MICCAI 2010 Tutorial

Intensity-based Deformable Registration

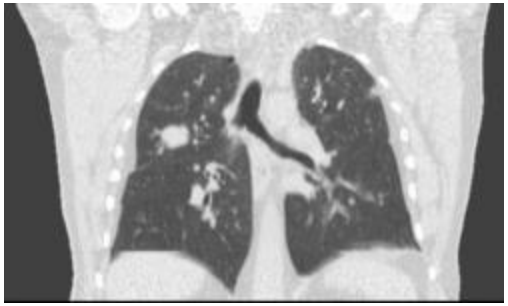
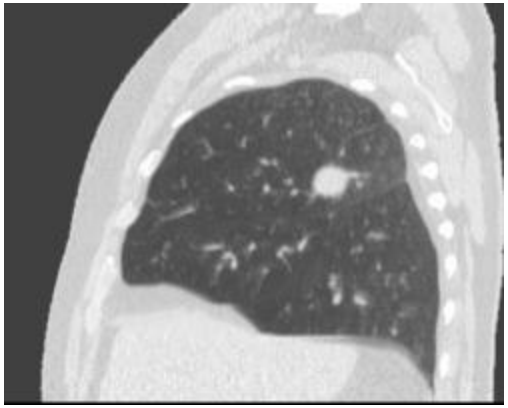
24 September 2010, Beijing, China

Overview of Intensity-based Deformable Registration

Darko Zikic

Computer Aided Medical Procedures (CAMP), TU München, Germany

Reasons for Deformable Registration

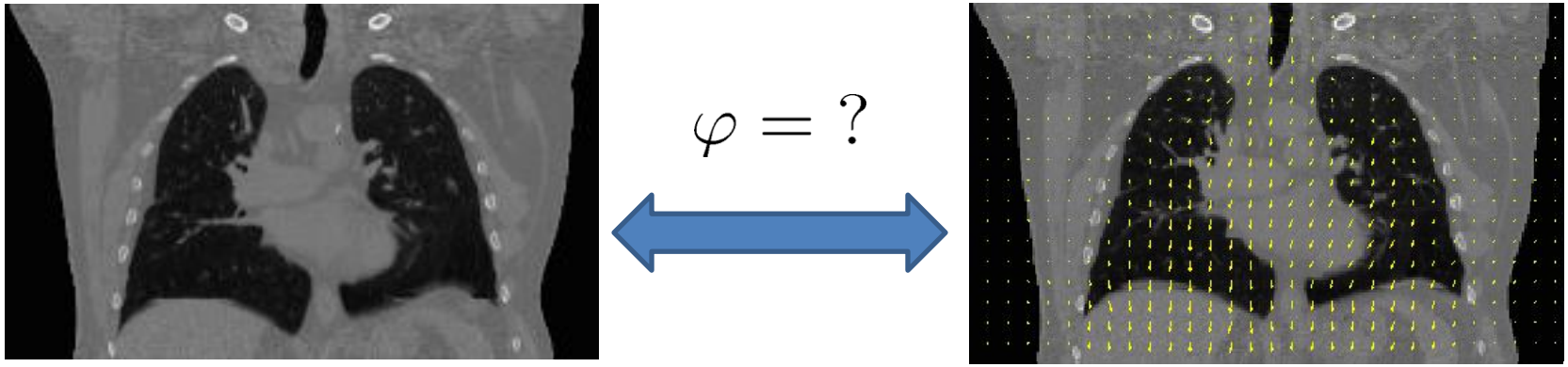




- Patients move *(alignment of temporal series)*
- Patients change *(pre- / post-treatment images)*
- Patients differ *(creation of atlases)*

Animated images from the webpage of
The POPI-model, a Point-validated Pixel-based Breathing Thorax Model
http://www.creatis.insa-lyon.fr/rio/popi-model
See also:
Vandemeulebroucke, J., Sarrut, D. and Clarysse, P.. *The POPI-model, a point-validated pixel-based breathing thorax model.* In *XVth International Conference on the Use of Computers in Radiation Therapy (ICCR)*, 2007.

Deformable Registration

Computing a non-linear spatial transformation between ***corresponding*** structures in two images

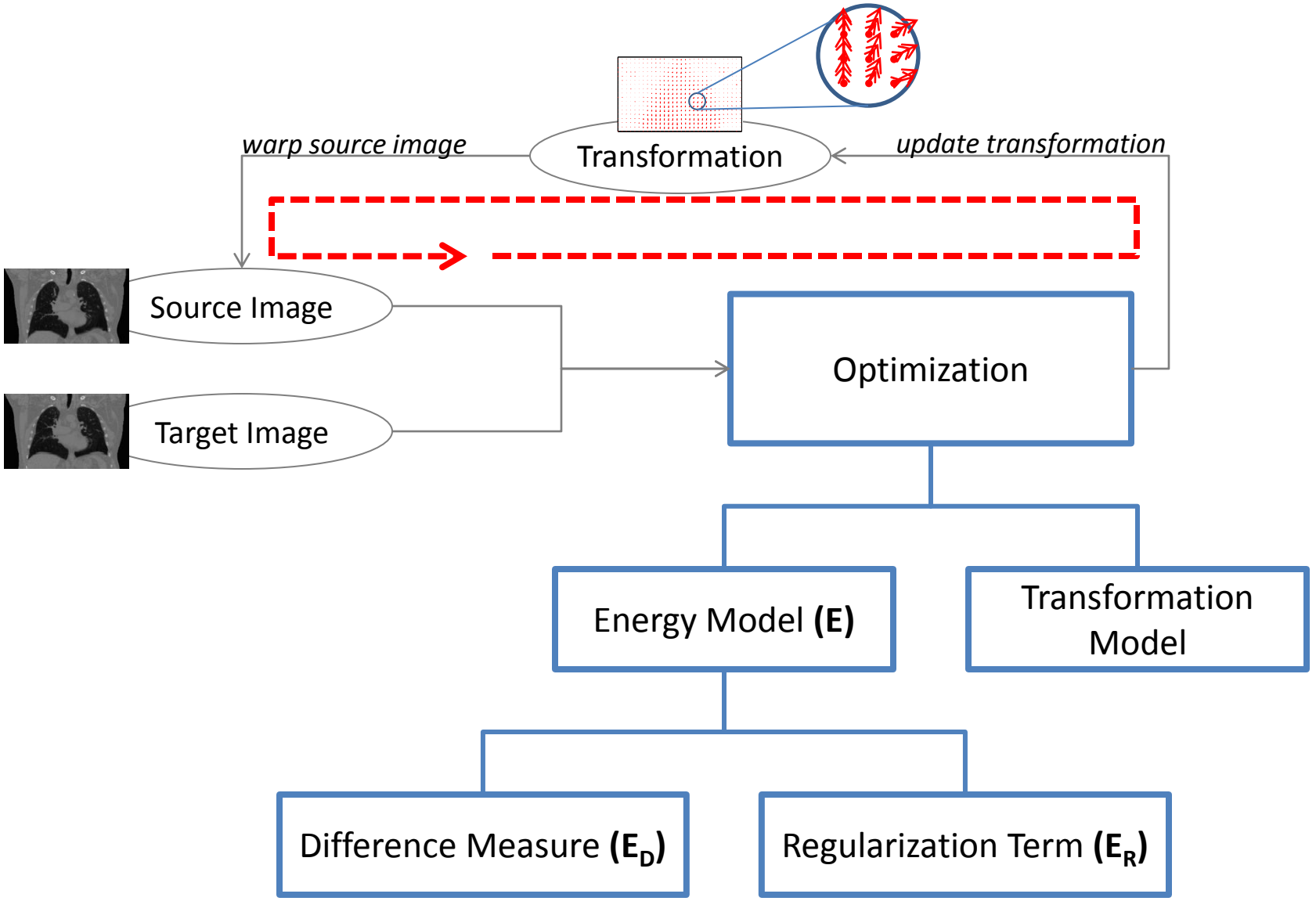


Intensity-based registration:
Minimize a difference term, based on the (pre-processed) image intensities.

No feature-based registration:

- extraction of distinct, sparsely located features
- matching of extracted features

Deformable Registration: General Framework



Formalization of Deformable Registration as a Minimization Problem

Images (target I_T, source I_S): $I : \Omega \to \mathbb{R}$

Image domain: $\Omega \subset \mathbb{R}^d \ , \ d=2,3$

Warped Source Image

$$u' \ = \ \arg\min_{u \in \mathrm{H}} \ E_D(\ I_T \ , \ I_S(\varphi) \ ) \ + \ \lambda \, E_R(\ u \ )$$

Displacements are elements of a Hilbert space H, e.g.: $u \in \mathrm{L}^2$

Deformation: $\varphi = \mathrm{Id} + u \ \ , \ \ \varphi : \Omega \to \mathbb{R}^d$

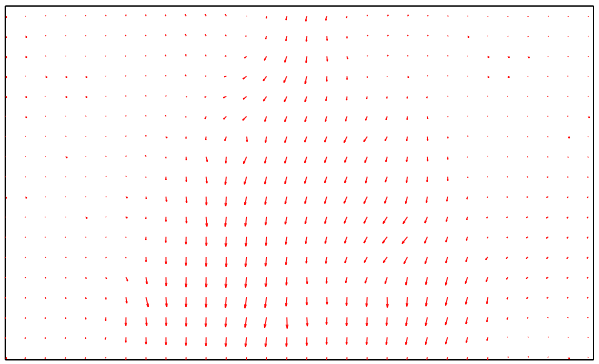
or point-wise: $\varphi(x) = x + u(x)$

Displacement:

$$u : \Omega \to \mathbb{R}^d$$

e.g. $u = [u_x, u_y, u_z]$

difference between original position of a point x and its transformation φ(x)



Please note: Highly heterogeneous notation in the field

Why do we need Regularization?

Since minimizing the difference measure is not enough...

- Motivation for regularization:
  - → **Necessity:**
    Minimization of difference measure only can be ill-posed (#measurements < #unknowns)
    (Optical Flow community: *"Aperture Problem"*)
  - → **Modeling:**
    Regularization can be used to include prior knowledge, for example about underlying tissue properties
  - → **Practical Reasons:**
    Without regularization: High number of local minima in the energy function (→ bad for optimization)

Regularization Strategies

$$u' = \arg\min_{u \in \mathrm{H}} E_D(I_T, I_S(\mathrm{Id}+u)) + \lambda E_R(u)$$

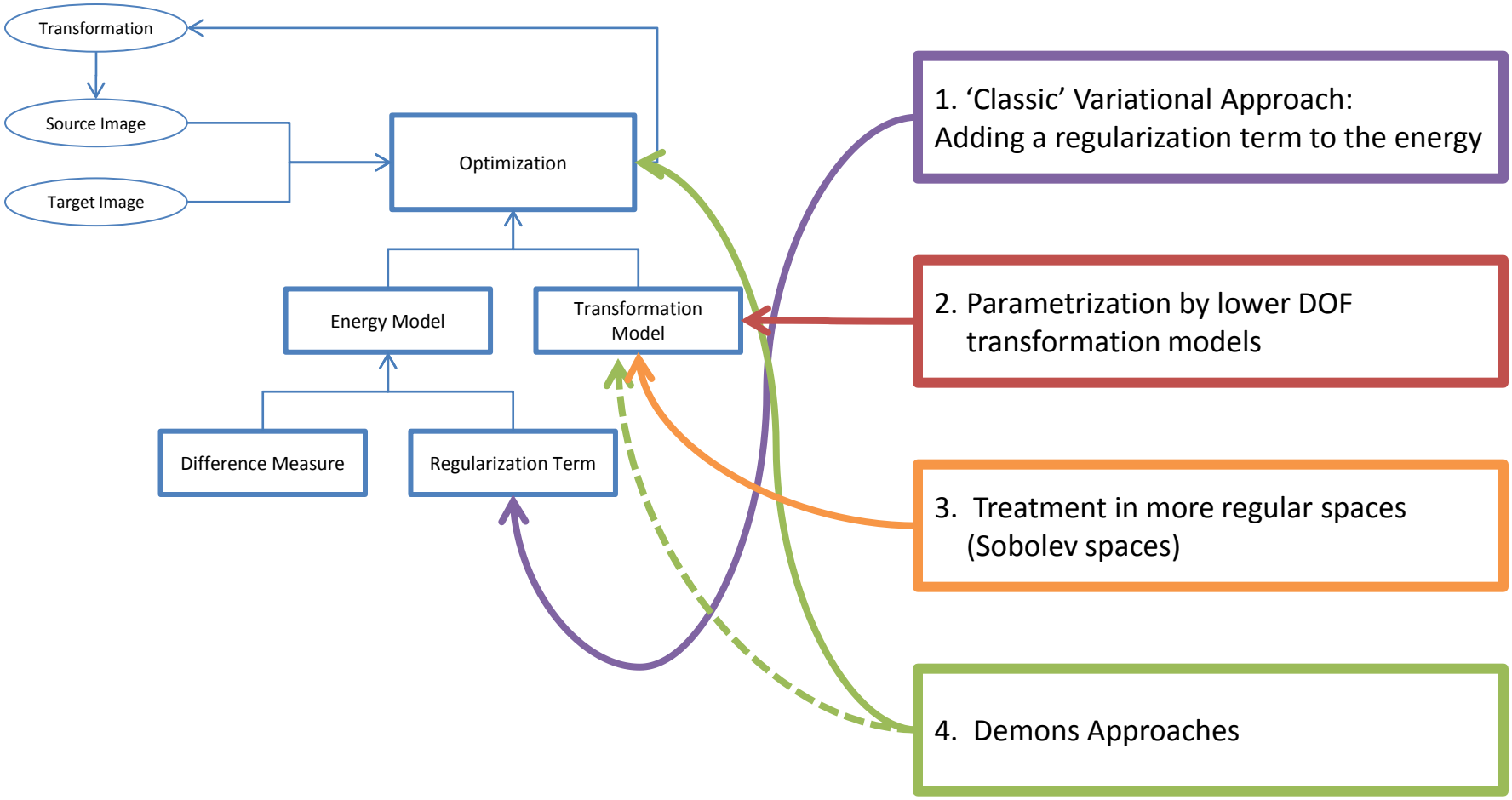
Restricting the space of deformations to a subspace with certain regularity properties:

- Parametrization reducing the number of degrees of freedom
  (e.g. FFD B-Splines)
- Assumption of function spaces which are more restricted than L^2
  → Sobolev spaces

Adding a regularization term to the energy formulation, e.g.:

- Diffusion Regularization
- Curvature Regularization
- Bending Energy
- Linear Elasticity
- Volume Preservation

Regularization Strategies



Outline of this Talk

(structured by different regularization strategies of the approaches)

- **Part I – "Classic" Variational Approach**
  Minimization of energy with a regularization term, in L^2 space

- **Part II – Parametric Models**
  Transformation Model: Restricting the space of deformations
  - Lower DOF transformation models → Parametrization

- **Part III – Sobolev Spaces**
  Transformation Model: Restricting the space of deformations
  - Different choice of underlying function space → Sobolev spaces

- **Part IV – Demons Approaches**
  Efficient forces and regularization by a smooth strategy

- **Part V – Further Points**

Please note:

THIS PRESENTATION

IS A POSTER

PART I

“Classic“ Variational Approach

Outline

- Energy: formulation and properties
- Gradient-based optimization schemes

Classic Variational Approach

- Treatment of transformation in L^2
- Explicit definition of regulariztion term in model

$$u' = \underset{u \in L^2}{\arg\min}\ E_D(\, I_T\,,\, I_S(\mathrm{Id}+u)\,) + \lambda\, E_R(\, u\,)$$

- Non-linear with respect to the displacement beacuse of the dependence on the image function
- High-dimensional problem: e.g. 3 x 256^3 = 50 331 648

→ Numerous local minima

- This line of work started with [Broit 1981], [Bajcsy and Broit 1982]
- Details e.g. in [Modersitzki 2004]

Examplary Model Problem

$$E(u) = E_D(I_T, I_S(\varphi)) + \lambda E_R(u)$$

- Difference Measure: Sum of Square Differences (SSD)

$$E_{\mathrm{D}} = \frac{1}{2} \int_{\Omega} (I_T(x) - I_S(\varphi(x)))^2 \, \mathrm{d}x$$

