Computational methods in applied inverse problems

Uri Ascher

Department of Computer Science
University of British Columbia
October 2017

Four lectures

- Calibration and simulation of deformable objects
- Data manipulation and completion
- Estimating the trace of a large implicit matrix and applications
- Numerical analysis in visual computing: not too little, not too much

Visual computing

Visual computing: where the "eyeball-norm" rules

- Case studies
  - Two examples to start
  - Calibration and large-step time integration in elastodynamics
  - Image and surface processing
- Conclusions

Outline

- Case studies
    - Two examples to start
    - Calibration and large-step time integration in elastodynamics
    - Image and surface processing
- Conclusions

Visual computing: where the “eyeball-norm” rules

Numerical linear algebra in cloth simulation

- Cloth simulation:
  1. Use a mass-spring system
  2. Assemble large ODE system $M\ddot{\mathbf{q}}(t) = \mathbf{f}_{\rm els}(\mathbf{q}) + \mathbf{f}_{\rm dmp}(\mathbf{q}, \mathbf{v}) + \mathbf{f}_{\rm ext}$
  3. Elastic forces include stretching, bending and shearing. Stretching can be stiff.
- [Barraf & Witkin, '98]
  1. Simulate using large time steps, not reproducing rapid solution oscillations.
  2. Apply semi-implicit backward Euler (SI). To implement this, need to solve at each time step a linear algebraic system $A\mathbf{x} = \mathbf{b}$.
  3. Here $\mathbf{x} = \mathbf{v}_{n+1}$ are the particle velocities, some obeying the discretized ODE, others constrained.
  4. Devise modified preconditioned conjugate gradient (MPCG) method.
- [A.& Boxerman, '03]
  1. Explain and improve this MPCG method.
  2. *So here, math analysis has proved useful! Not only we understand B&W better and have a convergence proof, also the algorithm efficiency has been improved in the process.*

Numerical linear algebra in cloth simulation

- Cloth simulation:
  1. Use a mass-spring system
  2. Assemble large ODE system $M\ddot{\mathbf{q}}(t) = \mathbf{f}_{\rm els}(\mathbf{q}) + \mathbf{f}_{\rm dmp}(\mathbf{q}, \mathbf{v}) + \mathbf{f}_{\rm ext}$
  3. Elastic forces include stretching, bending and shearing. Stretching can be stiff.
- [Barraf & Witkin, '98]
  1. Simulate using large time steps, not reproducing rapid solution oscillations.
  2. Apply semi-implicit backward Euler (SI). To implement this, need to solve at each time step a linear algebraic system $A\mathbf{x} = \mathbf{b}$.
  3. Here $\mathbf{x} = \mathbf{v}_{n+1}$ are the particle velocities, some obeying the discretized ODE, others constrained.
  4. Devise modified preconditioned conjugate gradient (MPCG) method.
- [A.& Boxerman, '03]
  1. Explain and improve this MPCG method.
  2. *So here, math analysis has proved useful! Not only we understand B&W better and have a convergence proof, also the algorithm efficiency has been improved in the process.*

Numerical linear algebra in cloth simulation

- Cloth simulation:
  1. Use a mass-spring system
  2. Assemble large ODE system $M\ddot{\mathbf{q}}(t) = \mathbf{f}_{\rm els}(\mathbf{q}) + \mathbf{f}_{\rm dmp}(\mathbf{q}, \mathbf{v}) + \mathbf{f}_{\rm ext}$
  3. Elastic forces include stretching, bending and shearing. Stretching can be stiff.
- [Barraf & Witkin, '98]
  1. Simulate using large time steps, not reproducing rapid solution oscillations.
  2. Apply semi-implicit backward Euler (SI). To implement this, need to solve at each time step a linear algebraic system $A\mathbf{x} = \mathbf{b}$.
  3. Here $\mathbf{x} = \mathbf{v}_{n+1}$ are the particle velocities, some obeying the discretized ODE, others constrained.
  4. Devise modified preconditioned conjugate gradient (MPCG) method.
- [A.& Boxerman, '03]
  1. Explain and improve this MPCG method.
  2. *So here, math analysis has proved useful! Not only we understand B&W better and have a convergence proof, also the algorithm efficiency has been improved in the process.*

MPCG

- B & W defined 3×3 matrix "filters" S_i to eliminate components $\mathbf{v}_i^{(n+1)}$ in constrained directions. Set $S = \mathrm{diag}\{S_1, S_2, \ldots, S_N\}$ and solve $SA\mathbf{x} = S\mathbf{b}$ using a modified CG.
- A & B noticed that S is an orthogonal projection: can write

$$\mathbf{u} = S\mathbf{x}, \ \mathbf{w} = (I - S)\mathbf{x}, \quad \mathbf{x} = \mathbf{u} + \mathbf{w}, \ \mathbf{u}^T\mathbf{w} = 0.$$

- So, equations of motion hold in $\mathrm{range}(S)$, while solution is prescribed in $\mathrm{range}(I - S)$:

$$SA\mathbf{x} = S\mathbf{b}, \qquad (I - S)\mathbf{x} = (I - S)\mathbf{z}.$$

- To enable convergence theorem, had to assume initial MPCG starting point $\mathbf{x}_0 = S\mathbf{v}_n + (I - S)\mathbf{z}$.
  Compared to original guess $\mathbf{x}_0 = \mathbf{z}$ this improved convergence speed by 25-50%.

Pressure-Poisson equation (PPE)

- [Gundelman, Selle, Locasso & Fedkiw, '05] :
  "Coupling Water and Smoke to Thin Deformable and Rigid Shells"
- They devise a multi-splitting method, adding various aspects and contributions in sequence.
- Concentrate on fluid simulation: in a time step from t_n to t_{n+1},
  1. advect velocity $\mathbf{u}^n$ and add gravity to get $\mathbf{u}^*$,
  2. project $\mathbf{u}^*$ obtaining $\mathbf{u}^{n+1}$ by computing pressure to enforce incompressibility.
- What's behind this cryptic description?
- This step in itself can be considered as a splitting method.

Pressure-Poisson equation (PPE)

- [Gundelman, Selle, Locasso & Fedkiw, '05] :
  "Coupling Water and Smoke to Thin Deformable and Rigid Shells"
- They devise a multi-splitting method, adding various aspects and contributions in sequence.
- Concentrate on fluid simulation: in a time step from t_n to t_{n+1},
  1. advect velocity $\mathbf{u}^n$ and add gravity to get $\mathbf{u}^*$,
  2. project $\mathbf{u}^*$ obtaining $\mathbf{u}^{n+1}$ by computing pressure to enforce incompressibility.
- What's behind this cryptic description?
- This step in itself can be considered as a splitting method.

