Eurographics 2012 Tutorial

A Practical Guide to POLYGON MESH REPAIRING

Marcel Campen
Leif Kobbelt

RWTH Aachen University, Germany

Marco Attene

IMATI-GE, CNR, Italy

Where are we located !?

Encyclopedia of the Universe
- (1) Science
  - (1.1) Computer Science
    - (1.1.1) Computer Graphics
      - (1.1.1.a) Geometry Processing
        - (1.1.1.a.a) Mesh Processing
          - **(1.1.1.a.a.i) Mesh Repair**

Motivation

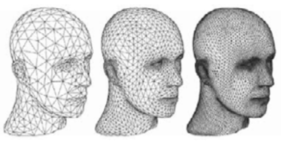
- demand for digital 3D models is ubiquitous
  - CAD / CAM
  - Simulation
  - Gaming
  - Cultural heritage
  - Medicine
  - Bioinformatics

Motivation

- depending on the application, 3D models need to be:
  - visualized
  - analyzed
  - processed
  - converted
- advanced algorithms in these contexts often have strict requirements on model quality and integrity

Motivation

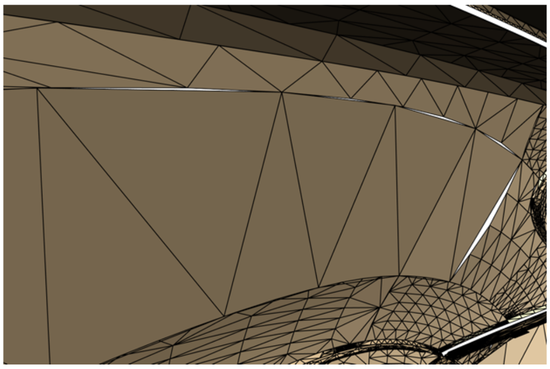
- polygon meshes are a de facto standard in numerous domains
  - extremely flexible and descriptive
  - supported by hardware acceleration



- their versatility, at the same time, allows for a variety of defects and flaws in the representation

Motivation

Motivation

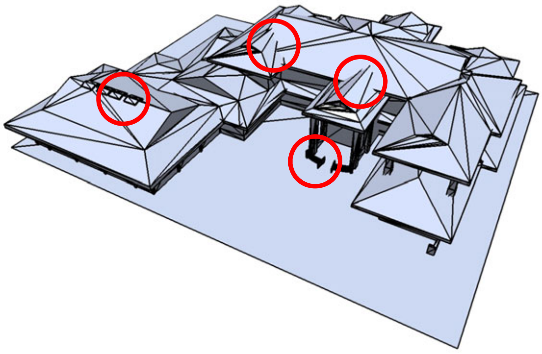


Motivation

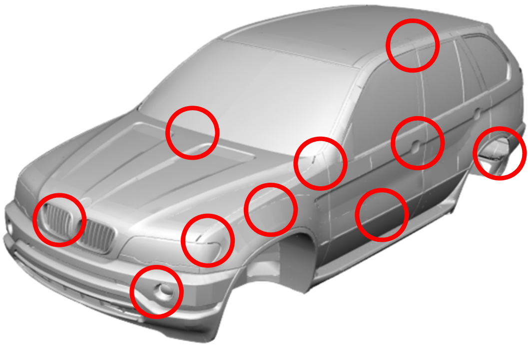
- *real world* meshes often contain various defects, depending on their origin.
- but many applications assume *ideal* meshes free from defects or flaws.

- ***Mesh Repairing*** adapts raw mesh models to specific application requirements.

