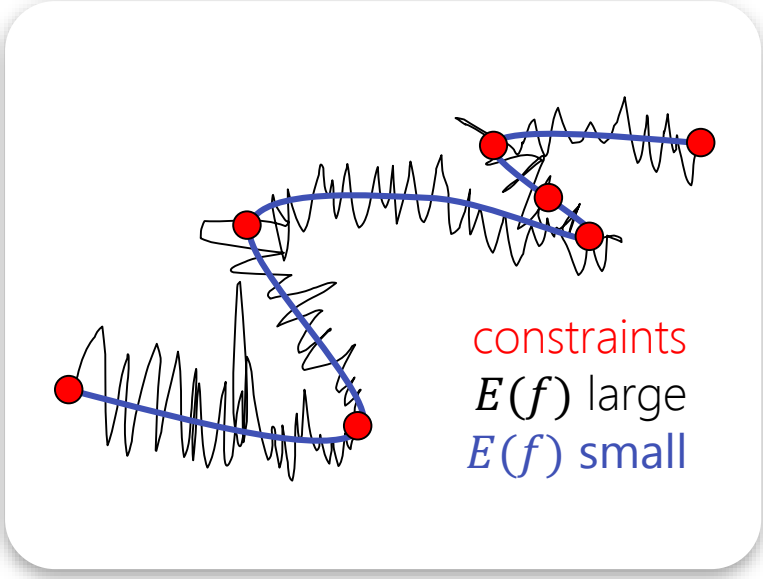
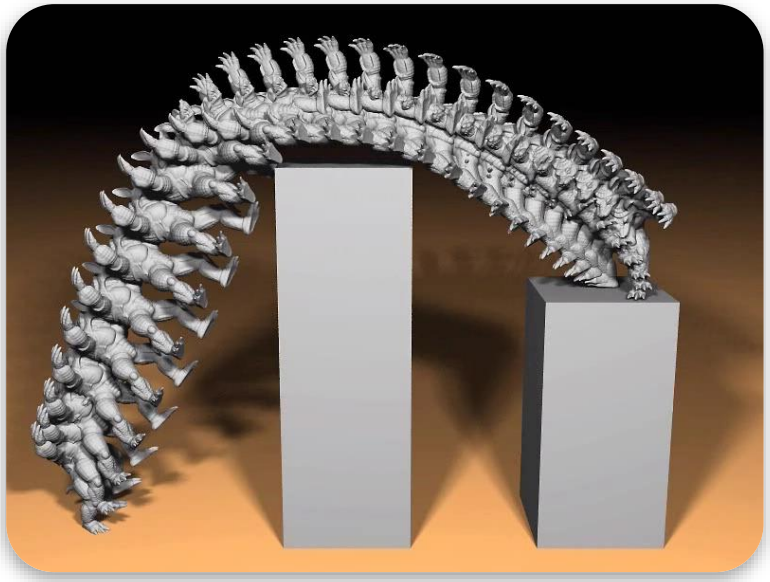
Modelling 1

SUMMER TERM 2020

JOHANNES GUTENBERG UNIVERSITÄT MAINZ





LECTURE 22

Advanced Variational Modeling

Michael Wand · Institut für Informatik · michael.wand@uni-mainz.de

Finite Elements for Functional Equations

We got a basis

Linear Ansatz

- We use a linear ansatz:

$$f(\mathbf{x}) \approx \tilde{f}(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i b_i(\mathbf{x})$$

- $\tilde{f}$ lives in a finite dimensional subspace
- Coordinates: $\lambda_1 \dots \lambda_n$

Linear Functional Equations

Abstract notation

$$Lf + g = 0$$

$$\forall x \in \mathcal{D}: Lf(x) + g(x) = 0$$

We cannot solve that

- Infinitely many conditions
- Subspace:
  extremely unlikely to permit exact solution

Least-Squares!

Abstract notation

$$Lf + g = 0$$

$$\forall x \in \mathcal{D}: Lf(x) + g(x) = 0$$

What to do instead?

Least-squares:

$$\int_{\mathcal{D}} \left(Lf(x) + g(x)\right)^2 dx \to min$$

...Least-Squares...

Least-Squares:

$$\int_{\mathcal{D}} \left(Lf(x) + g(x) \right)^2 dx$$

...Least-Squares...

Least-Squares:

$$\int_{\mathcal{D}} \big(Lf(x) + g(x)\big)^2 dx$$

$$= \int_{\mathcal{D}} \big(Lf(x)\big)^2 + 2\big(Lf(x)\big)g(x) + g(x)^2 dx$$

...Least-Squares...

Least-Squares:

$$\int_{\mathcal{D}} \big(Lf(x) + g(x)\big)^2 dx$$

$$= \int_{\mathcal{D}} \big(Lf(x)\big)^2 + 2\big(Lf(x)\big)g(x) + g(x)^2 dx$$

Now:

$$f(x) = \sum_{i=1}^{n} \lambda_i b_i(x)$$

...Least-Squares...

Least-Squares:

$$\int_{\mathcal{D}} \left[\left(\sum_{i=1}^{n} \lambda_i L b_i(x) \right)^2 + 2 \left(\sum_{i=1}^{n} \lambda_i L b_i(x) \right) g(x) + g(x)^2 \right] dx$$

...Least-Squares

Least-Squares:

$$\int_{\mathcal{D}} \left[\left(\sum_{i=1}^{n} \lambda_i L b_i(x) \right)^2 + 2 \left(\sum_{i=1}^{n} \lambda_i L b_i(x) \right) g(x) + g(x)^2 \right] dx$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j \int_{\mathcal{D}} L b_i(x) L b_j(x) dx + 2 \sum_{i=1}^{n} \lambda_i \int_{\mathcal{D}} L b_i(x) g(x) dx$$

$$+ \int_{\mathcal{D}} g^2(x) dx$$

Least-Squares Objective

Objective Function:

$$\boldsymbol{\lambda}^{\mathrm{T}}\,\mathbf{M}\,\boldsymbol{\lambda} + 2\mathbf{b}^{\mathrm{T}}\boldsymbol{\lambda} + c$$

with

$$\mathbf{M} = \begin{pmatrix} \ddots & & \iddots \\ & \int_{\mathcal{D}} Lb_i(x)\,Lb_j(x)\,dx & \\ \iddots & & \ddots \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} \vdots \\ \int_{\mathcal{D}} Lb_i(x)\,g(x)\,dx \\ \vdots \end{pmatrix}$$

$$c = \int_{\mathcal{D}} g^2(x)\,dx$$

Normal Equations

Objective Function:

$$\boldsymbol{\lambda}^{\mathrm{T}}\,\mathbf{M}\,\boldsymbol{\lambda} + 2\mathbf{b}^{\mathrm{T}}\boldsymbol{\lambda} + c$$

Minimization:

$$\nabla_{\boldsymbol{\lambda}}\left(\boldsymbol{\lambda}^{\mathrm{T}}\,\mathbf{M}\,\boldsymbol{\lambda} + 2\mathbf{b}^{\mathrm{T}}\boldsymbol{\lambda} + c\right) = 2\mathbf{M}\,\boldsymbol{\lambda} + 2\mathbf{b}$$

Solve:

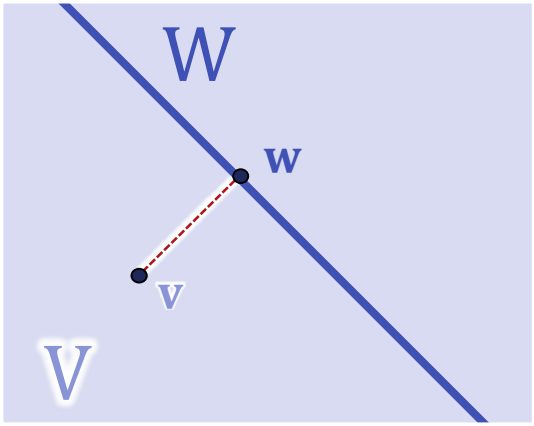
$$\mathbf{M}\,\boldsymbol{\lambda} = -\mathbf{b}$$

Alternative: Garlerkin Method

Abstract notation

$$Lf + g = 0$$

$$\forall x \in \mathcal{D}: Lf(x) + g(x)$$



Alternative option?

Residual perpendicular to subspace:

$$\forall i = 1..n: \left\langle L \sum_{i=1}^{n} \lambda_i b_i(x) + g(x), b_i(x) \right\rangle = 0$$

Alternative: Garlerkin Method

$$\left\langle L \sum_{i=1}^{n} \lambda_i b_i(x) + g(x), b_i(x) \right\rangle$$

$$= \sum_{i=1}^{n} \lambda_i \langle L b_i(x), b_i(x) \rangle + \langle g(x) b_i(x) \rangle$$

We obtain…

Solve:

$$\mathbf{M}\,\boldsymbol{\lambda} = -\mathbf{b}$$

with

$$\mathbf{M} = \begin{pmatrix} \ddots & & \iddots \\ & \int_{\mathcal{D}} L b_i(x) b_j(x) dx & \\ \iddots & & \ddots \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} \cdots & \int_{\mathcal{D}} b_i(x) g(x) dx & \cdots \end{pmatrix}$$

Difference to Least-Squares

Solve:

$$\mathbf{M}\,\boldsymbol{\lambda} = -\mathbf{b}$$

with

$$\mathbf{M} = \begin{pmatrix} \ddots & & \iddots \\ & \int_{\mathcal{D}} L b_i(x)\, b_j(x)\, dx & \\ \iddots & & \ddots \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} \cdots & \int_{\mathcal{D}} b_i(x)\, g(x)\, dx & \cdots \end{pmatrix}$$

(Same solution, but different system)

(Half-Serious) Example: Plugging in the Rendering Equation

(Least-squares)

Least-Squares Objective

$$\int_{S\times\Omega}\left[\left|\sum_{i=1}^{n}\lambda_i b_i(\mathbf{x},\mathbf{w}) - E_0(\mathbf{x},\mathbf{w}) - \int_{\mathbf{w}_1\in\Omega}\left(\sum_{i=1}^{n}\lambda_i b_i(\mathbf{x},\mathbf{w})\right)\cdot\rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2)\cdot\cos\ \theta_1 d\mathbf{w}_1\right|\right]^2 d\mathbf{w}\,d\mathbf{x}$$

$$= \int_{S\times\Omega}\left[\left|\sum_{i=1}^{n}\lambda_i b_i(\mathbf{x},\mathbf{w}) - E_0(\mathbf{x},\mathbf{w}) - \sum_{i=1}^{n}\lambda_i \int_{\mathbf{w}_1\in\Omega} b_i(\mathbf{x},\mathbf{w}_1)\cdot\rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2)\cdot\cos\ \theta_1 d\mathbf{w}_1\right|\right]^2$$

$$= \sum_{i=1}^{n}\sum_{i=1}^{n}\lambda_i\lambda_j \int_{S\times\Omega} b_i(\mathbf{x},\mathbf{w})b_j(\mathbf{x},\mathbf{w})\,d\mathbf{w}\,d\mathbf{x}$$

$$- \sum_{i=1}^{n}\lambda_i \int_{S\times\Omega} b_i(\mathbf{x},\mathbf{w})E_0(\mathbf{x},\mathbf{w})\,d\mathbf{w}\,d\mathbf{x}$$

$$- \sum_{j=1}^{n}\sum_{i=1}^{n}\lambda_i\lambda_j \int_{S\times\Omega} b_i(\mathbf{x},\mathbf{w})\cdot\int_{\mathbf{w}_1\in\Omega} b_j(\mathbf{x},\mathbf{w}_1)\cdot\rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2)\cdot\cos\ \theta_1 d\mathbf{w}_1 d\mathbf{w}\,d\mathbf{x}$$

$$- \sum_{i=1}^{n}\lambda_i \int_{S\times\Omega} b_i(\mathbf{x},\mathbf{w})\cdot E_0(\mathbf{x},\mathbf{w})\,d\mathbf{w}\,d\mathbf{x}$$

$$+ \int_{S\times\Omega} E_0(\mathbf{x},\mathbf{w})^2\,d\mathbf{w}\,d\mathbf{x}$$

$$+ \sum_{i=1}^{n}\lambda_i \int_{S\times\Omega} E_0(\mathbf{x},\mathbf{w})\cdot\int_{\mathbf{w}_1\in\Omega} b_i(\mathbf{x},\mathbf{w}_1)\cdot\rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2)\cdot\cos\ \theta_1 d\mathbf{w}_1 d\mathbf{w}\,d\mathbf{x}$$

$$- \sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_j\lambda_i \int_{S\times\Omega} b_i(\mathbf{x},\mathbf{w})\cdot\int_{\mathbf{w}_1\in\Omega} b_j(\mathbf{x},\mathbf{w}_1)\cdot\rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2)\cdot\cos\ \theta_1 d\mathbf{w}_1 d\mathbf{w}\,d\mathbf{x}$$

$$+ \sum_{i=1}^{n}\lambda_i \int_{S\times\Omega} E_0(\mathbf{x},\mathbf{w})\cdot\int_{\mathbf{w}_1\in\Omega} b_i(\mathbf{x},\mathbf{w}_1)\cdot\rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2)\cdot\cos\ \theta_1 d\mathbf{w}_1 d\mathbf{w}\,d\mathbf{x}$$

$$+ \sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j \int_{S\times\Omega}\int_{\mathbf{w}_1\in\Omega} b_i(\mathbf{x},\mathbf{w}_1)\cdot\rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2)\cdot\cos\ \theta_1 d\mathbf{w}_1\cdot\int_{\mathbf{w}_1\in\Omega} b_j(\mathbf{x},\mathbf{w}_1)\cdot\rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2)\cdot\cos\ \theta_1 d\mathbf{w}_1 d\mathbf{w}\,d\mathbf{x}$$

(there might be typos...)

