

# Computer Vision - Lecture 22

## Repetition

06.02.2017

Bastian Leibe

RWTH Aachen

<http://www.vision.rwth-aachen.de>

[leibe@vision.rwth-aachen.de](mailto:leibe@vision.rwth-aachen.de)

# Announcements

- **Exam**

- **1<sup>st</sup> Date:** Monday, 29.02., 13:30 - 17:30h
- **2<sup>nd</sup> Date:** Thursday, 31.03., 09:30 - 12:30h
- **Closed-book exam, the core exam time will be 2h.**
- We will send around an announcement with the exact starting times and places by email.

- **Test exam**

- **Date:** Thursday, 11.02., 14:15 - 15:45h, room UMIC 025
- **Core exam time will be 1h**
- **Purpose:** Prepare you for the questions you can expect.
- *Possibility to collect bonus exercise points!*

# Announcements (2)

- Feedback to the lecture evaluation

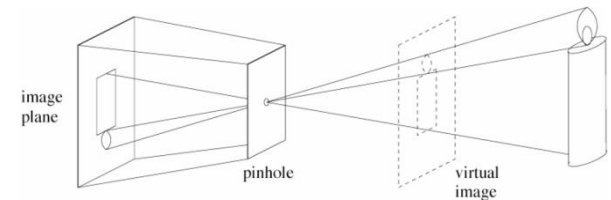
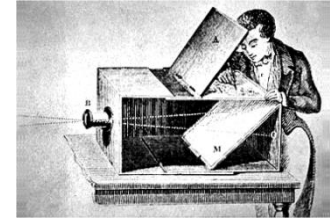
# Announcements

- Today, I'll summarize the most important points from the lecture.
  - It is an opportunity for you to ask questions...
  - ...or get additional explanations about certain topics.
  - *So, please do ask.*
- Today's slides are intended as an index for the lecture.
  - But they are not complete, won't be sufficient as only tool.
  - Also look at the exercises - they often explain algorithms in detail.

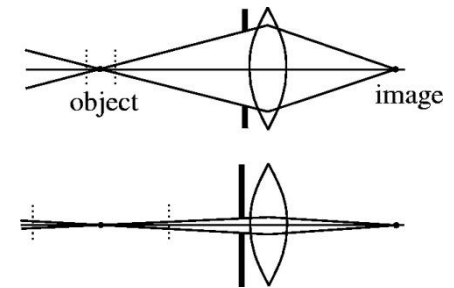


# Repetition

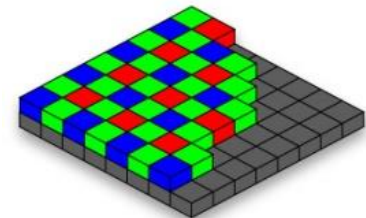
- Image Processing Basics
  - Image Formation
  - Binary Image Processing
  - Linear Filters
  - Edge & Structure Extraction
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
- Motion and Tracking



*Pinhole camera model*



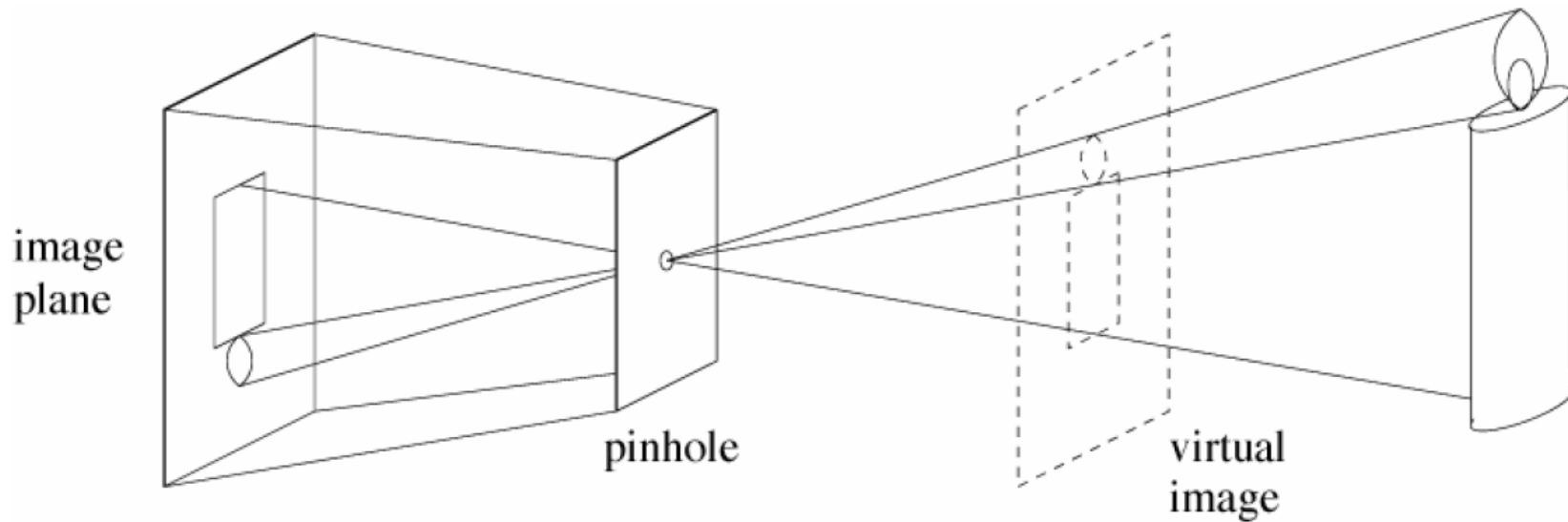
*Lenses, focal length, aperture*



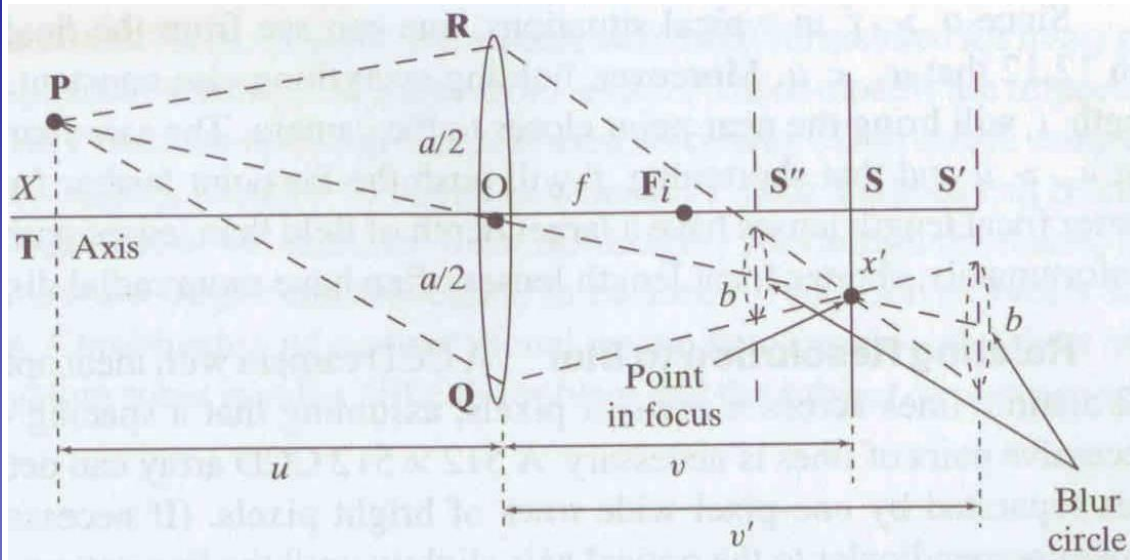
*Color sensors*

# Recap: Pinhole Camera

- (Simple) standard and abstract model today
  - Box with a small hole in it
  - Works in practice



# Recap: Focus and Depth of Field



“circles of confusion”

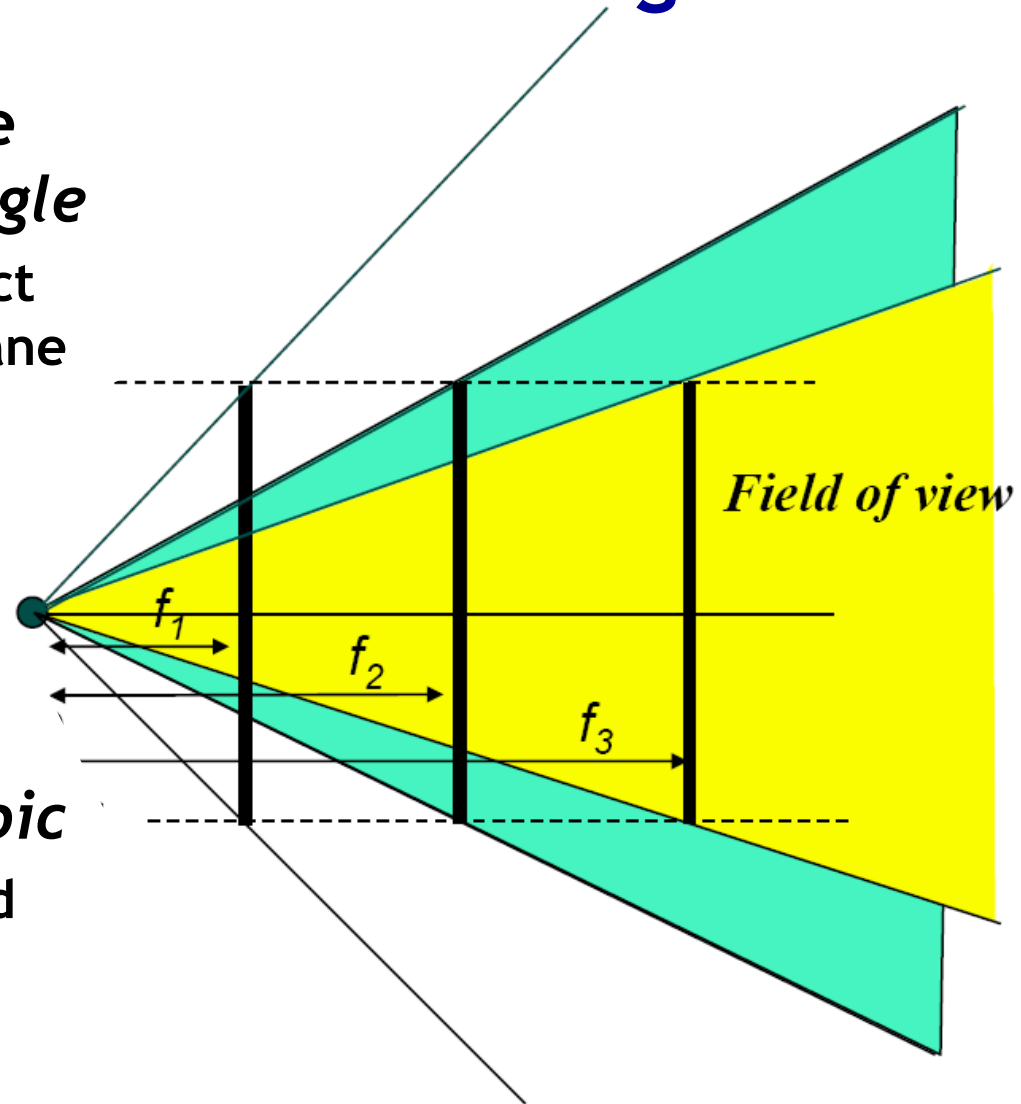
Thin lens: scene points at distinct depths come in focus at different image planes.

(Real camera lens systems have greater depth of field.)

- Depth of field: distance between image planes where blur is tolerable

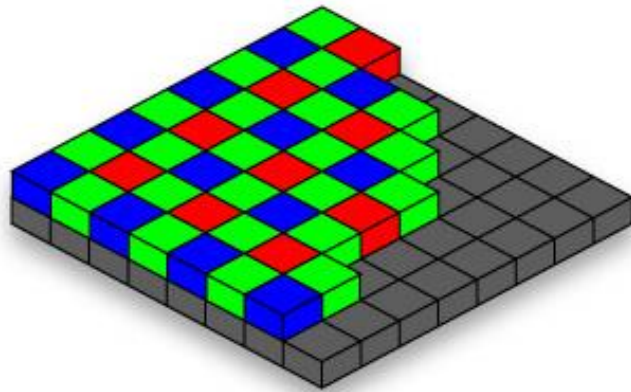
# Recap: Field of View and Focal Length

- As  $f$  gets smaller, image becomes more *wide angle*
  - More world points project onto the finite image plane
- As  $f$  gets larger, image becomes more *telescopic*
  - Smaller part of the world projects onto the finite image plane

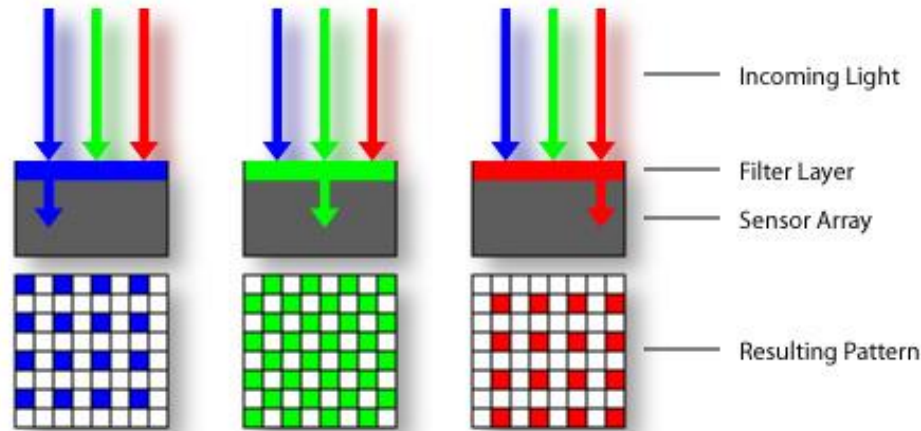


# Recap: Color Sensing in Digital Cameras

Bayer grid

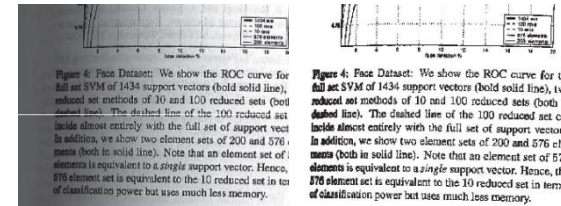
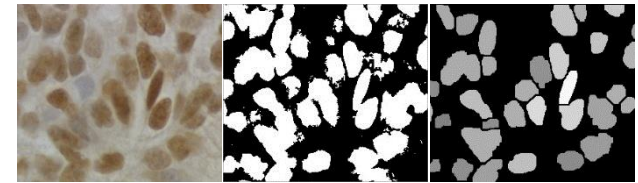


Estimate missing components from neighboring values (demaosaicing)



# Repetition

- Image Processing Basics
  - Image Formation
  - Binary Image Processing
  - Linear Filters
  - Edge & Structure Extraction
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
- Motion and Tracking

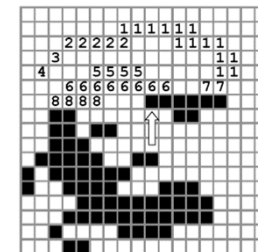


## Thresholding



$$A \circ B = (A \ominus B) \oplus B$$

## Morphological Operators

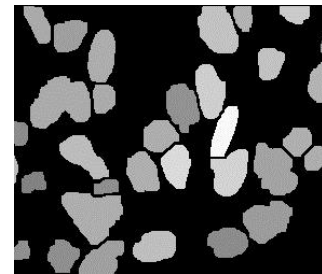
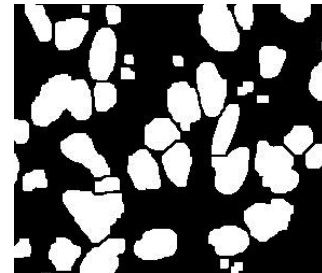
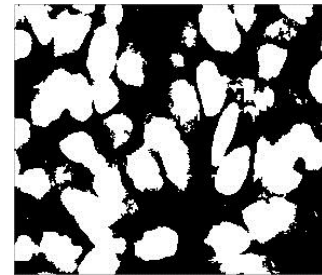
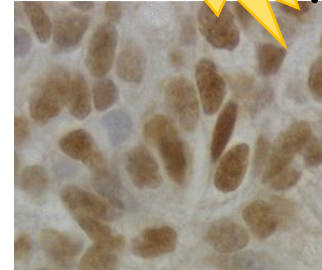


## Connected Components

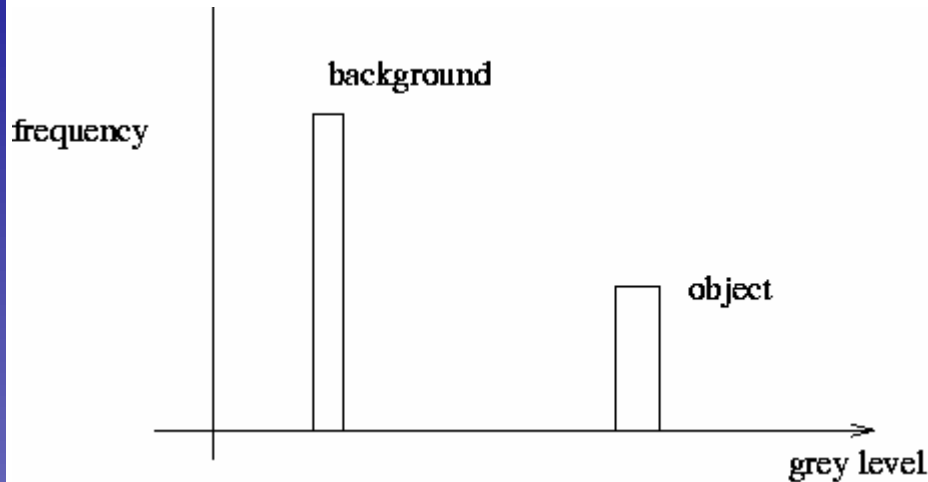
see  
"Haribo" Demo!

# Recap: Binary Processing Pipeline

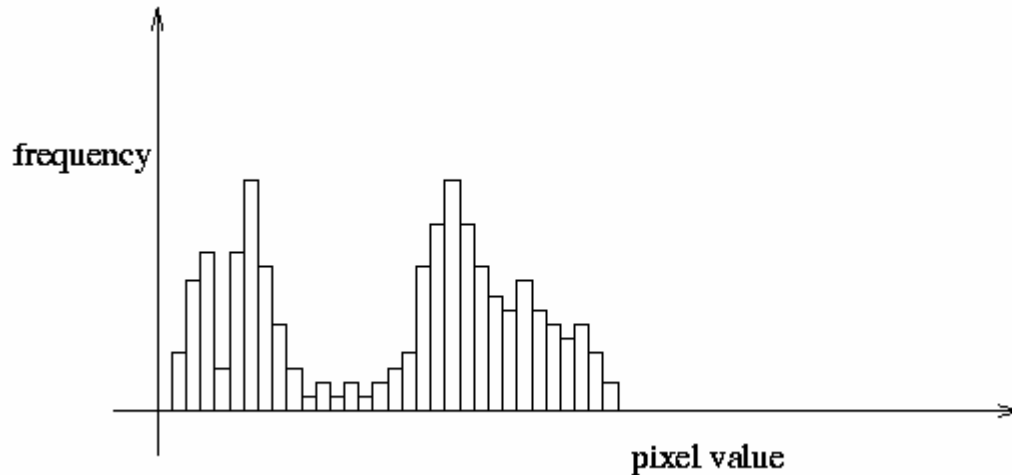
- Convert the image into binary form
  - Thresholding
- Clean up the thresholded image
  - Morphological operators
- Extract individual objects
  - Connected Components Labeling
- Describe the objects
  - Region properties



# Recap: Robust Thresholding



Ideal histogram,  
light object on  
dark background



Actual observed  
histogram with  
noise

Assumption here:  
only two modes



# Recap: Global Binarization [Otsu'79]

- Precompute a cumulative grayvalue histogram  $h$ .
- For each potential threshold  $T$

1.) Separate the pixels into two clusters according to  $T$ .

2.) Compute both cluster means  $\mu_1(T)$  and  $\mu_2(T)$ .

Look up  $n_1, n_2$  in  $h$

$$n_1(T) = |\{I_{(x,y)} < T\}|, \quad n_2(T) = |\{I_{(x,y)} \geq T\}|$$

3.) Compute the between-class variance  $\sigma_{between}^2(T)$

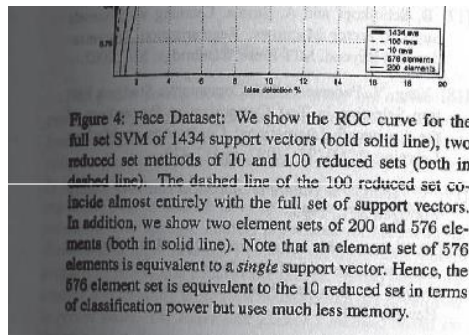
$$\sigma_{between}^2(T) = n_1(T)n_2(T) [\mu_1(T) - \mu_2(T)]^2$$

- Choose the threshold that maximizes

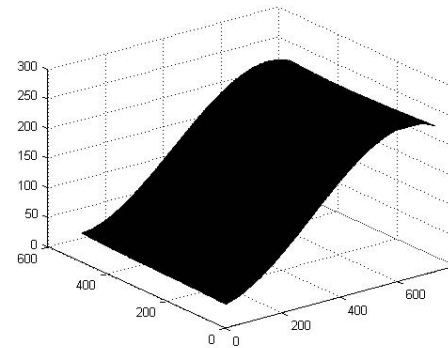
$$T^* = \arg \max_T [\sigma_{between}^2(T)]$$

# Recap: Background Surface Fitting

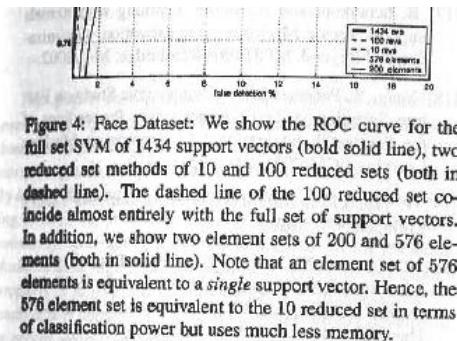
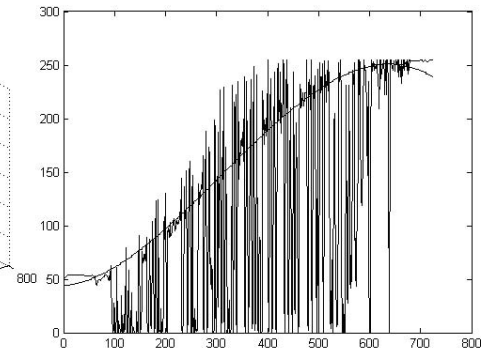
- Document images often contain a smooth gradient
- ⇒ *Try to fit that gradient with a polynomial function*



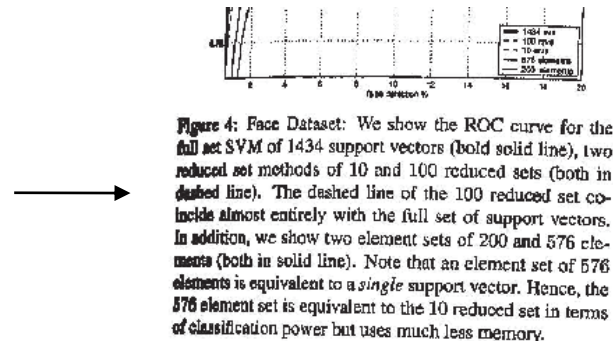
Original image



Fitted surface



Shading compensation

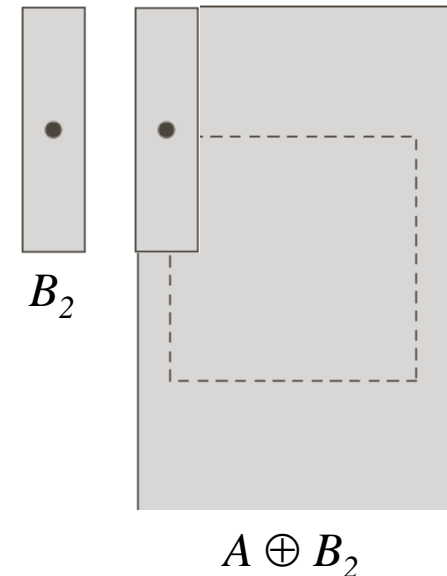
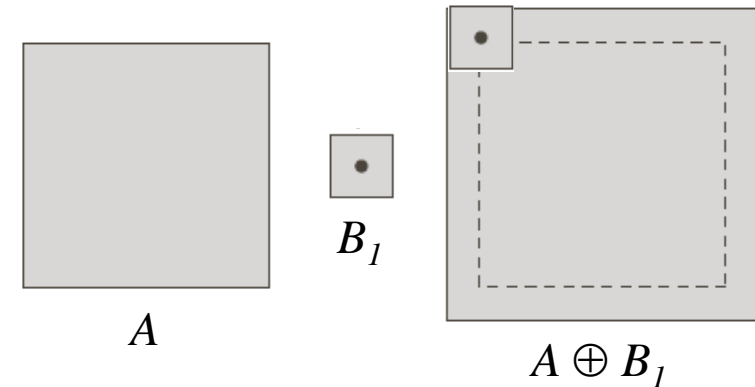


Binarized result

# Recap: Dilation

- **Definition**

- “The dilation of  $A$  by  $B$  is the set of all displacements  $z$ , such that  $(\hat{B})_z$  and  $A$  overlap by at least one element”.
  - $(\hat{B})_z$  is the mirrored version of  $B$ , shifted by  $z$



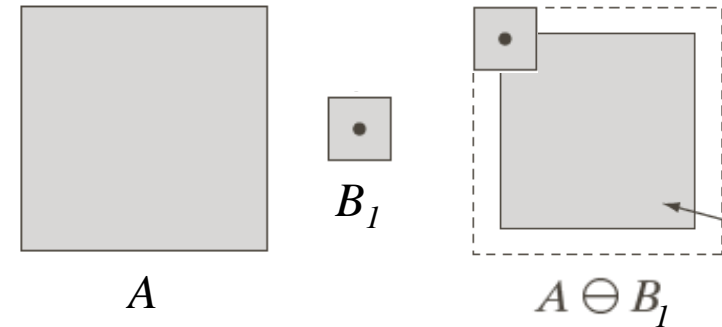
- **Effects**

- If current pixel  $z$  is foreground, set all pixels under  $(B)_z$  to foreground.
  - ⇒ Expand connected components
  - ⇒ Grow features
  - ⇒ Fill holes

# Recap: Erosion

## • Definition

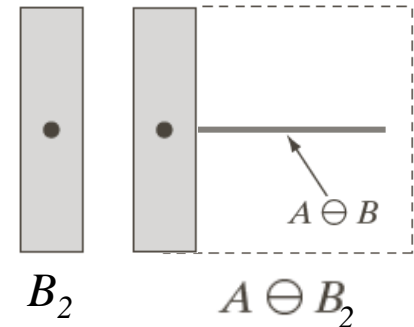
- “The erosion of  $A$  by  $B$  is the set of all displacements  $z$ , such that  $(B)_z$  is entirely contained in  $A$ ”.



## • Effects

- If not every pixel under  $(B)_z$  is foreground, set the current pixel  $z$  to background.

- ⇒ Erode connected components
- ⇒ Shrink features
- ⇒ Remove bridges, branches, noise



# Recap: Opening

- Definition

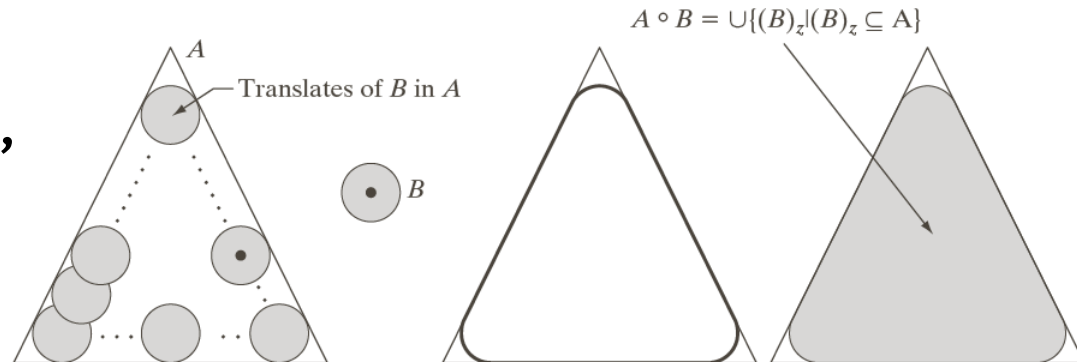
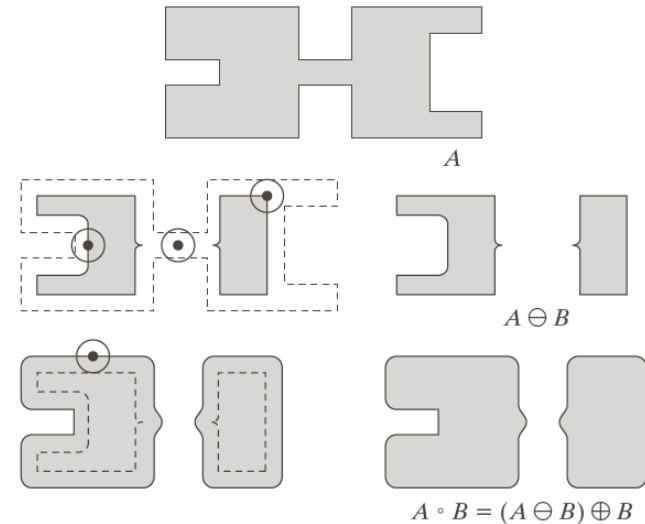
- Sequence of Erosion and Dilation

$$A \circ B = (A \ominus B) \oplus B$$

- Effect

- $A \circ B$  is defined by the points that are reached if  $B$  is rolled around inside  $A$ .

⇒ Remove small objects, keep original shape.

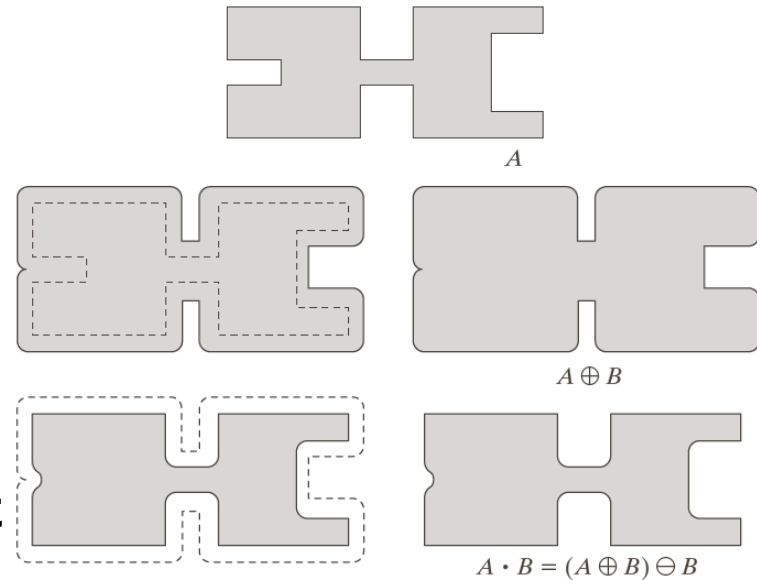


# Recap: Closing

- **Definition**

- Sequence of **Dilation and Erosion**

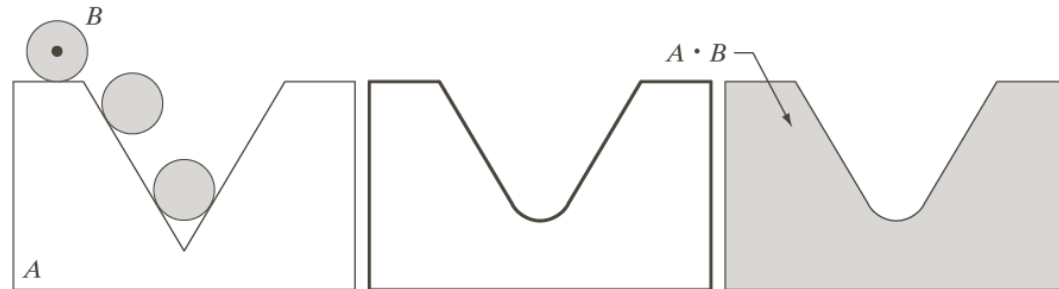
$$A \cdot B = (A \oplus B) \ominus B$$



- **Effect**

- $A \cdot B$  is defined by the points that are reached if  $B$  is *rolled around on the outside* of  $A$ .

⇒ Fill holes,  
keep original shape.



# Recap: Connected Components Labeling

- Process the image from left to right, top to bottom:

1.) If the next pixel to process is 1



i.) If only one of its neighbors (top or left) is 1, copy its label.



ii.) If both are 1 and have the same label, copy it.

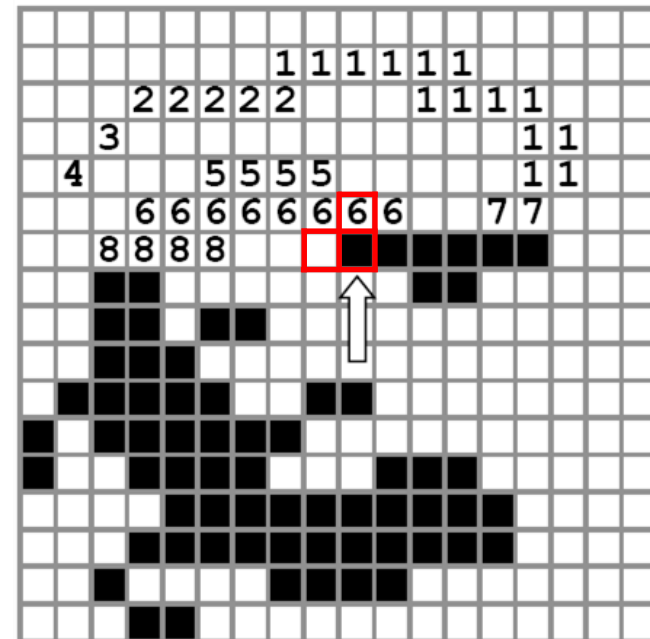


iii.) If they have different labels  
 – Copy the label from the left.  
 – Update the equivalence table.



iv.) Otherwise, assign a new label.

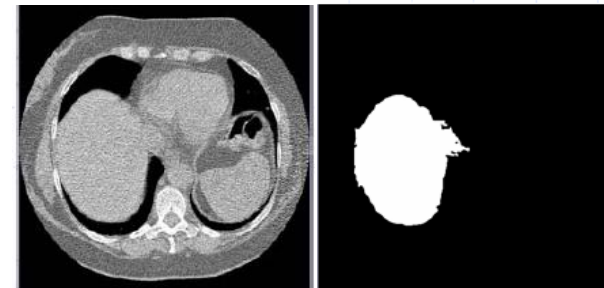
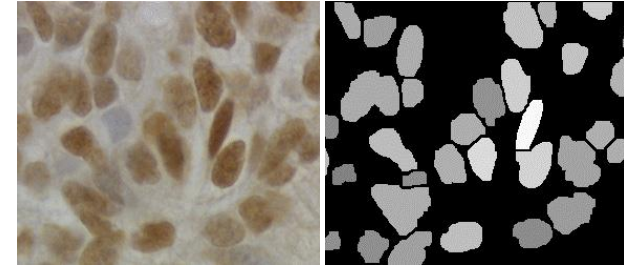
- Re-label with the smallest of equivalent labels



1	2, 7}
3	
4	
5	6, 8}

# Recap: Region Properties

- From the previous steps, we can obtain separated objects.
- Some useful features can be extracted once we have connected components, including
  - Area
  - Centroid
  - Extremal points, bounding box
  - Circularity
  - Spatial moments





# Recap: Moment Invariants

- Normalized central moments

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma}, \quad \gamma = \frac{p+q}{2} + 1$$

- From those, a set of *invariant moments* can be defined for object description.

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

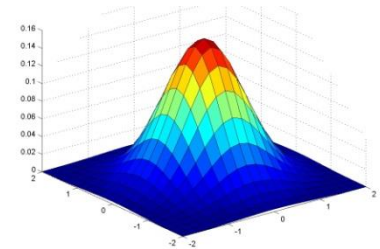
$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

(Additional invariant moments  $\phi_5, \phi_6, \phi_7$  can be found in the literature).

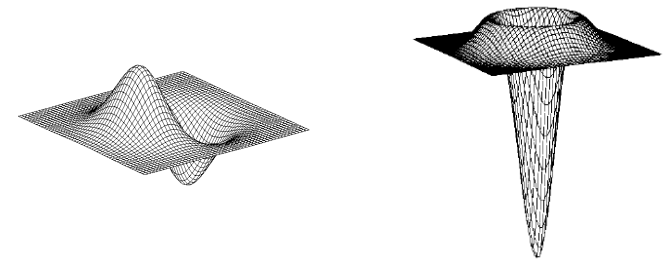
- Robust to translation, rotation & scaling, but don't expect wonders (still summary statistics).

# Repetition

- Image Processing Basics
  - Image Formation
  - Binary Image Processing
  - Linear Filters
  - Edge & Structure Extraction
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
- Motion and Tracking



*Gaussian Smoothing*



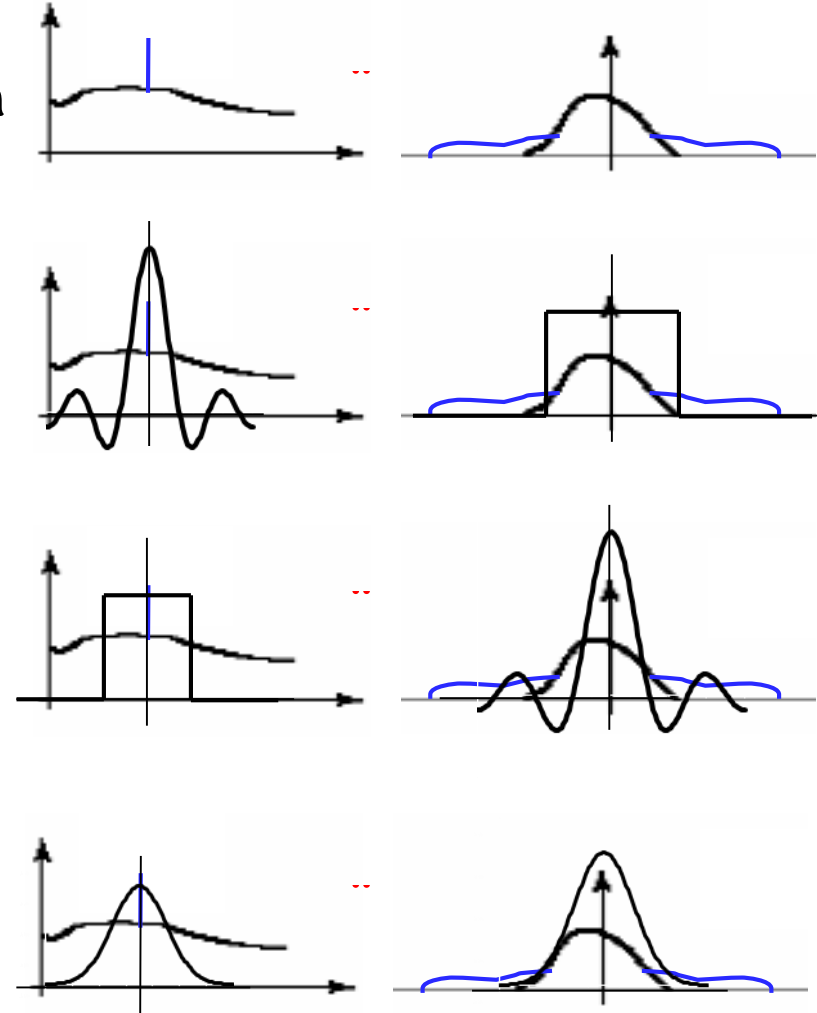
*Derivative operators*



*Gaussian/Laplacian pyramid*

# Recap: Effect of Filtering

- Noise introduces high frequencies. To remove them, we want to apply a “low-pass” filter.
- The ideal filter shape in the frequency domain would be a box. But this transfers to a spatial sinc, which has infinite spatial support.
- A compact spatial box filter transfers to a frequency sinc, which creates artifacts.
- A Gaussian has compact support in both domains. This makes it a convenient choice for a low-pass filter.

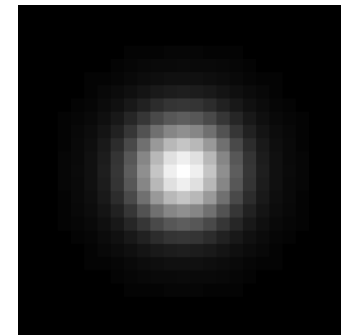
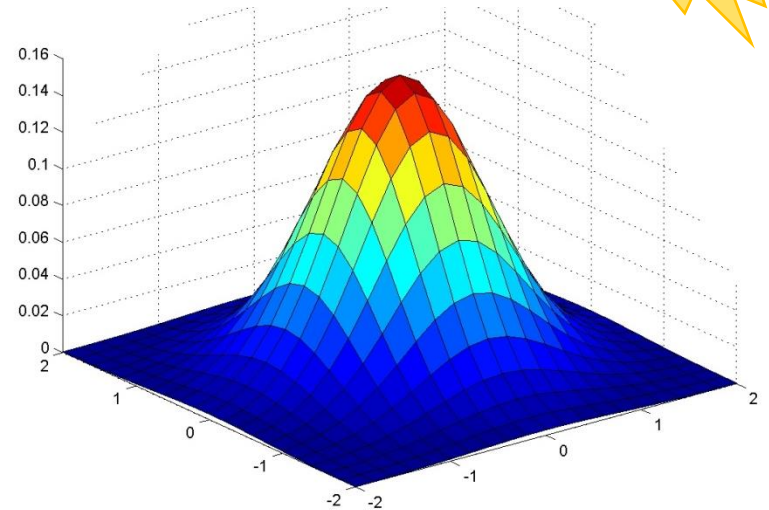


# Recap: Gaussian Smoothing

- Gaussian kernel

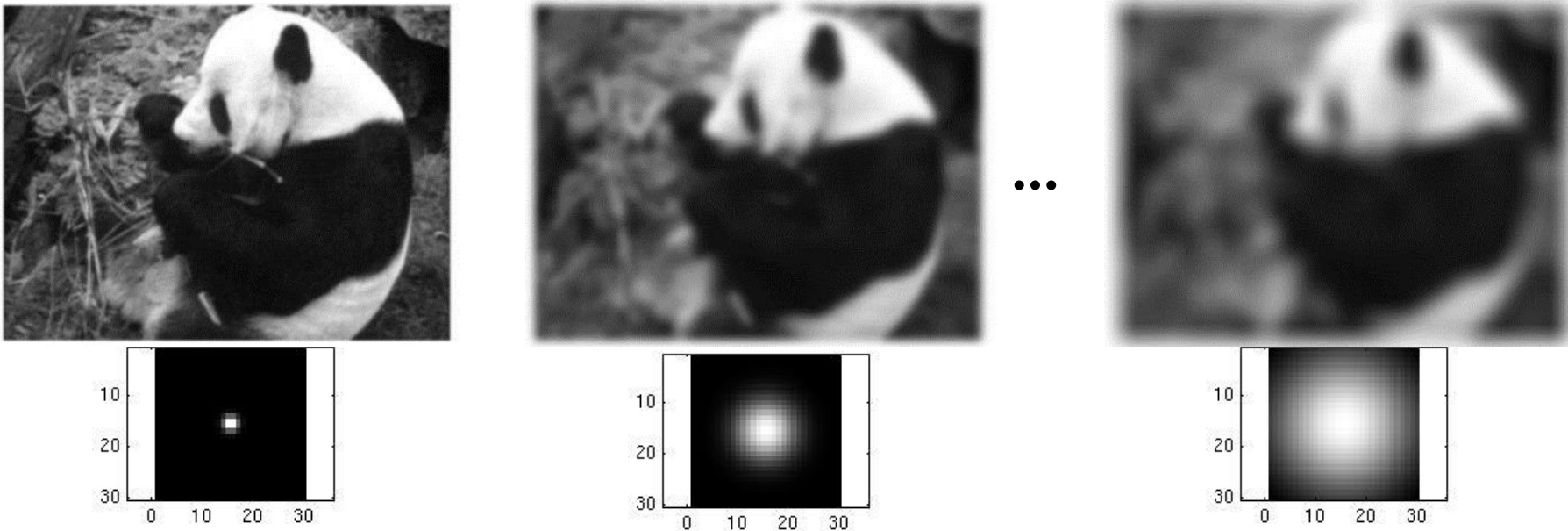
$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- Rotationally symmetric
- Weights nearby pixels more than distant ones
  - This makes sense as ‘probabilistic’ inference about the signal
- A Gaussian gives a good model of a fuzzy blob



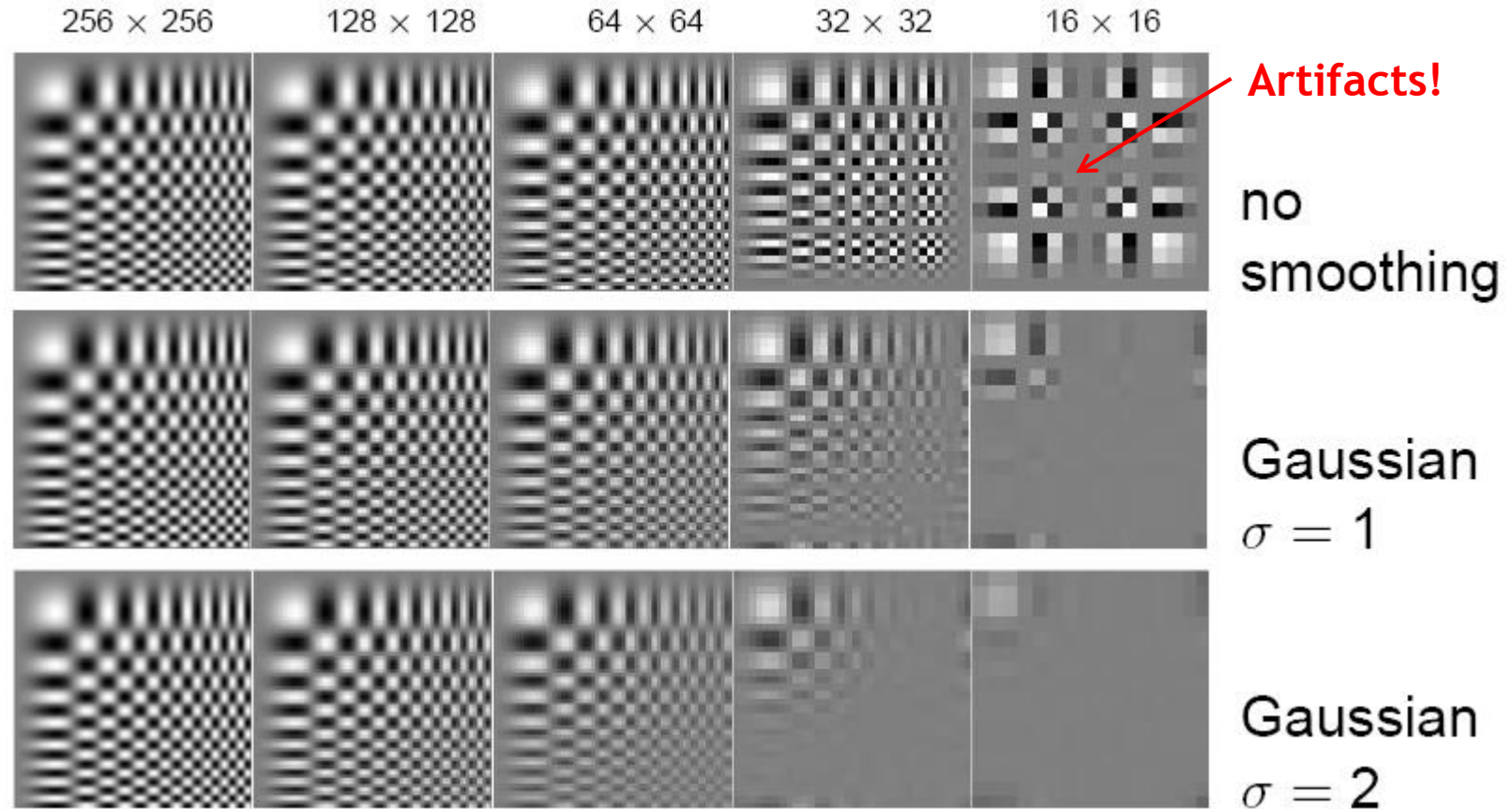
# Recap: Smoothing with a Gaussian

- Parameter  $\sigma$  is the “scale” / “width” / “spread” of the Gaussian kernel and controls the amount of smoothing.



```
for sigma=1:3:10
    h = fspecial('gaussian', fsize, sigma);
    out = imfilter(im, h);
    imshow(out);
    pause;
end
```

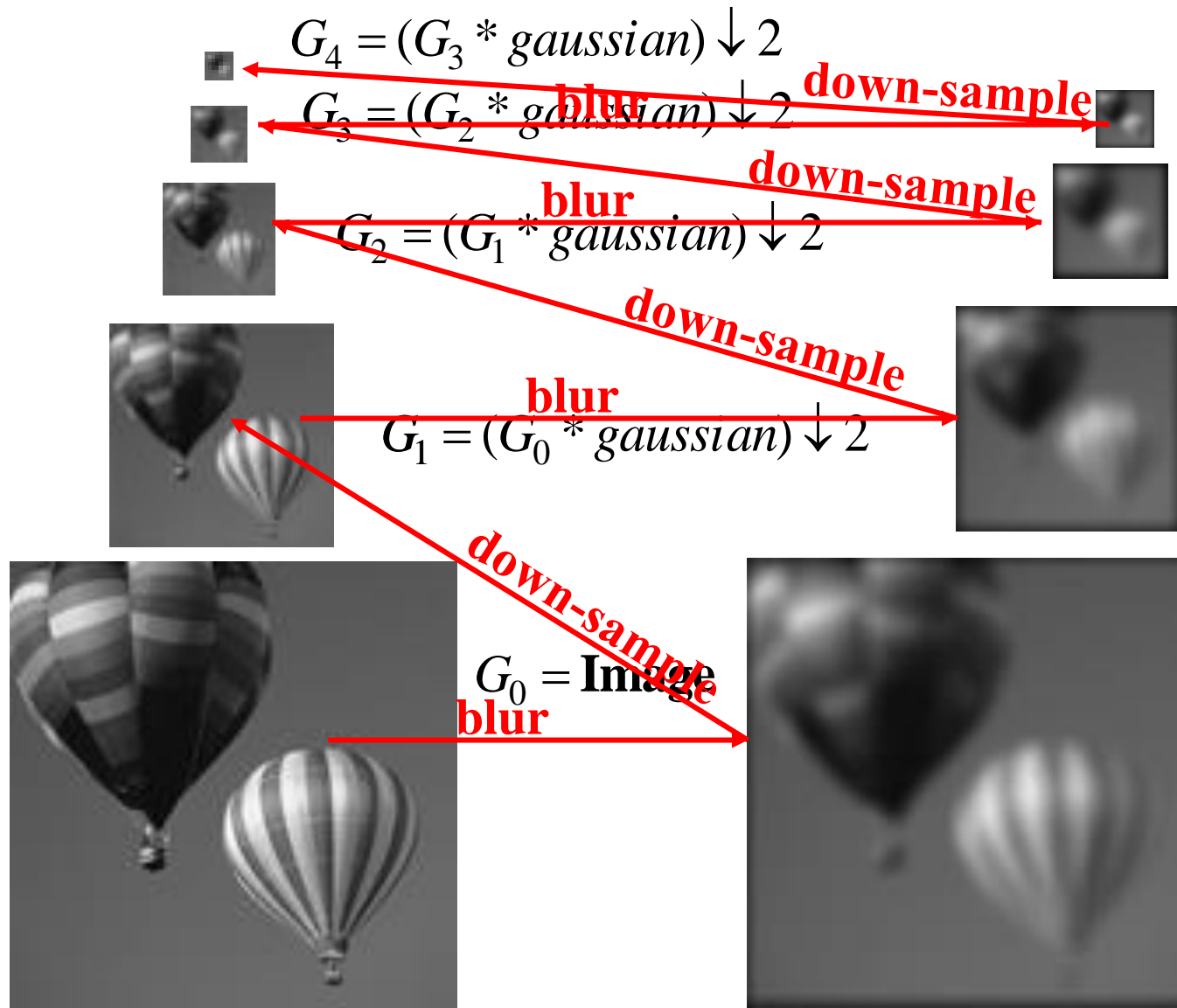
# Recap: Resampling with Prior Smoothing



- Note: We cannot recover the high frequencies, but we can avoid artifacts by smoothing before resampling.

