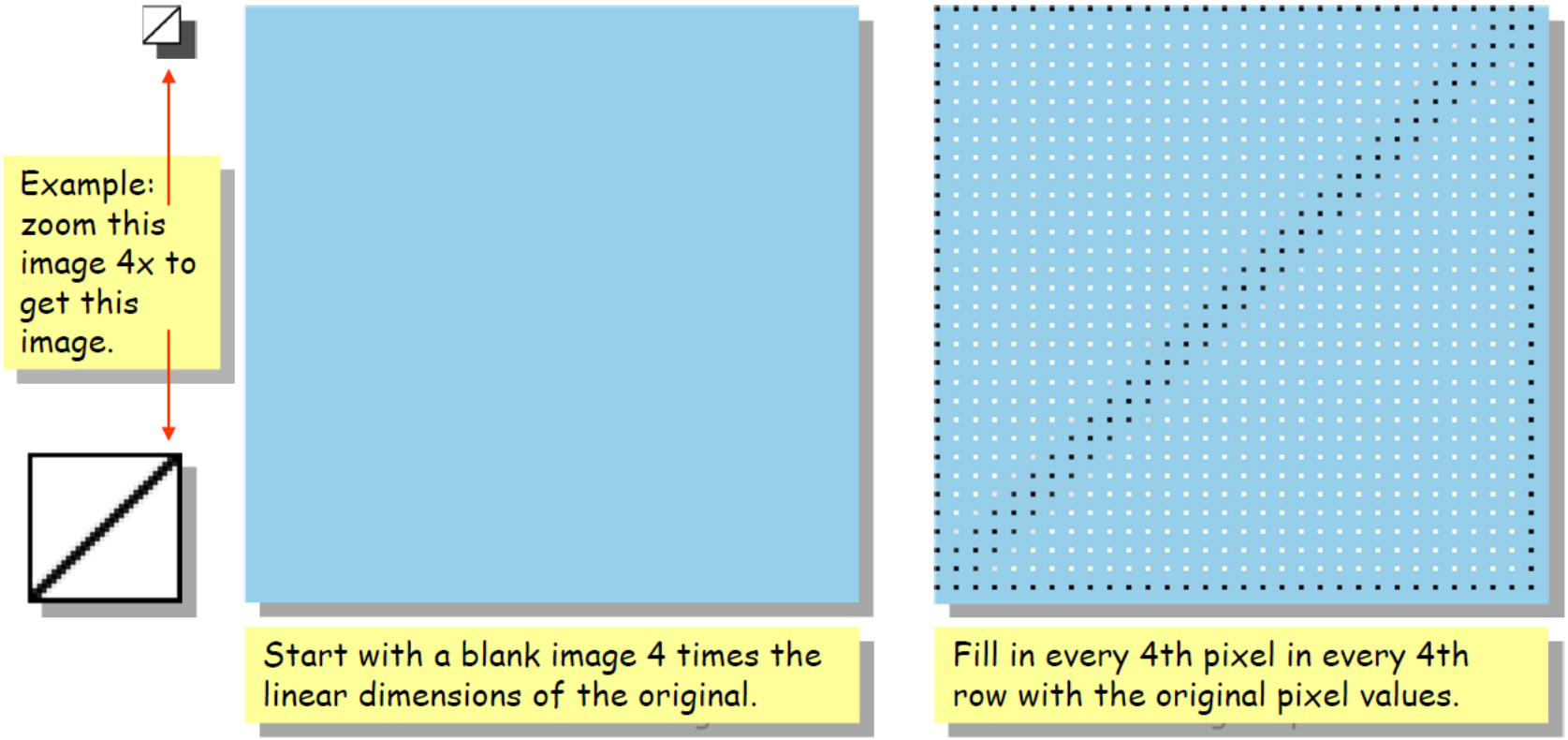
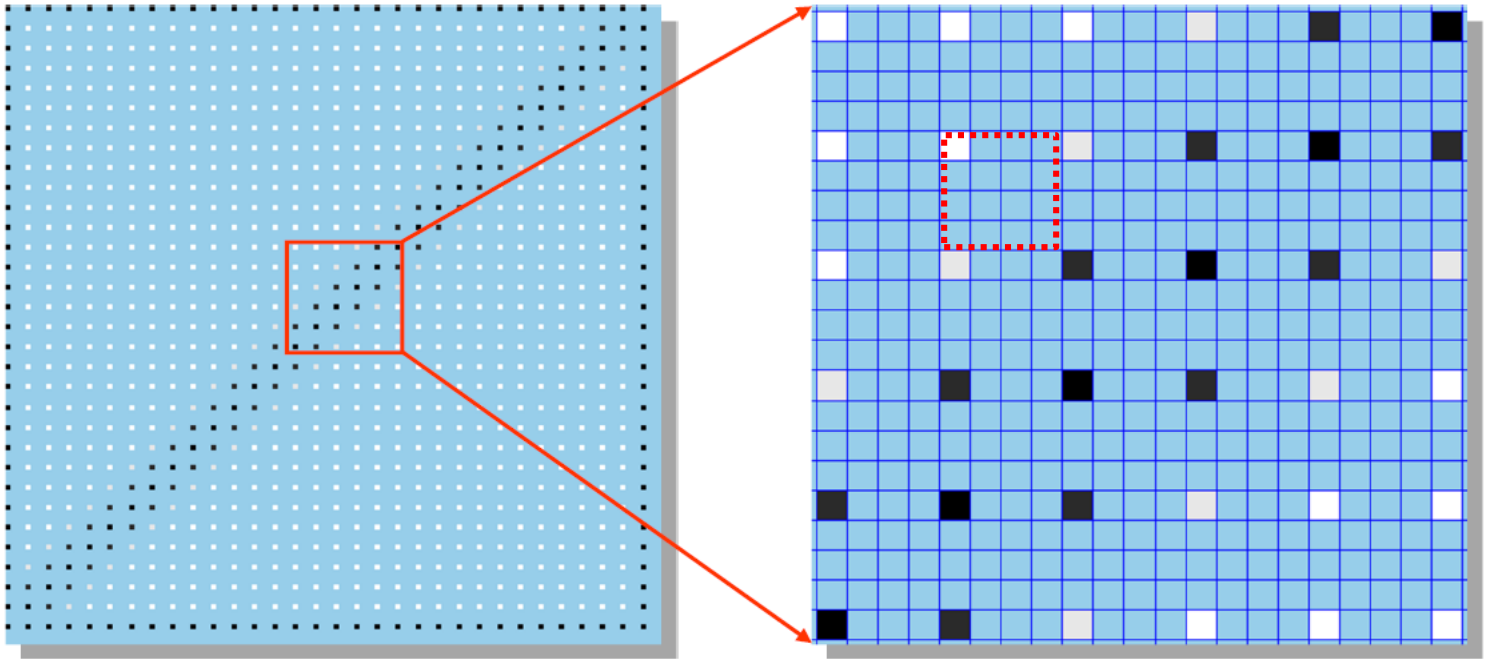
Resizing image

Enlarging Images Through Pixel Replication



Resizing image

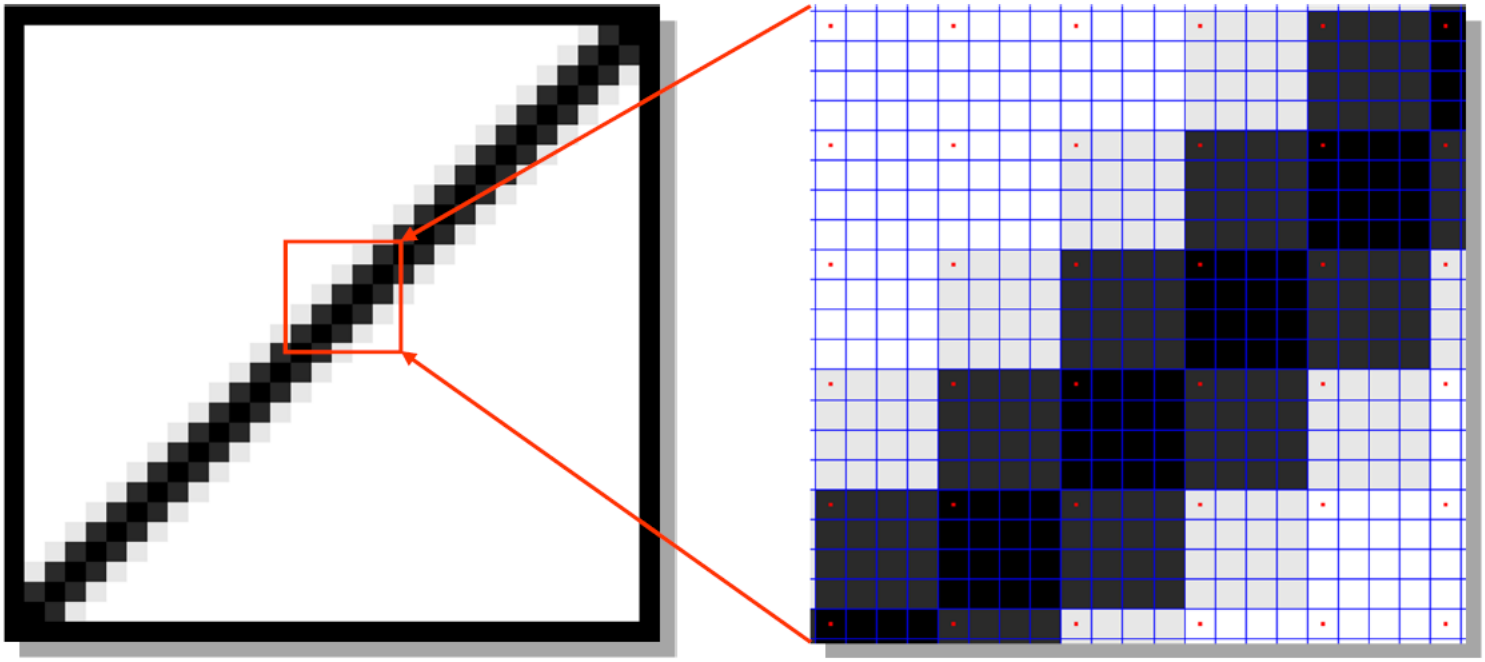
Enlarging Images Through Pixel Replication



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

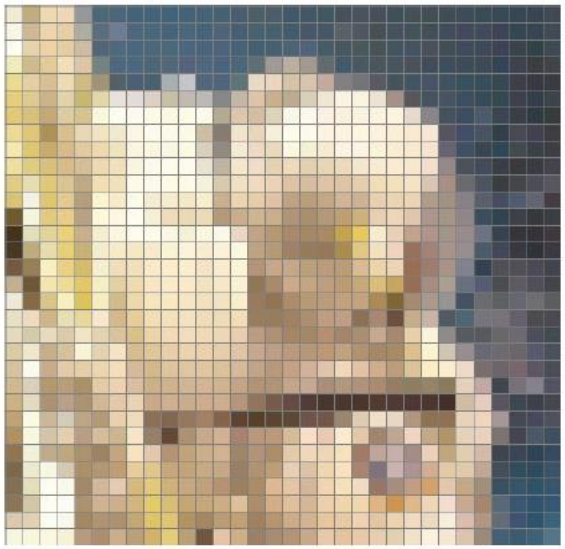
Resizing image

Enlarging Images Through Pixel Replication



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

Pixel Indexing in Matlab



```
r = 1:n:R;
I(r,:,:)
```

```
c = 1:n:C;
I(:,c,:)
```

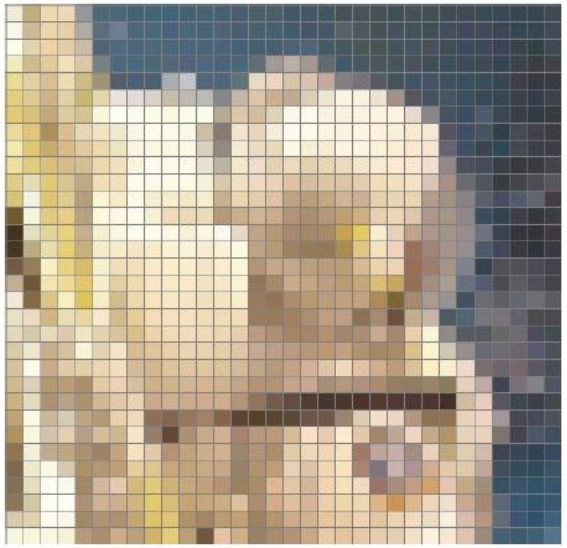
Here, n=3

r = [1 4 7 10 13 16 19 22 25 28 31]

c = [1 4 7 10 13 16 19 22 25 28 31]

To decimate the above image by a factor of *n*, create a vector, **r**, that contains the index of every *n*th row, and a similar vector, **c**.

Pixel Indexing in Matlab

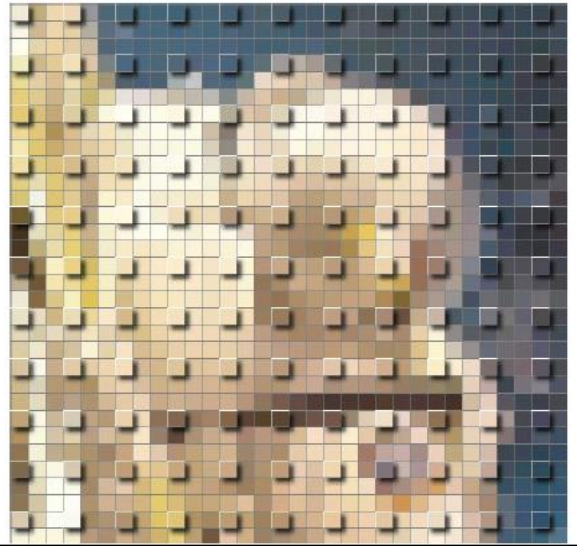


This is called, 'vectorizing'.

Take the pixels indexed by both r and c.

```
I(r,c)
```

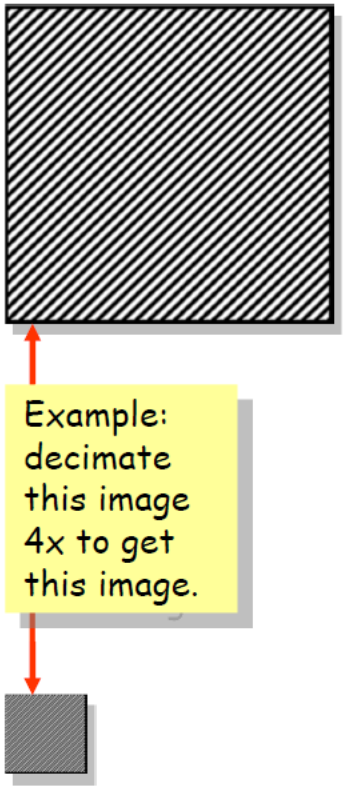
Here, n=3



Then, vectors **r** and **c** used as index arguments for image I select every nth column in every nth row.

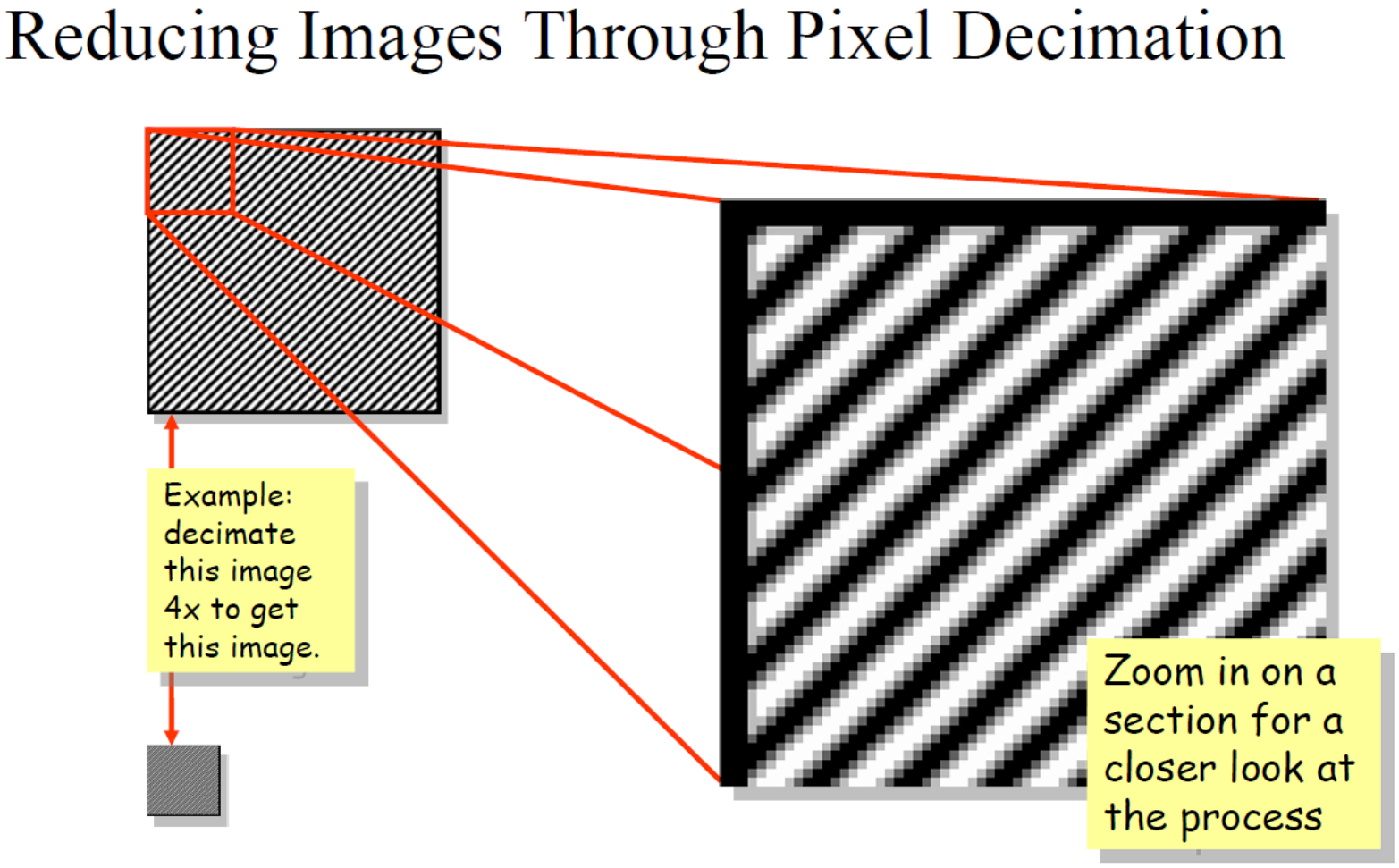
Resizing image

Reducing Images Through Pixel Decimation



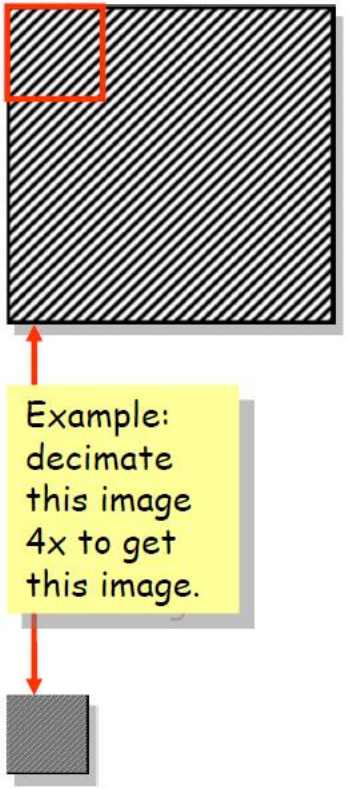
Decimation by a factor of n: take every nth pixel in every nth row

Resizing image



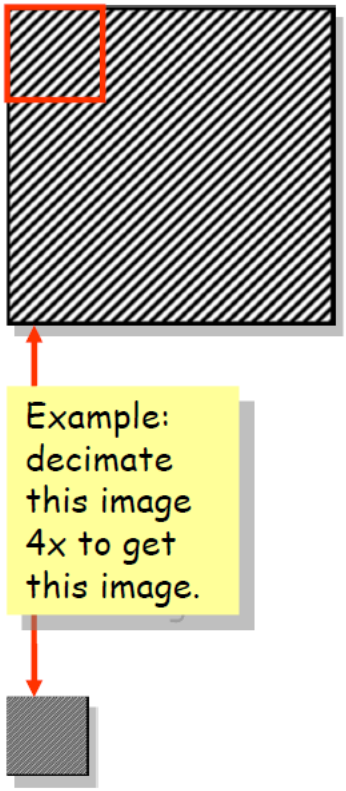
Resizing image

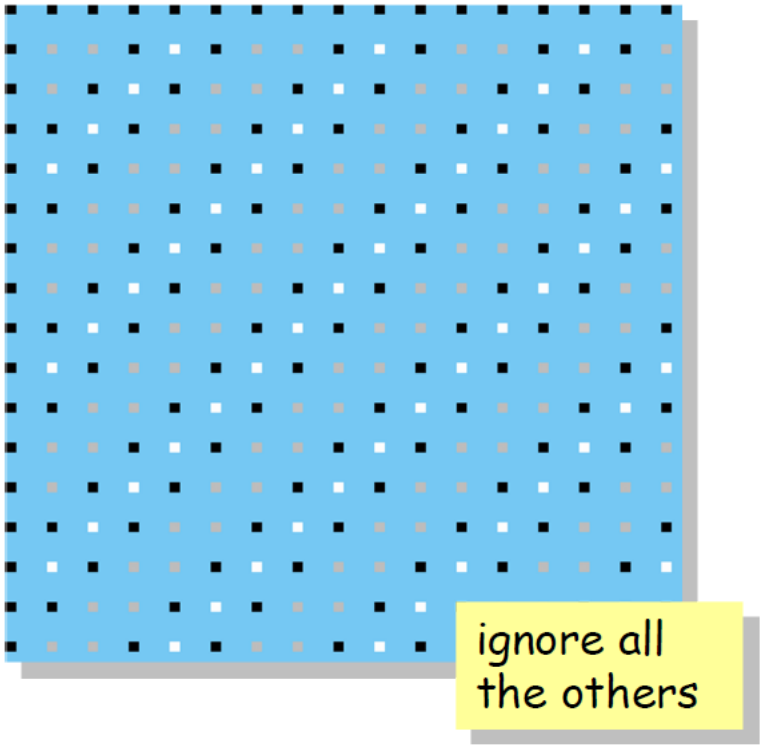
Reducing Images Through Pixel Decimation



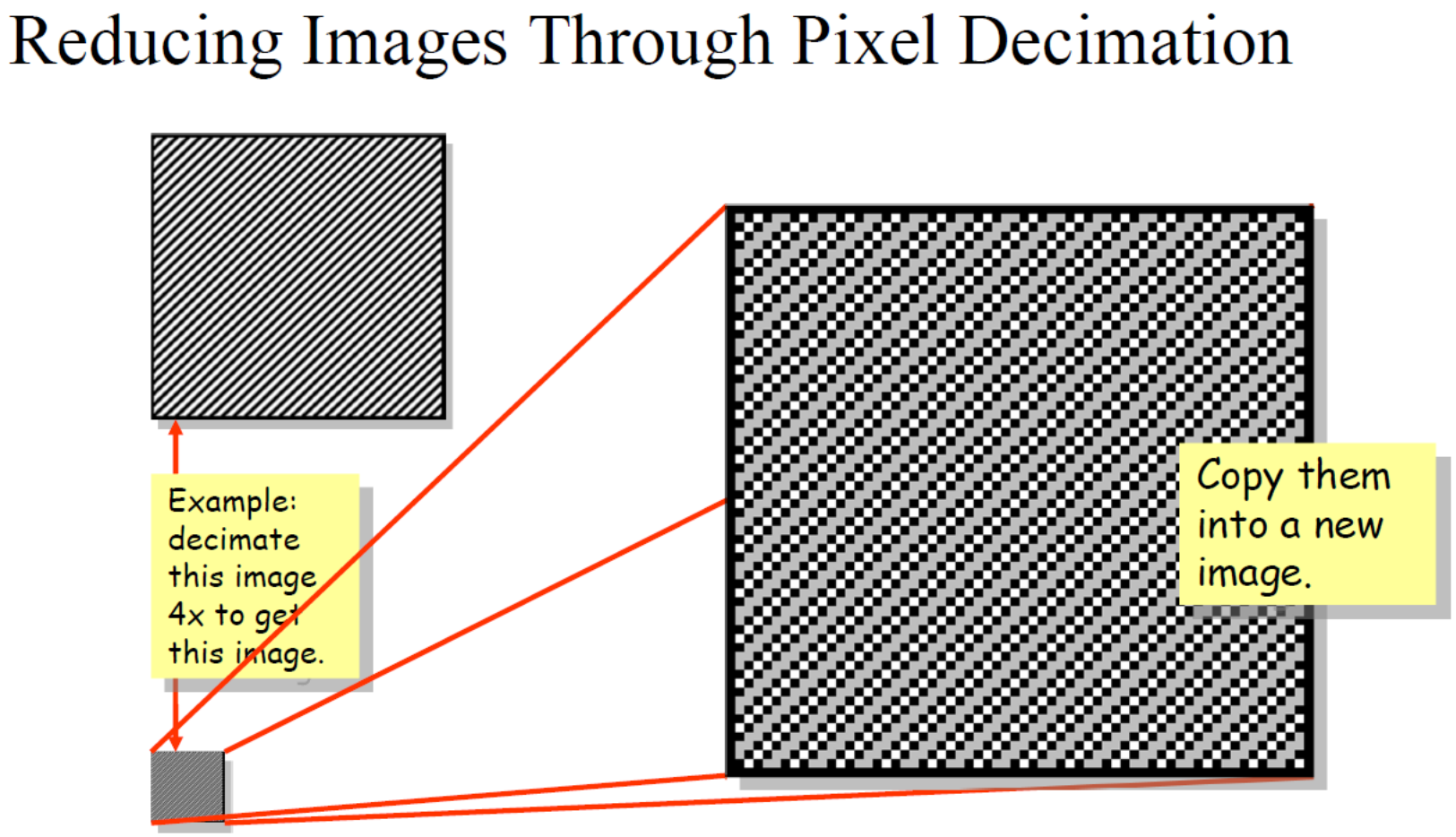
Resizing image

Reducing Images Through Pixel Decimation





Resizing image



Resizing image

Nearest
Neighbor
Resampling

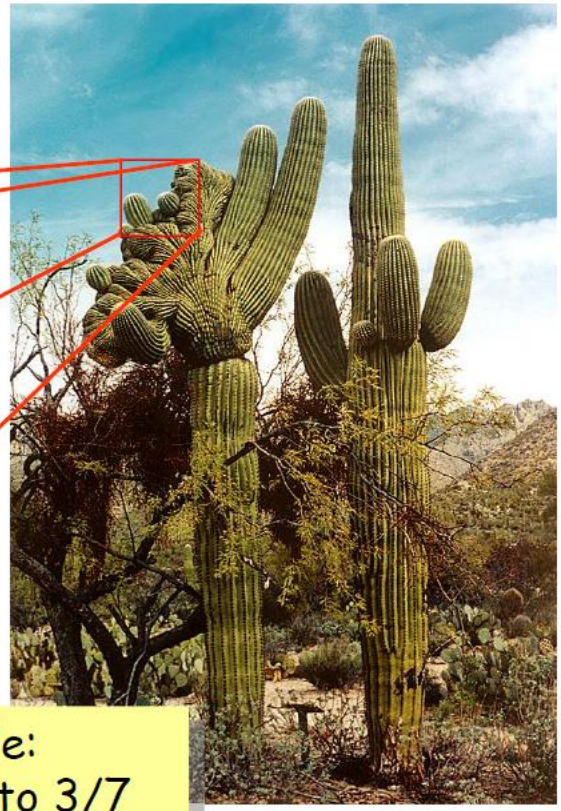
The "Nearest Neighbor" algorithm is a generalization of pixel replication and decimation.

