Computer Vision

IFT6758 - Data Science

Sources:

http://www.cs.cmu.edu/~16385/

http://cs231n.stanford.edu/2018/syllabus.html

http://www.cse.psu.edu/~rtc12/CSE486/

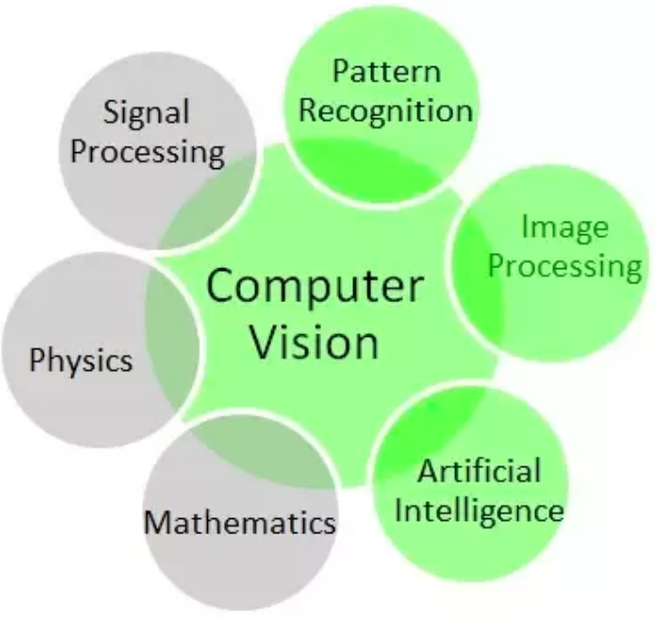
Mila

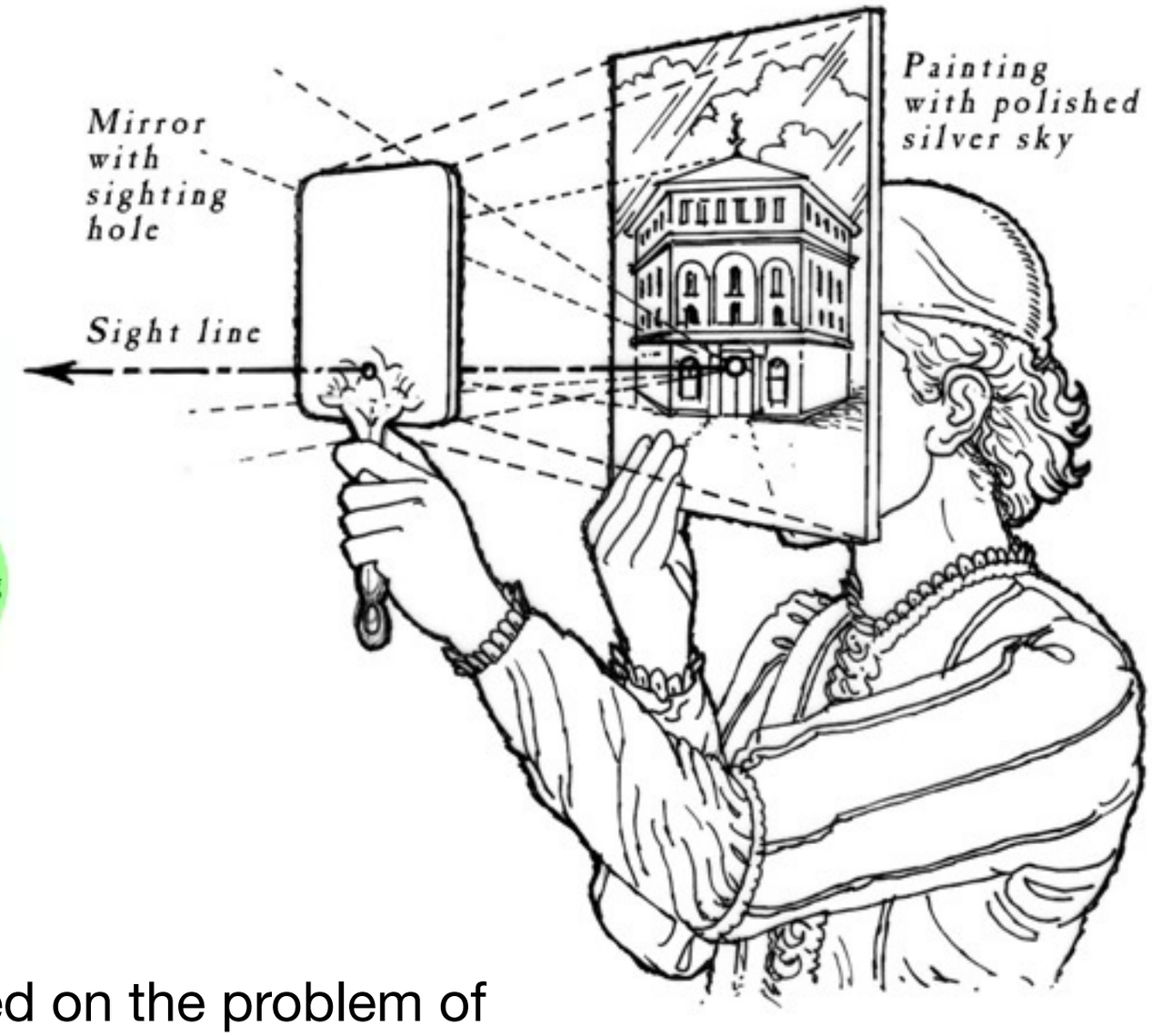
Université de Montréal

What is Computer vision?

“
Vision is the act of knowing what is where by looking.
”

--Aristotle





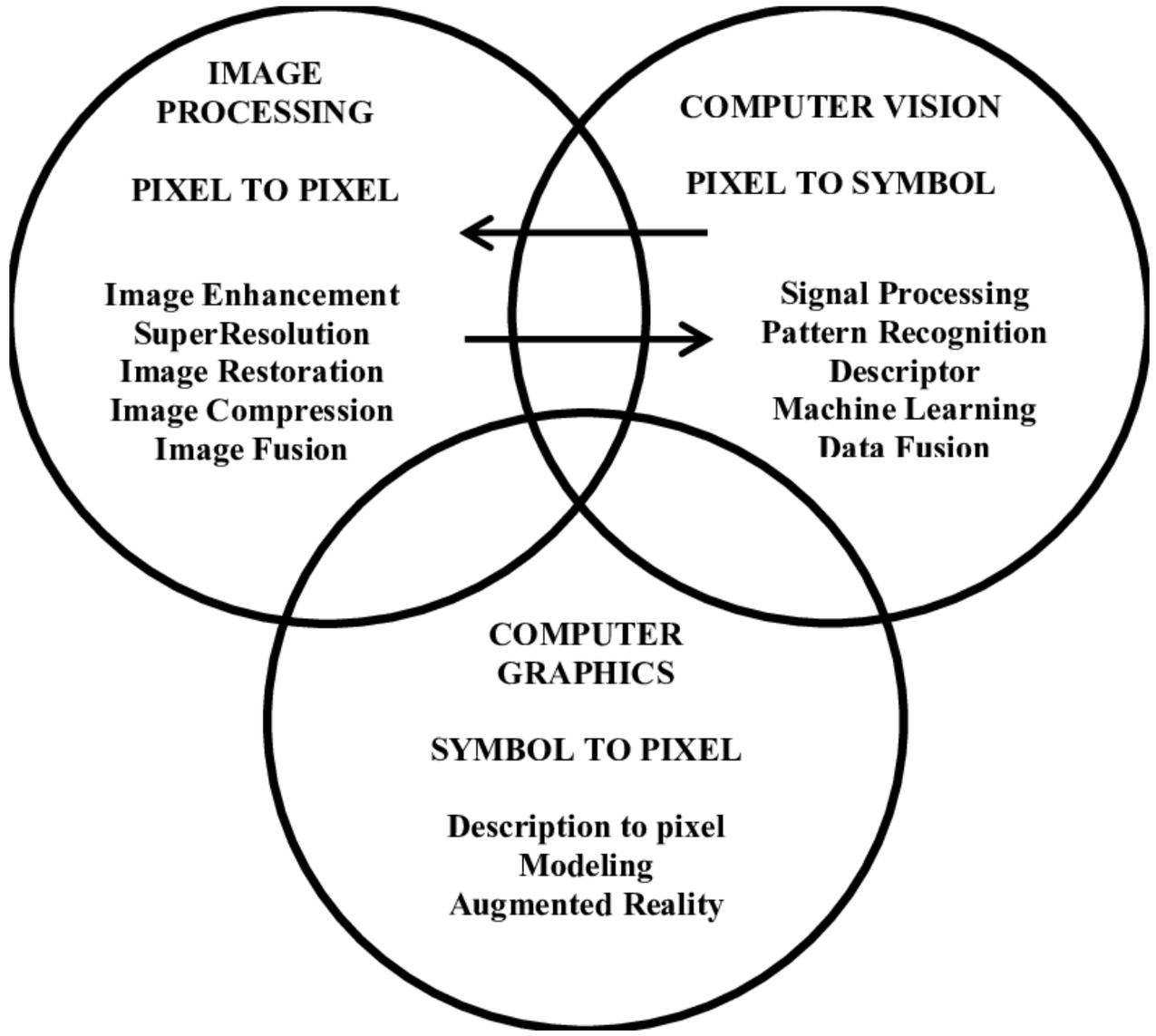
- Computer vision is a field of study focused on the problem of helping computers to see.

Computer vision vs. Image processing

- Computer vision is distinct from image processing.

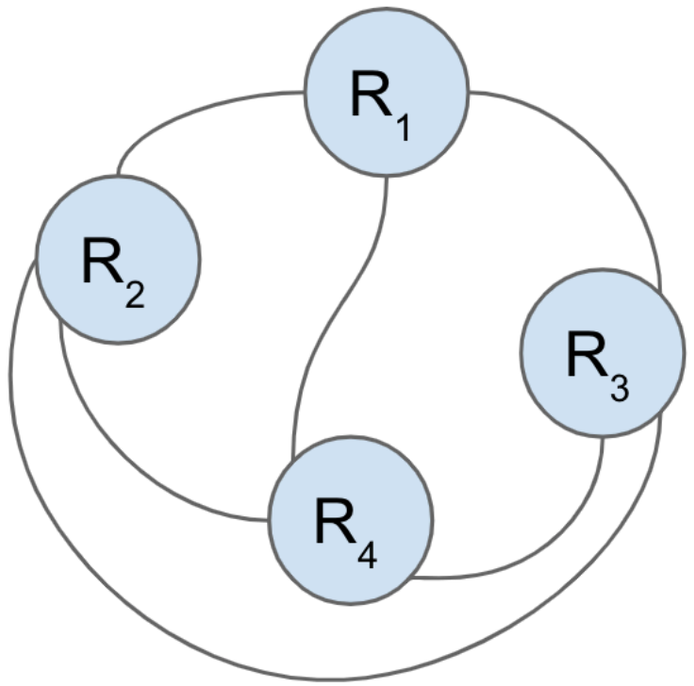
- Image processing is the process of **creating a new image** from an existing image, typically simplifying or enhancing the content in some way.

- Computer vision is concerned with understanding the content of an image.



CV tasks (4 Rs)

1. **R**econstruction
2. **R**egistration
3. **R**eorganization
4. **R**ecognition



CV tasks (4 Rs)

1.**R**econstruction

Left View

Right View

Multiview Geometry, 3D Vision, Shape-from-X

CV tasks (4 Rs)

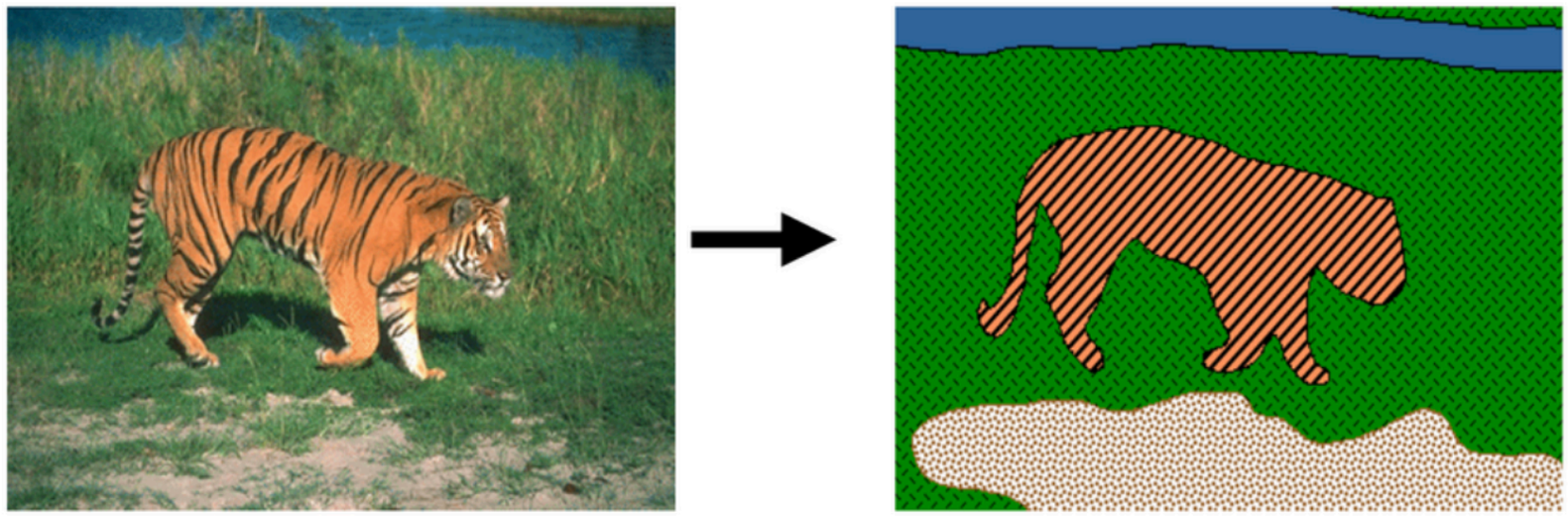
2.**R**egistration

Tracking, Alignment, Optical Flow, Correspondence

CV tasks (4 Rs)

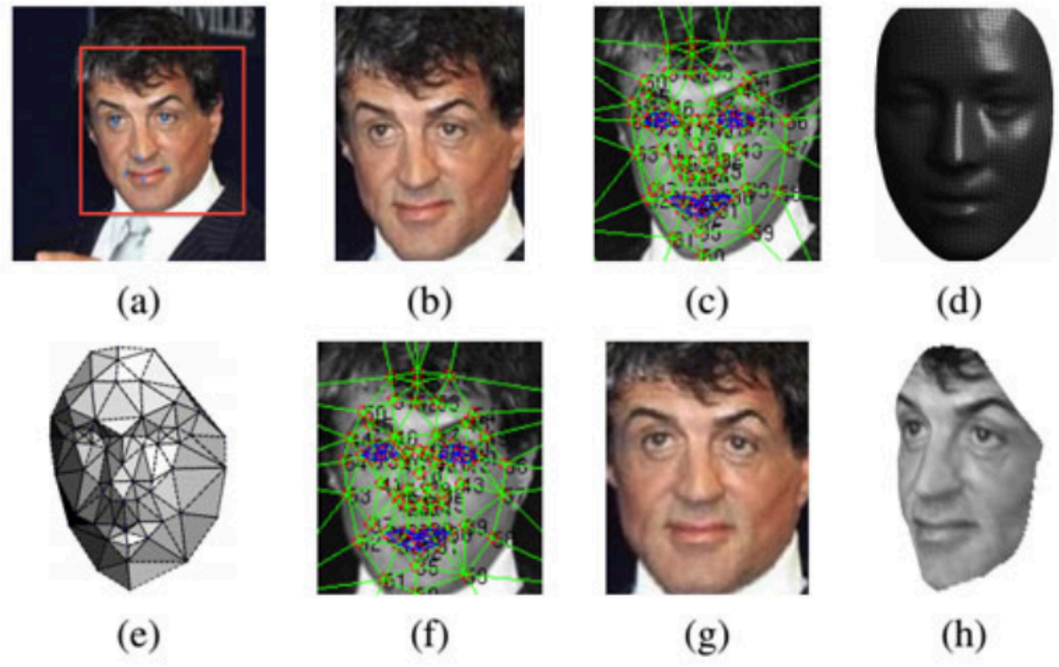
Clustering, Unsupervised Learning, Segmentation, Perceptual Organization

3.**R**eorganization



CV tasks (4 Rs)

[Ricoh]



[DeepFace]

4.**R**ecognition

Verification, Identification, Detection

Why study Computer Vision?

- Images and movies are everywhere
- Fast-growing collection of useful applications
    - building representations of the 3D world from pictures
    - automated surveillance (who's doing what)
    - movie post-processing
    - face finding
- Greater understanding of human vision

Earth view (3d modelling)

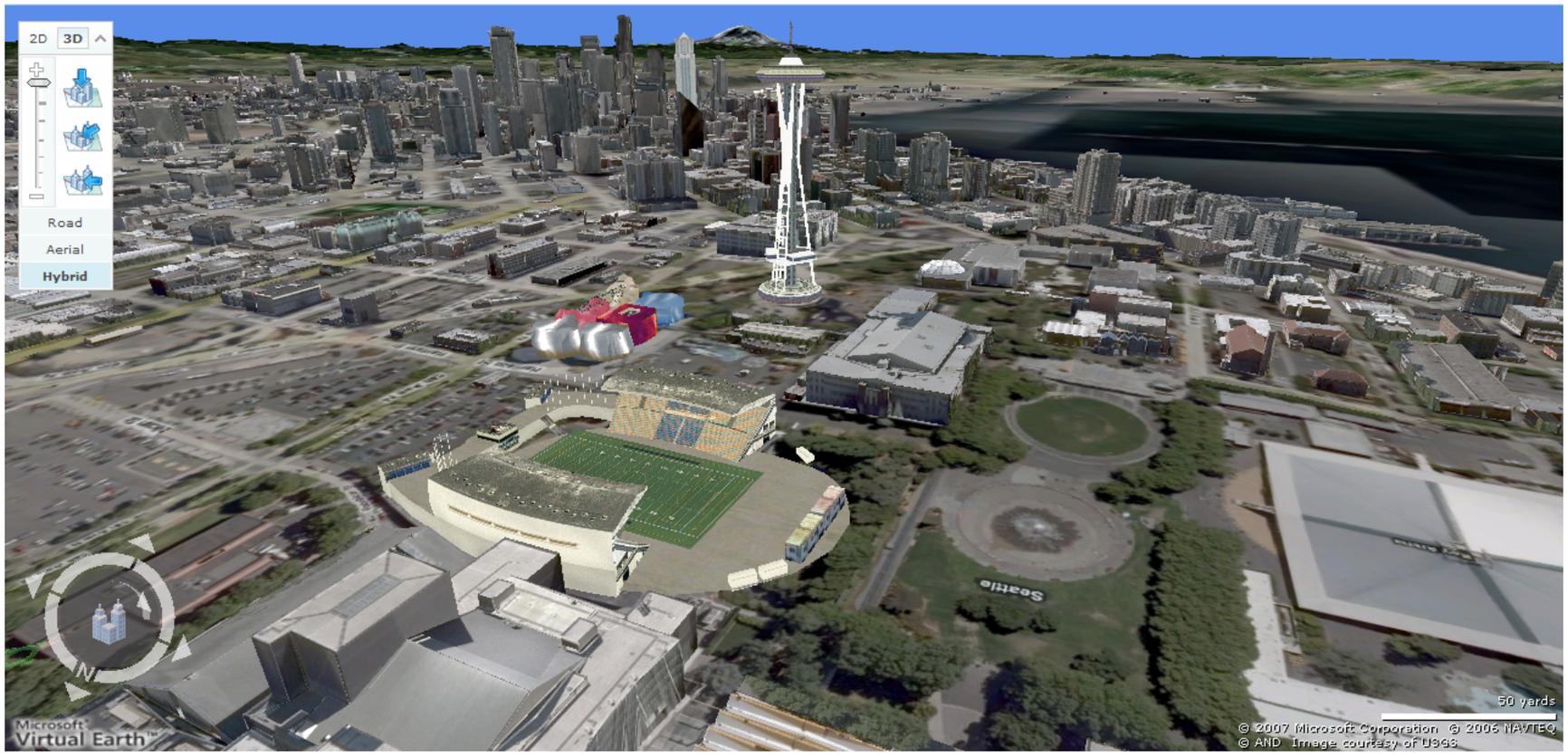
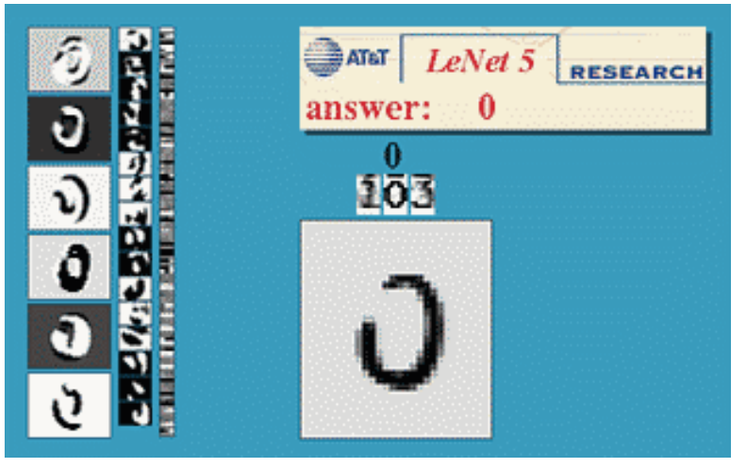


