Geometric Modeling

Summer Semester 2010

Point-Based Modeling

3D Scanning · MLS-Surfaces · Reconstruction & Registration

UNIVERSITÄT DES SAARLANDES

M2CI CLUSTER OF EXCELLENCE

max planck institut informatik

Overview...

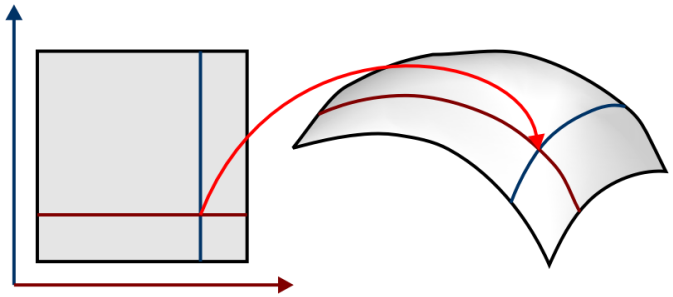
Topics:

- Subdivision Surfaces
- Implicit Functions
- Variational Modeling
- Point-Based Modeling
  - Introduction
  - 3D Acquisition Techniques
  - Data Processing Pipeline
  - Point Cloud Registration Algorithms
  - Moving-Least Squares Techniques
  - Point-Based Modeling

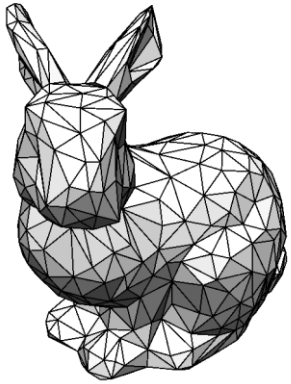
Point-Based Modeling

Introduction

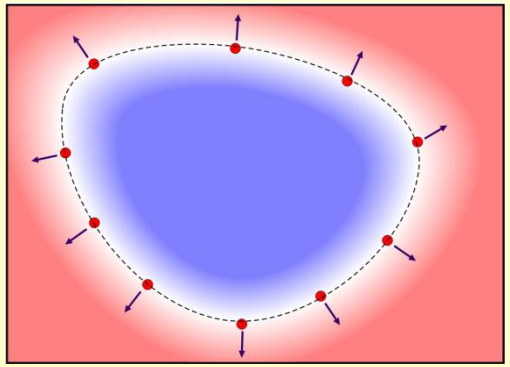
Modeling Zoo



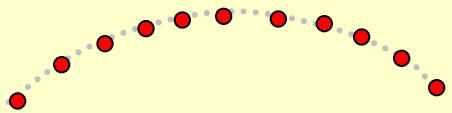
Parametric Models



Primitive Meshes



Implicit Models



Particle Models

3D Scanning

3D Scanning Devices:

- Typically based on point-wise distance measurement
- Almost all scanners output point clouds
- We need further processing to create a useful model
- 3D scanning is one of the main driving forces for “point-based modeling” research
  - Topology agnostic multi-resolution modeling is probably the other important one (e.g., rendering complex scenes like forests in real-time).

Problems

Point cloud (3D scanner data) related problems:

- Give a set of points, how does this define a continuous surface?

  $\Rightarrow$ *Surface reconstruction*

- How to assemble partial scans to a full model?

  $\Rightarrow$ *Surface registration*

- How to estimate normals, curvature, etc.?

  $\Rightarrow$ *Patch fitting, MLS*

- How to deal with noise & outliers?

  $\Rightarrow$ *Surface smoothing, outlier detection*

- Can we do modeling *just* with points?

Acquiring Point Clouds

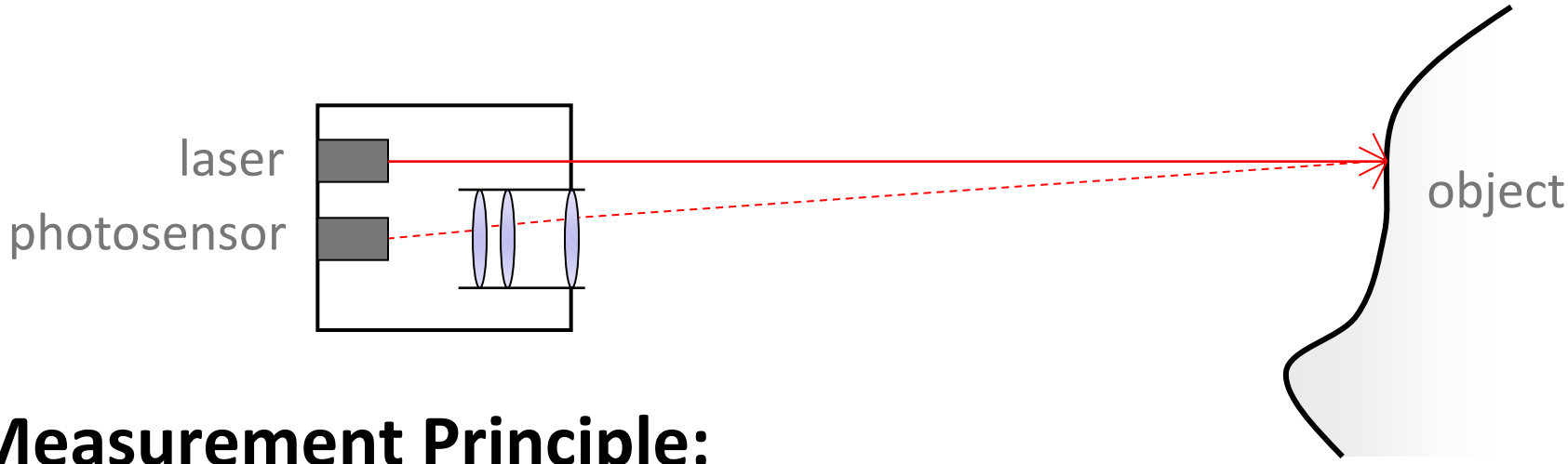
3D Scanners

Types of 3D Scanners

Scanning Techniques:

- Time-of-flight
  - Time-of-flight laser scanner
  - Time-of-flight depth cameras (dynamic)
- Triangulation
  - Laser line sweep
  - Structured light
- Stereo / computer vision
  - Passive stereo
  - Active stereo / space time stereo
  - Other techniques

Time of Flight Laser Scanner



Measurement Principle:

- Send out laser beam
  - Modulated at about 3-30 Mhz (phase length 10-100m)
- Measure phase difference with a photosensor (PLL)
  - Can resolve distances up to (modulo) phase length
  - Measures distance to a single point
- Application: Outdoor scanning, buildings, drive-by / fly-by scanning

Example Scans (Similar System)

Example Scans (Similar System)

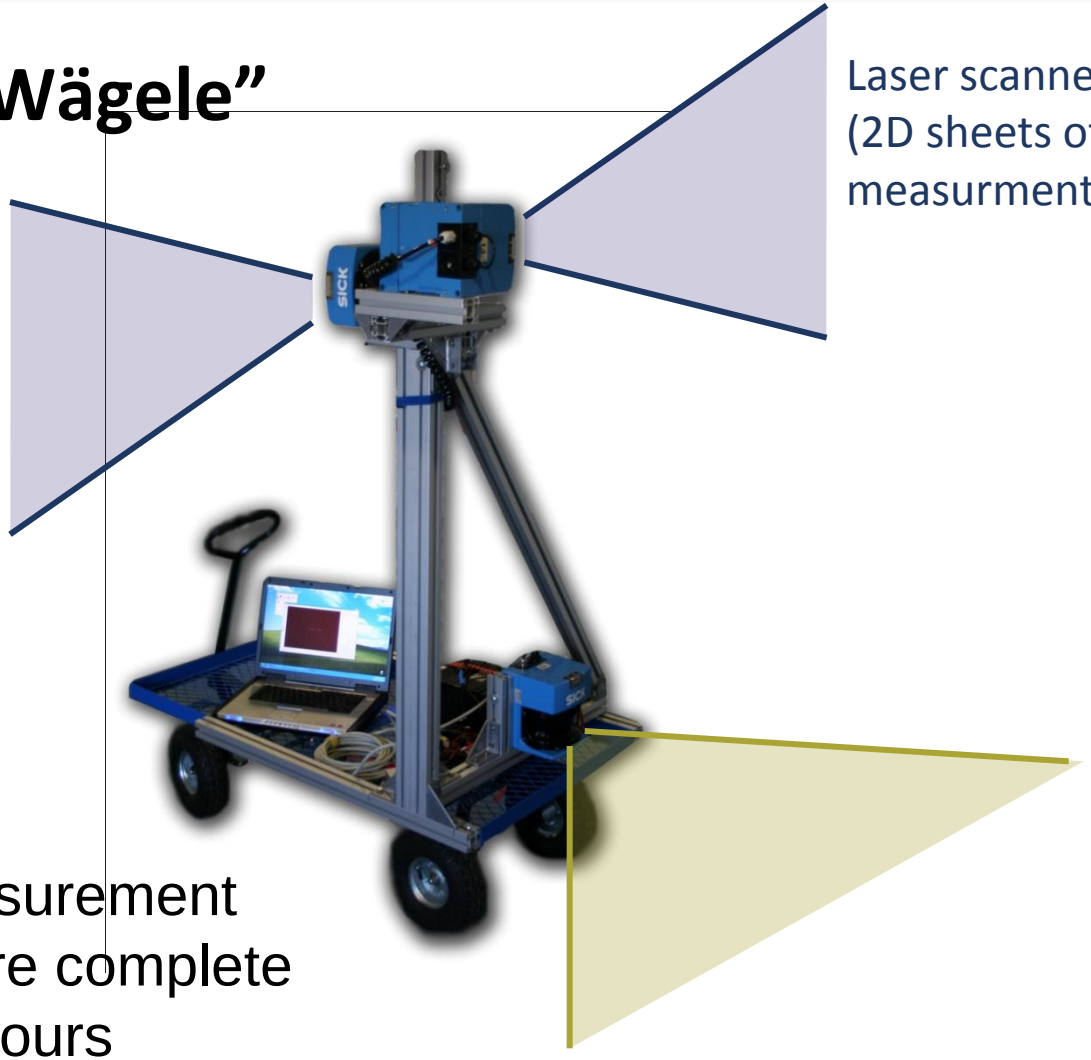
Acquisition Systems

Acquisition Systems:

- Rotating scanner head
  - Rotating mirror for vertical scanning (calibrated)
  - Rotating scanning head (incl. rot. mirror) for horizontal scanning
  - Mode of operation:
    - Position scanner
    - Push a button and wait a few minutes
    - A panoramic depth map is acquired
- Drive-by systems
  - 2D laser scanners (one rotating mirror)
  - Mounted on a vehicle with positioning system (GPS, rotation/acceleration sensors, aux. scanners)

Drive-by System

Example: The “Wägele”

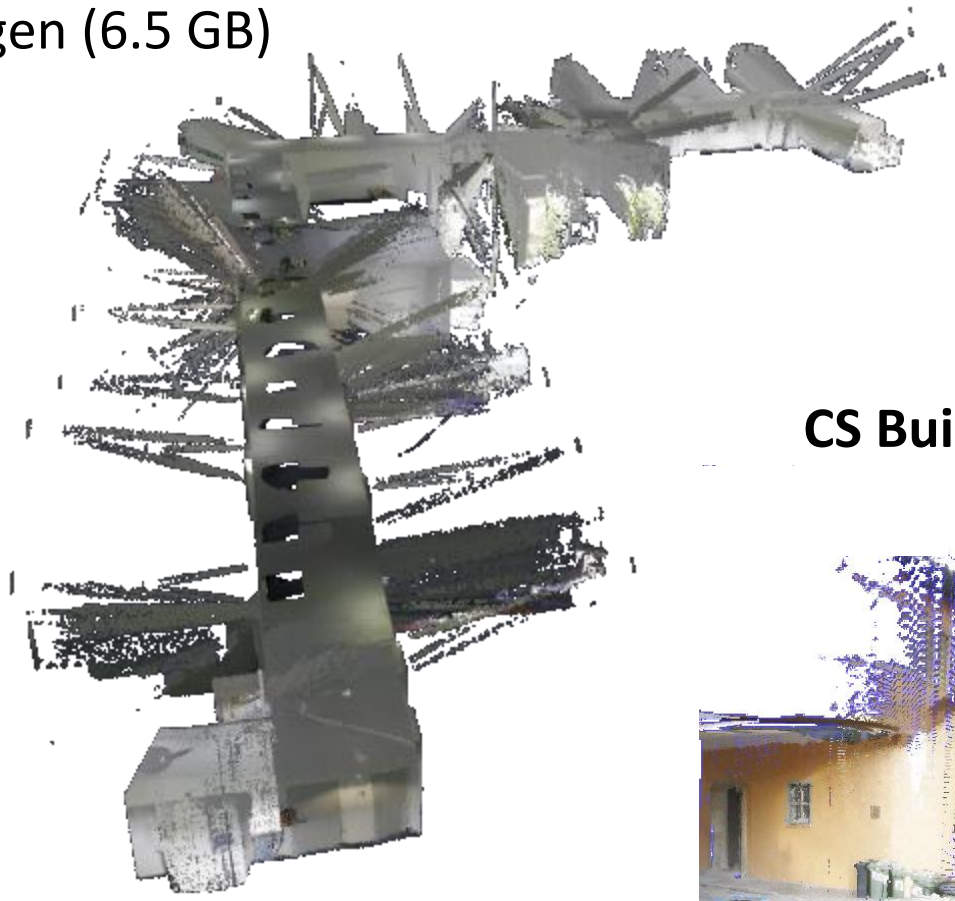


[Biber et al. 2005]

A pull-through measurement device – can acquire complete buildings in a few hours

This is what you get...

