SAIR

Spatial AI & Robotics Lab

CSE 473/573-A

W4: PYRAMIDS, HISTOGRAM, & FEATURE

Chen Wang

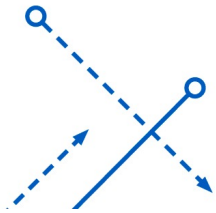
Spatial AI & Robotics Lab

Department of Computer Science and Engineering

University at Buffalo The State University of New York

Content

- Image Pyramids
  - Gaussian, Laplacian
  - Convolution and Transposed Convolution
- Image Histogram
  - Equalization, Matching
  - Image Enhancement
- Feature Extraction
  - Local features, Pyramids for invariant feature detection
  - Invariant descriptors and matching: Harris Detection, SIFT.
  - Integral Images, SURF



SAIR

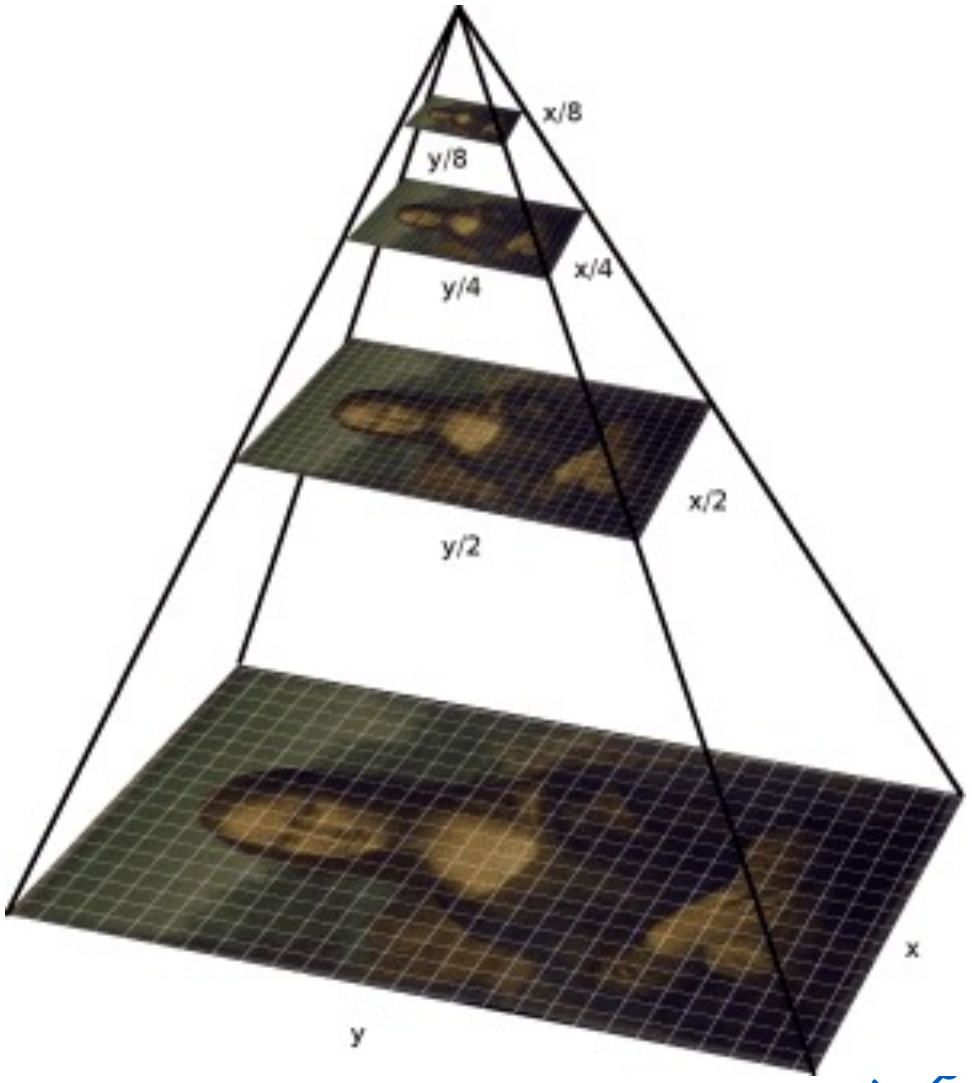
Spatial AI & Robotics Lab

IMAGE PROCESSING

Pyramids

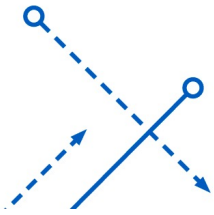
Image Pyramids

- Gaussian pyramid
- Laplacian pyramid
- Transposed Convolution

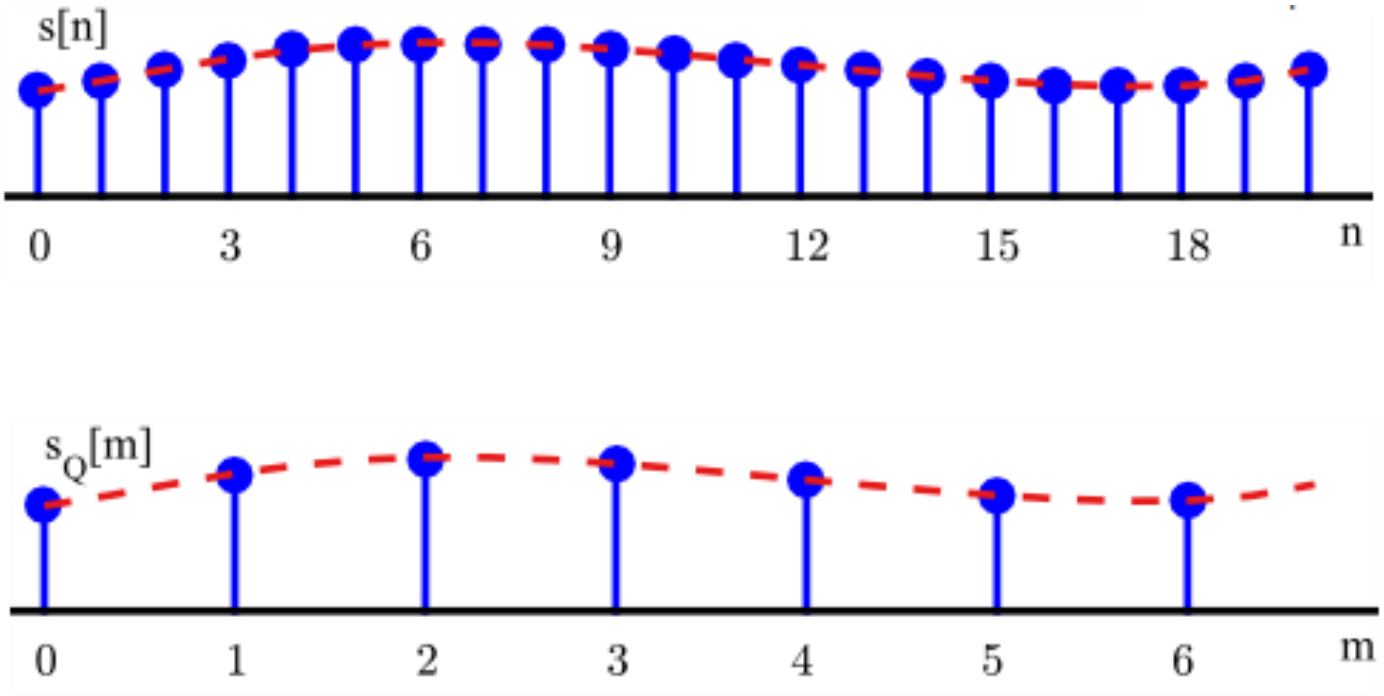


Pyramids applications

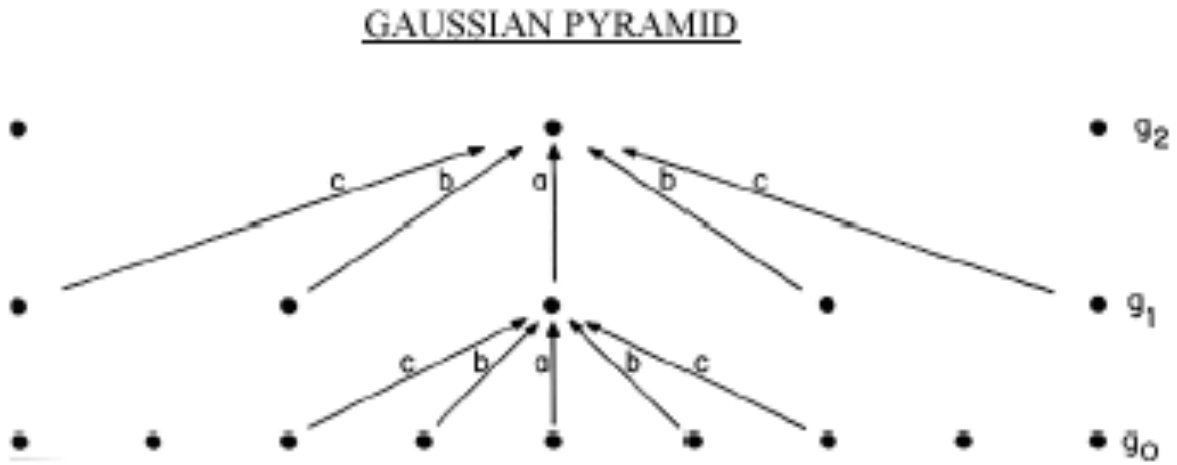
- Up- or Down- sampling images.
- Multi-resolution image analysis
  - Look for an object over various spatial scales
  - Coarse-to-fine image processing
    - form blur estimate or the motion analysis on very low-resolution image, up-sample and repeat.
  - Often a successful strategy for avoiding local minima in complicated estimation tasks.



Down-sampling



The Gaussian pyramid



g_0 = IMAGE

g_L = REDUCE [g_{L-1}]

Fig 1. A one-dimensional graphic representation of the process which generates a Gaussian pyramid Each row of dots represents nodes within a level of the pyramid. The value of each node in the zero level is just the gray level of a corresponding image pixel. The value of each node in a high level is the weighted average of node values in the next lower level. Note that node spacing doubles from level to level, while the same weighting pattern or "generating kernel" is used to generate all levels.

The Gaussian pyramid

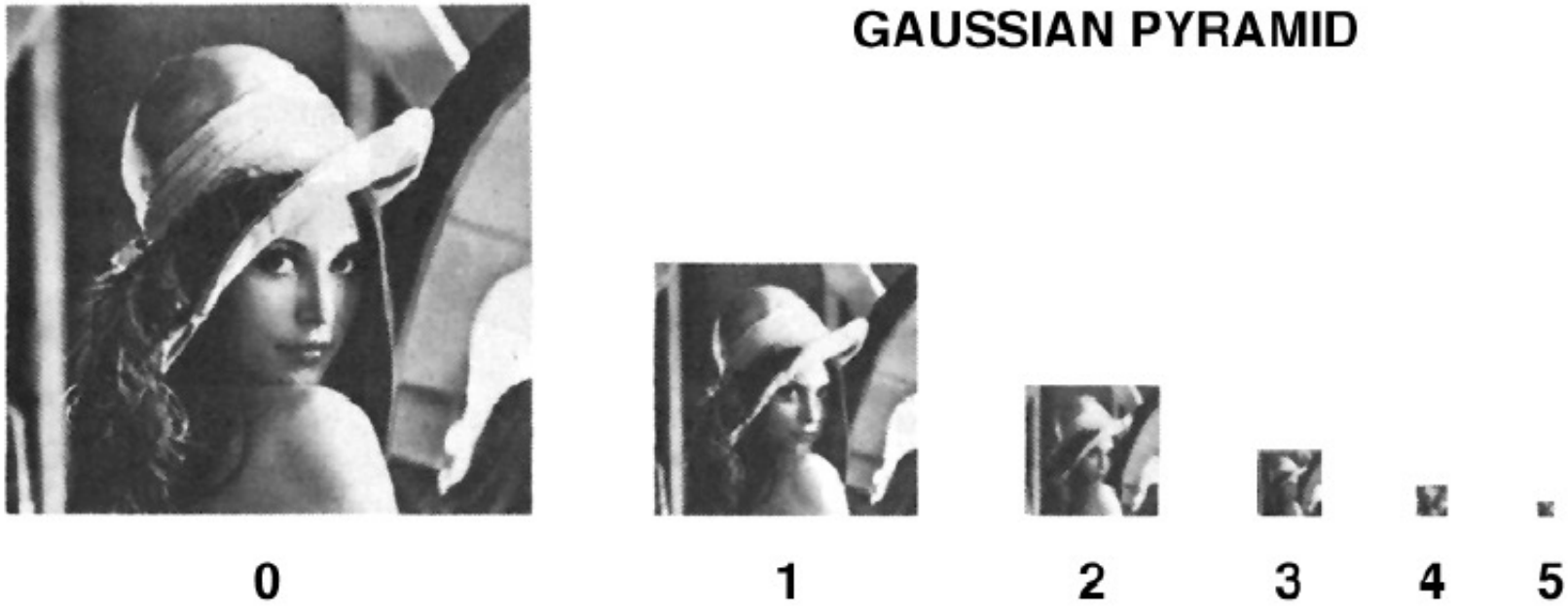


Fig. 4. First six levels of the Gaussian pyramid for the "Lady" image The original image, level 0, meusures 257 by 257 pixels and each higher level array is roughly half the dimensdons of its predecessor. Thus, level 5 measures just 9 by 9 pixels.

The Gaussian pyramid

512 256 128 64 32 16 8

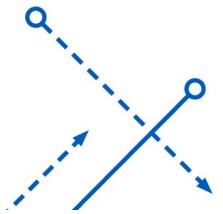
Matrix operation for Down-sampling (1D)

- Assume x_1 is a signal with 16 elements.
- Down-sampled signal x_2 can be expressed as

$$x_2 = G_1 x_1$$

$$G_1 = \begin{bmatrix}
1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0
\end{bmatrix}$$

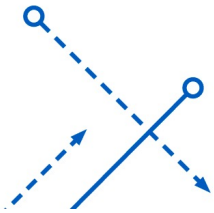
(Normalization constant of 1/16 omitted for visual clarity.)



Next pyramid level

$$x_3 = G_2 x_2$$

$$G_2 = \begin{matrix} 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \end{matrix}$$



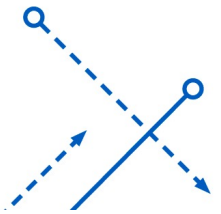
The combined effect of the two pyramid levels

- Smooth with Gaussians, because
  - a Gaussian * Gaussian = another Gaussian

$$x_3 = G_2 G_1 x_1$$

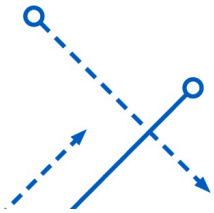
$$G_2 G_1 =$$

1	4	10	20	31	40	44	40	31	20	10	4	1	0	0	0	0	0	0	0
0	0	0	0	1	4	10	20	31	40	44	40	31	20	10	4	1	0	0	0
0	0	0	0	0	0	0	0	1	4	10	20	31	40	44	40	30	16	4	0
0	0	0	0	0	0	0	0	0	0	0	0	1	4	10	20	25	16	4	0



Up-sampling

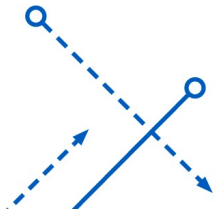
$$y_2 = F_3 x_3 \qquad F_3 = \begin{matrix} 6 & 1 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 4 \end{matrix}$$



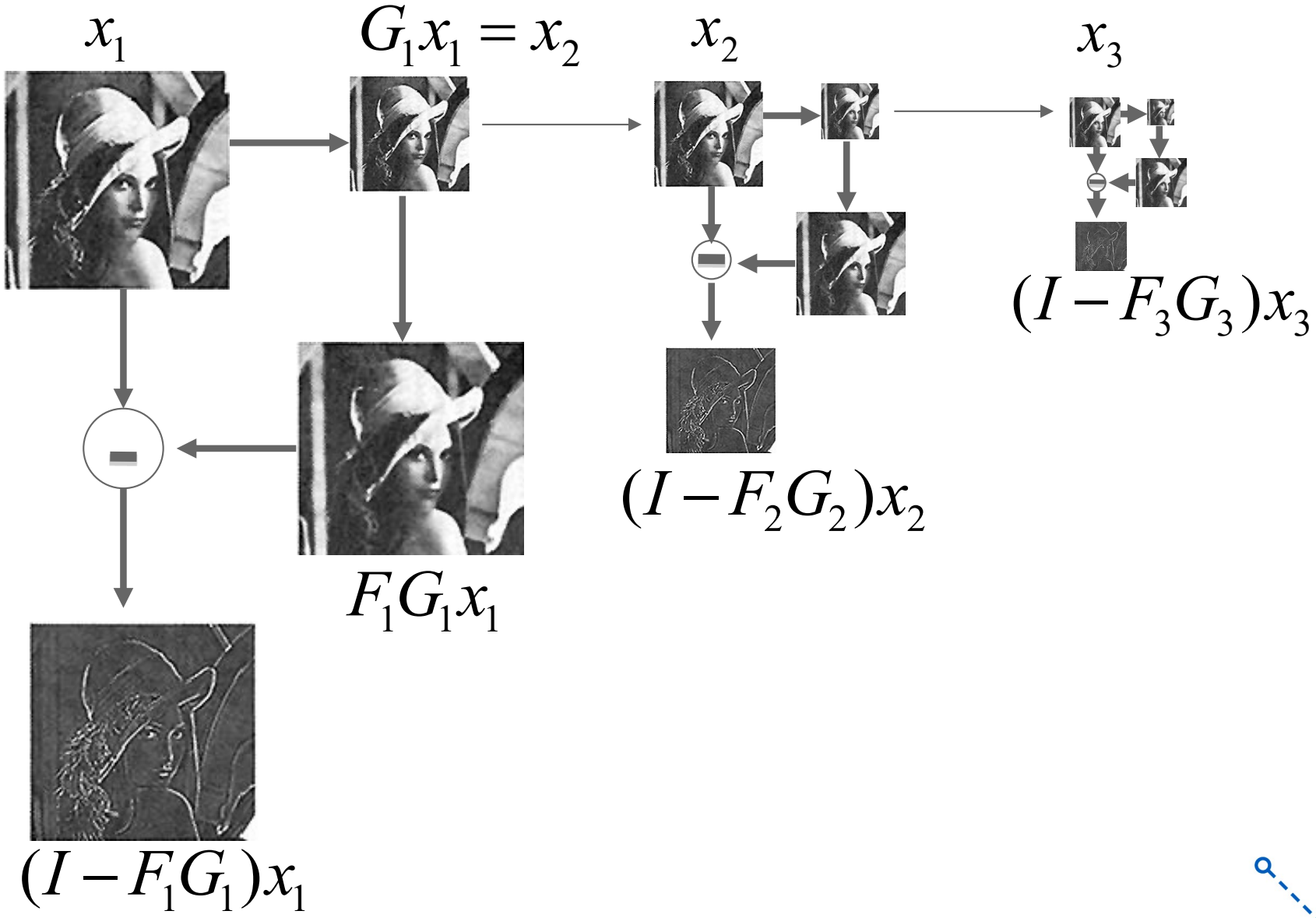
The Laplacian Pyramid

- **Difference between up-sampled Gaussian pyramid and Gaussian pyramid.**

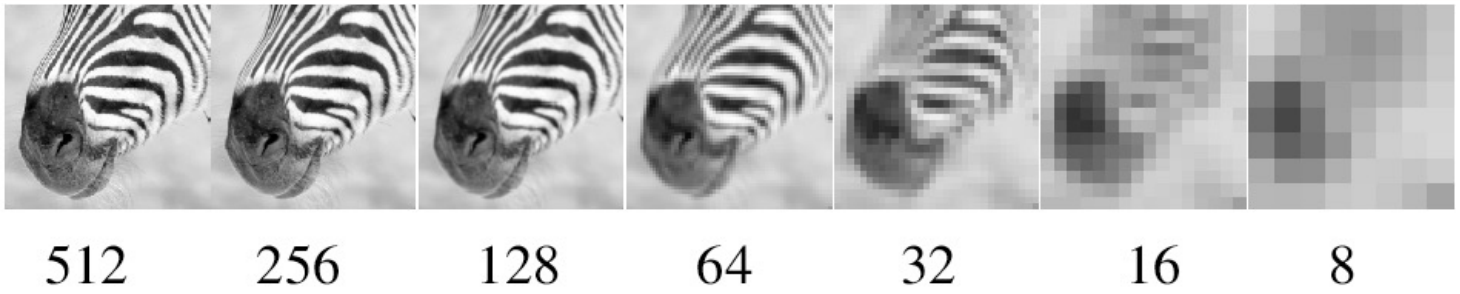
- **Band pass filter** - each level represents spatial frequencies (largely) unrepresented at other level.



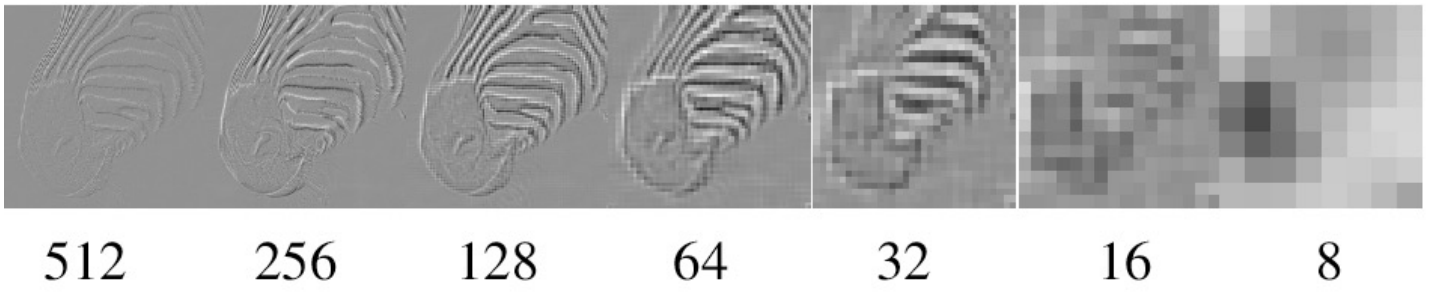
Laplacian pyramid algorithm

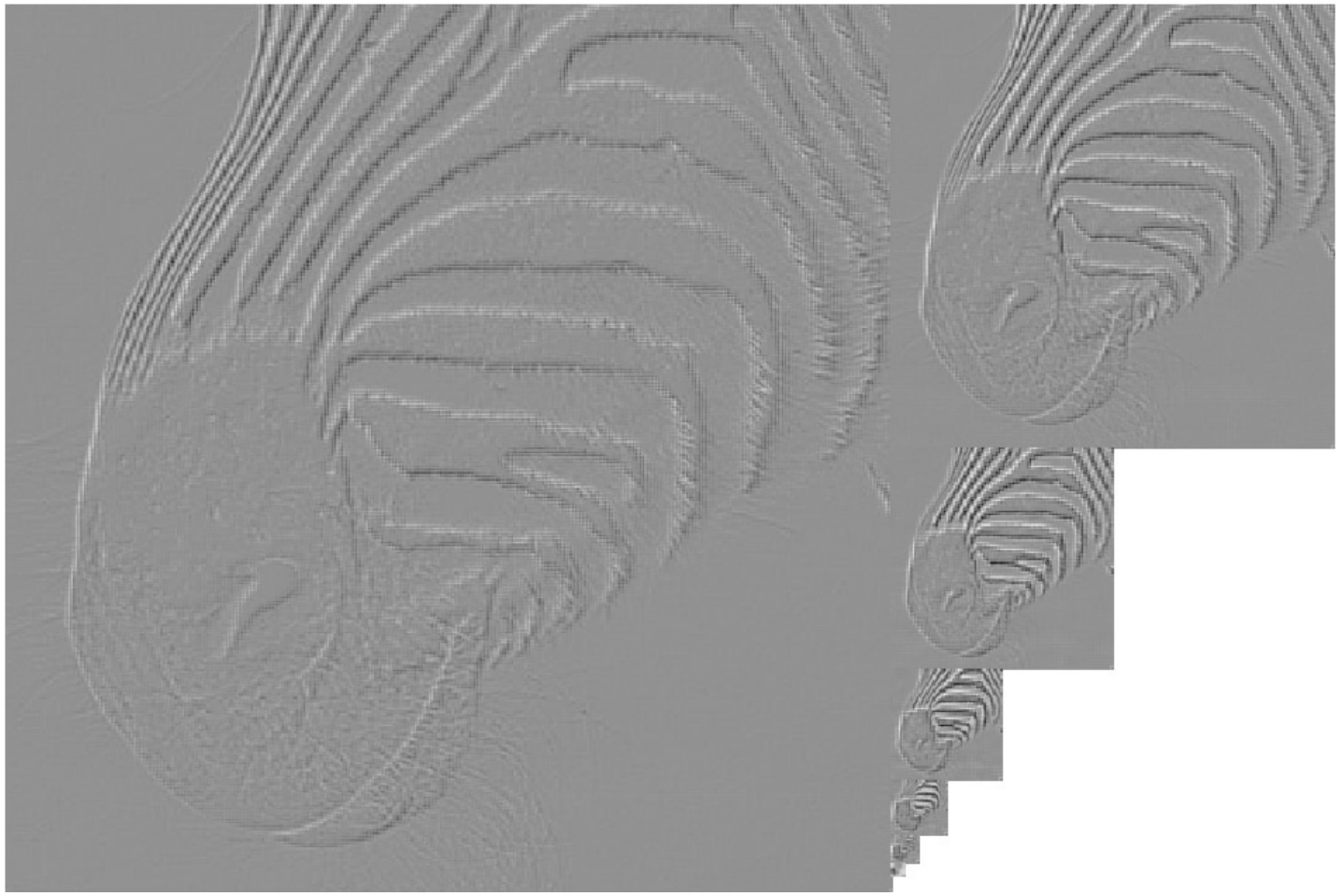


Gaussian pyramid



Laplacian pyramid





Laplacian Pyramid

- Information captured at each level of a Gaussian (top) and Laplacian (bottom) pyramid
  - showing full resolution.

