On a first-order primal-dual algorithm

Thomas Pock[1] and Antonin Chambolle[2]

[1]Institute for Computer Graphics and Vision,
Graz University of Technology,
8010 Graz, Austria
[2]Centre de Mathématiques Appliquées, Ecole Polytechnique,
91128 Palaiseau, France

CMP Colloquium, Prague, 21.10.2010

Computer vision deals with inverse problems

- In computer vision, we have to determine the model parameters based on observations $\rightarrow$ **inverse problem**
- Computer vision problems are typically ill-posed

Computer vision deals with inverse problems

- In computer vision, we have to determine the model parameters based on observations $\rightarrow$ **inverse problem**
- Computer vision problems are typically ill-posed

Regularization

- How to infer a physically meaningful solution?

Regularization

- How to infer a physically meaningful solution?
- Idea is to introduce a certain smoothness assumption on the solution

Regularization

- How to infer a physically meaningful solution?
- Idea is to introduce a certain smoothness assumption on the solution

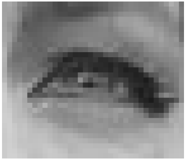
This leads to the **variational approach**:

$$\min_u \mathcal{R}(u) + \|Ku - f\| \ ,$$

Instead of trying to solve the problem exactly, the variational approach tries to find a tradeoff between data fit and smoothness.

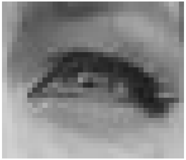
Which regularization for images?

- Images exhibit a high degree of spatial coherence
- Given the intensity of some pixel, it is very likely, that its neighboring pixels have the same intensity



Which regularization for images?

- Images exhibit a high degree of spatial coherence
- Given the intensity of some pixel, it is very likely, that its neighboring pixels have the same intensity



- It turns out that the so-called total variation (TV) is a good candidate for imaging problems

$$\mathcal{R}(u) = \int_\Omega |\nabla u| dx$$

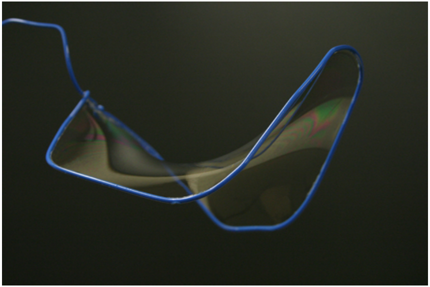
- Generalization to higher order derivatives: Total generalized variation (TGV) [Bredies, Kunisch, Pock '10]

Why to pose the inverse problem as an optimization problem?

- Nature solves optimization problems all the time
- **Example:** Soapfilms, Trees, ...

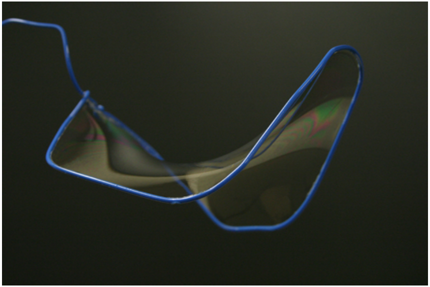
Why to pose the inverse problem as an optimization problem?

- Nature solves optimization problems all the time
- **Example:** Soapfilms, Trees, ...



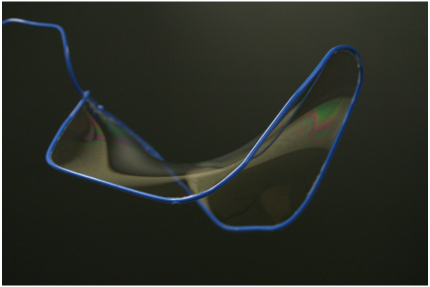
Why to pose the inverse problem as an optimization problem?

- Nature solves optimization problems all the time
- **Example:** Soapfilms, Trees, ...



Why to pose the inverse problem as an optimization problem?

- Nature solves optimization problems all the time
- **Example:** Soapfilms, Trees, ...



- Many laws of nature are nothing but optimality conditions
- Often expressed in terms of a minimum energy principle

Optimization problems are unsolvable

A general mathematical optimization problem it can be written as:

$$\begin{aligned} & \min f_0(x) \\ \text{s.t.} \quad & f_i(x) \le 0 \;, \quad i = 1 \ldots m \\ & x \in \mathcal{S} \;, \end{aligned}$$

where $f_0(x) ... f_m(x)$ are real-valued functions, $x = (x_1, ... x_n)^T \in \mathbb{R}^n$ is a n-dimensional real-valued vector, and $\mathcal{S}$ is a subset of $\mathbb{R}^n$

How to solve this problem?

Optimization problems are unsolvable

A general mathematical optimization problem it can be written as:

$$\begin{aligned} &\min f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \;, \quad i = 1 \ldots m \\ & x \in \mathcal{S} \;, \end{aligned}$$

where $f_0(x) \ldots f_m(x)$ are real-valued functions, $x = (x_1, \ldots x_n)^T \in \mathbb{R}^n$ is a n-dimensional real-valued vector, and $\mathcal{S}$ is a subset of $\mathbb{R}^n$

How to solve this problem?

- Naive: "Download a commercial package ..."

