Feature Detection

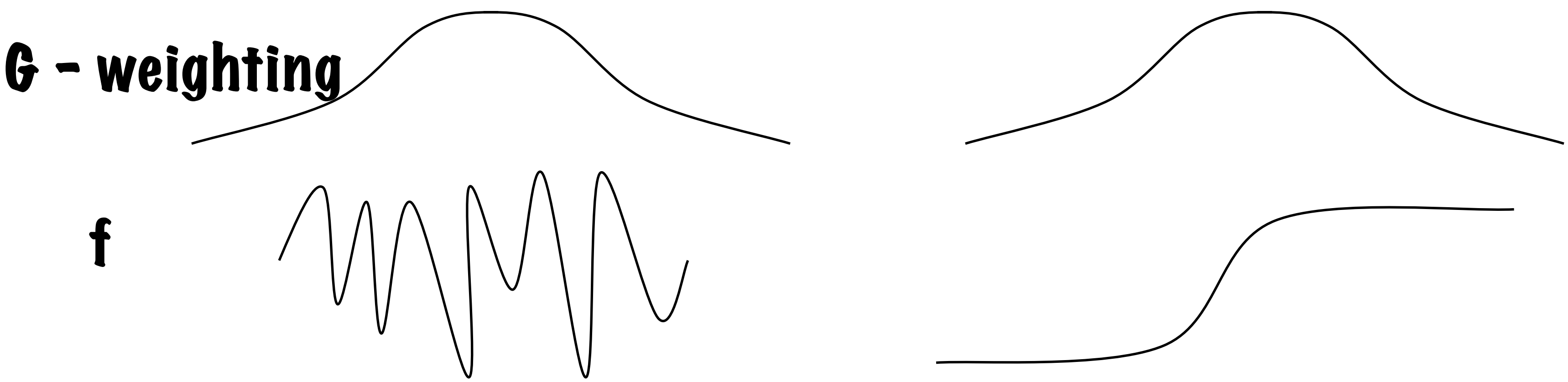
Features

- Places where intensities vary is some prescribed way in a small neighborhood
- How to quantify this variability
  - Derivatives - direcitonal derivatives, magnitudes
    - Scale and smoothing
  - Statistics
    - Variance of some property (Gaussian weights on

$$\text{edgeness} = \mathrm{E}\{\mathrm{f}^2\} - \mathrm{E}\{\mathrm{f}\}^2 = \int \mathrm{G}(\mathrm{x},\mathrm{y})\mathrm{f}^2(\mathrm{x},\mathrm{y}) - \left[\int \mathrm{G}(\mathrm{x},\mathrm{y})\mathrm{f}(\mathrm{x},\mathrm{y})\right]^2$$

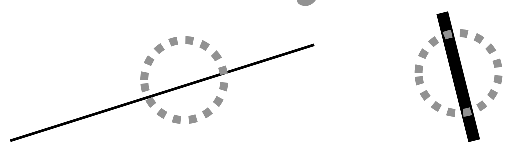
Differential vs Stastical Variation

- Derivatives measure monotonicity and direction
- Variance captures more general types of variation

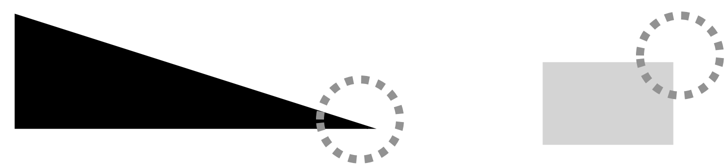


Other Types of Features

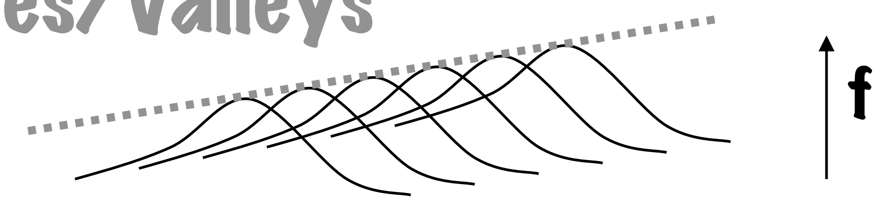
- Lines (contrast, width, orientation)



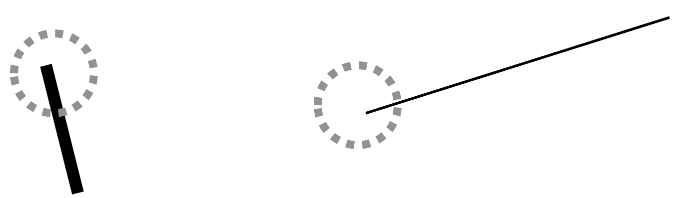
- Corners (contrast, angle, orientation)