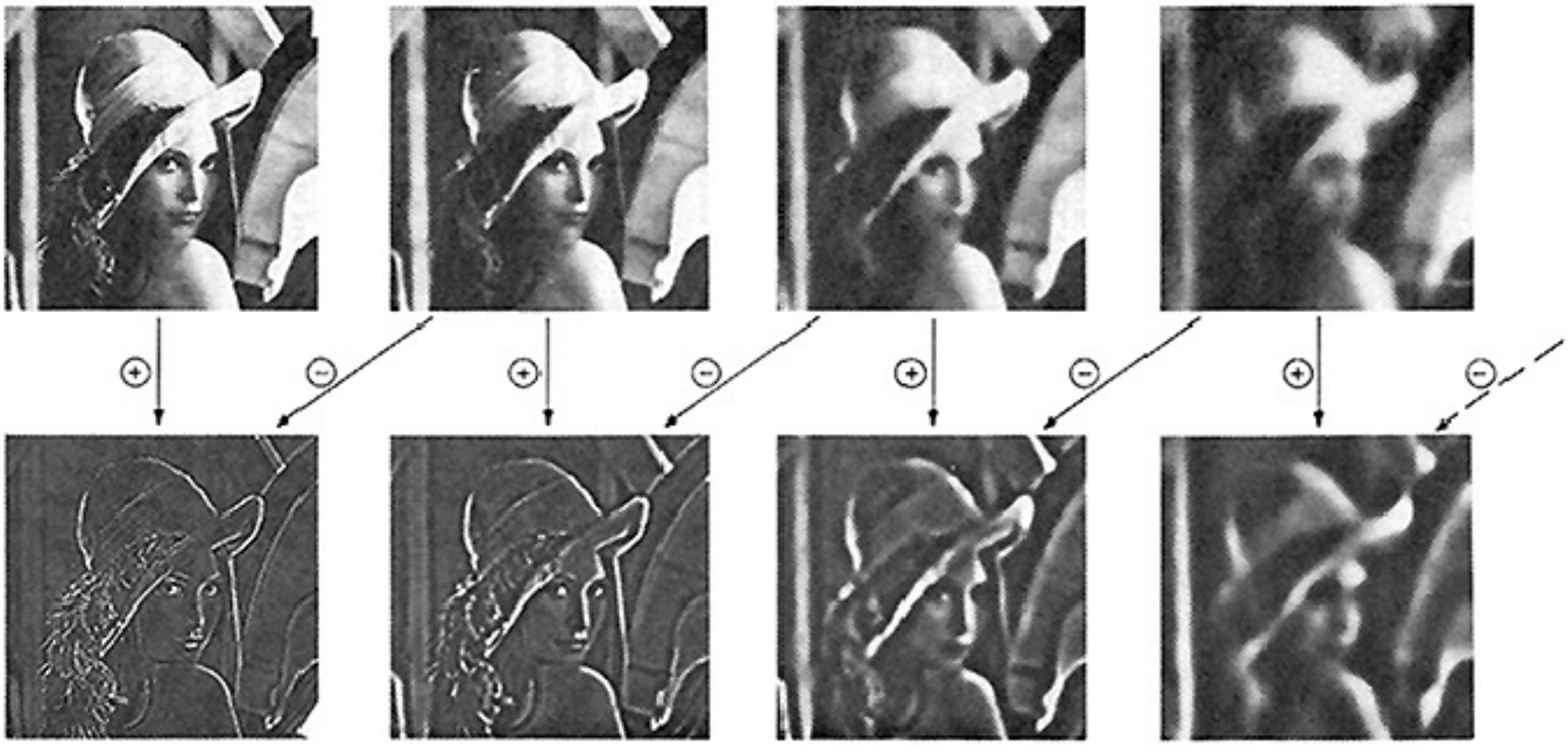
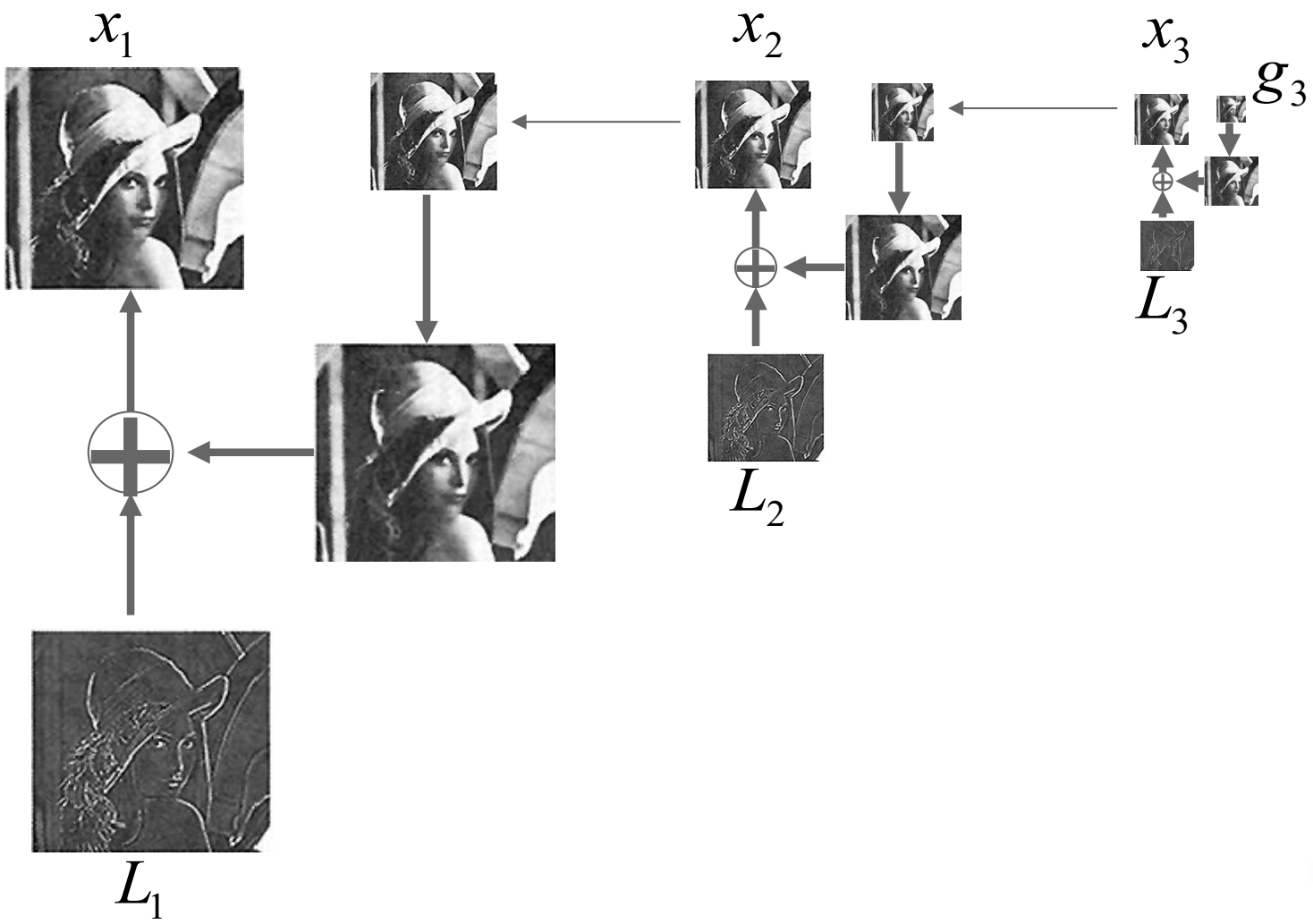


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the

Laplacian Pyramid

- Reconstruction: recover x_1 from L_1, L_2, L_3 and g_3



Laplacian Pyramid Reconstruction algorithm

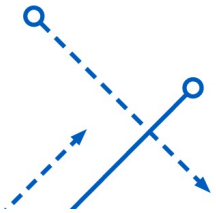
G# is the blur-and-downsample operator at pyramid level #
F# is the blur-and-upsample operator at pyramid level #

Laplacian pyramid elements:

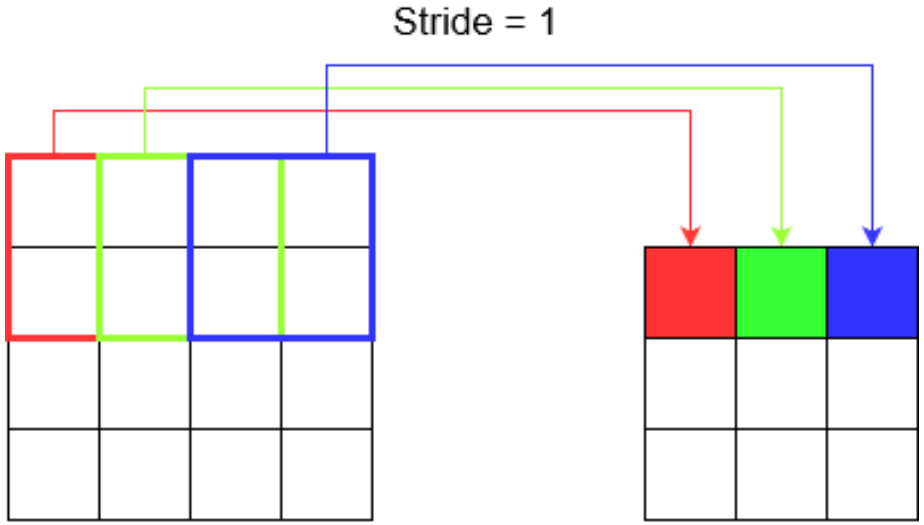
$L1 = (I - F1\ G1)\ x1$
$x2 = G1\ x1$
$L2 = (I - F2\ G2)\ x2$
$x3 = G2\ x2$
$L3 = (I - F3\ G3)\ x3$
$x4 = G3\ x3$

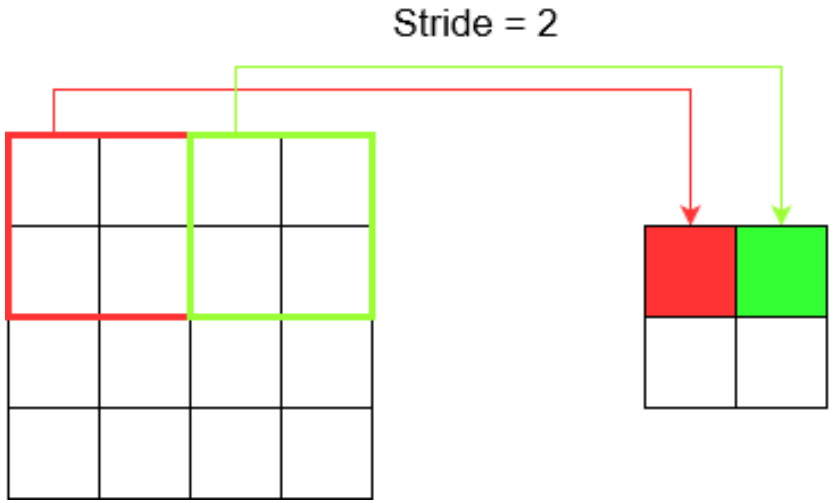
Reconstruction of original image (x1) from Laplacian pyramid elements:

$x3 = L3 + F3\ x4$
$x2 = L2 + F2\ x3$
$x1 = L1 + F1\ x2$

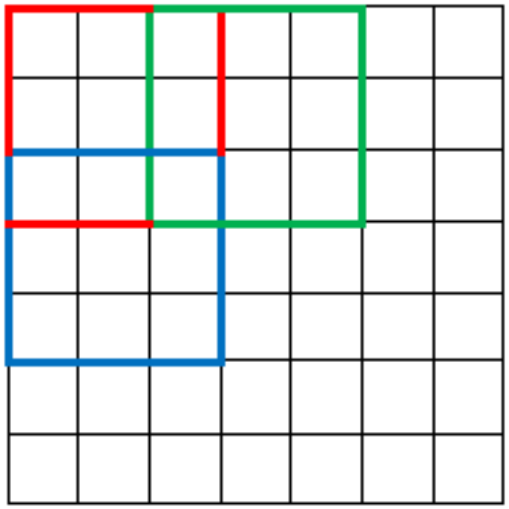


Convolution for Down-sampling

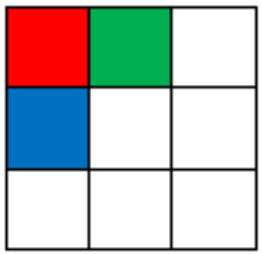




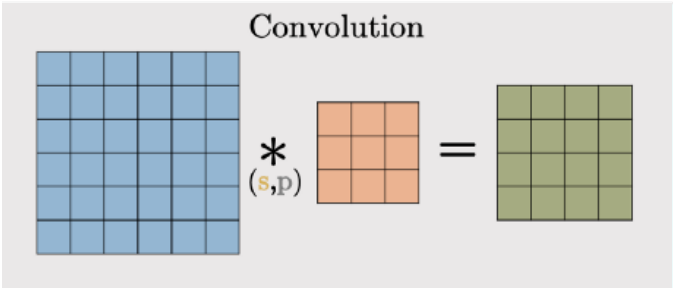
7 x 7 Input Volume

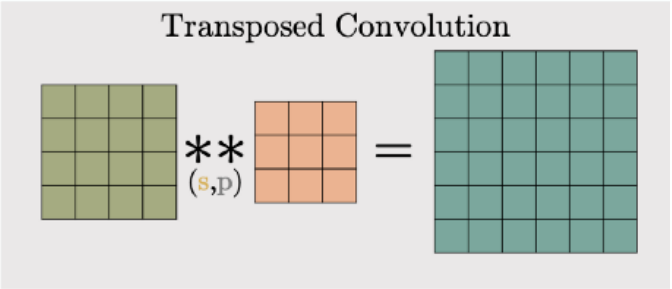


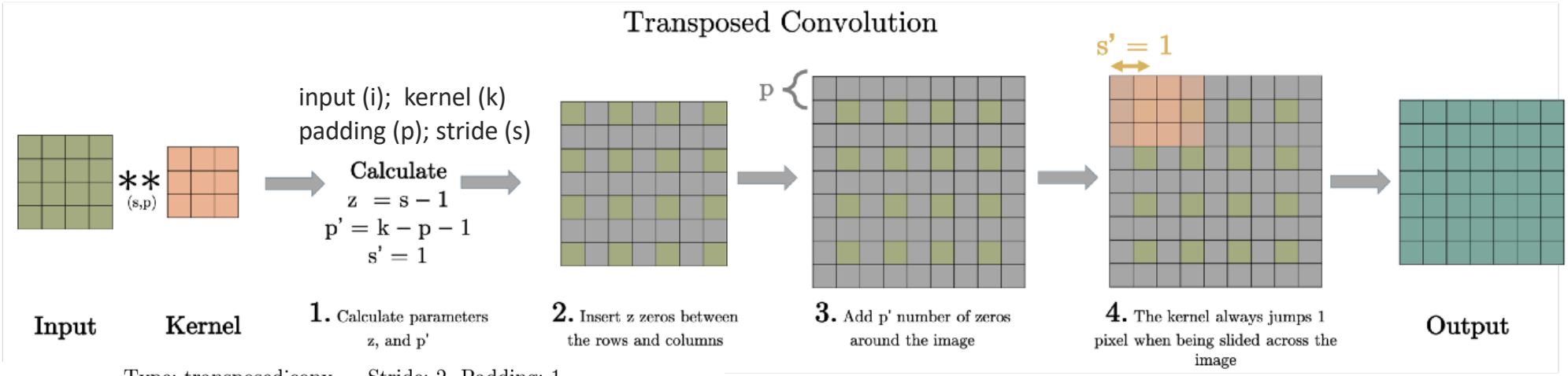
3 x 3 Output Volume



Transposed Convolution for Up-sampling







Type: transposed conv - Stride: 2 Padding: 1

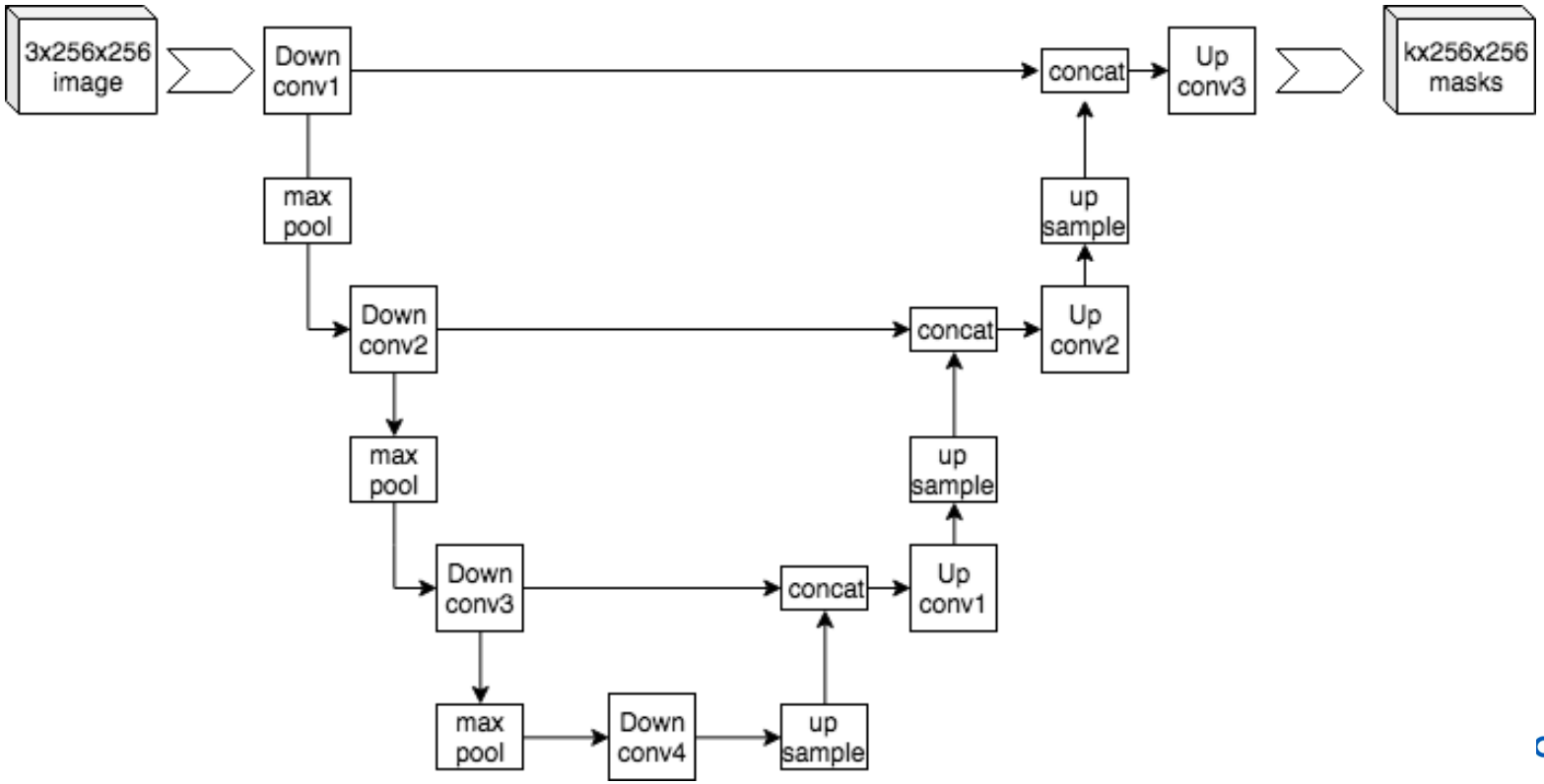

```
import torch, torch.nn.functional as F

inputs = torch.randn(2, 4, 5, 5)
kernel = torch.randn(4, 8, 3, 3)
F.conv_transpose2d(inputs, kernel, stride=2, padding=1)
```

Guess the shape of output. (2, 8, 9, 9)

https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

Why use these representations?

- Handle real-world size variations
- Remove noise, Analyze texture
- Recognize objects, Label image features



Shelhamer E, Long J, Darrell T (April 2017). "Fully Convolutional Networks for Semantic Segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **39** (4): 640–651

SAIR

Spatial AI & Robotics Lab

IMAGE PROCESSING

Histogram

Histogram

- Histograms plots how many times (frequency) each intensity value in image occurs
  - Image (left) has 256 distinct gray levels (8 bits)
  - Histogram (right) shows frequency (how many times) each gray level occurs

Histogram

- Many cameras display real time histograms of scene
  - Helps avoid taking over-exposed pictures

Histogram

- A histogram for a grayscale image with intensity values in range

$$I(u,v) \in [0, K-1]$$

  - would contain exactly K entries
- For 8-bit grayscale image, K = 2^8 = 256
- Each histogram entry is defined as:
  - h(i) = number of pixels with intensity i (0<i<K).
  - h(255) = number of pixels with intensity i = 255.

Formal definition

$$\mathrm{h}(i) = \mathrm{card}\{(u,v) \mid I(u,v) = i\}$$

Number (size of set) of pixels

such that

Normalized Histogram

Histogram $h(r_k) = n_k$

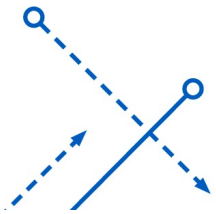
r_k is the k^{th} intensity value

n_k is the number of pixels in the image with intensity r_k

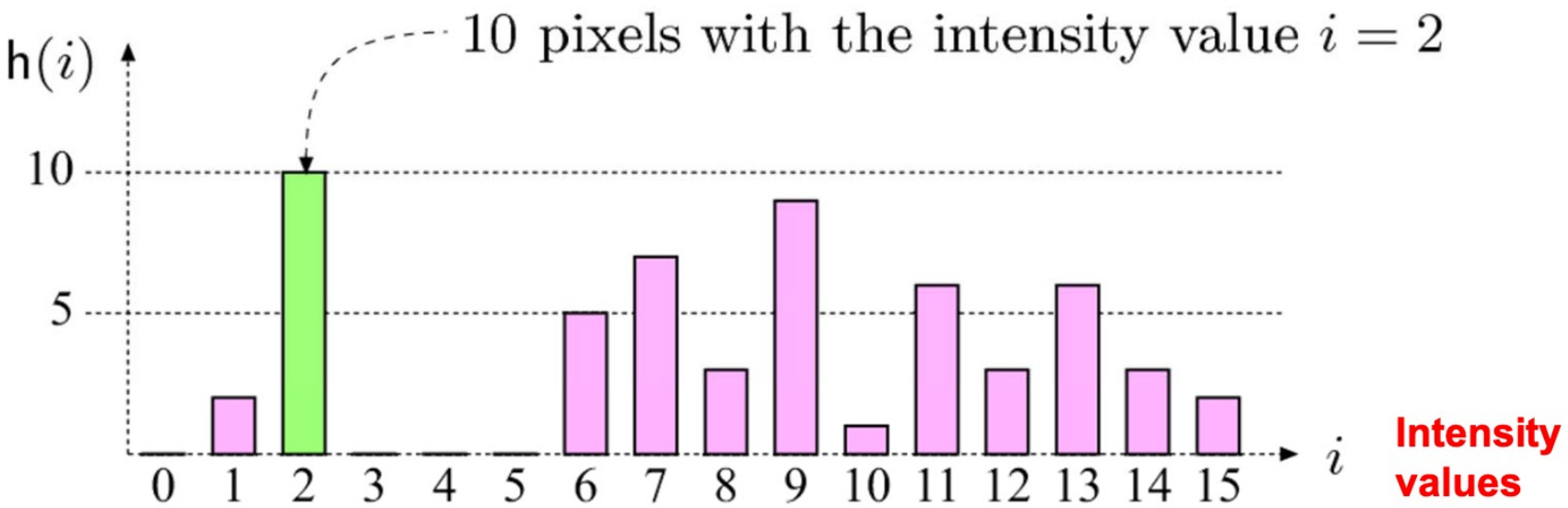
Normalized histogram $p(r_k) = \frac{n_k}{MN}$

n_k: the number of pixels in the image of size $M \times N$ with intensity r_k

Probability density function



Histogram

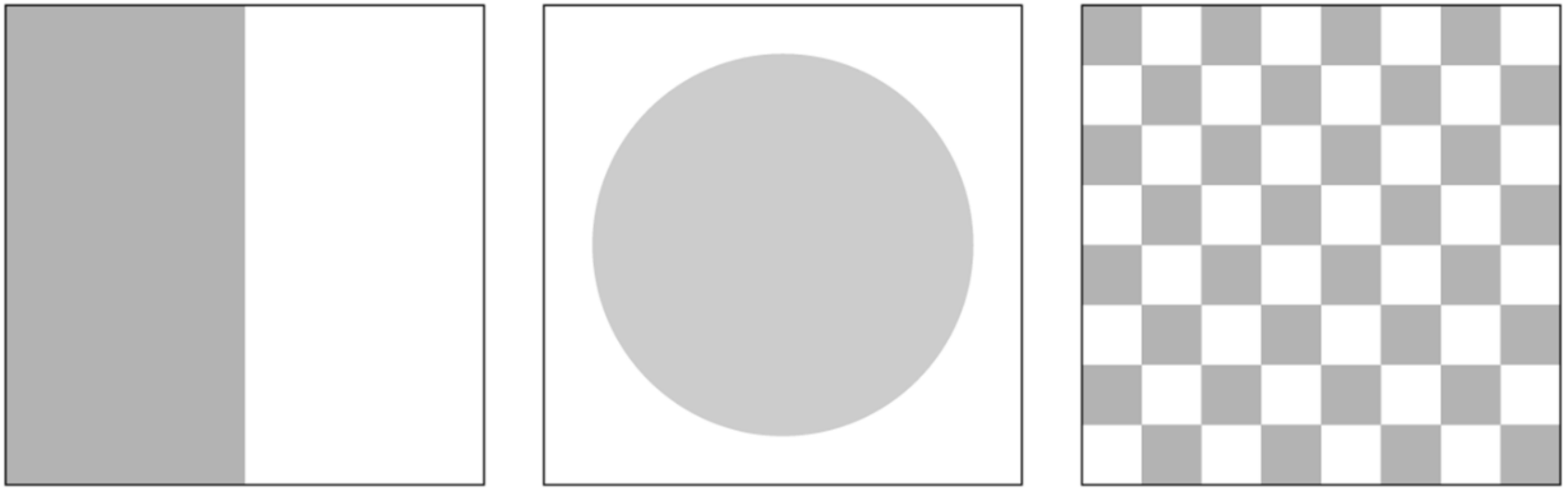


$h(i)$	0	2	10	0	0	0	5	7	3	9	1	6	3	6	3	2
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- E.g., K = 16, 10 pixels have intensity value = 2
- Histograms: only statistical information
- No indication of location of pixels

Histogram

- Different images can have same histogram
- 3 images below have same histogram



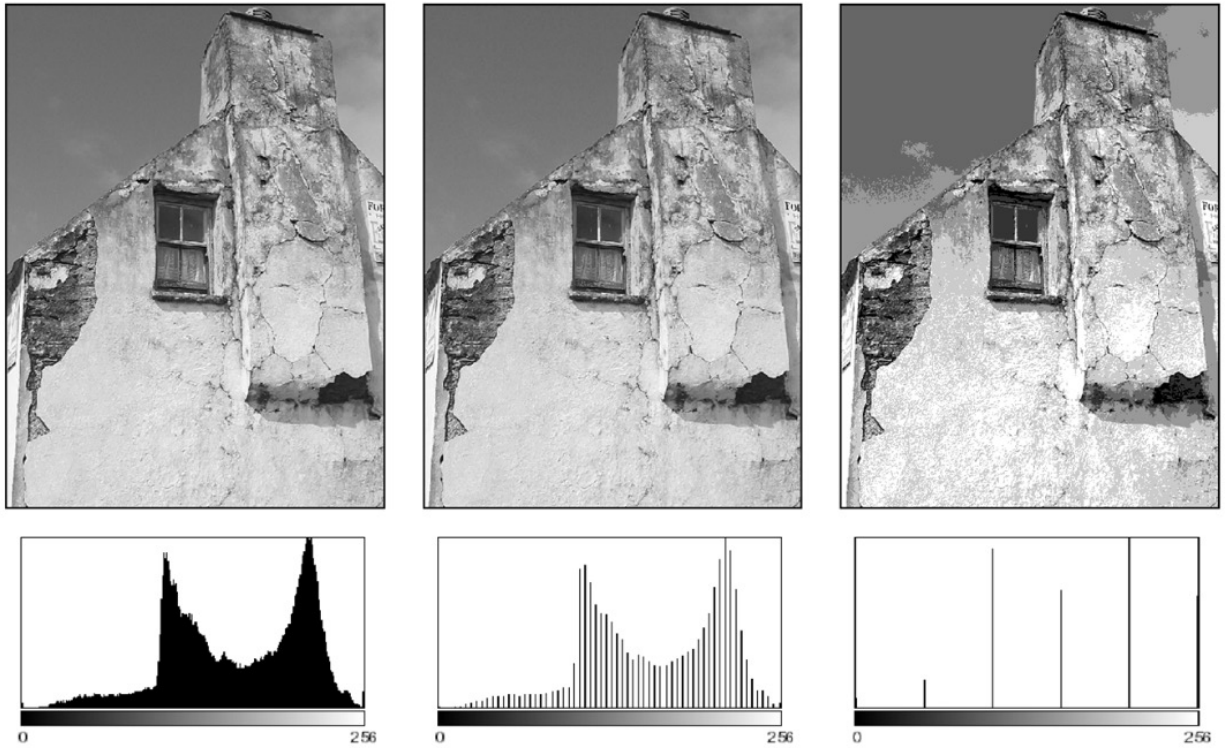
- Half of pixels are gray, half are white
  - Same histogram = same statistics
  - Distribution of intensities could be different
- Can we reconstruct image from histogram? No!

Histogram and Contrast

- What is Good Contrast?
  - Widely spread intensity values
  - Large difference between min and max intensity

Histogram and Dynamic Range

- Dynamic Range: number of distinct pixels in image
  - High dynamic range means very bright and very dark parts in a single image (many distinct values)



(a) **High Dynamic Range**

(b) **Low Dynamic Range (64 intensities)**

(c) **Extremely low Dynamic Range (6 intensity values)**

Large Histograms: Binning

- High resolution image can yield very large histogram
- E.g., 32-bit image = 2^{32} = 4,294,967,296 columns
  - Such a large histogram impractical to display
- Solution is Binning
  - Combine ranges of values into histogram columns

So, given the image $I : \Omega \rightarrow [0, K-1]$, the binned histogram for I is the function

$$h(i) = \text{card}\{(u,v) | \; a_i \leq I(u,v) < a_{i+1}\},$$

where $0 = a_0 < a_1 < \ldots < a_B = K.$

Number (size of set) of pixels

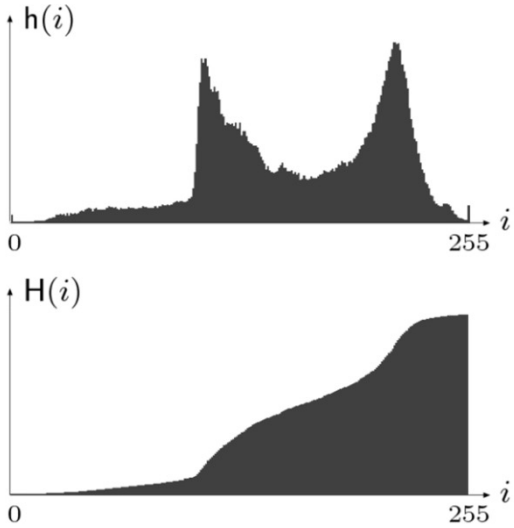
such that

Pixel's intensity is between a_i and a_{i+1}

Cumulative Histogram

- Useful for histogram equalization (introducing later)
- Similar to the Cumulative Density Function (CDF)
- Definition

$$\mathrm{H}(i) = \sum_{j=0}^{i} \mathrm{h}(j) \quad \text{for} \quad 0 \leq i < K$$



- Monotonically increasing

$$\mathrm{H}(K-1) = \sum_{j=0}^{K-1} \mathrm{h}(j) = M \cdot N$$

Last entry of Cum. histogram

Total number of pixels in image

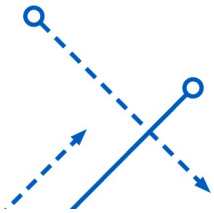
Point Operation

- Point operations changes a pixel's intensity value.

$$a' \leftarrow f(a)$$

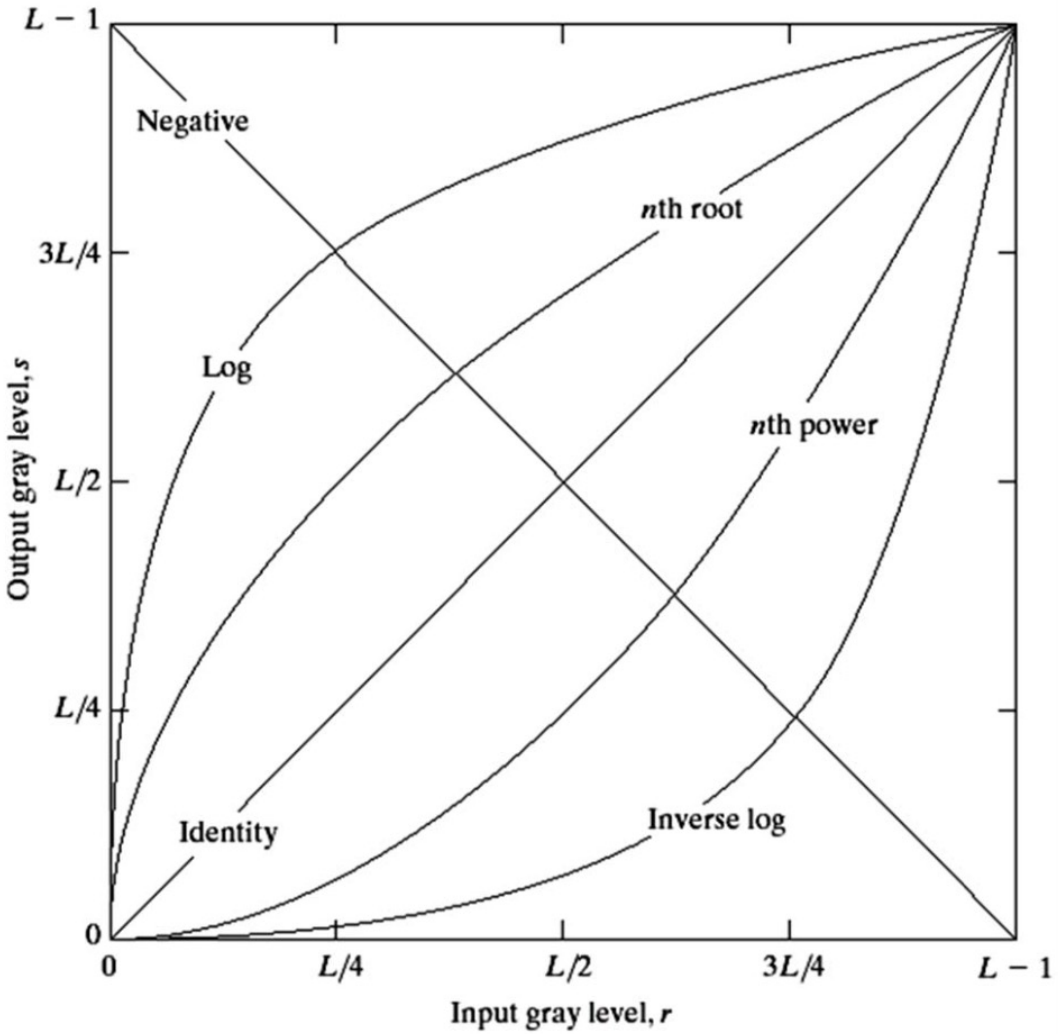
$$I'(u,v) \leftarrow f\big(I(u,v)\big)$$