- Regularization Term: Diffusion Regularization
  (a.k.a. Tikhonov regularization, 1st order regularization)

$$E_R(u) = \frac{1}{2} \int_{\Omega} \sum_{i=1}^{d} \|\nabla u_d(x)\|^2 \mathrm{d}x$$

$$\text{e.g.} \quad E_R(u) = \frac{1}{2} \int_{\Omega} \|\nabla u_x(x)\|^2 + \|\nabla u_y(x)\|^2 \mathrm{d}x$$

→ **Non-linear Least-squares Energy**

General Energy Formulation

$$E(u) = E_D(I_T, I_S(\varphi)) + \lambda E_R(u)$$

$$E(u) = \int_\Omega \rho_D\big(e_D(u)\big)\,\mathrm{d}x + \lambda \int_\Omega \rho_R\big(e_R(u)\big)\,\mathrm{d}x$$

- Energy is defined by
  - an error term based on the displacement: $e(u)$
  - penalty function applied to the error term: ρ
    - assures that the error terms are positive
    - weights the error term

Examples of penalty functions

$\rho(x) = x^2$ → results in application of the L^2 norm

$\rho(x) = |x|$ → results in application of the L^1 norm

General Energy Formulation

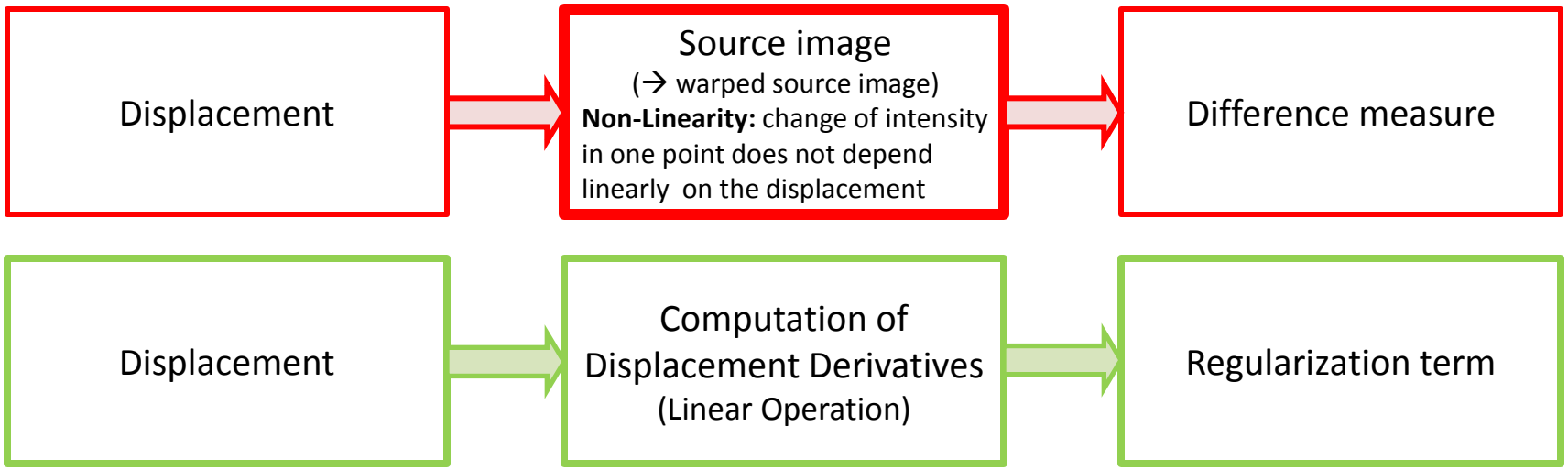
$$E(u) = \int_\Omega \rho_D\big(e_D(u)\big)\,\mathrm{d}x + \lambda \int_\Omega \rho_R\big(e_R(u)\big)\,\mathrm{d}x$$

Depends on deformation through images

$$e_D(u) \equiv e_D(I_T, I_S \circ (\mathrm{Id} + u))$$

→ Non-linearity

- In many cases, the error term for the regularization is linear in the displacement
- Linear operator is mostly a differential operator (e.g. G=∇, G=Δ, ...)



The Non-Linear Least Squares Framework (use of L^2 norm)

$$E(u) = \frac{1}{2}\int_\Omega e(u)^2 = \frac{1}{2}\langle e(u), e(u)\rangle \quad \text{with} \quad e = [e_D, \sqrt{\lambda}\, e_R]^\top$$

$$= \frac{1}{2}\langle e_D(u), e_D(u)\rangle + \lambda\frac{1}{2}\langle e_R(u), e_R(u)\rangle$$

$$= \frac{1}{2}\langle e_D(u), e_D(u)\rangle + \lambda\frac{1}{2}\langle Gu, Gu\rangle$$

$$= \frac{1}{2}\langle e_D(u), e_D(u)\rangle + \lambda\frac{1}{2}\langle u, G^*G\ u\rangle$$

The energy is a squared L2 norm of a non-linear error term.

- Often, the regularization error term is linear in the displacement:
  - G is mostly a differential operator, e.g.:
    - Diffusion: G=∇
    - Curvature: G=Δ
- Exceptions: e.g. Volume preservation (since based on det(J(u)))

Introduces non-linearity through dependence on images

Nice form for taking the derivative

Notes:

- Focus on L^2 in this talk since: popular choice, simpler derivatives
- Example for L^1 regularization, see e.g. [Pock 2007]
- NLLSQ framework *required* only for NLLSQ methods (GN,LM)
- Non least-squares energies can be re-written to fit into a NLLSQ framework (however, this makes the error term non-linear)
  cf. e.g. [Appendix 6.8, Hartley and Zisserman]

Optimization

Iterative Optimization

$$u' = \arg\min_u E(u)$$

- Start with an initial estimate u_0
- Estimate a series of updates $h_1, \ldots, h_n$ such that

$$u' \approx u_0 + h_1 + \ldots + h_n$$

```
do repeat:
        h = compute_update(E,φ);
        φ = φ + h;
```

How to determine the updates h_i?

→ different optimization schmes

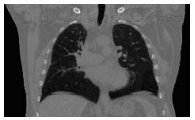
– Gradient-based optimization

$$h = -\tau \ \text{some\_function}\Big(\nabla E(u)\Big)$$

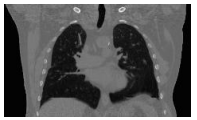
– Gradient-free optimization

$$h = \text{some\_other\_function}\Big(E_D, E_R, u\Big)$$

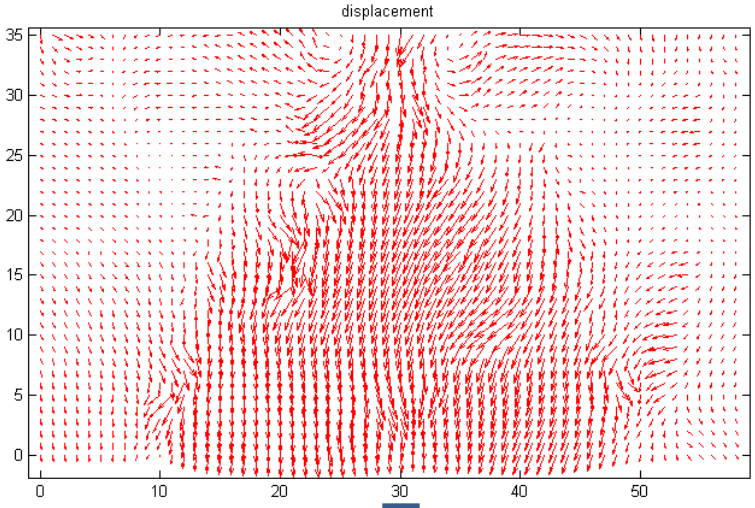
→ More in Ben‘s talk this afternoon

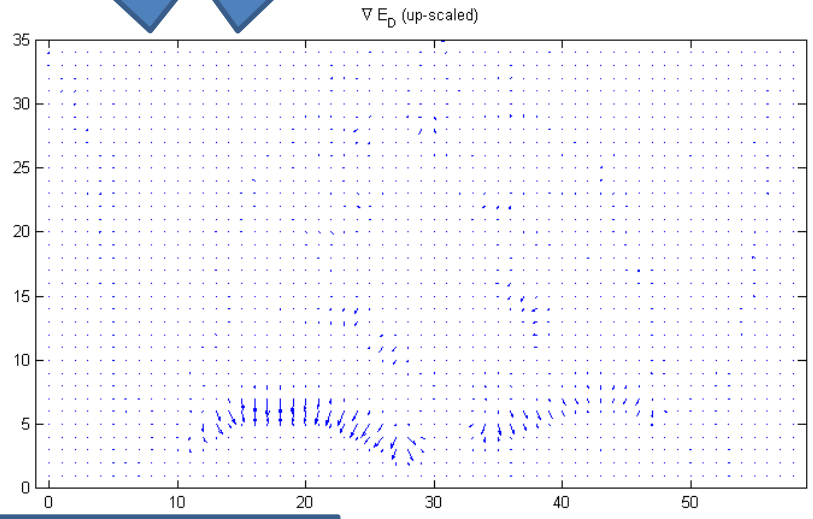


Source image

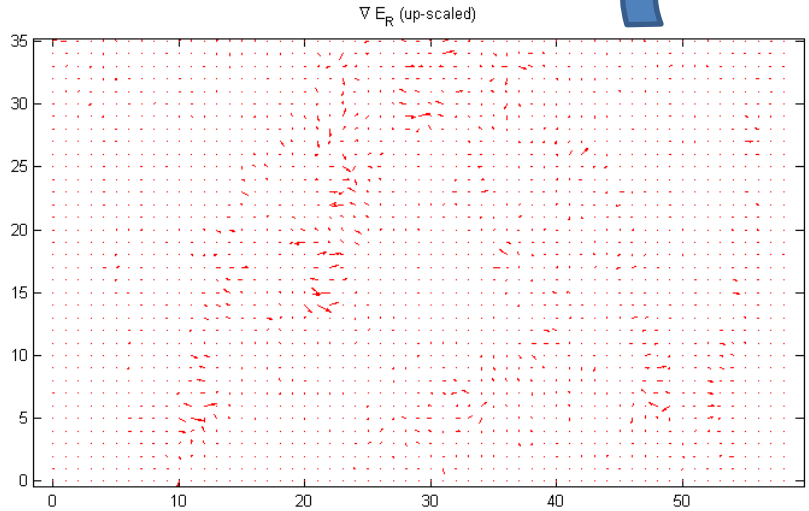


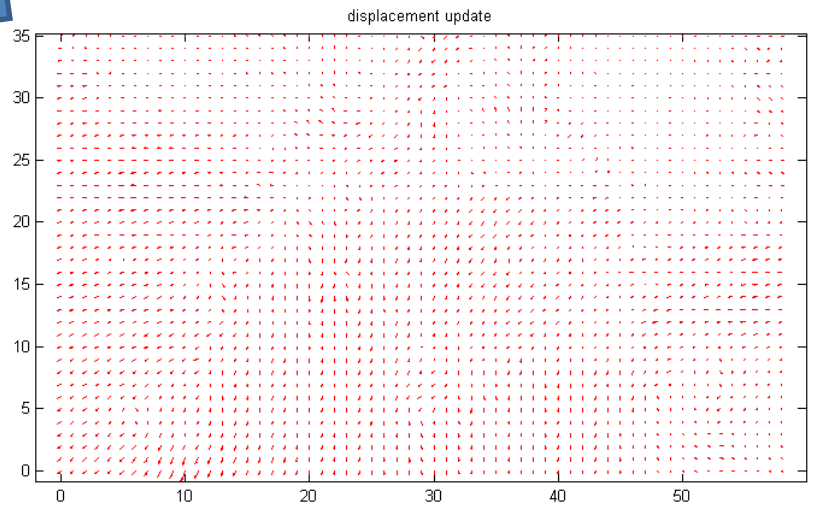
Target image





$$h = -\tau \; P^{-1} \left(\nabla E_D(u) + \lambda \nabla E_R(u) \right)$$





Different Gradient Descent Schemes

1. **Steepest Gradient Descent**

2. **PDE inspired (PDE=partial differential equation)**

3. **Preconditioned Gradient Descent**
   (also known as Quasi-Newton Methods)
   → generalizes 1. and 2.

Optimization by Gradient Descent

Steepest Gradient Descent:

Starting with initial φ_0, repeat until convergence:

$$h = -\tau \, \nabla E(\varphi)$$ *// compute update based on gradient of energy*

$$\varphi = \varphi + h$$ *// apply the update*

Generalized Formulation as Preconditioned Gradient Descent / Quasi-Newton:

$$h = -\tau \, P^{-1} \nabla E(\varphi)$$

$$\varphi = \varphi + h$$

Conditions on P:

- P is a linear operator
- P is symmetric positive definite

Different Choice of P → Different Optimization Scheme

Overview of Gradient Descent Based Optimization Methods

Method	Update Rule	Comment
Steepest Descent	$h = -\tau \ \mathrm{Id}^{-1} \ \nabla E(u)$	+ Simple implementation + Only gradient required – Numerical issues: requires small time steps → many iterations needed
PDE inspired Semi-implicit Discretization	$h = -\tau \ \left(\mathrm{Id} + \tau\lambda\nabla E_R\right)^{-1} \ \nabla E(u)$	+ Numerically stable also for large time steps + Linear operator determined by regularization → difference measure easily exchangable – Poor convergence speed
Gauß-Newton	$h = -\tau \ \left(J_e^\top J_e\right)^{-1} \ \nabla E(u)$	+ Numarically stable also for large time steps + Good convergence speed – Linear operator depends on both, the regularization *and* the difference term – Applicable only to least-squares problems – J_e must be sparse for efficient treatment
Preconditioned Gradient Descent	$h = -\tau \ P^{-1} \ \nabla E(u)$ With P approximating the Hessian of E, e.g.: • Jacobi preconditioning • BFGS	Most general formulation of the above. Properties depend heavily on choice of P. *"Finding a good preconditioner (…) is often viewed as a combination of art and science."* Y. Saad, Iterative Methods for Sparse Linear Systems

1. Steepest Gradient Descent

- Energy: $E(u) = E_D(I_T, I_S(\varphi)) + \lambda E_R(u)$

Starting with initial φ_0, repeat until convergence:

$$h = -\tau \nabla E(\varphi)$$ *// compute update based on gradient of energy*

$$\varphi = \varphi + h$$ *// apply the update*

Only the derivative of the energy w.r.t the displacement is required:

$$\nabla E(\varphi) = \nabla E_D(\varphi) + \lambda \nabla E_R(\varphi)$$

→ derivative of difference measure

→ derivative of regularization term

$\varphi = \mathrm{Id} + u$

Derivative of the Difference Measure

General formulation of the derivative of the difference measure:

$$\frac{\partial E_D(I_T, I_S(\varphi))}{\partial u} = \underbrace{\frac{\partial E_D(I_T, I_S(\varphi))}{\partial I_S(\varphi)}}_{W(I_T, I_S(\varphi))} \underbrace{\frac{\partial I_S(\varphi)}{\partial \varphi}}_{(\nabla I_S)(\varphi)} \underbrace{\frac{\partial \varphi}{\partial u}}_{\mathrm{Id}}$$

Point-wise evaluation at $x \in \Omega$: **point-wise rescaling of the warped gradient of I_S**

$$\underbrace{\frac{\partial E_D(I_T, I_S(\varphi))}{\partial u}(x)}_{\in \mathbb{R}^d} = \underbrace{W(I_T, I_S(\varphi))(x)}_{\in \mathbb{R}} \underbrace{(\nabla I_S)(\varphi(x))}_{\in \mathbb{R}^d}$$

For the **Sum of Squared Differences**, we get:

$$E_D = \frac{1}{2}\int_\Omega (I_T(x) - I_S(\varphi(x)))^2 \mathrm{d}x \quad \Rightarrow \quad \frac{\partial E_D}{\partial u} = -(I_T - I_S(\varphi))\,\nabla I_S(\varphi)$$

