Image Processing

Spatial Resolution

Yeong Jun Koh

Department of Computer Science & Engineering

Chungnam National University

The imshow Function

- If x is a matrix of type uint8

```
imshow(x)
```

  - uint8 restricts values to be integers between 0 and 255

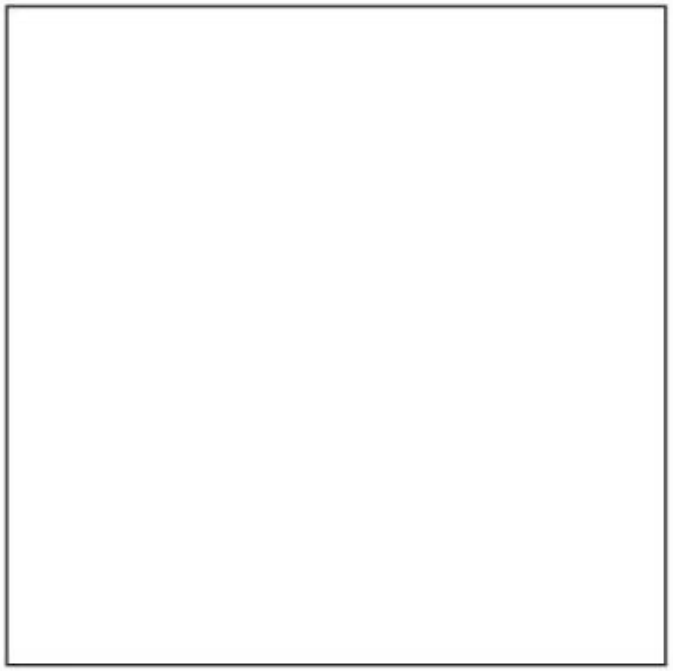
- Two choices with a matrix of type double
  - Convert to type uint8 and then display
  - Display the matrix directly

The imshow Function

```
>> c=imread('caribou.tif');
>> cd=double(c);
>> imshow(c),figure,imshow(cd)
```

Capt. Budd Christman/NOAA Corps.

(a)



(b)

FIGURE 3.1 *An attempt at data type conversion. (a) The original image. (b) After conversion to type* `double`.

The imshow Function

```
>> c=imread('caribou.tif');
>> cd=double(c);
>> imshow(c),figure,imshow(cd)
```

```
>> imshow(cd/255)
```

Capt. Budd Christman/NOAA

(a) (b)

FIGURE 3.1 *An attempt at data type conversion. (a) The original image. (b) After conversion to type* `double`.

The imshow Function

```
>> imshow(cd/512)
>> imshow(cd/128)
```

FIGURE 3.2 *Scaling by dividing an image matrix by a scalar. (a) The matrix* `cd` *divided by 512. (b) The matrix* `cd` *divided by 128.*

The imshow Function

- double - type change only
- im2double – type change, value scaling

```
>> cd=im2double(c);
```

- Convert back to an image of type uint8 in two ways

```
>> c2=uint8(255*cd);
>> c3=im2uint8(cd);
```

The imshow Function

- Binary image
  - logical flag

```
>> c1=c>120;
```

  - Check c1 with whos

```
Name        Size            Bytes  Class     Attributes
c1        256x256           65536  logical
```

The imshow Function

- Binary image

FIGURE 3.3 *Making the image binary. (a) The caribou image turned binary. (b) After conversion to type* `uint8`.

Bit Planes

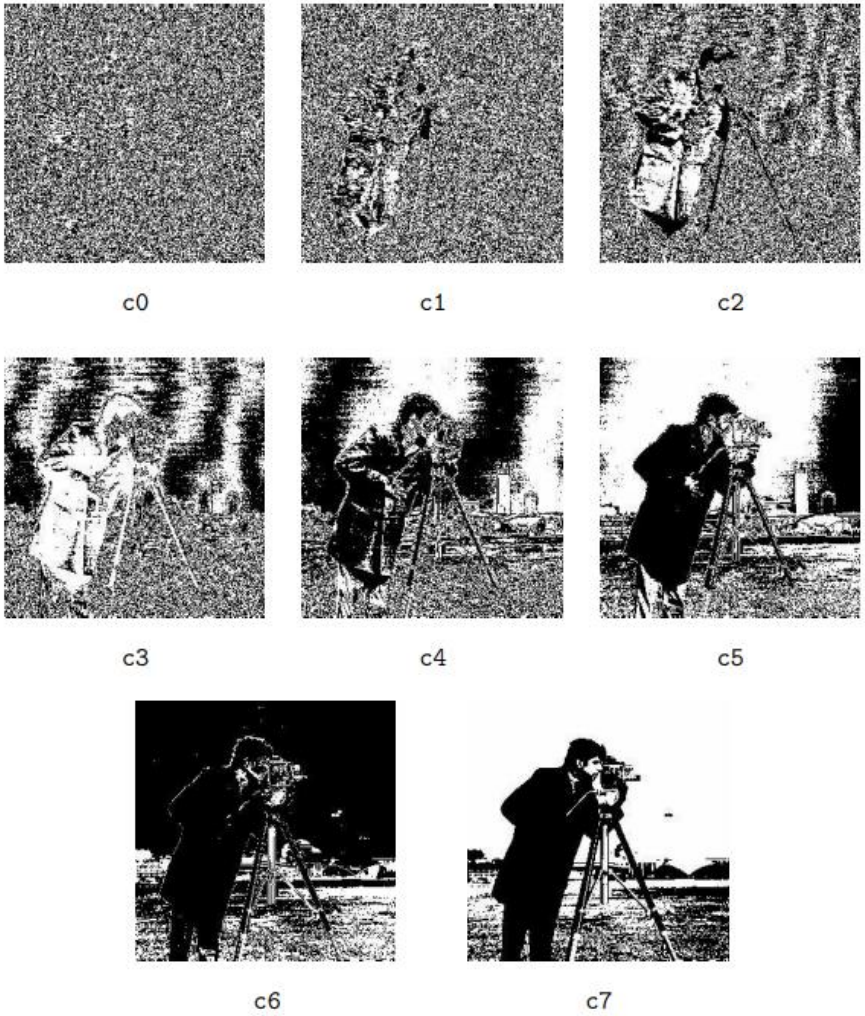
- Grayscale images can be transformed into a sequence of binary images by breaking them up into bit planes
- Pixel value 0~255 -> 8 bit 00000000(0)~11111111(255)
- The zeroth bit plane
  - The least significant bit plane
- The seventh bit plane
  - The most significant bit plane