- New pixel intensity depends on
  - Pixel's previous intensity I(u, v)
  - Mapping function f( )
- Does not depend on
  - Pixel's location (u, v)
  - Intensities of neighboring pixels



Basic Grey Level Point Operation

- 3 most common gray level operation
  - Linear
    - Negative/Identity
  - Logarithmic
    - Log/Inverse log
  - Power law
    - nth power/nth root



Power Law Transformations

- Power law transformations have the form

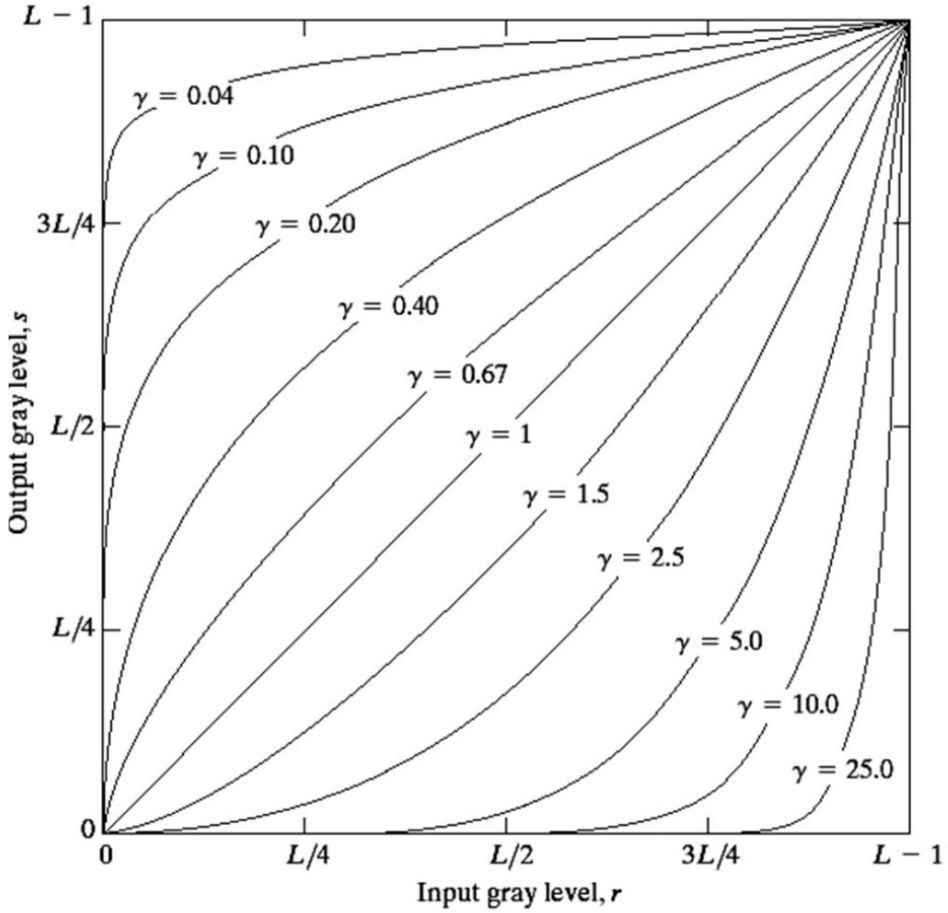
$$s = c * r^{\gamma}$$

Power

New pixel value

Constant

Old pixel value



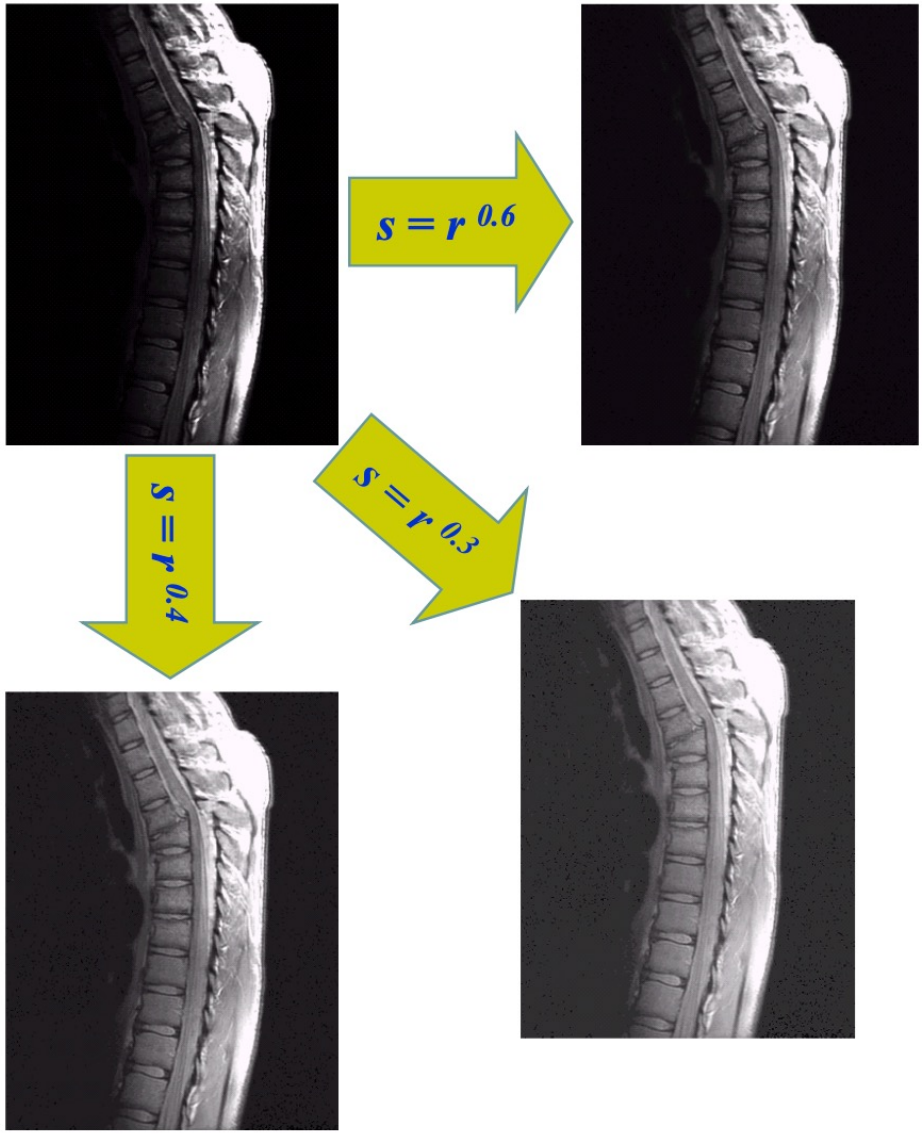
- Map narrow range of dark input values into wider range of output values or vice versa

- Varying γ gives a whole family of curves

Power Law Example

- Magnetic Resonance (MR) image of fractured human spine

- Different power values highlight different details

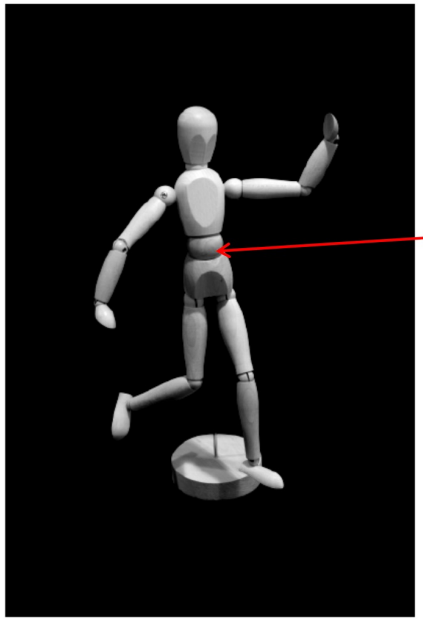


Intensity Windowing

- A clamp operation, then linearly stretching image intensities to fill possible range
- To window an image in [a,b] with max intensity M

$$f(p) = \begin{cases} 0 & \text{if } p < a \\ M \times \frac{p-a}{b-a} & \text{if } a \le p \le b \\ M & \text{if } p > b \end{cases}$$

Original Image



Windowed Image

Contrasts easier to see

Automatic Contrast Adjustment

- Modify pixel intensities such that the available range of range of pixels is full covered.
  - Algorithm
    - Find high and lowest pixel intensities, a_{low} and b_{high}
    - Linear stretching the intensity range.

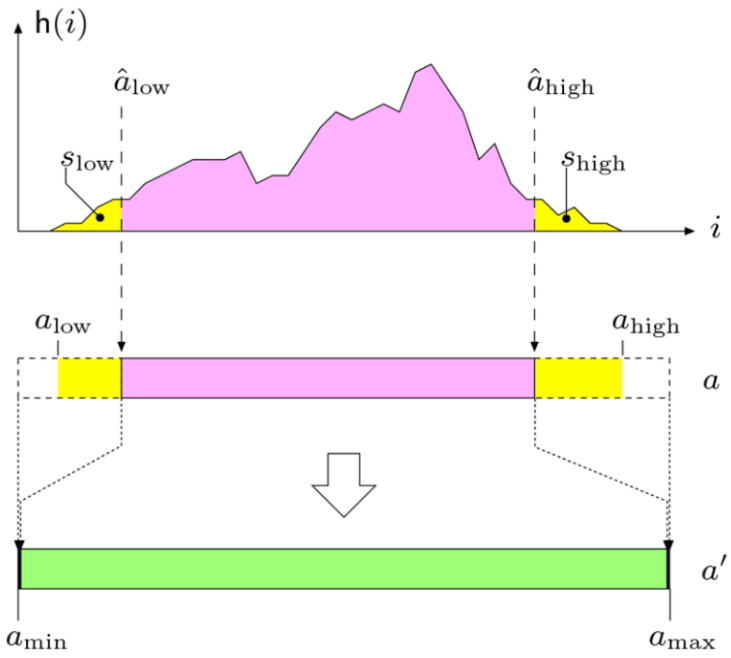
$$f_{\text{ac}}(a) = a_{\text{min}} + (a - a_{\text{low}}) \cdot \frac{a_{\text{max}} - a_{\text{min}}}{a_{\text{high}} - a_{\text{low}}}$$

If a_{min} = 0 and a_{max} = 255

$$f_{\text{ac}}(a) = (a - a_{\text{low}}) \cdot \frac{255}{a_{\text{high}} - a_{\text{low}}}$$

Modified Contrast Adjustment

- Better to map only certain range of values.
- Get rid of tails (usually noises), with predefined percentiles s_{low} and s_{high}

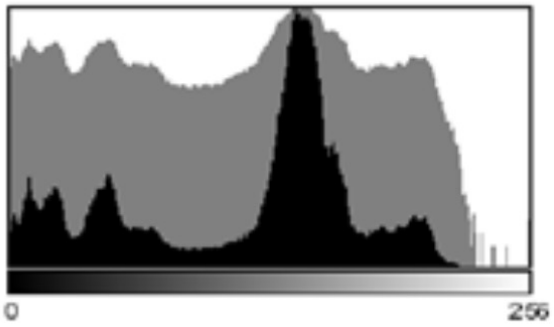


$$\hat{a}_{\text{low}} = \min\{\, i \mid \mathsf{H}(i) \geq M \cdot N \cdot s_{\text{low}} \}$$

$$\hat{a}_{\text{high}} = \max\{\, i \mid \mathsf{H}(i) \leq M \cdot N \cdot (1 - s_{\text{high}}) \}$$

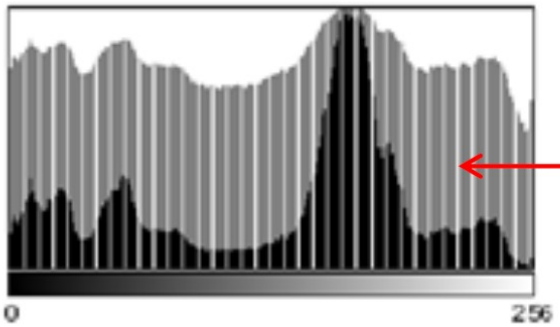
$$f_{\text{mac}}(a) = \begin{cases} a_{\text{min}} & \text{for } a \leq \hat{a}_{\text{low}} \\ a_{\text{min}} + (a - \hat{a}_{\text{low}}) \cdot \dfrac{a_{\text{max}} - a_{\text{min}}}{\hat{a}_{\text{high}} - \hat{a}_{\text{low}}} & \text{for } \hat{a}_{\text{low}} < a < \hat{a}_{\text{high}} \\ a_{\text{max}} & \text{for } a \geq \hat{a}_{\text{high}} \end{cases}$$

Effects of Automatic Contrast Adjustment



(a)

Original



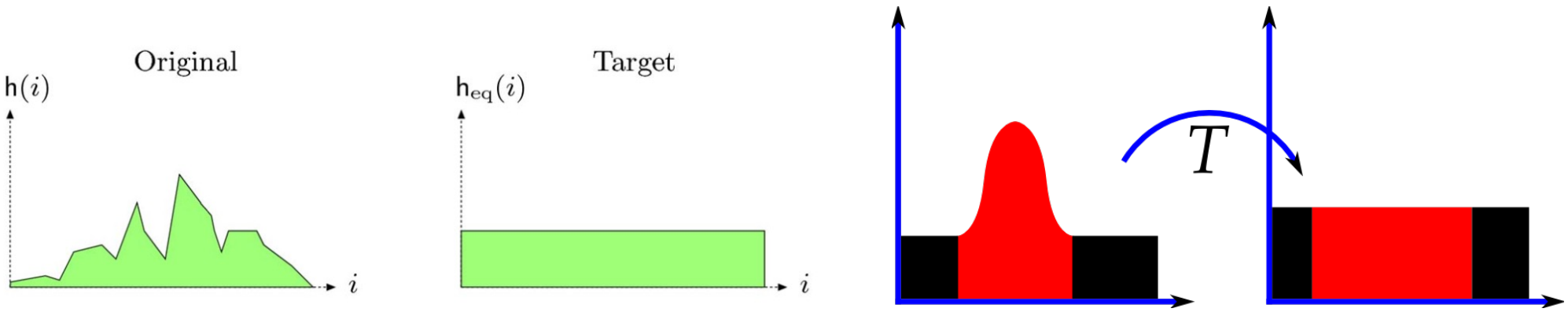
(b)

Result of automatic Contrast Adjustment

Linearly stretching range causes gaps in histogram

Histogram Equalization

- Adjust 2 different images to make their histograms (intensity distributions) similar
- Apply a point operation that changes histogram of modified image into uniform distribution.



Histogram Equalization: Algorithm

- Normalized Histogram

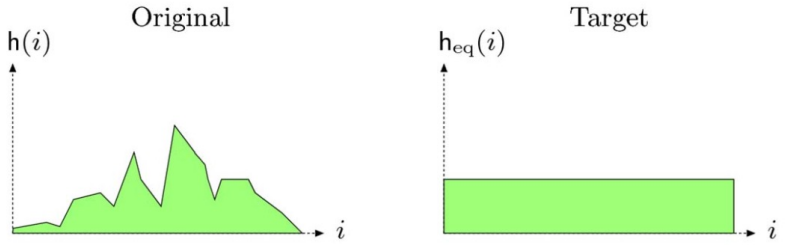
$$p_x(i) = p(x = i) = \frac{n_i}{n}, \quad 0 \leq i < L$$

- Cumulated Normalized Histogram

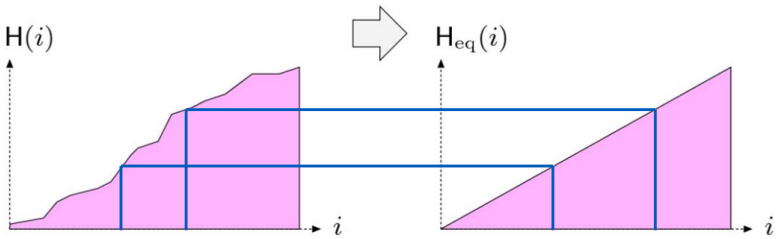
$$\mathrm{cdf}_x(i) = \sum_{j=0}^{i} p_x(x = j),$$

- A transform for histogram equalization, WHY?

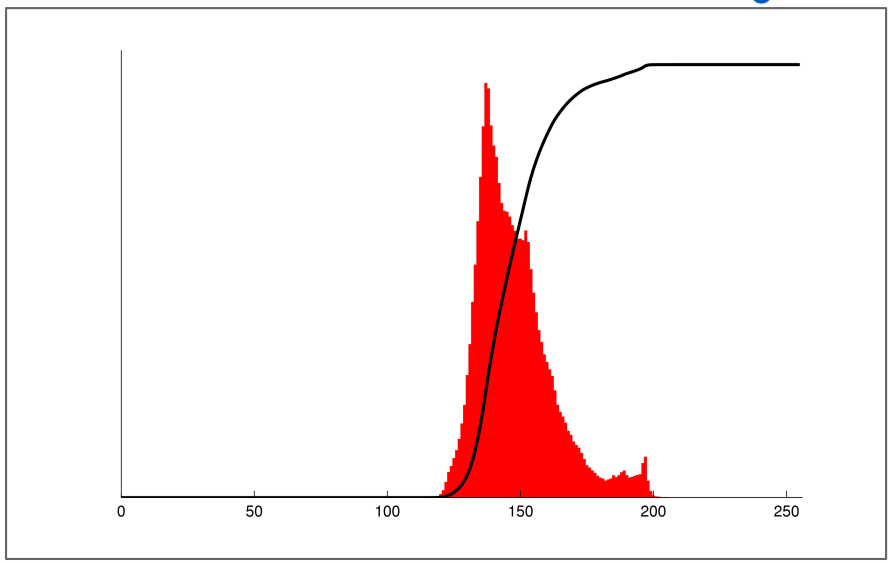
$$y = T(k) = \mathrm{cdf}_x(k)$$

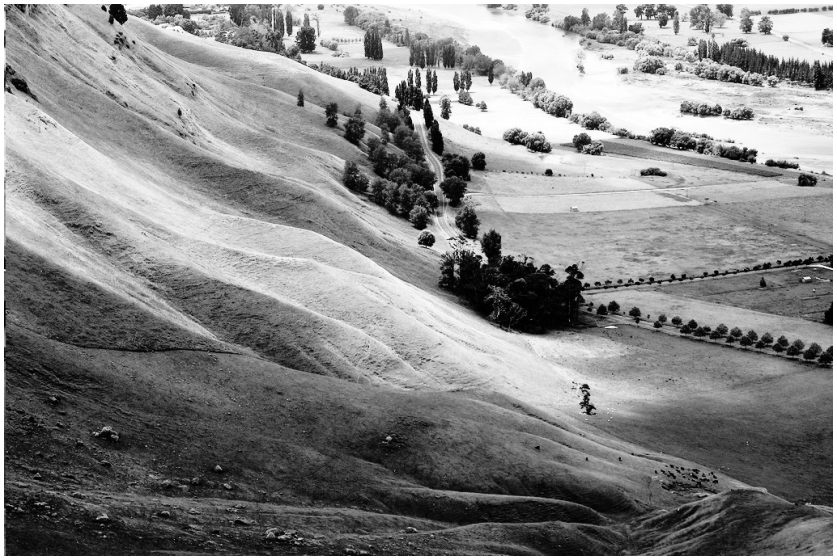


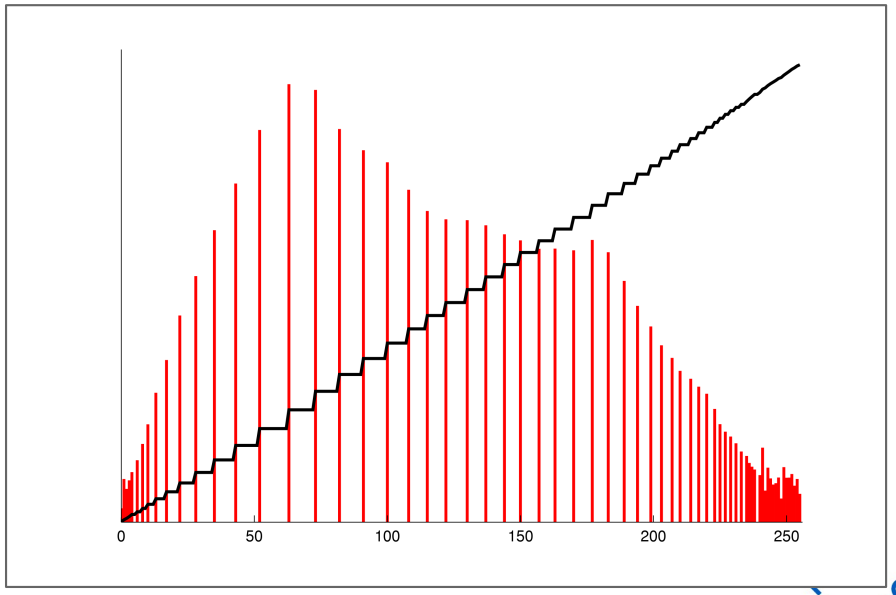
Cumulative Histogram



Histogram Equalization Example







Detailed explanation and proof are here.

SAIR

Spatial AI & Robotics Lab

IMAGE PROCESSING

Feature Detection & Matching

Feature Detection and Matching

- Local features
- Pyramids for invariant feature detection
- Invariant descriptors
- Matching

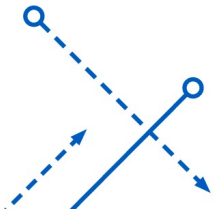


Image matching

by Diva Sian

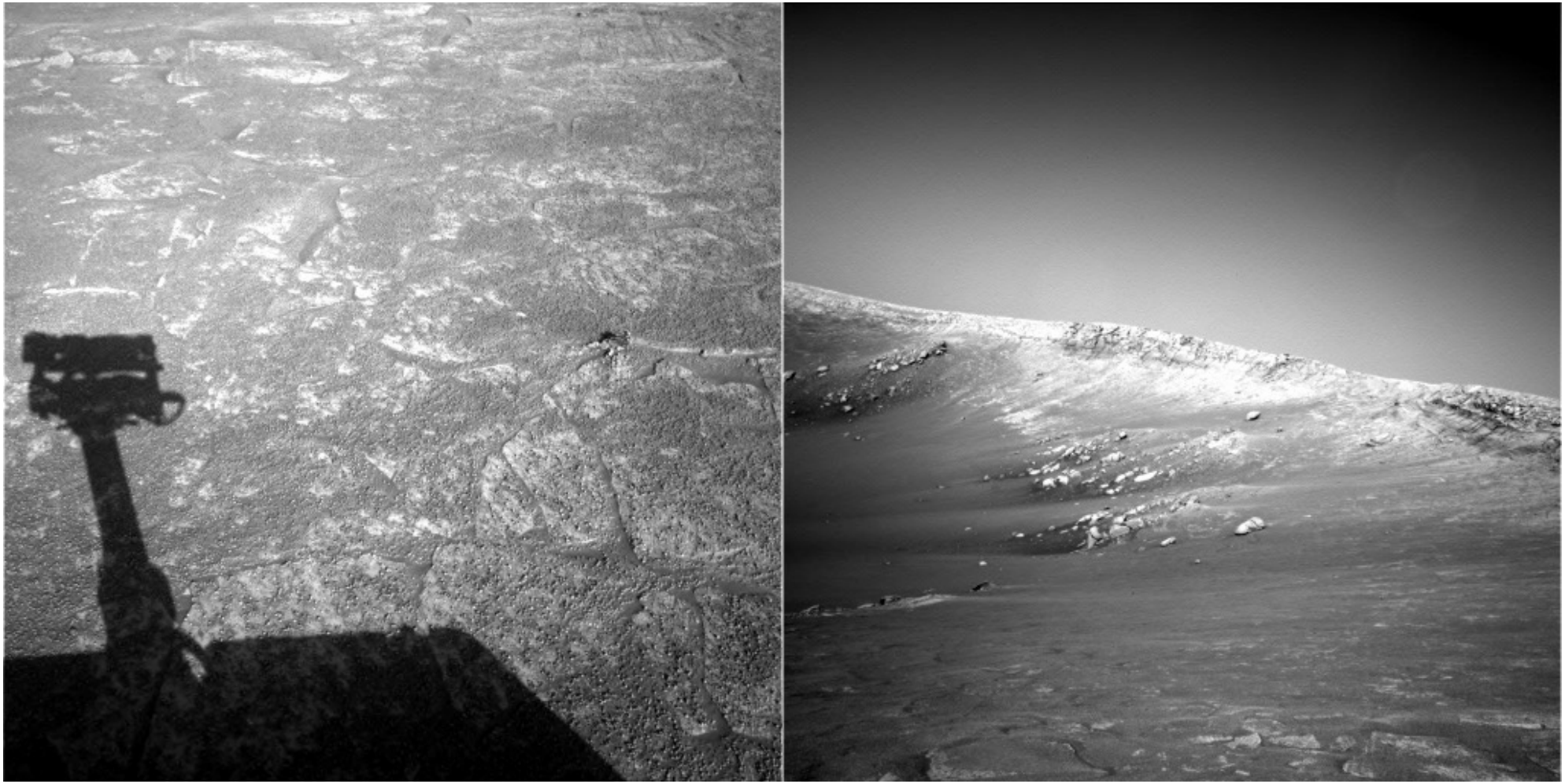
by swashford

Harder case

by Diva Sian

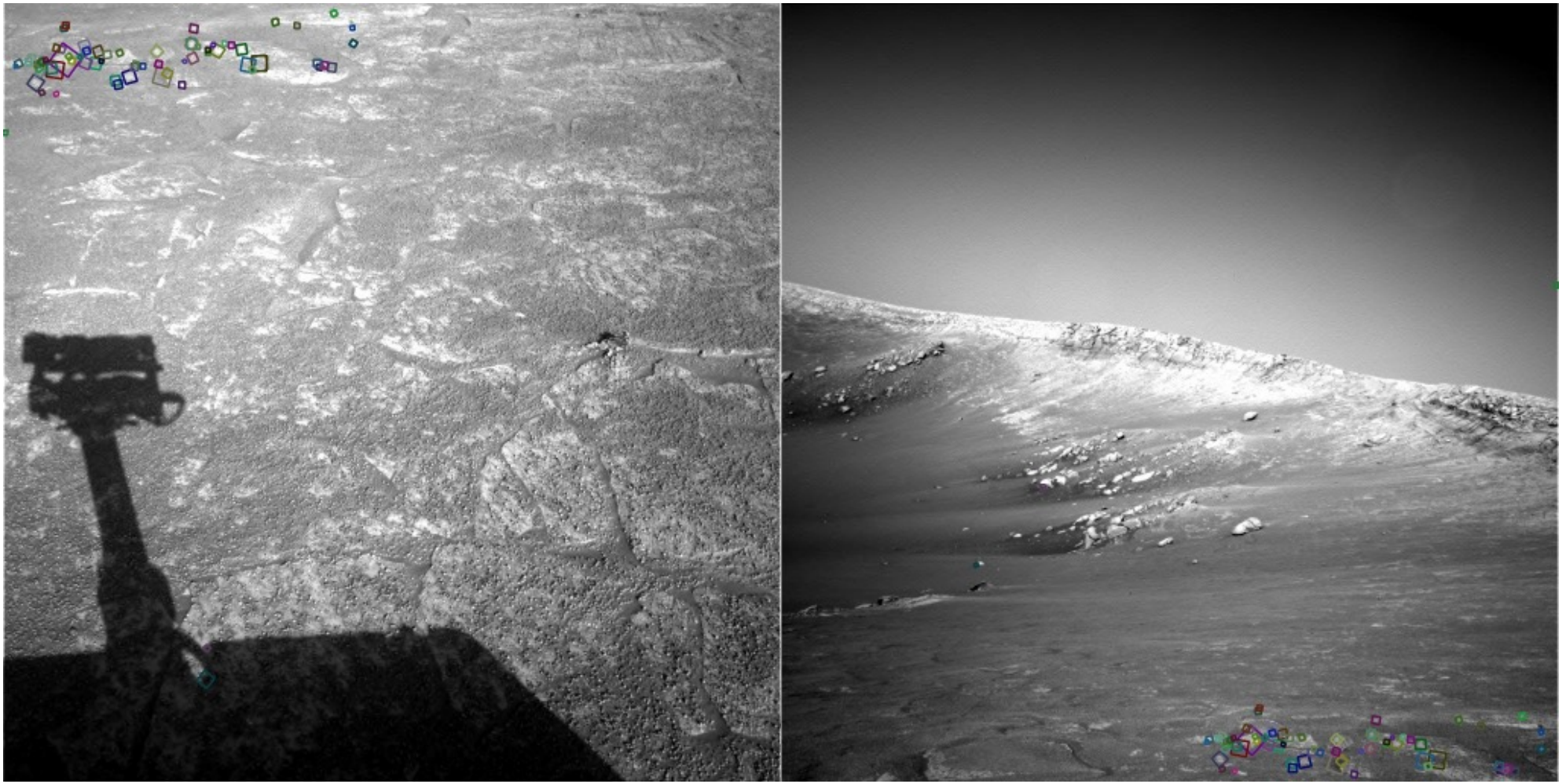
by scgbt

Harder still?



