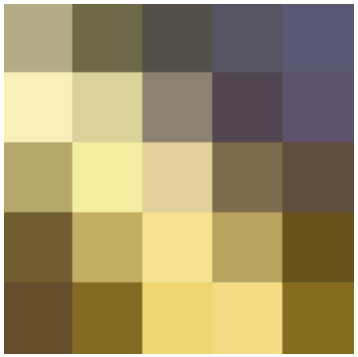
$(\Delta r,\, \Delta c)$ are the fractional parts of the row and column locations, $(r_f\,,\, c_f\,)$.

$$(\Delta r,\, \Delta c\,) = (\, r_f - r_0\,,\, c_f - c_0\,)$$

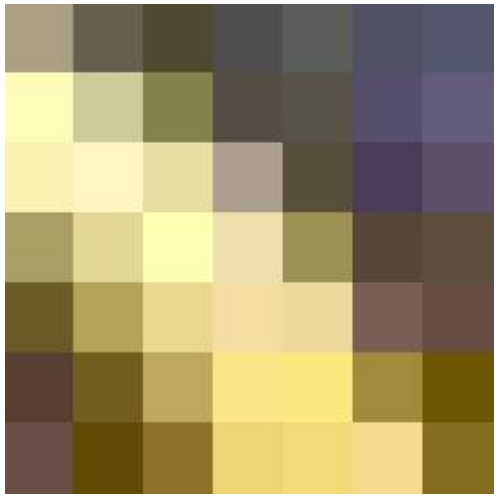
Then the value of each output pixel is given by

$$\begin{aligned} \mathbf{J}(r',c') = {} & \mathbf{I}(r_0,c_0)\cdot(1-\Delta r)\cdot(1-\Delta c) \\ & + \mathbf{I}(r_0+1,c_0)\cdot\Delta r\cdot(1-\Delta c) \\ & + \mathbf{I}(r_0,c_0+1)\cdot(1-\Delta r)\cdot\Delta c \\ & + \mathbf{I}(r_0+1,c_0+1)\cdot\Delta r\cdot\Delta c. \end{aligned}$$

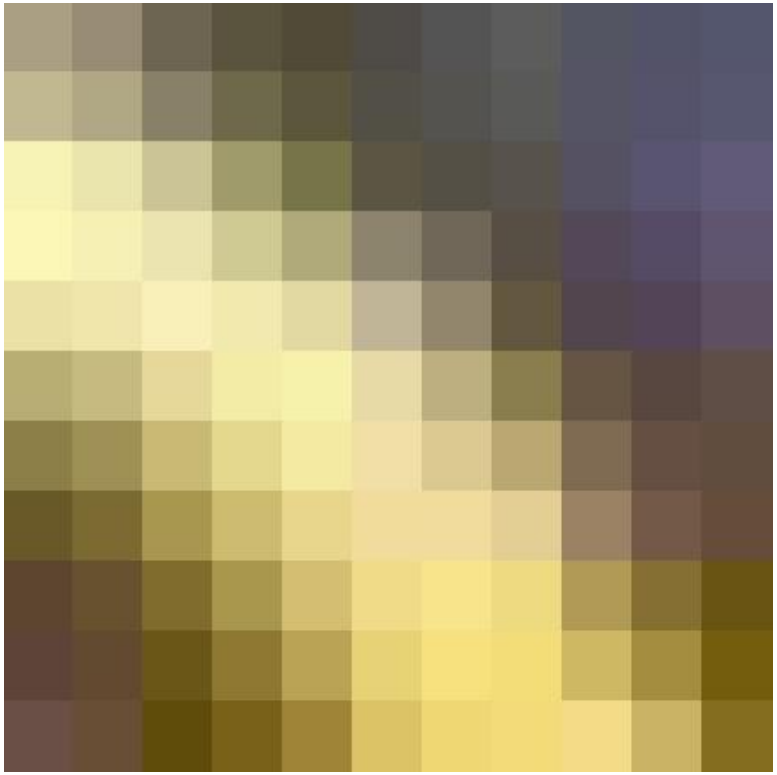
Bilinear Interpolation



5:7

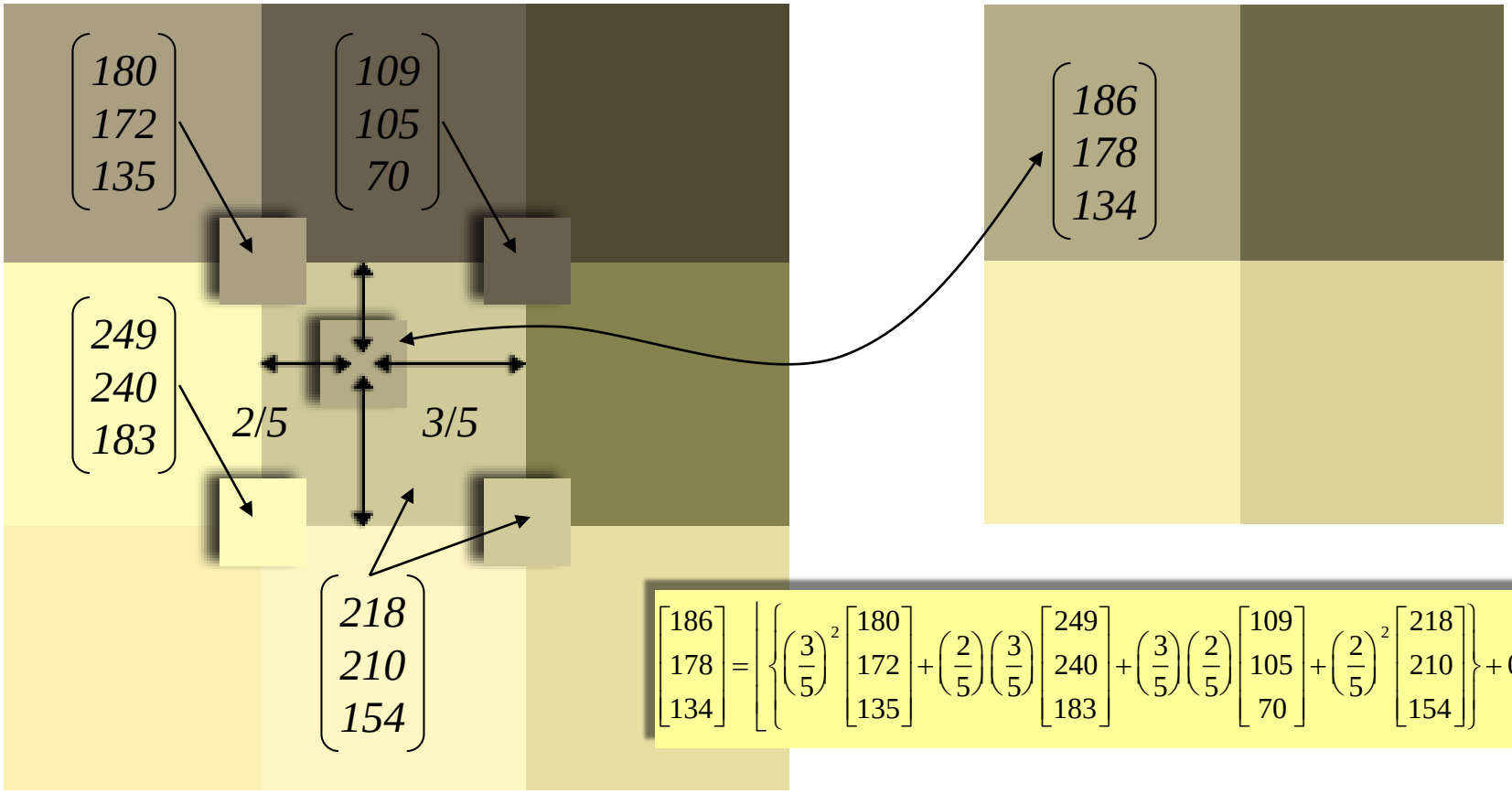


1:1



11:7

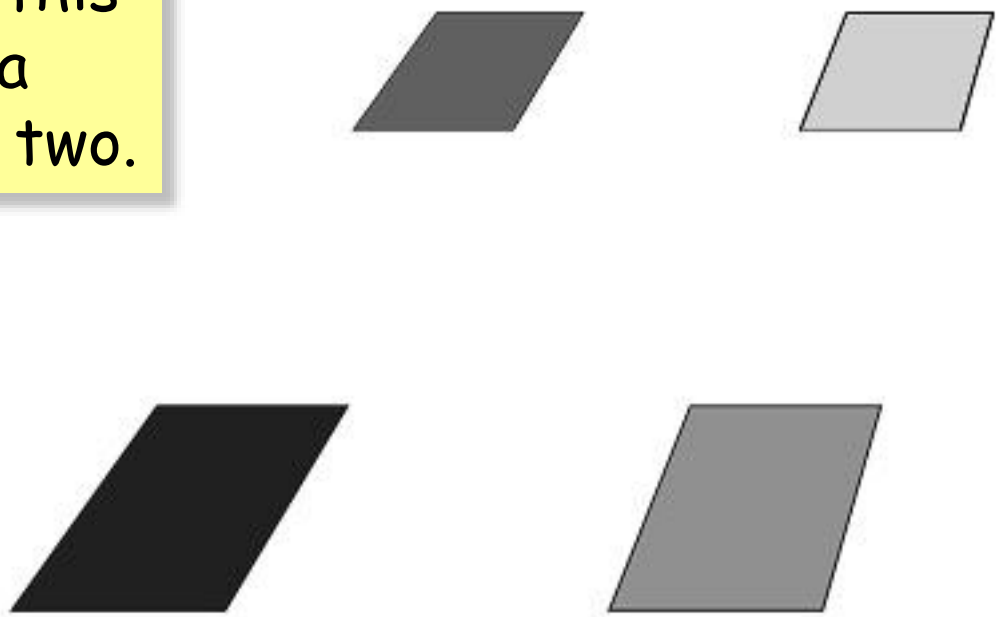
Bi-Interp Example: resize to 5/7 of original dimensions.



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

Resampling Through Bilinear Interpolation

Want to upsample this image by a factor of two.

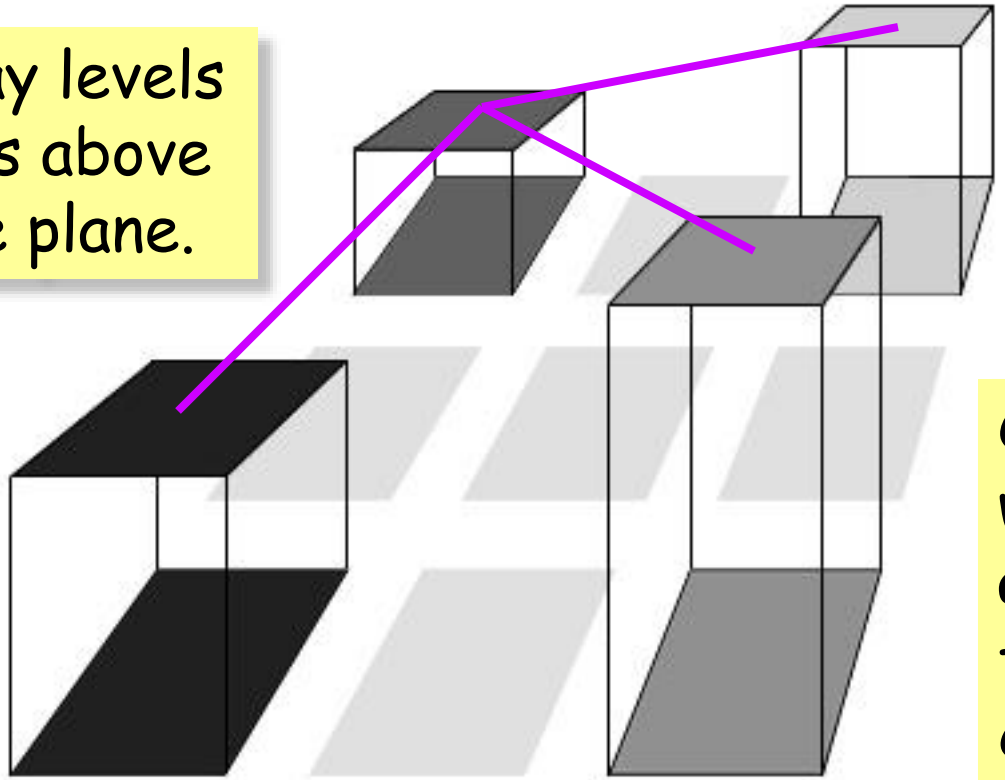


Resampling Through Bilinear Interpolation

New samples to be added at light gray locations.

Resampling Through Bilinear Interpolation

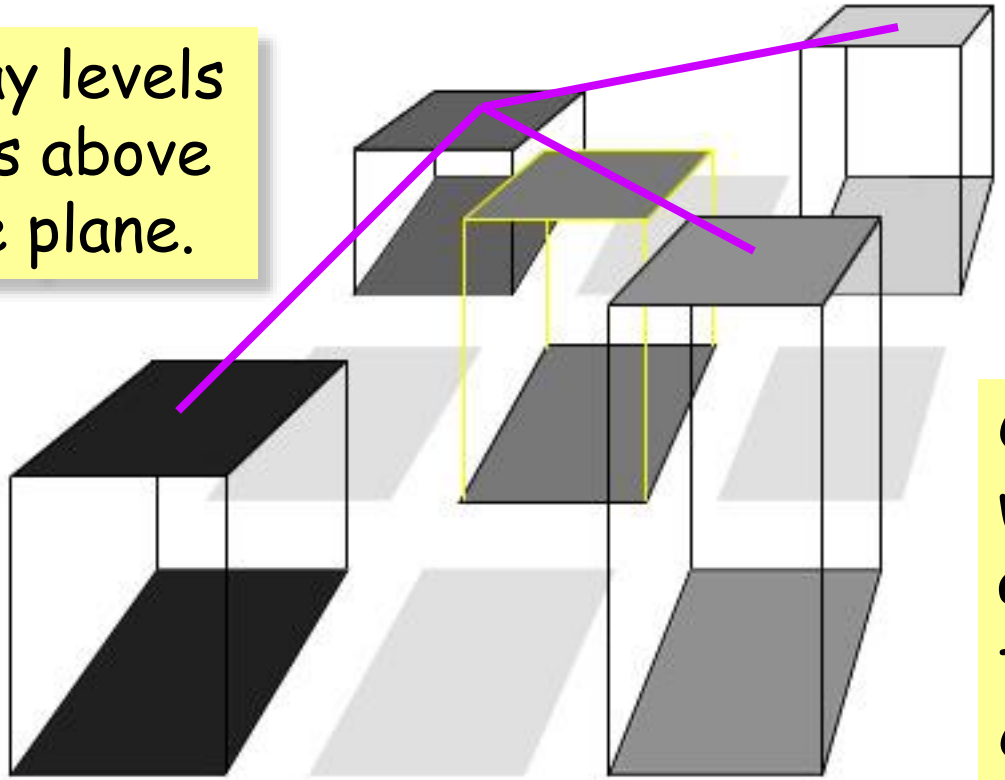
Treat gray levels as heights above the image plane.



Center = weighted average of four corners.

Resampling Through Bilinear Interpolation

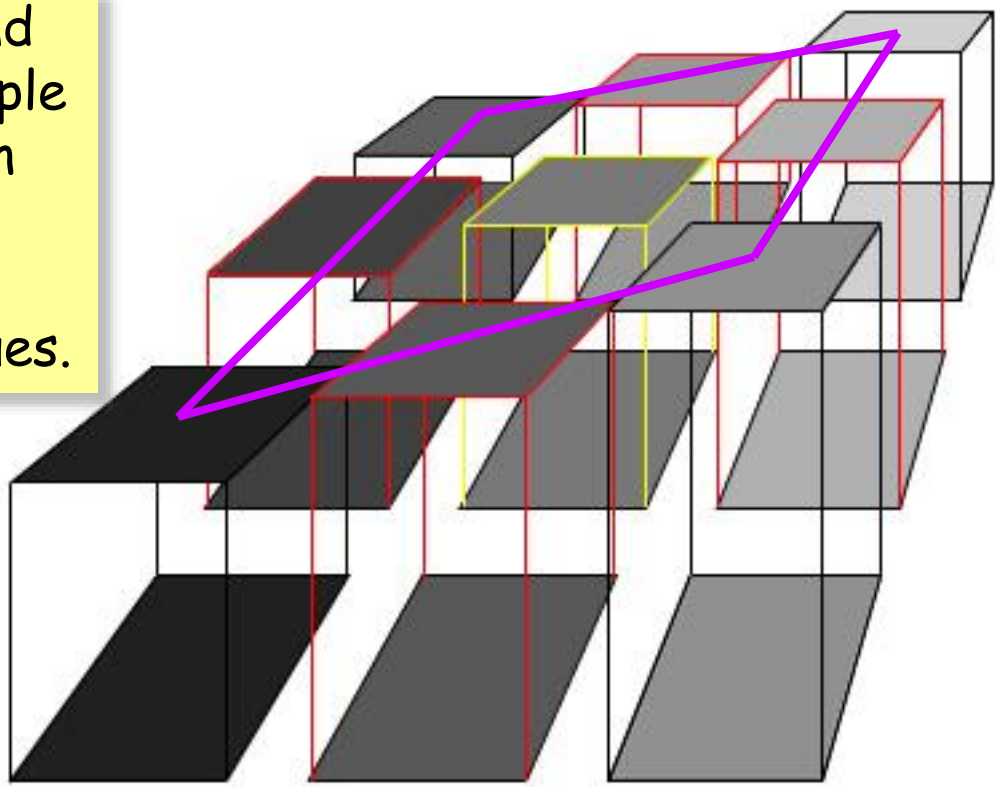
Treat gray levels as heights above the image plane.



Center = weighted average of four corners.

