Computer Vision for Robotics

Undergraduate course (Spring 2020)

Nguyen Do Van, PhD

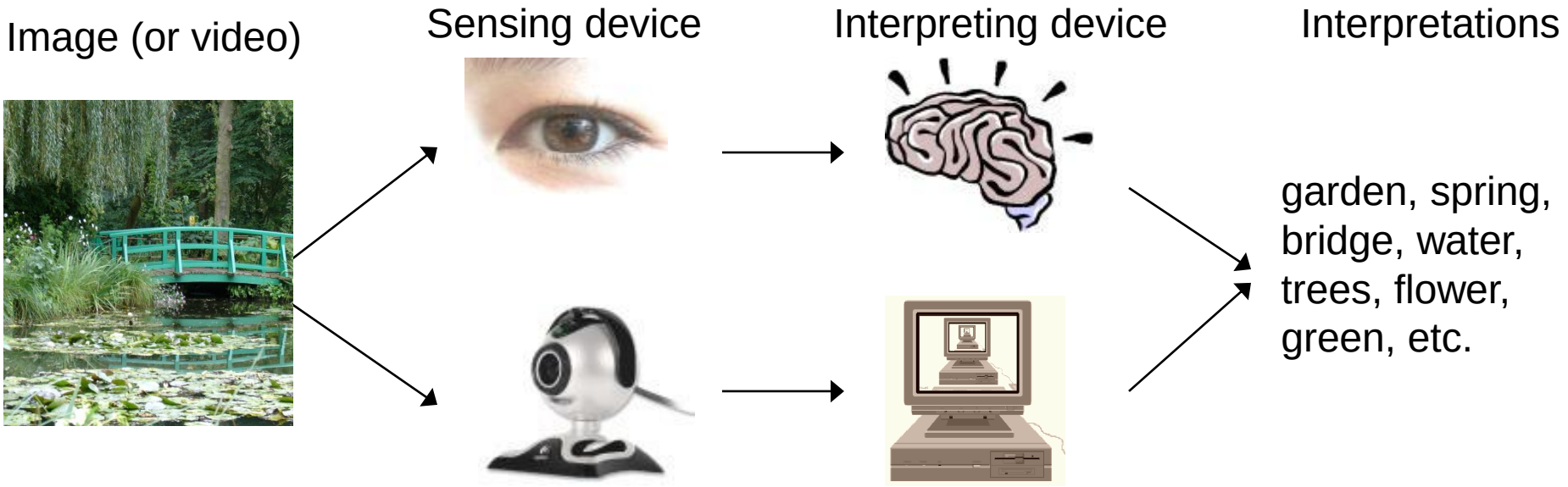
Assignments and Examinations

- First assignment:
  - Simple programming
  - Times: After Lecture 8
- Middle term examination:
  - Written form (Or online)
  - Content: Artificial Intelligence in Robots
  - Time: After Lecture 8

What are “Autonomous Robots”?

- Mobile machines with power, sensing, and computing on-board
- Environments
  - Land (on and under)
  - Water (ditto)
  - Air
  - Space
  - ???

What is (computer) vision?



Computer vision vs human vision

What we see

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

What a computer sees

What Skills Do Robots Need?

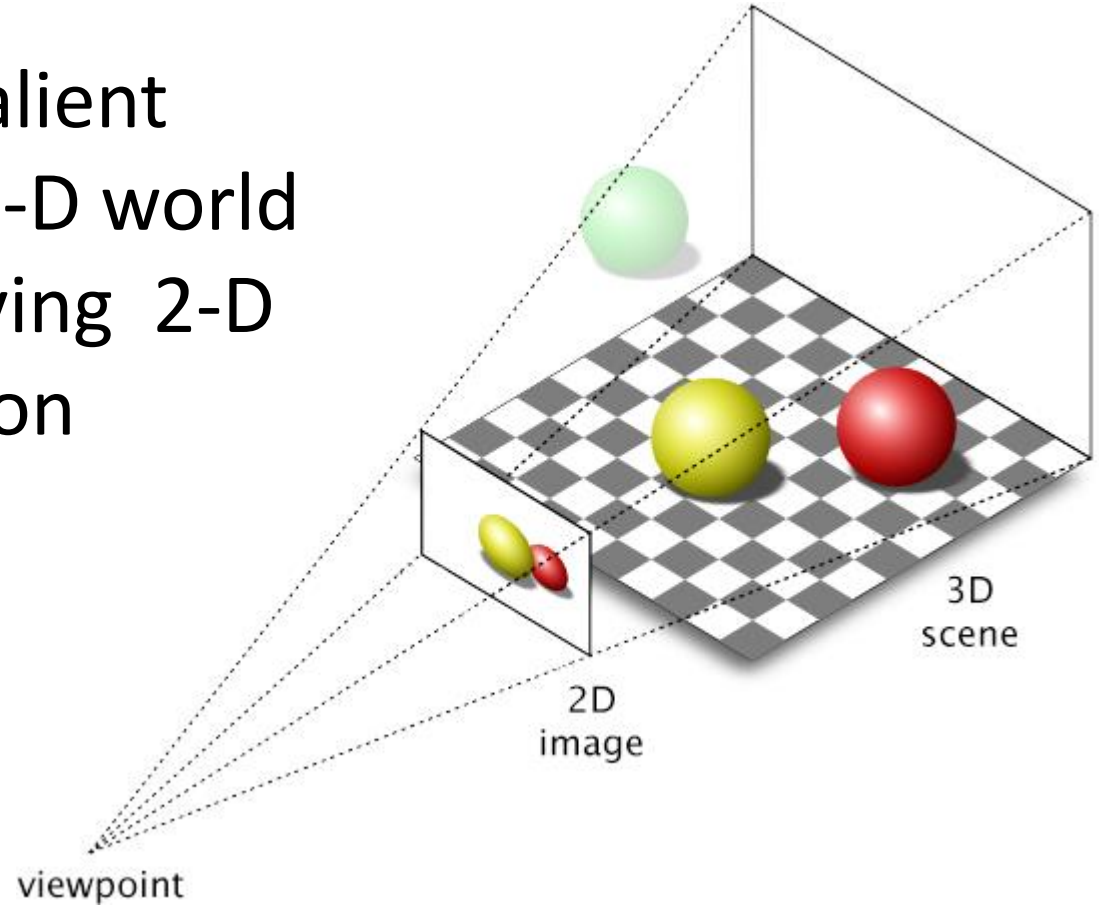
- **Identification**: *What/who is that?*
  - Object detection, recognition
- **Movement**: *How do I move safely?*
  - Obstacle avoidance, homing
- **Manipulation**: *How do I change that?*
  - Interacting with objects/environment
- **Navigation**: *Where am I?*
  - Mapping, localization

Why Vision?

- Pluses
  - Rich stream of complex information about the environment
  - Primary human sense
  - Good cameras are fairly cheap
- Minuses
  - Line of sight only
  - Passive $\rightarrow$ Dependent on ambient illumination

The Vision Problem

How to infer salient properties of 3-D world from time-varying 2-D image projection



Computer Vision Outline

- Image formation
- Image processing
- Motion & Estimation
- Classification & Clustering

Outline: Image Formation

- 3-D geometry
- Physics of light
- Camera properties
  - Focal length
  - Distortion
- Sampling issues
  - Spatial
  - Temporal

Outline: Image Processing

- Filtering
  - Edge
  - Color
  - Shape
  - Texture
- Feature detection
- Pattern comparison

Outline: Motion & Estimation

- Computing temporal image change
  - Magnitude
  - Direction
- Fitting parameters to data
  - Static
  - Dynamic (e.g., tracking)
- Applications
  - Motion Compensation
  - Structure from Motion

Outline: Classification & Clustering

- Categorization
  - Assignment to known groups
- Clustering
  - Inference of group existence from data
  - Special case: Segmentation

Visual Skills: Identification

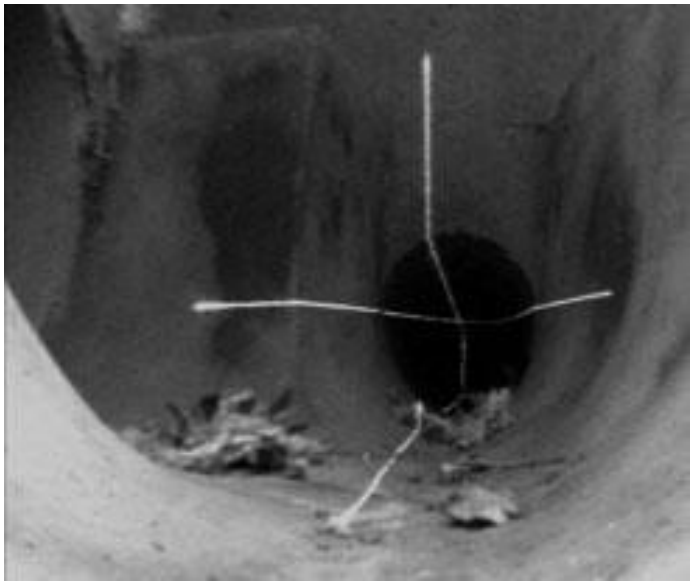
- Recognizing face/body/structure: *Who/what do I see?*
  - Use shape, color, pattern, other static attributes to distinguish from background, other hypotheses
- Gesture/activity: *What is it doing?*
  - From low-level motion detection & tracking to categorizing high-level temporal patterns
- Feedback between static and dynamic

