Computer Vision

UNIT II

Multi-Level Vision (Low, Mid, High):

LOW: Feature Detection and Matching, Edge Detection, (Chap 7)

MID: Image Alignment(Chap 8)

HIGH: Instance Recognition, Object Detection, Image Classification, Semantic Segmentation, Video Understanding, Vision and Language.(Chap 6)

Today

- Image matching
- Features
- Features detection
- Feature descriptors
- Features Matching

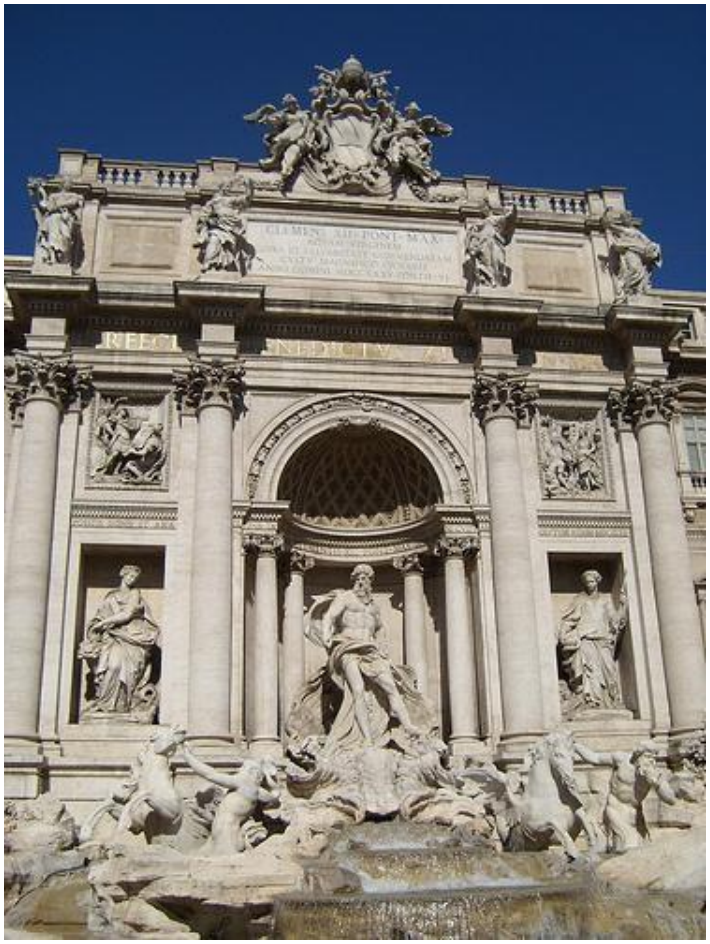
Readings

- Szeliski, Ch 7
- (optional) K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors. In PAMI 27(10):1615-1630
  - http://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/mikolajczyk_

Image matching

Trevi fountain-Rome

by Diva Sian



by swashford

Slides by Steve Seitz, Hayes and Hoiem

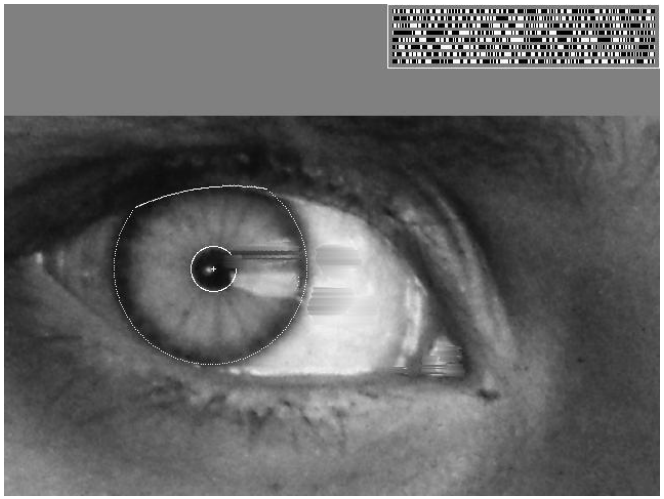
Harder case

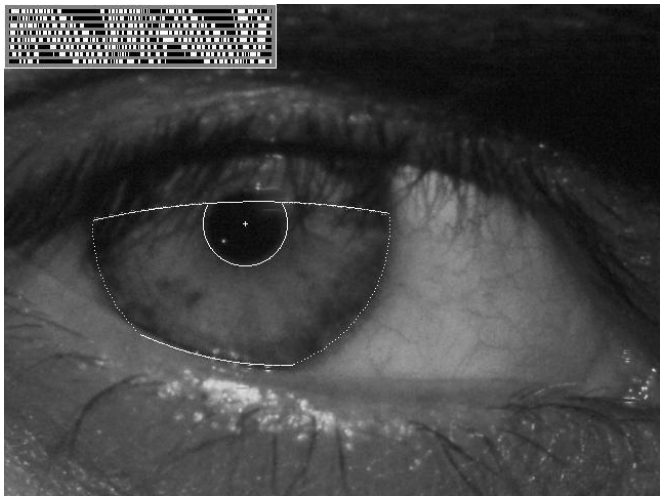
by Diva Sian

by scgbt

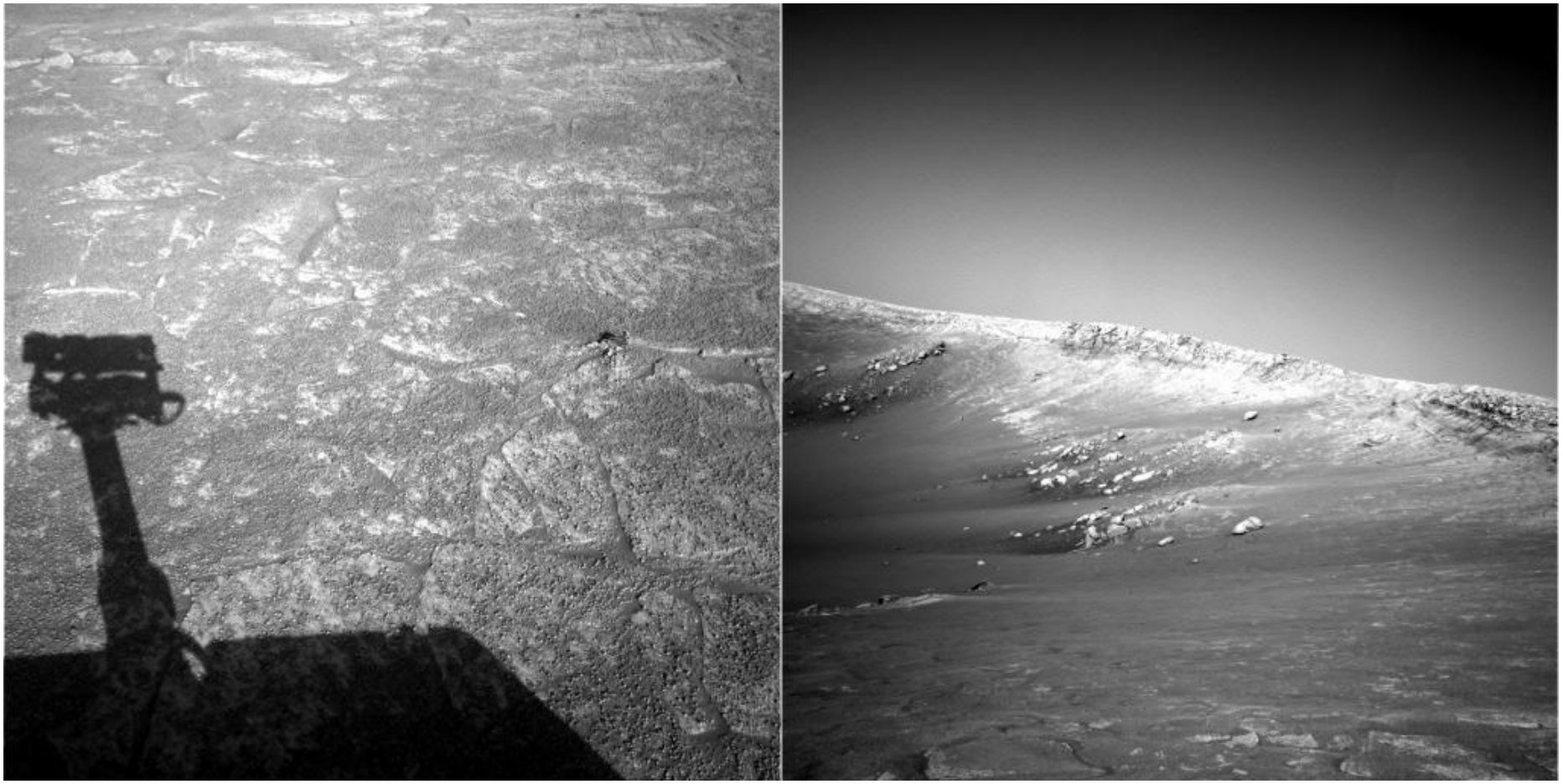
Even harder case

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



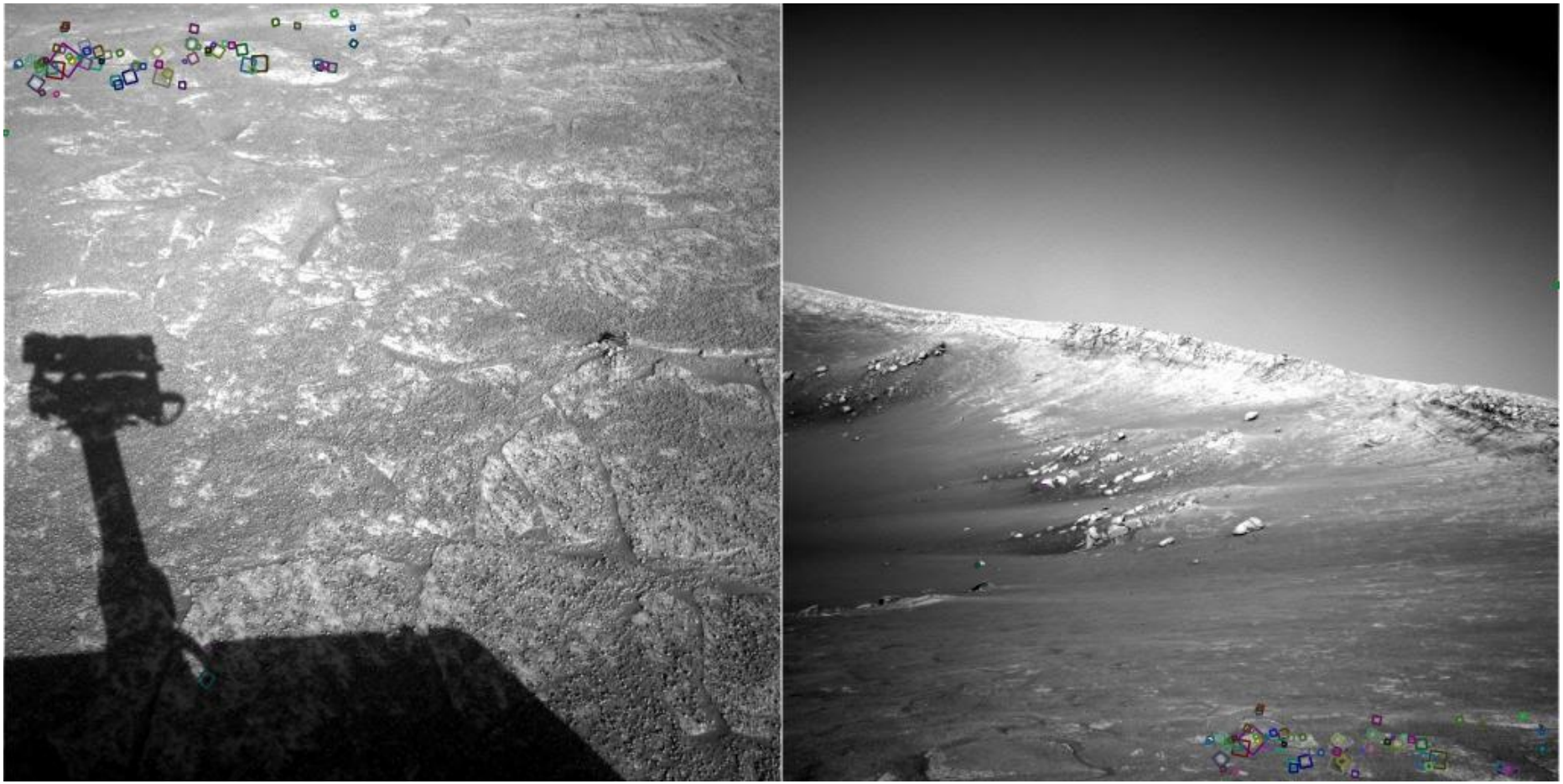


Harder still?



NASA Mars Rover images

Answer below (look for tiny colored squares…)

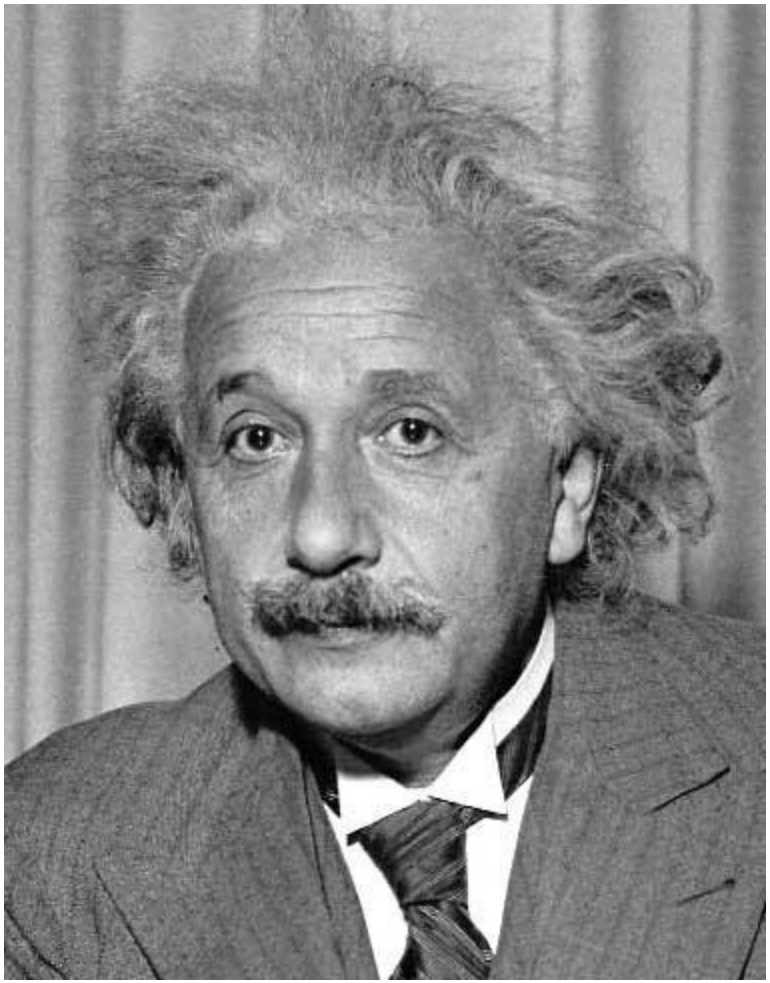


NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Template matching

- Goal: find  in image

- Main challenge: What is a good similarity or distance measure between two patches?
  - Correlation
  - Zero-mean correlation
  - Sum Square Difference
  - Normalized Cross Correlation



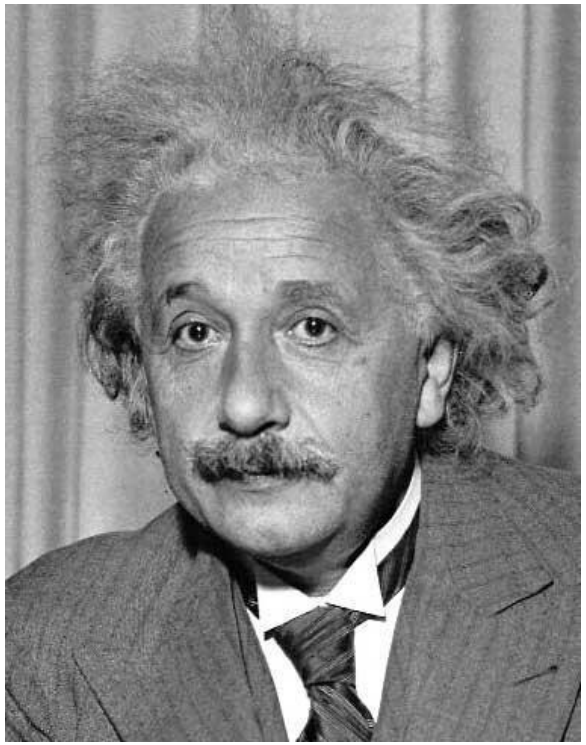
Slides by James Hayes

Matching with filters

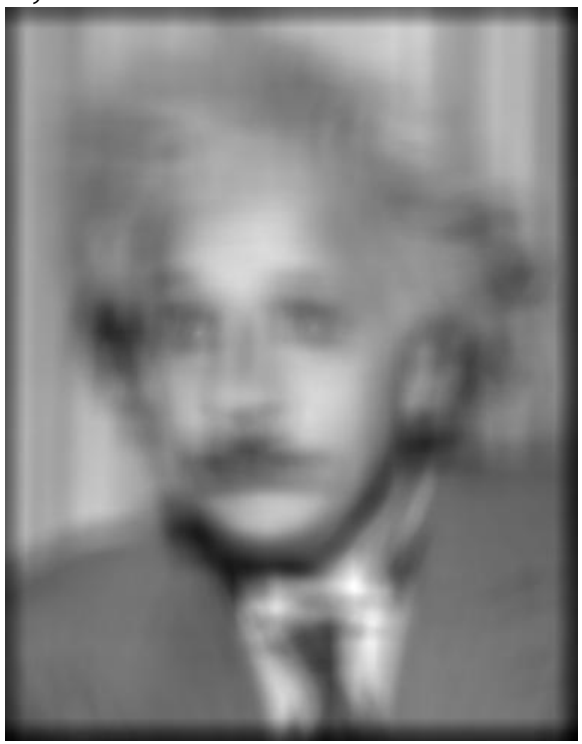
- Goal: find  in image
- Method 0: filter the image with eye patch

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k, n+l]$$

f = image
g = filter



Input



Filtered Image

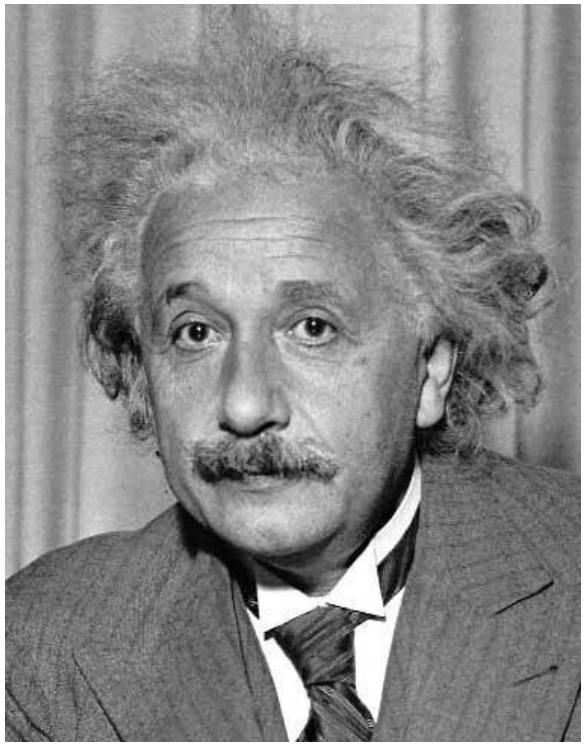
What went wrong?
Problem: response is stronger for higher intensity

Matching with filters

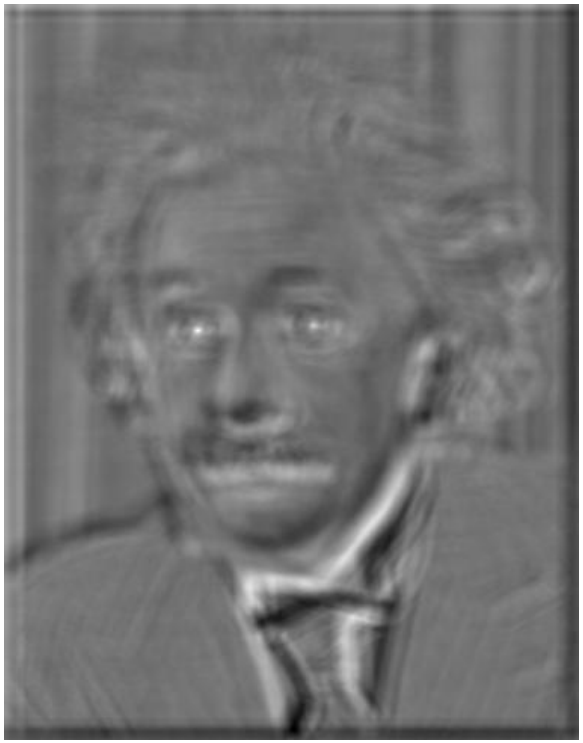
- Goal: find  in image
- Method 1: filter the image with zero-mean eye

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

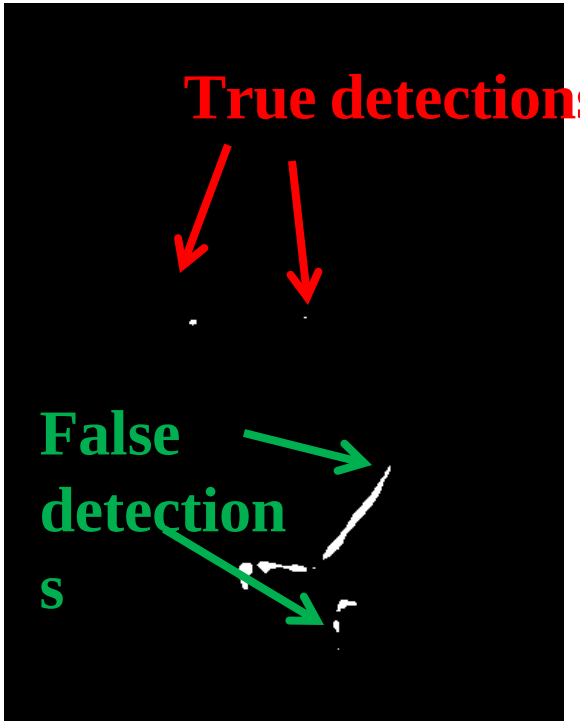
mean of f



Input



Filtered Image (scaled)

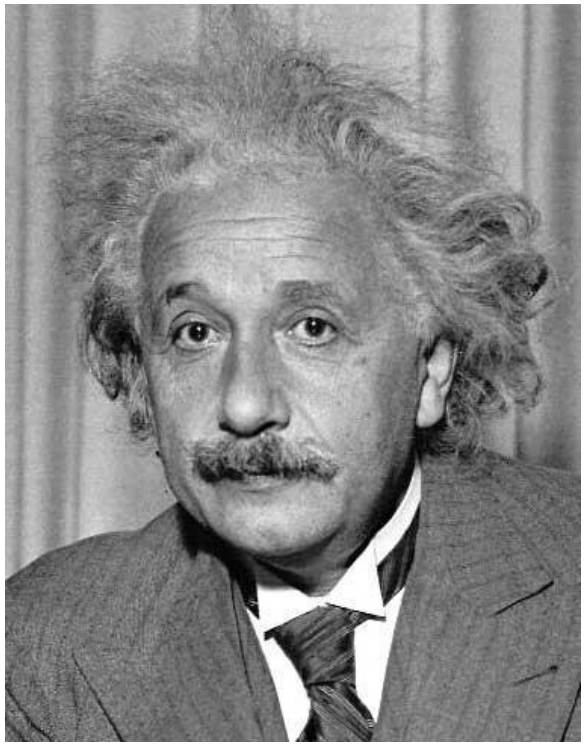


Thresholded Image

Matching with filters

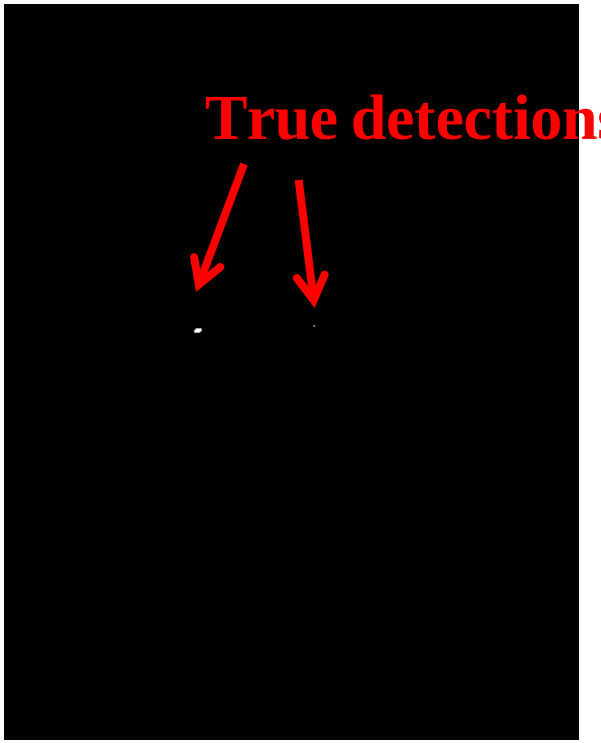
- Goal: find  in image
- Method 2: SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$



Input

1- sqrt(SSD)



Thresholded Image

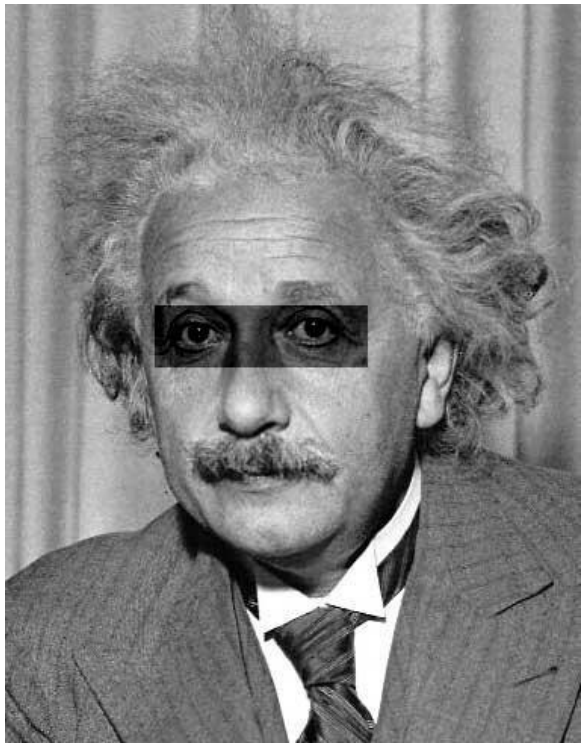
Matching with filters

- Goal: find  in image
- Method 2: SSD

What's the potential downside of SSD?

: SSD sensitive to average intensity

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$



Input

1- sqrt(SSD)

Matching with filters

- Goal: find  in image
- Method 3: Normalized cross-correlation

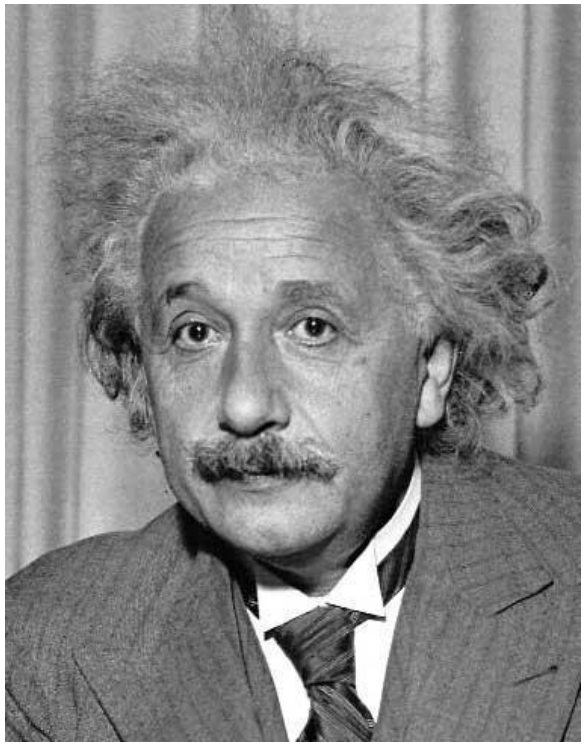
mean template **mean image patch**

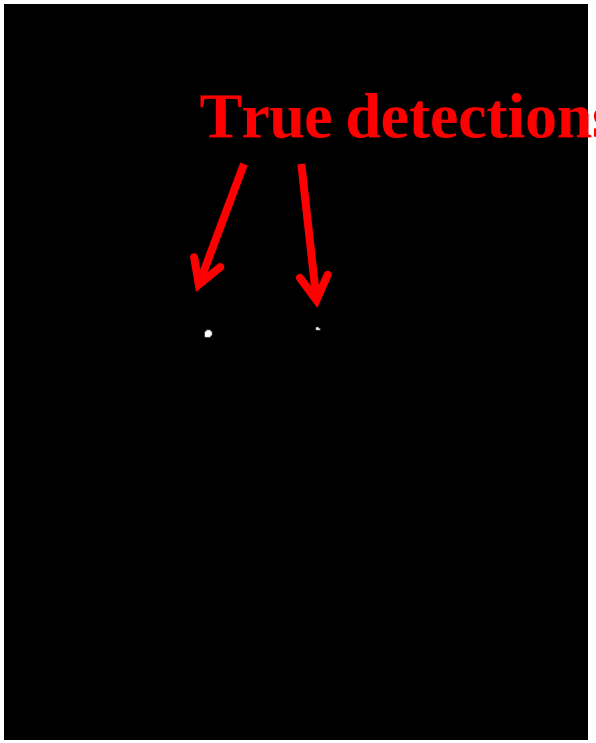
$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m-k,n-l]-\bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2\sum_{k,l}(f[m-k,n-l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

Matlab: `normxcorr2(template, im)`

Matching with filters

- Goal: find  in image
- Method 3: Normalized cross-correlation

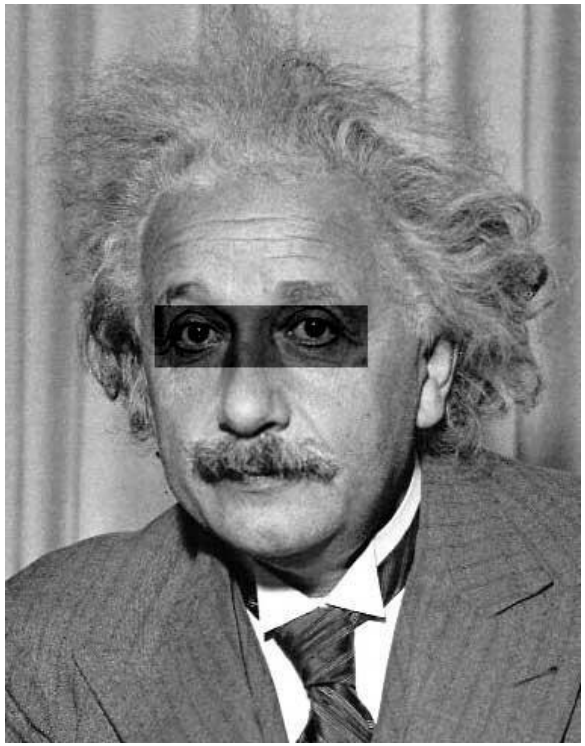


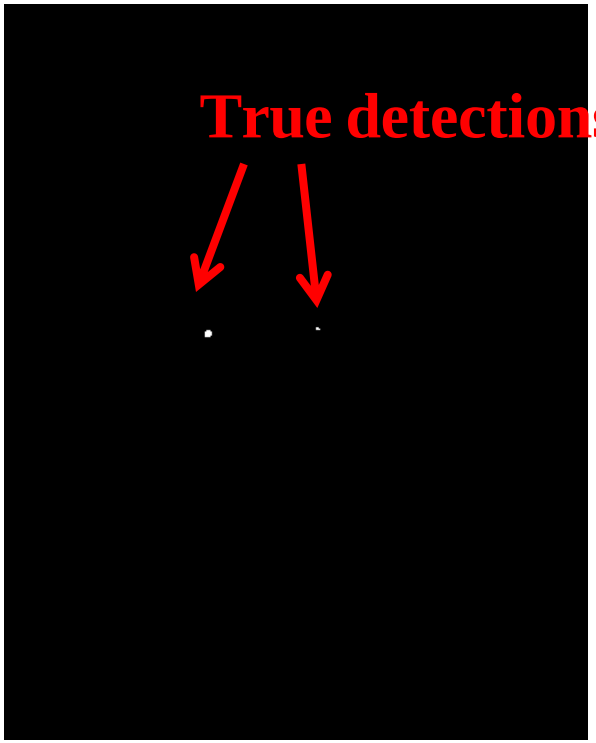


Input **Normalized X-Correlation** **Thresholded Image**

Matching with filters

- Goal: find  in image
- Method 3: Normalized cross-correlation





Input **Normalized X-Correlation** **Thresholded Image**

Q: What is the best method to use?

A: Depends

- SSD: faster, sensitive to overall intensity
- Normalized cross-correlation: slower, invariant to local average intensity and contrast

Q: What if we want to find larger or smaller eyes?

- A: Image Pyramid

Review of Sampling

Gaussian pyramid

512 256 128 64 32 16 8

Source: Forsyth

Template Matching with Image Pyramids

Input: Image, Template

1. Match template at current scale

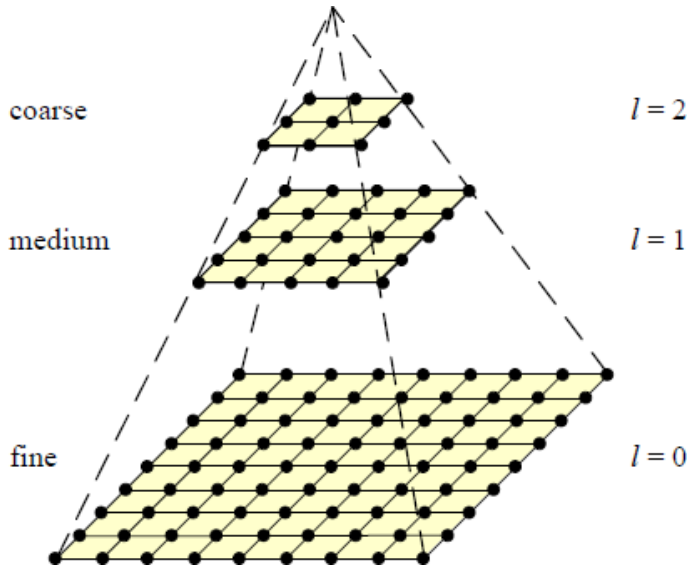
2. Downsample image

3. Repeat 1-2 until image is very small

4. Take responses above some threshold, perhaps with non-maxima suppression

Coarse-to-fine Image Registration

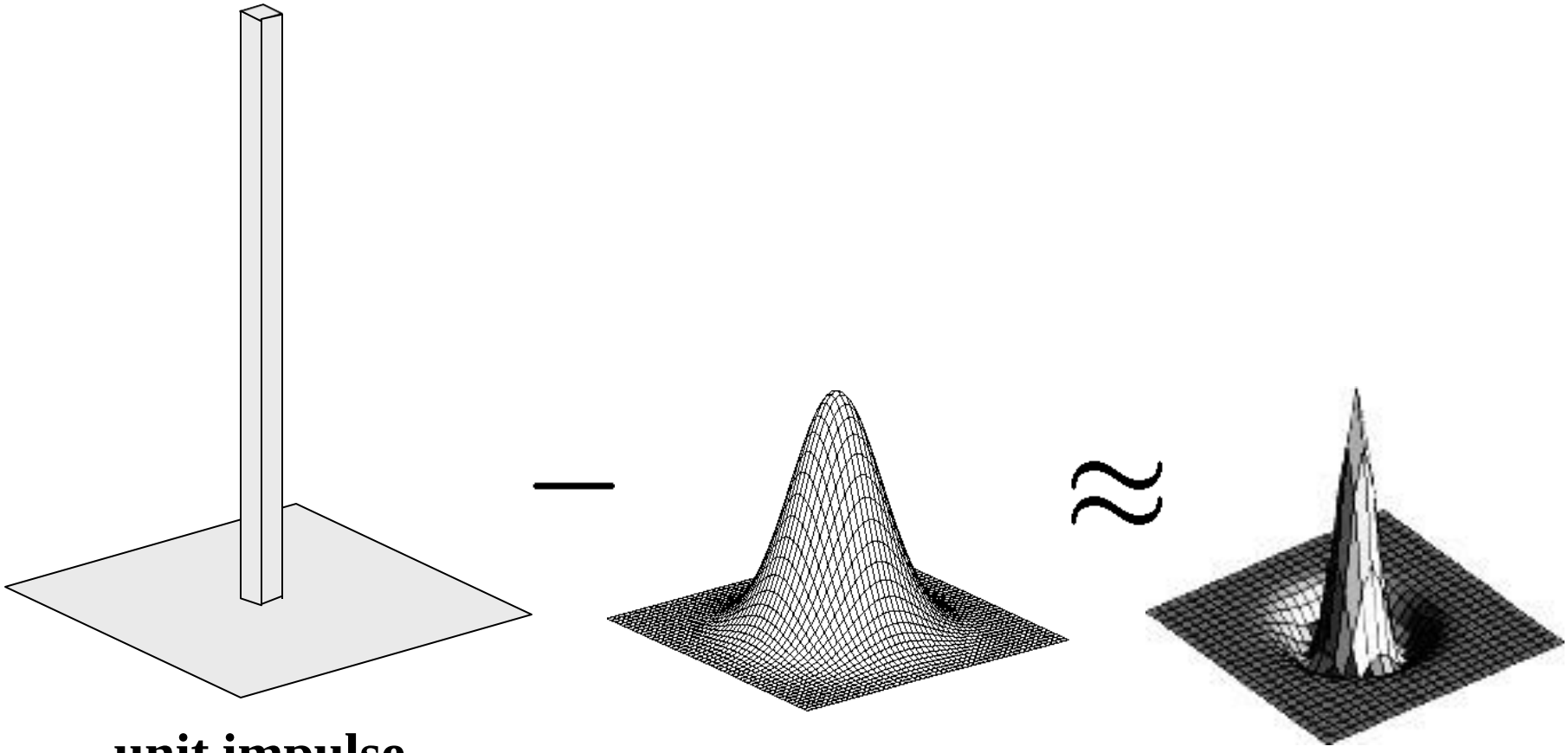
1. Compute Gaussian pyramid
2. Align with coarse pyramid
3. Successively align with finer pyramids
   - Search smaller range



Why is this faster?

