G22.2274-001, Fall 2009

Advanced Computer Graphics

Project details and tools

Project Topics

- Computer Animation

- Geometric Modeling

- Computational Photography
  Image processing

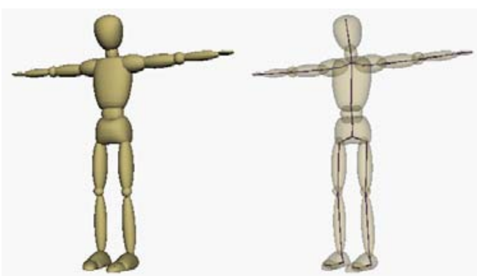
Optimization

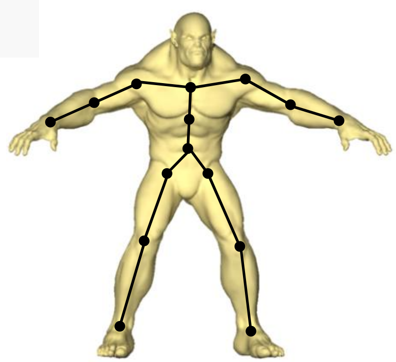
- All projects have a global optimization component:
  - Define some energy functional $\mathrm{E}(\mathbf{x})$
    - $\mathbf{x}$ is the shape, or the image…
  - Compute the optimal x such that $\mathrm{E}(\mathbf{x})$ is minimized
  - There are some constraints to the minimization

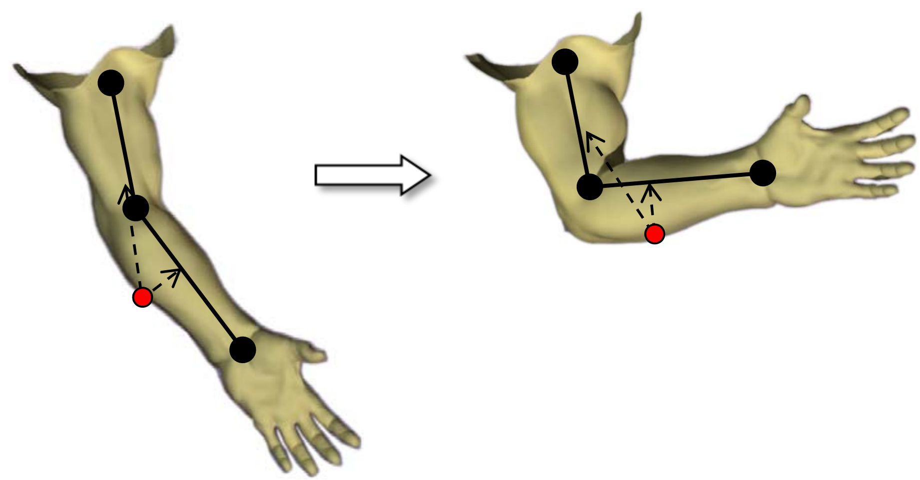
Computer Animation

Character animation

- Create a system for rigging and posing of digital characters (shapes)
  - Step 1: compute a skeleton





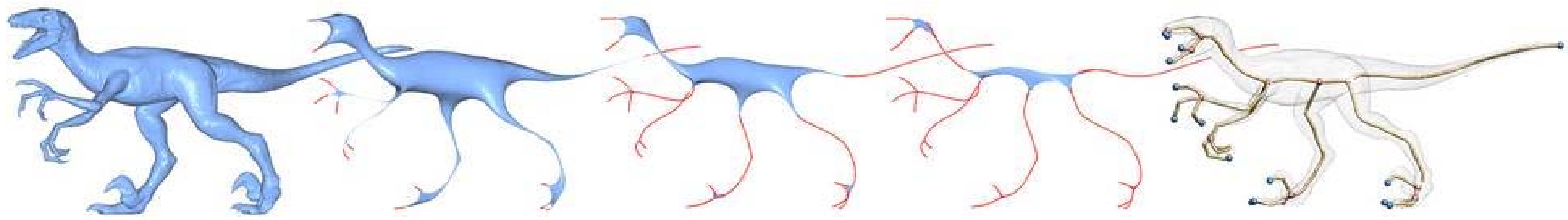


Computer Animation

Character animation

- Create a system for rigging and posing of digital characters (shapes)
  - Step 1: compute a skeleton

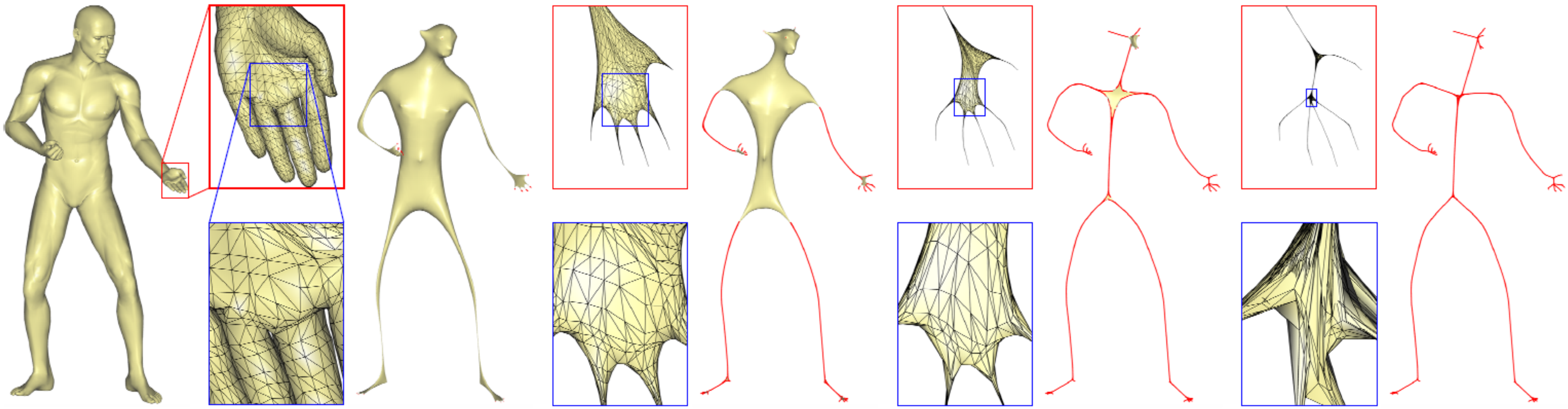
    Suggestion: implement the paper "Skeleton Extraction by Mesh Contraction", SIGGRAPH 2008



  - also implement GUI to manually create a skeleton

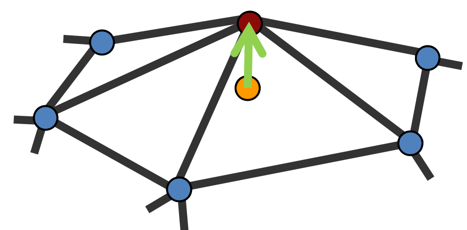
Computer Animation

Character animation



- Function $\mathrm{E}(\mathbf{x})$ measures how smooth the shape $\mathbf{x}$ is

$$\mathrm{E}(\mathbf{x}) = \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j \in N(i)} w_{ij} \mathbf{x}_j \right\|^2$$

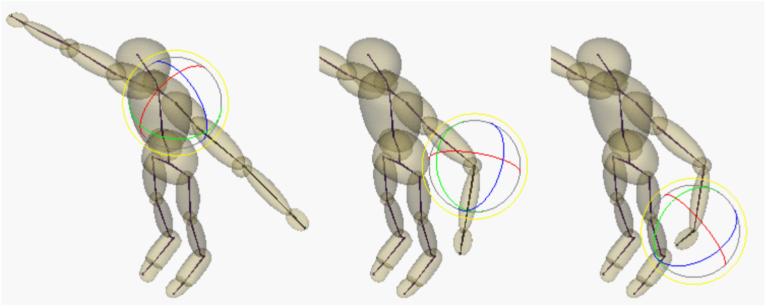


- Constraints: extremities (finger tips, ears, etc.)

Computer Animation

Character animation

- Create a system for rigging and posing of digital characters (shapes)
  - Step 2: GUI for skeleton posing
    - Forward kinematics – just change joint angles



    - Optional: also add inverse kinematics

Computer Animation

Character animation

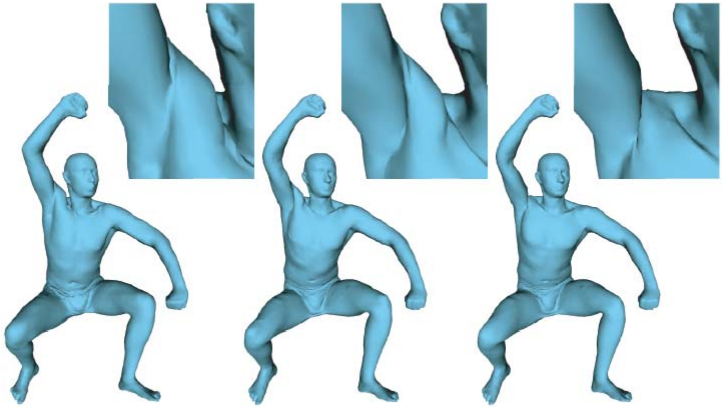
- Create a system for rigging and posing of digital characters (shapes)
  - Step 3: implement skinning algorithm
    - Basic linear blend skinning
    - Dual Quaternions (Kavan et al. 2008)
    - Optional: also implement one of the recent optimization-based skinning techniques, such as "Real-Time Enveloping with Rotational Regression", SIGGRAPH 2007

Computer Animation

Character animation

- Create a system for rigging and posing of digital characters (shapes)
  - Step 3: implement skinning algorithm
    - Basic linear blend skinning
    - Dual Quaternions (Kavan et al. 2008)
    - The functional $\mathrm{E}(\mathbf{x})$ measures how similar the deformed shape x is to the original shape in terms of local details