# Recap: The Gaussian Pyramid

Low resolution



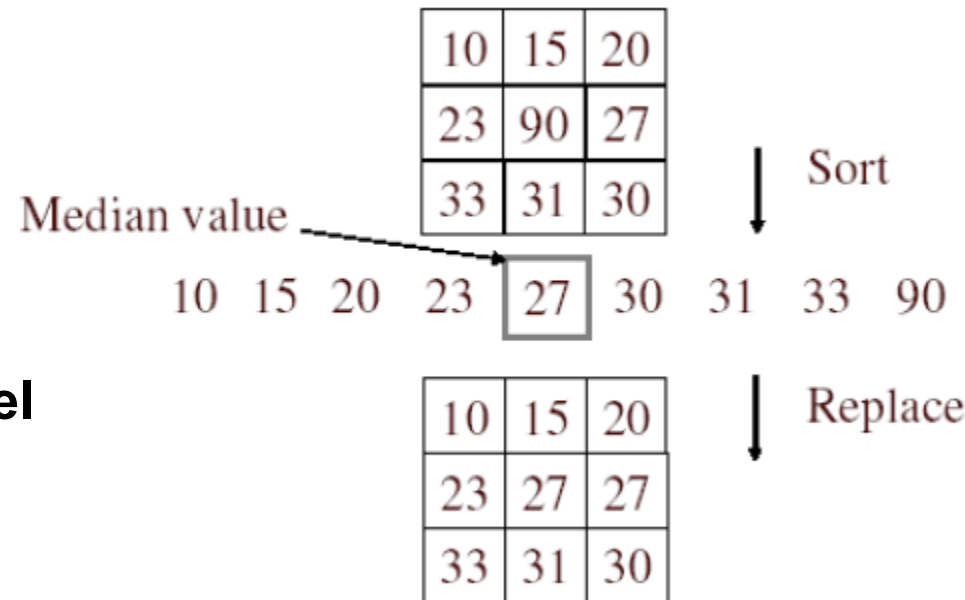
High resolution



# Recap: Median Filter

- **Basic idea**

- Replace each pixel by the median of its neighbors.

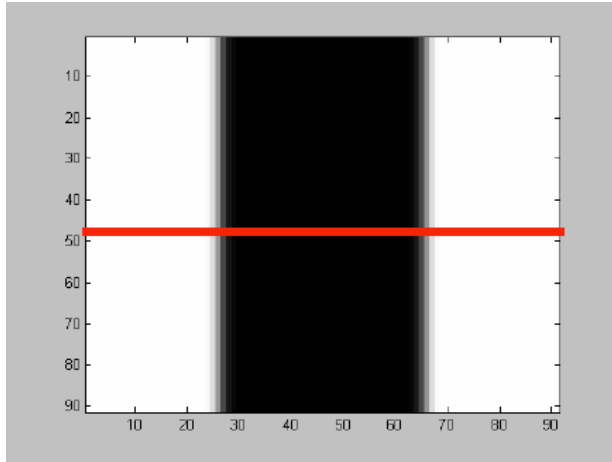


- **Properties**

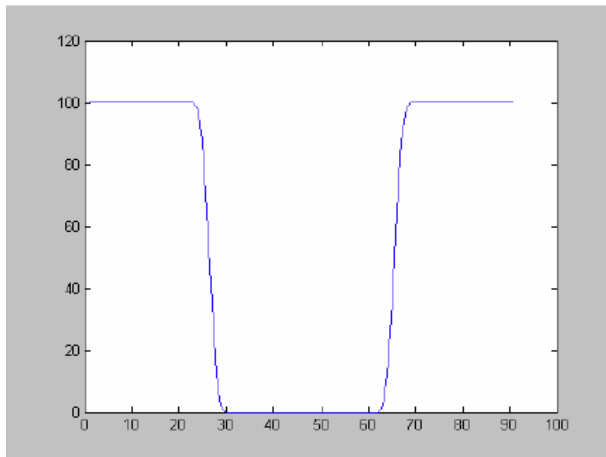
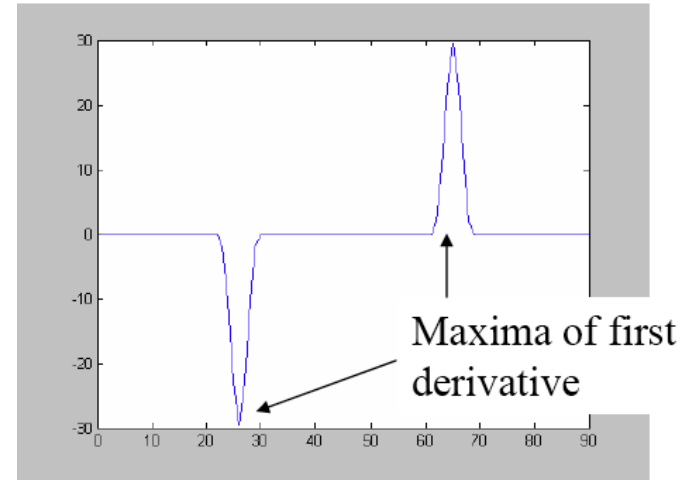
- Doesn't introduce new pixel values
- Removes spikes: good for impulse, salt & pepper noise
- Nonlinear
- Edge preserving



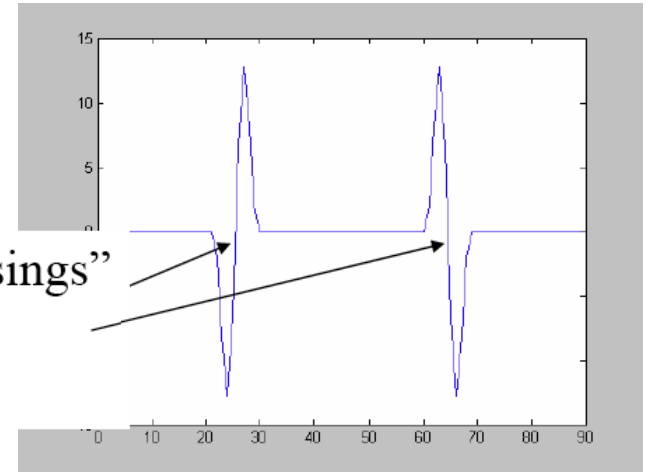
# Recap: Derivatives and Edges...



1st derivative

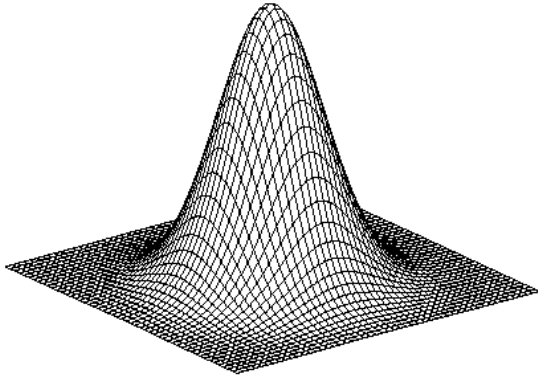


2nd derivative



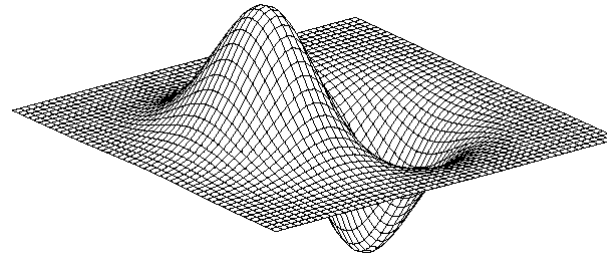
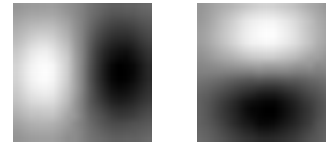
“zero crossings”  
of second  
derivative

# Recap: 2D Edge Detection Filters



Gaussian

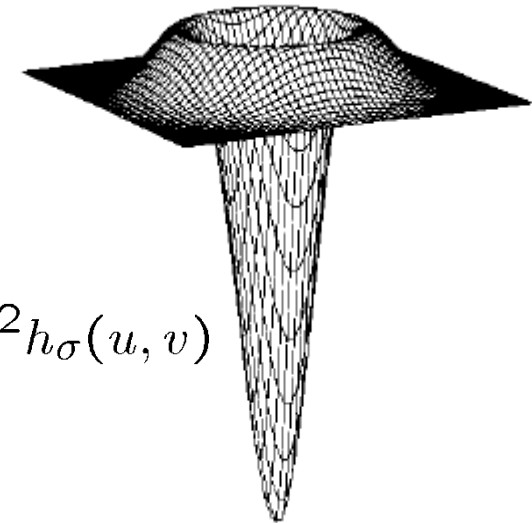
$$h_{\sigma}(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian

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

Laplacian of Gaussian



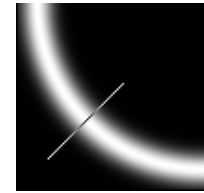
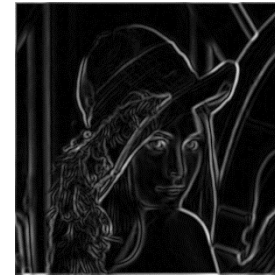
$$\nabla^2 h_{\sigma}(u, v)$$

- $\nabla^2$  is the Laplacian operator:

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

# Repetition

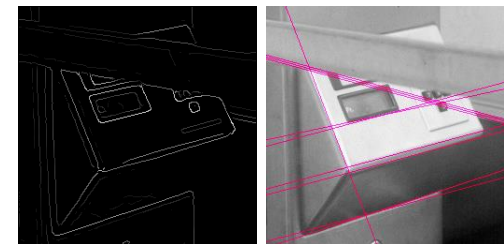
- Image Processing Basics
  - Image Formation
  - Binary Image Processing
  - Linear Filters
  - Edge & Structure Extraction
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
- Motion and Tracking



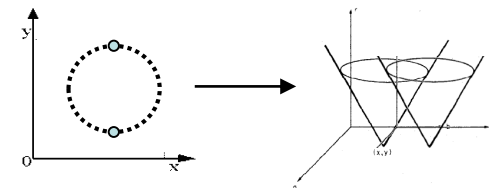
*Canny edge detector*



*Chamfer matching*



*Hough transform for lines*



*Hough transform for circles*

# Recap: Canny Edge Detector

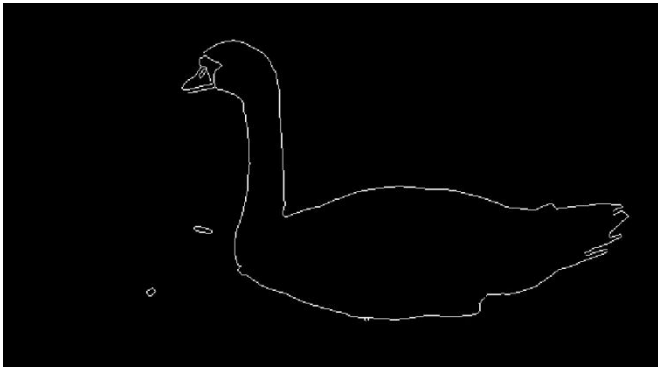
1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
  - Thin multi-pixel wide “ridges” down to single pixel width
4. Linking and thresholding (hysteresis):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

- **MATLAB:**

```
>> edge (image, 'canny' ) ;  
>> help edge
```



# Recap: Edges vs. Boundaries



Edges useful signal to indicate occluding boundaries, shape.

Here the raw edge output is not so bad...



...but quite often boundaries of interest are fragmented, and we have extra “clutter” edge points.

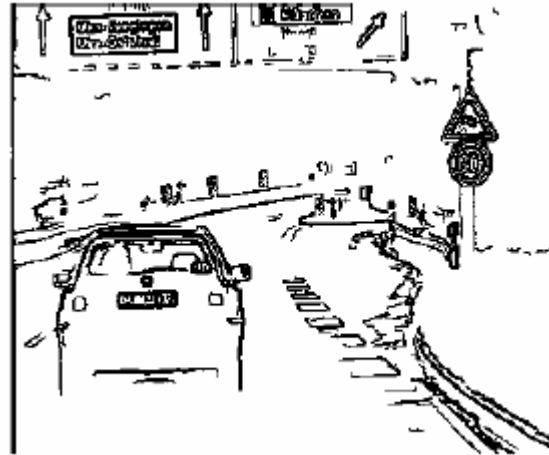
# Recap: Chamfer Matching

- Chamfer Distance

- Average distance to nearest feature

$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

- This can be computed efficiently by correlating the edge template with the distance-transformed image



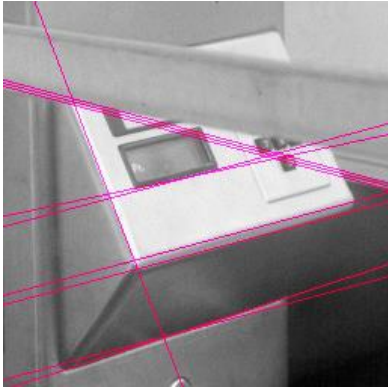
Edge image



Distance transform image



# Recap: Fitting and Hough Transform



Given a model of interest, we can overcome some of the missing and noisy edges using fitting techniques.



With voting methods like the Hough transform, detected points vote on possible model parameters.

# Recap: Hough Transform

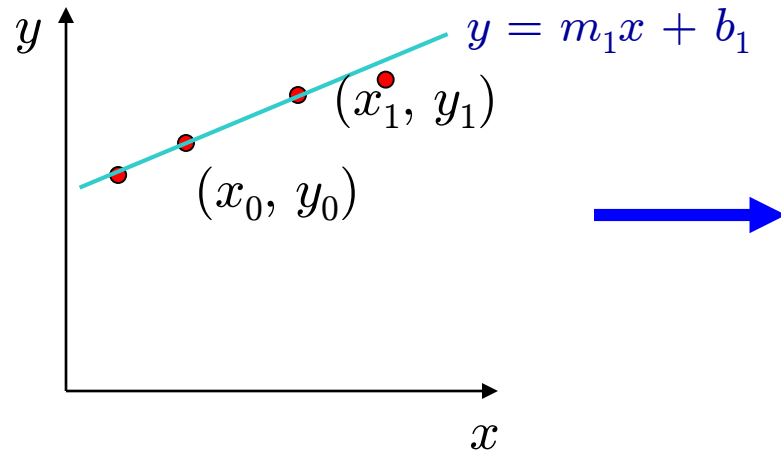
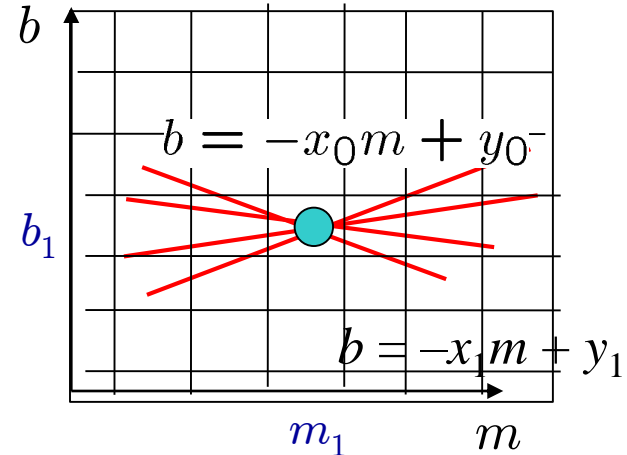


Image space



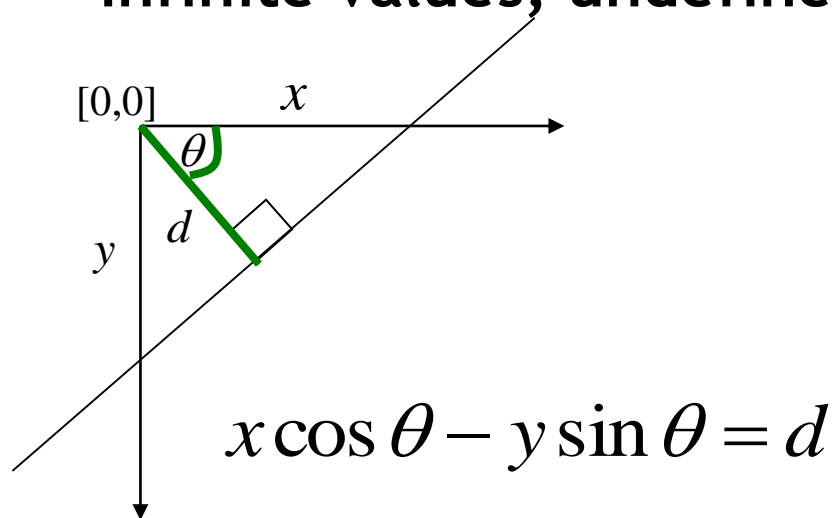
Hough (parameter) space

- How can we use this to find the most likely parameters  $(m, b)$  for the most prominent line in the image space?
  - Let each edge point in image space *vote* for a set of possible parameters in Hough space
  - Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.



# Recap: Hough Transf. Polar Parametrization

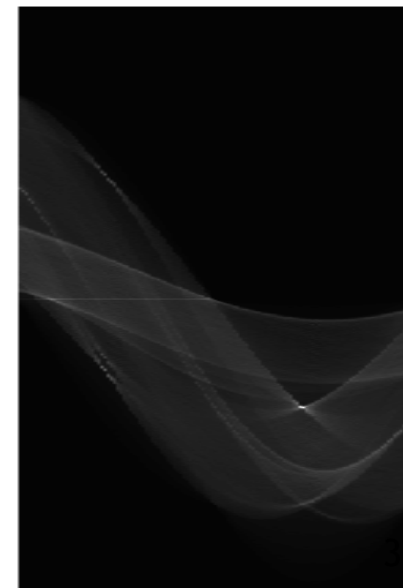
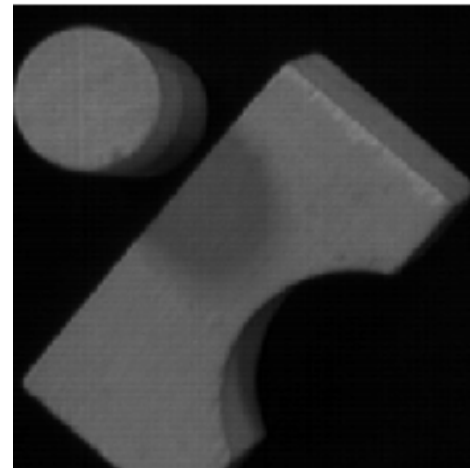
- Usual  $(m,b)$  parameter space problematic: can take on infinite values, undefined for vertical lines.



$d$  : perpendicular distance from line to origin

$\theta$  : angle the perpendicular makes with the x-axis

- Point in image space  $\Rightarrow$  sinusoid segment in Hough space



# Recap: Hough Transform for Circles

see  
Exercise 3.1!

- Circle: center  $(a,b)$  and radius  $r$

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius  $r$ , unknown gradient direction

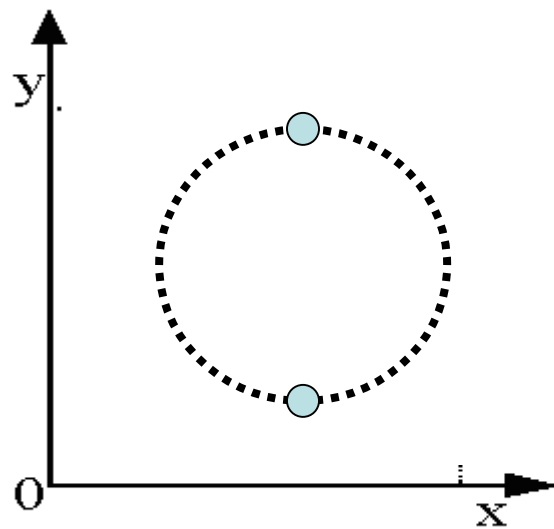
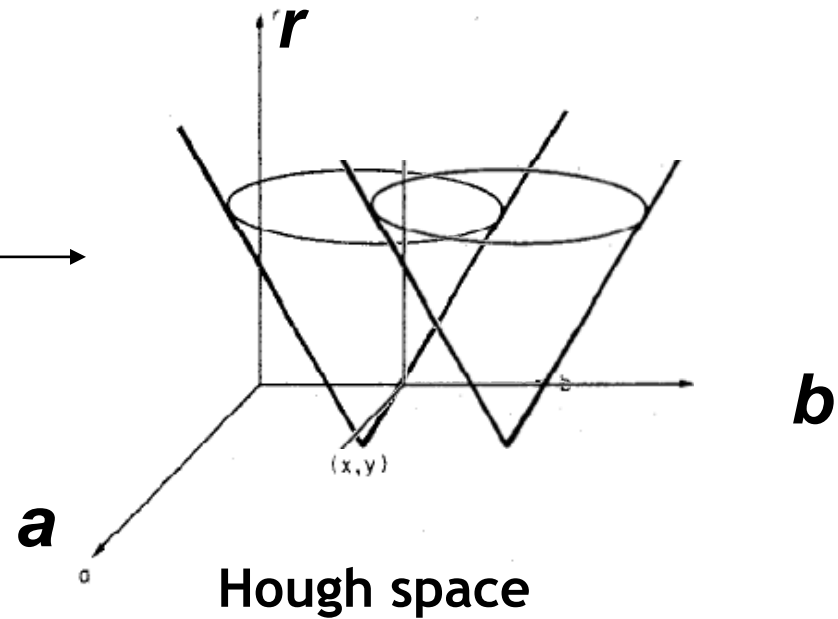
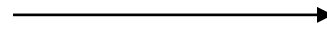


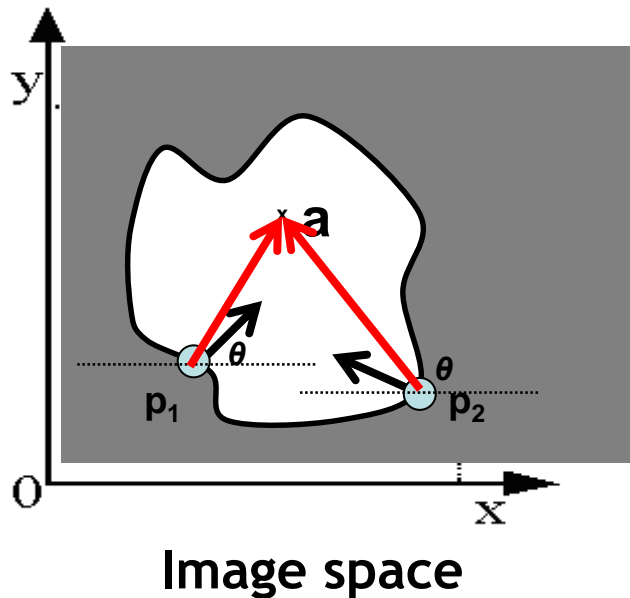
Image space



Hough space

# Recap: Generalized Hough Transform

- What if want to detect arbitrary shapes defined by boundary points and a reference point?



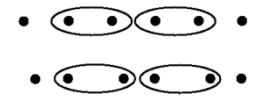
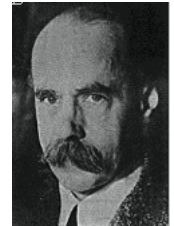
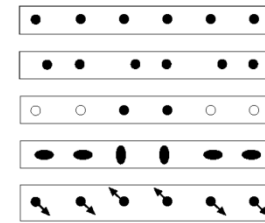
At each boundary point, compute displacement vector:  $r = a - p_i$ .

For a given model shape: store these vectors in a table indexed by gradient orientation  $\theta$ .

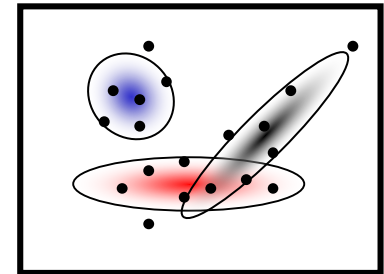
[Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]

# Repetition

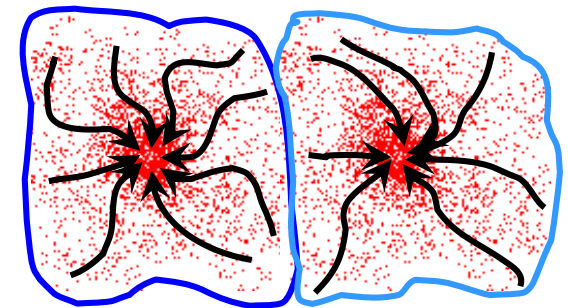
- Image Processing Basics
- Segmentation & Grouping
  - Segmentation and Grouping
  - Segmentation as Energy Minimization
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
- Motion and Tracking



*Gestalt factors*



*K-Means & EM clustering*



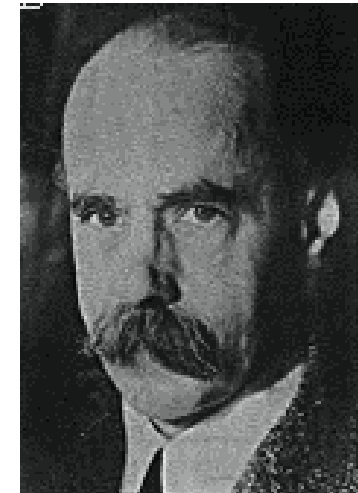
*Mean-shift clustering*

# Recap: Gestalt Theory

- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)

*“I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have “327”? No. I have sky, house, and trees.”*

**Max Wertheimer**  
(1880-1943)



Untersuchungen zur Lehre von der Gestalt,  
*Psychologische Forschung*, Vol. 4, pp. 301-350, 1923

<http://psy.ed.asu.edu/~classics/Wertheimer/Forms/forms.htm>

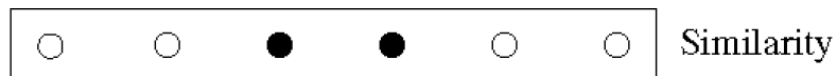
# Recap: Gestalt Factors



Not grouped



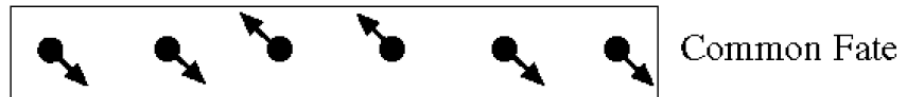
Proximity



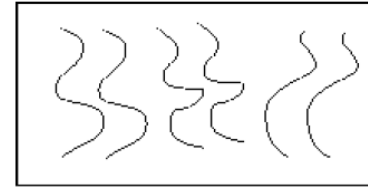
Similarity



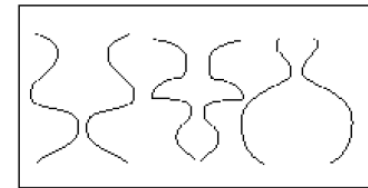
Similarity



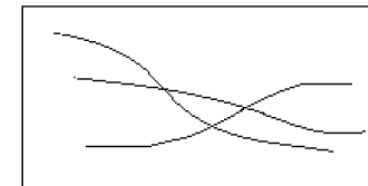
Common Fate



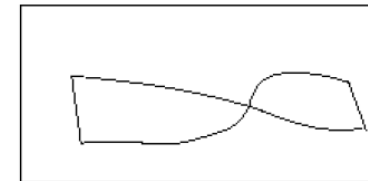
Parallelism



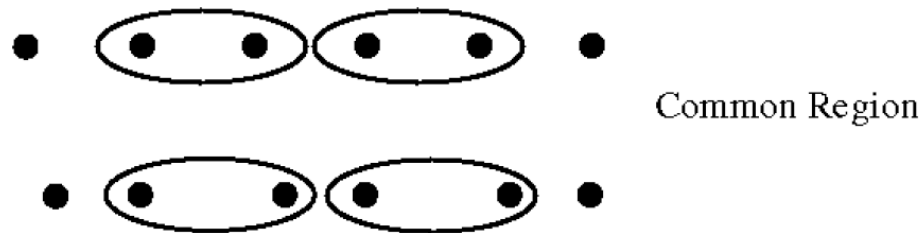
Symmetry



Continuity



Closure

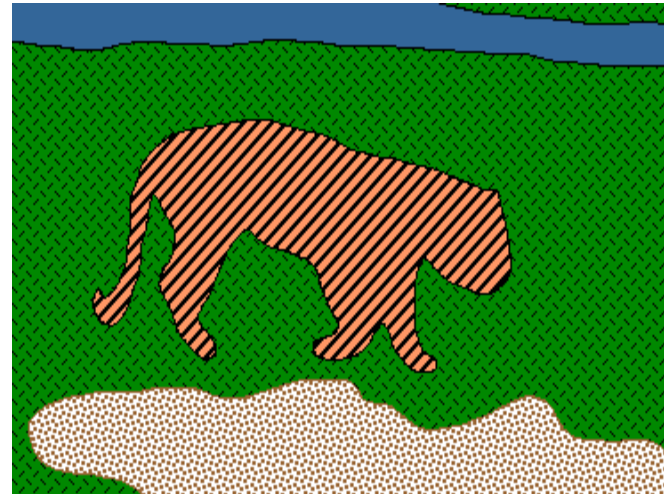


Common Region

- These factors make intuitive sense, but are very difficult to translate into algorithms.

# Recap: Image Segmentation

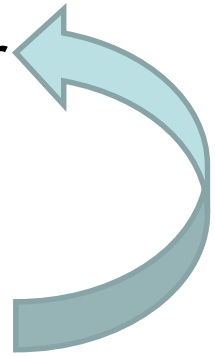
- Goal: identify groups of pixels that go together



# Recap: K-Means Clustering

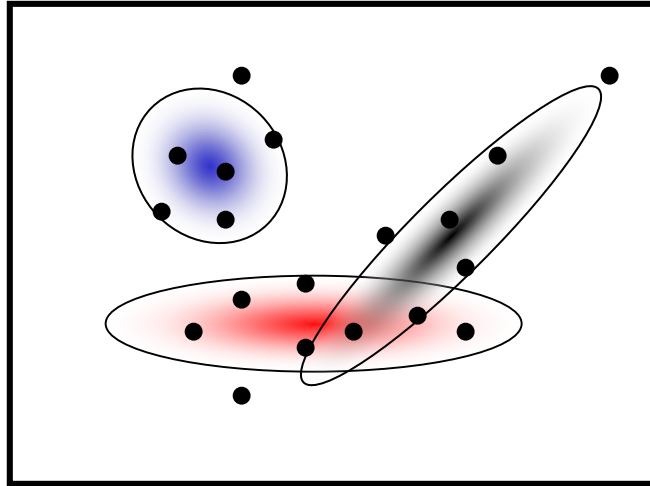
- **Basic idea:** randomly initialize the  $k$  cluster centers, and iterate between the two following steps
  1. Randomly initialize the cluster centers,  $c_1, \dots, c_k$
  2. Given cluster centers, determine points in each cluster
    - For each point  $p$ , find the closest  $c_i$ . Put  $p$  into cluster  $i$
  3. Given points in each cluster, solve for  $c_i$ 
    - Set  $c_i$  to be the mean of points in cluster  $i$
  4. If  $c_i$  have changed, repeat Step 2
- **Properties**
  - Will always converge to *some* solution
  - Can be a “local minimum”
    - Does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points } p \text{ in cluster } i} \|p - c_i\|^2$$





# Recap: Expectation Maximization (EM)



- **Goal**

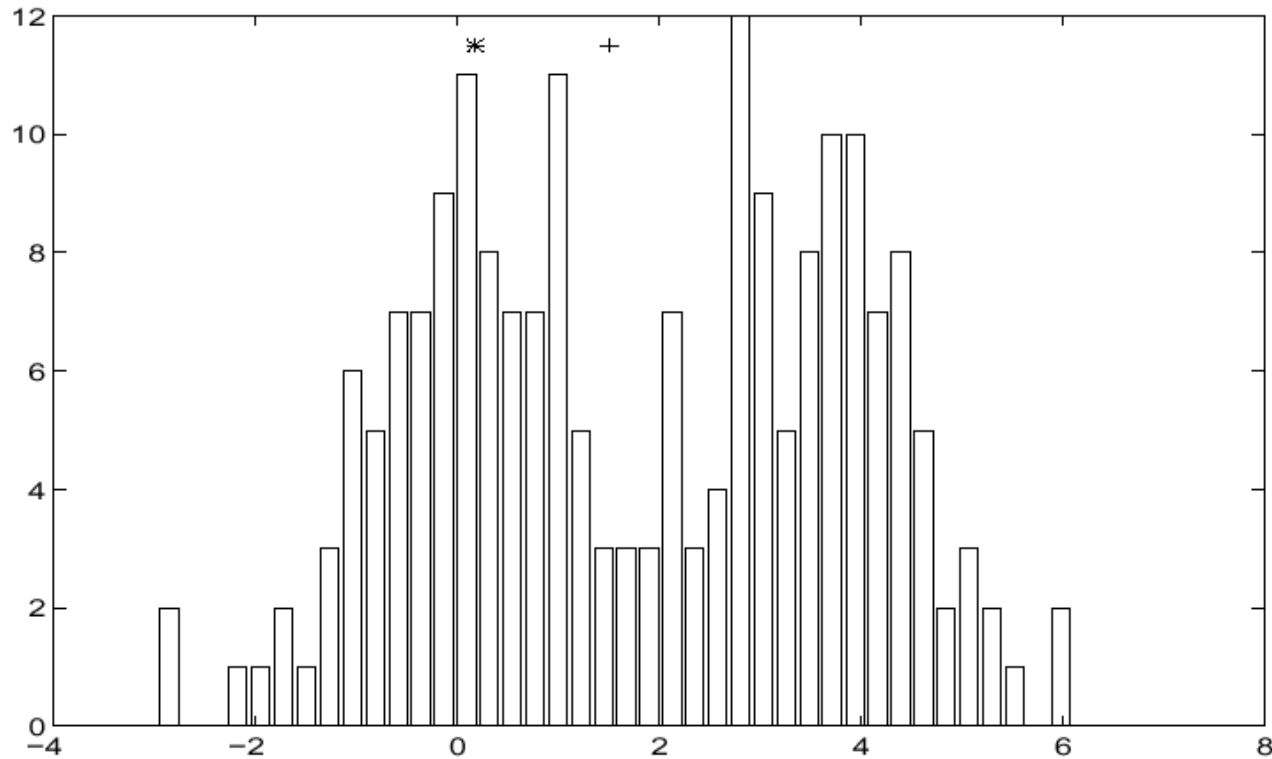
- Find blob parameters  $\theta$  that maximize the likelihood function:

$$p(\text{data}|\theta) = \prod_{n=1}^N p(\mathbf{x}_n|\theta)$$

- **Approach:**

1. **E-step:** given current guess of blobs, compute ownership of each point
2. **M-step:** given ownership probabilities, update blobs to maximize likelihood function
3. **Repeat until convergence**

# Recap: Mean-Shift Algorithm

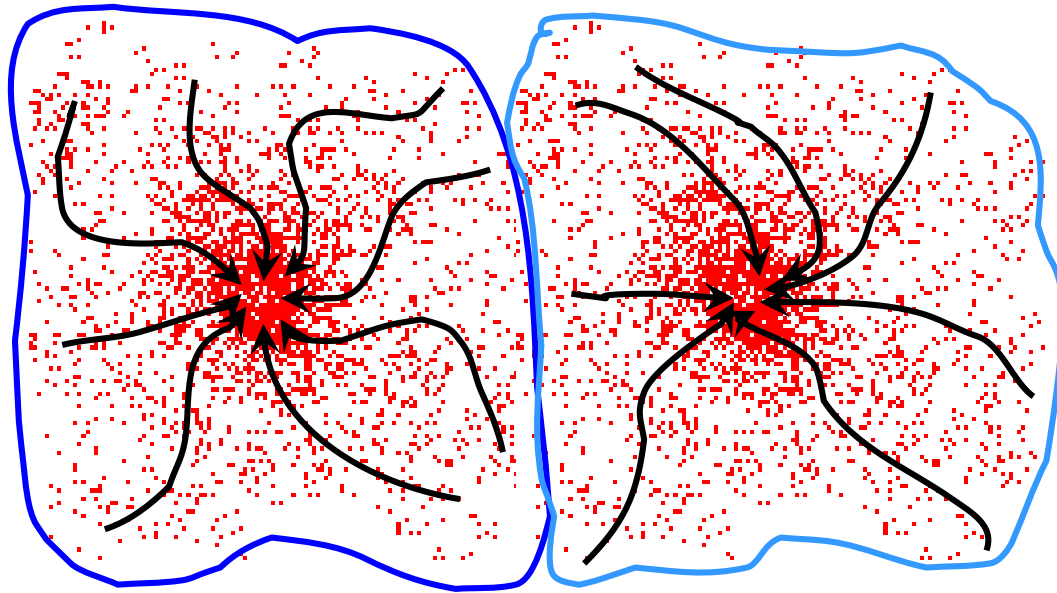


- **Iterative Mode Search**

1. Initialize random seed, and window  $W$
2. Calculate center of gravity (the “mean”) of  $W$ :  $\sum_{x \in W} xH(x)$
3. Shift the search window to the mean
4. Repeat Step 2 until convergence

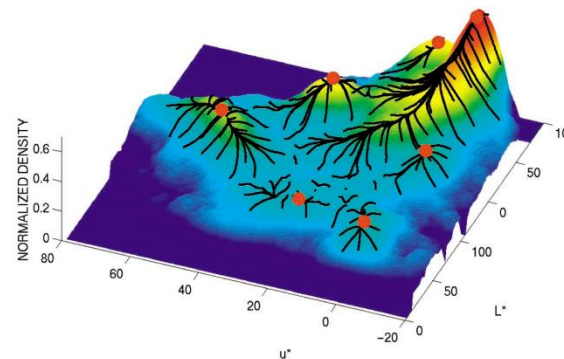
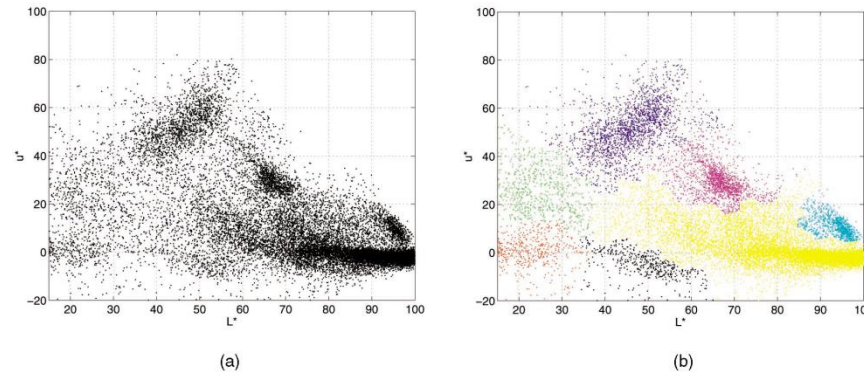
# Recap: Mean-Shift Clustering

- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



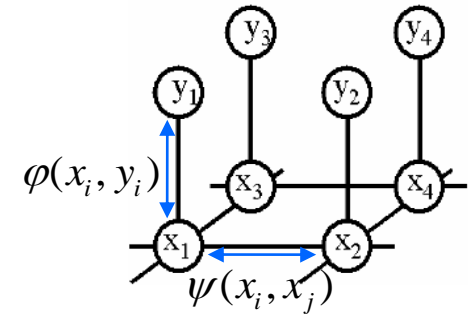
# Recap: Mean-Shift Segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode

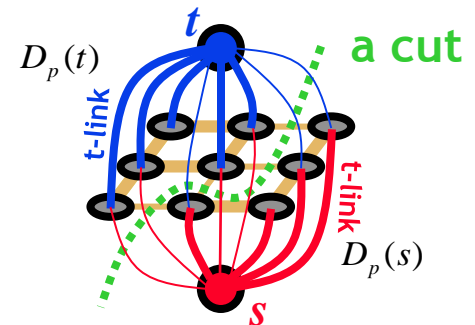


# Repetition

- Image Processing Basics
- Segmentation & Grouping
  - Segmentation and Grouping
  - Segmentation as Energy Minimization
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
- Motion and Tracking



*Markov Random Fields*



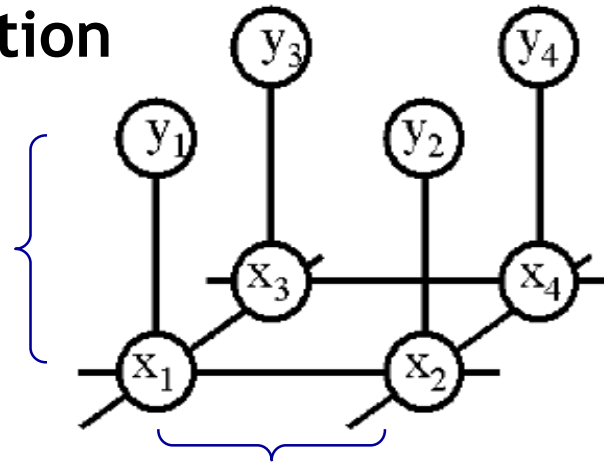
*Graph cuts*

# Recap: MRFs for Image Segmentation

- MRF formulation

Unary potentials

$$\phi(x_i, y_i)$$



Pairwise potentials

$$\psi(x_i, x_j)$$

⇒ Minimize the energy

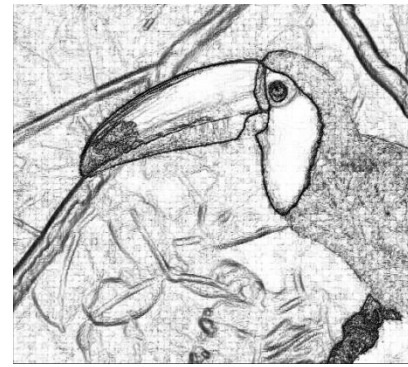
$$E(\mathbf{x}, \mathbf{y}) = \sum_i \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j)$$



Data (D)



Unary likelihood



Pair-wise Terms

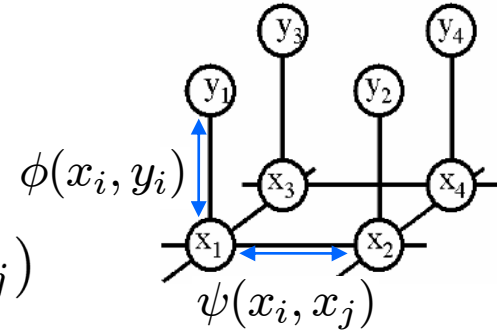


MAP Solution

# Recap: Energy Formulation

- Energy function

$$E(\mathbf{x}, \mathbf{y}) = \sum_i \underbrace{\phi(x_i, y_i)}_{\text{Unary potentials}} + \sum_{i,j} \underbrace{\psi(x_i, x_j)}_{\text{Pairwise potentials}}$$

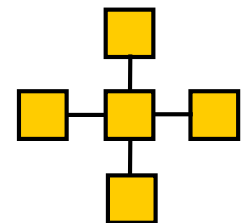


- Unary potentials  $\phi$

- Encode local information about the given pixel/patch
- How likely is a pixel/patch to belong to a certain class (e.g. foreground/background)?

- Pairwise potentials  $\psi$

- Encode neighborhood information
- How different is a pixel/patch's label from that of its neighbor? (e.g. based on intensity/color/texture difference, edges)



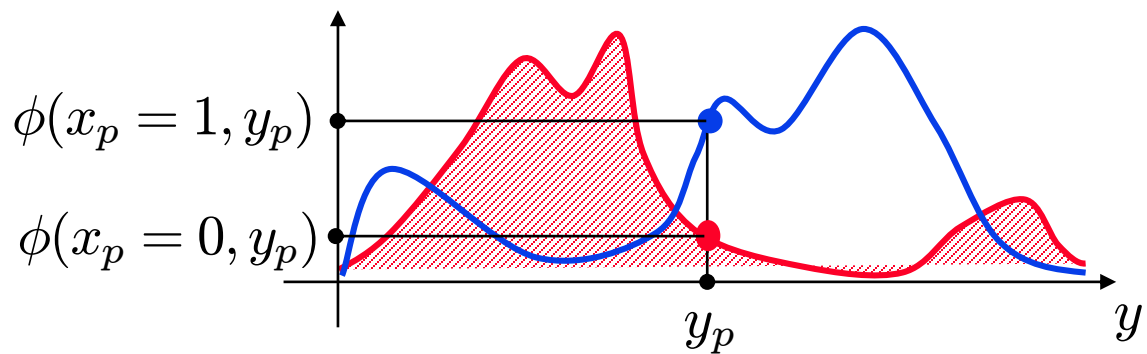
# Recap: How to Set the Potentials?

- Unary potentials

- E.g. color model, modeled with a Mixture of Gaussians

$$\phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_\phi(x_i, k) p(k|x_i) \mathcal{N}(y_i; \bar{y}_k, \Sigma_k)$$

⇒ Learn color distributions for each label





# Recap: How to Set the Potentials?

- Pairwise potentials

- Potts Model

$$\psi(x_i, x_j; \theta_\psi) = \theta_\psi \delta(x_i \neq x_j)$$

- Simplest discontinuity preserving model.
- Discontinuities between any pair of labels are penalized equally.
- Useful when labels are unordered or number of labels is small.

- Extension: “Contrast sensitive Potts model”

$$\psi(x_i, x_j, g_{ij}(y); \theta_\psi) = \theta_\psi g_{ij}(y) \delta(x_i \neq x_j)$$

where

$$g_{ij}(y) = e^{-\beta \|y_i - y_j\|^2} \quad \beta = 2 / \text{avg} \left( \|y_i - y_j\|^2 \right)$$

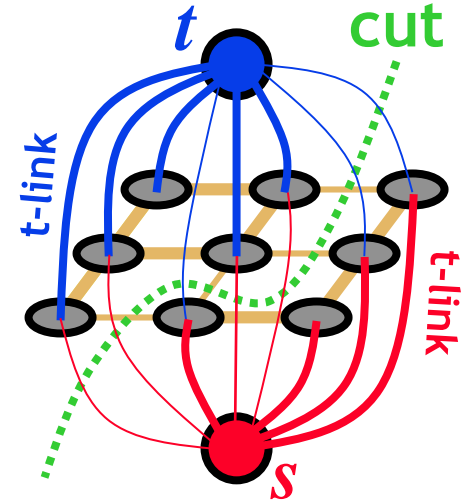
⇒ Discourages label changes except in places where there is also a large change in the observations.

# Recap: Graph-Cuts Energy Minimization

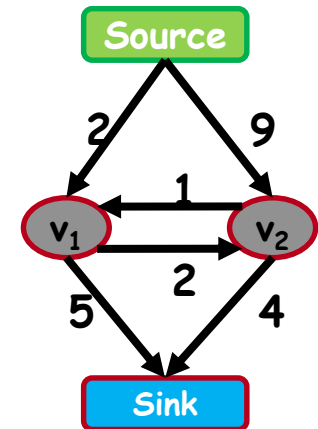
see  
Exercise 3.4!

- Solve an equivalent graph cut problem
  1. Introduce extra nodes: source and sink
  2. Weight connections to source/sink (t-links) by  $\phi(x_i = s)$  and  $\phi(x_i = t)$ , respectively.
  3. Weight connections between nodes (n-links) by  $\psi(x_i, x_j)$ .
  4. Find the minimum cost cut that separates source from sink.

⇒ Solution is equivalent to minimum of the energy.



- s-t Mincut can be solved efficiently
  - Dual to the well-known max flow problem
  - Very efficient algorithms available for regular grid graphs (1-2 MPixels/s)
  - Globally optimal result for 2-class problems



# Recap: When Can s-t Graph Cuts Be Applied?

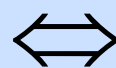
$$E(L) = \sum_p E_p(L_p) + \sum_{pq \in N} E(L_p, L_q)$$

Unary potentials                      Pairwise potentials

t-links                                      n-links                       $L_p \in \{s, t\}$

- s-t graph cuts can only globally minimize **binary energies** that are **submodular**. [Boros & Hummer, 2002, Kolmogorov & Zabih, 2004]

$E(L)$  can be minimized  
by s-t graph cuts



$$E(s, s) + E(t, t) \leq E(s, t) + E(t, s)$$

Submodularity (“convexity”)

- Submodularity is the discrete equivalent to convexity.
  - Implies that every local energy minimum is a global minimum.
  - ⇒ Solution will be globally optimal.