Image from Microsoft's Virtual Earth
(see also: Google Earth)

Optical character recognition (OCR)

Technology to convert scanned docs to text

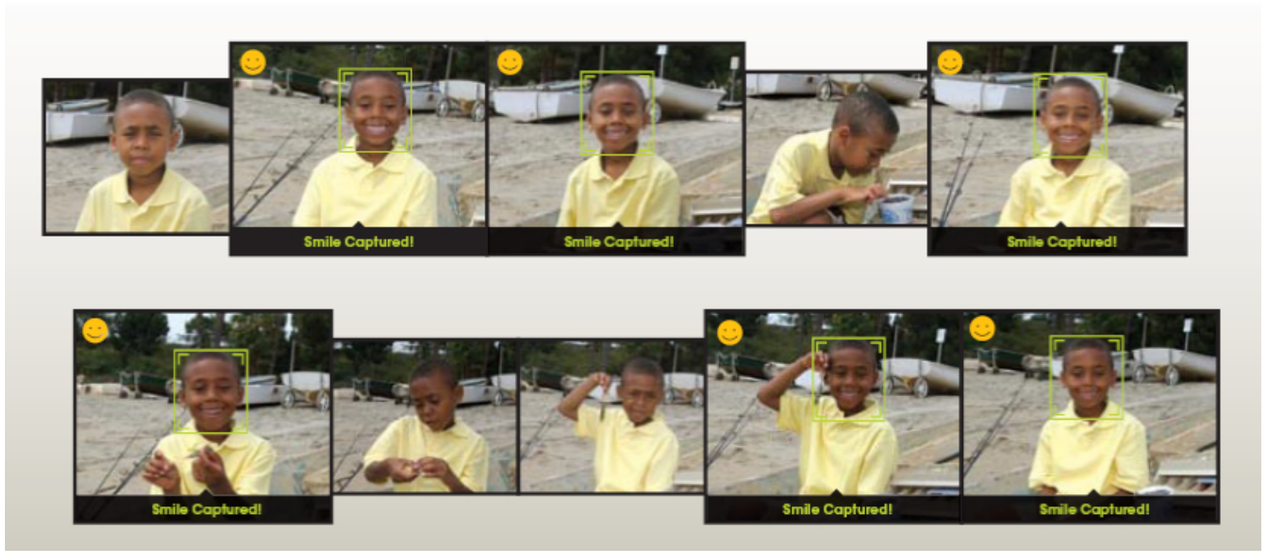
- If you have a scanner, it probably came with OCR software



Digit recognition, AT&T labs
http://www.research.att.com/~yann/

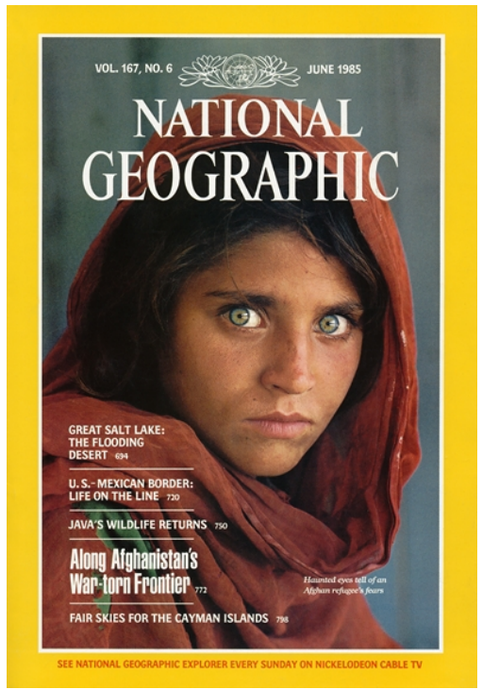
License plate readers
http://en.wikipedia.org/wiki/Automatic_number_plate_recognition

Face and smile detection



Many new digital cameras now detect faces

- Canon, Sony, Fuji, …



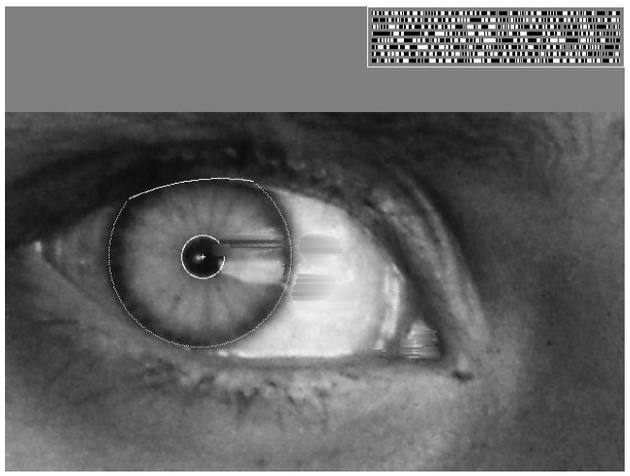
Who is she?

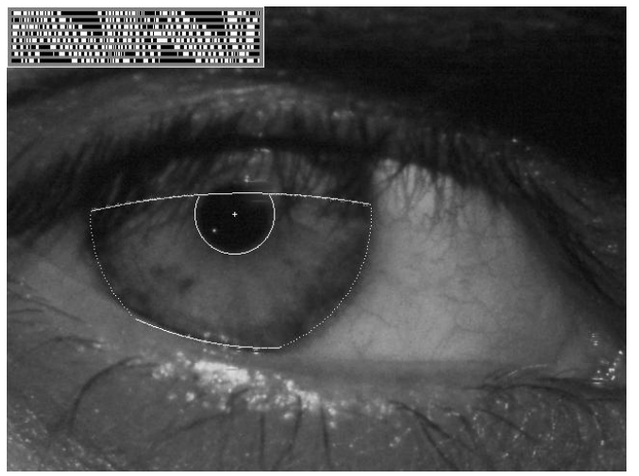
[Sony Cyber-shot® T70 Digital Still Camera]

Mila

Université de Montréal

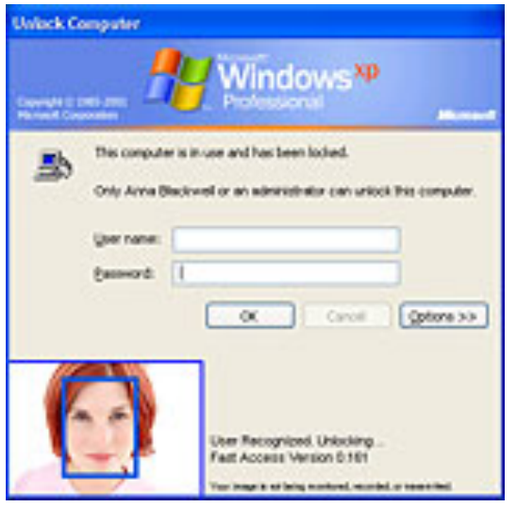
Vision biometric





"*How the Afghan Girl was Identified by Her Iris Patterns*" Read the story

Fingerprint scanners on many new laptops, other devices



Face recognition systems now beginning to appear more widely
http://www.sensiblevision.com/

Object recognition

[LaneHawk by EvolutionRobotics]
"A smart camera is flush-mounted in the checkout lane, continuously watching for items… "

This is becoming real:

- Microsoft Research
- Point & Find, Nokia

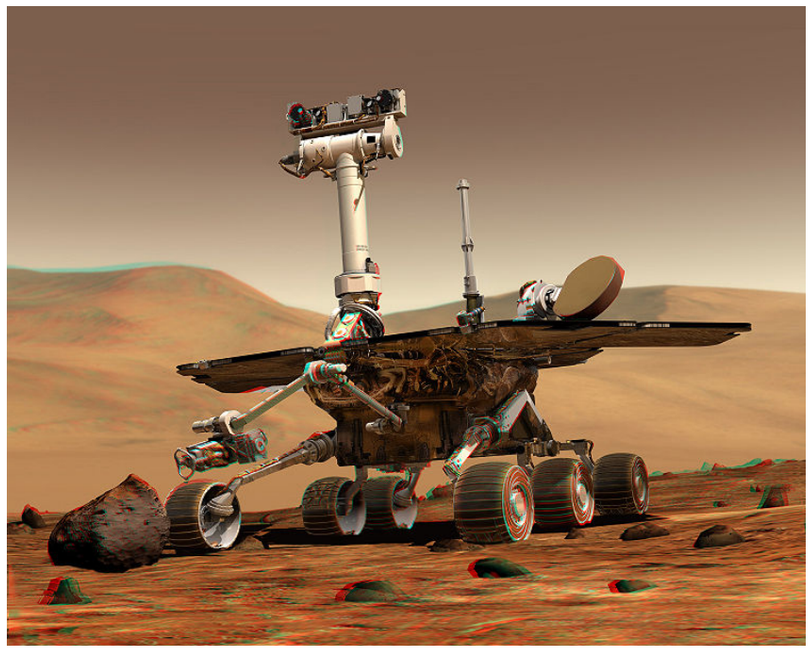
Sports and games

Sportvision first down line
Nice explanation on www.howstuffworks.com

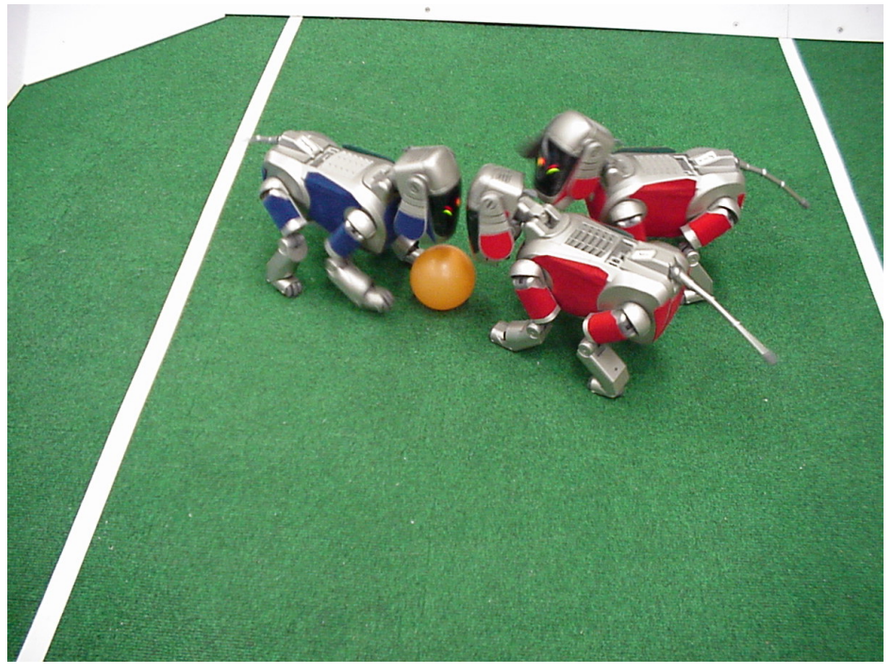
Digimask: put your face on a 3D avatar.

Nintendo Wii has camera-based IR tracking built in.

Robotics

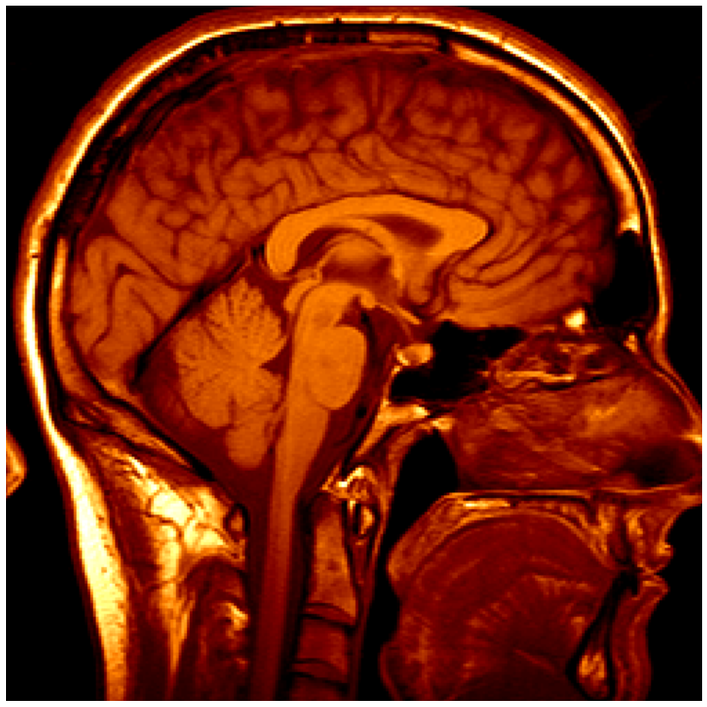


NASA's Mars Spirit Rover
http://en.wikipedia.org/wiki/Spirit_rover



http://www.robocup.org/

Medical imaging



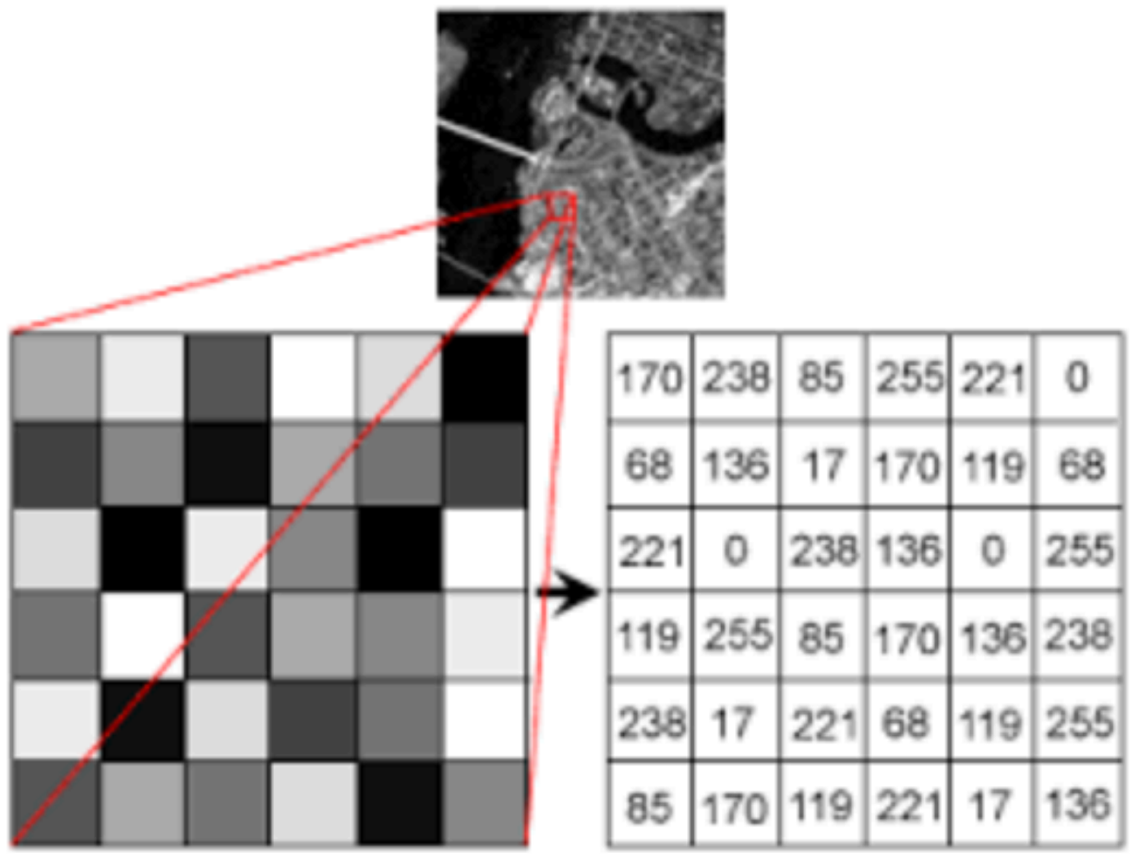
3D imaging
MRI, CT

Image guided surgery
Grimson et al., MIT

CV Challenges

How machines see an image?

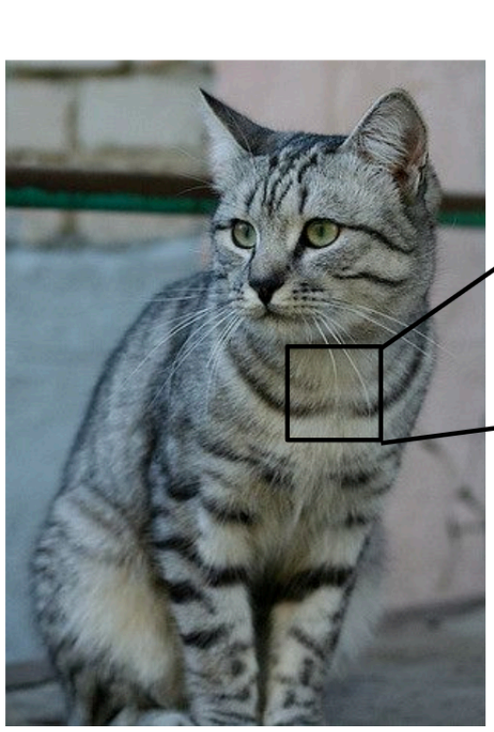
- Machines see and process everything using numbers, including images and text. How do you convert images to numbers?



Image

- Every number represents the **pixel intensity** at that particular location. e.g., for a grayscale image where every pixel contains only one value i.e. the intensity of the black color at that location.