Are we guaranteed to get the same result?

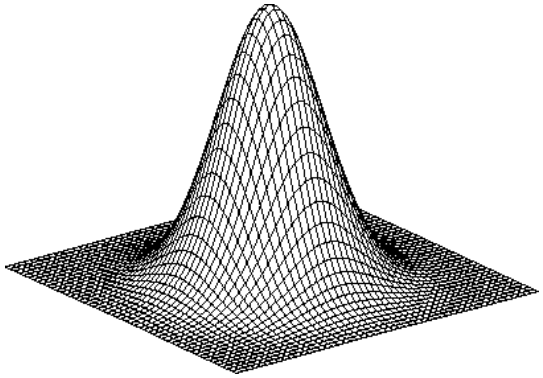
Laplacian filter



unit impulse **Gaussian** **Laplacian of Gaussian**

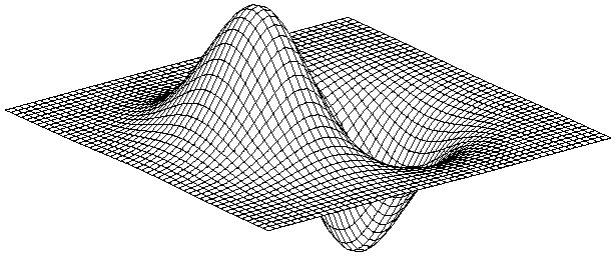
Source: Lazebnik

2D edge detection filters



Gaussian

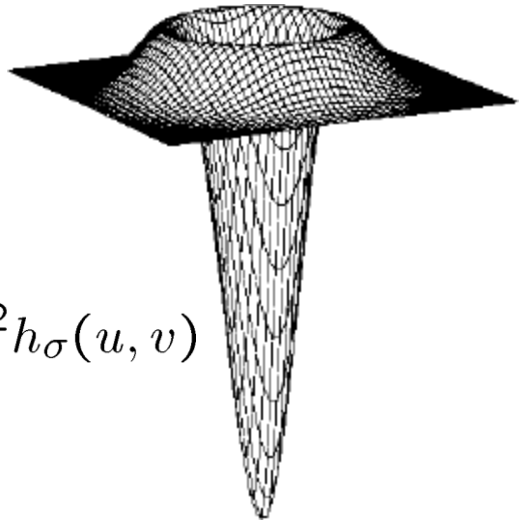
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian

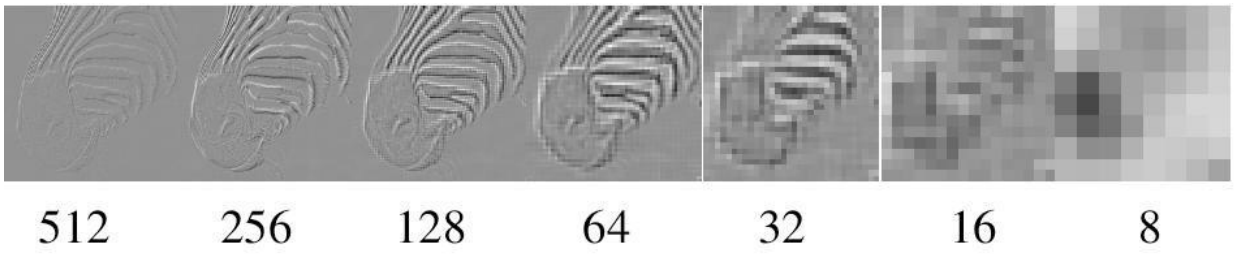


$$\nabla^2 h_\sigma(u,v)$$

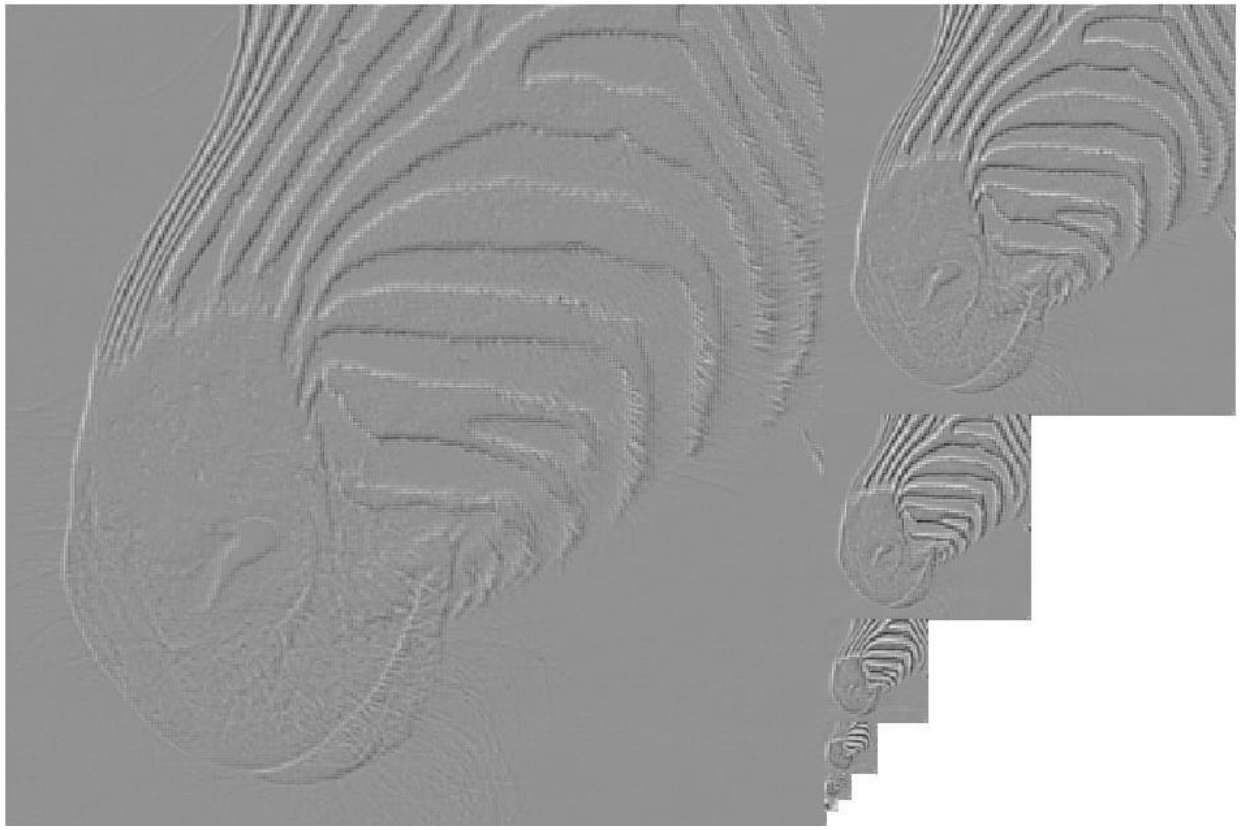
∇^2 is the Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Laplacian pyramid

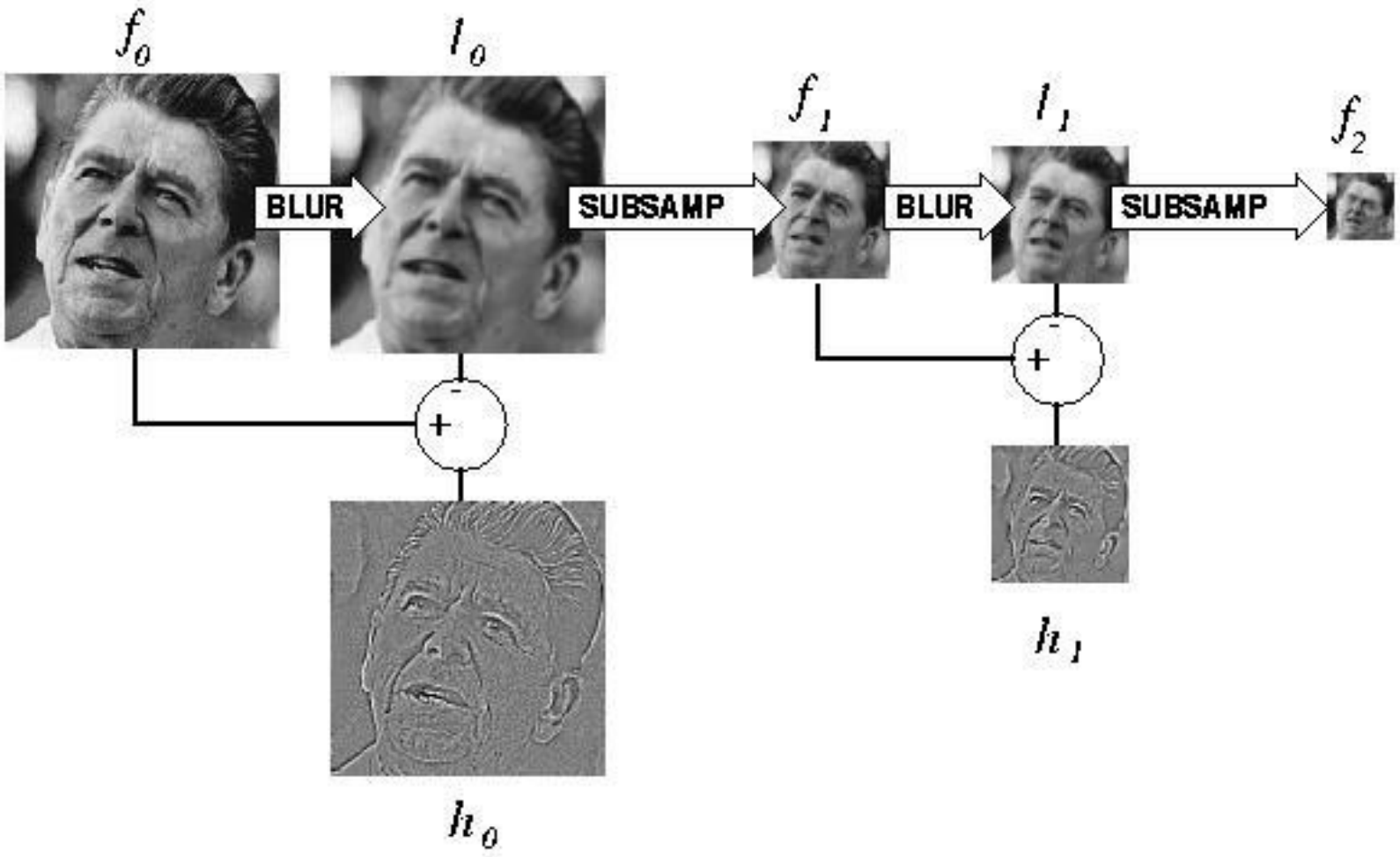


512 256 128 64 32 16 8



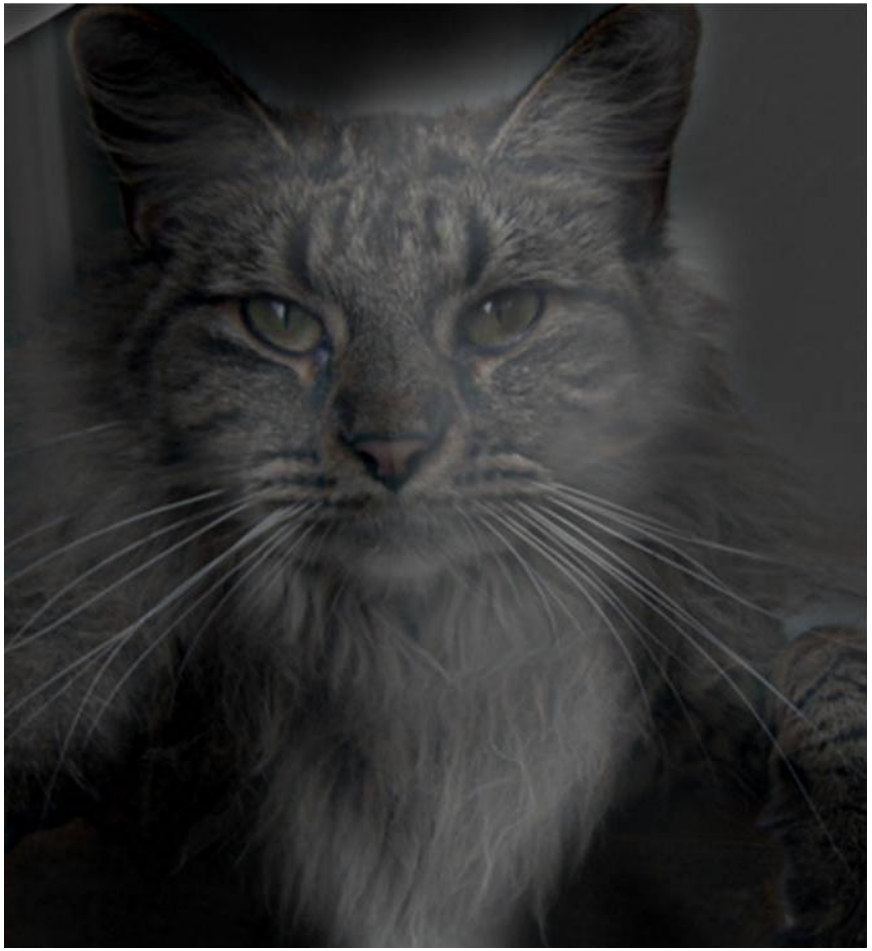
Source: Forsyth

Computing Gaussian/Laplacian Pyramid



http://sepwww.stanford.edu/~morgan/texturematch/paper_html/node3.html

Hybrid Image



Hybrid Image in Laplacian Pyramid

High frequency → Low frequency

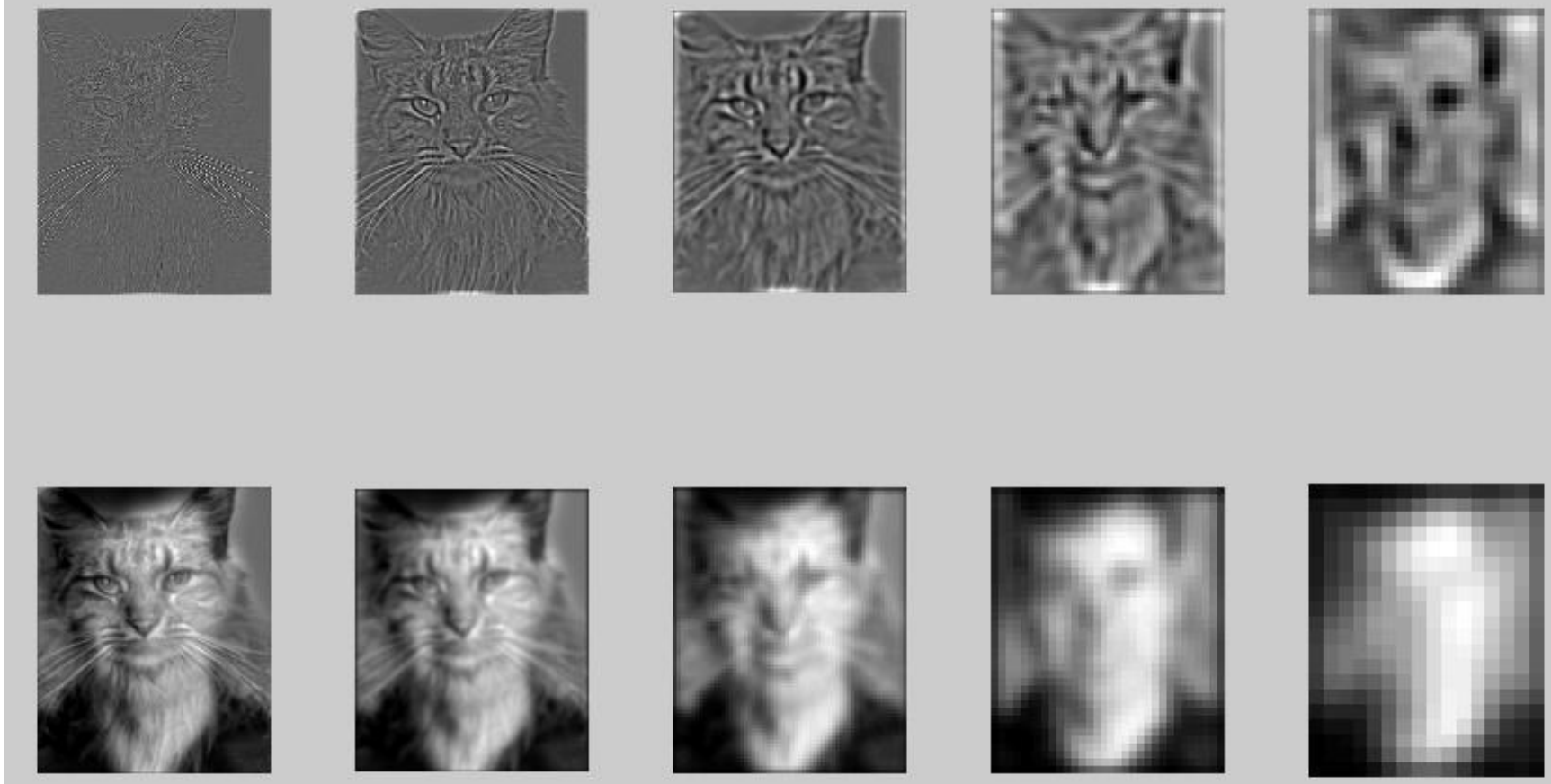
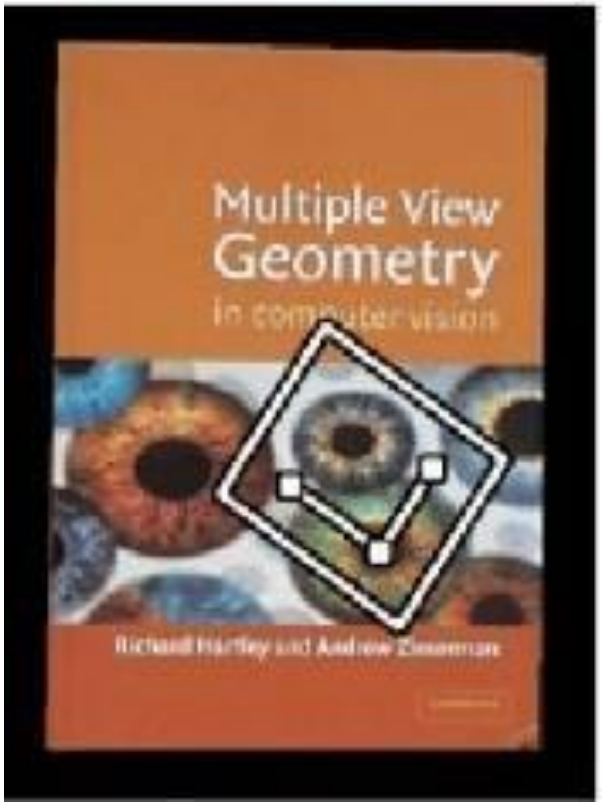
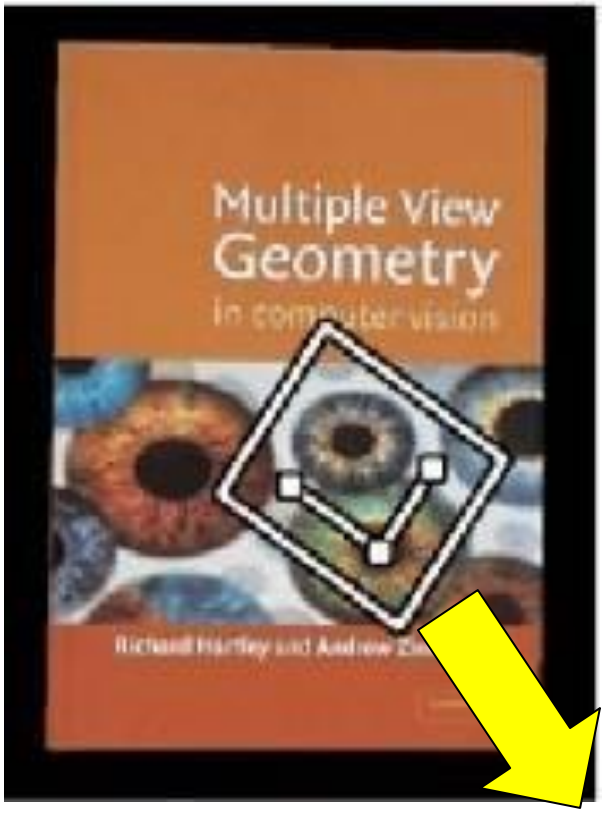


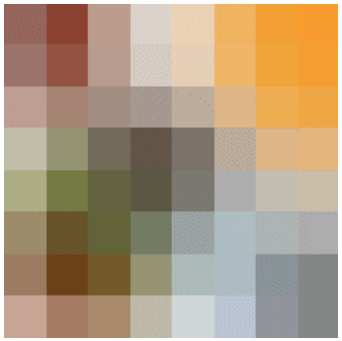
Image Matching

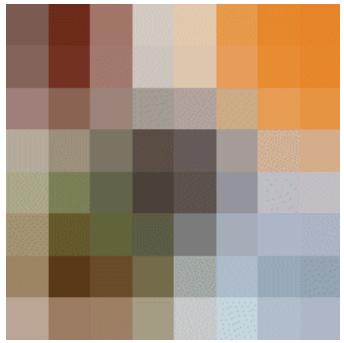


Slides by Steve Seitz

Image Matching



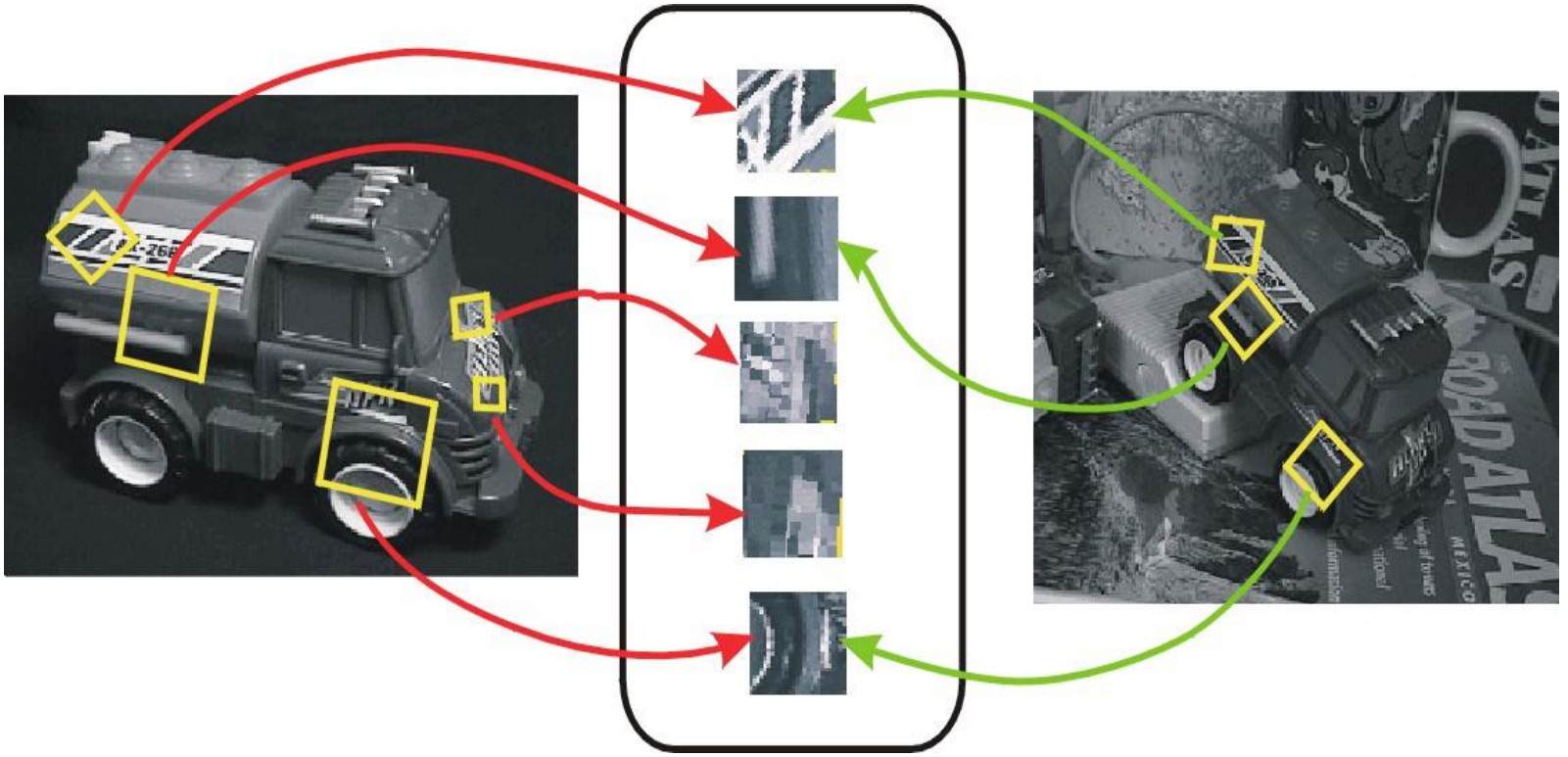




Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …



Feature Descriptors

Invariant Local Feature Methods

Method	Geometric Invariance	Photometric Invariance
Harris Corner Detector	Translation	✗
SIFT (Scale-Invariant Feature Transform)	Translation, Rotation, Scale	Partial brightness invariance
SURF	Translation, Rotation, Scale	Partial
ORB	Translation, Rotation, Scale	Partial
BRISK	Translation, Rotation, Scale	Partial

Advantages of 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

More motivation…

Feature points are used for:

- Image alignment
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

What makes a good feature?

Want uniqueness

Look for image regions that are unusual

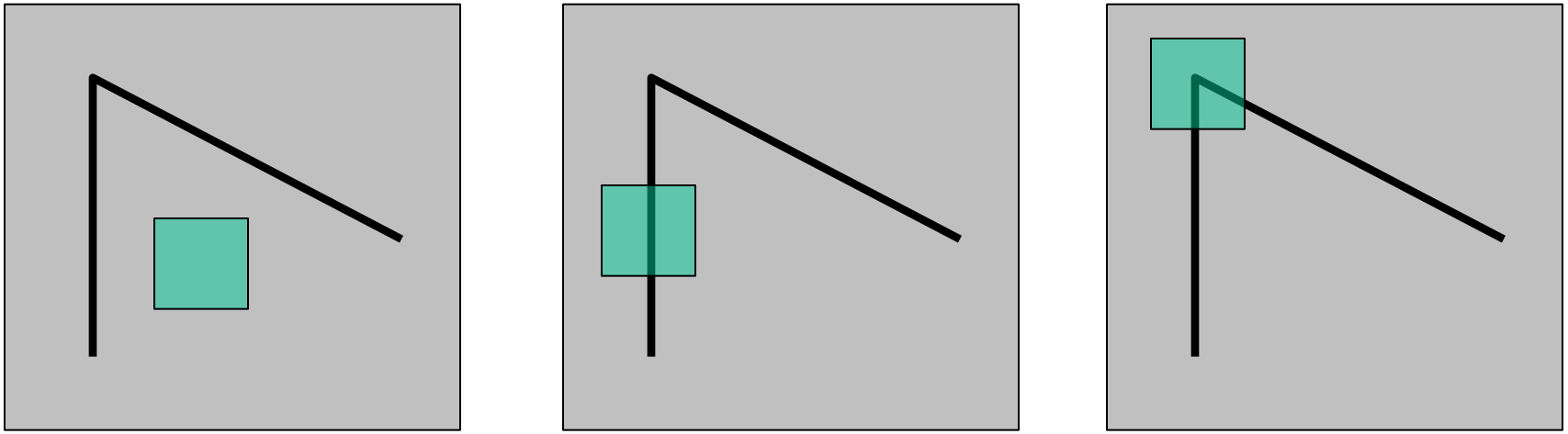
- Lead to unambiguous matches in other images

How to define “unusual”?

Local measures of uniqueness

Suppose we only consider a small window of pixels

- What defines whether a feature is a good or bad candidate?

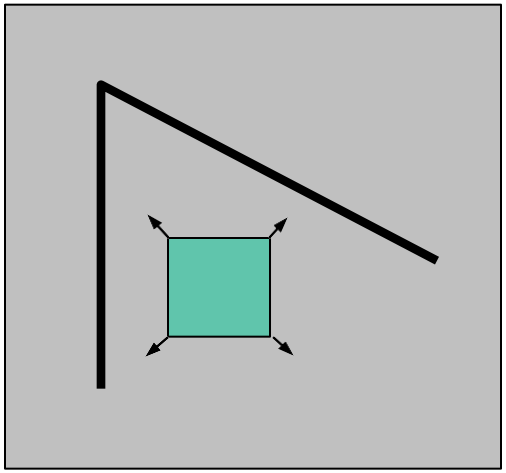


Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

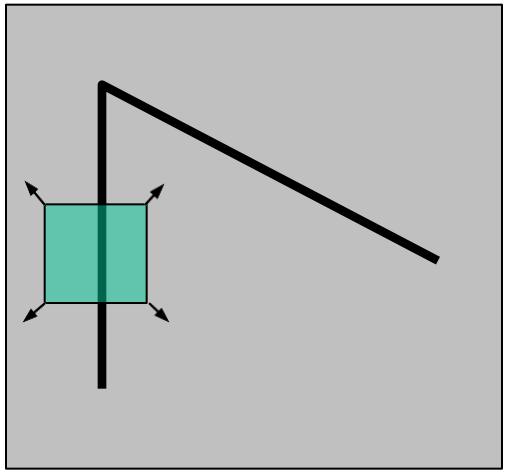
Feature detection

Local measure of feature uniqueness

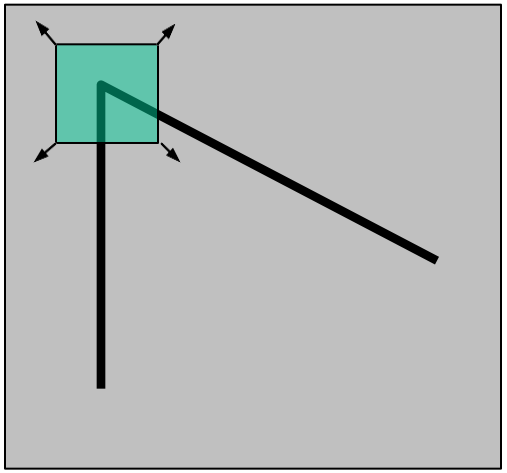
- How does the window change when you shift it?
- Shifting the window in *any direction* causes a *big change*



"flat" region:
no change in all directions



"edge":
no change along the edge direction



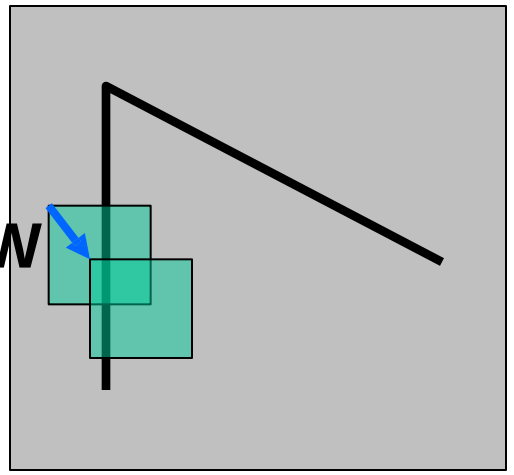
"corner":
significant change in all directions

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

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 by summing up the squared differences (SSD)
- this defines an SSD “error” of *E(u,v)*:



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

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

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

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

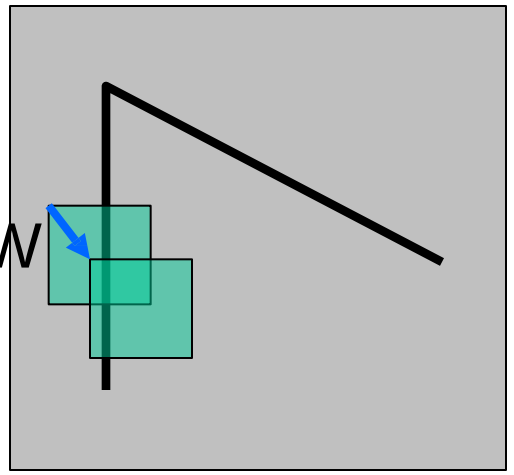
$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

Plugging this into the formula on the previous slide…

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 by summing up the squared differences
- this defines an "error" of E(u,v):



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

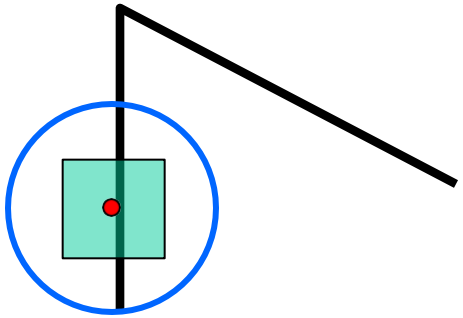
$$\approx \sum_{(x,y)\in W} [I(x,y) + [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)]^2$$

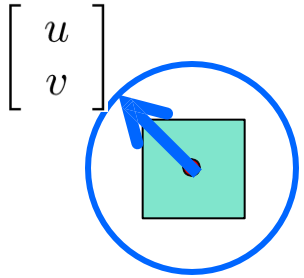
$$\approx \sum_{(x,y)\in W} \left[[I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} \right]^2$$

Feature detection: the math

This can be rewritten:

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





For the example above

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

Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy:

$$Ax = \lambda x$$

The scalar λ is the **eigenvalue** corresponding to **x**

- The eigenvalues are found by solving:

$$det(A - \lambda I) = 0$$

- In our case, ***A*** = ***H*** is a 2x2 matrix, so we have

$$det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

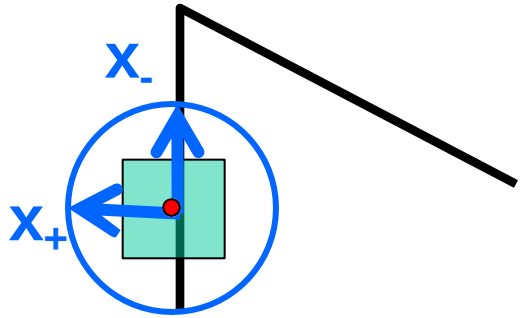
Once you know λ, you find **x** by solving

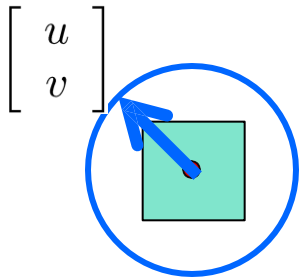
$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

Feature detection: the math

This can be rewritten:

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





Eigenvalues and eigenvectors of H

- Define 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_-

$$Hx_+ = \lambda_+ x_+$$
$$Hx_- = \lambda_- x_-$$

Feature detection: the math

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

- What's our feature scoring function?

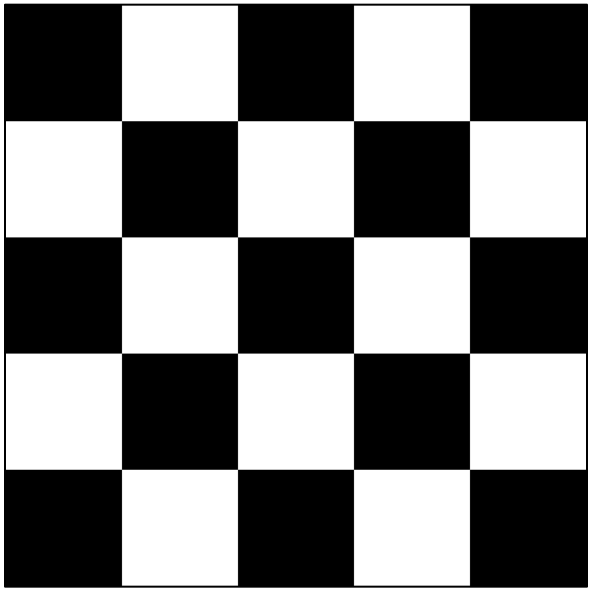
Feature detection: the math

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

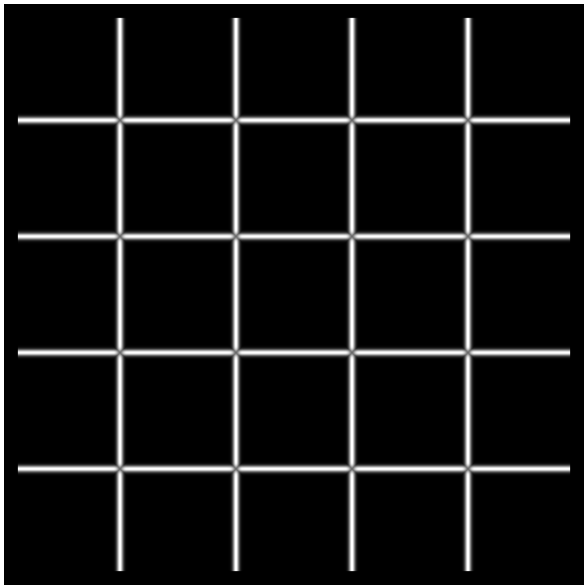
- What's our feature scoring function?

