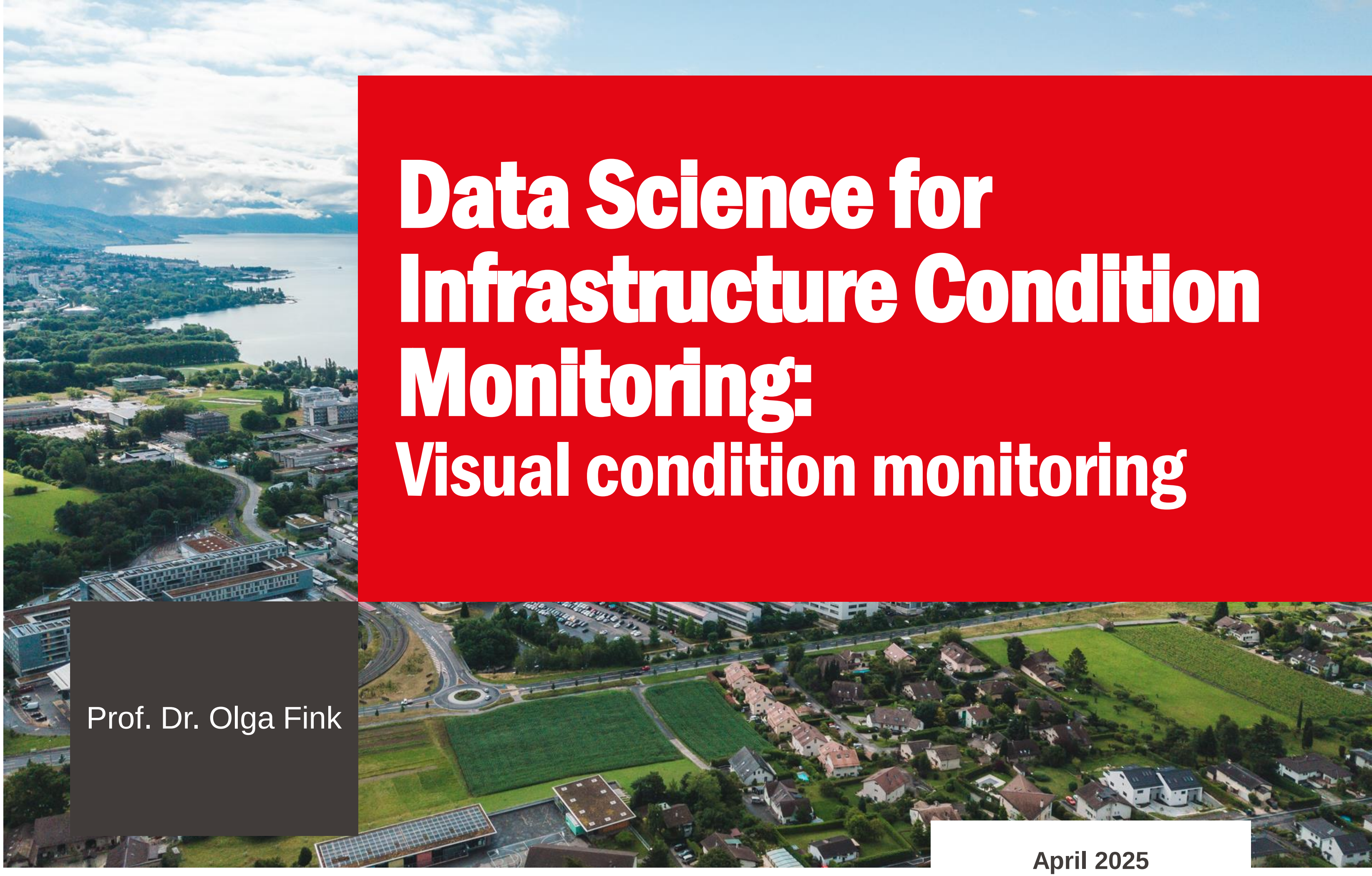


Data Science for Infrastructure Condition Monitoring:

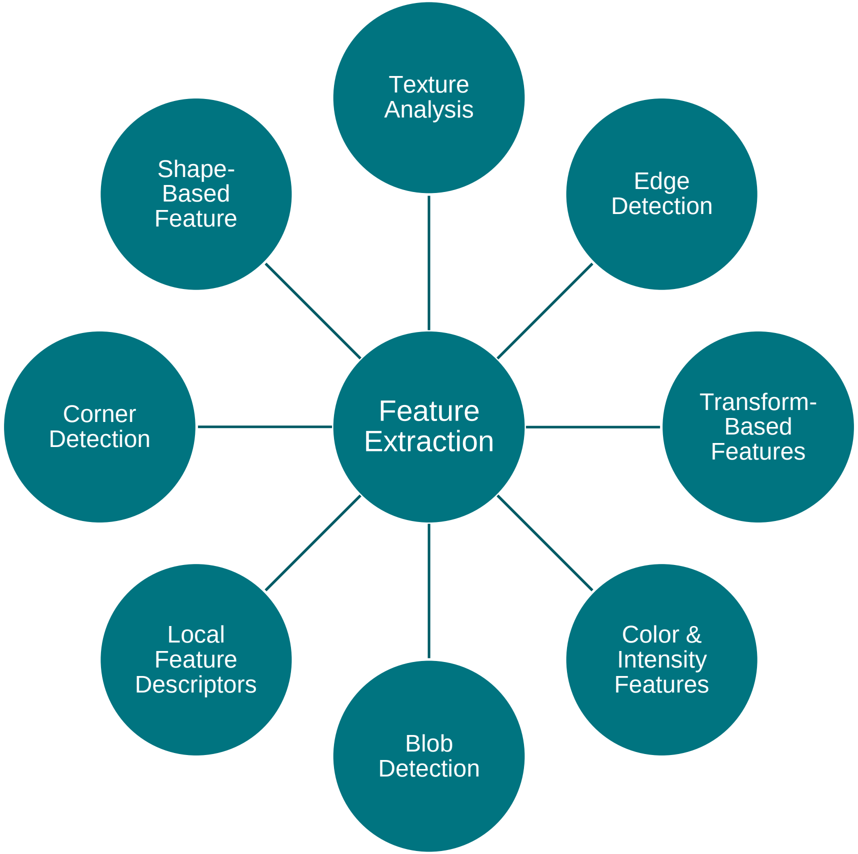
Visual condition monitoring

Prof. Dr. Olga Fink

April 2025

École polytechnique fédérale de Lausanne

Feature Extraction Techniques in Image Processing



Brief Description of Feature Extraction Techniques

- **Edge Detection**
  - Identifies points in an image where brightness changes sharply, used for object boundaries.
- **Texture Analysis**
  - Examines surface patterns and variations to classify regions based on their textures.
- **Corner Detection**
  - Detects points where edges meet, useful in motion tracking and object recognition.
- **Blob Detection**
  - Identifies regions in an image that differ in properties like brightness or color.
- **Shape-Based Feature**
  - Captures geometric properties of objects like contours and region boundaries.
- **Transform-Based Features**
  - Extracts features using transformations like Fourier or Wavelet for frequency domain analysis.
- **Local Feature Descriptors**
  - Captures distinctive patterns in image patches like SIFT, SURF, or ORB.
- **Color & Intensity Features**
  - Uses pixel color and brightness levels as features, helpful in segmentation and recognition

Visual condition monitoring: Examples

Types of infrastructures monitored visually

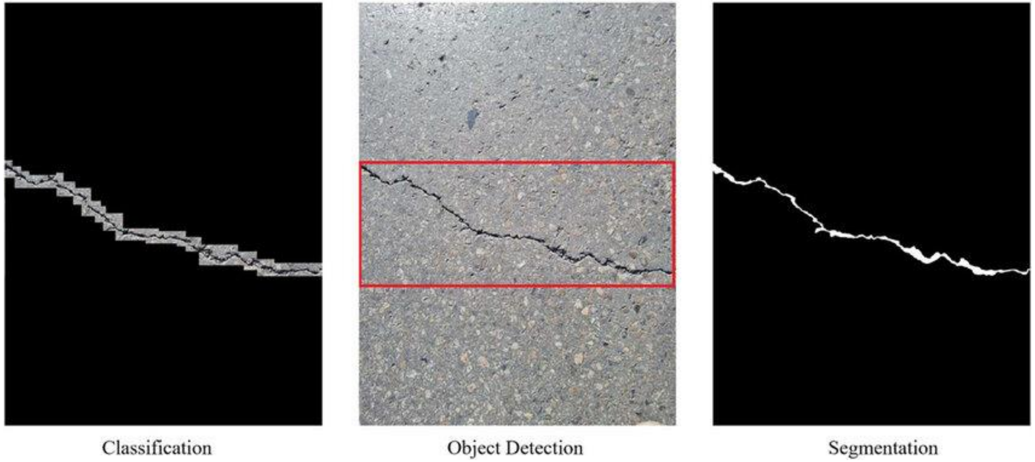
- Bridges
- Tunnels
- Underground pipes
- Roads: e.g. asphalt pavement
- Railway infrastructure (rails, sleepers, ballast, supporting walls…)
- Subsee infrastructure monitoring (e.g. pipeline corrosion)
- …

Essential steps in ML-based predictions

Condition monitoring data → Feature extraction → ML algorithm (e.g. classifier) → Prediction (label)

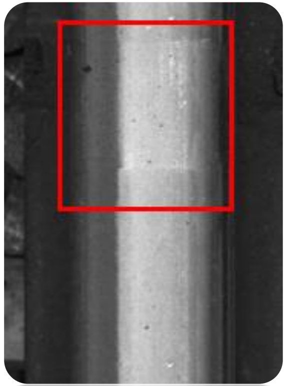
Performed tasks

- Classification
- Regression
- Segmentation
- Object detection
- Anomaly detection
- Reconstruction
- Matching
- …