Bit Planes

- Isolate the bit planes

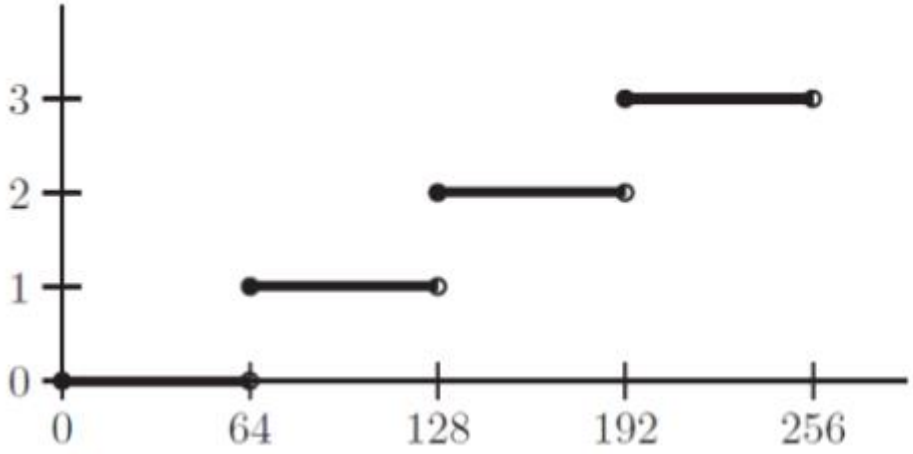
```
>> c=imread('cameraman.tif');
>> cd=double(c);
>> c0=mod(cd,2);
>> c1=mod(floor(cd/2),2);
>> c2=mod(floor(cd/4),2);
>> c3=mod(floor(cd/8),2);
>> c4=mod(floor(cd/16),2);
>> c5=mod(floor(cd/32),2);
>> c6=mod(floor(cd/64),2);
>> c7=mod(floor(cd/128),2);
```

Bit Planes



Quantization

- Uniform quantization



Original values	Output value
0–63	0
64–127	1
128–191	2
192–255	3

- Suppose x to be a matrix of type uint8

$$f = \mathrm{floor}(\mathrm{double}(x)/64)$$

$$q = \mathrm{floor}(f * 64)$$

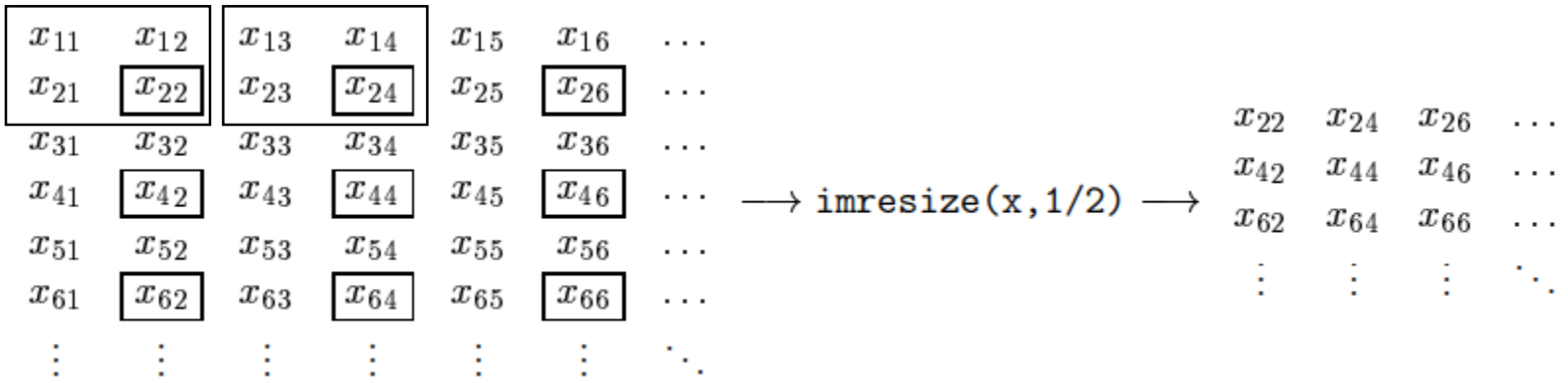
Quantization

Quantization

Spatial Resolution

- The greater the spatial resolution, the more pixels are used to display the image