Incompressible Navier-Stokes (INS)

- For simplicity, write (simplified) equations in two space dimensions: $\mathbf{u} = (u, v)$–velocity, p–pressure, ν–viscosity constant.

$$\begin{aligned} u_t + uu_x + vu_y + p_x &= \nu \Delta u, \\ v_t + uv_x + vv_y + p_y &= \nu \Delta v, \\ u_x + v_y &= 0. \end{aligned}$$

- We know how to handle advection equations, so separate them from the rest by splitting.
- However, coupling in this PDE system is too strong: want 3rd eqn to be "for the pressure".
- Differentiate 1st eqn by x, 2nd by y, and add, using 3rd. Obtain pressure-Poission equation (PPE)

$$\Delta p \equiv p_{xx} + p_{yy} = -\left((u_x)^2 + 2u_y v_x + (v_y)^2 \right).$$

[Gresho & Sani '87; Sidilkover & A. '95]

Splitting INS

Assume $\nu = 0$ (good for crisp animations).

$$\begin{aligned} u_t + uu_x + vu_y + p_x &= 0, \\ v_t + uv_x + vv_y + p_y &= 0, \\ p_{xx} + p_{yy} &= -\left((u_x)^2 + 2u_y v_x + (v_y)^2\right). \end{aligned}$$

At time level n:

1. Solve for velocities

$$\begin{aligned} u_t + u^n u_x + v^n u_y + p_x^n &= 0, \\ v_t + u^n v_x + v^n v_y + p_y^n &= 0. \end{aligned}$$

   Call result $u^{n+1} = u,\ v^{n+1} = v$.

2. Solve for pressure

$$p_{xx} + p_{yy} = -\left((u_x^{n+1})^2 + 2u_y^{n+1} v_x^{n+1} + (v_y^{n+1})^2\right).$$

   Call result $p^{n+1} = p$.

Splitting INS

- The order of the split (fractional) steps matters.
- The first fractional step can be carried out using a semi-Lagrangian method, avoiding stability restriction on time step.
- The second fractional step involves solving Poisson's equation (PPE). The latter is the expensive item. Can in some situations be solved on a coarser spatial mesh.
- *Some mathematicians frown upon this method because the space in which solution resides changes and boundary condition issues arise.*
- *Nonetheless this is a favourite method in some computer graphics applications. Indeed, what may bother mathematicians is of no great concern in this physics-based setting.*

Splitting INS

- The order of the split (fractional) steps matters.
- The first fractional step can be carried out using a semi-Lagrangian method, avoiding stability restriction on time step.
- The second fractional step involves solving Poisson's equation (PPE). The latter is the expensive item. Can in some situations be solved on a coarser spatial mesh.
- *Some mathematicians frown upon this method because the space in which solution resides changes and boundary condition issues arise.*
- *Nonetheless this is a favourite method in some computer graphics applications. Indeed, what may bother mathematicians is of no great concern in this physics-based setting.*

Splitting INS

- The order of the split (fractional) steps matters.
- The first fractional step can be carried out using a semi-Lagrangian method, avoiding stability restriction on time step.
- The second fractional step involves solving Poisson's equation (PPE). The latter is the expensive item. Can in some situations be solved on a coarser spatial mesh.
- *Some mathematicians frown upon this method because the space in which solution resides changes and boundary condition issues arise.*
- *Nonetheless this is a favourite method in some computer graphics applications. Indeed, what may bother mathematicians is of no great concern in this physics-based setting.*

Outline

- Case studies
  - Two examples to start
  - Calibration and large-step time integration in elastodynamics
  - Image and surface processing
- Conclusions

Visual computing: where the “eyeball-norm” rules

Soft object simulation, calibration, control and fabrication

[Edwin Chen, Dinesh Pai; Danny Kaufman, Dave Levin; Bin Wang, Hui Huang]

- Ubiquitous in current computer graphics and robotics research.
- High quality simulations can be very expensive.
- The model typically requires calibration, e.g., specifying Young's modulus and damping properties.
- These are expressed as (distributed) parameters in the elastodynamics differential equations governing the motion.
- For control and fabrication may require more accurate simulations than before.

Soft object calibration and simulation

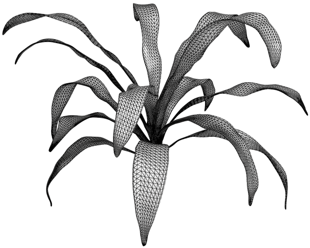
1. For a given calibration, semi-discretize elastodynamics equations in variational form using (co-rotated) FEM on a coarse moving tetrahedral mesh.
2. To obtain parameters (i.e., calibrate model), acquire position data in controlled environment and solve inverse problem.
3. Use physics-based simulation: in many applications, require result to look good, rather than be accurate to within **tol**. In particular:
   - It's the motion simulation results, rather than accuracy of parameters, that is eventually observed.
   - Can often use semi-implicit methods with large time steps to dampen invisible high oscillations.

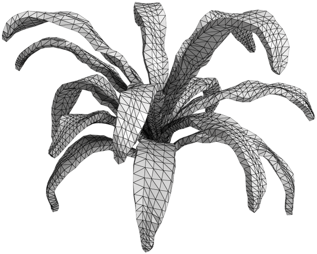
Soft object calibration and simulation

1. For a given calibration, semi-discretize elastodynamics equations in variational form using (co-rotated) FEM on a coarse moving tetrahedral mesh.
2. To obtain parameters (i.e., calibrate model), acquire position data in controlled environment and solve inverse problem.
3. Use physics-based simulation: in many applications, require result to look good, rather than be accurate to within **tol**. In particular:
   - It's the motion simulation results, rather than accuracy of parameters, that is eventually observed.
   - Can often use semi-implicit methods with large time steps to dampen invisible high oscillations.

Soft object calibration and simulation

1. For a given calibration, semi-discretize elastodynamics equations in variational form using (co-rotated) FEM on a coarse moving tetrahedral mesh.
2. To obtain parameters (i.e., calibrate model), acquire position data in controlled environment and solve inverse problem.
3. Use physics-based simulation: in many applications, require result to look good, rather than be accurate to within **tol**. In particular:
   - It's the motion simulation results, rather than accuracy of parameters, that is eventually observed.
   - Can often use semi-implicit methods with large time steps to dampen invisible high oscillations.





