DELINEATION

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

FROM GRADIENTS TO OUTLINES

FROM DEEP-NETS TO OUTLINES

(a) original image (b) ground truth (c) HED: output

—> Still work to do!

MAPPING AND OVERLAYS

Connectivity matters!

CONNECTOMICS

—> Topology needed

Courtesy of C. Petersen

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

ROAD IMAGE

INTERACTIVE DELINEATION

LIVE WIRE

Mortensen and Barrett, SIGGRAPH, 1995

1D DYNAMIC PROGRAMMING

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

$$h(x_k, x_k + 1) = \mathrm{diff}(\phi(x_k), \phi(x_{k+1})) \ ,$$

where ϕ denotes the gradient orientation.

1D DYNAMIC PROGRAMMING

- N Locations
- Q Quantized values

→ Global optimum $O(NQ^2)$

1D DYNAMIC PROGRAMMING

To find

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

where

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

define

$$\begin{aligned}
f_1(x_2) &= \min_{x_1} (h(s, x_1) + h(x_1, x_2)) \\
f_2(x_3) &= \min_{x_2} (h(x_2, x_3) + f_1(x_2)) \\
\vdots \quad & \quad \vdots \quad \vdots \\
f_{n-1}(x_n) &= \min_{x_{n-1}} (h(x_{n-1}, x_n) + f_{n-2}(x_{n-1}))
\end{aligned}$$

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

2D DYNAMIC PROGRAMMING

Notations:

s	Start point
L	List of active nodes
c(u,v)	Local costs for link u -> v
d(v)	Total cost from s to v

DIJKSTRA'S ALGORITHM

Initialization:

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

$T = \varnothing$

$v = s$

Loop until goal is reached:

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

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

if $d(v) + c(v,u) < d(u)$

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

end

end

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

Maintain a sorted list of paths

LIVE WIRE

- Sorting is the expensive operation. Normally nlog(n), but can be reduced to log(n) if all costs are integer costs

- Local costs computed using gradient:

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

- 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

(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

ITERATING

At every step:

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

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

→ Solve two linear equations at each iteration.

DERIVATIVES OF THE GRADIENT

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

$$\frac{\partial E_G}{\partial X} = \left[\frac{\partial E_G}{\partial x_1} \quad \cdots \quad \frac{\partial E_G}{\partial x_N}\right] \quad , \quad \frac{\partial E_G}{\partial Y} = \left[\frac{\partial E_G}{\partial y_1} \quad \cdots \quad \frac{\partial E_G}{\partial y_N}\right]$$

$$\frac{\partial E_G}{\partial x_i} = -\frac{1}{N}\frac{\partial G}{\partial x_i}(x_i, y_i) \quad , \quad \frac{\partial E_G}{\partial y_i} = -\frac{1}{N}\frac{\partial G}{\partial y_i}(x_i, y_i)$$

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

NETWORK SNAKES

--> Updated field boundaries.

Butenuth, Phd 2008

RIBBON SNAKES

$$
\begin{aligned}
E &= E_G + 1/2 X^t K X + 1/2 Y^t K Y \\
W &= [w_1, \ldots, w_N]^t \\
K_W &= \begin{bmatrix}
. & . & . & . & . & . & . & . & . & . & . \\
. & . & . & -1 & 2 & -1 & . & . & . & . & . \\
. & . & . & . & -1 & 2 & -1 & . & . & . & . \\
. & . & . & . & . & -1 & 2 & -1 & . & . & . \\
. & . & . & . & . & . & . & . & . & . & .
\end{bmatrix}
\end{aligned}
$$

DYNAMICS EQUATIONS

$$
\begin{aligned}
(K + \alpha I)X_t &= \alpha X_{t-1} - \frac{\delta E_G}{\delta X} \\
(K + \alpha I)Y_t &= \alpha Y_{t-1} - \frac{\delta E_G}{\delta Y} \\
(K + \alpha I)W_t &= \alpha W_{t-1} - \frac{\delta E_G}{\delta W}
\end{aligned}
$$

→ Solve three linear equations at each iteration.