- Ridges/valleys

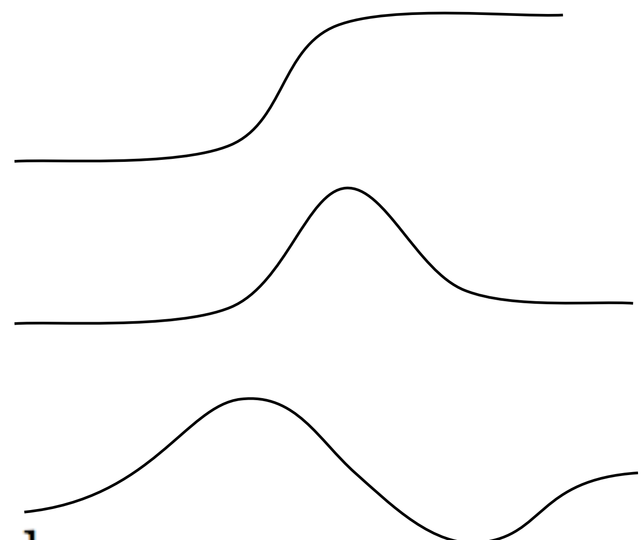


- Line ends



Edges

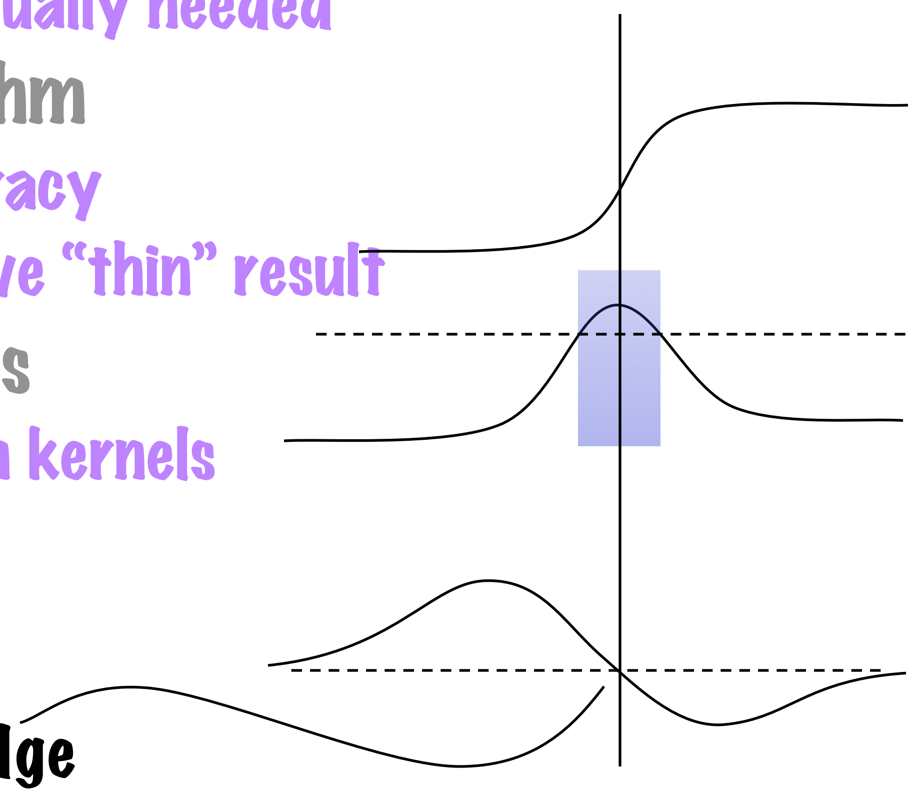
- Places of “sharp” change in brightness
- How to quantify this...
- In 1D - model
  - Gauss conv w/step
    - f(x) = g(x)*s(x)
    - Derivative of step is delta



$$
\begin{aligned}
f(x) &= \mathrm{erf}_\sigma(\mathrm{x}) = \textstyle\int_{-\infty}^{\mathrm{x}} \mathrm{G}_\sigma(\mathrm{a})\mathrm{da} \\
f'(x) &= G_\sigma(x) \\
f''(x) &= \tfrac{x}{\sigma^2} G_\sigma(x)
\end{aligned}
$$

Edges

- High derivatives with Non-maximal supression
  - Local max of f'(x)
  - | f'(x) | > T ; f''(x) = 0 ; f'''(x) <> 0
  - f'''(x) condition not usually needed
- Zero-crossing algorithm
  - Can do sub-grid accuracy
  - For pixels - like to have "thin" result
- Computing derivatives
  - Derivative of Gaussian kernels
  - Parameter tuning
    - Sigma, T



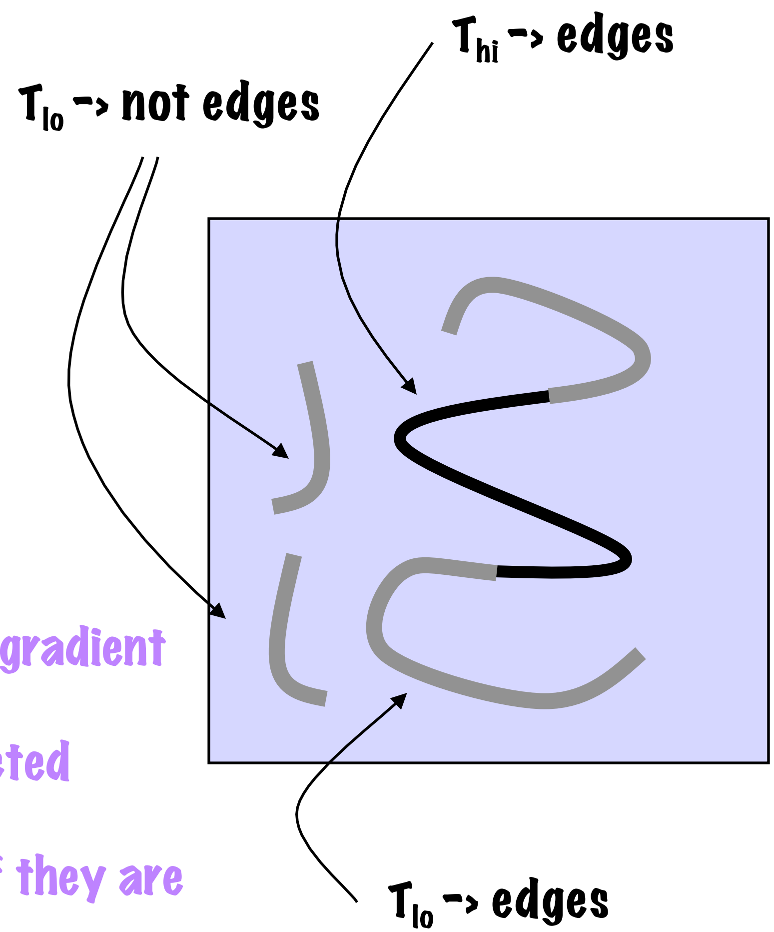
Generalizing To Multiple Dimensions

- Marr-Hildreth
  - Gradient threshold
  - Zero crossings in the *Laplacian* of f(x)
  - $f_{xx}(x) + f_{yy}(x) = 0$; $f_x(x)^2 + f_y(x)^2 > T^2$
- Canny Edges (1980s)
  - Do nonmaximal suppression along the direction of the gradient

Canny Edges

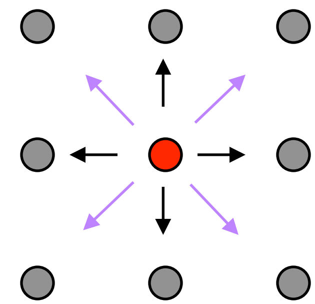
$$
\begin{aligned}
\nabla f &= \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \end{pmatrix} \\
|\nabla f(x)| &> T \\
H_f &= \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} \\
(\nabla f)^T H_f \nabla f &= 0 \\
(\nabla f)^T H_f \nabla &= f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_y^2 f_{yy}
\end{aligned}
$$

- Preprocessing - denoising
- Derivatives - computed with kernels
- Zero crossings - 8-connected thin lines
- Hysterises thresholding (on gradient)
  - All pixels that satisfy zero-crossing and gradient > T_{hi} are edges
  - All accept all pixels that in the T_{hi} connected components using T_{lo}
  - Lower gradient edges can be brought in if they are connected to high-gradient edges

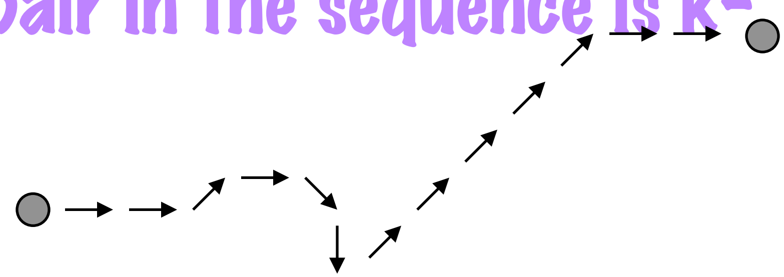


Connectivity In Discrete Domains

- Neighborhood relationship
  - 4 or 8 neighbors in 2D
  - More complex in 3D
  - Symmetric



