

Bastian Goldlücke, University of Konstanz

Variational Methods

Continuous convex models and optimization



THE 35TH ANNUAL CONFERENCE OF THE EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

EUROGRAPHICS 2014
Strasbourg, France

CONFERENCE 7-11 APRIL 2014 STRASBOURG PALAIS DES CONGRÈS





Overview

- 1 Introduction to variational methods
- 2 Convex optimization primer: proximal gradient methods
- 3 Variational inverse problems
- 4 Segmentation and labeling problems
- 5 3D reconstruction
- 6 Summary

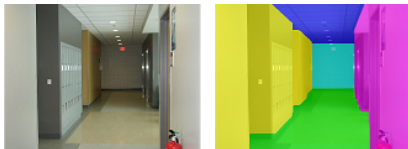




- 1 Introduction to variational methods**
- 2 Convex optimization primer: proximal gradient methods
- 3 Variational inverse problems
- 4 Segmentation and labeling problems
- 5 3D reconstruction
- 6 Summary



Image labeling problems



Segmentation
and Classification



Stereo



Optic flow



3D Reconstruction



- **Unknown object: Vector-valued function**

$$u : \Omega \rightarrow \mathbb{R}^m$$

on a **continuous** domain Ω .

- **Problem solution:** minimizer of an **energy functional**

$$\operatorname{argmin}_{u \in \mathcal{V}} \left\{ \underbrace{J(u)}_{\text{regularizer}} + \underbrace{F(u)}_{\text{data term}} \right\}.$$

on an infinite dimensional (function) space \mathcal{V} .



Advantages:

- Realistic physical models of the world naturally continuous
- Exact modeling of geometric quantities (length, curvature) ...
- No grid bias (rotation invariance)
- Solvers easy to parallelize, fast on GPU

Disadvantages:

- Less flexible than probabilistic graphical models
- Iterative solvers (usually no runtime guarantee)
- Slow on single core



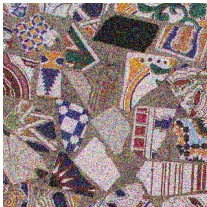
Rudin-Osher-Fatemi (ROF), or TV- \mathcal{L}^2 model (1992)

Given an image f and a smoothness parameter $\lambda > 0$, compute a **denoised** image as the minimizer

$$\operatorname{argmin}_{u \in L^2(\Omega)} \left\{ \|\nabla u\|_1 + \frac{1}{2\lambda} \|u - f\|_2^2 \right\}.$$

- *Can be interpreted as maximum a posteriori (MAP) estimate under assumption of Gaussian noise on f .*
- *The energy is convex, but not differentiable, in particular gradient based methods do not work*





Original

Noisy

Solution



- The TV- \mathcal{L}^2 model

$$\operatorname{argmin}_u \left\{ \|\nabla u\|_1 + \frac{1}{2\lambda} \|u - f\|_2^2 \right\}.$$

is a building block for several more general algorithms. It is therefore important to fully understand it.

- Note that the total variation is not differentiable, so it cannot be exactly minimized by simple gradient descent, although it is convex.
- In this section, we discuss discretization and the most straight-forward exact solver I know of.



In the continuous setting, an image is a function or scalar field $u : \Omega \rightarrow \mathbb{R}$ with $\Omega \subset \mathbb{R}^2$. For simplicity, we assume in this course that the domain is always two-dimensional.

Discretization of an image

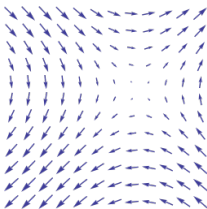
- the domain Ω is a regular grid of size $M \times N$
- an image u is a matrix $u \in \mathbb{R}^{M \times N}$
- the gray value or intensity at a pixel (i, j) is written as $u_{i,j}$



Color images have several components, usually red, green and blue for the primary colors. Thus, they are functions $u : \Omega \rightarrow \mathbb{R}^3$. In the discretization, each component u^1 is represented by its own matrix of intensity values.



A **vector field** is a map $\mathbf{p} : \Omega \rightarrow \mathbb{R}^2$. One can imagine it as an assignment of a little arrow to each point:



Discretization of a vector field

- a vector field \mathbf{p} consists of two components p^1, p^2 , each of which is a matrix of size $M \times N$
- the vector at a pixel (i, j) is written as $\begin{bmatrix} p_{i,j}^1 \\ p_{i,j}^2 \end{bmatrix}$.



In the continuous setting, one of the basic operators acting on a function is the derivative. The vector field of the two partial derivatives of a function $u : \Omega \rightarrow \mathbb{R}$ is called the **gradient** of u ,

$$\nabla u = \begin{bmatrix} \partial_x u \\ \partial_y u \end{bmatrix}.$$

Discretization of the gradient with forward differences

For $i < N$ and $j < M$:

$$(\nabla u)_{i,j} = \begin{bmatrix} u_{i+1,j} - u_{i,j} \\ u_{i,j+1} - u_{i,j} \end{bmatrix}.$$

Neumann boundary conditions $\frac{\partial u}{\partial \mathbf{n}} = 0$ are used for functions, i.e.

$$(\nabla u)_{M,j}^1 = 0 \text{ and } (\nabla u)_{i,N}^2 = 0.$$



In the continuous setting, the divergence of a vector field \mathbf{p} is the scalar field or function $\text{div}(\mathbf{p}) : \Omega \rightarrow \mathbb{R}$ defined as

$$\text{div}(\mathbf{p}) = \partial_x p^1 + \partial_y p^2.$$

Discretization of the divergence with backward differences

Derivatives on vector fields use Dirichlet boundary conditions $\mathbf{p}|_{\partial\Omega} = 0$. Thus,

$$\begin{aligned} (\text{div}(\mathbf{p}))_{i,j} &= \begin{cases} p_{i,j}^1 - p_{i-1,j}^1 & \text{if } 1 < i \leq M \\ p_{i,j}^1 & \text{if } i = 1 \end{cases} \\ &+ \begin{cases} p_{i,j}^2 - p_{i,j-1}^2 & \text{if } 1 < j \leq N \\ p_{i,j}^2 & \text{if } j = 1 \end{cases} \end{aligned}$$



Let $u \in \mathbb{R}^{M \times N}$ be a matrix, then its **total variation** is defined as

$$\text{TV}(u) := \sum_{i=1}^M \sum_{j=1}^N |(\nabla u)_{i,j}|_2,$$

where the gradient is computed with forward differences and Neumann boundary conditions.

Note: Actually computing the discrete TV requires to compute the sum over all elements in a matrix, which is a not fully parallelizable type of operation called a **reduction**.



The Euclidean norm in \mathbb{R}^n can be written for any vector v as

$$|v|_2 = \sup_{p \in \mathbb{R}^n, |p|_2 \leq 1} (v, p),$$

where (v, p) denotes the usual inner product.

