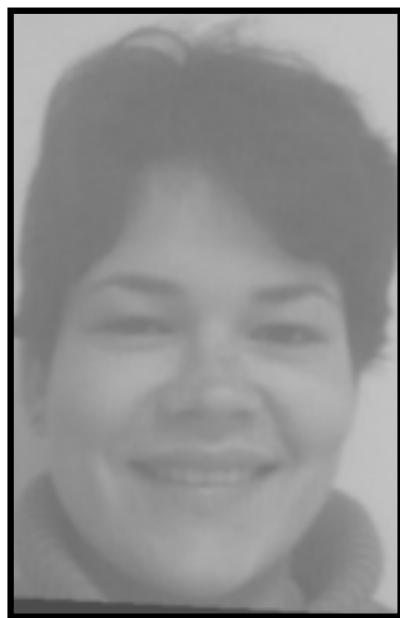


# DELINEATION

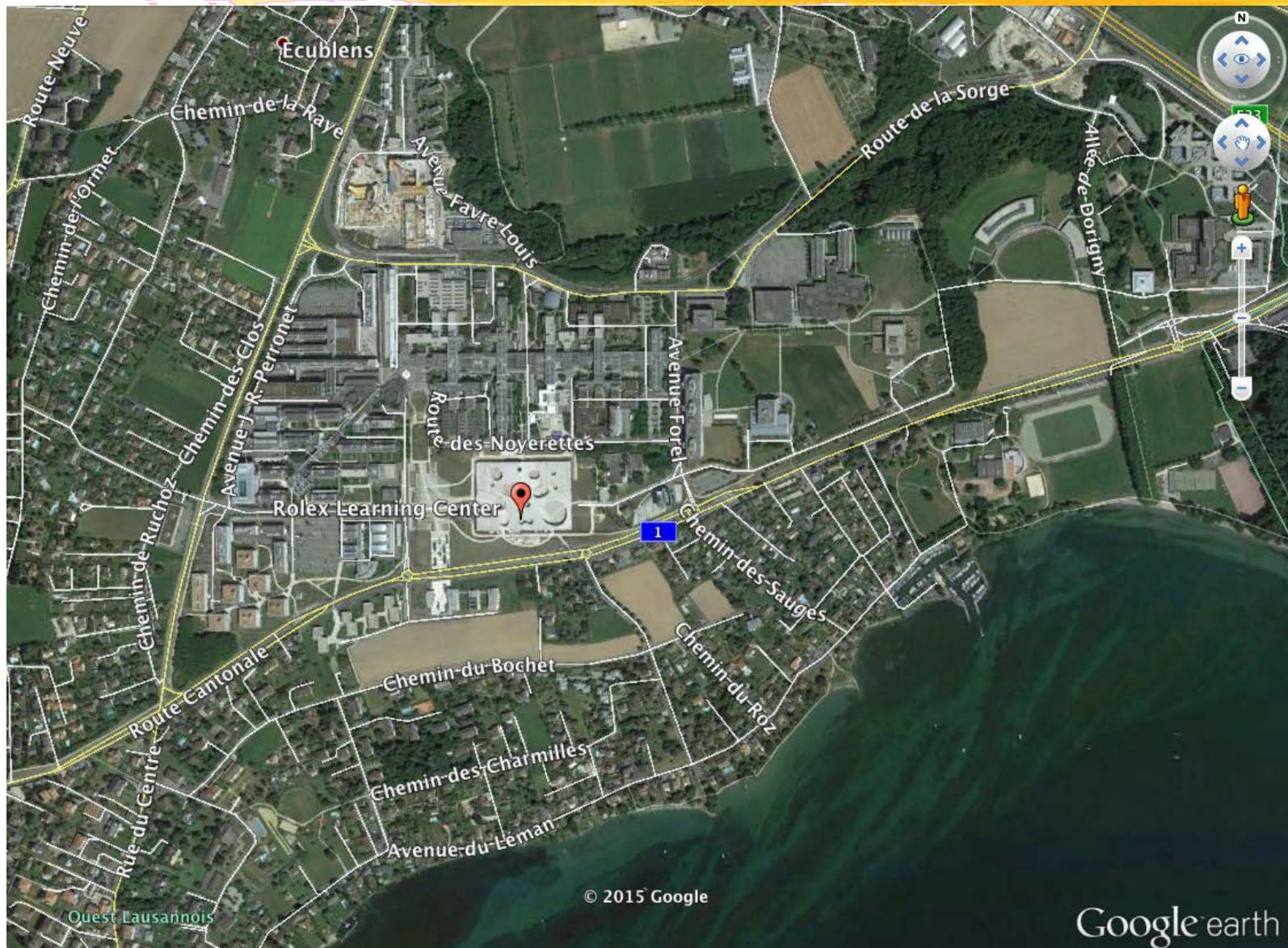
- Dynamic Programming
- Deformable Models
- Hough Transform
- Graph Based Approaches



# FROM LOCAL EDGES TO OUTLINES



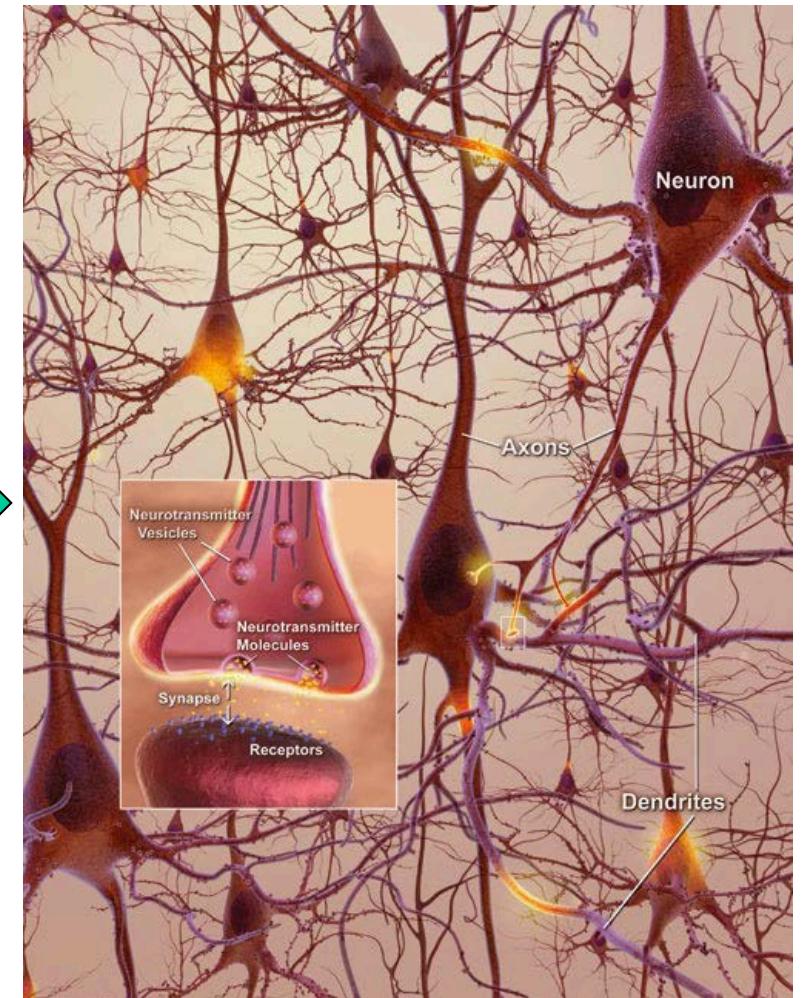
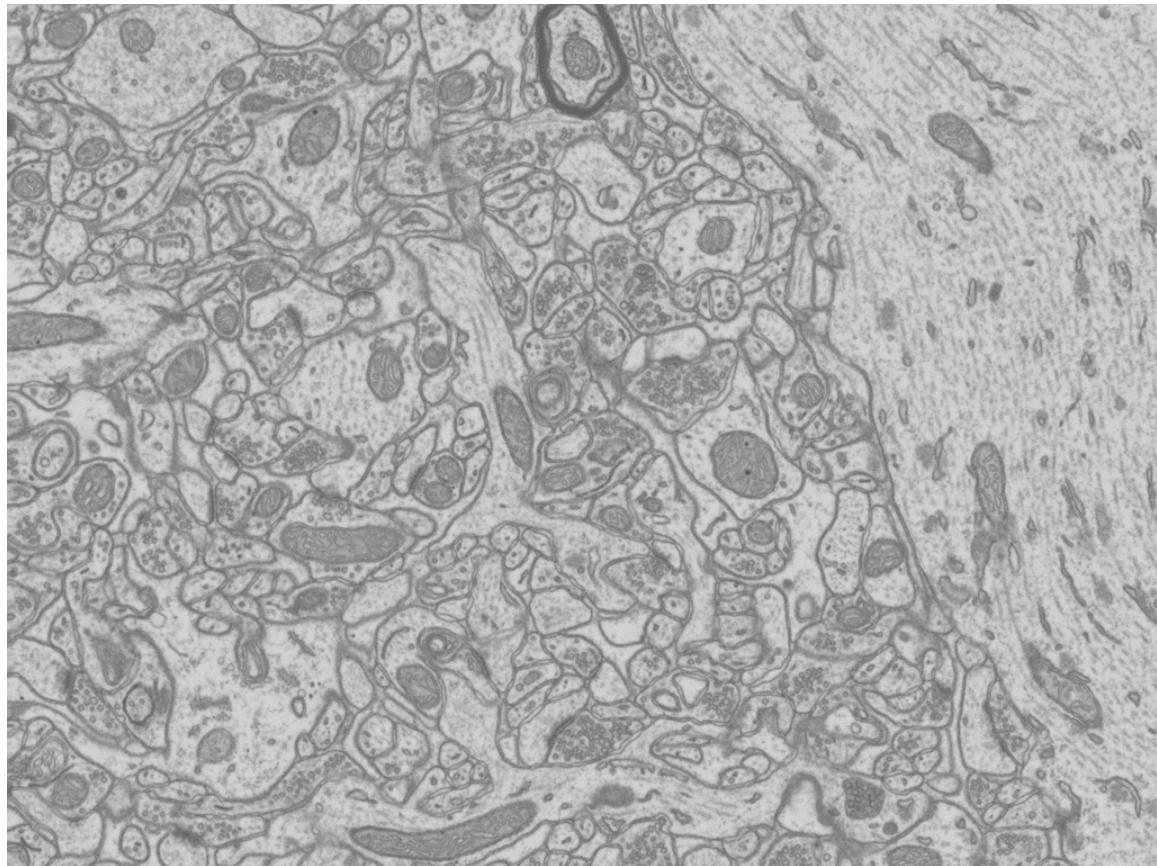
# MAPS AND OVERLAYS



# ROAD IMAGE



# CONNECTOMICS



# ANALOGY



Low level processing

- Uses Deep Nets to find the most promising locations to focus on.

High level processing

- Performs Tree based search when possible.
- Relies on reinforcement learning and other ML techniques to train.

# TECHNIQUES



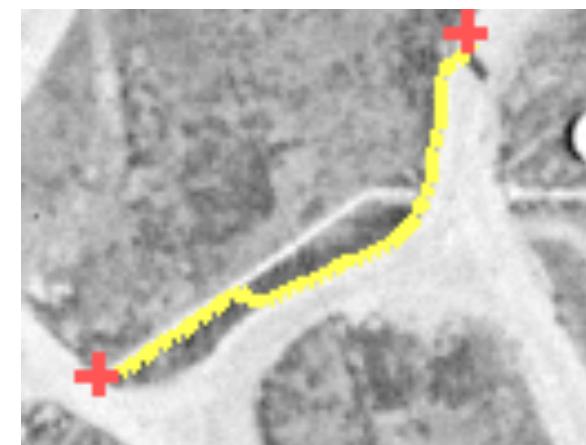
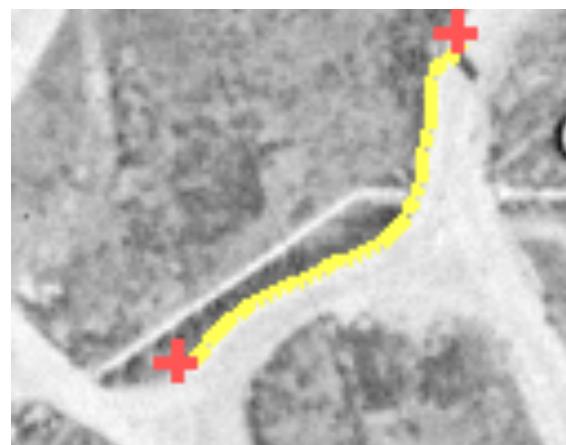
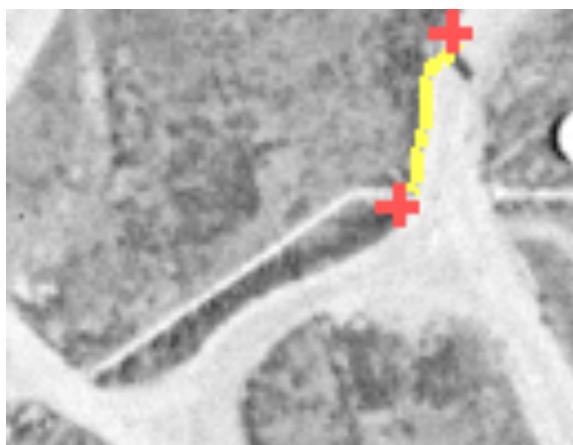
Semi-Automated Techniques:

- Dynamic Programming
- Deformable Models

Fully Automated Techniques:

- Hough Transform
- Graph Based Approaches

# INTERACTIVE DELINEATION

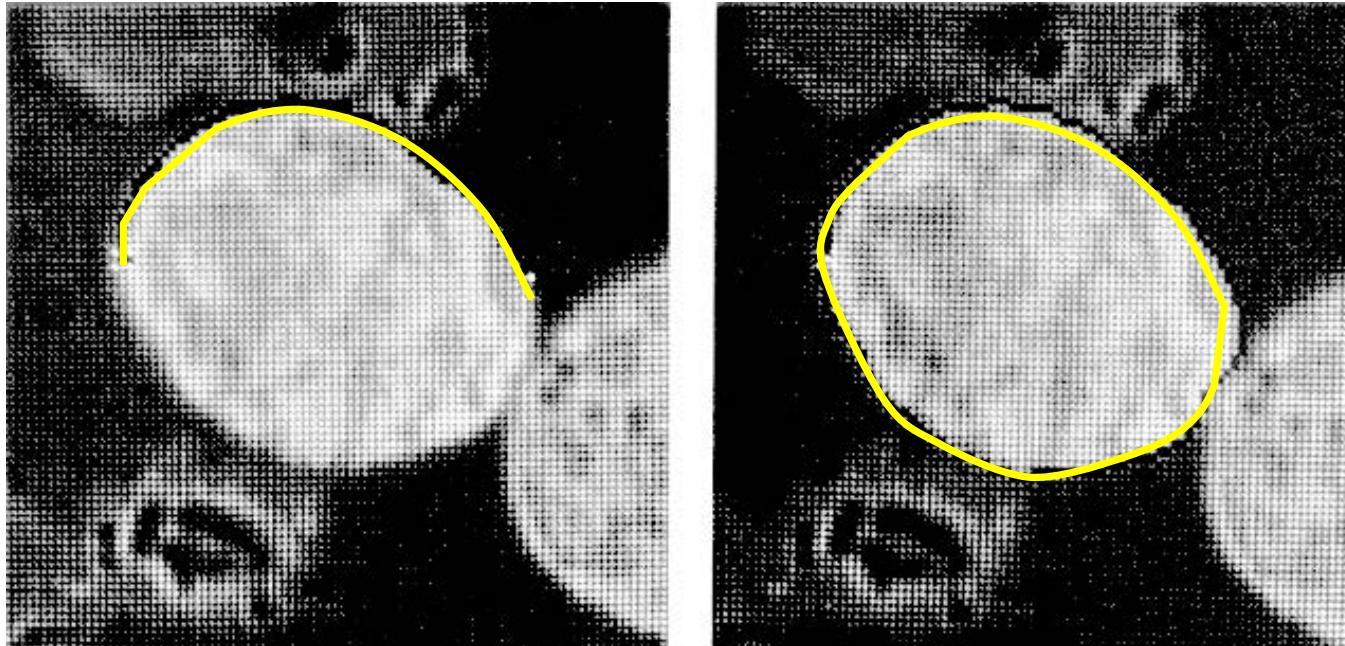


# LIVE WIRE



Mortensen and Barrett,  
SIGGRAPH, 1995

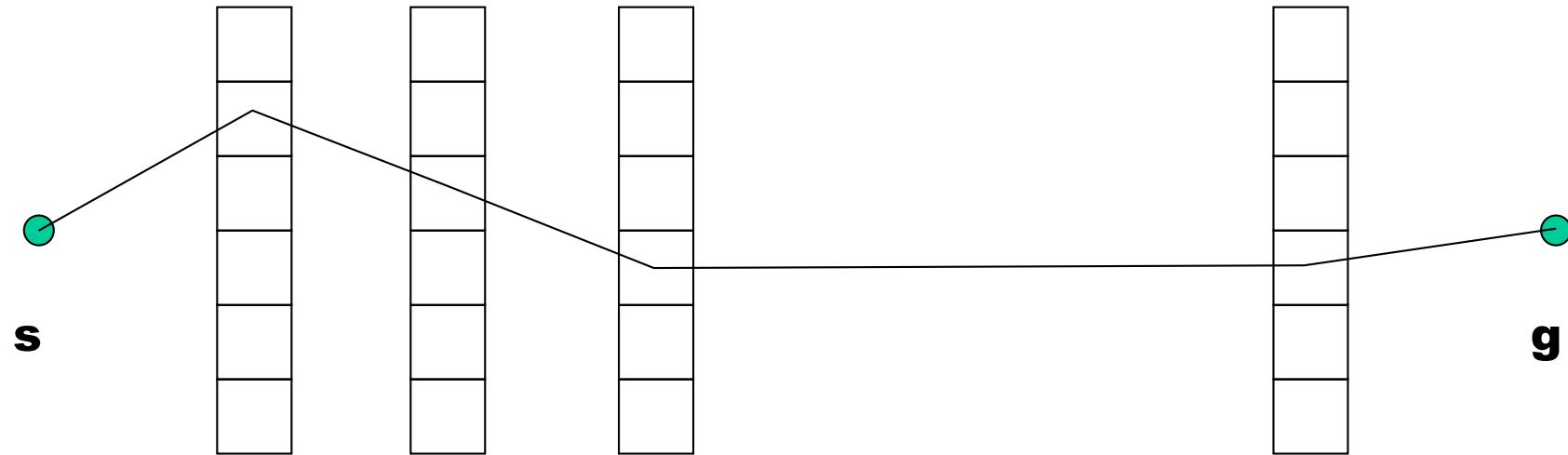
# DYNAMIC PROGRAMMING



$$h(x_1, x_2, \dots, x_n) = - \sum_{k=1}^n g(x_k) + \sum_{k=1}^{n-1} q(x_k, x_{k+1})$$

$$q(x_k, x_{k+1}) = \text{diff}[\phi(x_k), \phi(x_{k+1})]$$

# 1D DYNAMIC PROGRAMMING



- N Locations
  - Q Quantized values
- Global optimum  $O(NQ^2)$

# 1D DYNAMIC PROGRAMMING

To find

$$\min_{x_i} h(x_1, x_2, \dots, x_n)$$

where

$$h(x_1, x_2, \dots, x_n) = h(s, x_1) + \sum_{i=1}^{n-1} h(x_i, x_{i+1}) + h(x_n, g)$$

define

$$f_1(x_2) = \min_{x_1} (h(s, x_1) + h(x_1, x_2))$$

$$f_2(x_3) = \min_{x_2} (h(x_2, x_3) + f_1(x_2))$$

$$\vdots \quad \vdots \quad \vdots$$

$$f_{n-1}(x_n) = \min_{x_{n-1}} (h(x_{n-1}, x_n) + f_{n-2}(x_{n-1}))$$

$$\Rightarrow \min h(x_1, x_2, \dots, x_n) = \min_{x_n} (h(x_n, g) + f_{n-1}(x_n))$$

# 2D DYNAMIC PROGRAMMING

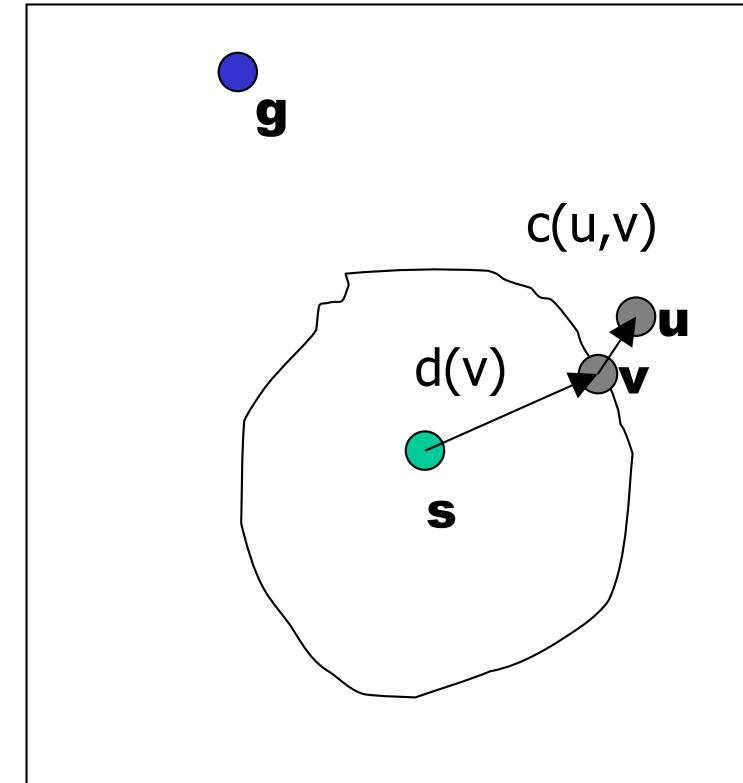
## Notations:

s Start point

L List of active nodes

$c(u,v)$  Local costs for link  $u \rightarrow v$

$d(v)$  Total costs from  $v$  to  $s$



# DIJKSTRA'S ALGORITHM

Initialization :

$$d(s) \leftarrow 0 \text{ and } d(u) \leftarrow \infty \text{ for } u \neq s$$

$$T = \emptyset$$

$$v = s$$

Loop until goal is reached :

$$T \leftarrow T \cup \{v\}$$

for all  $v \rightarrow u$  edges such that  $u \notin T$

$$\text{if } d(v) + c(v,u) < d(u)$$

$$d(u) \leftarrow d(v) + c(v,u)$$

end

end

$$v = \operatorname{argmin}_{w \notin T} d(w)$$

Maintain a sorted list of paths

# LIVE WIRE



- Sorting is the expensive operation. Normally  $n \log(n)$ , but can be reduced to  $\log(n)$  if all costs are integer costs
- Local costs computed using gradient:

$$c(u,v) = 255 - \frac{1}{2} (g(u) + g(v))$$

- Diagonal penalized by multiplying cost of non diagonal edges by:

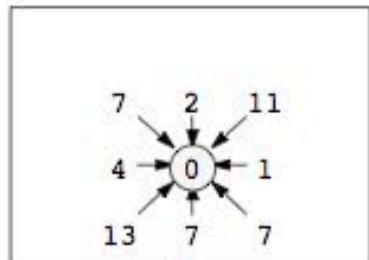
$$\frac{1}{\sqrt{2}} = \frac{5}{7}$$

- Add a constant cost for each edge.