Want *E(u,v)* to be ***large*** for small shifts in ***all*** directions

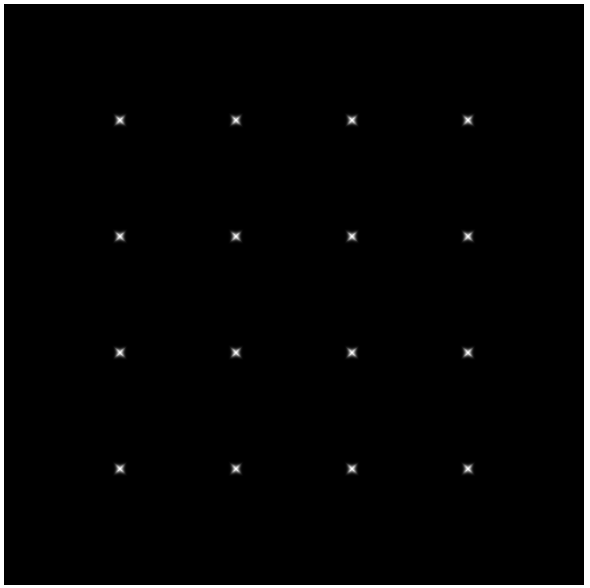
- the *minimum* of *E(u,v)* should be large, over all unit vectors [u v]
- this minimum is given by the smaller eigenvalue (λ_-) of ***H***



I



λ_+

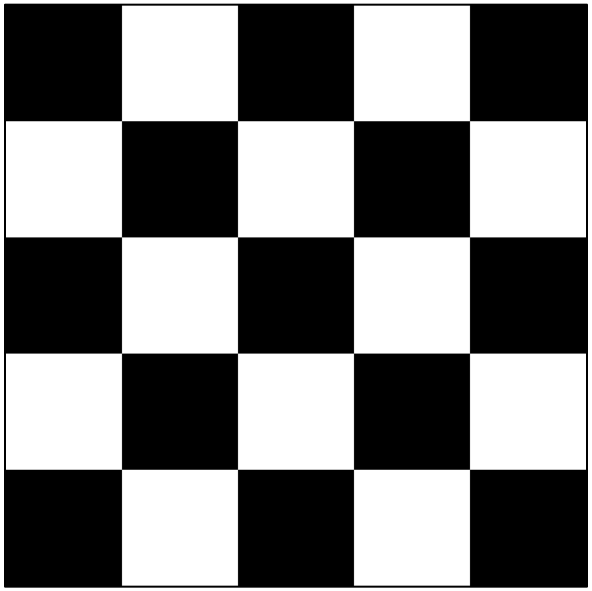


λ_-

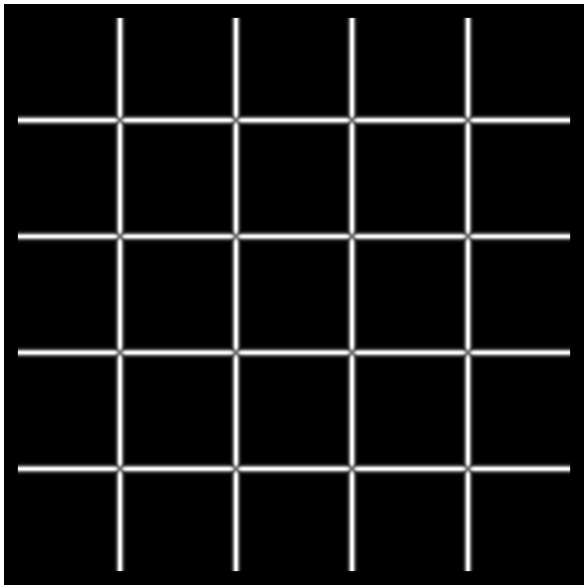
Feature detection summary (λ_-)

Here's what you do

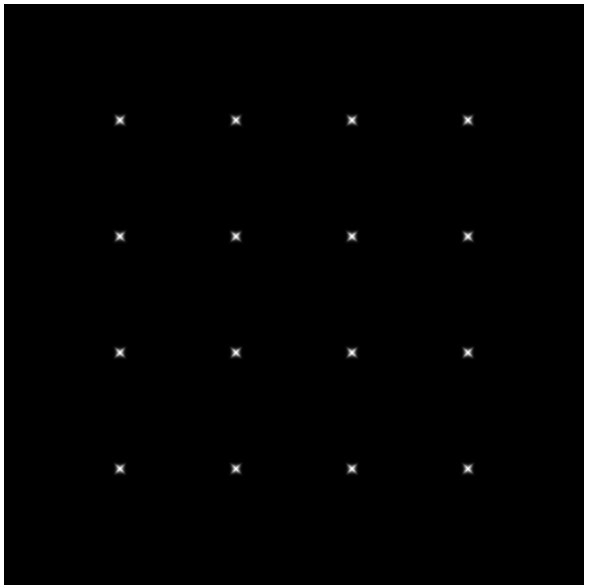
- Compute the gradient at each point in the image
- Create the ***H*** matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as features



I



λ_+

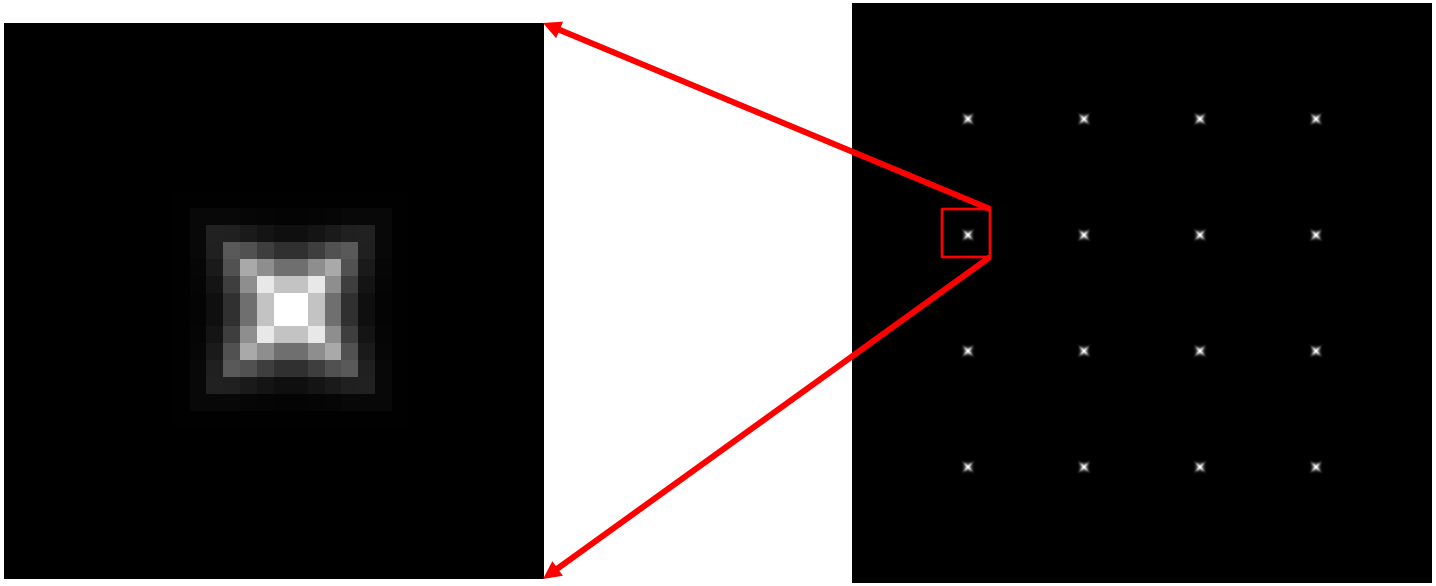


λ_-

Feature detection summary (λ_-)

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 response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as features



λ_-

The Harris operator

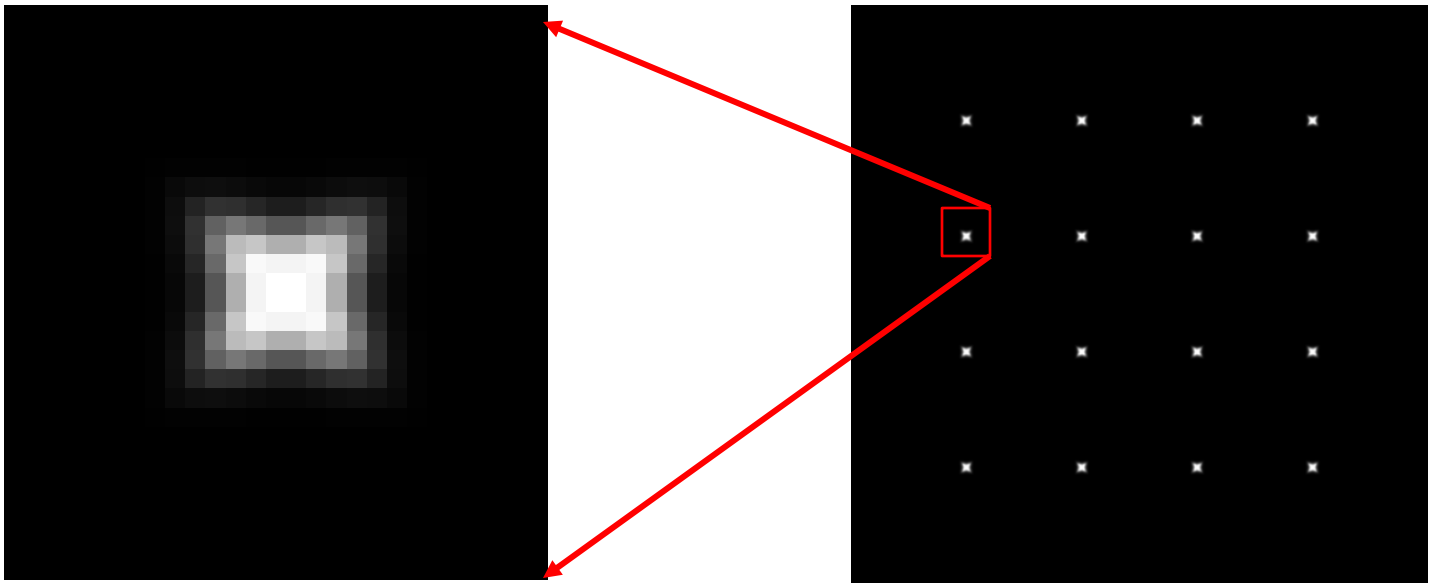
λ_- is a variant of the "Harris operator" for feature detection

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

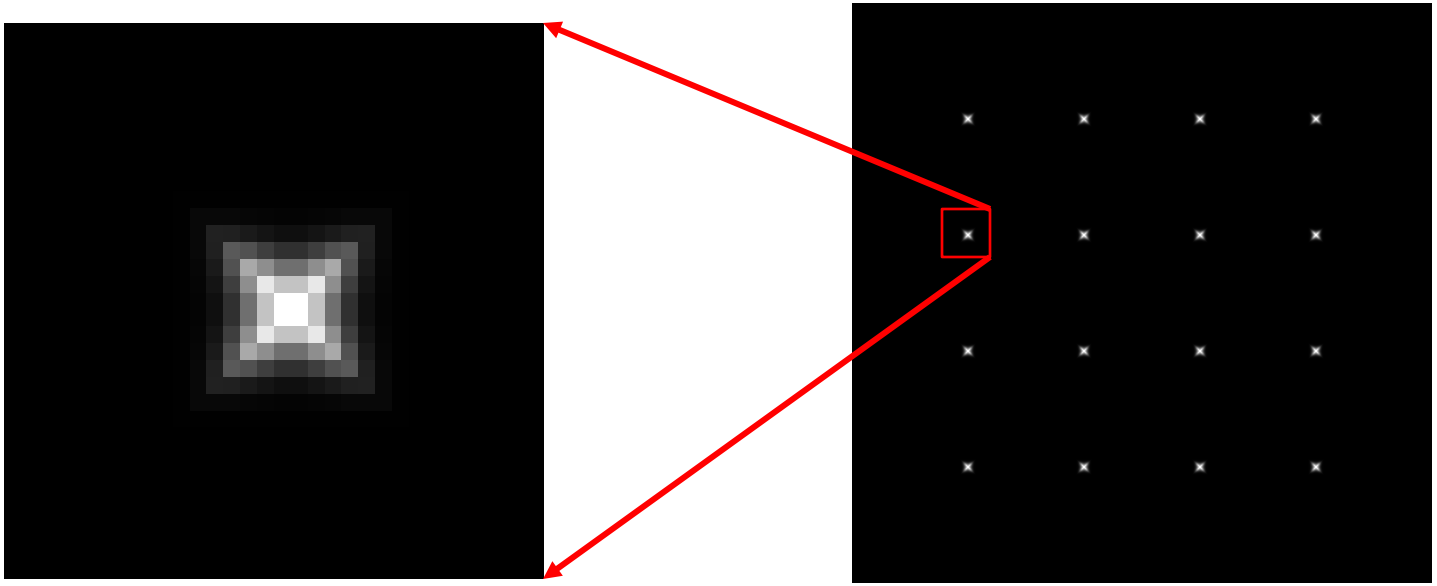
$$= \frac{determinant(H)}{trace(H)}$$

- The *trace* is the sum of the diagonals, i.e., *trace(H)* = $h_{11} + h_{22}$
- Very similar to λ_- but less expensive (no square root)
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

The Harris operator



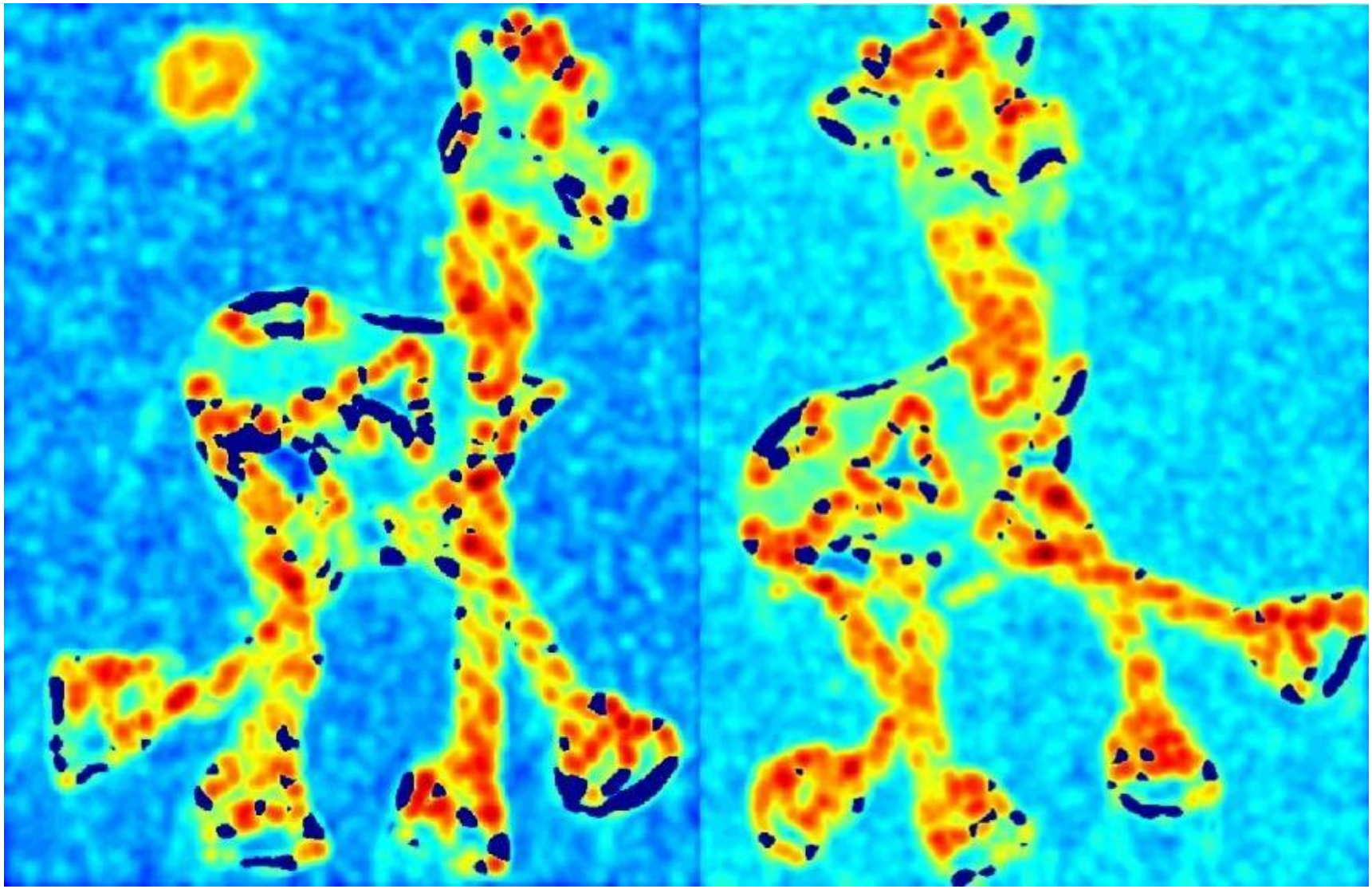
Harris operator



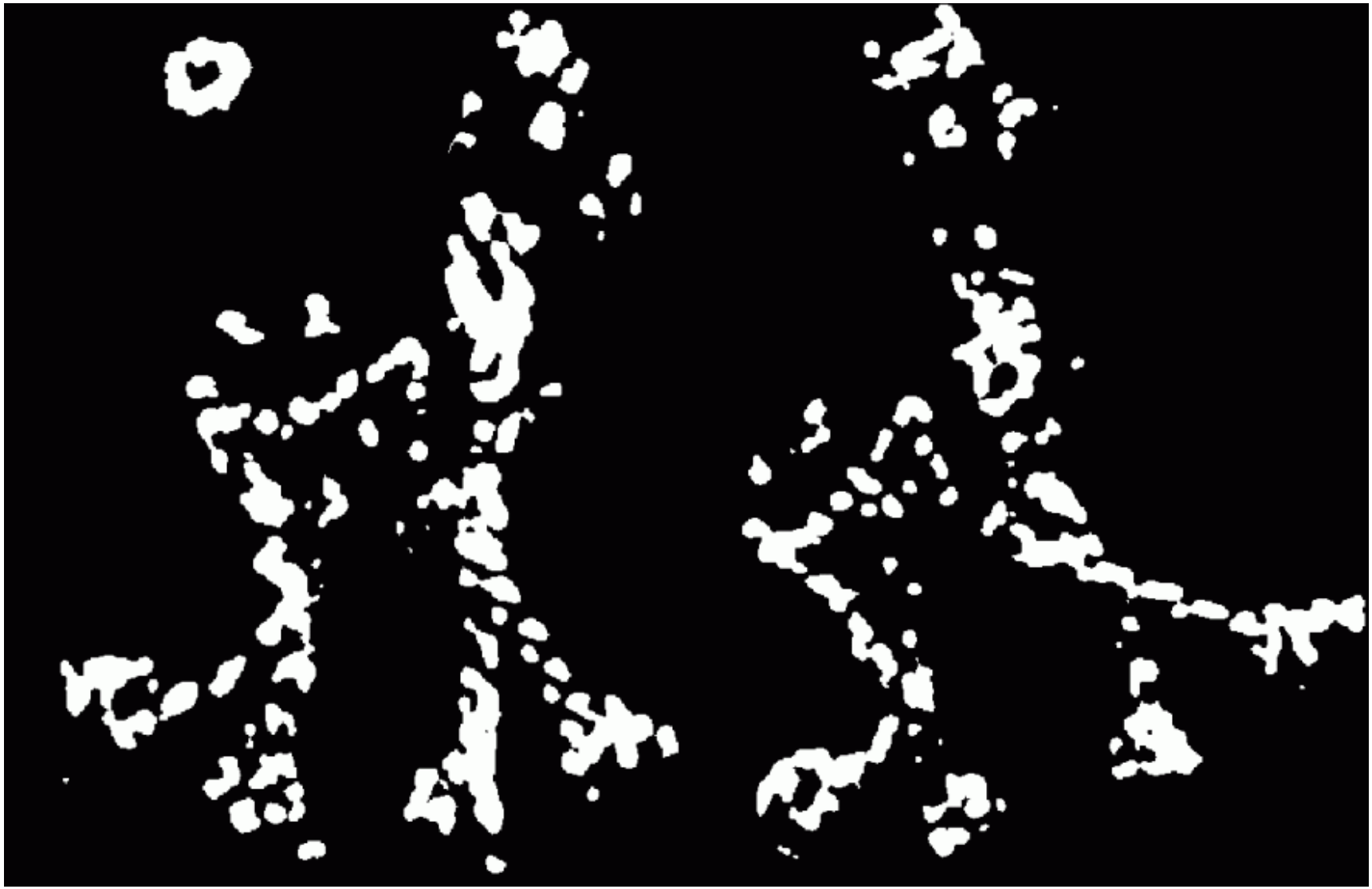
λ_-

Harris detector example

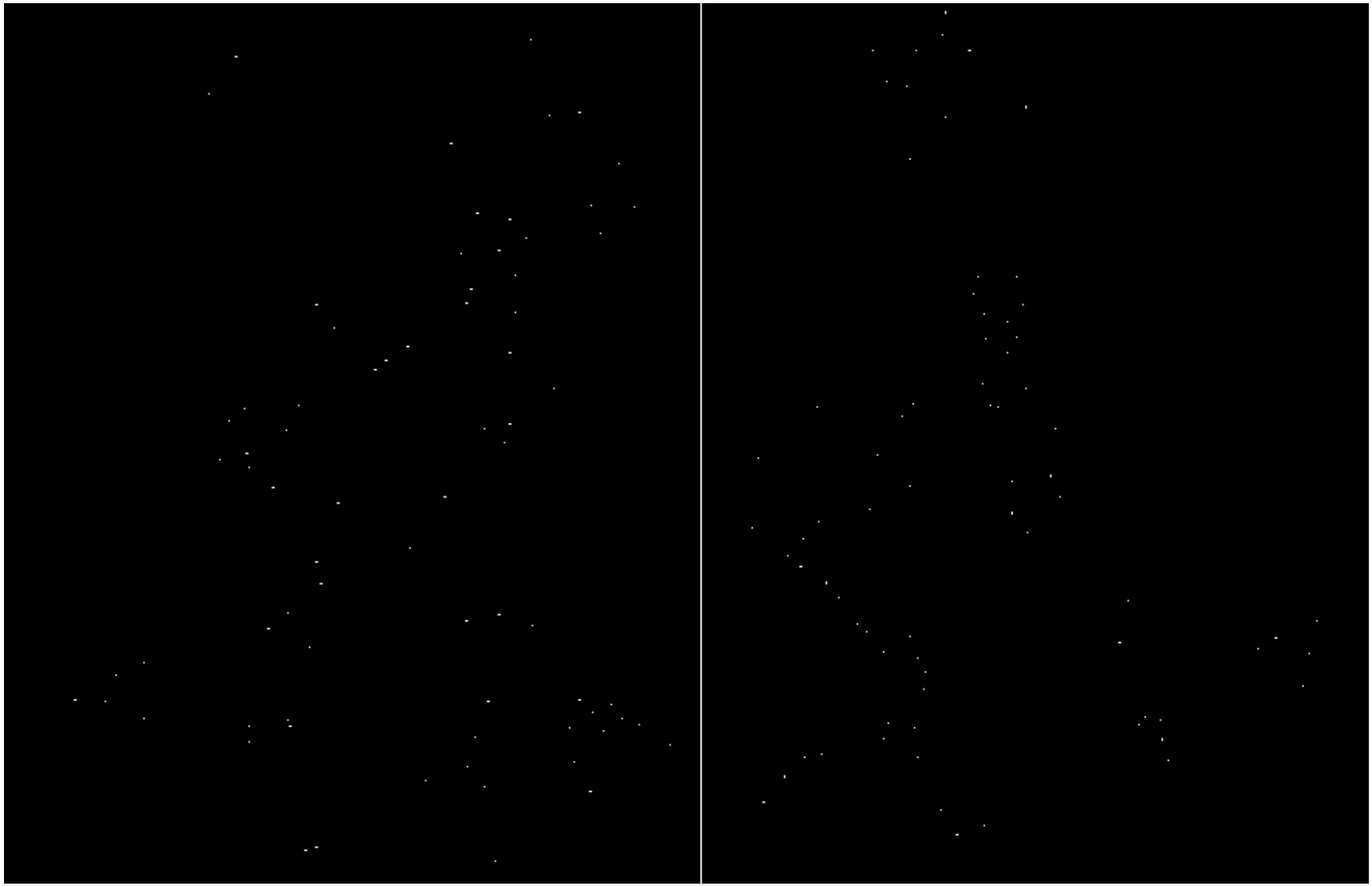
f value (red high, blue low)



Threshold (f > value)



Find local maxima of f



Harris features (in red)

Invariance

Suppose you **rotate** the image by some angle

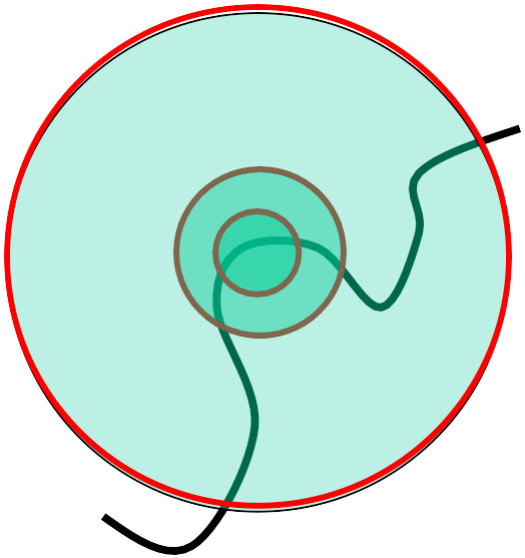
- Will you still pick up the same features?

What if you change the brightness?

Scale?

Scale invariant detection

Suppose you're looking for corners

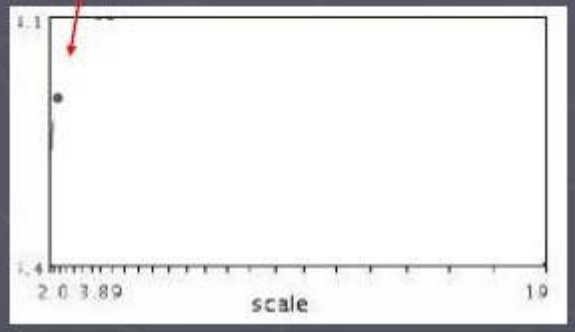


Key idea: find scale that gives local maximum of f

- f is a local maximum in both position and scale
- Common definition of f: Laplacian
  (or difference between two Gaussian filtered images with different sigmas)

Automatic scale selection

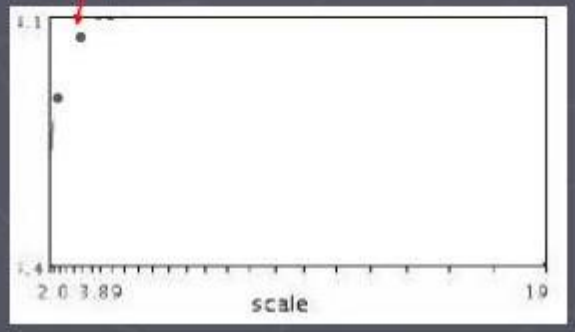
Lindeberg et al., 1996



$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

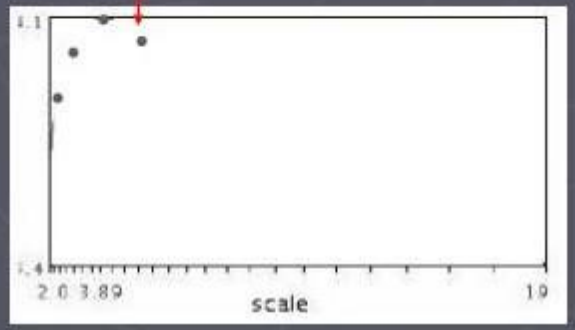
Slide from Tinne Tuytelaars

Automatic scale selection



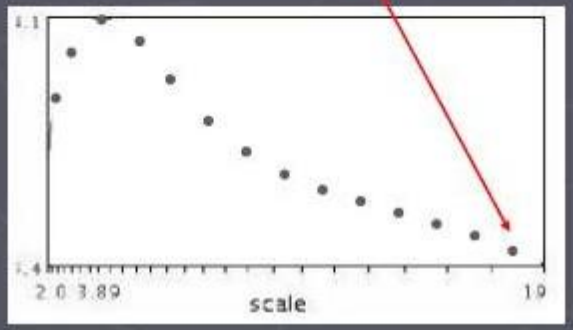
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Automatic scale selection



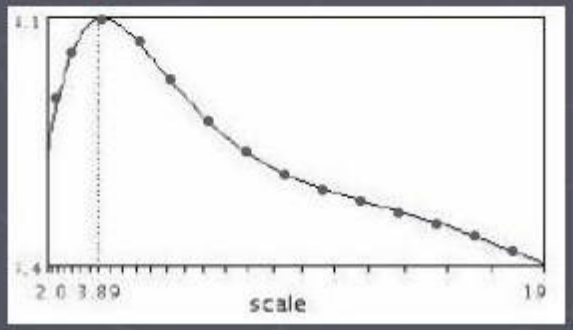
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Automatic scale selection



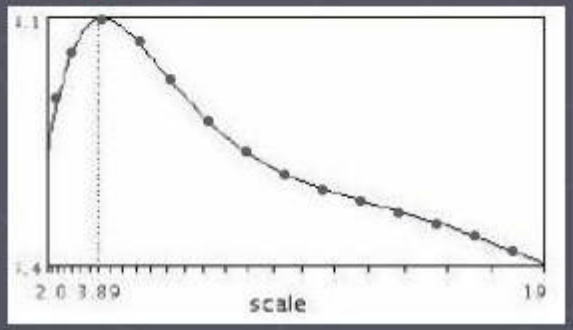
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Automatic scale selection

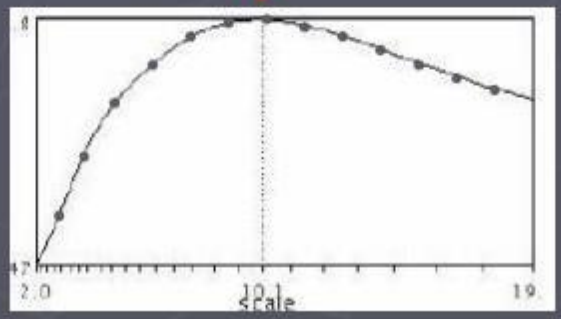


$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection



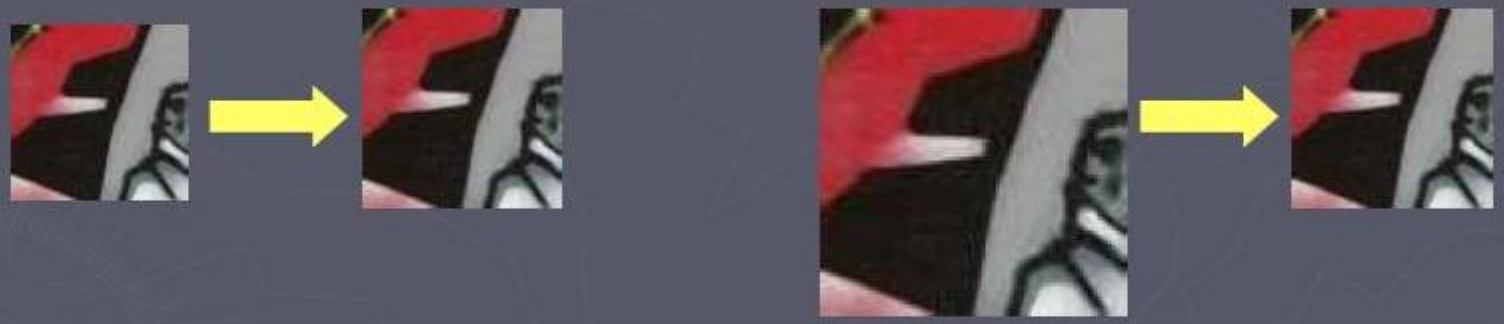
$f(I_{i_1 \ldots i_m}(x,\sigma))$



$f(I_{i_1 \ldots i_m}(x',\sigma'))$

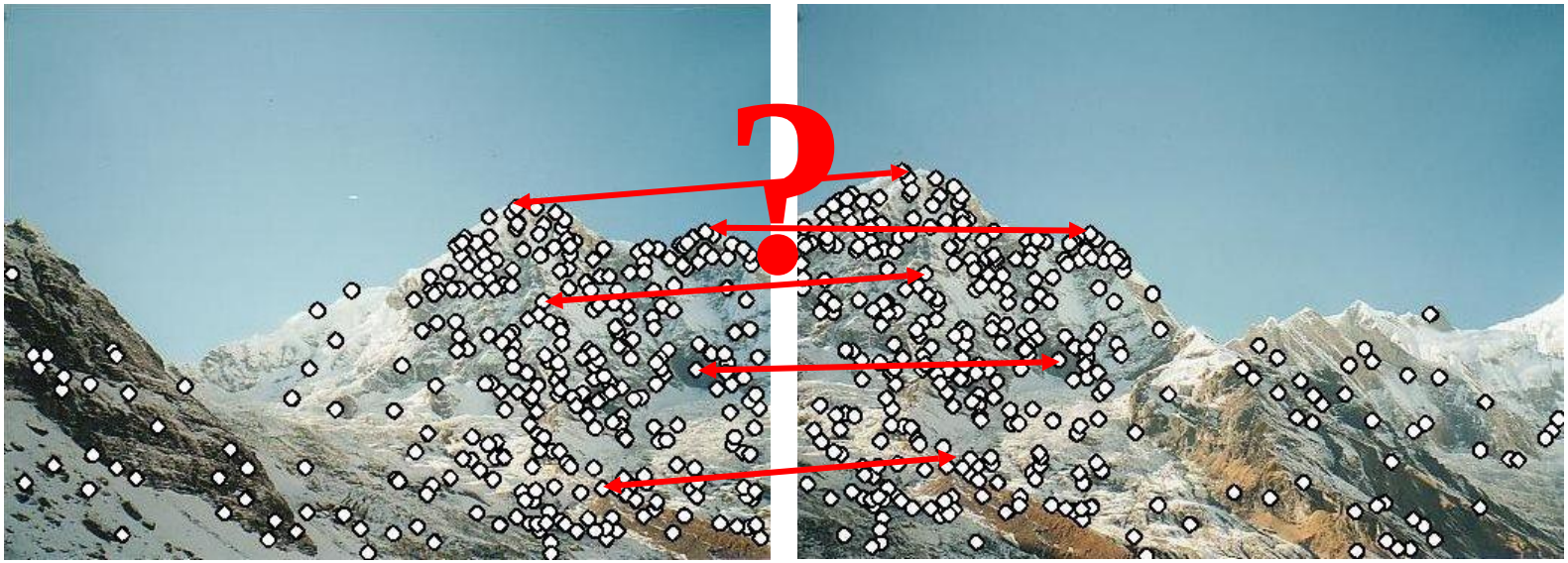
Automatic scale selection

Normalize: rescale to fixed size



Feature descriptors

We know how to detect good points
Next question: **How to match them?**



Lots of possibilities (this is a popular research area)

- Simple option: match square windows around the point
- State of the art approach: SIFT
  - David Lowe, UBC http://www.cs.ubc.ca/~lowe/keypoints/

Invariance

Suppose we are comparing two images I_1 and I_2

- I_2 may be a transformed version of I_1
- What kinds of transformations are we likely to encounter in practice?