Optimization problems are unsolvable

A general mathematical optimization problem it can be written as:

$$\begin{aligned} & \min f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \,, \quad i = 1 \ldots m \\ & x \in \mathcal{S} \,, \end{aligned}$$

where $f_0(x)\ldots f_m(x)$ are real-valued functions, $x = (x_1, \ldots x_n)^T \in \mathbb{R}^n$ is a n-dimensional real-valued vector, and $\mathcal{S}$ is a subset of $\mathbb{R}^n$

How to solve this problem?

- Naive: "Download a commercial package ..."
- Reality: "Finding a solution is far from being trivial!"

Optimization problems are unsolvable

A general mathematical optimization problem it can be written as:

$$\begin{aligned} & \min f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \,, \quad i = 1 \ldots m \\ & x \in \mathcal{S} \,, \end{aligned}$$

where $f_0(x) \ldots f_m(x)$ are real-valued functions, $x = (x_1, \ldots x_n)^T \in \mathbb{R}^n$ is a n-dimensional real-valued vector, and $\mathcal{S}$ is a subset of $\mathbb{R}^n$

How to solve this problem?

- Naive: "Download a commercial package ..."
- Reality: "Finding a solution is far from being trivial!"

"Optimization problems are unsolvable" [Nesterov '04]

Complexity bounds for global optimization (1)

- Consider the problem class $\mathcal{C}_0$ [Nesterov '04]

$$\min_{x \in \mathcal{B}_n} f(x) \ ,$$

where $\mathcal{B}_n$ is the n-dimensional unit box defined by

$$\mathcal{B}_n = \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1, \ i = 1...n\}$$

Complexity bounds for global optimization (1)

- Consider the problem class $\mathcal{C}_0$ [Nesterov '04]

$$\min_{x \in \mathcal{B}_n} f(x) \ ,$$

where $\mathcal{B}_n$ is the n-dimensional unit box defined by

$$\mathcal{B}_n = \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1, \ i = 1 \ldots n\}$$

- What is the lower complexity bound to find an ε - approximate solution?

Complexity bounds for global optimization (2)

A remark on Lipschitz continuity: A function $f(x)$ is called Lipschitz continuous on $\mathcal{B}_n$ if

$$|f(x) - f(y)| \leq L\|x - y\|_\infty, \ \forall x, y \in \mathcal{B}_n \ ,$$

where the constant L is an upper bound to the maximum steepness of $f(x)$

Complexity bounds for global optimization (2)

A remark on Lipschitz continuity: A function $f(x)$ is called Lipschitz continuous on $\mathcal{B}_n$ if

$$|f(x) - f(y)| \leq L\|x - y\|_\infty, \ \forall x, y \in \mathcal{B}_n \ ,$$

where the constant L is an upper bound to the maximum steepness of $f(x)$

- **Theorem** [Nesterov '04]
  For $\frac{1}{2}L > \varepsilon > 0$, the complexity of a zero-order method to find an ε - approximate solution of the problem class $\mathcal{C}_0$ is at least

$$\left(\left\lfloor \frac{L}{2\varepsilon} \right\rfloor\right)^n$$

Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$

Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$
- Results in at least 10^{20} function calls, that is, $31\,250\,000$ years on a workstation!

Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$
- Results in at least 10^{20} function calls, that is, $31\,250\,000$ years on a workstation!
- If we change n to $n + 1$ the estimate is multiplied by $1/\varepsilon = 100$ and hence would take much longer

Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$
- Results in at least 10^{20} function calls, that is, $31\,250\,000$ years on a workstation!
- If we change n to $n + 1$ the estimate is multiplied by $1/\varepsilon = 100$ and hence would take much longer
- Contrary, if we change ε to 8% we would need only two weeks

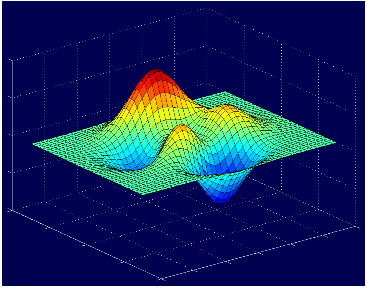
Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$
- Results in at least 10^{20} function calls, that is, $31\,250\,000$ years on a workstation!
- If we change n to $n + 1$ the estimate is multiplied by $1/\varepsilon = 100$ and hence would take much longer
- Contrary, if we change ε to 8% we would need only two weeks
- Complexity bounds for higher order methods (gradient descend, Newton, ...) are not much better

Example

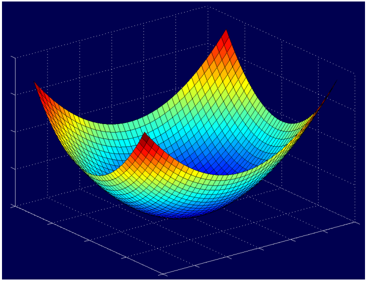
- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$
- Results in at least 10^{20} function calls, that is, $31\,250\,000$ years on a workstation!
- If we change n to $n + 1$ the estimate is multiplied by $1/\varepsilon = 100$ and hence would take much longer
- Contrary, if we change ε to 8% we would need only two weeks
- Complexity bounds for higher order methods (gradient descend, Newton, ...) are not much better
- Comparison to NP-hard problems: Hard combinatorial problems need 2^n arithmetic operations (only)!

