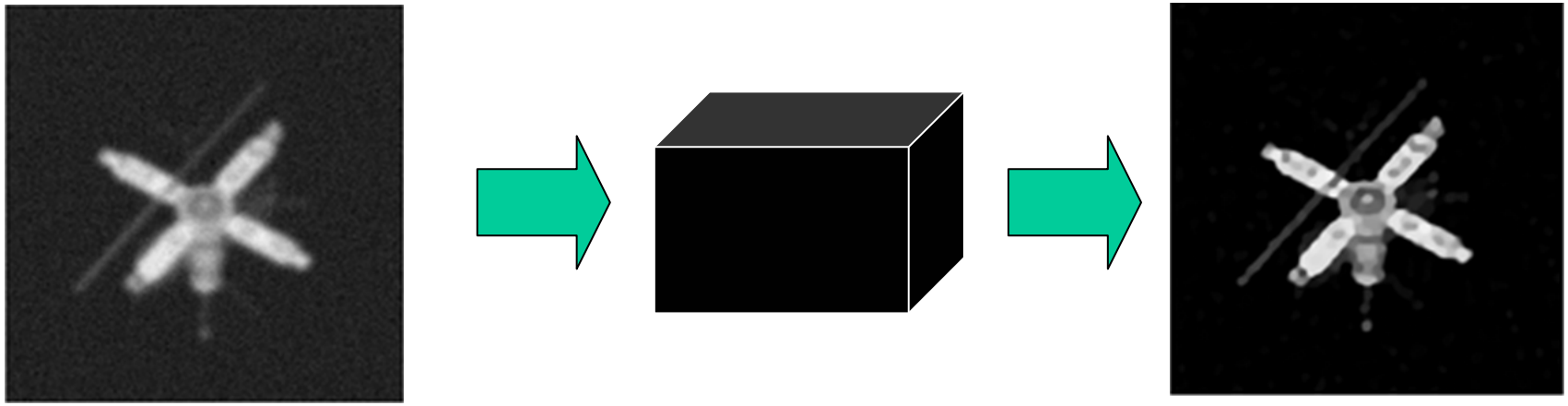
Optimization Methods for Total Variation Based Image Restoration

D.Goldfarb (Columbia University)
joint with W.Yin (Rice University)

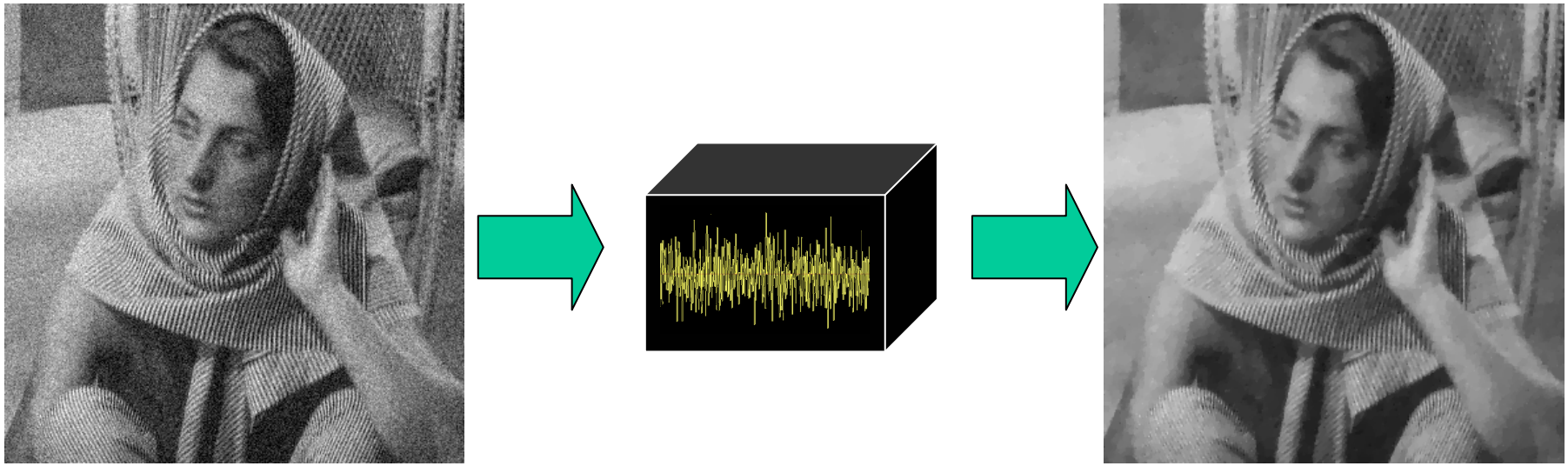
Workshop on Inverse Problem at Columbia University, 05-03-2007

Image Processing

filter black box



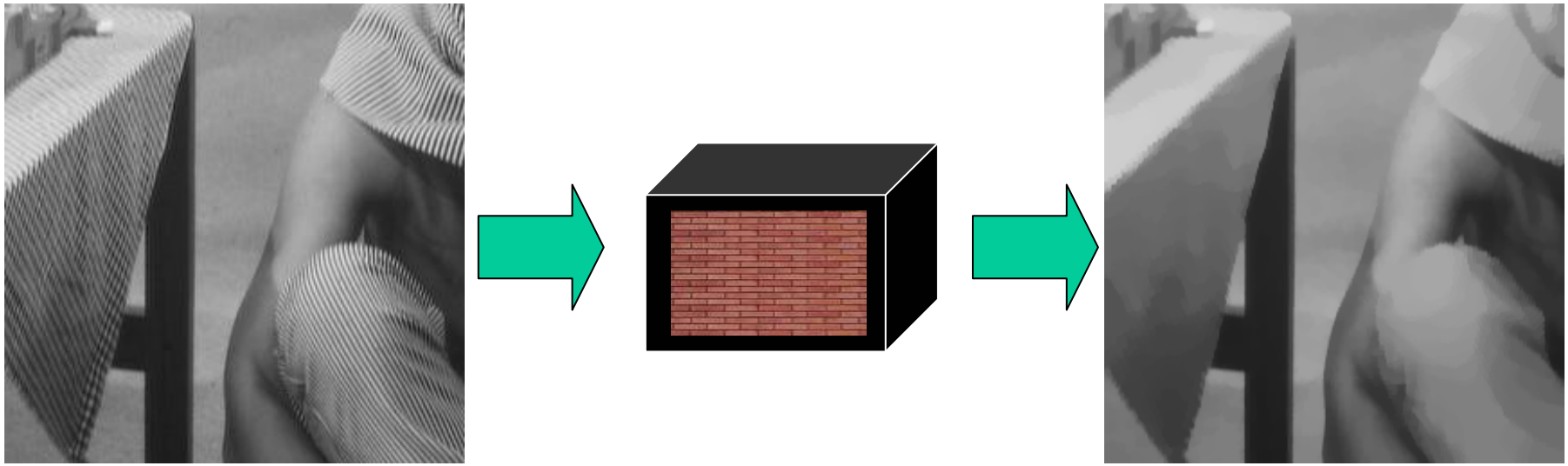
f u



f

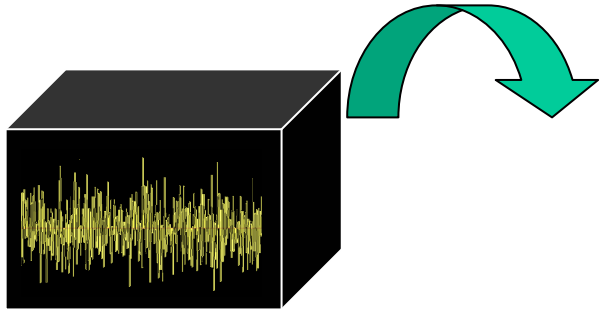
Noise removal filter

u

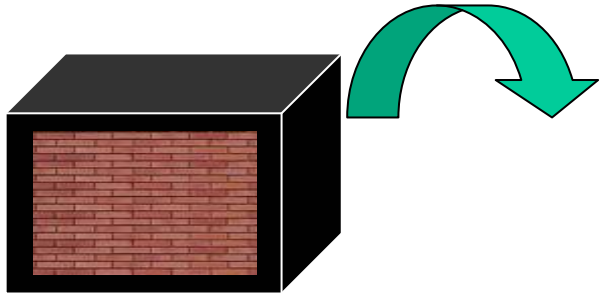


f u

One minimizes different functional to obtain *u* for these two cases



$$\min_u \; TV(u) + \lambda \|f - u\|_{L^2}^2 \text{ or}$$
$$\min_u \; TV(u), \text{ s.t. } \|f - u\|_{L^2} \le \sigma$$



$$\min_u \; TV(u) + \lambda \|f - u\|_{L^1} \text{ or}$$
$$\min_u \; TV(u), \text{ s.t. } \|f - u\|_{L^1} \le \sigma$$

$$TV(u) := \int_\Omega |\nabla u(x)| \mathrm{d}x$$

- Convex problems

ROF: Rudin-Osher-Fatemi, TV/L1: Alliney, Nikonova, Chan-Esedoglu, Yin-Goldfarb-Osher

Methods

PDE-based Gradient descent:

- low memory usage
- slow convergence

SOCP / interior-point method:

- high memory usage
- better convergence

Network flows methods:

- low memory usage
- very fast
- *not as general*

The PDE-based gradient descent approach

- Euler-Lagrange Eqns for the *unconstrained* ROF model:

$$0 = g(u) := \frac{1}{2\lambda}\nabla \cdot \frac{\nabla u}{|\nabla u|} + (f - u)$$

- Solve $\dfrac{\partial u}{\partial t} = g(u), \quad u(0) = f$ (Homogeneous Neumann boundary condition)

  to steady state

- must regularize $\|\nabla u_{ij}\| :\approx \sqrt{|\nabla_1 u_{ij}|^2 + |\nabla_2 u_{ij}|^2 + \varepsilon}$

where
$$\begin{aligned} \nabla_1 u_{i,j} &:= u_{i+1,j} - u_{i,j} \\ \nabla_2 u_{i,j} &:= u_{i,j+1} - u_{i,j} \end{aligned}$$

The Second-Order Programming (SOCP) approach

Discrete ROF:

$$\min \sum_{1 \le i,j \le n} t_{ij}$$

$$\text{s.t.}$$

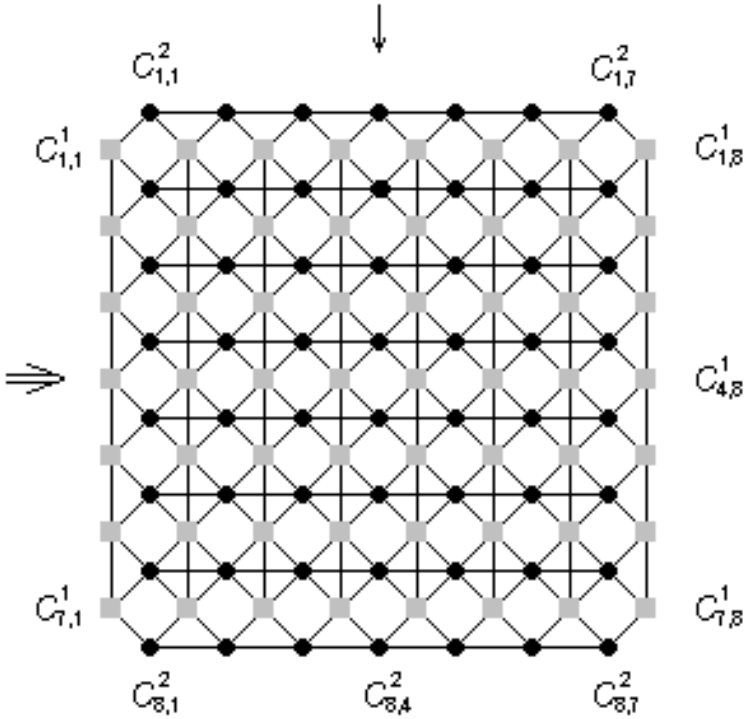
$$\text{(linear)} \quad u + v = f$$

$$\text{(SOCs)} \quad \|\nabla u_{i,j}\| \le t_{ij}$$

$$\text{(SOC)} \quad \|v\|_{l^2}^2 \le \sigma^2.$$

- Handles all *constrained* and *unconstrained* ROF and TV/L^1 models
- Does not require regularization
- Solved by interior-point methods
- Linear algebra is accelerated by applying *nested dissection*

SOCPs for TV-based models: Goldfarb-Yin 05'