The Problem: Semantic Gap



This image by Nikita is licensed under CC-BY 2.0

```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

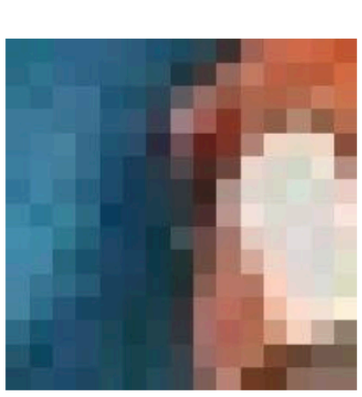
What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

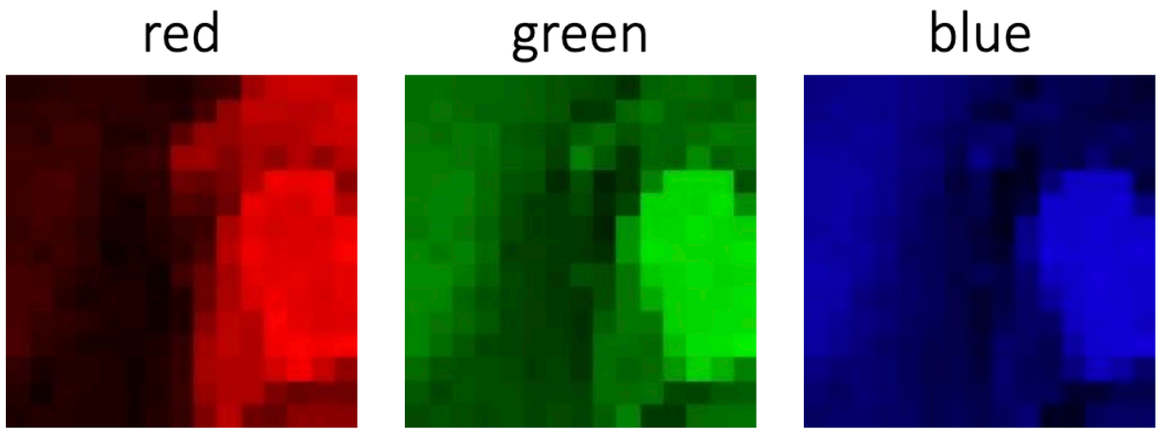
What is an image?

- Color images will have multiple values for a single pixel. These values represent the **intensity of respective channels – Red, Green and Blue channels for RGB images**, for instance.

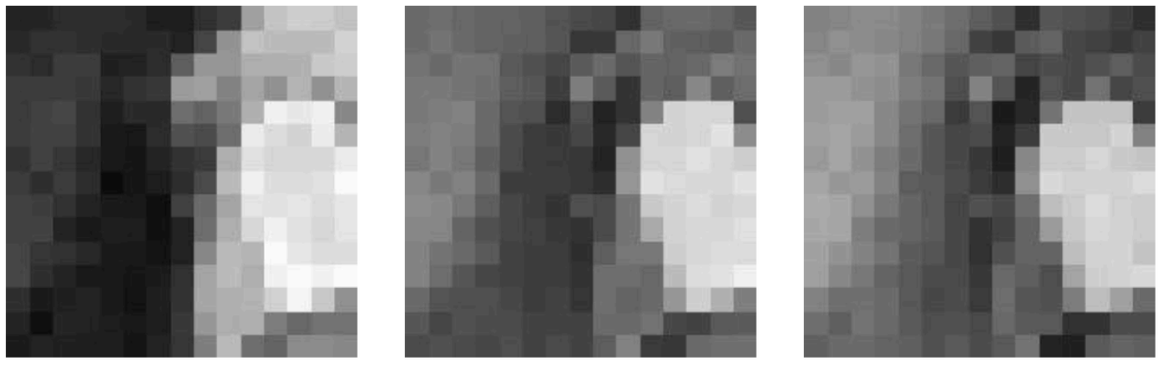


color image patch

How many bits are the intensity values?



colorized for visualization

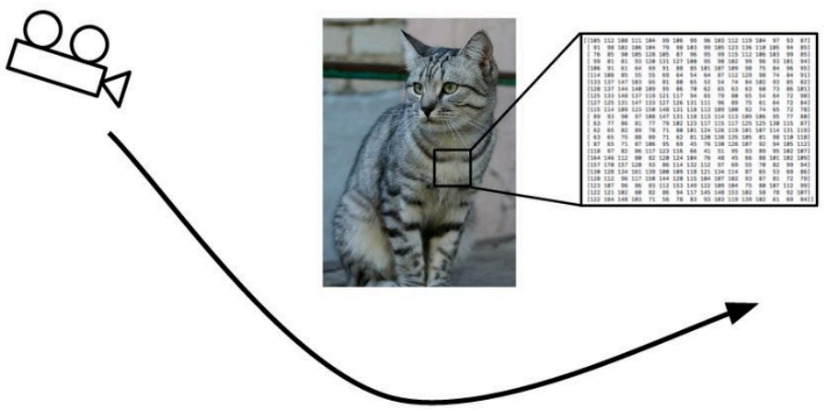


actual intensity values per channel

Each channel is a 2D array of numbers.

Challenges of recognition

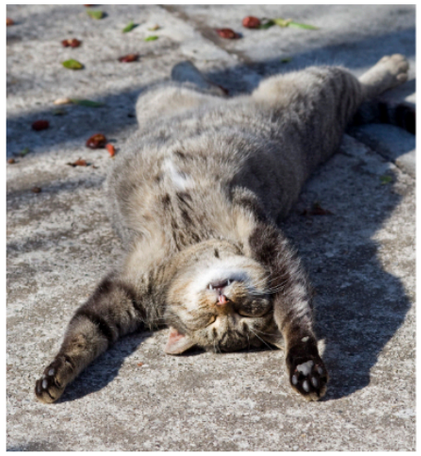
Viewpoint



Illumination

This image is CC0 1.0 public domain

Deformation



This image by Umberto Salvagnin is licensed under CC-BY 2.0

Occlusion

This image by jonsson is licensed under CC-BY 2.0

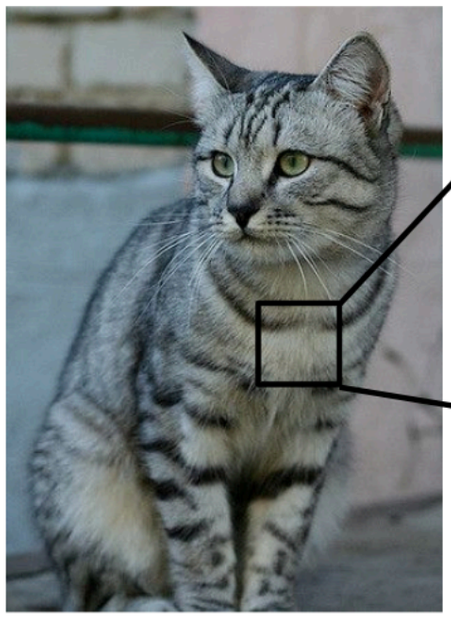
Clutter

This image is CC0 1.0 public domain

Intraclass Variation

This image is CC0 1.0 public domain

Challenge: Viewpoint variation

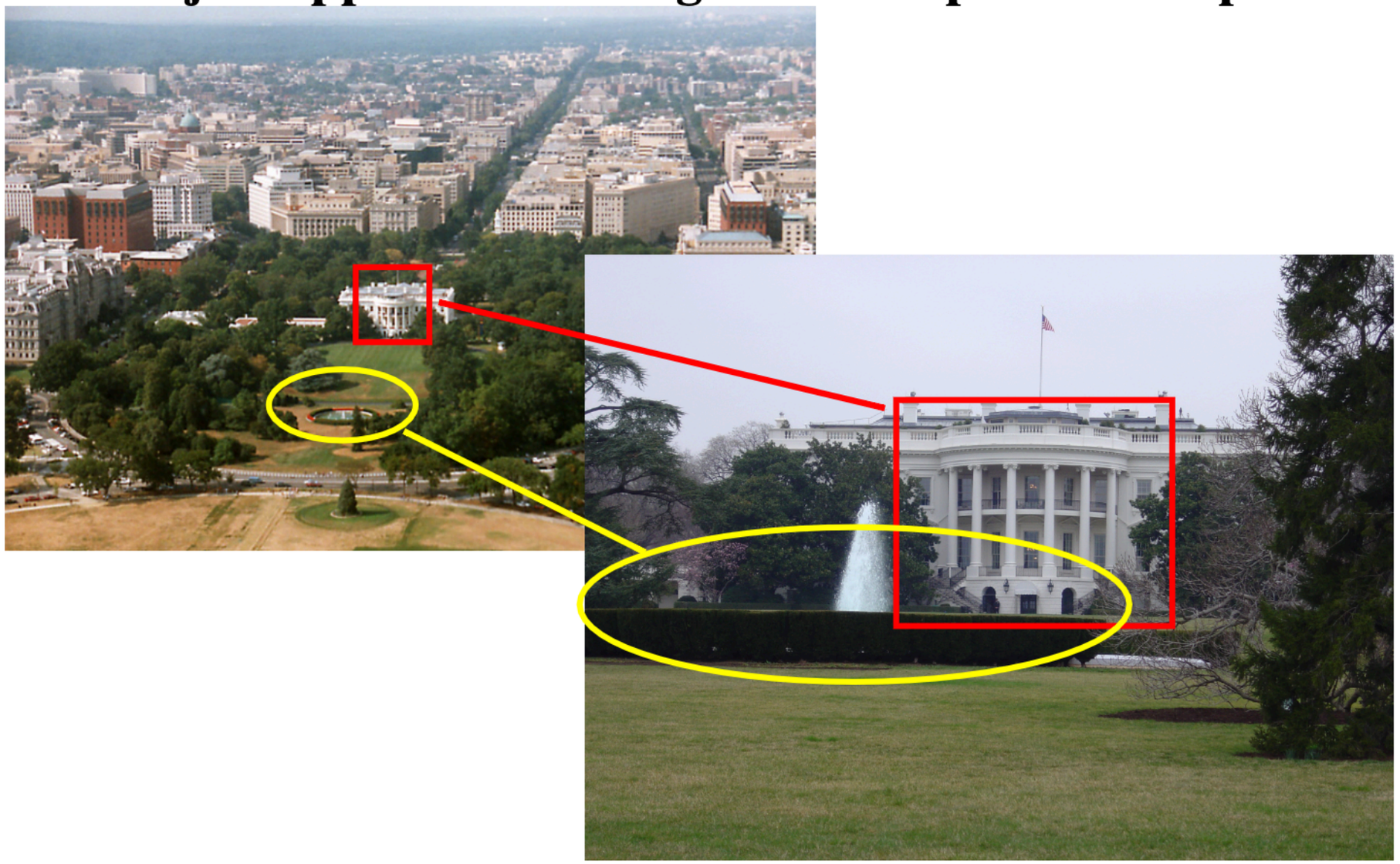


```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

All pixels change when the camera moves!

Challenge: Viewpoint variation

Object appearance changes with respect to viewpoint

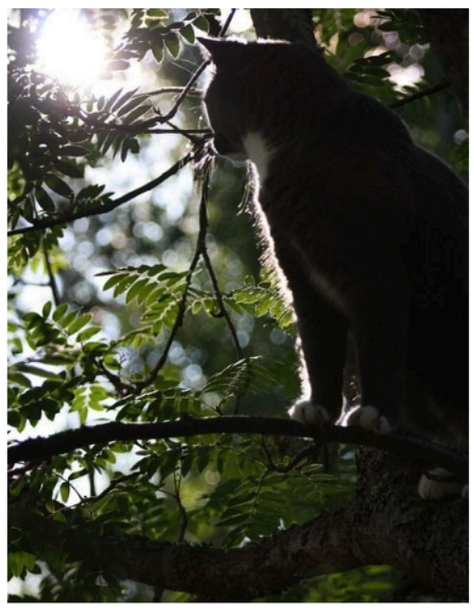


Challenge: Illumination

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

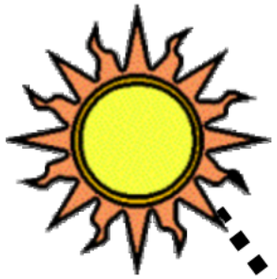


This image is CC0 1.0 public domain

Object appearance changes with respect to lighting magnitude and direction.

What is Color?

Source emits photons



Photons travel in a straight line

Illumination

And then some reach an eye/camera and are measured.

Sensor Response

They hit an object. Some are absorbed, some bounce off in a new direction.

Surface Reflection

Color

- Color percepts are a composition of three factors (**illumination, surface reflectance, sensor response**)

- We can’t easily factor the color we see in the image to infer illumination and material (even if sensor properties are fixed and known).

Some things to think about:

- “red” typically means “appears red to a human observer under white light”.
- white objects appear red under a red light.
- nothing looks red if you are red/green color blind.

“normal” color perception

red/green color blind

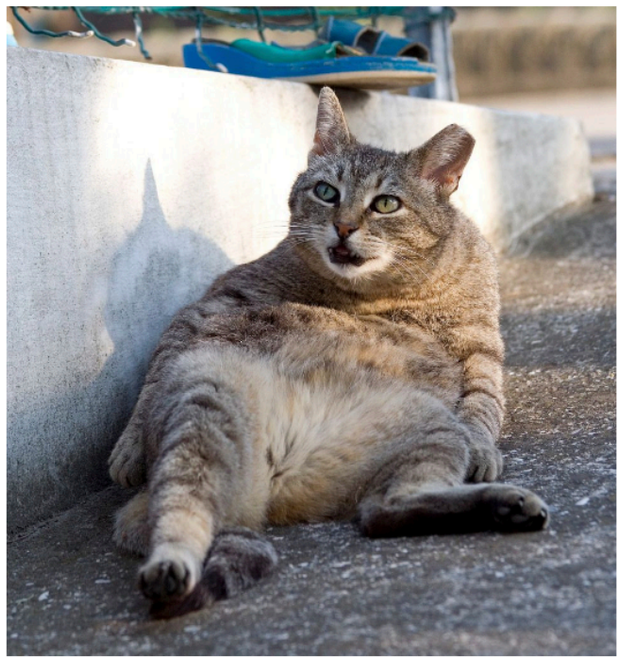
Is The Dress Blue and Black or White and Gold?

This dress manages to simultaneously gather more than 670,000 people on Buzzfeed, and convince 900,000 visitors to take a poll.

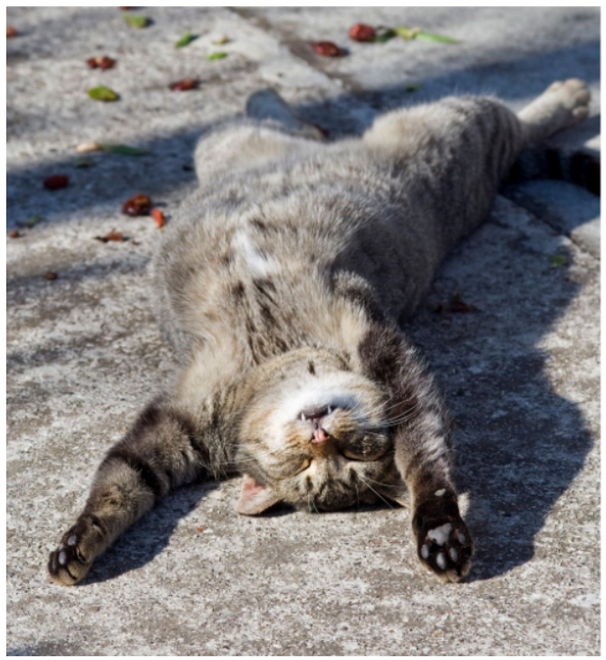
Challange: Color Constancy

Humans are very good at recognizing the same material colors under different illumination. Not clear how this is achieved in the general case.

Challenge: Deformation

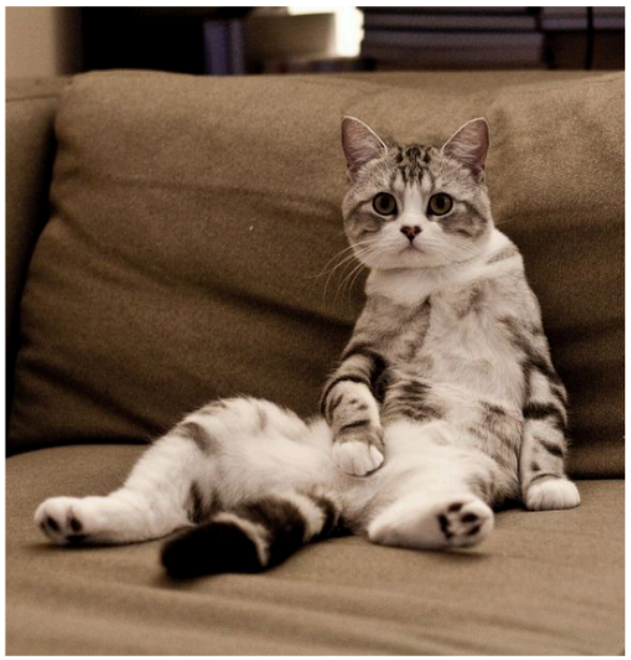


This image by Umberto Salvagnin is licensed under CC-BY 2.0



This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by sare bear is licensed under CC-BY 2.0



This image by Tom Thai is licensed under CC-BY 2.0

Challenge: Occlusion

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

Challenge: Variation

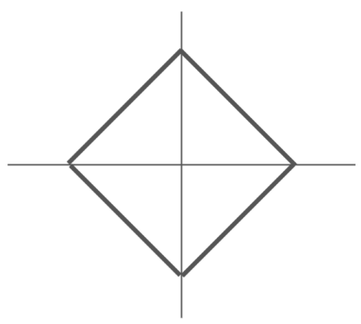
Distance Metric to compare images

L1 distance: $$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image					training image				pixel-wise absolute value differences				
56	32	10	18	-	10	20	24	17	= 46	12	14	1	add → 456
90	23	128	133		8	10	89	100	82	13	39	33	
24	26	178	200		12	16	178	170	12	10	0	30	
2	0	255	220		4	32	233	112	2	32	22	108	

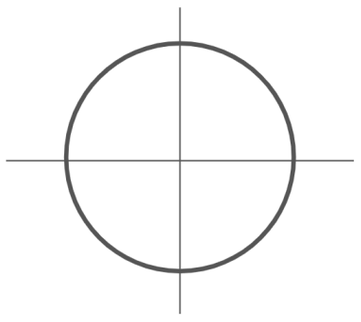
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



Distance metrics on pixels

Original image is CC0 public domain

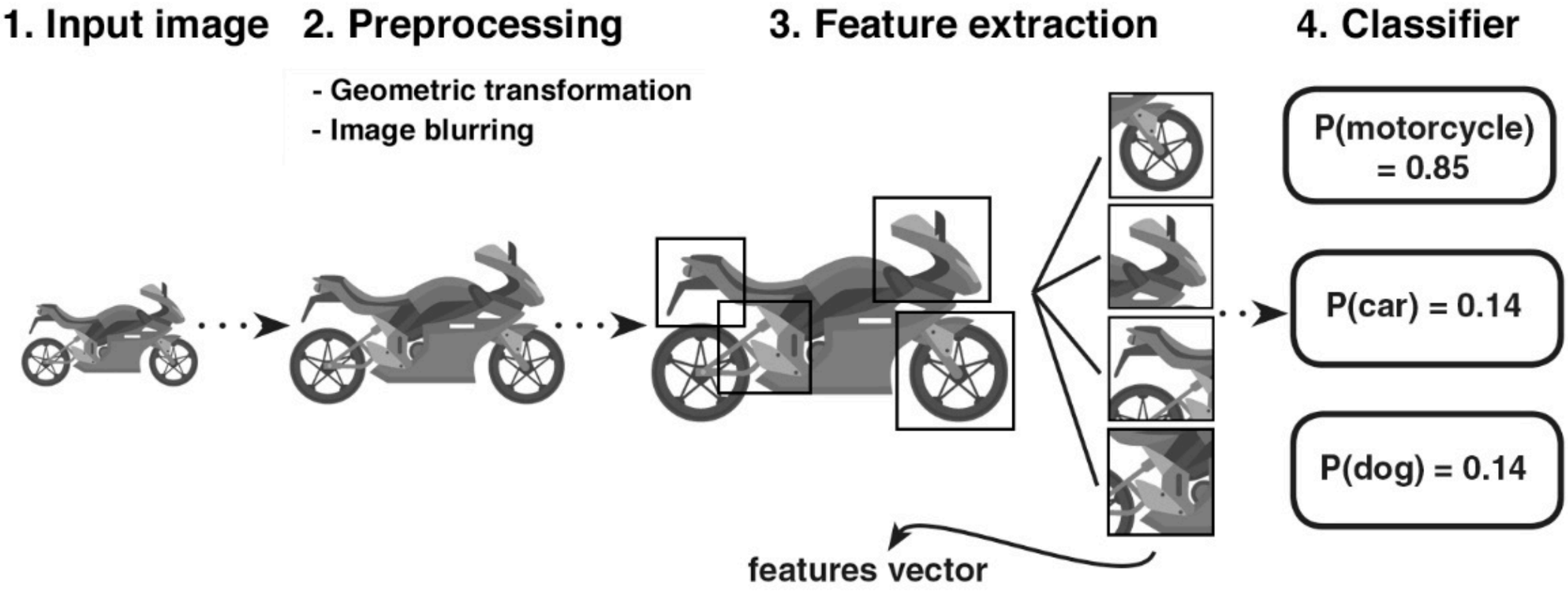
(all 3 images have same L2 distance to the one on the left)

k-Nearest Neighbor on images **never used.**

- Very slow at test time
- Distance metrics on pixels are not informative

CV main Operations

CV pipeline


1. Input image
2. Preprocessing
- Geometric transformation
- Image blurring
3. Feature extraction
4. Classifier
P(motorcycle) = 0.85
P(car) = 0.14
P(dog) = 0.14
features vector