Linear System

$$\mathbf{M}\,\boldsymbol{\lambda} = -\mathbf{b}$$

with:

$$m_{ij} = \int_{S\times\Omega} b_i(\mathbf{x},\mathbf{w})\, b_j(\mathbf{x},\mathbf{w})\, d\mathbf{w}\, d\mathbf{x}$$

$$- \int_{S\times\Omega} b_i(\mathbf{x},\mathbf{w}) \cdot \int_{\mathbf{w}_1\in\Omega} b_j(\mathbf{x},\mathbf{w}_1) \cdot \rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2) \cdot \cos\,\theta_1\, d\mathbf{w}_1\, d\mathbf{w}\, d\mathbf{x}$$

$$- \int_{S\times\Omega} b_i(\mathbf{x},\mathbf{w}) \cdot \int_{\mathbf{w}_1\in\Omega} b_j(\mathbf{x},\mathbf{w}_1) \cdot \rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2) \cdot \cos\,\theta_1\, d\mathbf{w}_1\, d\mathbf{w}\, d\mathbf{x}$$

$$+ \int_{S\times\Omega} \int_{\mathbf{w}_1\in\Omega} b_i(\mathbf{x},\mathbf{w}_1) \cdot \rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2) \cdot \cos\,\theta_1\, d\mathbf{w}_1 \cdot \int_{\mathbf{w}_1\in\Omega} b_j(\mathbf{x},\mathbf{w}_1) \cdot \rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2) \cdot \cos\,\theta_1\, d\mathbf{w}_1\, d\mathbf{w}\, d\mathbf{x}$$

$$b_i = 2 \int_{S\times\Omega} E_0(\mathbf{x},\mathbf{w}) \cdot \int_{\mathbf{w}_1\in\Omega} b_i(\mathbf{x},\mathbf{w}_1) \cdot \rho_{\mathbf{x}}(\mathbf{w}_1,\mathbf{w}_2) \cdot \cos\,\theta_1\, d\mathbf{w}_1\, d\mathbf{w}\, d\mathbf{x} - \int_{S\times\Omega} b_i(\mathbf{x},\mathbf{w})\, E_0(\mathbf{x},\mathbf{w})\, d\mathbf{w}\, d\mathbf{x}$$

Some
Omitted Aspects

Further Topics

More things to be considered:

- Designing “good” basis functions
- Condition of the resulting linear system
- High-order differentials
  - Distinguish test and basis functions
  - Order can then be reduced by partial integration
    - Split order for the two sets
    - Makes numerics & algebra easier

Numerical Aspects

Solving the Linear System

Quadratic energy and quadratic constraints

- Discretization: multivariate quadratic polynomial
  - Gradient: linear expression.
- Linear system properties
  - Symmetric matrix
  - Positive (semi-)definite
  - Usually sparse
    - coefficients of basis functions only interact with neighbors (overlapping support).
  - Iterative sparse system solvers
    - For example, conjugate gradients (SPD matrix)
    - CG is available in GeoX, SciPy / NumPy, etc.

Non-Linear Problems

Type 1: Convex Problems

- (Theoretically) always solvable
- Only one global minimum

Type 2: Non-Convex Problems

- Might be very hard
- 2a: PCA – non-convex, but solvable exactly
- 2b: Non-convex but easy
  - Few minima, or equivalent minima, or good initialization
- 2c: The rest – this is difficult

Solving Non-Linear Systems

Convex functionals

- Sufficient conditions for being able to find a *global minimum*:
  - Convex domain $\Omega \subset \mathbb{R}^n$
  - Convex function $f: \Omega \to \mathbb{R}$
    - Can be checked by $\mathbf{H}_f \geq \mathbf{0}$ for $f \in C^2$
  - Numerical descent (in principle) finds global optimum

Quadratic functions

- Unconstrained quadratic optimization is convex
- Linear system

…for “easy” problems…

Descent-Based Non-Linear Solvers

1st Order: Gradient Descent

Gradient Descent:

- ∇E = direction of steepest ascent
  - Take small steps in direction $-\nabla E$
  - When $\nabla E = \mathbf{0}$, a critical point is found.
- Small enough steps guarantee convergence
  - In theory
  - In practice: usually slow, unstable
  - Does not work for ill-conditioned problems

Line Search

Gradient descent line search

- Step size for gradient descent
  - Fit 1D parabola to *E* in gradient direction
  - Perform 1D Newton search
  - If E does not decrease at the new position
    - Try to half step width (say up to 10-20 times).
    - If this still does not decrease *E*, stop and output local minimum.

2nd Order Non-Linear Solvers

Newton optimization

- Iteratively solve linear problems
- 2nd order Taylor expansions. Requires:
  - Function values
  - Gradient
  - Hessian matrix
- Typically, Hessian matrices are sparse.
  - Should be SPD (otherwise: trouble)
- Use conjugate gradients to solve for critical points

Newton Optimization

Newton Optimization

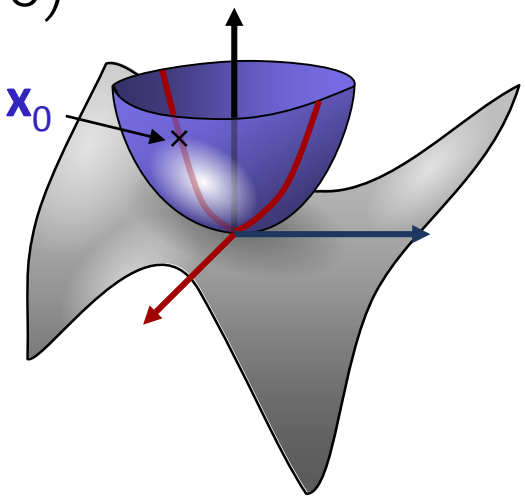
- Basic idea: Local quadratic approximation of E:

$$E(\mathbf{x}) \approx E(\mathbf{x}_0) + \nabla E(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^{\mathrm{T}} \cdot H_E(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0)$$

- Solve for vertex (critical point) of the fitted parabola
- Iterate until a minimum is found ($\nabla E = 0$)

Properties:

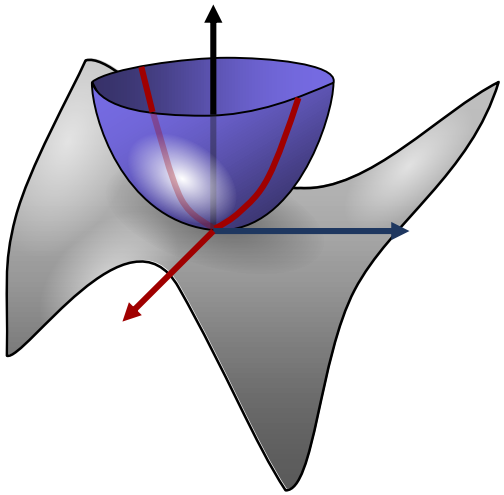
- Typically much faster convergence, more stable
- No convergence guarantee



Newton Line Search

Line search for Newton-optimization:

- Following the quadratic fit might overshoot
- Line search:
  - Test value of E at new position
  - Half step width until error decreases (say 10-20 iterations)
  - Switch to gradient descent, if this does not work



Newton Optimization

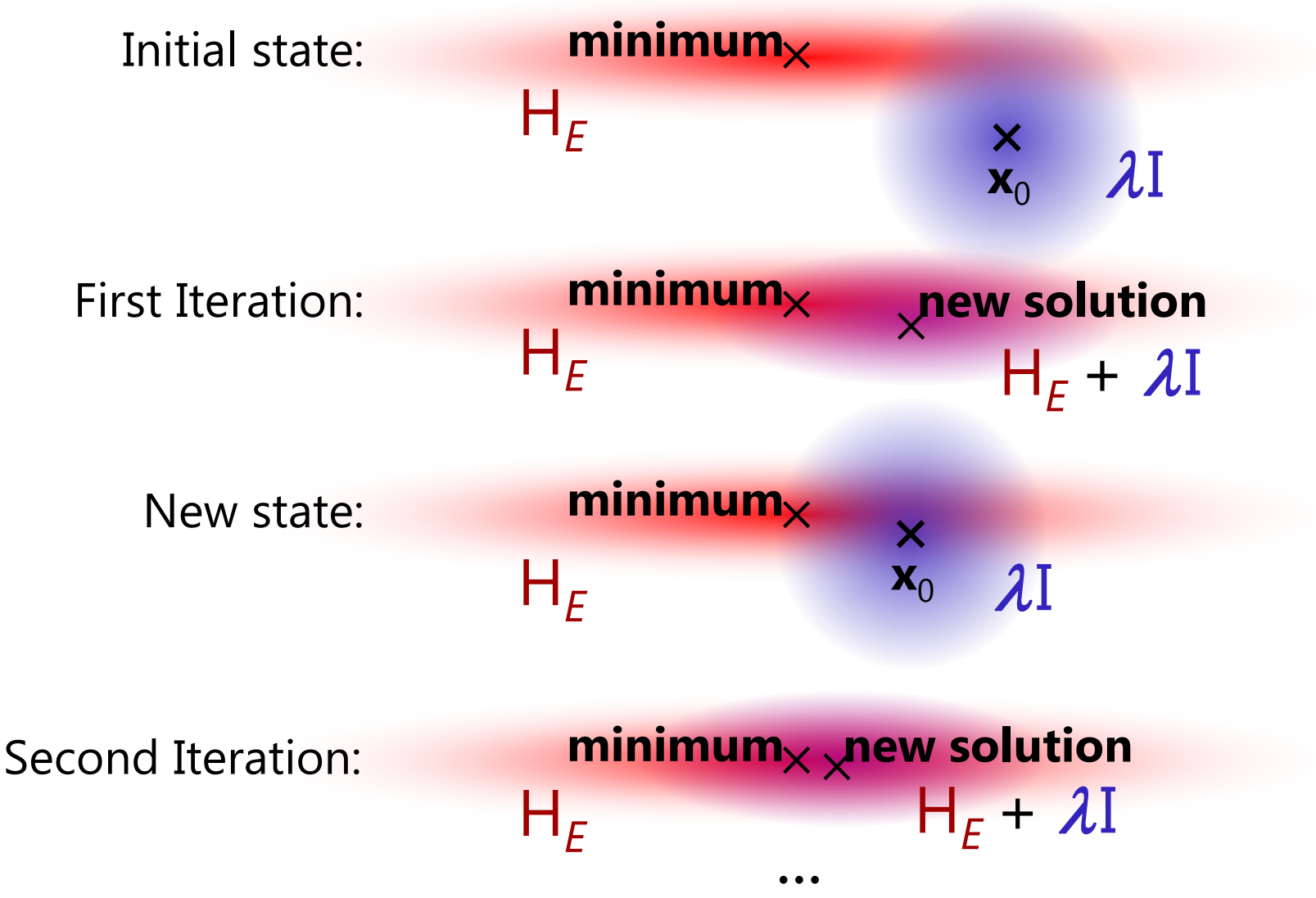
Problem:

- Steps might go uphill
- (Near-) zero or negative eigenvalues make problem ill-conditioned.

Simple solution

- Add $\lambda \mathbf{I}$ to the Hessian for a small λ.
- Sum of two quadrics: $\lambda \mathbf{I}$ keeps solution at $\mathbf{x}_0$.
- Comprehensive method: Levenberg-Marquant

Handling Indefinite Situations



What if I Hate Deriving the Hessian?

Gauss Newton

$$E(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})^2 \Rightarrow \tilde{E}(\mathbf{x}) = \sum_{i=1}^{n} \left(\nabla f_i(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + f_i(\mathbf{x}_0)\right)^2$$

LBFGS

- "Quasi-Newton" method
- "Black box-solver"
  - Needs only gradient + function values

Non-linear conjugate gradients:

- With line search
- Usually faster than simple gradient decent

“Easy” Non-Linear Problem


ground truth

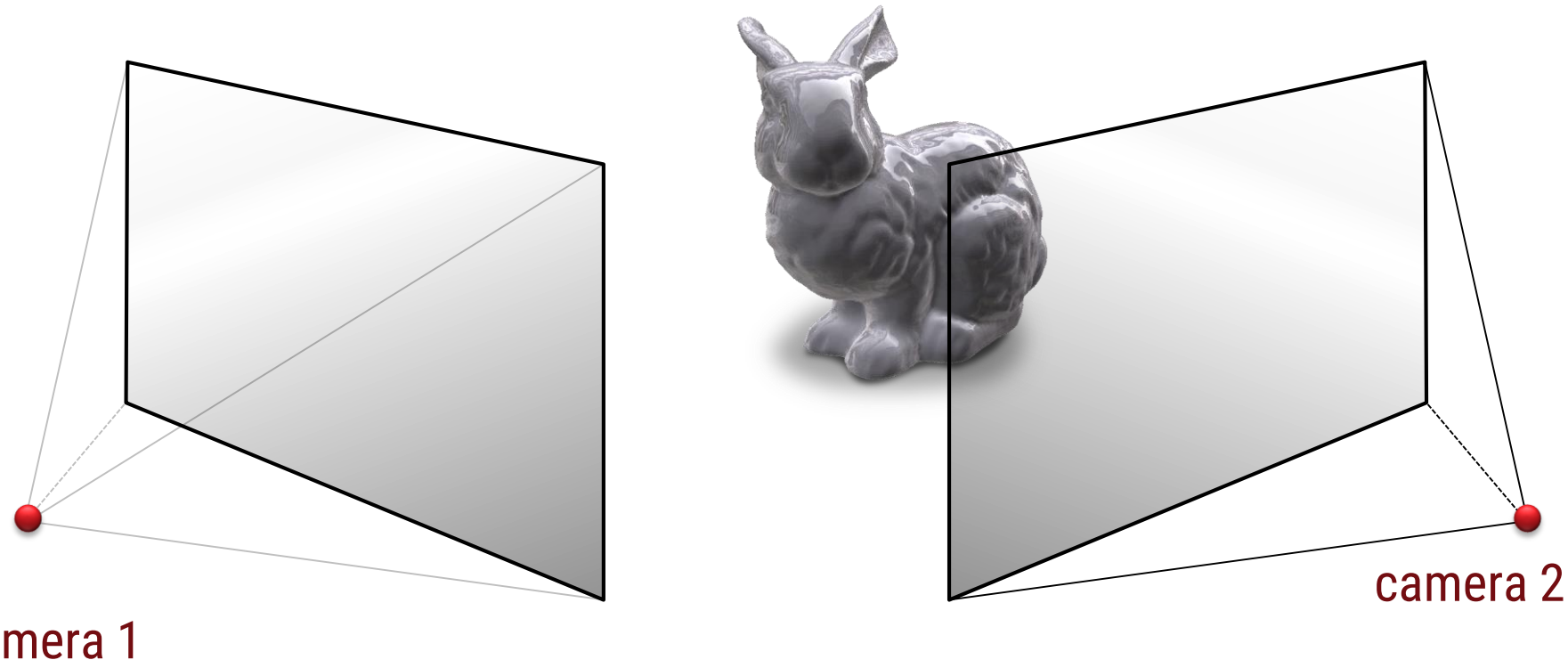

reconstruction (Gaussian prior)


input (noisy)