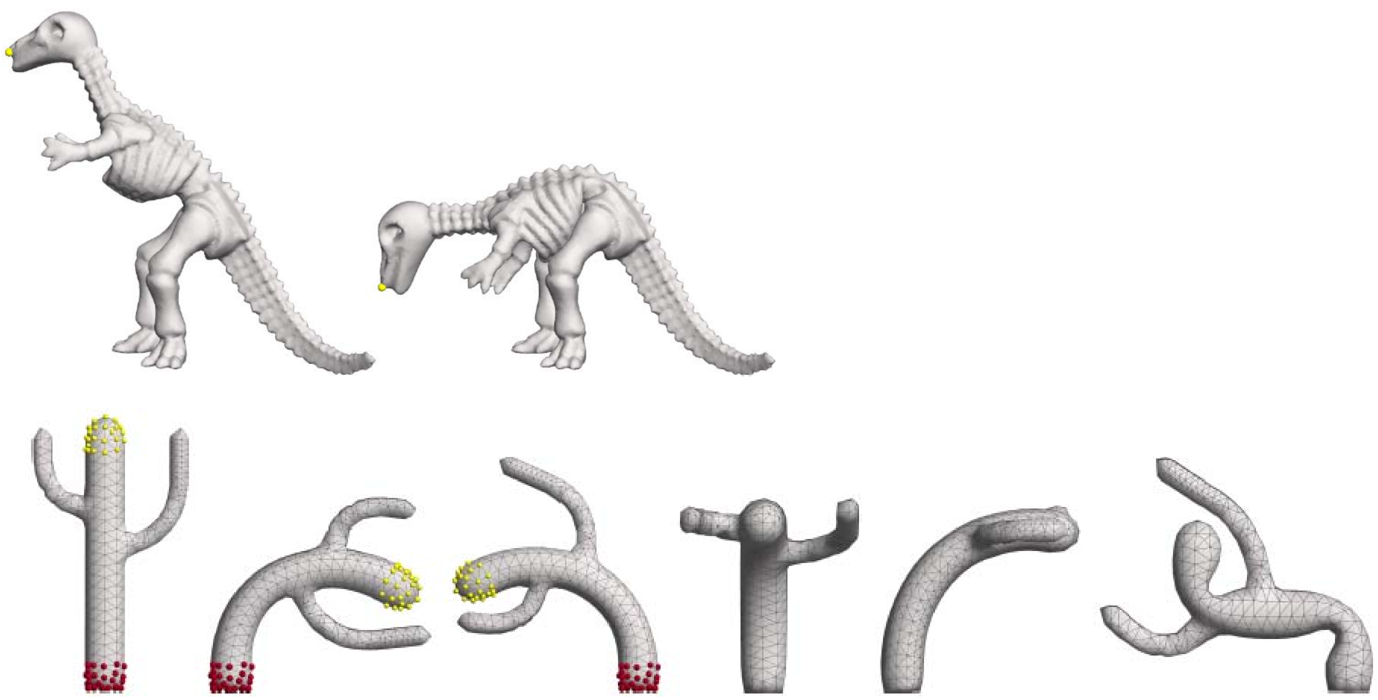
$$\mathrm{E}(\mathbf{x}) = \sum_{i=1}^{n} \left\| \mathrm{L}(\mathbf{x}_i) - \mathrm{T}\left(\mathrm{L}(\mathbf{x}^{\mathrm{orig}}{}_i)\right) \right\|^2$$



Geometric Modeling

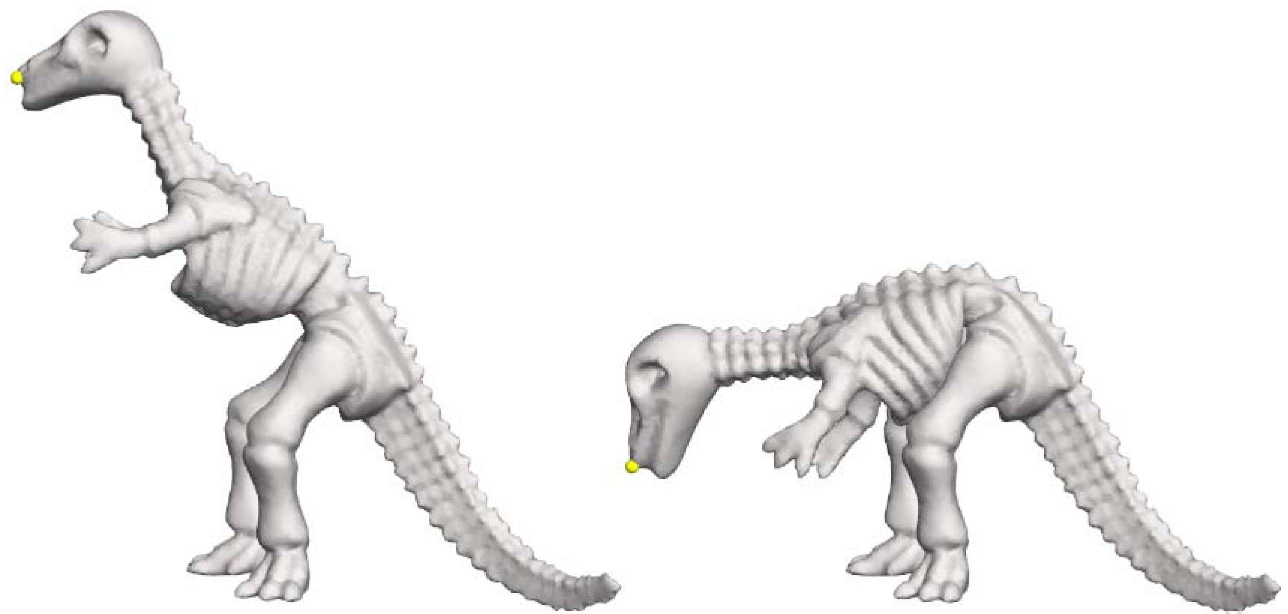
Interactive shape editing system

- A system to edit shapes interactively by “grab-and-drag” interface



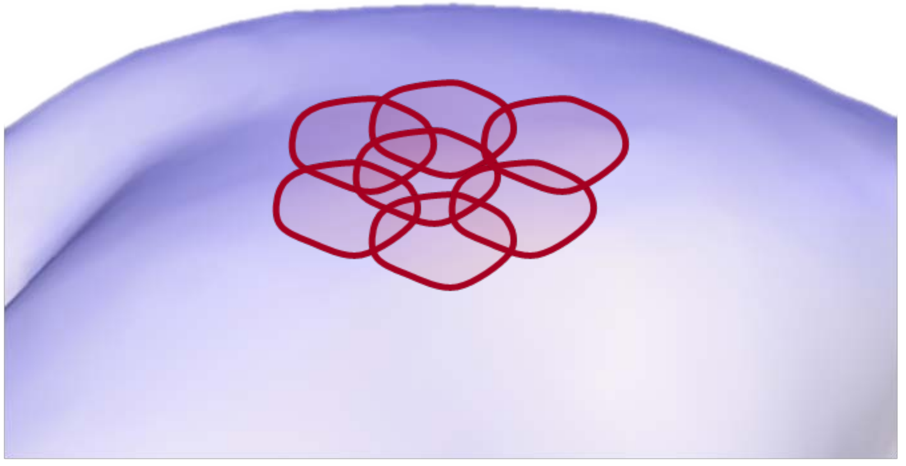
As-rigid-as-possible surface deformation

- Smooth effect on the large scale
- As-rigid-as-possible effect on the small scale (preserves details)



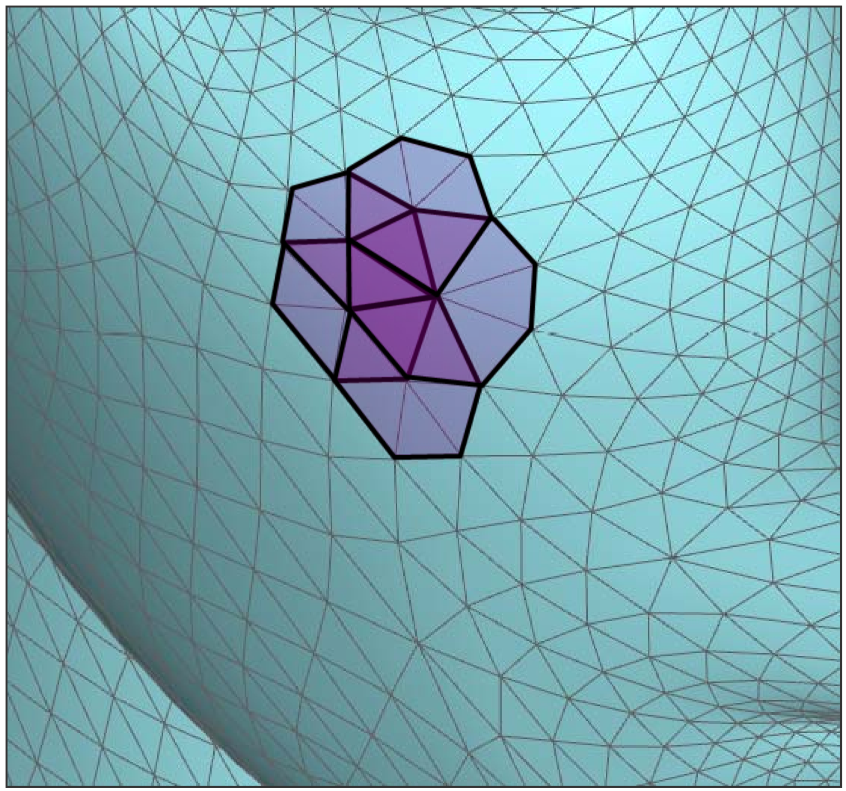
Direct ARAP modeling

- We actually may want to preserve the shapes of cells covering the surface



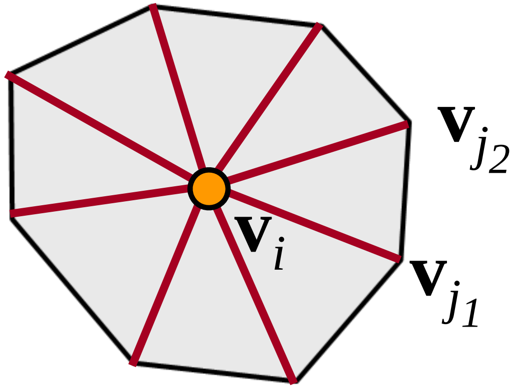
Direct ARAP modeling

- Let's look at cells on a mesh



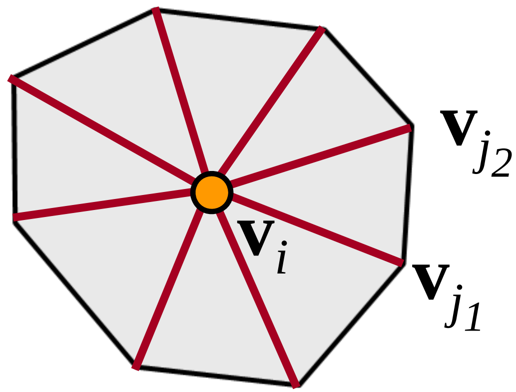
Direct ARAP modeling

- Ask all the star edges to transform rigidly, then the shape of the cell is preserved