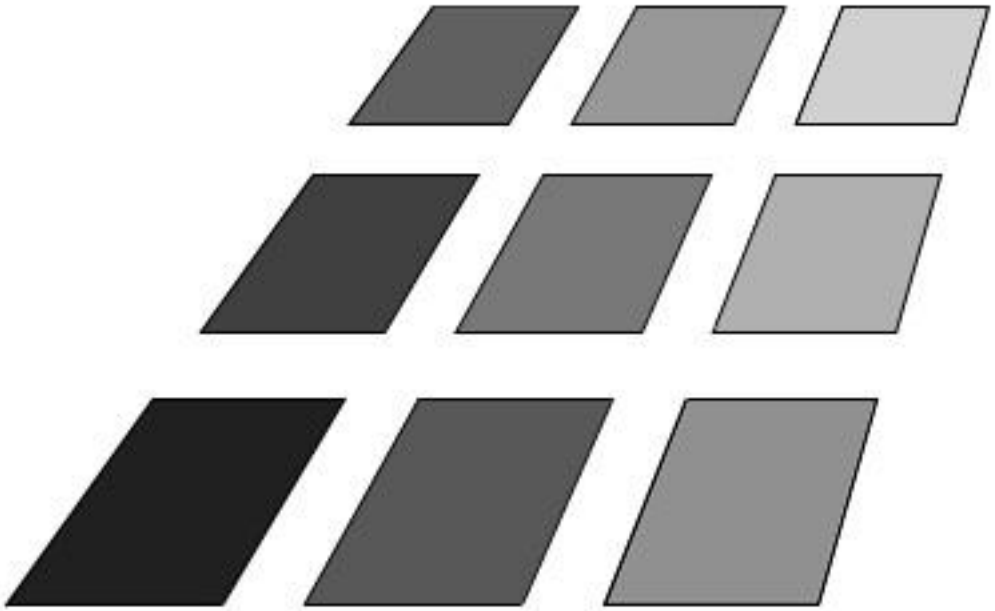
Resampling Through Bilinear Interpolation

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 …

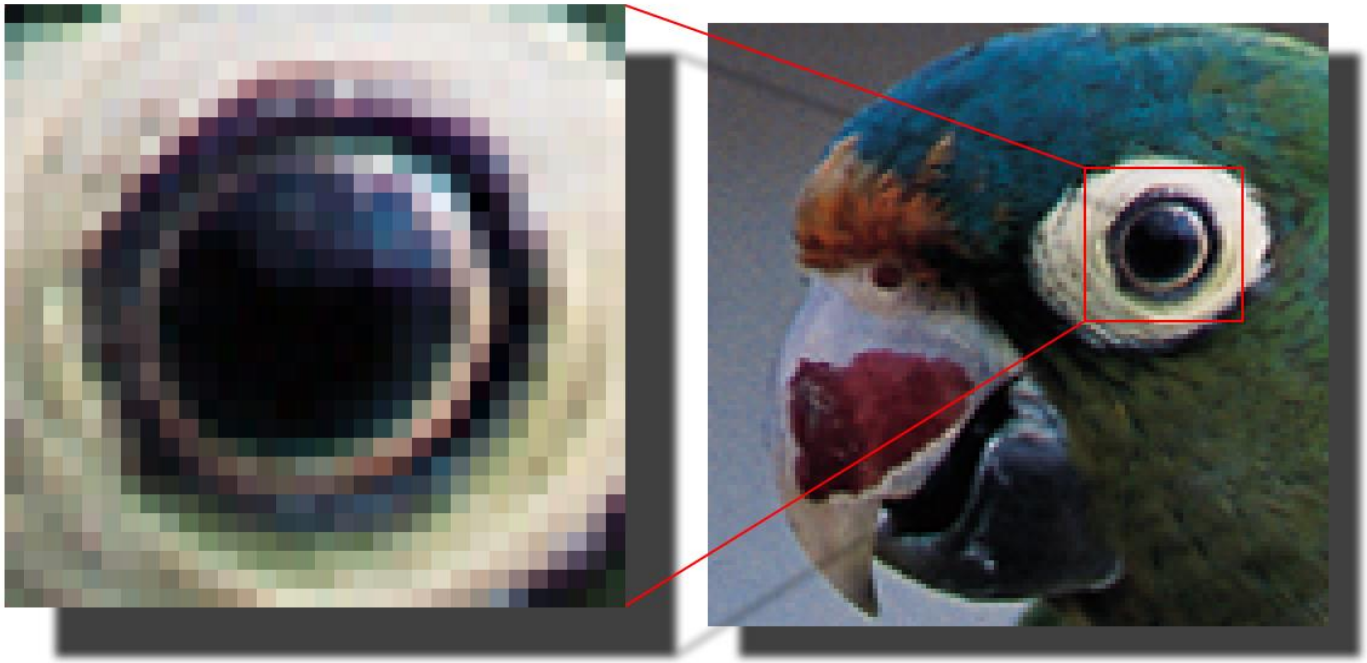
… via bilinear interpolation and compare it to a nearest neighbor enlargement.

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



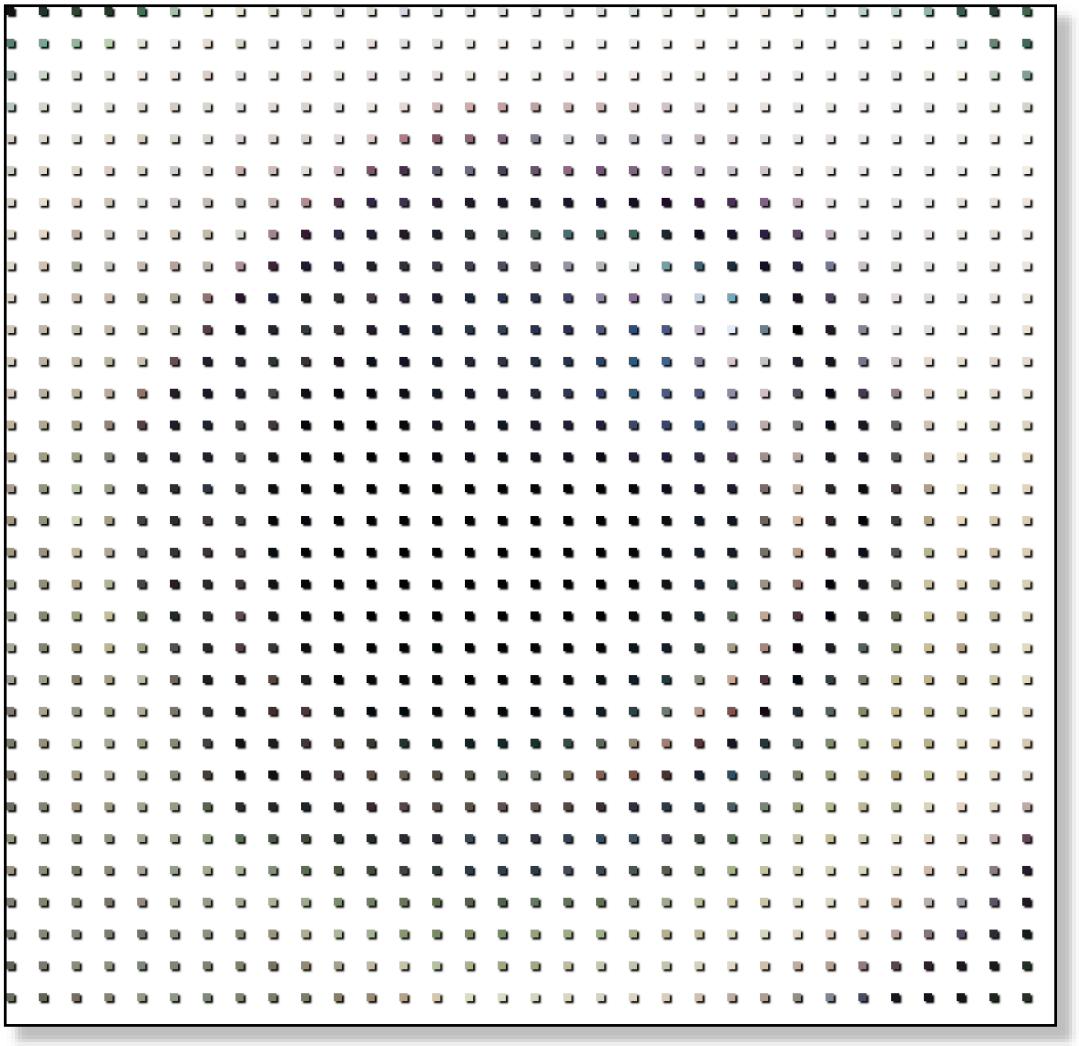
To better see what happens, we'll look at the parrot's eye.

Bilinear Interpolation

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

Bilinear Interpolation

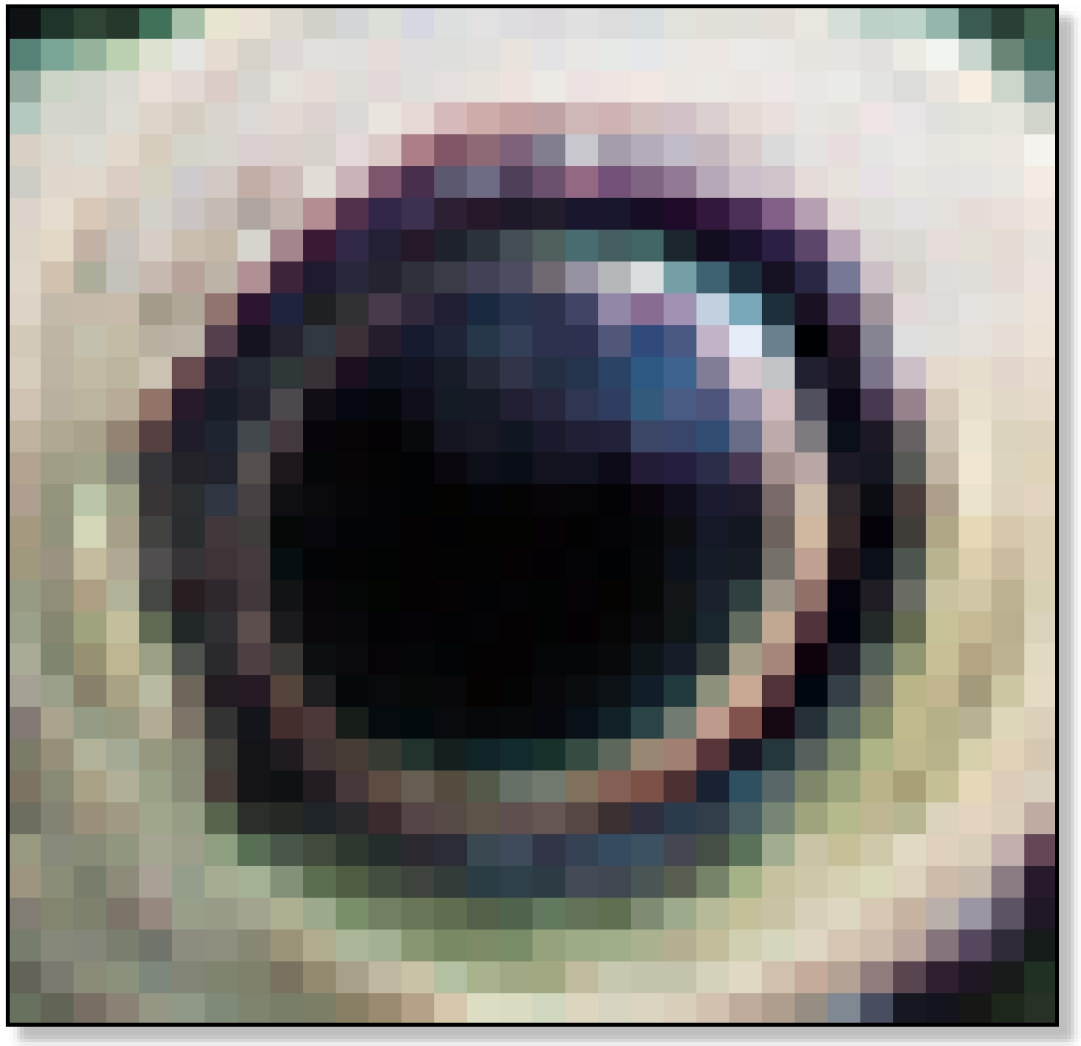
Then fill in every 4th pixel in every 4th row with the original values.



Bilinear Interpolation

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

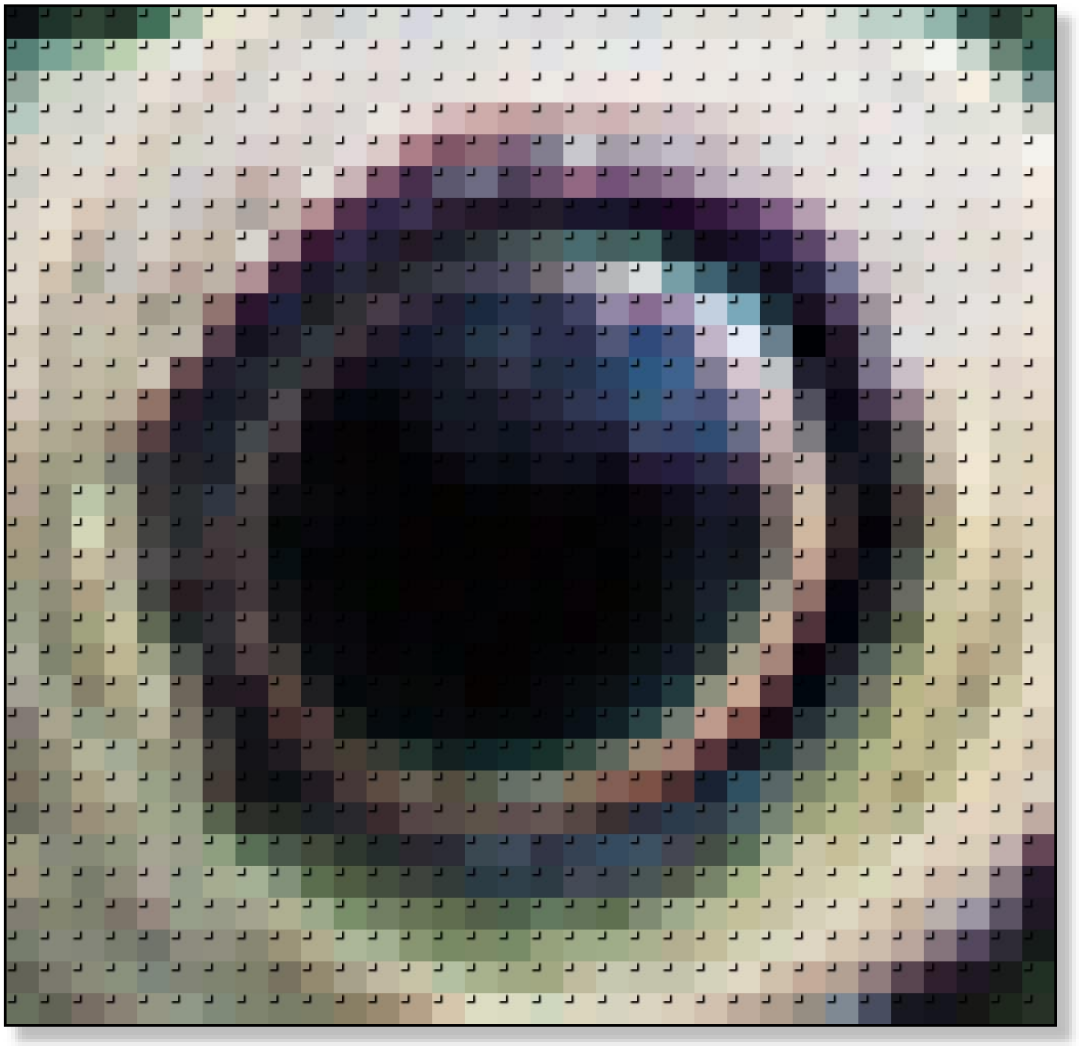
4x replication



Bilinear Interpolation

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

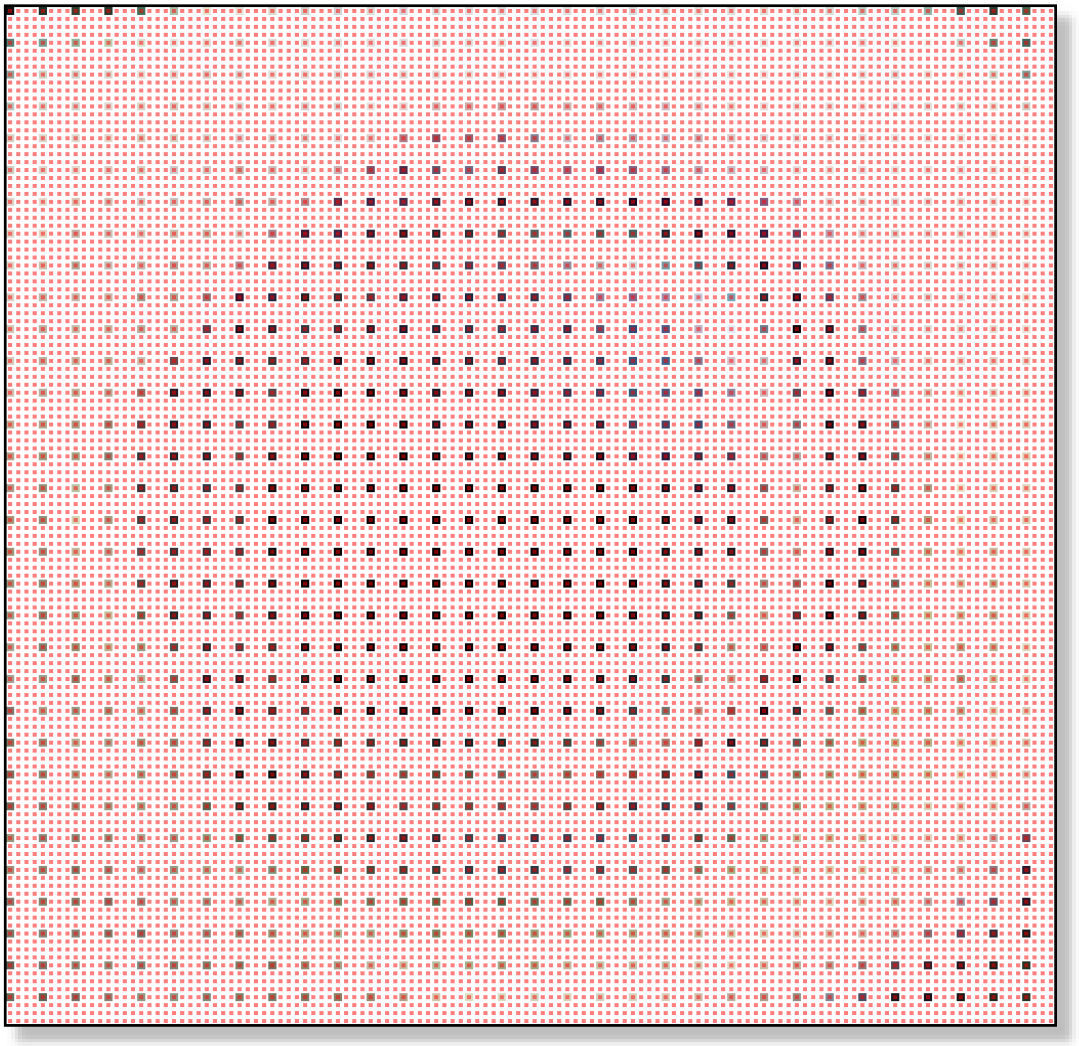
4x replication with the original pixels overlaid.