We'd like to find the same features regardless of the transformation

- This is called transformational ***invariance***
- Most feature methods are designed to be invariant to
  - Translation, 2D rotation, scale
- They can usually also handle
  - Limited 3D rotations (SIFT works up to about 60 degrees)
  - Limited affine transformations (some are fully affine invariant)
  - Limited illumination/contrast changes

How to achieve invariance

Need both of the following:

1. Make sure your 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 information in a region around the detected feature point
   - The simplest descriptor: a square window of pixels
     - What’s this invariant to?
   - Let’s look at some better approaches…

Rotation invariance for feature descriptors

Find dominant orientation of the image patch

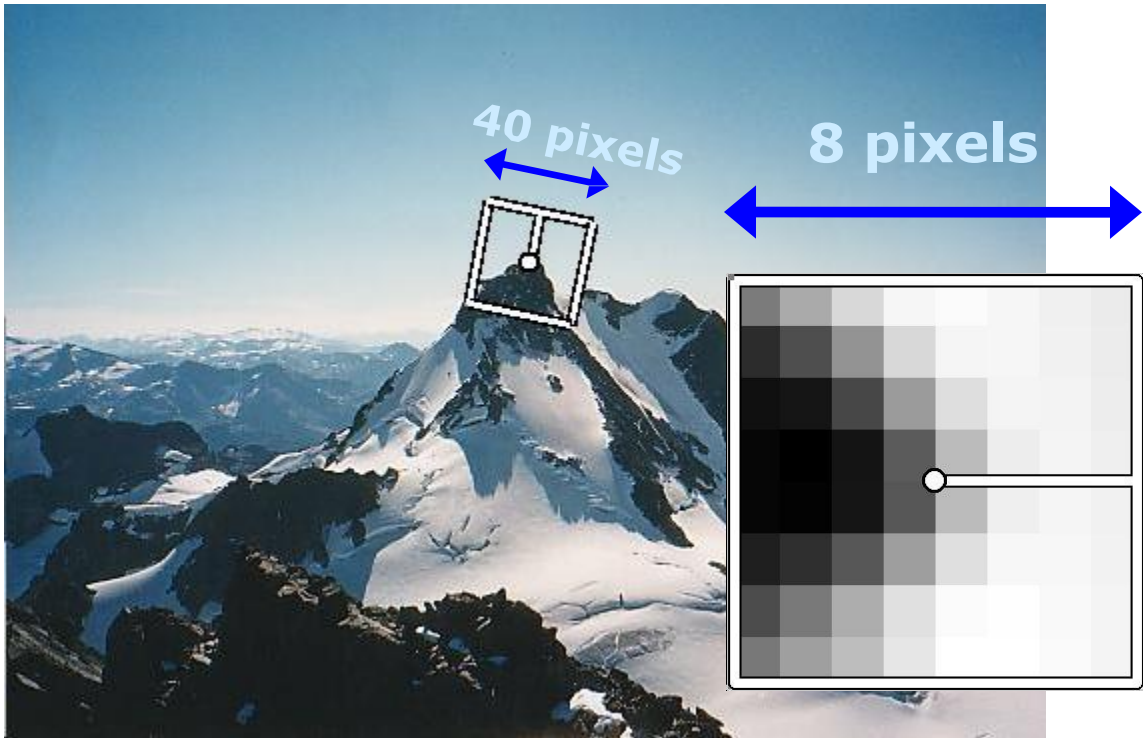
- This is given by $\mathbf{x}_+$, the eigenvector of $\mathbf{H}$ corresponding to λ_+
  - λ_+ is the *larger* eigenvalue
- Rotate the patch according to this angle

Figure by Matthew Brown

Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



Adapted from slide by Matthew Brown

Detections at multiple scales

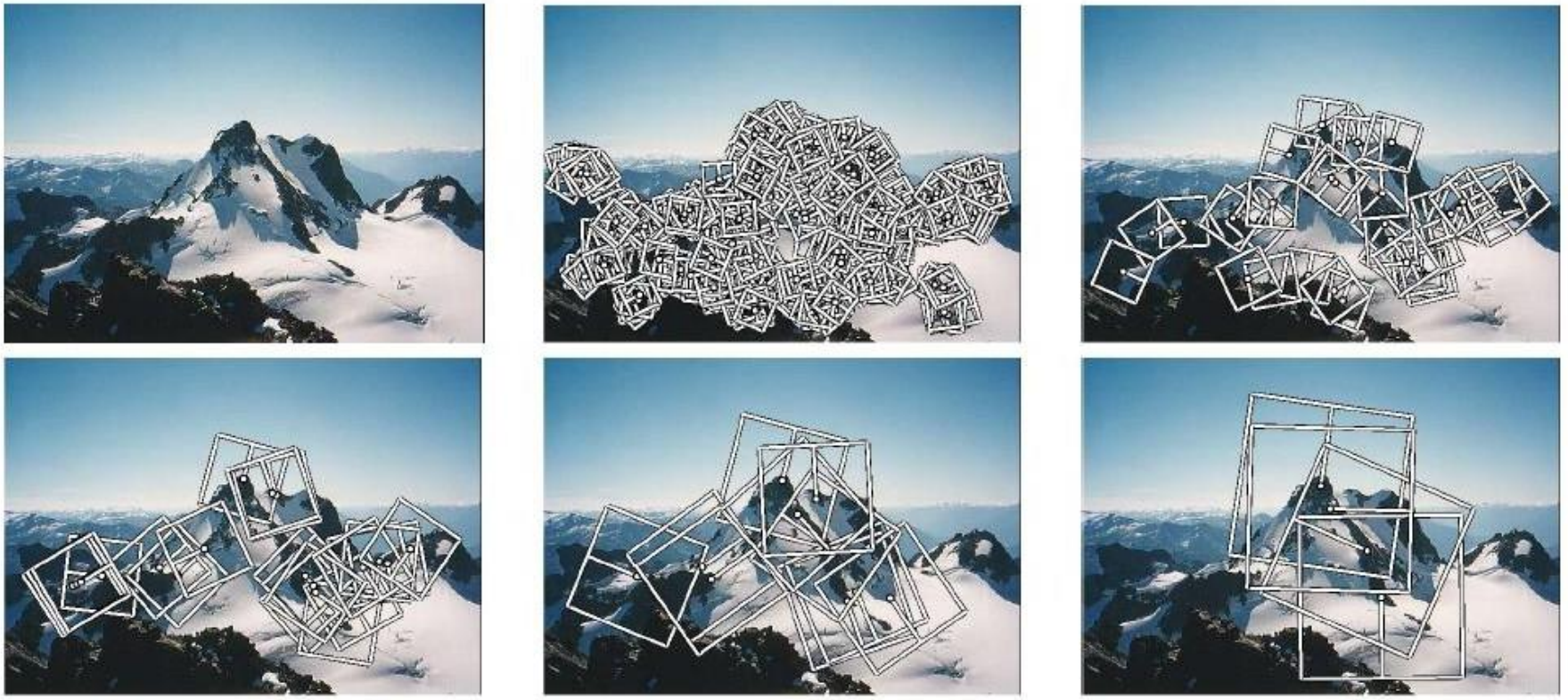
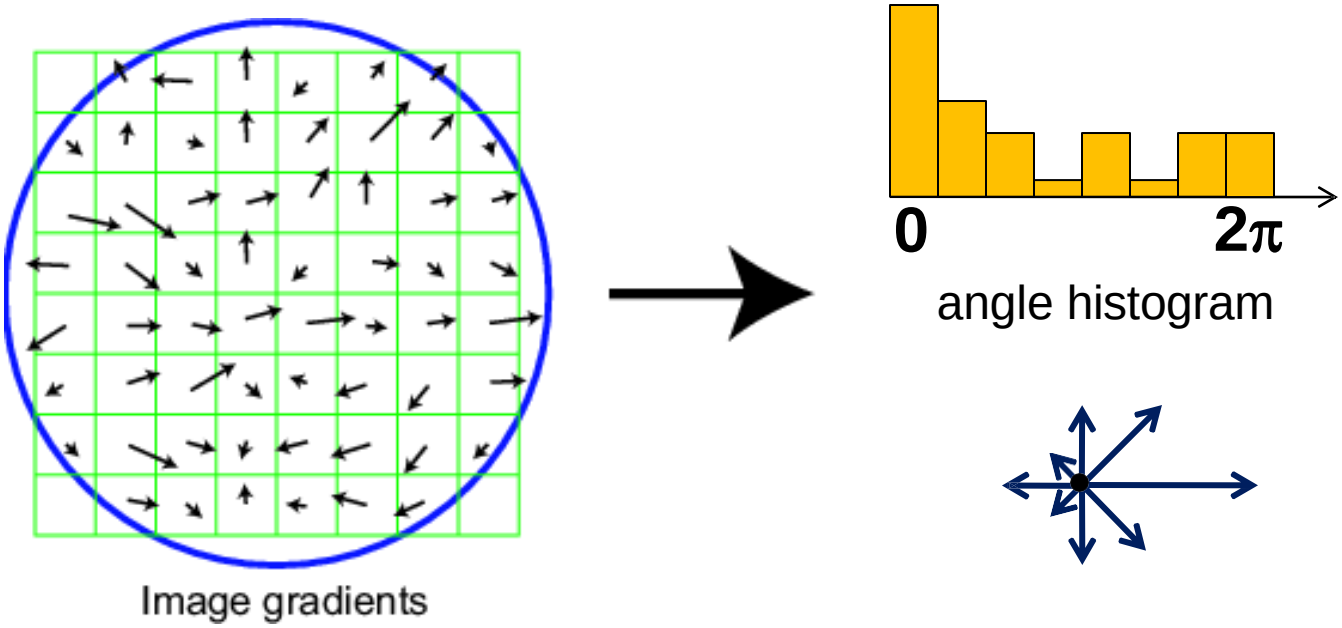


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

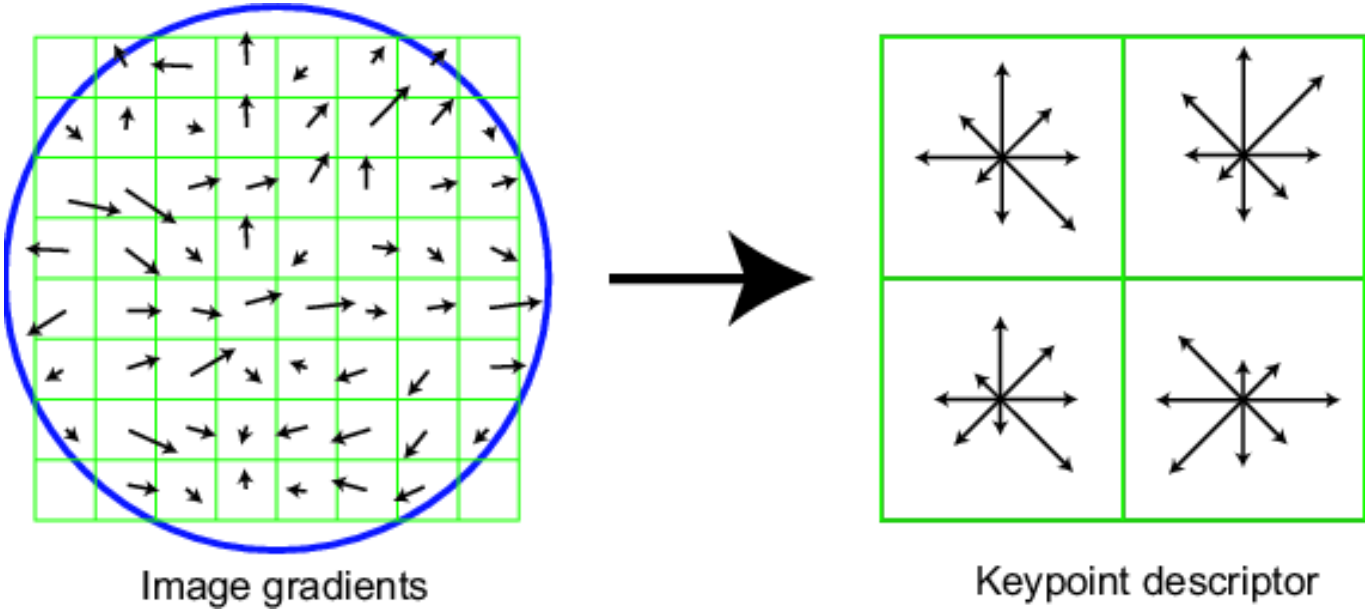


Adapted from slide by David Lowe

SIFT descriptor

Full version

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

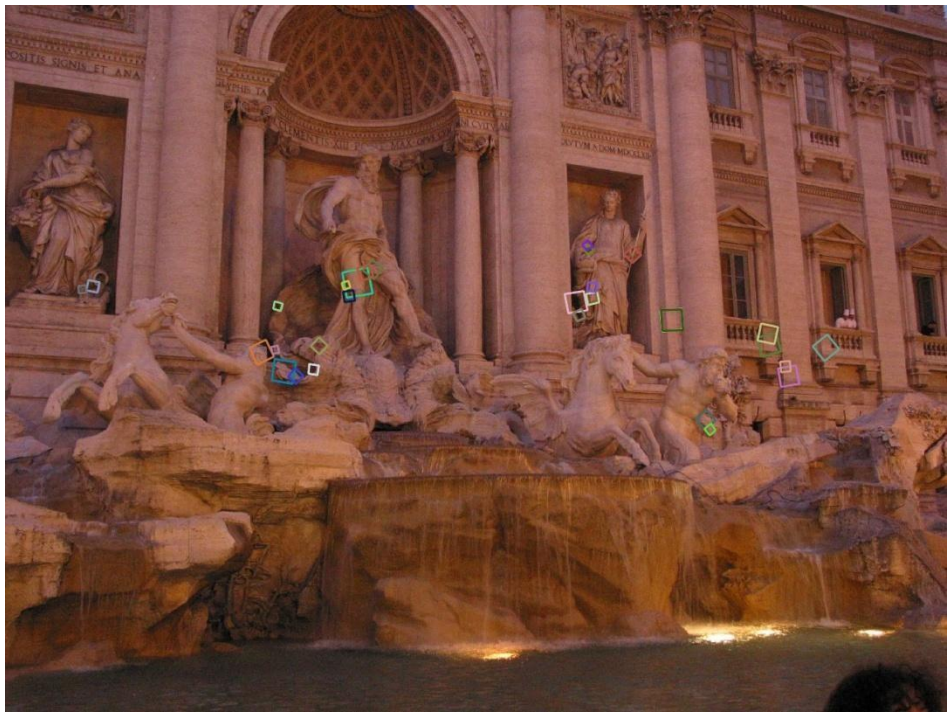


Adapted from slide by David Lowe

Properties of 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 (below)
- Fast and efficient—can run in real time
- Lots of code available
  - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT



Feature matching

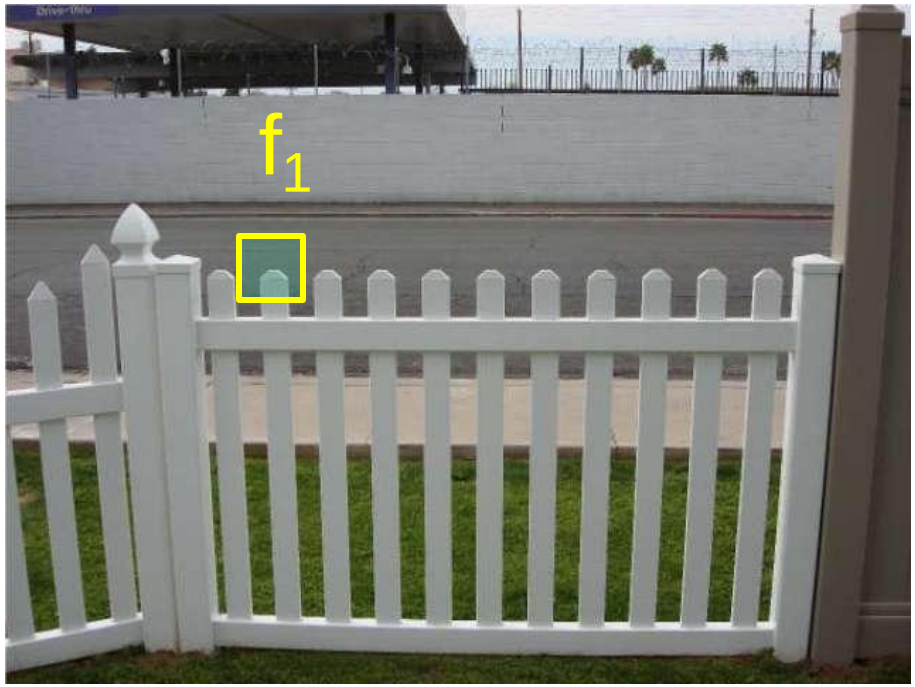
Given a feature in I_1, how to find the best match in I_2?

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

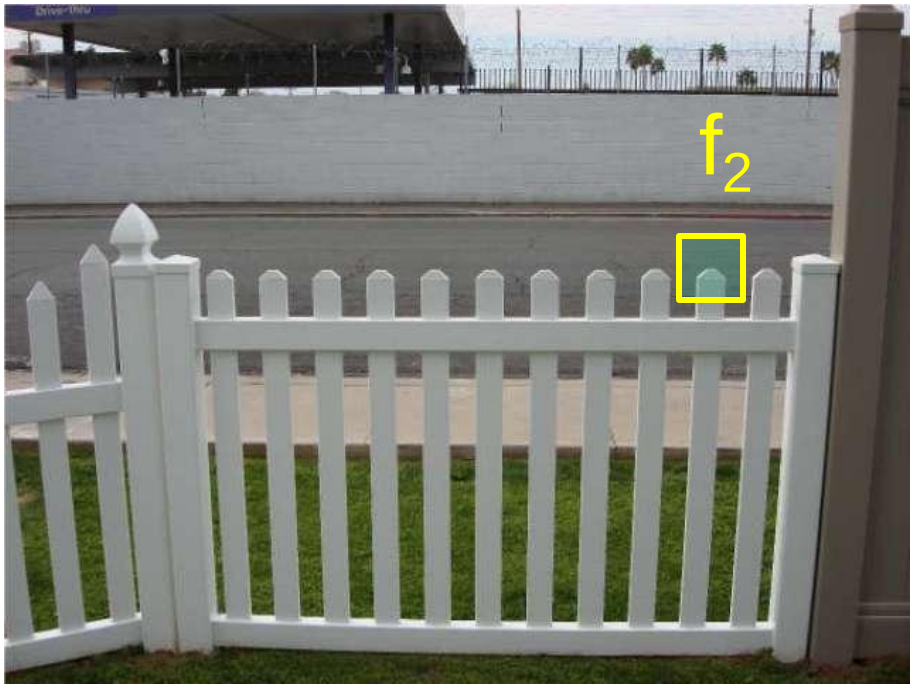
Feature distance

How to define the difference between two features f_1, f_2?

- 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



I_1

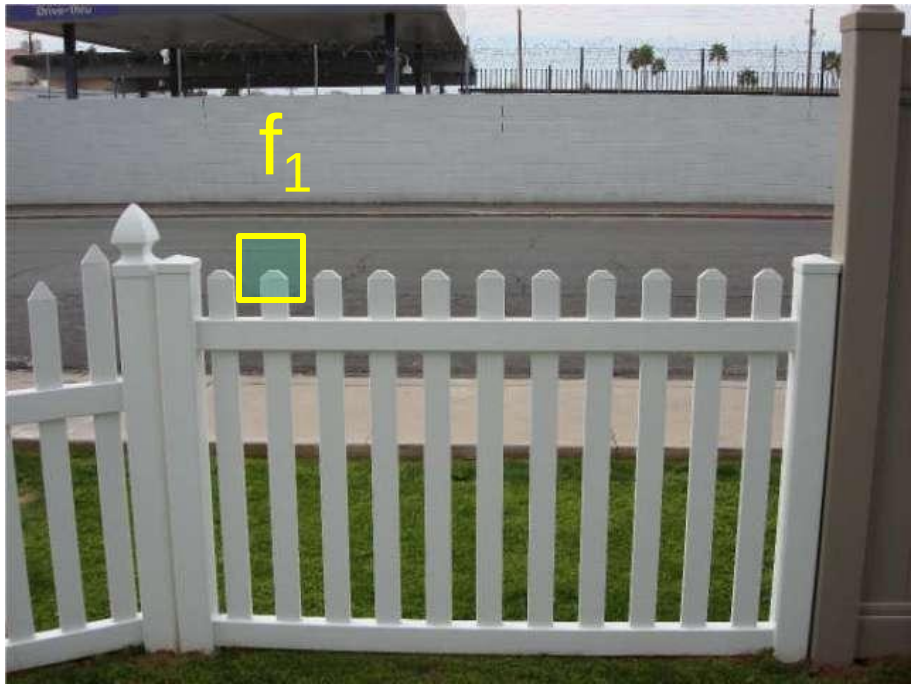


I_2

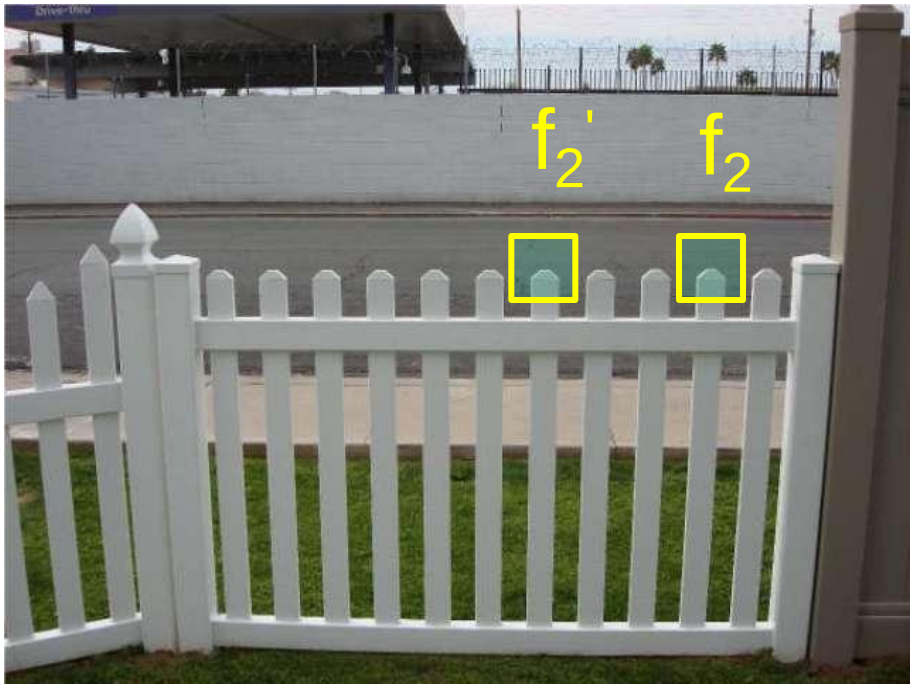
Feature distance

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
  - gives small values for ambiguous matches



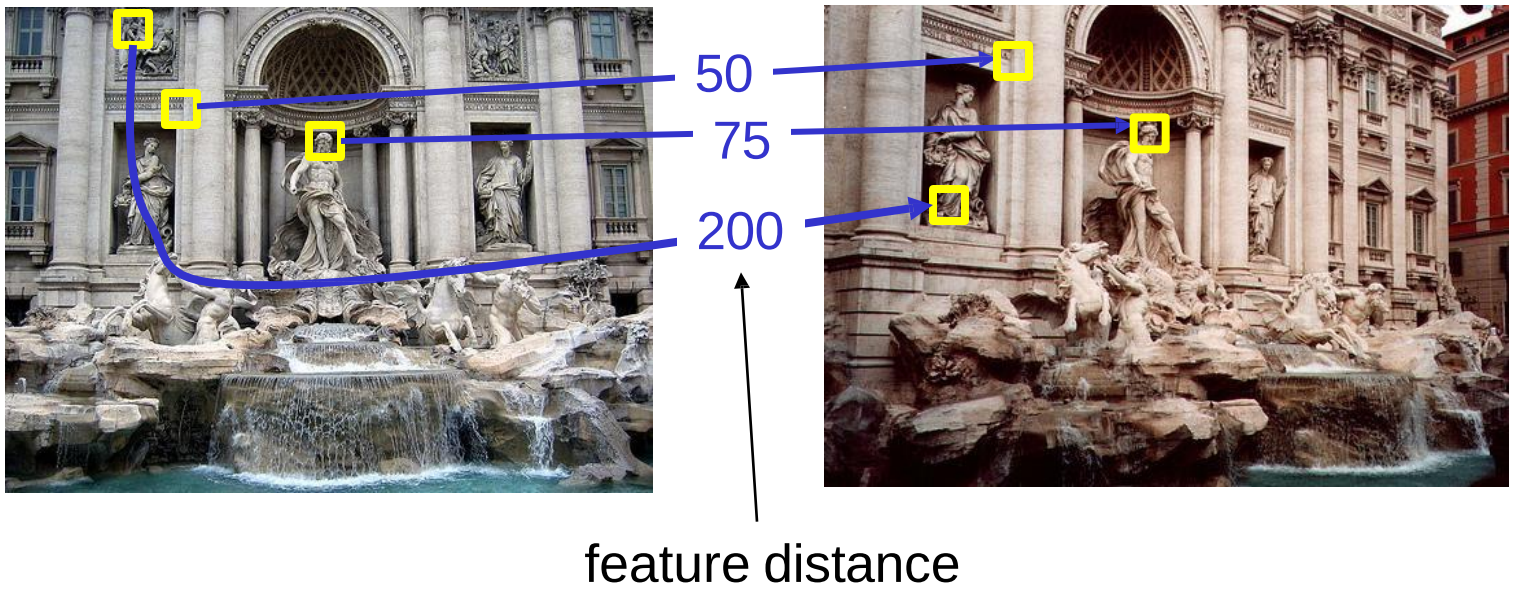
I_1



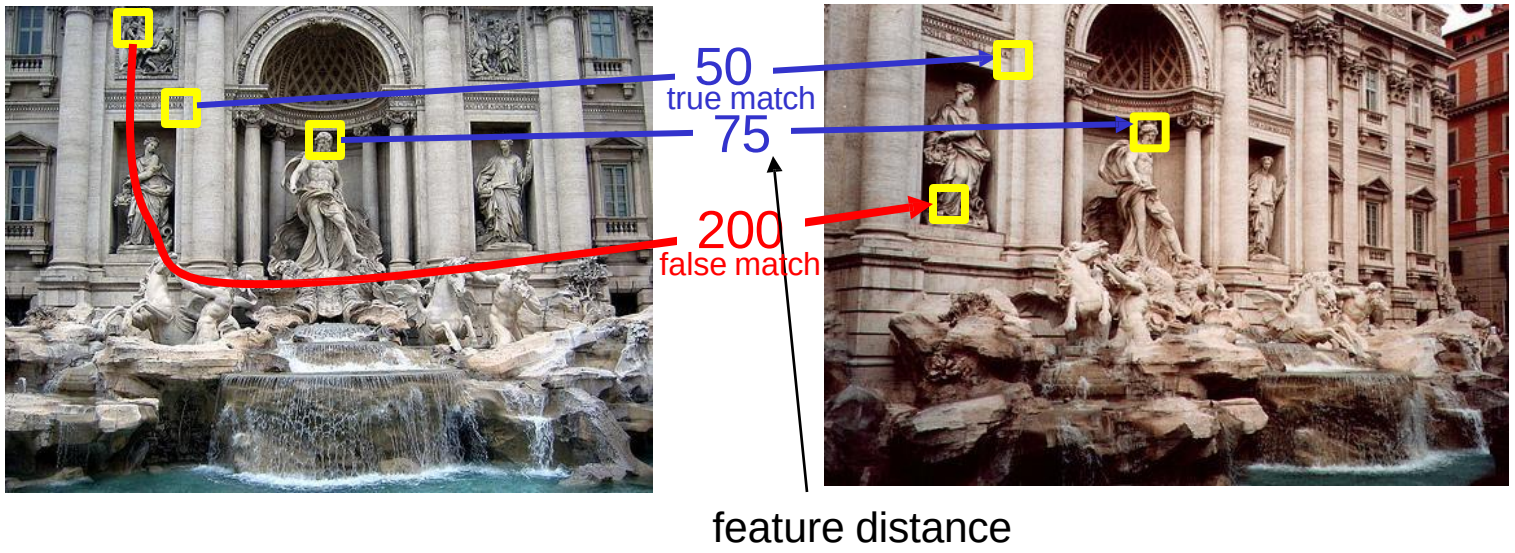
I_2

Evaluating the results

How can we measure the performance of a feature matcher?



True/false positives



The distance threshold affects performance

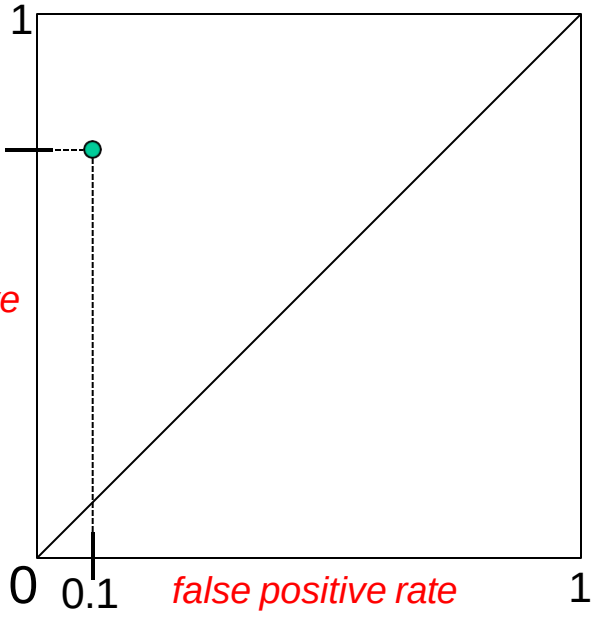
- True positives = # of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- False positives = # of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?

Evaluating the results

How can we measure the performance of a feature matcher?

$$\frac{\text{\# true positives}}{\text{\# matching features (positives)}}$$

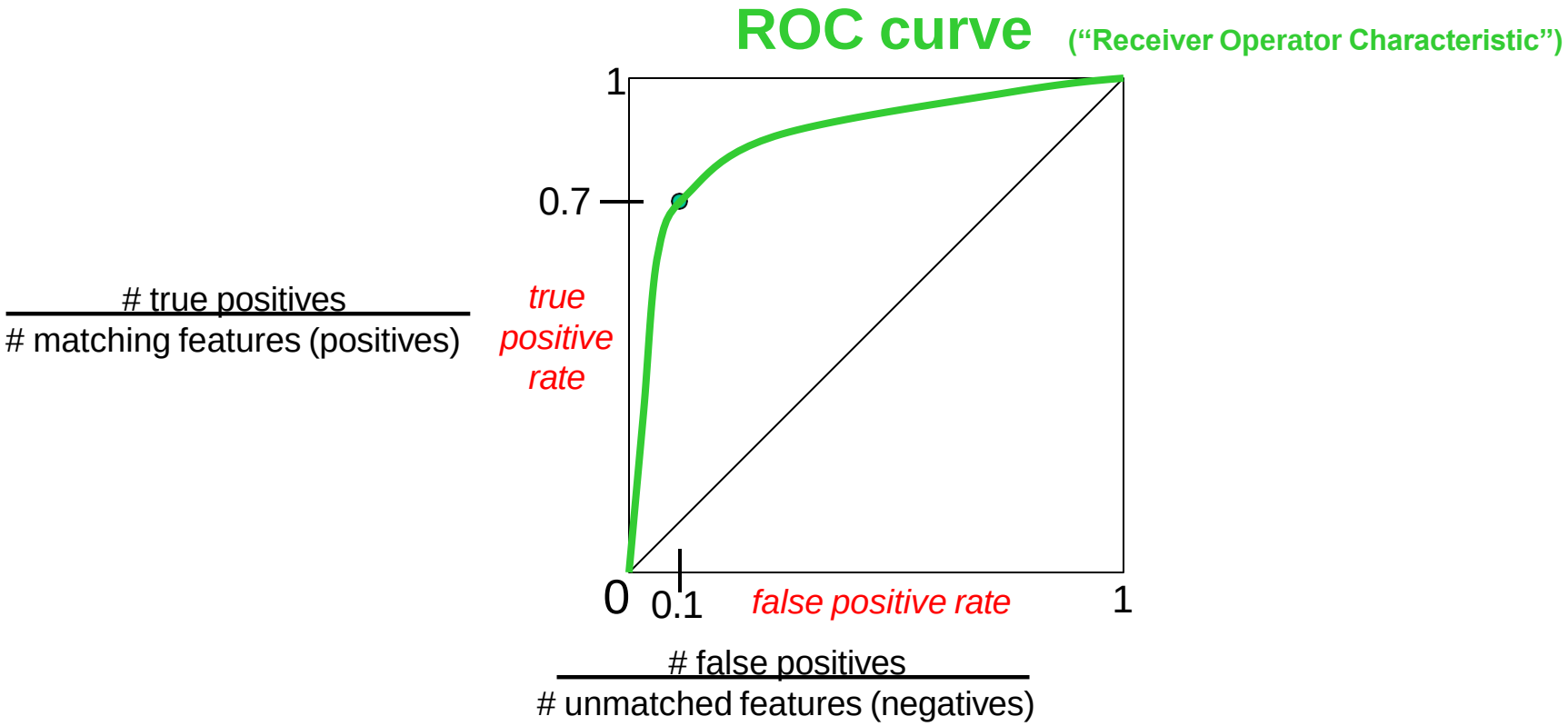
true positive rate



$$\frac{\text{\# false positives}}{\text{\# unmatched features (negatives)}}$$

Evaluating the results

How can we measure the performance of a feature matcher?



ROC Curves

- Generated by counting # current/incorrect matches, for different threholds
- Want to maximize area under the curve (AUC)
- Useful for comparing different feature matching methods
- For more info: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

More on feature detection/description

Publications

Region detectors

- *Harris-Affine & Hessian Affine*: K. Mikolajczyk and C. Schmid, Scale and Affine invariant interest point detectors. In IJCV 1(60):63-86, 2004. PDF
- *MSER*: J.Matas, O. Chum, M. Urban, and T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions. In BMVC p. 384-393, 2002. PDF
- *IBR & EBR*: T.Tuytelaars and L. Van Gool, Matching widely separated views based onaffine invariant regions. In IJCV 1 (59):61-85, 2004. PDF
- *Salient regions*: T. Kadir, A. Zisserman, and M. Brady, An affine invariant salient region detector. In ECCV p. 404-416, 2004. PDF

Region descriptors

- *SIFT*: D. Lowe, Distinctive image features from scale invariant keypoints. In IJCV 2(60):91-110, 2004. PDF

Performance evaluation

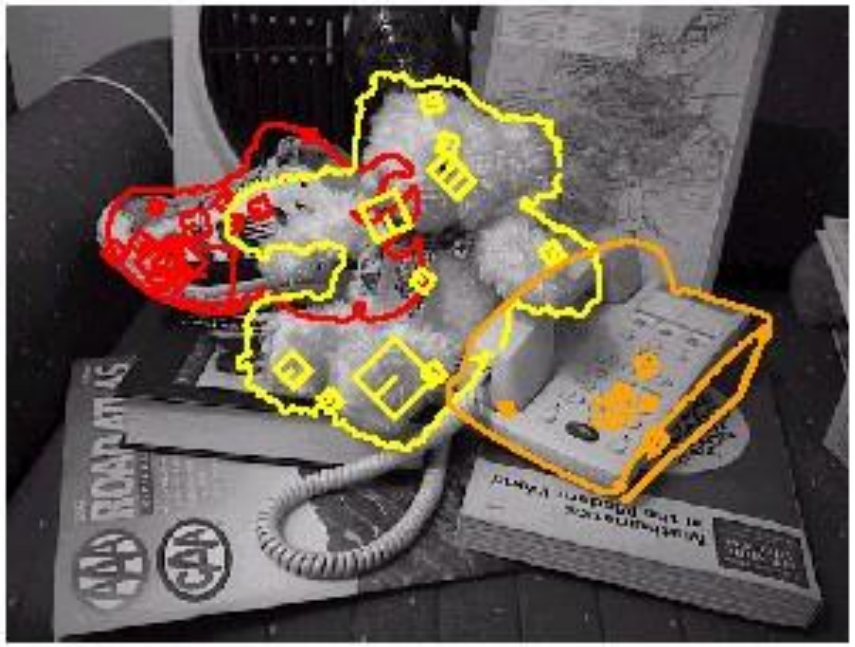
- K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir and L. Van Gool, A comparison of affine region detectors. Technical Report, accepted to IJCV. PDF
- K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors. Technical Report, accepted to PAMI. PDF

Lots of applications

Features are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

Object recognition (David Lowe)



Sony Aibo

SIFT usage:

- Recognize charging station
- Communicate with visual cards
- Teach object recognition

Computer Vision

UNIT II

HIGH Level: Instance Recognition, Object Detection, Image Classification, Semantic Segmentation, Video Understanding, Vision and Language.(Chap 6)

6.1 Instance Recognition

Figure 6.2 *Recognizing objects in a cluttered scene (Lowe 2004) © 2004 Springer. Two of the training images in the database are shown on the left. They are matched to the cluttered scene in the middle using SIFT features, shown as small squares in the right image. The affine warp of each recognized database image onto the scene is shown as a larger parallelogram in the right image.*

6.1 Instance Recognition

1. Definition

•General object recognition has two categories:

- **Instance Recognition**:
  - Recognizing a **specific known 2D or 3D rigid object**.
  - Challenges: new viewpoints, cluttered background, partial occlusions.
  - Example: Recognizing objects in cluttered scenes (Figure **6.2**).
- **Class Recognition (Category-level recognition)**:
  - Recognizing any instance of a **general class** (e.g., *cat, car, bicycle*).
  - Much more difficult.

Early Approaches

Geometric-based methods: Extracting lines, contours, 3D surfaces → match to 3D object models.

•Appearance-based methods: Capturing many images under various viewpoints & illuminations → eigenspace decomposition (Murase & Nayar, 1995).

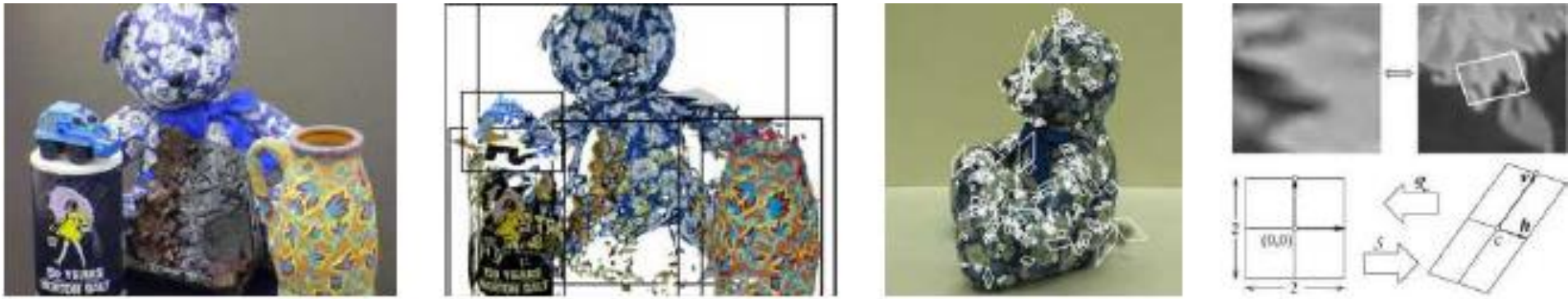
Modern Approaches

- **Use viewpoint-invariant 2D features (e.g., SIFT, affine regions).**
- **Pipeline:**
  - Extract sparse, informative 2D features from image & database.
  - Match features using **feature matching strategies**.
  - Verify matches via **geometric transformation alignment**.
- **Example: SIFT-based recognition and affine warping of objects onto cluttered scenes (Figure 6.2).**

Modern Approaches

- **Steps in recognition:**
  - Extract **interest points** & store descriptors in a search structure (e.g., tree).
  - Extract features from new image.
  - Find ≥3 matches for an object.
  - Perform **match verification** to check spatial arrangement consistency.
- **Challenges: Outliers due to clutter & shared features.**
- **Solutions:**
  - **Lowe (2004):**
    - Uses **Hough transform** to accumulate votes for **affine transformations**.
    - Works for planar / quasi-planar objects.
    - Shown in Figure **6.2** (affine warp shown as parallelogram).
  - **Rothganger, Lazebnik et al. (2006):**
    - Detect affine regions → rectify patches.
    - Compute **SIFT + color histogram** descriptors.
    - Build **3D affine models** (factorization → Euclidean reconstruction).
    - Use **local geometric constraints** for verification.
    - Example results: **5 recognized objects in cluttered scene** (Figure **6.3a–b**).
    - Affine regions and rectified patches shown in Figure **6.3c–d**.

Modern Approaches



(a) (b) (c) (d)

Figure 6.3 *3D object recognition with affine regions (Rothganger, Lazebnik* et al. *2006)* © *2006 Springer: (a) sample input image; (b) five of the recognized (reprojected) objects along with their bounding boxes; (c) a few of the local affine regions; (d) local affine region (patch) reprojected into a canonical (square) frame, along with its geometric affine transformations.*

Modern Approaches

- **Pixel-level segmentation approaches:**
  - Ferrari, Tuytelaars, Van Gool (2006): Recognition + segmentation.
  - Kannala, Rahtu et al. (2008): Extended to **non-rigid deformations**.
- **Shift in research focus:**
  - Early-mid 2000s → locating known 3D objects
  - Later → **Instance retrieval (CBIR)** = searching in large image databases.
  - Related areas:
    - **Visual similarity search**
    - **3D pose estimation**

Key Techniques & Tools

- **Feature descriptors: SIFT, affine regions.**
- **Verification methods: Hough transform, geometric constraints.**
- **3D reconstruction: Factorization → Euclidean upgrade.**
- **Applications:**
  - Recognition in cluttered scenes (Figures **6.2, 6.3**).
  - Large-scale **image retrieval**.
  - **Recognition + segmentation**.