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.)

Resizing image

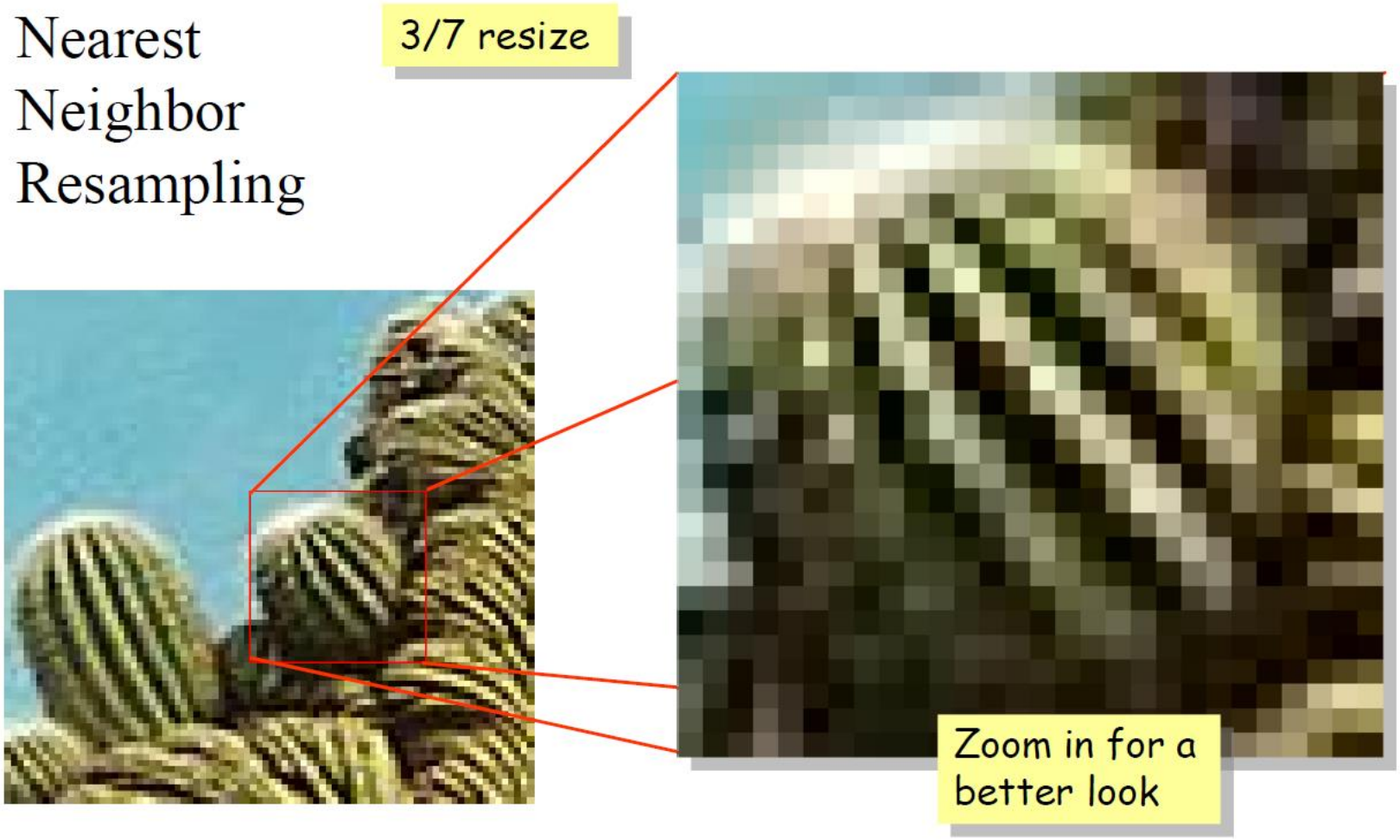
Nearest Neighbor Resampling

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



Example: resize to 3/7 of the original

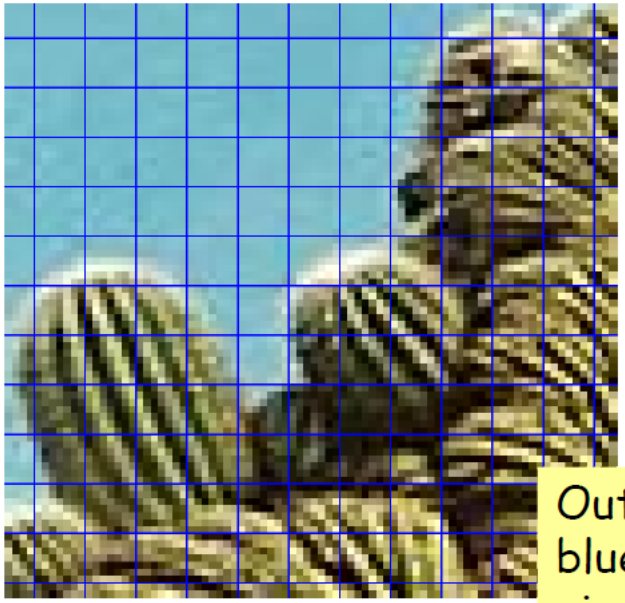
Resizing image

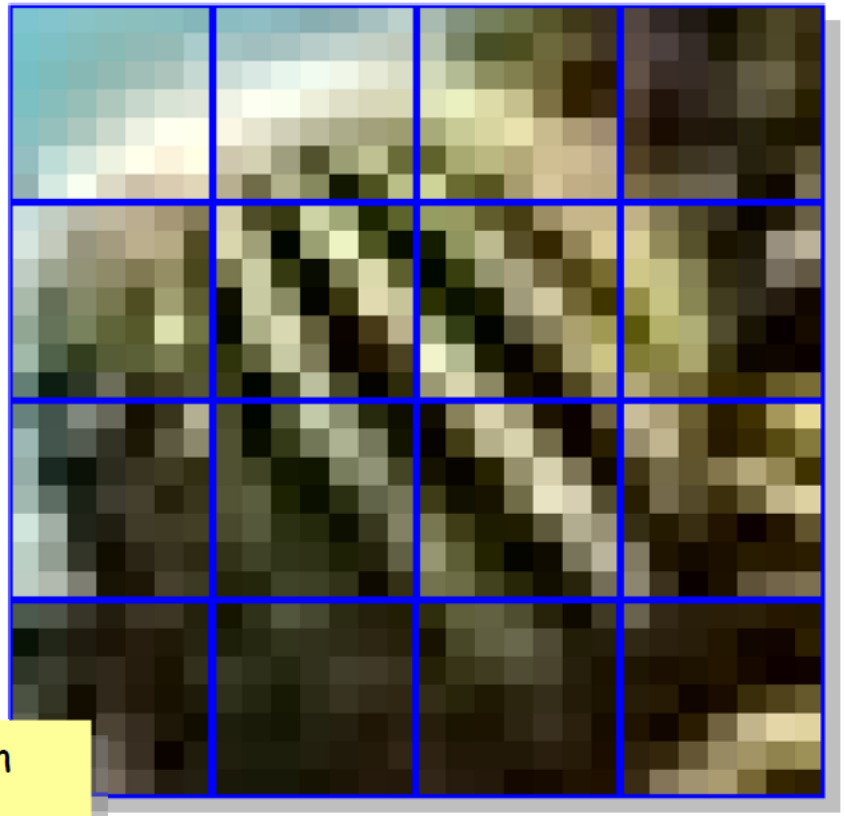


Resizing image

Nearest Neighbor Resampling

3/7 resize



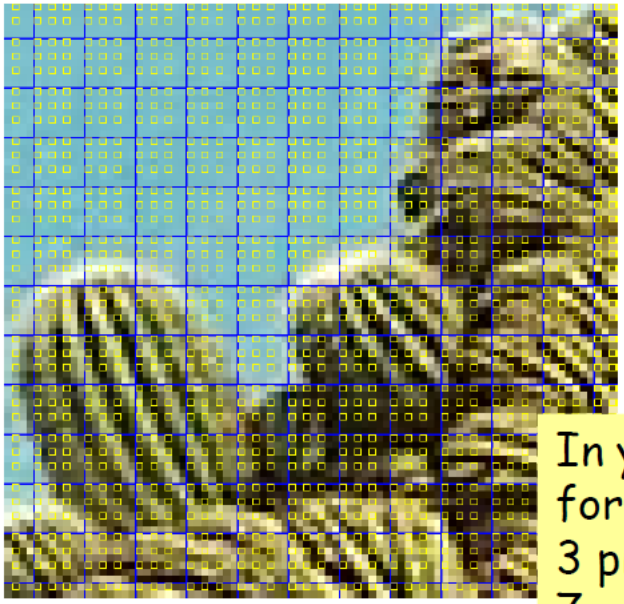


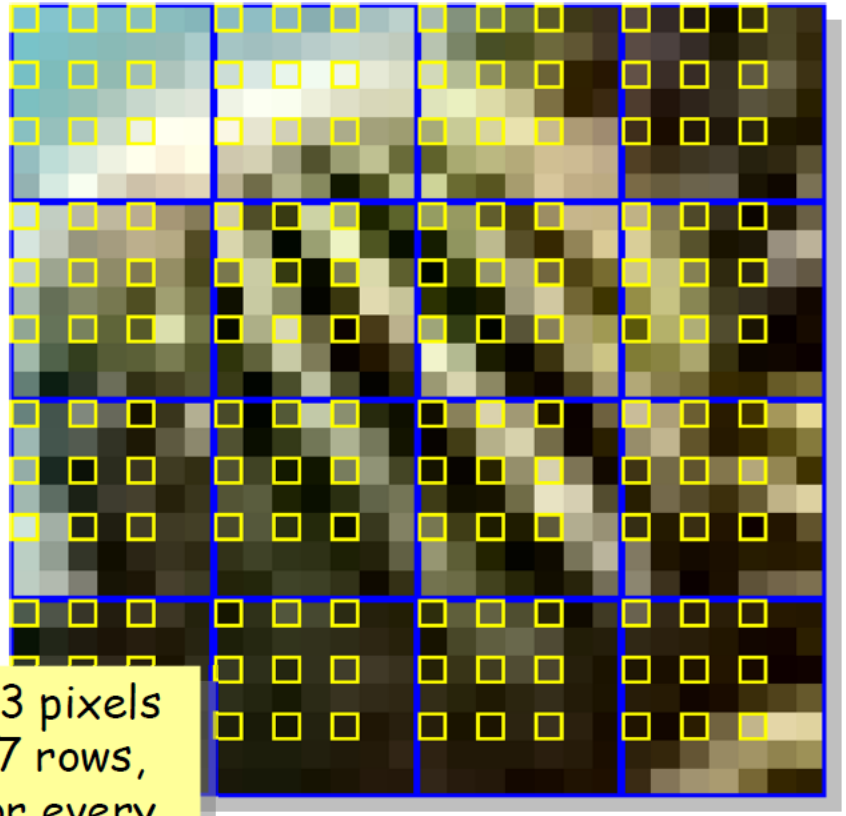
Outlined in blue: 7x7 pixel squares

Resizing image

Nearest Neighbor Resampling

3/7 resize



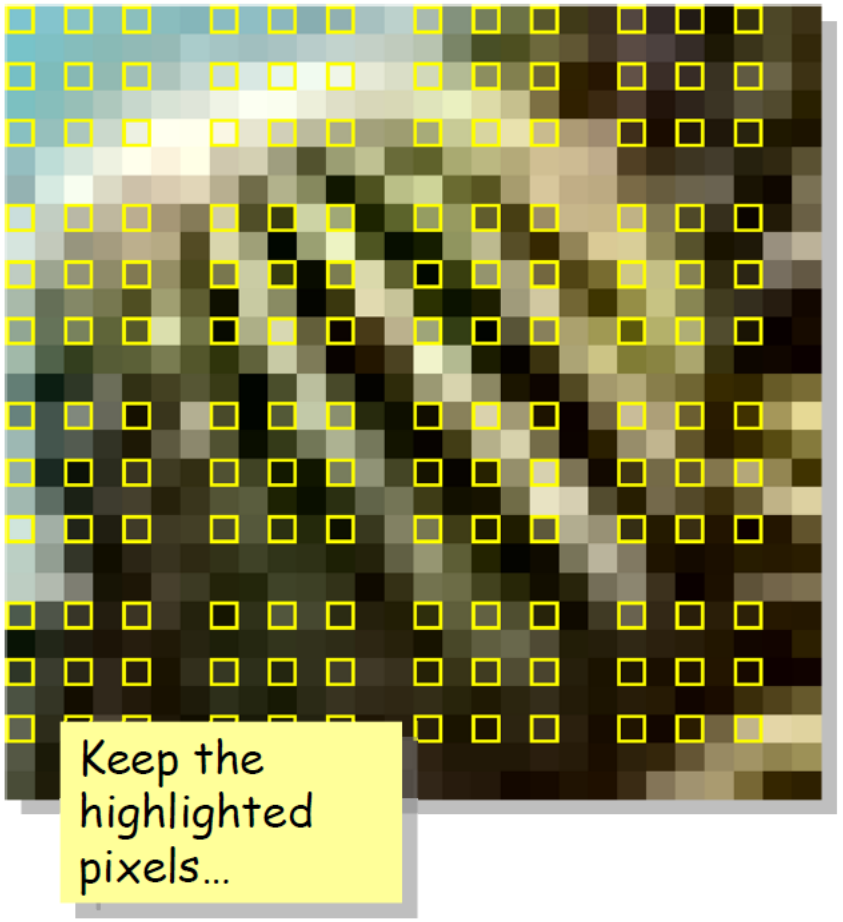


In yellow: 3 pixels for every 7 rows, 3 pixels for every 7 cols.

Resizing image

Nearest Neighbor Resampling

3/7 resize

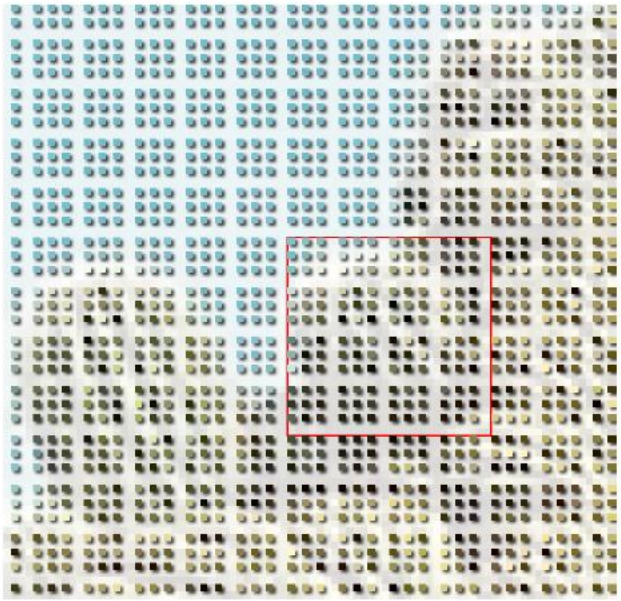


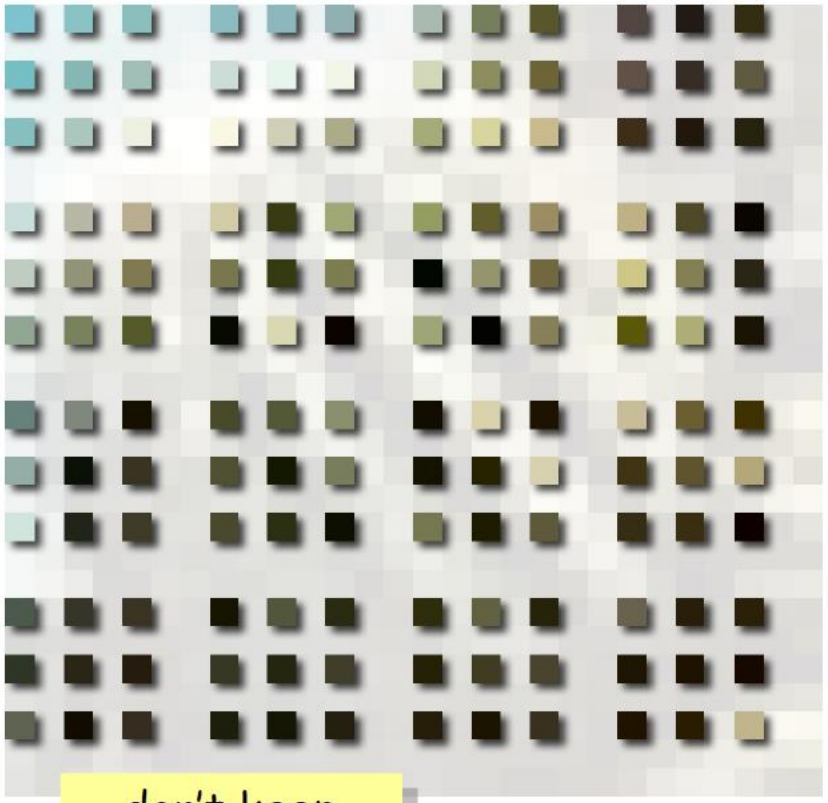
Keep the highlighted pixels...

Resizing image

Nearest Neighbor Resampling

3/7 resize



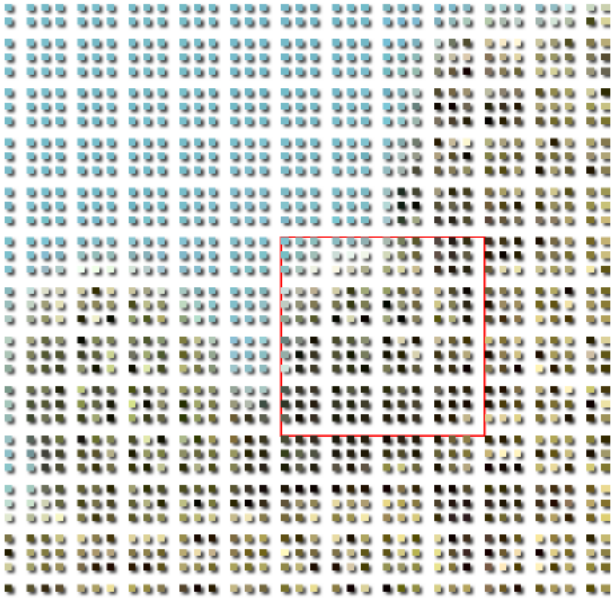


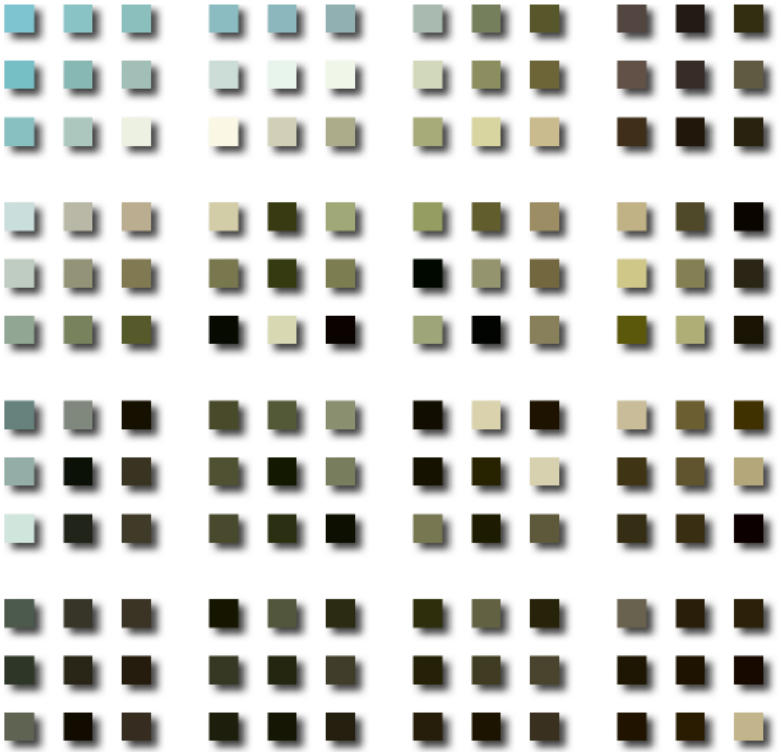
... don't keep the others.

Resizing image

Nearest
Neighbor
Resampling

3/7 resize



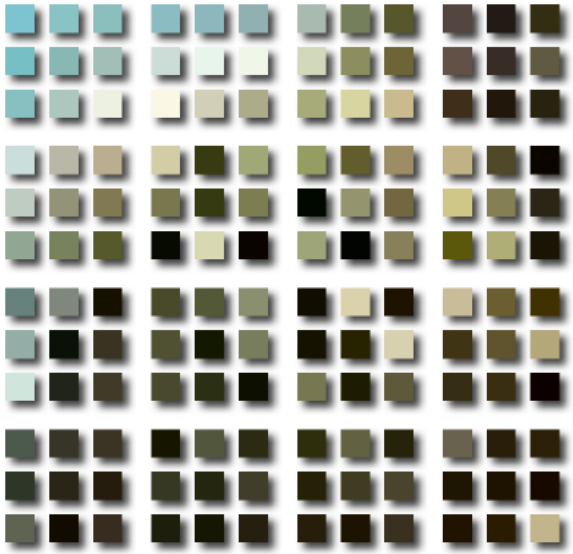


Copy them into
a new image.