Bilinear Interpolation

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

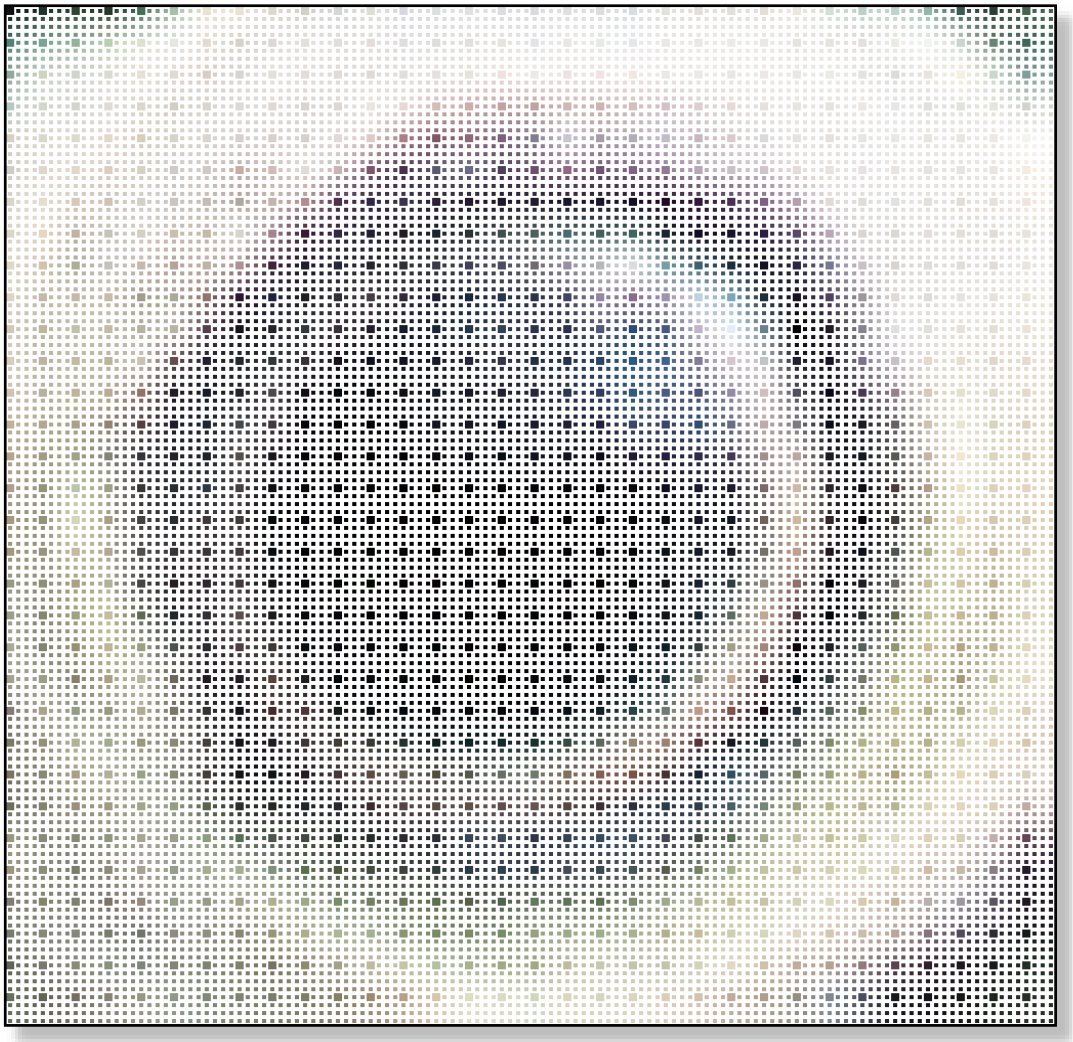
Each pink dot is a pixel location.



Bilinear Interpolation

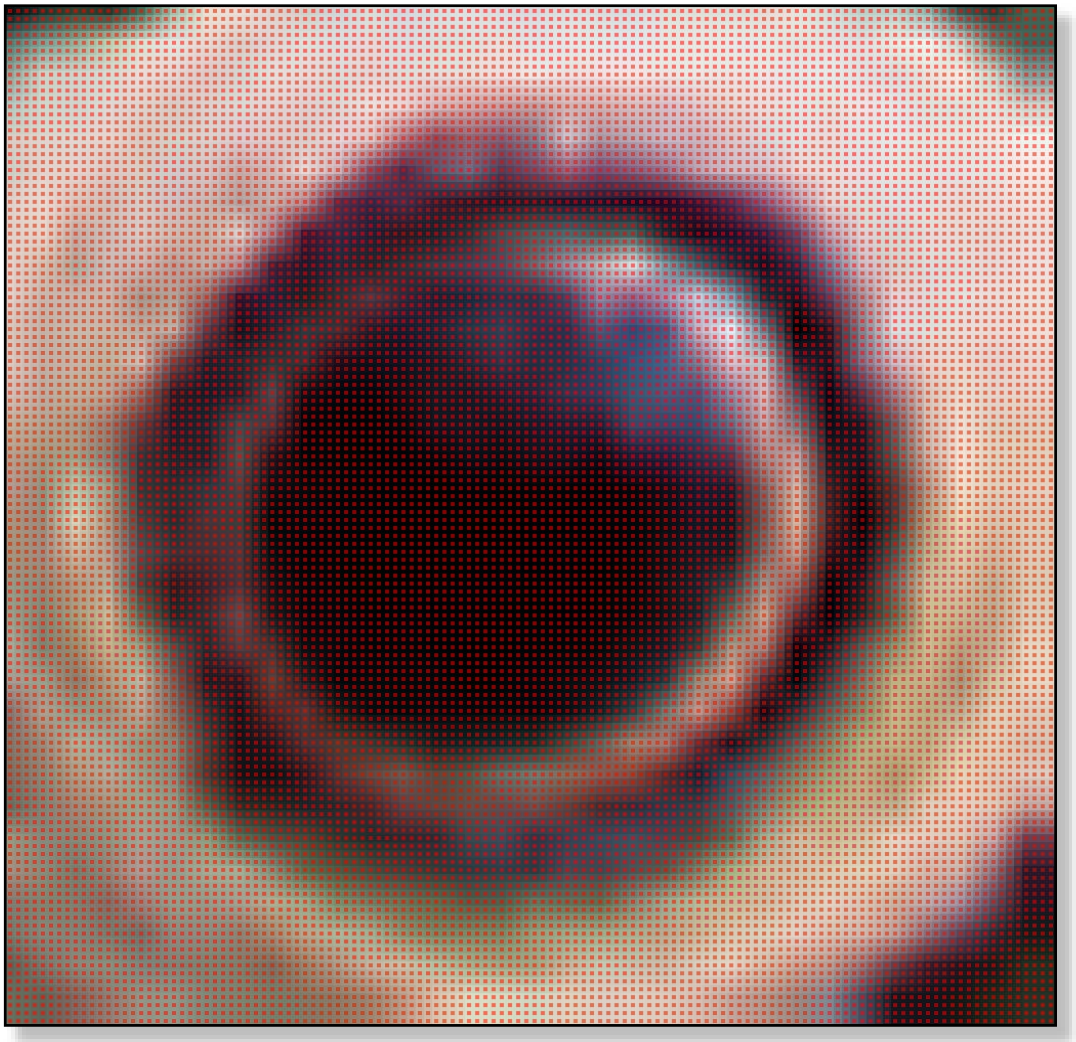
Next you want to fill in the other 15 pixels in each block with the intermediate values.

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.

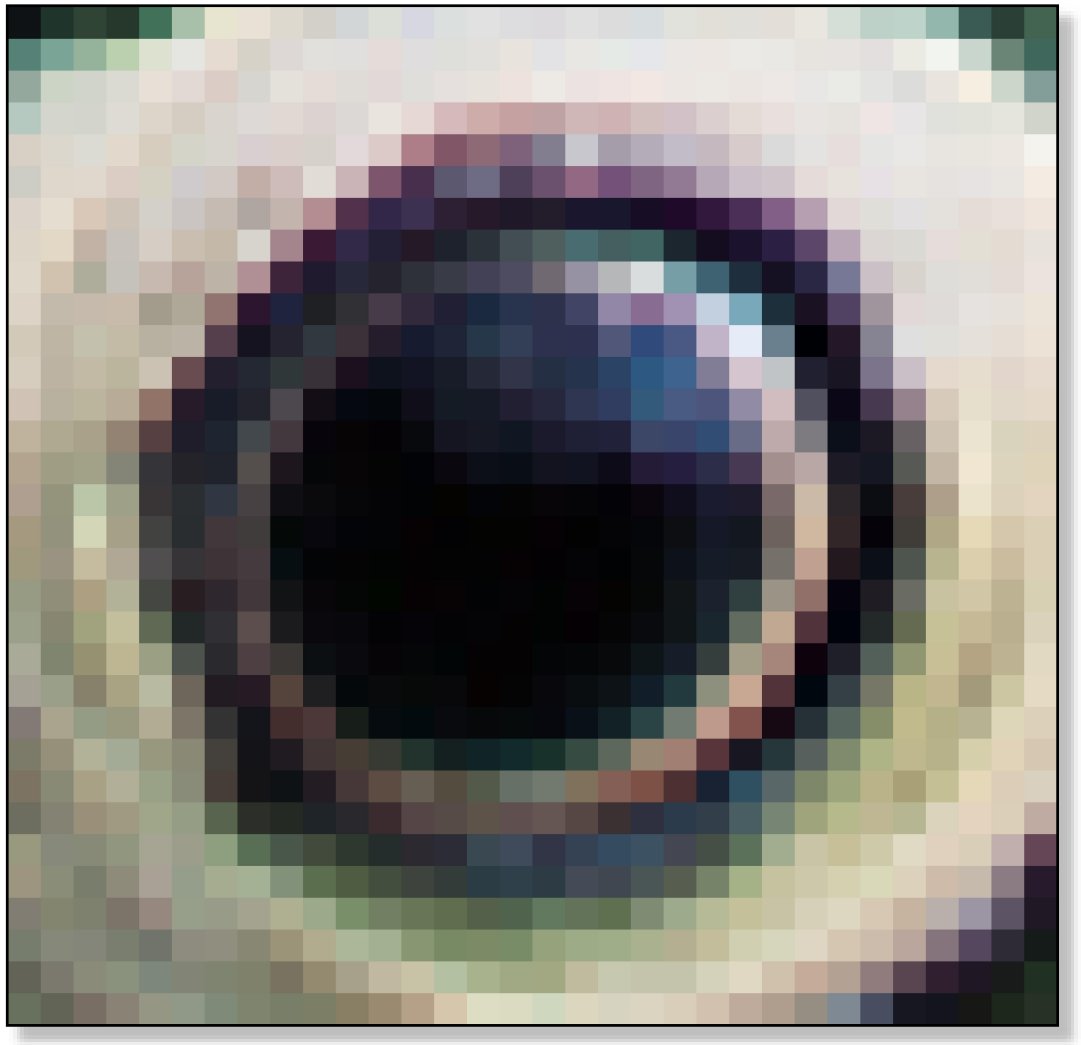
Bilinear Interpolation

The result:

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

Pixel Replication

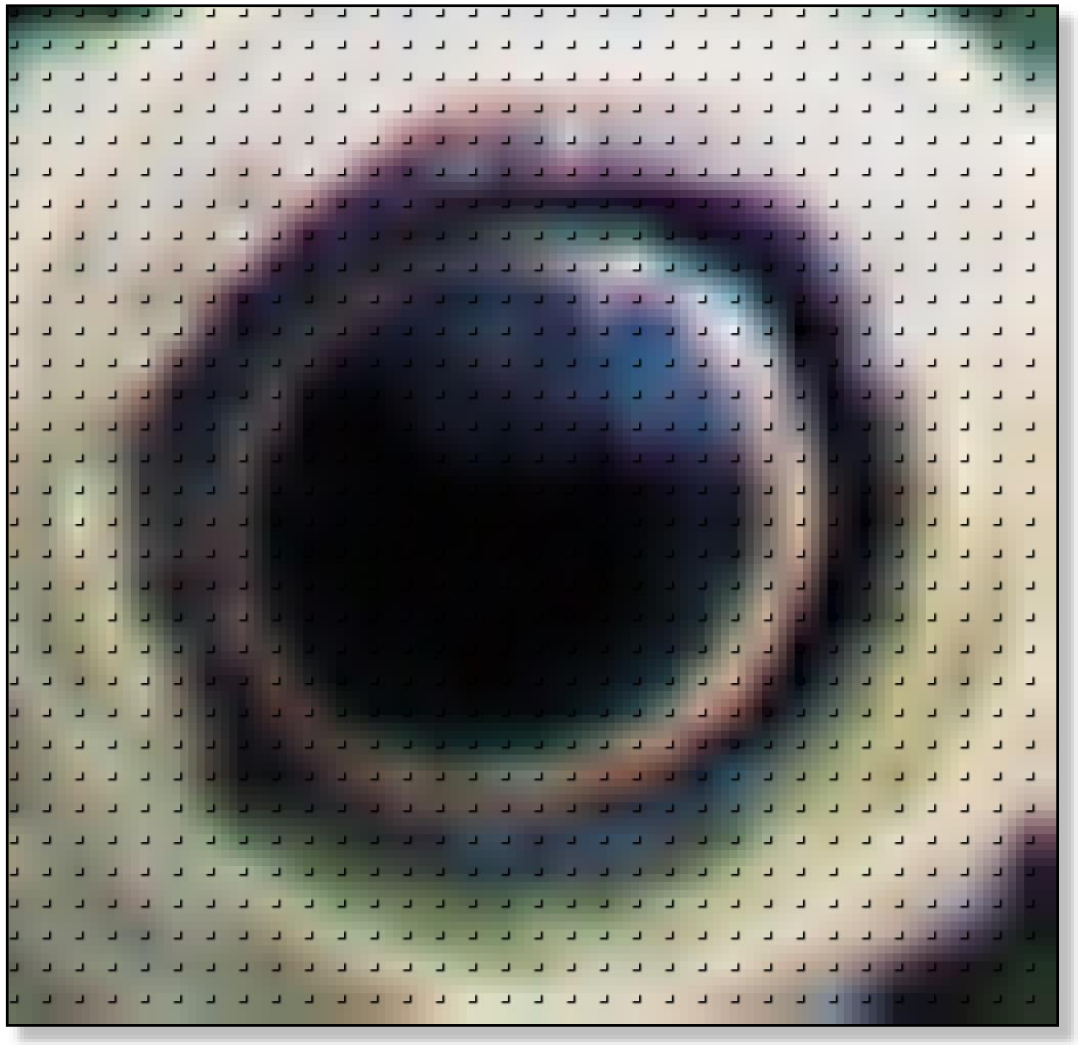
The result:



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

Bilinear Interpolation

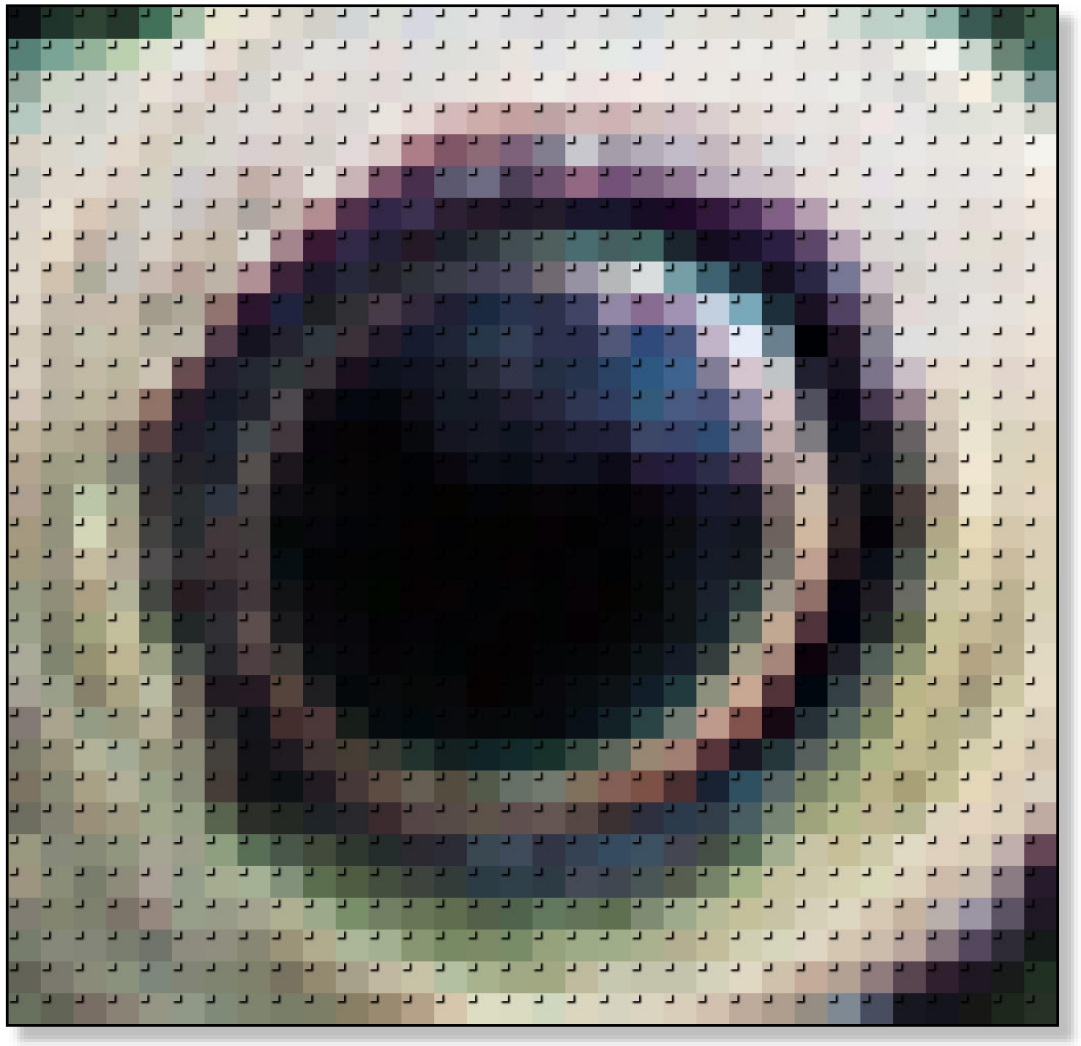
Result with original pixels marked:



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

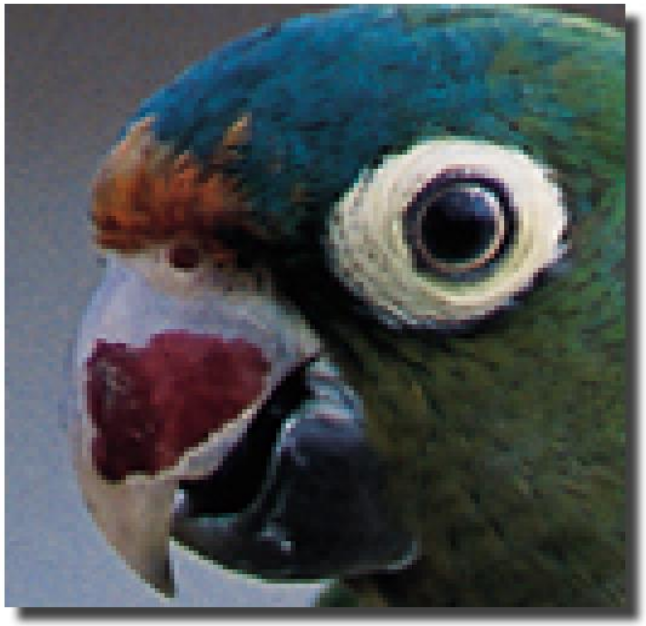
Pixel Replication

Result with original pixels marked:



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

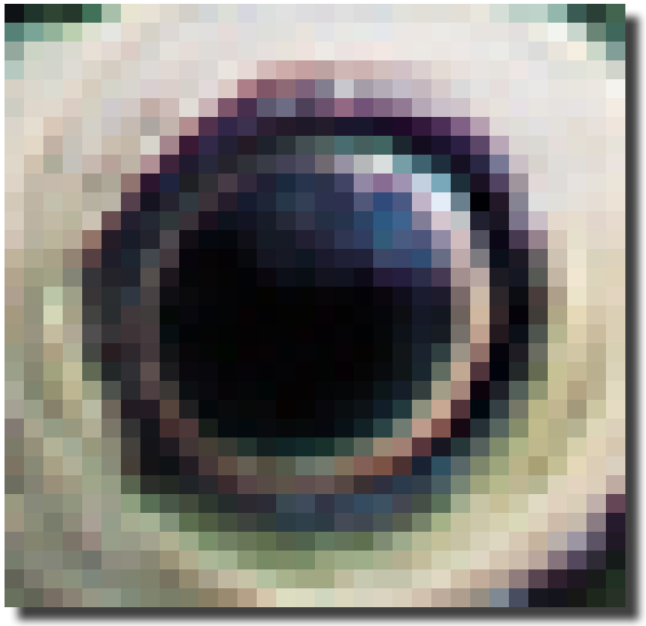
Pixel Replication vs. Bilinear Interpolation



Pixel replication

Bilinear interpolation

Pixel Replication vs. Bilinear Interpolation

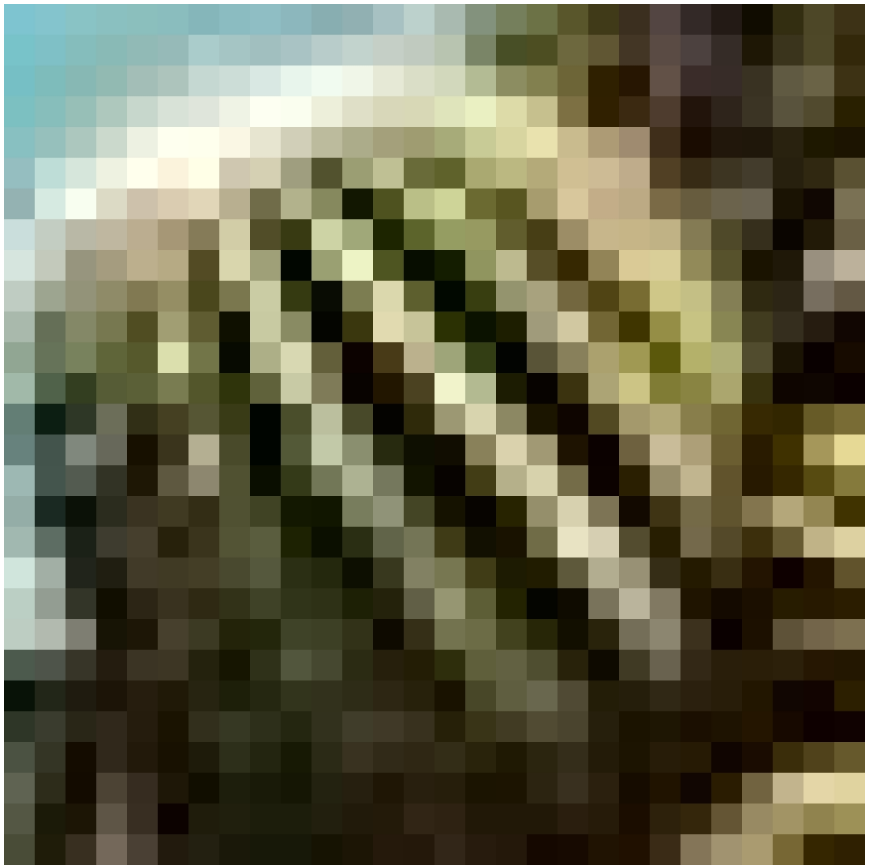


Pixel replication

Bilinear interpolation

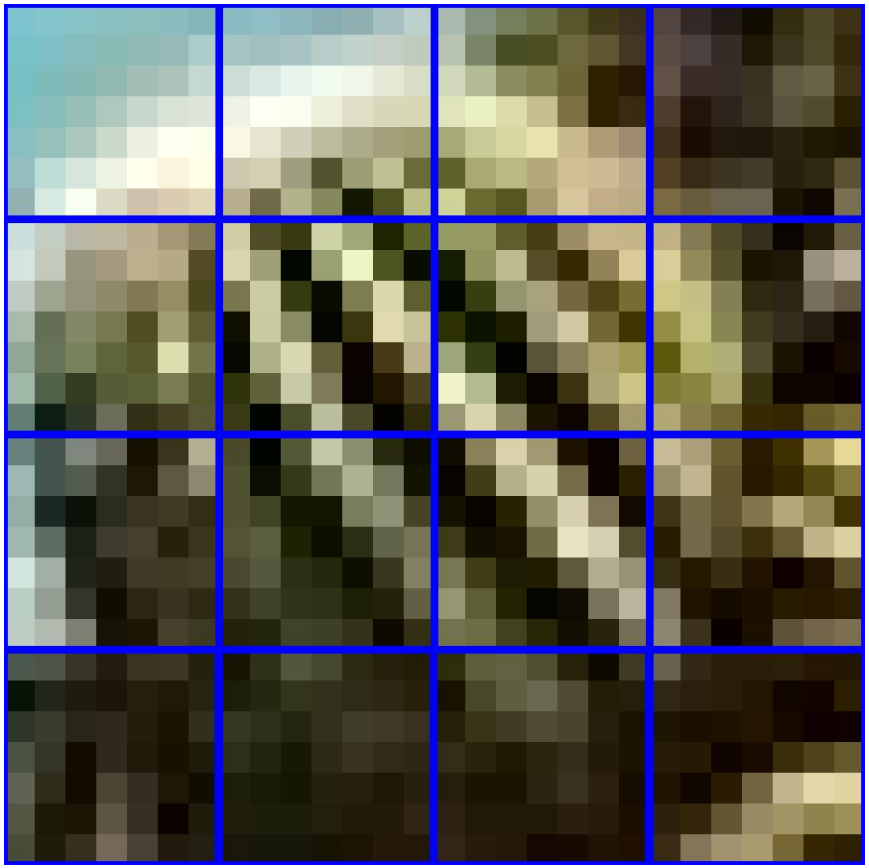
Resampling Through Bilinear Interpolation

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



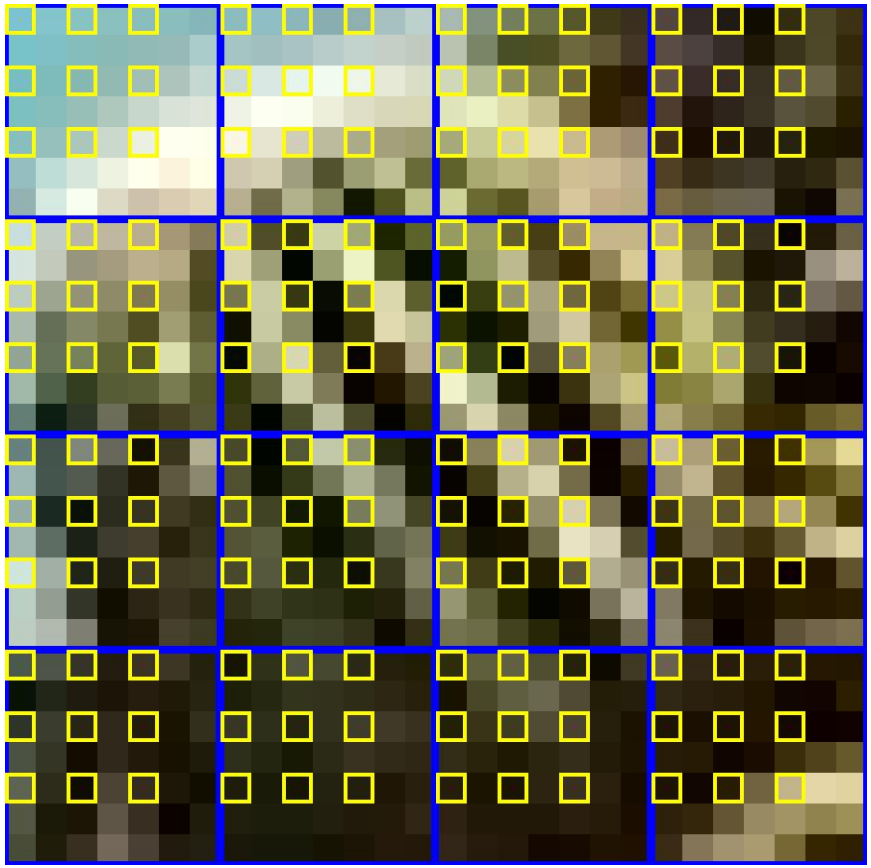
Resampling Through Bilinear Interpolation

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



Resampling Through Bilinear Interpolation

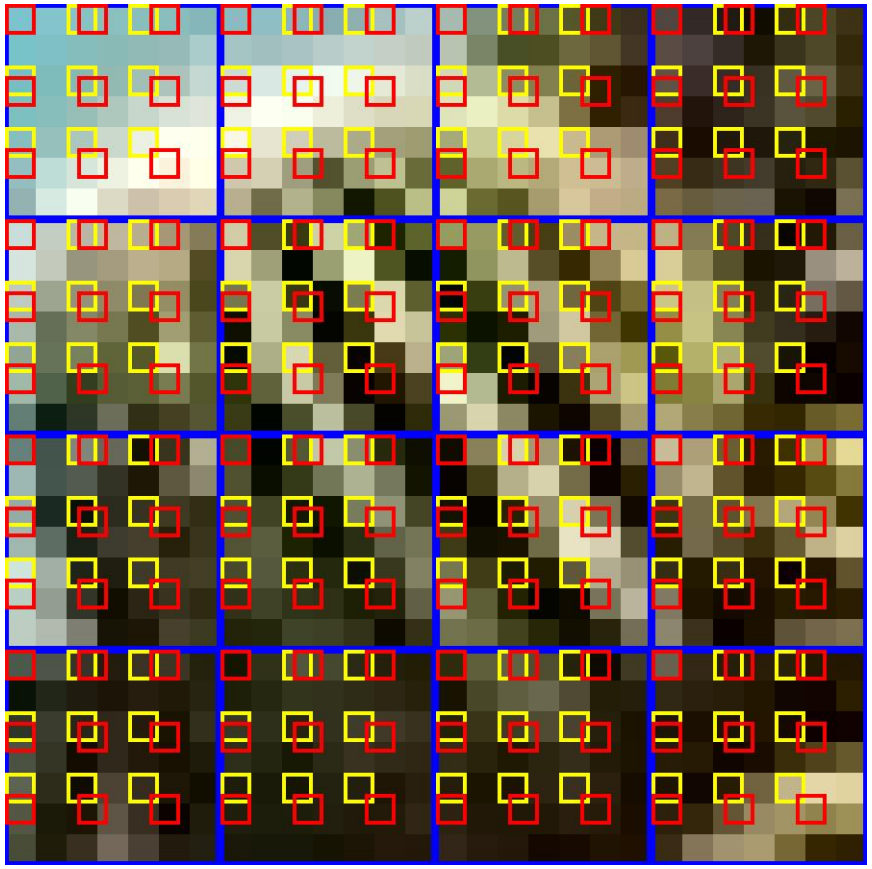
For nearest neighbor sampling the 9 pixel locations correspond to pixel locations in the original image.



Nearest neighbor selected pixels outlined in yellow.

Resampling Through Bilinear Interpolation

In bilinear interpolation the 9 pixel locations are distributed evenly.

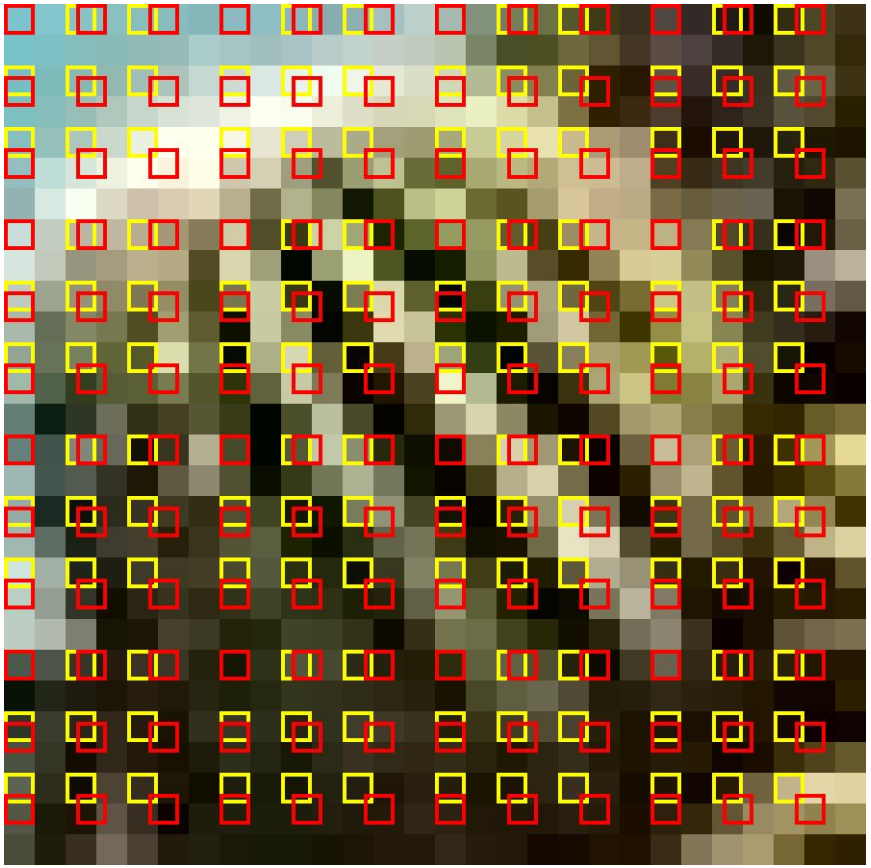


Nearest neighbor selected pixels outlined in yellow.

Bilinear interp. pixels locations outlined in red.

Resampling Through Bilinear Interpolation

In bilinear interpolation the 9 pixel locations are distributed evenly.

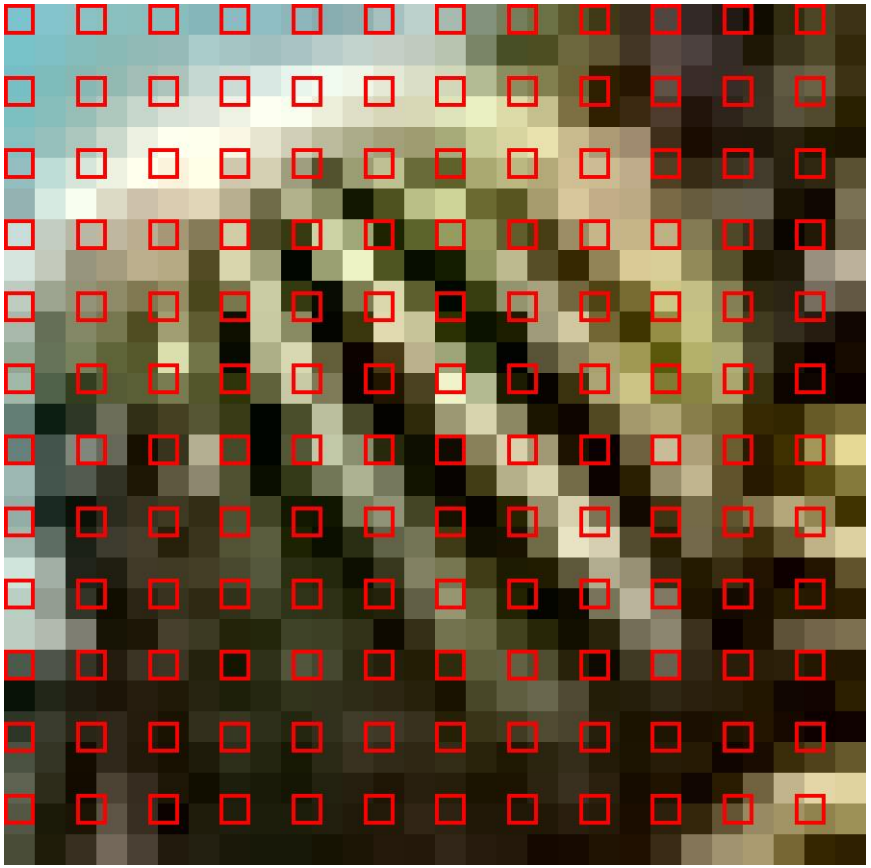


Nearest neighbor selected pixels outlined in yellow.

