Image Processing Basics

Week-2

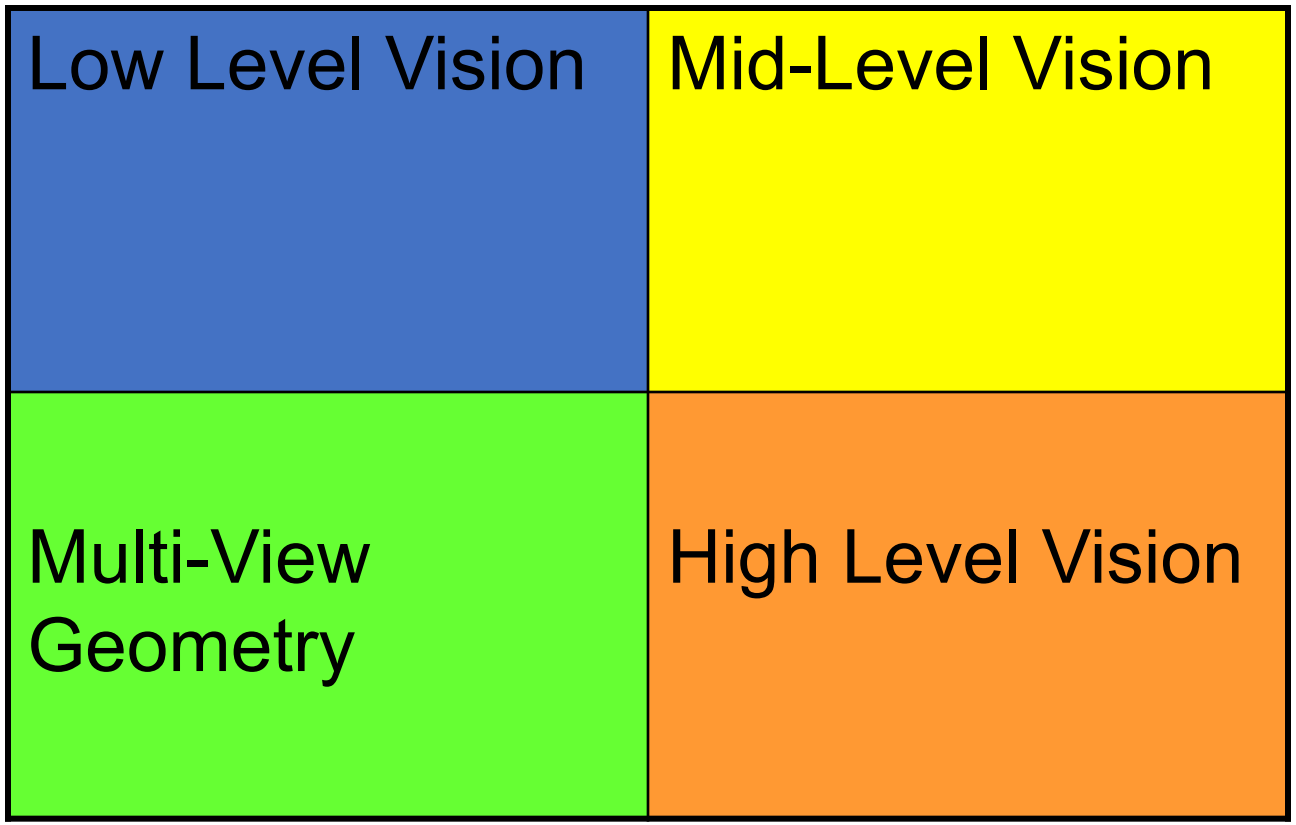
COMP/ENGN4528/6528 Computer Vision

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%

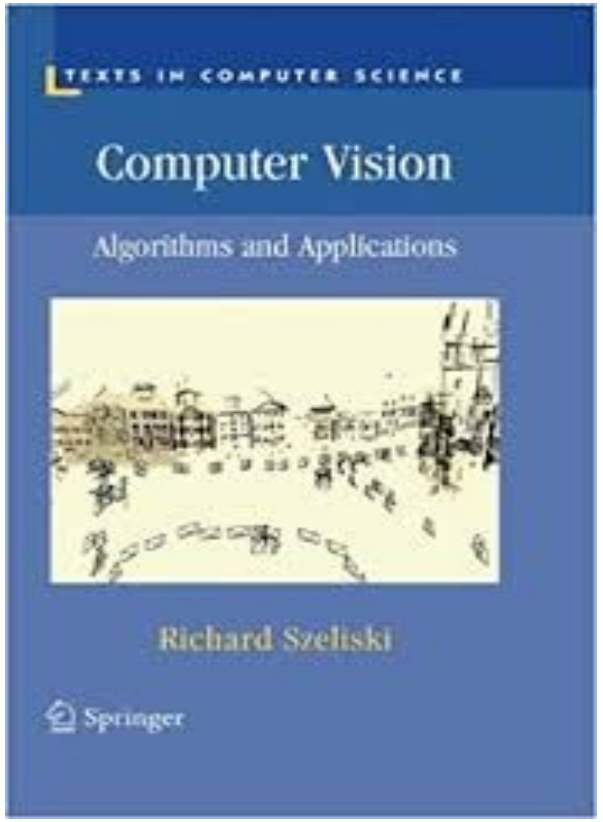
Four modules



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

- Read textbook:

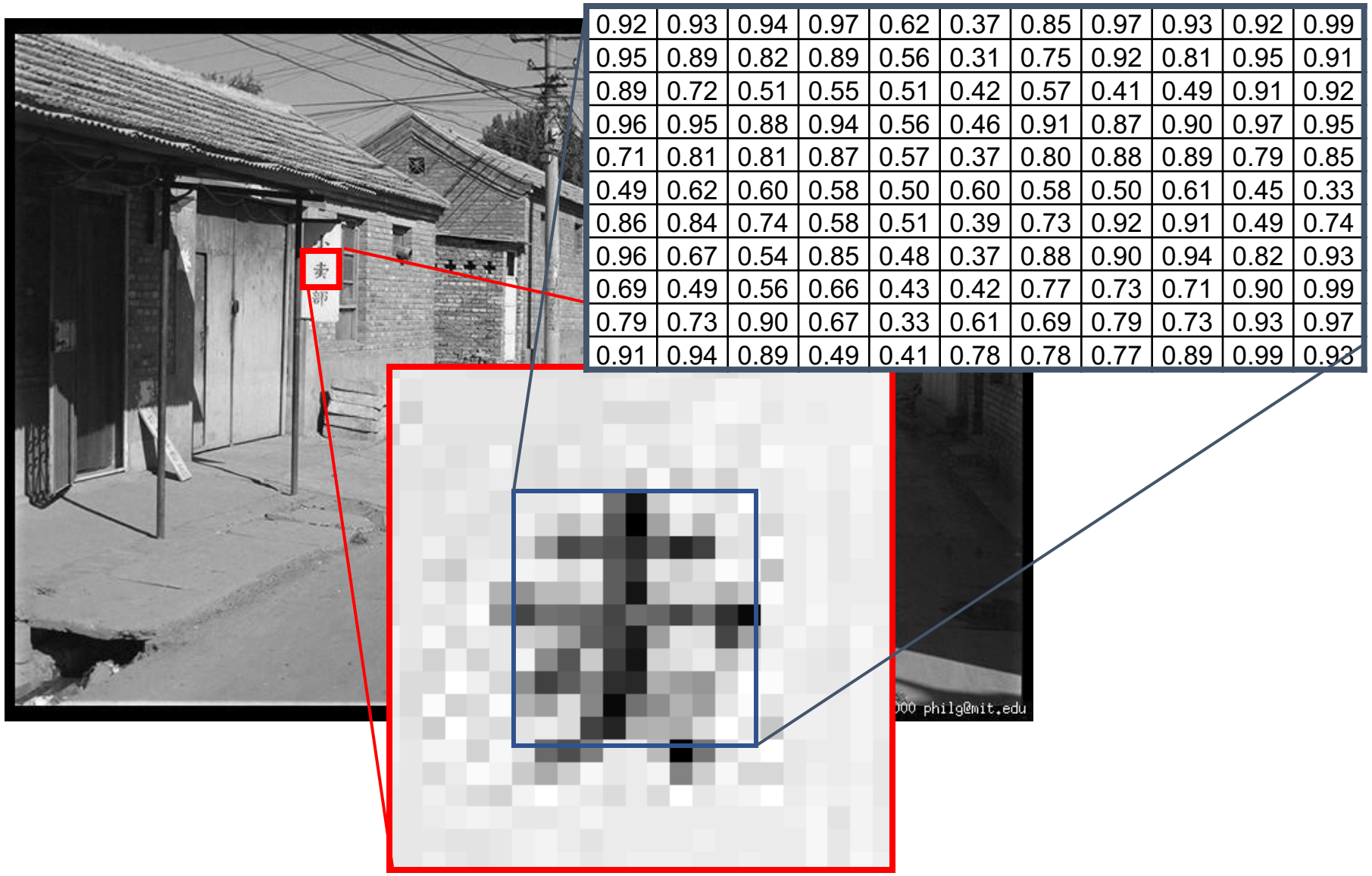
- Everything in Chapter1, Chapter-2 and Chapter-3.



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

0-1 or 0(black)-255(white) (8-bit)

$$I = f(x, y) : \mathrm{R}^2 \rightarrow \mathrm{R}$$ (Gray-scale/intensity image)

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

2D coordinates on images

RGB values

Recall: Effect of Spatial Resolution

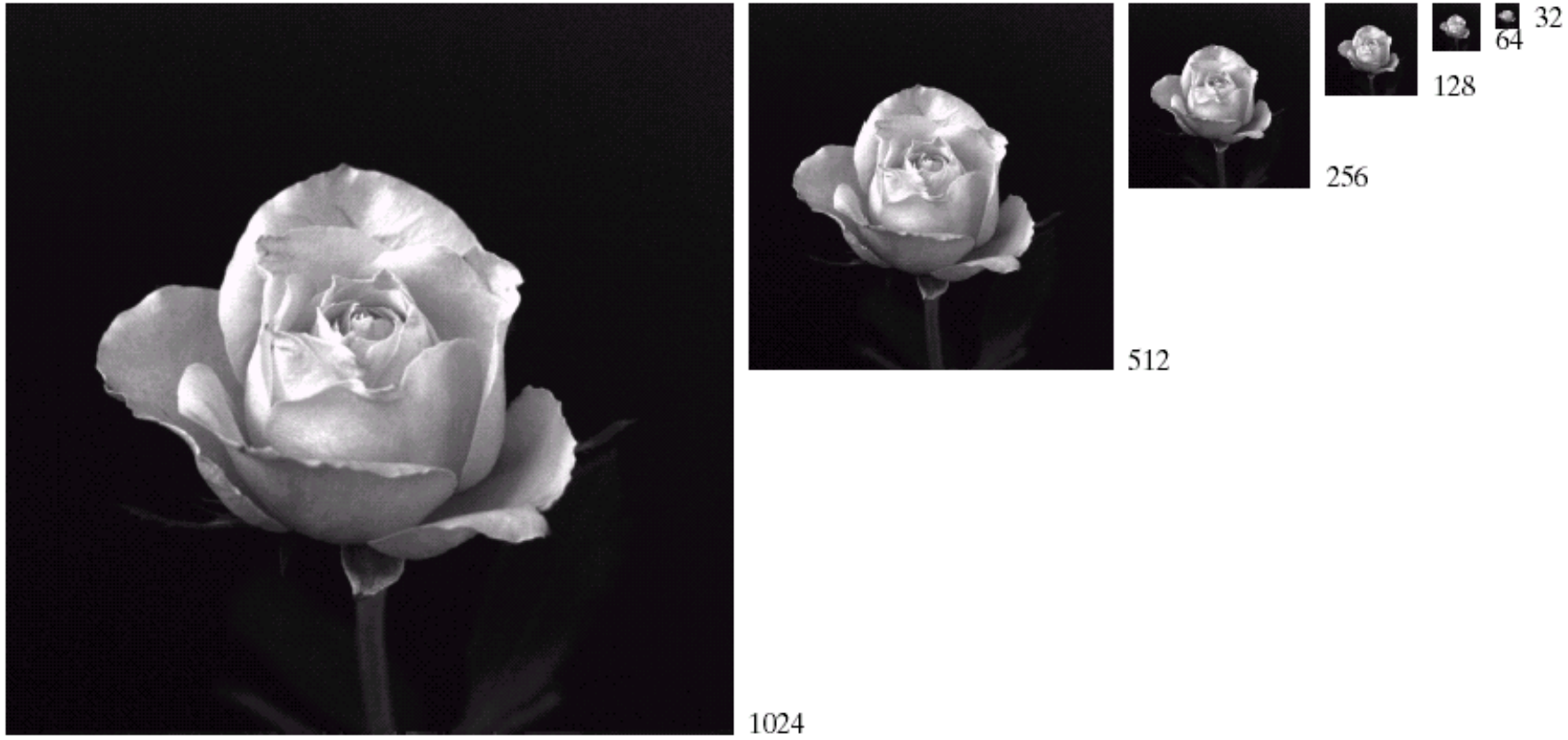
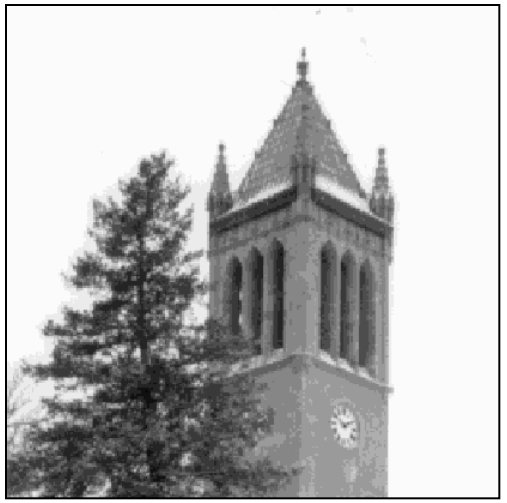


FIGURE 2.19 A 1024 × 1024, 8-bit image subsampled down to size 32 × 32 pixels. The number of allowable gray levels was kept at 256.

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

Recall: Effect of Quantization Levels (cont.)

16 levels



8 levels

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

4 levels

2 levels

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 the corresponding input **pixel (intensity, color) value**.

Fundamentals of Digital Images

$$I = f(x, y) : \mathrm{R}^2 \rightarrow \mathrm{R} \quad \text{(Gray scale image)}$$

$$I = f(x, y) : \mathrm{R}^2 \rightarrow \mathrm{R}^3 \quad \text{(Color image)}$$

Point Operations

- Operation on Pixel's value.
  No neighbours
- Context free (memory-less).
- Operation can be performed on the *image histogram*.
- Example:

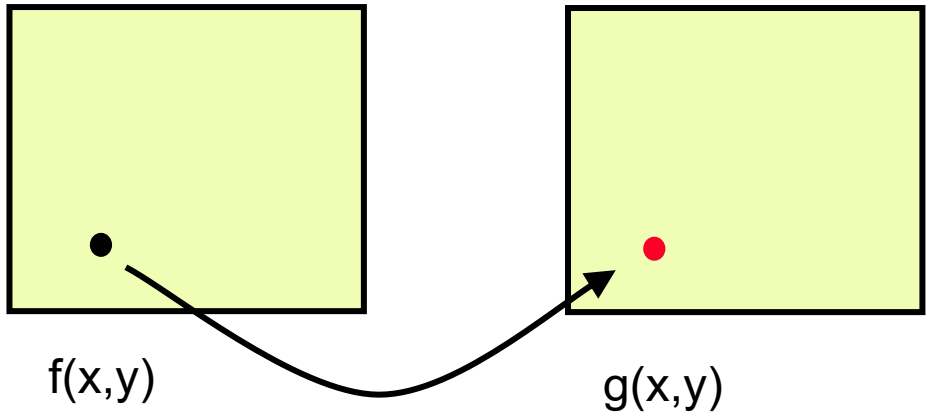
target source

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

α is a constant, controlling the contrast.

β is a constant, controlling the brightness.

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



Point Operations

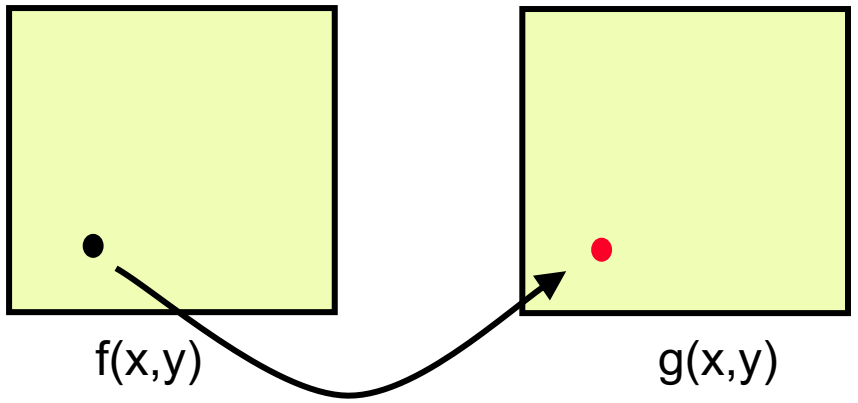
- Operation on Pixel's value.
- Context free (memory-less).
- Operation can be performed on the *image histogram*.
- Example:

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

α is a spatially varying.

β iss a spatially varying.

