A multigrid solver for the Rada-Chen selective segmentation model

SIAM Imaging Conference 2016

Mike Roberts, Ke Chen and Klaus Irion

25 May 2016

Centre for Mathematical Imaging Techniques, University of Liverpool

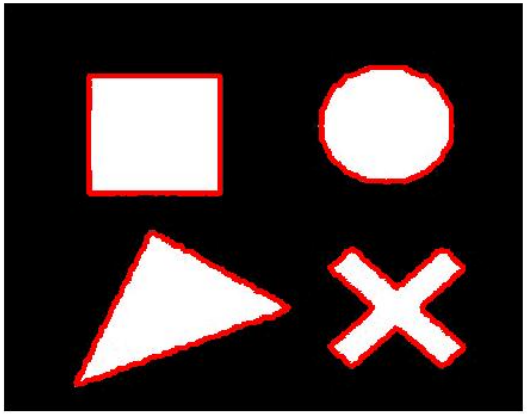
Table of contents

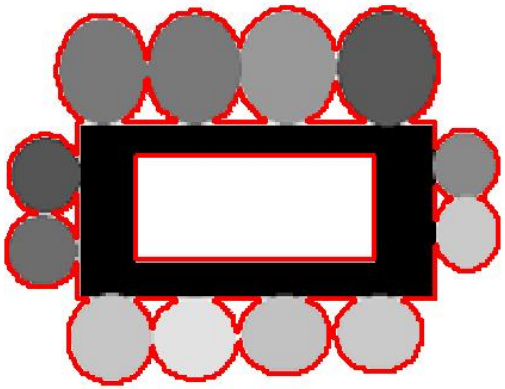
Image Segmentation

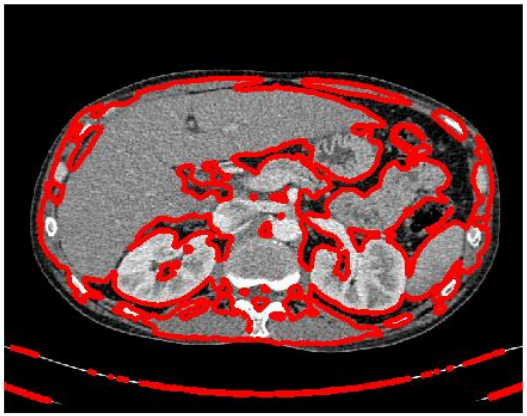
Image Segmentation

In two-phase segmentation we look for a contour Γ that separates the objects in an image into **foreground** and **background**.

Image Segmentation: Global





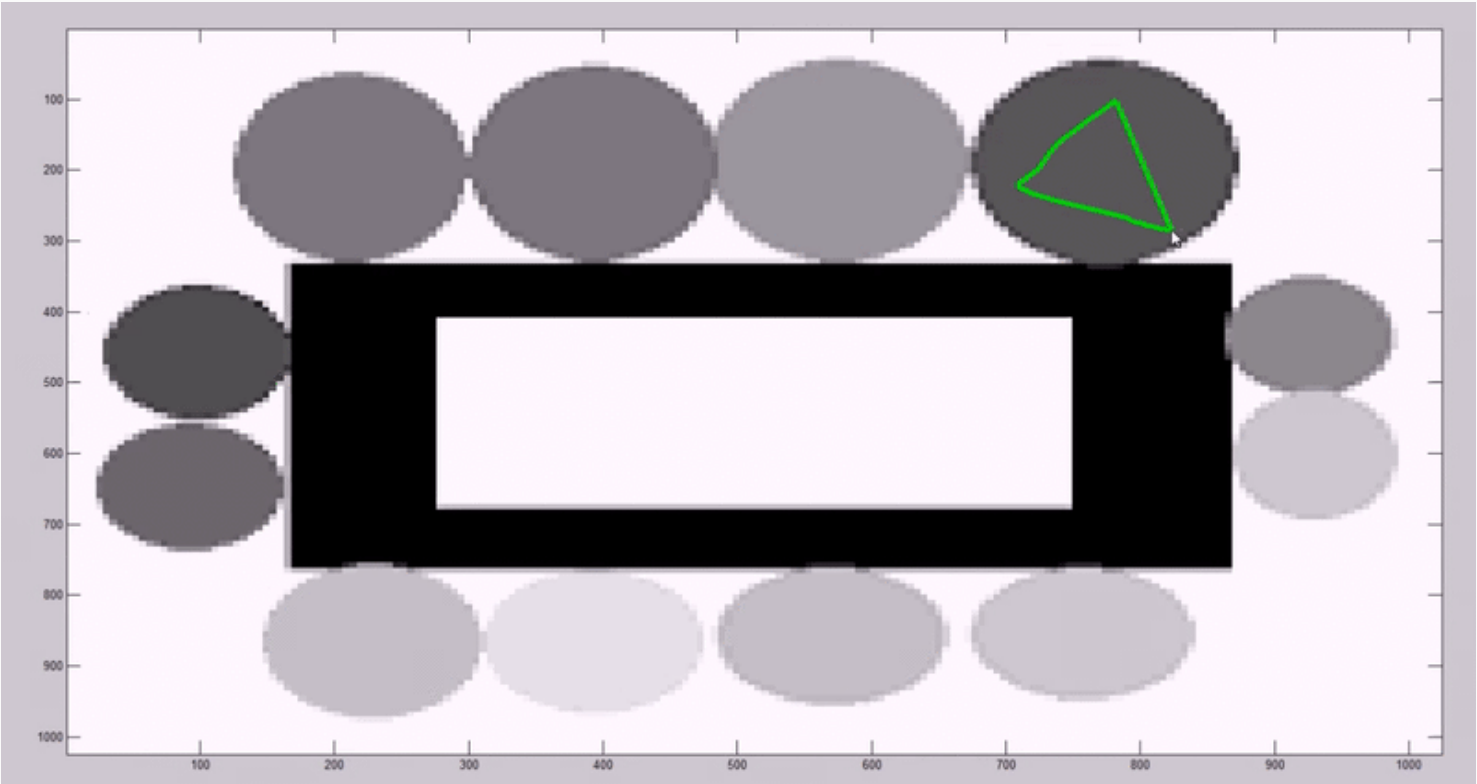


Pro: good segmentation result.

Con: result is not useful necessarily, e.g. medical imaging.

Image Segmentation: Selective

The user selects an object to segment.



$$\mathcal{S} = \{(x_i, y_i) \in \Omega, 1 \leq i \leq k\}$$

Image Segmentation: Selective - Motivational Results

Motivational medical segmentation results

Variational Approach

We use a **variational** approach, i.e. look to minimise functionals.

Our basic model is:

$$\min_{\Gamma} F(\Gamma)$$

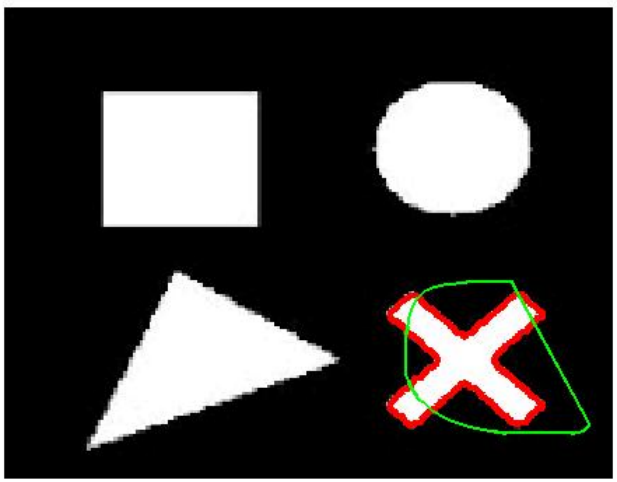
subject to some conditions on Γ enforced by functional F.

Conditions on Γ

In selective segmentation we want

1. Γ to be on the edges of objects.
2. Γ to be close to the points of $\mathcal{S}$.
3. Γ to be as short as possible.
4. the inside of Γ to have the same characteristics, e.g. intensity or texture.
5. the size of the shape to be similar to the user desires.