Convex versus non-convex



- Non-convex problems
  - Often give more accurate models
  - In general no chance to find the global minimizer
  - Result strongly depends on the initialization
  - Dilemma: Wrong model or wrong algorithm?

Convex versus non-convex



- Convex problems
  - Convex models often inferior
  - Any local minimizer is a global minimizer
  - Result is independent of the initialization
  - Note: Convex does not mean easy!

A class of problems

Let us consider the following class of structured convex optimization problems [Chambolle, Pock '10]

$$\min_{x \in X} F(Kx) + G(x) \; ,$$

- $K : X \to Y$ is a linear and continuous operator from a Hilbert space X to a Hilbert space Y.
- F, G are "simple" convex, proper, l.s.c. functions, and hence easy to compute prox operator:

$$(1 + \tau \partial F)^{-1}(z) = \arg\min_{x} \frac{\|x - z\|^2}{2\tau} + F(x)$$

A class of problems

Let us consider the following class of structured convex optimization problems [Chambolle, Pock '10]

$$\min_{x \in X} F(Kx) + G(x) \ ,$$

- $K : X \to Y$ is a linear and continuous operator from a Hilbert space X to a Hilbert space Y.
- F, G are "simple" convex, proper, l.s.c. functions, and hence easy to compute prox operator:

$$(1 + \tau \partial F)^{-1}(z) = \arg\min_x \frac{\|x - z\|^2}{2\tau} + F(x)$$

- It turns out that many low-level vision problems can be cast in this framework.

Primal, dual, primal-dual

A nice feature of convex functions is duality.

Recall the convex conjugate:

$$F^*(p^*) = \max_p \langle p, p^* \rangle - F(p) \ ,$$

we can transform our initial problem [Rockafellar '70]

$$\min_{x \in X} F(Kx) + G(x) \quad \text{(Primal)}$$

Primal, dual, primal-dual

A nice feature of convex functions is duality.

Recall the convex conjugate:

$$F^*(p^*) = \max_p \langle p, p^* \rangle - F(p) \ ,$$

we can transform our initial problem [Rockafellar '70]

$$\min_{x \in X} F(Kx) + G(x) \quad \text{(Primal)}$$

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y) \quad \text{(Primal-Dual)}$$

Primal, dual, primal-dual

A nice feature of convex functions is duality.

Recall the convex conjugate:

$$F^*(p^*) = \max_p \langle p, p^* \rangle - F(p) \ ,$$

we can transform our initial problem [Rockafellar '70]

$$\min_{x \in X} F(Kx) + G(x) \quad \text{(Primal)}$$

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y) \quad \text{(Primal-Dual)}$$

$$\max_{y \in Y} -\left(F^*(y) + G^*(-K^*y)\right) \quad \text{(Dual)}$$

Primal, dual, primal-dual

A nice feature of convex functions is duality.

Recall the convex conjugate:

$$F^*(p^*) = \max_p \langle p, p^* \rangle - F(p) \ ,$$

we can transform our initial problem [Rockafellar '70]

$$\min_{x \in X} F(Kx) + G(x) \quad \text{(Primal)}$$

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y) \quad \text{(Primal-Dual)}$$

$$\max_{y \in Y} -\left(F^*(y) + G^*(-K^*y)\right) \quad \text{(Dual)}$$

Allows to compute the so-called primal-dual gap:

$$\mathcal{G}(x, y) = [F(Kx) + G(x)] + [F^*(y) + G^*(-K^*y)] \ ,$$

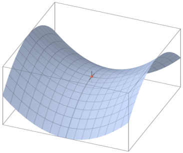
Optimality conditions

We focus on the primal-dual formulation:

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y)$$

We assume, there exists a saddle-point $(\hat{x}, \hat{y}) \in X \times Y$ which satisfies the Euler-Lagrange equations

$$\begin{cases} K\hat{x} - \partial F^*(\hat{y}) & \ni & 0 \\ K^*\hat{y} + \partial G(\hat{x}) & \ni & 0 \end{cases}$$



Example for a saddle-point of a convex-concave function

First-order versus second-order methods

- First order methods
  - Some even work in case of non-smoothness
  - Need only first order derivatives
  - More iterations but the cost of one iteration is low
- Second order methods
  - Need some smoothness in the function
  - Need first and second order derivatives and need to invert the Hessian matrix
  - Fewer iterations but the cost and memory per iterations is huge

First-order versus second-order methods

- First order methods
  - Some even work in case of non-smoothness
  - Need only first order derivatives
  - More iterations but the cost of one iteration is low
- Second order methods
  - Need some smoothness in the function
  - Need first and second order derivatives and need to invert the Hessian matrix
  - Fewer iterations but the cost and memory per iterations is huge
- What do we call an iteration?