Bilinear interp. pixels locations outlined in red.

Resampling Through Bilinear Interpolation

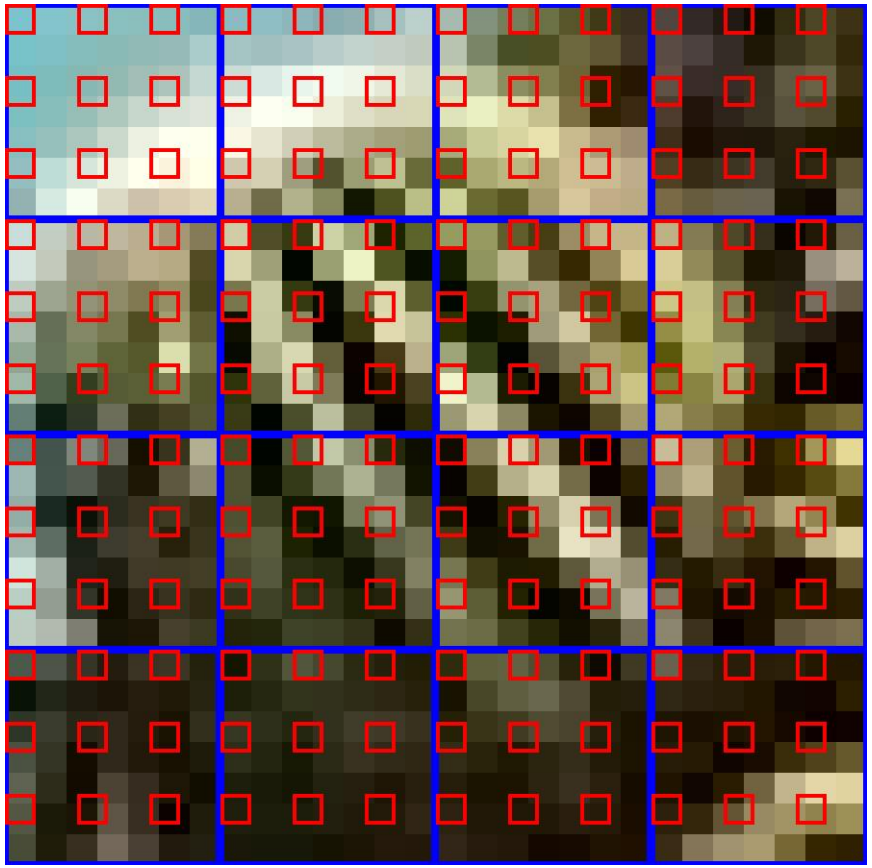
In bilinear interpolation the 9 pixel locations are distributed evenly.



Notice that the locations overlap pixels in the original image.

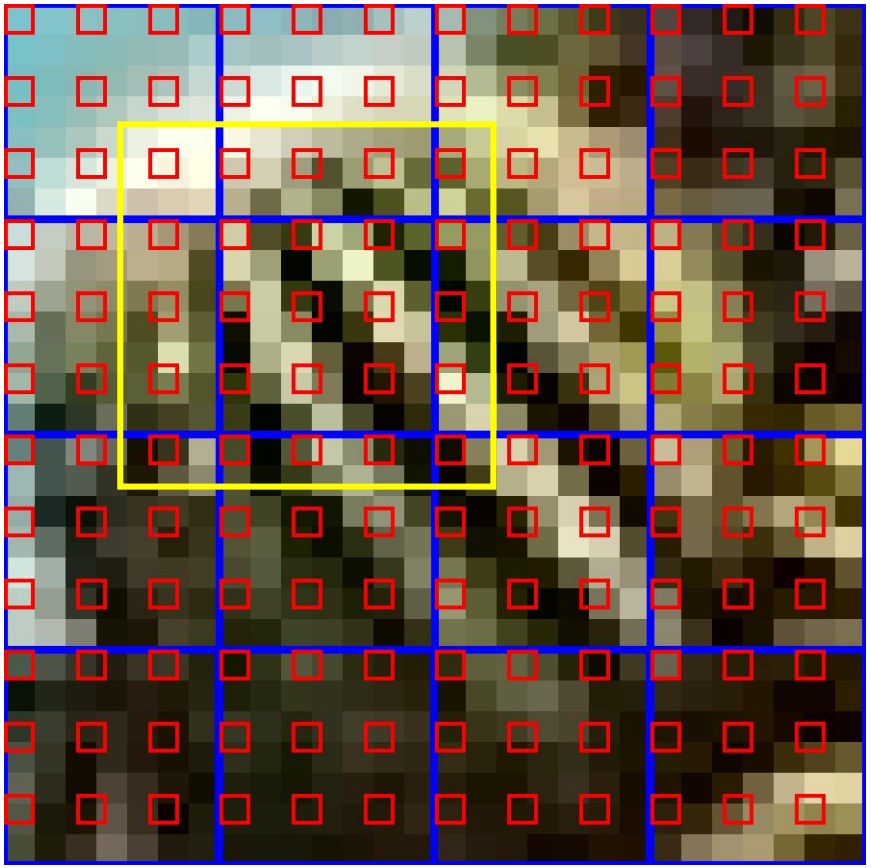
Resampling Through Bilinear Interpolation

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

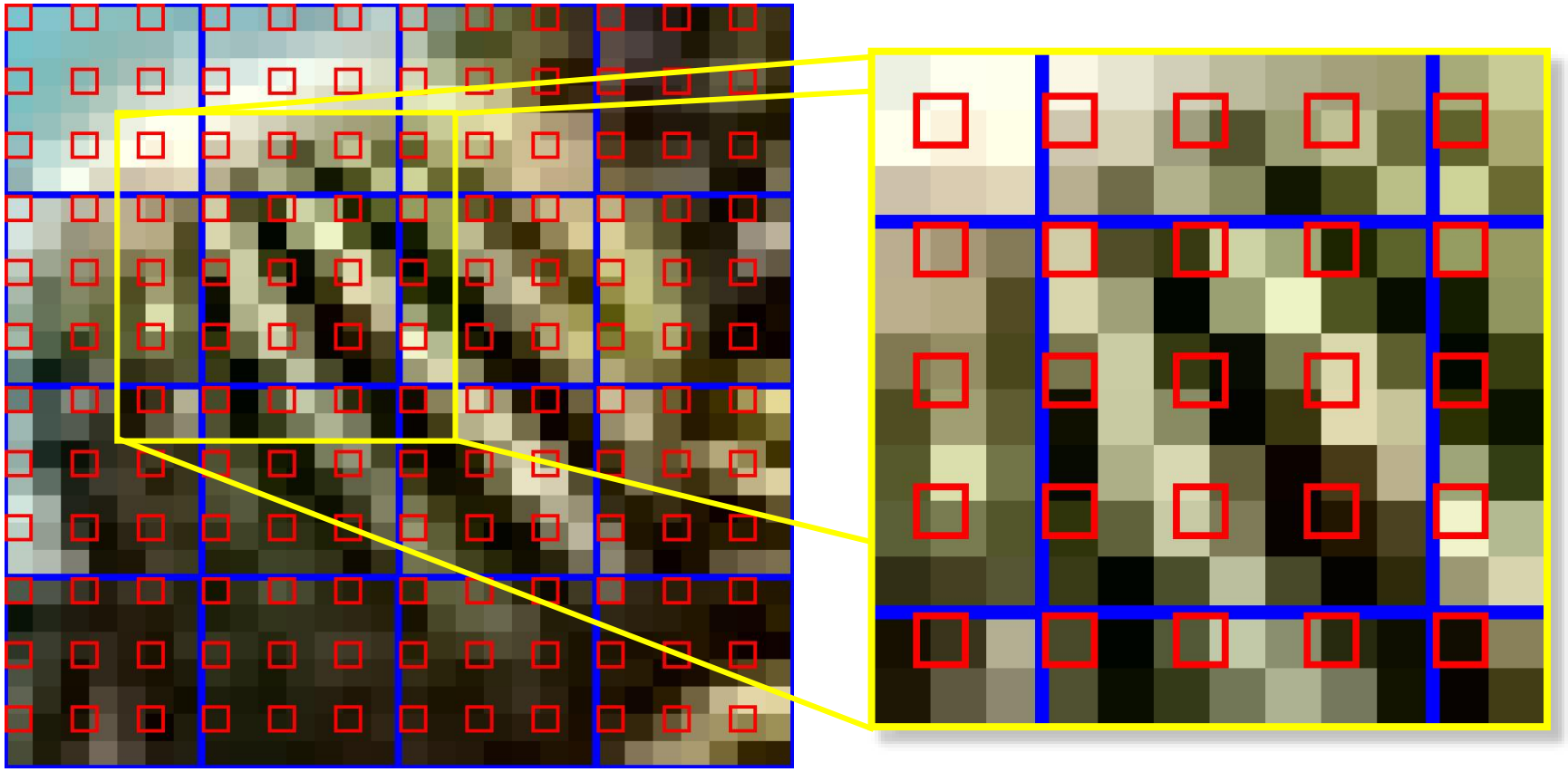


Resampling Through Bilinear Interpolation

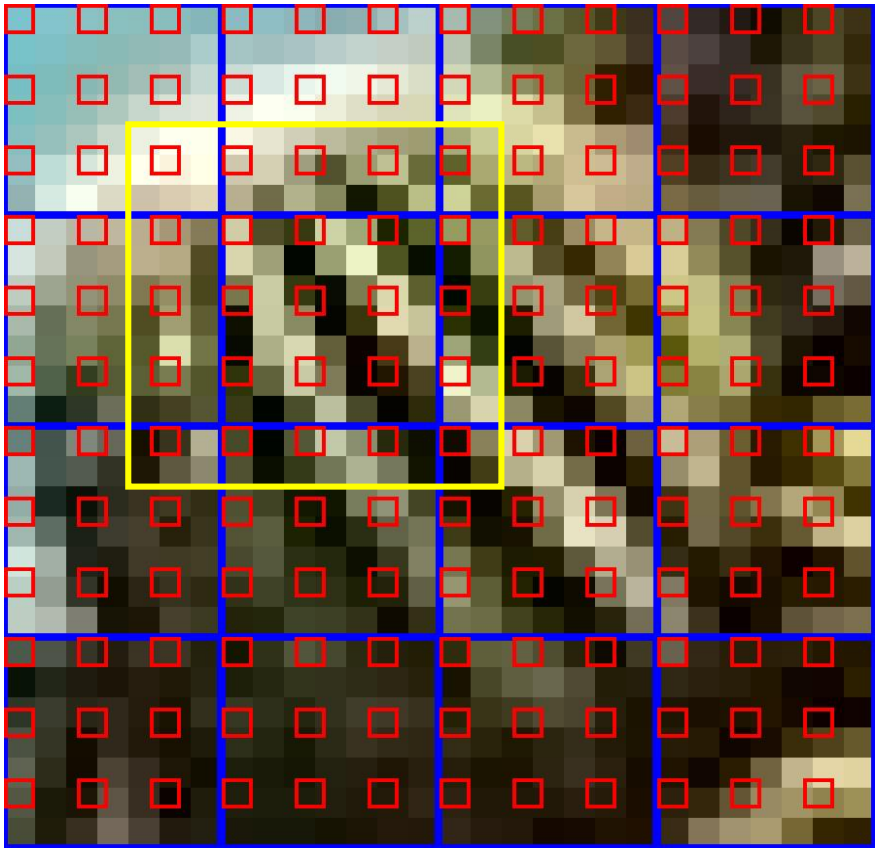
Examine one section in detail.

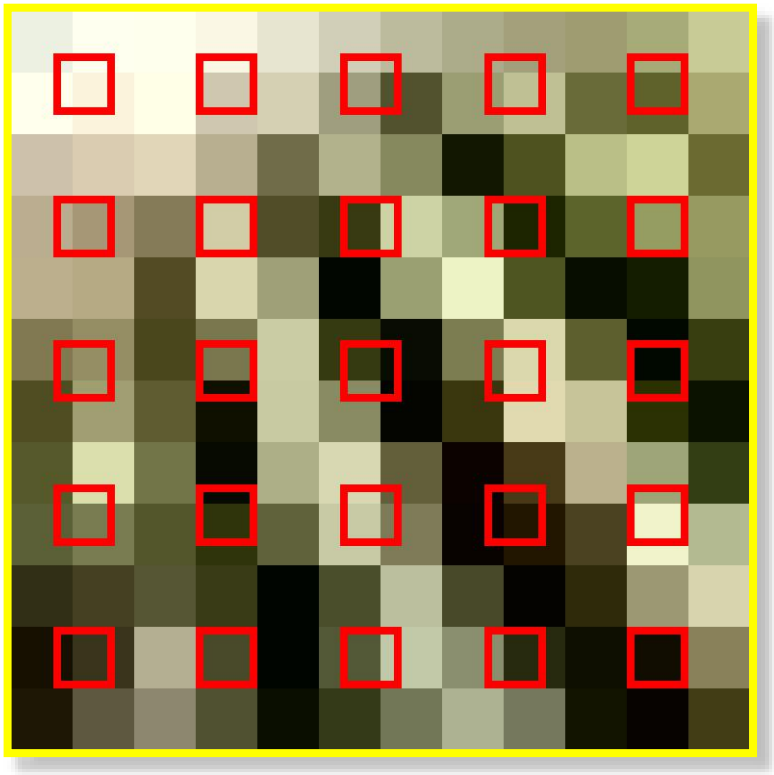


Resampling Through Bilinear Interpolation

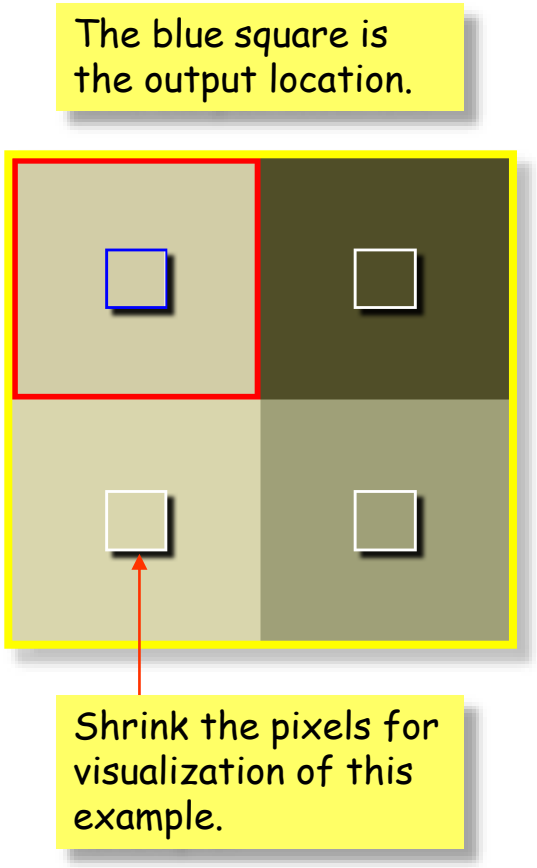


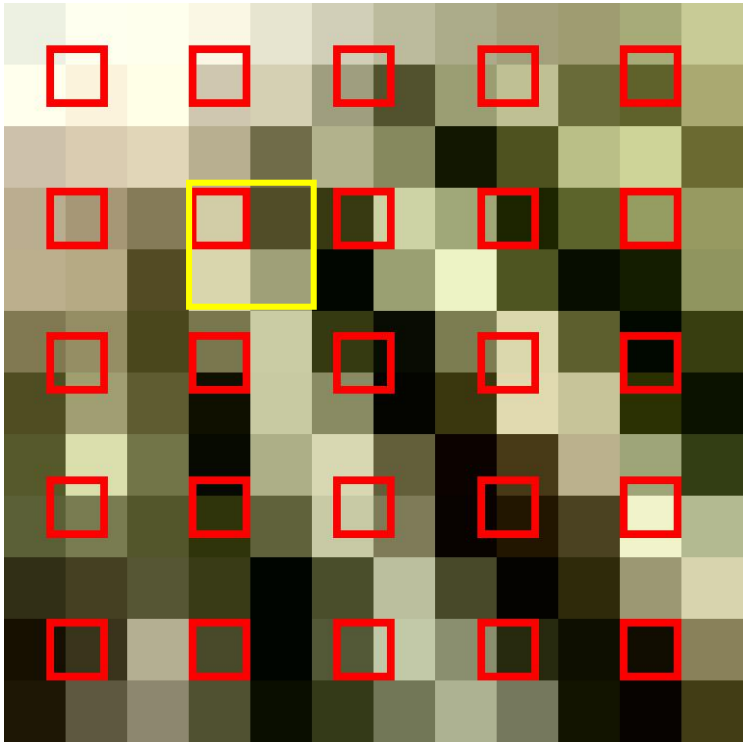
Resampling Through Bilinear Interpolation