We can use this fact to rewrite the total variation of a matrix as follows:

$$\text{TV}(u) = \max_{\mathbf{p} \in K} \sum_{i,j} ((\nabla u)_{i,j}, \mathbf{p}_{i,j}),$$

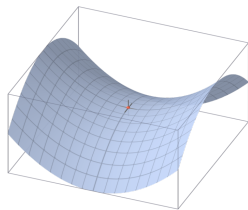
where the (convex) set K consists of all vector fields such that their vectors in each point have length at most one,

$$K := \left\{ \mathbf{p} \in \mathbb{R}^{M \times N} \times \mathbb{R}^{M \times N} : |\mathbf{p}_{i,j}|_2 \leq 1 \text{ for all } i, j \right\}.$$



Instead of finding a minimizer for ROF, we can look for a **saddle point** (u, \mathbf{p}) of

$$\min_{u \in \mathbb{R}^{M \times N}} \max_{\mathbf{p} \in K} \sum_{i,j} ((\nabla u)_{i,j}, \mathbf{p}_{i,j}) + \frac{1}{2\lambda} (u_{i,j} - f_{i,j})^2.$$

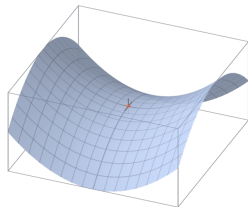


- Note that the energy is now differentiable, so as a simple method to find a saddle point we can iterate gradient descent in u and ascent in \mathbf{p} starting from an initial point (u_0, \mathbf{p}_0) .
- We will formulate a slightly faster method, which is based on the observation that for given \mathbf{p} , one can solve the minimization in u explicitly.



The energy of the saddle point problem is

$$E(u, \mathbf{p}) = \sum_{i,j} ((\nabla u)_{i,j}, \mathbf{p}_{i,j}) + \frac{1}{2\lambda} (u_{i,j} - f_{i,j})^2.$$



Its derivative with respect to \mathbf{p} is just

$$\frac{d}{d\mathbf{p}} E(u, \mathbf{p}) = \nabla u,$$

so a (discrete) gradient ascent step for \mathbf{p} is given by setting

$$\mathbf{p}^{n+1} = \mathbf{p}^n + \tau \nabla u^n,$$

where (\mathbf{p}^n, u^n) is the n th iterate for the saddle point and $\tau > 0$ is a small enough step size (to be determined later).



The allowed range for \mathbf{p} is the set K . After the update step, \mathbf{p}^n might lie outside K , so we have to back-project it onto K . This can be done by projecting each single vector of the field \mathbf{p} back onto the unit disk,

$$\Pi_K(\mathbf{p}^n)_{i,j} = \frac{\mathbf{p}_{i,j}^n}{\max\left(1, \left|\mathbf{p}_{i,j}^n\right|_2\right)}.$$



The reason the finite difference operators introduced in the first lecture were defined in this way is that they need to satisfy

$$\sum_{i,j} ((\nabla u)_{i,j}, \mathbf{p}_{i,j}) = - \sum_{i,j} u_{i,j} (\operatorname{div}(\mathbf{p}))_{i,j}$$

like their continuous counterparts.

This means we can rewrite the saddle point energy as

$$E(u, \mathbf{p}) = \sum_{i,j} -u_{i,j} (\operatorname{div}(\mathbf{p}))_{i,j} + \frac{1}{2\lambda} (u_{i,j} - f_{i,j})^2.$$



The gradient with respect to u is

$$\frac{d}{du}E(u, \mathbf{p}) = -\operatorname{div}(\mathbf{p}) + \frac{1}{\lambda}(u - f),$$

At a minimum for u , this must be zero. This leads to the update equation

$$u^{n+1} = f + \lambda \operatorname{div}(\mathbf{p}^{n+1})$$

which computes the exact solution for u^{n+1} explicitly, given the previously computed \mathbf{p}^{n+1} .



Algorithm to solve ROF (Bermudez-Moreno 1981, Chambolle 2005)

- Given an image f to be denoised and a smoothing parameter $\lambda > 0$.
- Set the step size to $\tau := \frac{1}{2\lambda\|\nabla\|} = \frac{1}{8\lambda}$.
- Start with initial u^0 , \mathbf{p}^0 , where u^0 is a scalar field and \mathbf{p}^0 is a vector field. Initial values can be arbitrary, choose e.g. zero for everything.
- Iterate until convergence:

$$\mathbf{p}^{n+1} = \Pi_K(\mathbf{p}^n + \tau \nabla u^n)$$

$$u^{n+1} = f + \lambda \operatorname{div}(\mathbf{p}^{n+1})$$





Overview

- 1 Introduction to variational methods
- 2 Convex optimization primer: proximal gradient methods**
- 3 Variational inverse problems
- 4 Segmentation and labeling problems
- 5 3D reconstruction
- 6 Summary

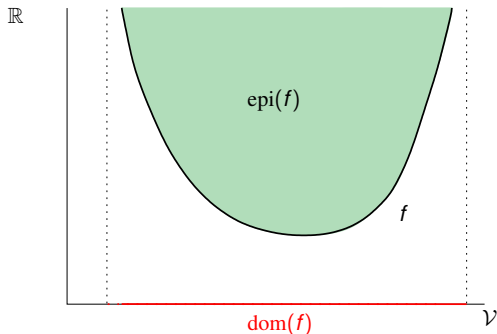


- A Hilbert space \mathcal{V} with
 - inner product (\cdot, \cdot) and
 - norm $\|\cdot\| = \sqrt{(\cdot, \cdot)}$,in this tutorial, this will be $\mathcal{V} = \mathbb{R}^N$ or $\mathcal{V} = L^2(\Omega)$.
- The dual space \mathcal{V}^* - in our setting, you can substitute $\mathcal{V}^* = \mathcal{V}$ if you are not familiar with the concept
- We consider “proper” functionals $f : \mathcal{V} \rightarrow \mathbb{R} \cup \{\infty\}$, i.e. they can take the value infinity, but we exclude the (boring) functional $f = \infty$ for technical reasons.



The **epigraph** $\text{epi}(f)$ of a functional $f : \mathcal{V} \rightarrow \mathbb{R} \cup \{\infty\}$ is the set “above the graph”, i.e.

$$\text{epi}(f) := \{(x, \mu) : x \in \mathcal{V} \text{ and } \mu \geq f(x)\}.$$

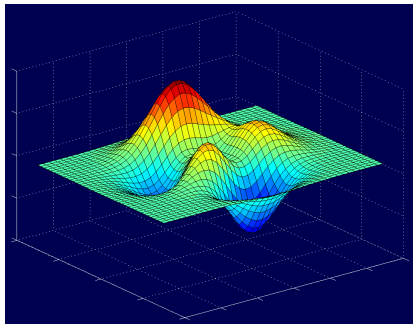


Definition

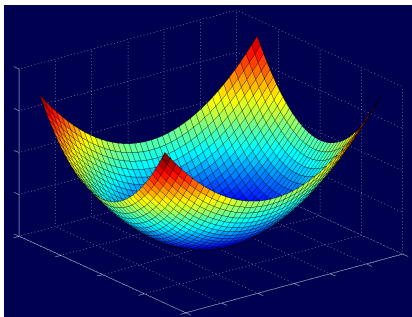
- A functional $f : \mathcal{V} \rightarrow \mathbb{R} \cup \{\infty\}$ is called **convex** if $\text{epi}(f)$ is a convex set.
- The set of all proper and convex functionals on \mathcal{V} is denoted $\text{conv}(\mathcal{V})$.