Corridor – CS Building
University of Tübingen (6.5 GB)



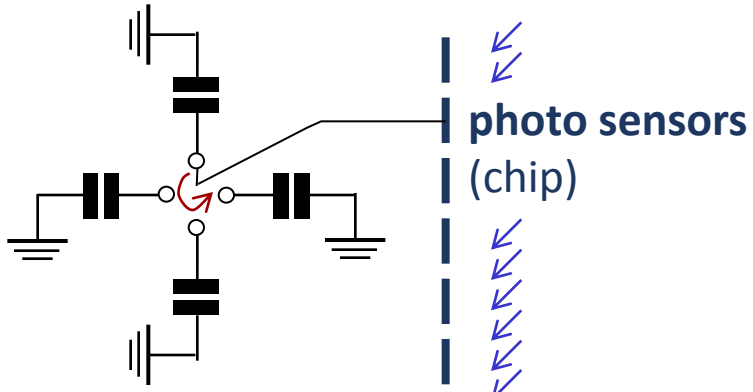
CS Building outdoors

Time of Flight Depth Cameras

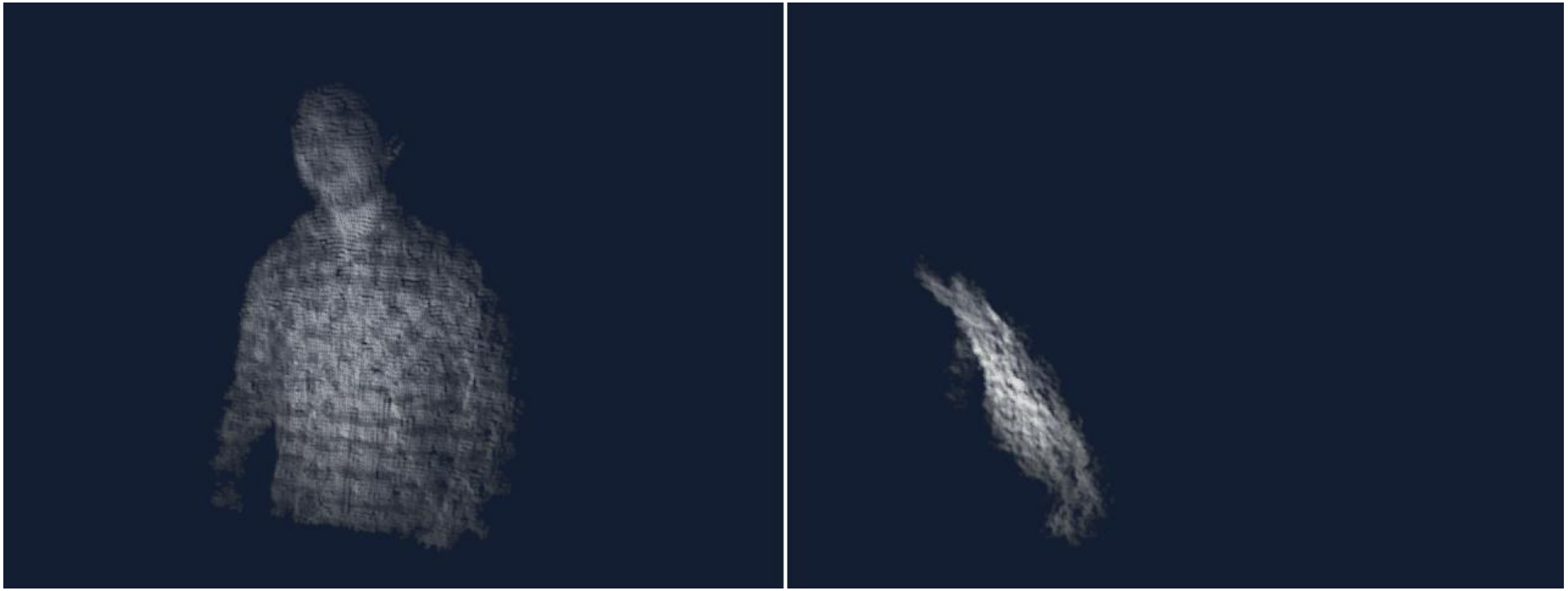
Real-time depth camera:

- Sends out modulated light (similar frequencies, O(Mhz))
- Measures phase in every pixel
- Acquire moving geometry in real-time
- Quality is much worse than static scans (lots of noise)

[PMD real-time time-of-flight camera]

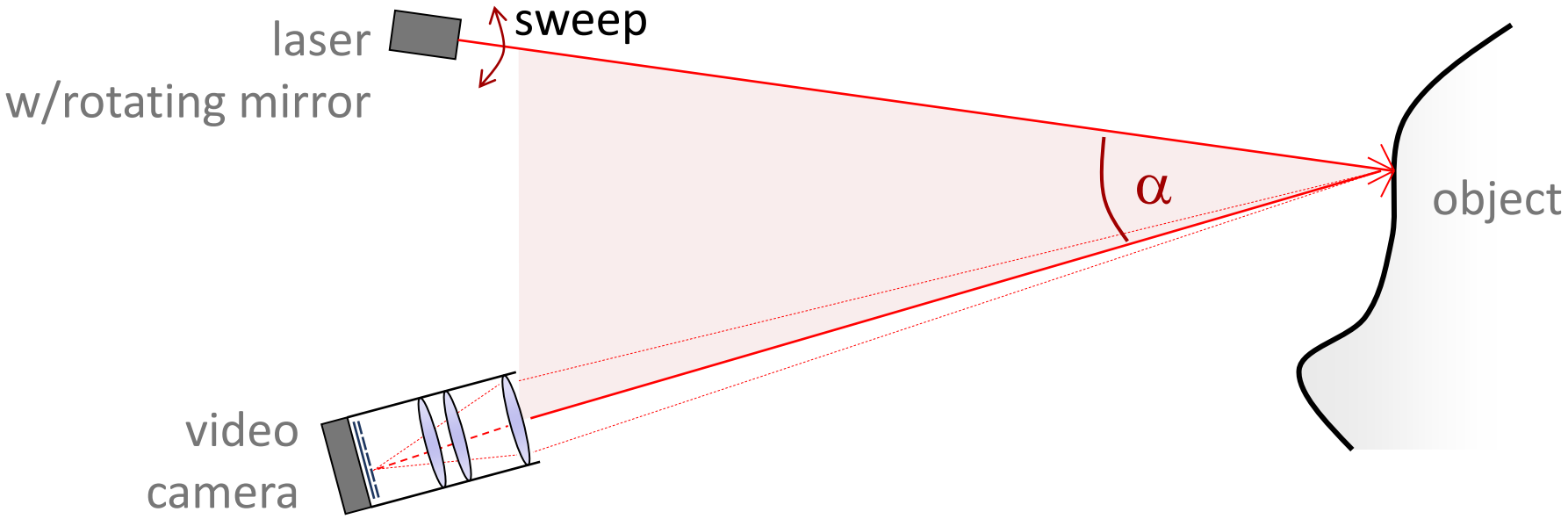


Example Scenes



“Swiss Ranger” Depth Camera

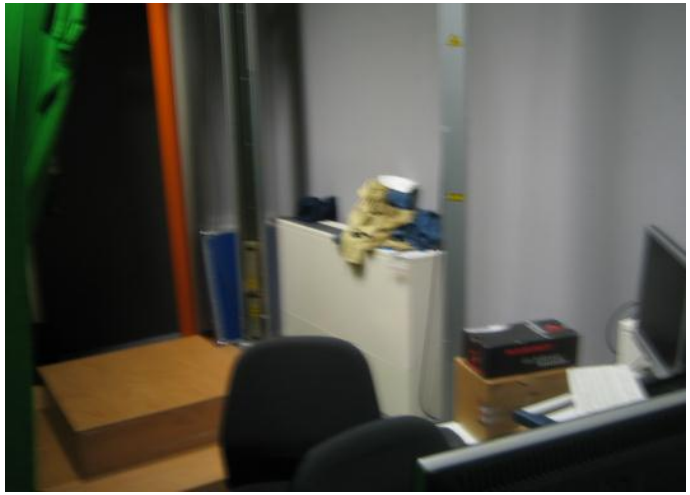
Triangulation Scanners



Measurement Principle (laser sheet scanners):

- Light the object with a light sheet
- View with camera from an angle
- We can compute the depth

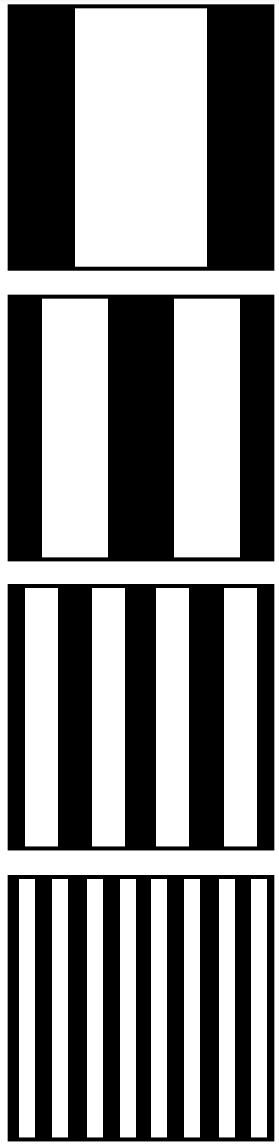
Example Device



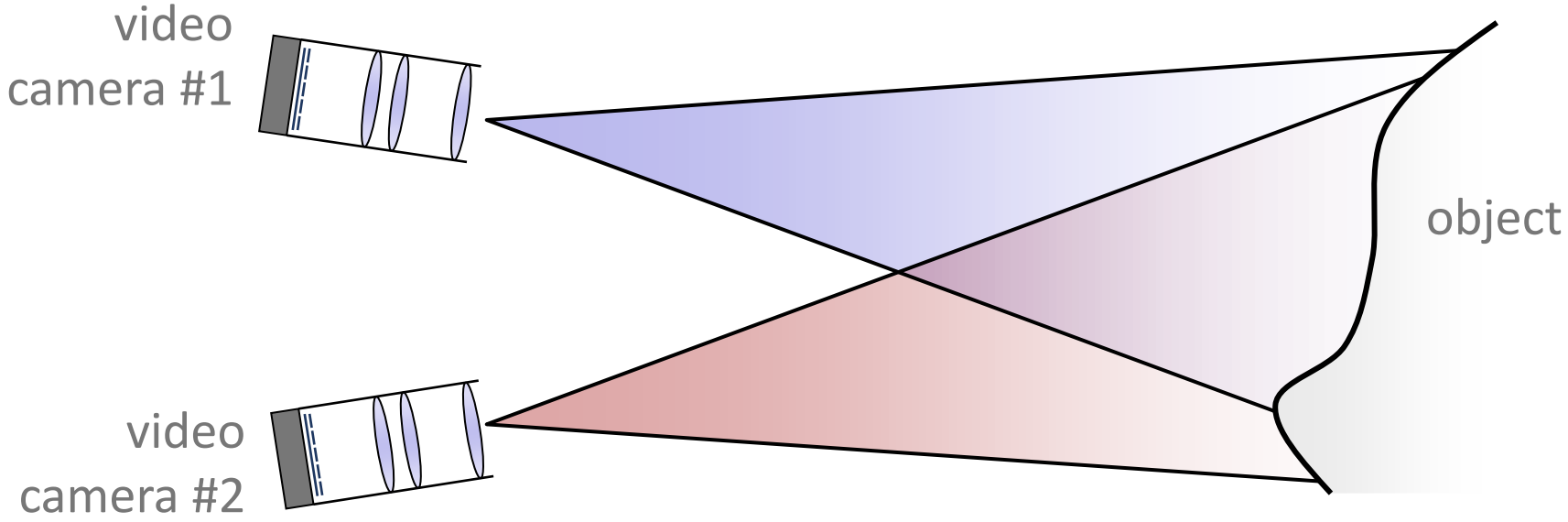
Structure Light Scanner

Idea:

- Replace laser by projector
- Project log(n) binary stripe codes instead of n light sheets
- Faster acquisition (exponential speedup)
  - Precision: Projector might be harder to focus
- Coding: Gray code
  - Any single bit error leads only to a shift by ± 1



Computer Vision Based Techniques



Stereo Matching

- Match points by similar color / shading
- Very general technique
- But: An inherently ill-posed problem
  - Typically bad reconstruction quality

Stereo Data

multi view matching (8 cameras)
(piecewise smooth variational surface
on presegmented images
solved with Bayesian belief propagation)

[Data set: Zitnick et al.,
Microsoft Research, Siggraph 2004]

multi view matching (6 cameras)
(photo-consistent space carving)

[Data set: Christan Theobald, MPII]

Improvement: Active Illumination

Stereo with active illumination:

- Project random pattern on the object
- Improves matching performance (more edges to match)
- “Space-Time Stereo”
  - Project a new random pattern each frame
  - Capture with two or more cameras
  - Gives good results, fully dynamic (animations)

Space Time Stereo

[Data set: James Davis, University of Santa Cruz]

[Davis et al. 2003]

Other Techniques

Other acquisition techniques:

- Computer vision:
  - Shape from shading
  - Shape from defocus
  - Shape from contours
  - Fluorescent fluid immersion scan (reflective / transparent objects)
- Other techniques:
  - Mechanical sampling
  - Radar (planes, satellites)

3D Scanner Point Cloud Processing

Data Processing Pipeline

Processing Pipeline

We get:

- A big cloud of sample points
  - Position, probably also color / laser intensity values
- Typically: A set of depth images

What we want in the end:

- A “nice” surface representation
- Typically: Triangle mesh

Processing Pipeline

Processing Pipeline:

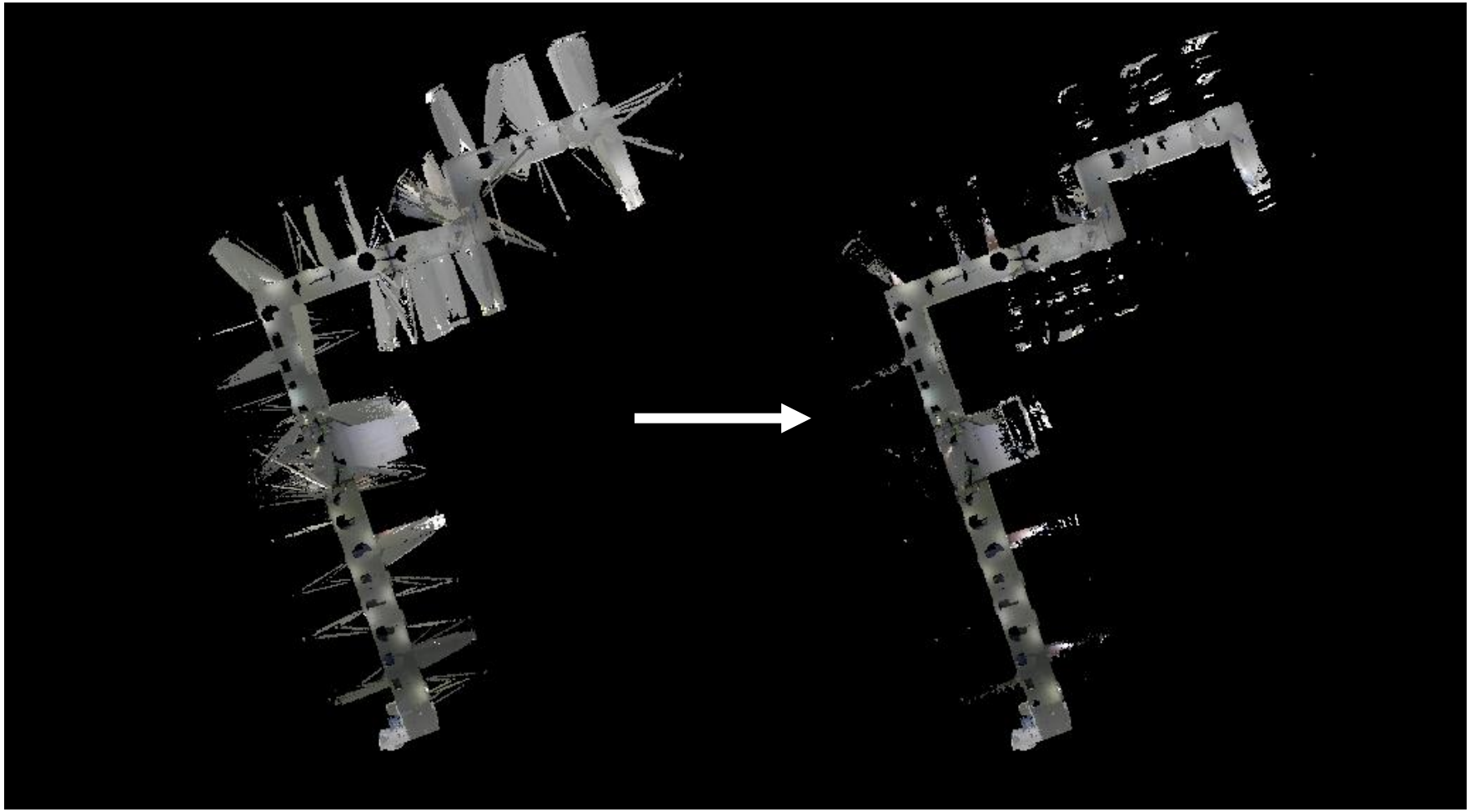
1. Outlier removal – throw away non-surface points (cause by scanner noise, dark surfaces, reflections etc.)
2. Registration – transform all scans into a common coordinate system
3. Surface smoothing – remove local noise
4. Normal direction estimation – needed for shading, reconstruction
5. Unify normal directions (maybe: look up depth images)
6. Surface reconstruction
   - Convert into triangle mesh
   - Alternatively: estimate sample spacing / resample and render points directly (for example tangential ellipsoid splats)

