

Image Processing Basics

Week-2

COMP/ENGN4528/6528 Computer Vision

Timetable for Lab-assignment 1

- Lab assignment has been released
- Lab Q&A with tutors: Week 02, Week03, Week 04
- Deadline for Lab Assignment one:

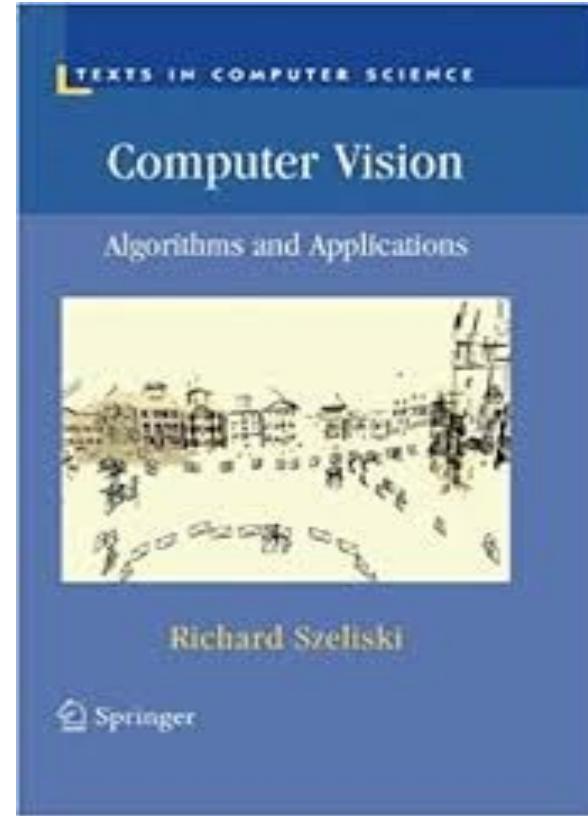
due 11:59pm, 15th March 2024 worth 15%

Four modules



Homework reading assignment (for week-1 and week-2)

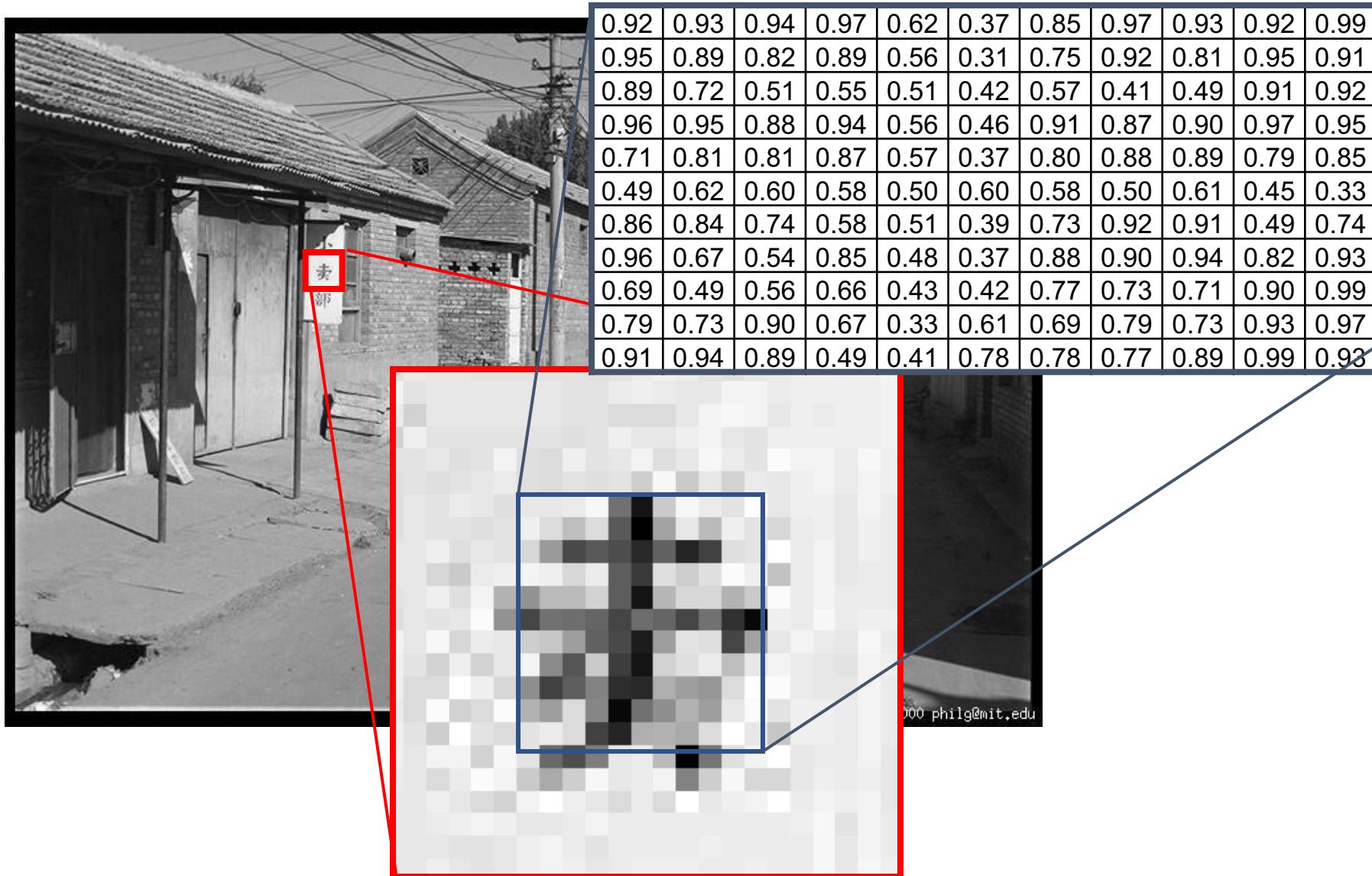
- Read textbook:
- Everything in Chapter1,
Chapter-2 and Chapter-3.



Types of Image Processing Operations

- Point-wise operations (Chapter 3.1)
 - Histogram Operation
- Neighbourhood operations
 - Spatial domain (image convolution) (Chapter 3.2)
- Geometric operations (Chapter 3.6)
 - Rotation, translation, affine...

Recall: digital image representation



Recall: Fundamentals of Digital Images


$$I = f(x, y) : \mathbf{R}^2 \rightarrow \mathbf{R}$$

(Gray-scale/intensity image)

$$I = f(x, y) : \mathbf{R}^2 \rightarrow \mathbf{R}^3$$

(Color image)

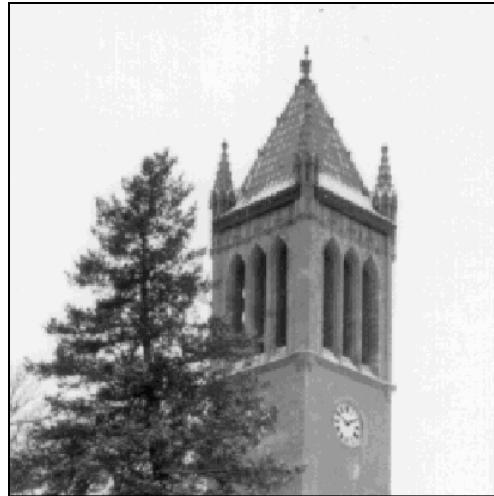
Recall: Effect of Spatial Resolution



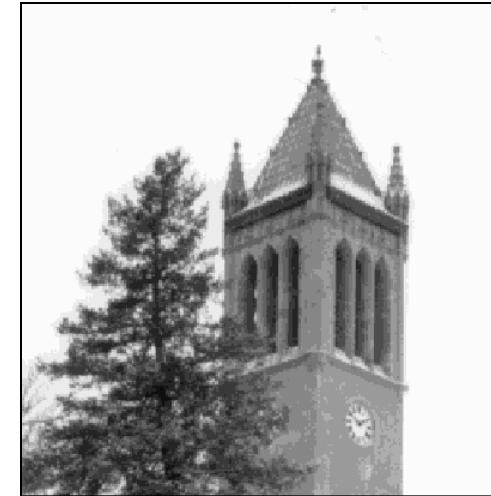
FIGURE 2.19 A 1024×1024 , 8-bit image subsampled down to size 32×32 pixels. The number of allowable gray levels was kept at 256.

(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

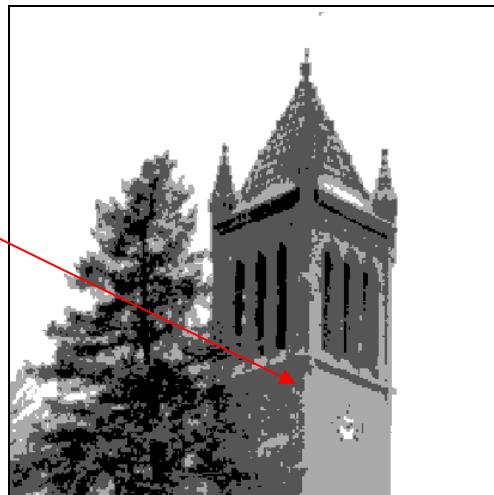
Recall: Effect of Quantization Levels (cont.)



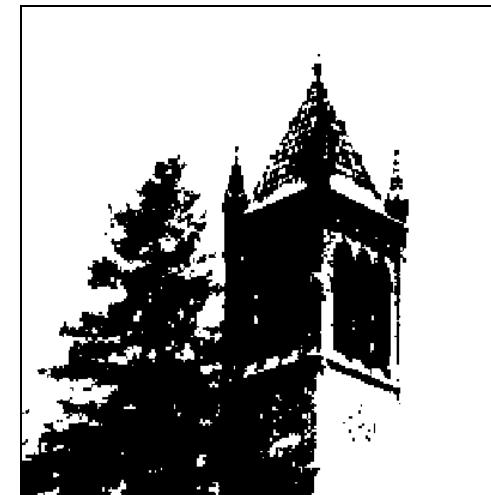
16 levels



8 levels



4 levels



2 levels

In this image,
it is easy to see
false contour.

Classifications of image processing operations

Three types of image processing operations:

1. Point operations (e.g. histogram operation)
2. Neighborhood operations (e.g. image convolution, filtering)
3. Geometric operations (e.g., rotation, translation, affine...)

Point Operations

- Image transformation
- The output of such transformation only depends on the corresponding input **pixel (intensity, color) value**.

Fundamentals of Digital Images


$$I = f(x, y) : \mathbf{R}^2 \rightarrow \mathbf{R}$$
 (Gray scale image)
$$I = f(x, y) : \mathbf{R}^2 \rightarrow \mathbf{R}^3$$
 (Color image)

Point Operations

- Operation on Pixel's value.
- Context free (memory-less).
- Operation can be performed on the *image histogram*.
- Example:

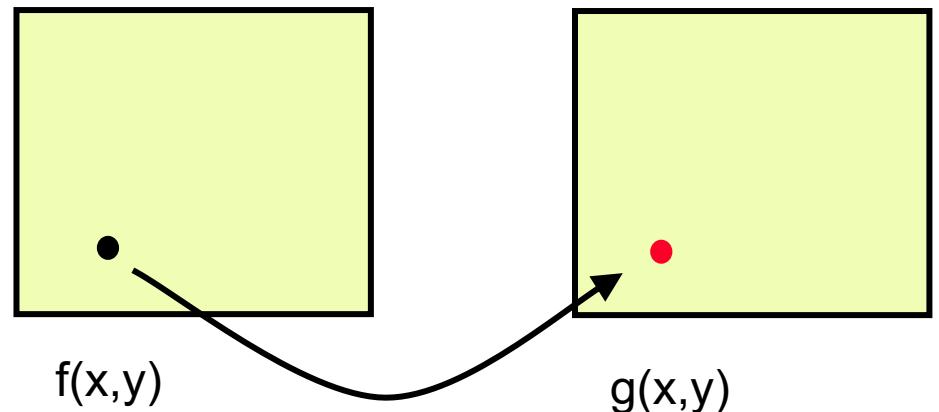
$$g(x, y) = \alpha \cdot f(x, y) + \beta$$

α is a constant, controlling the contrast.

β is a constant, controlling the brightness.



$$g(x, y) = M(f(x, y))$$



Point Operations

- Operation on Pixel's value.
- Context free (memory-less).
- Operation can be performed on the *image histogram*.
- Example:

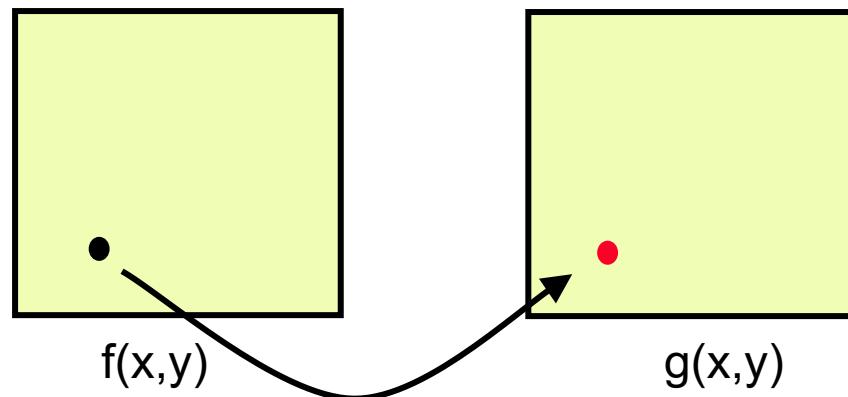


$$g(x, y) = M(f(x, y))$$

$$g(x, y) = \alpha(x, y)f(x, y) + \beta(x, y)$$

α is a spatially varying.

β is a spatially varying.



Neighborhood Operations

- Operation depends on Pixel's value and its spatial neighbors (coordinates-dependent).

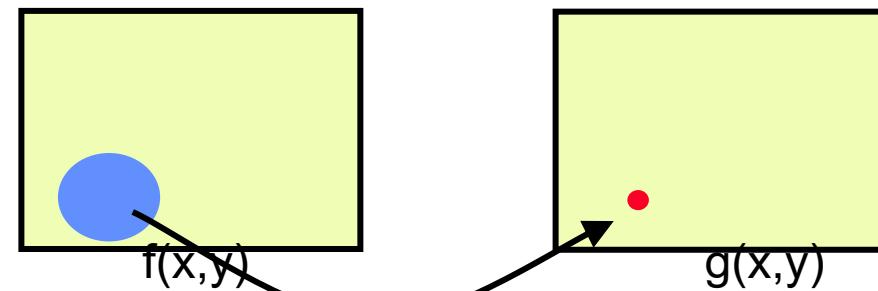


- Context dependent.

$$g(x, y) = M(\{f(i, j) | (i, j) \in N(x, y)\})$$

- Example:

$$g(x, y) = \sum_{i, j \in N(x, y)} f(i, j) / n$$



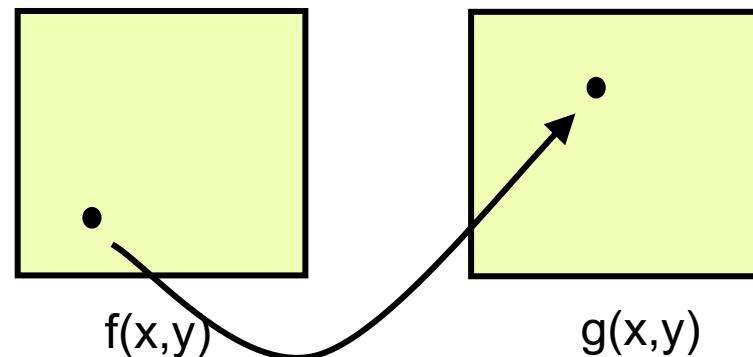
Geometric Operations

- Operation applied on pixel's coordinates.
- Independent of pixels gray scale/intensity value.
- Context free.
- Example: translation

$$g(x, y) = f(x + a, y + b)$$



$$g(x, y) = f(G(x, y))$$



Point operations:

(Histogram operation)

Point Operations

- A point operation can be defined as a mapping function: $v_{new} = M(v_{old})$
where v stands for pixel gray scale/intensity values.
- $M(v)$ takes any (intensity) value v in the source image into v_{new} in the target image.
- Simplest case - Linear Mapping:

$$M(v) = \alpha v + \beta$$

- Often implemented on the **image histogram**.