Derivative of the Regularization Term

General formulation of the derivative for a regularization term with a quadratic form:

- General regularization term:
- Assume:
  - Squared L2 norm:
    $$E_R(u) = \frac{1}{2}\int_\Omega e(u)^2 = \frac{1}{2}\langle e(u), e(u)\rangle$$
  - Error term is linear in u:
    $$\langle e_R(u), e_R(u)\rangle = \langle Gu, Gu\rangle = \langle u, G^*G\ u\rangle$$

$$E_R(u) = \int_\Omega \rho_R(\ e_R(u)\ )\ \mathrm{d}x$$

$$E_R(u) = \frac{1}{2}\langle u, G^*Gu\rangle$$

- Then, the derivative reads:

$$\frac{\mathrm{d}E_R(u)}{\mathrm{d}u} = G^*Gu$$

For **diffusion regularization**, we get:

$$E_R(u) = \frac{1}{2}\int_\Omega \sum_{i=1}^{d} \|\nabla u_d(x)\|^2 \mathrm{d}x \quad \Rightarrow \quad \frac{\mathrm{d}E_R(u)}{\mathrm{d}u} = \nabla^*\nabla u = -\Delta u$$

1. Steepest Gradient Descent - Summary

- Energy: $E(u) = E_D(I_T, I_S(\varphi)) + \lambda E_R(u)$

Starting with initial φ_0, repeat until convergence:

$$
\begin{aligned}
h &= -\tau \, \nabla E(\varphi_i) \\
&= -\tau \left(\nabla E_D(\varphi_i) + \lambda \nabla E_R(\varphi_i) \right) \\
&= -\tau \left(-(I_T - I_S(\varphi_i)) \, \nabla I_S(\varphi_i) \; - \; \lambda \Delta u_i \right) \\
\varphi_{i+1} &= \varphi_i + h
\end{aligned}
$$

2. PDE Inspired Gradient Flows

Re-formulation as

- time dependent process and computation of the steady state,

$$\arg\min_u E(u) \stackrel{!}{=} \arg\min_u \lim_{t\to\inf} E(u(t))$$

- with the change of the displacement (=updates) opposing the energy gradient

$$\frac{\partial u}{\partial t} = -\frac{\partial E}{\partial u}$$

→ Partial differential Equation (PDE)

(non-linear, time independent problem → linear, time dependent problem)

Discretiztion of time (time→iterations):

→ Explicit time discretization (=steepest gradient descent)

→ Semi-implicit time discretization

2. PDE Inspired Gradient Flows

- Energy: $E(u) = E_D(I_T, I_S(\varphi)) + \lambda E_R(u)$
- Gradient Flow: $-\frac{\partial u}{\partial t} = \frac{\partial E}{\partial u}$ Notation: $h := u_{t+\tau} - u_t$, $\nabla E := \frac{\partial E}{\partial u}$

2.1. Explicit time discretization

$$-\frac{u_{t+\tau} - u_t}{\tau} = \nabla E_D(u_t) + \lambda \nabla E_R(u_t)$$

$$h = -\tau\left(\nabla E_D(u_t) + \lambda \nabla E_R(u_t)\right)$$

$$h = -\tau\ \nabla E(u_t)$$

Regularization reacts to irregularities of displacement from last step

equivalent to **steepest gradient descent**!

2.2 Semi-implicit time discretization

$$-\frac{u_{t+\tau} - u_t}{\tau} = \nabla E_D(u_t) + \lambda \nabla E_R(u_{t+\tau})$$

$$h = -\tau\left(\mathrm{Id} + \tau\lambda\nabla E_R\right)^{-1}\left(\nabla E_D(u_t) + \lambda \nabla E_R(u_t)\right)$$

$$= -\tau\ \left(\mathrm{Id} + \tau\lambda\nabla E_R\right)^{-1}\ \nabla E(u_t)$$

Regularization assures regularity of displacement in next step

PDE Inspired Gradient Flows

2. 1 Explicit time discretization:

$$h = -\tau \left(\nabla E_D(u_t) + \lambda \nabla E_R(u_t)\right)$$

Numerically stable only for small time steps
→ many iterations required

Simple computation, only gradient of energy required

2.2 Semi-implicit time discretization:

$$h = -\tau \left(\mathrm{Id} + \tau\lambda\nabla E_R\right)^{-1} \nabla E(u_t)$$

$$\Leftrightarrow$$

$$\left(\mathrm{Id} + \tau\lambda\nabla E_R\right) h = -\tau \nabla E(u_t)$$

- No numerical restriction on time step
- Note: setting-dependent step size limit exists!

- Solution of linear system required
- linear operator depends on regularizer
- Underlying assumption: regularizer is a quadratic form

EXAMPLE:
Semi-implicit time discretization of gradient flow

General Semi-Implicit Flow: $h = -\tau \left(\mathrm{Id} + \tau\lambda\nabla E_R\right)^{-1} \left(\nabla E_D(u_t) - \lambda\nabla E_R(u_t)\right)$

- SSD: $E_{\mathrm{D}} = \frac{1}{2}\int_\Omega (I_T(x) - I_S(\varphi(x)))^2 \, \mathrm{d}x \quad \Rightarrow \quad \nabla E_{\mathrm{D}}(u) = -(I_T - I_S(\varphi)) \ \nabla I_S(\varphi)$
- Diffusion Reg.: $E_R = \frac{1}{2}\int_\Omega \sum_{i=1}^{d} \|\nabla u_d(x)\|^2 \mathrm{d}x \quad \Rightarrow \quad \nabla E_R(u) = -\Delta u \quad \Rightarrow \quad \nabla E_R = -\Delta$

Semi-Implicit Flow for SSD and Diffusion Regularization:

$$h = -\tau \left(\mathrm{Id} - \tau\lambda\Delta\right)^{-1} \left(-(I_T - I_S(\varphi))\nabla I_S(\varphi) - \lambda\Delta u\right)$$

EXAMPLE:
Semi-implicit time discretization of gradient flow

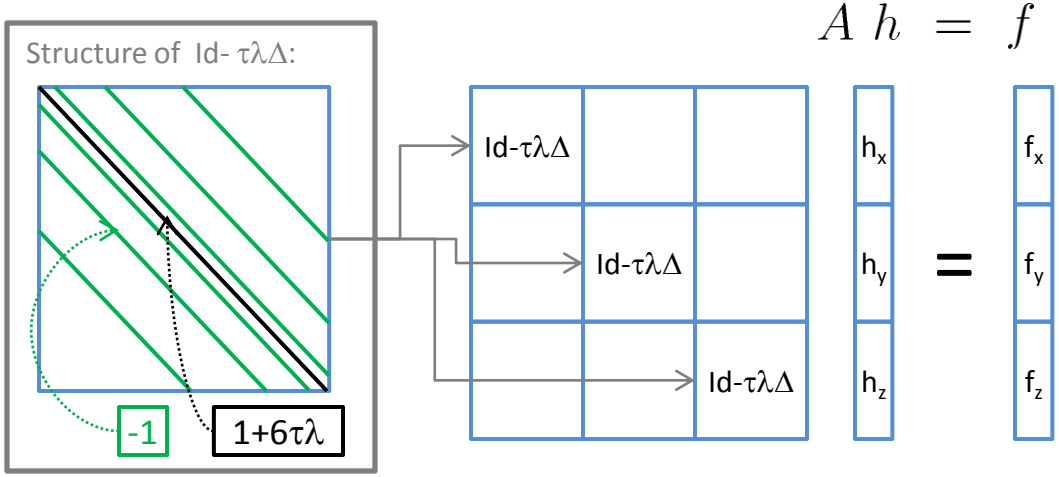
Semi-Implicit Flow for SSD and Diffusion Regularization:

$$h = -\tau \left(\mathrm{Id} - \tau\lambda\Delta\right)^{-1} \left(-(I_T - I_S(\varphi))\nabla I_S(\varphi) - \lambda\Delta u \right)$$

One possible way of computing *h*: solving the corresponding linear system:

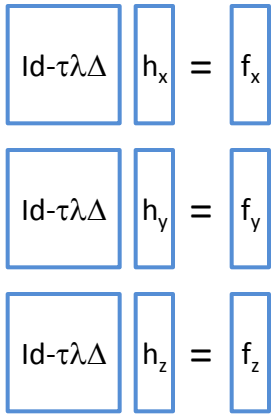
$$\left(\mathrm{Id} - \tau\lambda\Delta\right)\ h \ = \ -\tau \left(-(I_T - I_S(\varphi))\nabla I_S(\varphi) - \lambda\Delta u \right)$$

$$A\ h \ = \ f$$



Side note:
The linear operator ***A*** couples the dimensions of ***h***
→ single dimensions can be solved for independently.

This is not the case in general, e.g it does not hold for Linear Elasticity regularization, or Gauß-Newton optimzation for Diffusion regularization.



An alternative way of solving for *h*: Convolution with corresponding Green's Function

(proposed for registration in [Bro-Nielsen and Gramkow 1996])

- With $\boldsymbol{K}$ being the Green's Function to $\boldsymbol{P}$ we get:

$$h = -\tau \; P^{-1} \nabla E$$
$$= -\tau \; K * \nabla E$$

- Green's Function $\boldsymbol{K}$ to operator $\boldsymbol{P}$ defined by:

$$PK(x,s) = \delta(x-s) \quad \Rightarrow \quad Pu = f \Rightarrow u = K * f$$

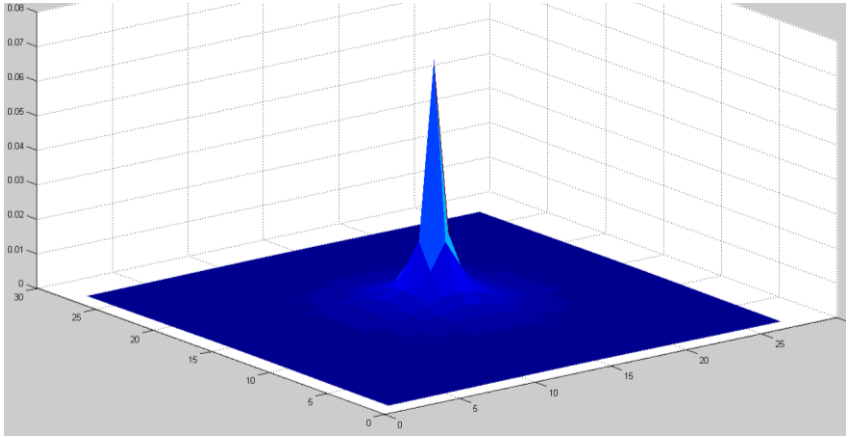
- Computation of Green's function:
  - Analysis of the eigenfunctions of the operator
  - Simple approximative approach: solve discretized system $\widehat{P}\widehat{K} = \widehat{\delta}$

Green's Function Example

- For diffusion regularization: $E_R = \int_\Omega \sum_d \|\nabla u_d(x)\|^2 \mathrm{d}x$

  the update rule resulting from semi-implicit discretization reads

$$h = -\tau \left(\mathrm{Id} - \tau\lambda\Delta\right)^{-1} \nabla E$$



Green's function corresponding to $\left(\mathrm{Id} - \tau\lambda\Delta\right)$

Gauß-Newton Optimization (GN)

Gauß-Newton Optimization:

GN computes the update ***h*** in every iteration, by computing the critical point of a quadratic function, based on the linearization of the error term ***e*** by $l(h) = e(u) + J_e(u)h$

For $$E = \frac{1}{2}\langle e, e\rangle$$

we get

$$h = \arg\min_{h'} \frac{1}{2}\langle l(h'), l(h')\rangle \quad \frac{\mathrm{d}}{\mathrm{d}h'} \stackrel{!}{=} 0$$

resulting in

$$h = -\left(J_e^\top J_e\right)^{-1} \nabla E(u)$$

Application to SSD + Diffusion Reg.:

$$E = \frac{1}{2}\langle e, e\rangle \quad \text{with} \quad e = [e_D, e_R]^\top$$

$$e_\mathrm{D} = [I_\mathrm{T} - I_\mathrm{S}(\varphi)]$$

$$\Rightarrow \quad l_\mathrm{D} = I_\mathrm{T} - I_\mathrm{S}(\varphi) - \nabla I_\mathrm{S}(\varphi)^\top h$$

$$e_\mathrm{R} = \sqrt{\lambda}[\nabla u_x, \nabla u_y] \Rightarrow \quad l_\mathrm{R} = e_\mathrm{R}$$

$$J_e^\top J_e = \nabla I_\mathrm{S}(\varphi)\nabla I_\mathrm{S}(\varphi)^\top - \lambda\Delta$$

Gauß-Newton update step for SSD and Diffusion Regularization:

$$h = -\tau\left(\nabla I_\mathrm{S}(\varphi)\nabla I_\mathrm{S}(\varphi)^\top - \lambda\Delta\right)^{-1}\left(-(I_T - I_S(\varphi))\nabla I_S(\varphi) - \lambda\Delta u\right)$$

EXAMPLE: Gauß-Newton Optimization

$$h = -\tau\left(\nabla I_S(\varphi)\nabla I_S(\varphi)^\top - \lambda\Delta\right)^{-1}\left(-(I_T - I_S(\varphi))\nabla I_S(\varphi_t) - \lambda\Delta u\right)$$

$$\Leftrightarrow$$

$$\left(\nabla I_S(\varphi)\nabla I_S(\varphi)^\top - \lambda\Delta\right) h = -\tau\left(-(I_T - I_S(\varphi))\nabla I_S(\varphi) - \lambda\Delta u\right)$$

$$A\,h = f$$

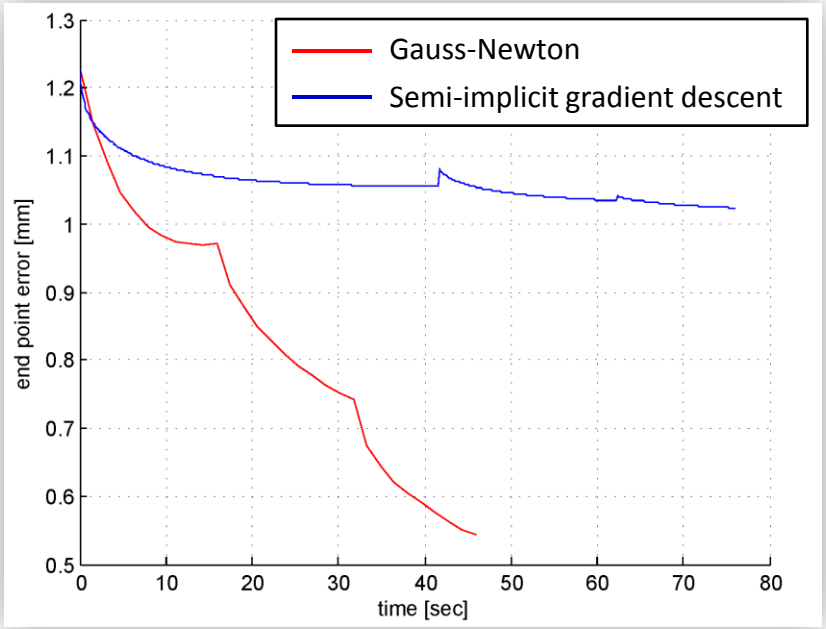
$$\begin{pmatrix} \nabla I_S^x \nabla I_S^x & \nabla I_S^x \nabla I_S^y & \nabla I_S^x \nabla I_S^z \\ \nabla I_S^y \nabla I_S^x & \nabla I_S^y \nabla I_S^y & \nabla I_S^y \nabla I_S^z \\ \nabla I_S^z \nabla I_S^x & \nabla I_S^z \nabla I_S^y & \nabla I_S^z \nabla I_S^z \end{pmatrix} - \lambda \begin{pmatrix} \Delta & & \\ & \Delta & \\ & & \Delta \end{pmatrix} \begin{pmatrix} h_x \\ h_y \\ h_z \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}$$

Side notes:
- The linear operator ***A*** couples the dimensions of ***h***
- The linear operator ***A*** is sparse

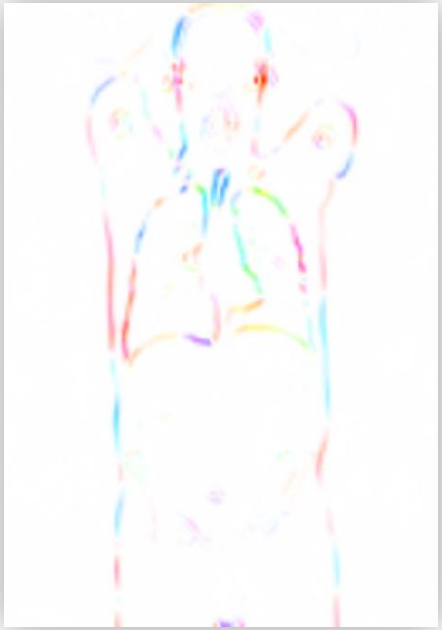
J^TJ becomes dense for statistical difference measures (CR,MI) → not (efficiently) applicable!