Fine surface mesh | Coarser volumetric mesh

[Wang, Wu, Yin, A., Liu & Huang '15]

Capture data

Motion tracking

Equations of motion

- Masses times accelerations equal forces ($\mathbf{v} = \dot{\mathbf{q}}$)

$$M\ddot{\mathbf{q}}(t) = \mathbf{f}_{\mathrm{els}}(\mathbf{q}) + \mathbf{f}_{\mathrm{dmp}}(\mathbf{q}, \mathbf{v}) + \mathbf{f}_{\mathrm{ext}},$$

  with the elastic and damping forces

$$\mathbf{f}_{\mathrm{els}}(\mathbf{q}) = -\frac{\partial}{\partial \mathbf{q}} W(\mathbf{q}(t)), \quad \mathbf{f}_{\mathrm{dmp}}(\mathbf{q}, \mathbf{v}) = -D\mathbf{v}(t),$$

  where $W(\mathbf{q}(t))$ is the elastic potential of the corresponding model.
- In a linear elasticity model, this elastic potential is quadratic.

A 1ST ORDER ODE SYSTEM

- Rewrite at some time $t = t_n$ as $\dot{\mathbf{u}}(t) = \mathbf{b}(\mathbf{u}(t))$:

$$\dot{\mathbf{u}}(t) \equiv \begin{pmatrix} \dot{\mathbf{q}}(t) \\ \dot{\mathbf{v}}(t) \end{pmatrix} = \begin{pmatrix} \mathbf{v} \\ M^{-1}\mathbf{f}_{\text{tot}}(\mathbf{q},\mathbf{v}) \end{pmatrix}$$
$$= \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix} \begin{pmatrix} \mathbf{q}(t) \\ \mathbf{v}(t) \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{g}(\mathbf{u}(t)) \end{pmatrix},$$

  where $K = -\frac{\partial}{\partial \mathbf{q}}\mathbf{f}_{\text{els}}(\mathbf{q})$ is the tangent stiffness matrix at $\mathbf{q} = \mathbf{q}(t)$.
- Often there is highly oscillatory stiffness, even though the observed motion is damped and does not vibrate rapidly. This happens when
  - the scale of the simulation is large, and/or
  - the material stiffens under large deformation.
- Another example is cloth simulation.

Large steps methods

- Want to use a time step size τ commensurate with the damped motion.
- Can't use explicit Runge-Kutta (RK) discretization.
- Moreover, implicit RK requires solution of nonlinear system at each step: can be nasty if the step size τ is large.
- Can use a semi-implicit (SI) method, i.e., backward Euler (BE) with only one Newton iteration at each time step starting from $\mathbf{u}_n$. [Baraff & Witkin '98; A. '08].
  This is by far the most popular method in use to date.
- However, heavy step-size dependent damping is introduced: not easy for an artist to work with; and affects different materials differently.
- Why does it work at all?

Large steps methods

- Want to use a time step size τ commensurate with the damped motion.
- Can't use explicit Runge-Kutta (RK) discretization.
- Moreover, implicit RK requires solution of nonlinear system at each step: can be nasty if the step size τ is large.
- Can use a semi-implicit (SI) method, i.e., backward Euler (BE) with only one Newton iteration at each time step starting from $\mathbf{u}_n$. [Baraff & Witkin '98; A. '08].
  This is by far the most popular method in use to date.
- However, heavy step-size dependent damping is introduced: not easy for an artist to work with; and affects different materials differently.
- Why does it work at all?

Large steps methods

- Want to use a time step size τ commensurate with the damped motion.
- Can't use explicit Runge-Kutta (RK) discretization.
- Moreover, implicit RK requires solution of nonlinear system at each step: can be nasty if the step size τ is large.
- Can use a semi-implicit (SI) method, i.e., backward Euler (BE) with only one Newton iteration at each time step starting from $\mathbf{u}_n$. [Baraff & Witkin '98; A. '08].
  This is by far the most popular method in use to date.
- However, heavy step-size dependent damping is introduced: not easy for an artist to work with; and affects different materials differently.
- Why does it work at all?

Time integration of elastodynamics equations

- Backward Euler (BE), semi-implicit (SI) and stabilized SI.
- Symplectic: explicit leap-frog, implicit midpoint (IM)
- Energy and/or momentum conserving
- Mixing: θ methods
- Newmark and Generalized α
- Exponential time differencing (like SI, no need to solve nonlinear equations)
- Decoupling of fast and slow scales.

Time integration of elastodynamics equations

- Backward Euler (BE), semi-implicit (SI) and stabilized SI.
- Symplectic: explicit leap-frog, implicit midpoint (IM)
- Energy and/or momentum conserving
- Mixing: θ methods
- Newmark and Generalized α
- Exponential time differencing (like SI, no need to solve nonlinear equations)
- Decoupling of fast and slow scales.

Time integration of elastodynamics equations

- Backward Euler (BE), semi-implicit (SI) and stabilized SI.
- Symplectic: explicit leap-frog, implicit midpoint (IM)
- Energy and/or momentum conserving
- Mixing: θ methods
- Newmark and Generalized α
- Exponential time differencing (like SI, no need to solve nonlinear equations)
- Decoupling of fast and slow scales.

Time integration of elastodynamics equations

- Backward Euler (BE), semi-implicit (SI) and stabilized SI.
- Symplectic: explicit leap-frog, implicit midpoint (IM)
- Energy and/or momentum conserving
- Mixing: θ methods
- Newmark and Generalized α
- Exponential time differencing (like SI, no need to solve nonlinear equations)
- Decoupling of fast and slow scales.

Standard integration methods

$$\begin{aligned} \dot{\mathbf{u}} &= \mathbf{b}(\mathbf{u}) \\ &= J\mathbf{u} + \mathbf{c}(\mathbf{u}), \text{ where } J = \frac{\partial \mathbf{b}}{\partial \mathbf{u}}. \end{aligned}$$

Step from $t = t_n$ to $t = t_{n+1} = t_n + \tau$.

- Backward Euler (BE)

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \tau \mathbf{b}(\mathbf{u}_{n+1})$$

- Implicit midpoit (IM)

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \tau \mathbf{b}((\mathbf{u}_{n+1} + \mathbf{u}_n)/2)$$

- Semi-implicit backward Euler (SI):
  Apply to BE just one *Newton iteration* starting at $\mathbf{u}_n$ towards solving for $\mathbf{u}_{n+1}$.