Image Classification (6.2)

1. Introduction

- **Goal: Assign a label to an image based on the main object or scene.**
- **Difference from Instance Recognition**
  - Instance recognition → recognizes a *specific known object*.
  - Classification → recognizes *any member of a class* (e.g., “cat,” “car”).
- **Challenges: High variation in pose, scale, lighting, background, occlusion.**
- **Applications: Large-scale datasets and benchmarks have fueled progress.**

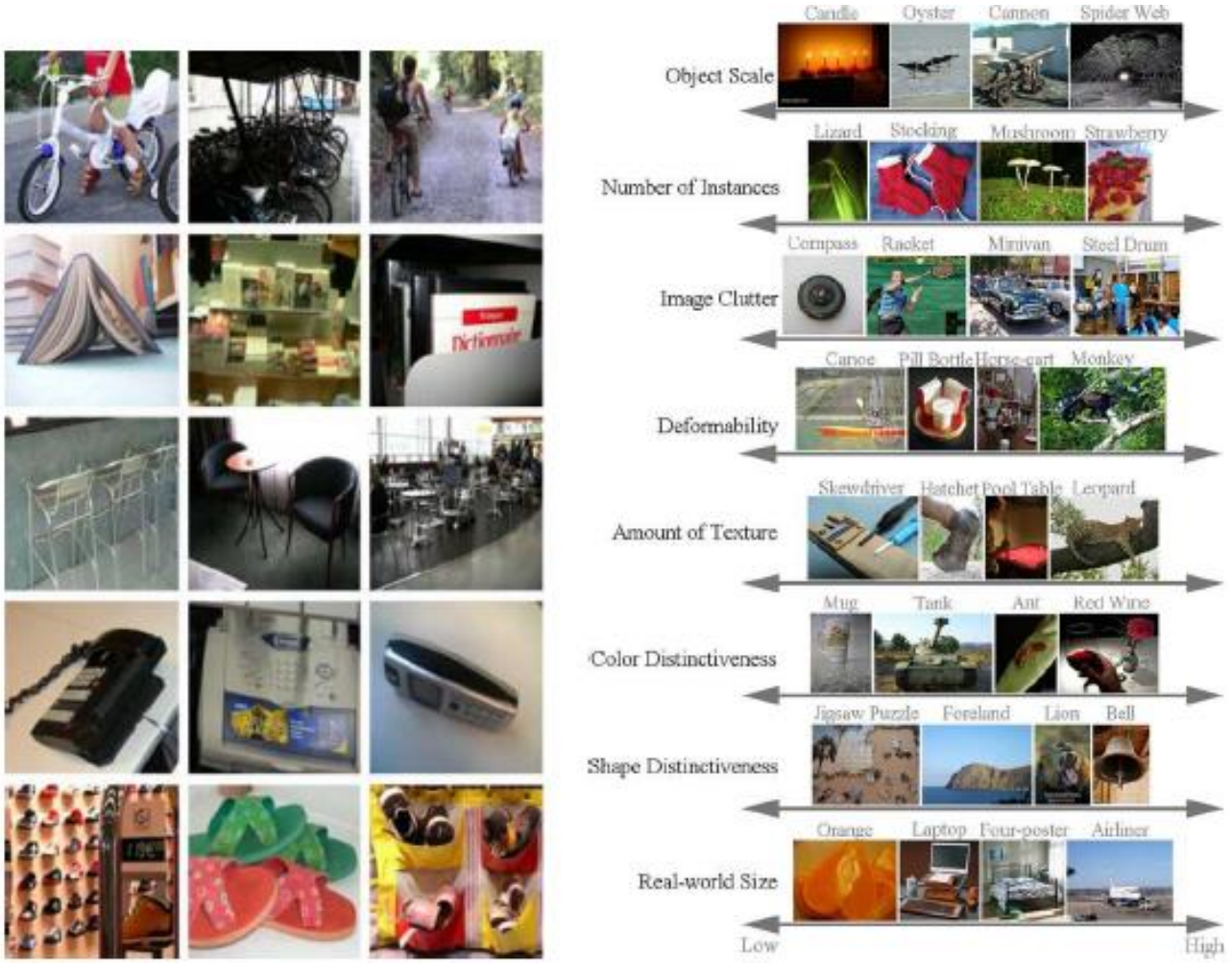


Figure 6.4 *Challenges in image recognition: (a) sample images from the Xerox 10 class dataset (Csurka, Dance* et al. *2006) © 2007 Springer; (b) axes of difficulty and variation from the ImageNet dataset (Russakovsky, Deng* et al. *2015) © 2015 Springer.*

Feature-based Methods

- **Early classification systems used handcrafted features.**
- **Bag-of-Features (BoF) pipeline:**
  - Extract local descriptors (e.g., SIFT, HOG).
  - Quantize into **visual words** (clustering).
  - Represent image as a **histogram of word counts**.
  - Classify using SVM or other classifiers.
- **Spatial Pyramid Matching (Lazebnik et al., 2006): Adds spatial layout info → improves accuracy (Figure 6.5).**
- **Attribute-based representations: Images described by semantic attributes like "striped," "furry" (Figure 6.6).**
- **Large-scale benchmarks:**
  - Caltech-101/256, Pascal VOC (Figure **6.7**).
  - SUN, ImageNet → drove progress.

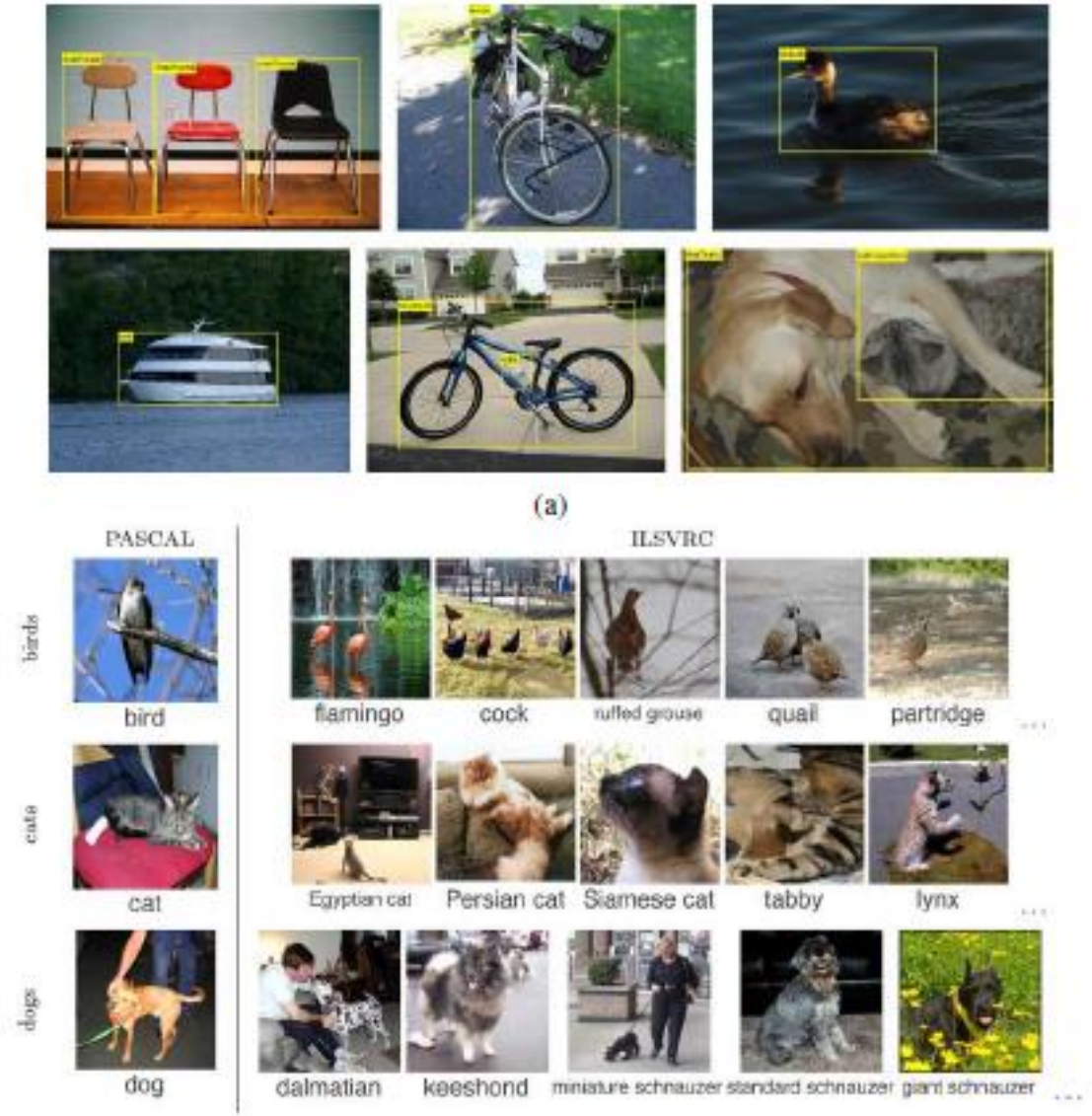


Figure 6.5 *Sample images from two widely used image classification datasets: (a) Pascal Visual Object Categories (VOC) (Everingham, Eslami* et al. *2015) © 2015 Springer; (b) ImageNet (Russakovsky, Deng* et al. *2015) © 2015 Springer.*

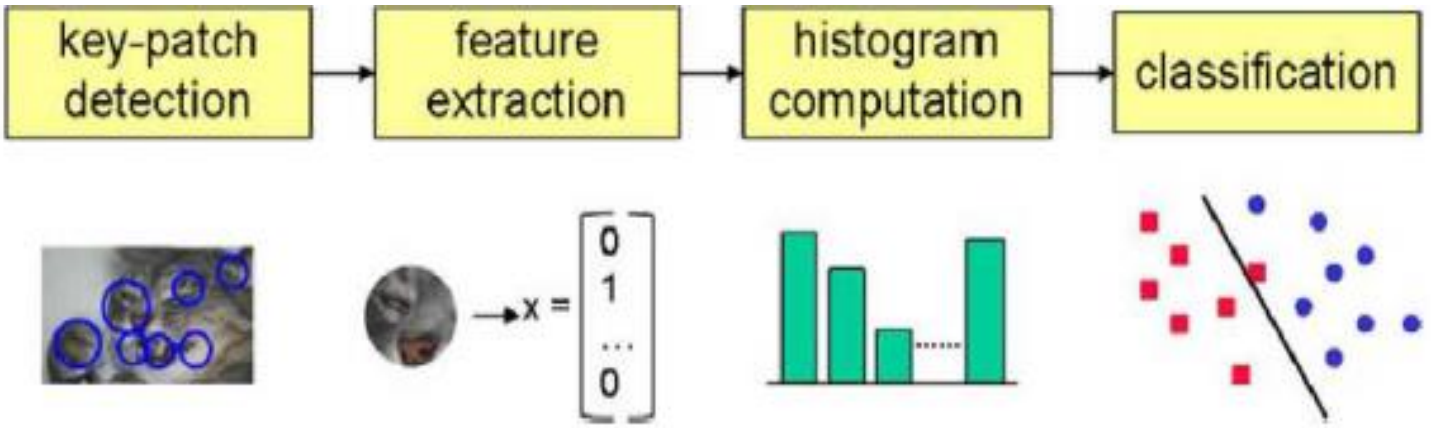


Figure 6.6 *A typical processing pipeline for a bag-of-words category recognition system (Csurka, Dance* et al. *2006) © 2007 Springer. Features are first extracted at keypoints and then quantized to get a distribution (histogram) over the learned* visual words *(feature cluster centers). The feature distribution histogram is used to learn a decision surface using a classification algorithm, such as a support vector machine.*

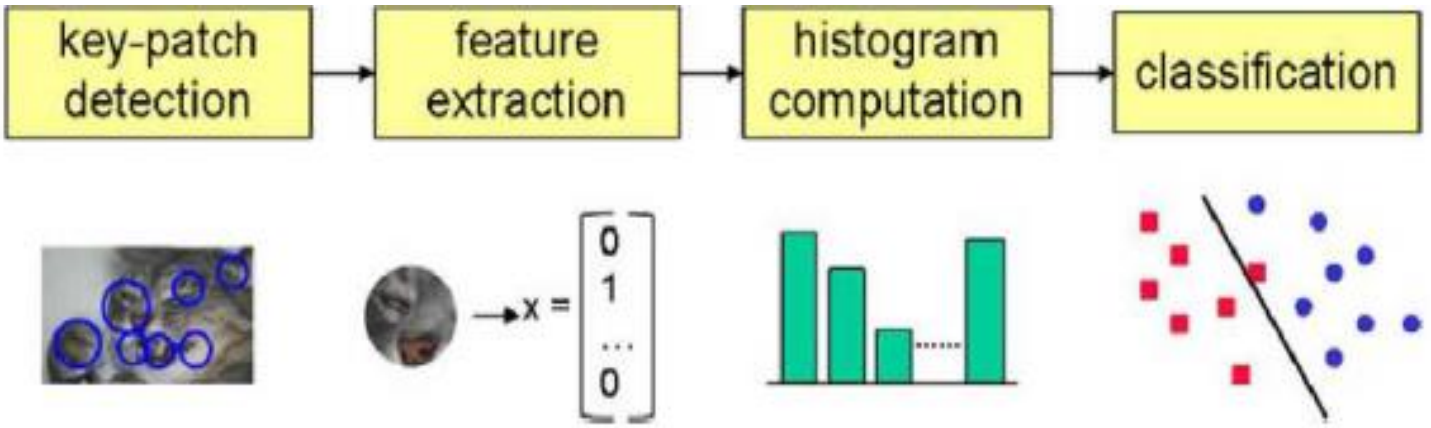


Figure 6.6 *A typical processing pipeline for a bag-of-words category recognition system (Csurka, Dance* et al. *2006) © 2007 Springer. Features are first extracted at keypoints and then quantized to get a distribution (histogram) over the learned* visual words *(feature cluster centers). The feature distribution histogram is used to learn a decision surface using a classification algorithm, such as a support vector machine.*

Deep Networks

- **Revolution: Deep Convolutional Neural Networks (CNNs).**
- **AlexNet (2012): Won ImageNet competition, massive jump in accuracy (Figure 6.11).**
- **CNN pipeline:**
  - Convolution + pooling layers extract hierarchical features.
  - Fully connected layers → classification.
- **Improvements:**
  - Deeper networks: VGG , ResNet
  - Efficient training with GPUs + large datasets.
  - Transfer learning: pretrained models used for many tasks.
- **Visualization: Filters and feature maps show how CNNs learn shapes & patterns (Figure 6.14).**
- **Modern networks:**
  - GoogLeNet (Inception)
  - ResNet with skip connections
  - DenseNet, EfficientNet
- **Performance trend: Accuracy steadily improved across ImageNet challenges (Figure 6.18).**

Figure 6.10 *Fine-grained category recognition. (a) The iNaturalist website and app allows citizen scientists to collect and classify images on their phones (Van Horn, Mac Aodha* et al. *2018) © 2018 IEEE. (b) Attributes can be used for fine-grained categorization and zero-shot learning (Lampert, Nickisch, and Harmeling 2014) © 2014 Springer. These images are part of the* Animals with Attributes *dataset.*

Figure 6.13 *Humans can recognize low-resolution faces of familiar people (Sinha, Balas* et al. *2006) © 2006 IEEE.*

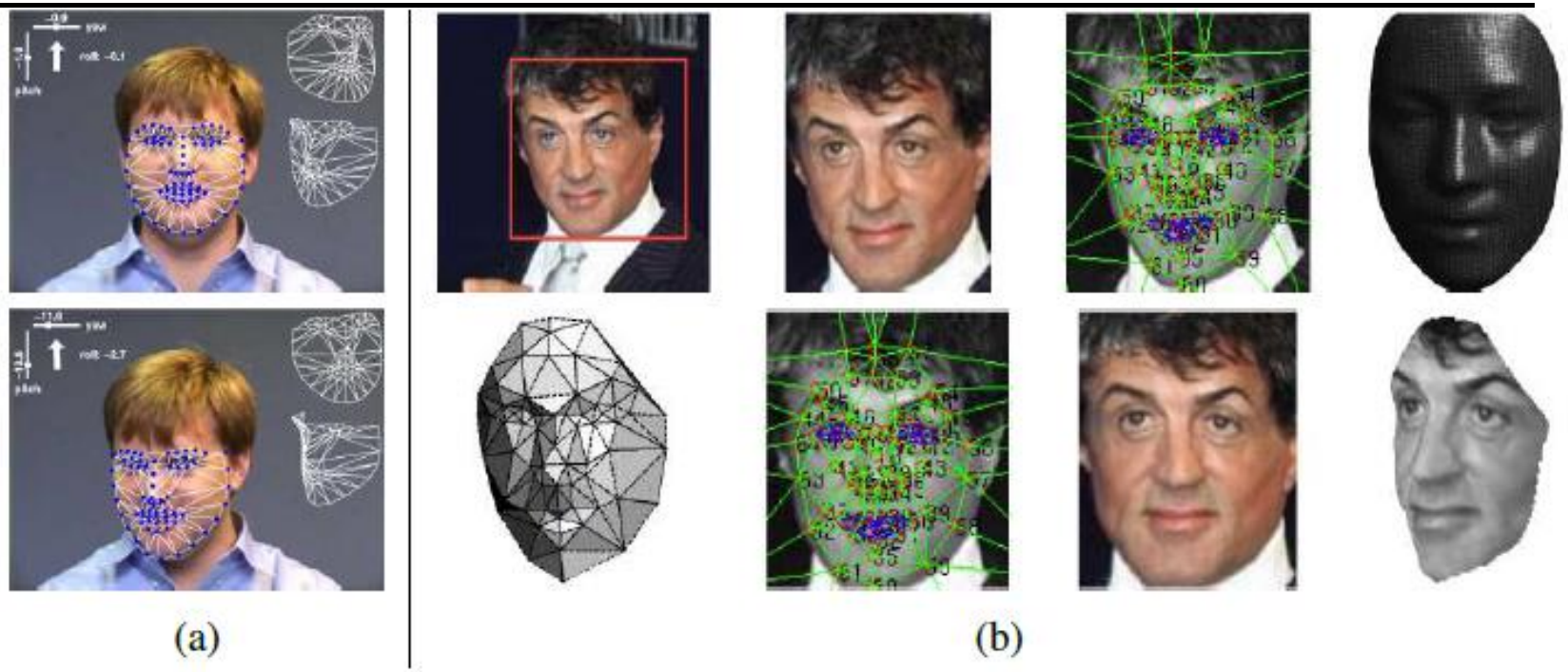


Figure 6.16 *Head tracking and frontalization: (a) using 3D active appearance models (AAMs) (Matthews, Xiao, and Baker 2007) © 2007 Springer, showing video frames along with the estimated yaw, pitch, and roll parameters and the fitted 3D deformable mesh; (b) using six and then 67 fiducial points in the DeepFace system (Taigman, Yang* et al. *2014) © 2014 IEEE, used to frontalize the face image (bottom row).*

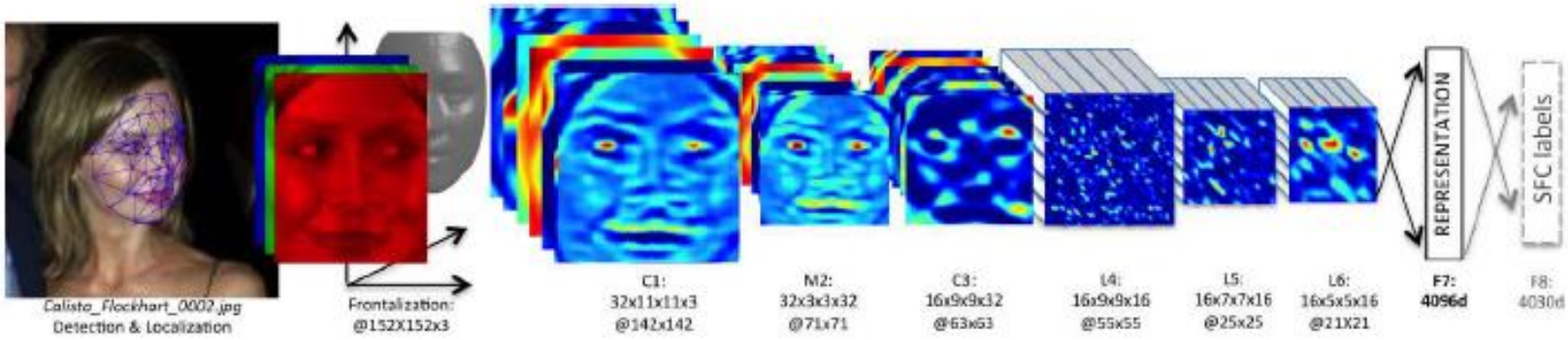


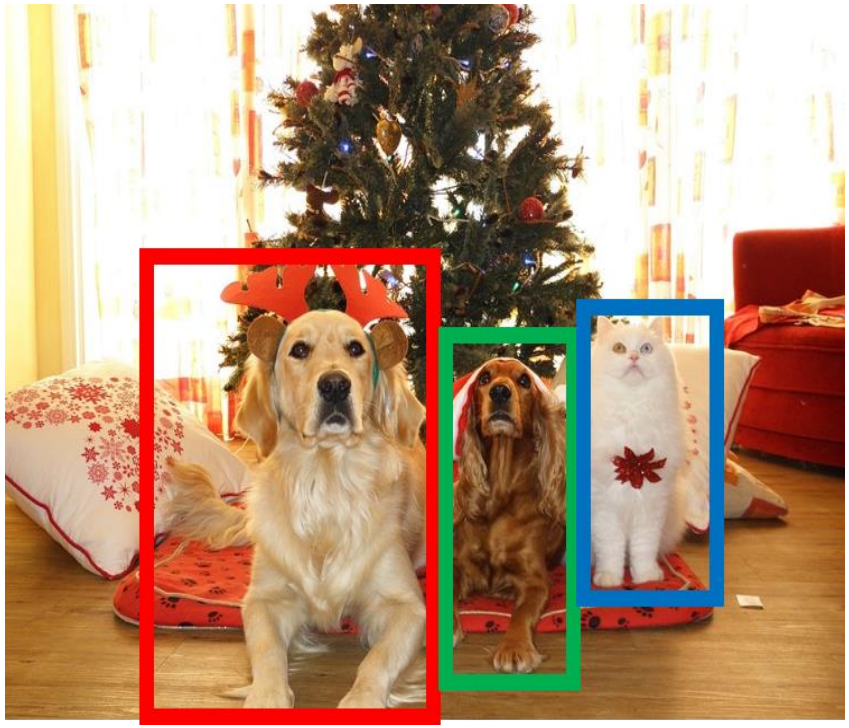
Figure 6.17 *The DeepFace architecture (Taigman, Yang* et al. *2014) © 2014 IEEE, starts with a frontalization stage, followed by several locally connected (non-convolutional) layers, and then two fully connected layers with a K-class softmax.*

Object Detection: Task Definition

Input: Single RGB Image

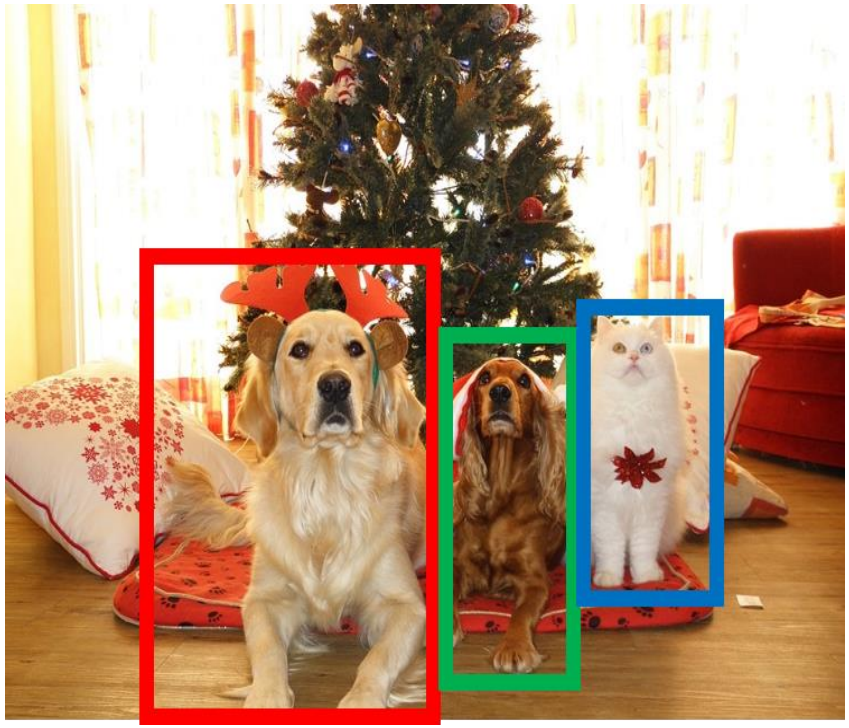
Output: A set of detected objects;
For each object predict:

1. Category label (from fixed, known set of categories)
2. Bounding box (four numbers: x, y, width, height)



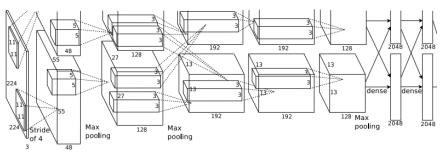
Object Detection: Challenges

- **Multiple outputs**: Need to output variable numbers of objects per image
- **Multiple types of output**: Need to predict ”what” (category label) as well as “where” (bounding box)
- **Large images**: Classification works at 224x224; need higher resolution for detection, often ~800x600



Detecting a single object

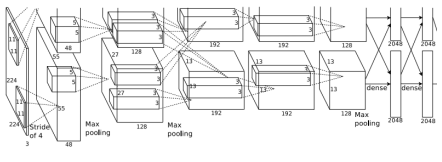
This image is CC0 public domain



Vector:
4096

Detecting a single object "What"

This image is CC0 public domain



Vector:
4096

Fully Connected:
4096 to 1000

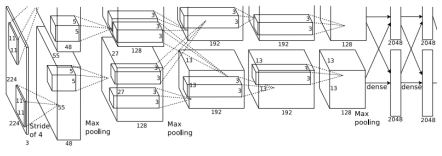
Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label:
Cat

Softmax Loss

Detecting a single object "What"

This image is CC0 public domain



Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

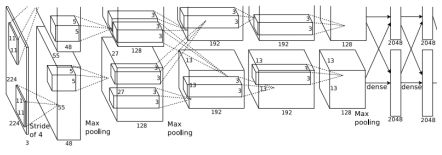
Box Coordinates (x, y, w, h)

L2 Loss

Correct box: (x', y', w', h')

"Where"

Detecting a single object "What"



Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

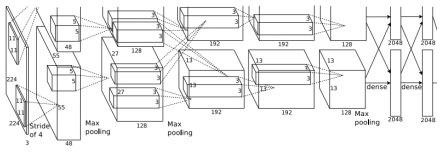
L2 Loss

Correct box: (x', y', w', h')

Weighted Sum → **Loss**

"Where"

Detecting a single object "What"



Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

Softmax Loss

Multitask Loss

Weighted Sum

Loss

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

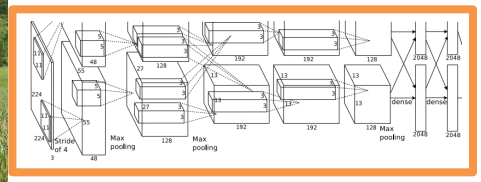
Correct box: (x', y', w', h')

"Where"

Detecting a single object "What"

This image is CC0 public domain

Often pretrained on ImageNet (Transfer learning)



Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

Softmax Loss

Multitask Loss

Weighted Sum

Loss

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

Correct box: (x', y', w', h')

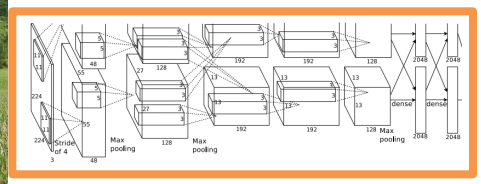
Treat localization as a regression problem!

"Where"

Detecting a single object "What"

This image is CC0 public domain

Often pretrained on ImageNet (Transfer learning)



Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label:
Cat

Softmax Loss

Multitask Loss

Weighted Sum

Loss

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x', y', w', h')

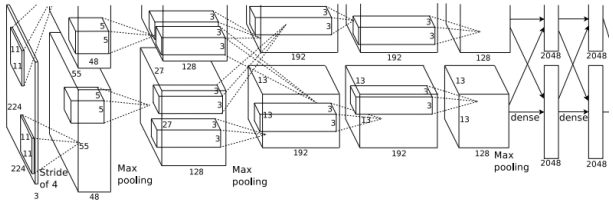
Treat localization as a regression problem!

"Where"

Problem: Images can have more than one object!

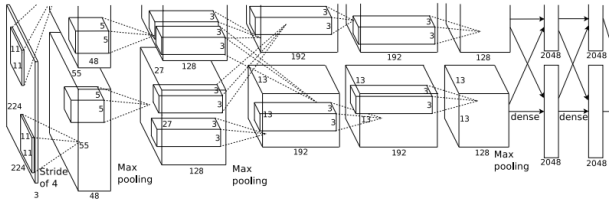
Detecting Multiple Objects

Need different numbers of outputs per image



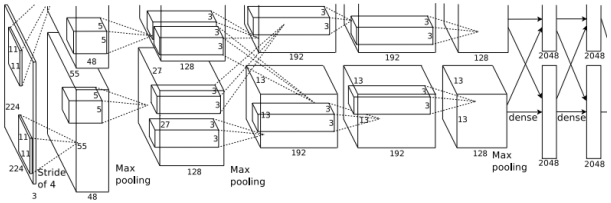
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

16 numbers

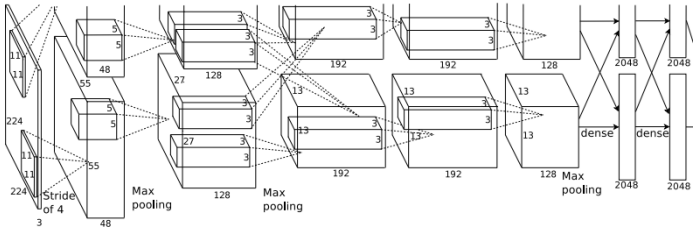


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Detecting Multiple Objects: **Sliding Window**

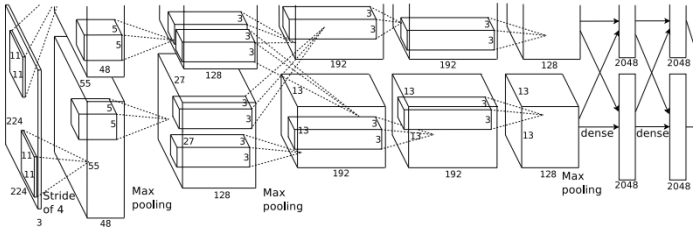
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Detecting Multiple Objects: **Sliding Window**

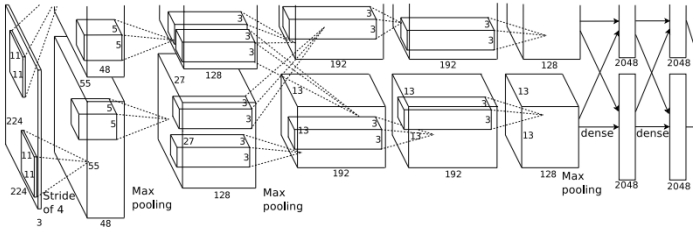
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

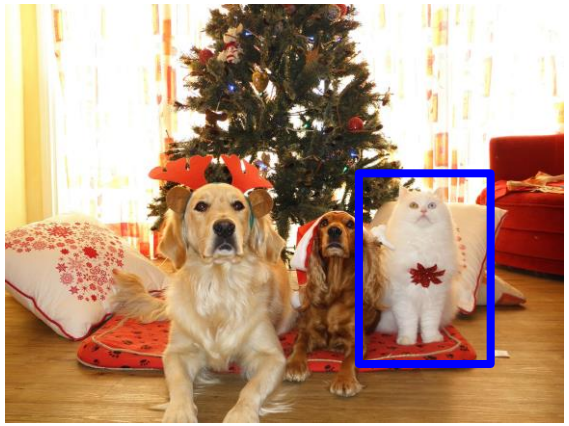
Detecting Multiple Objects: **Sliding Window**

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

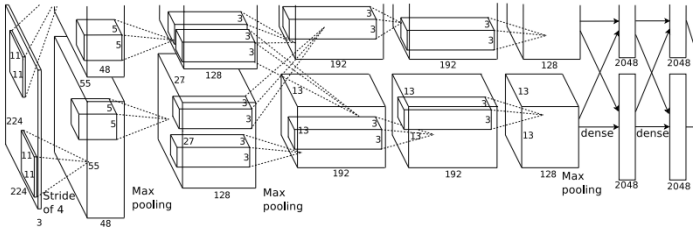


Dog? YES
Cat? NO
Background? NO

Detecting Multiple Objects: **Sliding Window**

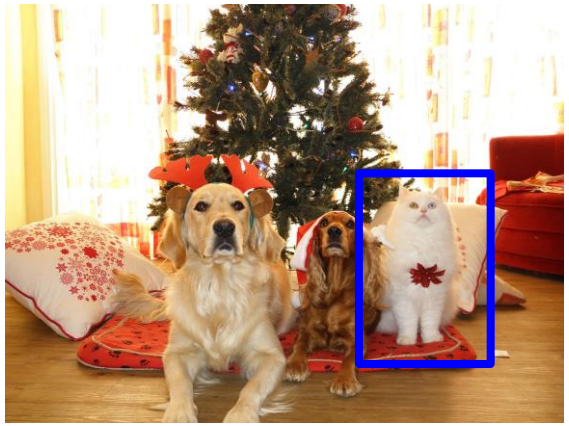


Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

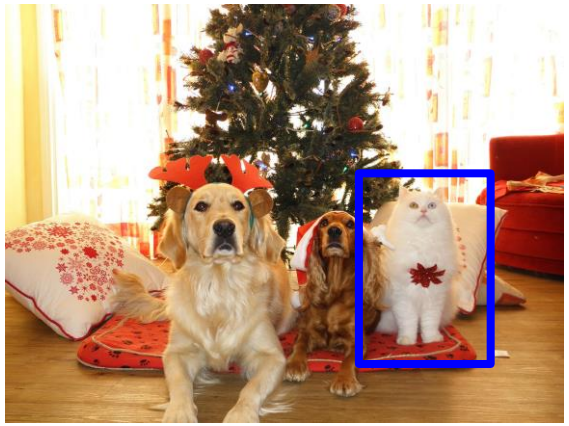
Detecting Multiple Objects: **Sliding Window**



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Question: How many possible boxes are there in an image of size H x W?