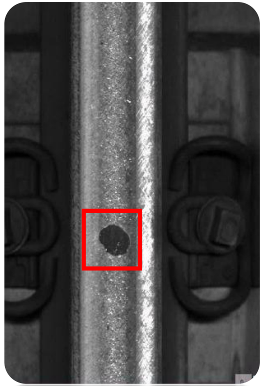


Defect detection of track: example SBB

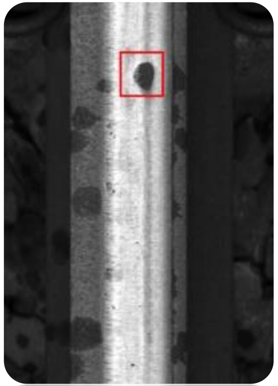
Welding



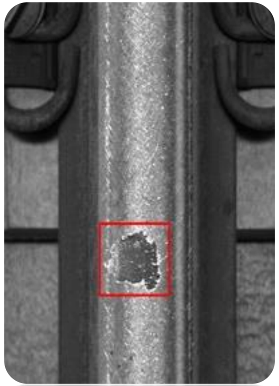
Plastic Particle



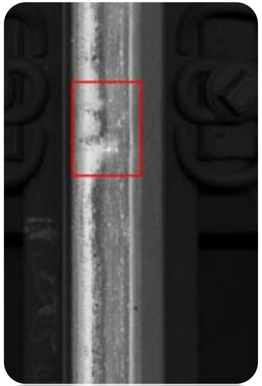
Surface Defect



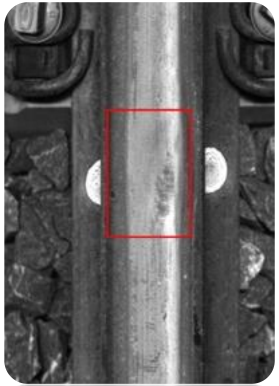
Chewing Gum



Squat

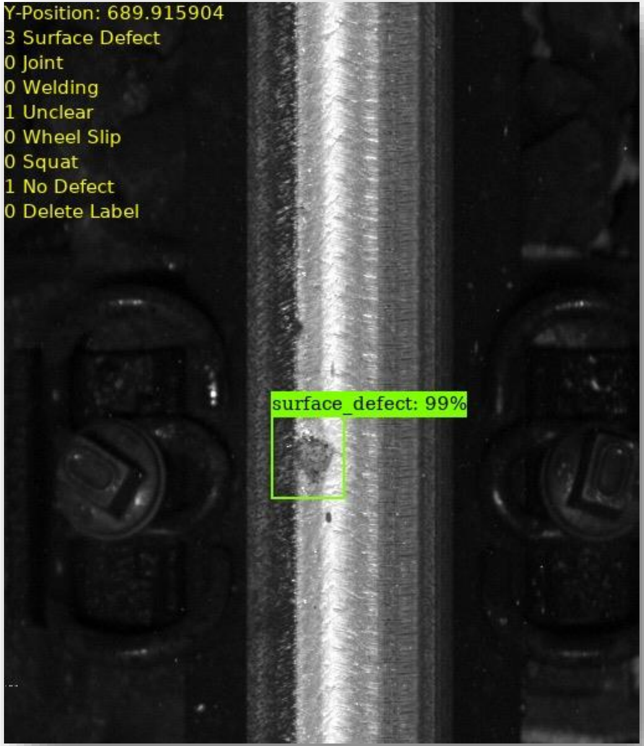


Wheel Slip



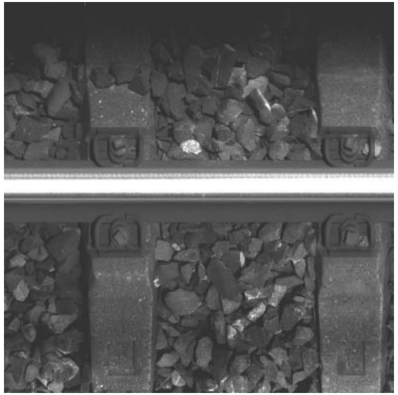
Source: J. Casutt, SBB

Defect detection of track: example SBB



Source: J. Casutt, SBB

Defect Type Classification of Sleepers : example SBB



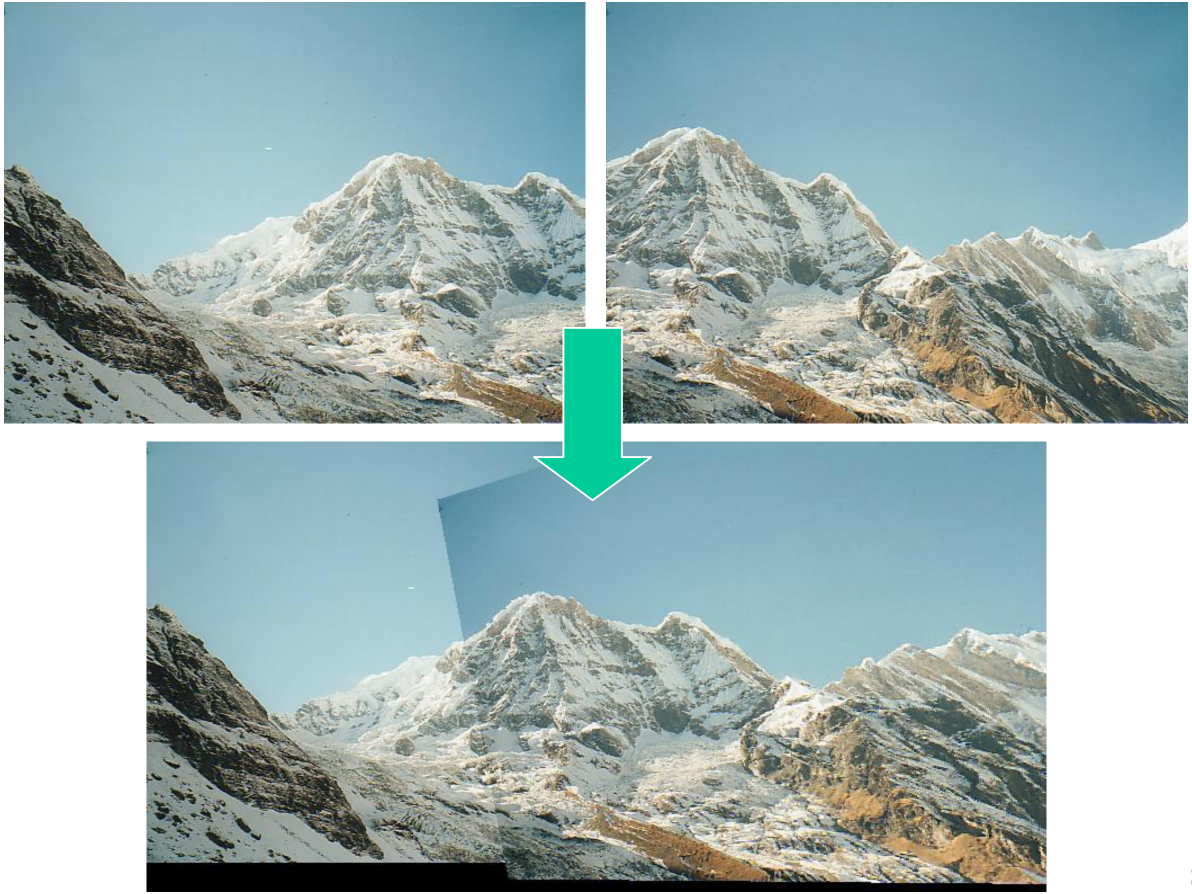
Healthy

Crack

Spalling

Feature detection and matching

Suppose you want to create a panorama



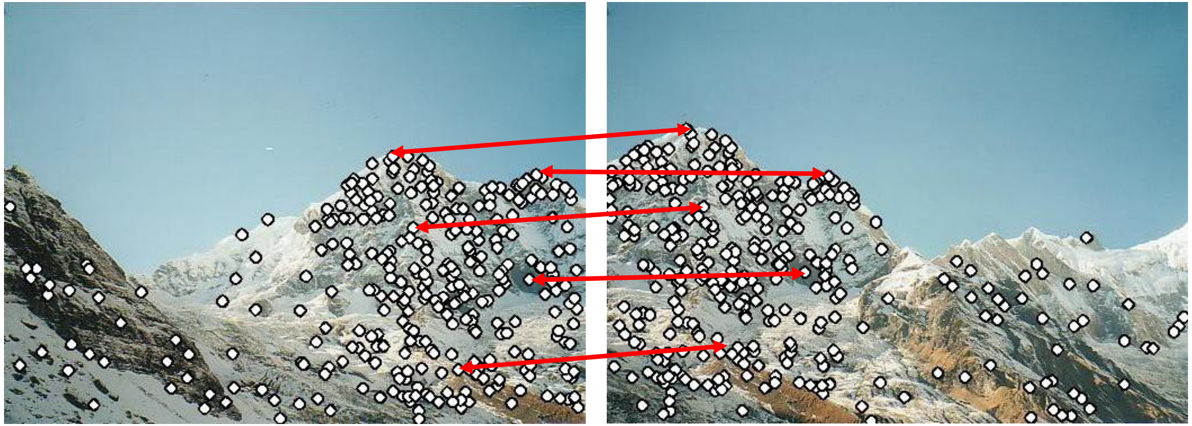
Source: Matthew Brown

What is the first step?

Need to match portions of images

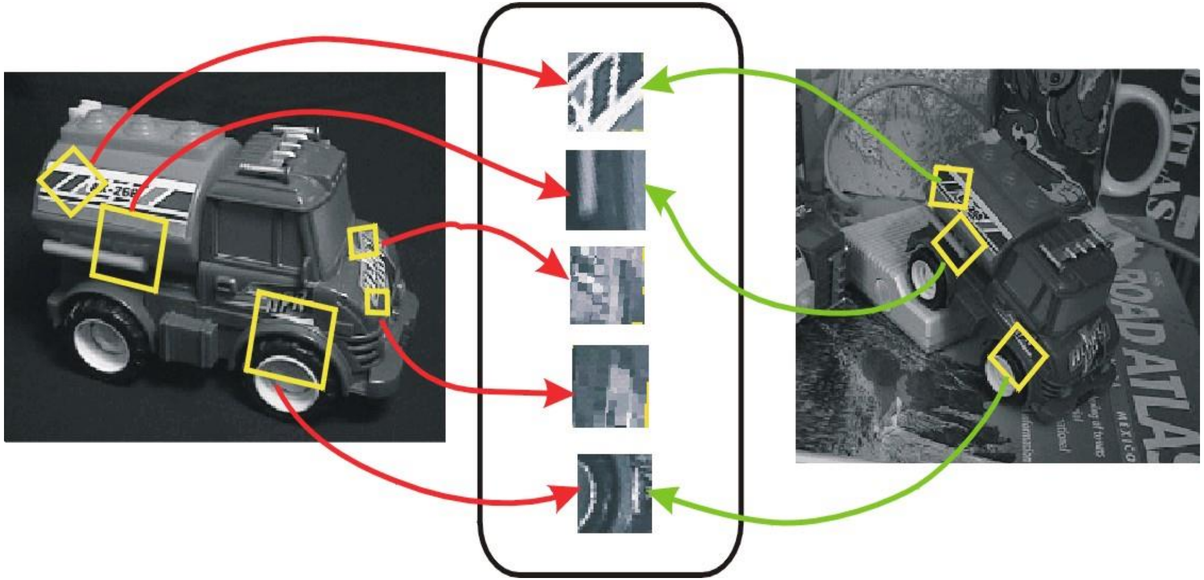
Source: UW CSE vision faculty

Solution: Match image regions using local features



Source: UW CSE vision faculty

Features can also be used for object recognition



Feature Descriptors

Source: UW CSE vision faculty

Why local features?

- Locality
  - features are local, so robust to occlusion and clutter
- Distinctiveness:
  - can differentiate a large database of objects
- Quantity
  - hundreds or thousands in a single image
- Efficiency
  - real-time performance achievable
- Generality
  - exploit different types of features in different situations

Source: UW CSE vision faculty

Applications

- Features are used for:
  - Image alignment (e.g., panoramic mosaics)
  - Object recognition
  - 3D reconstruction
  - Motion tracking
  - Indexing and content-based retrieval
  - …

Source: UW CSE vision faculty

Feature detection

What about edges?

- Edges can be invariant to brightness changes but typically not invariant to other transformations

Source: UW CSE vision faculty

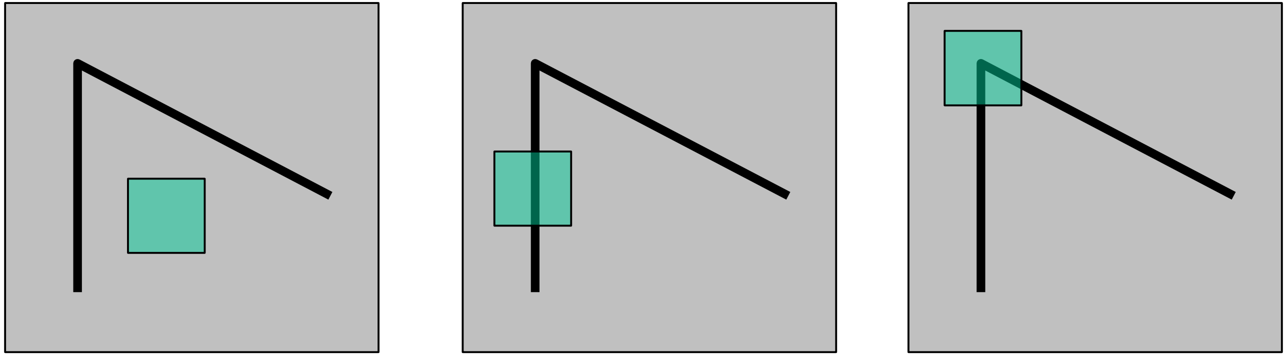
What makes a good feature?

- Want uniqueness
  - Leads to unambiguous matches in other images
- Look for “interest points”: image regions that are unusual
- How to define “unusual”?

Source: UW CSE vision faculty

Finding interest points in an image

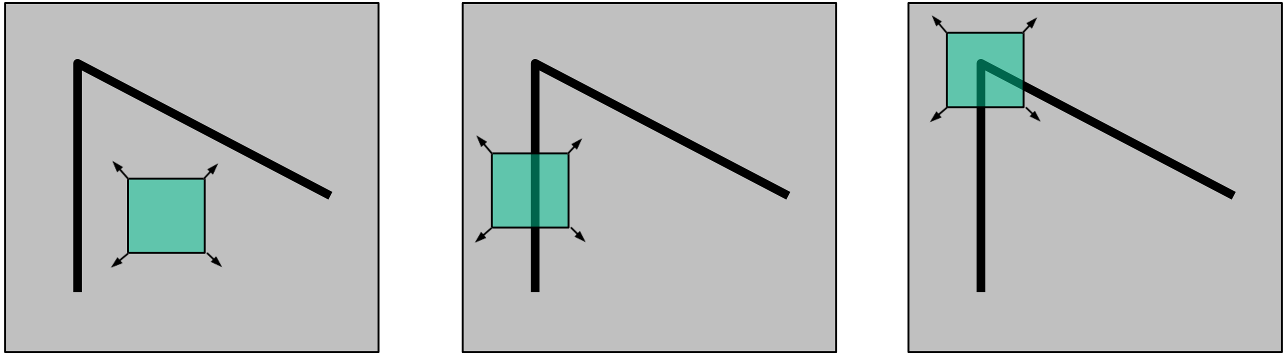
- Suppose we only consider a small window of pixels
  - What defines whether a feature is a good or bad candidate?



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

Finding interest points in an image

- How does the window change when you shift it?



"flat" region: no change in all directions

"edge": no change along the edge direction

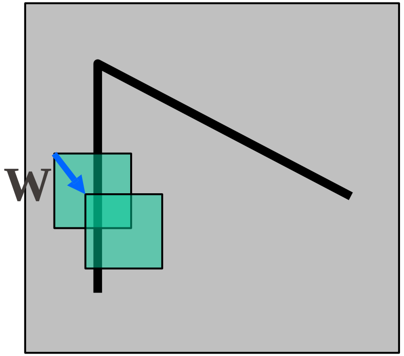
"corner": significant change in all directions, i.e., even the minimum change is large

- Find locations such that the minimum change caused by shifting the window in any direction is large

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

Finding interest points (Feature Detection): the math

- Consider shifting the window W by (u,v)
  - how do the pixels in W change?
  - compare each pixel before and after using the sum of squared differences (SSD)
  - this defines an SSD “error” E(u,v):



$$E(u,v) = \sum_{(x,y)\in W} \left[I(x+u, y+v) - I(x,y)\right]^2$$

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

Small motion assumption

Taylor Series expansion of *I*:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (*u*,*v*) is small, then first order approx. is good

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

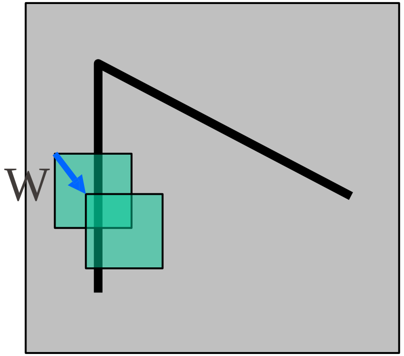
$$\approx I(x,y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

Plugging this into the formula on the previous slide…

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

Feature detection: the math



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2$$

$$\approx \sum_{(x,y)\in W} \left[I(x,y) + [I_x \; I_y]\begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)\right]^2$$

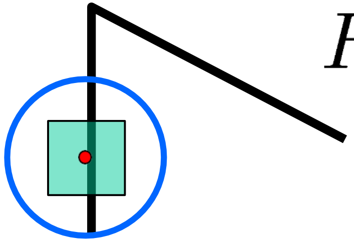
$$\approx \sum_{(x,y)\in W} \left[[I_x \; I_y]\begin{bmatrix} u \\ v \end{bmatrix}\right]^2$$

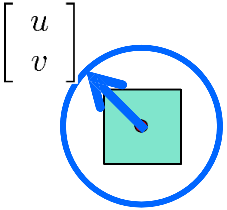
Slide adapted form Darya Frolova, Denis Simakov, Weizmann Institute.

Feature detection: the math

This can be rewritten:

$$E(u,v) \approx [u \;\; v] \underbrace{\left(\sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right)}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$





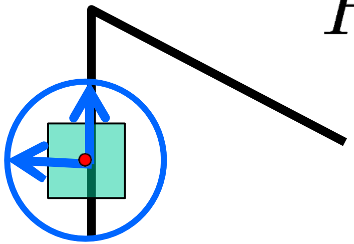
For the example above:

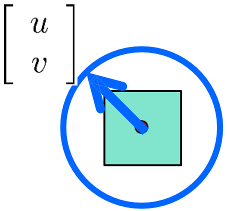
- You can move the center of the green window to anywhere on the blue unit circle
- How do we find directions that will result in the largest and smallest E values?
- Find these directions by looking at the eigenvectors of ***H***

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

Feature detection: the math

$$E(u,v) \approx [u \;\; v] \underbrace{\left(\sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right)}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$





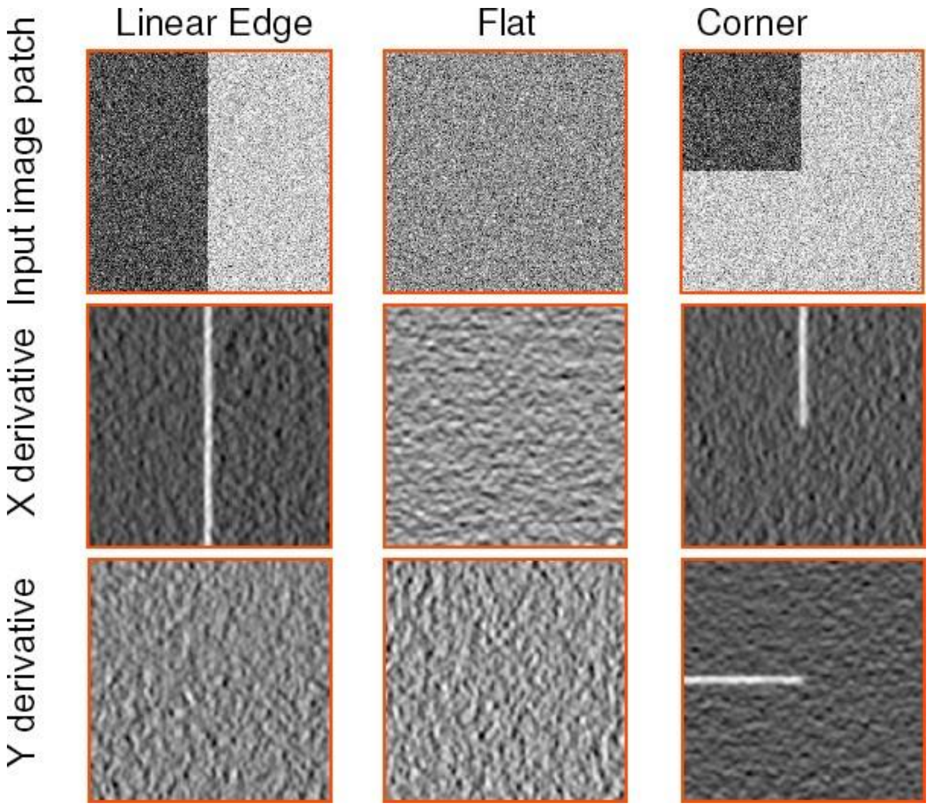
Eigenvalues and eigenvectors of H

- Capture shifts with the smallest and largest change (E value)
- x_+ = direction of **largest** increase in E.
- λ_+ = amount of increase in direction x_+
- x_- = direction of **smallest** increase in E.
- λ_- = amount of increase in direction x_-