Intuition about the Difference in the Behavior of the Different Methods

- Steepest Descent vs. Preconditioned Descent



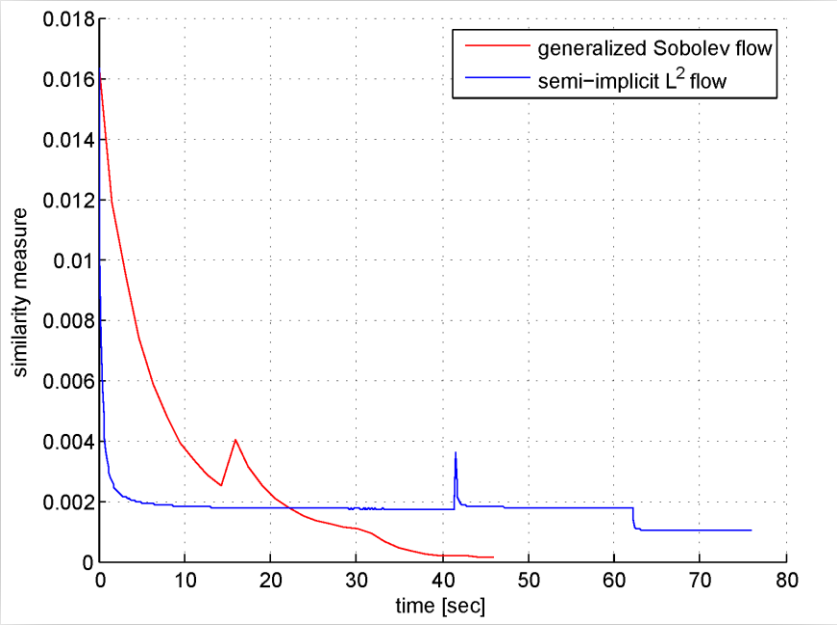
Difference in convergence speed



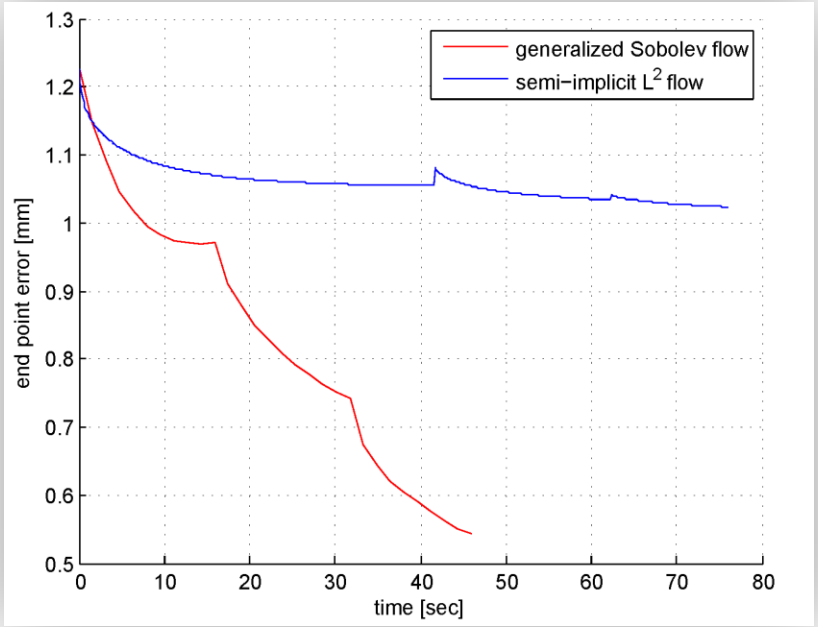
Structure of update for semi-implicit

Structure of update for Gauss-Newton

The Pitfall of Premature Convergence



a) Decrease of Difference Measure



b) Decrease of Error

Blue *seems* to converge much faster than red.
Note: log of update magnitude exhibits similar behavior.

SUMMARY: Overview of Gradient Descent Based Optimization Methods

Method	Update Rule	Comment
Steepest Descent	$h = -\tau \ \mathrm{Id}^{-1} \ \nabla E(u)$	+ Simple implementation + Only gradient required – Numerical instable: requires small time steps → many iterations needed
PDE inspired Semi-implicit Discretization	$h = -\tau \ \left(\mathrm{Id} + \tau\lambda\nabla E_R\right)^{-1} \ \nabla E(u)$	+ Numerically stable also for large time steps + Linear operator determined by regularization → difference measure easily exchangable – Poor convergence speed
Gauß-Newton	$h = -\tau \ \left(J_e^\top J_e\right)^{-1} \ \nabla E(u)$	+ Numarically stable also for large time steps + Good convergence speed – Linear operator depends on both, the regularization *and* the difference term – Applicable only to least-squares problems – J_e must be sparse for efficient treatment
Preconditioned Gradient Descent	$h = -\tau \ P^{-1} \ \nabla E(u)$ With P approximating the Hessian of E, e.g.: • Jacobi preconditioning • BFGS, c.f. [Modersitzki 2009] • For def. Registration: [Zikic 2010 MICCAI]	Most general formulation of the above. Properties depend heavily on choice of P. *"Finding a good preconditioner (...) is often viewed as a combination of art and science."* Y. Saad, Iterative Methods for Sparse Linear Systems

Some Further Points

Equilibrium of Forces at Convergence

$$h = -\tau\ P^{-1}\ \nabla E(\varphi)$$

$$= -\tau\ P^{-1}\ \underbrace{\left(\nabla E_D(\varphi) + \lambda \nabla E_R(\varphi)\right)}_{\nabla E = 0 \ \rightarrow \ h = 0}$$

$$\varphi = \varphi + h$$

Procedure converges when updates vanish,
that is at equilibrium of gradients of the difference measure and the regularization term

$$\nabla E_D(\varphi) = -\lambda \nabla E_R(\varphi)$$

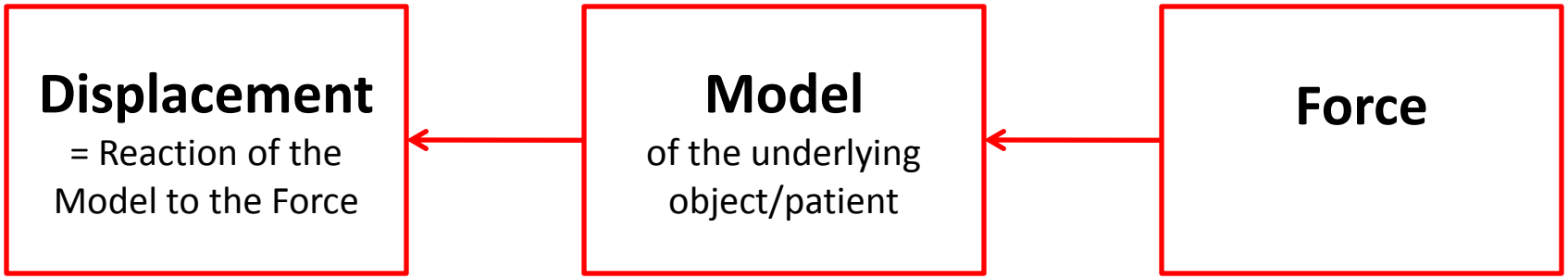
Until convergence, the gradient of the difference measure dominates:

$$\|\nabla E_D(\varphi)\| > \|\lambda \nabla E_R(\varphi)\|$$

Simulation Point of View

e.g. in [Broit 1981], [Bajcsy and Broit 1982]

- For PDE-insipred approach (semi-implicit discretization):



$$u_{t+\tau} = \left(\mathrm{Id} + \tau\lambda\nabla E_R\right)^{-1} \underbrace{\left(-\tau\nabla E_{\mathrm{D}}(u_t)\right)}_{f}$$

- Other implementations of the Model are possible:
  - Spring-Mass model
  - Finite Element Model (equivalent to FE discretization of the linear operator)
- Other forces possible:
  - Distances between landmarks

Elastic and Fluid Registration Types

- **Fluid registration** [Christensen 1994]

  MOTIVATION:

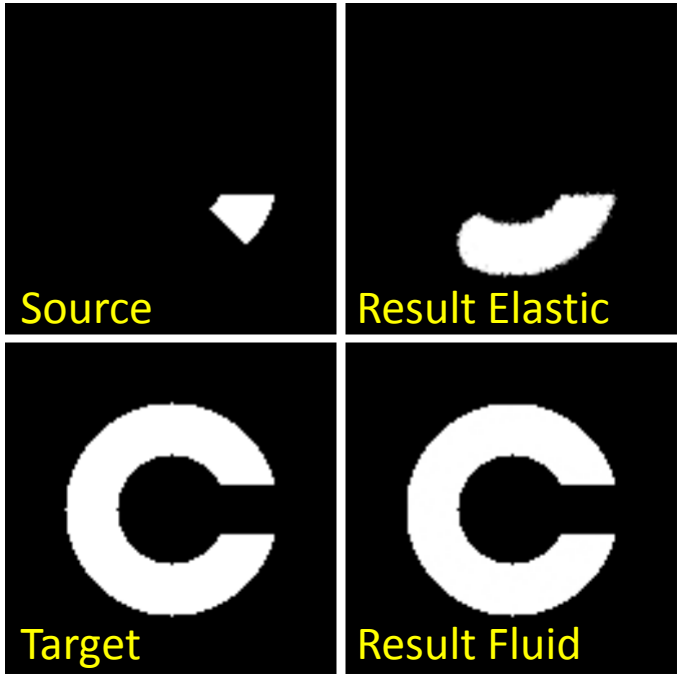
  achieve LARGE deformations

  IDEA:

  use no conservation of energy

  → Simulation of a fluid acting under forces

  → regularization of velocities

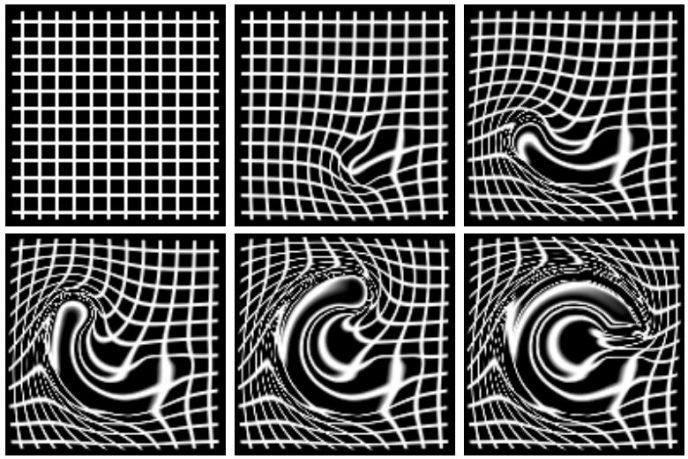


Images from [Christensen 1994]

Modern view:

fluid-type registration = regularization of updates

(UPDATES = change of displacement = VELOCITIES)

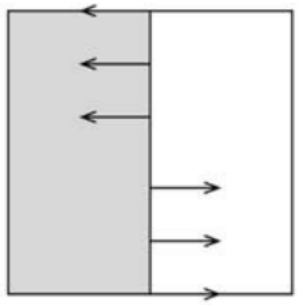


"Time progression of the fluid trans-formation applied to a rectangular grid"

- Challenge to maintain the regular deformation without foldings
- No transport of deformation in homogeneous regions

Update Modes: Additional and Compositional

[Chefd'hotel 2002][Stefanescu 2004][Vercauteren 2009]



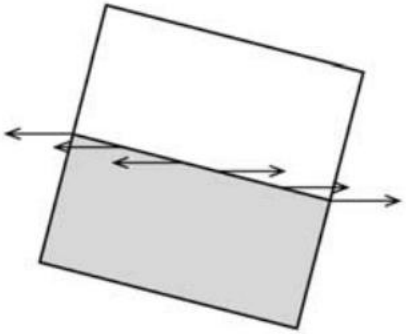
Source Image with gradient
(images from [Stefanescu 2004])

Appliction of updates:

- Additional Mode: $\varphi_{i+1} = \varphi_i + h$

$$\Rightarrow \nabla E_{\mathrm{D}}(\varphi) = \omega \nabla I_{\mathrm{S}} \circ \varphi = \omega \cdot (\nabla I_{\mathrm{S}}) \circ \varphi$$

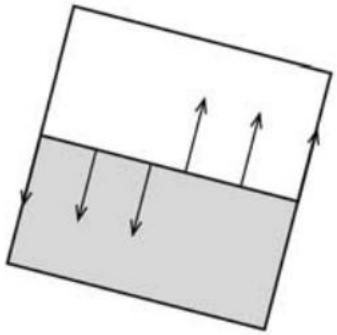
"Warping of image gradient"



- Compositional Mode: $\varphi_{i+1} = \varphi_i \circ (\mathrm{Id} + h)$

$$\Rightarrow \nabla E_{\mathrm{D}}(\varphi) = \omega \nabla (I_{\mathrm{S}} \circ \varphi)$$

"Gradient of warped image"

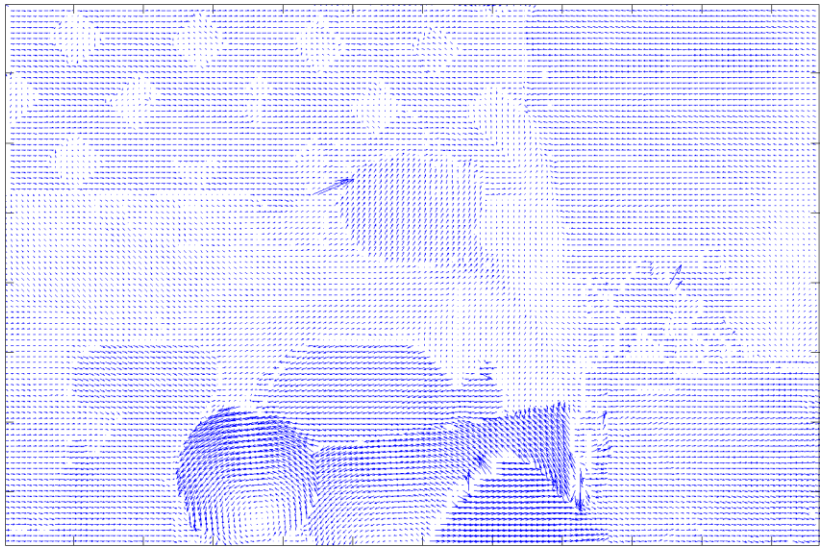


Connection to Optical Flow

OPTICAL FLOW: Computation of the *visible motion** between corresponding structures in two input images

*visible motion = projection of the actual 3D motion of visible structures into the image plane

- Basically a motion estimation problem
  ("Optical Flow" is a problem, not a method!)



For overview of current methods and evaluation, see Middlebury Flow Data Base:
http://vision.middlebury.edu/flow/eval/results/results-e1.php

Connection to Optical Flow

Differences between Optical Flow and Medical Image Registration:

- Input data:
  optical images, no real multi-modality
  (only shadows / changes in lighting)
  → less work on difference measures
- Discontinuities in displacement fields:
  → anisotropic regularization, use of robust norms (e.g. L^1)

Horn and Schunck Optical Flow Method

[B.K.P. Horn and B.G. Schunck. Determining optical flow. Artificial Intelligence, 1981.]