Resizing image

Nearest Neighbor Resampling

3/7 resize

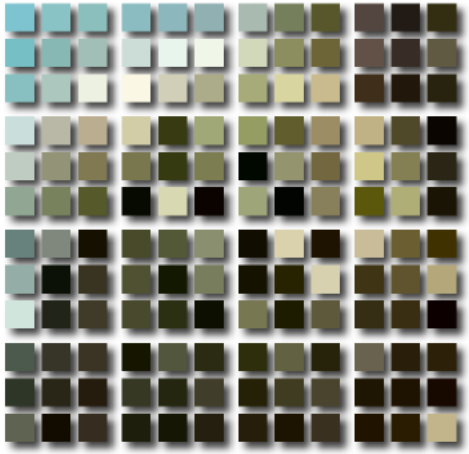


Copy them into a new image.

Resizing image

Nearest Neighbor Resampling

3/7 resize

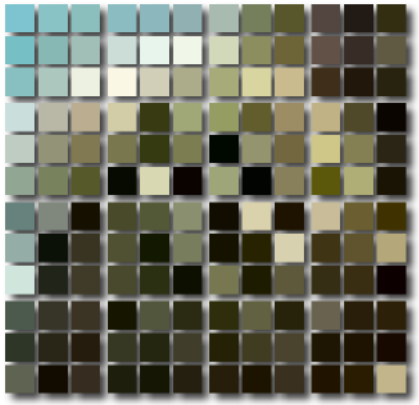


Copy them into a new image.

Resizing image

Nearest
Neighbor
Resampling

3/7 resize

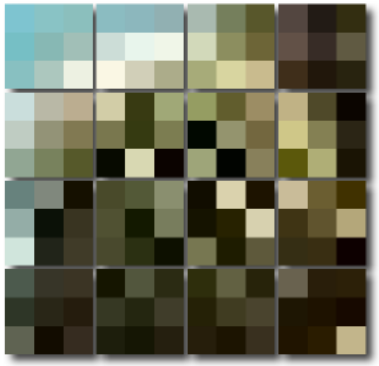


Copy them into
a new image.

Resizing image

Nearest
Neighbor
Resampling

3/7 resize

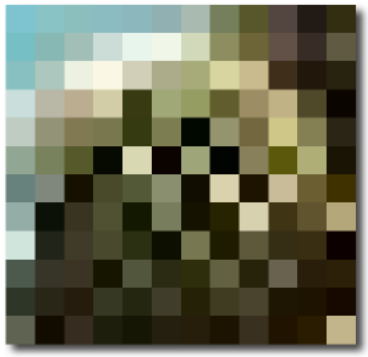


Copy them into
a new image.

Resizing image

Nearest Neighbor Resampling

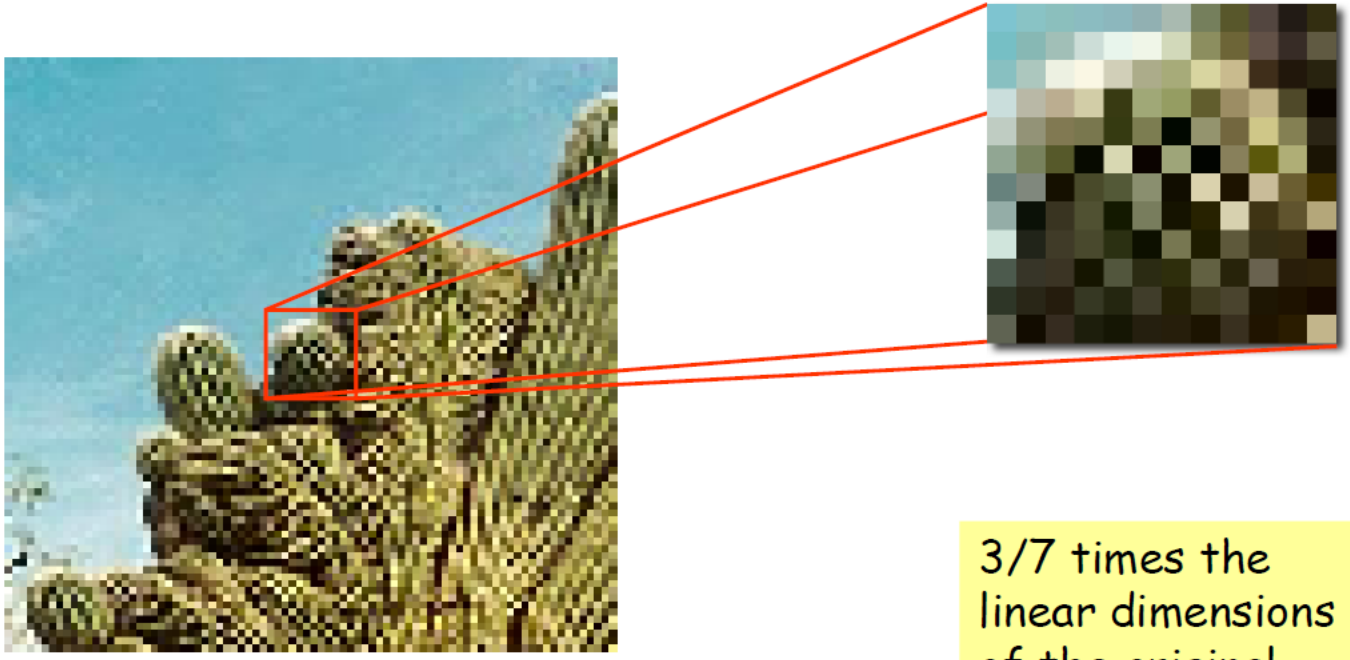
3/7 resize



3/7 times the linear dimensions of the original

Resizing image

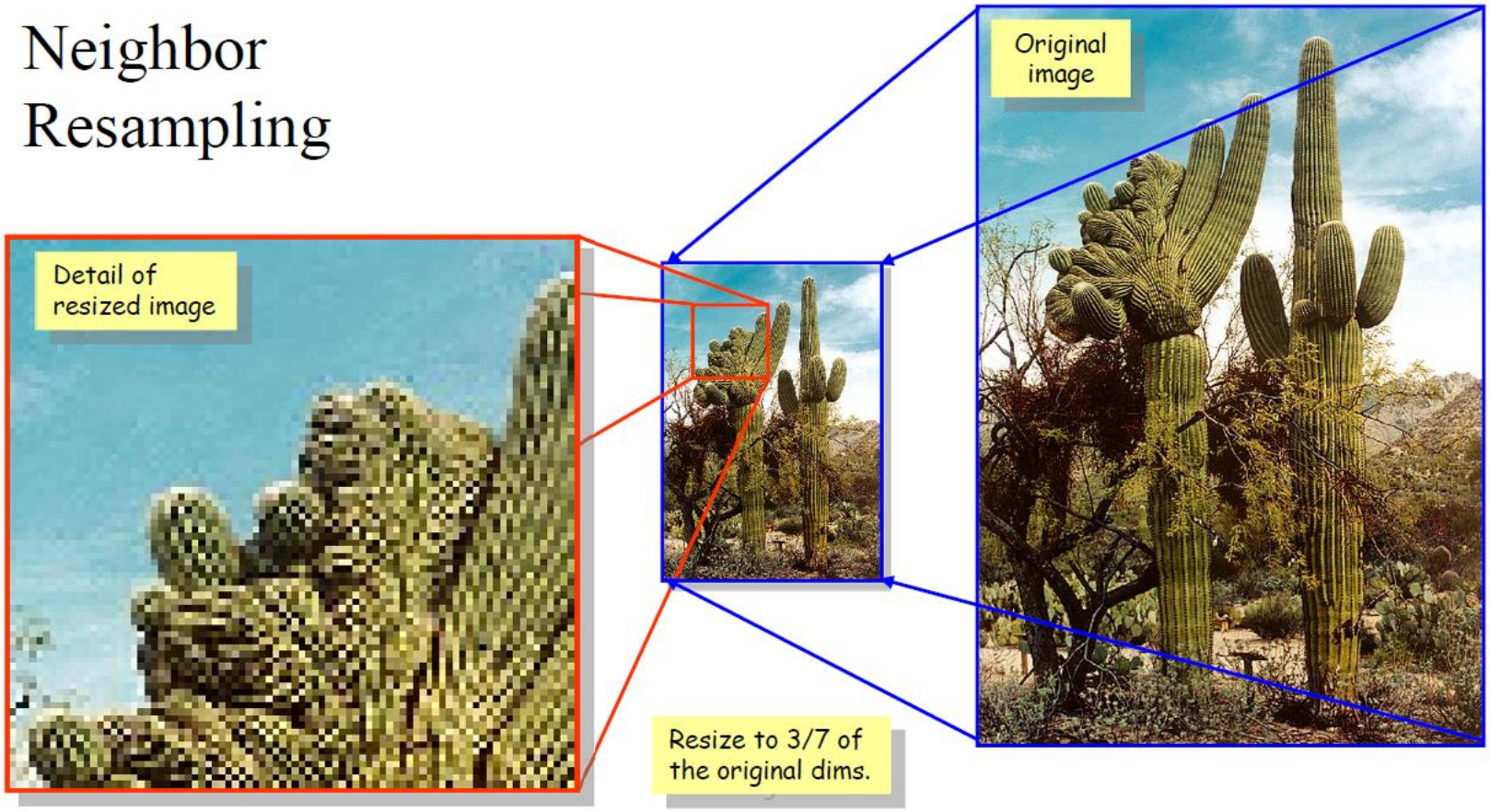
Nearest
Neighbor
Resampling



3/7 times the linear dimensions of the original

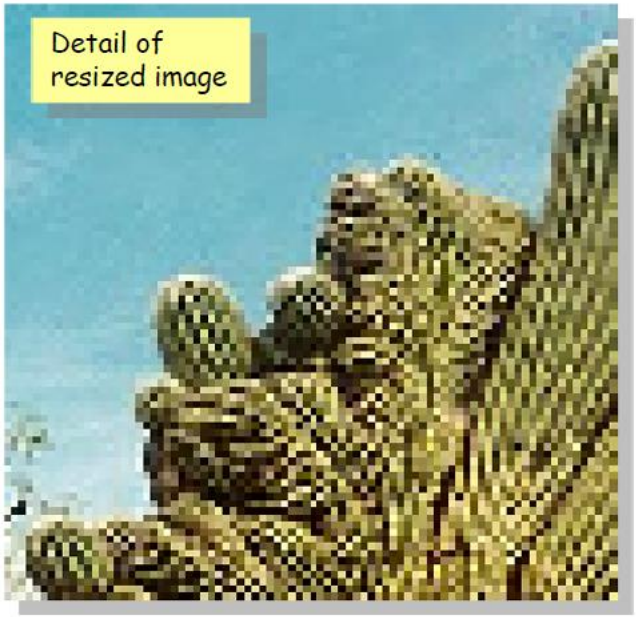
Resizing image

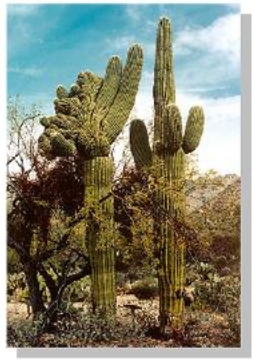
Nearest
Neighbor
Resampling



Resizing image

Nearest
Neighbor
Resampling

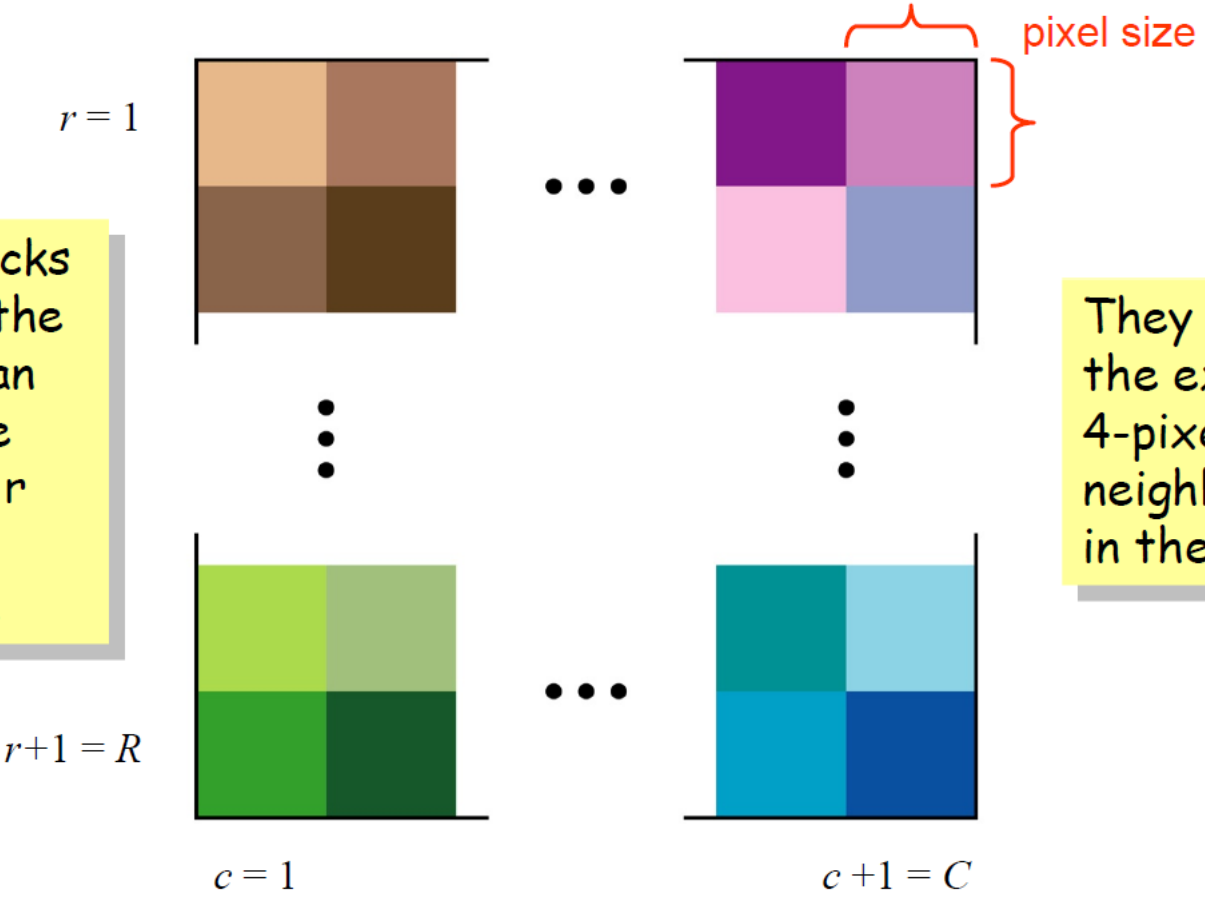




Resize to 3/7 of
the original dims.

Resizing image

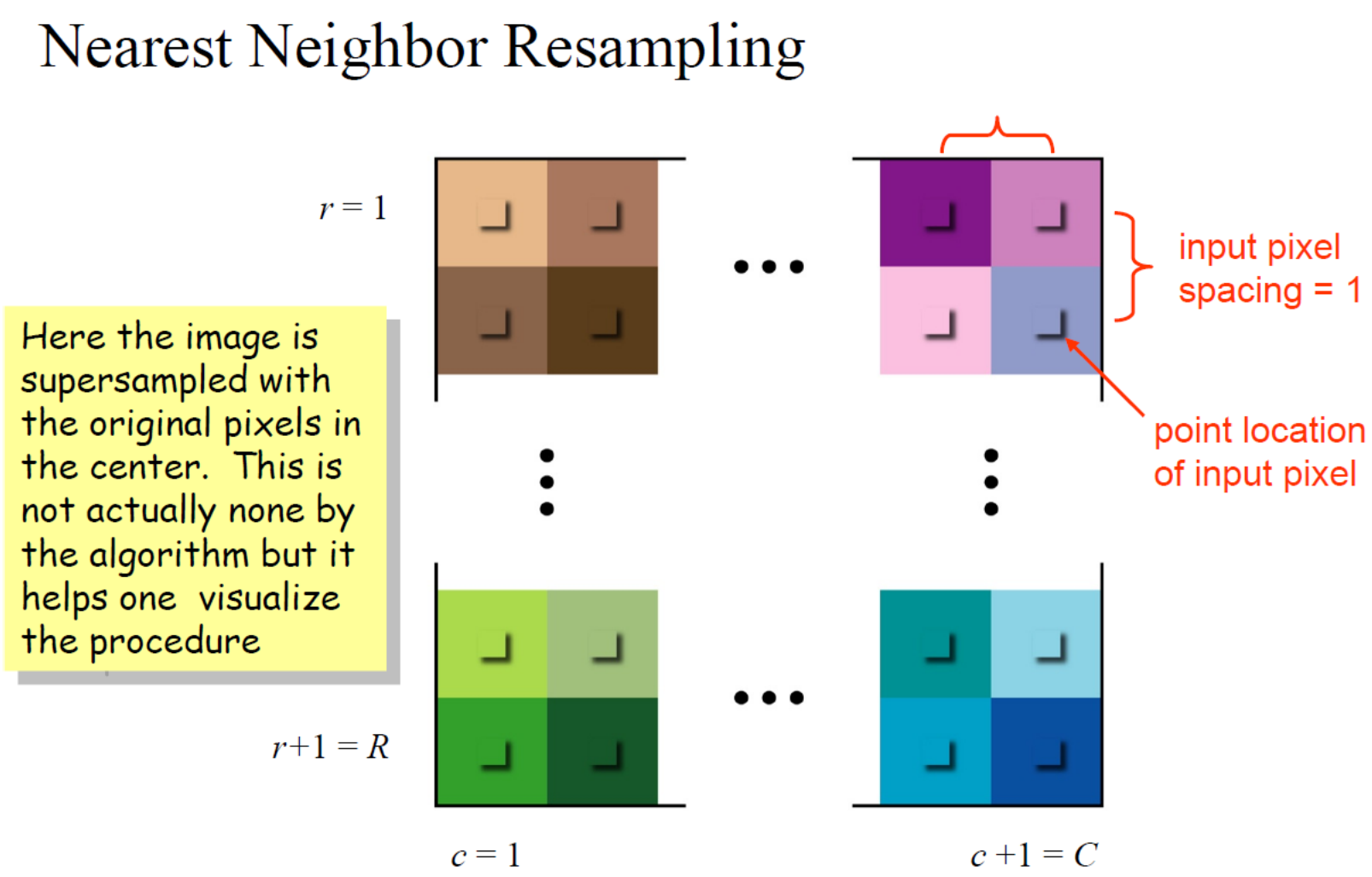
Nearest Neighbor Resampling



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

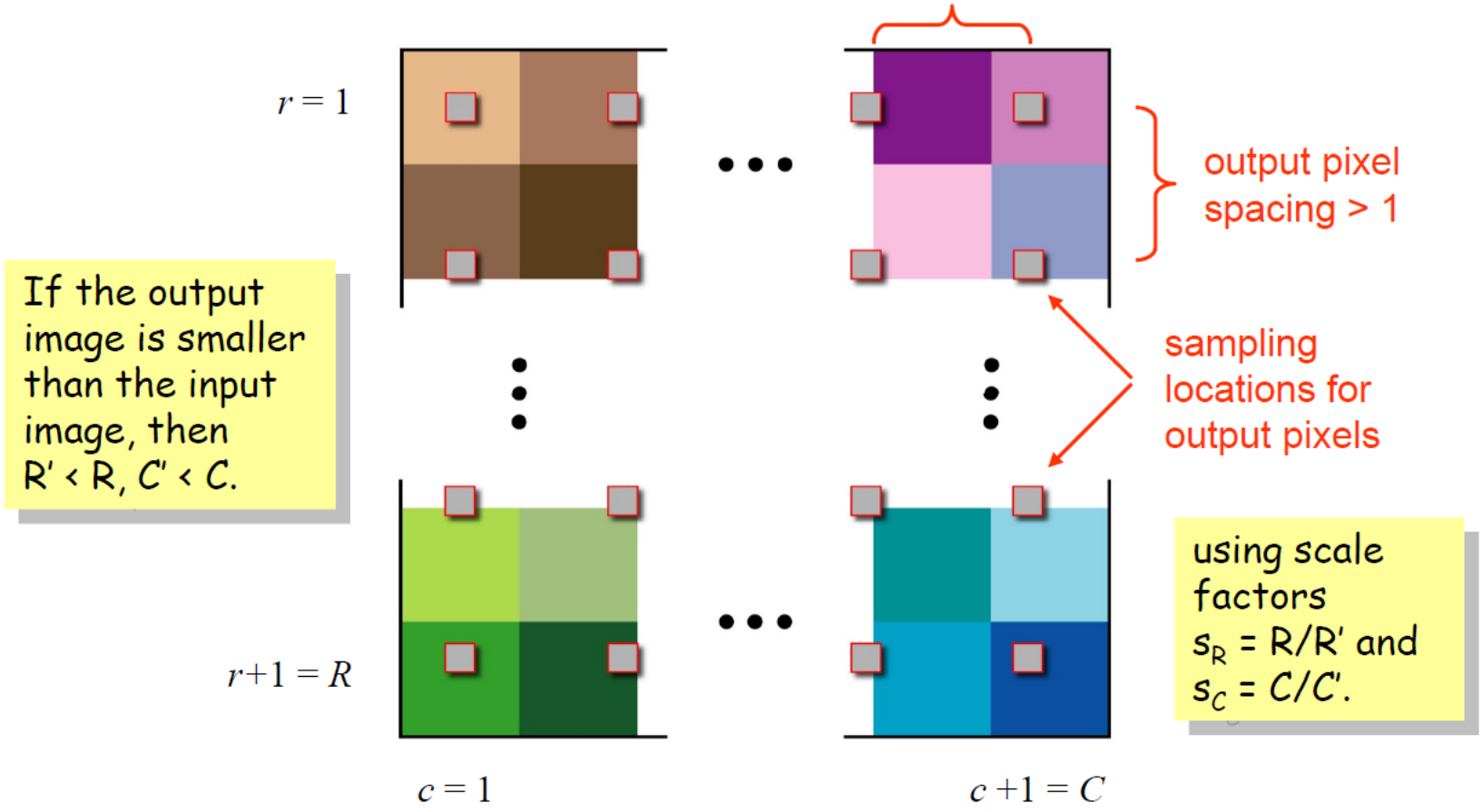
They are also the extrema of 4-pixel neighborhoods in the image.

