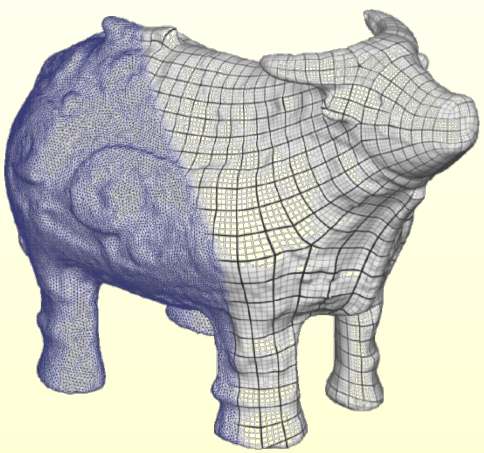
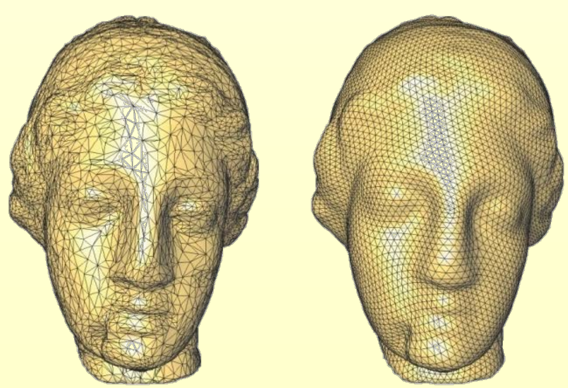
CS348a: Computer Graphics -- Geometric Modeling and Processing



Leonidas Guibas
Computer Science Dept.
Stanford University

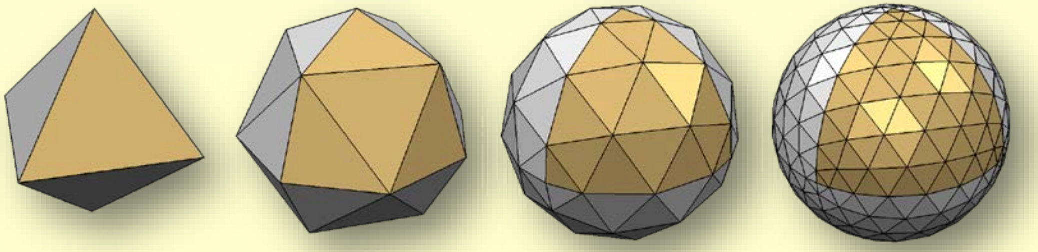


Acquired Shapes Overview I

- Shape acquisition
  - geometric 3D models derived from 3D scanners or other sensors (e.g., cameras)
- Geometry processing
  - techniques and algorithms for manipulating such raw geometric data to transform them into useful representations
- Geometry representation: triangle meshes
  - main questions:
    - why are triangle meshes a suitable representation for geometry processing?
    - what are the central processing algorithms?
    - how can they be implemented efficiently?

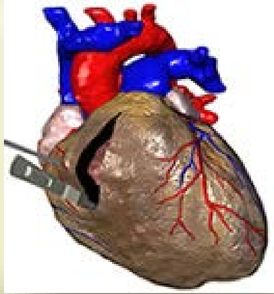
Acquired Shapes Overview II

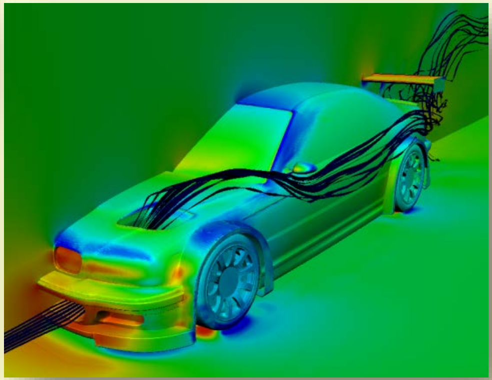
- Triangle meshes are splined surfaces
  - triangular patch surfaces of degree 1
- Triangle meshes can be big (1 billion vertices)
  - need efficient techniques and algorithms for manipulating such acquired geometric shapes



Application Areas

- Computer games
- Movie production
- Engineering
- Medical applications
- Architecture
- etc.





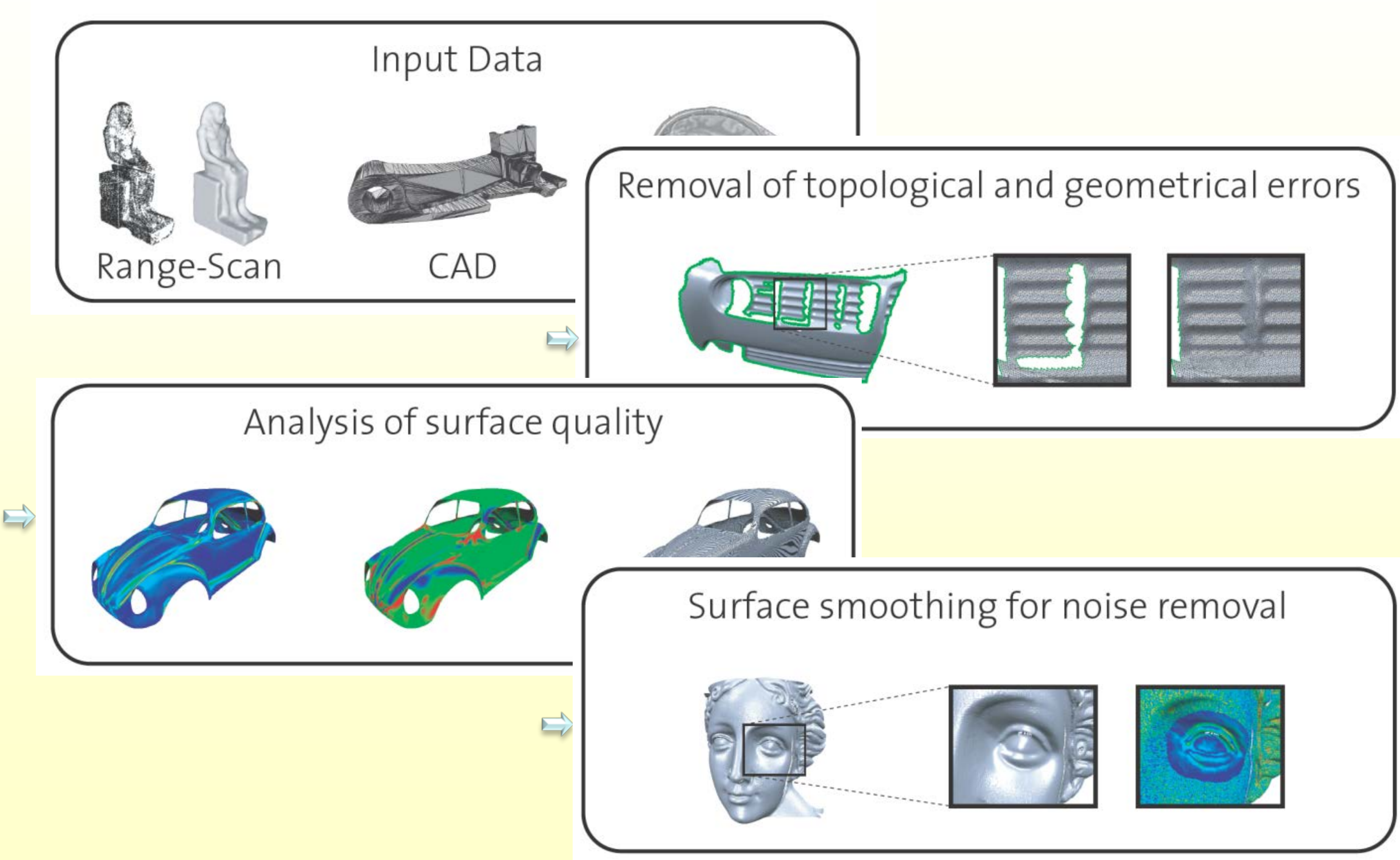
What is Geometry Processing About?

- Acquiring
- Analyzing/Improving
- Manipulating

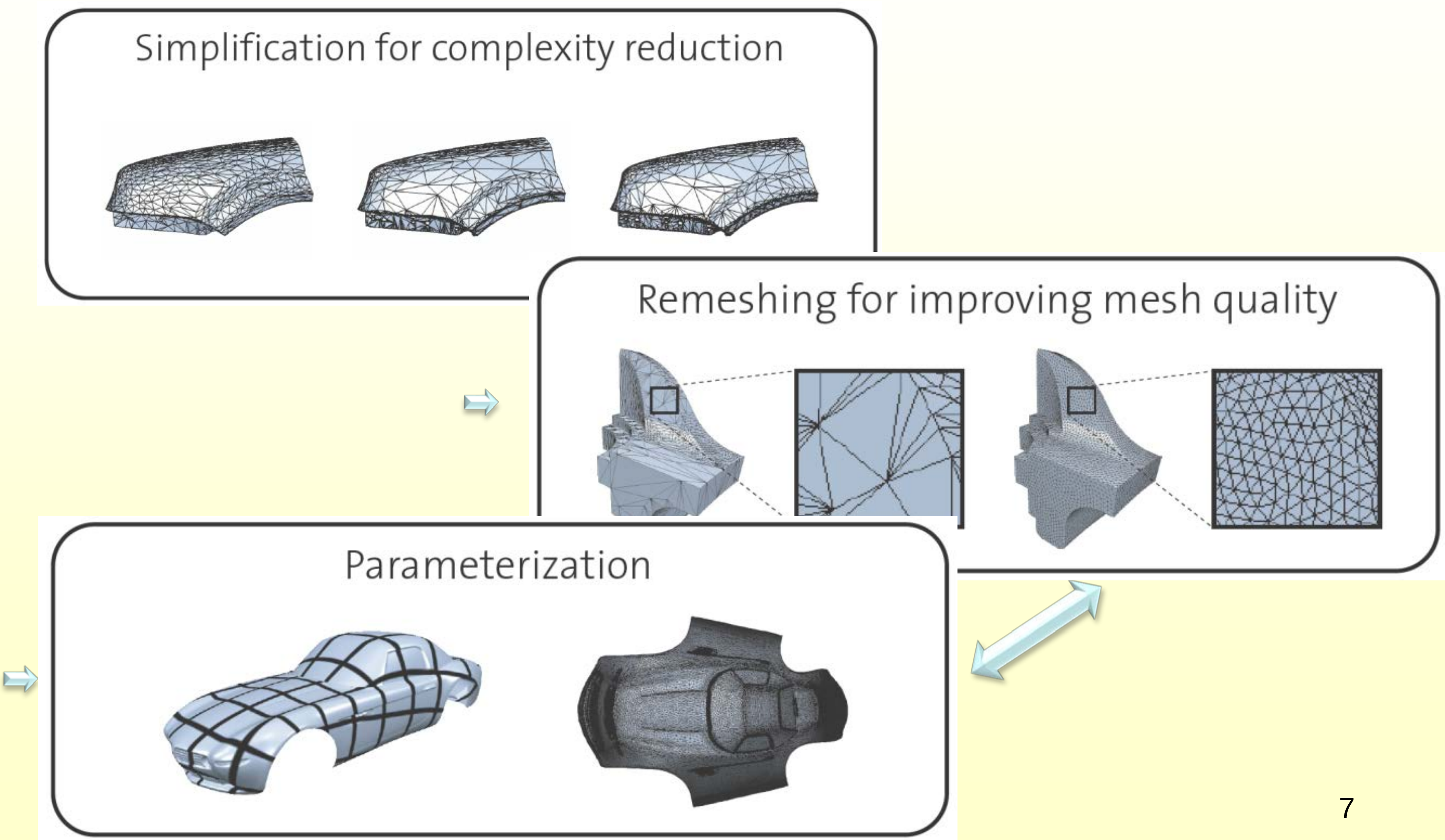
3D Models

A Geometry Processing Pipeline

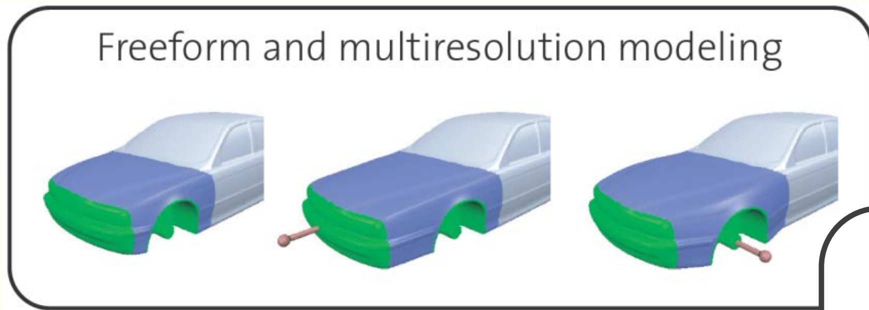
Low Level Algorithms

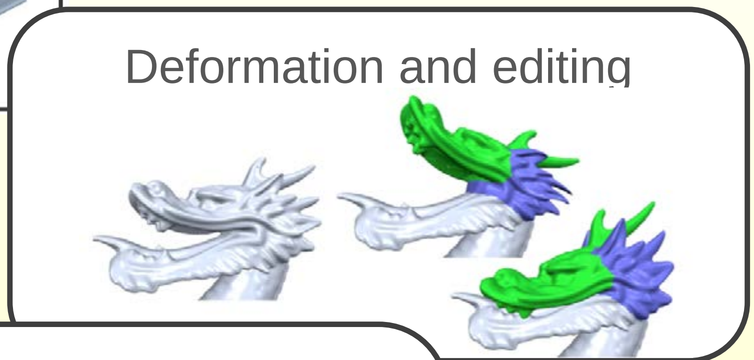


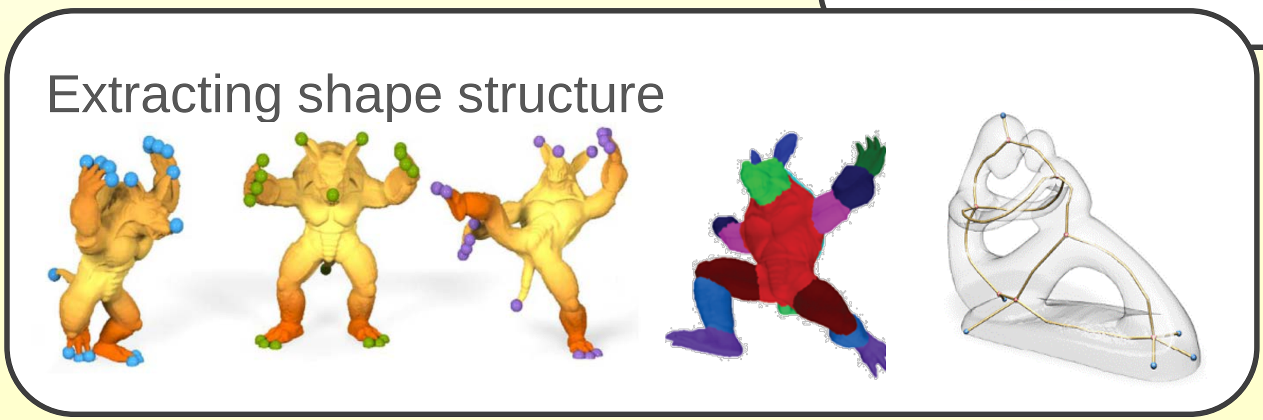
A Geometry Processing Pipeline



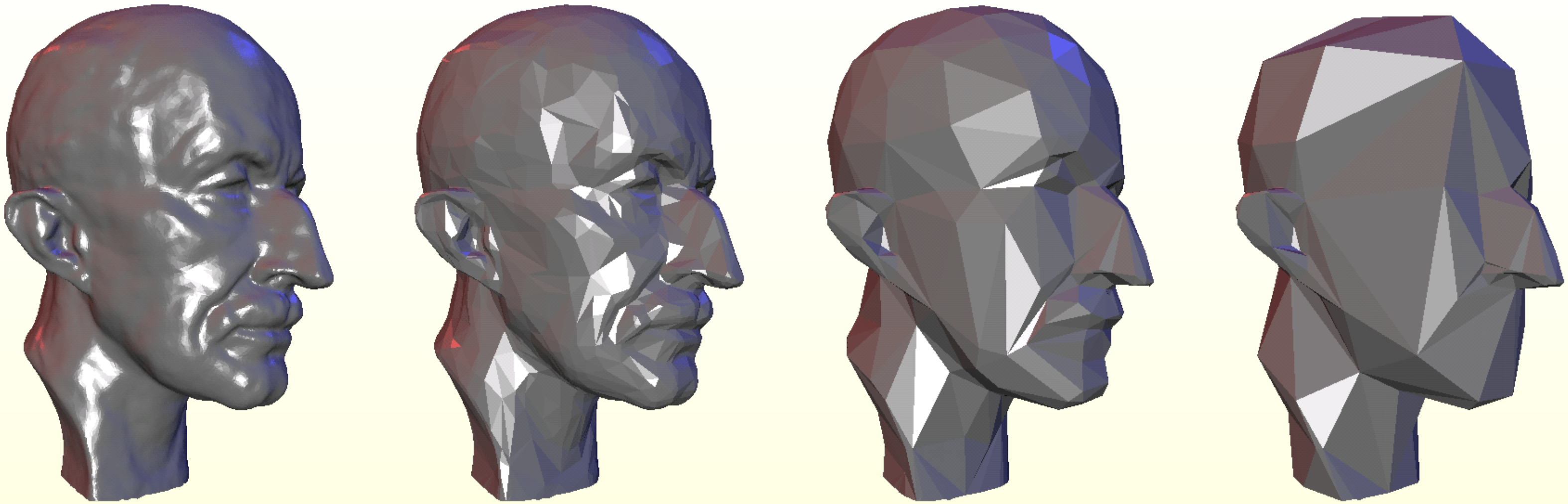
A Geometry Processing Pipeline
High Level Algorithms

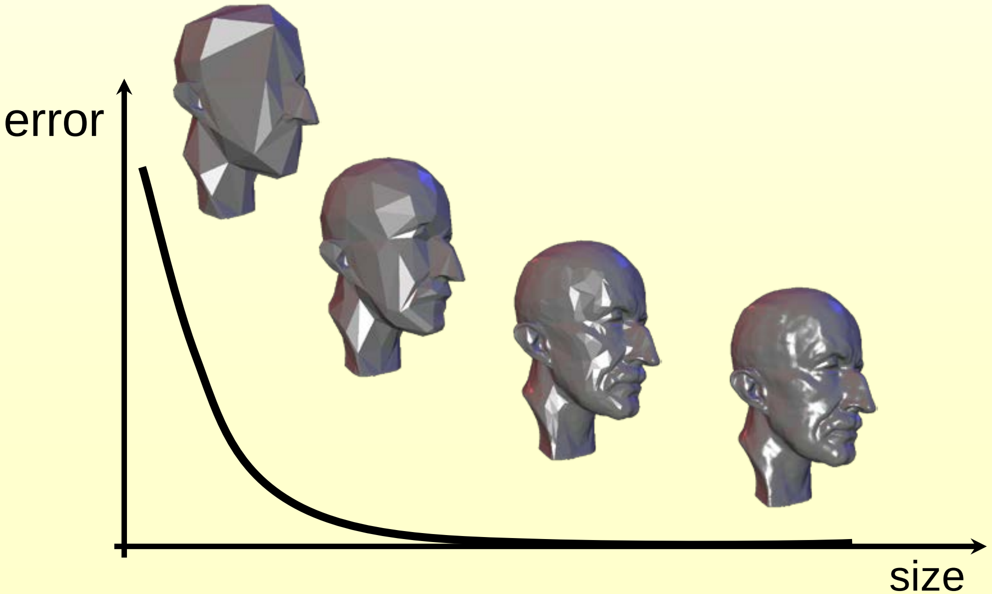






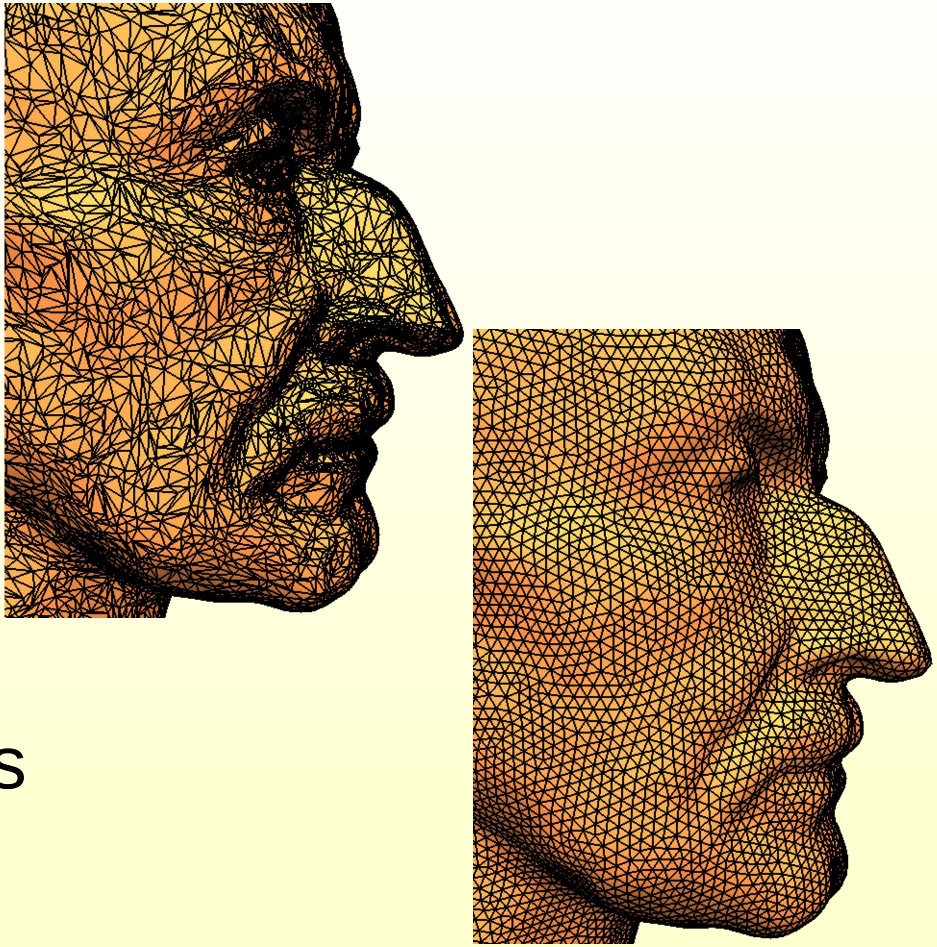
Simplification



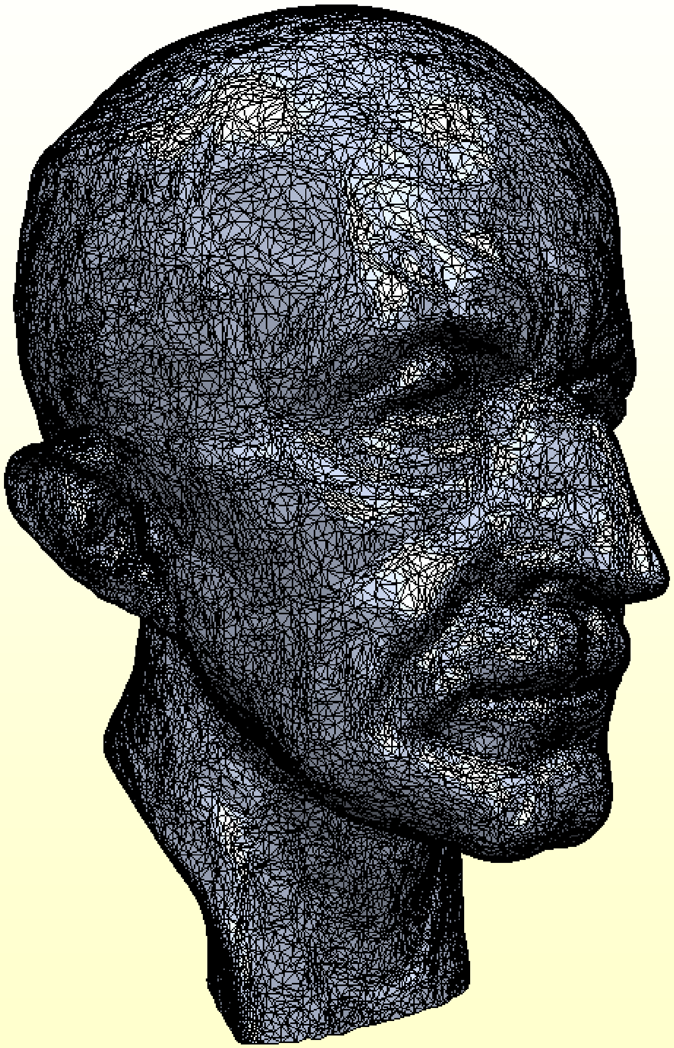


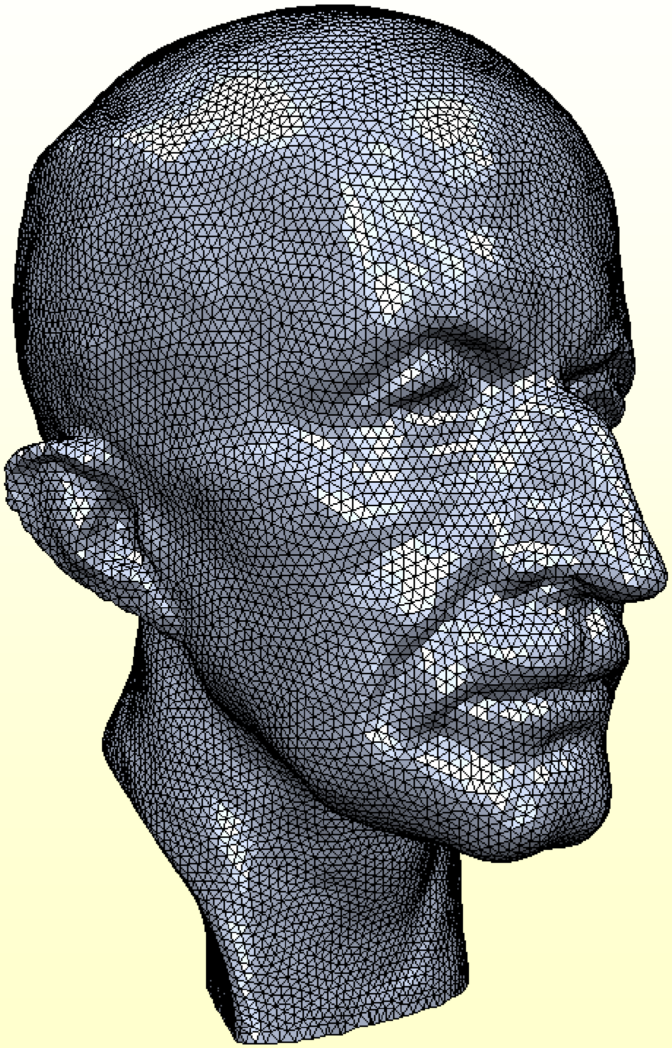
Mesh Quality Criteria

- Smoothness
  - Low geometric noise
- Adaptive tessellation
  - Low complexity
- Triangle shape
  - Numerical robustness



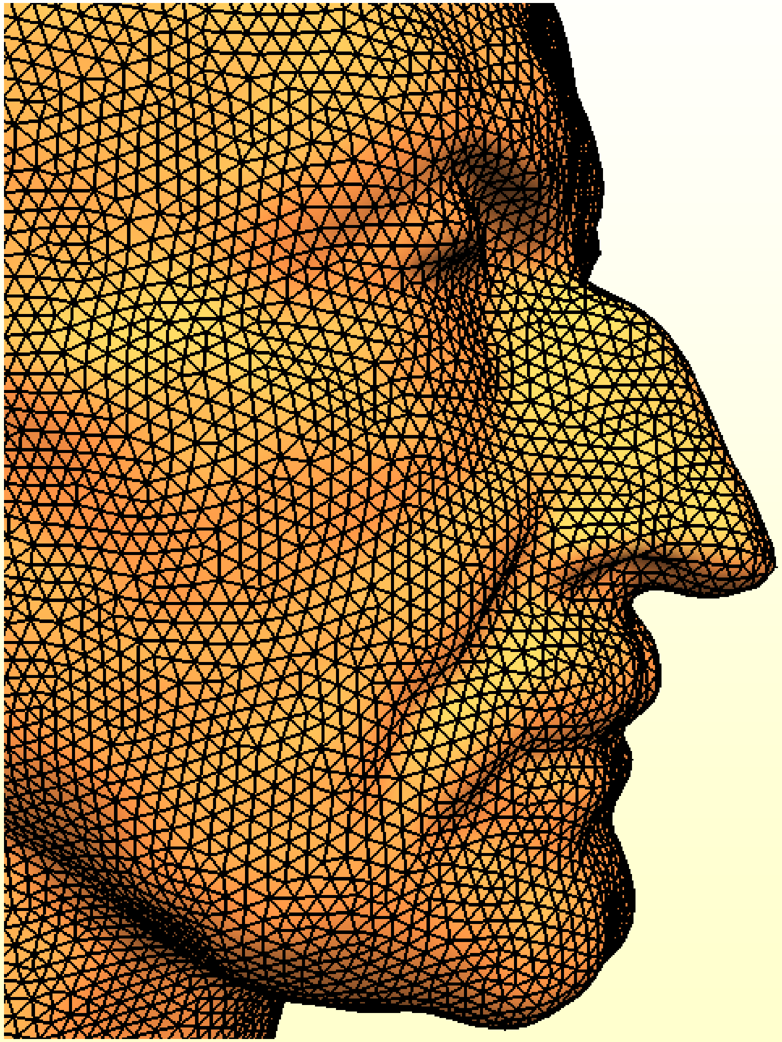
What is a Good Mesh?





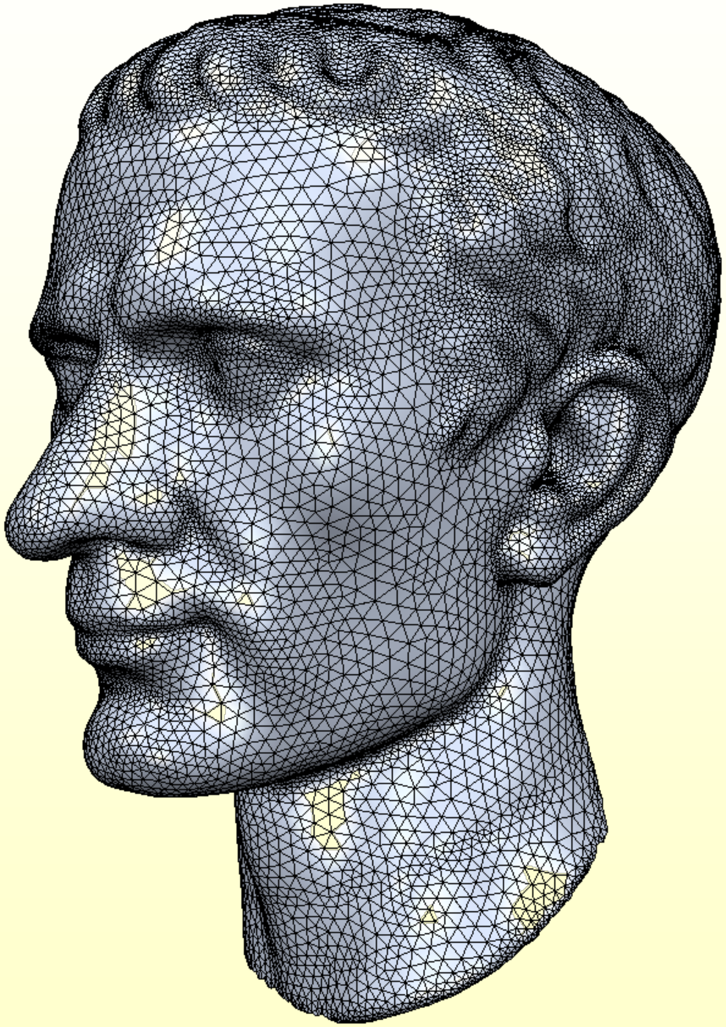
What is a Good Mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6



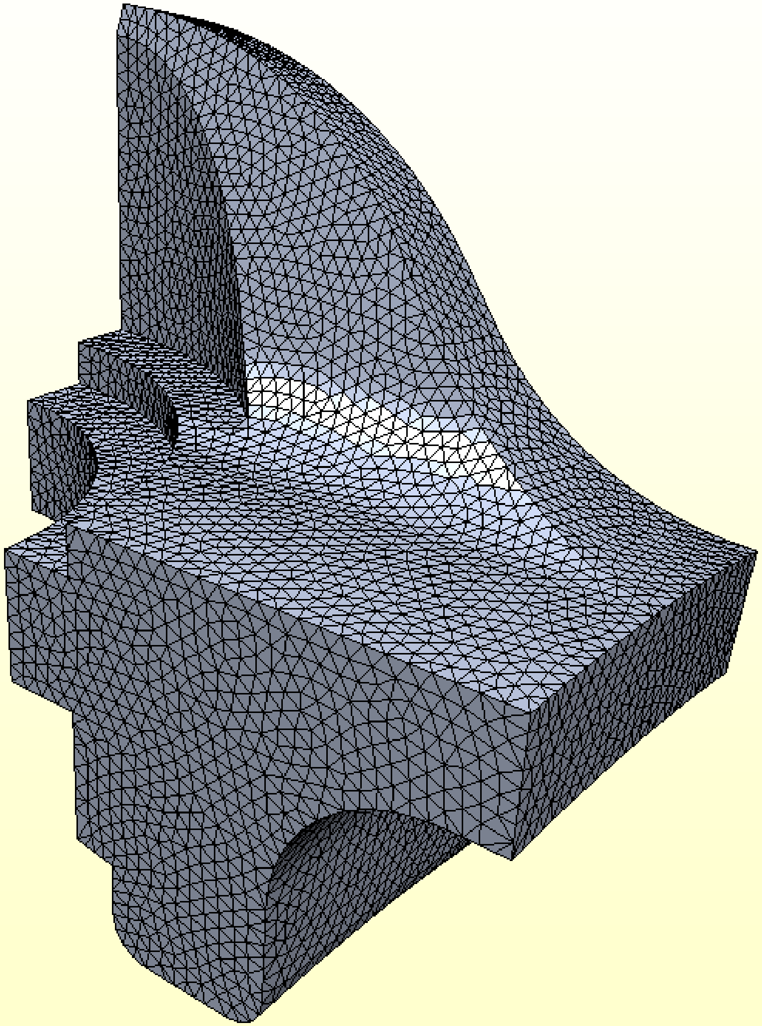
What is a Good Mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling



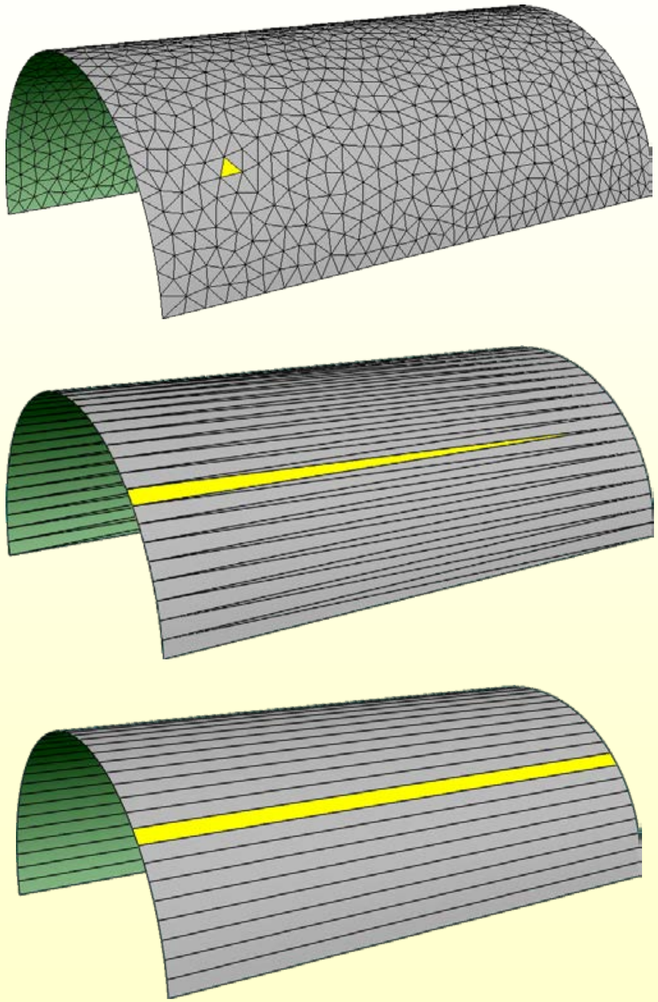
What is a Good Mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling
- Feature preservation



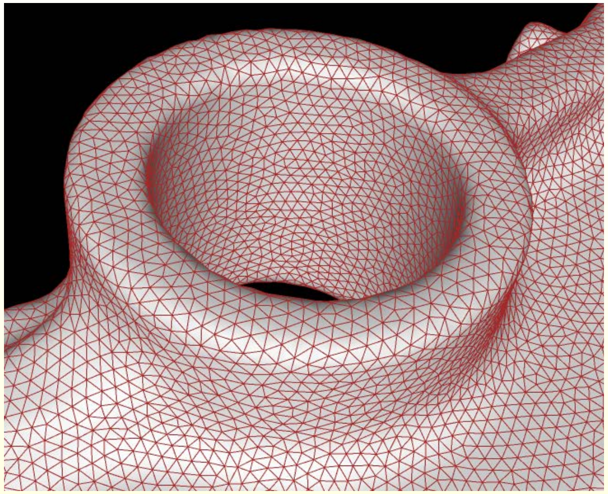
What is a Good Mesh?