The incremental Horn and Schunck method

=

optimization of a non-linear energy (SSD+Diffusion)

by the Gauss-Newton (GN) method. [Zikic 2010 BMVC]

SUMMARY

- Energy: formulation and properties
- Gradient-based optimization schemes
  → variations of preconditioned gradient descent
- Solving the Linear system:
  - "Standard" solvers
  - Green's function
- Registration types: Elastic and Fluid
- Update modes: Additional and Compositional

BREAK

Outline of this Talk

- **Part I – "Classic" Variational Approach**
  Minimization of Energy which includes a regularization term in L2 space

- **Part II – Parametric Models**
  Transformation Model: Restricting the space of deformations
  - Lower DOF transformation models →"parametrization"

- **Part III – Sobolev Spaces**
  Transformation Model: Restricting the space of deformations
  - Different choice od underlying space →Sobolev spaces

- **Part IV – Demons Approaches**
  Regularization by pair and smooth strategy

- **Part V – Discussion: Connections Between the Different Approaches and Further Points**
  - **Elastic vs. Fluid-type Regularization**
  - **Solving the Linear Systems**

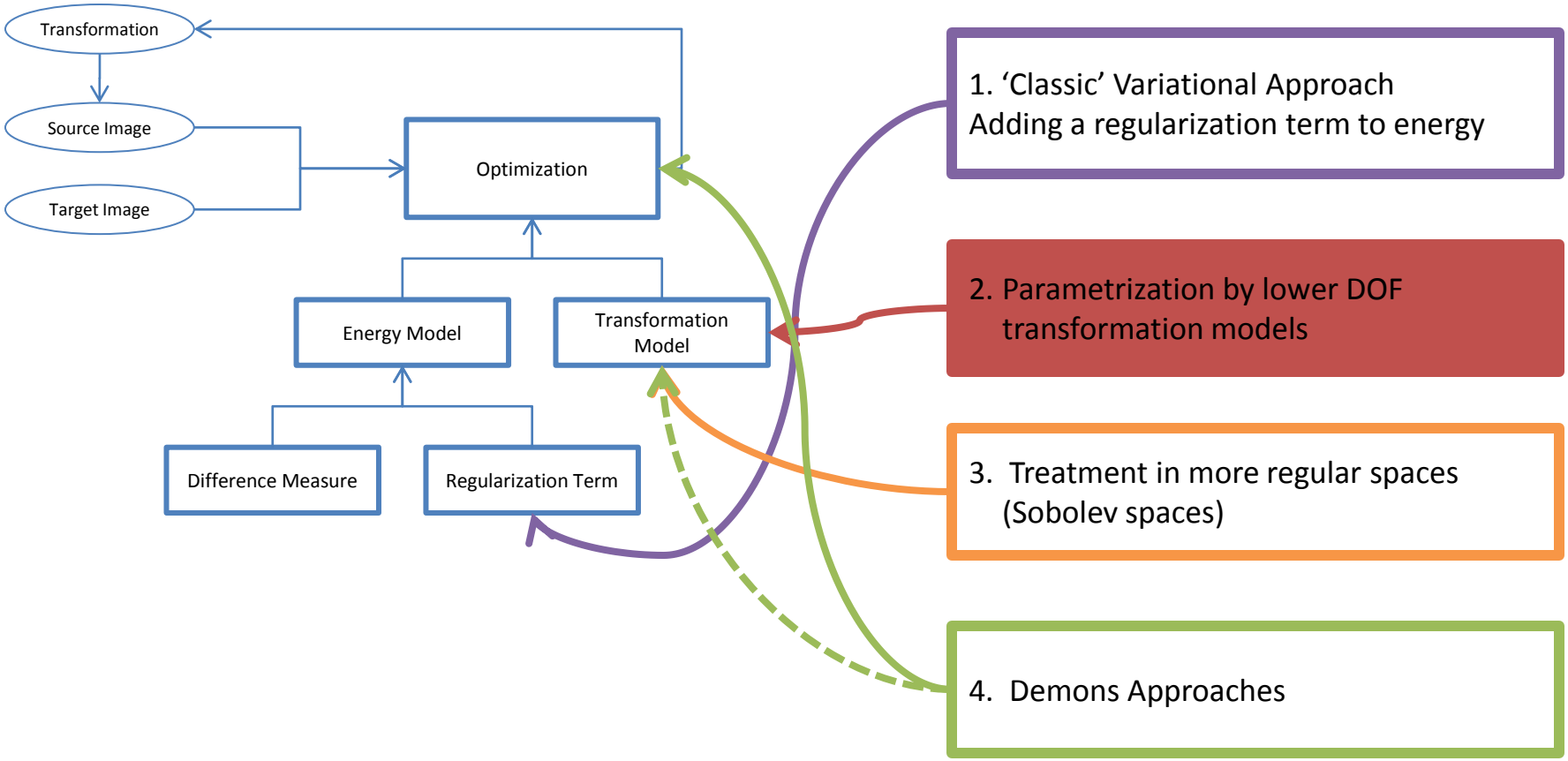
Changing the space in which the optimization is performed

- Same energy is minimized
- Regularity induced by construction
  (continuous analogon to parametrization)
  → explicit regularization term no longer necessary
- Different local minima
- Different minimization paths (a.k.a. gradient flows)

PART II

Parametrization by Linear Models

Regularization Strategies



Outline

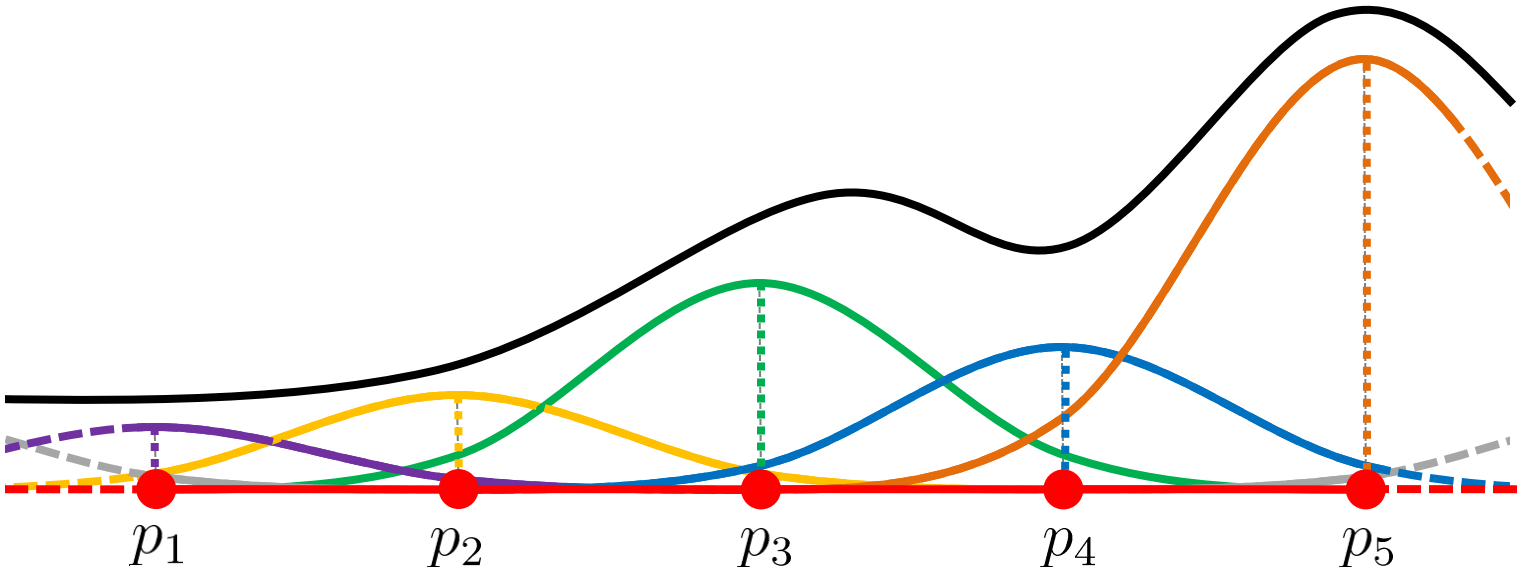
- Parametrization by linear models
- Examples of parametric methods
  - Free-form Deformation B-Splines (FFD B-Splines)
  - Trigonometric Functions
    (Discrete Fourier/Cosine Transformation Bases)
  - Radial Basis Functions (RBF)
- Gradient descent on linear models

The Idea

$$u_p(x) = \sum_k p_k B_k(x)$$

Parameters: $p_k \in \mathbb{R}^d$

Basis functions: $B_k : \Omega \to \mathbb{R}$

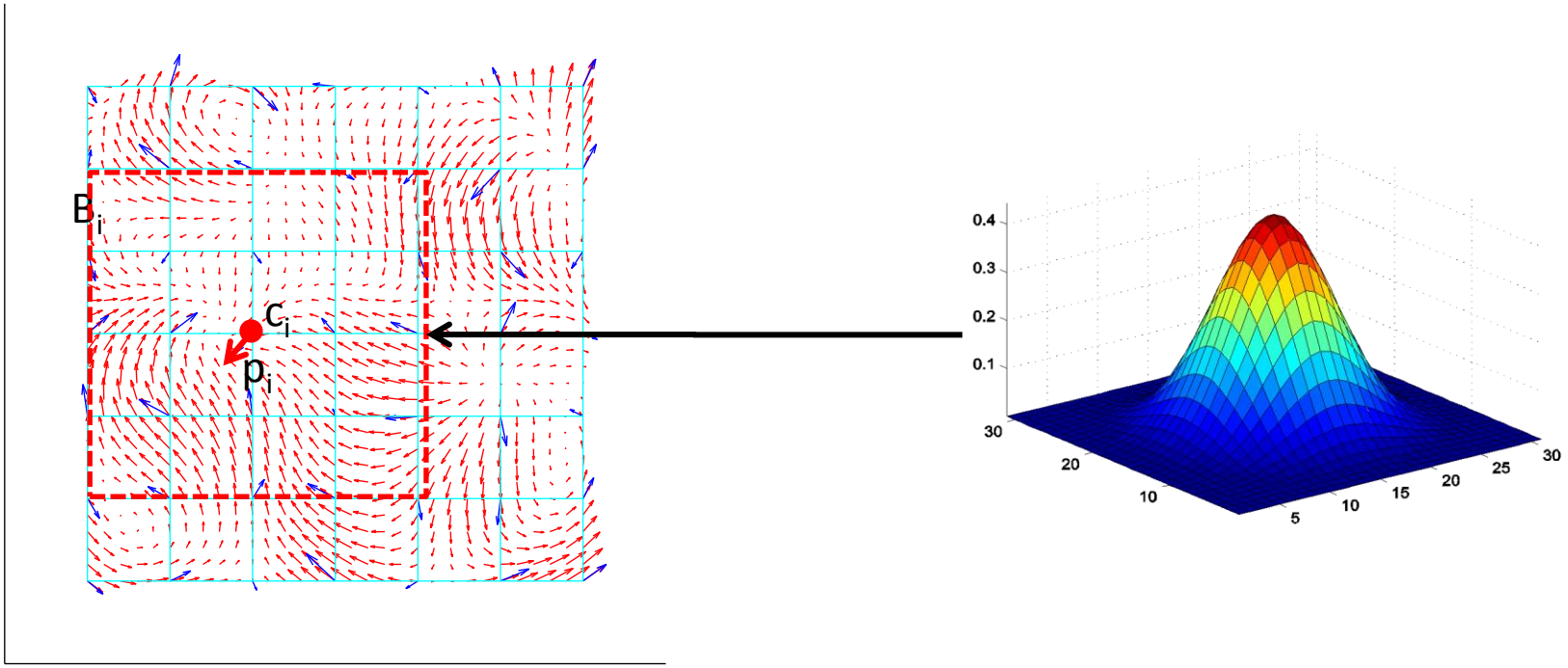


The Idea

$$u_p(x) = \sum_k p_k B_k(x)$$

Parameters: $p_k \in \mathbb{R}^d$

Basis functions: $B_k : \Omega \to \mathbb{R}$



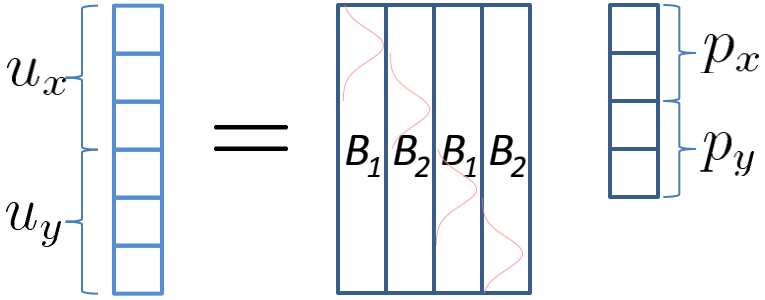
Recap: Parametrization by Linear Models

- Parametrization (linear model):

$p_k \in \mathbb{R}^d$ $\quad B_k : \Omega \to \mathbb{R}$

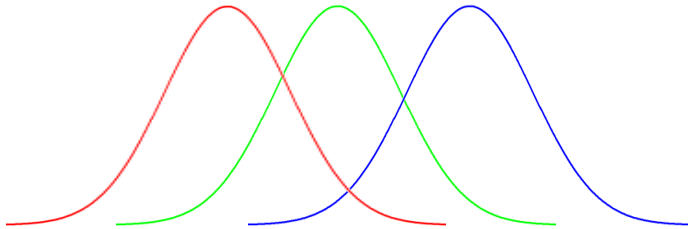
$$u_p(x) = \sum_k p_k B_k(x)$$

$$u = B\ p$$

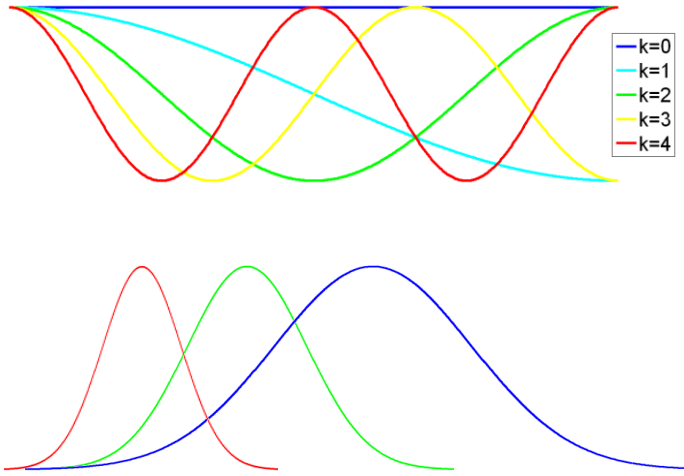


Characterization of Linear Parametrization Basis Type – **Shape** of $\boldsymbol{B_k}$'s

- Same shape of all $\boldsymbol{B_k}$'s
  $\boldsymbol{B_k}$'s are translated versions of $\boldsymbol{B}$:
  $\boldsymbol{B_k(x)=B(x-c_k)}$
  - Free-form deformation (FFD) B-Splines

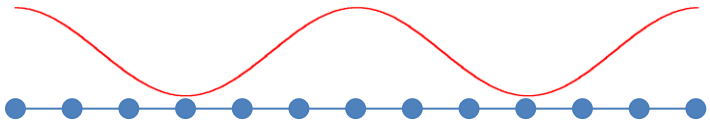


- Different shape of $\boldsymbol{B_k}$'s
  - Fourier/Cosine Bases
  - RBFs with different parameters
    (e.g. Gaussians with different variance)

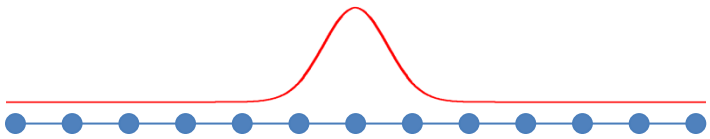


Characterization of Linear Parametrization Basis Type – **Support** of $\boldsymbol{B_k}$'s

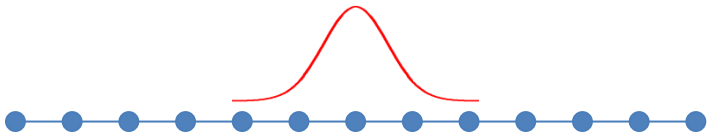
- Global Support
  - Fourier/Cosine Bases