Convergence rates

The notion of Q-linear convergence is defined as

$$\mu = \lim_{n\to\infty} \frac{\|x^{n+1} - \hat{x}\|}{\|x^n - \hat{x}\|}$$

sublinear convergence	if	$\mu = 1$
linear convergence	if	$\mu \in (0, 1)$
superlinear convergence	if	$\mu = 0$

Convergence rates

The notion of Q-linear convergence is defined as

$$\mu = \lim_{n\to\infty} \frac{\|x^{n+1} - \hat{x}\|}{\|x^n - \hat{x}\|}$$

sublinear convergence	if	$\mu = 1$
linear convergence	if	$\mu \in (0,1)$
superlinear convergence	if	$\mu = 0$

- Lower bound for black-box oriented first-order methods: $O(1/\sqrt{N})$[Nemirovski '83]

Convergence rates

The notion of Q-linear convergence is defined as

$$\mu = \lim_{n \to \infty} \frac{\|x^{n+1} - \hat{x}\|}{\|x^n - \hat{x}\|}$$

sublinear convergence	if	$\mu = 1$
linear convergence	if	$\mu \in (0, 1)$
superlinear convergence	if	$\mu = 0$

- Lower bound for black-box oriented first-order methods: $O(1/\sqrt{N})$[Nemirovski '83]
- Lower bound for any first-order method exploiting the structure of the problem: $O(1/N)$ [Nesterov '04]

Convergence rates

The notion of Q-linear convergence is defined as

$$\mu = \lim_{n\to\infty} \frac{\|x^{n+1} - \hat{x}\|}{\|x^n - \hat{x}\|}$$

sublinear convergence	if	$\mu = 1$
linear convergence	if	$\mu \in (0,1)$
superlinear convergence	if	$\mu = 0$

- Lower bound for black-box oriented first-order methods: $O(1/\sqrt{N})$[Nemirovski '83]
- Lower bound for any first-order method exploiting the structure of the problem: $O(1/N)$ [Nesterov '04]
- Note that this does not mean that there does not exist some other first order algorithm which is faster on a sub-class of problems

Standard approaches

- Classical Arrow-Hurwicz method [Arrow-Hurwicz, '58]
- Proximal-point algorithm [Martinet '70, Rockafellar '76]
- Douglas-Rachford splitting [Mercier,Lions '79]
- Extragradient-methods [Korpelevich '76, Popov '80]
- Nesterov's smoothing method [Nesterov '03]

Forward-backward splitting

- Consider the problem $\min_x f_1(x) + f_2(x)$, where $f_1(x)$ is a convex function and $f_2(x)$ is a convex function with L-Lipschitz continuous gradient ∇f_2, i.e.

$$\|\nabla f_2(x) - \nabla f_2(y)\| \leq L\|x - y\| \ , \forall x, y \in \mathsf{dom}(f_2)$$

- It can be shown that a minimizer can be characterized by the fixed point equation [Combettes, Pesquet '05]

$$x = (I + \tau \partial f_1)^{-1}(x - \tau \nabla f_2(x))$$

- **Note:** This is exactly the case for our saddle-point formulation

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y)$$

The algorithm

- Initialization: Choose $\tau, \sigma > 0$, $\theta \in [0,1]$, $(x^0, y^0) \in X \times Y$ and set $\bar{x}^0 = x^0$.
- Iterations $(n \geq 0)$: Update $x^n, y^n, \bar{x}^n$ as follows:

$$\begin{cases} y^{n+1} & = & (I + \sigma \partial F^*)^{-1}(y^n + \sigma K \bar{x}^n) \\ x^{n+1} & = & (I + \tau \partial G)^{-1}(x^n - \tau K^* y^{n+1}) \\ \bar{x}^{n+1} & = & x^{n+1} + \theta(x^{n+1} - x^n) \end{cases}$$

The algorithm has been first presented in a less general setting where G, F^* are restricted to indicator functions, by [Pock, Cremers, Bischof, Chambolle, ICCV'09]

Convergence of the algorithm

Theorem [Chambolle, Pock '10]
Let $L = \|K\|$ choose $\theta = 1$, $\tau\sigma L^2 < 1$ and let $(x^n, \bar{x}^n, y^n)$ be defined as in Algorithm 1.

(a) If the dimension of the spaces X and Y is finite, then there exists a saddle-point $(\hat{x}, \hat{y})$ such that $x^n \to \hat{x}$ and $y^n \to \hat{y}$ as $n \to \infty$.

(b) If we let $x_N = (\sum_{n=1}^{N} x^n)/N$ and $y_N = (\sum_{n=1}^{N} y^n)/N$, one has for all (x, y)

$$[\langle Kx_N, y\rangle - F^*(y) + G(x_N)] - [\langle Kx, y_N\rangle - F^*(y_N) + G(x)] \leq \frac{1}{N}\left(\frac{\|y - y^0\|^2}{2\sigma} + \frac{\|x - x^0\|^2}{2\tau}\right)$$

Moreover, the weak cluster points (x_N, y_N) are saddle points

Convergence rates

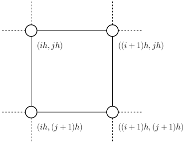
The algorithm gives optimal convergence rates on different subclasses by optimal choices on τ, σ, and θ.

- Completely non-smooth problem: $O(1/N)$ for the gap
- Objective function sum of a smooth and a non-smooth function: $O(1/N^2)$ for the error $\|x - x^*\|^2$
- Objective function completely smooth: $O(\omega^N)$, $\omega < 1$ for the error $\|x - x^*\|^2$

Proofs can be found in [Chambolle, Pock '10]

Parallel computing?