Processing Pipeline

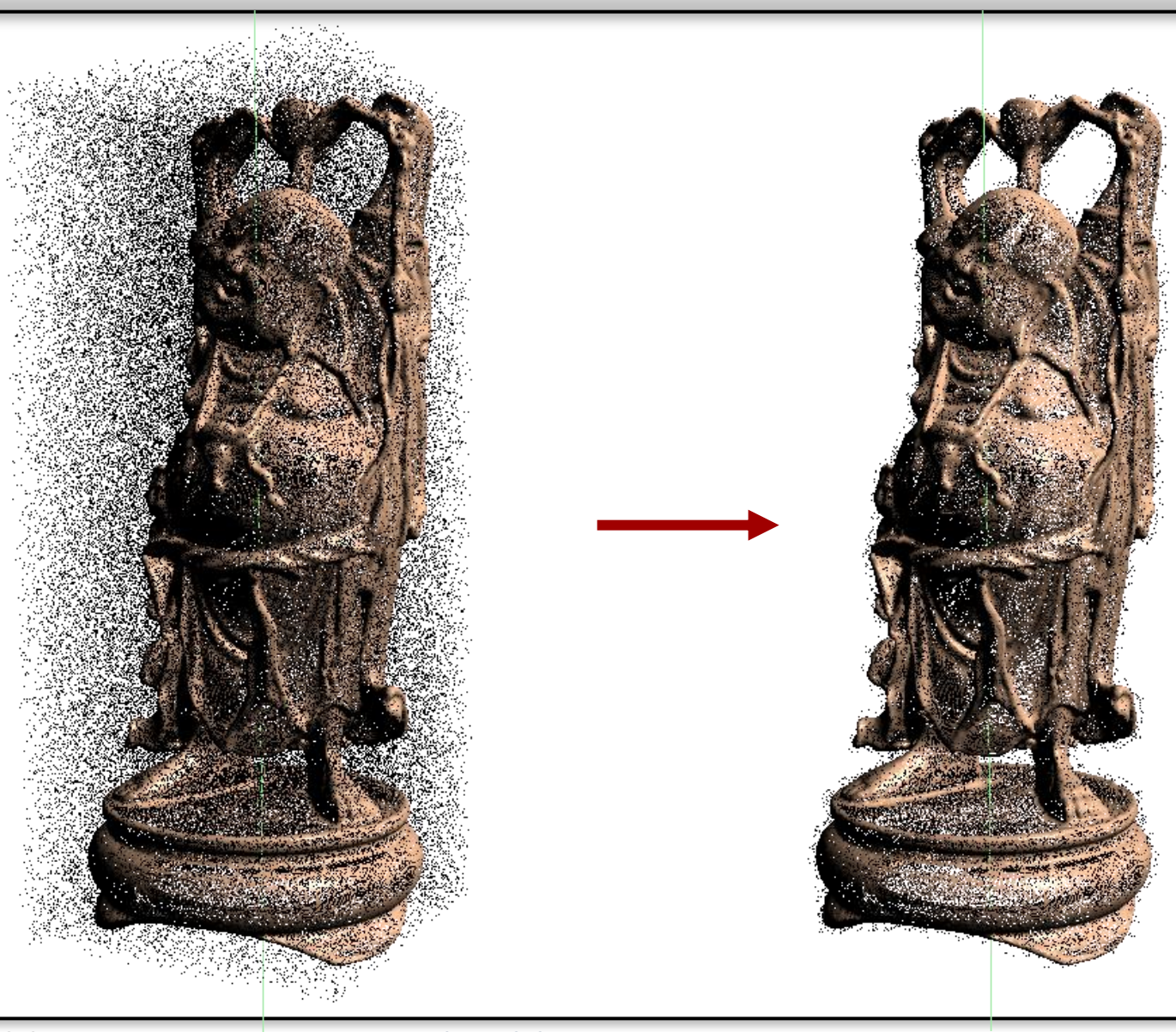
Processing Pipeline:

1. Outlier removal – throw away non-surface points (cause by scanner noise, dark surfaces, reflections etc.)
2. Registration – transform all scans into a common coordinate system
3. Surface smoothing – remove local noise
4. Normal direction estimation – needed for shading, reconstruction
5. Unify normal directions (maybe: look up depth images)
6. Surface reconstruction
   - Convert into triangle mesh
   - Alternatively: estimate sample spacing / resample and render points directly (for example tangential ellipsoid splats)

Automatic Outlier Removal



Automatic Outlier Removal

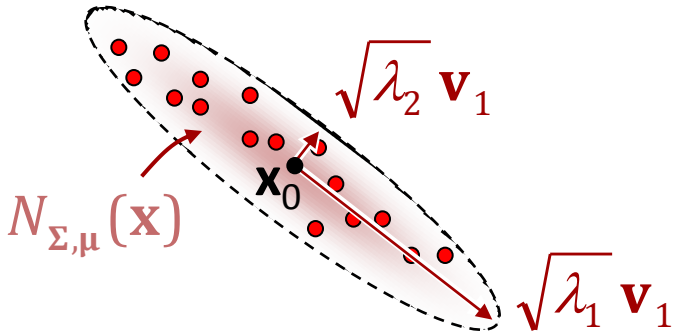


Algorithm

Very simple outlier removal algorithm:

- For each point compute its 20 nearest neighbors
- Compute the principal component analysis (plane fit with total least squares)
- If the *third* eigenvalue (normal direction) is larger than $1/(1+\varepsilon)$ times the *second* eigenvalue, delete the point as an outlier

PCA Plane Fitting (Recap)



$$\boldsymbol{\mu} = \mathbf{x}_0 = \frac{1}{n}\sum_{i=1}^{n} d_i$$

$$\tilde{\boldsymbol{\Sigma}} = \frac{1}{n-1}\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{d}_i - \mathbf{x}_0)(\mathbf{d}_i - \mathbf{x}_0)^{\mathrm{T}}$$

$$N_{\Sigma,\mu}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\det(\boldsymbol{\Sigma})^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

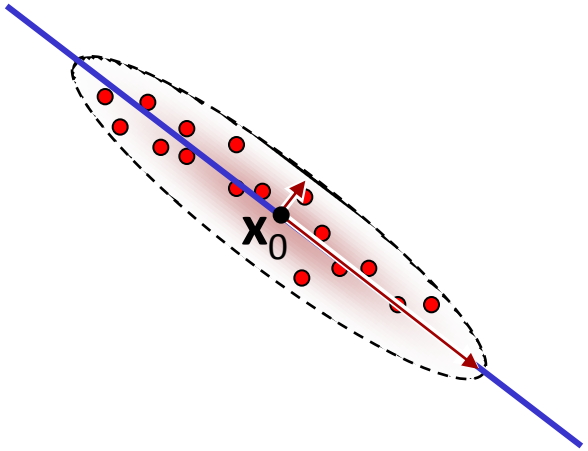
Reminder:

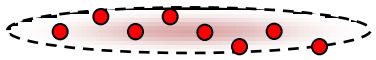
- PCA can be interpreted as fitting a Gaussian distribution and computing the main axes

PCA Plane Fitting (Recap)

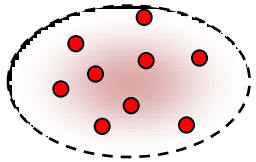
Plane Fitting in $\mathbb{R}^3$:

- Sample mean and the two directions of maximum eigenvalues
- Smallest eigenvalue
  - Eigenvector points in normal direction
  - Aspect ratio (λ_3 / λ_2) is a measure of "flatness" (quality of fit)
- Total least squares optimal *normal direction* (up to sign) given by eigenvector with smallest eigenvalue





(λ_2 / λ_1) small



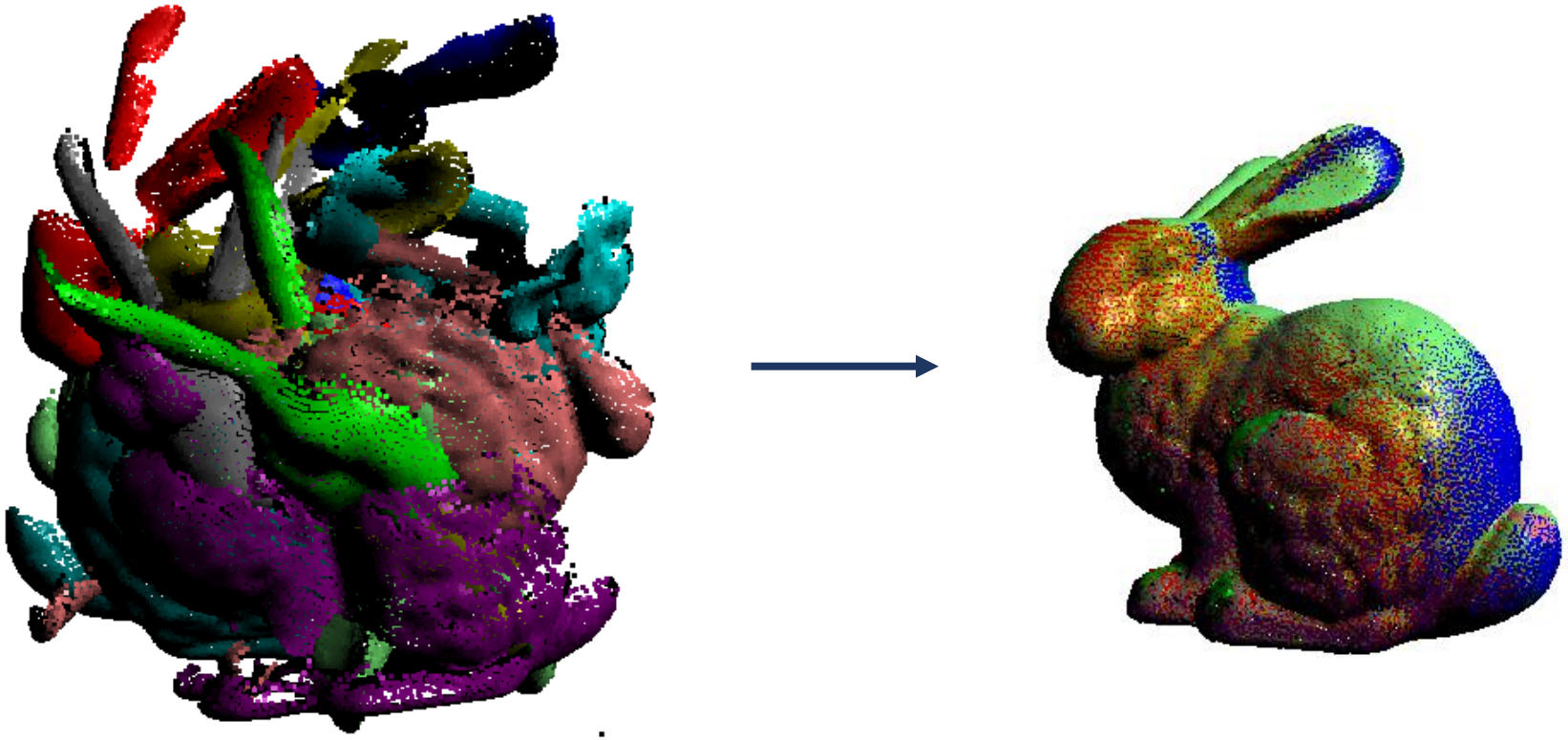
(λ_2 / λ_1) larger

Processing Pipeline

Processing Pipeline:

1. Outlier removal – throw away non-surface points (cause by scanner noise, dark surfaces, reflections etc.)
2. Registration – transform all scans into a common coordinate system
3. Surface smoothing – remove local noise
4. Normal direction estimation – needed for shading, reconstruction
5. Unify normal directions (maybe: look up depth images)
6. Surface reconstruction
   - Convert into triangle mesh
   - Alternatively: estimate sample spacing / resample and render points directly (for example tangential ellipsoid splats)

Surface Registration



more details on this later...

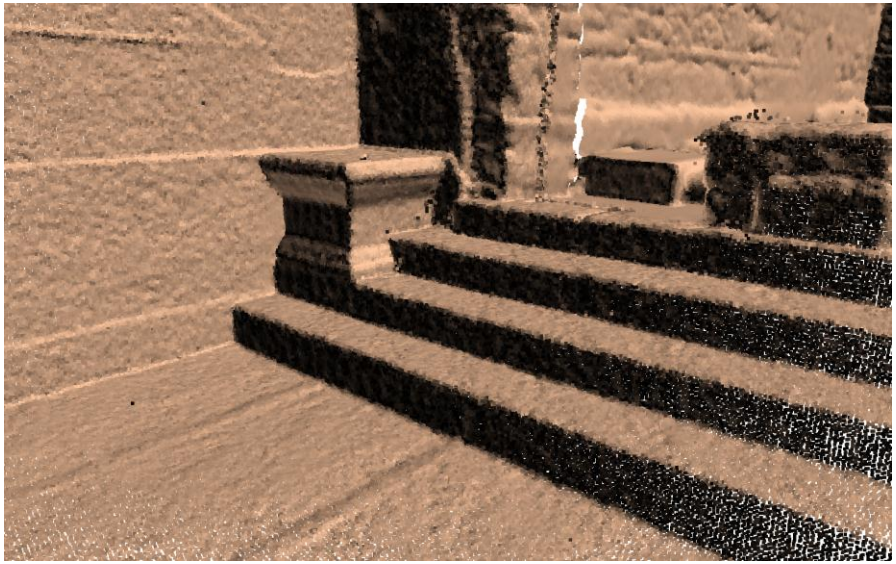
[Implementation: Martin Bokeloh (Diploma thesis)]

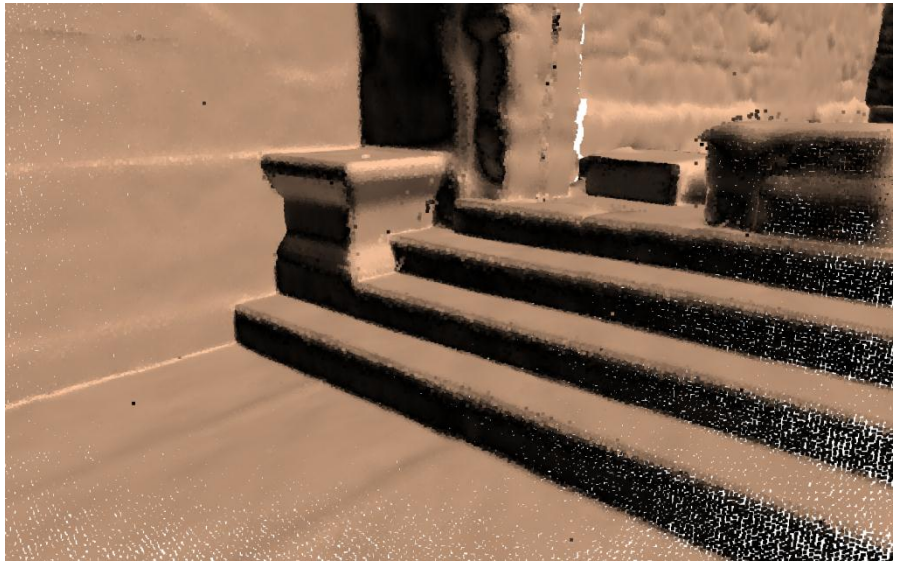
Processing Pipeline

Processing Pipeline:

1. Outlier removal – throw away non-surface points (cause by scanner noise, dark surfaces, reflections etc.)
2. Registration – transform all scans into a common coordinate system
3. Surface smoothing – remove local noise
4. Normal direction estimation – needed for shading, reconstruction
5. Unify normal directions (maybe: look up depth images)
6. Surface reconstruction
   - Convert into triangle mesh
   - Alternatively: estimate sample spacing / resample and render points directly (for example tangential ellipsoid splats)

Geometry Smoothing





Smoothing:

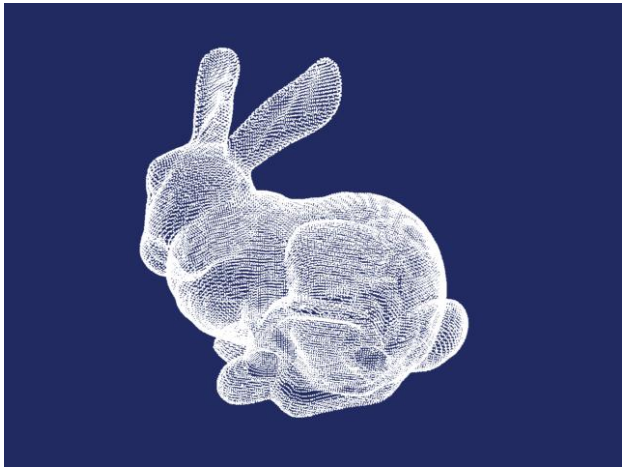
- This example: Bilateral geometry filter
- Removes noise while preserving sharp features
- More details on this later (MLS surface reconstruction)...