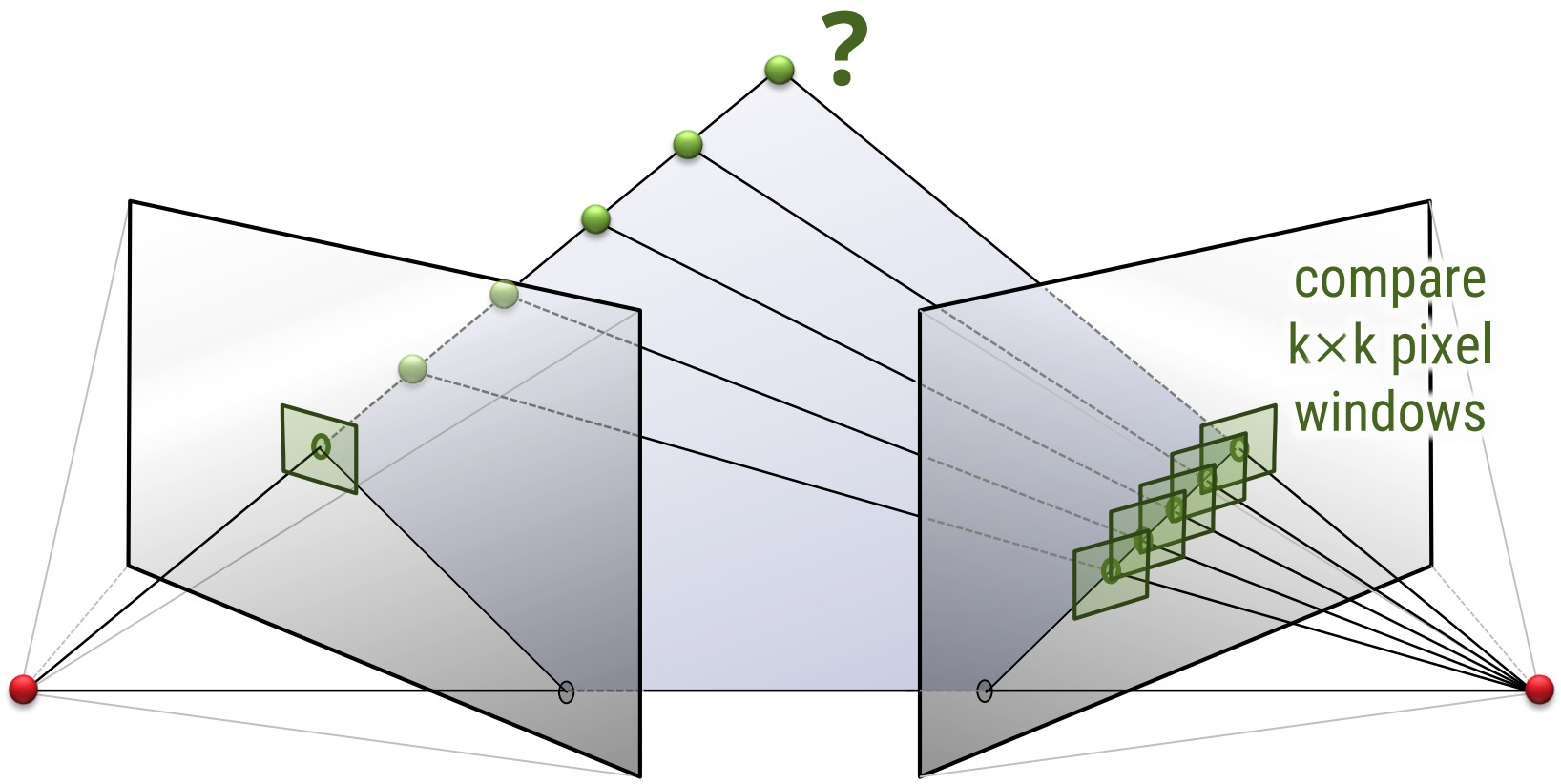

reconstruction (Laplace prior)

“Hard” Non-Linear Problem

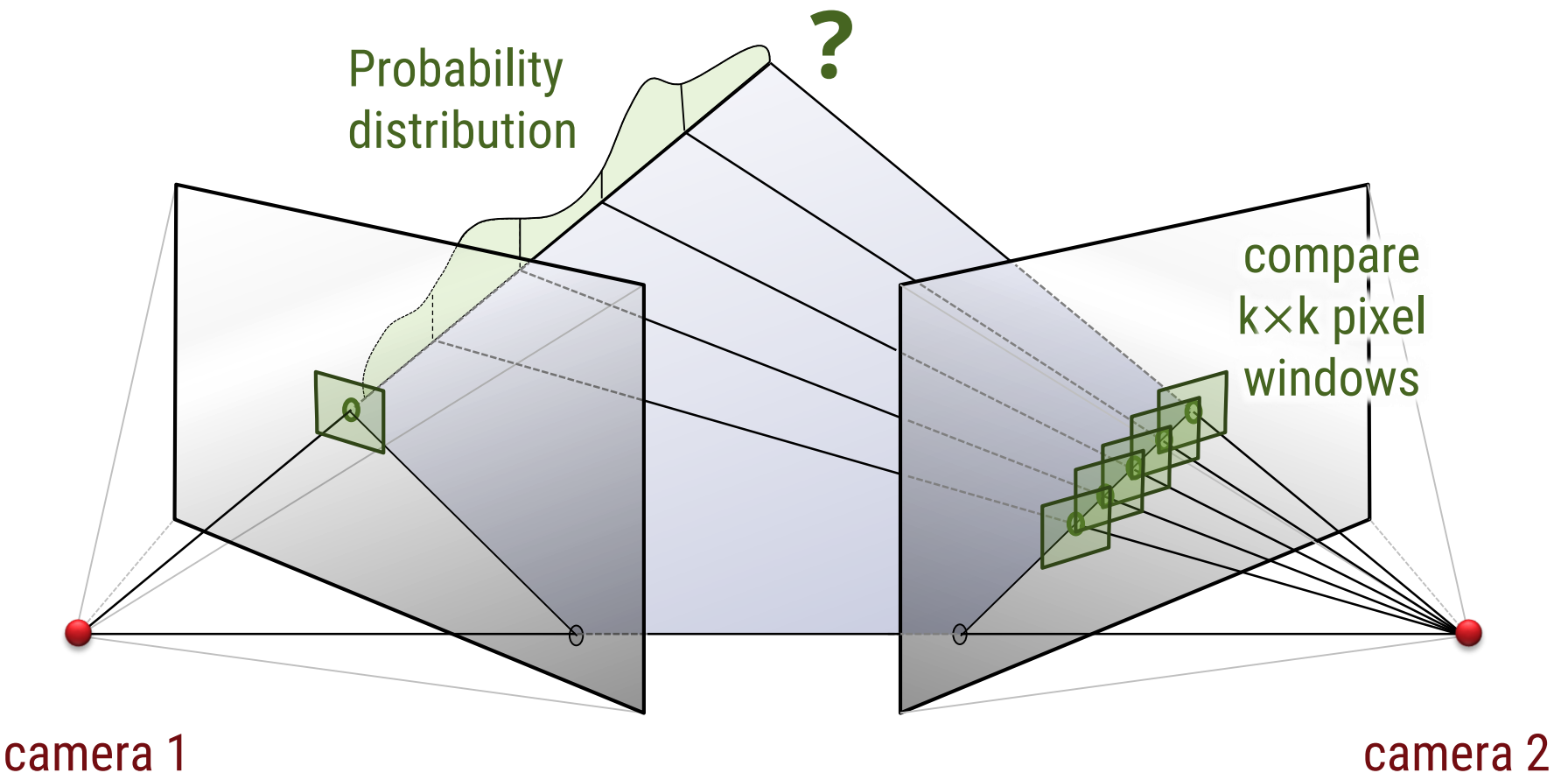
Difficult Problem: Stereo Vision



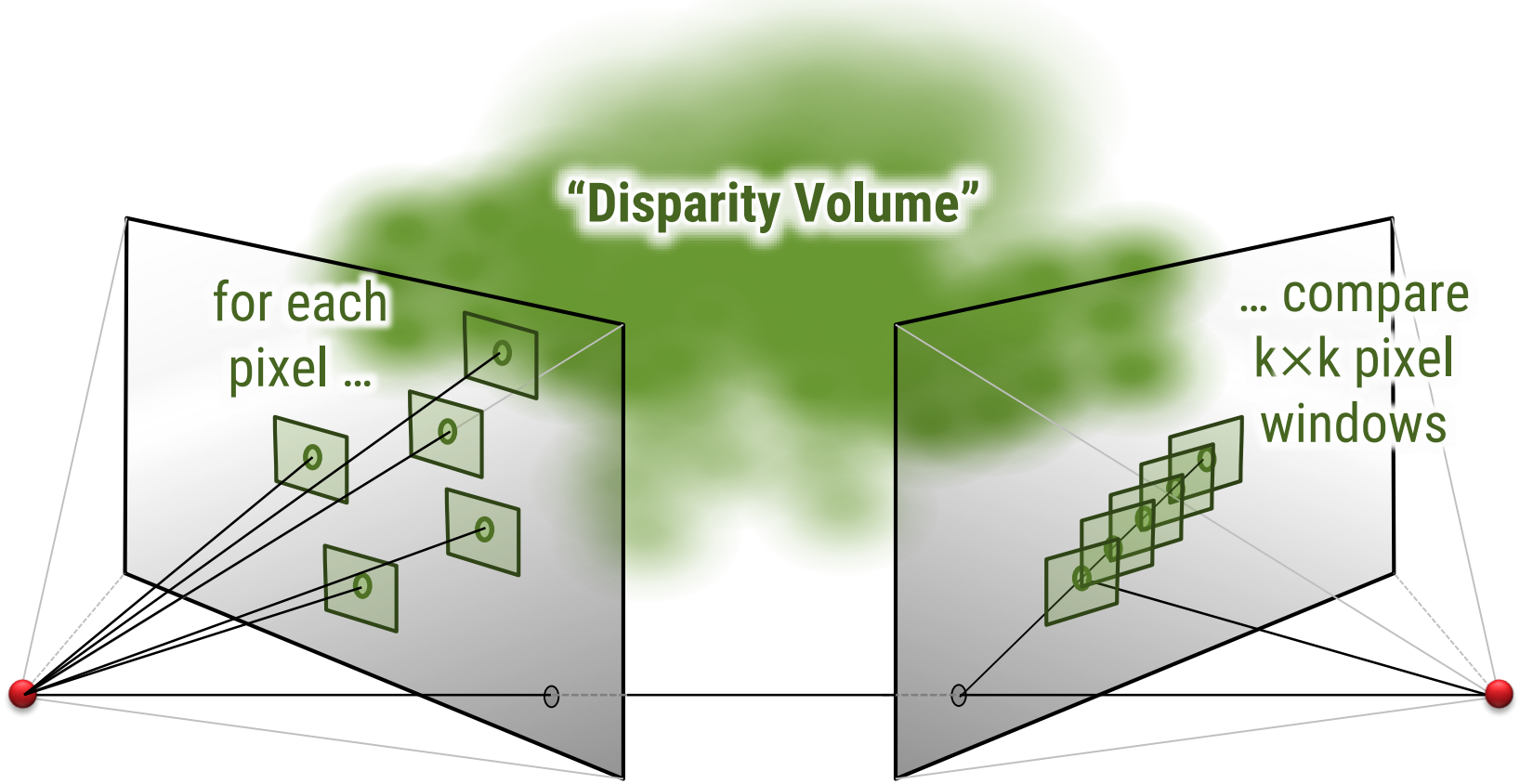
Stereo Vision



Stereo Vision



Stereo Vision



“Disparity Volume”

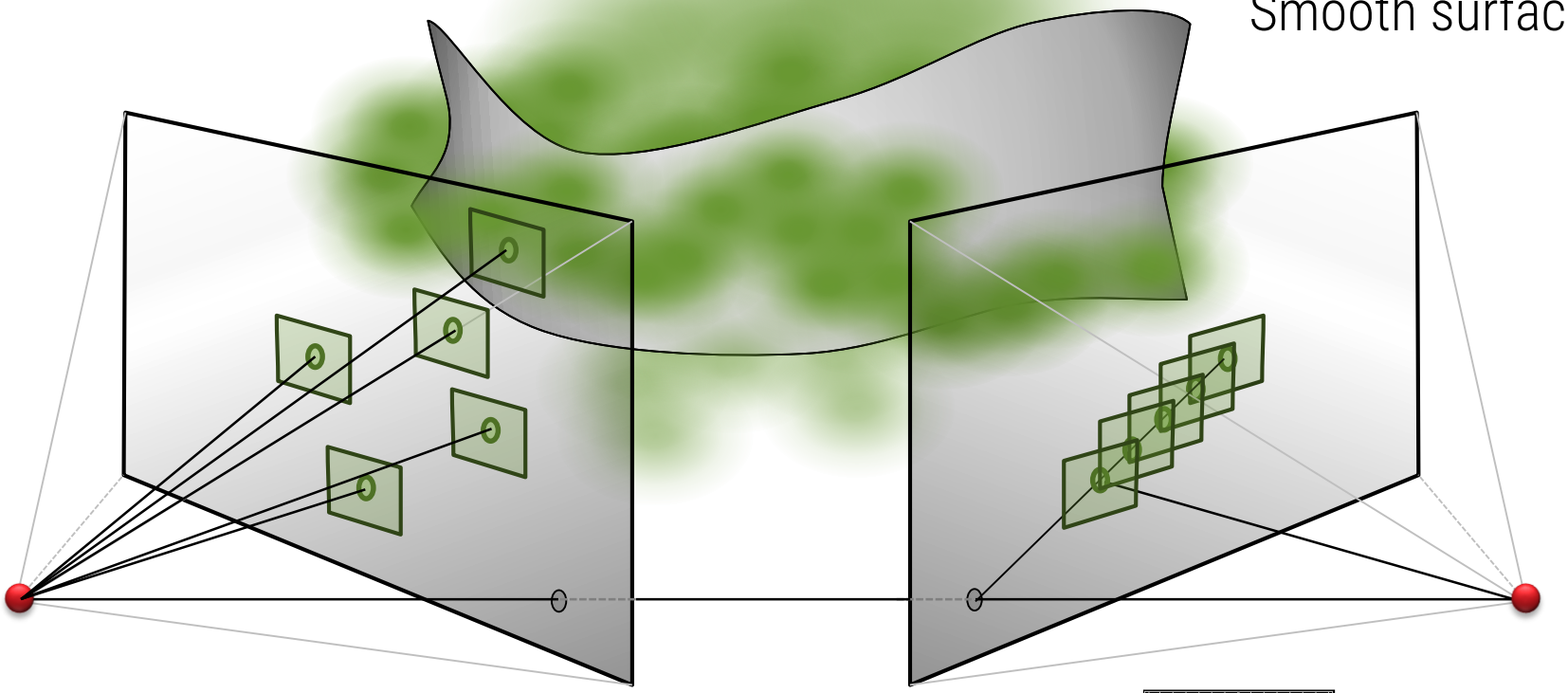
Approximation:

statistical dependencies neglected

(for example: occlusion)