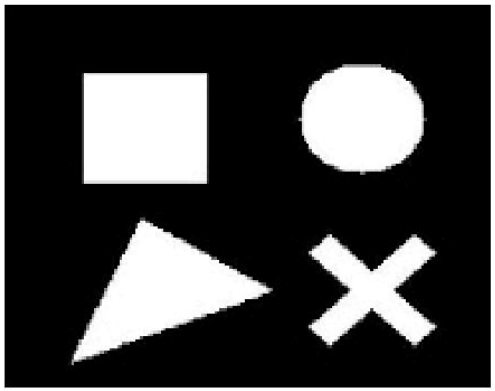
How can we enforce these through F?

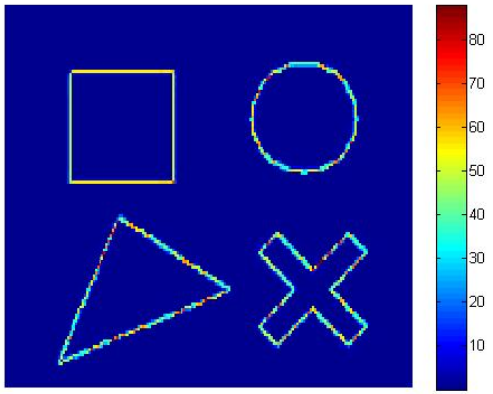


Constructing F: Edges, Distance, Short, Intensity, Size

We want a function $g \approx 0$ at edges, $g \approx 1$ in homogenous areas.

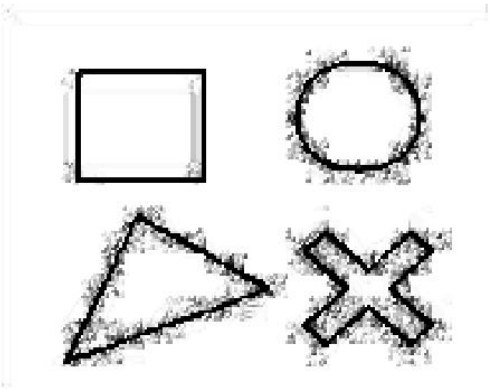
Look at the gradient $|\nabla z(x, y)|$ of the image.





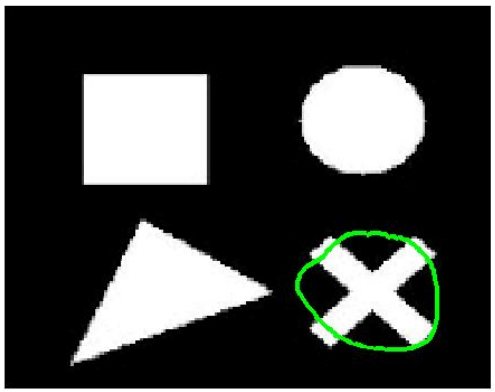
Large at edges, small away from edges.

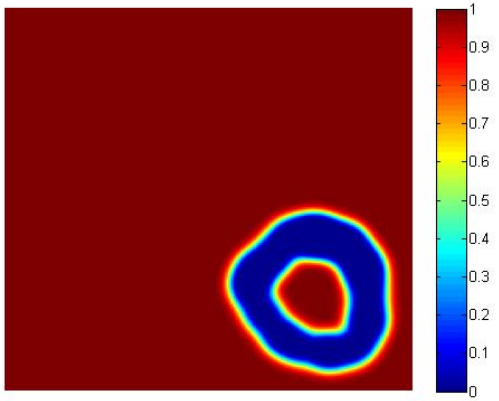
$$g(|\nabla z(x, y)|) = \frac{1}{1 + \beta |\nabla z(x, y)|^2}$$



Constructing F: Edges, Distance, Short, Intensity, Size

We want a distance function $d \approx 0$ near $\mathcal{S}$, $d \approx 1$ away from $\mathcal{S}$.





Such a function is

$$d(x,y) = \prod_{i=1}^{k} \left(1 - e^{-\frac{(x_i - x)^2}{2\sigma^2}} e^{-\frac{(y_i - y)^2}{2\sigma^2}}\right) \qquad \forall (x,y) \in \Omega, (x_i, y_i) \in \mathcal{S}.$$

σ is a fixed non-negative tuning parameter.

Constructing F: Edges, Distance, Short, Intensity, Size

We minimise the length of a contour by minimising

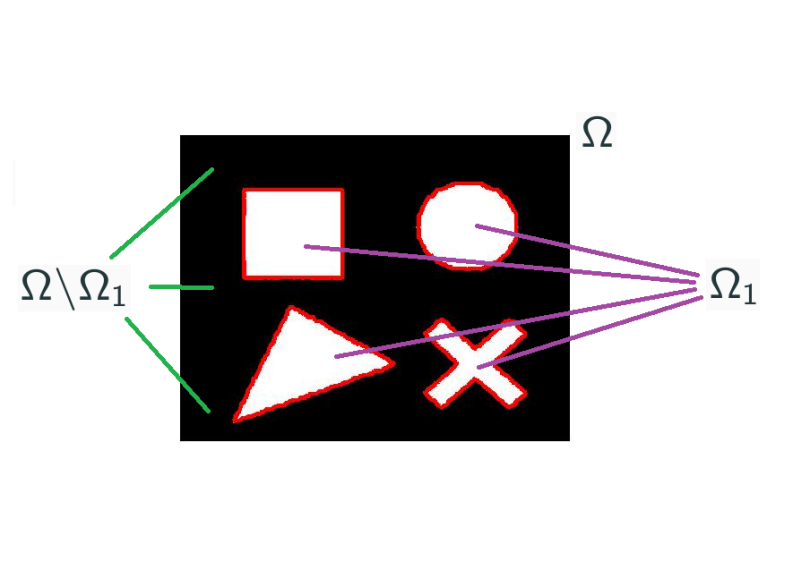
$$\int_{\Gamma} ds$$

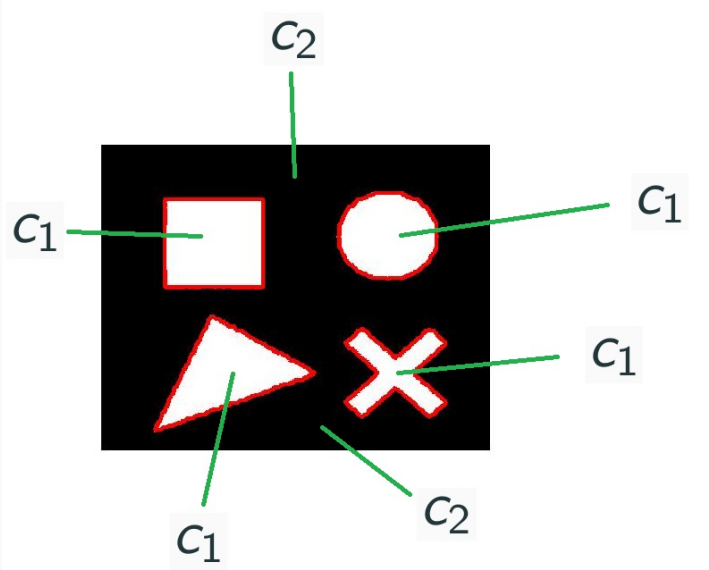
In practice, we get short Γ near image edges and the points of $\mathcal{S}$ by minimising

$$\int_{\Gamma} d(x, y) g(|\nabla z|) ds$$

Constructing F: Edges, Distance, Short, Intensity, Size

Define Ω_1 inside Γ, $\Omega \backslash \Omega_1$ outside it.