Minerva Face Detection

Finding people to interact with

Visual Skills: Movement

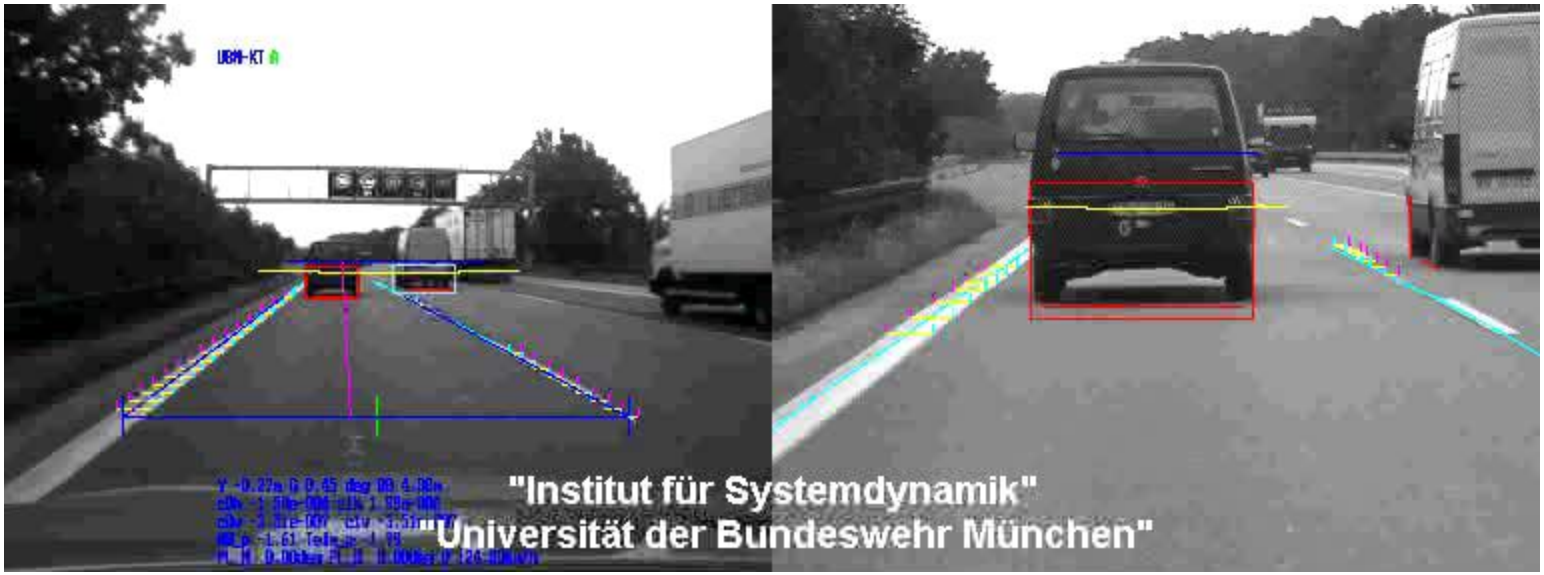
- Steering, foot placement or landing spot for entire vehicle



MAKRO sewer shape pattern

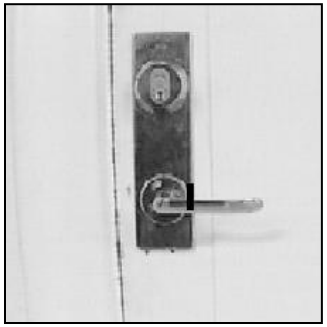
Demeter region boundary detection

UBM Lane & vehicle tracking (with radar)



Visual Skills: Manipulation

- Moving other things
  - Grasping: Door opener (KTH)
  - Pushing, digging, cranes

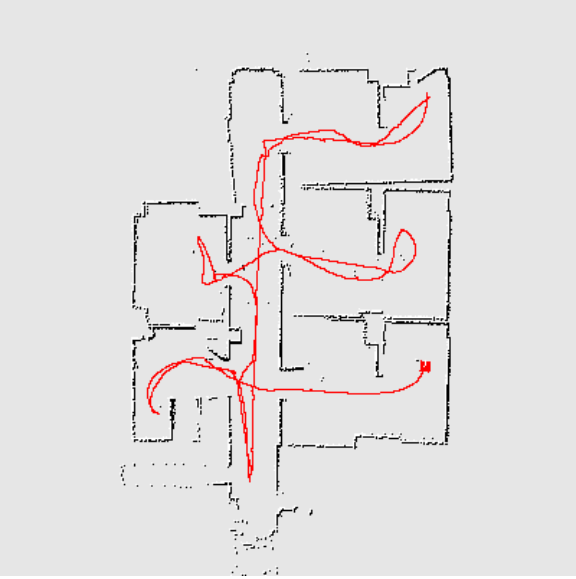


KTH robot & typical handle

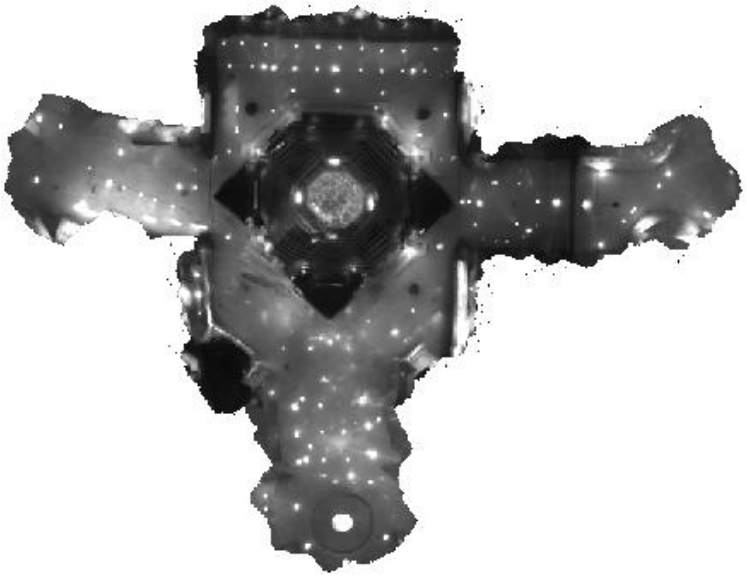
Clodbusters push a box cooperatively

Visual Skills: Navigation

- Building a map
- Localization/place recognition
  - *Where are you in the map?*



Laser-based wall map (CMU)

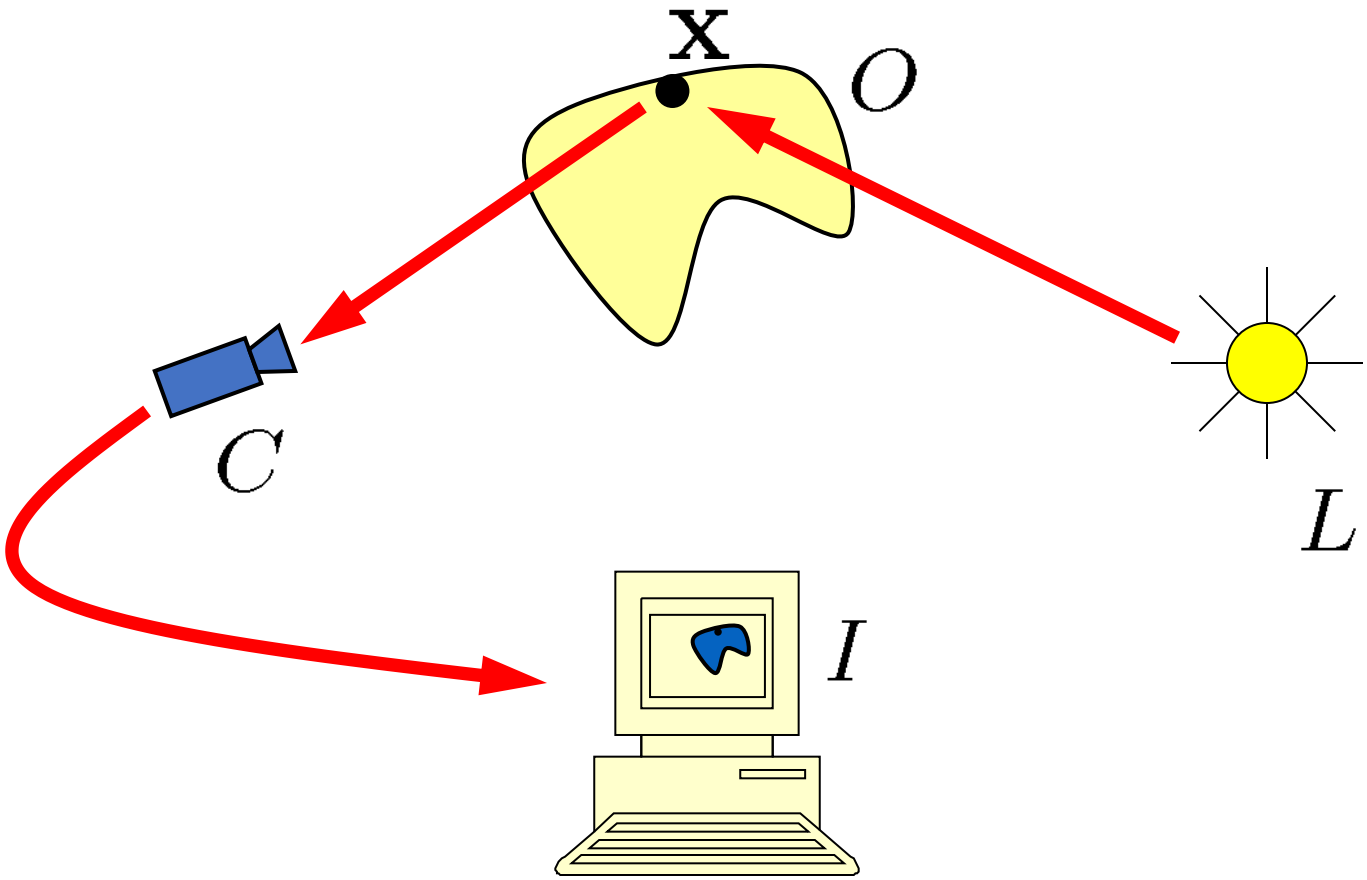


Minerva's ceiling map

Outline: Image Formation

- Geometry
  - Coordinate systems, transformations
  - Perspective projection
  - Lenses
- Radiometry
  - Light emission, interaction with surfaces
- Analog $\rightarrow$ Digital
  - Spatial sampling
  - Dynamic range
  - Temporal integration

The Image Formation Pipeline



Coordinate System Conventions

- $\mathbf{i}, \mathbf{j}, \mathbf{k}$ unit vectors along positive $\mathbf{k} = \mathbf{i} \times \mathbf{j}$ axes, X, Y, Z respectively;
- Right- vs. left-handed coordinates
- Local coordinate systems: camera, world, etc.