# First Applications Take Up Shape...



Line  
detection

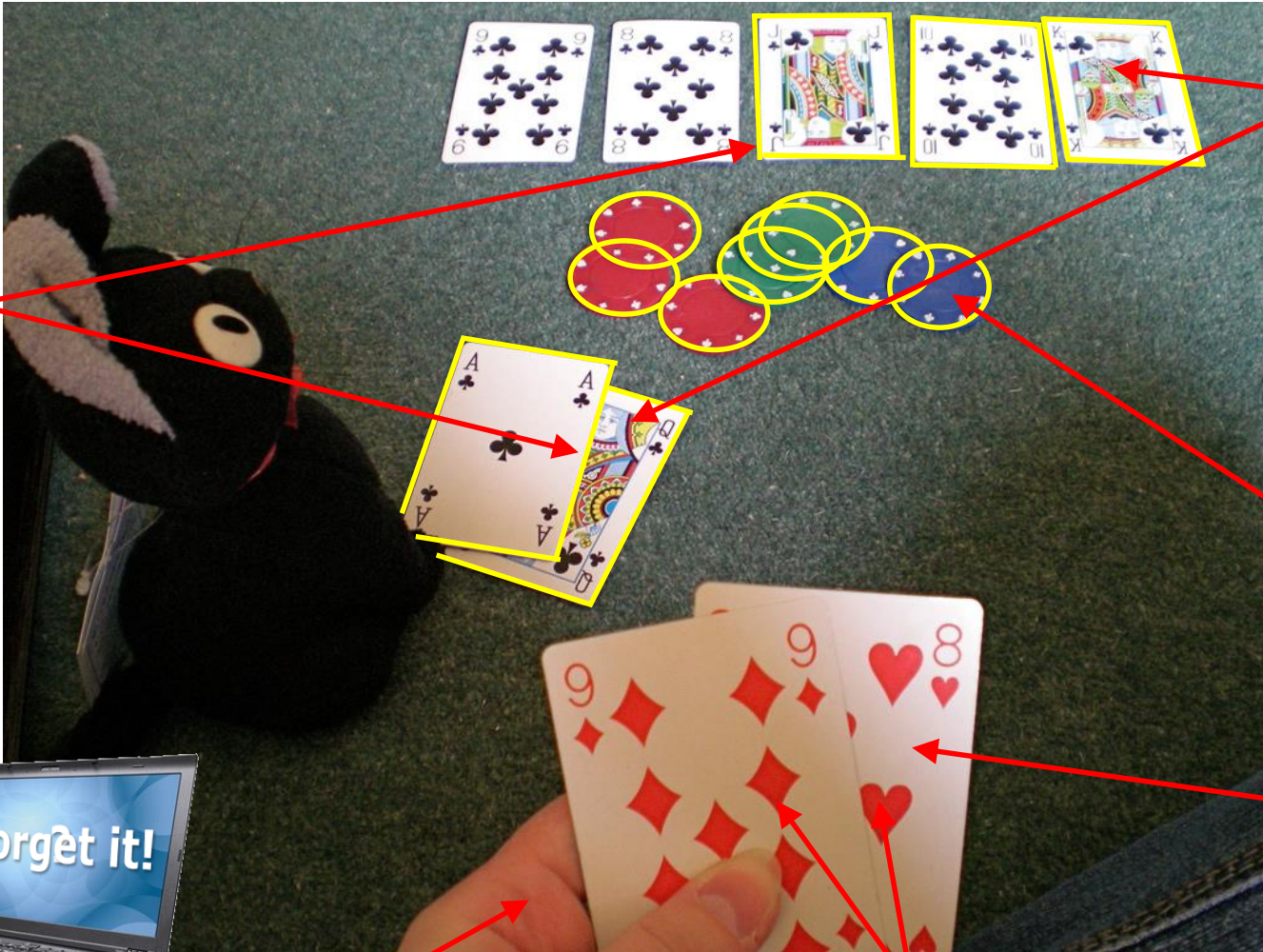


Image  
recognition

Circle  
detection

Binary  
Segmen-  
tation

Skin color detection

Moment descriptors

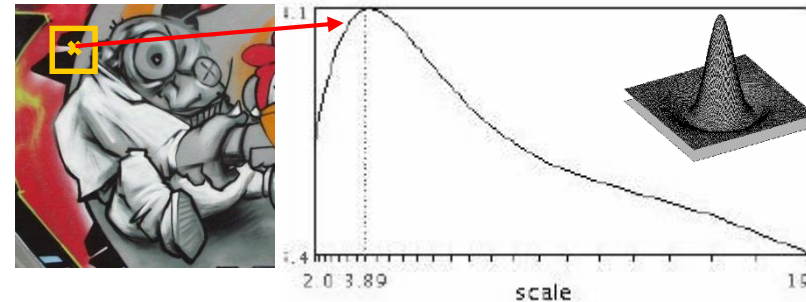
# Repetition

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
  - Local Features - Detection and Description
  - Recognition with Local Features
- Object Categorization
- 3D Reconstruction
- Motion and Tracking

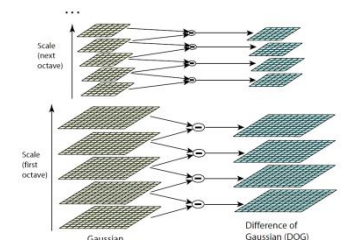
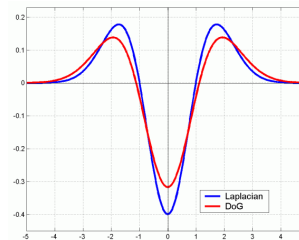
$$M(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

*Harris & Hessian detector*

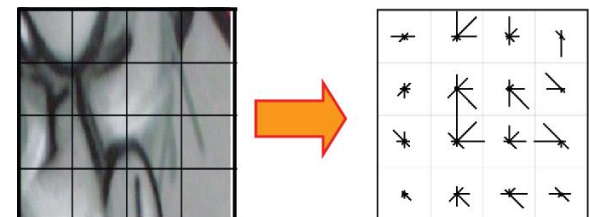
$$Hes(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$



*Laplacian scale selection*



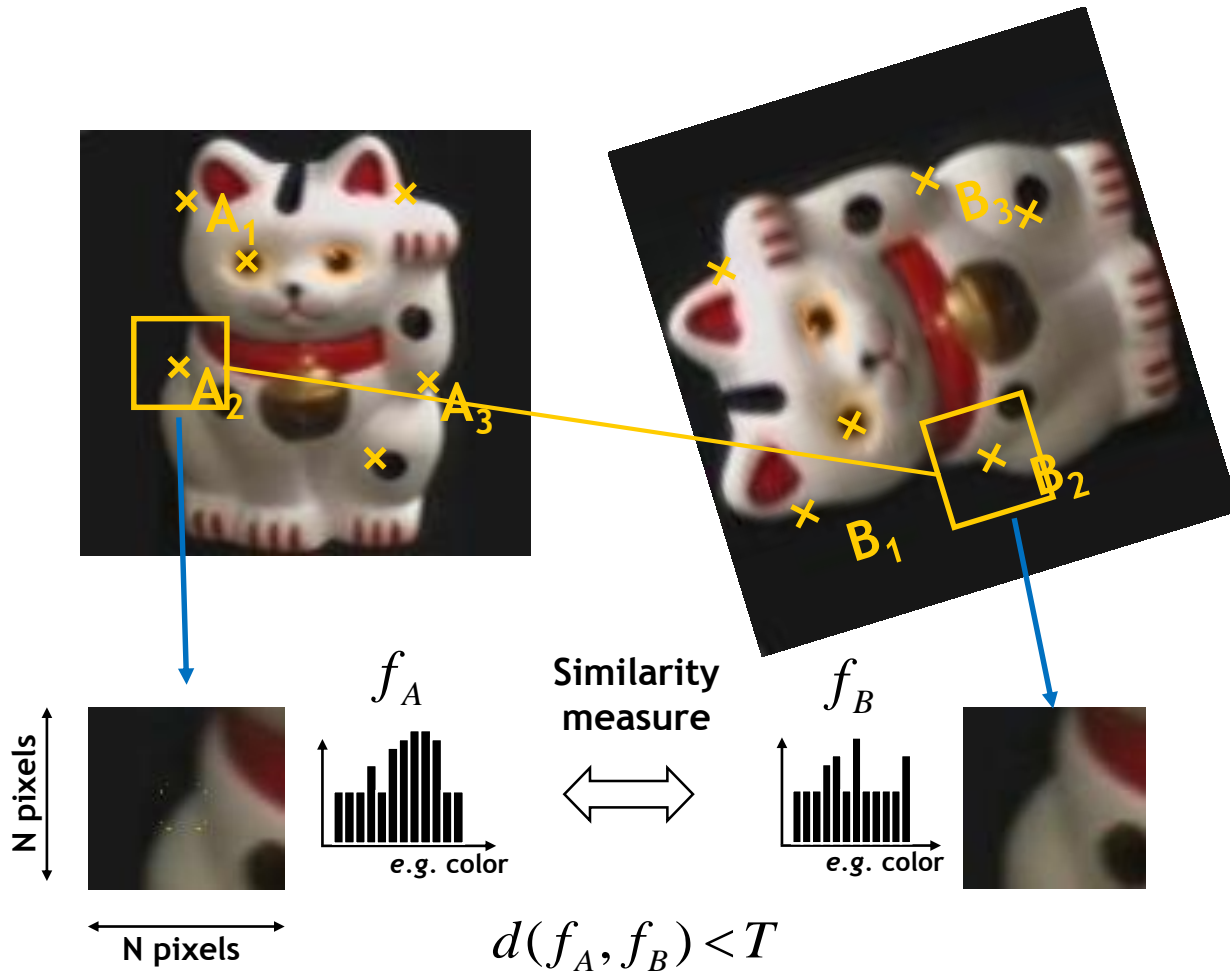
*Difference-of-Gaussian (DOG)*



*SIFT descriptor*



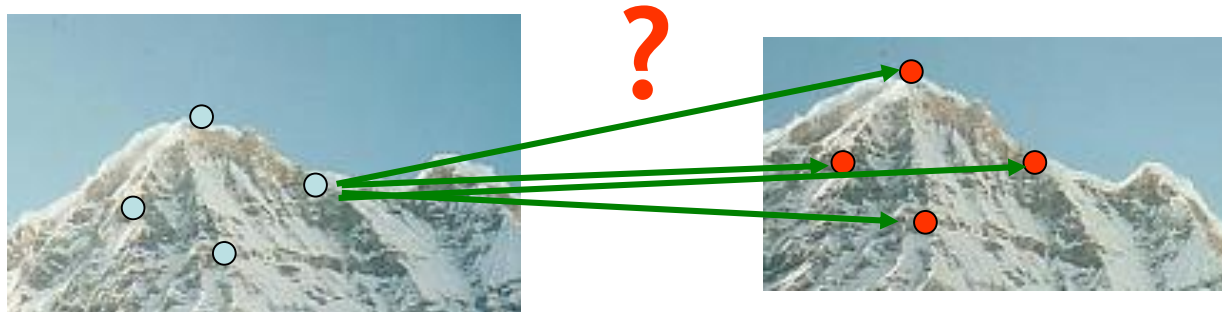
# Recap: Local Feature Matching Pipeline



1. Find a set of distinctive keypoints
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

# Recap: Requirements for Local Features

- Problem 1:
  - Detect the same point *independently* in both images
- Problem 2:
  - For each point correctly recognize the corresponding one



**We need a repeatable detector!**

**We need a reliable and distinctive descriptor!**

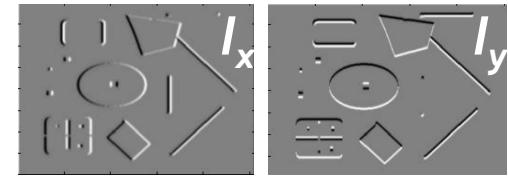
see  
Exercise 5.2!

# Recap: Harris Detector [Harris88]

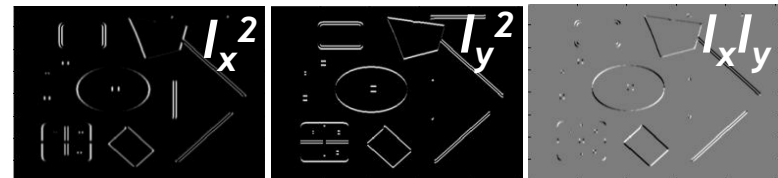
- Compute second moment matrix (autocorrelation matrix)

$$M(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives



2. Square of derivatives



3. Gaussian filter  $g(\sigma_I)$



4. Cornerness function - two strong eigenvalues

$$\begin{aligned} R &= \det[M(\sigma_I, \sigma_D)] - \alpha[\text{trace}(M(\sigma_I, \sigma_D))] \\ &= g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2 \end{aligned}$$

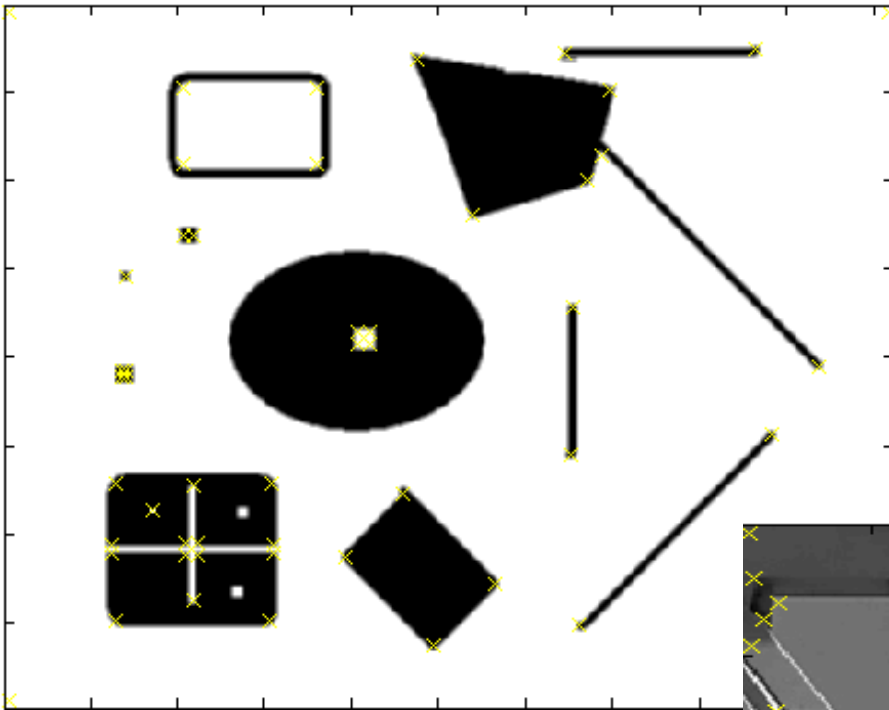
5. Perform non-maximum suppression



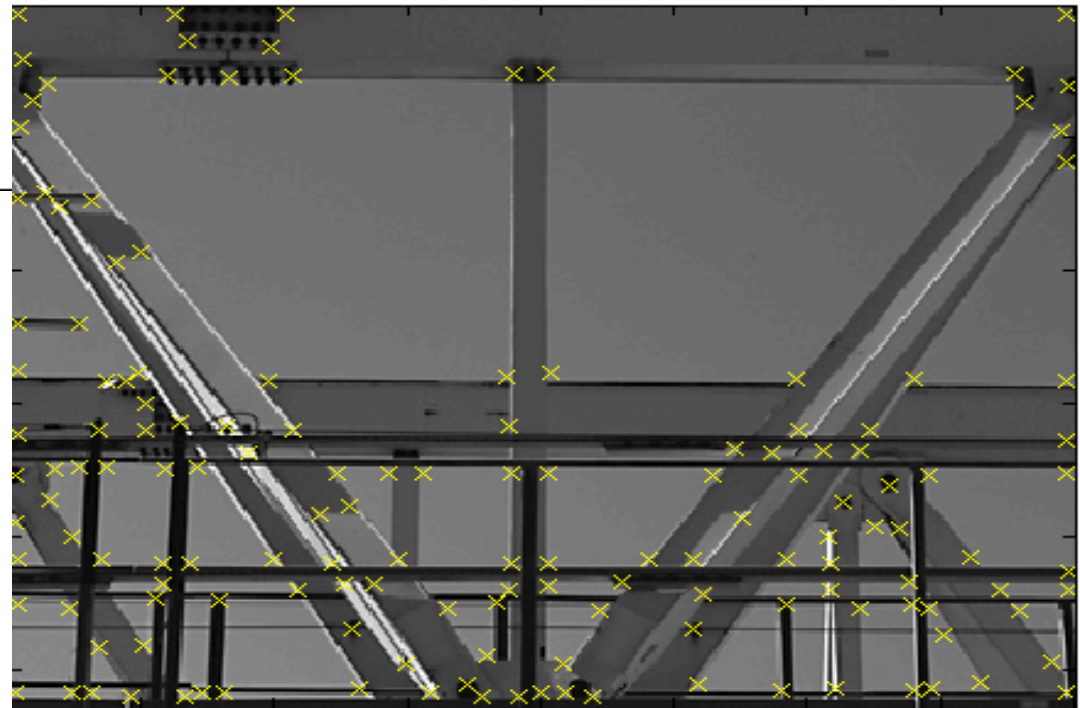
R 60



# Recap: Harris Detector Responses [Harris88]



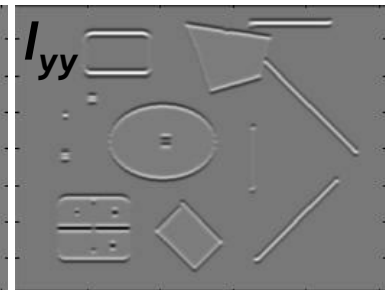
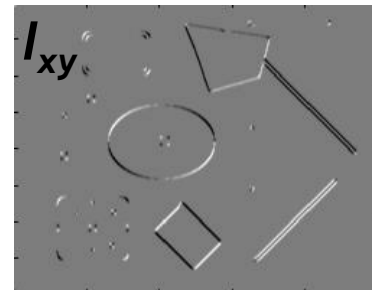
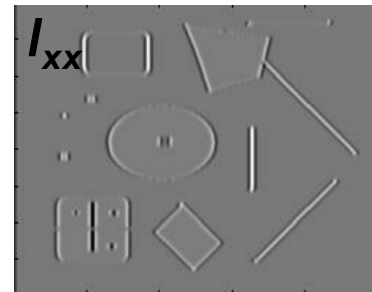
*Effect:* A very precise corner detector.



# Recap: Hessian Detector [Beaudet78]

- Hessian determinant

$$\text{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$



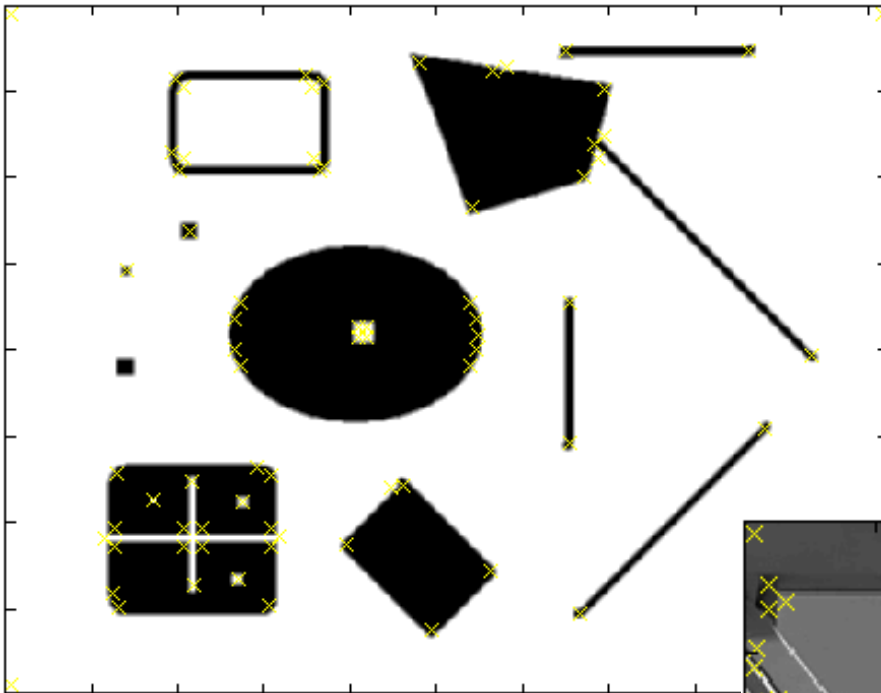
$$\det(\text{Hessian}(I)) = I_{xx}I_{yy} - I_{xy}^2$$

In Matlab:

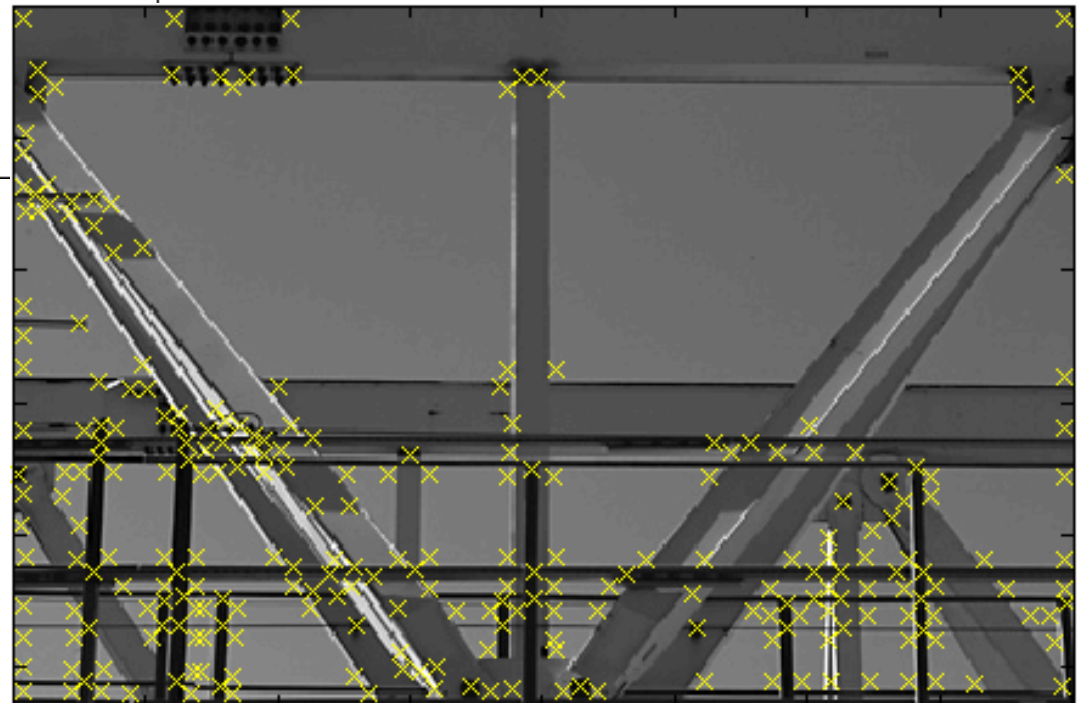
$$I_{xx} \cdot I_{yy} - (I_{xy})^2$$



# Recap: Hessian Detector Responses [Beaudet78]

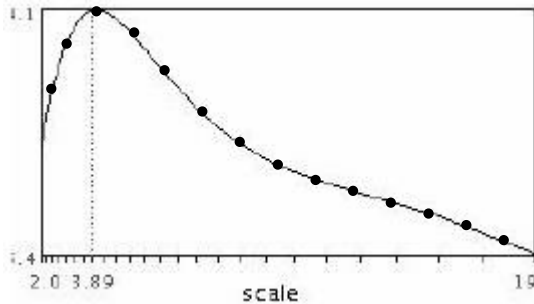
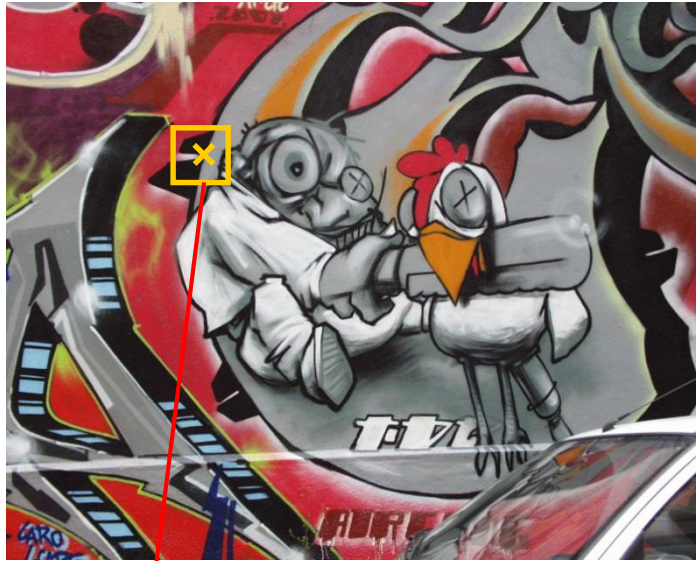


**Effect:** Responses mainly on corners and strongly textured areas.

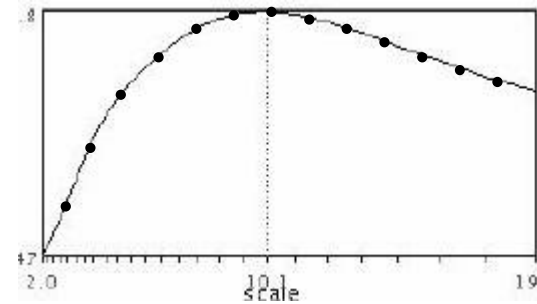


# Recap: Automatic Scale Selection

- Function responses for increasing scale (scale signature)



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

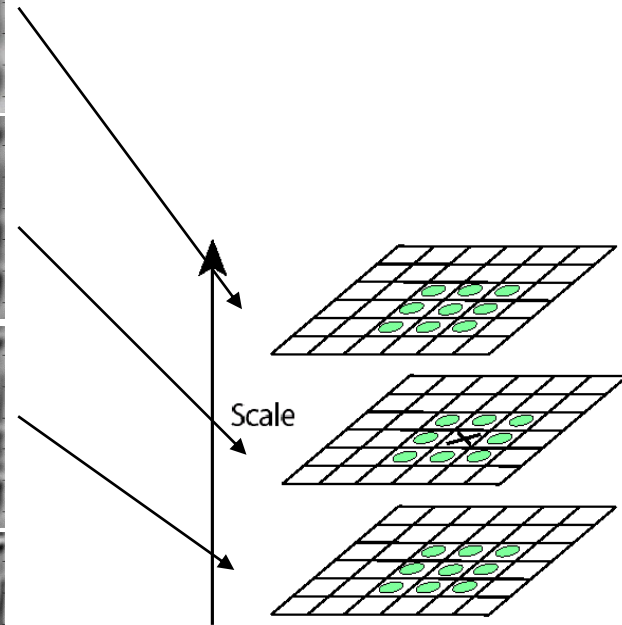
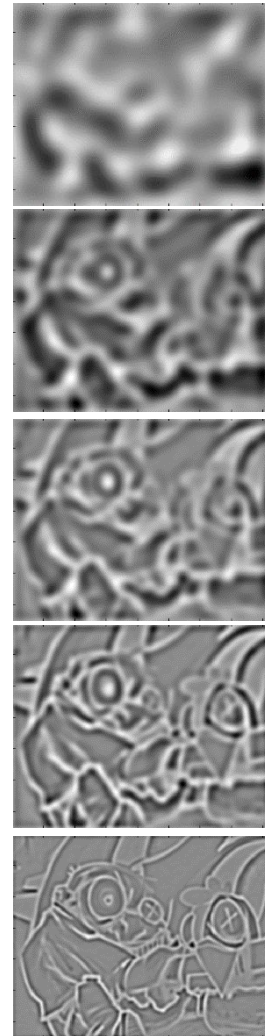
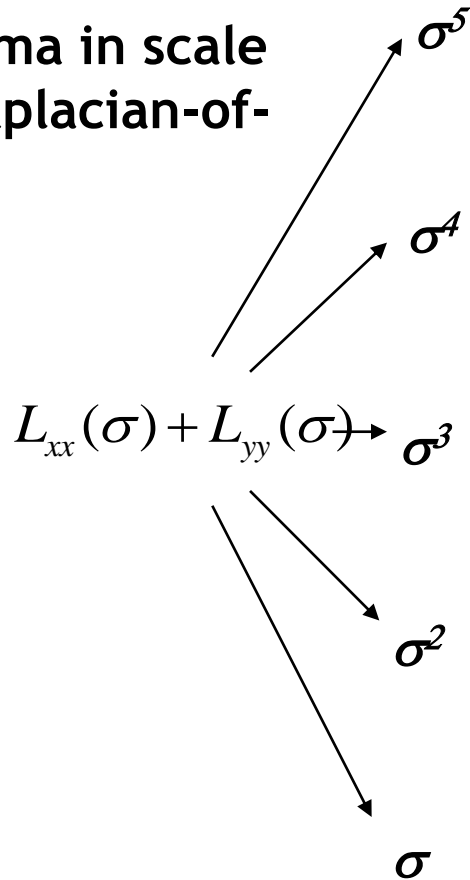
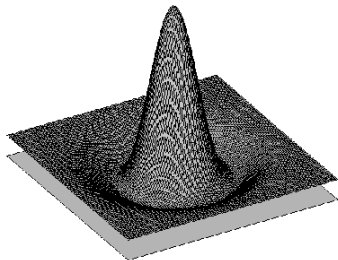


$$f(I_{i_1...i_m}(x', \sigma'))$$

# Recap: Laplacian-of-Gaussian (LoG)

- Interest points:

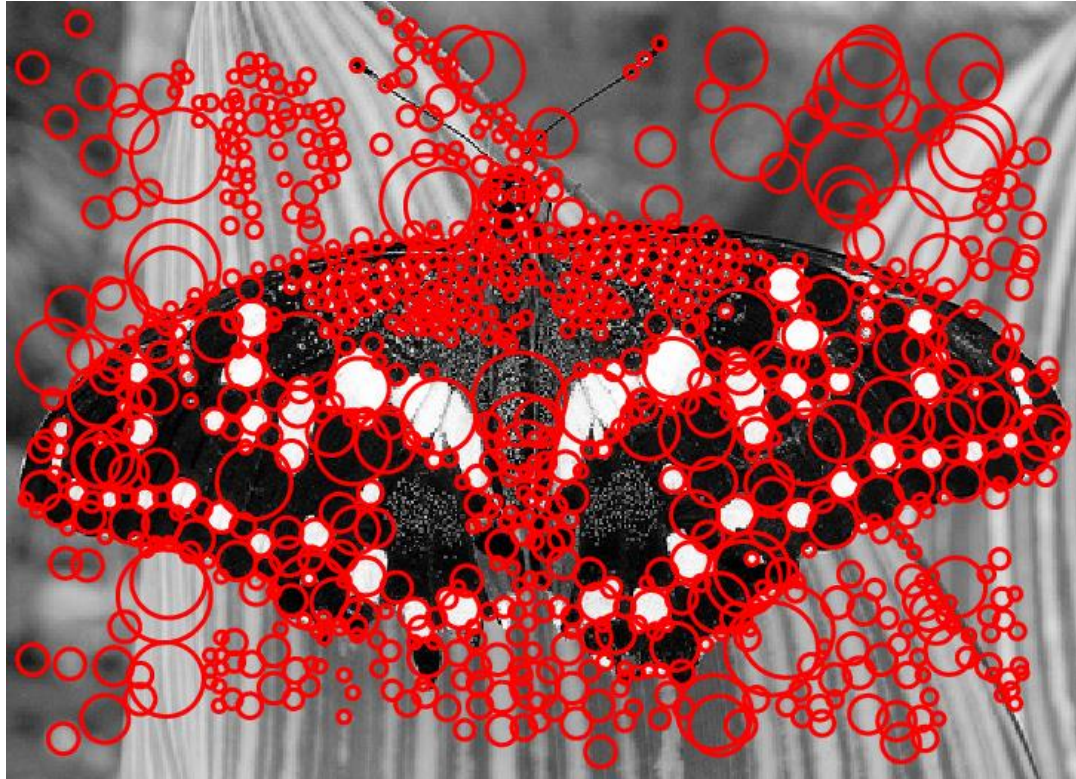
- Local maxima in scale space of Laplacian-of-Gaussian



⇒ List of  $(x, y, \sigma)$

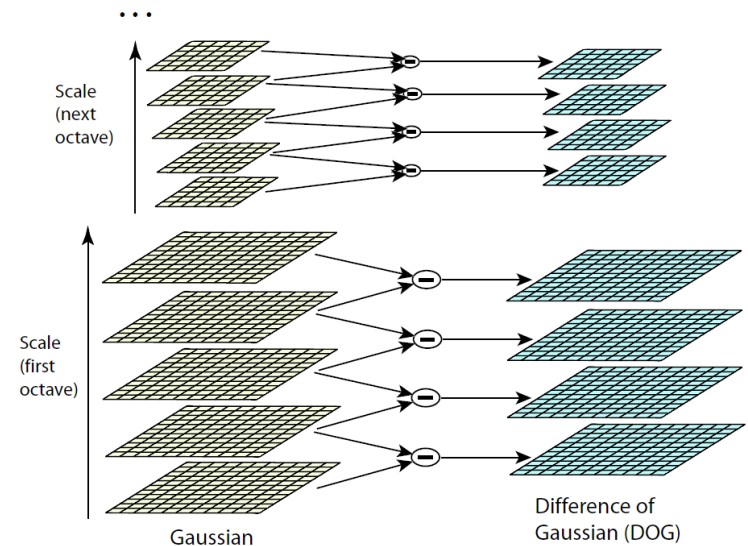
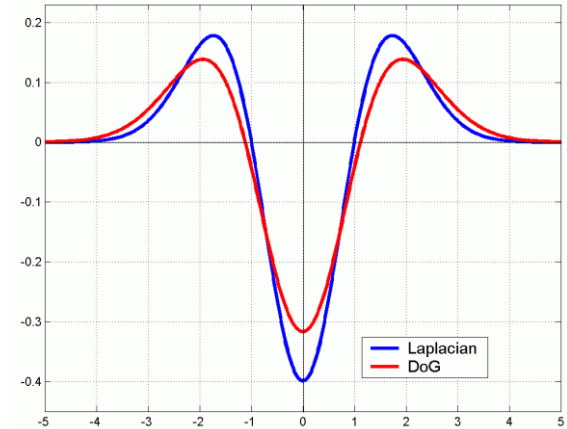


# Recap: LoG Detector Responses



# Recap: Key point localization with DoG

- **Efficient implementation**
  - Approximate LoG with a difference of Gaussians (DoG)
- **Approach DoG Detector**
  - Detect maxima of difference-of-Gaussian in scale space
  - Reject points with low contrast (threshold)
  - Eliminate edge responses

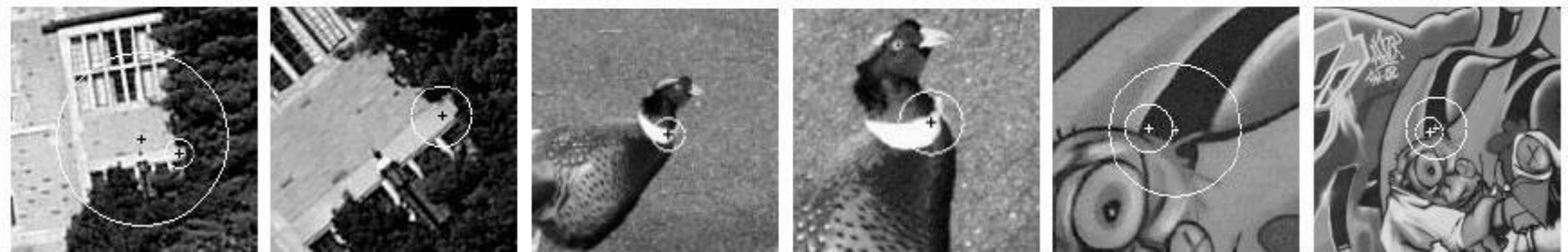
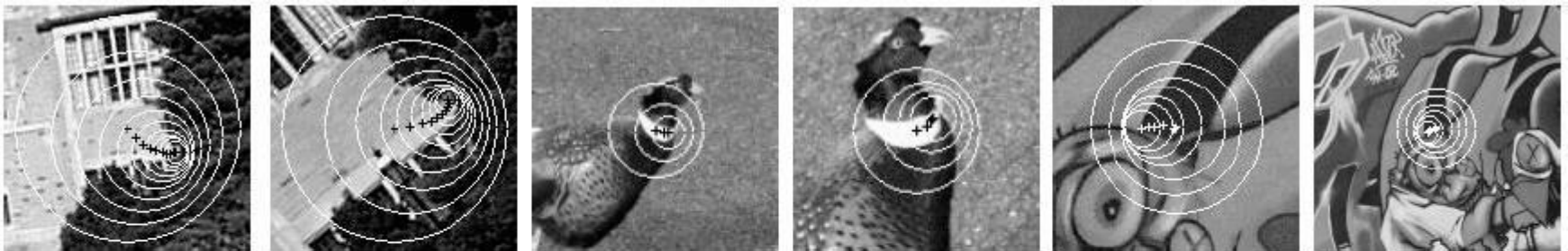


**Candidate keypoints:  
list of  $(x, y, \sigma)$**

# Recap: Harris-Laplace [Mikolajczyk '01]

1. Initialization: Multiscale Harris corner detection
2. Scale selection based on Laplacian  
(same procedure with Hessian  $\Rightarrow$  Hessian-Laplace)

Harris points



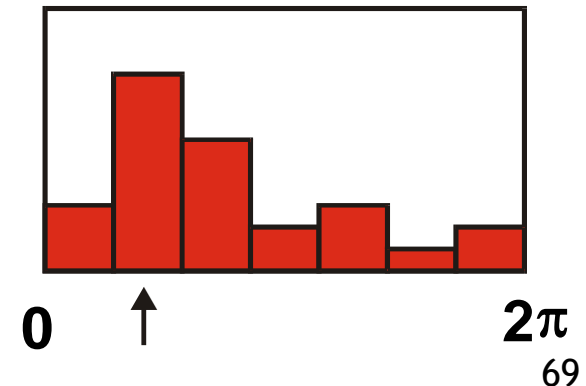
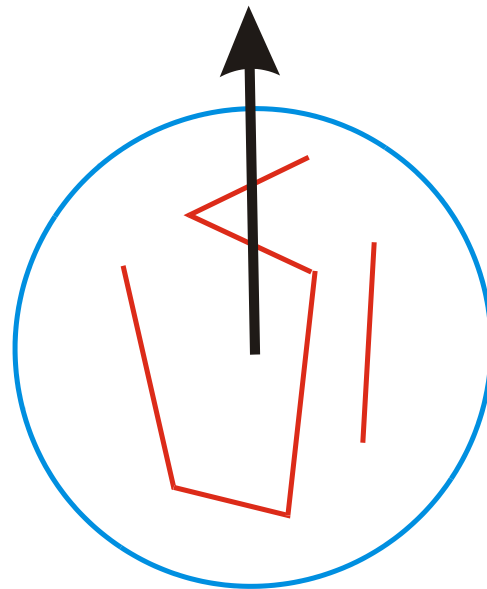
Harris-Laplace points



# Recap: Orientation Normalization

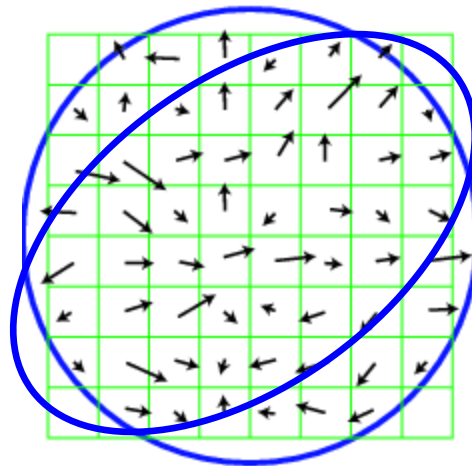
- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation

[Lowe, SIFT, 1999]



# Recap: Affine Adaptation

- **Problem:**
  - Determine the characteristic shape of the region.
  - Assumption: shape can be described by “local affine frame”.
- **Solution: iterative approach**
  - Use a circular window to compute second moment matrix.
  - Compute eigenvectors to adapt the circle to an ellipse.
  - Recompute second moment matrix using new window and iterate...



# Recap: Iterative Affine Adaptation



1. Detect keypoints, e.g. multi-scale Harris
2. Automatically select the scales
3. Adapt affine shape based on second order moment matrix
4. Refine point location

# Recap: Affine-Inv. Feature Extraction

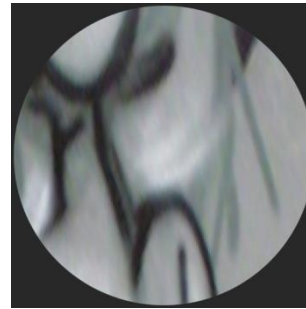
Extract affine regions



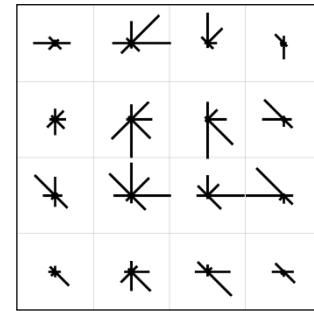
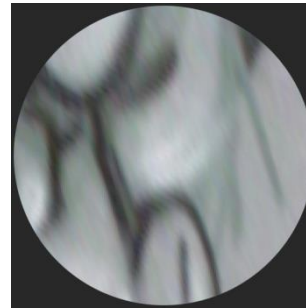
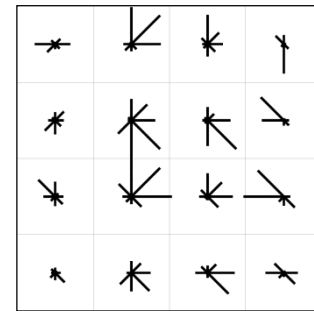
Normalize regions



Eliminate rotational ambiguity

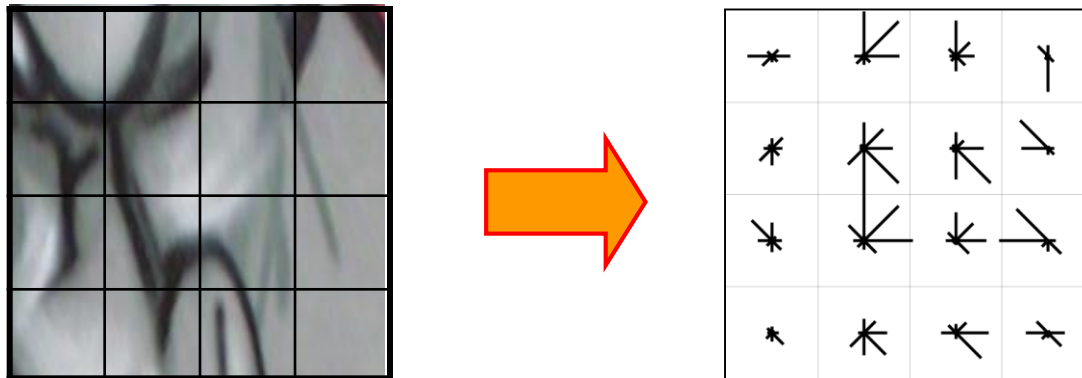


Compare descriptors



# Recap: SIFT Feature Descriptor

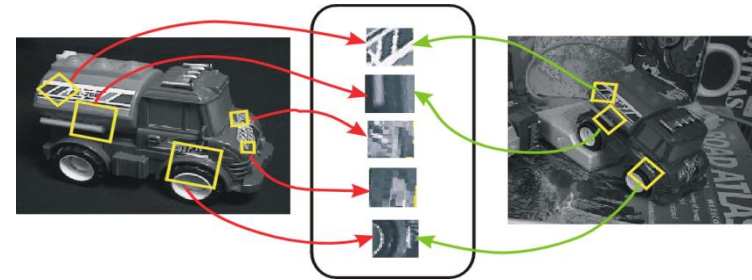
- **S**cale **I**nvariant **F**eature **T**ransform
- **D**escriptor computation:
  - Divide patch into 4x4 sub-patches: 16 cells
  - Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch
  - Resulting descriptor:  $4 \times 4 \times 8 = 128$  dimensions



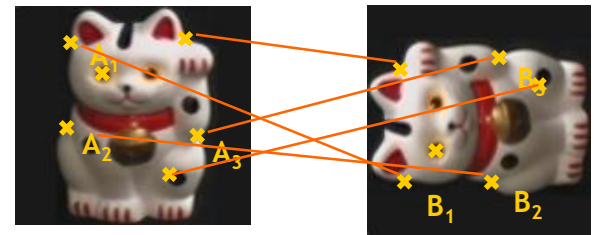
David G. Lowe. "Distinctive image features from scale-invariant keypoints."  
*IJCV* 60 (2), pp. 91-110, 2004.

# Repetition

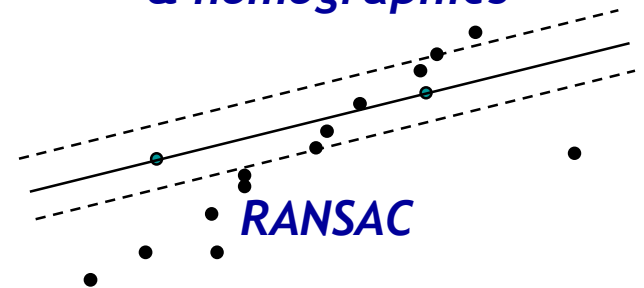
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
  - Local Features - Detection and Description
  - Recognition with Local Features
- Object Categorization
- 3D Reconstruction
- Motion and Tracking



*Recognition pipeline*



*Fitting affine transformations & homographies*

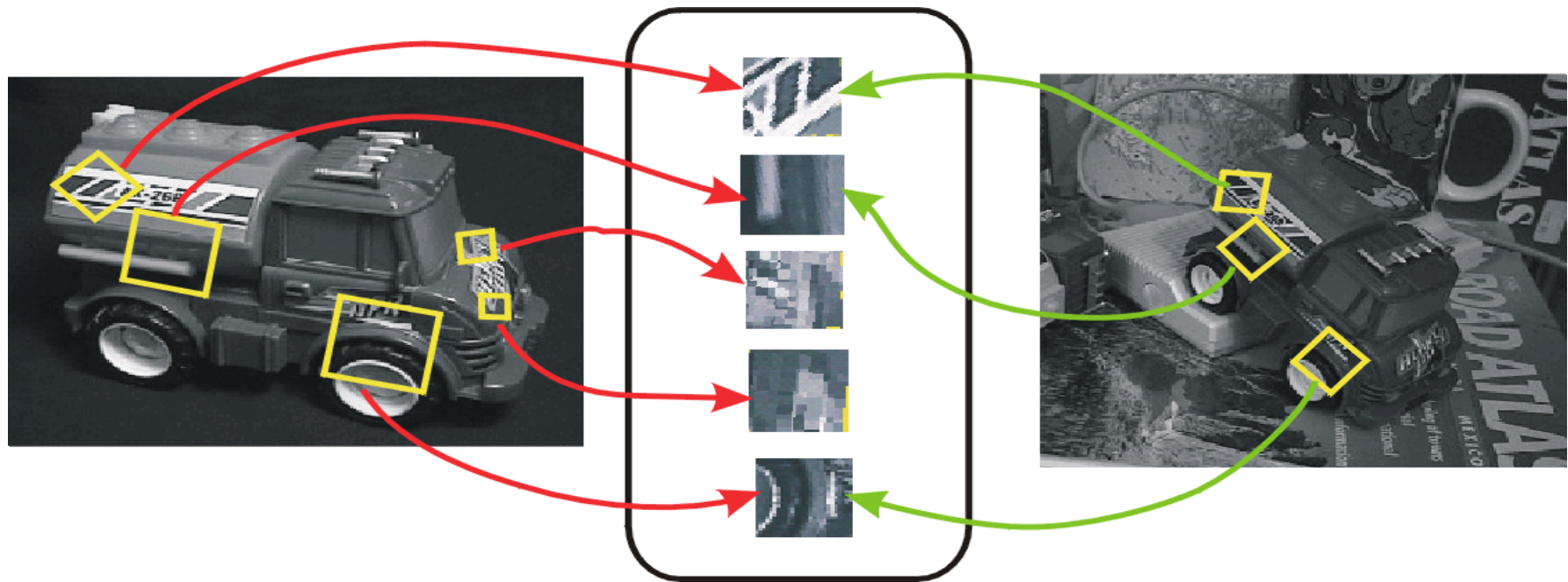


*Gen. Hough Transform*



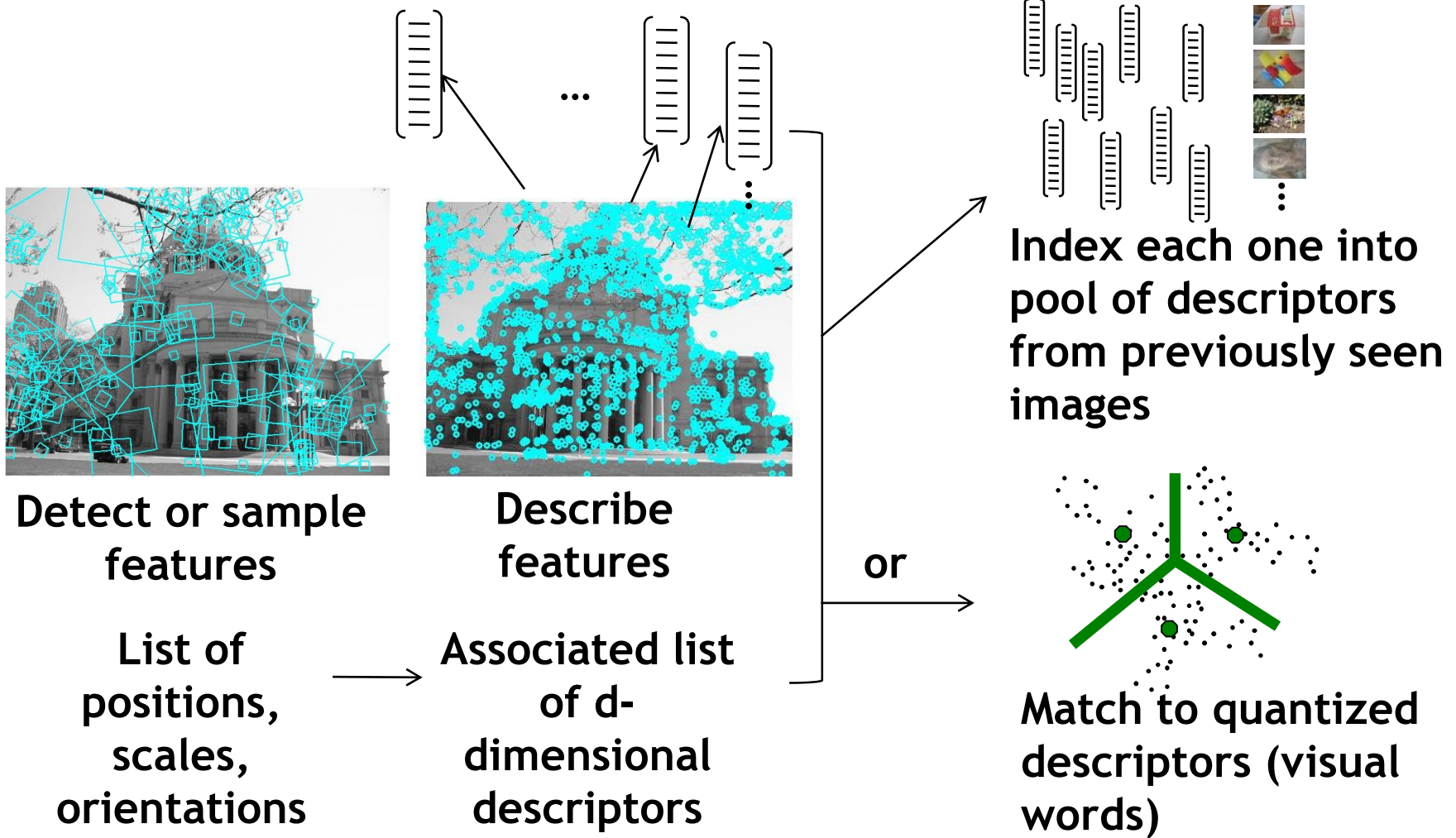
# Recap: Recognition with Local Features

- Image content is transformed into local features that are invariant to translation, rotation, and scale
- Goal: Verify if they belong to a consistent configuration



Local Features,  
e.g. SIFT

# Recap: Indexing features



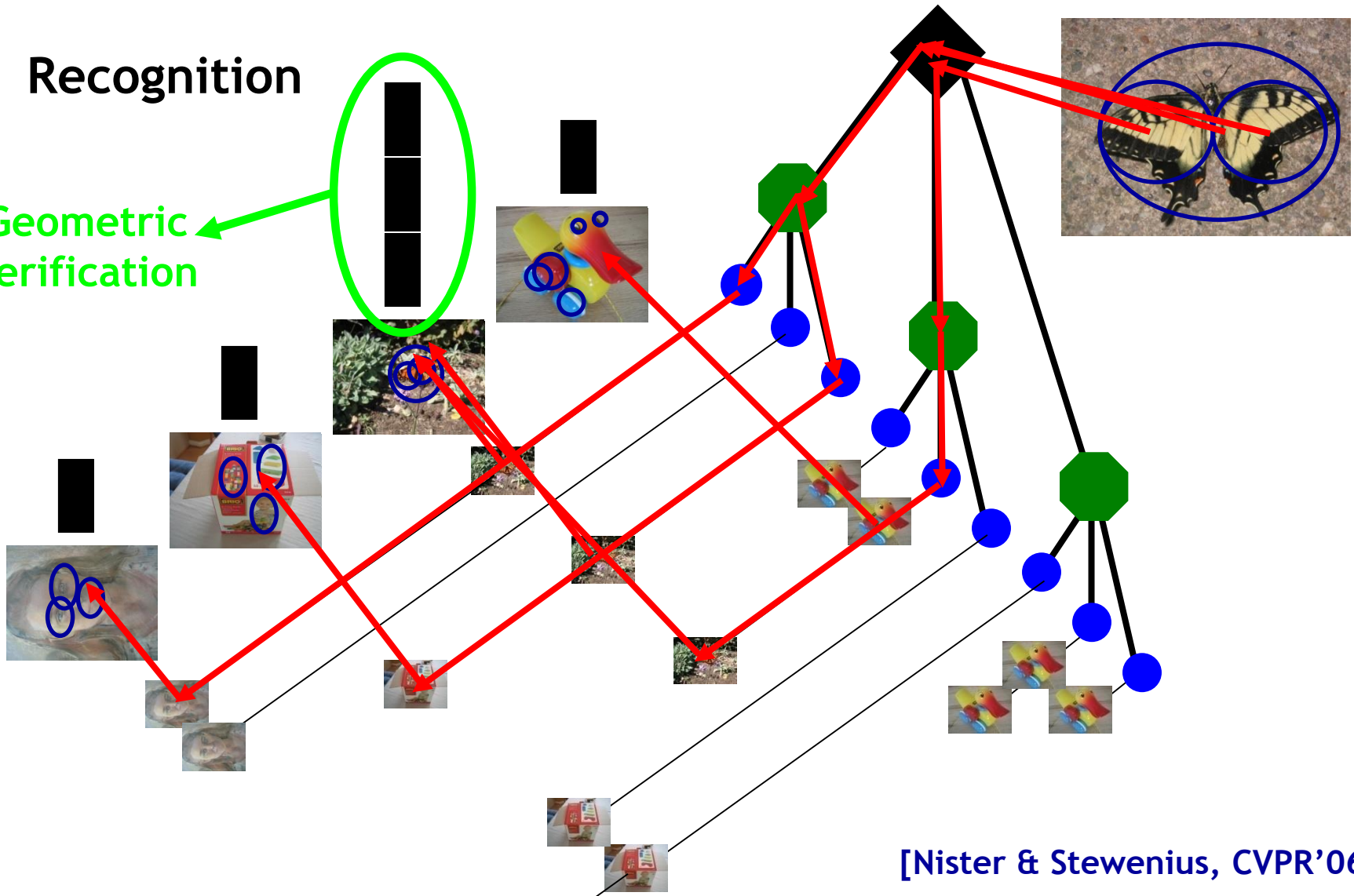
⇒ *Shortlist of possibly matching images + feature correspondences*



# Recap: Fast Indexing with Vocabulary Trees

- Recognition

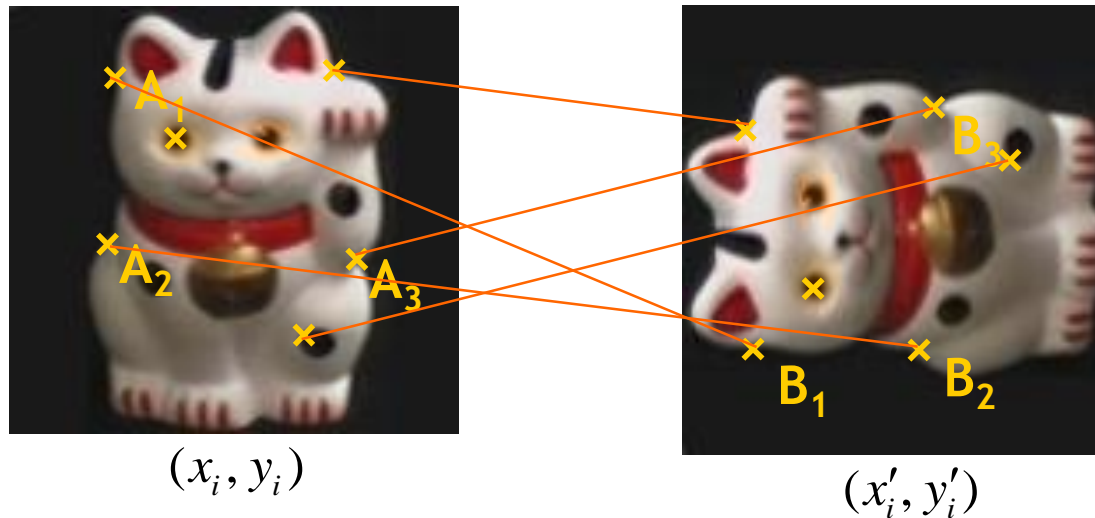
Geometric verification



[Nister & Stewenius, CVPR'06]

# Recap: Fitting an Affine Transformation

- Assuming we know the correspondences, how do we get the transformation?