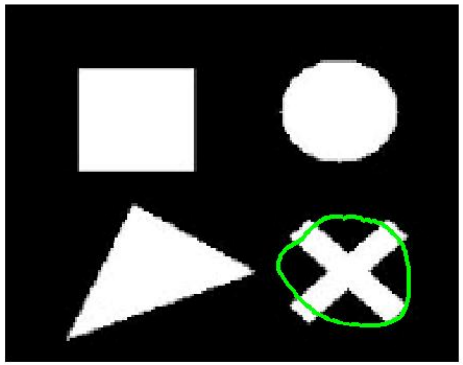
We aim to minimise

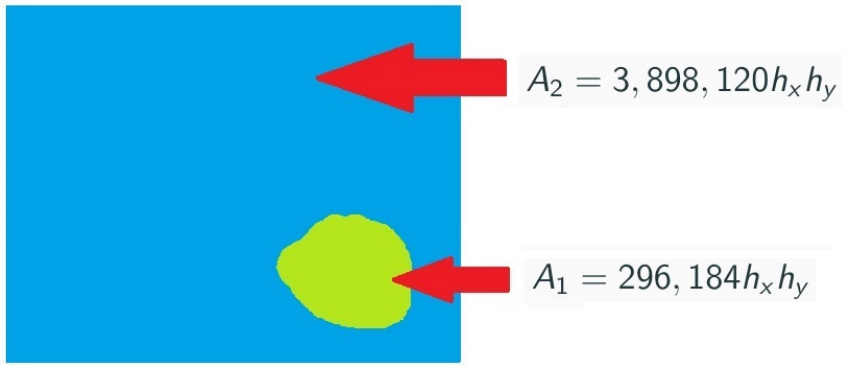
$$\lambda_1 \int_{\Omega_1} |z(x,y) - c_1|^2 \mathrm{d}x\mathrm{d}y + \lambda_2 \int_{\Omega \backslash \Omega_1} |z(x,y) - c_2|^2 \mathrm{d}x\mathrm{d}y$$

which penalise intensity differences inside and outside Γ.

Constructing F: Edges, Distance, Short, Intensity, Size

We want a term which penalises the area inside Γ being different to the user desired area.





We aim to minimise

$$\nu \left[\left(\int_{\Omega_1} \mathrm{d}x\mathrm{d}y - A_1 \right)^2 + \left(\int_{\Omega \setminus \Omega_1} \mathrm{d}x\mathrm{d}y - A_2 \right)^2 \right]$$

which penalises the area inside and outside Γ.

The Rada-Chen model

Rada and Chen (2012) introduced a new model

$$\min_{\Gamma, c_1, c_2} F_{RC}(\Gamma, c_1, c_2)$$

where

$$\begin{aligned} F_{RC}(\Gamma) = \mu \int_{\Gamma} d(x,y) g(|\nabla z(x,y)|) ds \\ + \lambda_1 \int_{\Omega_1} |z(x,y) - c_1|^2 \mathrm{d}x\mathrm{d}y + \lambda_2 \int_{\Omega \setminus \Omega_1} |z(x,y) - c_2|^2 \mathrm{d}x\mathrm{d}y \\ + \nu \left[\left(\int_{\Omega_1} \mathrm{d}x\mathrm{d}y - A_1 \right)^2 + \left(\int_{\Omega \setminus \Omega_1} \mathrm{d}x\mathrm{d}y - A_2 \right)^2 \right] \end{aligned}$$

which addresses the 5 required properties.

Minimising for Γ: Level set formulation

We have the following relations

$$\int_\Gamma ds = \int_\Omega |\nabla H_\varepsilon(\phi)| \mathrm{d}x\mathrm{d}y, \qquad \int_{\Omega_1} \mathrm{d}x\mathrm{d}y = \int_\Omega H_\varepsilon(\phi)\mathrm{d}x\mathrm{d}y,$$
$$\int_{\Omega\setminus\Omega_1} \mathrm{d}x\mathrm{d}y = \int_\Omega (1 - H_\varepsilon(\phi))\mathrm{d}x\mathrm{d}y$$

where $H_\varepsilon(\phi)$ is a regularised Heaviside.

Using level sets we can rewrite our problem as

$$\begin{aligned} F_{RC}(\phi, c_1, c_2) =& \mu \int_\Omega d(x,y) g(|\nabla z(x,y)|) |\nabla H_\varepsilon(\phi)| \mathrm{d}x\mathrm{d}y \\ &+ \lambda_1 \int_\Omega (z(x,y) - c_1)^2 H_\varepsilon(\phi)\mathrm{d}x\mathrm{d}y + \lambda_2 \int_\Omega (z(x,y) - c_2)^2 (1 - H_\varepsilon(\phi))\mathrm{d}x\mathrm{d}y \\ &+ \nu \left[\left(\int_\Omega H_\varepsilon(\phi)\mathrm{d}x\mathrm{d}y - A_1 \right)^2 + \left(\int_\Omega (1 - H_\varepsilon(\phi))\mathrm{d}x\mathrm{d}y - A_2 \right)^2 \right] \end{aligned}$$

Minimising for Γ: Euler-Lagrange form

To minimise F_{RC} we must find

$$\nabla F_{RC} = 0$$

where

$$\begin{aligned}\nabla F_{RC} = \delta_\varepsilon(\phi) \Bigg\{ & \mu \nabla \cdot \left(\frac{d(x,y) \cdot g(|\nabla z(x,y)|) \nabla \phi}{|\nabla \phi|} \right) \\ & - \left(\lambda_1 (z(x,y) - c_1)^2 - \lambda_2 (z(x,y) - c_2)^2 \right) \\ & - \nu \left[\left(\int_\Omega H_\varepsilon(\phi) \mathrm{d}x\mathrm{d}y - A_1 \right) - \left(\int_\Omega (1 - H_\varepsilon(\phi)) - A_2 \right) \right] \Bigg\} = 0\end{aligned}$$

with $\delta_\varepsilon(\phi) = H'_\varepsilon(\phi)$.

Minimising for Γ: Discretisation on Ω

We simplify this to

$$\delta_\varepsilon(\phi)\Big\{\mu\nabla\cdot(G\nabla\phi) - \left[\lambda_1(z(x,y)-c_1)^2 - \lambda_2(z(x,y)-c_2)^2\right] - \nu\left[\left(\int_\Omega H_\varepsilon(\phi)\mathrm{d}x\mathrm{d}y - A_1\right) - \left(\int_\Omega(1-H_\varepsilon(\phi)) - A_2\right)\right]\Big\} = 0$$

Discretising on Ω and using finite differences and we obtain

$$\delta_\varepsilon(\phi_{i,j})\Big\{\mu\left[\frac{G_{i+\frac{1}{2},j}}{h_x^2}(\phi_{i+1,j}-\phi_{i,j}) - \frac{G_{i-\frac{1}{2},j}}{h_x^2}(\phi_{i,j}-\phi_{i-1,j}) + \frac{G_{i,j+\frac{1}{2}}}{h_y^2}(\phi_{i,j+1}-\phi_{i,j}) - \frac{G_{i,j-\frac{1}{2}}}{h_y^2}(\phi_{i,j}-\phi_{i,j-1})\right] - \left[\lambda_1(z_{i,j}-c_1)^2 - \lambda_2(z_{i,j}-c_2)^2\right] - \nu\left[\left(h_xh_y\sum_{k,l}H_\varepsilon(\phi_{k,l}) - A_1\right) - \left(h_xh_y\sum_{k,l}(1-H_\varepsilon(\phi_{k,l})) - A_2\right)\right]\Big\} = 0$$

where $G = \frac{d(x,y)\cdot g(|\nabla z(x,y)|)}{|\nabla\phi|}$

Minimising for Γ: Numerical scheme

We simplify to

$$A_{i,j}\phi_{i+1,j} + B_{i,j}\phi_{i-1,j} + C_{i,j}\phi_{i,j+1} + D_{i,j}\phi_{i,j-1} - S_{i,j}\phi_{i,j}$$

$$-\delta_\varepsilon(\phi_{i,j})\Big[\lambda_1(z_{i,j}-c_1)^2 - \lambda_2(z_{i,j}-c_2)^2\Big]$$

$$-\nu\delta_\varepsilon(\phi_{i,j})\Big[\Big(h_x h_y \sum_{k,l} H_\varepsilon(\phi_{k,l}) - A_1\Big) - \Big(h_x h_y \sum_{k,l}(1 - H_\varepsilon(\phi_{k,l})) - A_2\Big)\Big] = 0$$

where

$$A_{i,j} = \frac{\mu\delta_\varepsilon(\phi_{i,j})}{h_x^2} G_{i+\frac{1}{2},j} \qquad B_{i,j} = \frac{\mu\delta_\varepsilon(\phi_{i,j})}{h_x^2} G_{i-\frac{1}{2},j} \qquad C_{i,j} = \frac{\mu\delta_\varepsilon(\phi_{i,j})}{h_y^2} G_{i,j+\frac{1}{2}}$$

$$D_{i,j} = \frac{\mu\delta_\varepsilon(\phi_{i,j})}{h_y^2} G_{i,j-\frac{1}{2}} \qquad S_{i,j} = A_{i,j} + B_{i,j} + C_{i,j} + D_{i,j}$$