NASA Mars Rover images

Answer below (look for tiny colored squares…)



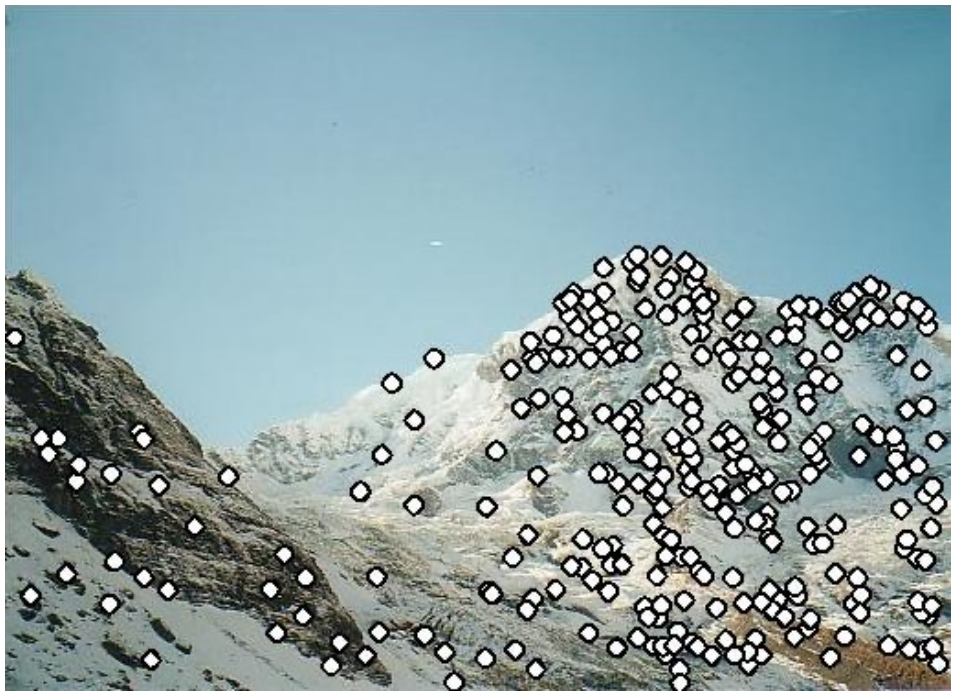
NASA Mars Rover images with SIFT feature matches
Figure by Noah Snavely

Local features and alignment

- We need to match (align) images
- Global methods sensitive to occlusion, lighting, parallax.
- Look for local features that match well.
- How would you do it by eye?

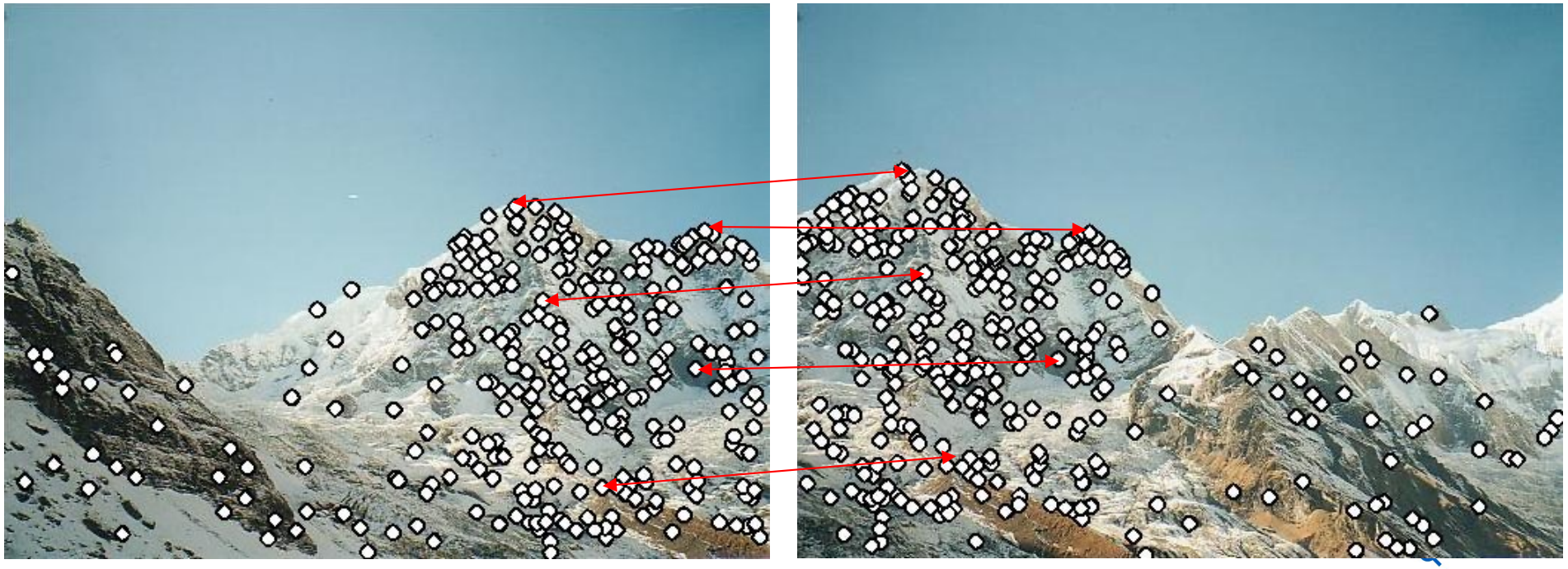
Local features and alignment

- 1. Detect feature points in both images.



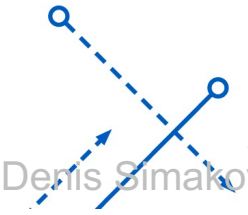
Local features and alignment

- 1. Detect feature points in both images.
- 2. Find corresponding pairs.



Local features and alignment

- 1. Detect feature points in both images.
- 2. Find corresponding pairs.
- 3. Use these pairs to align images



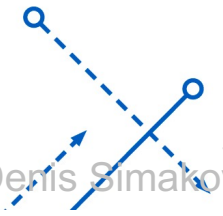
[Darya Frolova and Denis Simakov]

Local features and alignment

- Problem 1:
  - Detect the *same* points *independently* in both images

no chance to match!

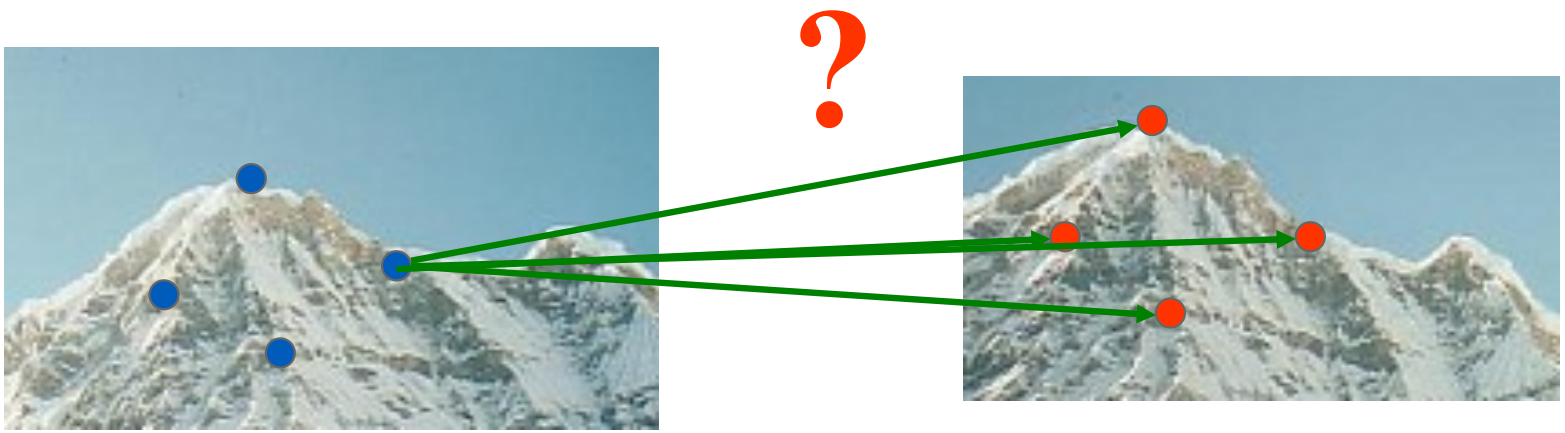
We need a repeatable detector



[Darya Frolova and Denis Simakov]

Local features and alignment

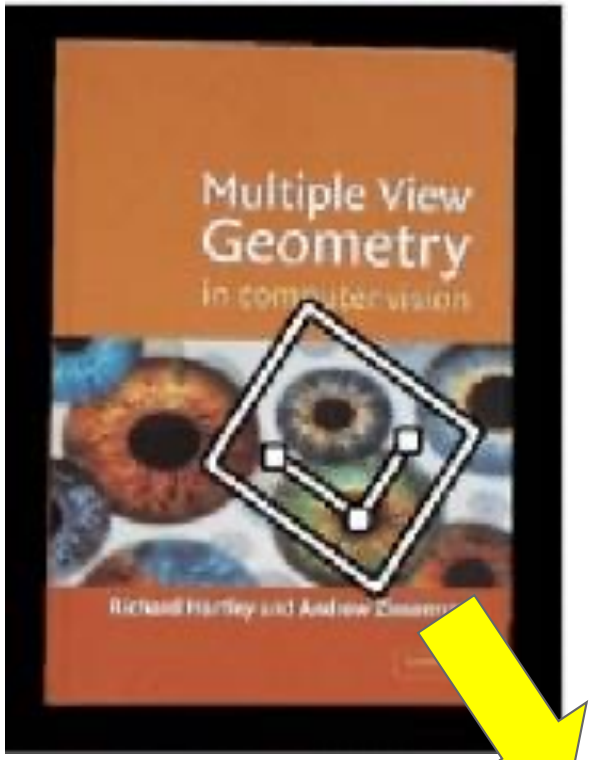
- Problem 2:
  - For each point correctly recognize the corresponding one

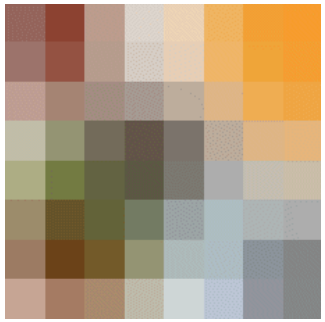


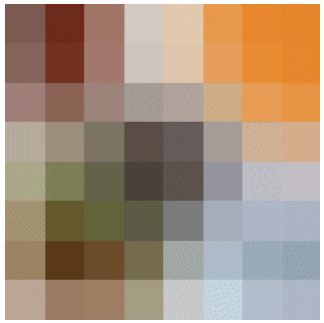
We need a reliable and distinctive descriptor

[Darya Frolova and Denis Simakov]

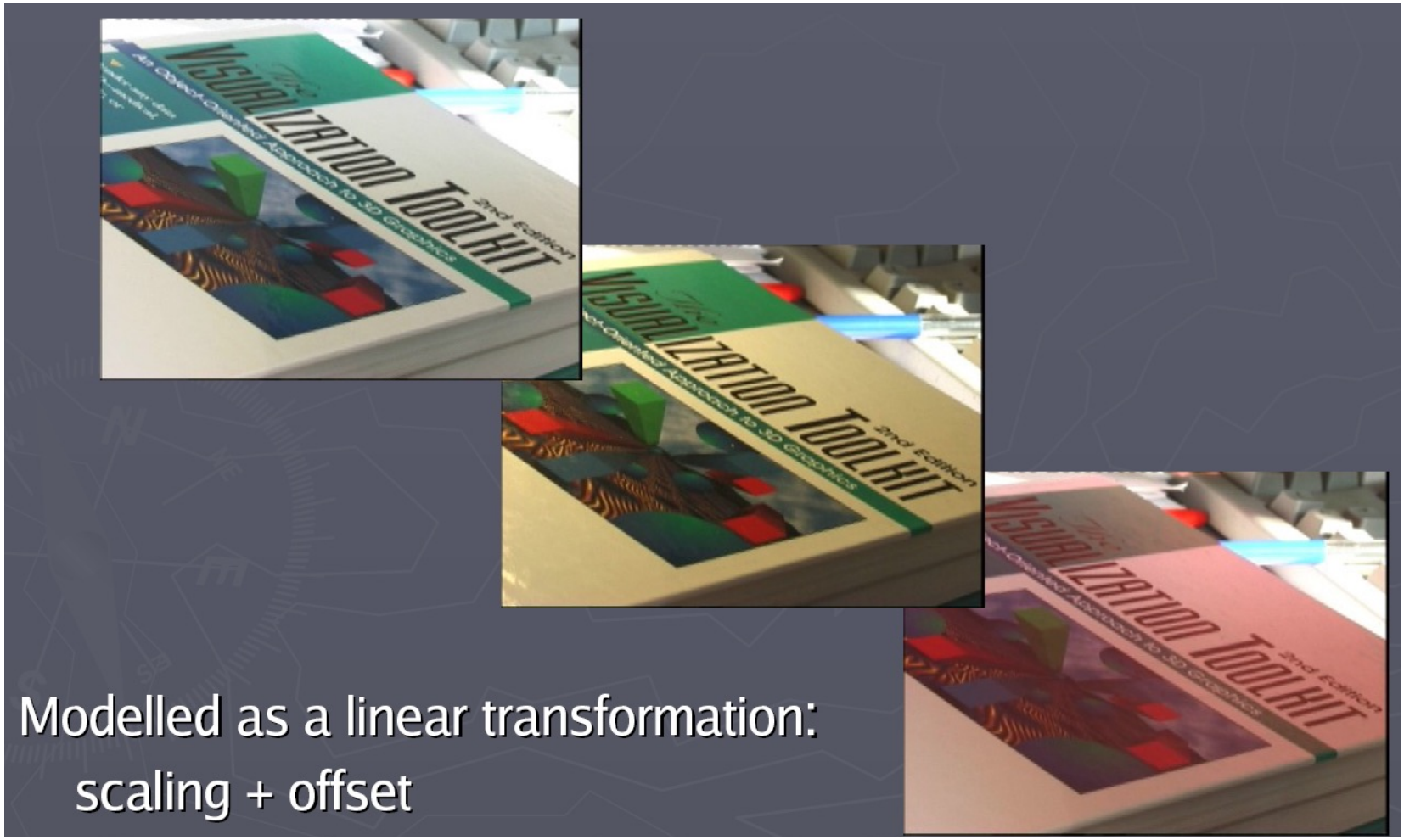
Geometric transformations







Photometric transformations



Other challenges: Noise, Blur, Compression, Artifacts, etc.

Figure from T. Tuytelaars ECCV 2006 tutorial

Invariant local features

Subset of local feature types designed to be invariant to common geometric and photometric transformations.

Basic steps:

1) Detect distinctive interest points
2) Extract invariant descriptors

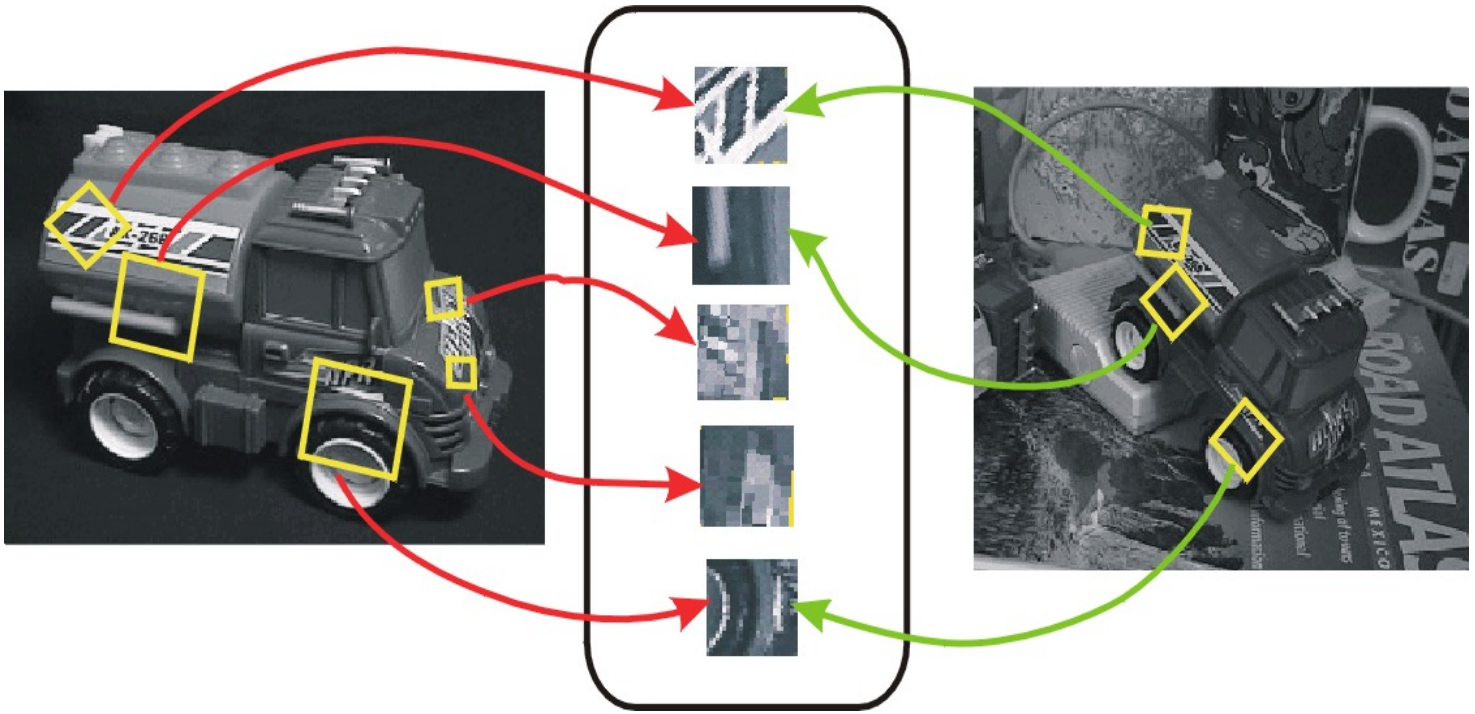
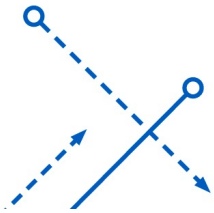


Figure: David Lowe

Main questions

- Where will the interest points come from?
- What are salient features that we'll *detect* in multiple views?
- How to *describe* a local region?
- How to establish *correspondences*, i.e., compute matches?



Feature Vectors – Image Templates

- How do we do image template matching (correlation)?
  - Given an image and a template, pass the template over the image
  - Find the max response over the image

- What are the challenges using image templates?
  - Does not generalize
  - Scale dependent
  - Must match at all locations in image.
  - Must match each template separately

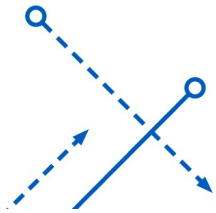
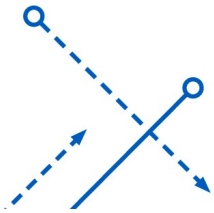


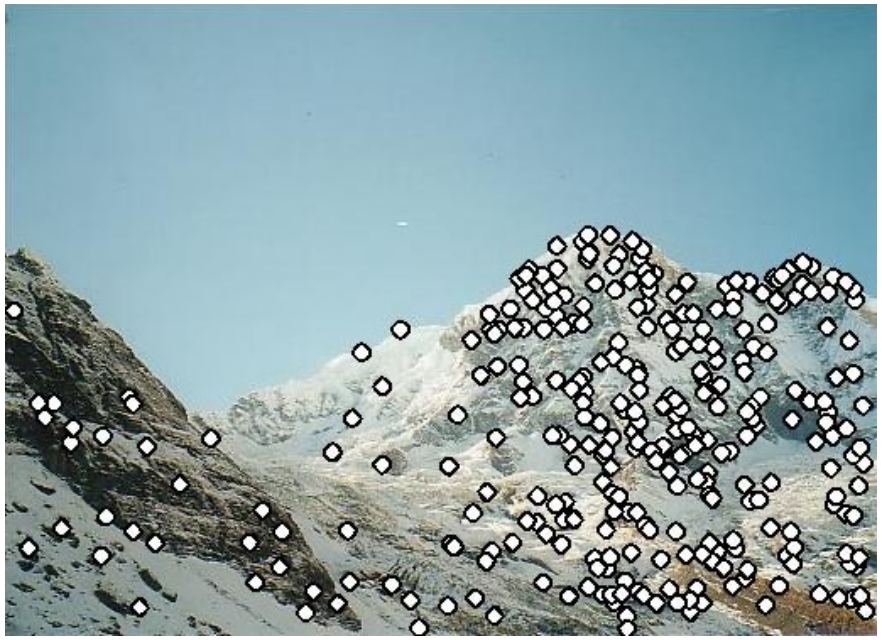
Image Templates – Improvements*

- Scale Dependency?
  - Rescale the images and templates
- Matching all locations?
  - Change the Stride of matching
- Match templates separately?
- Other Efficiency Issues
  - Extract and match other features
    - –smaller feature vector
  - Efficient Image Computation (later)

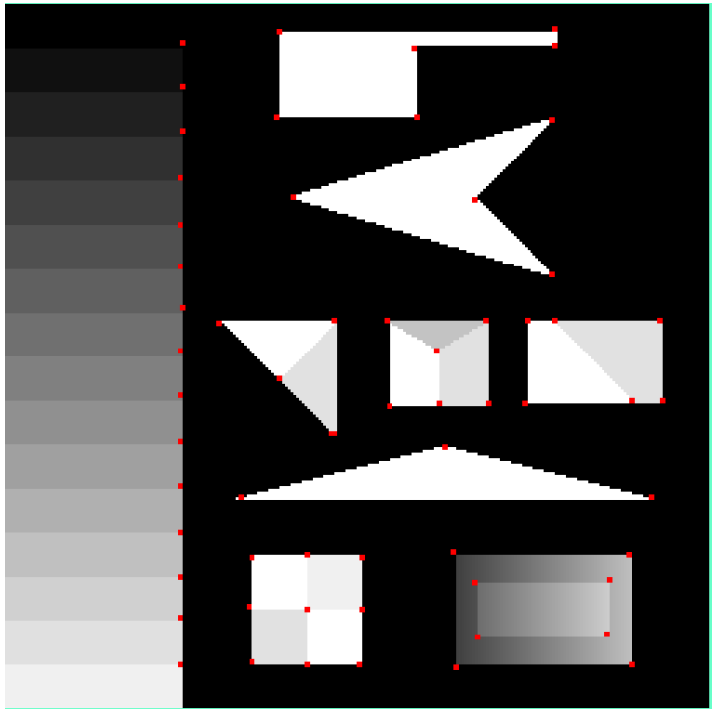


Local features: Main components

1. Detection: Identify the **interest points.**
2. Description: Extract vector **feature descriptor** surrounding each interest point.
3. Matching: Determine **correspondence** between descriptors in two views.



Detection: Corners

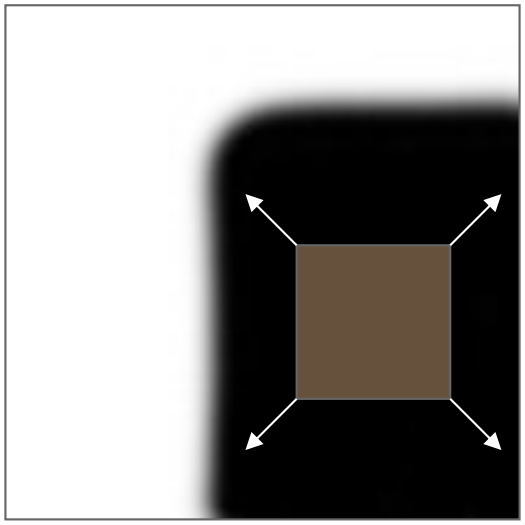


- Key property: in the region around a corner, image **gradient** has **two or more dominant directions**.
- Corners are **repeatable** and **distinctive.**

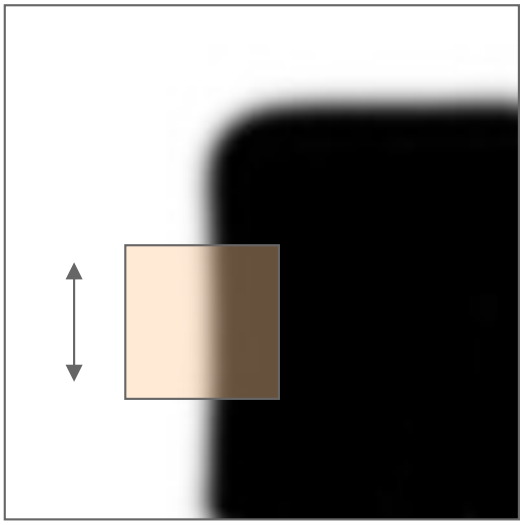
C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147--151.

Corner Detection: Basic Idea

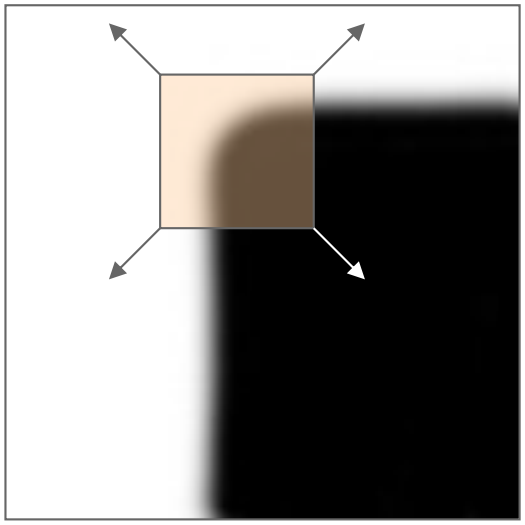
- Recognize the point by looking through a window.
- Shifting a window in ***any direction*** should give *a* ***large change*** **in intensity.**



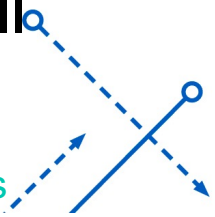
"flat" region:
no change in
all directions



"edge":
no change
along the edge
direction



"corner":
significant
change in all
directions



Source: A. Efros

Harris Detector

- Consider an image patch and shift it for $[\Delta x, \Delta y]$.
- *Sum of squared differences* (SSD) of two patches:

$$f(\Delta x, \Delta y) = \sum_{(x_k, y_k) \in W} (I(x_k, y_k) - I(x_k + \Delta x, y_k + \Delta y))^2$$

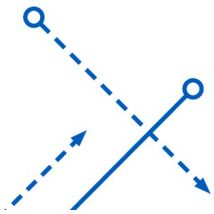
- $I(x + \Delta x, y + \Delta y)$ can be approximated by Taylor expansion. Let I_x, I_y be the partial derivative of I.

$$I(x + \Delta x, y + \Delta y) \approx I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y$$

- This produces $f(\Delta x, \Delta y) \approx \sum_{(x,y) \in W} (I_x(x, y)\Delta x + I_y(x, y)\Delta y)^2$

$$f(\Delta x, \Delta y) \approx (\Delta x \quad \Delta y) M \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y]$$



Harris Detector

- The SSD becomes:

$$f(\Delta x, \Delta y) \approx \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} M \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

where *M* is a 2×2 matrix with image derivatives:

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix}$$

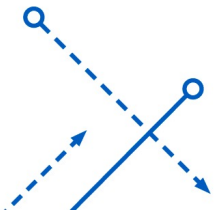
Structure Tensor

Sum over the patch

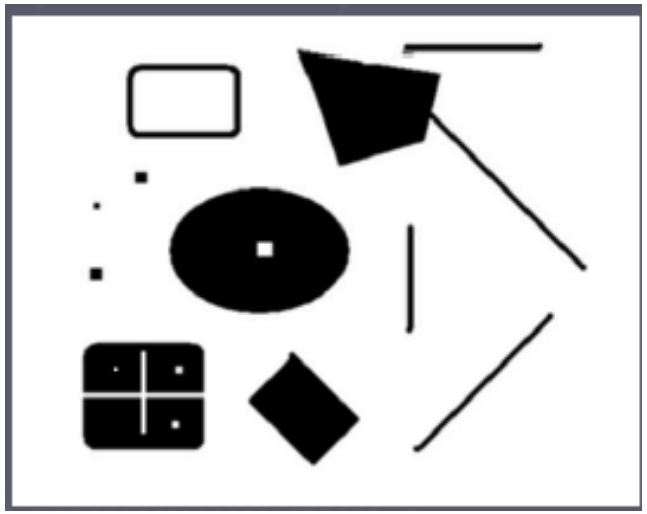
Gradient with respect to x,
times gradient with respect to y

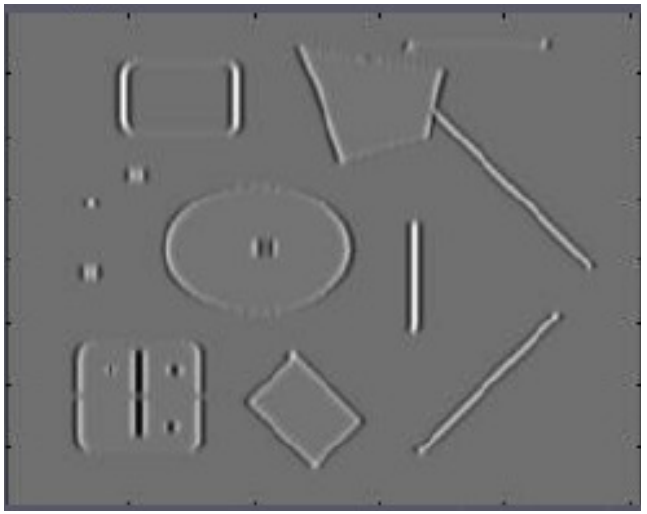
- We sometimes add window weights to the SSD:

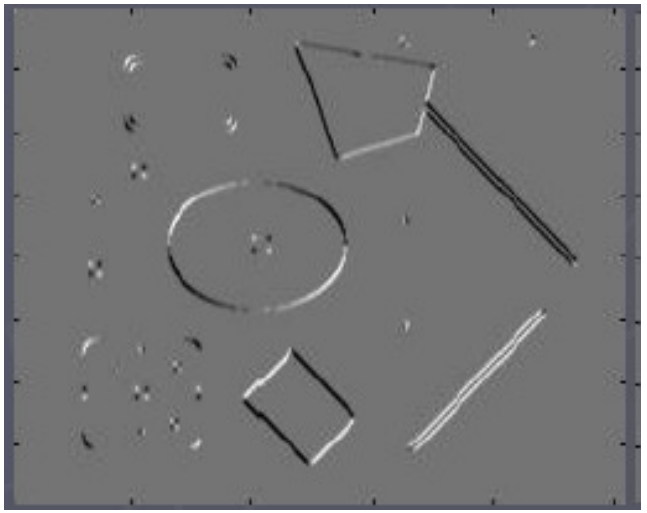
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Harris Detector

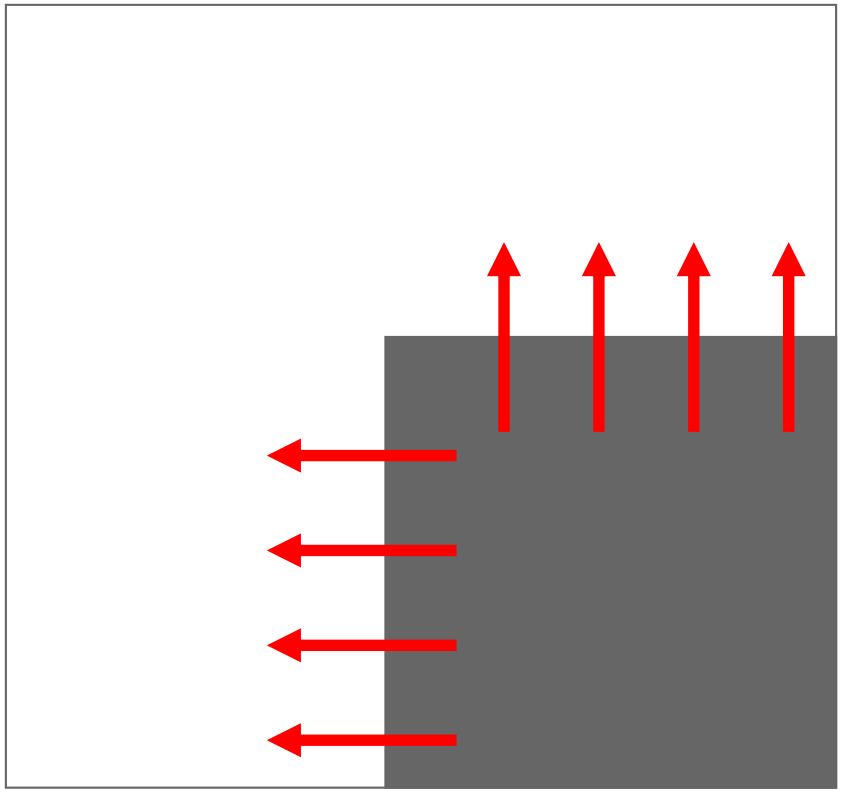






What does this matrix reveal?