- Adjust the spatial resolution with imresize function



$$(H, W) \rightarrow \left(\frac{H}{2}, \frac{W}{2}\right)$$

Spatial Resolution

```
x2=imresize(imresize(x,1/2),2);
```

→

$$\begin{array}{cccc}
\begin{bmatrix} x_{22} & x_{22} \\ x_{22} & x_{22} \end{bmatrix} & \begin{bmatrix} x_{24} & x_{24} \\ x_{24} & x_{24} \end{bmatrix} & \begin{bmatrix} x_{26} & x_{26} \\ x_{26} & x_{26} \end{bmatrix} & \cdots \\
\begin{bmatrix} x_{42} & x_{42} \\ x_{42} & x_{42} \end{bmatrix} & \begin{bmatrix} x_{44} & x_{44} \\ x_{44} & x_{44} \end{bmatrix} & \begin{bmatrix} x_{46} & x_{46} \\ x_{46} & x_{46} \end{bmatrix} & \cdots \\
\begin{bmatrix} x_{62} & x_{62} \\ x_{62} & x_{62} \end{bmatrix} & \begin{bmatrix} x_{64} & x_{64} \\ x_{64} & x_{64} \end{bmatrix} & \begin{bmatrix} x_{66} & x_{66} \\ x_{66} & x_{66} \end{bmatrix} & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{array}$$

Command	Effective resolution
`imresize(imresize(x,1/4),4);`	64×64
`imresize(imresize(x,1/8),8);`	32×32
`imresize(imresize(x,1/16),16);`	16×16
`imresize(imresize(x,1/32),32);`	8×8