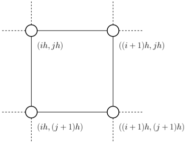
- The algorithm basically alternates updates of the primal and the dual variables.
- For most images, x and y are defined on a regular grid.



- Each iteration of the algorithm can be done fully in parallel
- The processor of my dreams would look like this:

Parallel computing?

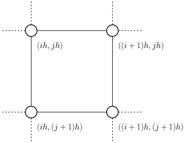
- The algorithm basically alternates updates of the primal and the dual variables.
- For most images, x and y are defined on a regular grid.



- Each iteration of the algorithm can be done fully in parallel
- The processor of my dreams would look like this:
  - One simple processor for each pixel

Parallel computing?

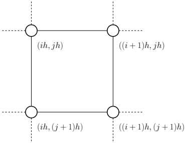
- The algorithm basically alternates updates of the primal and the dual variables.
- For most images, x and y are defined on a regular grid.



- Each iteration of the algorithm can be done fully in parallel
- The processor of my dreams would look like this:
  - One simple processor for each pixel
  - Each processor has a small amount of local memory

Parallel computing?

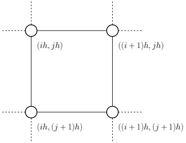
- The algorithm basically alternates updates of the primal and the dual variables.
- For most images, x and y are defined on a regular grid.



- Each iteration of the algorithm can be done fully in parallel
- The processor of my dreams would look like this:
  - One simple processor for each pixel
  - Each processor has a small amount of local memory
  - Each processor can inter-change data with its neighboring processors

Parallel computing?

- The algorithm basically alternates updates of the primal and the dual variables.
- For most images, x and y are defined on a regular grid.



- Each iteration of the algorithm can be done fully in parallel
- The processor of my dreams would look like this:
  - One simple processor for each pixel
  - Each processor has a small amount of local memory
  - Each processor can inter-change data with its neighboring processors
- **We have to think parallel from scratch!**

ROF denoising

Consider the ROF model for image denoising

$$\min_u \|\nabla u\|_{2,1} \; + \; \frac{\lambda}{2}\|u - g\|_2^2$$

Is the sum of a non-smooth and a sum function

(a) Clean

(b) Noisy

(c) Denoised

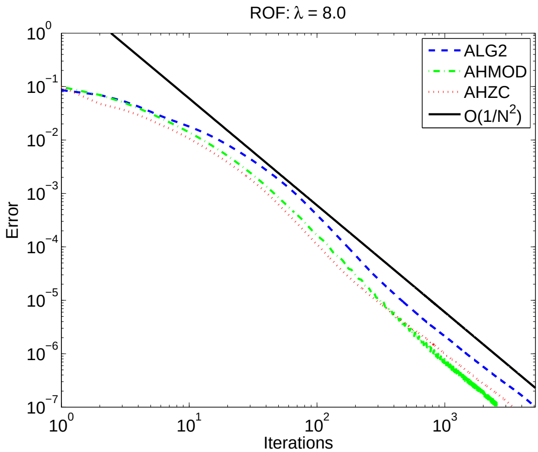
Example image used in the performance evaluation

Comparison

	$\lambda = 16$		$\lambda = 8$	
	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-6}$
PD	108 (1.95s)	937 (14.55s)	174 (2.76s)	1479 (23.74s)
AHZC	65 (0.98s)	634 (9.19s)	105 (1.65s)	1001 (14.48s)
FISTA	107 (2.11s)	999 (20.36s)	173 (3.84s)	1540 (29.48s)
NEST	106 (3.32s)	1213 (38.23s)	174 (5.54s)	1963 (58.28s)
ADMM	284 (4.91s)	25584 (421.75s)	414 (7.31s)	33917 (547.35s)
PGD	620 (9.14s)	58804 (919.64s)	1621 (23.25s)	–
CFP	1396 (20.65s)	–	3658 (54.52s)	–

- Arrow Hurwicz method performs best but can not be shown to converge within $O(1/N^2)$.
- PD performs slightly worse but still better than established $O(1/N^2)$ methods such as FISTA and Nesterov.

Convergence



Convergence of the top-performing methods for ROF model

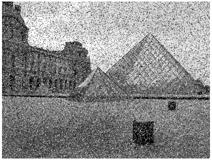
$TV - L^1$ denoising

The $TV - L^1$ model is given by

$$\min_u \|\nabla u\|_{2,1} + \lambda \|u - g\|_1 \ .$$

This problem is completely non-smooth.

(a) Clean image



(b) Noisy image

(c) ROF

(d) TV-L^1

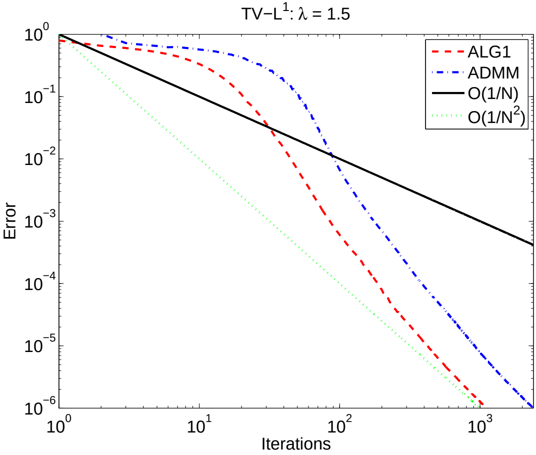
Example image used in the performance evaluation

Comparison

	$\lambda = 1.5$	
	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$
PD	187 (15.81s)	421 (36.02s)
ADMM	385 (33.26s)	916 (79.98s)
EGRAD	2462 (371.13s)	8736 (1360.00s)
NEST	2406 (213.41s)	15538 (1386.95s)

- PD performs best, ADMM reasonable well
- Nesterov's smoothing method seems to perform quite worse
- Paradoxically, it seems that for this example, our algorithm converges with $O(1/N^2)$ for the function value

Convergence



Convergence of the top-performing methods for the TV-L^1 model.