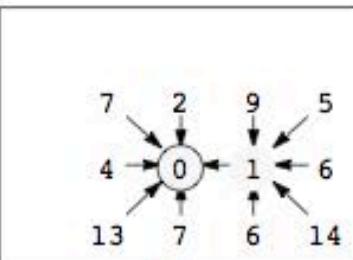
# COST EXPANSION

11	13	12	9	5	8	3	1	2	4	10
14	11	7	4	2	5	8	4	6	3	8
11	6	3	5	7	9	12	11	10	7	4
7	4	6	11	13	18	17	14	8	5	2
6	2	7	10	15	15	21	19	8	3	5
8	3	4	7	9	13	14	15	9	5	6
11	5	2	8	3	4	5	7	2	5	9
12	4	(2)	1	5	6	3	2	4	8	12
10	9	7	5	9	8	5	3	7	8	15

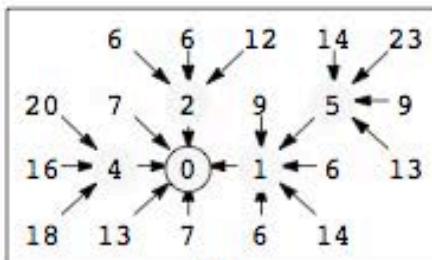
(a)



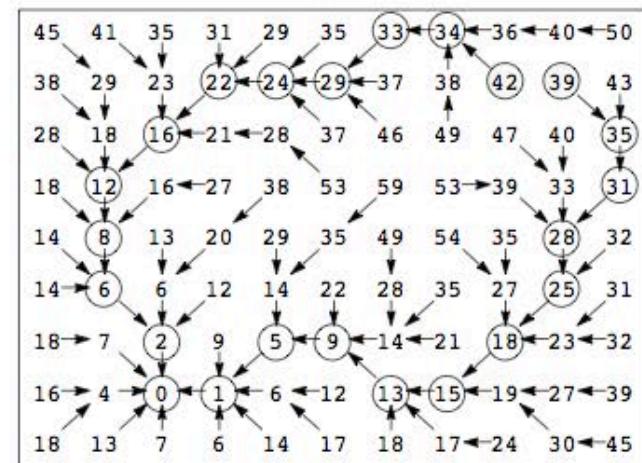
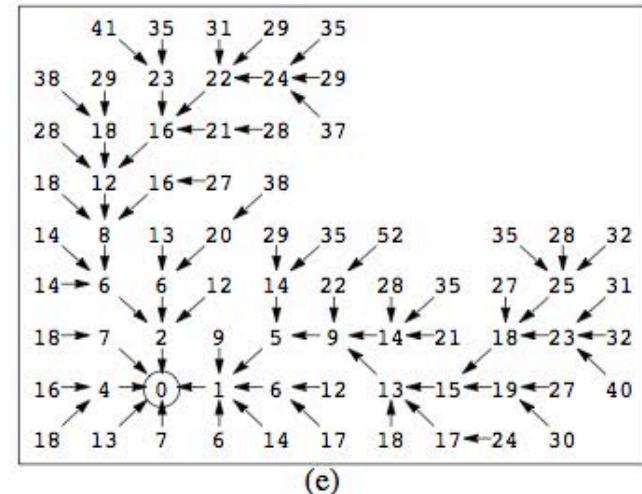
(b)



(c)

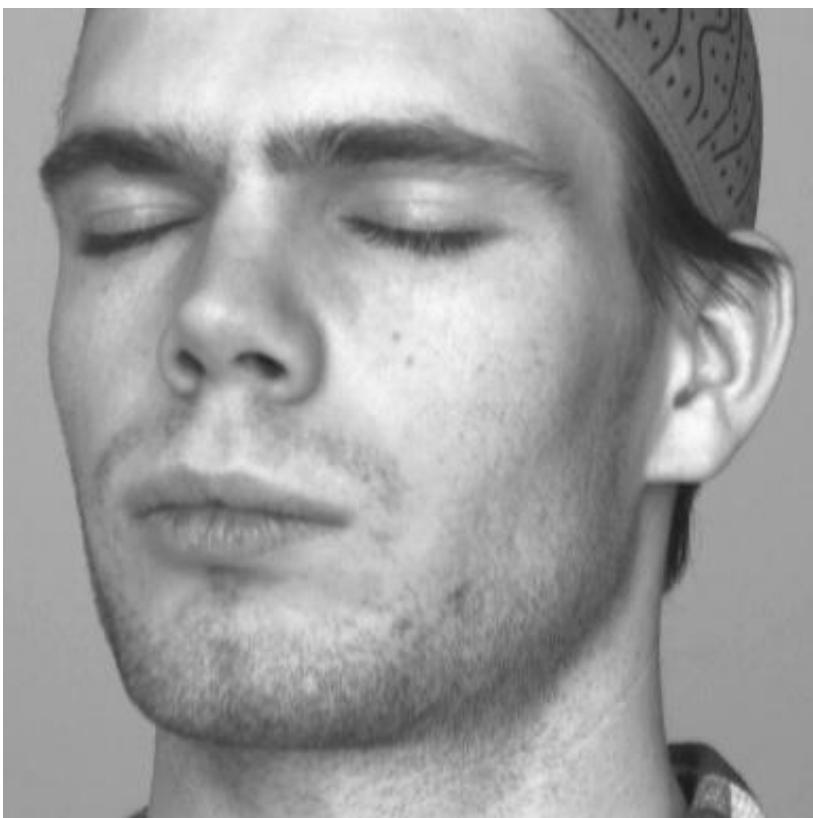


(d)

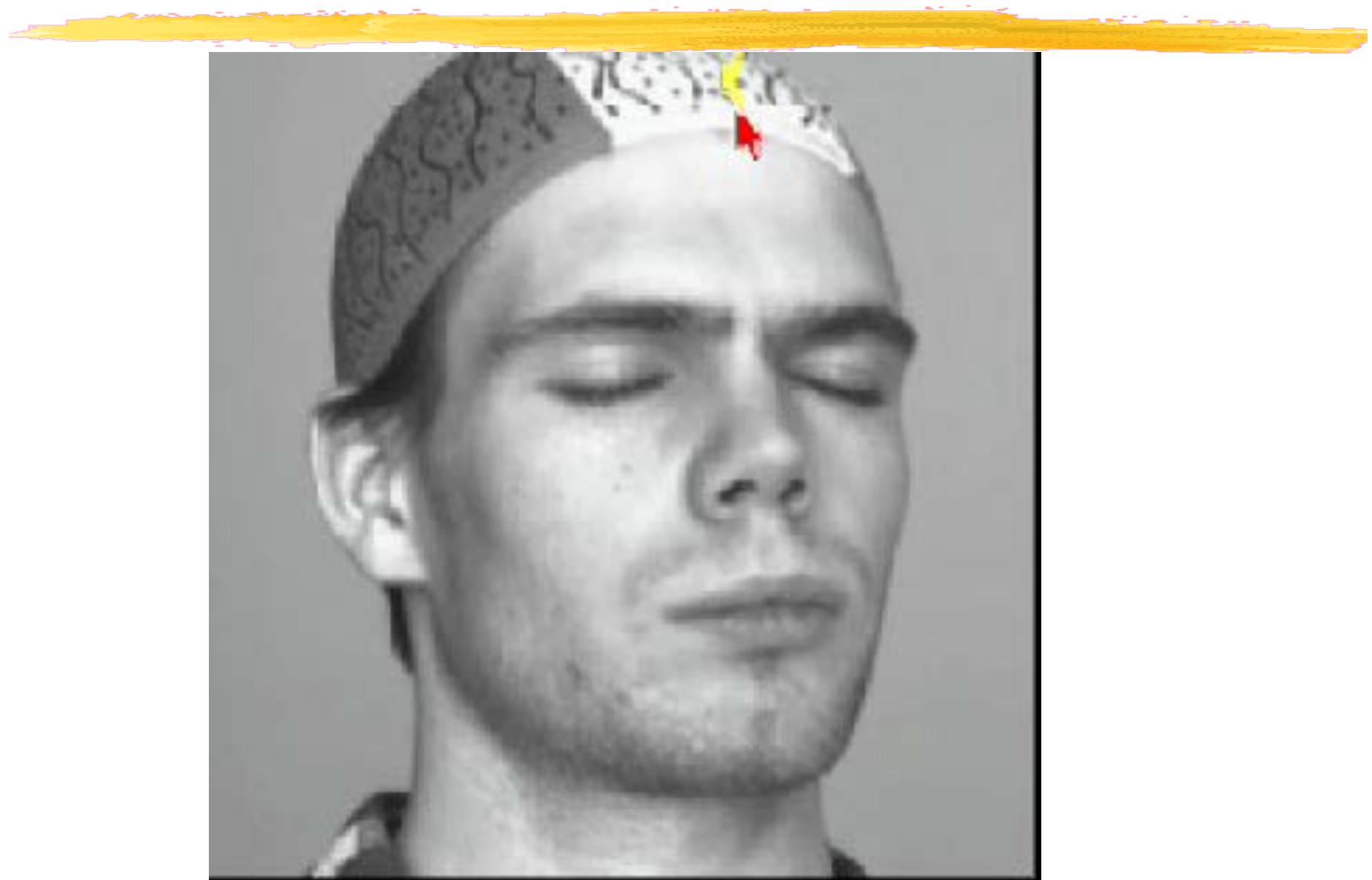


(a) Local cost map. (b) Seed point expanded. (c) 2 points expanded. (d) 5 points expanded. (e) 47 points expanded. (f) Completed cost path-pointer map with optimal paths shown from nodes with total costs 42 and 39.

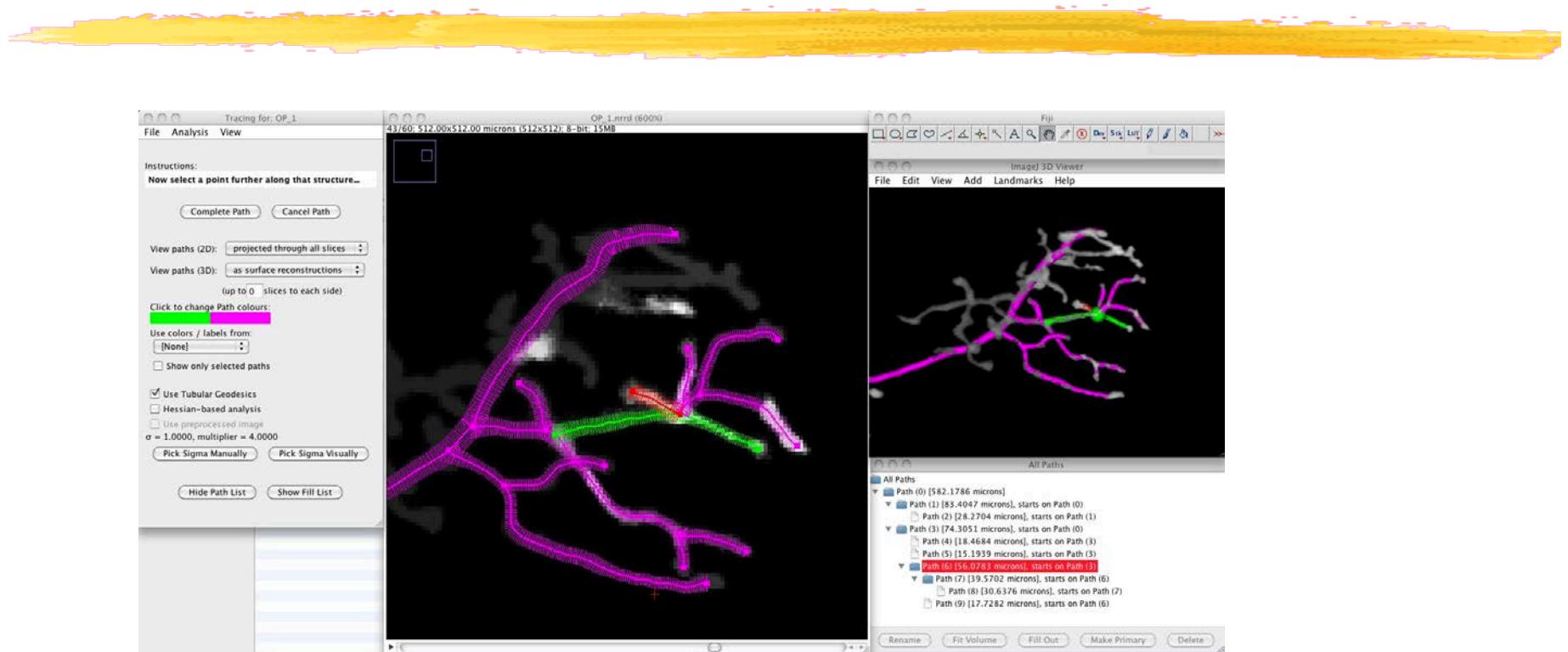
# FACE IMAGE



# LIVE WIRE

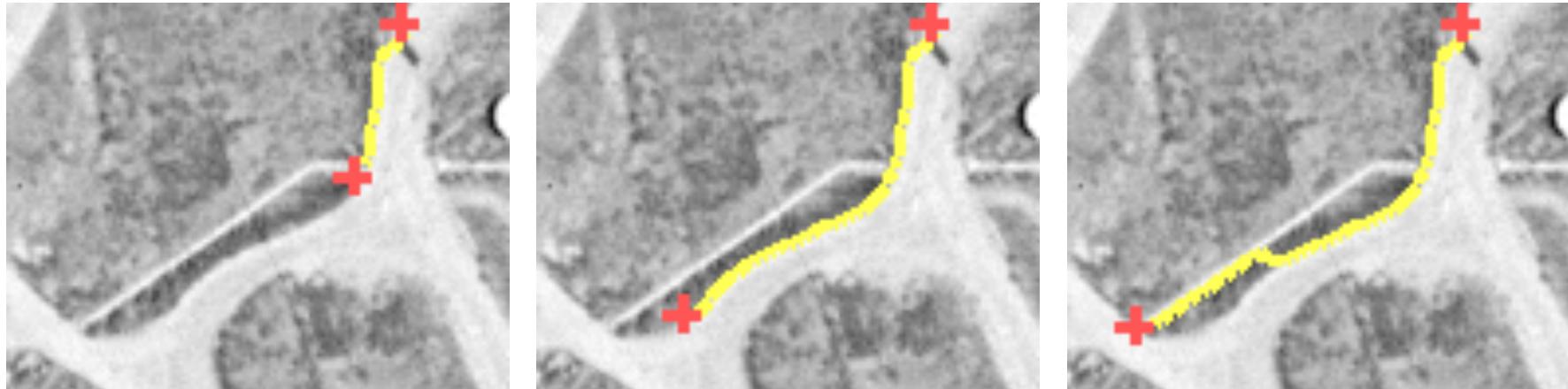


# TRACING NEURONS



- Link user provided points and extracts centerline points.
- Fiji Plug-In downloadable from <http://cvlab.epfl.ch/software/delin>

# OPEN ISSUES



- The “optimal” path is not always the “best” one.
  - Difficult to impose global constraints.
  - The cost grows exponentially with the dimension of the problem.
- > Must often look for local, as opposed to global, optimum using gradient descent techniques.

# TECHNIQUES



## Semi-Automated Techniques:

- Dynamic programming
- Deformable Models

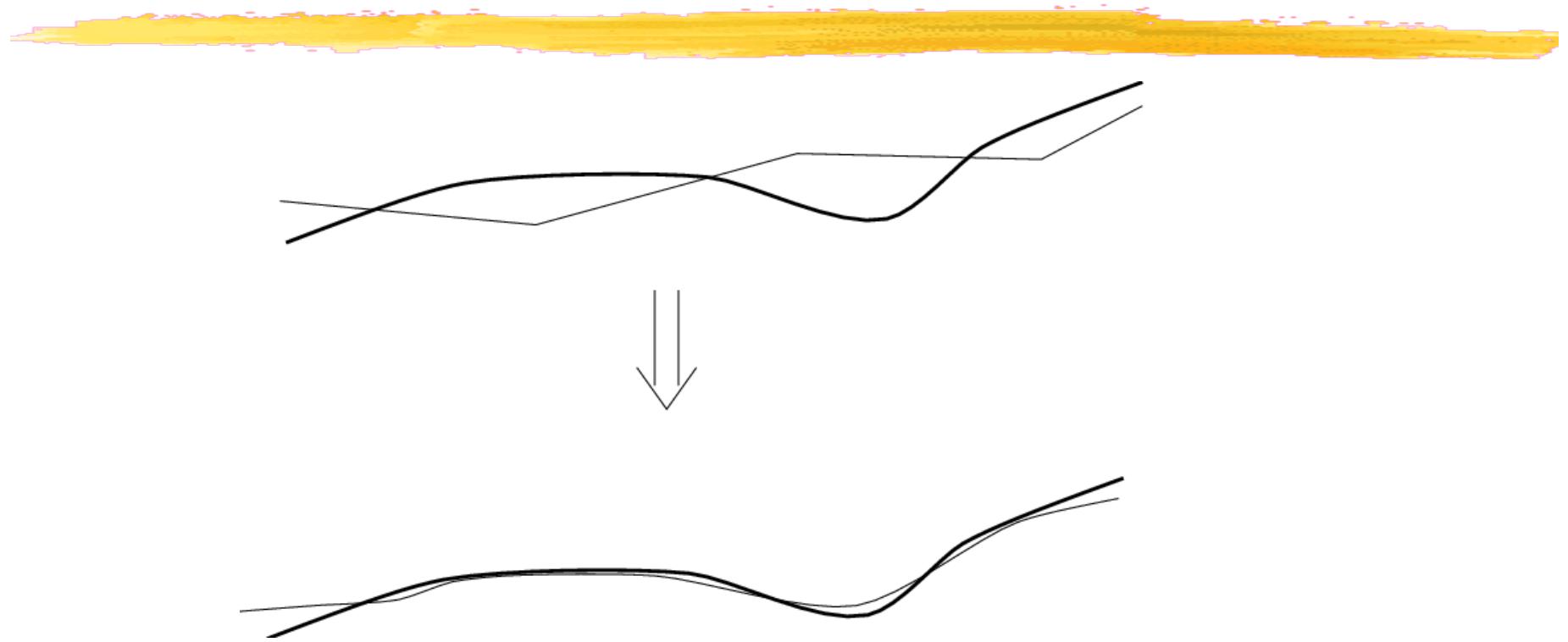
## Fully Automated Techniques:

- Hough transform
- Graph Based Approaches

# SNAKES



# 2—D SNAKE

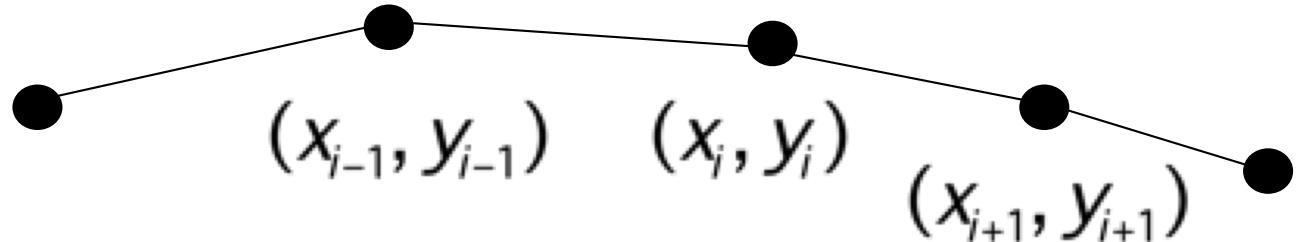


Deformable contours that

- Maximize the gradient along the curve;
- Minimize their deformation energy.

--> Interactive tools for contour detection that can be generalized to handle sophisticated models

# ENERGY LANSCAPE



$$\begin{aligned} E &= 1/L \int_{s=0}^1 (-\lambda G(x(s), y(s)) + \gamma^2(s)) ds \\ &= 1/L \int_{s=0}^1 (-\lambda G(x(s), y(s)) + \frac{\delta^2 x}{\delta s^2} + \frac{\delta^2 y}{\delta s^2}) ds \\ &= 1/L(-\lambda \sum_{i=0}^N G(x_i, y_i) + \sum_{i=1}^{N-1} ((2x_i - x_{i-1} + x_{i+1})^2 + (2y_i - y_{i-1} + y_{i+1})^2) \end{aligned}$$

# MATRIX NOTATION

$$E = E_G + 1/2X^t K X + 1/2Y^t K Y$$

$$X = [x_1, \dots, x_N]^t$$

$$Y = [y_1, \dots, y_N]^t$$

# LOCAL OPTIMUM

$$\frac{\delta E}{\delta X} = \frac{\delta E_G}{\delta X} + KX = 0$$
$$\frac{\delta E}{\delta Y} = \frac{\delta E_G}{\delta Y} + KY = 0$$

But  $K$  is *not* invertible!

# DYNAMICS

Embed curve in a viscous medium and solve at each step:

$$0 = \frac{\delta E}{\delta X} + \alpha \frac{dX}{dt} = \frac{\delta E_G}{\delta X} + KX + \frac{dX}{dt}$$

$$0 = \frac{\delta E}{\delta Y} + \alpha \frac{dY}{dt} = \frac{\delta E_G}{\delta Y} + KY + \frac{dY}{dt}$$

# ITERATING



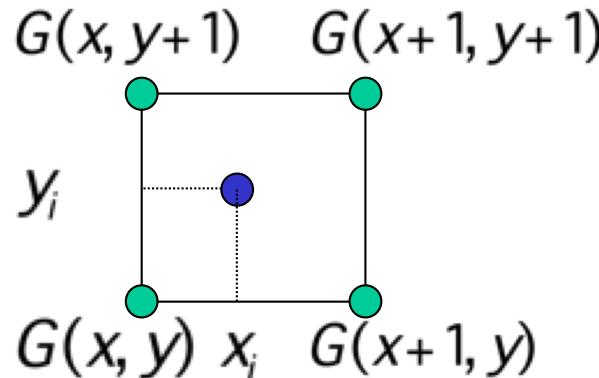
At every step:

$$0 = \frac{\delta E_G}{\delta X} + KX_t + \alpha(X_t - X_{t-1}) \Rightarrow (K + \alpha I)X_t = KX_{t-1} - \frac{\delta E_G}{\delta X}$$

$$0 = \frac{\delta E_G}{\delta Y} + KY_t + \alpha(Y_t - Y_{t-1}) \Rightarrow (K + \alpha I)Y_t = KY_{t-1} - \frac{\delta E_G}{\delta Y}$$

→ Solve two linear equations at each iteration.

