Geometric Modeling

Summer Semester 2012

Variational Modeling

Basic Techniques · Surface Modeling · Other Applications

UNIVERSITÄT DES SAARLANDES

M2CI CLUSTER OF EXCELLENCE

max planck institut informatik

Overview…

Topics:

- Triangle Meshes & Multi-Resolution Representations
- Implicit Functions
- Subdivision Surfaces
- Variational Modeling
  - Introduction
  - Variational Framework
  - Variational Function Fitting Toolkit
  - Euler & Lagrange – Some More Mathematical Background
  - Surface Modeling
  - Other Applications

Variational Modeling

Introduction

Motivation

Surface modeling techniques we have seen so far:

- Bivariate polynomial spline patches
  - Quad (tensor product) patches
  - Triangular patches
- Subdivision surfaces
- Implicit functions

Motivation

Problems:

- Bivariate polynomial spline patches
  - General topologies are hard to handle
  - Need to adapt base mesh to user constraints
    - control points, boundaries, etc.
- Subdivision surfaces
  - More flexible than spline patches
  - Problems:
    - Continuity at extraordinary vertices
    - Still need to build a base mesh
- Implicit functions
  - Nice tool – but how do we construct actual surfaces?

Variational Modeling

Variational Modeling:

- Different approach:
  - Formulate smoothness in terms of a penalty function
  - Set additional constraints (handle points, normals, etc)
  - Then solve for the “optimal function”
- No direct manipulation of control points...
  - No direct user interaction
    - Use e.g. B-Splines or implicit functions as numerical representation
    - Control points moved “automatically”
    - “Meta tool”: compute control points automatically
  - Instead: Sparse control points/handles with more semantics

Two Views:

In this lecture:

- Narrow view:
  - Use variational techniques for modeling shapes
- General view:
  - Short introduction / overview to variational calculus and practical techniques.
  - Application examples in geometry processing.

Applications beyond geometric modeling:

- Variational approaches ubiquitous
  - in computer graphics
  - in computer vision (in particular)

Variational Modeling

Basic Techniques

Calculus of Variation

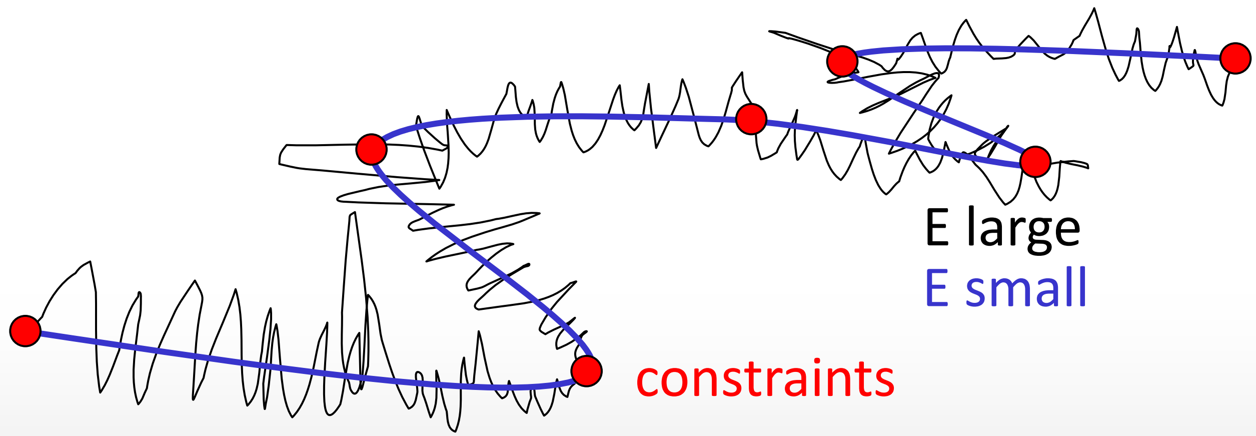
Basic Idea:

- We look at a set of functions $f: S \to D$
- Define "*energy functional*" $E: (S \to D) \to \mathbb{R}$
  - *Functional*: assigns real numbers to functions
  - Each function gets a "score"
  - "Energy" means: the smaller the better
- Add additional requirements ("constraints") on f.
  - *Soft constraints* $\to$ violation increases energy.
  - *Hard constraints* $\to$ violation not allowed.
- Compute function(s) f that minimize E.

Calculus of Variation

Very general framework:

- Many problems directly formulated this way
- Example 1:
  - Looking for a curve.
  - As smooth as possible (energy = non-smoothness).
  - It should go through a number of points (hard constraints).



Calculus of Variation

Another example:

- **Problem:** We want to go to the moon.
- **Given:**
  - Orbits of moons, planets and star(s).
  - Flight conditions (athmosphere, gravitation of stellar bodies)
- **Unknowns:**
  - Throttle (magnitude, direction) from rocket motors (vector function)
- **Energy function:**
  - Usage of rocket fuel (the fewer the better)
  - Perhaps: Overall travel time (maybe not longer than a week)

Calculus of Variation

To the moon:

- **Constraints:**
  - We want to start in Cape Canaveral (upright trajectory) and end up on the moon.
  - We do not want to hit moons or planets on our way.
  - We want to approach the moon at no more than 20 km/h relative speed upon touchdown.
  - The rocket motor has a limited range of forces it can create (not more than a certain thrust, no backward thrust)

So flying to the moon is just minimizing a functional.

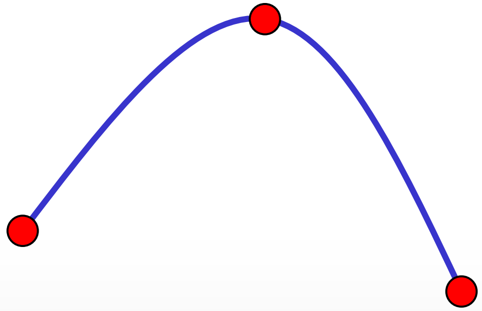
(ok, this is slightly simplified)

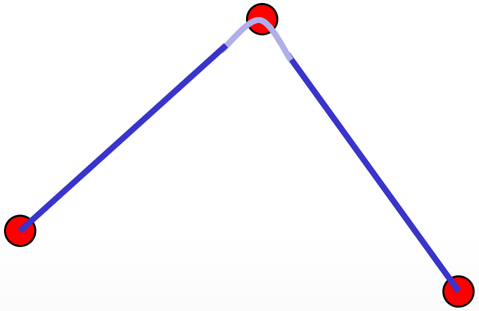
A Simple Example

Simple example: variational splines

- Energy:
  - We want smooth curves
  - Smooth translates to minimum curvature
  - Quadratic penalty:

$$E(f) = \int_{\text{curve}} |curvature_f(t)|^2 \, dt$$





A Simple Example

Simple example: variational splines

- Energy:
  - Problem: curvature is non-linear
  - Easier to minimize: second derivatives
  - Equivalent in case of a unit-speed parametrization (which is tricky to enforce)

$$E(f) = \int_{\text{curve}} \left[\frac{d^2}{dt^2} \mathbf{f}(t) \right]^2 dt$$

A Simple Example

Simple example: variational splines

- Constraints:
  - Hard constraints: we are given parameter values t_1, ..., t_n at which we should meet control points $\mathbf{p}_1$, ..., $\mathbf{p}_n$.

$$E(f) = \int_{t=t_1}^{t_n} \left[\frac{d^2}{dt^2} \mathbf{f}(t) \right]^2 dt$$

  - We already know the solution to this problem: Piecewise cubic interpolating spline.

A Simple Example

Simple example: variational splines

- More interesting: soft constraints
  - We are given parameter values $t_1,\ldots,t_n$ at which we should *approximately* meet control points $\mathbf{p}_1, \ldots, \mathbf{p}_n$.

$$E(f) = \int_{t=t_1}^{t_n} \left[\frac{d^2}{dt^2} \mathbf{f}(t) \right]^2 dt + \lambda \sum_{i=1}^{n} \left(\mathbf{f}(t_i) - \mathbf{p}_i \right)^2$$

  - λ controls the smoothness of the result. Large values reduce smoothness to meet the control points more precisely.