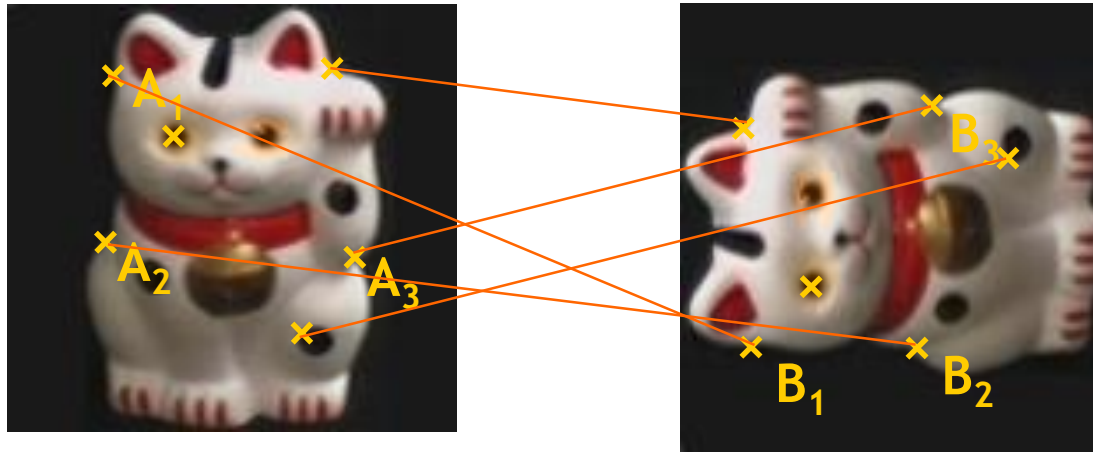


$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} \dots \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ \dots \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \dots \\ x'_i \\ y'_i \\ \dots \end{bmatrix}$$

# Recap: Fitting a Homography

- Estimating the transformation



Homogenous coordinates

Image coordinates

$$\begin{aligned} \mathbf{x}_{A_1} &\leftrightarrow \mathbf{x}_{B_1} \\ \mathbf{x}_{A_2} &\leftrightarrow \mathbf{x}_{B_2} \\ \mathbf{x}_{A_3} &\leftrightarrow \mathbf{x}_{B_3} \\ &\vdots \end{aligned}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ \frac{1}{z'} y' \\ z' \end{bmatrix}$$

Matrix notation

$$x' = Hx$$

$$x'' = \frac{1}{z'} x'$$

$$x_{A_1} = \frac{h_{11} x_{B_1} + h_{12} y_{B_1} + h_{13}}{h_{31} x_{B_1} + h_{32} y_{B_1} + 1} \quad y_{A_1} = \frac{h_{21} x_{B_1} + h_{22} y_{B_1} + h_{23}}{h_{31} x_{B_1} + h_{32} y_{B_1} + 1}$$

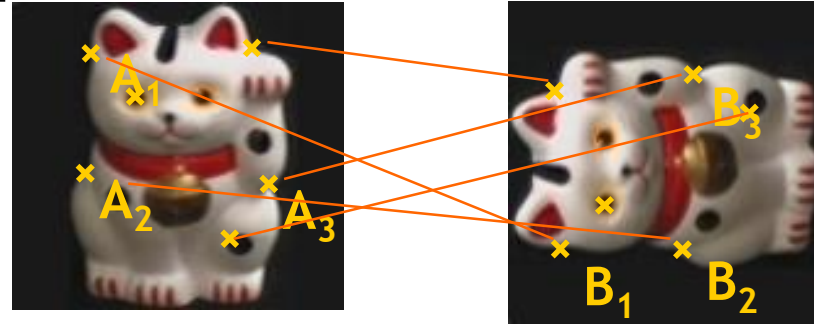
B. Leibe

# Recap: Fitting a Homography

- Estimating the transformation

$$h_{11} x_{B_1} + h_{12} y_{B_1} + h_{13} - x_{A_1} h_{31} x_{B_1} - x_{A_1} h_{32} y_{B_1} - x_{A_1} = 0$$

$$h_{21} x_{B_1} + h_{22} y_{B_1} + h_{23} - y_{A_1} h_{31} x_{B_1} - y_{A_1} h_{32} y_{B_1} - y_{A_1} = 0$$



$$\mathbf{x}_{A_1} \leftrightarrow \mathbf{x}_{B_1}$$

$$\mathbf{x}_{A_2} \leftrightarrow \mathbf{x}_{B_2}$$

$$\mathbf{x}_{A_3} \leftrightarrow \mathbf{x}_{B_3}$$

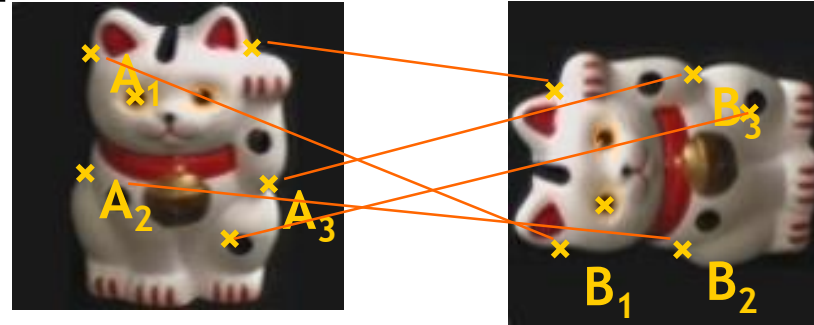
$$\vdots$$

$$\begin{bmatrix} x_{B_1} & y_{B_1} & 1 & 0 & 0 & 0 & -x_{A_1}x_{B_1} & -x_{A_1}y_{B_1} & -x_{A_1} \\ 0 & 0 & 0 & x_{B_1} & y_{B_1} & 1 & -y_{A_1}x_{B_1} & -y_{A_1}y_{B_1} & -y_{A_1} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

$$Ah = 0$$

# Recap: Fitting a Homography

- Estimating the transformation
- Solution:
  - Null-space vector of A
  - Corresponds to smallest eigenvector



$$\begin{aligned} \mathbf{x}_{A_1} &\leftrightarrow \mathbf{x}_{B_1} \\ \mathbf{x}_{A_2} &\leftrightarrow \mathbf{x}_{B_2} \\ \mathbf{x}_{A_3} &\leftrightarrow \mathbf{x}_{B_3} \\ &\vdots \end{aligned}$$

$$\begin{aligned} &\text{SVD} \\ &\downarrow \\ \mathbf{A} &= \mathbf{U}\mathbf{D}\mathbf{V}^T = \mathbf{U} \begin{bmatrix} d_{11} & \cdots & d_{19} \\ \vdots & \ddots & \vdots \\ d_{91} & \cdots & d_{99} \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{19} \\ \vdots & \ddots & \vdots \\ v_{91} & \cdots & v_{99} \end{bmatrix}^T \end{aligned}$$

$$Ah = 0$$

$$\mathbf{h} = \frac{[v_{19}, \dots, v_{99}]}{v_{99}}$$

Minimizes least square error

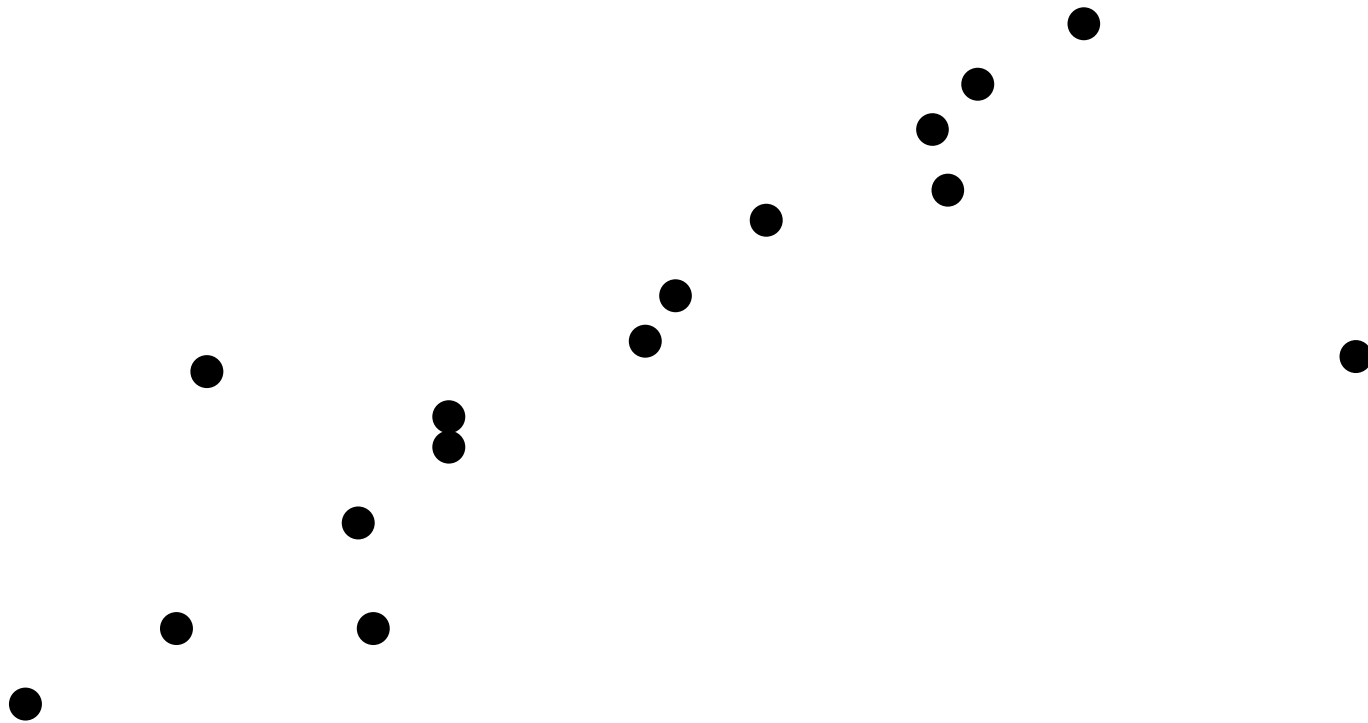
# Recap: RANSAC

## RANSAC loop:

1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
  2. Compute transformation from seed group
  3. Find *inliers* to this transformation
  4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers
- Keep the transformation with the largest number of inliers

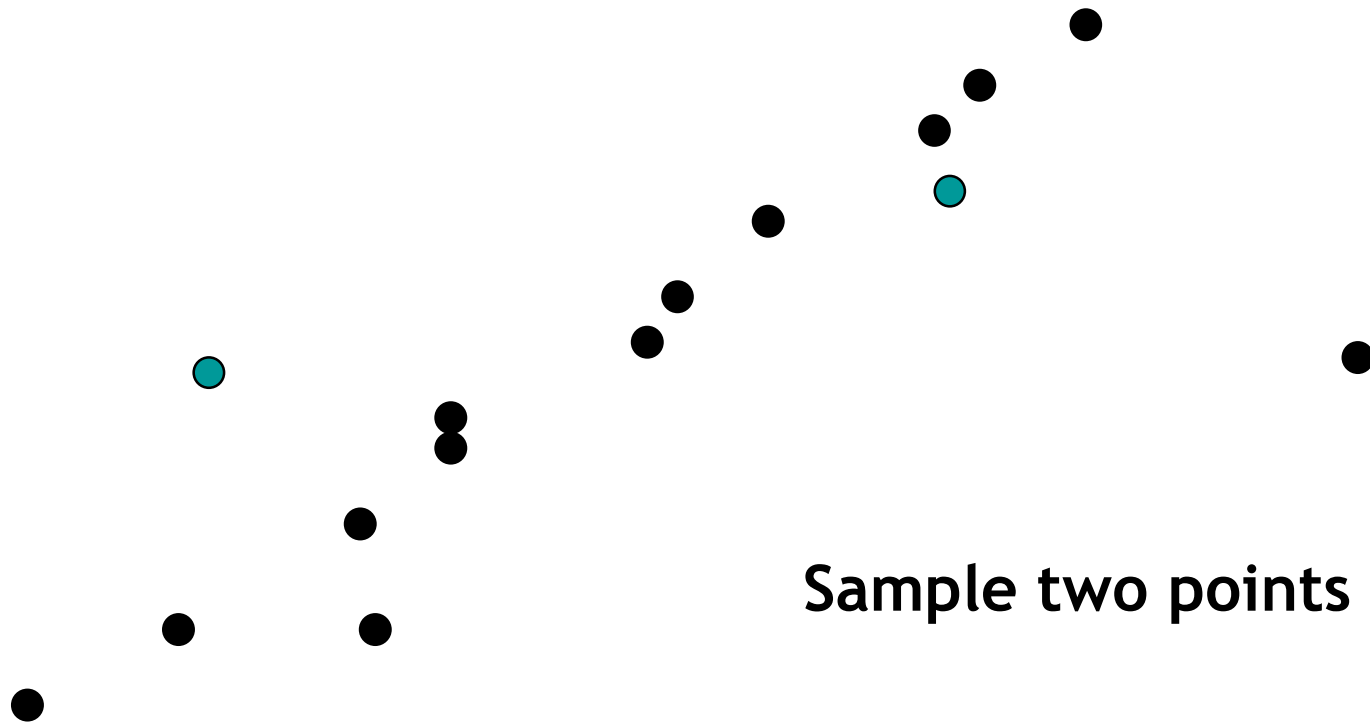
# Recap: RANSAC Line Fitting Example

- Task: Estimate the best line



# Recap: RANSAC Line Fitting Example

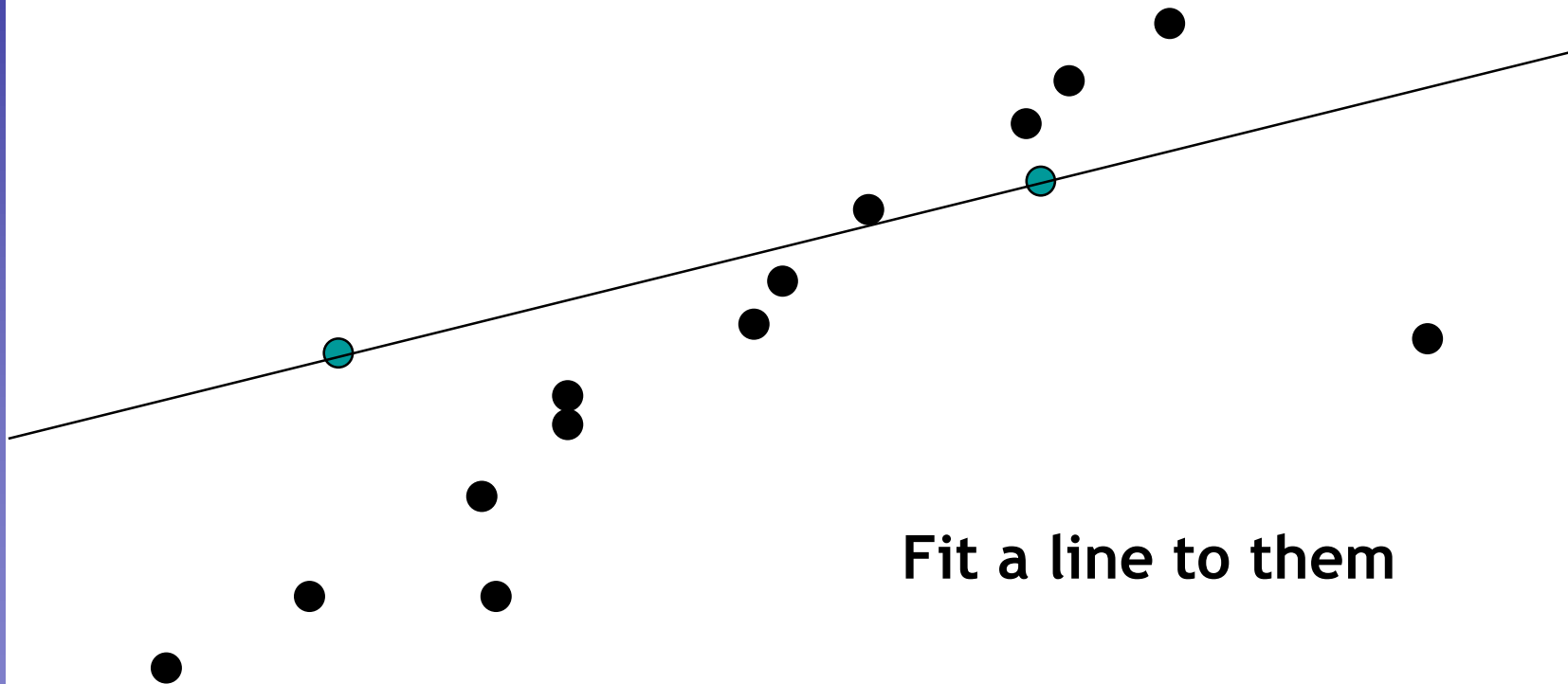
- Task: Estimate the best line





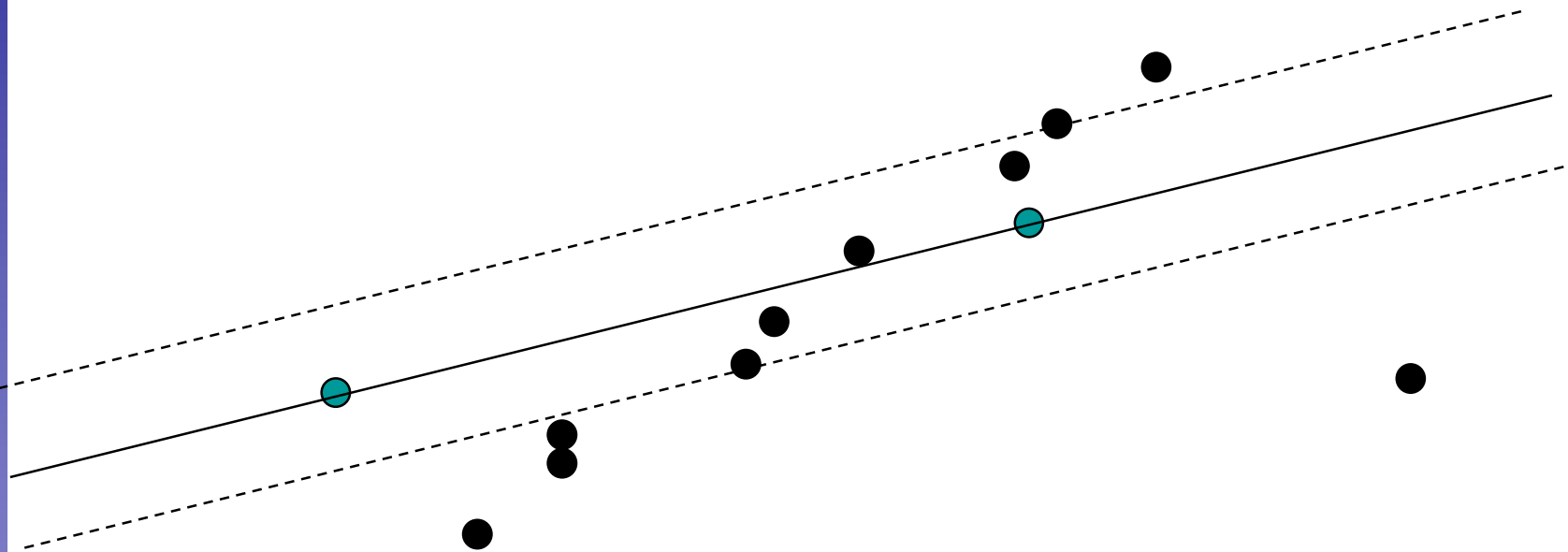
# Recap: RANSAC Line Fitting Example

- Task: Estimate the best line



# Recap: RANSAC Line Fitting Example

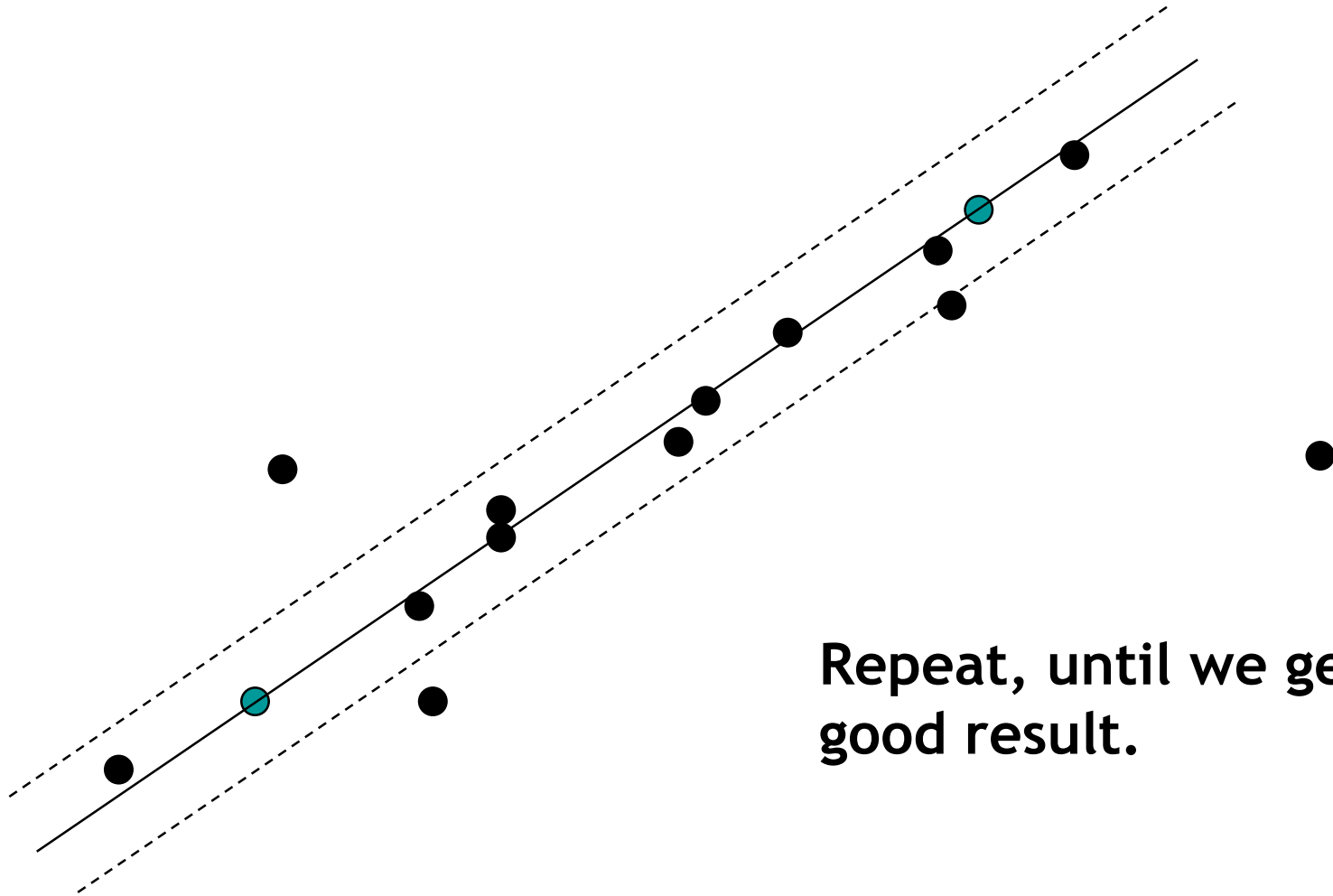
- Task: Estimate the best line



Total number of points  
within a threshold of  
line.

# Recap: RANSAC Line Fitting Example

- Task: Estimate the best line



Repeat, until we get a good result.

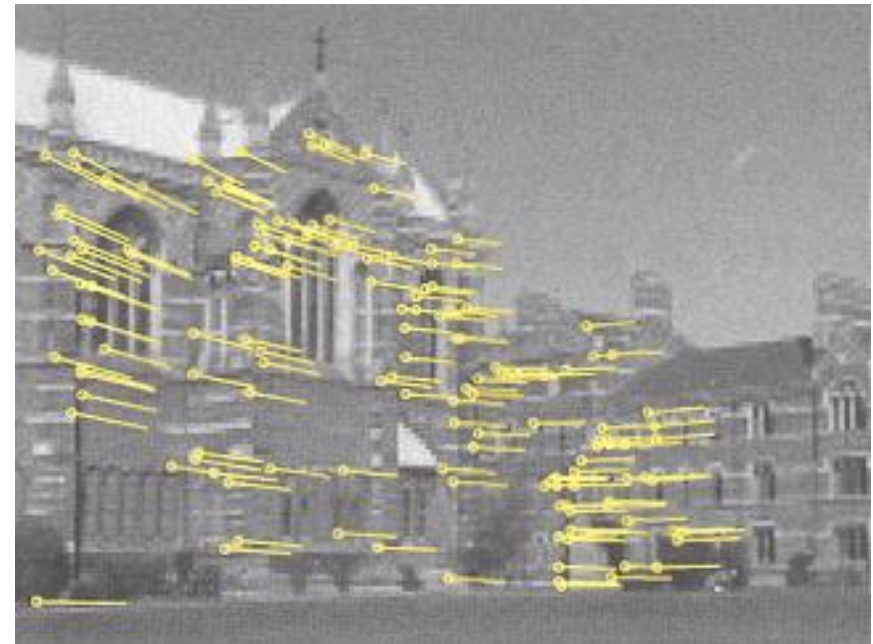
# Recap: Feature Matching Example

- Find best stereo match within a square search window (here  $300 \text{ pixels}^2$ )
- Global transformation model: epipolar geometry

before RANSAC



after RANSAC

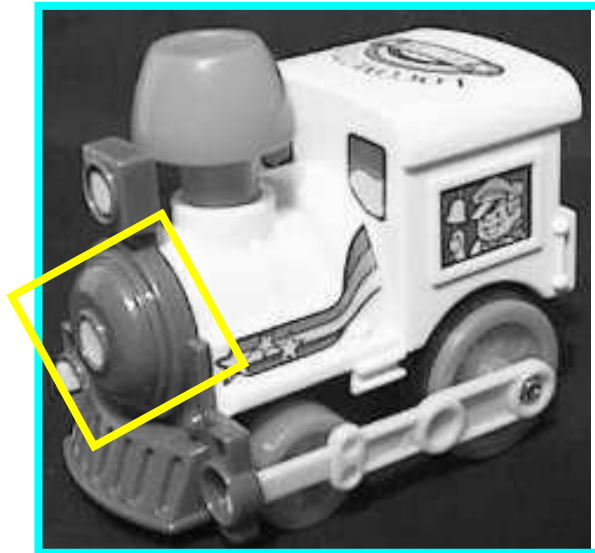


Images from Hartley & Zisserman

# Recap: Generalized Hough Transform

- Suppose our features are scale- and rotation-invariant
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation).

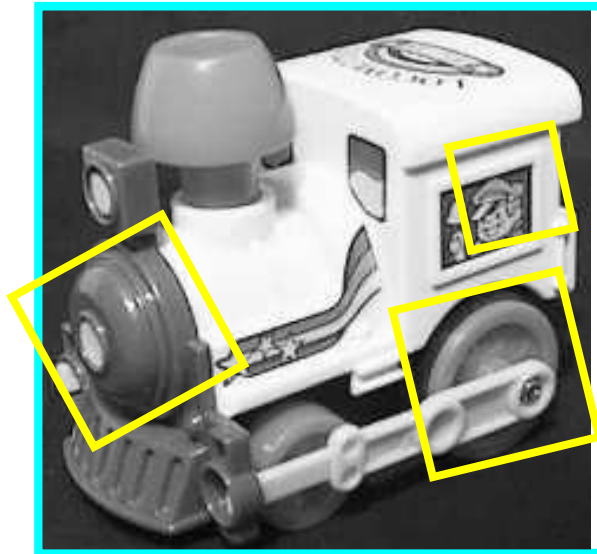
model



# Recap: Generalized Hough Transform

- Suppose our features are scale- and rotation-invariant
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation).
  - Of course, a hypothesis from a single match is unreliable.
  - Solution: let each match vote for its hypothesis in a Hough space with very coarse bins.

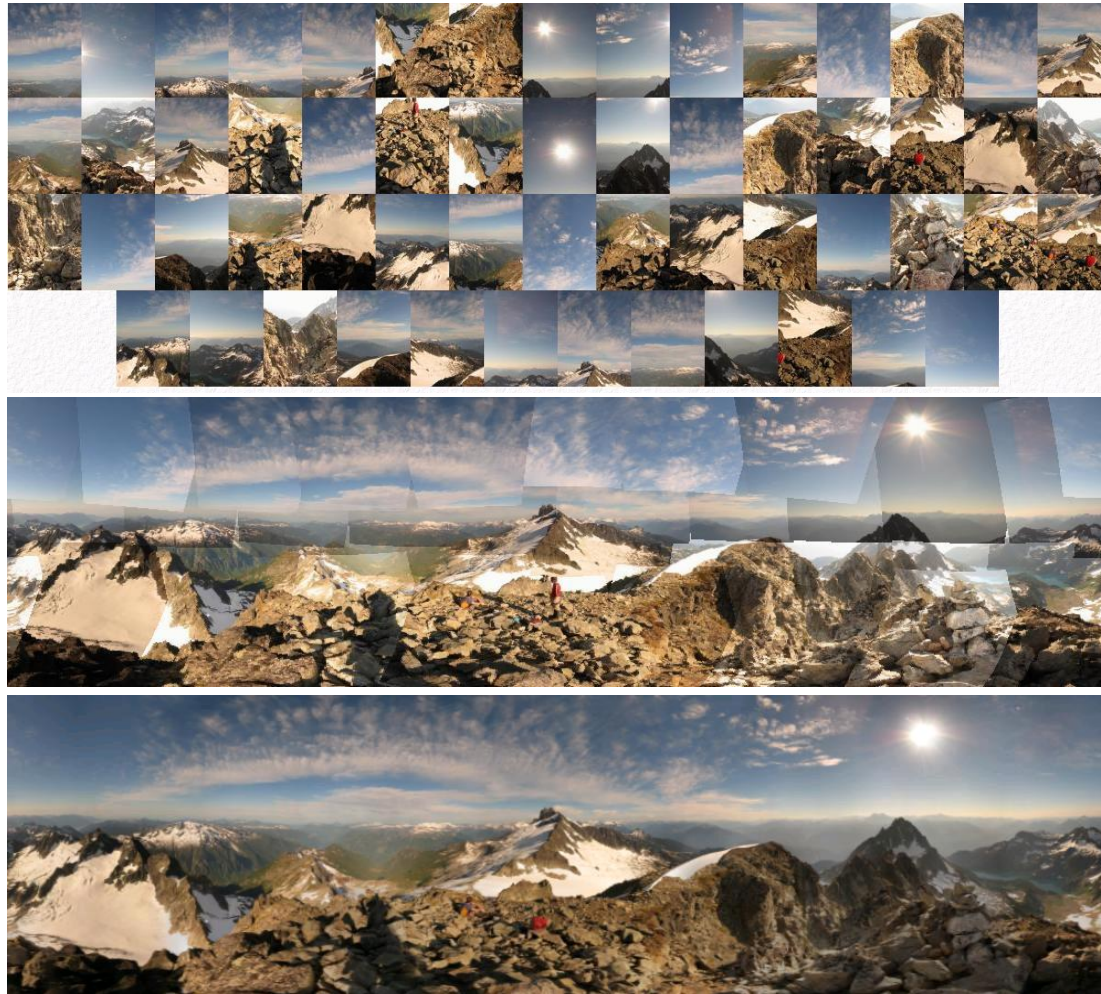
model





# Application: Panorama Stitching

see  
Panorama Demo!

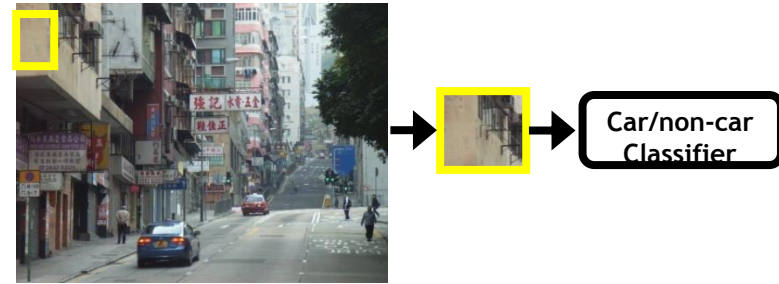


<http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html>

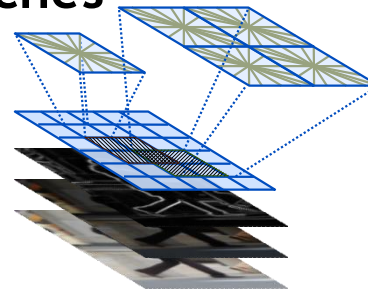
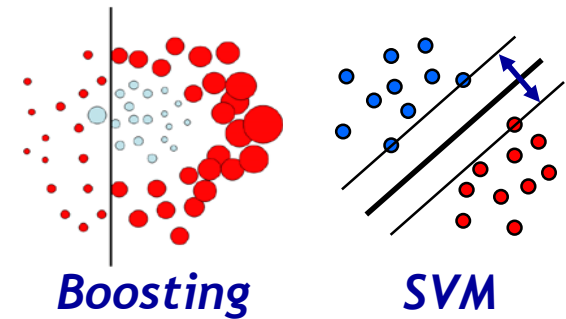


# Repetition

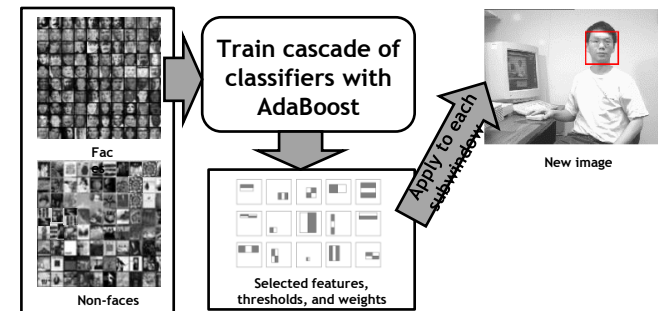
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
  - Sliding Window based Object Detection
  - Bag-of-Words Approaches
  - Deep Learning Approaches
- 3D Reconstruction
- Motion and Tracking



*Sliding window principle*



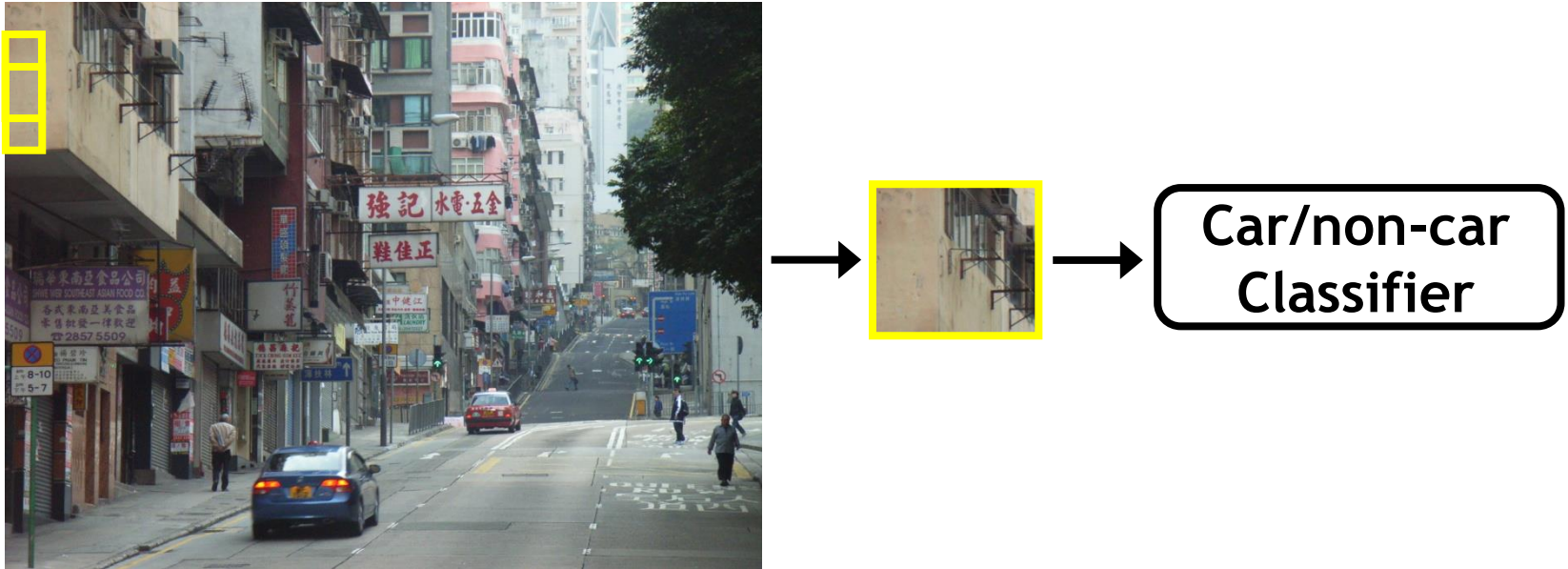
*HOG detector*



*Viola-Jones face detector*

# Recap: Sliding-Window Object Detection

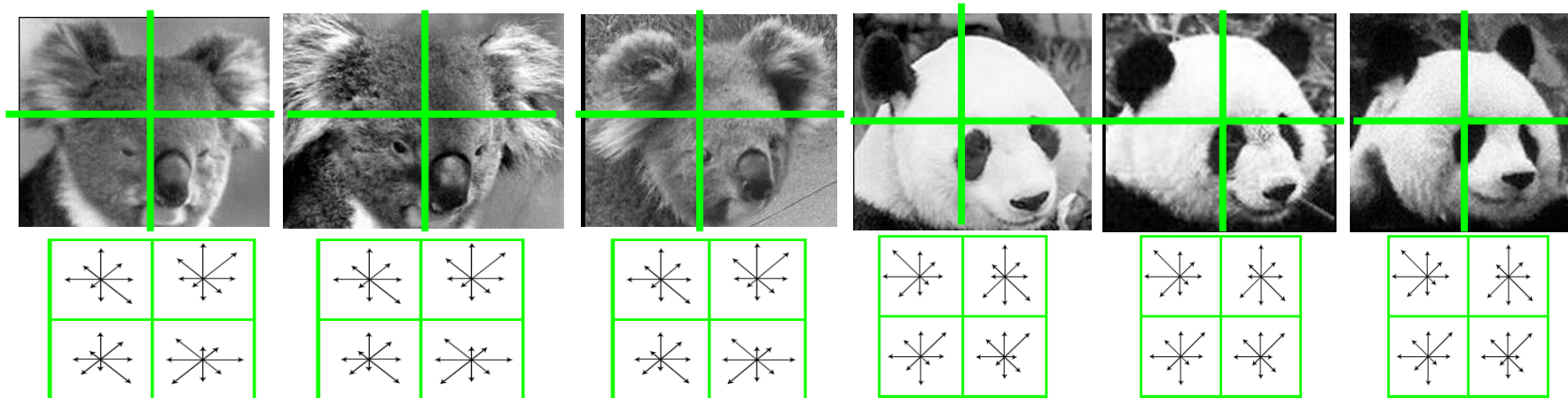
- If object may be in a cluttered scene, slide a window around looking for it.



- Essentially, this is a brute-force approach with many local decisions.

# Recap: Gradient-based Representations

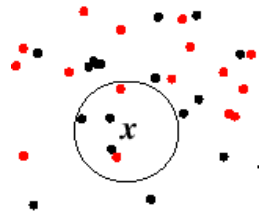
- Consider edges, contours, and (oriented) intensity gradients



- Summarize local distribution of gradients with histogram
  - Locally orderless: offers invariance to small shifts and rotations
  - Contrast-normalization: try to correct for variable illumination

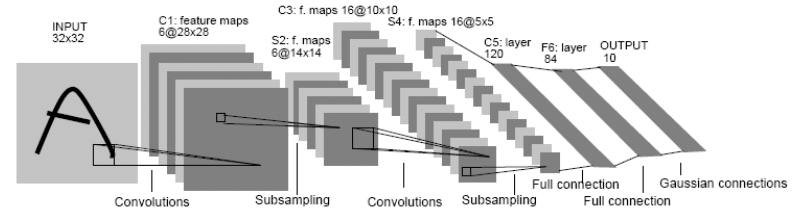
# Recap: Classifier Construction: Many Choices...

## Nearest Neighbor



Berg, Berg, Malik 2005,  
Chum, Zisserman 2007,  
Boiman, Shechtman, Irani 2008, ...

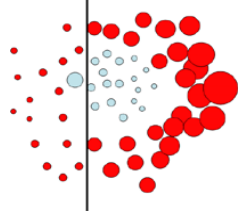
## Neural networks



LeCun, Bottou, Bengio, Haffner 1998  
Rowley, Baluja, Kanade 1998

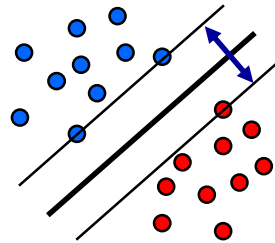
...

## Boosting



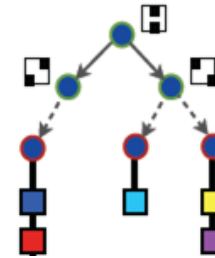
Viola, Jones 2001,  
Torralba et al. 2004,  
Opelt et al. 2006,  
Benenson 2012, ...

## Support Vector Machines



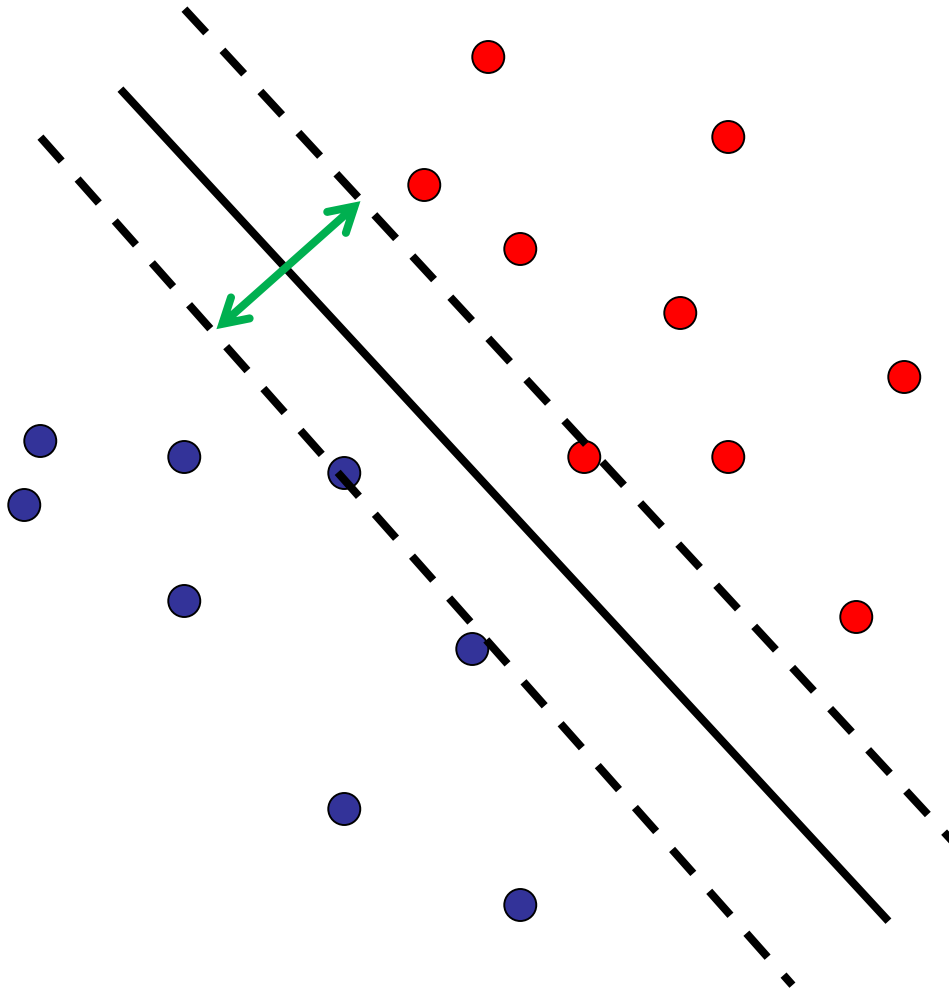
Vapnik, Schölkopf 1995,  
Papageorgiou, Poggio '01,  
Dalal, Triggs 2005,  
Vedaldi, Zisserman 2012

## Randomized Forests



Amit, Geman 1997,  
Breiman 2001,  
Lepetit, Fua 2006,  
Gall, Lempitsky 2009,...

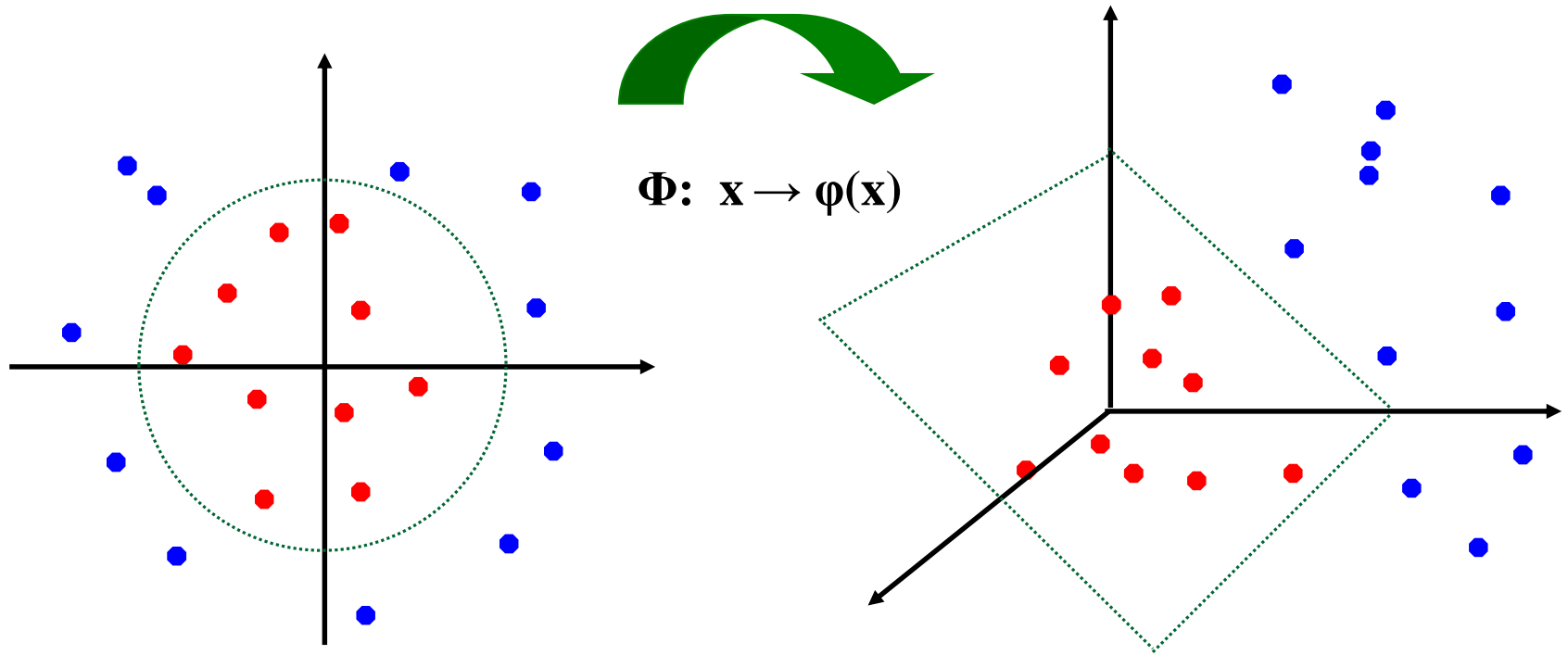
# Recap: Support Vector Machines (SVMs)



- Discriminative classifier based on *optimal separating hyperplane* (i.e. line for 2D case)
- Maximize the *margin* between the positive and negative training examples

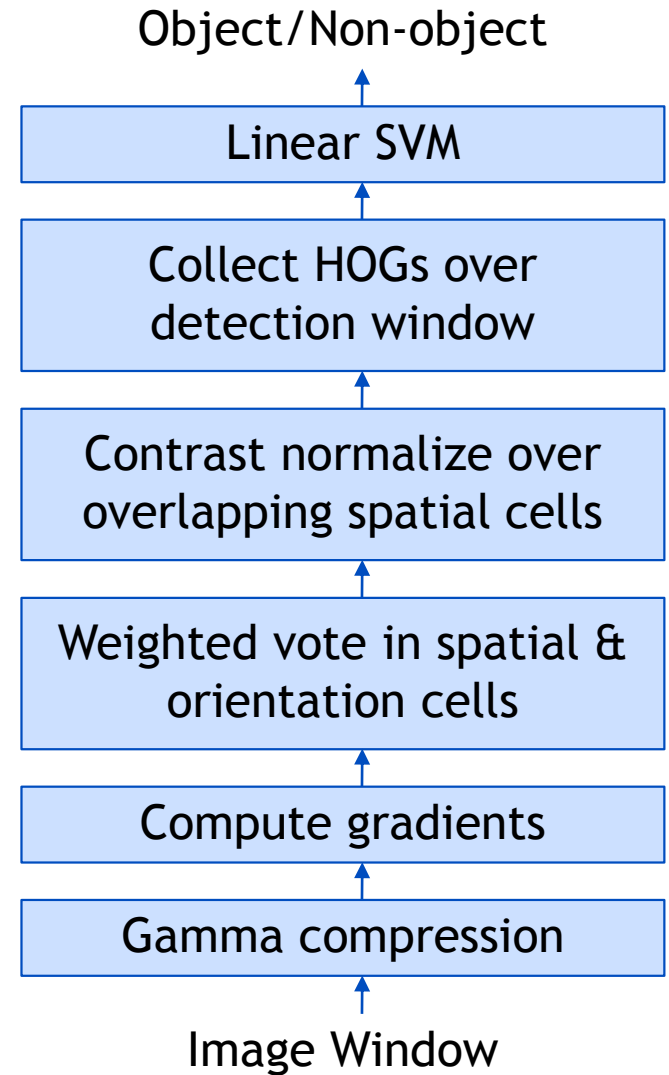
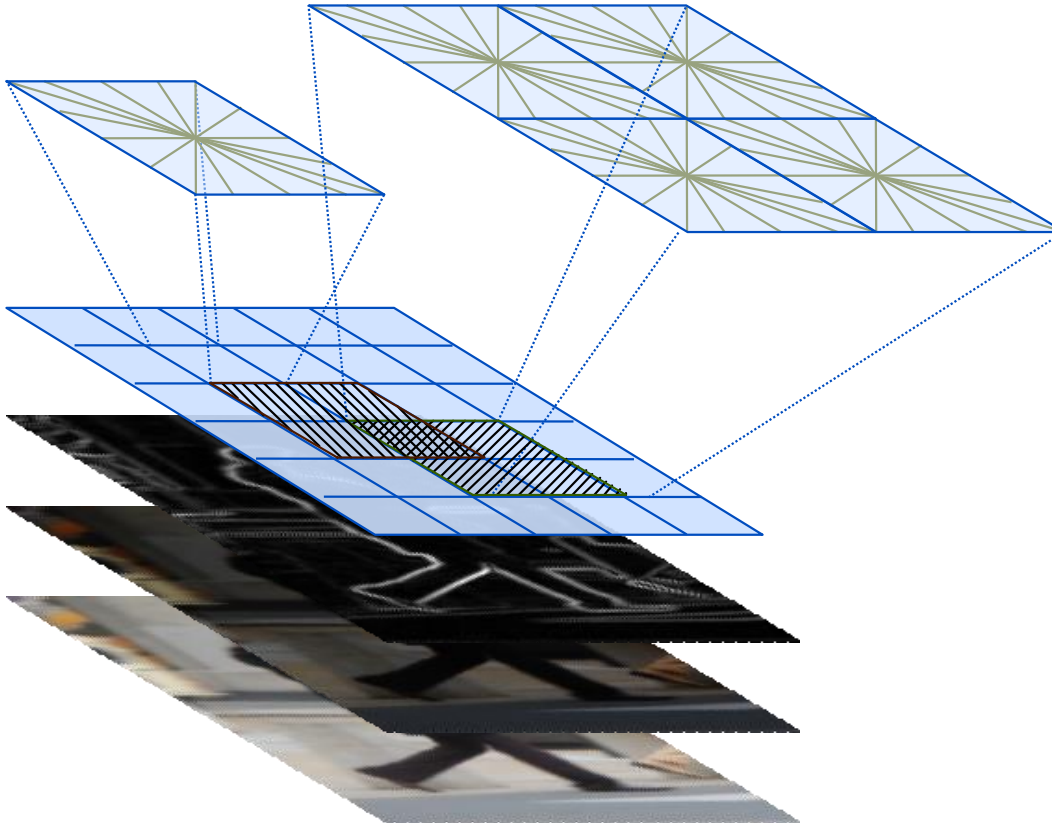
# Recap: Non-Linear SVMs

- General idea: The original input space can be mapped to some higher-dimensional feature space where the training set is separable:



# Recap: HOG Descriptor Processing Chain

- **SVM Classification**
  - Typically using a linear SVM





# Recap: HOG Cell Computation Details

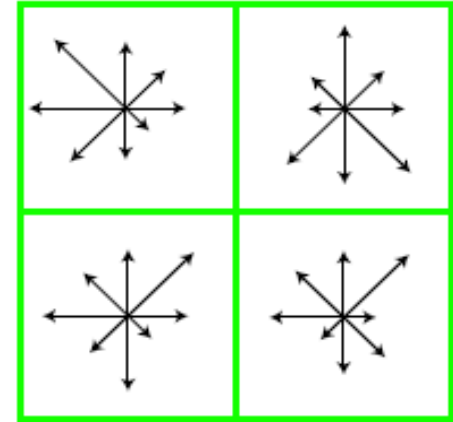
- Gradient orientation voting

- Each pixel contributes to localized gradient orientation histogram(s)
- Vote is weighted by the pixel's gradient magnitude



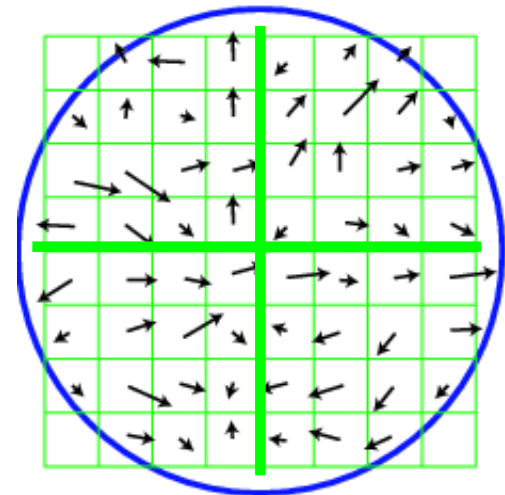
$$\theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$



- Block-level Gaussian weighting

- An additional Gaussian weight is applied to each 2x2 block of cells
- Each cell is part of 4 such blocks, resulting in 4 versions of the histogram.