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



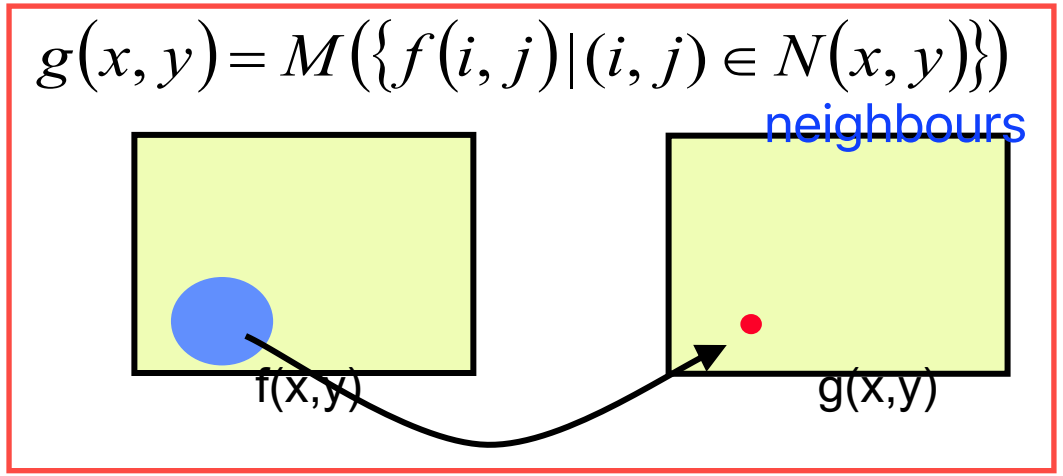
Neighborhood Operations

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

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

salt and peper noise

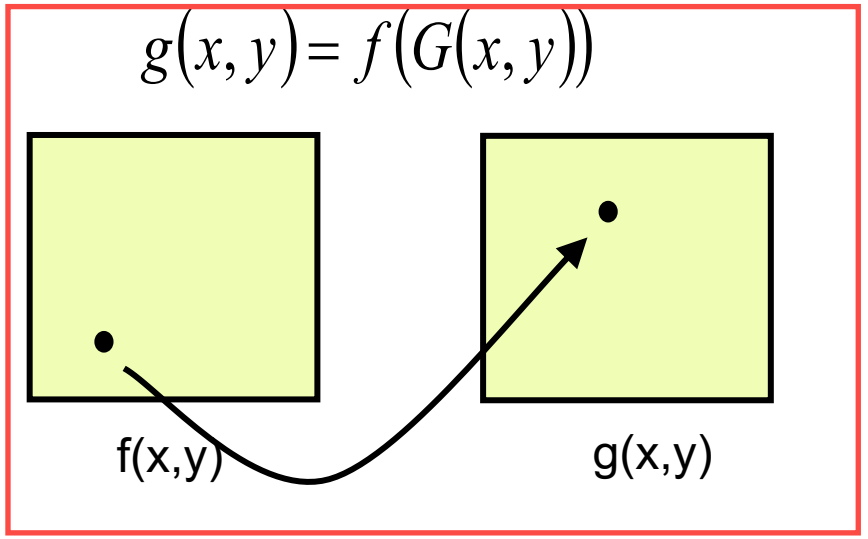
$$g(x,y) = M\left(\{f(i,j) \mid (i,j) \in N(x,y)\}\right)$$



Geometric Operations

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

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



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 $r_0, r_1, \cdots, r_{L-1}$

is the discrete function

e.g. grayscale values 0-255, where L=256

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

percentage of a gray value out of all pixels

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.

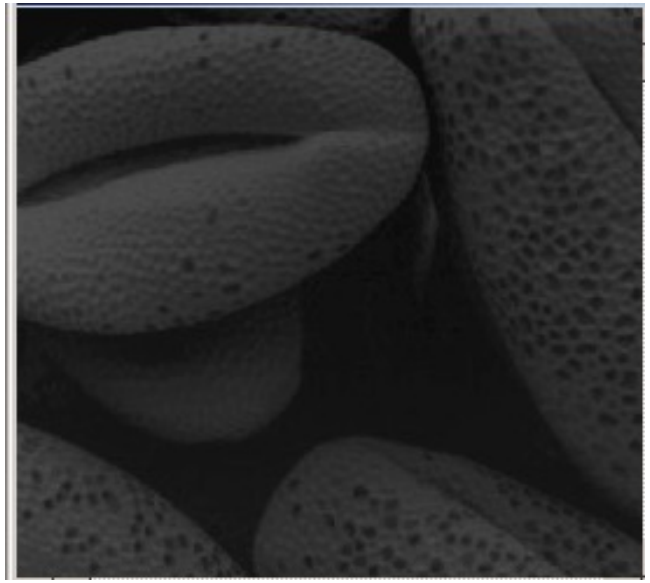
- Histogram provides a global description of the appearance of an image.

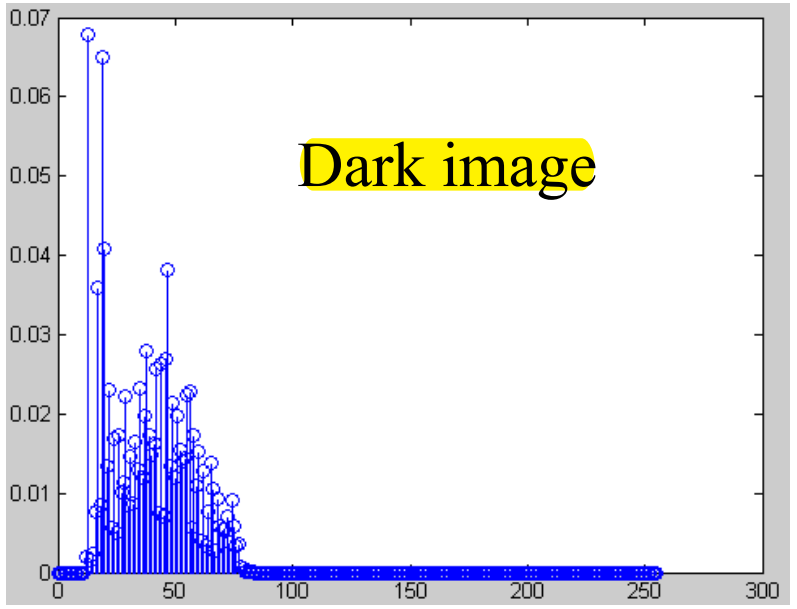
- If we consider the gray values in the image as realizations of a random variable R, with some probability density, histogram provides an **approximation** to this probability density.

- In other words, $\Pr(R = r_k) \approx p(r_k)$

Some Typical Histograms

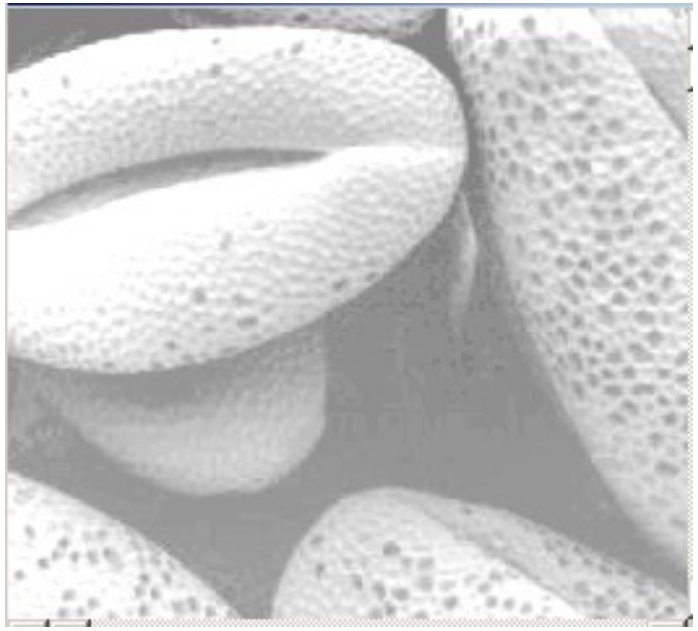
- The shape of a histogram provides useful information for image global appearance.

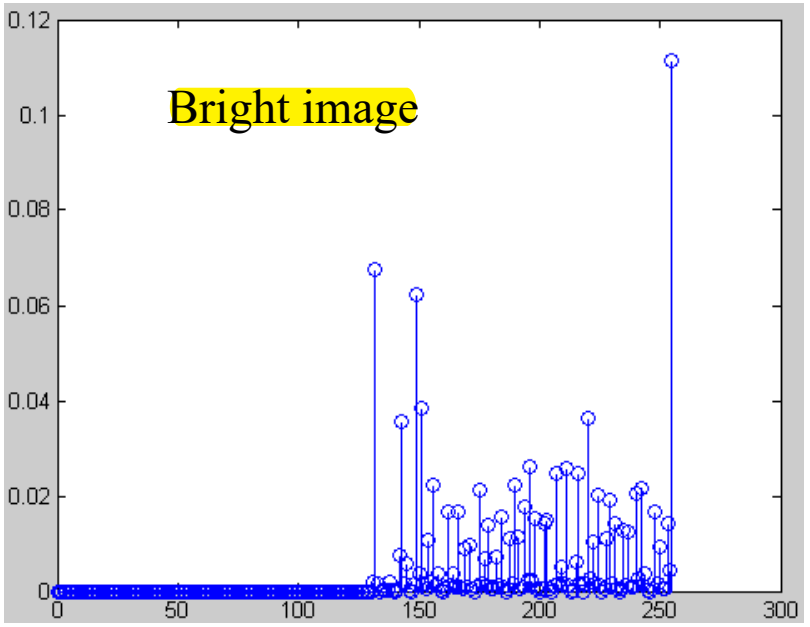


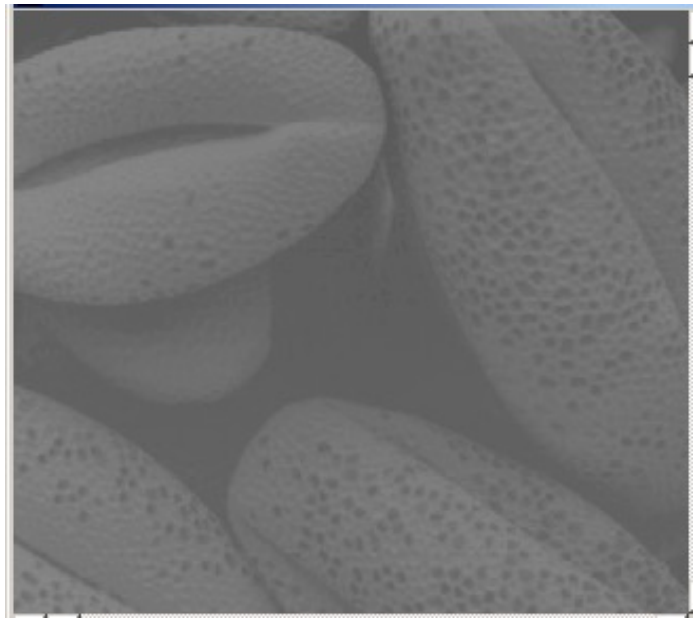


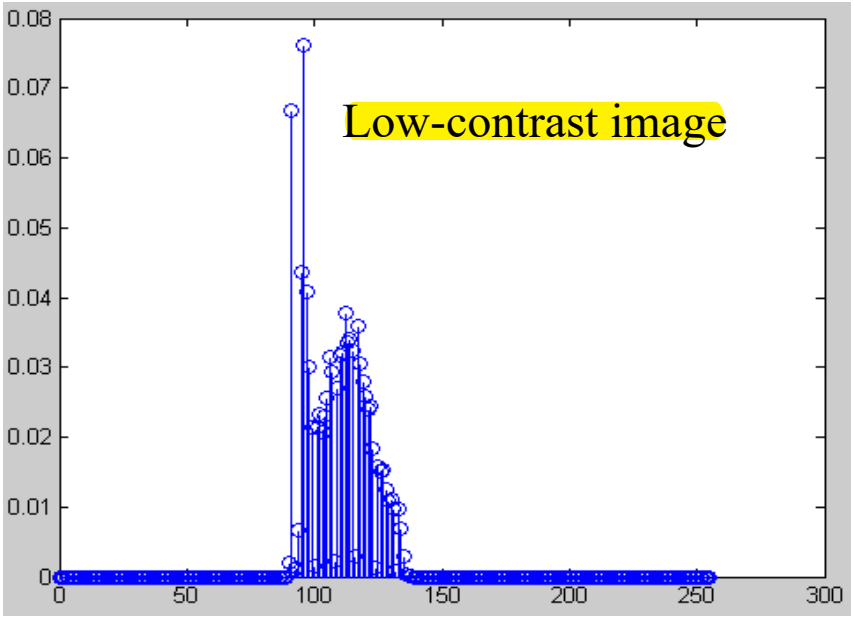
0: black

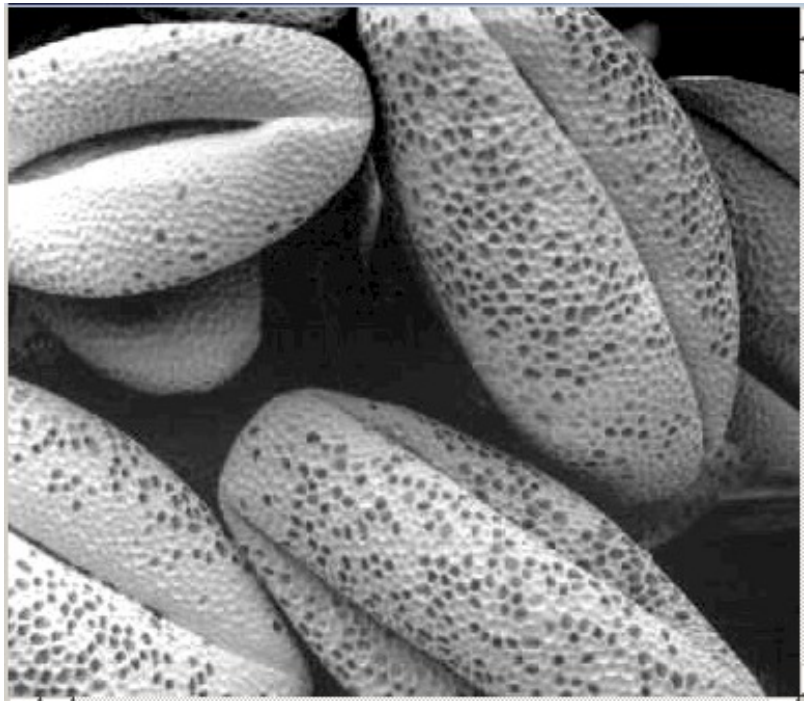
255: white

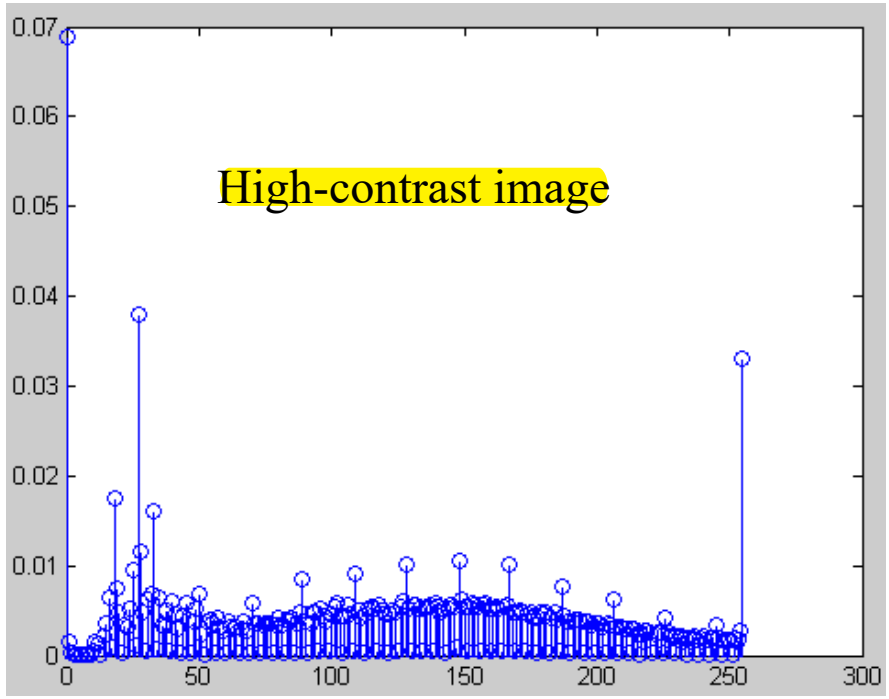



Bright image
0.12
0.1
0.08
0.06
0.04
0.02
0
0
50
100
150
200
250
300



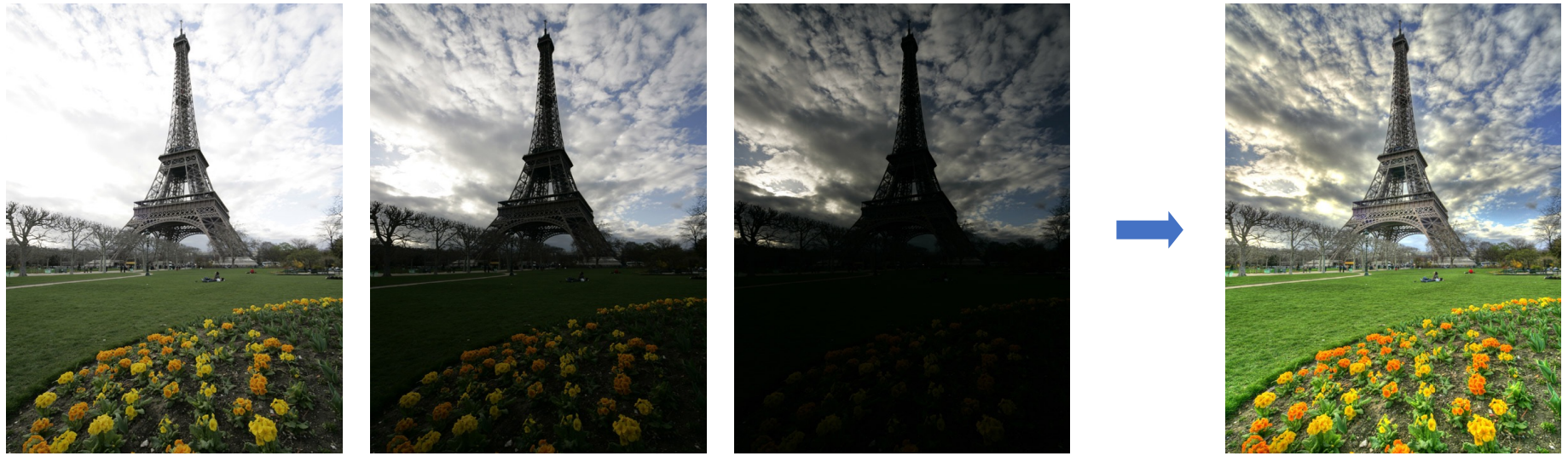

Low-contrast image
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
50
100
150
200
250
300





a broad distribution of pixel intensities, with peaks at the extremes representing the darkest and brightest regions.