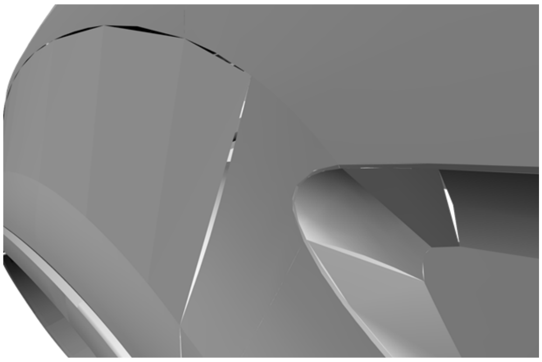
Motivation



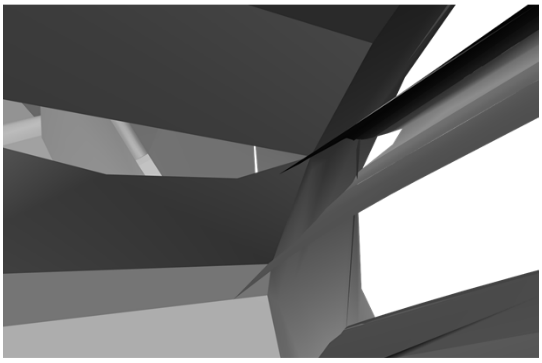
Motivation



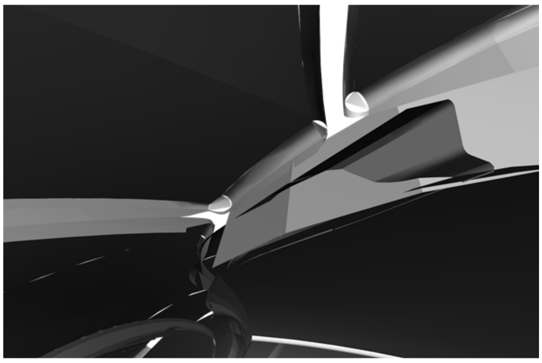
Motivation



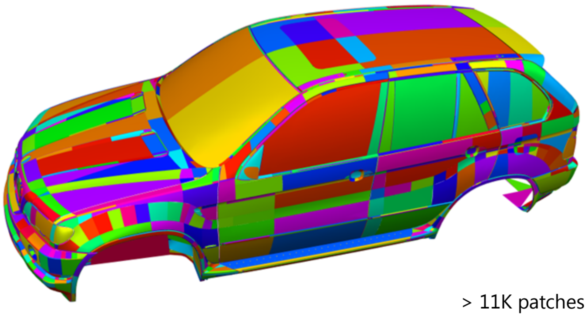
Motivation



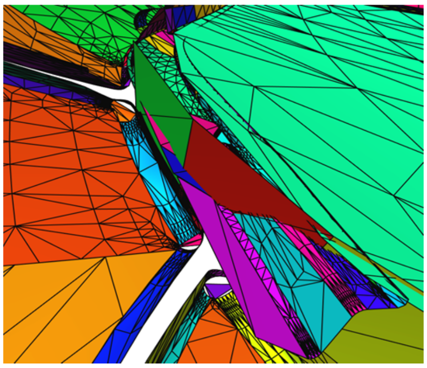
Motivation



Motivation

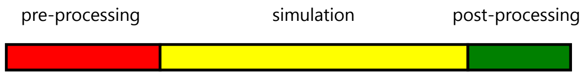


Motivation



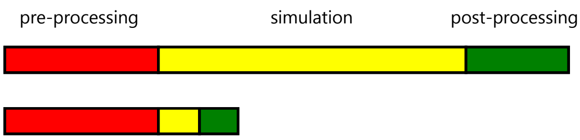
Motivation

- complexity of the repair task is often underestimated by non-experts



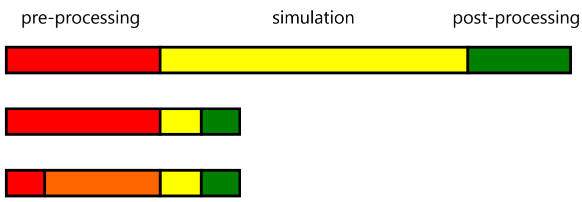
Motivation

- complexity of the repair task is often underestimated by non-experts



Motivation

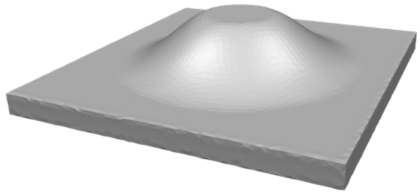
- complexity of the repair task is often underestimated by non-experts



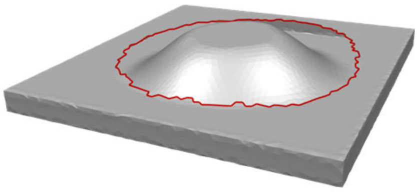
Motivation

- complexity of the repair task is often underestimated by non-experts
  - big difference between „looks good“ and „is good“
  - reliable handling of all degenerate cases is challenging to implement
  - most repair algorithms focus on certain defect types and ignore or even introduce others