- K-connected path between two pixels
  - Sequence of (unique) pixels that begins on (a) and ends at (b) and for which each consequetive pair in the sequence is k-connected



Connected Component

- A subset S of pixels in an image such that
  - For any pair of pixels [(a),(b)] E S, there exists a k-connected path between (a) and (b).
- Usefulness: find connected components S in an image that satisfy some conditions
  - Pixel conditions P(a)
    - Threshold or some other grey-level test
  - Region conditions R(S)
    - Aggregate quantities such as size, length, etc.
- Algorithm: flood fill

Flood Fill

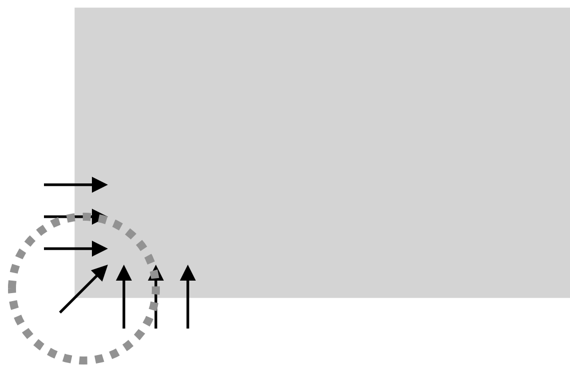
- Highlight regions in an image
- "Test(i, j)" - is value at pixel (i,j) between a and b
- Inputs: seed, values a&b
- Data structures: input array, output array, list of grid points to be processed

A Simple Algorithm: Flood Fill

- Empty list, clear output buffer (=0)
- Start at seed (i,j) and if Test(i,j), put (i,j) on list and mark out[i,j]=1
- Repeat until list of points is empty:
  - Remove point (i,j) from list
  - (Loop) for all 4 neighbors (i',j') of (i,j)
    - If (Test(i',j') and out[i,j]) put (i',j') on list and mark out[i',j']=1
- Properties
  - Guaranteed to stop
  - Worst case run time

Corners

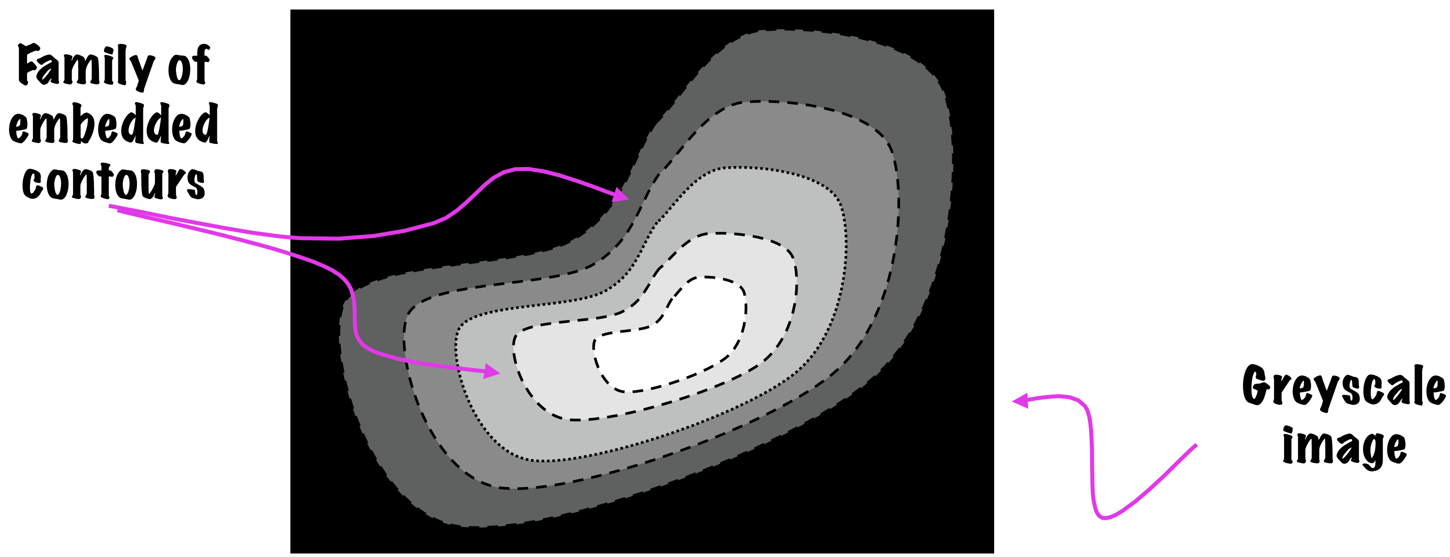
- Places where gradient directions vary (at some scale)
  - + high gradient (edge) possibly



- How to capture this...

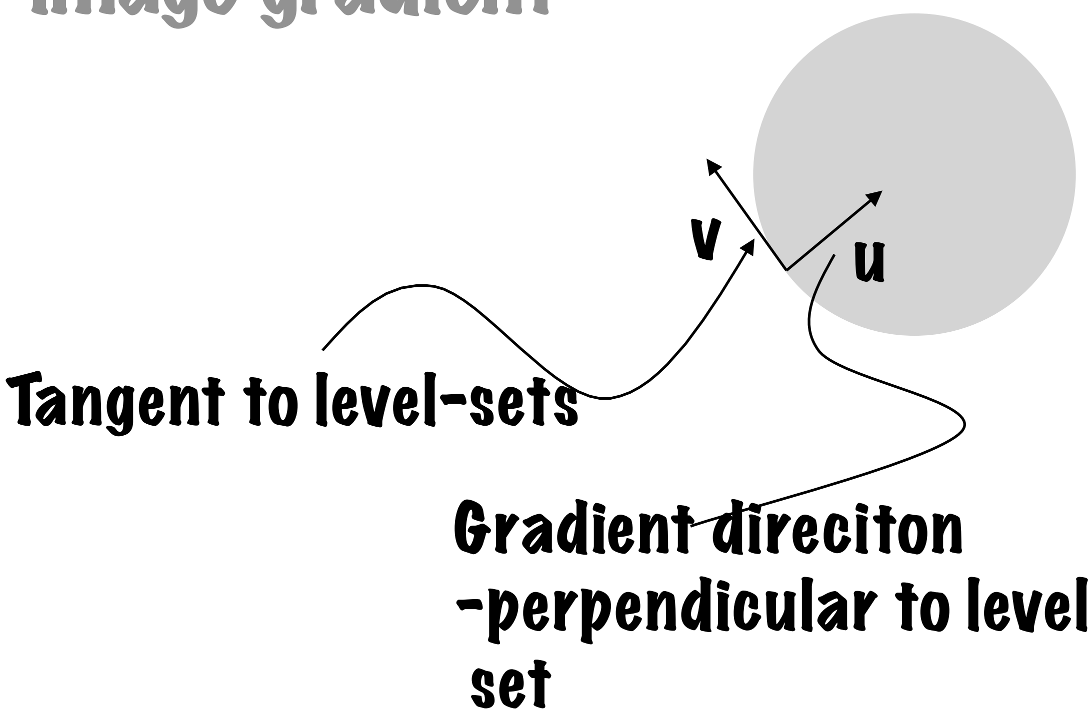
Level Sets of an Image

- Level set - set of points f(x, y)=k
  - "isophote", "isocontour", "isosurface"(3D)