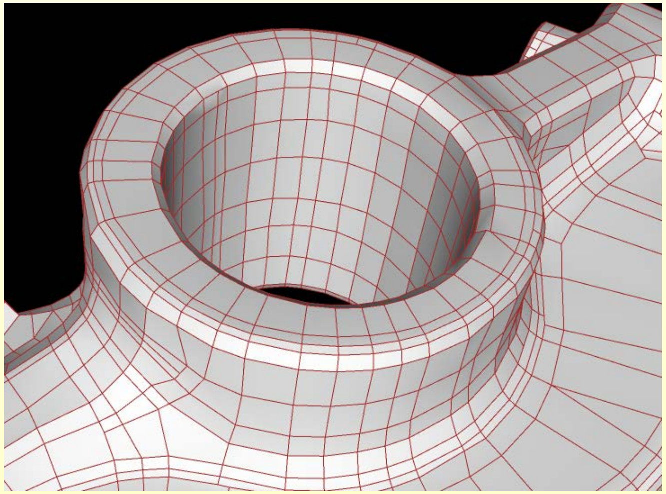
- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling
- Feature preservation
- Alignment to curvature lines
- Isotropic vs. anisotropic



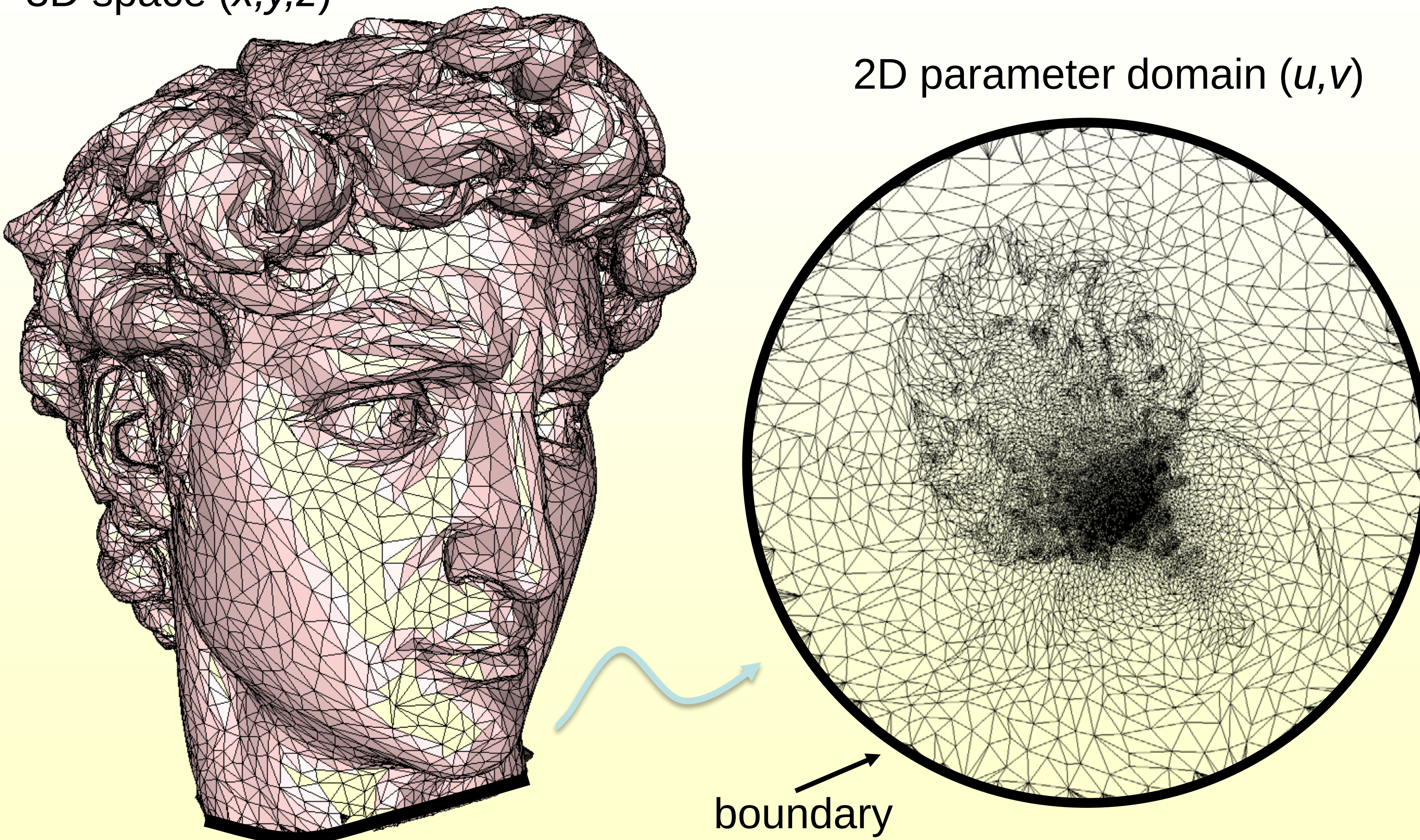
What is a Good Mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling
- Feature preservation
- Alignment to curvature lines
- Isotropic vs. anisotropic
- Triangles vs. quadrangles



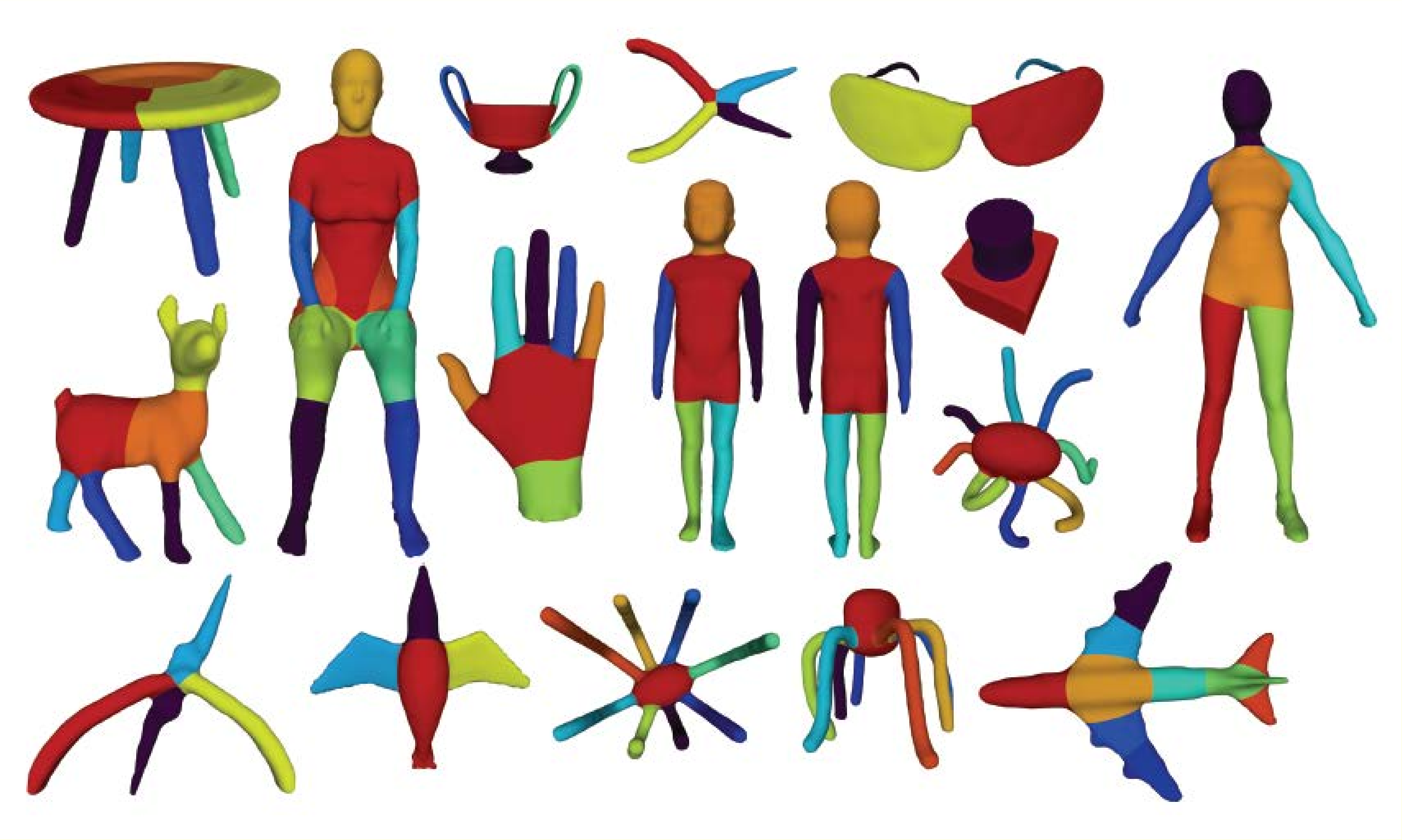


Parametrization

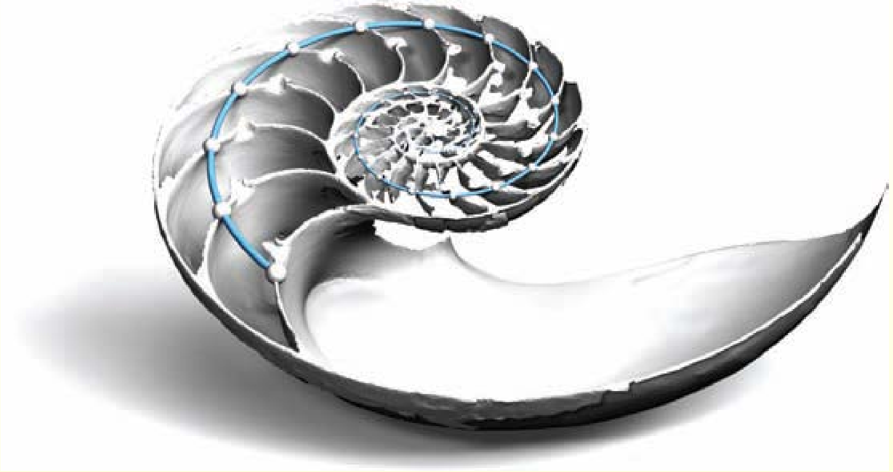


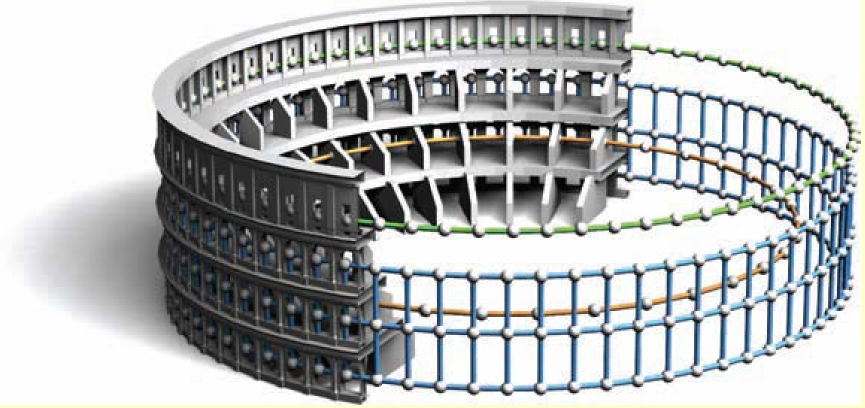
Application -- Texture Mapping

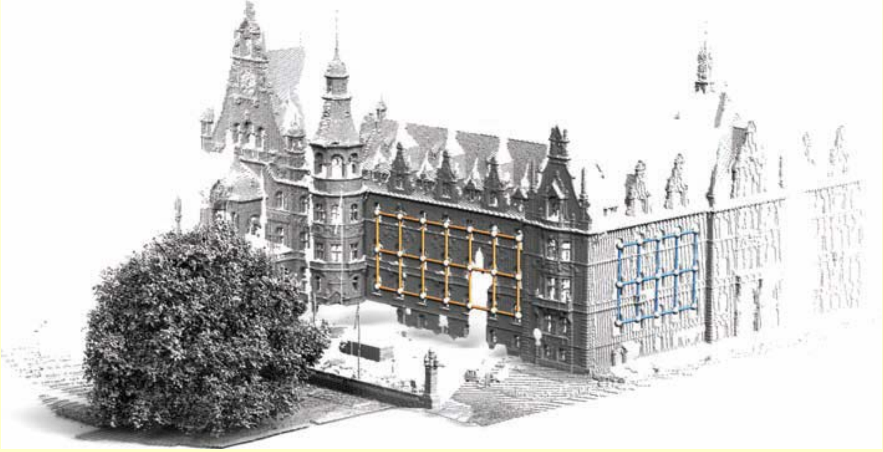
Segmentation



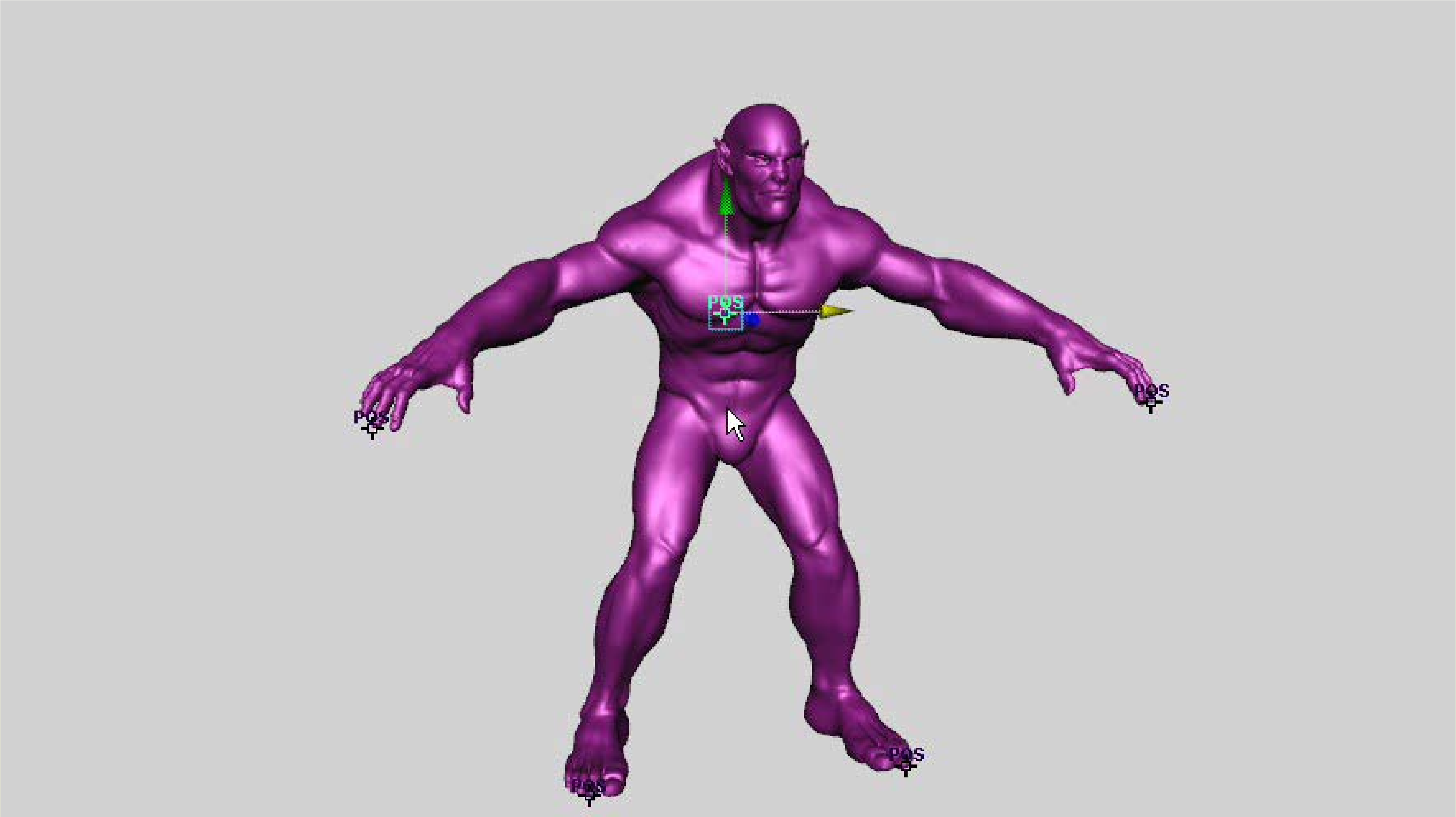
Symmetry Detection







Deformation / Manipulation



From Point Clouds to Surfaces

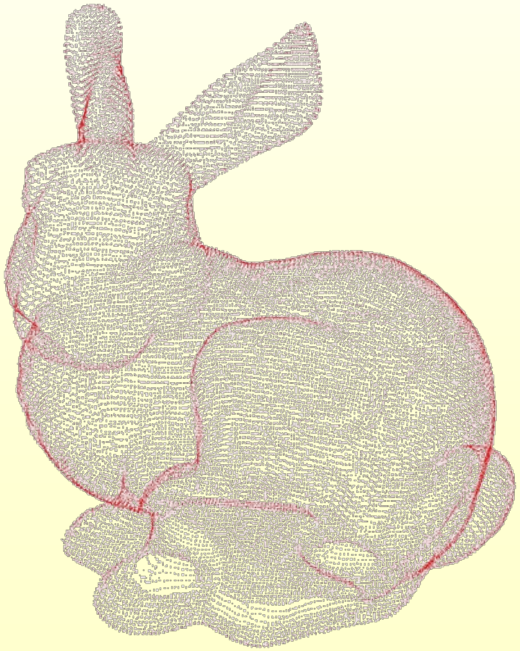
physical
model

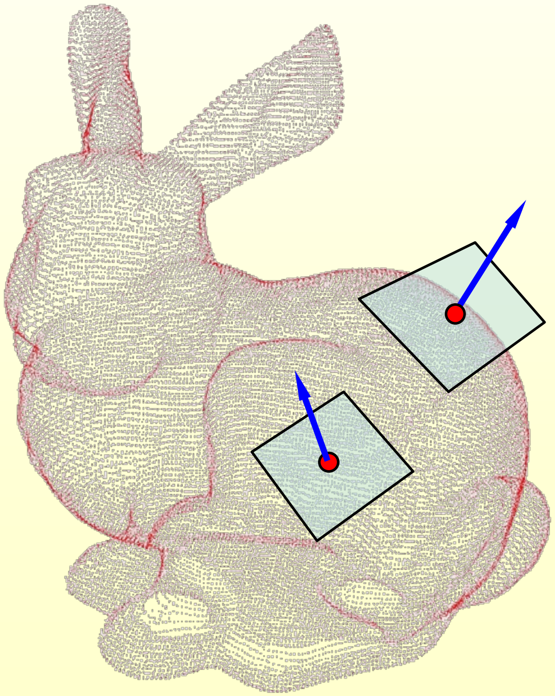
acquired
point cloud

reconstructed
3D model

Point Clouds

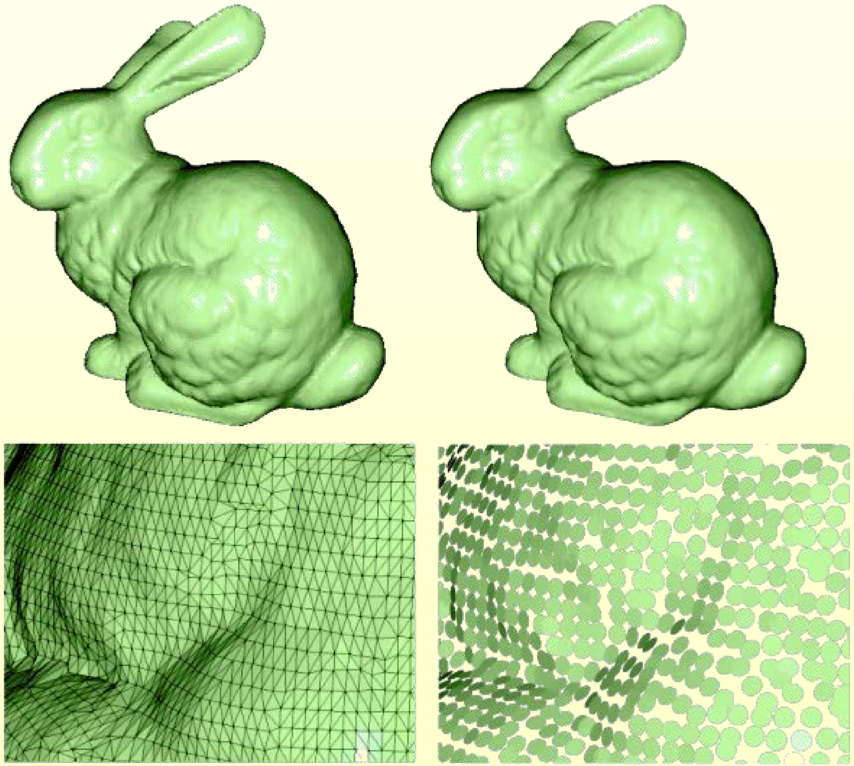
- Simplest representation: **only points,** no connectivity
- Collection of (x,y,z) coordinates, possibly with normals





Point Clouds

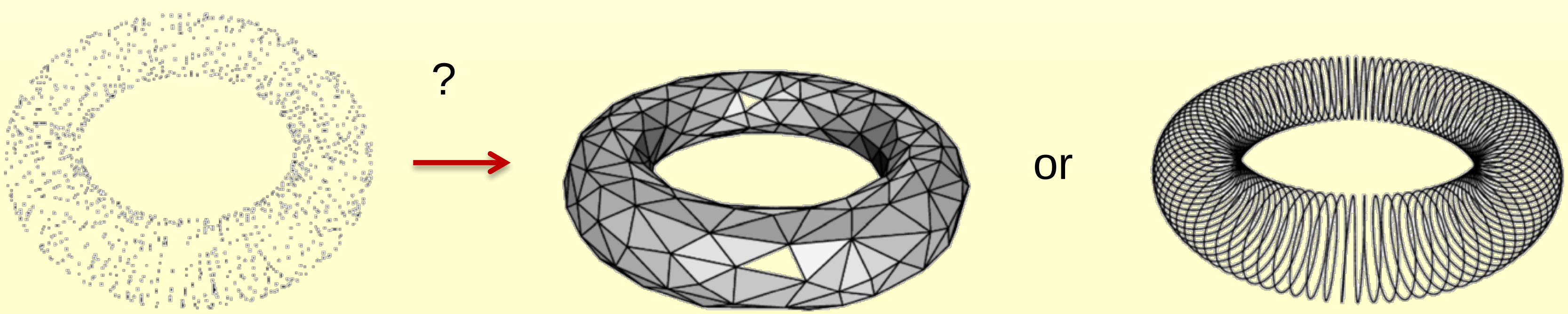
- Simplest representation: **only points,** no connectivity
- Collection of (x,y,z) coordinates, possibly with normals
- Points with orientation are called **surfels**



Filip van Bouwel

Point Clouds

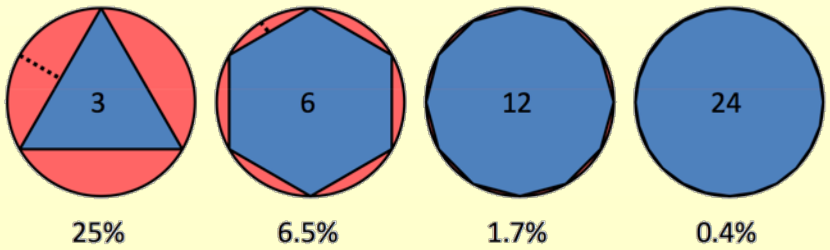
- Simplest representation: **only points,** no connectivity
- Collection of (x,y,z) coordinates, possibly with normals
- Points with orientation are called **surfels**
- Severe limitations:
  - **no** simplification or subdivision
  - **no** direct smooth rendering
  - **no** topological information



Point Clouds

- Simplest representation: **only points,** no connectivity
- Collection of (x,y,z) coordinates, possibly with normals
- Points with orientation are called **surfels**
- Severe limitations:
  - **no** simplification or subdivision
  - **no** direct smooth rendering
  - **no** topological information
  - weak approximation power:

- Piecewise linear approximation
  - Error is $O(h^2)$

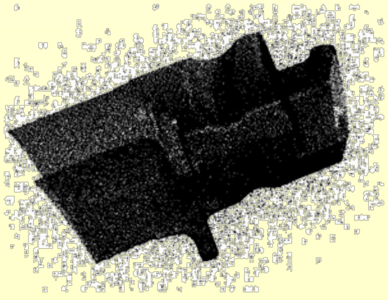


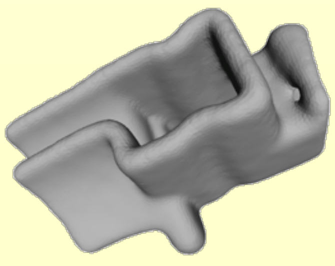
Point Clouds

- Simplest representation: **only points,** no connectivity
- Collection of (x,y,z) coordinates, possibly with normals
- Points with orientation are called **surfels**
- Severe limitations:
  - **no** simplification or subdivision
  - **no** direct smooth rendering
  - **no** topological information
  - weak approximation power: $O(h)$ for point clouds
    - need *square* number of points for the same approximation power as meshes

Point Clouds

- Simplest representation: **only points,** no connectivity
- Collection of (x,y,z) coordinates, possibly with normals
- Points with orientation are called **surfels**
- Severe limitations:
  - **no** Simplification or subdivision
  - **no** direct smooth rendering
  - **no** topological information
  - weak approximation power
  - noise and outliers

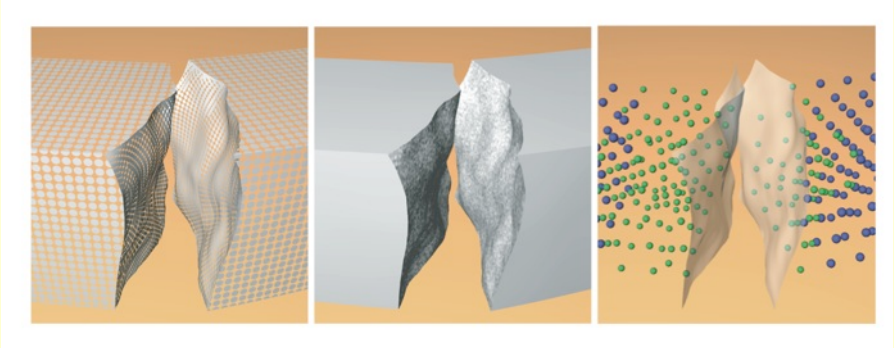




Why Point Clouds?

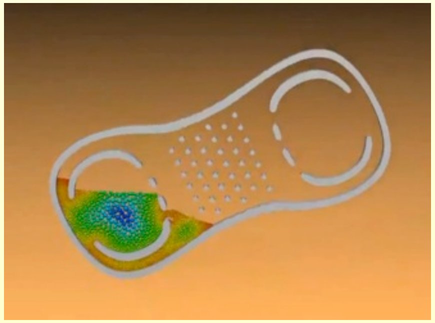
1) Typically, that's the only thing that's available
2) Isolation: sometimes, easier to handle (esp. in hardware).

Fracturing Solids



Meshless Animation of Fracturing Solids
Pauly et al., SIGGRAPH '05

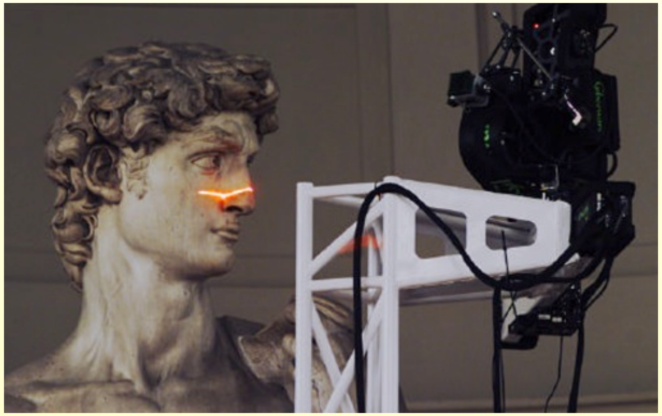
Fluids



Adaptively sampled particle fluids,
Adams et al. SIGGRAPH '07

Why Point Clouds?

- Typically, that’s the only thing that’s available
  - Nearly all 3D scanning devices produce point clouds

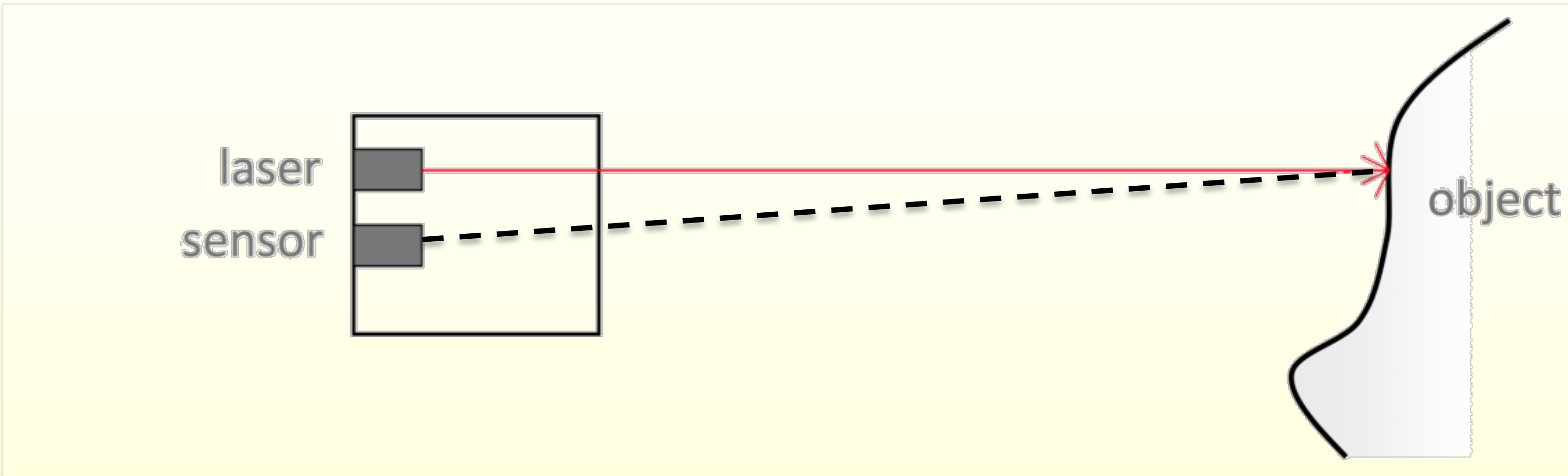


Surface Scanning Basics

Major types of 3D scanners

- **Range (emission-based) scanners**
  - Time-of-flight laser scanner
  - Phase-based laser scanner
- **Triangulation**
  - Laser line sweep
  - Structured light
- • **Stereo / computer vision**
  - Passive stereo
  - Active stereo / space-time stereo

Time of Flight Scanners



1. Emit a short pulse of laser
2. Capture the reflection.
3. Measure the time it took to come back.

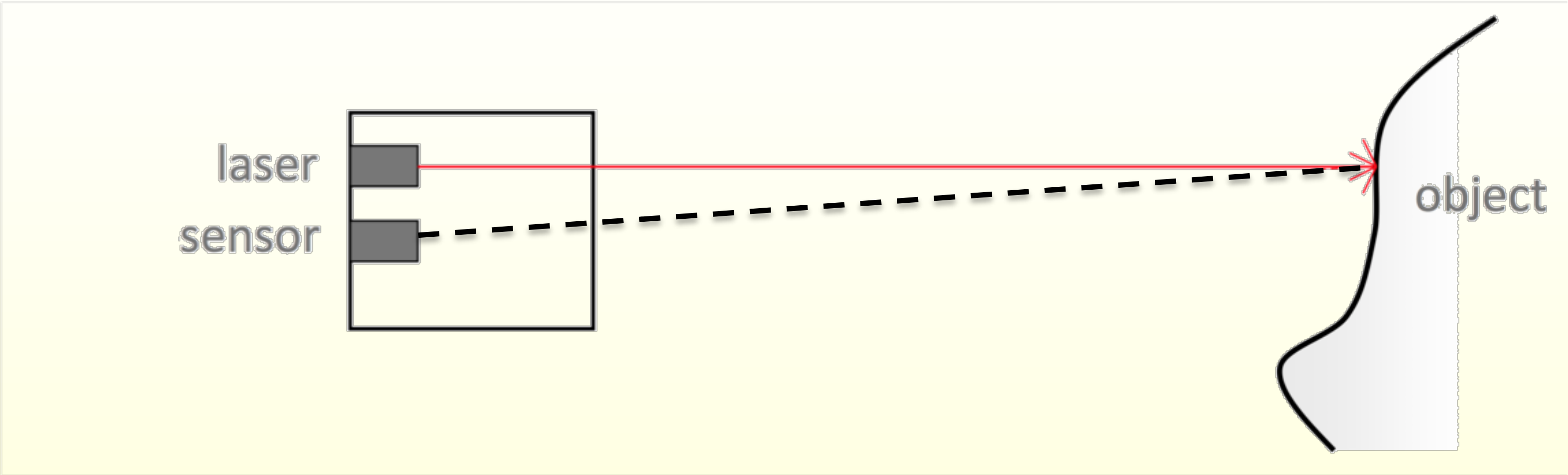
$$D = \frac{cT}{2}$$

c: speed of light (≈ 299 792 458 m/s)

Need a very fast clock: e.g. 1GHz achieves 0.15m (15cm) accuracy.

Guinness record: AMD Bulldozer, 8.429 GHz

Time of Flight Scanners

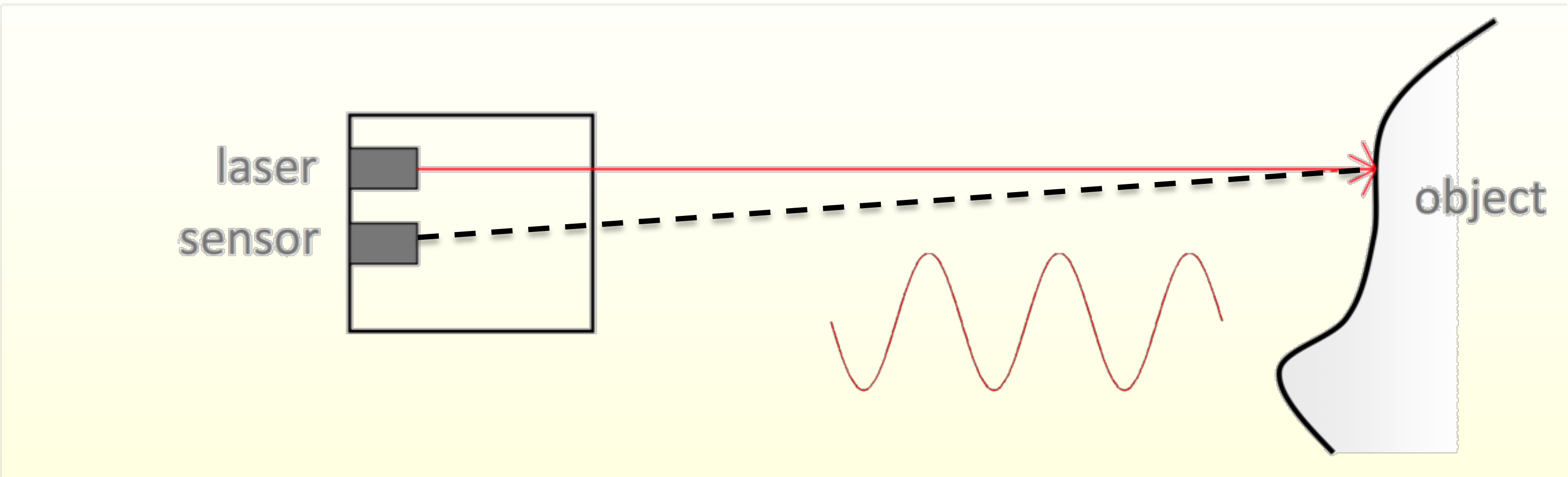


1. Emit a short pulse of laser
2. Capture the reflection.
3. Measure the time it takes to come back.
4. Need a very fast clock.
5. Main advantage: can be done over long distances.
6. Used in terrain scanning.

Time of Flight Scanners

source: Michael Wand

Phase-Based Range-Scanners



1. Instead of a pulse, emit a continuous **phase-modulated** beam
2. Capture the reflection
3. Measure the **phase-shift** between the output and input signals

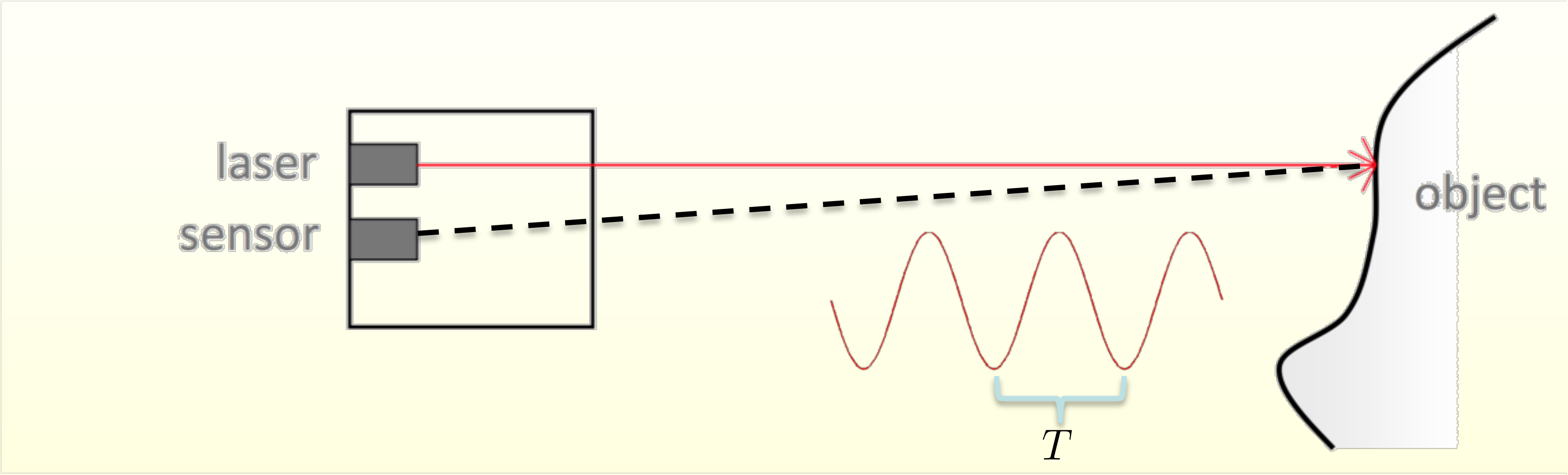
Output: $$e(t) = e \cdot \left[1 + \sin\left(\frac{F}{2\pi} \cdot t\right)\right]$$

F: Modulation frequency
e: emitted mean power

Input: $$s(t) = e \cdot \left[1 + \sin\left(\frac{F}{2\pi} \cdot t - \phi\right)\right]$$

φ: Phase delay arising from the object's distance

Phase-Based range-scanners

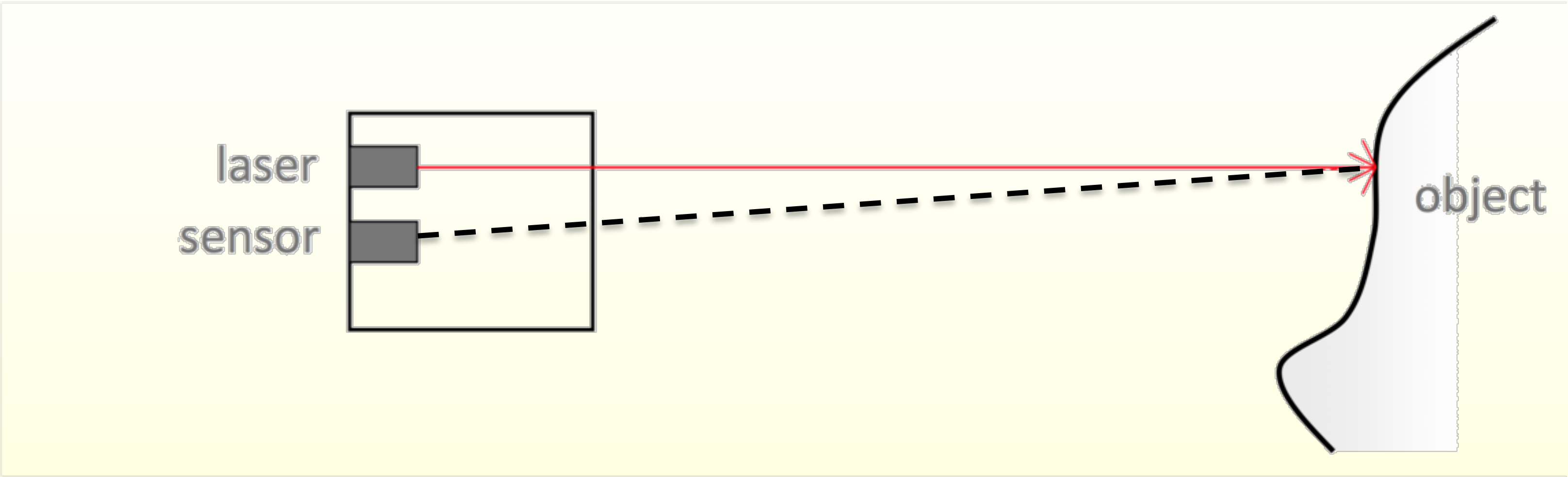


1. Instead of a pulse, emit a continuous **phase-modulated** beam
2. Capture the reflection
3. Measure the **phase-shift** between the output and input signals.
4. From the phase-shift, the distance can be computed **up to modulation period**
5. No fast clock required, **greater frequency and accuracy** but **shorter range**

e.g. 1,016,727 vs. 50,000 (ToF) points per second

up to 79 meters vs. hundreds of meters

Range-Scanners

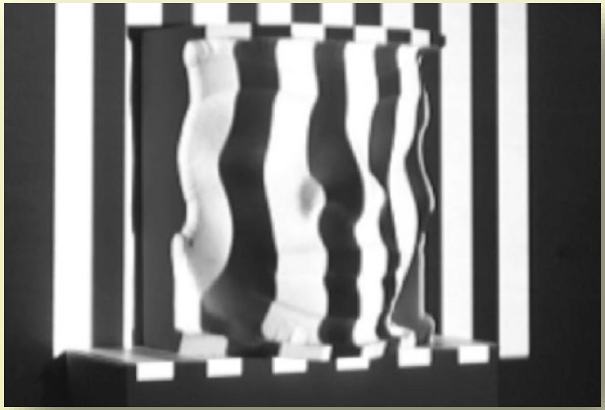


1. Typically, range scanners by themselves provide limited accuracy (noise, outliers, uneven sampling).
2. May require a lot of post-processing to get a good sampling.