Histogram

- The histogram of a digital image with gray values r_0, r_1, \dots, r_{L-1} is the discrete function

$$p(r_k) = \frac{n_k}{n}$$

n_k : Number of pixels with gray value r_k

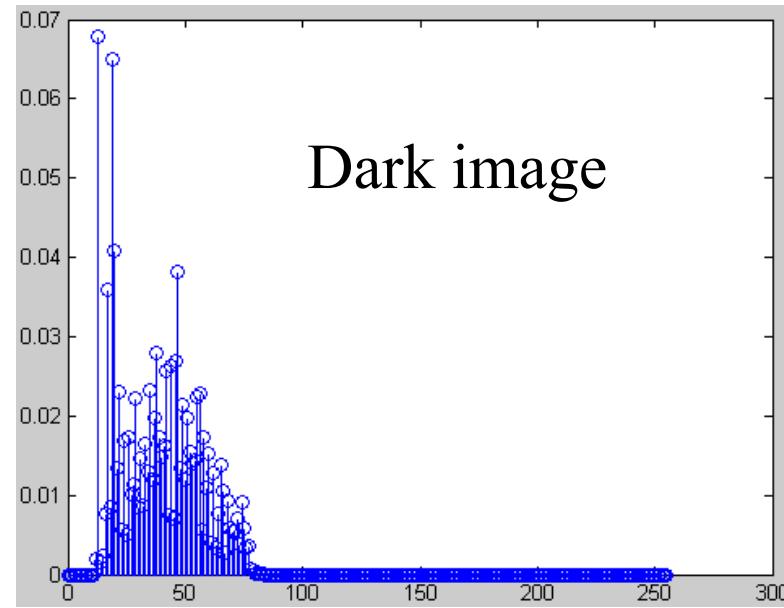
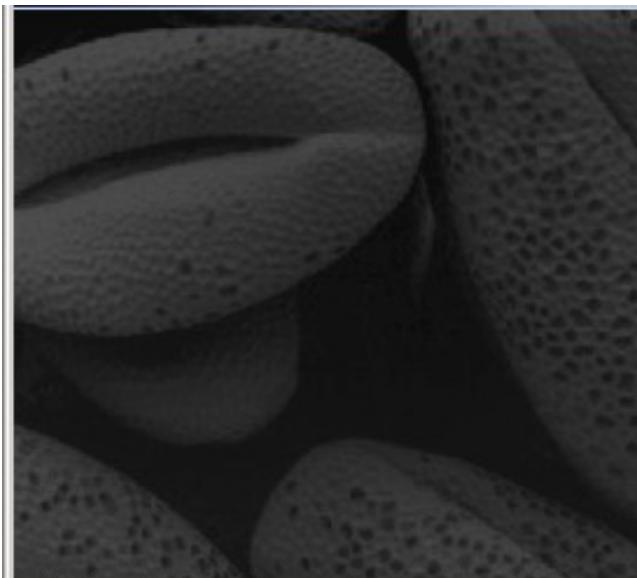
n : total Number of pixels in the image

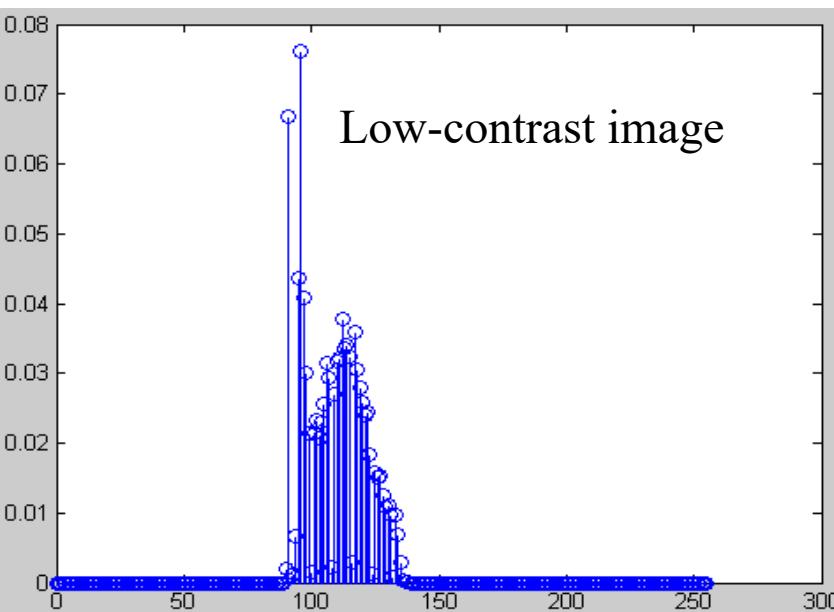
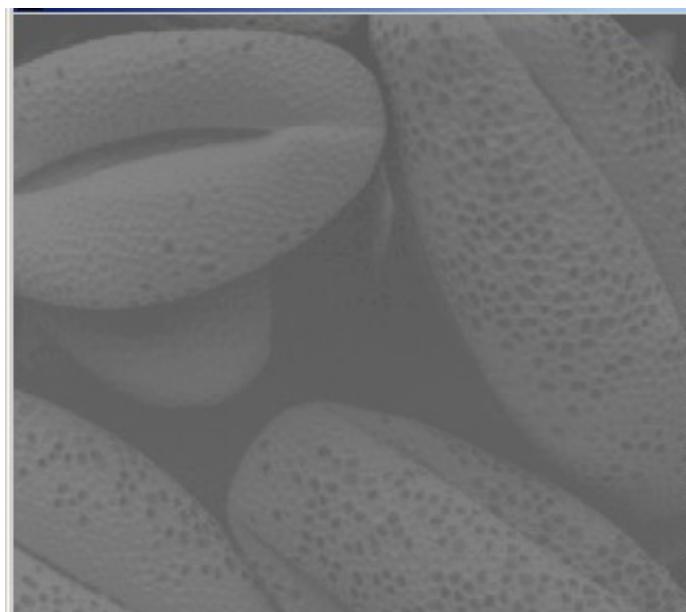
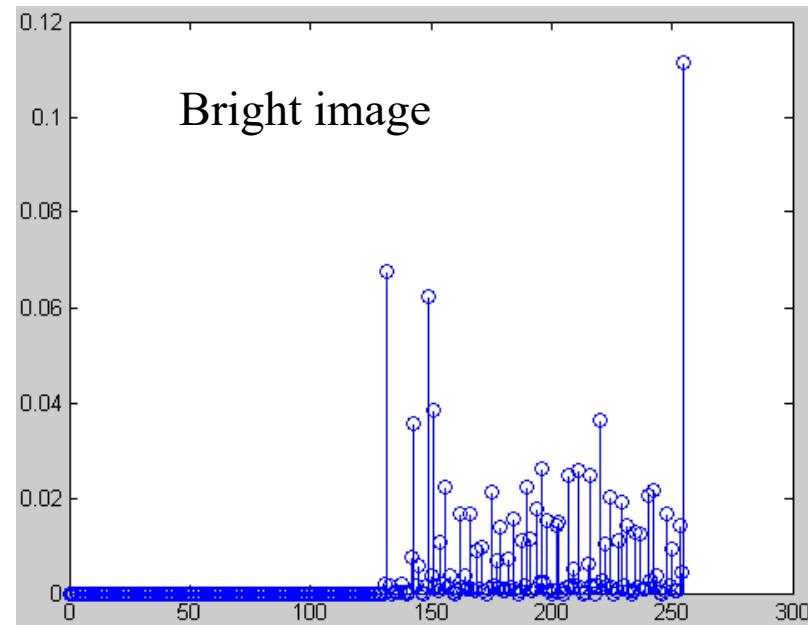
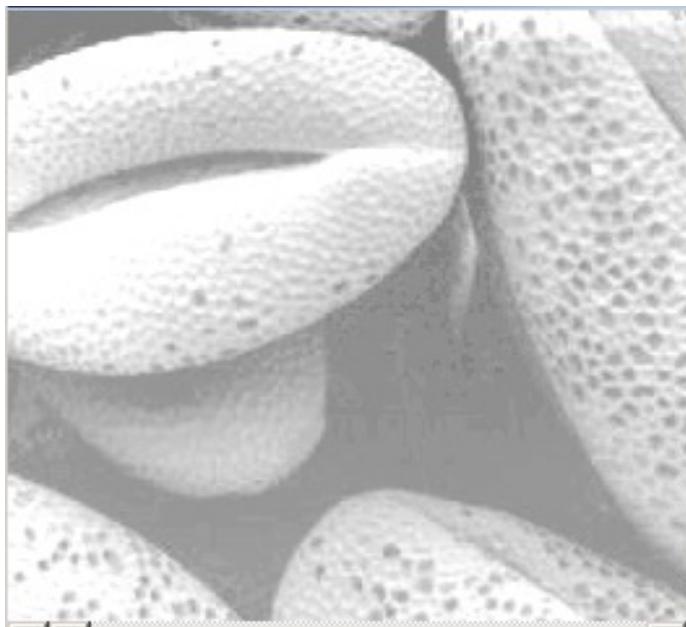
- The function $p(r_k)$ represents the fraction of the total number of pixels with gray value r_k .

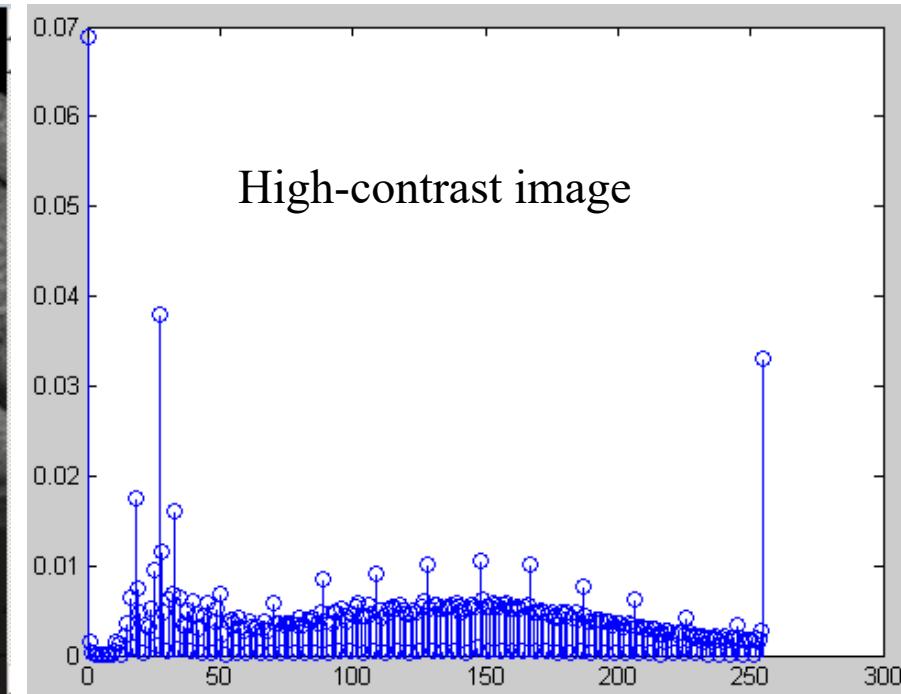
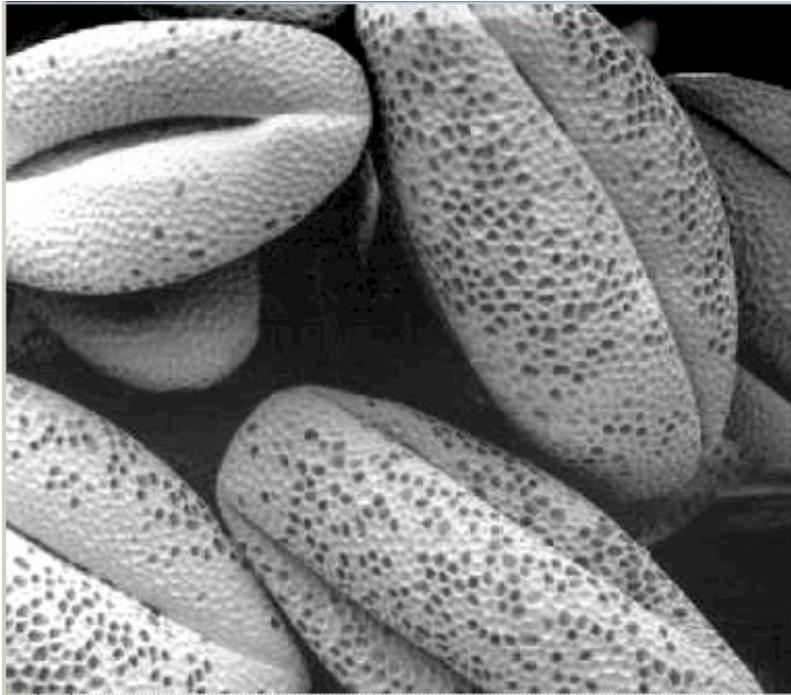
- Histogram provides a global description of the appearance of an image.
- If we consider the gray values in the image as realizations of a random variable R , with some probability density, histogram provides an **approximation** to this probability density.
- In other words, $\Pr(R = r_k) \approx p(r_k)$

Some Typical Histograms

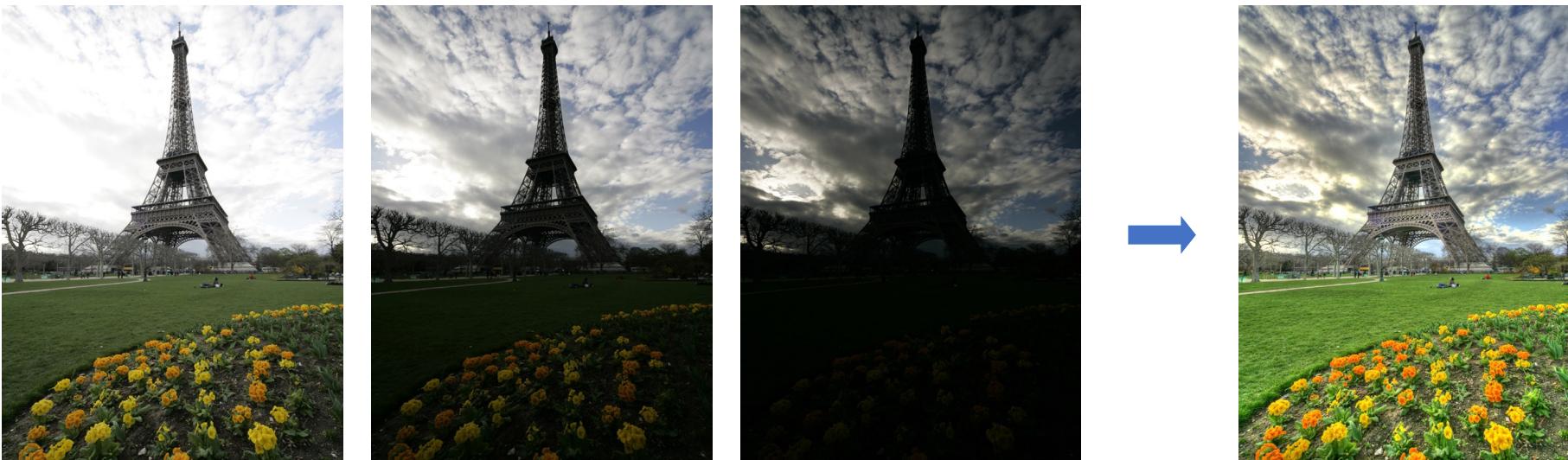
- The shape of a histogram provides useful information for image global appearance.