Minimising for Γ: Numerical scheme

We iterate the numerical scheme on all pixels

$$A_{i,j}\phi_{i+1,j} + B_{i,j}\phi_{i-1,j} + C_{i,j}\phi_{i,j+1} + D_{i,j}\phi_{i,j-1} - S_{i,j}\phi_{i,j}$$

$$-\delta_\varepsilon(\phi_{i,j})\Big[\lambda_1(z_{i,j} - c_1)^2 - \lambda_2(z_{i,j} - c_2)^2\Big]$$

$$-\nu\delta_\varepsilon(\phi_{i,j})\Big[\Big(h_x h_y \sum_{k,l} H_\varepsilon(\phi_{k,l}) - A_1\Big) - \Big(h_x h_y \sum_{k,l}(1 - H_\varepsilon(\phi_{k,l})) - A_2\Big)\Big] = 0$$

until convergence to a solution ϕ.

The level set $\phi = 0$ is the boundary of the object.

We can solve this pixel-by-pixel for $\phi_{i,j}$ or for entire lines of pixels $\phi_{\cdot,j}$.

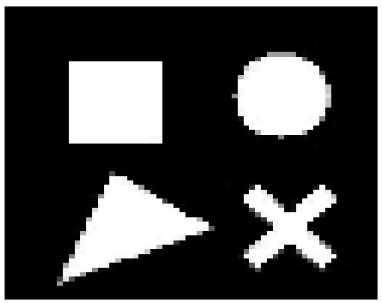
Problem: Expensive for large images.

Introduction to Multigrid

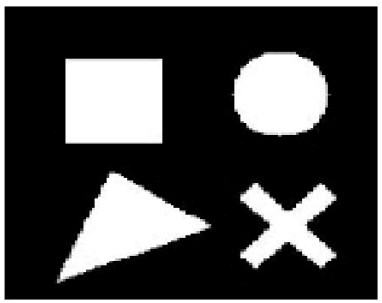
Introduction to Multigrid: Motivation

Key ideas:

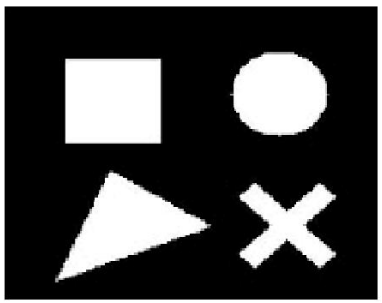
- If we solve our iterative scheme on fewer pixels it is computationally less expensive.
- Interpolate the solution.



64×64



128×128



256×256

Motivation - fast

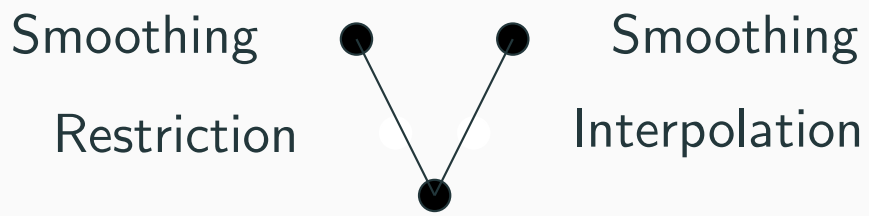
Image size	Semi-Implicit Method		Additive Operator Splitting		Multigrid	
	Iterations	Cpu-time(s)	Iterations	Cpu-time(s)	Iterations	Cpu-time(s)
128×128	30	29	35	14	2	9
256×256	85	511	80	138	3	20
512×512	-	-	5000	3.6×10^4	3	41
1024×1024	-	-	-	-	3	165

Table from Badshah and Chen 2008

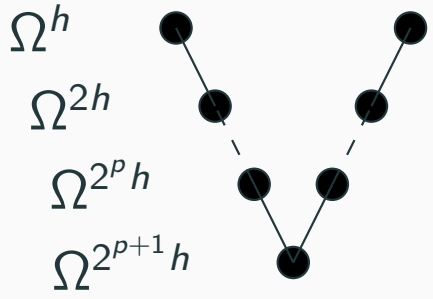
Introduction to Multigrid: Framework outline

Key elements:

- Restriction
- Interpolation
- Coarse grid solution
- Smoothing



Exact solution on Ω^{2h}



Key requirement:

- To move between different grid levels we need smooth errors, $e_{i,j}^{(k)} = \phi_{i,j} - \phi_{i,j}^{(k)}$.

Introduction to Multigrid: Smoother comparisons

Using Local Fourier Analysis (LFA) on each pixel we measure a smoothing rate $\mu_{i,j}$ where

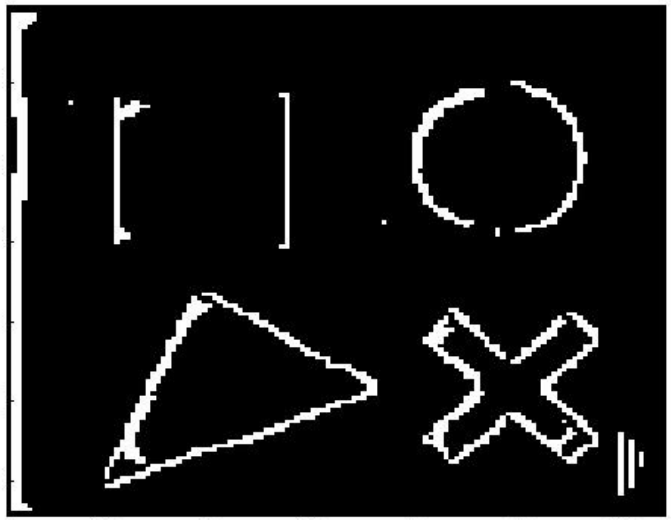
$$e_{i,j}^{(k+1)} = \mu_{i,j} e_{i,j}^{(k)}$$

- Gauss-Seidel and Newton iterative schemes smooth errors effectively for linear problems.
- They perform poorly for some pixels in non-linear problems.

Smoother	μ_{max}	μ_{ave}	Cpu-time (s)
GS Pixel - Global	0.9851	0.5073	135.6875
GS Pixel - Local	0.9999	0.6568	303.5000
GS Line - Global	0.9966	0.4499	73.8594
GS Line - Local	0.9966	0.4499	74.4844
Newton Pixel - Global	0.9847	0.5089	141.1719
Newton Pixel - Local	0.9999	0.6688	316.6719

Introduction to Multigrid: Problems with the Smoother

We find that the smoothing rate is poor mainly at edges.



We focus on these edge pixels in particular and use a different smoother at these pixels.