Processing Pipeline

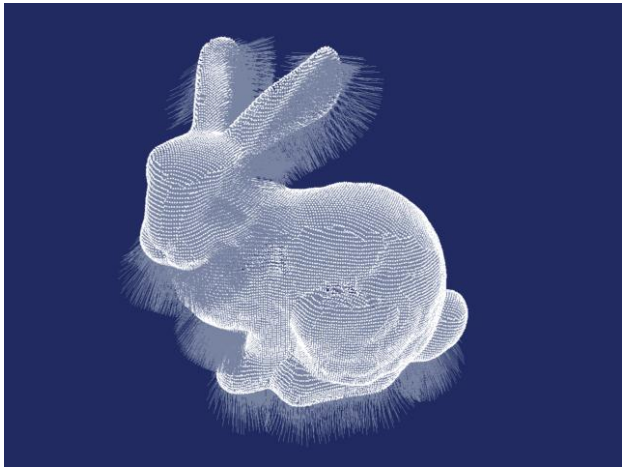
Processing Pipeline:

1. Outlier removal – throw away non-surface points (cause by scanner noise, dark surfaces, reflections etc.)
2. Registration – transform all scans into a common coordinate system
3. Surface smoothing – remove local noise
4. Normal direction estimation – needed for shading, reconstruction
5. Unify normal directions (maybe: look up depth images)
6. Surface reconstruction
   - Convert into triangle mesh
   - Alternatively: estimate sample spacing / resample and render points directly (for example tangential ellipsoid splats)

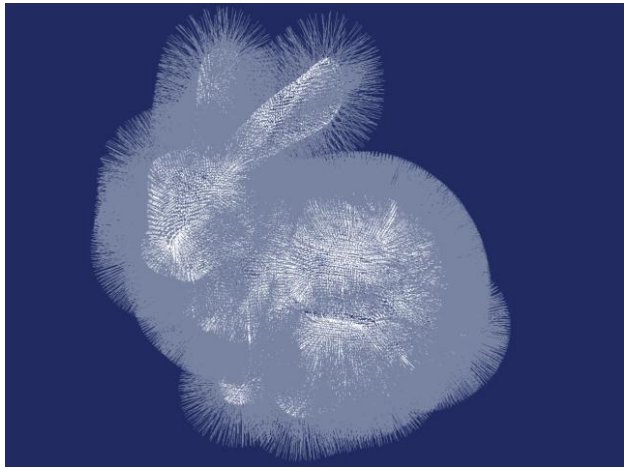
Normals for the Bunny...



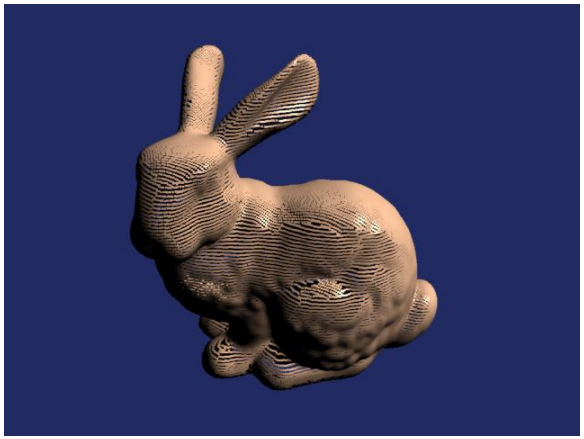
original point cloud



PCA normals
(k=20 nearest neighbors)



unified normals
(region growing)



◂ **final shading**

Processing Pipeline

Processing Pipeline:

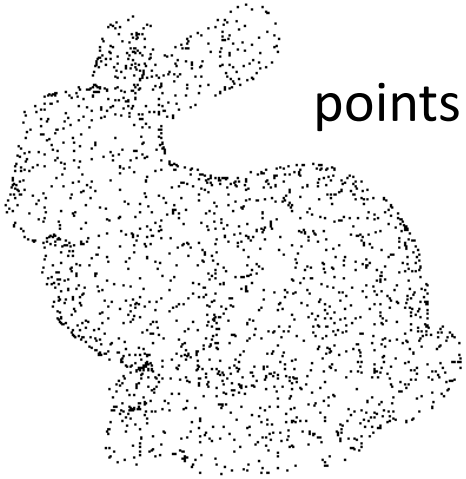
1. Outlier removal – throw away non-surface points (cause by scanner noise, dark surfaces, reflections etc.)
2. Registration – transform all scans into a common coordinate system
3. Surface smoothing – remove local noise
4. Normal direction estimation – needed for shading, reconstruction
5. Unify normal directions (maybe: look up depth images)
6. Surface reconstruction
   - Convert into triangle mesh
   - Alternatively: estimate sample spacing / resample and render points directly (for example tangential ellipsoid splats)

Surface Reconstruction

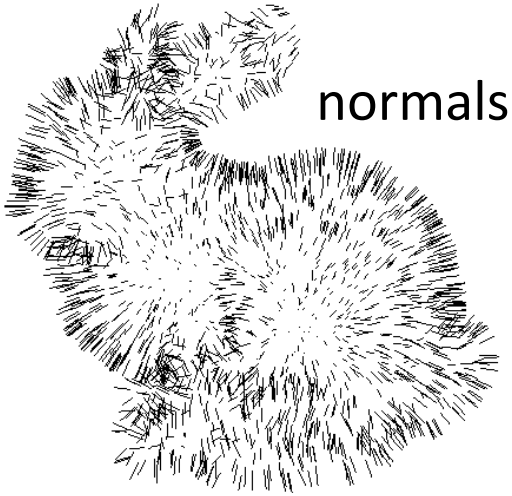
Reconstructing Triangle Meshes:

- Implicit methods
  - Fit an implicit surface
  - Use marching cubes
  - Postprocessing: Mesh simplification
- “Moving least squares (MLS)”
  - Special case of implicit surface fitting
  - more on this later...
- Voronoi methods
  - Compute Delaunay tetrahedrization
  - Filter out surface triangles (pole analysis)

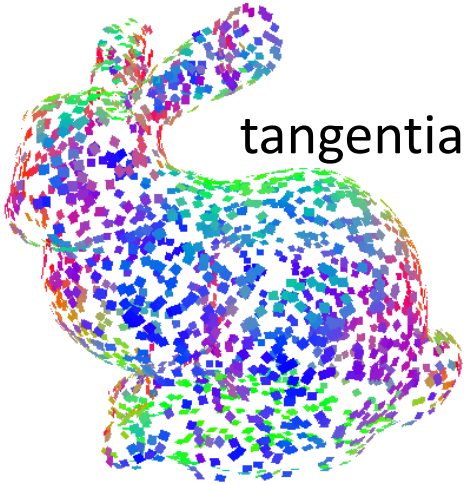
Direct Point Splatting



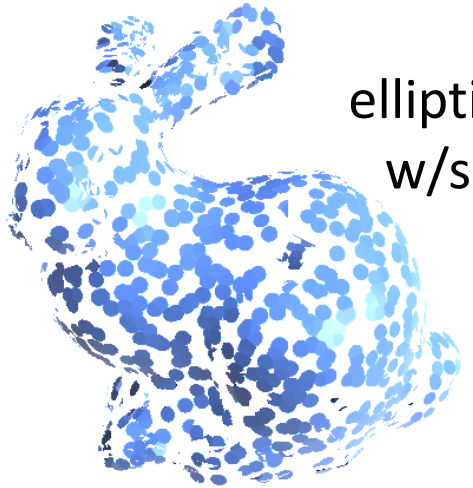
points



normals



tangential frames



elliptic splats w/shading

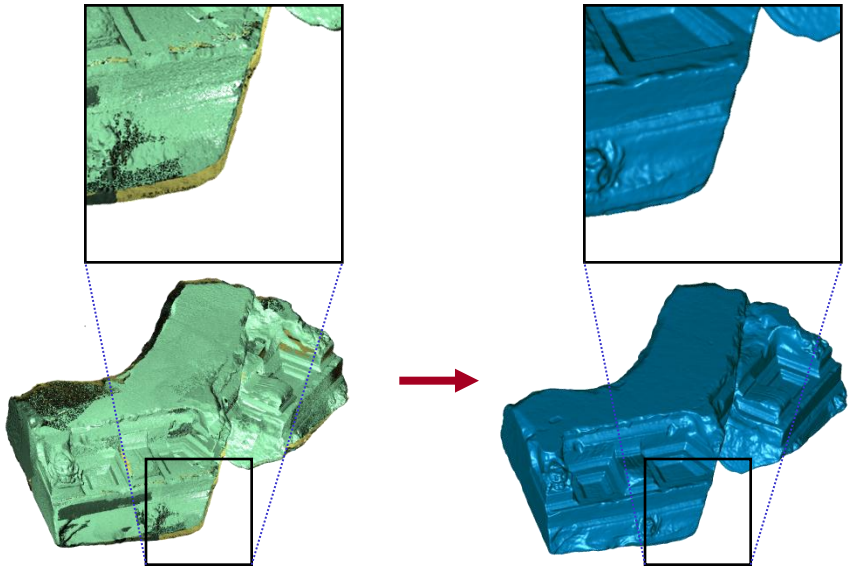
Surface Registration

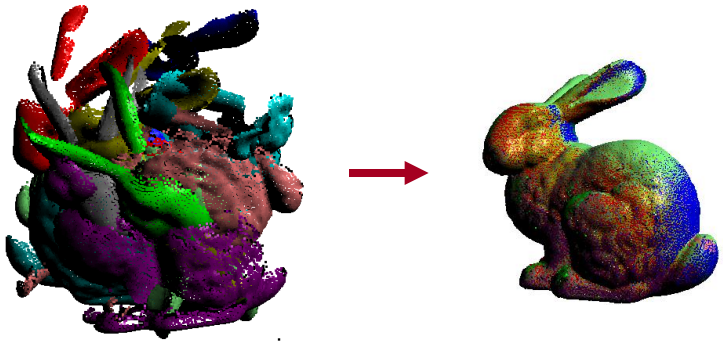
Point Cloud Matching

Point Cloud Matching

Two problems:

- Local matching
  - The individual scans are already roughly aligned
  - Need to optimize the alignment ("*snap in*")
  - Non-linear optimization
- Global matching
  - No initial alignment is known
  - We need to solve the problem *globally* (unconditional convergence)





Point Cloud Matching

Two problems:

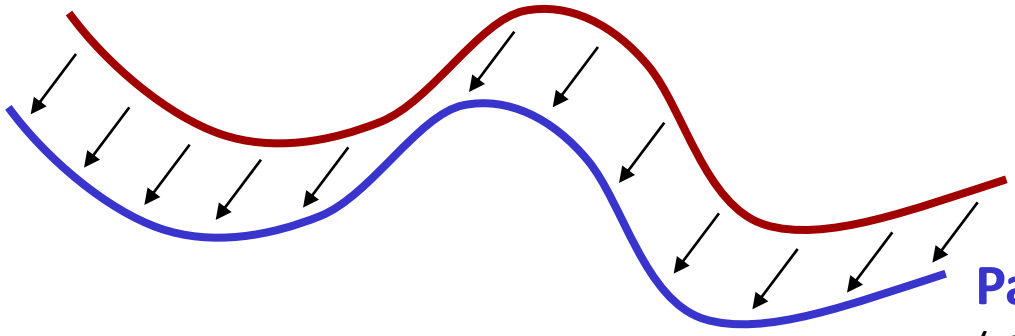
- Often two steps:
  - Global matching yields only a rough alignment
  - Followed by local alignment to compute accurate solution

Local Matching Algorithms

Local Matching Algorithms

- The standard algorithm: Iterated Closest Points (ICP)
  - Standard algorithm is easy to understand
  - Not too hard to implement (need some data structures)
  - Many variants to improve convergence speed and reliability
- Deformable ICP:
  - Allows deformations during matching
  - Compensate scanner calibration errors
  - Deformable matching
    - Tracking real-time animation scans (correspondences)
- Other techniques: for example NDT (normal distribution transform, useful for real-time applications)

Iterated Closest Points (ICP)



The main idea:

- Pairwise matching technique
  - Registers two scans
  - Multi-part matching is a different story (more on this later)
- We want to minimize the distance between the two parts
  - We set up a variational problem
  - Minimize distance “energy” by rigid motion of one part

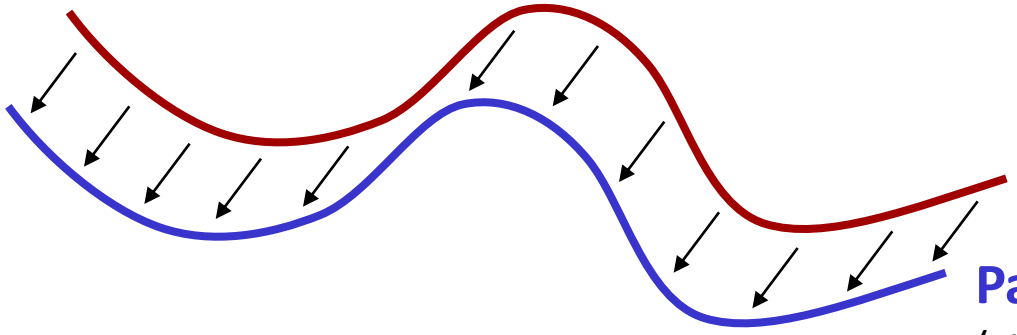
Iterated Closest Points (ICP)

Problem:

- How to compute the distance
- This is simple if we know the *corresponding points*.
  - Of course, we have in general no idea of what corresponds...
- ICP-idea: set closest point as corresponding point
- Full algorithm:
  - Compute closest point points
  - Minimize distance to these closest points by a rigid motion
  - Recompute new closest points and iterate

Closest Points

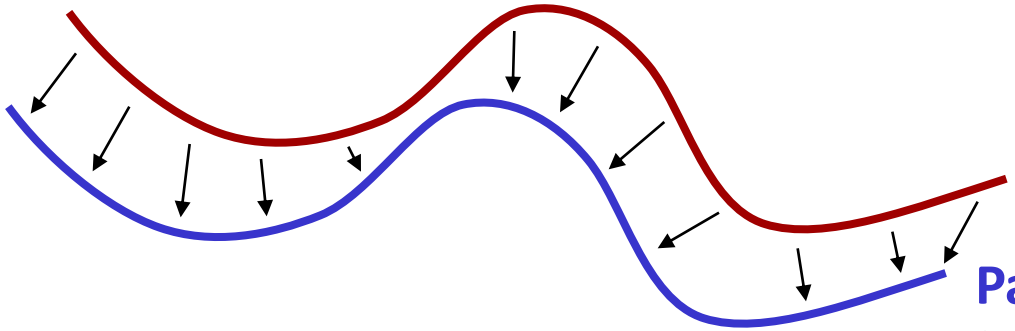
Distances:



Part B
(moves, rotation & translation)

Part A
(stays fixed)