We choose the geometric definition of a convex function here because it is more intuitive, the usual algebraic property is a simple consequence.





non-convex energy



convex energy

**Convex energies can be globally minimized -
for non-convex energies this is usually impossible.**



Definition

- A vector $\varphi \in \mathcal{V}^*$ is a **subgradient** of f at $x \in \mathcal{V}$ if

$$f(y) \geq f(x) + \langle y - x, \varphi \rangle \text{ for all } y \in \mathcal{V}.$$

- The set of all subgradients of f at x is the **subdifferential** $\partial f(x)$.

Geometrically: the function on the right hand side is an affine function

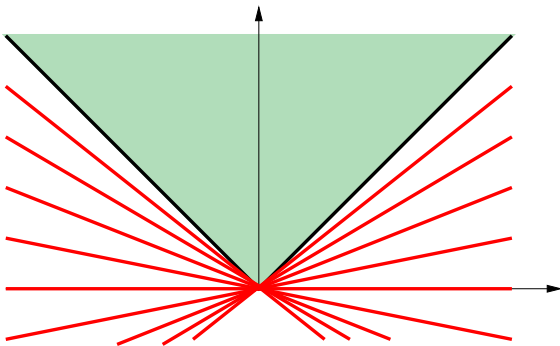
$$h(y) = f(x) + \langle y - x, \varphi \rangle$$

with slope φ , which touches the epigraph of f in x . The condition says that h needs to lie below f .



Example: the subdifferential of $f : x \mapsto |x|$ in 0 is

$$\partial f(0) = [-1, 1].$$



The subdifferential is a generalization of the Fréchet derivative (or the gradient in finite dimension), in the following sense.

Theorem

Let $f \in \text{conv}(\mathcal{V})$ be Fréchet differentiable at $x \in \mathcal{V}$. Then

$$\partial f(x) = \{df(x)\}.$$

The proof of the theorem is surprisingly involved - it requires to relate the subdifferential to one-sided directional derivatives. We will not explore this here, see e.g. [Rockafeller 1970].



Theorem

Let $f \in \text{conv}(\mathcal{V})$. Then \hat{x} is a **global minimum** of f if and only if

$$0 \in \partial f(\hat{x}).$$

In particular, a convex functional does not have minima which are local but not global.

Geometrically intuitive, but the proof is also surprisingly simple. Rewriting the subgradient definition, one sees that $0 \in \partial f(\hat{x})$ just means that

$$f(y) - \langle y, 0 \rangle \geq f(\hat{x}) - \langle \hat{x}, 0 \rangle \text{ for all } y \in \mathcal{V}.$$

This already implies the equivalence.



Proximity operator

- The **proximity operator** for a closed convex functional H and step size parameter $\tau > 0$ is defined as

$$\text{prox}_{\tau H}(x) = \underset{y}{\operatorname{argmin}} \left\{ \frac{1}{2\tau} \|y - x\|^2 + H(y) \right\}.$$

- At the minimizer $\hat{x} = \text{prox}_{\tau H}(x)$,

$$\begin{aligned} 0 &\in \frac{1}{\tau}(\hat{x} - x) + \partial H(\hat{x}) \\ \Leftrightarrow \hat{x} &\in x - \tau \partial H(\hat{x}). \end{aligned}$$

The proximity operator computes an implicit subgradient descent for the convex functional H with step size τ .



- Let $H = TV$, then

$$\text{prox}_{\lambda TV}(x) = \underset{y}{\operatorname{argmin}} \left\{ TV(y) + \frac{1}{2\tau} \|x - y\|^2 \right\}$$

is an instance of ROF-denoising.

- Solve e.g. with Bermudez-Moreno.



Examples for the prox operator (2)

- Let $\mathcal{C} \subset \mathcal{V}$ be a convex set, and

$$\delta_{\mathcal{C}}(x) := \begin{cases} 0 & \text{if } x \in \mathcal{C}, \\ \infty & \text{otherwise} \end{cases}$$

be its **indicator function**.

- Then

$$\begin{aligned} \text{prox}_{\delta_{\mathcal{C}}}(x) &= \operatorname{argmin}_{y \in \mathcal{V}} \left\{ \frac{1}{2\tau} \|y - x\|^2 + \delta_{\mathcal{C}}(y) \right\} \\ &= \operatorname{argmin}_{y \in \mathcal{C}} \left\{ \|y - x\|^2 \right\} \end{aligned}$$

is the projection onto \mathcal{C} .

- Adding an indicator function to a functional is equivalent to an optimization constraint.



Examples for the prox operator (3)

- Let $H = \|\cdot\|_1$ (sparsity inducing norm), e.g. interesting in compressive sensing or dictionary learning, next talk.

- Then (component-wise)

$$\text{prox}_{\tau\|\cdot\|_1}(x)_i = (|x_i| - \tau)\text{sgn}(x_i),$$

which is called the **shrinkage operator**.



Applicability: Minimization problems of the form

$$\operatorname{argmin}_{u \in \mathcal{V}} \{G(u) + F(u)\}$$

such that

- G is convex
- F is convex and differentiable
- dF is Lipschitz continuous, i.e. there exist $L > 0$ such that

$$\|dF(u) - dF(v)\| \leq L \|u - v\| \text{ for all } u, v \in \mathcal{V}.$$

- The proximation for G is (comparatively) easy to evaluate.



Iterative shrinkage and thresholding (ISTA)

Algorithm: Initialize with $u_0 = 0$,
then compute alternating “forward-backward steps”:

- gradient descent in F :

$$v_{n+1} = u_n - \frac{1}{L} dF(u_n).$$

- implicit subgradient descent in G :

$$u_{n+1} = \text{prox}_{\frac{1}{L}G}(v_{n+1}).$$

only $O(1/n)$ convergence (slow).

Bruck 1977, Passty 1979



Algorithm: Initialize with $u_0 = 0, \bar{u}_0 = 0, \mathbb{R} \ni t_0 = 1$, then iterate

- gradient descent in F :

$$v_{n+1} = \bar{u}_n - \frac{1}{L} dF(\bar{u}_n).$$

- implicit subgradient descent in G :

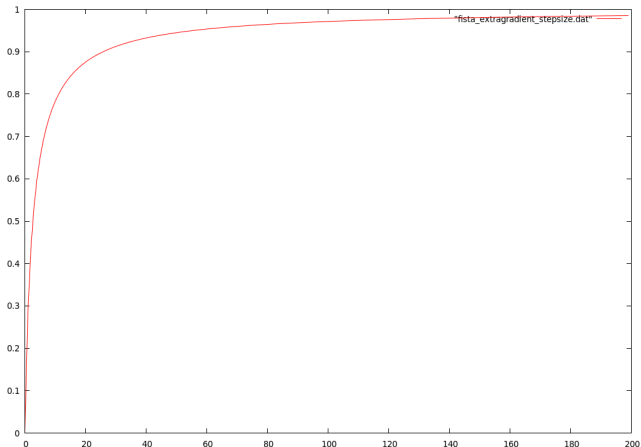
$$u_{n+1} = \text{prox}_{\frac{1}{L}G}(v_{n+1}).$$