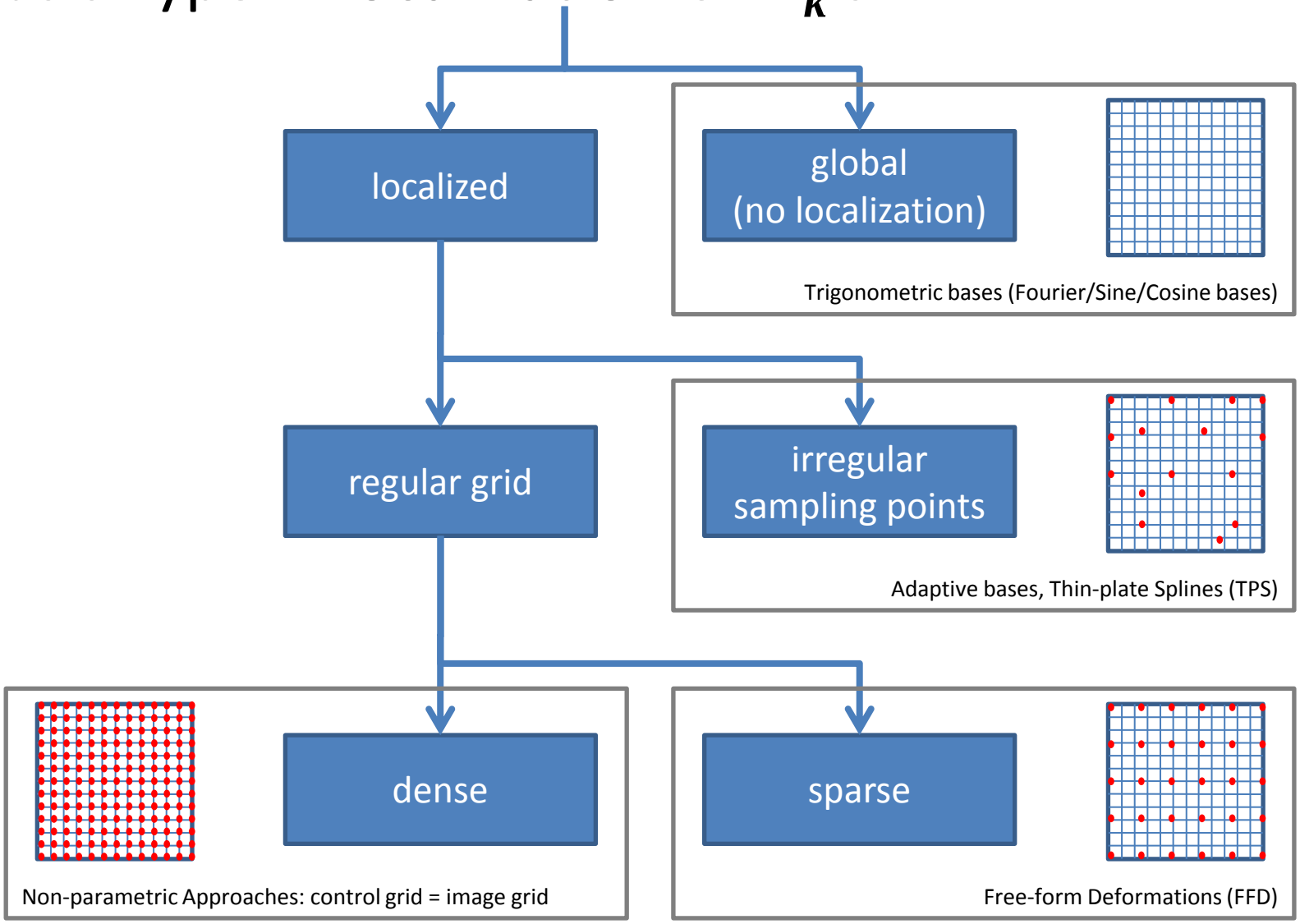
  - Radial basis functions RBFs (e.g. Thin-plate Splines (TPS))
  - Gaussians (in theory)



- Compact Support
  - B-Splines
  - Some RBFs
  - Gaussians (in practice)



Characterization of Linear Parametrization Basis Type – **Localization** of $\boldsymbol{B_k}$'s



Parametrization by Linear Models

Some Examples of Parametrizations

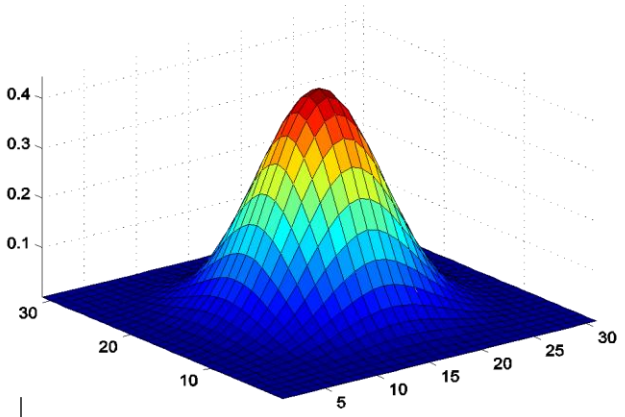
$$u_p(x) = \sum_k p_k B_k(x)$$

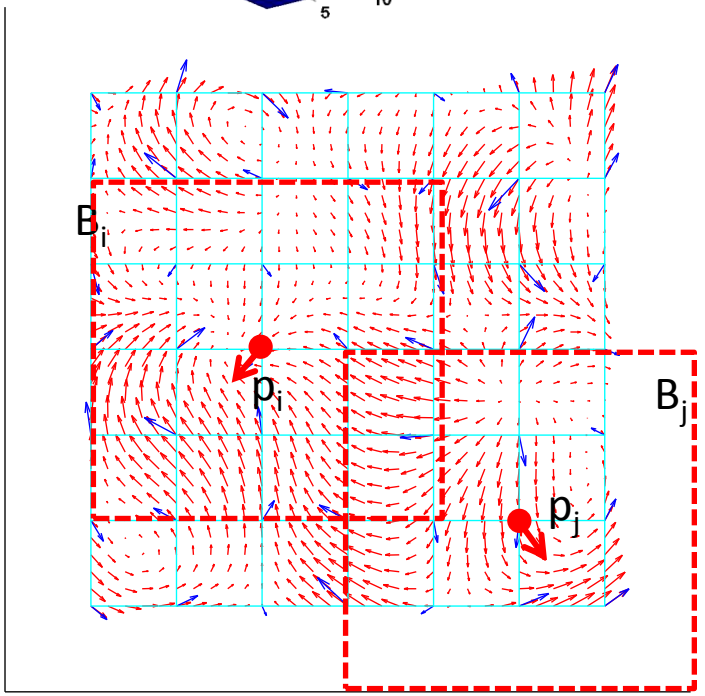
B-Spline Free-form Deformations (FFDs)

- **Type:**
  Cubic B-Splines $\boldsymbol{B_k(x) = B(x\text{-}c_k)}$
  with $\boldsymbol{B}$ being a tensor product of corresponding 1-dimensional B-Splines

- **Location:**
  Regular grid of control points

- **Support:**
  Compact, depending on degree of B-Spline, and image and grid resolution



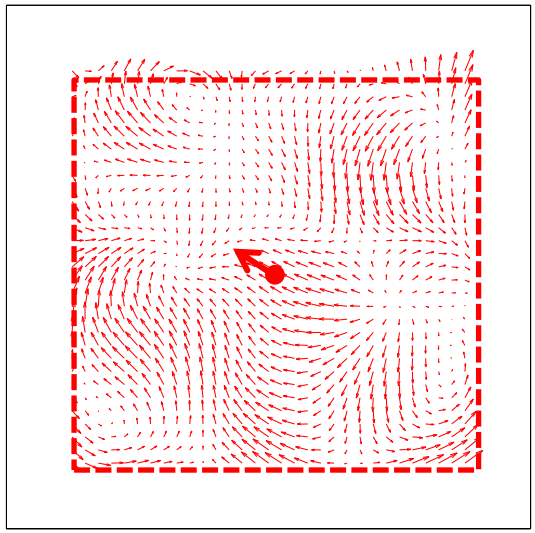


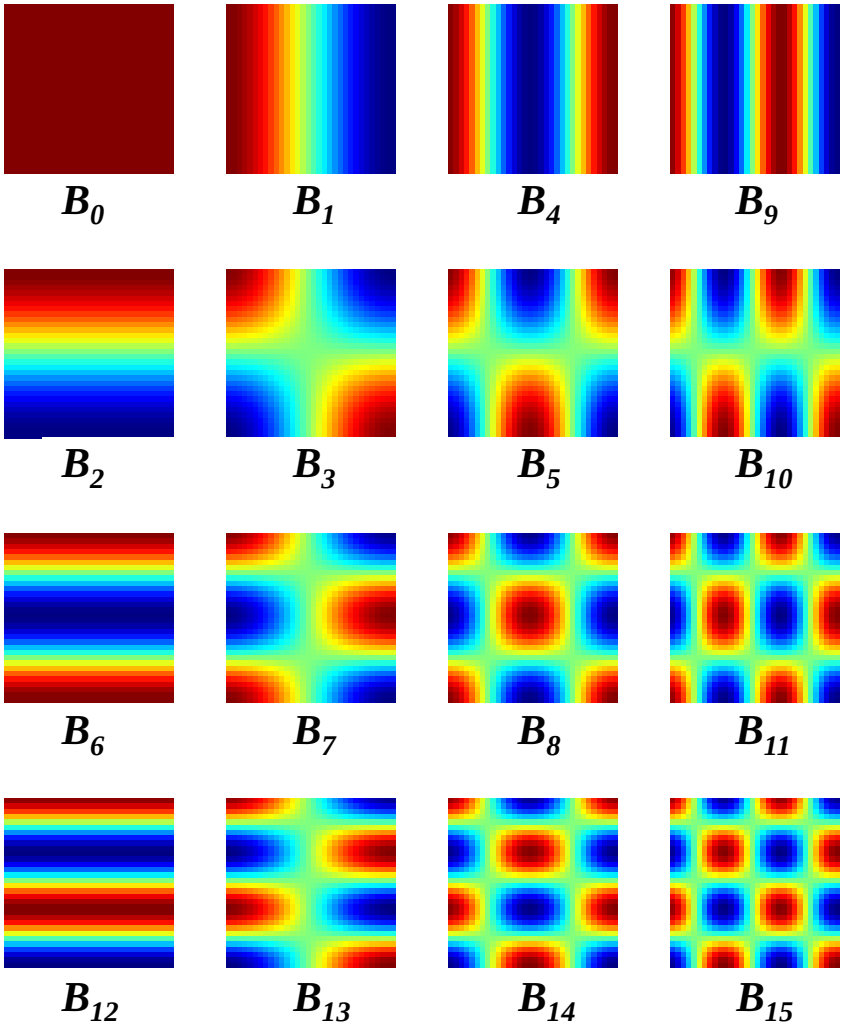
[Rueckert *et al.* 1998]

$$u_p(x) = \sum_k p_k B_k(x)$$

Trigonometric Bases – Fourier and Cosine Transforms

- **Type:**
  B_k are a set of orthogonal bases, based on trigonometric functions:
  - Discrete Fourier Transform (DFT)
  - Discrete Cosine Transform (DCT)
- **Location:**
  global,
  not localized
- **Support:**
  global





[Amit 1994], [Ashburner and Friston 1999], [Christensen and Johnson 2001]

$$u_p(x) = \sum_k p_k B_k(x)$$

Radial Basis Functions (RBF)

- **Type:**
  any RBF (value function of distance from control point), e.g.
  - "Adaptive Bases" [Rohde *et al.* 2003]:

    $$B_k(x) = [\max(1 - r, 0)]^4 \, (3r^3 +^2 r^2 + 6r + 4)$$

    $$\text{with } r = \|x - c_k\|$$

  - Gaussian
  - Thin-plate Splines (TPS)

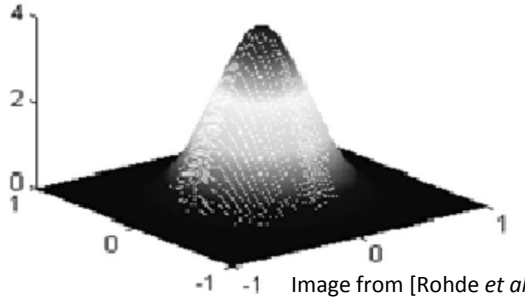
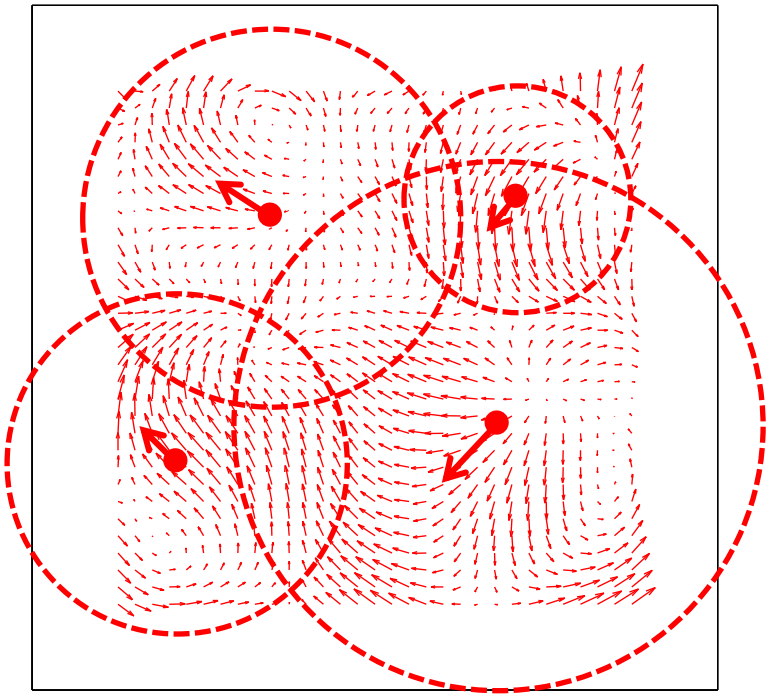


Image from [Rohde *et al.* 2003]

- **Location:**
  sparse, irregular:
  arbitrary location of control points
  → no regular grid



- **Support:**
  - In Rohde *et al.* 2003: compact
  - For TPS: global

Parametrization by Linear Models
Optimization by Gradient Descent

"Projection of dense updates into the parameter space."

Structure of the Difference Measure Gradient

$$\varphi = \mathrm{Id} + Bp \quad \Rightarrow \quad \frac{\mathrm{d}\varphi}{\mathrm{d}p_k} = B_k$$

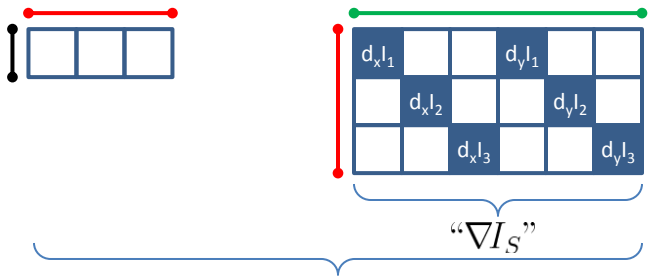
Vectorization of the involved entities:

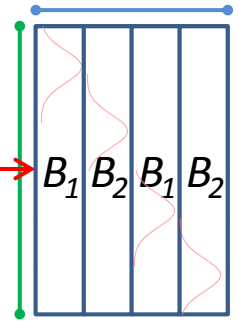
$$E_D = \square \qquad I_S = \begin{bmatrix}\square\\ \square\\ \square\end{bmatrix} \qquad \varphi = \begin{bmatrix}\square\\ \square\\ \square\\ \square\\ \square\\ \square\end{bmatrix} \begin{matrix}\varphi_x\\ \\ \\ \varphi_y\\ \\ \end{matrix} \qquad p = \begin{bmatrix}\square\\ \square\\ \square\\ \square\end{bmatrix} \begin{matrix}p_x\\ \\ p_y\\ \end{matrix}$$

$$\frac{\mathrm{d}E_D(I_S(\varphi(p)))}{\mathrm{d}p} = \frac{\mathrm{d}E_D}{\mathrm{d}I_S} \quad \frac{\mathrm{d}I_S}{\mathrm{d}\varphi} \quad \frac{\mathrm{d}\varphi}{\mathrm{d}p}$$

$$= \qquad \nabla f = \frac{\mathrm{d}f}{\mathrm{d}x}^{\top}$$

$$\nabla_p E_D =$$





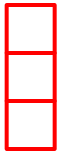
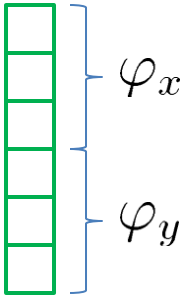
Dense Derivative: $\nabla E_D =$ "$W \cdot \nabla I_S$"

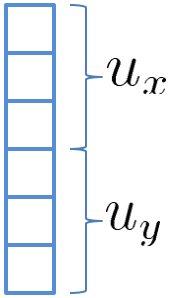
Projection of dense derivative into parameter space by $\nabla_p E_{D_i} = \langle \nabla E_D, B_i \rangle$.