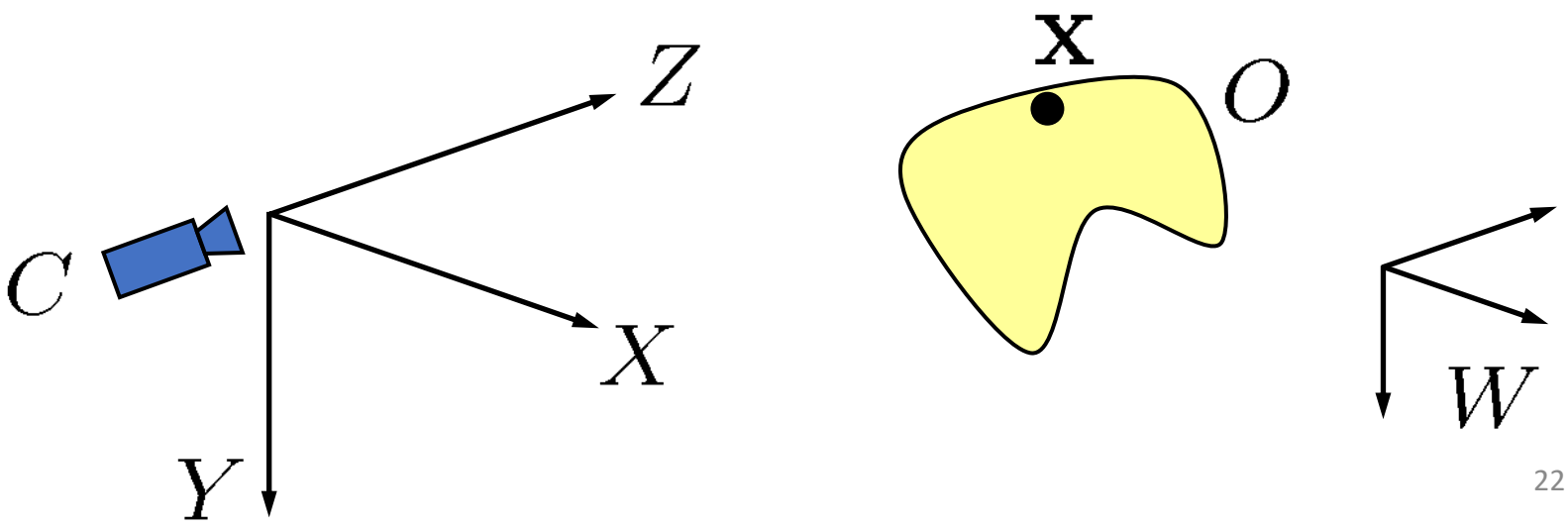


Image Processing Outline

- Images
- Binary operators
- Filtering
  - Smoothing
  - Edge, corner detection
- Modeling, matching
- Scale space

Images

- An image is a matrix of pixels $\mathbf{I}(x, y)$
- Resolution
  - Digital cameras: 1600 X 1200 at a minimum
  - Video cameras: ~640 X 480
- Grayscale: generally 8 bits per pixel $\rightarrow$ Intensities in range [0…255]
- RGB color: 3 8-bit color planes $\mathbf{I}_R, \mathbf{I}_G, \mathbf{I}_B$

Image Conversion

- RGB $\rightarrow$ Grayscale: Mean color value, or weight by perceptual importance

- Grayscale $\rightarrow$ Binary: Choose threshold based on histogram of image intensities

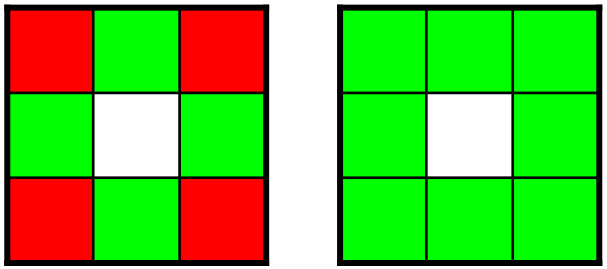
Color Representation

- RGB, HSV (hue, saturation, value), YUV, etc.
- Luminance: Perceived intensity
- Chrominance: Perceived color
  - HS(V), (Y)UV, etc.
  - Normalized RGB removes some illumination dependence:

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}$$

Binary Operations

- Dilation, erosion
  - Dilation: All 0's next to a 1 $\rightarrow$ 1 (Enlarge foreground)
  - Erosion: All 1's next to a 0 $\rightarrow$ 0 (Enlarge background)
- Connected components
  - Uniquely label each *n*-connected region in binary image
  - 4- and 8-connectedness



- Moments: Region statistics
  - Zeroth-order: Size
  - First-order: Position (centroid)
  - Second-order: Orientation

Image Transformations

- Geometric: Compute new pixel locations
  - Rotate
  - Scale
  - Undistort (e.g., radial distortion from lens)

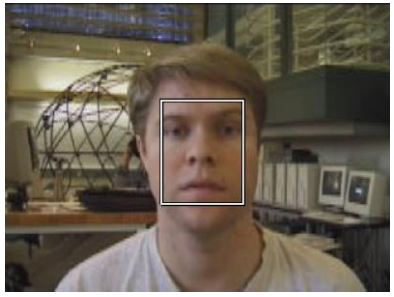
$$T(x, y) \rightarrow (x', y')$$

- Photometric: How to compute new pixel values when $T^{-1}(x', y')$ non-integral
  - Nearest neighbor: Value of closest pixel
  - Bilinear interpolation (2 x 2 neighborhood)
  - Bicubic interpolation (4 x 4)

Image Comparison: SSD

- Given a template image $\mathbf{I}_T$ and an image $\mathbf{I}$, how to quantify the similarity between them for a given alignment?
- Sum of squared differences (SSD)

$$\sum_{x,y} [\mathbf{I}_T(x,y) - \mathbf{I}(x,y)]^2$$




Cross-Correlation for Template Matching

- Note that SSD formula can be written:

$$\sum_{x,y} \mathbf{I}_T^2(x,y) + \mathbf{I}^2(x,y) - 2\mathbf{I}_T(x,y)\mathbf{I}(x,y)$$

- First two terms fixed $\rightarrow$ last term measures mismatch—the *cross-correlation:*

$$\sum_{x,y} \mathbf{I}_T(x,y) \cdot \mathbf{I}(x,y)$$

- In practice, normalize by image $\mathbf{I}$ magnitude when shifting template to search for matches

Filtering

- Idea: Analyze neighborhood around some point in image f with filter function h; put result in new image at corresponding location g
- System properties
  - Shift invariance: Same inputs give same outputs, regardless of location
  - Superposition: Output on sum of images = Sum of outputs on separate images
  - Scaling: Output on scaled image = Scaled output on image
- Linear shift invariance $\rightarrow$ **Convolution**

Discrete Filtering

- Linear filter: Weighted sum of pixels over rectangular neighborhood—*kernel* defines weights
- Think of kernel as template being matched by correlation
- Convolution: Correlation with kernel rotated 180°
- Dealing with image edges
  - Zero-padding
  - Border replication

1	-1	-1
1	2	-1
1	1	1

Filtering Example 1:

$$\mathbf{I}' = \mathbf{K} * \mathbf{I}$$

$\mathbf{K}$

1	-1	-1
1	2	-1
1	1	1

Rotate

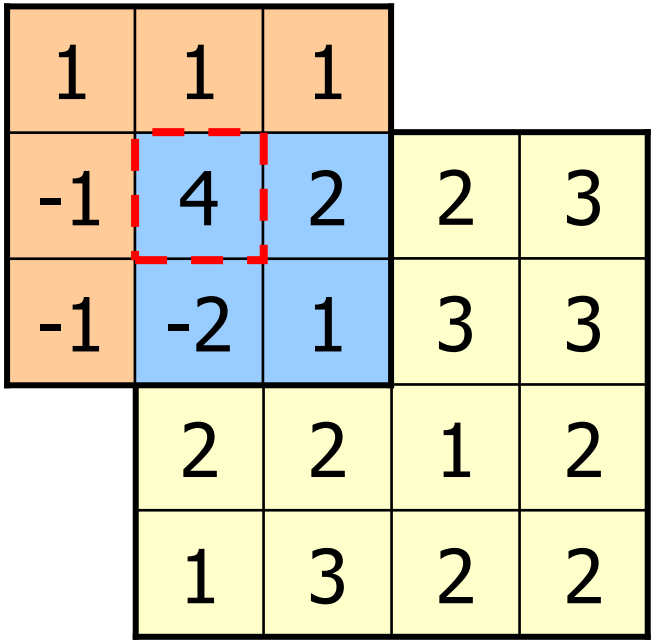
1	1	1
-1	2	1
-1	-1	1

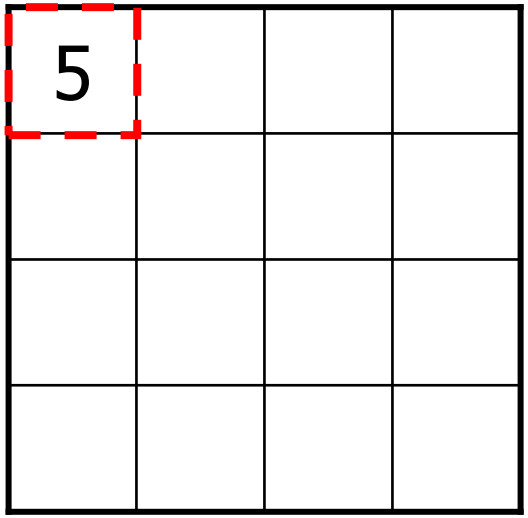
$\mathbf{I}$

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



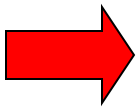


I I'

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

1	1	1	
-2	4	2	3
-2	-1	3	3
2	2	1	2
1	3	2	2



5	4		

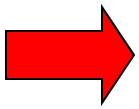
I

I'

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

	1	1	1
2	-2	4	3
2	-1	-3	3
2	2	1	2
1	3	2	2



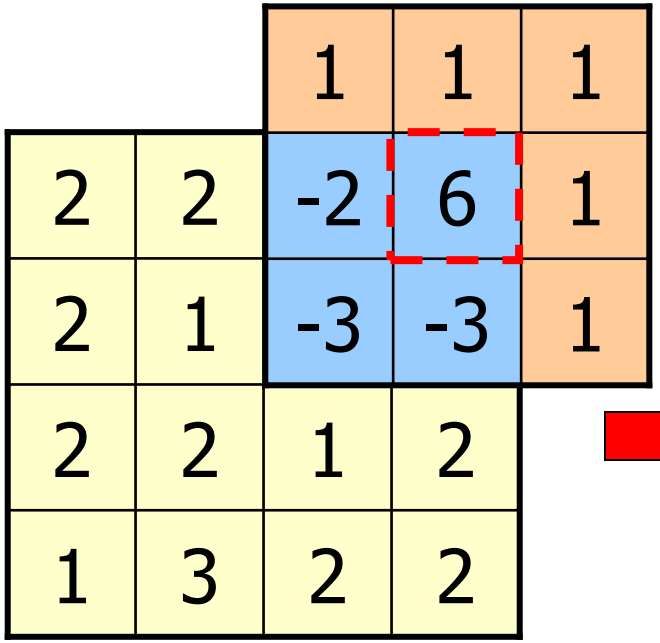
5	4	4	

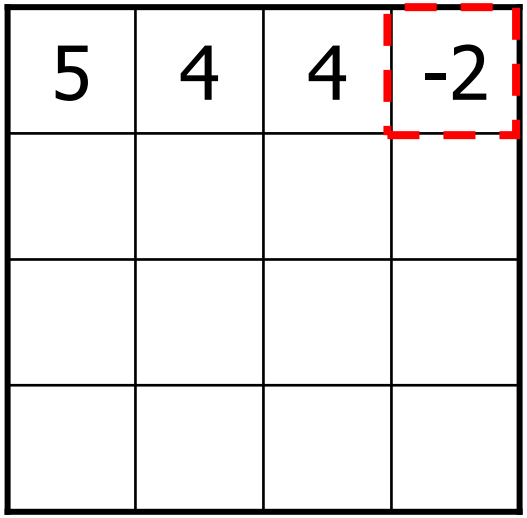
I

I'