- extragradient step:

$$t_{n+1} = \frac{1}{2}(1 + \sqrt{1 + 4t_n^2}),$$
$$\bar{u}_{n+1} = u_n + \frac{t_n - 1}{t_{n+1}}(u_{n+1} - u_n).$$



FISTA extragradient step size development



- Leads to $O(1/n^2)$ algorithm (optimal for a first-order method)
- Good general-purpose solver, easy to implement (in particular if proximation is already available)



Let $\mathcal{C} \subset \mathcal{V}$ be a convex constraint set, i.e. the solution shall be restricted to lie in \mathcal{C} . Thus, we want to solve

$$\operatorname{argmin}_{x \in \mathcal{V}} \{F(x)\} \text{ subject to } x \in \mathcal{C},$$

which could be e.g. a constrained least squares problem as in the first talk. This is equivalent so solving

$$\operatorname{argmin}_{x \in \mathcal{V}} \{F(x) + \delta_{\mathcal{C}}(x)\}.$$

When applying FISTA, $\delta_{\mathcal{C}}$ is the non-differentiable part taken care of via the prox operator, which equals the projection onto \mathcal{C} .



Many other proximal gradient methods ...

- Alternating projection (projection onto convex sets)
- Douglas-Rachford splitting, alternating direction method of multipliers ADMM (popular e.g. in learning)
- Split-Bregman (basis pursuit problems, i.e. constrained L^1 -minimization)
- Chambolle-Pock (swiss army knife if both F and G are non-differentiable, but convex with simple prox-operators).

All of these use the same basic building blocks
as the ones shown in the previous section.





Overview

- 1 Introduction to variational methods
- 2 Convex optimization primer: proximal gradient methods
- 3 Variational inverse problems**
- 4 Segmentation and labeling problems
- 5 3D reconstruction
- 6 Summary



*The convolution $b * u$ with a kernel b of total mass 1 can be interpreted as a blurring operation.*

Example: Gaussian blur (isotropic)



Example: Motion blur for diagonal motion (anisotropic)



Inverse problem: We observe the image f , which was created from an unknown original via

$$f = b * u + \text{Gaussian noise.}$$

More general, $f = Au + \text{noise}$ (linear inverse problem).

TV deblurring model

Recover a deblurred image u as the minimizer of

$$\operatorname{argmin}_u \left\{ \|\nabla u\|_1 + \frac{1}{2\lambda} \|b * u - f\|_2^2 \right\},$$

which is again the MAP estimate for Gaussian noise with TV prior.



*Original**Blurred + Noisy**Solution*

- The differentiable part is

$$F(u) = \frac{1}{2\lambda} \|Au - f\|_2^2,$$

which has functional derivative

$$dF(u) = \frac{1}{\lambda} A^T (Au - f),$$

which is Lipschitz-continuous with $L = \frac{1}{\lambda} \|A^T A\|_2$.

- The non-differentiable convex part is the regularizer, e.g. TV, with the proximation given by solving an instance of ROF.



Inpainting problem

- Restoration problem: recover missing regions of an image
- Input:
 - damaged region $\Gamma \subset \Omega$
 - partial image $f : \Omega \setminus \Gamma \rightarrow \mathbb{R}$
- TV inpainting: Solve

$$\min_u \|\nabla u\|_1 \text{ such that } u = f \text{ on } \Omega \setminus \Gamma.$$



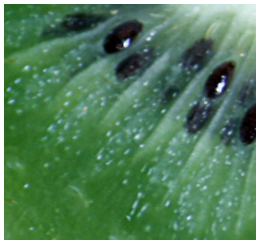
Damaged image f



Recovered image u



TV inpainting results (“textured” images)



Original



Damaged

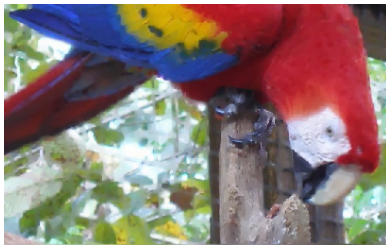
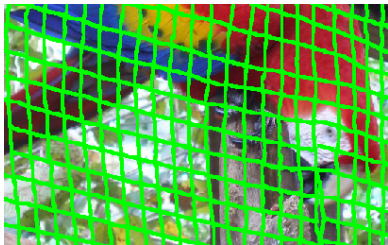


Inpainted

- TV inpainting is unconvincing for highly textured images if the missing regions are larger. The reason is that no structure is inferred from surrounding regions, and only boundary values of Γ are taken into account.
- A better variational model for inpainting can be found e.g. in papers on **non-local TV** by Osher et al., or check methods based on **dictionary learning**.



Once we have an inpainting algorithm, we can employ it to remove unwanted regions in an image by marking them as damaged.



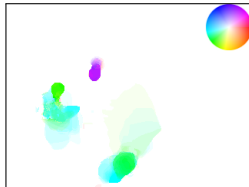
Uncaging a bird



Input

 I_t at time t  I_{t+1} at time $t + 1$

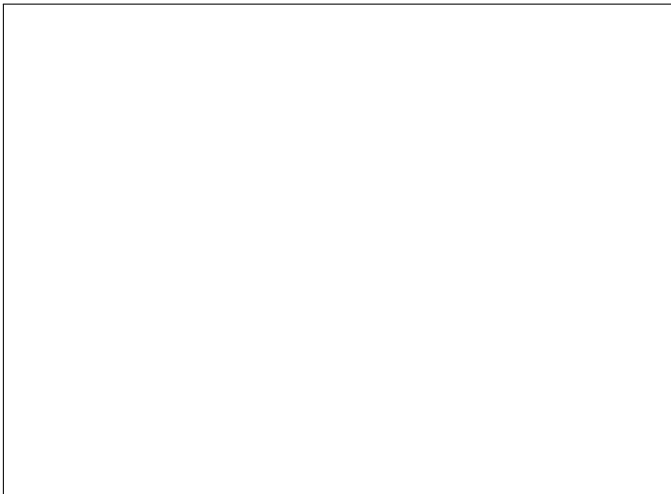
Unknown

flow field $u : \Omega \rightarrow \mathbb{R}^2$ TV- \mathcal{L}^1 optical flow (Zach et al. DAGM 2007)

$$\operatorname{argmin}_u \left\{ \underbrace{\operatorname{TV}(u_1) + \operatorname{TV}(u_2)}_{\text{regularizer } J(u)} + \frac{1}{2\lambda} \int_{\Omega} \underbrace{|I_{t+1}(x + u(x)) - I_t(x)|_1}_{\text{data term } \rho(u(x), x)} dx \right\}.$$



Example: real-time optic flow



Idea: introduce auxiliary variable v

- Decouples regularizer from the data term.
- Let $\theta > 0$ be a coupling parameter, define family of energies in the variables u, v ,

$$E_\theta(u, v) = \mathcal{J}(u) + \frac{1}{2\theta} \|u - v\|_2^2 + \int_{\Omega} \rho(x, v(x)) dx.$$

- For $\theta \rightarrow 0$, the quadratic term forces u to be close to v , so the solution of E_θ approximates the solution of E .



Algorithm

Set $\theta > 0$, start with initial u_0, v_0 . Then iterate the following steps until convergence.