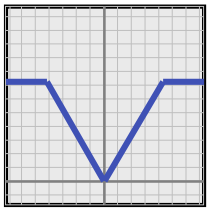
Stereo Vision

Assumption:
Smooth surface



camera 1

camera 2

$\text{pot}(\cdot)$ 

$$E(X|D) = \sum_{i=1}^{w}\sum_{j=1}^{h} \text{match}(x_i) + \sum_{i=1}^{w-1}\sum_{j=1}^{h-1} \text{pot}\left(\sqrt{\left(x_{i+1,j} - x_{i,j}\right)^2 + \left(x_{i,j+1} - x_{i,j}\right)^2}\right)$$

Hard Constraints

Hard Constraints

Hard Constraints:

- Properties of the solution to be met *exactly*
- Three options to implement hard constraints:
  - Strong soft constraints (easy, but not exact)
  - Variable elimination (exact, but limited)
  - Lagrange multipliers (most complex and general method)

Hard Soft Constraints

Simplest Implementation

- Soft constraints with large weight

$E(f) = E^{(data)}(f) + \lambda E^{(regularizer)}(f)$ with very large λ (say 10^6)

Problems

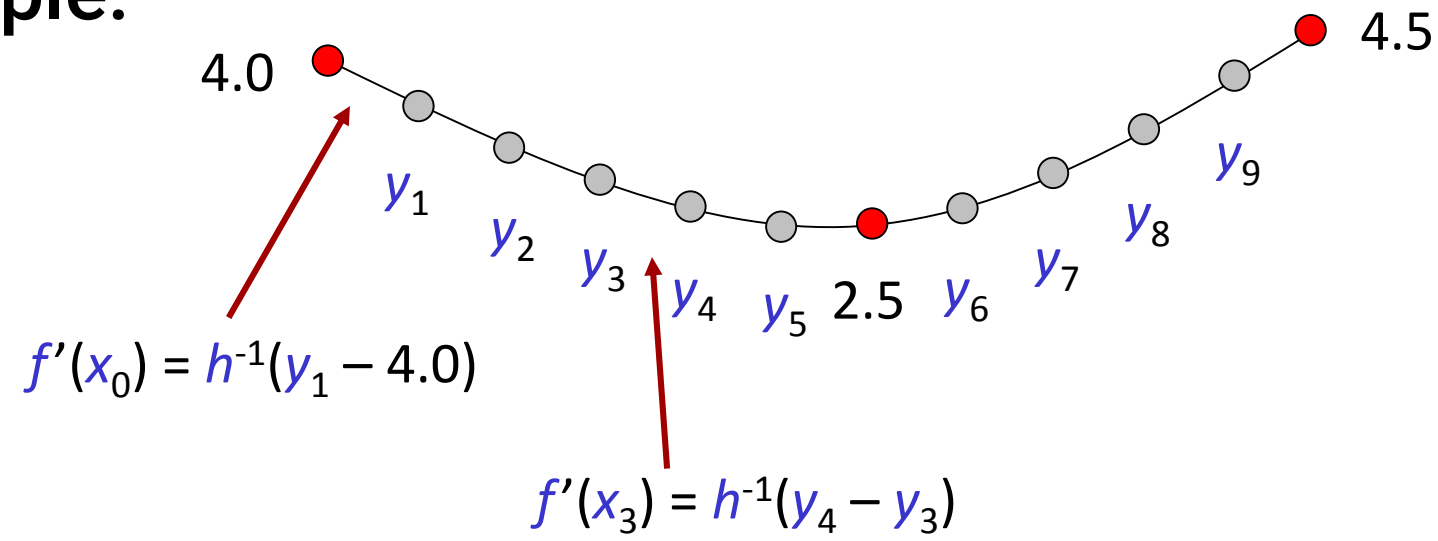
- Technique is not exact
- Stronger the constraints → larger weights
  - Condition number becomes large
  - Iterative solvers (e.g., CG) are slowed down
  - At some point, solution becomes impossible

Variable Elimination

Idea: Variable elimination

- We just replace variables by fixed numbers.
- Then solve the remaining system.

Example:



Variable Elimination

Advantages:

- Exact constraints
- Conceptually simple

Problems:

- Only works for simple constraints (variable = value)
- Need to augment system
  - Not easy to implement generically
- Does not work for FE methods (general basis functions)
  - Values are *sum* of scaled basis functions

Lagrange Multipliers

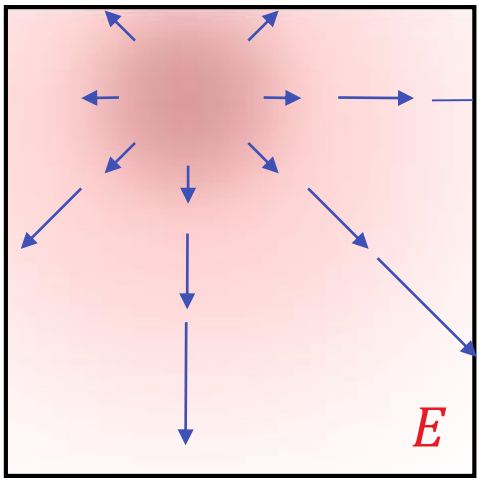
Most general technique: Lagrange multipliers

- Works for complex, composite constraints
- General basis functions
- Exact solutions (no approximation)

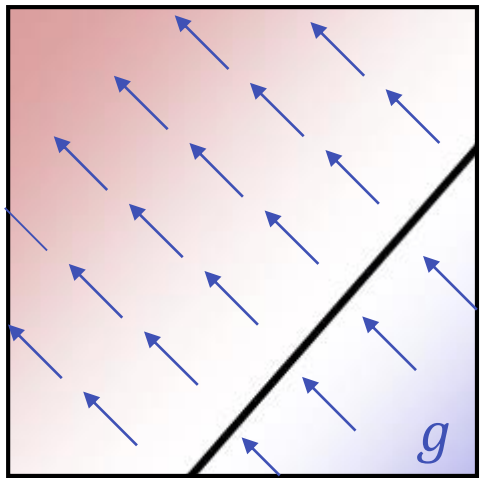
Lagrange Multipliers

Here is the idea:

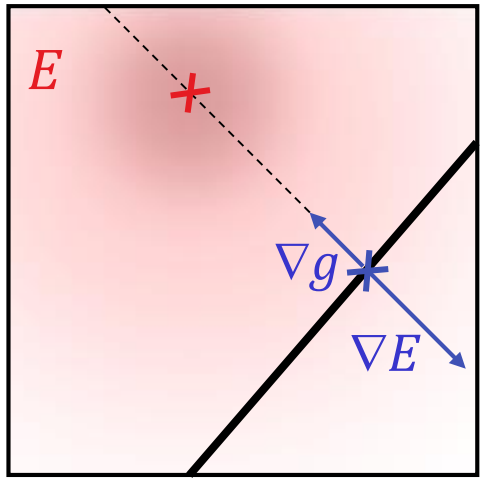
- Assume we want to optimize $E(x_1, ..., x_n)$ subject to an implicitly formulated constraint $g(x_1, ..., x_n) = 0$.
- This looks like this:



∇E



∇g



$\nabla E = \lambda \nabla g, g(\mathbf{x}) = 0$

Lagrange Multipliers

Formally:

- Optimize $E(x_1, ..., x_n)$ subject to $g(x_1, ..., x_n) = 0$.
- Formally, we want:

  $$\nabla E(\mathbf{x}) = \lambda \nabla g(\mathbf{x}) \text{ and } g(\mathbf{x}) = 0$$

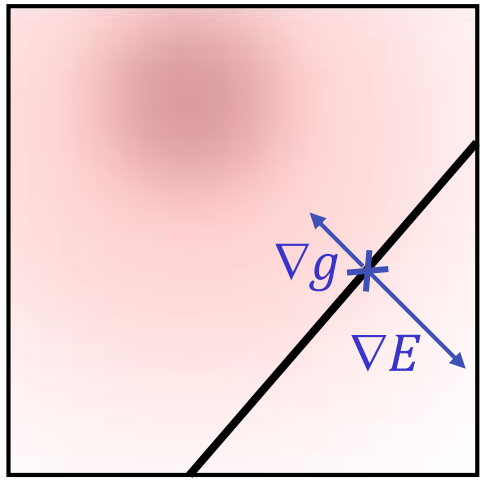
- We get a local optimum for:

  $$LG(\mathbf{x}) = E(\mathbf{x}) + \lambda g(\mathbf{x})$$

  $$\nabla_{\mathbf{x},\lambda} LG(\mathbf{x}) = 0$$

  $$\text{i.e.: } \left(\partial_{x_1}, ..., \partial_{x_n}, \partial_{\lambda}\right) LG(\mathbf{x}) = 0$$

- A critical point of this equation satisfies both $\nabla E(\mathbf{x}) = \lambda \nabla g(\mathbf{x})$ and $g(\mathbf{x}) = 0$



$\nabla E = \lambda \nabla g$

Example

Example: Optimizing a quadric function subject to a linear equality constraint

- We want to optimize: $E(\mathbf{x}) = \mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{b}\mathbf{x}$
- Subject to: $g(\mathbf{x}) = \mathbf{m}\mathbf{x} + n = 0$

We obtain: $LG(\mathbf{x}) = E(\mathbf{x}) + \lambda \mathbf{g}(x) = \mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{b}\mathbf{x} + \lambda(\mathbf{m}\mathbf{x} + n)$

- $\nabla_{\mathbf{x}}(LG(\mathbf{x})) = 2\mathbf{A}\mathbf{x} + \mathbf{b} + \lambda\mathbf{m}$

  $\nabla_{\lambda}(LG(\mathbf{x})) = \mathbf{m}\mathbf{x} + n$

- Linear system: $\begin{pmatrix} 2\mathbf{A} & \mathbf{m} \\ \mathbf{m}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} -b \\ -n \end{pmatrix}$

Multiple Constraints

Multiple Constraints:

- Similar idea
- Introduce multiple "Lagrange multipliers" λ.

$$E(x) \rightarrow \min$$

$$\text{subject to: } \forall i = 1 \ldots k : g_i(x) = 0$$

Lagrangian objective function:

$$LG(\mathbf{x}) = E(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i g_i(\mathbf{x})$$

$$\nabla_{\mathbf{x},\boldsymbol{\lambda}} LG(\mathbf{x}) = 0$$

$$\text{i.e.: } \left(\partial_{x_1}, \ldots, \partial_{x_n}, \partial_{\lambda_1}, \ldots, \partial_{\lambda_k}\right) LG(\mathbf{x}) = 0$$

Multiple Constraints

Example: Linear subspace constraints

- $E(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}\mathbf{x}$ subject to $g(\mathbf{x}) = \mathbf{M}\mathbf{x} + \mathbf{n} = \mathbf{0}$
- $LG(\mathbf{x}) = E(\mathbf{x}) + \sum_{i=1}^{n} \lambda_i \mathbf{g}_i(x) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}\mathbf{x} + \sum_{i=1}^{n} \lambda_i (\mathbf{m}_i\mathbf{x} + n_i)$
- Linear system: $\begin{pmatrix} 2\mathbf{A} & \mathbf{M}^T \\ \mathbf{M} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\mathbf{b} \\ -\mathbf{n} \end{pmatrix}$
- Remark: M must have full rank for this to work.

What can we do with this?

Multiple linear equality constraints

- Constrain
  - multiple function values
  - differential properties
  - integral values
- Area constraints:
  - Sample at each basis function of the discretization and prescribe a value
- Need to take care:
  - We need to make sure that the constraints are linearly independent at any time

What can we do with this?

Inequality constraints

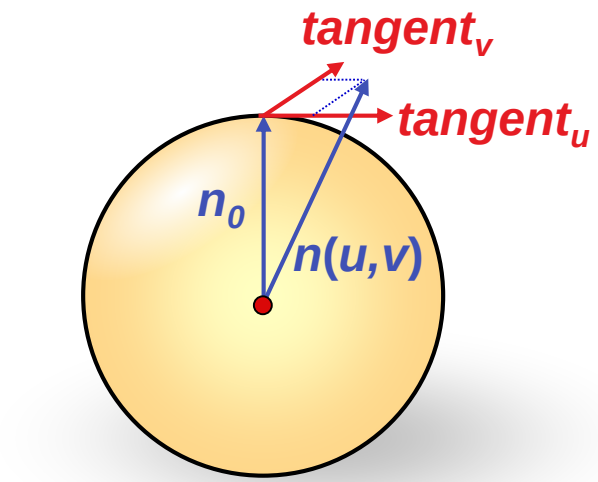
- There are efficient quadratic programming algorithms.
- Idea: turn on and off the constraints on demand
- Methods: Simplex, interior point methods

Manifold Constraints

Optimization on Unit Sphere

Solution: Local Parameterization

- Current normal estimate
- Tangent parameterization
- New variables *u*, *v*
- Renormalize
- Non-linear optimization
- No degeneracies



$$n(u,v) = n_0 + u \cdot tangent_u + v \cdot tangent_v$$

[Hoffer et al. 04]

Optimization on SO(3)

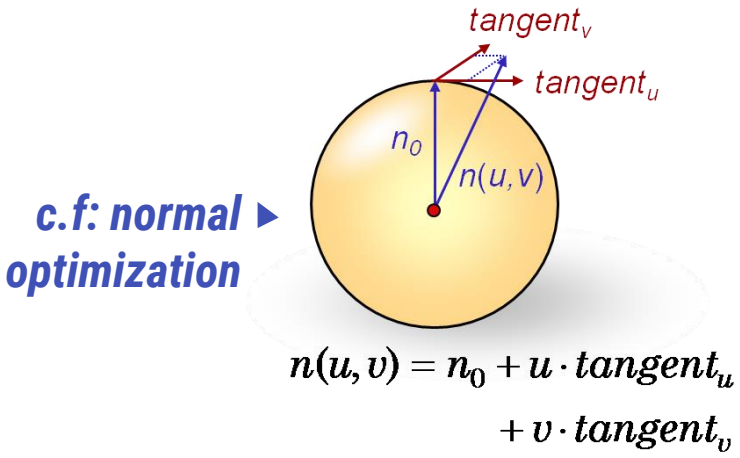
Orthonormal matrices

- Local, 1st order, non-degenerate parametrization:

$$\mathbf{C}_{\times i}^{(t)} = \begin{pmatrix} 0 & \alpha & \beta \\ -\alpha & 0 & \gamma \\ -\beta & -\gamma & 0 \end{pmatrix}$$

$$\mathbf{A}_i = \mathbf{A}_0 \exp(\mathbf{C}_{\times i}) \doteq \mathbf{A}_0 (I + \mathbf{C}_{\times i}^{(t)})$$

- Optimize parameters α, β, γ, then recompute A_0



The Euler Lagrange Equation

(connection to PDEs)

The Euler-Lagrange Equation

Conversion

- Integral energy minimization *to* differential equation
- Specific form

$$f : [a,b] \to \mathbb{R}$$

$$E(f) = \int_a^b F(x, f(x), f'(x))dx$$

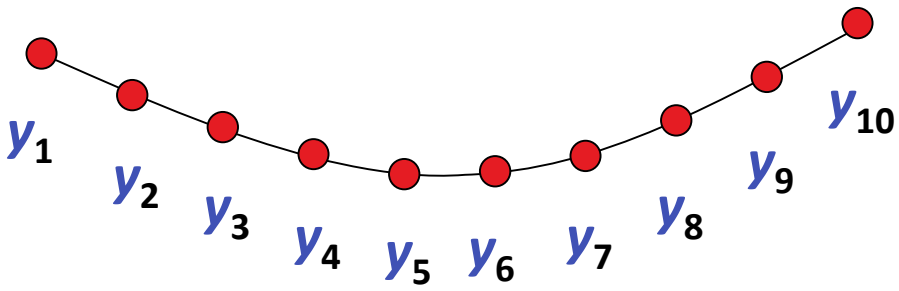
  - f: unknown function
  - F: energy "density" at each point x
  - F may depends on
    - position x
    - function value $f(x)$
    - first derivative $f'(x)$.