# DERIVATIVES OF THE GRADIENT

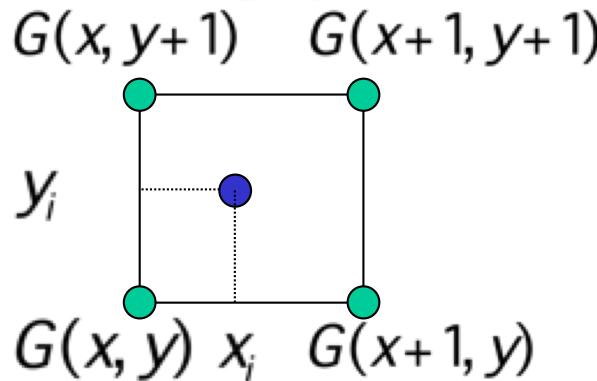


$$E_G = -\frac{I}{N} \sum_{i=1}^N G(x_i, y_i)$$

$$\frac{\partial E_G}{\partial X} = \begin{bmatrix} \frac{\partial E_G}{\partial x_1} & \dots & \frac{\partial E_G}{\partial x_N} \end{bmatrix}, \quad \frac{\partial E_G}{\partial Y} = \begin{bmatrix} \frac{\partial E_G}{\partial y_1} & \dots & \frac{\partial E_G}{\partial y_N} \end{bmatrix}$$

$$\frac{\partial E_G}{\partial x_i} = -\frac{I}{N} \frac{\partial G}{\partial x_i}(x_i, y_i), \quad \frac{\partial E_G}{\partial y_i} = -\frac{I}{N} \frac{\partial G}{\partial y_i}(x_i, y_i)$$

# BILINEAR INTERPOLATION



$$p = x_i - x$$

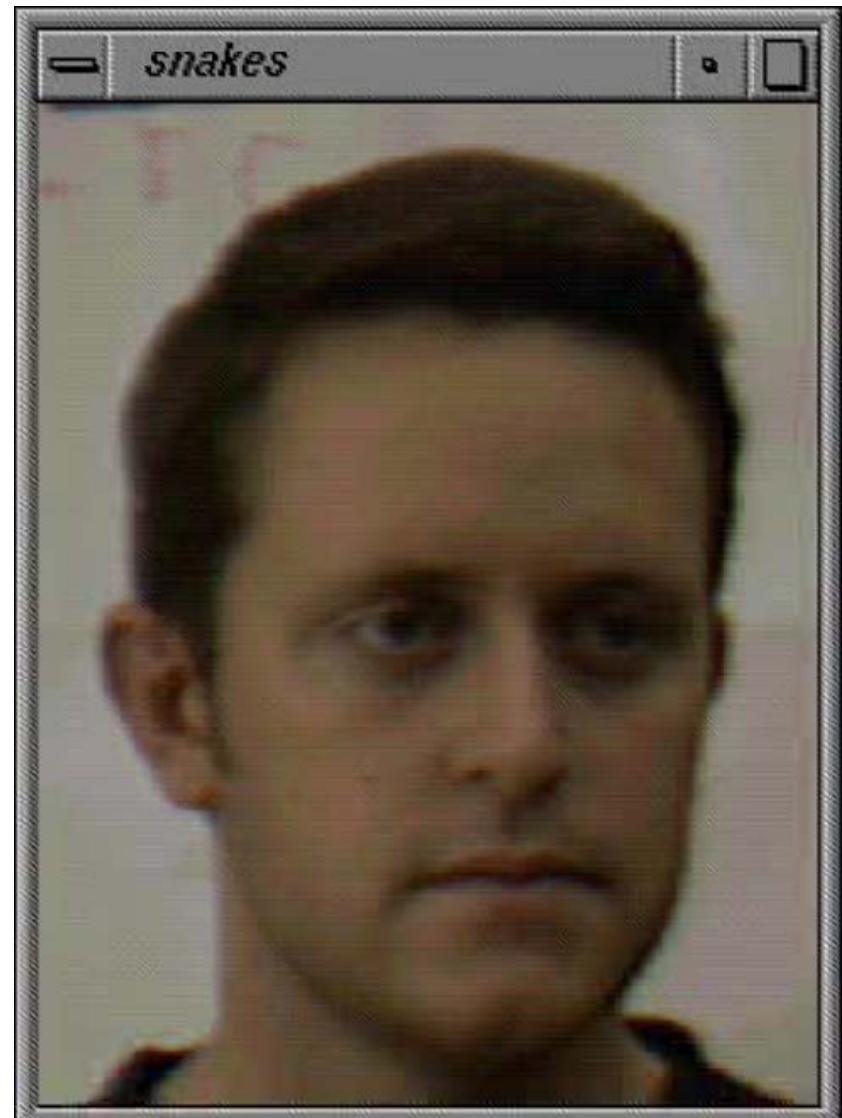
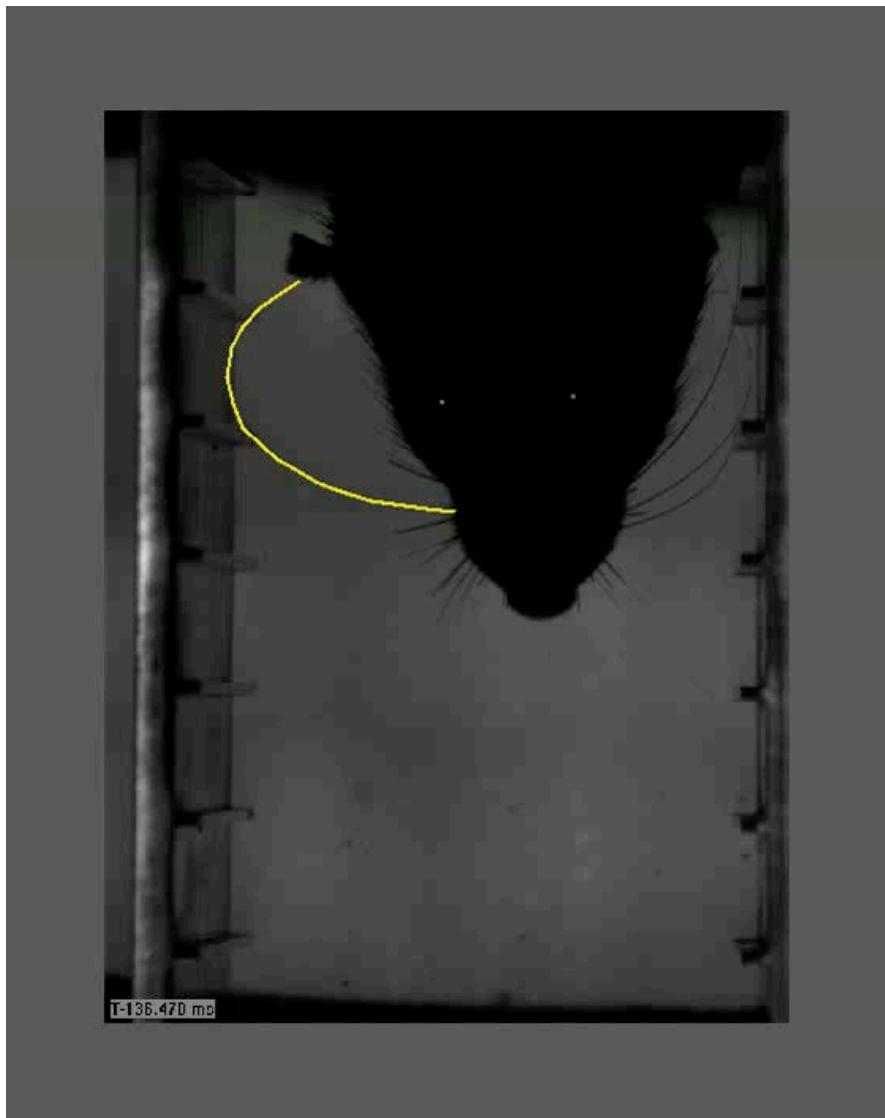
$$q = y_i - y$$

$$G(x_i, y_i) = (1-p)(1-q)G(x, y) + (1-p)qG(x, y+1) + p(1-q)G(x+1, y) + pqG(x+1, y+1)$$

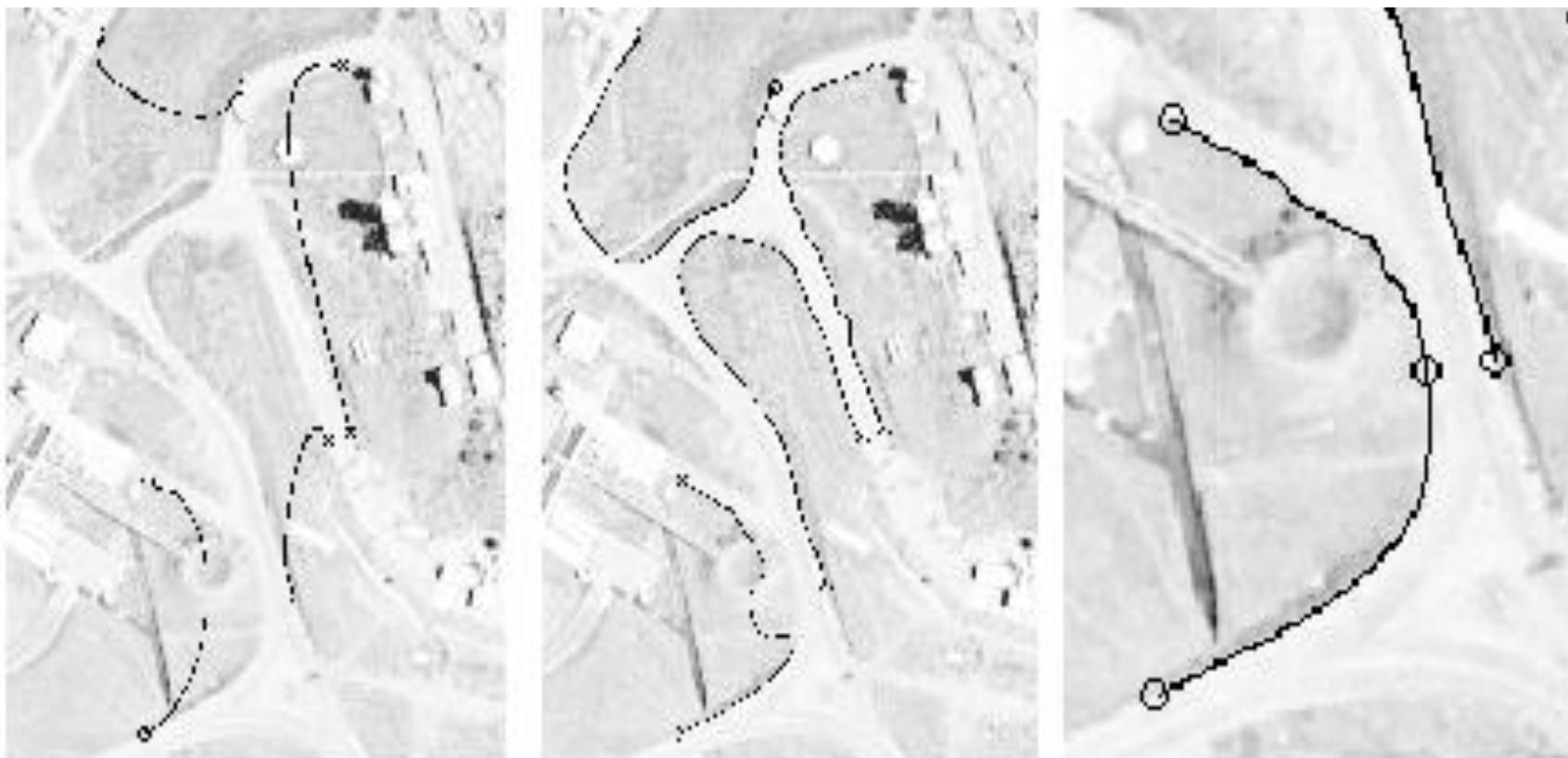
$$\frac{\partial G}{\partial x_i} = (1-q)(G(x+1, y) - G(x, y)) + q(G(x+1, y+1) - G(x, y+1))$$

$$\frac{\partial G}{\partial y_i} = (1-p)(G(x, y+1) - G(x, y)) + p(G(x+1, y+1) - G(x+1, y))$$

# OPEN AND CLOSED SNAKES



# ZIPLOCK SNAKES



Neuenschwander et al, IJCV'97

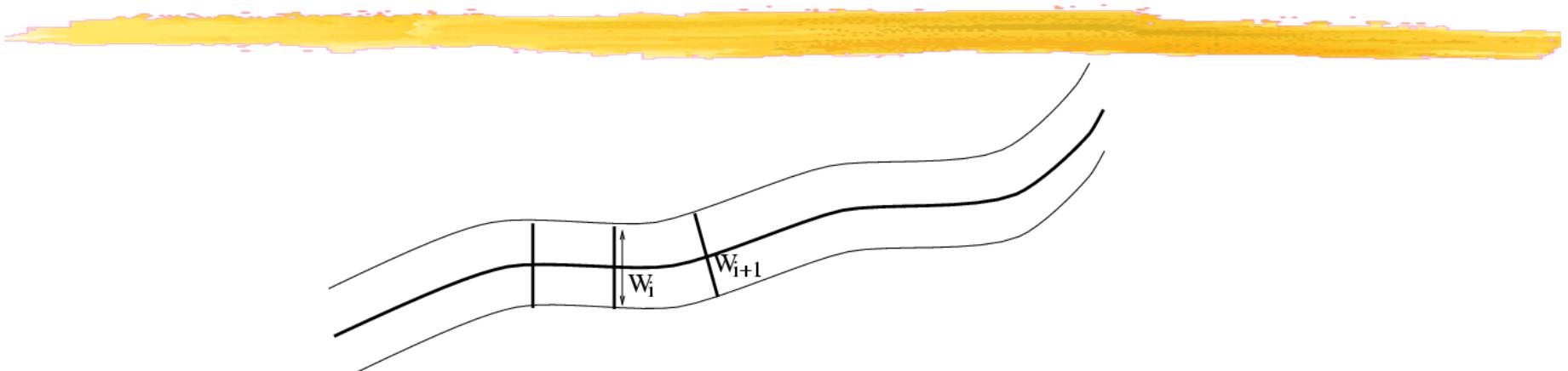
# NETWORK SNAKES



--> Updated field boundaries.

Butenuth, Phd 2008

# RIBBON SNAKES



$$E = E_G + 1/2X^t K X + 1/2Y^t K Y$$

$$W = [w_1, \dots, w_N]^t$$

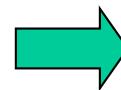
sum of squares of 1st derivatives

# DYNAMICS EQUATIONS

$$(K + \alpha I)X_t = KX_{t-1} - \frac{\delta E_G}{\delta X}$$
$$(K + \alpha I)Y_t = KY_{t-1} - \frac{\delta E_G}{\delta Y}$$
$$(K + \alpha I)W_t = KW_{t-1} - \frac{\delta E_G}{\delta W}$$

→ Solve three linear equations at each iteration.

# DELINEATING ROADS



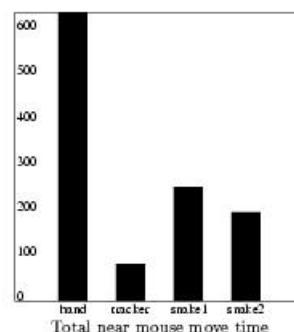
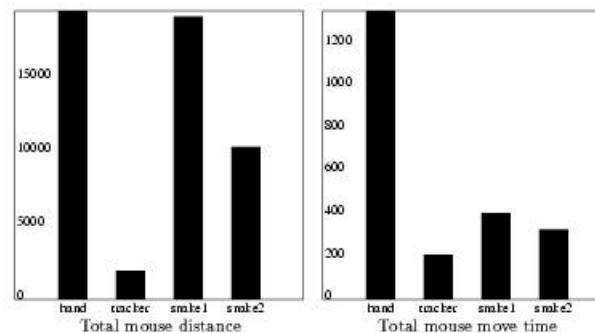
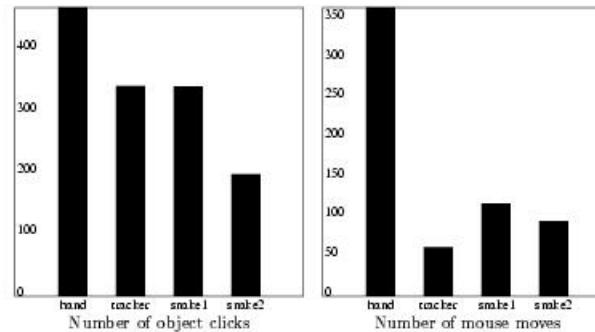
# DELINÉATING ROADS