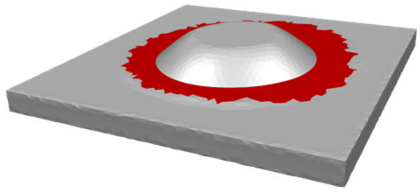
Motivation



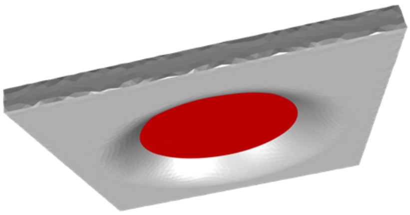
Motivation



Motivation



Motivation



The Mesh Repair Problem

- given: input mesh / polygon soup M
- find: output mesh M'
  - globally consistent manifold / solid "watertight"
  - tolerance: dist(M,M') < epsilon

Tolerances

- dist(p,q) = ||p – q||

Tolerances

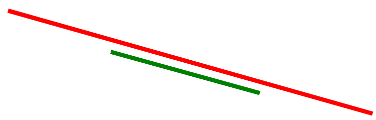
- dist(p,q) = ||p – q||
- dist(p,S) = min { dist(p,q) | ∀q∈S }

Tolerances

- dist(p,q) = ||p – q||
- dist(p,S) = min { dist(p,q) | ∀q∈S }
- dist(S,S′) = max { dist(p,S′)) | ∀p∈S }

Tolerances

- dist(p,q) = ||p – q||
- dist(p,S) = min { dist(p,q) | ∀q∈S }
- dist(S,S′) = max { dist(p,S′)) | ∀p∈S }

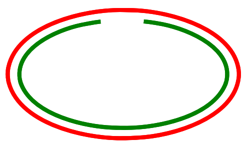


Tolerances

- dist(p,q) = ||p – q||
- dist(p,S) = min { dist(p,q) | ∀q∈S }
- dist(S,S′) = max { dist(p,S′)) | ∀p∈S }
- dist(S,S′) ≠ dist(S′,S)
- **Hausdorff** distance:
  max { dist(S,S′), dist(S′,S) }

The Mesh Repair Problem

- given: input mesh / polygon soup M
- find: output mesh M′
  - globally consistent manifold / solid "watertight"
  - tolerance: dist(M,M′) < epsilon

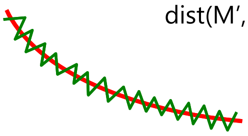


The Mesh Repair Problem

- given: input mesh / polygon soup M
- find: output mesh M'
  - globally consistent manifold / solid "watertight"
  - tolerance: dist(M,M') < epsilon
    dist(M',M) < delta
    dist(M',M) > epsilon only at ∂M

The Mesh Repair Problem

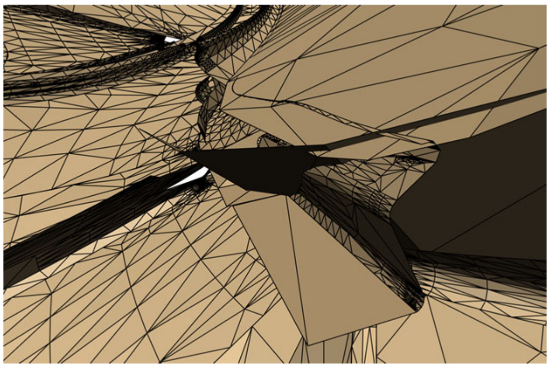
- given: input mesh / polygon soup M
- find: output mesh M'
  - globally consistent manifold / solid "watertight"
  - tolerance: dist(M,M') < epsilon
    dist(M',M) < delta
    dist(M',M) > epsilon only at ∂M



The Mesh Repair Problem

- given: input mesh / polygon soup M
- find: output mesh M'
  - globally consistent manifold / solid "watertight"
  - tolerance: dist(M,M') < epsilon
    dist(M',M) < delta
    dist(M',M) > epsilon only at ∂M
  - faithful normal reconstruction