- Solve an instance of TV- \mathcal{L}^2 :

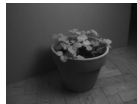
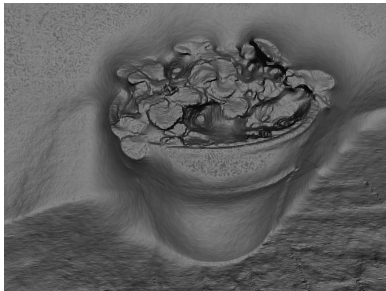
$$u_{k+1} = \operatorname{argmin}_u E_\theta(u, v_k).$$

- Solve the point-wise problem

$$v_{k+1} = \operatorname{argmin}_v E_\theta(u_{k+1}, v).$$



same technique (just reduced target dimension):
real-time disparity (depth) maps

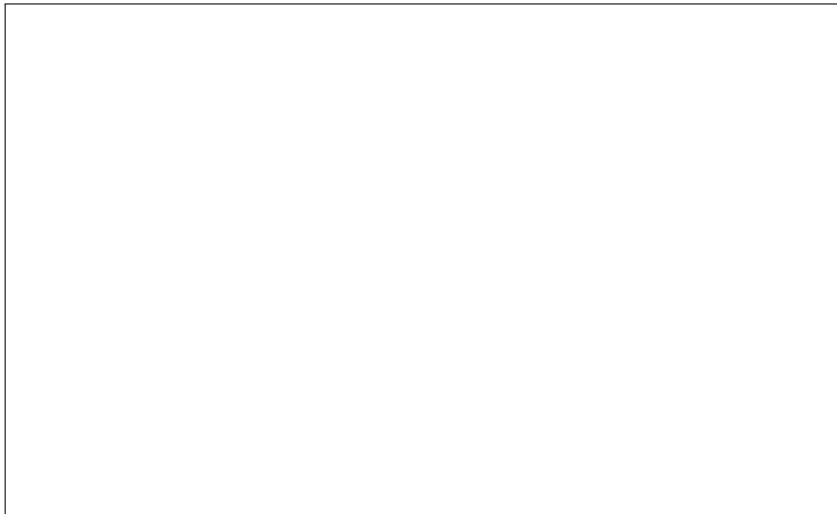


Stuehmer et al. DAGM 2010





Real-time scene flow



Wedel et al. ECCV 2008





Overview

- 1 Introduction to variational methods
- 2 Convex optimization primer: proximal gradient methods
- 3 Variational inverse problems
- 4 Segmentation and labeling problems**
- 5 3D reconstruction
- 6 Summary



Given:

- An *observed image* f on a domain Ω , from which one computes (e.g. via a trained classifier)
- a *foreground probability distribution*

$$P_{fg}(x) = \text{Probability that the point } x \text{ is in the foreground}$$

- A *background probability distribution*

$$P_{bg}(x) = \text{Probability that the point } x \text{ is in the background}$$

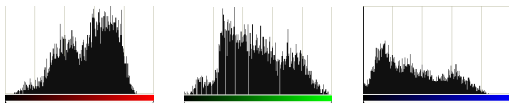
Wanted:

- The *MAP estimate for a binary segmentation* $u : \Omega \rightarrow \{0, 1\}$, where $u(x) = 1$ means a point is in the foreground, and $u(x) = 0$ means it is in the background:

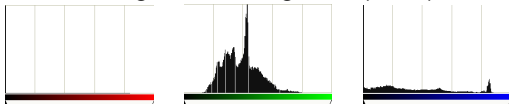
$$\operatorname{argmax}_{u: \Omega \rightarrow \{0, 1\}} P(u | f) = \operatorname{argmax}_{u: \Omega \rightarrow \{0, 1\}} P(f | u)P(u).$$



Example: histograms of user input (naive, illustration only)



Foreground histograms (RGB)



Background histograms (RGB)

$$P_{fg}(x) = \frac{\text{number of marked foreground pixels with color } f(x)}{\text{total number of marked foreground pixels}}$$

$$P_{bg}(x) = \frac{\text{number of marked background pixels with color } f(x)}{\text{total number of marked background pixels}}$$



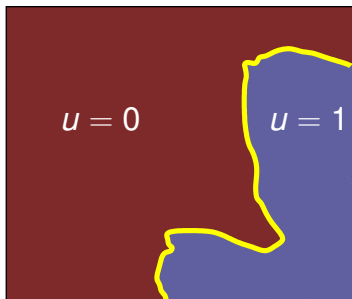
Under the above assumptions, the MAP estimate for the segmentation is

$$\operatorname{argmin}_{u:\Omega\rightarrow\{0,1\}} \left\{ J(u) + \int_{\Omega} u \cdot \log \left[\frac{p_{bg}}{p_{fg}} \right] dx \right\}.$$

Here, $J(u)$ is the **prior**. A typical prior is the length of the interface.



The binary case



Length of interface equals total variation of u ,

$$\text{TV}(u) = \int_{\Omega} |Du|$$

Binary segmentation problem

$$\operatorname{argmin}_{u: \Omega \rightarrow \{0,1\}} \text{TV}(u) + \int_{\Omega} cu \, dx,$$

with local assignment costs c .

Space of binary functions $u : \Omega \rightarrow \{0, 1\}$ not convex
but: globally optimal solution possible by
relaxation to $u : \Omega \rightarrow [0, 1]$ and subsequent thresholding.

Chan, Esedoglu and Nikolova 2006



Straightforward:

- Since $F(u) = (c, u)$, the derivative is constant,

$$dF(u) = a.$$

- It is Lipschitz-continuous with arbitrary Lipschitz constant $L > 0$, one can e.g. just choose $L = 1$.
- Thus, the gradient descent step in F just subtracts c from the solution.
- Proximation in the TV-regularizer is an instance of ROF.



Segmentation example



Input image



User marks



$\log(p_{bg}/p_{fg})$

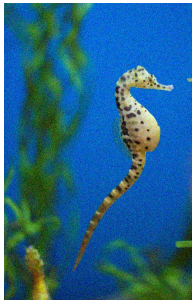


Result

Simple model works pretty well with easy, clean input ...



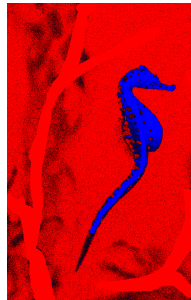
Segmentation example: noisy input



Input image



User marks



$\log(p_{bg}/p_{fg})$

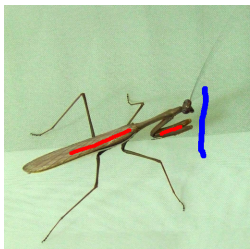


Result

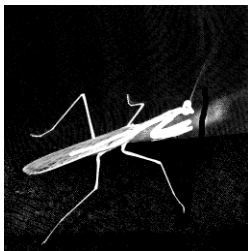
... but is not very robust against noise
(better classifiers requires, i.e. random forests)



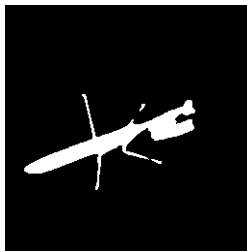
Segmentation example: elongated structures



Input with User marks



$\log(p_{bg}/p_{fg})$ (normalized)



TV (best result)


Length does not work well with elongated structures
(curvature regularity is better in that case, but complex)


Goldluecke and Cremers ICCV 2011





Indicator function $u_\gamma : \Omega \rightarrow \{0, 1\}$ assigned to each label γ :



 $u_1 = 1$, all others zero

 $u_2 = 1$, all others zero

 $u_3 = 1$, all others zero

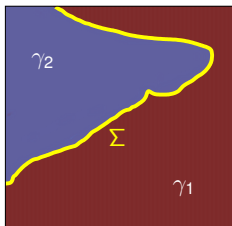
 $u_4 = 1$, all others zero

$\sum_\gamma u_\gamma$ must be one !