Spatial Resolution

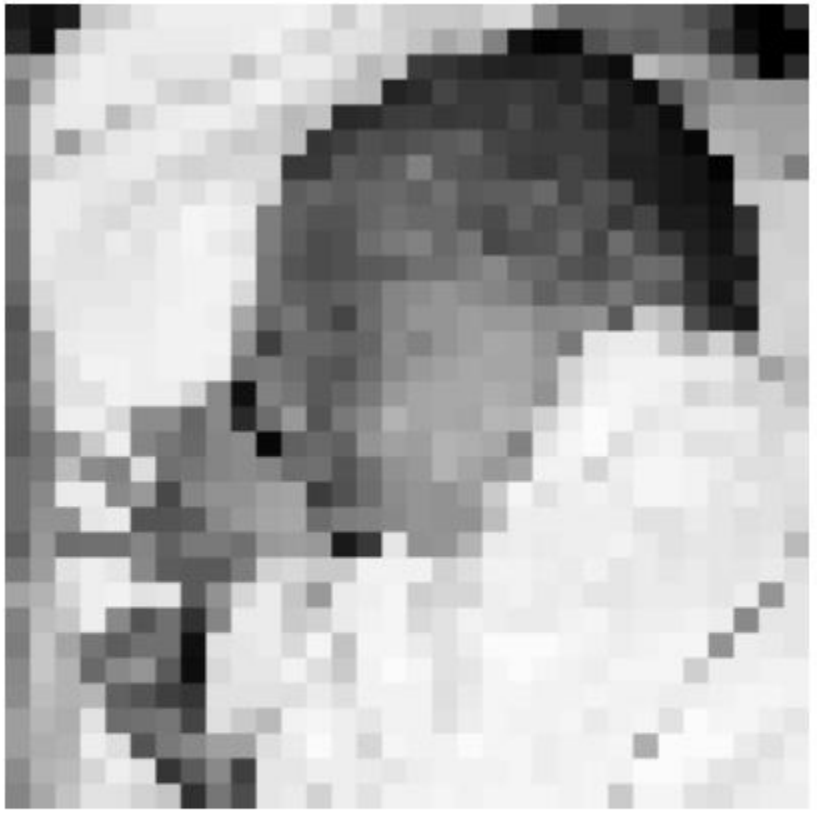
(a) The original image

(b) at 128×128 resolution

Figure 1.28: Reducing resolution of an image

Spatial Resolution

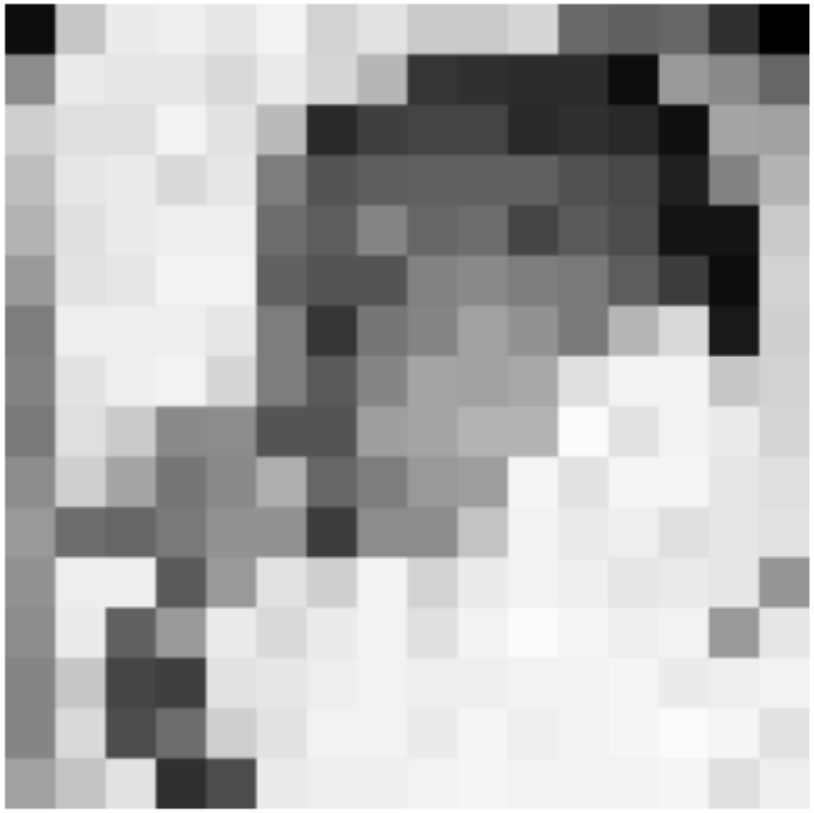
(a) At 64×64 resolution



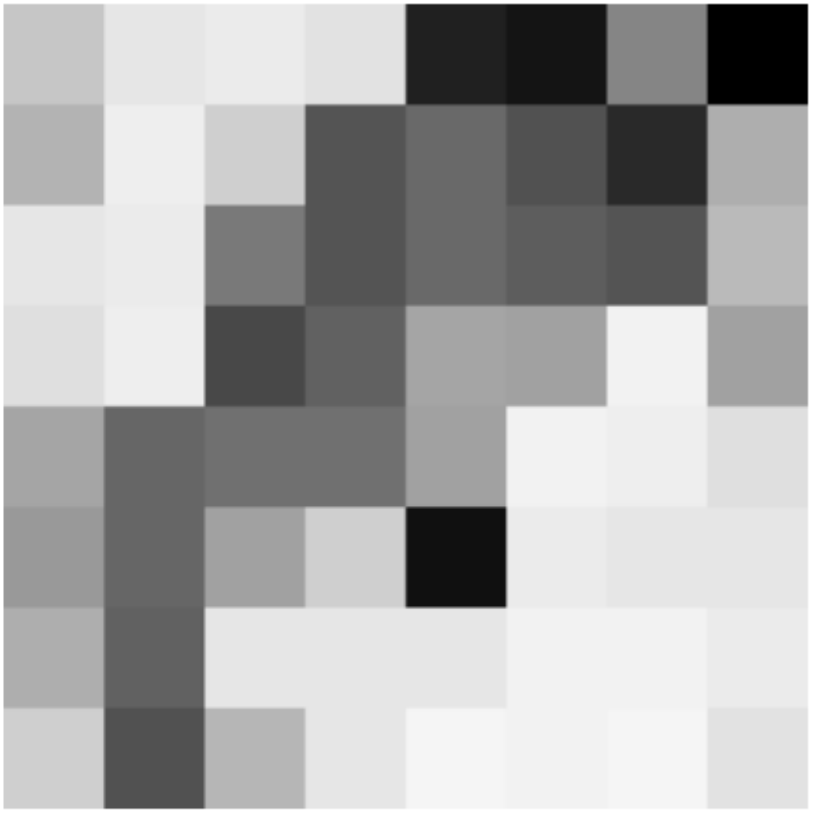
(b) At 32×32 resolution

Figure 1.29: Further reducing the resolution of an image

Spatial Resolution



(a) At 16×16 resolution



(b) at 8×8 resolution

Figure 1.30: Even more reducing the resolution of an image

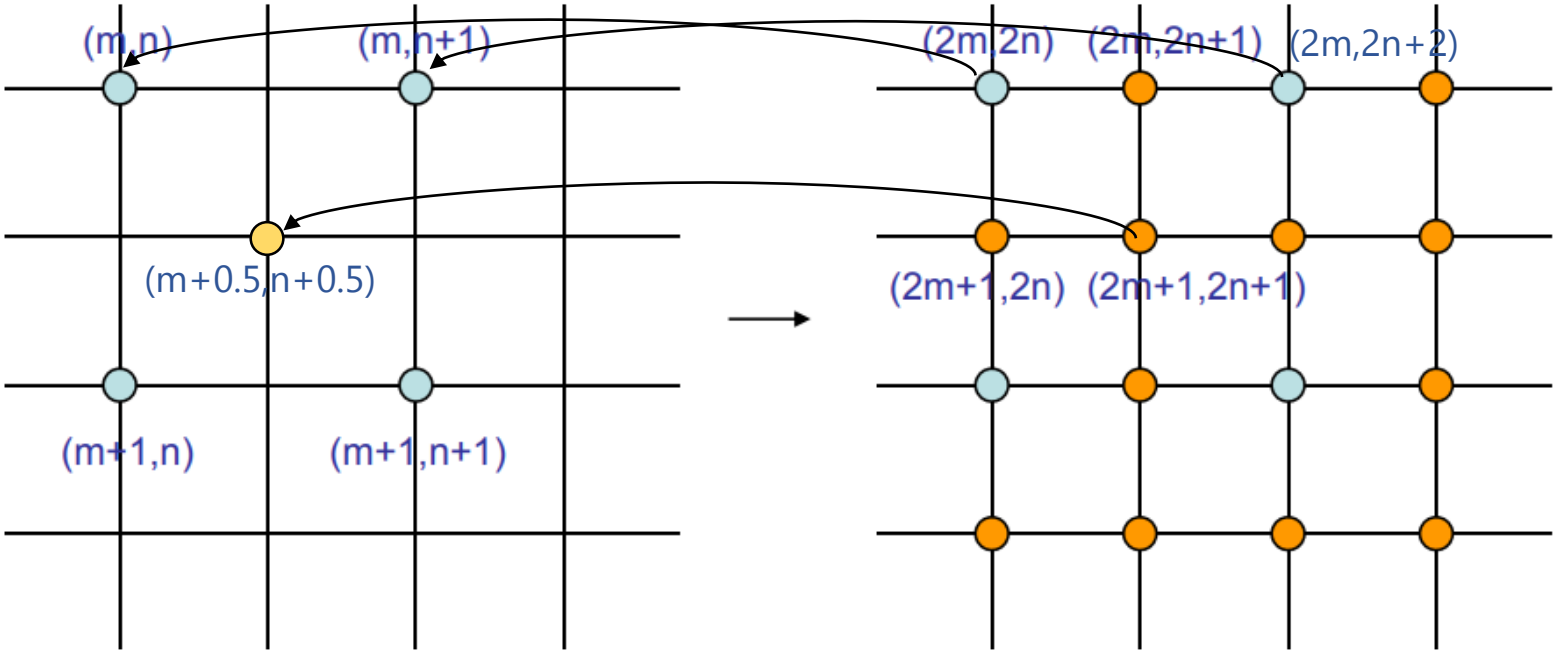
Spatial Resolution