A Simple Example

Simple example: variational splines

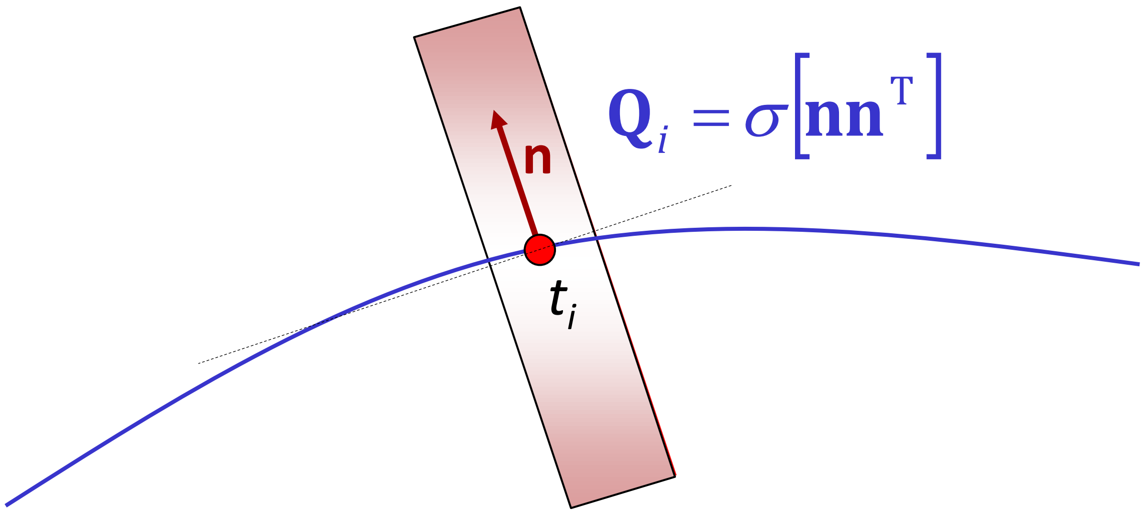
- Soft constraints
  - We are given parameter values $t_1,\ldots,t_n$ at which we should approximately meet control points $\mathbf{p}_1, \ldots, \mathbf{p}_n$, up to a specific accuracy for *each point*.
  - We can specify the accuracy by error quadrics $\mathbf{Q}_1, \ldots, \mathbf{Q}_n$.

$$E(f) = \int_{t=t_1}^{t_n} \left[\frac{d^2}{dt^2} \mathbf{f}(t) \right]^2 dt + \sum_{i=1}^{n} \left(\mathbf{f}(t_i) - \mathbf{p}_i \right)^{\mathrm{T}} \mathbf{Q}_i \left(\mathbf{f}(t_i) - \mathbf{p}_i \right)$$

Rank-Deficient Quadrics

The rank deficient error quadric trick:

- A rank-1 matrix constraints the curve in one direction only
- Useful for point-to-surface constraints (minimize normal direction deviation, tangential motion is free)



Numerical Treatment

Numerical computation:

- No closed form solution
- Instead:
  - Discretize (finite dimensional function space)
  - Solve for coefficients (coordinate vector in this function space)

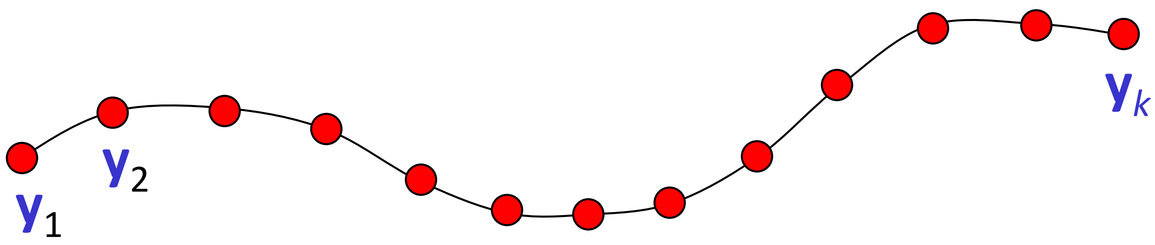
Finite Differences

FD solution:

- Represent curve as array of k values:

t	0	0.1	0.2	…	7.4	7.5
$\mathbf{y}$	$\mathbf{y}_0$	$\mathbf{y}_1$	$\mathbf{y}_2$	…	$\mathbf{Y}_{74}$	$\mathbf{y}_{75}$

- Unknowns are the curve points $\mathbf{y}_1, \ldots, \mathbf{y}_k$



Discretized Energy Function

Discretized Energy Function:

- Energy is a squared linear expression → quadratic discrete objective function
- Constraints are quadratic by construction
- Yields quadratic energy function
  - solved by a linear system

$$E(f) = \int_{t=t_1}^{t_n} \left[\frac{d^2}{dt^2} \mathbf{f}(t) \right]^2 dt + \sum_{i=1}^{n} \left(\mathbf{f}(t_i) - \mathbf{p}_i \right)^{\mathrm{T}} \mathbf{Q}_i \left(\mathbf{f}(t_i) - \mathbf{p}_i \right)$$

$$E^{(discr)}(f) = \sum_{i=1}^{k} \left[\frac{\mathbf{y}_{i-1} - 2\mathbf{y}_i + \mathbf{y}_{i+1}}{h^2} \right]^2 + \sum_{i=1}^{n} \left(\mathbf{y}_{index(t_i)} - \mathbf{p}_i \right)^{\mathrm{T}} \mathbf{Q}_i \left(\mathbf{y}_{index(t_i)} - \mathbf{p}_i \right)$$

(neglected here: handling boundary values)

Summary

Summary:

- Variational approaches look like this:

$$\text{compute } \arg\min_{f \in F} E(f),$$

$$E(f) = E^{(data)}(f) + E^{(regularizer)}(f),$$

$$f \in F = \{ f \mid f \text{ satifies hard constraints} \}$$

- Connection to statistics:
  - Bayesian maximum a posteriori estimation
  - $E^{(data)}$ is the data likelihood (log space)
  - $E^{(regularizer)}$ is a prior distribution (log space)

Variational Toolbox:

Data Fitting, Regularizer Functionals, Discretizations

Toolbox

In the following:

- We will discuss...
  - ...useful standard functionals.
  - ...how to model soft constraints.
  - ...how to model hard constraints.
  - ...how to discretize the model.
- Then snap & click your favorite custom variational modeling scheme.
- (Click & snap means: add together to a composite energy)

Functionals

Functionals

Standard Functional #1: Function norm

- Given a function **f**: $\mathbb{R}^m \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:

$$E^{(zero)}(f) = \int_{\Omega} \mathbf{f}(\mathbf{x})^2 \, d\mathbf{x}$$

- Means: the function values should not become too large
- Often useful to avoid numerical problems:
  - Assume an SPD quadratic functional
  - Add $\lambda E^{(zero)}$
    - smallest eigenvalue cannot become smaller than λ ($\rightarrow$ condition number)
    - system is always solvable

Functionals

Standard Functional #2: Harmonic energy

- Given a function **f**: $\mathbb{R}^m \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:

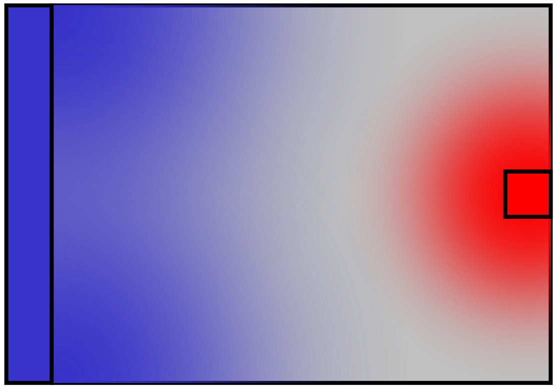
$$E^{(harmonic)}(f) = \int_{\Omega} \left(\nabla \mathbf{f}(\mathbf{x})\right)^2 d\mathbf{x}$$

- Objective: minimize differences to neighboring points
- Appears all the time in physics & engineering.
  - not really what we want for smooth curves...

Harmonic Energy

Example: Heat equation

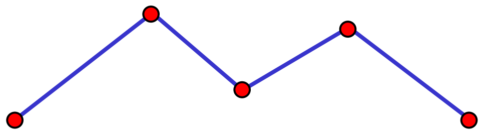
- Given a metal plate
- Hard constraints:
  - A heat source
  - A heat sink
- What is the final heat distribution?
  - Heat flow tends to equalize temperature.
    - Stronger heat flow for larger temperature gradients.
  - Gradients become as small as possible.



