Fast approximate curve evolution

James Malcolm   Yogesh Rathi
Anthony Yezzi   Allen Tannenbaum

Georgia Institute of Technology, Atlanta, GA
Brigham and Women's Hospital, Boston, MA

Introduction

- Curve evolution is robust technique for variational image segmentation

Introduction

- Curve evolution is robust technique for variational image segmentation
  - Active contours, level set methods

Introduction

- Curve evolution is robust technique for variational image segmentation
  - Active contours, level set methods
- Numerics involved make its use in high performance systems computationally prohibitive, e.g. upwind derivatives

$$g(x,y)|\nabla\phi| = \big(\max(g(x,y),0)\cdot\phi_x^{+} + \min(g(x,y),0)\cdot\phi_x^{-2} + \max(g(x,y),0)\cdot\phi_y^{+} + \min(g(x,y),0)\cdot\phi_y^{-2}\big)^{1/2}$$

Introduction

- Curve evolution is robust technique for variational image segmentation
  - Active contours, level set methods
- Numerics involved make its use in high performance systems computationally prohibitive, e.g. upwind derivatives

$$
\begin{aligned}
g(x,y)|\nabla\phi| = (& \max(g(x,y),0)\cdot\phi_x^{+} & + \min(g(x,y),0)\cdot\phi_x^{-2} \\
+ & \max(g(x,y),0)\cdot\phi_y^{+} & + \min(g(x,y),0)\cdot\phi_y^{-2})^{1/2}
\end{aligned}
$$

- We propose an approximate curve evolution scheme that is simple, fast, and accurate.

Energy minimization

- Formulate image segmentation as energy minimization problem

$$E(x) = E_{data}(x) + E_{smoothness}(x)$$

Energy minimization

- Formulate image segmentation as energy minimization problem

$$E(x) = E_{data}(x) + E_{smoothness}(x)$$

- Find the curve that optimally separates object from background

$$E(C) = E_{data}(C) + E_{smoothness}(C)$$

Equivalently use a signed distance function ϕ

$$E(\phi) = E_{data}(\phi) + E_{smoothness}(\phi)$$

Energy minimization

- Formulate image segmentation as energy minimization problem

$$E(x) = E_{data}(x) + E_{smoothness}(x)$$

- Find the curve that optimally separates object from background

$$E(C) = E_{data}(C) + E_{smoothness}(C)$$

Equivalently use a signed distance function ϕ

$$E(\phi) = E_{data}(\phi) + E_{smoothness}(\phi)$$

- Iteratively deform curve to minimize energy

$$\nabla E(\phi) = (g(\phi) + \mathcal{K}) \cdot \frac{\nabla \phi}{|\nabla \phi|}$$

Numerical implementation

- Signed distance function adhering to $|\nabla \phi| = 1$

Numerical implementation

- Signed distance function adhering to $|\nabla\phi| = 1$
- Nonlinear partial differential equation

Numerical implementation

- Signed distance function adhering to $|\nabla\phi| = 1$
- Nonlinear partial differential equation
- Stability requires upwinding, finite differencing schemes, forward Euler updates, regularization, etc.

Related work

Various techniques proposed, each with tradeoffs:

- Perform numerical computations only in narrow band around interface (Adalsteinson and Sethian 1995, Whitaker 1998)

Related work

Various techniques proposed, each with tradeoffs:

- Perform numerical computations only in narrow band around interface (Adalsteinson and Sethian 1995, Whitaker 1998)
    - Reduced domain, but still full numerics

Related work

Various techniques proposed, each with tradeoffs:

- Perform numerical computations only in narrow band around interface (Adalsteinson and Sethian 1995, Whitaker 1998)
  - Reduced domain, but still full numerics
- Binary representation (Gibou and Fedkiw 2005)

Related work

Various techniques proposed, each with tradeoffs:

- Perform numerical computations only in narrow band around interface (Adalsteinson and Sethian 1995, Whitaker 1998)
  - Reduced domain, but still full numerics
- Binary representation (Gibou and Fedkiw 2005)
  - Removes much of numerics, but force computed on entire domain

Related work

Various techniques proposed, each with tradeoffs:

- Perform numerical computations only in narrow band around interface (Adalsteinson and Sethian 1995, Whitaker 1998)
  - Reduced domain, but still full numerics
- Binary representation (Gibou and Fedkiw 2005)
  - Removes much of numerics, but force computed on entire domain
- Discrete representation and list switching (Shi and Karl 2005)