Introduction to Multigrid: Locating where smoothing is poor

Top 5 worst smoothing rates

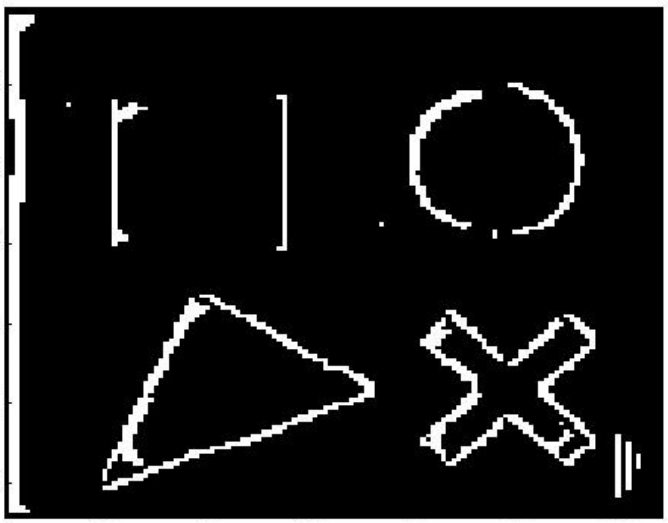
i	j	$\hat{\mu}_{i,j}$	$A_{i,j}$	$B_{i,j}$	$C_{i,j}$	$D_{i,j}$
52	55	0.9966	0.0027	0.0027	0.2125	0.0007
54	55	0.9966	0.0027	0.0027	0.2116	0.0007
49	55	0.9966	0.0027	0.0027	0.2132	0.0007
55	55	0.9966	0.0027	0.0027	0.2107	0.0007
46	55	0.9966	0.0027	0.0027	0.2134	0.0007

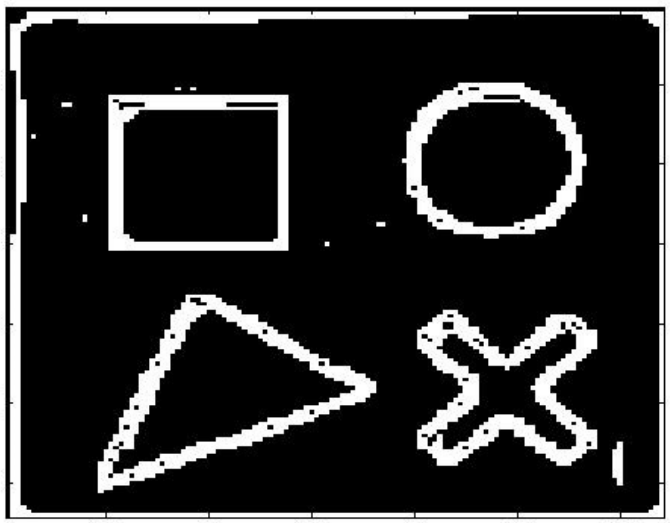
Suggests that the problem is due to significant variances in the sizes of $A_{i,j}, B_{i,j}, C_{i,j}$ and $D_{i,j}$ in our scheme

$$A_{i,j}\phi_{i+1,j} + B_{i,j}\phi_{i-1,j} + C_{i,j}\phi_{i,j+1} + D_{i,j}\phi_{i,j-1} - S_{i,j}\phi_{i,j} - K(\phi_{i,j}) = 0$$

Introduction to Multigrid: Locating where smoothing is poor

Smoothing rate over 0.5 vs. pixels where one of $A_{i,j}$, $B_{i,j}$, $C_{i,j}$ and $D_{i,j}$ differs by a factor of at least 50%





New idea: we look at those pixels where at least one of $A_{i,j}$, $B_{i,j}$, $C_{i,j}$ and $D_{i,j}$ is significantly different to the others.

Introduction to Multigrid: Considering cases

There are 14 cases to consider for the relative sizes of $A_{i,j}$, $B_{i,j}$, $C_{i,j}$ and $D_{i,j}$ when at least one is significantly different.

Case #	$A_{i,j}$	$B_{i,j}$	$C_{i,j}$	$D_{i,j}$
1	S	L	L	S
2	S	L	S	L
3	L	S	L	S
4	L	S	S	L
5	L	L	S	S
6	S	S	L	L
7	L	S	S	S

Case #	$A_{i,j}$	$B_{i,j}$	$C_{i,j}$	$D_{i,j}$
8	S	S	L	S
9	S	L	S	S
10	S	S	S	L
11	L	L	S	L
12	L	S	L	L
13	L	L	L	S
14	S	L	L	L

Let $\mathcal{D}$ be the set of pixels which are one of these cases.

Smoother	$\mu_{\max \mathcal{D}}$	$\mu_{\text{ave } \mathcal{D}}$	$\mu_{\max \Omega \setminus \mathcal{D}}$	$\mu_{\text{ave } \Omega \setminus \mathcal{D}}$
GS Line - Global	0.9966	0.5039	0.6372	0.4401

Away from $\mathcal{D}$ the rates aren't too bad.

Introduction to Multigrid: New smoother

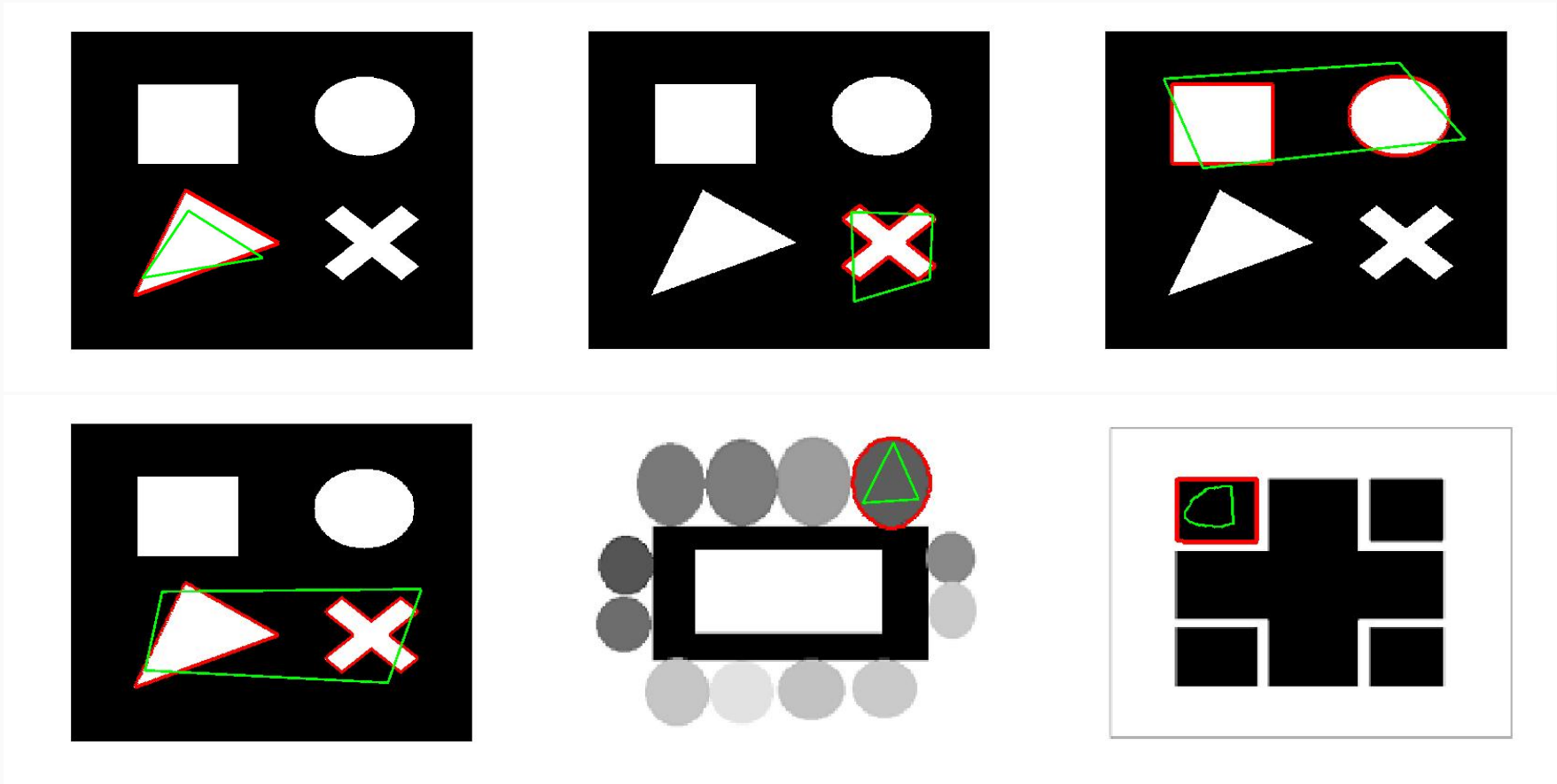
- We develop different schemes for each case with improved smoothing rate.