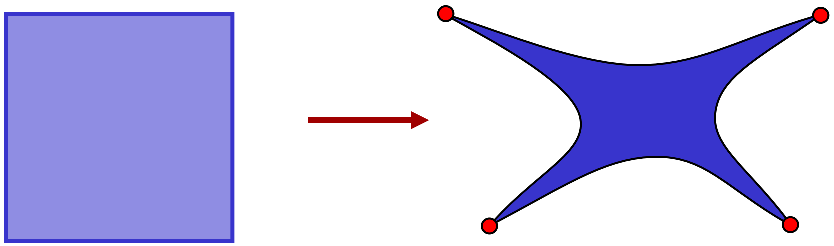
Harmonic Energy

Example: Harmonic energy

- Curves that minimize the harmonic energy:
  - Shortest path, a.k.a. polygons



- Two-dimensional parametric surface:



- Useful in parametrization (conformal mappings are harmonic)

Functionals

Standard Functional #3: Thin plate spline energy

- Given a function **f**: $\mathbb{R}^m \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:

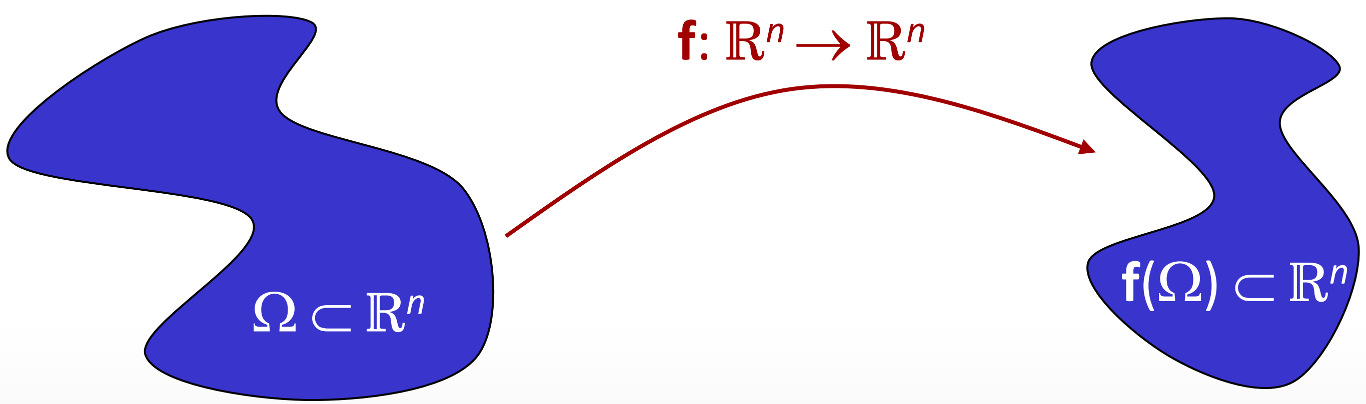
$$E^{(TSS)}(\mathbf{f}) = \int_\Omega \sum_{i=1}^{m} \sum_{j=1}^{m} \left(\left\| \frac{\partial^2}{\partial x_i \partial x_j} \mathbf{f}(\mathbf{x}) \right\|^2 \right) d\mathbf{x}$$

- Objective: minimize integral second derivatives
  - approximately: minimize curvature
- More common in geometric modeling/processing
  - yields smooth curves & surfaces
  - A true curvature based energy is rarely used (non-quadratic).

Energies for Vector Fields

Vector fields:

- The following energies are useful for mappings from $\mathbb{R}^n \rightarrow \mathbb{R}^n$ (e.g.: space deformations).
- Think of an object moving (over time).
- $\mathbf{f}(\mathbf{x})$ describes its deformation.
- $\mathbf{f}(\mathbf{x},t)$ describes its motion over time.



Functionals

Standard Functional #4: Green's deformation tensor

- Given a function **f**: $\mathbb{R}^n \supset \Omega \to \mathbb{R}^n$
- Minimize:

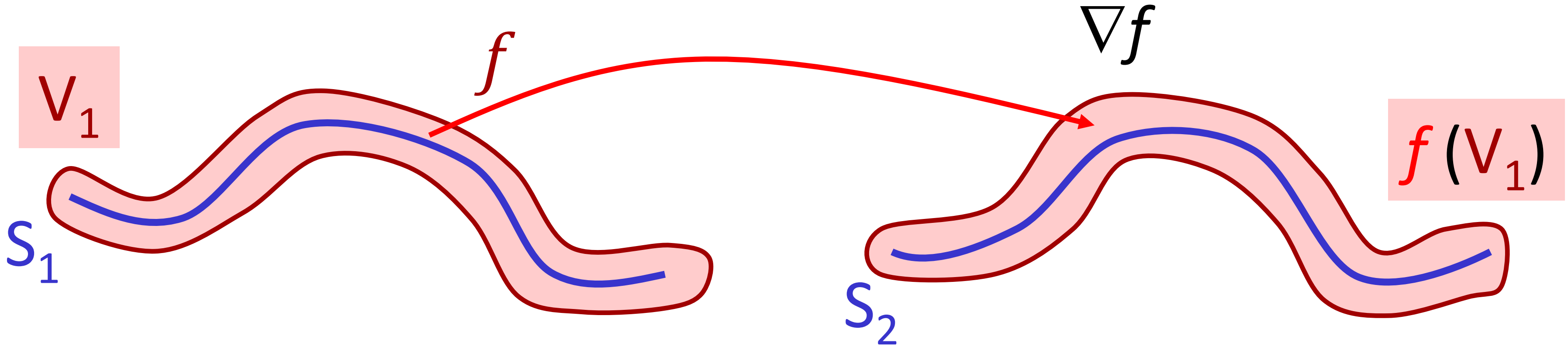
$$E^{(deform)}(\mathbf{f}) = \int_{\Omega} \left\| \mathbf{M}\left[\nabla \mathbf{f}^{\mathrm{T}} \nabla \mathbf{f} - \mathbf{I}\right] \right\|_F^2 d\mathbf{x}$$

- Objective: minimize metric distortion
  - Metric distortion $\widehat{=}$ non-identity first fundamental form
- Basis for physically-based deformation modeling:
  - Energy is invariant under rigid transformations.
  - Bending, scaling, shearing is penalized.
  - Energy is non-quadratic (non-linear optimization required).
  - Matrix $\mathbf{M}$ encodes material properties (often $\mathbf{M} = \mathbf{I}$).
    - Important: read $\mathbf{M} \cdot [\ldots]$ as Matrix-*Vector* product

How to Detect Deformations?

Model

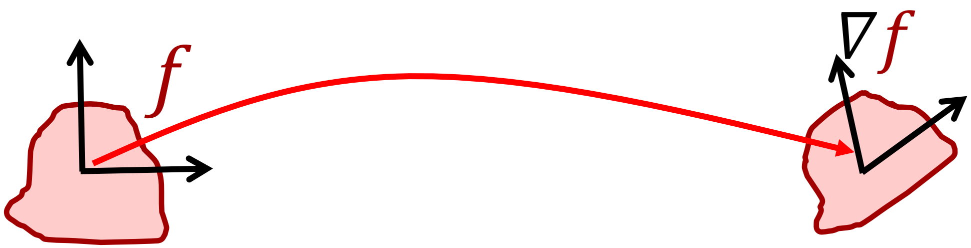
- Map volume to volume
- Function $f: V \to \mathbb{R}^3$



How to Detect Deformations?