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



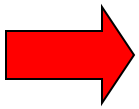


I I'

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

1	2	2	2	3
-1	4	1	3	3
-1	-2	2	1	2
	1	3	2	2



5	4	4	-2
9			

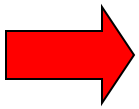
I

I′

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

2	2	2	3
-2	2	3	3
-2	-2	1	2
1	3	2	2



5	4	4	-2
9	6		

I

I'

Final Result

5	4	4	-2
9	6	14	5
11	7	6	5
9	12	8	5

$\mathbf{I}'$

Smoothing (Low-Pass) Filters

- Replace each pixel with average of neighbors
- Benefits: Suppress noise, aliasing
- Disadvantage: Sharp features blurred
- Types
  - Mean filter (box)
  - Median (nonlinear)
  - Gaussian

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

3 x 3 box filter

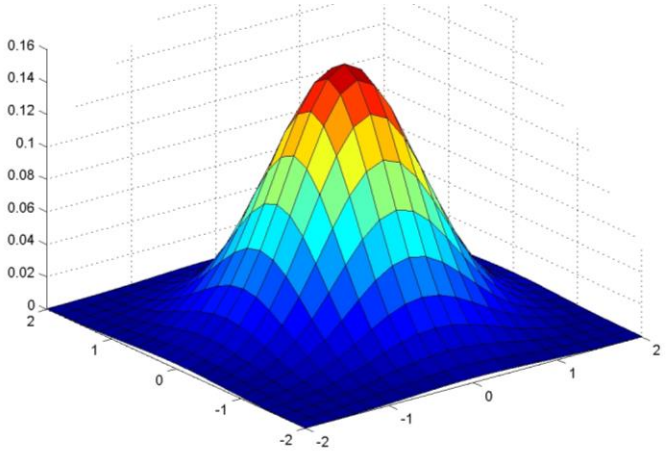
Box Filter: Smoothing

Original image

7 x 7 kernel

Gaussian Kernel

- Idea: Weight contributions of neighboring pixels by nearness



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

Gaussian: Smoothing

Original image

$\sigma = 1$

7 x 7 kernel

$\sigma = 3$

Gradient

- Think of image intensities as a function $\mathbf{I}(x, y)$. Gradient of image is a vector field as for a normal 2-D height function:

$$\nabla \mathbf{I} = (\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y})^T = (\mathbf{I}_x, \mathbf{I}_y)^T$$

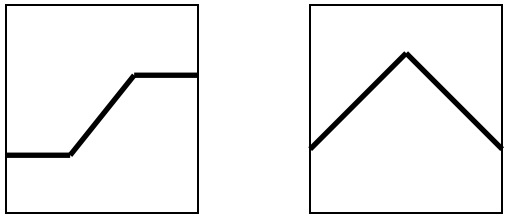
- **Edge**: Place where gradient magnitude is high; orthogonal to gradient direction

Edge Causes

- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)

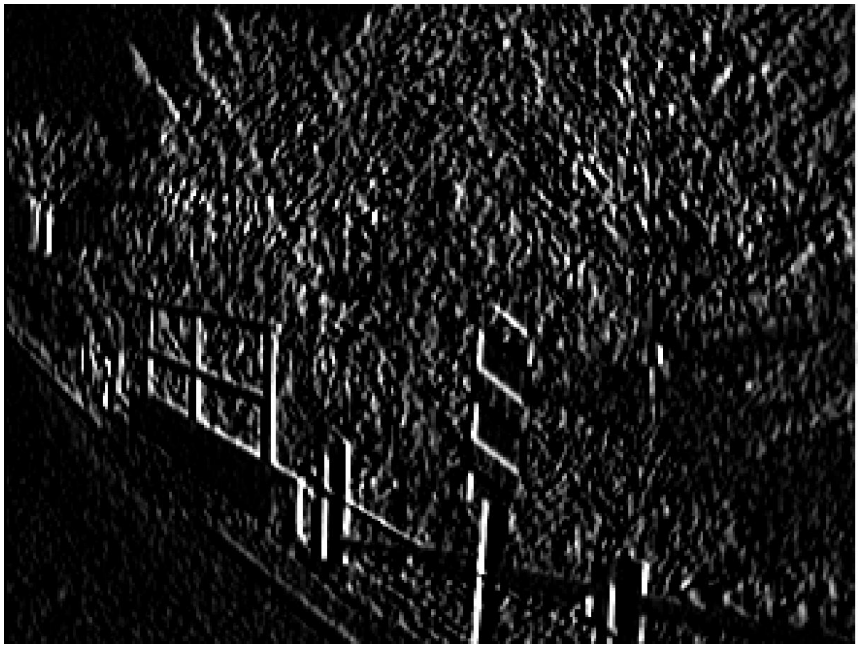
Edge Detection

- Edge Types
  - Step edge (ramp)
  - Line edge (roof)

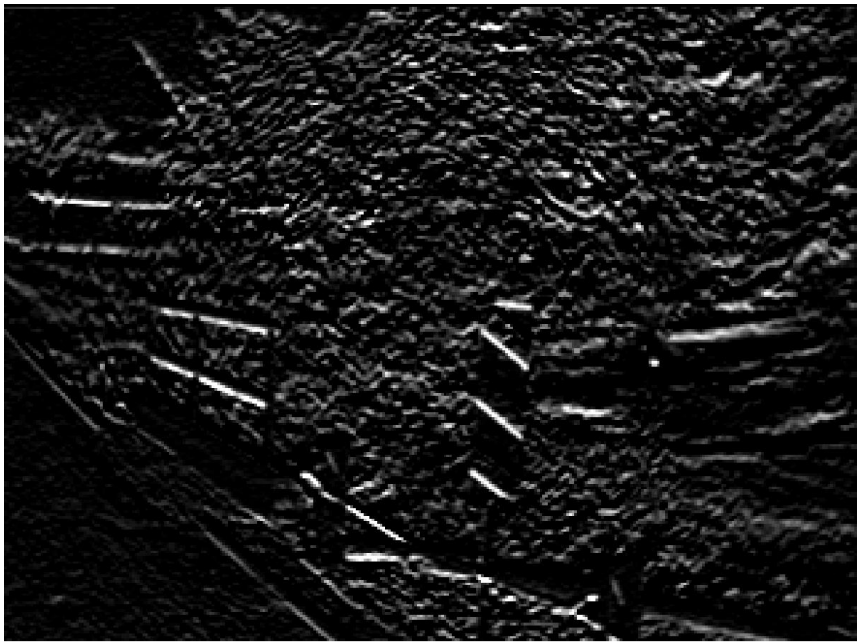


- Searching for Edges:
  - **Filter**: Smooth image
  - **Enhance**: Apply numerical derivative approximation
  - **Detect**: Threshold to find strong edges
  - **Localize/analyze**: Reject spurious edges, include weak but justified edges

Sobel Edge Detection: Gradient Approximation



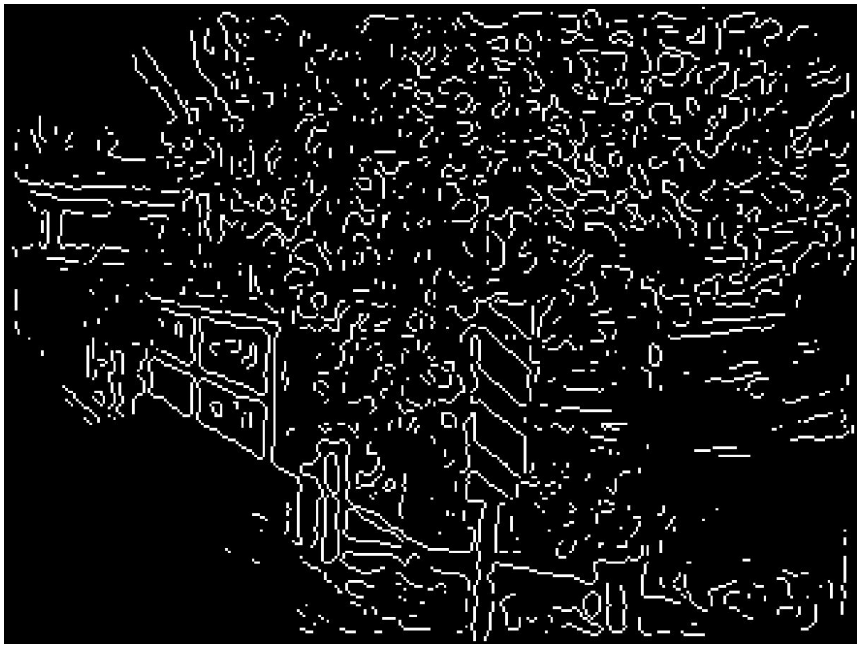
Horizontal



Vertical

Sobel vs. LoG Edge Detection:

Sobel

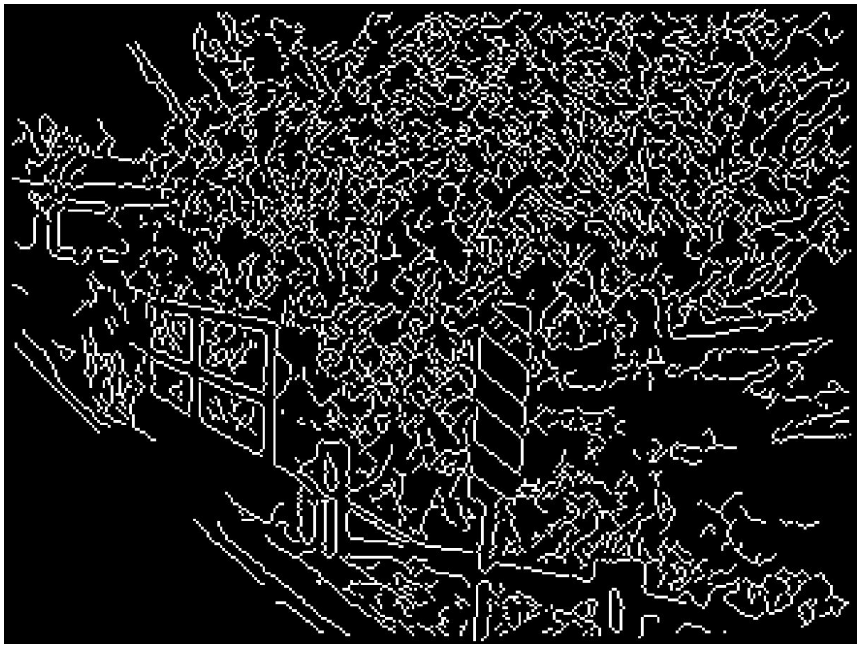


LoG

Canny Edge Detection

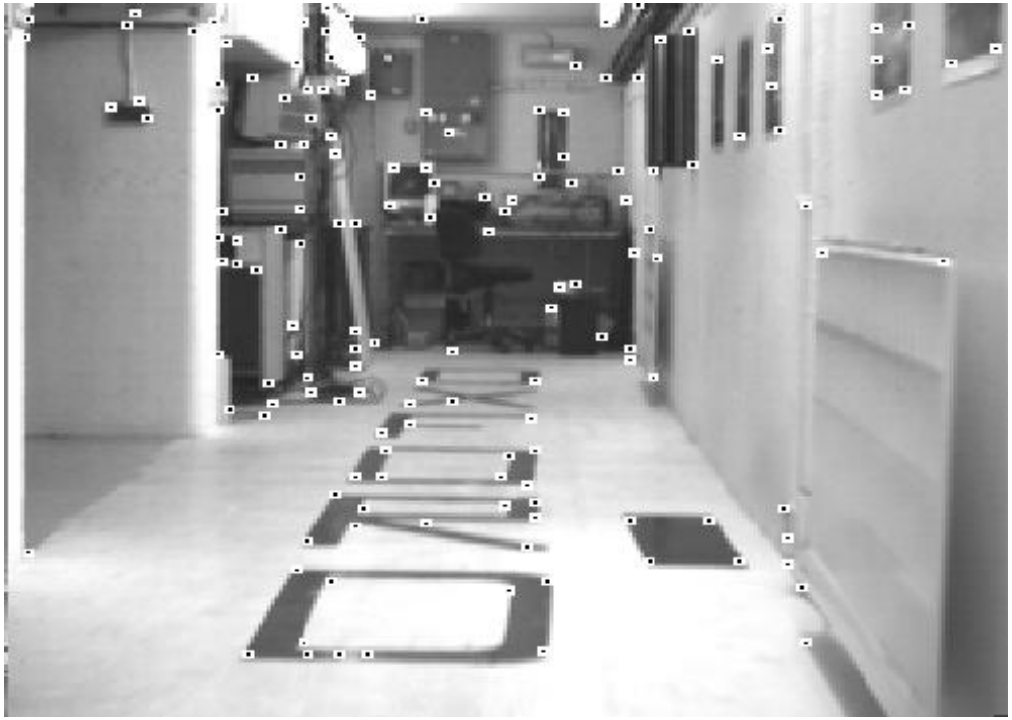
- Derivative of Gaussian
- Non-maximum suppression
  - Thin multi-pixel wide “ridges” down to single pixel
- Thresholding
  - Low, high edge-strength thresholds
  - Accept all edges over low threshold that are connected to edge over high threshold

Canny Edge Detection: Example



Corner Detection

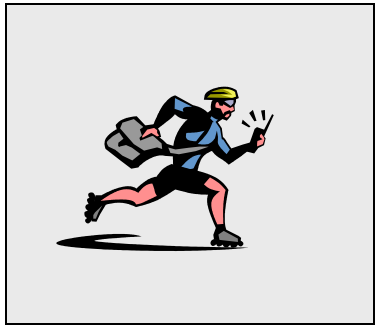
- Basic idea: Find points where two edges meet—i.e., high gradient in orthogonal directions
- Harris corners (Harris & Stephens, 1988), Susan corners (Smith & Brady, 1997)

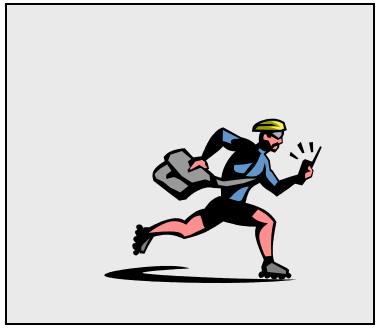


SUSAN corners

Change detection for surveillance

- Video frames: F1, F2, F3, …
- Objects appear, move, disappear
- Background pixels remain the same (simple case)

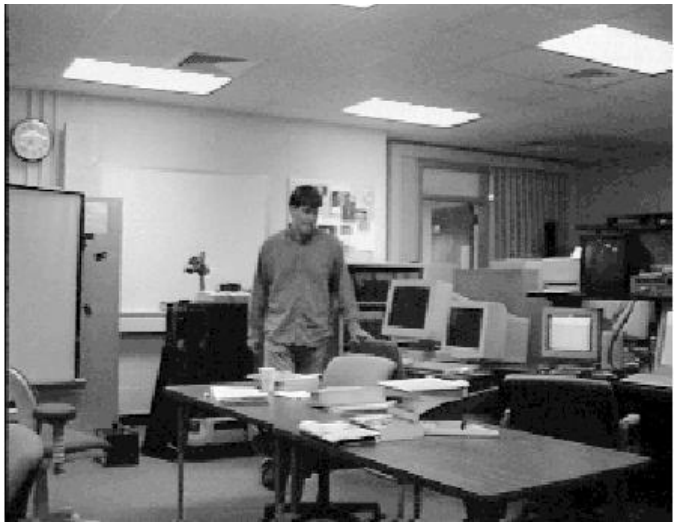




- How do you detect the moving objects?

- Simple answer: pixelwise subtraction

Example: Person detected entering room



- Pixel changes detected as difference components
- Regions are (1) person, (2) opened door, and (3) computer monitor.
- System can know about the door and monitor. Only the person region is “unexpected”.

Change Detection via Image Subtraction

for each pixel [r,c]
  if (|I1[r,c] - I2[r,c]| > threshold) then Iout[r,c] = 1 else Iout[r,c] = 0

Perform connected components on Iout.

Remove small regions.

Perform a closing with a small disk for merging close neighbors.

Compute and return the bounding boxes B of each remaining region.

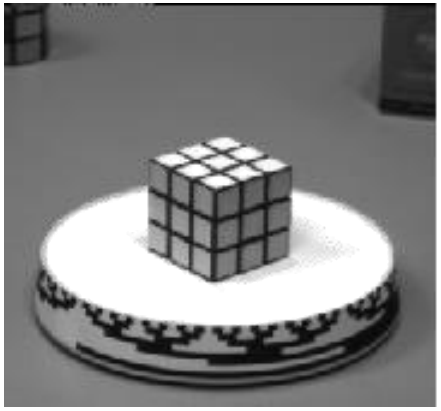
What assumption does this make about the changes?

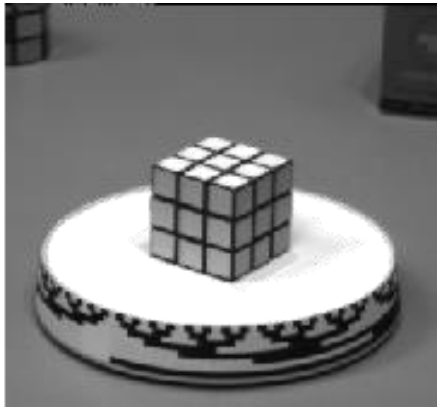
Change analysis

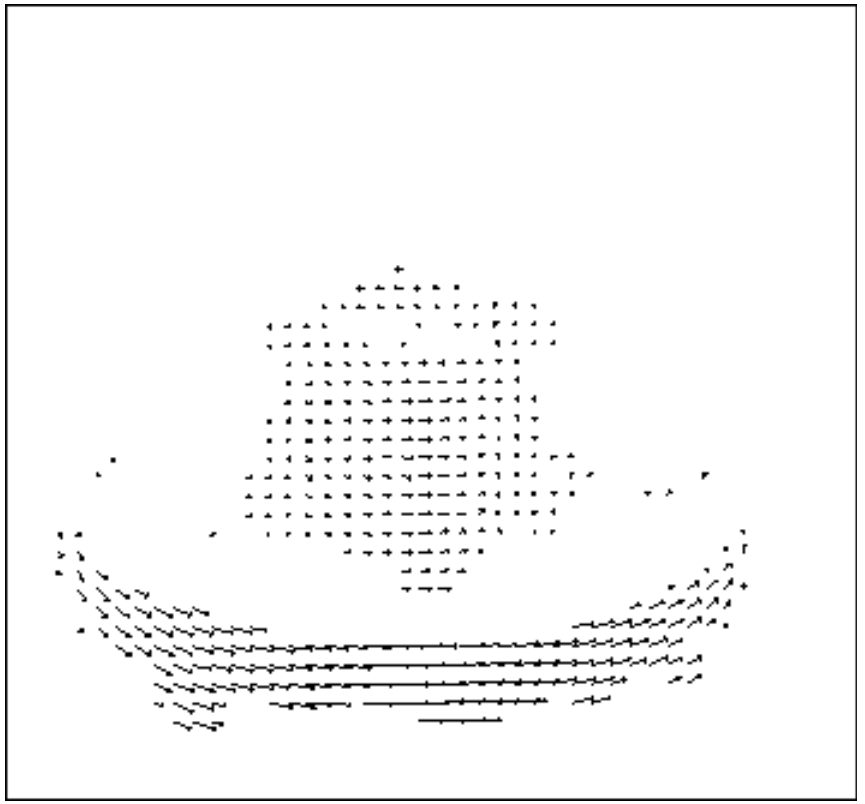
Known regions are ignored and system attends to the unexpected region of change.

Region has bounding box similar to that of a person. System might then zoom in on “head” area and attempt face recognition.

