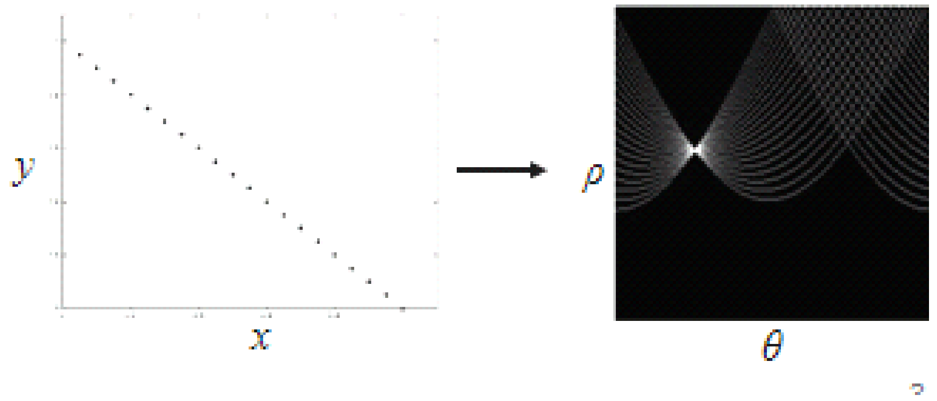
Hough Transform



Jiří Matas

Center for Machine Perception

Department of Cybernetics, Faculty of Electrical Engineering

Czech Technical University, Prague

Many slides thanks to Kristen Grauman and Bastian Leibe

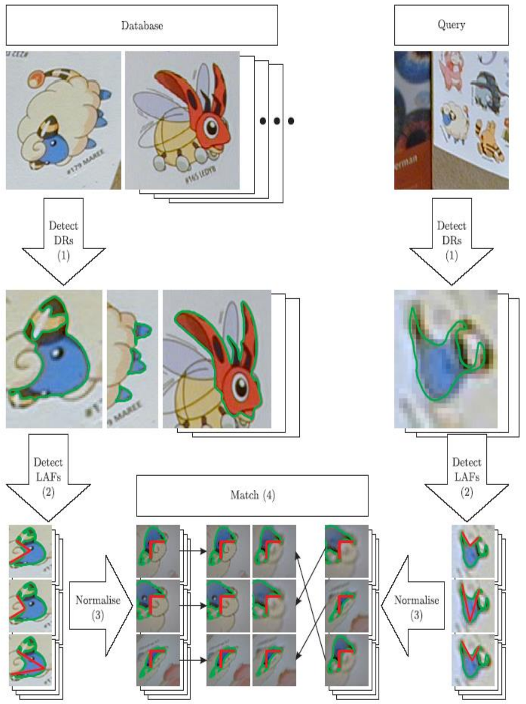
Why HT and not Recognition with Local Features?

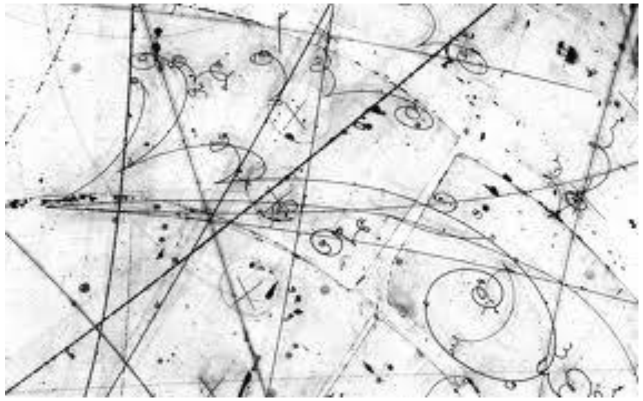
Strengths:

- applicable to many objects (e.g. in image stitching)
- is real-time
- scales well to very large problems (retrieval of millions of images)
- handles occlusion well
- insensitive to a broad class of image transformations

Weaknesses:

- applicable to recognition of specific objects (no categorization)
- applicable only to objects with distinguished local features

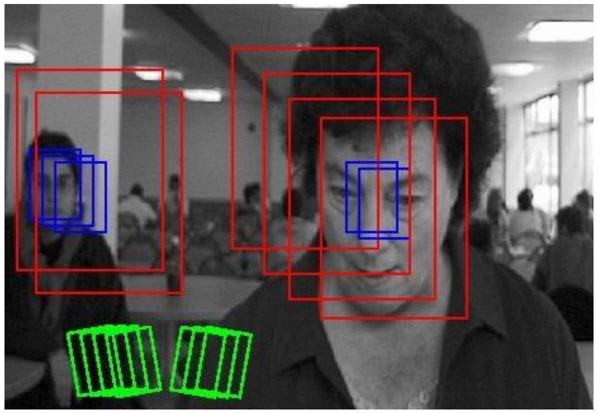




Why HT and not the Scanning Window (Viola-Jones) Method ?