Detect deformation

- Look at "deformation gradients"
- Jacobian matrix ∇f
- Function $\nabla f : V \to \mathbb{R}^3$



Criterion

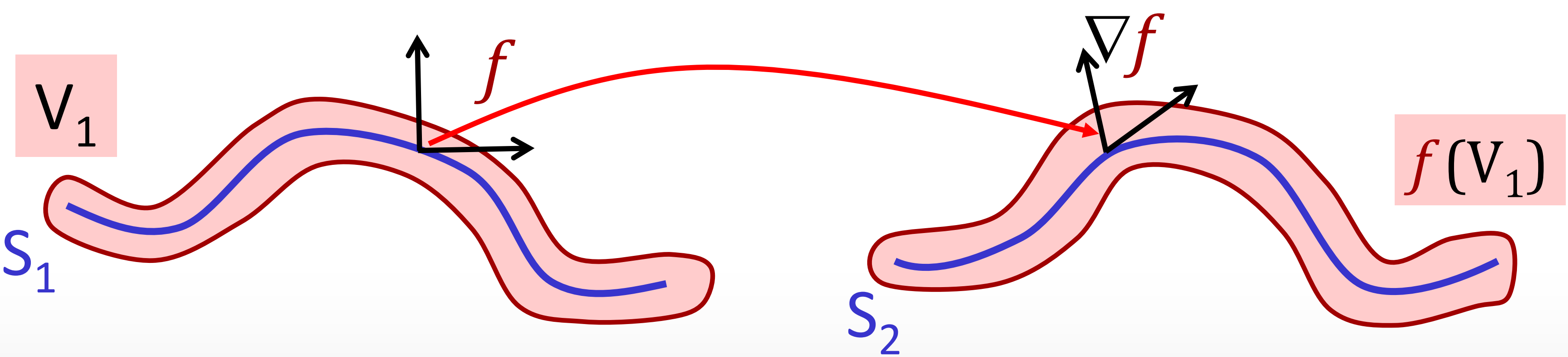
- *No deformation:* ∇f orthogonal
- *Deformation:* ∇f non-orthogonal

Elastic Volume Model

Extrinsic Volumetric "As-Rigid-As Possible"

- Measure orthogonality
- Integrate over deviation from orthogonality

$$E(f) = \int_{V_1} \left\| [\nabla f(\mathbf{x})][\nabla f(\mathbf{x})]^{\mathrm{T}} - \mathbf{I} \right\|_F^2 d\mathbf{x}$$



Functionals

Standard Functional #5: Volume preservation

- Given a function **f**: $\mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:

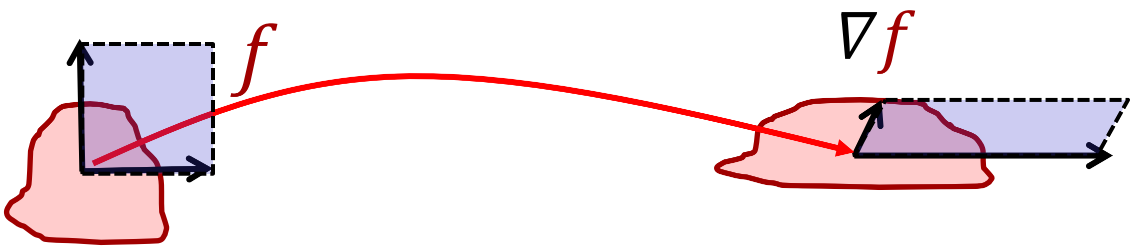
$$E^{(volume)}(\mathbf{f}) = \int_{\Omega} \left[\det(\nabla \mathbf{f}) - 1\right]^2 d\mathbf{x}$$

- Objective: minimize local volume changes
- This energy tries to preserve the volume at any point.
    - Physics: Incompressible materials (for example fluids)
    - The energy is invariant under rigid transformations.
    - This energy is non-quadratic (non-linear optimization required).
    - Often used in conjunction with deformation models.

Volume Preservation

Detect local change of volume

- Look at "deformation gradients"
- Jacobian matrix ∇f
- Function $\nabla f : V \to \mathbb{R}^3$



Criterion

- *Same volume:* ∇f maintains volume (= determinant)
- *Volume change:* $\det \nabla f$ changes

Functionals

Standard Functional #6: Infinitesimal volume preservation

- Given a function $\mathbf{v}: \mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n, \mathbf{v}(\mathbf{x}, t) = \frac{d}{dt}\mathbf{f}(\mathbf{x}, t)$
- Minimize:

$$E^{(volume)}(\mathbf{v}) = \int_\Omega \left(\mathrm{div}\, \mathbf{v}(\mathbf{x})\right)^2 d\mathbf{x} = \int_\Omega \left(\frac{\partial}{\partial x_1}\mathbf{v}_1(\mathbf{x}) + \cdots + \frac{\partial}{\partial x_n}\mathbf{v}_n(\mathbf{x})\right)^2 d\mathbf{x}$$

- Minimize local volume changes in a *velocity field*
- Difference to the previous case:
  - The vectors are instantaneous motions ($\mathbf{v}(\mathrm{x}) = \mathrm{d}/\mathrm{d}t\, \mathbf{f}(\mathbf{x},t)$)
  - A divergence free (time dependent) vector field will not introduce volume changes
  - This functional is linear, but does not work for large (rotational) displacements.

Functionals

Standard Functionals #7 & #8: Velocity & acceleration

- Given a function **v**: $(\mathbb{R}^n \times \mathbb{R}) \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:

$$E^{(velocity)}(\mathbf{f}) = \iint_\Omega \left(\frac{d}{dt} \mathbf{f}(\mathbf{x},t) \right)^2 d\mathbf{x}dt, \quad E^{(acc)}(\mathbf{f}) = \iint_\Omega \left(\frac{d^2}{dt^2} \mathbf{f}(\mathbf{x},t) \right)^2 d\mathbf{x}dt$$

- Objective: minimize velocity / acceleration
- Models air resistance, inertia.

Soft Constraints

Soft Constraints

Penalty functions

- Uniform
- General quadrics
- Differential constraints

Types of soft constraints

- Point-wise constraints
- Line / area constraints

Constraint functions

- Least-squares
- M-estimators

Uniform Soft Constraints

Uniform, point-wise soft constraints:

- Given a function $\mathbf{f}$: $\mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:

$$E^{(constr)}(\mathbf{f}) = \sum_{i=1}^{n} q_i \left(\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i\right)^2$$

constraint weights (certainty)

prescribed values $(\mathbf{x},\mathbf{y})_i$

Uniform Soft Constraints

General quadratic, point-wise soft constraints:

- Given a function $\mathbf{f}: \mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:

$$E^{(constr)}(\mathbf{f}) = \sum_{i=1}^{n} \left(\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i\right)^{\mathrm{T}} \mathbf{Q}_i \left(\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i\right)$$

constraint weights (general quadratic form, non-negative)

prescribed values $(\mathbf{x},\mathbf{y})_i$

Uniform Soft Constraints

Differential constraints:

- Given a function $\mathbf{f}: \mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:

$$E^{(constr)}(\mathbf{f}) = \sum_{i=1}^{n} \left(D\mathbf{f}(\mathbf{x}_i) - \left(D\mathbf{y}\right)_i\right)^{\mathrm{T}} \mathbf{Q}_i \left(D\mathbf{f}(\mathbf{x}_i) - \left(D\mathbf{y}\right)_i\right)$$

constraint weights (general quadratic form, non-negative)

prescribed values $(\mathbf{x}, D\mathbf{y})_i$

Differential operator: $D = \begin{pmatrix} \dfrac{\partial}{\partial x_{i_{1,1}} \ldots \partial x_{i_{k_1,1}}} \\ \vdots \\ \dfrac{\partial}{\partial x_{i_{1,m}} \ldots \partial x_{i_{k_m,m}}} \end{pmatrix}$

This is still a quadratic constraints ($\rightarrow$ linear system).

Examples

Examples of differential constraints:

- Prescribe normal orientation of a surface

$$\mathbf{f}:\mathbb{R}^2 \to \mathbb{R}^3, \quad E^{(constr)}(\mathbf{f}) = q\left(\begin{pmatrix} -\partial_u \\ -\partial_v \\ 1 \end{pmatrix}\mathbf{f} - \mathbf{n}\right)^2$$

- Prescribe rotation of a deformation field

$$\mathbf{f}:\mathbb{R}^3 \to \mathbb{R}^3, \quad E^{(constr)}(\mathbf{f}) = q\left\|\nabla\mathbf{f} - \mathbf{R}\right\|_F^2$$

- Prescribe velocity or acceleration of a particle trajectory

$$\mathbf{f}:\mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3, \mathbf{f}(\mathbf{x},t) = \mathbf{pos}, \quad E^{(constr)}(\mathbf{f}) = q(x,t)\left(\ddot{\mathbf{f}}(\mathbf{x},t) - \mathbf{a}(\mathbf{x},t)\right)^2$$

Line / Area Soft Constraints

Line and area constraints:

- Given a function **f**: $\mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:

$$E^{(constr)}(\mathbf{f}) = \int_{A \subseteq \Omega} \left(\mathbf{f}(\mathbf{x}) - \mathbf{y}(\mathbf{x})\right)^{\mathrm{T}} \mathbf{Q}(\mathbf{x}) \left(\mathbf{f}(\mathbf{x}) - \mathbf{y}(\mathbf{x})\right)$$

quadric error weights (may be position dependent)

prescribed values **y**(**x**) (function of position **x**)

area $A \subseteq \Omega$ on which the constraint is placed (line, area, volume...)