Huber denoising

The Huber model is given by

$$\min_u \|\nabla u\|_\alpha + \frac{\lambda}{2}\|u-g\|^2 .$$

where

$$F(y) = \|y\|_\alpha = \sum_{i,j} |\vec{y}_{i,j}|_\alpha$$

and

$$|\vec{p}|_\alpha = \begin{cases} \frac{|p|^2}{2\alpha} & \text{if } |p| \leq \alpha \\ |p| - \frac{\alpha}{2} & \text{else.} \end{cases}$$

This model is smooth in both terms.

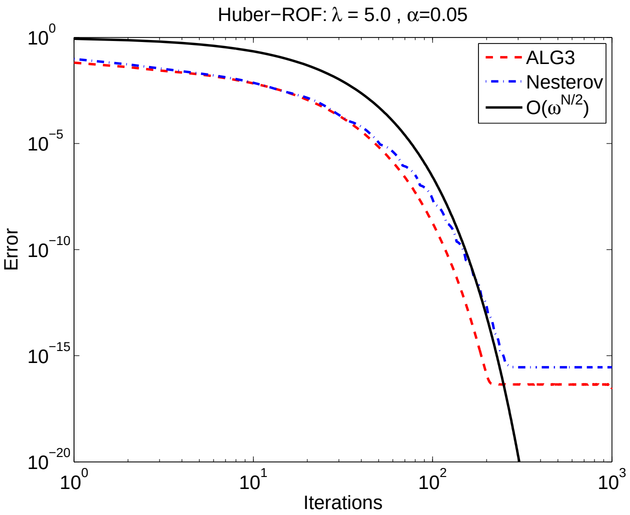
Comparison

(a) Clean (b) Noisy (c) ROF (d) Huber

Comparison between the ROF model and the Huber-ROF model

	$\lambda = 5,\ \alpha = 0.05$
	$\varepsilon = 10^{-15}$
PD	187 (3.85s)
NEST	248 (5.52s)

- PD performs quite well in comparison to a restarted variant of Nesterov's $O(1/N^2)$ method.



Linear convergence of PD and NEST for the Huber-ROF model. Note that after approximately 200 iterations, PD reaches machine precision

General structural sparsity

- It is well known that ℓ_1 norm minimization leads to sparse solutions
- Total variation is probably the most simple example of structural sparsity
- Straight foward to replace ∇ by a better model, e.g. a wavelet or curvelet transform Ψ

$$\min_{u \in X} \|\Psi u\|_1 + \frac{\lambda}{2}\|u - g\|_2^2 \ ,$$

TV versus wavelet denoising

Noisy image

TV versus wavelet denoising

TV denoising

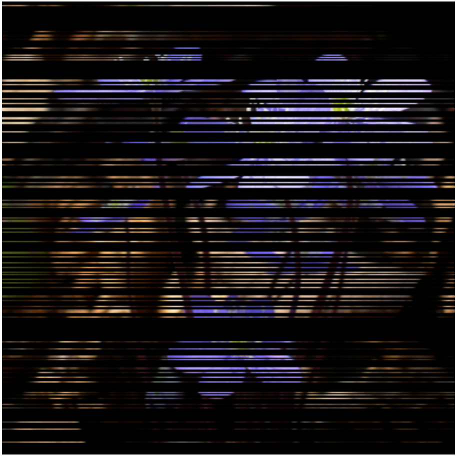
TV versus wavelet denoising

DTCWT denoising

TV versus curvelet inpainting

Original image

TV versus curvelet inpainting



80 % lost lines

TV versus curvelet inpainting

TV inpainting

TV versus curvelet inpainting

Curvelet inpainting

Computing minimal Partitions

- The "continuous" Potts model

$$\min_{E_l} \left\{ \frac{1}{2} \sum_{l=1}^{k} \mathrm{Per}(E_l;\Omega) + \sum_{l=1}^{k} \int_{E_l} f_l(x)\,dx \right\},$$

$$\text{such that } \bigcup_{l=1}^{k} E_l = \Omega, \quad E_s \cap E_t = \emptyset \, \forall s \neq t \,,$$

- Minimizes the total interface length (area) of the partitioning subject to some given external fields f_l
- Convex representation using labeling functions θ_l

$$\min_{\theta} \mathcal{J}(\theta) + \sum_{l=1}^{k} \int_{\Omega} \theta_l f_l dx, \quad \text{s.t. } \theta_l(x) \geq 0, \ \sum_{l=1}^{k} \theta_l(x) = 1, \ \forall x \in \Omega$$