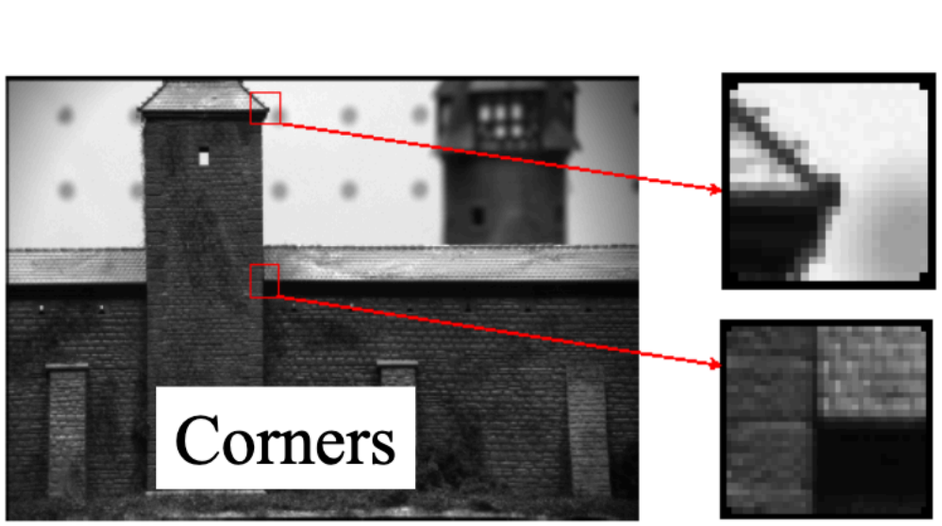
CV main Operations

Filtering and Smoothing

Linear operators
Convolution
Smoothing

CV main Operations

Feature Extraction



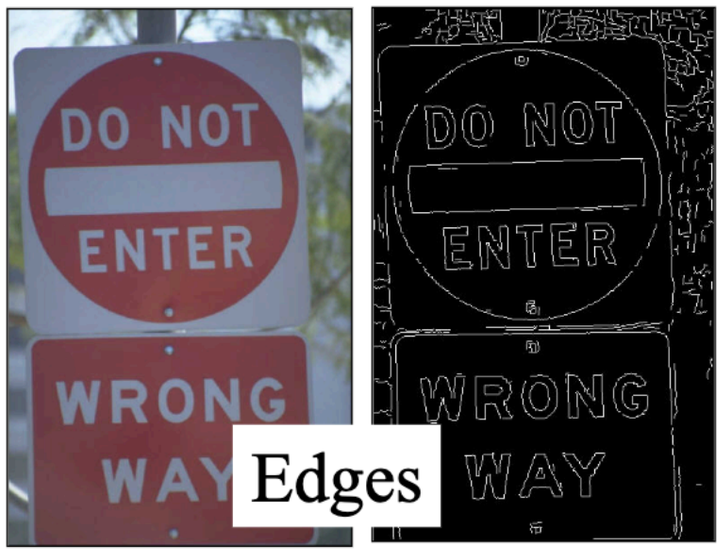
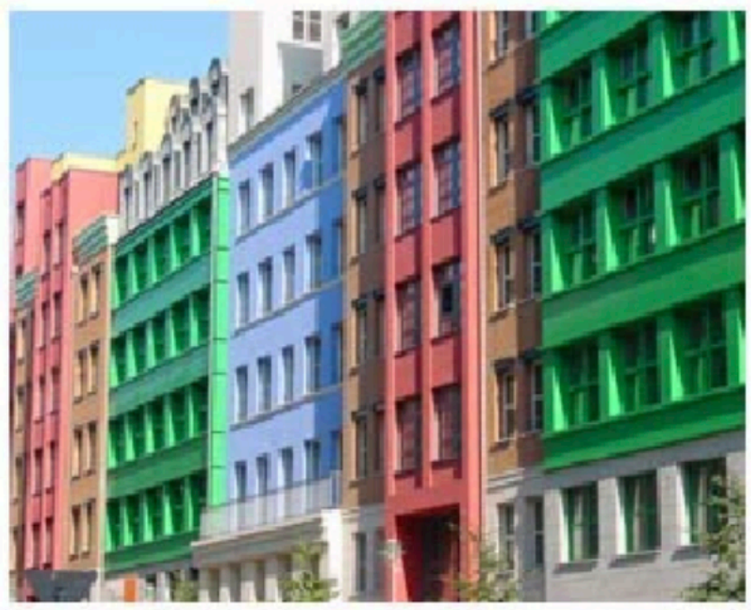


Image derivatives
Gradient operators
DoG/LoG operators
Harris corner detector

Why?
Seek more unique descriptors (than pixels) for matching

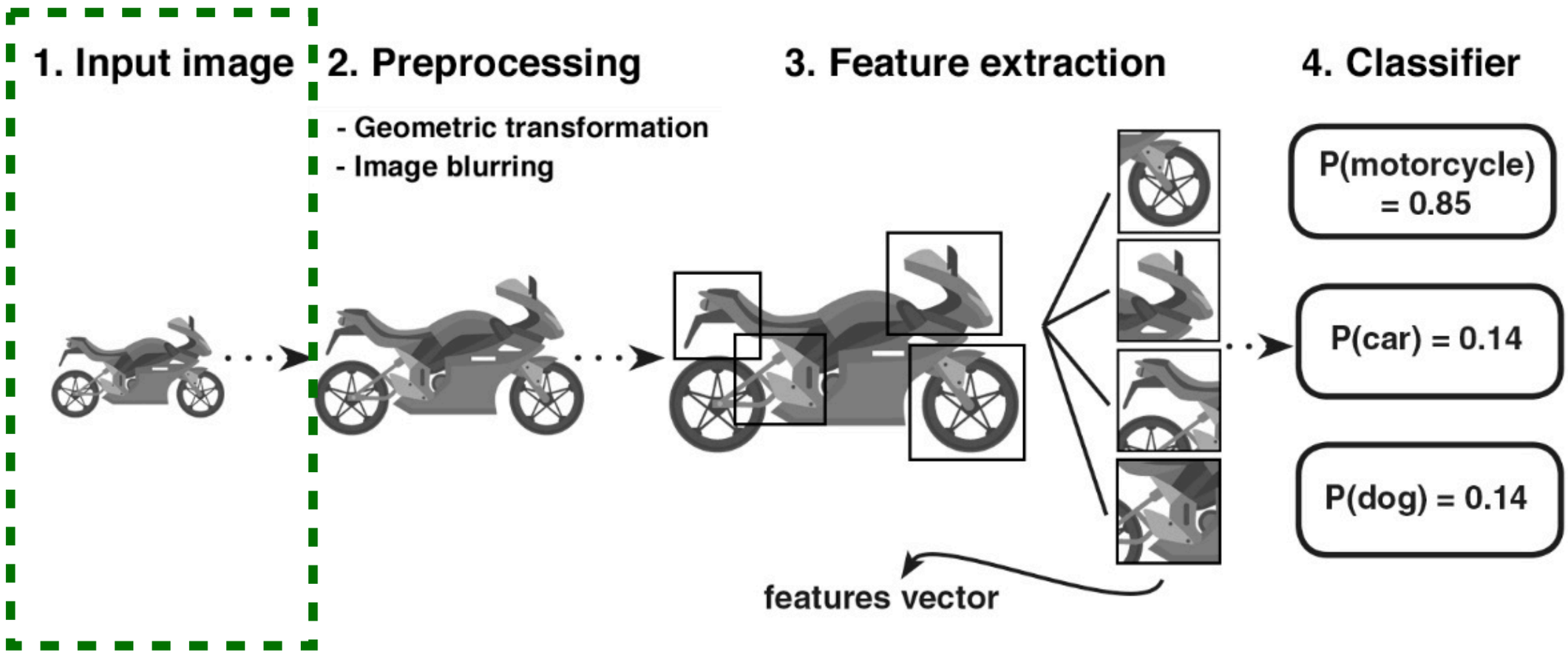
CV main Operations

Color and Light



Radiance / Reflection
Illumination / Shading
Chromaticity
Color Constancy

CV Pipeline


1. Input image
2. Preprocessing
- Geometric transformation
- Image blurring
3. Feature extraction
4. Classifier
P(motorcycle) = 0.85
P(car) = 0.14
P(dog) = 0.14
features vector

Images as functions

- **An Image** as a function *f* from R^2 to R^M:
  - $f(x, y)$ gives the **intensity** at position (x, y)
  - Defined over a rectangle, with a finite range:

$$f: [a,b] \times [c,d] \rightarrow [0,255]$$

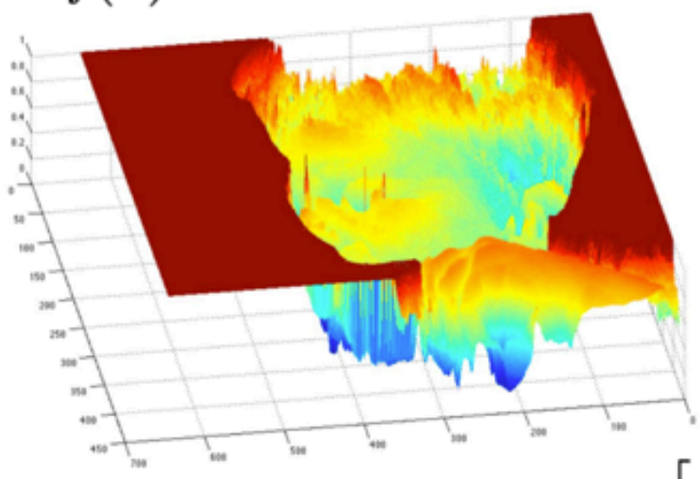
Domain support

range

grayscale image

What is the range of the image function f?

$f(\boldsymbol{x})$



domain $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

A (grayscale) image is a 2D function.

Images as functions

- **An Image** as a function *f* from R^2 to R^M:
  - $f(x, y)$ gives the **intensity** at position (x, y)
  - Defined over a rectangle, with a finite range:

$$f: \underbrace{[a,b] \times [c,d]}_{\text{Domain support}} \rightarrow \underbrace{[0,255]}_{\text{range}}$$

- A color image: $f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$

Input Image

```
#import the libraries
import numpy as np
import matplotlib.pyplot as plt
import cv2
%matplotlib inline

#reading the image

image = cv2.imread('index.png')
image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
#plotting the image
plt.imshow(image)

#saving image
cv2.imwrite('test_write.jpg',image)
```

- By default, the *imread* function reads images in the BGR (Blue-Green-Red) format. We can read images in different formats using extra flags in the *imread* function:

  cv2.IMREAD_COLOR: Default flag for loading a color image
  cv2.IMREAD_GRAYSCALE: Loads images in grayscale format
  cv2.IMREAD_UNCHANGED: Loads images in their given format, including the alpha channel. Alpha channel stores the transparency information – the higher the value of alpha channel, the more opaque is the pixel

CV Pipeline

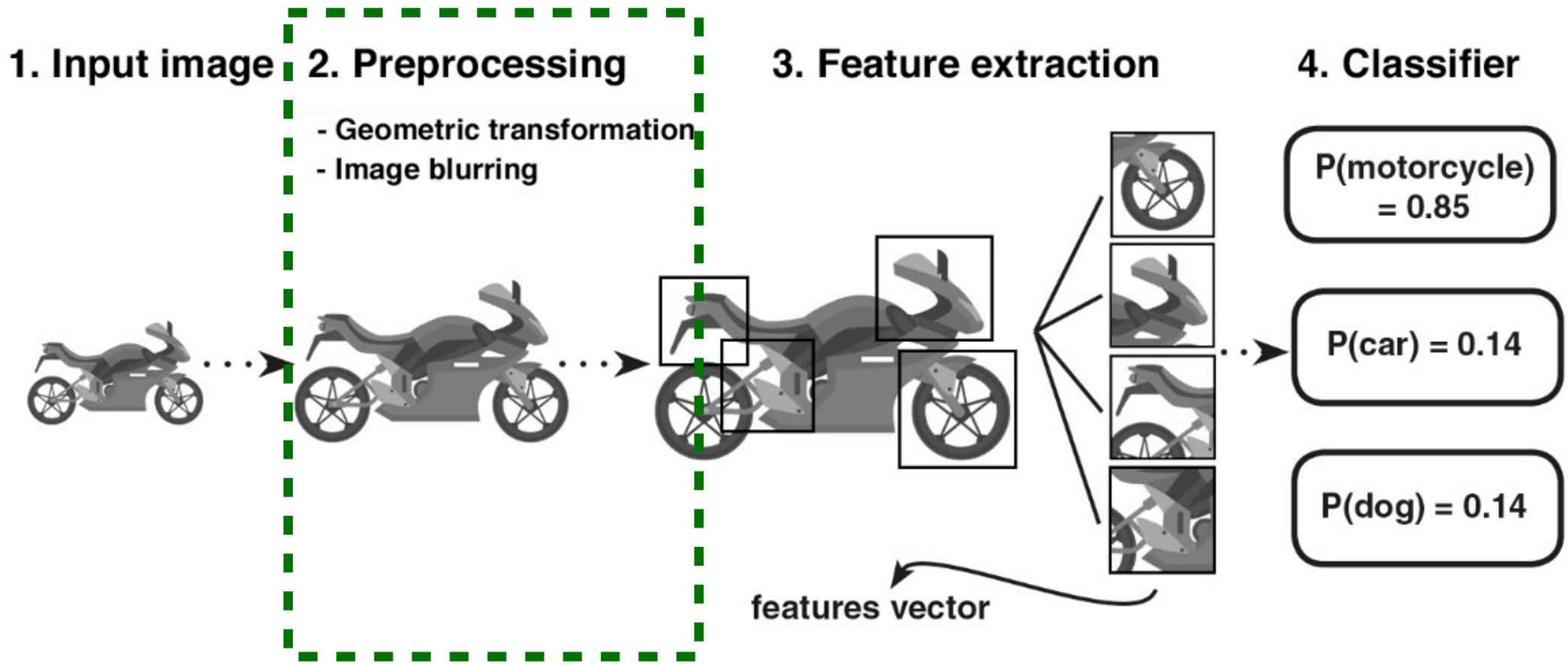

1. Input image
2. Preprocessing
- Geometric transformation
- Image blurring
3. Feature extraction
4. Classifier
P(motorcycle) = 0.85
P(car) = 0.14
P(dog) = 0.14
features vector

Image Augmentation

- Data augmentation uses the available data samples to produce the new ones, by applying image operations like **rotation, scaling, translation**, etc. This makes our model robust to changes in input and leads to better generalization.

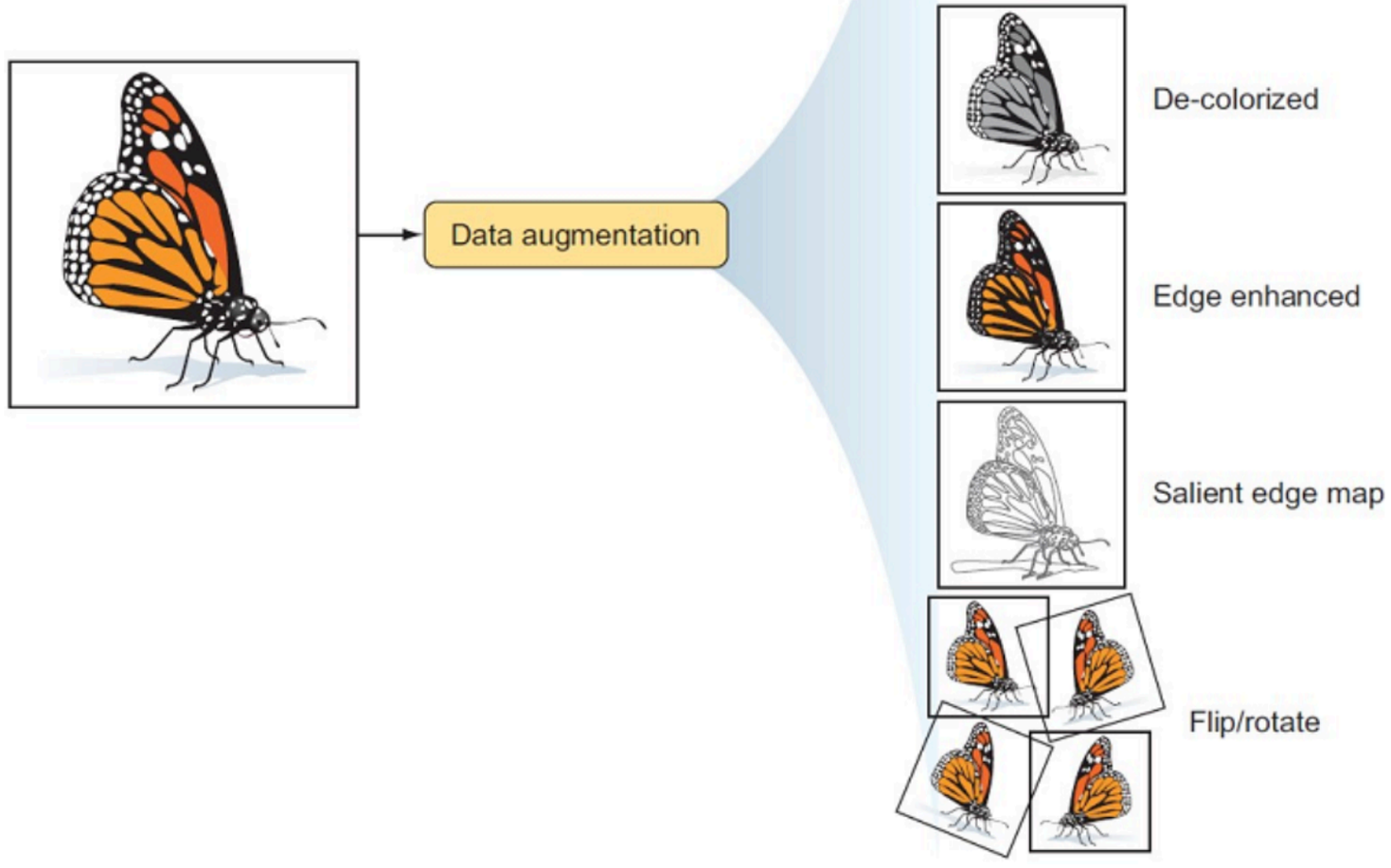
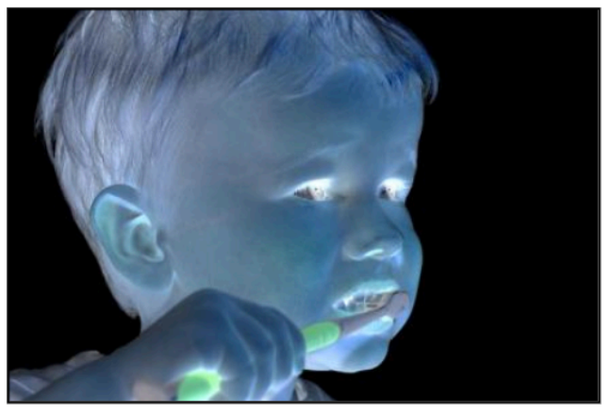