Application: image enhancement



Histogram modification

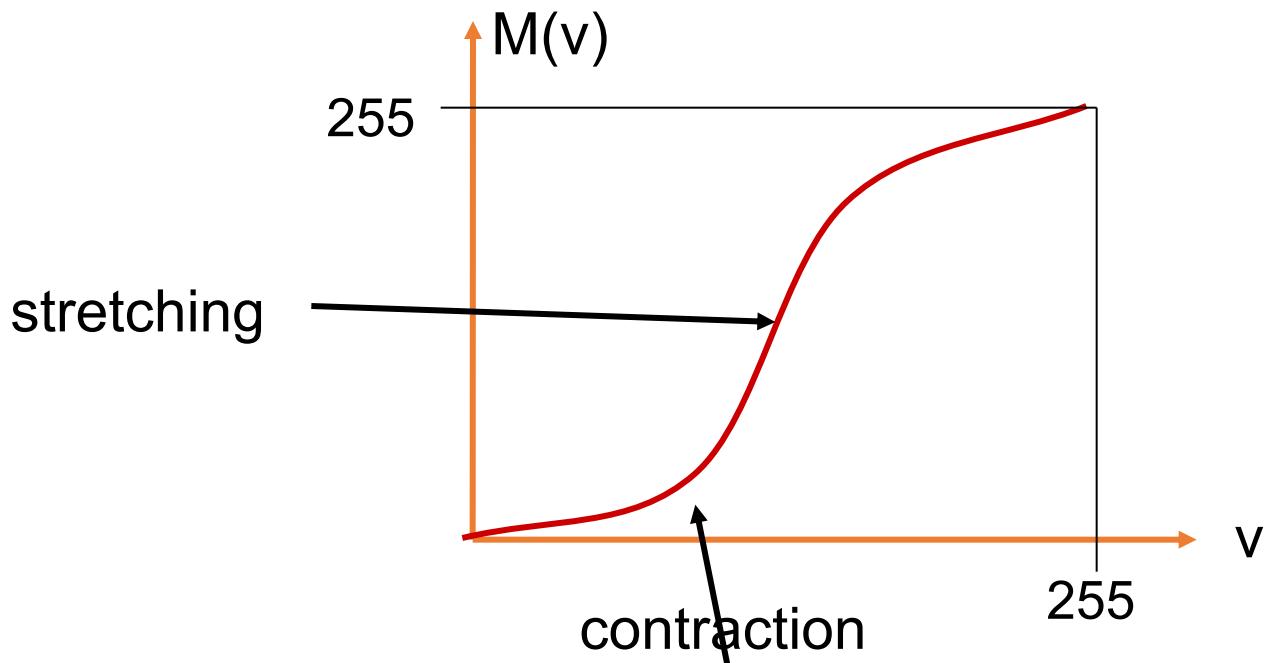
- Given a point operation:

$$v_b = M(v_a)$$

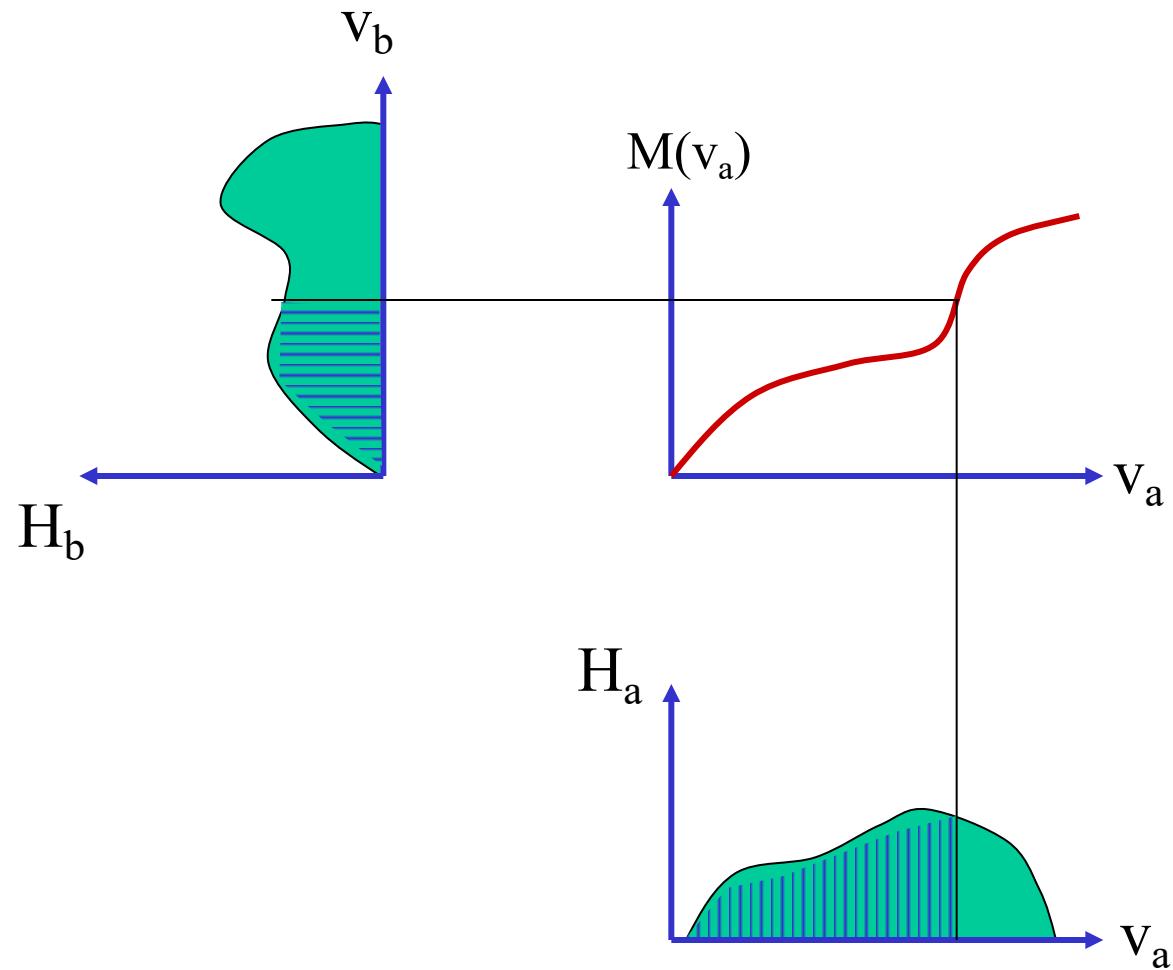
- $M(v_a)$ takes any value v_a in image A and maps it to v_b in image B.
- Requirement:** the mapping M is a non-descending function (M^{-1} exists).
- In this case, the area under H_a between 0 and v_a is equal to the area under H_b between 0 and v_b . Namely, the cumulative density function (CDF) for v_a and v_b are the same.

Monotonicity Requirement

- If it is required to map the full gray-level range (256 levels) to its full range while keeping the gray-level order, a **non-decreasing monotonic mapping function** is required:

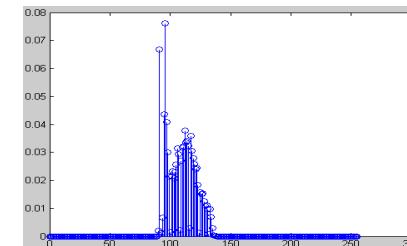
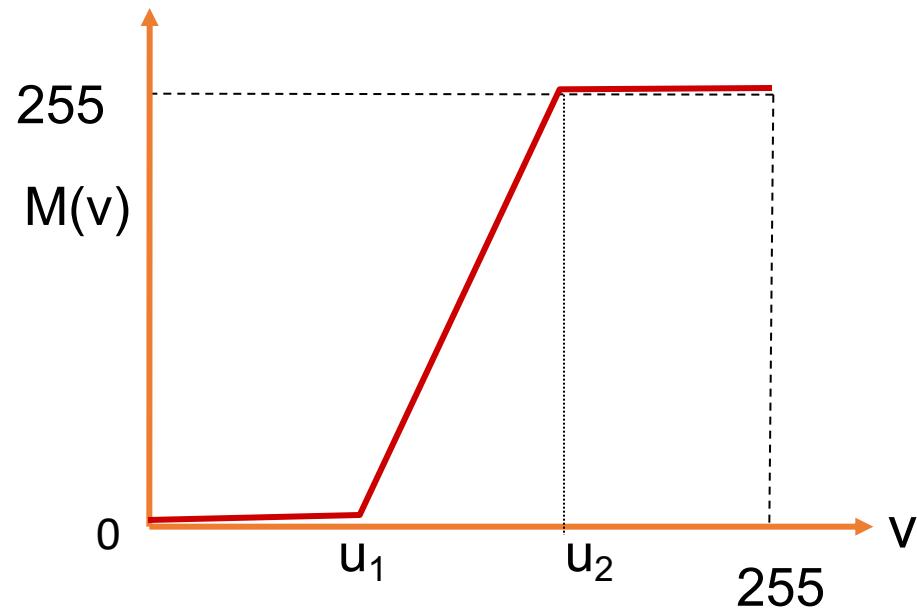


- The area under H_a between 0 and v_a is equal to the area under H_b between 0 and $v_b = M(v_a)$



Contrast Enhancement

- If most of the gray-levels in the image are in $[u_1 u_2]$, the following mapping increases the image contrast.
- The values u_1 and u_2 can be found by using the image's accumulated histogram.



Power-law transformations

$$s = cr^\gamma$$

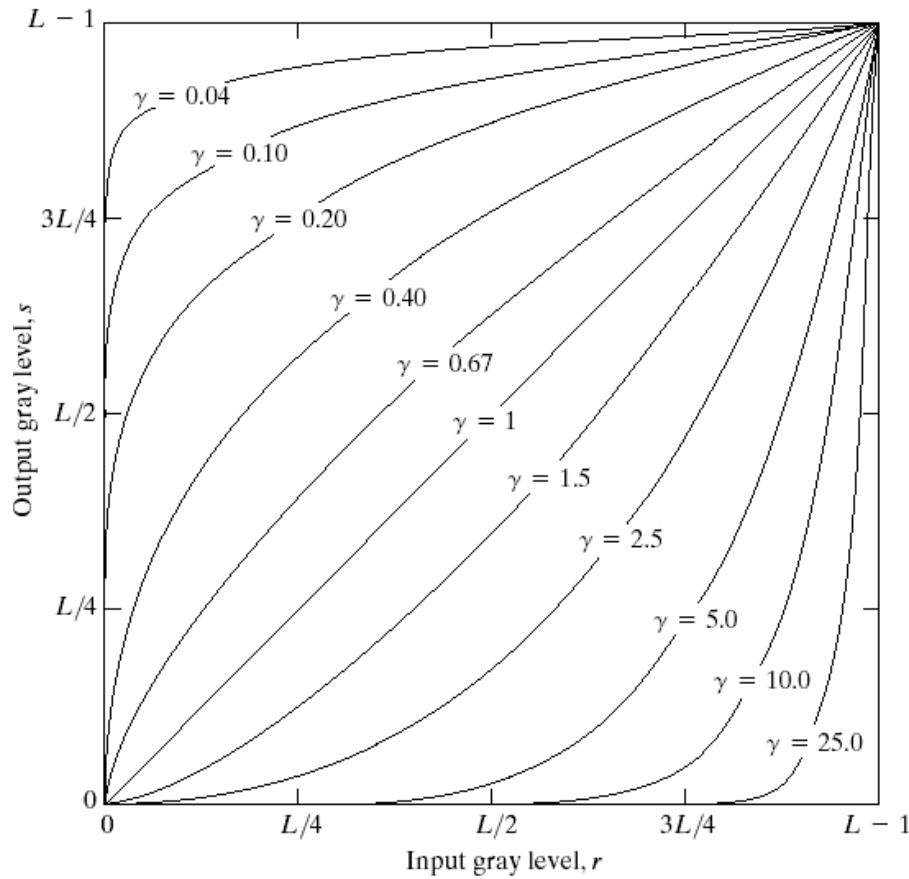
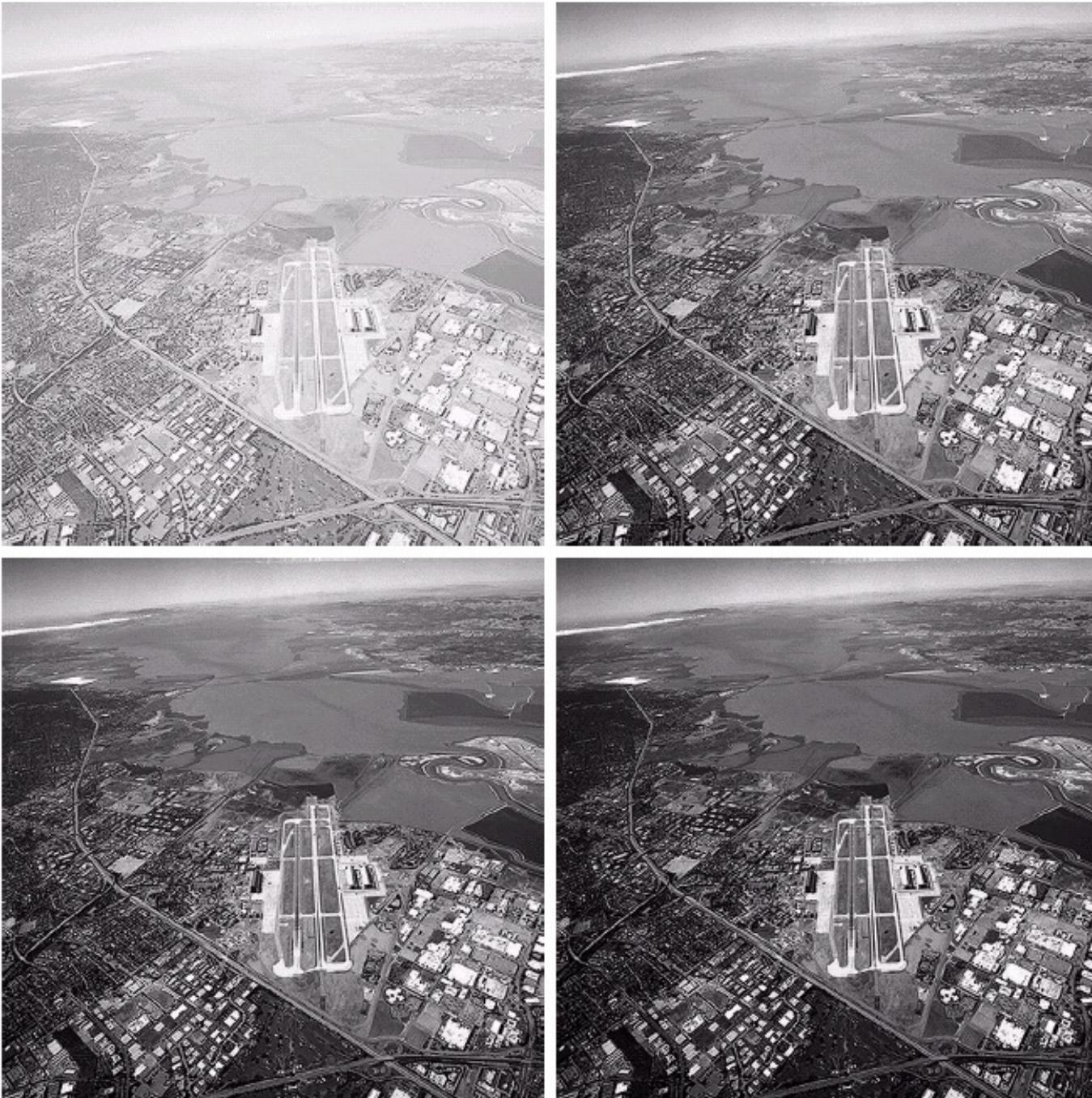


FIGURE 3.6 Plots of the equation $s = cr^\gamma$ for various values of γ ($c = 1$ in all cases).

a
b
c
d

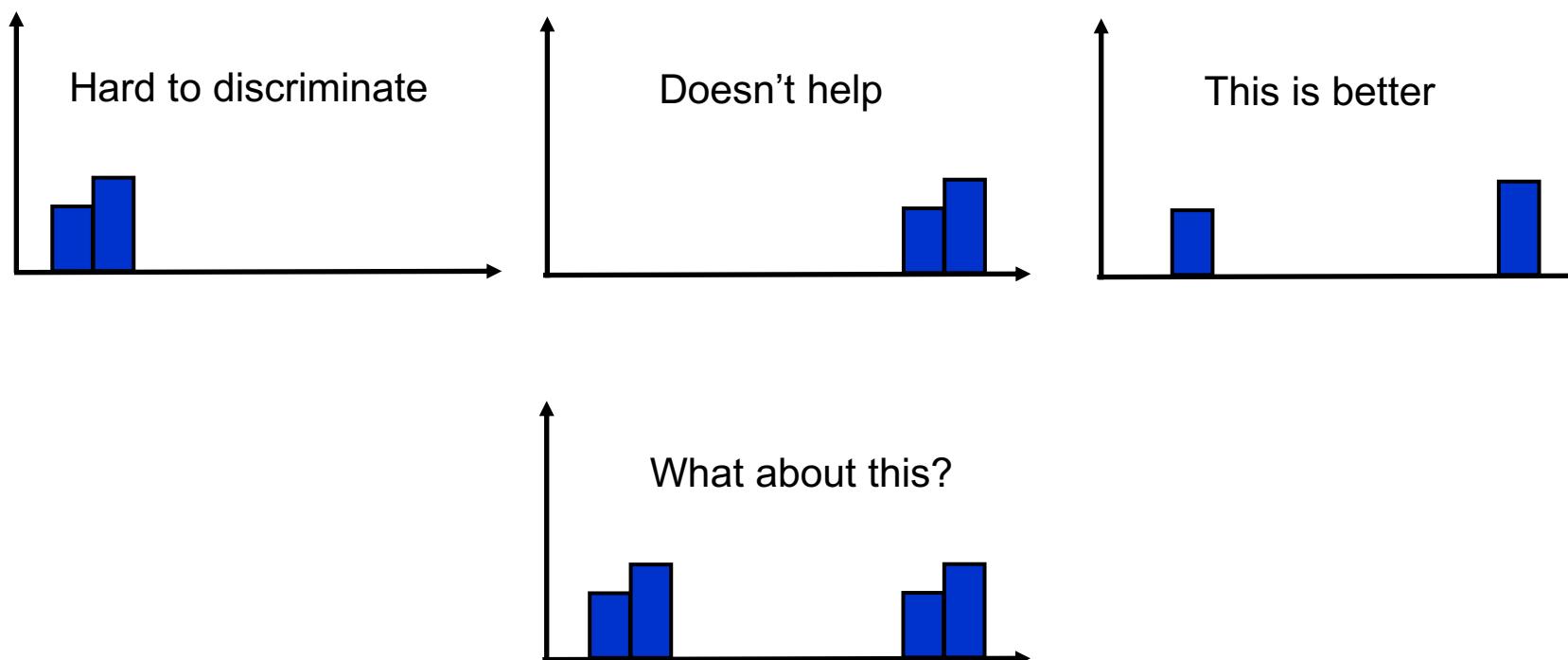
FIGURE 3.9

(a) Aerial image.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0 , respectively. (Original image for this example courtesy of NASA.)