Standard integration methods

$$\begin{aligned}\dot{\mathbf{u}} &= \mathbf{b}(\mathbf{u}) \\ &= J\mathbf{u} + \mathbf{c}(\mathbf{u}), \text{ where } J = \frac{\partial \mathbf{b}}{\partial \mathbf{u}}.\end{aligned}$$

Step from $t = t_n$ to $t = t_{n+1} = t_n + \tau$.

- Backward Euler (BE)

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \tau \mathbf{b}(\mathbf{u}_{n+1})$$

- Implicit midpoit (IM)

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \tau \mathbf{b}((\mathbf{u}_{n+1} + \mathbf{u}_n)/2)$$

- Semi-implicit backward Euler (SI):
  Apply to BE just one *Newton iteration* starting at $\mathbf{u}_n$ towards solving for $\mathbf{u}_{n+1}$.

Standard integration methods

$$\begin{aligned} \dot{\mathbf{u}} &= \mathbf{b}(\mathbf{u}) \\ &= J\mathbf{u} + \mathbf{c}(\mathbf{u}), \text{ where } J = \frac{\partial \mathbf{b}}{\partial \mathbf{u}}. \end{aligned}$$

Step from $t = t_n$ to $t = t_{n+1} = t_n + \tau$.

- Backward Euler (BE)

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \tau \mathbf{b}(\mathbf{u}_{n+1})$$

- Implicit midpoit (IM)

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \tau \mathbf{b}((\mathbf{u}_{n+1} + \mathbf{u}_n)/2)$$

- Semi-implicit backward Euler (SI):
  Apply to BE just one *Newton iteration* starting at $\mathbf{u}_n$ towards solving for $\mathbf{u}_{n+1}$.

Standard integration methods

$$\begin{aligned}\dot{\mathbf{u}} &= \mathbf{b}(\mathbf{u}) \\ &= J\mathbf{u} + \mathbf{c}(\mathbf{u}), \text{ where } J = \frac{\partial \mathbf{b}}{\partial \mathbf{u}}.\end{aligned}$$

Step from $t = t_n$ to $t = t_{n+1} = t_n + \tau$.

- Backward Euler (BE)

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \tau \mathbf{b}(\mathbf{u}_{n+1})$$

- Implicit midpoit (IM)

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \tau \mathbf{b}((\mathbf{u}_{n+1} + \mathbf{u}_n)/2)$$

- Semi-implicit backward Euler (SI):
  Apply to BE just one *Newton iteration* starting at $\mathbf{u}_n$ towards solving for $\mathbf{u}_{n+1}$.

Exponential Rosenbrock Euler (ERE)

[Chen, A. & Pai, '17]

Approximate

$$\mathbf{u}(t_{n+1}) = \exp(\tau J)\mathbf{u}_n + \int_{t_n}^{t_{n+1}} \exp((t_{n+1} - s)J)\mathbf{c}(\mathbf{u}(s))ds$$

to obtain

$$\begin{aligned} \mathbf{u}_{n+1} &= \exp(\tau J_n)\mathbf{u}_n + \tau\phi_1(\tau J_n)\mathbf{c}_n(\mathbf{u}_n) \\ &= \mathbf{u}_n + \tau\phi_1(\tau J_n)\mathbf{b}(\mathbf{u}_n) \end{aligned}$$

with $\phi_1(z) = z^{-1}(\exp(z) - 1)$.

- This basic form does not require the elastic energy to be convex.
- Can carry out step through evaluating

$$\mathbf{u}_{n+1} = \begin{bmatrix} I_N & 0_{N\times 1} \end{bmatrix} \exp(\tau A_n)\,\tilde{\mathbf{u}}_n, \quad \text{where}$$

$$A_n = \begin{bmatrix} J_n & \mathbf{c}_n(\mathbf{u}_n) \\ 0_{1\times N} & 0 \end{bmatrix}, \quad \tilde{\mathbf{u}}_n = \begin{bmatrix} \mathbf{u}_n \\ 1 \end{bmatrix},$$

Exponential Rosenbrock Euler (ERE)

[Chen, A. & Pai, '17]

Approximate

$$\mathbf{u}(t_{n+1}) = \exp(\tau J)\mathbf{u}_n + \int_{t_n}^{t_{n+1}} \exp((t_{n+1} - s)J)\mathbf{c}(\mathbf{u}(s))ds$$

to obtain

$$\begin{aligned} \mathbf{u}_{n+1} &= \exp(\tau J_n)\mathbf{u}_n + \tau\phi_1(\tau J_n)\mathbf{c}_n(\mathbf{u}_n) \\ &= \mathbf{u}_n + \tau\phi_1(\tau J_n)\mathbf{b}(\mathbf{u}_n) \end{aligned}$$

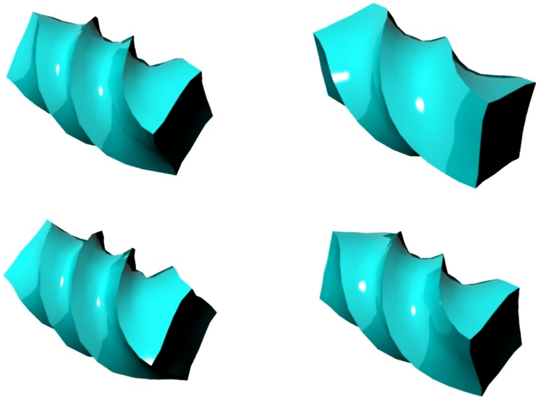
with $\phi_1(z) = z^{-1}(\exp(z) - 1)$.