- Types of image resizing algorithms
  - Nearest: nearest-neighbor interpolation
  - Bilinear: bilinear interpolation
  - Bicubic: cubic interpolation (default in MATLAB)
  - Lanczos: the best performance

Command	Effective resolution
`imresize(imresize(x,1/4),4);`	64×64
`imresize(imresize(x,1/8),8);`	32×32
`imresize(imresize(x,1/16),16);`	16×16
`imresize(imresize(x,1/32),32);`	8×8

```
imresize(imresize(x, factor, 'nearest'), factor, 'nearest');
```

Spatial Resolution



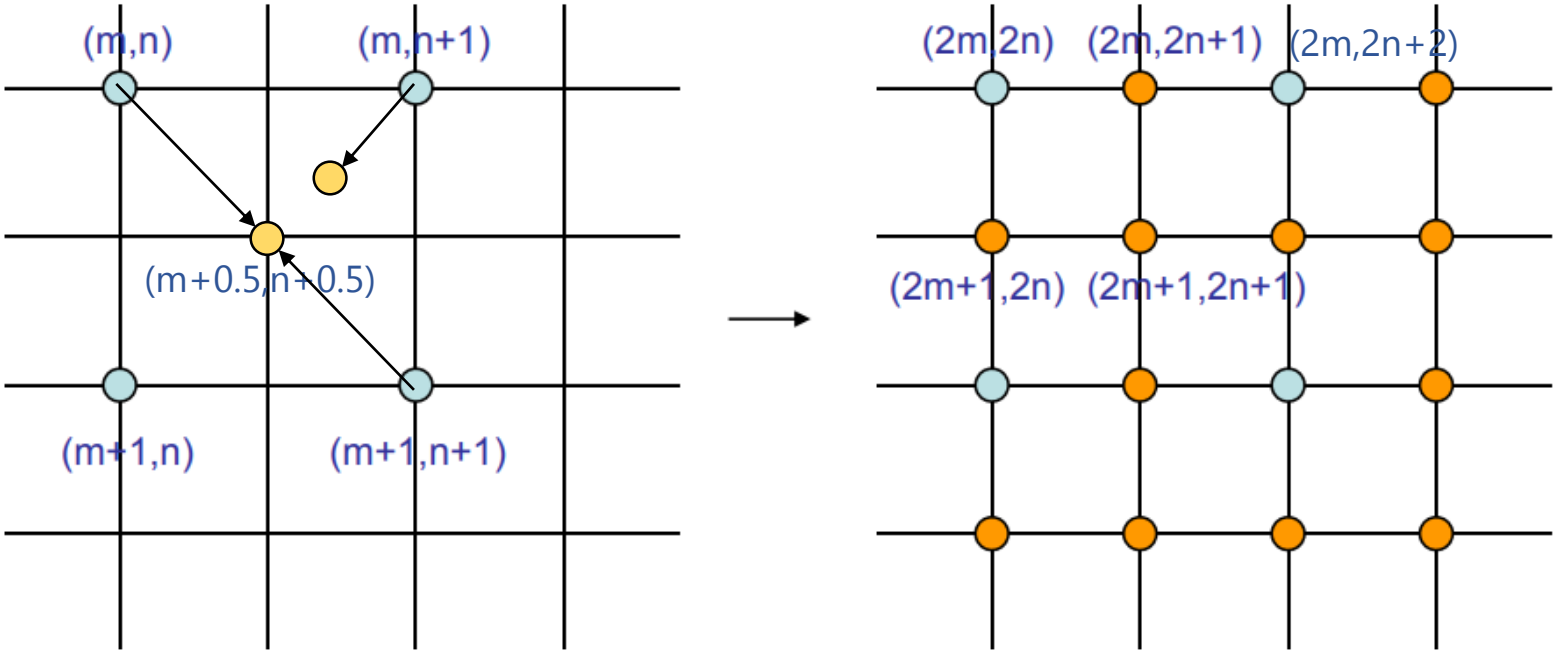
$(\frac{H}{2}, \frac{W}{2})$

(H, W)