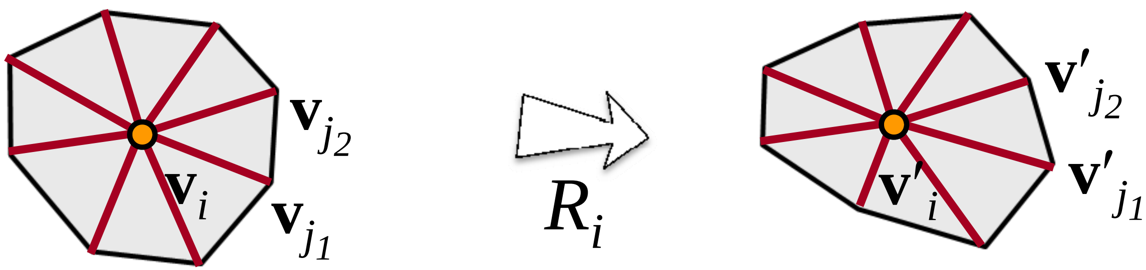
Direct ARAP modeling

- Cell energy: $\min \sum_{j \in N(i)} \left\| (\mathbf{v}'_i - \mathbf{v}'_j) - R_i (\mathbf{v}_i - \mathbf{v}_j) \right\|^2$



Direct ARAP modeling

- If $\mathbf{v}$, $\mathbf{v}'$ are known then R_i is uniquely defined



- It's the shape matching problem!
  - Build covariance matrix $\mathrm{S} = \mathrm{V}\mathrm{V}'^{\mathrm{T}}$
  - SVD: $\mathrm{S} = \mathrm{U}\Sigma\mathrm{P}^{\mathrm{T}}$
  - $R_i = \mathrm{U}\mathrm{P}^{\mathrm{T}}$

R_i is a non-linear function of $\mathbf{v}'$

Direct ARAP modeling

- Can formulate overall energy of the deformation:

$$\min_{\mathbf{v}'} \sum_{i=1}^{n} \sum_{j \in N(i)} \left\| (\mathbf{v}'_i - \mathbf{v}'_j) - R_i (\mathbf{v}_i - \mathbf{v}_j) \right\|^2$$

$$s.t.\ \mathbf{v}'_j = \mathbf{c}_j,\ j \in C$$

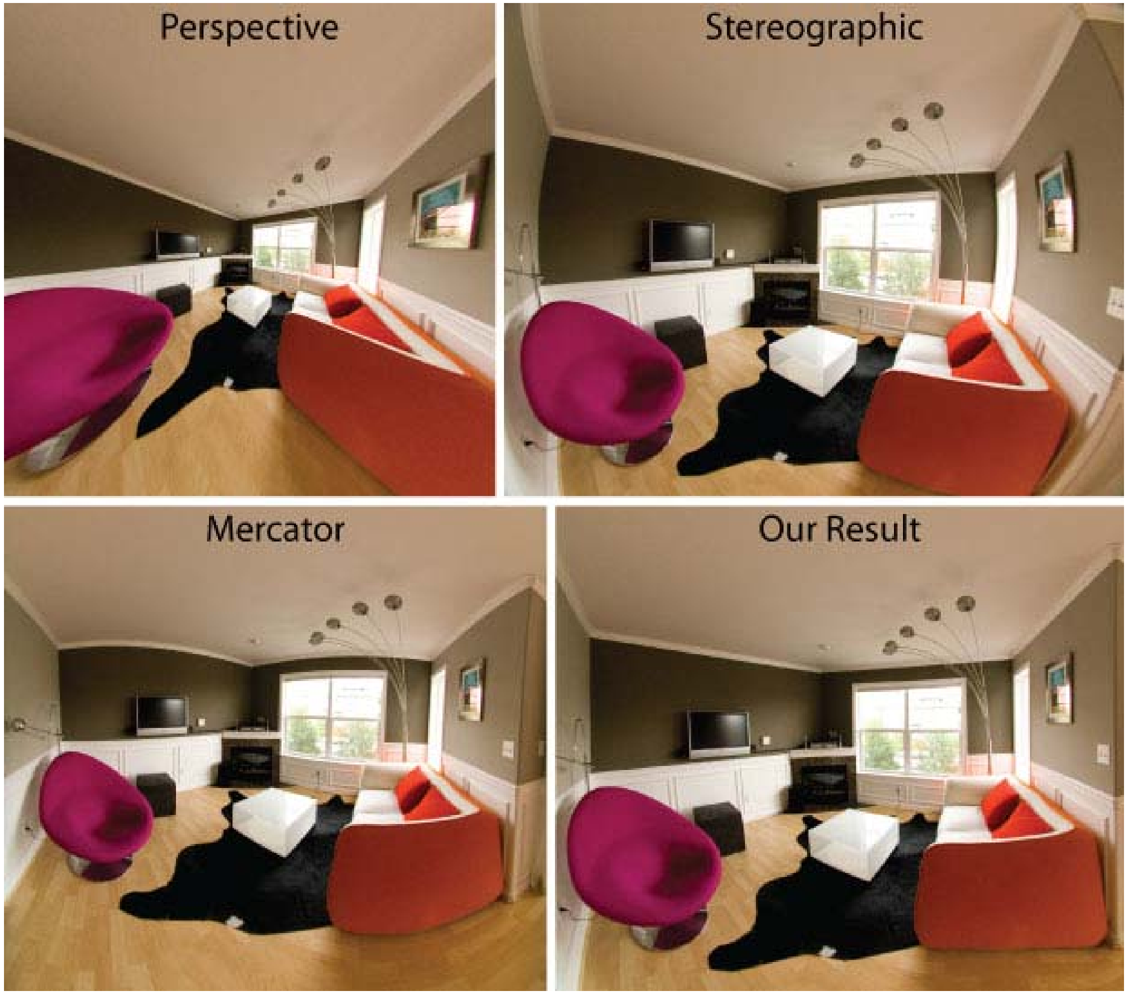
Geometric Modeling

Interactive shape editing system

- Implement one of the recent papers:
  - “PriMo”, SGP 2006
  - “As-rigid-as-possible surface modeling”, SGP 2007
    - Add multiresolution hierarchy
    - Try both the optimization method in the paper and direct Gauss-Newton optimization, helped by this paper: “Shape Decomposition Using Modal Analysis”, Eurographics 2009

Computational Photography

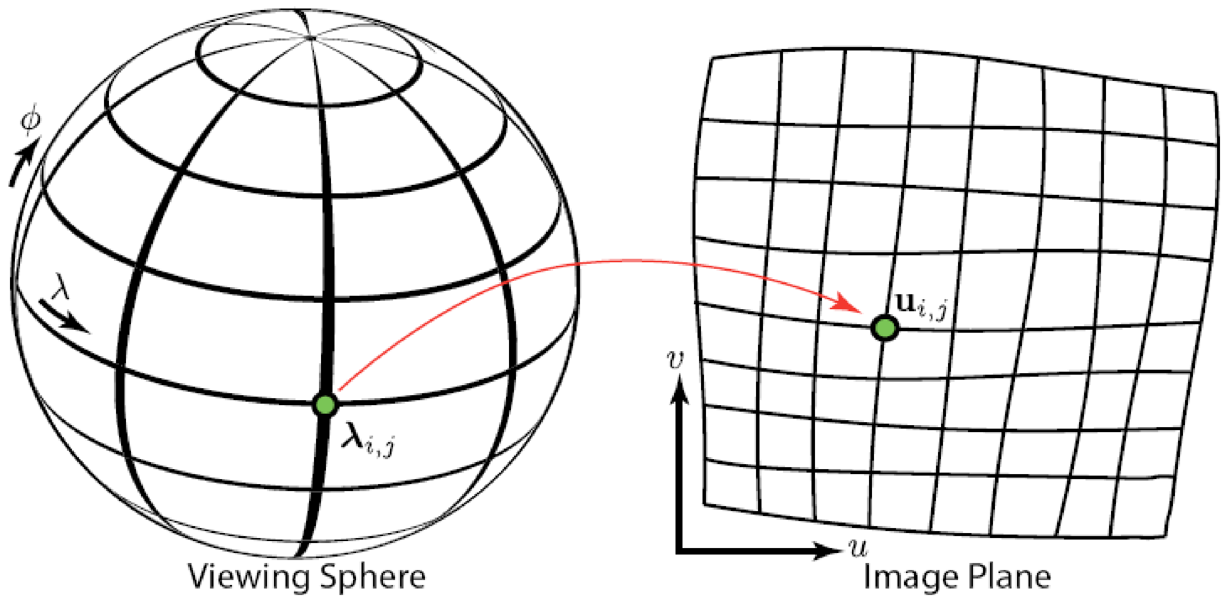
Rectifying fish-eye lens distortion



Computational Photography

Rectifying fish-eye lens distortion

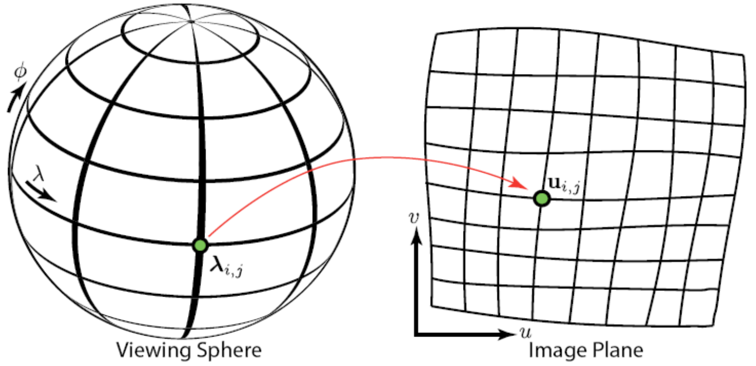
- Implement the paper "Optimizing Content-Preserving Projections for Wide-Angle Images", SIGGRAPH 2009
- They **optimize** the mapping from the viewing sphere onto the image plane