Problem relaxation

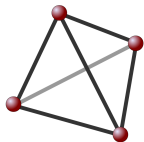
$$\operatorname{argmin}_{u_\gamma : \Omega \rightarrow [0, 1], \sum_\gamma u_\gamma = 1} \left[\mathcal{J}(\mathbf{u}) + \sum_\gamma \int_\Omega c_\gamma u_\gamma \, dx \right],$$





The regularization penalty is proportional to the **label distance** times the **length of the interface**.

In this example $d(\gamma_1, \gamma_2) \cdot L(\Sigma)$

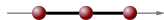


Euclidean representation of the label distance:

- Each label γ is *represented* by a point $a_\gamma \in \mathbb{R}^k$.
- Label distance $d(\gamma, \mu) = |a_\gamma - a_\mu|_2$.

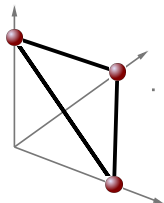


Ordered Labels



$$a_\gamma = \gamma \in \mathbb{R}$$

- Example: depth reconstruction
- Can be solved globally with functional lifting [Pock, Schönemann, Graber, Bischof, Cremers '08]
- Continuous version of [Ishikawa '03]

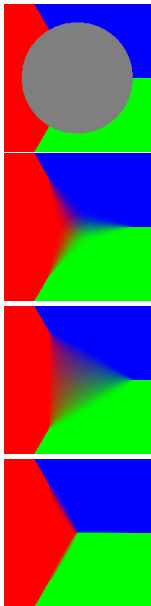


$$a_\gamma = e_\gamma \in \mathbb{R}^N$$

Potts model

- Example: segmentation
- No globally optimal solution possible if $N > 2$
- Continuous version of [Potts '52]





A comparison using the “triple-junction” problem

Zach, Gallup, Frahm, Niethammer '08

$$\mathcal{J}_1(\mathbf{u}) = \frac{1}{2} \sum_{\gamma} \int_{\Omega} |Du_{\gamma}|$$

Lellmann, Kappes, Yuan, Becker, Schnörr '08

$$\mathcal{J}_2(\mathbf{u}) = \int_{\Omega} \sqrt{\sum_{\gamma} \|Du_{\gamma}\|_A^2}, \quad \|v\|_A = \sqrt{v^T A^T A v}$$

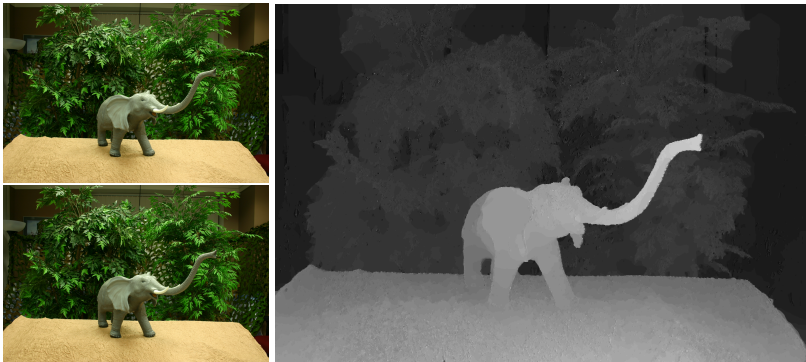
Chambolle, Cremers, Pock '08

$$\mathcal{J}_3(\mathbf{u}) = \int_{\Omega} \Psi(D\mathbf{u}), \quad \Psi(\mathbf{q}) = \sup_{\mathbf{p}} \{ \langle \mathbf{q}, \mathbf{p} \rangle : |p_{\gamma} - p_{\mu}| \leq d_{\gamma\mu} \}$$



Results: indoor stereo pair

- Stereo assignment cost $c_\gamma(x) = |I_{\text{left}}(x) - I_{\text{right}}(x + \gamma)|$
- Linear discontinuity penalty \Rightarrow globally optimal solution.



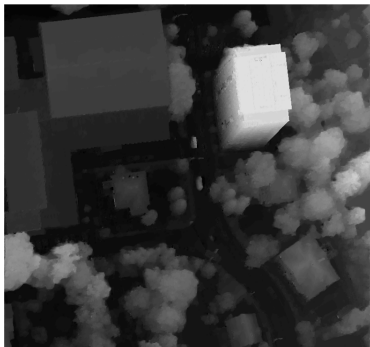
Images from UCSD lightfield repository



Results: aerial images (1400 x 1500 pixels)



One of two input images
(Courtesy of Microsoft Graz)



Depth reconstruction



Potts segmentation with $k = 10$ labels



Potts segmentation with $k = 16$ labels





*Labeling regions should have certain spatial relationships,
i.e. heaven is always above ground.*



Stekalovskiy and Cremers, ICCV 2011

The labeling penalty may depend also on the **normal \mathbf{n} of the interface** between two regions,

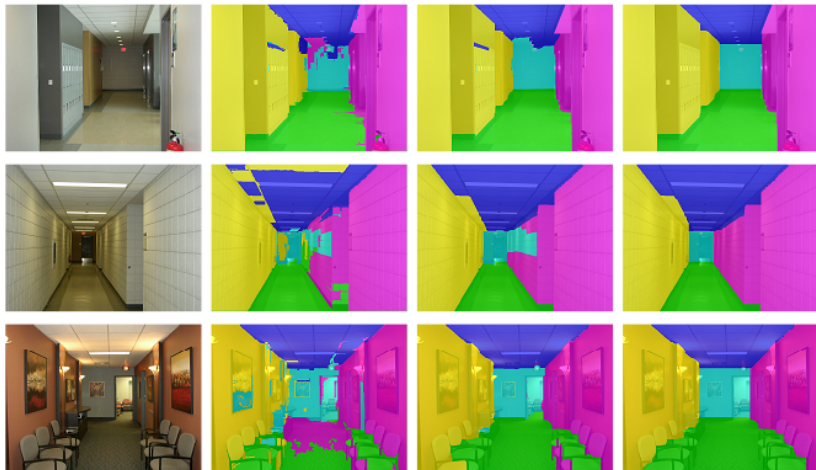
$$d : \Gamma \times \Gamma \times \mathbb{S} \rightarrow \mathbb{R}^+.$$