- This basic form does not require the elastic energy to be convex.
- Can carry out step through evaluating

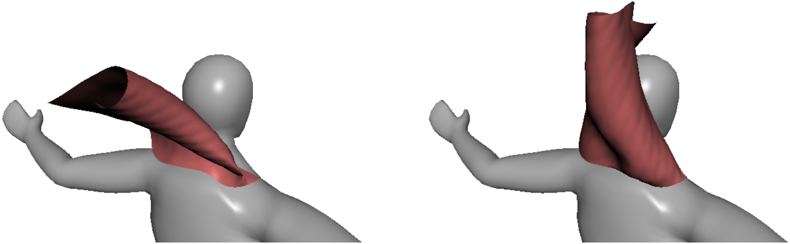
$$\mathbf{u}_{n+1} = \begin{bmatrix} I_N & 0_{N\times 1} \end{bmatrix} \exp\left(\tau A_n\right) \tilde{\mathbf{u}}_n, \quad \text{where}$$

$$A_n = \begin{bmatrix} J_n & \mathbf{c}_n(\mathbf{u}_n) \\ 0_{1\times N} & 0 \end{bmatrix}, \quad \tilde{\mathbf{u}}_n = \begin{bmatrix} \mathbf{u}_n \\ 1 \end{bmatrix},$$

Twisted bar: neo-Hookean material

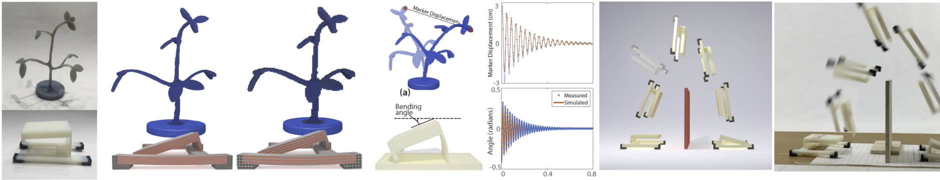


Dancer cape



Controlling soft objects

[Chen, Levin, Matusik & Kaufman, '17]



Analysis for the simplest case

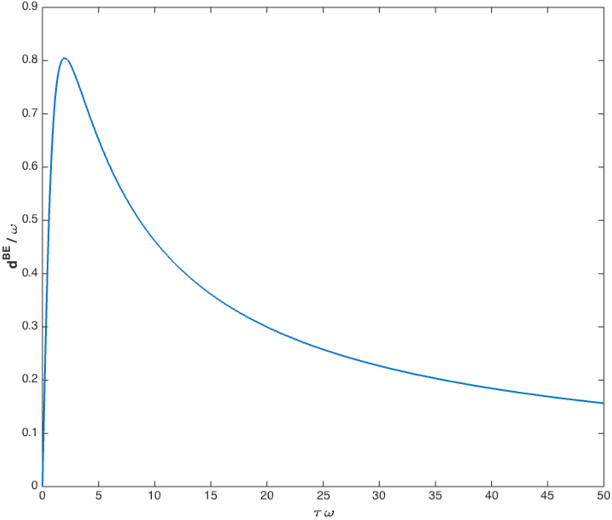
- Consider the scalar constant-coefficient ODE

$$\ddot{q} + d\dot{q} + \omega^2 q = 0,$$

 where $d \geq 0$ is a damping parameter, and $\omega > d/2$ is a real-valued frequency.

- Setting $d = 0$ apply numerical discretization.
- Associate resulting decay with artificial damping factor d^{method}.

BE ARTIFICIAL DAMPING CURVE



Energy and momentum conserving variants

- Implicit midpoint and all other conservative methods do not have artificial damping for the simpliest case: $d^{\mathrm{method}} \equiv 0$ when $d = 0$.
- Can sacrifice symplecticity but gain energy conservation in time. e.g., average vector field (AVF) methods.
- Can have an implicit Newmark midpoint-trapezoidal variant [Kane, Marsden, Ortiz & West '00] that is symplectic and conserves momentum when $D = 0$:

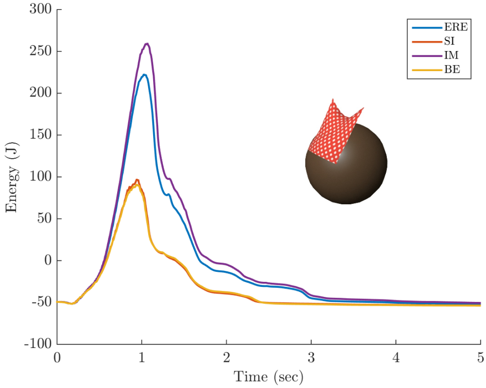
$$\begin{aligned}
\mathbf{v}_{n-1/2} &= \mathbf{v}_{n-1} - \frac{\tau}{4}(K_n\mathbf{q}_n + K_{n-1}\mathbf{q}_{n-1} + D_n\mathbf{v}_n + D_{n-1}\mathbf{v}_{n-1}) \\
&+ \frac{\tau}{2}\mathbf{g}_{n-1/2}, \\
\mathbf{v}_n &= 2\mathbf{v}_{n-1/2} - \mathbf{v}_{n-1}, \\
\mathbf{q}_n &= \mathbf{q}_{n-1} + \tau\mathbf{v}_{n-1/2}.
\end{aligned}$$

- These methods can be extended for problems with damping $D > 0$.

Cloth after colliding with a sphere

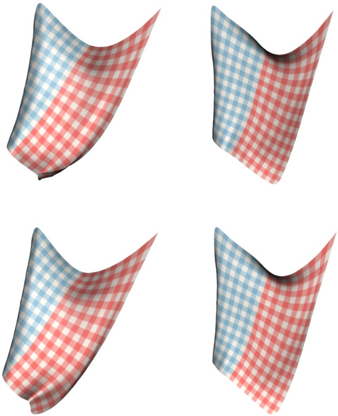
Top – left: ERE, right: SI
Bottom – left: IM, right: BE

Example cont.



Energy profile of each method in the simulation with cloth collision.

Cloth with mixed stiffness after colliding with a sphere



Top – left: ERE, right: SI
Bottom – left: IM, right: BE

Need them all

- Why not just discard SI and BE, concentrating on the good stuff?
- Because conservative methods quietly require the step size to be "large but not larger" ($\tau = \mathcal{O}(1/\omega)$ but not $\tau^2 = \mathcal{O}(1/\omega)$) [A. & Reich, 1999]
- Exponential method also loses charm when stiffness is too high
- So what can we do?
- Reduce artificial damping by mixing methods
- Decouple fast and slow scales
- Pray

Need them all

- Why not just discard SI and BE, concentrating on the good stuff?
- Because conservative methods quietly require the step size to be "large but not larger" ($\tau = \mathcal{O}(1/\omega)$ but not $\tau^2 = \mathcal{O}(1/\omega)$) [A. & Reich, 1999]
- Exponential method also loses charm when stiffness is too high
- So what can we do?
- Reduce artificial damping by mixing methods
- Decouple fast and slow scales
- Pray

Need them all

- Why not just discard SI and BE, concentrating on the good stuff?
- Because conservative methods quietly require the step size to be "large but not larger" ($\tau = \mathcal{O}(1/\omega)$ but not $\tau^2 = \mathcal{O}(1/\omega)$) [A. & Reich, 1999]
- Exponential method also loses charm when stiffness is too high
- So what can we do?
- Reduce artificial damping by mixing methods
- Decouple fast and slow scales
- Pray