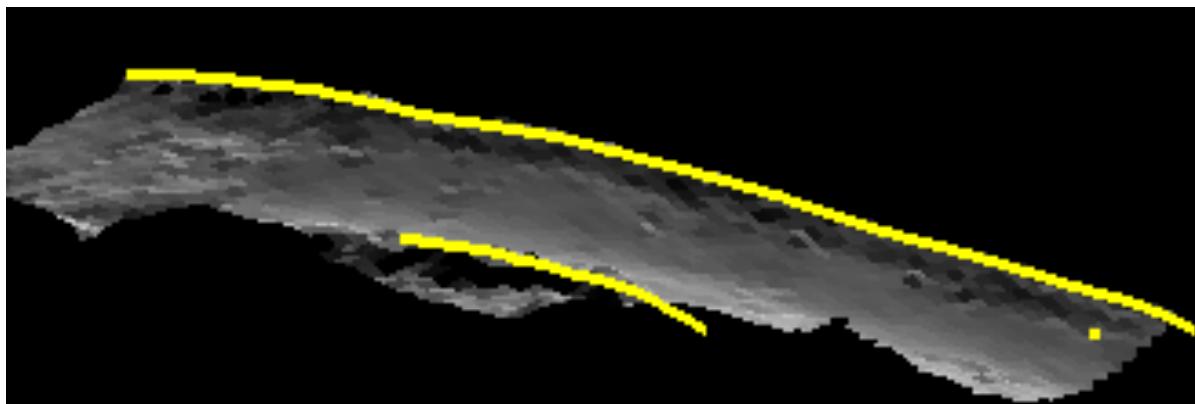
# DELINEATING ROADS



# EVALUATION



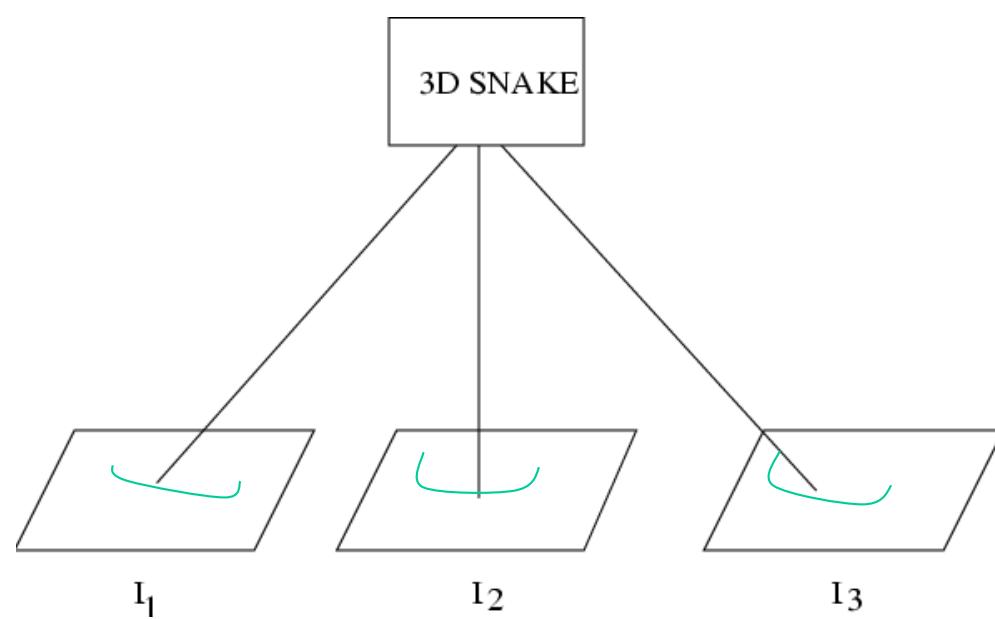
# RIDGE LINES



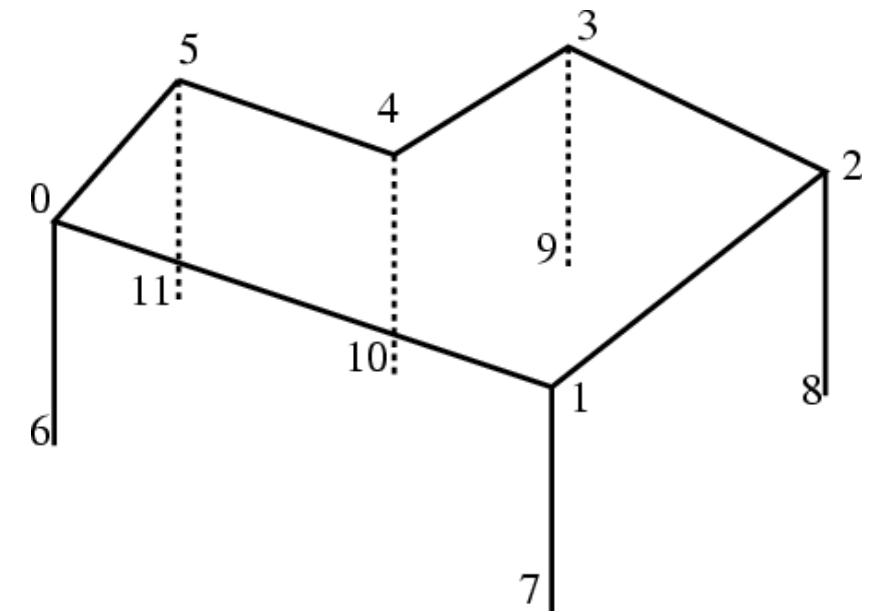
# BUILDING



# 3—D SNAKES



Smooth 3—D snake



Rectilinear 3—D snake

# DYNAMICS EQUATIONS

$$(K + \alpha I)X_t = KX_{t-1} - \frac{\delta E_G}{\delta X}$$

$$(K + \alpha I)Y_t = KY_{t-1} - \frac{\delta E_G}{\delta Y}$$

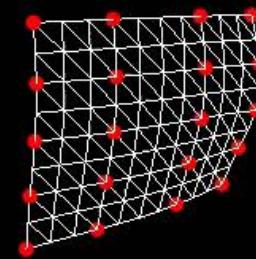
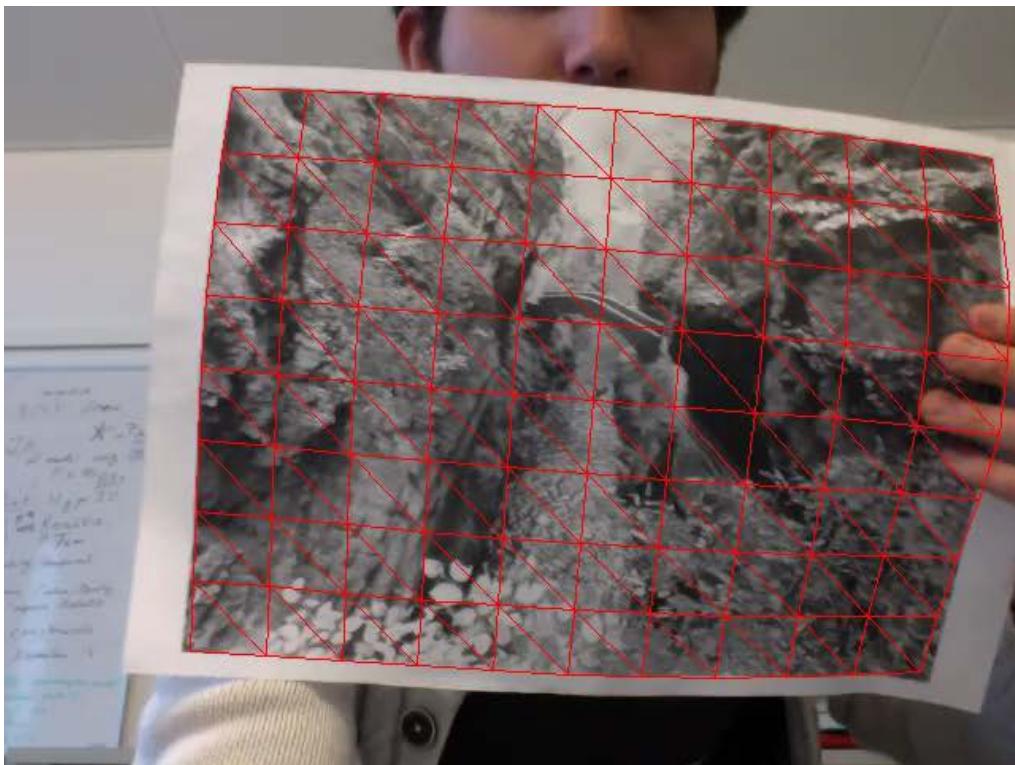
$$(K + \alpha I)Z_t = KZ_{t-1} - \frac{\delta E_G}{\delta Z}$$

→ Solve three linear equations at each iteration.

# SITE MODELING

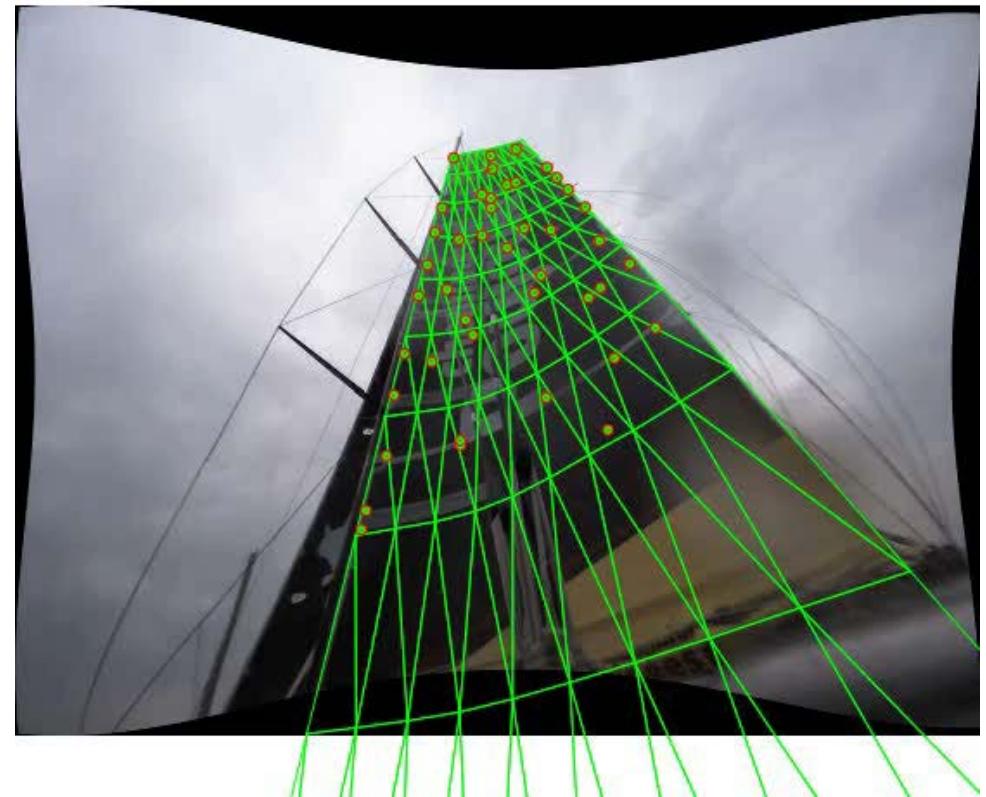


# REAL-TIME 3D SHAPE



Ostlund et al., ECCV'12.

# SAILS



→ Real-time performance evaluation.

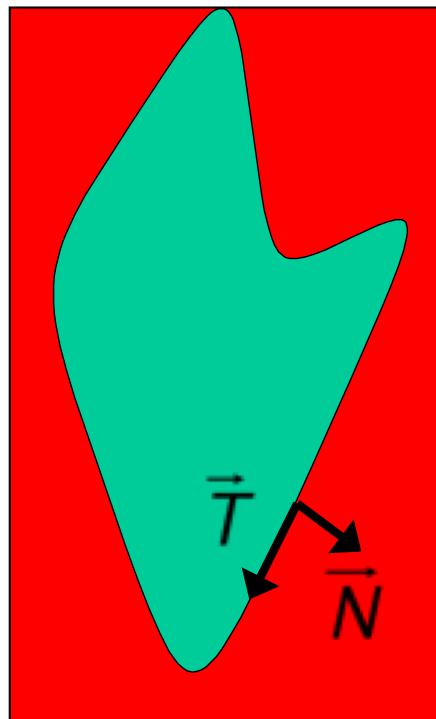
# WINGS



# LEVEL SETS



# CURVE EVOLUTION



**Generic formulation:**

$$\frac{\partial C}{\partial t} = \alpha(s, t)\vec{T} + \beta(s, t)\vec{N}$$

**Reparametrization:**

$$\frac{\partial C}{\partial t} = \beta(s, t)\vec{N}$$

**Special cases:**

- prairie fire model:
- reaction and diffusion model:

$$b = \pm 1$$

+1 expand, -1 contract

$$b = b_0 - b_1 k$$

constant - curvature ?

**Problems:**

1. Involves curve sampling and resampling.
2. Curvature estimates are numerically unstable.

low curvature regions  
move fast, high  
curvature regions  
move slowly. contour  
will go to a circle

# LEVEL SETS

Consider the curve as the zero level set of the surface.

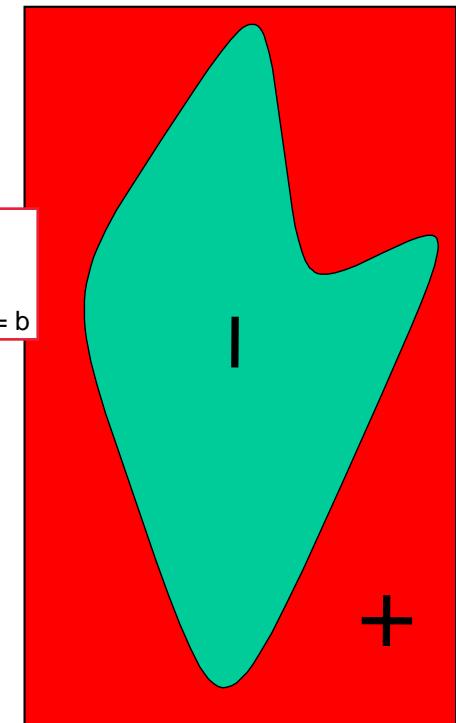
$$z = \Phi(x, y, t)$$

image that will evolve  
over time, because of t

$$\text{with evolution equation } \underline{\Phi}_t + \beta(\kappa) |\nabla \Phi| = 0$$

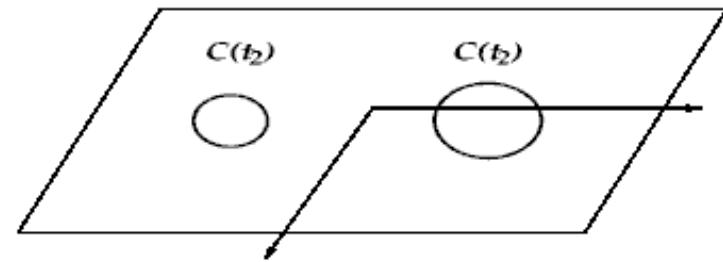
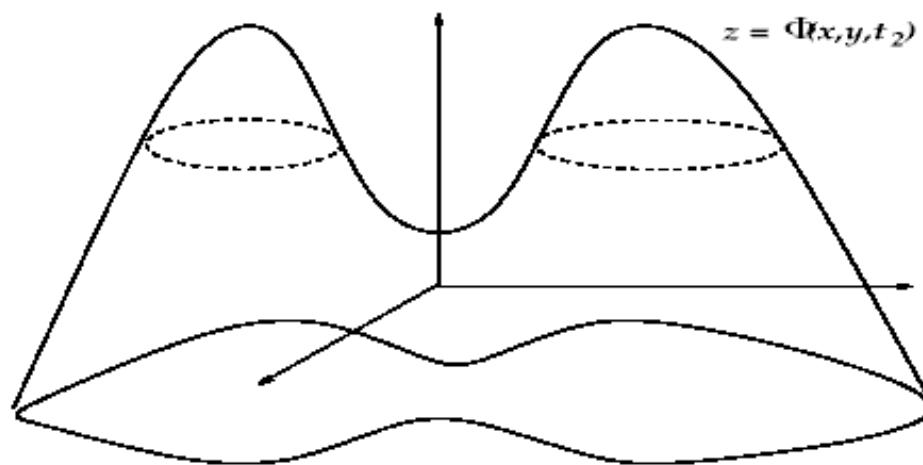
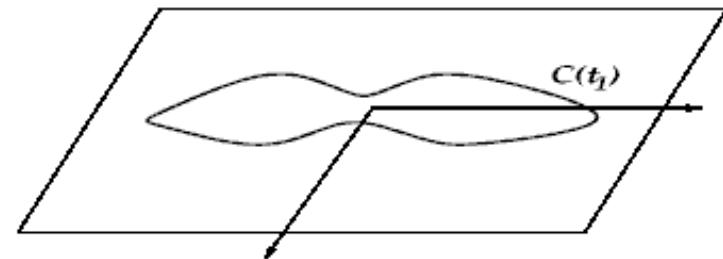
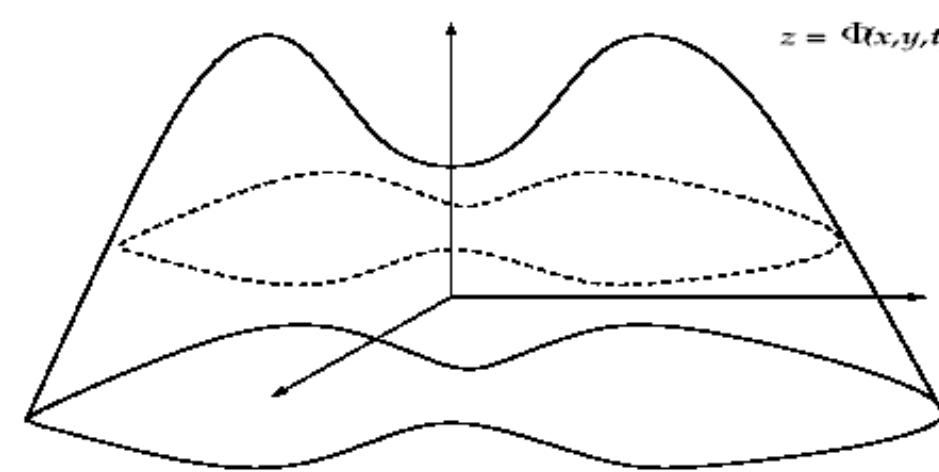
solving this equation  
will result in the  
expression in the  
previous slide  $dc/dt = b$

$$\kappa = \frac{(\Phi_{xx} \Phi_y^2 - 2\Phi_{xy} \Phi_x \Phi_y + \Phi_{yy} \Phi_x^2)}{(\Phi_x^2 + \Phi_y^2)}$$



→ Much better numerical stability.

# TOPOLOGY CHANGE



# LEVEL SET SMOOTHING



Smoothing occurs when  $b(k) = -k$

## Desirable properties:

- Converges towards circles.
- Total curvature decreases.
- Number of curvature extrema and zeros of curvature decreases.

## Relationship with Gaussian smoothing:

- Analogous to Gaussian smoothing of boundary over the short run, but does not cause self-intersections or overemphasize elongated parts.
- Can be implemented by Gaussian smoothing the characteristic function of a region.

# SHAPE RECOVERY



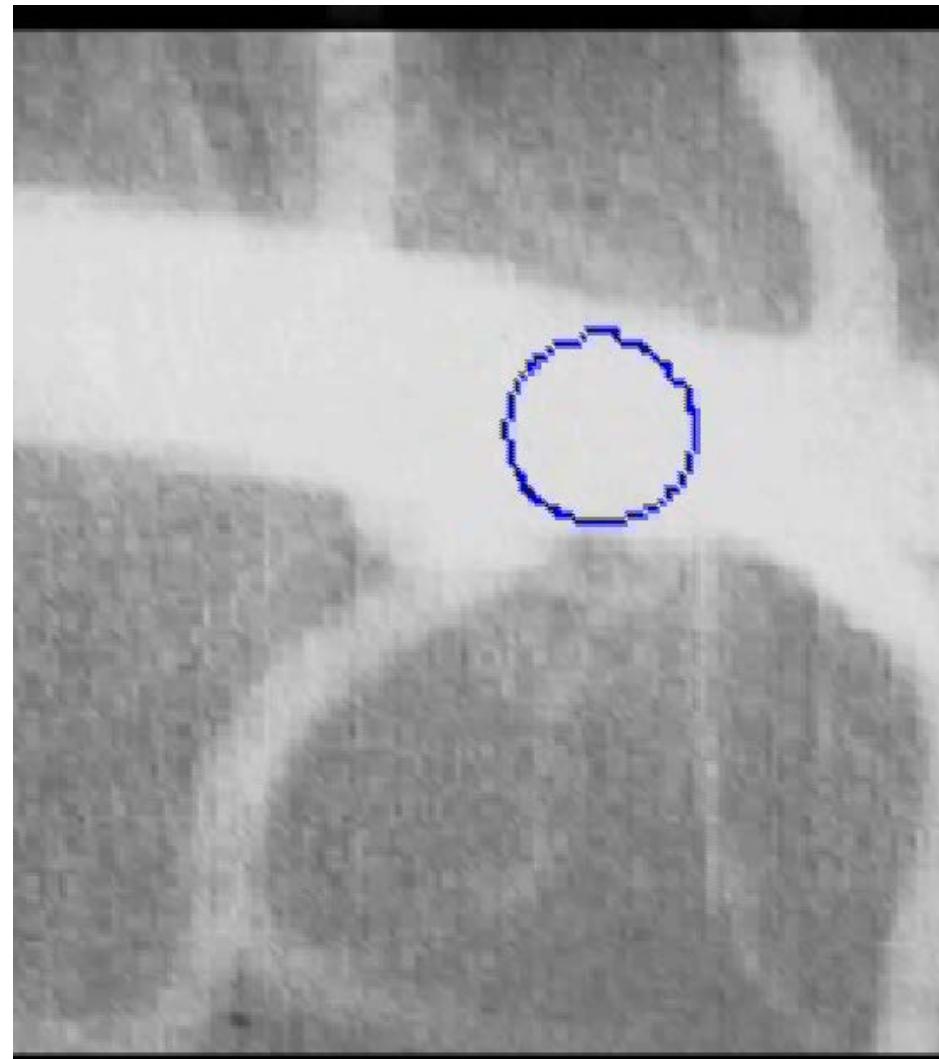
Evolution equation:  $\Phi_t + \beta(\kappa) |\nabla \Phi| = 0$

$$\beta(\kappa) = k_I(1 - \varepsilon\kappa)$$

$$k_I = \frac{1}{1 + \nabla I}$$

→ Expansion stops at the boundaries.

# LEVEL SETS



# LEVEL SETS



# TECHNIQUES



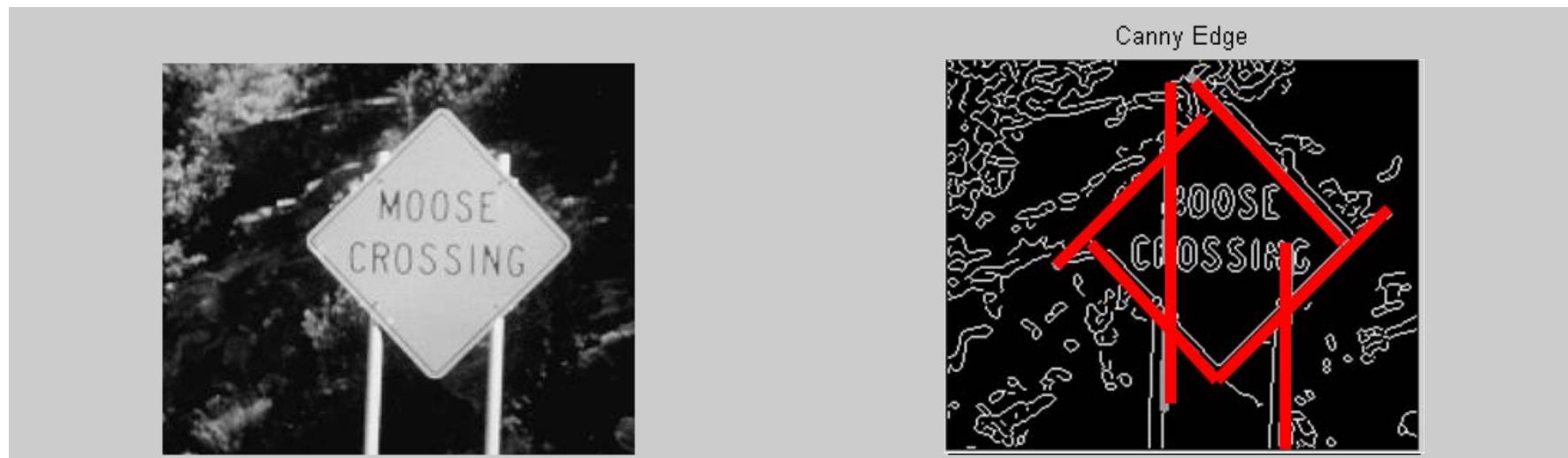
## Semi-Automated Techniques:

- Dynamic programming
- Deformable Models

## Fully Automated Techniques:

- Hough transform
- Graph Based Approaches

# FINDING LINES



Input:

- Canny edge points. why would i need the gradient magnitude ? to check if it is consistent with slope ?
- Gradient magnitude and orientation.

Output:

- All straight lines in image.

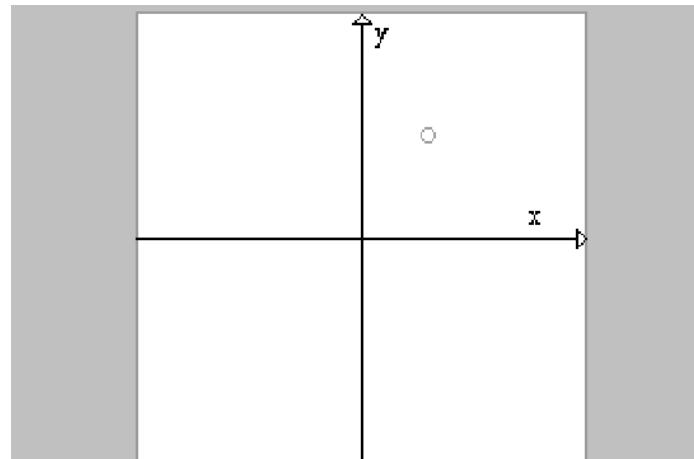
# HOUGH TRANSFORM



Given a parametric model of a curve:

- Map each contour point onto the set of parameter values for which the curves passes through it.
- Find the intersection for all parameter sets thus mapped.