Resizing image



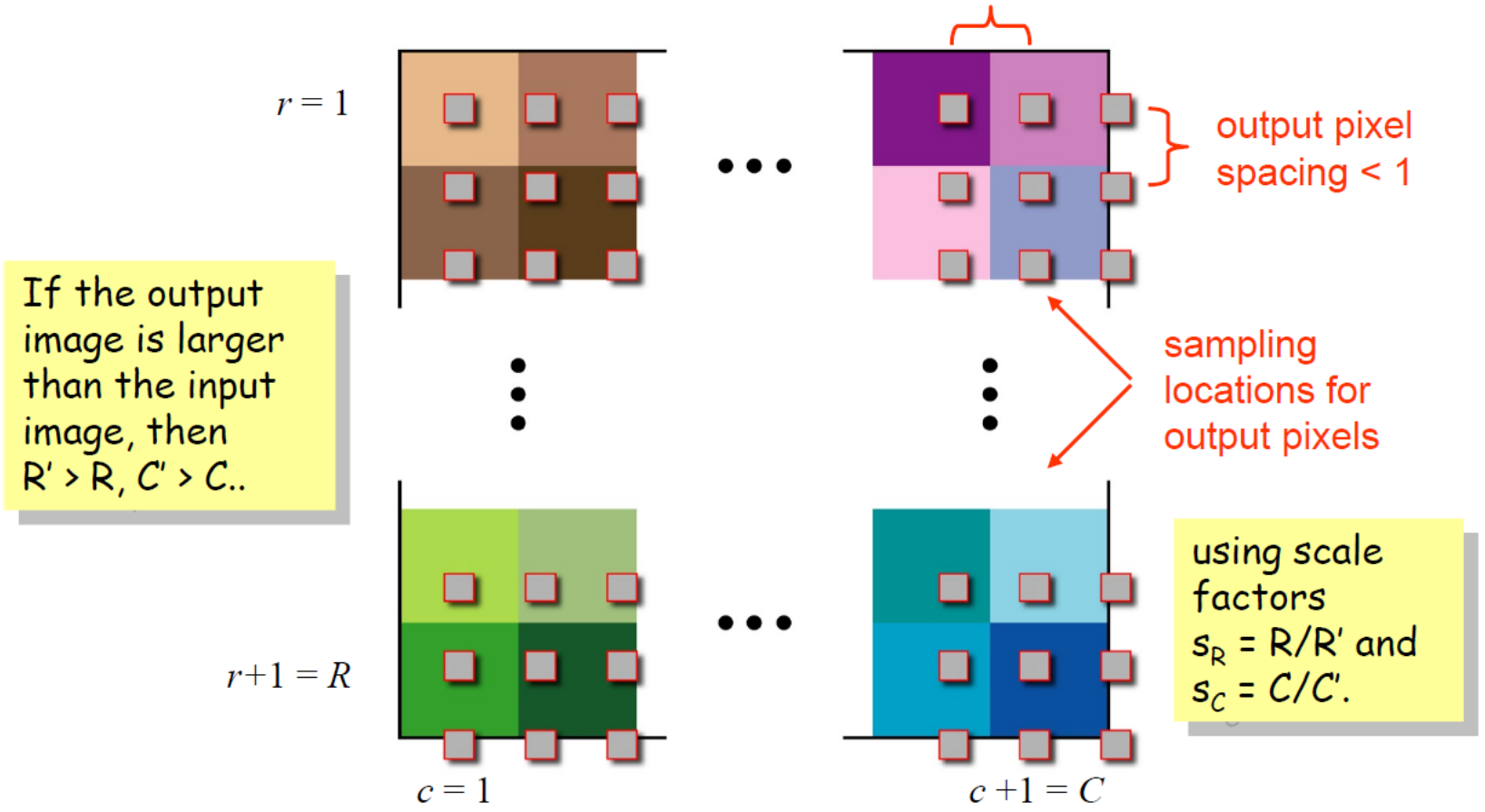
Resizing image

Nearest Neighbor Resampling

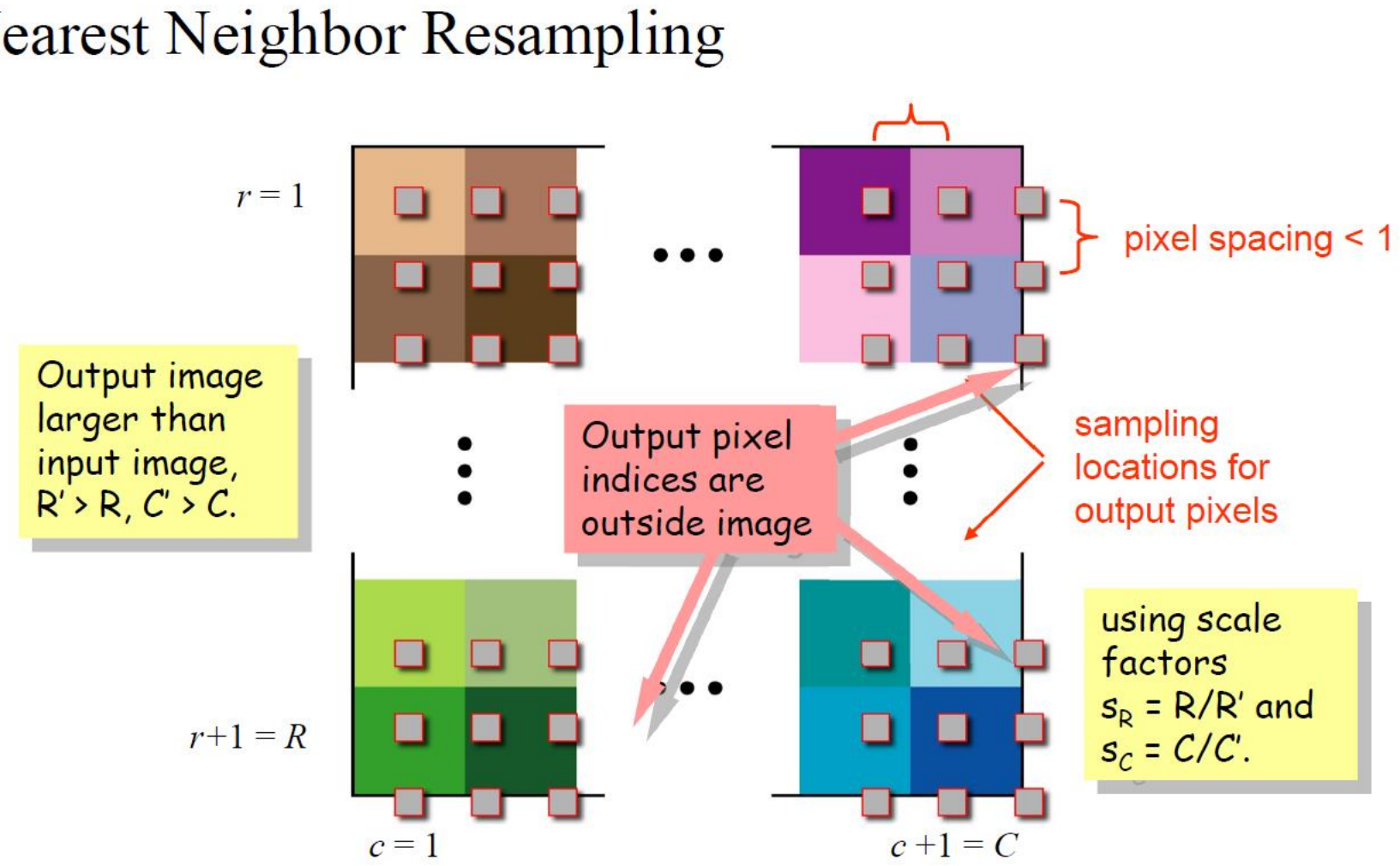


Resizing image

Nearest Neighbor Resampling

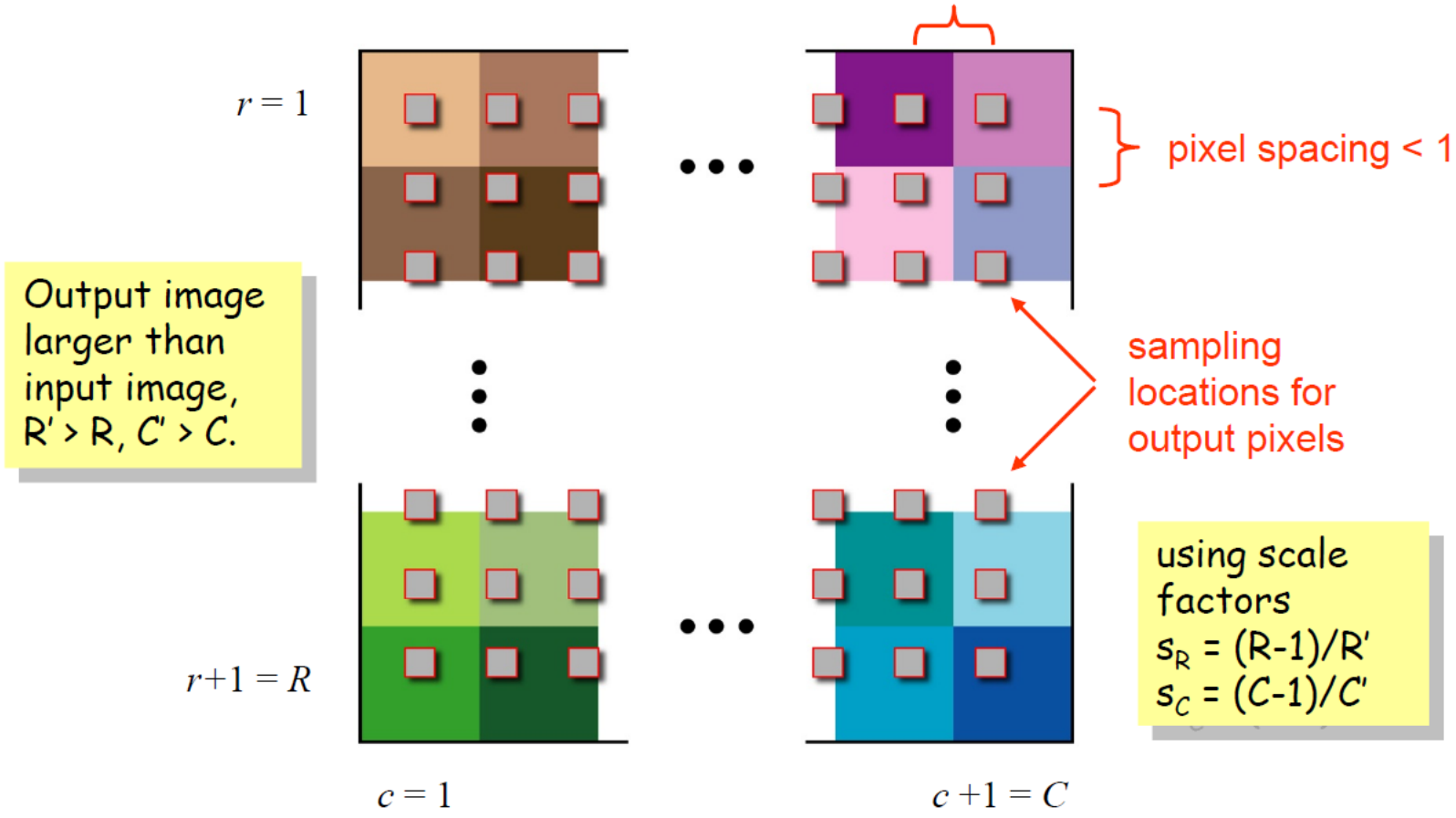


Resizing image



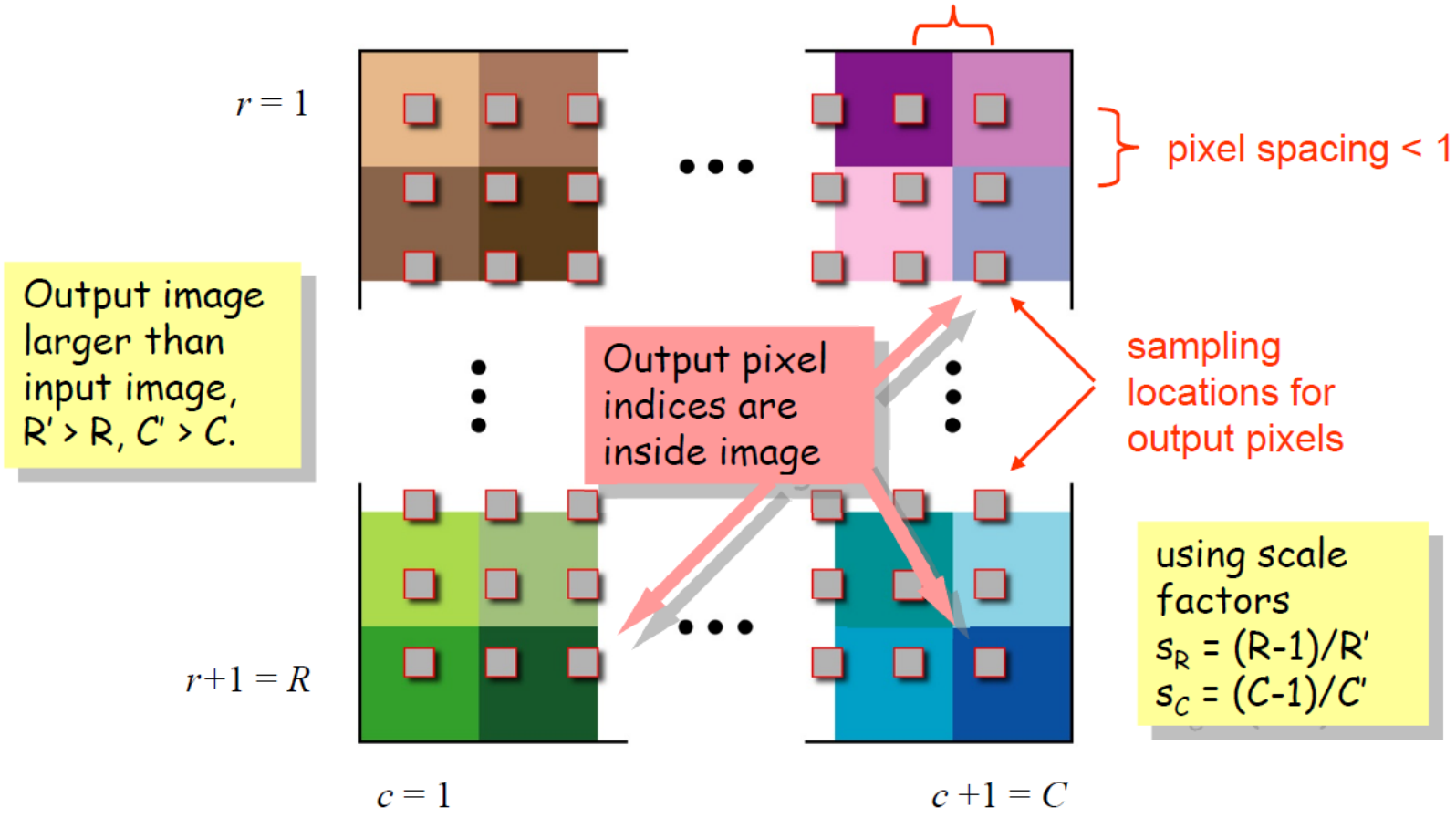
Resizing image

Nearest Neighbor Resampling



Resizing image

Nearest Neighbor Resampling



Resizing image

Resize this image to 3/5 of its original size

Resizing image

```
clear all
close all
p=3;
q=5;
I=imread('Exp_9_1.jpg');
ii=1;
I_new=zeros(size(I,1)*p/q,size(I,2)*p/q,3);
for i=1:q:size(I,1)-1
    jj=1;
    for j=1:q:size(I,2)-1
        I_new(ii:ii+p-1,jj:jj+p-1,:)=double(I(i:2:i+q-1,j:2:j+q-1,:));
        jj=jj+p;
    end
    ii=ii+p;
end
figure
image(I)
truesize
figure
image(uint8(I_new))
truesize
```

Function imresize

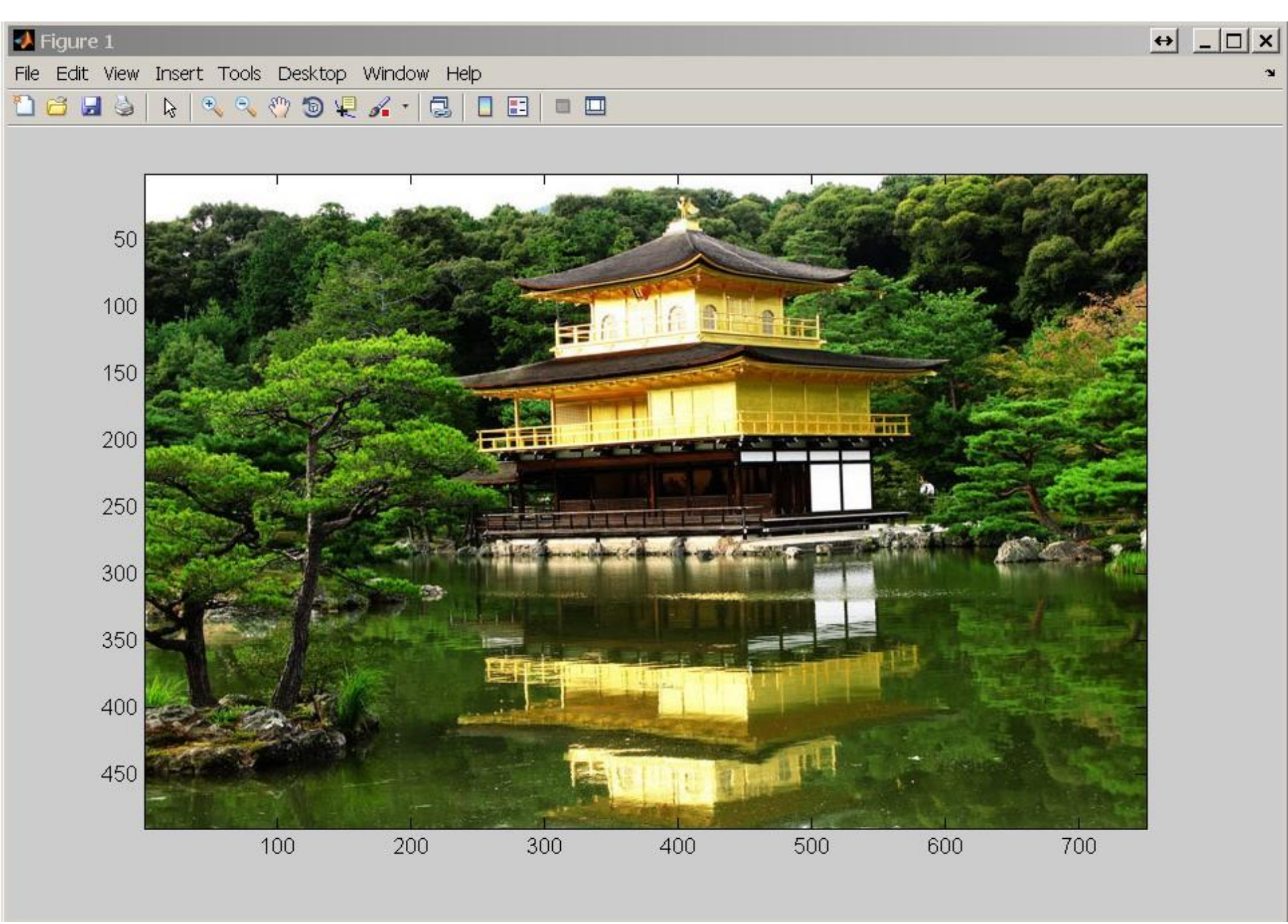
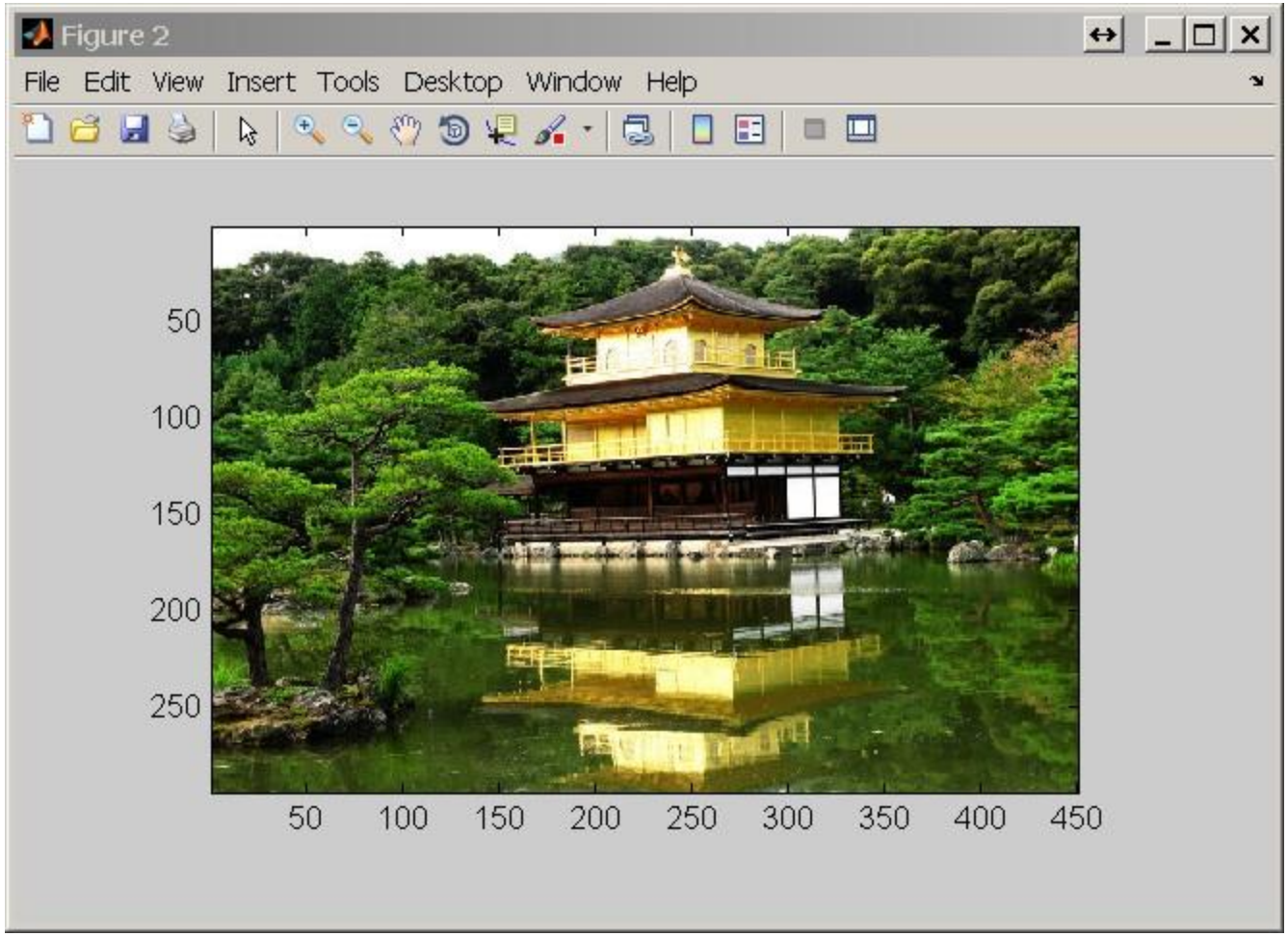
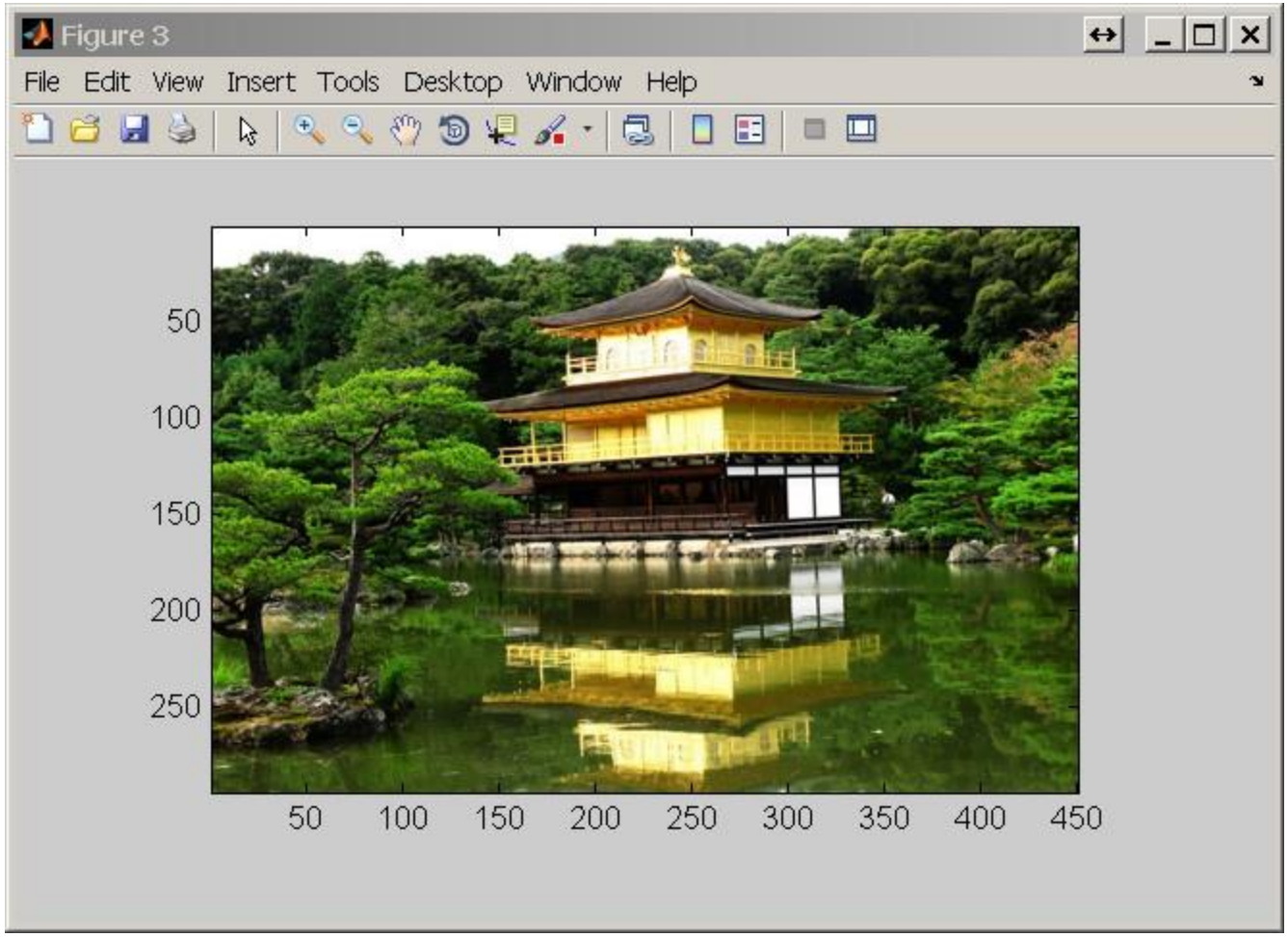