$$\begin{aligned} Hx_+ &= \lambda_+ x_+ \\ Hx_- &= \lambda_- x_- \end{aligned}$$

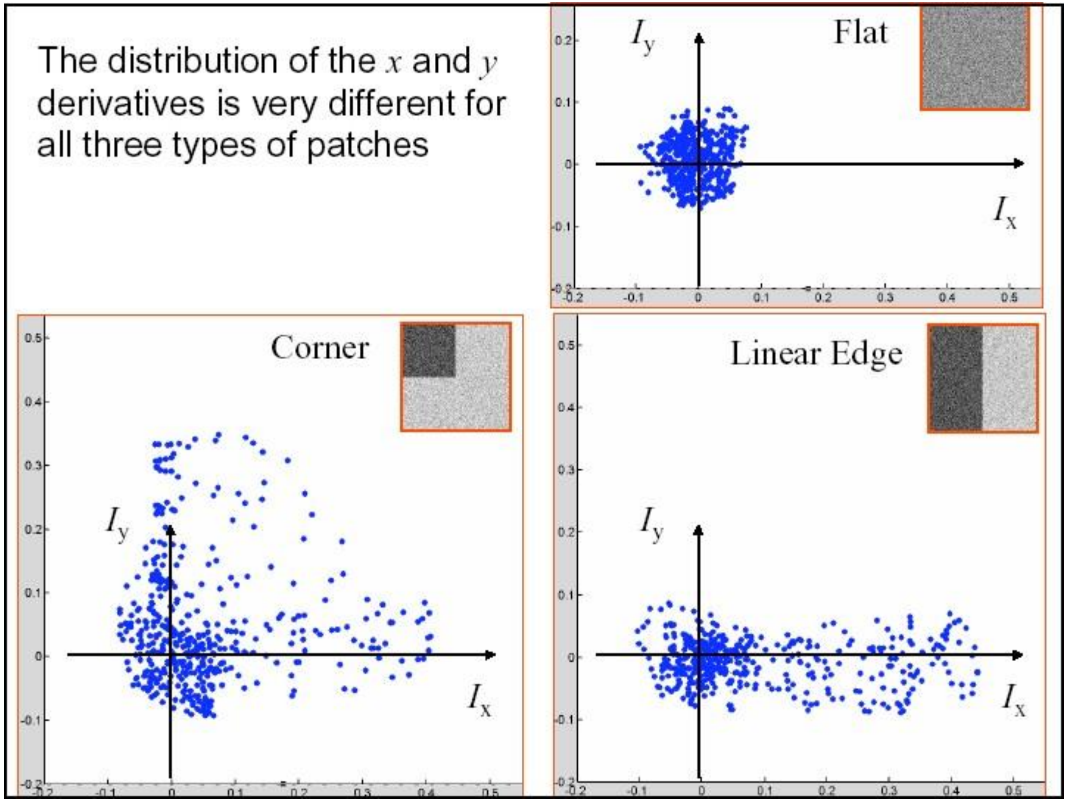
Slide adapted form Darya Frolova, Denis Simakov, Weizmann Institute.

Example: Cases and 2D Derivatives



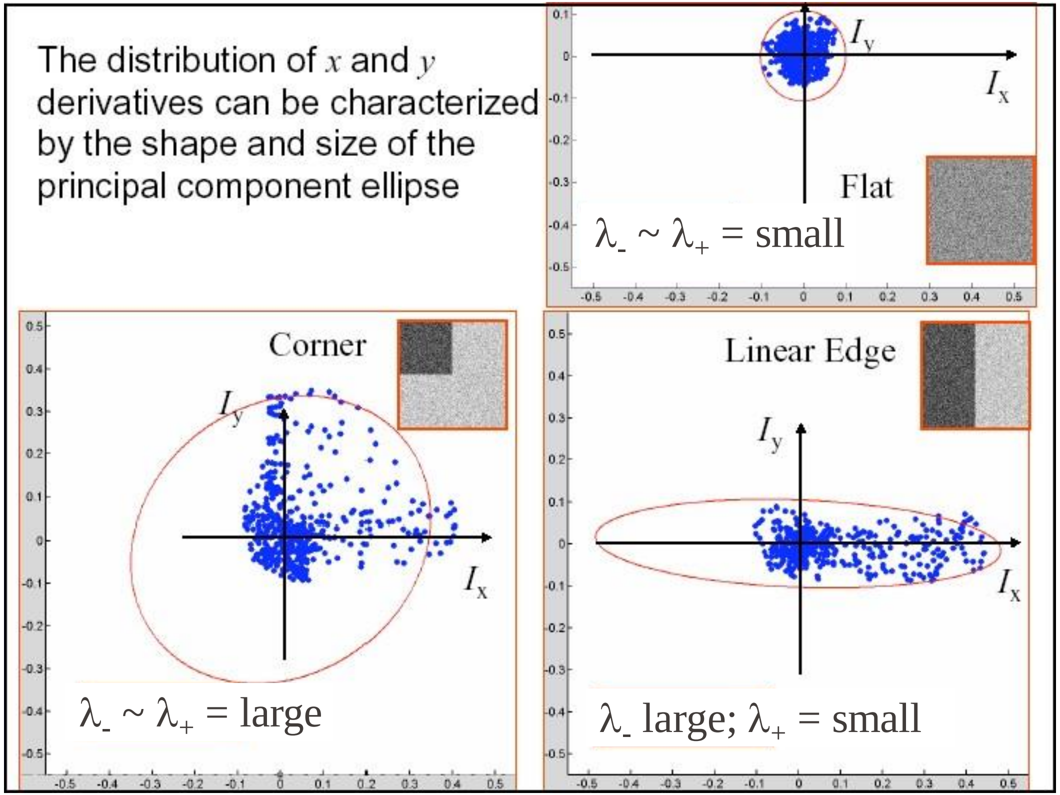
Slide adapted from Robert Collins

Plotting Derivatives as 2D Points



Slide adapted from Robert Collins

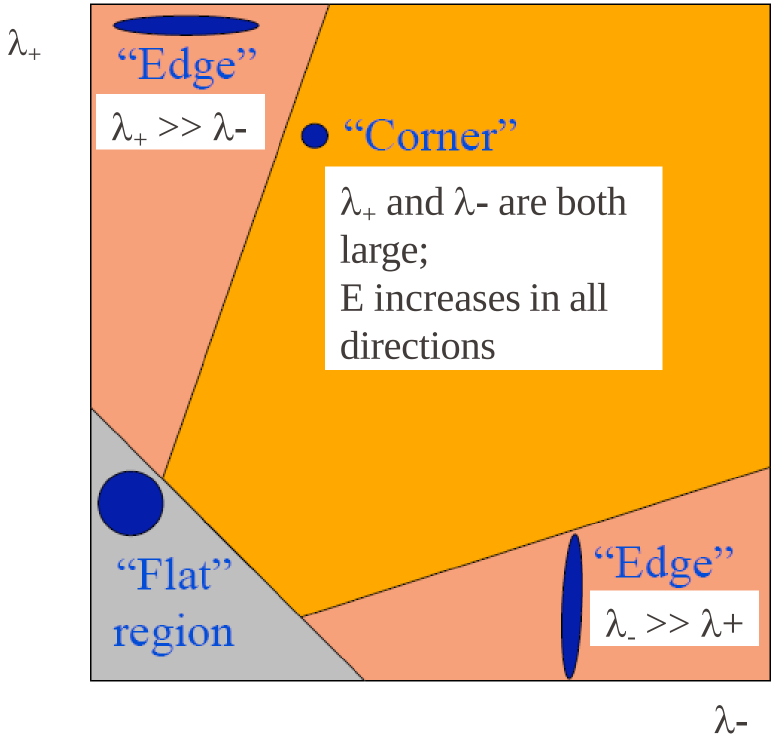
Fitting Ellipse to each Set of Points



Slide adapted from Robert Collins

Feature detection: the math

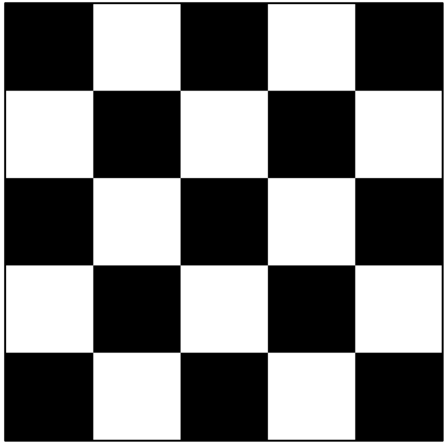
How are λ_+, $\mathbf{x}_+$, λ_-, and $\mathbf{x}_-$ relevant for feature detection?



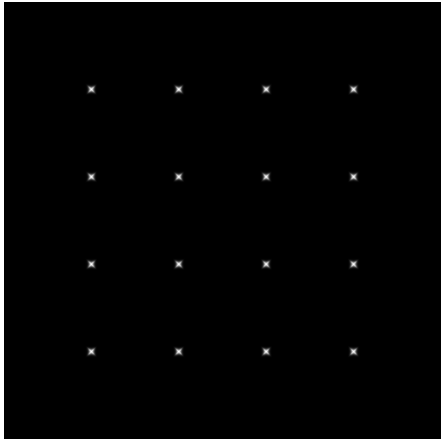
Source: UW CSE vision faculty

Feature detection summary

- Here's what you do
  - Compute the gradient at each point in the image
  - Create the H matrix from the entries in the gradient
  - Compute the eigenvalues
  - Find points with large λ_- (i.e., λ_- > threshold)
  - Choose points where λ_- is a local maximum as interest points



I

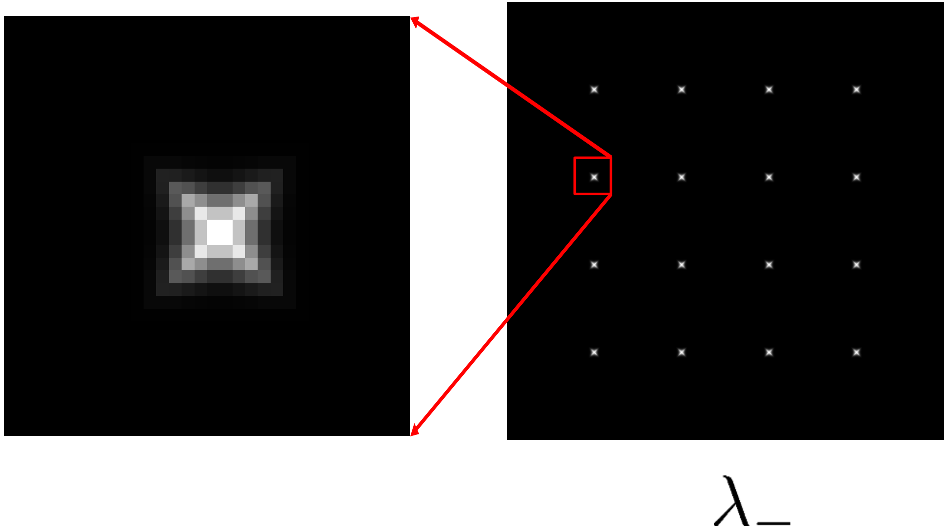


λ_-

Source: UW CSE vision faculty

Feature detection summary

- Here's what you do
  - Compute the gradient at each point in the image
  - Create the H matrix from the entries in the gradient
  - Compute the eigenvalues
  - Find points with large λ_- (i.e., λ_- > threshold)
  - Choose points where λ_- is a local maximum as interest points



Source: UW CSE vision faculty

The Harris operator

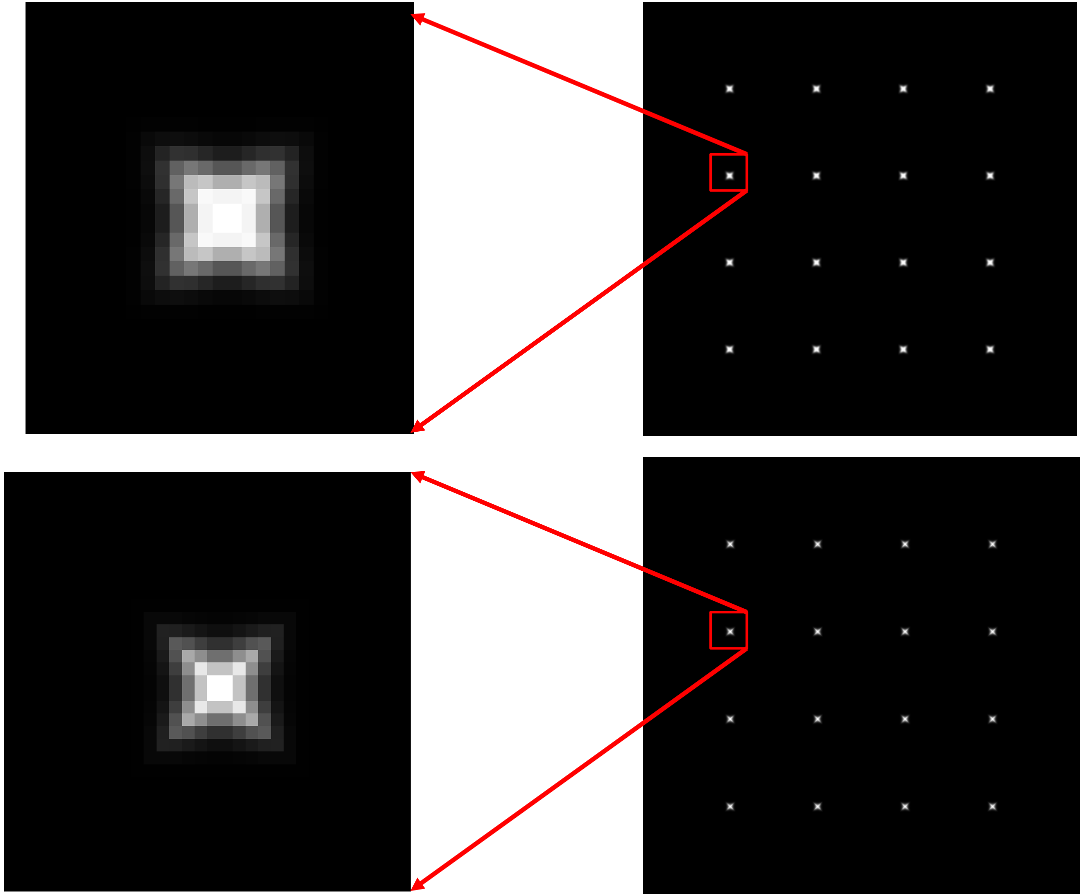
λ_- is a variant of the "Harris operator" for feature detection

$$f_{Harris} = \lambda_+ \lambda_- - k(\lambda_+ + \lambda_-)^2 = (h_{11}h_{22} - h_{12}h_{21}) - k(h_{11} + h_{22})^2$$
$$= \det(H) - k \operatorname{trace}(H)^2$$

- det is the determinant; trace = sum of diagonal elements of a matrix
- Very similar to λ_- but less expensive (no eigenvalue computation)
- Called the "Harris Corner Detector" or "Harris Operator"
- Most popular among all detectors

