Image Processing Basics

Week-2

- Course Representative: To be announced

Announcement

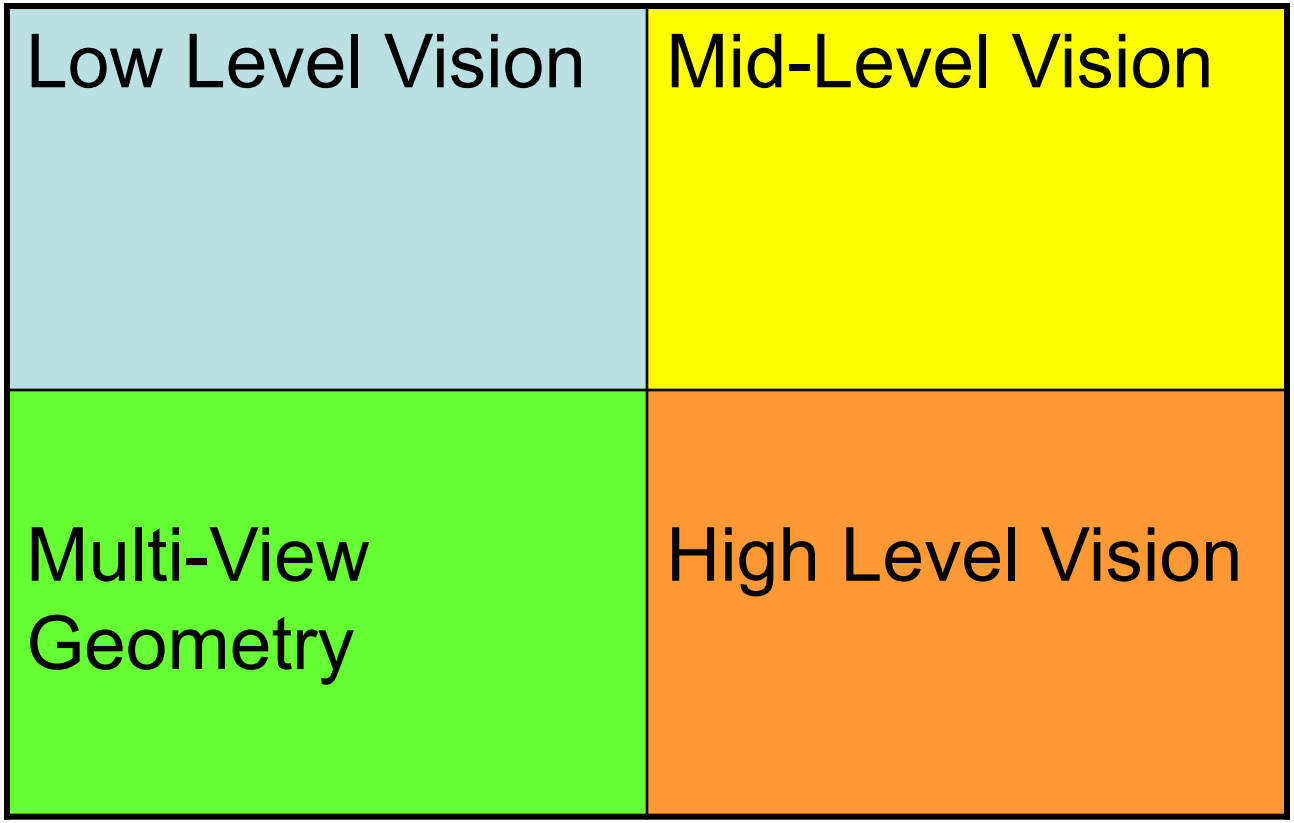
- Yujiao (tutor) will give a tutorial about basic programming in matlab and python on (4:00pm-5:00pm), 3rd Mar. 2021. (Please find the material about tutorial of using matlab and python on wattle)
- Please use the same zoom link for that session.
- Lab enrolment will end at 3rd Mar. 2021.
- After registering the lab session through the 'lab-enrollment', please join teams using the following class code: 4eub2m9
- We will have Clab-Group-(1,2,3,4) in teams based your preference in the 'lab-enrolment'.

Timetable for Lab-assignment 1

- Lab assignment one will be released by Friday (Mar. 5th, 2021)
- Lab Q&A with tutors: Week 03, Week 04
- Week05: students work on assignment by themselves.
- Deadline for Lab Assignment one:

due 11:59pm, 28th March 2021 (end of week 5), worth 15%

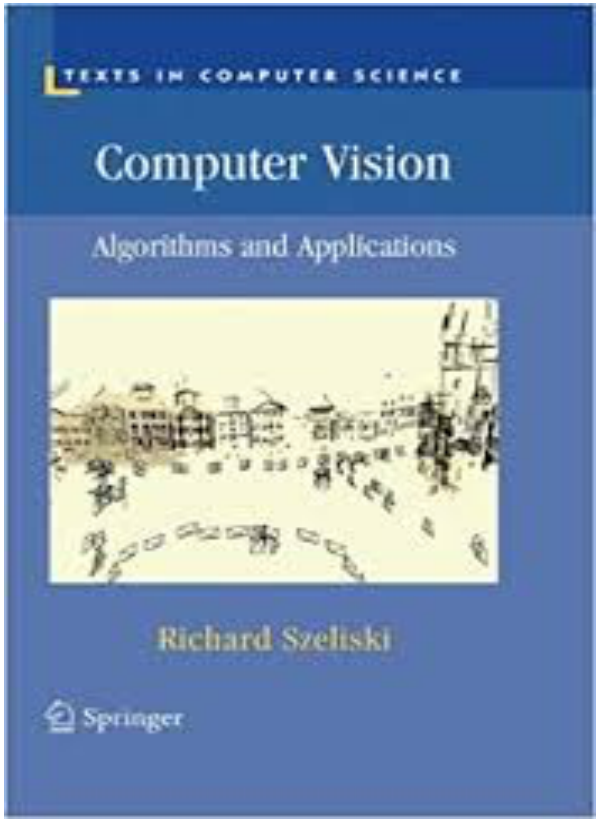
Four modules



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

- Read text book:

- 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)
  - Frequency domain (Fourier transform) (Chapter 3.4- Week 03)
  - Binary image analysis (Chapter 3.4 - Week03)
- Geometric operations (Chapter 3.6)
  - Rotation, translation, affine...

Recall: digital image representation

0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

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

- Image processing transforms
- The output of such transformation only depends on the the corresponding input pixel value.

Point Operations

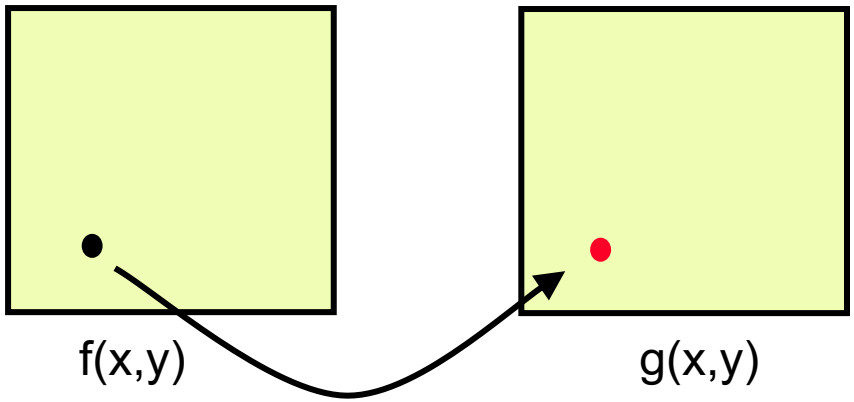
- Operation on Pixel's value.
- Context free (memory-less).
- Operation can be performed on the *image histogram*.
- Example:

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

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

α Is a constant, controlling the contrast.

β Is a constant, controlling the brightness.



Point Operations

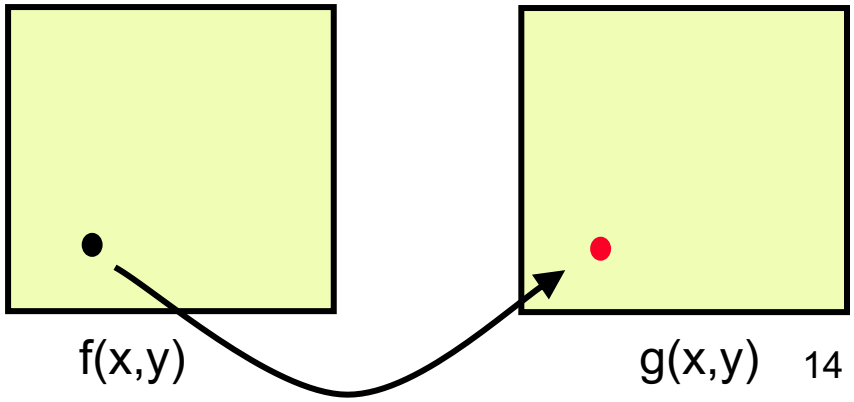
- Operation on Pixel's value.
- Context free (memory-less).
- Operation can be performed on the *image histogram*.
- Example:

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

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

α Is a spatially varying.

β Is a spatially varying.

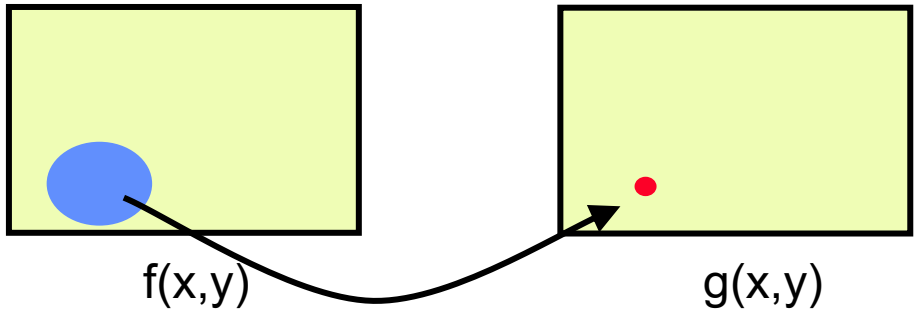


Neighbourhood Operations

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

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

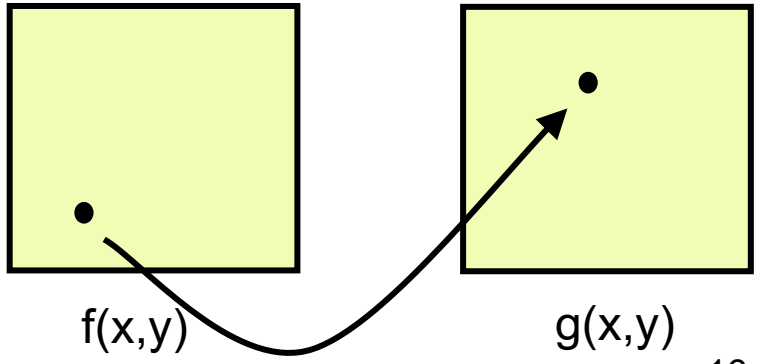
$$g(x,y) = M(\{f(i,j) | (i,j) \in N(x,y)\})$$



Geometric Operations

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

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



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

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 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

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

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.

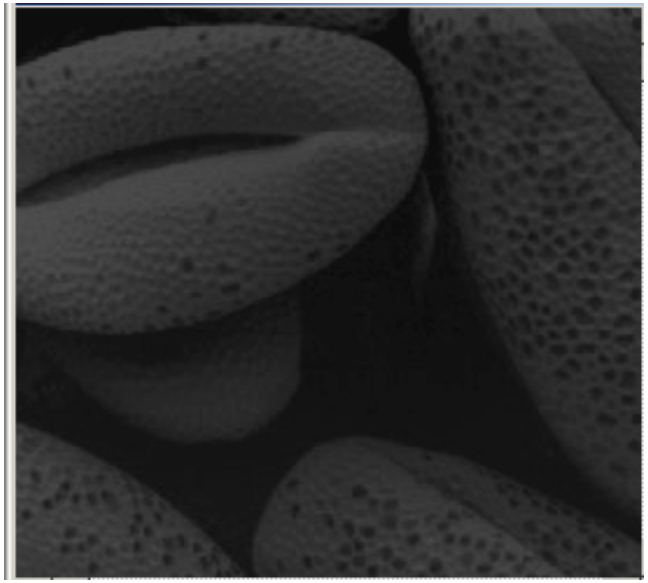
- Histogram provides a global description of the appearance of an image.

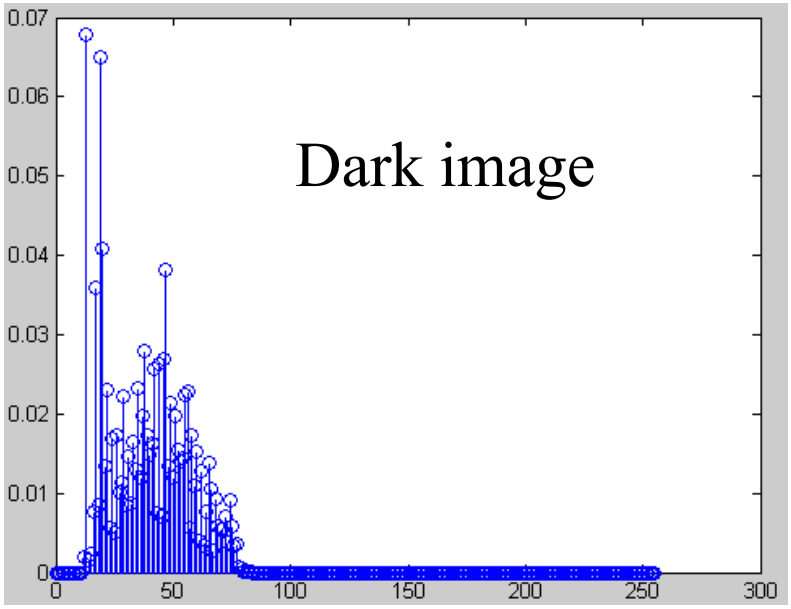
- If we consider the gray values in the image as realizations of a random variable R, with some probability density, histogram provides an approximation to this probability density.
- In other words,

$$\Pr(R = r_k) \approx p(r_k)$$

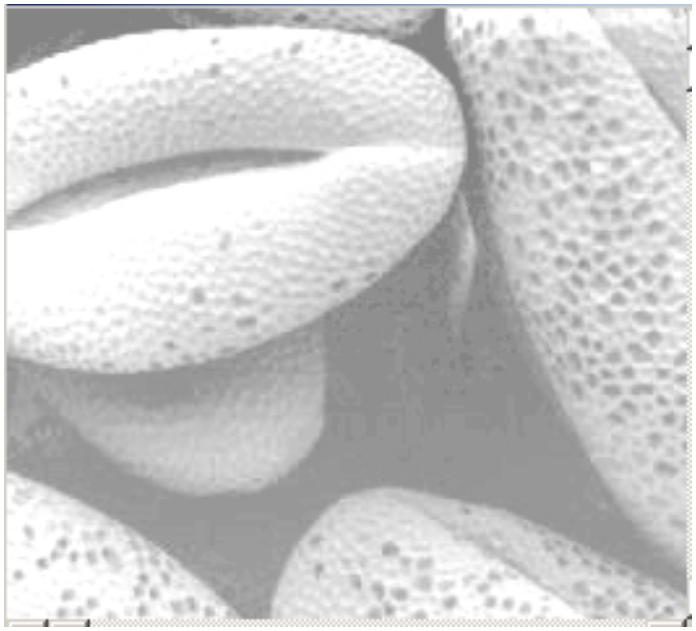
Some Typical Histograms

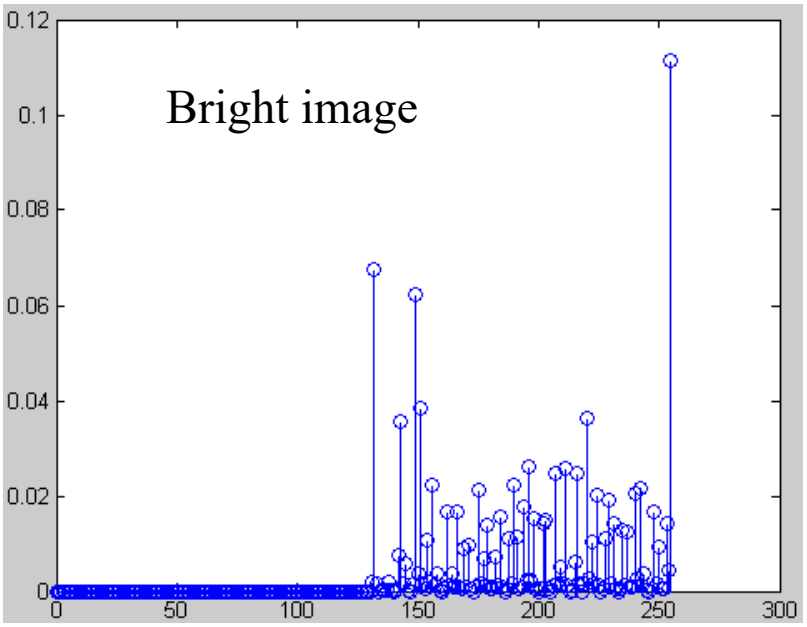
- The shape of a histogram provides useful information for image global appearance.