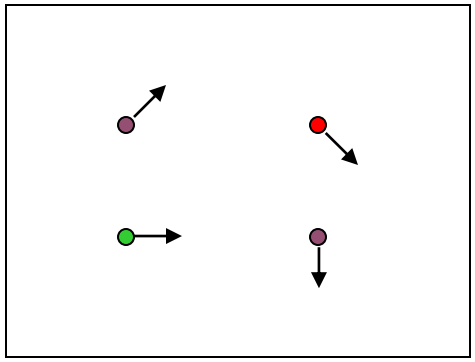
Optical flow



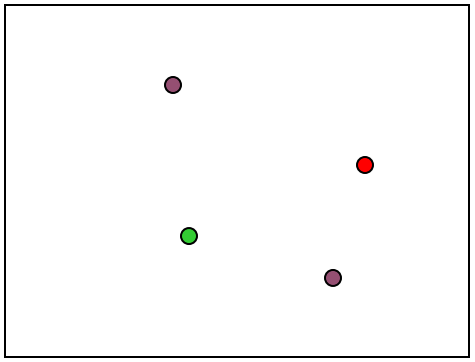




Problem definition: optical flow



$H(x, y)$



$I(x, y)$

- How to estimate pixel motion from image H to image I?
  - Solve pixel correspondence problem
    - given a pixel in H, look for nearby pixels of the same color in I

Key assumptions

- **color constancy**: a point in H looks the same in I
  - For grayscale images, this is **brightness constancy**
- **small motion**: points do not move very far

This is called the **optical flow** problem

Outline

- Classification terminology
- Unsupervised learning (clustering)
- Supervised learning
  - *k*-Nearest neighbors
  - Linear discriminants
    - Perceptron, Relaxation, modern variants
  - Nonlinear discriminants
    - Neural networks, etc.
- Applications to computer vision
- Miscellaneous techniques

Classification Terms

- **Data**: A set of *N* vectors **x**
  - Features are parameters of **x**; **x** lives in *feature space*
  - May be whole, raw images; parts of images; filtered images; statistics of images; or something else entirely
- **Labels**: *C* categories; each **x** belongs to some c_i
- **Classifier**: Create formula(s) or rule(s) that will assign unlabeled data to correct category
  - Equivalent definition is to parametrize a *decision surface* in feature space separating category members

Key Classification Problems

- What features to use? How do we extract them from the image?
- Do we even have labels (i.e., examples from each category)?
- What do we know about the structure of the categories in feature space?

Unsupervised Learning

- May know number of categories *C*, but not labels
- If we don't know *C*, how to estimate?
  - Occam's razor (formalized as Minimum Description Length, or MDL, principle): Favor simpler classifiers over more complex ones
  - Akaike Information Criterion (AIC)
- Clustering methods
  - k-means
  - Hierarchical
  - Etc.

k-means Clustering

- Initialization: Given *k* categories, *N* points. Pick *k* points randomly; these are initial means μ_1, …, μ_k
- (1) Classify *N* points according to nearest μ_i
- (2) Recompute mean μ_i of each cluster from member points
- (3) If any means have changed, goto (1)

Supervised Learning: Assessing Classifier Performance

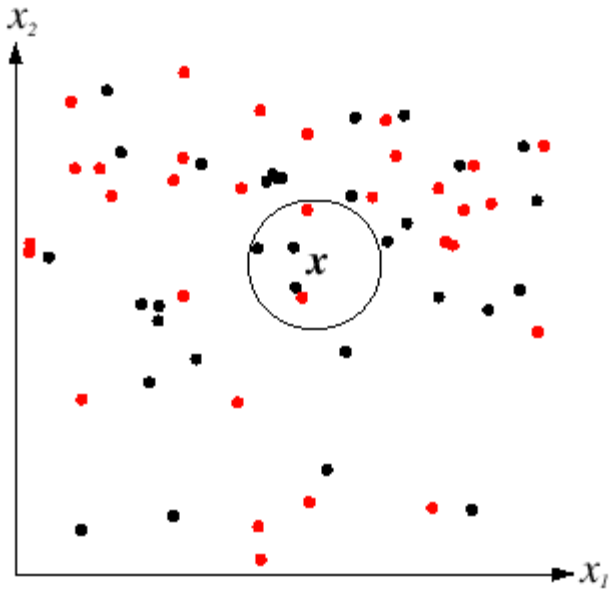
- Bias: Accuracy or quality of classification
- Variance: Precision or specificity—how stable is decision boundary for different data sets?
  - Related to generality of classification result $\rightarrow$ Overfitting to data at hand will often result in a very different boundary for new data

Supervised Learning: Procedures

- Validation: Split data into training and test set
  - Training set: Labeled data points used to guide parametrization of classifier
    - % misclassified guides learning
  - Test set: Labeled data points left out of training procedure
    - % misclassified taken to be overall classifier error
- *m*-fold Cross-validation
  - Randomly split data into *m* equal-sized subsets
  - Train *m* times on *m* - 1 subsets, test on left-out subset
  - Error is mean test error over left-out subsets
- Jackknife: Cross-validation with 1 data point left out
  - Very accurate; variance allows confidence measuring

k-Nearest Neighbor Classification

- For a new point, grow sphere in feature space until *k* labeled points are enclosed
- Labels of points in sphere vote to classify
- Low bias, high variance: No structure assumed



from Duda et al.

Linear Discriminants

- Basic: $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$
  - **w** is weight vector, **x** is data, w_0 is bias or threshold weight
  - Number of categories
    - **Two**: Decide c_1 if $g(\mathbf{x}) < 0$, c_2 if $g(\mathbf{x}) > 0$. $g(\mathbf{x}) = 0$ is decision surface—a hyperplane when $g(\mathbf{x})$ linear
    - **Multiple**: Define C functions $g_i(\mathbf{x}) = \mathbf{w}^T \mathbf{x}_i + w_{i0}$. Decide c_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$
- Generalized: $g(\mathbf{x}) = \mathbf{a}^T \mathbf{y}$
  - Augmented form: $\mathbf{y} = (1, \mathbf{x}^T)^T$, $\mathbf{a} = (w_0, \mathbf{w}^T)^T$
  - Functions $\mathbf{y}_i = y_i(\mathbf{x})$ can be nonlinear—e.g., $\mathbf{y} = (1, x, x^2)^T$

Dimensionality Reduction

- Functions $\mathbf{y}_i = y_i(\mathbf{x})$ can reduce dimensionality of feature space $\rightarrow$ More efficient classification
- If chosen intelligently, we won't lose much information and classification is easier
- Common methods
  - Principal components analysis (PCA): Maximize total "scatter" of data

$$S_T = \sum_{k=1}^{N} (x_k - \mu)(x_k - \mu)^T$$

  - Fisher's Linear Discriminant (FLD): Maximize ratio of between-class scatter to within-class scatter

PCA

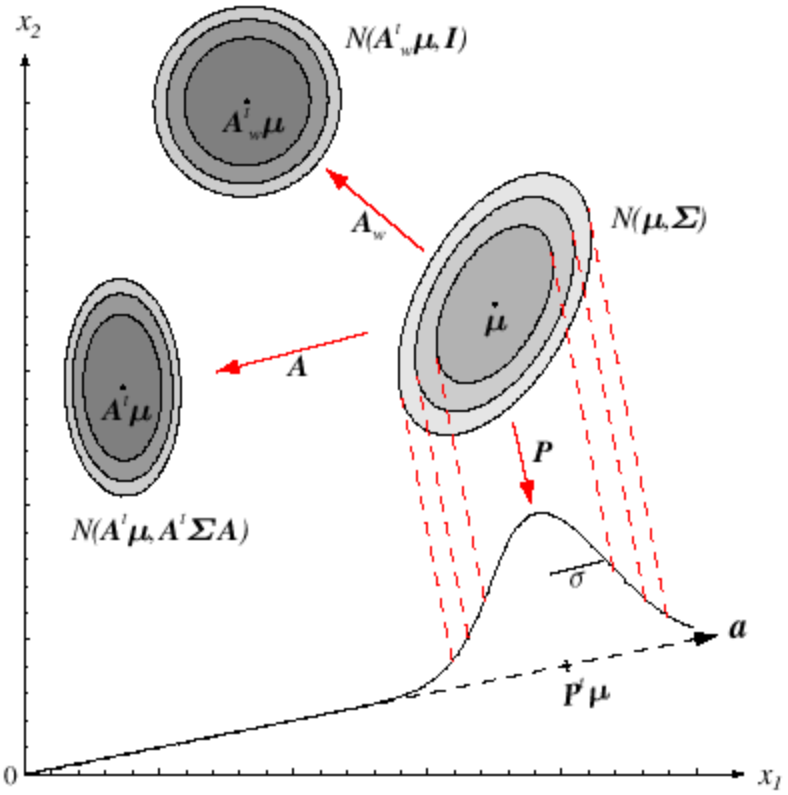


FIGURE 2.8. The action of a linear transformation on the feature space will convert an arbitrary normal distribution into another normal distribution. One transformation, **A**, takes the source distribution into distribution $N(\mathbf{A}^t\boldsymbol{\mu}, \mathbf{A}^t\boldsymbol{\Sigma}\mathbf{A})$. Another linear transformation—a projection **P** onto a line defined by vector **a**—leads to $N(\mu, \sigma^2)$ measured along that line. While the transforms yield distributions in a different space, we show them superimposed on the original x_1x_2-space. A whitening transform, $\mathbf{A}_w$, leads to a circularly symmetric Gaussian, here shown displaced. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Face Recognition (Belhumeur et al., 1996)