The Euler-Lagrange Equation

Now we look for a minimum:

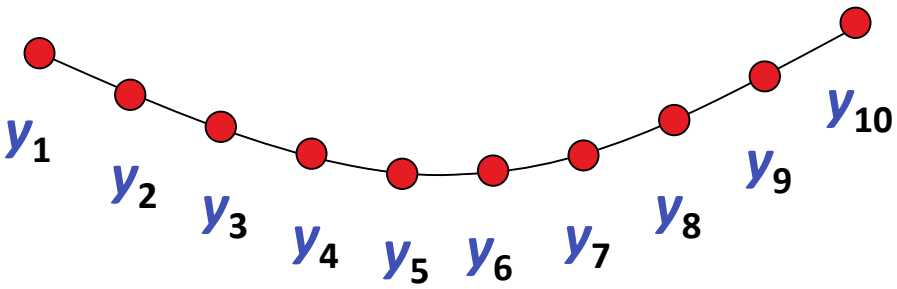
- Necessary condition:
- $\frac{d}{df} E(f) = 0$ (critical point)
- In order to compute this:
  - Approximate f by a polygon (finite difference approximation)
  - $\hat{f} = ((x_1, y_1), \ldots, (x_n, y_n))$
  - Equally spaced: $x_i - x_{i-1} = h$

(Formalized more precisely using *functional derivatives*)



The Euler-Lagrange Equation

Minimum condition:



$$E(f) = \int_a^b F(x, f(x), f'(x))dx$$

$$E(f) \approx \tilde{E}(\mathbf{y}) = \sum_{i=2}^{n} F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h} \right)$$

$$\nabla_{\mathbf{y}} \tilde{E} = \left(\partial_{y_1}, \ldots, \partial_{y_n} \right) \tilde{E}$$

$$= \sum_{i=2}^{n} \nabla_{\mathbf{y}} F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h} \right)$$

$$= \sum_{i=2}^{n} \left[\partial_2 F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h} \right) \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \partial_3 \frac{1}{h} F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h} \right) \begin{pmatrix} 0 \\ \vdots \\ -1 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \right]$$

The Euler-Lagrange Equation

Minimum condition:

$$\nabla_{\mathbf{y}} \widetilde{E} = \sum_{i=2}^{n} \left[\partial_2 F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h} \right) \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \partial_3 \frac{1}{h} F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h} \right) \begin{pmatrix} 0 \\ \vdots \\ -1 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \right]$$

ith entry:

$$\partial_{y_i} \widetilde{E} = \partial_2 F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h} \right) - \frac{1}{h}\left(\partial_3 F\left(x_i, y_i, \frac{y_{i+1} - y_i}{h} \right) - \partial_3 F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h} \right) \right)$$

Let $h \to 0$: continuous Euler-Lagrange equation

$$\partial_2 F(x, f(x), f'(x)) - \frac{d}{dx} \partial_3 F(x, f(x), f'(x)) = 0$$

Example: Harmonic Energy

$$E(f) = \int_a^b \left(\frac{d}{dx} f(x) \right)^2 dx$$

$$F(x, f(x), f'(x)) = f'(x)^2$$

$$\partial_2 F(x, f(x), f'(x)) - \frac{d}{dx} \partial_3 F(x, f(x), f'(x)) = 0$$

$$\Leftrightarrow 0 - \frac{d}{dx} \partial_{f'(x)} (f'(x))^2 = 0$$

$$\Leftrightarrow 0 - \frac{d}{dx} 2 \frac{d}{dx} f(x) = 0$$

$$\Leftrightarrow \frac{d^2}{dx^2} f(x) = 0$$

Generalizations

Multi-dimensional version:

$$f : \mathbb{R}^d \supseteq \Omega \to \mathbb{R}$$

$$E(f) = \int_{\Omega} F\left(x_1, \ldots, x_d, f(x), \partial_{x_1} f(\mathbf{x}), \ldots, \partial_{x_d} f(\mathbf{x})\right) dx_1 \ldots dx_d$$

Necessary condition for extremum:

$$\frac{\partial E}{\partial f(\mathbf{x})} - \sum_{i=1}^{d} \frac{d}{dx_i} \frac{\partial E}{\partial f_{x_i}} = 0$$

$$f_{x_i} := \frac{\partial}{\partial x_i} f(\mathbf{x})$$

This is a *partial differential equation (PDE)*.

Example

Example: General Harmonic energy

$$E^{(harmonic)}(f) = \int_{\Omega} \left(\nabla \mathbf{f}(\mathbf{x})\right)^2 d\mathbf{x}$$

Euler Lagrange equation:

$$\Delta \mathbf{f}(\mathbf{x}) = \left(\frac{\partial^2}{\partial x_1^{\,2}} f(\mathbf{x}) + \ldots + \frac{\partial^2}{\partial x_d^{\,2}} f(\mathbf{x}) \right) = 0$$

Summary

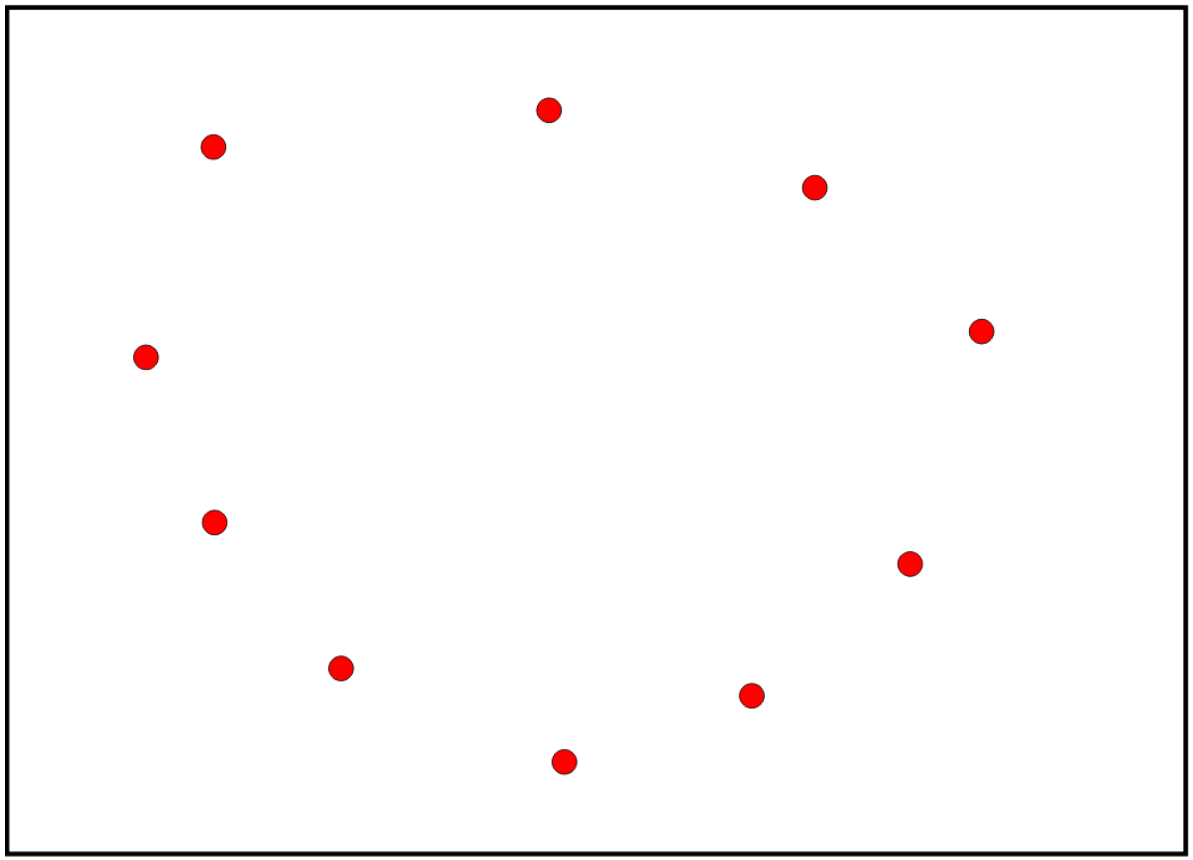
Euler Lagrange Equation:

- Converts integral minimization to ODE/PDE
  - Critical point in function space
  - Necessary, but not always sufficient condition for extremum
- Applications
  - No big change from numerical point of view
    - We could directly optimize the integral expression
    - Similarly complex to compute (boundary value problem for a PDE vs. variational problem).
  - Analytical tool
    - Helps understanding the minimizer functions.

Example 2

Reconstruction from Point Clouds

Plane Blending Method



Initial data

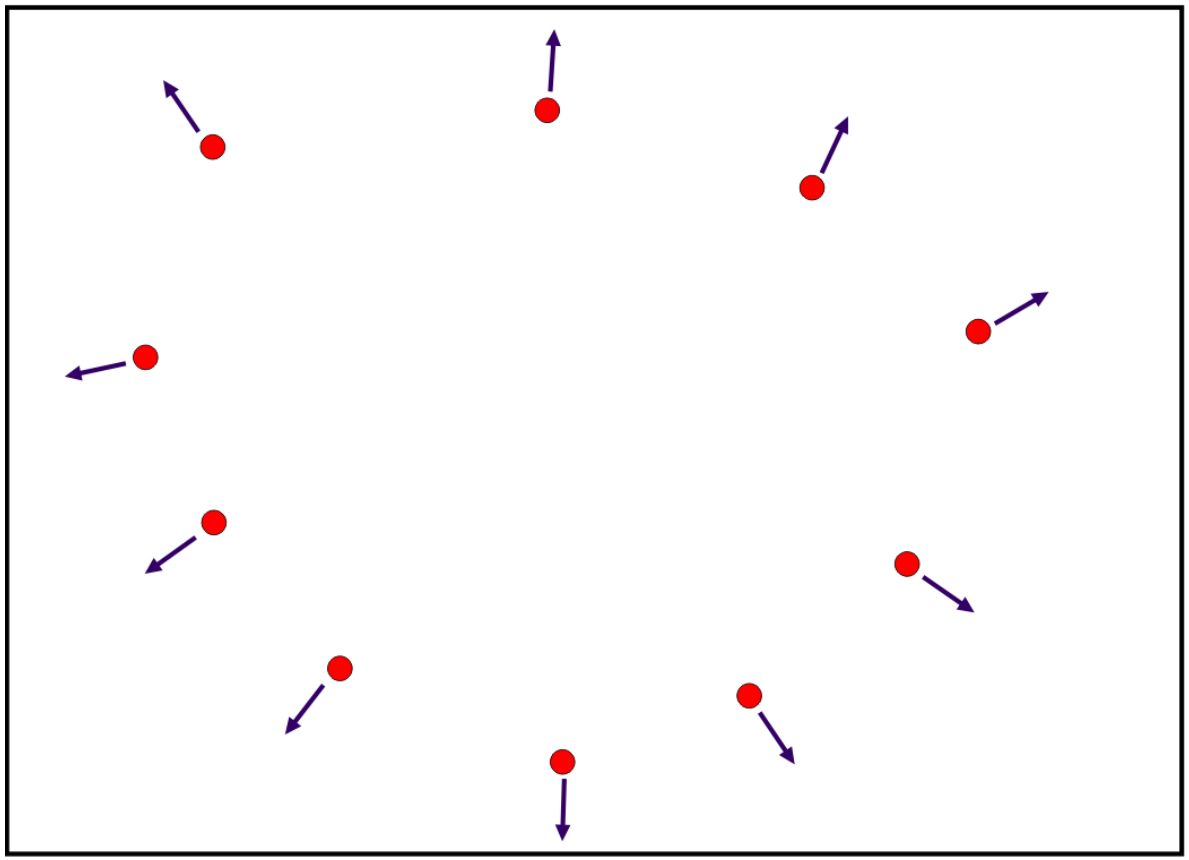
Estimate normals

Signed distance func.