Triangulation-Based Approaches (Laser or Structured Light)

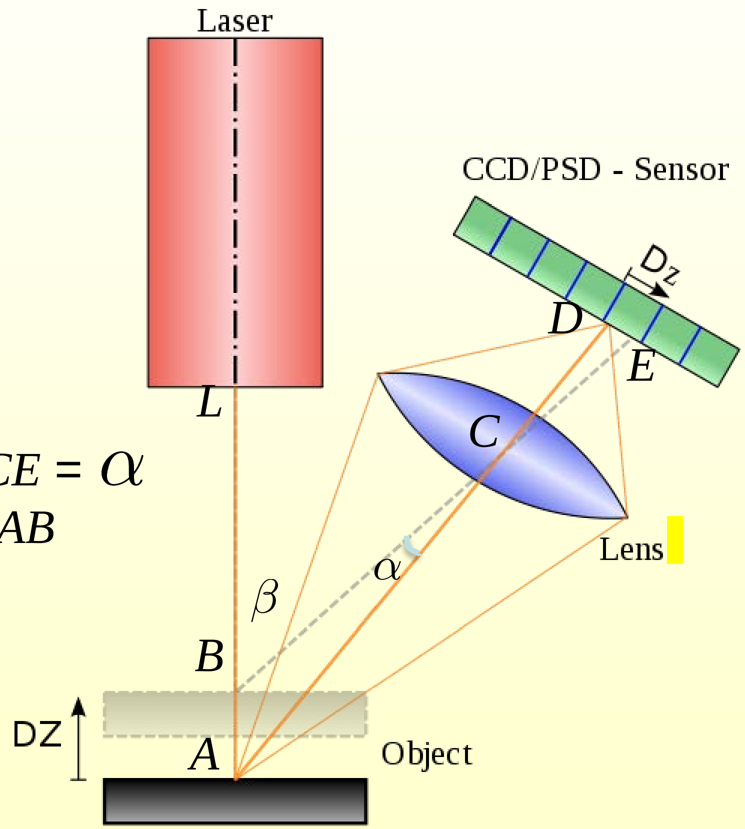
1. Add a **photometric sensor** (e.g. camera)
2. Record the position for a reference plane
3. Change in recording position can be used to recover the depth.

Intuition: the **depth** is related to the **shift** in the camera plane.



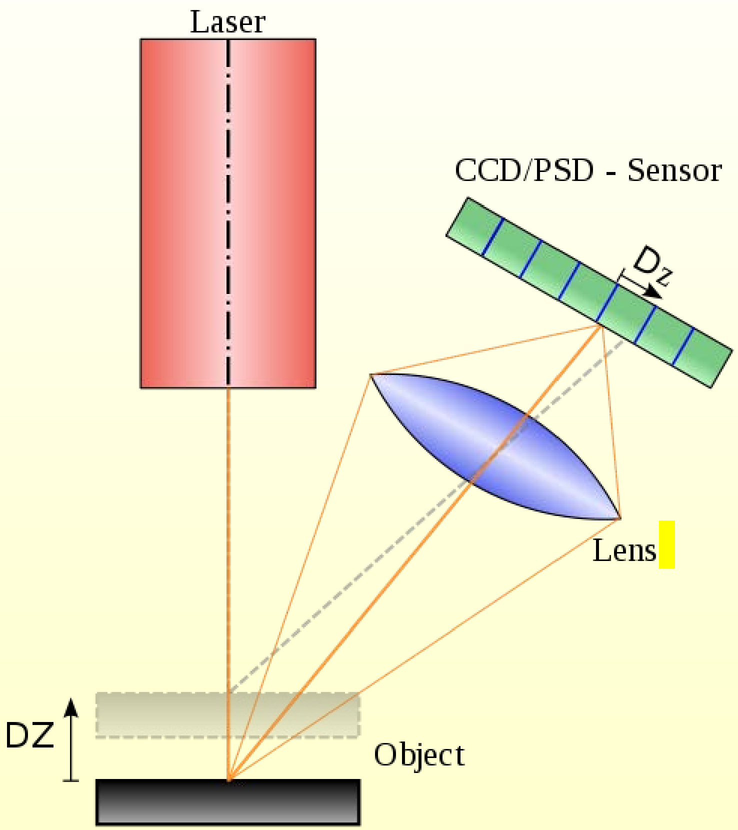
Triangulation-Based Approaches (Laser)

1. Add a **photometric sensor** (e.g. camera)
2. Record the position for a reference plane
3. Change in recording position can be used to recover the depth
   1. Using Dz, EC and $\angle CED$, compute $\angle DCE = \alpha$
   2. Using α, β and $BC = BE - CE$, compute AB
   3. The depth $LE = LB + AB$



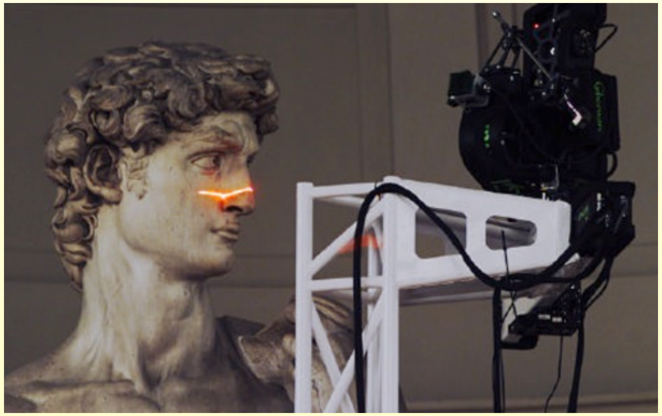
Triangulation-Based Approaches (Laser)

1. Add a **photometric sensor** (e.g. camera)
2. Record the position for a reference plane
3. Change in recording position can be used to recover the depth.
4. If well-calibrated, can lead to extremely accurate depth measurements

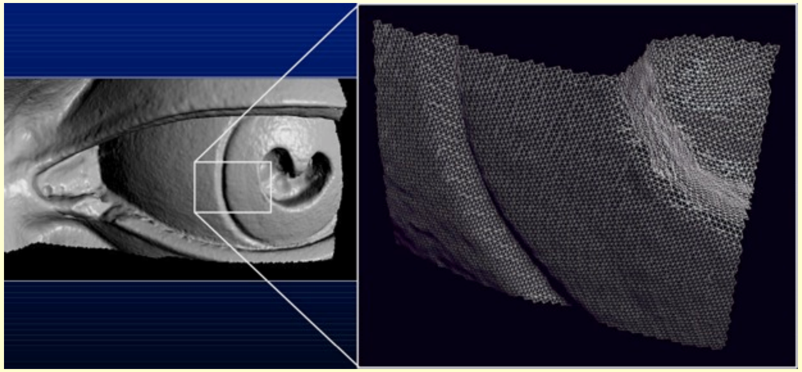


Triangulation-Based Approaches (Laser)

1. Add a **photometric sensor** (e.g. camera)
2. Record the position for a reference plane
3. Change in recording position can be used to recover the depth
4. If well-calibrated, can lead to extremely accurate depth measurements



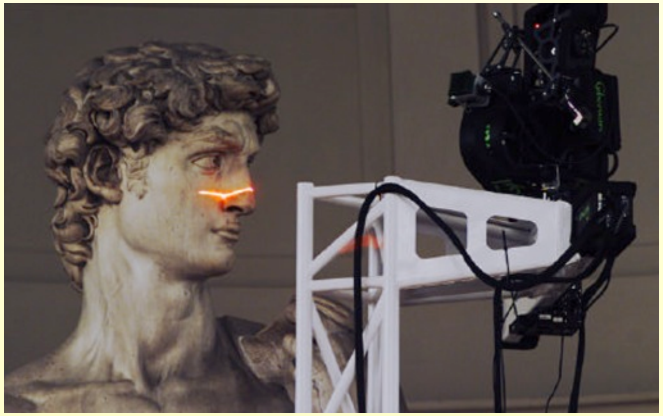
Similar technology used to scan Michelangelo's David 5m statue to 0.25mm accuracy.



David's left eye: source Levoy et al.

Triangulation-Based Approaches (Laser)

1. Add a **photometric sensor** (e.g. camera)
2. Record the position for a reference plane
3. Change in recording position can be used to recover the depth
4. If well-calibrated, can lead to extremely accurate depth measurements
5. Main problem: slow and expensive

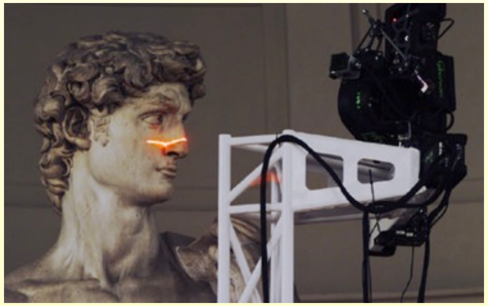


Structured-Light Scanners

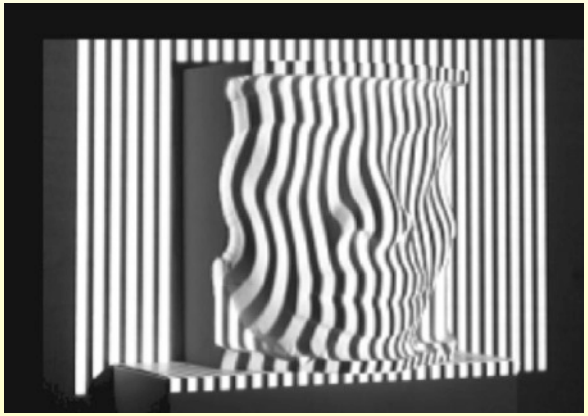
Same general idea as triangulation based scanner.

Main Idea: Replace laser with projector. Project stripes instead of sheets.

Challenge: Need to identify which (input/output) lines correspond.



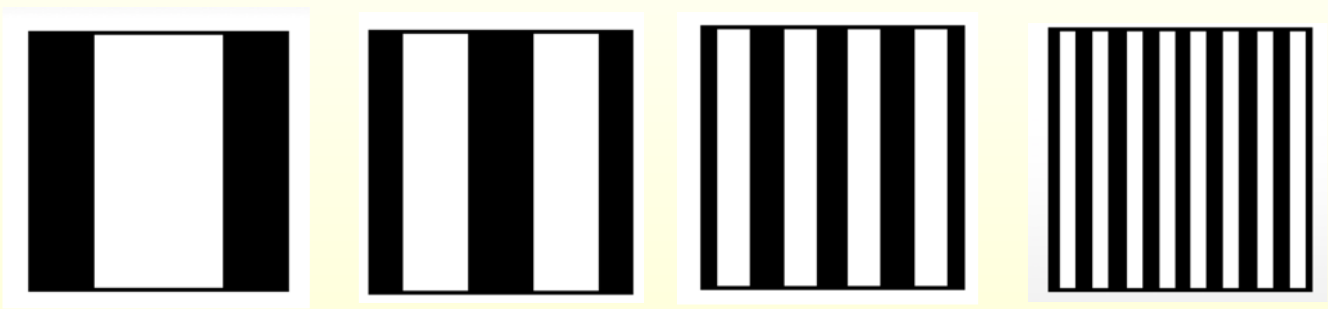
vs.



Structured-Light Scanners

Same idea as triangulation-based scanner.

Main Idea: Project multiple stripes to identify the position of a point.

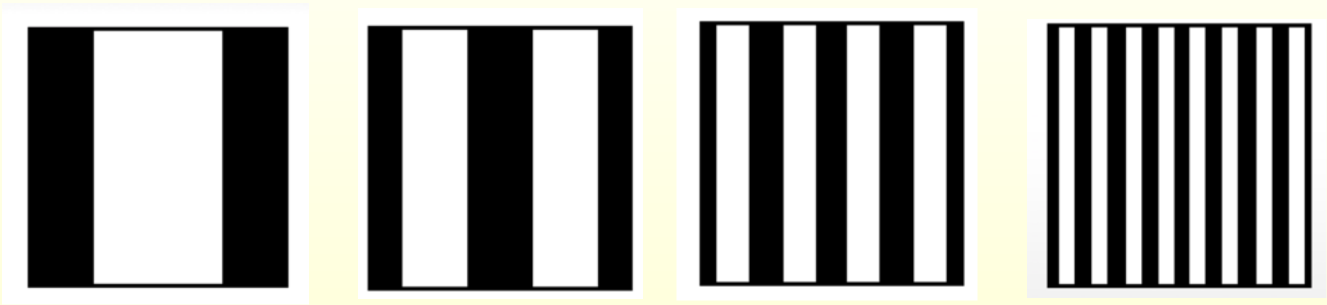


log(N) projections are sufficient to identify N stripes.

Structured-Light Scanners

Same idea as triangulation based scanner.

Main Idea: Project multiple stripes to identify the position of a point.



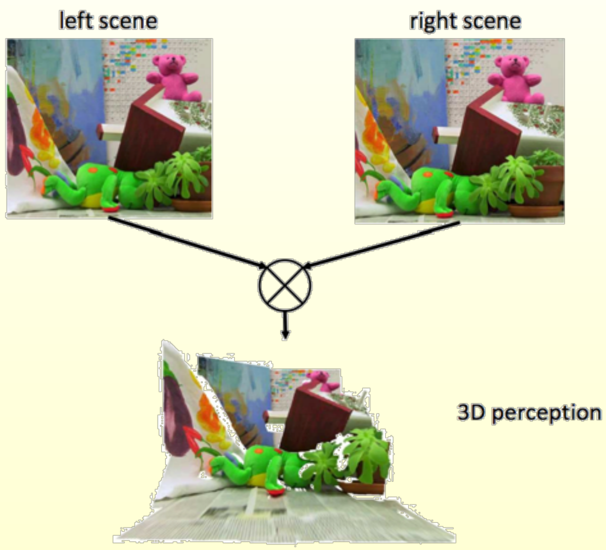
log(N) projections are sufficient to identify N stripes.

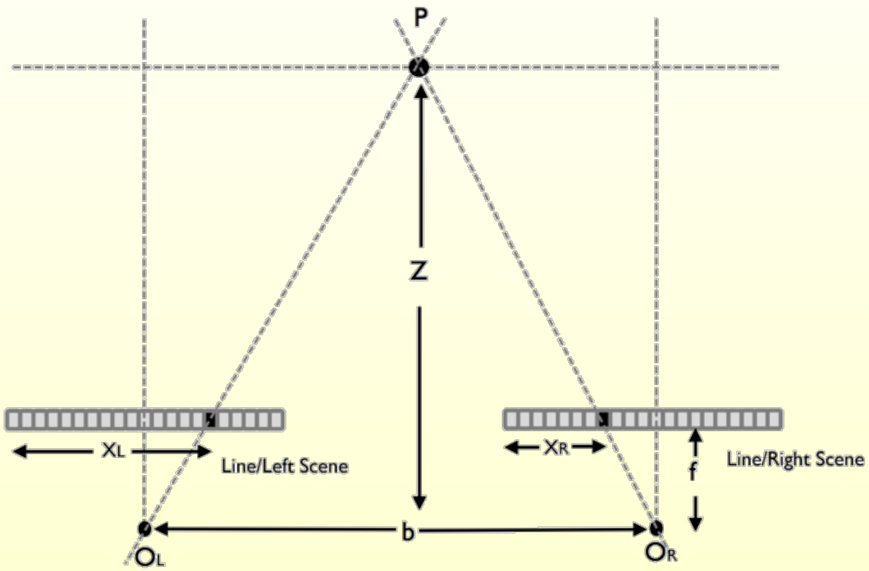
Advantage: cost and speed

Disadvantage: need controlled conditions & projector calibration.

Computer Vision Based Techniques

Depth from stereo:

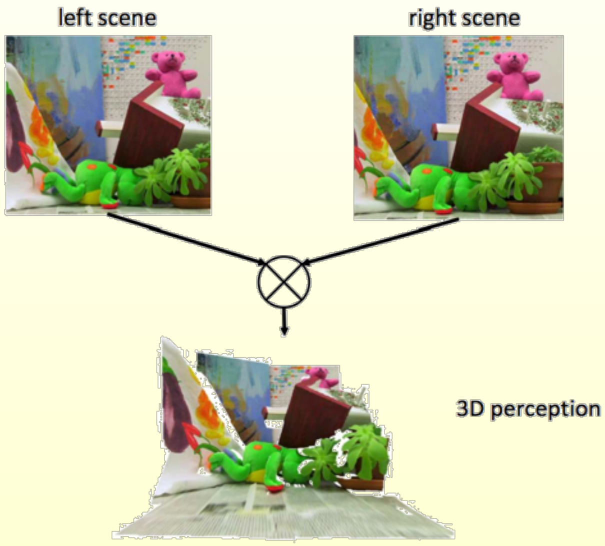


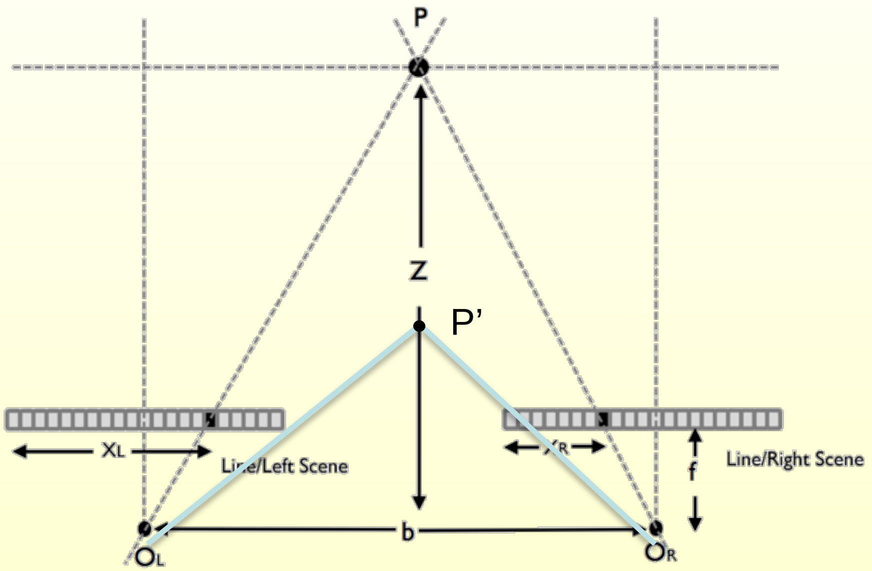


Given 2 images, shift in the x-axis is related to the depth.

Computer Vision based Techniques

Depth from stereo:





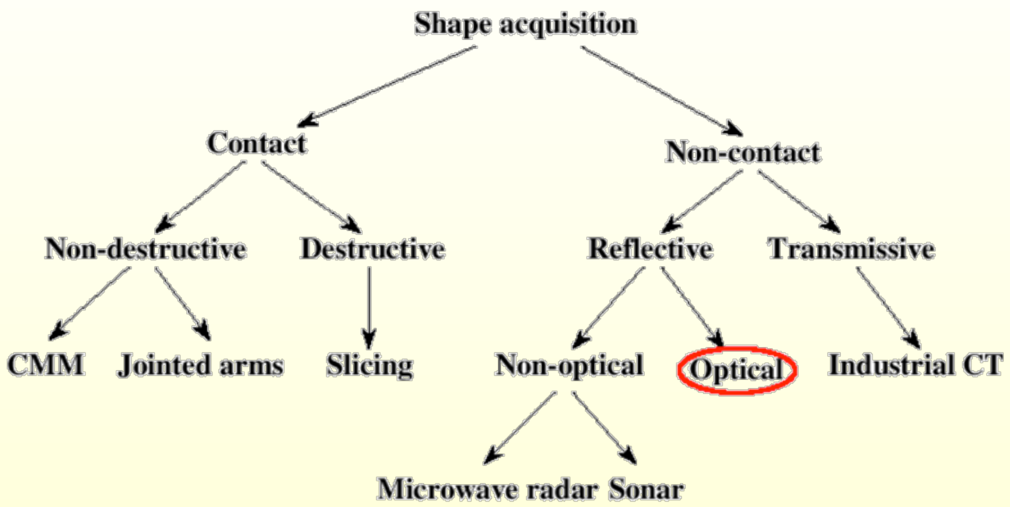
Given 2 images, shift in the x-axis is related to the depth.
Main challenge: establishing corresponding points across images: **very difficult.**

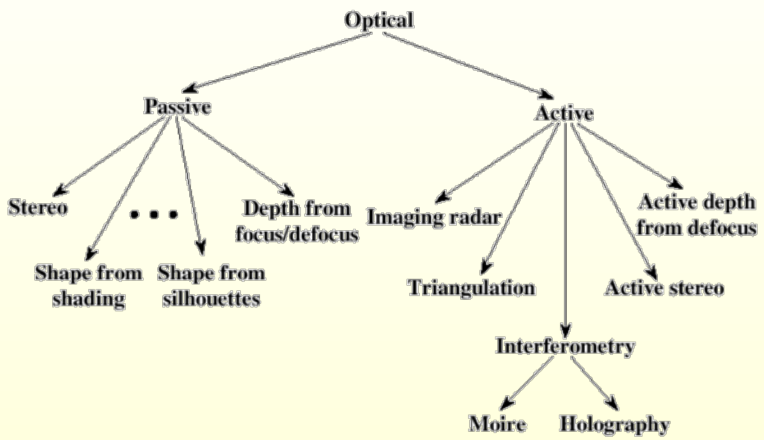
Computer Vision Based Techniques

Depth from blur:

Can approximate depth by detecting how blurry part of the image is for **known focal length.**

Multitude of other methods



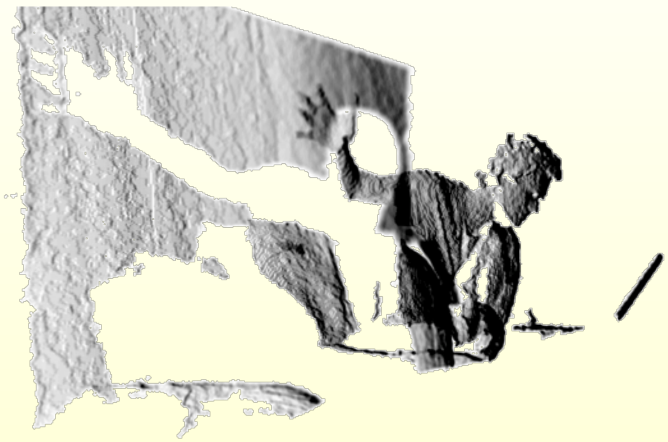


Rocchini et al. '01

Non-exhaustive taxonomy of 3D acquisition methods.

Microsoft Kinect Scanner

Low-cost ($200) 3D scanner – gadget for Xbox.

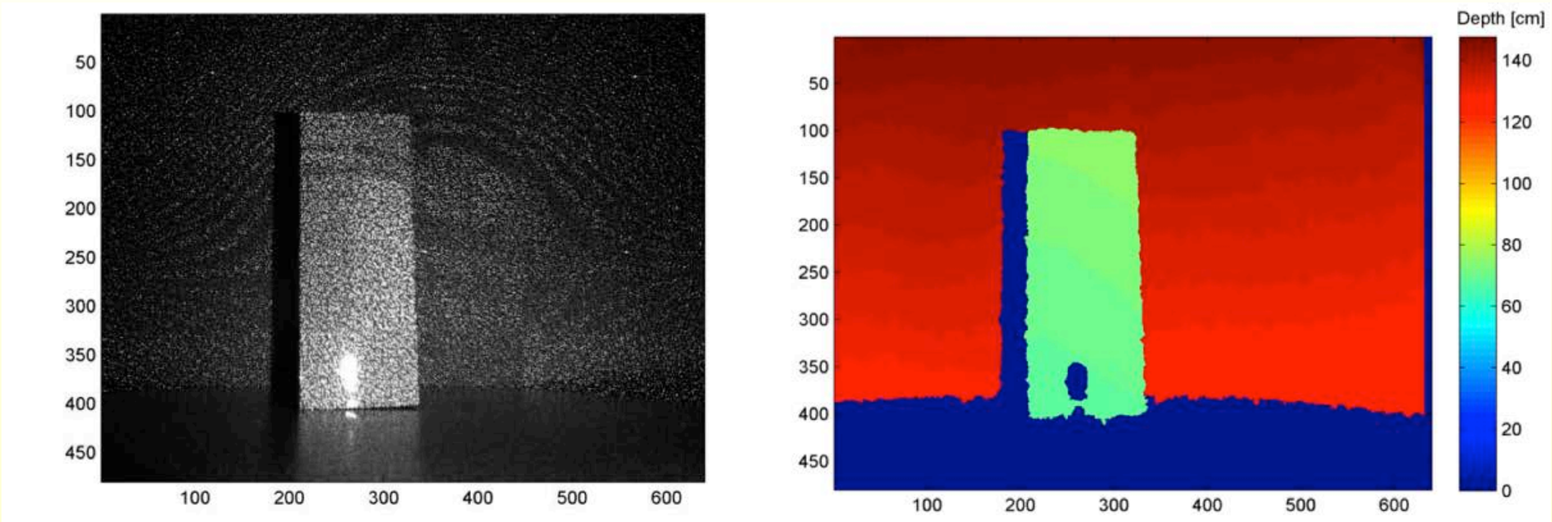


Allows to acquire Image (640 x 480) and 3D geometry (300k points) at 30 FPS.

Uses infrared active illumination with an infrared sensor **and** depth-from blur.
accuracy of ~1mm (at 0.5m distance) to 4cm (at 2m distance).

Microsoft Kinect Scanner

Low-cost ($200) 3D scanner – gadget for Xbox.



Allows us to acquire Image (640 x 480) and 3D geometry (300k points) at 30 FPS.

Uses infrared active illumination with an infrared sensor **and** depth-from blur. accuracy of ~1mm (at 0.5m distance) to 4cm (at 2m distance).

Affordable 3D Scanners

Microsoft Kinect

Google Tango

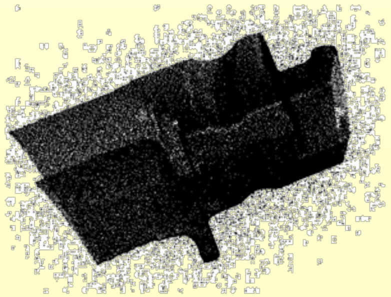
Intel RealSense

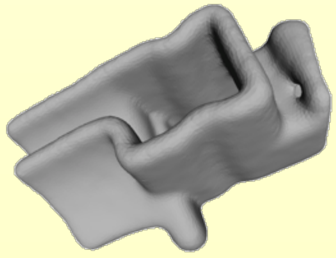
iSense 3D for iPad

3D Point Cloud Processing

Typically point cloud sampling of a shape is insufficient for most applications. Main stages in processing:

1. Outlier removal – throw away samples from non-surface areas
2. If we have multiple scans, align them
3. Smoothing – remove local noise
4. Estimate surface normals
5. Surface reconstruction
    - Implicit representation
    - Triangle mesh extraction

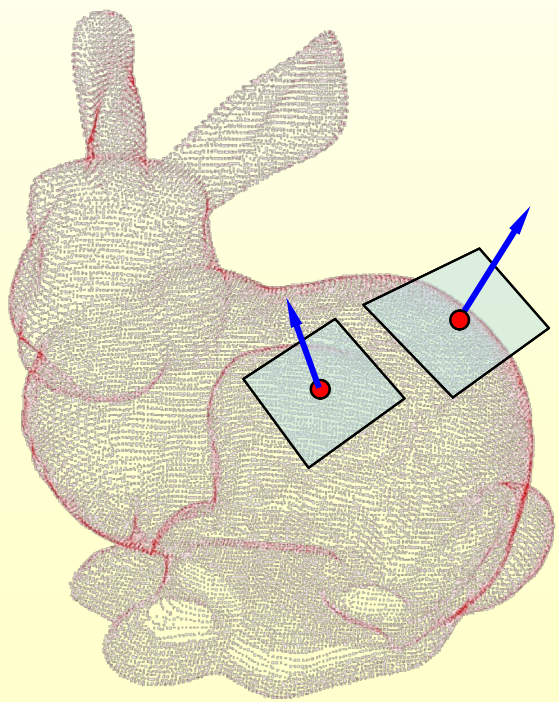




Normal Estimation and Outlier Removal

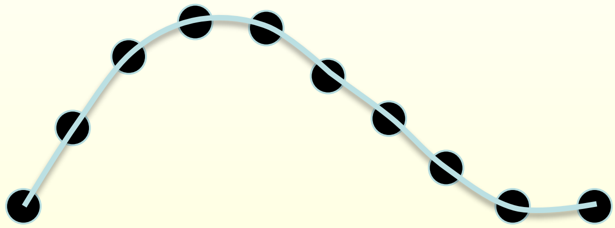
Fundamental problems in point cloud processing.

Although seemingly very different, can be solved with the same general approach – look at the “shape of neighborhoods” …



Normal Estimation

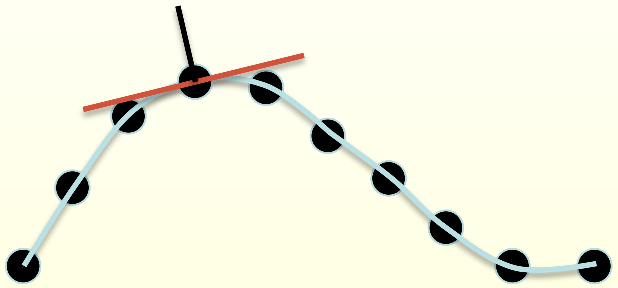
Assume we have a clean sampling of the surface. OK, start with a curve.



Our goal is to find the best approximation of the tangent direction, and thus of the normal to the curve.

Normal Estimation

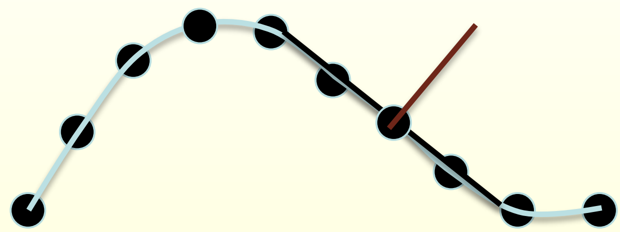
Assume we have a clean sampling of the surface. OK, start with a curve.



Our goal is to find the best approximation of the tangent direction, and thus of the normal to the line.

Normal Estimation

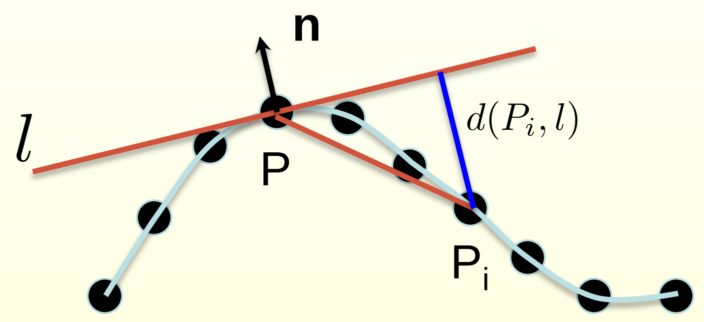
Assume we have a clean sampling of the surface. OK, start with a curve.



Our goal is to find the best approximation of the tangent direction, and thus of the normal to the line.

Normal Estimation

Assume we have a clean sampling of the surface. OK, start with a curve.



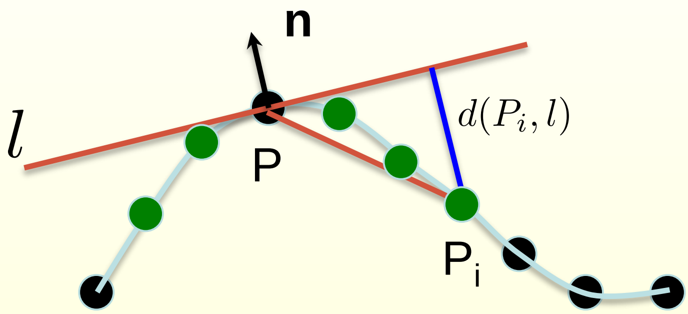
Goal: find best approximation of the normal at P.

Method: Given line l through P with normal **n**, for another point p_i:

$$d(p_i, l)^2 = \frac{((p_i - P)^T \mathbf{n})^2}{\mathbf{n}^T \mathbf{n}} = ((p_i - P)^T \mathbf{n})^2 \text{ if } \|\mathbf{n}\| = 1$$

Normal Estimation

Assume we have a clean sampling of the surface. OK, start with a curve.



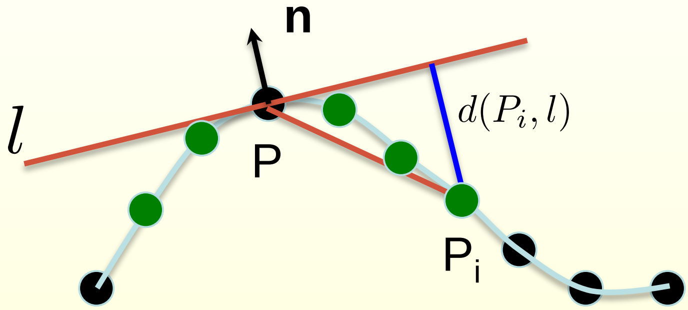
Goal: find best approximation of the normal at P.

Method: Find **n**, minimizing $\sum_{i=1}^{k} d(p_i, l)^2$ for a set of *k* points (e.g. *k* nearest neighbors of P.

$$\mathbf{n}_{\text{opt}} = \arg \min_{\|\mathbf{n}\|=1} \sum_{i=1}^{k} ((p_i - P)^T \mathbf{n})^2$$

Normal Estimation

Assume we have a clean sampling of the surface. OK, start with a curve.



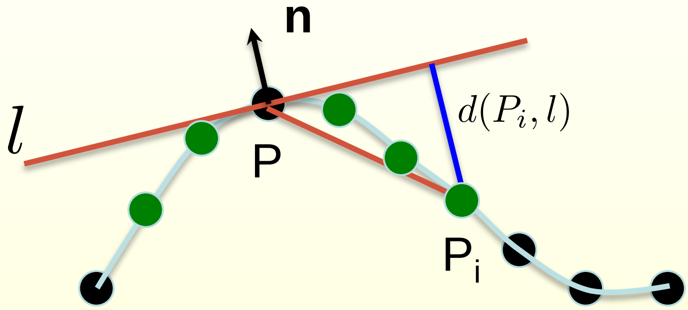
Using Lagrange multiplier:

$$\frac{\partial}{\partial \mathbf{n}}\left(\sum_{i=1}^{k}((p_i - P)^T \mathbf{n})^2\right) - \lambda \frac{\partial}{\partial \mathbf{n}}\left(\mathbf{n}^T \mathbf{n}\right) = 0$$

$$\sum_{i=1}^{k} 2(p_i - P)(p_i - P)^T \mathbf{n} = 2\lambda \mathbf{n}$$

Normal Estimation

Assume we have a clean sampling of the surface. OK, start with a curve.



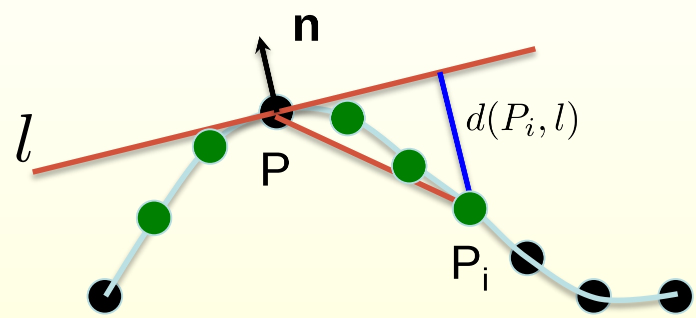
Using Lagrange multiplier:

$$\frac{\partial}{\partial \mathbf{n}}\left(\sum_{i=1}^{k}((p_i - P)^T\mathbf{n})^2\right) - \lambda\frac{\partial}{\partial \mathbf{n}}\left(\mathbf{n}^T\mathbf{n}\right) = 0$$

$$\left(\sum_{i=1}^{k}(p_i - P)(p_i - P)^T\right)\mathbf{n} = \lambda\mathbf{n} \quad \Rightarrow \quad C\mathbf{n} = \lambda\mathbf{n}$$

Normal Estimation

Assume we have a clean sampling of the surface. OK, start with a curve.



The normal **n** must be an eigenvector of the matrix:

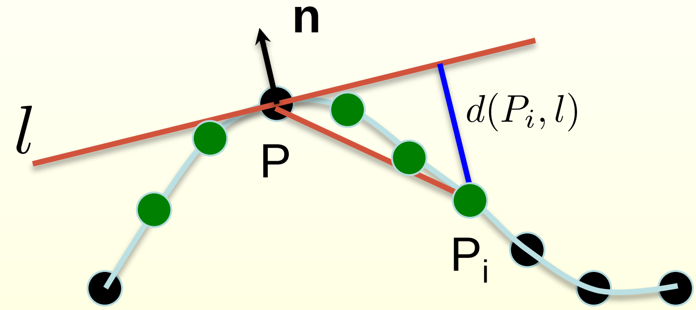
$$C\mathbf{n} = \lambda\mathbf{n} \qquad C = \sum_{i=1}^{k}(p_i - P)(p_i - P)^T$$

Moreover, since:

$$\mathbf{n}_{\mathrm{opt}} = \arg\min_{\|\mathbf{n}\|=1} \sum_{i=1}^{k}((p_i - P)^T\mathbf{n})^2 = \arg\min_{\|\mathbf{n}\|=1} \mathbf{n}^T C\mathbf{n}$$

Normal Estimation

Assume we have a clean sampling of the surface. OK, start with a curve.



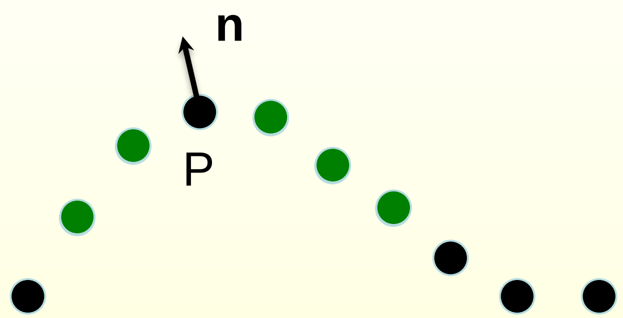
The normal **n** must be an eigenvector of the matrix:

$$C\mathbf{n} = \lambda\mathbf{n} \qquad C = \sum_{i=1}^{k}(p_i - P)(p_i - P)^T$$

Moreover, $\mathbf{n}_{opt}$ must be the eigenvector corresponding to the **smallest eigenvalue** of C.

Normal Estimation

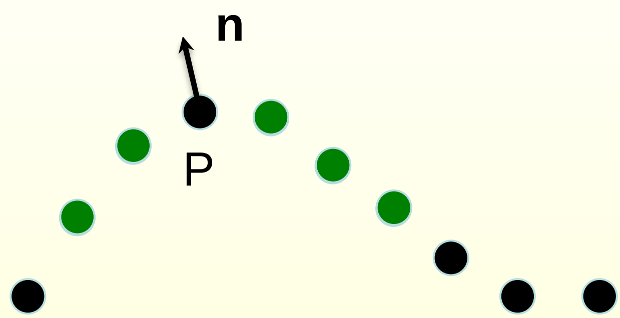
Method Outline (PCA):



1. Given a point P in the point cloud, find its *k* nearest neighbors.
2. Compute $C = \sum_{i=1}^{k} (p_i - P)(p_i - P)^T$
3. **n:** eigenvector corresponding to the smallest eigenvalue of *C*.