Side Note: A Parallel to the Classic Variational Approach

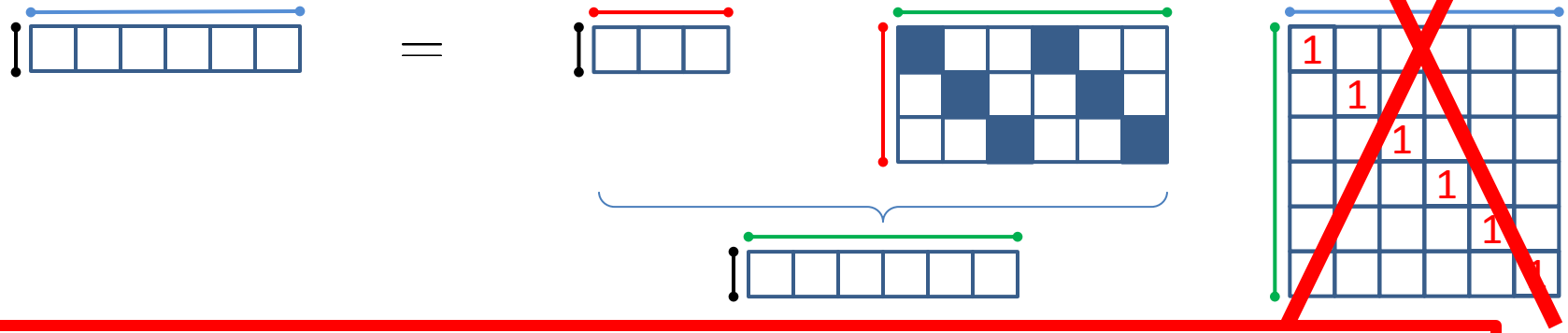
$$p = u$$

Vectorization of the involved entities:

$E_D = \square$ $\quad I_S =$  $\quad \varphi =$ 

$p = u =$ 

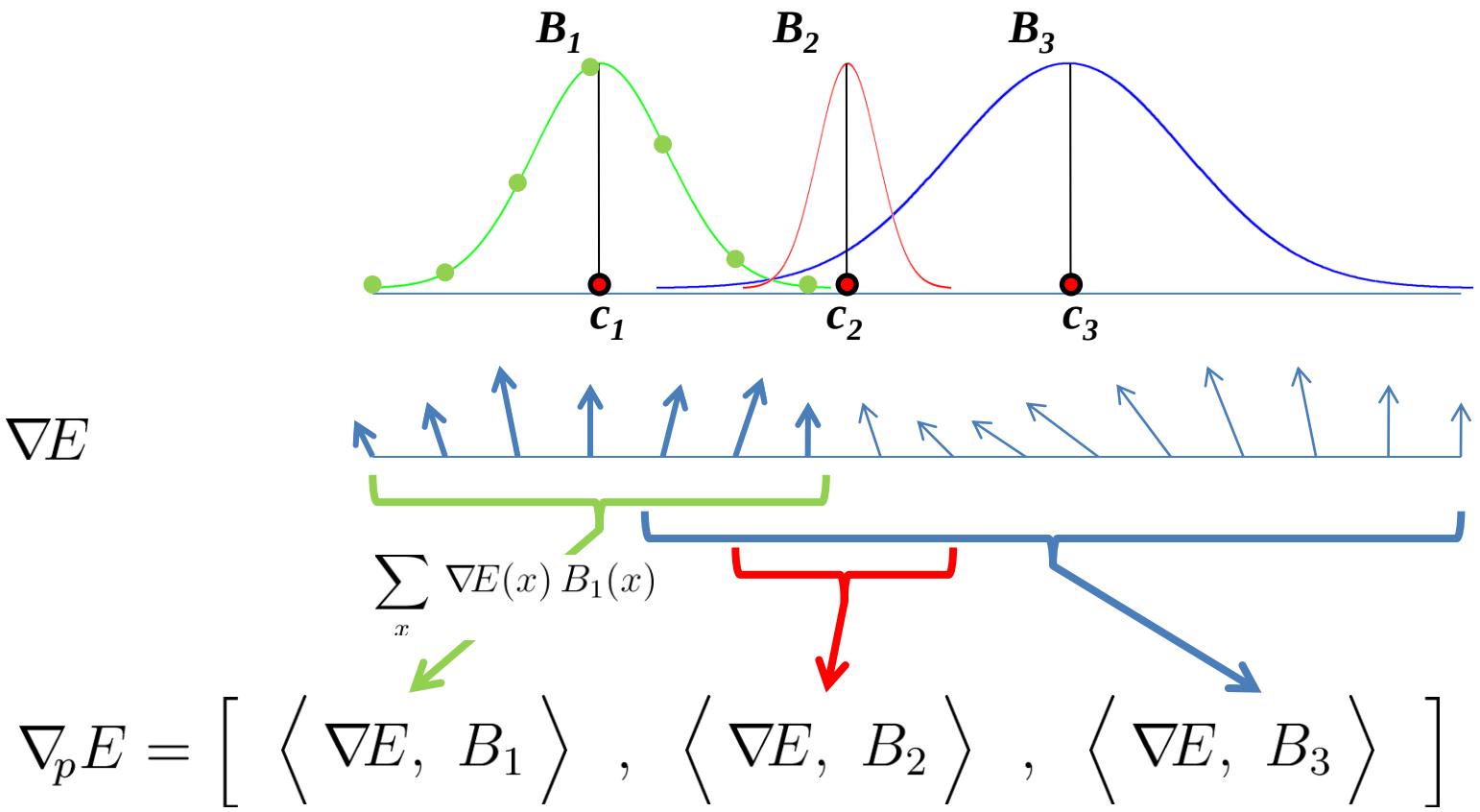
$$\frac{\mathrm{d}E_D(I_S(\varphi(p)))}{\mathrm{d}p} = \frac{\mathrm{d}E_D}{\mathrm{d}I_S}\frac{\mathrm{d}I_S}{\mathrm{d}\varphi}\frac{\mathrm{d}\varphi}{\mathrm{d}p}$$



→ The classic variational approach can be seen as parameterized by $B = \mathrm{Id}$

Visualization of Projection into Parameter Space

Illustration of $\nabla_p E = \nabla E \, \frac{\partial \varphi}{\partial p}$



Side note regarding intuition: If the B_k's have the same shape, then $\nabla_p E$ can be seen as a kind of a "generalized discrete convolution" of ∇E and B.

In the above we concentrated on the difference term...

- However, in the same way, we yield for the regularization (by applying the chain rule)

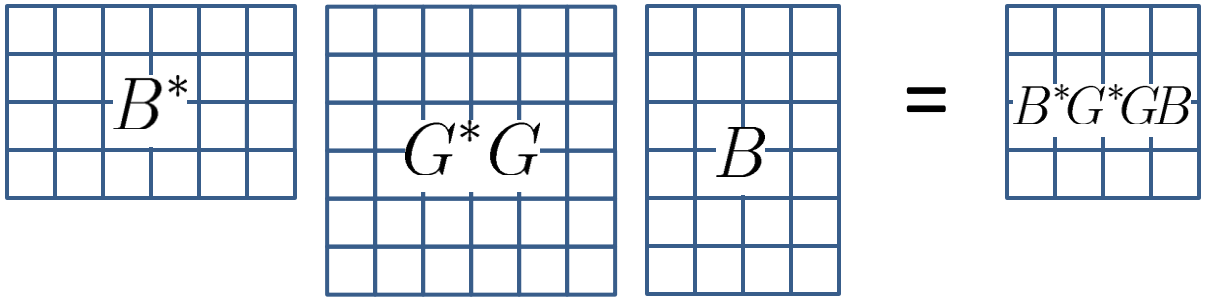
$$\nabla_p E_{R_i} = \langle \nabla E_R, B_i \rangle$$

- While this is formally correct, there are some interesting aspects of parametrization w.r.t. regularization, which are not revealed by this formulation

Derivative of Regularization Term

$$
\begin{aligned}
E_R(p) &= \frac{1}{2}\langle e_R(u(p)), e_R(u(p))\rangle \\
&= \frac{1}{2}\langle Gu(p), Gu(p)\rangle \\
&= \frac{1}{2}\langle GBp, GBp\rangle \\
&= \frac{1}{2}\langle p, (GB)^*(GB)p\rangle \\
&= \frac{1}{2}\langle p, B^*G^*GBp\rangle
\end{aligned}
\quad \rightarrow \quad
\frac{\mathrm{d}E_R(p)}{\mathrm{d}p} = B^*G^*GB\ p = \underbrace{B^*G^*G}_{\nabla E_R B}\ \underbrace{u}_{Bp}
$$

Side note: Equivalence to previous derivation

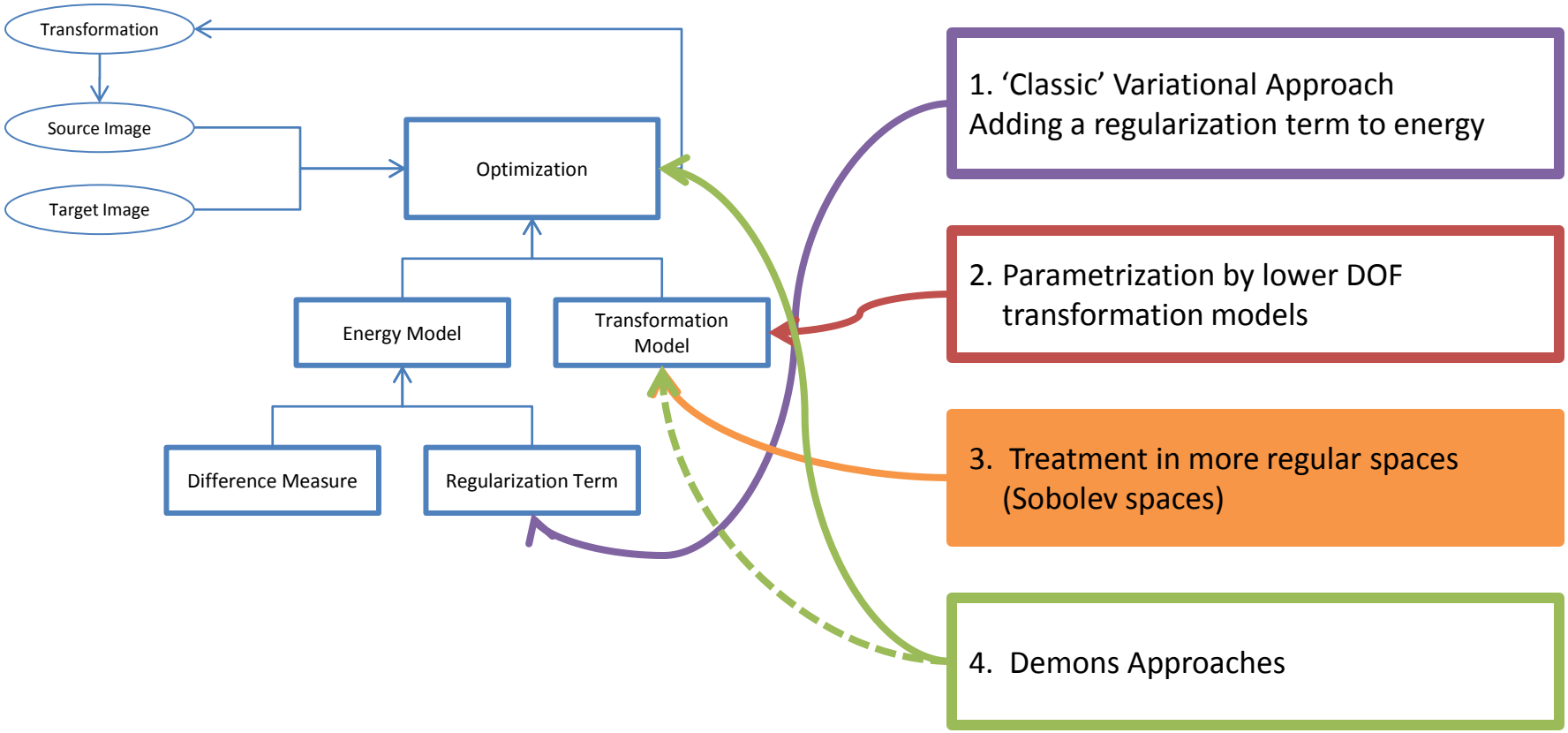


1. The resulting operator ***B*G*GB*** is much smaller than the original system ***G*G***
2. If the bases are eigenvectors to ***G*G***, ***B*G*GB*** becomes diagonal:
   for example for Diffusion Regularization and Trigonometric Bases
   (Fourier/Sine/Cosine - corresponding to boundary conditions)
   → **Efficient inversion becomes possible**

PART III

Minimization in Sobolev Spaces

Regularization Strategies



Registration in Sobolev Spaces (in 90 seconds)

- Minimization of energy E is not performed in L^2, but in a Sobolev space H^k
- Sobolev spaces contain only functions with certain regularity properties
- Sobolev space is defined by the scalar product

$$\langle u, u \rangle_{H^k} = \sum_{i=0}^{k} \lambda_k \langle u^{(i)}, u^{(i)} \rangle$$
$$= \langle \mathcal{L}u, \mathcal{L}u \rangle = \langle \mathcal{L}^* \mathcal{L} u, u \rangle$$

- Differential operator

$$\text{“}\mathcal{L} = [\mathrm{Id}, \nabla, \nabla^2, \ldots]^\top\text{”}$$

- Ultimately, employing Sobolev spaces leads to updates based on the Sobolev gradient

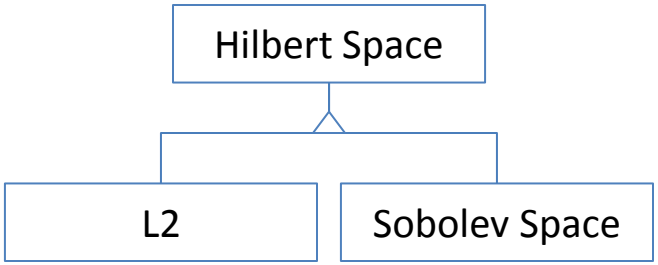
$$\nabla_{H^k} E = (\mathcal{L}^* \mathcal{L})^{-1} \nabla E$$

- Notes:
  - that the Sobolev gradient is based on the L^2 gradient
  - due to inherent displacement regularity, explicit regularization terms in energy model are not necessary (and mostly not employed)

Some Background

Some Background: Hilbert Space H

A Hilbert space H is a real (or complex) inner product space that is also a complete metric space with respect to the distance function induced by the inner product.