DELINEATING ROADS

DELINEATING ROADS

DELINEATING ROADS

EVALUATION

400
300
200
100
0
hand
tracker
smart1
smart2
Number of object clicks
350
300
250
200
150
100
50
0
hand
tracker
smart1
smart2
Number of mouse moves
15000
10000
5000
0
hand
tracker
smart1
smart2
Total mouse distance
1200
1000
800
600
400
200
0
hand
tracker
smart1
smart2
Total mouse move time
600
500
400
300
200
100
0
hand
tracker
smart1
smart2
Total near mouse move time

RIDGE LINES

BUILDING

Ft Hood 2x3
Fort Hood Image-4
Fort Hood Image-3
Fort Hood Image-4
Fort Hood Image-3
Fort Hood Image-4
Fort Hood Image-3

3—D SNAKES

Smooth 3—D snake

Rectilinear 3—D snake

DYNAMICS EQUATIONS

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

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

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

→ Solve three linear equations at each iteration.

CONSTRAINED OPTIMIZATION

Minimize F(S) subject to C(S) = 0

SITE MODELING

REAL-TIME 3D SHAPE

Ostlund et al., ECCV'12.

SAILS

→ Real-time performance evaluation.

WINGS

LEVEL SETS

CURVE EVOLUTION

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

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

Special cases:

$$\text{Prairie fire} \quad \beta = \pm 1$$

$$\text{Diffusion} \quad \beta = \beta_0 - \beta_1 \kappa$$

Problems:

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

LEVEL SETS

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

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

Corresponding evolution equation.

$$\begin{aligned} 0 &= \Phi_t + \beta(\kappa)\,|\nabla\Phi| \\ \text{where } \kappa &= \frac{\Phi_{xx}\Phi_y^2 - 2\Phi_{xy}\Phi_x\Phi_y + \Phi_{yy}\Phi_x^2}{\Phi_x^2 + \Phi_y^2} \end{aligned}$$

→ Much better numerical stability.

TOPOLOGY CHANGE

LEVEL SET SMOOTHING

Smoothing occurs when $\beta(\kappa) = -\kappa$

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

$$\text{where: } \beta(\kappa) = k_I(1 - \epsilon\kappa)$$

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

→ 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.
- 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

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

VOTING SCHEME

VOTING SCHEME

VOTING SCHEME

REAL WORLD LINES

Canny Edge

GENERIC ALGORITHM

- Quantize parameter space with 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:

$$\begin{aligned} x &= x_0 + r\cos(\theta) \\ y &= y_0 + r\sin(\theta) \end{aligned}$$

Therefore:

$$\begin{aligned} x_0 &= x - r\cos(\theta) \\ y_0 &= y - r\sin(\theta) \end{aligned}$$

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:

$$\begin{aligned} x &= x_0 + a\cos(\theta) \\ y &= y_0 + b\sin(\theta) \end{aligned}$$

Therefore:

$$\begin{aligned} x_0 &= x - a\cos(\theta) \\ y_0 &= y - b\sin(\theta) \end{aligned}$$

GRADIENT ORIENTATION

$$\frac{dx}{d\theta} = -a\sin(\theta)$$

$$\frac{dy}{d\theta} = \quad b\cos(\theta)$$

$$\phi = \mathrm{atan}(-\frac{dx}{d\theta}, \frac{dy}{d\theta})$$

$$= \mathrm{atan}(a\sin(\theta), b\cos(\theta))$$

$$= \mathrm{atan}(\frac{a}{b}\tan(\theta))$$

$$\tan(\phi) = \frac{a}{b}\tan(\theta)$$

$$\Rightarrow \theta = \mathrm{atan}(\frac{b}{a}\tan(\phi))$$

The accumulator need only be incremented for this θ.

GENERALIZED HOUGH

Finding a becher …

… or a lake.

GENERALIZED HOUGH

- We want to find a shape defined by its boundary points in terms of the location of a reference point $[x_c,y_c]$.
- For every boundary point p, we can compute the displacement vector $r = [x_c,y_c] - p$ as a function of local gradient orientation ϕ.

Ballard. , PR'81

R-TABLE

ϕ_1	$(r^1_1,\alpha^1_1),(r^1_2,\alpha^1_2),\ldots,(r^1_{n1},\alpha^1_{n1})$
ϕ_2	$(r^2_1,\alpha^2_1),(r^2_2,\alpha^1_2),\ldots,(r^2_{n2},\alpha^1_{n2})$
.	.
.	.
.	.
ϕ_m	$(r^m_1,\alpha^m_1),(r^m_2,\alpha^m_2),\ldots,(r^m_{nm},\alpha^m_{nm})$

$$x_c = x_i + r_i\cos(\alpha_i)$$

$$y_c = y_i + r_i\sin(\alpha_i)$$

Set of potential displacement vectors r,α given the boundary orientation ϕ.