Smoother	$\mu_{\max\ before}$	$\mu_{\max\ after}$
GS Line - Global	0.9966	0.8774

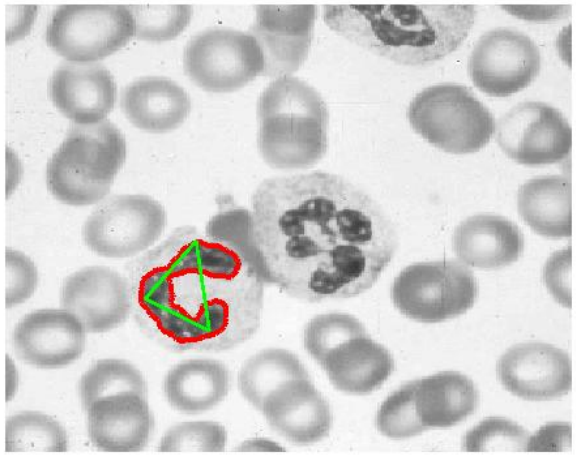
- Although the decrease is small, it is significant.
- To smooth out the errors by 90% we would have needed 677 iterations but now need only 18.
- We use the line smoother on **all** pixels then use iterative schemes on the pixels of $\mathcal{D}$.

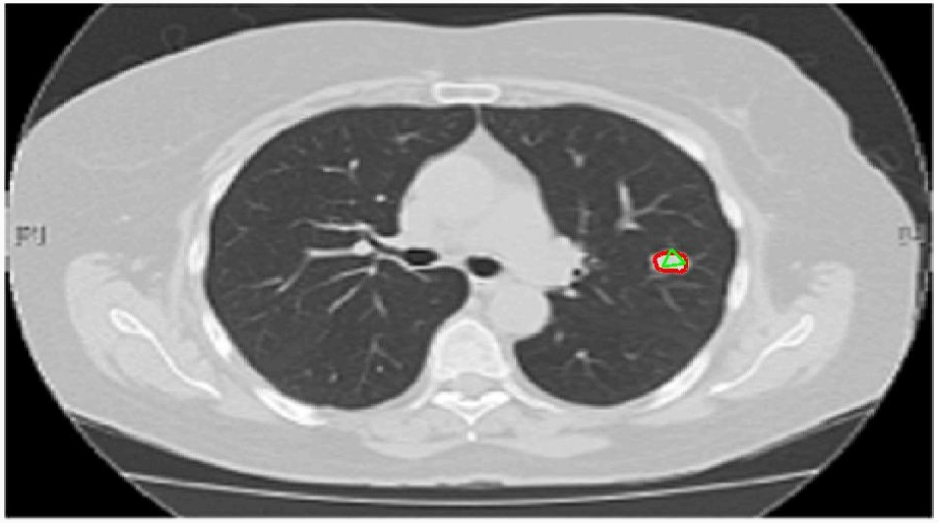
Preliminary Results

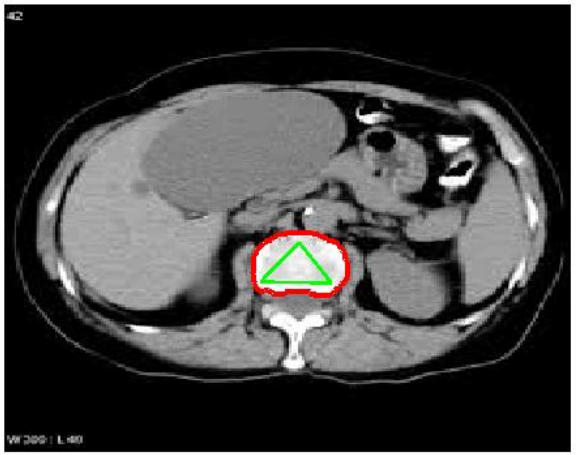
Preliminary Results: Synthetic images



Preliminary Results: Real images







End

Appendix A: Scheme derivation

Our PDE is

$$\mu\nabla\cdot(G\nabla\phi) - \left(\lambda_1(z(x,y)-c_1)^2 - \lambda_2(z(x,y)-c_2)^2\right)$$
$$-\nu\left[\left(\int_\Omega H(\phi)\mathrm{d}x\mathrm{d}y - A_1\right) - \left(\int_\Omega (1-H(\phi)) - A_2\right)\right] = 0$$

Using finite differences and discretising on Ω

$$\Delta_x(\phi_{i,j}) = \frac{\phi_{i+\frac{1}{2},j} - \phi_{i-\frac{1}{2},j}}{h_x} \qquad \Delta_y(\phi_{i,j}) = \frac{\phi_{i,j+\frac{1}{2}} - \phi_{i,j-\frac{1}{2}}}{h_y}$$

we can rewrite this as

$$\mu\left[\Delta_x(G_{i,j}\cdot\Delta_x(\phi_{i,j})) + \Delta_y(G_{i,j}\cdot\Delta_y(\phi_{i,j}))\right] - \left[\lambda_1(z_{i,j}-c_1)^2 - \lambda_2(z_{i,j}-c_2)^2\right]$$
$$-\nu\left[\left(\sum_{k,l} H_\varepsilon(\phi_{k,l}) - A_1\right) - \left(\sum_{k,l}(1-H_\varepsilon(\phi_{k,l})) - A_2\right)\right] = 0$$

Appendix A: Scheme derivation

$$\mu\left[\Delta_x(G_{i,j}\cdot\Delta_x(\phi_{i,j}))+\Delta_y(G_{i,j}\cdot\Delta_y(\phi_{i,j}))\right]-\left[\lambda_1(z_{i,j}-c_1)^2-\lambda_2(z_{i,j}-c_2)^2\right]$$
$$-\nu\left[(\sum_{k,l}H_\varepsilon(\phi_{k,l})-A_1)-(\sum_{k,l}(1-H_\varepsilon(\phi_{k,l}))-A_2)\right]=0$$

noting that $\sum_{k,l}1=n\cdot m$, $A_2=n\cdot m-A_1$ and using the finite difference operators we obtain

$$\mu\left[\frac{(G_{i+\frac{1}{2},j}\cdot\Delta_x(\phi_{i+\frac{1}{2},j}))}{h_x}-\frac{(G_{i-\frac{1}{2},j}\cdot\Delta_x(\phi_{i-\frac{1}{2},j}))}{h_x}+\frac{(G_{i,j+\frac{1}{2}}\cdot\Delta_y(\phi_{i,j+\frac{1}{2}}))}{h_y}-\right.$$
$$\left.\frac{(G_{i,j-\frac{1}{2}}\cdot\Delta_y(\phi_{i,j-\frac{1}{2}}))}{h_y}\right]-\left[\lambda_1(z_{i,j}-c_1)^2-\lambda_2(z_{i,j}-c_2)^2\right]$$
$$-\nu\left[(\sum_{k,l}H_\varepsilon(\phi_{k,l})-A_1)-(n\cdot m-\sum_{k,l}H_\varepsilon(\phi_{k,l})-(n\cdot m-A_1))\right]=0$$