New direction-aware regularizer:

$$\mathcal{J}(\mathbf{u}) = \sup_{\mathbf{p} \in \mathcal{C}} \sum_{\gamma} \int_{\Omega} \langle \mathbf{p}_{\gamma}, \nabla u_{\gamma} \rangle dx,$$

$$\text{with } \mathcal{C} = \{(\mathbf{p}_{\gamma} : \Omega \rightarrow \mathbb{R}^n) : \langle \mathbf{p}_{\mu} - \mathbf{p}_{\gamma}, \mathbf{n} \rangle \leq d(\gamma, \mu, \mathbf{n}) \quad \forall \gamma, \mu, \mathbf{n}\}.$$





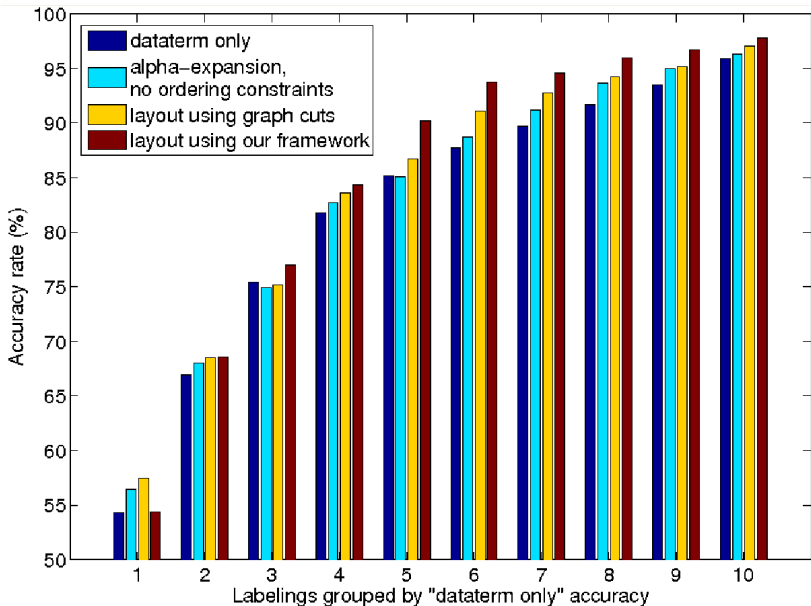
Input

Data term

Potts

Ordering







Overview

- 1 Introduction to variational methods
- 2 Convex optimization primer: proximal gradient methods
- 3 Variational inverse problems
- 4 Segmentation and labeling problems
- 5 3D reconstruction**
- 6 Summary



Given n images $\mathcal{I}_i : \Omega_i \rightarrow \mathbb{R}^3$
with projections $\pi_i : \mathbb{R}^3 \rightarrow \mathbb{R}^2$



Given n images $\mathcal{I}_i : \Omega_i \rightarrow \mathbb{R}^3$
with projections $\pi_i : \mathbb{R}^3 \rightarrow \mathbb{R}^2$



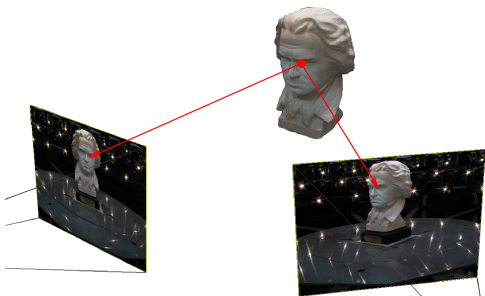
Find surface $\Sigma \subset \mathbb{R}^3$ with texture $T : \Sigma \rightarrow \mathbb{R}^3$ which optimally matches the input images.



Classical variational formulation (Faugeras and Keriven, 1998)

Find a surface $\Sigma \subset \mathbb{R}^3$ which minimizes the photo-consistency error,

$$\operatorname{argmin}_{\Sigma} \int_{\Sigma} \rho(s) ds.$$



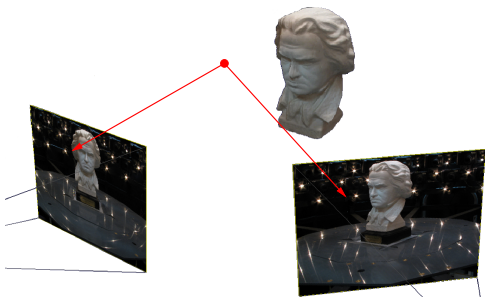
- Point **on surface**
- projections look **similar** in different views
- **small value** of ρ



Classical variational formulation (Faugeras and Keriven, 1998)

Find a surface $\Sigma \subset \mathbb{R}^3$ which minimizes the photo-consistency error,

$$\operatorname{argmin}_{\Sigma} \int_{\Sigma} \rho(s) ds.$$



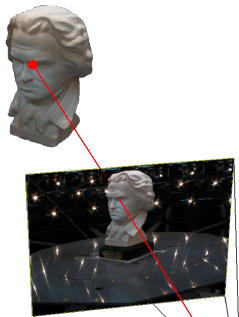
- Point **not on surface**
- projections look **different** in different views
- **large value** of ρ



Convex functional minimizes photo-consistency error:

$$\operatorname{argmin}_{u:\Omega\rightarrow\{0,1\}} \int_{\Omega} \rho |Du|$$

Silhouette constraints to avoid constant solutions



A ray through the **silhouette** must **intersect** the surface

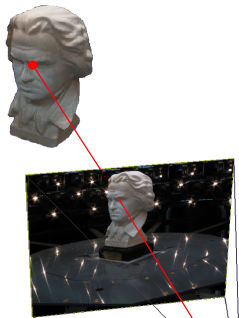
Kolev, Klodt, Brox, Cremers IJCV'09



Convex functional minimizes photo-consistency error:

$$\operatorname{argmin}_{u:\Omega\rightarrow\{0,1\}} \int_{\Omega} \rho |Du|$$

Silhouette constraints to avoid constant solutions



A ray through the **background** must **miss** the surface

Kolev, Klodt, Brox, Cremers IJCV'09





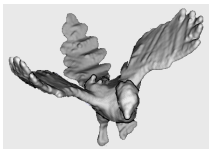
Kolev and Matiouk, Akademisches Kunstmuseum Bonn 2010



Given



n images $\mathcal{I}_i : \Omega_i \rightarrow \mathbb{R}$
 projections $\pi_i : \mathbb{R}^3 \rightarrow \mathbb{R}^2$



approximate surface Σ
 (assumed Lambertian)

Find



high-res color texture map

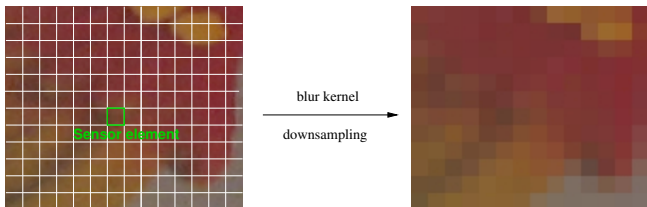
$$T : \Sigma \rightarrow \mathbb{R}^3$$

and accurate geometry

Idea:

Use information from multiple cameras for superresolution.