Application: image enhancement



Histogram modification

- Given a point operation:

after before

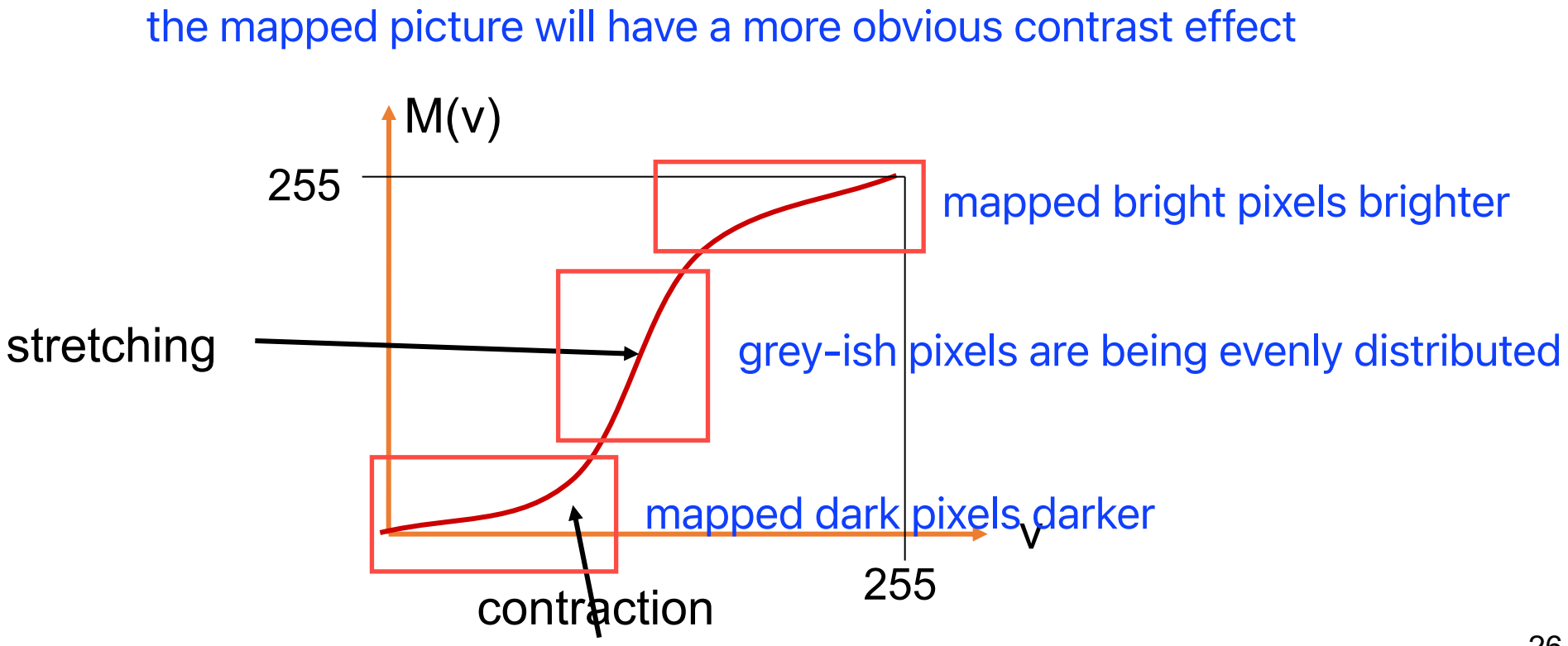
$$v_b = M(v_a)$$

- $M(v_a)$ 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.

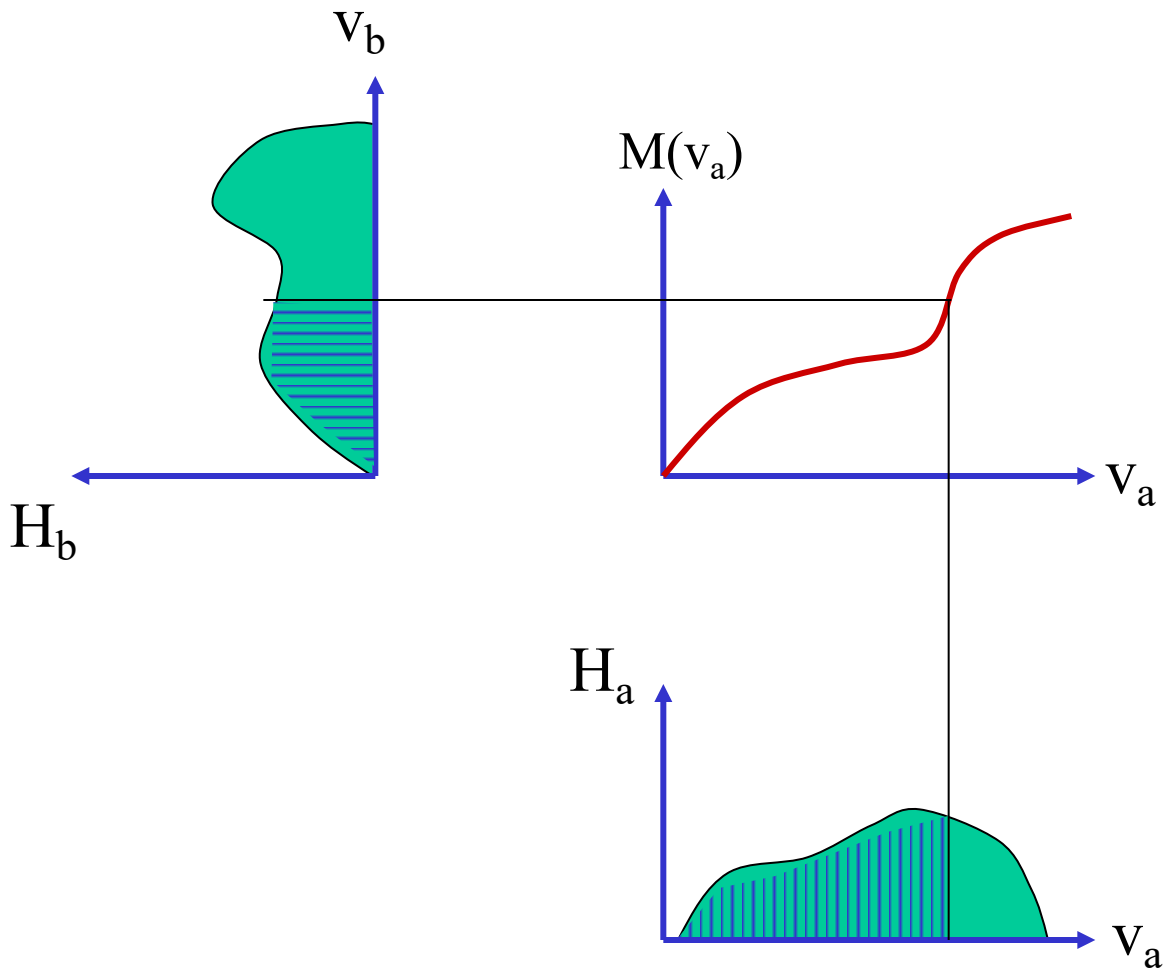
Monotonicity: preserve the order of pixel intensities, prevent informaiton loss or distortion

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:

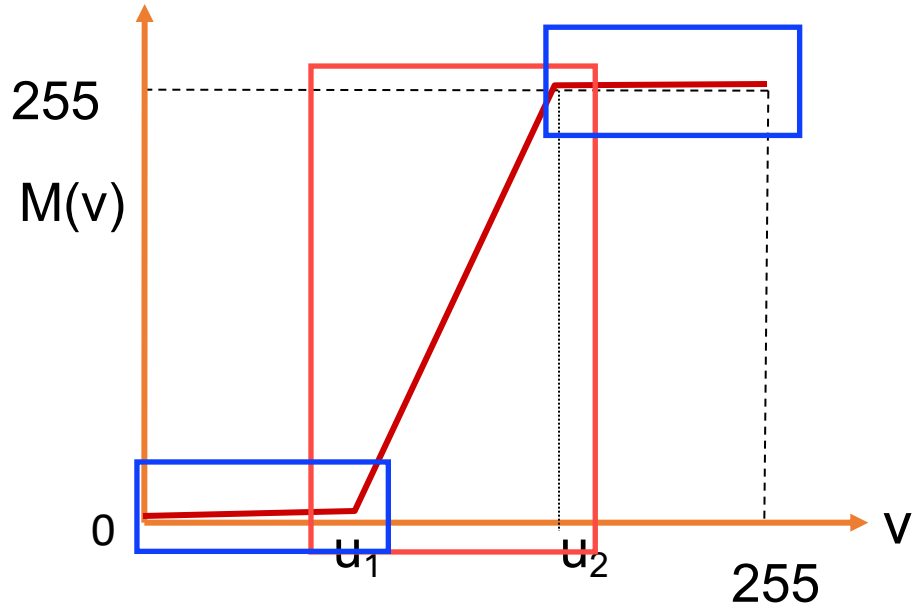


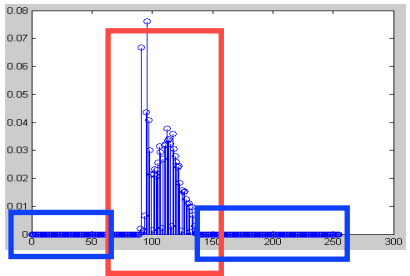
- The area under H_a between 0 and v_a is equal to the area under H_b between 0 and $v_b=M(v_a)$

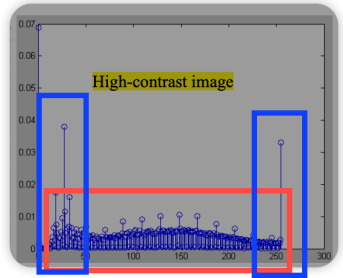


Contrast Enhancement

- 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

$$s = cr^{\gamma}$$

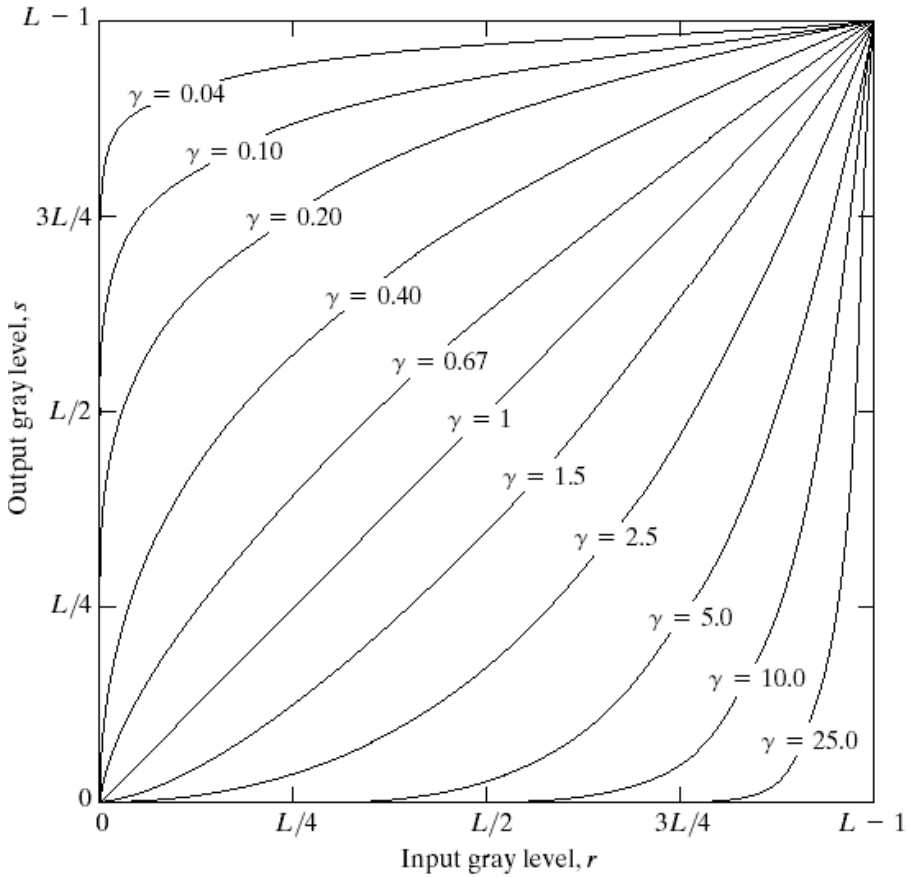
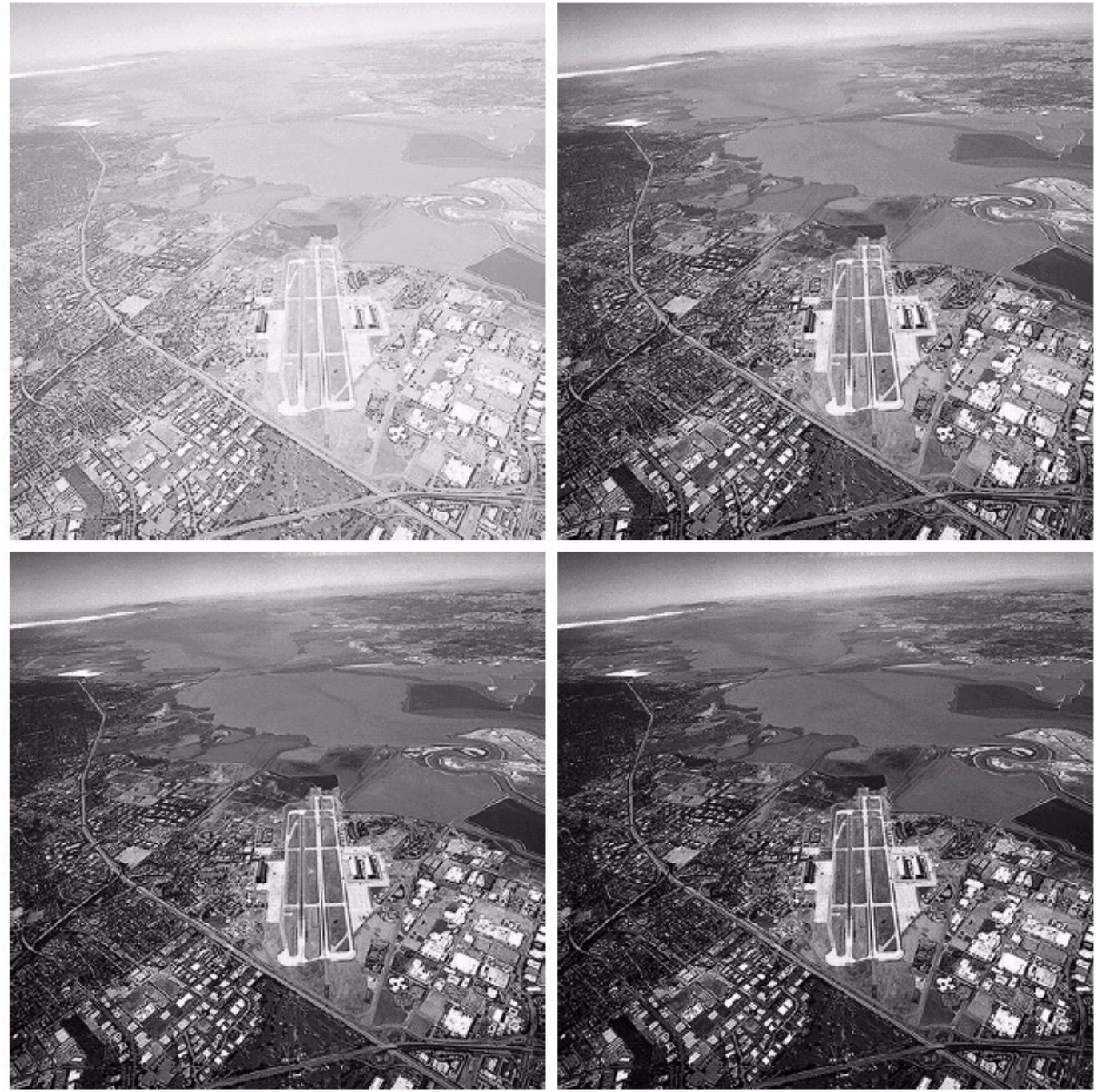