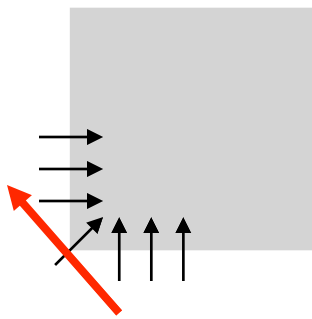
Gauge Coordinates

- Local coordinate system aligned with image gradient



Corners - differential approach

- High derivative of normal in direction perpendicular to gradient



$$\mathrm{cornerness} = |\nabla \mathrm{f}| \frac{\mathrm{d}\vec{\mathrm{n}}}{\mathrm{dv}} \cdot \vec{\mathrm{n}}$$

$$\vec{n} = \frac{\nabla f}{|\nabla f|}$$

- This is level-set/isophote curvature-κ

$$|\nabla f|\kappa = |\nabla f| \nabla \cdot \frac{\nabla f}{|\nabla f|}$$

Corners - differential approach

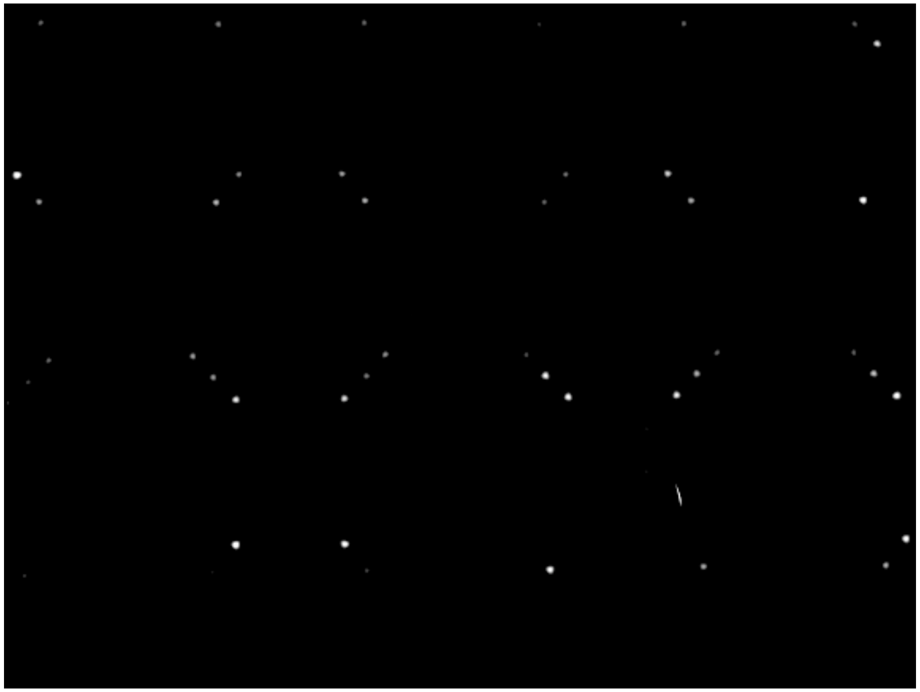
- Non-maximal supression

$$\frac{d\kappa}{dv} = 0$$ + Canny edge

Has a “sign” convex/concave

- Or local max of $|\nabla f||\kappa|$
  - Compare to neighbors

Differential Detector Example



Finding Local Point Maxima

- Zero crossings of x and y derivatives
- Pixel greater than its neighbors (4 or 8 connected)
- Threshold and find center of mass of connected component

VISPack Code

```
im = im.gaussDiffuse(3.0);
im_dx = im.dx();
im_dy = im.dy();

grad_mag = (im_dx.power(2) + im_dy.power(2)).sqrt();

curve = (im.dx(2)*im_dy.power(2) +
           im.dy(2)*im_dx.power(2) -
           2.0*im_dx.dy()*im_dx*im_dy).abs()
           /(grad_mag.power(2) + (float)1.0e-2);
curve = curve.setBorder(0.0f, 2);
curve = grad_mag*curve.abs();
```

Neighborhood Statistics
Harris (88) or Plessey Detector

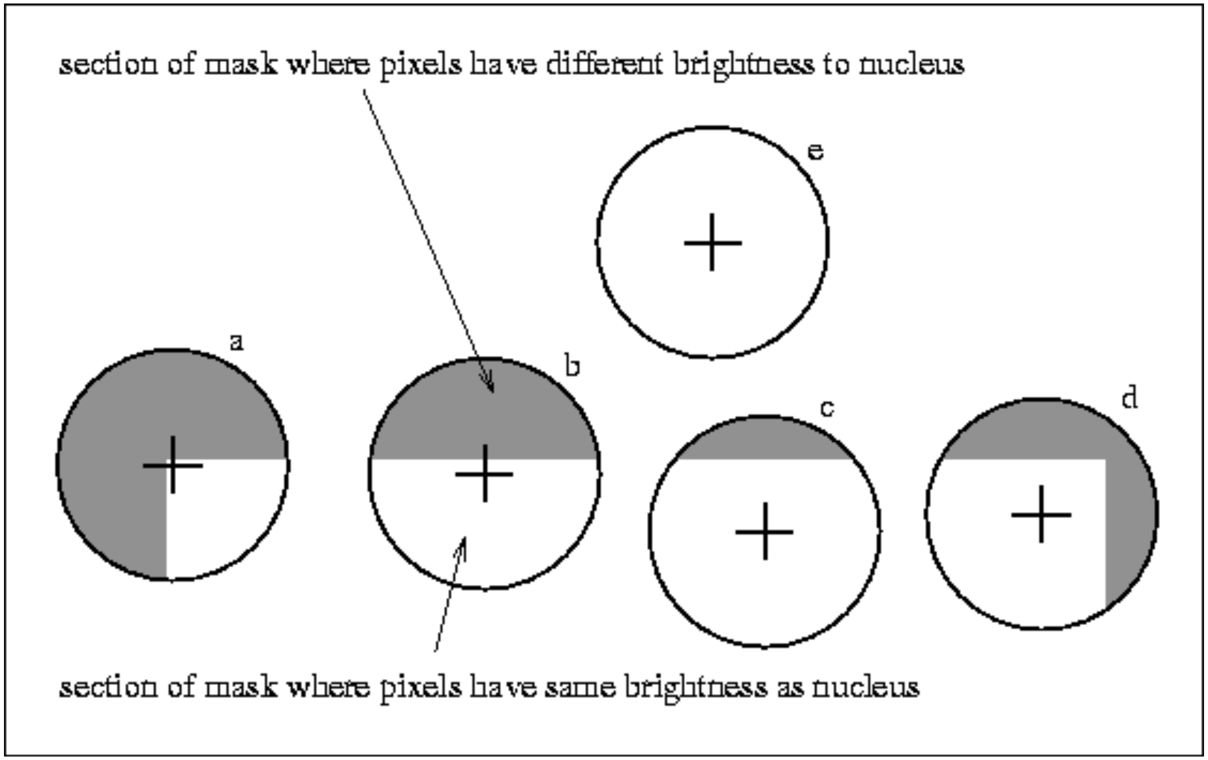
- Covariance of image gradient in neighborhood

$$M = \int G(x,y) \begin{pmatrix} f_x f_x & f_x f_y \\ f_x f_y & f_y f_y \end{pmatrix} dxdy$$

$$C = \det(\mathrm{M}) - \mathrm{k}\mathrm{Tr}(\mathrm{M})^2 = \alpha\beta - \mathrm{k}(\alpha + \beta)^2$$