Factor of 2 Up-Sampling

Spatial Resolution

- Nearest-neighbor interpolation

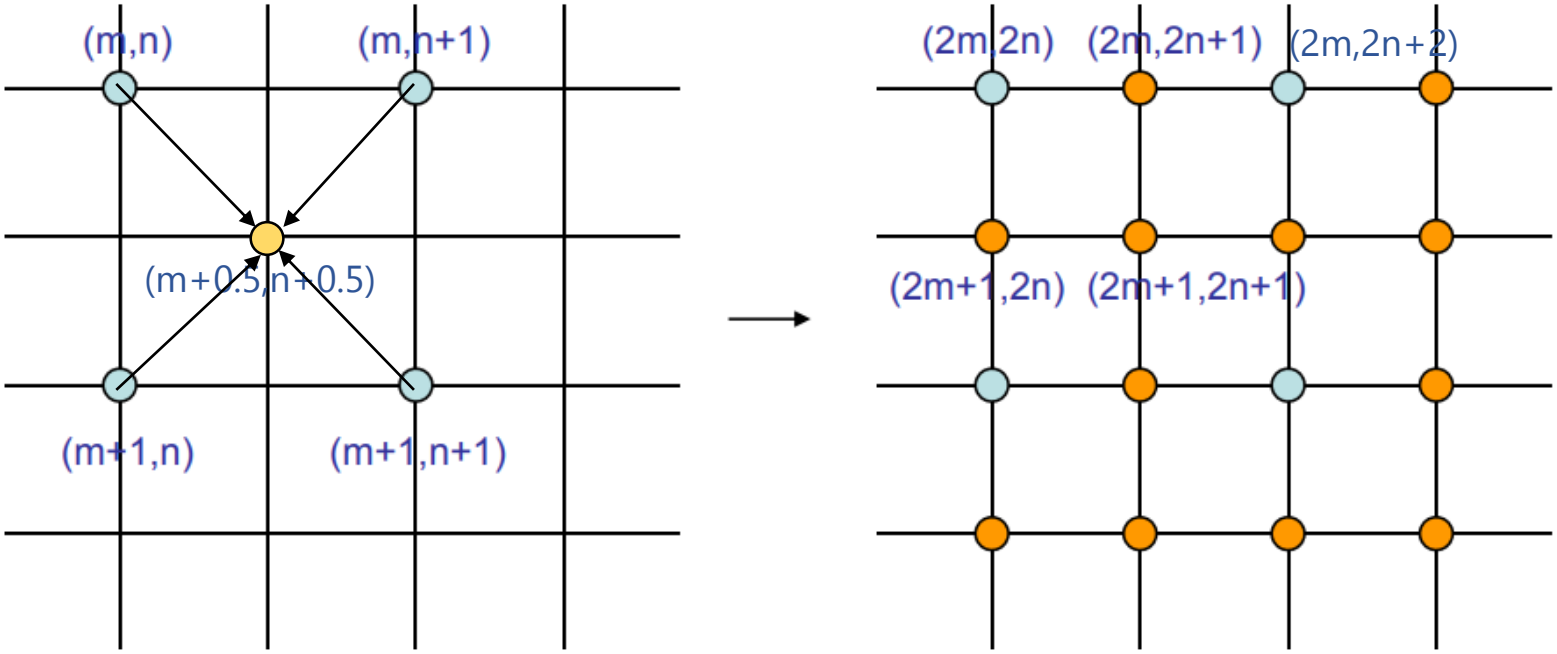


$(\frac{H}{2}, \frac{W}{2})$ (H, W)

Factor of 2 Up-Sampling

Spatial Resolution

- Bilinear interpolation

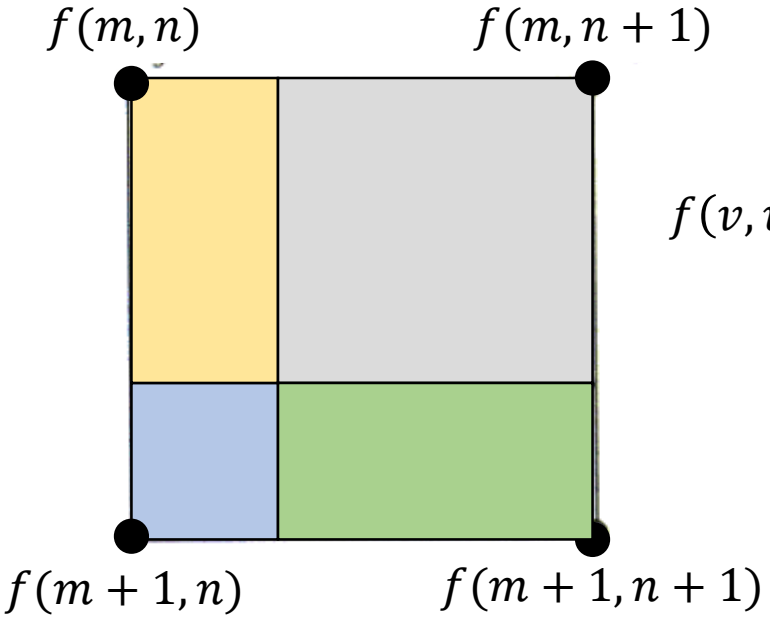


$(\frac{H}{2}, \frac{W}{2})$ (H, W)