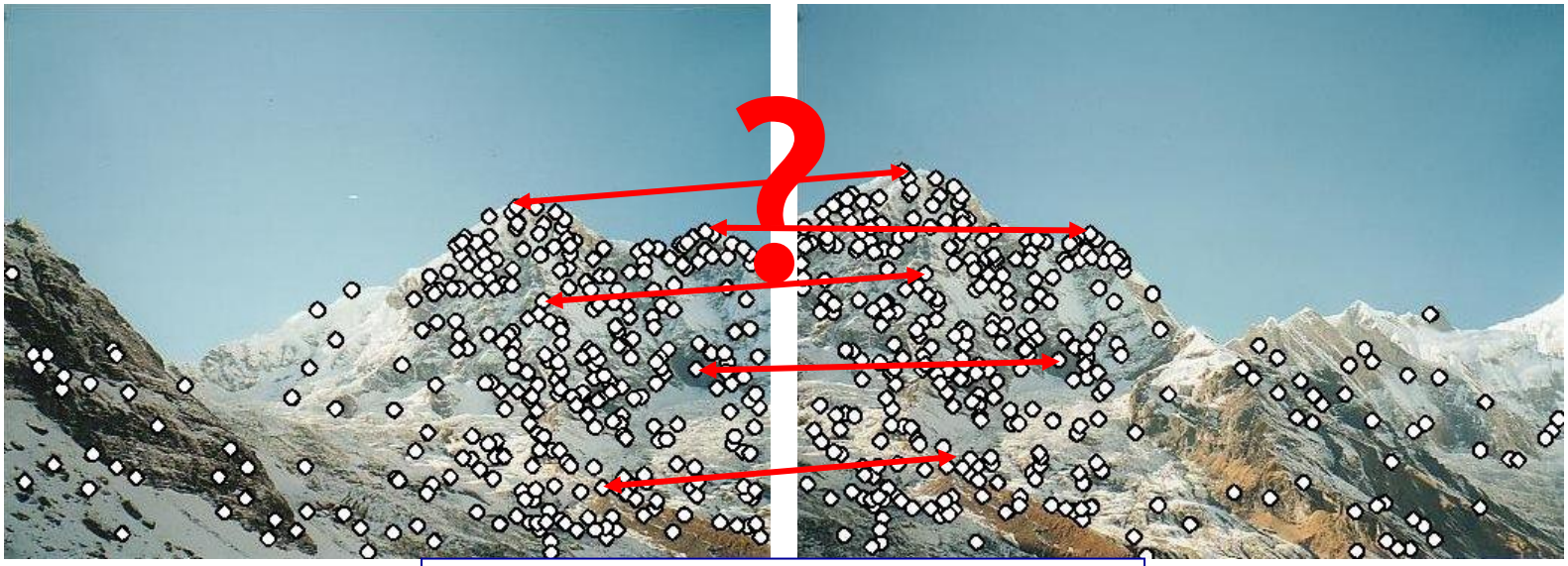
- Given cropped images {**I**} of faces with different lighting, expressions

- Nearest neighbor approach equivalent to correlation (**I**'s normalized to 0 mean, variance 1)
  - Lots of computation, storage
- PCA projection ("Eigenfaces")
  - Better, but sensitive to variation in lighting conditions
- FLD projection ("Fisherfaces")
  - Best (for this problem)

Local Descriptors

- We know how to detect points
- Next question:

How to *describe* them for matching?



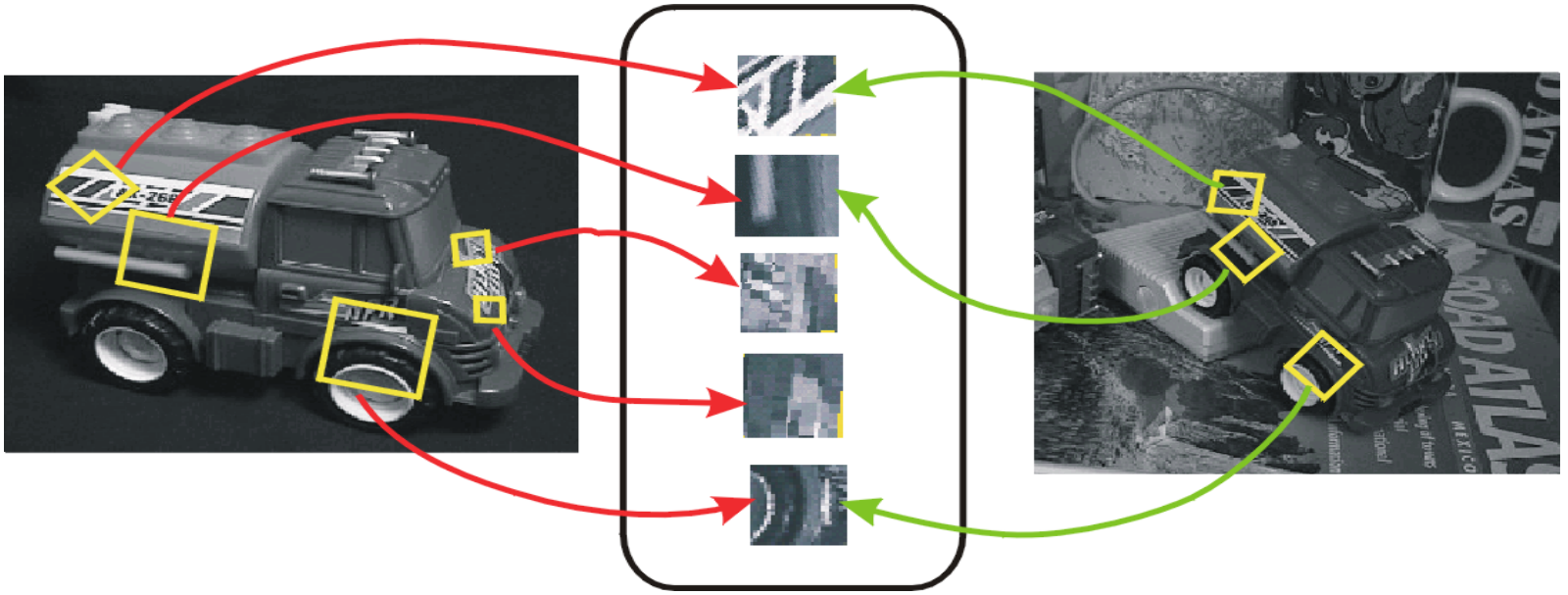
Point descriptor should be:

1. **Invariant**
2. **Distinctive**

Slide credit: Kristen Grauman

Invariant Local Features

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters

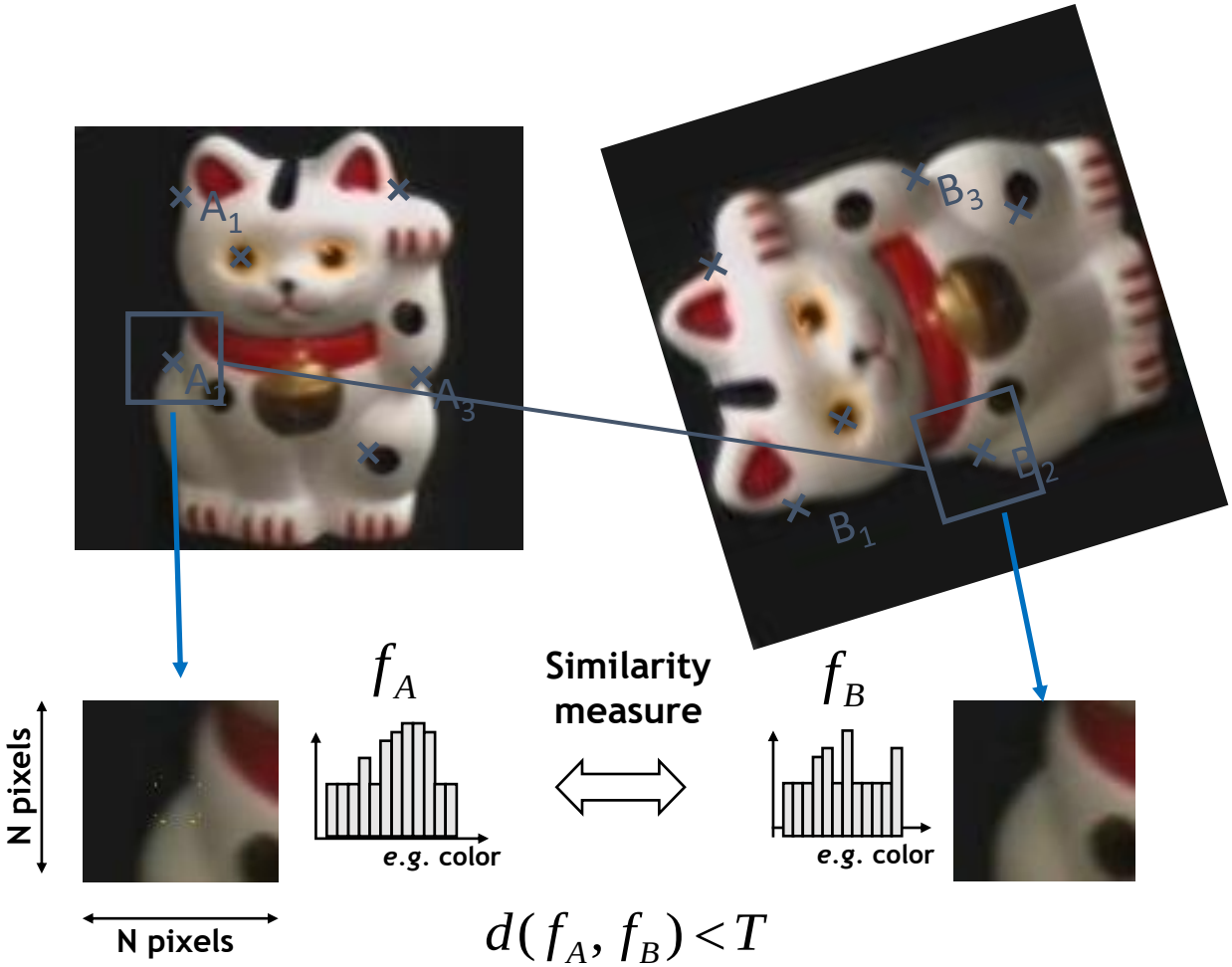


Following slides credit: CVPR 2003 Tutorial on **Recognition and Matching Based on Local Invariant Features** David Lowe

Advantages of invariant local features

- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance
- **Extensibility:** can easily be extended to wide range of differing feature types, with each adding robustness