Computational Photography

Rectifying fish-eye lens distortion

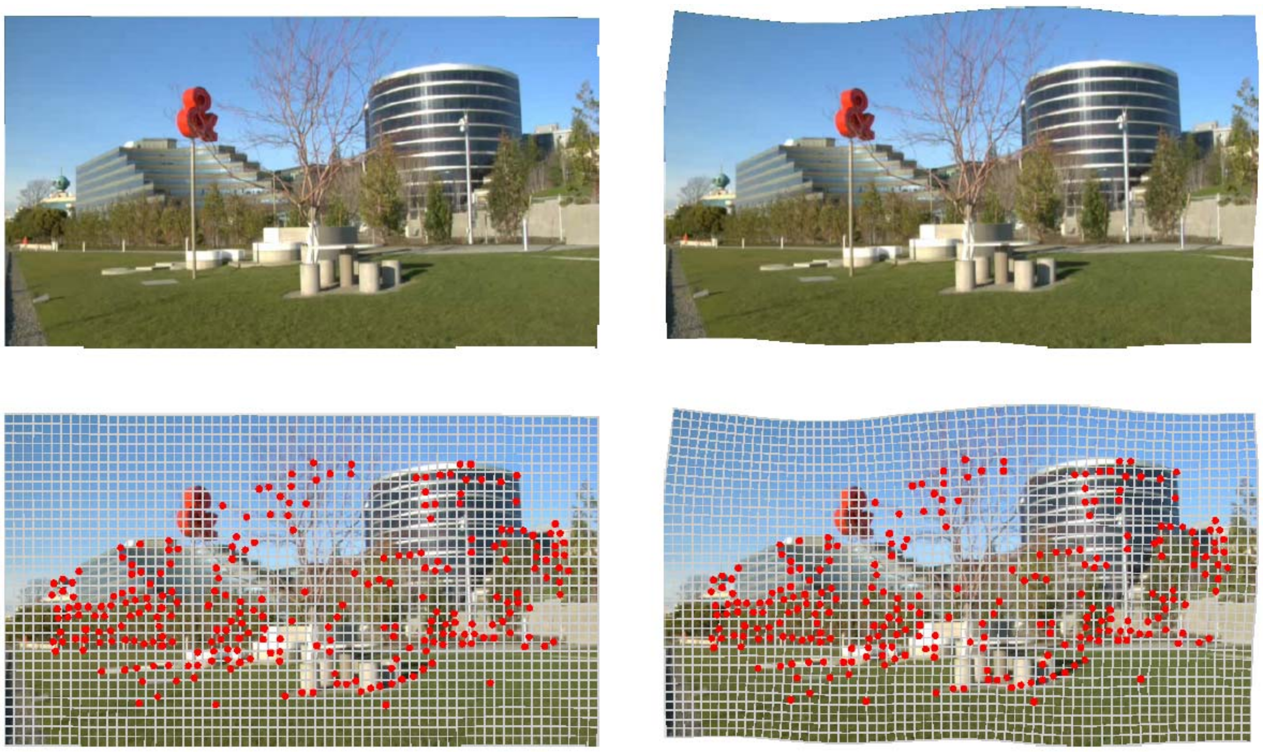
- Implement the paper "Optimizing Content-Preserving Projections for Wide-Angle Images", SIGGRAPH 2009
- They **optimize** the mapping from the viewing sphere onto the image plane
- $\mathrm{E}(\mathbf{x})$ measures:
  - How well the quads are preserved (want all angles to be 90)
  - How well straight lines that the user marked are preserved



Computational Photography

Shaky cam stabilization

- Implement "Content-Preserving Warps for 3D Video Stabilization", SIGGRAPH 2009



Computational Photography

Shaky cam stabilization

- Implement "Content-Preserving Warps for 3D Video Stabilization", SIGGRAPH 2009
- $\mathrm{E}(\mathbf{x})$ measures:
  - How well the quads are preserved (want all angles to be 90)
- Constraints: point-to-point

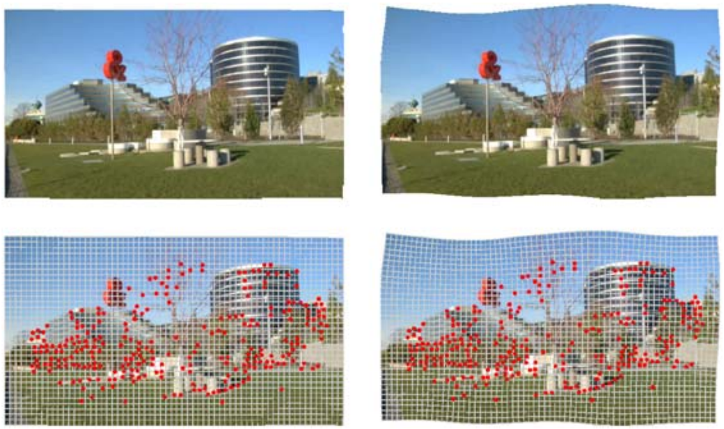


Image Processing

Image retargeting (resizing)

- The problem: resize an image to fit a display device with a different aspect ratio

Image Processing

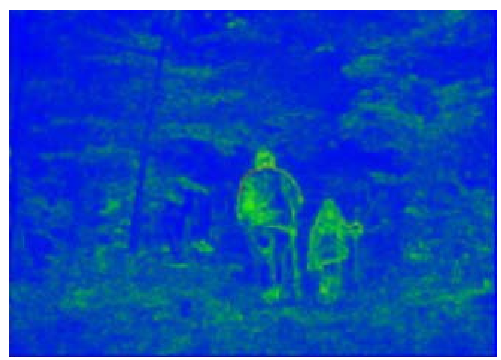
Image retargeting (resizing)

- The problem: resize an image to fit a display device with a different aspect ratio

Image Processing

Image retargeting (resizing)

- Approach:
  - Compute an importance map of the image
  - **Warp** the image such that regions with high importance are preserved at the expense of unimportant regions

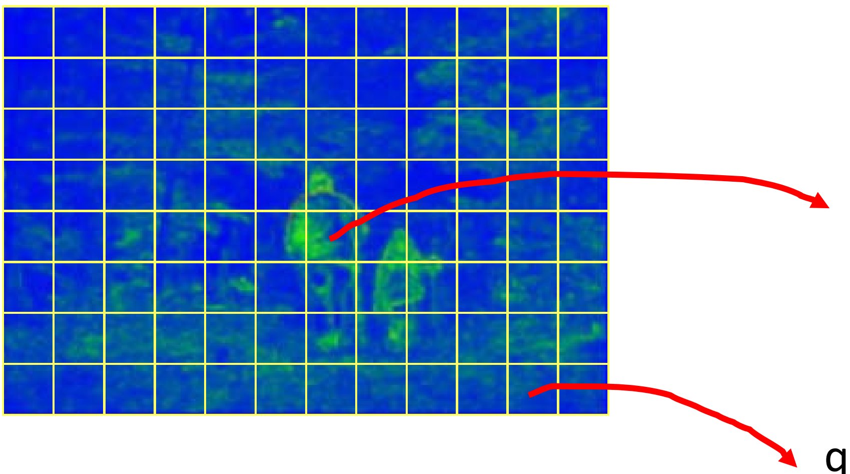


importance map

Image Processing

Image retargeting (resizing)

- Grid mesh, preserve the shape of the important quads



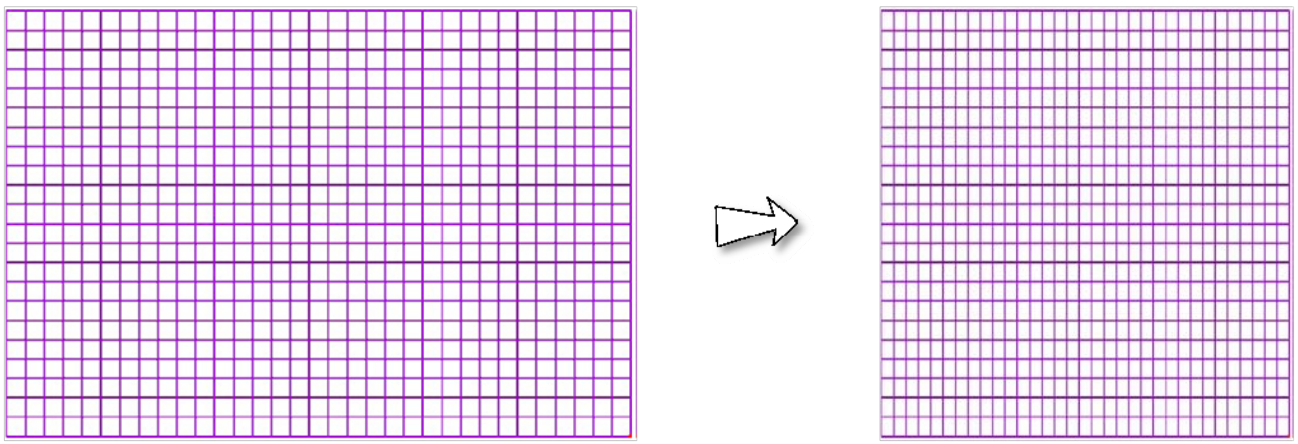
quads with high importance:
uniform scaling

quads with low importance:
allowed non-uniform scaling

- Optimize the location of mesh vertices, interpolate image

Image Processing
Image retargeting (resizing)

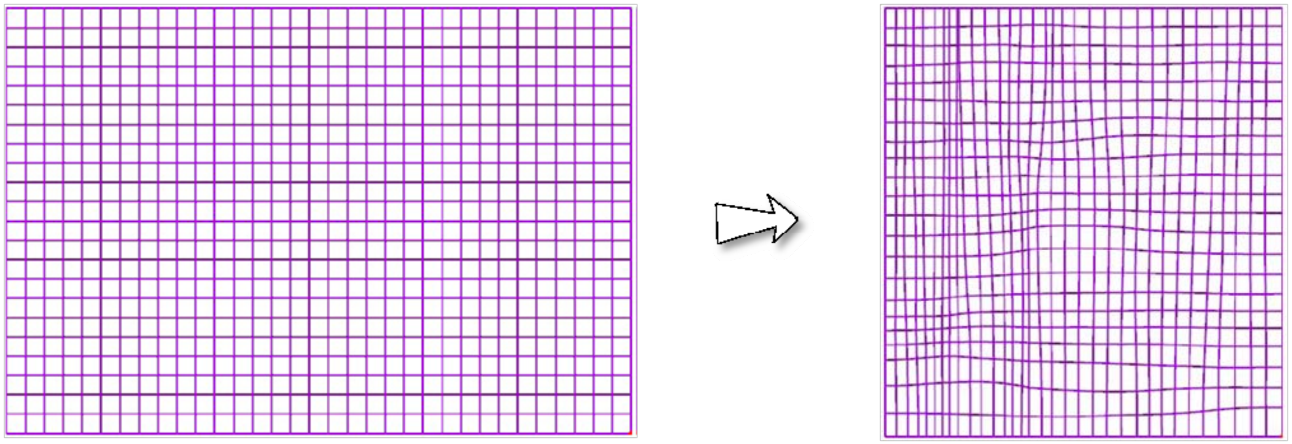
- Grid mesh, preserve the shape of the important quads



- Optimize the location of mesh vertices, interpolate image

Image Processing
Image retargeting (resizing)

- Grid mesh, preserve the shape of the important quads

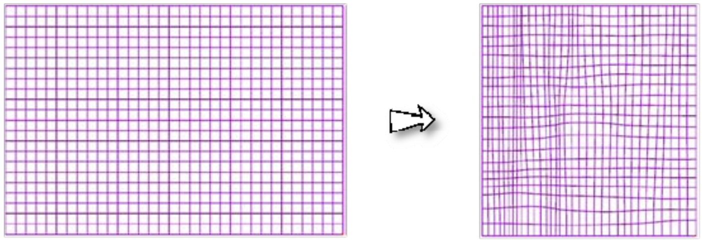


- Optimize the location of mesh vertices, interpolate image

Image Processing

Image retargeting (resizing)