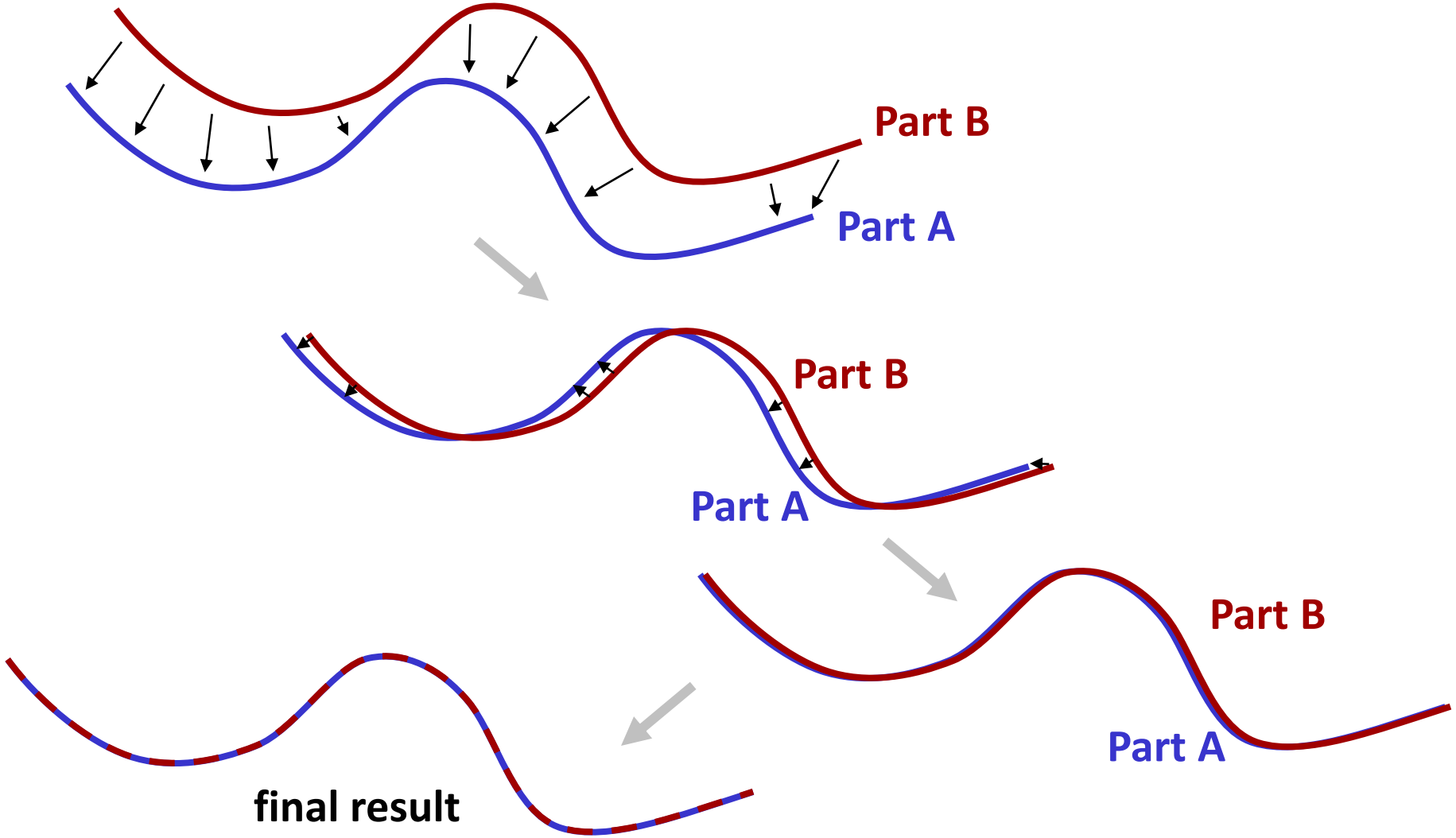
Closest points distances:



Part B
(moves, rotation & translation)

Part A
(stays fixed)

Iteration



Variational Formulation

Variational Formulation:

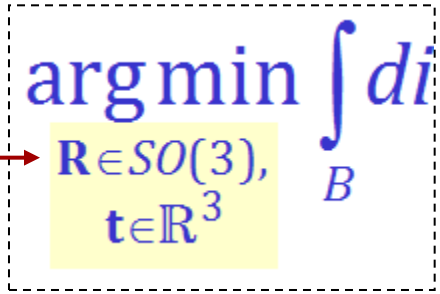
$$\underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \int_B dist(\mathbf{Rx}+\mathbf{t}, A)^2\, d\mathbf{x} \;\approx\; \underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \sum_{i=1}^{n} \left(\mathbf{R}\mathbf{p}_i^{(A)} + \mathbf{t} - \mathbf{p}_{nearest(i)}^{(B)}\right)^2$$

Variables: Orthogonal matrix **R**, translation vector **t**

Numerical Solution

Question: How to minimize this energy?

- The energy is quadratic
- There is only one problem...
  - Constraint optimization
  - We have to use an orthogonal matrix...
- This problem can (still) be solved exactly.



Solution

First step: computing the translation

- Easy to see: average translation is optimal (c.f. total least squares)
- $\mathbf{t} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i^{(A)} - \mathbf{p}_{nearest(i)}^{(B)}$
- This is independent of the rotation

Second step: compute the rotation

- (2a) Compute optimal linear map
- (2b) Orthogonalize

Optimal Linear Map

First:

- Subtract translation from points $\tilde{\mathbf{p}}_i^{(A)} = \mathbf{p}_i^{(A)} - \mathbf{t}$
- Then: Solve an unconstrained least-squares problem

$$\forall i = 1..n : \mathbf{M}\tilde{\mathbf{p}}_i^{(A)} = \mathbf{p}_{nearest(i)}^{(B)}$$

unknowns
(9 variables)

$$\forall i = 1..n : \begin{pmatrix} m_{1,1} & m_{2,1} & m_{3,1} \\ m_{1,2} & m_{2,2} & m_{3,2} \\ m_{1,3} & m_{2,3} & m_{3,3} \end{pmatrix} \tilde{\mathbf{p}}_i^{(A)} = \mathbf{p}_{nearest(i)}^{(B)}$$

- Finally: compute the orthogonal matrix **R** that is closest to **M**.

Least-Squares Optimal Rotation

How to compute a least-squares (Frobenius norm) orthogonal matrix that fits a general matrix:

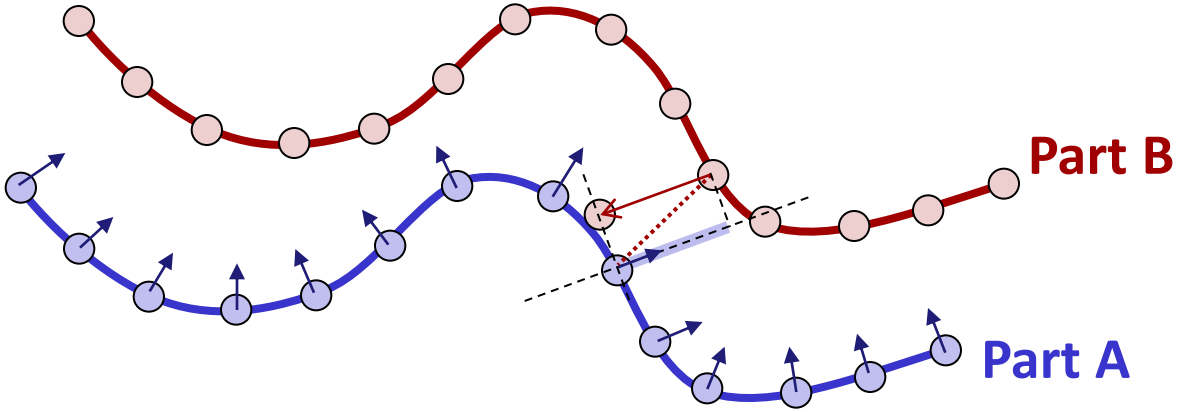
- Compute the SVD: $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^T$
- The least-squares orthogonal fit is: $\mathbf{R} = \mathbf{U}\mathbf{V}^T$
  (just set all singular values to one)
- We can compute this in one step:
  - Solve the least-squares matrix fitting problem using SVD
  - Omit the diagonal matrix straight ahead

Generalizations

Convergence speed:

- Convergence of basic "*point-to-point*" ICP is not so great
  - Typically: 20-50 iterations for simple examples
  - Problem: Zero-th order method
    (flip point correspondences in each step)
- Improvement: "*point-to-plane*" ICP
  - First order approximation
  - Match points to tangential planes rather than points
  - Converges much faster (3-5 iterations for similar examples)

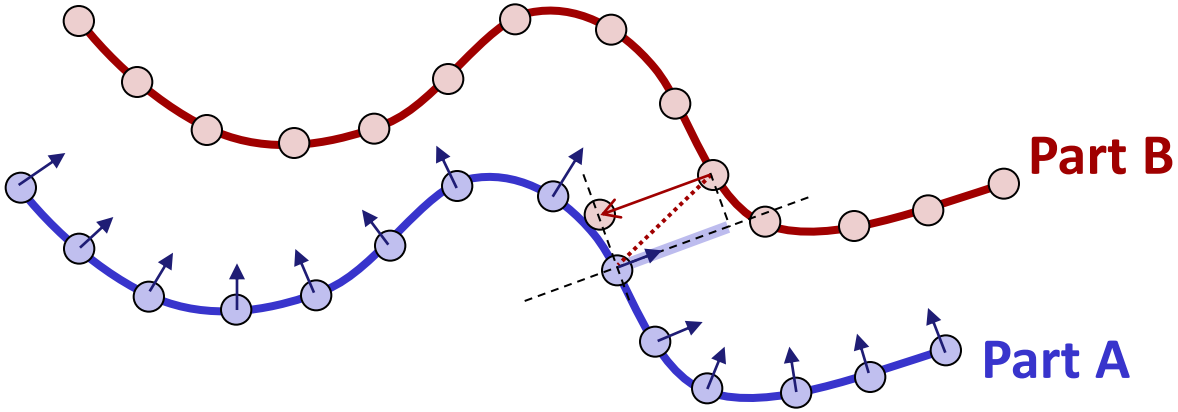
Implementation



$$\underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \int_B \left\langle \mathbf{Rx}+\mathbf{t}-nearest(A), \mathbf{n}(nearest(A)) \right\rangle^2 d\mathbf{x}$$

$$\approx \underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \sum_{i=1}^{n} \left\langle \mathbf{R}\mathbf{p}_i^{(A)}+\mathbf{t}-\mathbf{p}_{nearest(i)}^{(B)}, \mathbf{n}_{nearest(i)}^{(B)} \right\rangle^2$$

Implementation



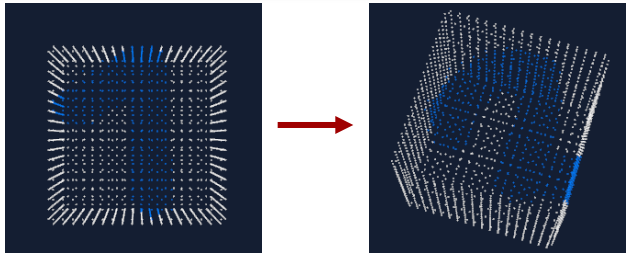
Implementation:

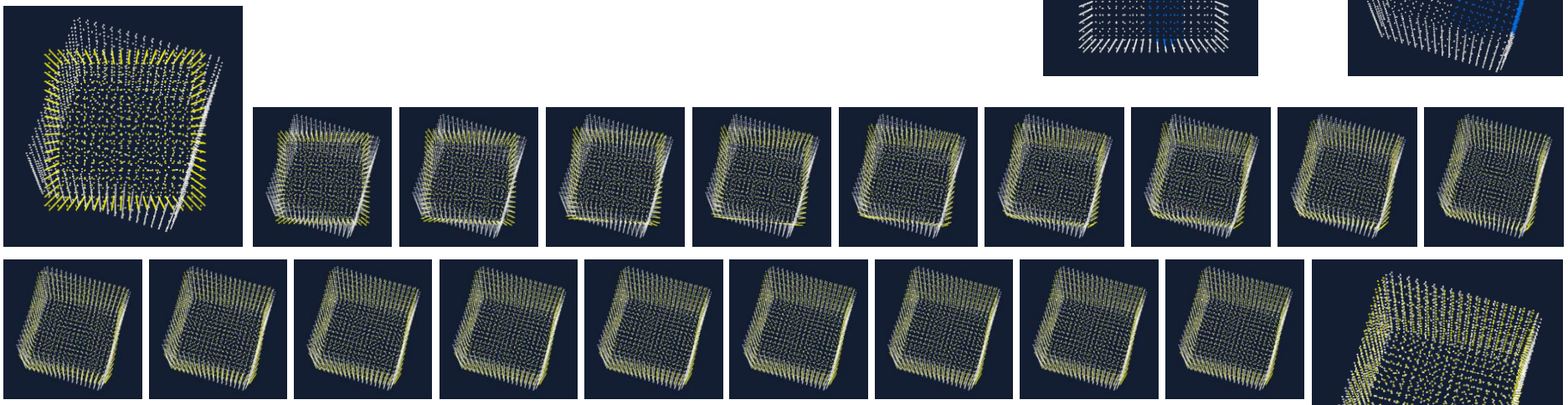
- We need normals for each point (unoriented) → kNN+PCA
- Compute closest point, project distance vector to its normal
- Minimize the sum of all such distances:

$$\underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \sum_{i=1}^{n} \left\langle \mathbf{R}\mathbf{p}_i^{(A)} + \mathbf{t} - \mathbf{p}_{nearest(i)}^{(B)}, \mathbf{n}_{nearest(i)}^{(B)} \right\rangle^2$$

Comparison

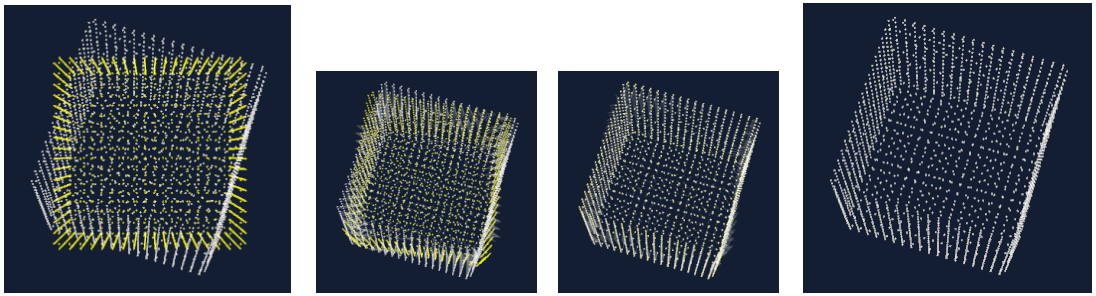
Point-to-point: 19 iterations





(accuracy problems)

Point-to-plane: 3 iterations



(much more accurate result)

Implementation

Problem:

- No closed form solution for the optimal rotation with point-to-plane correspondences

Solution:

- Numerical solution
- Setup non-linear optimization problem (rotation, translation = 6 parameters)
- Use non-linear optimization technique
- Remaining problem: *Parametrization of the rotations*
  - Trouble with singularities (spherical topology)

Local Linearization

Standard technique: local linearization

- Transformation: **T**(**x**) = **R**x + **t**
- Linearize rotations:

$$T_{\alpha,\beta,\gamma} = \begin{pmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & \sin(\gamma) \\ 0 & -\sin(\gamma) & \cos(\gamma) \end{pmatrix}$$

$$\nabla T_{\alpha,\beta,\gamma} = \begin{pmatrix} 0 & \alpha & \beta \\ -\alpha & 0 & \gamma \\ -\beta & -y & 0 \end{pmatrix}$$

$$\Rightarrow T_{\alpha,\beta,\gamma}(x) \approx \mathbf{x} + \nabla T_{\alpha,\beta,\gamma}\mathbf{x} = \left(\begin{pmatrix} 0 & \alpha & \beta \\ -\alpha & 0 & \gamma \\ -\beta & -y & 1 \end{pmatrix} + \mathbf{I} \right) \mathbf{x}$$

Local Linearization

Standard technique: local linearization

- Numerical solution: iterative solver
- We have a current rotation $\mathbf{R}^{(i-1)}$ from the last iteration:
- Taylor expension at $\mathbf{R}^{(i-1)}$:

$$T^{(i)}_{\alpha,\beta,\gamma}(\mathbf{x}) \approx \left(\begin{pmatrix} 0 & \alpha & \beta \\ -\alpha & 0 & \gamma \\ -\beta & -y & 1 \end{pmatrix} + \mathbf{I} \right) \mathbf{R}^{(i-1)}\mathbf{x}$$

- Solve for $\mathbf{t}, \alpha, \beta, \gamma$ (linear expressison $\rightarrow$ quadratic opt.)

$$\underset{\substack{\alpha,\beta,\gamma \in \mathbb{R} \\ \mathbf{t} \in \mathbb{R}^3}}{\arg\min} \sum_{j=1}^{n} \left\langle \mathbf{R}^{(i)}\mathbf{p}^{(A)}_j + \mathbf{t} - \mathbf{p}^{(B)}_{nearest(j)}, \mathbf{n}^{(B)}_{nearest(j)} \right\rangle^2$$

Local Linearization

Then:

- Project $\mathbf{R}^{(i)}$ back on the manifold of orthogonal matrices. (for example using the SVD-based algorithm discussed before)
- Then iterate, until convergence.

