Precise Thresholding

- or, a small Matlab package called Abmask

Anders Brun
Centre for Image Analysis
Swedish University of Agricultural Sciences
Uppsala University

Basic Ideas...

- Given a function \( u(x) \)
- Thresholding:
  - if \( u(x) > o \): \( T(u(x)) = 1 \)
  - else: \( T(u(x)) = 0 \)
- Soft version:
  - if \( u(x) > o \) inside pixel \( x \): \( T(u(x)) = 1 \)
  - else if \( u(x) \leq o \) inside pixel \( x \): \( T(u(x)) = 0 \)
  - else: \( T(u(x)) \) in \( [0,1] \)

What is \( u(x) \) inside a pixel?

- Function Value
- Function Gradient
- Linear Model

\[ \ddot{u}(x+dx) = u(x) + \text{grad}(u(x)) \cdot dx \]

What is \( u(x) \) inside a pixel?

- Cons: A linear model introduces a bias
- Pros: We can estimate a precise “coverage”

abmask1

- abmask1(\( u \), softness)
- In 1D we have partial coverage iff:
  \[ \frac{u'(x)}{2} > \text{abs}(x) \]
- Precise coverage from simple code:

```
fill = zeros(size(x));
fill((x - 0.5)*u > 0) & (x + 0.5)*u < 0) = 1;
fill((x - 0.5)*u > 0) & (x + 0.5)*u < 0) = 1;
idx = (x - 0.5)*u > 0) & (x + 0.5)*u < 0) = 1;
fill(idx) = (0.5*gradx(idx)-u(idx))./gradx(idx);
```

abmask2

- abmask2(\( u \), softness)
- In 2D we have / have not partial coverage if:
  \[ \text{abs}(\text{grad} u(x)) / \text{abs}(u(x)) \]
  or
  \[ \text{grad} u(x) / \text{sqrt}(\text{grad} u(x)^2 + \text{grad} u(x)^2) \]
- Precise coverage from trigonometry tricks:

```
theta = angle(gradient(u));
theta = pi/4 - \text{abs}(\text{mod}(\text{angle}(gradx+i*grady),pi/2)-pi/4);
x = u./sqrt(gradx.^2 + grady.^2);
a = -1/sqrt(2)*\cos(pi/4-theta);
d = -a;
b = -1/sqrt(2)*\sin(pi/4-theta);
c = -b;
fill = zeros(size(u));
m = x <= a;
fill(m) = double(u(m)>0);
m = (x > a) & (x <= b);
fill(m) = 0.5*(x(m)-a(m)).^2./(cos(theta(m)).*(b(m)-a(m)));
m = (x > b) & (x <= c);
fill(m) = 0.5*(b(m)-a(m))./cos(theta(m)) + (x(m)-b(m))./cos(theta(m));
m = (x > c) & (x < d);
fill(m) = 1 - 0.5*(-x(m)-a(m)).^2./(cos(theta(m)).*(b(m)-a(m)));
m = x >= d;
fill(m) = double(u(m)>0) ;
```
abmask3

- abmask3(u, softness)
- In 3D we have / have not partial coverage if:
  \[ |\nabla u(x)| \leq \text{softness} \] or \[ |\nabla u(x)| \leq \text{softness} \]
- Precise coverage from divide and conquer…
  - Divide voxel into 5 tetrahedra (simplices)
  - Compute precise coverage for each simplex & sum

Sub Pixel Precision is non-linear!

- Because of all the geometric cases involved, essentially the rotation variance of the pixel (it is a square, it is not round), sub pixel accuracy using linear models inside pixels yields a non-linear expression for the coverage inside a pixel.
- Could there be another representation of the image / gradient where the coverage is a linear function?

Softness, what?

- Softness:
  - Multiplies the gradient with a factor. High gradient yields a higher probability of partial coverage.
  - The mismatch between original function values and artificially larger gradients makes the fuzzy border bigger! Bug or feature?
- If softness > 1: soft border wider than > 1 pixel
- If 0 < softness < 1: more crisp border

Gradient, what?

- The gradient is either
  - Estimated from numerical differentiation of the function or
  - Provided analytically, because it is know to the user and then we can avoid the extra smoothing a numerical differentiation might give

“Precise” Enables Differentiation

- Enables numerical differentiation:
  - Compute volume of sphere with radius 0.500000001
  - Compute volume of sphere with radius 0.500000000
  - Divide the difference with 0.00000001
  - This is an estimate of the surface area
- Applies to surface area (3D) and circumference (2D) of arbitrary shapes
- Thresholding or sampled coverage... try! :-)

“Precise” Enables Differentiation

Increasing threshold moves the levelset curve
“The Eikonal equation”
Going from threshold T to T - dt:

\[ \text{moves curve segment } dN = dt/|\nabla u(x)| \]

\[ u(x) \]

\[ x \]

\[ dT \]
“Precise” Enables Differentiation

Thus, in 2D, area increases locally by $dN \cdot dl$, where $dl$ is the curve segment length inside the pixel.

So... we can measure circumference or surface area by this simple expression:

$$\text{Sum}((\text{abmask}(u-dt, t) - \text{abmask}(u, t)) \cdot dl$$

“Divide the band with its width and integrate”

Open Questions

• Generalization to N-D (N>3) and other grids
  — Divide and Conquer via N-D simplices is one way to go here...

• And hey... didn’t we throw away a little too much when we forgot the gradient direction?

• Given both coverage (a bitmap with values 0...1) and gradient direction, we have all information about the linear model inside every pixel. Useful?