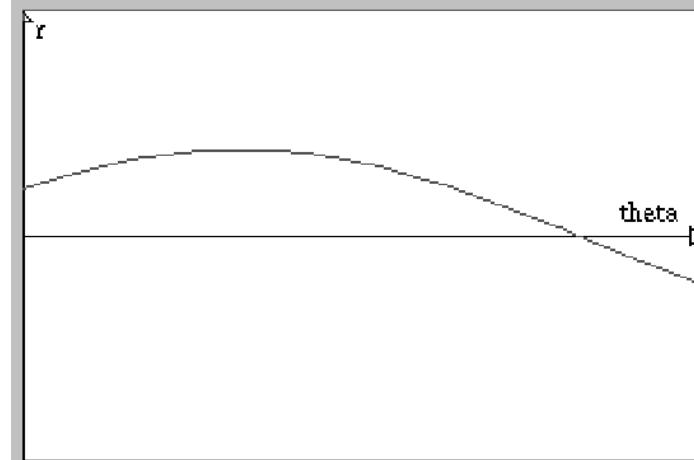
# VOTING SCHEME



$y = mx + c$  is mapped to  
 $c = xm - y$

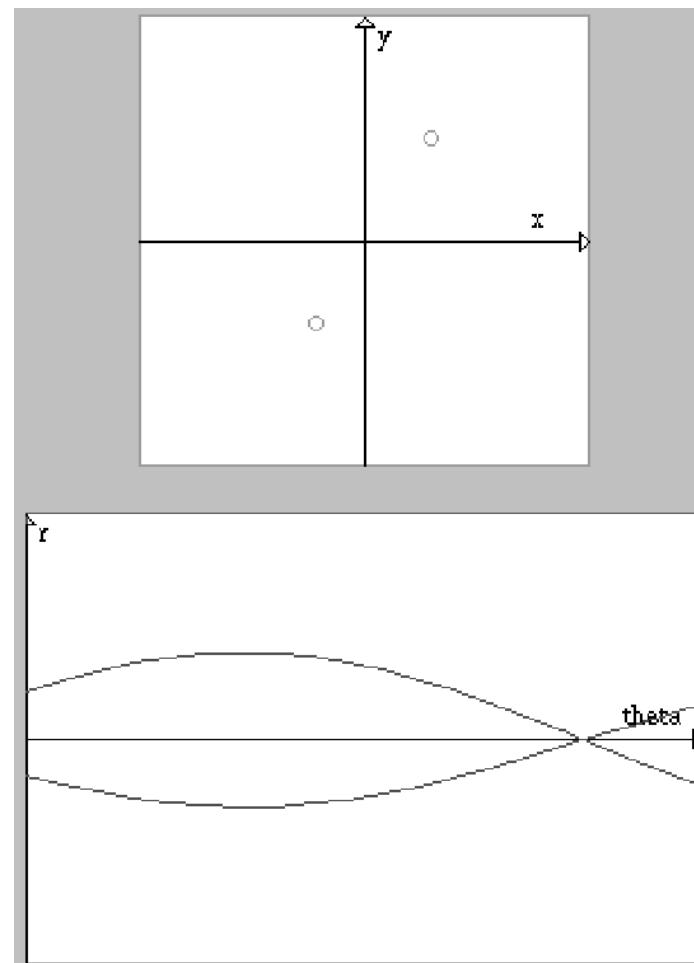
x and y now define the slope and intercept in the c,m space.

Then, iterate through all m values to get corresponding c value.

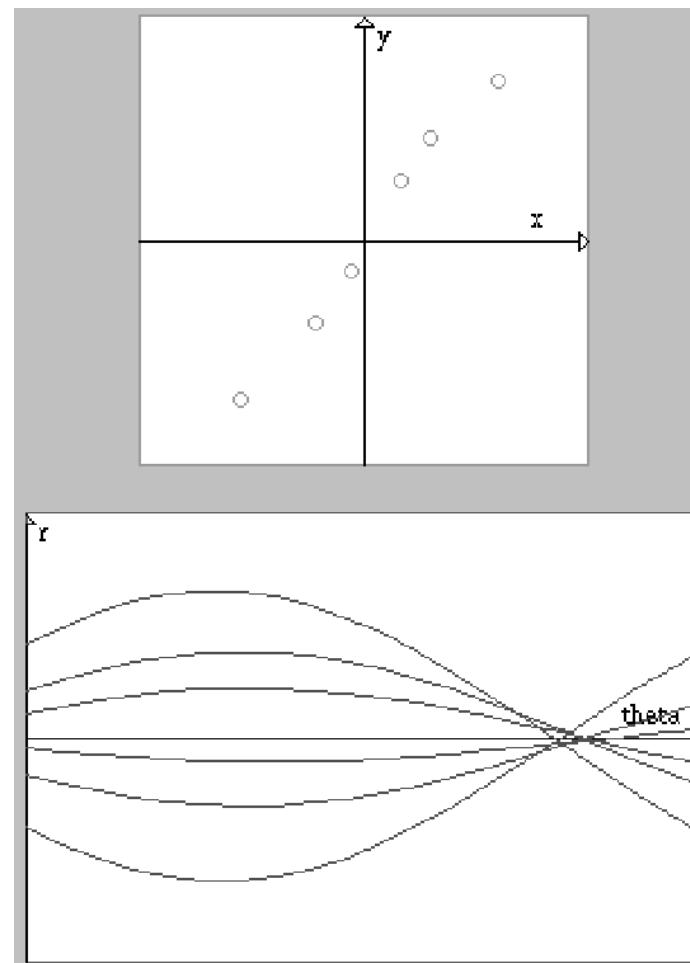


$$x \cos(\theta) + y \sin(\theta) = r, 0 \leq \theta < \pi$$

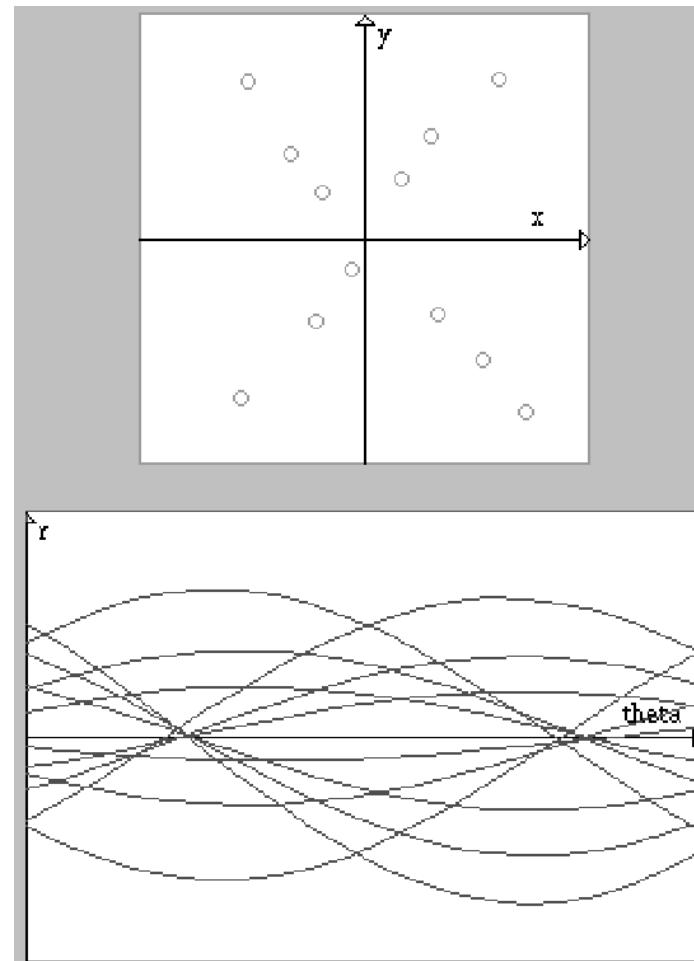
# VOTING SCHEME



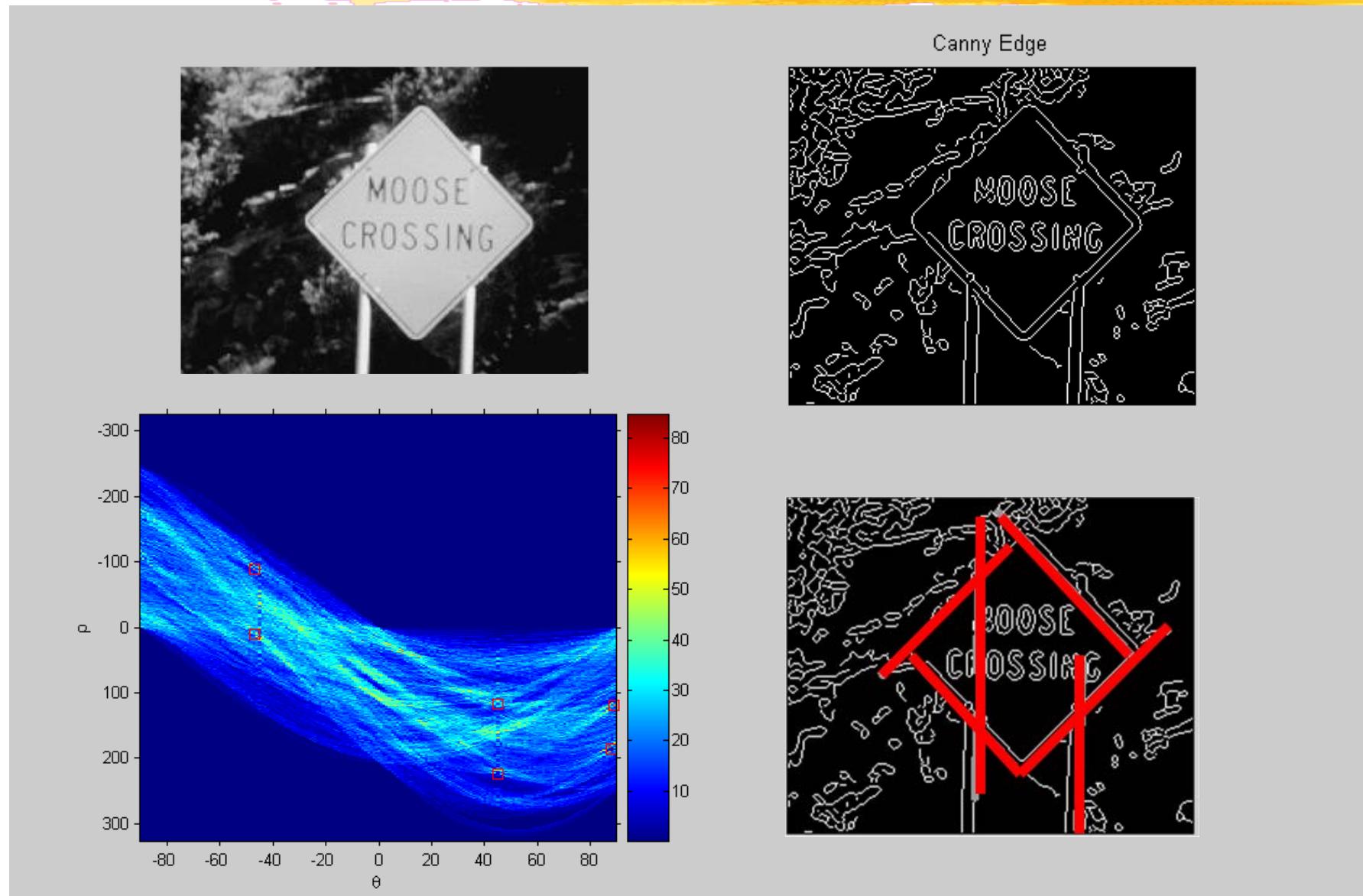
# VOTING SCHEME



# VOTING SCHEME



# REAL WORLD LINES



# ROAD EDGES

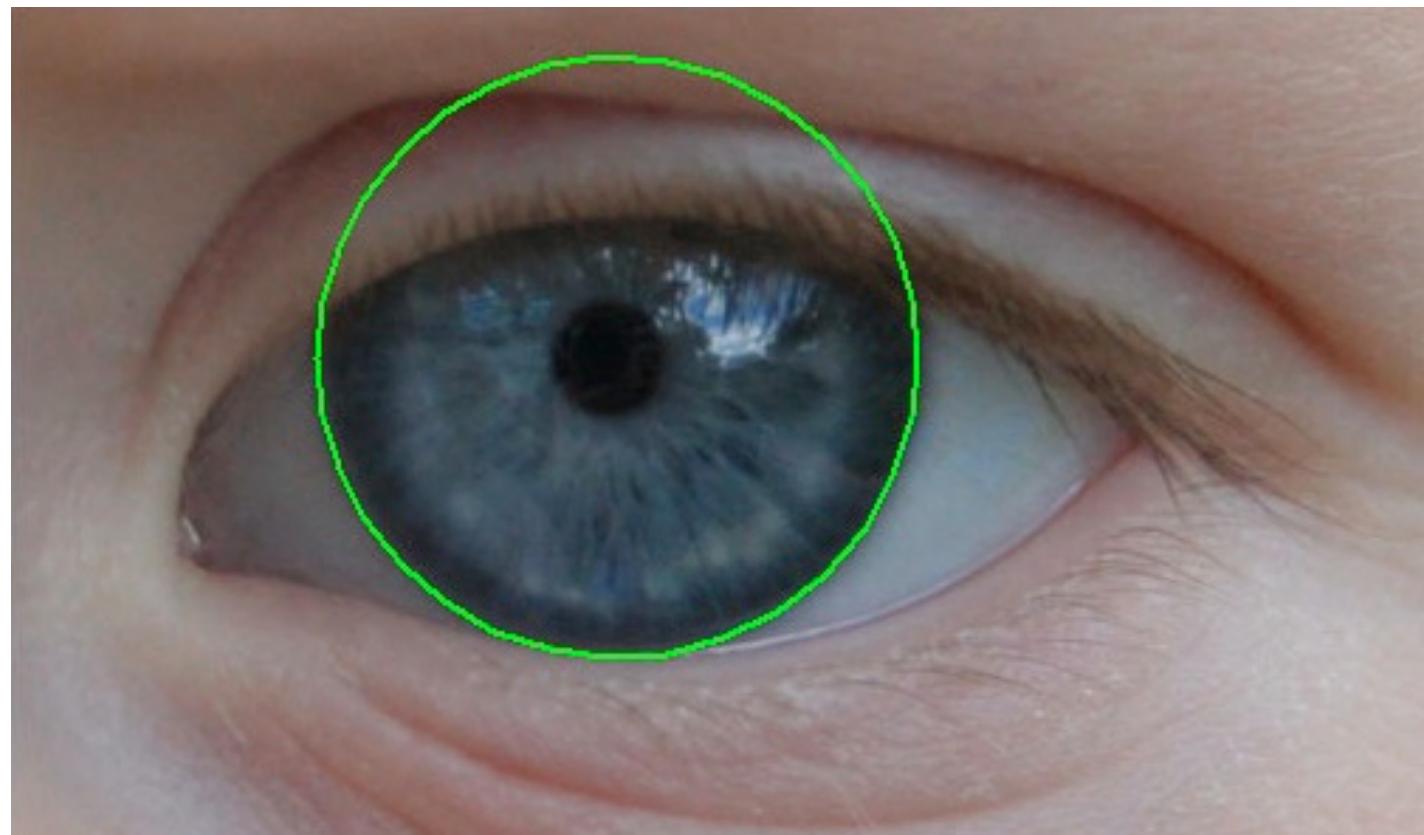


# GENERIC ALGORITHM



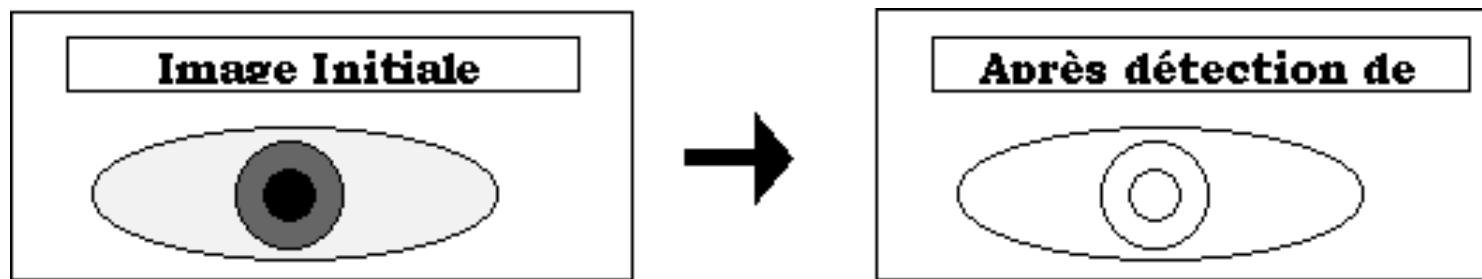
- Quantize parameter space (1 dimension per parameter)
- Form an accumulator array
- For each point in the gradient image such that the gradient strength exceeds a threshold, increment appropriate element of the accumulator
- Find local maxima in the accumulator

# IRIS DETECTION



# OCCLUSIONS

In theory:



In practice:



# CIRCLE DETECTION

Circle of equation:

$$x = x_0 + r \cos(\theta)$$

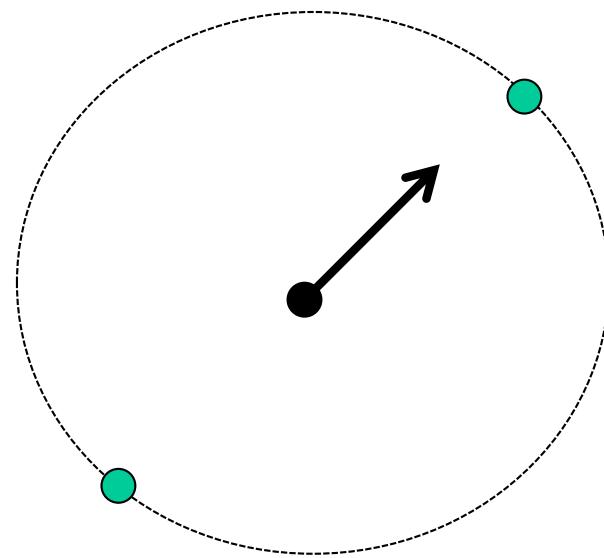
$$y = y_0 + r \sin(\theta)$$

Therefore:

$$x_0 = x - r \cos(\theta)$$

$$y_0 = y - r \sin(\theta)$$

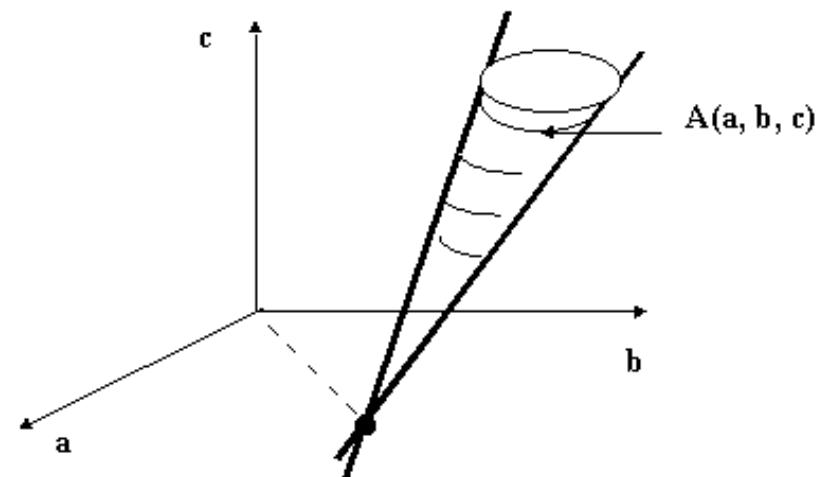
# GRADIENT ORIENTATION



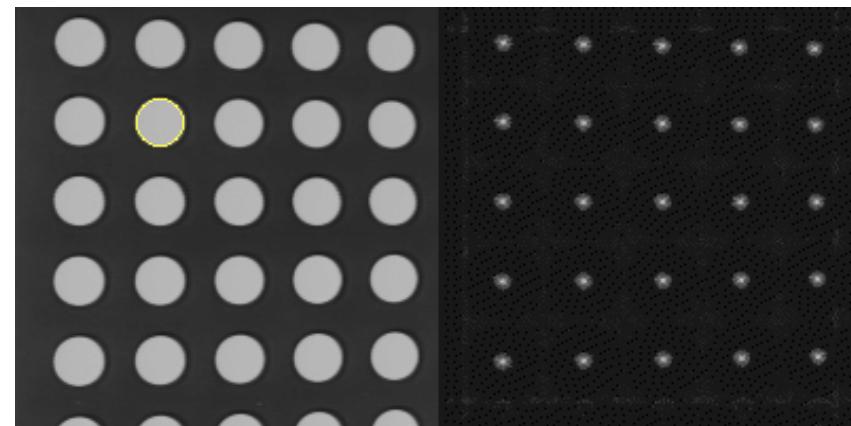
Can vote either along the entire circle or only at two points per value of the radius.