- First, consider an axis-aligned corner:

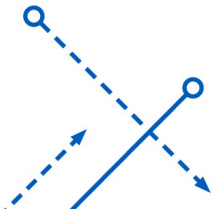


What does this matrix reveal?

- First, consider an axis-aligned corner:

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

- Dominant gradient directions align with x or y axis
- If either λ is close to 0, then this is not a corner, so look for locations where both are large.
- What if we have a corner that is not aligned with the image axes?

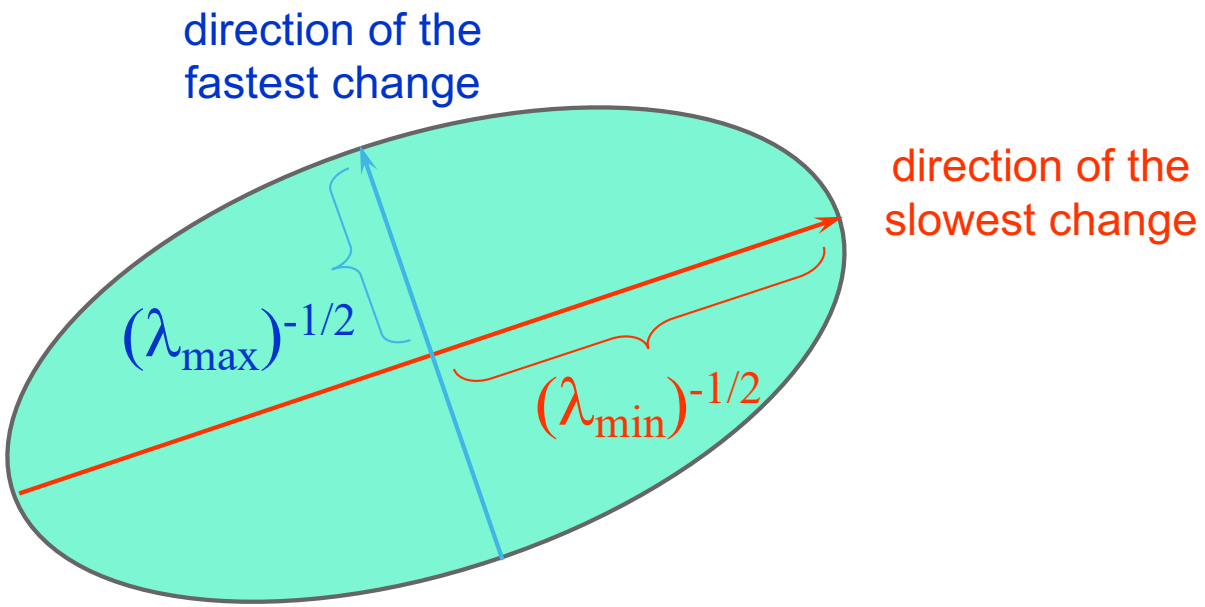


General Case

Since M is symmetric, we have

$$M = \mathbf{Q}^{-1}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}\mathbf{Q}$$

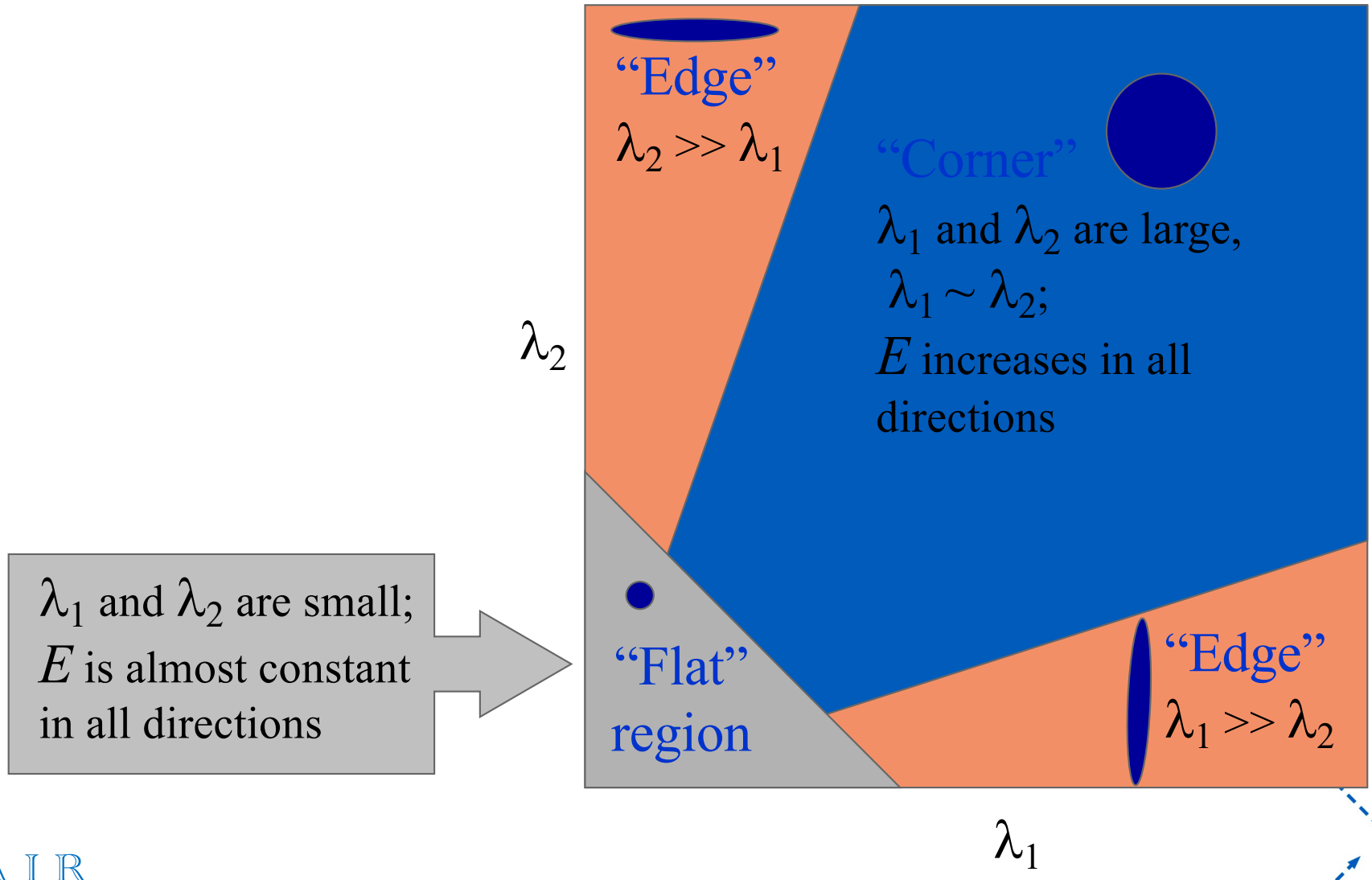
We can visualize *M* as an ellipse with axis lengths determined by the eigenvalues and orientation determined by *Q matrix.*



Slide adapted form Darya Frolova, Denis Simakov.

Interpreting the eigenvalues

- Classification of image points with eigenvalues of *M*:



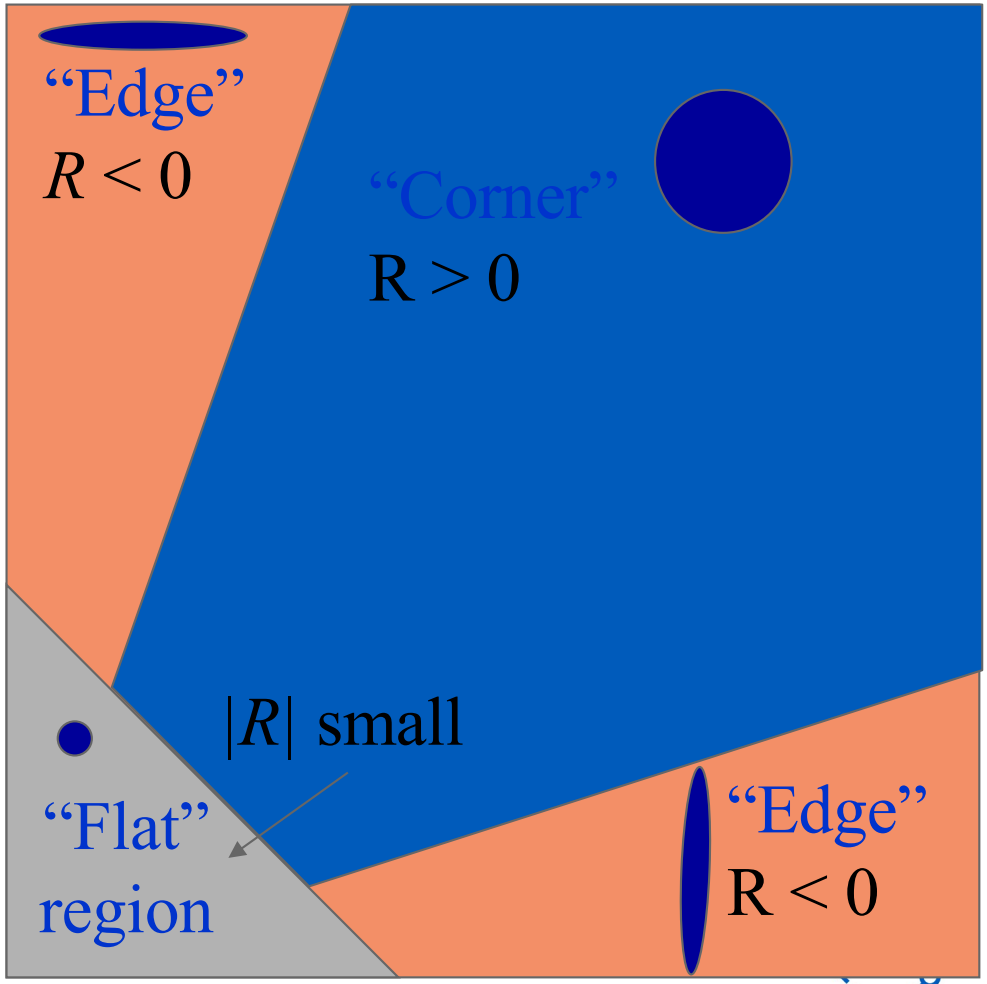
Harris Response Function

- The smallest eigenvalue of M:

$$R = \det(M) - \alpha \,\mathrm{trace}(M)^2$$
$$= \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

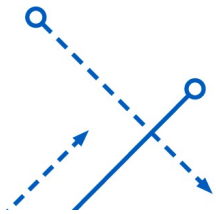
- where α is an empirically determined constant.

$$\alpha \in [0.04\ to\ 0.06]$$



Harris corner detector: Algorithm

1. Color to grayscale
2. Calculate spatial derivative
3. Compute structure tensor for all windows
4. Harris response calculation
   - Find points with large corner response: (R > threshold)
5. Non-maximum suppression
   - Take the points of local maxima of *R*

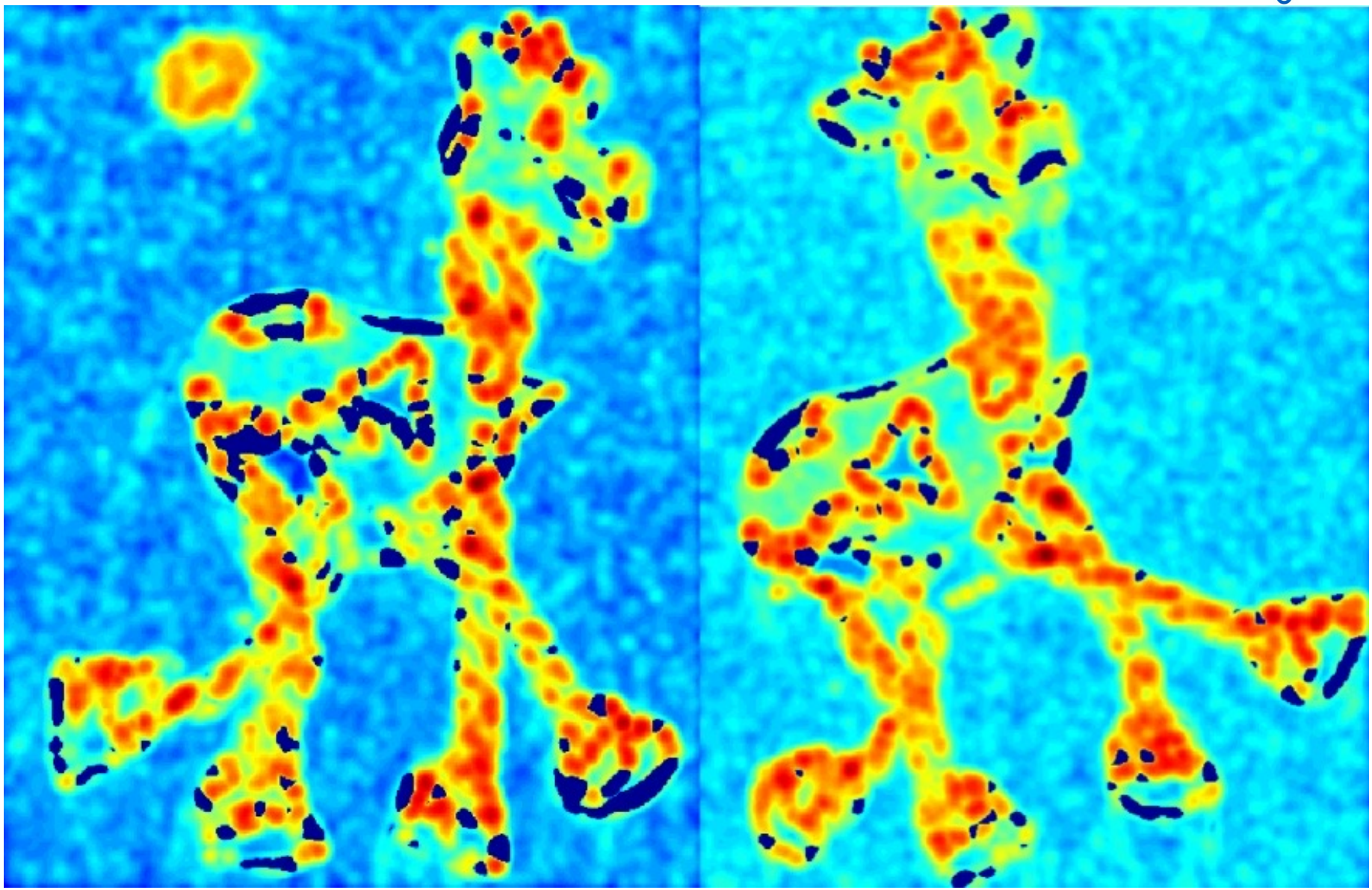


C.Harris and M.Stephens. "A Combined Corner and Edge Detector."
Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.

Harris Corner Detector

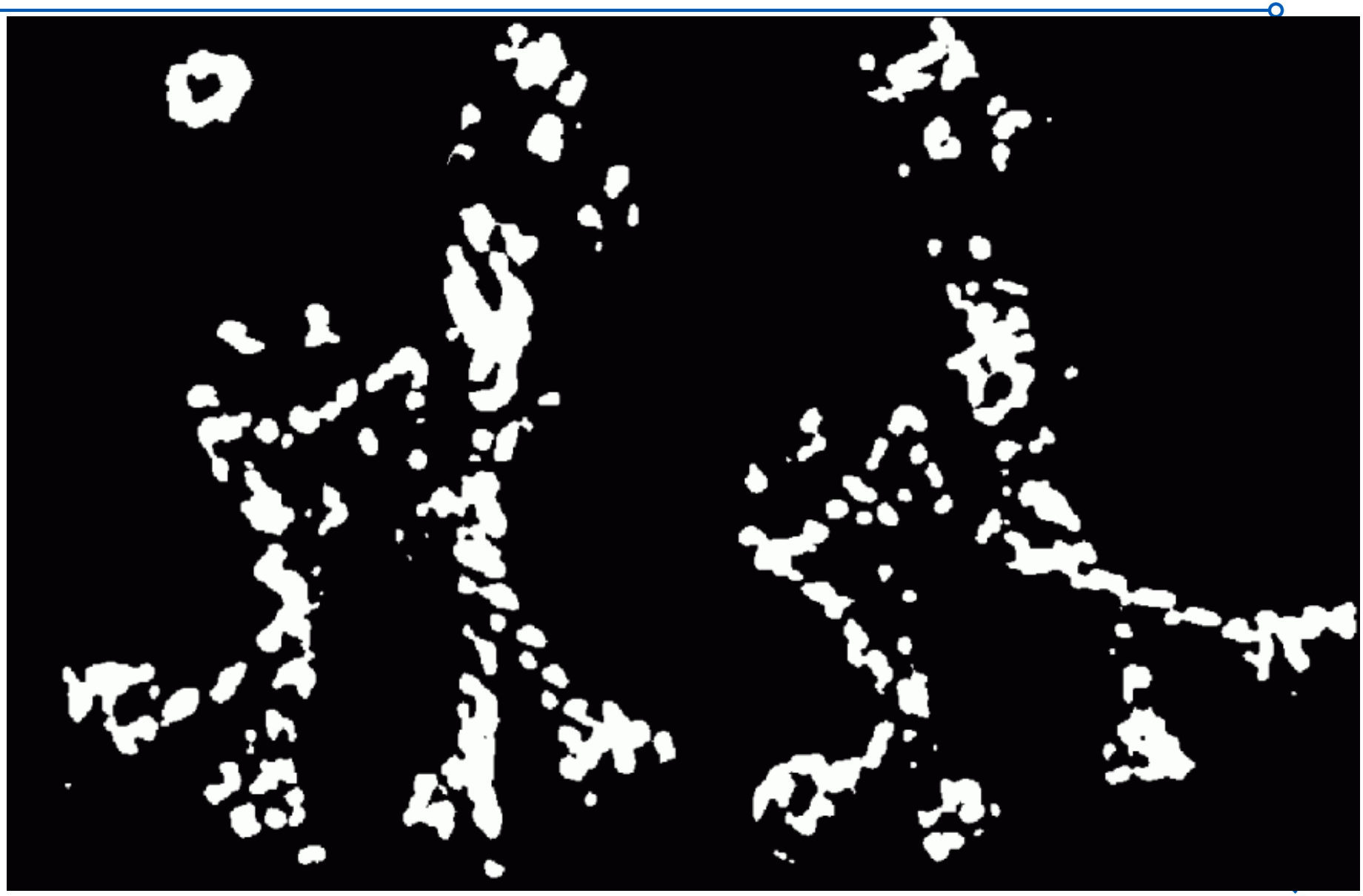
Slide adapted form Darya Frolova, Denis Simakov, Weizmann Institute.

Harris Corner detector



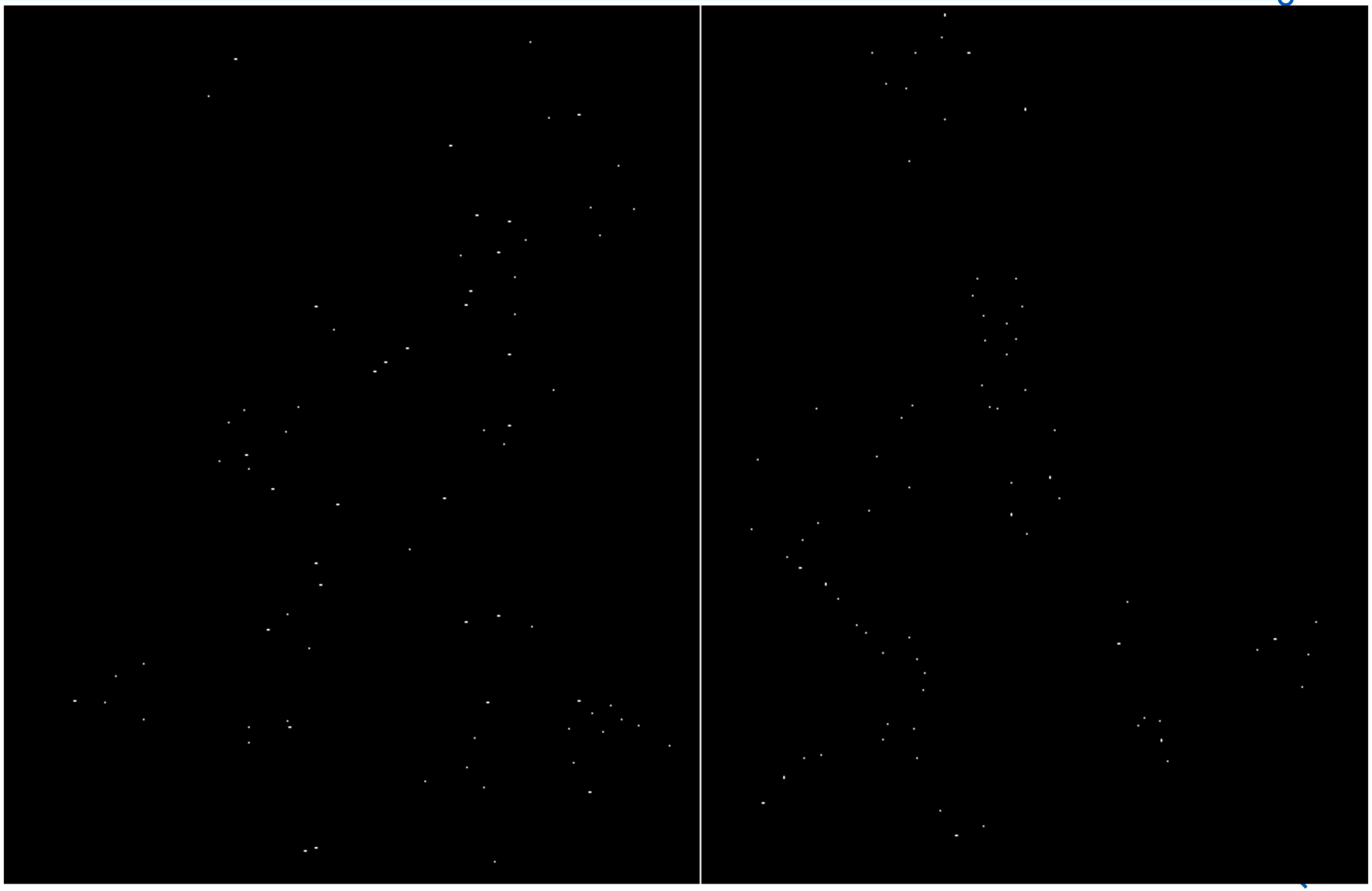
Corner response R

Harris Corner detector



Find points with large corner response: R>threshold

Harris Detector: Workflow



Take only the points of local maxima of R

Harris Corner

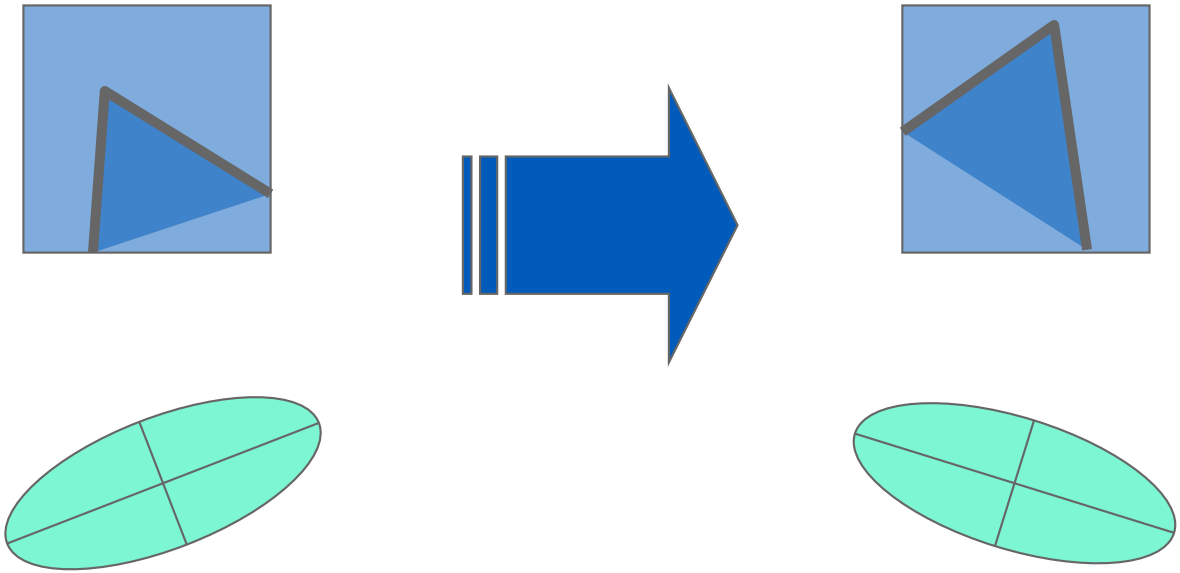
Image with Harris Corner Overlay.

Harris Corner

Original Image

Harris Detector: Properties

- Rotation invariance

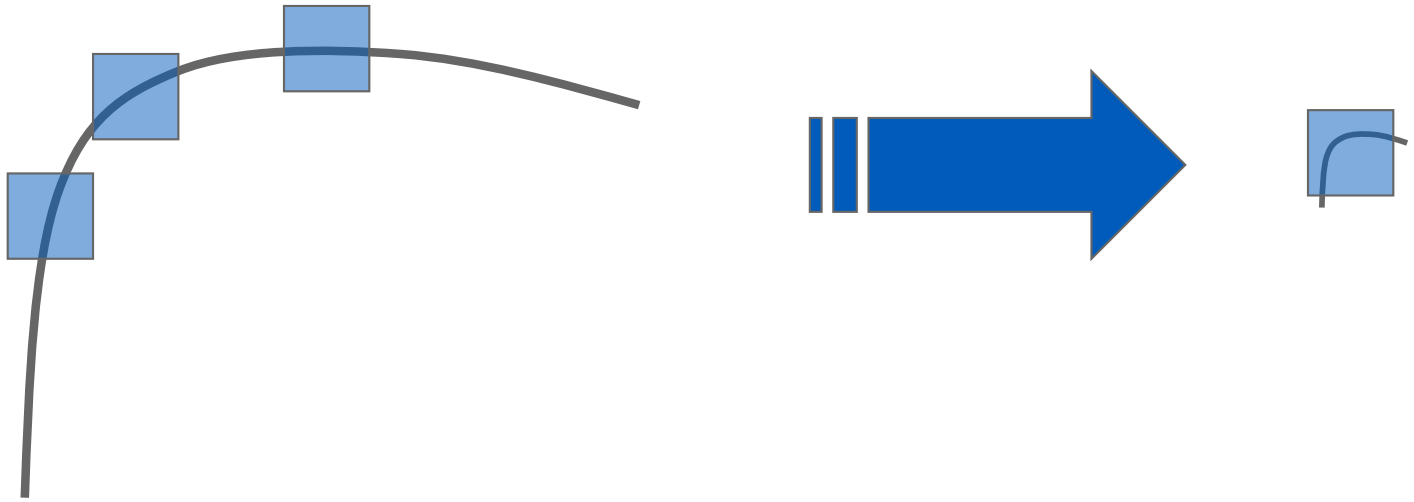


Ellipse rotates but its shape (eigenvalues) remains the same.