Convex relaxation

Different choices have been proposed

- The most straight-forward relaxation has been proposed in [Zach, Gallup, Frahm, Niethammer '08]

$$\mathcal{J}_1(\theta) = \frac{1}{2} \sum_{l=1}^{k} \int_{\Omega} |D\theta_l|$$

Convex relaxation

Different choices have been proposed

- The most straight-forward relaxation has been proposed in [Zach, Gallup, Frahm, Niethammer '08]

$$\mathcal{J}_1(\theta) = \frac{1}{2} \sum_{l=1}^{k} \int_{\Omega} |D\theta_l|$$

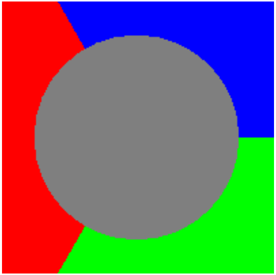
- A tighter relaxation using a local envelope approach has been proposed in [Chambolle, Cremers, Pock '08]

$$\mathcal{J}_2(\theta) = \int_{\Omega} \Psi(D\theta),$$

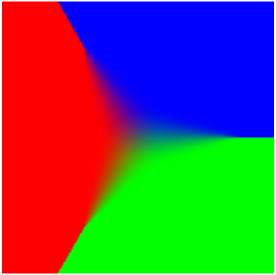
$$\Psi(p) = \sup_{q} \left\{ \sum_{l=1}^{k} \langle p_l, q_m \rangle : |q_l - q_m| \leq 1,\ 1 \leq l < m \leq k \right\}$$

Comparison

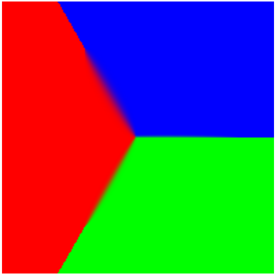
A comparison using the "triple-junction" problem



Input



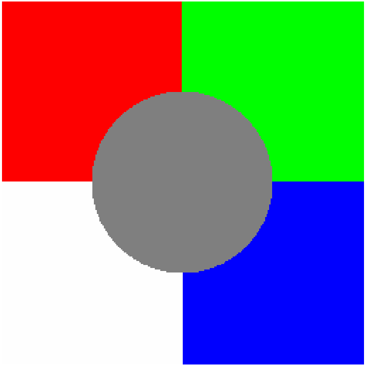
$\mathcal{J}_1$



$\mathcal{J}_2$

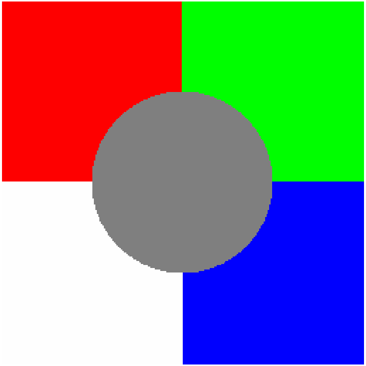
Examples

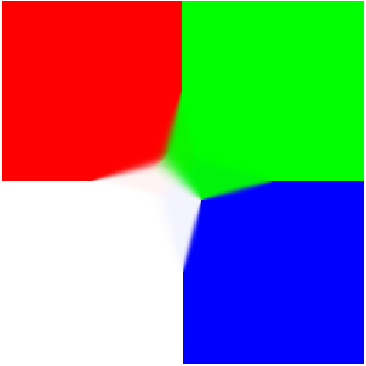
The “4-label” problem



Examples

The “4-label” problem

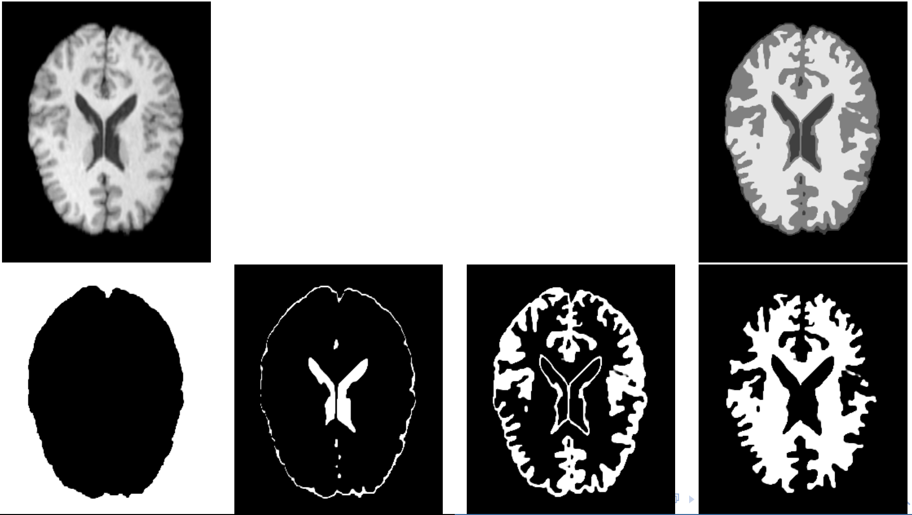




Note that the minimizer is composed of two “triple-junctions”