Detecting Multiple Objects: **Sliding Window**



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

800 x 600 image has ~58M boxes! No way we can evaluate them all

Question: How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

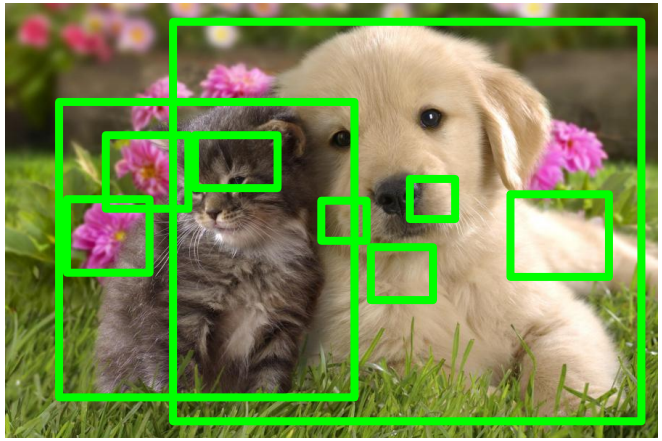
Total possible boxes:

$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1)$$

$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

Region Proposals

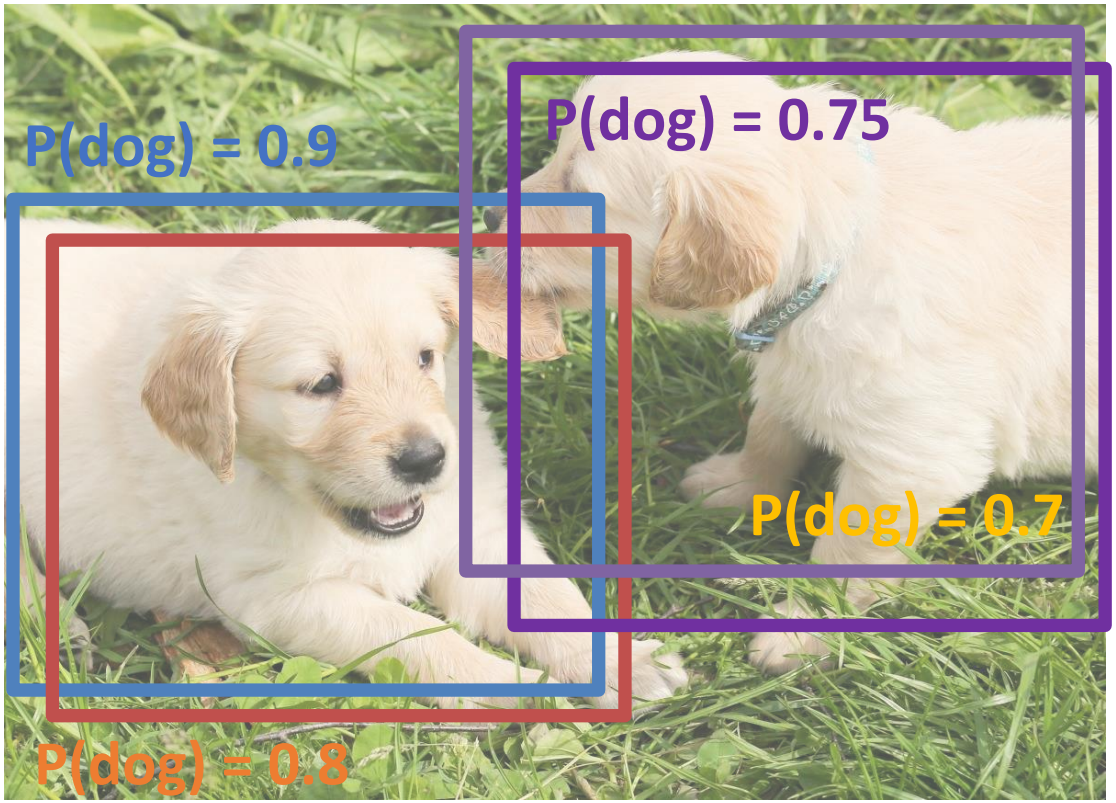
- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for “blob-like” image regions
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

Overlapping Boxes

Problem: Object detectors often output many overlapping detections:

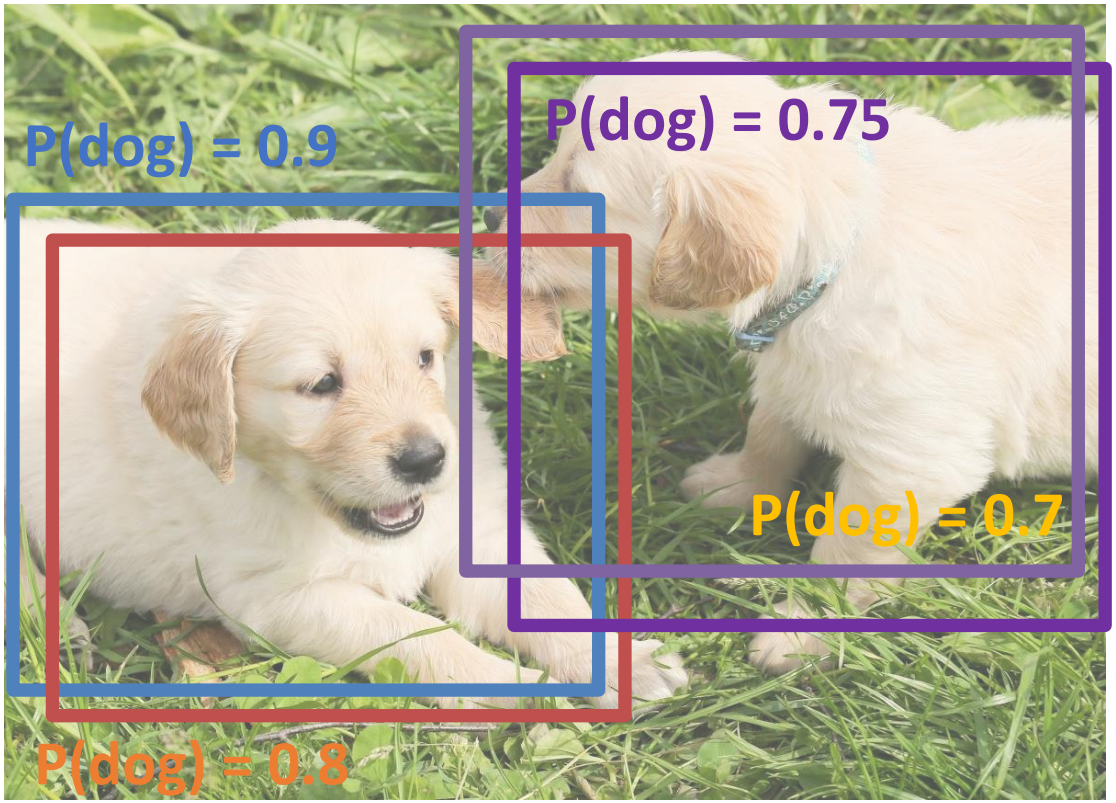


Overlapping Boxes: **Non-Max Suppression (NMS)**

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

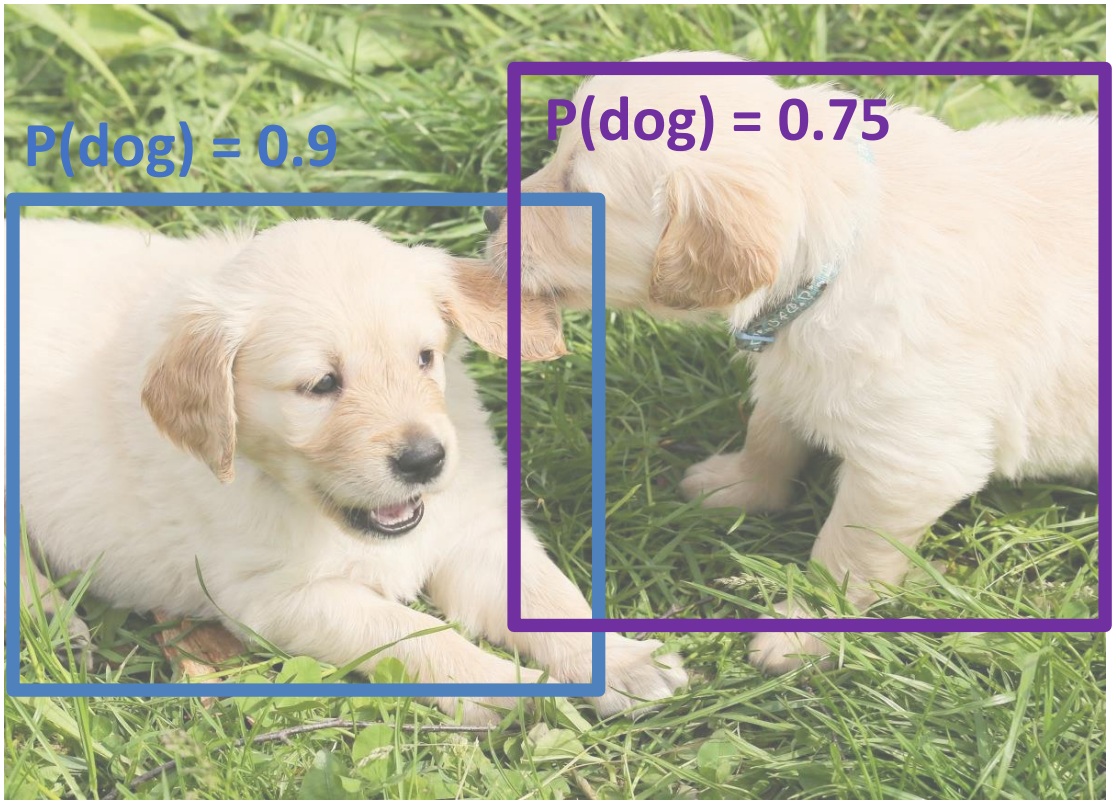


Overlapping Boxes: **Non-Max Suppression (NMS)**

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



Overlapping Boxes: **Non-Max Suppression (NMS)**

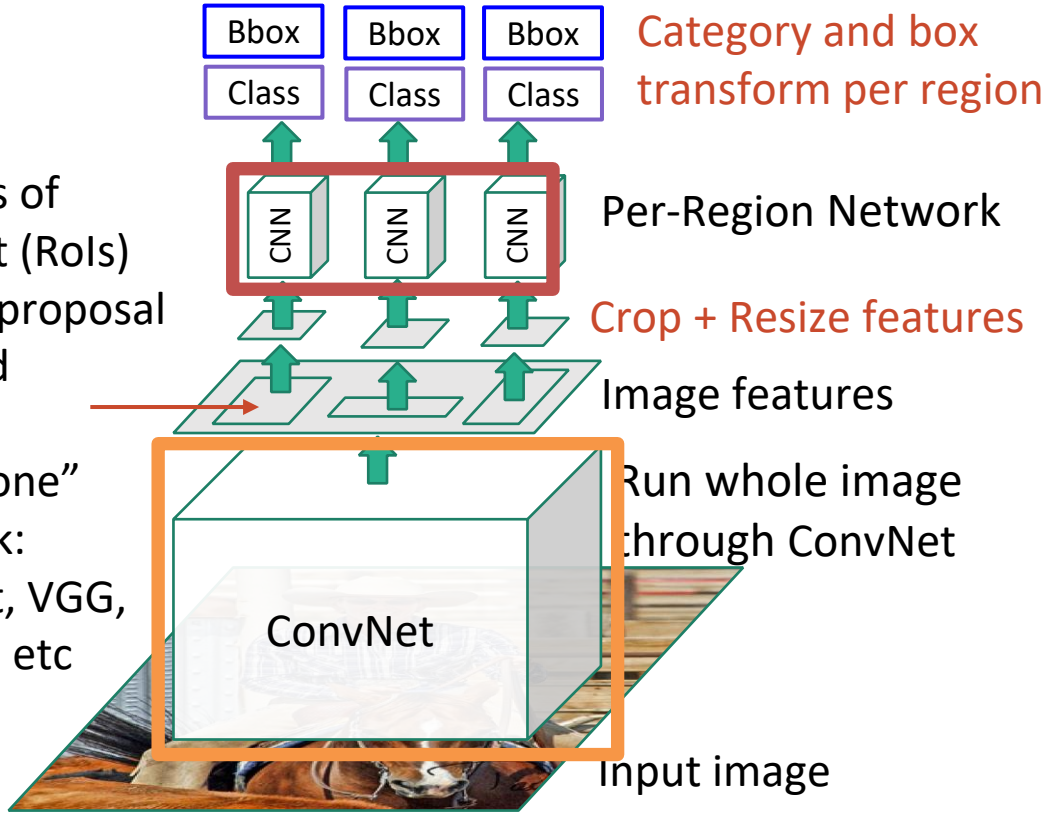
Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

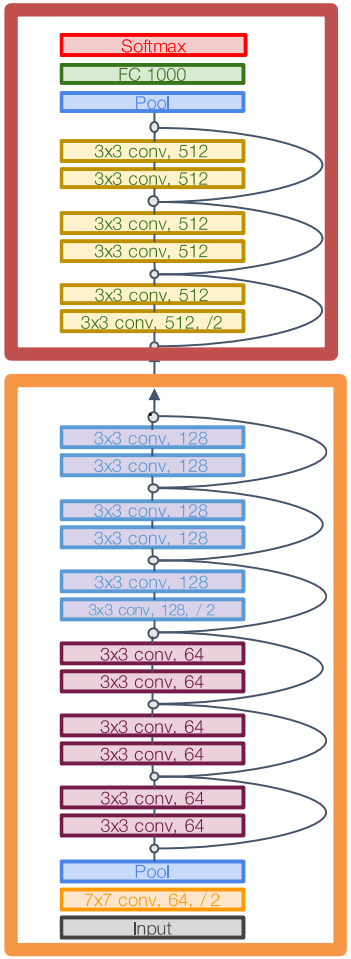
Problem: NMS may eliminate "good" boxes when objects are highly overlapping… no good solution =(

Fast R-CNN



Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc



Example:
For ResNet, last stage is used as per-region network; the rest of the network is used as backbone

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Cropping Features: RoI Align

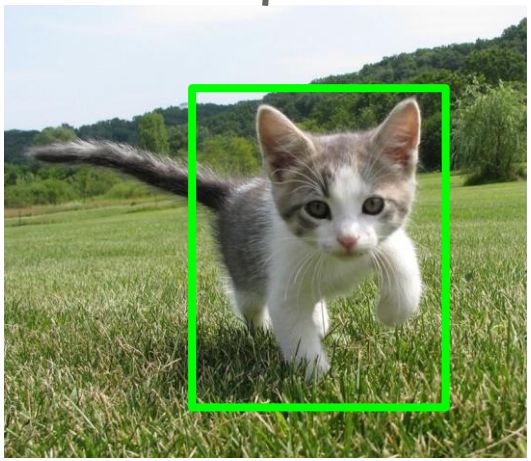
Divide into equal-sized subregions
(may not be aligned to grid!)

No “snapping”!

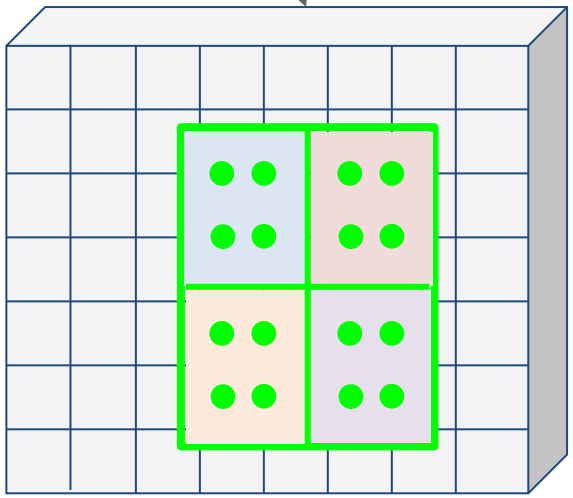
Project proposal
onto features

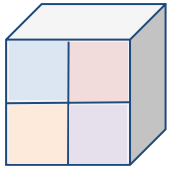
Sample features at
regularly-spaced points
in each subregion using
bilinear interpolation

After sampling, max-
pool in each subregion



CNN





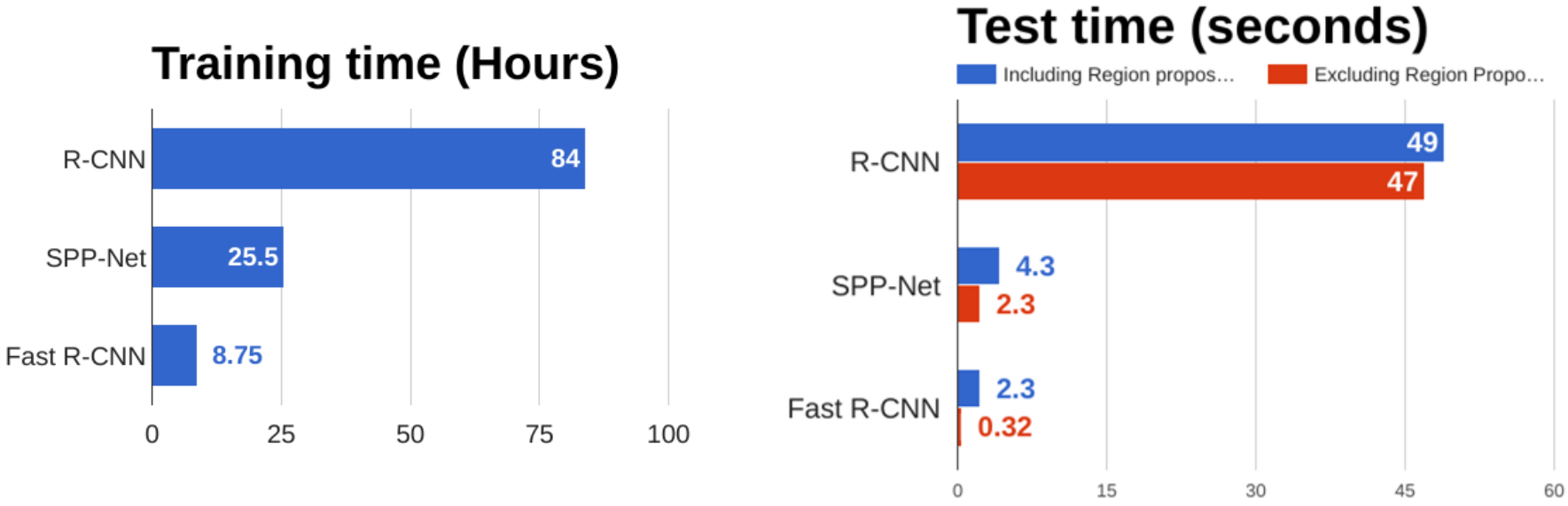
Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

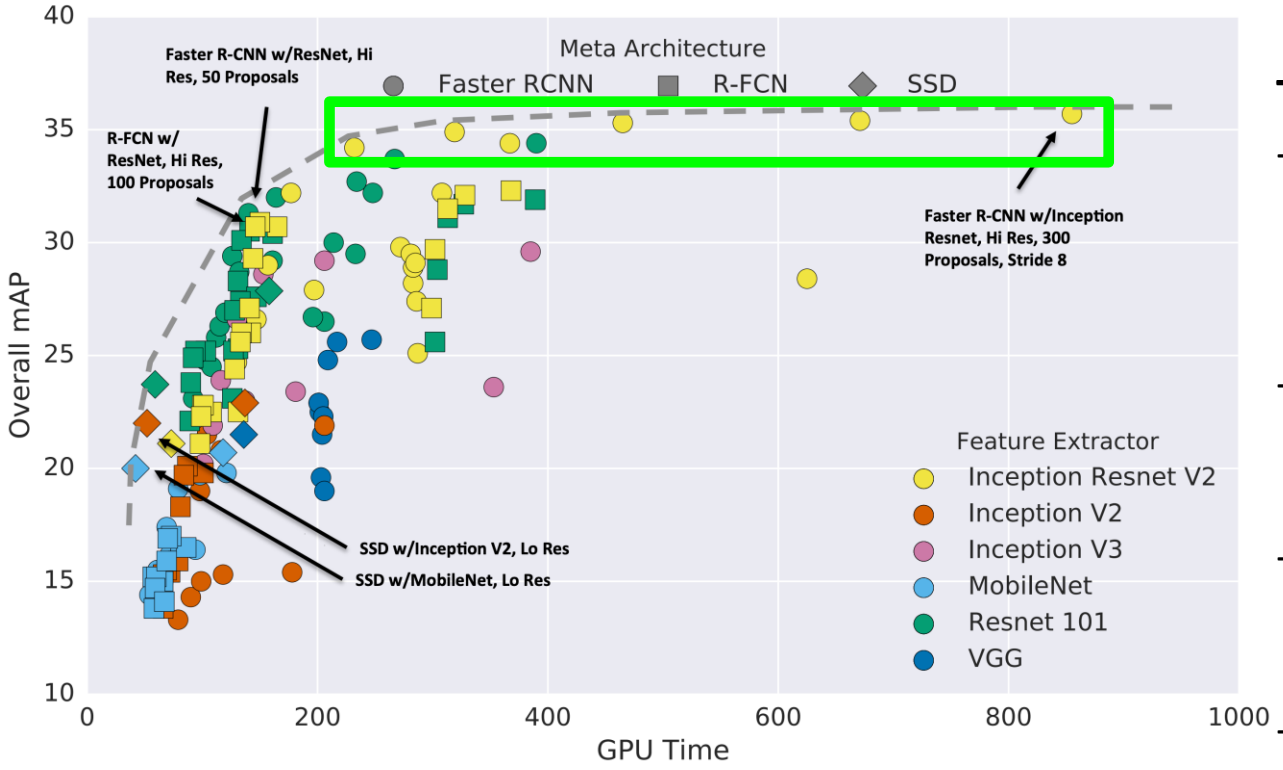
He et al, “Mask R-CNN”, ICCV 2017

Fast R-CNN vs “Slow” R-CNN



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Object Detection: Lots of variables!



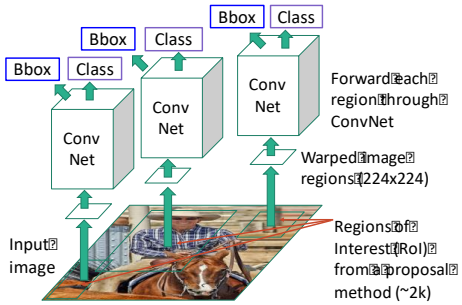
Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Takeaways:

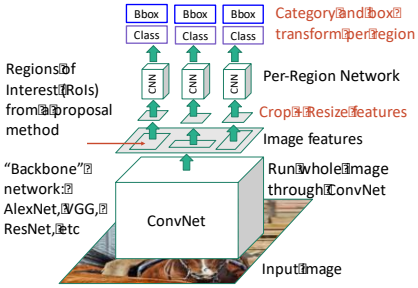
- Two stage method (Faster R-CNN) get the best accuracy, but are slower
- Single-stage methods (SSD) are much faster, but don't perform as well
- Bigger backbones improve performance, but are slower
- Diminishing returns for slower methods

Summary

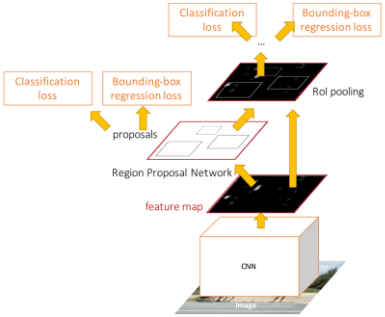
“Slow” R-CNN: Run CNN independently for each region



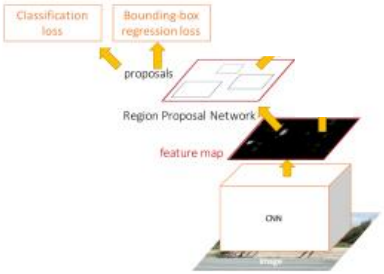
Fast R-CNN: Apply differentiable cropping to shared image features



Faster R-CNN: Compute proposals with CNN



Single-Stage: Fully convolutional detector



Segmentation and detection

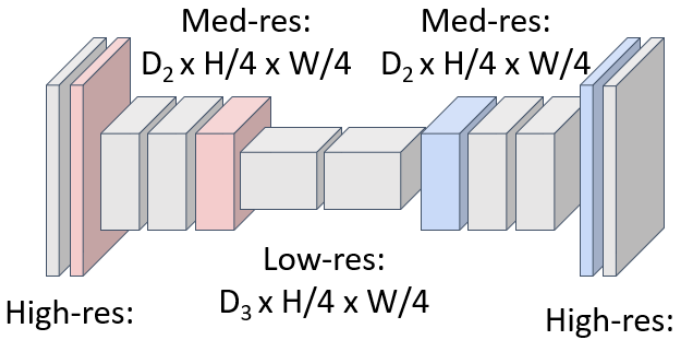
Adversarial examples

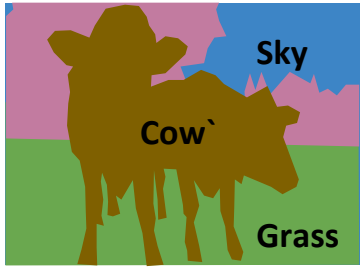
Bottlenecks and U-Nets

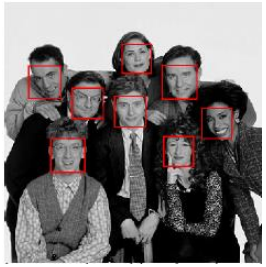
Segmentation

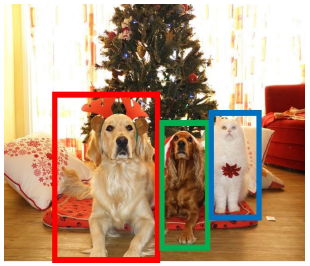
Face detection

Object detection

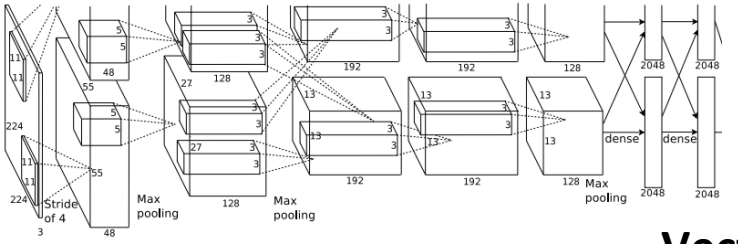








So far: Image Classification



Vector: 4096

Fully-Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

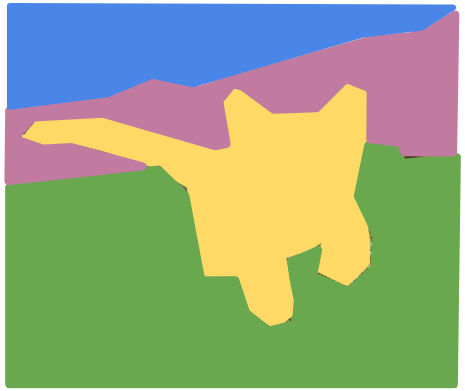
Computer Vision Tasks: Semantic Segmentation

Classification

CAT

No spatial extent

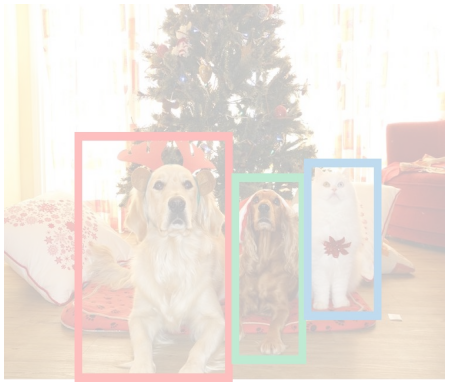
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation

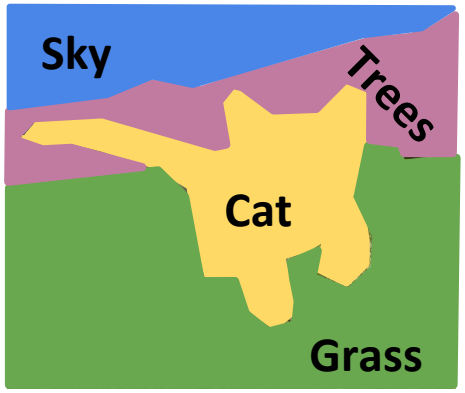
DOG, DOG, CAT

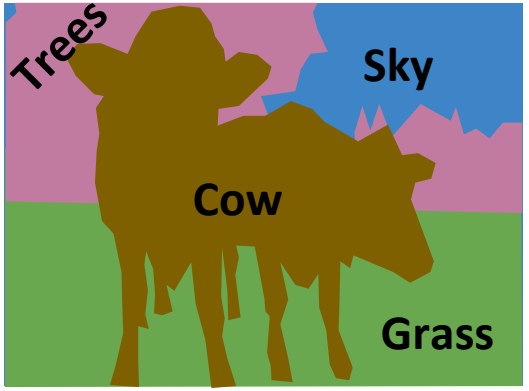
Multiple Objects

Semantic Segmentation

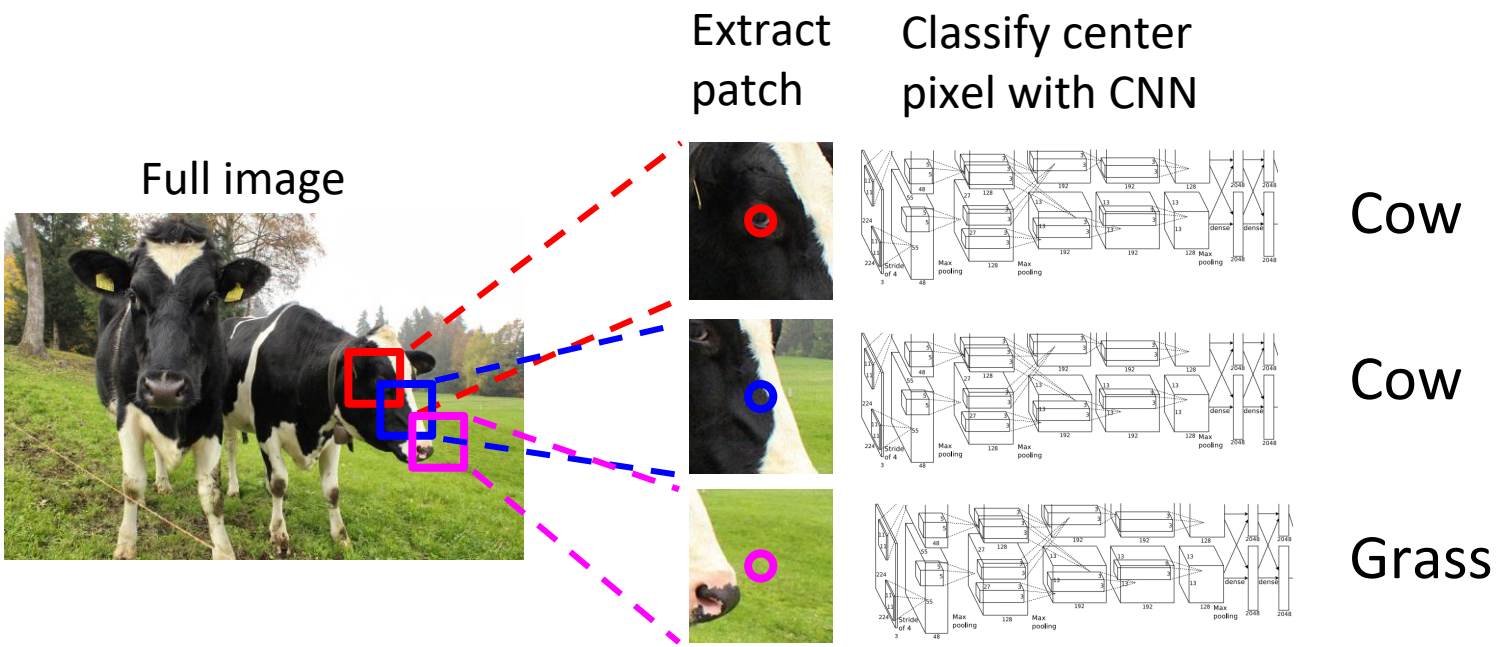
Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels





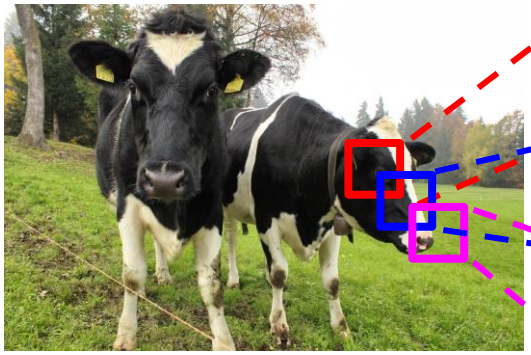
Semantic Segmentation Idea: Sliding Window



Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

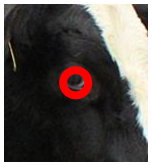
Semantic Segmentation Idea: Sliding Window

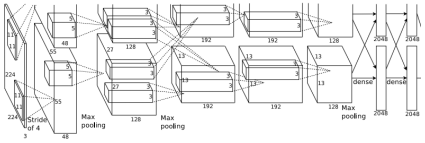
Full image



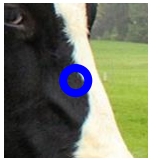
Extract patch

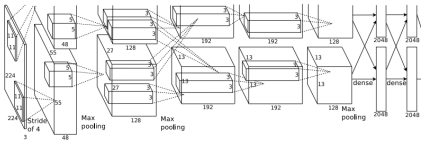
Classify center pixel with CNN



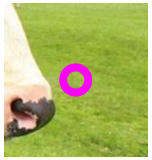


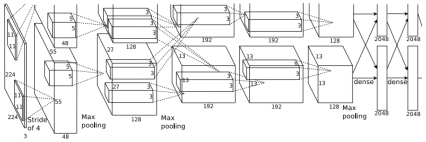
Cow





Cow





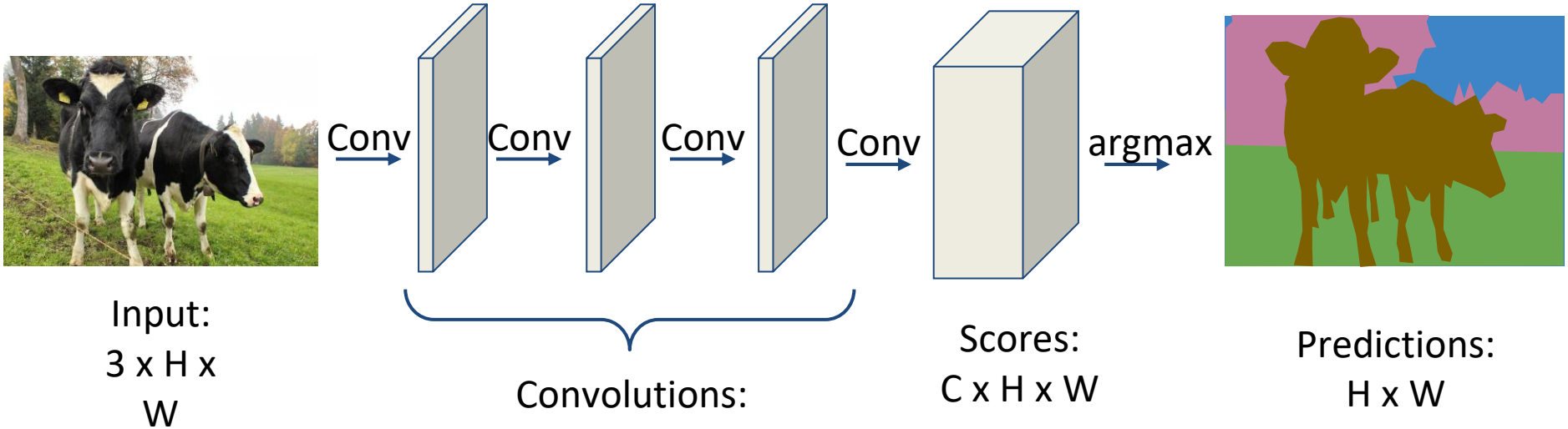
Grass

Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

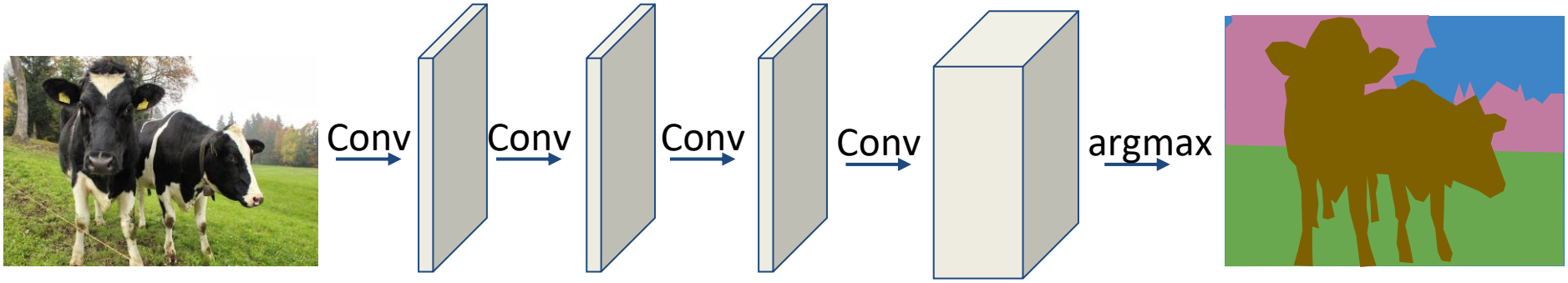


Loss function: Per-Pixel cross-entropy

Long et al, “Fully convolutional networks for semantic segmentation”, CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



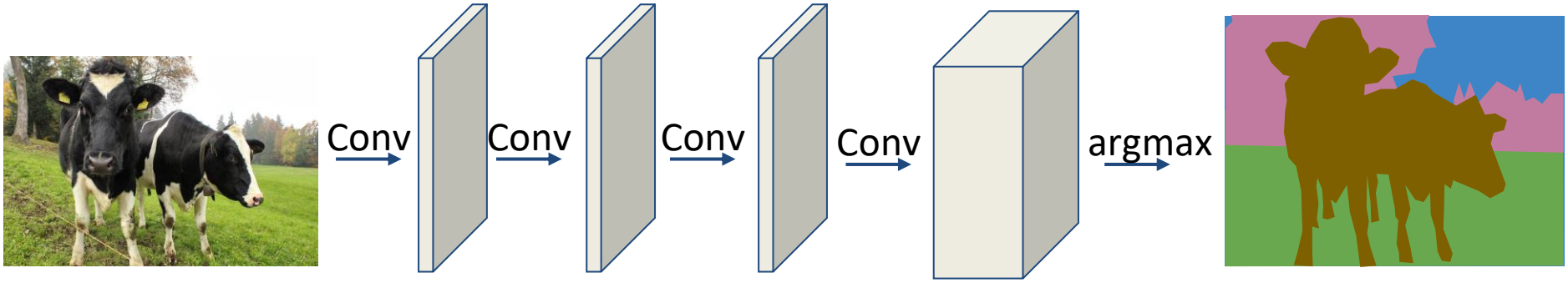
Input: 3 x H x W

Problem #1: Effective receptive field size is linear in number of conv layers: With L 3x3 conv layers, receptive field is 1+2L

Long et al, “Fully convolutional networks for semantic segmentation”, CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Input: 3 x H x W

Problem #1: Effective receptive field size is linear in number of conv layers: With L 3x3 conv layers, receptive field is 1+2L

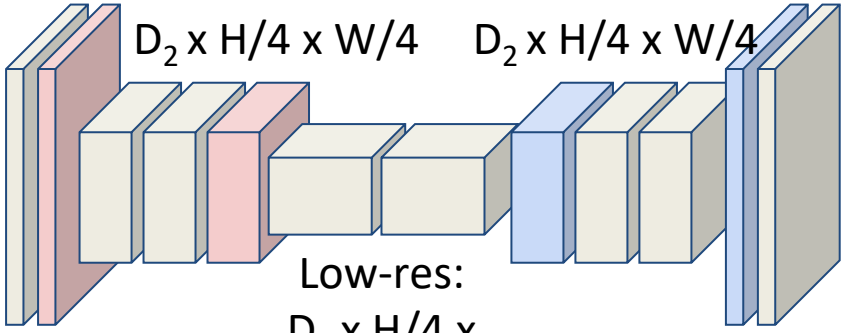
Problem #2: Convolution on high res images is expensive! Recall ResNet stem aggressively downsamples

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

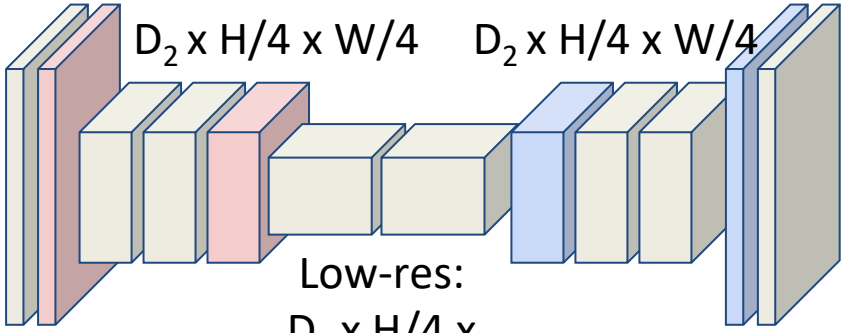
Semantic Segmentation: Fully Convolutional Network

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Downsampling: Pooling, strided convolution

Upsampling: ???