Figure 1
File Edit View Insert Tools Desktop Window Help
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600
700

Resizing image



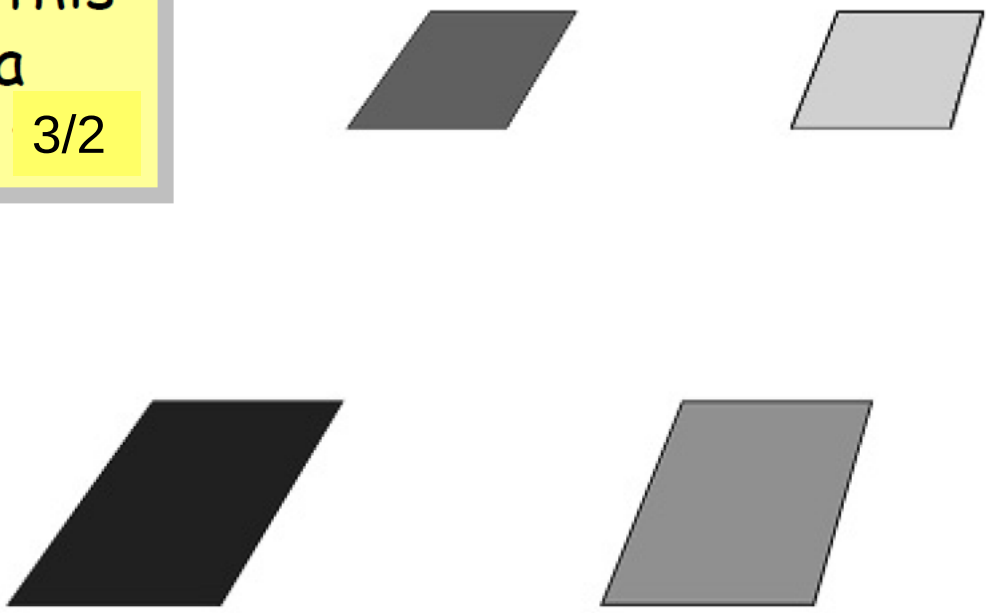
Resizing image



Resizing image

Resampling Through Bilinear Interpolation

want to upsample this image by a factor of 3/2



Resizing image

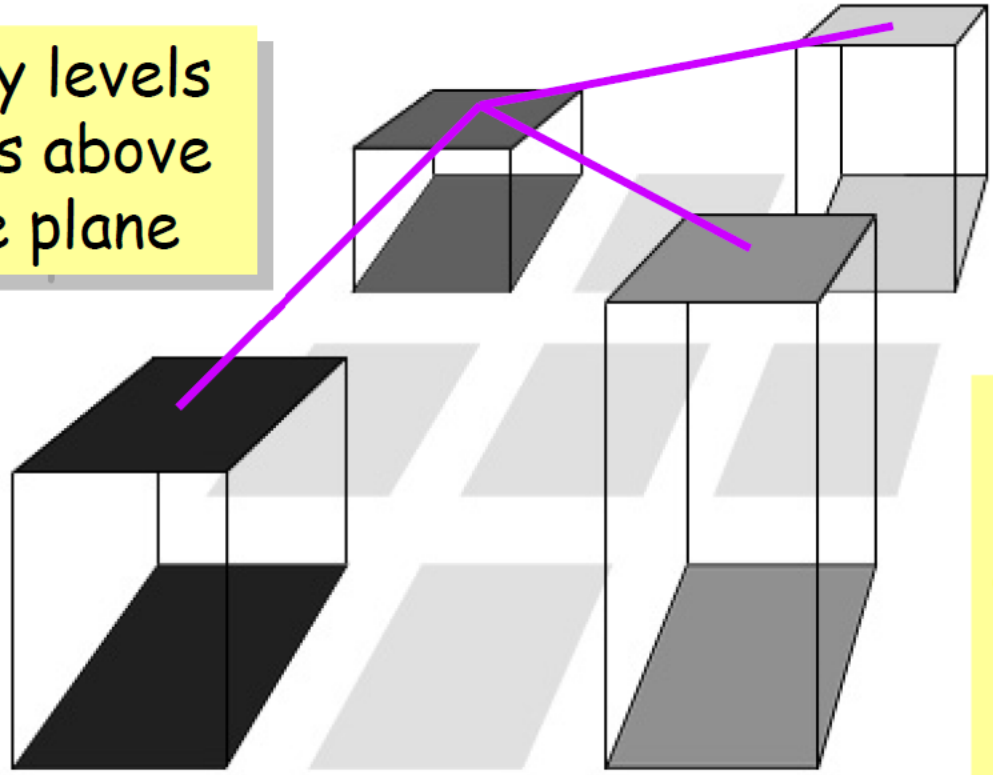
Resampling Through Bilinear Interpolation

new samples to be added at light gray locations.

Resizing image

Resampling Through Bilinear Interpolation

treat gray levels as heights above the image plane

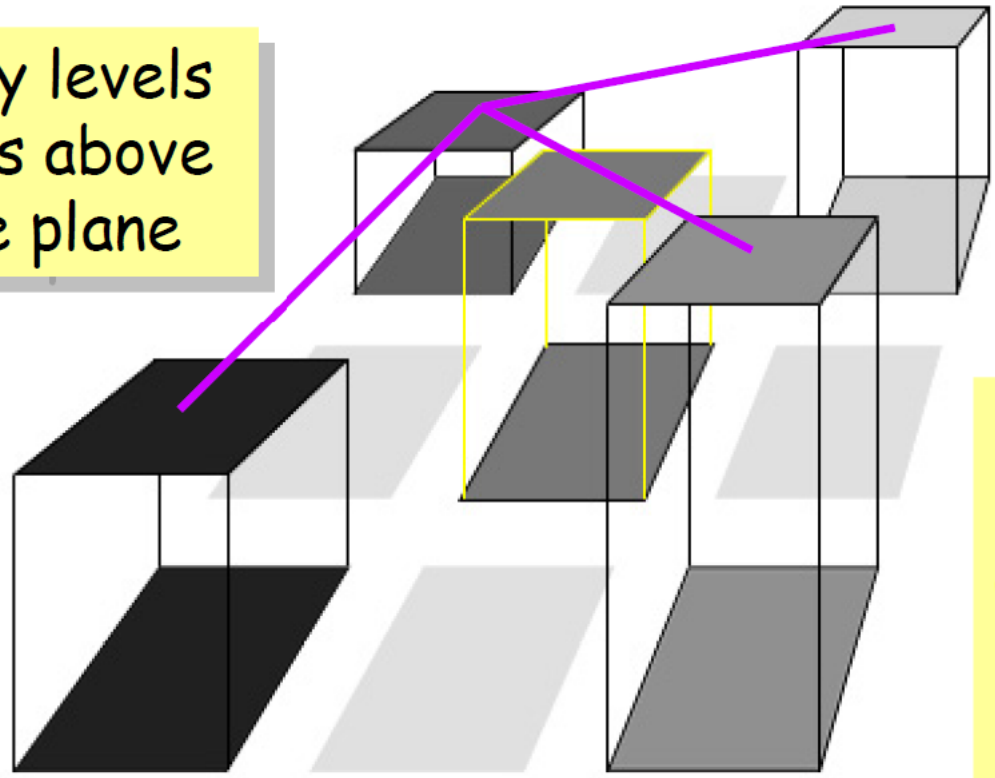


center = weighted average of four corners

Resizing image

Resampling Through Bilinear Interpolation

treat gray levels as heights above the image plane

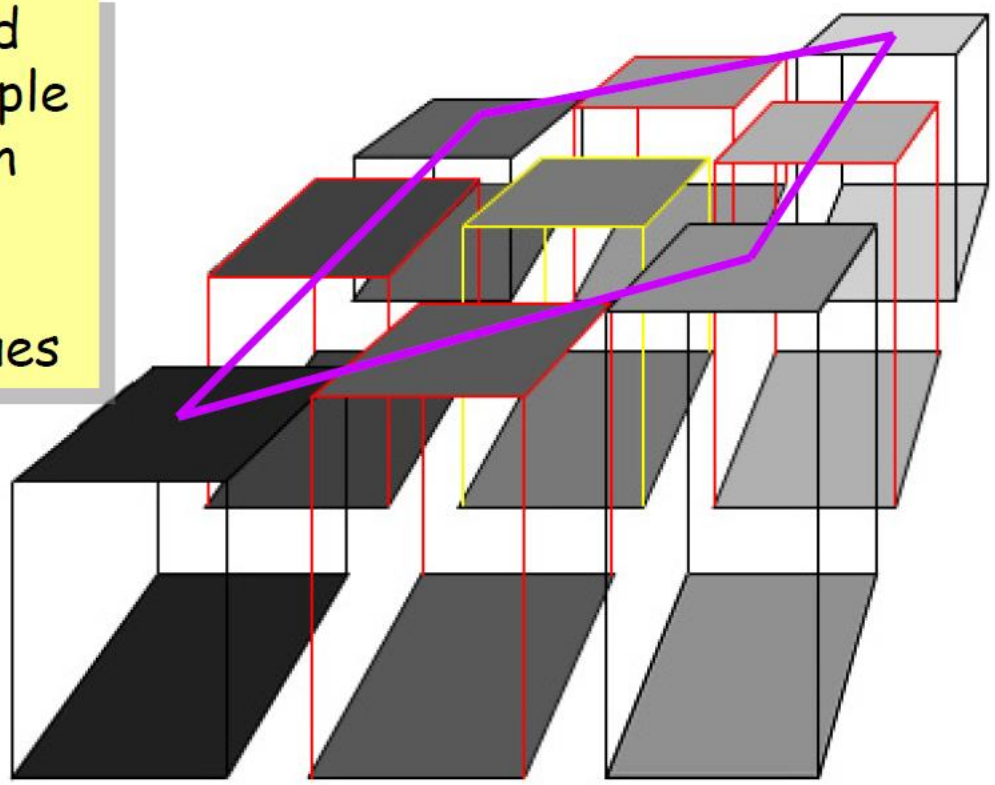


center = weighted average of four corners

Resizing image

Resampling Through Bilinear Interpolation

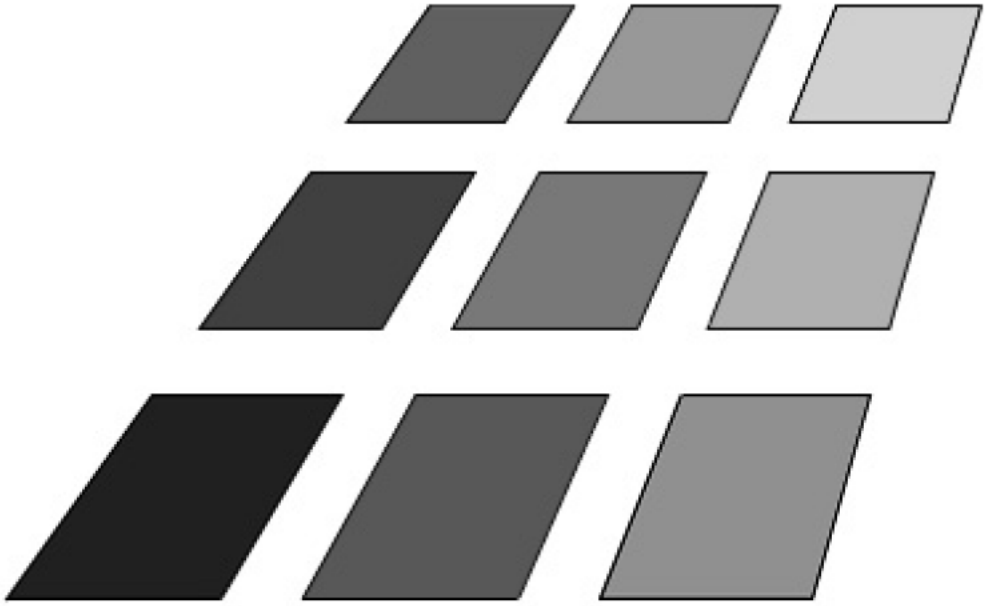
new row and column sample values lie on the lines connecting the old values



Resizing image

Resampling Through Bilinear Interpolation

the results



Resizing image

Bilinear Interpolation Example

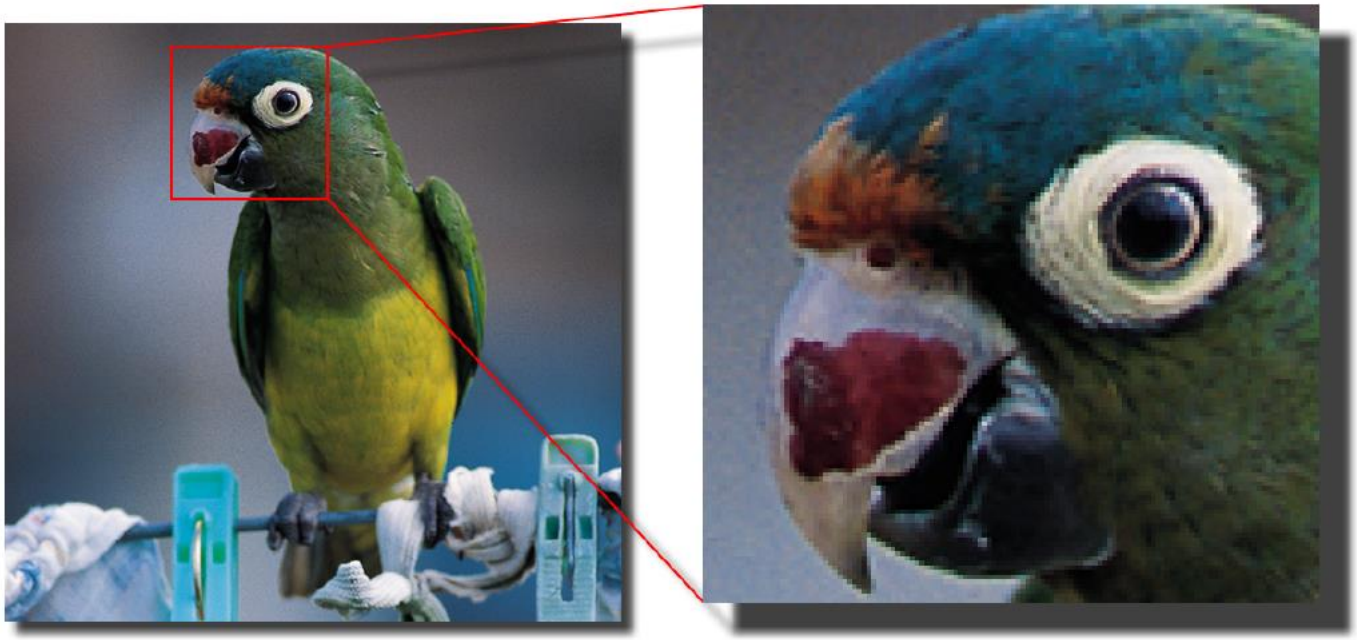
We'll enlarge this image by a factor of 4 ...

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

Resizing image

Example: Bilinear Interpolation – 4× Zoom

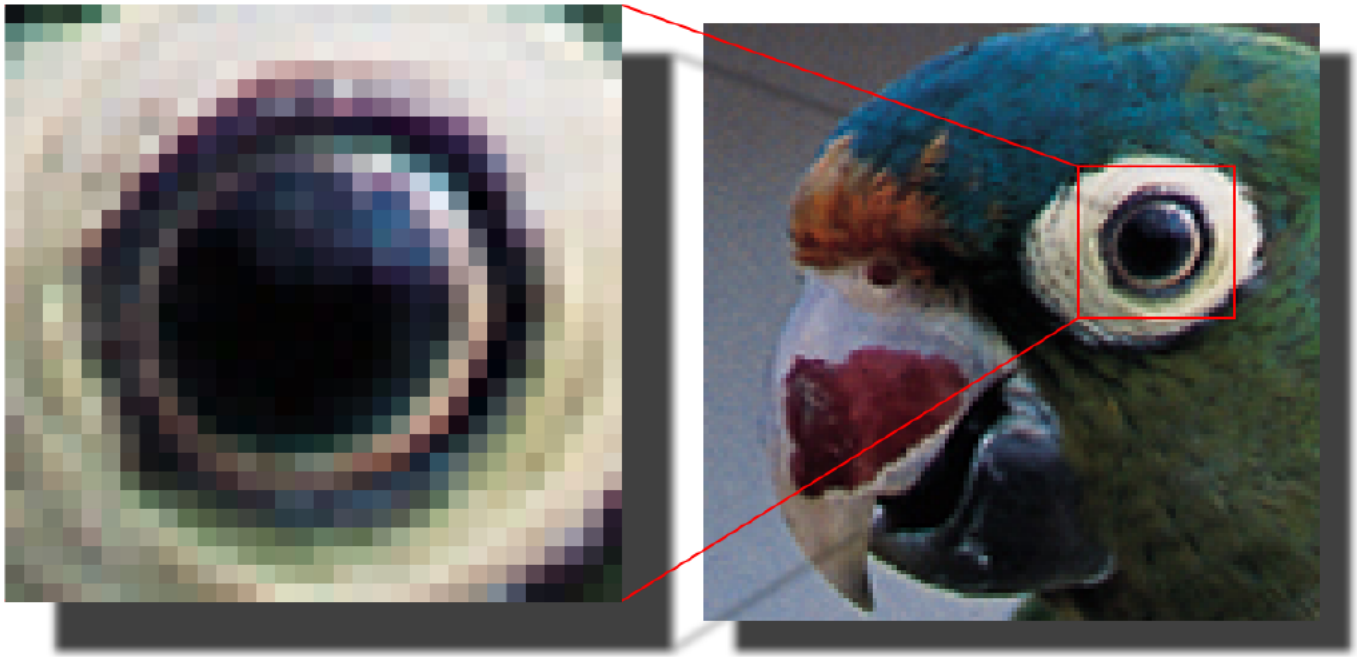
Original Image



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

Resizing image

Example: Bilinear Interpolation – 4× Zoom



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

Resizing image

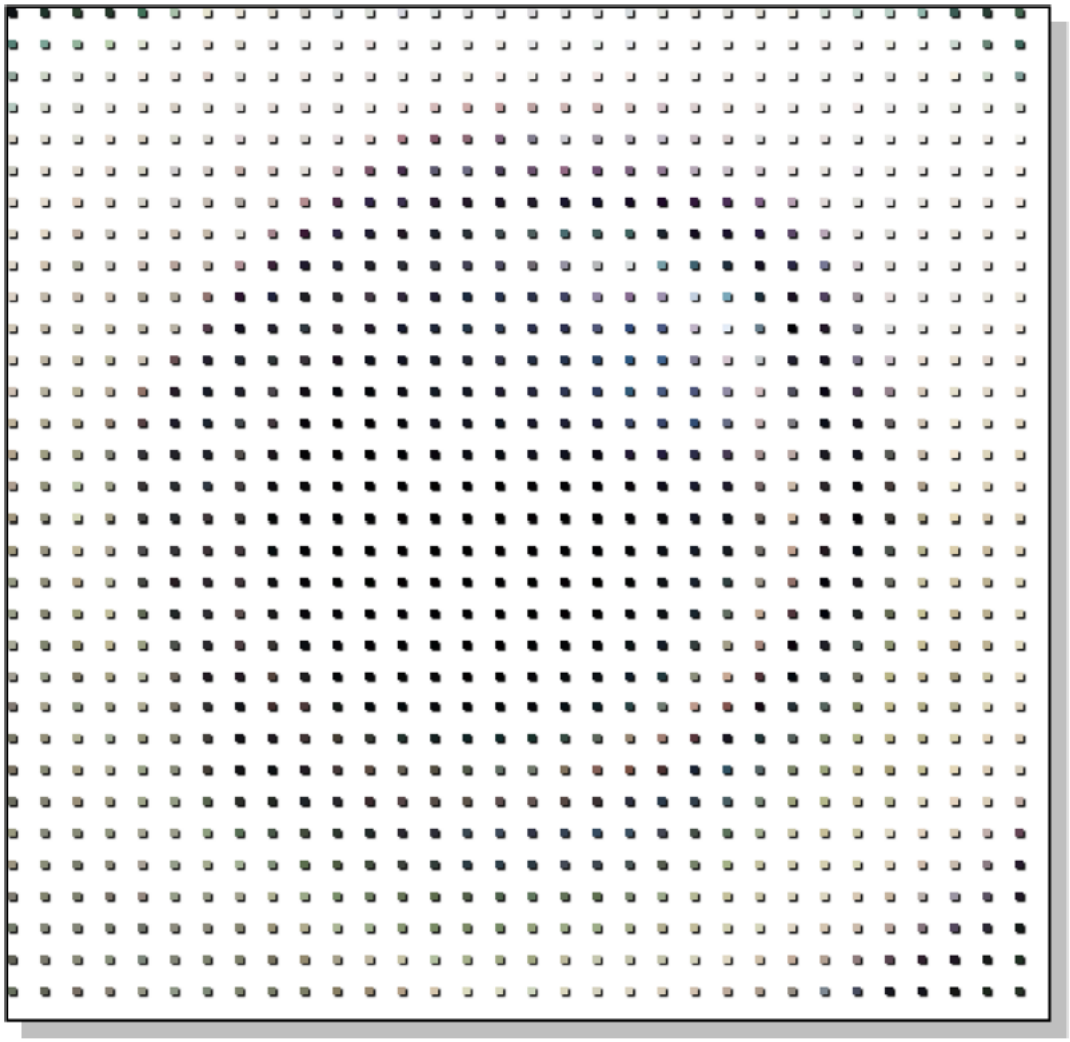
Bilinear Interpolation

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

Resizing image

Bilinear Interpolation

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

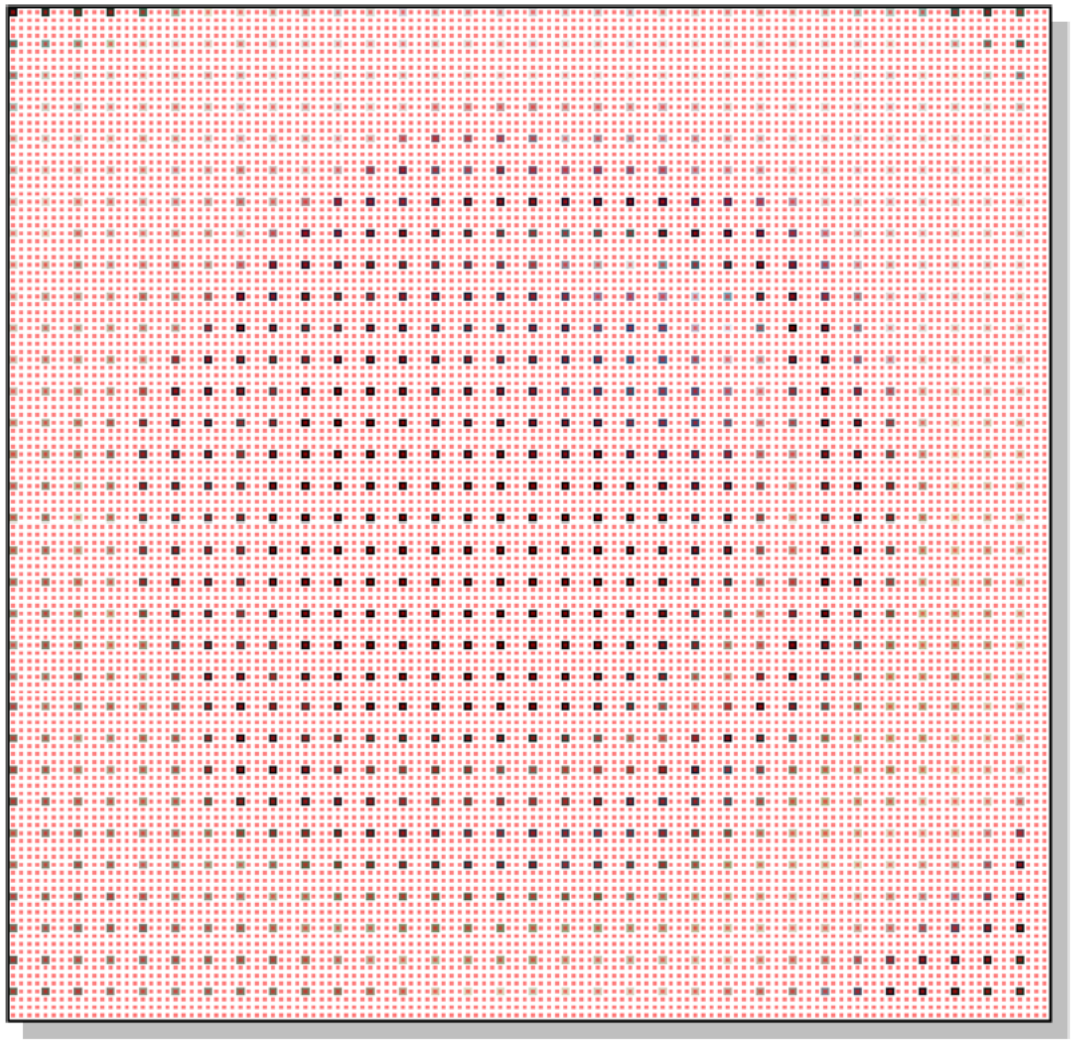


Resizing image

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.