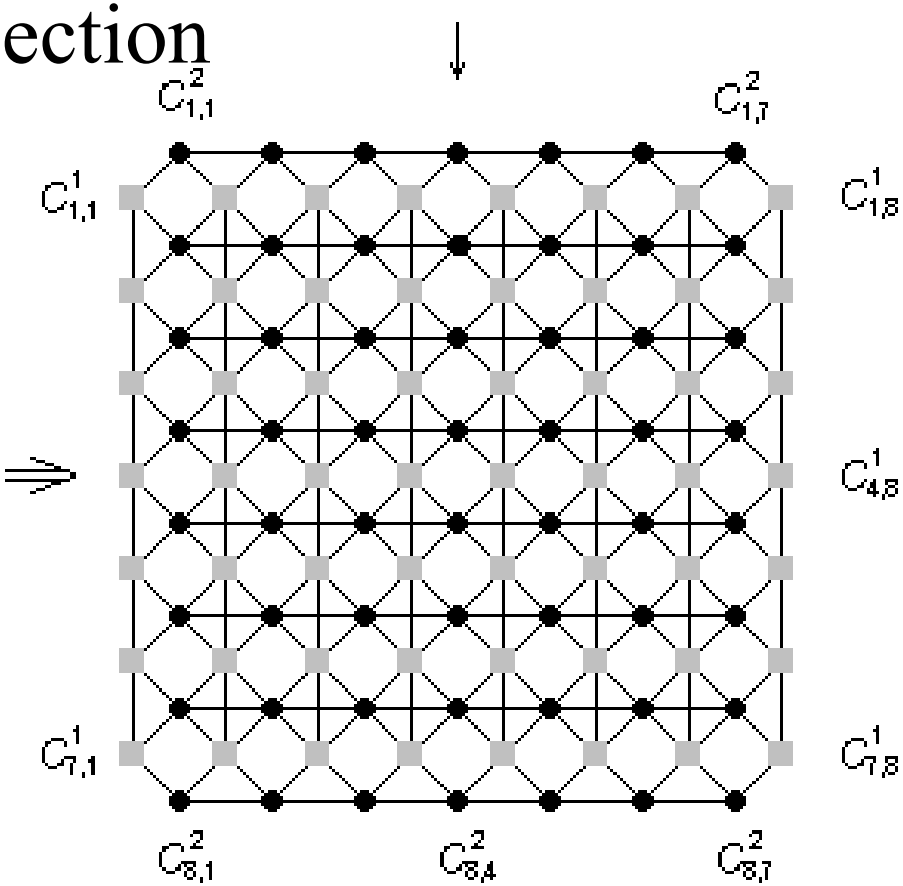
$$C^1: \quad (\partial_x^+ u)_{i,j} + (v_{i+1,j} - v_{i,j}) = f_{i+1,j} - f_{i,j}$$

$$C^2: \quad (\partial_y^+ u)_{i,j} + (v_{i,j+1} - v_{i,j}) = f_{i,j+1} - f_{i,j}$$

Cholesky with Nested Dissec.:	multiplications	$(943/84)n^3 + O(n^2 \log_2 n)$
	storage	$(31/4)n^2 \log_2 n + O(n^2)$

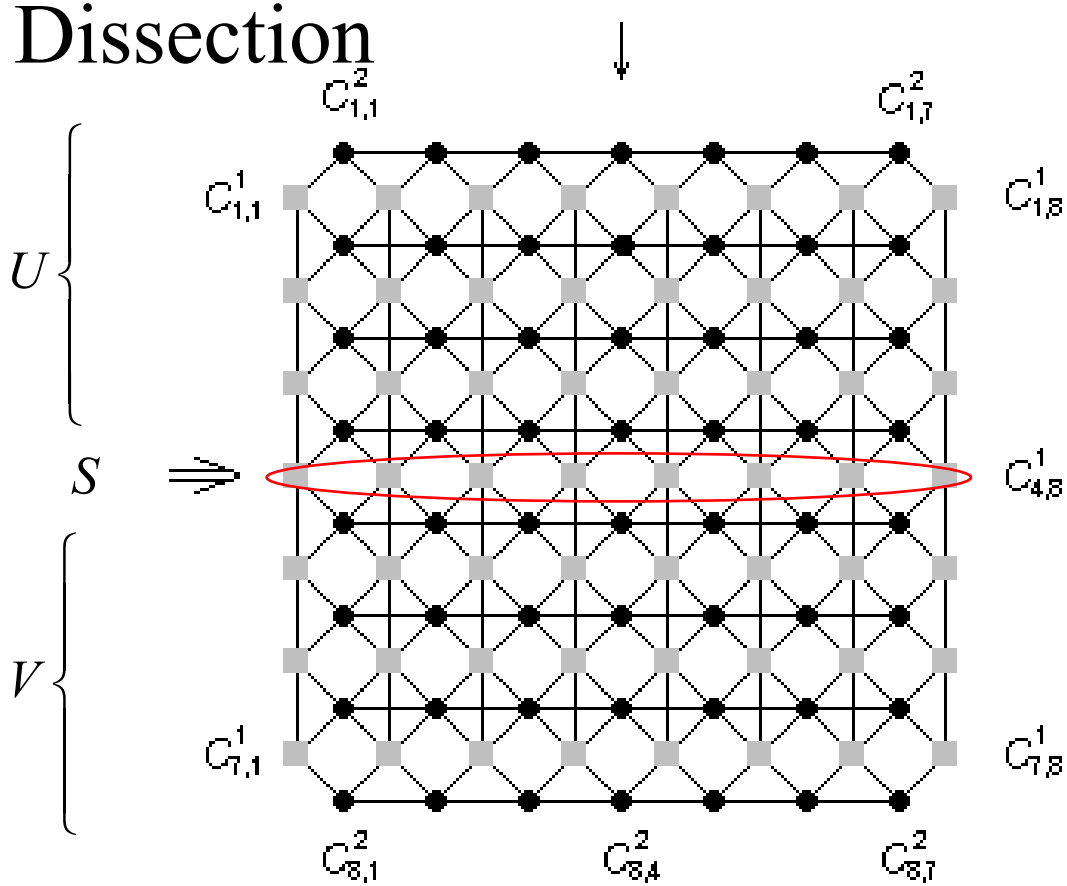
Nested Dissection: A.George 73'

Nested Dissection



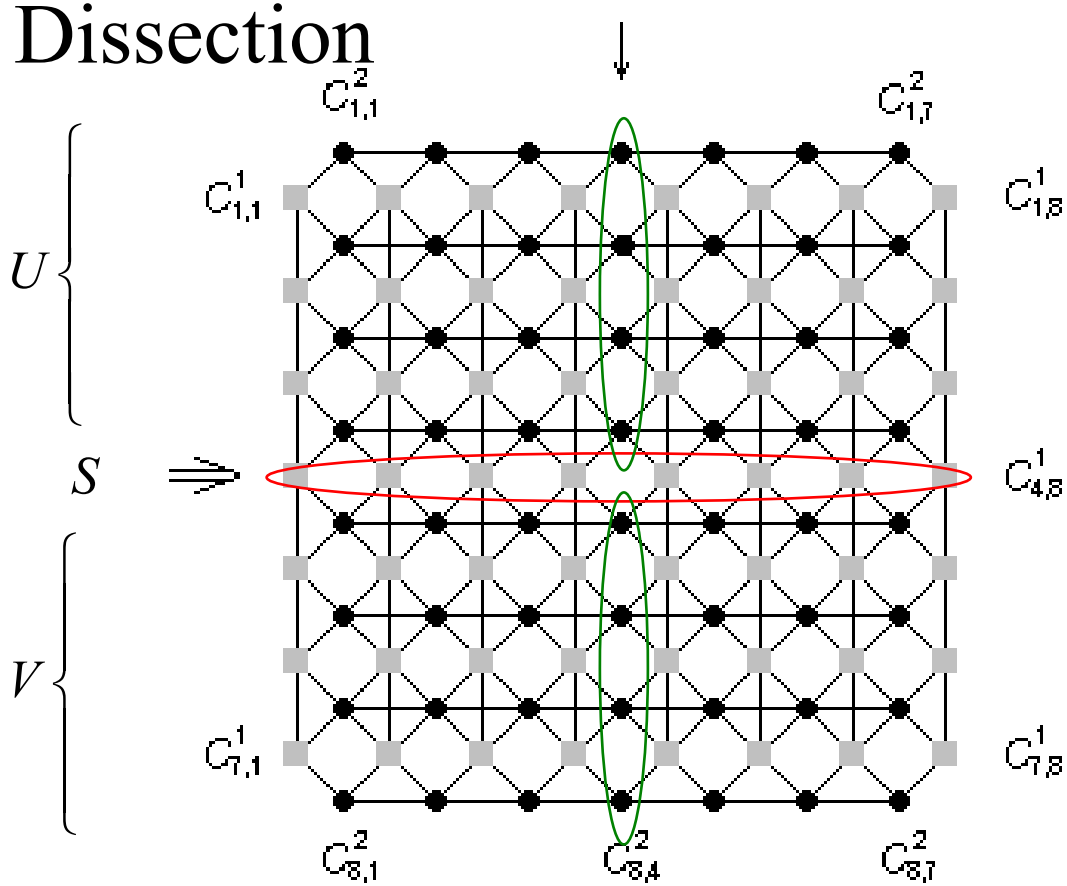
$$M = \begin{bmatrix} M_{UU} & 0 & M_{US} \\ 0 & M_{VV} & M_{VS} \\ M_{US}^{\mathrm{T}} & M_{VS}^{\mathrm{T}} & M_{SS} \end{bmatrix} \qquad L = \begin{bmatrix} L_{UU} & 0 & 0 \\ 0 & L_{VV} & 0 \\ L_{SU} & L_{SV} & L_{SS} \end{bmatrix}$$

Nested Dissection



$$M = \begin{bmatrix} M_{UU} & 0 & M_{US} \\ 0 & M_{VV} & M_{VS} \\ M_{US}^{\mathrm{T}} & M_{VS}^{\mathrm{T}} & M_{SS} \end{bmatrix} \qquad L = \begin{bmatrix} L_{UU} & 0 & 0 \\ 0 & L_{VV} & 0 \\ L_{SU} & L_{SV} & L_{SS} \end{bmatrix}$$

Nested Dissection



$$M = \begin{bmatrix} M_{UU} & 0 & M_{US} \\ 0 & M_{VV} & M_{VS} \\ M_{US}^{\mathrm{T}} & M_{VS}^{\mathrm{T}} & M_{SS} \end{bmatrix} \qquad L = \begin{bmatrix} L_{UU} & 0 & 0 \\ 0 & L_{VV} & 0 \\ L_{SU} & L_{SV} & L_{SS} \end{bmatrix}$$

SOCP(Matlab+Mosek) numerical results on tests using unconstrained ROF

Name	Size	best λ	total time*	# of itr.
books	64×64	0.03747189	4.42	15
dotplus	100×100	0.01805139	13.68	16
scale	252×252	0.03370882	157.86	17
barbara	256×256	0.03746579	130.13	16
barbara	350×350	0.03749693	283.56	16

Solving TV/L1 by SOCP is a few times slower

* seconds on a SUN E450 with 350Mhz CPUs and 4GB memory

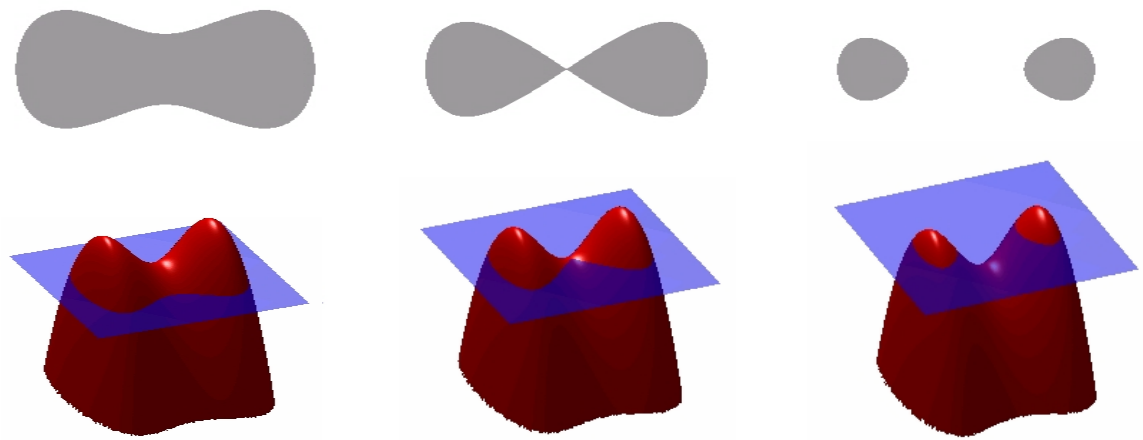
Max flow approach outline:

applicable to anisotropic TV(u) – i.e., l_1 norm

1. Decompose f into level sets $F_l = \{x \mid f(x) \geq l\}$

F_l

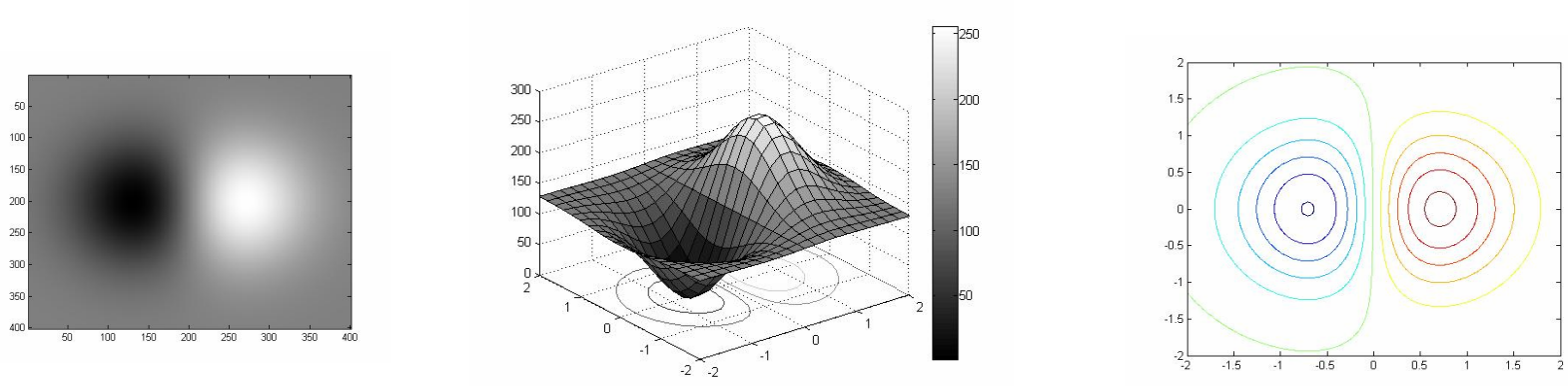
f, l



Max flow approach outline:

applicable to anisotropic TV(u) – i.e., l_1 norm

1. Decompose f into level sets $F_l = \{x \mid f(x) \geq l\}$



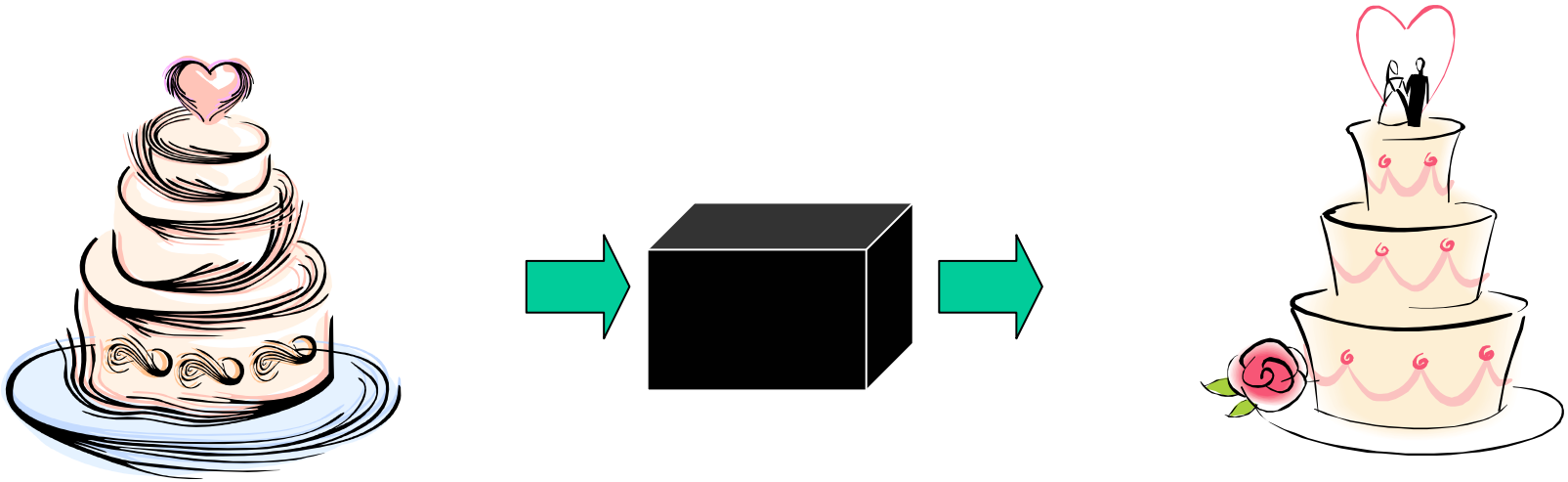
2. For each F_l, obtain U_l by solving a max-flow prob
   (solving a **binary** $\min_U \; TV(U) + \lambda \|F - U\|_{L^1}$)
3. Construct a minimizer u from the minimizers U_l

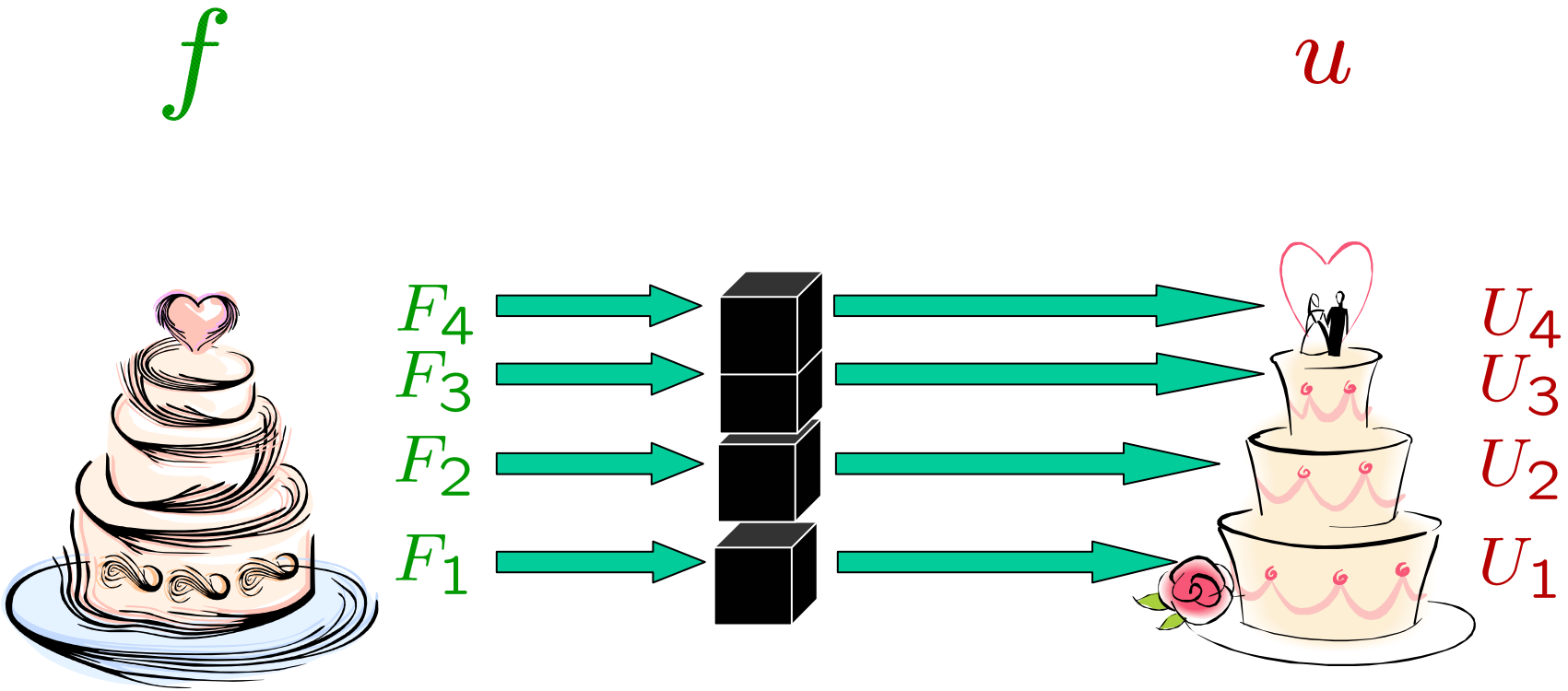
$$TV(u) + \lambda \|f - u\|_{L^1} = \int_{levels} \underbrace{\left[TV(U_l) + \lambda \|F_l - U_l\|_{L^1}\right]}_{\text{binary}} \mathrm{d}l$$

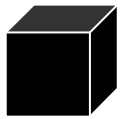
Layer-cake formula: Chan-Esedoglu 05'

f

u





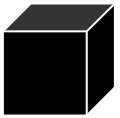
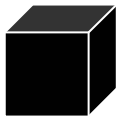
Requires monotonicity of 

Is it good to break the problem up into levels?

-  = finding a minimum cut of a capacitated network (Why? Answer coming next……)

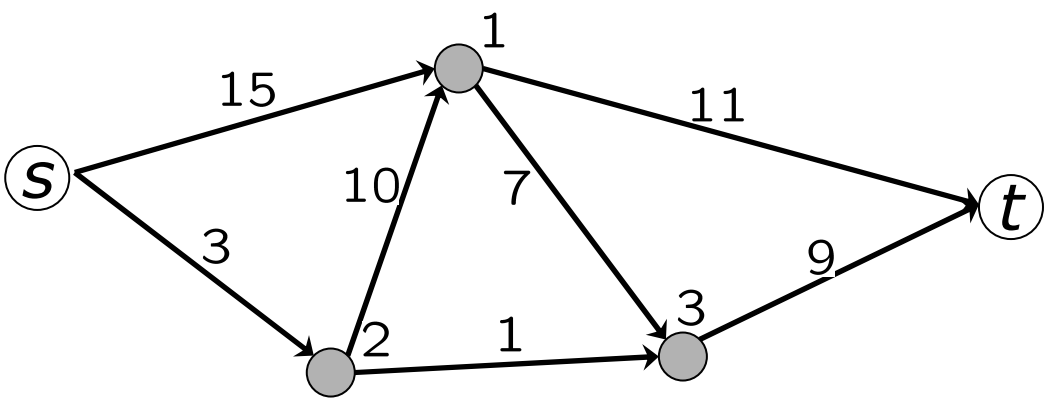
- For a 8-bit image, there are 2^8=256 levels

- For a 16-bit image, there are 2^{16}=65536 levels

- Answer depends on
  1. how fast we can solve each 
  2. how many  we need to solve

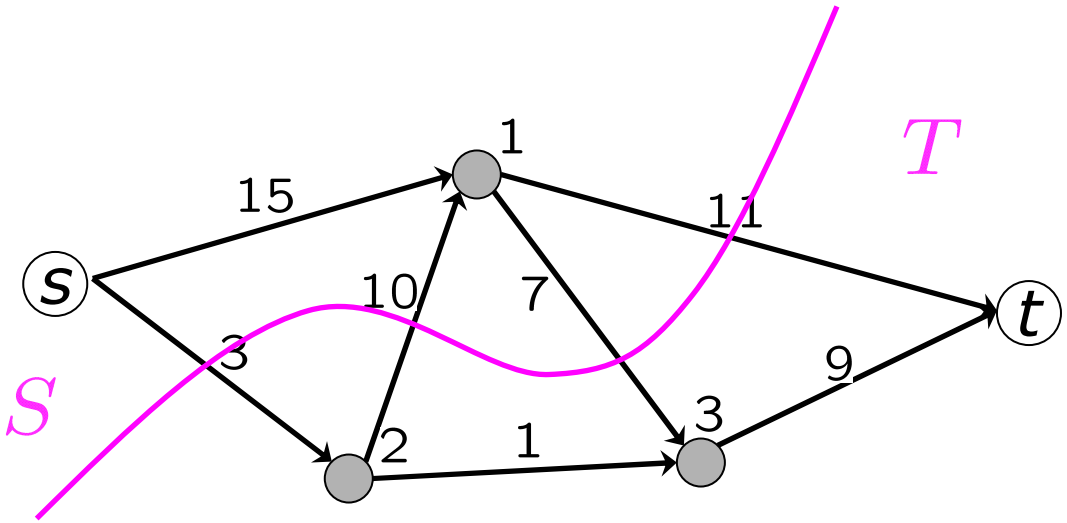
A capacitated network

- A Network is a graph *G* with *nodes* and *edges*: $G = (V, E)$
- Special nodes *s* (source) and *t* (sink)
- Edges carry flow
- Each edge (*i*,*j*) has a maximum capacity $c_{i,j}$



A capacitated network

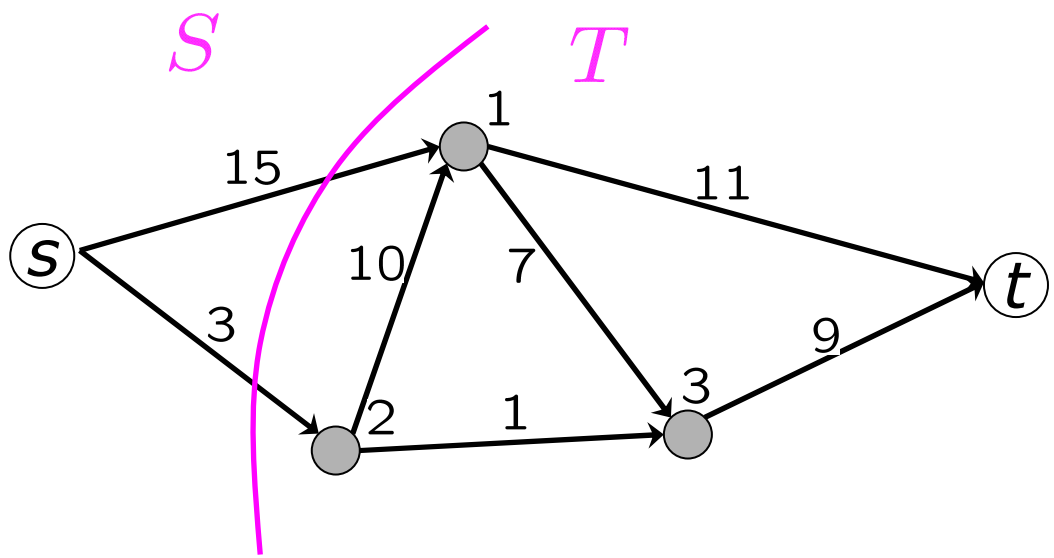
- A Network is a graph *G* with *nodes* and *edges*: $G = (V, E)$
- Special nodes *s* (source) and *t* (sink)
- Edges carry flow
- Each edge (*i*,*j*) has a maximum capacity $c_{i,j}$
- An *s*-*t* cut (*S*,*T*) is a 2-partition of *V* such that *s in S*, *t in T*