Corner response R is invariant to image rotation

Harris Detector: Properties

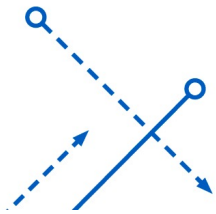
- Not invariant to image scale



All points will be classified as edges

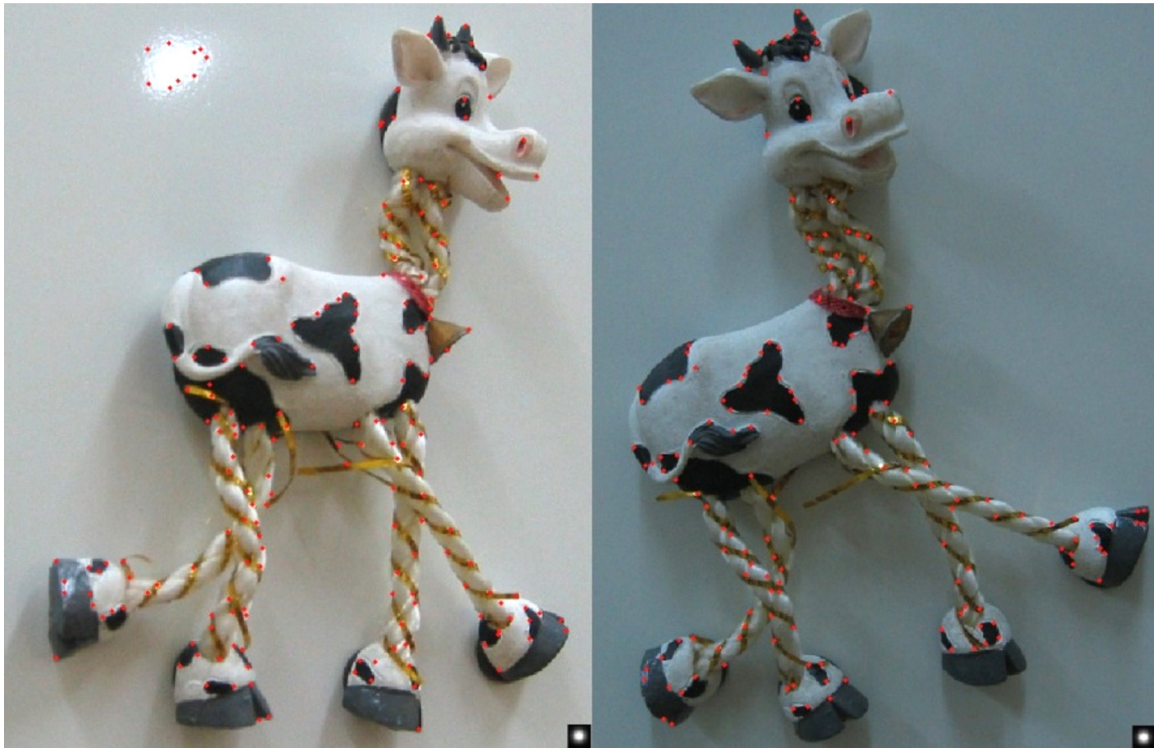
Corner !

- How can we detect **scale invariant** interest points?
  - **Image Pyramids!**



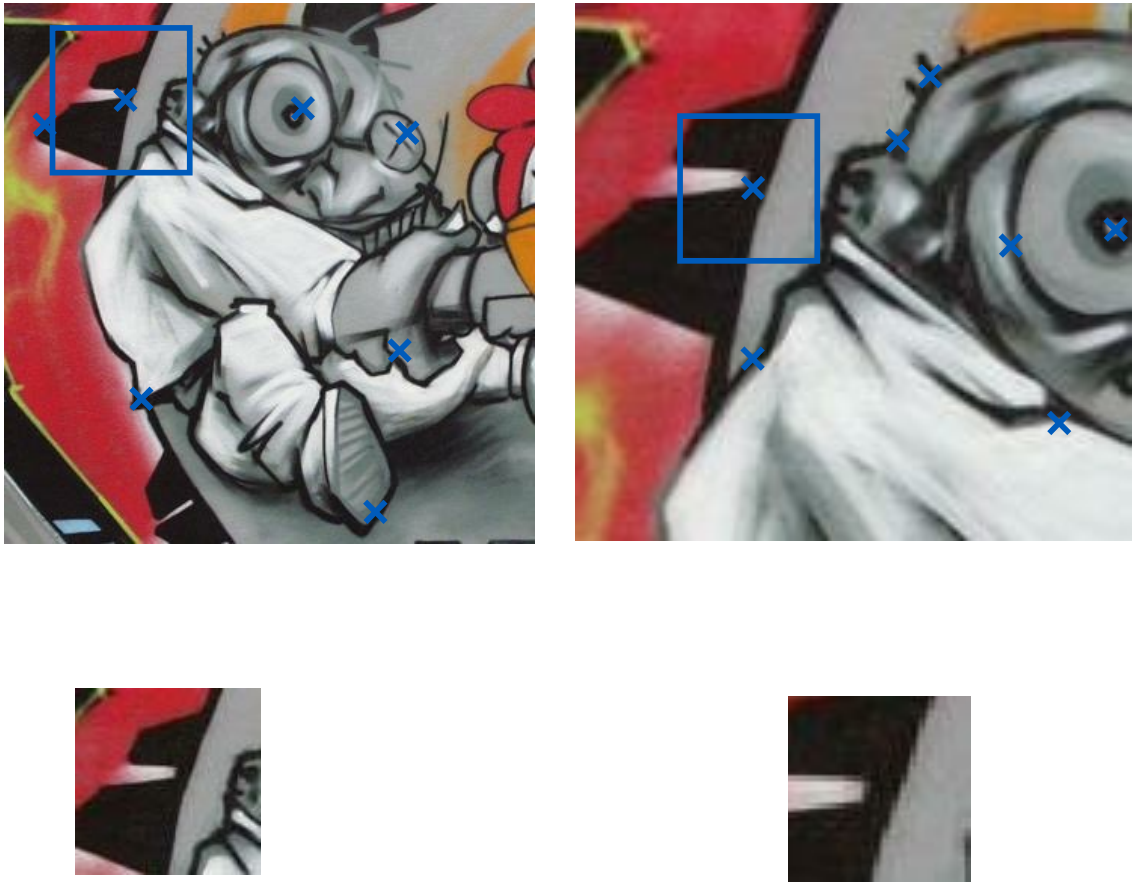
Invariance and Equivariance

- We want corner locations to be invariant to photometric transformations and equivariant to geometric transformations
  - **Invariance:** same corners are detected with photometric transformations, such as histogram equalization.
  - **Equivariance** (Covariance): Corners are detected in corresponding locations in geometrically transformed image.



Scale Invariance

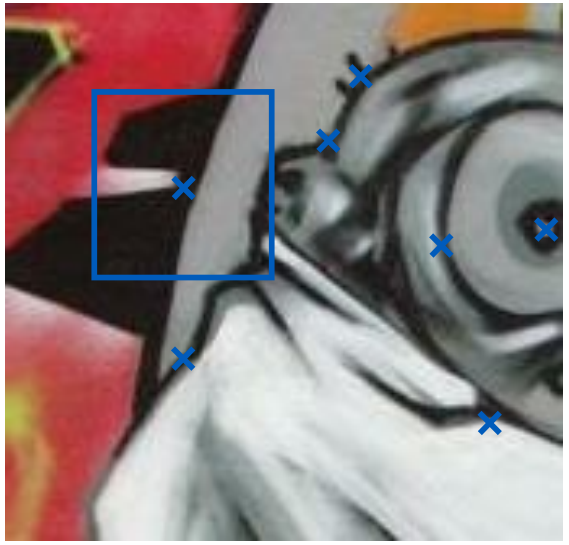
- Multi-scale approach (Image Pyramids)



Slide from T. Tuytelaars ECCV 2006 tutorial

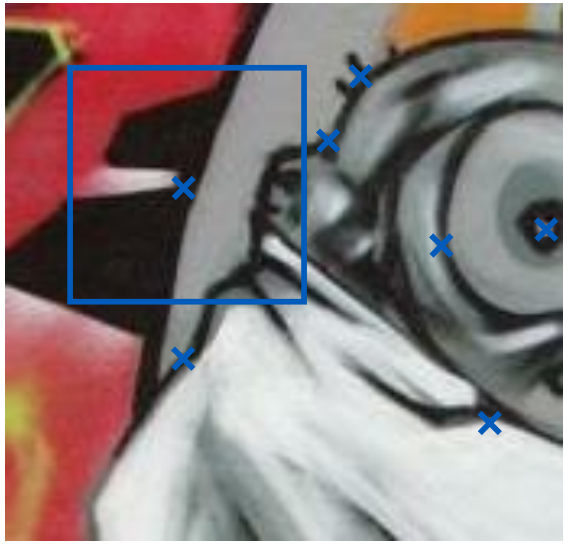
Scale Invariance

- Multi-scale approach (Image Pyramids)



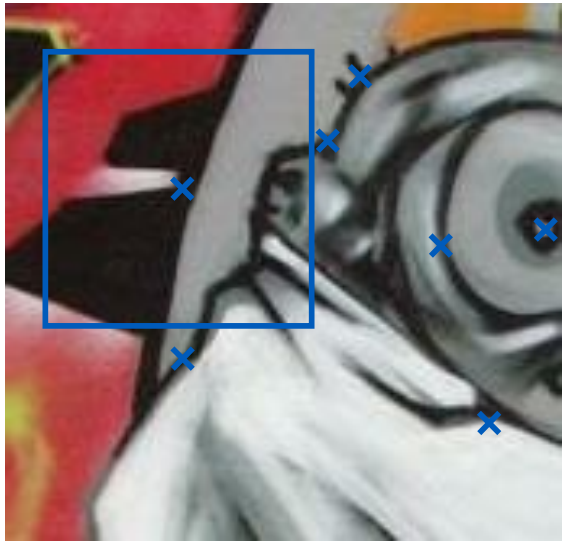
Scale Invariance

- Multi-scale approach (Image Pyramids)



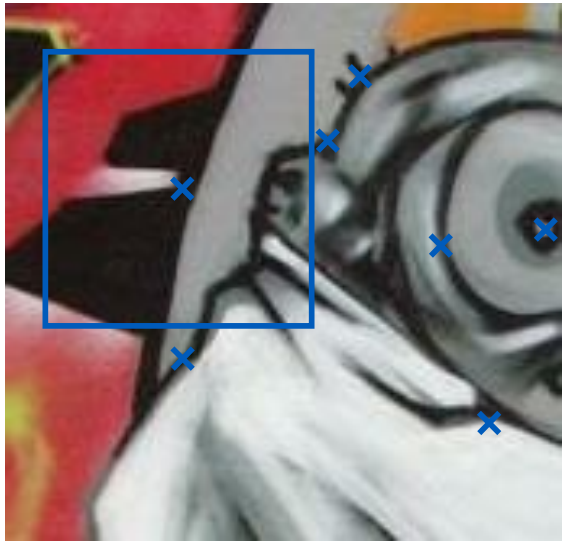
Scale Invariance

- Multi-scale approach (Image Pyramids)



Scale Invariance

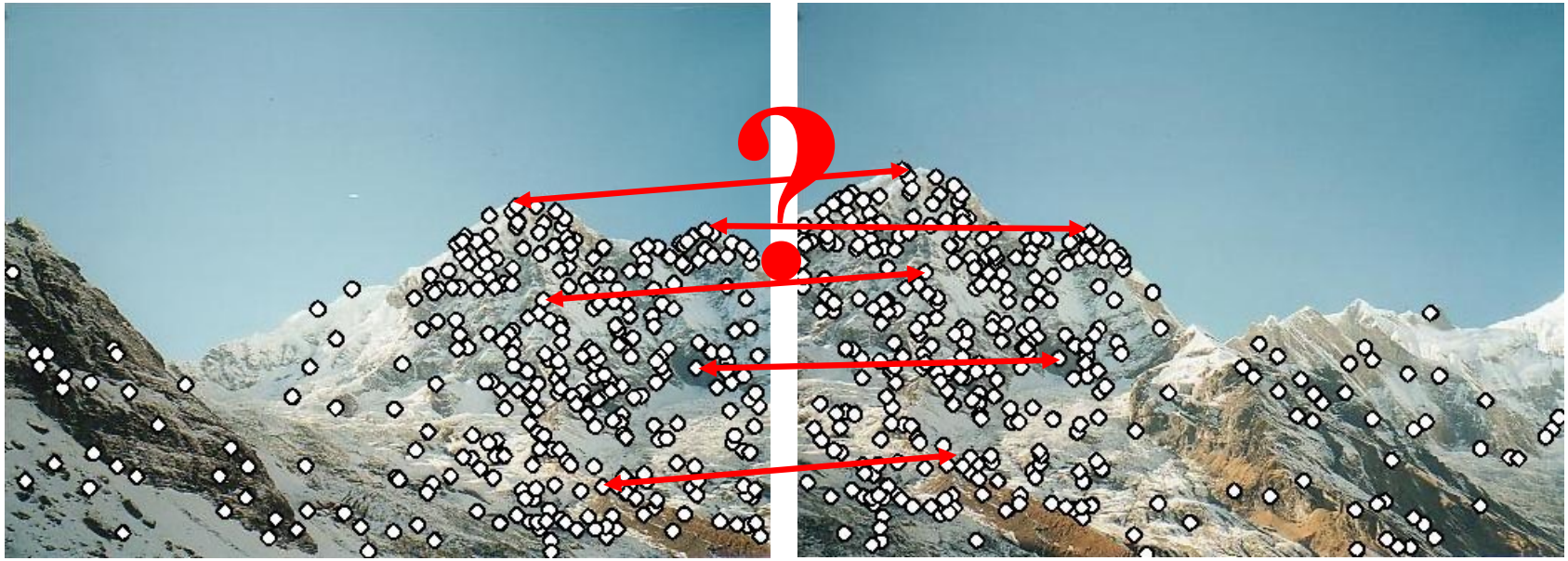
- Extract patch from each image individually



Local descriptors

- We know how to detect points
- Next question:

How to *describe* them for matching?



Point descriptor should be:

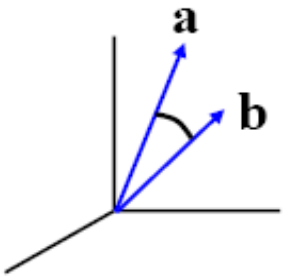
1. Invariant
2. Distinctive

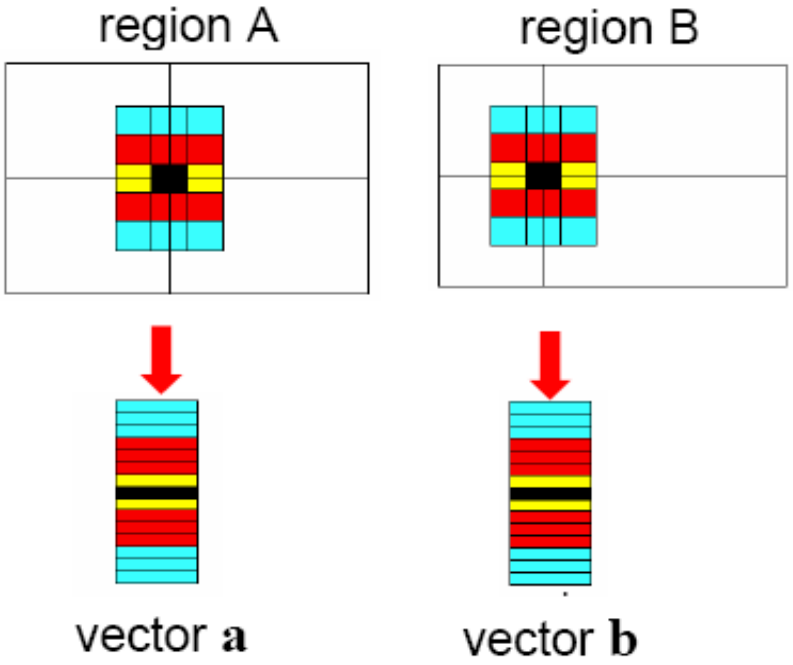
Local descriptors

- Simplest: list of intensities within a patch.
- What is this going to be invariant to?

Write regions as vectors

$$\mathtt{A} \rightarrow \mathbf{a}, \quad \mathtt{B} \rightarrow \mathbf{b}$$

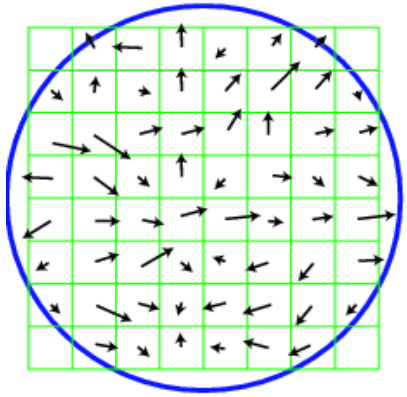


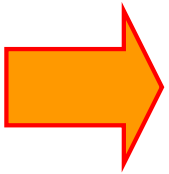


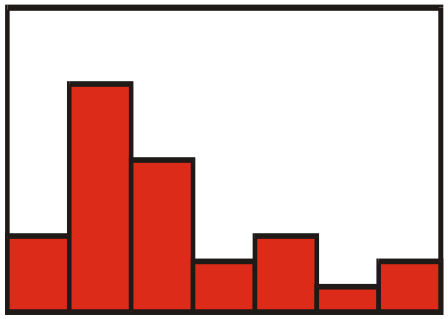
Feature descriptors

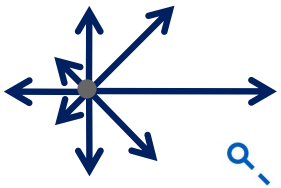
- Disadvantage of patches as descriptors:
  - Small shifts can affect matching score a lot

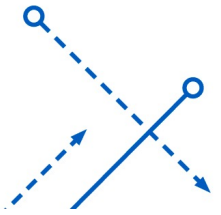
- Solution: histogram or oriented gradients.





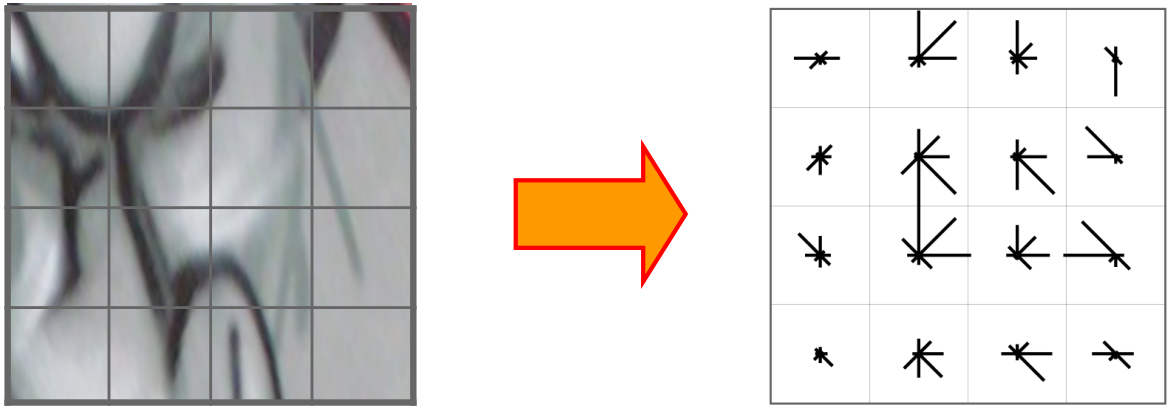






Feature descriptors: SIFT

- **S**cale **I**nvariant **F**eature **T**ransform
  - Divide patch into 4x4 sub-patches: 16 cells
  - Compute histogram of gradient orientations (8 reference angles) for all pixels in each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions



David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Rotation Invariant Descriptors

Dominant direction of gradient for the image patch

- Find local orientation

- Rotate patch according to this angle
  - This puts the patches into a canonical orientation.

Rotation Invariant Descriptors

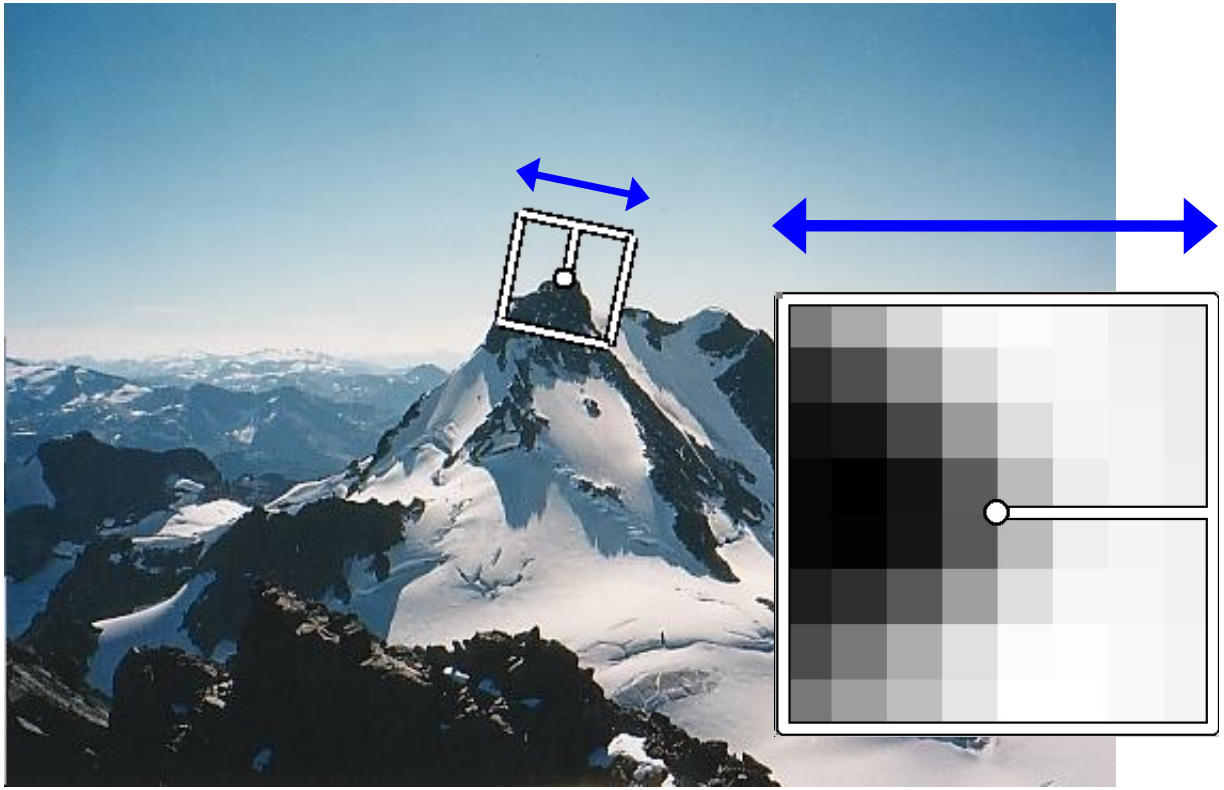


Image from Matthew Brown

Feature descriptors: SIFT

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint
    - Up to about 60 degree out of plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night
  - Fast and efficient—can run in real time (not for robots).

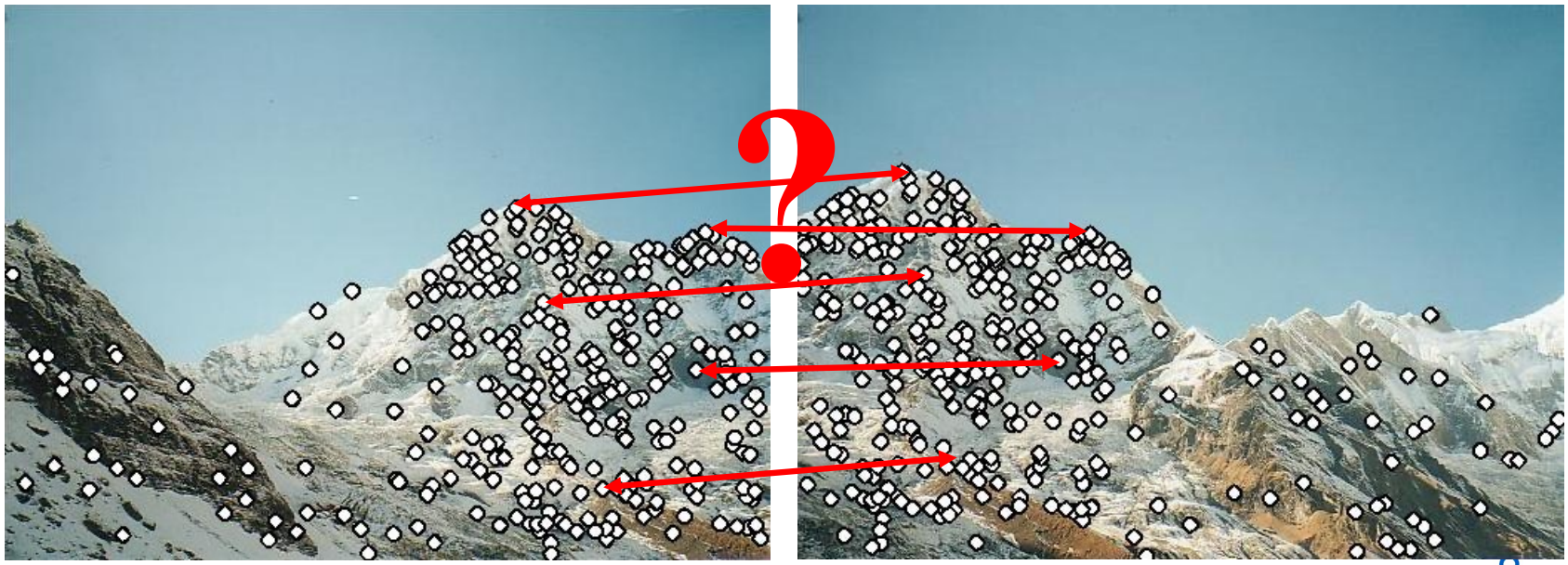
Working with SIFT descriptors

- One image yields:
  - n 128-dimensional descriptors:
    - each one is a histogram of the gradient orientations within a patch
    - [n x 128 matrix]
  - n scale parameters specifying the size of each patch
    - [n x 1 vector]
  - n orientation parameters specifying the angle of the patch
    - [n x 1 vector]
  - n 2d points giving positions of the patches
    - [n x 2 matrix]

Feature matching

We know how to detect **and describe** good points

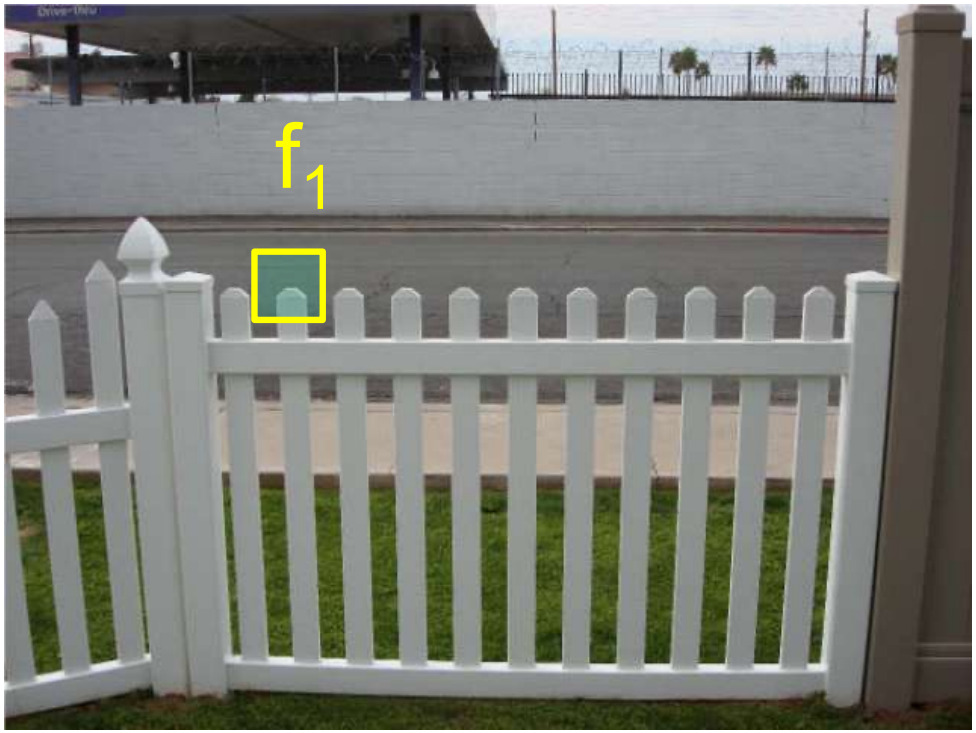
Next question: **How to match them?**

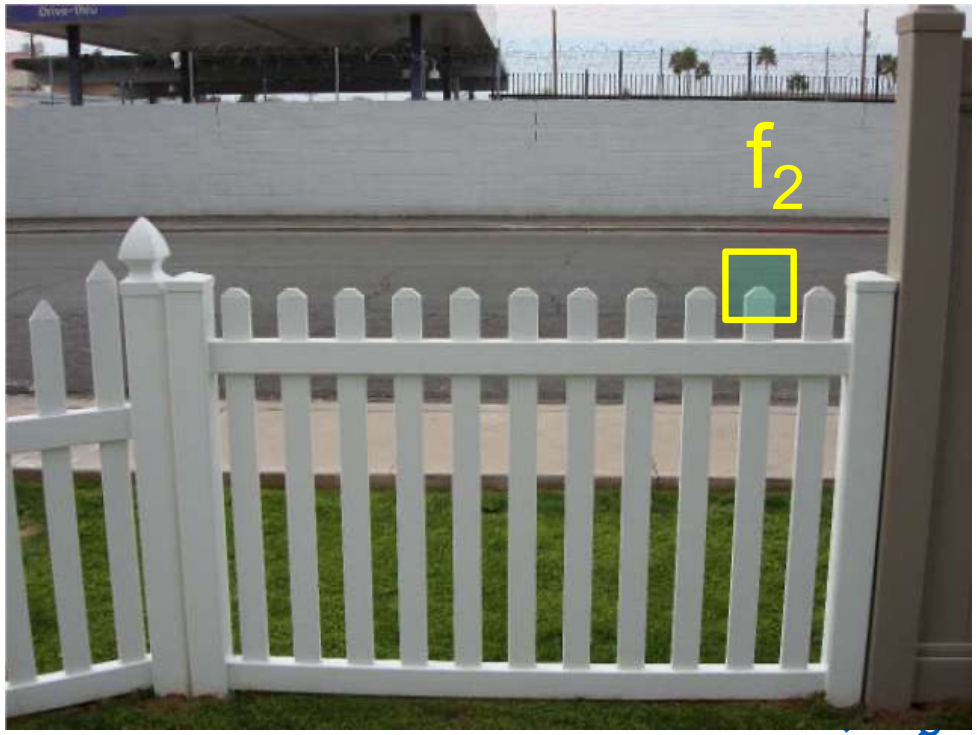


Feature distance

How to define difference of two features f_1, f_2?