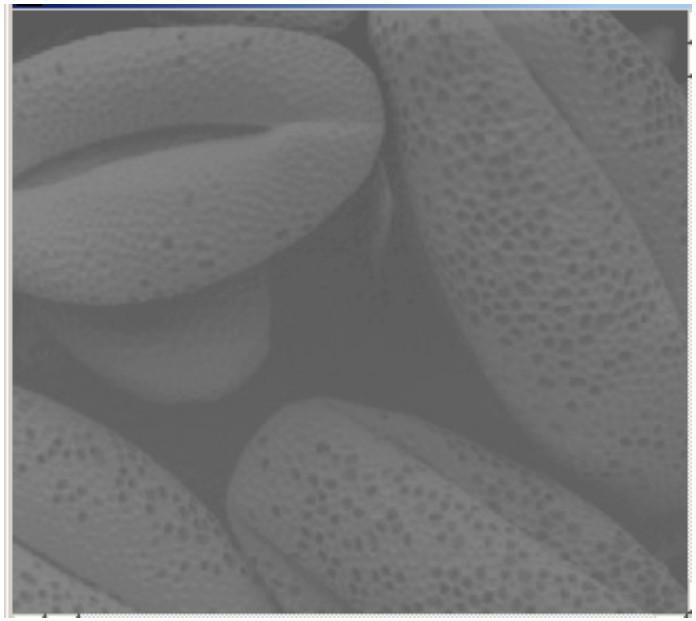


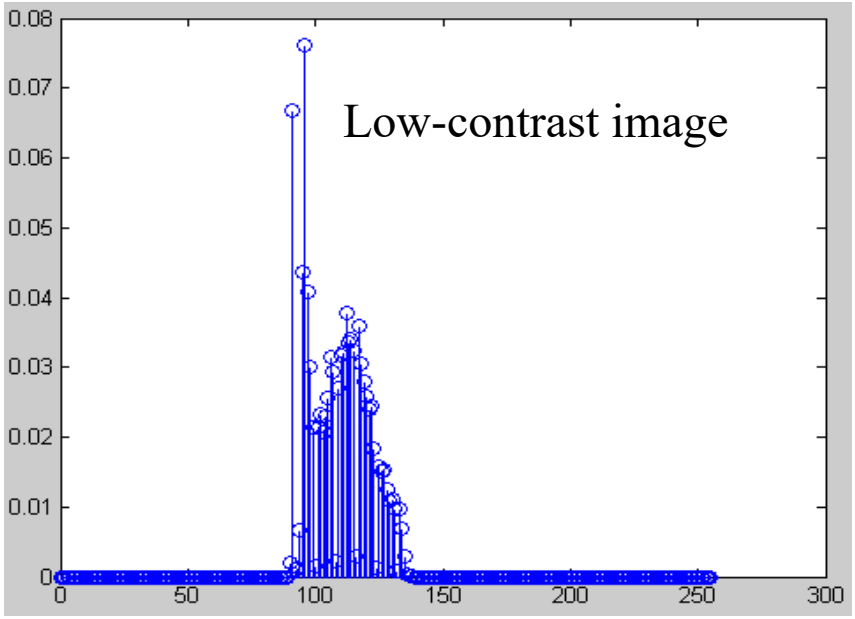


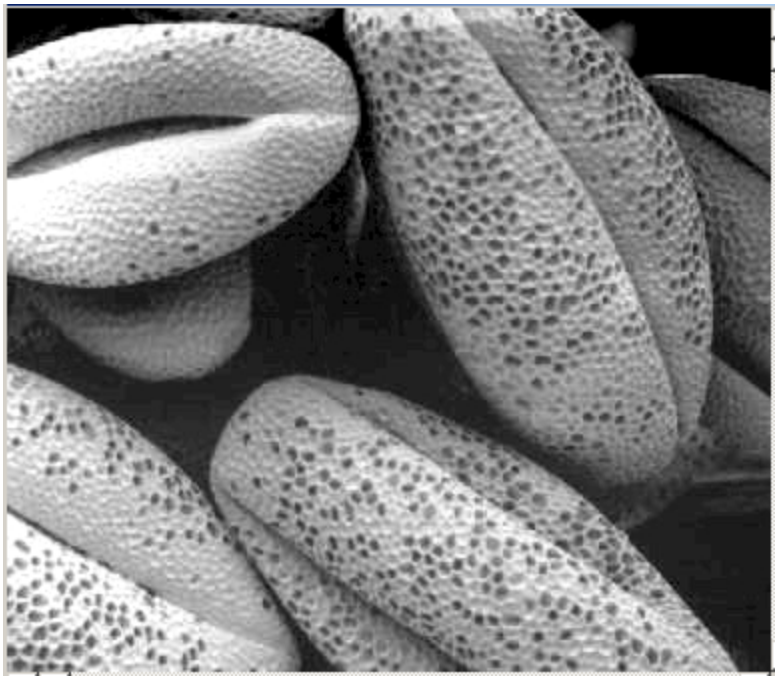

```
>> clear;
[ix,map]=imread('Fig3_15a.jpg');
imshow(ix)
figure;
ix=double(ix);
h=histogram(ix);
figure
stem(0:255,h);
```

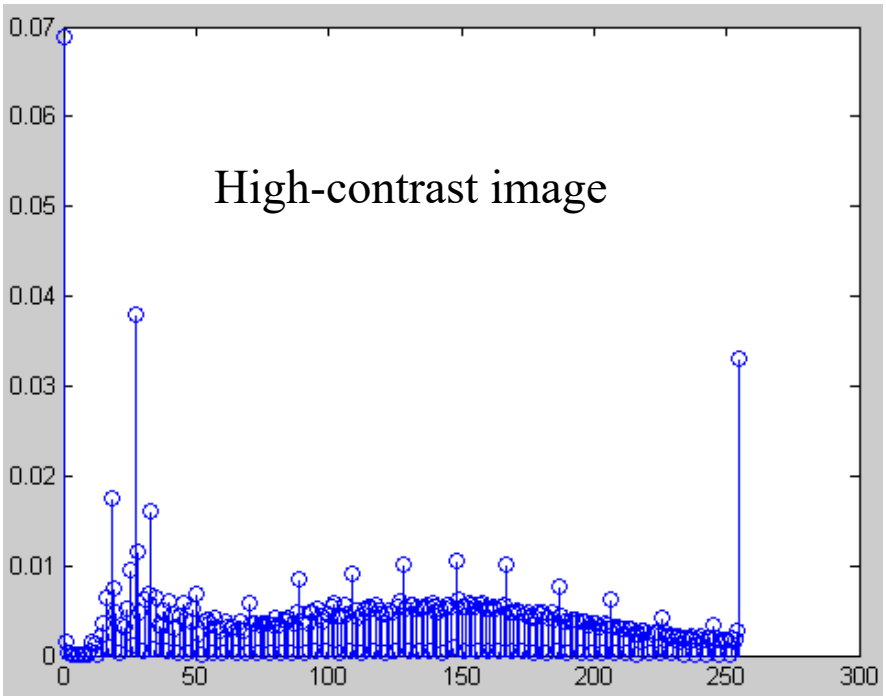



Bright image
0.12
0.1
0.08
0.06
0.04
0.02
0
0
50
100
150
200
250
300




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




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

Application: image enhancement

Histogram-based image enhancement/image editing

- Histogram modification
- Histogram equalization
- Histogram matching

Histogram modification

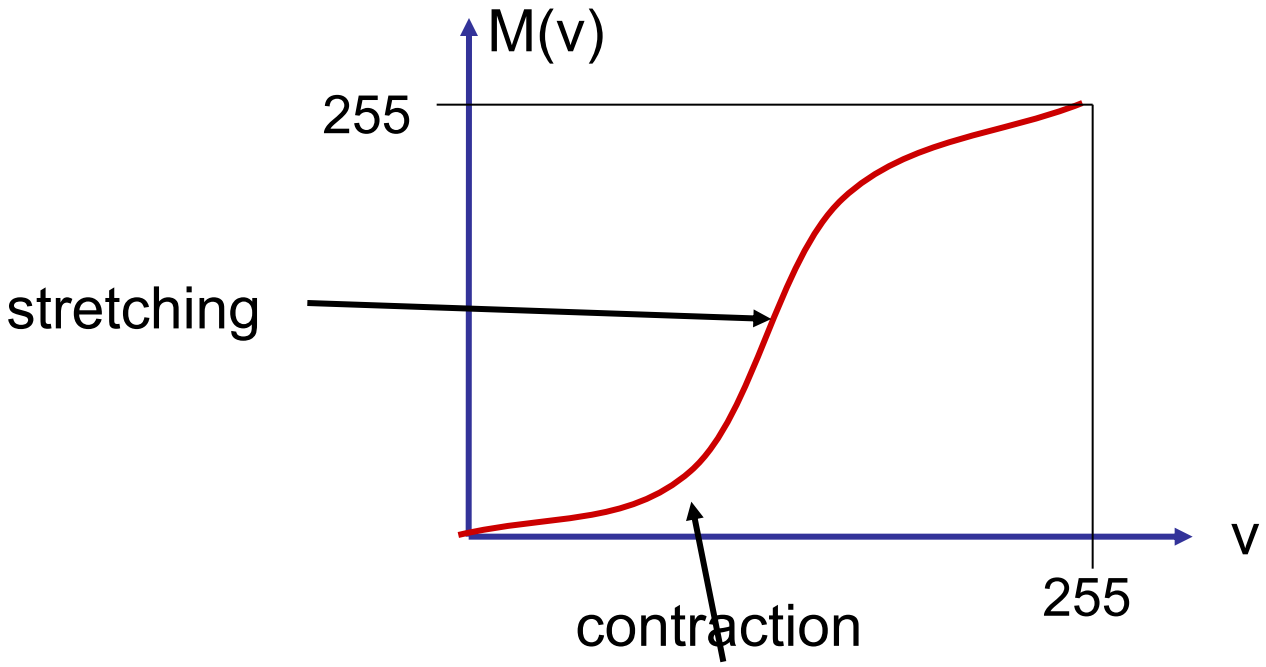
- Given a point operation:

$$v_b = M(v_a)$$

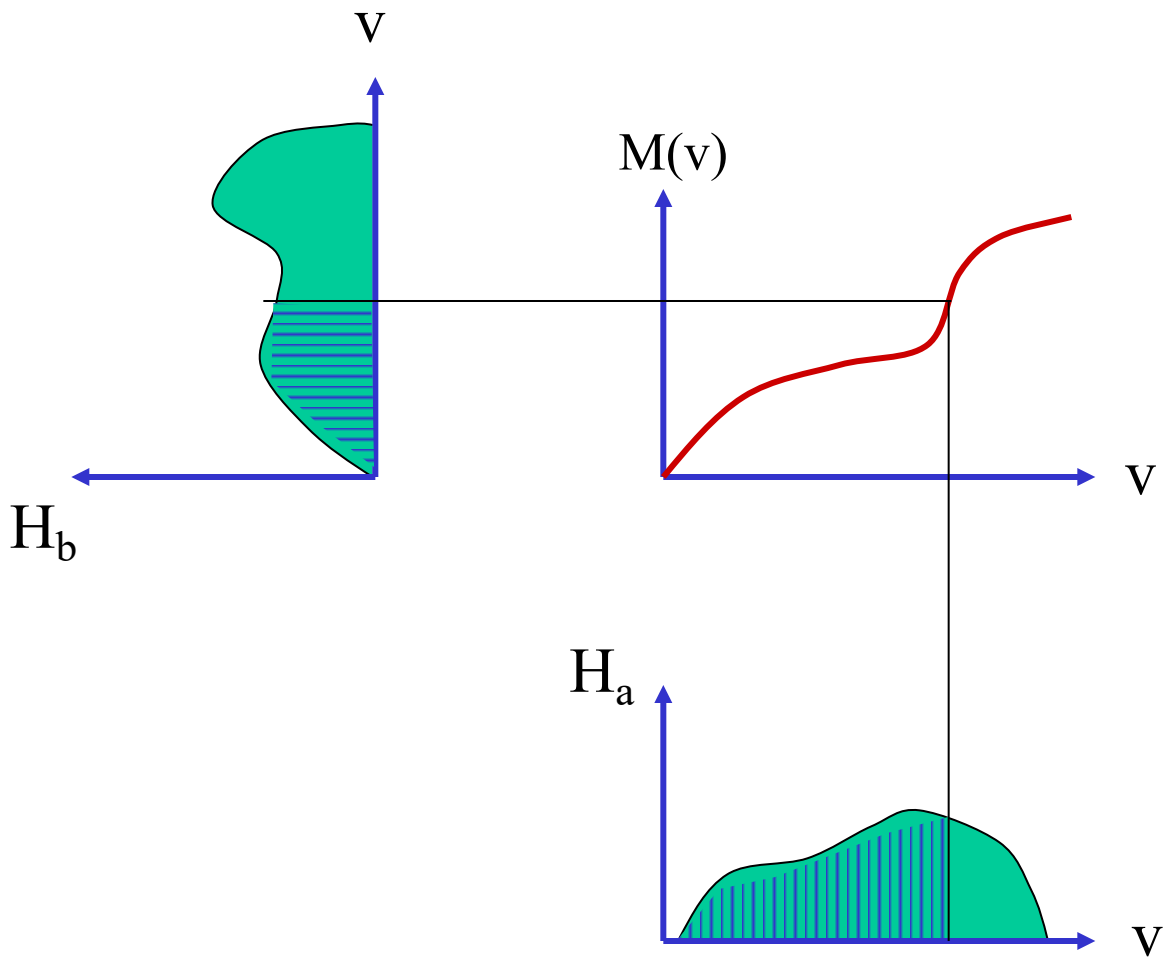
- $M(v_a)$ takes any value v_a in image A and maps it into v_b in image B.
- **Requirement**: the mapping M is a non-descending function (M^{-1} exists).
- In this case, the area under H_a between 0 and v_a is equal to the area under H_b between 0 and v_b. Namely, the cumulatively density function (CDF)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