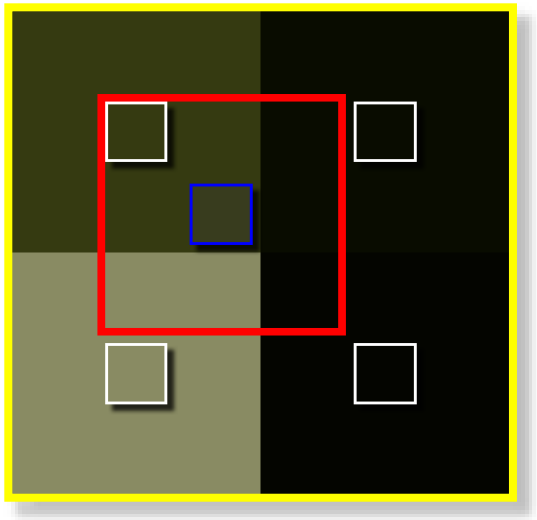


Resampling Through Bilinear Interpolation

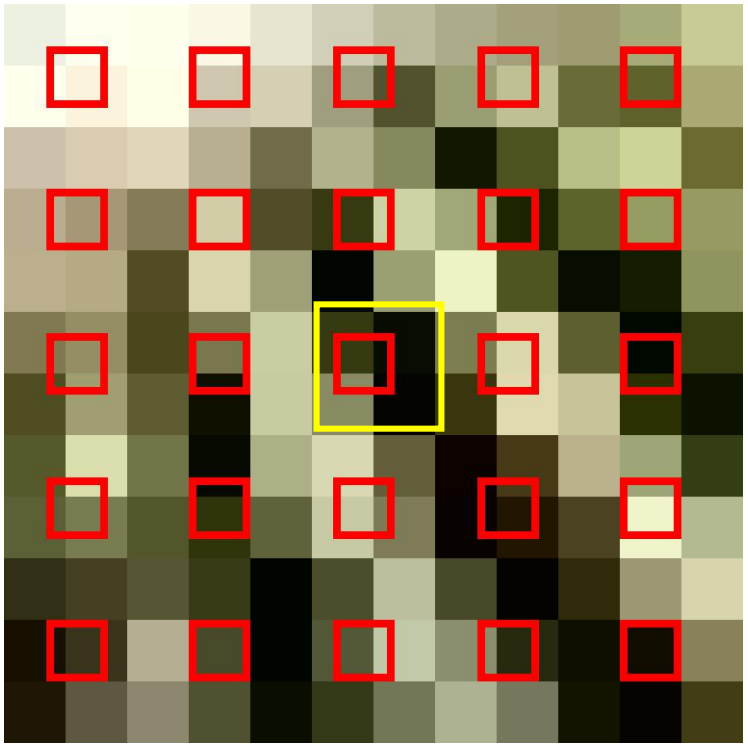




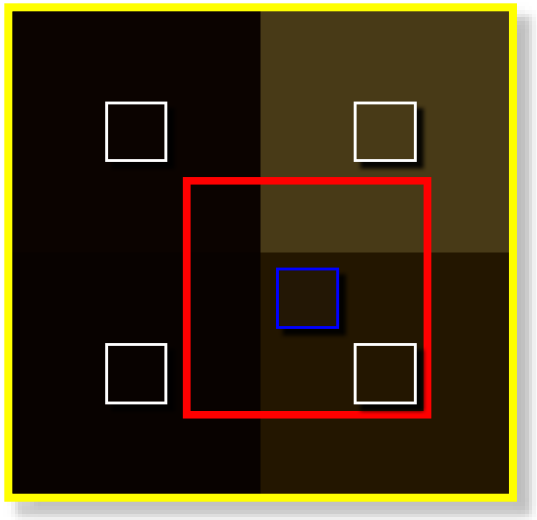
Resampling Through Bilinear Interpolation



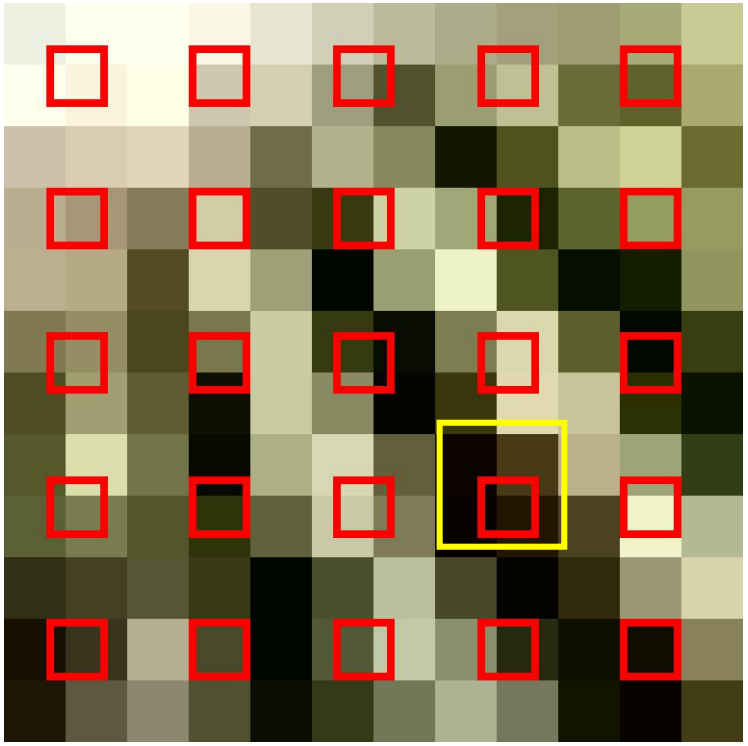
The blue square is the output location.



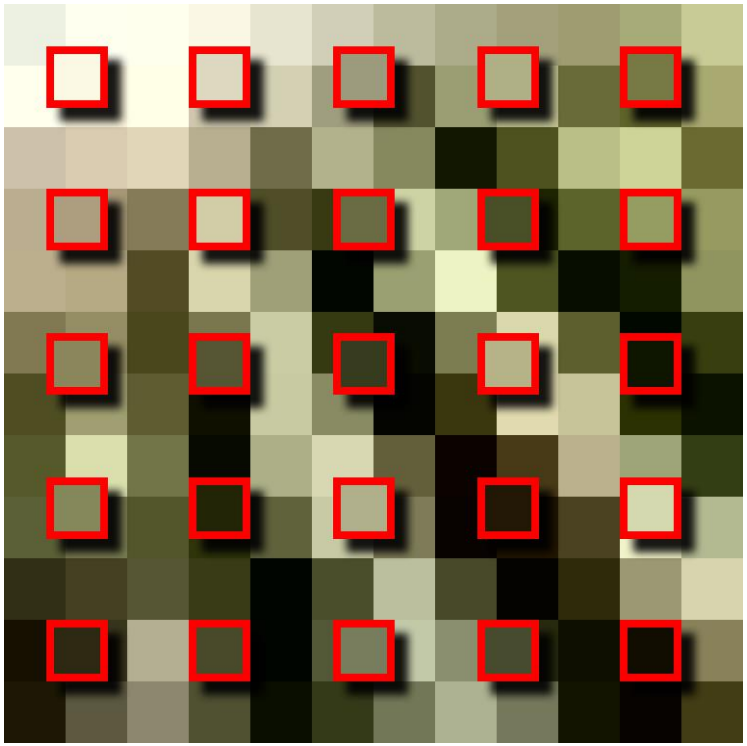
Resampling Through Bilinear Interpolation



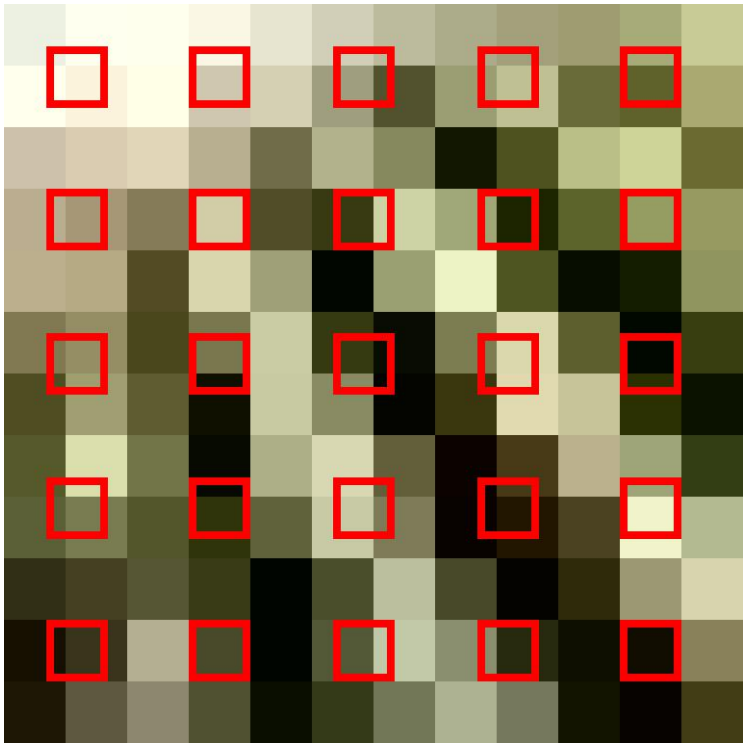
The blue square is the output location.



Resampling Through Bilinear Interpolation

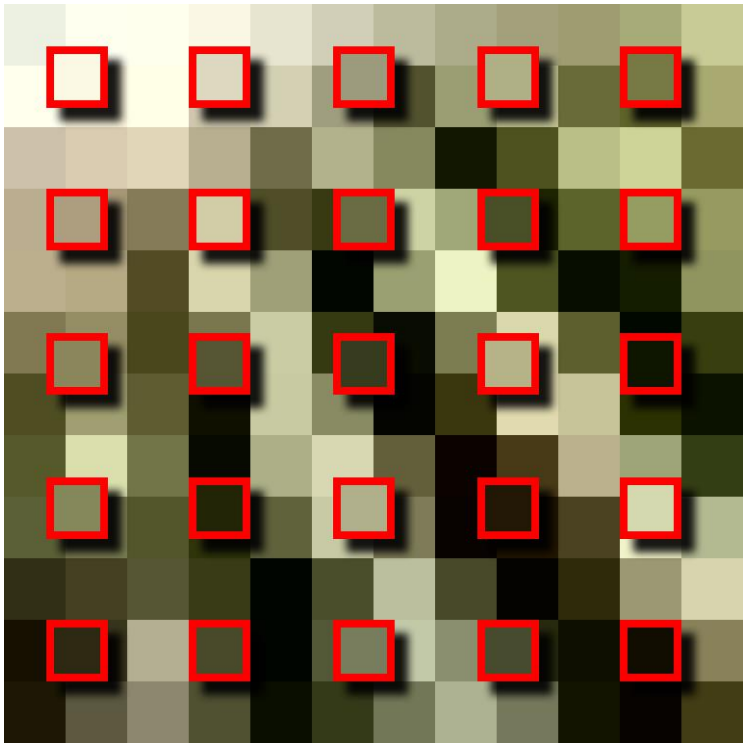


locs & colors of new pixels



locations of new pixels

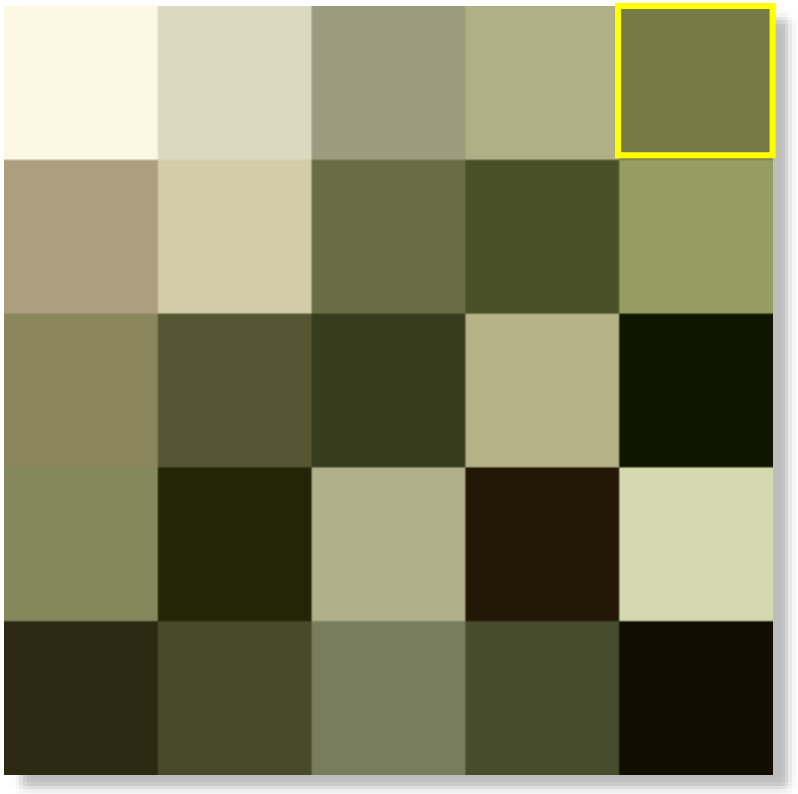
Resampling Through Bilinear Interpolation

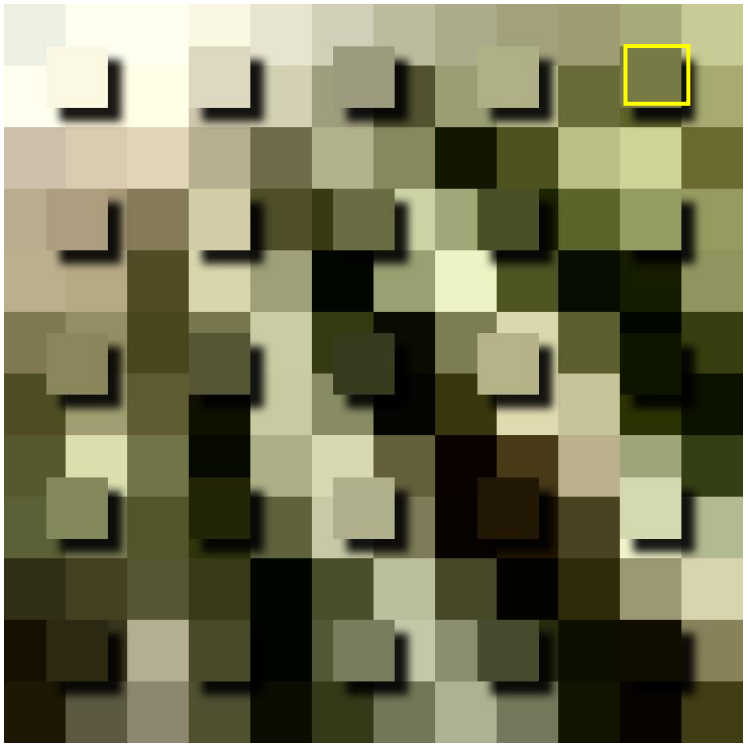


locs & colors of new pixels

locs & colors of new pixels

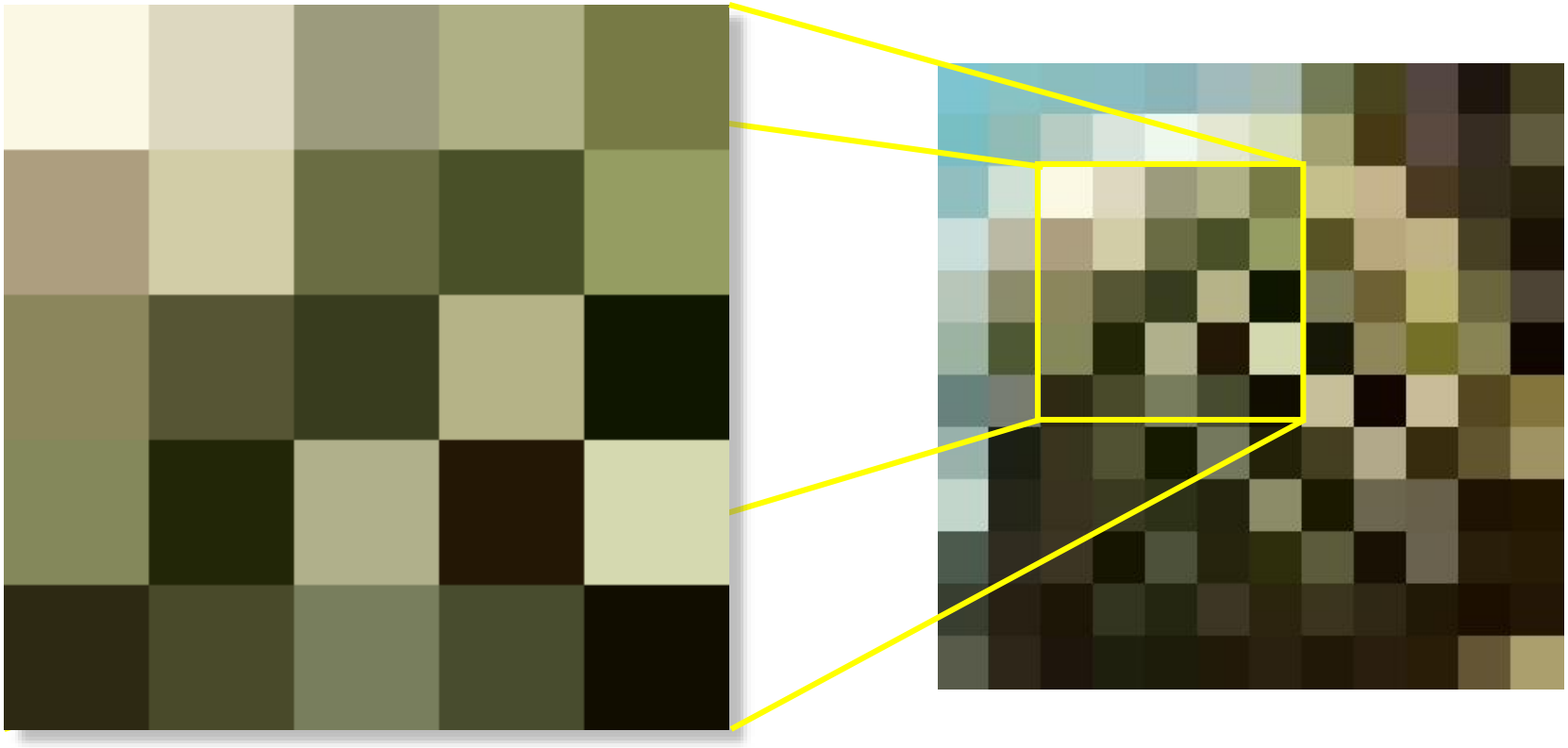
Resampling Through Bilinear Interpolation