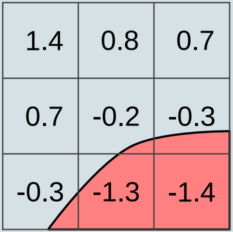
Related work

Various techniques proposed, each with tradeoffs:

- Perform numerical computations only in narrow band around interface (Adalsteinson and Sethian 1995, Whitaker 1998)
  - Reduced domain, but still full numerics
- Binary representation (Gibou and Fedkiw 2005)
  - Removes much of numerics, but force computed on entire domain
- Discrete representation and list switching (Shi and Karl 2005)
  - Removes numerics, but requires interpolation off the interface and computation of the force twice

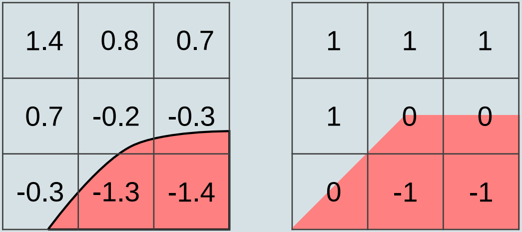
Continuous v. Discrete

Continuous representation



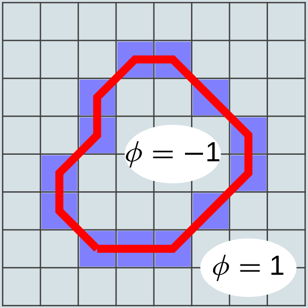
Continuous v. Discrete

Continuous representation — Discrete approximation



Discrete only uses: $\phi = -1$ inside, $\phi = 0$ interface, $\phi = 1$ outside.

Assumption: Subpixel error makes little difference at the macro level



Algorithm Overview

1. Based on energy, compute force only along interface
2. Points are propagated according to the force
3. Interface is cleaned up
4. Regional statistics are updated

Algorithm Overview

1. Based on energy, compute force only along interface
2. Points are propagated according to the force
3. Interface is cleaned up
4. Regional statistics are updated

- Each iteration has two phases: dilation, contraction

Algorithm Overview

1. Based on energy, compute force only along interface
2. Points are propagated according to the force
3. Interface is cleaned up
4. Regional statistics are updated

- Each iteration has two phases: dilation, contraction
- This work uses the discrete signed distance function: $\phi = -1$ inside, $\phi = 0$ interface, $\phi = 1$ outside.

Main loop

for each iteration **do**
  {*Contraction*}
  Callback: compute force
  Restrict to contraction (only allow positive forces)
  Propagate, Cleanup
  Callback: move points in and out
  {*Dilation*}
  Callback: compute force
  Restrict to contraction (only allow negative forces)
  Propagate, Cleanup
  Callback: move points in and out
end for

Note: Callbacks are energy specific.

Compute force

- Based on chosen energy, force is computed at each point along curve

Compute force

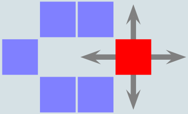
- Based on chosen energy, force is computed at each point along curve
- Only sign of this force matters

Compute force

- Based on chosen energy, force is computed at each point along curve
- Only sign of this force matters
- Discrete representation suitable for approximate first order derivatives

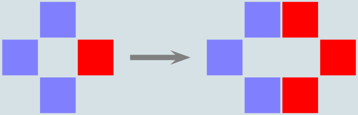
Movement of points

- Points only move in four directions: up, down, left, right

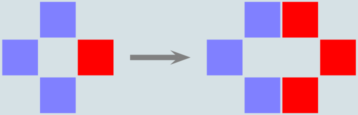


- Points move a unit distance in direction indicated by force

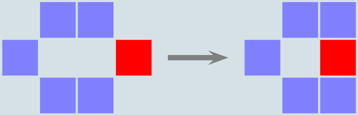
- Points move a unit distance in direction indicated by force
- Dilation



- Points move a unit distance in direction indicated by force
- Dilation

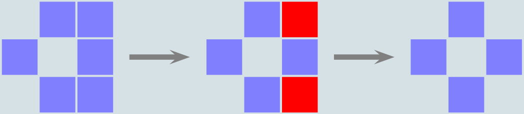


- Contraction



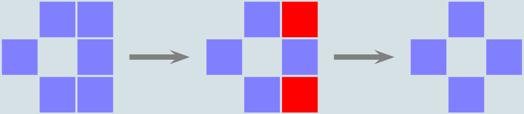
Maintaining a minimal interface

- Drop points that only touch one side of interface



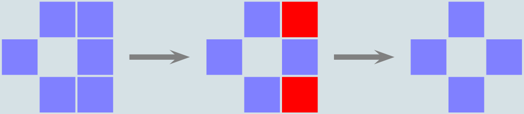
Maintaining a minimal interface

- Drop points that only touch one side of interface
  - We only need check up/down/left/right neighbors for decisions on movement