Strengths:

- applicable to many classes of objects
- not restricted to specific objects
- often real-time



Weaknesses:

- extension to a large number of classes not straightforward (standard implementation: linear complexity in the number of classes)
- occlusion handling not easy
- full 3D recognition requires too many windows to be checked
- training time is potentially very long

Hough Transform

- A method for detecting geometric primitives based on evaluation of an objective function:

$$J(\Omega_c) = \sum_{i=1}^{M} p(\mathbf{x}_i, \Omega_c)$$

$\Omega_c \in \mathcal{R}^N$ is the parameter space, $\mathbf{x}_i$ are *tokens* (image points of interest)

- Origin: Detection of straight lines

- Examples of Ω_c for different geometric primitives:
  - Straight line: $\Omega_c = (a, b) \in \mathcal{R}^2$ $\quad y - ax - b = 0$
  - Circle: $\Omega_c = (x_c, y_x, r) \in \mathcal{R}^3$ $\quad (y - y_c)^2 + (x - x_c)^2 - r^2 = 0$

- Parameters evaluated on a grid
  - Discretization of Ω_c: $\Omega = N_1 \times N_2 \times N_3 \times ...$

Comparison: Template Matching and HT

- Template Matching:

```
for all $\omega \in$
    $J(\omega) = 0$
    for all $\mathbf{x} = (x, y) \in \text{Image}$ // for all $\mathbf{x}_i \sim$ tokens
        if satisfies
            $J(\omega) = J(\omega) + p(\mathbf{x})$
        else
            /* nothing */
```

  - Complexity: $O(| \quad | \times |P|)$

- HT: (basic idea: each “token” votes for all primitives it is consistent with)

```
for all $\mathbf{x}_i$
    find $\quad (\mathbf{x}_i)$
        $J(\omega) = J(\omega) + p(\mathbf{x}_i)$
```

  - Complexity: $O(| \quad (\mathbf{x}_i)| \times |P|); \; | \quad (\mathbf{x}_i)| \ll | \quad |$

Hough Transform for Straight Lines

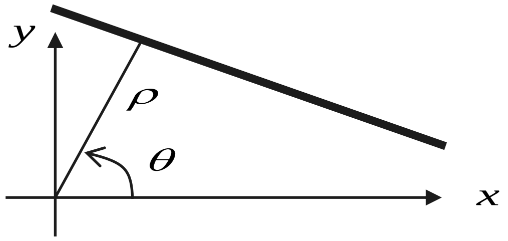
CENTER FOR MACHINE PERCEPTION

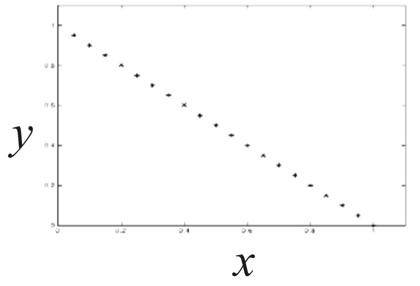
1. Define the *minimal* parametrisation (p,q) of the space of lines:
   - Most common: angle – distance from origin (ρ, θ)
   - Other options: tangent of angle – intercept (a,b) , nearest point to center, …
2. Quantize the Hough space:
   - Identify the maximum and minimum values of *a* and *b*, and the number of cells,
3. Create an accumulator array A(p,q); set all values to zero
4. (if gradient available) : For all edge points (x_i,y_i) in the image
   - Use gradient direction
   - Compute a from the equation
   - Increment A(p,q) by one

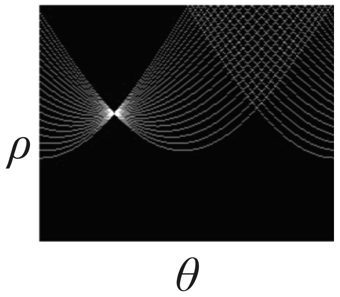
   (if gradient not available): For all edge points (x_i,y_i) in the image
   - Increment A(p,q) by one for all lines incident on x,y
5. For all cells in A(p,q)
   - Search for the maximum value of A(p,q)
   - Calculate the equation of the line
6. To reduce the effect of noise more than one element (elements in a neighborhood) in the accumulator array are increased

HT for Straight Lines: Variations

- Representation of a line
  - Usual form $y = a\,x + b$ has a singularity around 90°.
    Can be overcome by considering two cases, $y = a\,x + b$ and $x = a\,y + b$
  - Common parametrization parameterization: $x\cos(\theta) + y\sin(\theta) = \rho$