Histogram Equalization

- Visual contrast between objects depends on their gray-level separation.
- How can we improve the contrast **after** an image has been acquired?



Histogram Equalization

- For a better visual discrimination of an image, we would like to re-assign gray-levels so that gray-level resource will be optimally assigned.
- **Our goal:** finding a gray-level transformation $M(v)$ such that:
 - The histogram H_b is as flat as possible.
 - The order of gray-levels is maintained.
 - The histogram bars are not fragmented.

Algorithm: How to implement histogram equalization?

Step 1:For images with discrete gray values, compute:

$$p_{in}(r_k) = \frac{n_k}{n} \quad 0 \leq r_k \leq 1 \quad 0 \leq k \leq L-1$$

L: Total number of gray levels

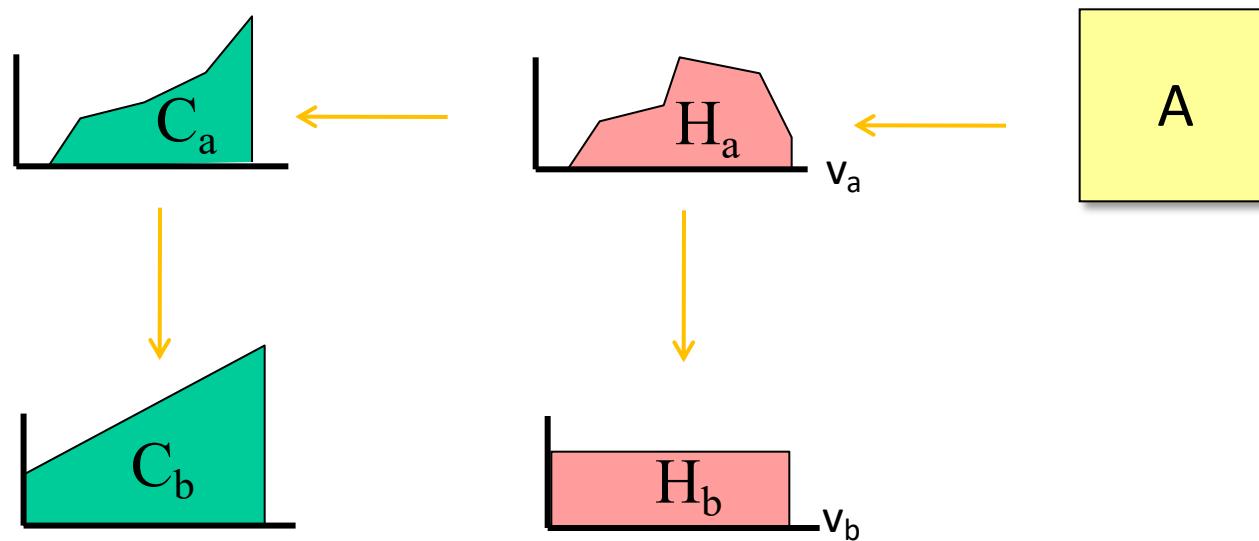
n_k : Number of pixels with gray value r_k

n: Total number of pixels in the image

Step 2: Based on CDF, compute the transformation :

$$s_k = T(r_k) = \sum_{j=0}^k p_{in}(r_j) \quad 0 \leq k \leq L-1$$

Histogram Equalization: Algorithm



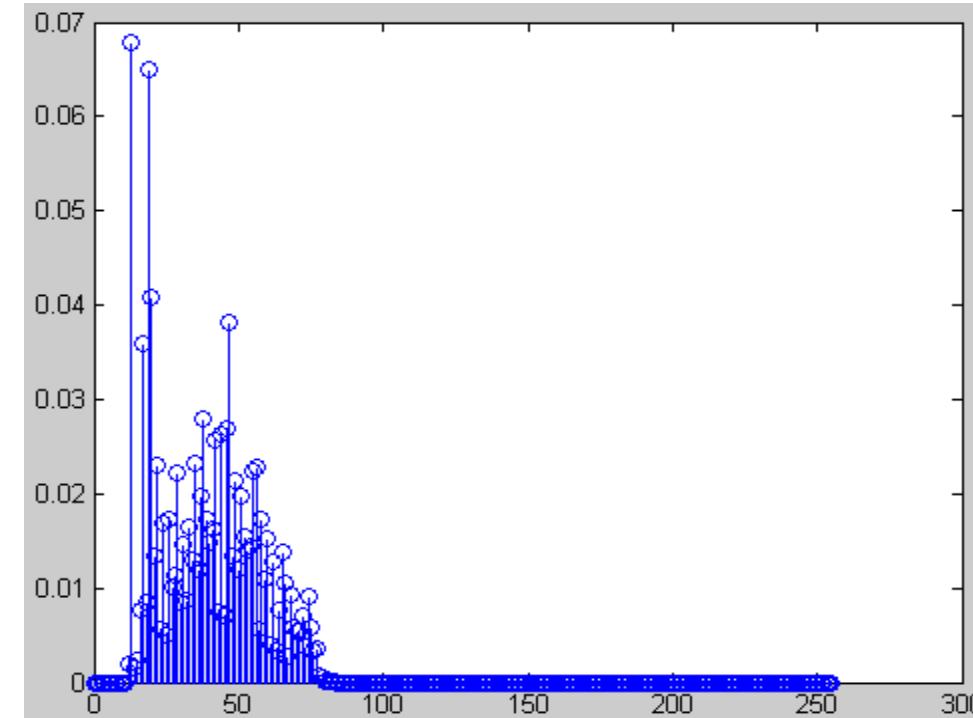
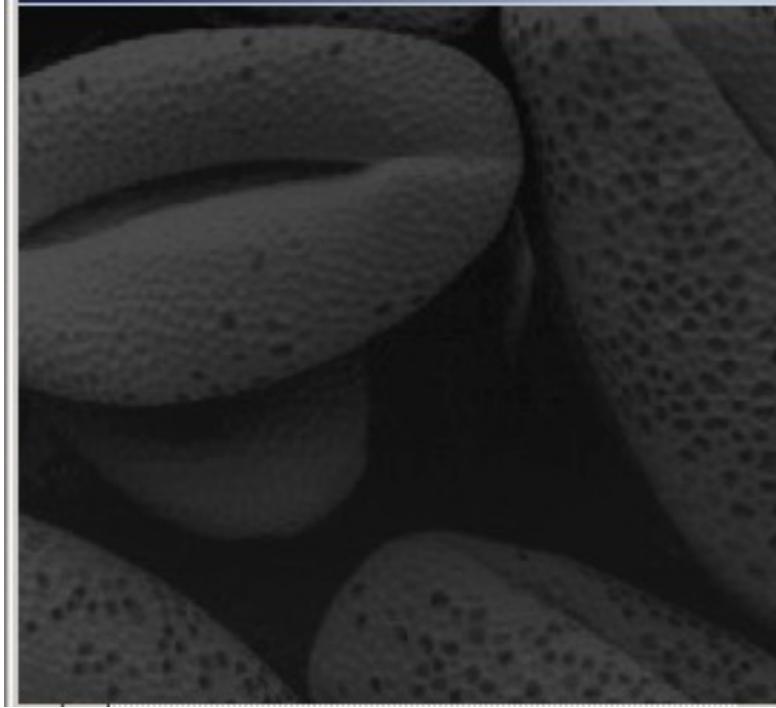
- Define:
- Assign:

$$C_b(v_b) = \sum_{j=0}^{v_b} \frac{1}{L}$$

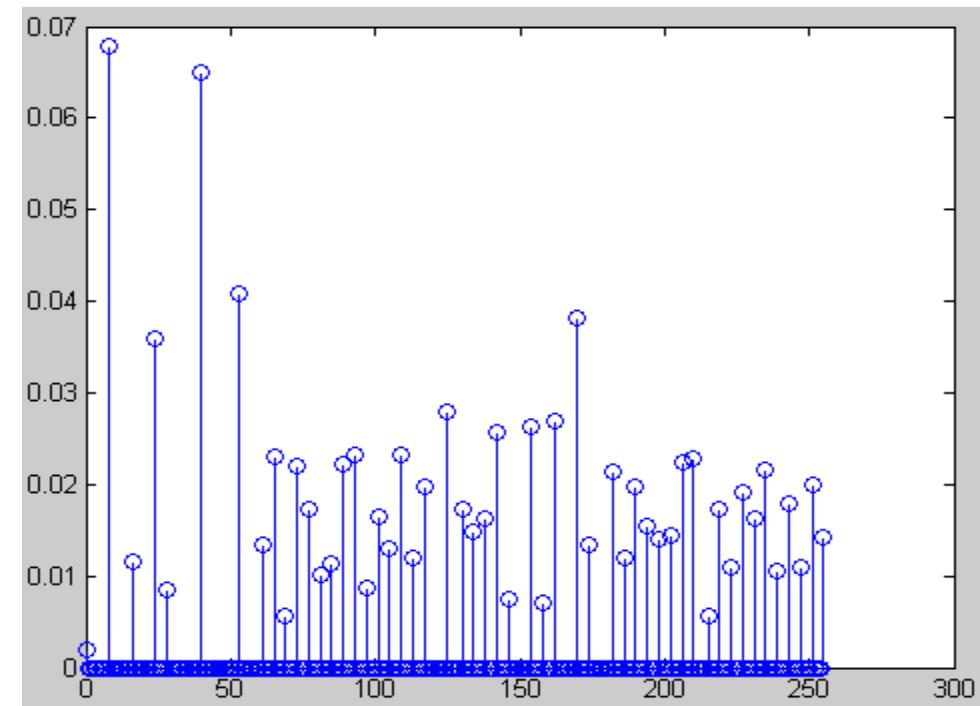
$$v_b = C_b^{-1}(C_a(v_a)) = M(v_a)$$

Example

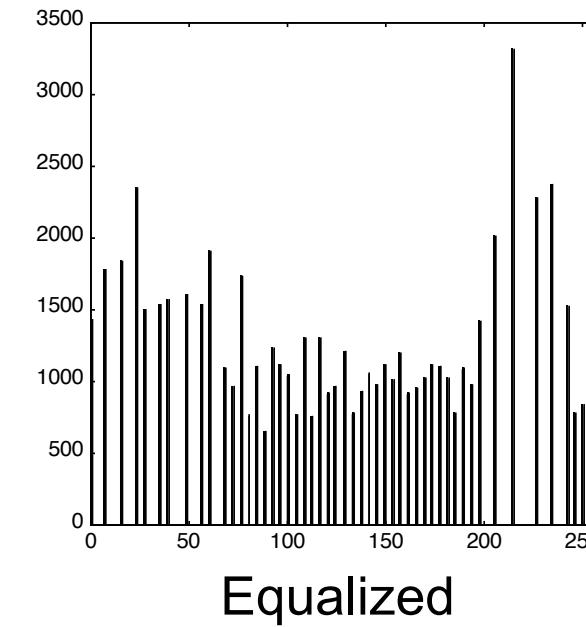
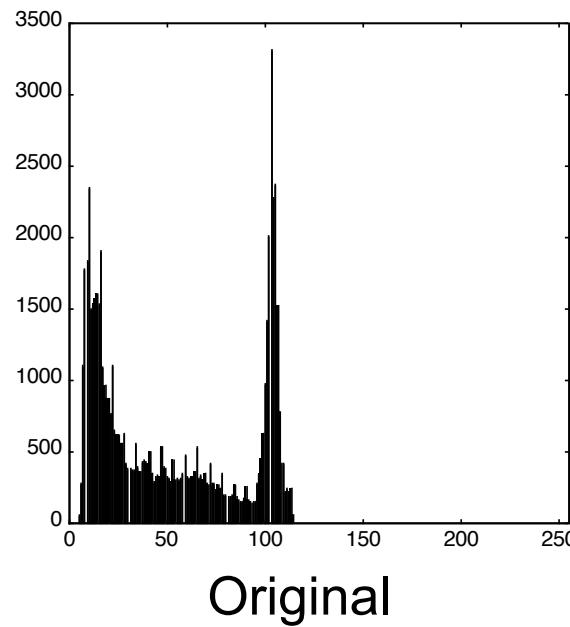
Original image and its histogram



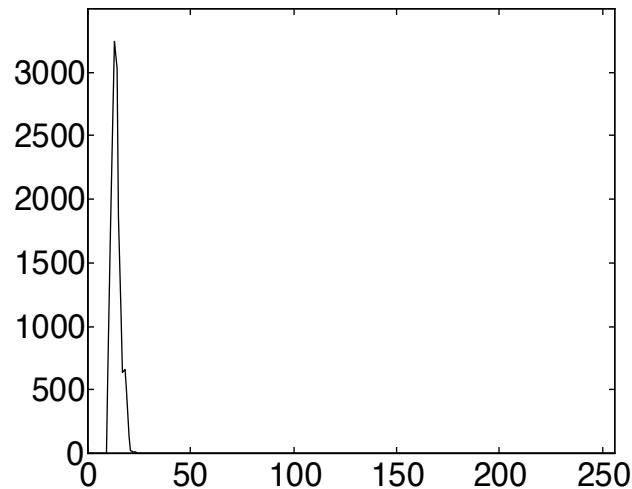
Histogram equalized image and its histogram