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

a b
c d

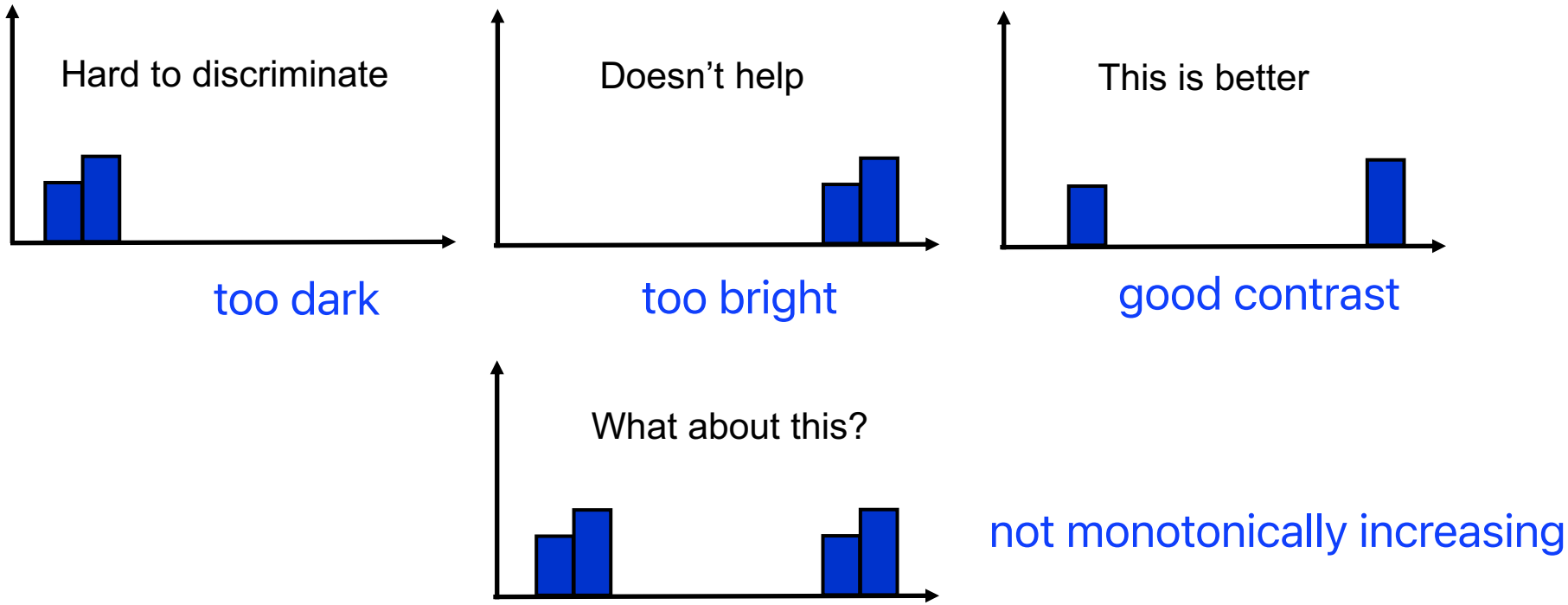
FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0, respectively. (Original image for this example courtesy of NASA.)



Histogram Equalization

- Visual contrast between objects depends on their gray-level separation.

- How can we improve the contrast **after** an image has been acquired?

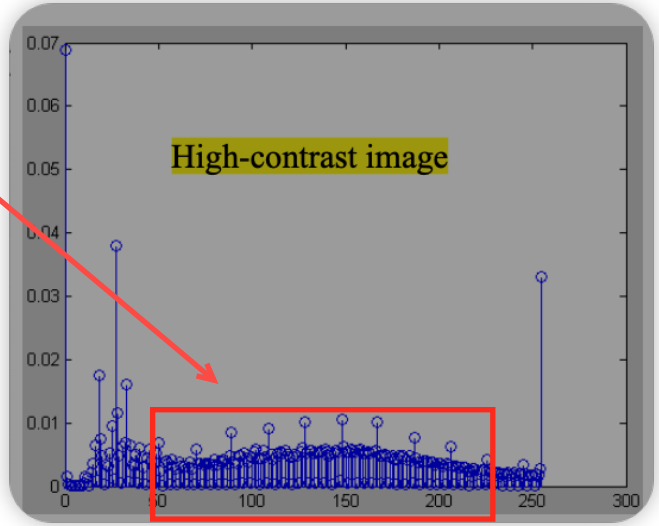


Histogram Equalization

- 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.

  maintain the continuity of pixel intensity transitions after histogram modification



Algorithm: How to implement histogram equalization?

compute the gray value percentage for each pixel

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

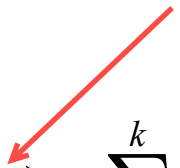
$$p_{in}(r_k) = \frac{n_k}{n} \qquad 0 \le r_k \le 1 \qquad 0 \le k \le L-1$$

L: Total number of gray levels

n_k: Number of pixels with gray value r_k

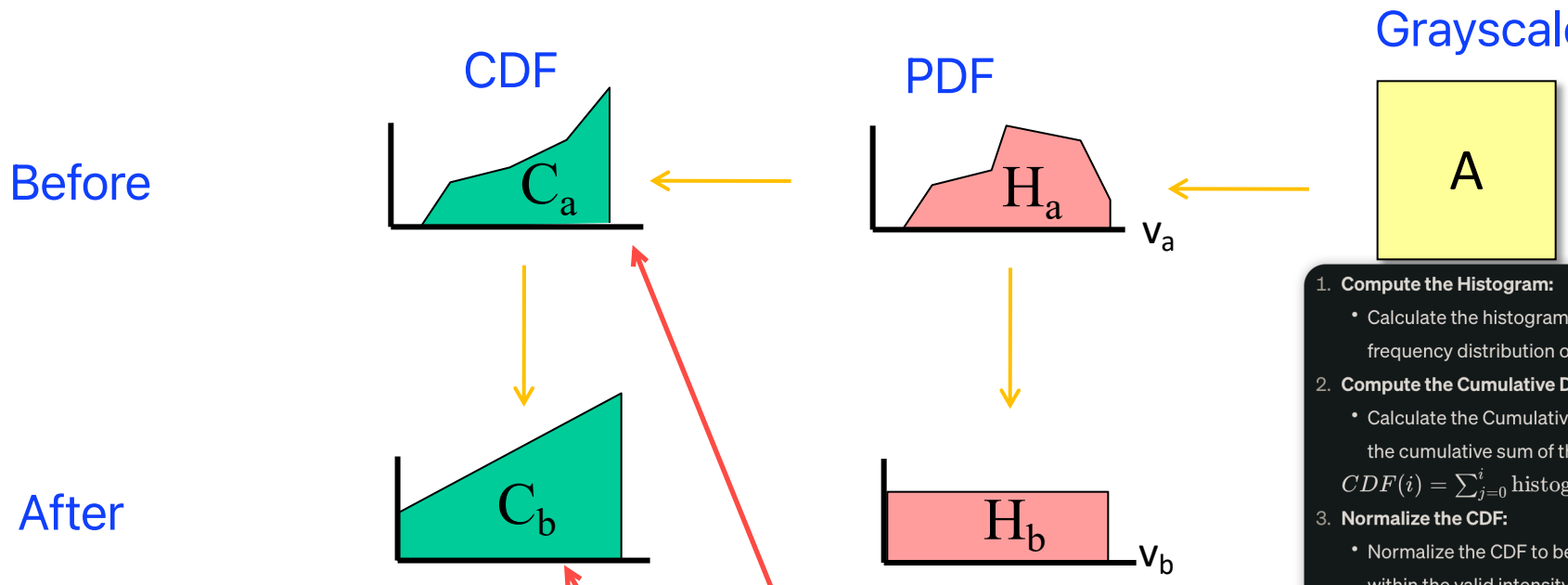
n: Total number of pixels in the image

Step 2: Based on CDF, compute the transformation :



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

Histogram Equalization: Algorithm



1. **Compute the Histogram:**
   - Calculate the histogram of the input grayscale image. The histogram represents the frequency distribution of pixel intensities.
2. **Compute the Cumulative Distribution Function (CDF):**
   - Calculate the Cumulative Distribution Function (CDF) from the histogram. The CDF represents the cumulative sum of the histogram values and ranges from 0 to the total number of pixels.

   $CDF(i) = \sum_{j=0}^{i} \text{histogram}(j)$
3. **Normalize the CDF:**
   - Normalize the CDF to be in the range [0, 1]. This step ensures that the mapping values are within the valid intensity range.

   $CDF_{\text{normalized}}(i) = \frac{CDF(i) - CDF_{\min}}{\text{Total number of pixels} - 1}$
4. **Map Original Intensities to New Intensities:**
   - Map the original pixel intensities to their corresponding equalized intensities using the normalized CDF.

   $\text{Equalized Intensity}(i) = \text{round}\left(\text{CDF}_{\text{normalized}}(i) \times (\text{Number of intensity levels} - 1)\right)$
5. **Create the Equalized Image:**
   - Replace each pixel in the original image with its corresponding equalized intensity to create the histogram-equalized image.

- Define: CDF of a uniform dist, L=256

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

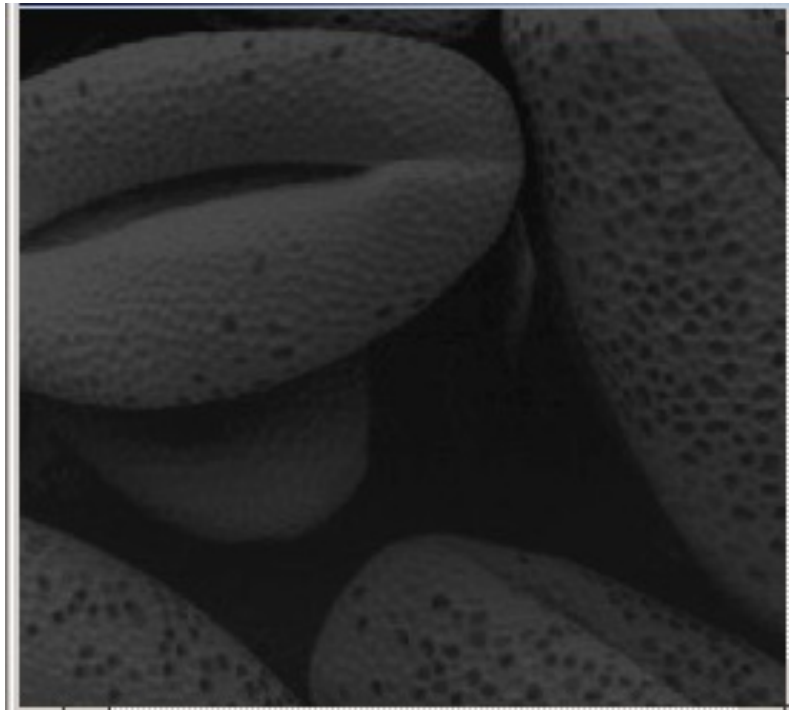
- Assign:

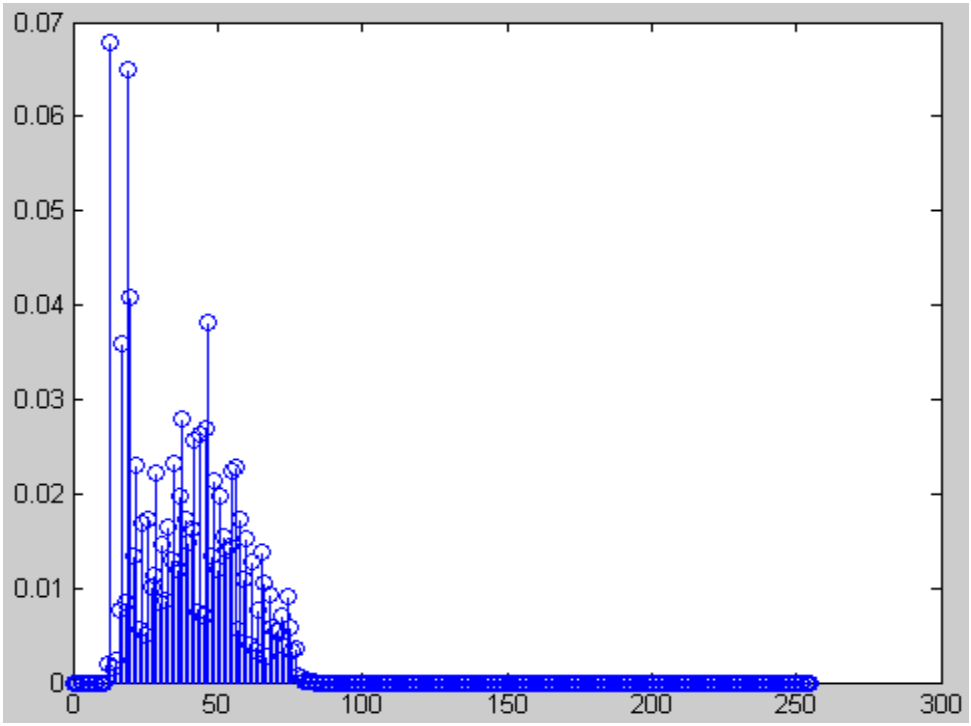
$$v_b = C_b^{-1}(C_a(v_a)) = M(v_a)$$

new gray value — old gray value

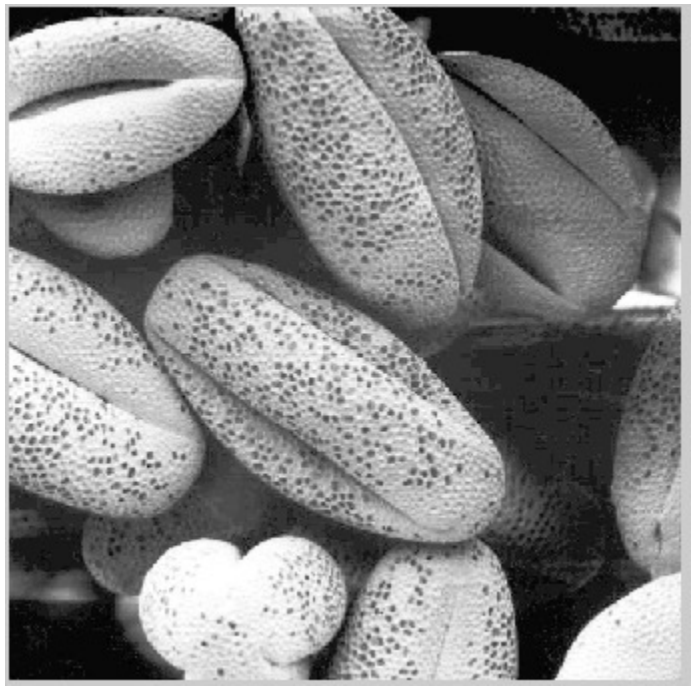
Example

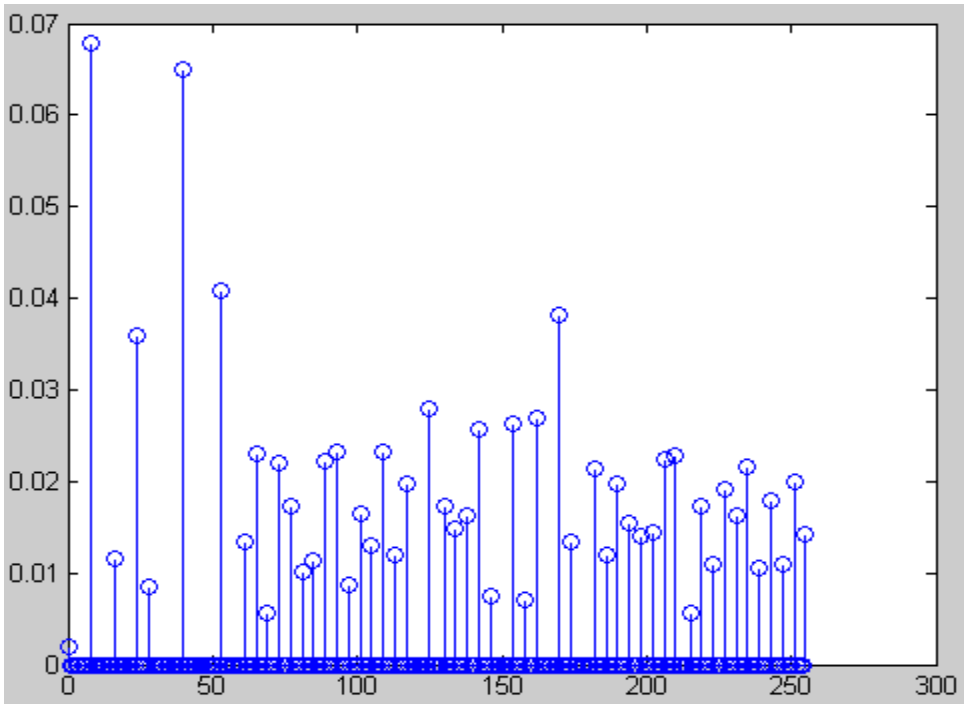
Original image and its histogram



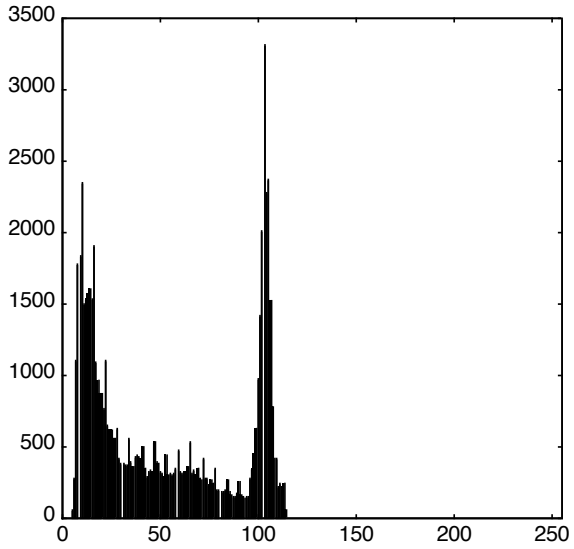


Histogram equalized image and its histogram

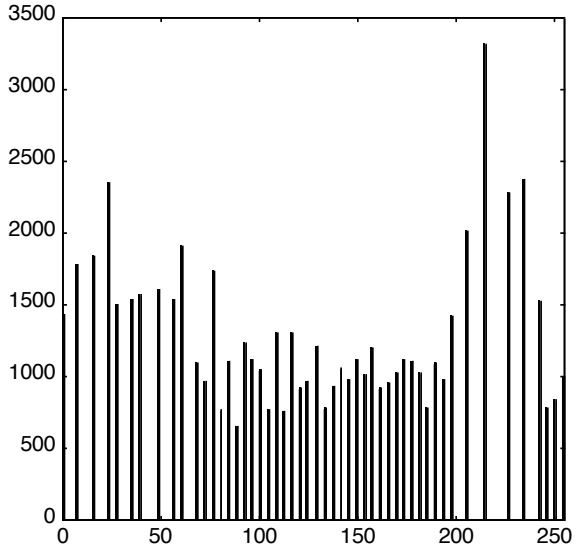




Hist. Eq.: Example 1

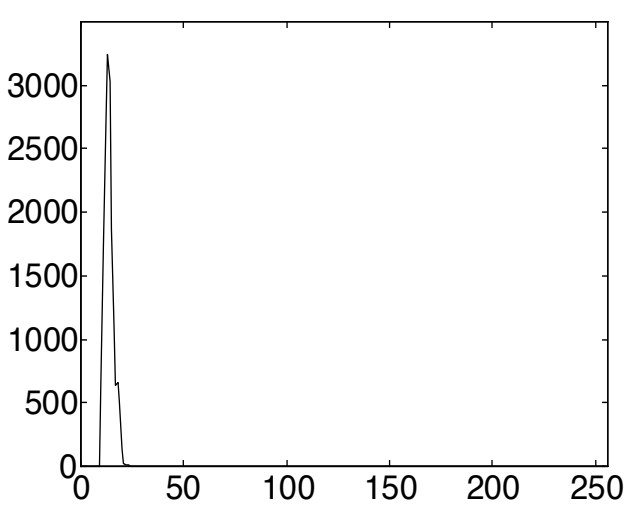


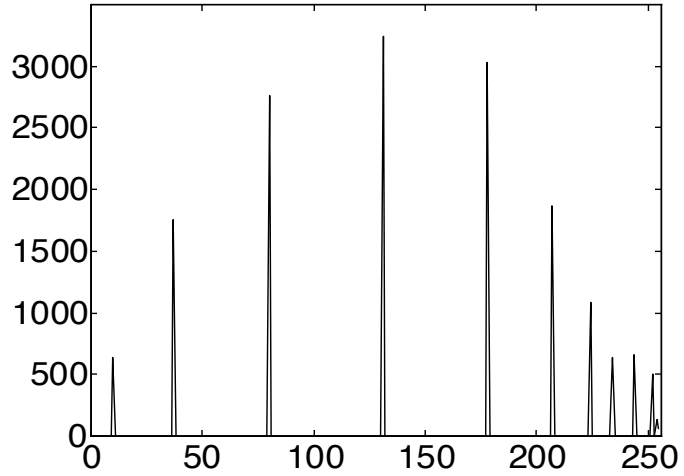
Original



Equalized

Hist. Eq.: Example 2



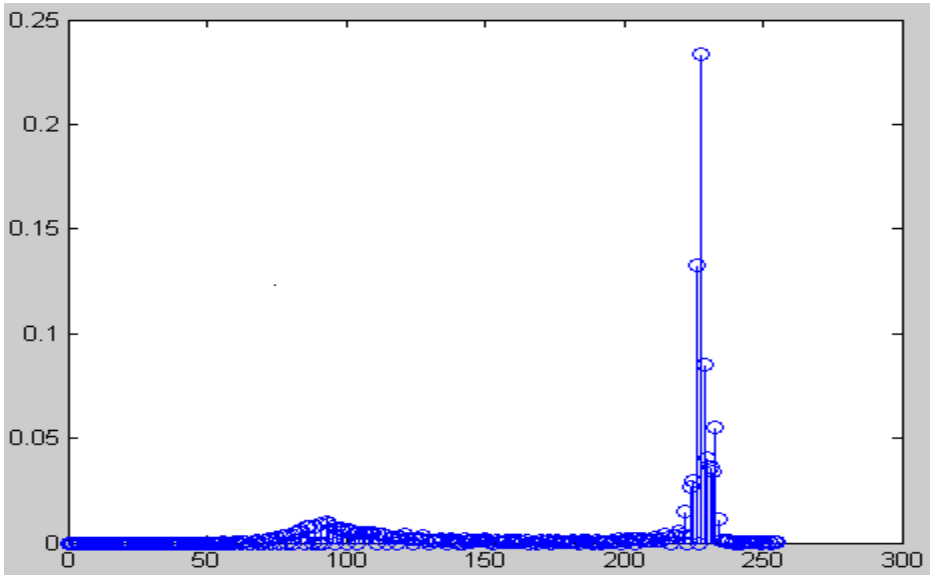


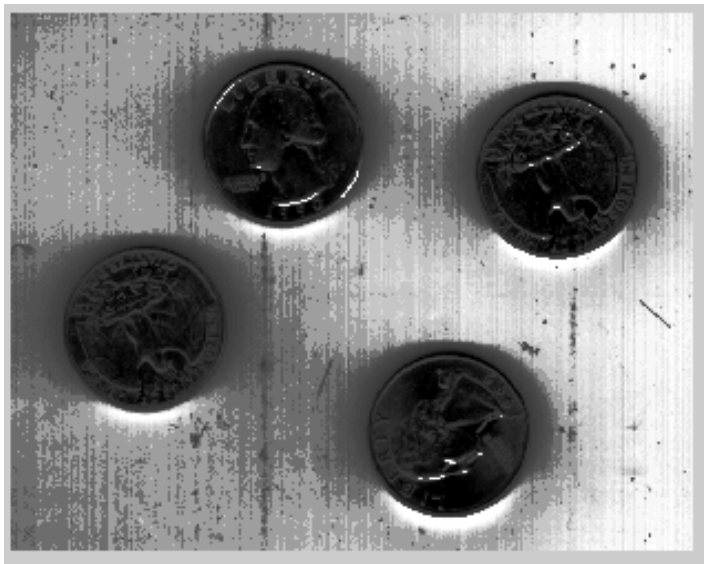
Original

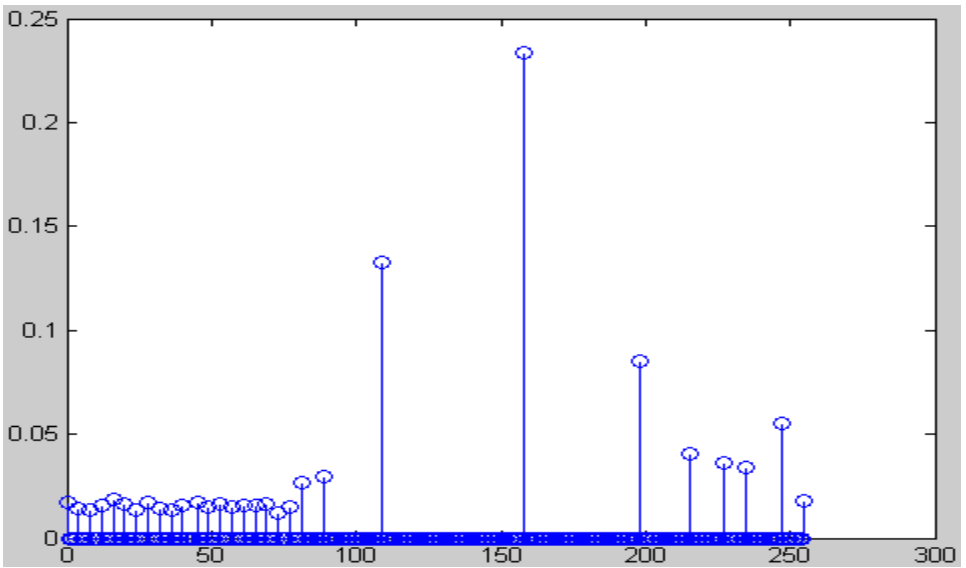
Equalized

- 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.”






0.25
0.2
0.15
0.1
0.05
0
0
50
100
150
200
250
300

Geometric Transformations

(spatial transformation, image warping)

Image Warping

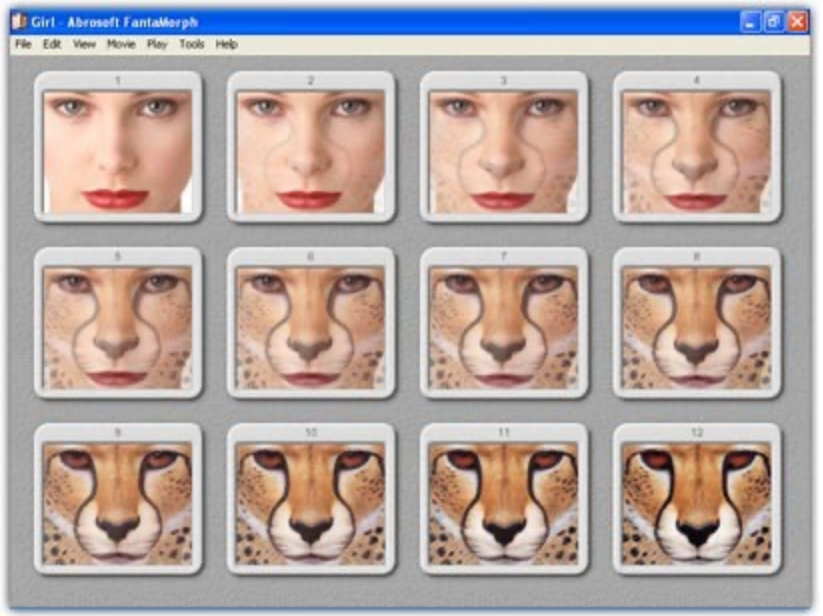
http://www.jeffrey-martin.com

Some slides from Steve Seitz

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

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

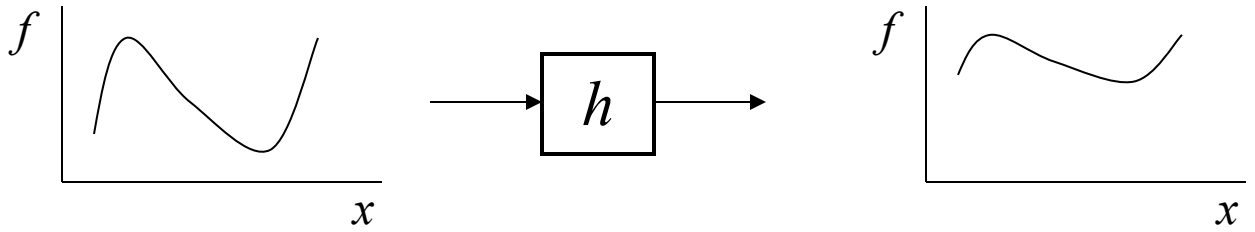


Image warping: change the spatial ***domain*** of image

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

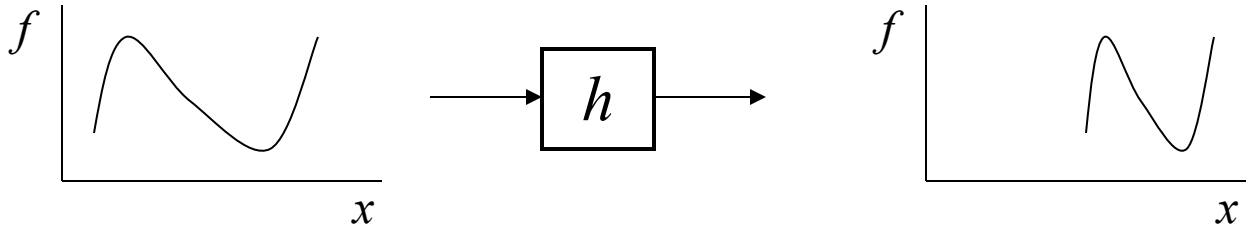


Image Warping

Image filtering: change ***range*** of image

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

a local operation that modifies pixel valus based on predefined filters

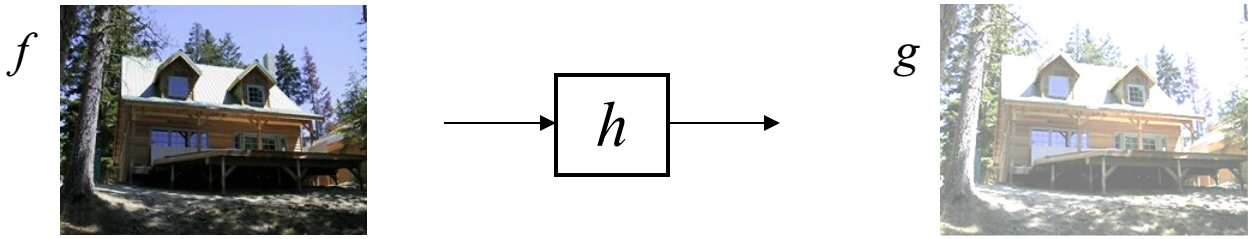
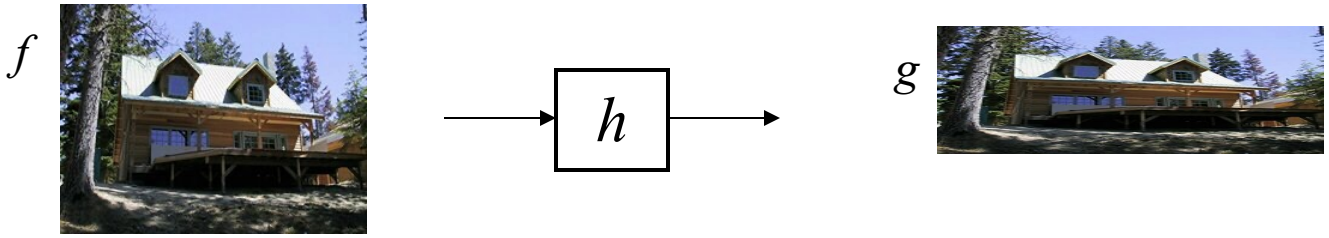


Image warping: change ***domain*** of image

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



a global operation that spatially transforms that arrangement of pixels in an image

Types of Image Warping

Simple geometric transformations:

- Rotation;
- Similarity
- Affine mapping
- Projective mapping

Other general types of transformations

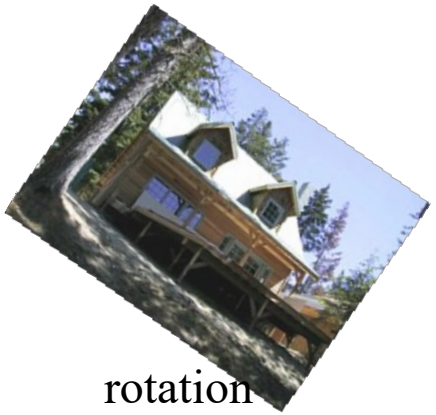
free-form deformation

Geometric transformation

Examples of transformation

image domain changes after transformation

translation



rotation

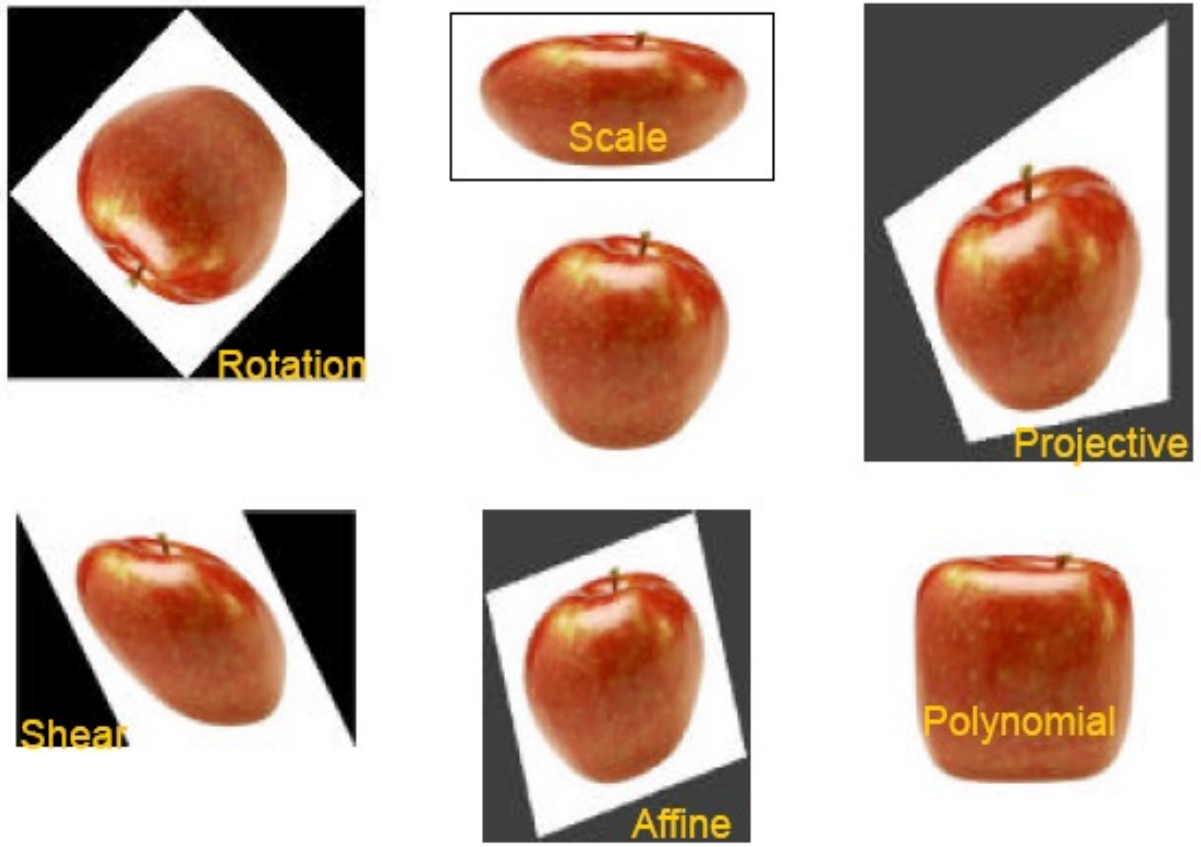
aspect

affine

perspective

cylindrical

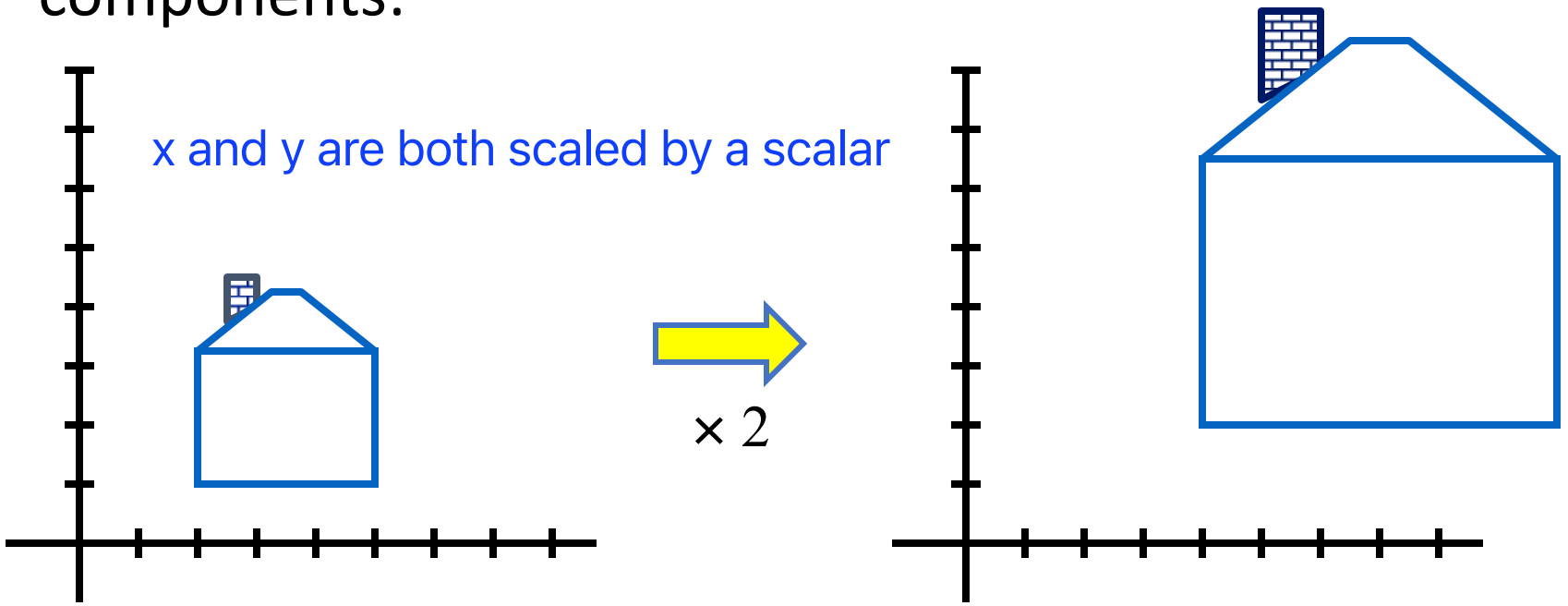
Types of Transformation



MATLAB functions: griddata, interp2, maketform, imtransform

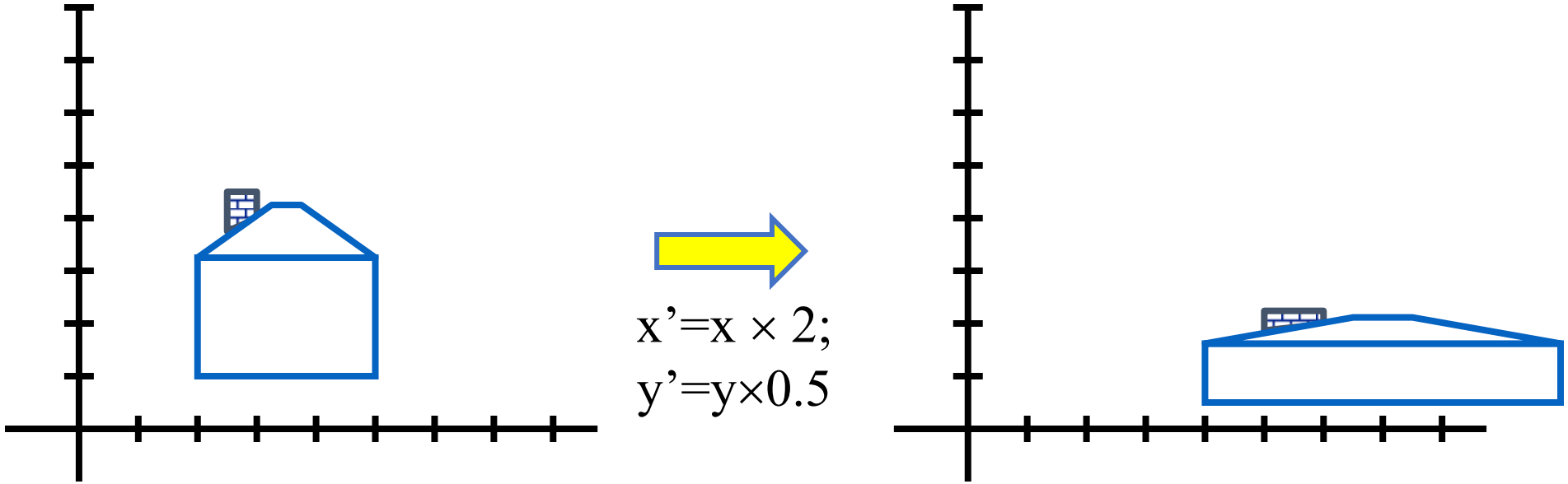
Scaling

- *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:



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?

Full rank = Invertable = non-zero determinant

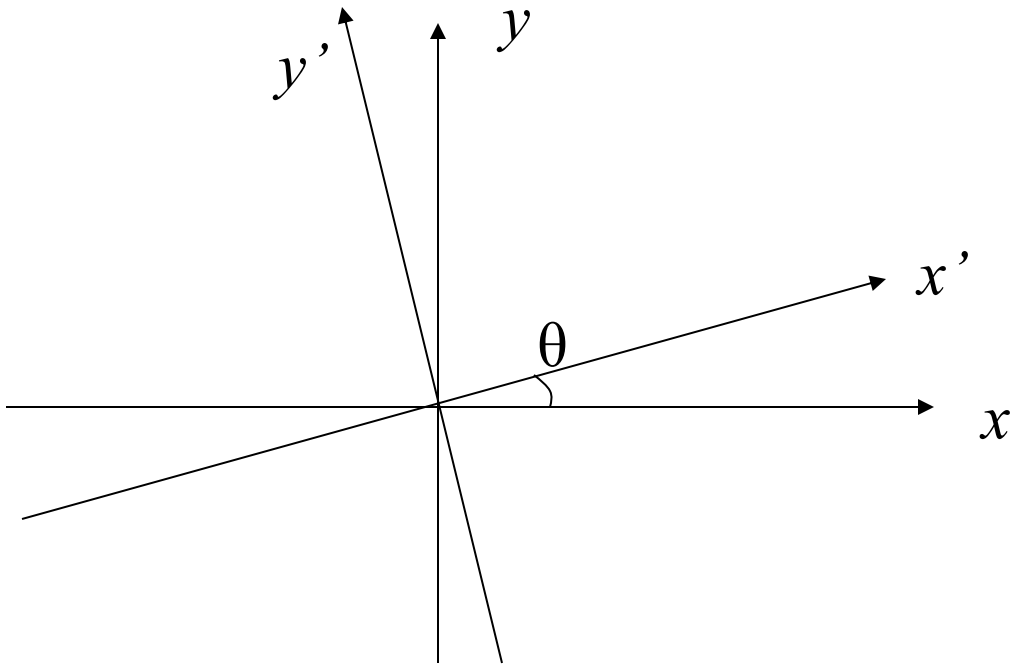
Scaling

a=2

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

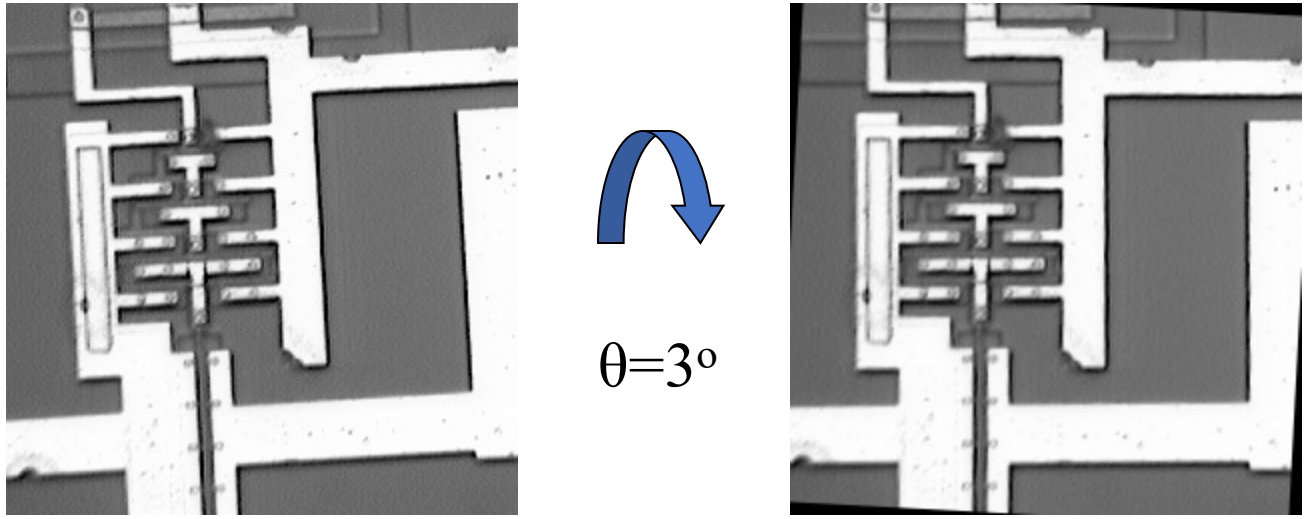
55

Rotation

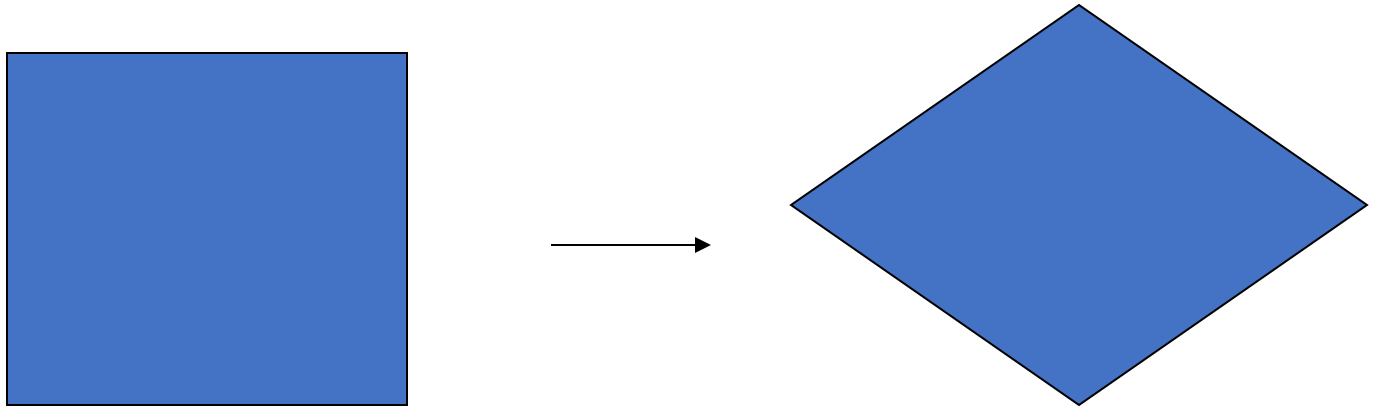


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation Example



Affine Transform



square

parallelogram

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

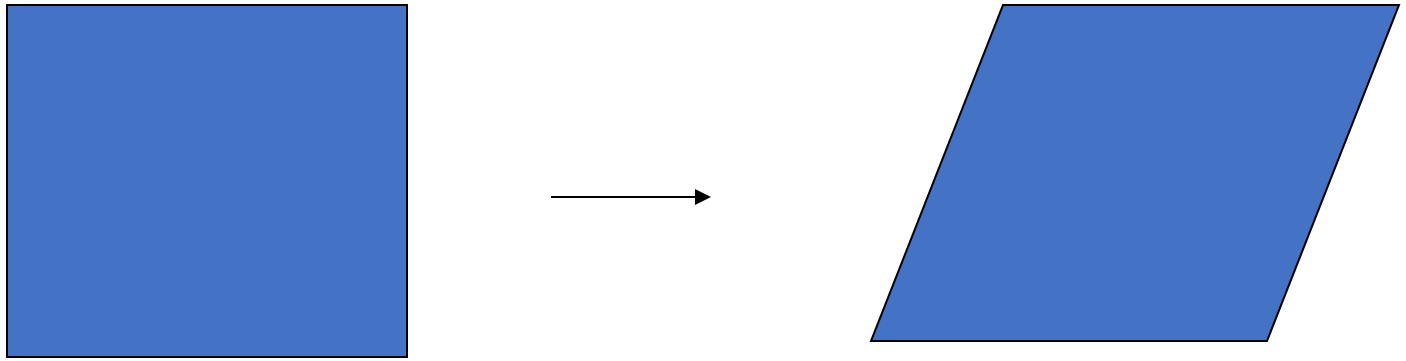
58

Affine Transform Example

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

59

Shear



square

parallelogram

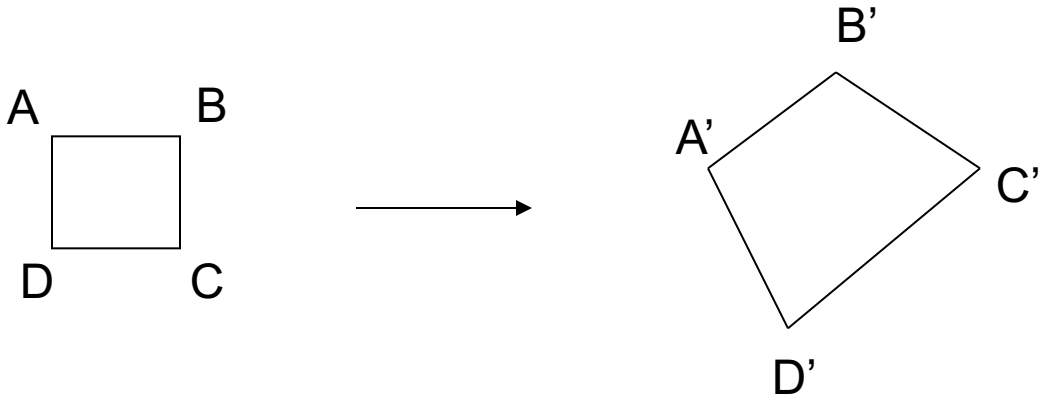
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Shear Example

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

61

Projective Transform



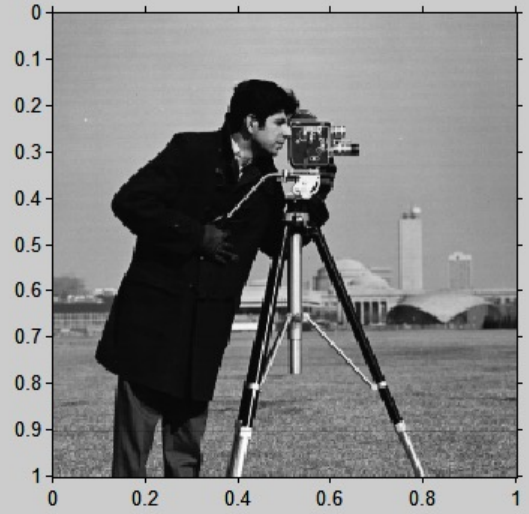
square quadrilateral

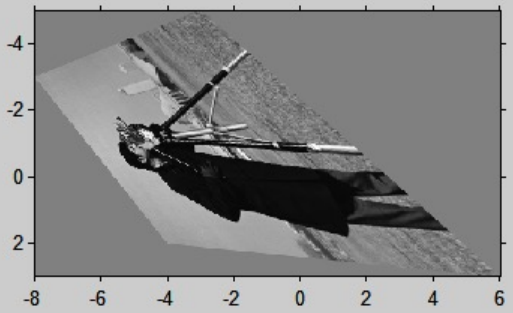
$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