SUSAN Corner Detector
"Smallest Univalue Segment Assimilating Nucleus"

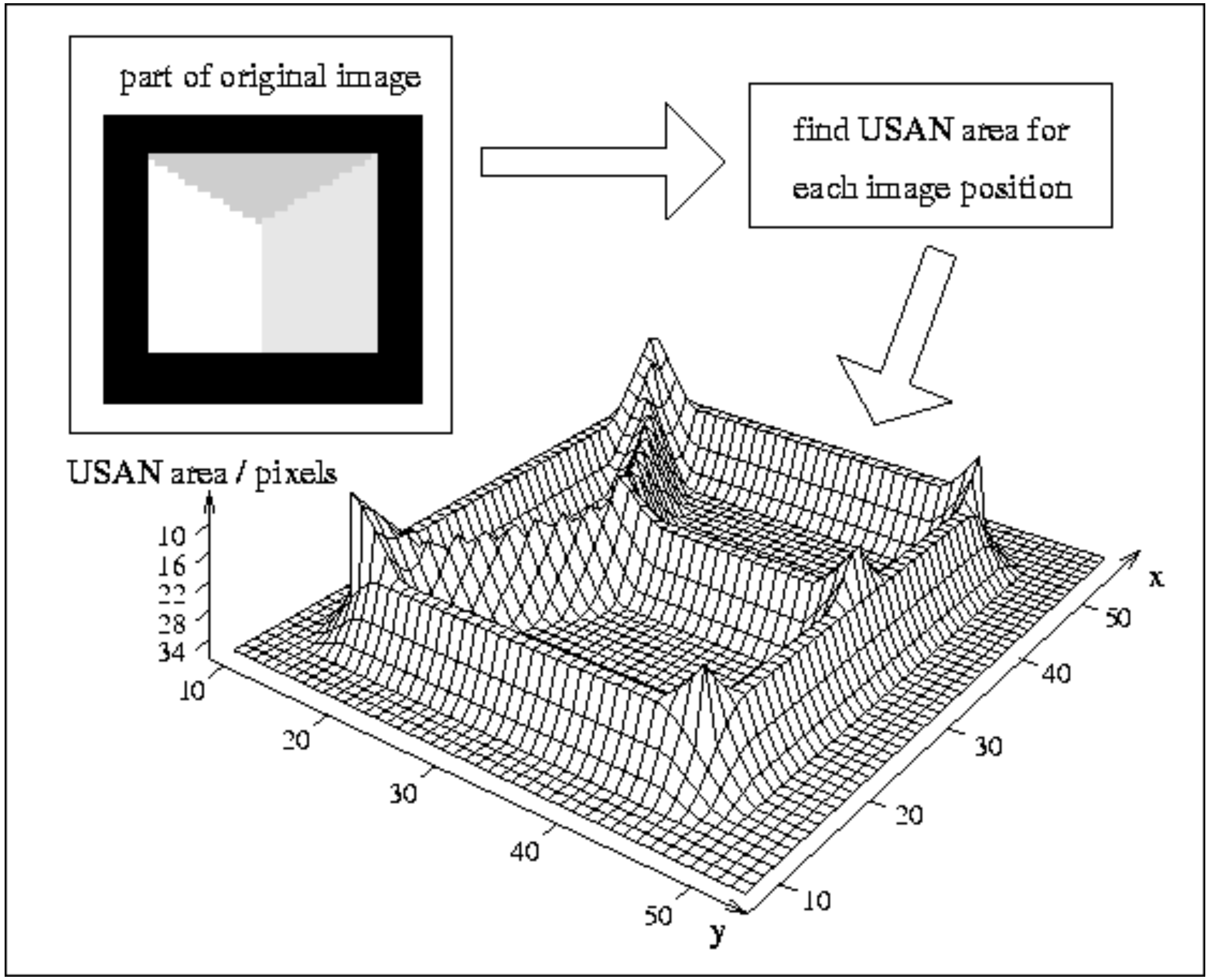
- Smith and Brady 1995
- Threshold pixels in nieghborhood (likeness to center) and compute ratio of areas