# Recap: HOG Cell Computation Details (2)

- Important for robustness: **Tri-linear interpolation**

- Each pixel contributes to (up to) 4 neighboring cell histograms
- Weights are obtained by **bilinear interpolation in image space**:

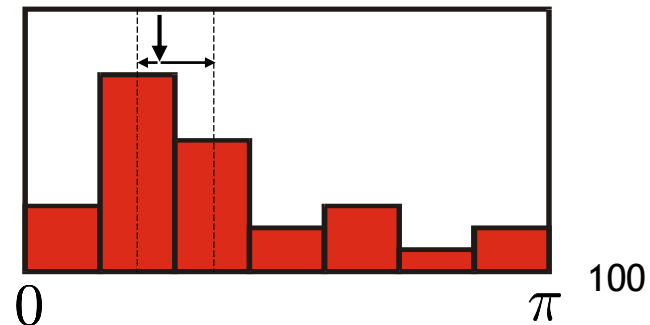
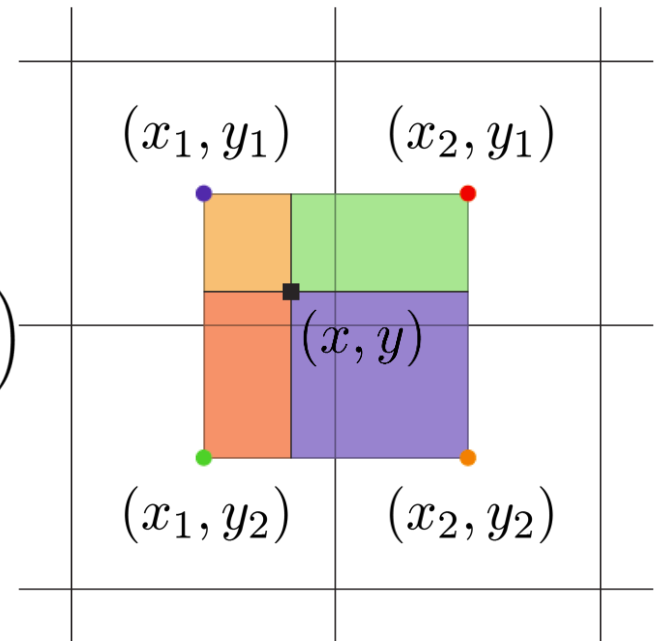
$$h(x_1, y_1) \leftarrow w \cdot \left(1 - \frac{x - x_1}{x_2 - x_1}\right) \left(1 - \frac{y - y_1}{y_2 - y_1}\right)$$

$$h(x_1, y_2) \leftarrow w \cdot \left(1 - \frac{x - x_1}{x_2 - x_1}\right) \left(\frac{y - y_1}{y_2 - y_1}\right)$$

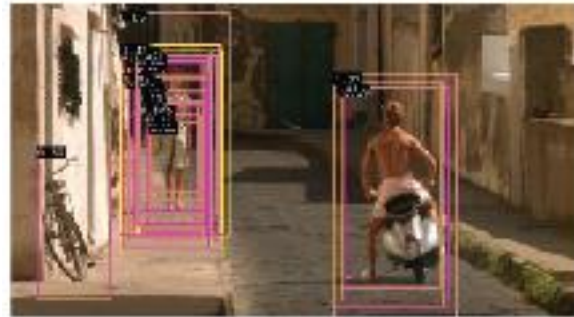
$$h(x_2, y_1) \leftarrow w \cdot \left(\frac{x - x_1}{x_2 - x_1}\right) \left(1 - \frac{y - y_1}{y_2 - y_1}\right)$$

$$h(x_2, y_2) \leftarrow w \cdot \left(\frac{x - x_1}{x_2 - x_1}\right) \left(\frac{y - y_1}{y_2 - y_1}\right)$$

- Contribution is further split over (up to) 2 neighboring orientation bins via **linear interpolation over angles**.



# Recap: Non-Maximum Suppression



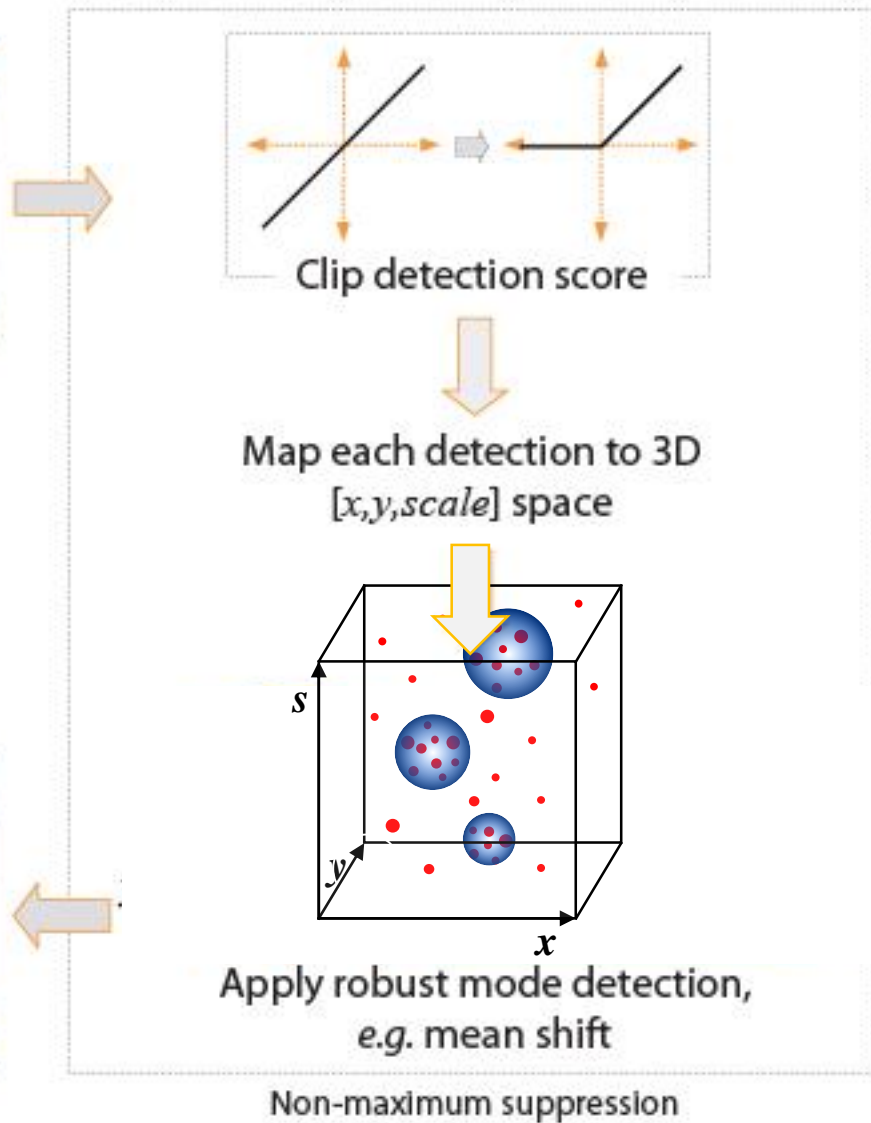
After multi-scale dense scan



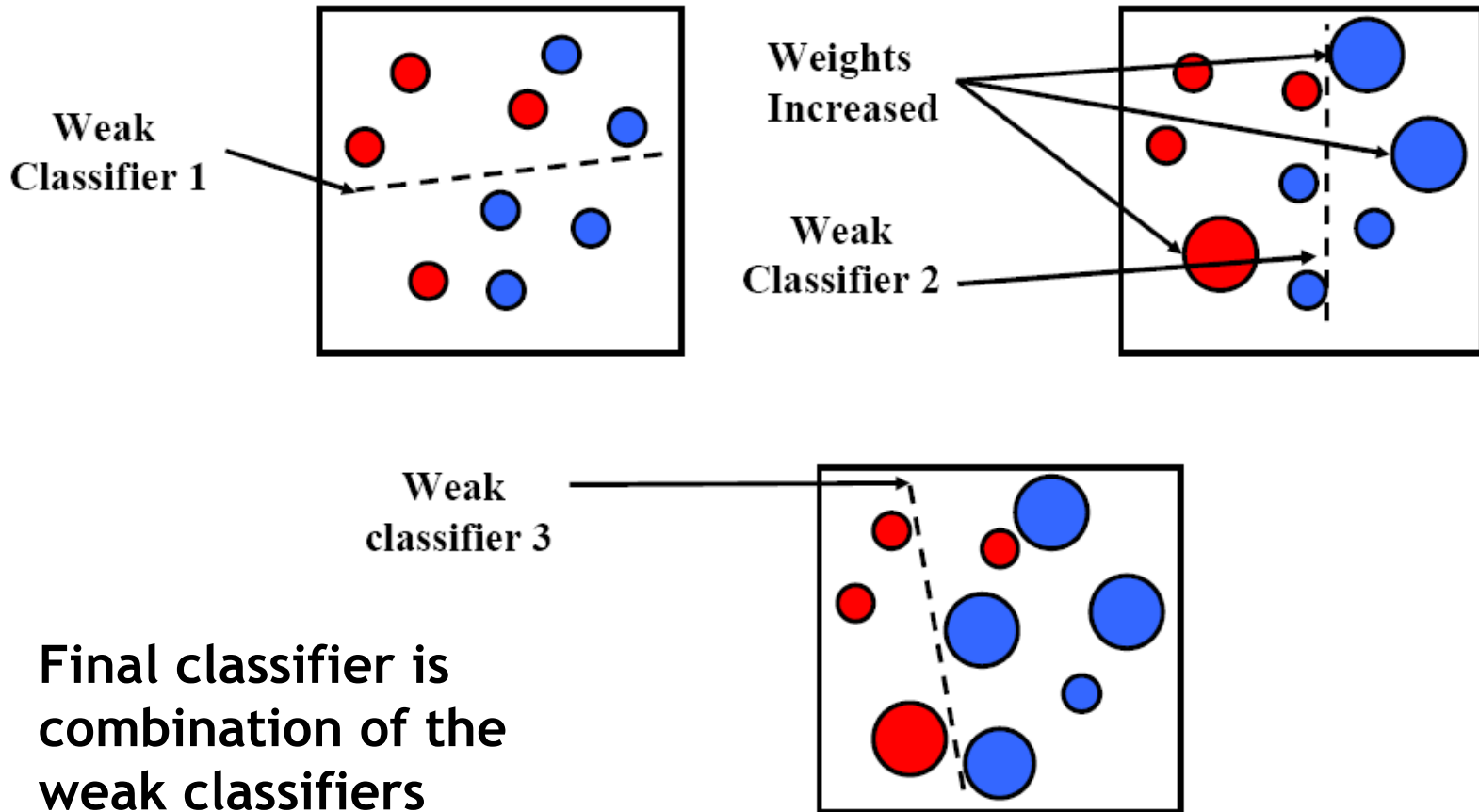
Goal



Fusion of multiple detections



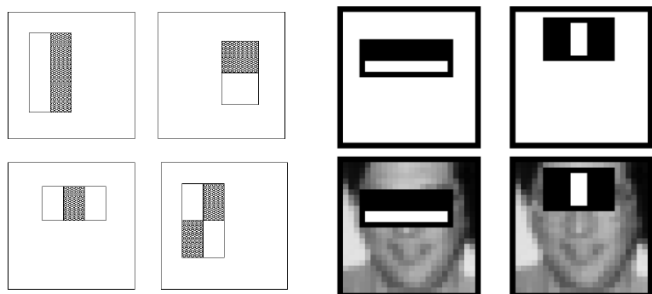
# Recap: AdaBoost



Final classifier is combination of the weak classifiers

# Recap: Viola-Jones Face Detection

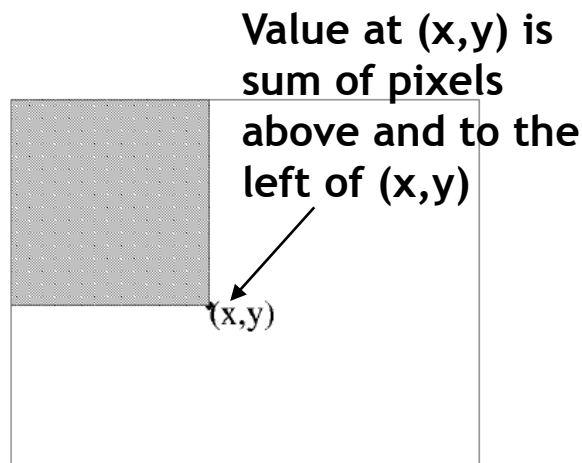
“Rectangular” filters



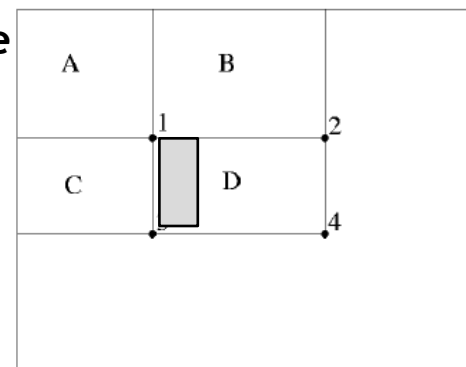
Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images → scale features directly for same cost



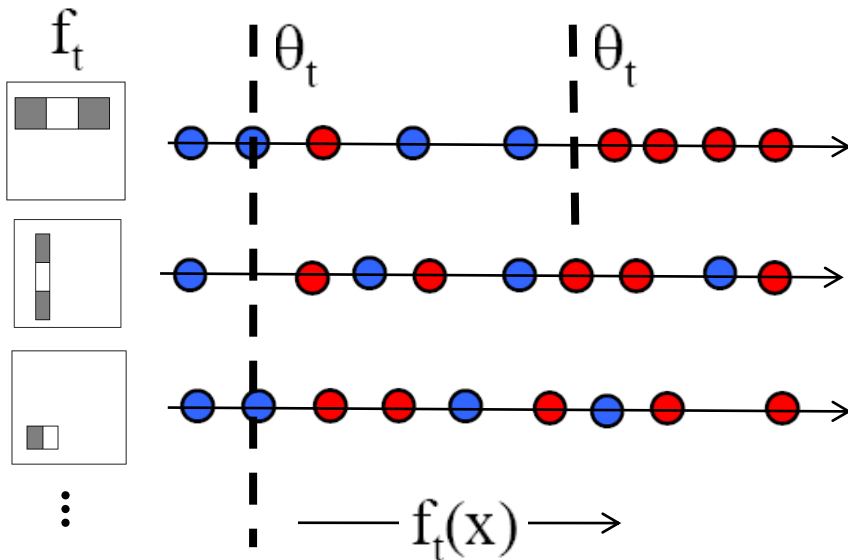
Integral image



$$\begin{aligned}
 D &= 1 + 4 - (2 + 3) \\
 &= A + (A + B + C + D) - (A + C + A + B) \\
 &= D
 \end{aligned}$$

# Recap: AdaBoost Feature+Classifier Selection

- Want to select the single rectangle feature and threshold that best separates **positive** (faces) and **negative** (non-faces) training examples, in terms of *weighted* error.



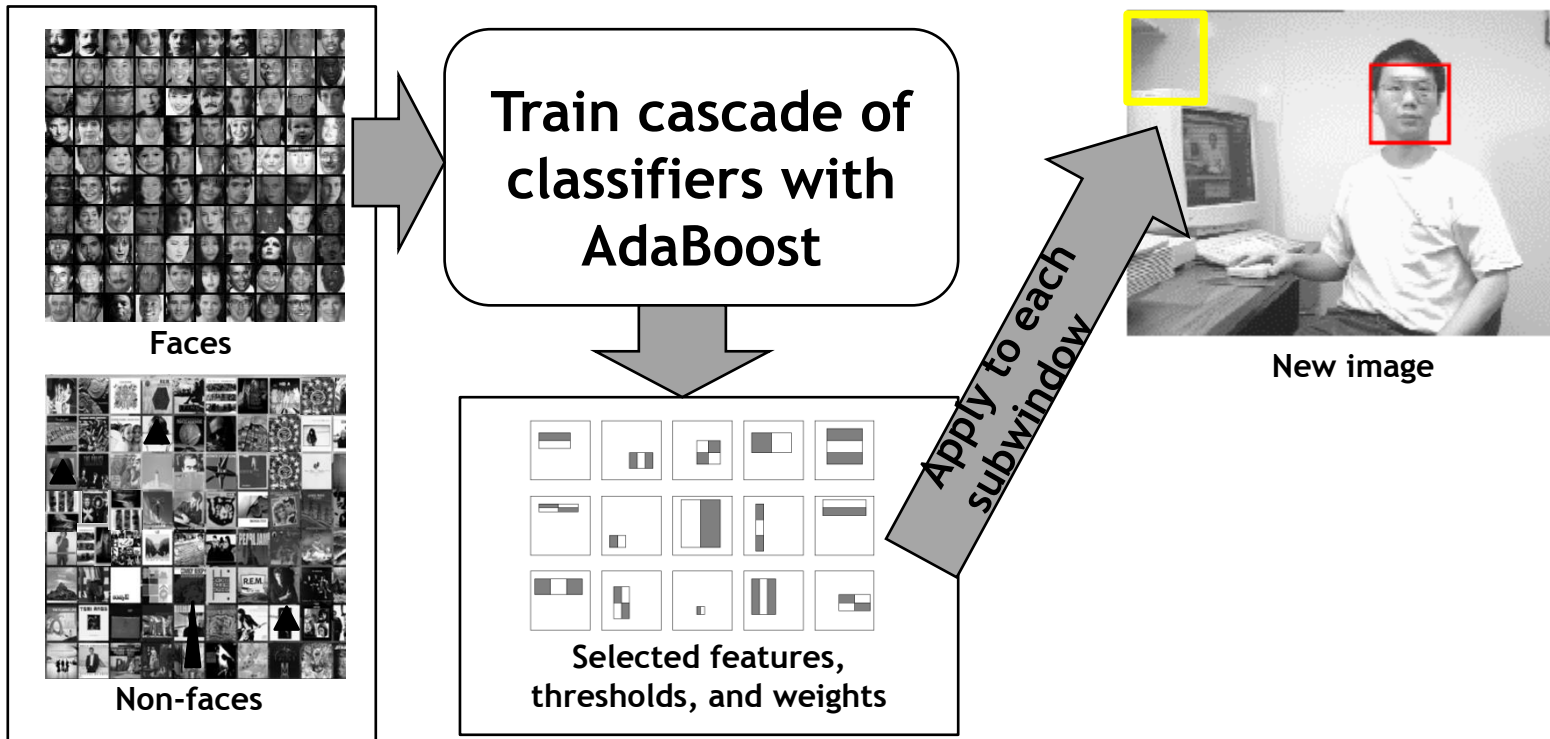
Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

For next round, reweight the examples according to errors, choose another filter/threshold combo.

# Application: Viola-Jones Face Detector

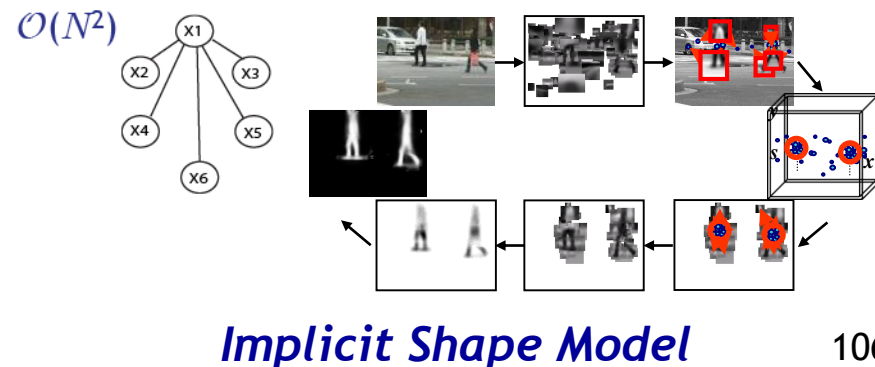
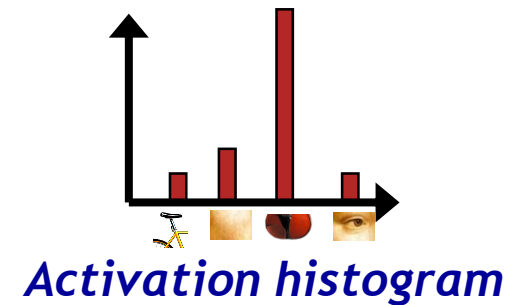
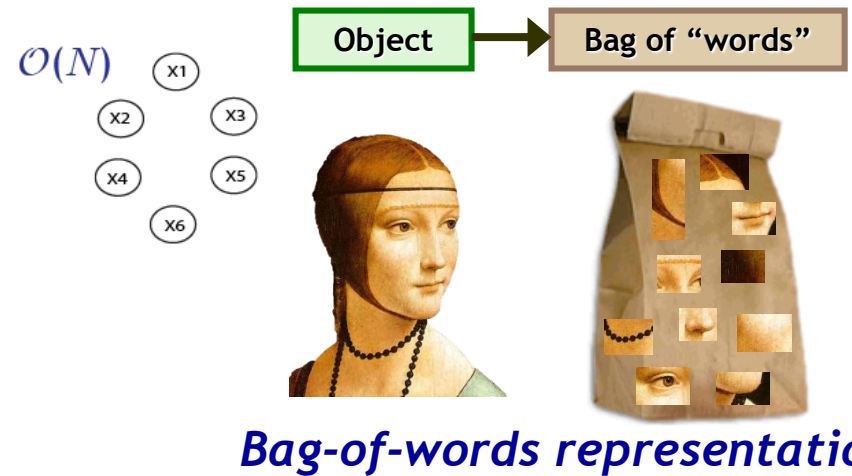


- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV:  
<http://sourceforge.net/projects/opencvlibrary/>]



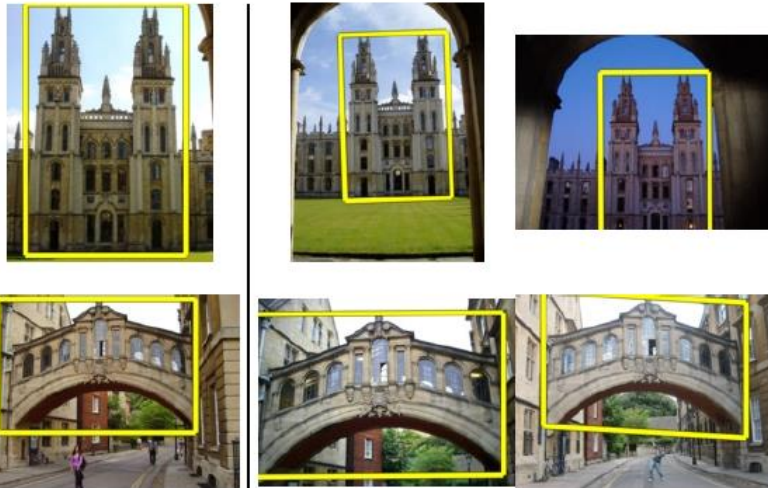
# Repetition

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
  - Sliding Window based Object Detection
  - Part-based Approaches
  - Deep Learning Approaches
- 3D Reconstruction
- Motion and Tracking



# Recap: Identification vs. Categorization

- Find *this particular* object
- Recognize ANY car

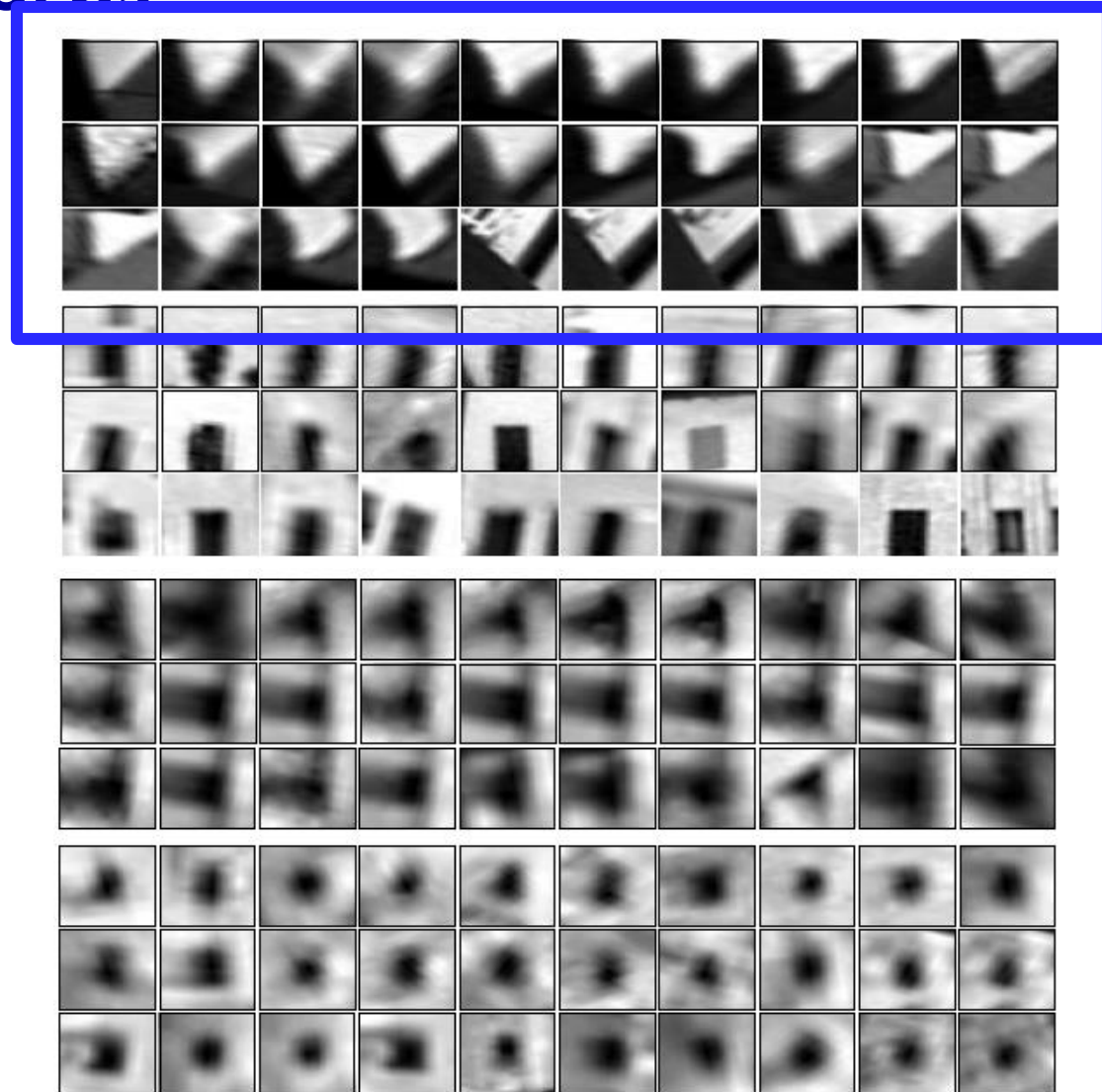
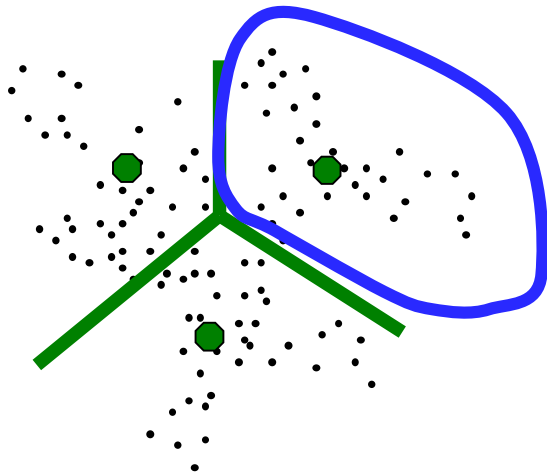


- Recognize ANY cow



# Recap: Visual Words

- Quantize the feature space into “visual words”
- Perform matching only to those visual words.



Exact feature matching → Match to same visual word

# Recap: Bag-of-Words Representations (BoW)

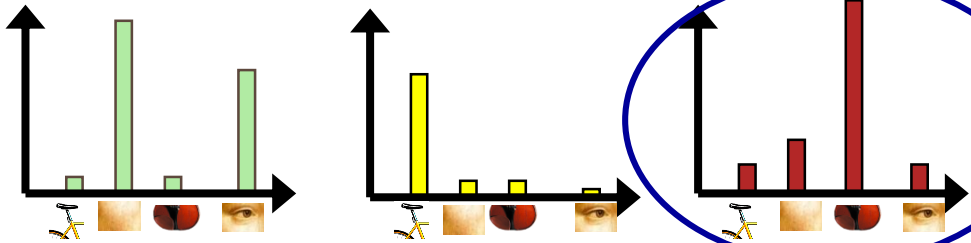




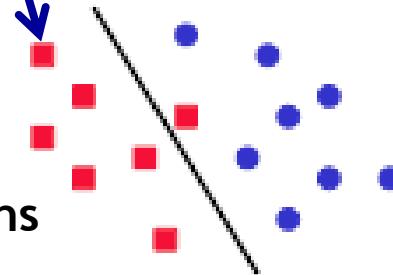
# Recap: Categorization with Bags-of-Words



- Compute the word activation histogram for each image.
- Let each such BoW histogram be a feature vector.
- Use images from each class to train a classifier (e.g., an SVM).

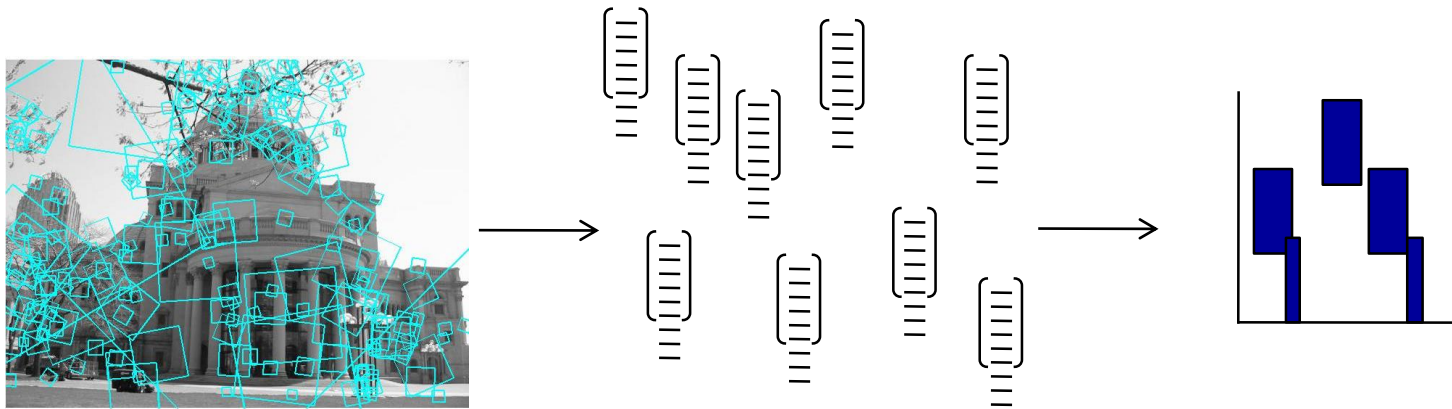


Violins



# Recap: Advantage of BoW Histograms

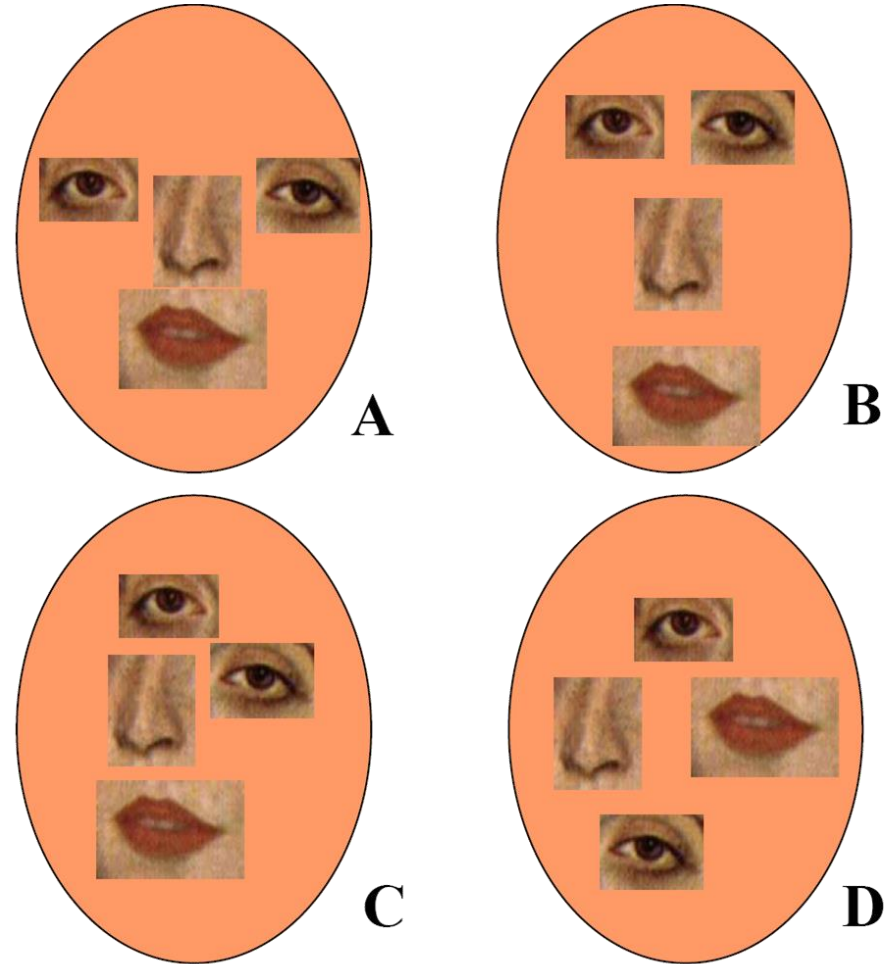
- Bag of words representations make it possible to describe the unordered point set with a single vector (of fixed dimension across image examples).



- Provides easy way to use distribution of feature types with various learning algorithms requiring vector input.

# Limitations of BoW Representations

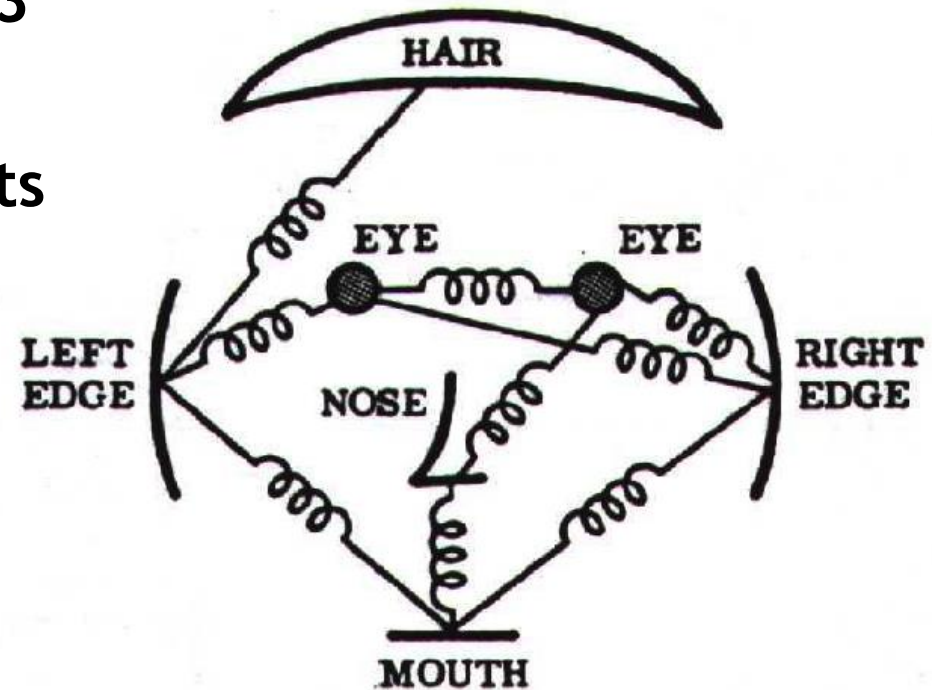
- The bag of words removes spatial layout.
- This is both a strength and a weakness.
- *Why a strength?*
- *Why a weakness?*



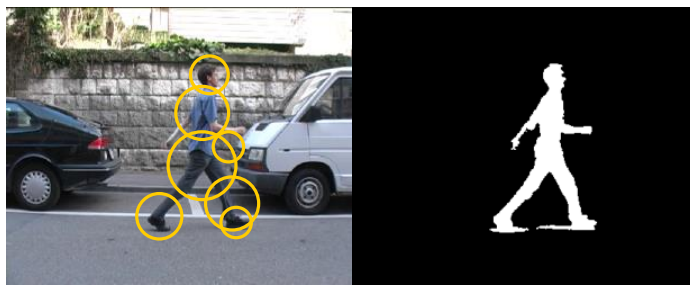


# Recap: Part-Based Models

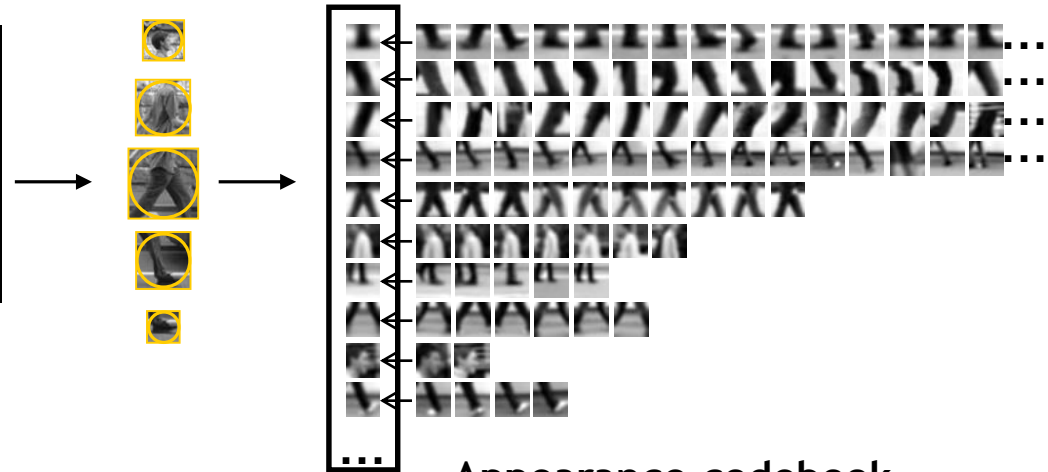
- Fischler & Elschlager 1973
- Model has two components
  - parts  
(2D image fragments)
  - structure  
(configuration of parts)



# Recap: Implicit Shape Model - Representation

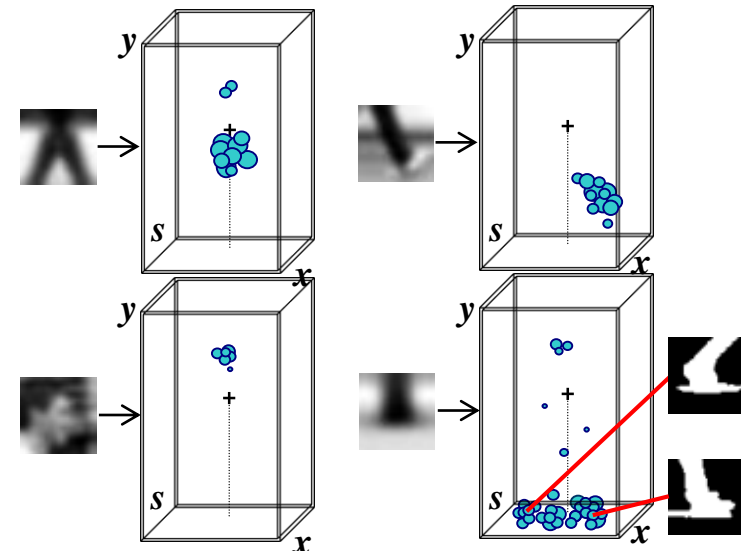


Training images  
(+reference segmentation)



Appearance codebook

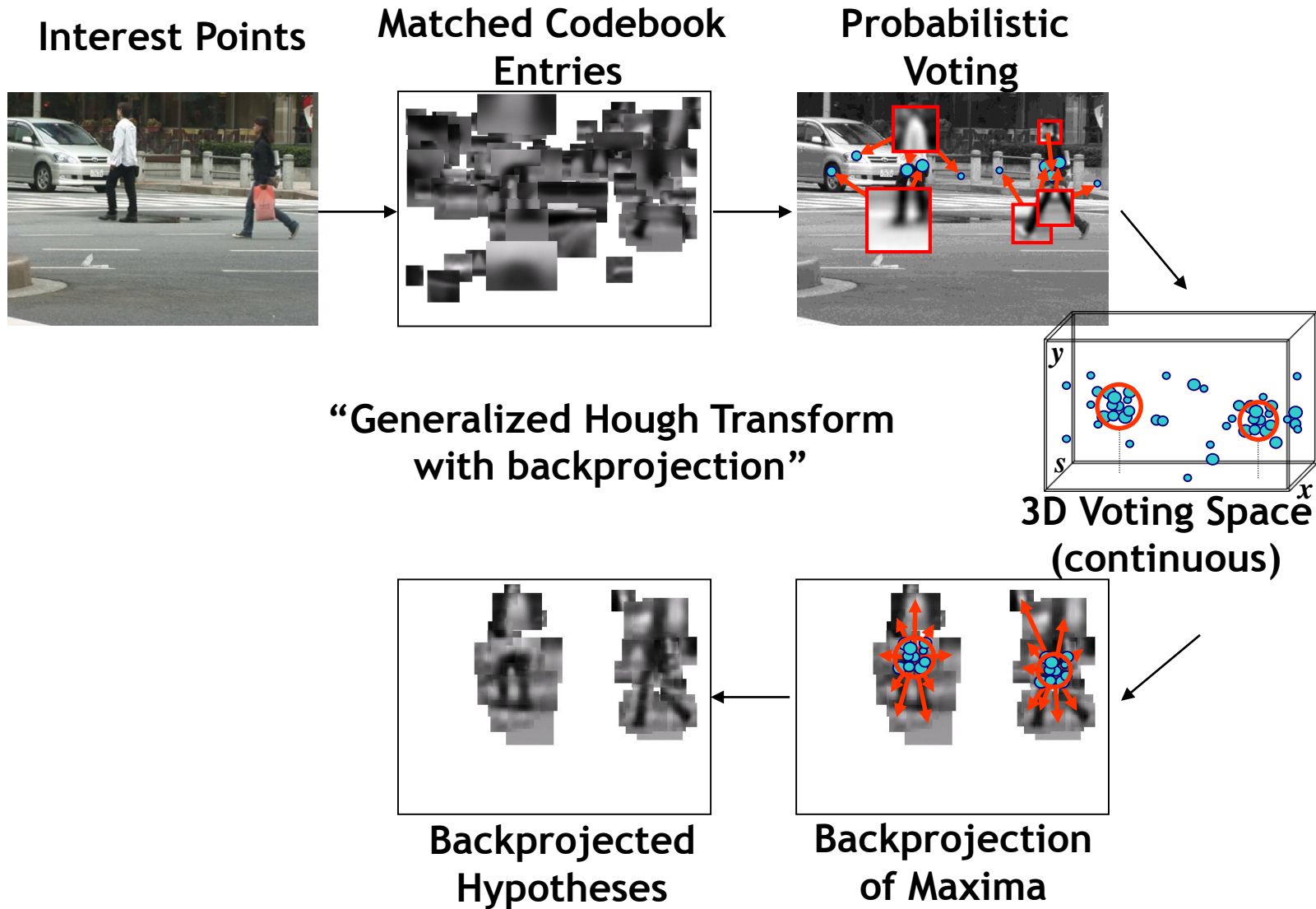
- Learn appearance codebook
  - Extract local features at interest points
  - Clustering  $\Rightarrow$  appearance codebook
- Learn spatial distributions
  - Match codebook to training images
  - Record matching positions on object



Spatial occurrence distributions

+ local figure-ground labels 114

# Recap: Implicit Shape Model - Recognition



# Recap: Scale Invariant Voting

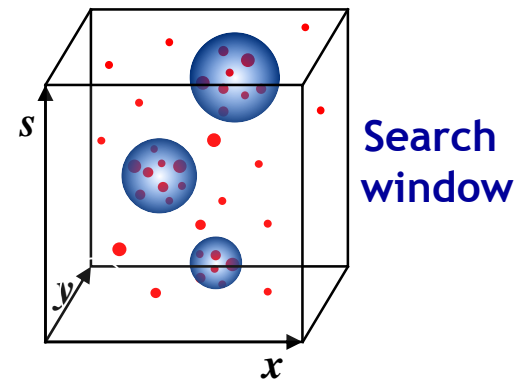
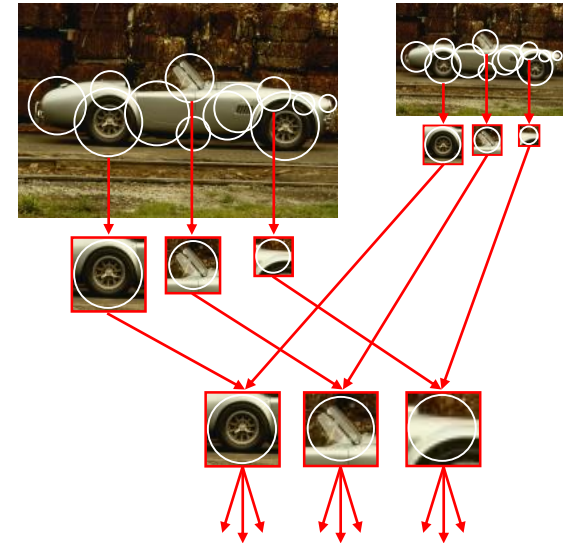
- Scale-invariant feature selection
  - Scale-invariant interest points
  - Rescale extracted patches
  - Match to constant-size codebook
- Generate scale votes
  - Scale as 3<sup>rd</sup> dimension in voting space

$$x_{vote} = x_{img} - x_{occ}(s_{img}/s_{occ})$$

$$y_{vote} = y_{img} - y_{occ}(s_{img}/s_{occ})$$

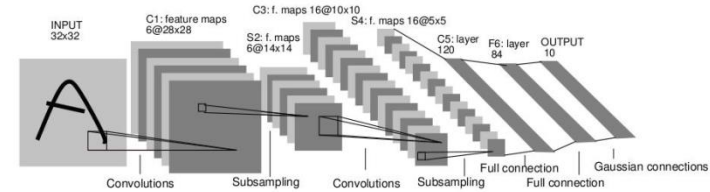
$$s_{vote} = (s_{img}/s_{occ}).$$

- Search for maxima in 3D voting space

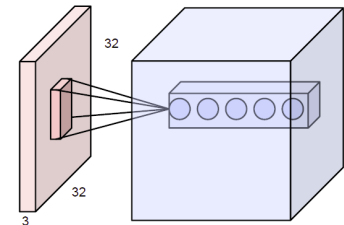


# Repetition

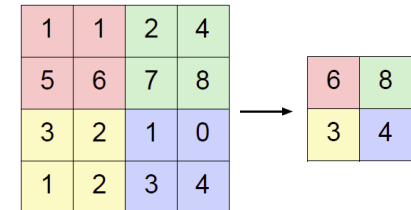
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
  - Sliding Window based Object Detection
  - Part-based Approaches
  - Deep Learning Approaches
- 3D Reconstruction
- Motion and Tracking



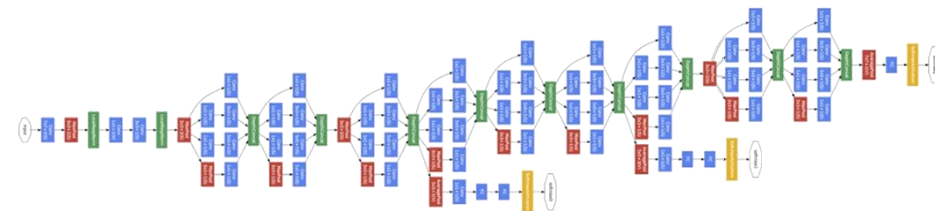
*Convolutional Neural Networks*



*Convolution layers*

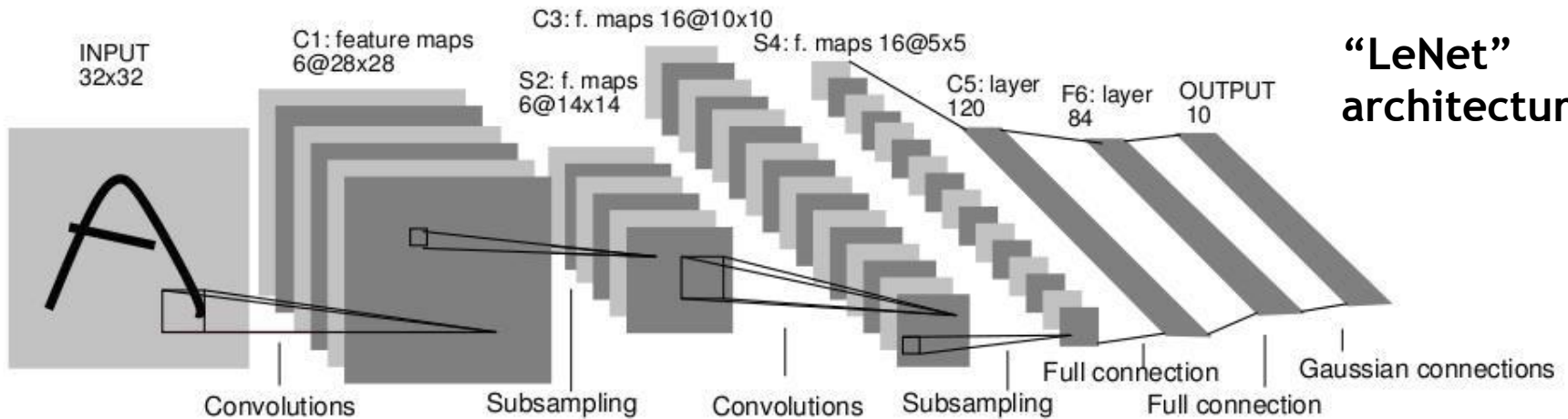


*Pooling layers*



*AlexNet, VGGNet, GoogLeNet, ResNet*

# Recap: Convolutional Neural Networks

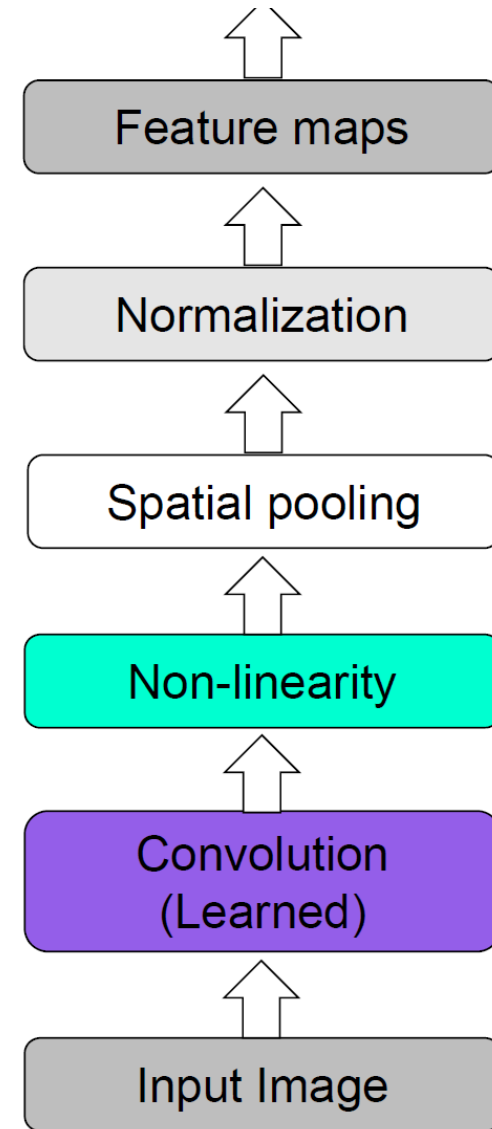


- **Neural network with specialized connectivity structure**
  - Stack multiple stages of feature extractors
  - Higher stages compute more global, more invariant features
  - Classification layer at the end

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, [Gradient-based learning applied to document recognition](#), Proceedings of the IEEE 86(11): 2278-2324, 1998.