- *Cut value*: the total *s*-*t* cap. across the cut=3+7+11=21

A capacitated network

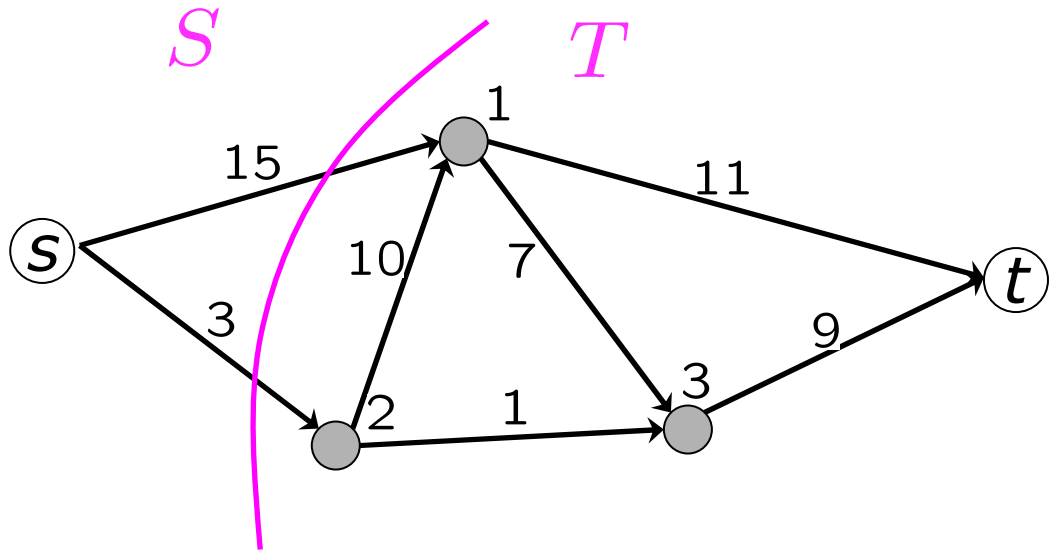
- A Network is a graph *G* with *nodes* and *edges*: $G = (V, E)$
- Special nodes *s* (source) and *t* (sink)
- Edges carry flow
- Each edge (*i*,*j*) has a maximum capacity $c_{i,j}$
- An *s*-*t* cut (*S*,*T*) is a 2-partition of *V* such that *s in S*, *t in T*
- A min *s*-*t* cut is one that gives the minimum cut value



- *Cut value*: the total *s*-*t* cap. across the cut=15+3=18

A capacitated network

- A Network is a graph *G* with *nodes* and *edges*: $G = (V, E)$
- Special nodes *s* (source) and *t* (sink)
- Edges carry flow
- Each edge (*i*,*j*) has a maximum capacity $c_{i,j}$
- An *s*-*t* cut (*S*,*T*) is a 2-partition of *V* such that *s in S*, *t in T*
- A min *s*-*t* cut is one that gives the minimum cut value
- *Finding a min-cut* = *finding a max-flow*



- *Cut value*: the total *s*-*t* cap. across the cut=15+3=18

Max flow problem

$G = (V, E)$

dual var. $\max_{x,v} \quad v$

s.t.

u_i $$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} v & i = s \\ 0 & i \in V \setminus \{s,t\} \\ -v & i = t \end{cases}$$

δ_{ij} $$0 \le x_{ij} \le c_{ij}, \ \forall (i,j) \in E.$$

Max flow problem $\qquad G = (V, E)$

dual var. $\max_{x,v} \quad v$

s.t.

u_i $$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} v & i = s \\ 0 & i \in V \setminus \{s,t\} \\ -v & i = t \end{cases}$$

δ_{ij} $$0 \le x_{ij} \le c_{ij}, \ \forall (i,j) \in E.$$

Min cut problem (dual of above)

$$\min_{u,\delta} \quad \sum_{(i,j)\in E} c_{ij}\delta_{ij}$$

s.t. $$u_i - u_j + \delta_{ij} \ge 0, \ \forall (i,j) \in E$$

$$u_s = 0, \ u_t = 1 \qquad (0 \le u_i \le 1, \ \forall i \text{ implicitly})$$

$$\delta_{ij} \ge 0, \ \forall (i,j) \in E.$$

$\exists$ a **binary** optimal solution u^*, δ^*. Min cut is given by $S := \{i : u_i^* = 0\}$, $T := \{i : u_i^* = 1\}$.

In 1D, discrete $\min_u \; TV(u) + \lambda \|f - u\|_{L^1}$ gives $\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$.

Let's consider the **binary** problem.

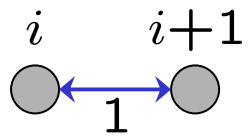
In 1D, discrete $\min_u \; TV(u) + \lambda \|f - u\|_{L^1}$ gives
$\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$.
Let's consider the **binary** problem.

1. Use $\min |x| \Leftrightarrow \min y_1 + y_2$, s.t. $x \leq y_1, \; -x \leq y_2, \; y_1, y_2 \geq 0$:

In 1D, discrete $\min_u \; TV(u) + \lambda \|f - u\|_{L^1}$ gives
$\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$.
Let's consider the **binary** problem.

1. Use $\min |x| \Leftrightarrow \min y_1 + y_2$, s.t. $x \le y_1, \; -x \le y_2, \; y_1, y_2 \ge 0$:
   We have
   each $\min \; |u_{i+1} - u_i| \;\Leftrightarrow\; \min \delta_{i,i+1} + \delta_{i+1,i}$

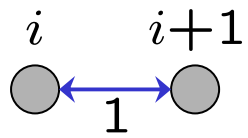
$$\begin{aligned} \text{s.t. } & u_{i+1} - u_i + \delta_{i+1,i} \ge 0 \\ & u_i - u_{i+1} + \delta_{i,i+1} \ge 0 \\ & \delta_{i+1,i}, \delta_{i,i+1} \ge 0. \end{aligned}$$



In 1D, discrete $\min_u \; TV(u) + \lambda \|f - u\|_{L^1}$ gives
$\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$.
Let's consider the **binary** problem.

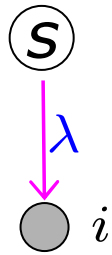
1. Use $\min |x| \Leftrightarrow \min y_1 + y_2$, s.t. $x \le y_1, \; -x \le y_2, \; y_1, y_2 \ge 0$:
   We have
   each $\min \; |u_{i+1} - u_i| \;\Leftrightarrow\; \min \delta_{i,i+1} + \delta_{i+1,i}$

$$\begin{aligned} \text{s.t. } & u_{i+1} - u_i + \delta_{i+1,i} \ge 0 \\ & u_i - u_{i+1} + \delta_{i,i+1} \ge 0 \\ & \delta_{i+1,i}, \delta_{i,i+1} \ge 0. \end{aligned}$$



2. Each $\min \; \lambda|0 - u_i| \Leftrightarrow \min \lambda\delta_{si}$, s.t. $\delta_{si} \ge u_i \quad (u_i \ge 0$ implicitly$)$

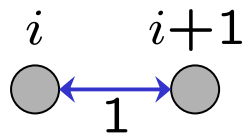
$$\Leftrightarrow \min \lambda\delta_{si}, \text{ s.t. } \begin{aligned} & u_s - u_i + \delta_{si} \ge 0 \\ & u_s = 0 \end{aligned}$$



In 1D, discrete $\min_u \; TV(u) + \lambda \|f - u\|_{L^1}$ gives
$\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$.
Let's consider the **binary** problem.

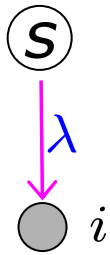
1. Use $\min |x| \Leftrightarrow \min y_1 + y_2$, s.t. $x \le y_1, \; -x \le y_2, \; y_1, y_2 \ge 0$:
   We have
   each $\min \; |u_{i+1} - u_i| \;\Leftrightarrow\; \min \delta_{i,i+1} + \delta_{i+1,i}$

$$\text{s.t. } u_{i+1} - u_i + \delta_{i+1,i} \ge 0$$
$$u_i - u_{i+1} + \delta_{i,i+1} \ge 0$$
$$\delta_{i+1,i}, \delta_{i,i+1} \ge 0.$$



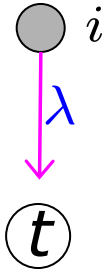
2. Each $\min \; \lambda|0 - u_i| \Leftrightarrow \min \lambda\delta_{si}$, s.t. $\delta_{si} \ge u_i$ $\quad$ ($u_i \ge 0$ implicitly)

$$\Leftrightarrow \min \lambda\delta_{si}, \text{ s.t. } u_s - u_i + \delta_{si} \ge 0$$
$$u_s = 0$$



3. Each $\min \; \lambda|1 - u_i| \Leftrightarrow \min \lambda\delta_{it}$, s.t. $\delta_{it} \ge 1 - u_i$ $\quad$ ($u_i \le 1$ implicitly)

$$\Leftrightarrow \min \lambda\delta_{si}, \text{ s.t. } u_i - u_t + \delta_{it} \ge 0$$
$$u_t = 1$$



Combining 1,2,3 gives a min cut formulation!

Alternative explanations for the 3 types of arcs:

In 1D, binary TV/L^1: $\min_U |\text{jumps}(U)| + \lambda |U \triangle F|$

1.: contribution to # of jumps in U

2. 3.: contribution to the length of $U \triangle F$

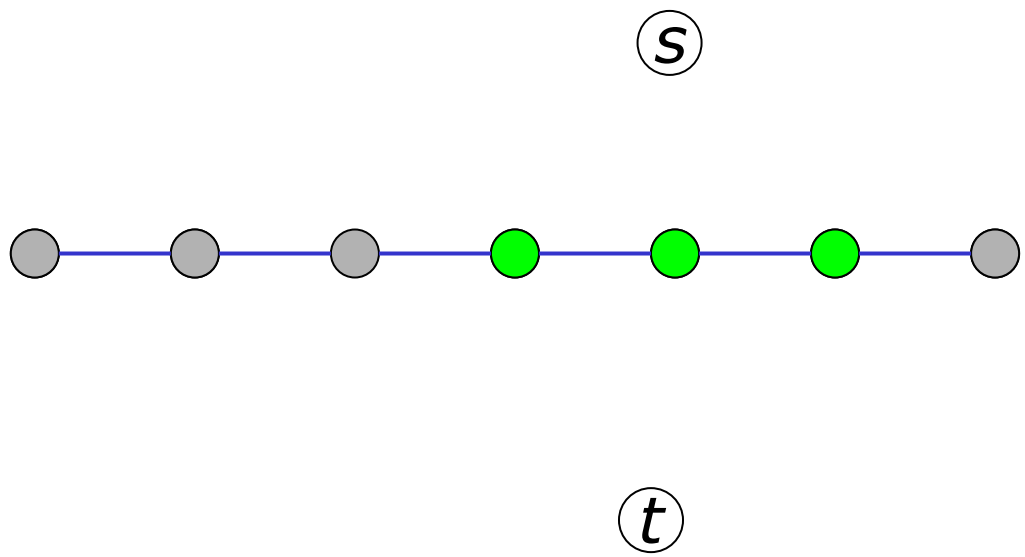
In 2D, binary TV/L^1: $\min_U \text{Per}(U) + \lambda \text{Area}(U \triangle F)$

1.: contribution to Per(U)

2. 3.: contribution to Area($U \triangle F$)

where $U \triangle F := (F \setminus U) \cup (U \setminus F)$

In 1D, discrete $\min_u \; TV(u) + \lambda \|f - u\|_{L^1}$ gives
$\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$.
Let's consider the **binary** problem.
Example: $u = (u_1, \ldots, u_7)$ and $f = (0, 0, 0, 1, 1, 1, 0)$.

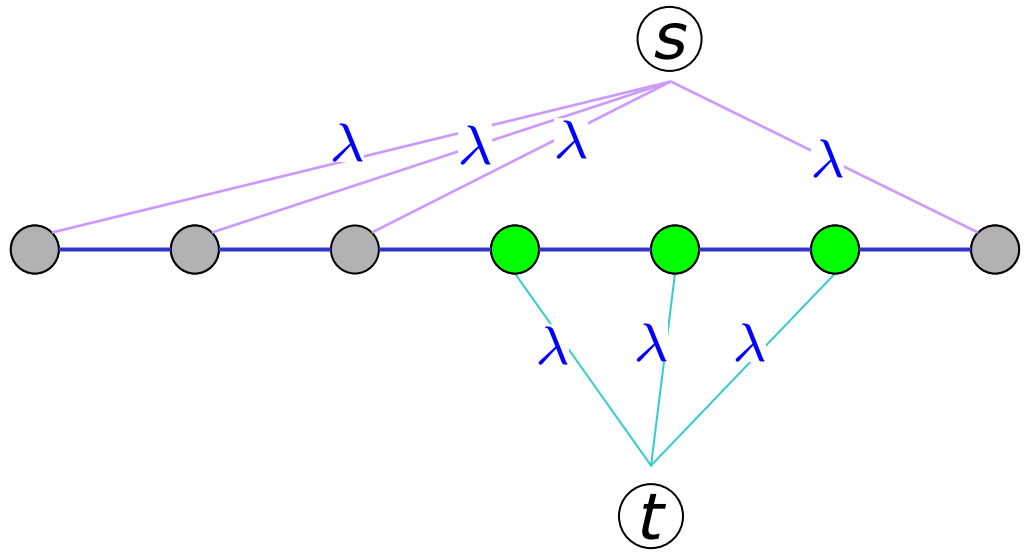


In 1D, discrete $\min_u \; TV(u) + \lambda \|f - u\|_{L^1}$ gives $\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$.

Let's consider the **binary** problem.

Example: $u = (u_1, \ldots, u_7)$ and $f = (0, 0, 0, 1, 1, 1, 0)$.