- $E(\mathbf{x})$ measures:
  - How well the quad shape is preserved
  - How smooth the grid lines are
  - In this project: how well straight lines are preserved
- Constraints:
  - The boundary of the image (has to fit new dimensions)



Optimization

In the projects:
mostly linear optimization, i.e.
linear least-squares problems

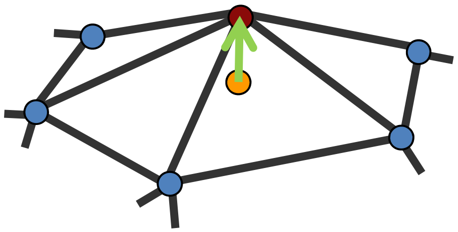
Solving linear systems in LS sense

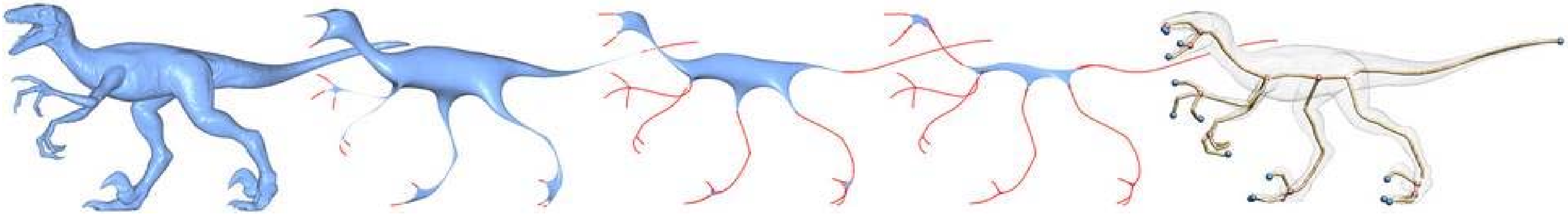
- Wish list of equations:

$$\forall i = 1, \ldots, n: \quad f(\mathbf{x}_i) = f_i$$

- Example: measuring surface smoothness

$$\mathrm{E}(\mathbf{x}) = \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j \in N(i)} w_{ij} \mathbf{x}_j \right\|^2$$





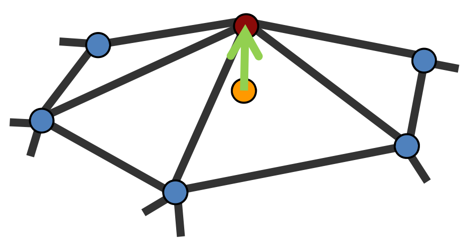
Solving linear systems in LS sense

- Wish list of equations:

$$\forall i = 1, \ldots, n: \quad f(\mathbf{x}_i) = f_i$$

- Example: measuring surface smoothness

$$\mathrm{E}(\mathbf{x}) = \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j \in N(i)} w_{ij} \mathbf{x}_j \right\|^2$$



$$\forall i = 1, \ldots, n: \quad \mathbf{x}_i - \sum_{j \in N(i)} w_{ij} \mathbf{x}_j = 0$$

⇐ Wish equation

Solving linear systems in LS sense

- We have an over-determined linear system $\mathrm{k} \times \mathrm{n}$:

$$a_{11}\,\mathbf{x}_1 + a_{12}\,\mathbf{x}_2 + \ldots + a_{1n}\,\mathbf{x}_n = b_1$$

$$a_{21}\,\mathbf{x}_1 + a_{22}\,\mathbf{x}_2 + \ldots + a_{2n}\,\mathbf{x}_n = b_2$$

$$\ldots$$

$$a_{\mathrm{k}1}\,\mathbf{x}_1 + a_{\mathrm{k}2}\,\mathbf{x}_2 + \ldots + a_{\mathrm{k}n}\,\mathbf{x}_n = b_{\mathrm{k}}$$

Solving linear systems in LS sense

- In matrix form:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & & \dots & \\ \vdots & \vdots & & \vdots \\ \\ a_{k1} & a_{k2} & \dots & a_{kn} \end{pmatrix} \begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \vdots \\ \mathrm{x}_n \end{pmatrix} = \begin{pmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \\ \vdots \\ \\ \mathrm{b}_k \end{pmatrix}$$

Solving linear systems in LS sense

- In matrix form:

$$\mathrm{A}\mathbf{x} = \mathbf{b}$$

where $\mathrm{A} = (a_{ij})$ is a rectangular $k \times n$ matrix, $k > n$

$\mathbf{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ $\qquad$ $\mathbf{b} = (b_1, b_2, \ldots, b_k)^{\mathrm{T}}$

Solving linear systems in LS sense

- More constrains than variables – no exact solutions generally exist
- We want to find something that is an "approximate solution":

$$\mathbf{x}_{opt} = \underset{\mathbf{x}}{\operatorname{argmin}} \|A\mathbf{x} - \mathbf{b}\|^2$$

$$\min_{\mathbf{x}} E(\mathbf{x})$$

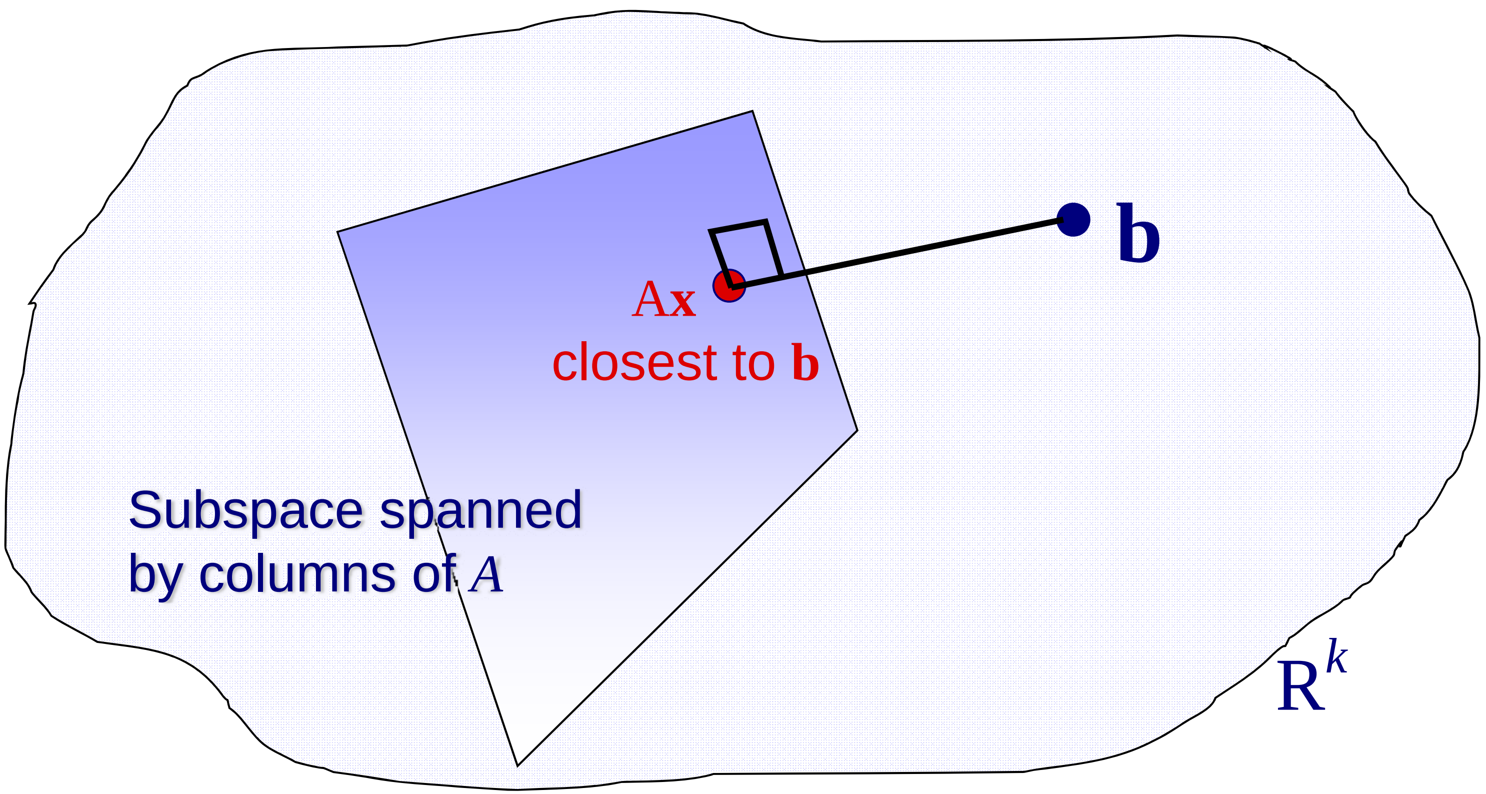
Finding the LS solution

- $\mathbf{x} \in \mathrm{R}^n$
- $\mathrm{A}\mathbf{x} \in \mathrm{R}^k$
- As we vary $\mathbf{x}$, $\mathrm{A}\mathbf{x}$ varies over the linear subspace of R^k spanned by the columns of A:

$$\mathrm{A}\mathbf{x} = \begin{pmatrix} A_1 & A_2 & & A_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{pmatrix} = x_1 A_1 + x_2 A_2 + \ldots + x_n A_n$$

Finding the LS solution

- We want to find the closest $A\mathbf{x}$ to $\mathbf{b}$: $\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|^2$



Finding the LS solution

- The vector $\mathrm{A}\mathbf{x}$ closest to $\mathbf{b}$ satisfies:

$$(\mathrm{A}\mathbf{x} - \mathbf{b}) \perp \{\text{subspace of A's columns}\}$$

$$\Updownarrow$$

$$\forall \text{ column } A_i: \langle A_i, \mathrm{A}\mathbf{x} - \mathbf{b}\rangle = 0$$

$$\forall\, i, \quad A_i^{\mathrm{T}}(\mathrm{A}\mathbf{x} - \mathbf{b}) = 0$$

$$\Updownarrow$$

$$\mathrm{A}^{\mathrm{T}}(\mathrm{A}\mathbf{x} - \mathbf{b}) = 0$$

$$(\mathrm{A}^{\mathrm{T}}\mathrm{A})\mathbf{x} = \mathrm{A}^{\mathrm{T}}\mathbf{b}$$

These are called **the normal equations**

Finding the LS solution

- We got a square symmetric system $\underset{(n\times n)}{(\mathrm{A}^{\mathrm{T}}\mathrm{A})}\mathbf{x} = \mathrm{A}^{\mathrm{T}}\mathbf{b}$
- If A has full rank (the columns of A are linearly independent) then $(\mathrm{A}^{\mathrm{T}}\mathrm{A})$ is invertible.