# Recap: CNN Structure

- **Feed-forward feature extraction**
  1. Convolve input with learned filters
  2. Non-linearity
  3. Spatial pooling
  4. (Normalization)
- **Supervised training of convolutional filters by back-propagating classification error**





# Recap: Intuition of CNNs

- Convolutional net

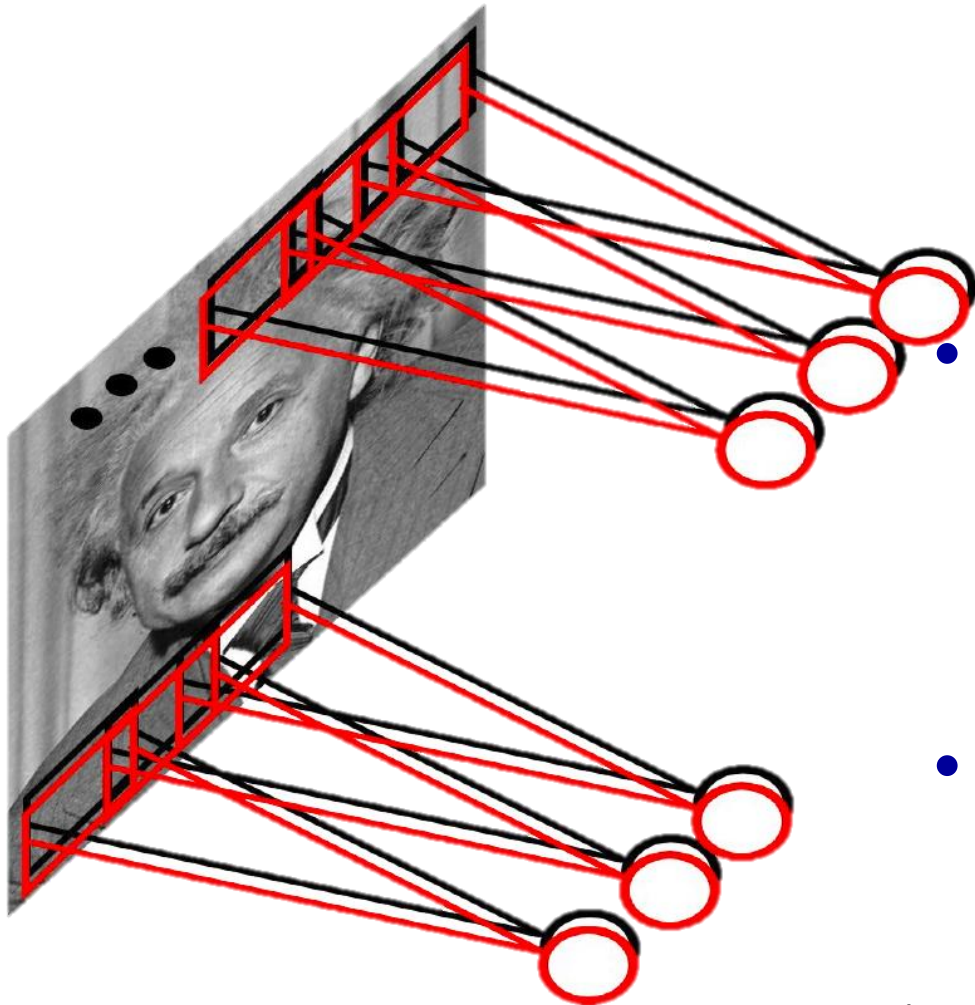
- Share the same parameters across different locations
- Convolutions with learned kernels

- Learn *multiple* filters

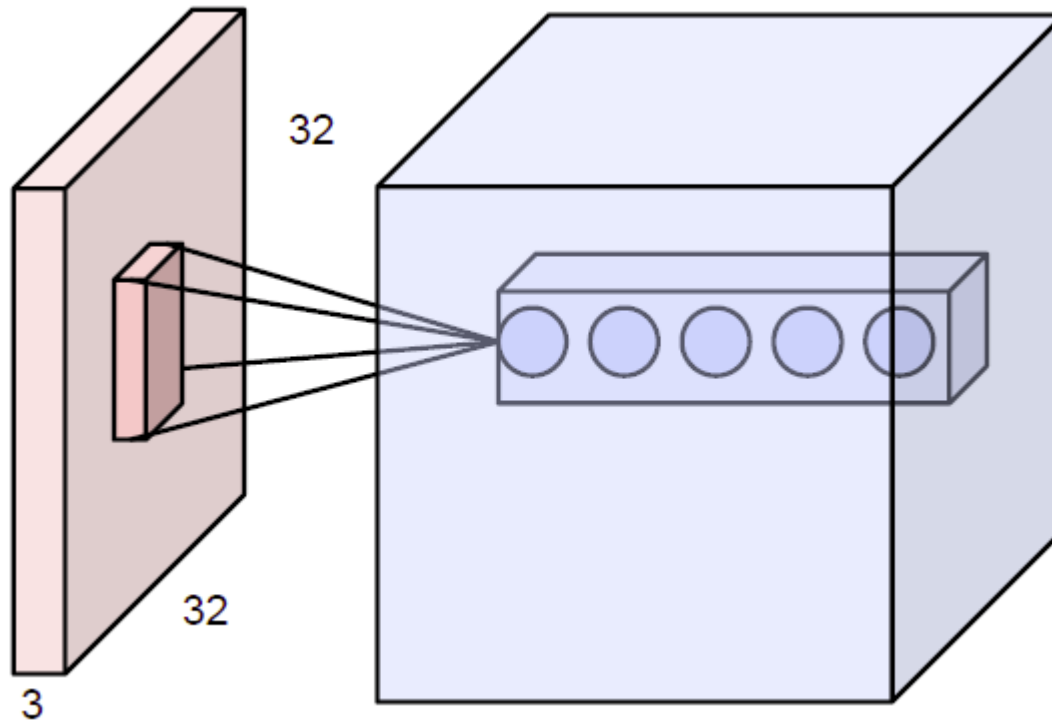
- E.g.  $1000 \times 1000$  image  
100 filters  
 $10 \times 10$  filter size  
⇒ only 10k parameters

- Result: Response map

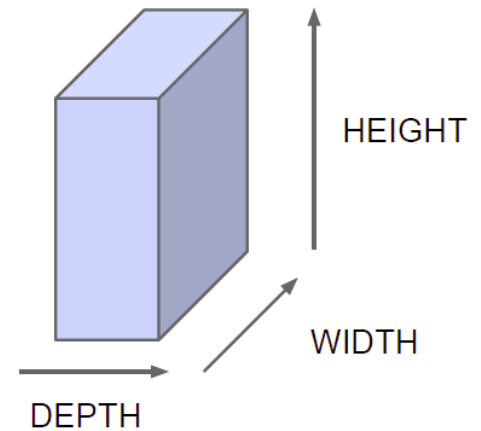
- size:  $1000 \times 1000 \times 100$
- Only memory, not params!



# Recap: Convolution Layers



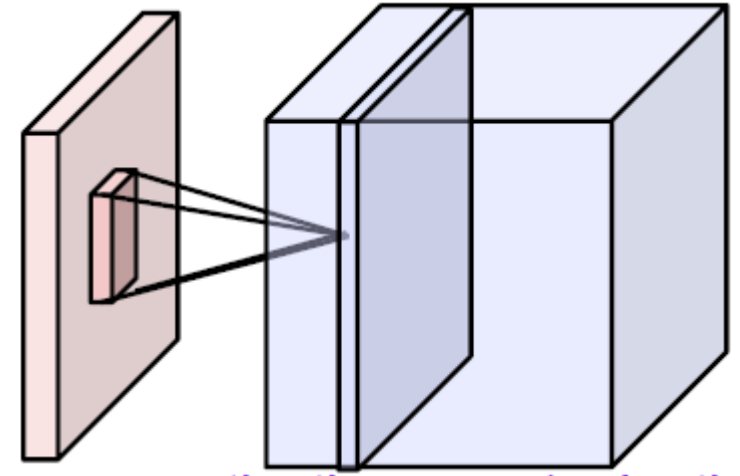
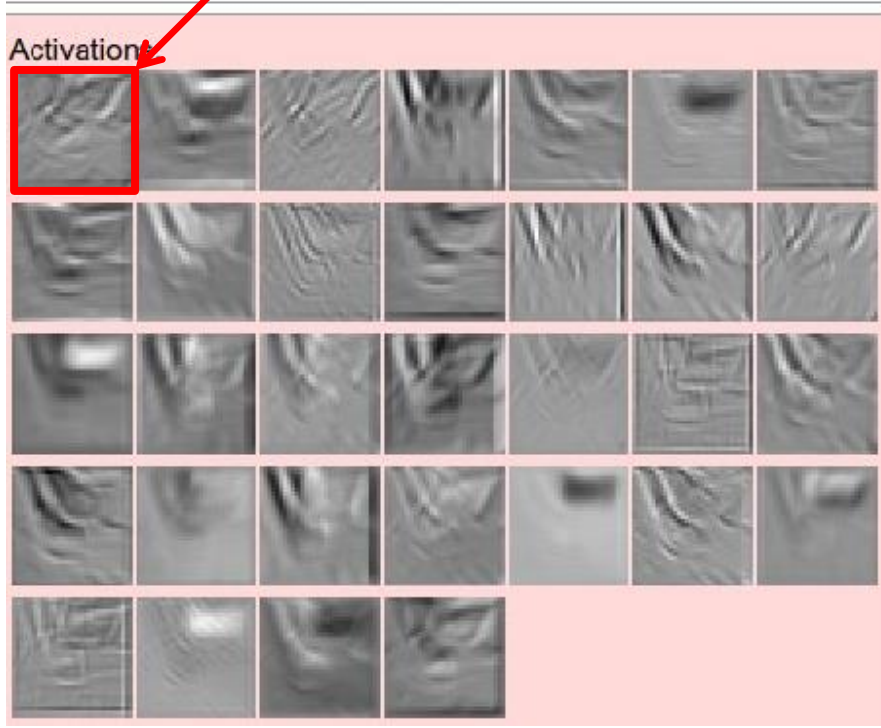
Naming convention:



- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth
  - Form a single  $[1 \times 1 \times \text{depth}]$  depth column in output volume.

# Recap: Activation Maps

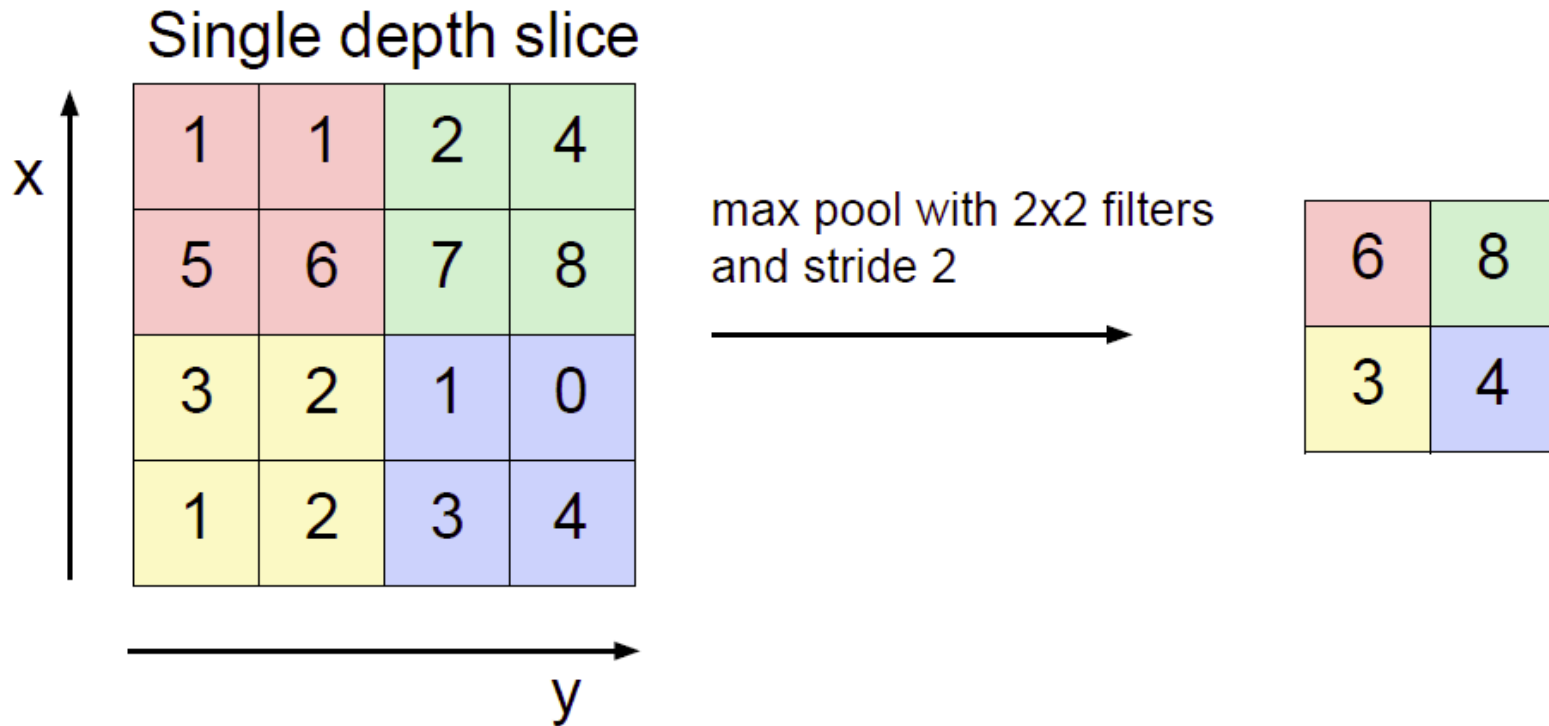
Activations:



Each activation map is a depth slice through the output volume.

Activation maps

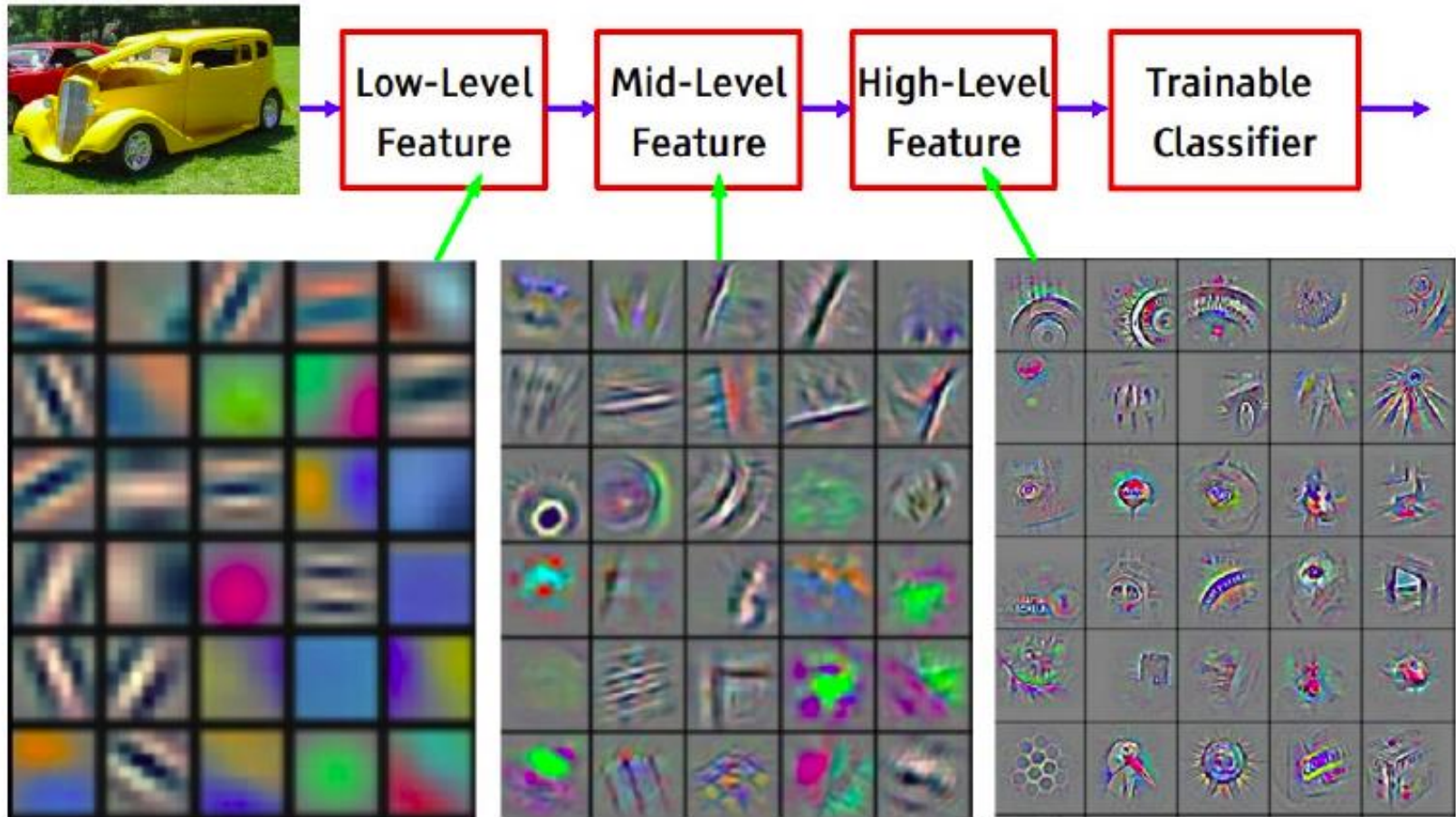
# Recap: Pooling Layers



- **Effect:**

- Make the representation smaller without losing too much information
- Achieve robustness to translations

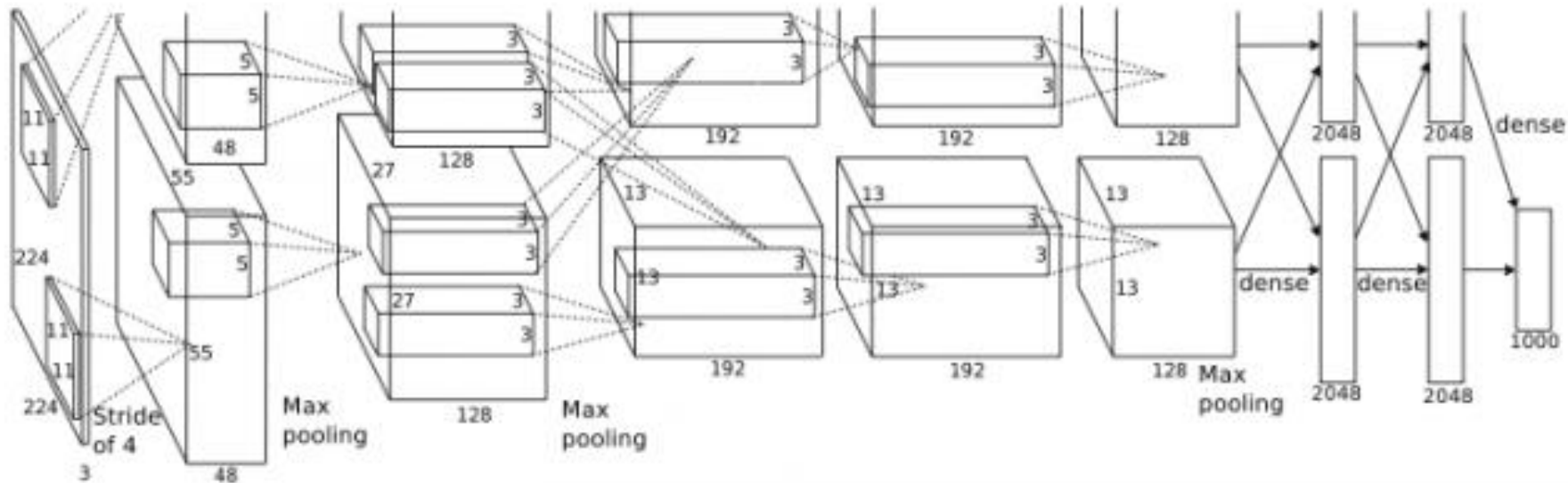
# Recap: Effect of Multiple Convolution Layers



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



# Recap: AlexNet (2012)



- **Similar framework as LeNet, but**
  - **Bigger model (7 hidden layers, 650k units, 60M parameters)**
  - **More data ( $10^6$  images instead of  $10^3$ )**
  - **GPU implementation**
  - **Better regularization and up-to-date tricks for training (Dropout)**

A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks](#), NIPS 2012.

# Recap: VGGNet (2014/15)

- Main ideas

- Deeper network
- Stacked convolutional layers with smaller filters (+ nonlinearity)
- Detailed evaluation of all components

- Results

- Improved ILSVRC top-5 error rate to 6.7%.

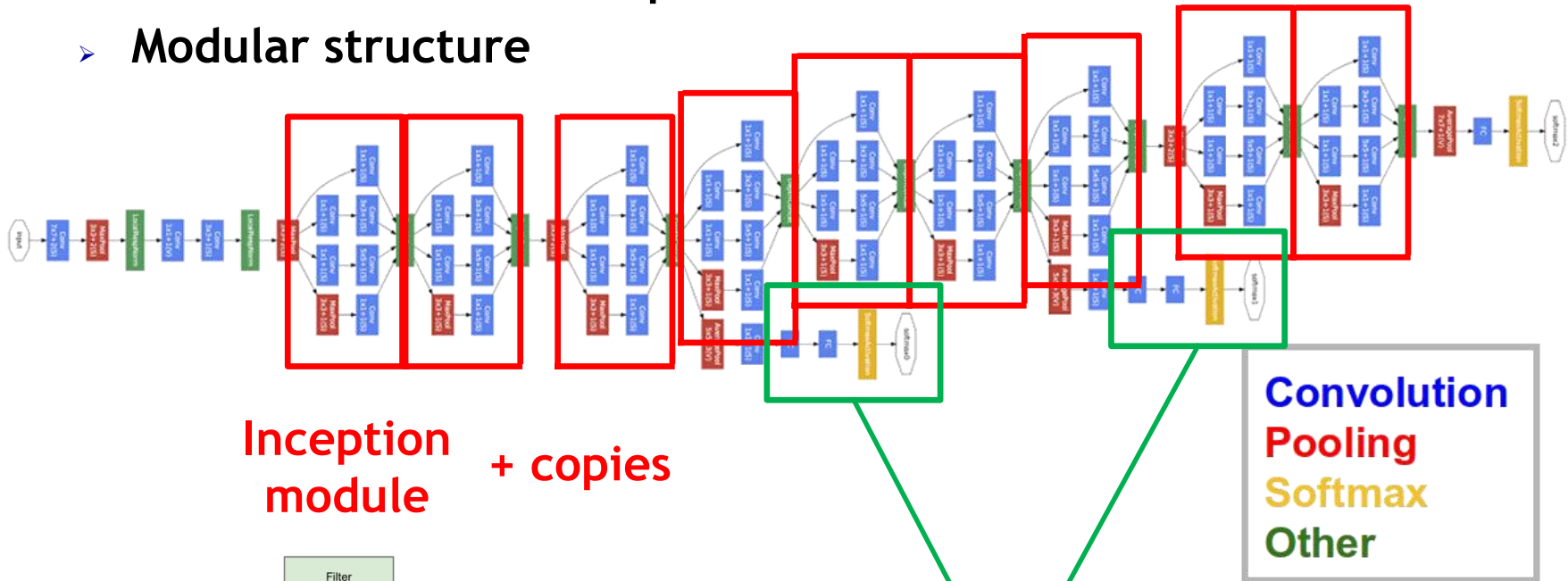
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 <b>LRN</b>	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Mainly used

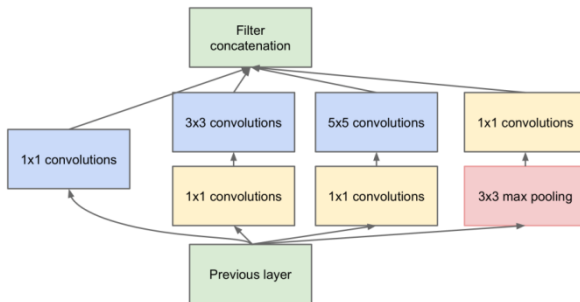


# Recap: GoogLeNet (2014)

- Ideas:
  - Learn features at multiple scales
  - Modular structure



Inception module + copies



(b) Inception module with dimension reductions

Auxiliary classification outputs for training the lower layers (deprecated)

**Convolution**  
**Pooling**  
**Softmax**  
**Other**

# Recap: Residual Networks

AlexNet, 8 layers  
(ILSVRC 2012)



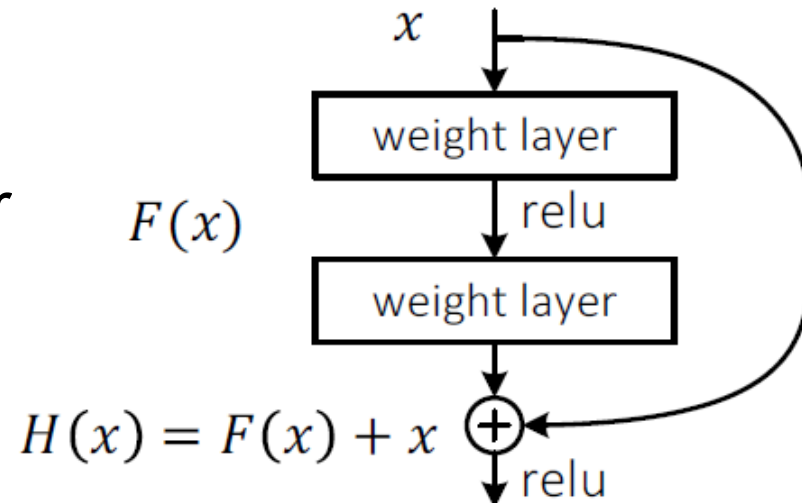
VGG, 19 layers  
(ILSVRC 2014)



ResNet, 152 layers  
(ILSVRC 2015)

- Core component

- Skip connections bypassing each layer
- Better propagation of gradients to the deeper layers
- This makes it possible to train (much) deeper networks.

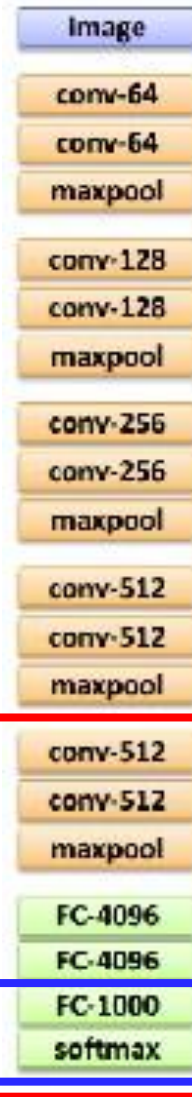


# Recap: Transfer Learning with CNNs



1. Train on ImageNet
2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

i.e., replace the Softmax layer at the end

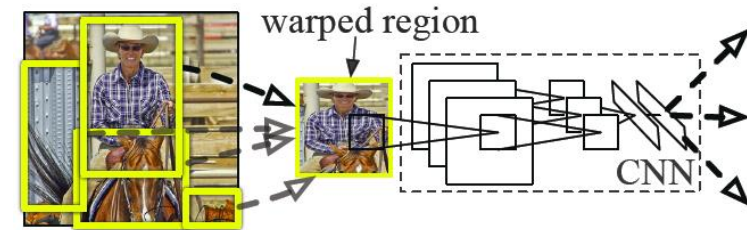


3. If you have a medium sized dataset, “finetune” instead: use the old weights as initialization, train the full network or only some of the higher layers.

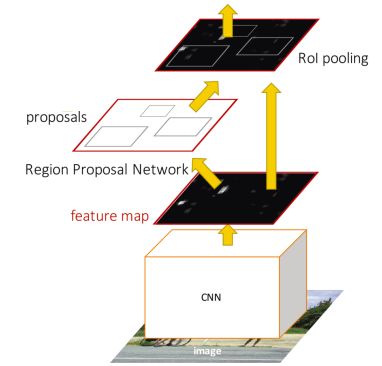
Retrain bigger part of the network

# Repetition

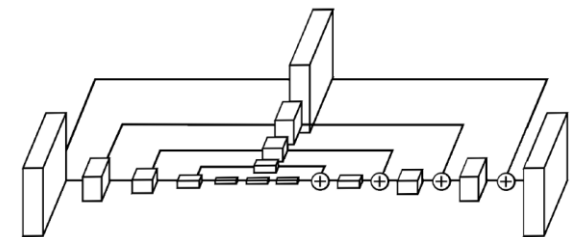
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
  - Sliding Window based Object Detection
  - Part-based Approaches
  - Deep Learning Approaches
- 3D Reconstruction
- Motion and Tracking



*R-CNN*



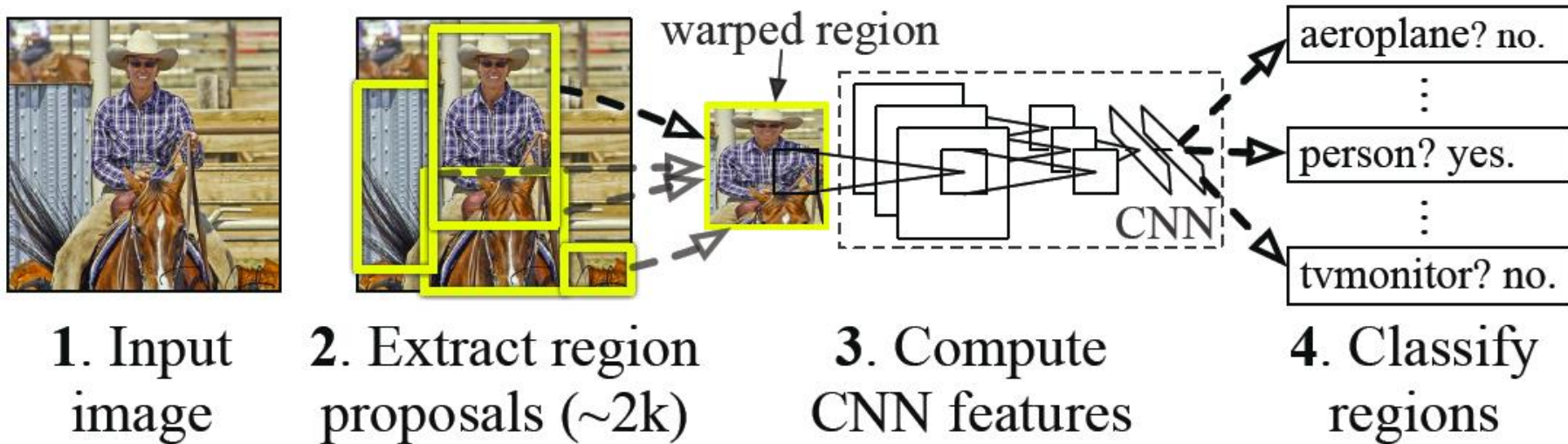
*Faster R-CNN*



*Fully Convolutional Networks*

# Recap: R-CNN for Object Detection

## R-CNN: *Regions with CNN features*



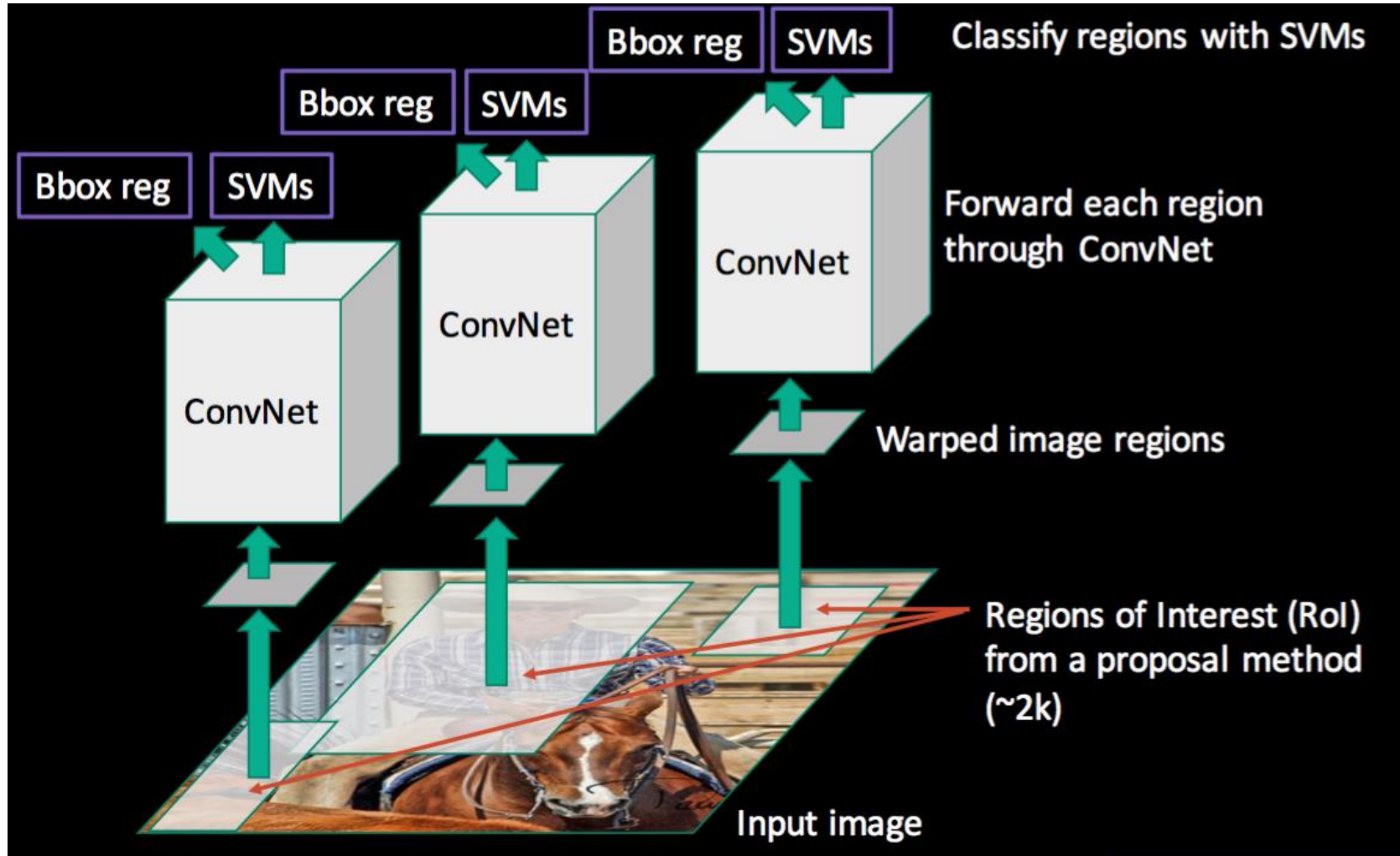
- **Key ideas**

- Extract region proposals (Selective Search)
- Use a pre-trained/fine-tuned classification network as feature extractor (initially AlexNet, later VGGNet) on those regions

R. Girshick, J. Donahue, T. Darrell, and J. Malik, [Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation](#), CVPR 2014



# Recap: R-CNN for Object Detection

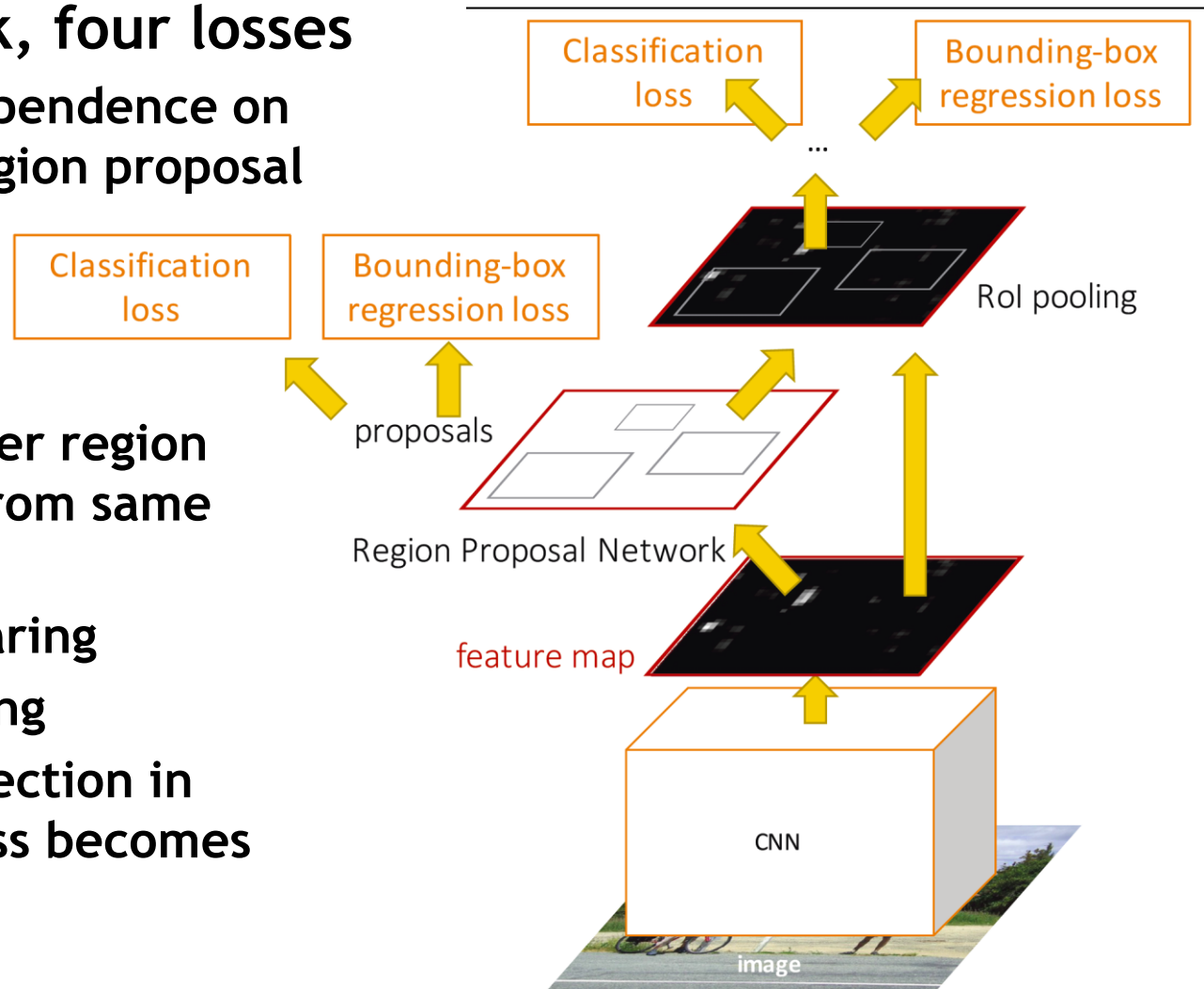


# Recap: Faster R-CNN

- One network, four losses

- Remove dependence on external region proposal algorithm.

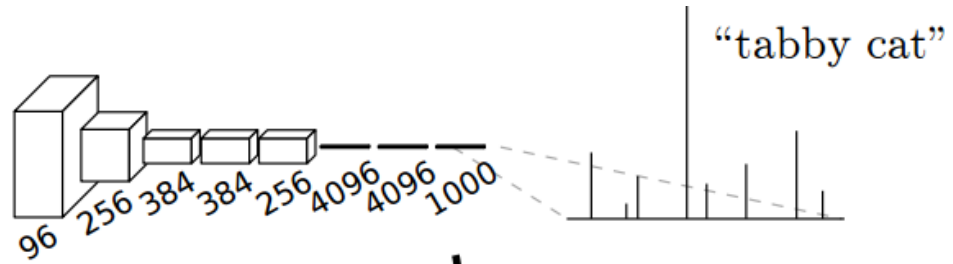
- Instead, infer region proposals from same CNN.
  - Feature sharing
  - Joint training
- ⇒ Object detection in a single pass becomes possible.



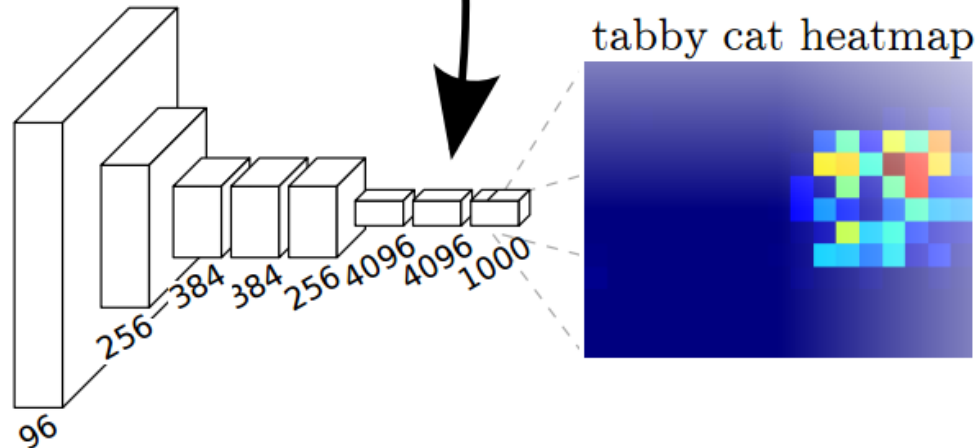


# Recap: Fully Convolutional Networks

- CNN



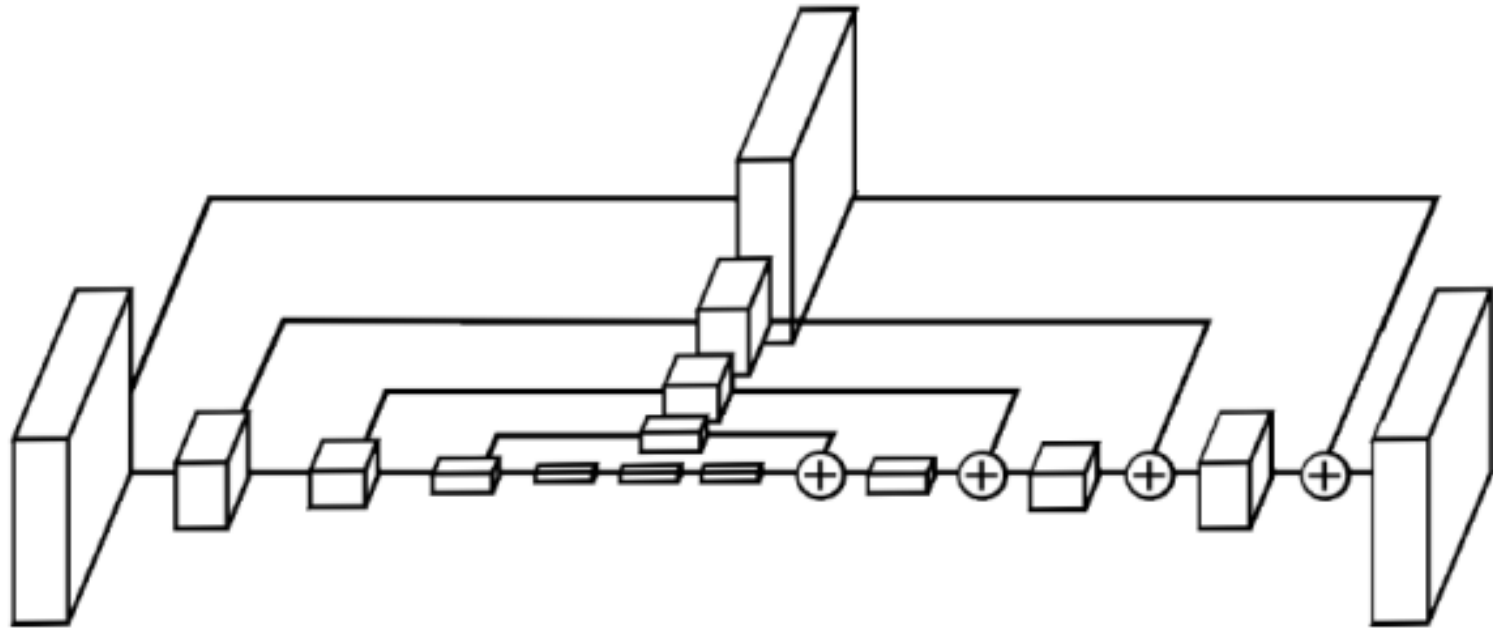
- FCN



- Intuition

- Think of FCNs as performing a sliding-window classification, producing a heatmap of output scores for each class

# Recap: Semantic Image Segmentation

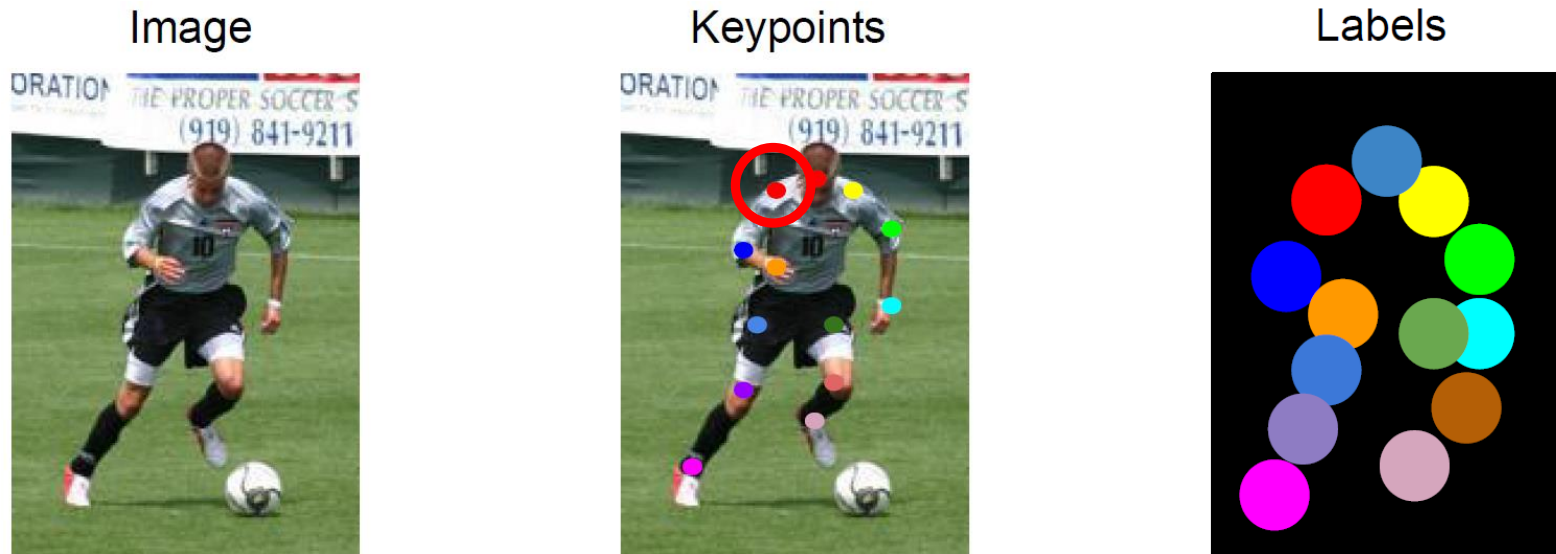


- **Encoder-Decoder Architecture**

- Problem: FCN output has low resolution
- Solution: perform upsampling to get back to desired resolution
- Use skip connections to preserve higher-resolution information

# Recap: FCNs for Human Pose Estimation

- **Input data**

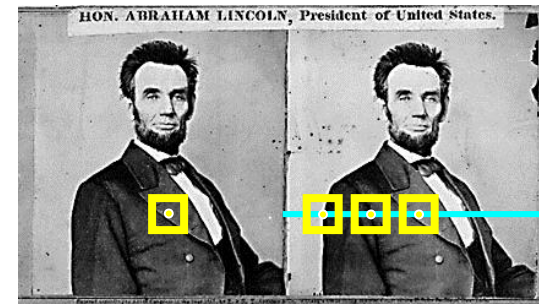
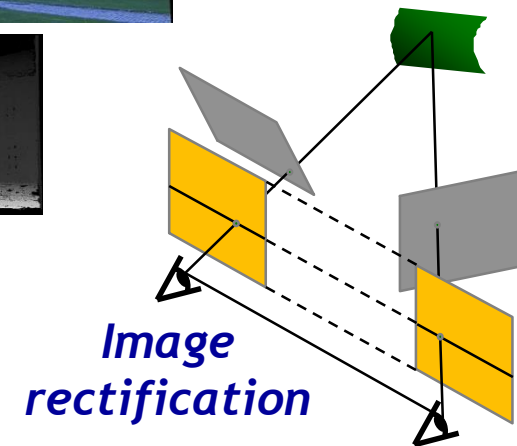
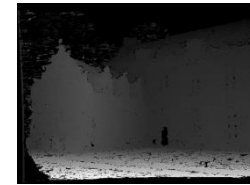
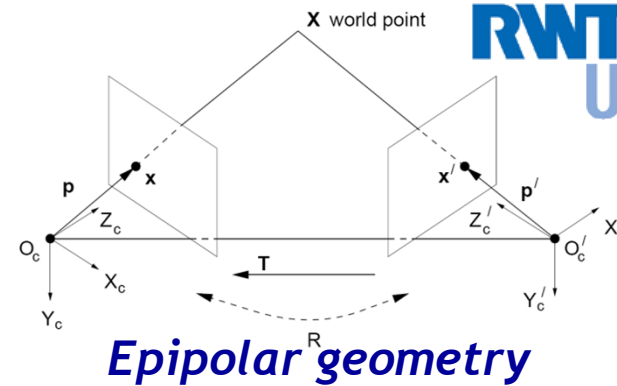


- **Task setup**

- Annotate images with keypoints for skeleton joints
- Define a target disk around each keypoint with radius  $r$
- Set the ground-truth label to 1 within each such disk
- Infer heatmaps for the joints as in semantic segmentation

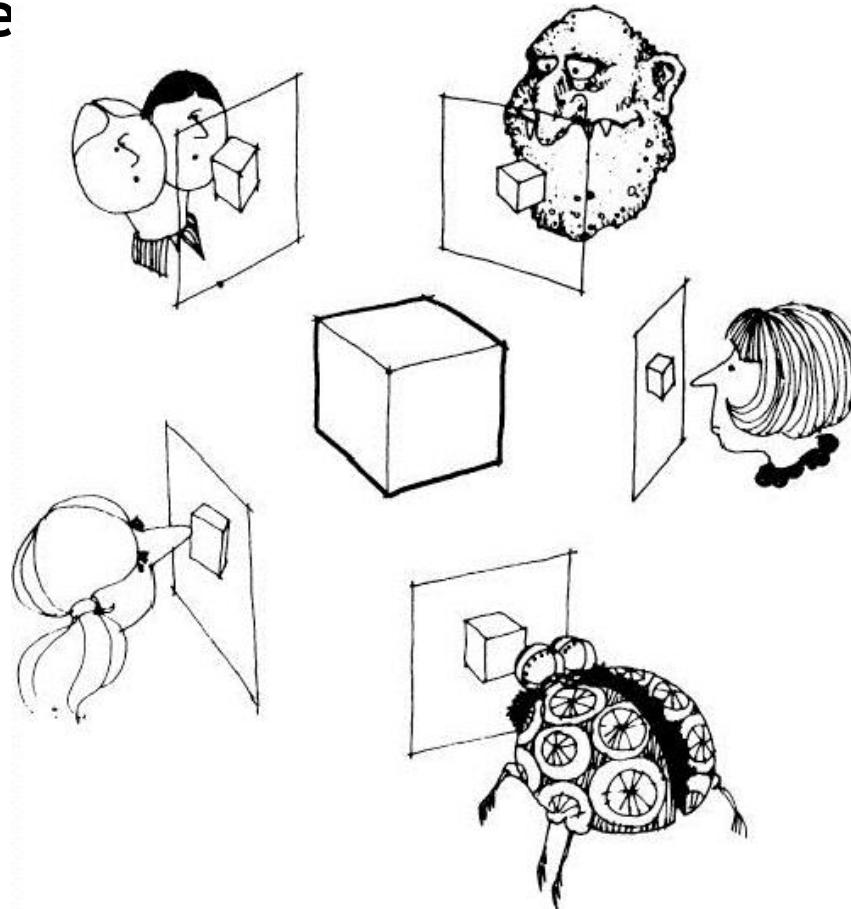
# Repetition

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
  - Epipolar Geometry and Stereo Basics
  - Camera Calibration & Uncalibrated Reconstruction
  - Structure-from-Motion
- Motion and Tracking