- construct a network with terminal arcs determined by f.

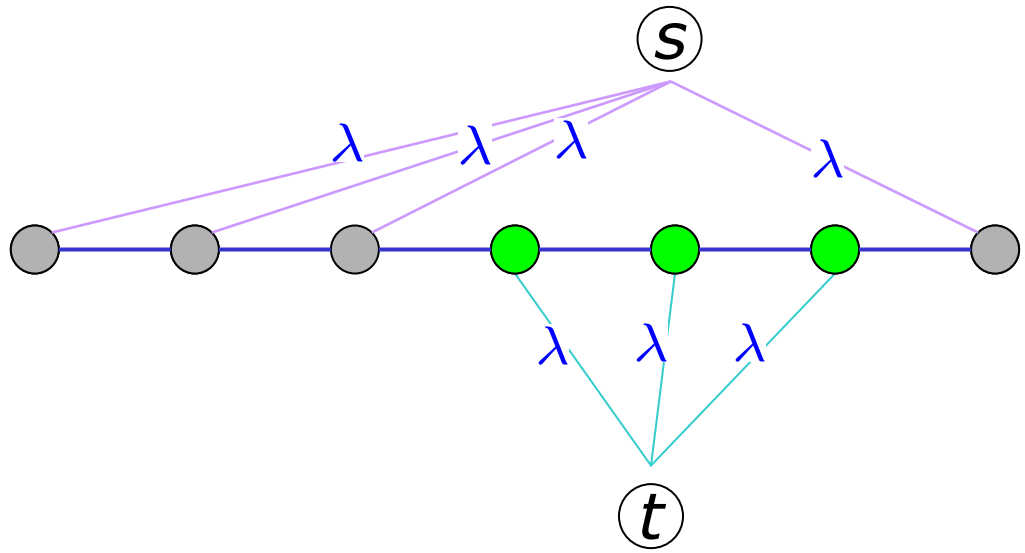


In 1D, discrete $\min_u \; TV(u) + \lambda \|f - u\|_{L^1}$ gives $\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$.

Let's consider the **binary** problem.

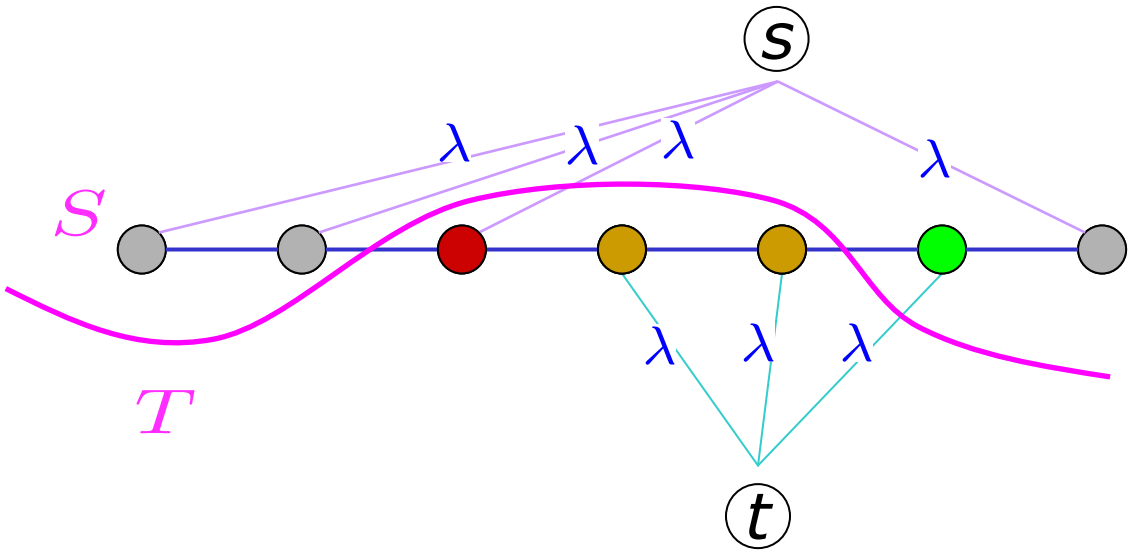
Example: $u = (u_1, \ldots, u_7)$ and $f = (0, 0, 0, 1, 1, 1, 0)$.

- construct a network with terminal arcs determined by f.
- define an S-T cut by letting $S = \{i : u_i = 0\}$.



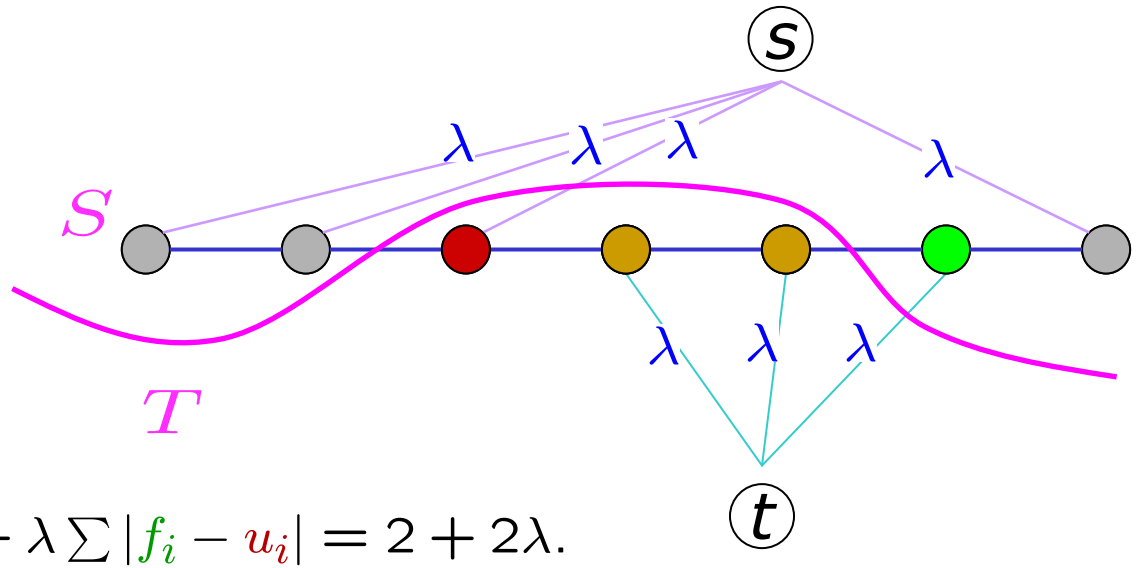
In 1D, discrete $\min_u \; TV(u) + \lambda \|f - u\|_{L^1}$ gives
$\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$.
Let's consider the **binary** problem.
Example: $u = (u_1, \ldots, u_7)$ and $f = (0, 0, 0, 1, 1, 1, 0)$.

- construct a network with terminal arcs determined by f.
- define an S-T cut by letting $S = \{i : u_i = 0\}$.
  example: $u_i = (0, 0, 1, 1, 1, 0, 0)$.



In 1D, discrete $\min_u \; TV(u) + \lambda \|f - u\|_{L^1}$ gives
$\min_u \sum_i |u_{i+1} - u_i| + \lambda \sum_i |f_i - u_i|$.
Let's consider the **binary** problem.
Example: $u = (u_1, \ldots, u_7)$ and $f = (0, 0, 0, 1, 1, 1, 0)$.

- construct a network with terminal arcs determined by f.
- define an S-T cut by letting $S = \{i : u_i = 0\}$.
  example: $u_i = (0, 0, 1, 1, 1, 0, 0)$.
- energy = cut value.

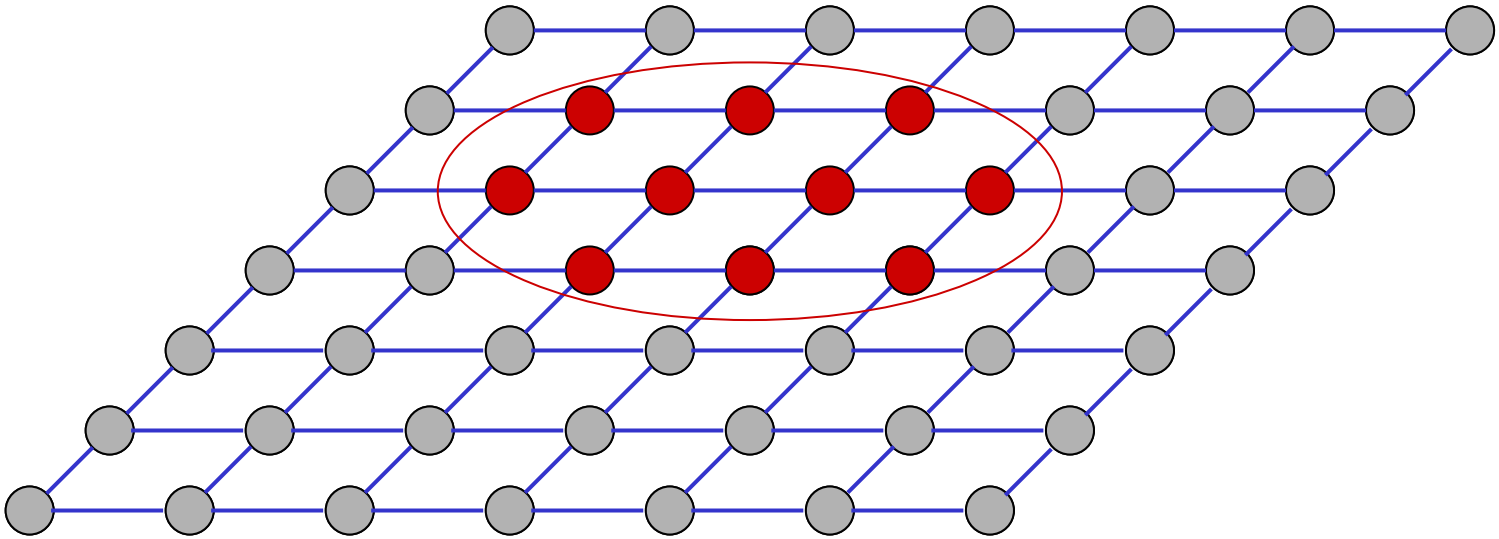


$\min_u \sum |u_{i+1} - u_i| + \lambda \sum |f_i - u_i| = 2 + 2\lambda$.

In 2D, we have

$U_l = \{$ ● $\}$ $\qquad$ $Per(U_l) =$ length(⬭)

$\approx |$edges cut by ⬭ $|$

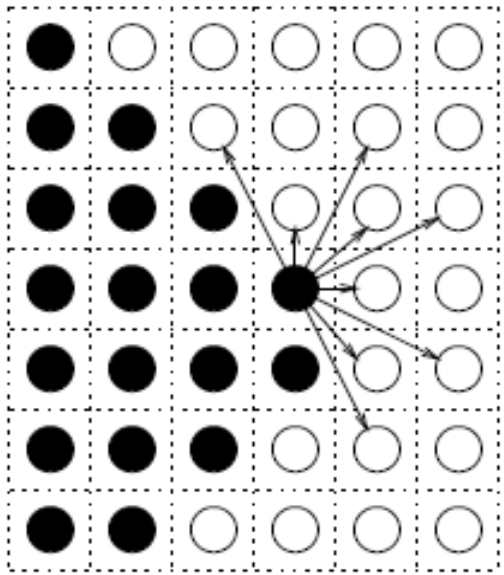


$Per(U_l)$ will get more accurately approximated if more neighbors are used

Isotropic *TV* v.s. Anisotropic *TV*

- $\nabla_1 u := u_{i+1,j} - u_{i,j}$ and $\nabla_2 u := u_{i,j+1} - u_{i,j}$
- The isotropic discretization of $TV(u)$: $\sqrt{|\nabla_1 u|^2 + |\nabla_2 u|^2}$
- An anisotropic discretization: $|\nabla_1 u| + |\nabla_2 u|$
- A better anisotropic discretization:

  Given a pixel $x \in U$, let $n_4(x)$, $n_8(x)$, and $n_k(x)$ be the number of pixels which are in the four, eight, and "knight-move" connected neighborhood of v outside S