Monotonicity Requirement

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

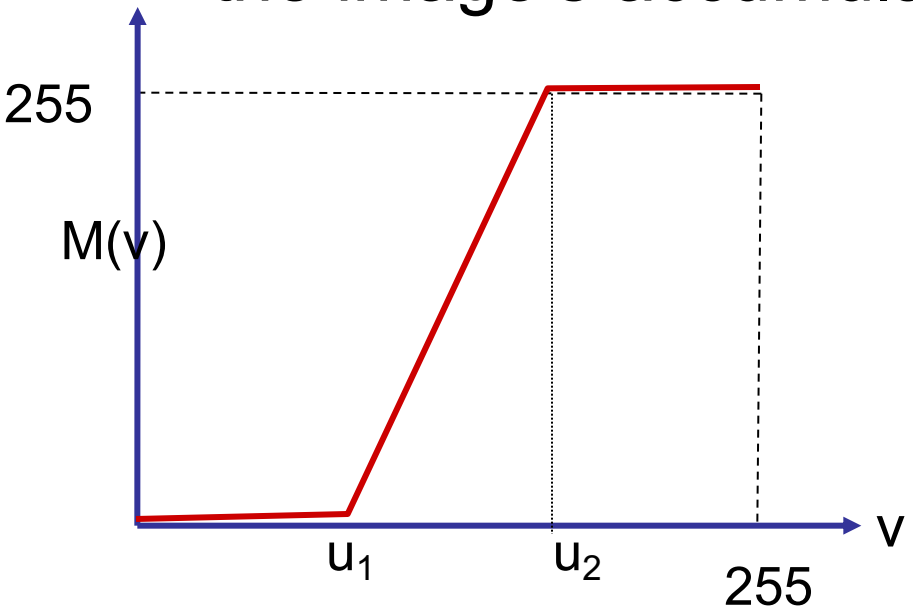


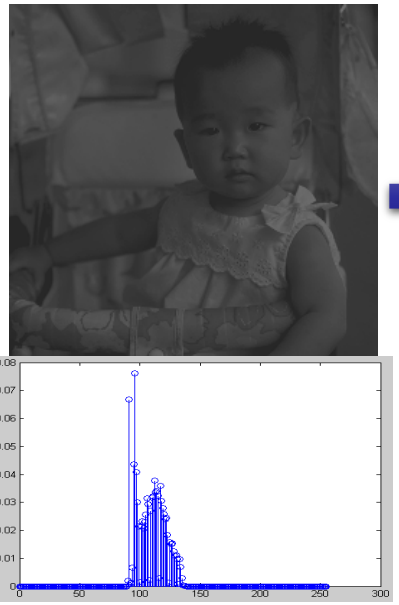
- The area under H_a between 0 and v_a is equal to the area under H_b between 0 and $v_b=M(v_a)$



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

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

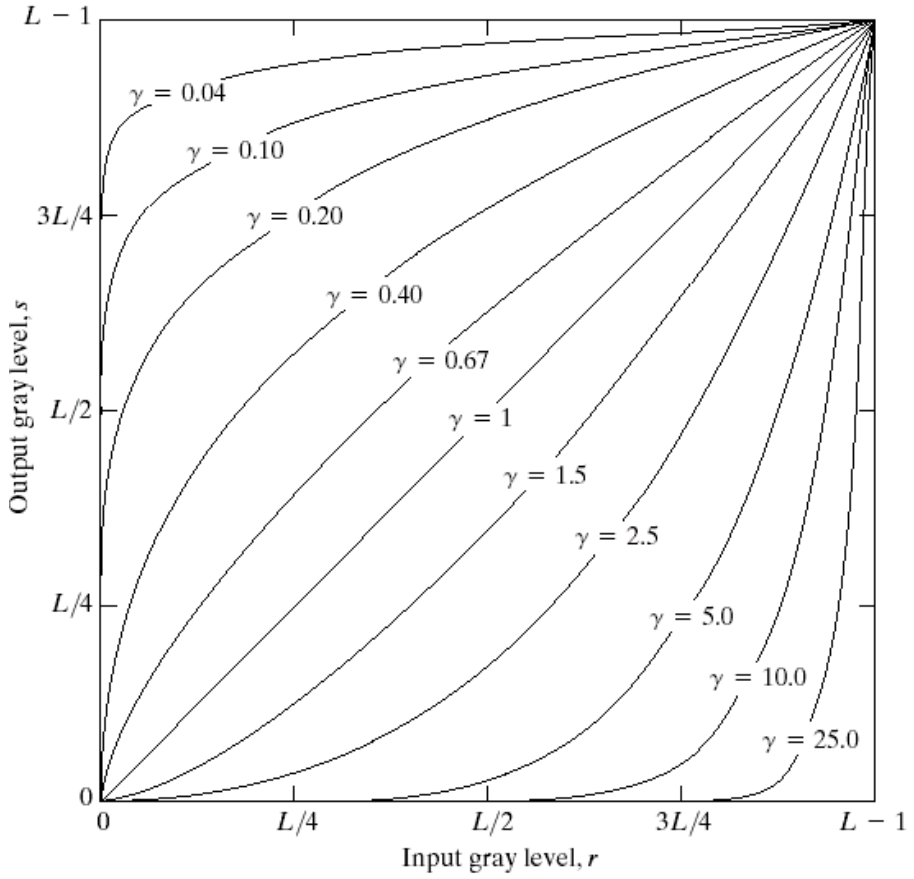


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

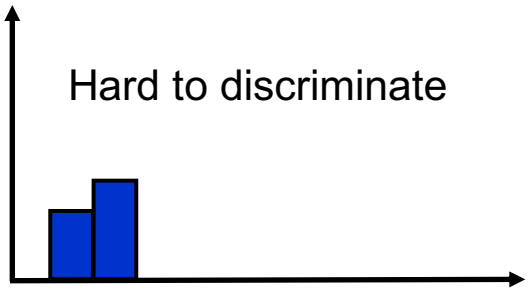
a b
c d

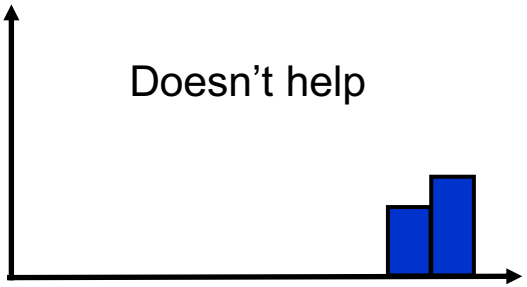
FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0, respectively. (Original image for this example courtesy of NASA.)

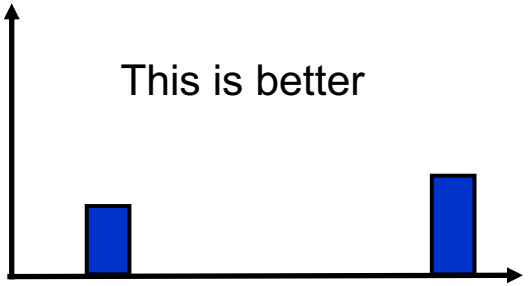
Histogram Equalization

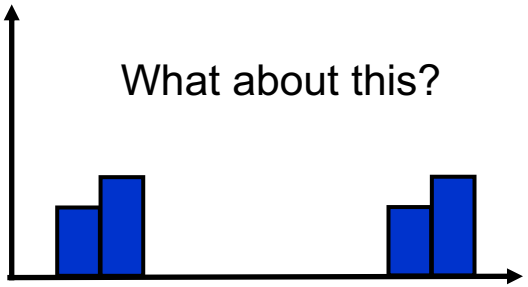
- Visual contrast between objects depends on the their gray-level separation.

- 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:

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

L: Total number of gray levels

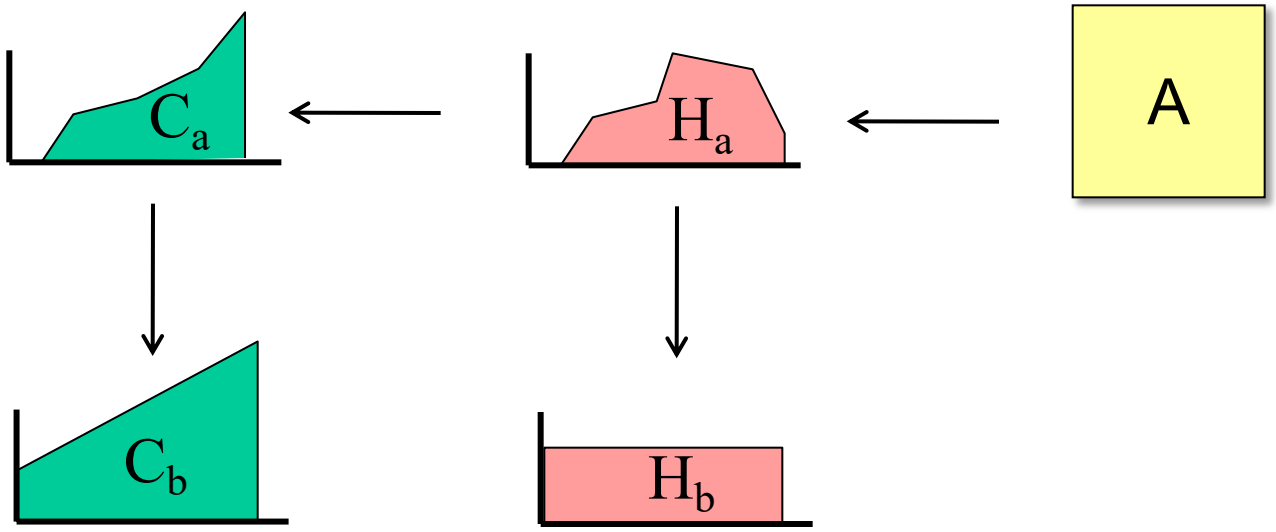
n_k: Number of pixels with gray value r_k

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

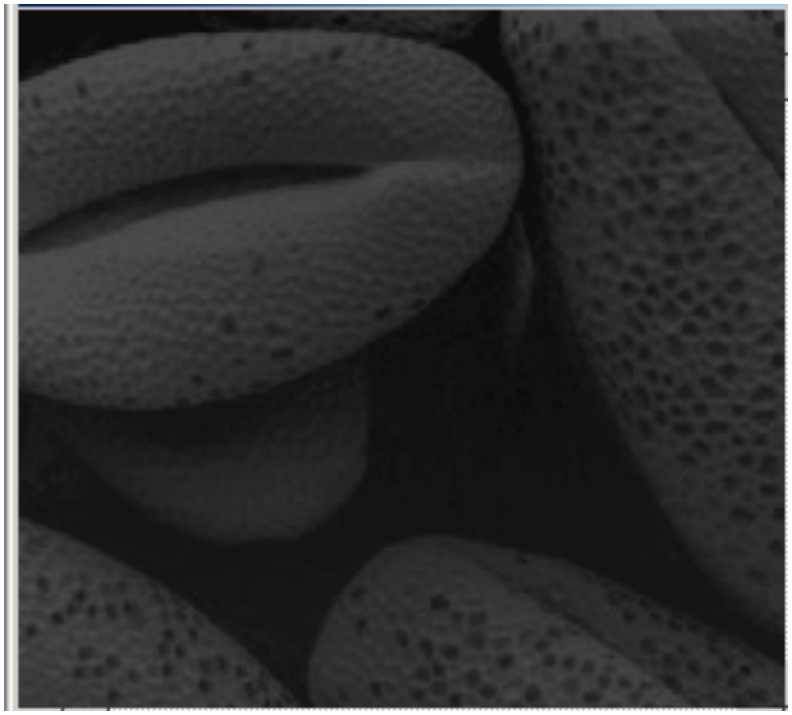


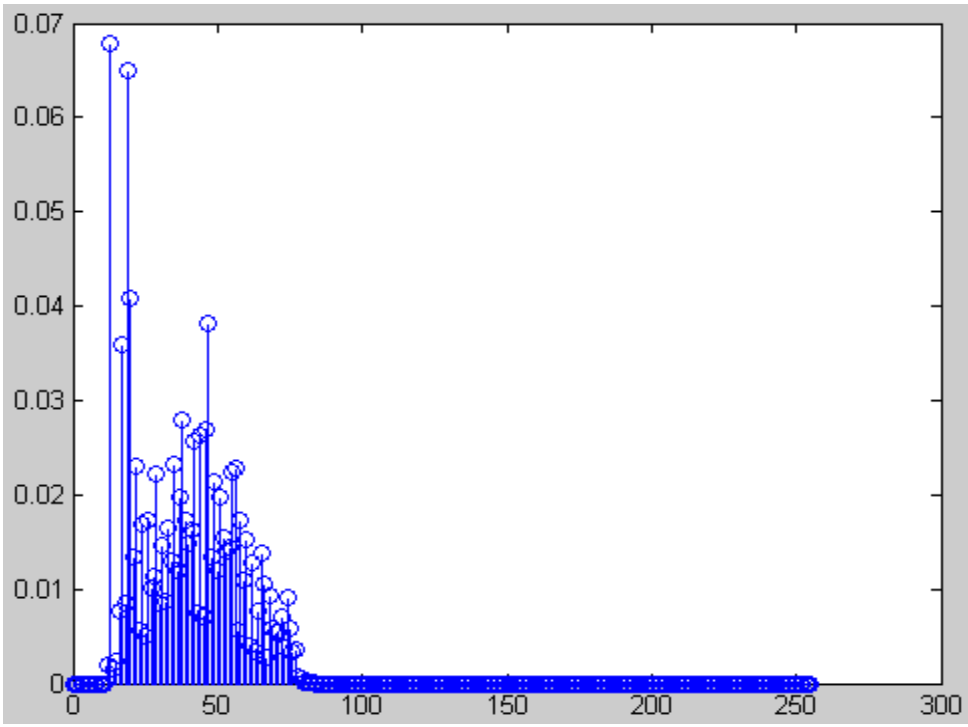
- Define: $C_b(v) = v * \frac{(\#\ pixels)}{\#\ grayValues}$ Or $C_b(v) = \frac{v}{L}$