Why does this work?

- The parametrization is non-degenerate
  - For large α, β, γ, the norm of the matrix increases arbitrarily (i.e.: the object size increases, away from the data)
  - Therefore, the least-squares optimization will perform a number of small steps rather than collapse.

More Tricks & Tweaks

ICP Problems:

- Partial matching might lead to distortions / bias
  - Remove outliers (M-estimator, delete “far away points”, e.g. 20% percentile in point-to-point distance)
  - Remove normal outliers
    (if connection direction deviates from normal direction)
- Sampling problems
  - Problem: for example flat surface with engraved letters
  - No convergence in that case
  - Improvement: Sample correspondence points with distribution to cover unit sphere of normal directions as uniformly as possible

Normal Distribution Transform

Idea:

- Regular grid on data
- Compute normal distribution of points in each grid cell
- Matching a second point cloud:
  - Maximize likelihood (sum of log-likelihoods) of all data points
  - Use M-estimator (truncated quadrics)
  - Overlapping grid / blending to avoid discontinuities

Advantages:

- Very fast – direct grid cell access per point
- Used in robotics (self localization and mapping “SLAM”)

Deformable ICP

Deformable ICP:

- Scanners are not perfectly calibrated
- Some deformation might be necessary in order to match objects
- Related problem: acquiring deformable shapes (e.g. humans in different poses)

Deformable ICP

Solution:

- Use a variational deformation model in combination with point-to-point or point-to-plane (preferable) constraints
- Regularization term: $\mathbf{f}: \mathbb{R}^3 \rightarrow \mathbb{R}^3$

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

- Data matching term:

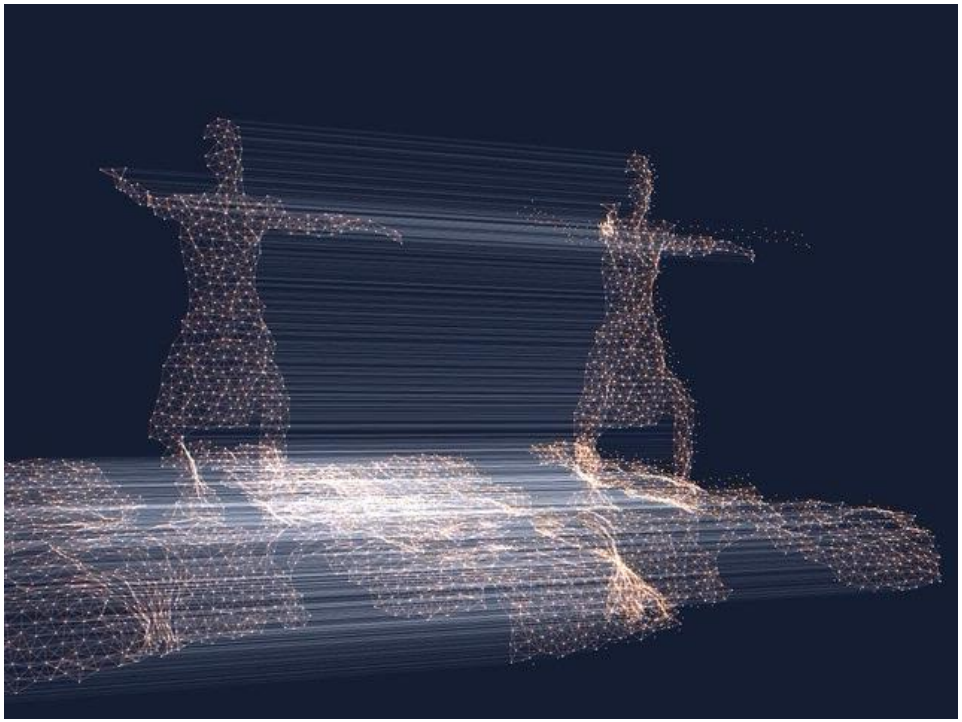
$$E^{(data)} = \sum_{i=1}^{n} \left\langle \mathbf{f}(\mathbf{p}_j^{(A)}) - \mathbf{p}_{nearest(j)}^{(B)}, \mathbf{n}_{nearest(j)}^{(B)} \right\rangle^2$$

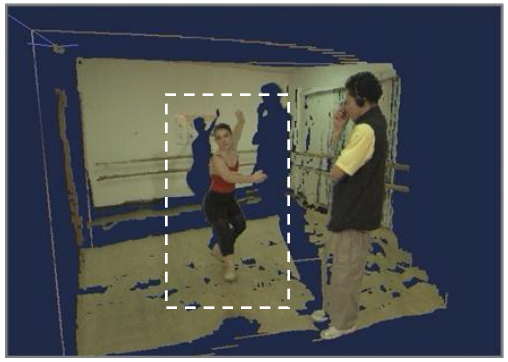
- Minimize: $\underset{\text{deformations } \mathbf{f}}{\arg\min} \; E^{(match)}(\mathbf{f}) = E^{(data)}(\mathbf{f}) + E^{(deform)}(\mathbf{f})$

Example

Example:

- Two frames
- Stereo vision scan of a ballet dancer (8 cameras)
- Deformable shape matching:
  - A to B and
  - B to A
  - (repeating)





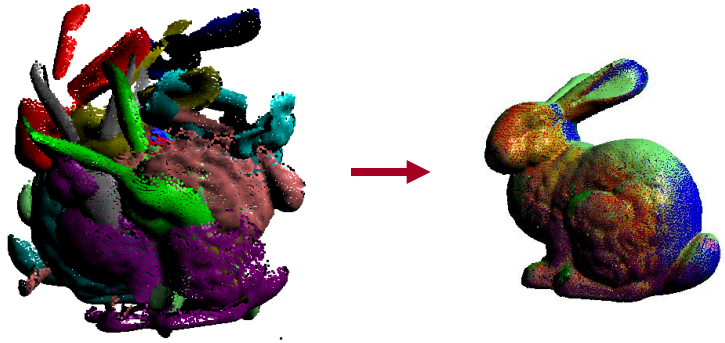
[data from Zitnick et al., Microsoft Research, 2004]

Global Matching

How to assemble the bunny (globally)?

Pipeline (rough sketch):

- Feature detection
- Feature descriptors
- Spectral validation



Feature Detection

Feature points (keypoints)

- Regions that can be identified locally
- “Bumps”, i.e. points with maximum principal curvatures
  - Fitting a quadratic heightfield to point cloud data (MLS) to compute curvatures
  - “SIFT” features – compute bumps at multiple scales:
    - Radius of geometry used for the fit as an additional parameter
    - Search for maxima in 3D surface-scale space
  - Output: list of keypoints

Descriptors

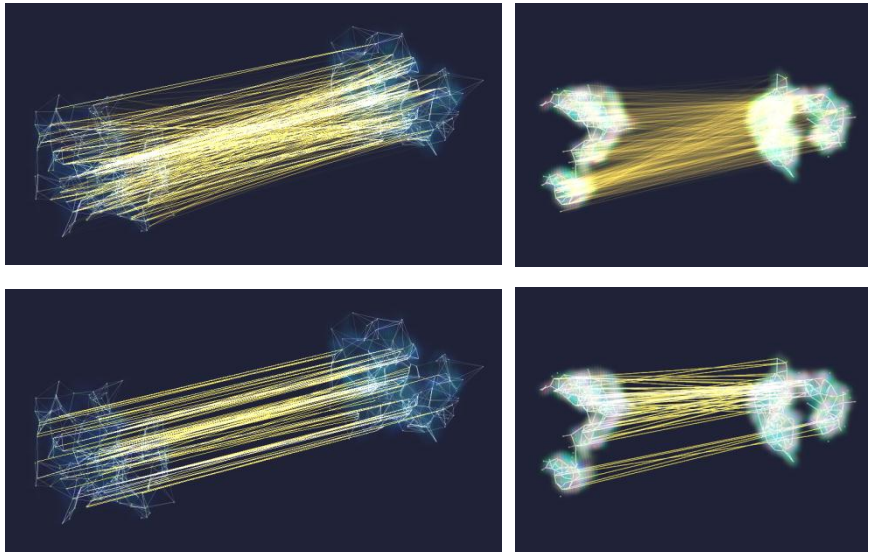
Feature descriptors:

- Rotation invariant description of local neighborhood (within scale of the feature point)
  - Translation already fixed by feature point
- In the bunny-case: histograms of principal curvature values
- Used to find match candidates
- Not 100% reliable (typically 3x – 5x outlier ratio)

Spectral Correspondence Validation

We have:

- Candidate matches
- But every keypoint matches 5 others on average
- At most one of these is correct



Validation Criterion:

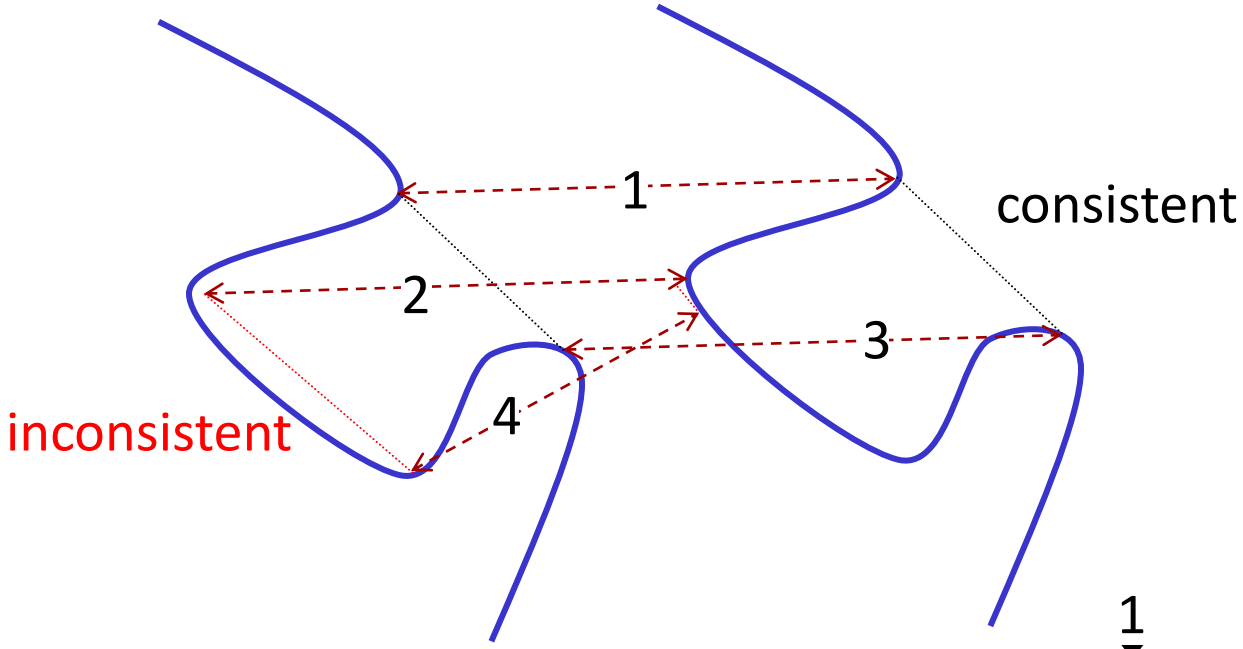
- Euclidian distance should be preserved (Deformable models: preserve geodesic distance)

Spectral Validation

Find largest set of correspondences that are all compatible:

- Form a vector with one entry for each correspondence (connecting two features)
- Build a matrix:
  - Write descriptor matching score $\in$ [0..1] on diagonal (1 = perfect match, 0 = unlikely)
  - Write pairwise compatibility $\in$ [0..1] on off-diagonals
    - Score decreases if correspondences do not preserve distances
- Compute largest eigenvalue of this matrix
- Approximation for largest consistent cluster

Consistency Check



$$\mathbf{M} = \begin{pmatrix} 0.95 & 1 & 1 & 0.3 \\ 1 & 0.9 & 1 & 0.2 \\ 1 & 1 & 0.9 & 0.25 \\ 0.3 & 0.2 & 0.25 & 0.7 \end{pmatrix}$$

1 2 3 4 (column indices) ◂ 1 ◂ 2 ◂ 3 ◂ 4 (row indices)

Quantization

Final Quantization:

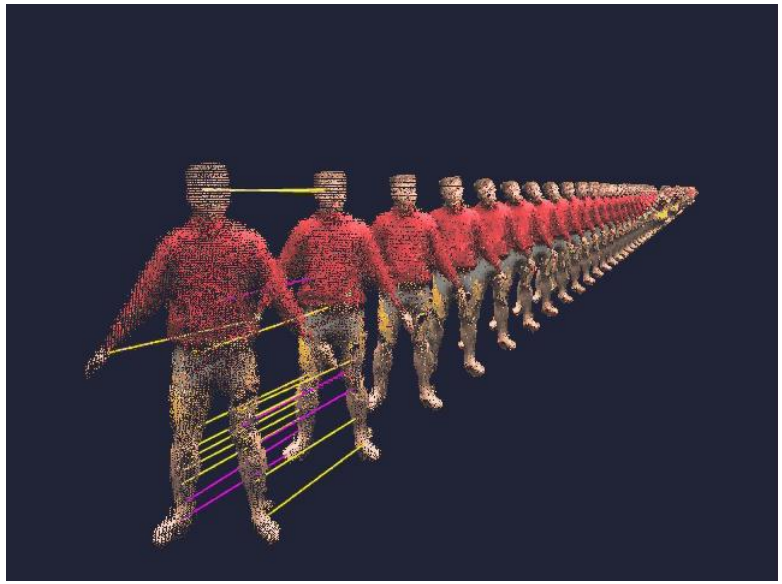
- Set largest eigenvector entry to one
- Set all others to zero that are not compatible (fixed threshold)
- Repeat until all entries are quantized

Reference: **M. Leordeanu, M. Hebert,** A Spectral Technique for Correspondence Problems Using Pairwise Constraints, ICCV 2005.

Deformable Global Matching

This technique also works for deformable matching:

- Replace Euclidian distance by geodesic (intrinsic, on-the-surface) distance
- Computed by Dijkstra algorithm on nearest neighbor graph of point samples.



[Data set: Christian Theobald, Implementation: Martin Bokeloh]

Surface Reconstruction

Moving Least Squares Techniques

Moving Least Squares

Motivation:

- Point sets sample the object they describe only sparsely.
- There is an infinite amount of emptiness in between the finite sample set.
- How can we fill in surface points?

Goals:

- Compute surface representations *locally*.
- We do not want to solve a global variational problem.
- Create smooth surfaces.
- Determine differential properties.

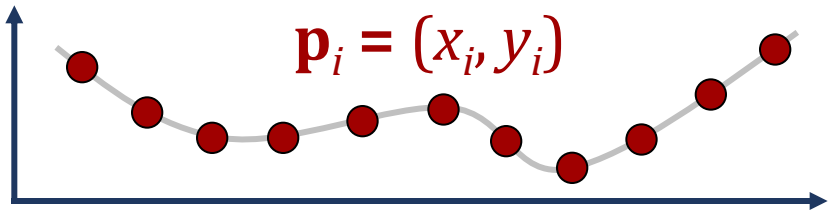
Moving Least Squares

Moving least squares (MLS):

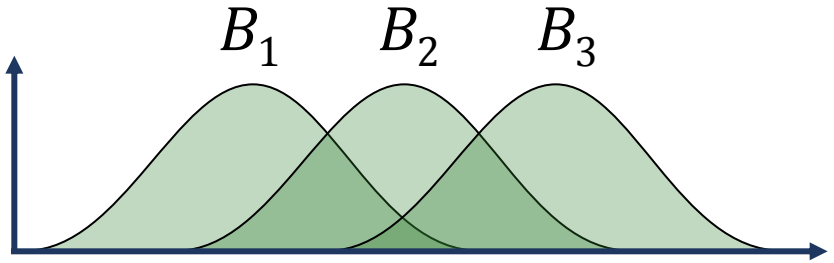
- MLS is a standard technique for scattered data interpolation.
- We will consider three variations:
  - The standard interpolation scheme.
  - How to build MLS basis functions for finite elements.
  - How to define surface projection operators.