Marching cubes

Final mesh

Plane Blending Method



unoriented normals
PCA among *k*-nearest neighbors

Initial data

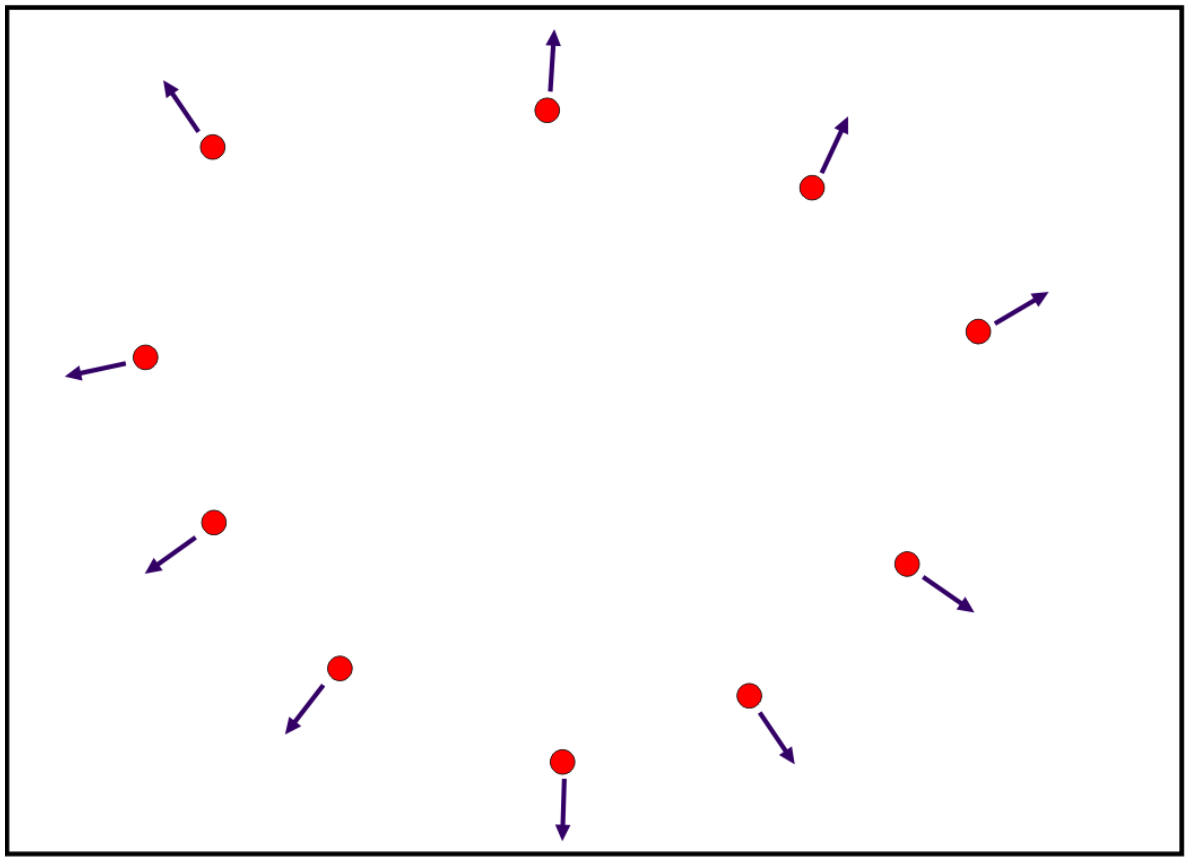
Estimate normals

Signed distance func.

Marching cubes

Final mesh

Plane Blending Method



consistent orientation
(e.g.: region growing)

Initial data

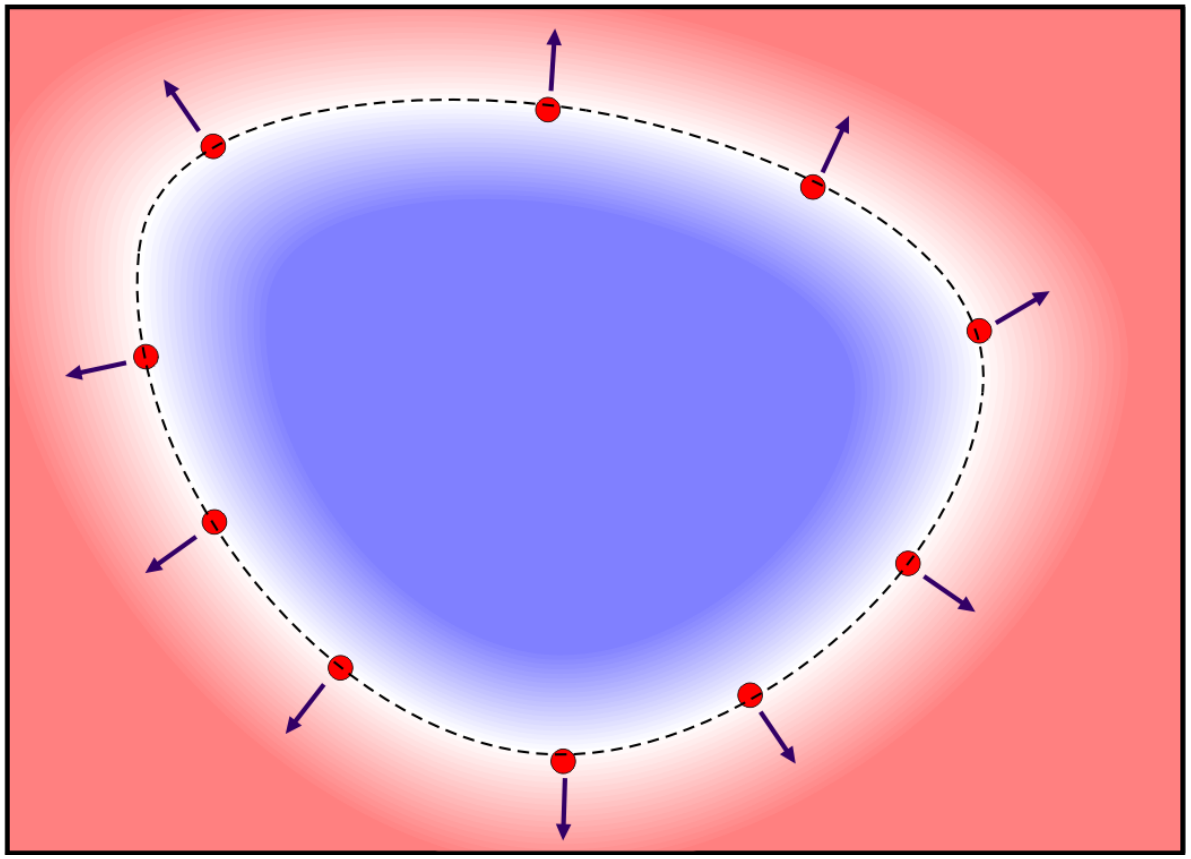
Estimate normals

Signed distance func.

Marching cubes

Final mesh

Plane Blending Method



global signed distance field
blend between signed distance functions

Initial data

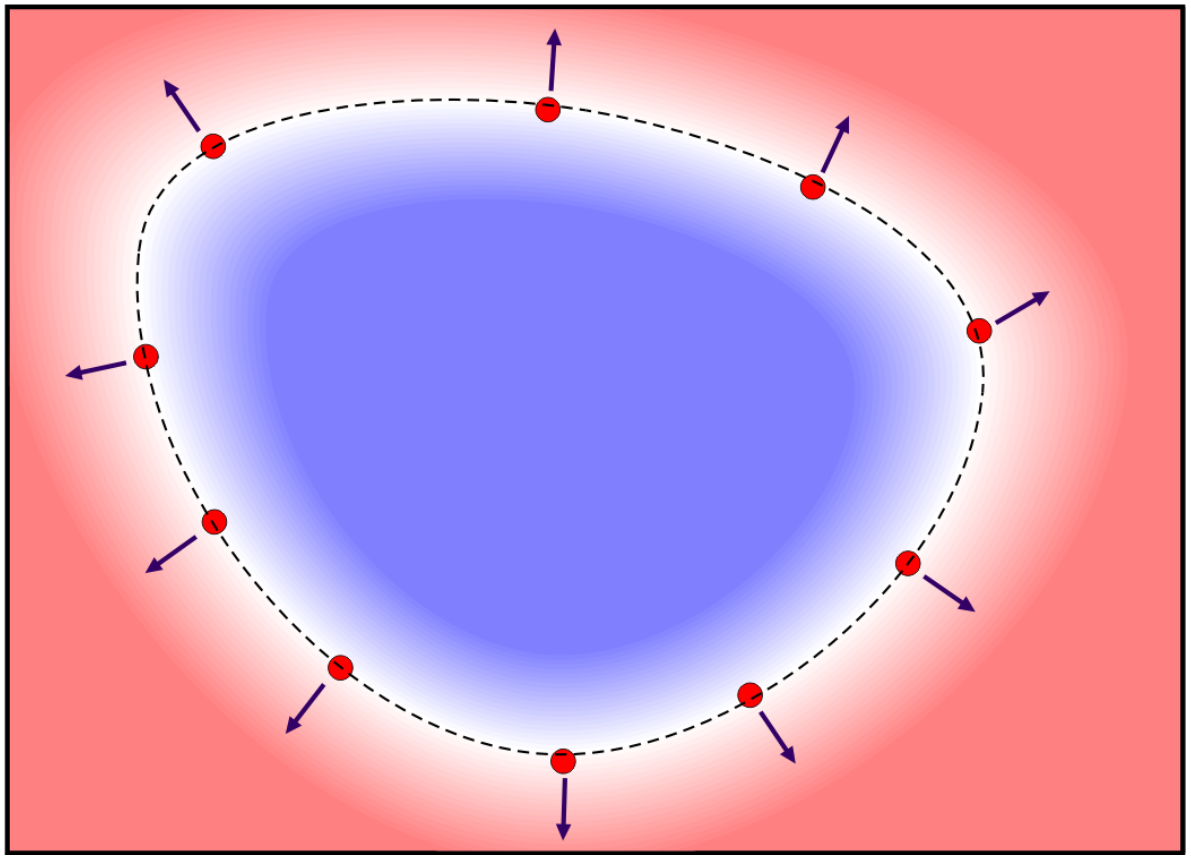
Estimate normals

Signed distance func.

Marching cubes

Final mesh

Plane Blending Method



signed distance function:
plane blending

Initial data

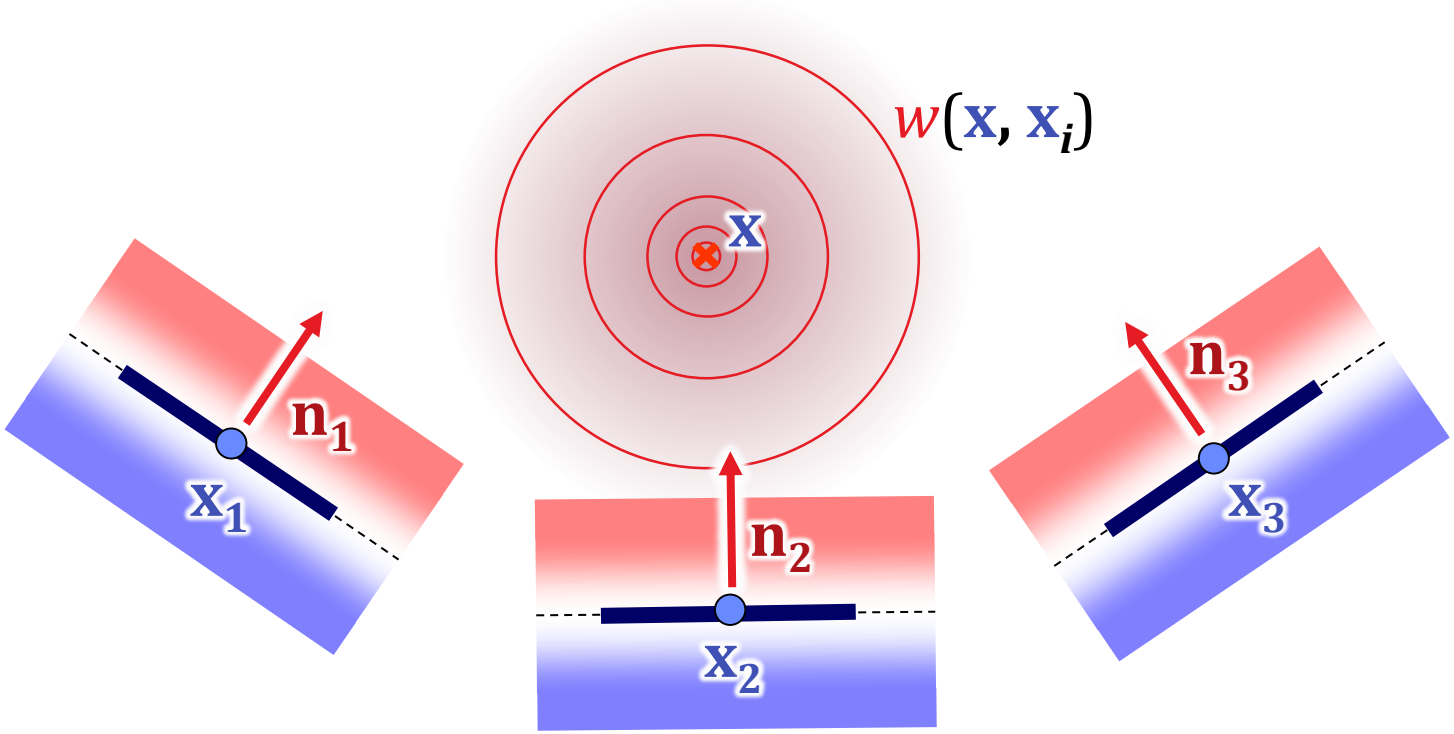
Estimate normals

Signed distance func.