- A.k.a: "Transfinite Constraints"

Constraint Functions

Constraint Functions:

- Typically, we use quadratic constraints
  - $E(x) = f(x)^2$
  - Easy to optimize (linear system)
  - Well-defined critical point (gradient vanishes)
  - Sensitive to outliers
- Constraints come from measured data
  - E.g.: 3D scanner data
  - Quadratic constraints may case trouble

Constraint Functions

Constraint Functions:

- Alternatives:
  - L_1-norm constraints:
    - $E(x) = |f(x)|$
    - more robust and still convex, i.e. can be optimized
  - Non-convex, truncated constraints:
    - $E(x) = \min(|f(x)|, C), C>0$
    - yet more robust
    - finding a global optimum can be problematic
    - c.f. least-squares chapter

Discretization

Finite Element Discretization

Finite-element discretization:

- **Step 1:** Choose a finite dimensional function space
  - Spanned by basis functions
- **Step 2:** Compute optimum in that space only
- Finite differences (FD) is a special case
  - grid of piecewise constant basis functions
- General approach:

$$\arg\min_{f} E(f) \rightarrow \arg\min_{\lambda} E(\tilde{f}_\lambda)$$

$$\tilde{f}_\lambda(x) = \sum_{i=1}^{k} \lambda_i b_i(x)$$

Finite Element Discretization

Derive a discrete equation:

- Just plug in the discrete $\tilde{f}$.
- Then minimize the it over the λ.
- For a differentiable energy function, we compute the critical point(s):

$$E\left(\tilde{f}_\lambda(x)\right) \to \min$$

$$\Rightarrow \forall i = 1 \ldots k : \frac{\partial}{\partial \lambda_i} E\left(\tilde{f}_\lambda(x)\right) = 0$$

- For quadratic functionals, this leads to a linear system.
- For non-linear functionals, we can apply
  - Newton-optimization
  - Gradient descent
  - etc.

Example

(Abstract) example:

- Minimize square integral of a differential operator
- Quadratic differential soft constraints
- We obtain a quadratic optimization problem
  - The unknowns are the coefficients
    (coordinates in function basis)

Example

(Abstract) example (cont):

$$E(f) = \int_{\Omega} \left(D^{(1)} f(x)\right)^2 dx + \mu \sum_{i=1}^{n} \left(D^{(2)} f(x_i) - y_i\right)^2$$

$$\tilde{f}_{\lambda}(x) = \sum_{i=1}^{k} \lambda_i b_i(x)$$

$$E(\tilde{f}_{\lambda}) = \int_{\Omega} \left(D^{(1)} \sum_{i=1}^{k} \lambda_i b_i(x) \right)^2 dx + \mu \sum_{i=1}^{n} \left(D^{(2)} \sum_{i=1}^{k} \lambda_i b_i(x) - y_i \right)^2$$

$$= \int_{\Omega} \left(\sum_{i=1}^{k} \lambda_i \left[D^{(1)} b_i\right](x) \right)^2 dx + \mu \sum_{i=1}^{n} \left(\sum_{i=1}^{k} \lambda_i D^{(2)} b_i(x) - y_i \right)^2$$

$$= \sum_{i=1}^{k} \sum_{j=1}^{k} \lambda_i \lambda_j \int_{\Omega} \left[D^{(1)} b_i\right](x) \left[D^{(1)} b_j\right](x) dx + \mu \sum_{i=1}^{n} \left(\sum_{i=1}^{k} \lambda_i D^{(2)} b_i(x) - y_i \right)^2$$

Numerical Aspects

How to solve the problems?

Solving the discretized variational problem:

- Quadratic energy and quadratic constraints:
  - The discretization is a quadratic function as well.
  - The gradient is a linear expression.
  - The matrix in this expression is symmetric.
  - Well-defined problem => matrix is semi-positive definite
  - Usually very sparse matrix
    - coefficients of basis functions only interact with neighbors
    - depends on overlap of support
  - We can use iterative sparse system solvers:
    - frequently used: conjugate gradients (needs SPD matrix). CG is available in GeoX.

How to solve the problems?

Solving the discretized variational problem:

- Non linear energy functions:
  - If the function is convex, we can get to a critical point that is the global minimum.
  - In general, we can only find a local optimum (or critical point).
  - Frequently used techniques are:
    - Newton optimization:
      - Iteratively compute 2nd order Taylor expansions (Hessian matrix, gradient) and solve linear problems.
      - Typically, Hessian matrices are sparse. Use conjugate gradients to solve for critical points.
      - Variants – Quasi Newton: Gauss-Newton, (L)BFGS
    - Non-linear conjugate gradients with line search.
    - In any case, we need a *good initialization*.

Hard Constraints

Hard Constraints

Hard Constraints:

- Sometimes, we want some properties of the solution to be met *exactly* rather than *approximately*.
  - Interpolation vs. approximation
  - Includes complex constraints (area constraints, differential properties etc.)
- Three options to implement hard constraints:
  - Strong soft constraints (easy, but not exact)
  - Variable elimination (exact, but limited)
  - Lagrange multipliers (most complex method)

Hard Soft Constraints

Simplest Implementation:

- Use soft constraints with a large weight

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

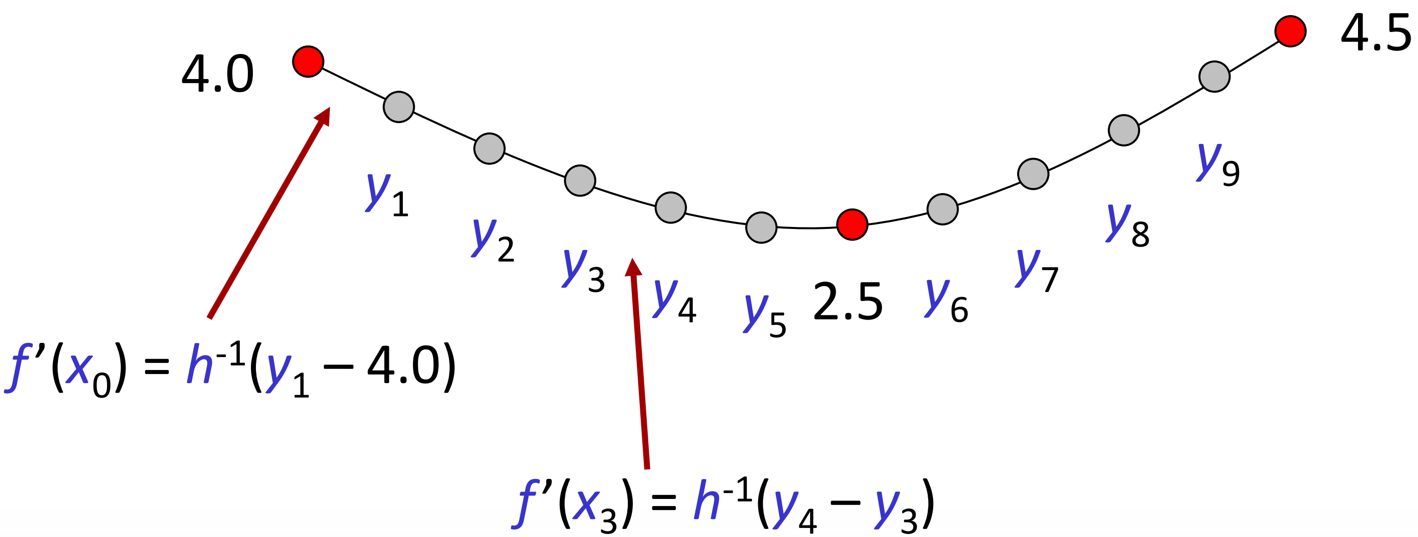
- This is simple to implement.
- A few serious problems:
  - The technique is not exact
    - For some applications this might be not acceptable.
  - The stronger the constraints, the larger the weight:
    - The condition number of the quadric matrix (condition of the Hessian in the non-linear case) becomes worse.
    - At some point, no solution is possible anymore.
    - Iterative solvers are slowed down (e.g. conjugate gradients)

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 so easy to implement generically)
- Does not work for FE methods (general basis functions)
  - Values at any point are *a sum* of scaled basis functions
- Does not work for complex constraints (area/integral constraints, differential constraints etc.)