Factor of 2 Up-Sampling

Spatial Resolution

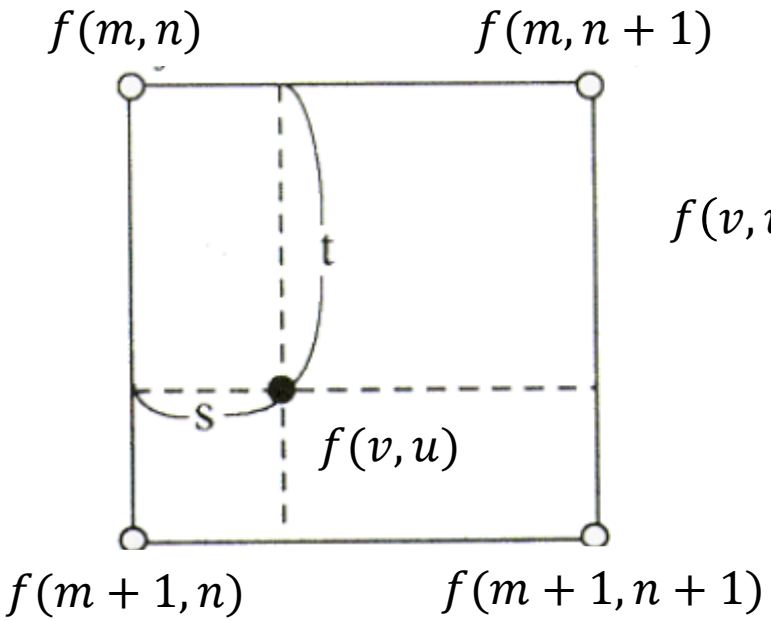
- Bilinear interpolation



$$f(v,u) = (1-s)(1-t)\cdot f(m,n) + s(1-t)\cdot f(m,n+1) + (1-s)t\cdot f(m+1,n) + st\cdot f(m+1,n+1)$$

Spatial Resolution

- Bilinear interpolation



$$f(v,u) = (1-s)(1-t)\cdot f(m,n) + s(1-t)\cdot f(m,n+1) + (1-s)t\cdot f(m+1,n) + st\cdot f(m+1,n+1)$$

Spatial Resolution

- Scale factor
  - $x = \text{factor} \times u$
  - $\text{y} = \text{factor} \times v$

$$f'(x,y) = f(\frac{x}{\text{factor}}, \frac{y}{\text{factor}})$$

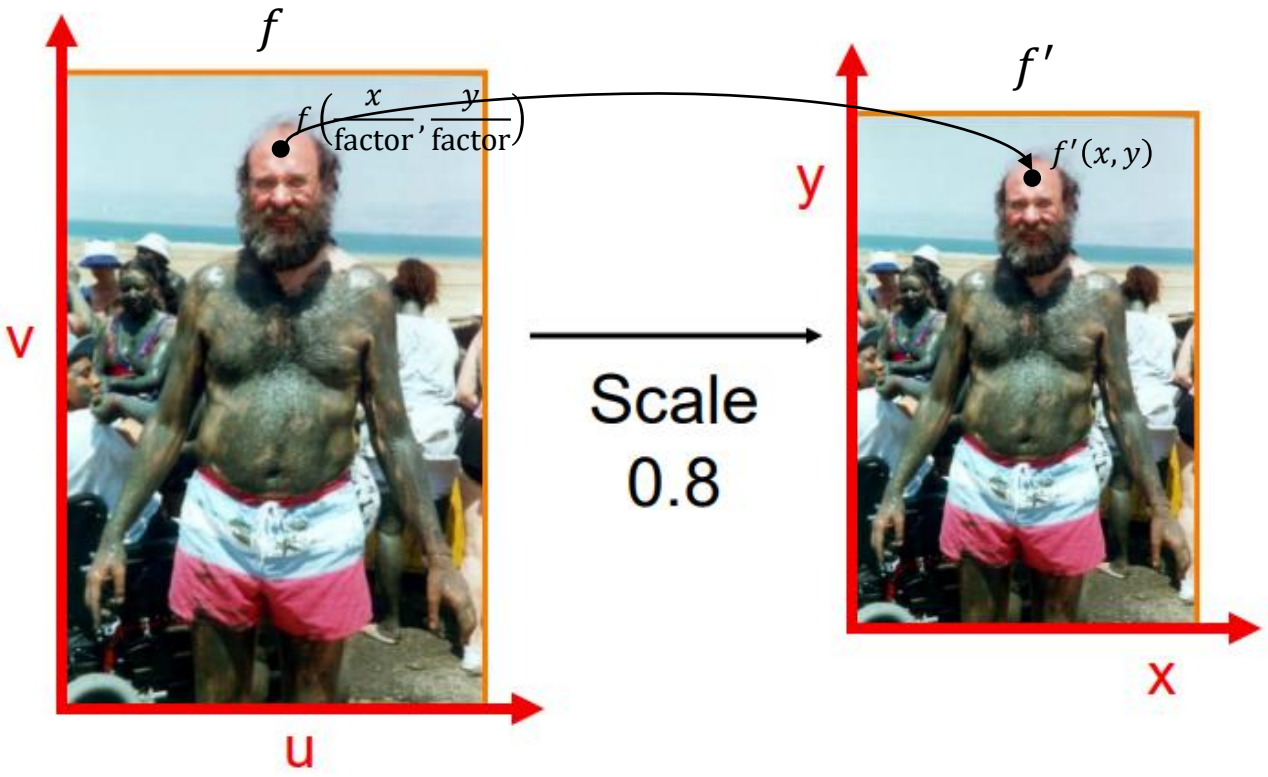
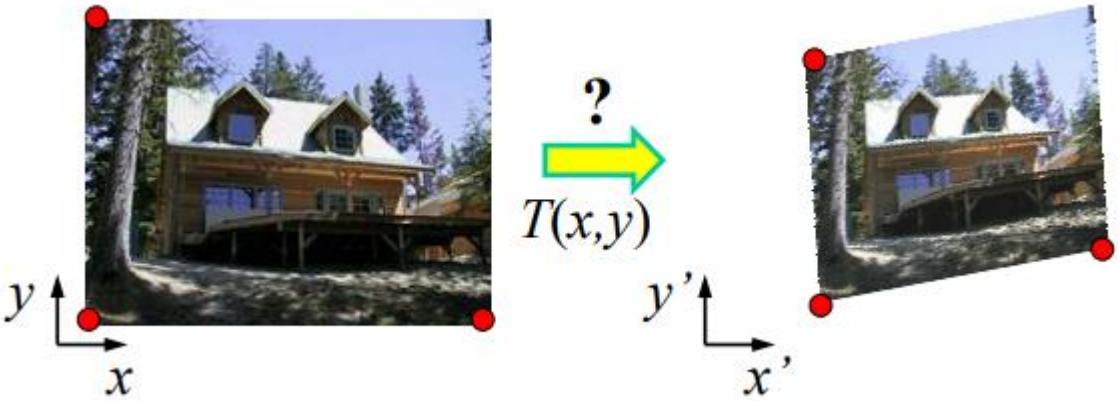