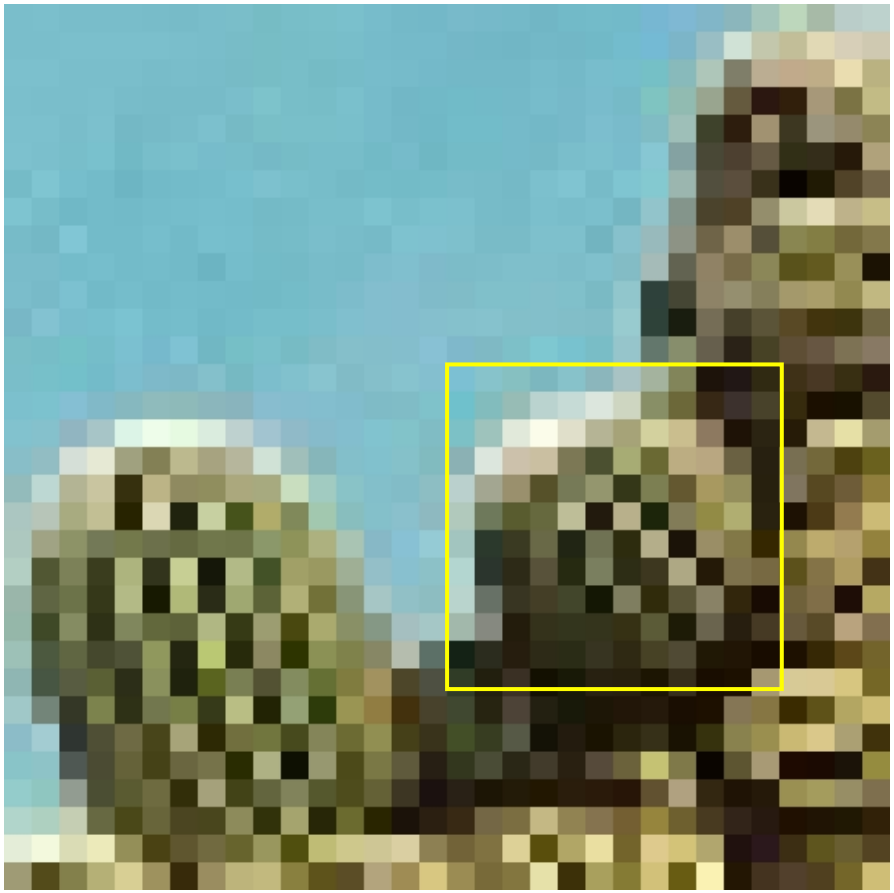


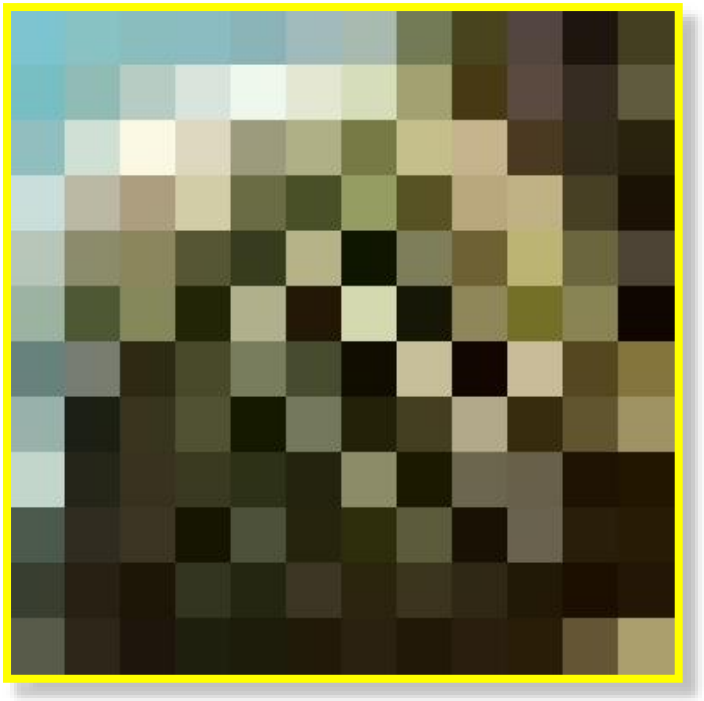
New image from new pixels

Resampling Through Bilinear Interpolation

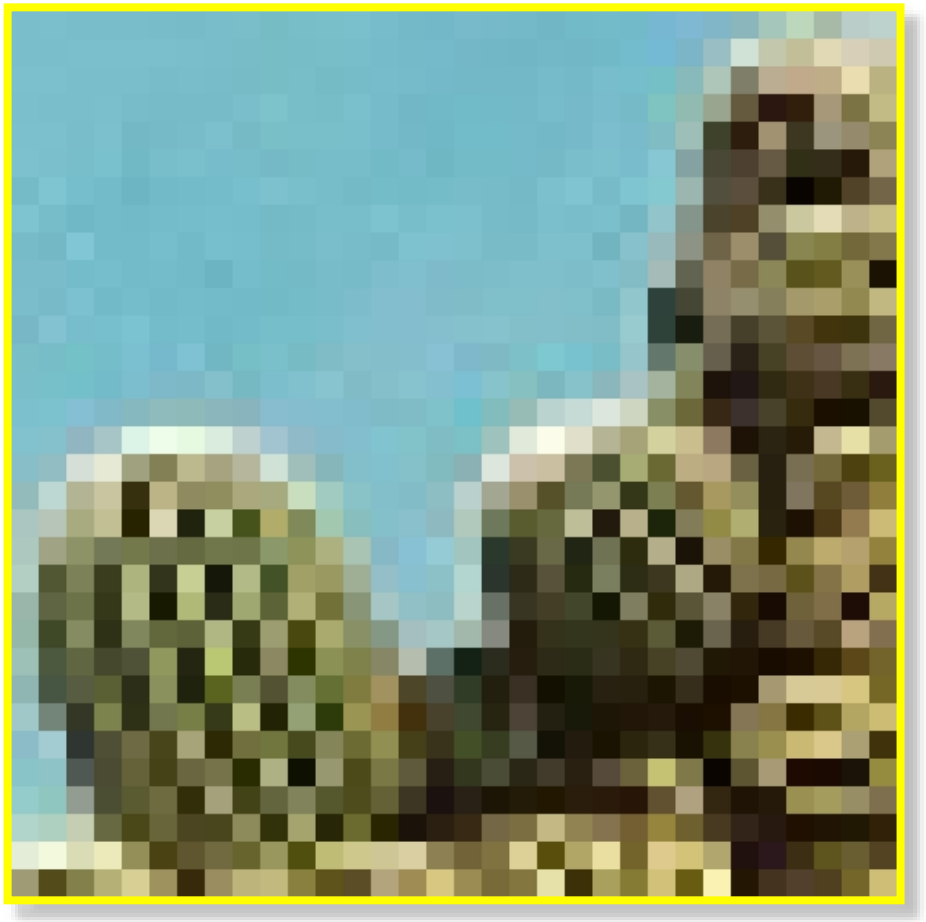


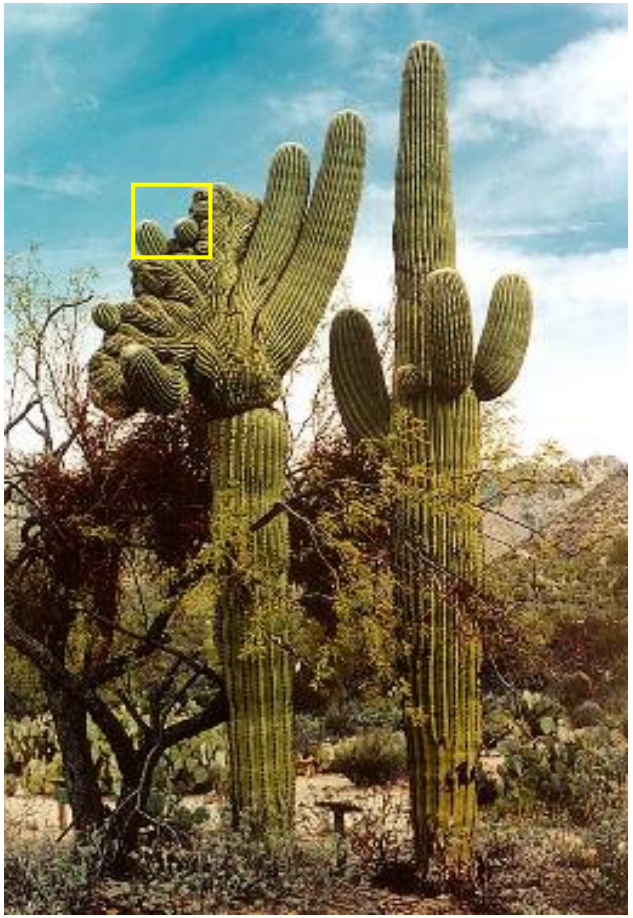
Resampling Through Bilinear Interpolation





Resampling Through Bilinear Interpolation





Resampling Through Bicubic Interpolation

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

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

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

$(r\text{-}1,c\text{-}1)$	$(r\text{-}1,c)$	$(r\text{-}1,c+1)$	$(r\text{-}1,c+2)$
$(r,c\text{-}1)$	(r,c)	$(r,c+1)$	$(r,c+2)$
$(r+1,c\text{-}1)$	$(r+1,c)$	$(r+1,c+1)$	$(r+1,c+2)$
$(r+2,c\text{-}1)$	$(r+2,c)$	$(r+2,c+1)$	$(r+2,c+2)$

The derivatives are are computed digitally on the 8-neighborhoods of the 4 pixel locs. Therefore a 4×4 neighborhood of the input image is needed for each output value.

Bicubic Interpolation

Bicubic interpolation uses a third-order polynomial to estimate the value of image **I** on subpixel (r_f, c_f). A bicubic polynomial on the square [0,1]×[0,1] is defined by

$$p_{\text{bicub}}(y,x) = \sum_{m=0}^{3}\sum_{n=0}^{3} \alpha_{m,n} y^m x^n.$$

To apply this to an image at pixel location (r_f, c_f), we substitute $y = r_f - r_0$ and $x = c_f - c_0$ where $r_0 = \lfloor r_f \rfloor$ and $c_0 = \lfloor c_f \rfloor$ so that p_{bicub} is a function on [0,1]×[0,1]. The 16 coefficients $\alpha_{i,j}$ are derived from the values of the image and its derivatives at the 4 pixel locations surrounding (r_f, c_f). In particular,

$$p_{\text{bicub}}(r,c) = \mathbf{I}(r,c), \quad \frac{\partial}{\partial r} p_{\text{bicub}}(r,c) = \frac{\partial}{\partial r}\mathbf{I}(r,c),$$
$$\frac{\partial}{\partial c} p_{\text{bicub}}(r,c) = \frac{\partial}{\partial c}\mathbf{I}(r,c), \quad \frac{\partial}{\partial cr} p_{\text{bicub}}(r,c) = \frac{\partial}{\partial cr}\mathbf{I}(r,c),$$

at pixel locations (r_0,c_0), (r_0+1,c_0), (r_0,c_0+1), and (r_0+1,c_0+1).

Bicubic Interpolation

The derivatives of the polynomial are

$$\frac{\partial}{\partial r} p_{\text{bicub}}(r,c) = \sum_{m=1}^{3}\sum_{n=0}^{3} \alpha_{m,n} m r^{m-1} c^{n},$$

$$\frac{\partial}{\partial c} p_{\text{bicub}}(r,c) = \sum_{m=0}^{3}\sum_{n=1}^{3} \alpha_{m,n} n r^{m} c^{n-1},$$

$$\frac{\partial}{\partial cr} p_{\text{bicub}}(r,c) = \sum_{m=1}^{3}\sum_{n=1}^{3} \alpha_{m,n} m n r^{m-1} c^{n-1}.$$

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

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

$$\frac{\partial}{\partial cr}\mathbf{I}(r,c) = \tfrac{1}{4}\left[\mathbf{I}(r+1,c+1) - \mathbf{I}(r+1,c-1) + \mathbf{I}(r-1,c-1) - \mathbf{I}(r-1,c+1)\right].$$

are the derivatives of the image.

Bicubic Interpolation

Bicubic interpolation uses a third-order polynomial to estimate the unknown value. A bicubic polynomial is defined by

$$p_{\text{bicub}}(r,c) = \sum_{m=0}^{3}\sum_{n=0}^{3} \alpha_{m,n} r^m c^n.$$

This has 16 coefficients $\alpha_{i,j}$ that are derived from the values of the image and its derivatives at the 4 pixel locations surrounding the desired point. Let A be the matrix of coefficients, and C a matrix of constants defined as ,

$$A = \begin{bmatrix} \alpha_{00} & \alpha_{01} & \alpha_{02} & \alpha_{03} \\ \alpha_{10} & \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{20} & \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{30} & \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}.$$

Bicubic Interpolation

Let D be the matrix of image values and derivatives,

$$D = \begin{bmatrix} \mathbf{I}(r,c) & \mathbf{I}(r,c+1) & \frac{\partial}{\partial c}\mathbf{I}(r,c) & \frac{\partial}{\partial c}\mathbf{I}(r,c+1) \\ \mathbf{I}(r+1,c) & \mathbf{I}(r+1,c+1) & \frac{\partial}{\partial c}\mathbf{I}(r+1,c) & \frac{\partial}{\partial c}\mathbf{I}(r+1,c+1) \\ \frac{\partial}{\partial r}\mathbf{I}(r,c) & \frac{\partial}{\partial r}\mathbf{I}(r,c+1) & \frac{\partial}{\partial c\partial r}\mathbf{I}(r,c) & \frac{\partial}{\partial c\partial r}\mathbf{I}(r,c+1) \\ \frac{\partial}{\partial r}\mathbf{I}(r+1,c) & \frac{\partial}{\partial r}\mathbf{I}(r+1,c+1) & \frac{\partial}{\partial c\partial r}\mathbf{I}(r+1,c) & \frac{\partial}{\partial c\partial r}\mathbf{I}(r+1,c+1) \end{bmatrix}.$$

Then the 16 coefficients, $\alpha_{i,j}$, for $i,j \in \{0,1,2,3\}$ are given by

$$A = CDC^{\mathrm{T}}$$

The derivatives in D are computed by central differences as shown in the next slides.

Bicubic Interpolation

Let D be the matrix of image values and derivatives,

pixel values

column derivatives

$$D = \begin{bmatrix} \mathbf{I}(r,c) & \mathbf{I}(r,c+1) & \frac{\partial}{\partial c}\mathbf{I}(r,c) & \frac{\partial}{\partial c}\mathbf{I}(r,c+1) \\ \mathbf{I}(r+1,c) & \mathbf{I}(r+1,c+1) & \frac{\partial}{\partial c}\mathbf{I}(r+1,c) & \frac{\partial}{\partial c}\mathbf{I}(r+1,c+1) \\ \frac{\partial}{\partial r}\mathbf{I}(r,c) & \frac{\partial}{\partial r}\mathbf{I}(r,c+1) & \frac{\partial}{\partial c\partial r}\mathbf{I}(r,c) & \frac{\partial}{\partial c\partial r}\mathbf{I}(r,c+1) \\ \frac{\partial}{\partial r}\mathbf{I}(r+1,c) & \frac{\partial}{\partial r}\mathbf{I}(r+1,c+1) & \frac{\partial}{\partial c\partial r}\mathbf{I}(r+1,c) & \frac{\partial}{\partial c\partial r}\mathbf{I}(r+1,c+1) \end{bmatrix}.$$

row derivatives

cross derivatives

Then the 16 coefficients, $\alpha_{i,j}$, for $i,j \in \{0,1,2,3\}$ are given by

$$A = CDC^{\mathrm{T}}$$

The derivatives in D are computed by central differences as shown in the next slides.

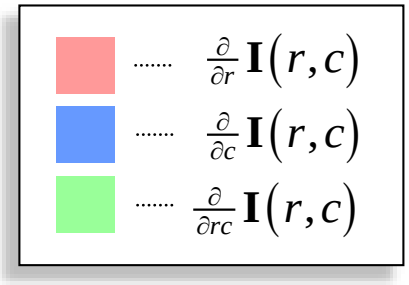
Resampling Through Bicubic Interpolation

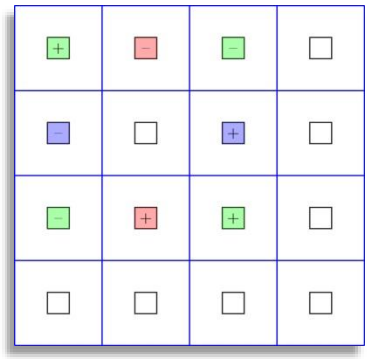
$\mathcal{N}(r,c)$ is the 8-pixel neighborhood of (r,c).

$$\frac{\partial}{\partial r}\mathbf{I}(r,c) = \text{ avg. of forward diff and backward diff @ } (r,c)$$

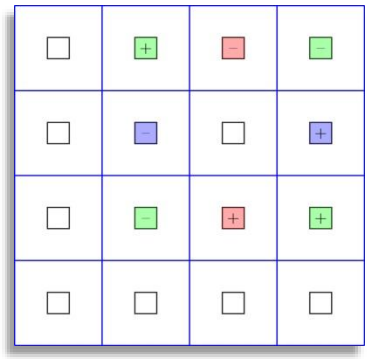
$$= \tfrac{1}{2}\Big[\big(\mathbf{I}(r+1,c)-\mathbf{I}(r,c)\big) + \big(\mathbf{I}(r,c)-\mathbf{I}(r-1,c)\big)\Big]$$

$$= \tfrac{1}{2}\Big[\mathbf{I}(r+1,c)-\mathbf{I}(r-1,c)\Big]$$

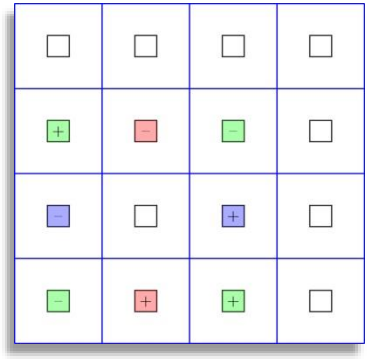




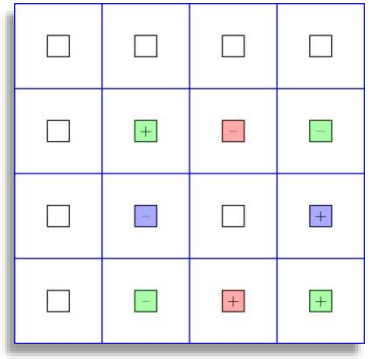
$\mathcal{N}(r,c)$



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



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



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

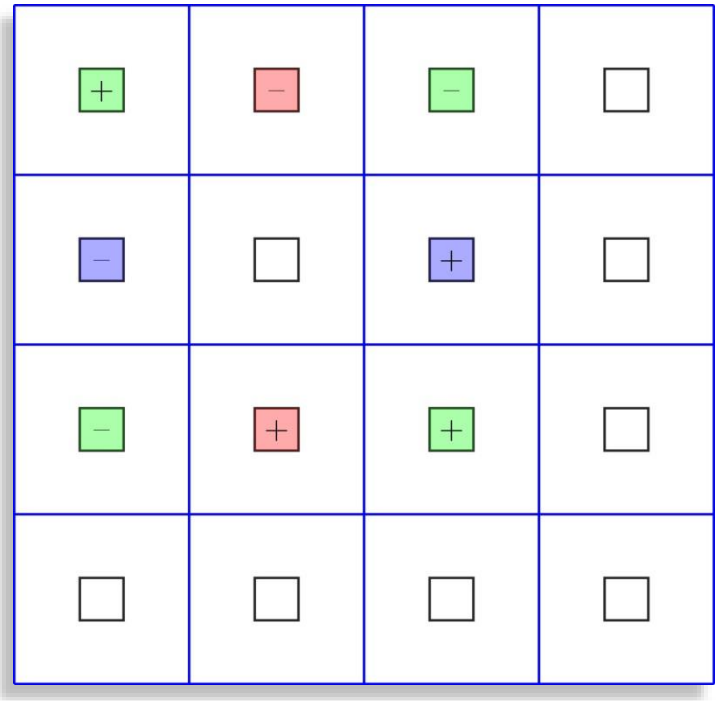
Resampling Through Bicubic Interpolation