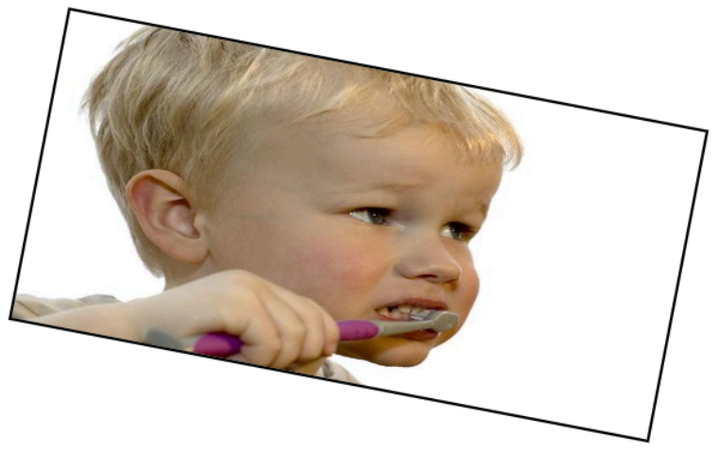
Image transformations

Filtering



changes pixel *values*

Warping



changes pixel *locations*

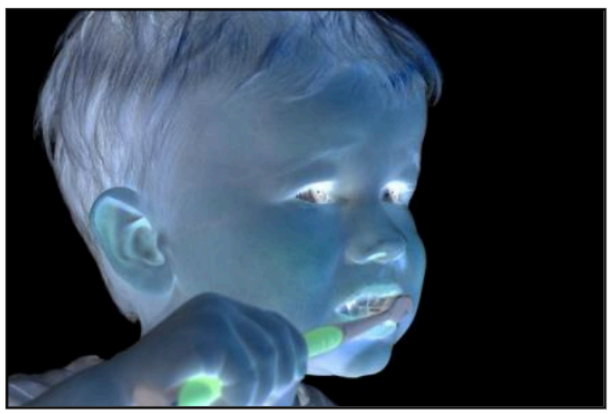
Image transformations

F

Filtering

$$G(\boldsymbol{x}) = h\{F(\boldsymbol{x})\}$$

G



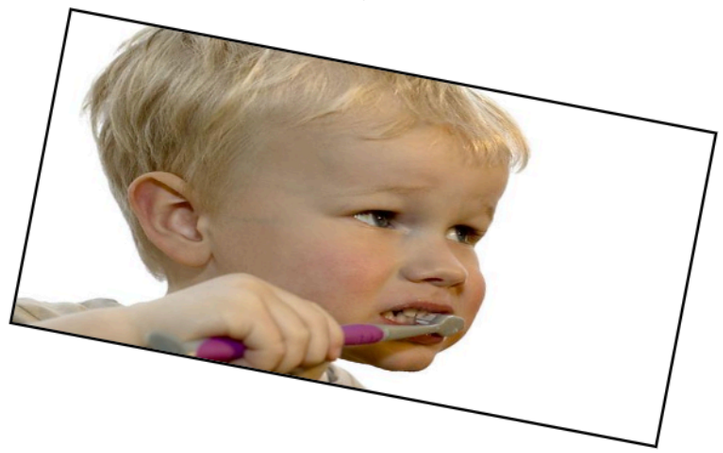
changes *range* of image function

F

Warping

$$G(\boldsymbol{x}) = F(h\{\boldsymbol{x}\})$$

G



changes *domain* of image function

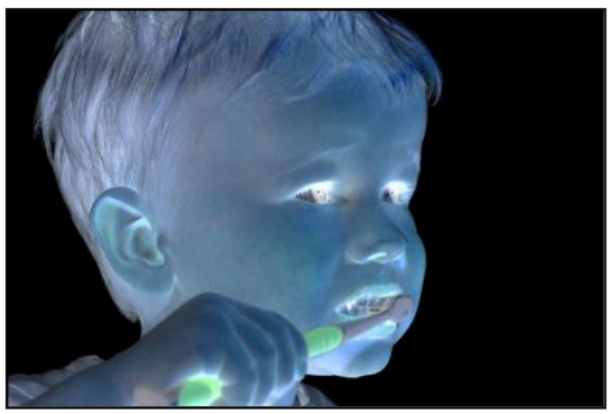
Image transformations

F

Filtering

$$G(\boldsymbol{x}) = h\{F(\boldsymbol{x})\}$$

G



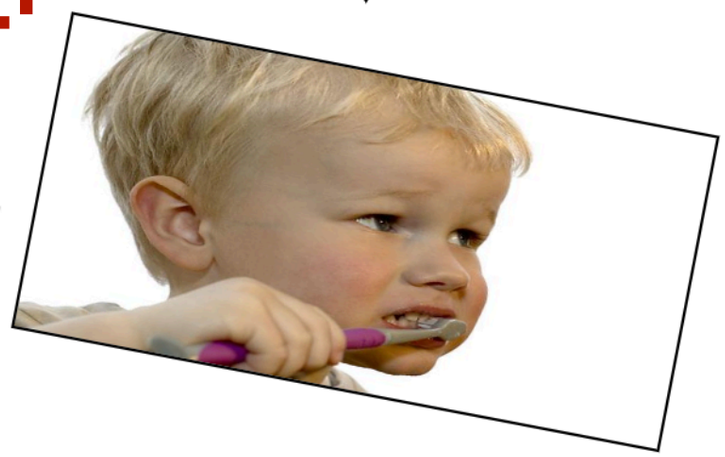
changes *range* of image function

F

Warping

$$G(\boldsymbol{x}) = F(h\{\boldsymbol{x}\})$$

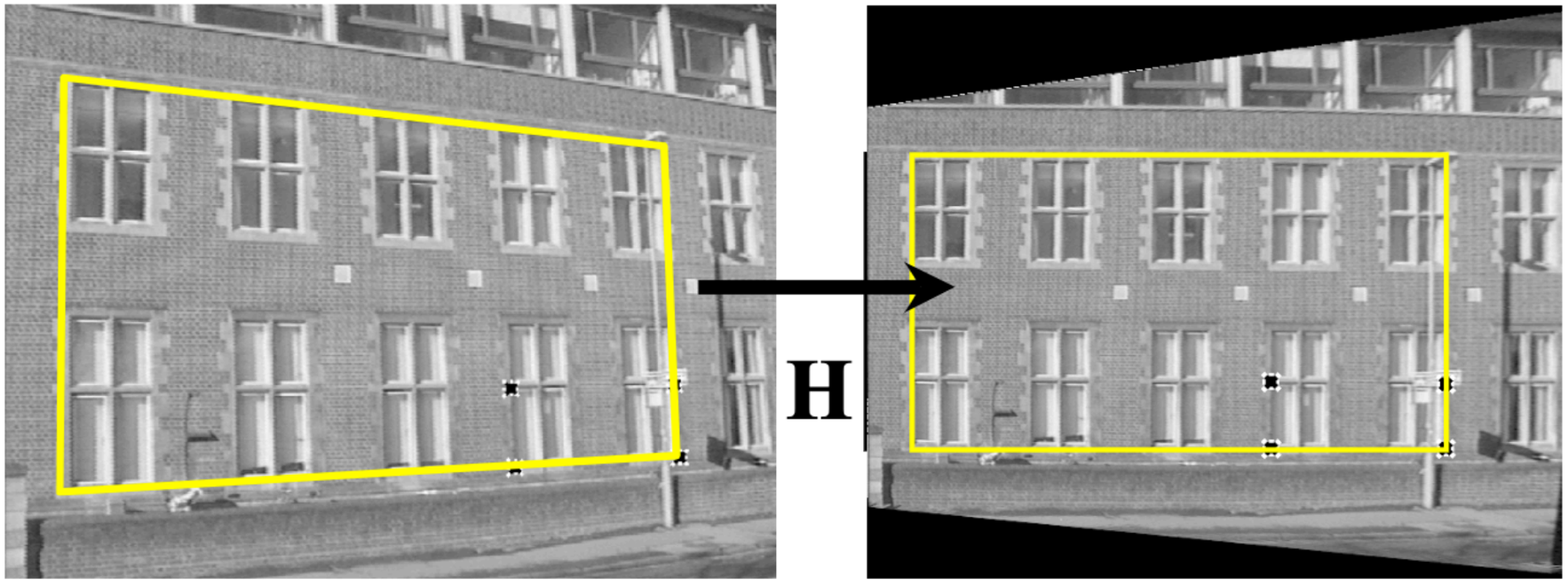
G



changes *domain* of image function

Transformation: Warping

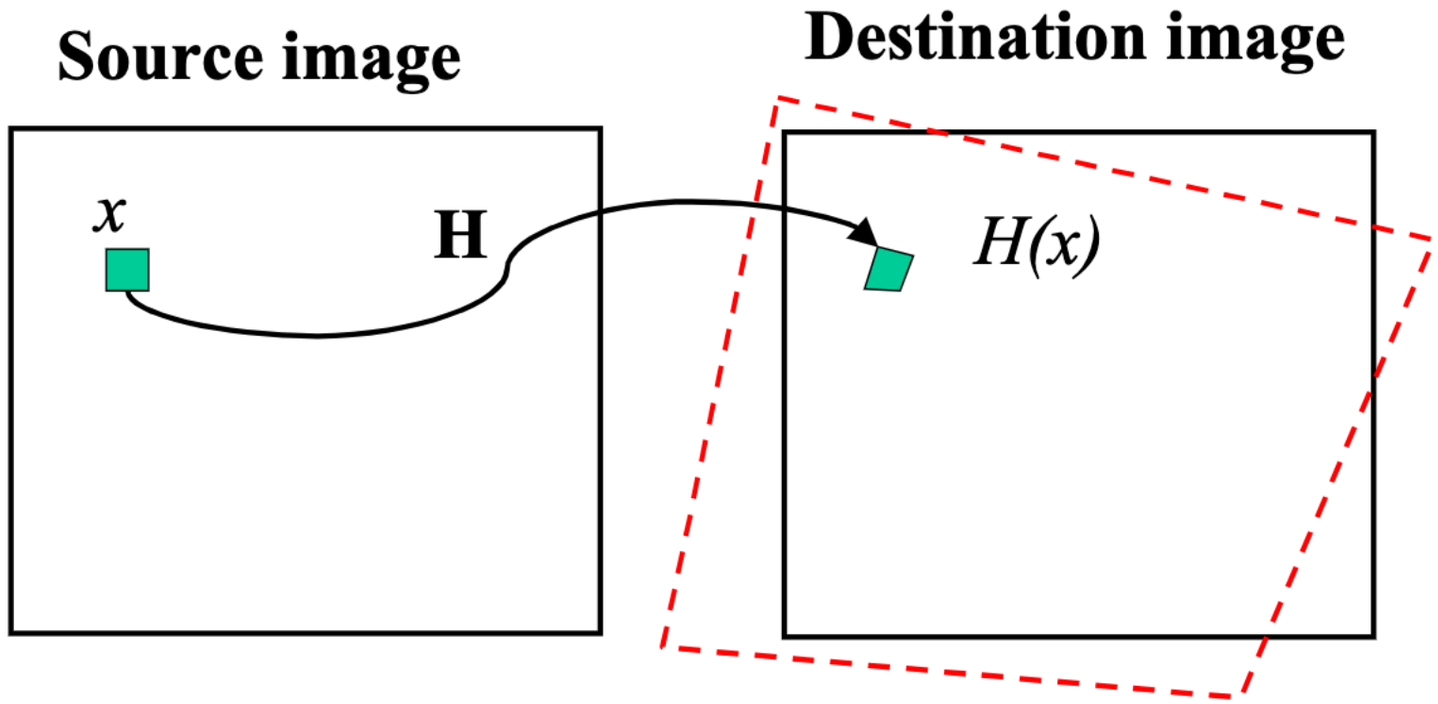
Transformation in this case is a projective transformation
(general 3x3 matrix, operating on homogeneous coords)



from Hartley & Zisserman

Source Image **Destination image**

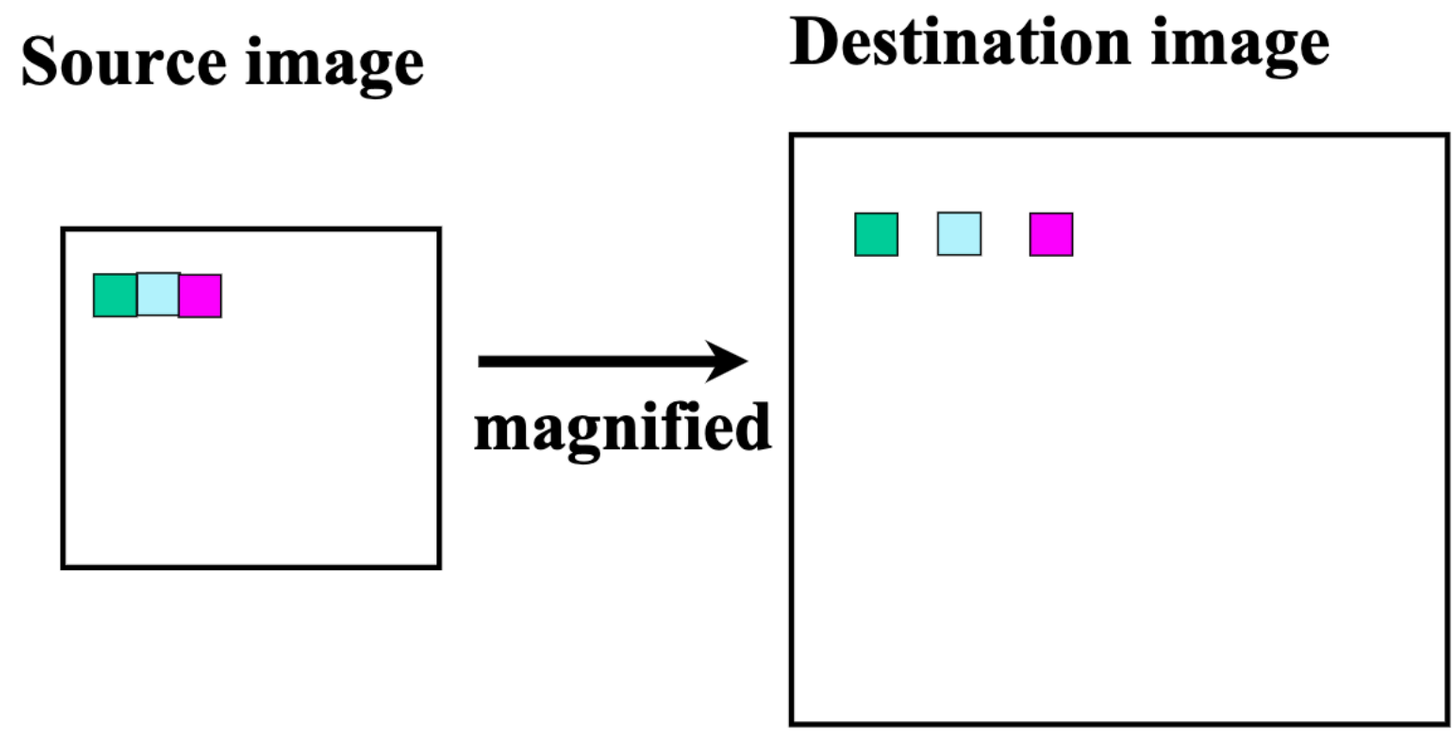
Forward Warping



- For each pixel *x* in the source image
- Determine where it goes as *H(x)*
- Color the destination pixel

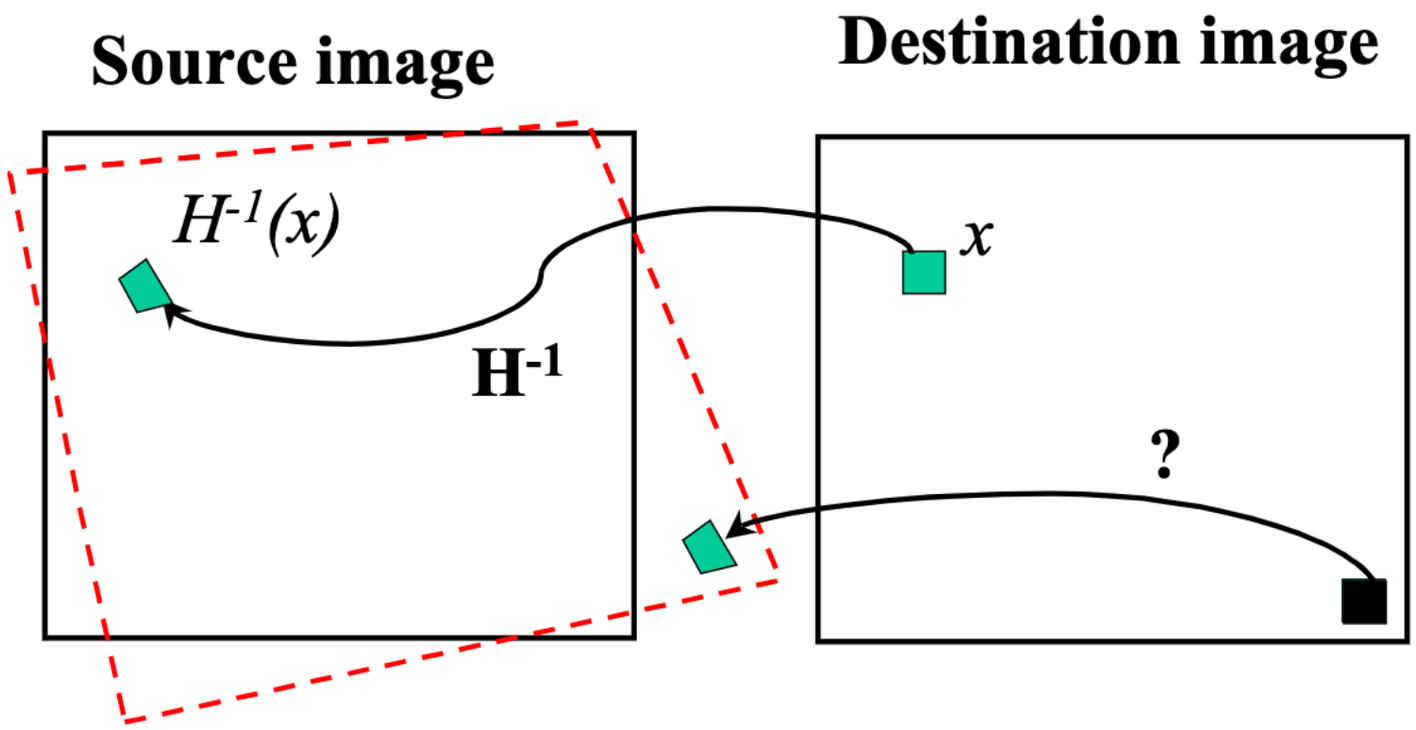
Problems?

Forward Warping (resizing)



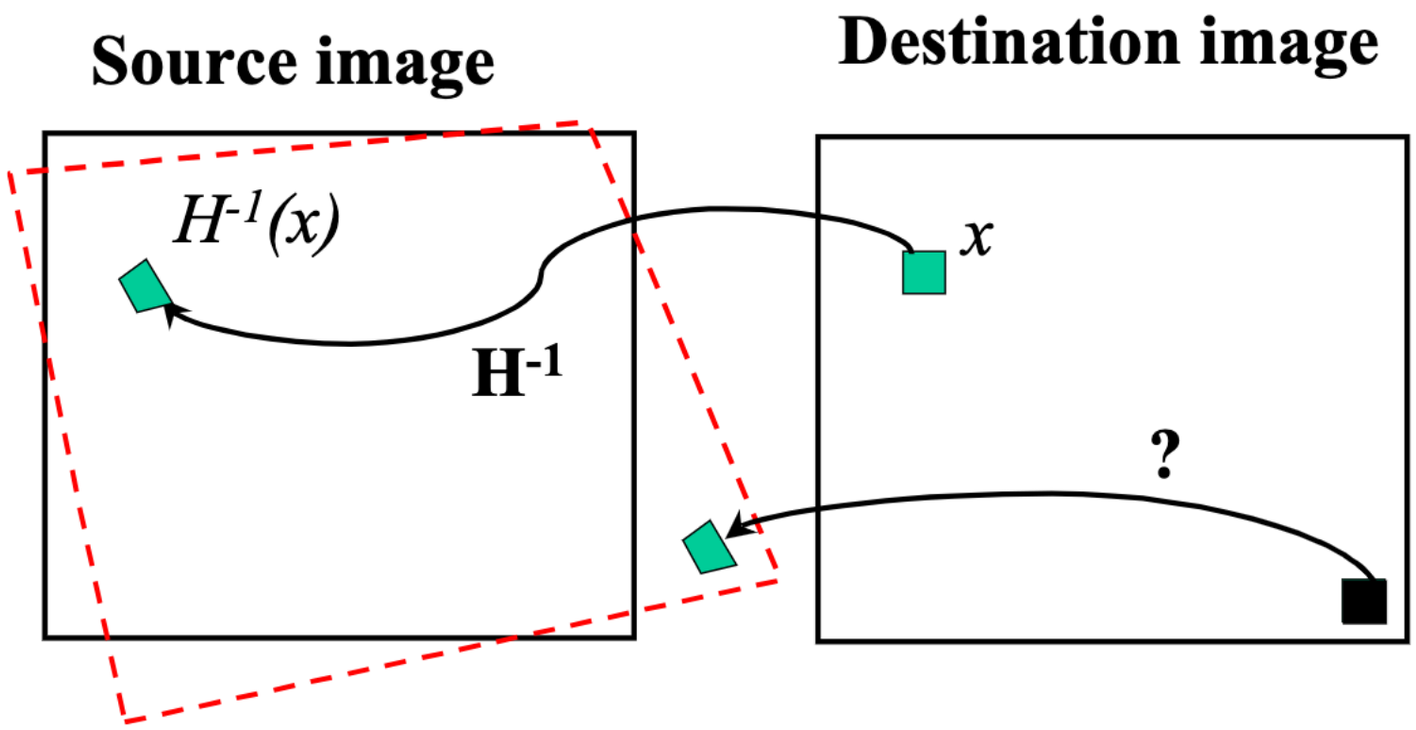
Can leave gaps!