Appendix A: Scheme derivation

$$\mu \left[\frac{(G_{i+\frac{1}{2},j} \cdot \Delta_x(\phi_{i+\frac{1}{2},j}))}{h_x} - \frac{(G_{i-\frac{1}{2},j} \cdot \Delta_x(\phi_{i-\frac{1}{2},j}))}{h_x} + \frac{(G_{i,j+\frac{1}{2}} \cdot \Delta_y(\phi_{i,j+\frac{1}{2}}))}{h_y} - \frac{(G_{i,j-\frac{1}{2}} \cdot \Delta_y(\phi_{i,j-\frac{1}{2}}))}{h_y} \right] - \left[\lambda_1 (z_{i,j} - c_1)^2 - \lambda_2 (z_{i,j} - c_2)^2 \right]$$

$$- \nu \left[(\sum_{k,l} H_\varepsilon(\phi_{k,l}) - A_1) - (n \cdot m - \sum_{k,l} H_\varepsilon(\phi_{k,l}) - (n \cdot m - A_1)) \right] = 0$$

using the finite difference operators again and simplifying the final term we obtain

$$\mu \left[\frac{G_{i+\frac{1}{2},j}}{h_x^2} (\phi_{i+1,j} - \phi_{i,j}) - \frac{G_{i-\frac{1}{2},j}}{h_x^2} (\phi_{i,j} - \phi_{i-1,j}) + \frac{G_{i,j+\frac{1}{2}}}{h_y^2} (\phi_{i,j+1} - \phi_{i,j}) - \frac{G_{i,j-\frac{1}{2}}}{h_y^2} (\phi_{i,j} - \phi_{i,j-1}) \right] - \left[\lambda_1 (z_{i,j} - c_1)^2 - \lambda_2 (z_{i,j} - c_2)^2 \right] - 2\nu \left[\sum_{k,l} H_\varepsilon(\phi_{k,l}) - A_1 \right]$$

$$= 0$$

Appendix B: Line smoother derivation

We start with

$$A_{i,j}\phi_{i+1,j} + B_{i,j}\phi_{i-1,j} + C_{i,j}\phi_{i,j+1} + D_{i,j}\phi_{i,j-1} - S_{i,j}\phi_{i,j}$$
$$- \left[\lambda_1(z_{i,j} - c_1)^2 - \lambda_2(z_{i,j} - c_2)^2\right] - 2\nu\left[\sum_{k,l} H_\varepsilon(\phi_{k,l}) - A_1\right] = 0$$

Grouping terms of the form $\phi_{\cdot,j}$ on the LHS we obtain

$$A_{i,j}\phi_{i+1,j} + B_{i,j}\phi_{i-1,j} - S_{i,j}\phi_{i,j}$$
$$= -C_{i,j}\phi_{i,j+1} - D_{i,j}\phi_{i,j-1} + \left[\lambda_1(z_{i,j} - c_1)^2 - \lambda_2(z_{i,j} - c_2)^2\right]$$
$$+ 2\nu\left[\sum_{k,l} H_\varepsilon(\phi_{k,l}) - A_1\right]$$

Recall that $A_{i,j} = \frac{\mu}{h_x^2} G_{i+\frac{1}{2},j}$ and $B_{i,j} = \frac{\mu}{h_x^2} G_{i-\frac{1}{2},j}$ and hence $A_{i-1,j} = B_{i,j}$.

Appendix B: Line smoother derivation

$$
\begin{aligned}
&A_{i,j}\phi_{i+1,j} + A_{i-1,j}\phi_{i-1,j} - S_{i,j}\phi_{i,j} \\
&\quad = -C_{i,j}\phi_{i,j+1} - D_{i,j}\phi_{i,j-1} + \left[\lambda_1(z_{i,j} - c_1)^2 - \lambda_2(z_{i,j} - c_2)^2\right] \\
&\qquad + 2\nu\left[\sum_{k,l} H_\varepsilon(\phi_{k,l}) - A_1\right] = F_{i,j}
\end{aligned}
$$

we now form a system

$$
\begin{pmatrix}
-S_{1,j} & A_{1,j} & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\
A_{1,j} & -S_{2,j} & A_{2,j} & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\
0 & A_{2,j} & -S_{3,j} & A_{3,j} & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & & & & \vdots \\
\vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & & & \vdots \\
\vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & & & \vdots \\
\vdots & & & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\
\vdots & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & 0 & 0 & \dots & 0 & A_{n-3,j} & -S_{n-2,j} & A_{n-2,j} & 0 \\
0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & A_{n-2,j} & -S_{n-1,j} & A_{n-1,j} \\
0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & A_{n-1,j} & -S_{n,j}
\end{pmatrix}
\cdot
\begin{pmatrix}
\phi_{1,j} \\ \phi_{2,j} \\ \phi_{3,j} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \phi_{n-2,j} \\ \phi_{n-1,j} \\ \phi_{n,j}
\end{pmatrix}
=
\begin{pmatrix}
F_{1,j} \\ F_{2,j} \\ F_{3,j} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ F_{n-2,j} \\ F_{n-1,j} \\ F_{n,j}
\end{pmatrix}
$$

Appendix B: Line smoother derivation

$$\begin{pmatrix} -S_{1,j} & A_{1,j} & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ A_{1,j} & -S_{2,j} & A_{2,j} & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & A_{2,j} & -S_{3,j} & A_{3,j} & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & A_{n-3,j} & -S_{n-2,j} & A_{n-2,j} & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & A_{n-2,j} & -S_{n-1,j} & A_{n-1,j} \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & A_{n-1,j} & -S_{n,j} \end{pmatrix} \cdot \begin{pmatrix} \phi_{1,j} \\ \phi_{2,j} \\ \phi_{3,j} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \phi_{n-2,j} \\ \phi_{n-1,j} \\ \phi_{n,j} \end{pmatrix} = \begin{pmatrix} F_{1,j} \\ F_{2,j} \\ F_{3,j} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ F_{n-2,j} \\ F_{n-1,j} \\ F_{n,j} \end{pmatrix}$$

We solve this for the entire column of index j in one iteration.

Note that this is a (strictly if enforced) diagonally dominant tridiagonal symmetric matrix.

Therefore a Gauss-Seidel method will converge to a solution.

Appendix C: Adaptive iterative scheme derivation

For each of the six cases we derive the schemes in the same way. We show case 1 in detail.

In this case we have $A_{i,j}$ and $D_{i,j}$ relatively small and $B_{i,j}$ and $C_{i,j}$ relatively large. We can therefore rewrite our scheme as

$$B_{i,j}\phi_{i-1,j} + C_{i,j}\phi_{i,j+1} - S_{i,j}\phi_{i,j} = f_{i,j} - A_{i,j}\phi_{i+1,j} - D_{i,j}\phi_{i,j-1}$$

by moving the small terms to the right hand side. We now look to solve $\phi_{i-1,j}, \phi_{i,j+1}$ and $\phi_{i,j}$ as a coupled system. We can rewrite this scheme, with the iteration number indicated, as

$$B_{i,j}\phi_{i-1,j}^{(k+1)} + C_{i,j}\phi_{i,j+1}^{(k+1)} - S_{i,j}\phi_{i,j}^{(k+1)} = f_{i,j} - A_{i,j}\phi_{i+1,j}^{(k)} - D_{i,j}\phi_{i,j-1}^{(k)}$$

Appendix C: Adaptive iterative scheme derivation

$$B_{i,j}\phi_{i-1,j}^{(k+1)} + C_{i,j}\phi_{i,j+1}^{(k+1)} - S_{i,j}\phi_{i,j}^{(k+1)} = f_{i,j} - A_{i,j}\phi_{i+1,j}^{(k)} - D_{i,j}\phi_{i,j-1}^{(k)}$$

We compare the smoothing rate of this scheme with GSLINE-I

Smoother	$\tilde{\mu}_{\max}$	$\tilde{\mu}_{\text{ave}}$
GSLINE-I	0.9427	0.7795
Adapted Scheme	0.4182	0.2022

Appendix C: Adaptive iterative scheme derivation

$$B_{i,j}\phi_{i-1,j}^{(k+1)} + C_{i,j}\phi_{i,j+1}^{(k+1)} - S_{i,j}\phi_{i,j}^{(k+1)} = f_{i,j} - A_{i,j}\phi_{i+1,j}^{(k)} - D_{i,j}\phi_{i,j-1}^{(k)}$$

We are solving for $\phi_{i-1,j}, \phi_{i,j+1}$ and $\phi_{i,j}$ simultaneously and as we have only one equation then we need to use two more equations in these variables to solve the system.