Source: UW CSE vision faculty

The Harris operator



Harris operator

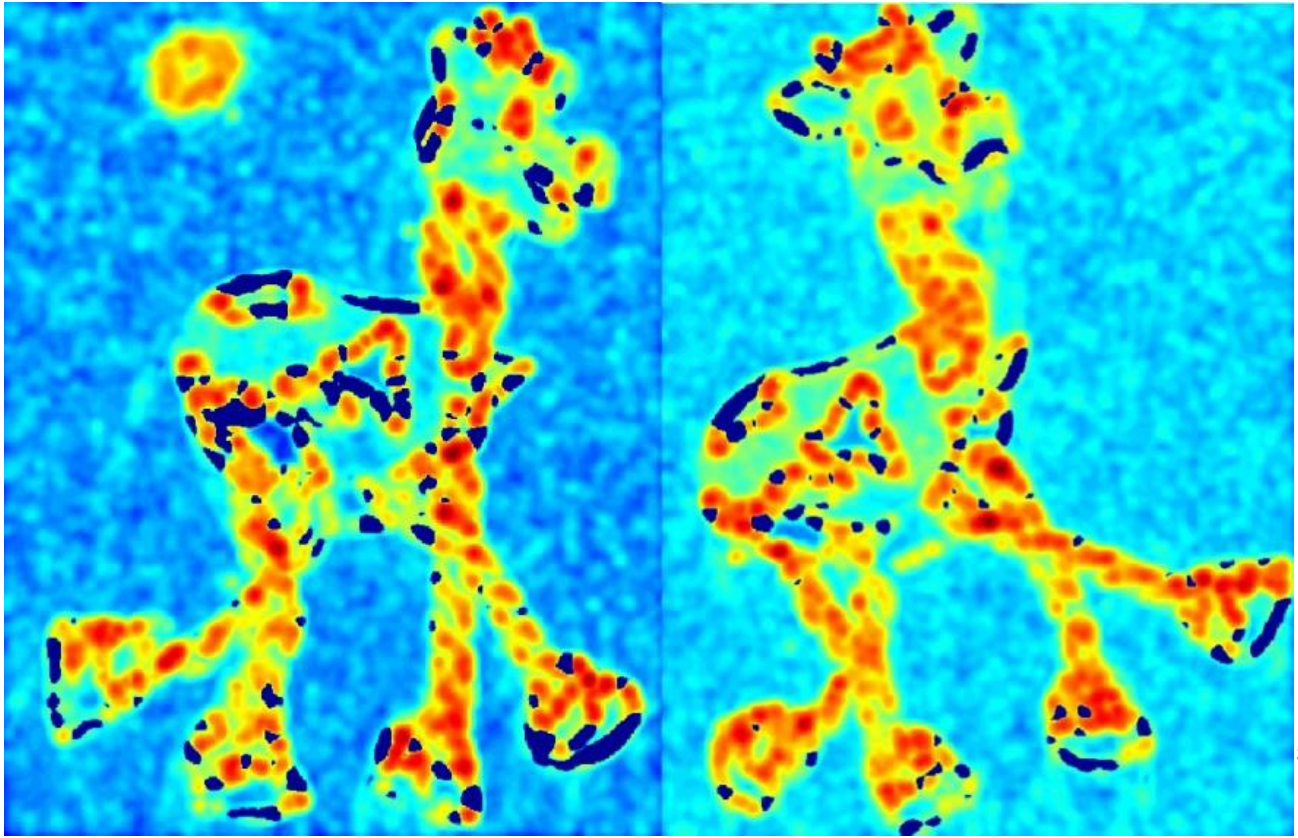
λ_-

Source: UW CSE vision faculty

Harris detector example

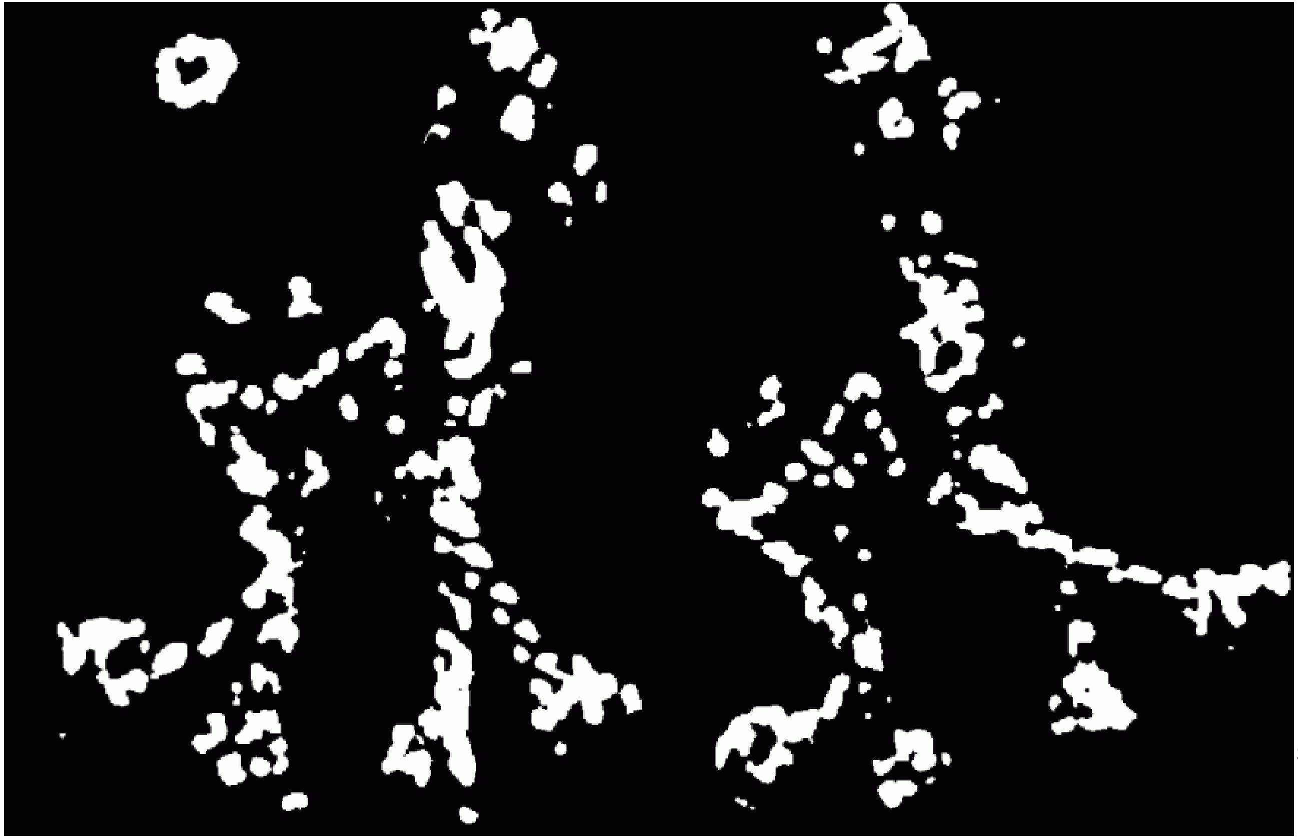
Source: UW CSE vision faculty

f_{Harris} value (red high, blue low)



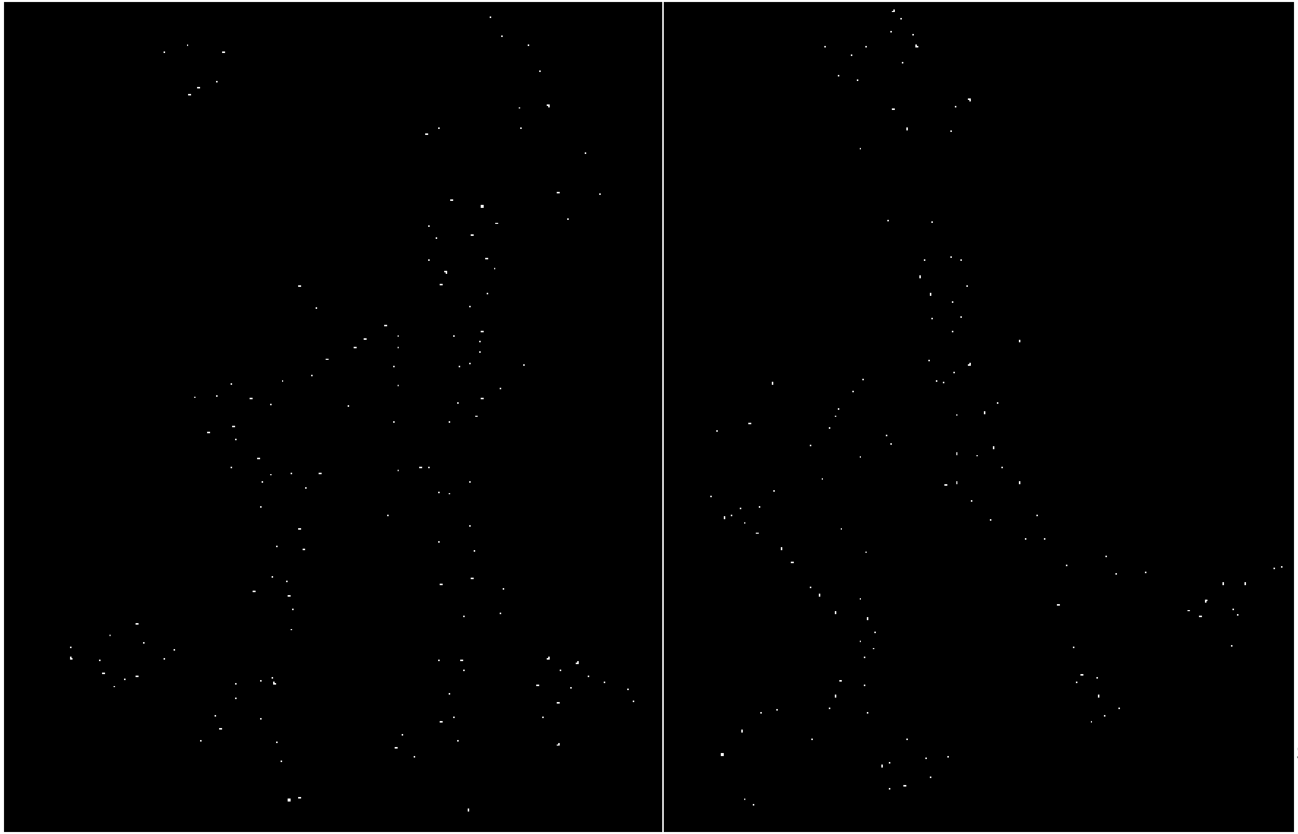
Source: UW CSE vision faculty

Threshold (f_{Harris} > threshold value)



Source: UW CSE vision faculty

Find local maxima of f_{Harris}



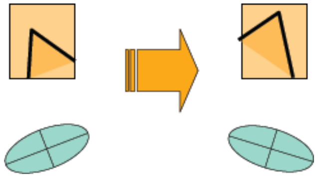
Source: UW CSE vision faculty

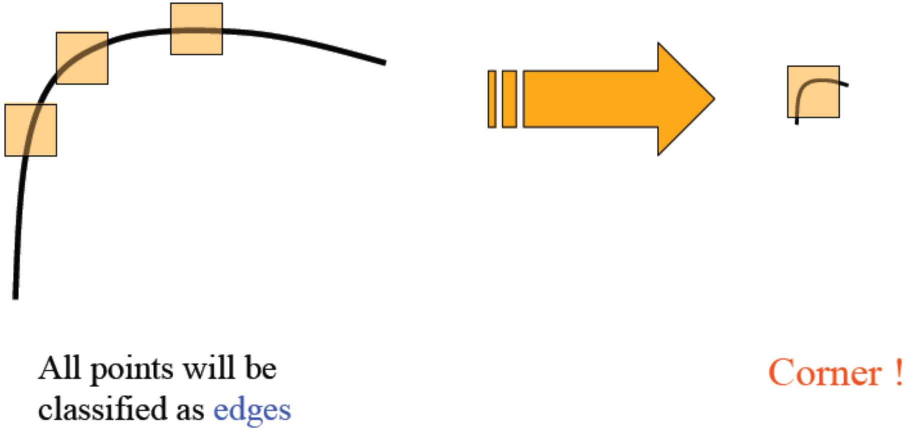
Harris features (in red)

Source: UW CSE vision faculty

Invariance of Eigenvalue-based feature detectors

- Suppose you rotate the image by some angle
  - Will you still pick up the same feature points?
  - Yes (since eigenvalues remain the same)
- What if you change the brightness?
  - Will you still pick up the same feature points?
  - Mostly yes (uses gradients which involve pixel differences)
- Scale?
  - No!





Source: UW CSE vision faculty

Feature matching

How to achieve invariance in image matching

- Two steps:

1. Make sure your feature detector is invariant
   - Harris is invariant to translation and rotation
   - Scale is trickier
     - common approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS)
     - More sophisticated methods find “the best scale” to represent each feature (e.g., SIFT)
2. Design an invariant feature descriptor
   - A descriptor captures the intensity information in a region around the detected feature point
   - The simplest descriptor: a square window of pixels
     - What’s this invariant to?

Source: UW CSE vision faculty

Rotation invariance for feature descriptors

Find dominant orientation of the image patch

- This is given by $\mathbf{x}_+$, the eigenvector of **H** corresponding to λ_+
  (λ_+ is the *larger* eigenvalue)
- Rotate the patch according to this angle

Figure by Matthew Brown

Source: UW CSE vision faculty

SIFT algorithm

Scale Invariant Feature Transform

- SIFT algorithm helps locate the local features in an image

→known as the '*keypoints*' of the image.

- These keypoints are **scale & rotation invariant**

→ can be used for various computer vision applications, like image matching, object detection, scene detection, etc

Source: UW CSE vision faculty

Four steps of the SIFT algorithm

1. **Constructing a Scale Space**

→ make sure that features are scale-independent

2. **Keypoint Localisation**

→ Identifying the suitable features or keypoints

3. **Orientation Assignment**

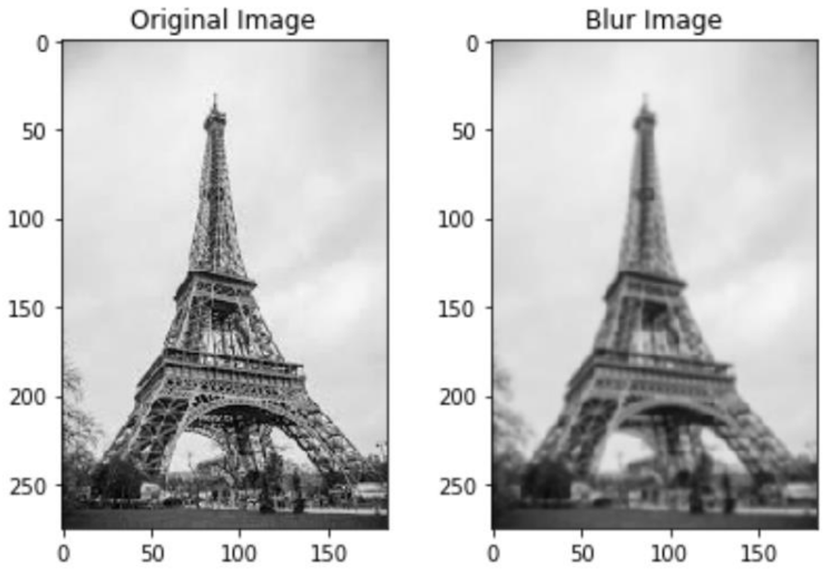
→ Ensure the keypoints are rotation invariant