# SIMPLE IMAGE

Voting scheme:



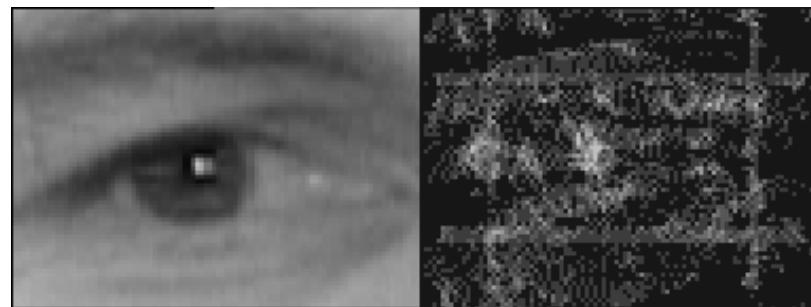
Result:



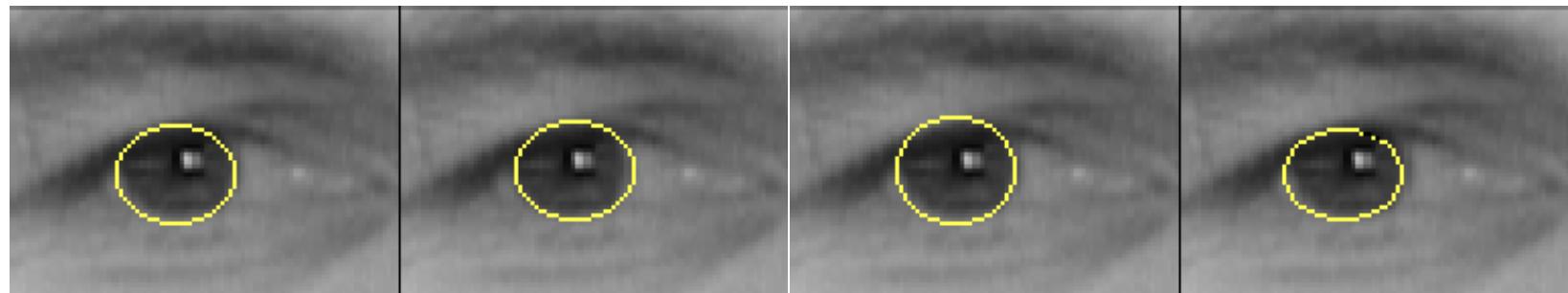
# EYE IMAGES



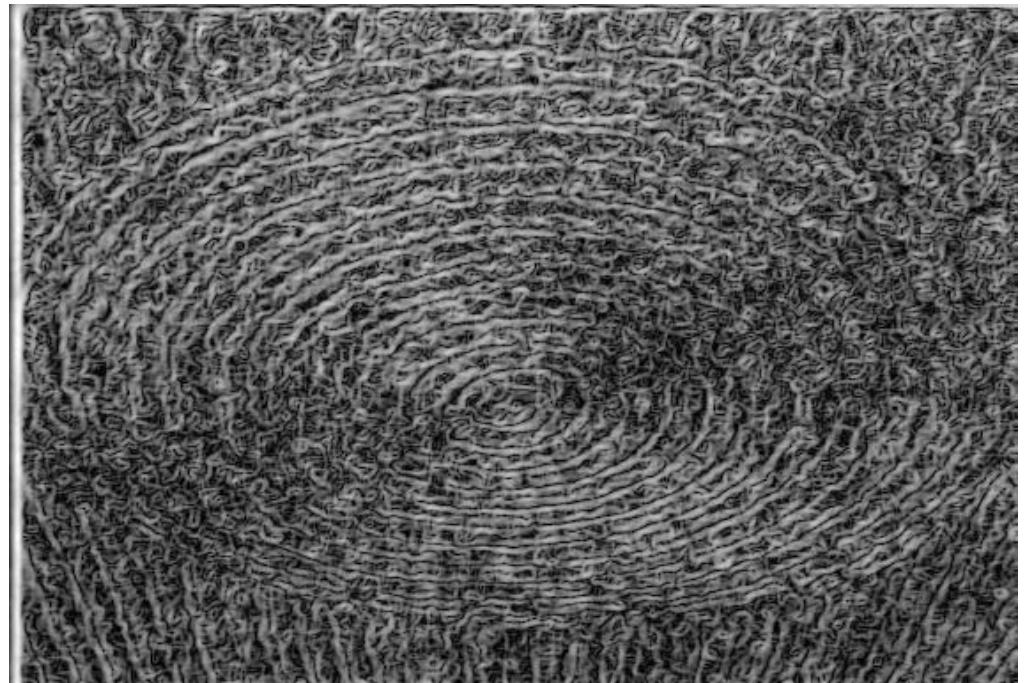
Image and accumulator:



Best four candidates:



# ELLIPSES



# ELLIPSE DETECTION

Ellipse of equation:

$$x = x_0 + a \cos(\theta)$$

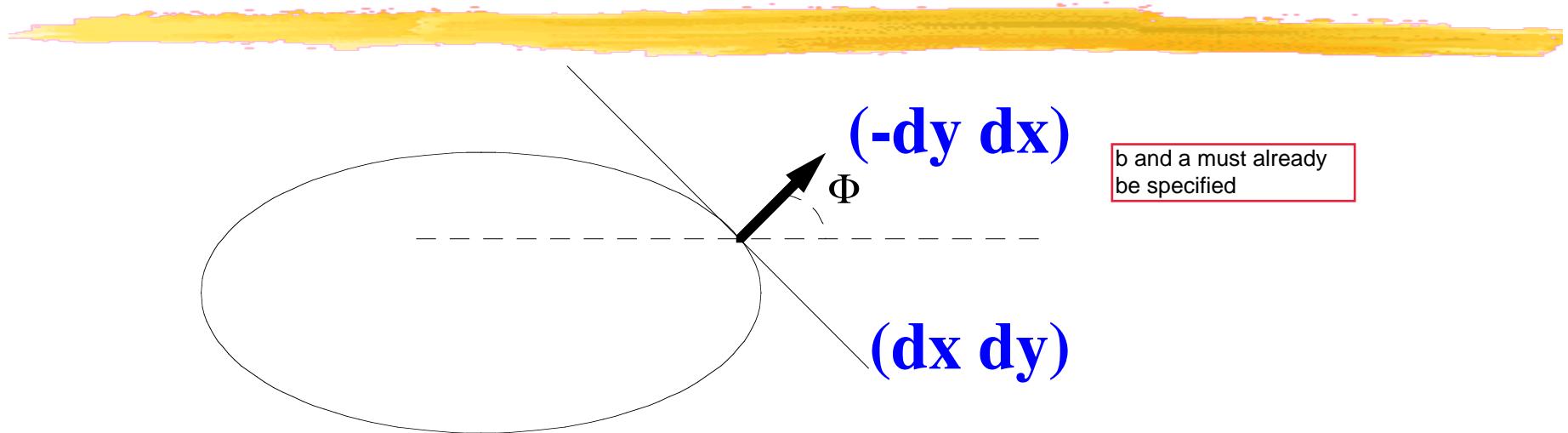
$$y = y_0 + b \sin(\theta)$$

Therefore:

$$x_0 = x - a \cos(\theta)$$

$$y_0 = y - b \sin(\theta)$$

# GRADIENT ORIENTATION



For each ellipse point:

$$\frac{dy}{dx} = -\frac{b}{a} \cdot \frac{\cos(\theta)}{\sin(\theta)}$$

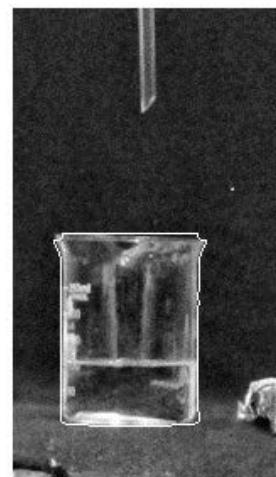
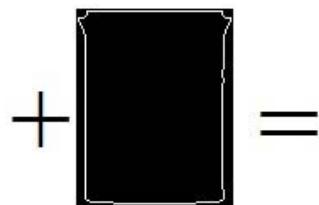
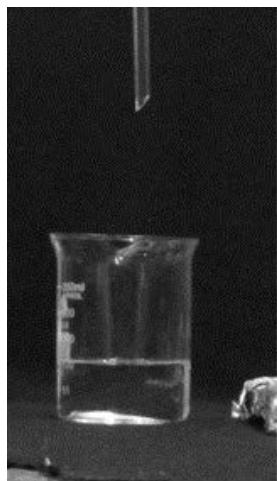
$$-\frac{dx}{dy} = \frac{a}{b} \cdot \tan(\theta)$$

The accumulator need only be incremented for:

$$\theta = \text{atan}\left(\frac{b}{a} \tan(\Phi)\right)$$

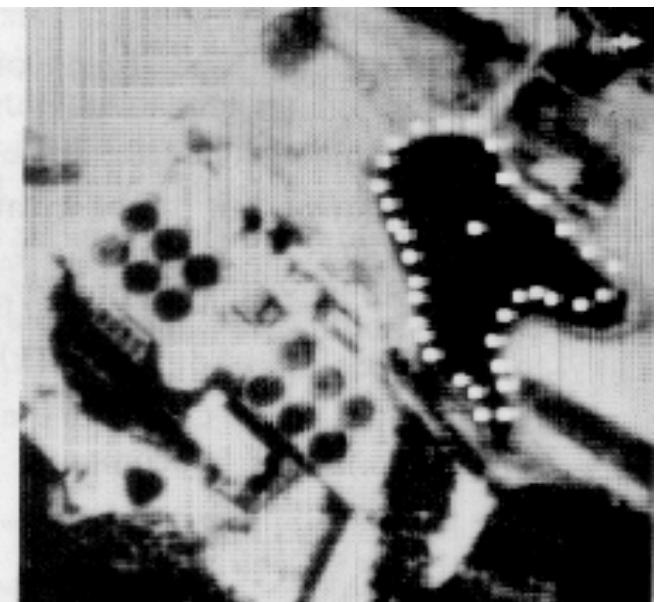
b and a must already  
be specified

# GENERALIZED HOUGH

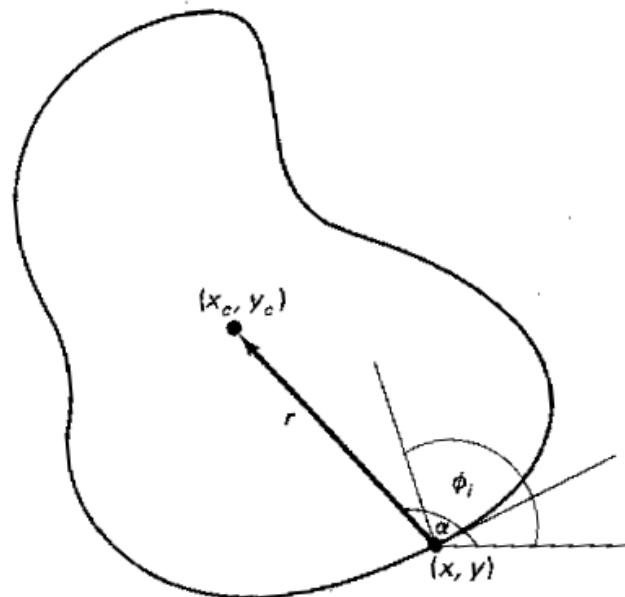


Finding a becher ...

... or a lake.



# R-TABLE



<i>Angle measured from figure boundary to reference point</i>	<i>Set of radii <math>\{r^k\}</math> where <math>r = (r, \alpha)</math></i>
$\phi_1$	$r_1^1, r_2^1, \dots, r_{n_1}^1$
$\phi_2$	$r_1^2, r_2^2, \dots, r_{n_2}^2$
$\vdots$	$\vdots$
$\phi_m$	$r_1^m, r_2^m, \dots, r_{n_m}^m$

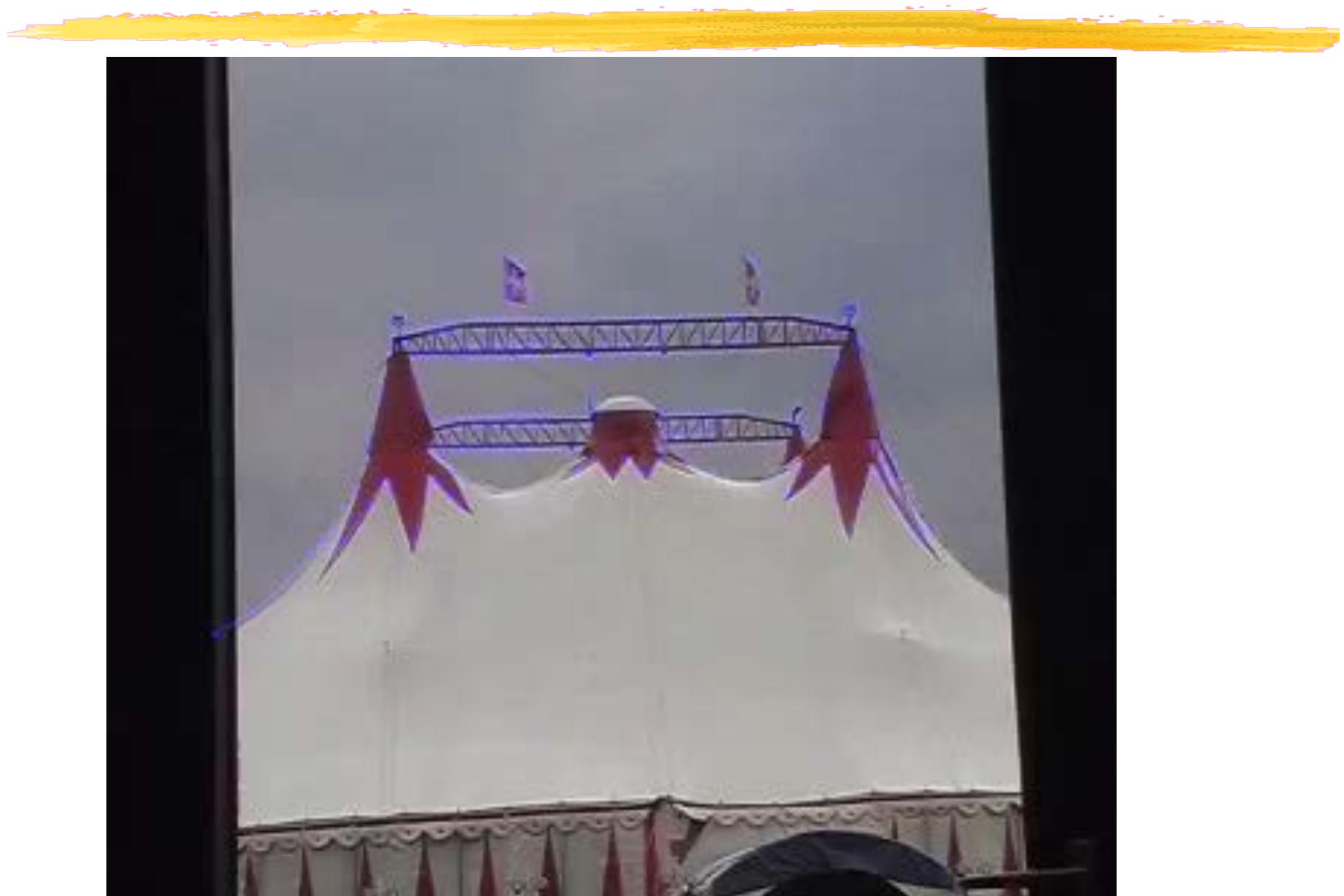
Code the shape in the form of an R-Table:  
Set of possible positions of a reference  
point given boundary orientation.  
--> Generalized template matching.

# ALGORITHM

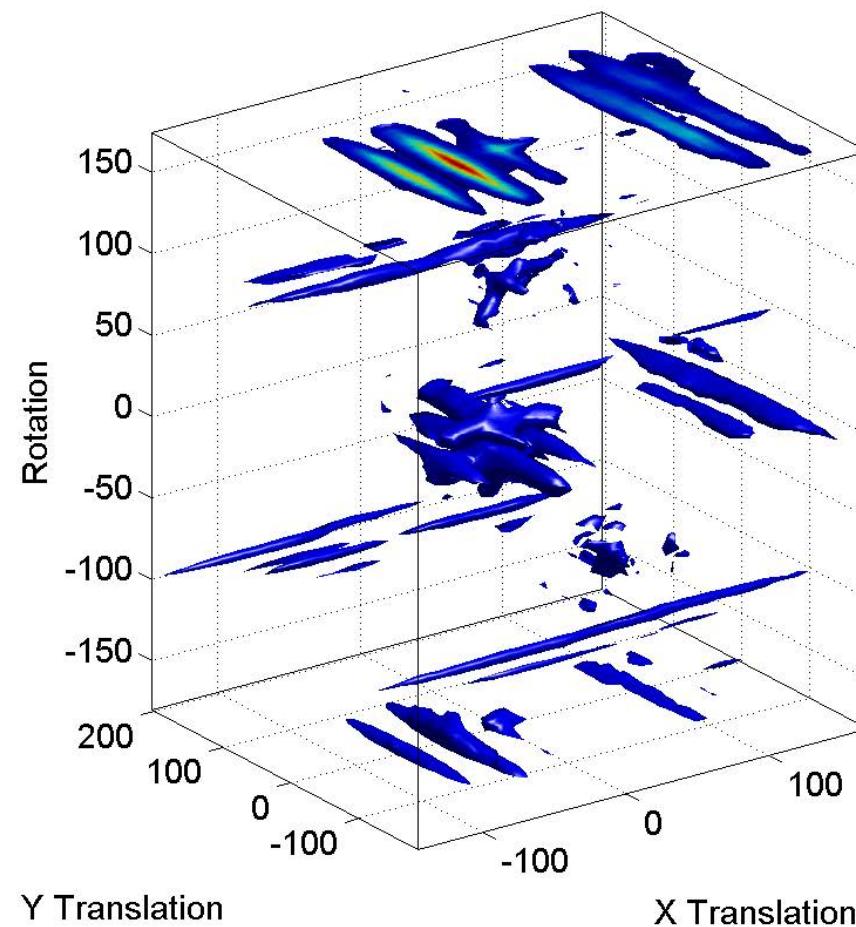


1. Make an R-table for the shape to be located.
2. Form an accumulator array of possible reference points initialized to zero.
3. For each edge point,
  - Compute the possible centers, that is, for each table entry, compute
$$x = x_e + r_\phi \cos(\theta(\phi))$$
$$y = y_e + r_\phi \sin(\theta(\phi))$$
  - Increment the accumulator array

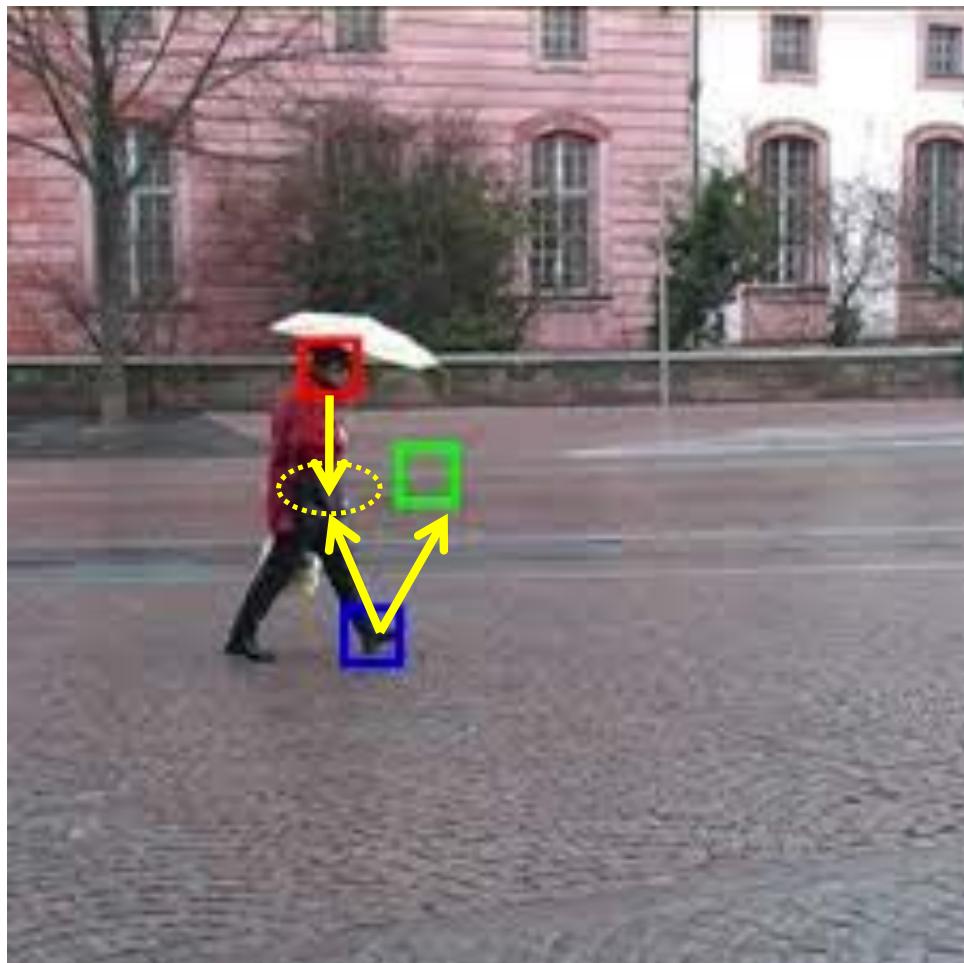
# REAL-TIME HOUGH



# ACCUMULATOR



# HOUGH FORESTS



# OCCLUSION HANDLING



# LIMITATIONS



Computational cost grows exponentially with the number of model parameters:

- Only works for objects whose shape can be defined by a small number of parameters.
- Approach is robust but lacks flexibility.

# TECHNIQUES



Semi-Automated Techniques:

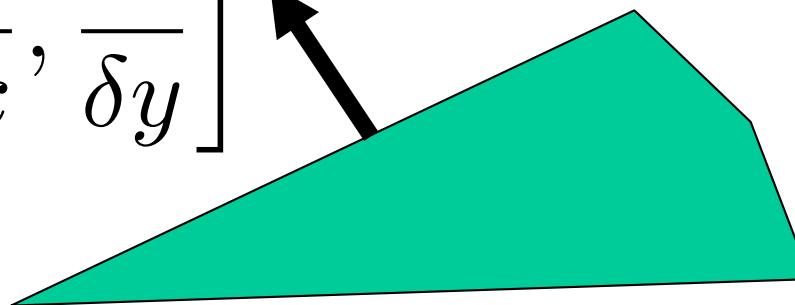
- Dynamic programming
- Deformable Models

Fully Automated Techniques:

- Hough transform
- Graph Based Approaches

# MAGNITUDE AND ORIENTATION

$$\left[ \frac{\delta I}{\delta x}, \frac{\delta I}{\delta y} \right]$$

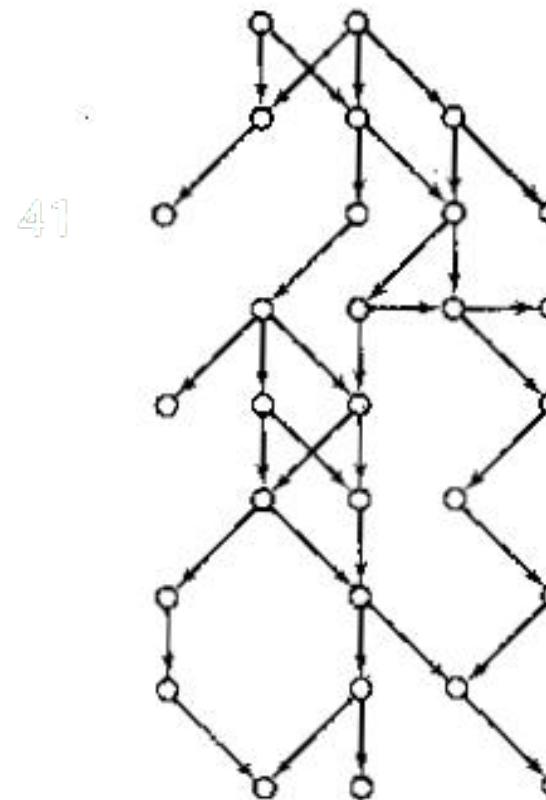
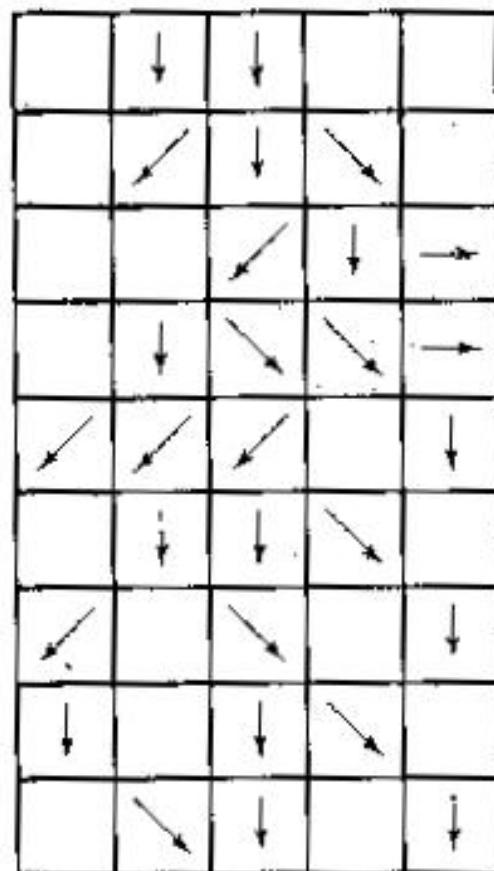


$$\text{Contrast: } G = \sqrt{\frac{\delta I^2}{\delta x} + \frac{\delta I^2}{\delta y}}$$

$$\text{Orientation: } \Theta = \arctan\left(\frac{\delta I}{\delta y}, \frac{\delta I}{\delta x}\right)$$