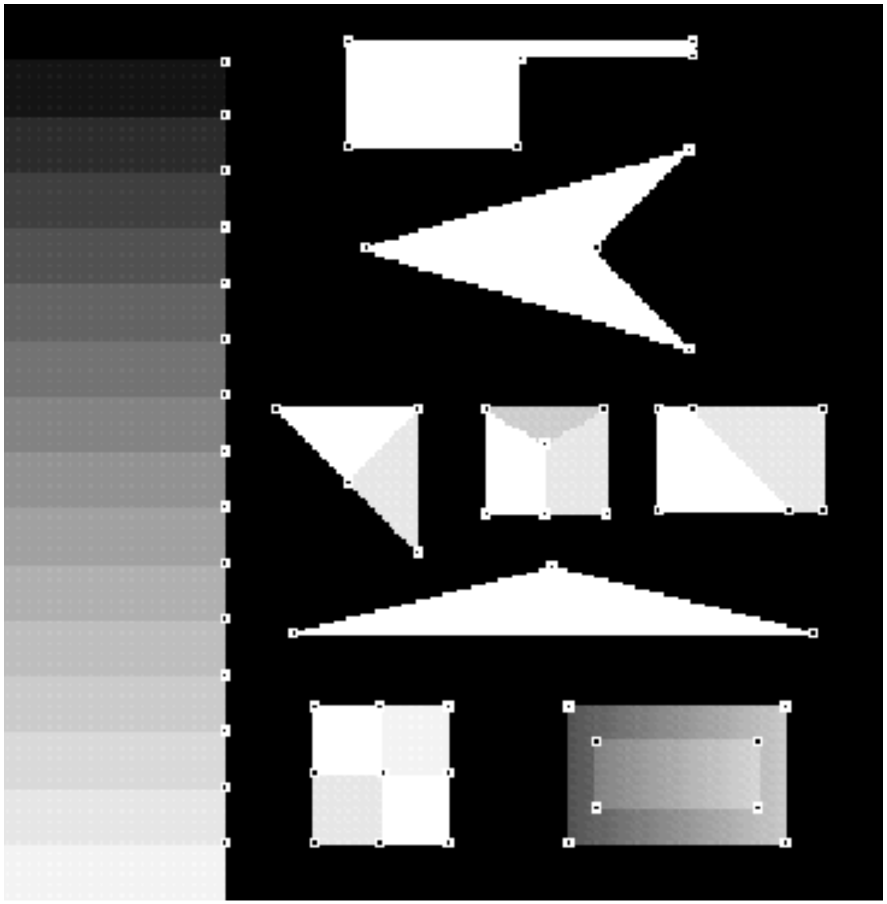
SUSAN Detector

Include local or directional

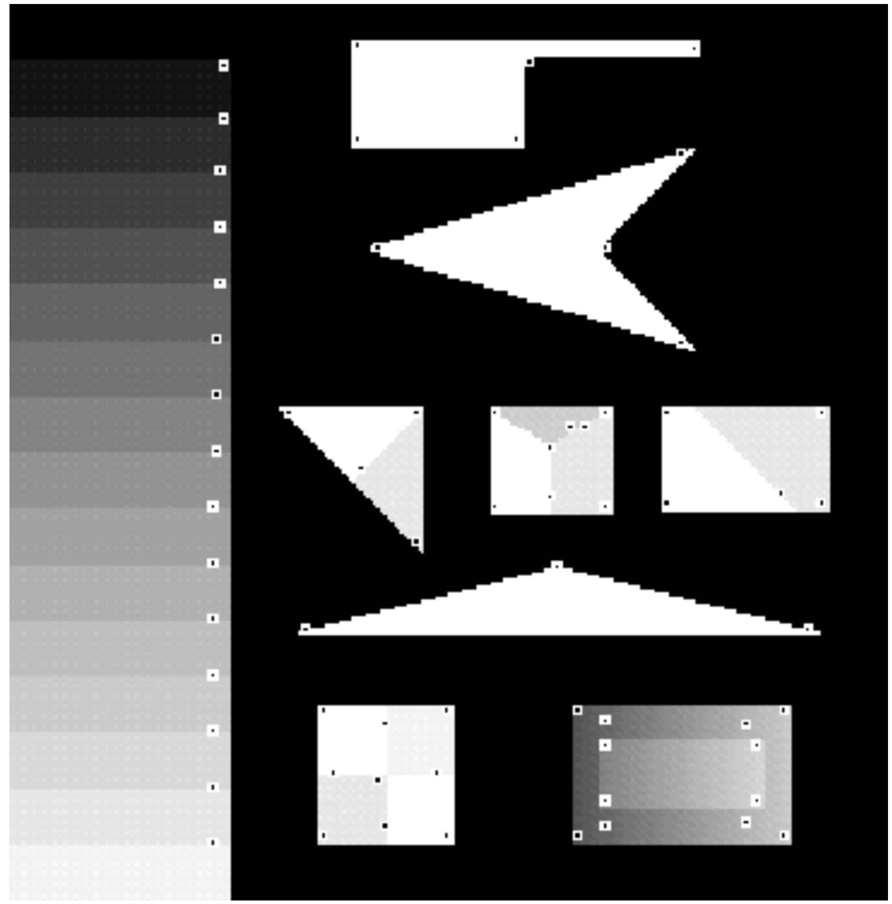
nonmaximal supression



SUSAN vs Plessey Detector



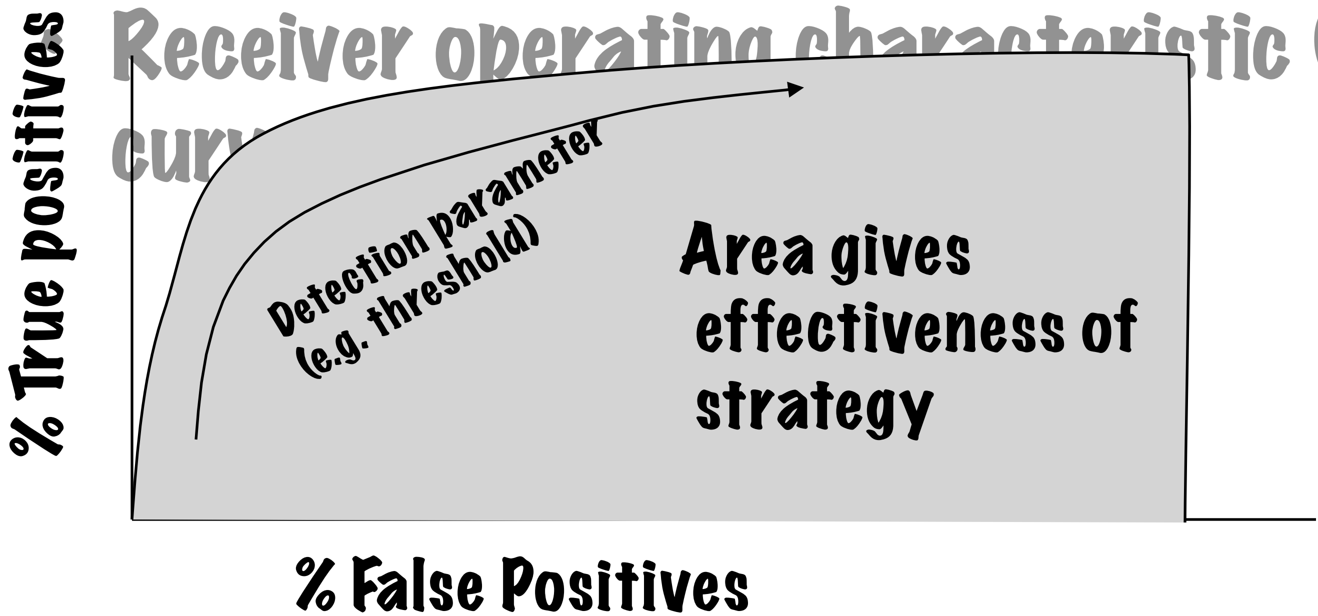
SUSAN



Plessey

Detection Performance

- False positives - something that wasn't real
- False negatives - missing something
- Location accuracy
- Receiver operating characteristic (ROC) curve