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

Example

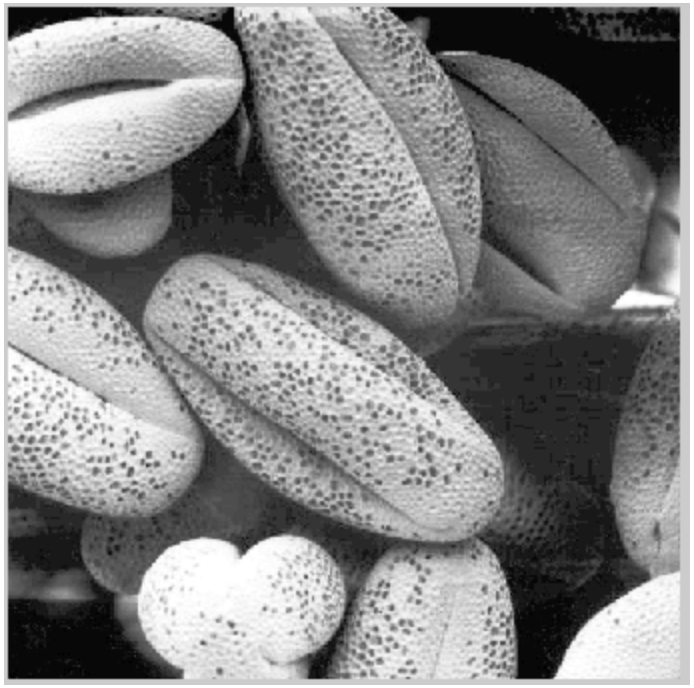
Original image and its histogram

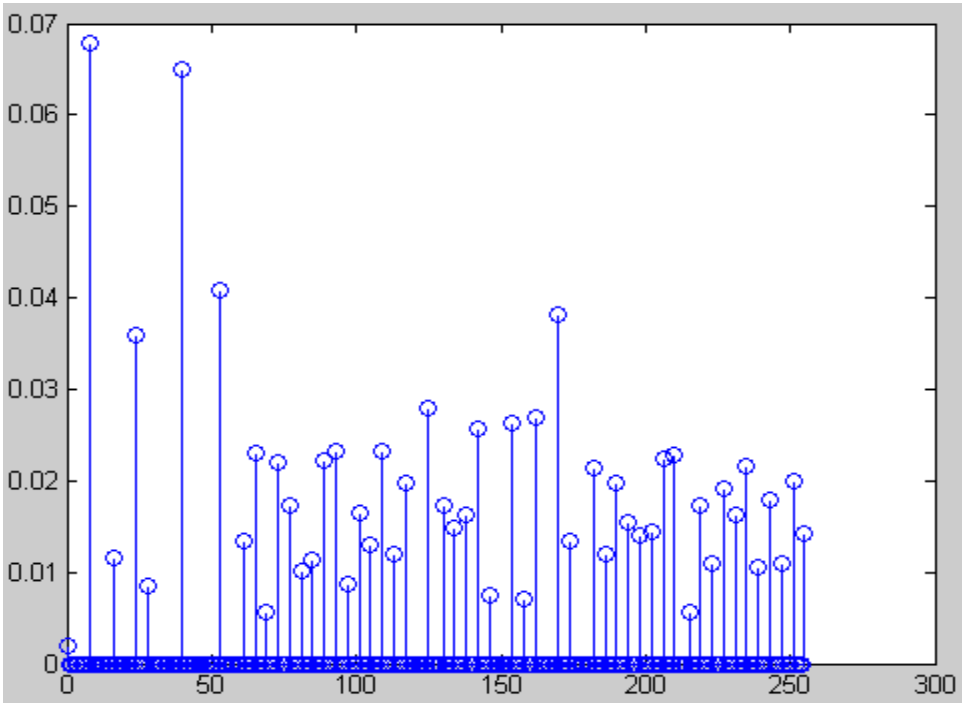





```
>> clear;
[ix,map]=imread('Fig3_15a.jpg');
imshow(ix)
figure;
ix=double(ix);
h=histogram(ix);
stem(0:255,h);
```

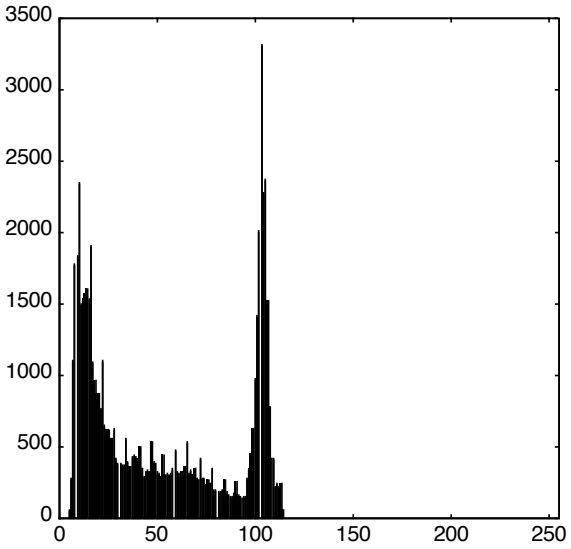
Histogram equalized image and its histogram



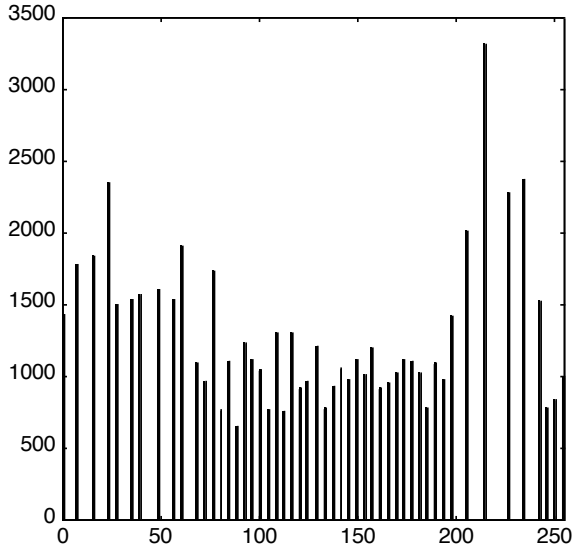



```
>> clear
>> [ix,map]=imread('Fig3_15a.jpg');
>> imshow(ix);
>> iy=histeq(ix);
>> figure
>> imshow(iy);
>> iy=double(iy);
>> hy=histogram(iy);
>> figure
>> stem(0:255,hy);
```

Hist. Eq.: Example 1

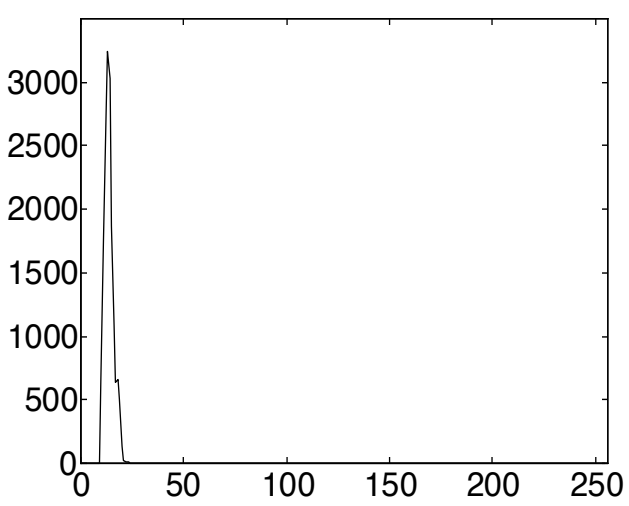


Original

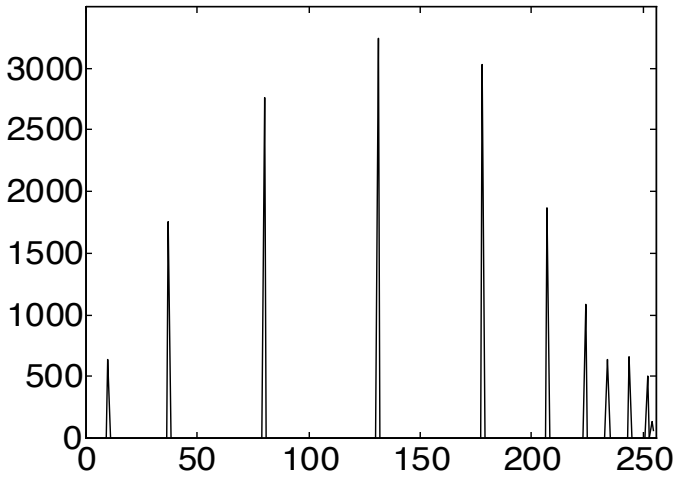


Equalized

Hist. Eq.: Example 2



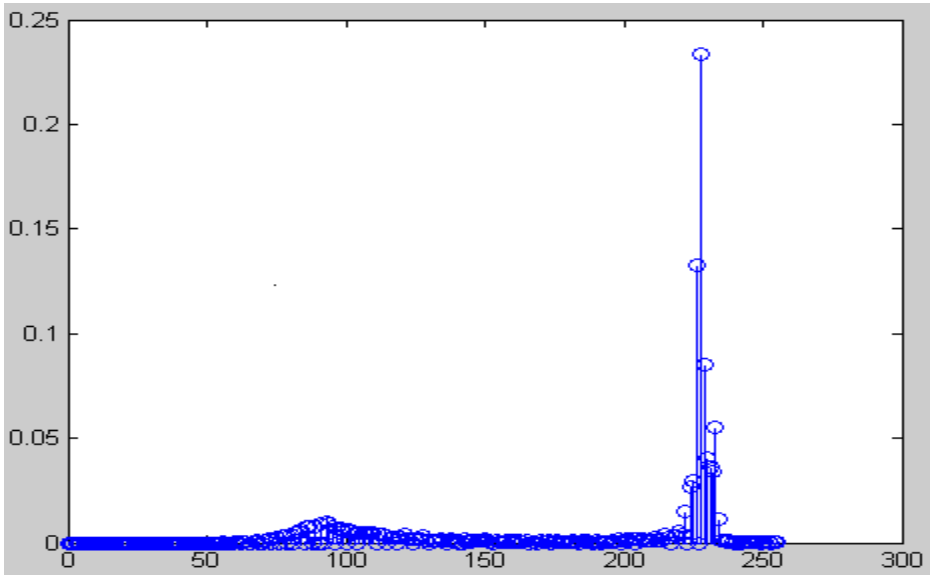
Original

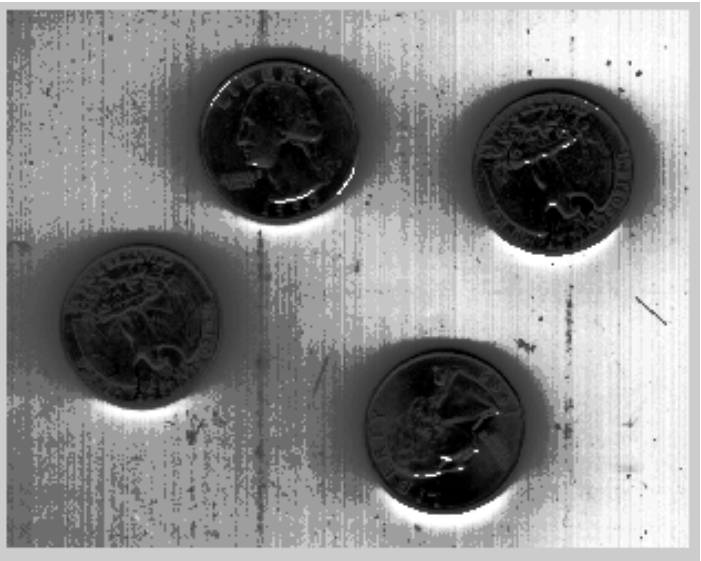


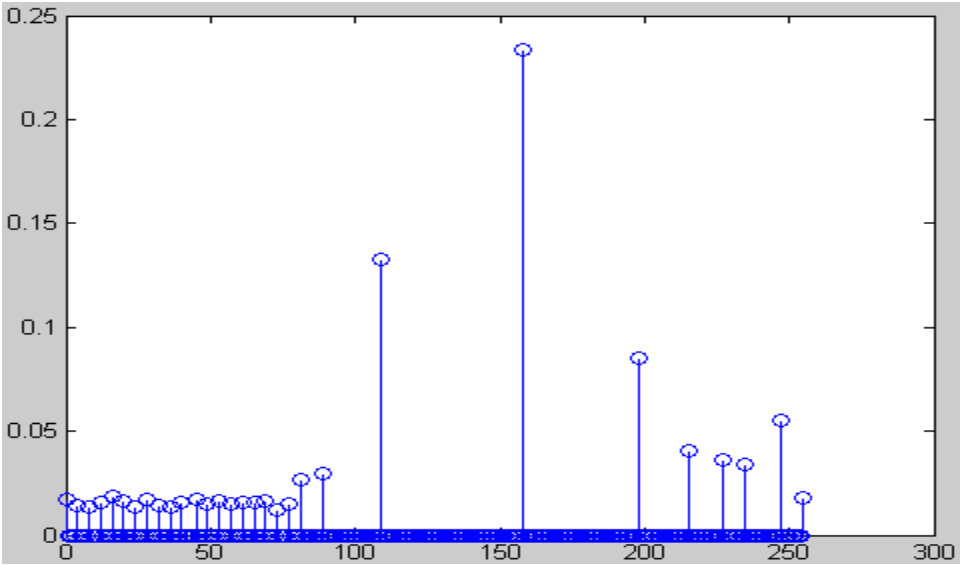
Equalized

• 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.”








```
>> clear
>> ix=imread('eight.tif');
>> imshow(ix);
>> iy=histeq(ix);
>> figure
>> imshow(iy);
>> ix=double(ix);
>> figure
>> hx=histogram(ix);
>> stem(0:255,hx);
>> iy=double(iy);
>> figure
>> hy=histogram(iy);
>> stem(0:255,hy);
```

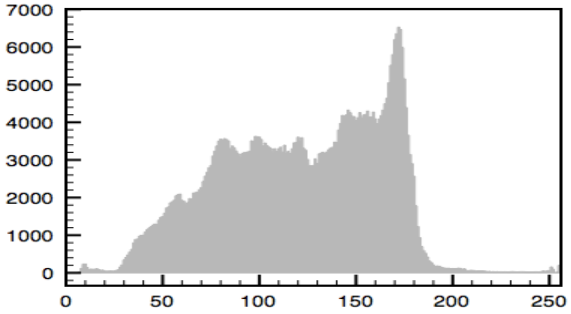
Histogram Matching

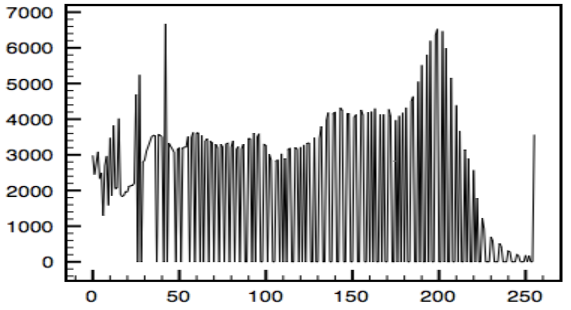
- Histogram matching is a process where an image is modified such that its histogram matches that of another (reference) image.

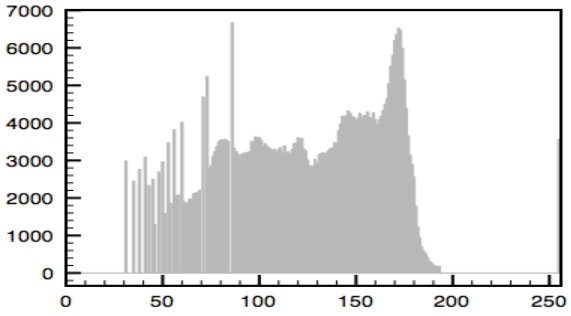
- A common application of this is to match the images from two sensors with slightly different responses, or from a sensor whose response changes over time.

- Homework: think about how to implement “histogram matching” ? (hint: use “histogram equalization” as an intermediate step.).