Marching cubes

Final mesh

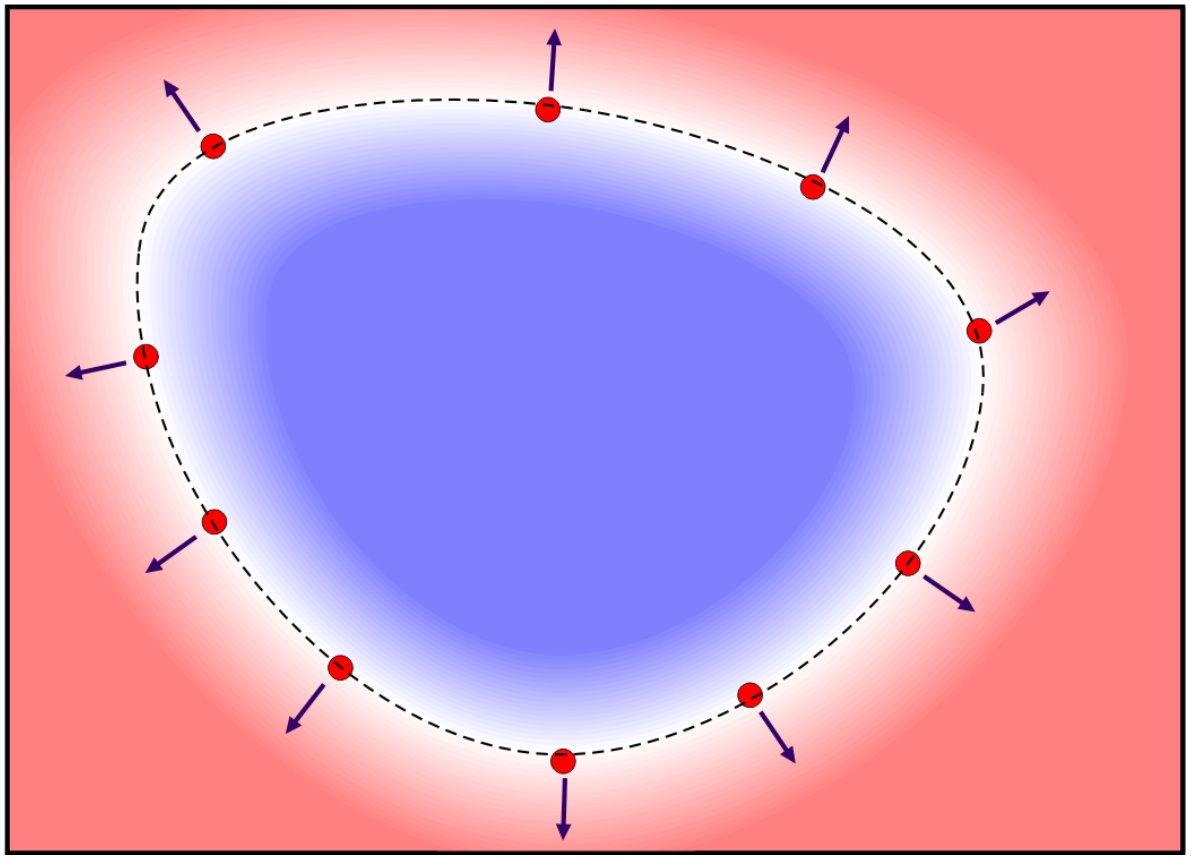
For Example: IMLS



$$f(\mathbf{x}) = \frac{\sum_{i=1}^{n} \langle \mathbf{n}_i, \mathbf{x} - \mathbf{x}_i \rangle \cdot w(\|\mathbf{x} - \mathbf{x}_i\|)}{\sum_{i=1}^{n} w(\|\mathbf{x} - \mathbf{x}_i\|)}$$

(partition of unity weights)

Plane Blending Method



Initial data

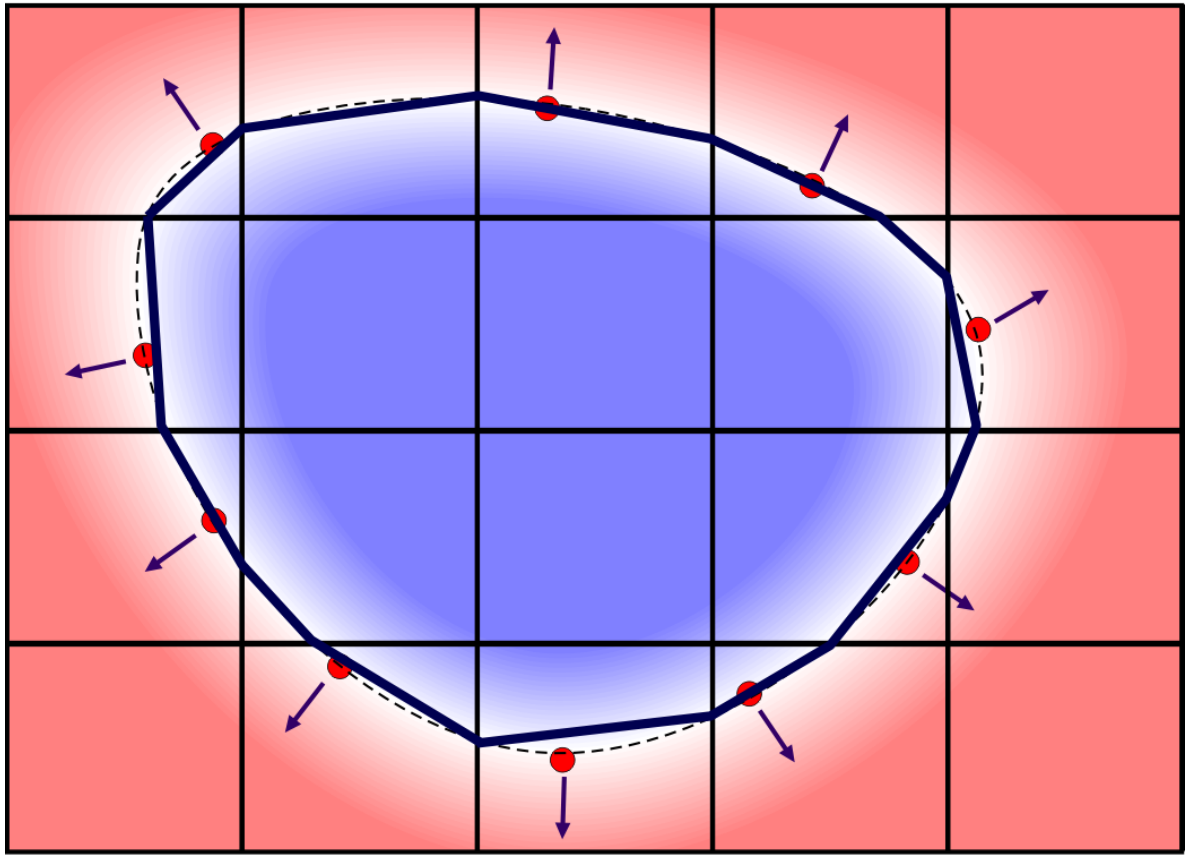
Estimate normals

Signed distance func.

Marching cubes

Final mesh

Plane Blending Method



Initial data

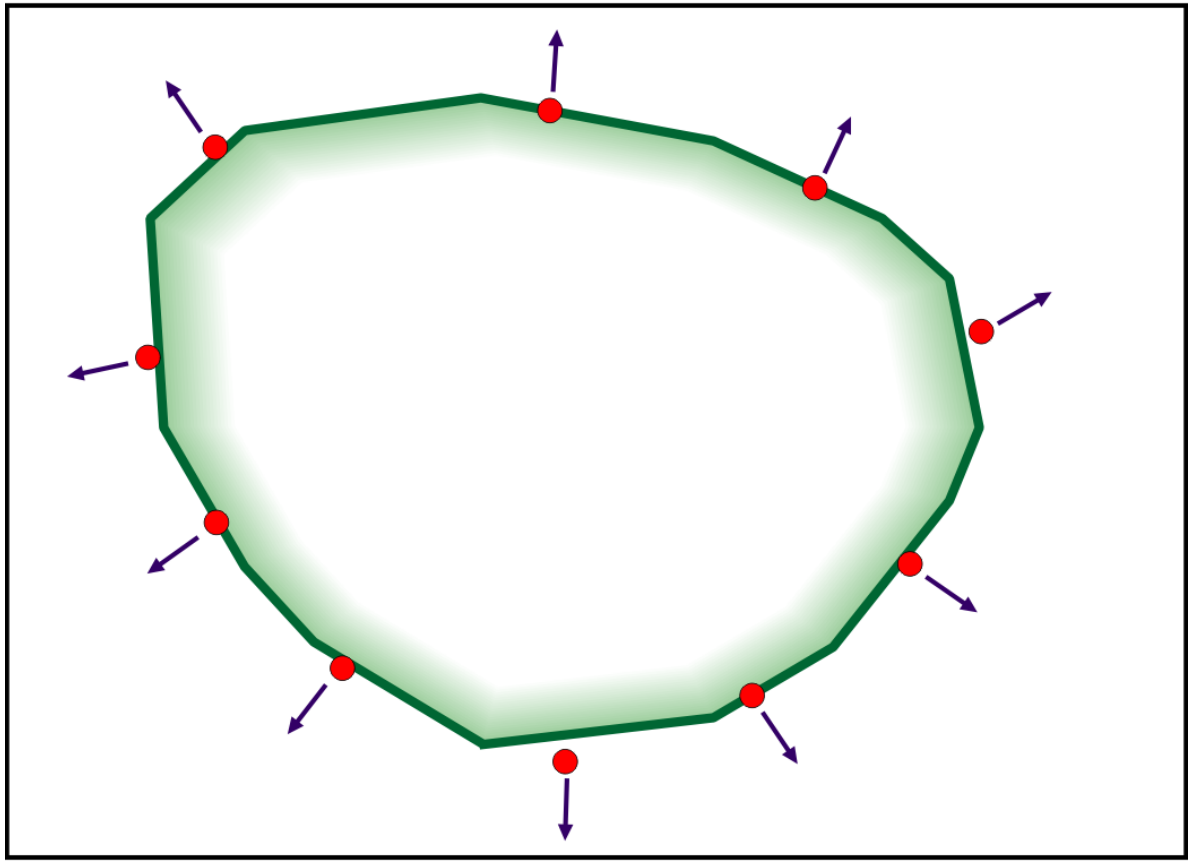
Estimate normals

Signed distance func.

Marching cubes

Final mesh

Plane Blending Method



Initial data

Estimate normals

Signed distance func.

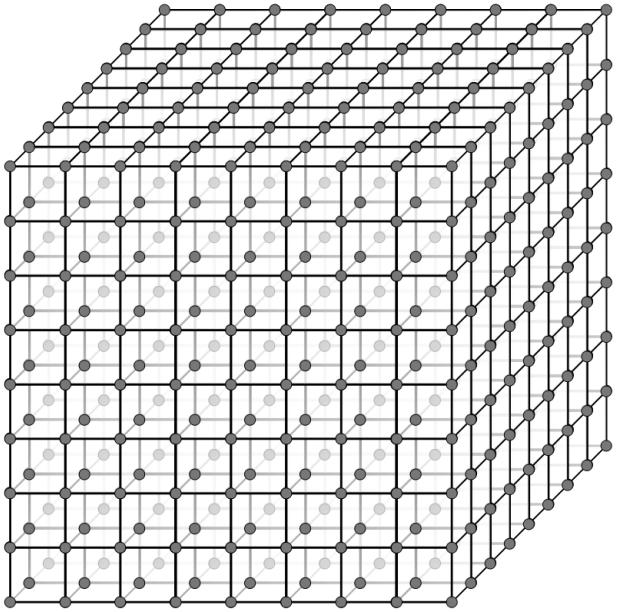
Marching cubes

Final mesh

Outer Loop

Outer Loop

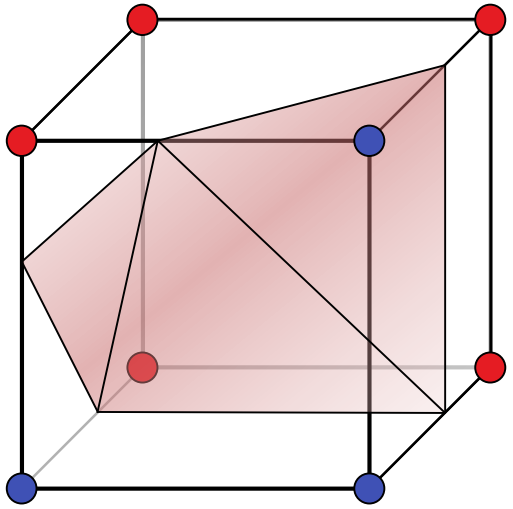
- Bounding box
- Divide into cubes
  - Regular grid
- Tessellate each sub cube



Marching Cubes

Marching Cubes:

- Local problem:
  - Cube with 8 vertices
  - Each vertex is either inside or outside the volume (i.e. $f(\mathbf{x}) < 0$ or $f(\mathbf{x}) \geq 0$)
  - How to triangulate this cube?
  - How to place the vertices?



Problems

Plane-Equation-Blending Method (“ILMS”)

- “Implicit-Moving-Least-Squares”
- Consistency order 0
- “partition of unity” (normalize weights)

Shortcomings

- Only works near points
  - Division by zero otherwise
- Thus cannot fill larger holes
  - Local reconstructions only
  - Cannot (easily) handle varying sampling density

Variational Model

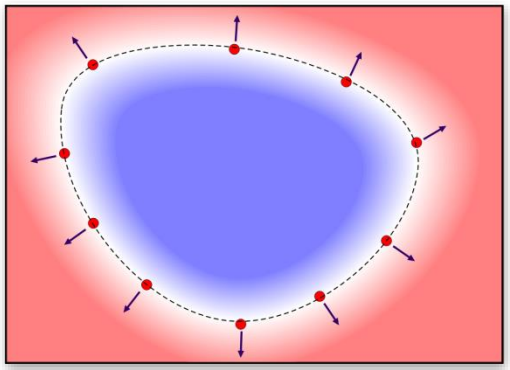
Implicit Function

Implicit surfaces

- Smooth implicit function

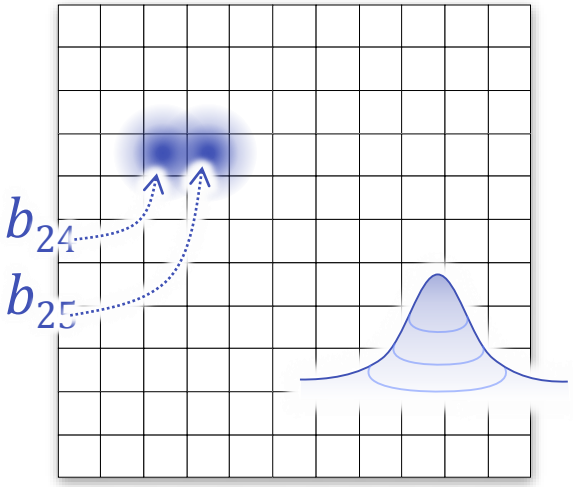
$$f: \mathbb{R}^3 \rightarrow \mathbb{R}$$

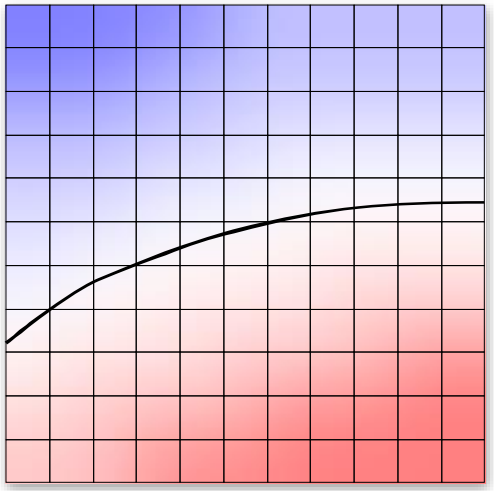
$$M = \{\mathbf{x} | f(\mathbf{x}) = 0, \mathbf{x} \in \mathbb{R}^3\}$$