--> 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

FROM DELINEATION TO DETECTION

Training image

Visual codeword with displacement vectors

- Instead of indexing displacements by gradient orientation, index by "visual codeword".

Leibe & al. , ECCV'04

FROM DELINEATION TO DETECTION

Test image

- Instead of indexing displacements by gradient orientation, index by “visual codeword”.

Leibe & al. , ECCV’04

TRAINING

1. Build codebook of patches around extracted interest points using clustering.
2. Map the patch around each interest point to closest codebook entry.
3. For each codebook entry, store all positions it was found, relative to object center.

--> Build an R table.

TESTING

1. Given test image, extract patches, match to codebook entry.
2. Cast votes for possible positions of object center.
3. Search for maxima in voting space.
4. Extract weighted segmentation mask based on stored masks for the codebook occurrences.

PEDESTRIAN DETECTION

Gall & al. , PAMI'11

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

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

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

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

MINIMUM SPANNING TREE

Image modeled as a graph:

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

DELINEATION 1998

Detect road centerlines

Find generic paths

Apply semantic filter

Find road widths

Fischler & Heller, 1998.

FROM IMAGE TO ROADS

ROAD DELINEATION

Fischler & Heller, 1998.

ROAD EDITING

DENDRITES AND AXONS

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

DELINEATION IN 2011

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

.. AND ROADS

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

Turetken et al, CVPR'12.

HISTOGRAM OF GRADIENT DEVIATIONS

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

—> 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

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

QMIP FORMULATION

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

given the root note v_r.

ROADS

BRAINBOW IMAGES

BLOOD VESSELS

DEEP TSUNAMI

The New York Times

Turing Award Won by Three Pioneers in Artificial Intelligence

From left, Yann LeCun, Geoffrey Hinton and Yoshua Bengio. The researchers worked on key developments for neural networks, which are reshaping how computer systems are built.

AlexNet (2012)

Task: Image classification

Training images: Large Scale Visual Recognition Challenge 2010

Training time: 2 weeks on 2 GPUs

Major Breakthrough: Training large networks has now been shown to be practical!!

DELINEATION 2018

Train Encoder-decoder U-Net architecture using binary cross-entropy

Minimize

$$L_{bce}(\mathbf{x}, \mathbf{y}; \mathbf{w}) = -\frac{1}{i}\sum_{1}^{P}[y_n \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

where

- $\hat{\mathbf{y}} = f_{\mathbf{w}}(\mathbf{x})$,
- $\mathbf{x}$ in an input image,
- $\mathbf{y}$ the corresponding ground truth.

Mosinska et al, CVPR'18.

NETWORK OUTPUT

Image

BCE Loss

Ground truth

ACCOUNTING FOR TOPOLOGY

Minimize

$$
\begin{aligned}
L(\mathbf{x}, \mathbf{y}; \mathbf{w}) &= L_{bce}(\mathbf{x}, \mathbf{y}; \mathbf{w}) + L_{top}(\mathbf{x}, \mathbf{y}; \mathbf{w}) \ , \\
L_{top}(\mathbf{x}, \mathbf{y}; \mathbf{w}) &\propto \sum_{n=1}^{N} \sum_{m=1}^{M_n} \| l_n^m(\mathbf{y}) - l_n^m(f(\mathbf{x}, \mathbf{w})) \|_2^2 \ ,
\end{aligned}
$$

L_{top}: Sum of square differences between feature maps in a pretrained VGG.

ACCOUNTING FOR TOPOLOGY

Image

BCE Loss

BCE andTopo Loss

Ground truth

ITERATIVE REFINEMENT

Image | Iter 1 | Iter 2 | Iter 3 | Ground truth

Use <u>the same network</u> to progressively refine the results keeping the number of parameters constant

DELINEATION STEPS

1. Compute a probability map.
2. Sample and connect the samples.
3. Assign a weight to the paths.
4. Retain the best paths.

DUAL USE U-NET

STREETS OF TORONTO

— False negatives
— False positives

DENDRITES AND AXONS

1998 - 2038

1998

2018

2038

It is difficult to make predictions, especially about the future.
Sometimes attributed to Niels Bohr.

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.