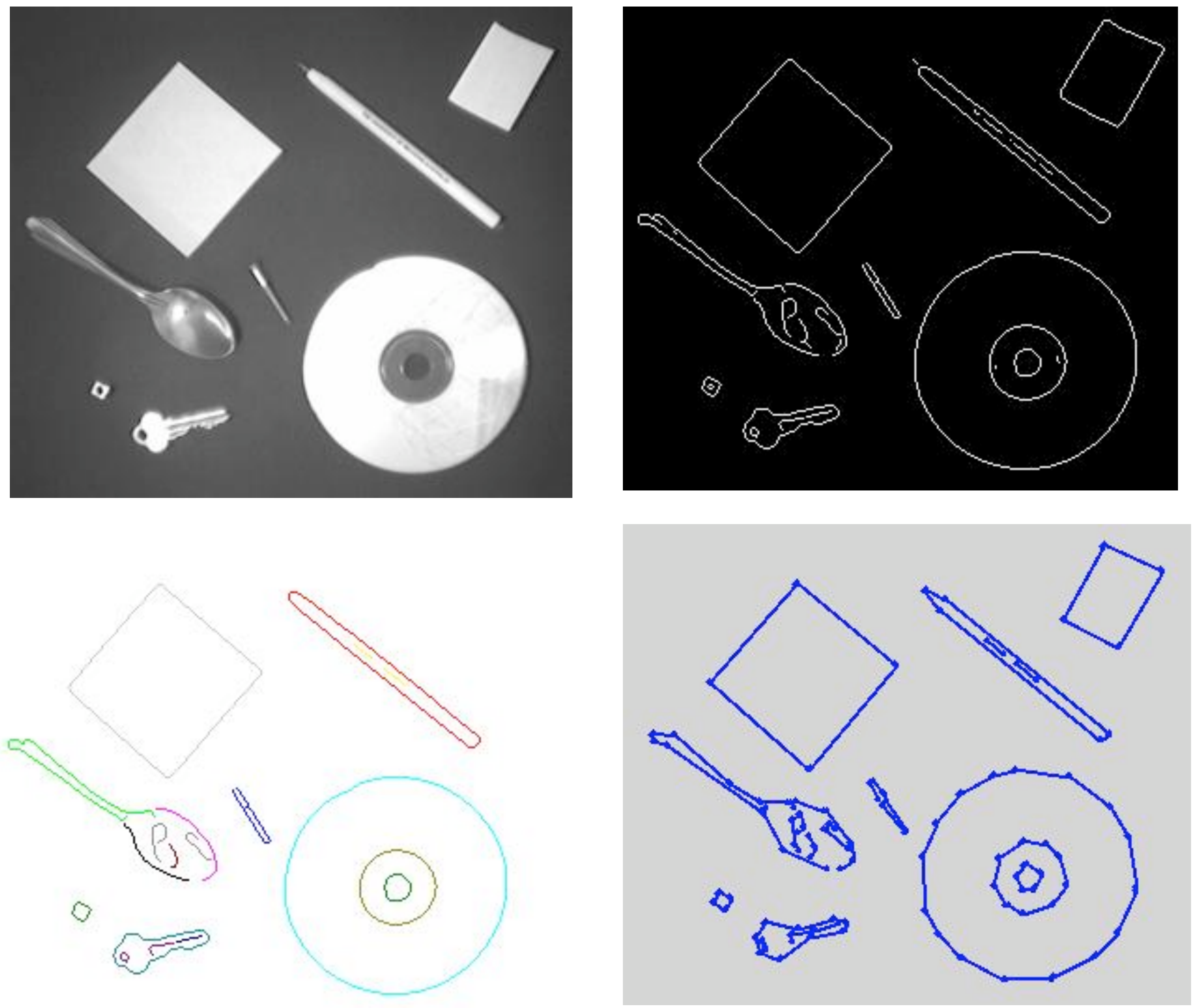
What do we do with features?

- Edge linking and grouping
- Correspondences
- Shape detection - Hough transform

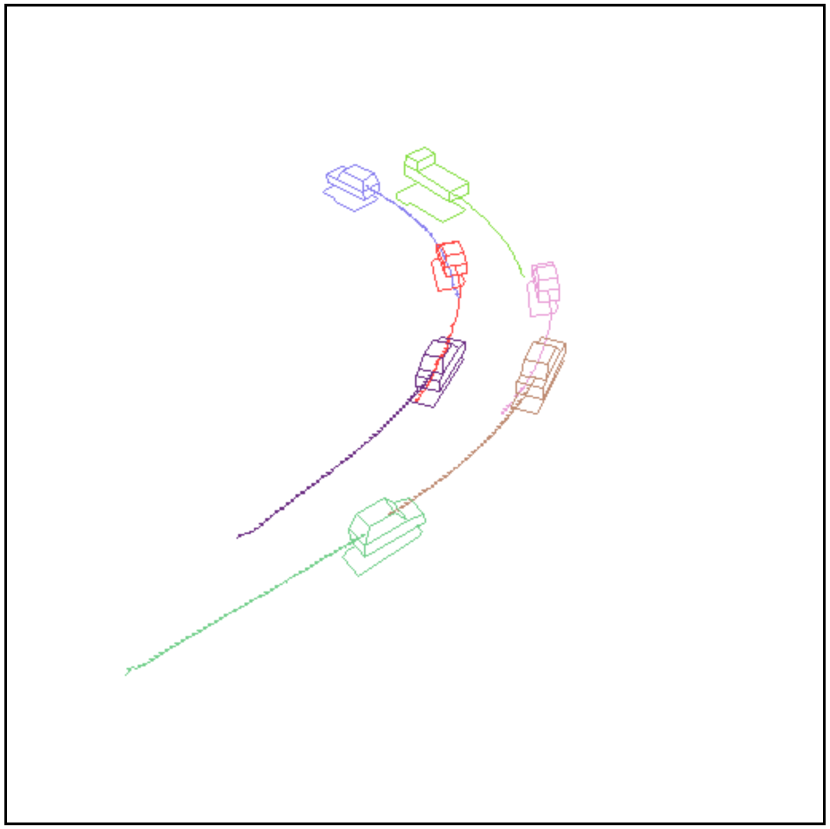
Edge Linking and Grouping

- Many man-made object have flat sides
  - Photographs have edges corresponding to object parts
- Strategies
  - Connected components (bridge gaps)
  - Look for curves that meet criteria
    - Line segments (fit to line)
    - Circular segments