$$\min_{\mathrm{x}} \left\| \mathrm{A}\mathbf{x} - \mathbf{b} \right\|^2$$

$$\Downarrow$$

$$\mathbf{x} = \left(\mathrm{A}^{\mathrm{T}}\mathrm{A}\right)^{-1} \mathrm{A}^{\mathrm{T}}\,\mathbf{b}$$

Weighted least squares

- If each constraint has a weight in the energy:

$$\min_{\mathbf{x}} \sum_{i=1}^{n} w_i \left(\mathrm{f}_i(\mathbf{x}_i) - \mathbf{b}_i\right)^2$$

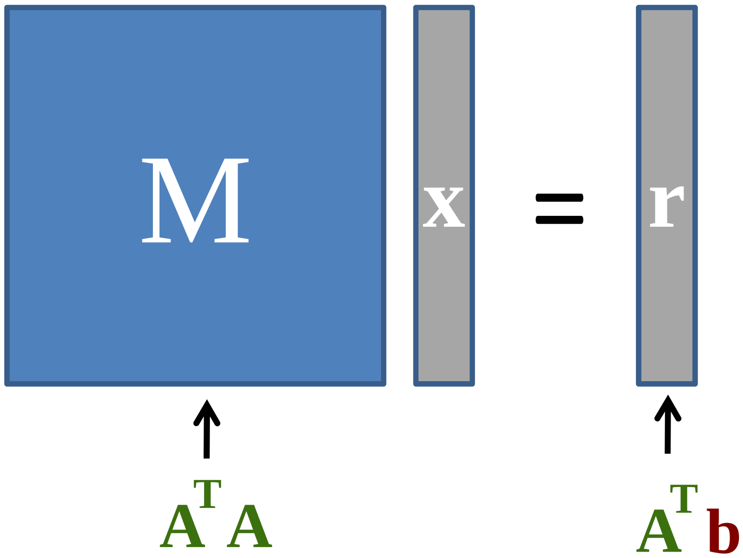
- The weights $w_i > 0$ and don't depend on $\mathbf{x}$
- Then:

$$\min\ (\mathrm{A}\mathbf{x} - \mathbf{b})^{\mathrm{T}} \mathrm{W} (\mathrm{A}\mathbf{x} - \mathbf{b}) \quad \text{where } \mathrm{W} = (w_i)_{ii}$$

$$(\mathrm{A}^{\mathrm{T}} \mathrm{W}\, \mathrm{A})\, \mathbf{x} = \mathrm{A}^{\mathrm{T}} \mathrm{W}\, \mathbf{b}$$

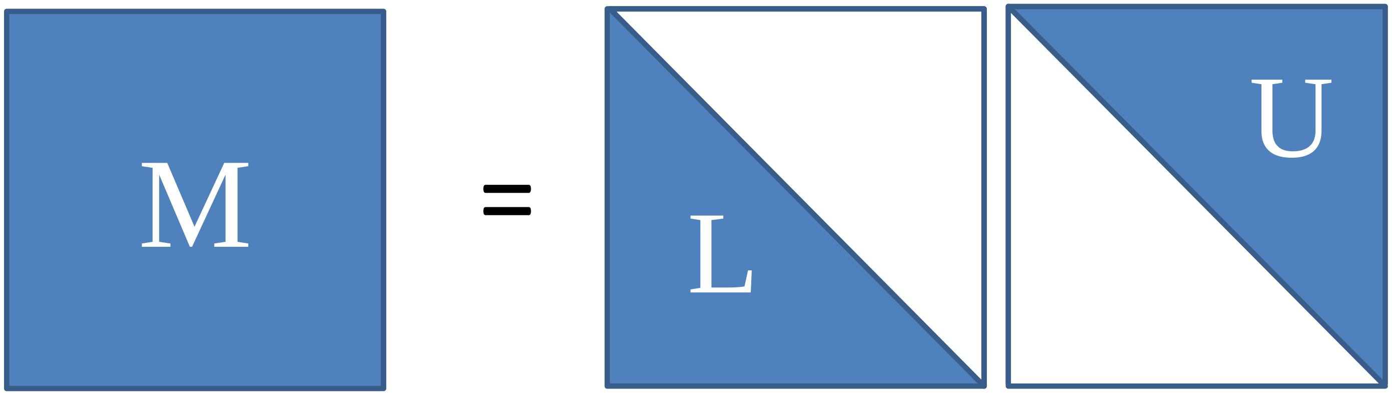
Linear Systems

- Matrix is often fixed, rhs changes



Matrix Factorization

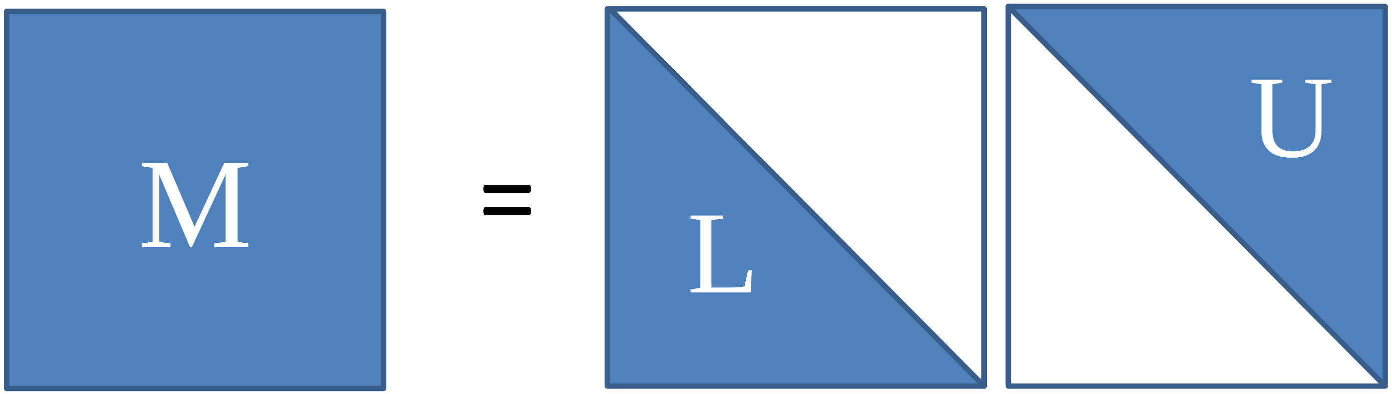
LU decomposition



$$M\mathbf{x} = \mathbf{r}$$
$$LU\mathbf{x} = \mathbf{r}$$

Matrix Factorization

LU decomposition

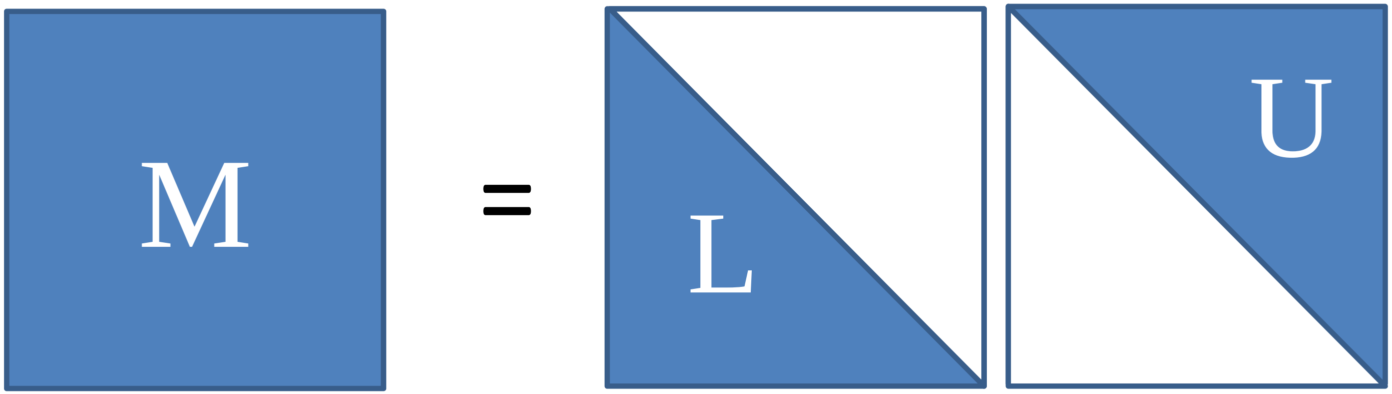


$$M\mathbf{x} = \mathbf{r}$$

$$L(U\mathbf{x}) = \mathbf{r}$$

Matrix Factorization

LU decomposition



$$\mathrm{M}\mathbf{x} = \mathbf{r}$$
$$\mathrm{L}(\mathrm{U}\mathbf{x}) = \mathbf{r}$$

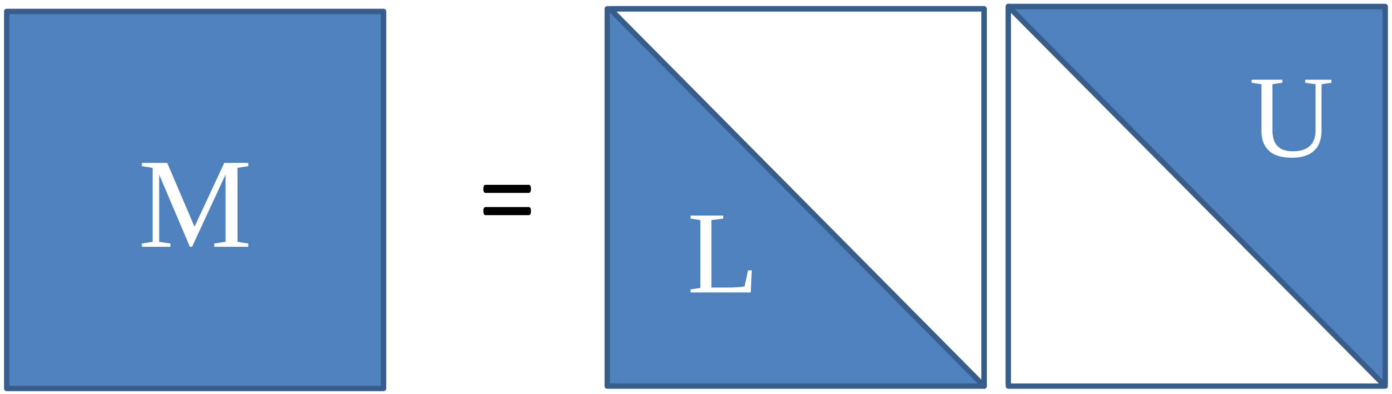
$\Rightarrow$

$$\mathrm{L}\mathbf{y} = \mathbf{r}$$
$$\mathrm{U}\mathbf{x} = \mathbf{y}$$

This is backsubstitution. If L, U are sparse it is very fast. The hard work is computing L and U