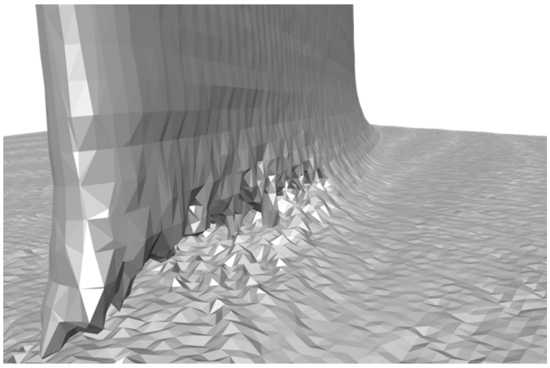
Spurious Geometry



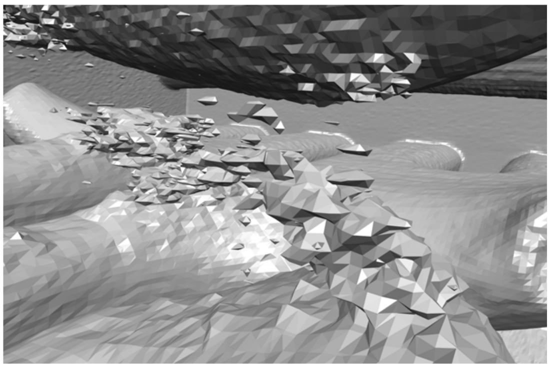
The Mesh Repair Problem

- given: input mesh / polygon soup M
- find: output mesh M'
  - globally consistent manifold / solid "watertight"
  - tolerance: dist(M,M') < epsilon
    dist(M',M) < delta
    dist(M',M) > epsilon only at ∂M
  - faithful normal reconstruction
  - remove spurious geometry

Topological Noise



Topological Noise



The Mesh Repair Problem

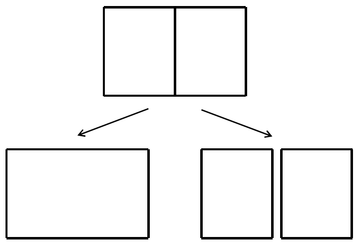
- given: input mesh / polygon soup M
- find: output mesh M'
  - globally consistent manifold / solid "watertight"
  - tolerance: dist(M,M') < epsilon
    dist(M',M) < delta
    dist(M',M) > epsilon only at ∂M
  - faithful normal reconstruction
  - remove spurious geometry
  - remove topological noise

The Mesh Repair Problem

- the *general* mesh repair problem is genuinely **ill-posed**

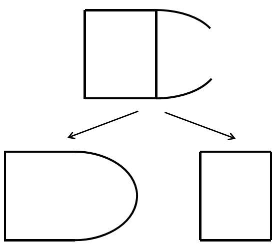
The Mesh Repair Problem

- the *general* mesh repair problem is genuinely **ill-posed**



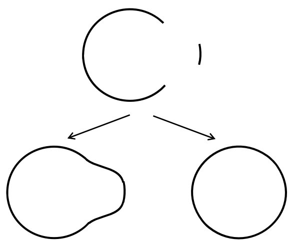
The Mesh Repair Problem

- the *general* mesh repair problem is genuinely **ill-posed**



The Mesh Repair Problem

- the *general* mesh repair problem is genuinely **ill-posed**



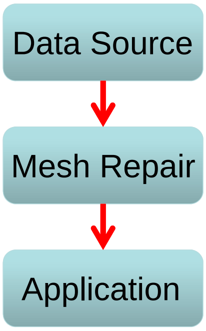
The Mesh Repair Problem

- the *general* mesh repair problem is genuinely **ill-posed**
  - inherent ambiguities (topological & geometrical)
- domain knowledge
- heuristics
- interactive user input
- all these are application specific …

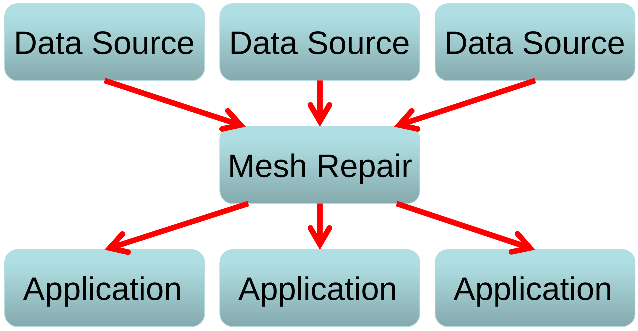
The Application Perspective

- *the* optimal mesh repair method does not (yet) exist
  - each has advantages and disadvantages
  - some defects are repaired, others introduced
  - the input needs to meet certain requirements
  - only certain (limited) guarantees about the output are provided
- hence, application context needs to be considered to make the best trade-off.