- Simple approach is SSD(f_1, f_2)
  - Sum of Square Differences between two descriptors
  - can give good scores to ambiguous (bad) matches

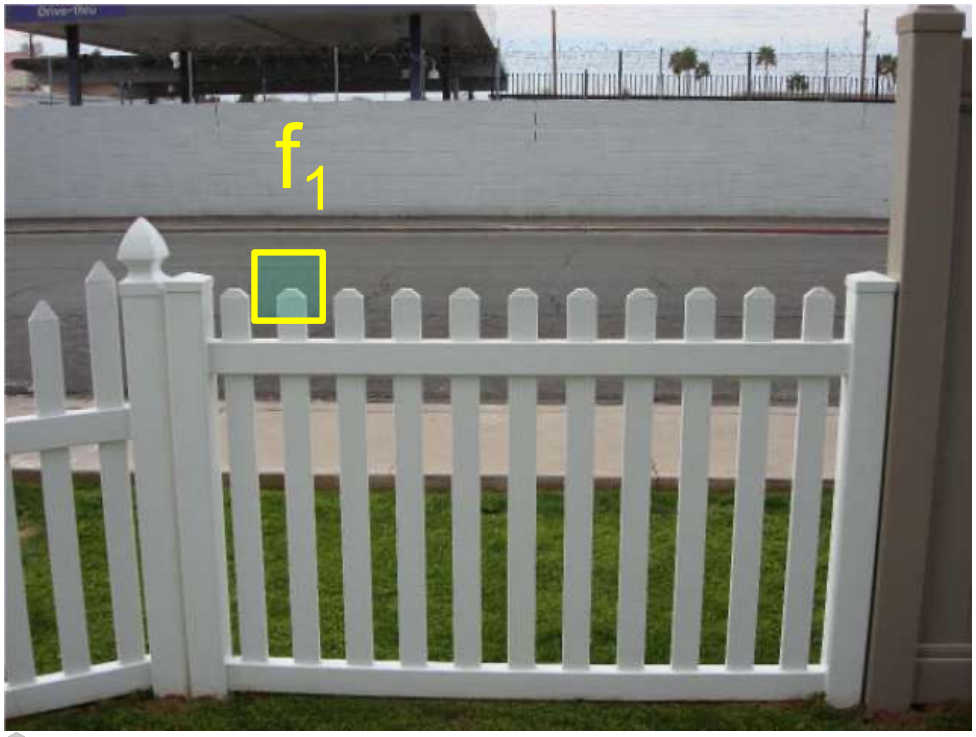


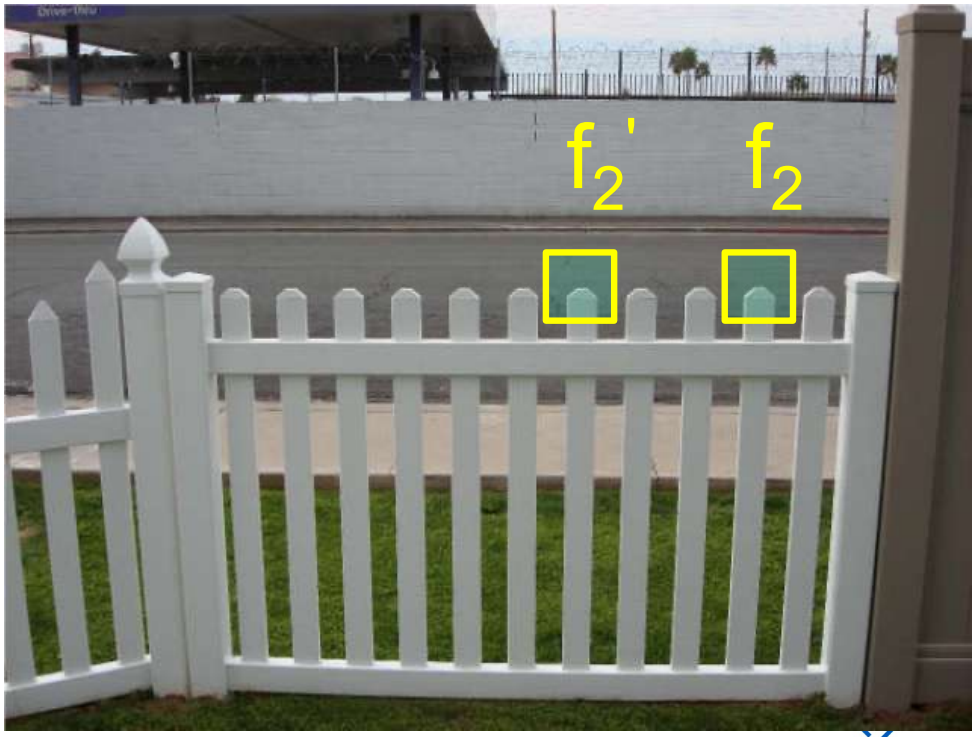


Feature distance

How to define difference between two features f_1, f_2?

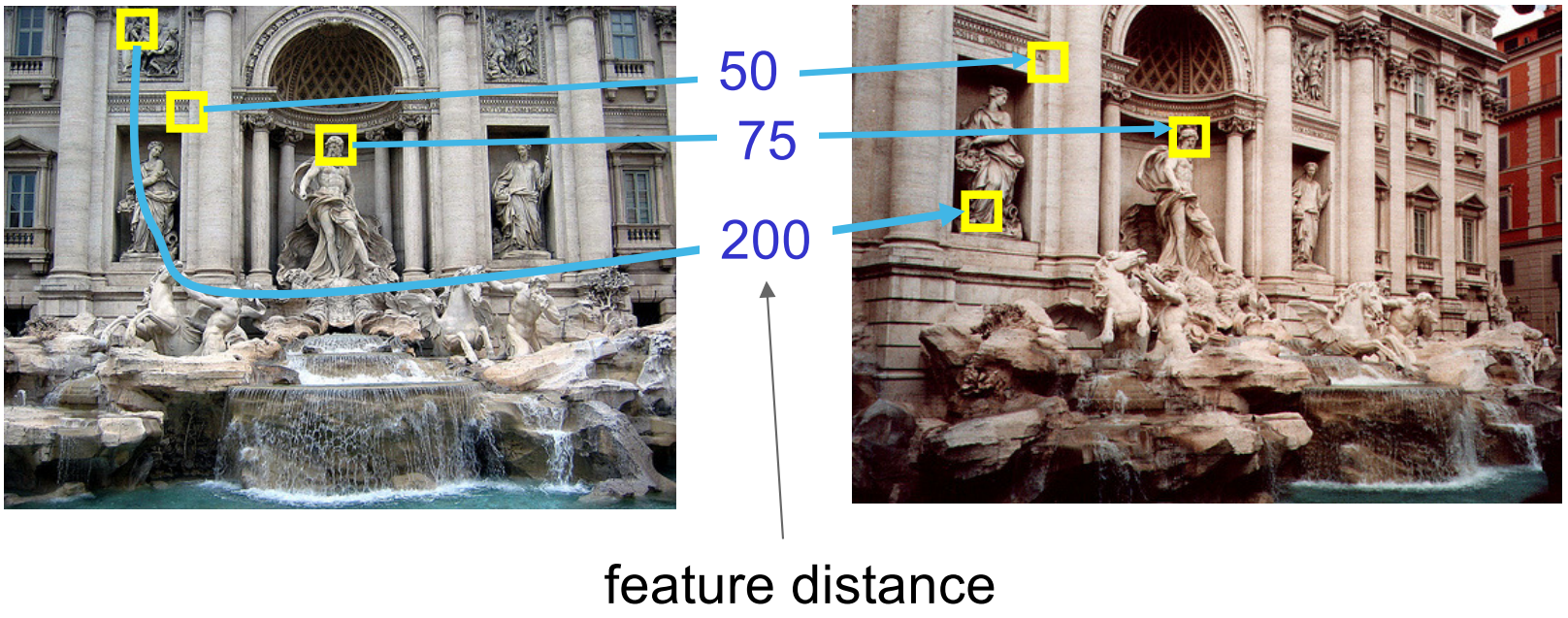
- Better approach:
  - ratio distance = SSD(f_1, f_2) / SSD(f_1, f_2').
  - f_2 is best SSD match to f_1 in I_2 , while f_2' is 2nd best.
  - gives small values for ambiguous matches.





Evaluation of matches

How can we measure the performance of a feature matcher?

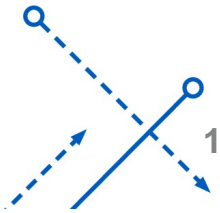


Feature Matching Summary

Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min difference

- Simple approach is SSD(f_1, f_2)
  - sum of square differences between entries of the two descriptors
  - can give good scores to very ambiguous (bad) matches
- Better approach: ratio distance = SSD(f_1, f_2) / SSD(f_1, f_2')
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2nd best SSD match to f_1 in I_2
  - gives small values for ambiguous matches



Other Distance Measures

- Sum of Squared differences (SSD)

$$distance(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Correlation

$$\rho_{X,Y} = \operatorname{corr}(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\operatorname{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

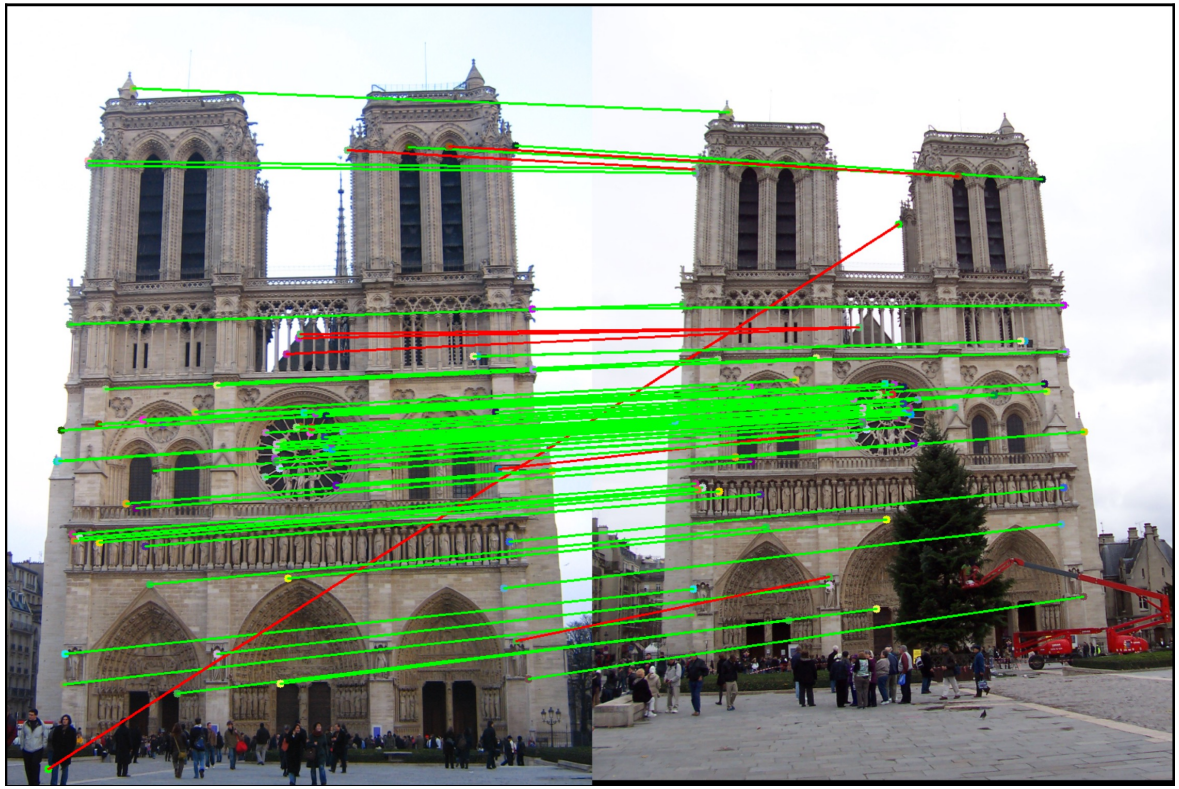
- Cosine Similarity

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

True/false positives

The distance threshold affects performance

- True positives = # of detected correct matches.
- False positives = # of detected incorrect matches.

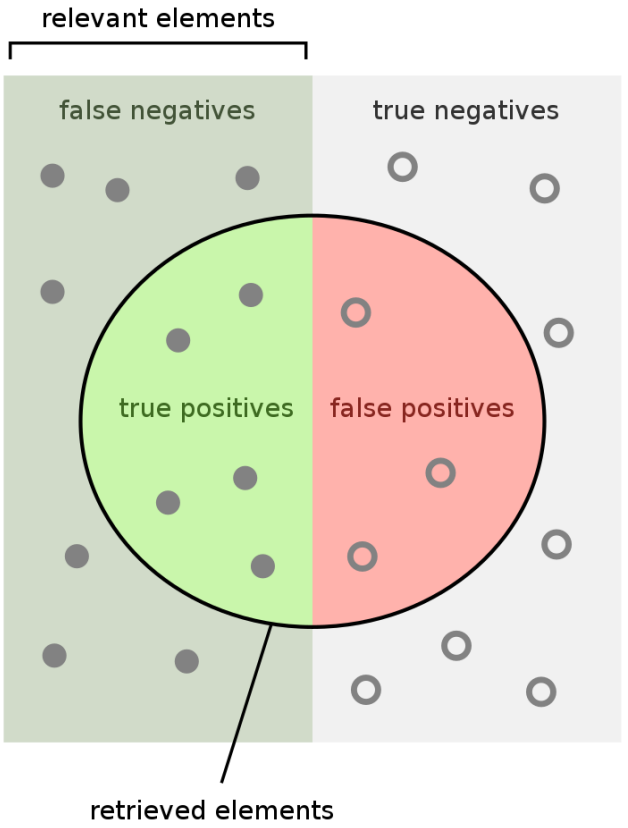


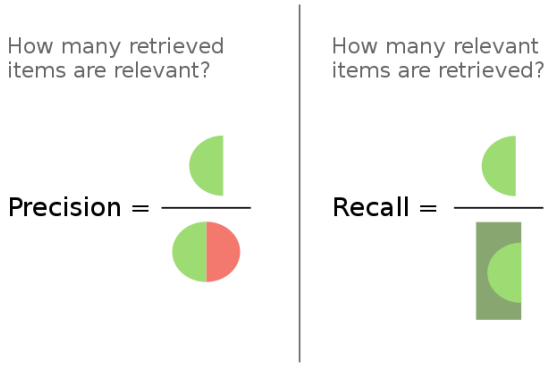
Precision, Recall, F1

$$\text{Precision} = \frac{\text{True Positive}}{\text{Predicted Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{Actual Positive}}$$

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

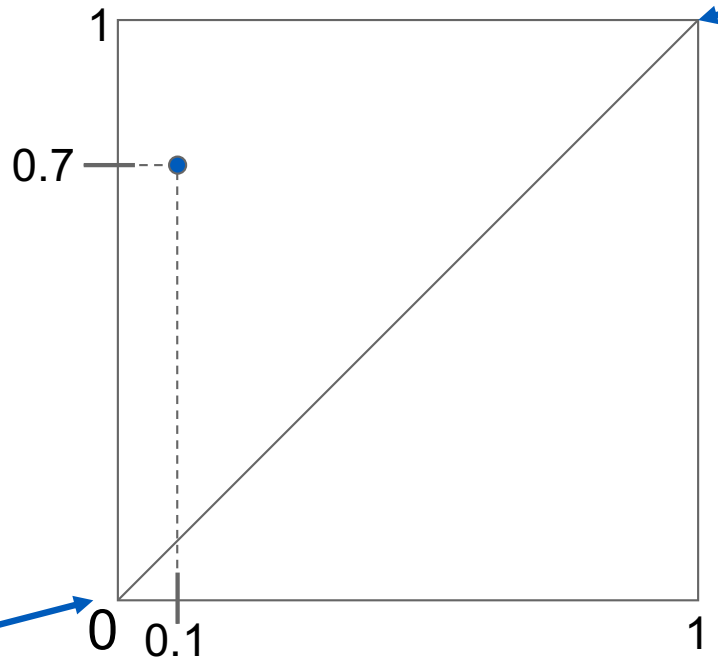


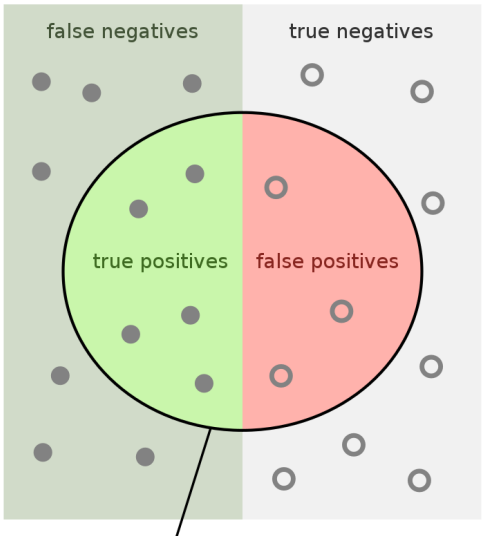


Evaluation of matches

- How can we measure the performance of a feature matcher?

Lowest threshold
all pairs as match





True Positive Rate

$$\frac{\#\,TP}{\#\,TP + FN}$$

Highest threshold
no match

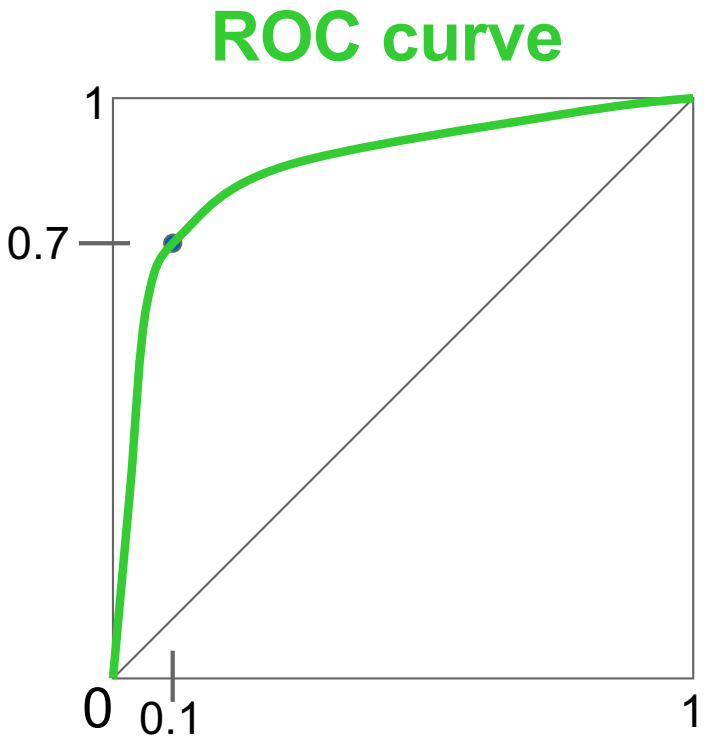
False Positive Rate

$$\frac{\#\,FP}{\#\,FP + TN}$$

Evaluation of matches

ROC Curves (Receiver Operator Characteristic)

- Generated by counting # correct/incorrect matches, for different thresholds.
- Want to maximize area under the curve (AUC)
- Useful for comparing different feature matching methods.



true positive rate

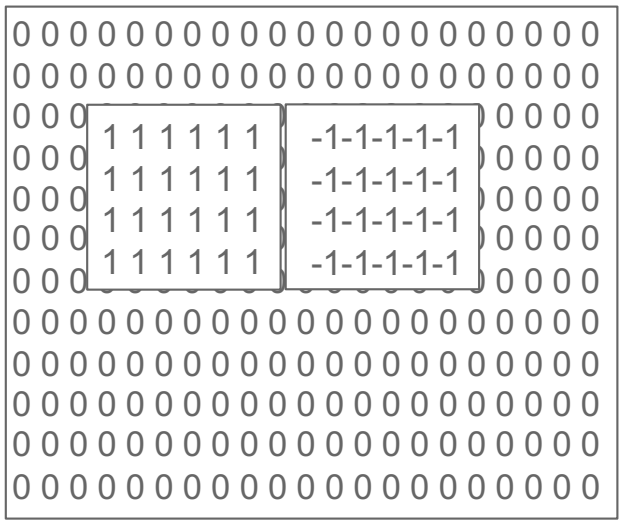
$$\frac{\#\,TP}{\#\,TP + FN}$$

false positive rate

$$\frac{\#\,FP}{\#\,FP + TN}$$

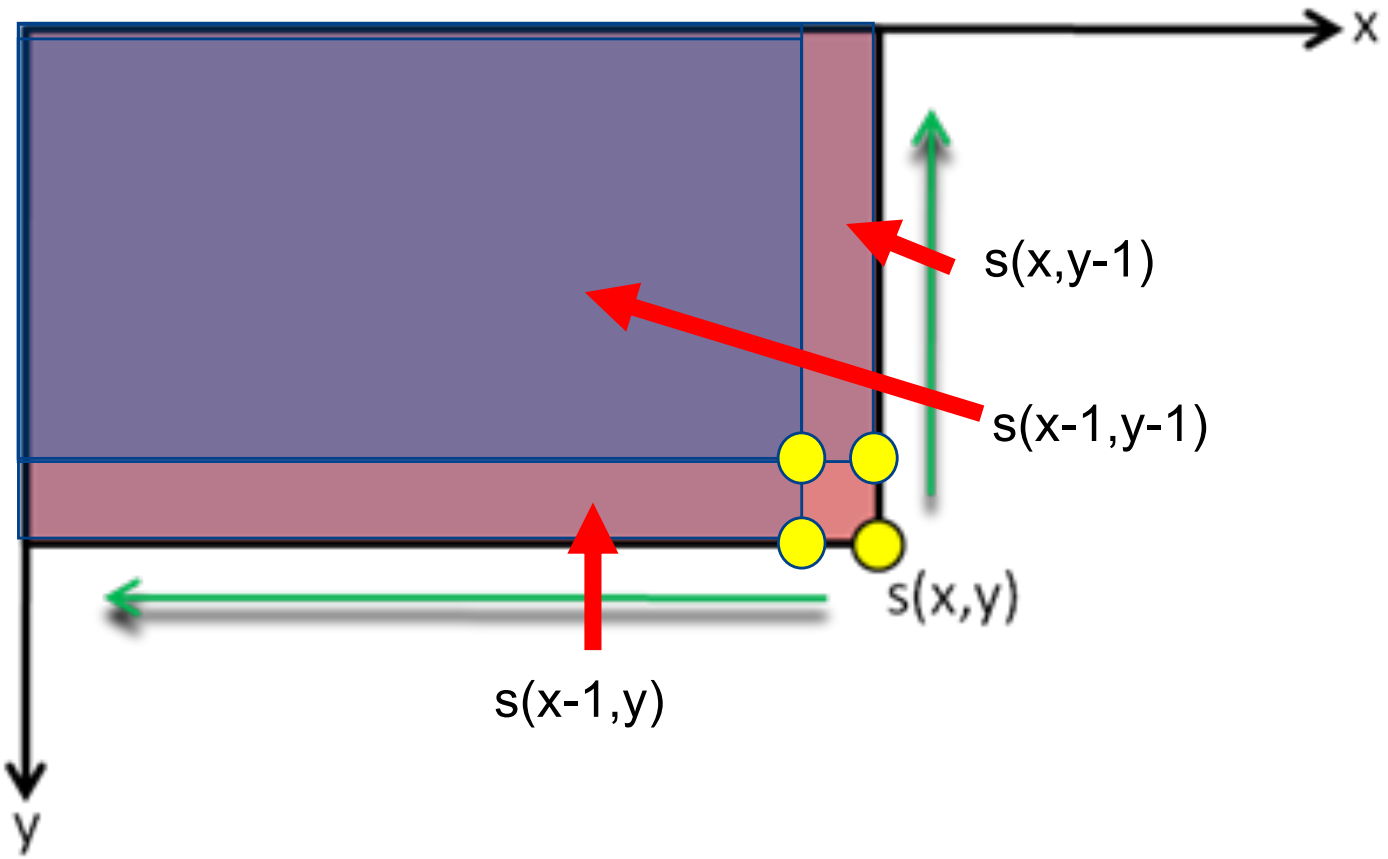
Integral Image: Motivation

- How can we make things faster?
- Many of the filters we have defined are made of rectangles or combinations of rectangles!



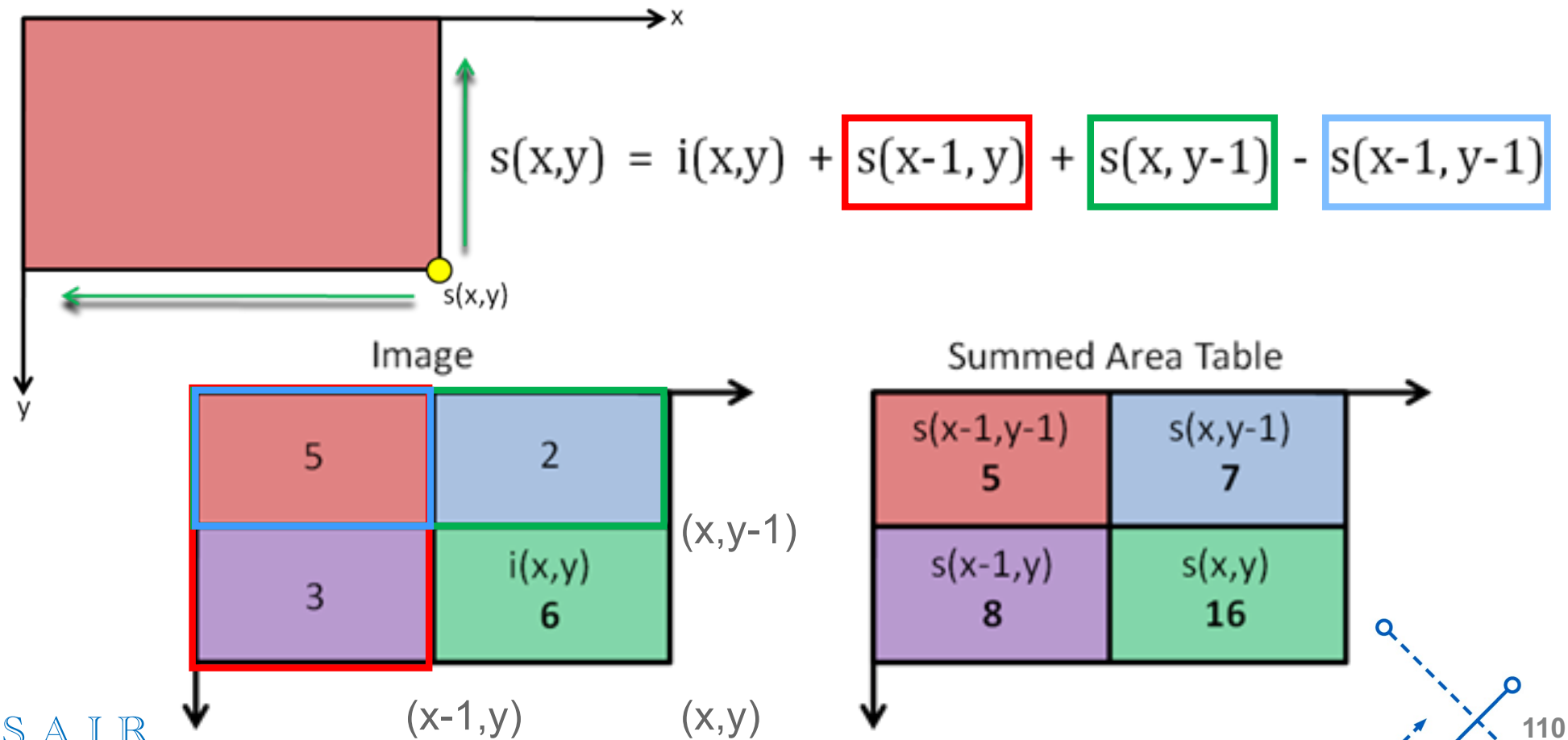
Integral Image

- A transformed image where every pixel is the sum of all pixels **above** and to the **left** of original image.