Normal Estimation

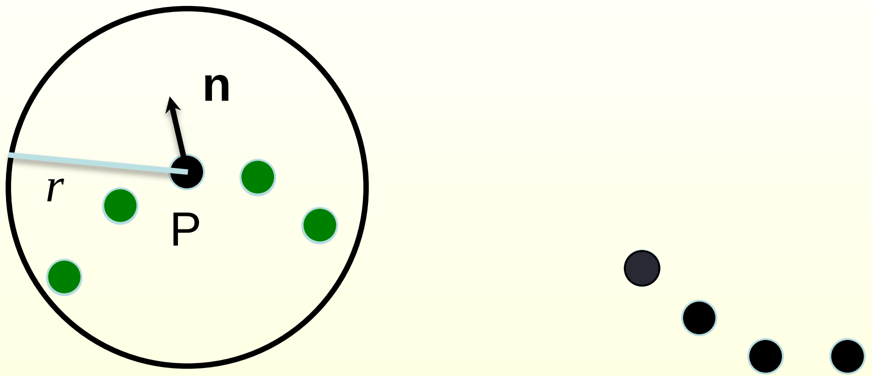
Method Outline (PCA):



1. Given a point P in the point cloud, find its *k* nearest neighbors.
2. Compute $C = \sum_{i=1}^{k} (p_i - P)(p_i - P)^T$
3. **n:** eigenvector corresponding to the smallest eigenvalue of *C*.

Variant on the theme: use $C = \sum_{i=1}^{k} (p_i - \overline{P})(p_i - \overline{P})^T, \quad \overline{P} = \frac{1}{k}\sum_{i=1}^{k} p_i$

Normal Estimation

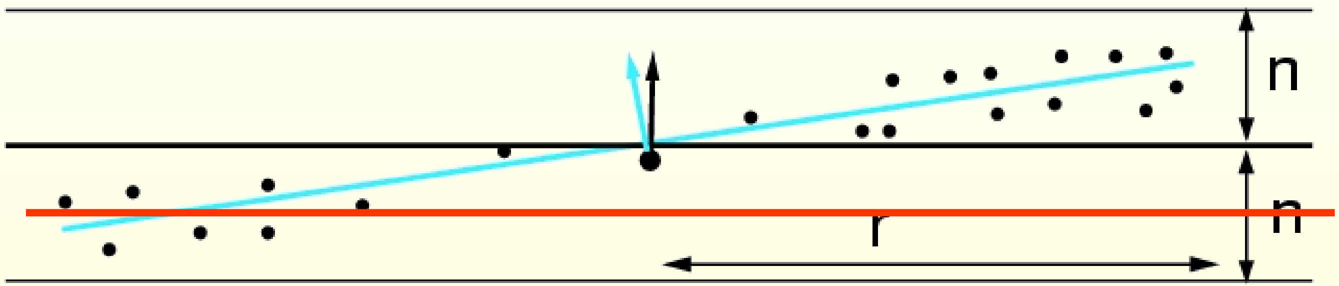


Critical parameter: *k*. Because of uneven sampling typically fix a radius *r*, and use all points **inside a ball of radius *r*.**

How to pick an optimal r?

Normal Estimation

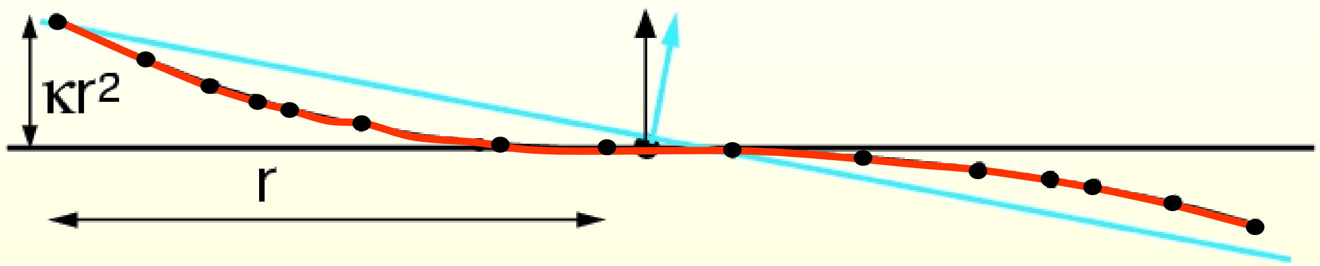
Collusive noise



Because of noise in the data, small *r* may lead to underfitting.

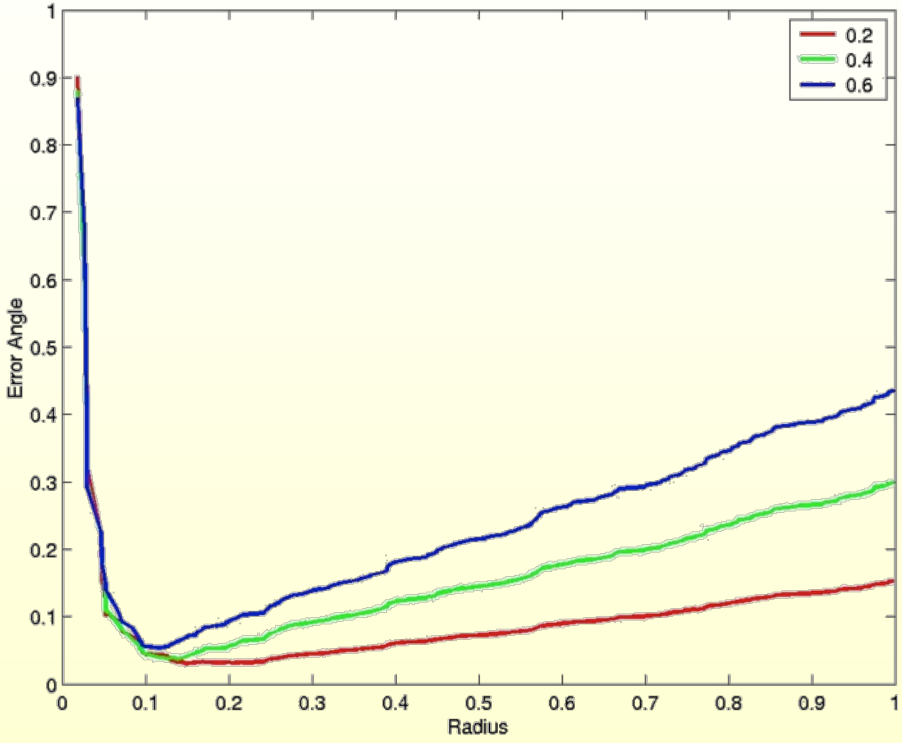
Normal Estimation

Curvature effect



Due to curvature, large *r* can lead to estimation bias.

Normal Estimation



source: Mitra et al. ‘04

$$error \leq \Theta(1)\kappa r + \Theta(1)\frac{\sigma_n}{\sqrt{\varepsilon\rho r^3}} + \Theta(1)\frac{\sigma_n^2}{r^2}$$

Estimation error under Gaussian noise for different values of curvature (2D)

Normal Estimation

A similar but involved analysis results in 3D,

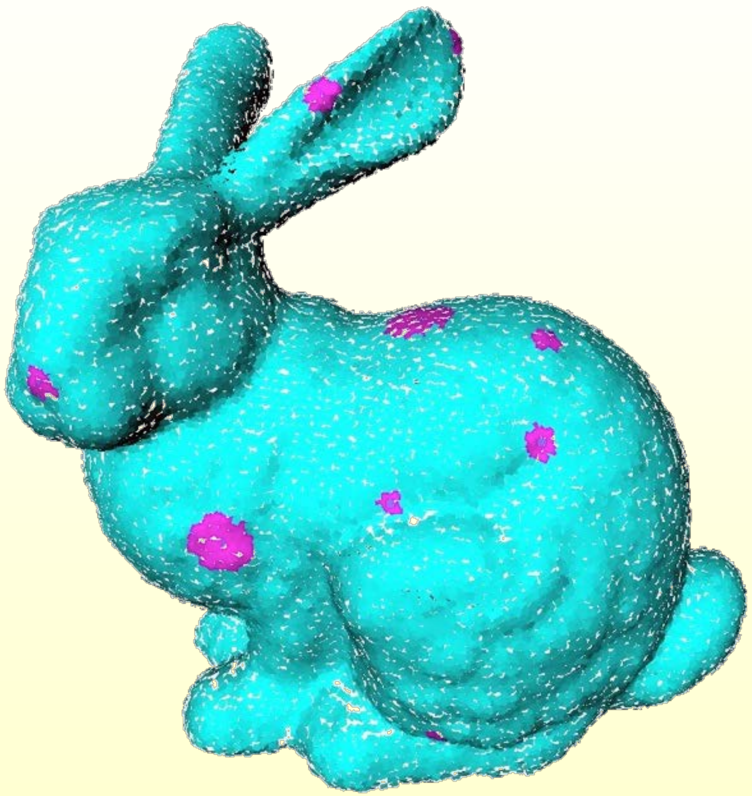
$$error \leq \Theta(1)\kappa r + \Theta(1)\sigma_n / (r^2 \sqrt{\varepsilon\rho}) + \Theta(1)\sigma_n^2 / r^2$$

A good choice of *r* is,

$$r = \left(\frac{1}{\kappa} \left(c_1 \frac{\sigma_n}{\sqrt{\varepsilon\rho}} + c_2 \sigma_n^2 \right) \right)^{1/3}$$

source: Mitra et al. '04

Normal Estimation – Neighborhood Size

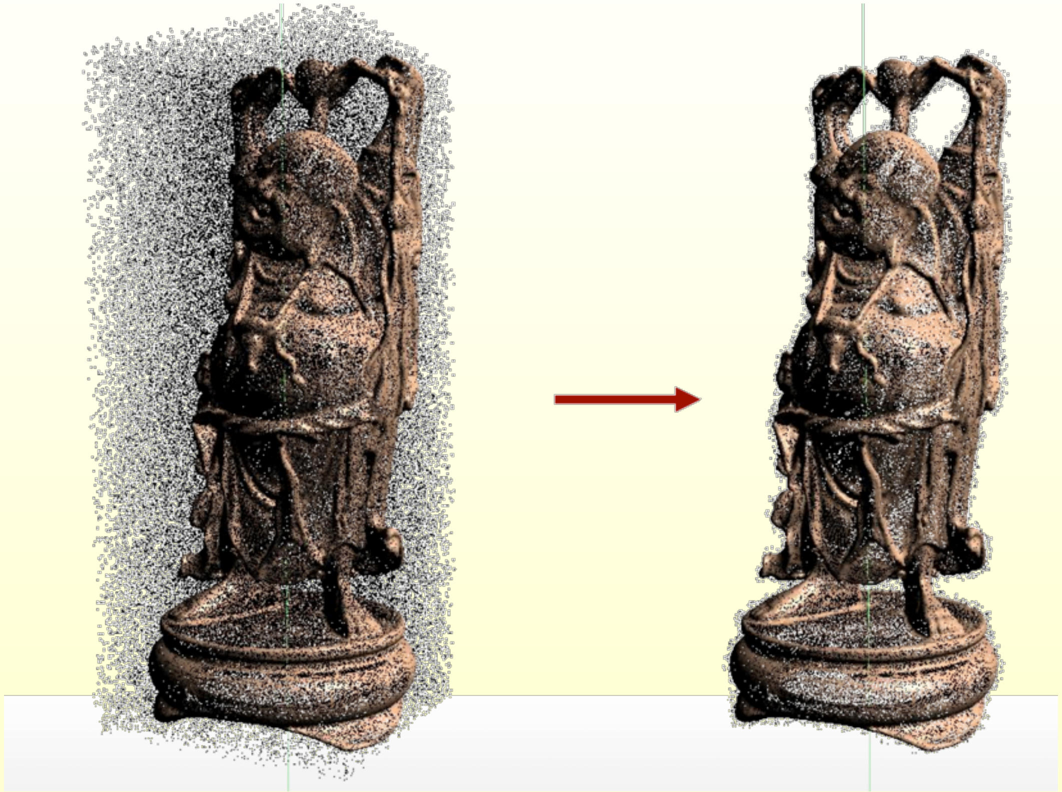


1x noise

2x noise

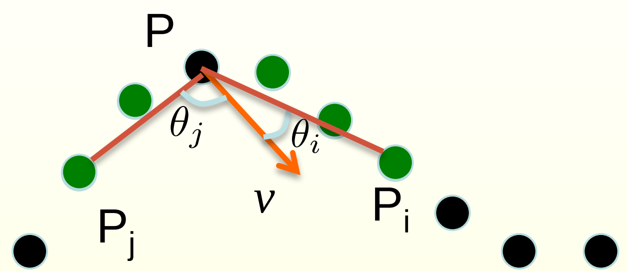
source: Mitra et al. ‘04

Outlier Removal



Goal: remove points that do not lie close to a surface.

Outlier Estimation



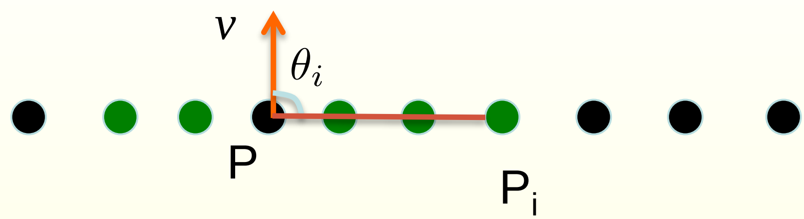
From the covariance matrix: $C = \sum_{i=1}^{k} (p_i - P)(p_i - P)^T$ we have:

for any vector v, the Rayleigh quotient:

$$\frac{v^T C v}{v^T v} = \sum_{i=1}^{k} \left((p_i - P)^T v \right)^2 \text{ if } \|v\| = 1$$
$$= \sum_{i=1}^{k} \left(\|p_i - P\| \cos(\theta_i) \right)^2$$

Intuitively, v_{min}, maximizes the sum of angles to each vector $(p_i - P)$.

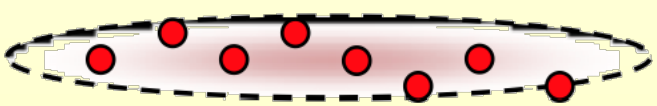
Outlier Estimation



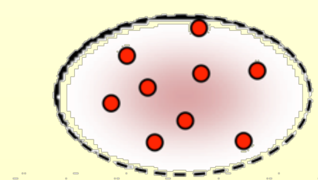
If all the points are on a line, then $\min_v \frac{v^T C v}{v^T v} = \lambda_{\min}(C) = 0$ and $\lambda_{\max}(C)$ is large.

There exists a direction along which the point cloud has no variability.

If points are scattered randomly, then: $\lambda_{\max}(C) \approx \lambda_{\min}(C)$.

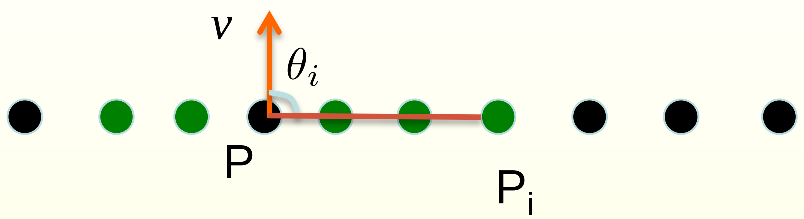


$\frac{\lambda_1}{\lambda_2}$ small



$$\frac{\lambda_1}{\lambda_2} \approx 1$$

Outlier Estimation



If all the points are on a line, then $\min_v \frac{v^T C v}{v^T v} = \lambda_{\min}(C) = 0$ and $\lambda_{\max}(C)$ is large.

There exists a direction along which the point cloud has no variability.

If points are scattered randomly, then: $\lambda_{\max}(C) \approx \lambda_{\min}(C)$.

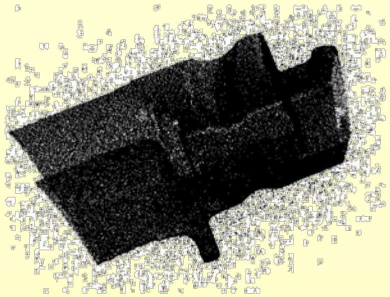
Thus, can remove points where $\frac{\lambda_1}{\lambda_2} > \epsilon$ for some threshold.

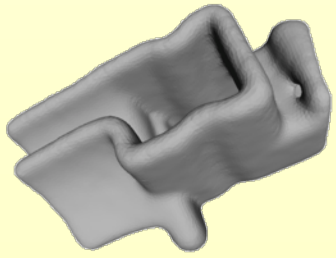
In 3D we expect two zero eigenvalues, so use $\frac{\lambda_2}{\lambda_3} > \epsilon$ for some threshold.

3D Point Cloud Processing

Typically point cloud sampling of a shape is insufficient for most applications. Main stages in processing:

1. Outlier removal – throw away samples from non-surface areas
2. If we have multiple scans, align them
3. Smoothing – remove local noise
4. Estimate surface normals
5. Surface reconstruction
   - Implicit representation
   - Triangle mesh extraction

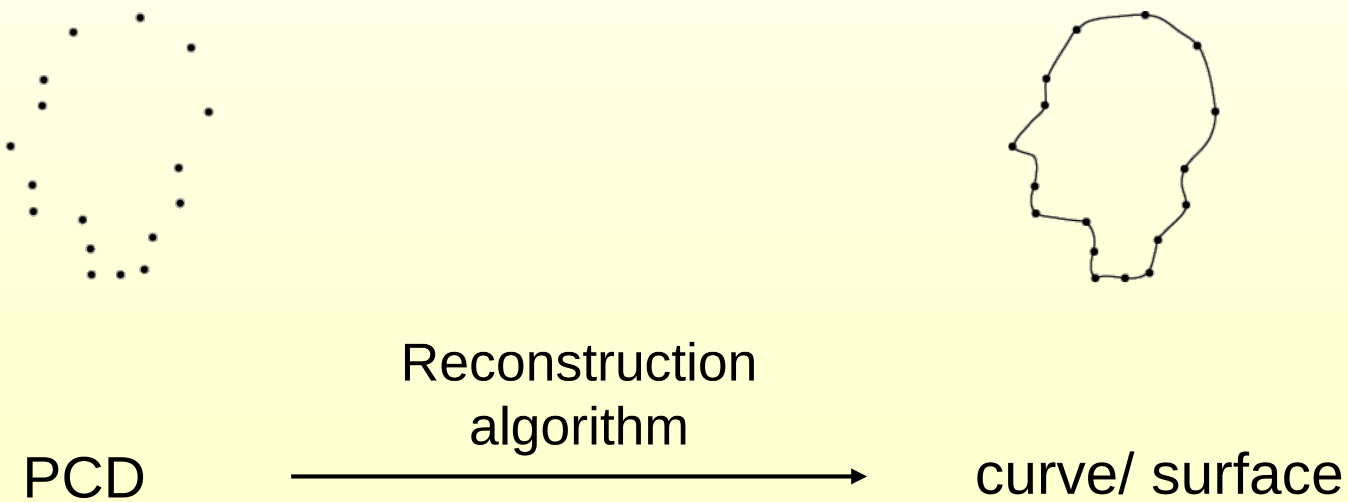




3D Point Cloud Reconstruction

Main Goal:

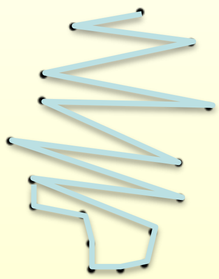
Construct a polygonal (e.g. triangle mesh) representation of the point cloud.



3D Point Cloud Reconstruction

Main Problem:

Data is **unstructured.** E.g. in 2D the points are not **ordered.**



PCD

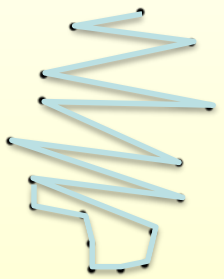
Reconstruction algorithm

curve/ surface

3D Point Cloud Reconstruction

Main Problem:

Data is **unstructured.** E.g. in 2D the points are not **ordered.**
Inherently **ill-posed** (aka difficult) problem.



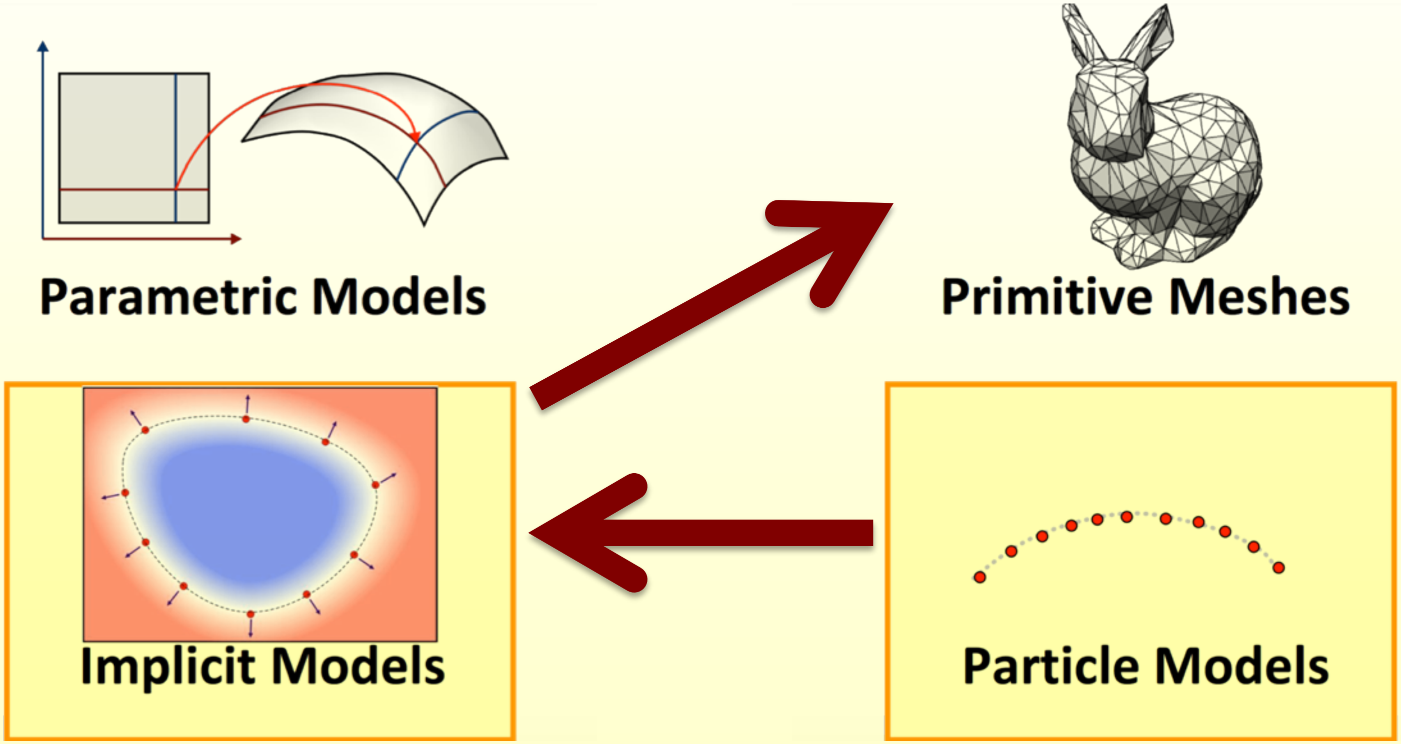
PCD

Reconstruction algorithm

curve/ surface

3D Point Cloud Reconstruction

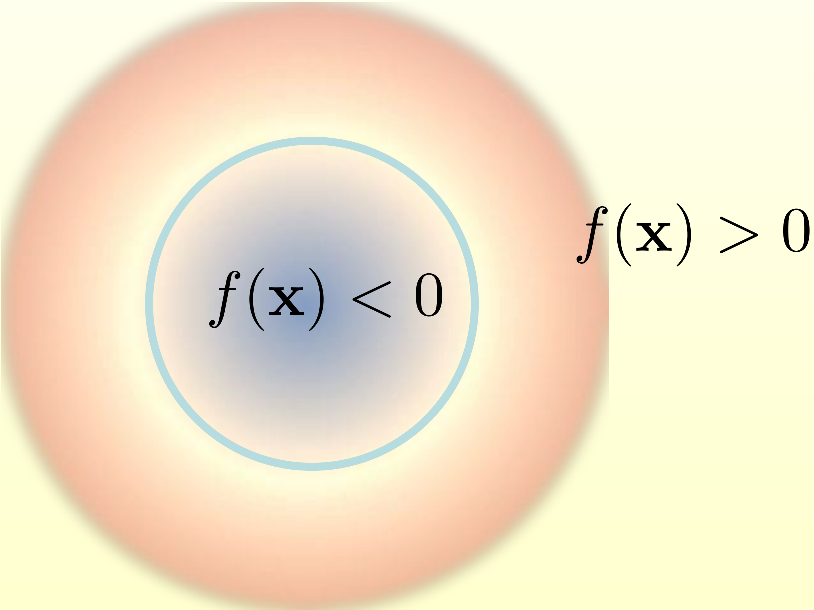
Reconstruction through Implicit models.



Implicit Surfaces

Given a function f(**x**), the surface is defined as:

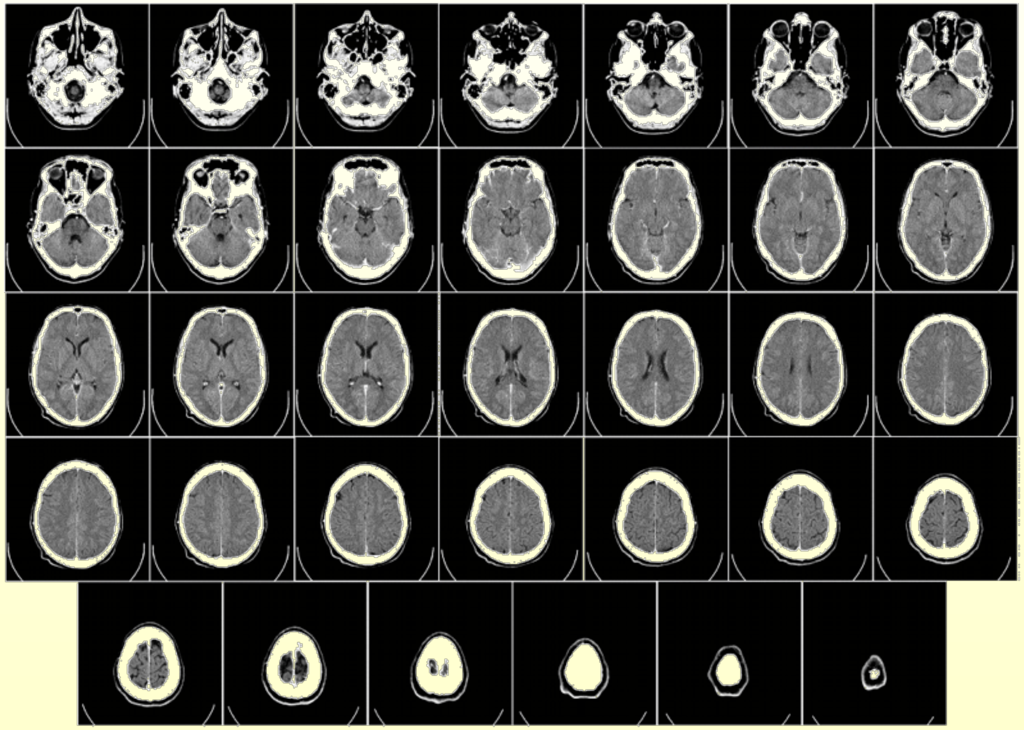
$$\{\mathbf{x}, \mathrm{s.t.} f(\mathbf{x}) = 0\}$$



$$f(x, y) = x^2 + y^2 - r^2$$

Implicit Surfaces

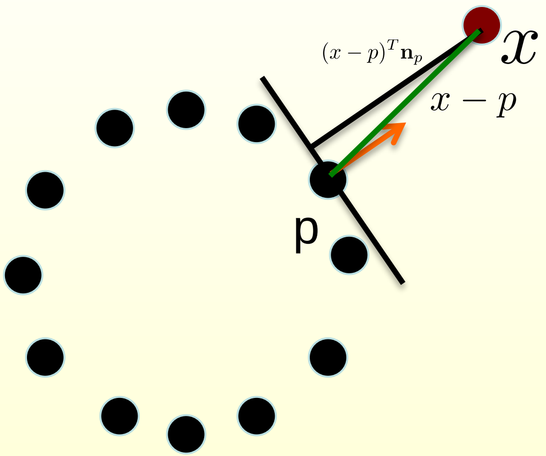
Some 3D scanning technologies (e.g. CT, MRI) naturally produce implicit representations



CT scans of a human brain

Implicit Surfaces

Converting from a point cloud to an implicit surface:



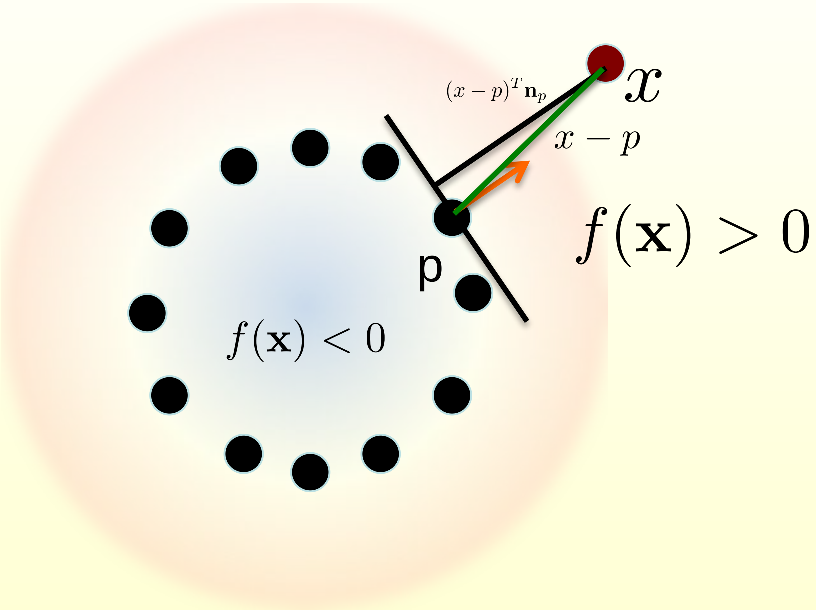
Simplest method:

1. Given a point x in space, find nearest point p in PCD.
2. Set $f(x) = (x-p)^T \mathbf{n}_p$ – signed distance to the tangent plane.

Hugues Hoppe: Surface reconstruction from unorganized points

Implicit Surfaces

Converting from a point cloud to an implicit surface:

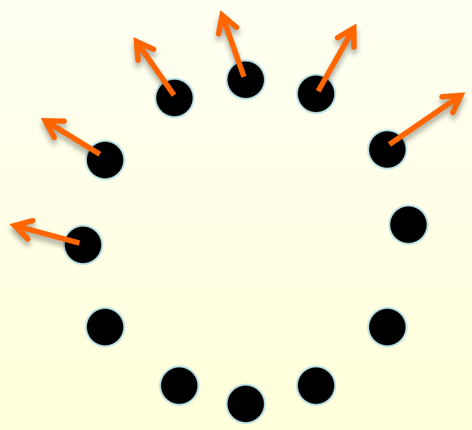


Simplest method:

1. Given a point x in space, find nearest point p in PCD.
2. Set $f(x) = (x-p)^T\mathbf{n}_p$ – signed distance to the tangent plane.

Implicit Surfaces

Converting from a point cloud to an implicit surface:



Simplest method:

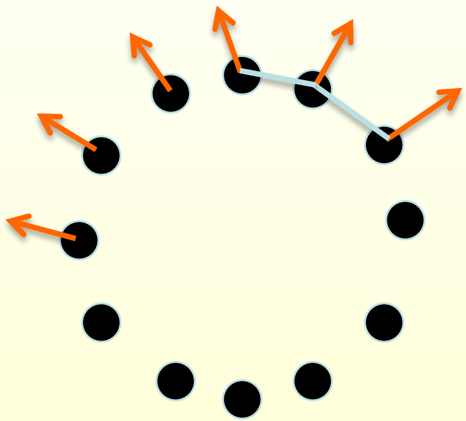
1. Given a point x in space, find nearest point p in PCD.
2. Set $f(x) = (x - p)^T \mathbf{n}_p$ – signed distance to the tangent plane.
3. Note: need consistently oriented normals.

    PCA only gives normals **up to orientation**

Hugues Hoppe: Surface reconstruction from unorganized points

Implicit Surfaces

Converting from a point cloud to an implicit surface:



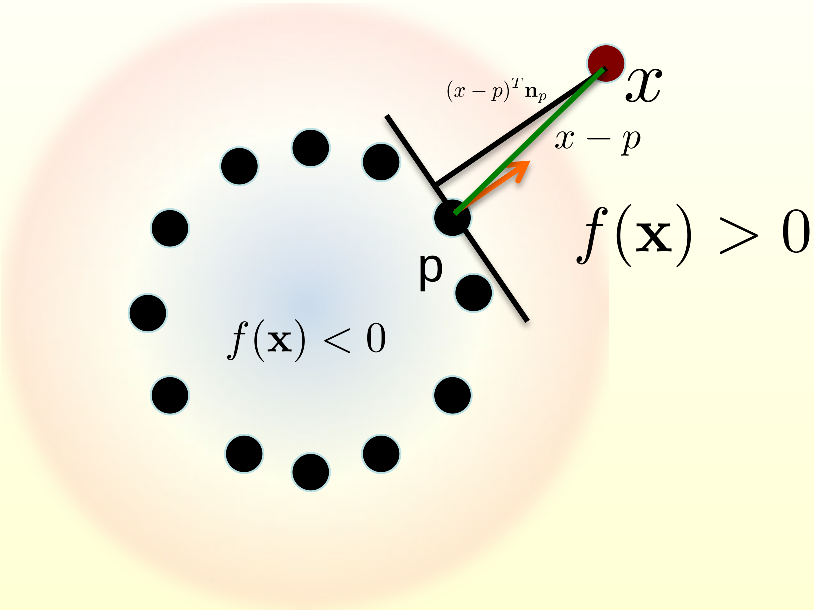
Simplest method:

1. Given a point x in space, find nearest point p in PCD.
2. Set $f(x) = (x - p)^T \mathbf{n}_p$ – signed distance to the tangent plane.
3. Note: need consistently oriented normals. In general, difficult problem, but can try to locally connect points and fix orientations.

Hugues Hoppe: Surface reconstruction from unorganized points

Implicit Surfaces

Converting from a point cloud to an implicit surface:



Simplest method:

1. Given a point x in space, find nearest point p in PCD.
2. Set $f(x) = (x-p)^T\mathbf{n}_p$ – signed distance to the tangent plane.

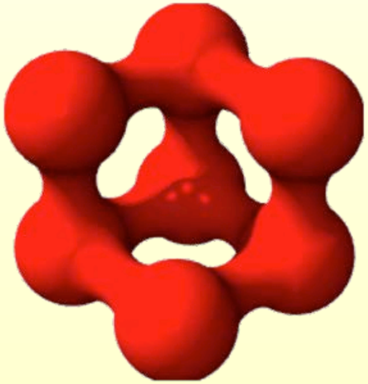
Note: many more advanced methods exist:
e.g. Moving Least Squares (MLS)

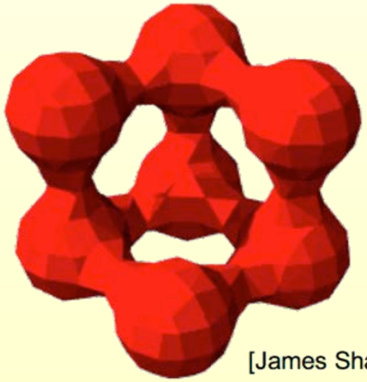
Marching Cubes

Converting from implicit to explicit representations.

Goal: Given an implicit representation: $\{\mathbf{x}, \mathrm{s.t.}\, f(\mathbf{x}) = 0\}$

Create a triangle mesh that approximates the surface.





[James Sharman]

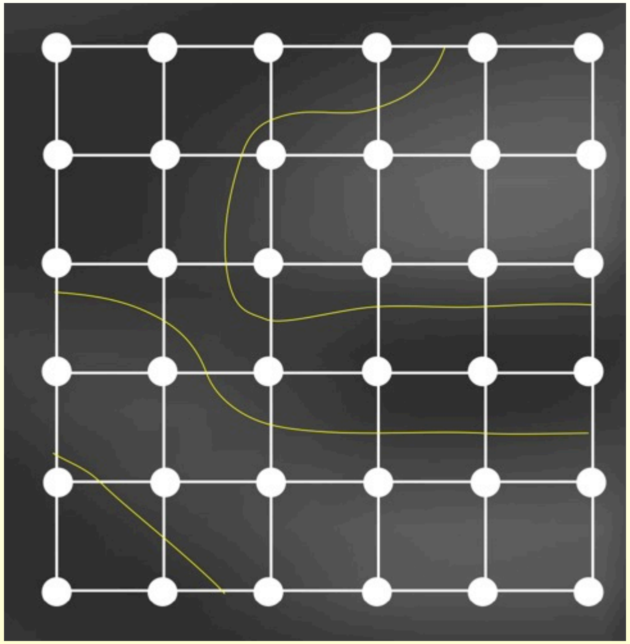
Lorensen and Cline, SIGGRAPH '87

One of the most cited computer graphics papers of all time.