Matrix Factorization

LU decomposition



$$\mathrm{M}\mathbf{x} = \mathbf{r}$$
$$\mathrm{L}(\mathrm{U}\mathbf{x}) = \mathbf{r}$$

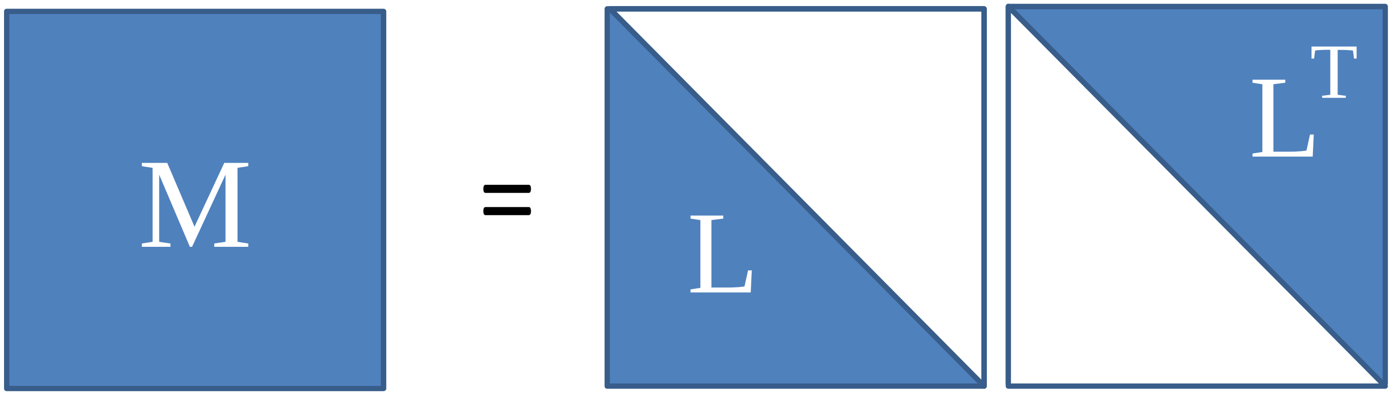
$\Rightarrow$

$$\mathbf{y} = \mathrm{L}^{-1}\mathbf{r}$$
$$\mathbf{x} = \mathrm{U}^{-1}\mathbf{y}$$

This is backsubstitution. If L, U are sparse it is very fast. The hard work is computing L and U

Matrix Factorization

Cholesky decomposition



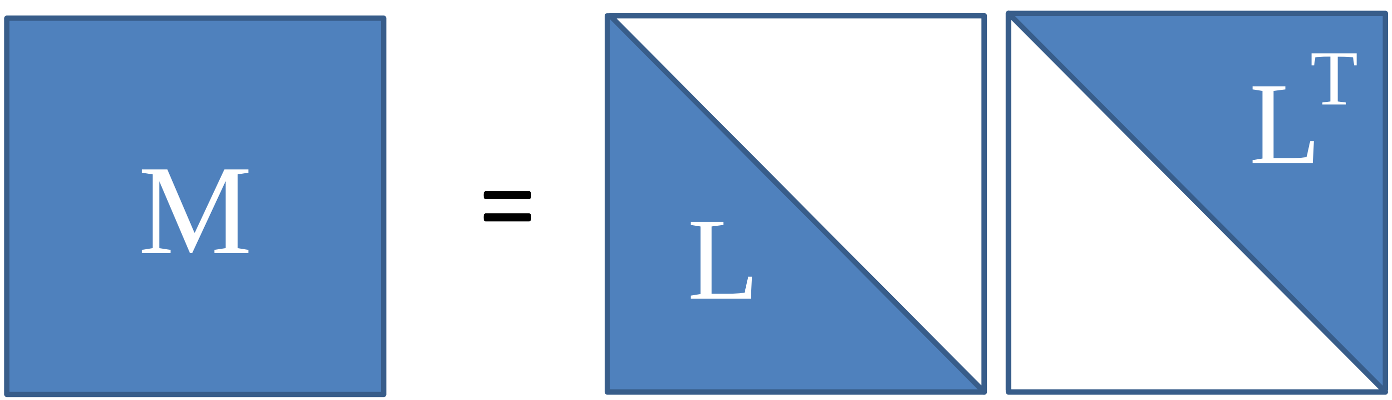
Cholesky factor exists if M is positive definite. It is even better than LU because we save memory.

Cholesky Decomposition

$$\mathrm{M} = \mathrm{L}\mathrm{L}^{\mathrm{T}}$$

- M is symmetric positive definite (SPD):

  $\forall\ \mathbf{x} \neq 0,\ \langle \mathrm{M}\mathbf{x}, \mathbf{x} \rangle > 0 \quad \Leftrightarrow \quad$ all M's eigenvalues > 0