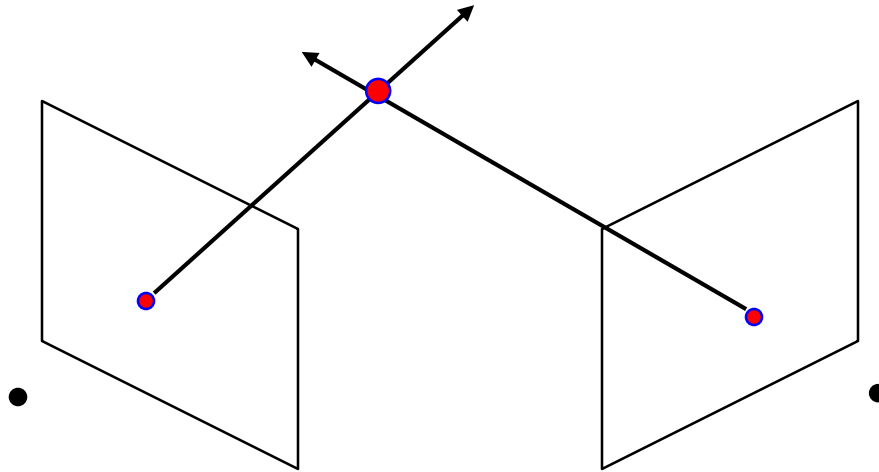


# Recap: What Is Stereo Vision?

- Generic problem formulation: given several images of the same object or scene, compute a representation of its 3D shape



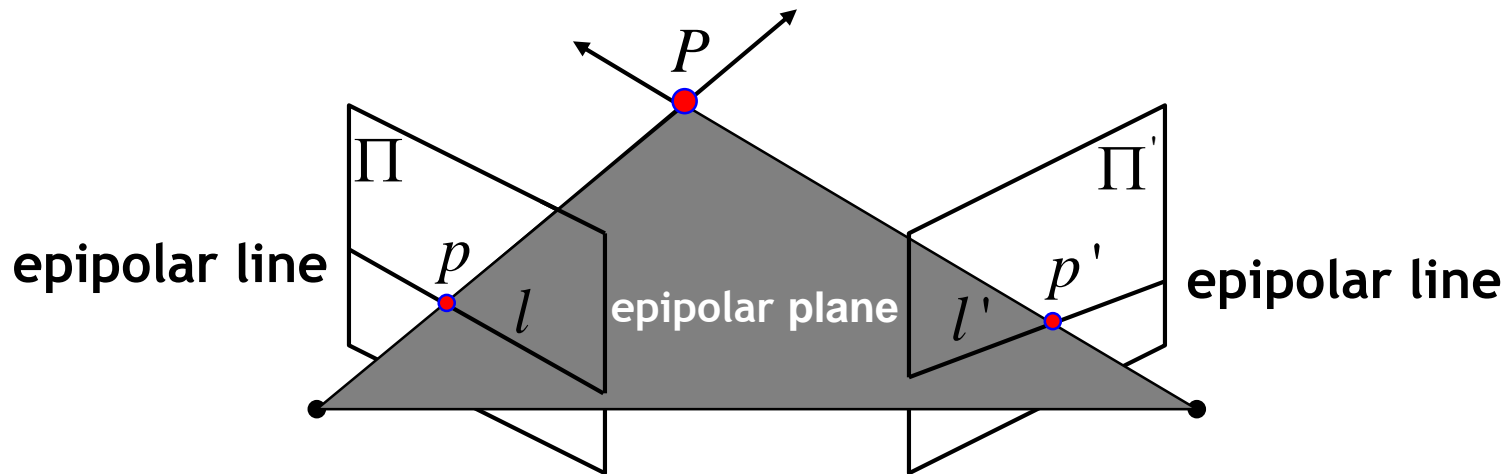
# Recap: Depth with Stereo - Basic Idea



- **Basic Principle: Triangulation**
  - Gives reconstruction as intersection of two rays
  - Requires
    - Camera pose (calibration)
    - Point correspondence

# Recap: Epipolar Geometry

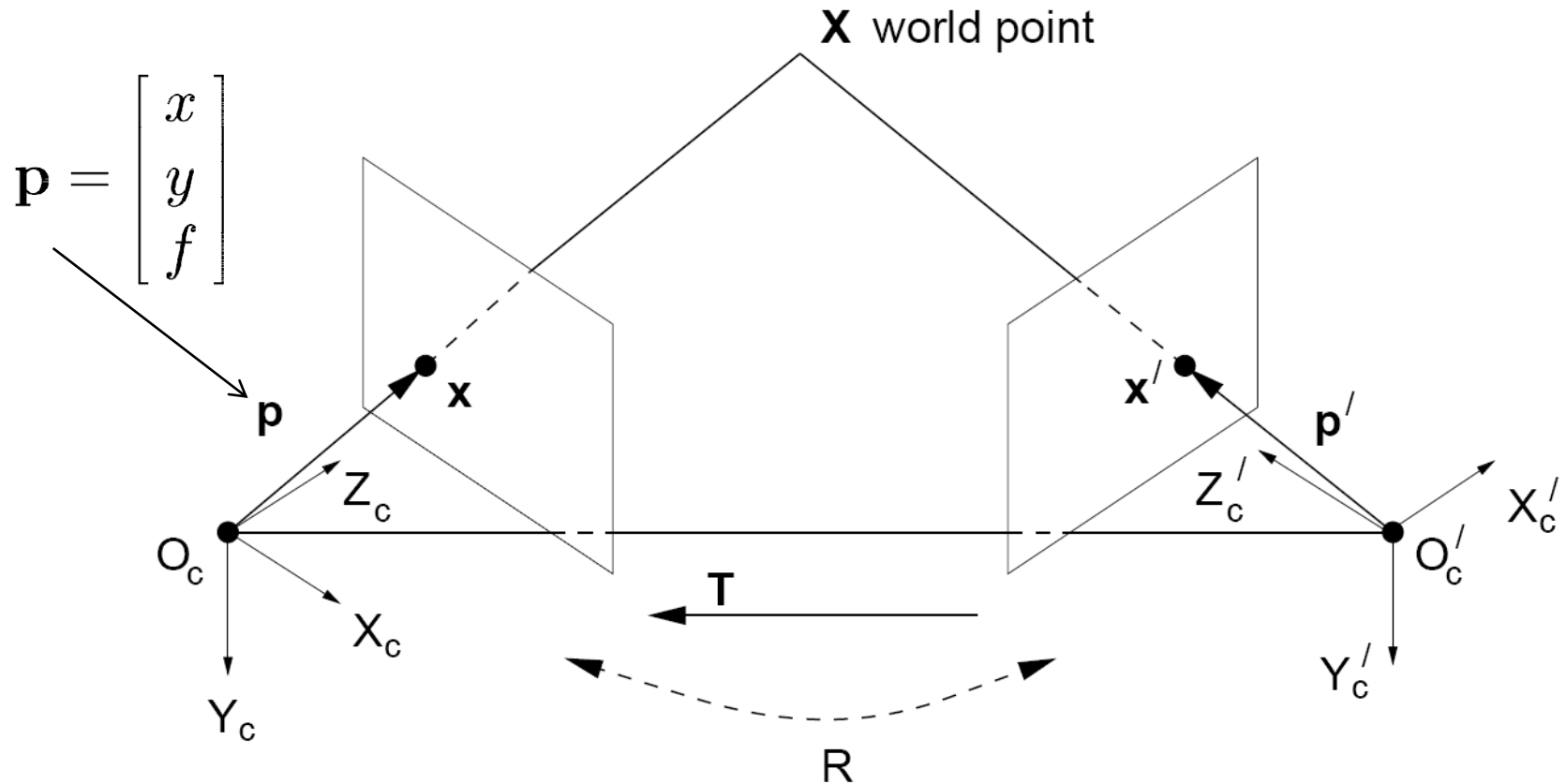
- Geometry of two views allows us to constrain where the corresponding pixel for some image point in the first view must occur in the second view.



- Epipolar constraint:
  - Correspondence for point  $p$  in  $\Pi$  must lie on the epipolar line  $l'$  in  $\Pi'$  (and vice versa).
  - Reduces correspondence problem to 1D search along conjugate epipolar lines.



# Recap: Stereo Geometry With Calibrated Cameras



- Camera-centered coordinate systems are related by known rotation  $\mathbf{R}$  and translation  $\mathbf{T}$ :

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T}$$

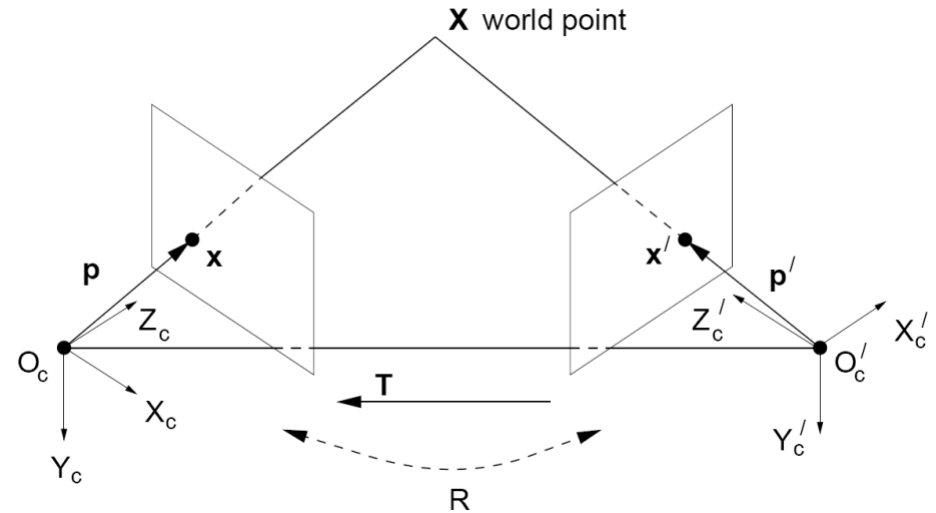
# Recap: Essential Matrix

$$\mathbf{X}' \cdot (\mathbf{T} \times \mathbf{R}\mathbf{X}) = 0$$

$$\mathbf{X}' \cdot (\mathbf{T}_x \mathbf{R}\mathbf{X}) = 0$$

Let  $\mathbf{E} = \mathbf{T}_x \mathbf{R}$

$$\mathbf{X}'^T \mathbf{E} \mathbf{X} = 0$$



- This holds for the rays  $p$  and  $p'$  that are parallel to the camera-centered position vectors  $X$  and  $X'$ , so we have:

$$\mathbf{p}'^T \mathbf{E} \mathbf{p} = 0$$

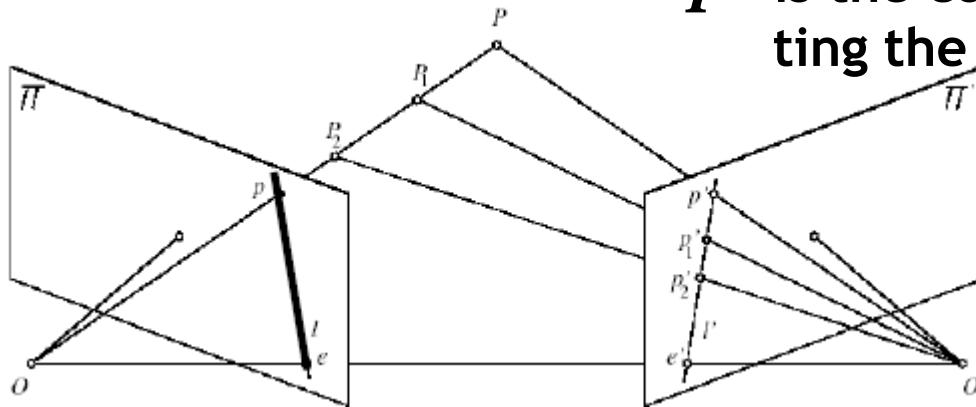
- $\mathbf{E}$  is called the essential matrix, which relates corresponding image points [Longuet-Higgins 1981]

# Recap: Essential Matrix and Epipolar Lines

$$\mathbf{p}'^T \mathbf{E} \mathbf{p} = 0$$

Epipolar constraint: if we observe point  $p$  in one image, then its position  $p'$  in second image must satisfy this equation.

$\mathbf{l}' = \mathbf{E} \mathbf{p}$  is the coordinate vector representing the epipolar line for point  $p$

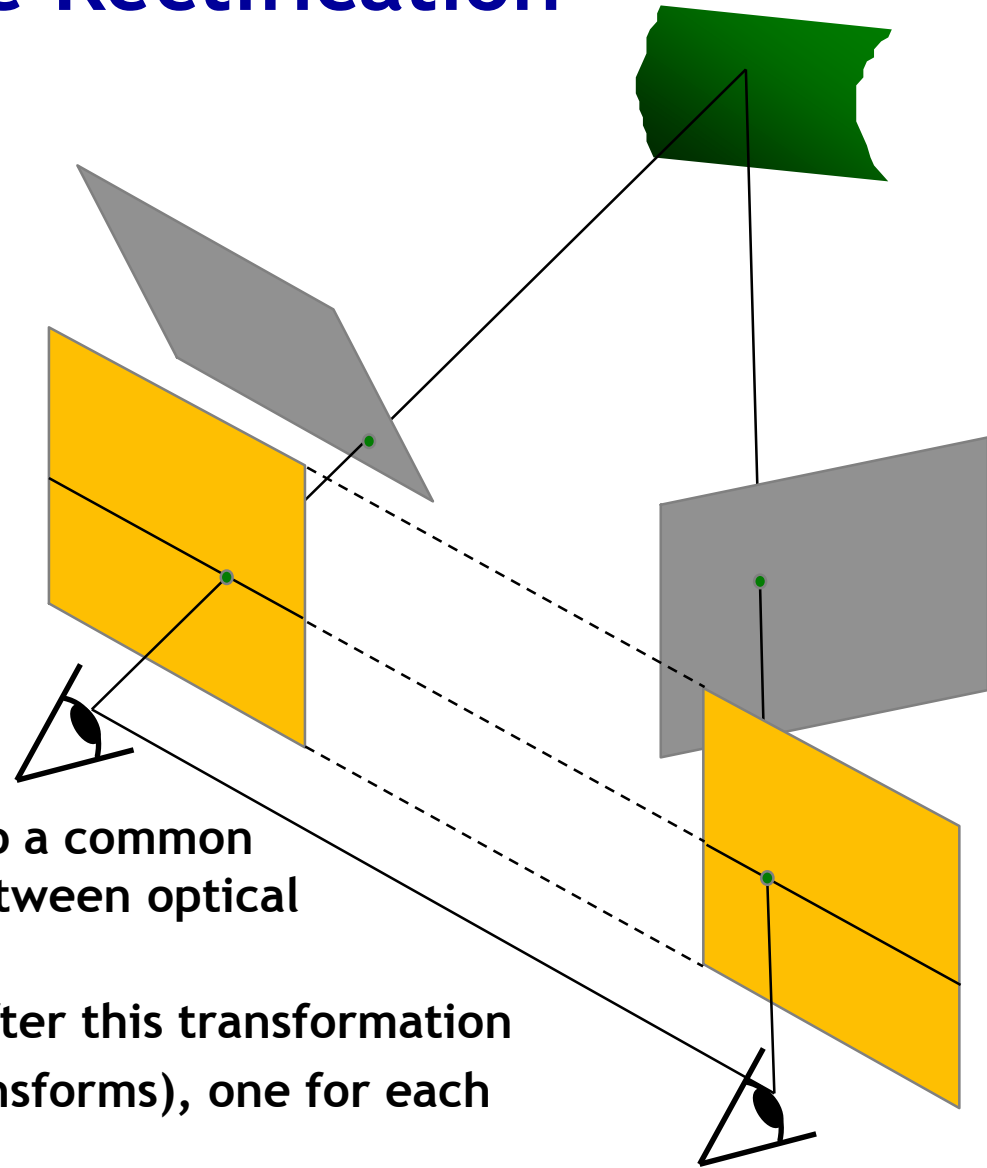


(i.e., the line is given by:  $\mathbf{l}'^T \mathbf{x} = 0$ )

$\mathbf{l} = \mathbf{E}^T \mathbf{p}'$  is the coordinate vector representing the epipolar line for point  $p'$

# Recap: Stereo Image Rectification

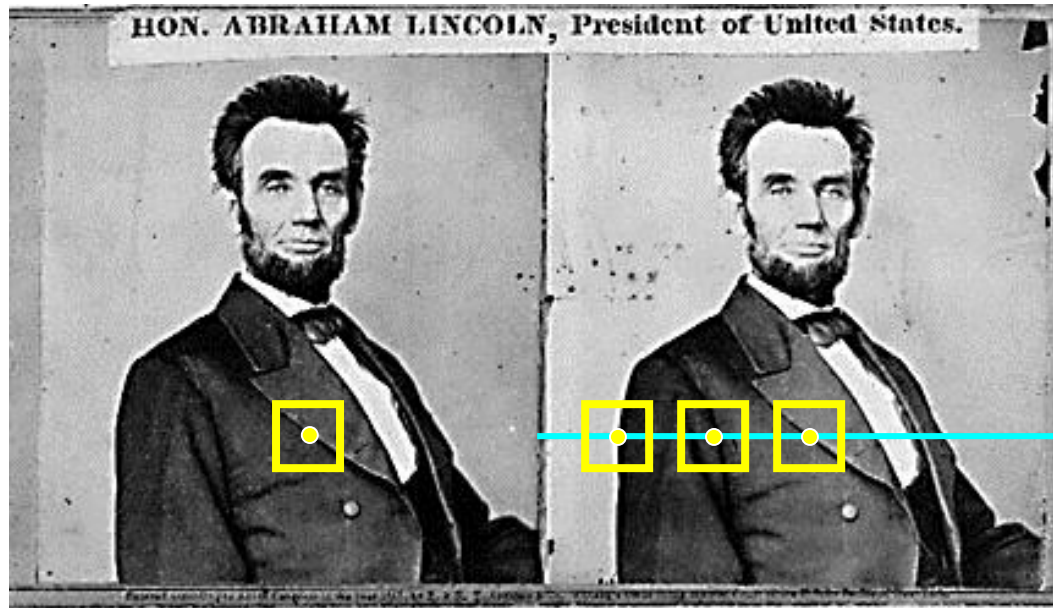
- In practice, it is convenient if image scanlines are the epipolar lines.



- **Algorithm**

- Reproject image planes onto a common plane parallel to the line between optical centers
- Pixel motion is horizontal after this transformation
- Two homographies (3x3 transforms), one for each input image reprojection

# Recap: Dense Correspondence Search

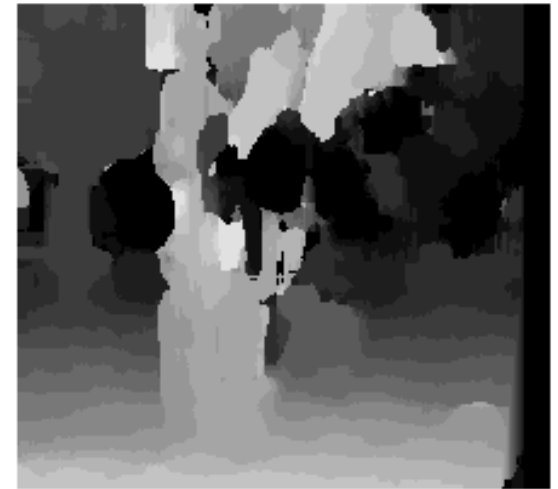


- For each pixel in the first image
  - Find corresponding epipolar line in the right image
  - Examine all pixels on the epipolar line and pick the best match (e.g. SSD, correlation)
  - Triangulate the matches to get depth information
- This is easiest when epipolar lines are scanlines  
⇒ Rectify images first

# Recap: Effect of Window Size



$W = 3$



$W = 20$

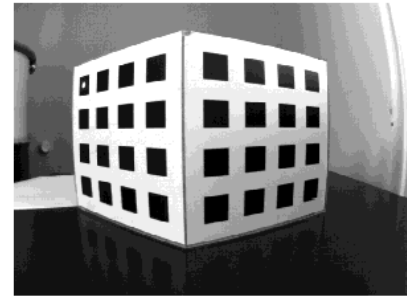
Want window large enough to have sufficient intensity variation, yet small enough to contain only pixels with about the same disparity.

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

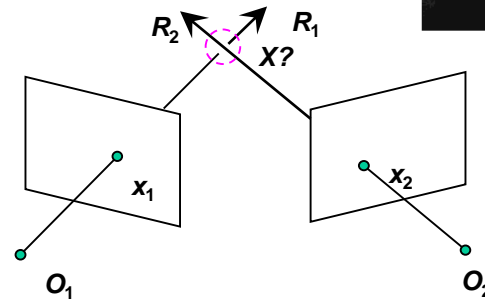
*Camera models*

# Repetition

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
  - Epipolar Geometry and Stereo Basics
  - **Camera Calibration & Uncalibrated Reconstruction**
  - Structure-from-Motion
- Motion and Tracking



*Camera calibration*



*Triangulation*

*Essential matrix,  
Fundamental matrix*

$$x^T E x' = 0$$

$$x^T F x' = 0$$

$$\begin{bmatrix} u_1 u'_1 & u_1 v'_1 & u_1 & v_1 u'_1 & v_1 v'_1 & v_1 & u'_1 & v'_1 & 1 \\ u_2 u'_2 & u_2 v'_2 & u_2 & v_2 u'_2 & v_2 v'_2 & v_2 & u'_2 & v'_2 & 1 \\ u_3 u'_3 & u_3 v'_3 & u_3 & v_3 u'_3 & v_3 v'_3 & v_3 & u'_3 & v'_3 & 1 \\ u_4 u'_4 & u_4 v'_4 & u_4 & v_4 u'_4 & v_4 v'_4 & v_4 & u'_4 & v'_4 & 1 \\ u_5 u'_5 & u_5 v'_5 & u_5 & v_5 u'_5 & v_5 v'_5 & v_5 & u'_5 & v'_5 & 1 \\ u_6 u'_6 & u_6 v'_6 & u_6 & v_6 u'_6 & v_6 v'_6 & v_6 & u'_6 & v'_6 & 1 \\ u_7 u'_7 & u_7 v'_7 & u_7 & v_7 u'_7 & v_7 v'_7 & v_7 & u'_7 & v'_7 & 1 \\ u_8 u'_8 & u_8 v'_8 & u_8 & v_8 u'_8 & v_8 v'_8 & v_8 & u'_8 & v'_8 & 1 \end{bmatrix} = 0$$

*Eight-point algorithm*

**SVD!**



# Recap: A General Point

- Equations of the form

$$Ax = 0$$

- How do we solve them? (always!)

- Apply SVD

$$\begin{array}{c} \text{SVD} \\ \downarrow \\ A = UDV^T = U \end{array}
 \begin{bmatrix} d_{11} & & & \\ & \ddots & & \\ & & d_{NN} & \\ & & & \end{bmatrix}
 \begin{bmatrix} v_{11} & \cdots & v_{1N} \\ \vdots & \ddots & \vdots \\ v_{N1} & \cdots & v_{NN} \end{bmatrix}^T$$

Singular values    Singular vectors

- Singular values of  $A$  = square roots of the eigenvalues of  $A^T A$ .
- The solution of  $Ax=0$  is the *nullspace* vector of  $A$ .
- This corresponds to the *smallest singular vector* of  $A$ .

# Recap: Camera Parameters

- Intrinsic parameters

- Principal point coordinates
- Focal length
- Pixel magnification factors
- *Skew (non-rectangular pixels)*
- *Radial distortion*

$$K = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & s & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & s' & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

- Extrinsic parameters

- Rotation R
- Translation t  
(both relative to world coordinate system)

- Camera projection matrix

$$P = K [R | t]$$

- ⇒ General pinhole camera: 9 DoF
- ⇒ CCD Camera with square pixels: 10 DoF
- ⇒ General camera: 11 DoF

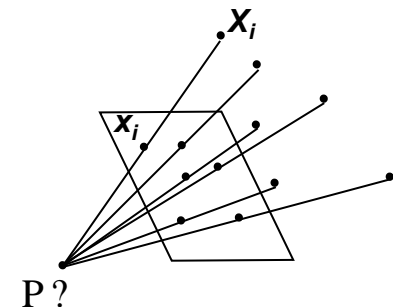
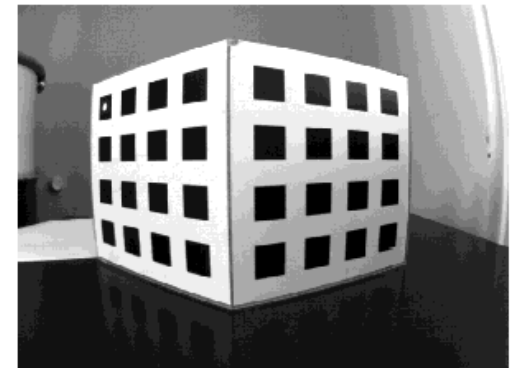
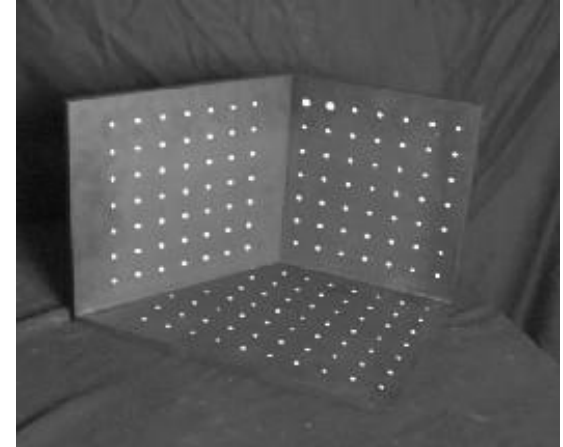
# Recap: Calibrating a Camera

## Goal

- Compute intrinsic and extrinsic parameters using observed camera data.

## Main idea

- Place “calibration object” with known geometry in the scene
- Get correspondences
- Solve for mapping from scene to image: estimate  $P = P_{\text{int}} P_{\text{ext}}$



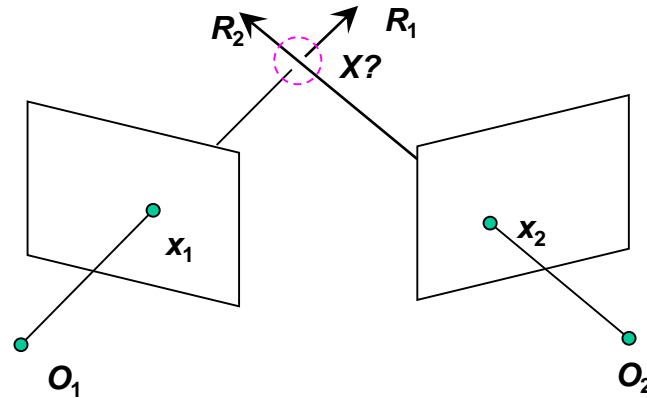
# Recap: Camera Calibration (DLT Algorithm)

$$\begin{bmatrix} 0^T & \mathbf{X}_1^T & -y_1 \mathbf{X}_1^T \\ \mathbf{X}_1^T & 0^T & -x_1 \mathbf{X}_1^T \\ \dots & \dots & \dots \\ 0^T & \mathbf{X}_n^T & -y_n \mathbf{X}_n^T \\ \mathbf{X}_n^T & 0^T & -x_n \mathbf{X}_n^T \end{bmatrix} \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix} = 0 \quad \mathbf{A}\mathbf{p} = 0$$

- **P has 11 degrees of freedom.**
- **Two linearly independent equations per independent 2D/3D correspondence.**
- **Solve with SVD (similar to homography estimation)**
  - **Solution corresponds to smallest singular vector.**
- **5 ½ correspondences needed for a minimal solution.**

# Recap: Triangulation - Lin. Alg. Approach

see  
Exercise 6.3!

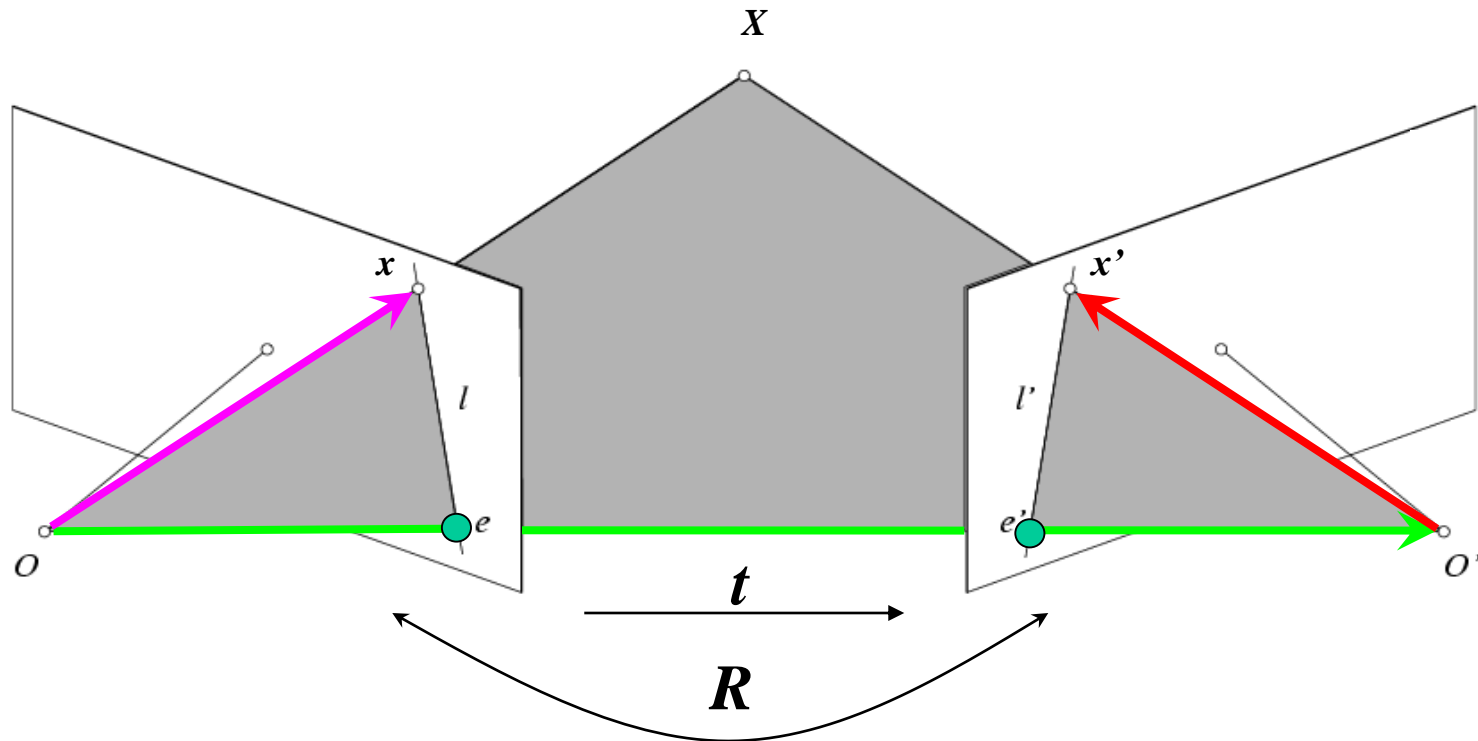


$$\lambda_1 x_1 = P_1 X \quad x_1 \times P_1 X = 0 \quad [x_{1 \times}] P_1 X = 0$$

$$\lambda_2 x_2 = P_2 X \quad x_2 \times P_2 X = 0 \quad [x_{2 \times}] P_2 X = 0$$

- Two independent equations each in terms of three unknown entries of  $X$ .
- Stack equations and solve with SVD.
- This approach nicely generalizes to multiple cameras.

# Recap: Epipolar Geometry - Calibrated Case



Camera matrix:  $[I|0]$

$$X = (u, v, w, 1)^T$$

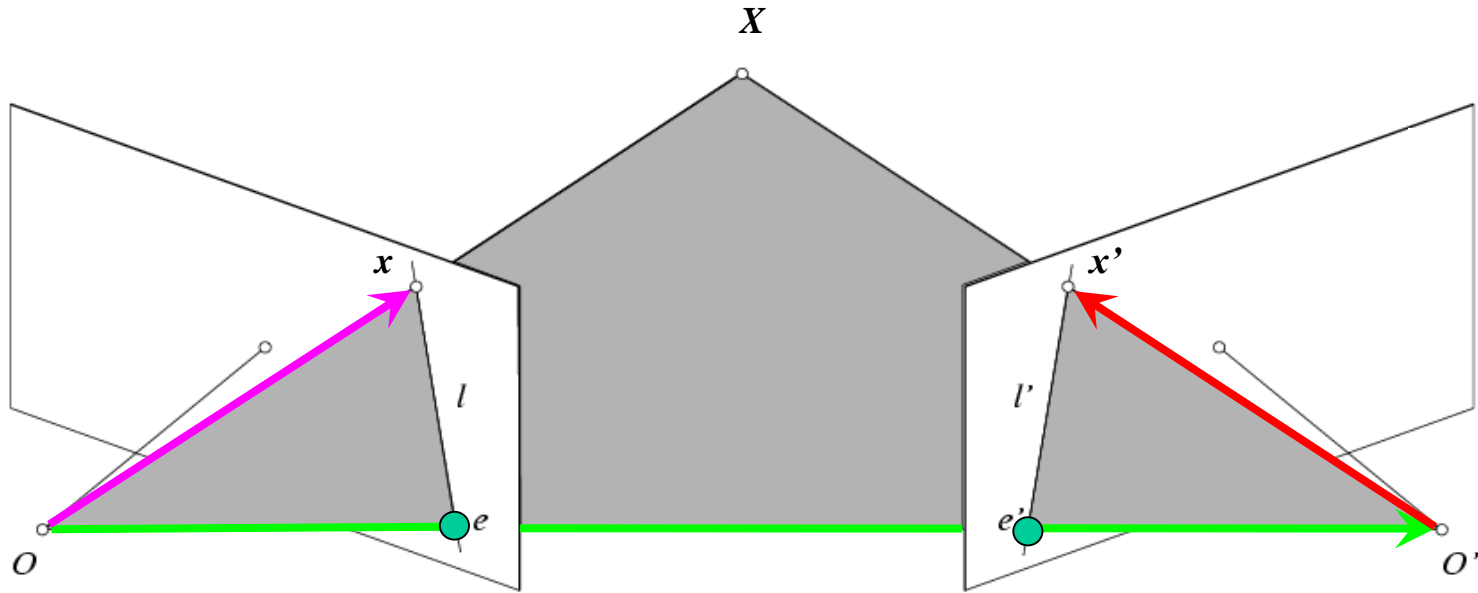
$$x = (u, v, w)^T$$

Camera matrix:  $[R^T | -R^T t]$

Vector  $x'$  in second coord. system has coordinates  $Rx'$  in the first one.

The vectors  $x$ ,  $t$ , and  $Rx'$  are coplanar

# Recap: Epipolar Geometry - Calibrated Case

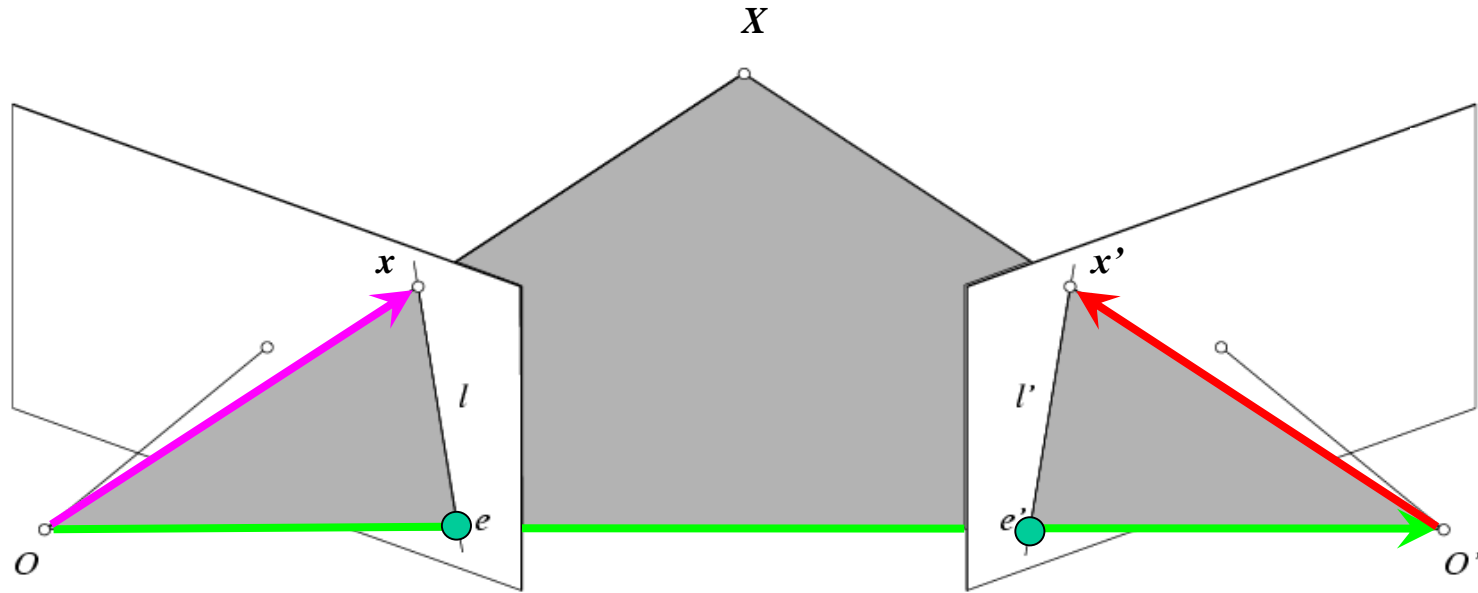


$$x \cdot [t \times (Rx')] = 0 \quad \Rightarrow \quad x^T E x' = 0 \quad \text{with} \quad E = [t_{\times}] R$$

**Essential Matrix**  
**(Longuet-Higgins, 1981)**



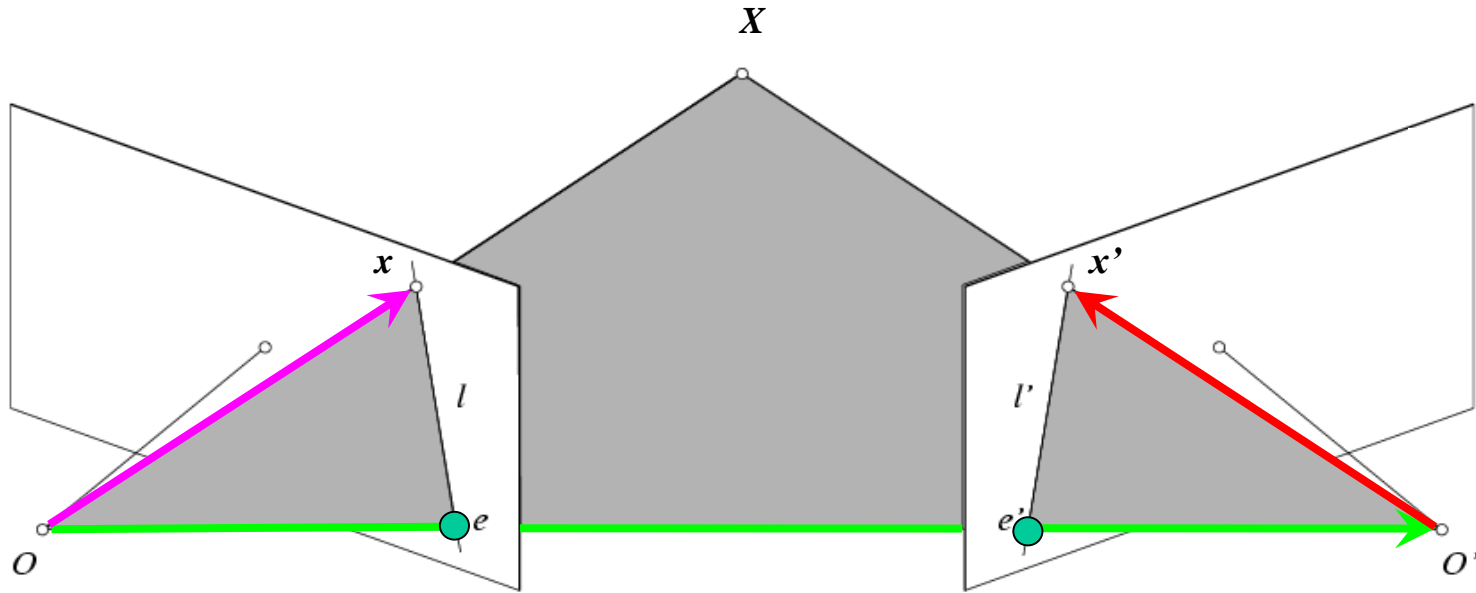
# Recap: Epipolar Geometry - Calibrated Case



$$x \cdot [t \times (Rx')] = 0 \quad \Rightarrow \quad x^T E x' = 0 \quad \text{with} \quad E = [t_{\times}] R$$

- $E x'$  is the epipolar line associated with  $x'$  ( $l = E x'$ )
- $E^T x$  is the epipolar line associated with  $x$  ( $l' = E^T x$ )
- $E e' = 0$  and  $E^T e = 0$
- $E$  is singular (rank two)
- $E$  has five degrees of freedom (up to scale)

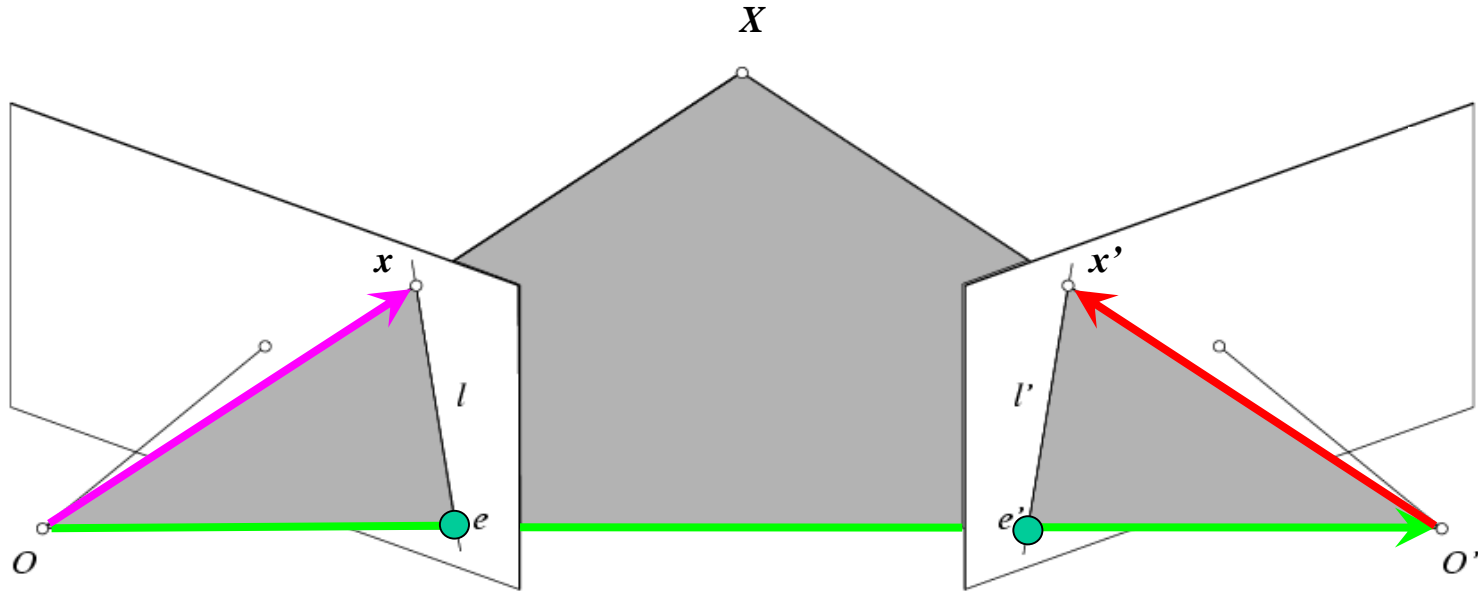
# Recap: Epipolar Geometry - Uncalibrated Case



- The calibration matrices  $K$  and  $K'$  of the two cameras are unknown
- We can write the epipolar constraint in terms of *unknown* normalized coordinates:

$$\hat{x}^T E \hat{x}' = 0 \quad x = K \hat{x}, \quad x' = K' \hat{x}'$$

# Recap: Epipolar Geometry - Uncalibrated Case



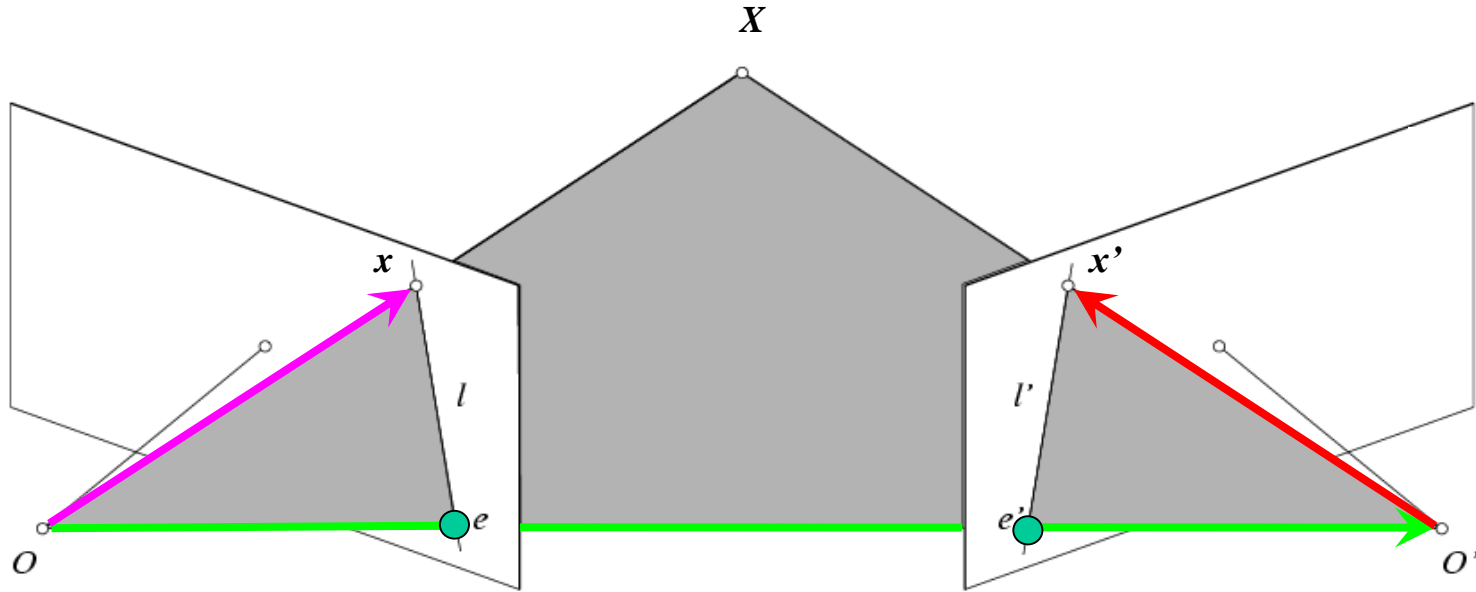
$$\hat{x}^T E \hat{x}' = 0 \quad \Rightarrow \quad x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

$$x = K \hat{x}$$

$$x' = K' \hat{x}'$$

**Fundamental Matrix  
(Faugeras and Luong, 1992)**

# Recap: Epipolar Geometry - Uncalibrated Case



$$\hat{x}^T E \hat{x}' = 0 \quad \longrightarrow \quad x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

- $F x'$  is the epipolar line associated with  $x'$  ( $l = F x'$ )
- $F^T x$  is the epipolar line associated with  $x$  ( $l' = F^T x$ )
- $F e' = 0$  and  $F^T e = 0$
- $F$  is singular (rank two)
- $F$  has seven degrees of freedom

see Exercise 6.1!

# Recap: The Eight-Point Algorithm

$$x = (u, v, 1)^T, \quad x' = (u', v', 1)^T$$

$$(u, v, 1) \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0 \quad \rightarrow \quad (uu', uv', u, vu', vv', v, u', v', 1) \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = 0$$

$$\begin{bmatrix} u_1 u'_1 & u_1 v'_1 & u_1 & v_1 u'_1 & v_1 v'_1 & v_1 & u'_1 & v'_1 & 1 \\ u_2 u'_2 & u_2 v'_2 & u_2 & v_2 u'_2 & v_2 v'_2 & v_2 & u'_2 & v'_2 & 1 \\ u_3 u'_3 & u_3 v'_3 & u_3 & v_3 u'_3 & v_3 v'_3 & v_3 & u'_3 & v'_3 & 1 \\ u_4 u'_4 & u_4 v'_4 & u_4 & v_4 u'_4 & v_4 v'_4 & v_4 & u'_4 & v'_4 & 1 \\ u_5 u'_5 & u_5 v'_5 & u_5 & v_5 u'_5 & v_5 v'_5 & v_5 & u'_5 & v'_5 & 1 \\ u_6 u'_6 & u_6 v'_6 & u_6 & v_6 u'_6 & v_6 v'_6 & v_6 & u'_6 & v'_6 & 1 \\ u_7 u'_7 & u_7 v'_7 & u_7 & v_7 u'_7 & v_7 v'_7 & v_7 & u'_7 & v'_7 & 1 \\ u_8 u'_8 & u_8 v'_8 & u_8 & v_8 u'_8 & v_8 v'_8 & v_8 & u'_8 & v'_8 & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0$$

- 1.) Solve with SVD. This minimizes  $\sum_{i=1}^N (x_i^T F x'_i)^2$
- 2.) Enforce rank-2 constraint using SVD

- Problem: poor numerical conditioning

# Recap: Normalized Eight-Point Alg.

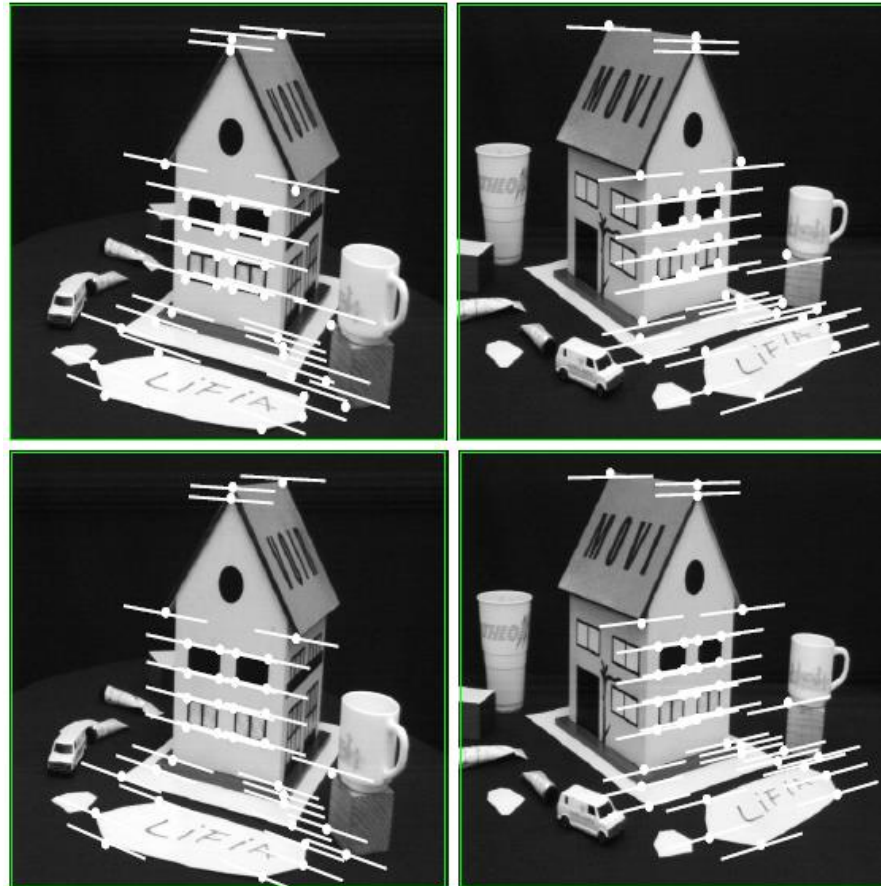
1. Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels.
2. Use the eight-point algorithm to compute  $F$  from the normalized points.
3. Enforce the rank-2 constraint using SVD.

$$F \stackrel{\text{SVD}}{=} U D V^T = U \begin{bmatrix} d_{11} & & & \\ & d_{22} & & \\ & & d_{33} & \\ & & & \dots \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{13} \\ \vdots & \ddots & \vdots \\ v_{31} & \cdots & v_{33} \end{bmatrix}^T$$

Set  $d_{33}$  to zero and reconstruct  $F$

4. Transform fundamental matrix back to original units: if  $T$  and  $T'$  are the normalizing transformations in the two images, then the fundamental matrix in original coordinates is  $T^T F T'$ .