Marching Squares (2D)

Given a function: $f(x)$

- $f(\mathbf{x}) < 0$ inside
- $f(\mathbf{x}) > 0$ outside

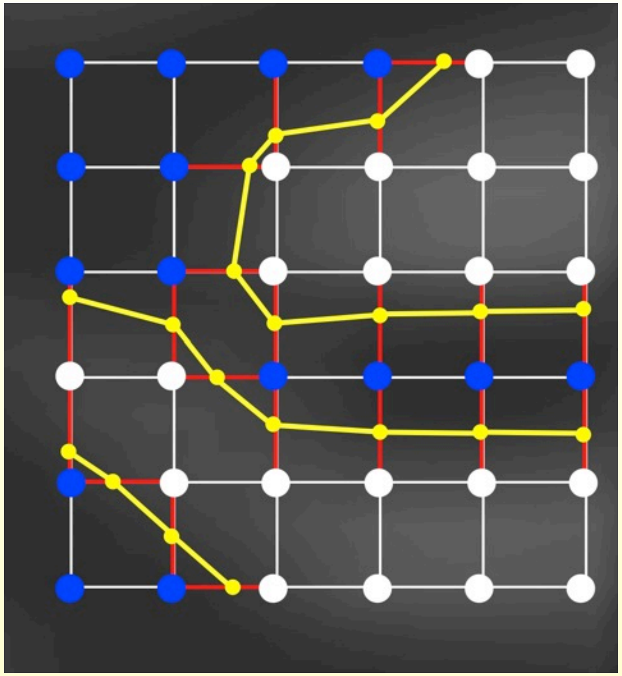


1. Discretize space.
2. Evaluate $f(x)$ on a grid.

Marching Squares (2D)

Given a function: $f(x)$

- $f(\mathbf{x}) < 0$ inside
- $f(\mathbf{x}) > 0$ outside



1. Discretize space.
2. Evaluate $f(x)$ on a grid.
3. Classify grid points (+/-)
4. Classify grid edges
5. Compute intersections
6. Connect intersections

Marching Squares (2D)

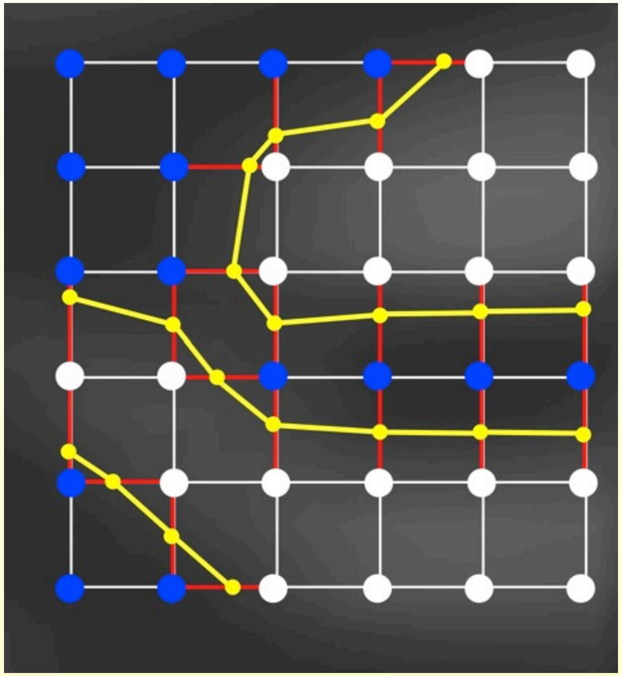
Computing the intersections:

- Edges with a sign switch contain intersections.

$$f(x_1) < 0, f(x_2) > 0 \Rightarrow$$
$$f(x_1 + t(x_2 - x_1)) = 0$$
$$\text{for some } 0 \le t \le 1$$

- Simplest way to compute t: assume f is linear between x1 and x2:

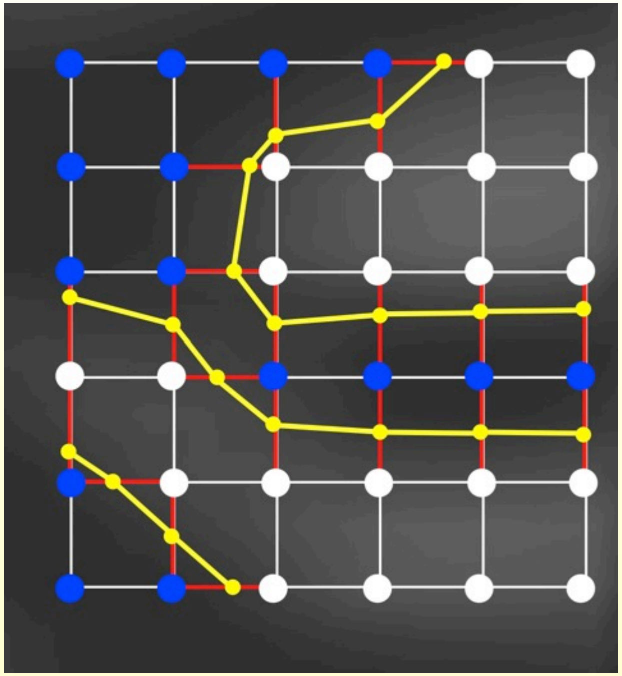
$$t = \frac{f(x_1)}{f(x_2) - f(x_1)}$$



Marching Squares (2D)

Connecting the intersections:

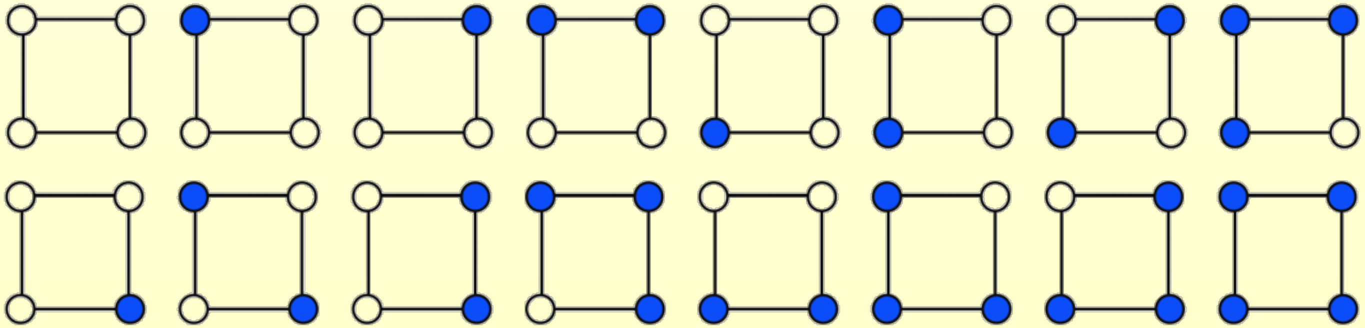
- Grand principle: treat each cell separately!
- Enumerate all possible inside/outside combinations.



Marching Squares (2D)

Connecting the intersections:

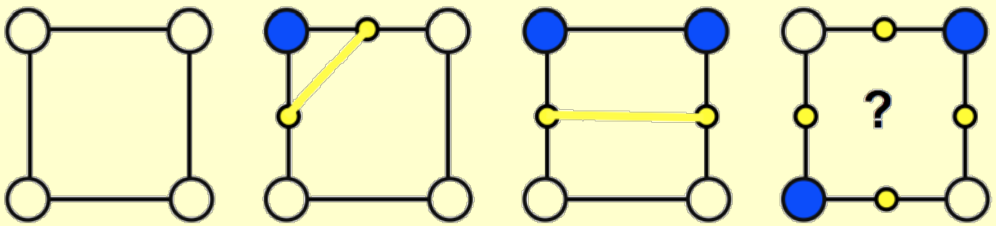
- Grand principle: treat each cell separately!
- Enumerate all possible inside/outside combinations.
- Group those leading to the same intersections



Marching Squares (2D)

Connecting the intersections:

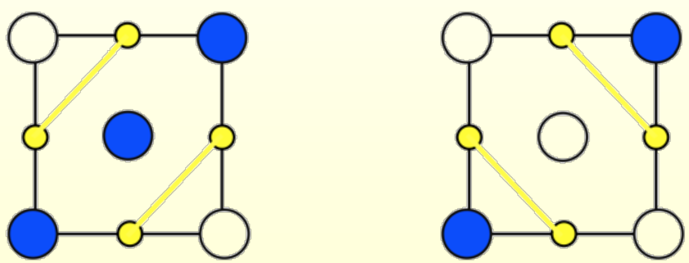
- Grand principle: treat each cell separately!
- Enumerate all possible inside/outside combinations.
- Group those leading to the same intersections.
- Group equivalent after rotation.
- Connect intersections

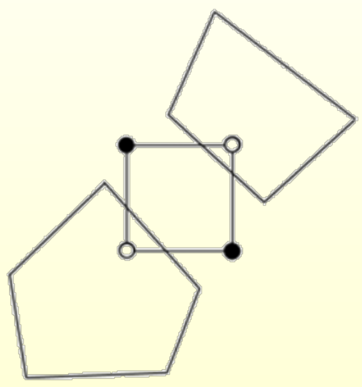


Marching Squares (2D)

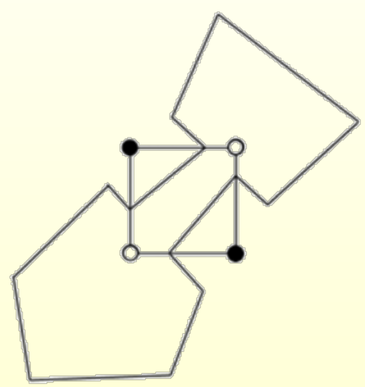
Connecting the intersections:

Ambiguous cases:





Break contour



Join contour

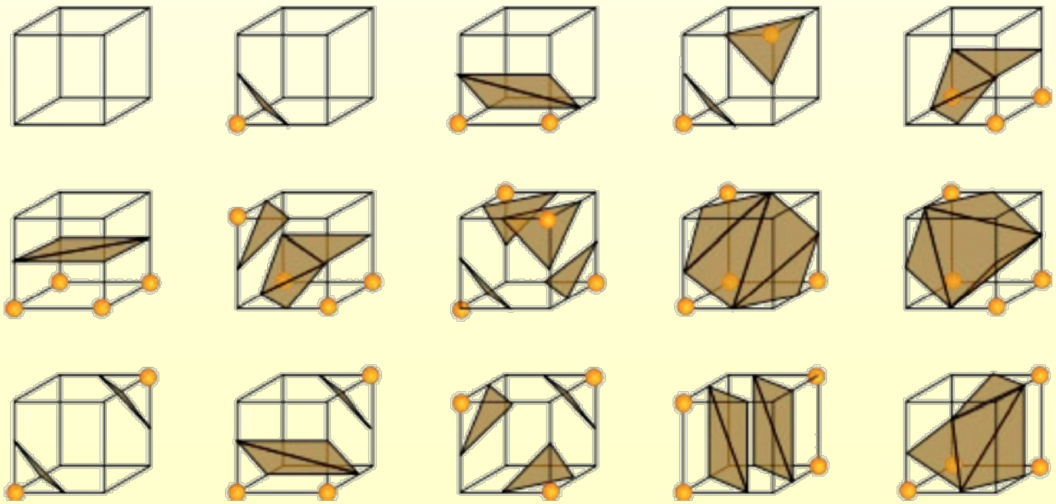
2 options:

1) Can resolve ambiguity by subsampling inside the cell.
2) If subsampling is impossible, pick one of the two possibilities.

Marching Cubes (3D)

Same basic machinery applies to 3D.
cells become **cubes** (voxels)
lines become **triangles**

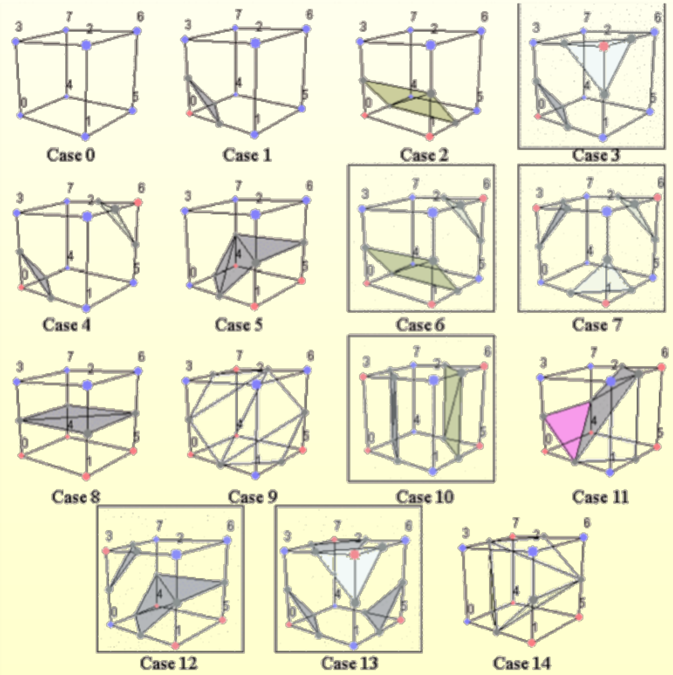
- 256 different cases
- 15 after symmetries



Marching Cubes (3D)

Same basic machinery applies to 3D.
cells become **cubes** (voxels)
lines become **triangles**

- 256 different cases
- 15 after symmetries
- 6 ambiguous cases (in boxes)



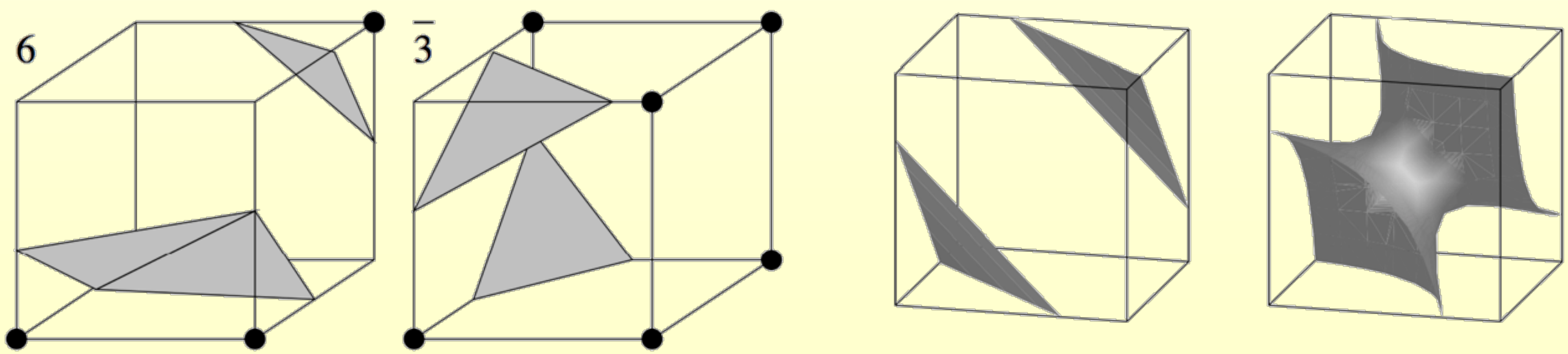
Marching Cubes (3D)

Same basic machinery applies to 3D.

cells become **cubes** (voxels)

lines become **triangles**

- 256 different cases
- 15 after symmetries
- 6 ambiguous cases (in boxes)
- Inconsistent triangulations can lead to holes and wrong topology.



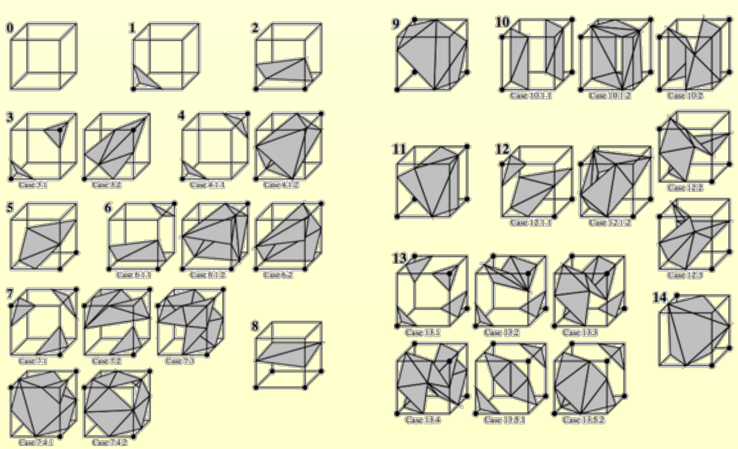
Marching Cubes (3D)

Same basic machinery applies to 3D.

cells become **cubes** (voxels)

lines become **triangles**

- 256 different cases
- 15 after symmetries
- 6 ambiguous cases (in boxes)
- Inconsistent triangulations can lead to holes and wrong topology.
- More subsampling rules – leads to 33 unique cases.



Chernyaev, Marching Cubes 33,'95

Marching Cubes (3D)

Main Strengths:

- Very multi-purpose.
- Extremely fast and parallelizable.
- Relatively simple to implement.

Main Weaknesses:

- Can create badly shaped (skinny) triangles.
- Basic versions do not provide topological guarantees.
- Many special cases (implemented as big lookup tables).

Marching Cubes (3D)

Main Strengths:

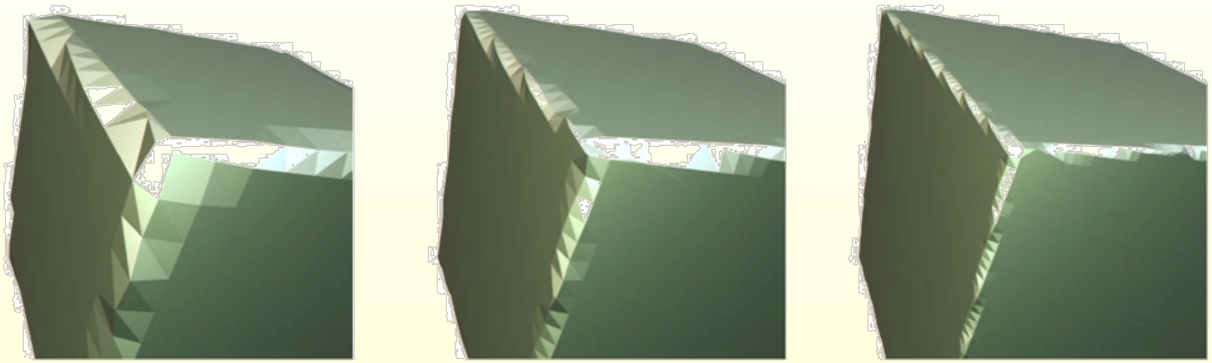
- Very multi-purpose.
- Extremely fast and parallelizable.
- Relatively simple to implement.
- Virtually parameter-free

Main Weaknesses:

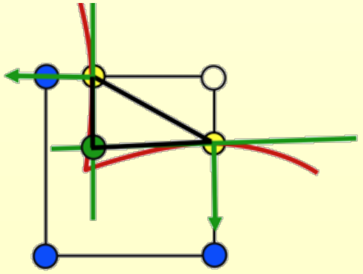
- Can create badly shaped (skinny) triangles.
- Basic versions do not provide topological guarantees.
- Many special cases (implemented as big lookup tables).
- No sharp features.

Marching Cubes (3D)

No sharp features.



1. Increasing grid resolution does not help
2. Normals do not converge.
3. Use normal information to find corners.
   Special treatment for corners



Extended Marching Cubes

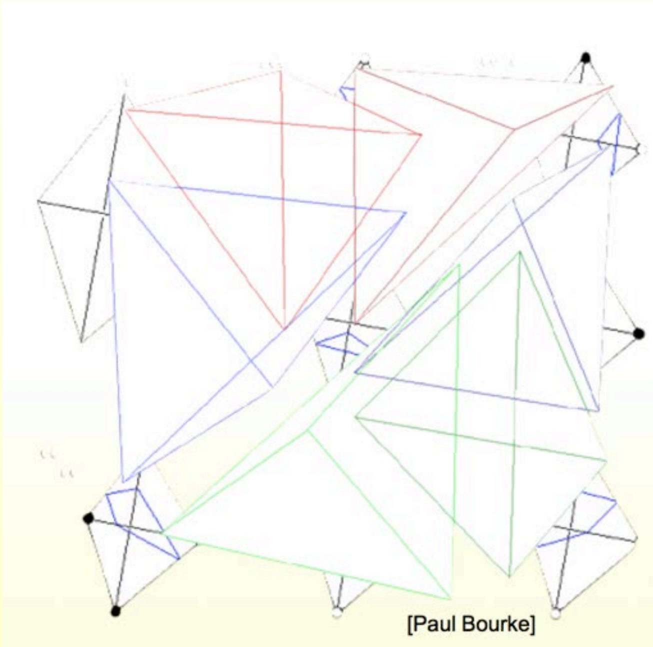
Marching Cubes – Extensions

Marching Tetrahedra.

Instead of cubes (grid) use **Tetrahedra.**

- 6 Tetrahedra per voxel
- 16 cases (8 after symmetry)
- Up to 2 triangles per tet
- **No ambiguities.**

Can be used when input is discretized as tetrahedra.



This slide is based on a slide set courtesy of Maks Ovsjanikov

Conclusions

Wide variety of 3d scanning techniques.

Reconstruction is a difficult, highly data-dependent problem.

Majority of reconstruction methods require normal information.

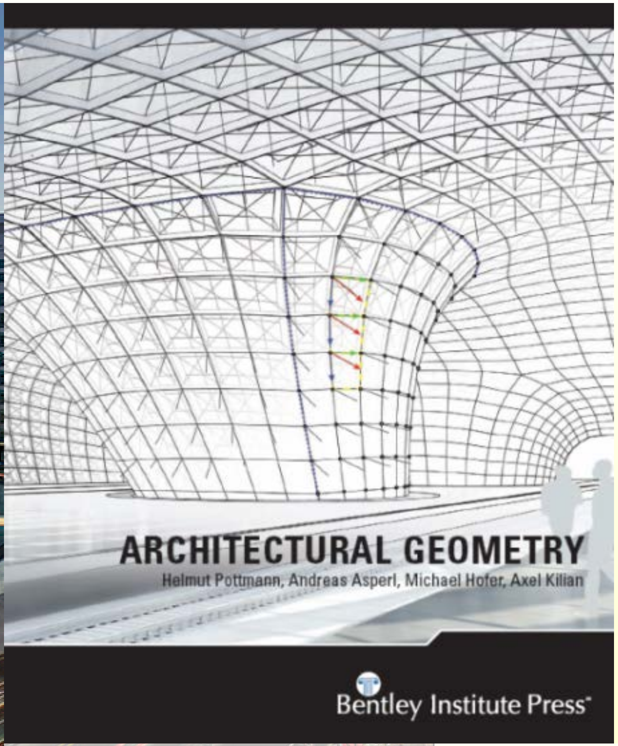
Marching cubes: classical method for converting an implicit to explicit representation.

Many extensions to:

- Improve topology
- Handle sharp features
- Improve quality

This slide is based on a slide set courtesy of Maks Ovsjanikov

Application: Geometry Processing for Freeform Architecture

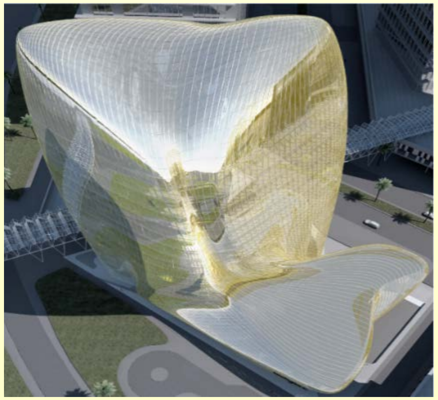


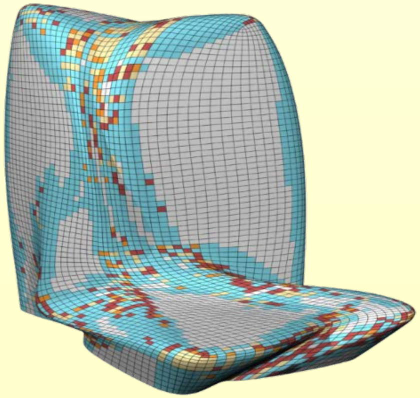
The following slides are based on slides courtesy of Prof. Helmut Pottmann

Dongdaemun Design Plaza, South Korea, by ZHA

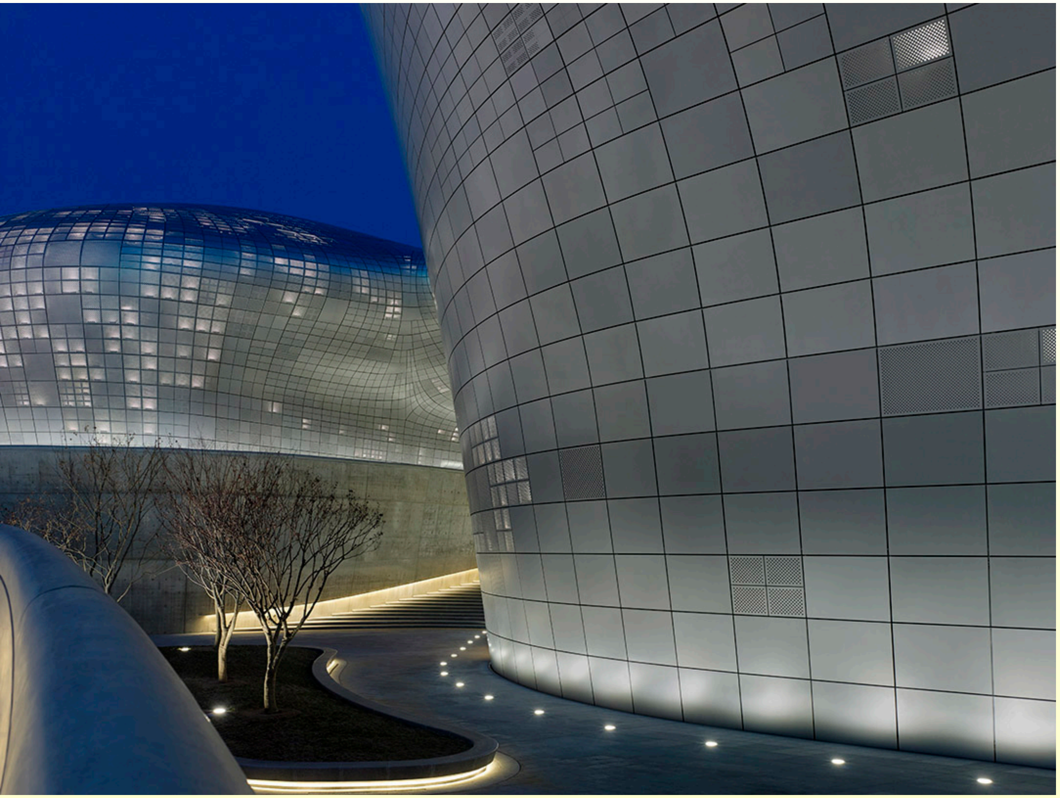
Freeform Architecture

- Large scale, architectural projects, involving complex freeform geometry
- Realization challenging and costly
- Available digital design technology is not adapted to the demands in this area.





Dongdaemun Design Plaza, South Korea, by ZHA



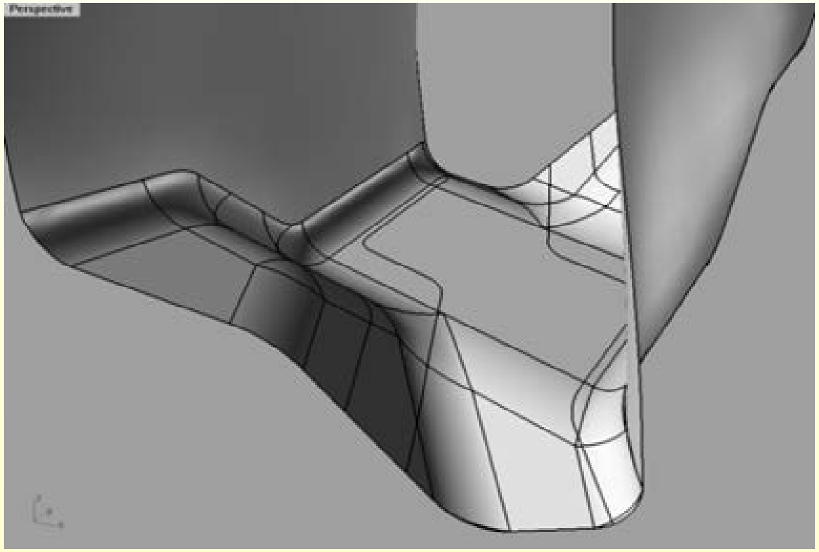
Yas Island Marina Hotel, Abu Dhabi, by Asymptote Architecture

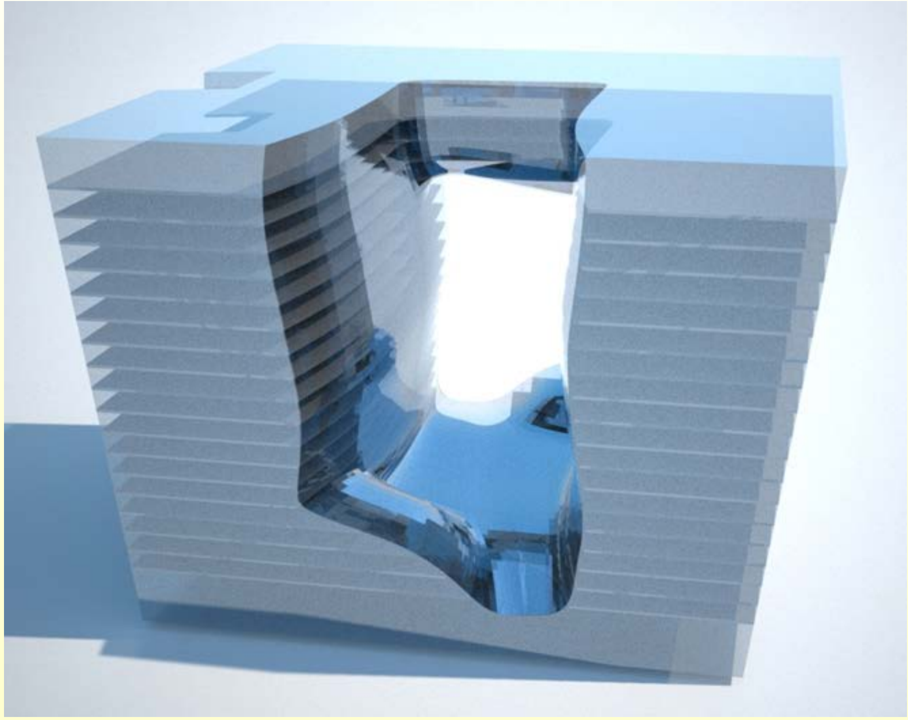
Production

- Digital design followed by digital production. In many industries based on CNC machining of moulds
- Mould fabrication costly, but large numbers of pieces are fabricated with a single mould.
- 3D printing becomes mature, but is usually employed for customized products of small size only

Designing freeform architecture

Uses digital design tools of
CAD/CAM based industries

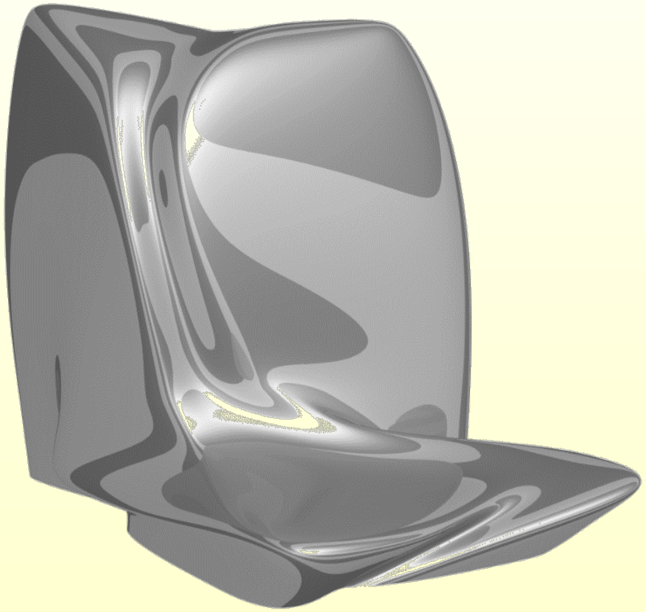


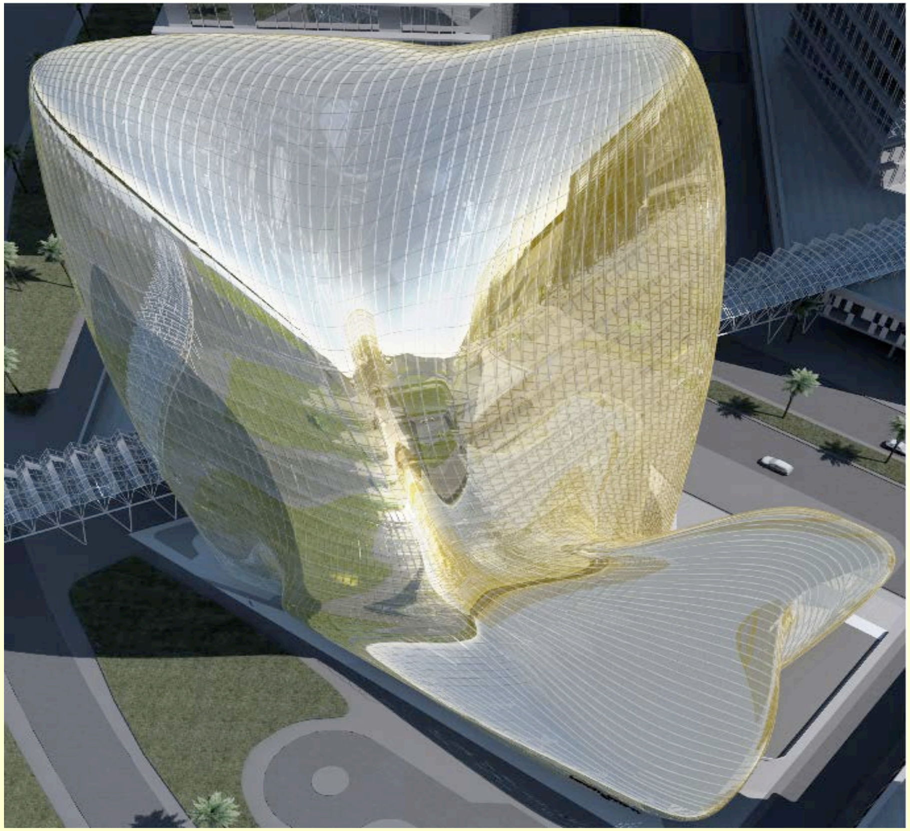


Opus, Zaha Hadid Architects

Producing freeform architecture?

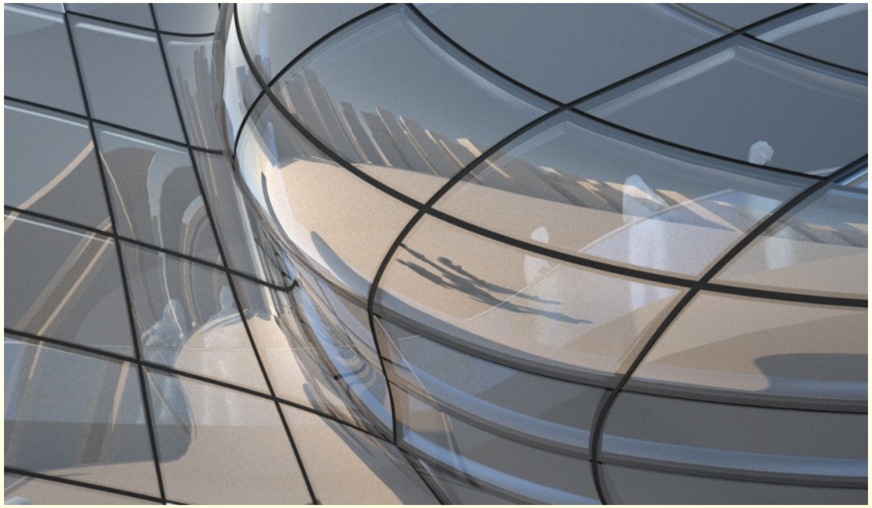
One cannot build such structures in a monolithic way

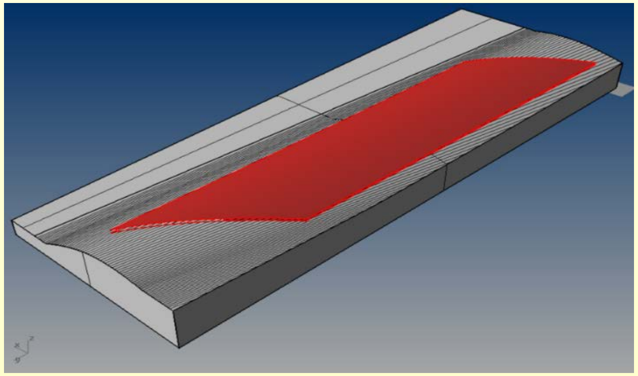




Panels

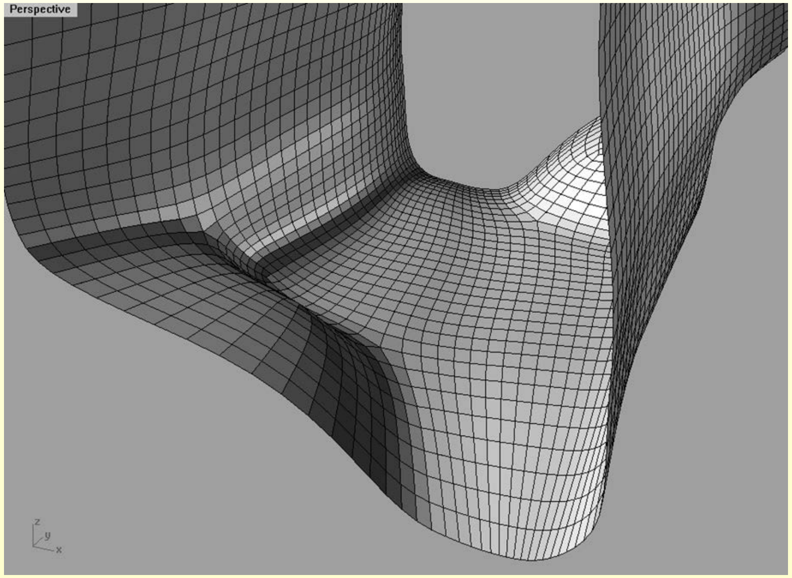
- Skins have to broken up into panels.
- For real freeform geometry, panels are different from each other
- Production of each piece requires a unique machine configuration for manufacturing or even a unique mould.
- Too costly!
- Alternatives?