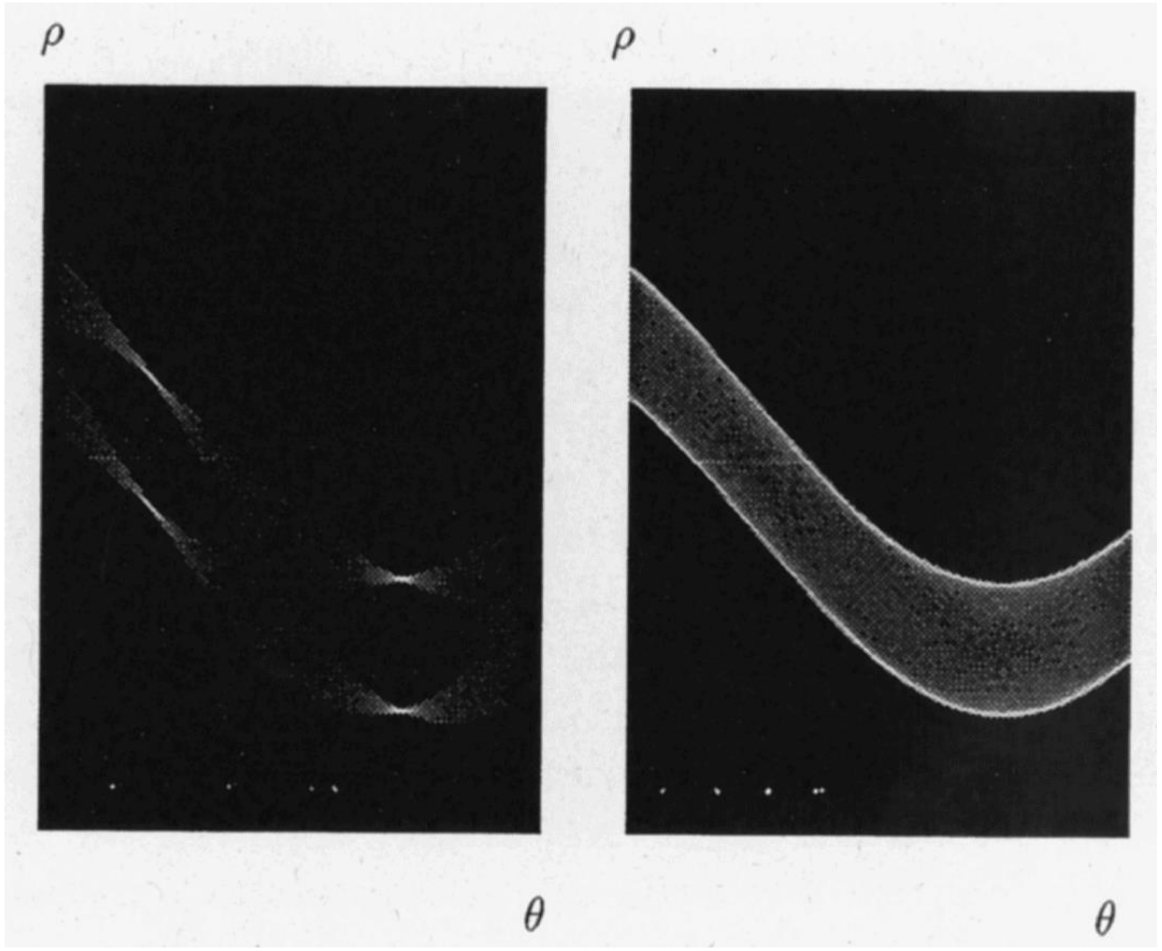




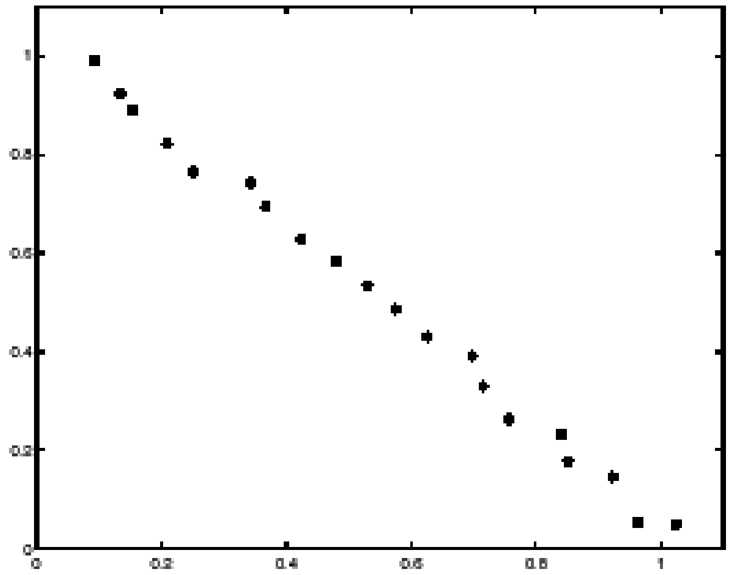
- Using gradient orientation
  - Uses not only point but also orientation consistent with the edge orientation
  - Variables: $P, \quad , \phi : P \rightarrow \langle 0, \pi)$
  - In HT: for $(\mathbf{x}_i, \phi(\mathbf{x}_i))$
  - Can be used by weighting the strength of the vote by: $|\phi - \psi|$
    ψ … line orientation, ϕ … gradient orientation

Examples

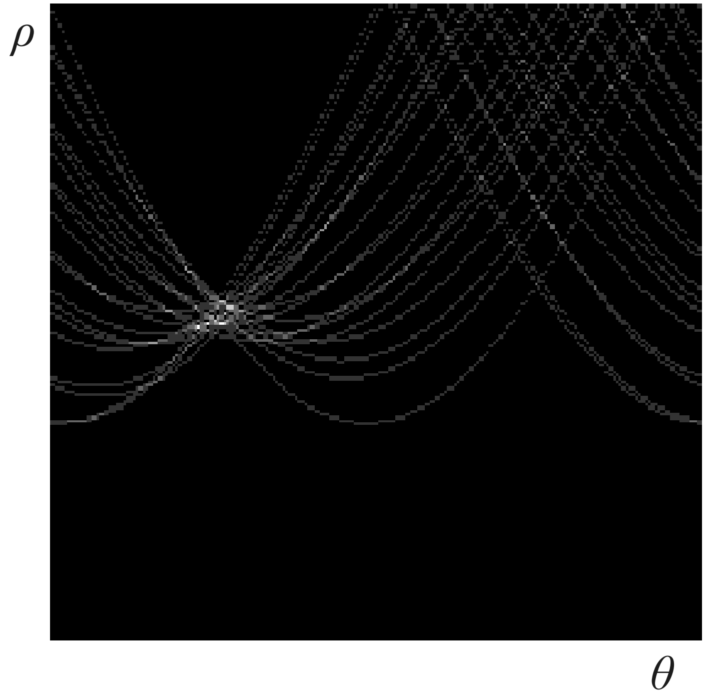
- Hough transform for a square (left) and a circle (right)