Need them all

- Why not just discard SI and BE, concentrating on the good stuff?
- Because conservative methods quietly require the step size to be "large but not larger" ($\tau = \mathcal{O}(1/\omega)$ but not $\tau^2 = \mathcal{O}(1/\omega)$) [A. & Reich, 1999]
- Exponential method also loses charm when stiffness is too high
- So what can we do?
- Reduce artificial damping by mixing methods
- Decouple fast and slow scales
- Pray

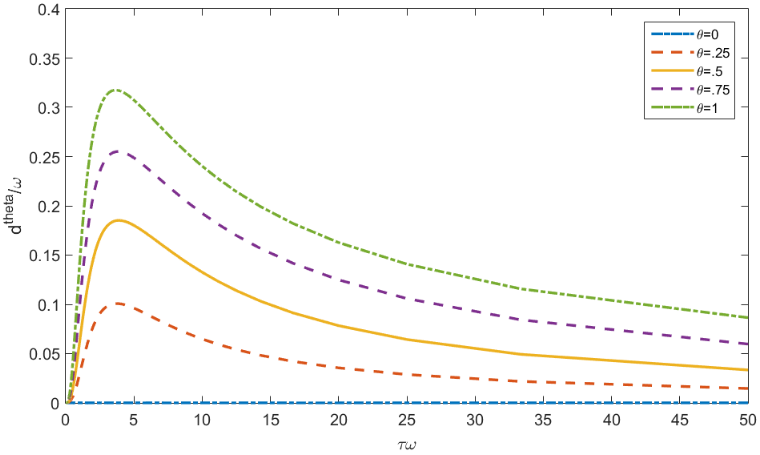
A second order θ method

BDF2 has similar characteristics to BE (although less damping), and mixing it with IM retains 2nd order accuracy:

$$\begin{pmatrix} \mathbf{q}_n \\ \mathbf{v}_n \end{pmatrix} = \begin{pmatrix} \mathbf{q}_{n-1} \\ \mathbf{v}_{n-1} \end{pmatrix} + \tau[\theta * (\mathrm{BDF2}) + (1-\theta) * (\mathrm{IM})],$$

Moreover, the use of a two-step method can often be accommodated in computer graphics applications.
Damping plots for $\theta = 0 : .25 : 1$ (the larger θ, the more damping):

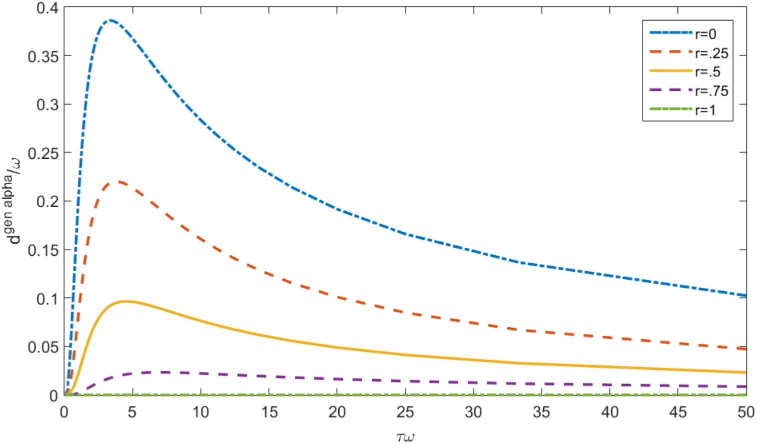
Artificial damping when mixing implicit midpoint and BDF2



Generalized α method

- Mechanical engineers often use the generalized α method [Chung & Hullbert, '93, Kobis & Arnold, '16] rather than backward Euler.
- It is a one-step Newmark-type method (discretize $\dot{\mathbf{v}} = \mathbf{a}, \; M\mathbf{a} = \mathbf{f}(\mathbf{q}, \mathbf{v})$, rather than $\dot{\mathbf{v}} = M^{-1}\mathbf{f}(\mathbf{q}, \mathbf{v})$).
- It has a parameter $r = 1 - \theta$ that can be tuned to select anywhere between BE-like strong damping of high frequencies and no damping at all.
- For any choice of $0 \leq r \leq 1$ the method is second order accurate.
- The size of nonlinear system to solve at each step is minimal.

Generalized α artificial damping curve



Generalized α (GA) curves d^{GA}/ω as a function of $\tau\omega$, for $r = 0 : .25 : 1$

So...

We have just seen a case study where the need to move from qualitative *to* quantitative *has caused the computer graphics community to get closer to the works of numerical analysts and applied mathematicians, e.g., in geometric integration.*

OUTLINE

- Case studies
    - Two examples to start
    - Calibration and large-step time integration in elastodynamics
    - Image and surface processing
- Conclusions

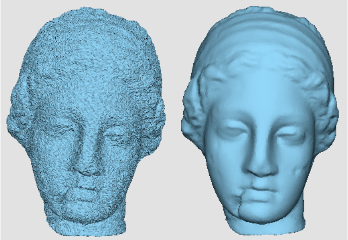
Visual computing: where the "eyeball-norm" rules

Image and surface processing

denoising, deblurring, inpainting, completion, salient features...

- Simplest example: denoising an image

- Next simplest: denoising a surface triangle mesh



Using PDE-based penalties

- Many researchers have considered regularization using diffusion or anisotropic diffusion: Given image b, find image u that solves

$$\begin{aligned} \min_u \quad & \frac{1}{2}\|f(u) - b\|^2 + \beta R(u), \\ R(u) \quad = \quad & \int_\Omega \rho(|\nabla u|), \quad \beta > 0, \end{aligned}$$

  - diffusion: $\rho(s) = s^2$ (ℓ_2 on gradient)
  - anisotropic diffusion: $\rho(s) = s$ (ℓ_1 on gradient, i.e., total variation)
  - combination of these two (e.g., Huber [A., Haber & Huang '06])
- A huge amount of literature follows this line, e.g., [Perona & Malik '90, Rudin,Osher & Fatemi '91, Weickert '98... Chan et al., ... A., Huang et al. ...; Desbrun et al.'99,'00 , Hildebrandt & Polthier '04]

Using PDE-based penalties

- Many researchers have considered regularization using diffusion or anisotropic diffusion: Given image b, find image u that solves

$$\begin{aligned} \min_u & \quad \frac{1}{2}\|f(u) - b\|^2 + \beta R(u), \\ R(u) &= \int_\Omega \rho(|\nabla u|), \;\; \beta > 0, \end{aligned}$$

  - diffusion: $\rho(s) = s^2$ (ℓ_2 on gradient)
  - anisotropic diffusion: $\rho(s) = s$ (ℓ_1 on gradient, i.e., total variation)
  - combination of these two (e.g., Huber [A., Haber & Huang '06])
- A huge amount of literature follows this line, e.g., [Perona & Malik '90, Rudin,Osher & Fatemi '91, Weickert '98... Chan et al., ... A., Huang et al. ...; Desbrun et al.'99,'00 , Hildebrandt & Polthier '04]

But is this always the best approach?