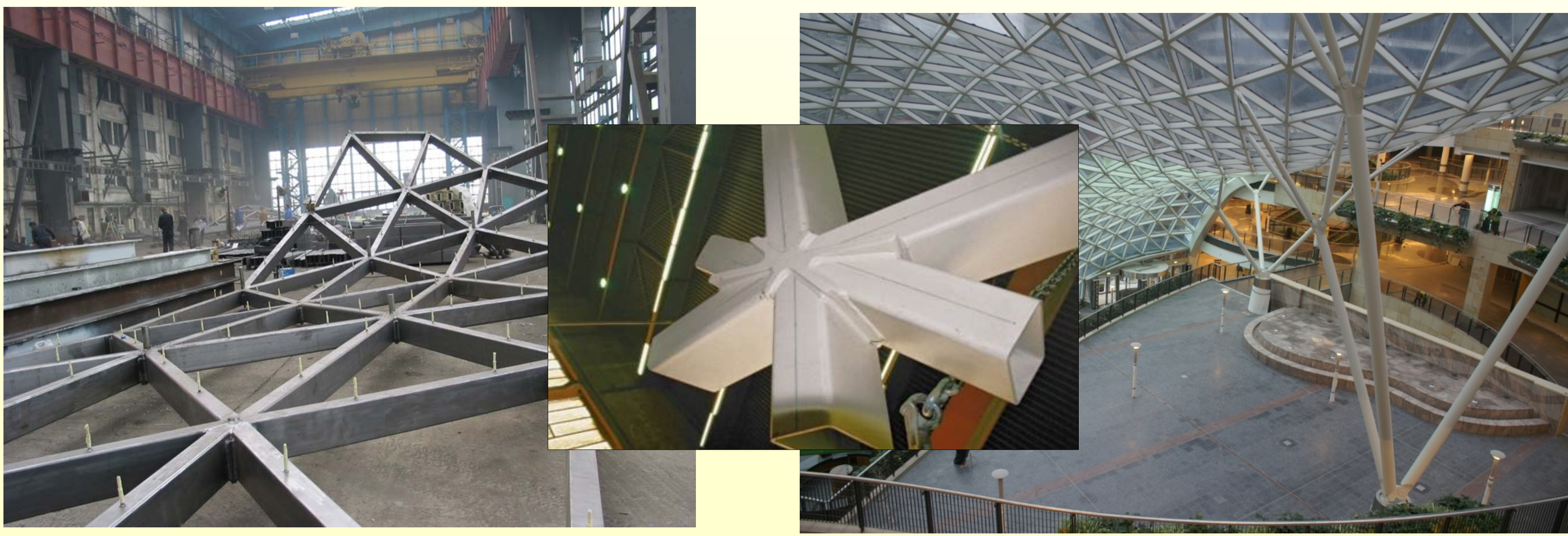
Realizing freeform architecture: simple panels

- Use simple panel types that can be built easily from the chosen material
- Even flat panels are challenging



Realizing freeform architecture: simple panels

- Even triangular panels are challenging because of the higher node complexity
- Yet another topic: substructure geometry



Realizing freeform architecture: geometry adapted to materials

- Ideal panel shape also depends on the material and fabrication technology
- Using sheet metal, „single curved (developable) panels“ are great candidates (F. Gehry)
- F. Gehry was among the first to realize the need for new tools: Gehry Technologies developed advanced software for developable surfaces embedded into the CAD system CATIA

F. Gehry, Walt Disney Concert Hall

Realizing freeform architecture: more ideas

- Repetitive elements: build a freeform structure using only a few different panel shapes, node configurations, molds,….
- Avoid designs which are hard to realize by embedding core aspects of the fabrication directly into the design process. This advances the current process of rationalization which takes place after the design phase (competition)
- Integrate aspects of statics into the shape modeling phase
- ………

Goals of Architectural Geometry

- Make geometrically complex architectural structures affordable through *novel computational tools by linking design, function and fabrication*
- Provide new mathematical methodology for this task … research in Architectural Geometry stimulated mathematical research
- Develop new tools to explore the variety of feasible / optimized designs
- Support innovation in architecture through computation at a level which goes far beyond currently available approaches and digital design systems
- Contribute to and learn from real-world projects

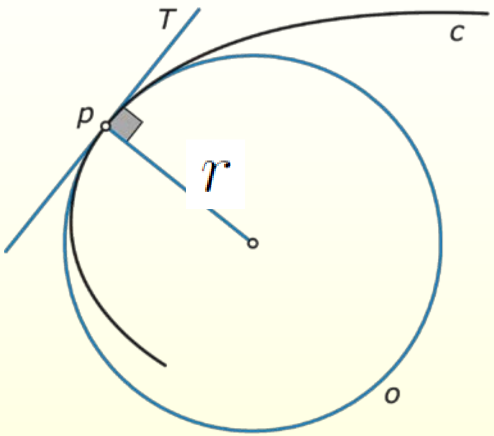
Planar quadrilateral panels

- quad meshes with planar faces (PQ meshes)
  - pioneering work by Schlaich & Schober

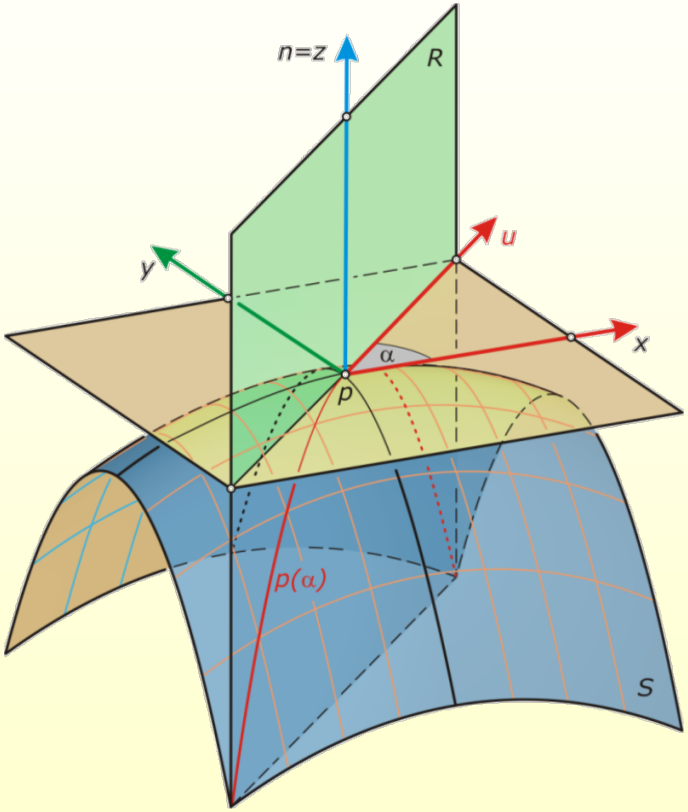
- capable of representing freeform shapes?
- Answer provided by Differential Geometry

Differential geometry: curvatures

- curve



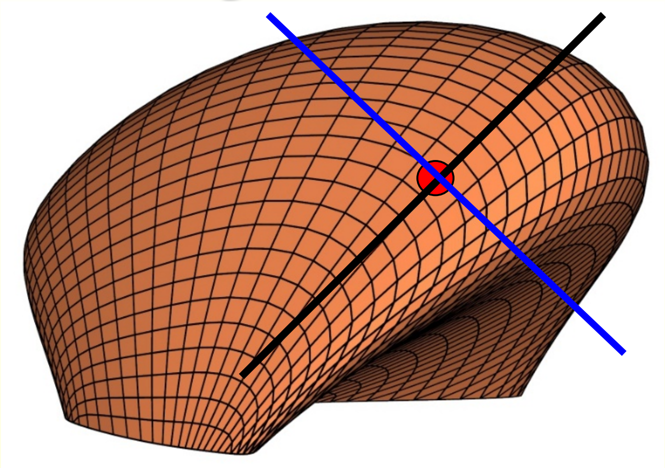
$$\kappa = \frac{1}{r}$$

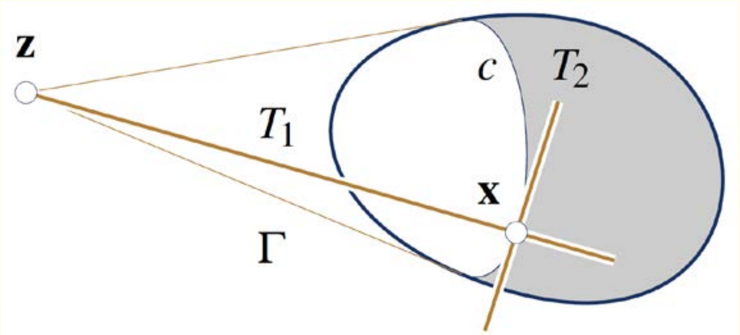


- surface: normal curvatures
- 2 extremal normal curvatures (principal curvatures) in orthogonal tangent directions (principal curvature directions)

Answer provided by differential geometry

- PQ meshes reflect curvature behavior of a smooth underlying surface
- Roughly speaking: given a design surface, at every point, one edge direction determines the other one (conjugate directions)
- *Discrete Differential Geometry*



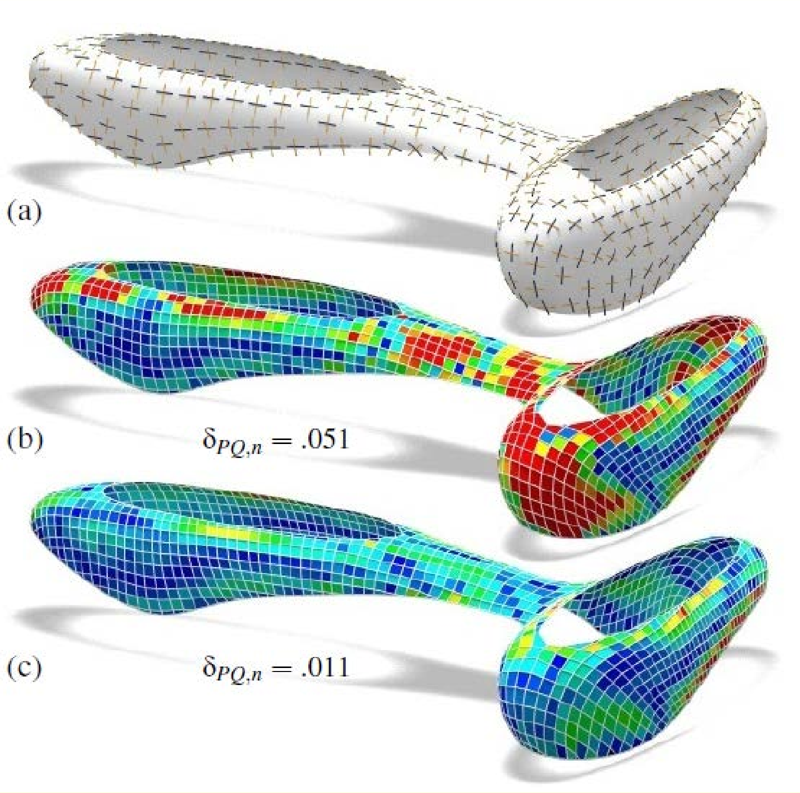


key insight on PQ meshes

- relation to discrete differential geometry: PQ meshes are discrete versions of *conjugate parameterizations*

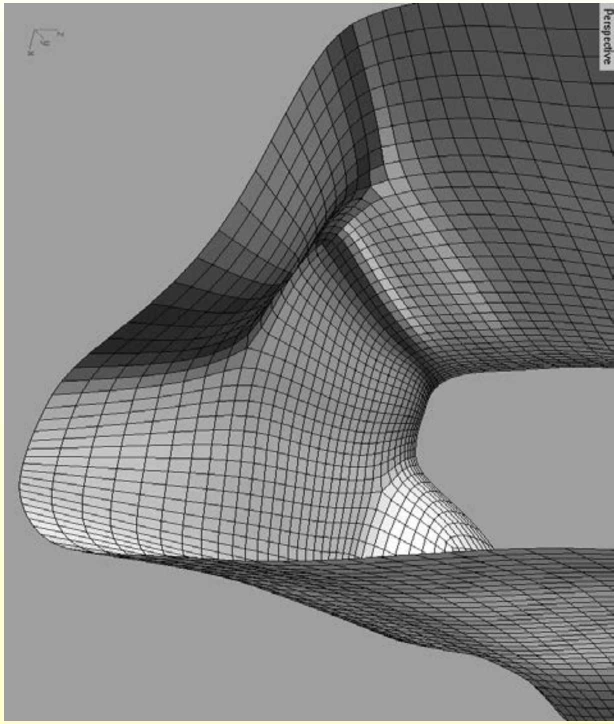
$$\mathbf{x}(u,v) \ \text{ with } \ \det(\mathbf{x}_u, \mathbf{x}_v, \mathbf{x}_{uv}) = 0$$

- any optimization has to be initialized respecting this fact



Computing PQ meshes

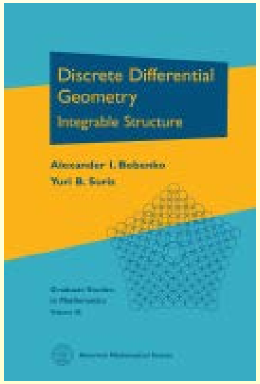
- Computation of a PQ mesh is based on numerical optimization:
- Optimization criteria
  - planarity of faces
  - aesthetics (fairness of mesh polylines)
  - proximity to a given reference surface
- Requires *initial mesh*, found via a careful evaluation of the curvature behavior!



Discrete Differential Geometry

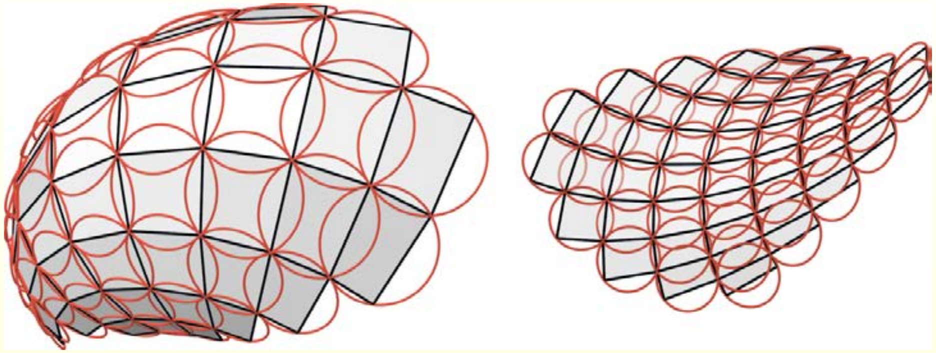
- Develops discrete equivalents of notions and methods of classical differential geometry
- The latter appears as limit of the refinement of the discretization
- Basic structures of DDG related to the theory of integrable systems
- A. Bobenko, Y. Suris: *Discrete Differential Geometry: Integrable Structure*, AMS, 2008

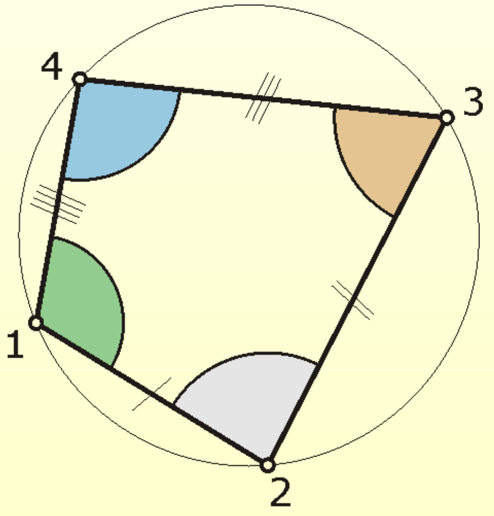
  influenced by research in Architectural Geometry
- Discretize the theory, not the equations!
- Several discretizations; which one is the best?



nearly rectangular panels

- panels as rectangular as possible
- Directions of edges essentially determined by underlying surface (close to principal curvature directions)
- *circular mesh*
- average of opposite angles in each quad = 90º
- concept of *Möbius sphere geometry*



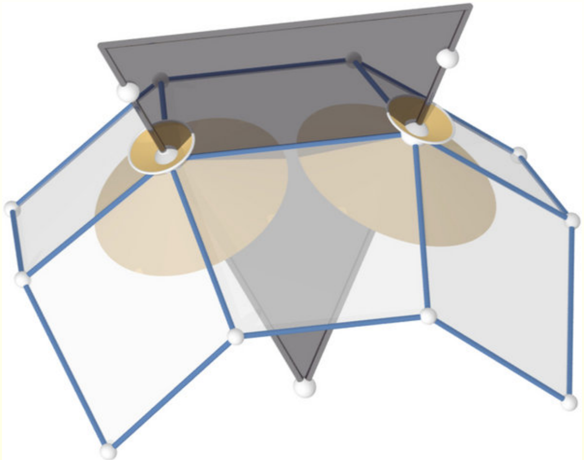


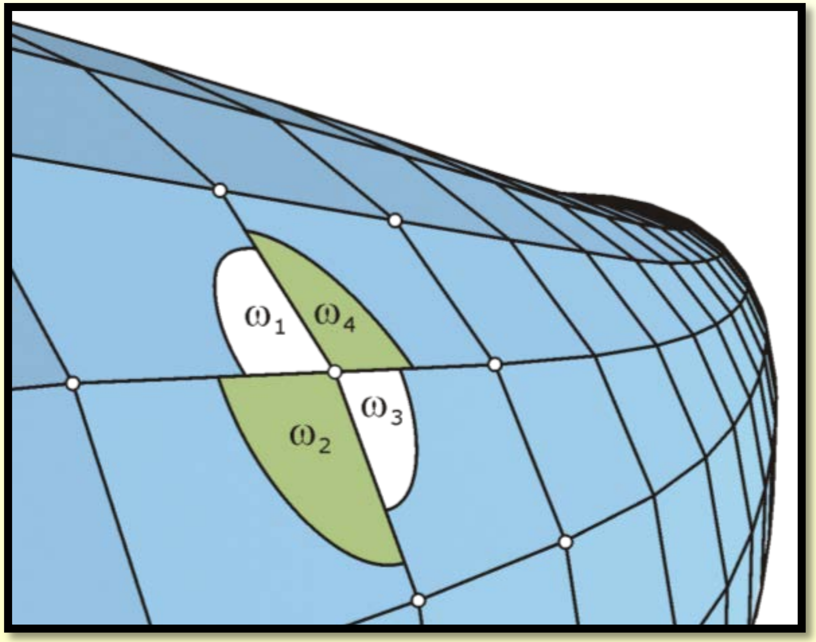
nearly rectangular panels

- for architecture, even better: *conical mesh*
- PQ mesh is conical if at each vertex the incident face planes are tangent to a right circular cone
- equal sum of opposite angles at each vertex (average close to 90º)

$$\omega_1 + \omega_3 = \omega_2 + \omega_4.$$

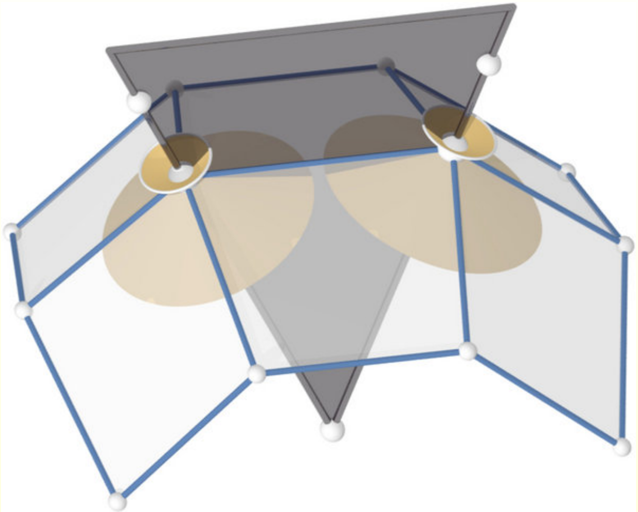
- concept of Laguerre sphere geometry

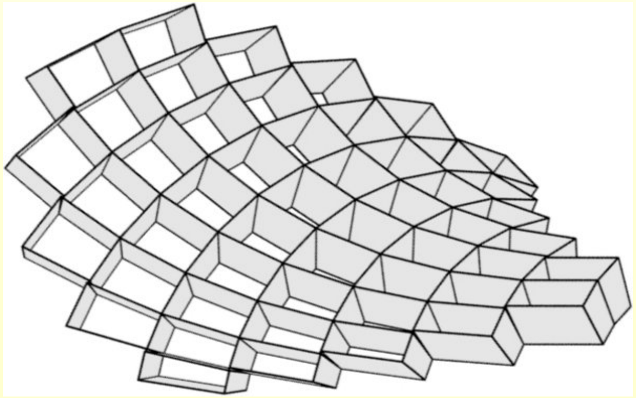


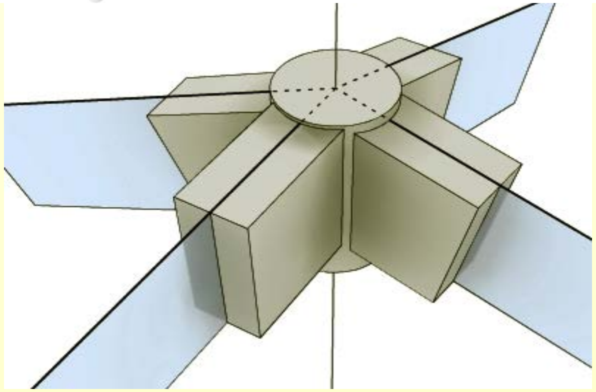


Conical meshes possess elegant support structures

- neighboring cone axes (discrete normals) are coplanar
- conical mesh has offsets at constant face-face distance and determines a *torsion-free support structure*



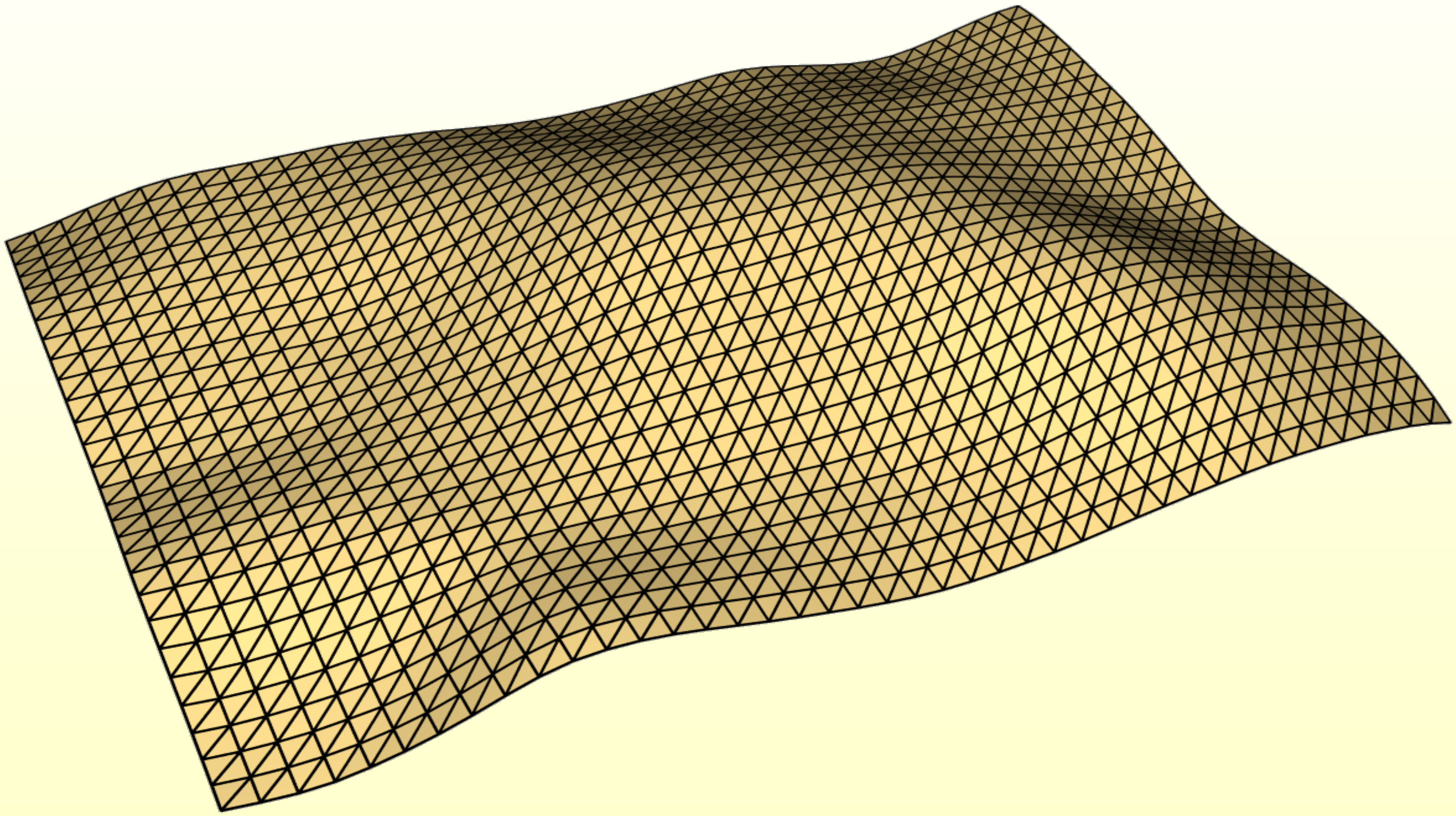


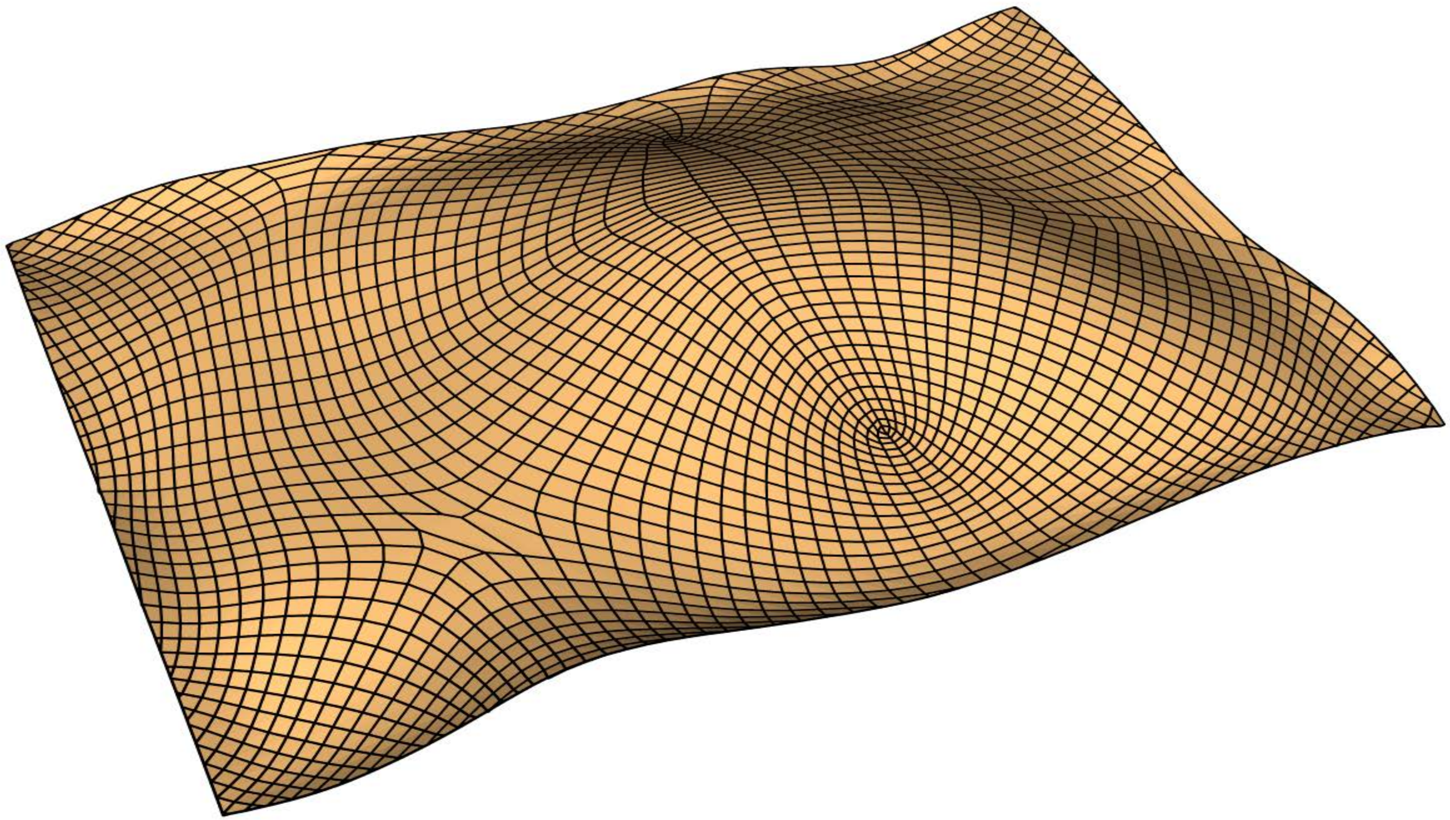


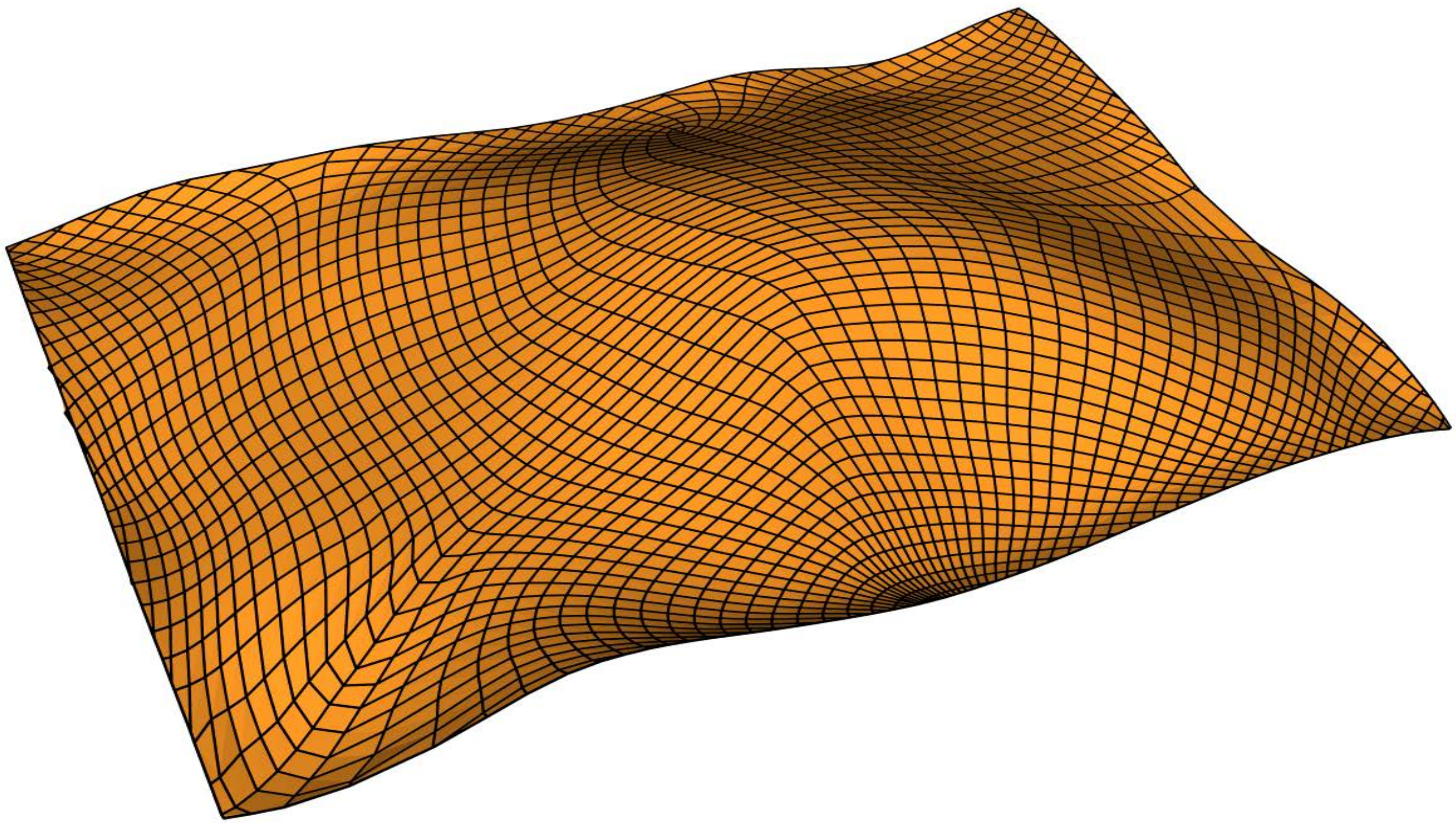
Conical mesh

Museum of Islamic Arts at Louvre
Architects: Mario Bellini Architects, Rudy Ricciotti
Construction: Waagner Biro
Geometry consulting: Evolute

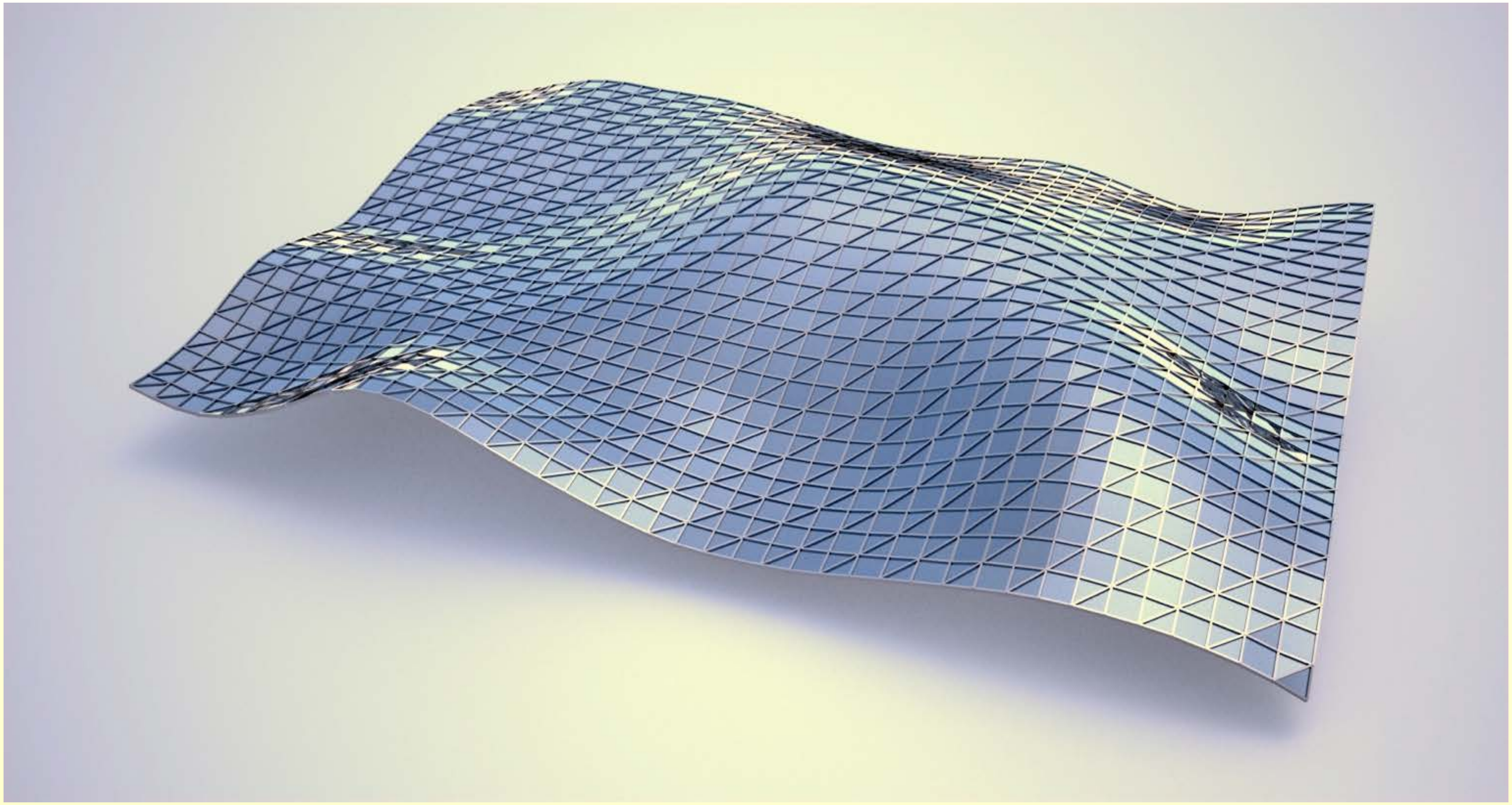
triangle mesh





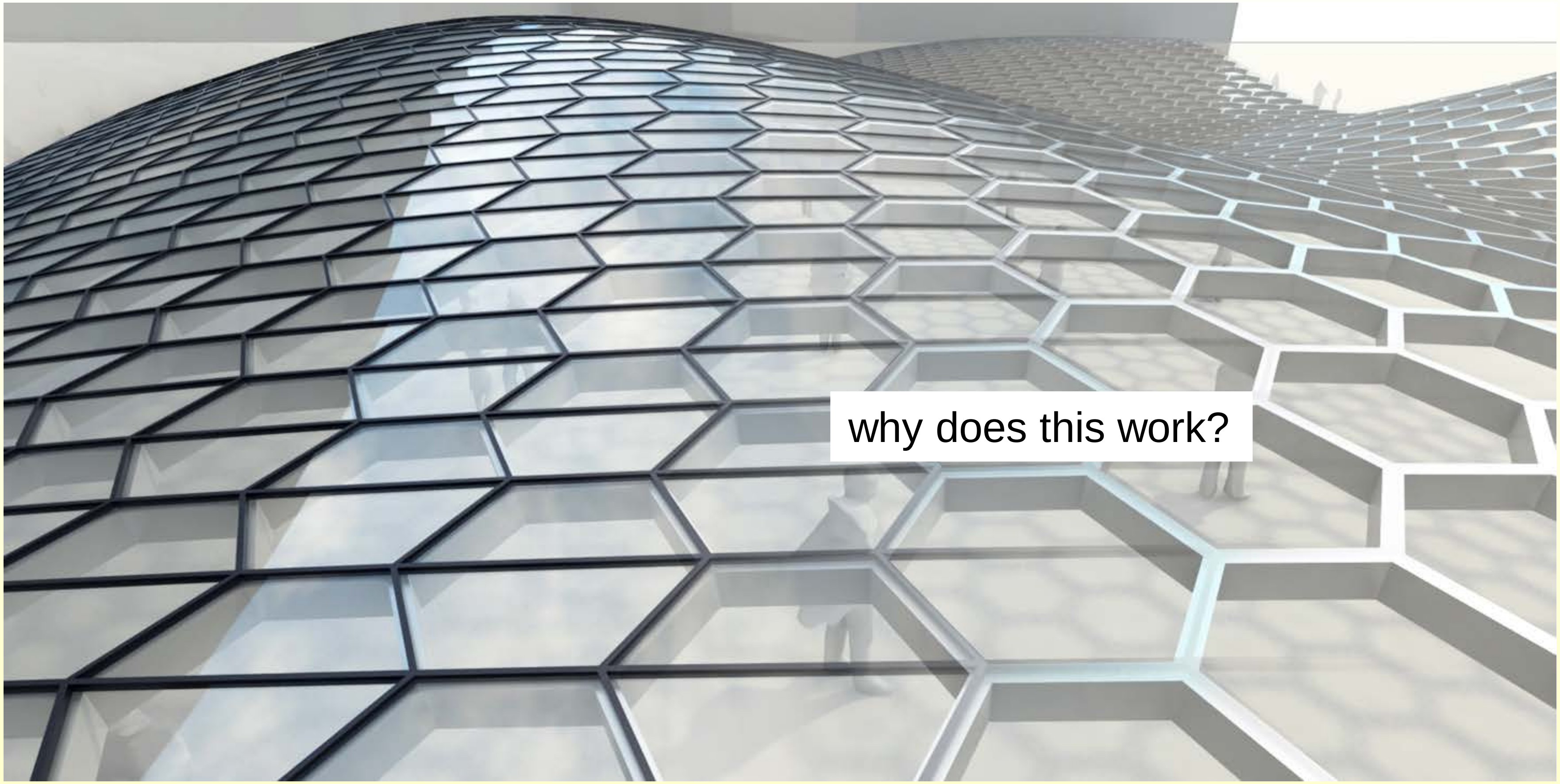


evolute.
research and consulting




31/05/2011

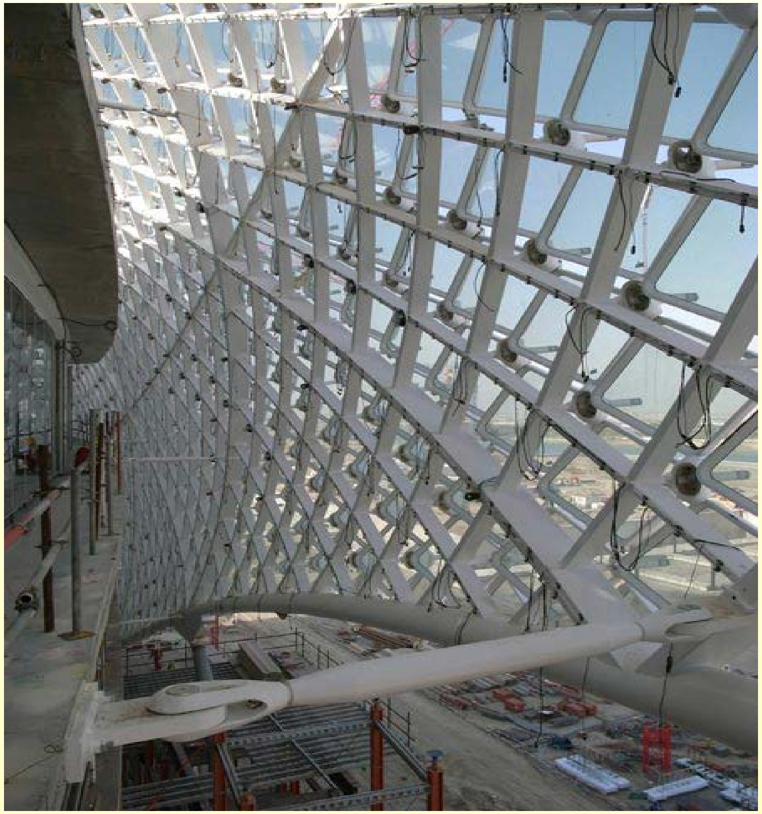
Planar quads on top of a honeycomb structure



Yas Island Marina Hotel
Abu Dhabi
Architect: Asymptote Architecture
Steel/glass construction: Waagner Biro

steel beam layout

- Faces of base quad mesh non-planar
- *node axes as solution of an optimization problem*



Quad-based structures for support and shading

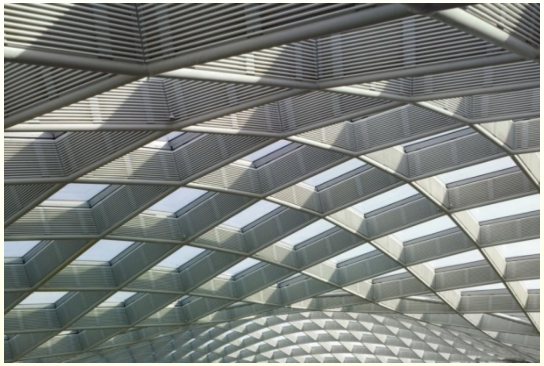
Kogod Courtyard, Smithsonian, Foster & Partners