Input: 3 x H x W

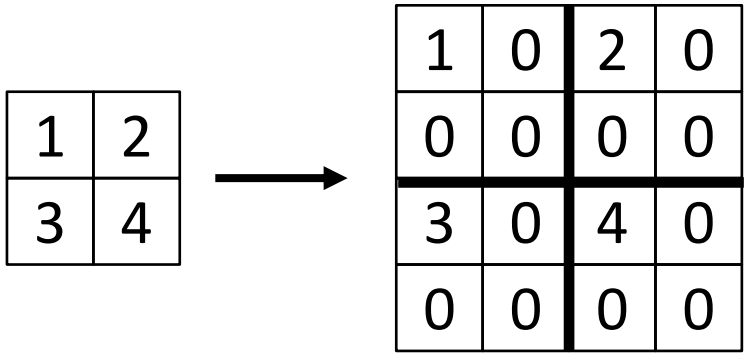


Predictions: H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

In-Network Upsampling: “Unpooling”

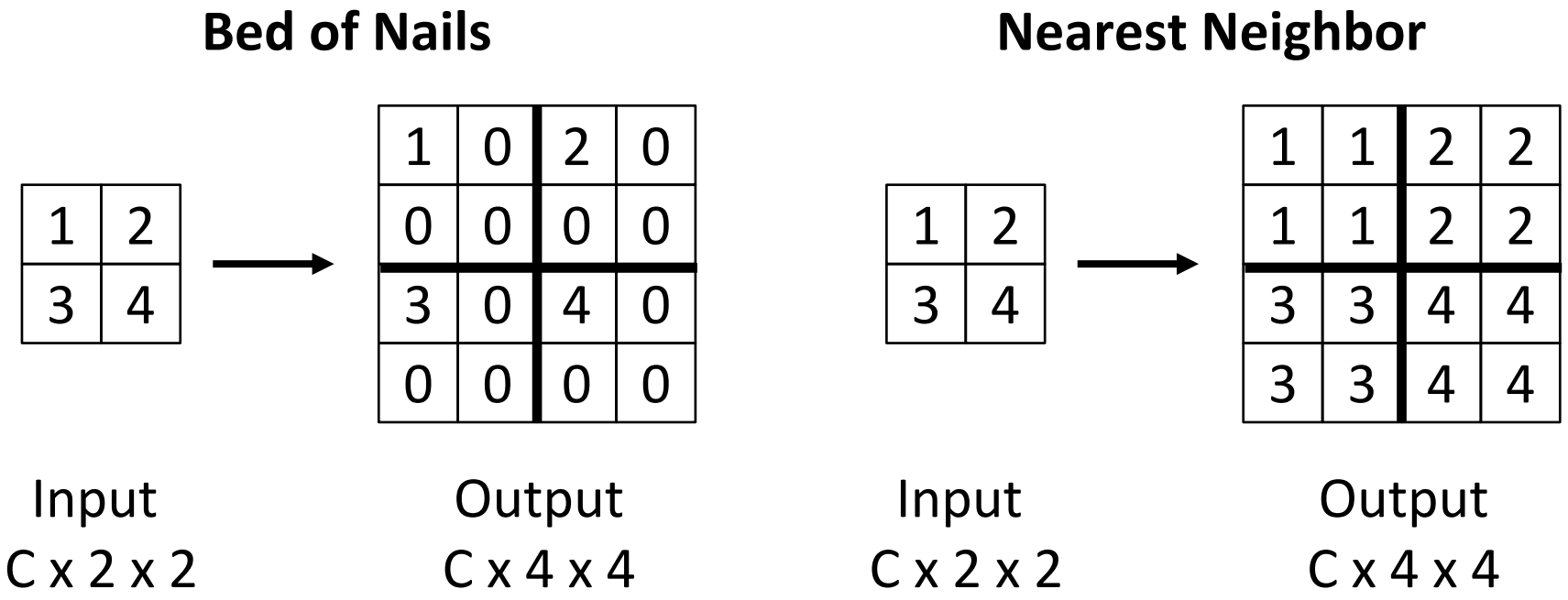
Bed of Nails



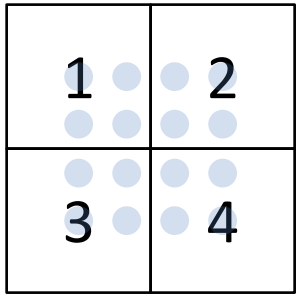
Input
C x 2 x 2

Output
C x 4 x 4

In-Network Upsampling: “Unpooling”



In-Network Upsampling: Bilinear Interpolation



1.00	1.25	1.75	2.00
1.50	1.75	2.25	2.50
2.50	2.75	3.25	3.50
3.00	3.25	3.75	4.00

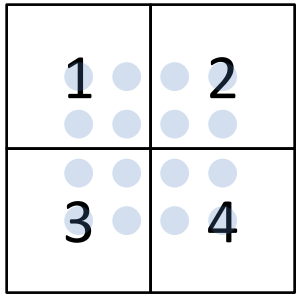
Input: C x 2 x 2

Output: C x 4 x 4

$$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|) \quad i \in \{\lfloor x \rfloor - 1, \ldots, \lceil x \rceil + 1\} \quad j \in \{\lfloor y \rfloor - 1, \ldots, \lceil y \rceil + 1\}$$

Use two closest neighbors in x and y to construct linear approximations

In-Network Upsampling: Bicubic Interpolation



0.68	1.02	1.56	1.89
1.35	1.68	2.23	2.56
2.44	2.77	3.32	3.65
3.11	3.44	3.98	4.32

Input: C x 2 x 2

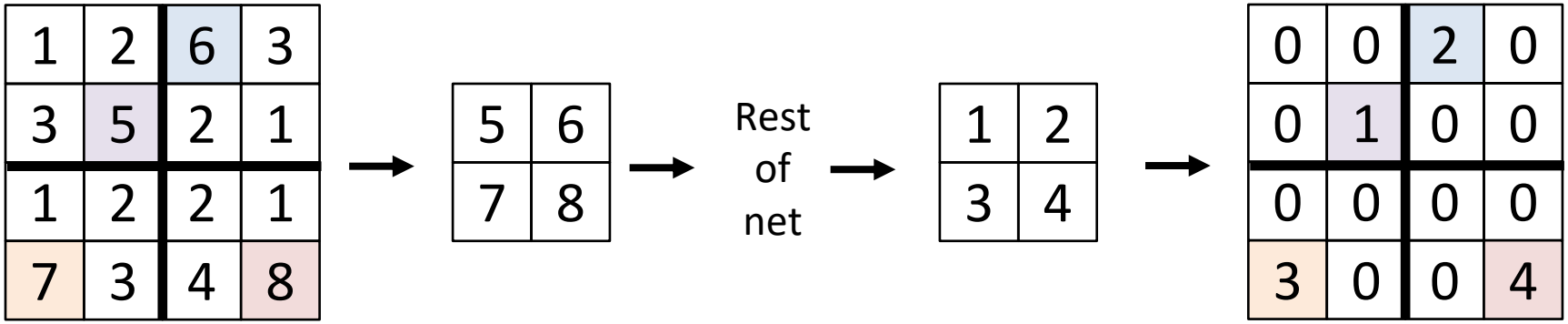
Output: C x 4 x 4

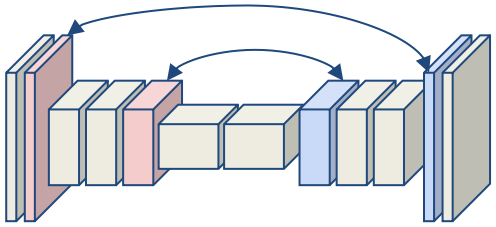
Use **three** closest neighbors in x and y to construct **cubic** approximations
(This is how we normally resize images!)

In-Network Upsampling: “Max Unpooling”

Max Pooling: Remember which position had the max

Max Unpooling: Place into remembered positions



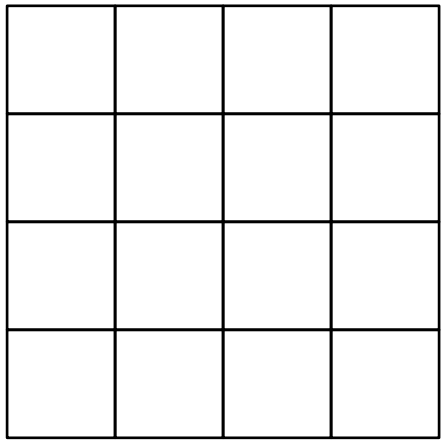


Pair each downsampling layer with an upsampling layer

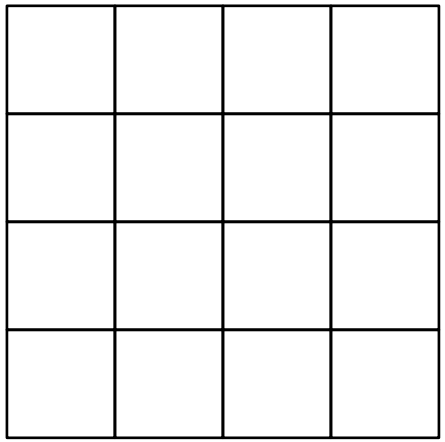
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 1, pad 1



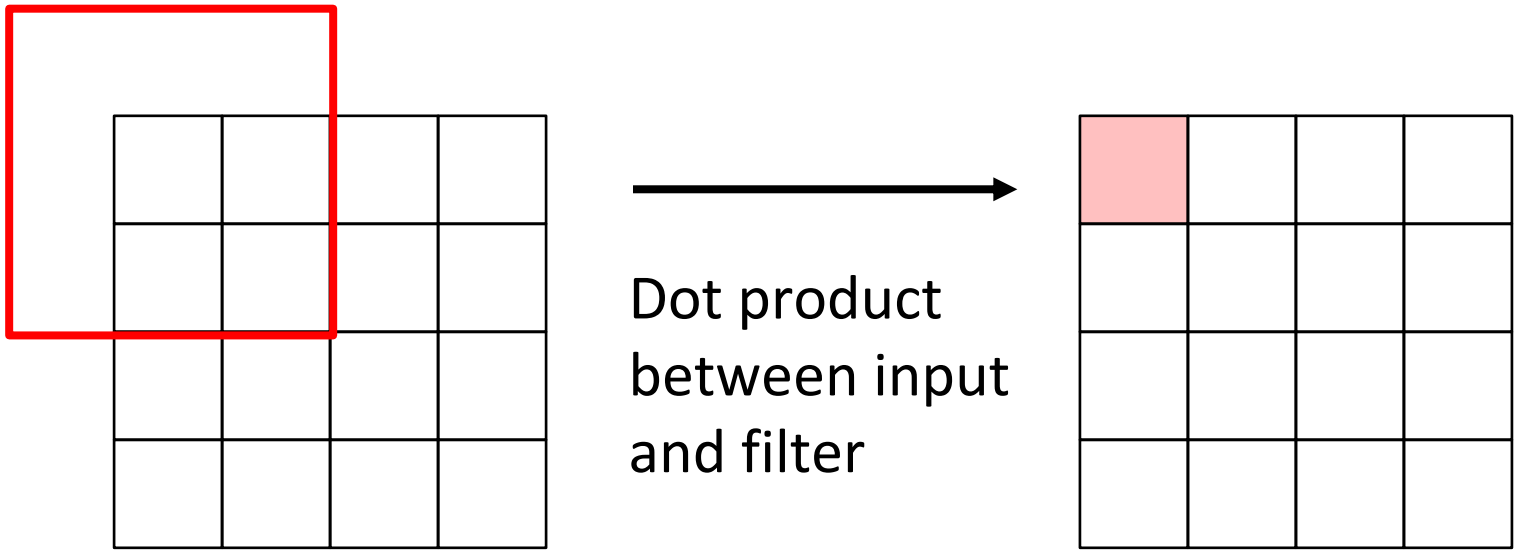
Input: 4 x 4



Output: 4 x 4

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 1, pad 1

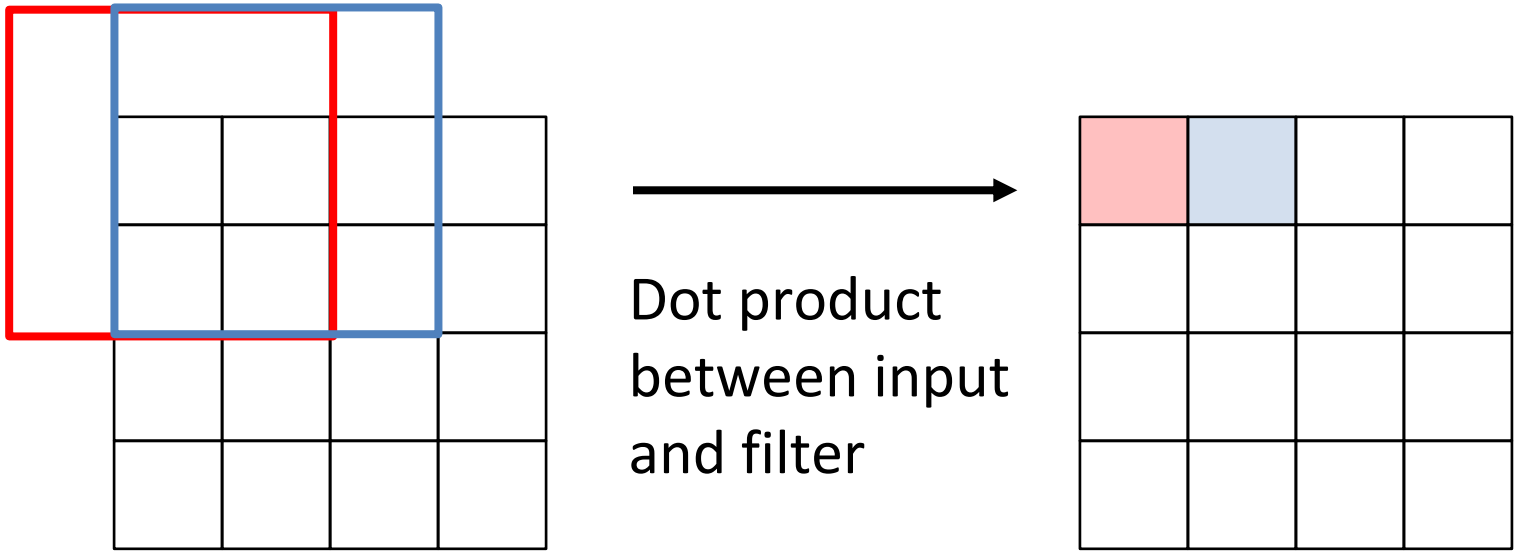


Input: 4 x 4

Output: 4 x 4

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 1, pad 1

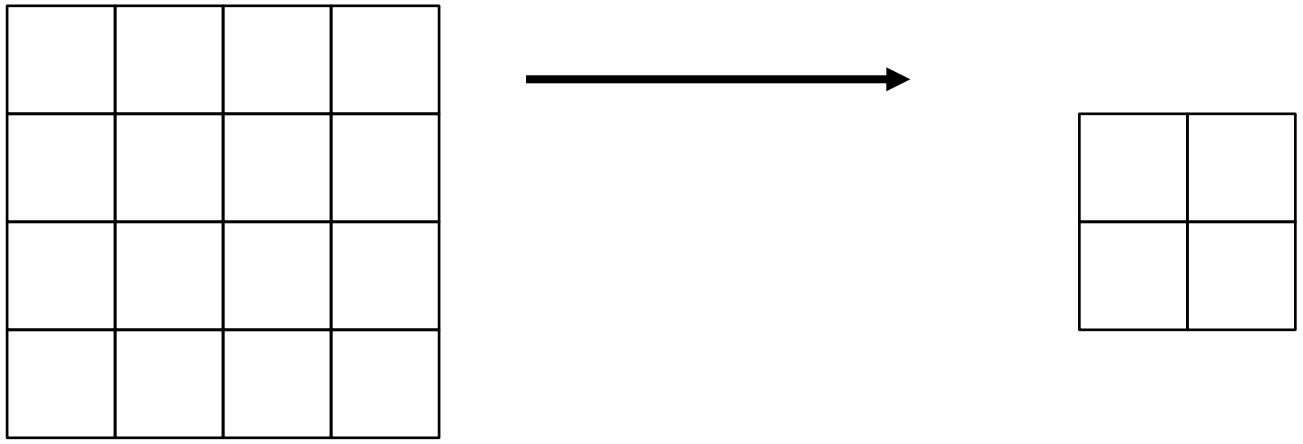


Input: 4 x 4

Output: 4 x 4

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2, pad 1

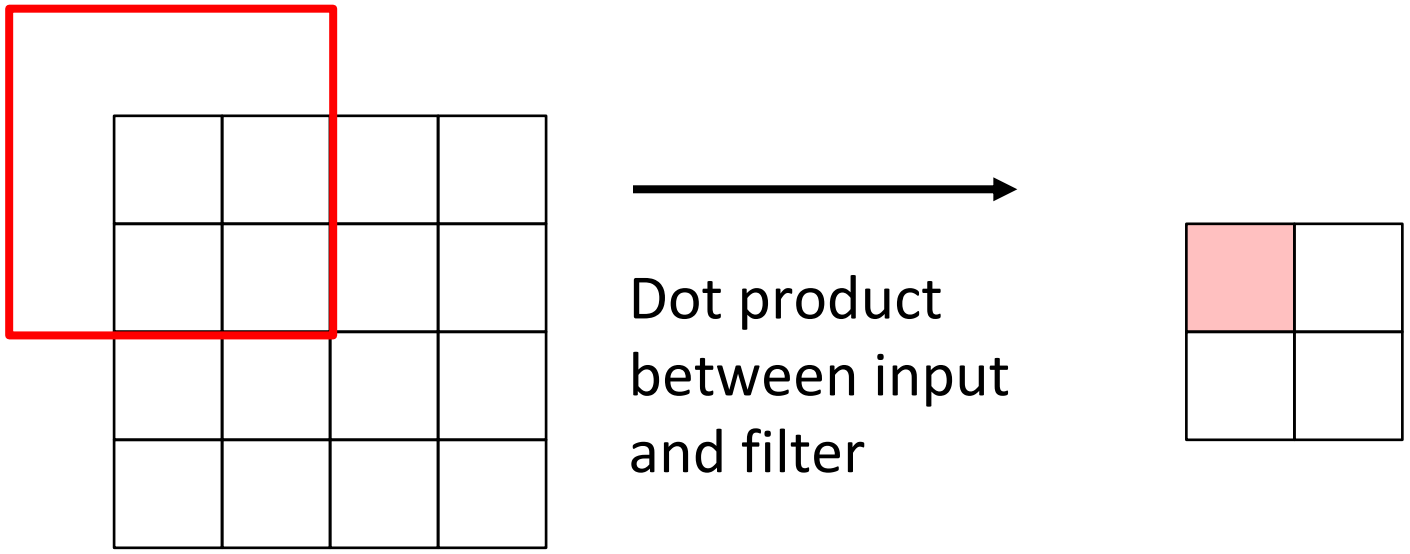


Input: 4 x 4

Output: 2 x 2

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2, pad 1

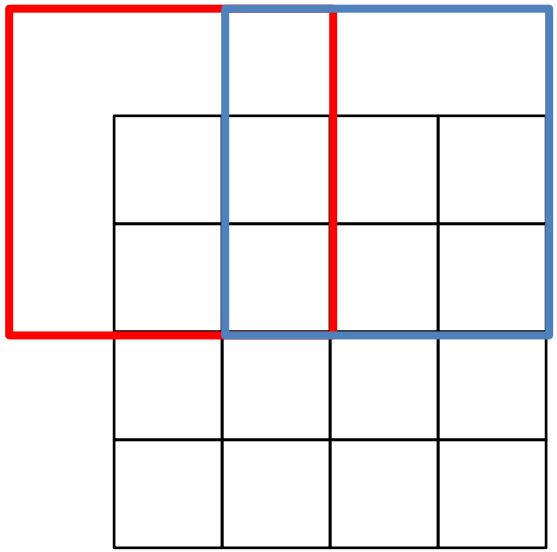


Input: 4 x 4

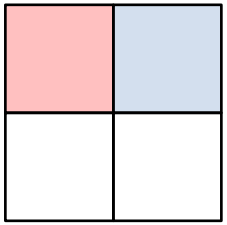
Output: 2 x 2

Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2, pad 1



Dot product between input and filter

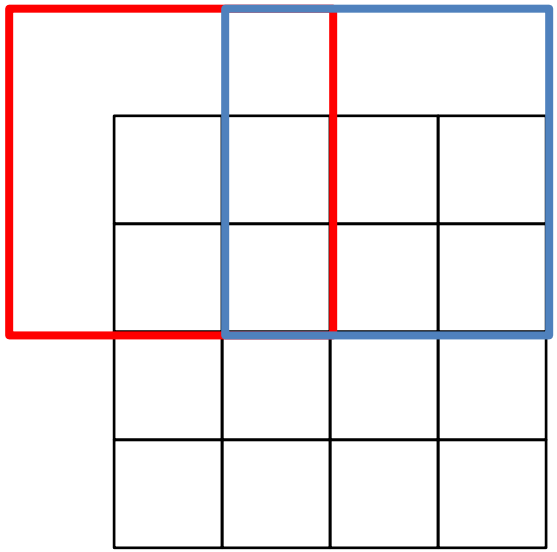


Input: 4 x 4

Output: 2 x 2

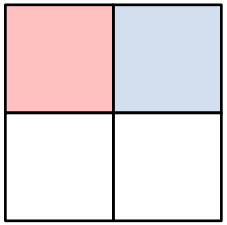
Learnable Upsampling: Transposed Convolution

Recall: Normal 3 x 3 convolution, stride 2, pad 1



Convolution with stride > 1 is “Learnable Downsampling”
Can we use stride < 1 for “Learnable Upsampling”?

Dot product between input and filter

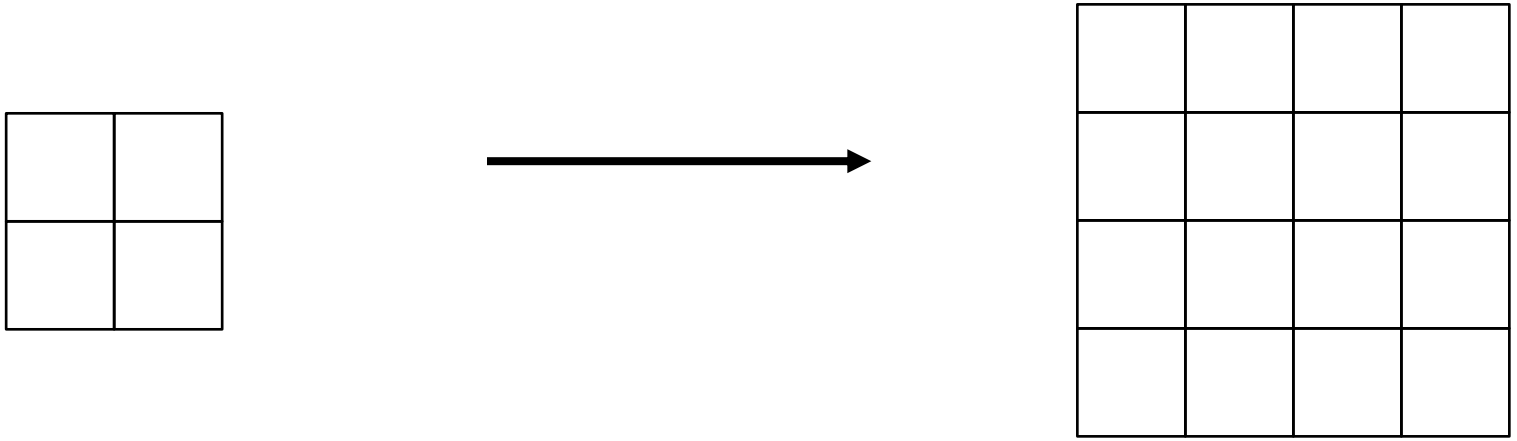


Input: 4 x 4

Output: 2 x 2

Learnable Upsampling: Transposed Convolution

3 x 3 **convolution transpose**, stride 2

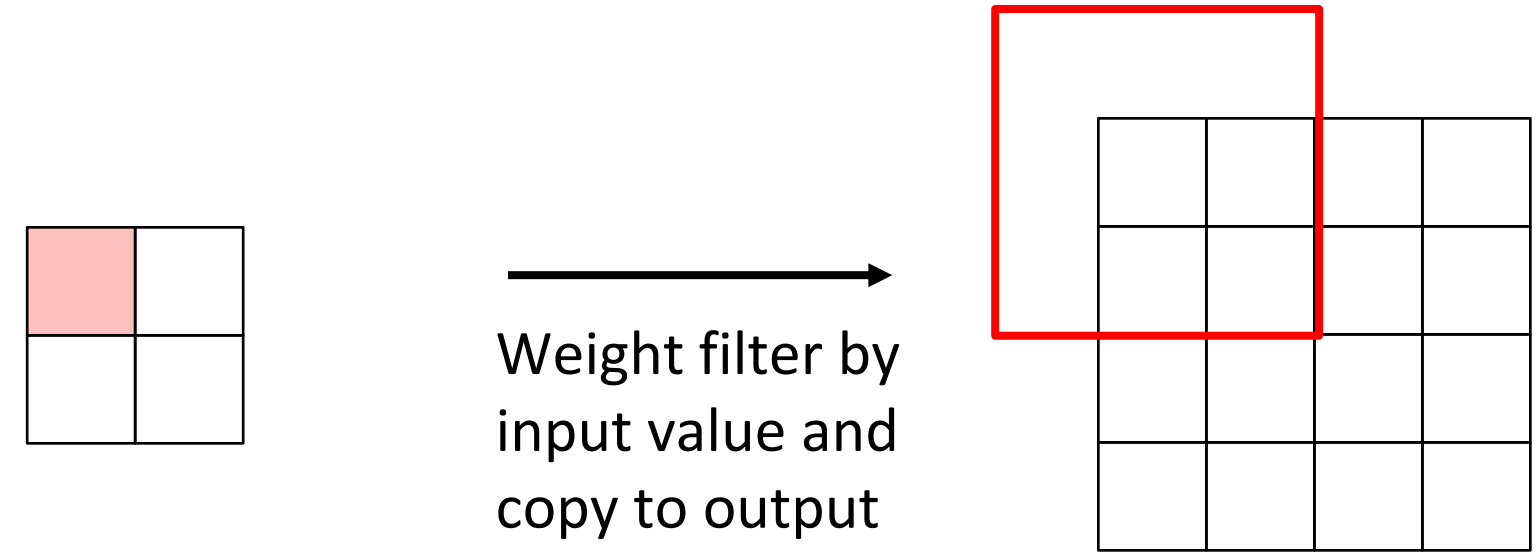


Input: 2 x 2

Output: 4 x 4

Learnable Upsampling: Transposed Convolution

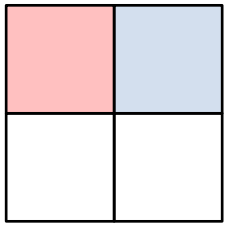
3 x 3 **convolution transpose**, stride 2



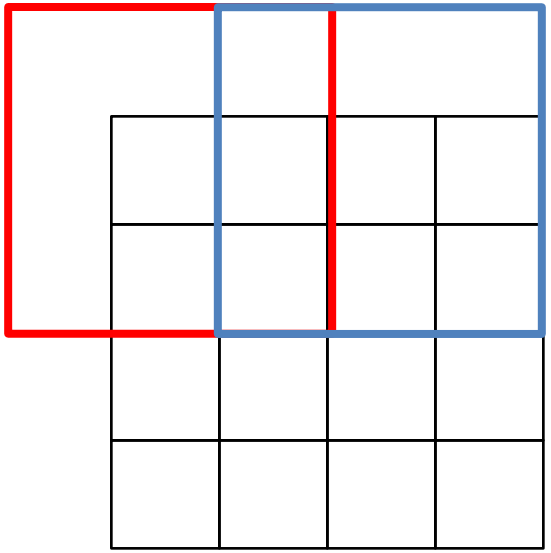
Learnable Upsampling: Transposed Convolution

3 x 3 **convolution transpose**, stride 2

Filter moves 2 pixels in output
for every 1 pixel in input



Weight filter by input value and copy to output



Input: 2 x 2

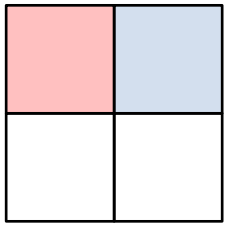
Output: 4 x 4

Learnable Upsampling: Transposed Convolution

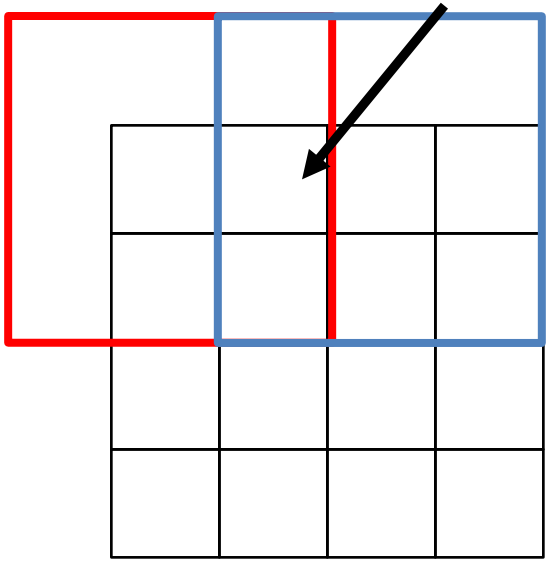
3 x 3 **convolution transpose**, stride 2

Sum where output overlaps

Filter moves 2 pixels in output for every 1 pixel in input



Weight filter by input value and copy to output



Input: 2 x 2

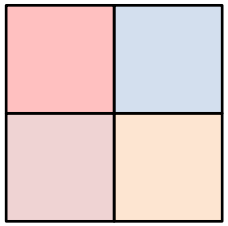
Output: 4 x 4

Learnable Upsampling: Transposed Convolution

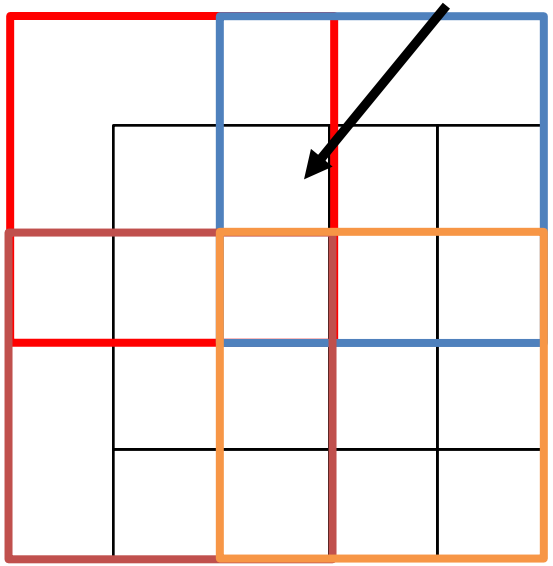
3 x 3 **convolution transpose**, stride 2

Sum where output overlaps

This gives 5x5 output – need to trim one pixel from top and left to give 4x4 output



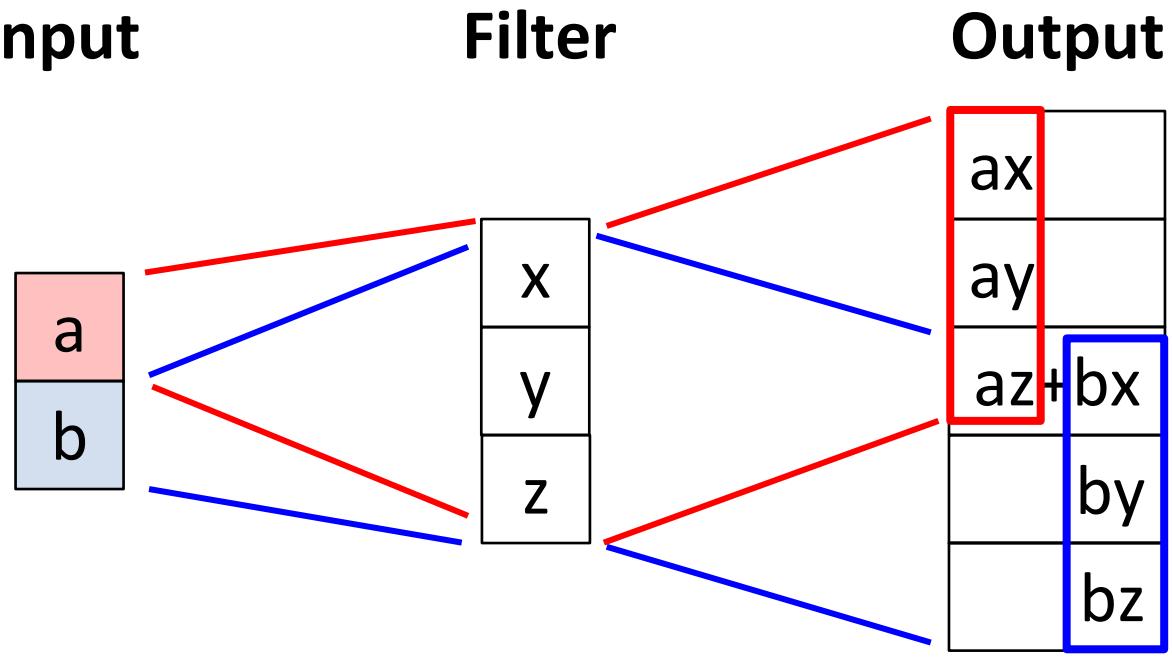
Weight filter by input value and copy to output



Input: 2 x 2

Output: 4 x 4

Transposed Convolution: 1D example

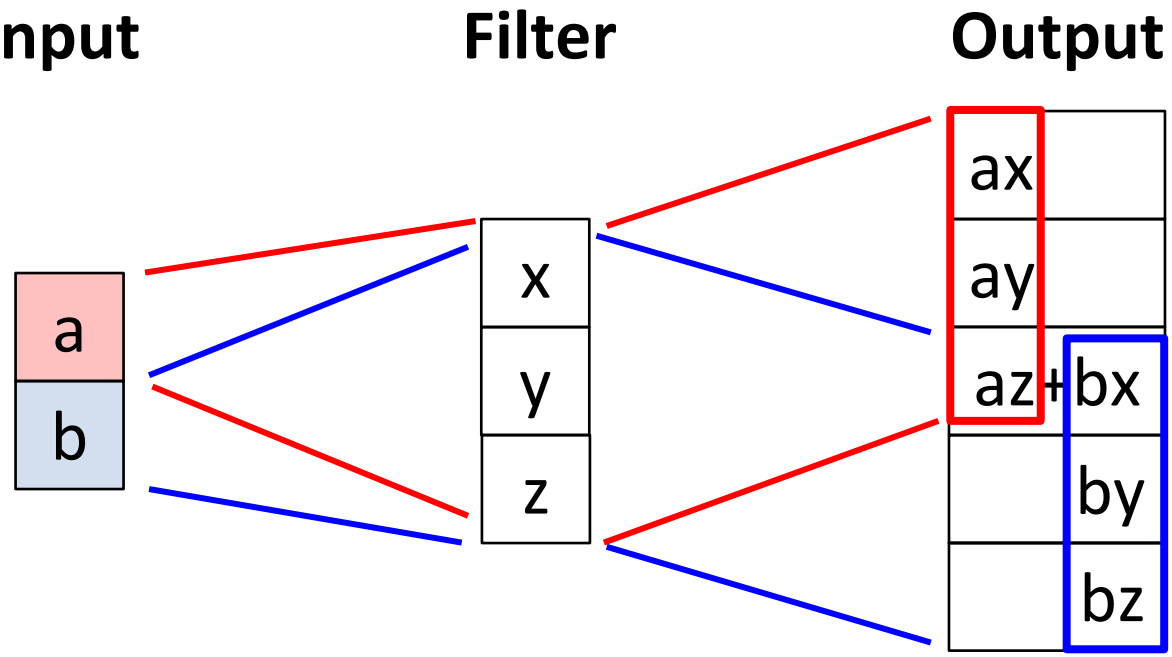


Output has copies of filter weighted by input

Stride 2: Move 2 pixels output for each pixel in input

Sum at overlaps

Transposed Convolution: 1D example



This has many names:

- ~~Deconvolution (bad)!~~
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution
- Transposed Convolution (best name)

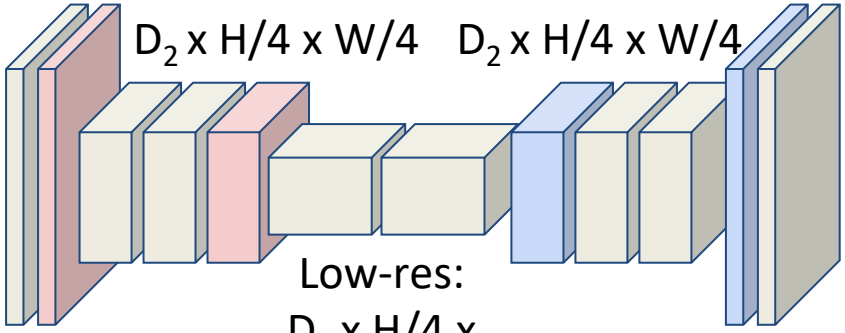
Semantic Segmentation: Fully Convolutional Network

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Downsampling: Pooling, strided convolution

Upsampling: linterpolation, transposed conv

Input: 3 x H x W



Predictions: H x W

Loss function: Per-Pixel cross-entropy

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

U-Nets

Skip connections between encoding and decoding stages
Widely used in image denoising, inpainting, flow, stereo, …