- Indirect: start with discrete image b, $\uparrow$ move to function spaces, $\rightarrow$ manipulate there, $\downarrow$ obtain discrete image u.
- Uses a global, not local prior.

Paradigm:

$$\min_u \quad \frac{1}{2}\|f(u) - b\|^2 + \beta R(u),$$
$$R(u) = \int_\Omega \rho(|\nabla u|), \quad \beta > 0,$$

$\rho(s) = s^2$ or $\rho(s) = s$ or $\rho(s)$ is a combination of the two.

But is this always the best approach?

- Advantage: can often obtain a more solid theoretical backing to algorithms.
- Disadvantage: may be outperformed by more brute force techniques, especially if forward operator f is simple and data b is of high quality.

Paradigm:

$$\begin{aligned} \min_u \quad & \frac{1}{2}\|f(u) - b\|^2 + \beta R(u), \\ R(u) \quad = \quad & \int_\Omega \rho(|\nabla u|), \quad \beta > 0, \end{aligned}$$

$\rho(s) = s^2$ or $\rho(s) = s$ or $\rho(s)$ is a combination of the two.

Examples where this paradigm is worthwhile

Basically, whenever the advantages outweighs the disadvanges...

- If the forward operator $f(u)$ itself contains differential terms.
  But let's concentrate on cases where the differential operator in the penalty dominates other differential terms.
- If the model approximation in $f(u)$ or the data b (or both) are not of high quality; e.g., blind deconvolution, time-of-flight data [Heide ... Heidrich '16].
- Where the paradigm is used to generate primitives for graphics use. e.g., "kelvinlets" [de Goes & James, '17].

Paradigm:

$$\min_u \quad \frac{1}{2}\|f(u) - b\|^2 + \beta R(u),$$

$$R(u) = \int_\Omega \rho(|\nabla u|), \quad \beta > 0,$$

Examples where this paradigm is worthwhile

Basically, whenever the advantages outweighs the disadvanges...

- If the forward operator $f(u)$ itself contains differential terms. But let's concentrate on cases where the differential operator in the penalty dominates other differential terms.
- If the model approximation in $f(u)$ or the data b (or both) are not of high quality; e.g., blind deconvolution, time-of-flight data [Heide ... Heidrich '16].
- Where the paradigm is used to generate primitives for graphics use. e.g., "kelvinlets" [de Goes & James, '17].

Paradigm:

$$\min_u \quad \frac{1}{2}\|f(u) - b\|^2 + \beta R(u),$$

$$R(u) = \int_\Omega \rho(|\nabla u|), \quad \beta > 0,$$

Examples where this paradigm is worthwhile

Basically, whenever the advantages outweighs the disadvanges...

- If the forward operator $f(u)$ itself contains differential terms. But let's concentrate on cases where the differential operator in the penalty dominates other differential terms.
- If the model approximation in $f(u)$ or the data b (or both) are not of high quality; e.g., blind deconvolution, time-of-flight data [Heide ... Heidrich '16].
- Where the paradigm is used to generate primitives for graphics use. e.g., "kelvinlets" [de Goes & James, '17].

Paradigm:

$$\min_u \quad \frac{1}{2}\|f(u) - b\|^2 + \beta R(u),$$

$$R(u) \quad = \quad \int_\Omega \rho(|\nabla u|), \quad \beta > 0,$$

Examples where this paradigm is worthwhile

Basically, whenever the advantages outweighs the disadvanges...

- If the forward operator $f(u)$ itself contains differential terms. But let's concentrate on cases where the differential operator in the penalty dominates other differential terms.
- If the model approximation in $f(u)$ or the data b (or both) are not of high quality; e.g., blind deconvolution, time-of-flight data [Heide ... Heidrich '16].
- Where the paradigm is used to generate primitives for graphics use. e.g., "kelvinlets" [de Goes & James, '17].

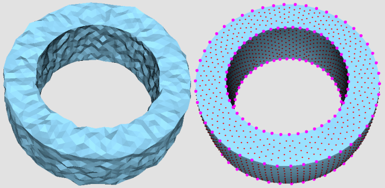
Paradigm:

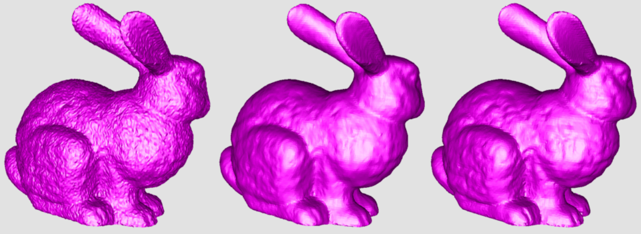
$$\min_u \quad \frac{1}{2}\|f(u) - b\|^2 + \beta R(u),$$

$$R(u) = \int_\Omega \rho(|\nabla u|), \quad \beta > 0,$$

Where other approaches are better: triangle mesh denoising

[Huang & A. '08]





Surface mesh denoising vs image denoising

- Relevant literature sample in image processing
  - Anisotropic diffusion [Perona & Malik '90; Catte et al. '92; Black et al. '98; Weickert '98]
  - Bilateral filtering [Tomasi & Manduchi '98; Sapiro '01; Barash '02]
  - *Multi-scale iterative refinement* [Tadmor et al. '04; Osher et al. '05]
- Differences from Image Processing
  - No separation between mesh locations and intensity heights; vertex drift
  - Mesh sampling irregularity
  - Volume shrinkage
  - *Our proposed algorithms do not have direct parallels in image processing*

Surface mesh denoising vs image denoising

- Relevant literature sample in image processing
  - Anisotropic diffusion [Perona & Malik '90; Catte et al. '92; Black et al. '98; Weickert '98]
  - Bilateral filtering [Tomasi & Manduchi '98; Sapiro '01; Barash '02]
  - *Multi-scale iterative refinement* [Tadmor et al. '04; Osher et al. '05]
- Differences from Image Processing
  - No separation between mesh locations and intensity heights; vertex drift
  - Mesh sampling irregularity
  - Volume shrinkage
  - *Our proposed algorithms do not have direct parallels in image processing*