Hough Transform: Noisy Line



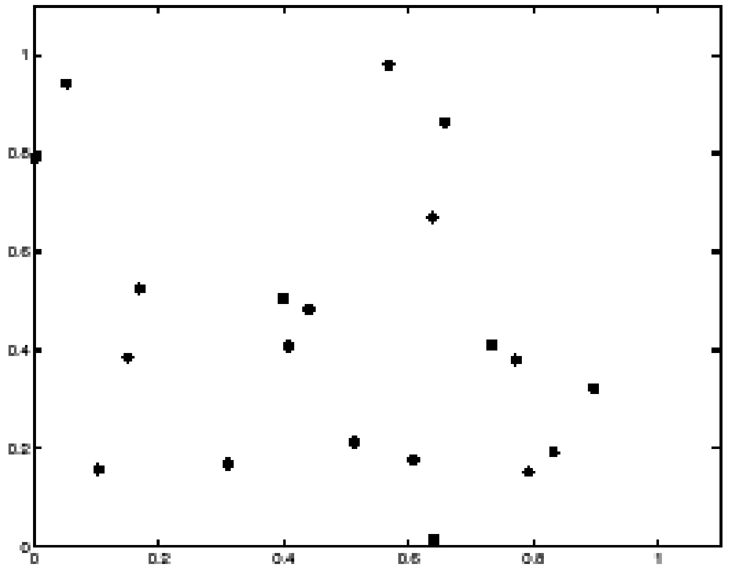
Tokens



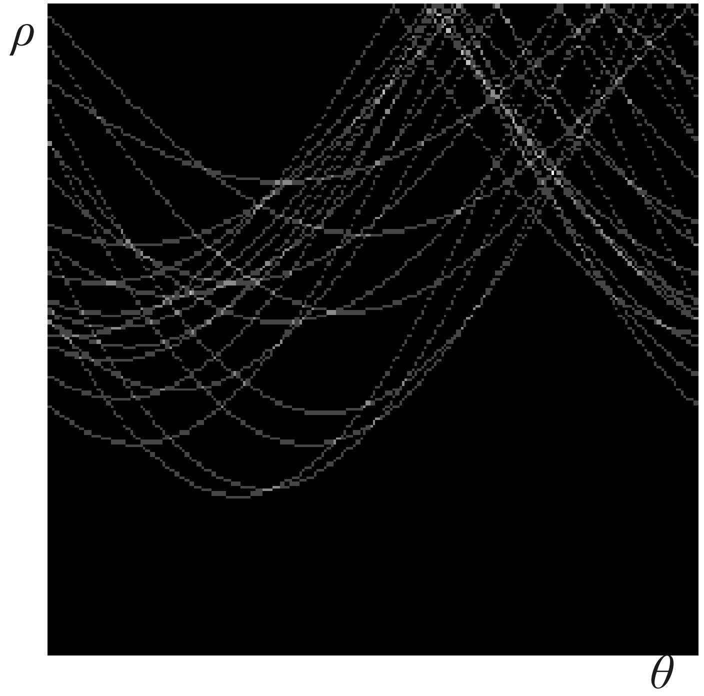
Votes

- Problem: Finding the true maximum

Hough Transform: Noisy Input



Tokens



Votes

- Problem: Lots of spurious maxima

HT for different primitive: circles, ...

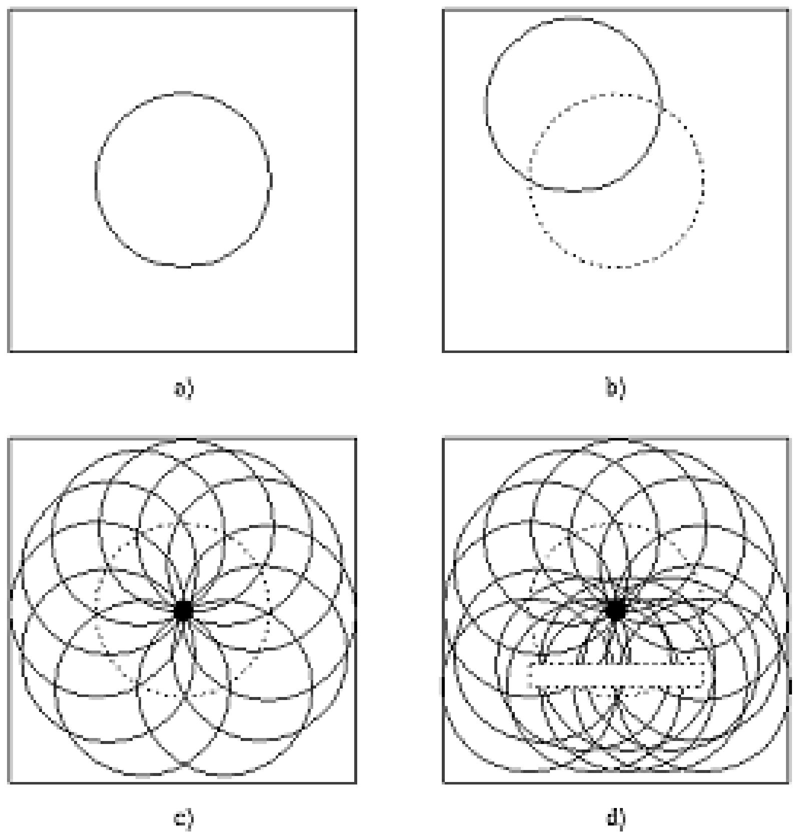


Figure 5.29 *Hough transform - example of circle detection. (a) Original image of a dark circle (known radius r) on a bright background, (b) for each dark pixel, a potential circle-center locus is defined by a circle with radius r and center at that pixel, (c) the frequency with which image pixels occur in the circle-center loci is determined; the highest-frequency pixel represents the center of the circle (marked by •), (d) the Hough transform correctly detects the circle (marked by •) in the presence of incomplete circle information and overlapping structures (see Figure 5.34 for a real-life example).*