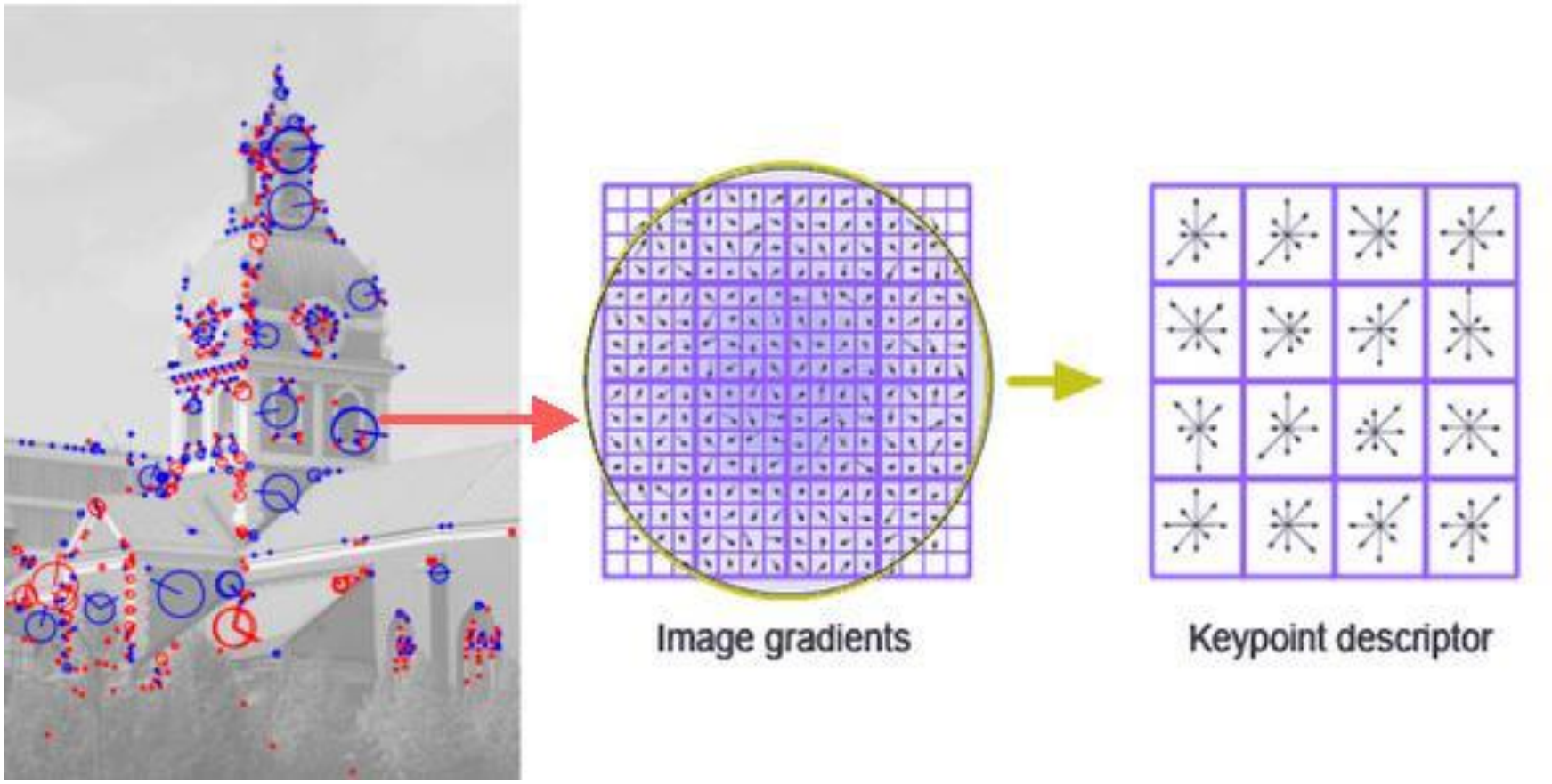
Features Descriptors: General Approach



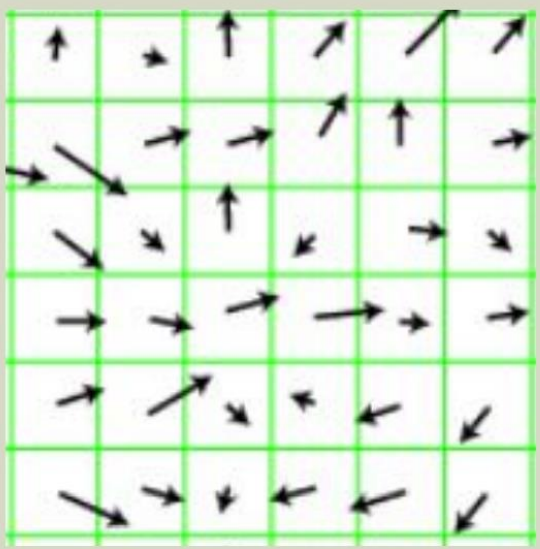
1. Find a set of distinctive key-points
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

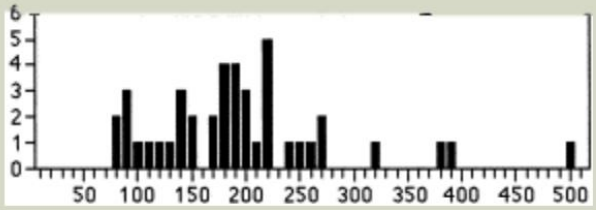
Slide credit: Bastian Leibe

SIFT descriptor formation

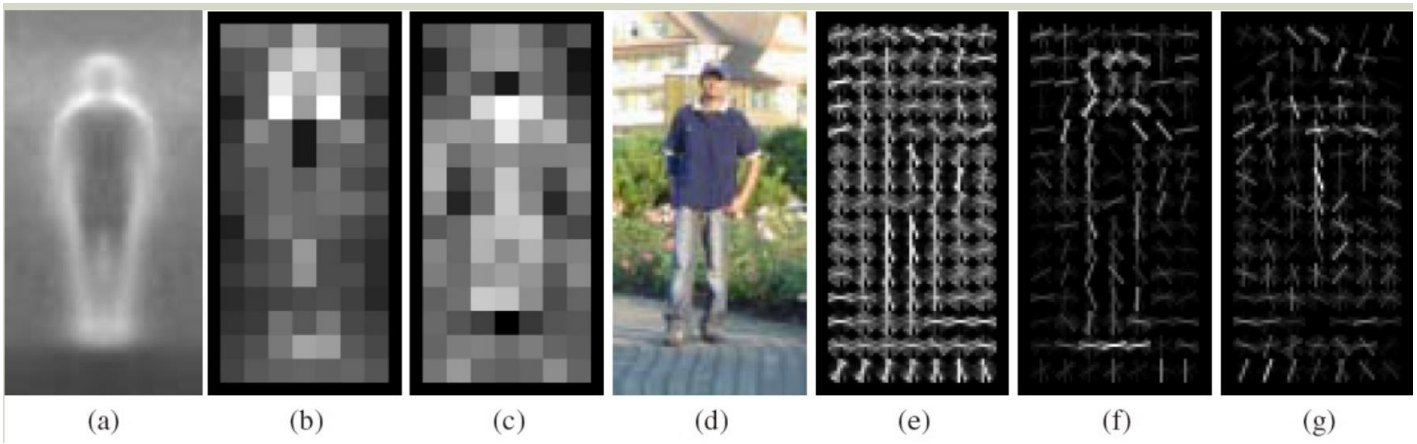


Histogram of Oriented Gradients



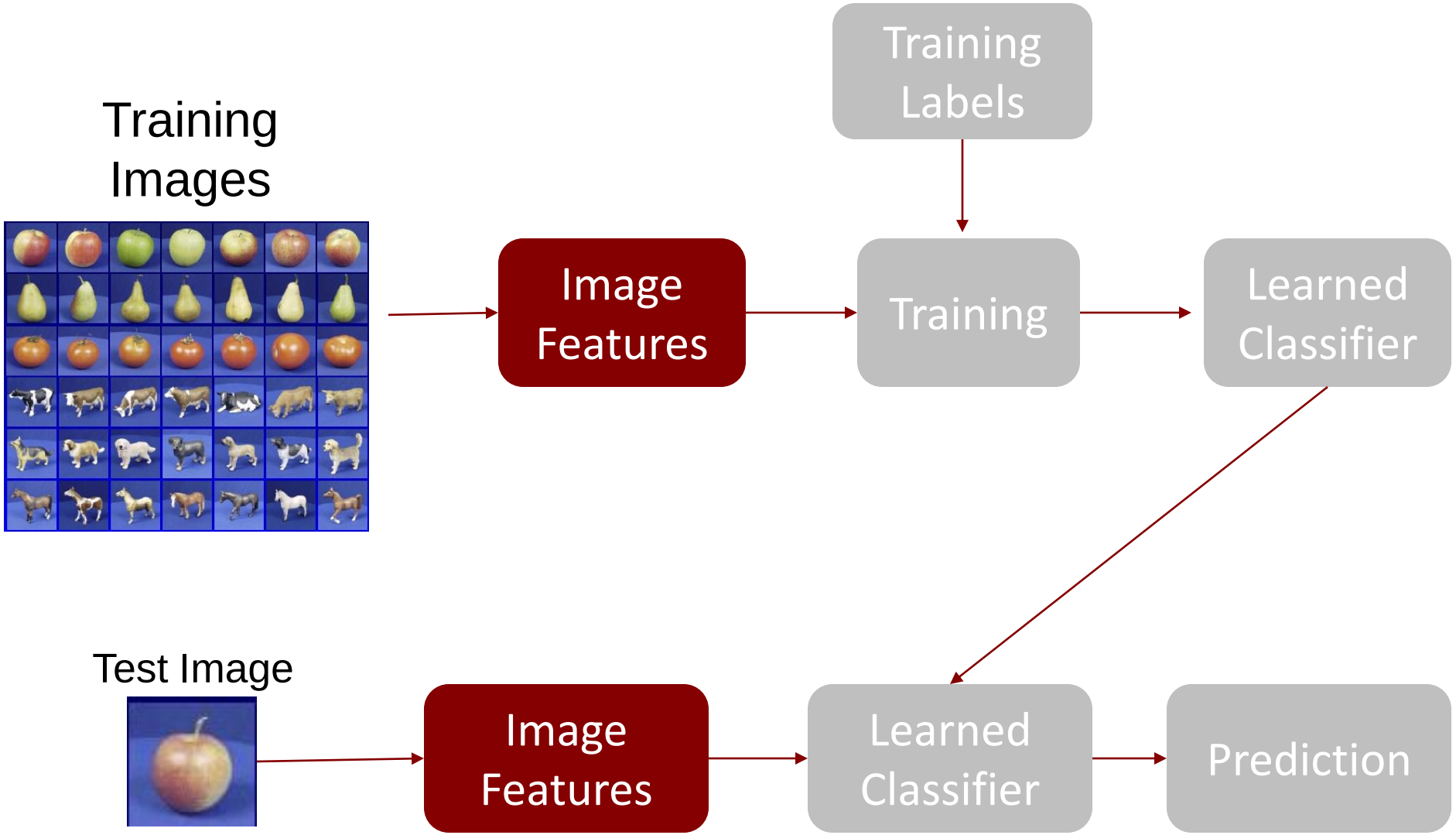


Visualizing HoG



a. Average gradient over positive examples
b. Maximum positive weight in each block
c. Maximum negative weight in each block
d. A test image
e. It's R-HOG descriptor
f. R-HOG descriptor weighted by positive weights
g. R-HOG descriptor weighted by negative weights

A simple pipeline



Advanced topics:

- Object detection and recognition
- Artificial Neural Network and Deep Learning