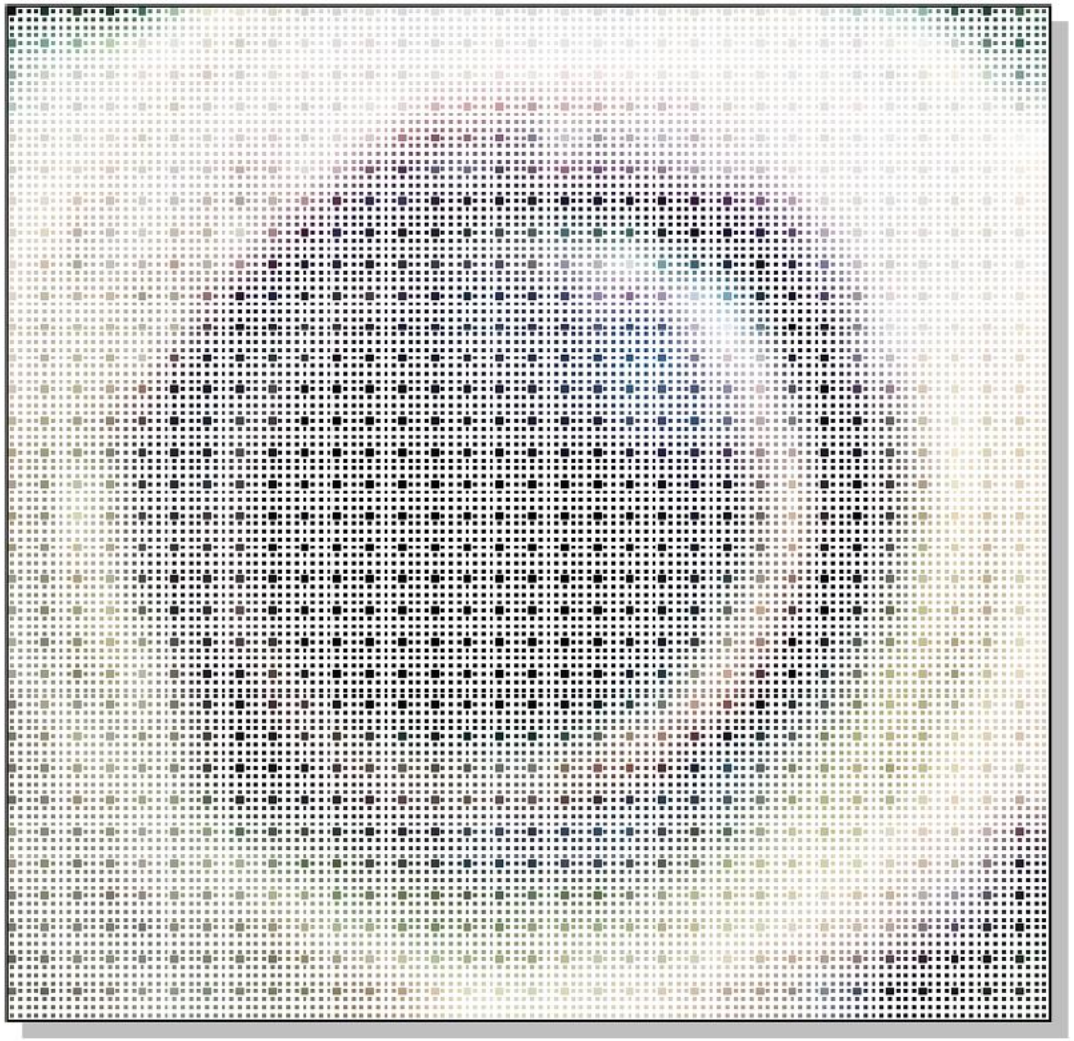


Resizing image

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.

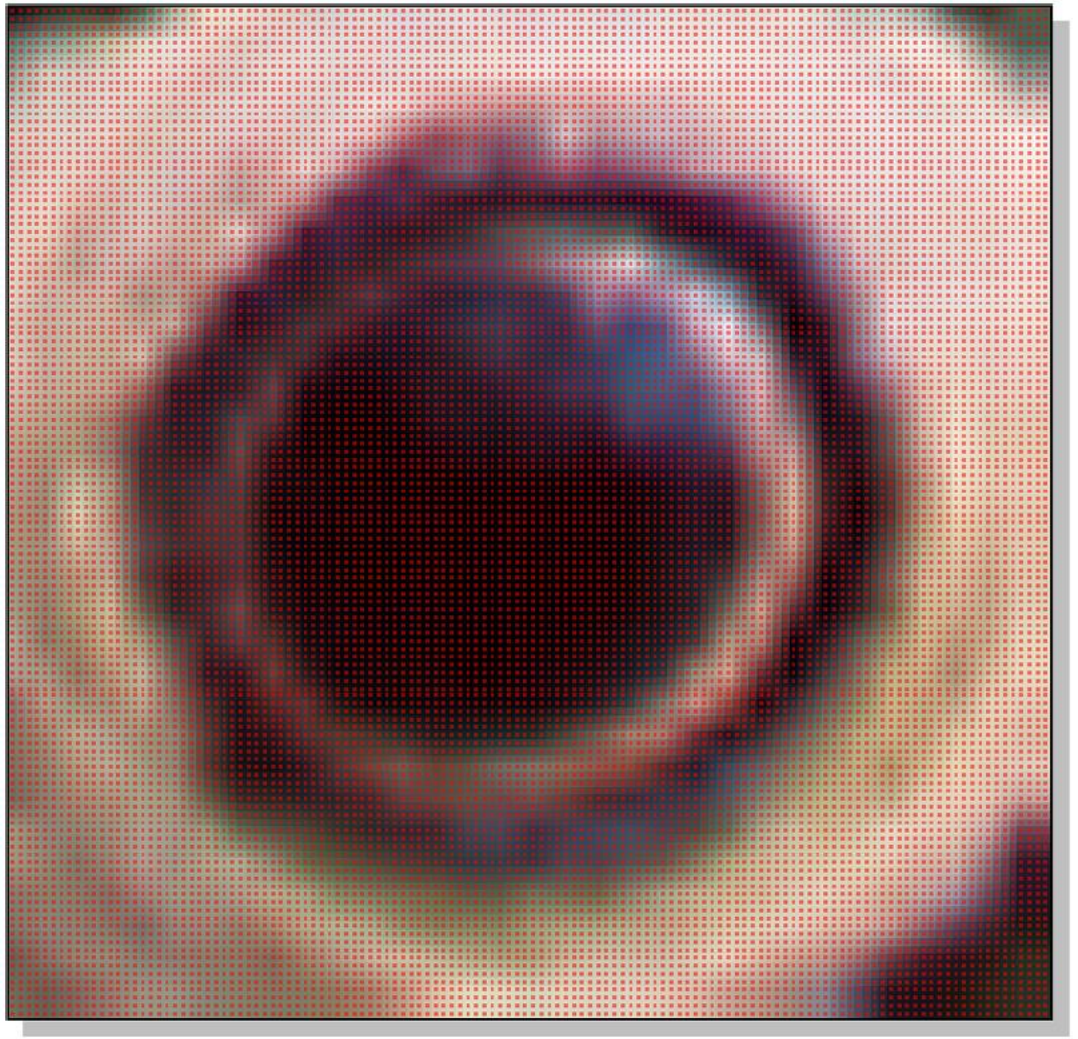


Resizing image

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.



Resizing image

Bilinear Interpolation

The result:

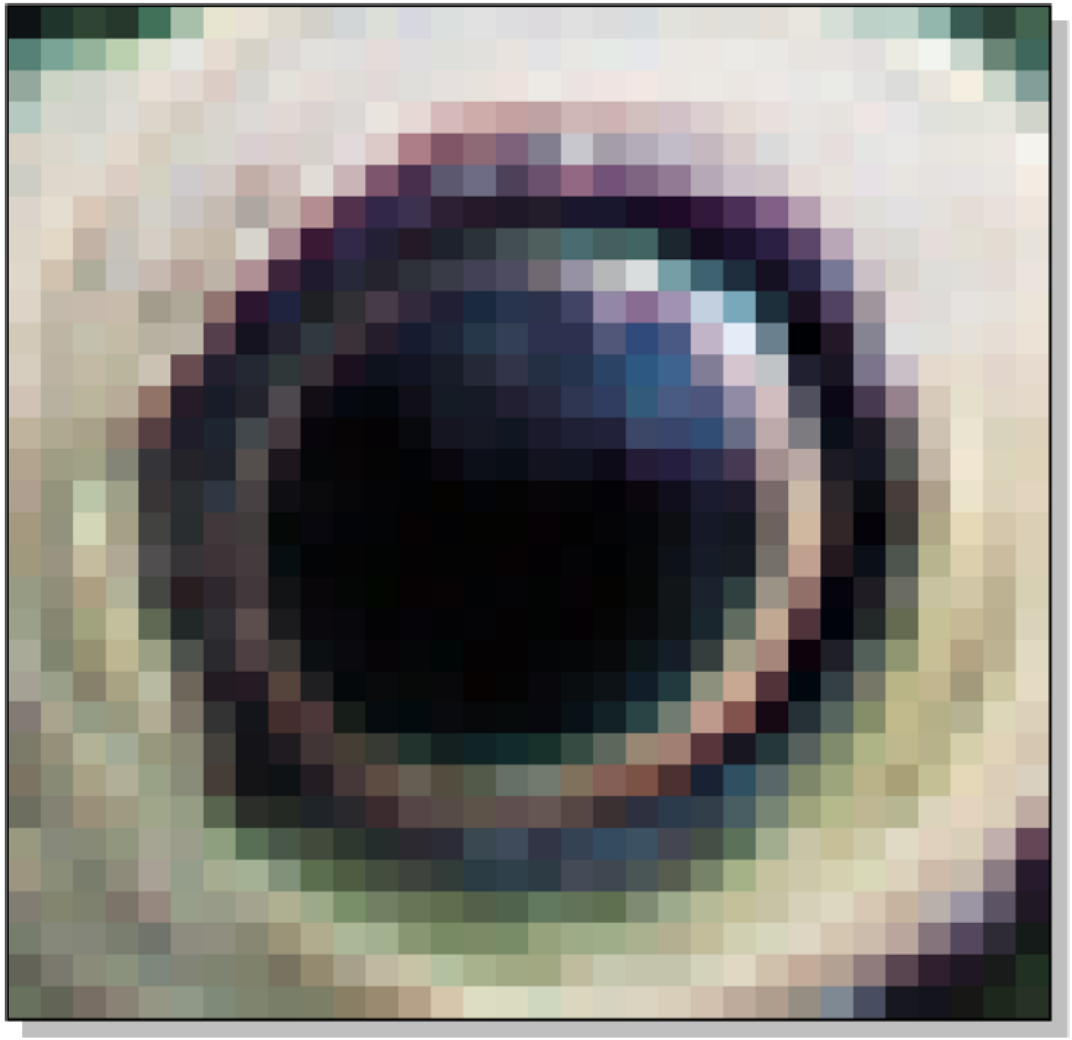
Compare to the next slide which contains a 4x pixel zoom via pixel replication.

Resizing image

Pixel Replication

The result:

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

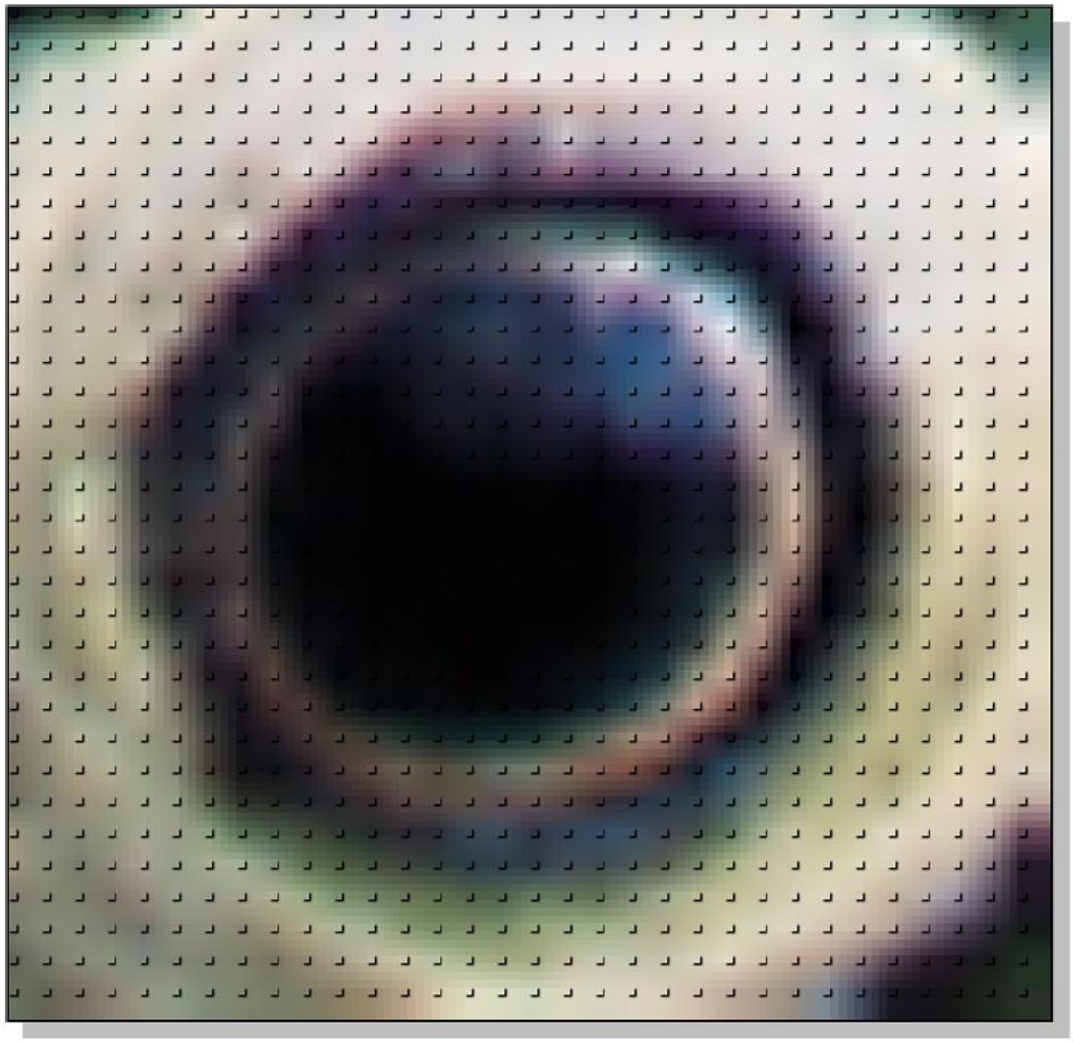


Resizing image

Bilinear Interpolation

Result with original pixels marked:

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

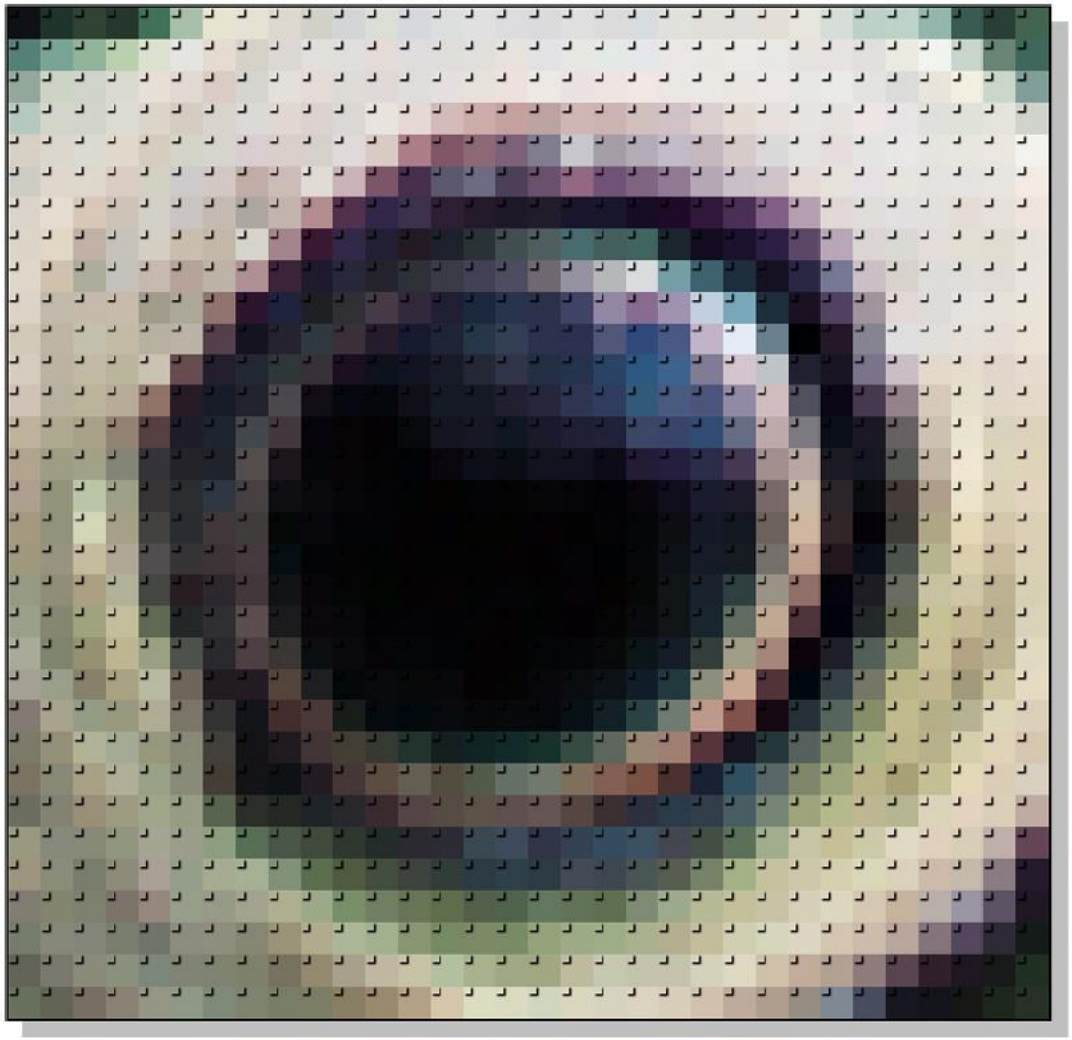


Resizing image

Pixel Replication

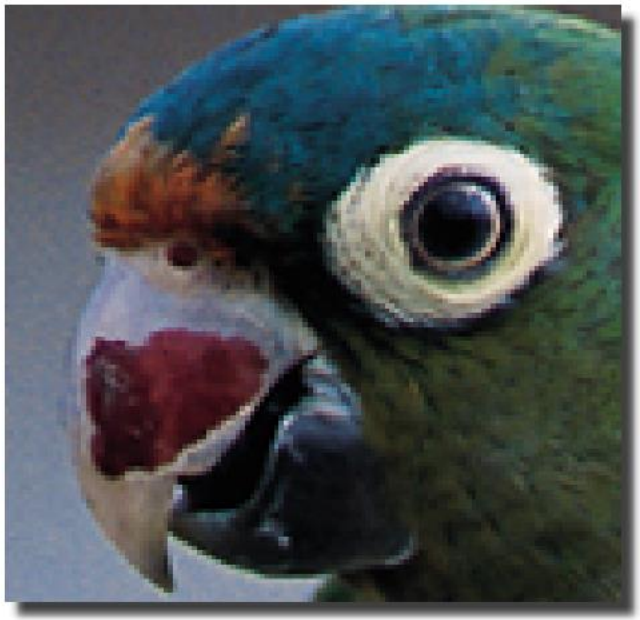
Result with original pixels marked:

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

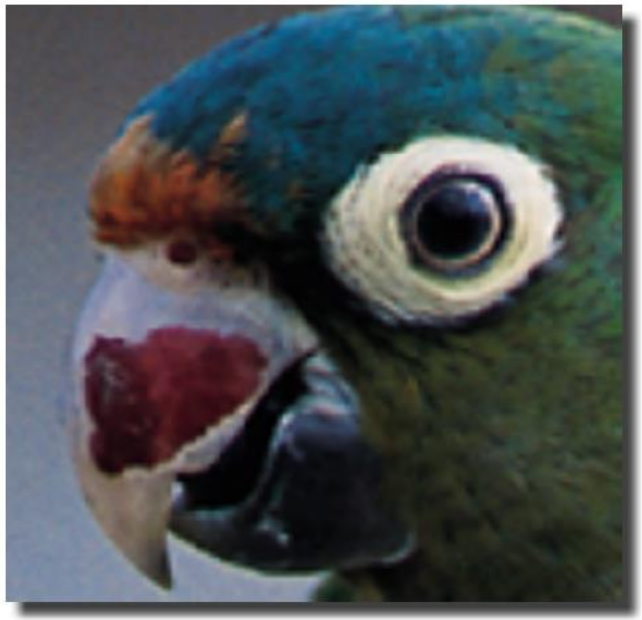


Resizing image

Pixel Replication vs. Bilinear Interpolation



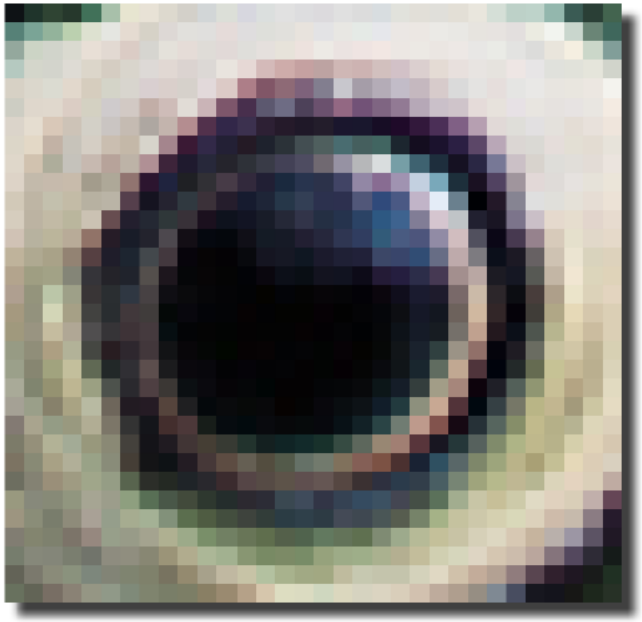
Pixel replication



Bilinear interpolation

Resizing image

Pixel Replication vs. Bilinear Interpolation



Pixel replication

Bilinear interpolation

Resizing image

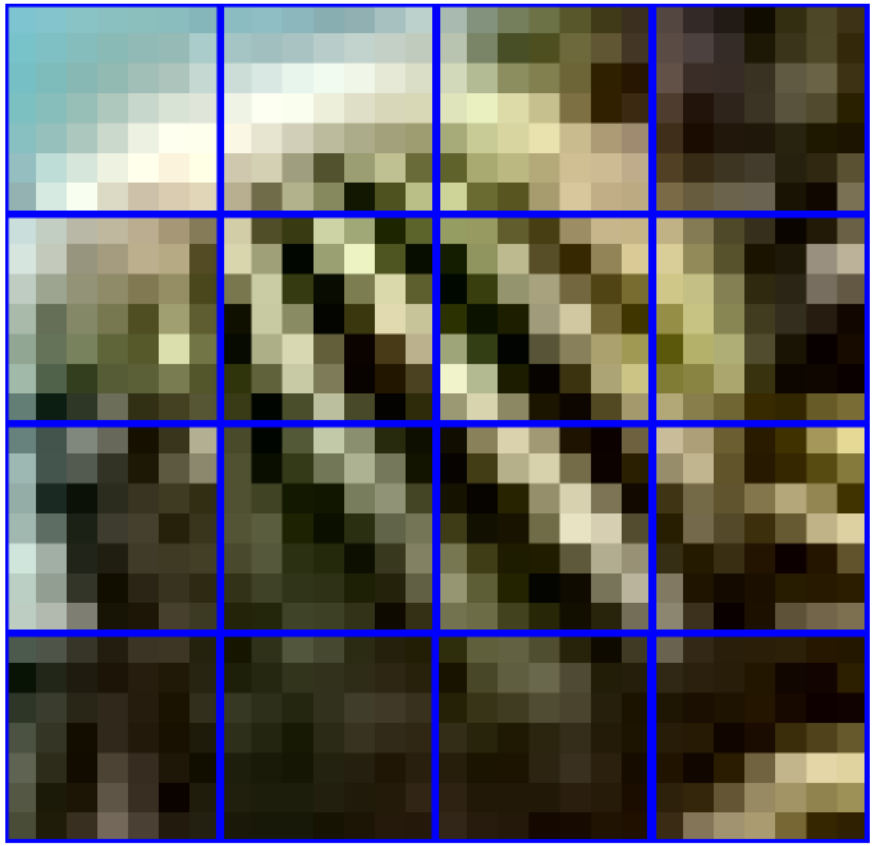
Resampling Through Bilinear Interpolation