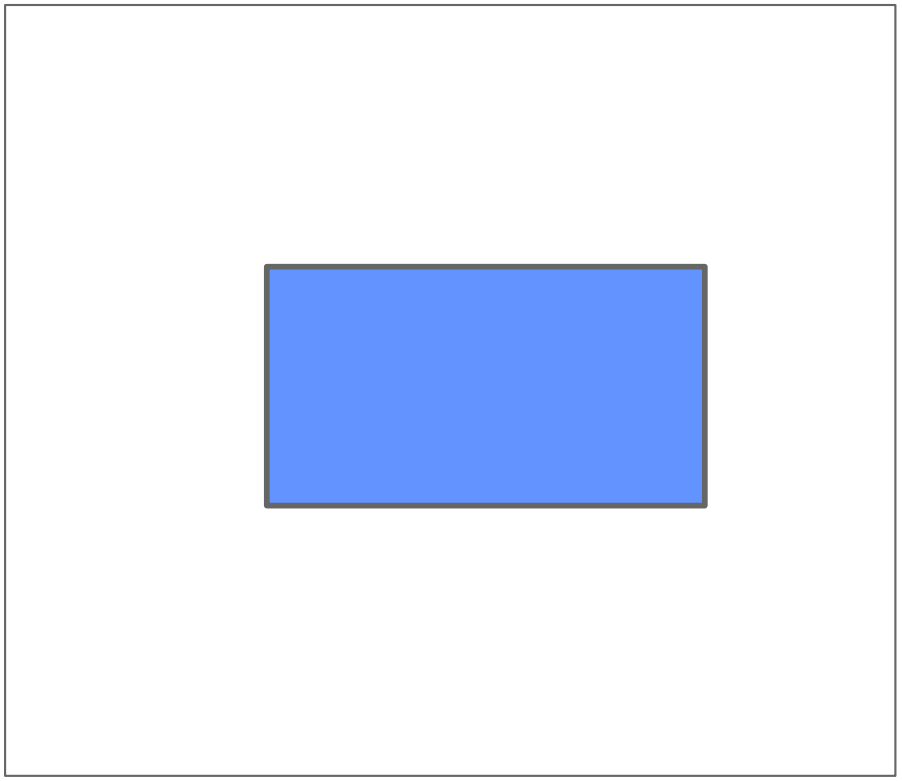


Integral image

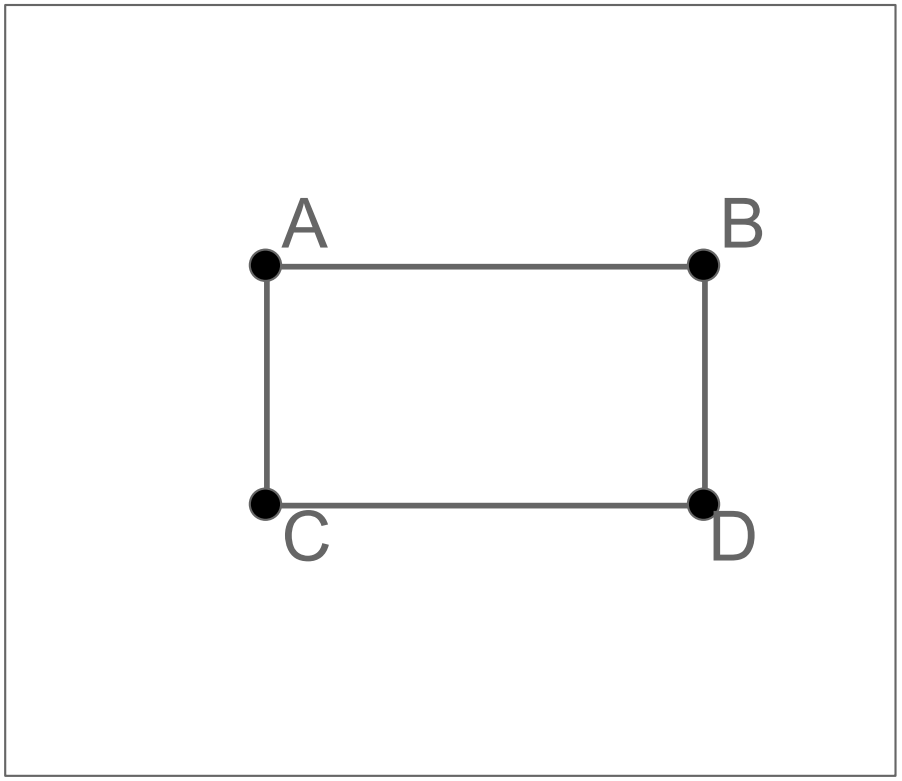
- a quick and effective way of calculating the sum of values (pixel values)



Integral images



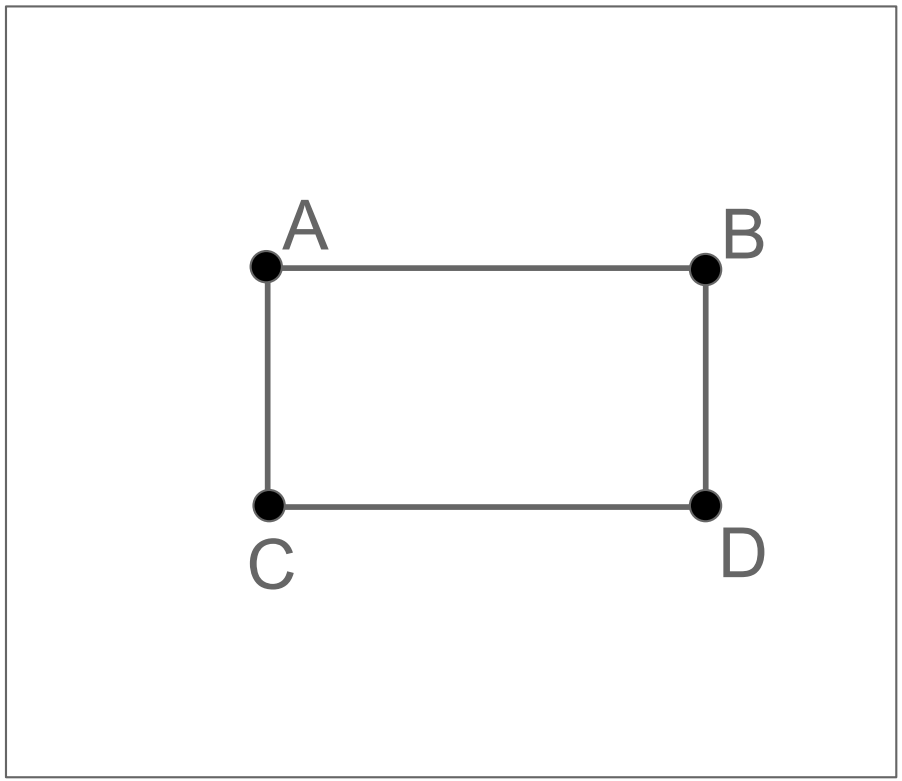
input image



integral image

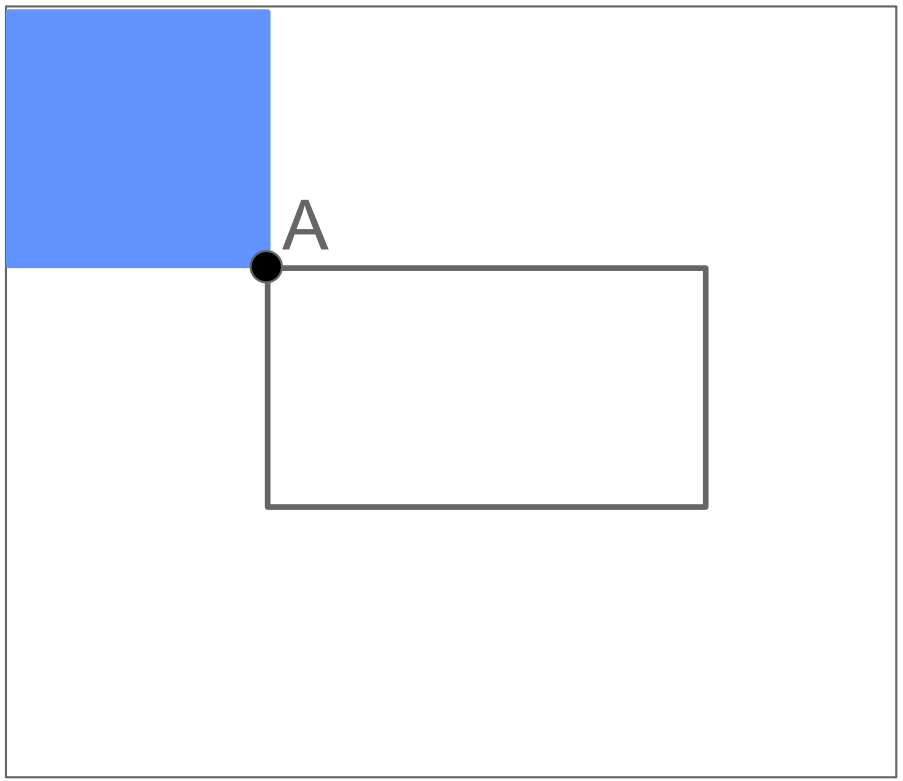
- What's the sum of pixels in the blue rectangle?

Integral images



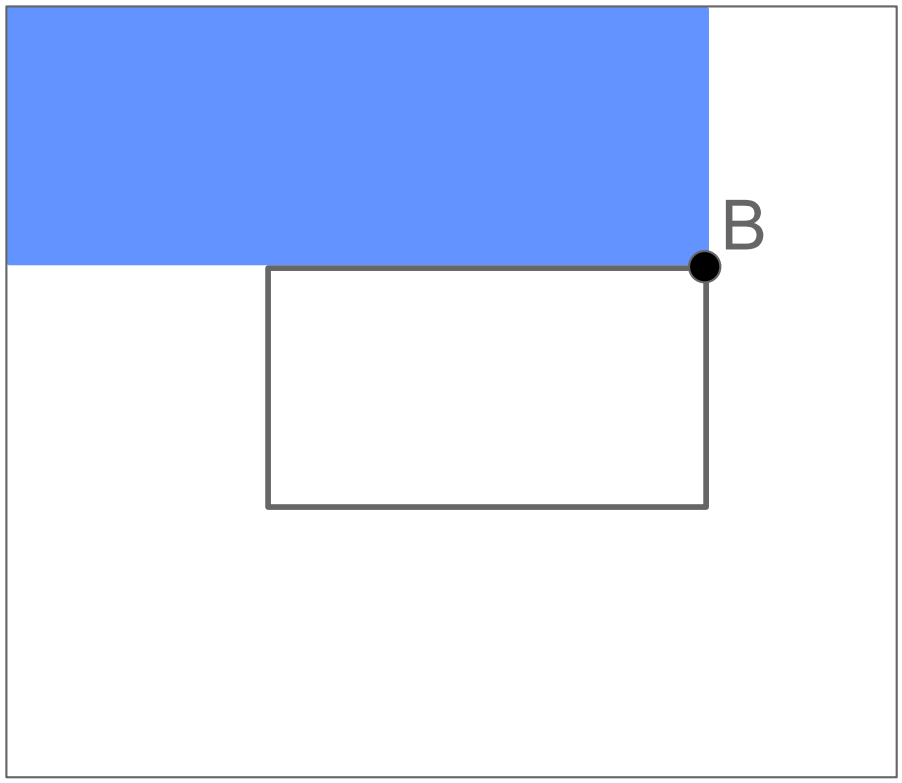
integral image

Integral images

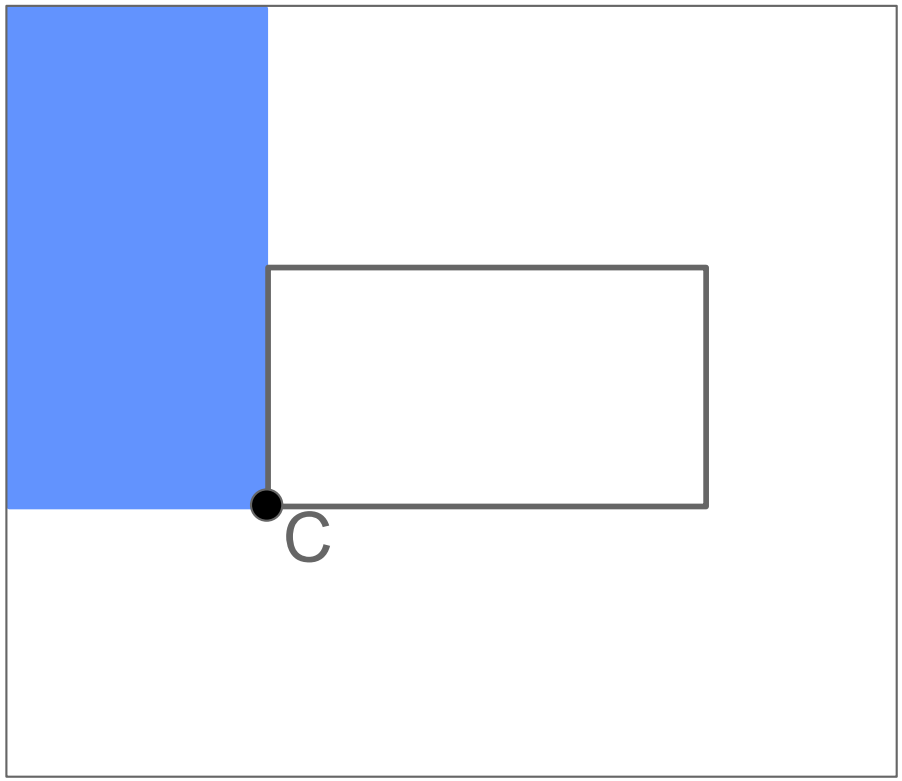


integral image

Integral images

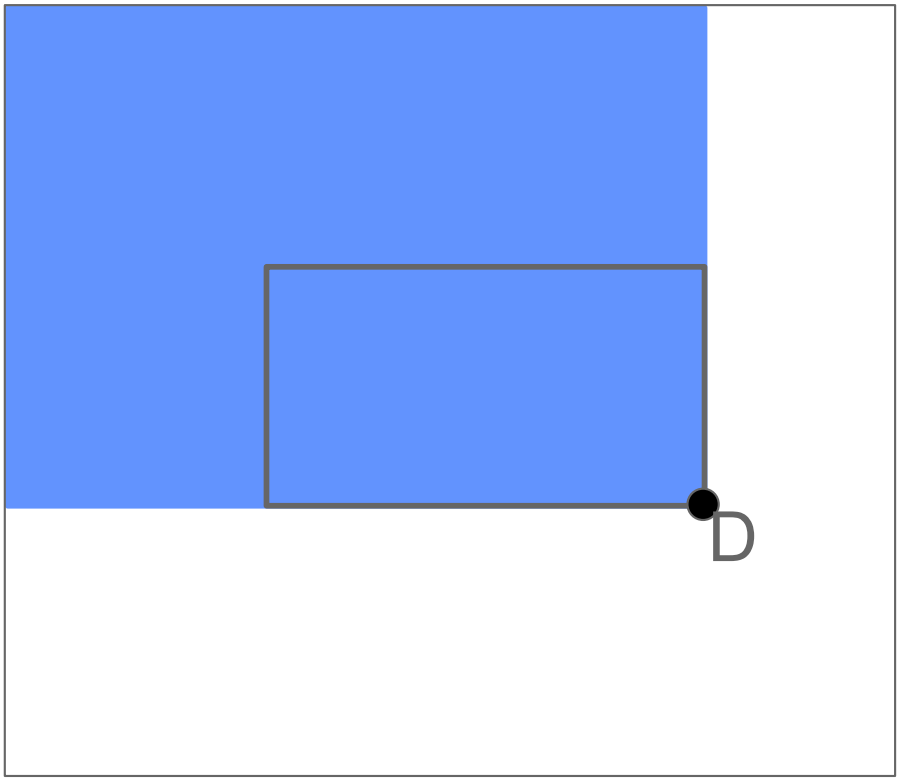


Integral images



integral image

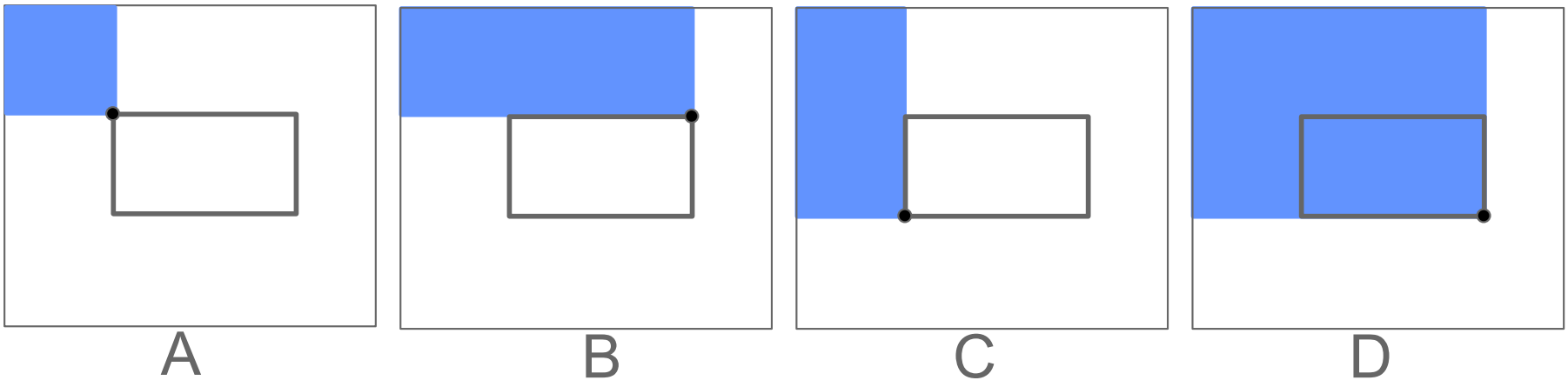
Integral images



integral image

Integral images

- What's the sum of pixels in the rectangle?

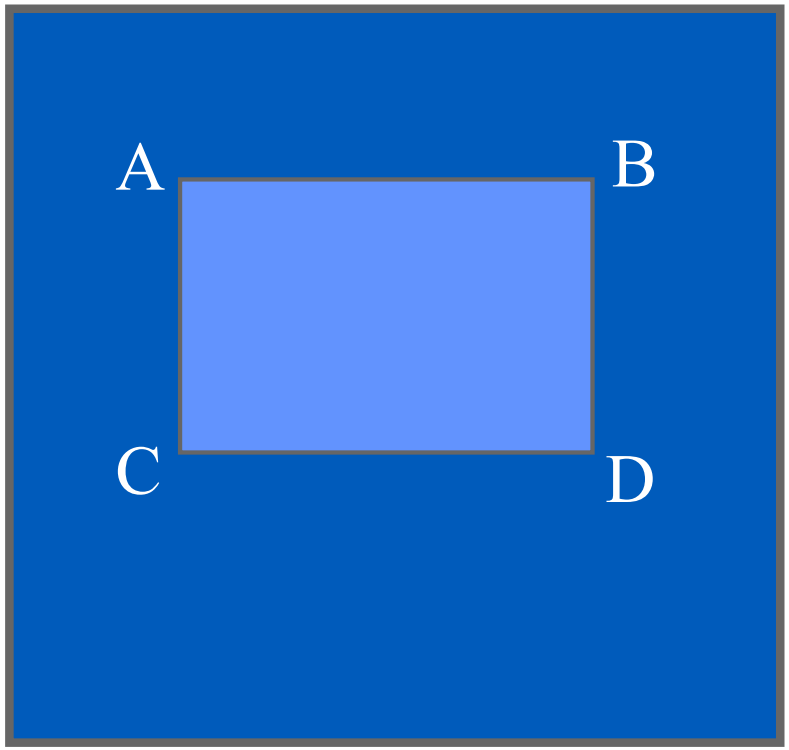


Computing sum within a rectangle

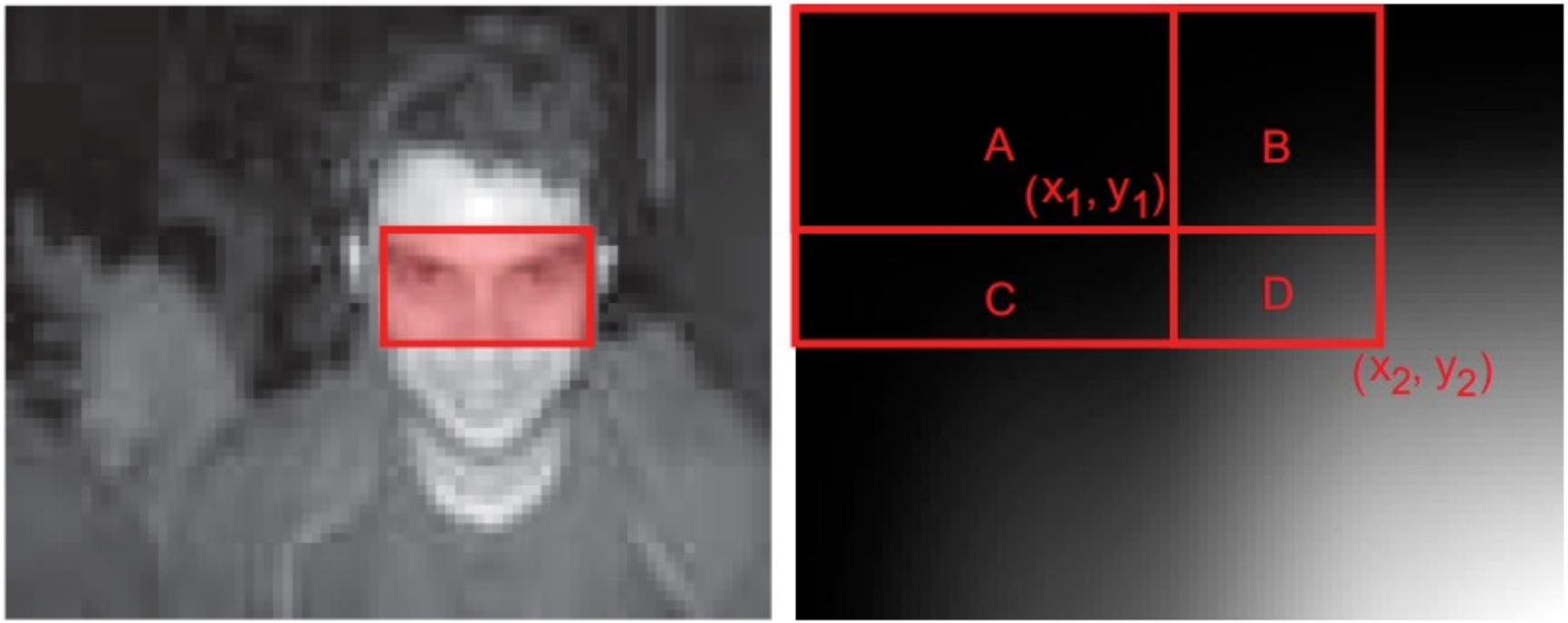
- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

  sum = D – B – C + A

- Only 3 additions are required for any size of rectangle!

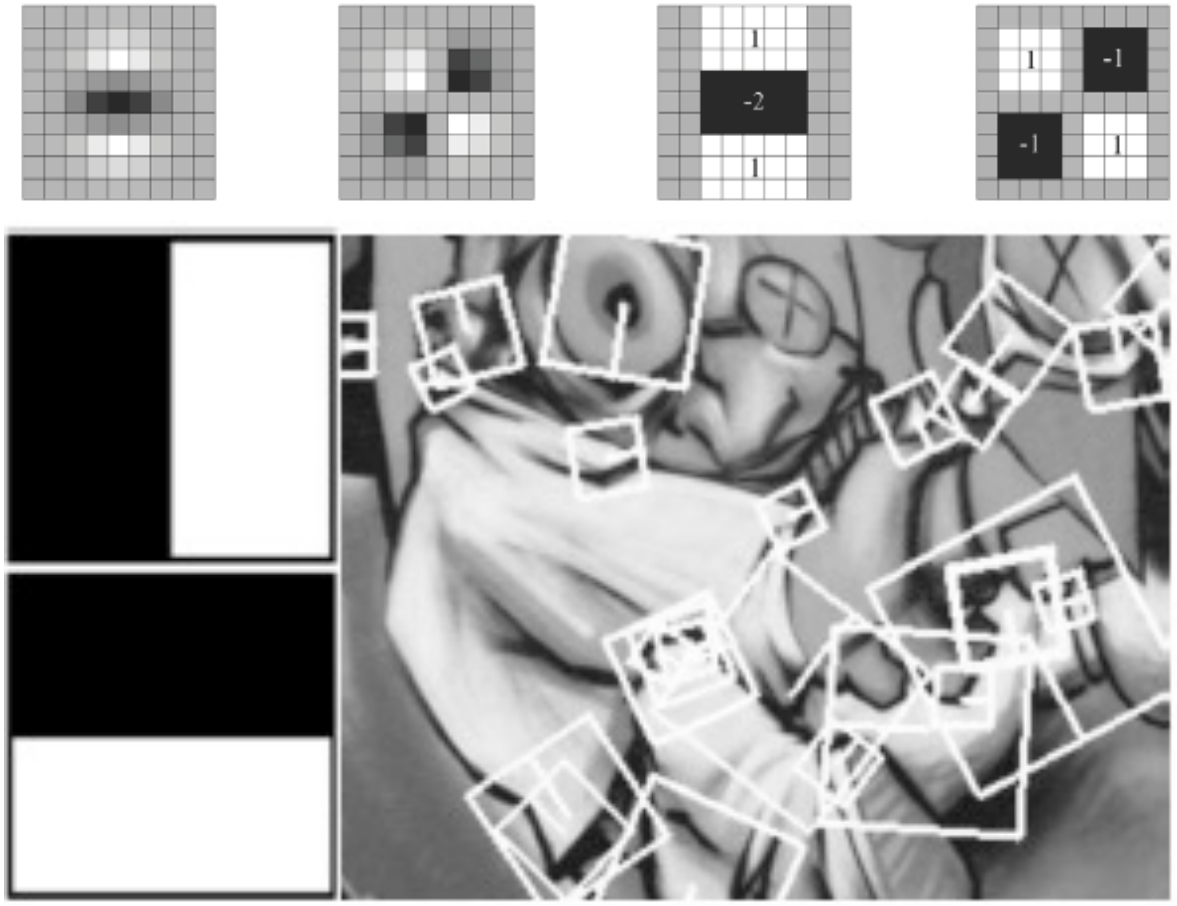


Integral Image Example



Local Descriptors: SURF

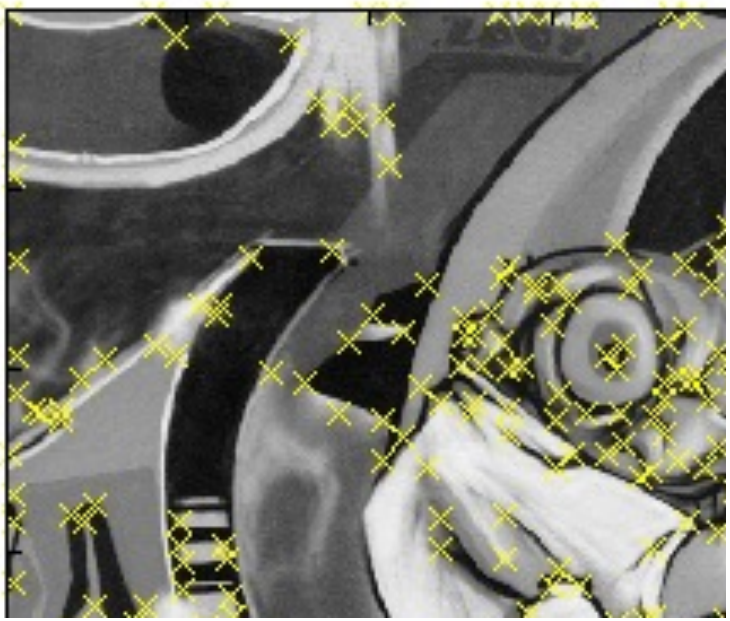
- Speeded up robust features (SURF)
  - Fast approximation of SIFT, 6 times faster.
- Accelerated by 2D filters (Harris) & **integral images.**



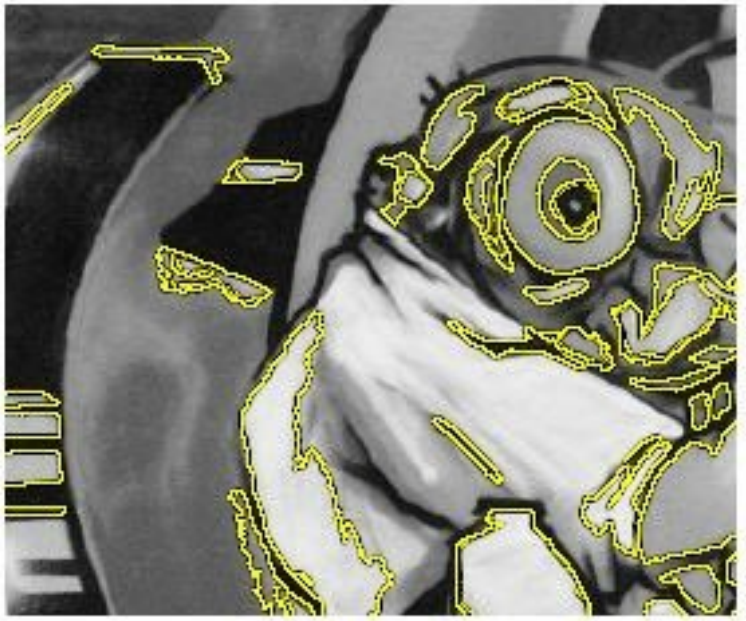
Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." *ECCV* (2006): 404-417.

Other Types of Detector

- Blob detection
  - regions that differ in properties, such as brightness or color, compared to surrounding regions.
- MSER detector (Maximally Stable Extremal Regions)
  - Constructed through trying multiple thresholds.



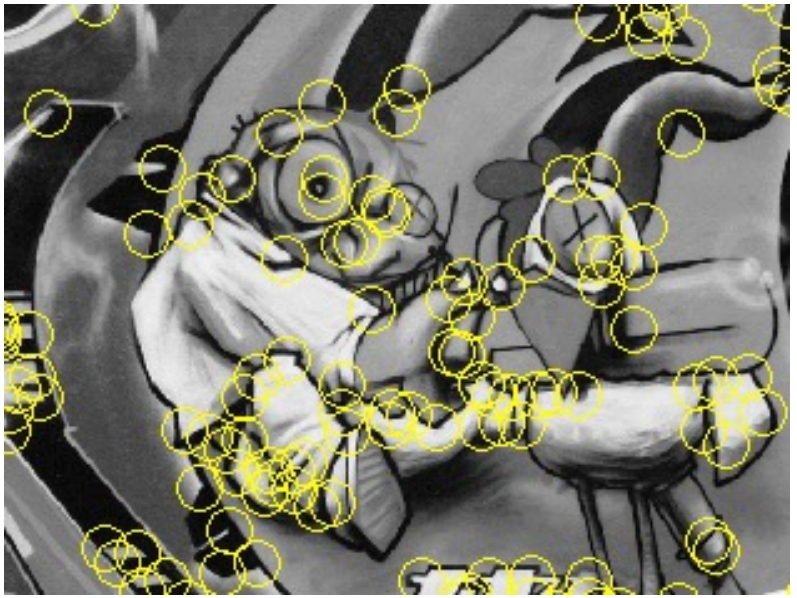
Harris



MSER

Important Concepts

- Key-point/Blob detection
  - Corners
  - Repeatable and Distinctive
  - Harris, DoG



- Descriptors
  - Robust and selective
  - Histograms of gradient orientation
  - SIFT, SURF