4. **Keypoint Descriptor**

→ Assign a unique fingerprint to each keypoint

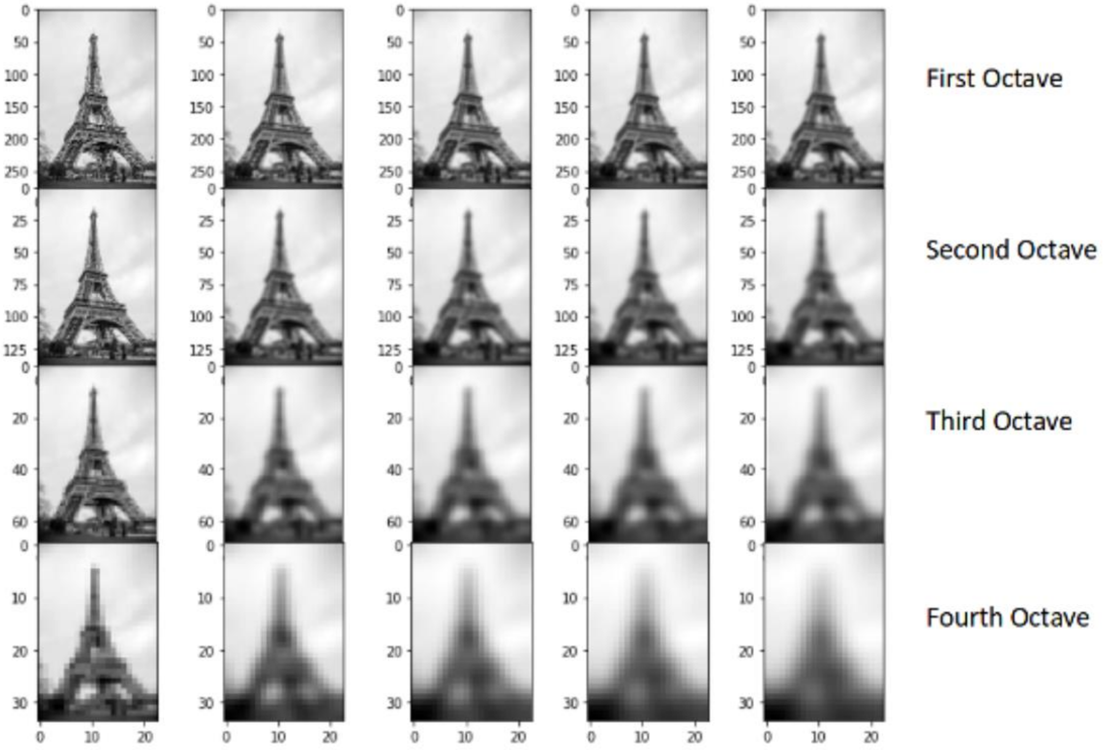
Source: UW CSE vision faculty

Gaussian Blurring



Source: www.datascience.com

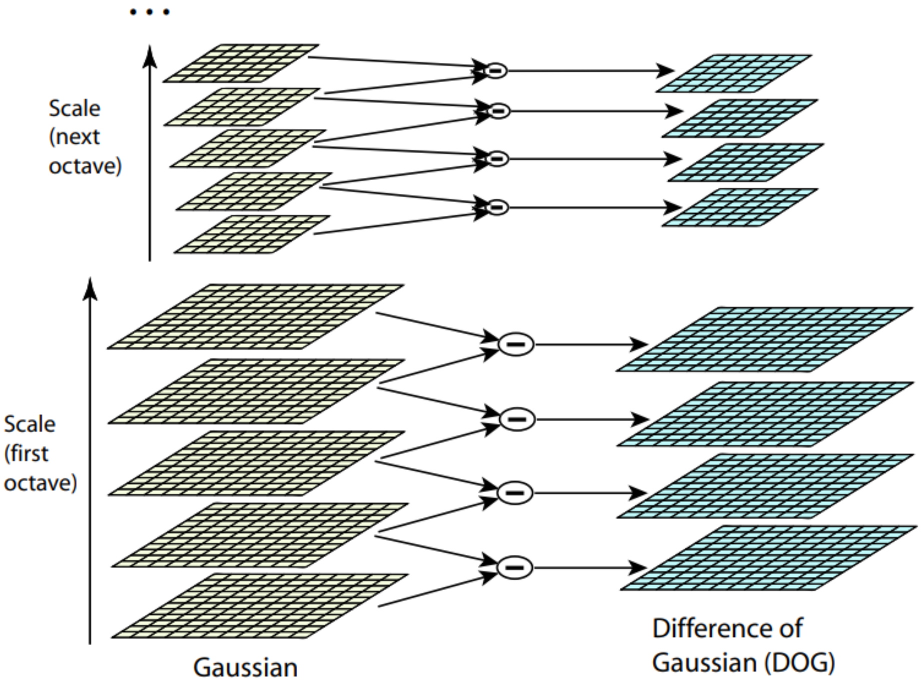
Scale + blur



Scale space is a collection of images having different scales, generated from a single image

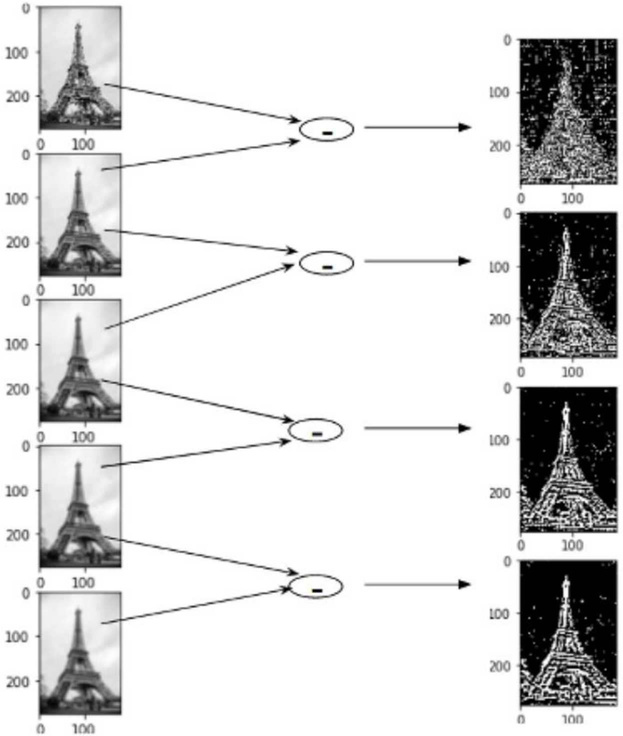
Source: www.datascience.com

Difference of Gaussians
Detection of scale-space extrema



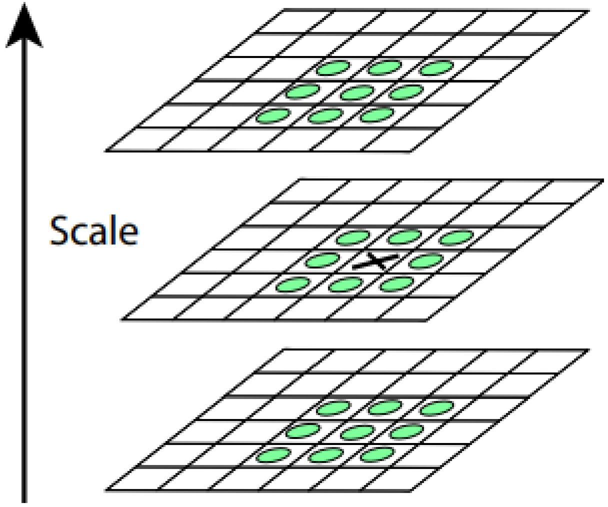
Source: Lowe, Distinctive Image Features from Scale-Invariant Keypoints, 2004

Difference of Gaussians
Detection of scale-space extrema



Source: www.datascience.com

Maxima and minima of the difference-of-Gaussian images

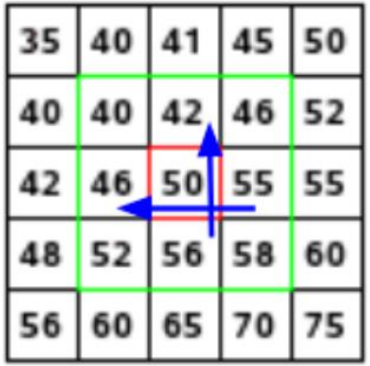


→ comparing a pixel (marked with X) to its 26 neighbors in 3x3 regions at the current and adjacent scales

Source: Lowe, Distinctive Image Features from Scale-Invariant Keypoints, 2004

Orientation asignment

- Calculate the magnitude and orientation
- Create a histogram for magnitude and orientation
- The bin at which we see the peak will be the orientation for the keypoint



Feature descriptor

- Use the neighboring pixels, their orientations, and their magnitude to generate a unique fingerprint for each keypoint called a ‘descriptor’
- Basic idea:
  - Take 16x16 square window around detected interest point (8x8 shown below)
  - Compute edge orientation (angle of the gradient minus 90°) for each pixel
  - Throw out weak edges (threshold gradient magnitude)
  - Create histogram of surviving edge orientations (8 bins)

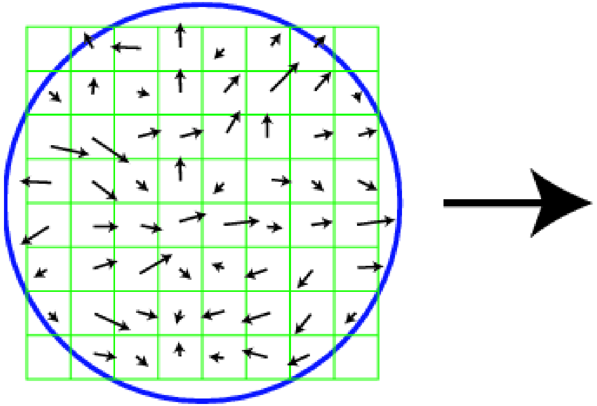
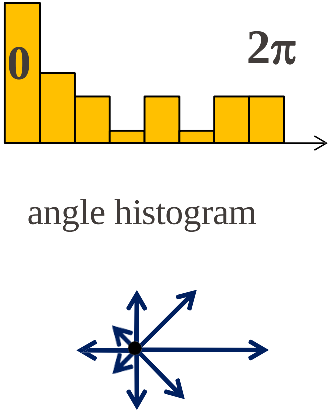


Image gradients

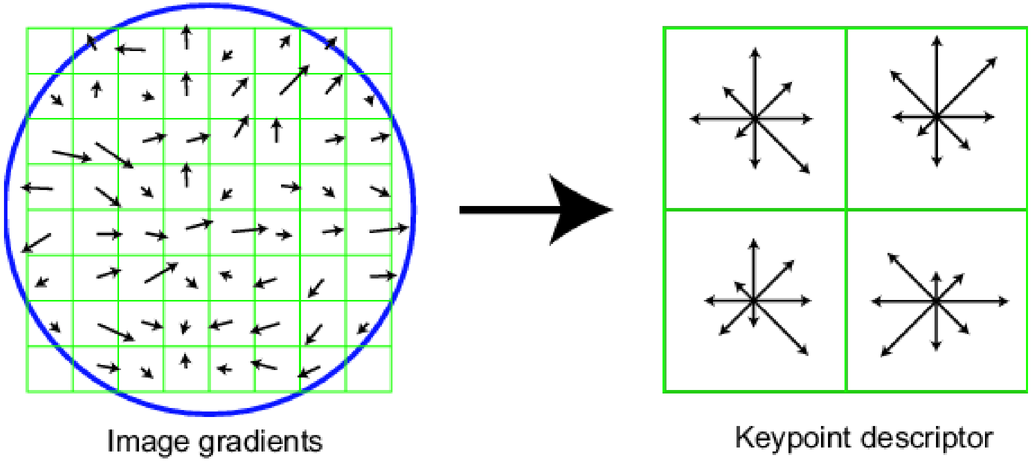


angle histogram

Source: Lowe, Distinctive Image Features from Scale-Invariant Keypoints, 2004

SIFT descriptor

- Full version
  - Divide the 16x16 window into a 4x4 grid of cells
  - (8x8 window and 2x2 grid shown below for simplicity)
  - Compute an orientation histogram for each cell
  - 16 cells * 8 orientations = 128 dimensional descriptor



Source: Lowe, Distinctive Image Features from Scale-Invariant Keypoints, 2004

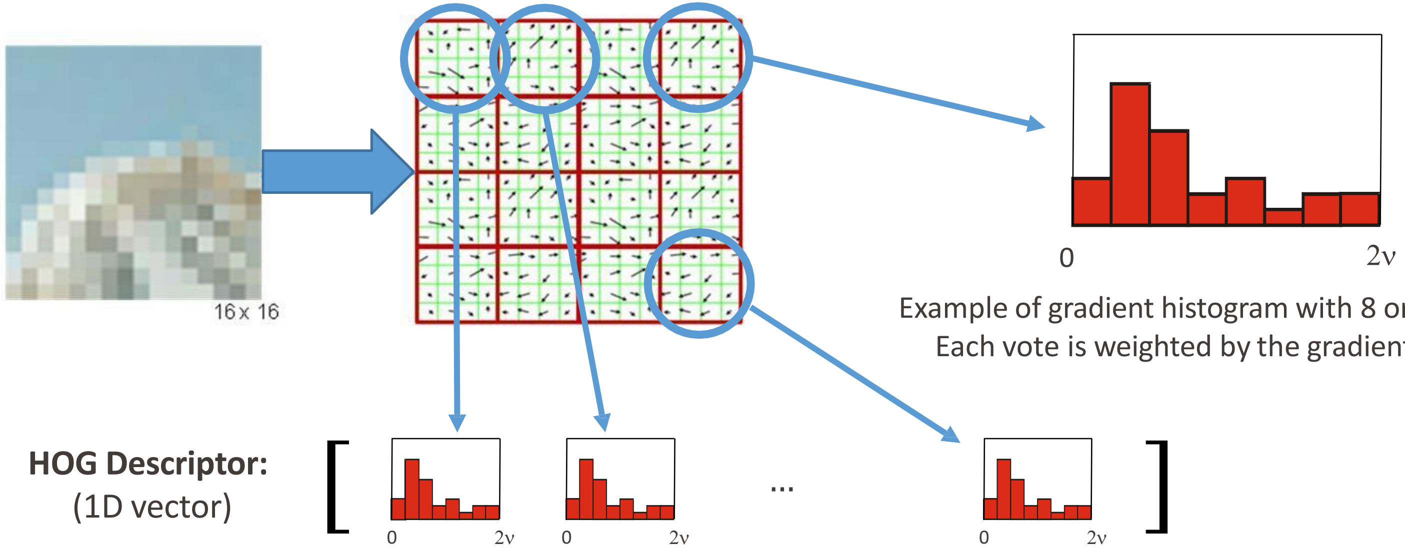
Properties of SIFT-based matching

- 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 (below)
  - Fast and efficient — can run in real time


Source: UW CSE vision faculty

HOG Descriptor (Histogram of Oriented Gradients)

- The patch is divided into a **grid of cells** and for each cell a **histogram of gradient directions** is compiled.
- The HOG descriptor is the **concatenation of these histograms** (used in SIFT)
- Differently from the patch descriptors, HOG has **float values**.



Example of gradient histogram with 8 orientation bins.
Each vote is weighted by the gradient magnitude

SIFT vs HOG Descriptors

Aspect	SIFT Descriptor	HOG Descriptor
Input Region	Local patch around each keypoint	Regular dense grid of patches across whole image
Detection	Tied to detected keypoints (interest points)	Not tied to keypoints; computed over sliding windows
Invariant to	Scale, rotation, illumination	Mostly illumination and pose
Descriptor Size	128-dim vector per keypoint	Large vector (e.g., 3780-dim) for whole image region
Orientation Encoding	Orientation histograms in 4x4 grid around point	Orientation histograms in cells (e.g., 8x8 px)
Normalization	Local (each keypoint is normalized)	Block-level normalization (overlapping cells)
Sparsity	Sparse descriptors (only at keypoints)	Dense descriptors (computed on fixed grid)
Computation	More complex (keypoint detection + descriptor)	Faster (just compute gradients + histograms)
SIFT Descriptor	4×4 grid of subregions, 8-bin histograms	N/A
HOG Descriptor	N/A	Histograms over blocks (2x2 cells), high dimensional
Intuition	Describes unique local keypoint features	Describes texture/edge flow of a region
Matching keypoints	Yes	No
Object detection	Rarely	Yes

Feature matching

Feature matching

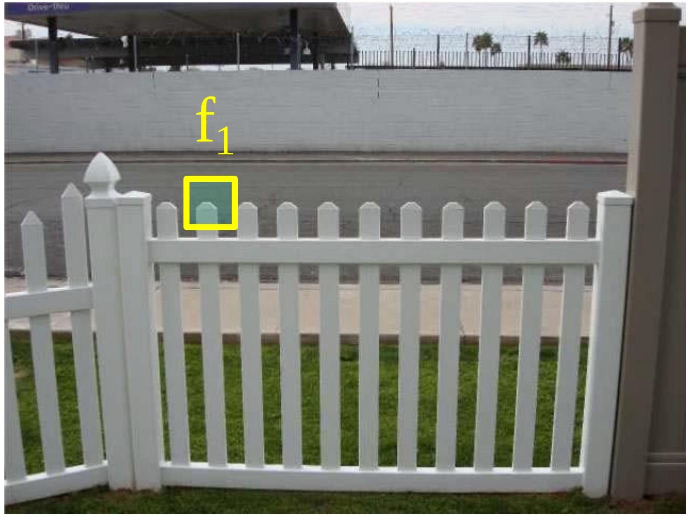
- Given a feature in I_1, how to find the best match in I_2?