- Flexible topology

$$f(\mathbf{x}) = \sum_{i=1}^{d} \lambda_i b_i(\mathbf{x})$$

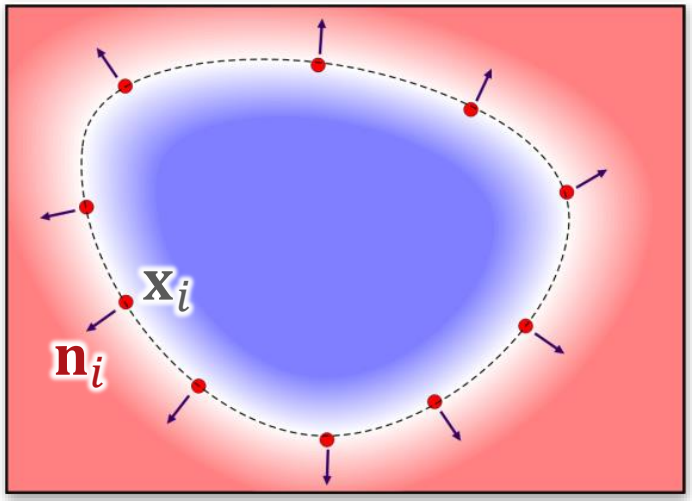




Data Term

Neg-log. Baysian rule:

$$E(M|D) \sim E(D|M) + E(M)$$



Data Term

$$E(D|M) = \sum_{i=1}^{n} \left(f(\mathbf{x}_i)^2 + \left(\mathbf{n}_i - \nabla f(\mathbf{x}_i) \right)^2 \right)$$

Regularizer (Prior)

$$E(M) = \int_{\Omega} \left\| H_f(\mathbf{x}) \right\|_F^2 d\mathbf{x}$$

Solving

Objective Function

$$\underbrace{\sum_{i=1}^{n}\left(f(\mathbf{x}_i)^2+\left(\mathbf{n}_i-\nabla f(\mathbf{x}_i)\right)^2\right)}_{\text{data term}}+\underbrace{\int_{\Omega}\left\|H_f(\mathbf{x})\right\|_F^2 d\mathbf{x}}_{\text{regularizer}}$$

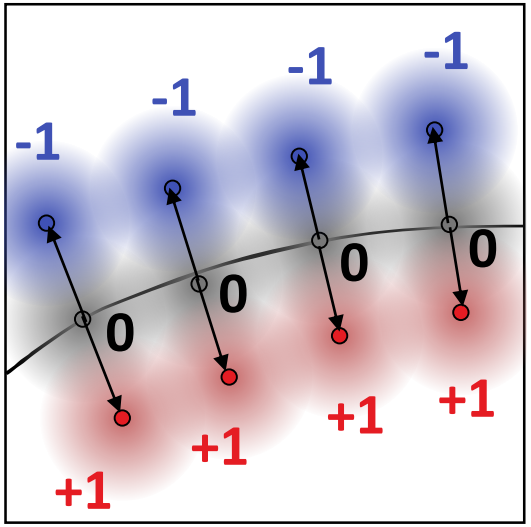
Discretization (Example)

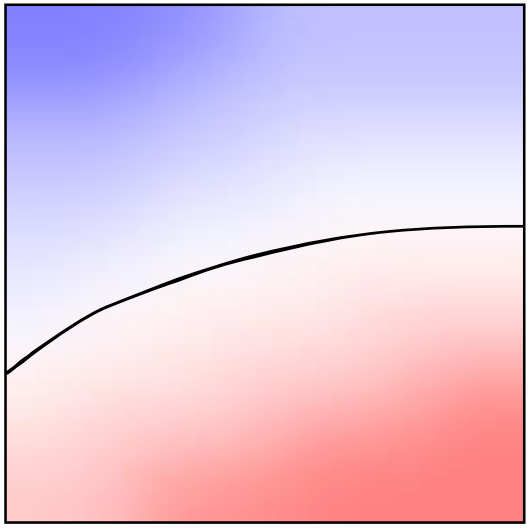
- Regular grid
- Finite differences for H_f (using coefficients on grid)
- Interpolation with basis functions for data term (sub-grid precision)
- Solve linear system with conjugate gradients

Direct Solution

Simpler [Carr et al. 2001]

- Place two more in normal and negative normal direction
- Prescribe values +1,0,-1





Radial Basis Functions

Radial basis functions [Carr et al. 2001]

- Globally supported functions
  - Thin-plate spline basis functions:

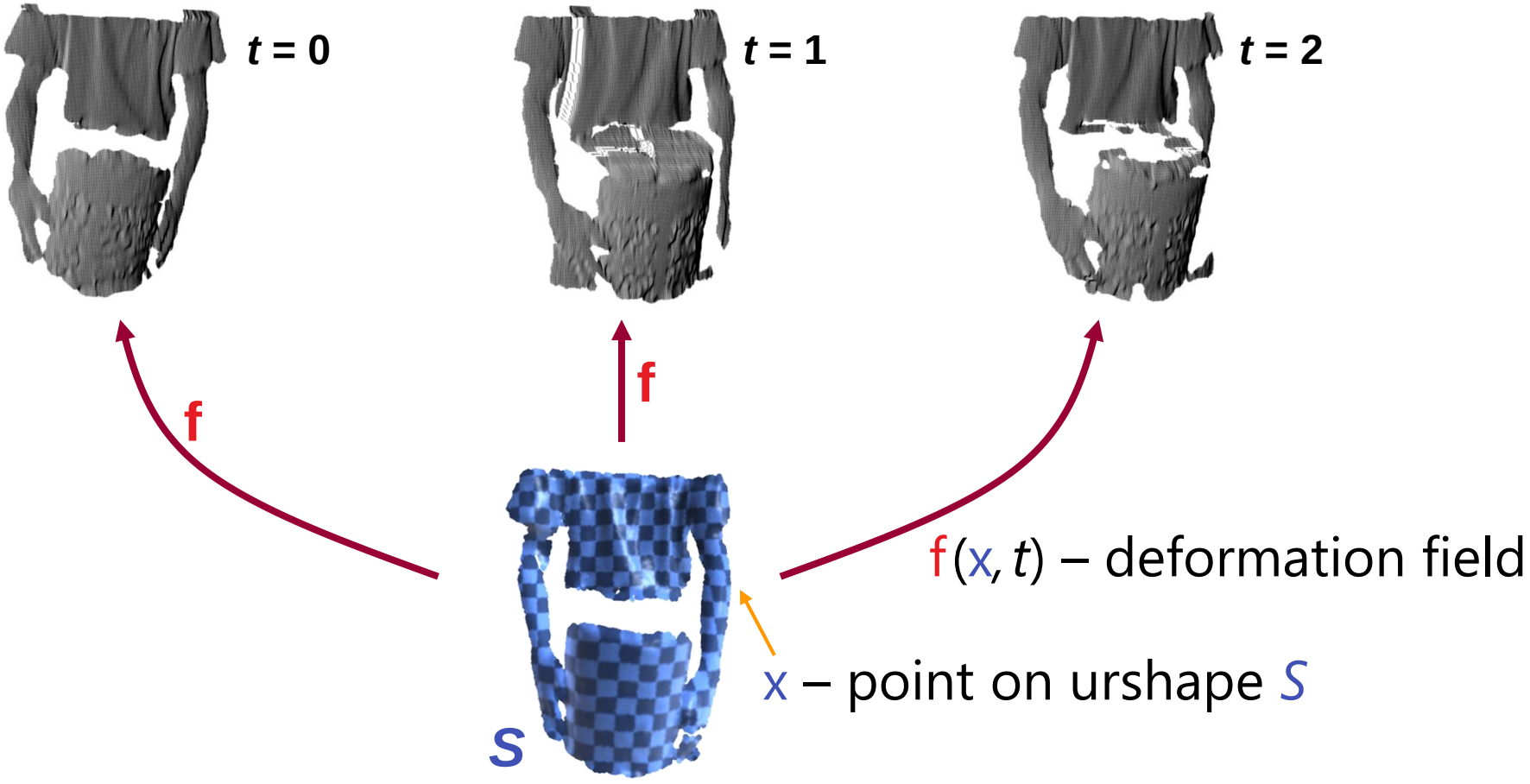
    $\|x - x_0\|^2 \ln\|x - x_0\|$ **(2D)** $\quad$ $\|x - x_0\|^3$ **(3D)**

  - Guarantee minimal integral second derivatives.
- Problem: complexity
  - Every basis function interacts with each other one
  - Dense $n \times n$ linear system
- Fast multi pole method
  - Clusters far away nodes in bigger octree boxes
  - O(log n) interactions per function, overall O(n log n)

More Complex Application Examples

Animation Reconstruction

Variational Animation Modeling



Variational Framework

$$E(\mathbf{f}) = \underbrace{E_{match}(\mathbf{f})}_{\text{constraints}} + \underbrace{(E_{rigid} + E_{volume} + E_{accel} + E_{velocity})(\mathbf{f})}_{\text{deformation}}$$

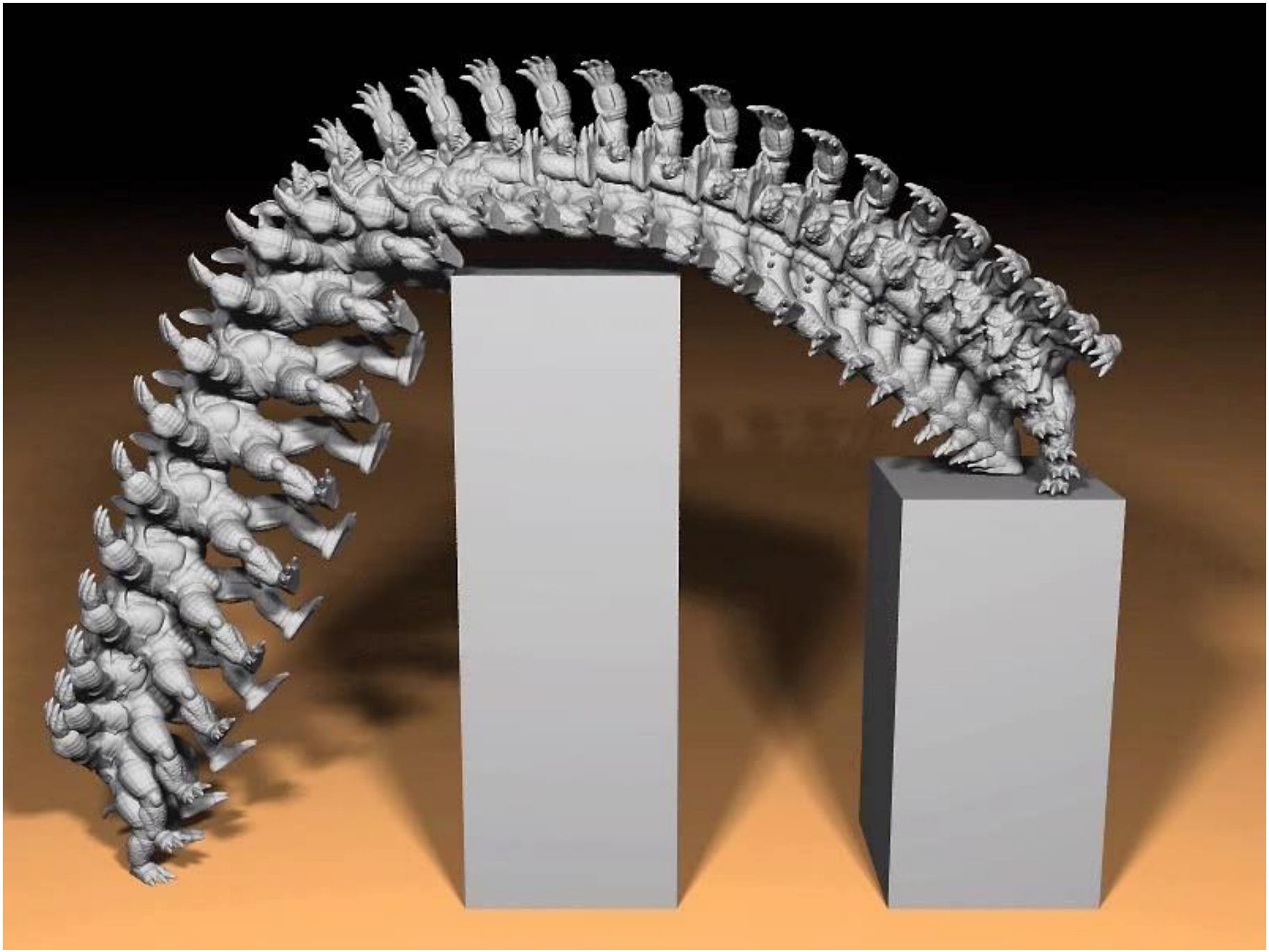
$$E_{match}(\mathbf{f}) = \sum_{t=1}^{T} \sum_{i=1}^{n_t} dist(d_i, f(S))^2$$

$$E_{rigid}(\mathbf{f}) = \int_{V(S)} \omega_{rigid}(x) \left\| \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x},t)^T \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x},t) - \mathbf{I} \right\|_F^2 dx$$

$$E_{volume}(\mathbf{f}) = \int_{V(S)} \omega_{vol}(x) \left(\left| \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x},t) \right| - 1 \right)^2 dx$$

$$E_{accel}(\mathbf{f}) = \int_{S} \omega_{acc}(x) \left(\frac{\partial^2}{\partial t^2} \mathbf{f}(\mathbf{x},t) \right)^2 dx$$

$$E_{velocity}(\mathbf{f}) = \int_{S} \omega_{velocity}(x) \left(\frac{\partial}{\partial t} \mathbf{f}(\mathbf{x},t) \right)^2 dx$$



[B. Adams, M. Ovsjanikov, M. Wand, L. Guibas, H.-P. Seidel, SCA 2008]

Meshless Modeling of Deformable Shapes and their Motion

Bart Adams[1,2] Maks Ovsjanikov[1] Michael Wand[3]
Hans-Peter Seidel[4] Leonidas J. Guibas[1]

[1]Stanford University
[2]Katholieke Universiteit Leuven
[3]Max Planck Center for Visual Computing and Communication
[4]Max Planck Institut Informatik

[B. Adams, M. Ovsjanikov, M. Wand, L. Guibas, H.-P. Seidel, SCA 2008]

Data Set:

"Popcorn Tin"

94 frames
data: 53K points/frame
rec: 25K points/frame

[M. Wand, B. Adams, M. Ovsjanikov, M. Bokeloh, A. Berner, P. Jenke, L. Guibas, H.-P. Seidel, A. Schilling, 2008] (data set courtesy of P. Phong, Stanford. U.)