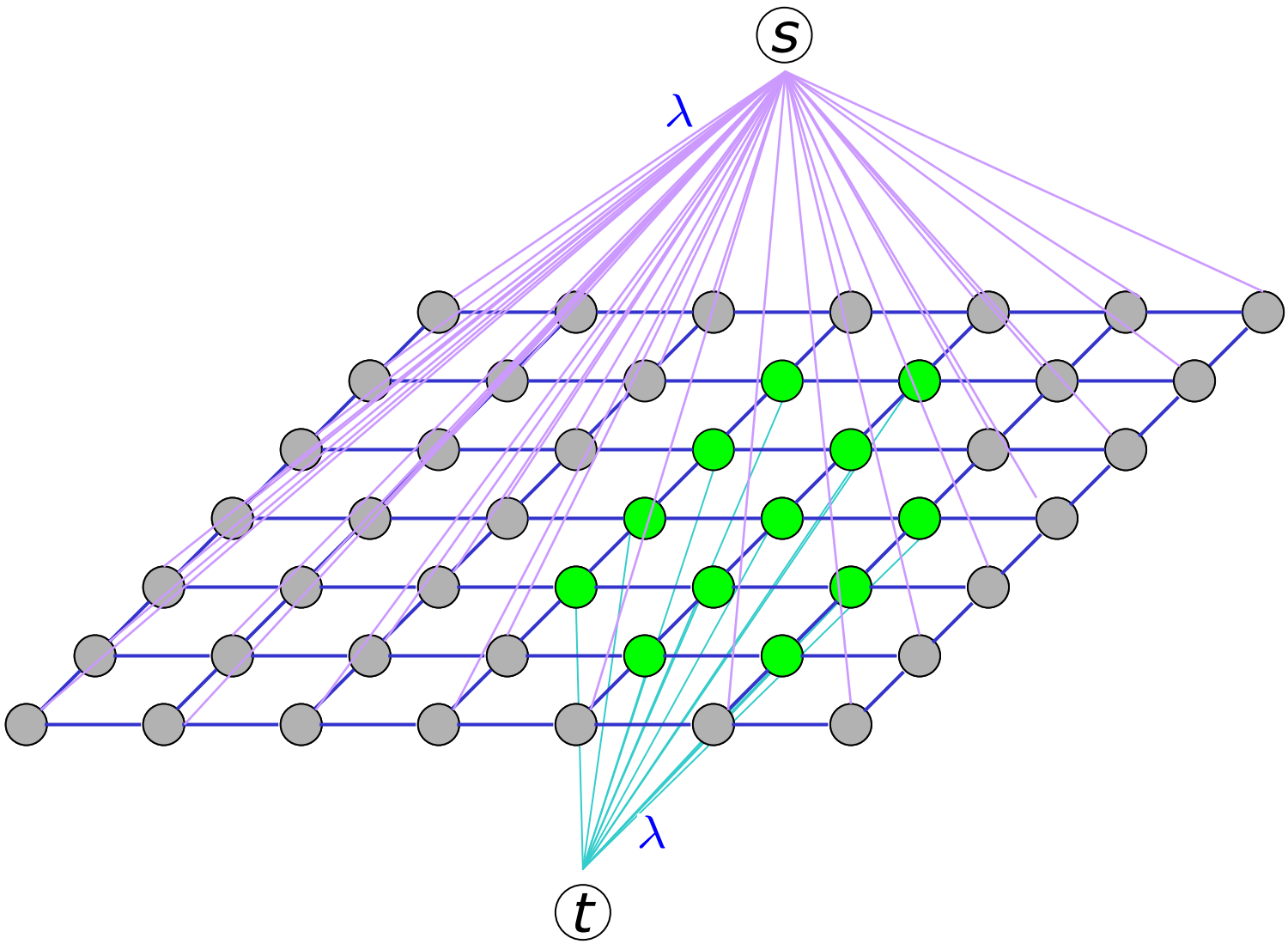
$n_4(x) = 2$
$n_8(x) = 2$
$n_k(x) = 5$

$$\mathrm{Per}(U) = \sum_{x \in U} [0.26 n_4(x) + 0.19 n_8(x) + 0.06 n_k(x)]$$

Watersnake: Nguyen-Worring-van den Boomgaard 03'

In 2D, we solve

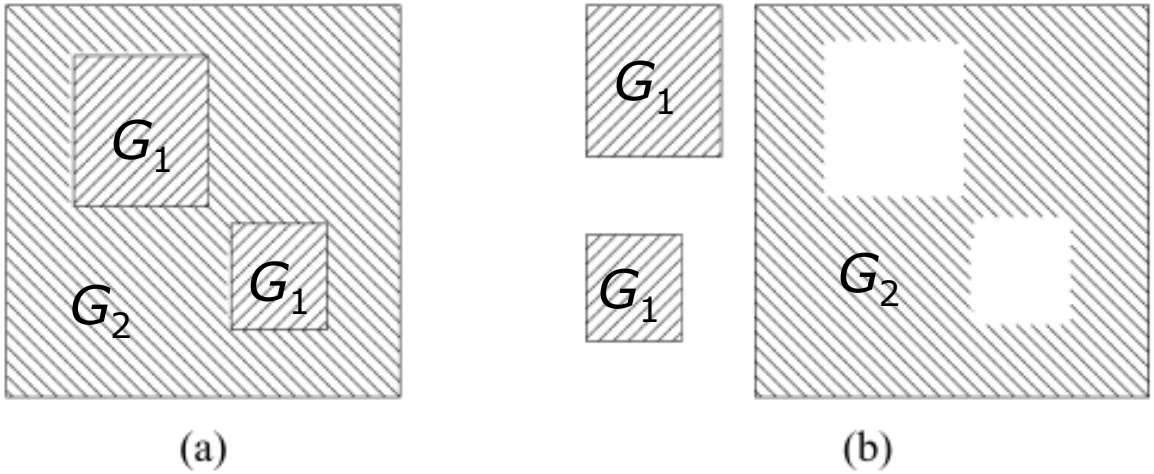
$$\min_{U_l} Per(U_l) + \lambda\, Area(F_l \triangle U_l)$$

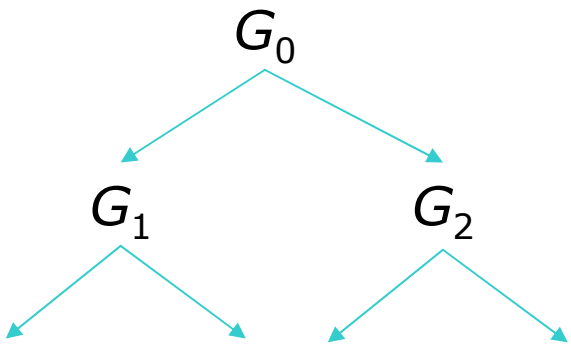


Outline: Further steps

1. Decompose f into K level sets F_i
2. For each F_i, obtain U_i
   1. U_i ⟵ min-cut of a network (Graph-Cut)
   2. min-cut ⟵ max-flow
   3. (For TV/L^1) Combine K networks (para. max flow)
   4. (For ROF) Reduce K max-flows to $\log K$ parametric max- flows (e.g., $K=2^{16}=65536$, $\log K=16$)
3. Construct a minimizer u from the minimizers U_i

Divide and conquer (Darbon & Siegelle)

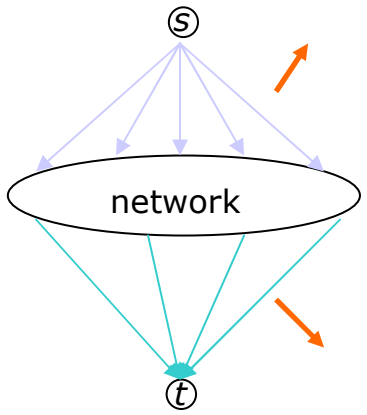




Divide and Conquer: Darbon-Siegelle 06’

Max flow / min cut algorithms

- Preflow push (Goldberg-Tarjan)
  - Best complexity: $O(nm\log(n^2/m))$
- Boykov-Kolmogrov, push on path[λ]
  - Uses approximate shortest path
  - Not strongly polynomial
  - Very fast on graph with small neighborhoods
- Parametric max flow (Gallo et al.)
  - Complexity same as preflow push: $O(nm\log(n^2/m))$, if # of levels is $O(n)$
  - Arcs out of source have *increasing* capacities
  - Arcs into sink have *decreasing* capacities



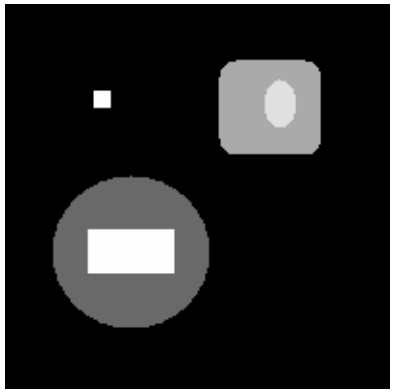
Preflow: Goldberg-Tarjan, B-K: Boykov-Kolmogrov 04'
Parametric: Gallo, Grigoriadis, Tarjan 89', Hochbaum 01'

Max-flow (Matlab/C++) numerical results

Model	Name	Size	best λ	total time
TV/L^1	Barbara (8-bit)	512×512	0.5	0.96
TV/L^1	Barbara (8-bit)	1024×1024	0.5	3.98
ROF	Barbara (8-bit)	512×512	0.0375	1.83
ROF	Barbara (8-bit)	1024×1024	0.0375	7.50
ROF	Barbara (**16-bit**)	1024×1024	0.0375	13.5

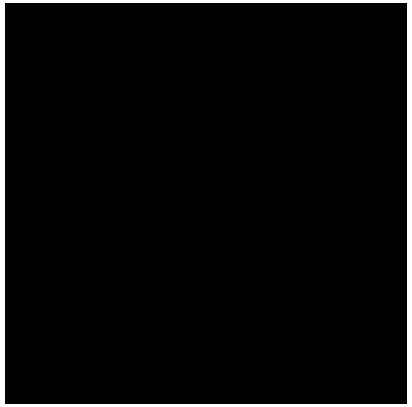
Laptop - CPU: Pentium Duo 2.0GHz, Memory: 1.5 GB

Input f



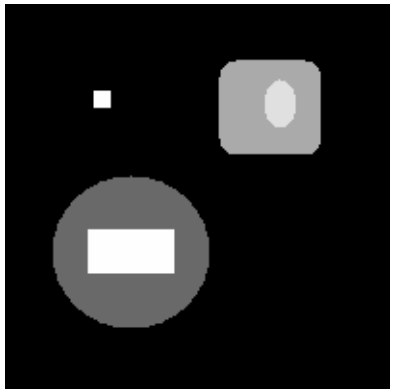
$\lambda > \lambda_1$

Output u



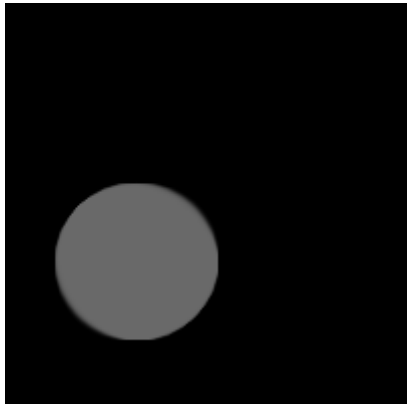
S_1	S_2	S_3	S_4	S_5
λ_1	λ_2	λ_3	λ_4	λ_5
19.40	13.40	7.96	4.57	2.35

Input f



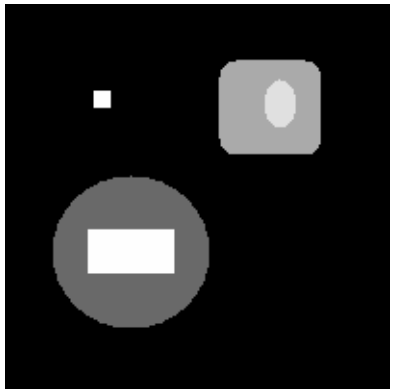
$\lambda_1 > \lambda > \lambda_2$

Output u



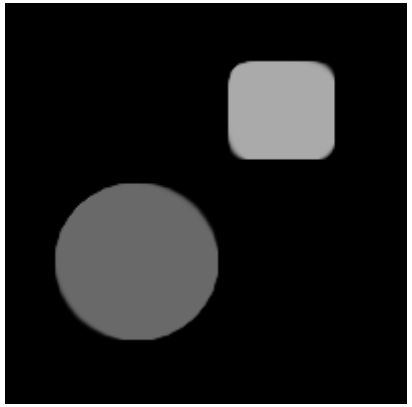
S_1	S_2	S_3	S_4	S_5
λ_1	λ_2	λ_3	λ_4	λ_5
19.40	13.40	7.96	4.57	2.35

Input f



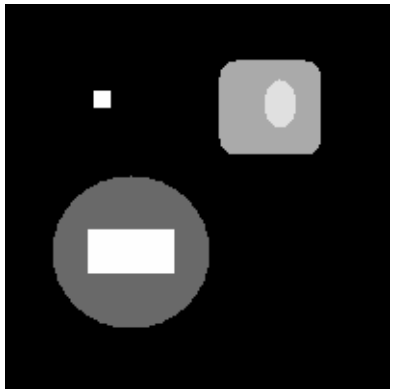
$\lambda_2 > \lambda > \lambda_3$

Output u



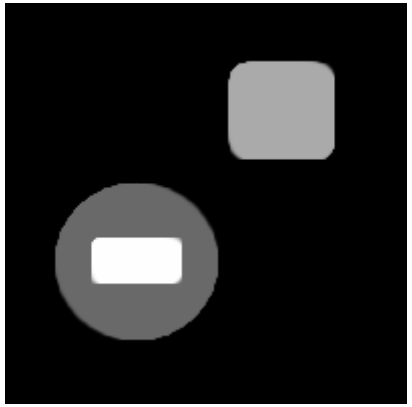
S_1	S_2	S_3	S_4	S_5
λ_1	λ_2	λ_3	λ_4	λ_5
19.40	13.40	7.96	4.57	2.35

Input f



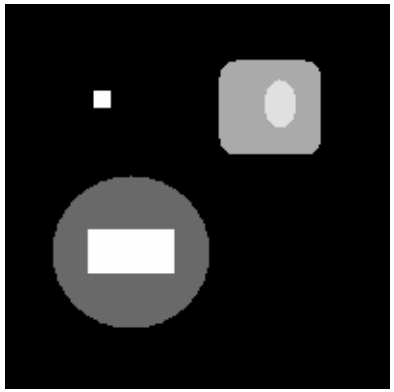
$\lambda_3 > \lambda > \lambda_4$

Output u



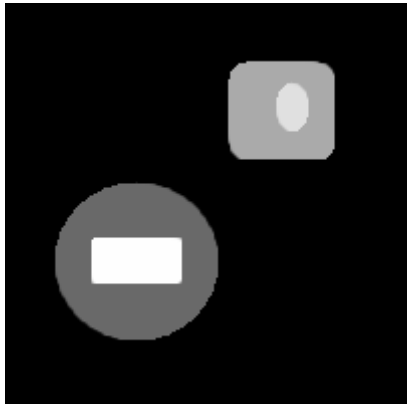
S_1	S_2	S_3	S_4	S_5
λ_1	λ_2	λ_3	λ_4	λ_5
19.40	13.40	7.96	4.57	2.35