Histogram matching example







Geometric Transformations

(spatial transformation, image warping)

Image Warping

http://www.jeffrey-martin.com

Some slides from Steve Seitz

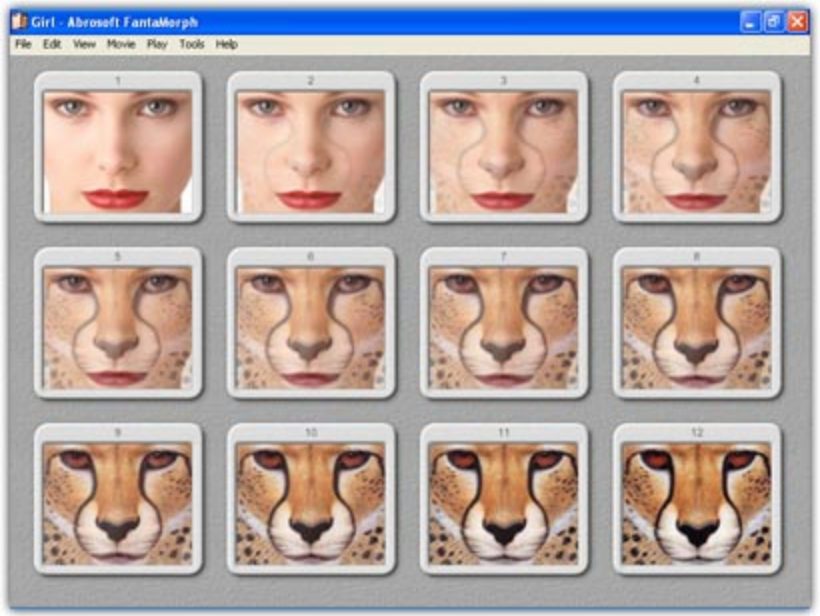
Image Warping

Why?

- texture mapping

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

- image morphing/blending



- image mosaic (stitching)

- image based-modeling & rendering

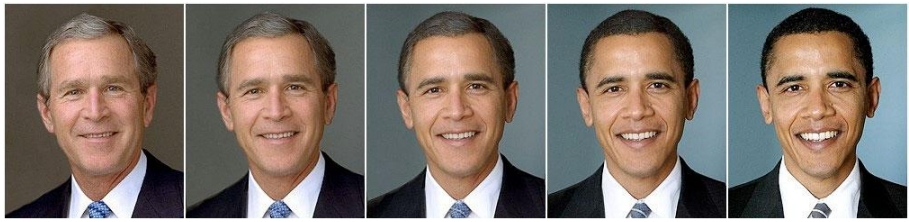


Image Warping

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

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

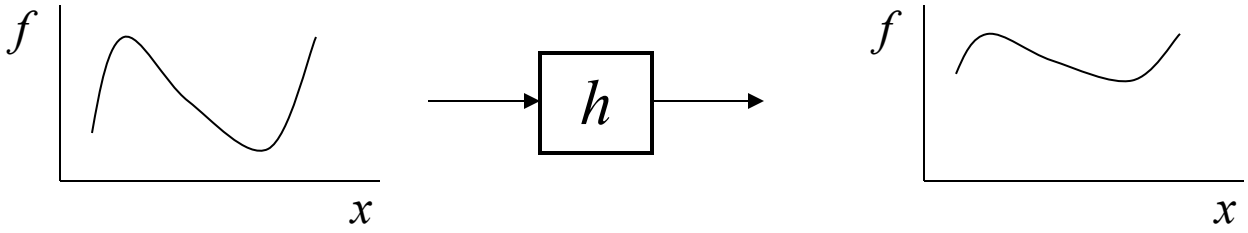


Image warping: change the spatial ***domain*** of image

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

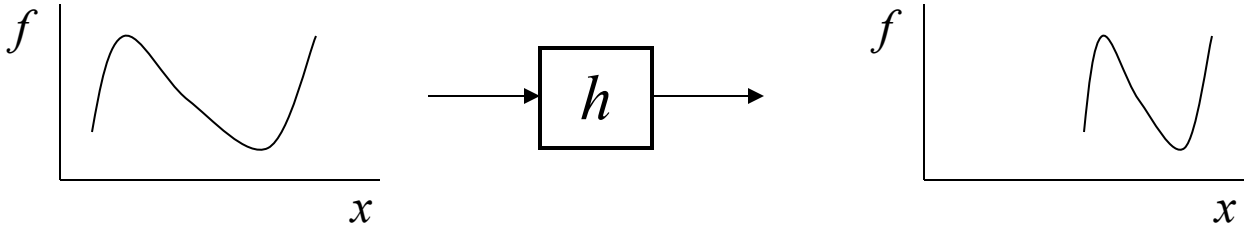


Image Warping

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

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

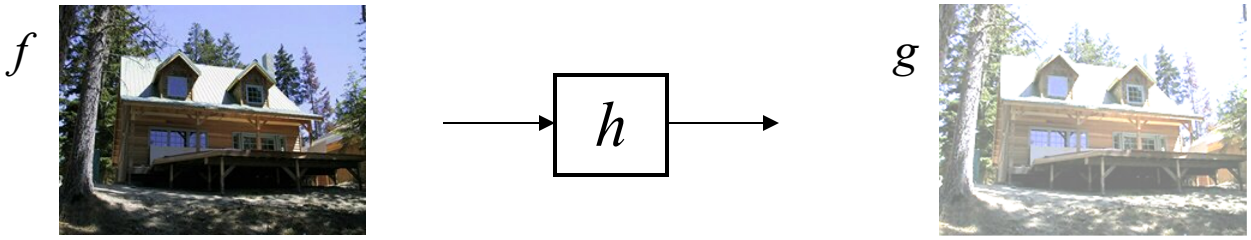
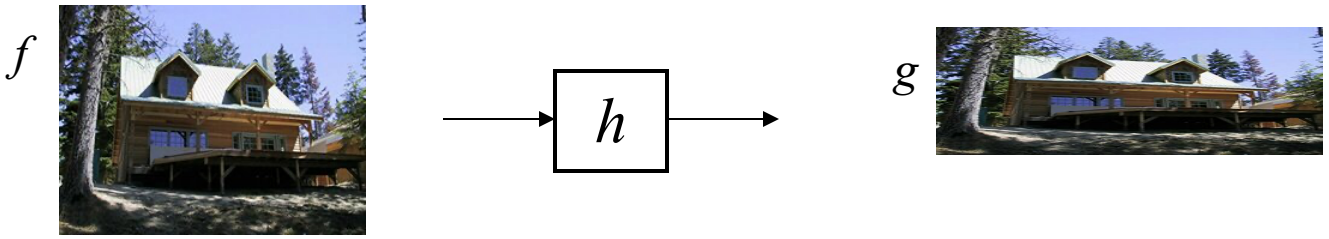


Image warping: change ***domain*** of image

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



Types of Image Warping

Simple geometric transformations:

- Rotation;
- Similarity
- Affine mapping
- Projective mapping

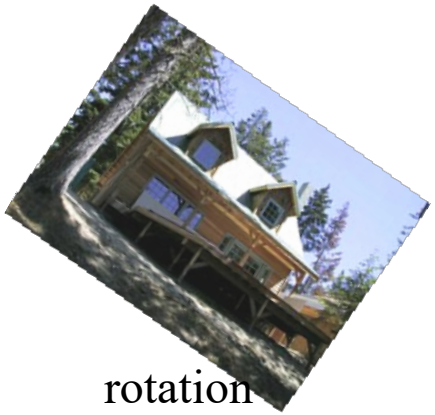
Other general types of transformations

free-form deformation

Geometric transformation

Examples of transformation

translation



rotation

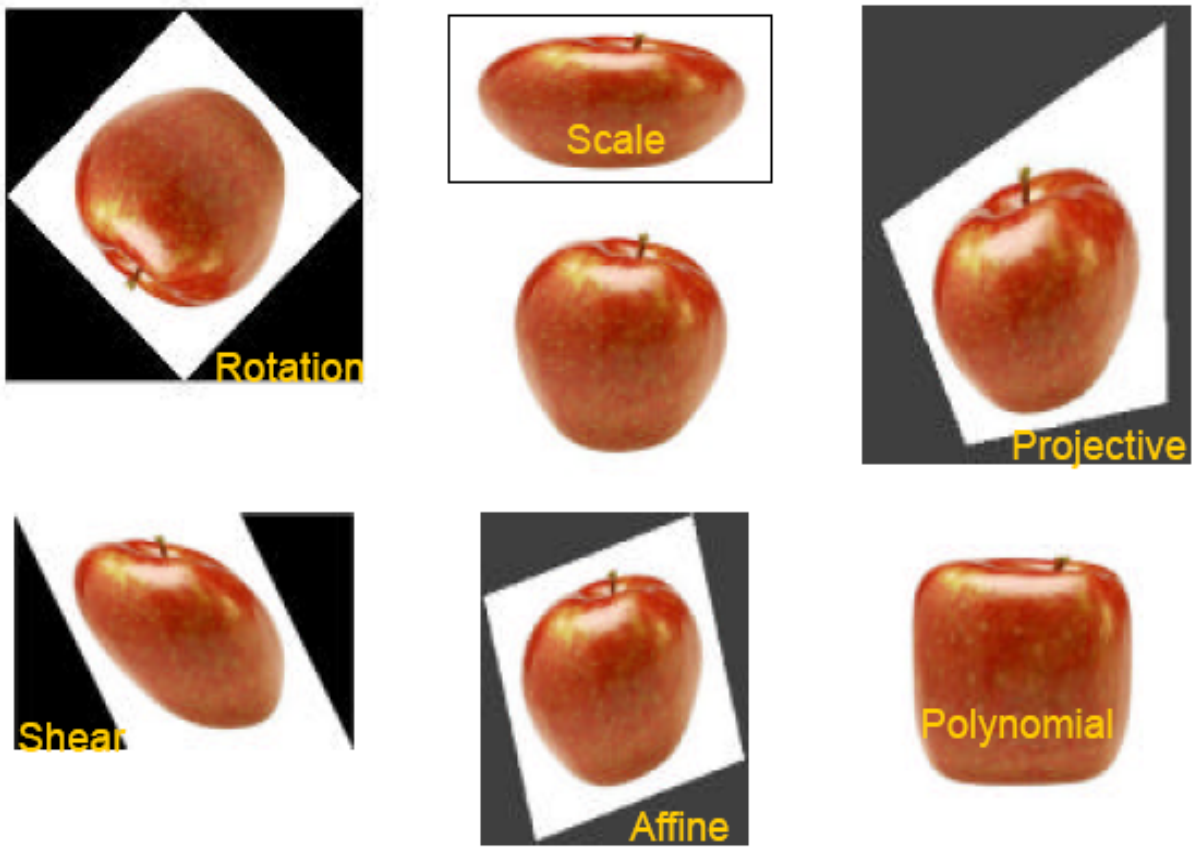
aspect

affine

perspective

cylindrical

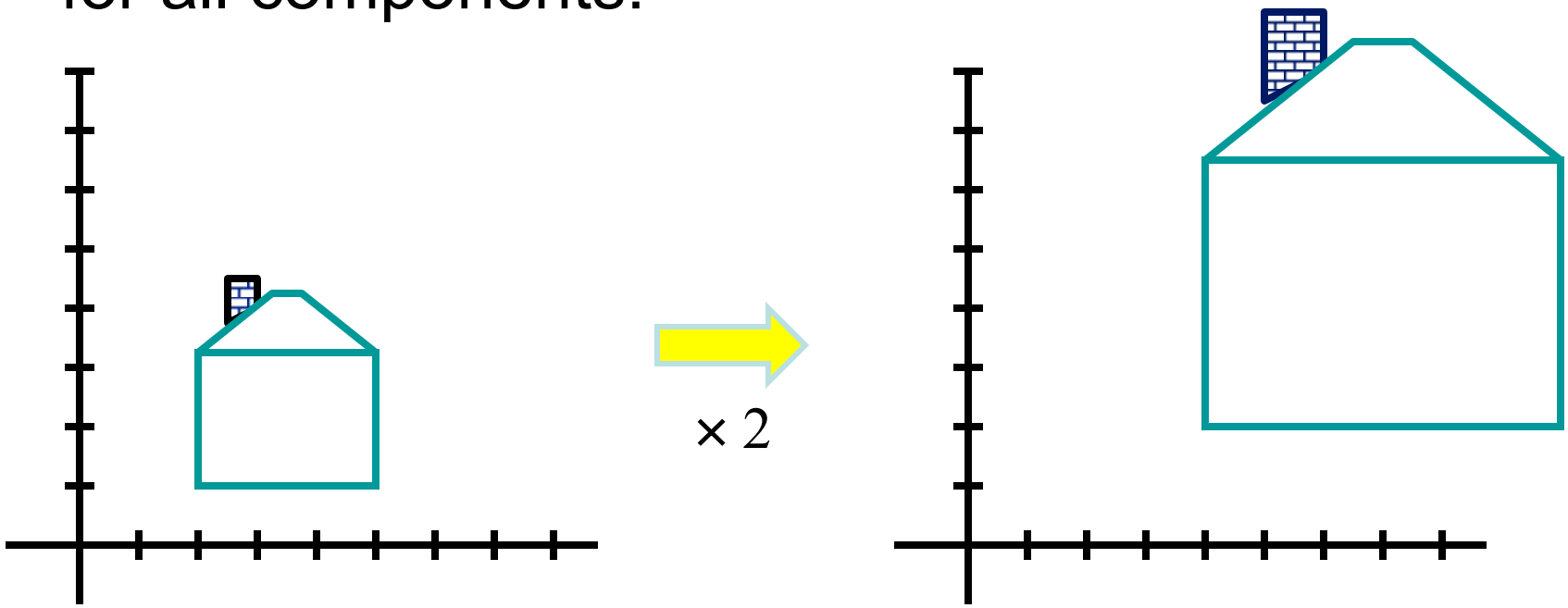
Types of Transformation



MATLAB functions: griddata, interp2, maketform, imtransform

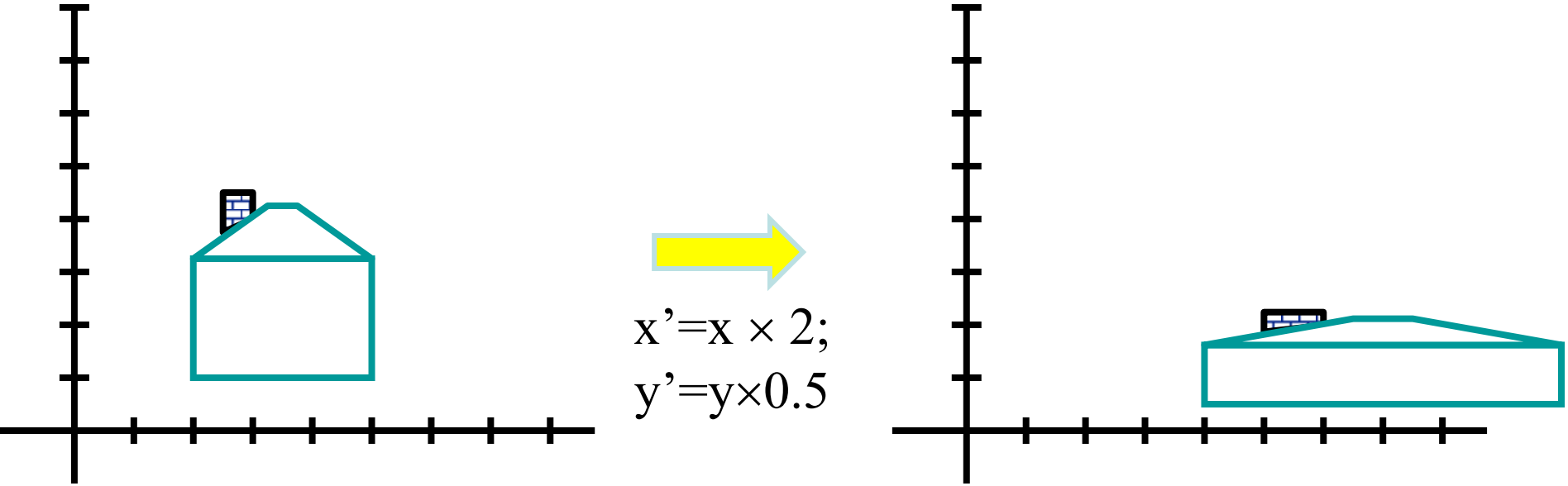
Scaling

- *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:



Scaling

- Scaling operation: $x' = ax$

  $y' = by$

- Or, in matrix form:

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

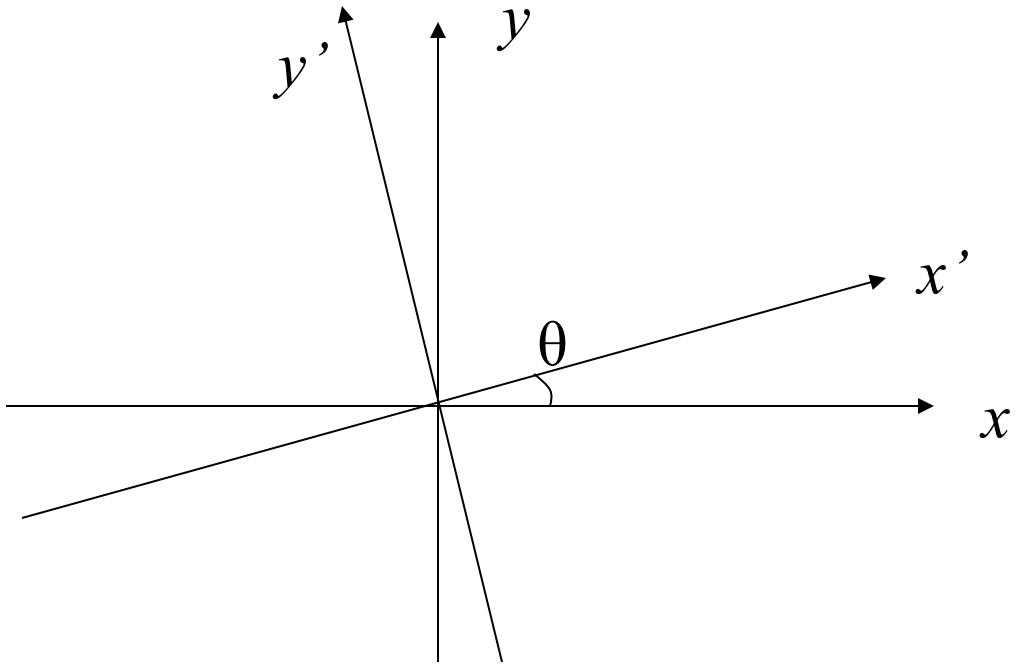
What's inverse of S?

Scaling

a=1/2

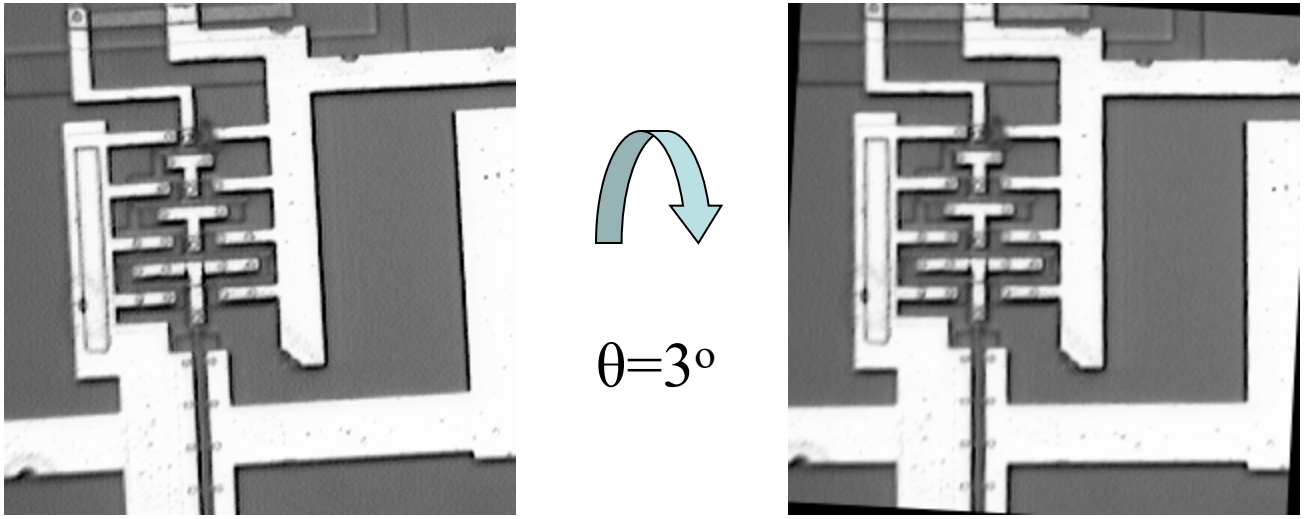
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & 1/a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation

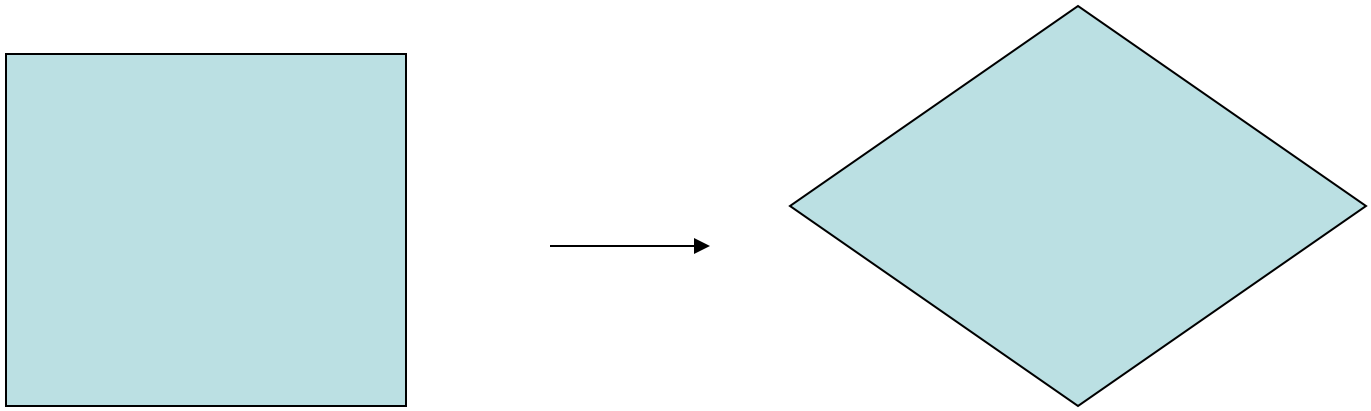


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation Example



Affine Transform



square

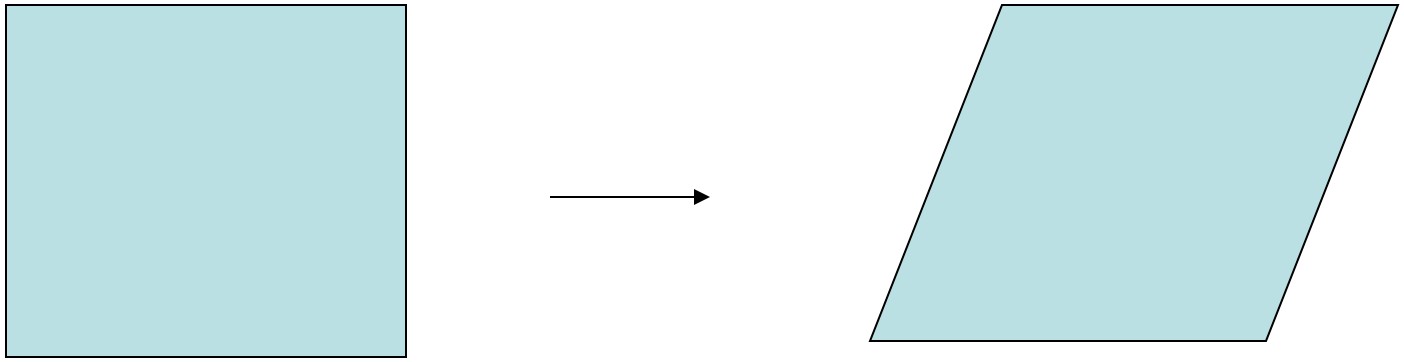
parallelogram

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

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

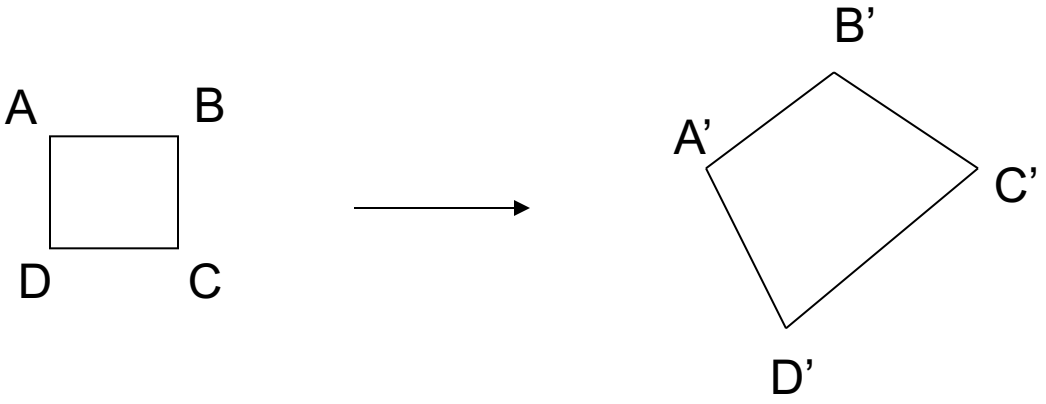
parallelogram

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

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

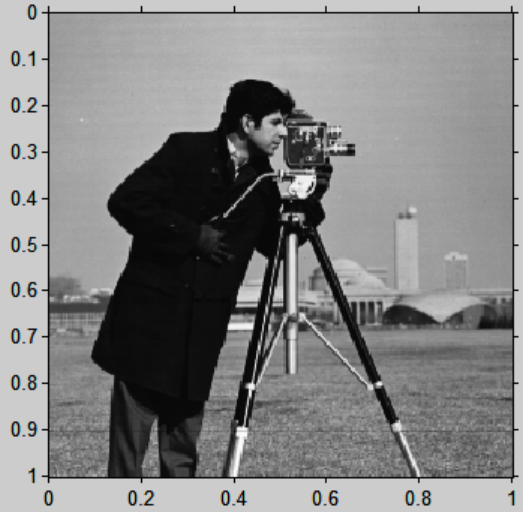


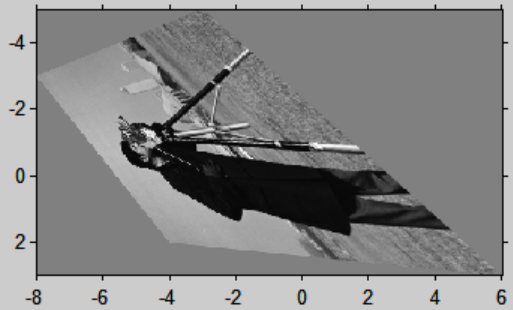
square quadrilateral

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

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

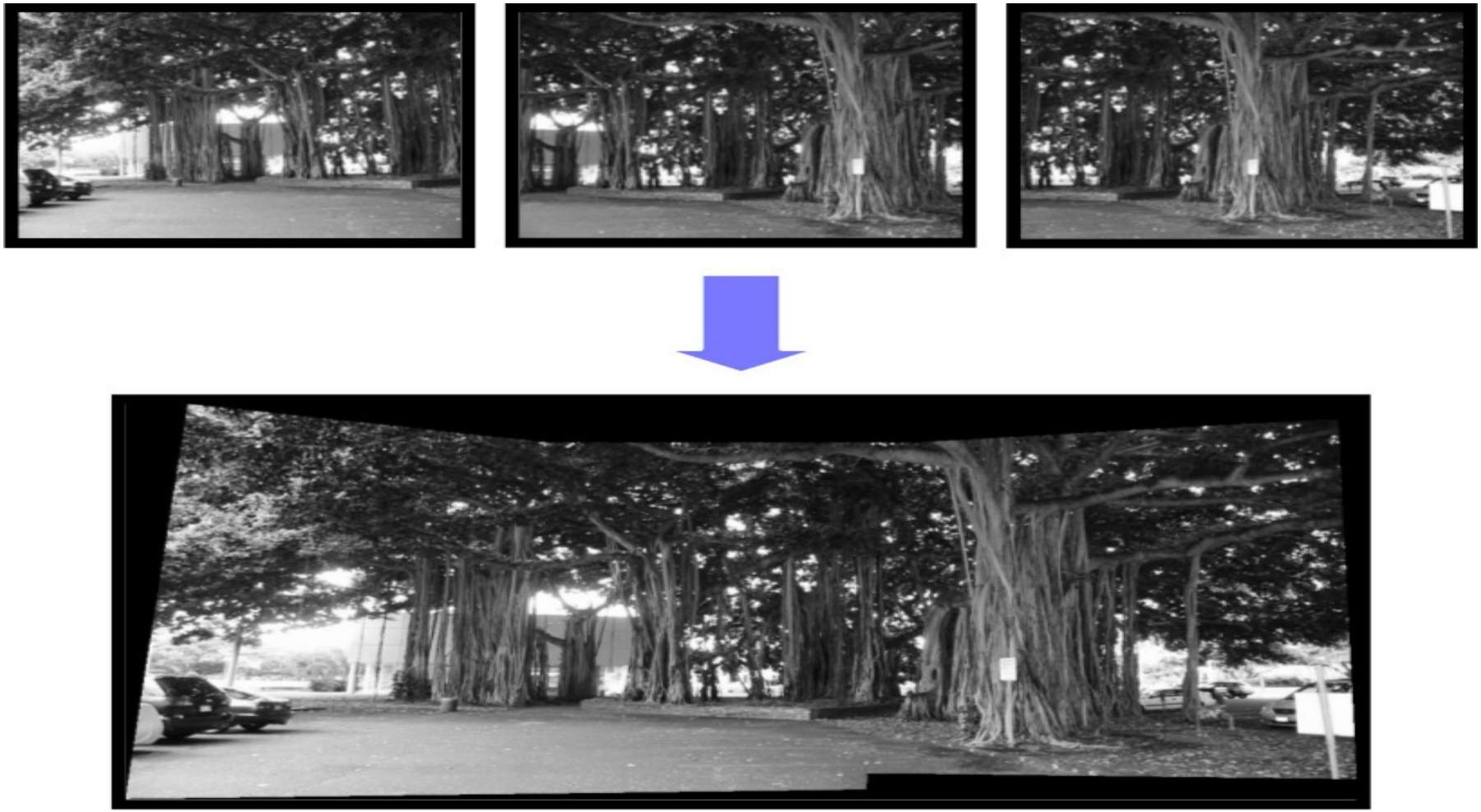
Projective Transform Example





[0 0; 1 0; 1 1; 0 1] ⟶ [-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

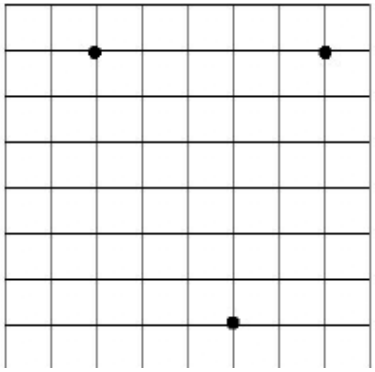
Target Image: this image is obtained by transforming the reference image.