Backward Warping



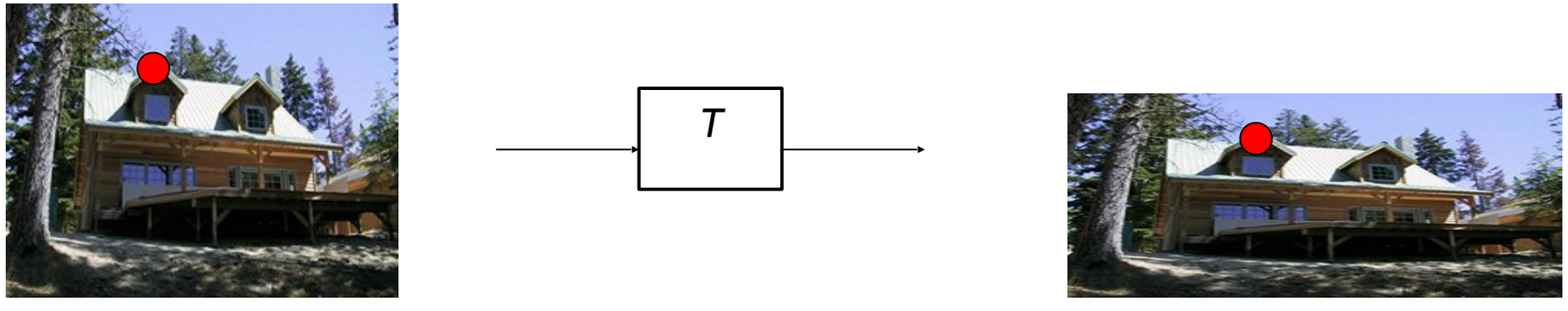
- For each pixel x in the destination image
- Determine where it comes from as $H^{-1}(x)$
- Get color from that location

Backward Warping



- For each pixel x in the destination image
- Determine where it comes from as $H^{-1}(x)$
- Get color from that location

Where the pixels go?



- Transformation T is a coordinate-changing machine:

$$p' = T(p)$$

- What does it mean that *T* is global?
  - Is the same for any point p
  - can be described by just a few numbers (parameters)
- Let’s consider *linear* xforms (can be represented by a 2D matrix):

$$\mathbf{p}' = \mathbf{T}\mathbf{p} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

Linear Transformation

- Uniform scaling by *s*:

(0,0)

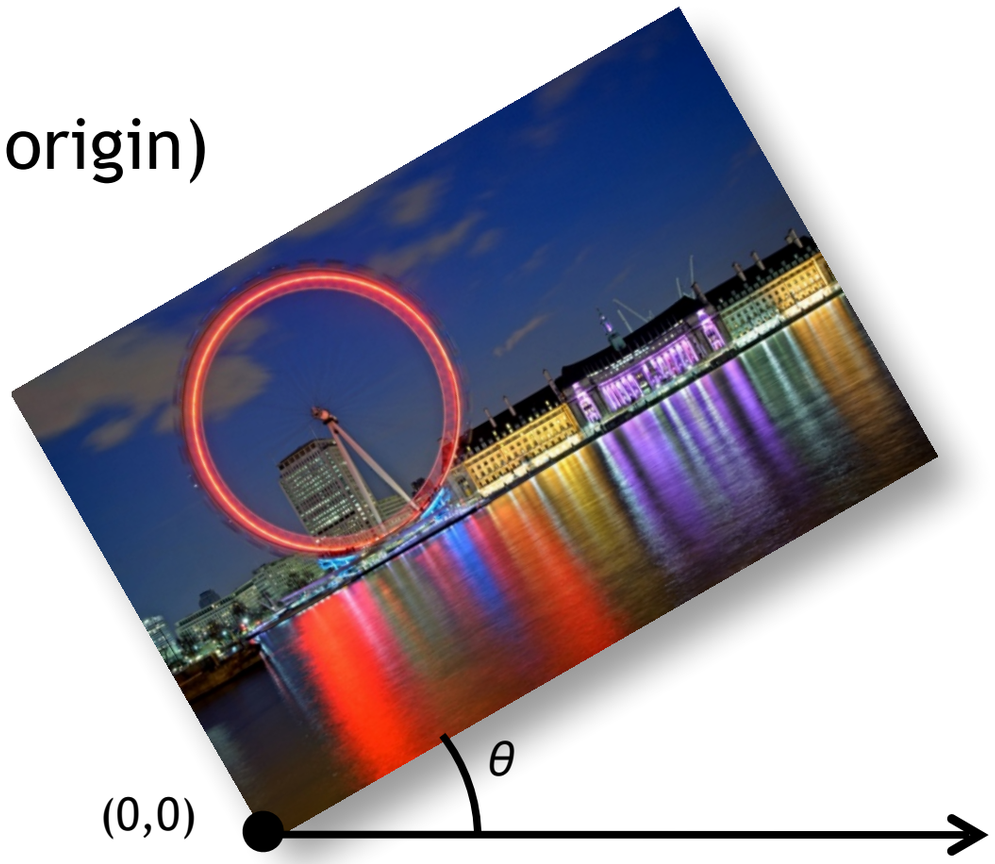
(0,0)

$$\mathbf{S} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

What is the inverse?

Linear Transformation

- Rotation by angle θ (about the origin)



$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

What is the inverse?

For rotations:

$$\mathbf{R}^{-1} = \mathbf{R}^{T}$$

Transformation with 2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D mirror about Y axis?

$$\begin{aligned} x' &= -x \\ y' &= y \end{aligned} \qquad \mathbf{T} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

2D mirror across line y = x?

$$\begin{aligned} x' &= y \\ y' &= x \end{aligned} \qquad \mathbf{T} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Affine Transformation

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations

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

- 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

Affine Transformation

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

any transformation with last row [0 0 1] we call an *affine* transformation

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

Basic transformation

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

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

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

2D *in-plane* rotation

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

Shear

Projective transformation

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

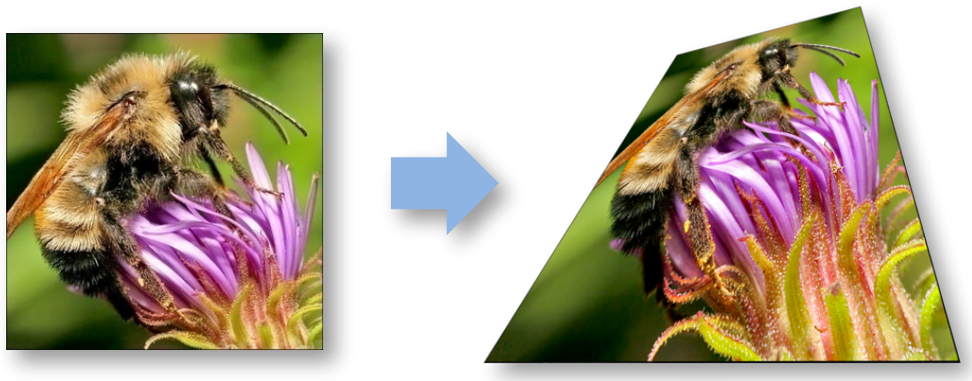
affine transformation

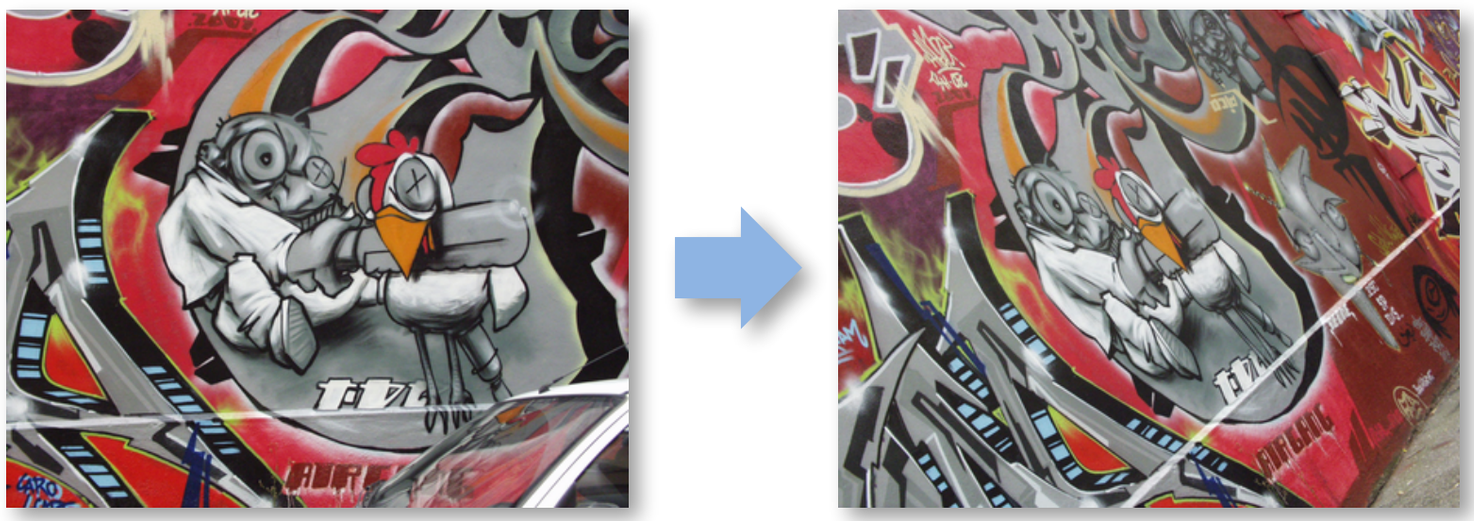
what happens when we mess with this row?

Projective transformation

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a *homography*
(or *planar perspective map*)





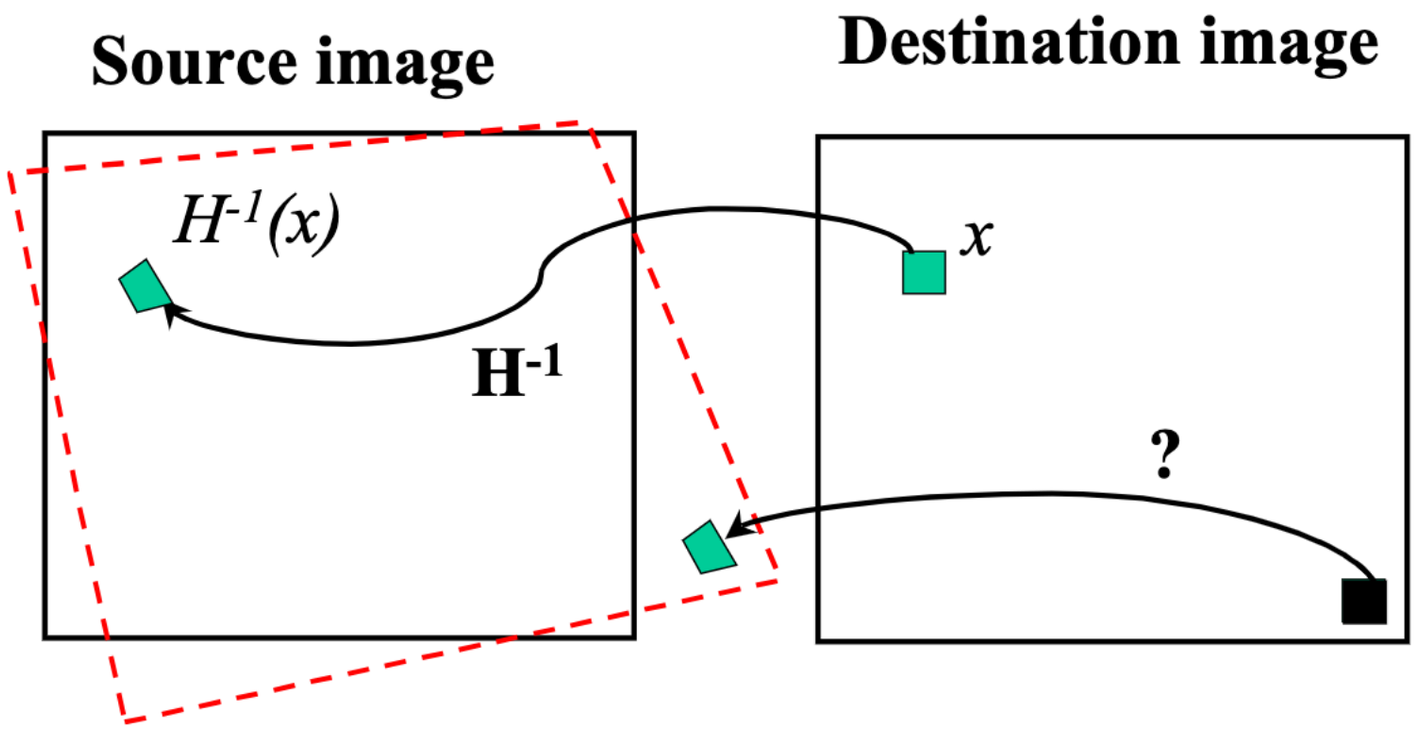
Projective transformation

- Projective transformations …
  - –Affine transformations, and
  - –Projective warps

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

- 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

Backward Warping

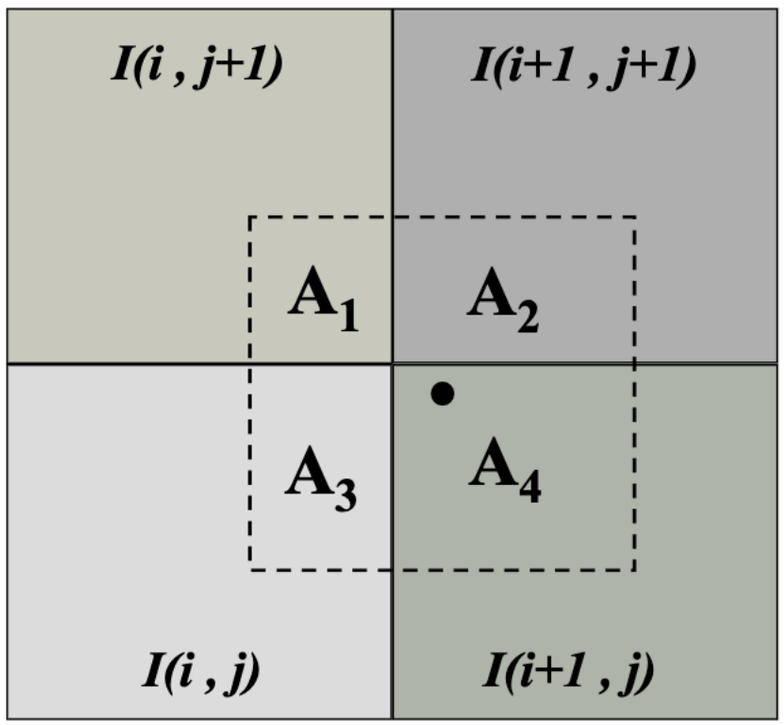


- For each pixel x in the destination image
- Determine where it comes from as $H^{-1}(x)$
- Get color from that location

Bilinear interpolation

What do we mean by "get color from that location"?

Consider grey values. What is intensity at (x,y)?



Bilinear Interpolation:

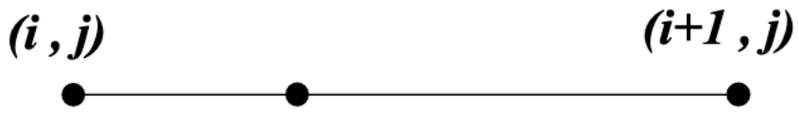
Weighted average

$$I(x,y) = A3*I(i,j) + A4*I(i+1,j) + A2*I(i+1,j+1) + A1*I(i,j+1)$$

Bilinear interpolation; the output pixel value is a weighted average of pixels in the nearest 2-by-2 neighborhood

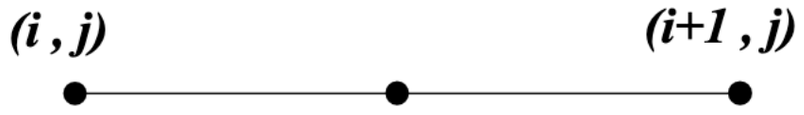
Linear Interpolation (recall)

First, consider linear interpolation



Inituition: Given two pixel values, what should the value be at some intermediate point between them?

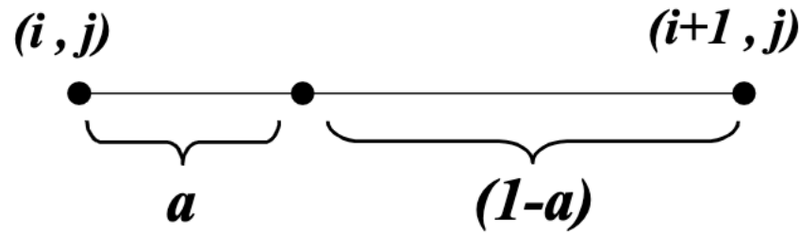
If close to (i,j), should be intensity similar to I(i,j)

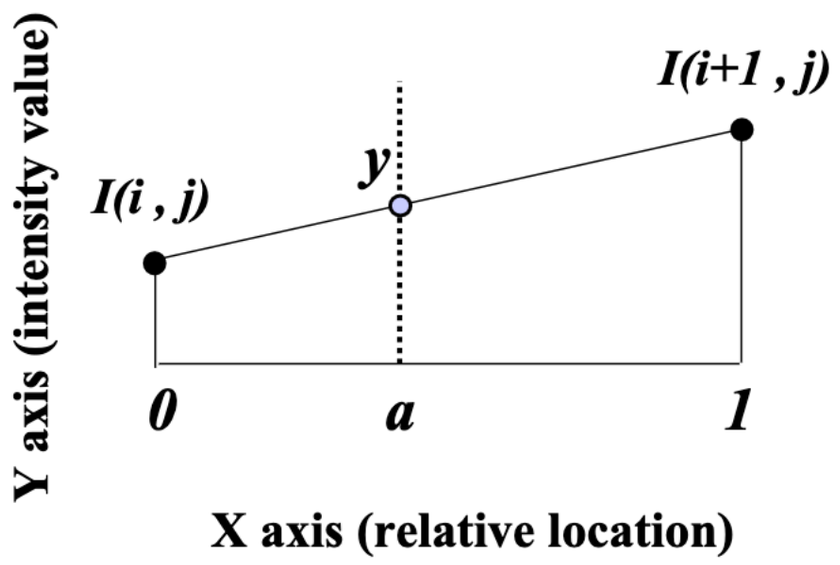


If equidistant from both, should be average of the two intensities

If close to (i+1,j), should be intensity similar to I(i+1,j)

Linear Interpolation (recall)





Recall:

$$y - y_1 = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1)$$

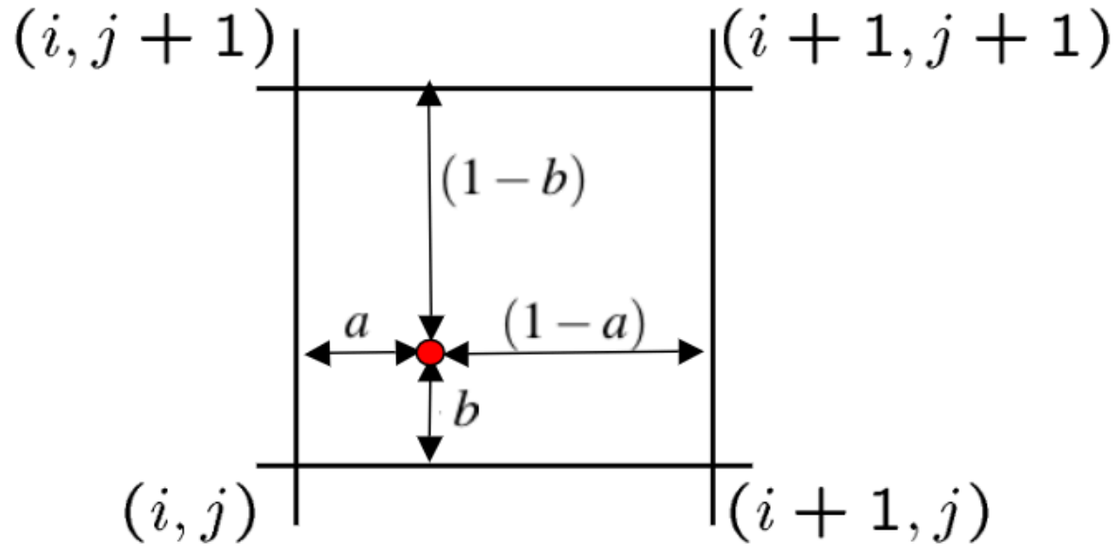
Instantiate

$$y - I(i,j) = \frac{(I(i+1,j) - I(i,j))}{(1-0)}(a-0)$$

Solve

$$y = (1-a)\, I(i,j) + a\ I(i+1,j)$$

Bilinear Interpolation

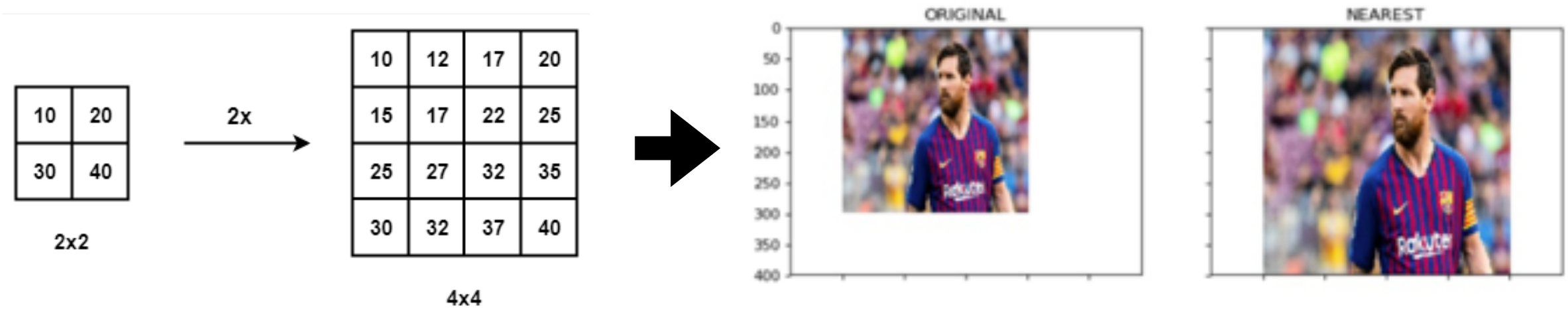


$$\begin{aligned}\mathbf{I} = \quad & (1-a)(1-b)\ I(i,j) \\ & + a\ (1-b)\ I(i+1,j) \\ & + (1-a)\ b\ I(i,j+1) \\ & + a\ b\ I(i+1,j+1)\end{aligned}$$