Input f



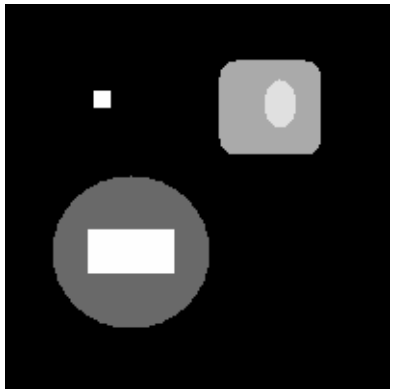
$\lambda_4 > \lambda > \lambda_5$

Output u



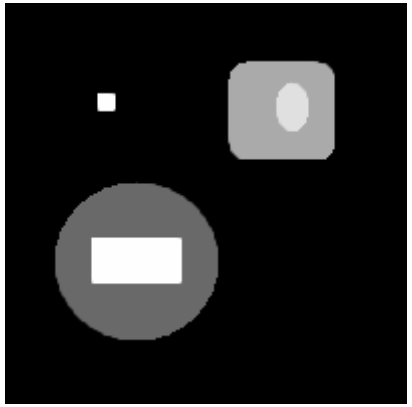
S_1	S_2	S_3	S_4	S_5
λ_1	λ_2	λ_3	λ_4	λ_5
19.40	13.40	7.96	4.57	2.35

Input f



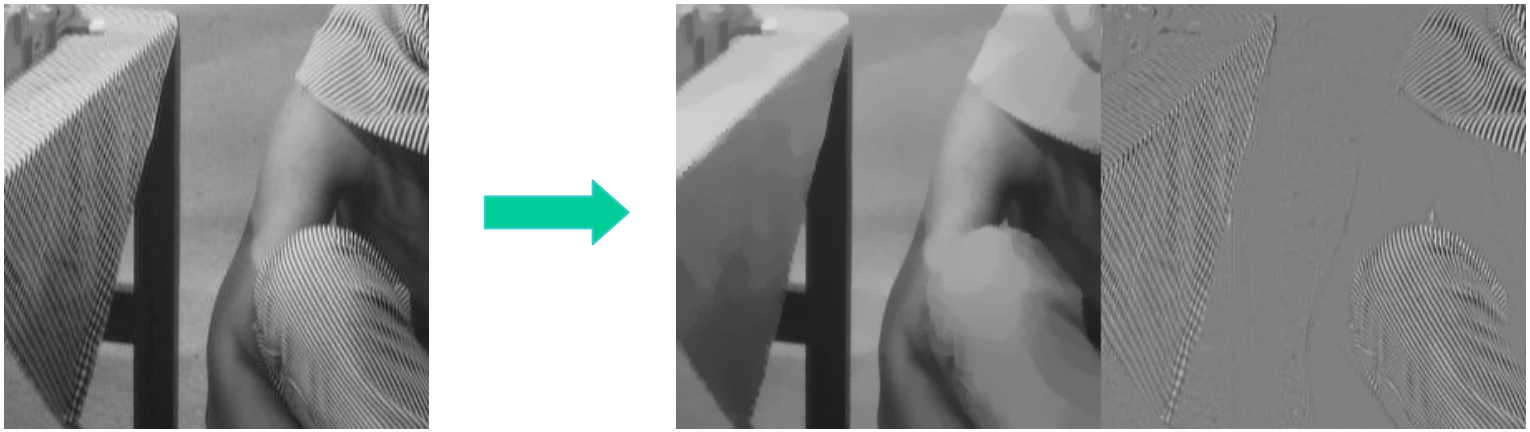
$\lambda_5 > \lambda > 0$

Output u

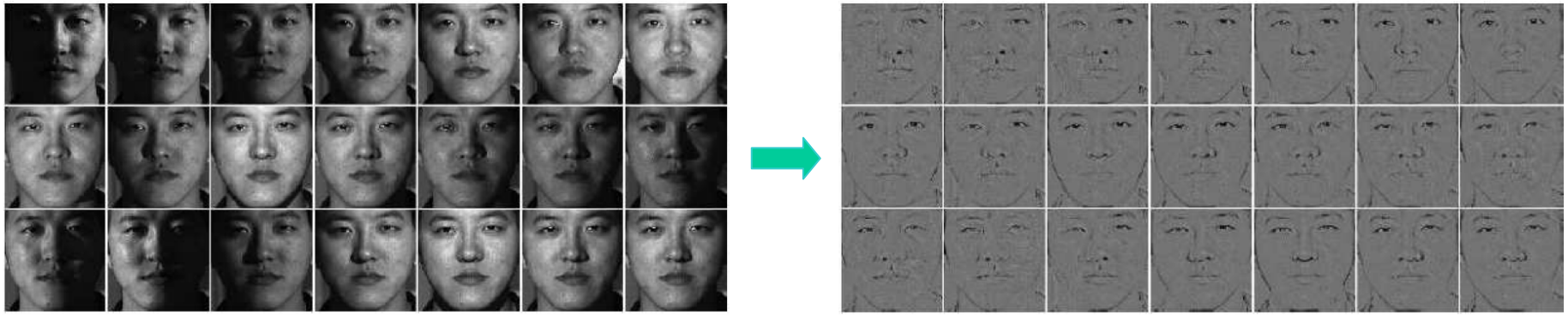


S_1	S_2	S_3	S_4	S_5
λ_1	λ_2	λ_3	λ_4	λ_5
19.40	13.40	7.96	4.57	2.35

$$f \ (TV/L^1) \Longrightarrow u + v$$



$$f \ (TV/L^1) \Longrightarrow u + v \Longrightarrow u/v$$



Face illumination correction: Chen-Yin-Zhou-Domaniciu-Huang 06’

Image	Barbara 512x512
Model	TV/L1 1-threaded
Grayscale	8-bit
Lambda	1.0
# neighbors	16
Initial graph construction time	0.48
Parametric max-flow time	1.37
Divide-n-conquer max-flow time	2.49

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux

Image	Noisy Barbara 512x512
Model	ROF Divide-n-Con.
Grayscale	8-bit
Lambda	0.025
# neighbors	16
Initial graph construction time	0.84
Parametric max-flow time	3.59
Divide-n-conquer max-flow time	6.39

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux

Image	CT 512x512
Model	TV/L1 1-threaded
Grayscale	8-bit
Lambda	0.5
# neighbors	16
Initial graph construction time	0.78
Parametric max-flow time	2.47
Divide-n-conquer max-flow time	4.96

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux

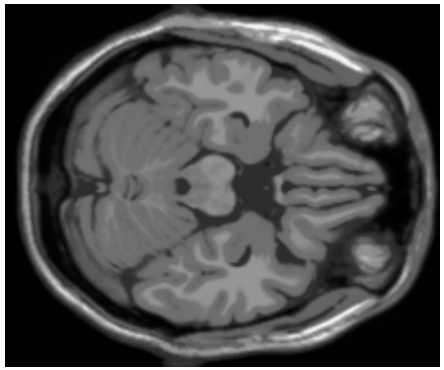
Image	CT 512x512
Model	TV/L1 1-threaded
Grayscale	8-bit
Lambda	1.0
# neighbors	16
Initial graph construction time	0.48
Parametric max-flow time	1.12
Divide-n-conquer max-flow time	2.33

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux

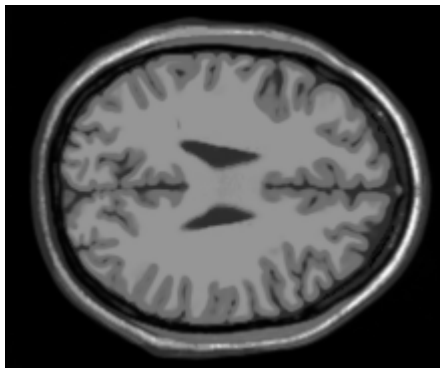
3D Brain MRI Image, Original, Size: 181x217x181 (1mmx1mmx1mm)
Used as input for TV/L1

T1 3D image:

z=50

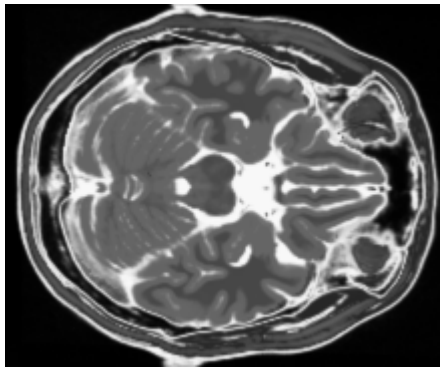


z=100

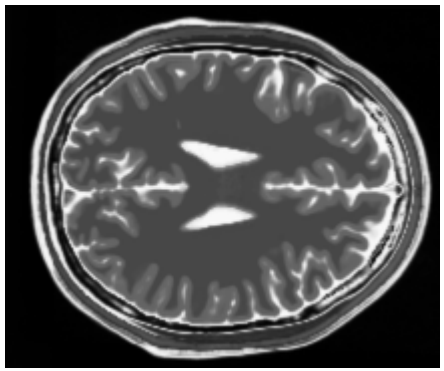


T2 3D image:

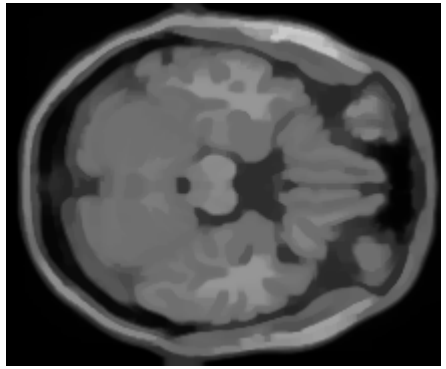
z=50



z=100



z=50



z=100

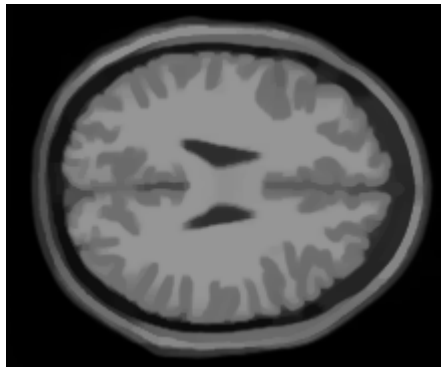


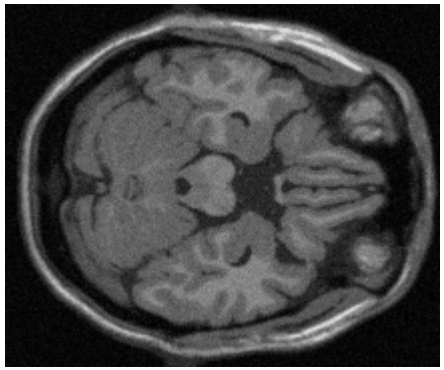
Image	Brain MRI T1 181x217x181
Model	TV/L1 1-threaded
Grayscale	8-bit
Lambda	1.0
# neighbors	6
Initial graph construction time	6.55
Parametric max-flow time	23.18
Divide-n-conquer max-flow time	67.56

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux

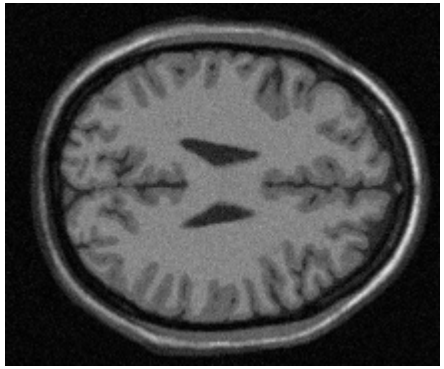
3D Brain MRI Image, 5% noise, Size: 181x217x181 (1mmx1mmx1mm)
Used as input for ROF

T1 3D image:

z=50

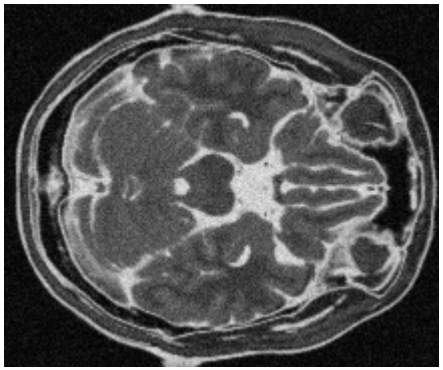


z=100

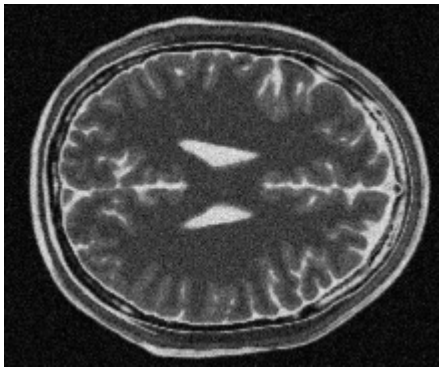


T2 3D image:

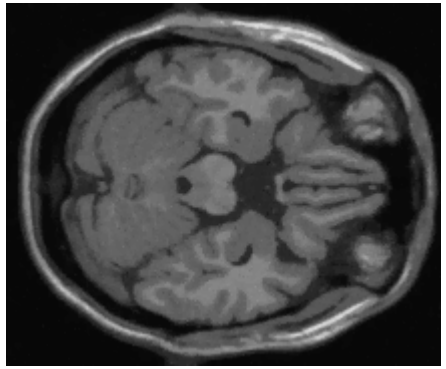
z=50



z=100



z=50



z=100

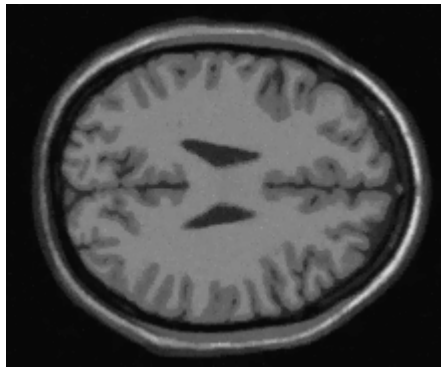
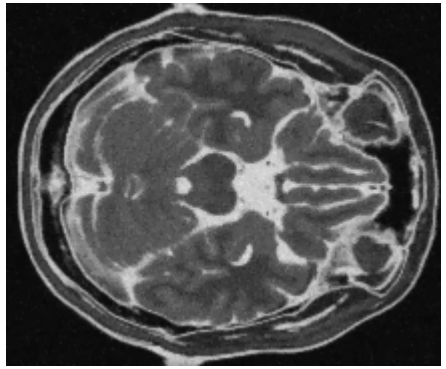


Image (noisy)	Brain MRI T1 181x217x181
Model	ROF Divide-n-Con.
Grayscale	8-bit
Lambda	0.3
# neighbors	6
Initial graph construction time	7.09
Parametric max-flow time	
Divide-n-conquer max-flow time	41.18

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux

z=50



z=100

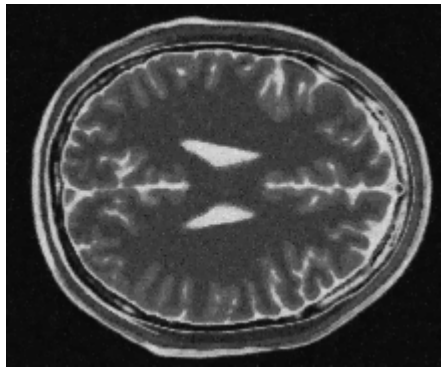


Image (noisy)	Brain MRI T2 181x217x181
Model	ROF Divide-n-Con.
Grayscale	8-bit
Lambda	0.3
# neighbors	6
Initial graph construction time	7.11
Parametric max-flow time	
Divide-n-conquer max-flow time	34.12

CPU: AMD Opteron 1Ghz, Memory: 3GB Op. System: Linux

Iterative regularization

$$\min_u \{J(u) + H(u, f)\}$$

J and H are convex functionals of u, $J(u) \geq 0, J(0) = 0, H(u, f) \geq 0$

Iterate $$u_k = \arg\min_{u \in BV(\Omega)} \{J(u) + H(u, f) - < u, p_{k-1} >\}$$

where subgradient $p_0 = 0 \in \partial J(u_0)$, $p_{k-1} \in \partial J(u_{k-1})$

Bregman Distance

$$D(u,v) = J(u) - J(v) - < p, u - v >$$

for $p \in \partial J(u)$, is the Bregman distance assoc. with $J(\cdot)$

- $D(u,v) \geq 0$ and $D(u,v) = 0$ iff $u = v$ (for $J(\cdot)$ str. convex)
- $D(u,v) \neq D(v,u)$ in general
- Δ inequality does not hold

Convergence Analysis

- $\{H(u_k, f)\}$ is monotonically non - increasing

$$H(u_k, f) \le H(u_k, f) + D(u_k, u_{k-1}) \le H(u_{k-1}, f)$$

- MAIN THEOREM

If (i) g is the true noise free image

(ii) $H(g, g) = 0$

(iii) $H(g, f) \le \delta^2$ (δ^2 : noise level)

then the distance between u_k and g decreases, i.e.,

$$D(g, u_k) \le D(g, u_k) + D(u_k, u_{k-1}) \le D(g, u_{k-1})$$

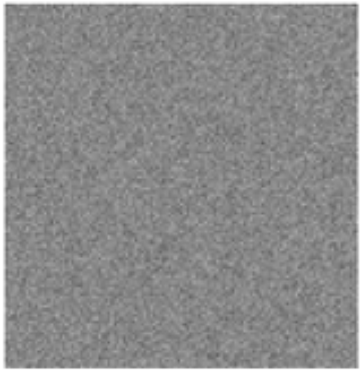
as long as $H(u_k, f) > \delta^2$

One-step ROF

(a) original

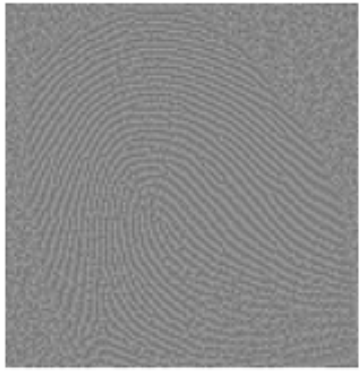
(b) noisy f, SNR=14.8

(c) noise+128, δ=10.0



(d) u: ROF

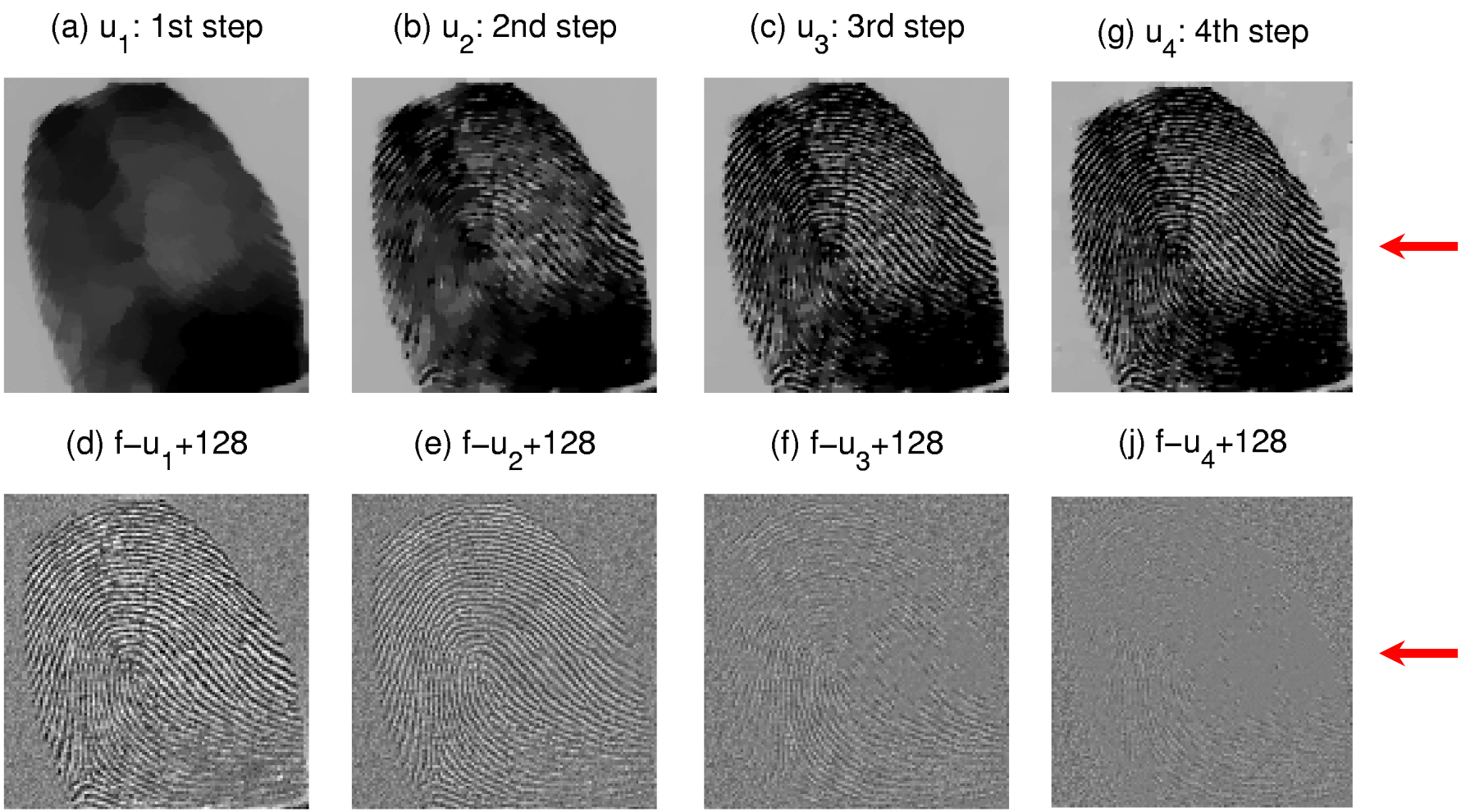
(e) f−u+128, $||f-u||_{L^2}$=10.2



(f) (f−u)−noise+128

ROF with $\lambda = 0.085$, $||f-u||_{L^2} \approx \delta$, *signal contained in* $v = f - u$.

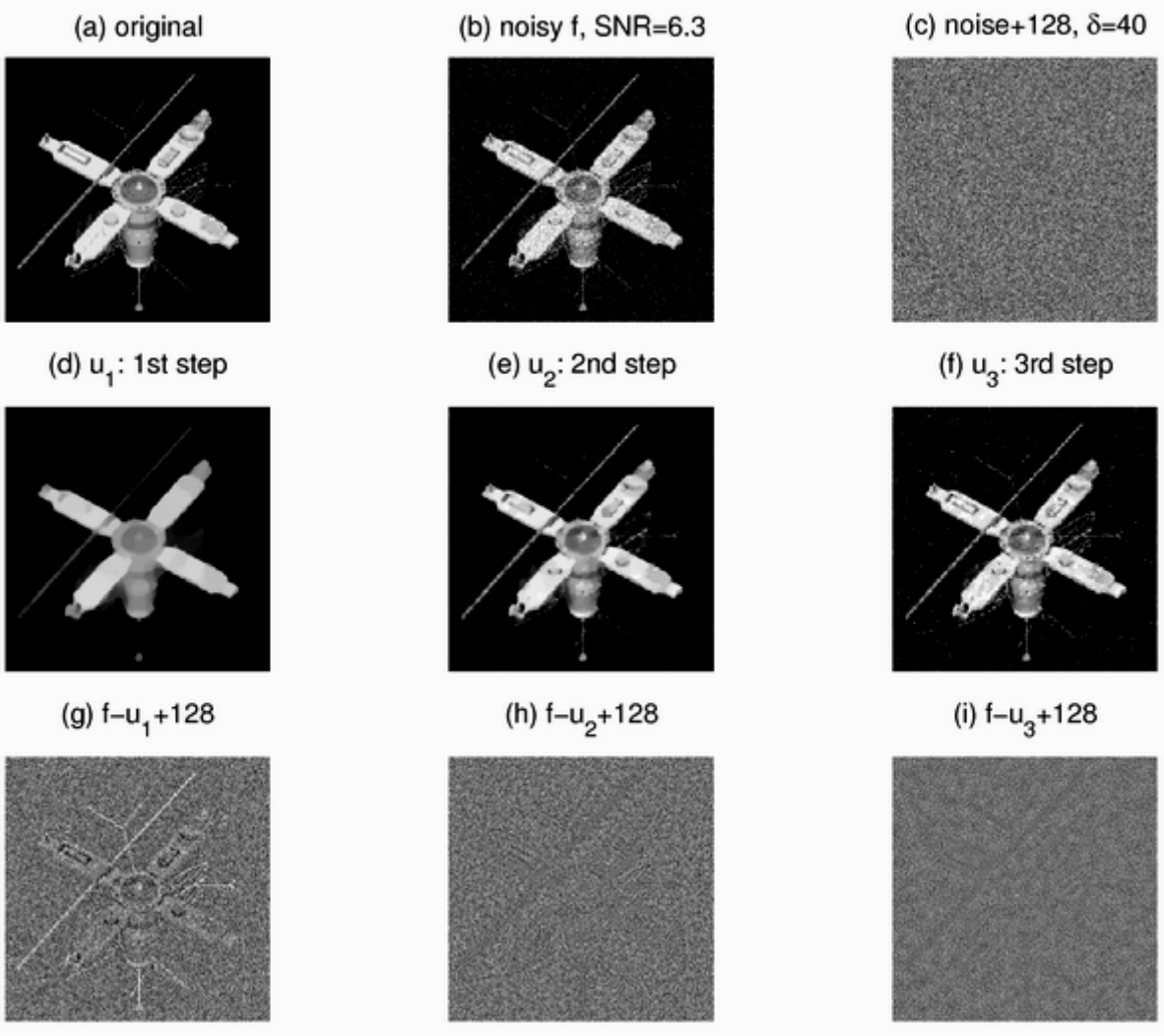
Iterative ROF regularization



ROF with iterated refinement with $\delta = 10$ and $\lambda = 0.013$

Best resolution obtained at $k = 4$. Noise returns in $u_5, u_6, \ldots$

Iterated ROF regularization



Deblurring + Denoising

Gaussian blur/noise, δ=10 and λ=0.1