# Recap: Comparison of Estimation Algorithms

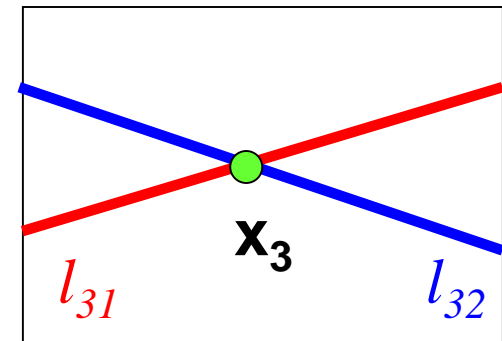
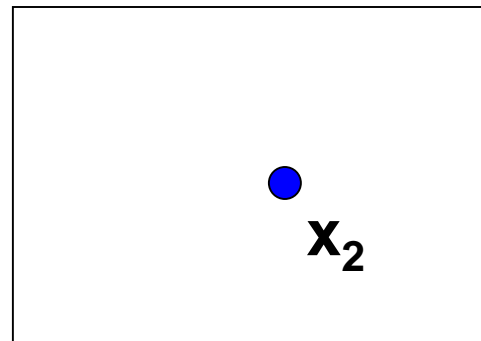
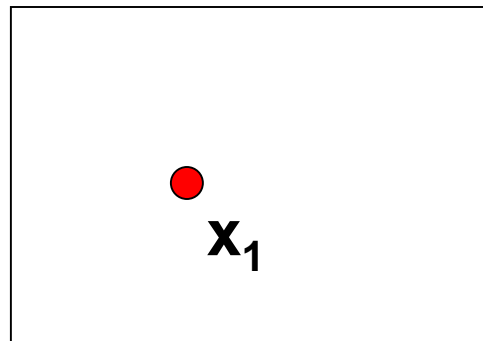


	8-point	Normalized 8-point	Nonlinear least squares
Av. Dist. 1	2.33 pixels	0.92 pixel	0.86 pixel
Av. Dist. 2	2.18 pixels	0.85 pixel	0.80 pixel



# Recap: Epipolar Transfer

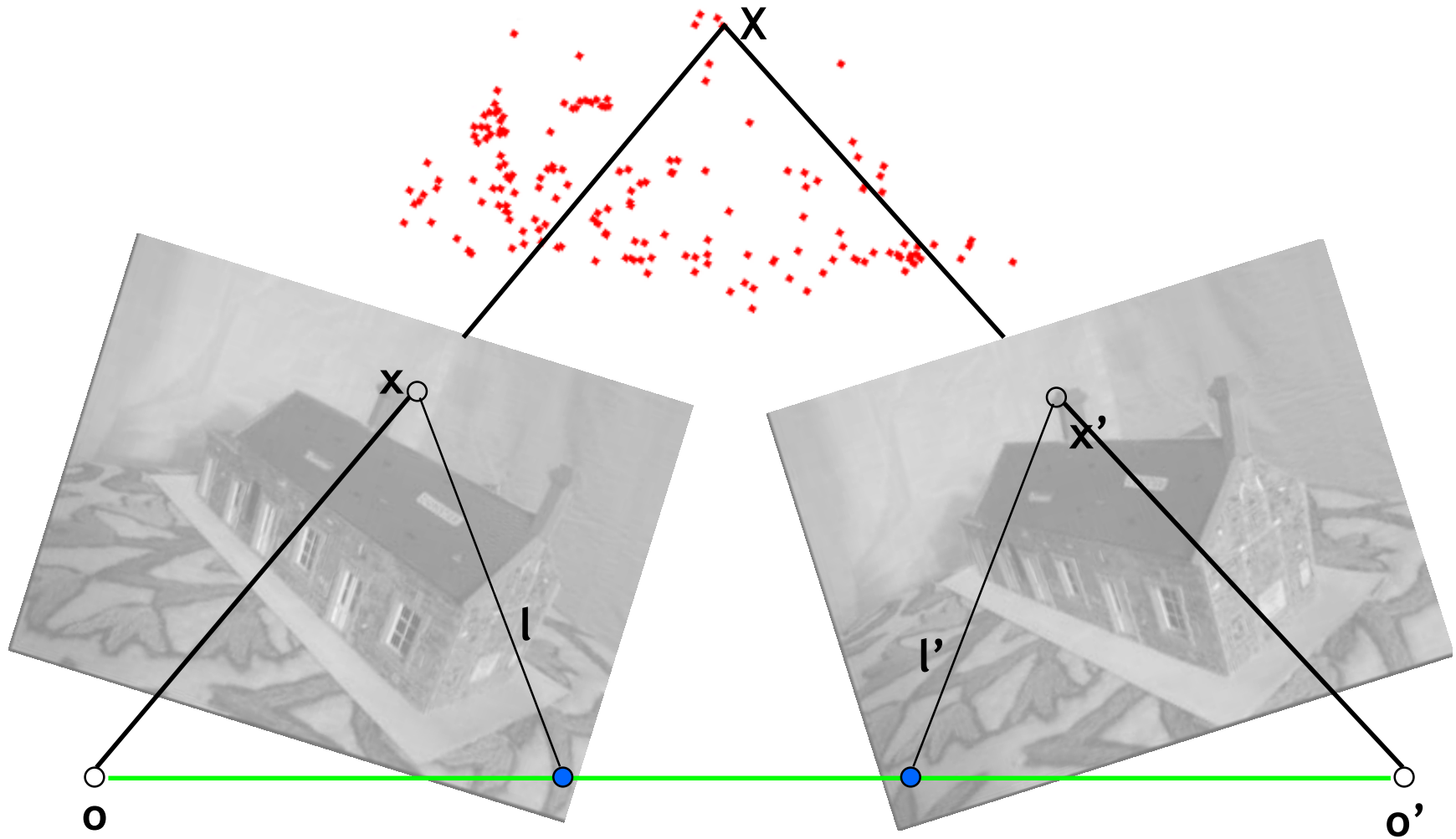
- Assume the epipolar geometry is known
- Given projections of the same point in two images, how can we compute the projection of that point in a third image?



$$l_{31} = F^T_{13} x_1$$

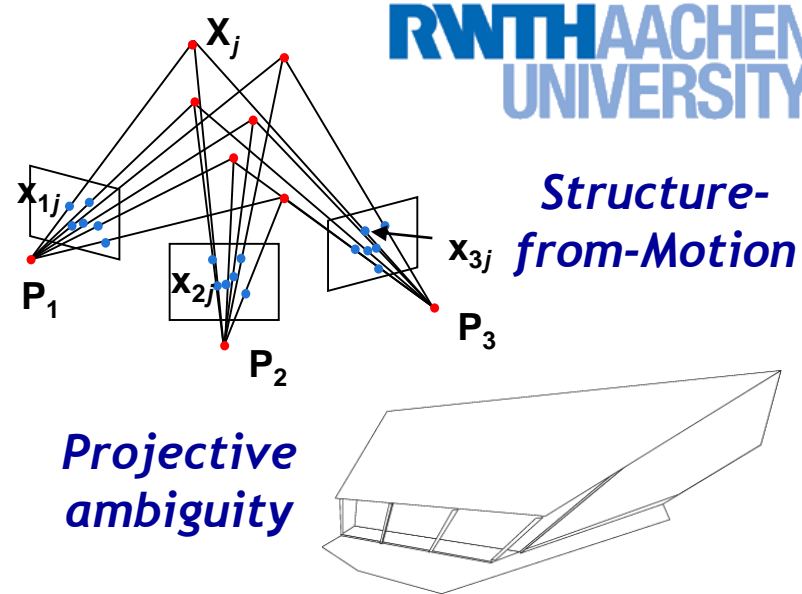
$$l_{32} = F^T_{23} x_2$$

# Applications: 3D Reconstruction



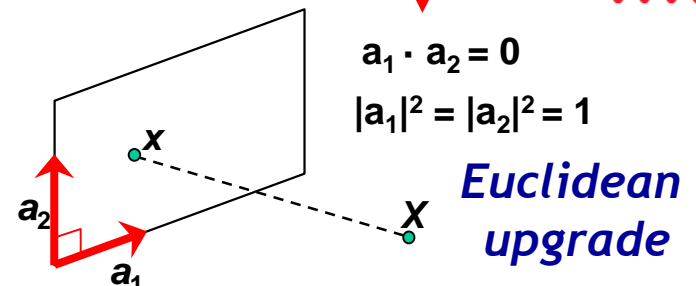
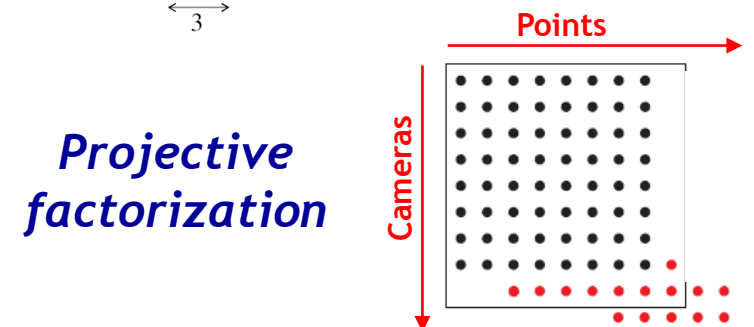
# Repetition

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
  - Epipolar Geometry and Stereo Basics
  - Camera Calibration & Uncalibrated Reconstruction
  - **Structure-from-Motion**
- Motion and Tracking

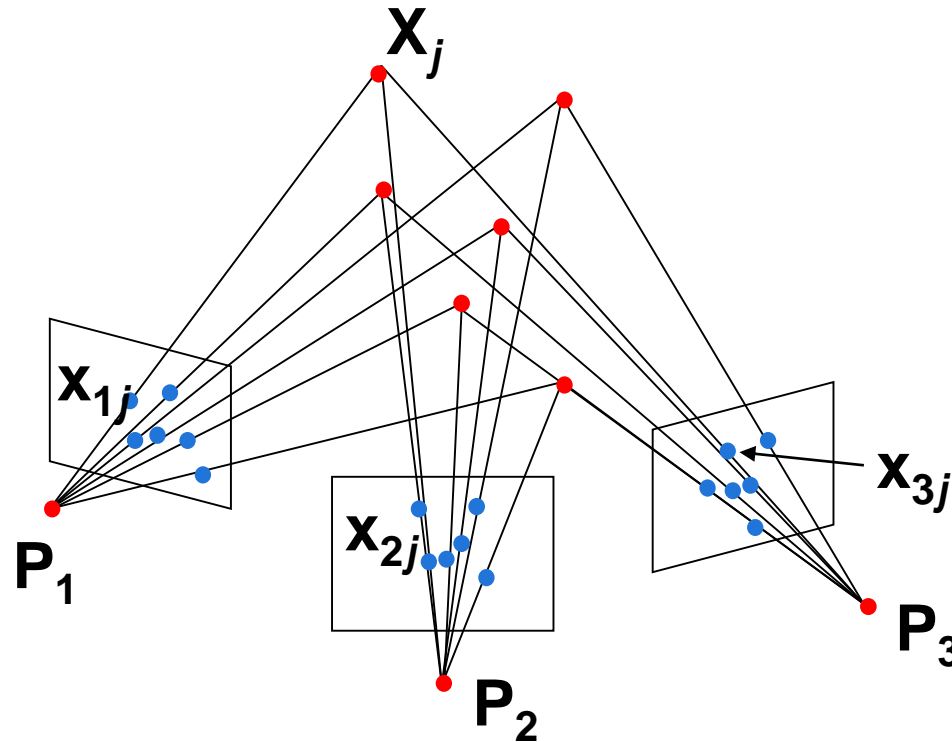


$$\begin{matrix} 2m \\ \downarrow \\ \text{D} \\ \uparrow \\ 2m \end{matrix} = \begin{matrix} \downarrow \\ \text{U}_3 \\ \uparrow \\ 3 \end{matrix} \times \begin{matrix} \leftarrow 3 \\ \text{W}_3 \\ \rightarrow 3 \end{matrix} \times \begin{matrix} \leftarrow n \\ \text{V}_3^T \\ \rightarrow 3 \end{matrix}$$

*Affine factorization*



# Recap: Structure from Motion



- Given:  $m$  images of  $n$  fixed 3D points

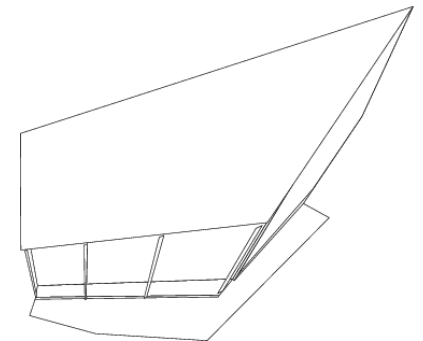
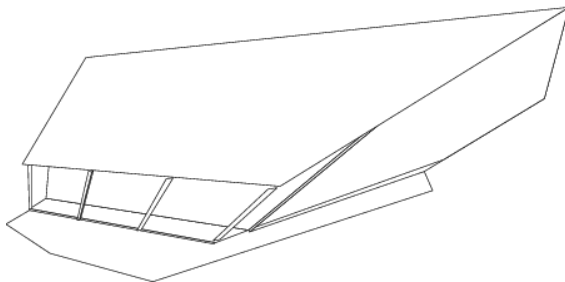
$$\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \dots, m, \quad j = 1, \dots, n$$

- Problem: estimate  $m$  projection matrices  $\mathbf{P}_i$  and  $n$  3D points  $\mathbf{X}_j$  from the  $mn$  correspondences  $\mathbf{x}_{ij}$

# Recap: Structure from Motion Ambiguity

- If we scale the entire scene by some factor  $k$  and, at the same time, scale the camera matrices by the factor of  $1/k$ , the projections of the scene points in the image remain exactly the same.
- More generally: if we transform the scene using a transformation  $Q$  and apply the inverse transformation to the camera matrices, then the images do not change

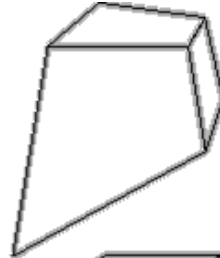
$$\mathbf{x} = \mathbf{P}\mathbf{X} = (\mathbf{P}\mathbf{Q}^{-1})\mathbf{Q}\mathbf{X}$$



# Recap: Hierarchy of 3D Transformations

Projective  
15dof

$$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$$



Preserves intersection  
and tangency

Affine  
12dof

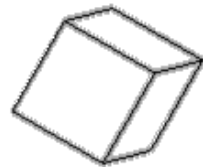
$$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$$



Preserves parallelism,  
volume ratios

Similarity  
7dof

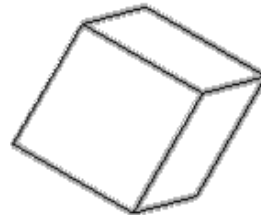
$$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$$



Preserves angles, ratios  
of length

Euclidean  
6dof

$$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$$



Preserves angles,  
lengths

- With no constraints on the camera calibration matrix or on the scene, we get a *projective* reconstruction.
- Need additional information to *upgrade* the reconstruction to affine, similarity, or Euclidean.

# Recap: Affine Structure from Motion

- Let's create a  $2m \times n$  data (measurement) matrix:

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{X}}_{11} & \hat{\mathbf{X}}_{12} & \cdots & \hat{\mathbf{X}}_{1n} \\ \hat{\mathbf{X}}_{21} & \hat{\mathbf{X}}_{22} & \cdots & \hat{\mathbf{X}}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{X}}_{m1} & \hat{\mathbf{X}}_{m2} & \cdots & \hat{\mathbf{X}}_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}$$

Points ( $3 \times n$ )

Cameras  
( $2m \times 3$ )

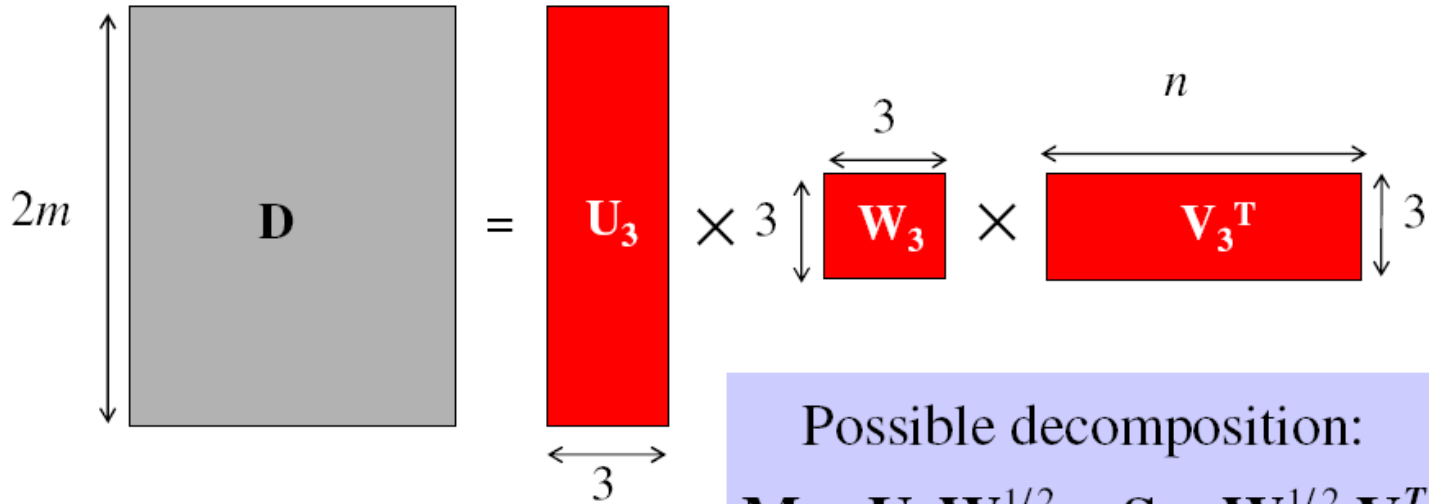
- The measurement matrix  $\mathbf{D} = \mathbf{MS}$  must have rank 3!

C. Tomasi and T. Kanade. [Shape and motion from image streams under orthography: A factorization method.](#) *IJCV*, 9(2):137-154, November 1992.

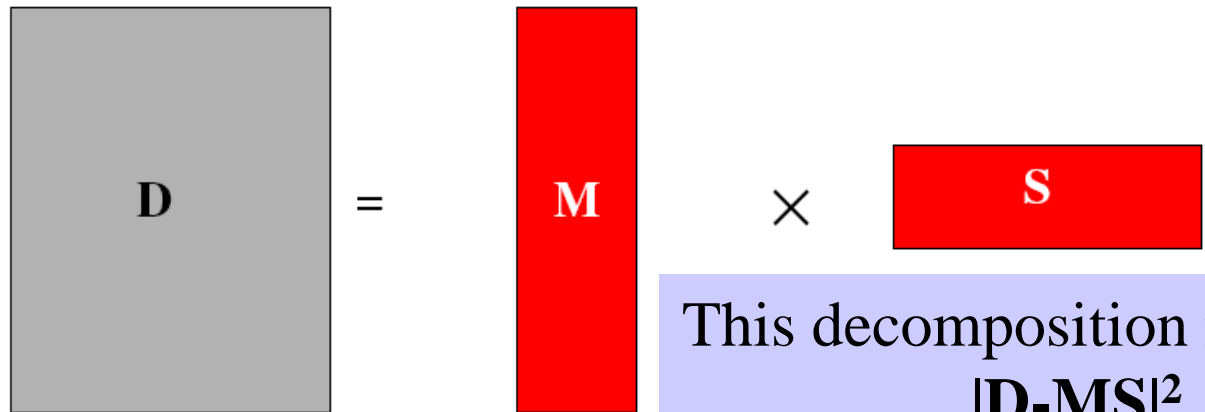


# Recap: Affine Factorization

- Obtaining a factorization from SVD:



Possible decomposition:  
 $M = U_3 W_3^{1/2}$      $S = W_3^{1/2} V_3^T$



This decomposition minimizes  $|D - MS|^2$

# Recap: Projective Factorization

$$\mathbf{D} = \begin{bmatrix} z_{11}\mathbf{X}_{11} & z_{12}\mathbf{X}_{12} & \cdots & z_{1n}\mathbf{X}_{1n} \\ z_{21}\mathbf{X}_{21} & z_{22}\mathbf{X}_{22} & \cdots & z_{2n}\mathbf{X}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{m1}\mathbf{X}_{m1} & z_{m2}\mathbf{X}_{m2} & \cdots & z_{mn}\mathbf{X}_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \vdots \\ \mathbf{P}_m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}$$

Points ( $4 \times n$ )

Cameras  
( $3m \times 4$ )

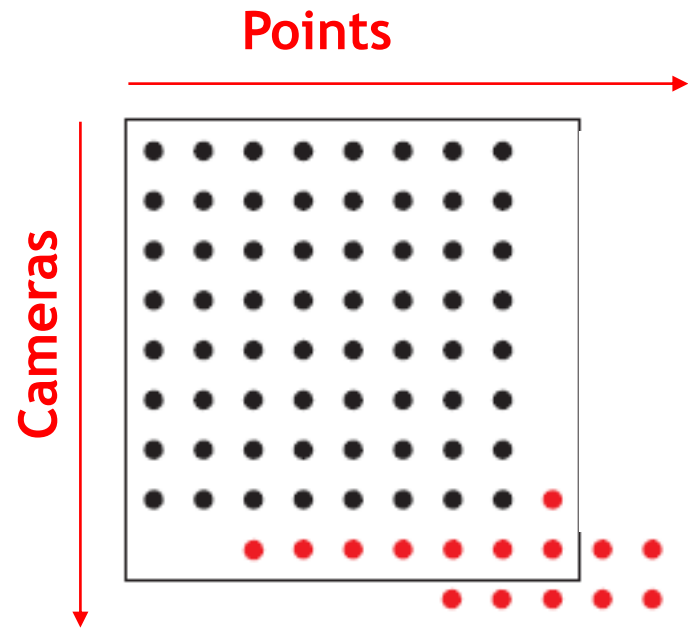
$\mathbf{D} = \mathbf{M}\mathbf{S}$  has rank 4

- If we knew the depths  $z$ , we could factorize  $\mathbf{D}$  to estimate  $\mathbf{M}$  and  $\mathbf{S}$ .
- If we knew  $\mathbf{M}$  and  $\mathbf{S}$ , we could solve for  $z$ .
- Solution: iterative approach (alternate between above two steps).



# Recap: Sequential Projective SfM

- Initialize motion from two images using fundamental matrix
- Initialize structure
- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image - *calibration*
  - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera - *triangulation*
- Refine structure and motion: *bundle adjustment*

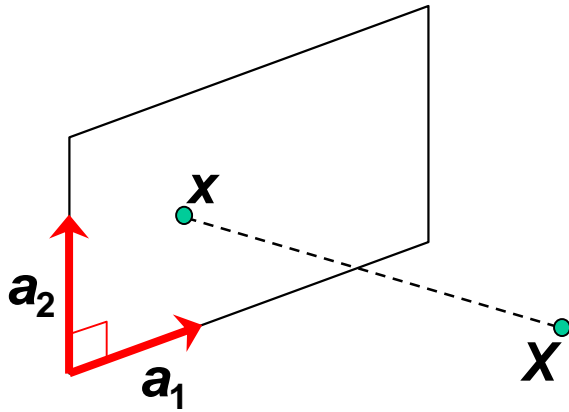


# Recap: Estimating the Euclidean Upgrade

- Goal: Estimate ambiguity matrix  $C$ 
  - Orthographic assumption:

$$D = M \times S$$

$M \rightarrow MC, S \rightarrow C^{-1}S$



- 1) Image axes are perpendicular  
 $a_1 \cdot a_2 = 0$
- 2) Scale is 1  
 $|a_1|^2 = |a_2|^2 = 1$

- This can be converted into a system of  $3m$  equations:

$$\begin{cases} \hat{a}_{i1} \cdot \hat{a}_{i2} = 0 \\ |\hat{a}_{i1}| = 1 \\ |\hat{a}_{i2}| = 1 \end{cases} \Leftrightarrow \begin{cases} a_{i1}^T C C^T a_{i2} = 0 \\ a_{i1}^T C C^T a_{i1} = 1 \\ a_{i2}^T C C^T a_{i2} = 1 \end{cases}, \quad i = 1, \dots, m$$

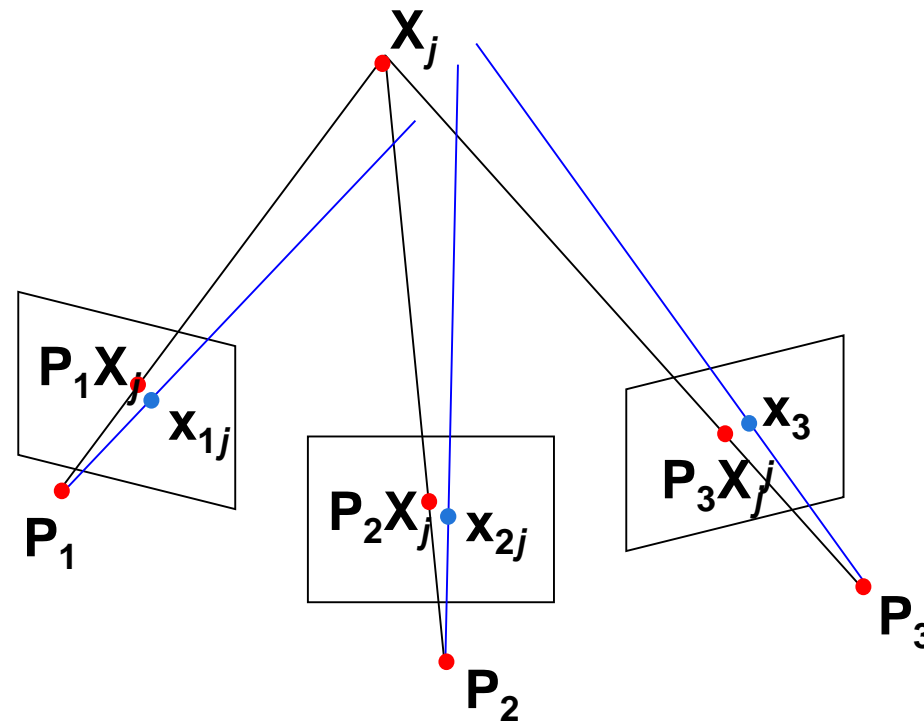
with  $L = C C^T$   
this translates to

$$A_i L A_i^T = I$$

# Recap: Bundle Adjustment

- Non-linear method for refining structure and motion
- Minimizing mean-square reprojection error

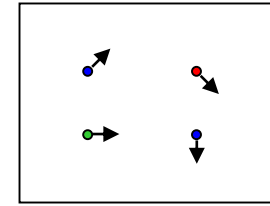
$$E(\mathbf{P}, \mathbf{X}) = \sum_{i=1}^m \sum_{j=1}^n D(\mathbf{x}_{ij}, \mathbf{P}_i \mathbf{X}_j)^2$$



B. Leibe

# Repetition

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
- Motion and Tracking
  - Motion and Optical Flow

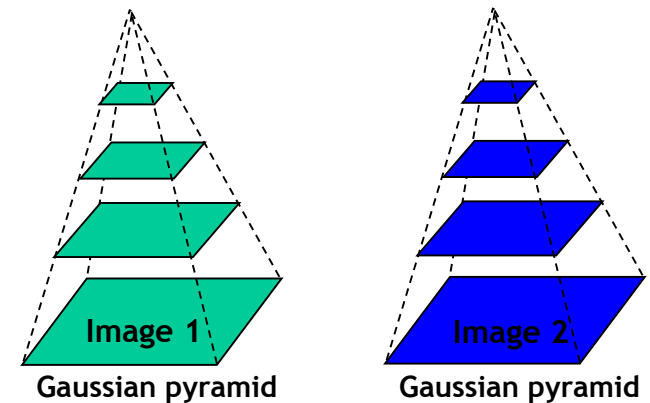


*Motion field*

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

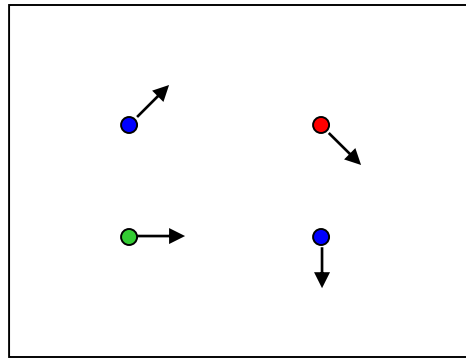
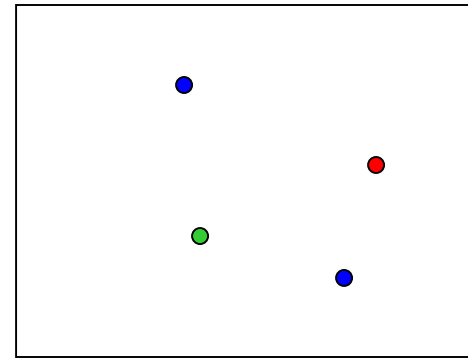
$A^T A$    $A^T b$

*Lucas-Kanade optical flow*



*Coarse-to-fine estimation*

# Recap: Estimating Optical Flow

 $I(x, y, t-1)$  $I(x, y, t)$ 

- Given two subsequent frames, estimate the apparent motion field  $u(x, y)$  and  $v(x, y)$  between them.
- Key assumptions
  - **Brightness constancy:** projection of the same point looks the same in every frame.
  - **Small motion:** points do not move very far.
  - **Spatial coherence:** points move like their neighbors.



# Recap: Lucas-Kanade Optical Flow

- Use all pixels in a  $K \times K$  window to get more equations.
- Least squares problem:

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix} \quad \begin{matrix} A & d = b \\ 25 \times 2 & 2 \times 1 & 25 \times 1 \end{matrix}$$

- Minimum least squares solution given by solution of

$$\begin{matrix} (A^T A) & d = A^T b \\ 2 \times 2 & 2 \times 1 & 2 \times 1 \end{matrix}$$

Recall the  
Harris detector!

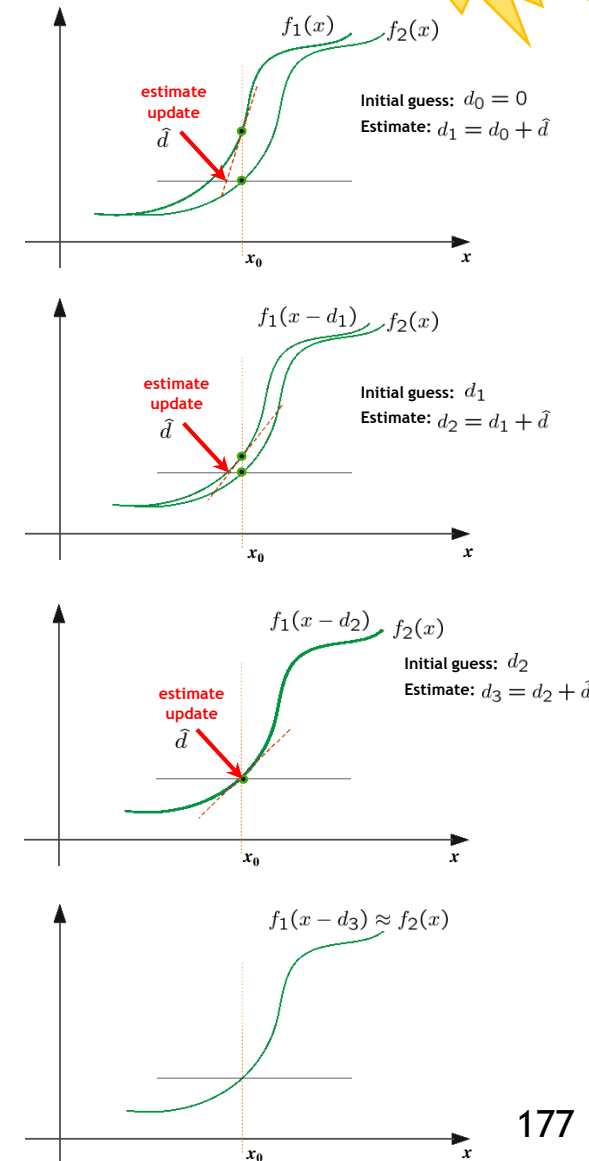
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$$\begin{matrix} A^T A & A^T b \end{matrix}$$

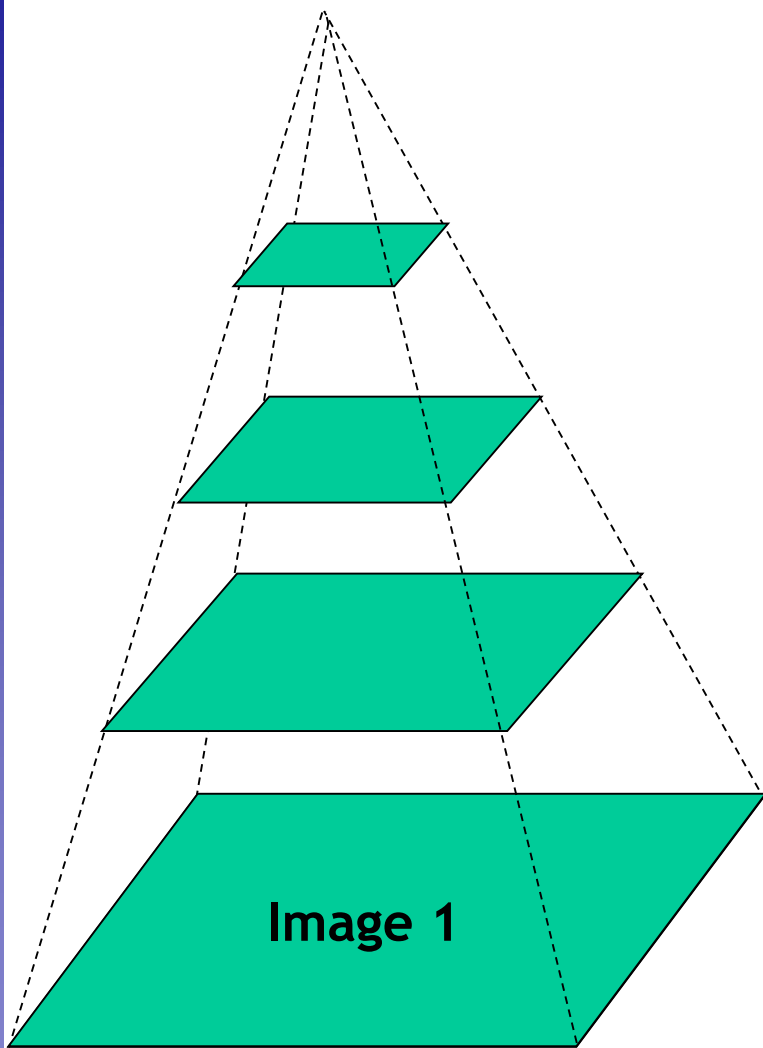
see  
Exercise 6.4!

# Recap: Iterative Refinement

- Estimate velocity at each pixel using one iteration of LK estimation.
- Warp one image toward the other using the estimated flow field.
- Refine estimate by repeating the process.
- Iterative procedure
  - Results in subpixel accurate localization.
  - Converges for small displacements.



# Recap: Coarse-to-fine Estimation



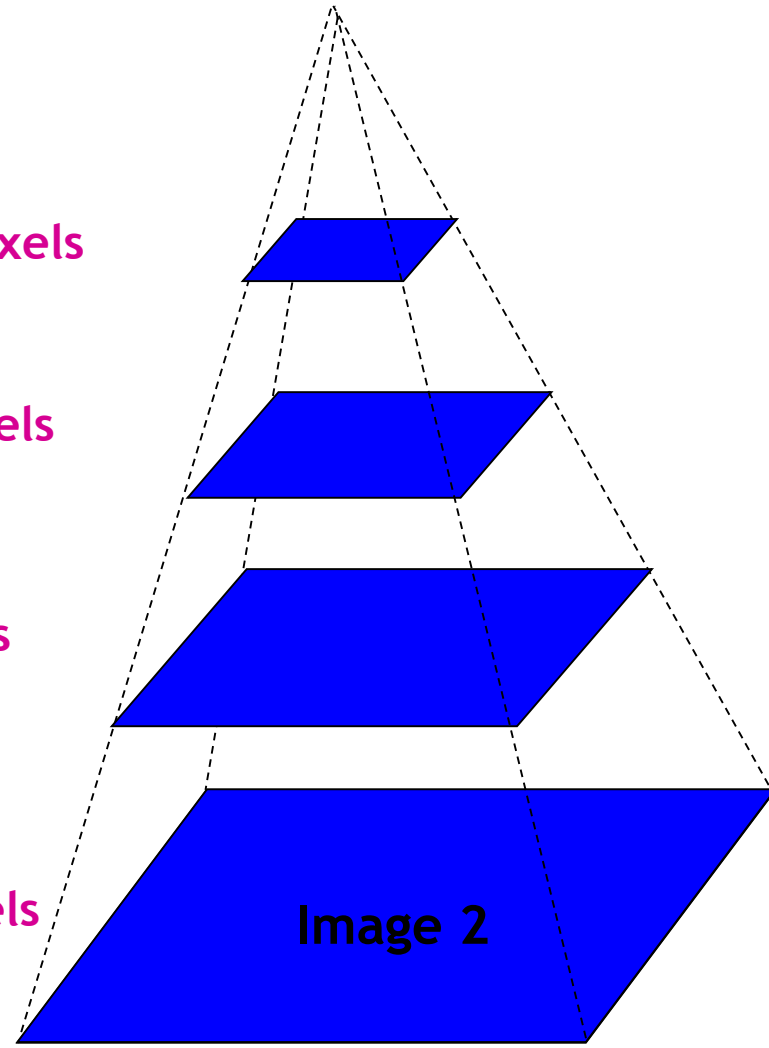
Gaussian pyramid of image 1

$u=1.25$  pixels

$u=2.5$  pixels

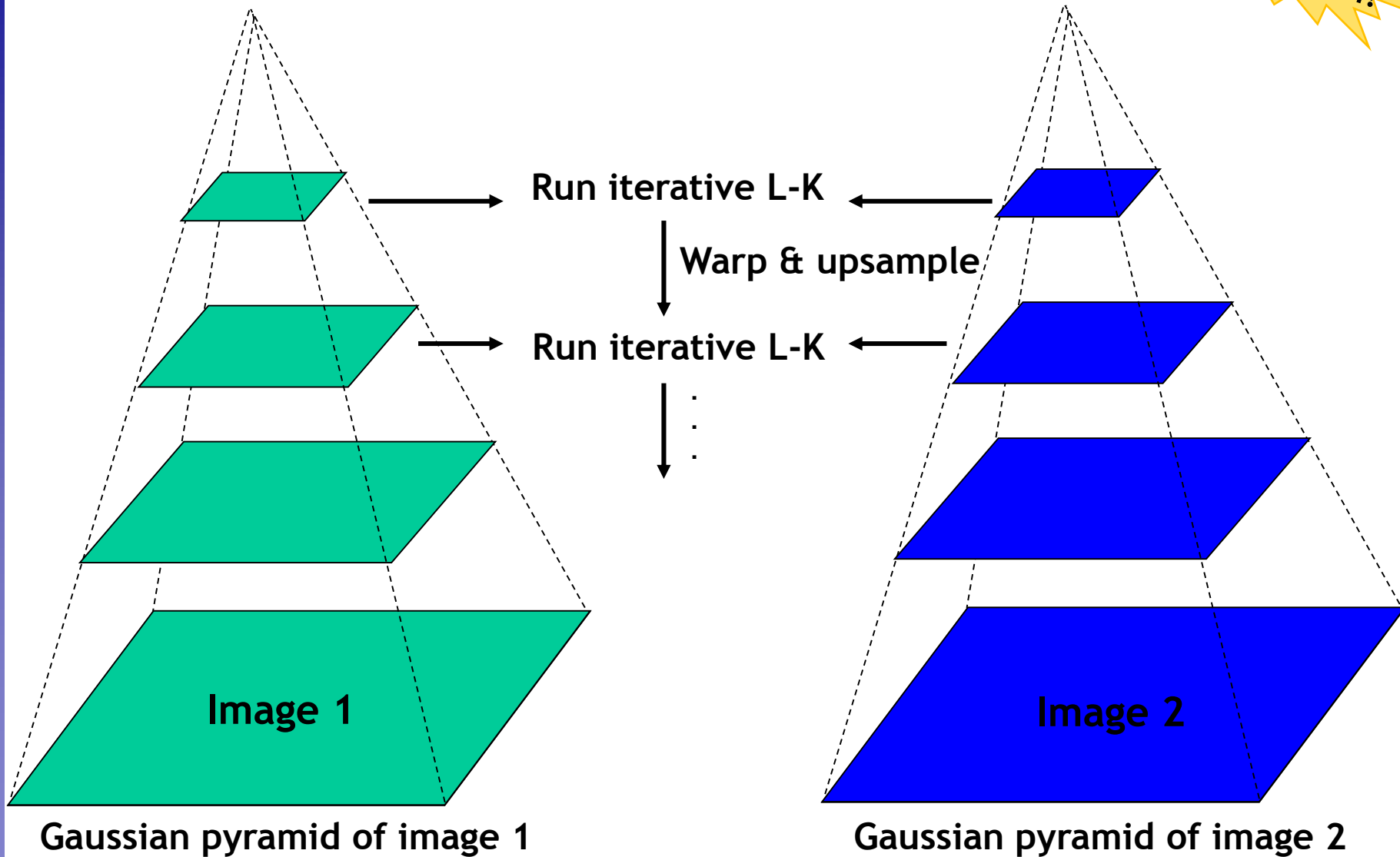
$u=5$  pixels

$u=10$  pixels



Gaussian pyramid of image 2

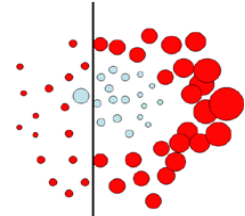
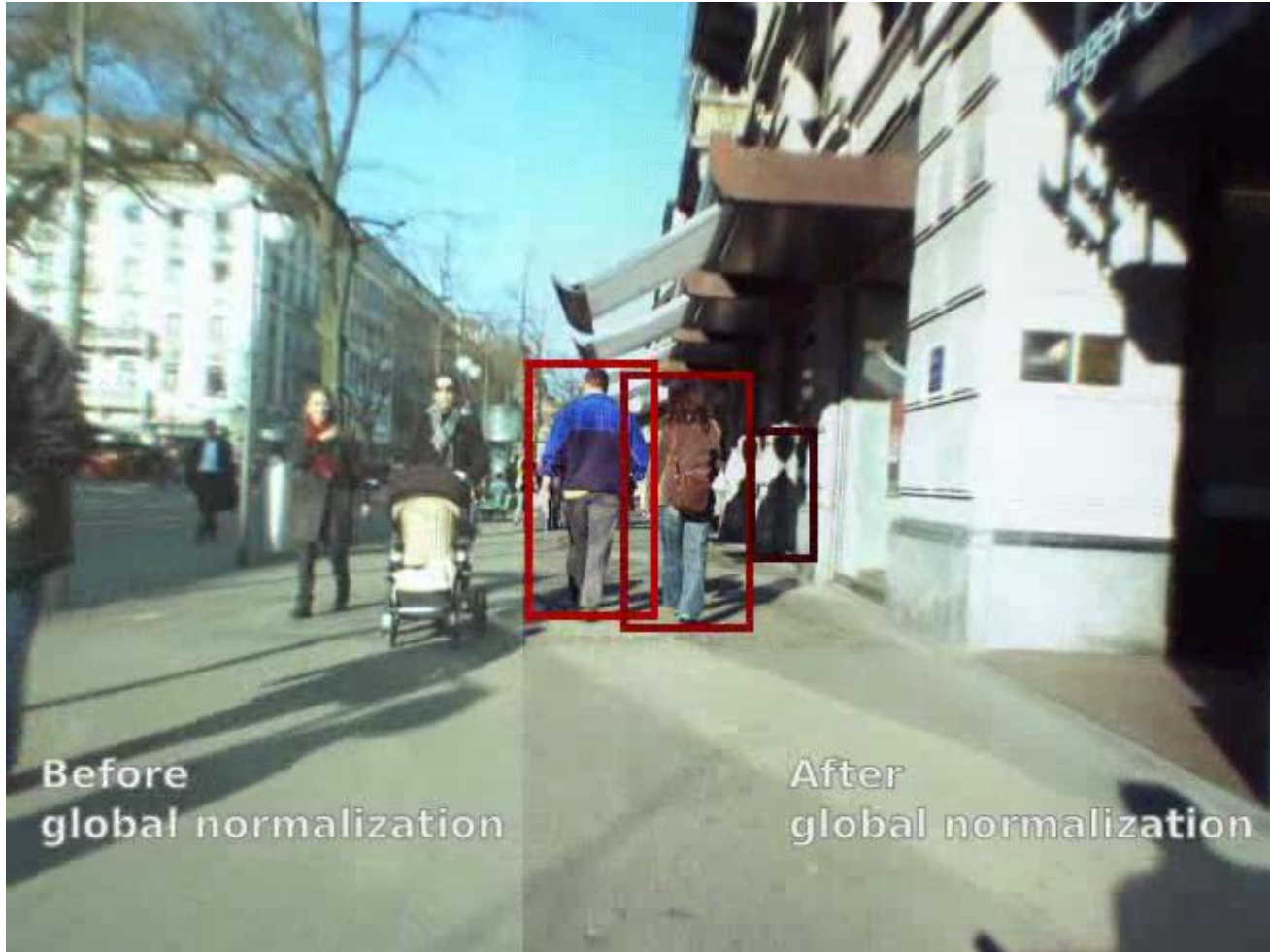
# Recap: Coarse-to-fine Estimation



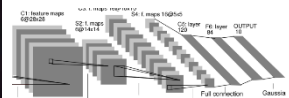
# Any Questions?

*So what can you do with all of this?*

# Application: Object Detection

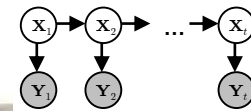
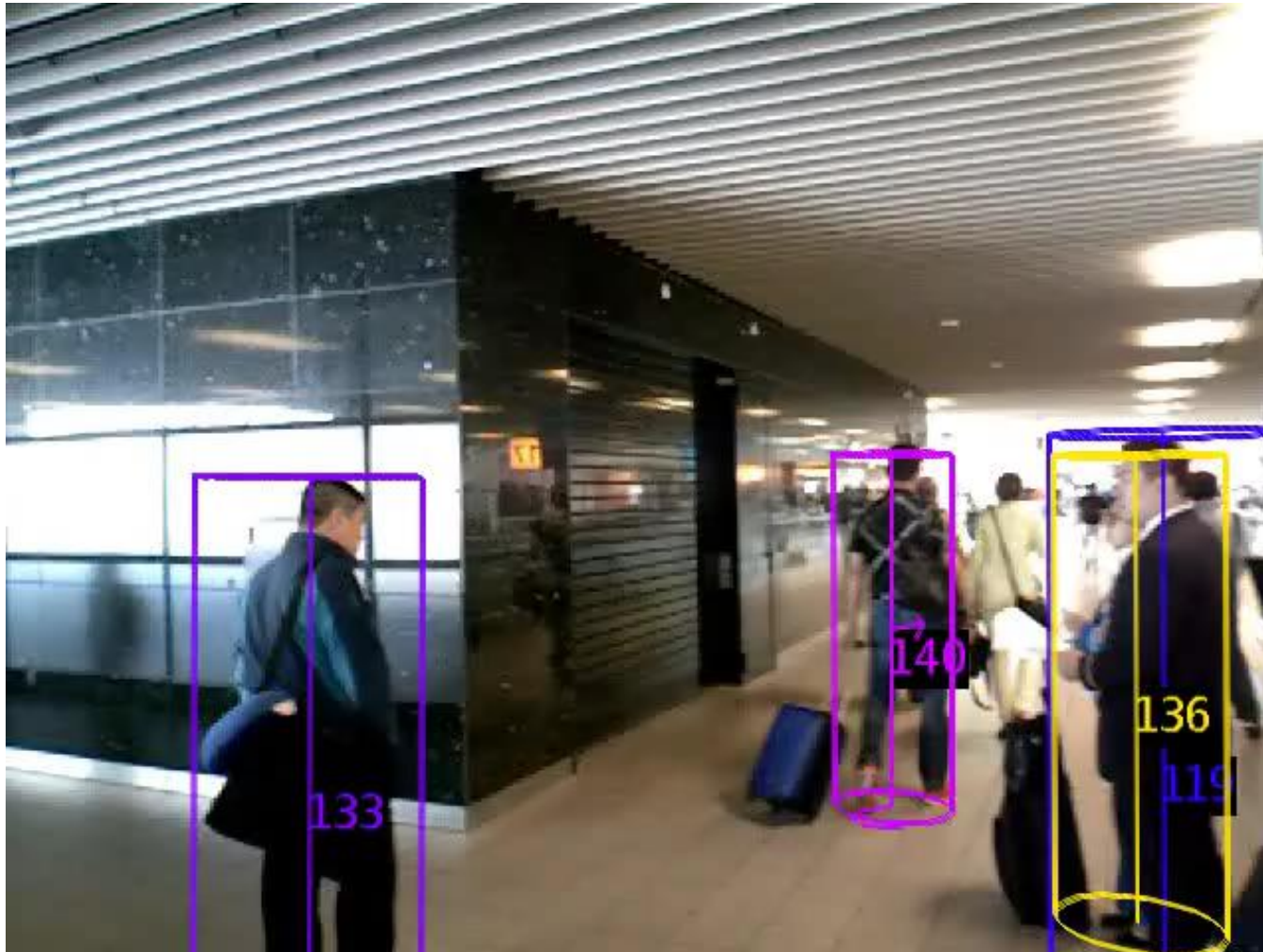


AdaBoost

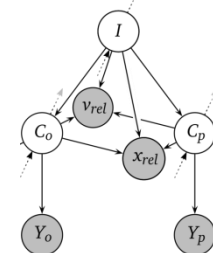


CNNs

# Application: Multi-Object Tracking



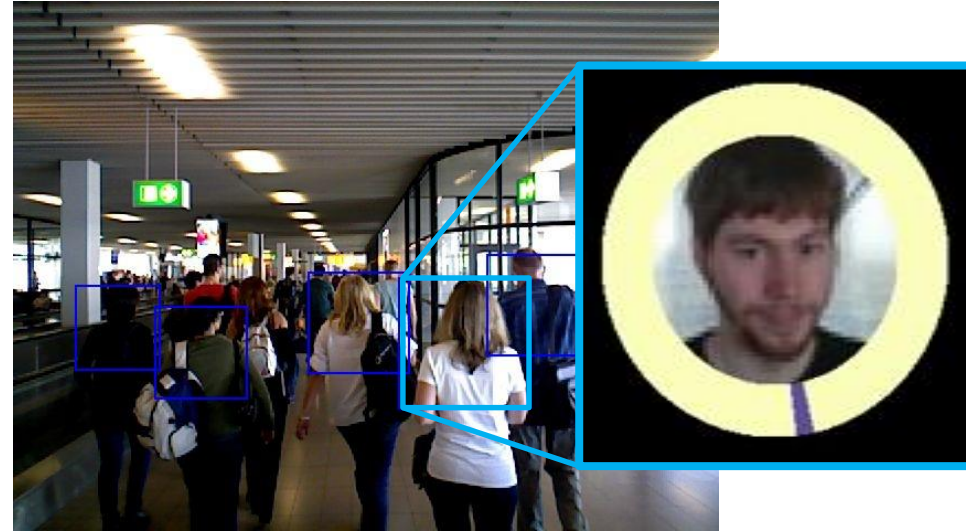
Kalman Filters



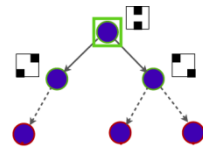
Dynamic Bayesian Networks



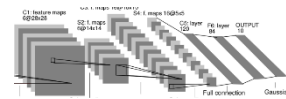
# Application: Detailed Person Analysis



- **Additional analysis modules for tracked people**
  - Articulated body pose estimation
  - Head orientation estimation
  - Human attribute recognition
  - ...



Random  
Forests



CNNs

# Semantic Segmentation



[Pohlen, Hermans, Mathias, Leibe, arXiv 2016]

- **More recent results**
  - Based on an extension of ResNets

# Automotive Applications



# Any More Questions?

*Good luck for the exam!*