Maintaining a minimal interface

- Drop points that only touch one side of interface
  - We only need check up/down/left/right neighbors for decisions on movement
  - Prevents artifacts from developing



Incorporating an energy

Arbitrary energies defined by three functions:

Incorporating an energy

Arbitrary energies defined by three functions:

- `computeforce()` – compute positive/negative energy gradient at each point along curve, e.g. $\nabla E(C) \cdot \mathcal{N} = g(C) \cdot \mathcal{N}$.

Incorporating an energy

Arbitrary energies defined by three functions:

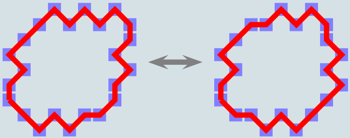
- `computeforce()` – compute positive/negative energy gradient at each point along curve, e.g. $\nabla E(C) \cdot \mathcal{N} = g(C) \cdot \mathcal{N}$.
- `movein()`, `moveout()` – update regional statistics based on specified points moving across interface

Why the two phased approach?

- Without subpixel resolution, curve can oscillate along an object boundary

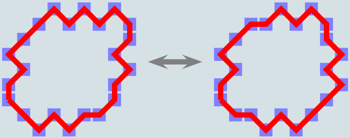
Why the two phased approach?

- Without subpixel resolution, curve can oscillate along an object boundary
- Oscillation produces jagged edges



Why the two phased approach?

- Without subpixel resolution, curve can oscillate along an object boundary
- Oscillation produces jagged edges



- Contour remains roughly in same position

Recap

for each iteration **do**
  {*Contraction*}
  Callback: compute force
  Restrict to contraction (only allow positive forces)
  Propagate, Cleanup
  Callback: move points in and out
  {*Dilation*}
  Callback: compute force
  Restrict to contraction (only allow negative forces)
  Propagate, Cleanup
  Callback: move points in and out
end for

Example energies

- This work demonstrates two statistical intensity-based energies from the literature:

Example energies

- This work demonstrates two statistical intensity-based energies from the literature:
    1. Separating regions represented by their mean intensity

Example energies

- This work demonstrates two statistical intensity-based energies from the literature:
  1. Separating regions represented by their mean intensity
  2. Separating regions represented by their full density

Example energies

- This work demonstrates two statistical intensity-based energies from the literature:
  1. Separating regions represented by their mean intensity
  2. Separating regions represented by their full density
- Can ignore $\delta(\phi)$ since operating along interface

Mean intensity separation

- Characterize both regions by the average intensity

Mean intensity separation

- Characterize both regions by the average intensity
- Energy favors regions with low variance about the mean intensity (cartoon model) (Chan and Vese 2001, Yezzi et al. 1999).

Mean intensity separation

- Characterize both regions by the average intensity
- Energy favors regions with low variance about the mean intensity (cartoon model) (Chan and Vese 2001, Yezzi et al. 1999).
- Energy and gradient:

$$E = \int (I(x) - v)^2 H(\phi(x)) + (I(x) - u)^2 H(-\phi(x)) dx$$

$$\nabla E = \delta(\phi) \left[(I(x) - v)^2 - (I(x) - u)^2 \right]$$

Mean intensity separation

- Characterize both regions by the average intensity
- Energy favors regions with low variance about the mean intensity (cartoon model) (Chan and Vese 2001, Yezzi et al. 1999).
- Energy and gradient:

$$E = \int (I(x) - v)^2 H(\phi(x)) + (I(x) - u)^2 H(-\phi(x)) dx$$

$$\nabla E = \delta(\phi) \left[(I(x) - v)^2 - (I(x) - u)^2 \right]$$

- Recursively update means as points move in and out

Full density separation

- Characterize both regions by their full intensity distribution

Full density separation

- Characterize both regions by their full intensity distribution
- Minimize the similarity between regions judged via Bhattacharyya measure (Zhang and Freedman 2003, Rathi et al. 2006)