Discrete model

- Discrete triangle mesh: vertex set V, edge set E. For each $\mathbf{q}_i \in V$ define the one-ring neighborhood $\mathcal{N}(i) \equiv \{j \mid \mathbf{e}_{i,j} = \mathbf{q}_j - \mathbf{q}_i \in E\}$.
- Vertex normal $\mathbf{n}_i$: the average of neighbouring face normals.
- A denoising iteration is derived as updating each vertex $\mathbf{q}_i$ by

$$\mathbf{q}_i \longleftarrow \mathbf{q}_i + \tau \Delta \mathbf{q}_i + \lambda_i(\hat{\mathbf{q}}_i - \mathbf{q}_i),$$

  where $\{\hat{\mathbf{q}}_i;\ i = 1, \ldots, N\}$ is the given noisy data.
- Choose

$$\begin{aligned} \Delta \mathbf{q}_i &= \sum_{j \in \mathcal{N}(i)} W_{i,j} \mathbf{e}_{i,j} \\ W_{i,j} &= w_{i,j} \mathbf{n}_i \mathbf{n}_i^T. \end{aligned}$$

  This way, all sum contributions are proportional to the normal $\mathbf{n}_i$

Discrete anisotropic Laplacian

Compute $\mathbf{h}_i = \{h_{i,j} = \mathbf{e}_{i,j}^T \mathbf{n}_i \mid j \in \mathcal{N}(i)\}$ and define AL operator

$$\Delta \mathbf{q}_i = \frac{1}{C_i} \left(\sum_{j \in \mathcal{N}(i)} g(h_{i,j}) h_{i,j} \right) \mathbf{n}_i$$

- Edge stopping function: $g(h_{i,j}) = \exp(-\frac{h_{i,j}^2}{2\sigma_i^2})$;
- Robust local scaling factor: $\sigma_i = 2 \cdot \mathtt{mean}(\mathtt{abs}(\mathbf{h}_i - \mathtt{mean}(\mathbf{h}_i)))$;
- Normalization factor $C_i = \sum_{j \in \mathcal{N}(i)} g(h_{i,j})$ yields the step size $\tau = 1$.
- Set $\lambda_i = 0, \ \forall i$. Method can be seen as a simplification of bilateral filtering with Gaussian splitting.
- Very fast ($\mathcal{O}(N)$ ops) and effective so long as there are no excessive texture or sharp edges.

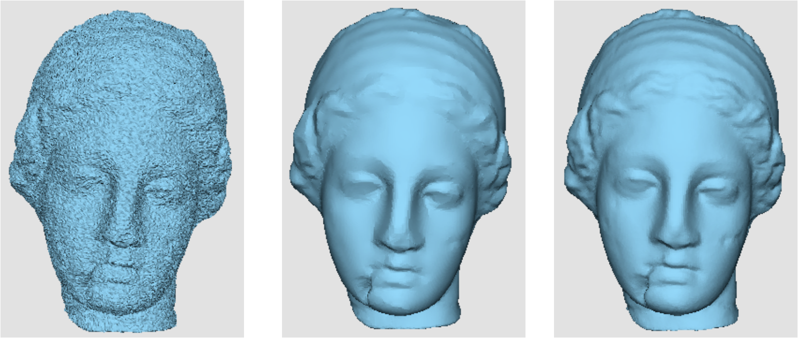
MSAL: handling intrinsic texture

Recapture true higher frequency data components by gradually increasing data fidelity:
for $k = 0, 1, 2 \dots$

$$\mathbf{q}_i \longleftarrow \mathbf{q}_i + \xi^k \Delta \mathbf{q}_i + \lambda_i(\hat{\mathbf{q}}_i - \mathbf{q}_i), \quad i = 1, 2 \dots N$$

- $\tau = \xi^k, 0 < \xi < 1 \Leftarrow$ reduce effect of smoothing gradually.
- $0 < \lambda_i \leq 1 \Leftarrow$ accumulate smoothing contributions monotonically;
- $\lambda_i = \sigma_i / \bar{\sigma}, \quad \bar{\sigma} = \max\{\sigma_j;\ j = 1, \dots, N\} \Leftarrow$ want more data fidelity where there is more fine scale action;
- As $k \to \infty$ the process converges at steady state to the given data $\hat{\mathbf{q}}$.

Comparing with anisotropic diffusion



(a) (b) (c)

Figure: (a) Fine Igea model (135K verts) with intrinsic texture corrupted by noise; (b) smoothed by anisotropic diffusion [Hildebrandt & Polthier, 2004], 25 itns, 33 secs; (c) smoothed by MSAL, $\xi = 1/2$, 4 itns, 13 secs.

Where other approaches are better: salient features and tele-registration

[Huang, Yin, Gong, Lischinski, Cohen-Or, A., Chen '13]

Example: mending a dish

Essential algorithm steps

1. Detect salient curves inside each image piece
2. Attempt to find for each curve a matching curve from an adjacent piece, across the gap
3. Use this to construct an *ambient vector field* surrounding all the pieces
4. Transform (translation, rotation and scaling) each piece so salient curves line up
5. Construct smooth *bridging curves* that connect such pairs across gaps
6. Fill the gaps using structure-driven synthesis, while any remaining inside/outside holes are completed using standard inpainting tools [Barnes et al. '09, Darabi et al. '12].

Where other approaches are better: salient features and tele-registration

Example: removing obstruction

Where other approaches are better: salient features and tele-registration

Example: visual archaeology (Banteay Chhmar, Cambodia)

Conclusions

- Incorporating more mathematically sound techniques into methods and algorithms for computer graphics and image processing.
- Significant practical advantages gained in visual computing using physics-based simulation, data-driven model calibration, etc.
- May occasionally be able to use math to obtain solid justification of algorithms: both on when they work and on when they won't.
- Can even bridge the gap between qualitative and quantitative (which is our secret wet dream).

Conclusions cont.

- Do not get swayed by sheer mathematical prowess.
- Watch out for situations where the gap between physics and physics-based is too wide (e.g., in finding fluid viscosity or damping of soft body).
- Insisting on solving differential equations or satisfying mathematical topology theorems might lead to inferior algorithms for visual tasks.

"There is always a well-known solution to every human problem – neat, plausible, and wrong." [H.L. Mencken, The Economist Espresso 12/9/17]