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

Resizing image

Resampling Through Bilinear Interpolation

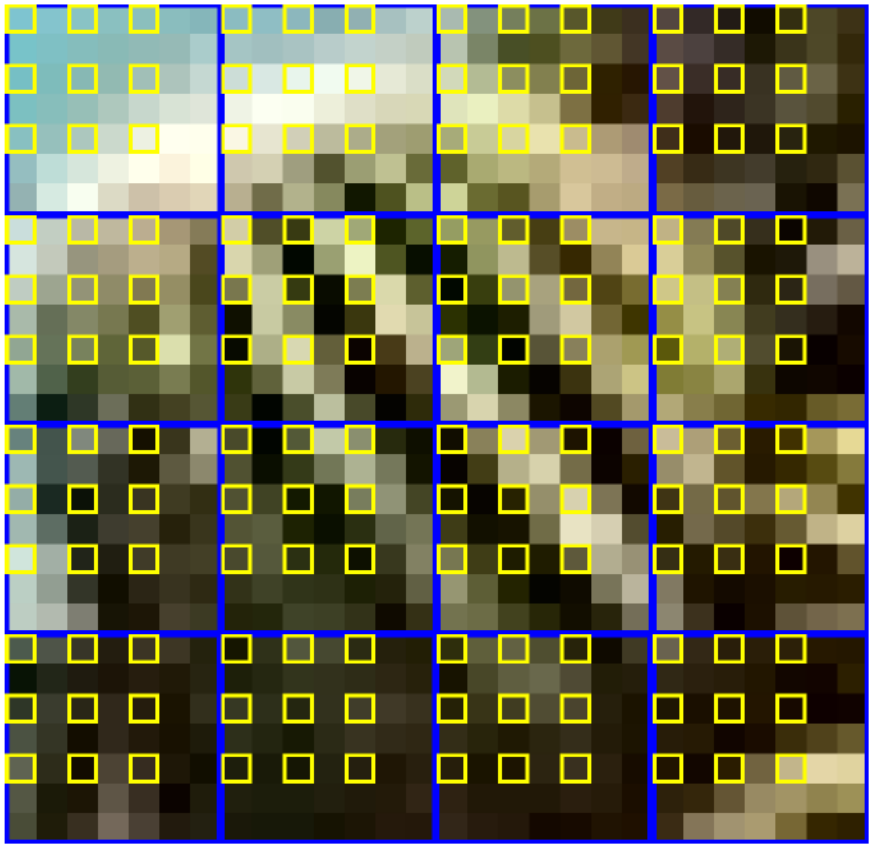
For each 7x7 block of pixels select 3x3 = 9 pixel locations.



Resizing image

Resampling Through Bilinear Interpolation

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

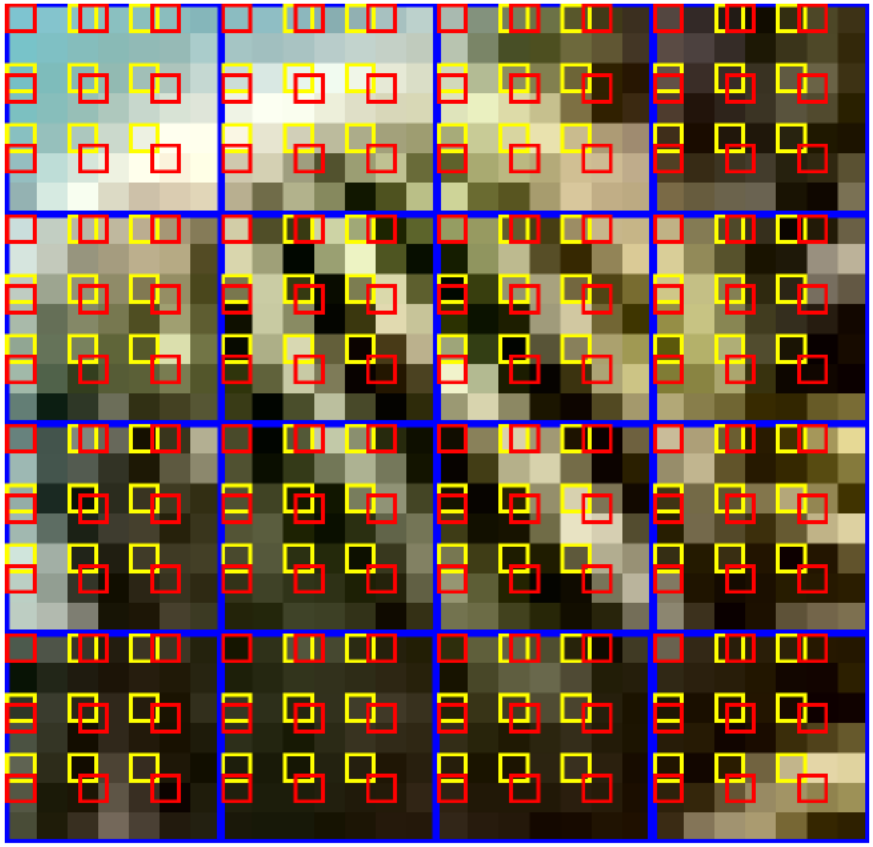


Nearest neighbor selected pixels outlined in yellow.

Resizing image

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.

Resizing image

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.

Resizing image

Resampling Through Bilinear Interpolation

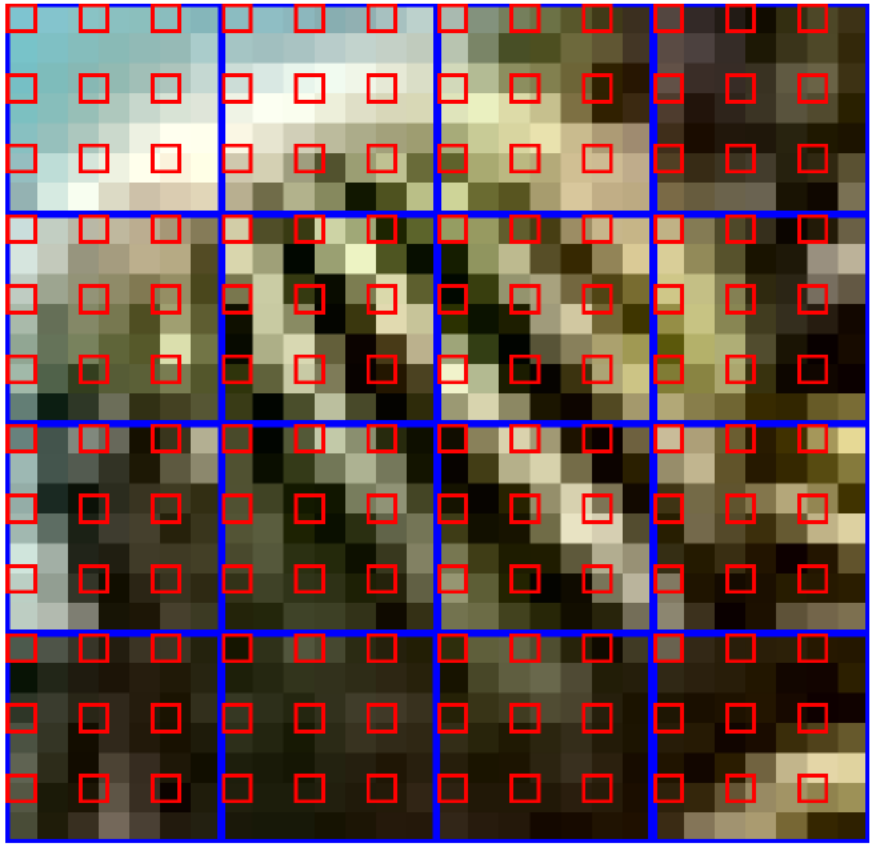
In bilinear interpolation the 9 pixel locations are distributed evenly.

Notice that the locations overlap pixels in the original image.

Resizing image

Resampling Through Bilinear Interpolation

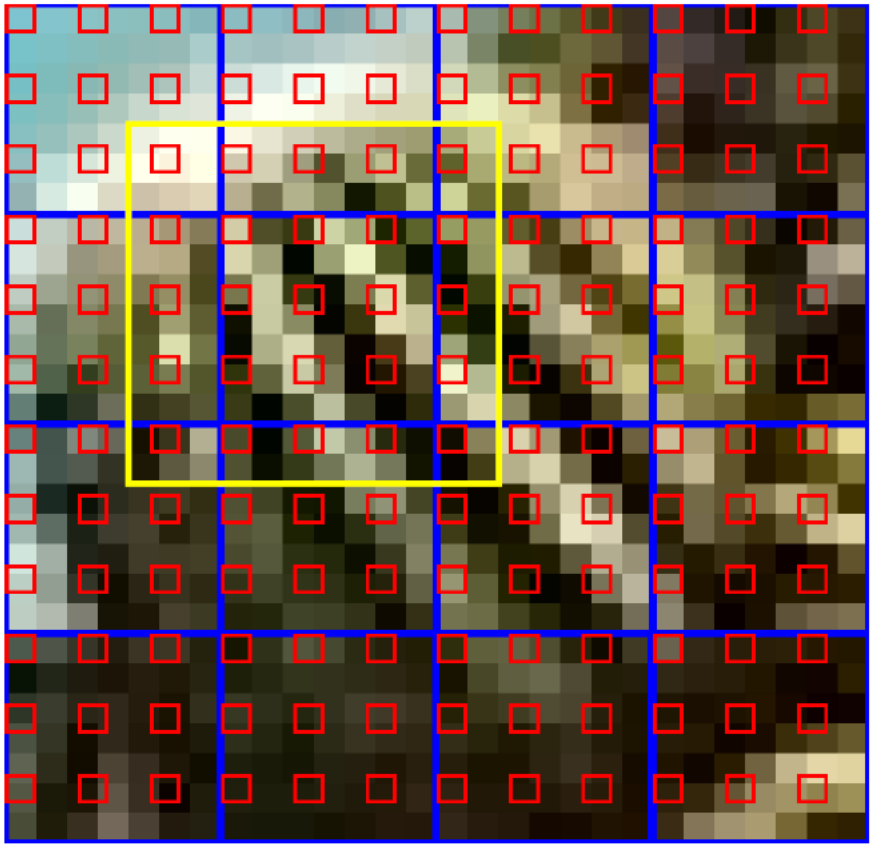
For each 7x7 block of pixels select 3x3 = 9 pixel locations.



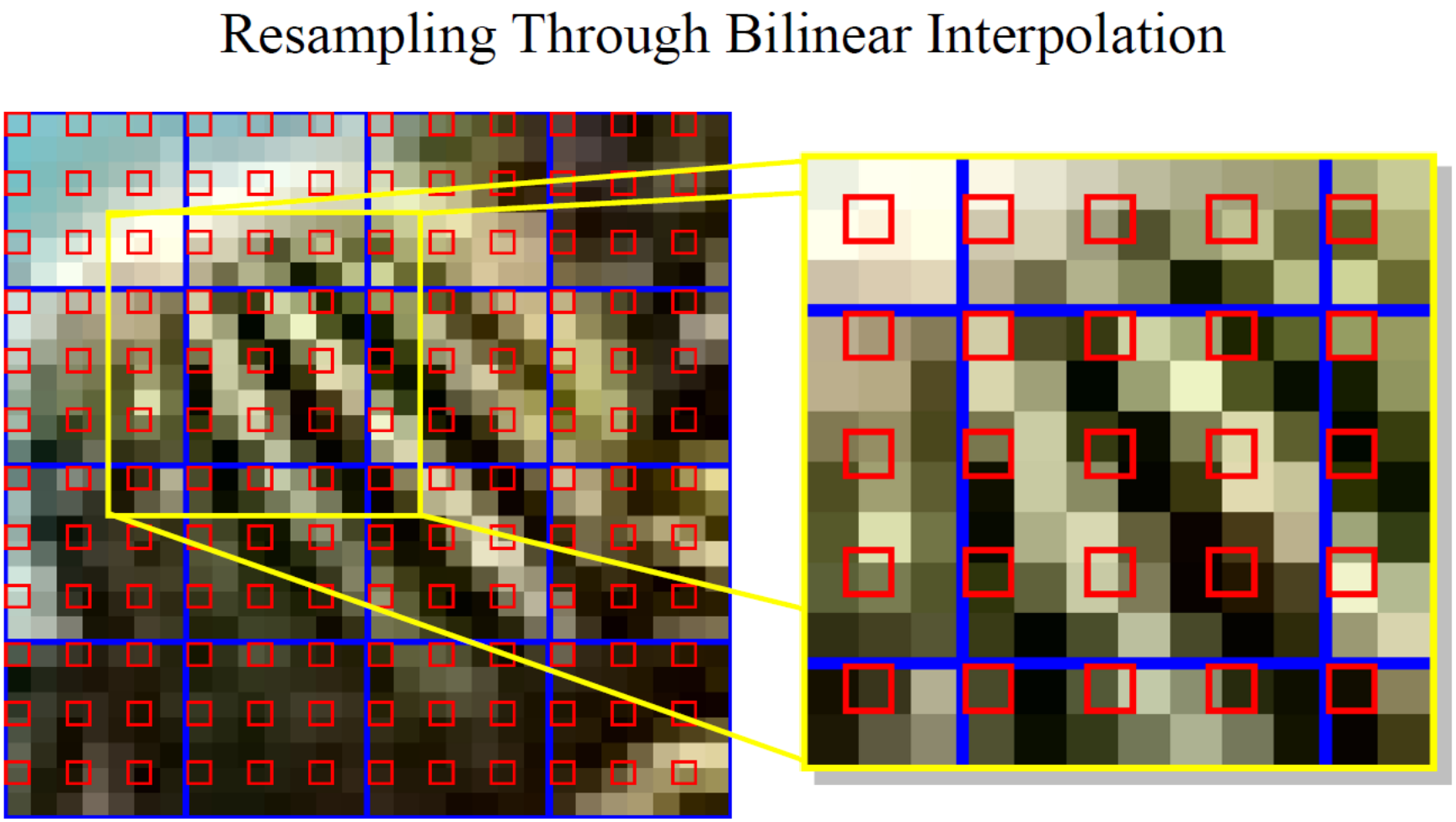
Resizing image

Resampling Through Bilinear Interpolation

Examine one section in detail.

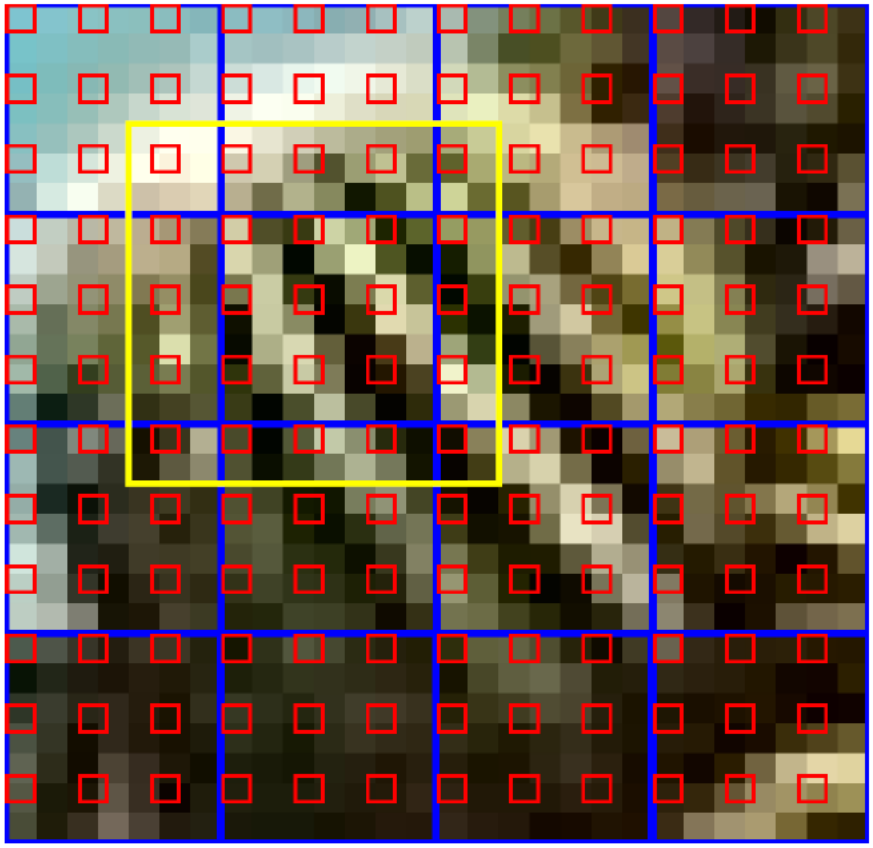


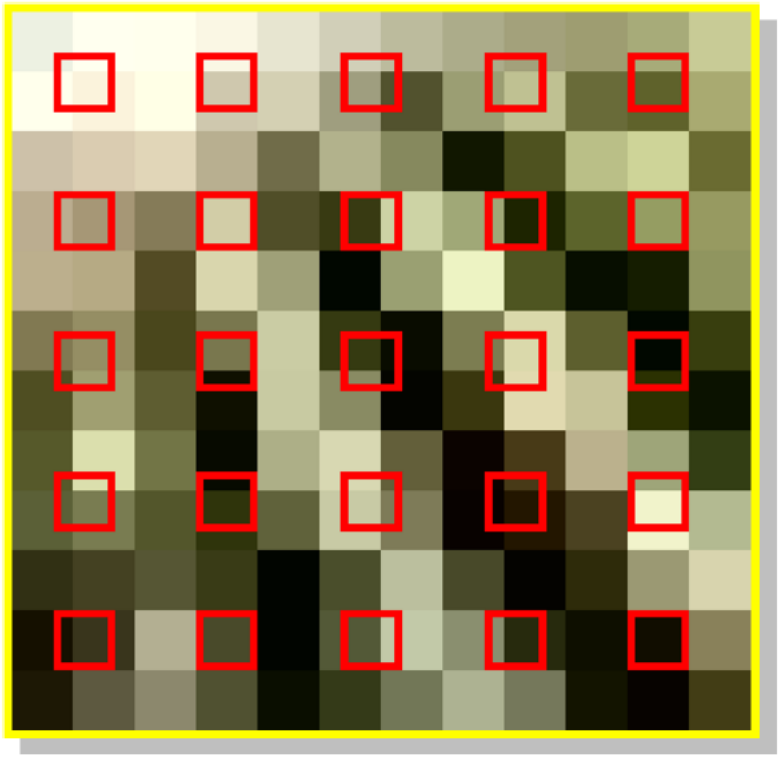
Resizing image



Resizing image

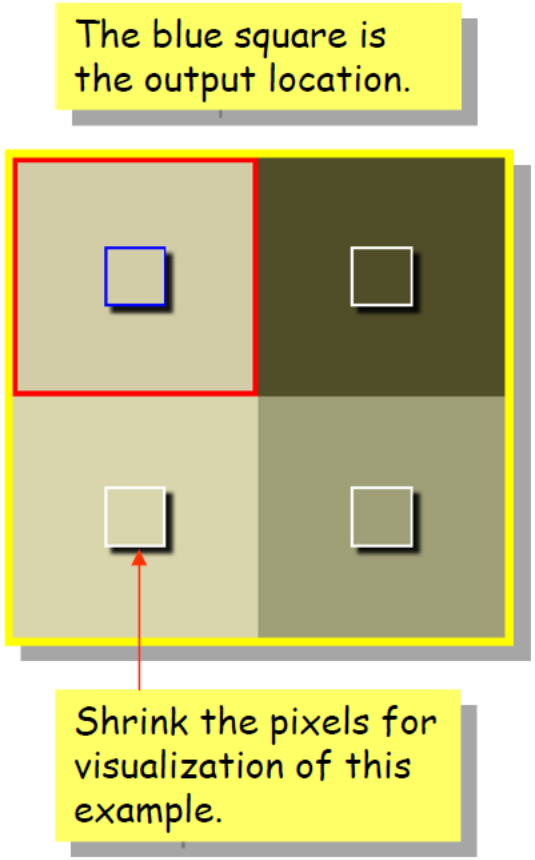
Resampling Through Bilinear Interpolation

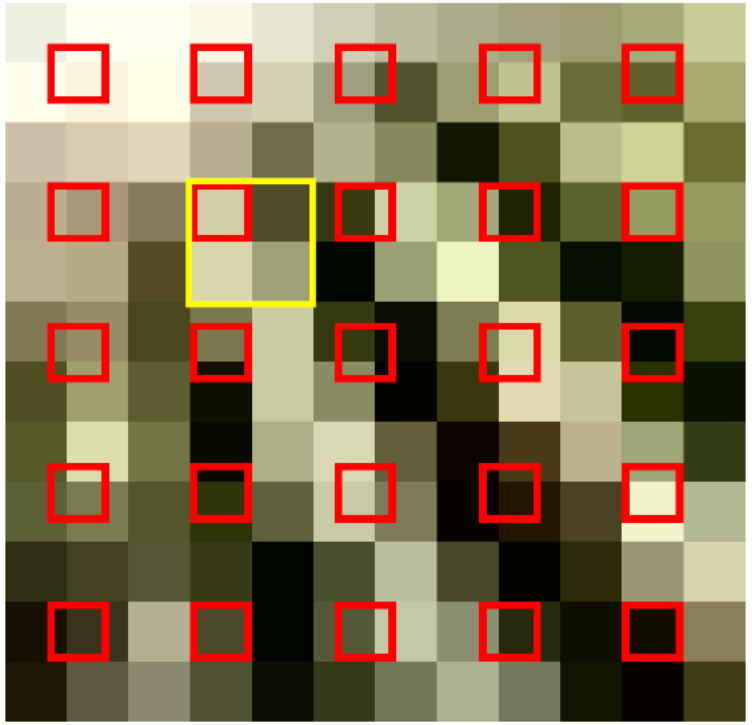




Resizing image

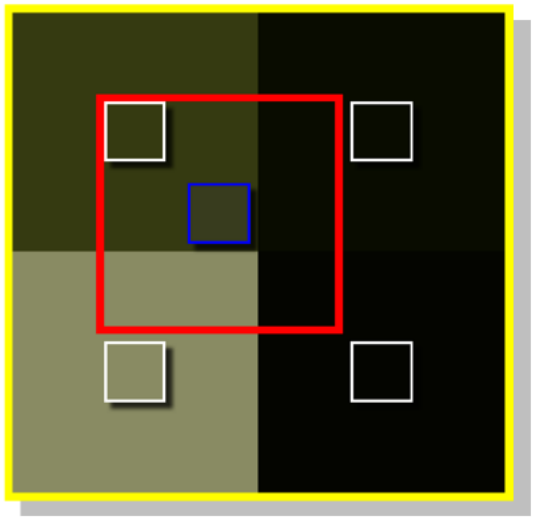
Resampling Through Bilinear Interpolation



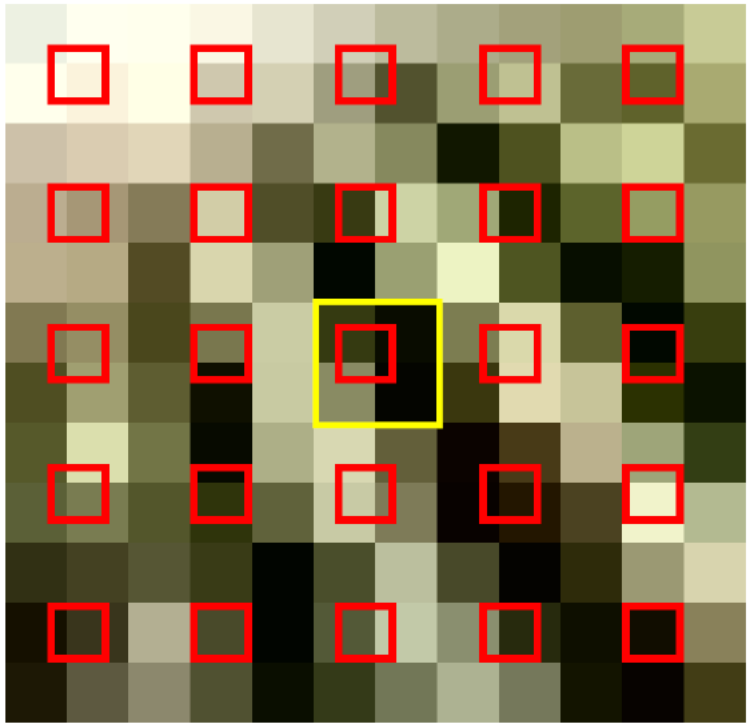


Resizing image

Resampling Through Bilinear Interpolation

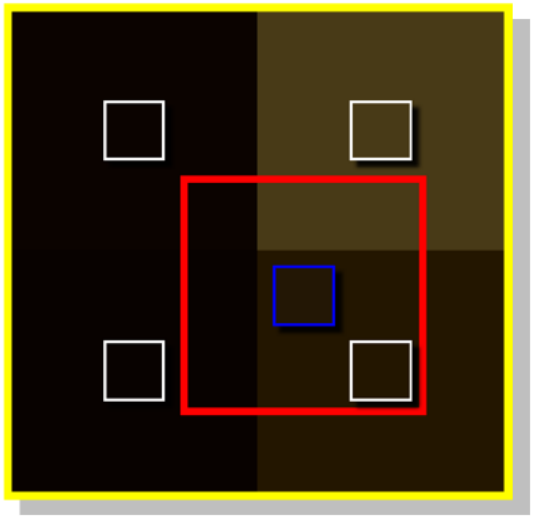


The blue square is the output location.