Image Warping

- Change appearance of image by geometric transform



https://en.wikipedia.org/wiki/Image_warping

- Image morphing http://youtube.com/watch?v=nUDIoN-_Hxs
- Video stabilization
- Semantic alignment

Image Warping

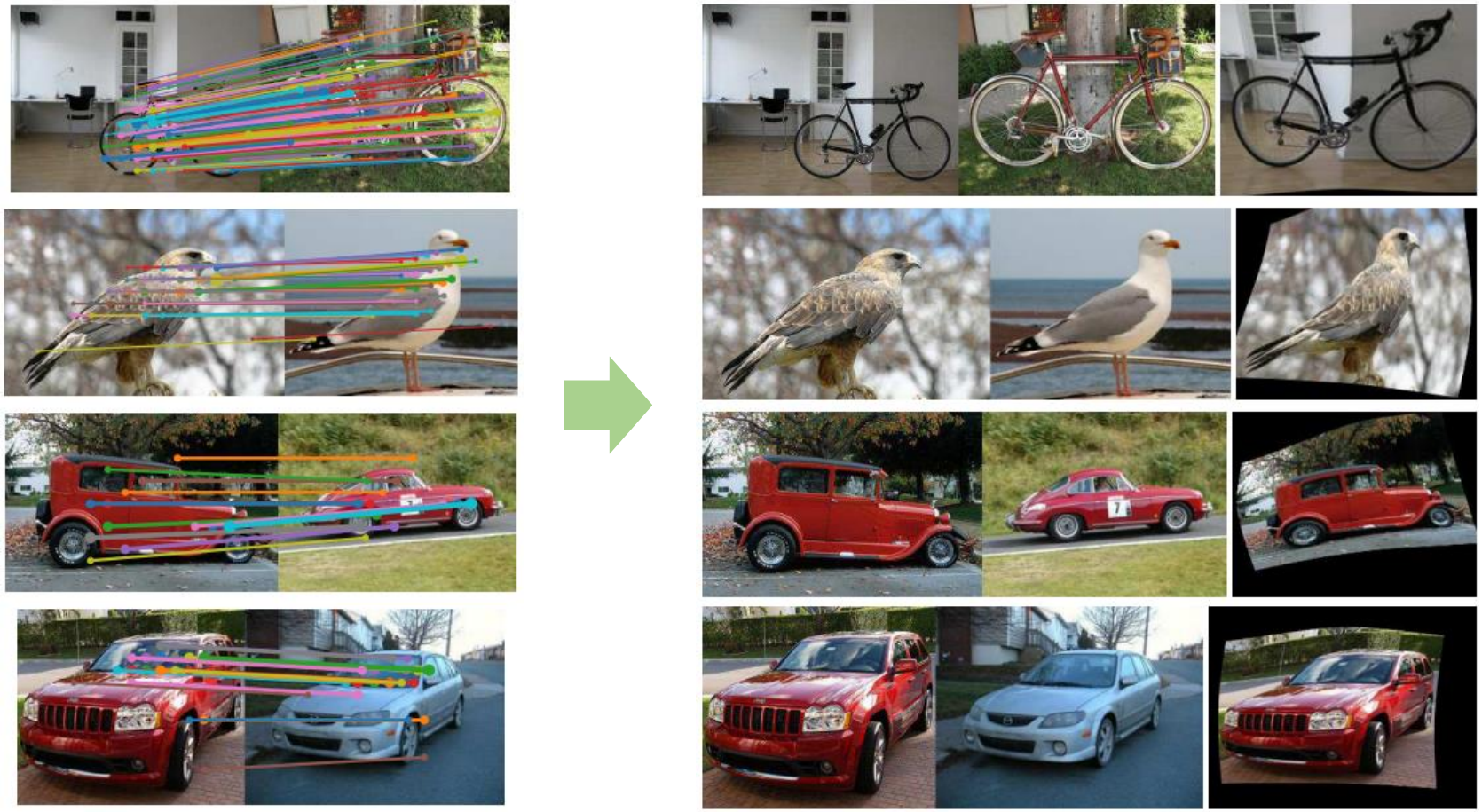
- Video stabilization

original

result

Image Warping

- Semantic alignment



I. Rocco et. al, "End-to-end weakly-supervised semantic alignment," CVPR 2018

Image Warping

- Parametric (global) warping

translation

rotation

aspect

affine

perspective

cylindrical