Examples

White/gray matter segmentation of the brain with $k = 4$ labels

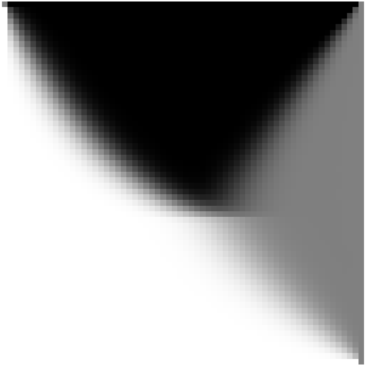


Examples

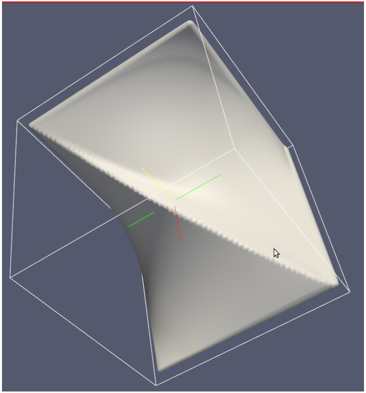
Piecewise constant Mumford-Shah segmentation with $k = 16$ labels

Examples

The “triple-junction” problem in 3D



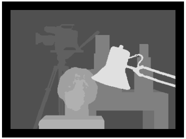
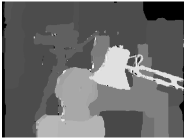
One slice



3D rendering

Examples

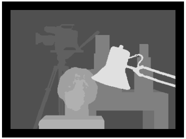
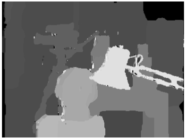
Disparity estimation using the Potts model, simply use a different data term: $f_l(x) = |I_{\mathsf{left}}(x) - I_{\mathsf{right}}(x + \mathsf{disp}_l)|$

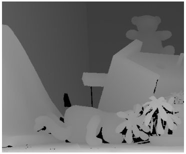
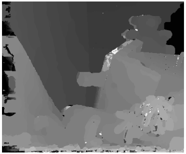
Tsukuba data set, 64 labels, 300 it, 7.7s on a Tesla GPU

Examples

Disparity estimation using the Potts model, simply use a different data term: $f_l(x) = |I_{\text{left}}(x) - I_{\text{right}}(x + \text{disp}_l)|$

Tsukuba data set, 64 labels, 300 it, 7.7s on a Tesla GPU

Teddy data set, 256 labels, 300 it, 175,6s on a Tesla GPU

TV-L^1 optical flow

Optical Flow is an important topic in computer vision [Horn, Schunck, '81]
A typical formulation is given by [Chambolle, Pock '10]

$$\min_{u \in X, v \in Y} \|\nabla v\|_1 + \mu \|\nabla u\|_1 + \lambda \|\rho(u, v)\|_1 \ ,$$

- $u : \Omega \to \mathbb{R}$ models the illumination changes
- $v = (v_1, v_2)^T : \Omega \to \mathbb{R}^2$ is the motion field
- $\rho(u, v) = I_t + (\nabla I)^T (v - v^0) + u$ is the optical flow constraint, explicitly modeling additive illumination changes.

TV-L^1 optical flow

Optical Flow is an important topic in computer vision [Horn, Schunck, '81]
A typical formulation is given by [Chambolle, Pock '10]

$$\min_{u \in X, v \in Y} \|\nabla v\|_1 + \mu \|\nabla u\|_1 + \lambda \|\rho(u, v)\|_1 ,$$

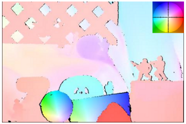
- $u : \Omega \to \mathbb{R}$ models the illumination changes
- $v = (v_1, v_2)^T : \Omega \to \mathbb{R}^2$ is the motion field
- $\rho(u, v) = I_t + (\nabla I)^T (v - v^0) + u$ is the optical flow constraint, explicitly modeling additive illumination changes.

The problem is completely non-smooth.

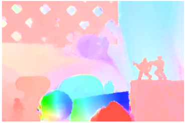
Real-time implementation

- Optical flow constraint is only valid in a small neighborhood of v_0
- Algorithm has to be integrated into a coarse-to-fine / warping framework
- GPU-implementation yields real-time performance (> 30 fps) for 640×480 images using a recent Nvidia graphics card

(a) Input



(b) Ground truth



(c) Estimated motion

(d) Illumination

TV-L^1 optical flow for a sequence of the Middlebury optical flow benchmark

Summary & future work

- First-order primal-dual algorithm for a class of convex optimization problems
- Easy to implement, easy to parallelize
- Matches optimal convergence rates on several subclasses

- Preconditioning of the algorithm for badly scaled problems
- Convergence rates on standard problems, e.g. LP, SOCP
- Further exploit the intrinsic parallelism of variational problems
- Application to machine learning problems, e.g. SVM, LPBoost, ...

Summary & future work

- First-order primal-dual algorithm for a class of convex optimization problems
- Easy to implement, easy to parallelize
- Matches optimal convergence rates on several subclasses

- Preconditioning of the algorithm for badly scaled problems
- Convergence rates on standard problems, e.g. LP, SOCP
- Further exploit the intrinsic parallelism of variational problems
- Application to machine learning problems, e.g. SVM, LPBoost, ...

Here is a hammer: Find the nails!

Thank you for your attention!