62

Projective Transform Example

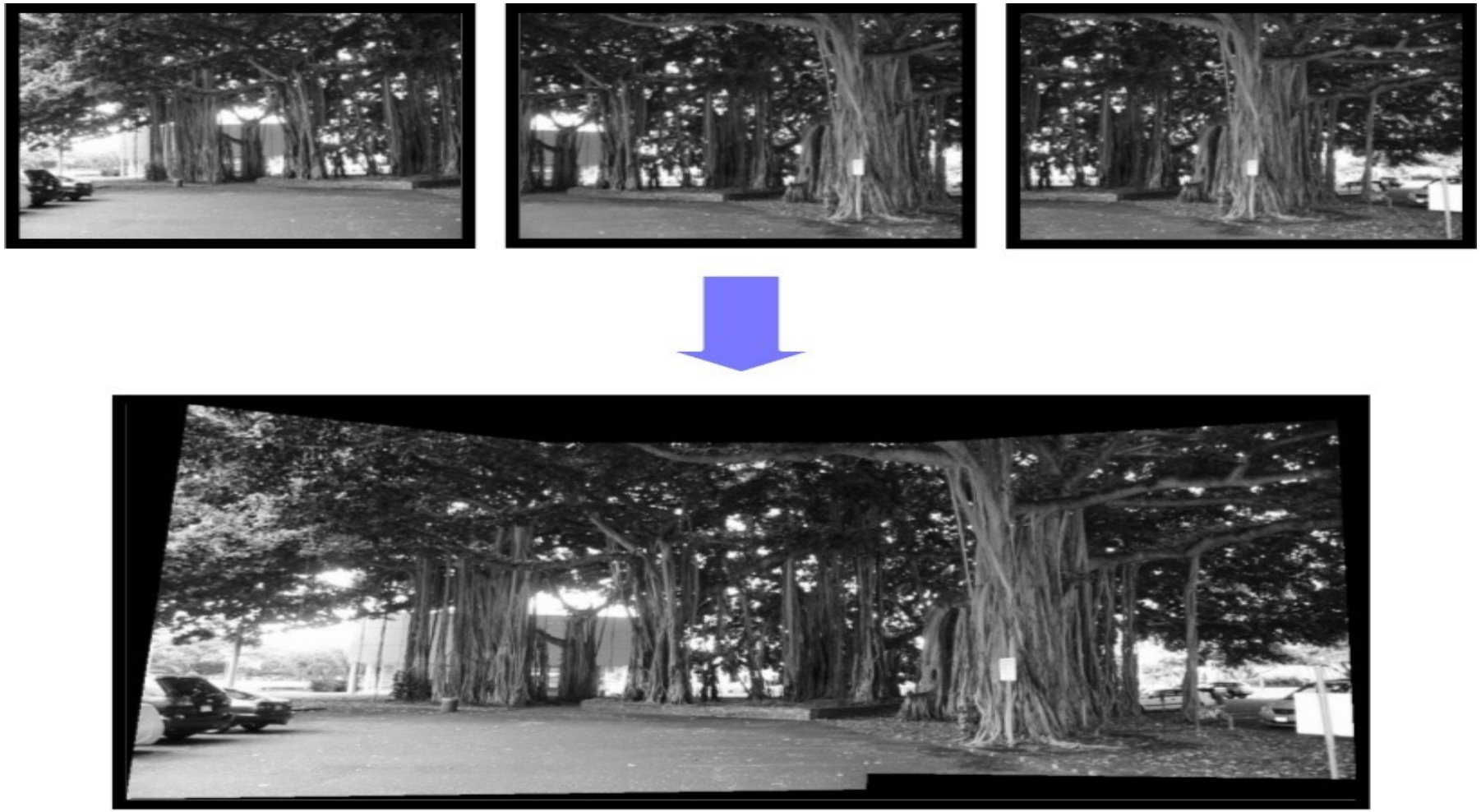




[0 0; 1 0; 1 1; 0 1] ⟶ [-4 2; -8 -3; -3 -5; 6 3]

63

Projective Image stitching



64

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

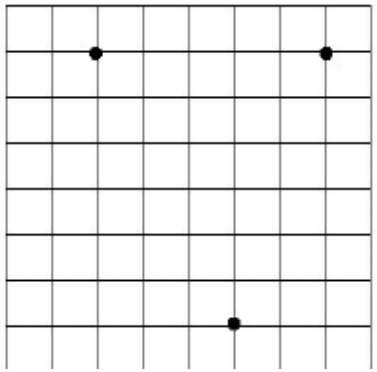
Target Image: this image is obtained by transforming the reference image.

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

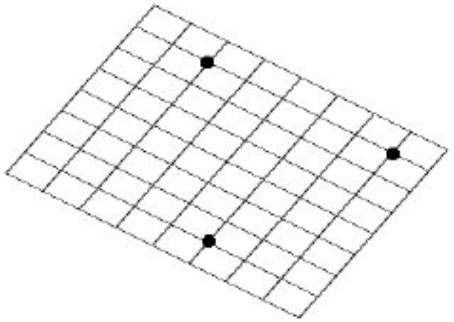
(x’,y’): (or [u,v]) coordinates of points in the target image

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.

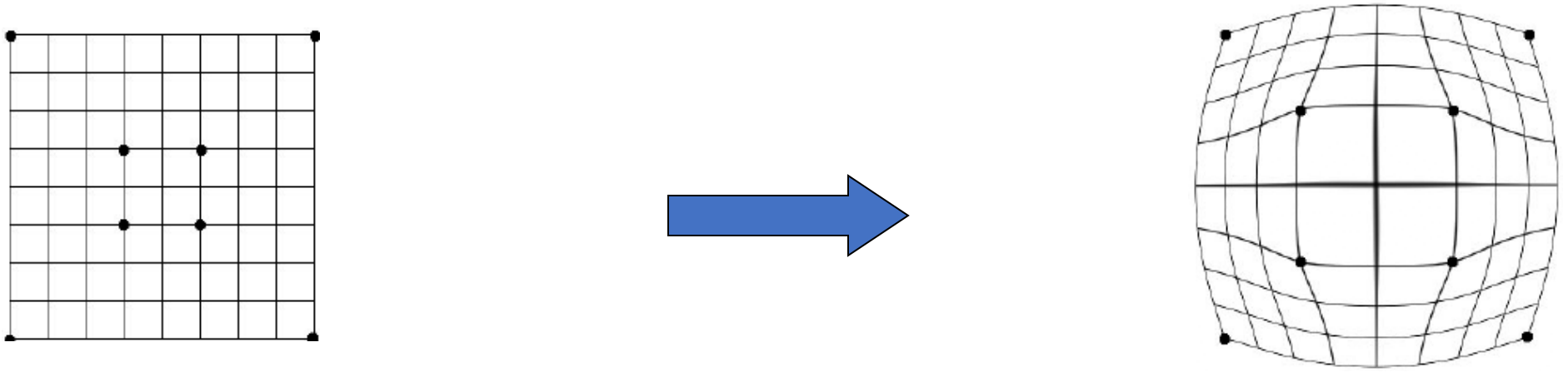


Source Image



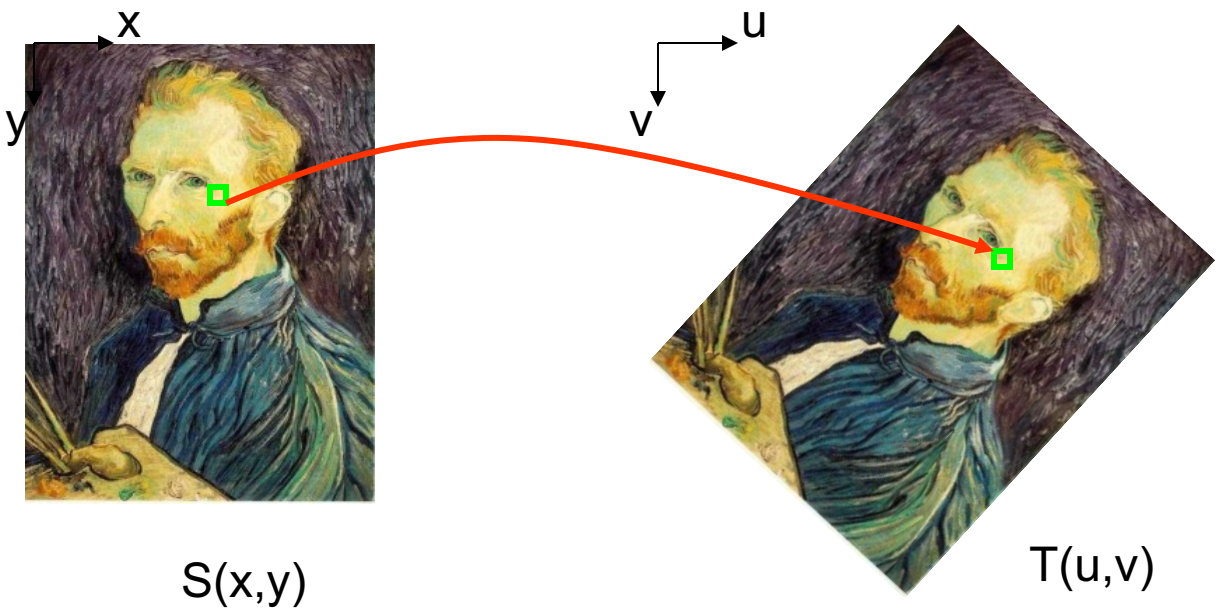
Target Image

Transformation Function



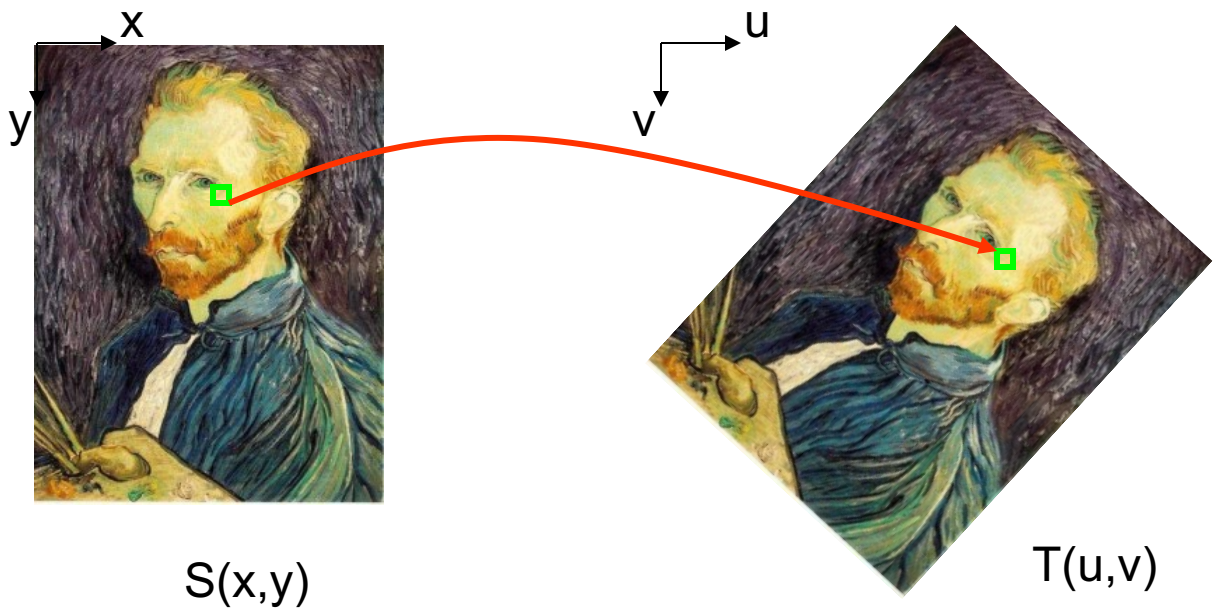
Transform the domain of an image to a desired geometry