We get these by considering

$$B_{i,j}\phi_{i-1,j} + C_{i,j}\phi_{i,j+1} - S_{i,j}\phi_{i,j} = f_{i,j} - A_{i,j}\phi_{i+1,j} - D_{i,j}\phi_{i,j-1}$$

at the pixels $(i,j) \to (i,j+1)$ and $(i,j) \to (i-1,j)$ which gives us the two equations

$$D_{i,j+1}\phi_{i,j} - S_{i,j+1}\phi_{i,j+1} = f_{i,j+1} - A_{i,j+1}\phi_{i+1,j+1} - B_{i,j+1}\phi_{i-1,j+1} - C_{i,j+1}\phi_{i,j+2}$$

$$A_{i-1,j}\phi_{i,j} - S_{i-1,j}\phi_{i-1,j} = f_{i-1,j} - B_{i-1,j}\phi_{i-2,j} - C_{i-1,j}\phi_{i-1,j+1} - D_{i-1,j}\phi_{i-1,j-1}$$

Appendix C: Adaptive iterative scheme derivation

$$B_{i,j}\phi_{i-1,j} + C_{i,j}\phi_{i,j+1} - S_{i,j}\phi_{i,j} = f_{i,j} - A_{i,j}\phi_{i+1,j} - D_{i,j}\phi_{i,j-1}$$

$$D_{i,j+1}\phi_{i,j} - S_{i,j+1}\phi_{i,j+1} = f_{i,j+1} - A_{i,j+1}\phi_{i+1,j+1} - B_{i,j+1}\phi_{i-1,j+1} - C_{i,j+1}\phi_{i,j+2}$$

$$A_{i-1,j}\phi_{i,j} - S_{i-1,j}\phi_{i-1,j} = f_{i-1,j} - B_{i-1,j}\phi_{i-2,j} - C_{i-1,j}\phi_{i-1,j+1} - D_{i-1,j}\phi_{i-1,j-1}$$

Noting $D_{i,j+1} = C_{i,j}$ and $A_{i-1,j} = B_{ii,j}$ we form a system

$$\begin{pmatrix} -S_{i,j} & C_{i,j} & B_{i,j} \\ C_{i,j} & -S_{i,j+1} & 0 \\ B_{i,j} & 0 & -S_{i-1,j} \end{pmatrix} \cdot \begin{pmatrix} \phi_{i,j} \\ \phi_{i,j+1} \\ \phi_{i-1,j} \end{pmatrix} = \begin{pmatrix} f_{i,j} - A_{i,j}\phi_{i+1,j} - D_{i,j}\phi_{i,j-1} \\ f_{i,j+1} - A_{i,j+1}\phi_{i+1,j+1} - B_{i,j+1}\phi_{i-1,j+1} - C_{i,j+1}\phi_{i,j+2} \\ f_{i-1,j} - C_{i-1,j}\phi_{i-1,j+1} - D_{i-1,j}\phi_{i-1,j-1} - B_{i-1,j}\phi_{i-2,j} \end{pmatrix}$$

This is strictly diagonally dominant, tridiagonal and symmetric with a kiwer smoothing rate than GSLINE-I.

Appendix C: Adaptive iterative scheme derivation

For completeness the remaining schemes are given below.

Scheme for case 2

$$\begin{pmatrix} -S_{i,j} & A_{i,j} & C_{i,j} \\ A_{i,j} & -S_{i+1,j} & 0 \\ C_{i,j} & 0 & -S_{i,j+1} \end{pmatrix} \cdot \begin{pmatrix} \phi_{i,j} \\ \phi_{i+1,j} \\ \phi_{i,j+1} \end{pmatrix} = \begin{pmatrix} f_{i,j} - B_{i,j}\phi_{i-1,j} - D_{i,j}\phi_{i,j-1} \\ f_{i+1,j} - C_{i+1,j}\phi_{i+1,j+1} - D_{i+1,j}\phi_{i+1,j-1} - A_{i+1,j}\phi_{i+2,j} \\ f_{i,j+1} - A_{i,j+1}\phi_{i+1,j+1} - B_{i,j+1}\phi_{i-1,j+1} - C_{i,j+1}\phi_{i,j+2} \end{pmatrix}$$

Scheme for case 3

$$\begin{pmatrix} -S_{i,j} & A_{i,j} \\ A_{i,j} & -S_{i+1,j} \end{pmatrix} \cdot \begin{pmatrix} \phi_{i,j} \\ \phi_{i+1,j} \end{pmatrix} = \begin{pmatrix} f_{i,j} - B_{i,j}\phi_{i-1,j} - C_{i,j}\phi_{i,j+1} - D_{i,j}\phi_{i,j-1} \\ f_{i+1,j} - C_{i+1,j}\phi_{i+1,j+1} - D_{i+1,j}\phi_{i+1,j-1} - A_{i+1,j}\phi_{i+2,j} \end{pmatrix}$$

Scheme for case 4

$$\begin{pmatrix} -S_{i,j} & C_{i,j} \\ C_{i,j} & -S_{i,j+1} \end{pmatrix} \cdot \begin{pmatrix} \phi_{i,j} \\ \phi_{i,j+1} \end{pmatrix} = \begin{pmatrix} f_{i,j} - A_{i,j}\phi_{i+1,j} - B_{i,j}\phi_{i-1,j} - D_{i,j}\phi_{i,j-1} \\ f_{i,j+1} - A_{i,j+1}\phi_{i+1,j+1} - B_{i,j+1}\phi_{i-1,j+1} - C_{i,j+1}\phi_{i,j+2} \end{pmatrix}$$

Scheme for case 5

$$\begin{pmatrix} -S_{i,j} & B_{i,j} \\ B_{i,j} & -S_{i-1,j} \end{pmatrix} \cdot \begin{pmatrix} \phi_{i,j} \\ \phi_{i-1,j} \end{pmatrix} = \begin{pmatrix} f_{i,j} - A_{i,j}\phi_{i+1,j} - C_{i,j}\phi_{i,j+1} - D_{i,j}\phi_{i,j-1} \\ f_{i-1,j} - C_{i-1,j}\phi_{i-1,j+1} - D_{i-1,j}\phi_{i-1,j-1} - B_{i-1,j}\phi_{i-2,j} \end{pmatrix}$$

Appendix C: Adaptive iterative scheme derivation

Scheme for case 6

$$\begin{pmatrix} -S_{i,j} & A_{i,j} & B_{i,j} & C_{i,j} \\ A_{i,j} & -S_{i+1,j} & 0 & 0 \\ B_{i,j} & 0 & -S_{i-1,j} & 0 \\ C_{i,j} & 0 & 0 & -S_{i,j+1} \end{pmatrix} \cdot \begin{pmatrix} \phi_{i,j} \\ \phi_{i+1,j} \\ \phi_{i-1,j} \\ \phi_{i,j+1} \end{pmatrix} = \begin{pmatrix} f_{i,j} - D_{i,j}\phi_{i,j-1} \\ f_{i+1,j} - C_{i+1,j}\phi_{i+1,j+1} - D_{i+1,j}\phi_{i+1,j-1} - A_{i+1,j}\phi_{i+2,j} \\ f_{i-1,j} - C_{i-1,j}\phi_{i-1,j+1} - D_{i-1,j}\phi_{i-1,j-1} - B_{i-1,j}\phi_{i-2,j} \\ f_{i,j+1} - A_{i,j+1}\phi_{i+1,j+1} - B_{i,j+1}\phi_{i-1,j+1} - C_{i,j+1}\phi_{i,j+2} \end{pmatrix}$$