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

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

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

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



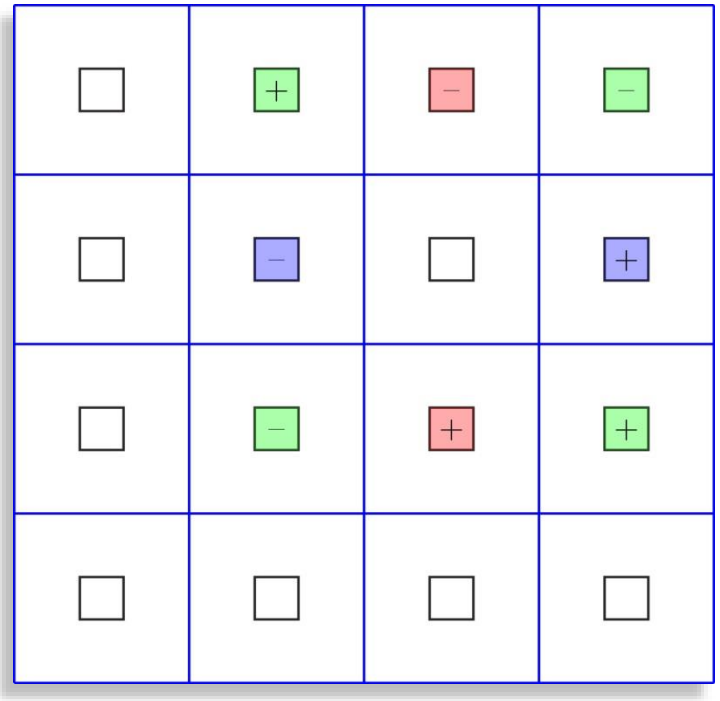
Resampling Through Bicubic Interpolation

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

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

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

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



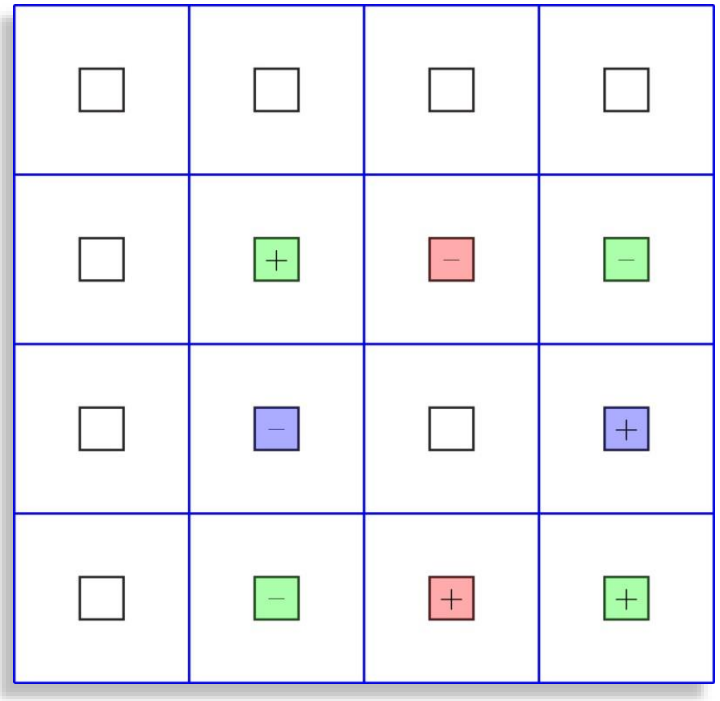
Resampling Through Bicubic Interpolation

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

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

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

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



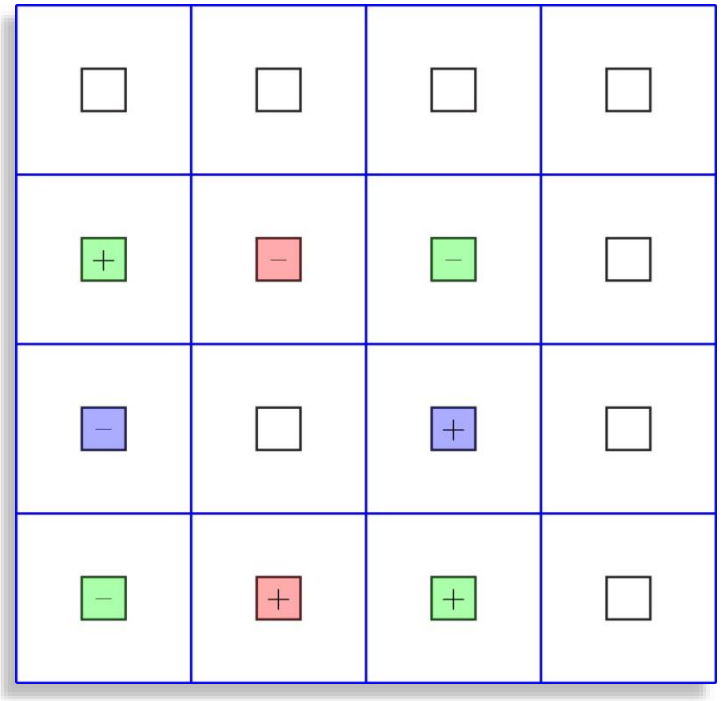
Resampling Through Bicubic Interpolation

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

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

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

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



Resampling Through Bicubic Interpolation

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

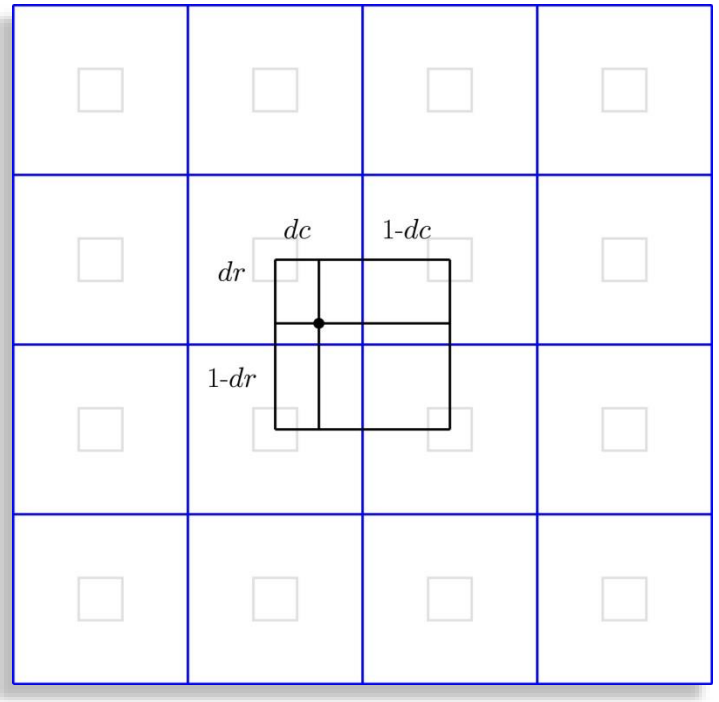
$$S_c = \begin{cases} C/C', & \text{if } C > C', \\ (C-1)/C', & \text{if } C < C'. \end{cases}$$

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

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

$$(dr,\, dc) = \left(r_f - r\ ,\ c_f - c\right)$$

This part is the same as NN interpolation.



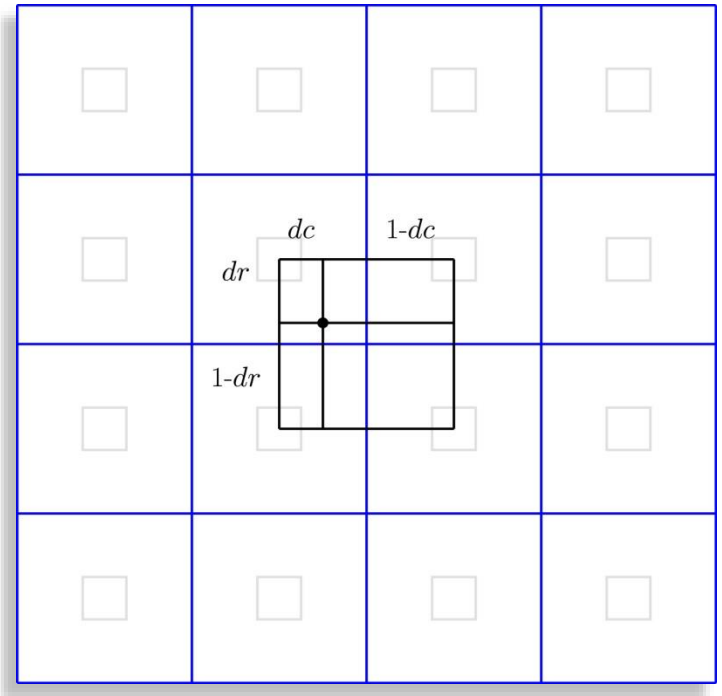
Resampling Through Bicubic Interpolation

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

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

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

The differentials on pp 112-115 are combined in this sum of products of polynomials.



Size Reduction 3/7

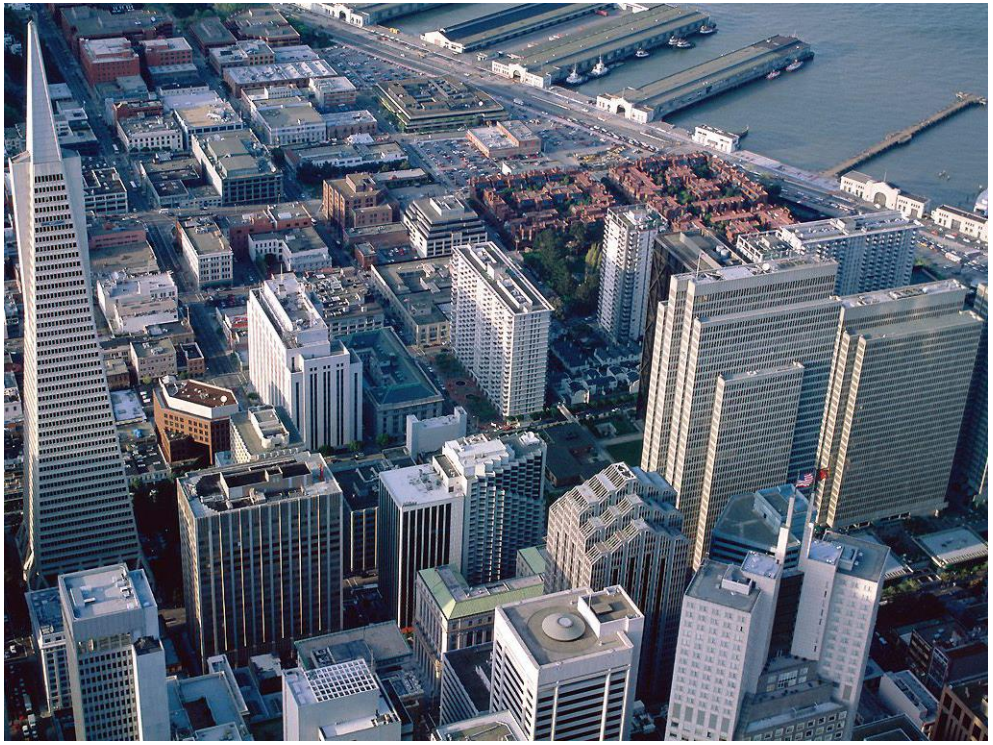
original

nearest neighbor

bilinear

bicubic

Size Reduction 3/7

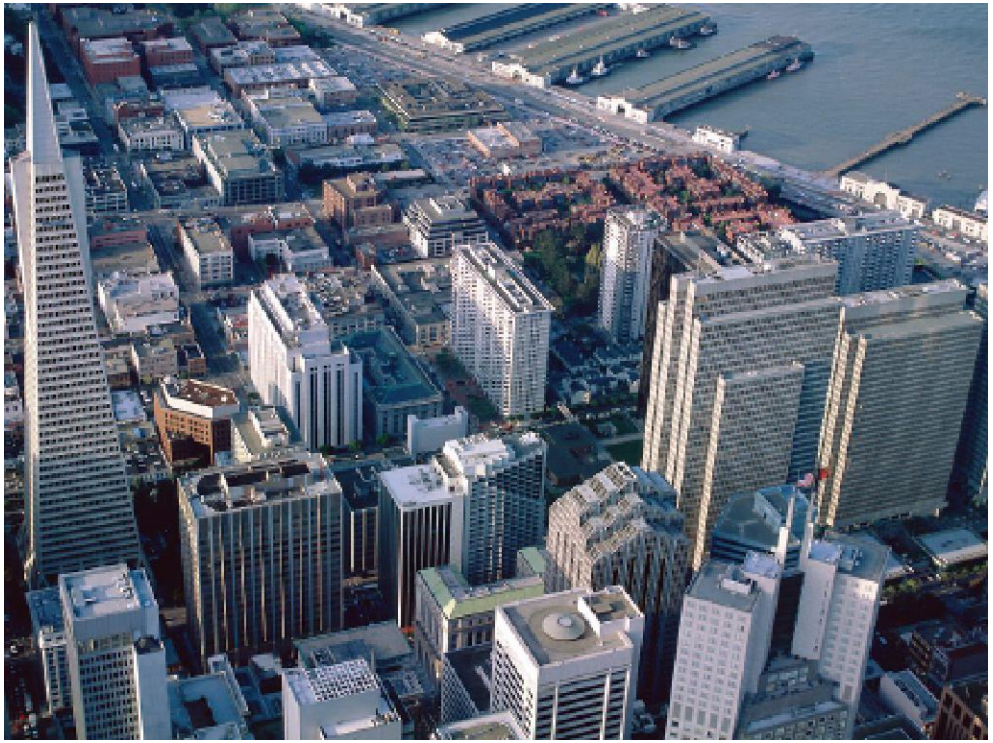


original

Size Reduction 3/7 (zoomed)

nearest neighbor

Size Reduction 3/7 (zoomed)

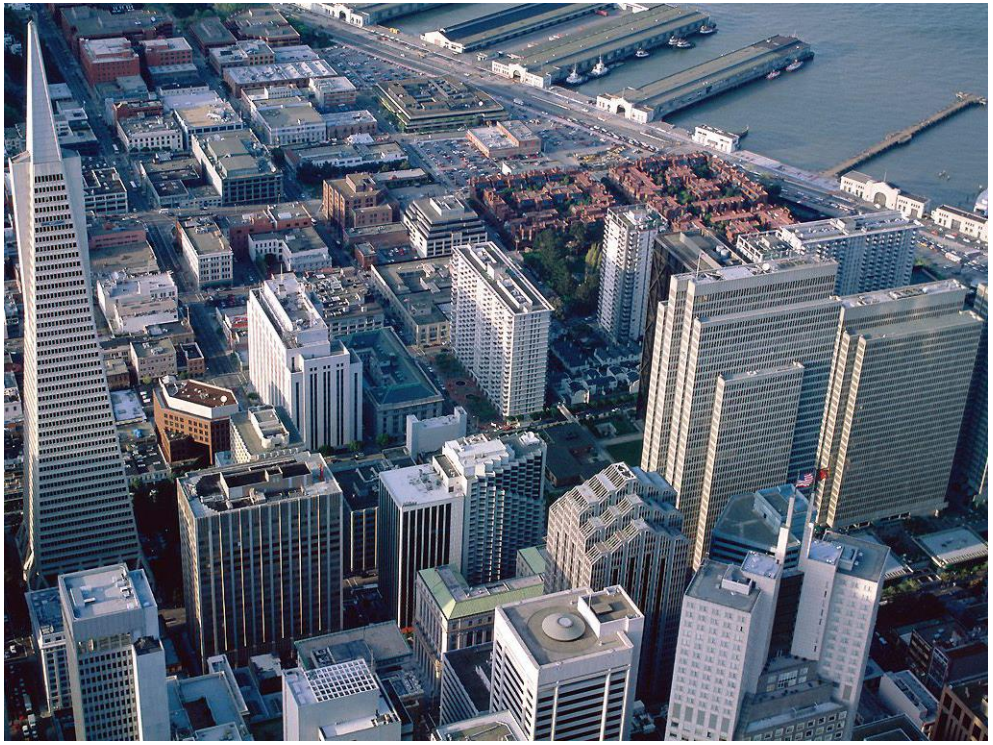


bilinear

Size Reduction 3/7 (zoomed)

bicubic

Size Reduction 3/7



original

Enlargement

nearest neighbor
7/3

Enlargement

bilinear 7/3

Enlargement

bicubic 7/3

General Case: Warping + Interpolation

1. Assume the input image, $\mathbf{I}$, has infinite spatial resolution.
2. Calculate the size, $R_{\text{out}} \times C_{\text{out}} \times B$, of the output image, $\mathbf{J}$, and allocate it.
3. Create an image map (a warping function, Φ) as follows:
   a) Allocate an $R_{\text{out}} \times C_{\text{out}} \times 2$ array, Φ.
   b) For every pixel location (r,c) in $\mathbf{J}$ find the corresponding real-valued pixel location (r_f, c_f) in $\mathbf{I}$.
   c) Set $\Phi(r,c,1) = r_f$ and set $\Phi(r,c,2) = c_f$.
4. Create an interpolation function, Θ, that generates a pixel value from the values of $\mathbf{I}$ on a neighborhood, $\mathcal{N}(r_f, c_f)$.
5. Then set $\mathbf{J}(r,c) = \Theta\{\mathbf{I}; \mathcal{N}(r_f, c_f)\}$.