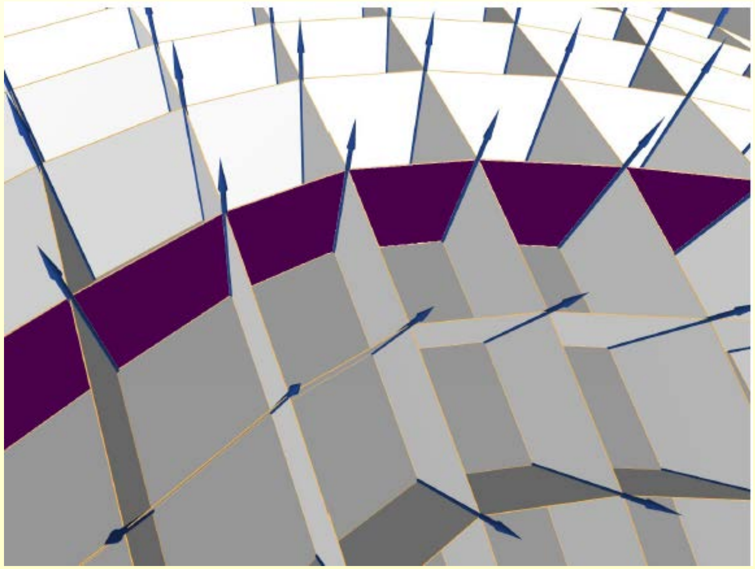
mathematical framework

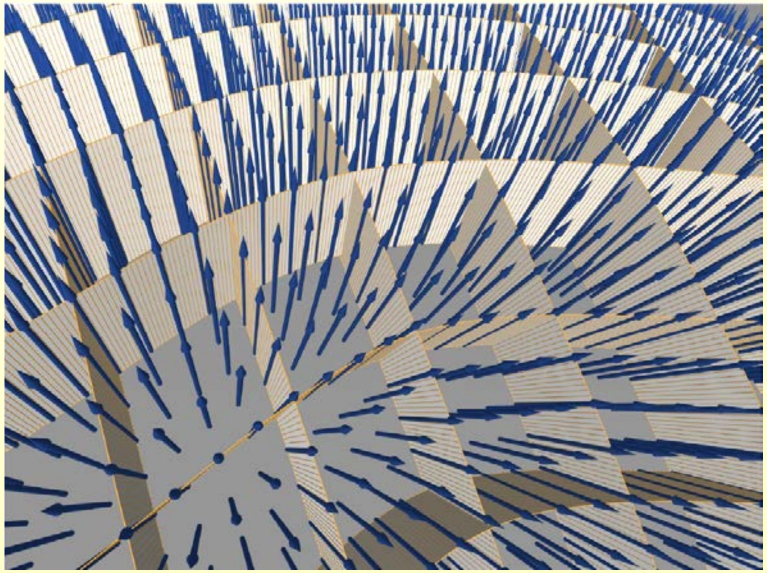
- *What is the right theoretical framework for such structures?*

Discrete line congruences

torsal parameterizations [Doliwa et al. 2000]





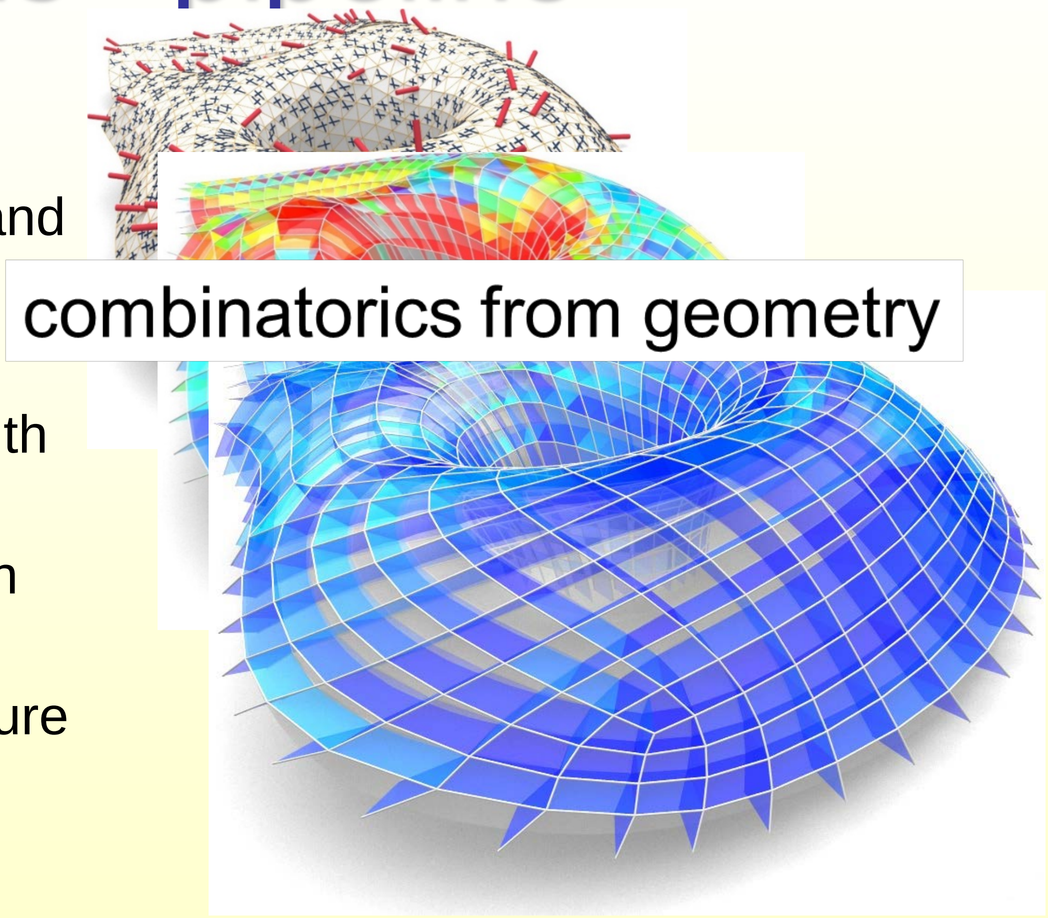


Support structures and shading systems - pipeline

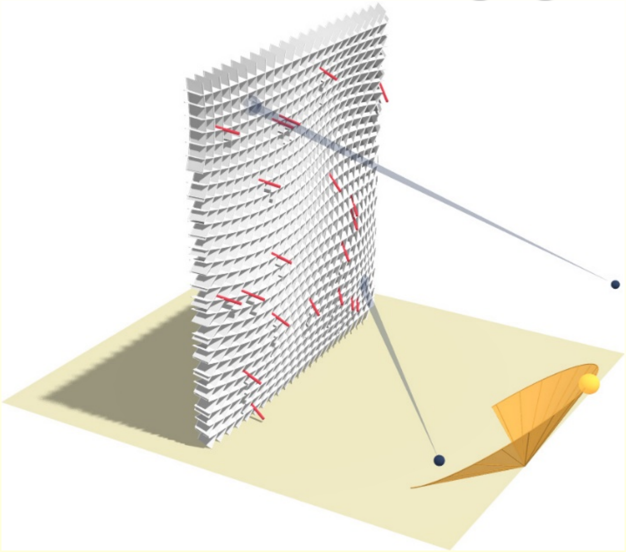
Problem: compute torsion-free structure with desired properties (e.g. orientations) of node axes and planes (shading panels).

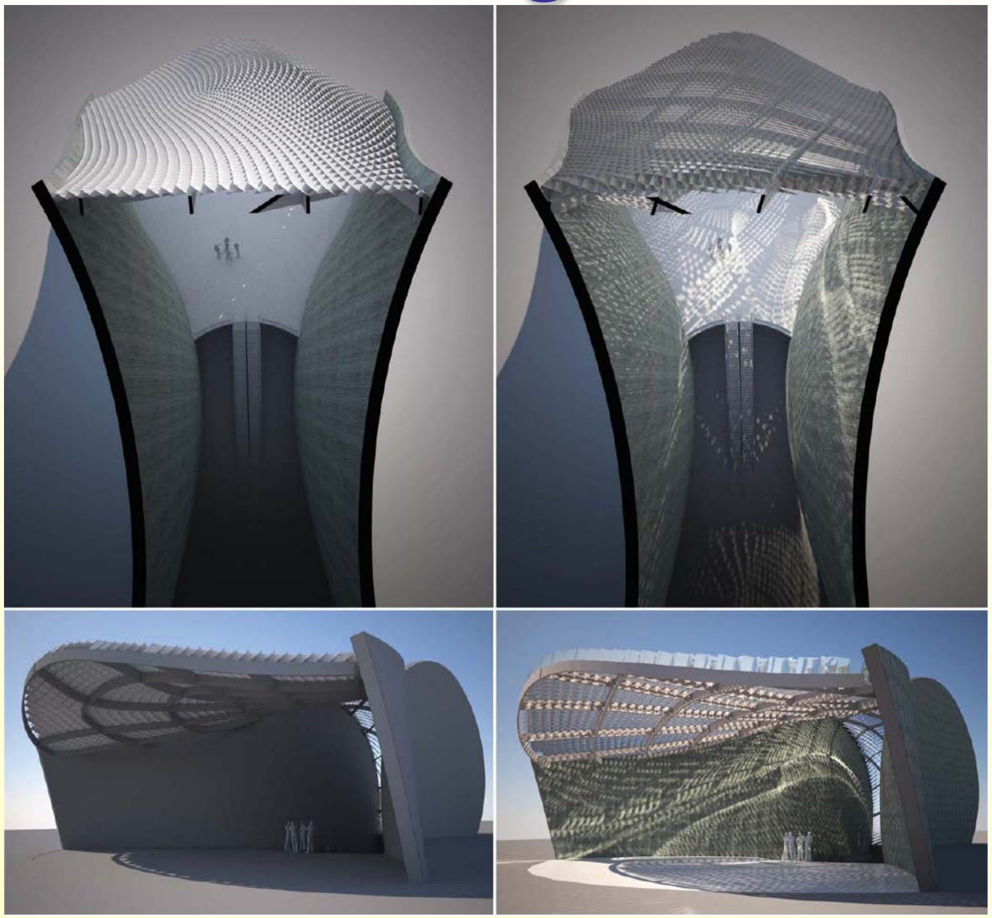
Main idea:

- compute a line congruence with required properties
- extract torsal parameterization (central algorithm)
- final optimization of the structure

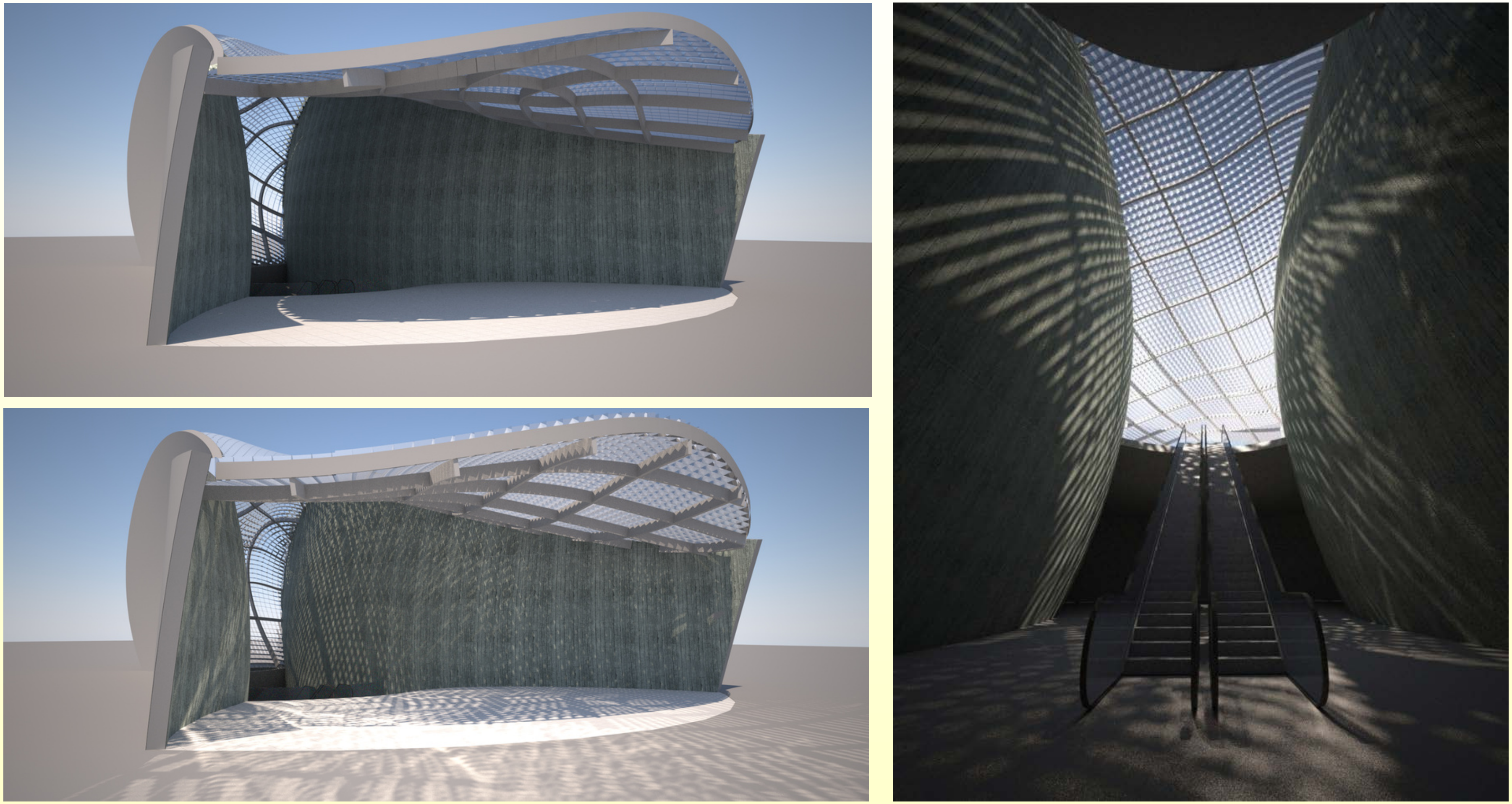


Quad-based structures for support and shading





Systems for shading and indirect lighting



(Jiang et al., SGP 2013)

developable surfaces in architecture

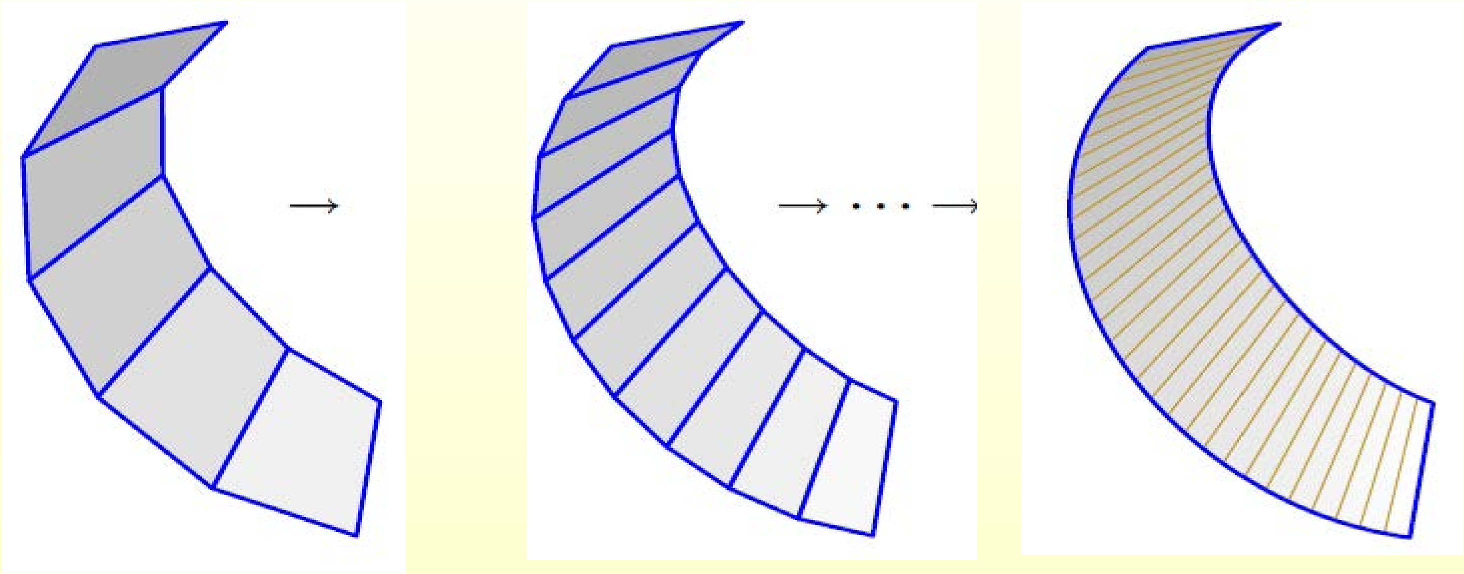
- (nearly) developable surfaces

F. Gehry, Guggenheim Museum, Bilbao

F. Gehry, Walt Disney Concert Hall, Los Angeles

developable surface strips

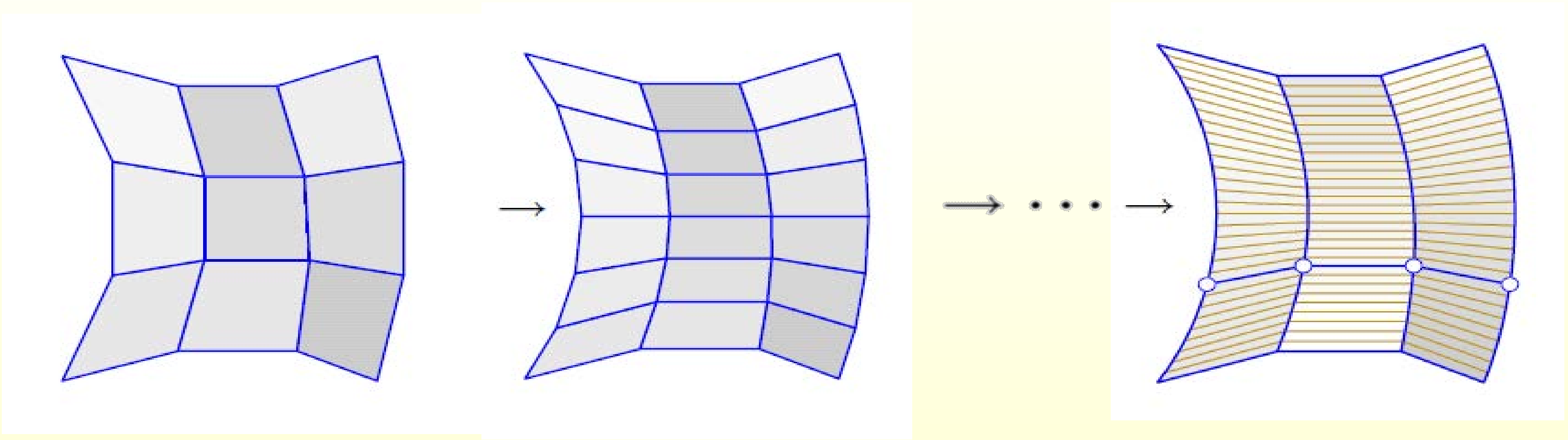
- Refinement of a PQ strip (iterate between subdivision and PQ optimization)



Limit: developable surface strip

D-strip models

One-directional limit of a PQ mesh:

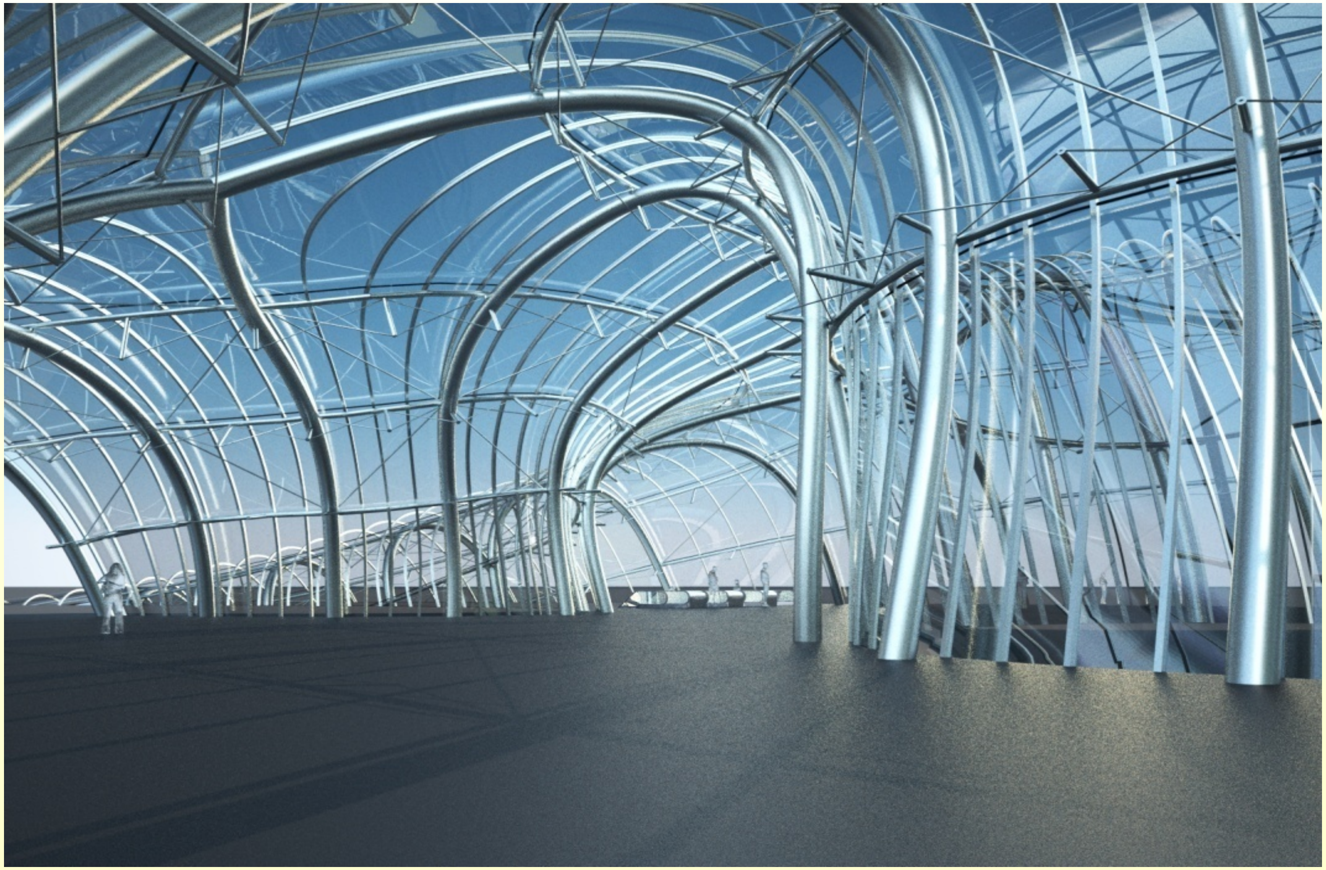


developable strip model (D-strip model)

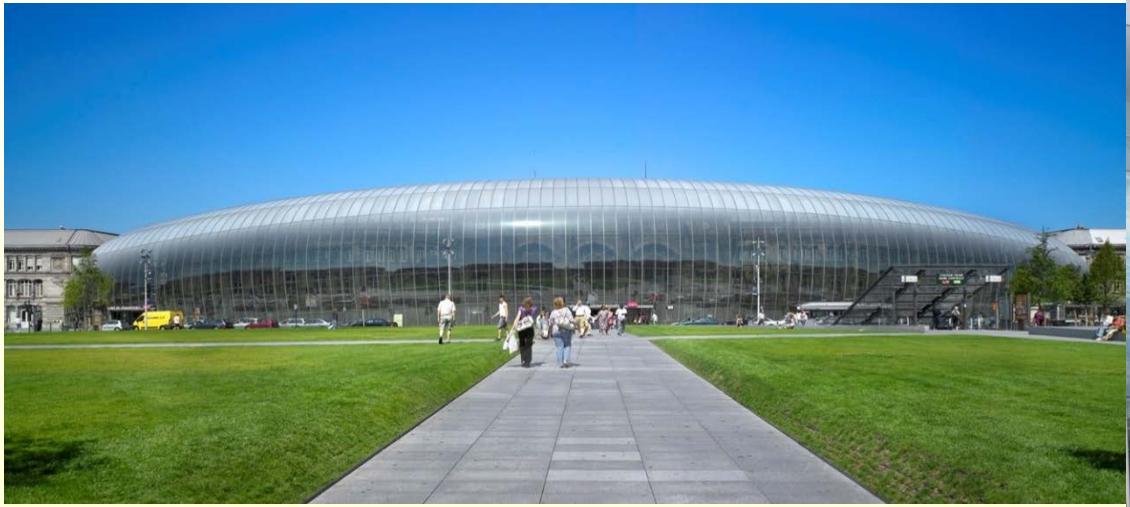
semi-discrete surface representation

initiated research on semi-discrete surface representations

Design from single-curved panels
based on subdivision modeling



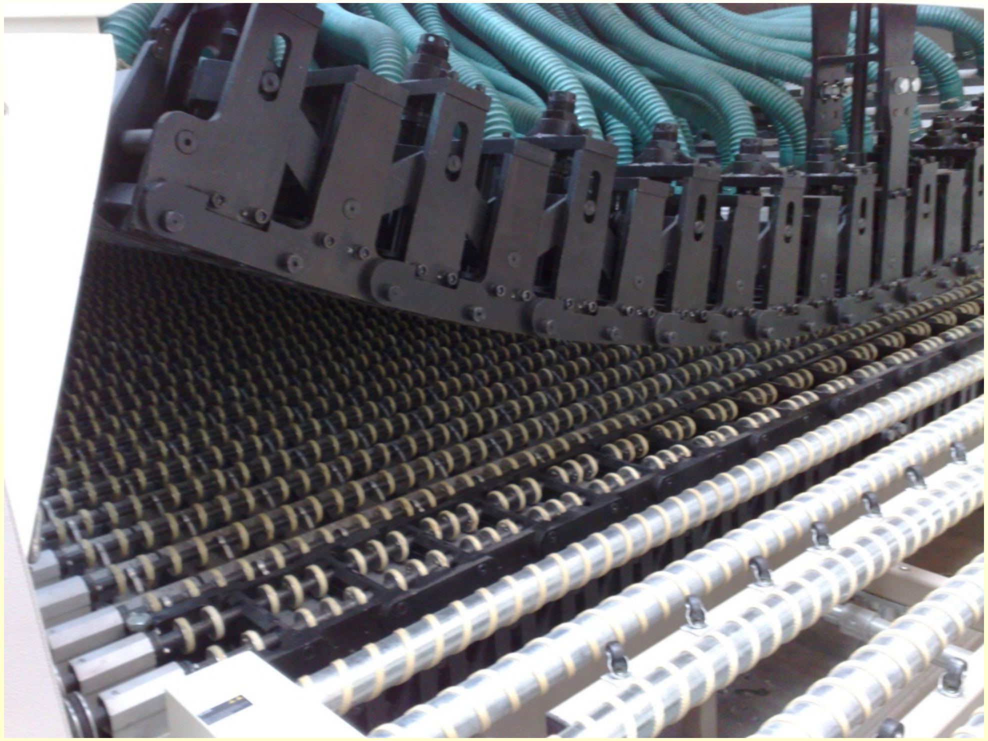
Cylindrical panels

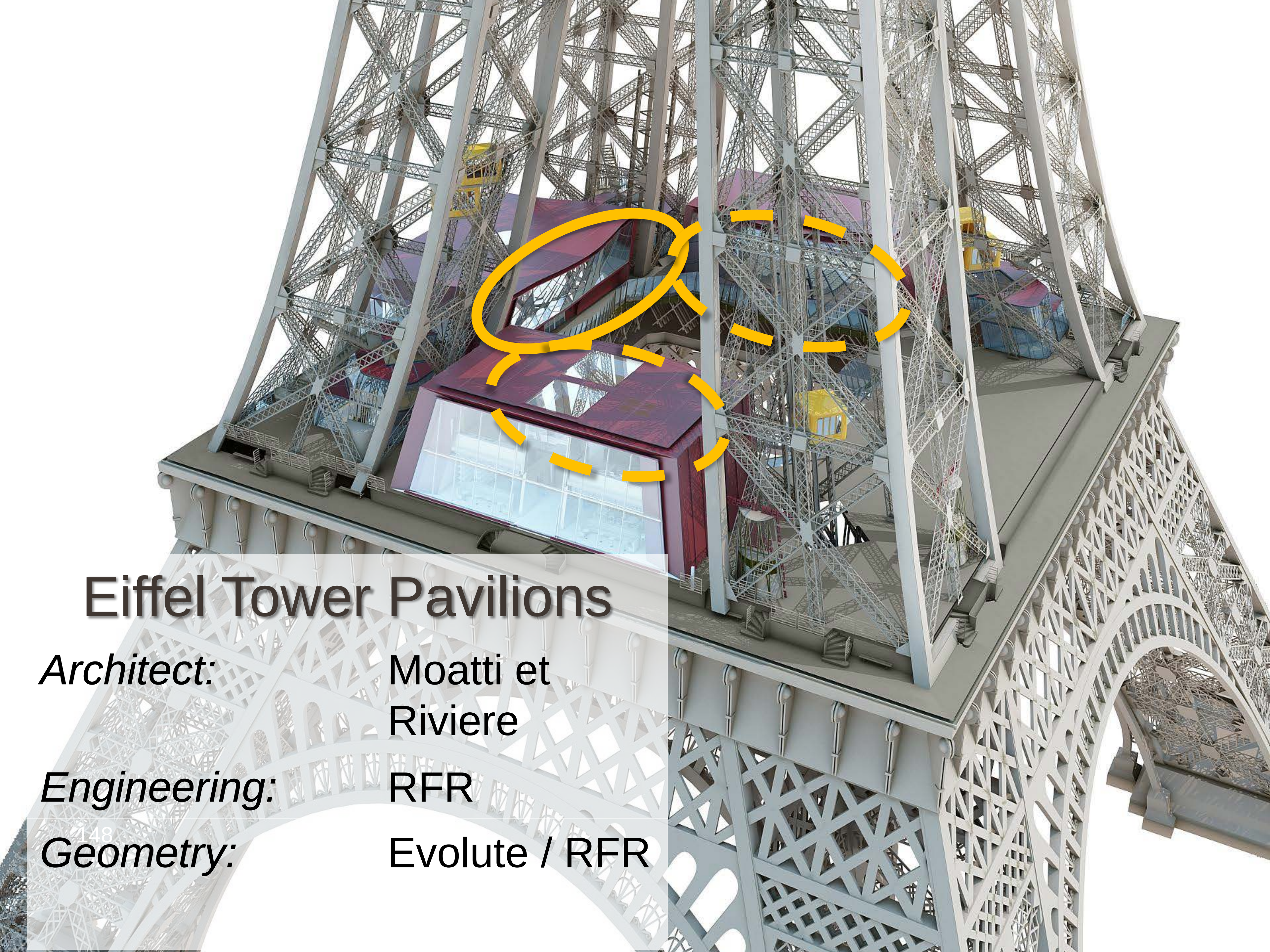


TGV station, Strasbourg

curved glass: *right circular cylinders* preferred

machines to produce cylindrical glass panels



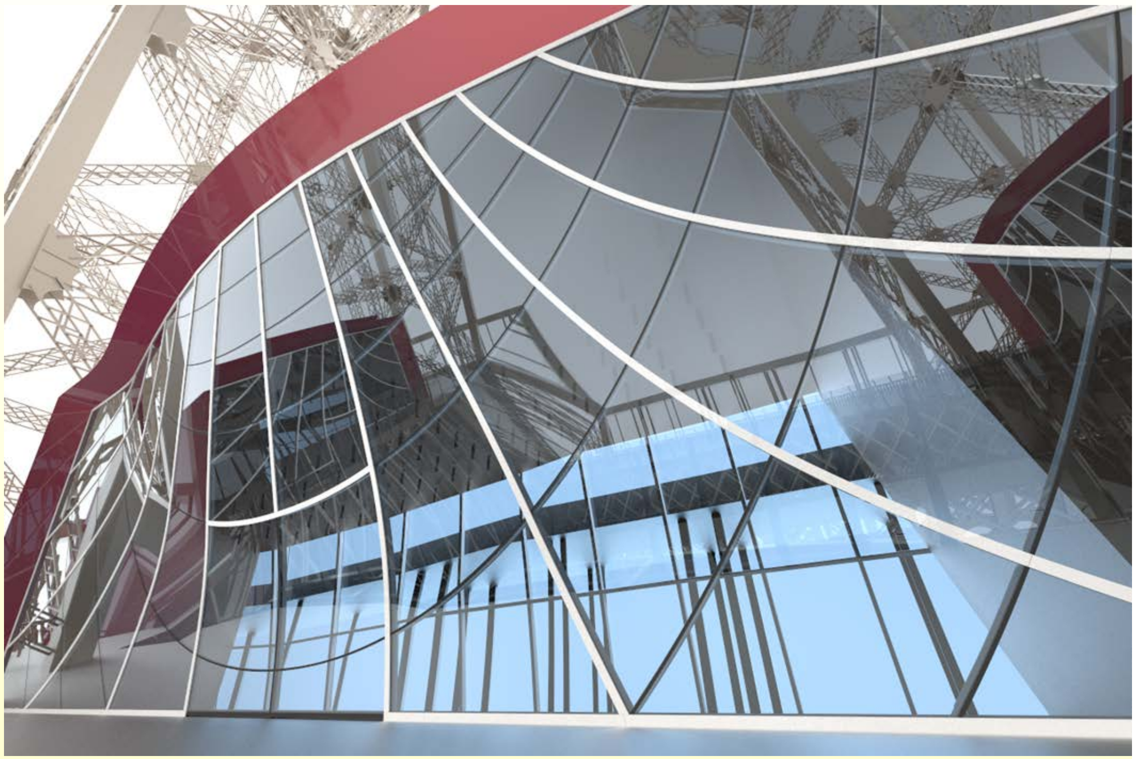


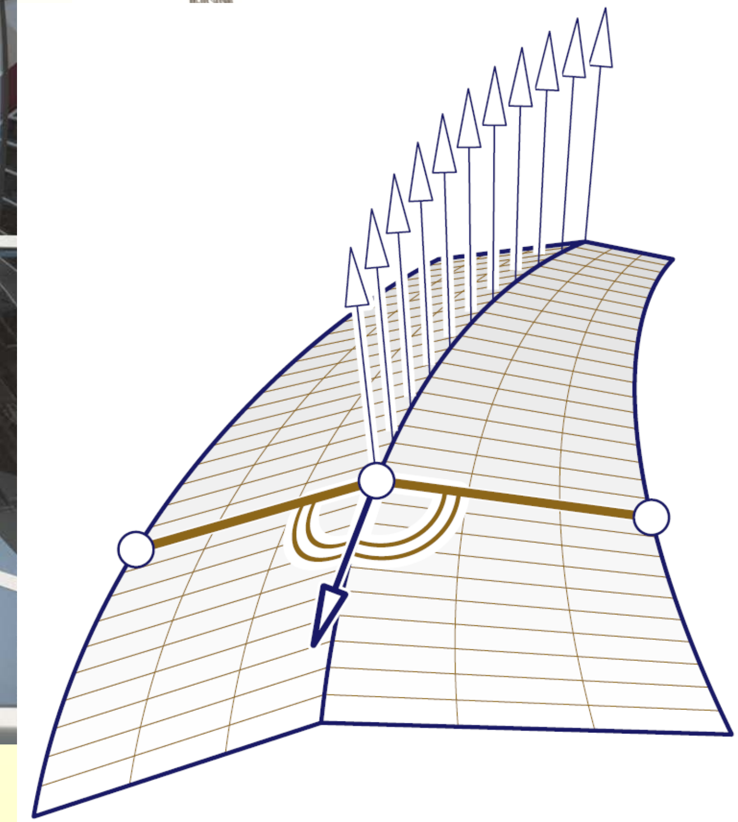
Eiffel Tower Pavilions

Architect:	Moatti et Riviere
Engineering:	RFR
Geometry:	Evolute / RFR

Eiffel Tower Pavilions

architects.
Moatti et Rivière

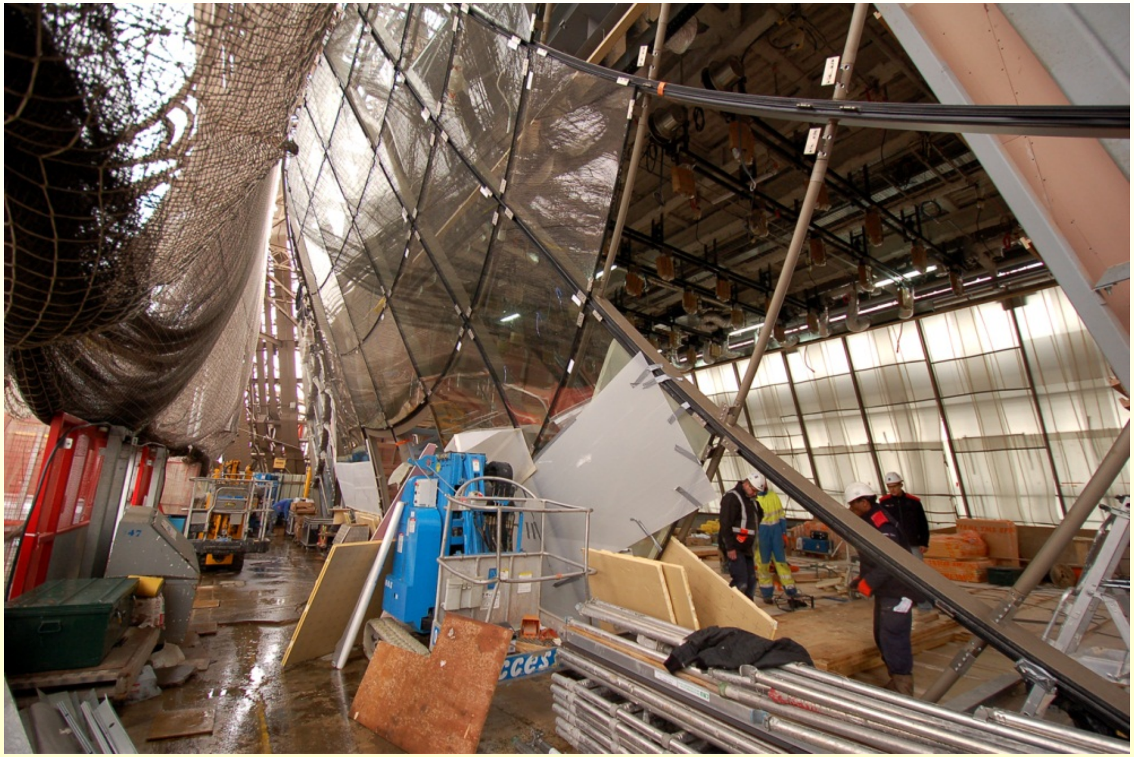




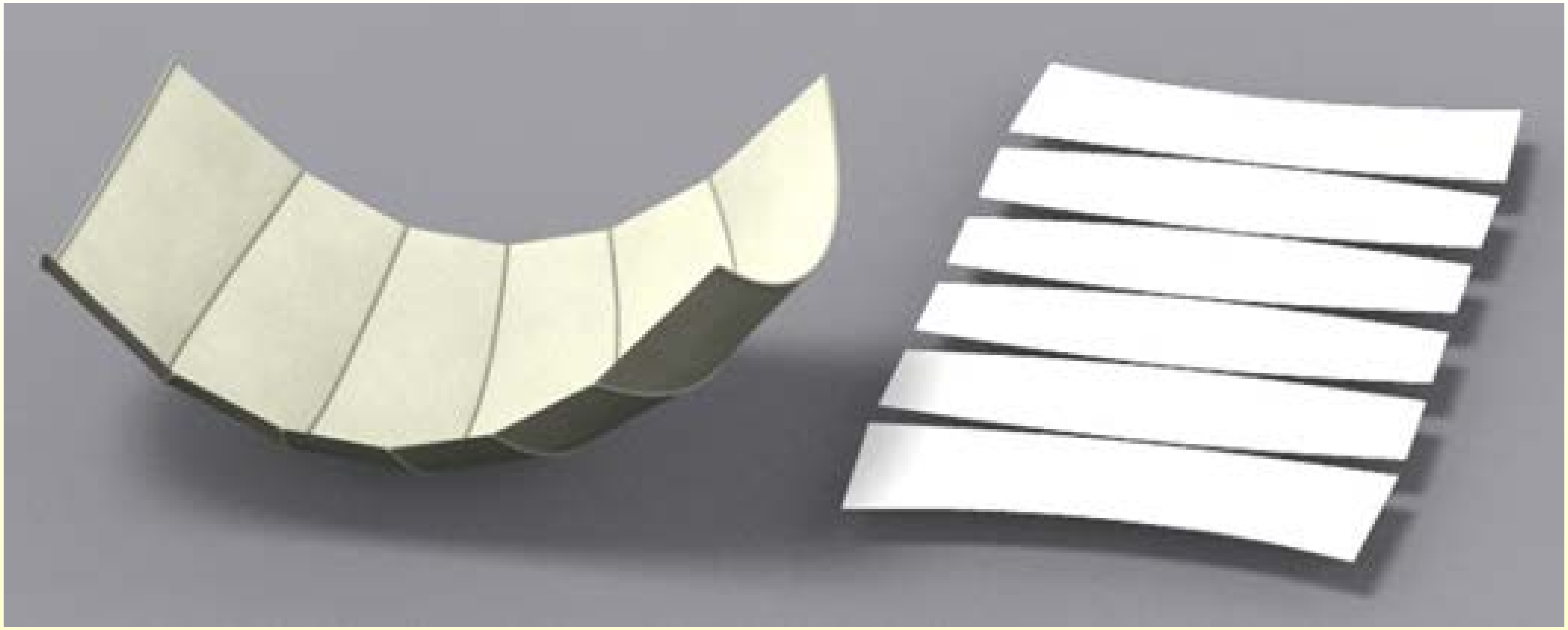
evolute.
research and consulting

approximate by cylindrical panels

curved supporting beams



development

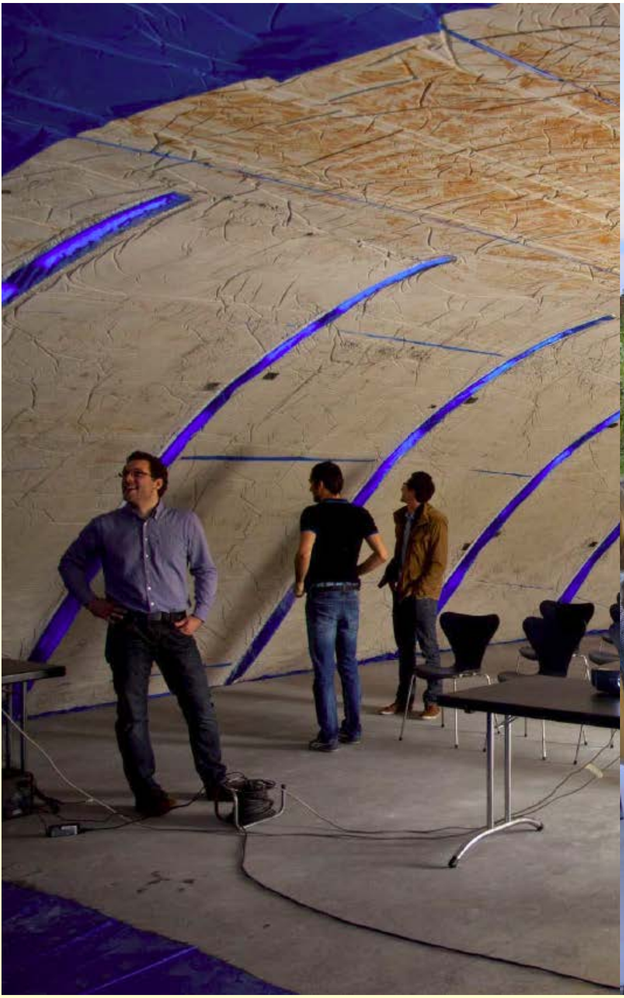


Bending concrete: Pneumatic forming of hardened concrete

- J. Kollegger, B. Kromoser, TU Wien


Toyo Ito: Taichung Metropolitan Opera

Pneumatic forming of hardened concrete

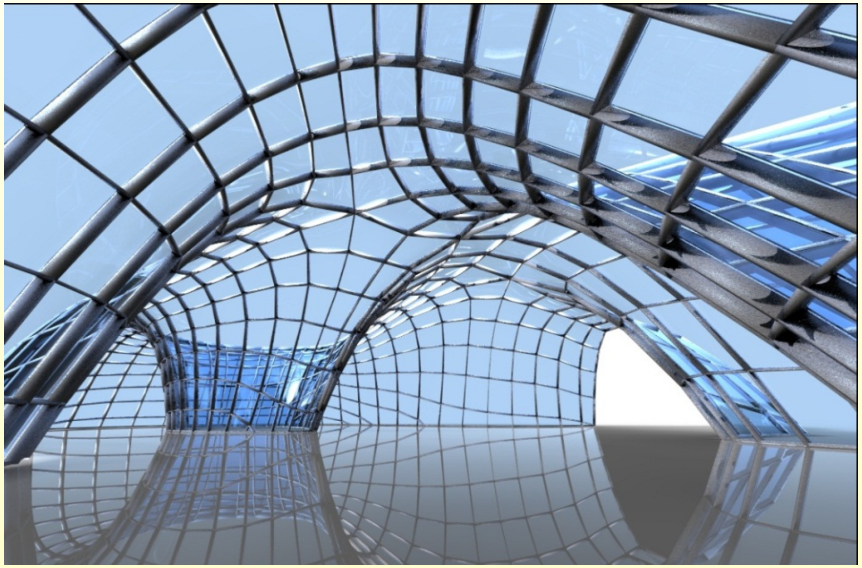


P. Block, ETH Zürich

Shape modeling with constraints from statics and manufacturing

- Design of self-supporting *freeform* surfaces
- Design of self-supporting PQ meshes
- Connections to discrete differential geometry?