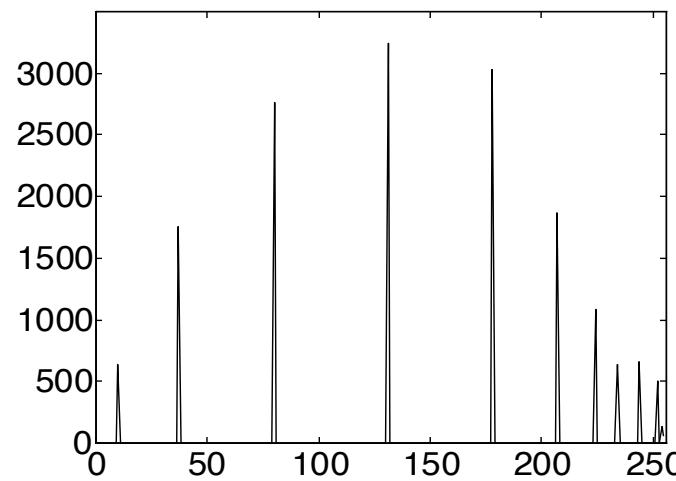
Hist. Eq.: Example 1



Hist. Eq.: Example 2



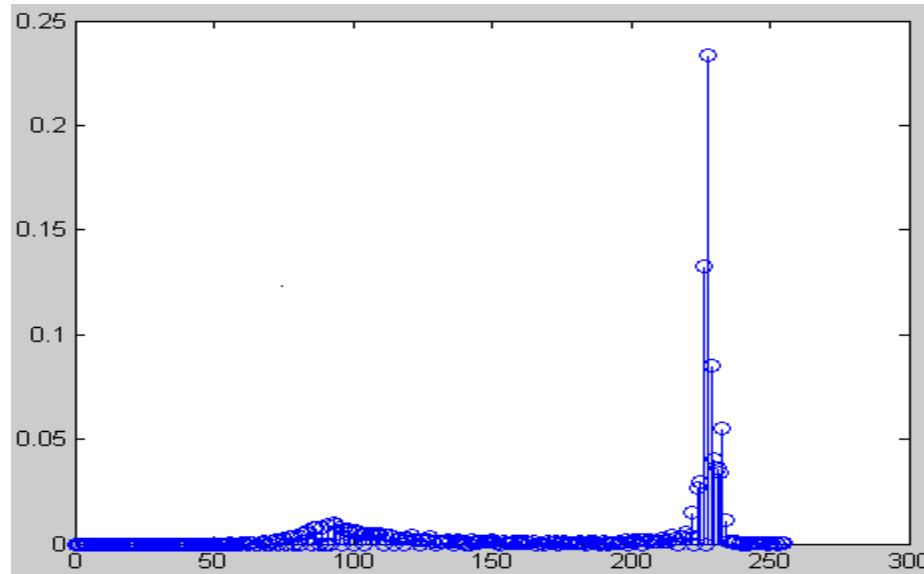
Original

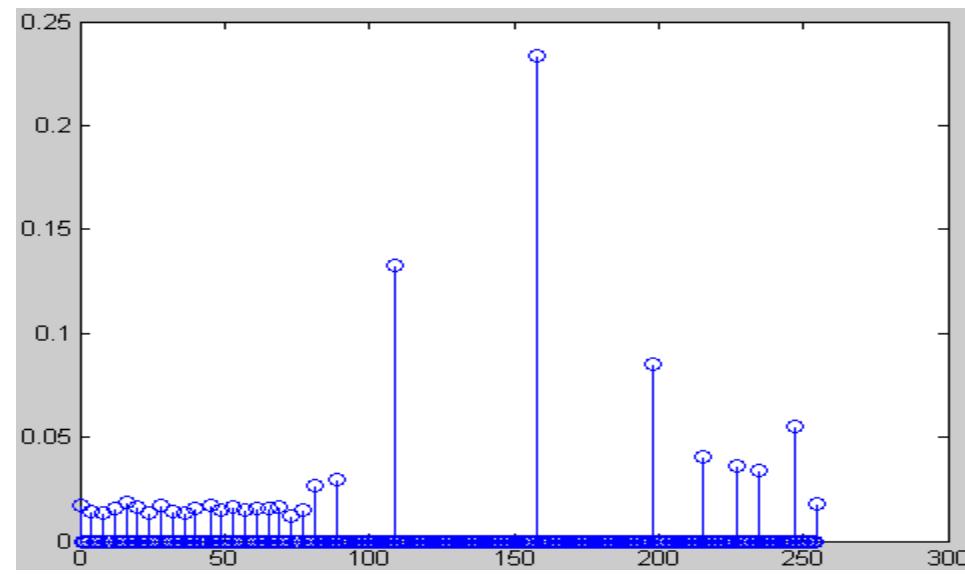
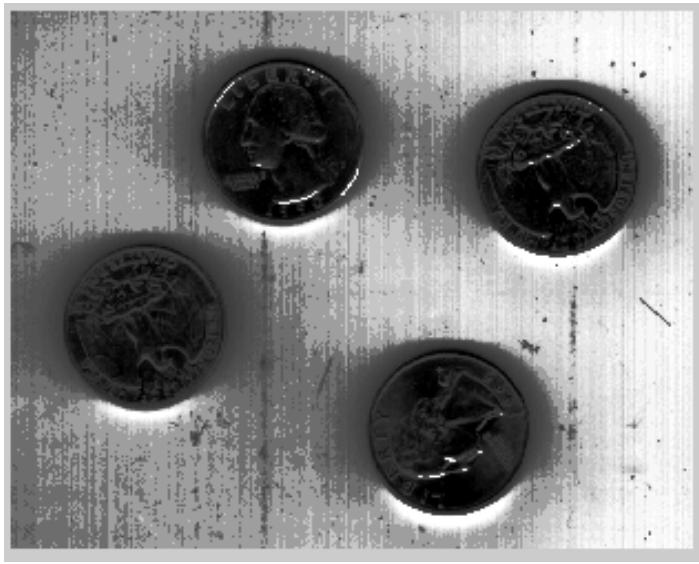


Equalized

- Comments:

Histogram equalization may not always produce desirable results, particularly if the given histogram is very narrow. It can produce false edges and regions. It can also increase image “graininess” and “patchiness.”





Geometric Transformations

(spatial transformation, image warping)

Image Warping



<http://www.jeffrey-martin.com>

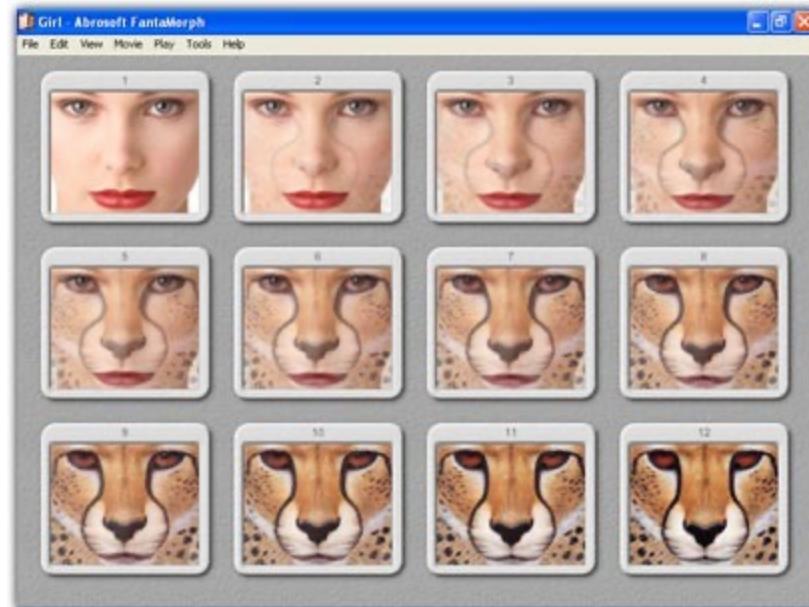
Image Warping

Why?

- texture mapping



- image processing (rotation, zoom in/out, etc)



- image morphing/blending

- image mosaic (stitching)

- image based-modeling & rendering



Image Warping

- Point operation and neighborhood operation:
- change **range** of image intensity value

$$g(x) = h(f(x))$$

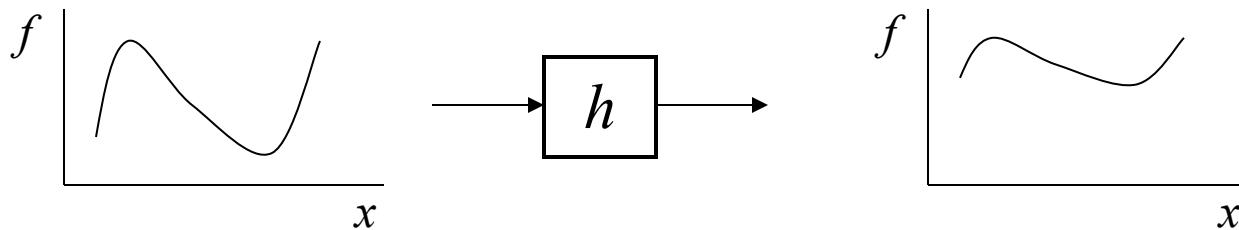


Image warping: change the spatial **domain** of image

$$g(x) = f(h(x))$$

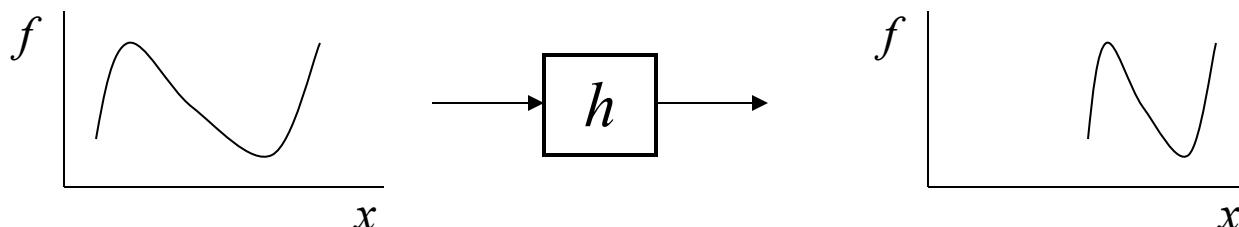


Image Warping

Image filtering: change *range* of image

$$g(x) = h(f(x))$$

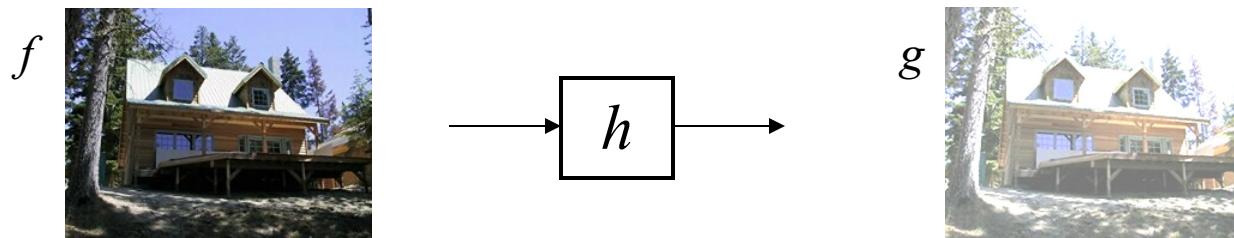
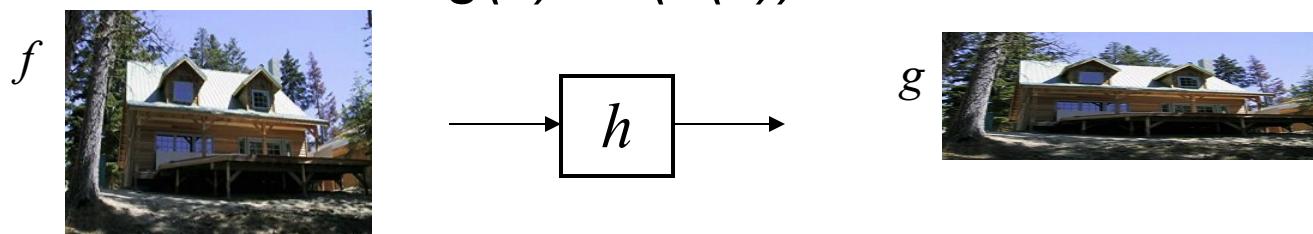


Image warping: change *domain* of image

$$g(x) = f(h(x))$$



Types of Image Warping

Simple geometric transformations:

- Rotation;
- Similarity
- Affine mapping
- Projective mapping

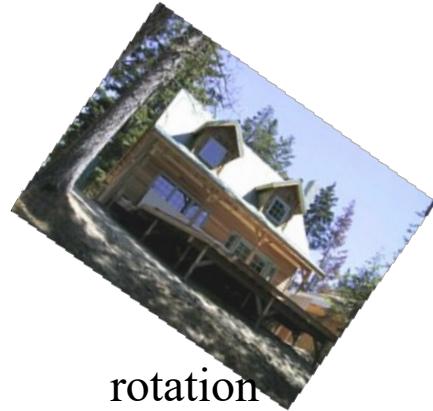
Other general types of transformations
free-form deformation

Geometric transformation

Examples of transformation



translation



rotation



aspect



affine

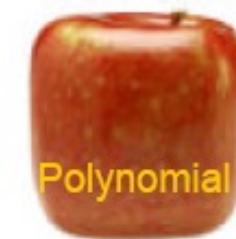
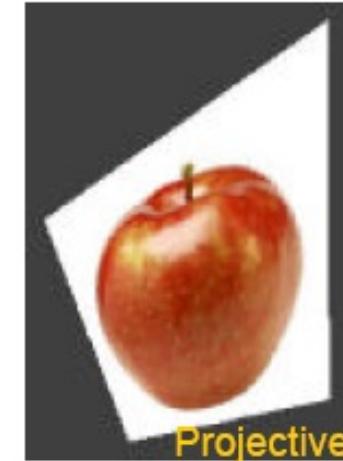
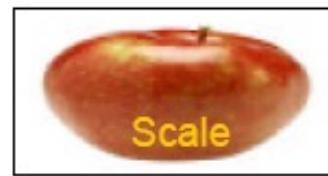
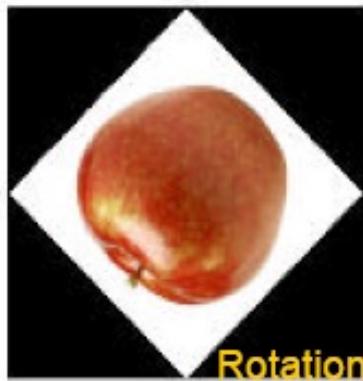


perspective



cylindrical

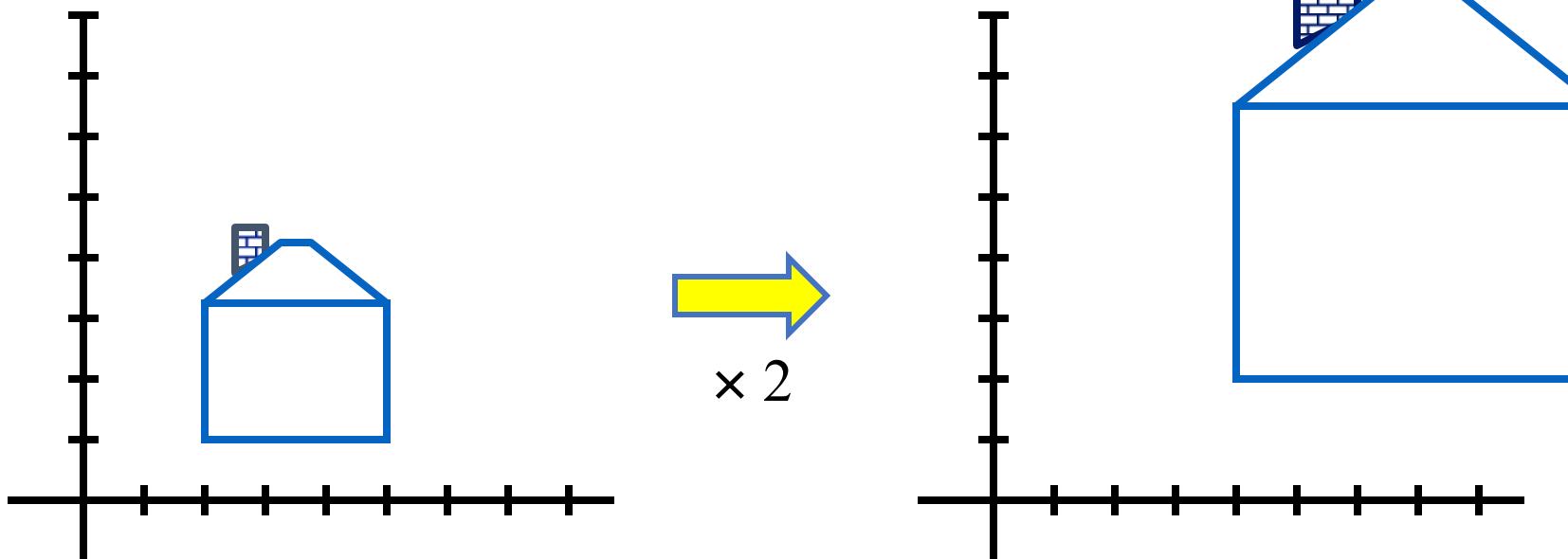
Types of Transformation



MATLAB functions: griddata, interp2, maketform, imtransform

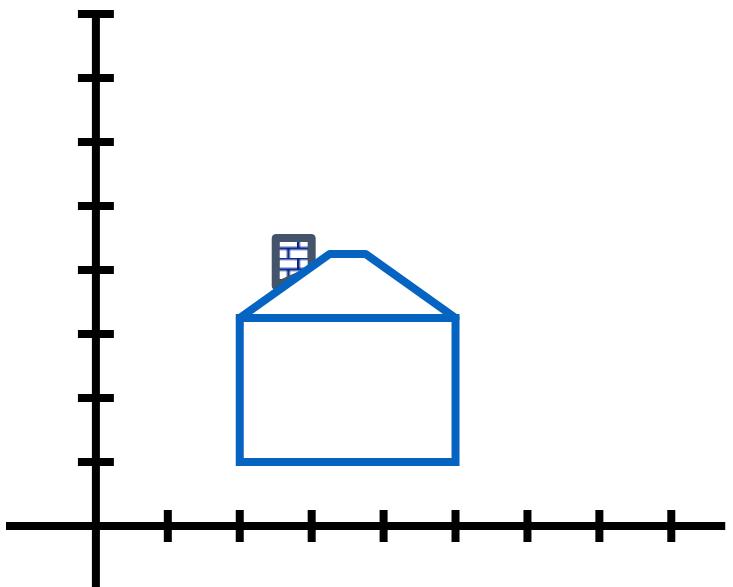
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:

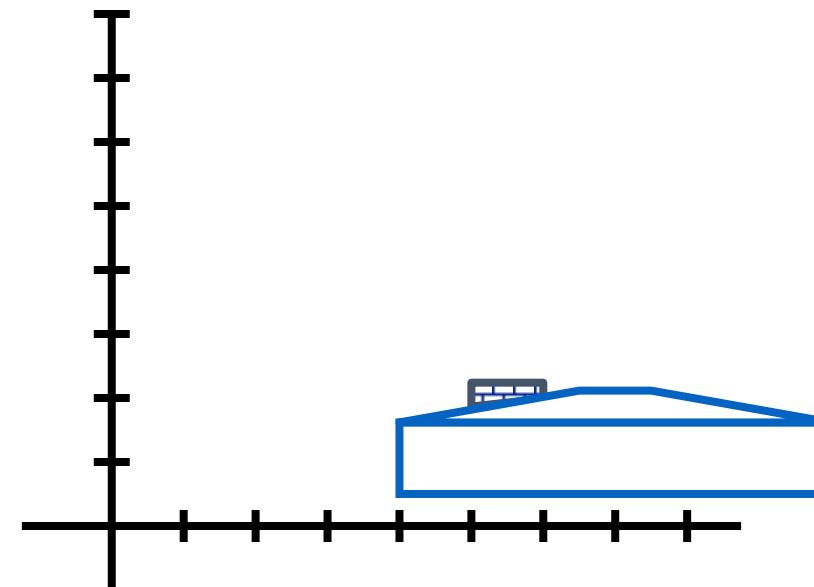


Scaling

- *Non-uniform scaling*: different scalars per component:



→
 $x' = x \times 2;$
 $y' = y \times 0.5$



Scaling

- Scaling operation:

$$x' = ax$$

$$y' = by$$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

What's the inverse of S?

Scaling

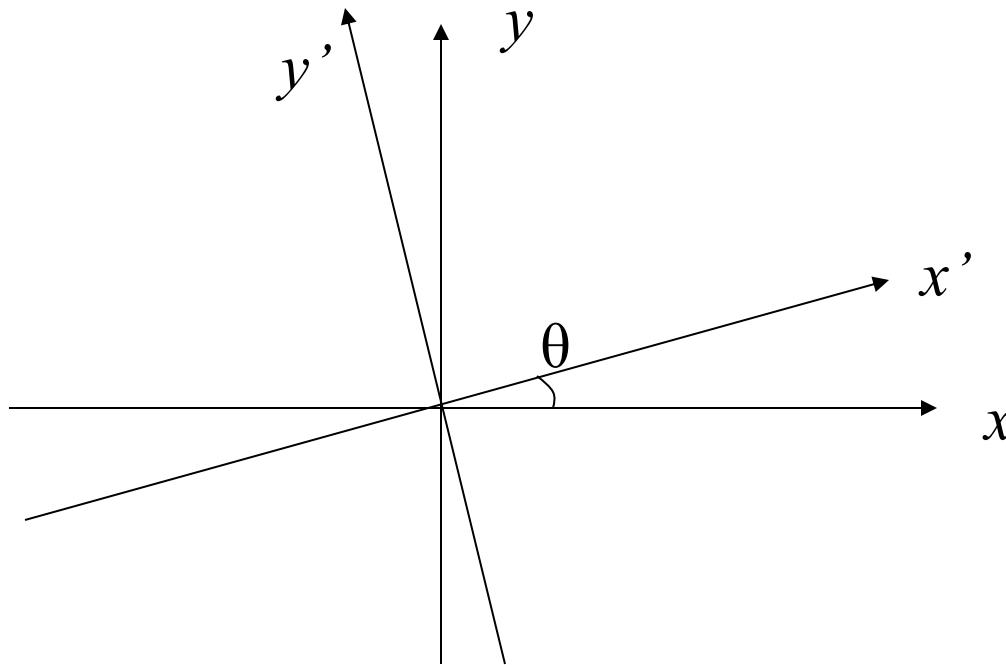


$\xrightarrow{a=2}$



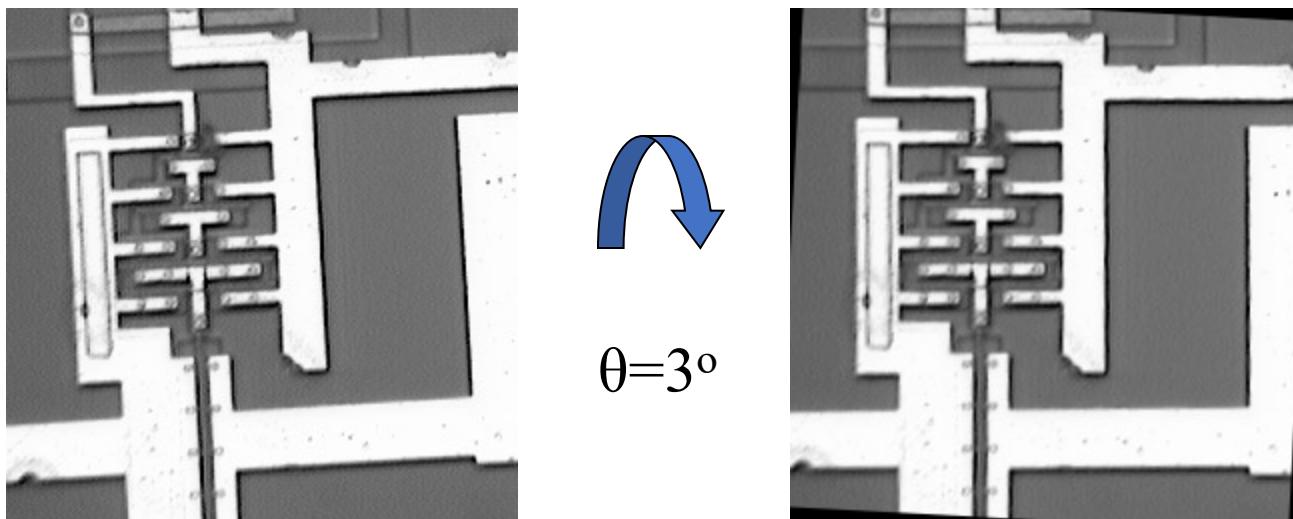
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & 1/a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation

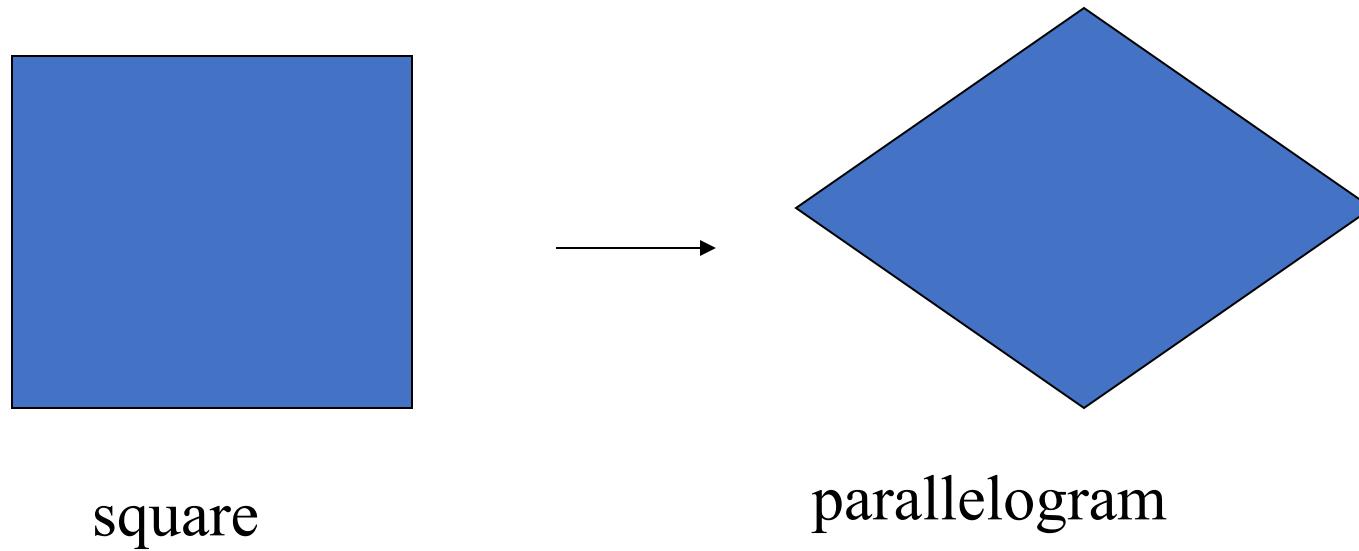


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation Example



Affine Transform

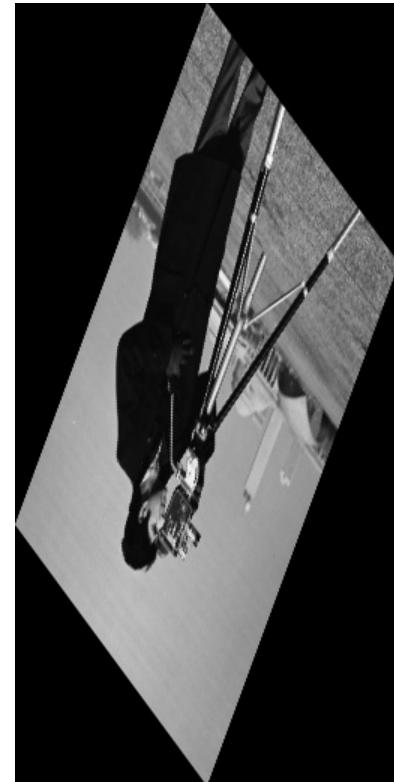


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

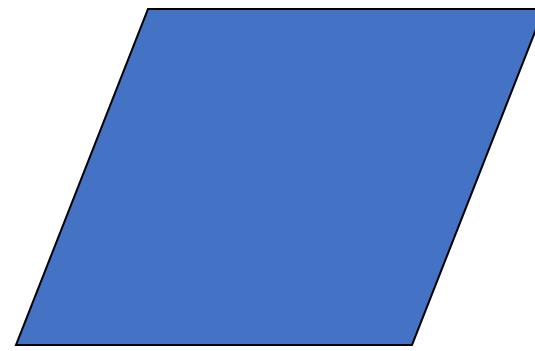
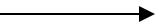
Affine Transform Example



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} .5 & 1 \\ .5 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



Shear



square

parallelogram

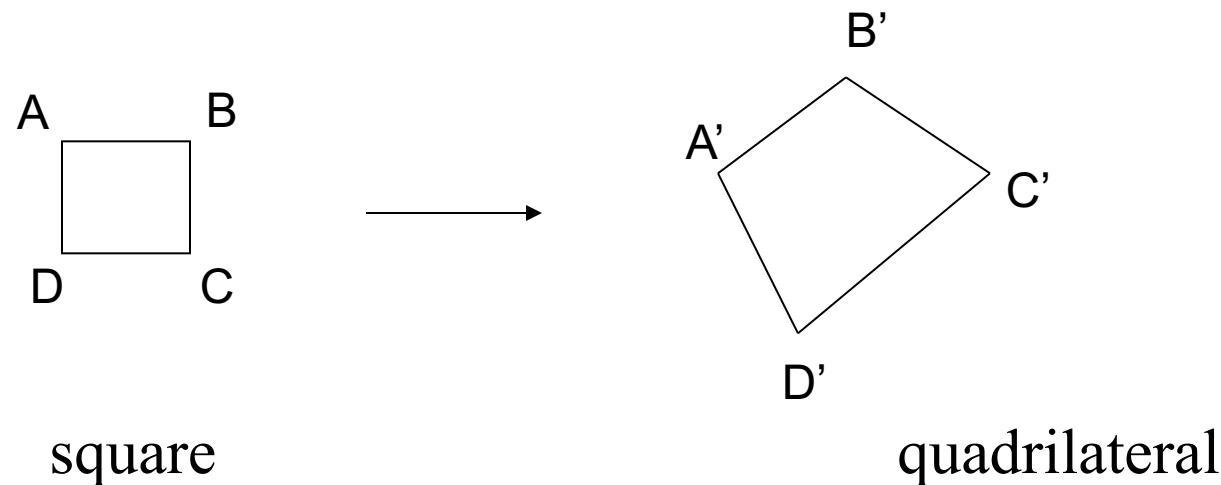
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Shear Example



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ .5 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

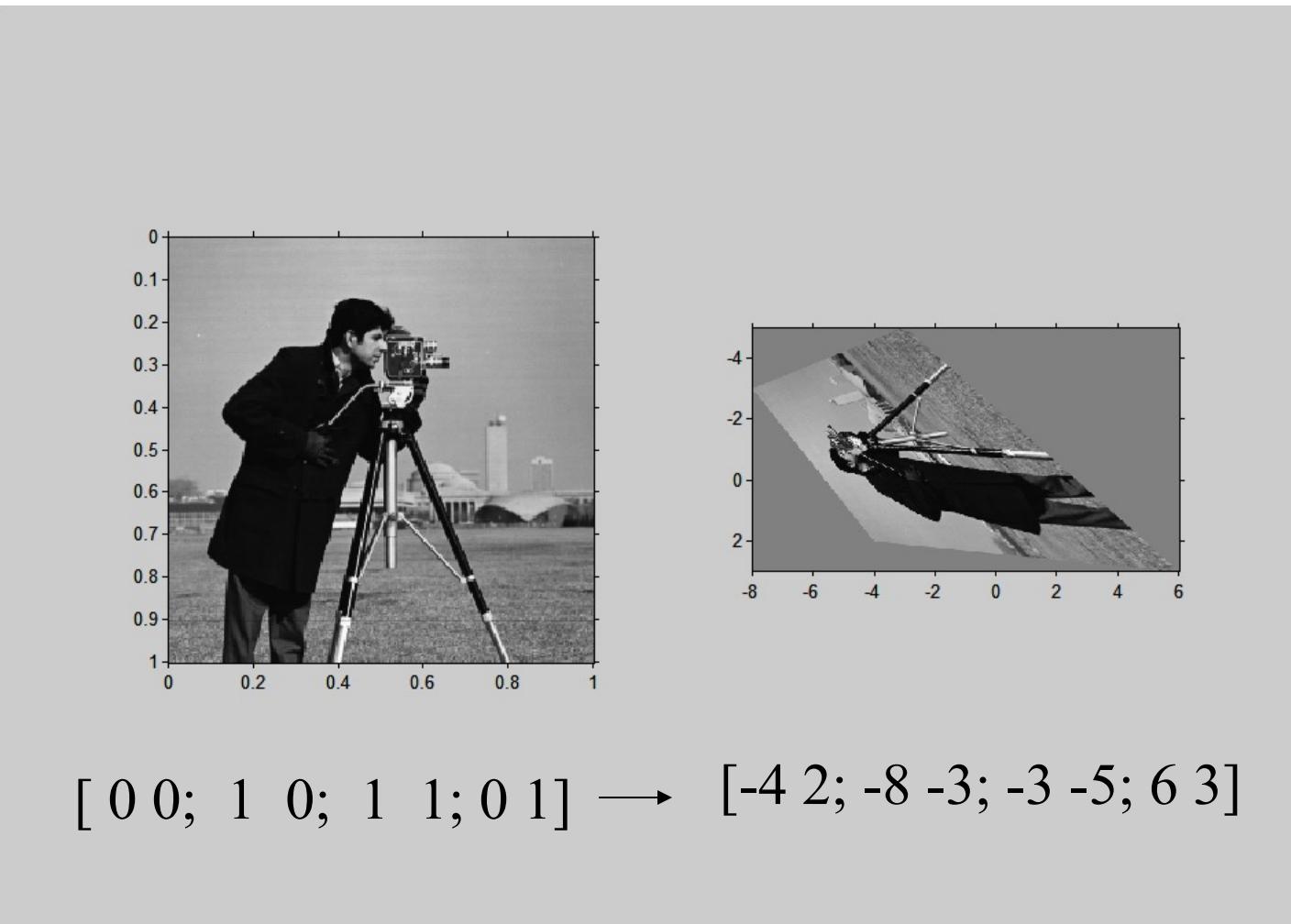
Projective Transform



$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4x + a_5y + a_6}{a_7x + a_8y + 1}$$

Projective Transform Example



$$[0\ 0; 1\ 0; 1\ 1; 0\ 1] \rightarrow [-4\ 2; -8\ -3; -3\ -5; 6\ 3]$$

Projective Image stitching



How to compute image warping ?