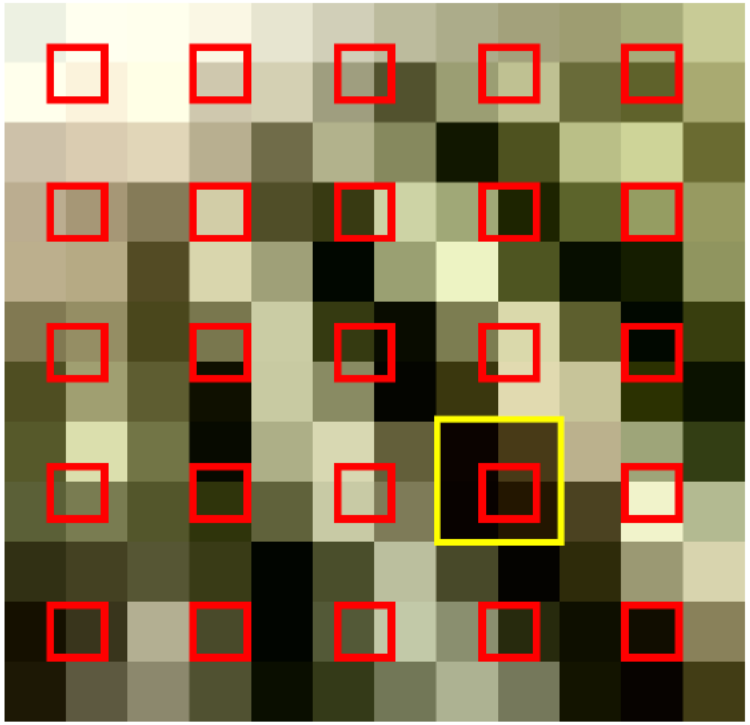


Resizing image

Resampling Through Bilinear Interpolation

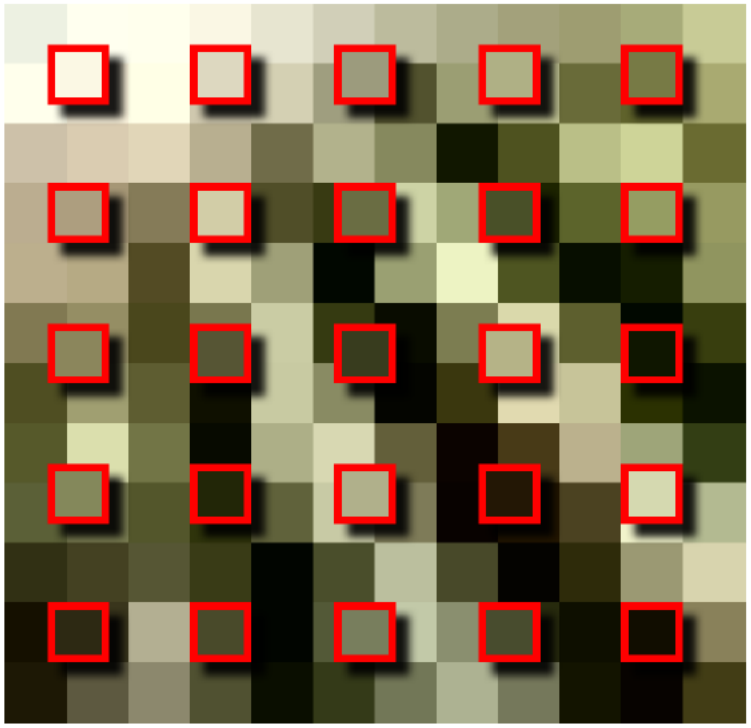


The blue square is the output location.

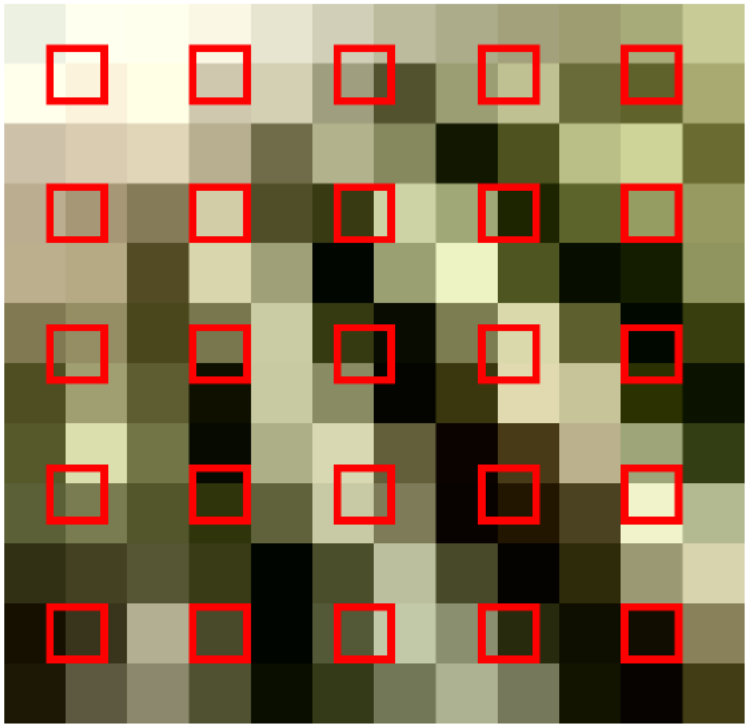


Resizing image

Resampling Through Bilinear Interpolation



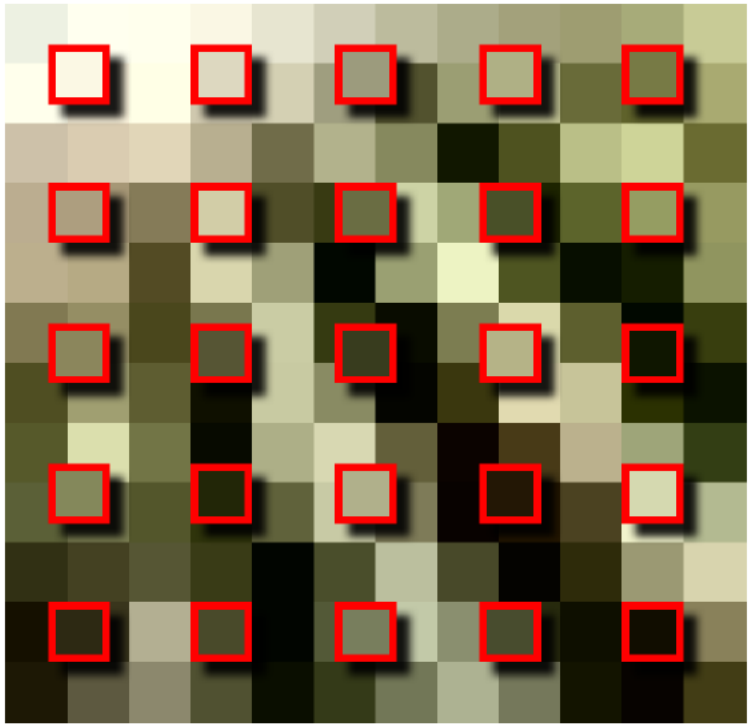
locs & colors of new pixels



locations of new pixels

Resizing image

Resampling Through Bilinear Interpolation

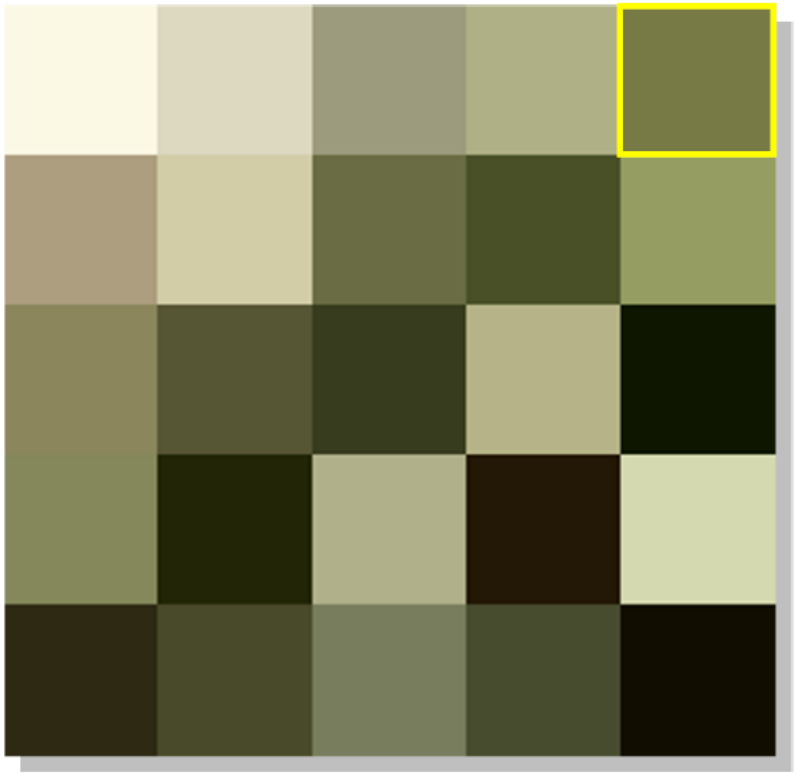


locs & colors of new pixels

locs & colors of new pixels

Resizing image

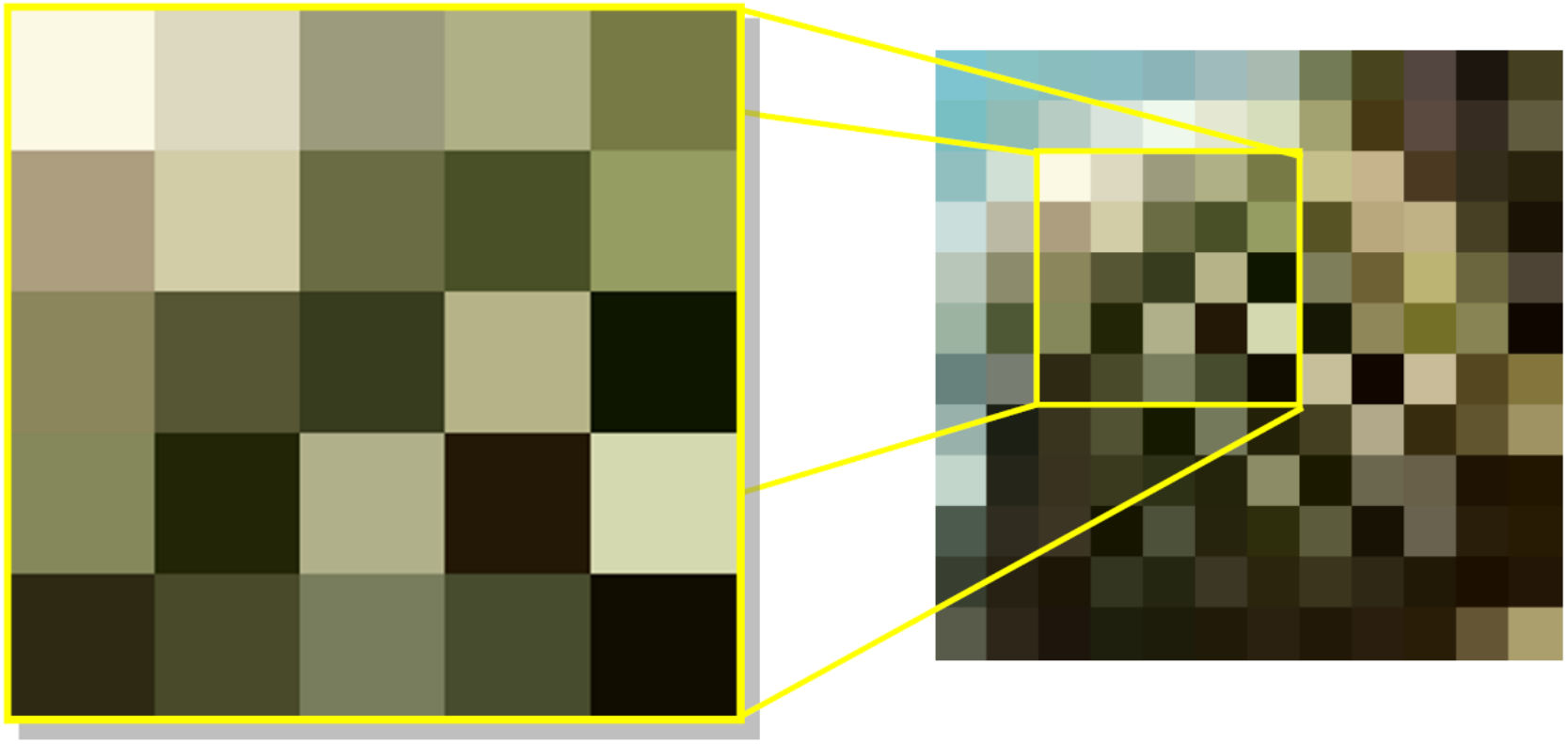
Resampling Through Bilinear Interpolation



New image from new pixels

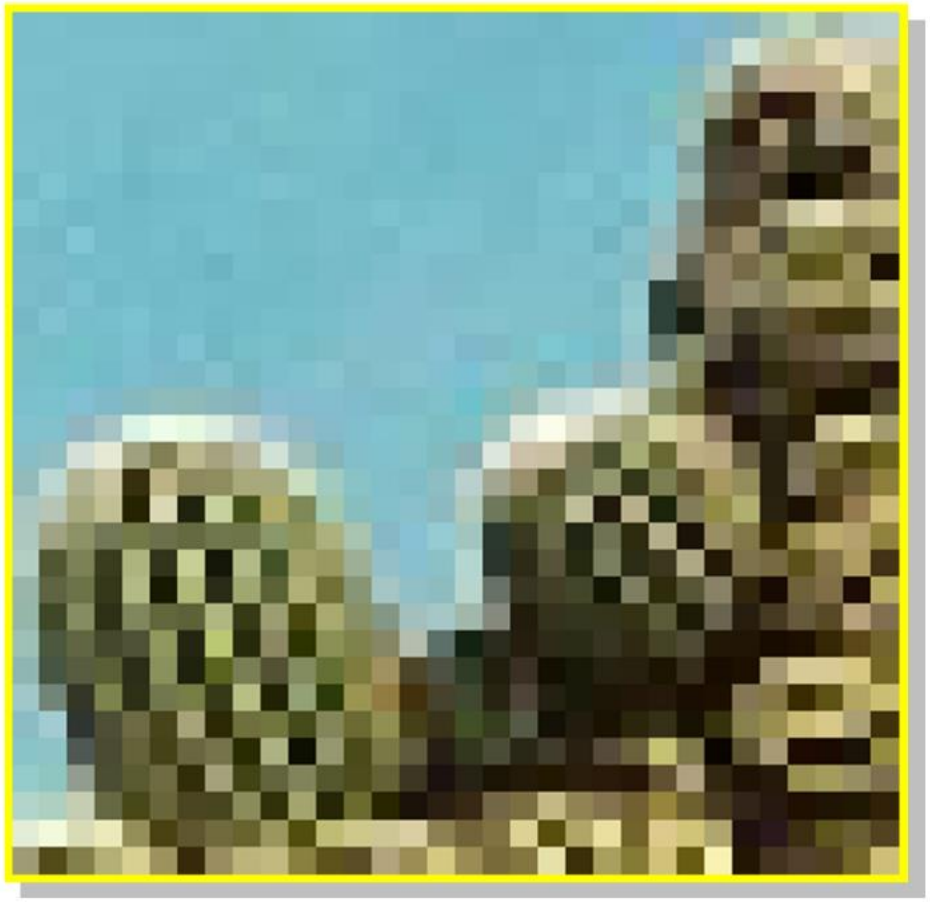
Resizing image

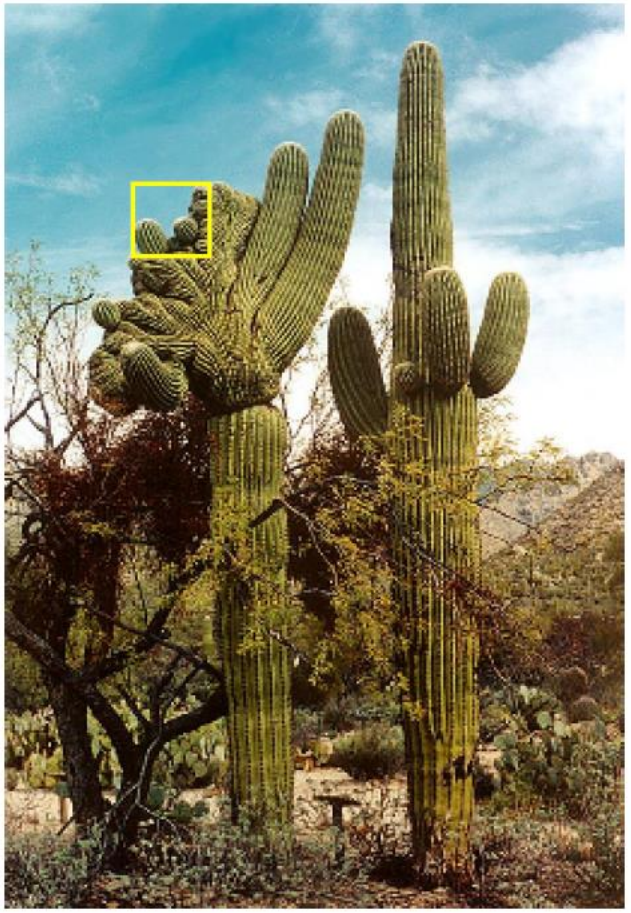
Resampling Through Bilinear Interpolation



Resizing image

Resampling Through Bilinear Interpolation





Resizing image

Resampling Through Bilinear Interpolation

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

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

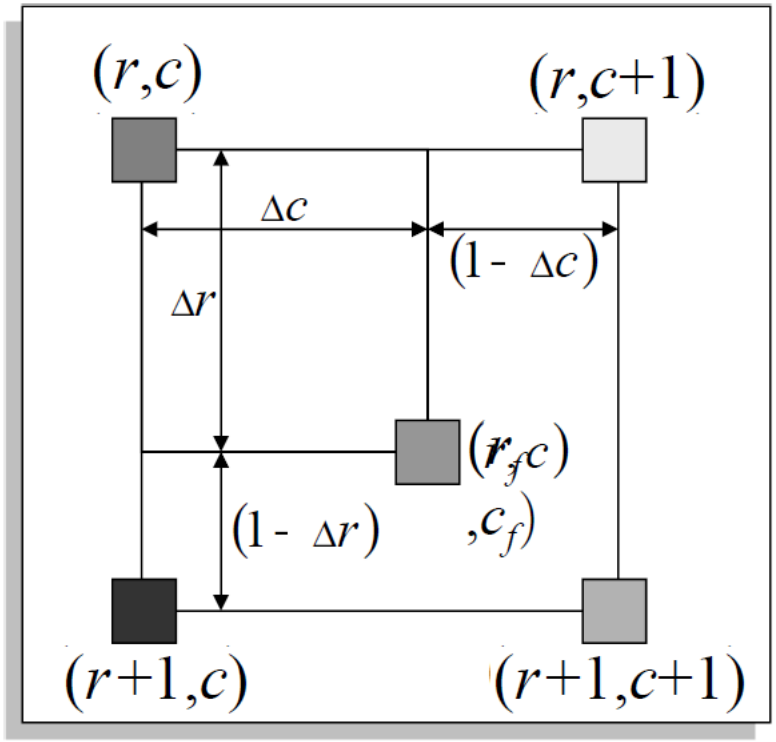
Let $r_f = r' \cdot s_R$ for $r' = 1, \ldots, R'$

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 $\Delta c = c_f - c$.

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



Resizing image

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).

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

(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

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

Resizing image

Size Reduction 3/7

original

nearest neighbor

bilinear

bicubic

Resizing image

Size Reduction 3/7

original

Resizing image

Size Reduction 3/7 (zoomed)

nearest neighbor

Resizing image

Size Reduction 3/7 (zoomed)

bilinear

Resizing image

Size Reduction 3/7 (zoomed)

bicubic

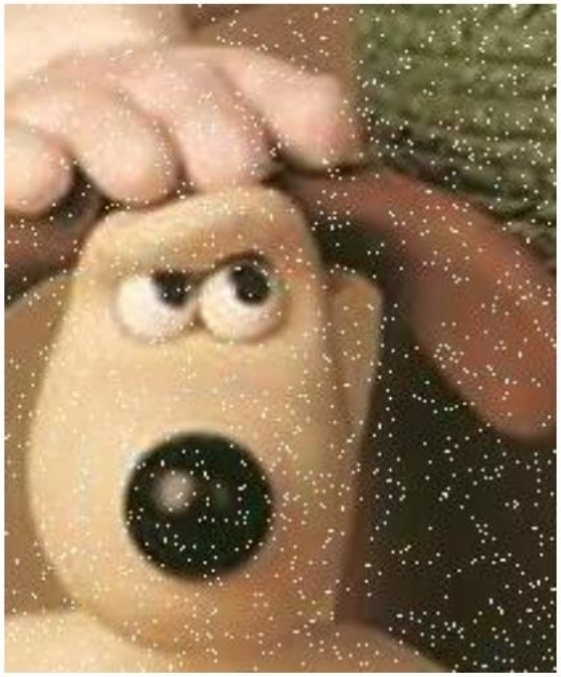
Noise Reduction

blurred image

color noise

color-only blur

Shot Noise or Salt & Pepper Noise



+ shot noise

s&p noise

- shot noise

Nonlinear Filters: the Median

original

s&p noise

median filter

Nonlinear Filters: Min and Maxmin

+ shot noise

min filter

maxmin filter

Nonlinear Filters: Max and Minmax

- shot noise

max filter

minmax