Block & Ochsendorf, 2007,…



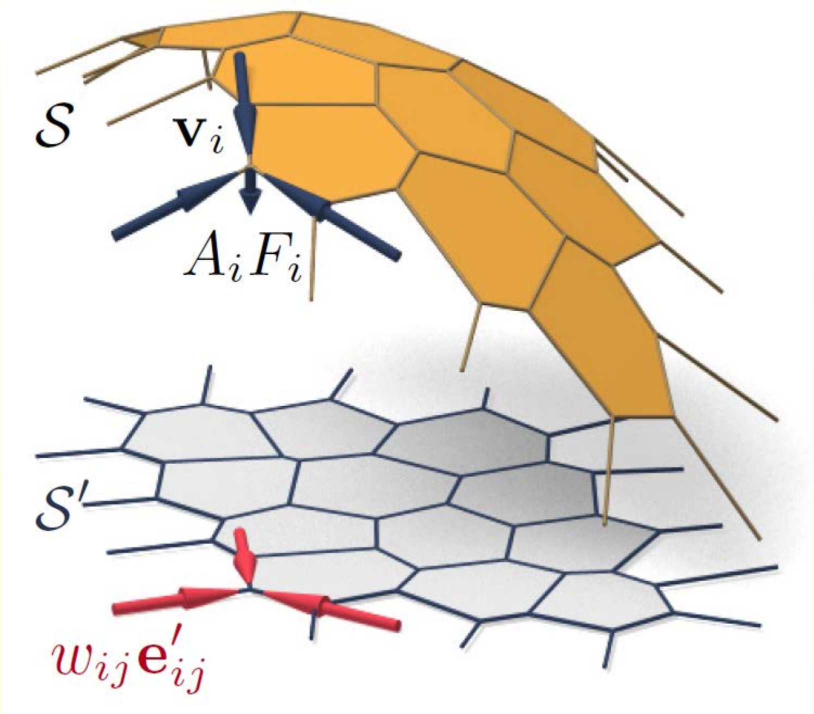
thrust network analysis to check self-supportig property

- *thrust network*: network of compressive forces inside the envelope of the structure, in balance with the loads
- Edge forces balance gravity:

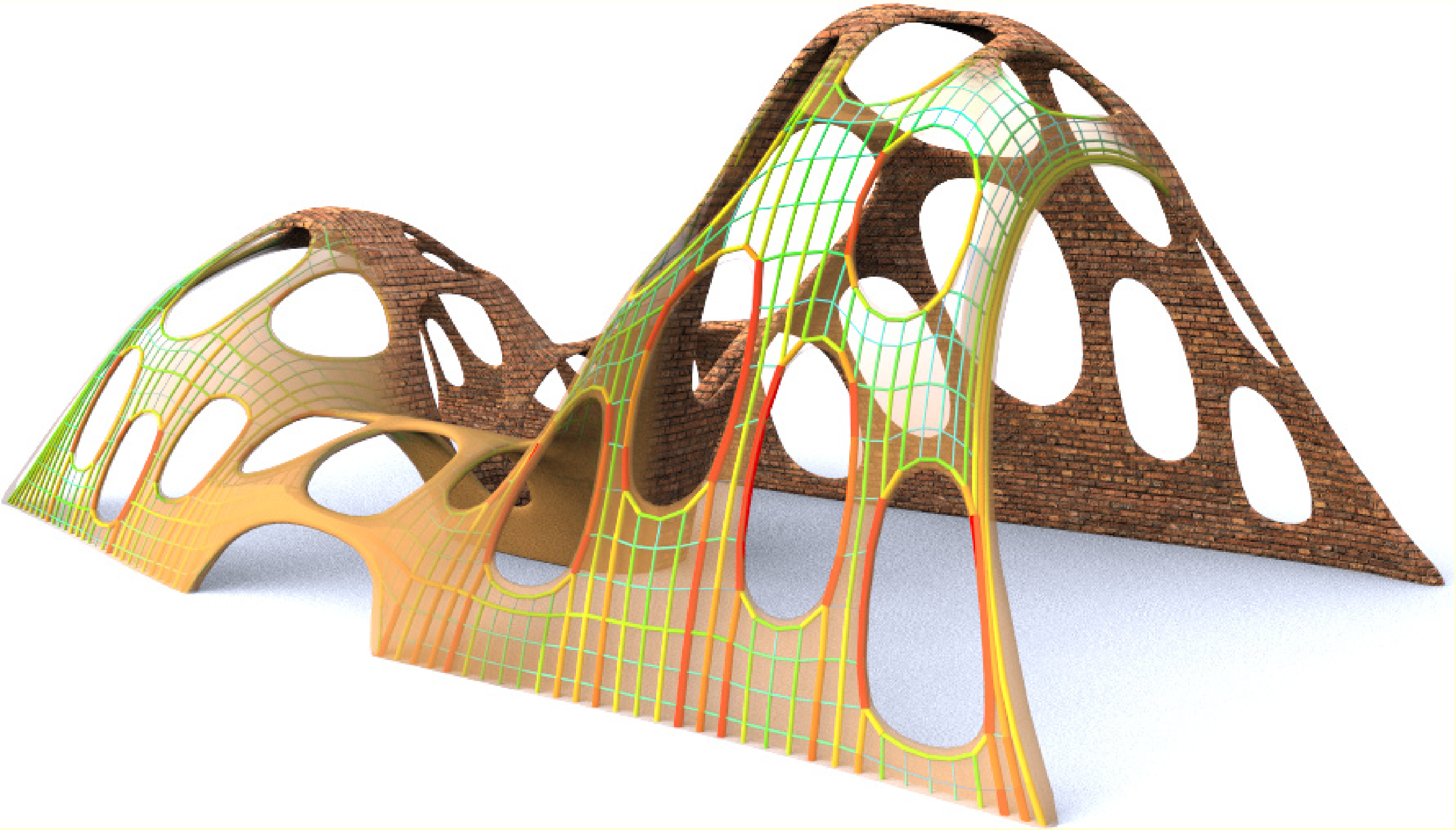
$$\sum_j w_{ij}\mathbf{e}_{ij} = \begin{pmatrix} 0 \\ 0 \\ A_i F_i \end{pmatrix}$$

- the planar system of horizontal force components is in equilibrium

$$\sum_j w_{ij}\mathbf{e}'_{ij} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

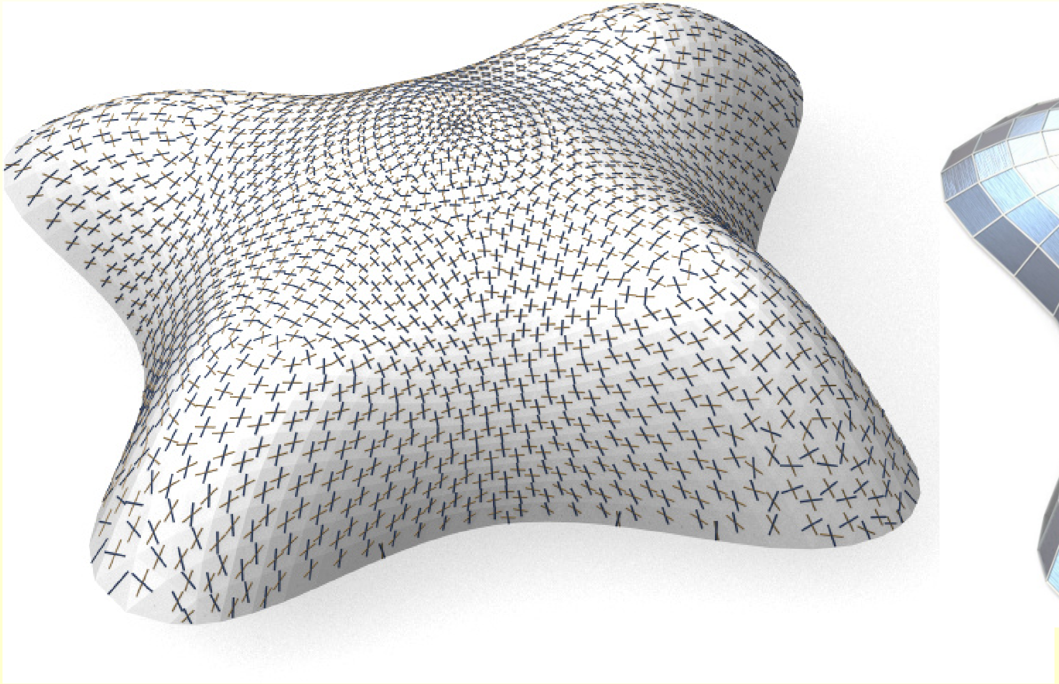


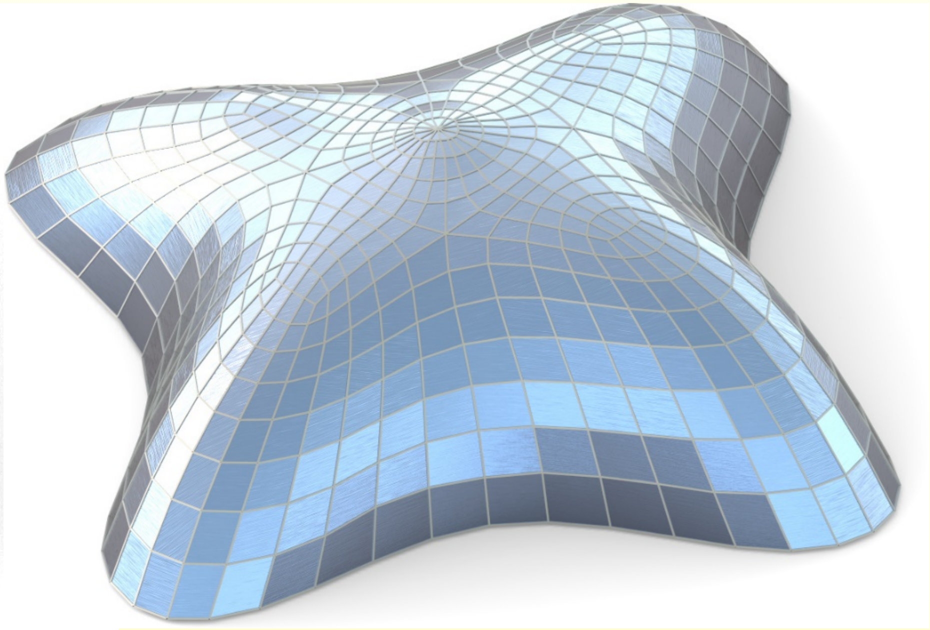
Design via optimization



PQ meshes in static equilibrium

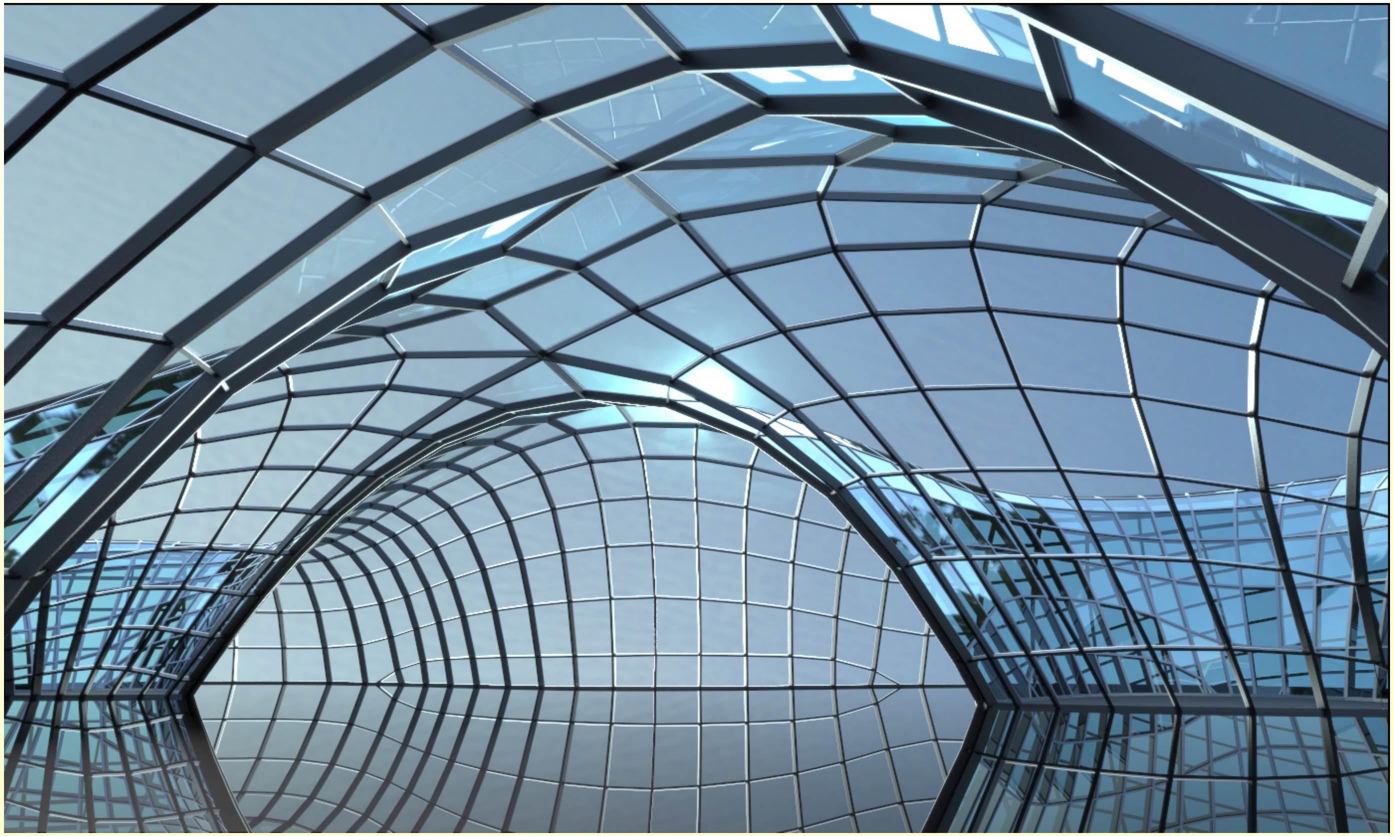
- PQ mesh in static equilibrium has a geometric characterization: *discrete version of the network of relative principal curvature lines with respect to the Airy stress surface*





PQ meshes in static equilibrium

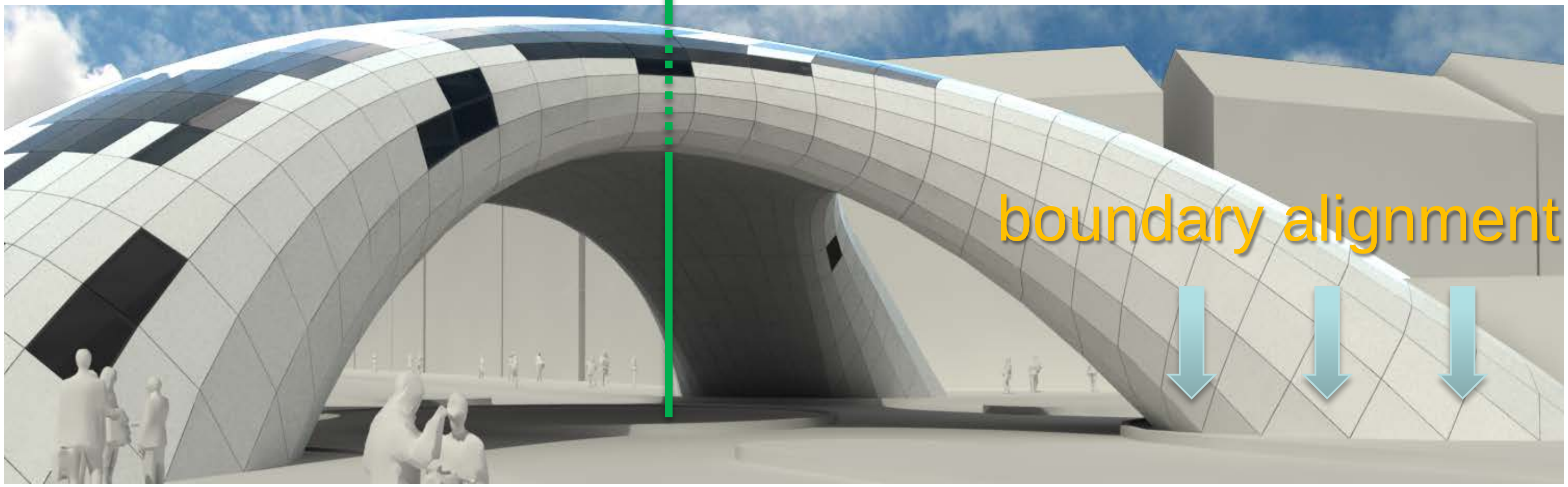
steel/glass structures with low moments in nodes



Form-finding with polyhedral meshes

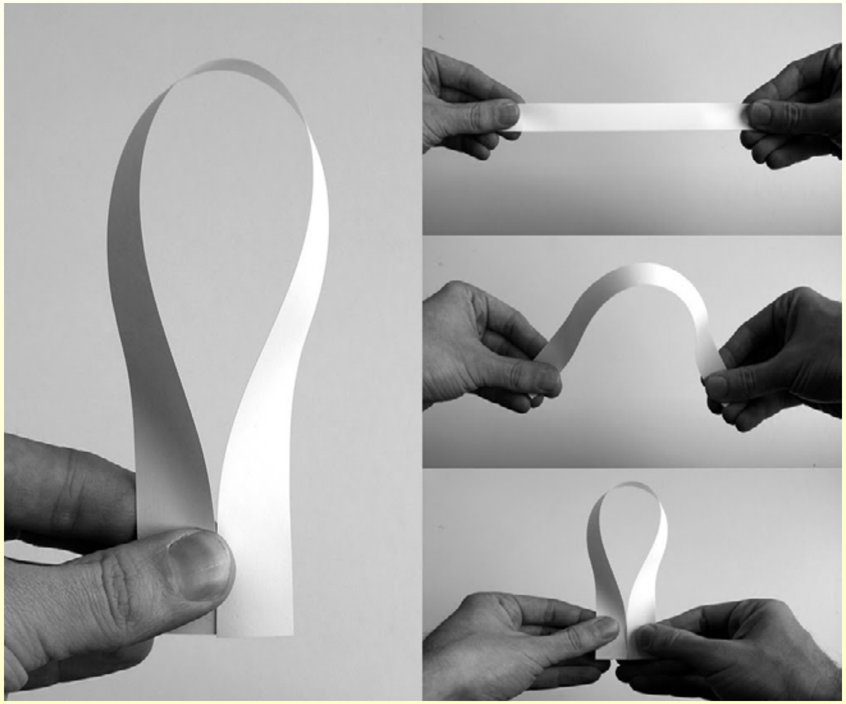
- Interactive system for form-finding and shape exploration of polyhedral meshes (with additional constraints on statics, manufacturing,…)

non-height field self-supporting



Developable surfaces

Working with materials which bend, but do not stretch



Bending wood

Burj Khalifa, F. Gehry

Bending sheet metal

F. Gehry

Design: Kyungeun Ko

Curved folding

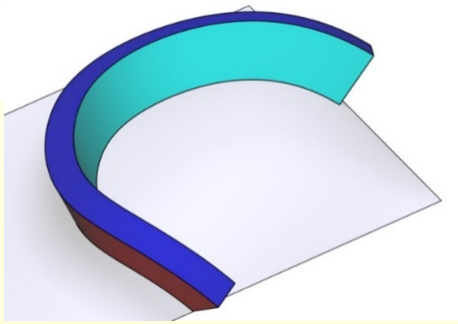
E. & M. Demaine

Eiffel Tower Pavilions: dev. surfaces from glass

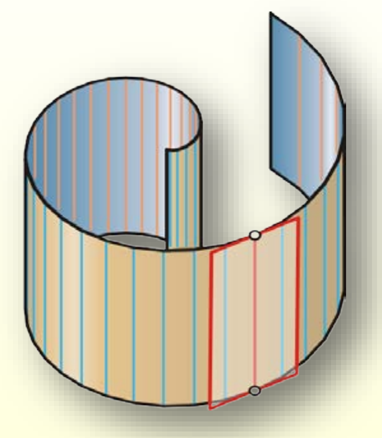
Moatti et Riviere, RFR, Evolute

developable supporting beams

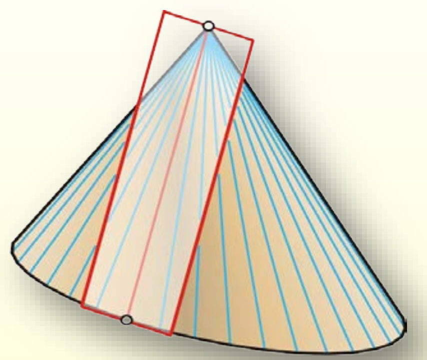
evolute.
research and consulting



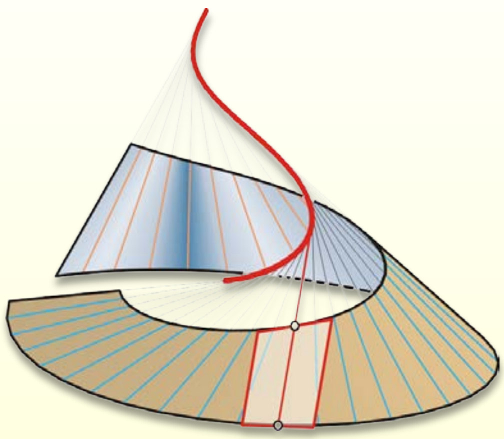
Geometry: basic types of developable surfaces



cylinder



cone

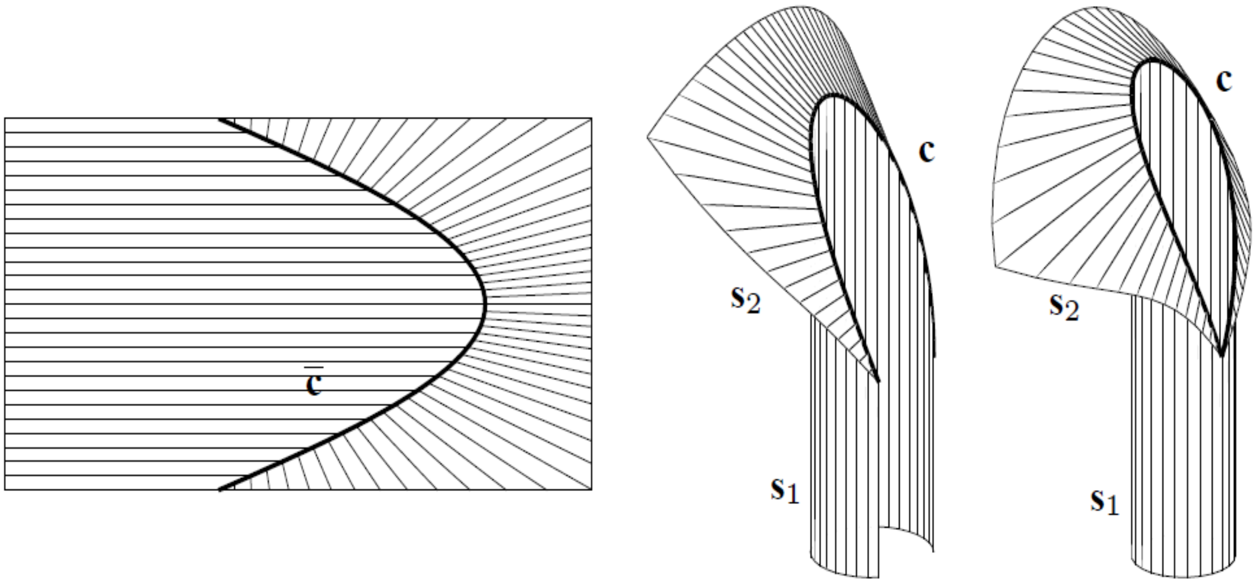


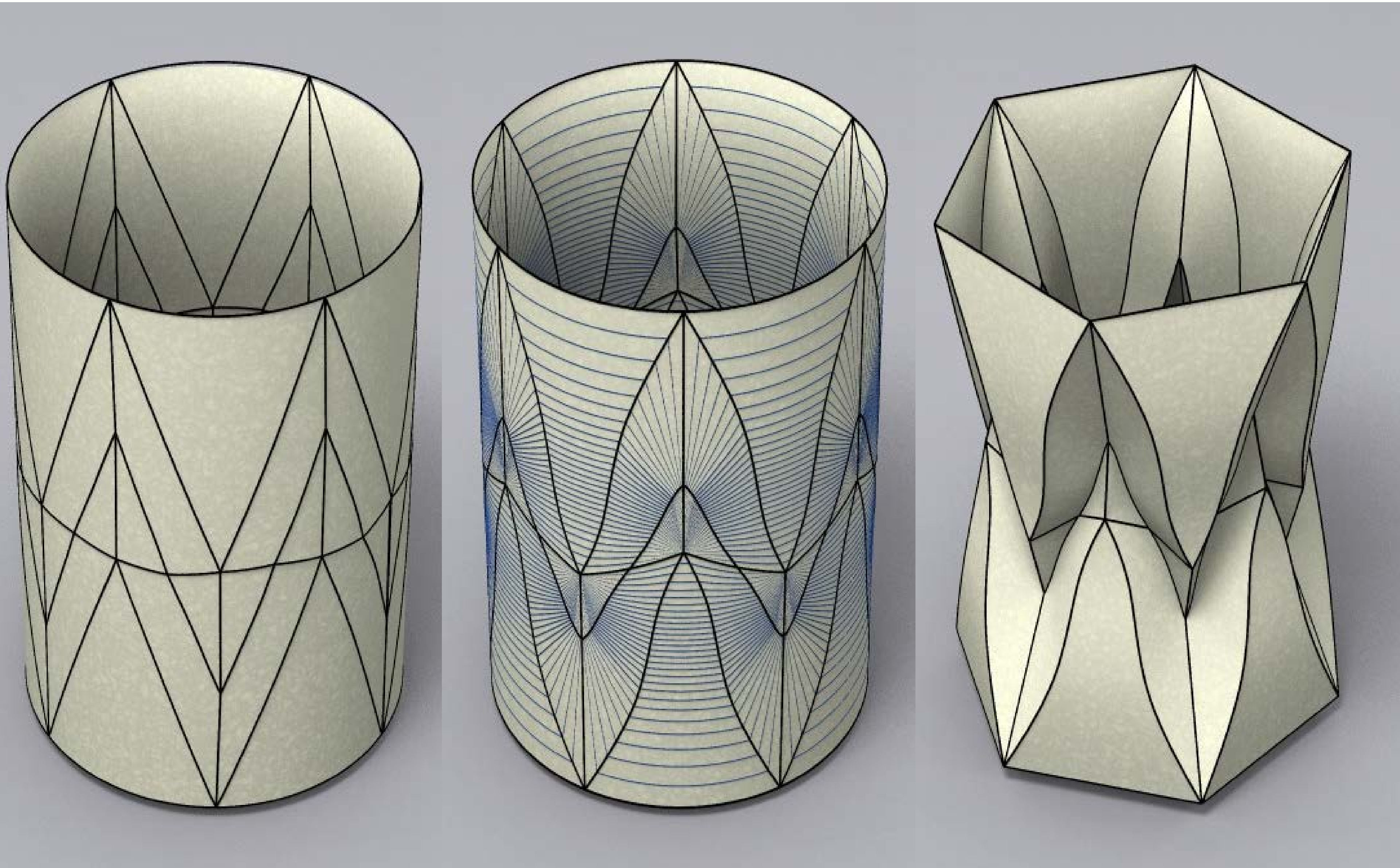
tangent surface of space curve

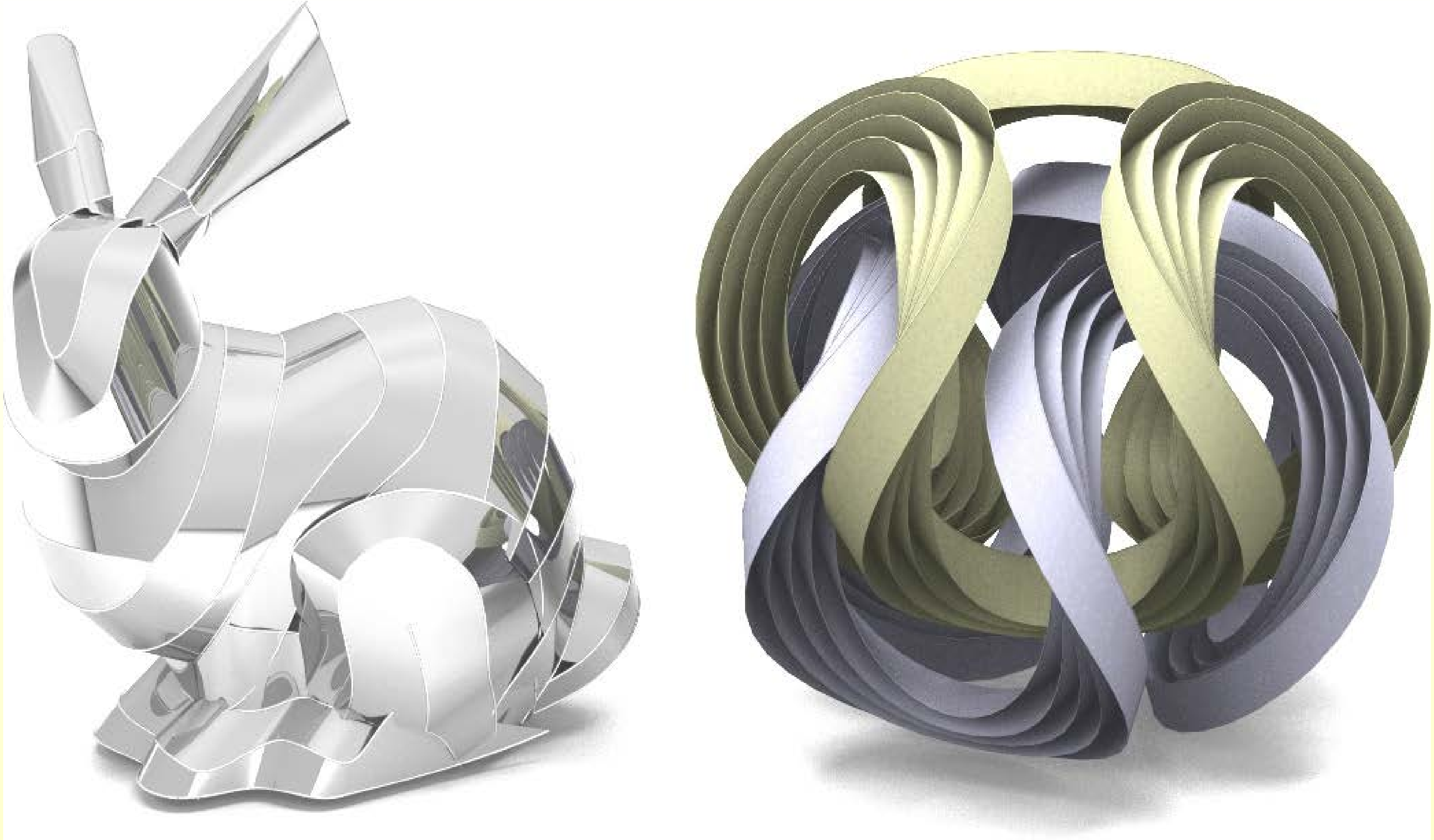
These surfaces can be unfolded into the plane without stretching or tearing

Curved folds

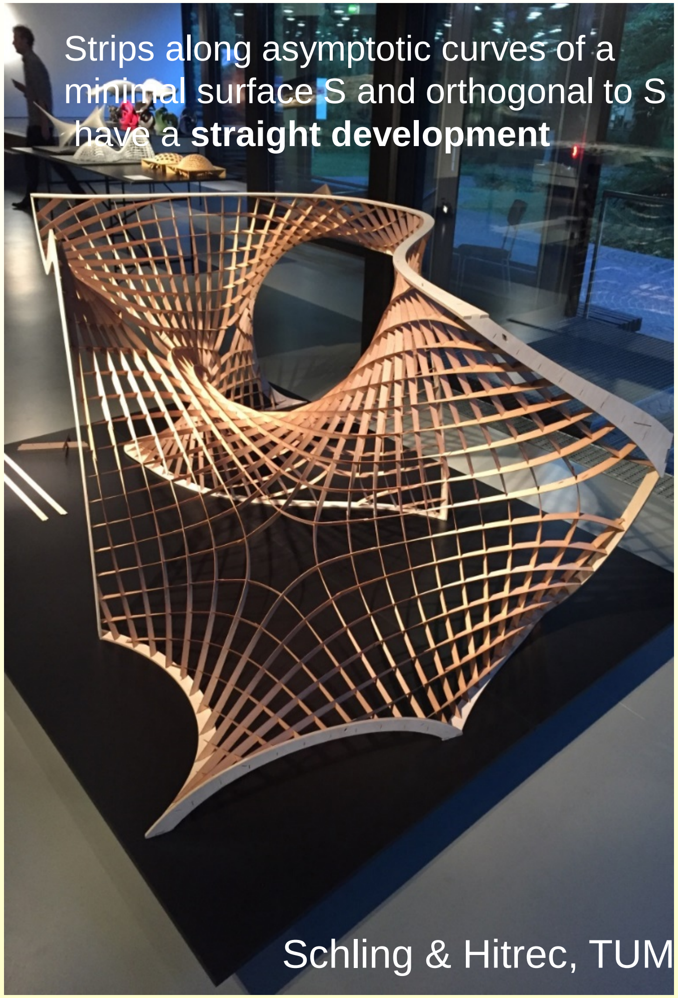
Proposition *In all points of a crease curve* **c** *in a developable surface, the osculating plane of* **c** *in the point* $\mathbf{c}(t)$ *is a bisector plane of the two tangent planes of the surface in the point* $\mathbf{c}(t)$.







Curved support structures



Architecture and Computational Design

- Architecture and Computational Design pose challenging mathematical problems
- Unexpected use of theoretical geometry and motivation for theoretical developments beyond the classical setting. Especially interesting: discrete differential geometry, geometric computing involving optimization
- Future research should develop novel computational design tools which combine shape modeling with aspects of function and fabrication