- Each sensor element samples incoming light over its area
- Sampling modeled by blur kernel b
- Leads to image formation model

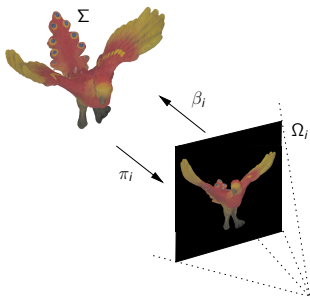
$$\mathcal{I}_{\text{observed}} (\text{low-res}) = b * \mathcal{I}_{\text{incoming}} (\text{high-res})$$



Data term: squared difference between input images and downsampled high-resolution rendering

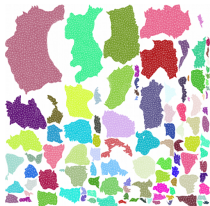
$$E(T) := \sum_{i=1}^n \int (\underbrace{b * (T \circ \beta_i)}_{\text{Subsampled hi-res rendering}} - \underbrace{\mathcal{I}_i}_{\text{Input image}})^2 dx$$

Regularizer: total variation on the surface.

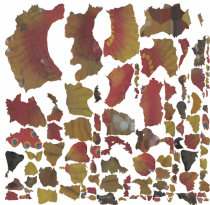


Goldluecke and Cremers,
ICCV 2009





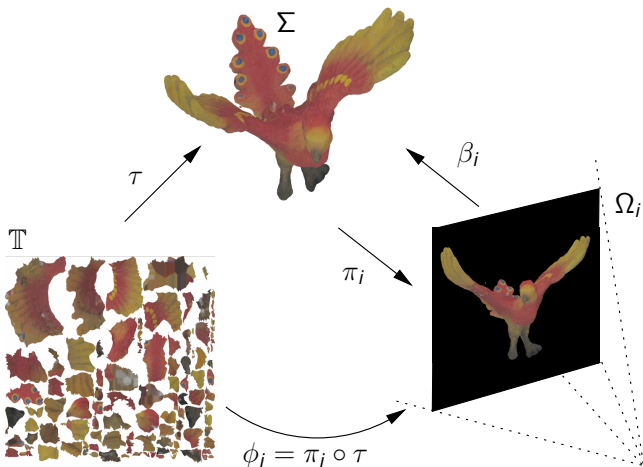
$$\mathbb{T} \xrightarrow{\tau} \Sigma$$



Conformal atlas: transforms energy to 2D texture space



$$\frac{1}{\lambda} \operatorname{div} \left(\sqrt{\lambda} \frac{\nabla \mathcal{T}}{\|\nabla \mathcal{T}\|} \right) + \sum_{i=1}^n \frac{v_i}{\sigma} ((\mathcal{J}_i \mathcal{E}_i) \circ \phi_i) = 0 \text{ on } \mathbb{T}$$





Bird



Beethoven

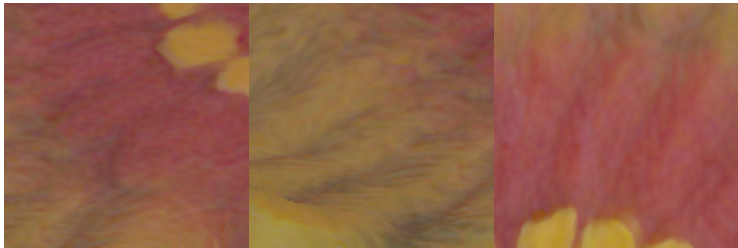


Bunny

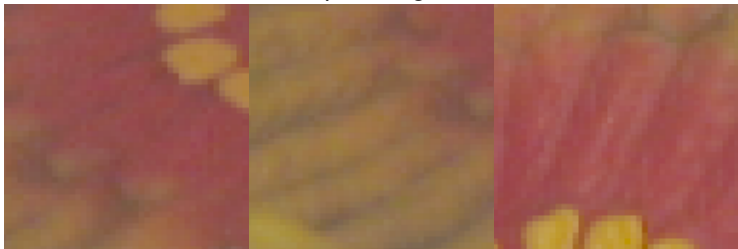
- 30 cameras, input image resolution 768×576
- Initial geometry: [Kolev and Cremers, ECCV 2008](#)
- Texture resolution 2048×2048



Rendered model



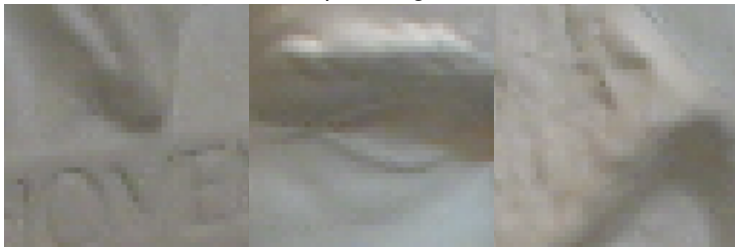
Input image



Rendered model



Input image



Additional dependance on **displacement map** $D : \mathbb{T} \rightarrow \mathbb{R}$,

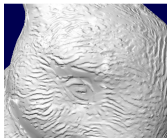
$$E(T, D) := \sum_{i=1}^n \int_{\hat{S}_i} \left(b * (T \circ \beta_i^D) - \mathcal{I}_i \right)^2 dx + E_{\text{tv}}(T, D).$$

- In T : energy is convex
- In D : multilabel problem with convex regularizer, **global optimization in D possible**

Geometry



Input



Estimated displacement

Estimated Texture



Without displacement



With displacement

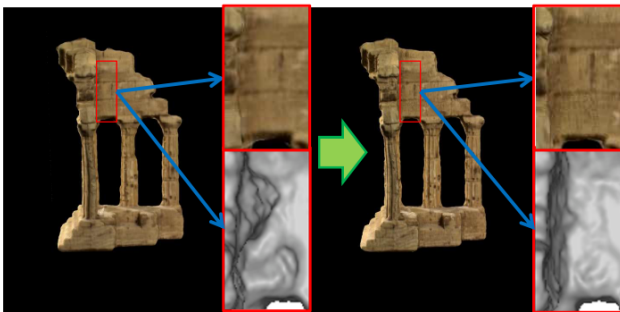
Goldlücke and Cremers DAGM 2009



Idea: assume the **projection parameters** π are **unknowns** in the superresolution energy,

$$E(T, \pi) := \sum_{i=1}^N \int_{S_i} \|b * (T \circ \beta_i) - \mathcal{I}_i\| dx.$$

Optimize alternately for texture and projection. In a way, this can be thought of as a continuous version of bundle adjustment.



Goldluecke, Aubry, Kolev, Cremers, IJCV 2013.





Textured bunny model





Overview

- 1 Introduction to variational methods
- 2 Convex optimization primer: proximal gradient methods
- 3 Variational inverse problems
- 4 Segmentation and labeling problems
- 5 3D reconstruction
- 6 Summary**





Summary

- Variational methods: a powerful tool for realistic modeling
- Non-convex energies: gradient descent, local minimum, usually requires good initialization
- Convex energies: no local minima, powerful optimization tools (e.g. proximal gradient methods)
- For $\operatorname{argmin}_u \{F(u) + G(u)\}$ with convex G and differentiable and convex F , FISTA is a good general-purpose method with optimal convergence rate.
- If F is only convex, check out e.g. [Chambolle-Pock 2012] or [Nesterov 2005].
- Numerous applications: variational inverse problems, segmentation and labeling, 3D reconstruction ...