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

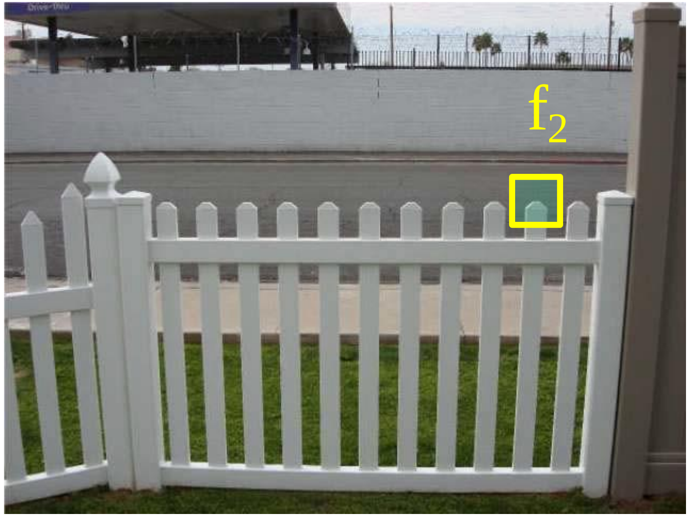
Source: UW CSE vision faculty

Feature distance: SSD

- How to define the similarity between two features f_1, f_2?
  - Simple approach is SSD(f_1, f_2)
  - Sum of square differences (SSD) between entries of the two descriptors
  - Doesn't provide a way to discard ambiguous (bad) matches



I_1

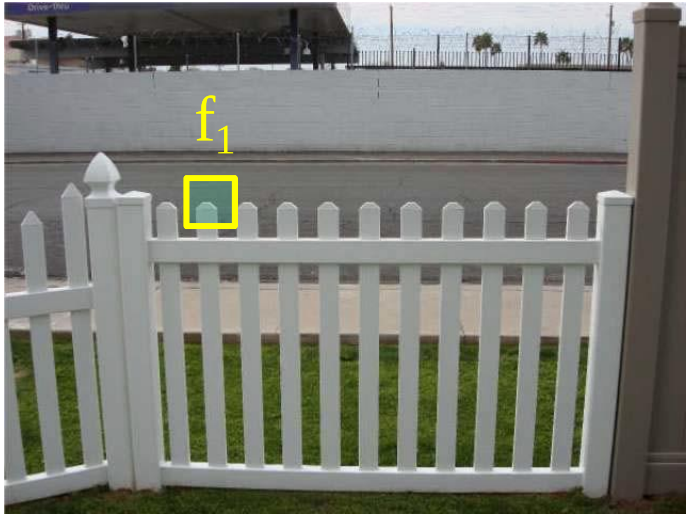


I_2

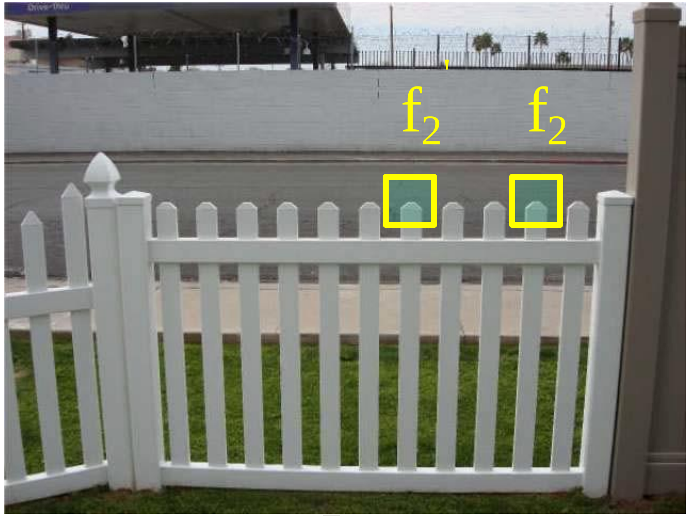
Source: UW CSE vision faculty

Feature distance: Ratio of SSDs

- How to define the 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
    - f_2' is 2nd best SSD match to f_1 in I_2
    - An ambiguous/bad match will have ratio close to 1
    - Look for unique matches which have low ratio



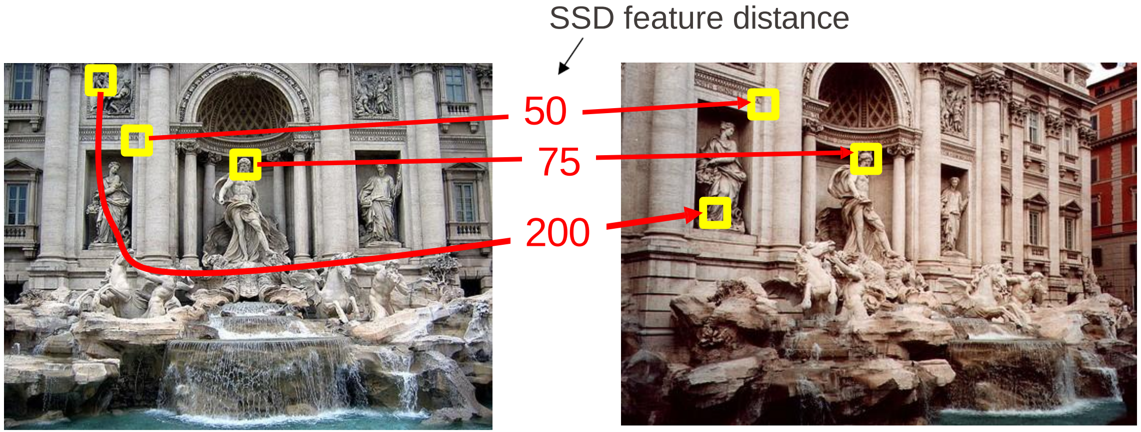
I_1



I_2

Source: UW CSE vision faculty

Image matching



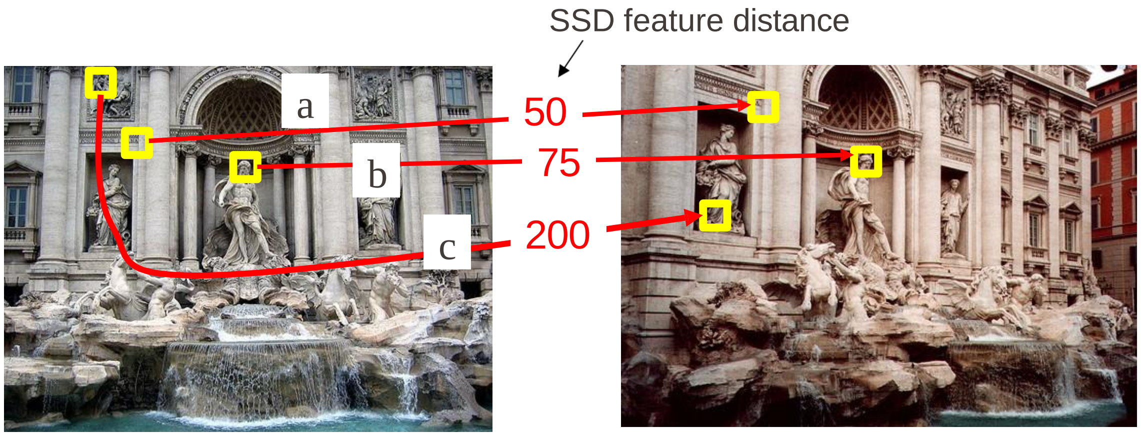
Suppose we use SSD
Small values are possible matches but how small?

Decision rule: Accept match if SSD < T where T is a threshold

What is the effect of choosing a particular T?

Source: UW CSE vision faculty

Effect of threshold T



Decision rule: Accept match if SSD < T

Example: Large T

T = 250 → a, b, c are all accepted as matches

a and b are true matches (“true positives”)

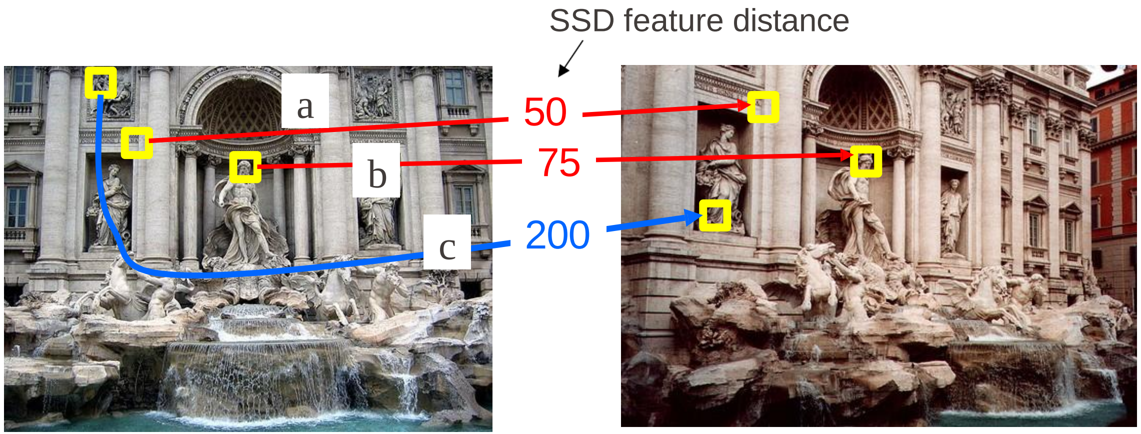
– they are actually matches

c is a false match (“false positive”)

– actually not a match

Source: UW CSE vision faculty

Effect of threshold T



Decision rule: Accept match if SSD < T

Example: Smaller T

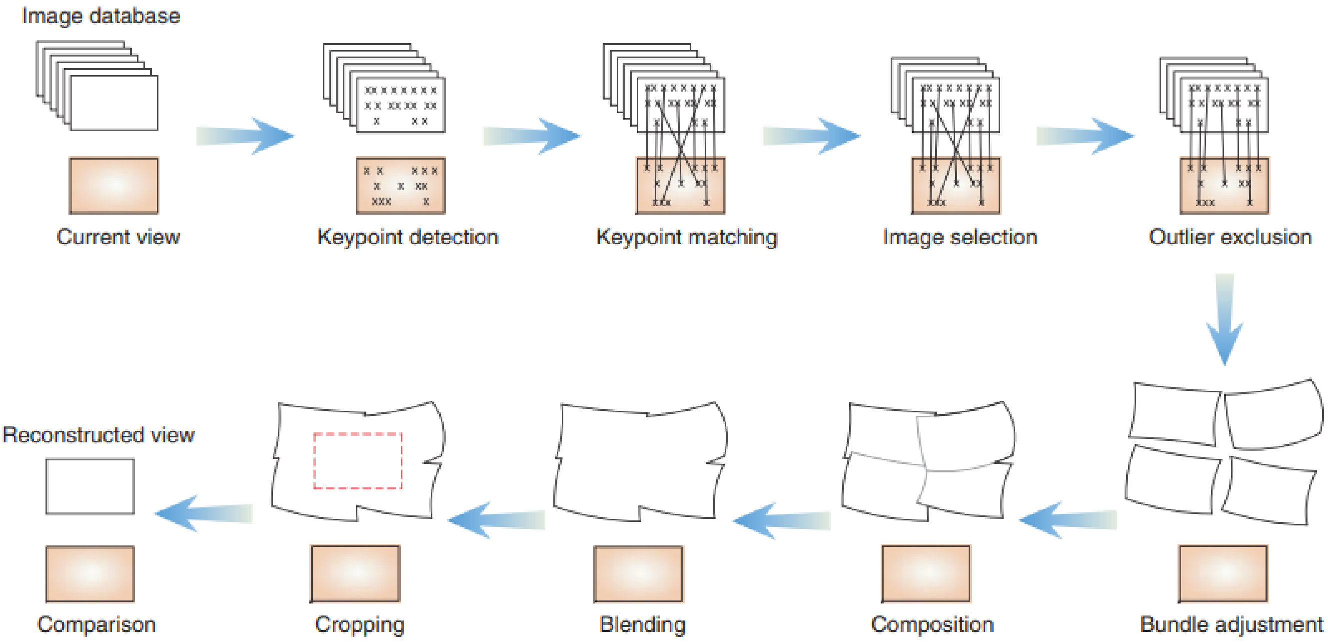
T = 100 → only a and b are accepted as matches a and b are true matches ("true positives")

c is no longer a "false positive" (it is a "true negative")

Source: UW CSE vision faculty

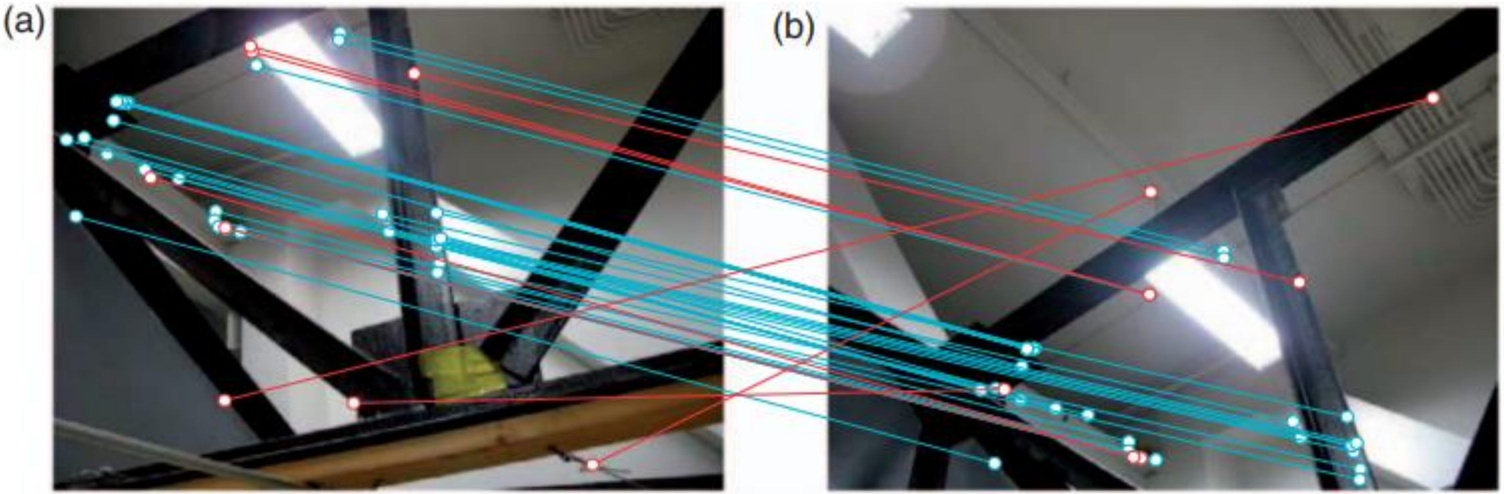
Example applications of feature detection + matching

Multi-image stitching and scene reconstruction for evaluating defect evolution in structures



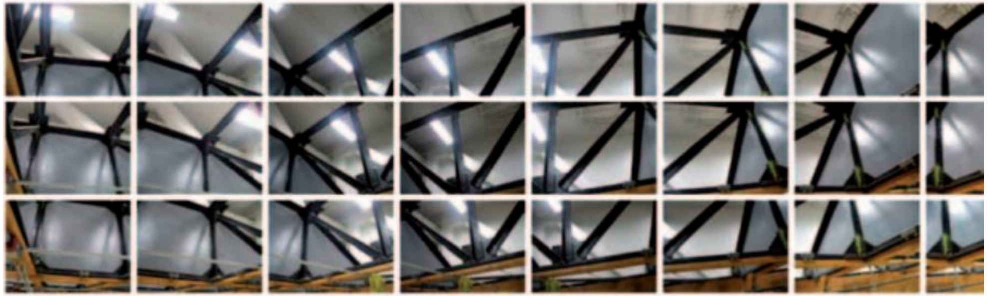
Jahanshahi, M. R., Masri, S. F., & Sukhatme, G. S. (2011). Multi-image stitching and scene reconstruction for evaluating defect evolution in structures. *Structural Health Monitoring*, *10*(6), 643-657.