Definition: Image Warping

Source Image: Image to be used as the reference. The geometry of this image is not changed

Target Image: this image is obtained by transforming the reference image.

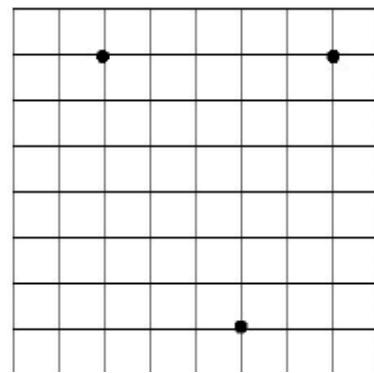
(x,y): coordinates of points in the reference/source image

(x',y'): (or [u,v]) coordinates of points in the target image

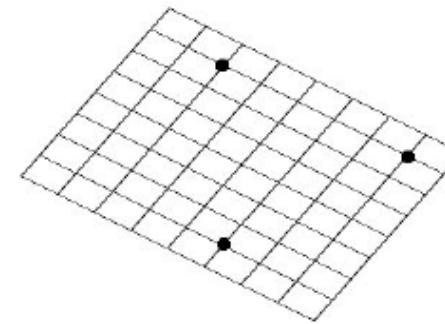
Definition: Image Warping

Control points: Unique points in the reference and target images.

The coordinates of corresponding control points in images are used to determine a transformation function.

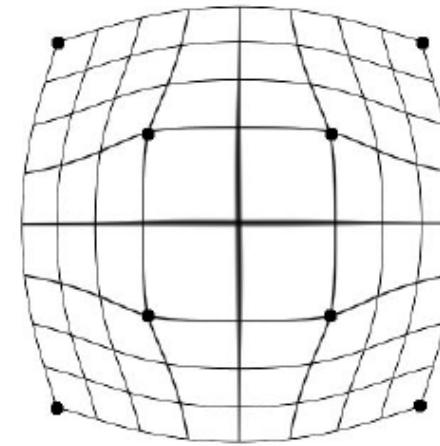
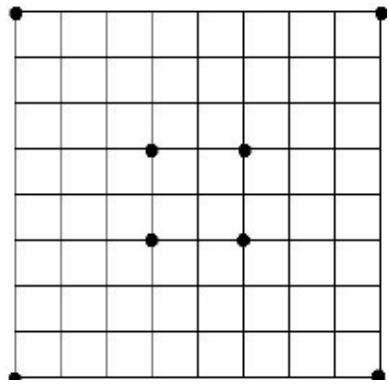


Source Image



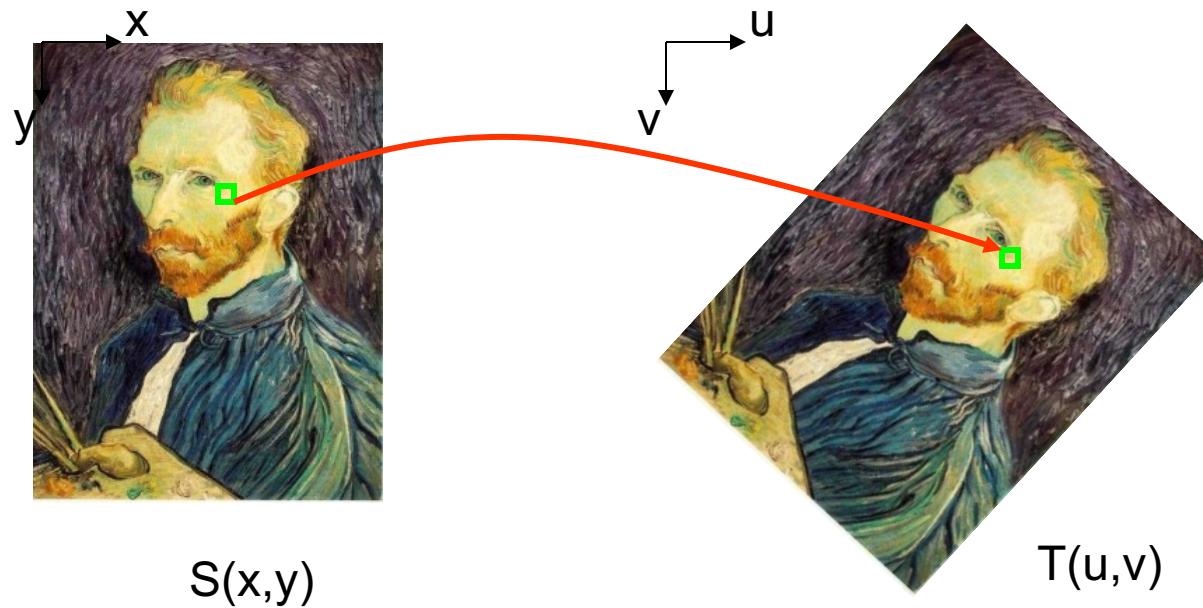
Target Image

Transformation Function



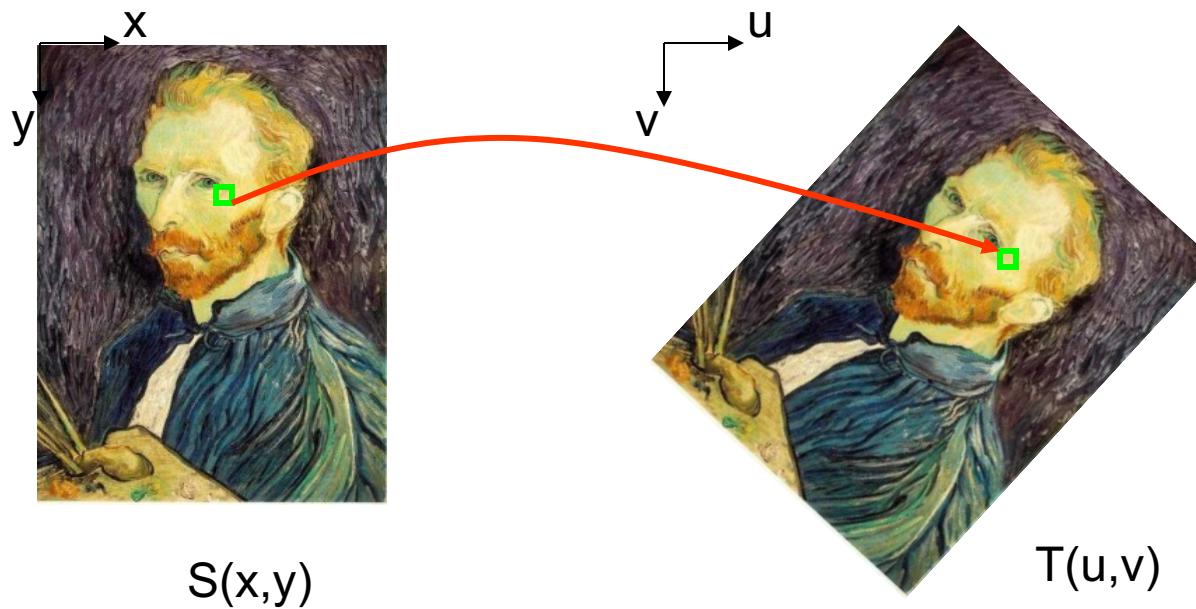
Transform the domain of an image to a desired geometry

Image Warping Computation



- Given a coordinate transform function $[f, g]$, where $u = f(x,y); v = g(x,y)$, (or the inverse i.e., $[F,G]$) and source image $S(x,y)$,
- how do we compute a transformed target image $T(u,v)$?

Forward Warping



Forward warping algorithm:

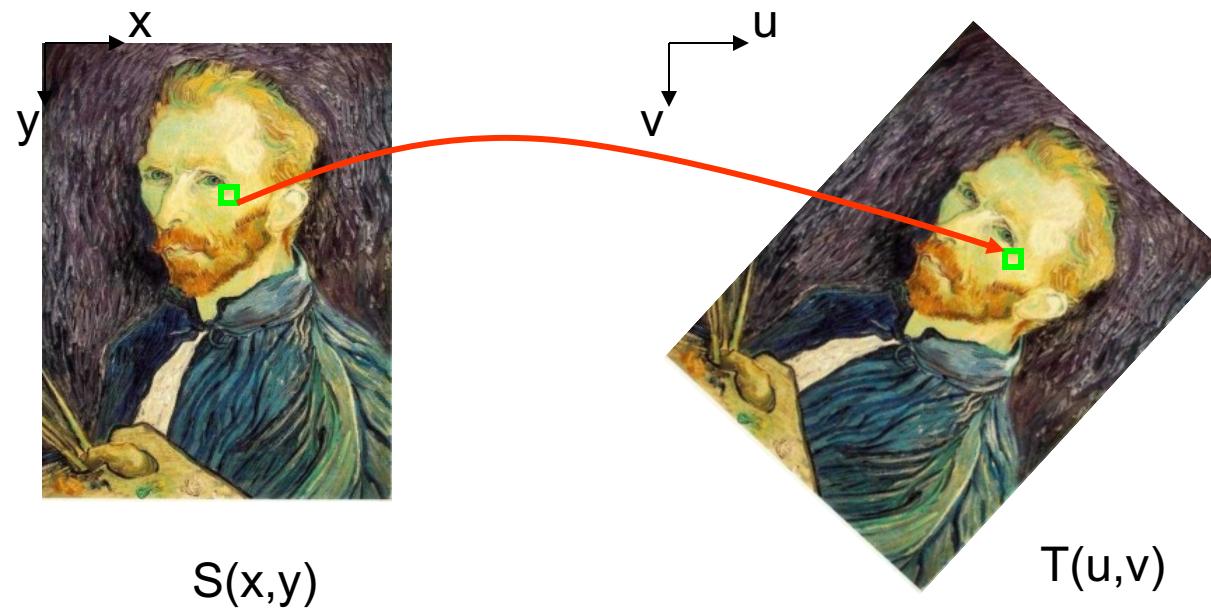
for $y = y_{\min}$ to y_{\max}

 for $x = x_{\min}$ to x_{\max}

$$u = f(x,y); v = g(x,y)$$

 copy pixel intensity value or color at source image: $S(x,y)$ to target image $T(u,v)$

Forward Warping



Forward warping algorithm:

for $y = y_{\min}$ to y_{\max}

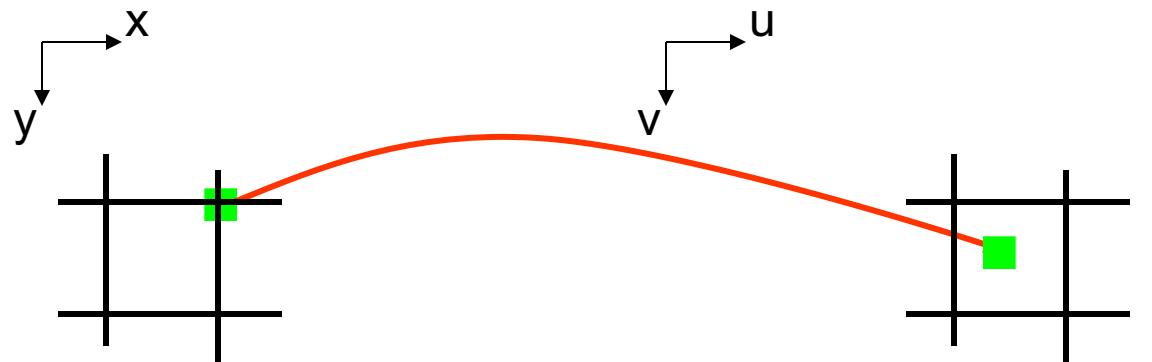
 for $x = x_{\min}$ to x_{\max}

$u = f(x,y); v = g(x,y)$

 copy pixel intensity value or color at source image: $S(x,y)$ to target image $T(u,v)$

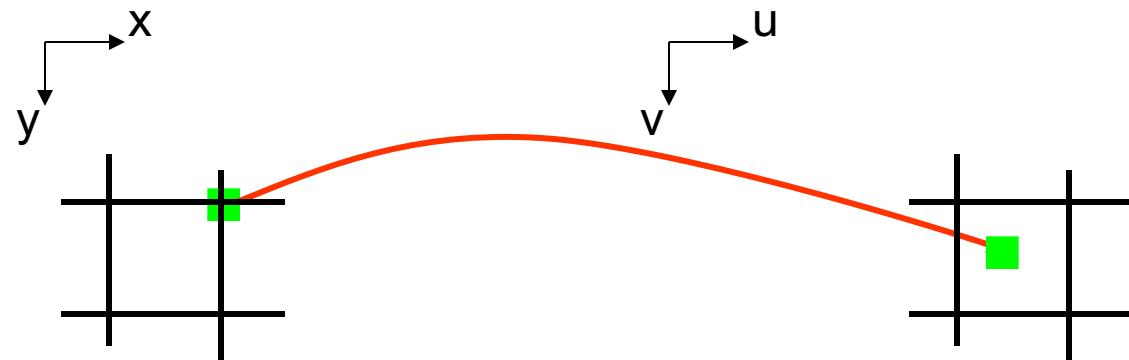
- Any problems for forward warping?

Forward Warping



Q: What if the transformed pixel located between pixels?

Forward Warping



Q: What if the transformed pixel located between pixels?

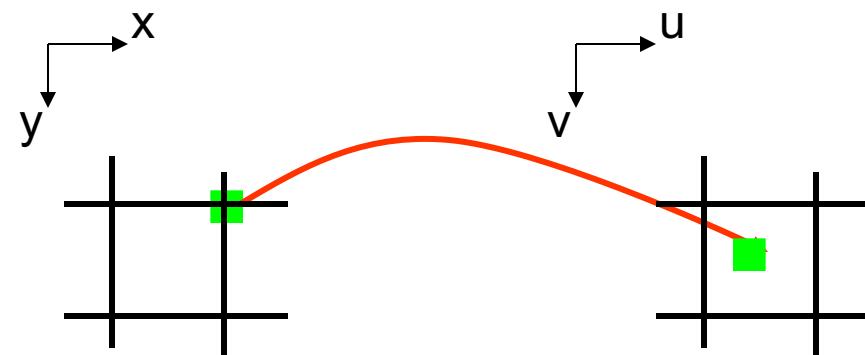
A: Distribute color among neighboring pixels

- known as “splatting”

Forward Warping

- Iterate over source, sending pixels to destination
- Some source pixels map to the same target pixel
- Some target pixels may have no corresponding source
- Holes in reconstruction
- Must splat etc.

```
for y = ymin to ymax
    for x = xmin to xmax
        u = f(x,y); v = g(x,y)
        copy pixel intensity values at source S(x,y) to T(u,v)
```



Forward Warping

- Iterate over source, sending pixels to destination
- Some source pixels map to the same target pixel
- Some target pixels may have no corresponding source
- Holes in reconstruction
- Must splat etc.