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

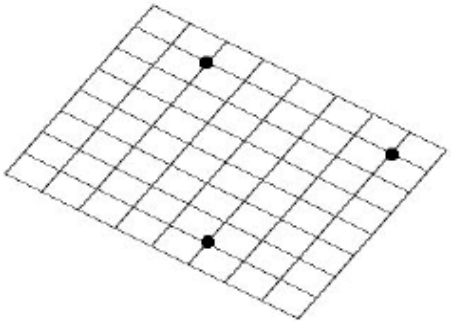
(x',y'): (or [u,v]) coordinates of points in the target image

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.

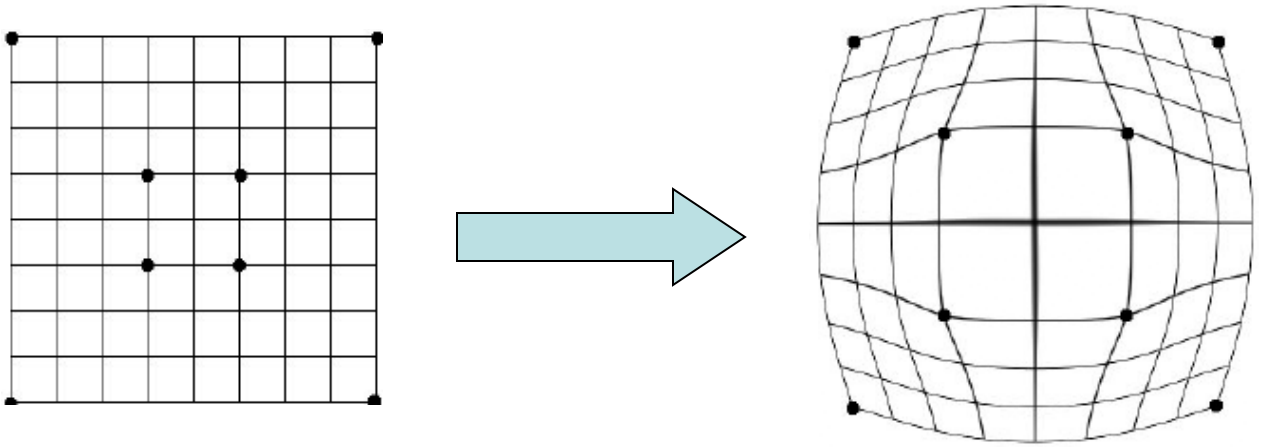


Source Image



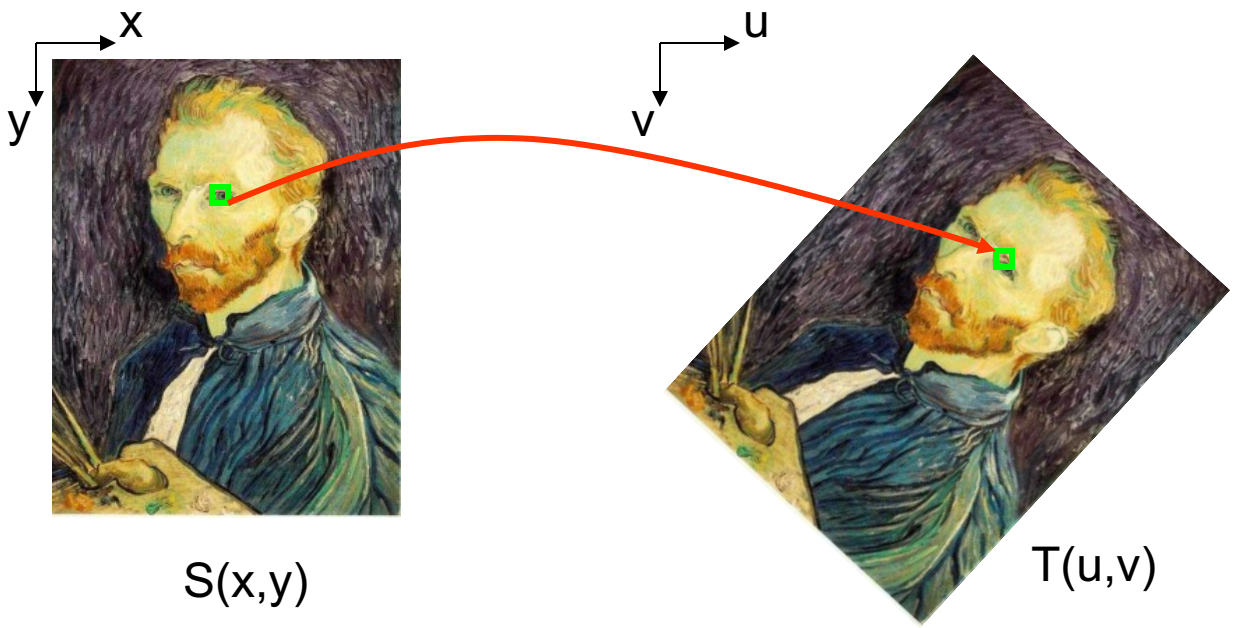
Target Image

Transformation Function



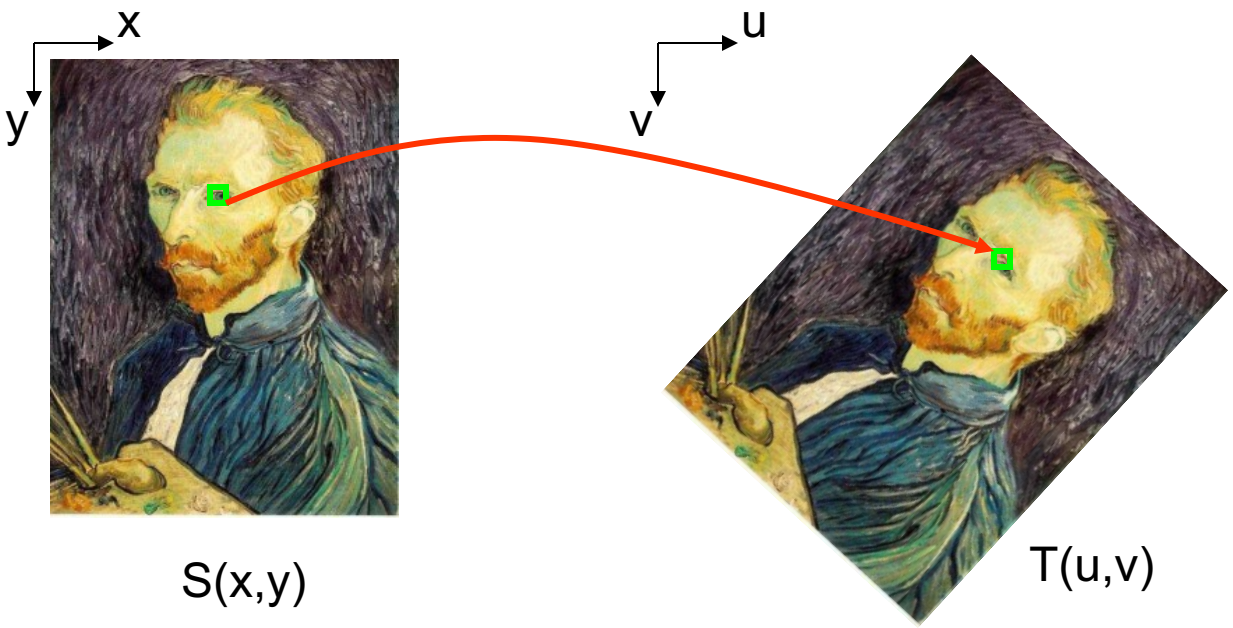
Transform the domain of an image to a desired geometry

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:

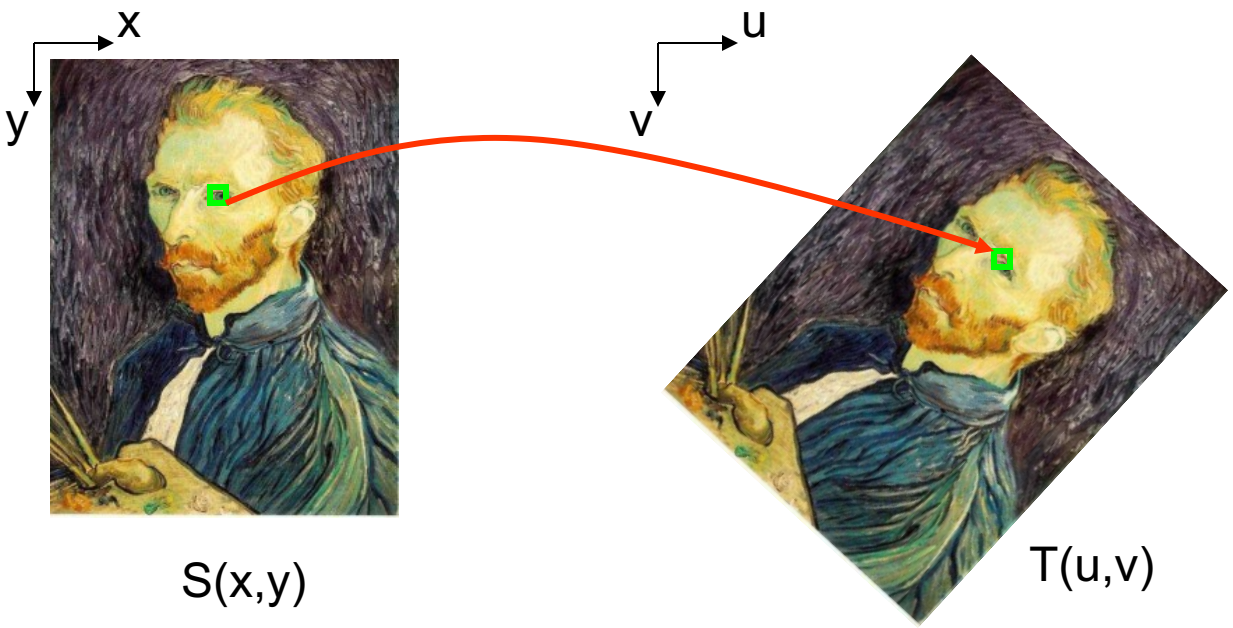
for $y = y_{min}$ to y_{max}

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

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

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

Forward Warping



Forward warping algorithm:

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

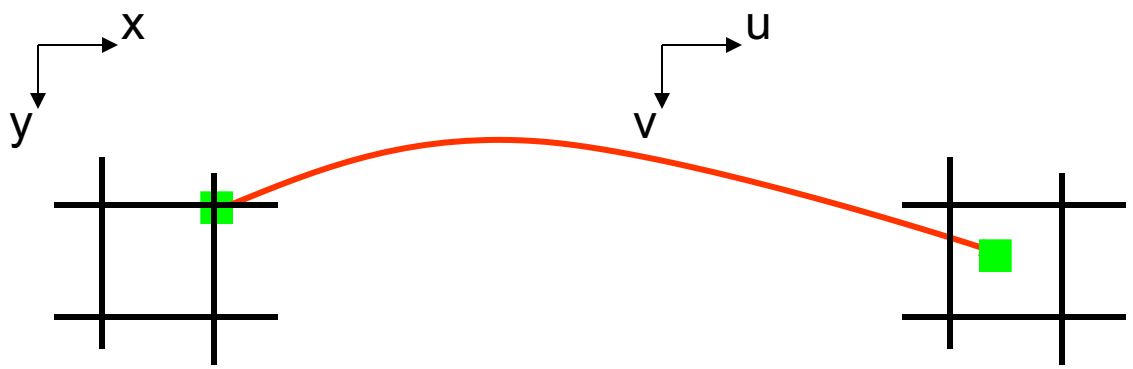
  for $x = x_{min}$ to x_{max}

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

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

- Any problems for forward warping?

Forward Warping

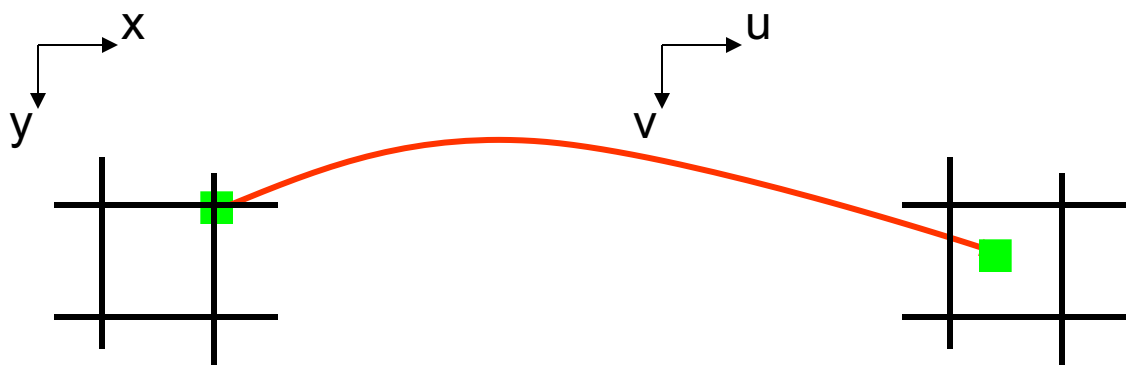


S(x,y)

T(u,v)

Q: What if the transformed pixel located between pixels?

Forward Warping



S(x,y)

T(u,v)

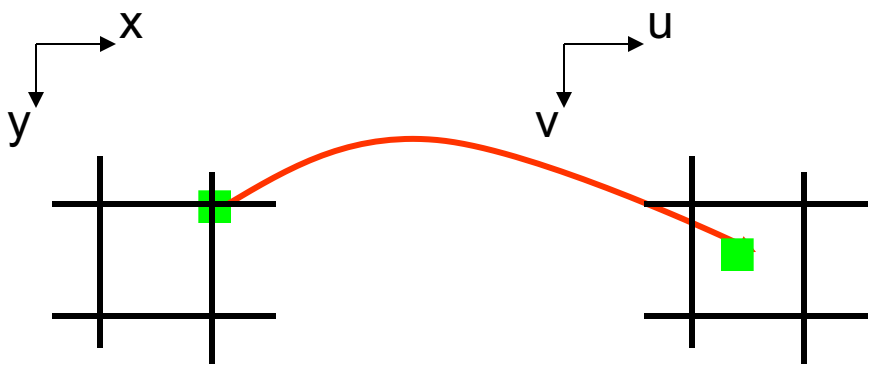
Q: What if the transformed pixel located between pixels?