Full density separation

- Characterize both regions by their full intensity distribution
- Minimize the similarity between regions judged via Bhattacharyya measure (Zhang and Freedman 2003, Rathi et al. 2006)
- Energy:

$$E = d^2(\mathbf{p}, \mathbf{q}) = \int_{\mathcal{Z}} \sqrt{\mathbf{p}(z)\mathbf{q}(z)}dz$$

Full density separation

- Characterize both regions by their full intensity distribution
- Minimize the similarity between regions judged via Bhattacharyya measure (Zhang and Freedman 2003, Rathi et al. 2006)
- Energy:

$$E = d^2(\mathbf{p}, \mathbf{q}) = \int_{\mathcal{Z}} \sqrt{\mathbf{p}(z)\mathbf{q}(z)}dz$$

- Simplified gradient:

$$\nabla E = \frac{d^2(\mathbf{p}, \mathbf{q})}{2}\left(\frac{1}{A_p} - \frac{1}{A_q}\right) + \frac{\delta(\phi)}{2}\left(\frac{1}{A_q}\sqrt{\frac{\mathbf{p}(z)}{\mathbf{q}(z)}} - \frac{1}{A_p}\sqrt{\frac{\mathbf{q}(z)}{\mathbf{p}(z)}}\right)$$

Extensions

- Smoothing (Gibou and Fedkiw 2005, Shi and Karl 2005)

Extensions

- Smoothing (Gibou and Fedkiw 2005, Shi and Karl 2005)
    - Alternate evolution and smoothing via Gaussian kernel

Extensions

- Smoothing (Gibou and Fedkiw 2005, Shi and Karl 2005)
  - Alternate evolution and smoothing via Gaussian kernel
- Single phase for faster evolution at the cost of jagged edges

Extensions

- Smoothing (Gibou and Fedkiw 2005, Shi and Karl 2005)
  - Alternate evolution and smoothing via Gaussian kernel
- Single phase for faster evolution at the cost of jagged edges
  - Monotonic front propagation

- Speeds depend on initialization and image size

- Speeds depend on initialization and image size
- Benchmarked at speeds ranging from 0.8-50 ms per iteration

- Speeds depend on initialization and image size
- Benchmarked at speeds ranging from 0.8-50 ms per iteration
- Convergence often in 10 or fewer iterations due to unit propagation

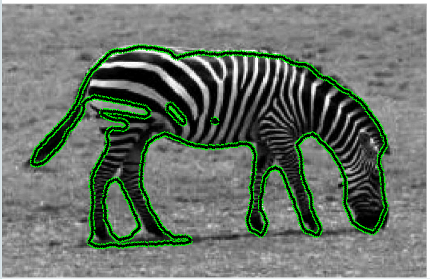
Mean intensity

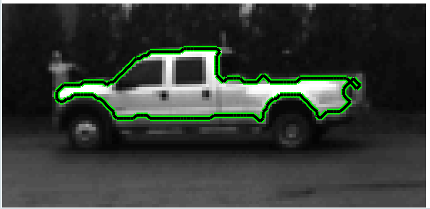
Mean intensity

Mean intensity

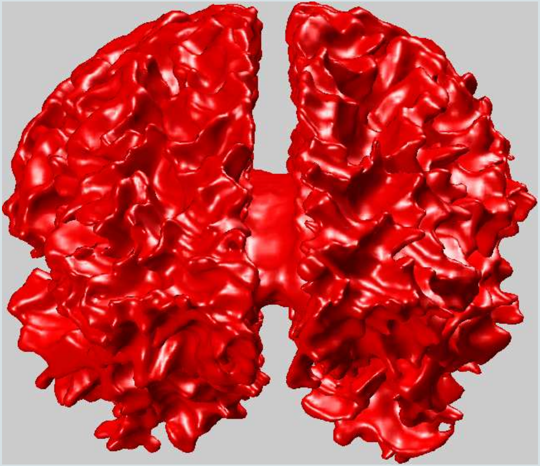
Full density

Approximation





Volumetric



Conclusion

- Discrete approximation of sign distance function

Conclusion

- Discrete approximation of sign distance function
- Simple curve mechanics

Conclusion

- Discrete approximation of sign distance function
- Simple curve mechanics
- High performance on uniprocessors

Conclusion

- Discrete approximation of sign distance function
- Simple curve mechanics
- High performance on uniprocessors

Conclusion

- Discrete approximation of sign distance function
- Simple curve mechanics
- High performance on uniprocessors

Questions?