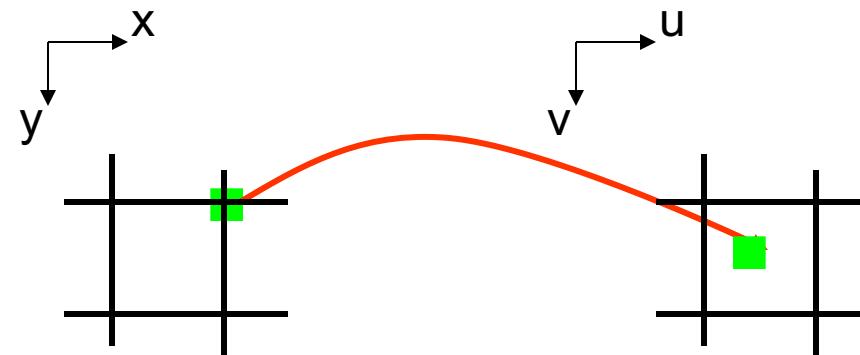
- How to remove the holes?

for $y = y_{\min}$ to y_{\max}

 for $x = x_{\min}$ to x_{\max}

$u = f(x,y); v = g(x,y)$

 copy pixel at source $S(x,y)$ to $T(u,v)$

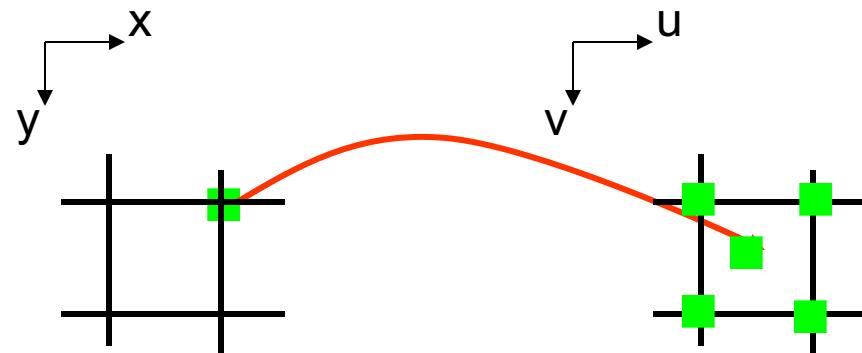


Forward Warping

- Iterate over source, sending pixels to destination
- Some source pixels map to the same target pixel
- Some target pixels may have no corresponding source
- Holes in reconstruction
- Must “splat” etc.

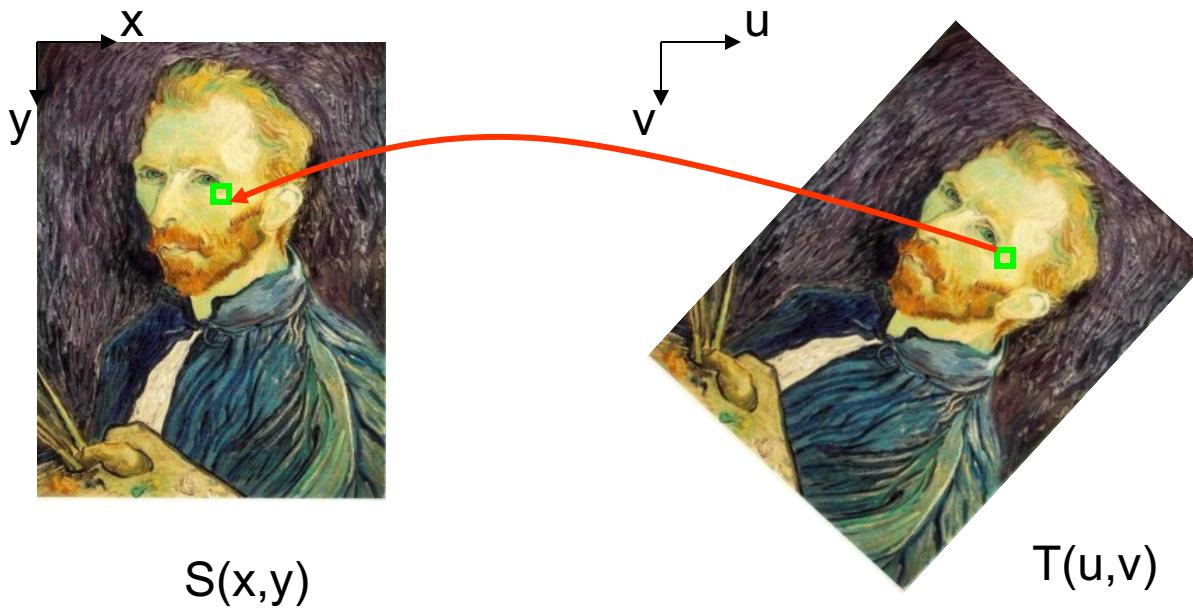
- How to remove the holes?

```
for y = ymin to ymax
    for x = xmin to xmax
        u = f(x,y); v = g(x,y)
        copy pixel at source S(x,y) to T(u,v)
```



copy pixel at source $S(x,y)$ to $T(u,v)$

Inverse Warping



Inverse warping algorithm:

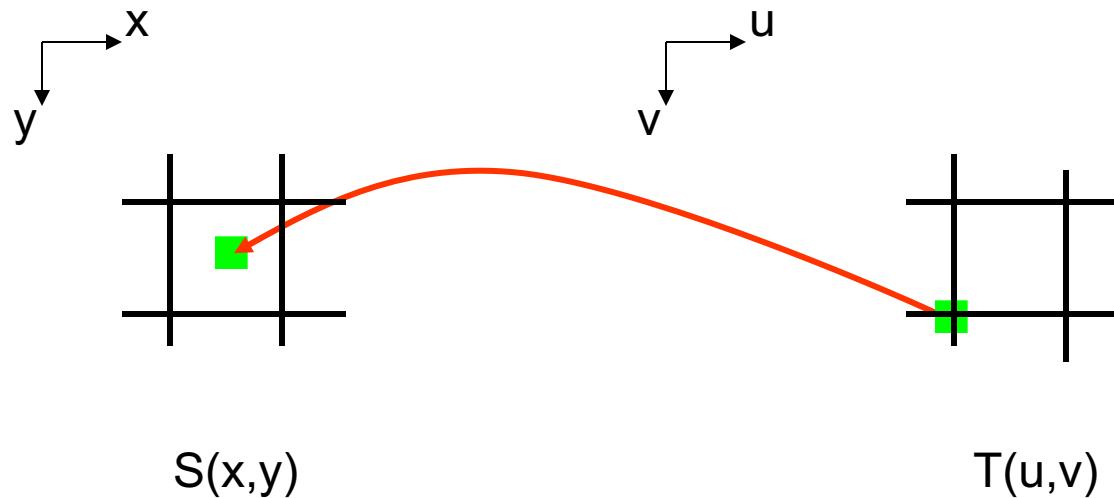
for $v = v_{\min}$ to v_{\max}

 for $u = u_{\min}$ to u_{\max}

$x = F(u,v); y = G(u,v)$

 copy pixel at source $S(x,y)$ to $T(u,v)$

Inverse Warping



Q: What if pixel comes from “between” two pixels?

A: Interpolate color values from neighboring pixels

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

Inverse Warping

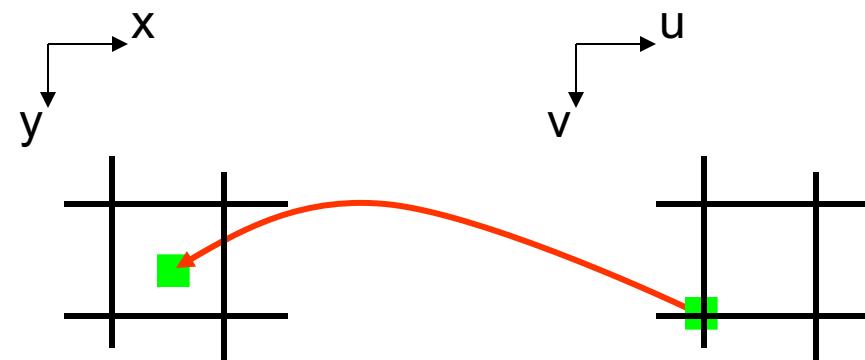
- Iterate over dest., finding pixels from source
- Non-integer evaluation source image, resample
- May oversample source
- But no holes
- Simpler, better than forward mapping

```
for v = vmin to vmax
```

```
    for u = umin to umax
```

```
        x = F(u,v); y = G(u,v)
```

```
        copy pixel at source S(x,y) to T(u,v)
```



Forward vs. inverse warping

- Q: which method is better?
- A: usually inverse—eliminates holes
 - however, it requires an invertible warp function—not always possible...

2D geometric transformations in homogeneous coordinates representation

Homogeneous Coordinates

- Q: How can we represent translation as a 3x3 matrix?

$$x' = x + t_x$$

$$y' = y + t_y$$

Homogeneous Coordinates

- A: Using the rightmost column:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

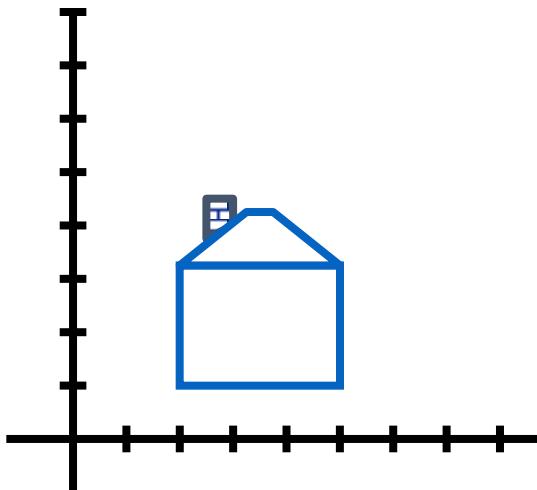
Translation

- Example of translation

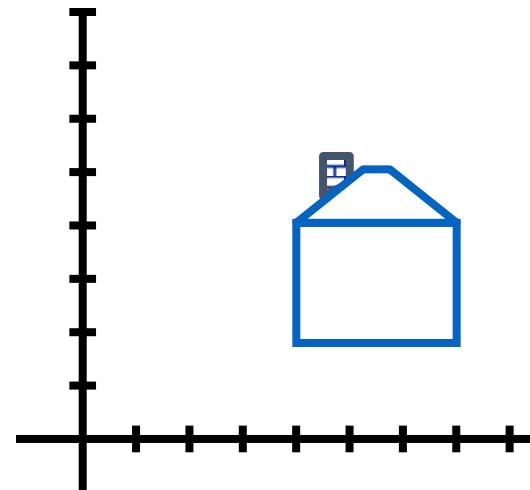
Homogeneous Coordinates



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$



$$t_x = 2$$
$$t_y = 1$$



Basic 2D Transformations in homogeneous representation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta & 0 \\ \sin \Theta & \cos \Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

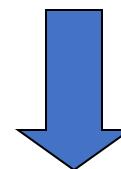
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

Affine Transformations

- Affine transformations are combinations of ...
 - Linear transformations, and
 - Translations
- Properties of affine transformations:
 - Origin does not necessarily map to origin
 - Lines map to lines
 - Parallel lines remain parallel
 - Ratios are preserved
 - Closed under composition
 - Models change of basis

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$



$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Transformation Composition

- Transformations can be combined by matrix multiplication

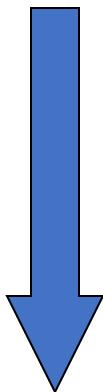
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \Theta & -\sin \Theta & 0 \\ \sin \Theta & \cos \Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$
$$\mathbf{p}' = T(t_x, t_y) R(\Theta) S(s_x, s_y) \mathbf{p}$$

Projective Transformations

- Projective transformations ...
 - Affine transformations, and
 - Projective warps
- Properties of projective transformations:
 - Origin does not necessarily map to origin
 - Lines map to lines
 - Parallel lines do not necessarily remain parallel
 - Ratios are not preserved
 - Closed under composition
 - Models change of basis

$$x' = \frac{a_1x + a_2y + a_3}{a_7x + a_8y + 1}$$

$$y' = \frac{a_4x + a_5y + a_6}{a_7x + a_8y + 1}$$



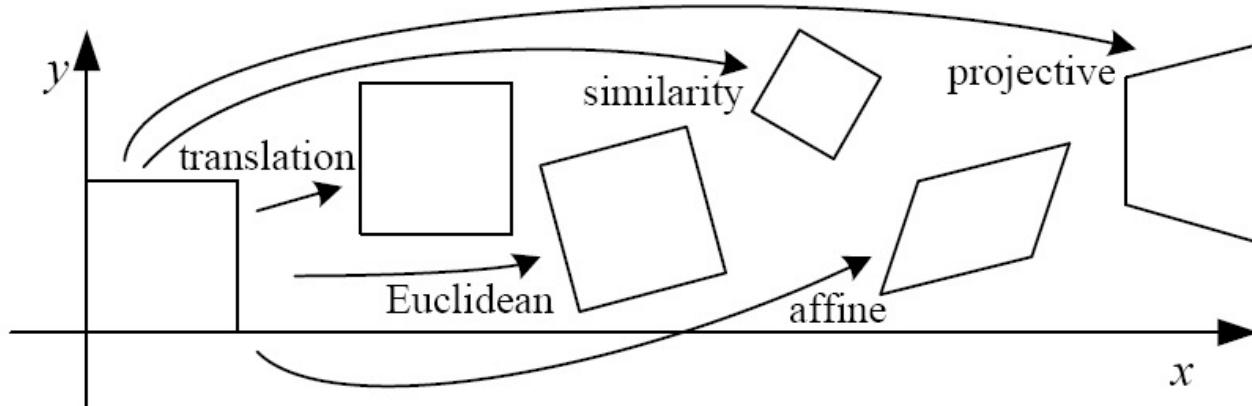
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Perspective effects



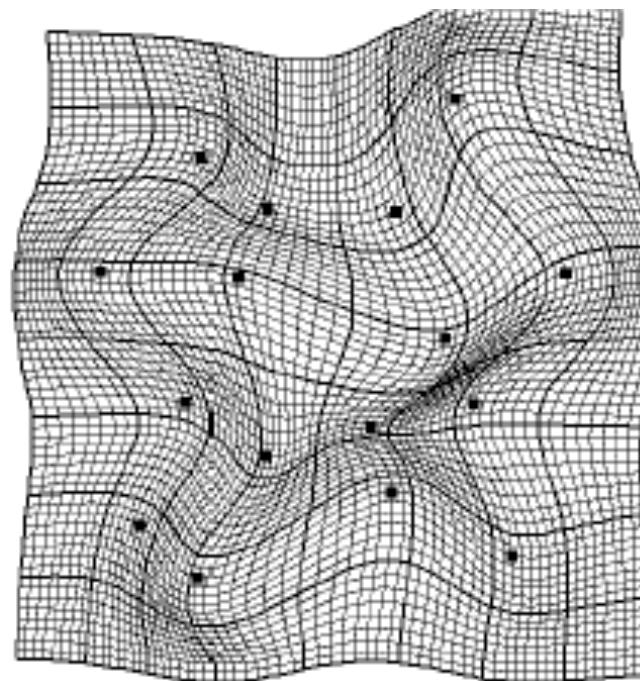
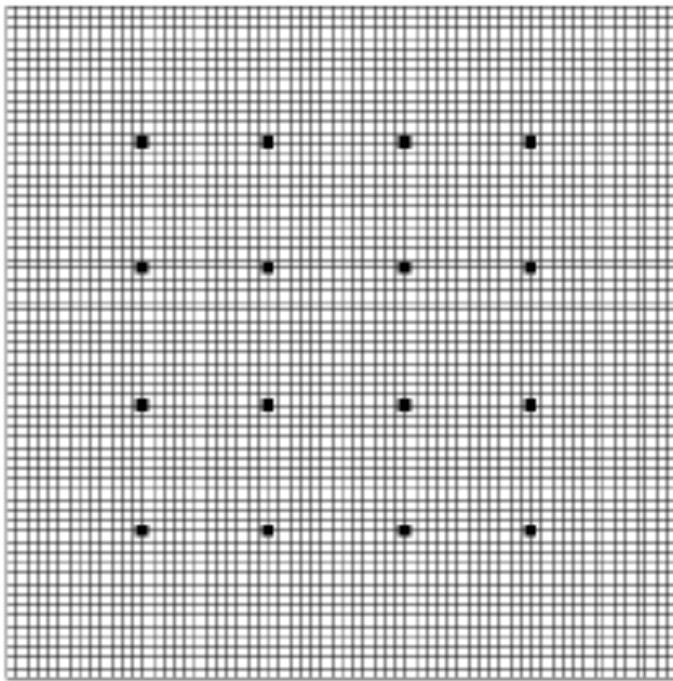
3D computer vision aims to solve an inverse problem of computer graphics

Hierarchy of 2D image transformations (stratification)



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2\times 3}$	2	orientation + ...	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}$	3	lengths + ...	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}$	4	angles + ...	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2\times 3}$	6	parallelism + ...	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

Free Form Warping



Recap

- Point operation and histogram equalization can enhance the image contrast
- Geometrical operation is used in spatial transformation or image wrapping
- Neighborhood operation, image filtering, convolution, next lectures.