- circles with fixed radius
- circles
- squares with a known orientation and size
- rectangles

HT for multiple instances

1. $p_1 = HT(P, \quad)$: strongest result of HT
2. Set $P_1 = P \setminus p_1$
3. Unvote p_1
4. $p_2 = HT(P_1, \quad)$
5. Cont. to get as many instances as required

- Greedy
- Sequential

Hough Transform Problems

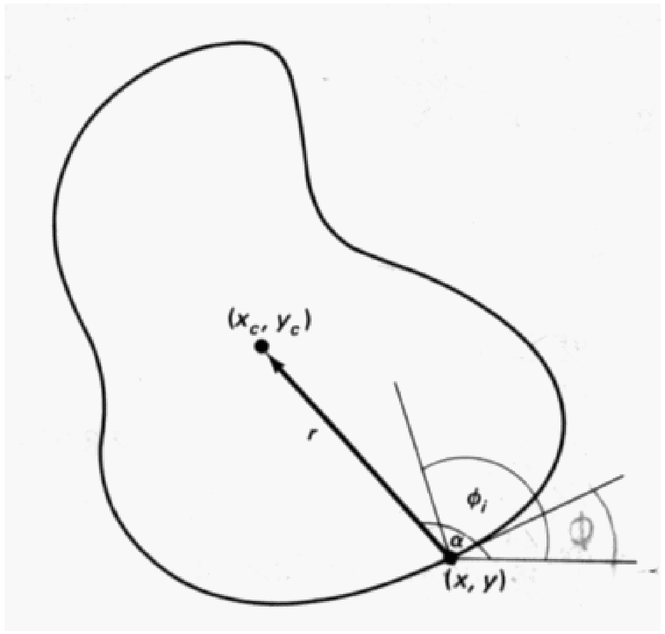
1. Search space (accumulator size) gets prohibitively large easily
   - Line segments: θ, ρ, t_1, t_2
   - Circular arc: r, c_x, c_y, t_1, t_2
2. Cost function must be additive.
3. Greedy assignment rule of a token to primitive
4. No global objective function for multiple primitives (global optimization for one primitive only)

When is the Hough transform useful?

- Textbooks often imply that it is useful mostly for finding lines
  - In fact, it can be very effective for recognizing arbitrary shapes or objects (Generalized HT)
- The key to efficiency is to have each feature (token) determine as many parameters as possible
  - For example, lines can be detected much more efficiently from small edge elements (or points with local gradients) than from just points
  - For object recognition, each token should predict location, scale, and orientation (4D array)
- Bottom line: The Hough transform can extract feature groupings from clutter in linear time!

Generalized Hough Transform [Ballard81]

- Generalization for an arbitrary contour or shape
  - Choose reference point for the contour (e.g. center)
  - For each point on the contour remember where it is located w.r.t. to the reference point
  - Remember radius r and angle ϕ relative to the contour tangent
  - Recognition: whenever you find a contour point, calculate the tangent angle and 'vote' for all possible reference points

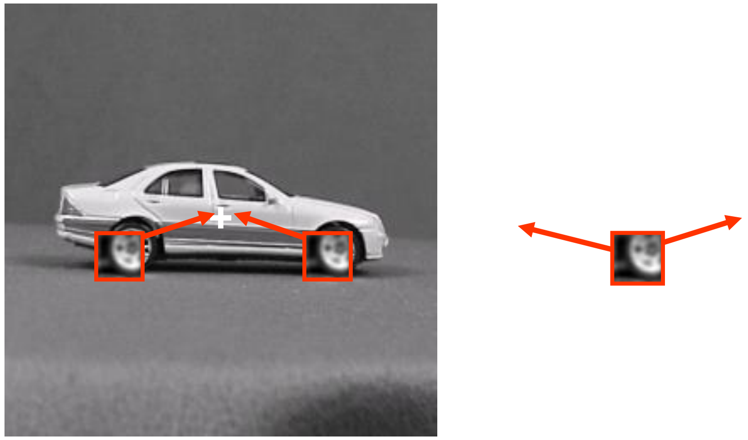


  - Instead of reference point, can also vote for transformation

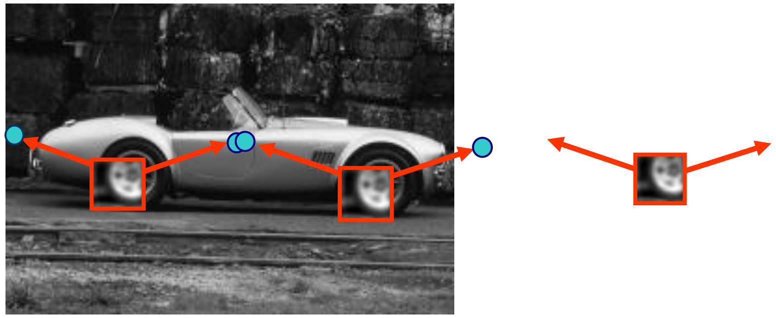
$\Rightarrow$ The same idea can be used with local features!