Image resizing

- Machine learning models work with a fixed sized input. The same idea applies to computer vision models as well. The images we use for training our model must be of the **same size.**

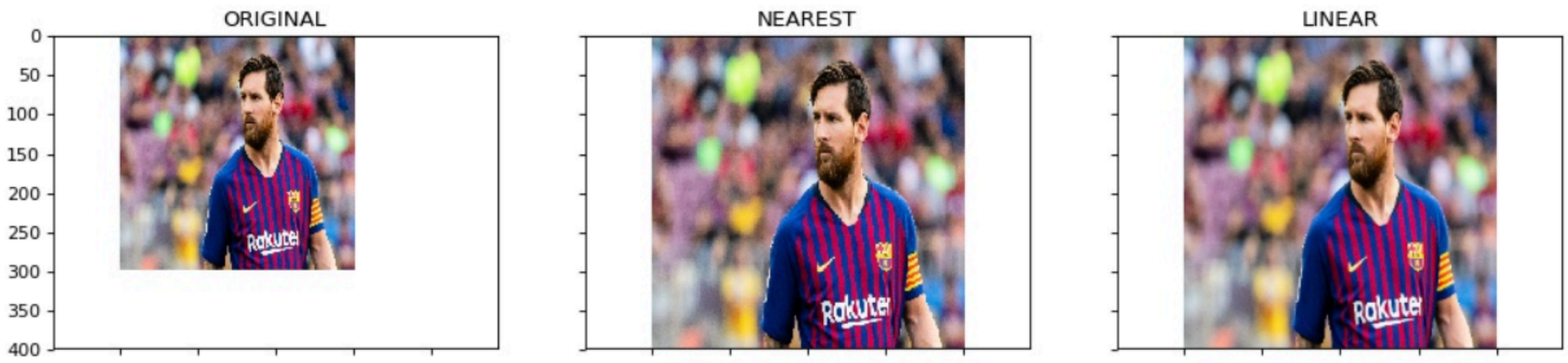
- Images can be easily scaled up and down



- Different interpolation and downsampling methods are supported by OpenCV. OpenCV's resize function uses bilinear interpolation by default.

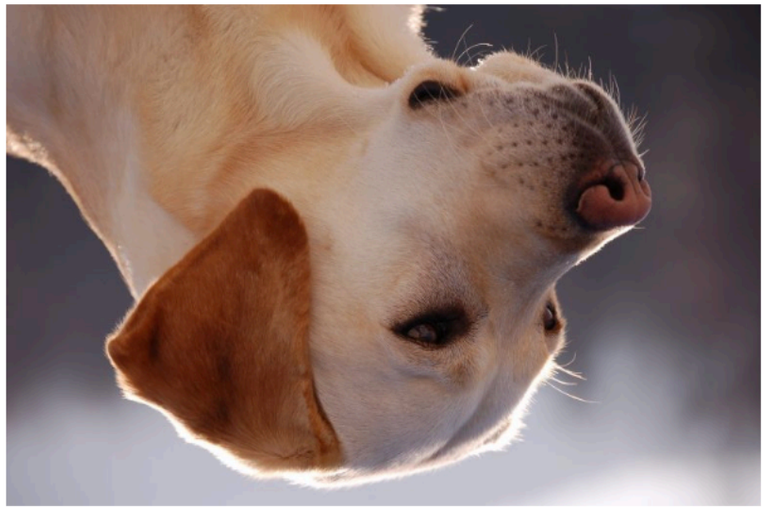
Resizing with OpenCV

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#reading the image
image = cv2.imread('index.jpg')
#converting image to size (100,100,3)
smaller_image = cv2.resize(image,(100,100),inerpolation='linear')
#plot the resized image
plt.imshow(smaller_image)
```



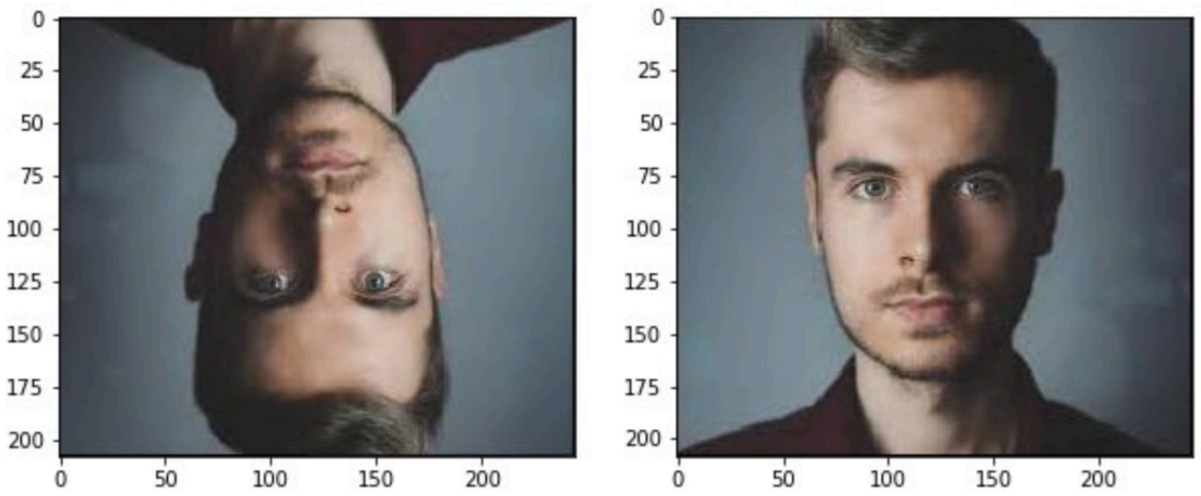
Rotate

- Suppose we are building an image classification model for identifying the animal present in an image. So, both the images shown below should be classified as ‘dog’:



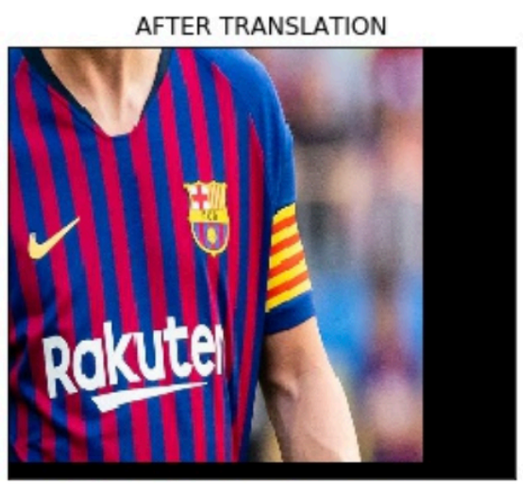
Rotation with OpenCV

```
#importing the required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
image = cv2.imread('index.png')
rows,cols = image.shape[:2]
#(col/2,rows/2) is the center of rotation for the image
# M is the cordinates of the center
M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
dst = cv2.warpAffine(image,M,(cols,rows))
plt.imshow(dst)
```



Shifting