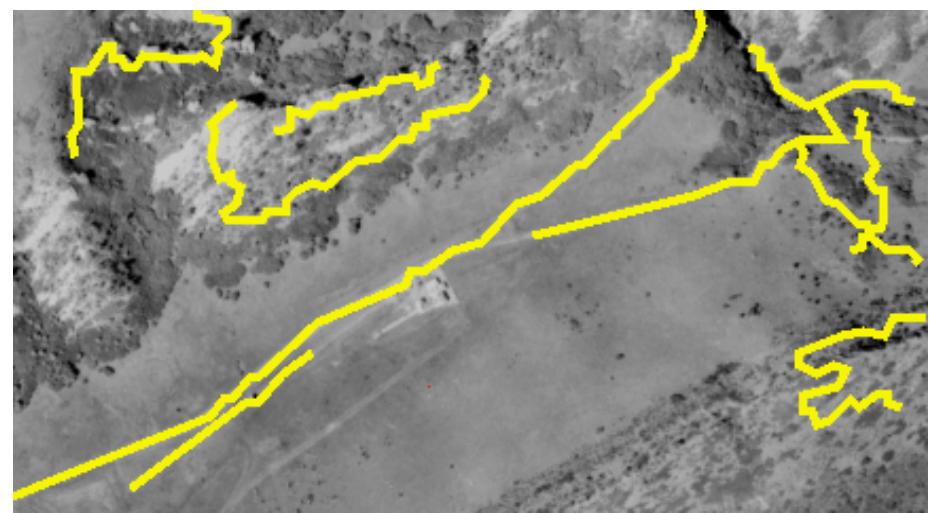
# MINIMUM SPANNING TREE

Image modeled as a graph:



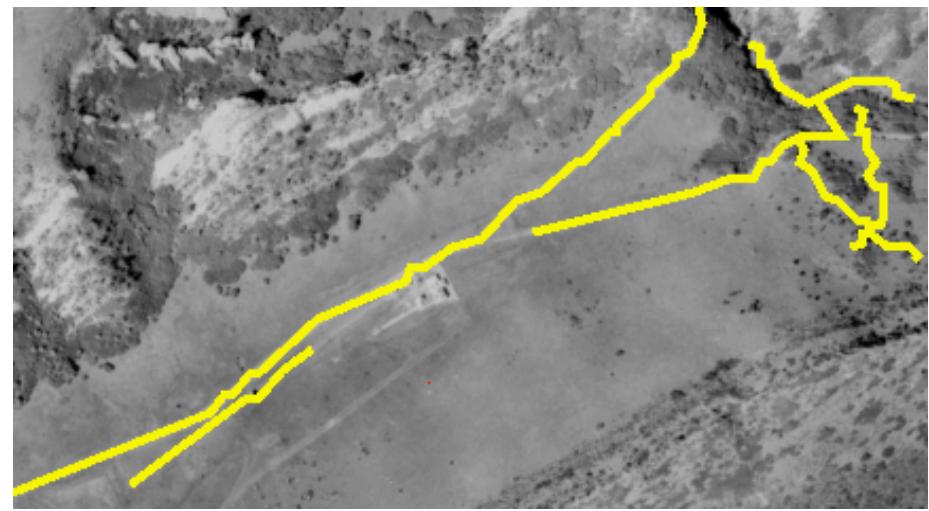
-> Generate minimal distance graph  $O(N \log(N))$  algorithm)

# ROAD DELINEATION

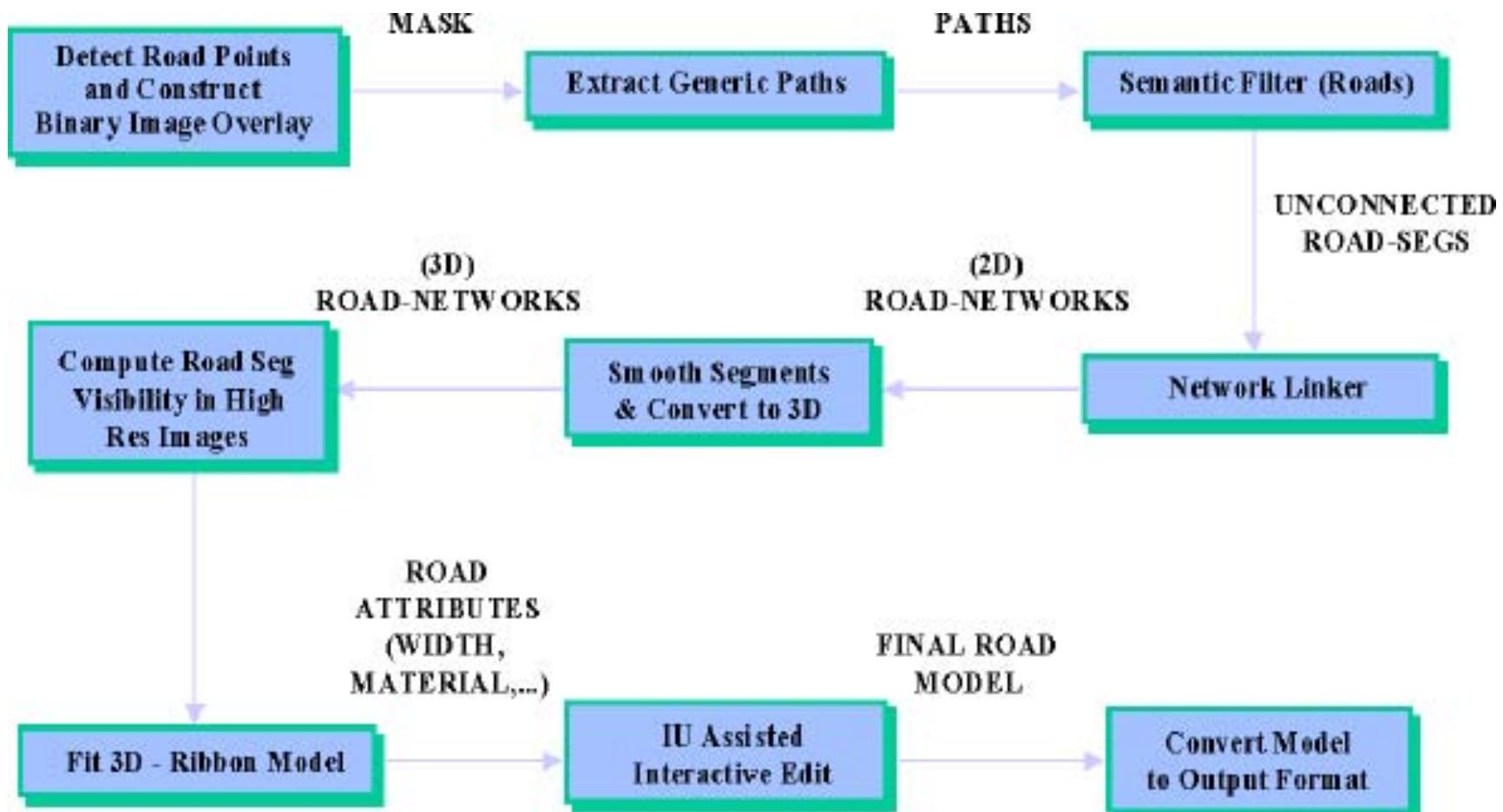


Fischler & Heller, 1998.

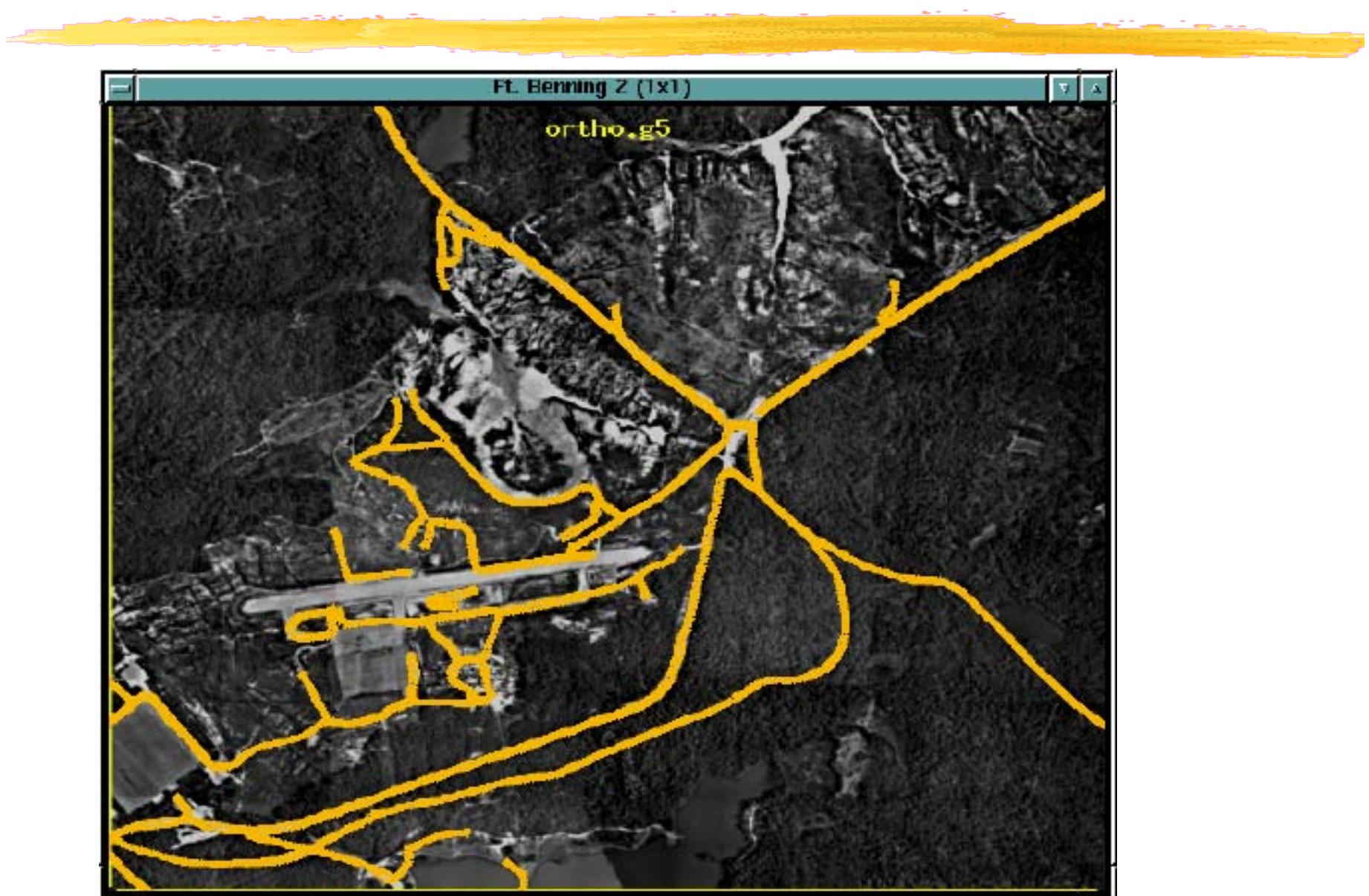
# ROAD EDITING



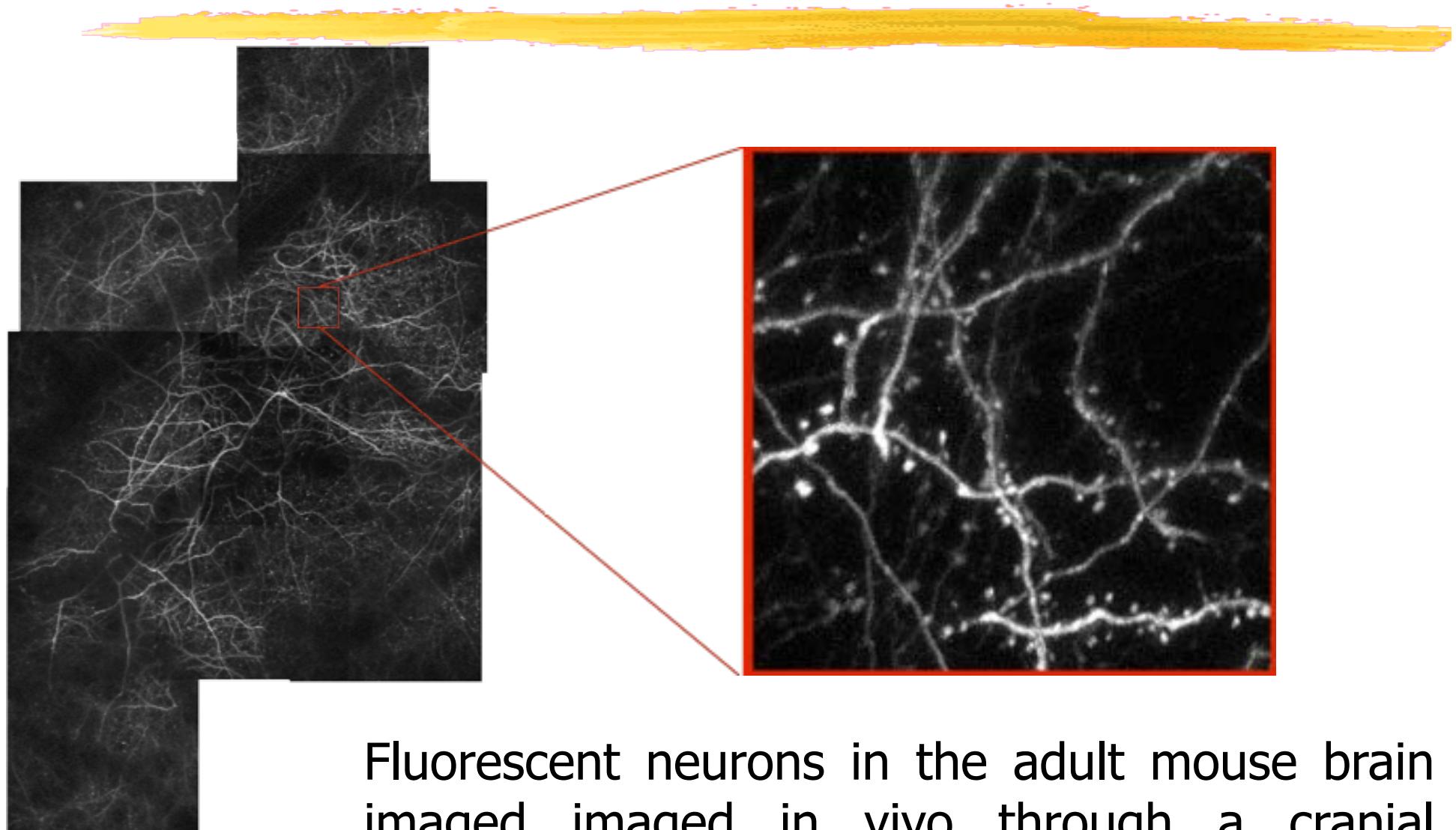
# FROM IMAGE TO ROADS



# PROCESSING STEPS

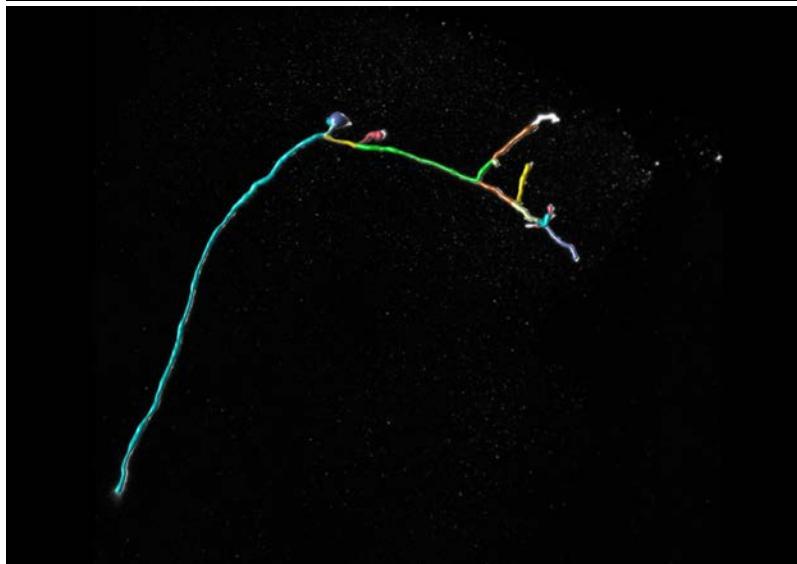
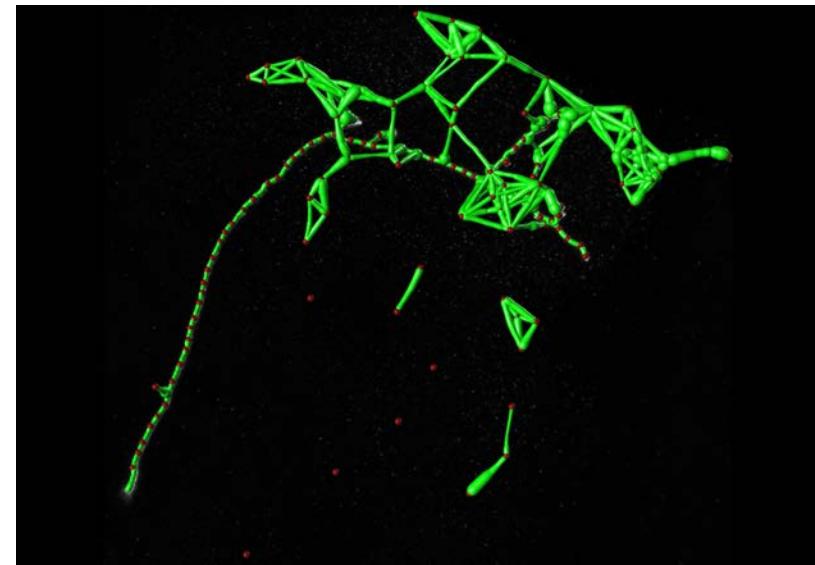
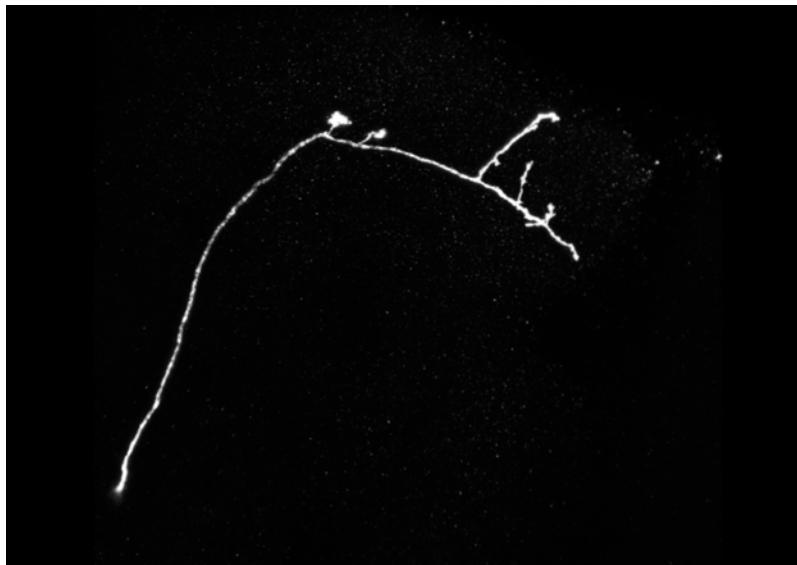


# DENDRITES AND AXONS



Fluorescent neurons in the adult mouse brain imaged *in vivo* through a cranial window using a 2-photon microscope.