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

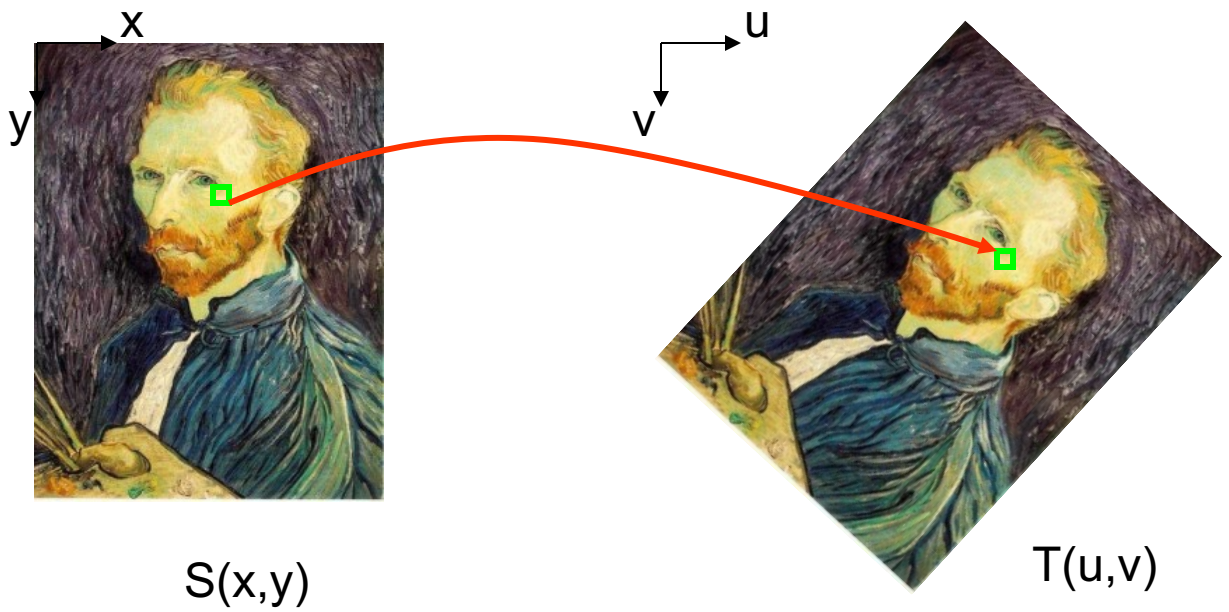


Forward warping algorithm:

```
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}

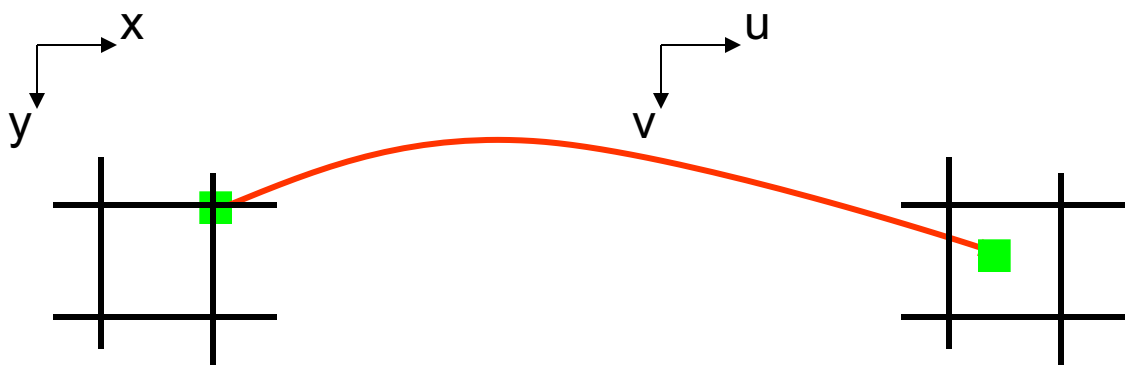
  for $x = x_{min}$ to x_{max}

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

    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

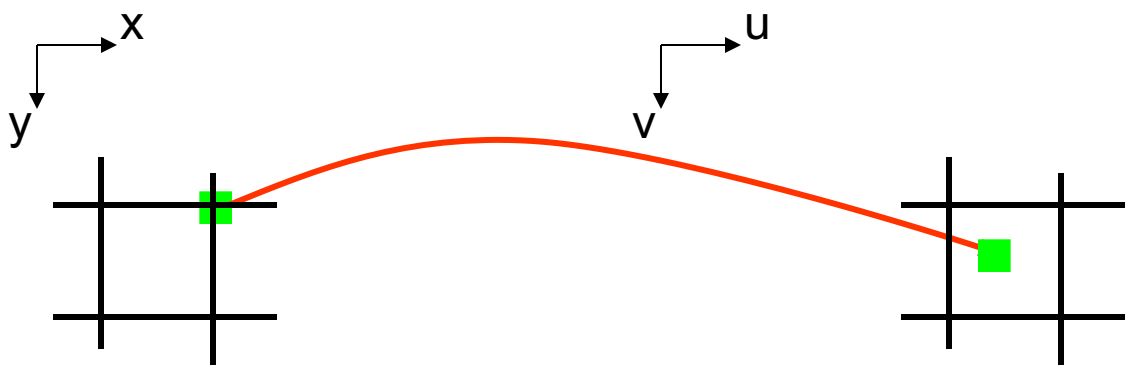


S(x,y)

T(u,v)

Q: What if the transformed pixel located between pixels?

Forward Warping



S(x,y)

T(u,v)

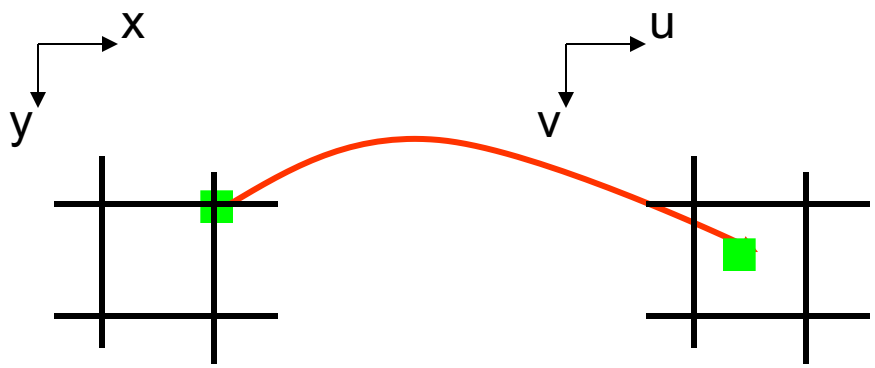
Q: What if the transformed pixel located between pixels?

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

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

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

    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.

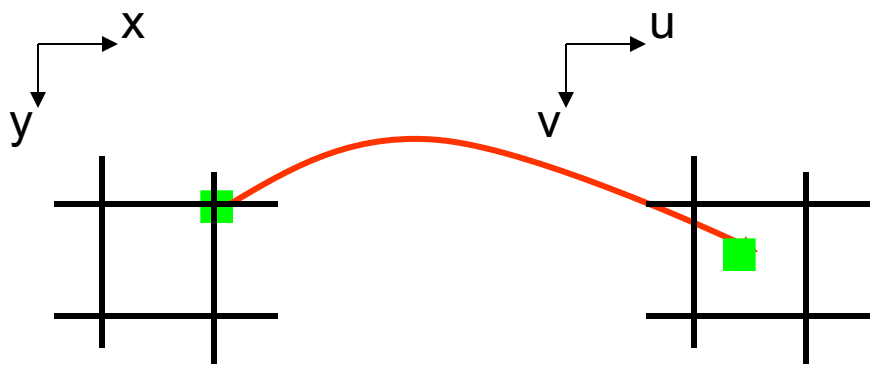
- How to remove the holes?

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

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

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

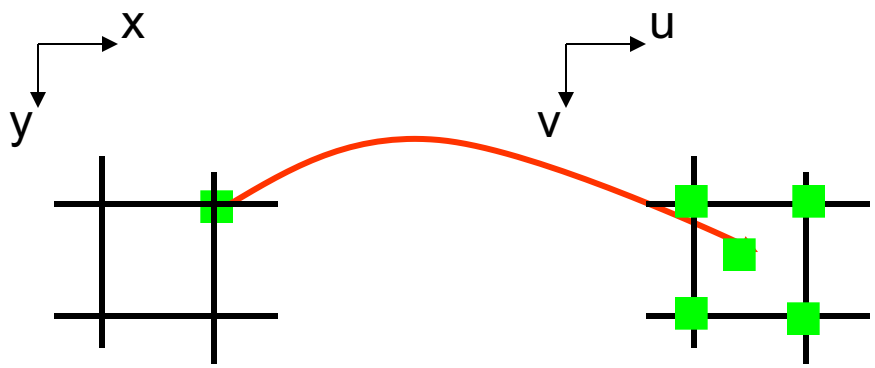
    copy pixel at source S(x,y) to T(u,v)



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?



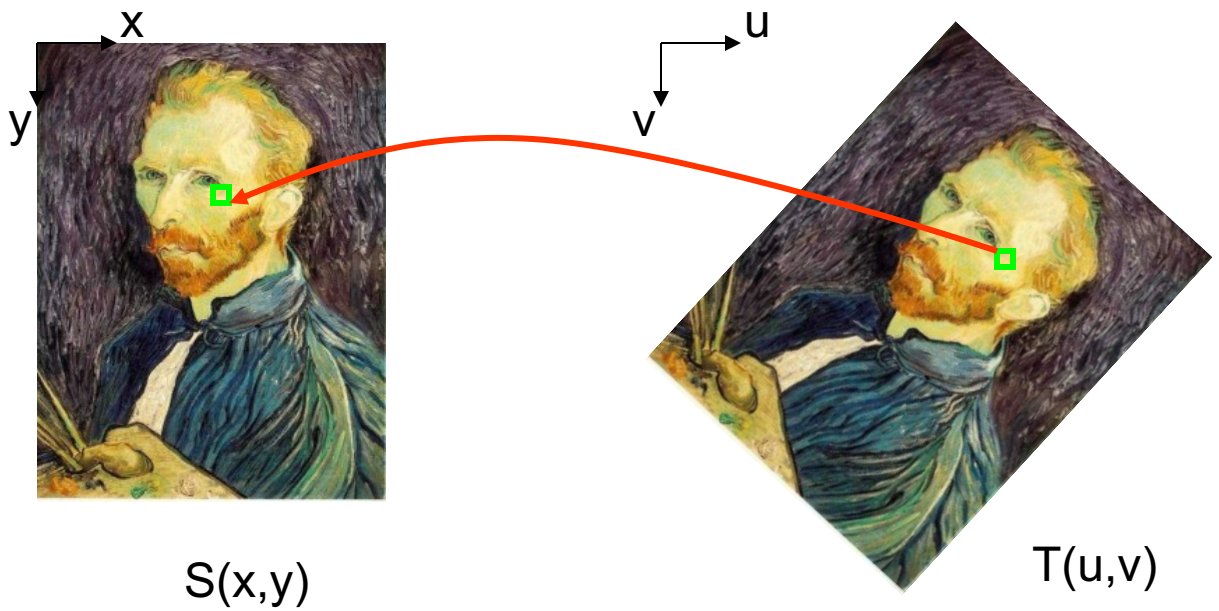
for y = y_{min} to y_{max}

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)

Inverse Warping



Inverse warping algorithm:

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

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

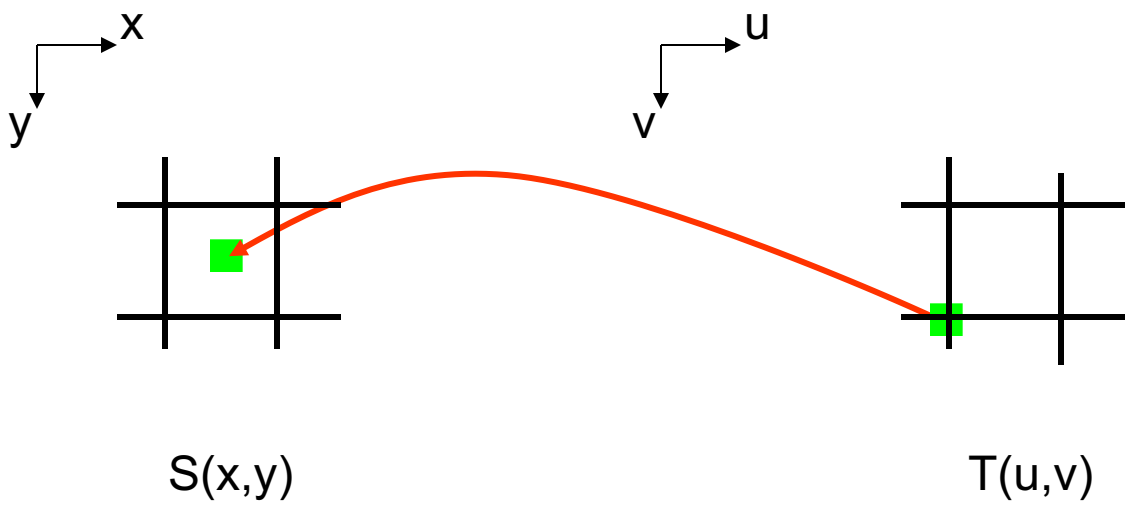
x = F(u,v); y = G(u,v)

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

map through the new coordinates

compute through the inverse transformation function

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

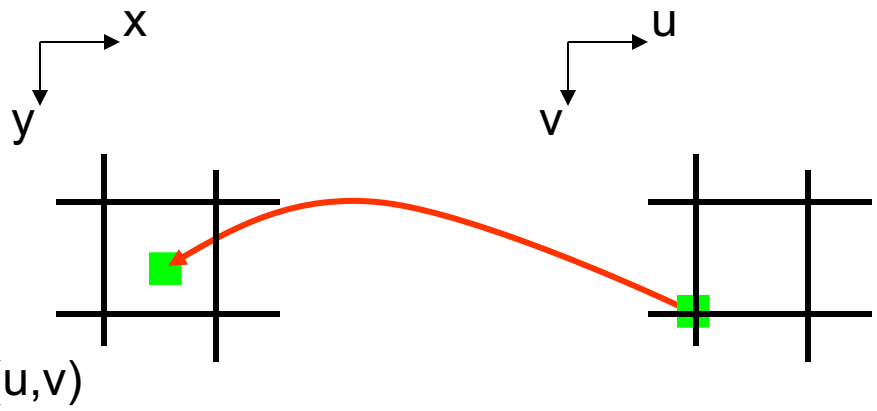
- Iterate over dest., finding pixels from source
- Non-integer evaluation source image, resample
- May oversample source
- But no holes
- Simpler, better than forward mapping

for $v = v_{min}$ 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)



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?

$$x' = x + t_x$$

$$y' = y + t_y$$

Transforming Homogeneous to Cartesian Coordinates:

Given a point in homogeneous coordinates (x, y, w), the corresponding Cartesian coordinates are obtained by dividing each homogeneous coordinate by w:

$x_{\text{cartesian}} = \frac{x}{w}$

$y_{\text{cartesian}} = \frac{y}{w}$

Transforming Cartesian to Homogeneous Coordinates:

Given a point in Cartesian coordinates (x, y), the corresponding homogeneous coordinates are obtained by adding a homogeneous coordinate (w):

$x_{\text{homogeneous}} = x$

$y_{\text{homogeneous}} = y$

$w_{\text{homogeneous}} = 1$

Homogeneous Coordinates

- A: Using the rightmost column:

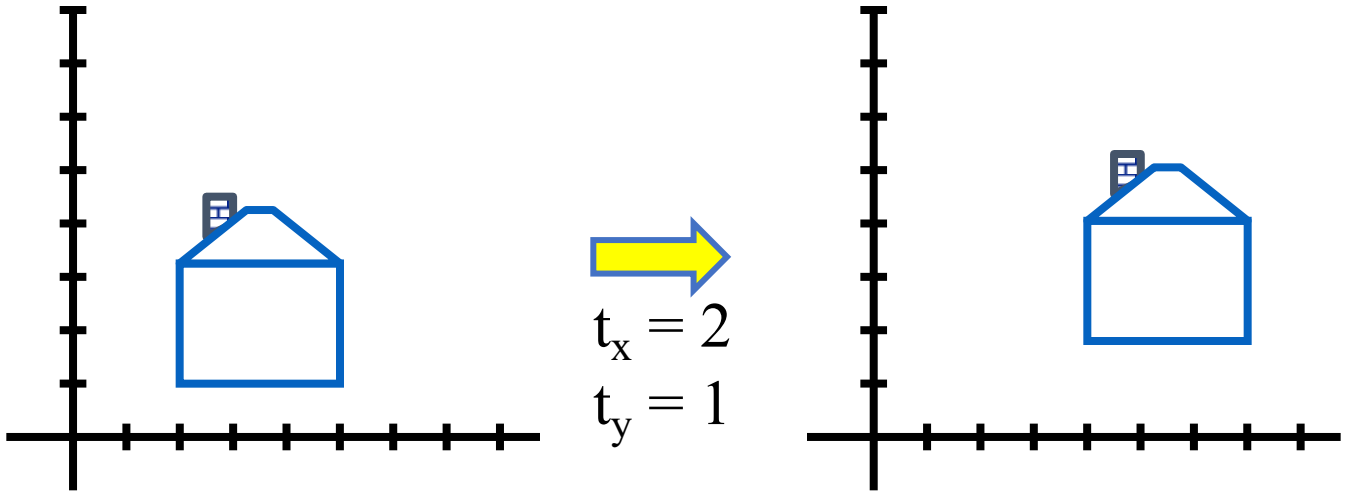
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation

- Example of translation

Homogeneous Coordinates

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



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

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

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

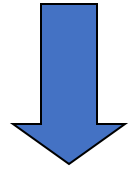
$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

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

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$\mathbf{p}'$ = $T(t_x,t_y)$ $R(\Theta)$ $S(s_x,s_y)$ $\mathbf{p}$

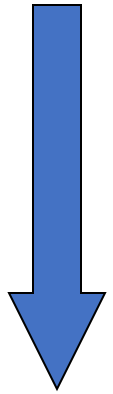
translate rotation shear

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
  - nder composition
  - change of basis

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

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$



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

In two-dimensional space, an affine transformation can be expressed using the following mathematical equations:

$$x' = a \cdot x + b \cdot y + c$$
$$y' = d \cdot x + e \cdot y + f$$

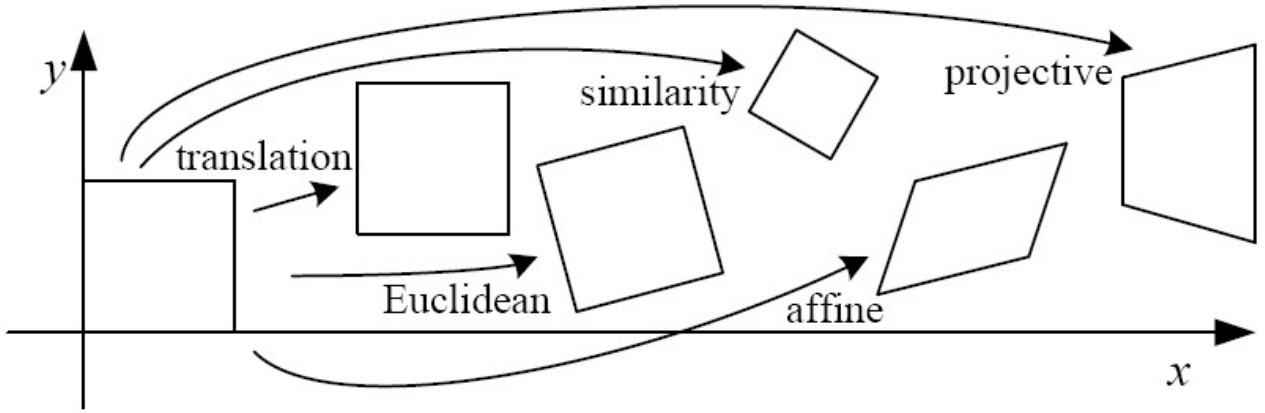
Here, (x, y) are the coordinates of a point in the original space, and (x', y') are the coordinates of the transformed point. The coefficients a, b, c, d, e, and f define the transformation. The matrix representation of the transformation is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

Perspective effects

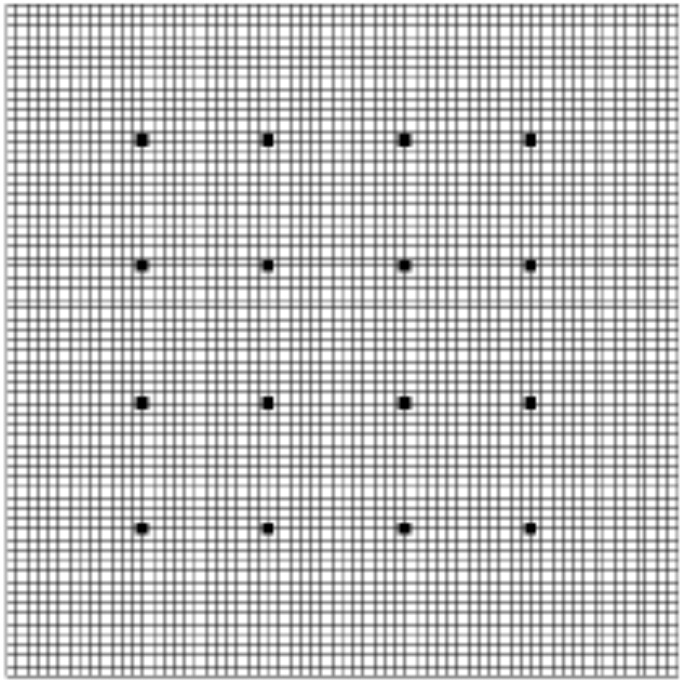
3D computer vision aims to solve an inverse problem of computer graphics

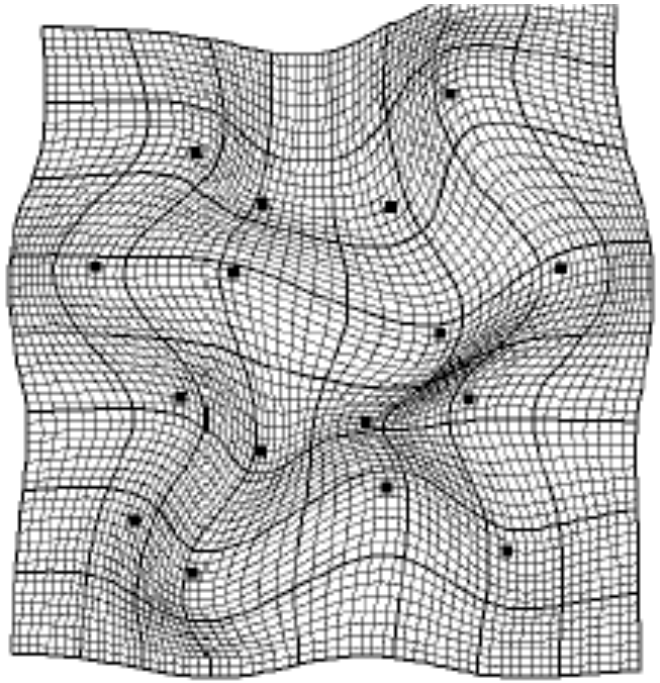
Hierarchy of 2D image transformations (stratification)



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

Free Form Warping





94

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.