- A Hilbert space H features a **scalar product** $\langle u, u \rangle_{\mathrm{H}}$

→ Scalar product induces the **norm**, e.g. $\|u\|_{\mathrm{H}} = \sqrt{\langle u, u \rangle_{\mathrm{H}}}$

→ Scalar product induces the **distance**, e.g. $\mathrm{d}(u, h) = \|u - h\|_{\mathrm{H}}$

→ Scalar product defines the **gradient** $\partial_h E = \langle \nabla_{\mathrm{H}} E \ , \ h \ \rangle_{\mathrm{H}}$

$\partial_h f$: derivative in direction h

L^2 – Space:

Space of square-integrable functions

- Scalar product $\langle\, u\ ,\ h\,\rangle_{L^2} = \int_\Omega u(x)h(x)\ \mathrm{d}x$

- Norm $\|u\|_{L^2} = \sqrt{\langle\, u\ ,\ u\,\rangle_{L^2}}$

$$= \left(\int_\Omega u(x)^2\ \mathrm{d}x\right)^{\frac{1}{2}}$$

- Definition of L^2:

$$L^2 \equiv \{\ u\ :\ \|u\|_{L^2} < \infty\ \}$$

Sobolev Spaces H^s

- Function spaces equipped with a norm that is a combination of L2-norms of the function itself as well as its derivatives up to a given order.

$$\langle\ u\ ,\ h\ \rangle_{\mathrm{H}^s} = \sum_{k=0}^{s} \langle\ \mathrm{D}^k u\ ,\ \mathrm{D}^k h\ \rangle_{\mathrm{L}^2}$$

e.g. for s=1

$$\langle u, h \rangle_{\mathrm{H}^1} = \langle u, h \rangle_{\mathrm{L}^2} + \langle \nabla_{\mathrm{L}^2} u, \nabla_{\mathrm{L}^2} h \rangle_{\mathrm{L}^2}$$

Sobolev Spaces H^s

- In contrast to L2, H^s bound not only the functions, but also their derivatives

$$\mathrm{H}^s \equiv \{\ u\ :\ \|u\|_{\mathrm{H}^s} < \infty\ \}$$

$$\text{with}\quad \|u\|_{\mathrm{H}^s} = \sqrt{\langle\ u\ ,\ u\ \rangle_{\mathrm{H}^s}} = \left[\sum_{k=0}^{s} \langle\ \mathrm{D}^k u\ ,\ \mathrm{D}^k u\ \rangle_{\mathrm{L}^2}\right]^{\frac{1}{2}}$$

This provides a restriction to more regular functions

Gradient Descent in Hilbert Spaces

Starting with φ_0, loop until convergence:

$$h = -\tau \nabla_{\mathrm{H}} E(u_i)$$

$$\varphi_i = \varphi_i + h$$

The difference between treatment in the different spaces consists in the use of a different gradient.

Computation of Gradients in Hilbert Spaces based on a Metric Tensor

For a space H defined by

$$\langle\, u \;,\, h \,\rangle_{\mathrm{H}} = \langle\, \boxed{M} u \;,\, h \,\rangle_{\mathrm{L}^2}$$

the gradient is defined by

$$\nabla_{\mathrm{H}} E = \boxed{M^{-1}} \nabla_{\mathrm{L}^2} E$$

Sobolev Gradients [Neuberger 1997]

with $\mathcal{L} = \left(\mathrm{D}^0, \ldots, \mathrm{D}^s\right)_s$, we get

$$\begin{aligned}
\langle\, u \,,\, h \,\rangle_{\mathrm{H}^s} &= \sum_{k=0} \langle\, \mathrm{D}^k u \,,\, \mathrm{D}^k h \,\rangle_{\mathrm{L}^2} \\
&= \langle\, \mathcal{L}u \,,\, \mathcal{L}h \,\rangle_{\mathrm{L}^2} \\
&= \langle\, u \,,\, \mathcal{L}^*\mathcal{L}\, h \,\rangle_{\mathrm{L}^2}
\end{aligned}$$

and for the gradient

$$\nabla_{\mathrm{H}^s} E = \left(\mathcal{L}^*\mathcal{L}\right)^{-1} \nabla_{\mathrm{L}^2} E$$

Sobolev Gradients Example: H^1

with $\mathcal{L} = \left(\mathrm{D}^0, \ldots, \mathrm{D}^s\right)$, we have

$$\mathcal{L}^*\mathcal{L} = \sum_{i=0}^{s} (-1)^i \ \Delta^i$$

So that for H^1, we get

$$\mathcal{L}^*\mathcal{L} = \mathrm{Id} - \Delta$$

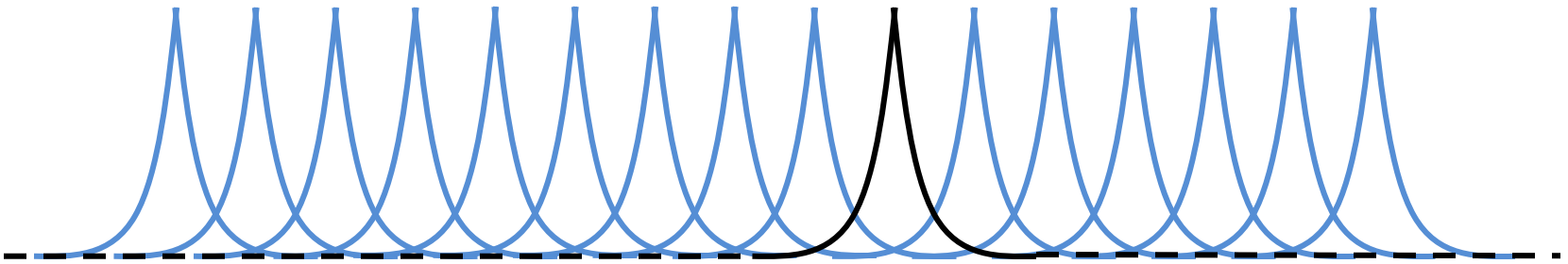
and

$$\begin{aligned} \nabla_{\mathrm{H}^1} E &= \left(\mathcal{L}^*\mathcal{L}\right)^{-1} \ \nabla_{\mathrm{L}^2} E \\ &= \left(\mathrm{Id} - \Delta\right)^{-1} \ \nabla_{\mathrm{L}^2} E \end{aligned}$$

Intuition about Sobolev Spaces

Operating with a transformation model using a dense parametrization based on the Green's function to the Sobolev operator

$$\mathcal{L}^*\mathcal{L} = \mathrm{Id} - \Delta$$



Please note the relation to parametric approaches, cf. e.g. [Zikic 2010 WBIR].

Deformable Registration in Sobolev Spaces

Deformable Registration in Sobolev Spaces

(Trouvé 1998, Chefd'hotel 2002, 2005)

$$u' = \arg\min_u E_D\Big(I_T, I_S \circ (\mathrm{Id} + u)\Big) \quad \text{with } u \in \mathrm{H}^s$$

Loop until convergence:

$$h = -\tau \ (\mathcal{L}^*\mathcal{L})^{-1} \nabla E_D(u)$$

$$\varphi = \varphi \circ (\mathrm{Id} + h)$$

end

- IDEA: Use of Sobolev spaces to restrict the space of solutions to more regular functions
  → No explicit regularization needed
- Minimization of difference measure E_D only
  – allows larger deformations (fluid-type approach)

Deformable Registration in Sobolev Spaces

(Trouvé 1998, Chefd'hotel 2002, 2005)

$$\mathcal{L}^*\mathcal{L} = \sum_{i=0}^{s} (-1)^i \ \alpha_i \ \Delta^i$$

Choice #1: $\alpha_i = 1, \ \alpha_i = \lambda$

$$\mathcal{L}^*\mathcal{L} = \mathrm{Id} - \lambda\Delta$$

$$\nabla_{\mathrm{H}^1} E = (\mathrm{Id} - \lambda\Delta)^{-1} \ \nabla_{\mathrm{L}^2} E$$

Choice #2: $\alpha_i = \sigma^{2i}/(i!2^i), \ s = \infty$

$$\mathcal{L}^*\mathcal{L} = \sum_{i=0}^{\infty} (-1)^i \ \sigma^{2i}/(i!2^i) \ \Delta^i$$

$$\nabla_{\mathrm{H}^\infty} E = (\mathcal{L}^*\mathcal{L})^{-1} \ \nabla_{\mathrm{L}^2} E$$

$$= G_\sigma * \ \nabla_{\mathrm{L}^2} E$$

Large Deformations Diffeomorphic Metric Mapping (LDDMM)

[Trouvé 1995][Dupuis 1998][Beg 2005]

$$\min_{v} \left(\left\| I_S \circ \varphi^{-1} - I_T \right\|_{L^2}^2 + \int_0^1 \left\| v_t \right\|_H^2 \, dt \right)$$

E_D: Standard SSD term

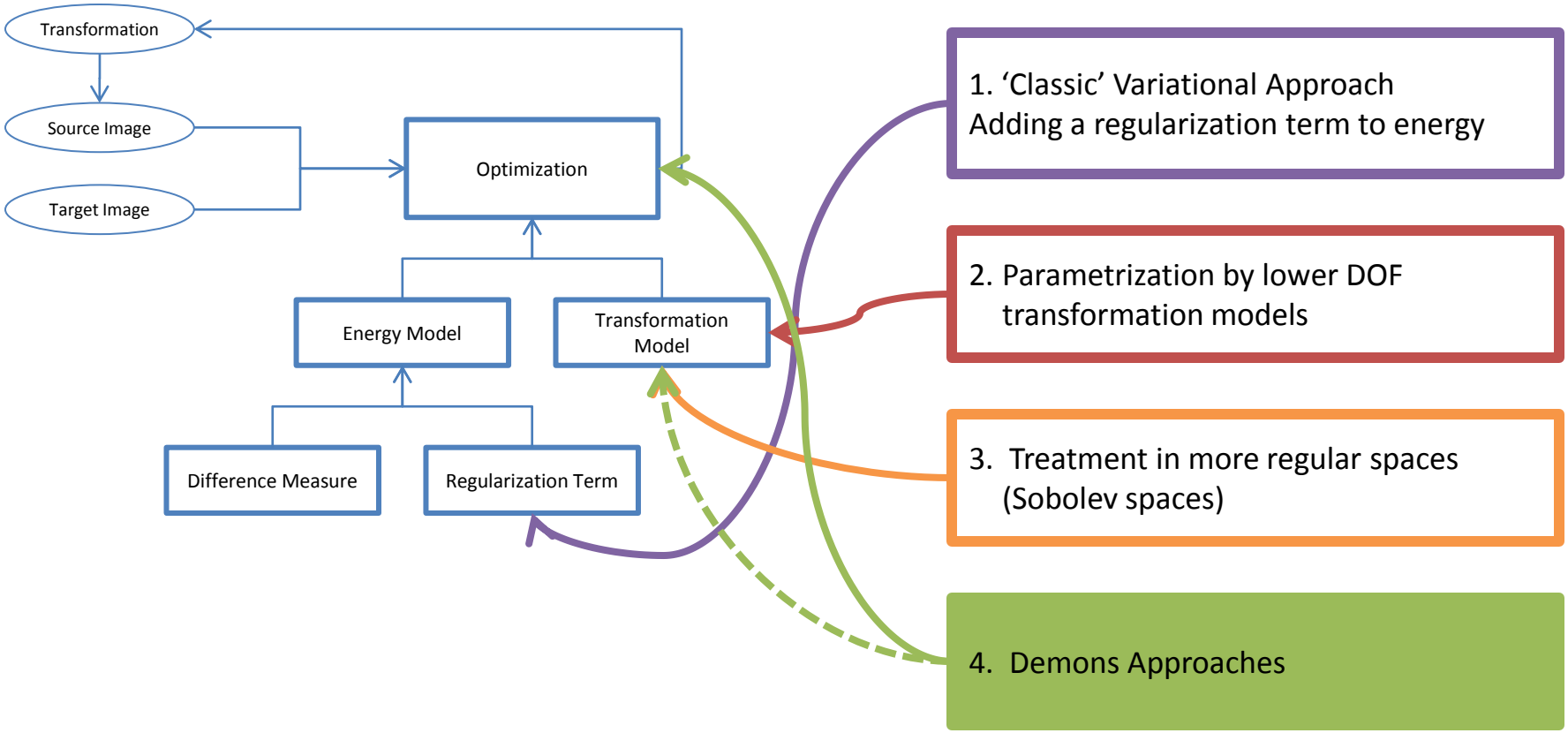
Regularization:
- Integration over velocities
  → enforces the geodesic shortest path in space of displacements
- Treatment of velocities in Sobolev space H

- Applied often in settings with very different anatomies (e.g. inter-patient brain)
- The integration of velocities
  - requires appropriate numerical schemes
  - Increases the computational requirements
    (memory for storage of velocity fields → results in longer computation)

PART IV

Demons Approaches

Regularization Strategies



Demons Registration

- Thirion 1996/1998: Heuristic approach, several alternatives
- Motivation: Speed
- Alternative 1 (the important one):

Loop until convergence:

$$h = \frac{(I_{\mathrm{T}} - I_{\mathrm{S}}(\varphi_u))\nabla I_{\mathrm{S}}(\varphi_u)}{\|\nabla I_{\mathrm{S}}(\varphi_u)\|^2 + (I_{\mathrm{T}} - I_{\mathrm{S}}(\varphi_u))^2}$$

$$u = u + \tau h$$

$$u = G^{\sigma} * u$$

end

Demons' Efficiency

- The preprocessing of forces:

$$h = \frac{(I_{\mathrm{T}} - I_{\mathrm{S}}(\varphi_u))\nabla I_{\mathrm{S}}(\varphi_u)}{\|\nabla I_{\mathrm{S}}(\varphi_u)\|^2 + (I_{\mathrm{T}} - I_{\mathrm{S}}(\varphi_u))^2} = \mathcal{F}(\nabla E_{\mathrm{D}})$$

  - Approximation to 2nd order optimization of E_{D} [Pennec 1999][Vercauteren 2009]
  - Approximative normalization of point-wise magnitudes of ∇E_{D}

- Efficient regularization by filtering:

$$u = G^{\sigma} * u$$

Elastic and Fluid Demons

Elastic-type Demons:
(original formulation)

Loop until convergence:

$$h = \mathcal{F}\left(\nabla E_{\mathrm{D}}\right)$$
$$u = u + \tau h$$
$$u = G^{\sigma} * u$$

end

Fluid-type Demons:

Loop until convergence:

$$h = \mathcal{F}\left(\nabla E_{\mathrm{D}}\right)$$
$$h = -G^{\sigma} * \tau h$$
$$u = u + h$$

end

[Bro-Nielsen 1996], [Pennec 1999]

Relations between Different Approaches

Demons fluid:

$$h = -G^{\sigma} * \tau \mathcal{F}\left(\nabla E_{\mathrm{D}}\right)$$

Demons elastic:

$$h = -G^{\sigma} * \tau \left(P^{-1} \nabla E_{\mathrm{D}} + \nabla E_{\mathrm{R}}\right)$$

Sobolev H^{∞}:

$$\mathcal{L}^{*}\mathcal{L} = \sum_{i=0}^{\infty} (-1)^{i} \, \sigma^{2i} / (i! 2^{i}) \, \Delta^{i}$$

$$\nabla_{\mathrm{H}^{\infty}} E = \left(\mathcal{L}^{*}\mathcal{L}\right)^{-1} \nabla E$$

$$= G_{\sigma} * \nabla E$$

PDE-Inspired, semi-implicit:

$$h = -\tau \left(\mathrm{Id} + \tau \lambda \nabla E_{R}\right)^{-1} \nabla E$$

for diffusion:

$$h = -\tau \left(\mathrm{Id} - \tau \lambda \Delta\right)^{-1} \nabla E$$

Sobolev H^{1}:

$$\mathcal{L}^{*}\mathcal{L} = \mathrm{Id} - \lambda \Delta$$

$$\nabla_{\mathrm{H}^{1}} E = \left(\mathrm{Id} - \lambda \Delta\right)^{-1} \nabla E$$

Gauß-Newton:

$$h = -\tau \left(J_{e}^{\top} J_{e}\right)^{-1} \nabla E$$

for SSD+diffusion:

$$h = -\tau \left(\nabla I_{\mathrm{S}} \nabla I_{\mathrm{S}}^{\top} - \lambda \Delta\right)^{-1} \nabla E$$

Preconditioned Descent:

$$h = -\tau \, P^{-1} \, \nabla E$$

END