# QUADRATIC MIXED INTEGER PROGRAMMING



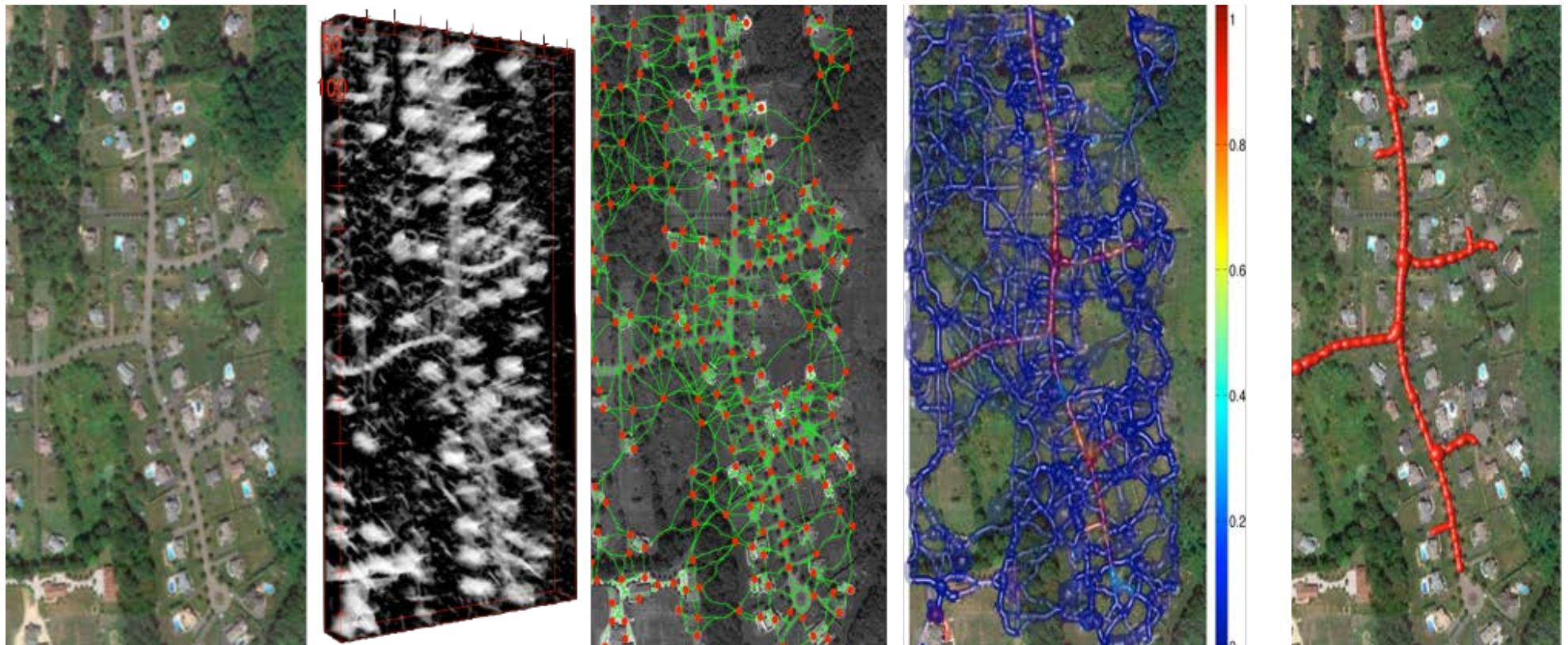
$$\begin{aligned} \mathbf{t}^* &= \underset{\mathbf{t} \in \mathcal{T}(G)}{\operatorname{argmax}} P(\mathbf{T} = \mathbf{t} | I) , \\ &= \underset{\mathbf{t} \in \mathcal{T}(G)}{\operatorname{argmax}} P(I | \mathbf{T} = \mathbf{t}) P(\mathbf{T} = \mathbf{t}) , \\ &= \underset{\mathbf{t} \in \mathcal{T}(G)}{\operatorname{argmin}} \sum_{e_{ij} \in G} \left[ -\log \frac{P(T_{ij} = 1 | I_{ij})}{P(T_{ij} = 0 | I_{ij})} \right] t_{ij} \\ &\quad + \sum_{e_{ij}, e_{jk} \in G} f(e_{ij}, e_{jk}) t_{ij} t_{jk} . \end{aligned}$$

# QMIP FORMULATION

$$\begin{aligned} & \min \sum_{e_{ij} \in E, e_{jk} \in E} c_{ijk} t_{ij} t_{jk} \\ \text{s.t. } & \sum_{v_j \in V \setminus \{v_r\}} y_{rj}^l \leq 1, \quad \forall v_l \in V \setminus \{v_r\}, \\ & \sum_{v_j \in V \setminus \{v_k\}} y_{jk}^l \leq 1, \quad \forall v_l \in V \setminus \{v_r\}, \\ & \sum_{v_j \in V \setminus \{v_i, v_r\}} y_{ij}^l - \sum_{v_j \in V \setminus \{v_i, v_l\}} y_{ji}^l = 1, \quad \forall v_k \in V \setminus \{v_r\}, \\ & \quad \forall v_i \in V \setminus \{v_r, v_k\}, \\ & y_{ij}^l \leq t_{ij}, \quad \forall e_{ij} \in E, v_l \in V \setminus \{v_r, v_i, v_j\}, \\ & y_{il}^l = t_{il}, \quad \forall e_{il} \in E, \\ & y_{ij}^l \geq 0, \quad \forall e_{ij} \in E, v_l \in V \setminus \{v_r, v_i\}, \\ & t_{ij} \in \{0, 1\}, \quad \forall e_{ij} \in E. \end{aligned}$$

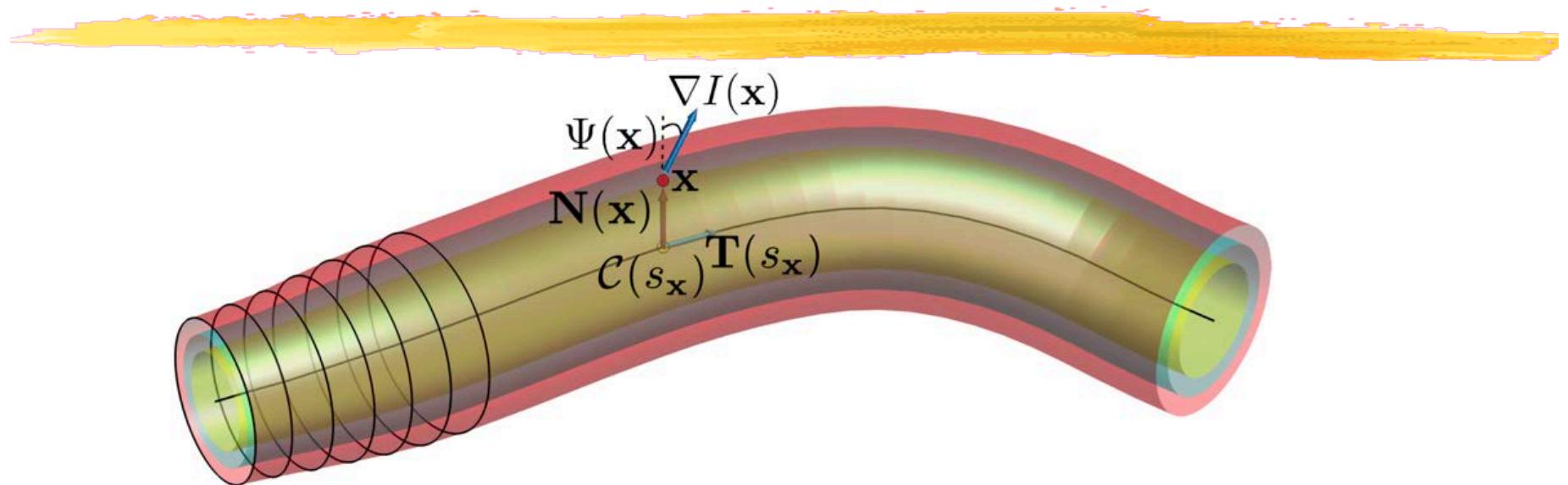
given the root note  $v_r$ .

# ROAD DELINEATION AGAIN



→ Machine plays a crucial role to ensure that the **same algorithm works in different situations.**

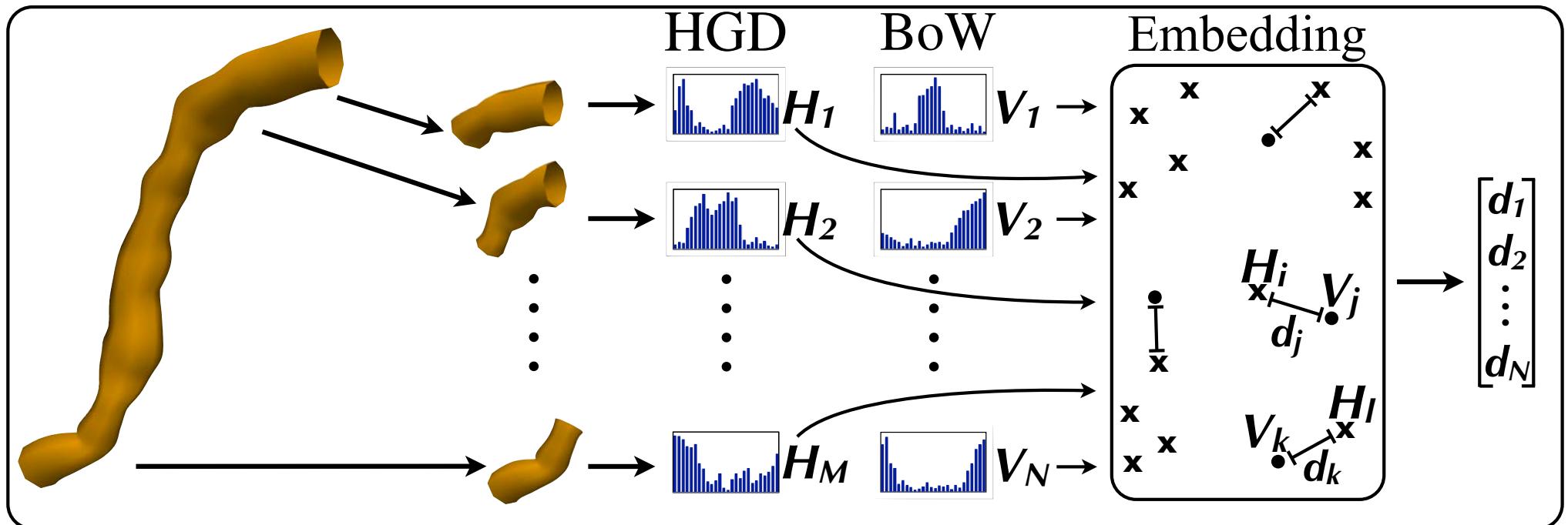
# HISTOGRAM OF GRADIENT DEVIATIONS



$$\Psi(\mathbf{x}) = \begin{cases} \text{angle}(\nabla I(\mathbf{x}), \mathbf{N}(\mathbf{x})) , & \text{if } \|\mathbf{x} - \mathcal{C}(s_{\mathbf{x}})\| > \varepsilon \\ \text{angle}(\nabla I(\mathbf{x}), \Pi(\mathbf{x})) , & \text{otherwise,} \end{cases}$$

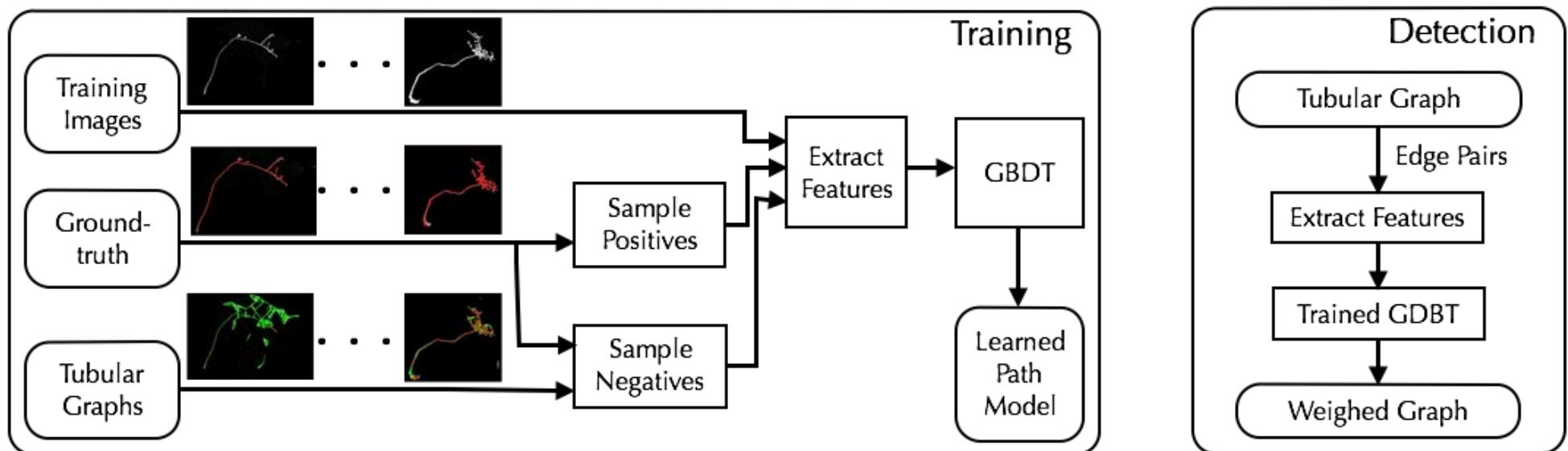
→ One histogram per radius interval plus four geometry features (curvature, tortuosity, ....)

# EMBEDDING

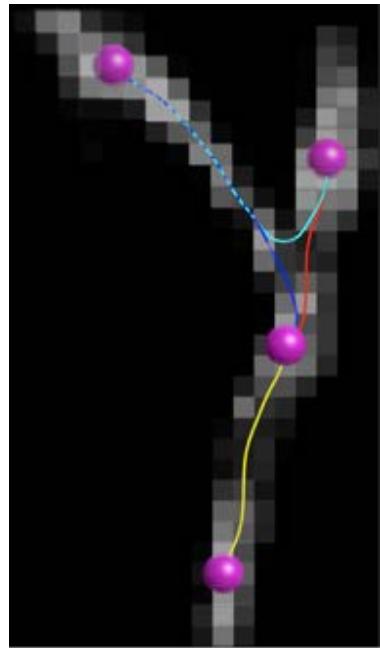


→ Same length feature vectors whatever the actual length of the path.

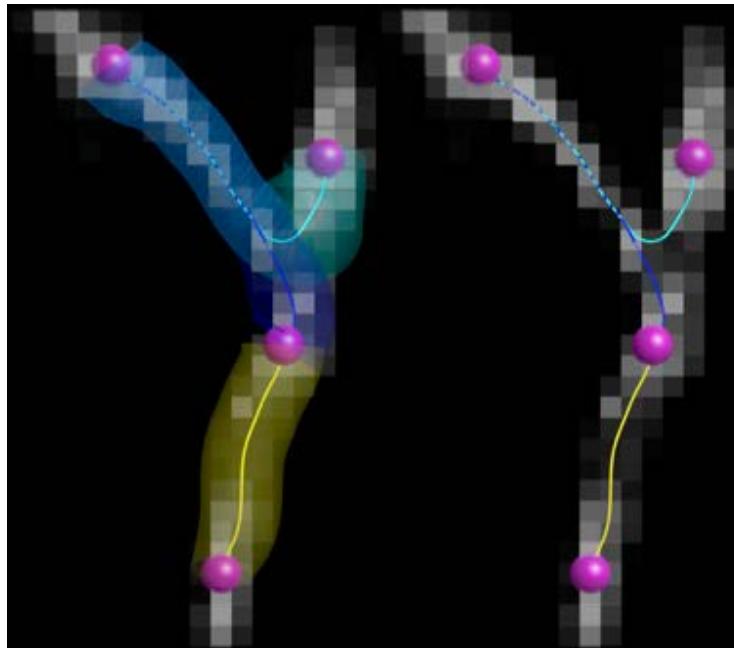
# PATH CLASSIFICATION



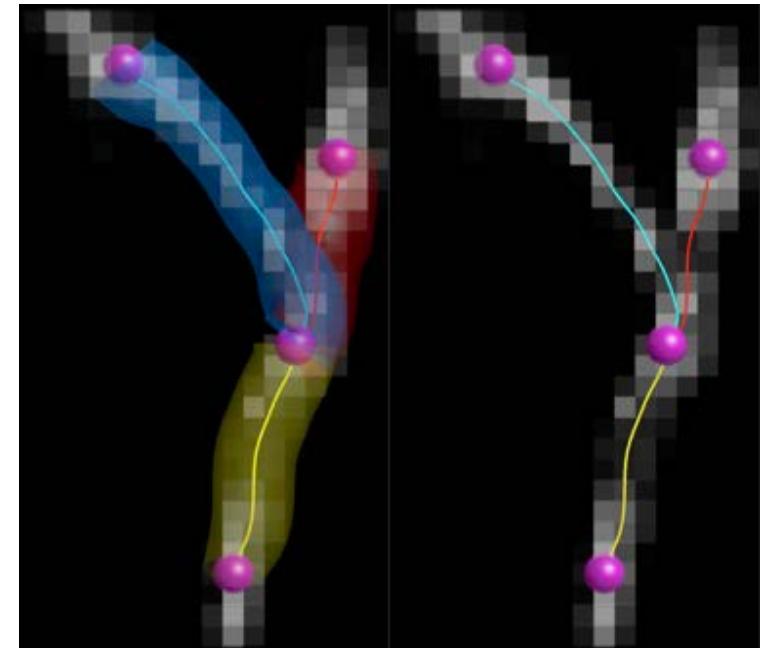
# FINDING THE BEST TREE



Tubularity graph



Without edge pair term



With edge pair term

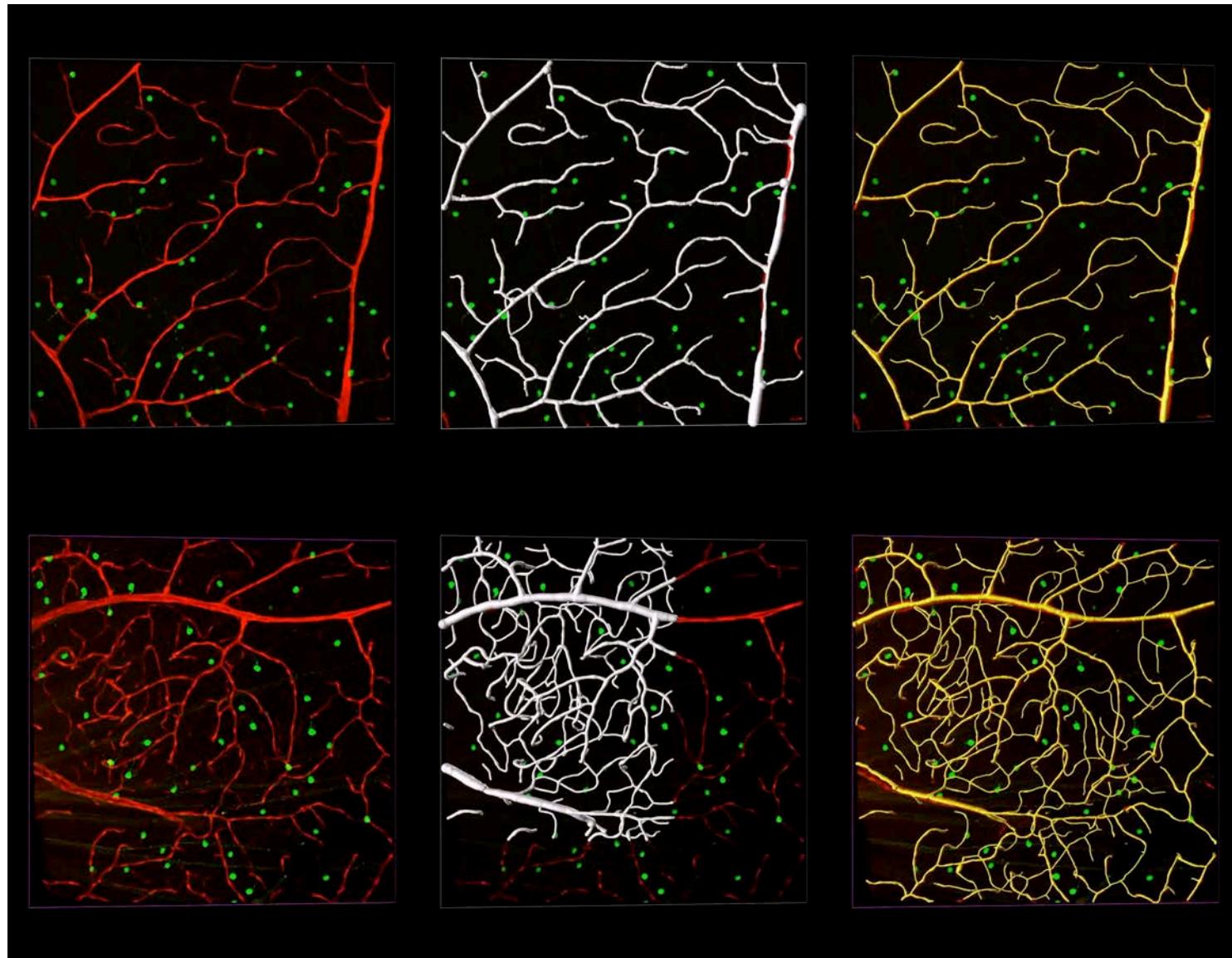
$$\begin{aligned} \mathbf{t}^* &= \underset{\mathbf{t} \in \mathcal{T}(G)}{\operatorname{argmax}} P(\mathbf{T} = \mathbf{t} | I) , \\ &= \underset{\mathbf{t} \in \mathcal{T}(G)}{\operatorname{argmax}} P(I | \mathbf{T} = \mathbf{t}) P(\mathbf{T} = \mathbf{t}) , \\ &= \underset{\mathbf{t} \in \mathcal{T}(G)}{\operatorname{argmin}} \sum_{e_{ij} \in G} c_{ij}^d t_{ij} \end{aligned}$$

# BRAINBOW IMAGES

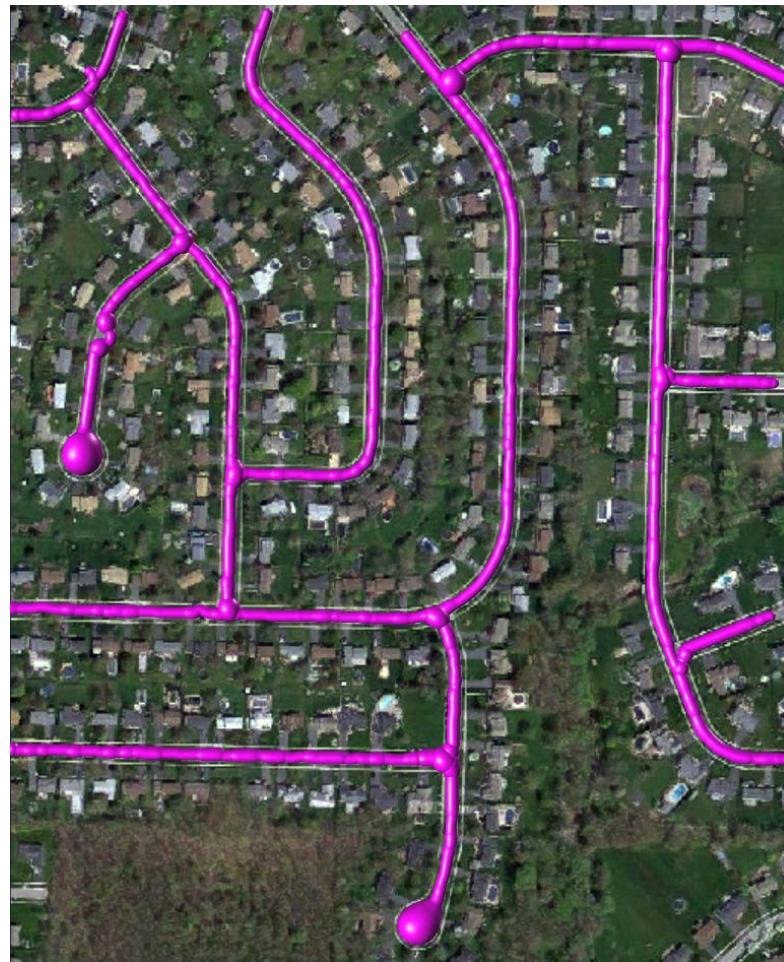


Turetken et al. , CVPR'13

# BLOOD VESSELS



# IMPROVED ROADS



# **IN SHORT**



- Edge and image information is noisy.
  - Models are required to make sense of it.
- An appropriate combination of graph-based techniques, Machine Learning, and semi-automated tools is required.