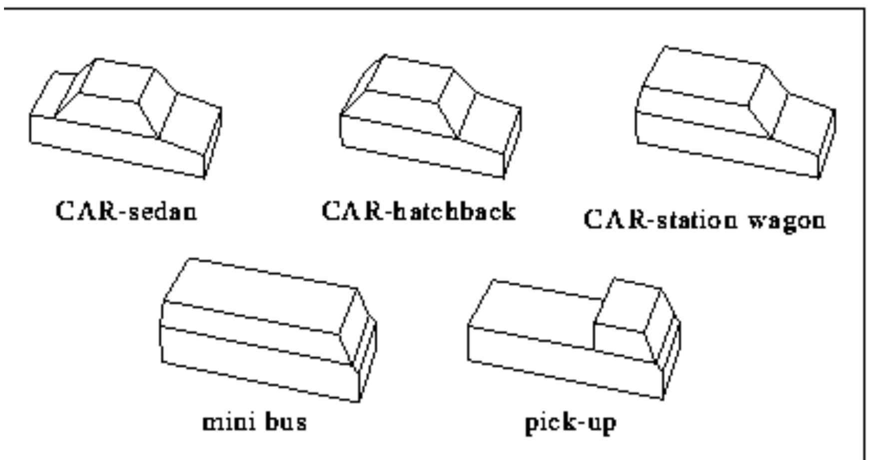
Edge Linking - P. Kovesi

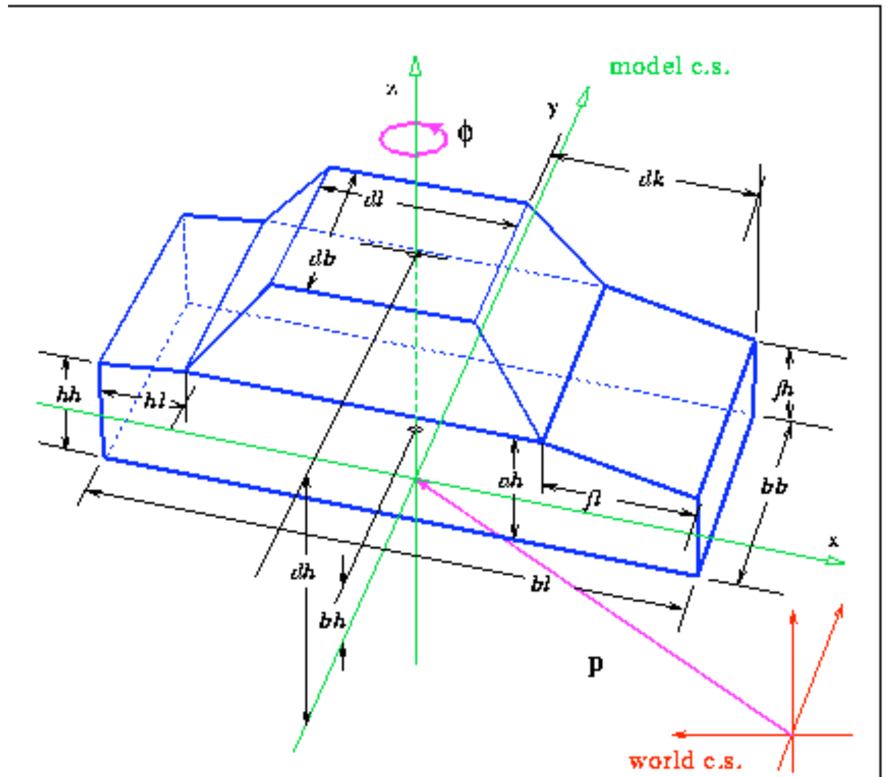


Vehicle Tracking-D. Kohler

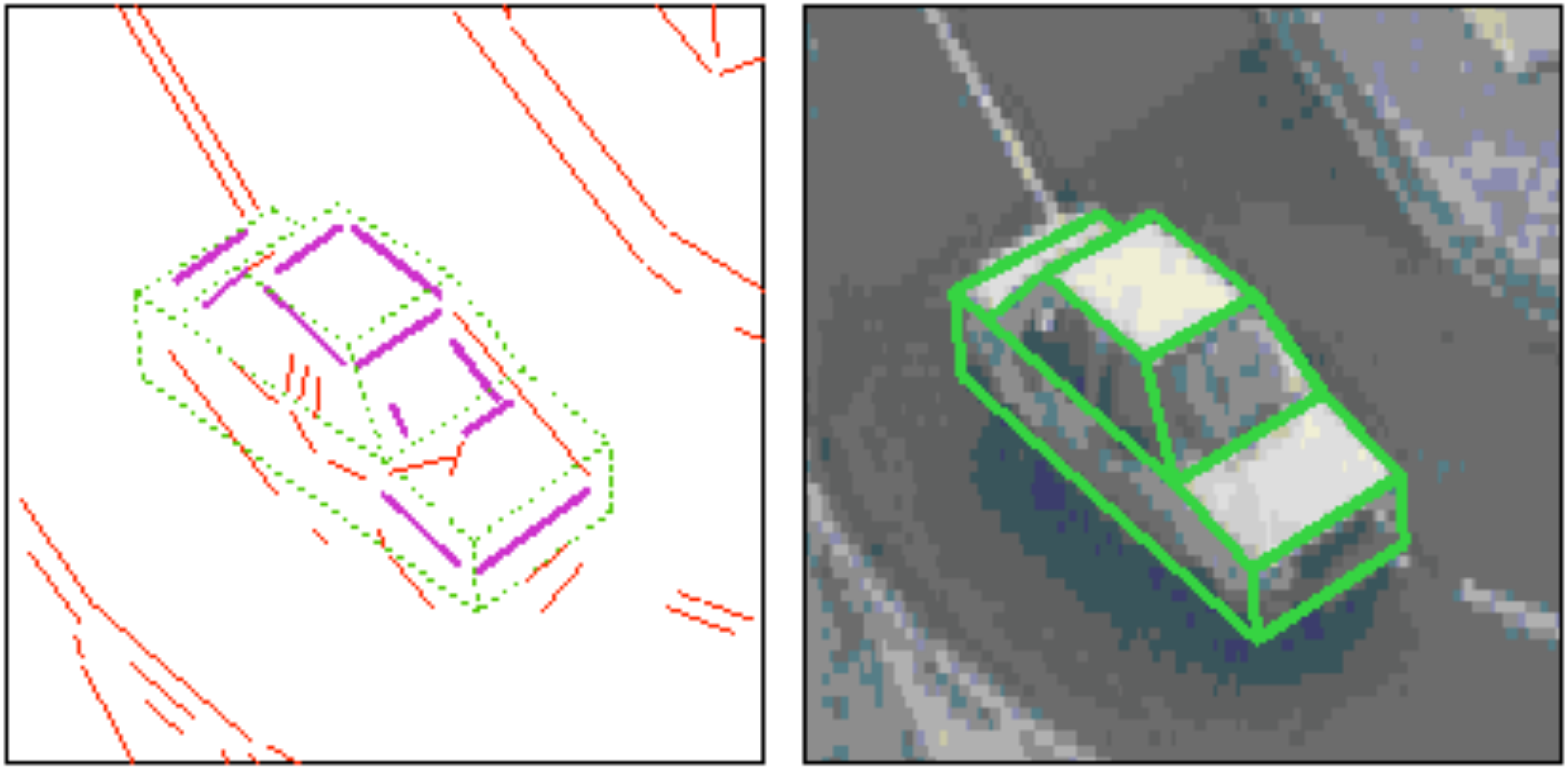


Vehicle Models





Align Model Based on Edge Correspondences



Correspondences

- Hard problem
  - Two images containing N pts -> exponential time
  - Also, false positives/negatives
- Anything better than exhaustive search?

Correspondences with Signatures

- Establish signature for each detected point/line
  - Invariant
- Try combinations starting with best signature matches
  - Sort correspondences by signature match
- Signatures
  - Local greyscale histogram
  - Spin images (Heckbert) - average greyscale as a function of distance
  - Correlation (not invariant)

Correspondences-RANSAC "Random Sample Consensus"

- Outliers a problem

1. Choose M correspondences at random
   - Very few, minimum for establishing transformation
2. Compute transformation
3. Find out how well these matches predict the other correspondences (threshold on distance)
4. Repeat a lot of times
   - Choose small random set which is best predictor
5. Establish inliers
6. Compute transformation on all inliers

Hough Transform

- See book and notes