Lagrange Multipliers

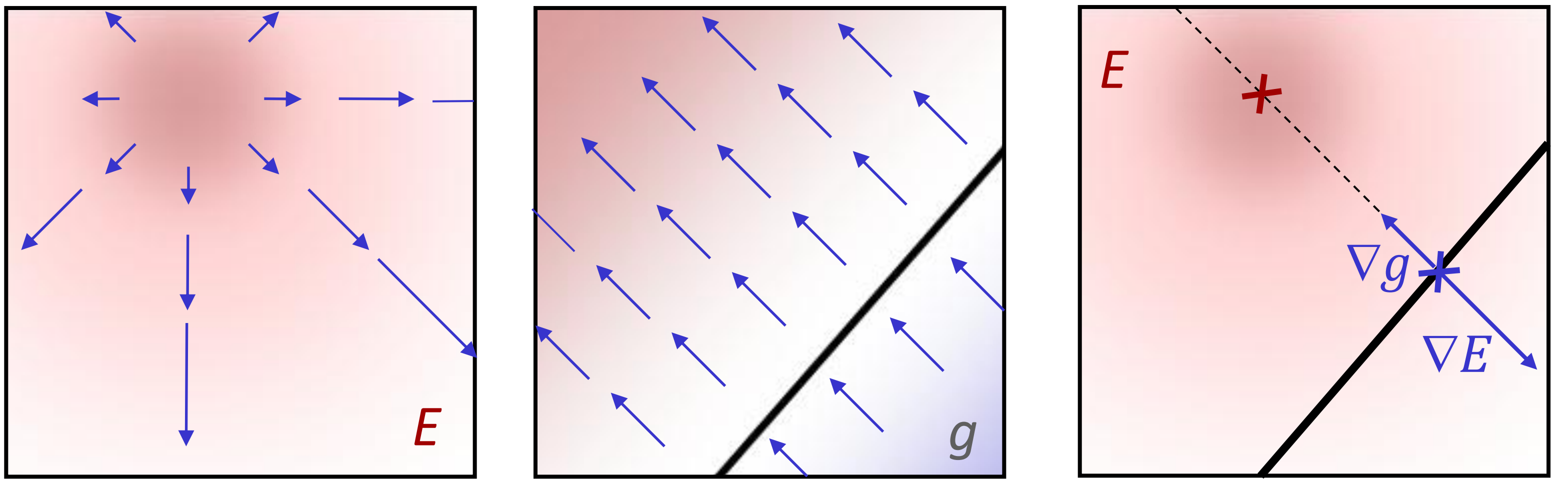
Most general technique: Lagrange multipliers

- This method works for complex, composite constraints
- No problems with general basis functions
  - Not restricted to finite difference discretizations
- The technique is exact.

Lagrange Multipliers

Here is the idea:

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

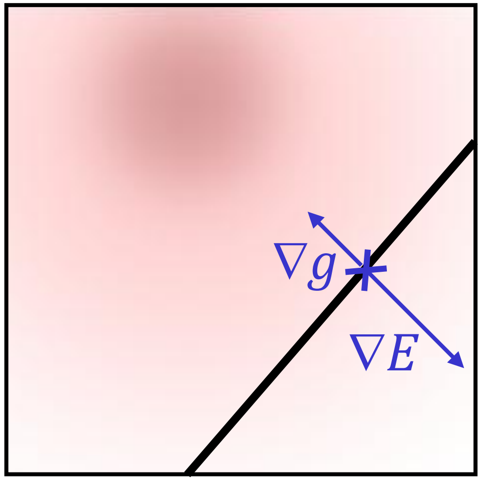


∇E $\quad$ ∇g $\quad$ $\nabla E = \lambda \nabla g,\ g(\mathbf{x}) = 0$

Lagrange Multipliers

Formally:

- Optimize $E(x_1, ..., x_n)$ subject to $g(x_1, ..., x_n) = 0$.
- Formally, we want:
  $\nabla E(\mathbf{x}) = \lambda \nabla g(\mathbf{x})$ and $g(\mathbf{x}) = 0$
- We get a local optimum for:
  $LG(\mathbf{x}) = E(\mathbf{x}) + \lambda g(\mathbf{x})$
  $\nabla_{\mathbf{x},\lambda} LG(\mathbf{x}) = 0$
  i.e.: $(\partial_{x_1}, ..., \partial_{x_n}, \partial_\lambda) LG(\mathbf{x}) = 0$
- 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 subject to a linear equality constraint

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

We obtain:

- $LG(\mathbf{x}) = E(\mathbf{x}) + \lambda \mathbf{g}(x) = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} + \mathbf{b}\mathbf{x} + \lambda(\mathbf{m}\mathbf{x} + n)$
  $\nabla_{\mathbf{x}}(LG(\mathbf{x})) = 2\mathbf{A}\mathbf{x} + \mathbf{b} + \lambda\mathbf{m}$
  $\nabla_{\lambda}(LG(\mathbf{x})) = \mathbf{m}\mathbf{x} + n$
- Linear system: $\begin{pmatrix} 2\mathbf{A} & \mathbf{m} \\ \mathbf{m}^{\mathrm{T}} & 0 \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} -b \\ -n \end{pmatrix}$

Multiple Constraints

Multiple Constraints:

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

$E(x) \to \min$

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

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

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

What can we do with this?

Multiple linear equality constraints:

- We can 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:
  - Need to make sure that constraints are linearly independent

What can we do with this?

Inequality constraints:

- There are efficient quadratic programming algorithms.
  - Idea: turn on and off the constraints intelligently.
- Examples:
  - Simplex method
  - Interior-point method

The Euler Lagrange Equation

(some more math)

The Euler-Lagrange Equation

Theoretical Result:

- An integral energy minimization problem can be reduced to a differential equation.
- We look at energy functions of a specific form:

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

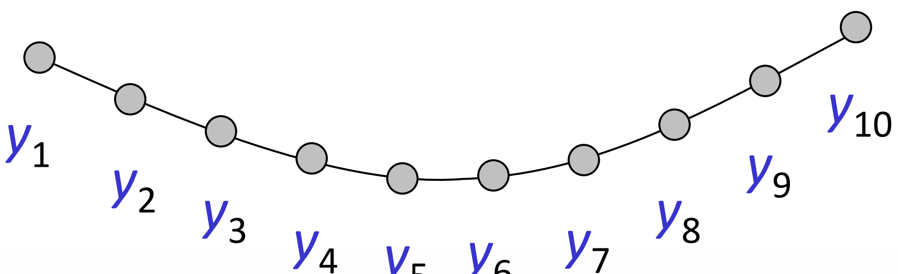
$$E(f) = \int_a^b F(x, f(x), f'(x))dx$$

  - f is the unknown function
  - F is the energy at each point x to be integrated
  - F depends (at most) on the position x, the function value $f(x)$ and the 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)
  - $f \mathrel{\hat{=}} ((x_1, y_1), \ldots, (x_n, y_n))$
  - Equally spaced: $x_i - x_{i-1} = h$

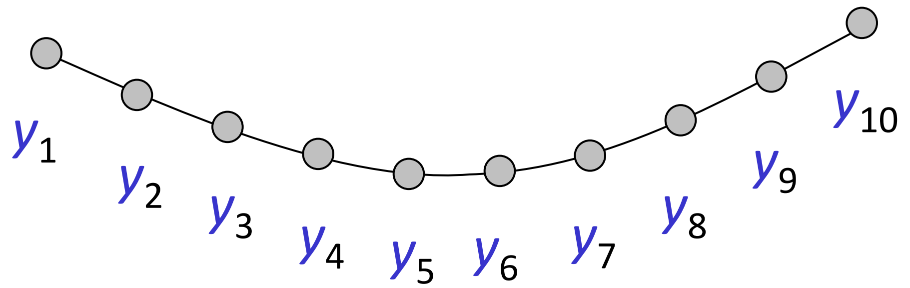


(Can be formalized more precisely using *functional derivatives*)

The Euler-Lagrange Equation

Minimum condition:

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



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

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

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

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

The Euler-Lagrange Equation

Minimum condition:

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

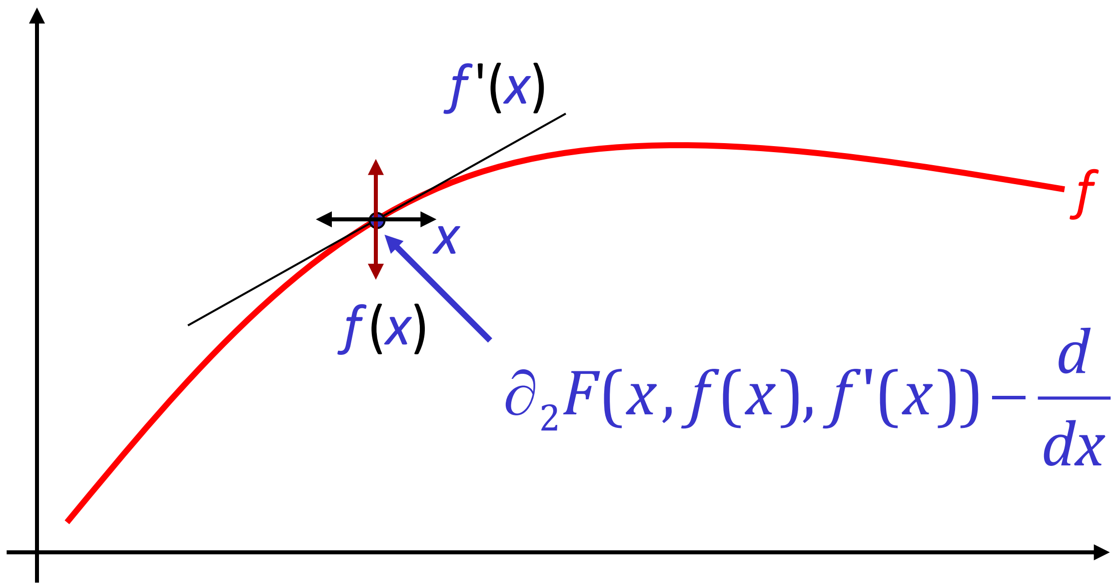
ith entry:

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

Letting $h \to 0$, we obtain the continuous Euler-Lagrange differential equation:

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

The Euler-Lagrange Equation



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

(at every point x)

Example

Example: Harmonic Energy

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

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

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

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

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

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

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 problem into ODE or PDE.
- Gives a necessary, but not sufficient condition for extremum (critical "point", read: function f)
- Application:
  - From a numerical point of view, no big difference:
    - We can directly optimize the integral expression
    - Same discrete system of equations
  - Analytical tool
    - Helps understanding the minimizer functions.

Surface Modeling

Applications

Variational Surface Modeling:

Two Examples:

- Parametric surfaces
  [Welch & Witkin: "Variational Surface Modeling", Siggraph 1992]
- Implicit surfaces
  [Turk, O'Brien: "Variational Implicit Surfaces.", TR, Georgia-Tec, 1999]

Parametric Surfaces

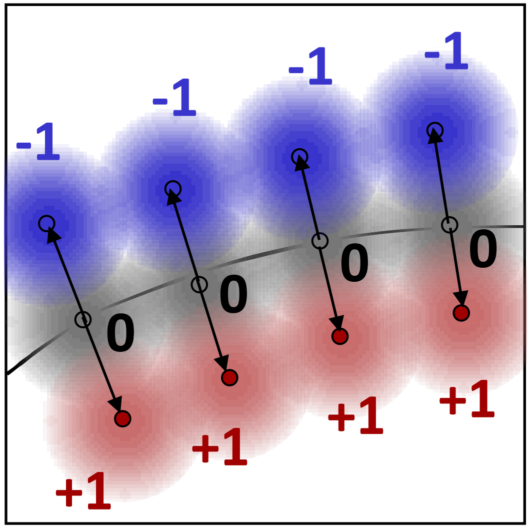
Domain:

- Parametric patch: f: $[0,1]^2 \to \mathbb{R}^3$.
- Representation (discretization):
  - Grid of uniform tensor-product B-Splines
  - Refine by dilated functions (subdivision) until convergence
- Energy:
  - Thin-plate-spline energy
- Constraints:
  - Points (soft / hard, langrange multipliers)
  - Transfinite constraints (curves, soft constraints only)
- Numerics:
  - Quadratic objective $\to$ solver sparse linear system

Implicit Surface

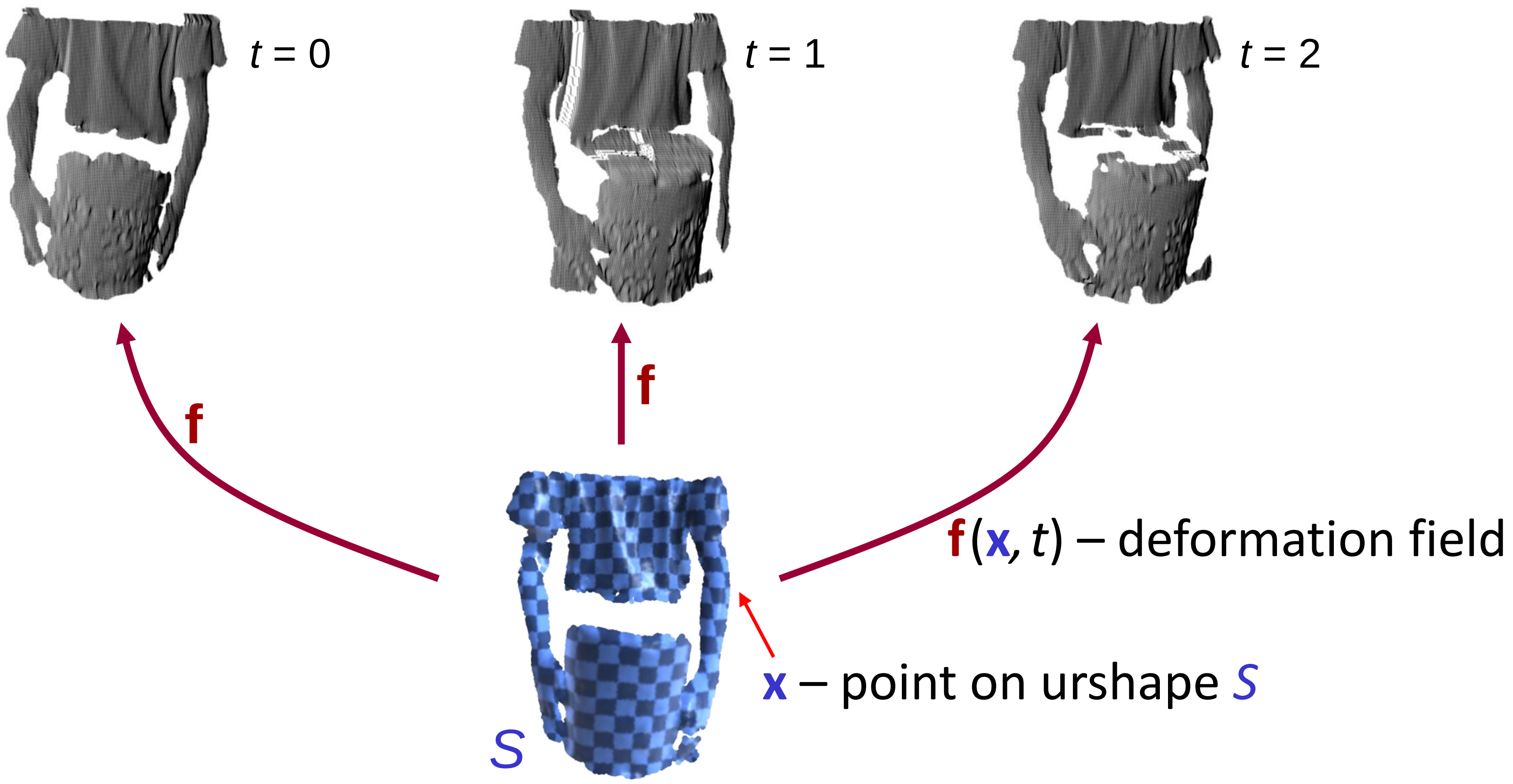
Domain:

- Implicit function: f: $[0,1]^3 \to \mathbb{R}$.
- Representation (discretization):
  - Radial basis functions of fundamental solutions
- Energy:
  - Thin-plate-spline energy
- Constraints:
  - Points with normals (hard, variable elimination)
- Numerics:
  - Radial basis functions around points and $\pm$ normal
  - Solve linear system for interpolation problem
  - Energy implicitly encoded in fundamental solutions