Weighted Least-Squares

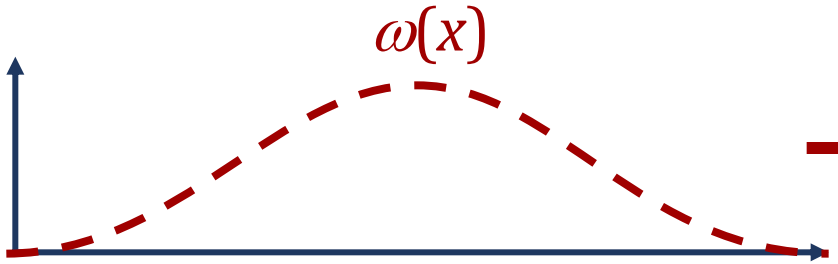
Least Squares Approximation:



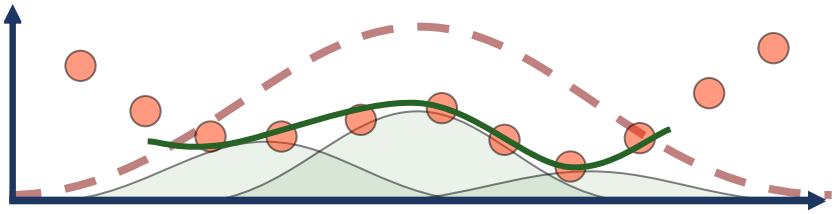
target values



basis functions



weighting functions



least squares fit

Least-Squares

Least Squares Approximation:

$$\tilde{y}(x) = \sum_{i=1}^{n} c_i B_i(x)$$

Best Fit:

$$\underset{c_i}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| \left(\tilde{y}(x_i) - y_i \right) \omega(x_i) \right\|^2$$

Least-Squares

Normal Equations: $\left(\mathbf{B}^T\mathbf{W}^2\mathbf{B}\right)\boldsymbol{\lambda} = \left(\mathbf{B}^T\mathbf{W}^2\right)\mathbf{y}$

Solution: $\boldsymbol{\lambda} = \left(\mathbf{B}^T\mathbf{W}^2\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{W}^2\mathbf{y}$

Evaluation: $\tilde{y}(x) = <\mathbf{b}(x), \boldsymbol{\lambda}> = \mathbf{b}(x)^{\mathrm{T}}\left(\mathbf{B}^{\mathrm{T}}\mathbf{W}^2\mathbf{B}\right)^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{W}^2\mathbf{y}$

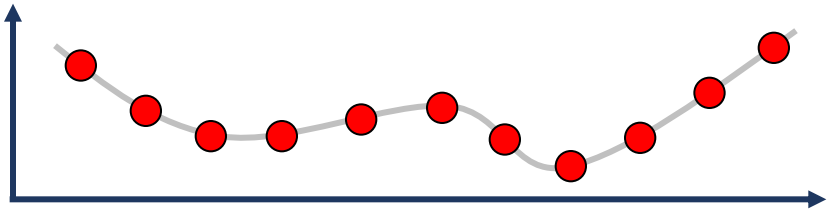
MLS approximation

Notation: $\mathbf{b} := \left[B_1, \ldots, B_n\right]$

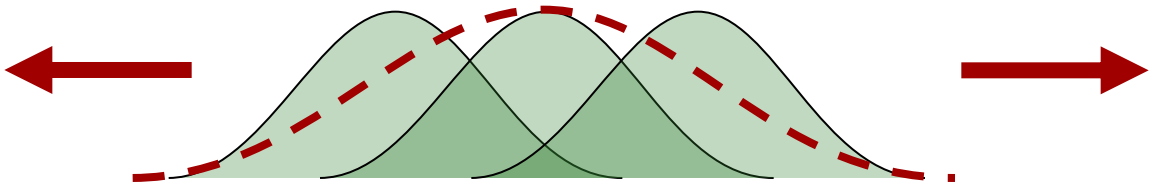
$$\mathbf{B} := \begin{bmatrix} -\mathbf{b}(x_1)- \\ \vdots \\ -\mathbf{b}(x_n)- \end{bmatrix} \quad \mathbf{y} := \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{c} := \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} \quad \mathbf{W} := \begin{bmatrix} \omega(x_1) & & \\ & \ddots & \\ & & \omega(x_n) \end{bmatrix}$$

Moving Least-Squares

Moving Least Squares Approximation:



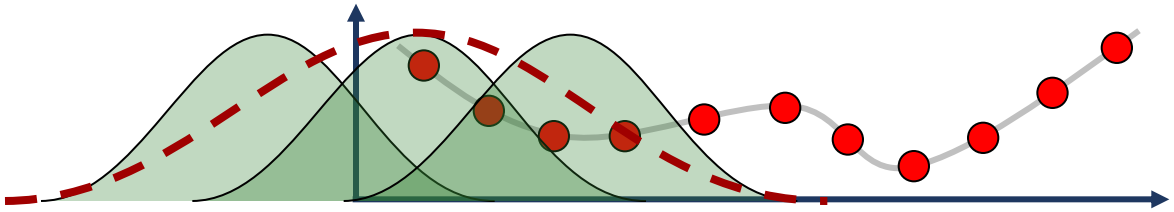
target values



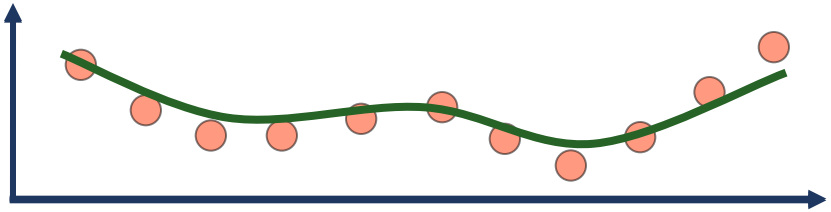
move basis and weighting function,
recompute approximation $\tilde{y}(x)$

Moving Least-Squares

Moving Least Squares Approximation:



target values



approximation

Summary: MLS

Standard MLS approximation:

- Choose set of basis functions
  - Typically monomials of degree 0,1,2
- Choose weighting function
  - Typicall choices: Gaussian, Wendland function, B-Splines
  - Solution will have the same continuity as the weighting function.
- Solve a weighted least squares problem at each point:

$$\tilde{y}(x) = \mathbf{b}(x)^{\mathrm{T}} \left(\mathbf{B}(x)^{\mathrm{T}} \mathbf{W}(x)^2 \mathbf{B}(x) \right)^{-1} \mathbf{B}(x)^{\mathrm{T}} \mathbf{W}(x)^2 \mathbf{y}$$

moment matrix

- Need to invert the “moment matrix” at each evaluation.
- Use SVD if sampling requirements are not guaranteed.

Defining FE Basis Functions

Meshless finite elements:

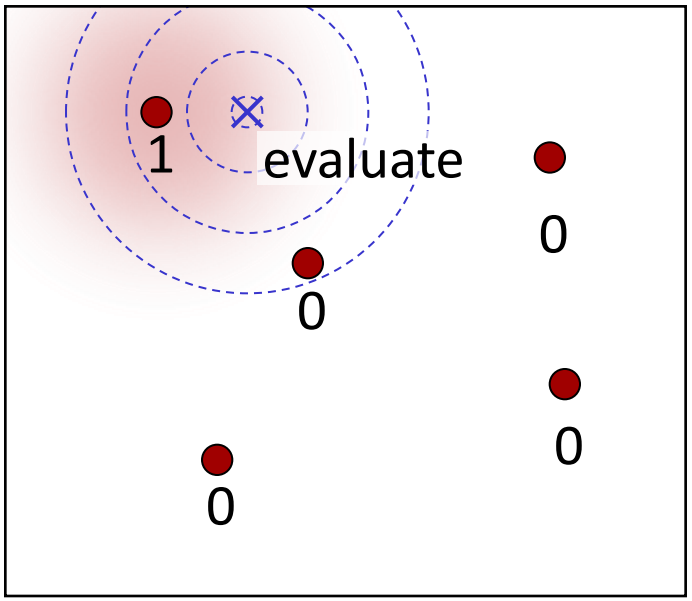
- Alternative to meshing
  (define basis functions on regular grid or triangle mesh)
- Easy to obtain arbitrary consistency orders
- Main drawback: need to invert moment matrices in basis function evaluations (efficiency problem)

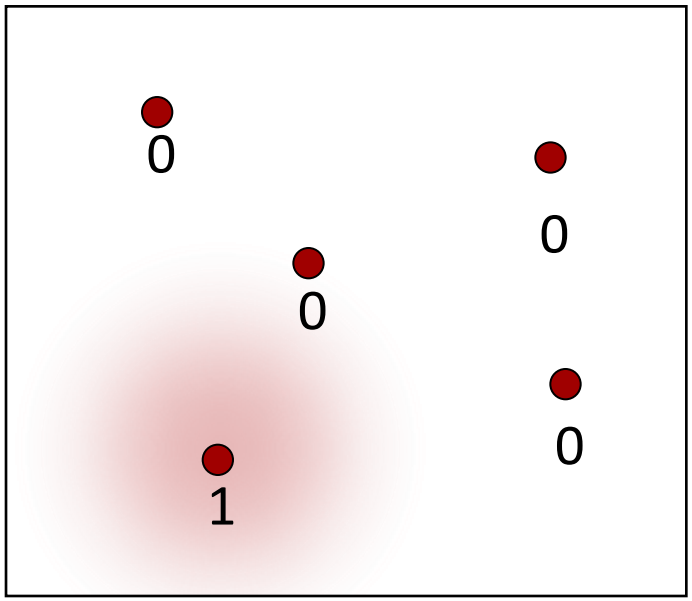
Here is the main idea:

- Given *n* points in space
- Form radial basis functions by an MLS interpolation
- For each basis function assign a “1” to one of the points

Meshless MLS-FE Basis

Constructing the basis:





MLS Basis Functions

Properties:

- The basis is smooth (smoothness corresponding to the windowing function).
- The functions form a partition of unity (sum to one at each point).
- The consistency order is the same as that of the basis functions (for example: quadratic basis leads to quadratic precision, reproduces 2nd order functions exactly)

MLS Basis Functions

Main advantage:

- Just need to place points to form a discretization
- Multi-resolution / adaptive / dynamic sampling much easier than with standard FE tools
- Used frequently in dynamic simulations (fluids, free boundaries, fracturing and crack propagation)

Main disadvantage:

- Function evaluation needs moment matrix inversion.
- Slower than fixed basis functions.
- Computing FE integrals (numerically) is more expensive.

FE Integration (recap)

Discretizing a variational problem:

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

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

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

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

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

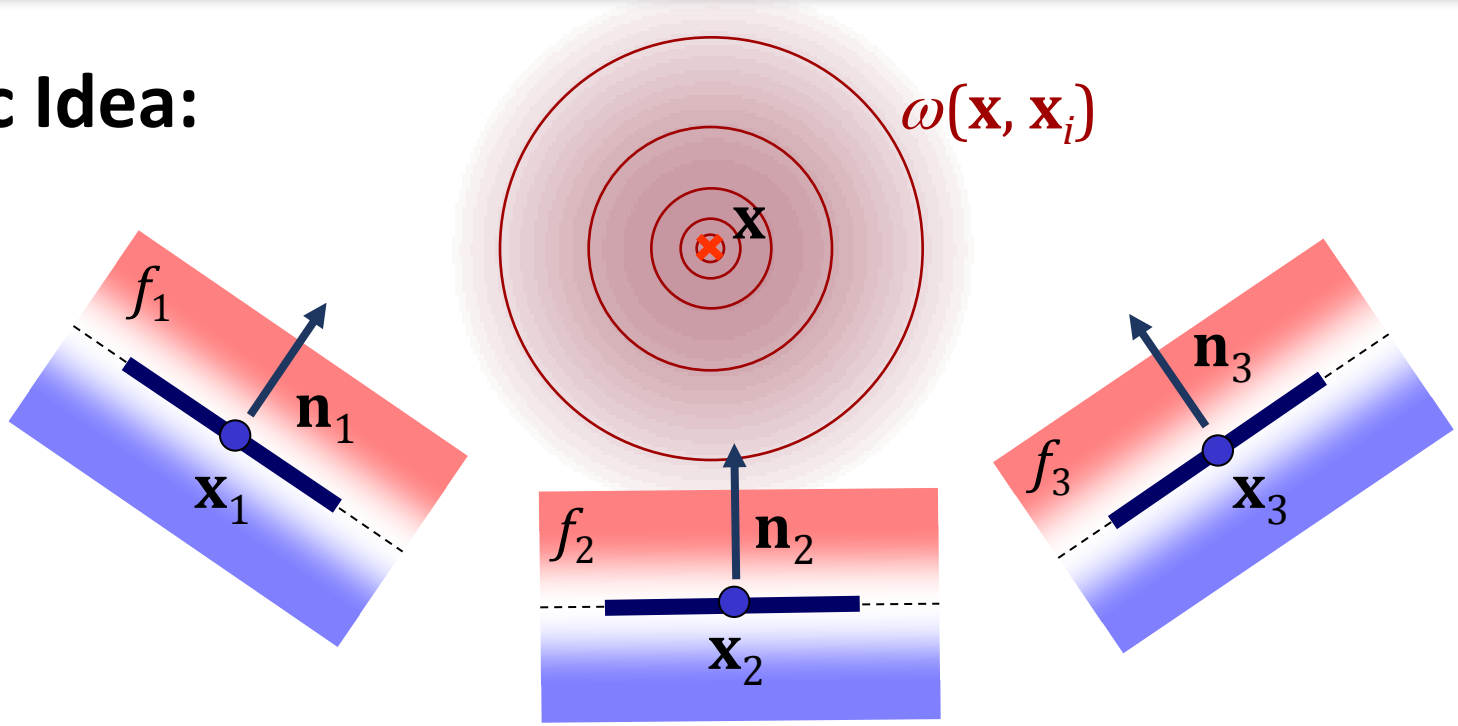
Surface Definition

Question: How to define surfaces via MLS?

- Two alternatives (as examples)
  - Implicit function definition for points with oriented normals.
  - Surface fitting for points without normals
- Many more variants known in literature...