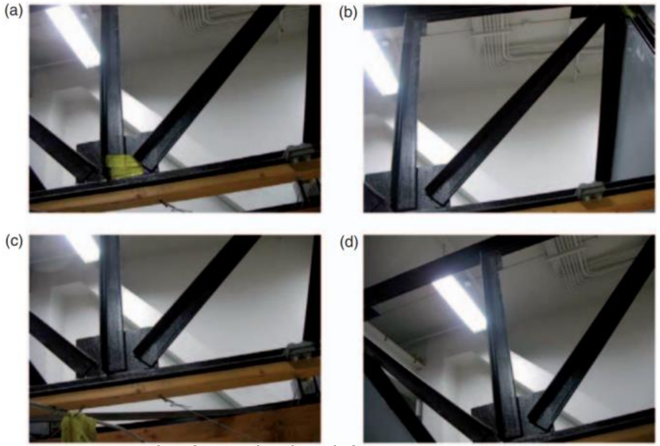
Matching SIFT keypoints in two overlapping images



Jahanshahi, M. R., Masri, S. F., & Sukhatme, G. S. (2011). Multi-image stitching and scene reconstruction for evaluating defect evolution in structures. *Structural Health Monitoring*, *10*(6), 643-657.

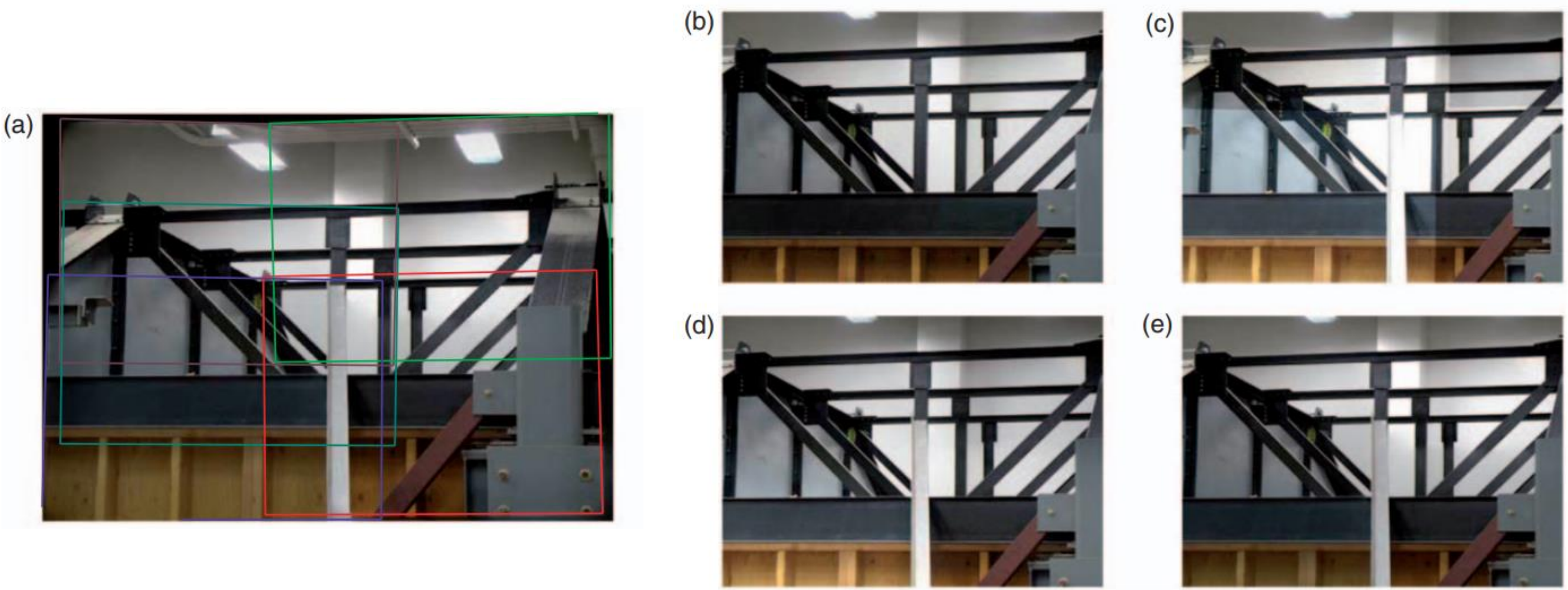
The reconstructed scene and the contribution of the selected images





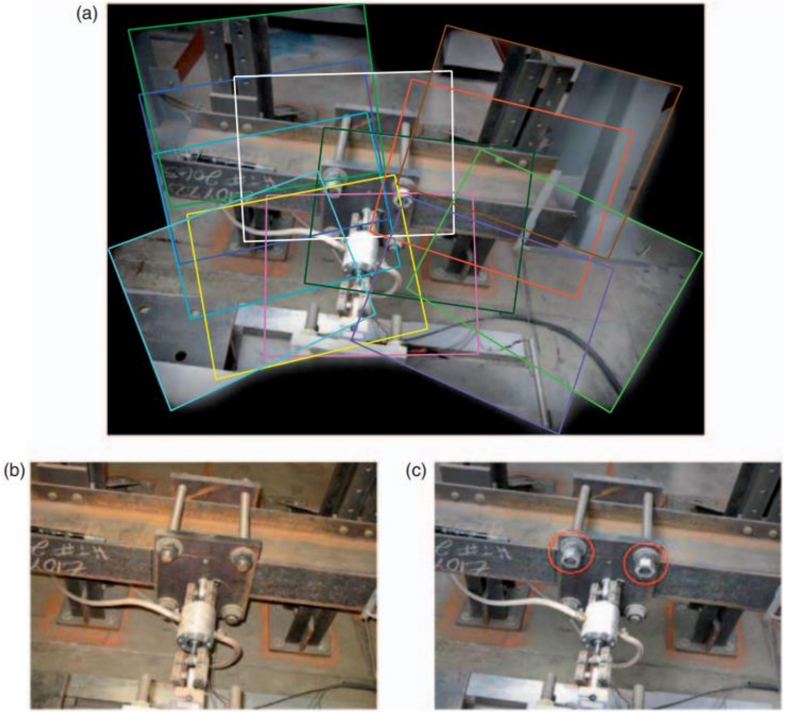
Jahanshahi, M. R., Masri, S. F., & Sukhatme, G. S. (2011). Multi-image stitching and scene reconstruction for evaluating defect evolution in structures. *Structural Health Monitoring*, *10*(6), 643-657.

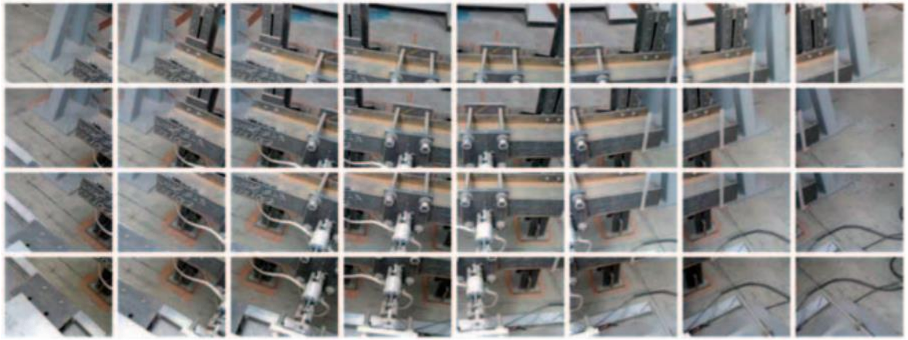
The reconstructed scene and the contribution of the selected images



Jahanshahi, M. R., Masri, S. F., & Sukhatme, G. S. (2011). Multi-image stitching and scene reconstruction for evaluating defect evolution in structures. *Structural Health Monitoring*, *10*(6), 643-657.

The reconstructed scene and the contribution of the selected images





Jahanshahi, M. R., Masri, S. F., & Sukhatme, G. S. (2011). Multi-image stitching and scene reconstruction for evaluating defect evolution in structures. *Structural Health Monitoring*, *10*(6), 643-657.

Recap: Convolutional neural networks

General intuition behind unsing convolutional filters

- Image filters can enhance image attributes
- Convolutional neural networks are similar to conventional image filtering
- Filter kernels are learnt

Input image

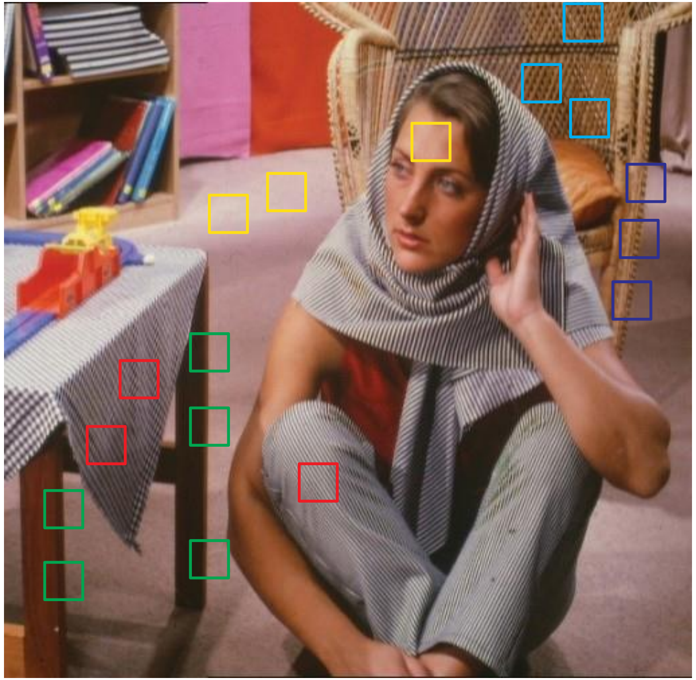
Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

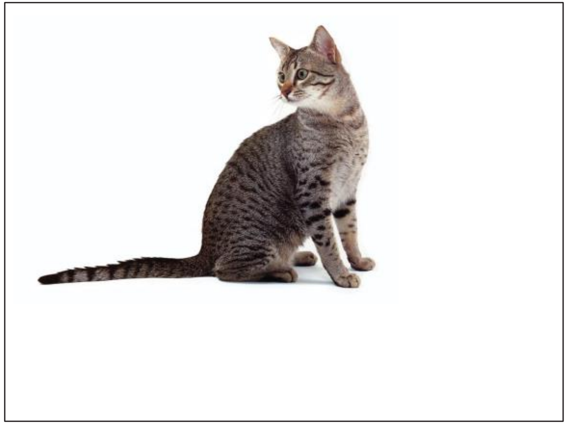
Source: UIO, 2017

Stationarity and Self-similarity



Data is self-similar across the domain

Translation invariance (image classification tasks)



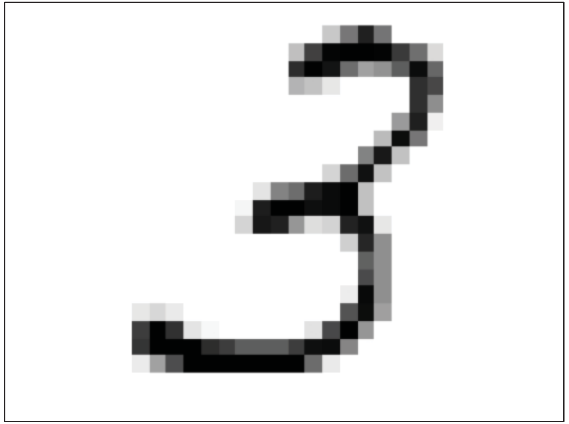
$$f(\mathcal{T}_{\mathbf{v}}x) = f(x) \quad \forall x, \mathbf{v}$$

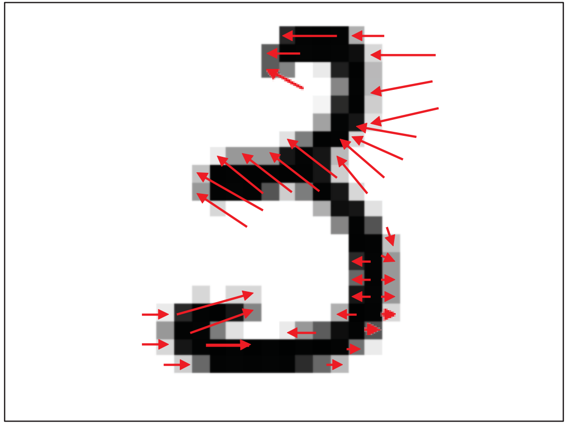
where

- image is modeled as a function $x \in L^2([0,1]^2)$
- $\mathcal{T}_{\mathbf{v}}x(\mathbf{u}) = x(\mathbf{u} - \mathbf{v})$ is a translation operator
- $\mathbf{v} \in [0,1]^2$ is a translation vector
- $f : L^2([0,1]^2) \rightarrow \{1, \ldots, L\}$ is a classification functional

Bruna, Mallat 2012

Deformation invariance (image classification tasks)





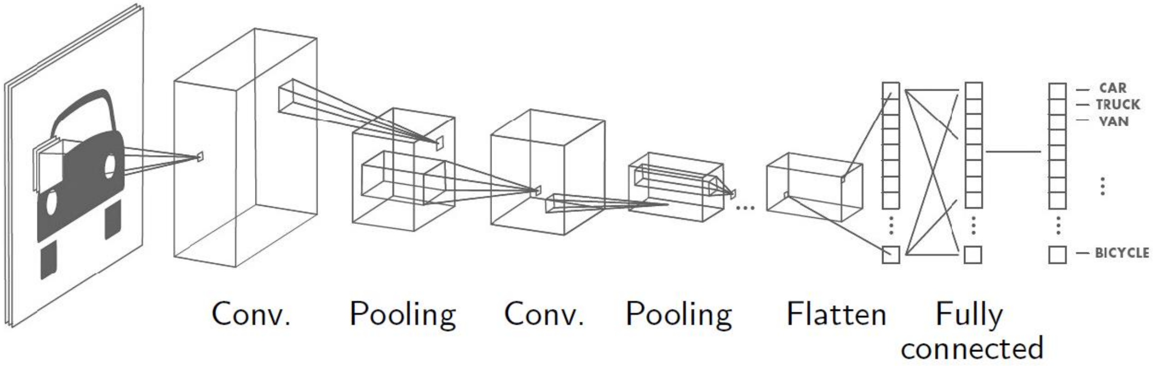
$$|f(\mathcal{L}_\tau x) - f(x)| \approx \|\nabla \tau\| \quad \forall f, \tau$$

where

- image is modeled as a function $x \in L^2([0,1]^2)$
- $\mathcal{L}_\tau x(\mathbf{u}) = x(\mathbf{u} - \tau(\mathbf{u}))$ is a warping operator
- $\tau : [0,1]^2 \to [0,1]^2$ is a smooth deformation field
- $f : L^2([0,1]^2) \to \{1, \ldots, L\}$ is a classification functional

Bruna, Mallat 2012

Convolutional Neural Networks (CNN)



Conv. layer $$x_{\ell'}^{(l+1)}(\mathbf{u}) = \xi\left(\sum_{\ell=1}^{d^{(l)}} (w_{\ell'\ell}^{(l+1)} \star x_{\ell}^{(l)})(\mathbf{u})\right)$$