```python
#importing the required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
#reading the image
image = cv2.imread('index.png')
#shifting the image 100 pixels in both dimensions
M = np.float32([[1,0,-100],[0,1,-100]])
dst = cv2.warpAffine(image,M,(cols,rows))
plt.imshow(dst)
```



CV Pipeline

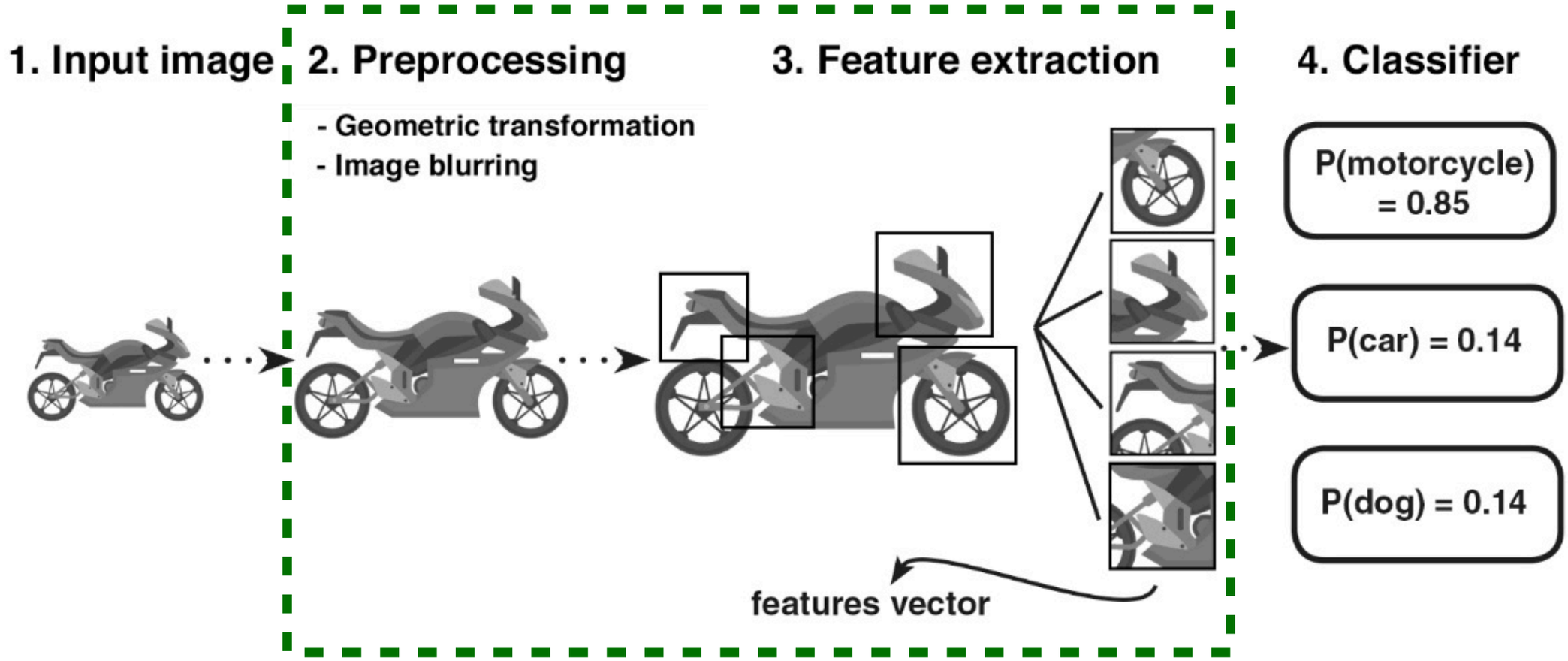

1. Input image
2. Preprocessing
- Geometric transformation
- Image blurring
3. Feature extraction
features vector
4. Classifier
P(motorcycle) = 0.85
P(car) = 0.14
P(dog) = 0.14

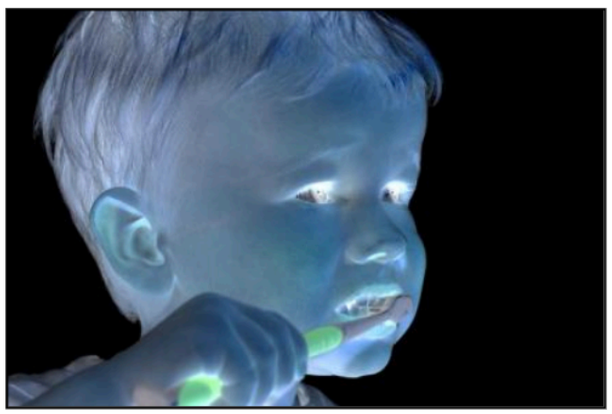
Image transformations

F

Filtering

$$G(\boldsymbol{x}) = h\{F(\boldsymbol{x})\}$$

G



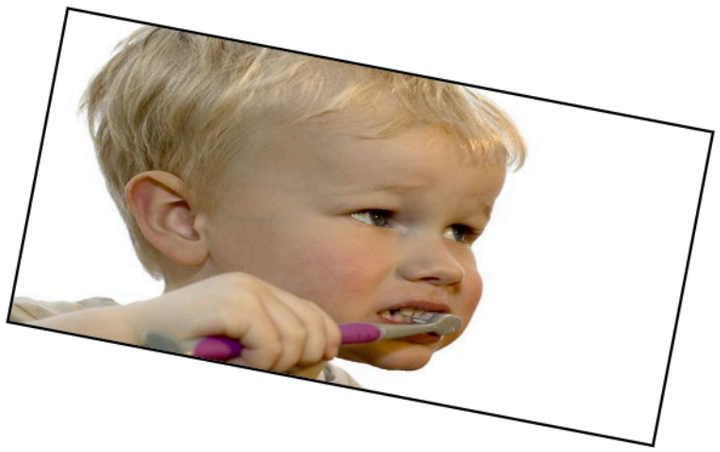
changes *range* of image function

F

Warping

$$G(\boldsymbol{x}) = F(h\{\boldsymbol{x}\})$$

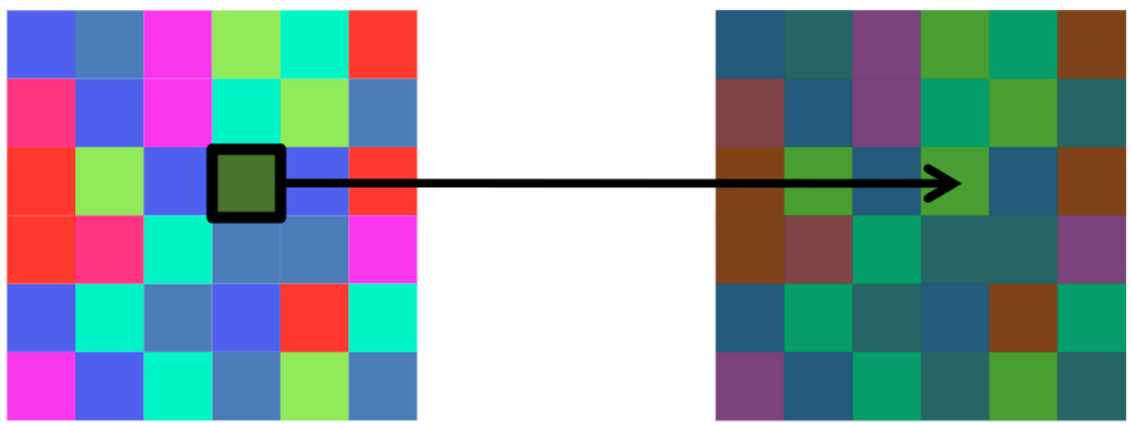
G



changes *domain* of image function

Image filtering

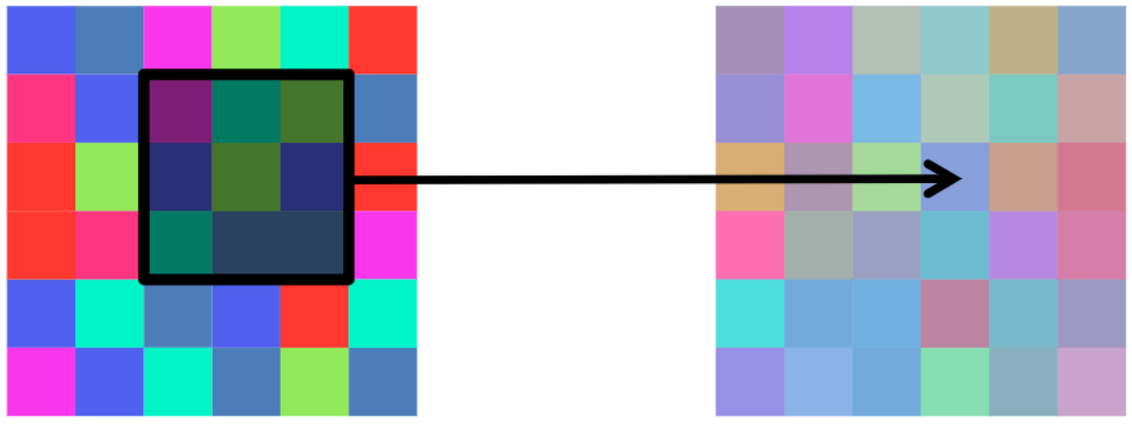
Point Operation



1D

point processing

Neighborhood Operation

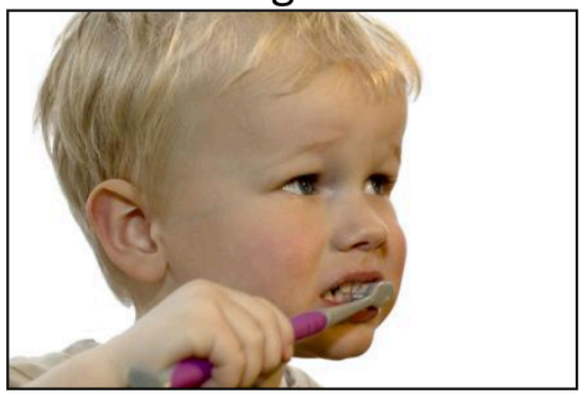


2D

“filtering”

Point processing

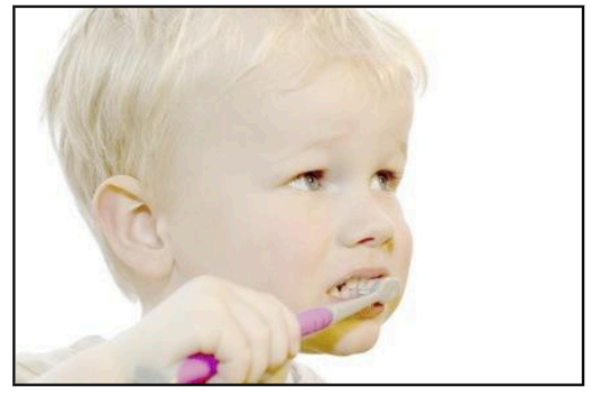
original



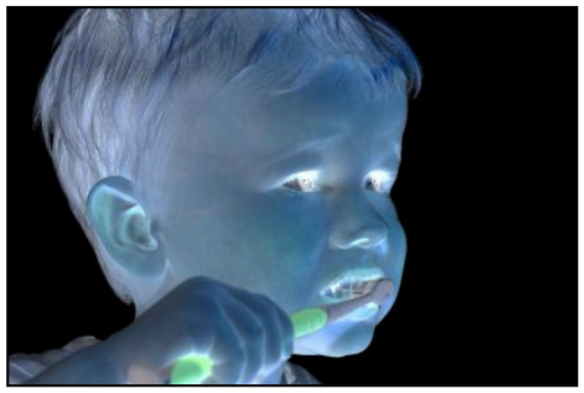
darken

lower contrast

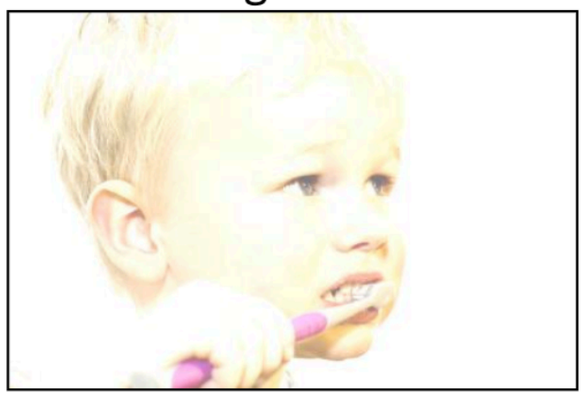
non-linear lower contrast



invert

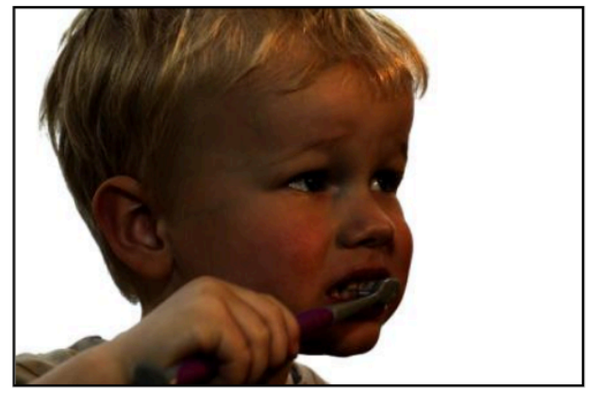


lighten



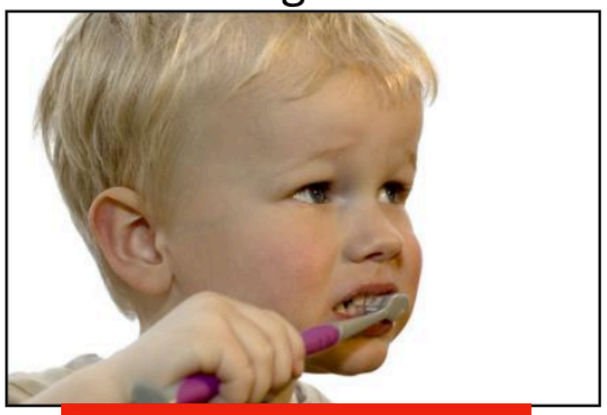
raise contrast

non-linear raise contrast



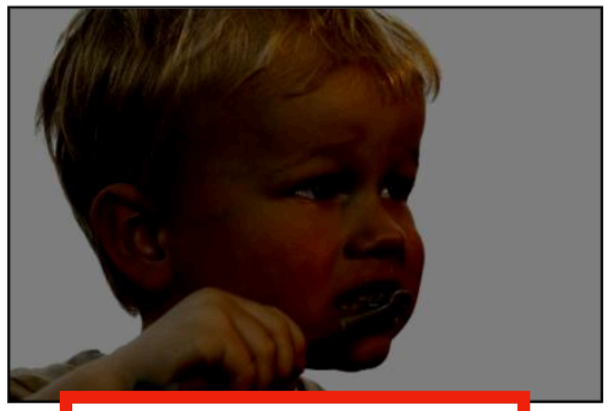
Point processing
(How to implement them?)

original



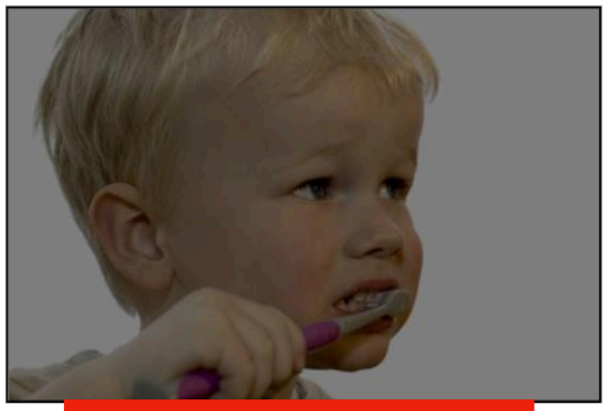
x

darken



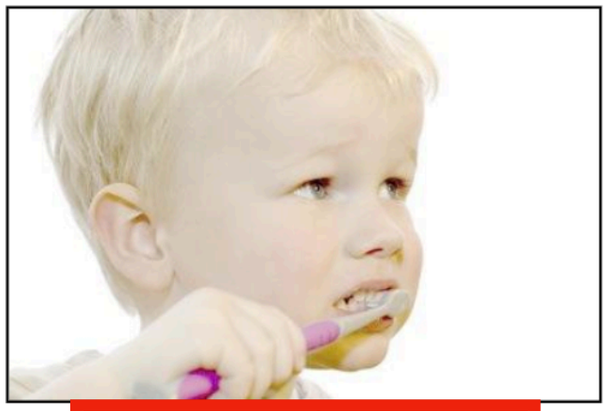
$x - 128$

lower contrast



$\frac{x}{2}$

non-linear lower contrast

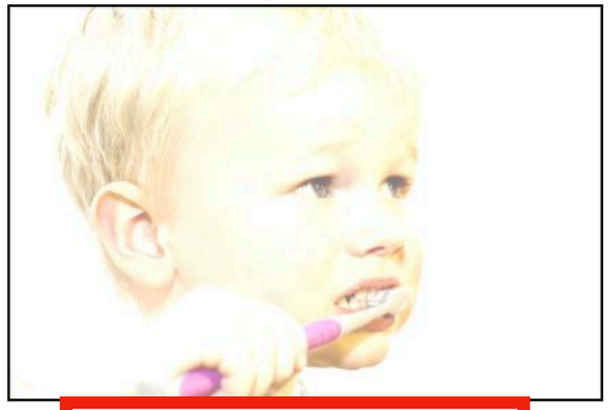


$\left(\frac{x}{255}\right)^{1/3} \times 255$

invert

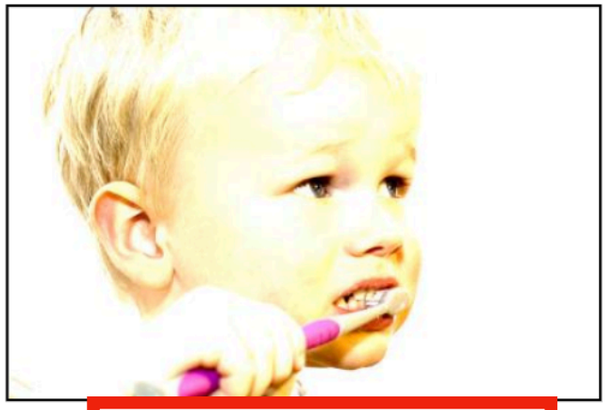
$255 - x$

lighten



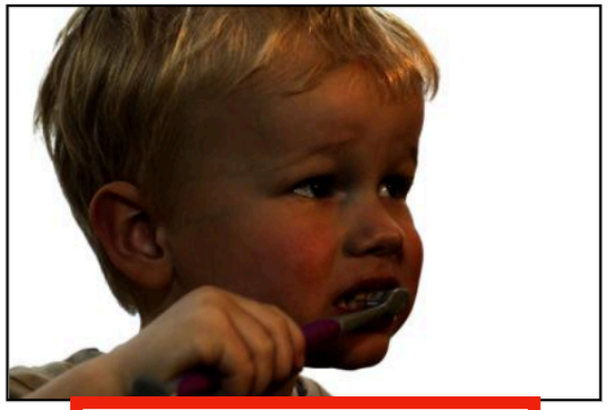
$x + 128$

raise contrast



$x \times 2$

non-linear raise contrast



$\left(\frac{x}{255}\right)^{2} \times 255$

Filtering

- **Filtering:**
  - Form a new image whose pixels are a combination original pixel values

Goals:

-Extract useful information from the images

- Features (edges, corners, blobs…)

- Modify or enhance image properties:

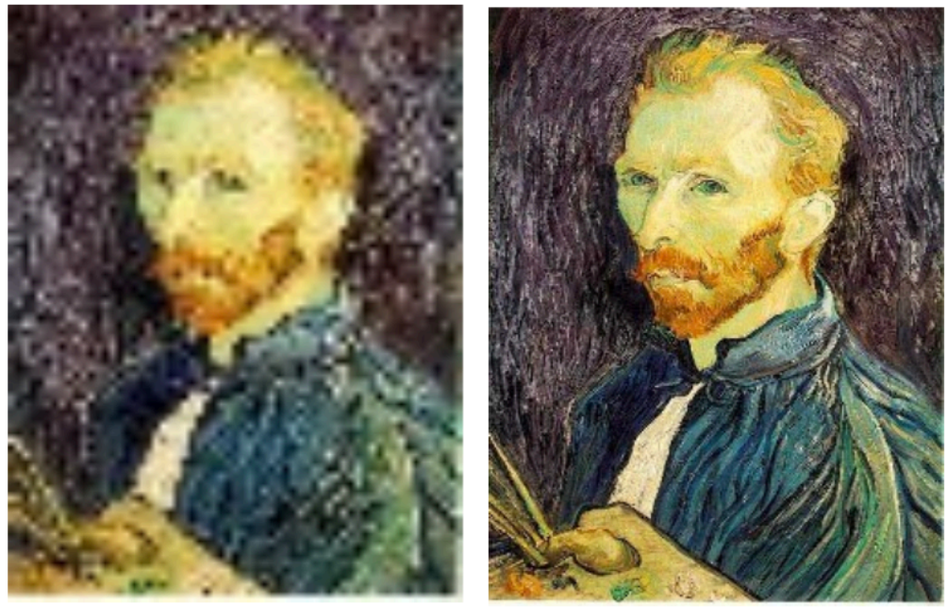
- super-resolution; in-painting; de-noising

Enhancing Examples

De-noising

Salt and pepper noise

Super-resolution



In-painting

Bertamio et al

Image filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel

Local image data

Modified image data

2D discrete-space systems (filters)

$$f[n,m] \rightarrow \boxed{\text{System } \mathcal{S}} \rightarrow g[n,m]$$

$$g = \mathcal{S}[f], \quad g[n,m] = \mathcal{S}\{f[n,m]\}$$

$$f[n,m] \xrightarrow{\mathcal{S}} g[n,m]$$

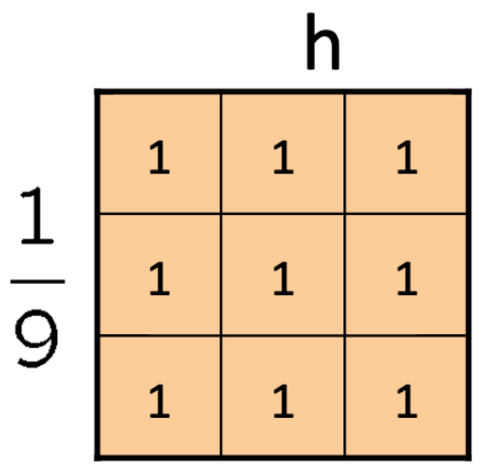
Filter example: Moving average

- Also known as **Box filter**

$$g[n,m] = \frac{1}{9} \sum_{k=n-1}^{n+1} \sum_{l=m-1}^{m+1} f[k,l]$$

$$= \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} f[n-k, m-l]$$

$$(f * h)[m,n] = \frac{1}{9} \sum_{k,l} f[k,l]\, h[m-k, n-l]$$



mask or weight kernel

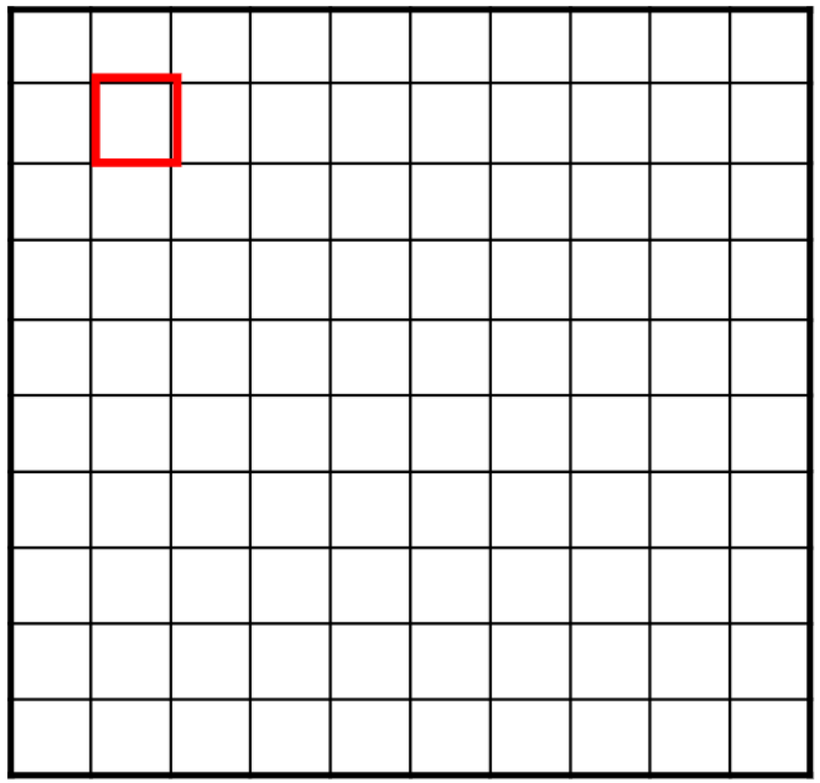
- 2D moving average over a 3×3 window of neighborhood

Filter example: Moving average

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$



$$(f * h)[m,n] = \sum_{k,l} f[k,l]\, h[m-k,n-l]$$

Filter example: Moving average

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10							

$$(f * h)[m,n] = \sum_{k,l} f[k,l]\, h[m-k,n-l]$$

Filter example: Moving average

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20						

$$(f * h)[m,n] = \sum_{k,l} f[k,l]\, h[m-k,n-l]$$

Filter example: Moving average

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

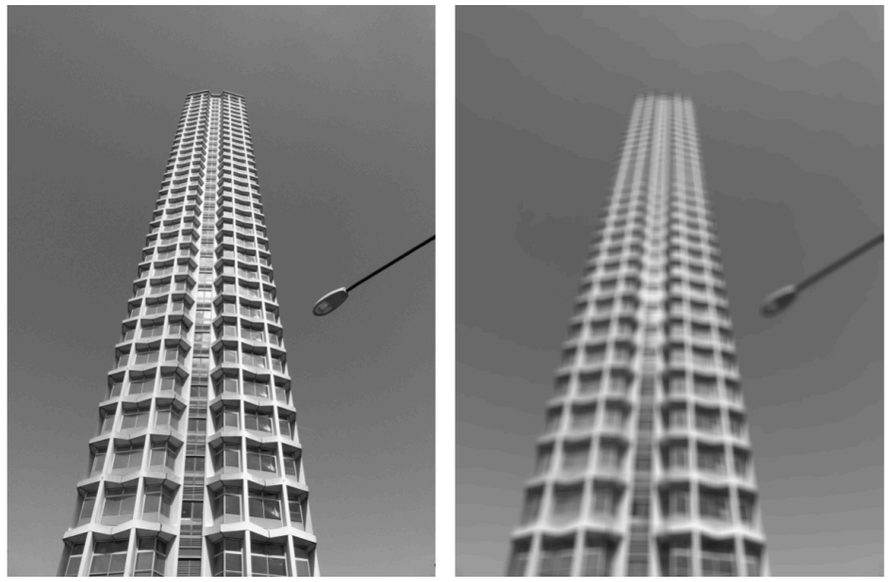
$G[x, y]$

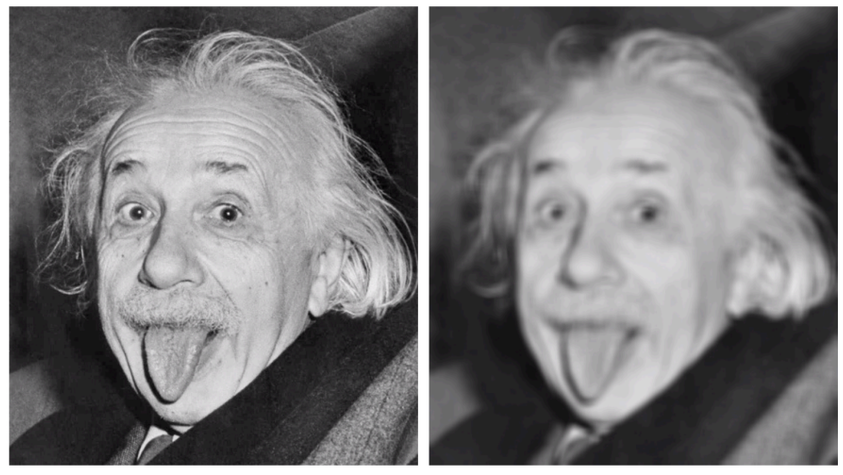
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$(f * h)[m,n] = \sum_{k,l} f[k,l]\, h[m-k,n-l]$$

Source: S. Seitz

Filter example: Moving average





Achieve smoothing effect (remove sharp features)

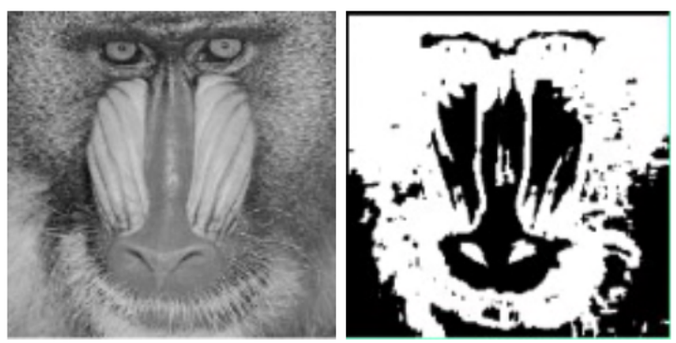
Filter example: Image Segmentation

- Image segmentation based on a simple threshold:

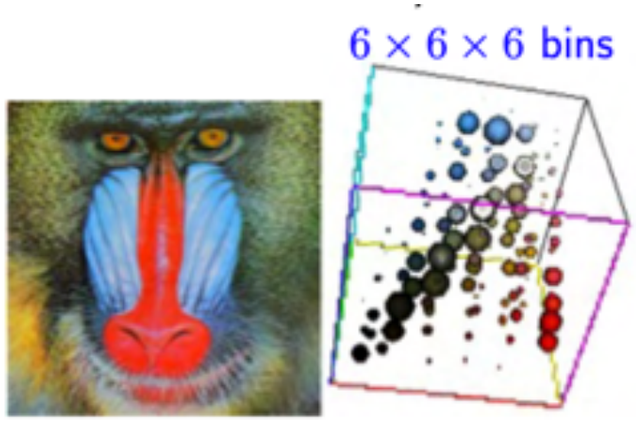
$$g[n,m] = \begin{cases} 255, & f[n,m] > 100 \\ 0, & \text{otherwise.} \end{cases}$$

Filter example: Image Segmentation

- Non-contextual: grouping pixels with similar global features.



- Contextual: grouping pixels with similar features and in close locations.



Filtering properties

Commutative: $f * g = g * f$

Associative: $(f * g) * g = f * (g * h)$

Distributive: $(f + g) * h = f * h + g * h$

Linear: $(a\ f + b\ g) * h = a\ f * h + b\ g * h$

Shift Invariant: $f(x+t) * h = (f * h)(x+t)$

Differentiation rule:

$$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$$

Shift invariant

- Filter replaces each pixel by a linear combination of its neighbors (and possibly itself). The combination is determined by the filter's kernel.

If $f[n,m] \xrightarrow{\mathcal{S}} g[n,m]$ then

$$\boxed{f[n-n_0, m-m_0] \xrightarrow{\mathcal{S}} g[n-n_0, m-m_0]}$$

for every input image f[n,m] and shifts n_0,m_0

Shift invariant

Is the moving average system shift invariant?

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Shift invariant

Is the moving average system shift invariant?

$$f[n,m] \quad \xrightarrow{\mathcal{S}} \quad g[n,m] = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[n-k,m-l]$$

$$f[n-n_0,m-m_0]$$

$$\xrightarrow{\mathcal{S}} \quad g[n,m] = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[n-k,m-l]$$

$$= \quad \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[(n-n_0)-k,(m-m_0)-l]$$

$$= \quad g[n-n_0,m-m_0]$$

Yes!

Linear filtering

Linear filtering means linear combination of neighboring pixel values.

- **S** is a linear system (function) iff it *S satisfies*

$$\mathcal{S}[\alpha f_1 + \beta f_2] = \alpha \mathcal{S}[f_1] + \beta \mathcal{S}[f_2]$$

superposition property

- Is the moving average system a linear system?
- Is thresholding a linear system?

Convolution

- Any linear, shift invariant operator can be represented as **convolution**!