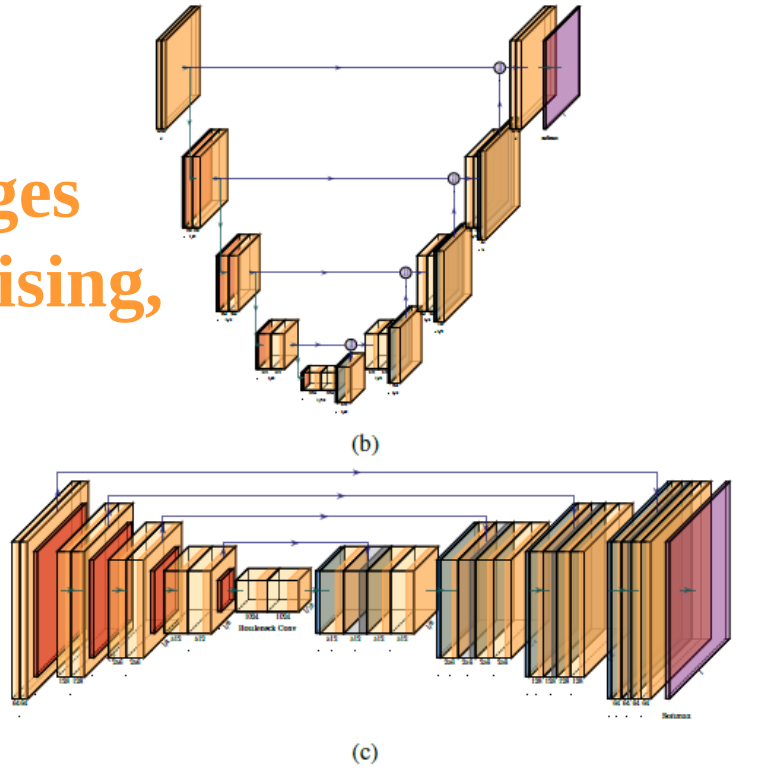
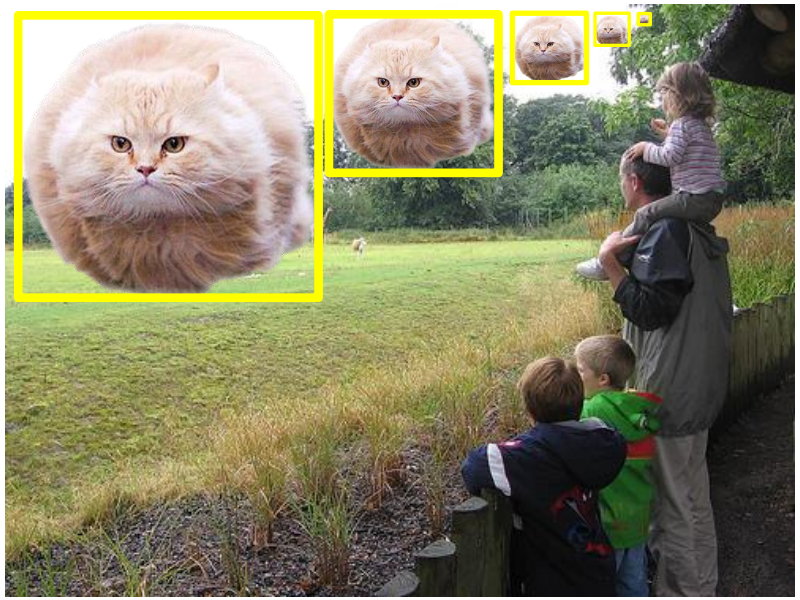
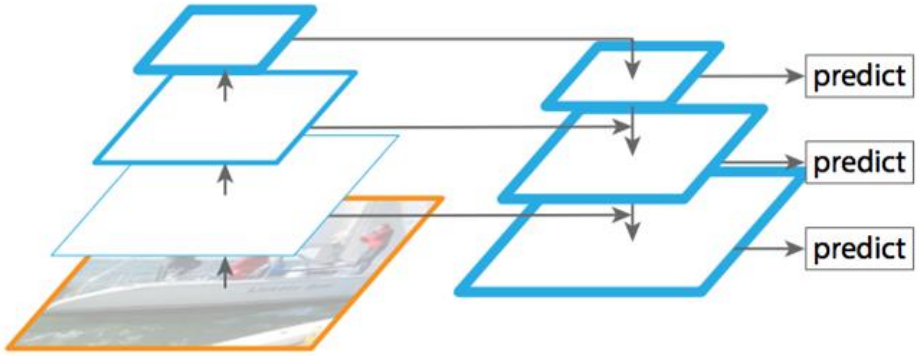


Figure 5.41 *(a) The deconvolution network of Noh, Hong, and Han (2015) © 2015 IEEE and (b–c) the U-Net of Ronneberger, Fischer, and Brox (2015), redrawn using the PlotNeuralNet LaTeX package by Matt Dietke. In addition to the fine-to-coarse-to-fine bottleneck used in (a), the U-Net also has skip connections between encoding and decoding layers at the same resolution.*

Feature Pyramid Network



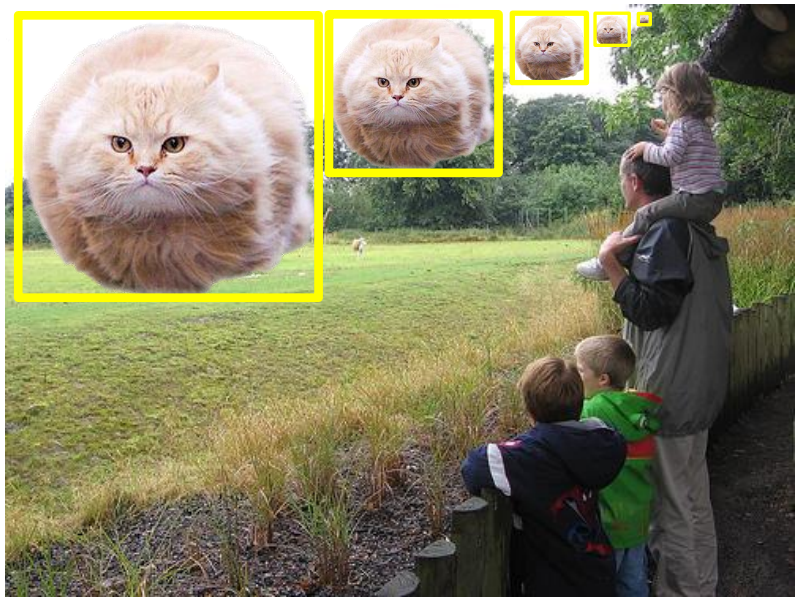


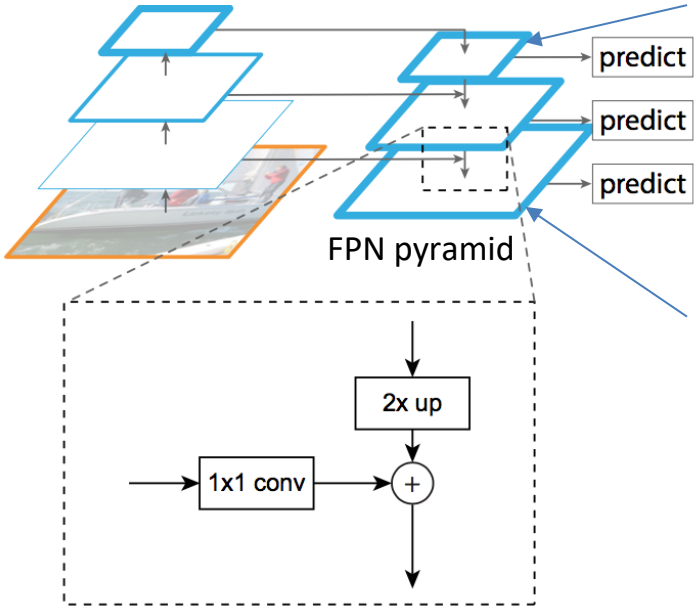
(d) Feature Pyramid Network

Top-down enrichment of high-res features – *fast, less suboptimal*

Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017.

No Compromise on Feature Quality, Still Fast



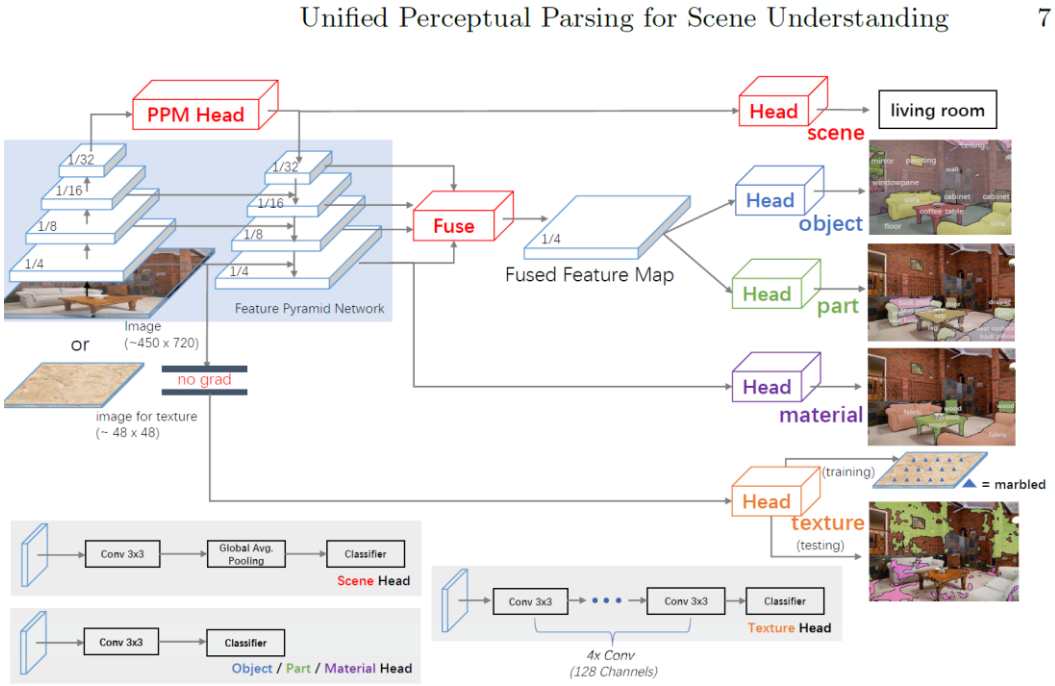


Low resolution,
Strong features

High resolution,
Strong features

Lin et al. Feature Pyramid Networks for Object Detection. CVPR 2017. See also: Shrivastava's TDM.

UPerNet: FPN + fusion



Application: monocular depth estimation

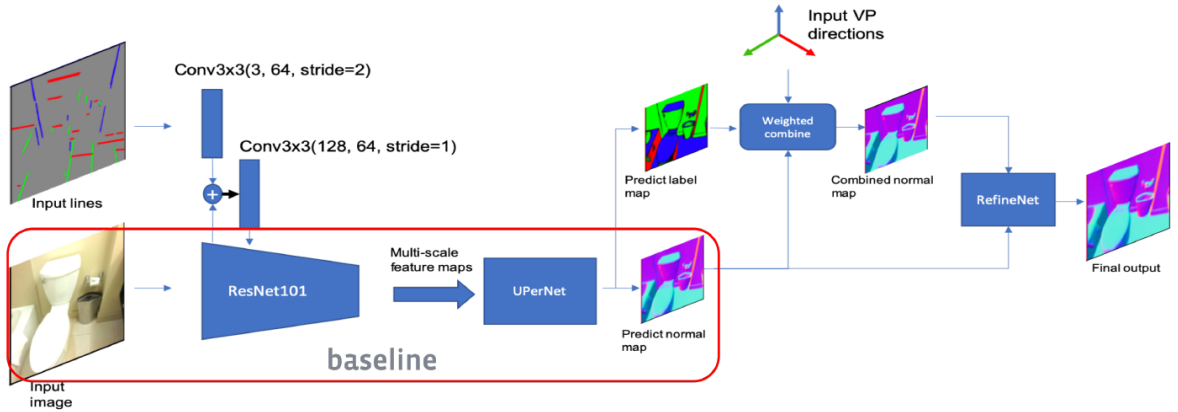


Figure 2. The full pipeline of our proposed model. The network takes an RGB image and a Manhattan line map as input, and produces a Manhattan label map and a raw normal prediction as intermediate output. These intermediate outputs are then combined with the analytically computed dominant vanishing points to generate a "combined normal map". This operation is differentiable. Finally, the combined and raw normal maps are fused through a refinement network to produce the final normal prediction.

6.5 Video Understanding - Definition

Extension of image understanding to video
Detecting and describing human actions
Actions form sequences of activity

Early Work (1990s)

Human activity recognition + motion tracking
Techniques: Point & mesh tracking, spatio-temporal signatures
Survey: Aggarwal & Cai (1999)

Early Work (1990s)

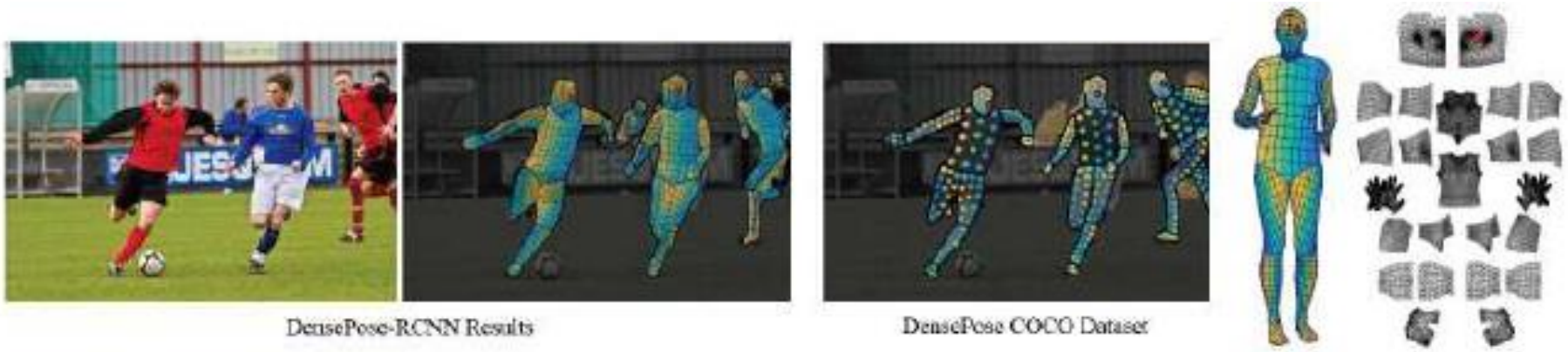
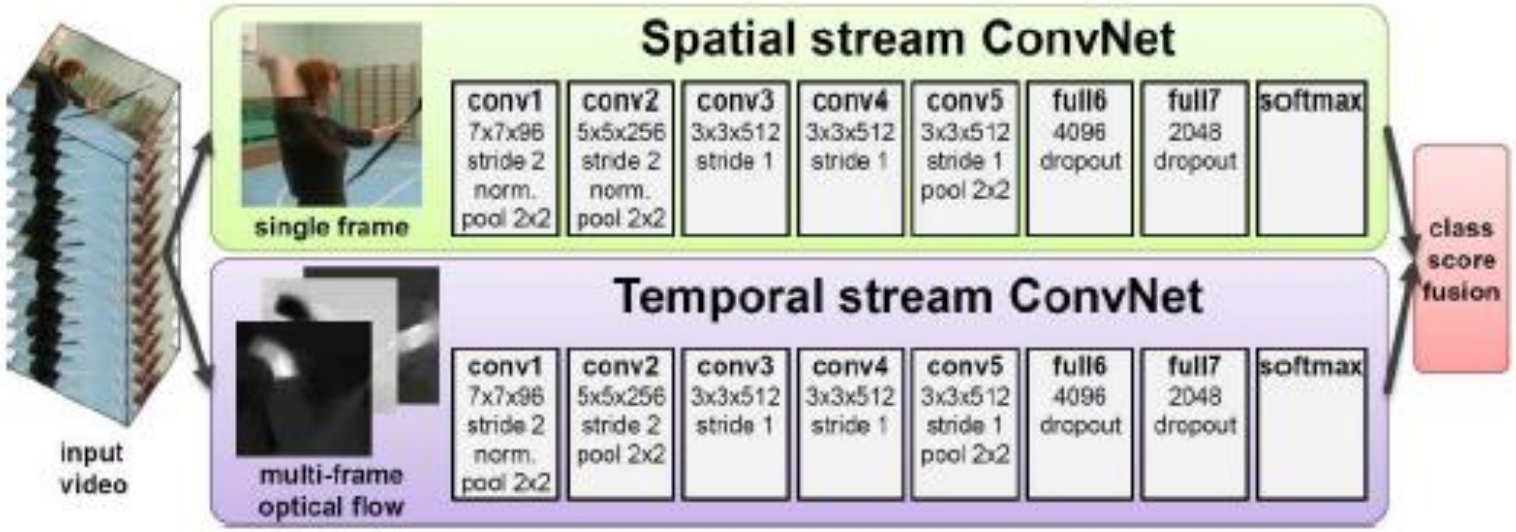


Figure 6.43 *Dense pose estimation aims at mapping all human pixels of an RGB image to the 3D surface of the human body (Güler, Neverova, and Kokkinos 2018) © 2018 IEEE. The paper describes DensePose-COCO, a large-scale ground-truth dataset containing manually annotated image-to-surface correspondences for 50K persons and a DensePose-RCNN trained to densely regress UV coordinates at multiple frames per second.*

2000s – Spatio-temporal Features

Optical flow patches (Efros et al. 2003)
Spatio-temporal feature detectors (Laptev et al. 2008)
Enhancements: image context, tracked trajectories



Dataset for Video Understanding and Action Recog.

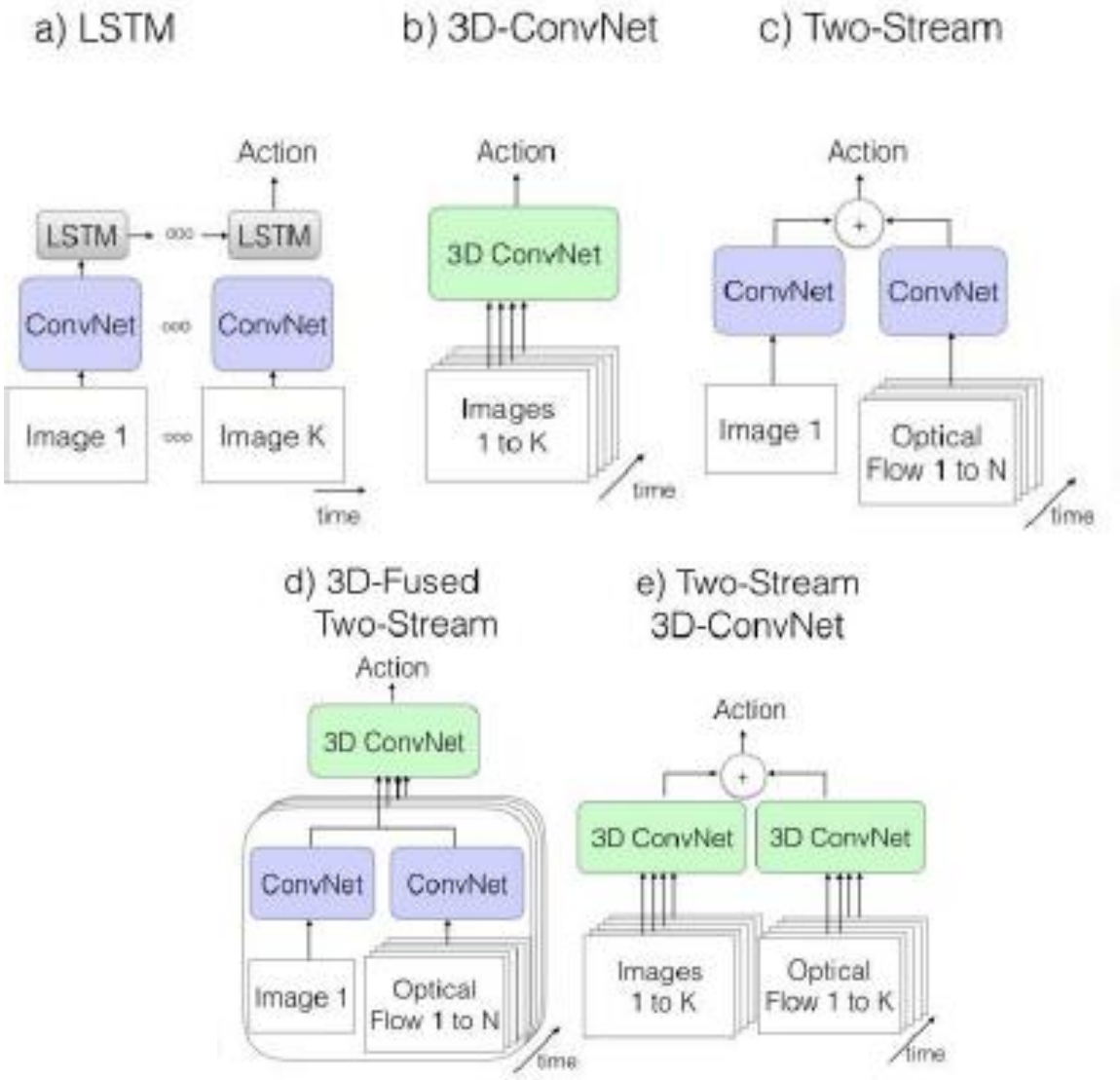
Name/URL	Metadata	Contents/Reference
Charades	Actions, objects, descriptions	9.8k videos
https://prior.allenai.org/projects/charades		Sigurdsson, Varol *et al.* (2016)
YouTube8M	Entities	4.8k visual entities, 8M videos
https://research.google.com/youtube8m		Abu-El-Haija, Kothari *et al.* (2016)
Kinetics	Action classes	700 action classes, 650k videos
https://deepmind.com/research/open-source/kinetics		Carreira and Zisserman (2017)
"Something-something"	Actions with objects	174 actions, 220k videos
https://20bn.com/datasets/something-something		Goyal, Kahou *et al.* (2017)
AVA	Actions	80 actions in 430 15-minute videos
https://research.google.com/ava		Gu, Sun *et al.* (2018)
EPIC-KITCHENS	Actions and objects	100 hours of egocentric videos
https://epic-kitchens.github.io		Damen, Doughty *et al.* (2018)

Table 6.3 *Datasets for video understanding and action recognition.*

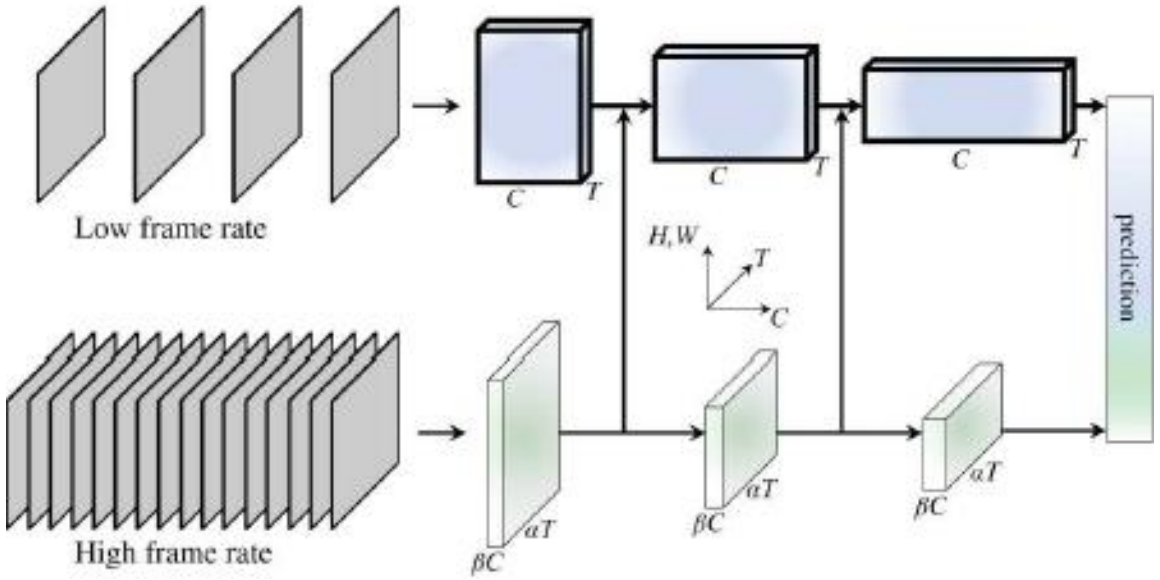
2010s – Deep Learning for Video

3D ConvNets (Ji et al. 2013, Karpathy et al. 2014, Tran et al. 2015)
Two-stream CNNs (Simonyan & Zisserman, 2014)
Hybrid CNN + LSTM models
SlowFast networks (Feichtenhofer et al. 2019)
Long-term feature bank (Wu et al. 2019)

2010s – Deep Learning for Video



2010s – Deep Learning for Video



a SlowFast network
with a low frame rate, low temporal resolution Slow pathway and a high frame rate, higher
temporal resolution Fast pathway (Feichtenhofer, Fan et al. 2019) © 2019 IEEE.

Recent Advances

Self-supervised learning for video
Multi-modal input: video + audio
Beyond actions: Dynamic scene recognition

6.6 Vision & Language - Goal

Toward Artificial General Intelligence (AGI)
Integration with speech, language, reasoning & knowledge

Early Work

Image captioning & labeling
BabyTalk system (Kulkarni et al. 2013): detect objects, infer labels, generate captions

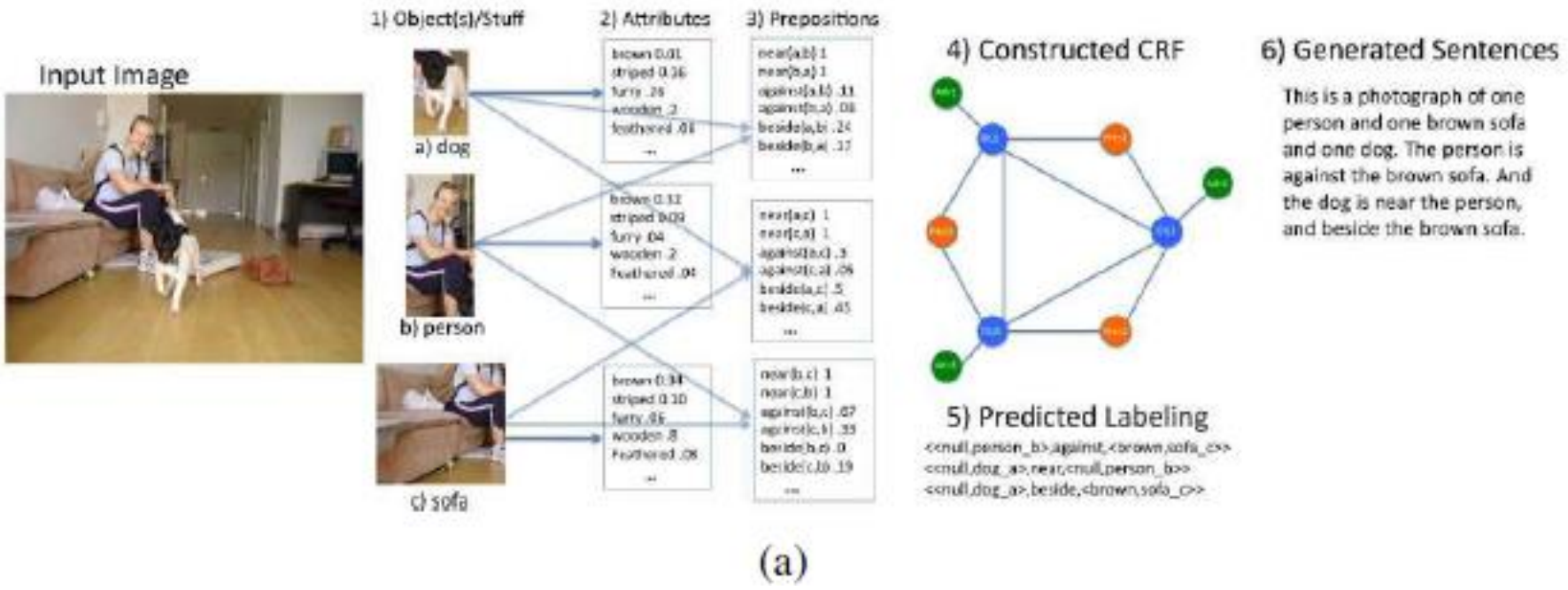


Image Captioning Systems

CNN + RNN/LSTM pipelines
Techniques: MIL, maximum entropy, attention
Nearest-neighbor captioning (Devlin et al. 2015)
Transformers for captioning

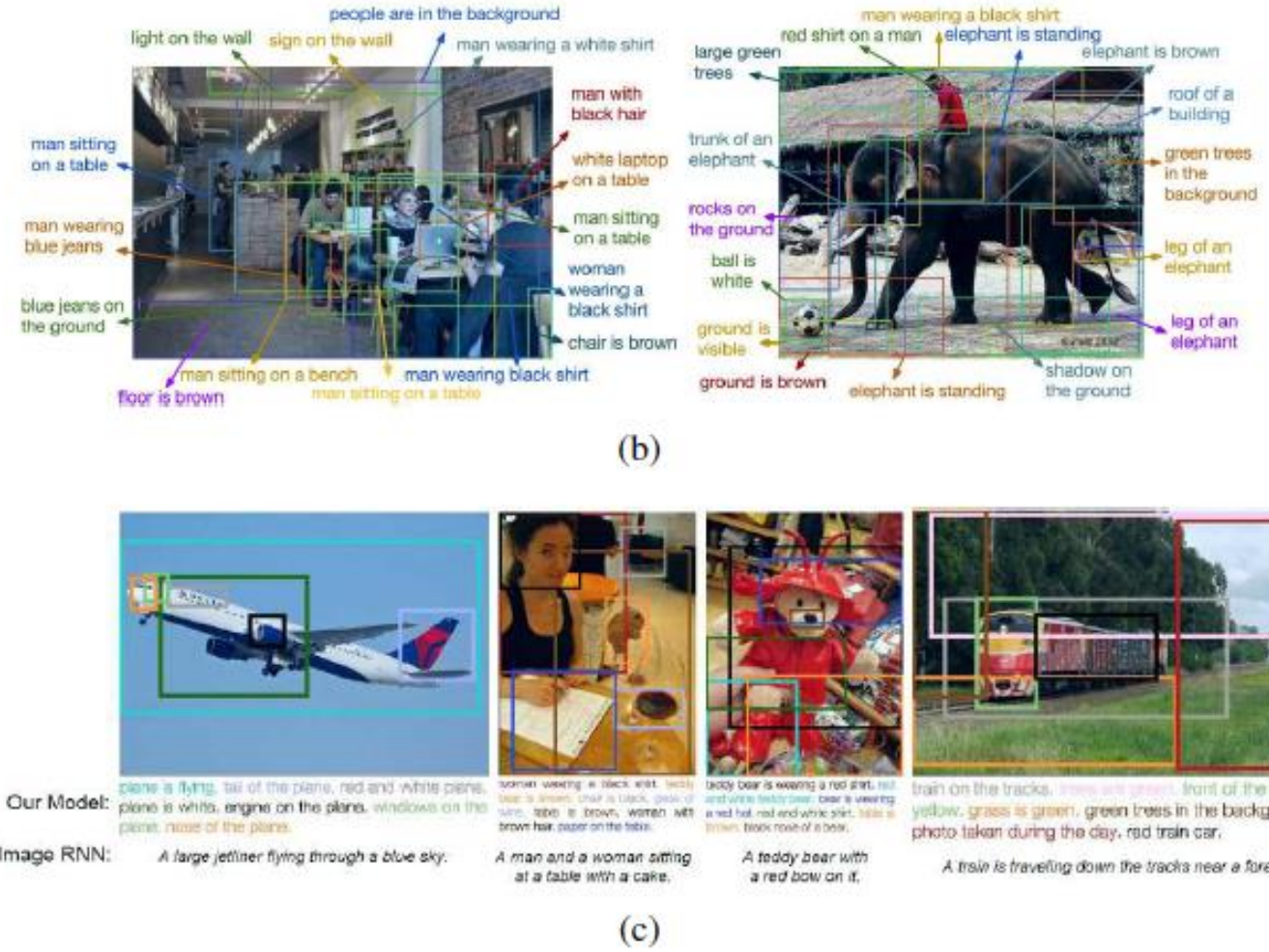


Figure 6.45 *Image captioning systems: (a) BabyTalk detects objects, attributes, and positional relationships and composes these into image captions (Kulkarni, Premraj* et al. *2013) © 2013 IEEE; (b–c) DenseCap associates word phrases with regions and then uses an RNN to construct plausible sentences (Johnson, Karpathy, and Fei-Fei 2016) © 2016 IEEE.*

Image Captioning Systems

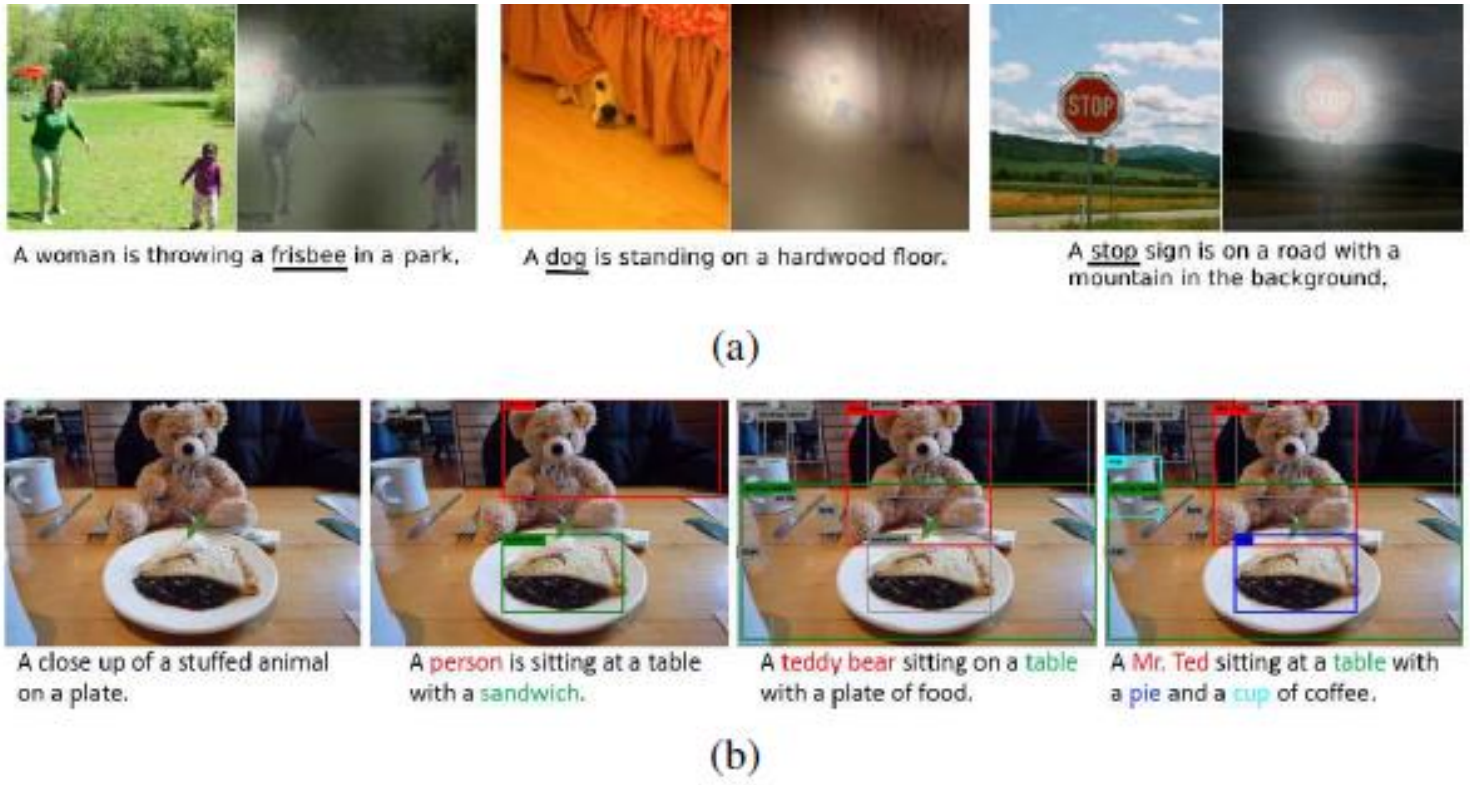


Figure 6.46 *Image captioning with attention: (a) The "Show, Attend, and Tell" system, which uses hard attention to align generated words with image regions © Xu, Ba* et al. *(2015); (b) Neural Baby Talk captions generated using different detectors, showing the association between words and grounding regions (Lu, Yang* et al. *2018) © 2018 IEEE.*

Zero-shot Models

CLIP (Radford et al. 2021)
Trained on 400M text–image pairs
Contrastive learning for matching
Enables zero-shot classification
Robust but vulnerable to adversarial attacks

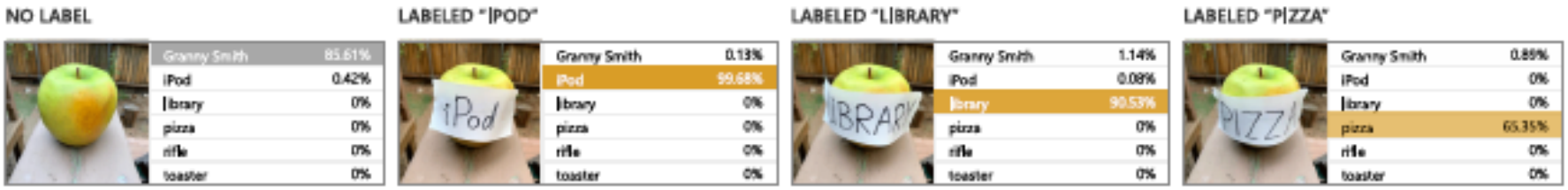


Figure 6.47 *An adversarial typographic attack used against CLIP (Radford, Kim* et al. *2021) discovered by ©Goh, Cammarata* et al. *(2021). Instead of predicting the object that exists in the scene, CLIP predicts the output based on the adversarial handwritten label.*

Datasets for Vision + Language

Name/URL	Metadata	Contents/Reference
Flickr30k (Entities)	Image captions (grounded)	30k images (+ bounding boxes)
https://shannon.cs.illinois.edu/DenotationGraph		Young, Lai *et al.* (2014)
http://bryanplummer.com/Flickr30kEntities		Plummer, Wang *et al.* (2017)
COCO Captions	Whole image captions	1.5M captions, 330k images
https://cocodataset.org/#captions-2015		Chen, Fang *et al.* (2015)
Conceptual Captions	Whole image captions	3.3M image caption pairs
https://ai.google.com/research/ConceptualCaptions		Sharma, Ding *et al.* (2018)
YFCC100M	Flickr metadata	100M images with metadata
http://projects.dfki.uni-kl.de/yfcc100m		Thomee, Shamma *et al.* (2016)
Visual Genome	Dense annotations	108k images with region graphs
https://visualgenome.org		Krishna, Zhu *et al.* (2017)
VQA v2.0	Question/answer pairs	265k images
https://visualqa.org		Goyal, Khot *et al.* (2017)
VCR	Multiple choice questions	110k movie clips, 290k QAs
https://visualcommonsense.com		Zellers, Bisk *et al.* (2019)
GQA	Compositional QA	22M questions on Visual Genome
https://visualreasoning.net		Hudson and Manning (2019)
VisDial	Dialogs for chatbot	120k COCO images + dialogs
https://visualdialog.org		Das, Kottur *et al.* (2017)

Evaluation Metrics

BLEU, ROUGE, METEOR
CIDEr (consensus-based), SPICE (semantic-based)

Text-to-Image Generation

Early: RNNs (Mansimov 2016), GANs (Reed 2016)

Recent: DALL·E (Ramesh 2021)

VQ-VAE-2 + Transformer decoder

High-resolution creative images & image completion

Visual Question Answering (VQA)

Beyond captioning: reasoning-based QA
Datasets: VQA, VCR, GQA
Methods: attention, fusion models, transformers

Visual Dialog

Conversational Q&A about images
Dataset: VisDial (Das et al. 2017)
Applications: Image-grounded chatbots

Vision-Language Pre-training

Large-scale multimodal pre-training
Examples: ViLBERT, Oscar, other transformer-based models