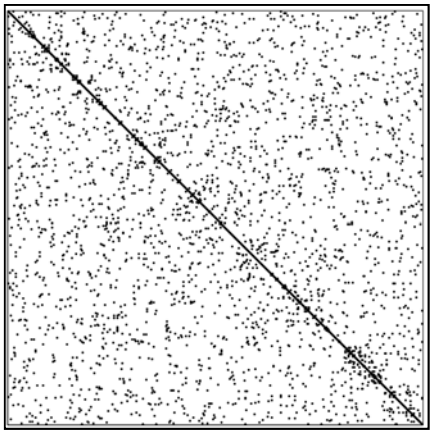


Dense Cholesky Factorization

$M = LL^T$

500×500 matrix

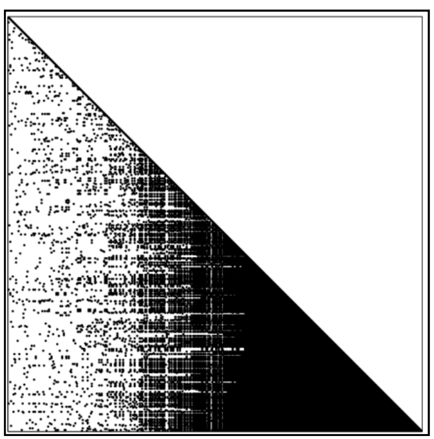
3500 nonzeros



Cholesky Factorization

L

36k nonzeros

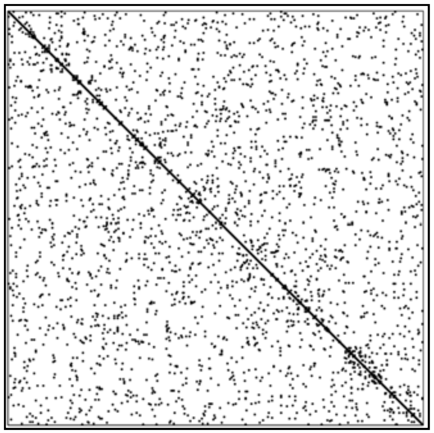


Sparse Cholesky Factorization

$M = LL^T$

500×500 matrix

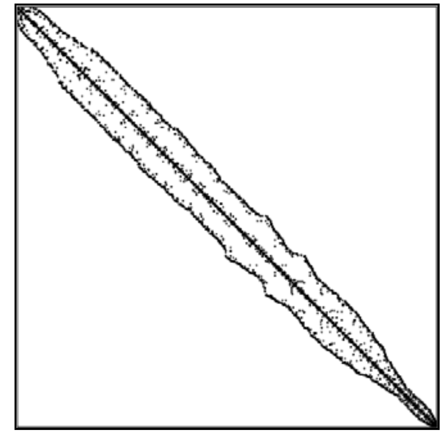
3500 nonzeros



Reordering

PMP^T

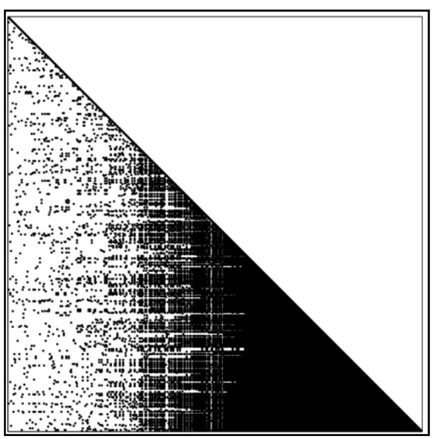
reverse Cuthill-McKee algorithm



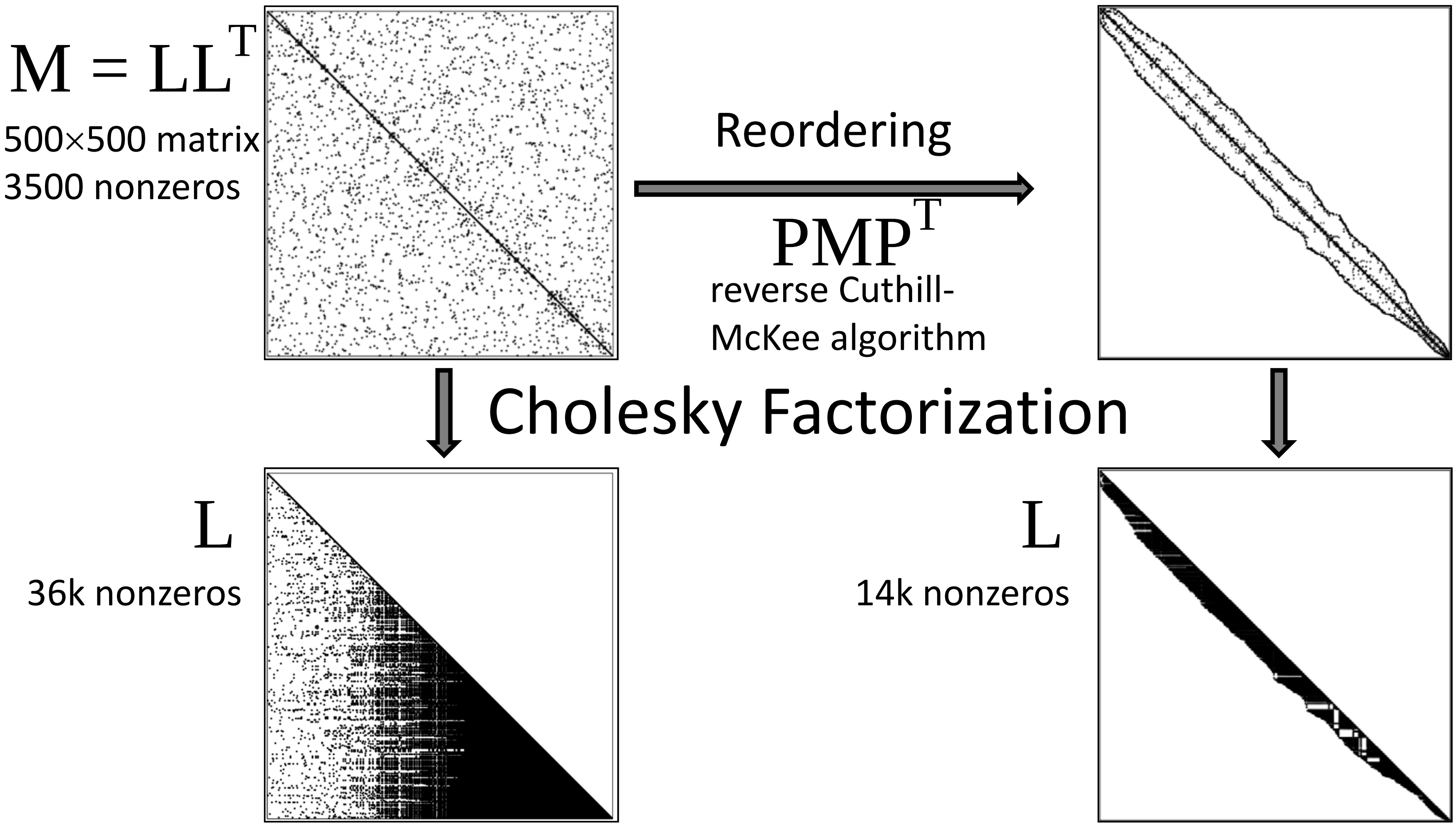
Cholesky Factorization

L

36k nonzeros



Sparse Cholesky Factorization

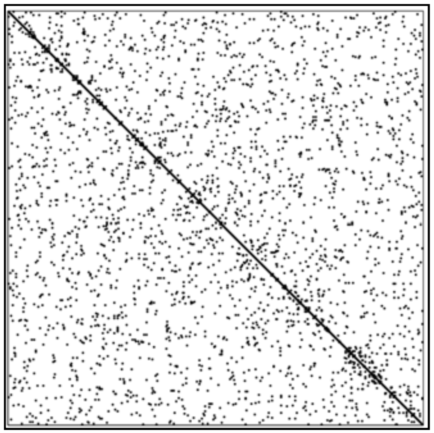


Sparse Cholesky Factorization

$M = LL^T$

500×500 matrix

3500 nonzeros

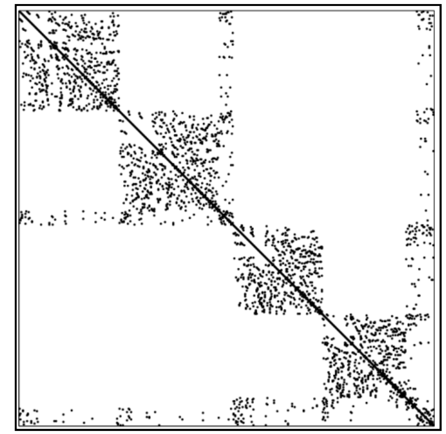


Reordering

PMP^T

nested dissection

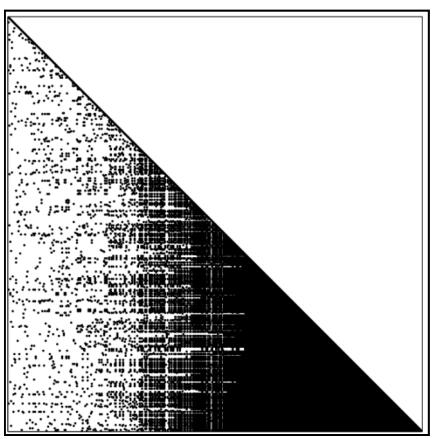
(parallelizable)



Cholesky Factorization

L

36k nonzeros

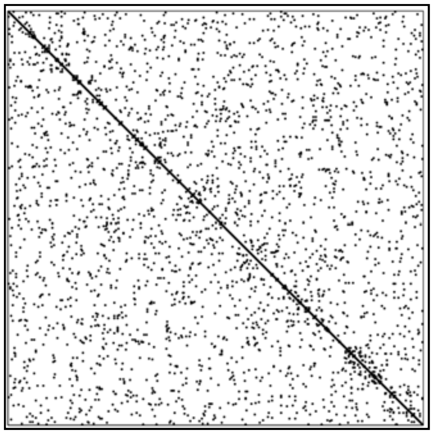


Sparse Cholesky Factorization

$M = LL^T$

500×500 matrix

3500 nonzeros

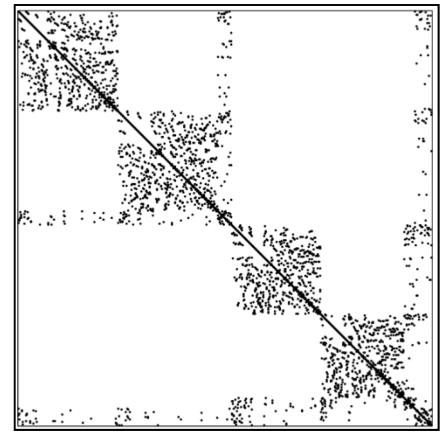


Reordering

PMP^T

nested dissection

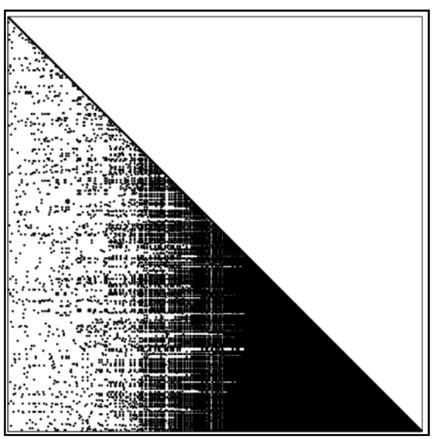
(parallelizable)



Cholesky Factorization

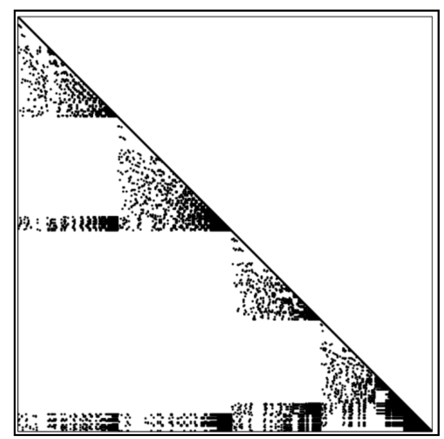
L

36k nonzeros



L

7k nonzeros



Direct Solvers

Discussion

- Highly accurate
  - Manipulate matrix structure
  - No iterations, everything is closed-form
- Easy to use
  - Off-the-shelf library, no parameters
- If M stays fixed, changing rhs ($\mathbf{r}$) is cheap
  - Just need to back-substitute (factor precomputed)

Direct Solvers

Discussion

- High memory cost
  - Need to store the factor, which is typically denser than the matrix M
- If the matrix M changes, need to re-compute the factor (expensive)

TAUCS tutorial

- TAUCS: a library of sparse linear solvers
  - Has both iterative and direct solvers
  - Direct (Cholesky and LU) use reordering and are very fast
- I provide a wrapper for TAUCS on the course homepage

TAUCS tutorial

- Basic operations:
  - Define a sparse matrix structure
  - Fill the matrix with its nonzero values (i, j, v)
  - Factor $A^T A$
  - Provide an rhs and solve

TAUCS tutorial

- Basic operations:
  - Define a sparse matrix structure

```
InitTaucsInterface();

int idA;
idA = CreateMatrix(4, 3);
```

#rows #cols

TAUCS tutorial

- Basic operations:
  - Fill the matrix A with its nonzero values (i, j, v)

```
SetMatrixEntry(idA, i, j, v);
```

TAUCS tutorial

- Basic operations:
  - Fill the matrix A with its nonzero values (i, j, v)

```
SetMatrixEntry(idA, i, j, v);
```

matrix ID, obtained in CreateMatrix

TAUCS tutorial

- Basic operations:
  - Fill the matrix A with its nonzero values (i, j, v)

```
SetMatrixEntry(idA, i, j, v);
```

row index i, column index j, zero-based

TAUCS tutorial

- Basic operations:
  - Fill the matrix A with its nonzero values (i, j, v)

```
SetMatrixEntry(idA, i, j, v);
```

value of matrix entry ij
for instance, $-w_{ij}$

TAUCS tutorial

- Basic operations:
  - Factor the matrix $A^T A$

```
FactorATA(idA);
```

TAUCS tutorial

- Basic operations:
  - Provide an rhs and solve

```
taucsType b[4] = {3, 4, 5, 6};
taucsType x[3];

SolveATA(idA, b, x, 1);
```

TAUCS tutorial

- Basic operations:
  - Provide an rhs and solve

```
taucsType b[4] = {3, 4, 5, 6};
taucsType x[3];

SolveATA(idA, b, x, 1);
```

typedef for `double`

TAUCS tutorial

- Basic operations:
  - Provide an rhs and solve

```
taucsType b[4] = {3, 4, 5, 6};
taucsType x[3];

SolveATA(idA, b, x, 1);
```

ID of the A matrix

TAUCS tutorial

- Basic operations:
  - Provide an rhs and solve

```
taucsType b[4] = {3, 4, 5, 6};
taucsType x[3];

SolveATA(idA, b, x, 1);
```

rhs for the LS system Ax = b

TAUCS tutorial

- Basic operations:
  - Provide an rhs and solve

```
taucsType b[4] = {3, 4, 5, 6};
taucsType x[3];

SolveATA(idA, b, x, 1);
```

array for the solution

TAUCS tutorial

- Basic operations:
  - Provide an rhs and solve

A is 4x3

```
taucsType b[4] = {3, 4, 5, 6};
taucsType x[3];

SolveATA(idA, b, x, 1);
```

number of rhs's

TAUCS tutorial

- Basic operations:
  - Provide an rhs and solve

A is 4x3

```
taucsType b2[8] = {3, 4, 5, 6, 7, 8, 9, 10};
taucsType xy[6];

SolveATA(idA, b2, xy, 2);
```

number of rhs's

TAUCS tutorial

- If the matrix A is square a priori, no need to solve the LS system
- Then just use `FactorA()`and `SolveA()`

Further Reading

- **Efficient Linear System Solvers for Mesh Processing**
  Mario Botsch, David Bommes, Leif Kobbelt
  Invited paper at IMA Mathematics of Surfaces XI, Lecture Notes in Computer Science, Vol 3604, 2005, pp. 62-83.

Next week

- By **September 28** you must:
  - Read up on all the projects and decide on your priorities
  - Discuss with me if you'd like to do a customized project (your own ideas)
  - **E-mail me your project preference (ranked list)**
- **No class meeting (do homework! ☺)**

Next week and after

- On September 28-29 I will assign a project to everyone
- **Next class: October 5; everyone presents their initial project plan**
- Up to 20 minutes presentation (can be shorter)
  - Explain the project in your words, with technical details
  - Outline your work plan
  - Bring up things that are unclear/challenges you foresee