Other Applications

Variational Animation Modeling



Variational Framework

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

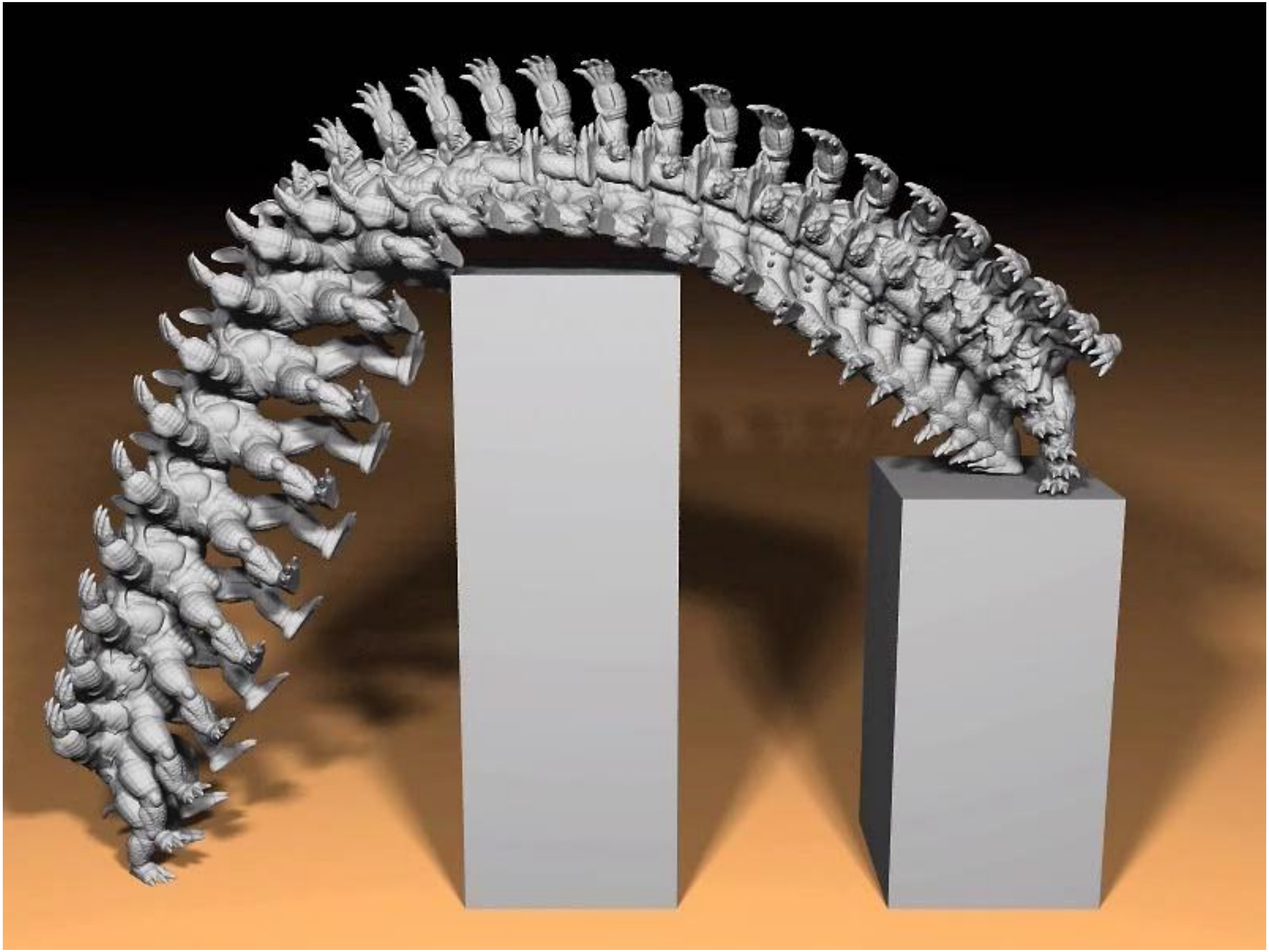
$$E_{match}(\mathbf{f}) = \sum_{t=1}^{T} \sum_{i=1}^{n_t} dist(d_i, f(S))^2$$

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

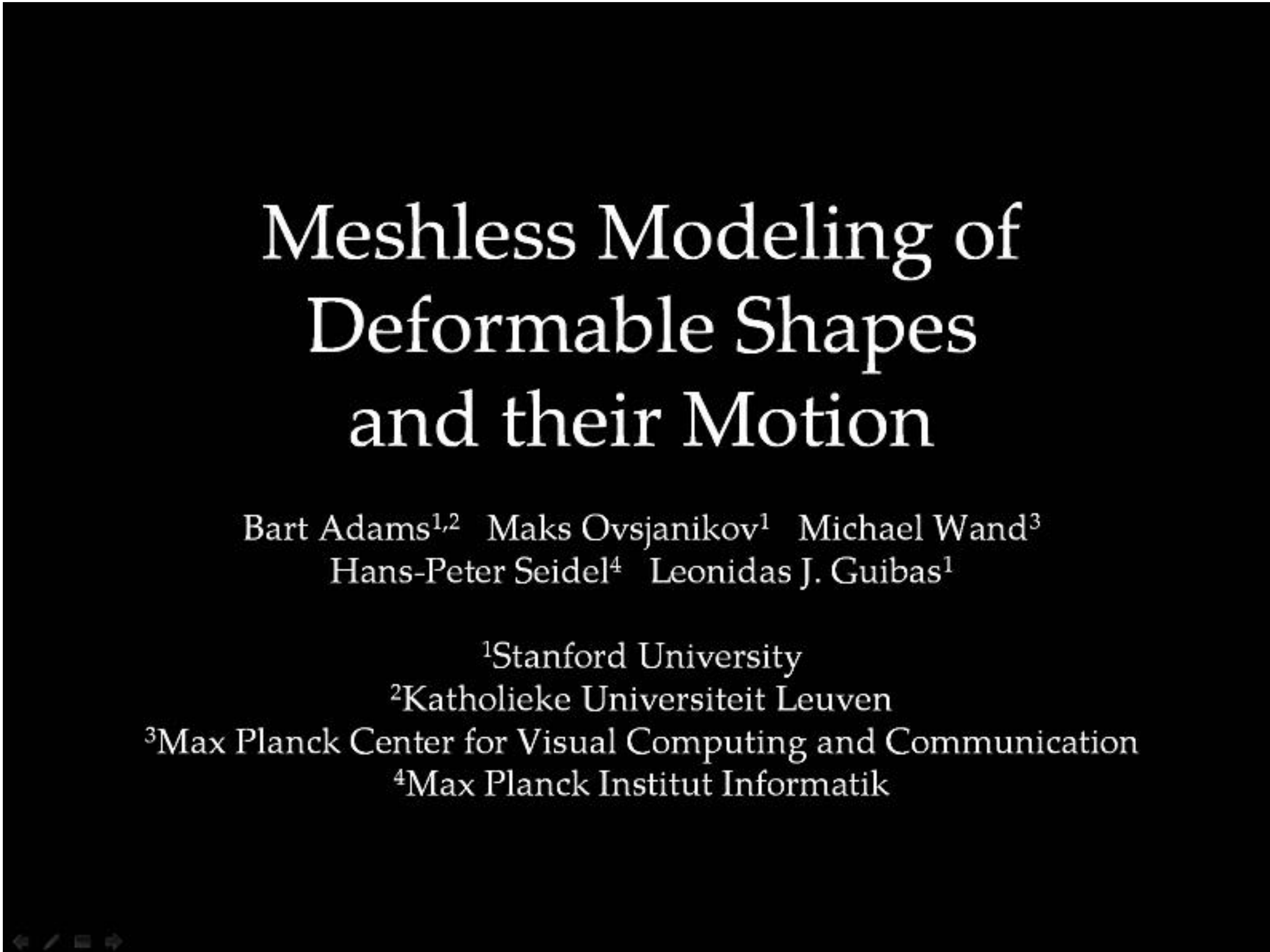
$$E_{volume}(\mathbf{f}) = \int_{V(S)} \omega_{vol}(x) \left(\left| \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x},t) \right| - 1 \right)^2 dx$$

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

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



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



[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.)