Gen. Hough Transform with Local Features

- For every feature, store possible “occurrences”



For new image, let the matched features vote for possible object positions



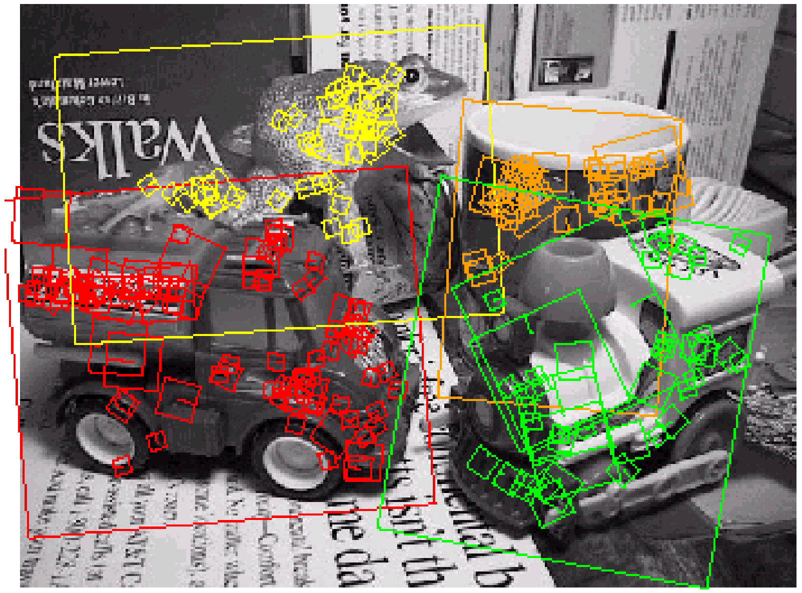
– Object identity
– Pose
– Relative position

K. Grauman, B. Leibe

Finding Consistent Configurations

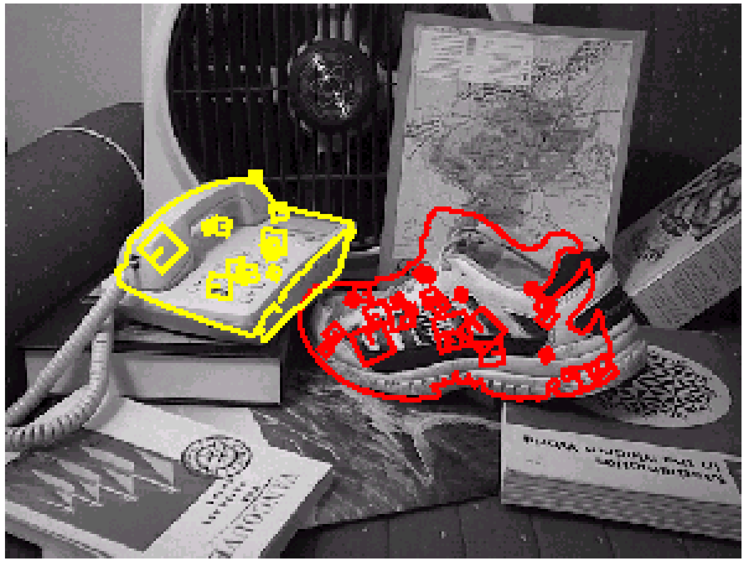
- Global spatial models
  - Generalized Hough Transform [Lowe99]
  - RANSAC [Obdrzalek02, Chum05, Nister06]
  - Basic assumption: object is planar

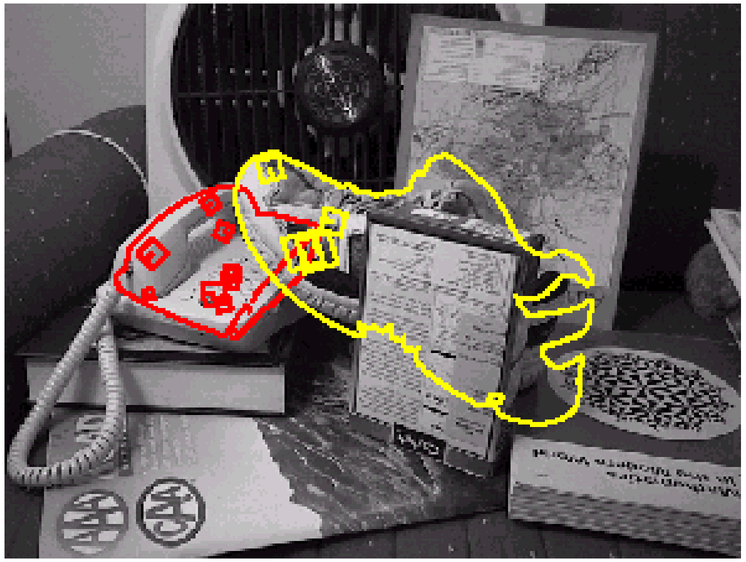
- Assumption is often justified in practice
  - Valid for many structures on buildings
  - Sufficient for small viewpoint variations on 3D objects