Implicit Function Definition

Basic Idea:



$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^{\mathrm{T}} \left(\mathbf{B}(\mathbf{x})^{\mathrm{T}} \mathbf{W}(\mathbf{x})^2 \mathbf{B}(\mathbf{x})\right)^{-1} \mathbf{B}(\mathbf{x})^{\mathrm{T}} \mathbf{W}(\mathbf{x})^2 \mathbf{y} \qquad \mathbf{y} = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{pmatrix}$$

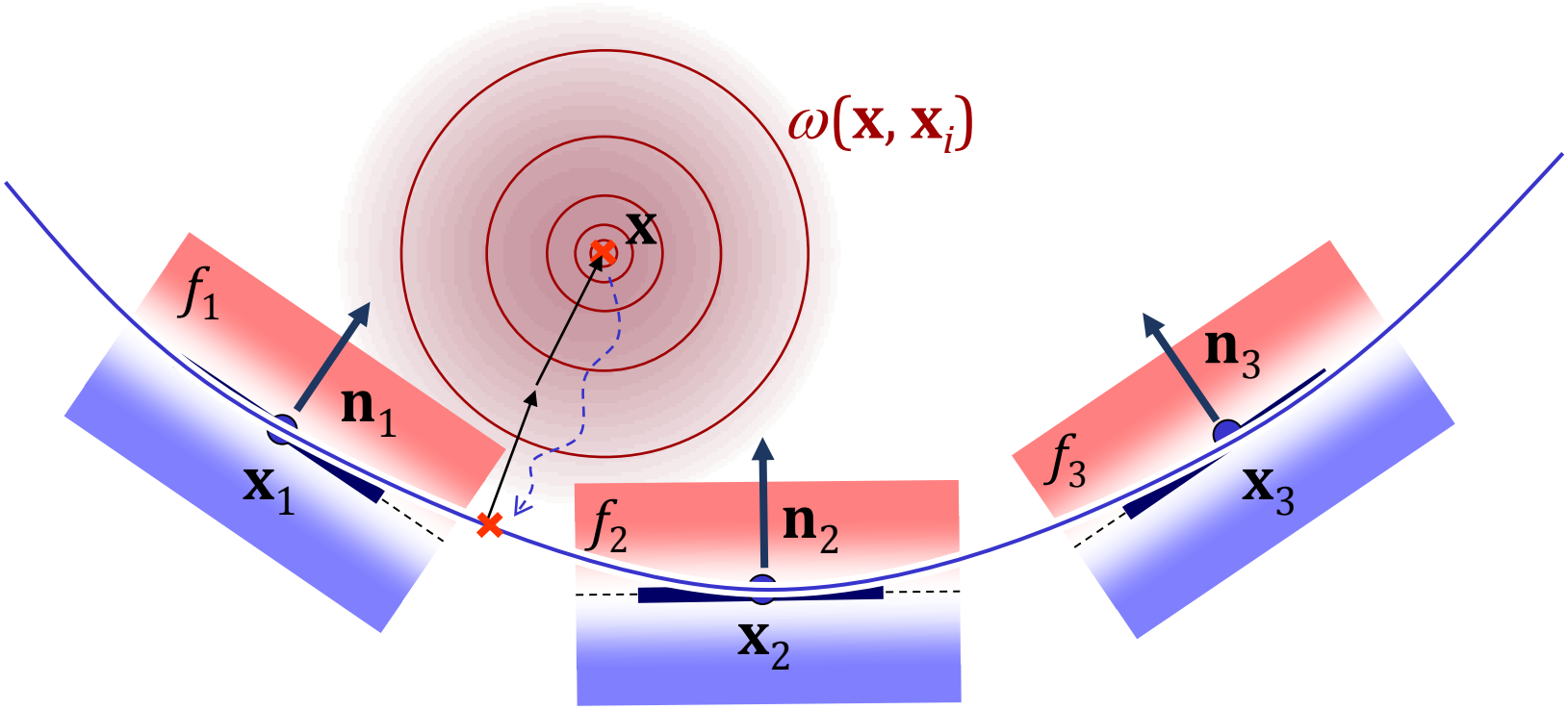
Projection Operator

Problem:

- We want to insert additional points in the proximity of other points
- Define a “projection operator”:
  - Compute implicit function
  - Add a new point somewhere
  - Move (gradient decent, or Newton’s method) point onto zero-level set
  - Move in normal direction
    (i.e. gradient of approximated implicit function)
  - The operation that maps a point to the local zero level set by following the gradient (stationary point of an ODE) is called the “projection operator”.

Implicit Function Definition

Projection:



Unoriented Point Sets

Problem:

- This requires normals with consistent orientation.
- Hard to get, in particular locally.
- For the general case, there is another MLS scheme that does not construct a signed implicit function.

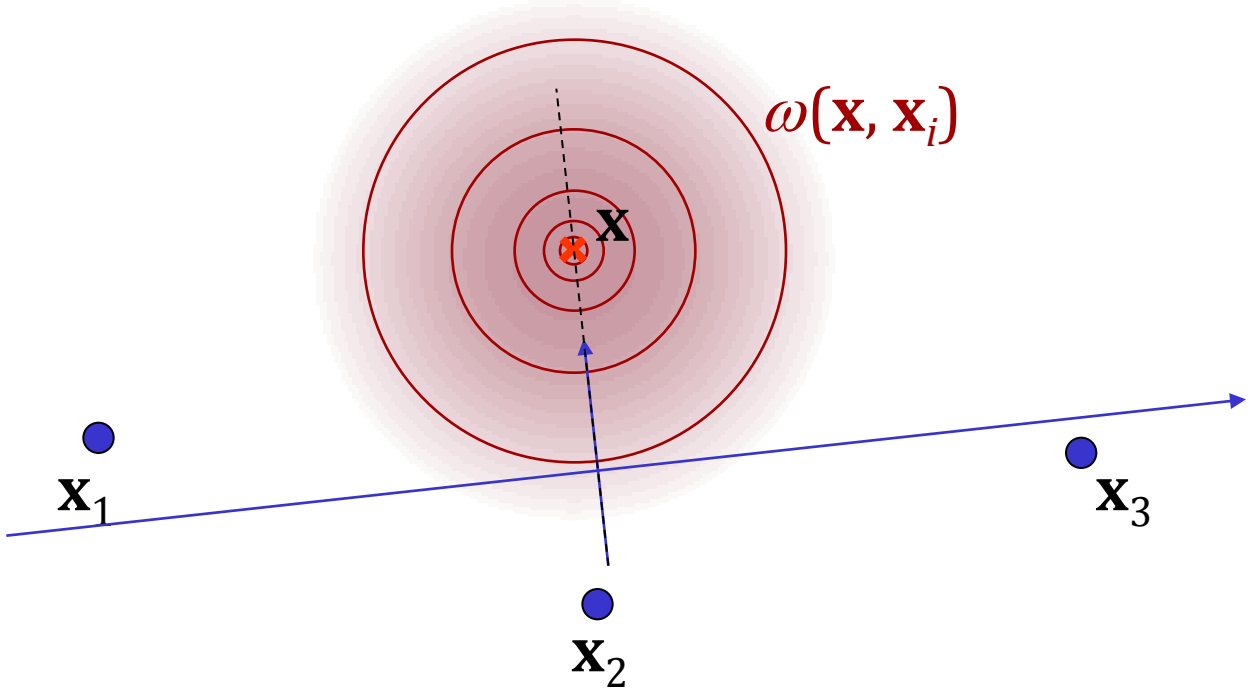
Point Set Surfaces

Point Set Surfaces:

- Discussed here: a variant of [Alexa et al. 2001].
- Start with just points in space.
- Again use a weighting function.
- Then perform three steps:
  - First, compute a coordinate system
  - Second, compute a weighted least squares fit for higher order consistency.
  - Third, project point on the computed function fit.

1. Coordinate system

Establishing an MLS coordinates frame

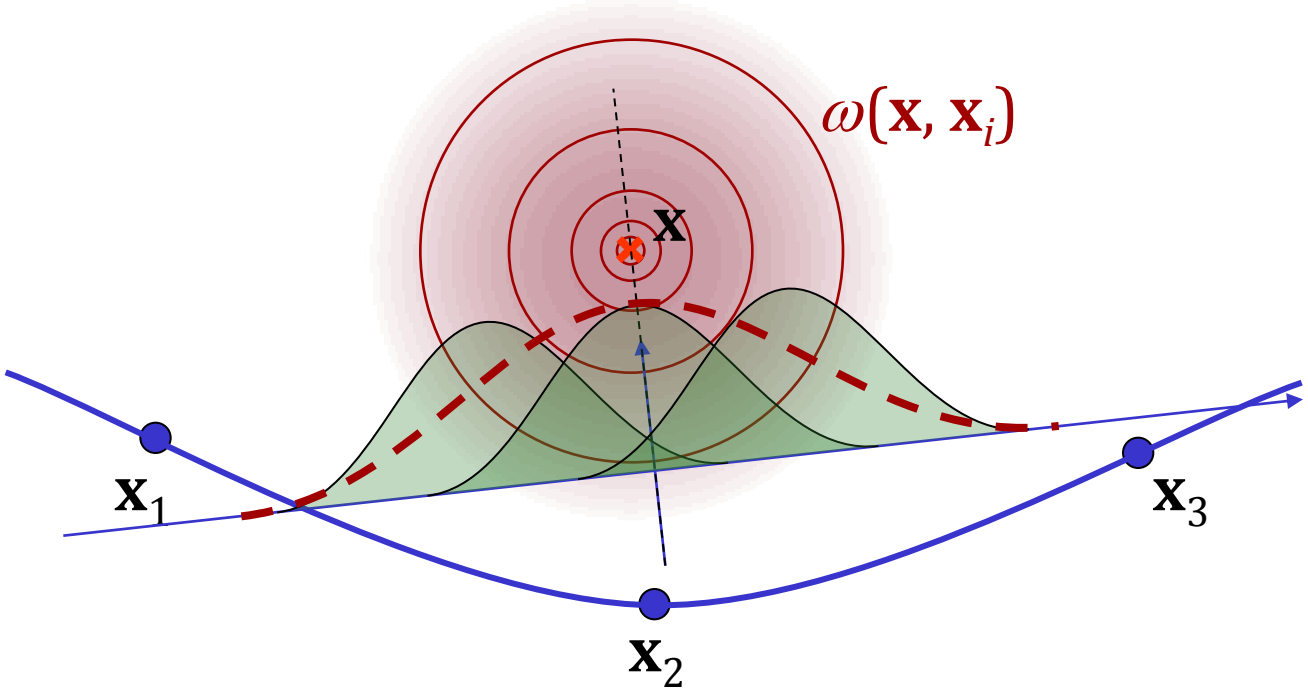


Implementation:

- This can be done using *weighted* total least squares (PCA)
- The original paper uses a non-linear optimization

2. Basis Function Fit

Weighted least-squares fit to a moving basis system:



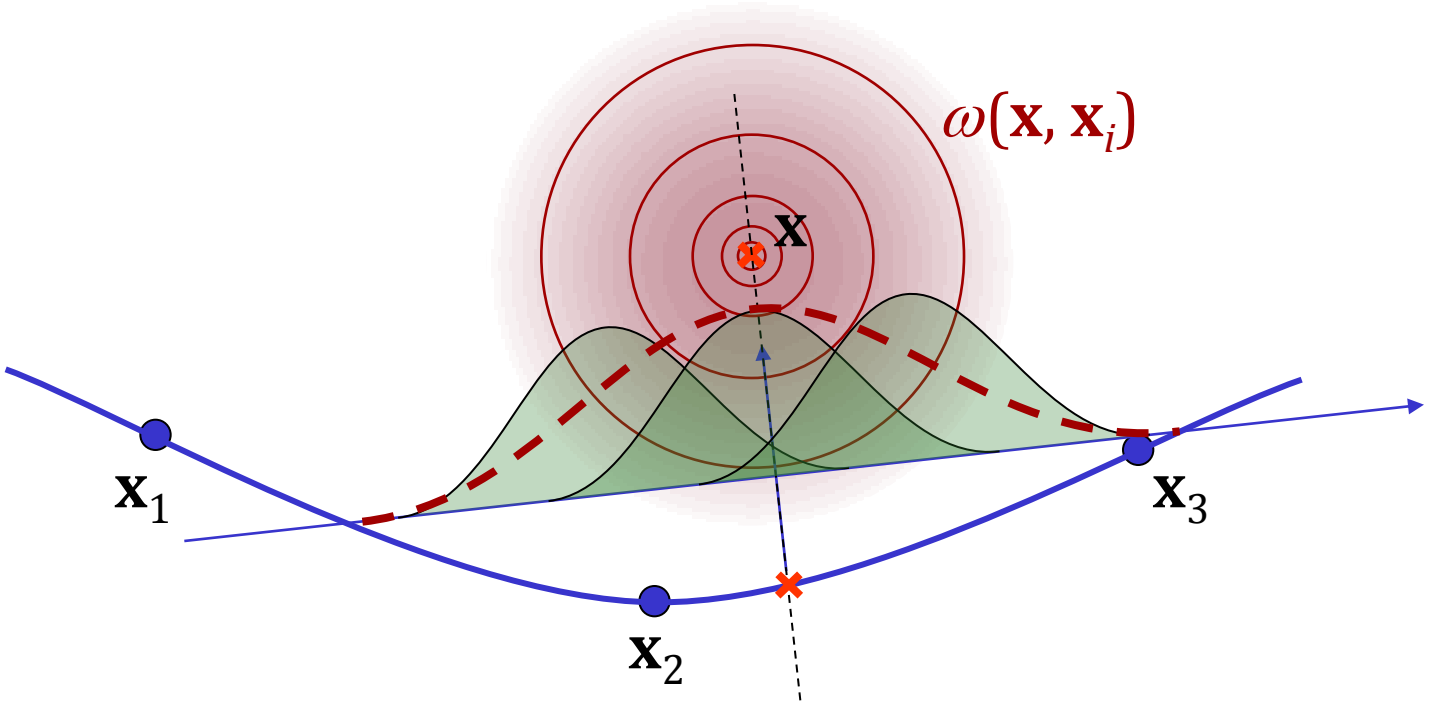
Implementation:

- Ordinary *weighted* least squares.
- Use the same spatial windowing function ω for continuity.

3. Projection

Projection Step:

- Project evaluation point on surface



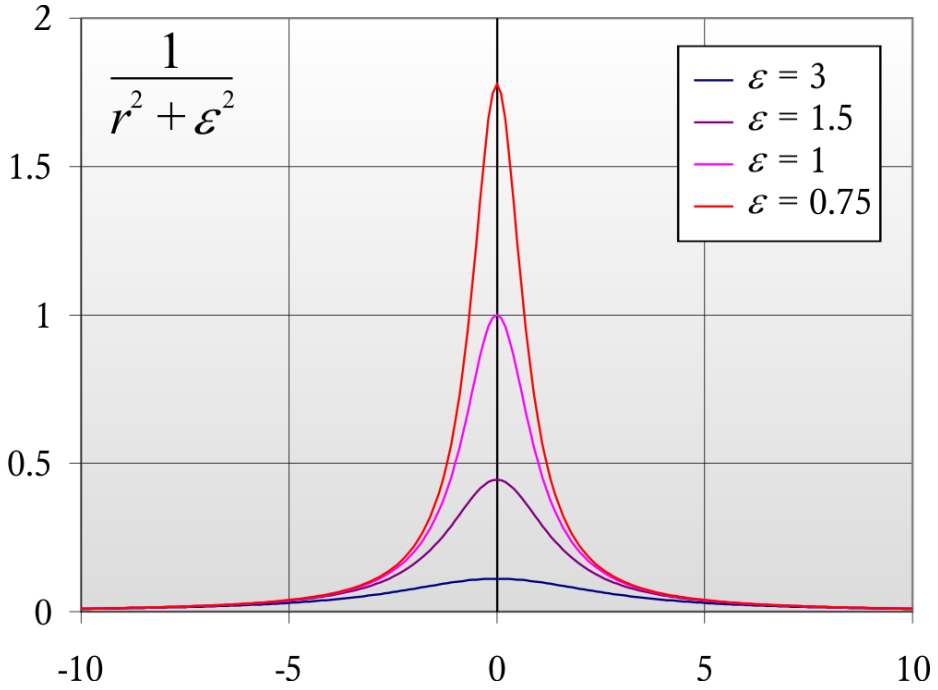
Continuity Control

Approximation / Interpolation:

- Weighting function shape & support determine tightness of fit.
- Special case: Integrable, singular weighting functions allow for interpolation
- Example: Fitting an MLS surface to a polynomial surface [Shen et al., Siggraph 2004]

Weighting Function

Weighting Function:



Vary ε to adjust tightness of fit.

Polygonal Constraints

Point Constraints:

$$\left(\sum_{i=1}^{N} w^2(x,x_i)\,b(x_i)\,b^T(x_i)\right)c(x) = \sum_{i=1}^{N} w^2(x,x_i)\,b(x_i)\,y_i$$

Polygonal Constraints:

$$\left(\sum_{k=1}^{N} \int_{Poly_k} w^2(x,p)\,b(p)\,b^T(p)\,dp\right)c(x) = \sum_{k=1}^{N} \int_{Poly_k} w^2(x,p)\,b(p)\,y_k\,dp$$

(just integrate over all polygon points)

Point-Based Modeling

Direct Point-Based Modeling Techniques

Direct Point-Based Modeling

Example papers:

- Mark Pauly, Markus Gross:
  Spectral Processing of Point-Sampled Geometry
  Siggraph 2001.
- Mark Pauly, Richard Keiser, Leif Kobbelt, Markus Gross:
  Shape Modeling with Point-Sampled Geometry
  Siggraph 2003.