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}

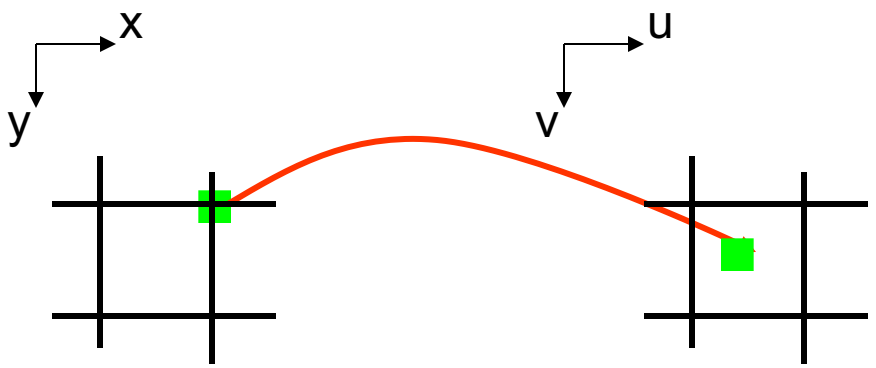
u = f(x,y); v = g(x,y)

copy pixel 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}

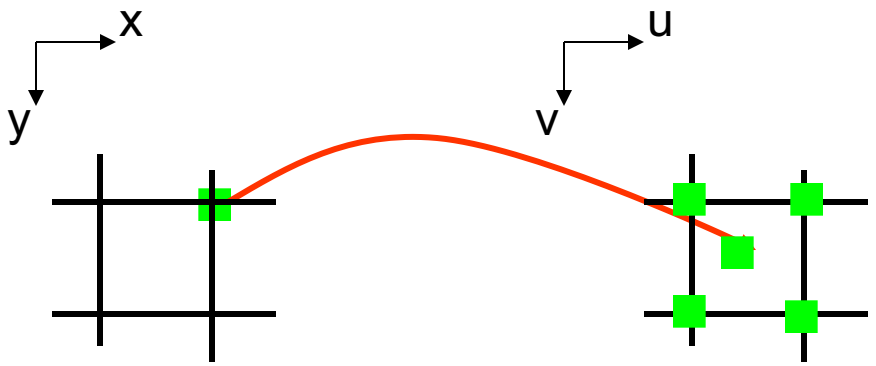
u = f(x,y); v = g(x,y)

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?



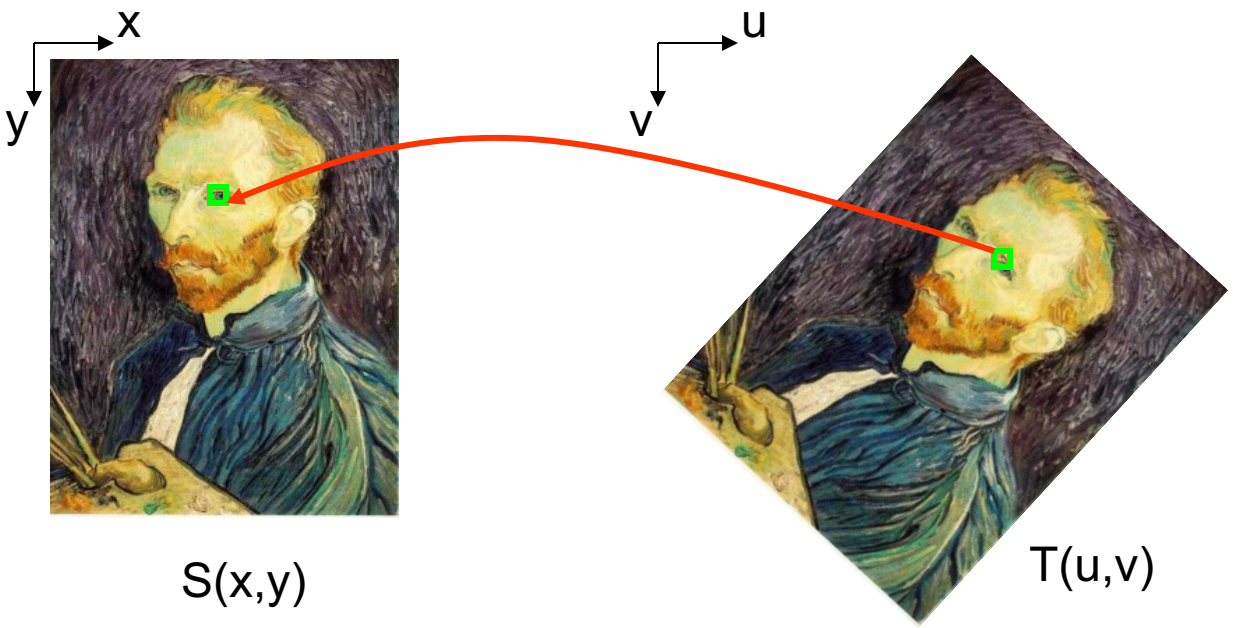
for y = y_{min} to y_{max}

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:

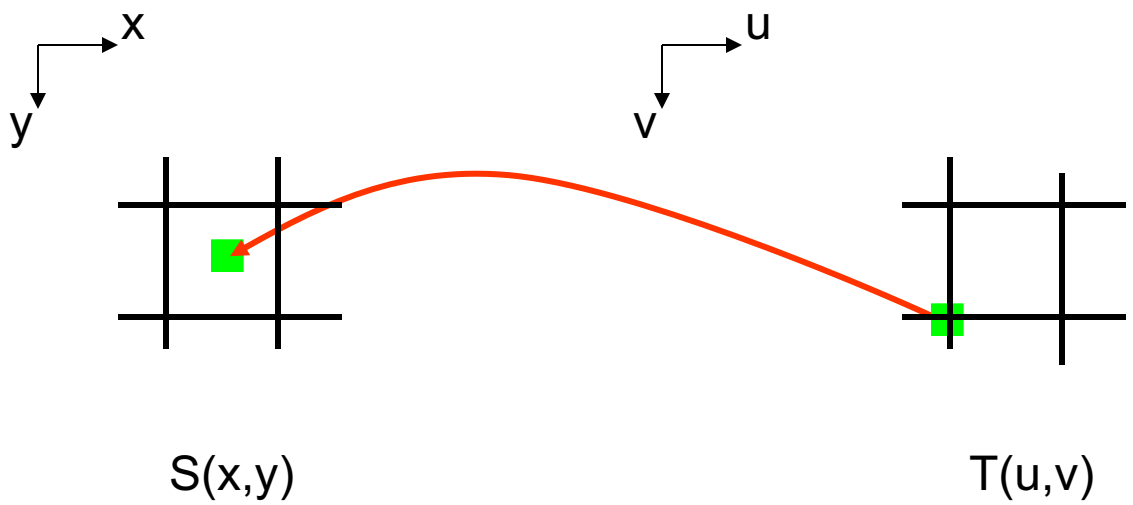
for v = v_{min} to v_{max}

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)

Inverse Warping



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

A: Interpolate color values from neighboring pixels

Inverse Warping

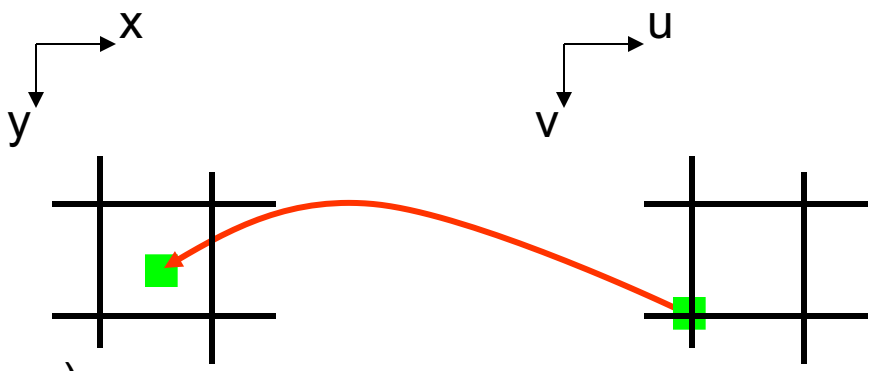
- Iterate over dest., finding pixels from source
- Non-integer evaluation source image, resample
- May oversample source
- But no holes
- Simpler, better than forward mapping

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 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:

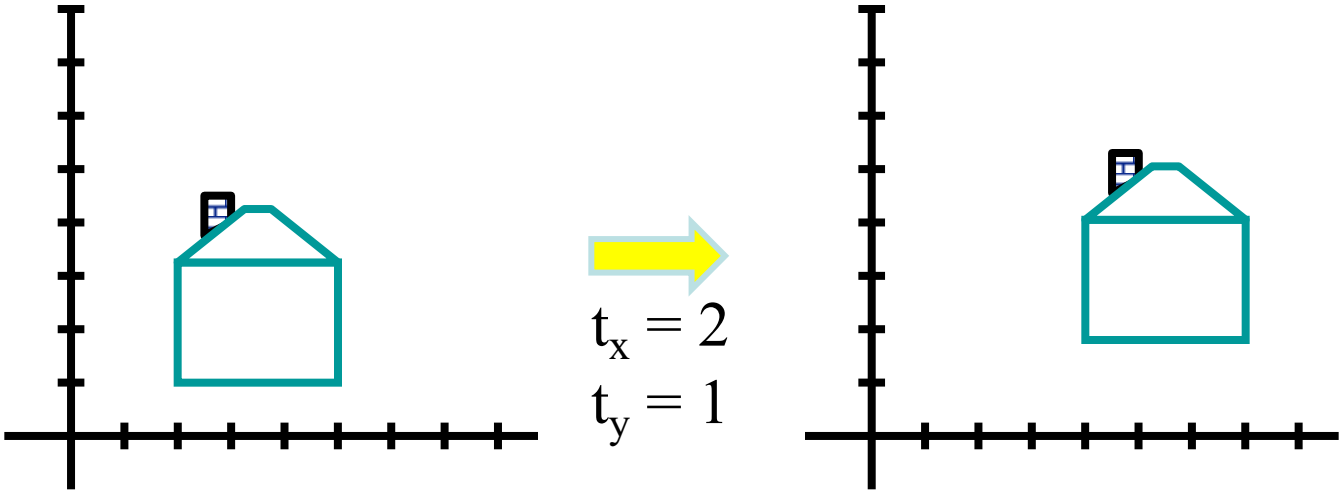
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation

- Example of translation

Homogeneous Coordinates

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



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

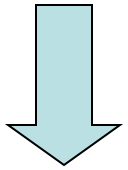
$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

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

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

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

Projective Transformations

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

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

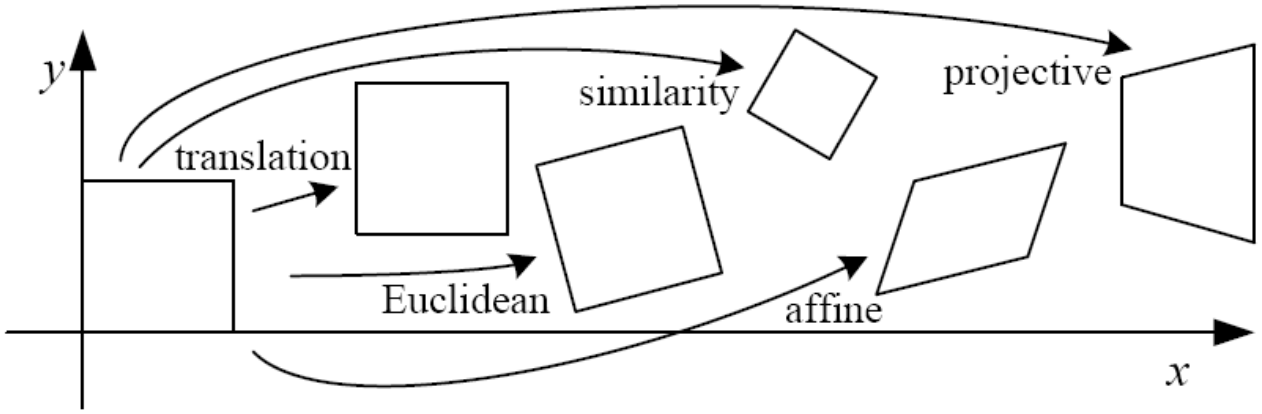
- Projective transformations …
  - Affine transformations, and
  - Projective warps
- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition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

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

Perspective effects

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

Hierarchy of 2D image transformations (stratification)

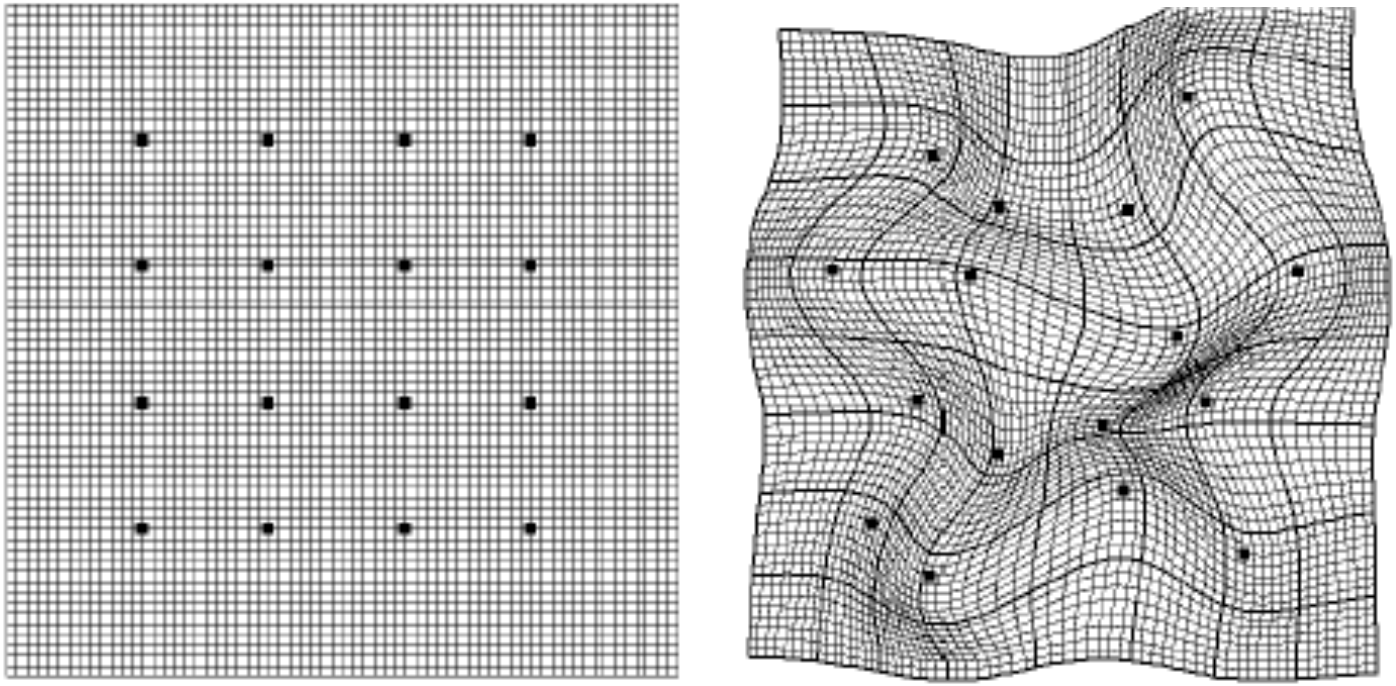


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times3}$	8	straight lines	

These transformations are a nested set of groups

- Closed under composition and inverse is a member

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.