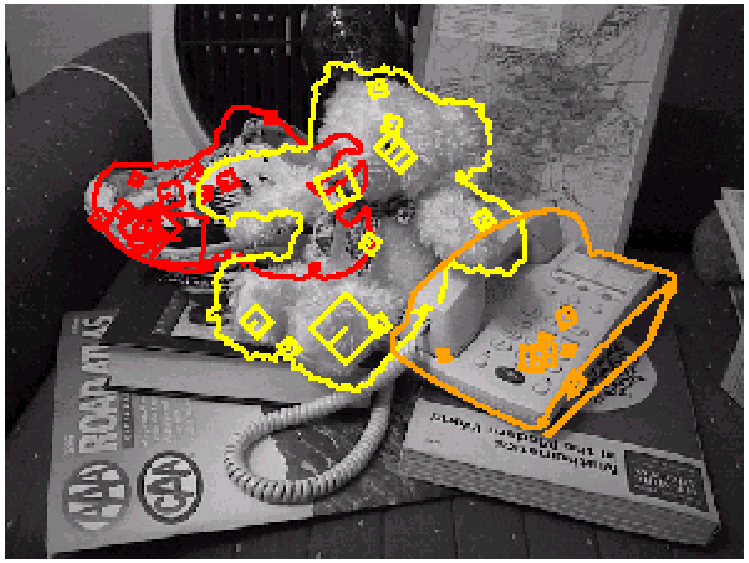


3D Object Recognition

- Gen. HT for Recognition
  - Typically only 3 feature matches needed for recognition
  - Extra matches provide robustness
  - Affine model can be used for planar objects







K. Grauman, B. Leibe

Comparison

Gen. Hough Transform

- Advantages
  - Very effective for recognizing arbitrary shapes or objects
  - Can handle high percentage of outliers (¿95%)
  - Extracts groupings from clutter in linear time
- Disadvantages
  - Quantization issues
  - Only practical for small number of dimensions (up to 4)
- Improvements available
  - Probabilistic Extensions
  - Continuous Voting Space

  $\Big\}$ **[Leibe08]**

RANSAC

- Advantages
  - General method suited to large range of problems
  - Easy to implement
  - Independent of number of dimensions
- Disadvantages
  - Only handles moderate number of outliers (¡50%)
- Many variants available, e.g.
  - PROSAC: Progressive RANSAC [Chum05]
  - Preemptive RANSAC [Nister05]

RHT = Randomized Hough Transform [Xu93]

CENTER FOR MACHINE PERCEPTION

In: $E = \{e_i\}, m(\quad, e) = 0$

Out: $\quad_{S_1}, \quad_{S_2}, \ldots, \quad_{S_N}$

Repeat:

I. Hypothesis
1. Select random M feature points $e_{k_1}, \ldots, e_{k_M}$
2. Compute $\quad_k : m(\quad_k, e_{k_j}) = 0, j = 1, \ldots, M$

II. Pre-Verification
3. Add 1 to accumulator $\quad_k$
4. If (accumulator$(\quad_k) > T_1$) goto *III.*
   Else goto *I.*

III. Verification
5. Find support for $\quad_k$
6. If (support$(\quad_k) > T_2$) output $\quad_k$
7. Reset accumulator

Proc v Out Omega-S-N, a v algoritmu Omega-N

Probabilistic Hough Transform [Kiryati et al. 91]

Idea: Evaluate $\sum_{i=1}^{N} p(x_i, \quad)$ using only a fraction $f = \frac{k_{MAX}}{N}$ of N points x_i

Algorithm:

1. Select k_{MAX} points at random
2. Perform standard HT

Analysis:

- Selection of k_{MAX} is incorrect

  $\Rightarrow$ the number *L* of selected points from L_N points of a line in a random subset of k_{MAX} points is governed by hypergeometric, not binomial distance

$$P(L_N) = \frac{\binom{L}{L_N}\binom{N-L}{k_{MAX}-L_N}}{\binom{N}{k_{MAX}}}$$

$$\mu = \overline{N} \qquad \sigma^2 = k_{MAX}\frac{L_N(N-L_N)}{N^2}\left(1 - \frac{k_{MAX}-1}{N-1}\right)$$

PHT = Monte Carlo Evaluation of $\sum_{i=1}^{N} p(x_i, \quad)$

CENTER FOR MACHINE PERCEPTION

Idea:

1. Evaluate $\sum_{i=1}^{N} p(x_i, \Omega)$ using only a fraction $f = \frac{k_{MAX}}{N}$ of N points x_i
2. Apply standard MC analysis to find k_{MAX} in PHT to guarantee $P\{\text{false\_positive}\}$ and $P\{\text{false\_negative}\} < \epsilon$

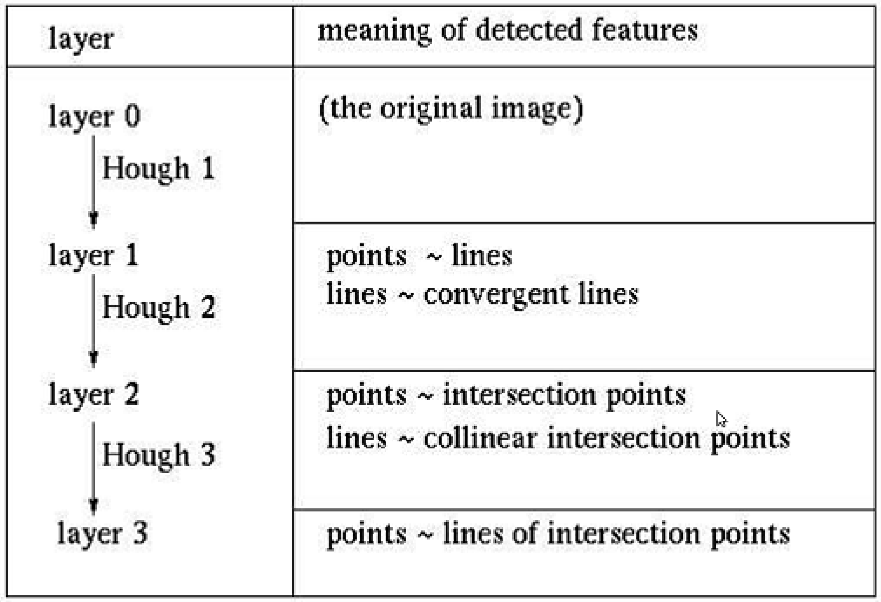
Algorithm:

1. Select a random point
2. Vote and return it
3. Finish if k_{MAX} reached

CHT = Cascaded Hough Transform [Tuytelaars et al. 97]

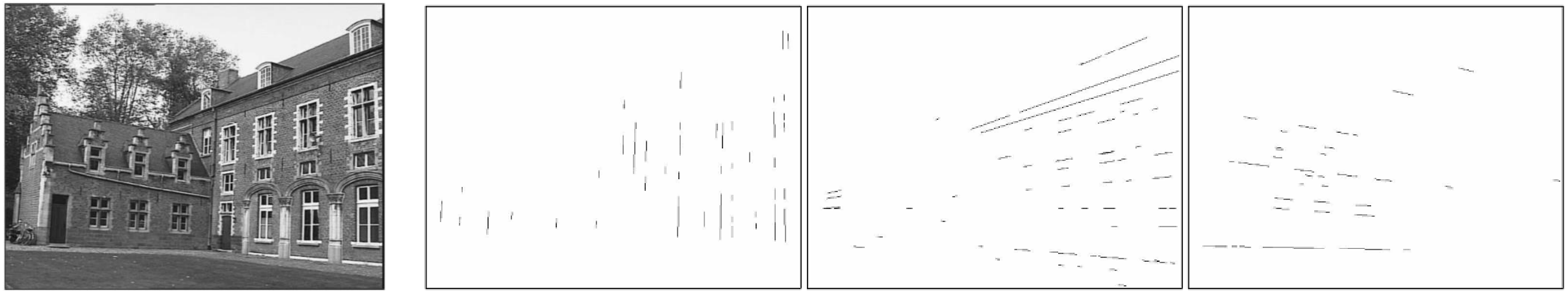
- Finds structures at different hierarchical levels by iterating one kind of HT (fixed points, fixed lines, lines of fixed points, pencils of fixed lines)
- Uses duality of lines and points in image and parameter spaces
- Algorithm:

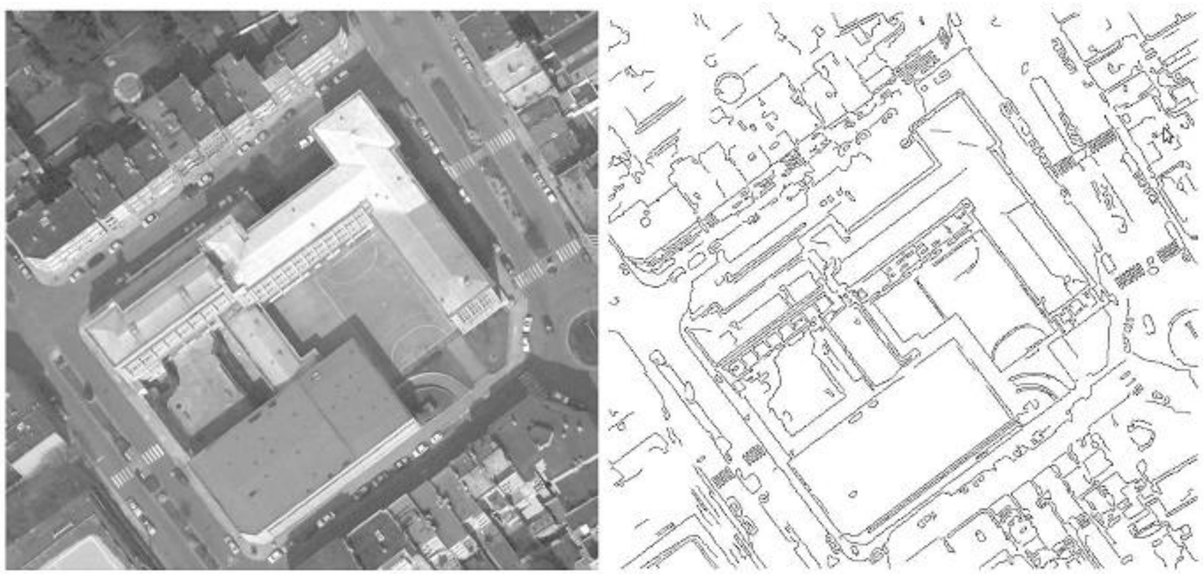
1. First HT: detects lines in the image and keeps dominant peaks in the parameter space
2. Second HT: detects lines of collinear peaks in parameter space and keeps vertices where several straight lines in the original image intersect (*vanishing points*)
3. Third HT: applied to the peaks of thto detect collinear vertices (*vanishing lines*)



layer	meaning of detected features
layer 0	(the original image)
↓ Hough 1	
layer 1	points ~ lines lines ~ convergent lines
↓ Hough 2	
layer 2	points ~ intersection points lines ~ collinear intersection points
↓ Hough 3	
layer 3	points ~ lines of intersection points

CHT: Experiments


Lines belonging to one of the three detected vanishing points


Aerial image of buildings and streets (left), the corresponding edges (right)

macros.tex
sfmath.sty
cmpitemize.tex

Thank you for your attention.