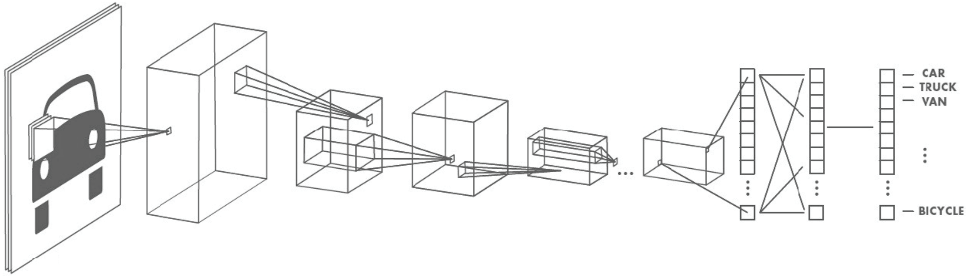
Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters filters $W^{(1)}, \ldots, W^{(L)}$

Pooling $x_{\ell}^{(l+1)}(\mathbf{u}) = \|x_{\ell}^{(l)}(\mathbf{u}') : \mathbf{u}' \in \mathcal{N}(\mathbf{u})\|_p$ $p = 1, 2,$ or ∞

LeCun et al. 1989 (Image: Debarko De)

Key properties of CNNs

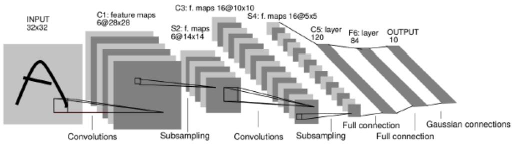


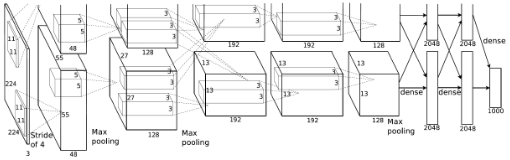
- ☺ Convolutional filters **(Translation invariance+Self-similarity)**
- ☺ Multiple layers **(Compositionality)**
- ☺ Filters localized in space **(Locality)**
- ☺ $\mathcal{O}(1)$ parameters per filter (independent of input image size n)
- ☺ $\mathcal{O}(n)$ complexity per layer (filtering done in the spatial domain)
- ☺ $\mathcal{O}(\log n)$ layers in classification tasks

LeCun et al. 1989

Convolutional Neural Networks (historical perspective)

1989	2012





1989

- 3 convolutional + 1 fully connected layer
- 1M parameters
- Trained on MNIST 70K
- CPU-based
- $\tanh$ non-linearity

2012

- 5 convolutional + 3 fully connected layers
- 60M parameters
- Trained on ImageNet 1.5M
- GPU-based
- ReLU, Dropout

LeCun et al. 1998; Krizhevsky, Sutskever, Hinton 2012

Filters

0	0	0	0	0	0	0
0	2	4	9	1	4	0
0	2	1	4	4	6	0
0	1	1	2	9	2	0
0	7	3	5	1	3	0
0	2	3	4	8	5	0
0	0	0	0	0	0	0

Image

X

1	2	3
-4	7	4
2	-5	1

Filter / Kernel

=

21	59	37	-19	2
30	51	66	20	43
-14	31	49	101	-19
59	15	53	-2	21
49	57	64	76	10

Feature

$$O = \frac{n - f + 2p}{s} + 1$$

Where O is the output height/length, n is input height / length, f is filter size, p is the padding, and s is the stride

Pooling

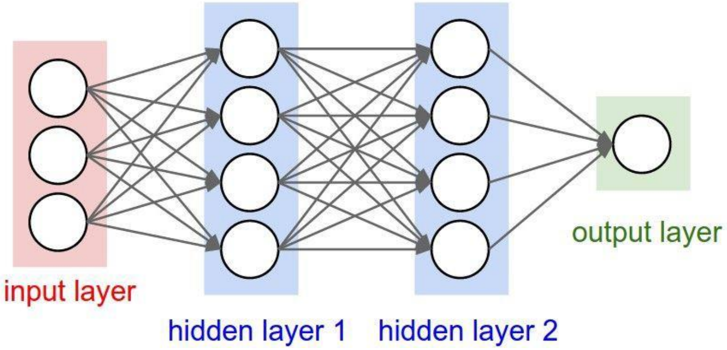
6	8	6	3	1	0
9	13	10	5	2	0
9	14	11	6	3	0
9	13	11	6	2	0
8	13	10	5	3	0
6	7	5	3	1	0

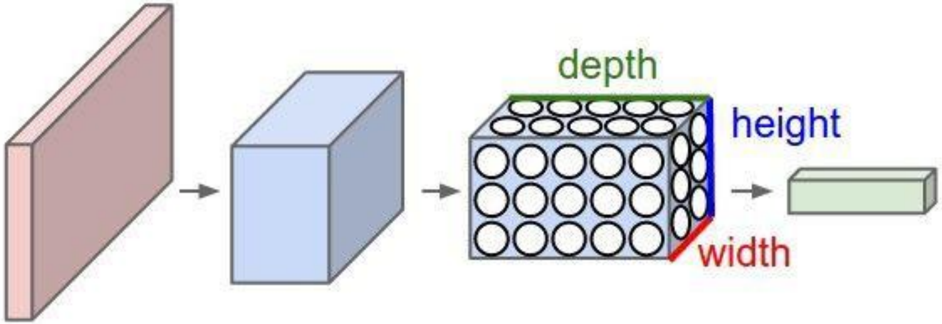
Feature map

13	10	2
14	11	3
13	10	3

Max pooling

In convolutional networks, layers are 3D…

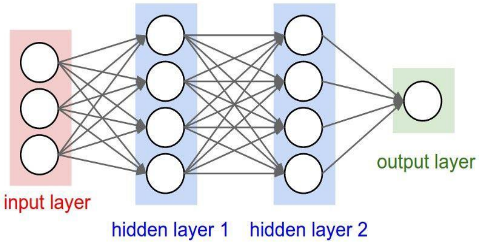


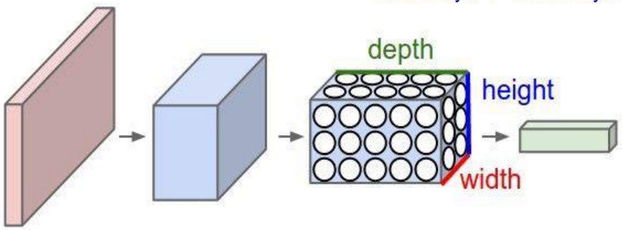


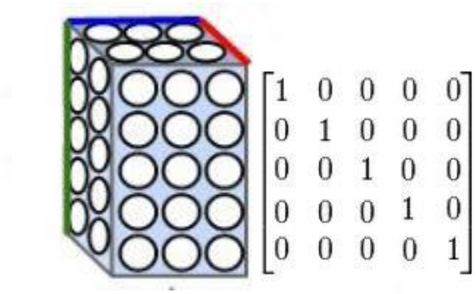
Source: UIO, 2017

… kernels are 4D

- If we combine all the filters we get a 4D tensor
- The operation can be viewed as:
  - a matrix multiplication for each spatial position
  - a sum over spatial dimensions
- This is a useful representation as many deep learning frameworks present it in this way

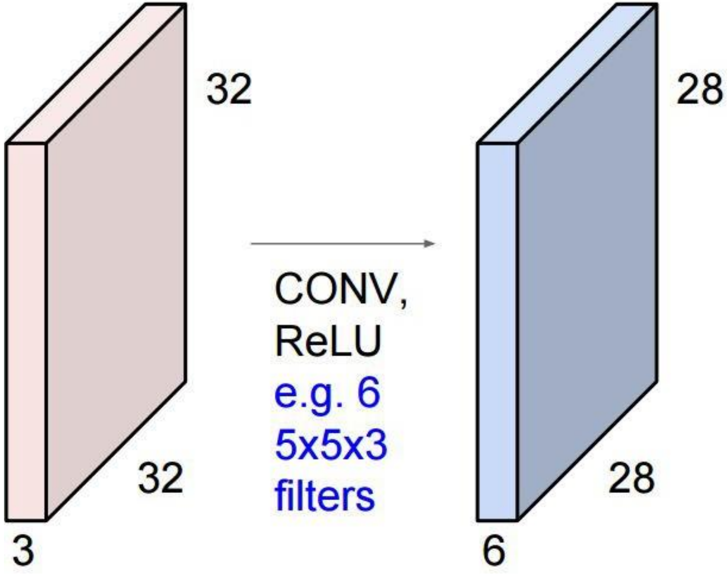






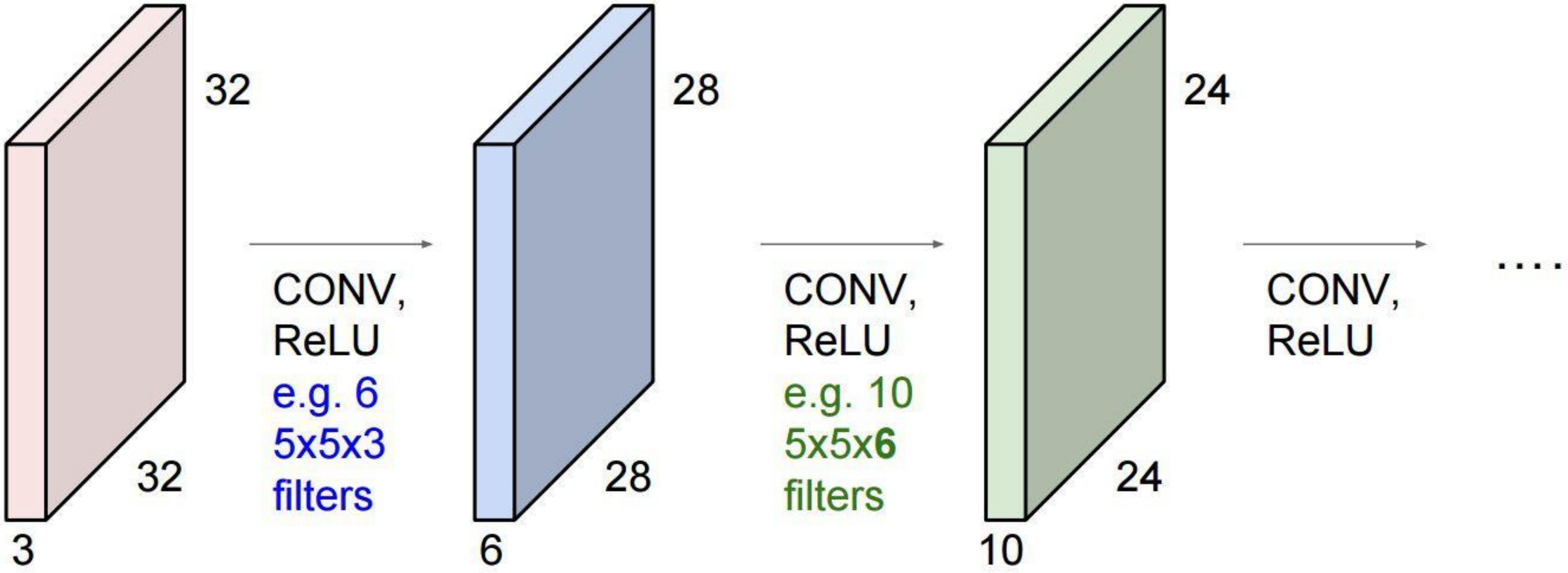
Source: UIO, 2017

Convolutional neural network consist of multiple layers



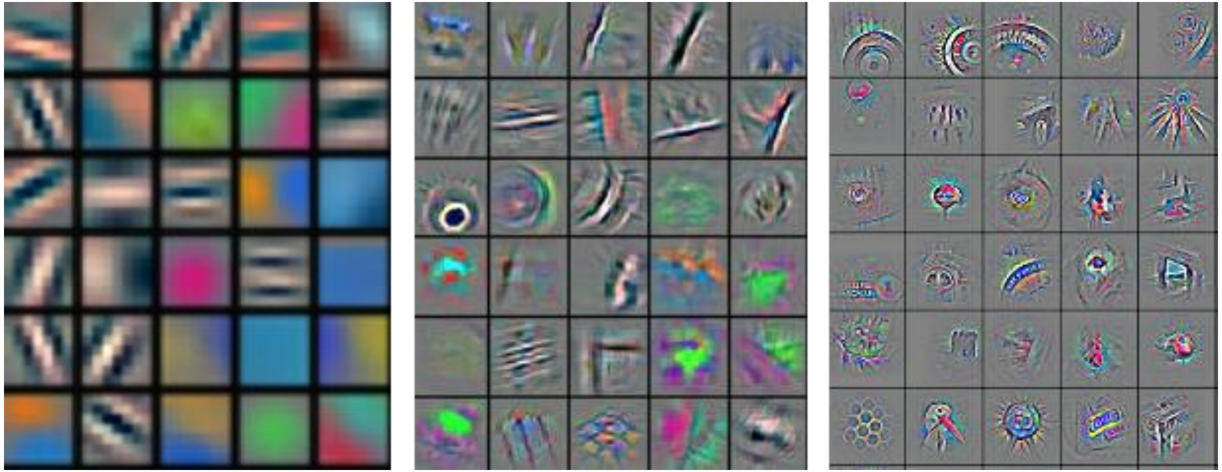
Source: UIO, 2017

Convolutional neural network consist of multiple layers



Source: UIO, 2017

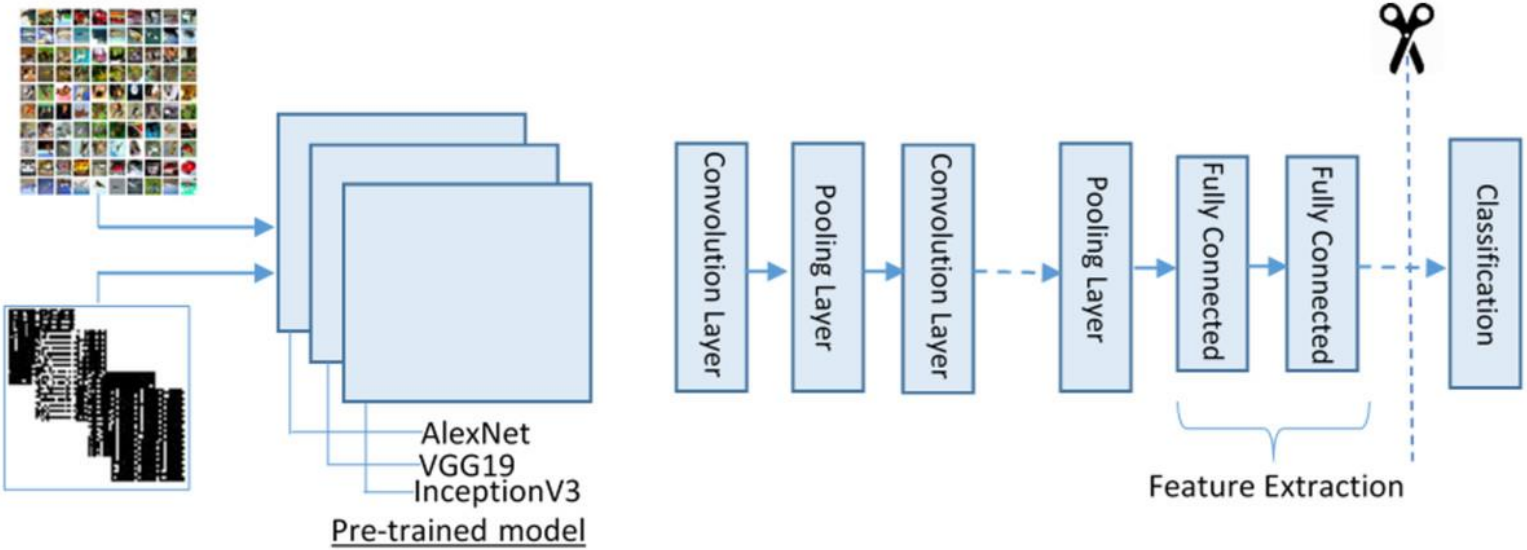
Hierarchy and Compositionality



Typical features learned by a CNN becoming increasingly complex at deeper layers

Zeiler, Fergus 2013

Making use of pre-trained models



Transfer Learning with CNNs