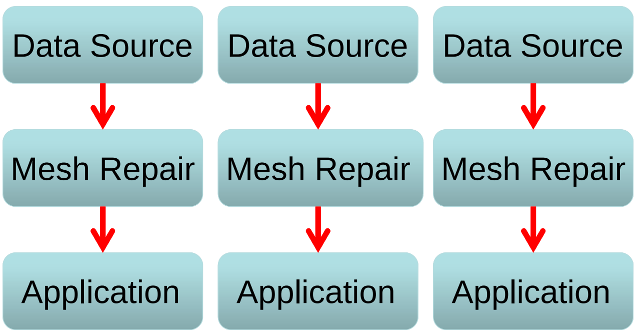
The Application Perspective



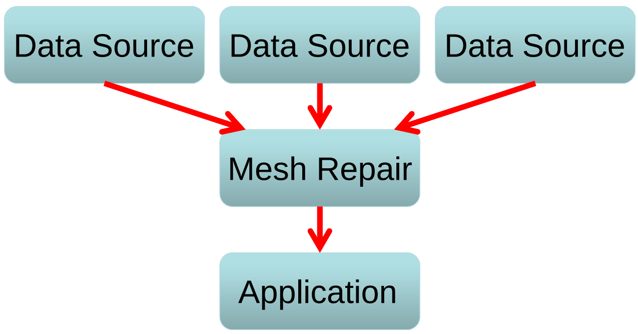
The Application Perspective



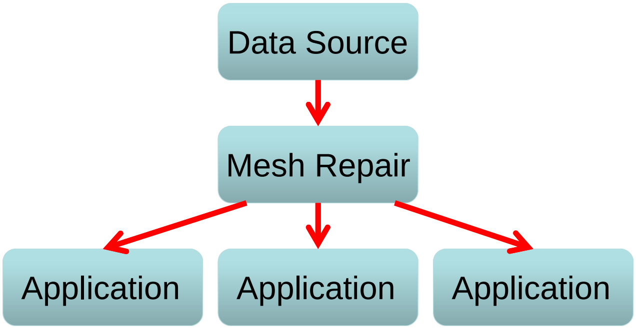
The Application Perspective



The Application Perspective



The Application Perspective



The Application Perspective

- categorization of:
  - defect types
  - upstream applications / data sources
    - based on typical defects of output meshes.
  - downstream applications
    - based on typical requirements on input meshes.
  - repair approaches
    - along with specific requirements and guarantees

The Application Perspective

- based on these criteria and by looking at the combinatorics of
  - upstream application
  - repair algorithm
  - downstream application

  guidelines can be derived to find methods well-suited for a specific problem setting

Mesh Repair Recipe

1. what is the upstream application?
   → determines characteristics and defects of M
2. what is the downstream application?
   → determines requirements on M'
3. is it actually necessary to repair M?
   4. does there exist a suitable algorithm?
   5. can several methods be combined?
   6. otherwise:
      → there is a gap in the state of the art …

Tutorial Outline

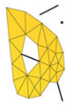
- defect types
- upstream applications
- downstream applications
- repair approaches
  - local ⇔ global
  - mesh-based ⇔ volumetric
  - geometrical ⇔ topological
- repair workflows – an example
- discussion & open problems

DEFECT TYPES

- We distinguish issues about:
  - Local connectivity
    - „The set of polygons does not represent a combinatorially manifold simplicial complex"
  - Global topology
    - „The overall topological structure (number of components, genus, orientability) is wrong"
  - Geometry
    - „The geometric realization is flawed (holes, gaps, noise, …)"

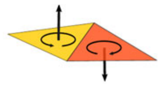
DEFECT TYPES

- Local connectivity
  - Isolated vertices
    - „A vertex that is not incident to any edge"
  - Dangling edges
    - „Edges without any incident triangles"
  - Singular edges
    - „Edges with more than two incident triangles"
  - Singular vertices
    - „Vertices with a non-disc neighborhood"

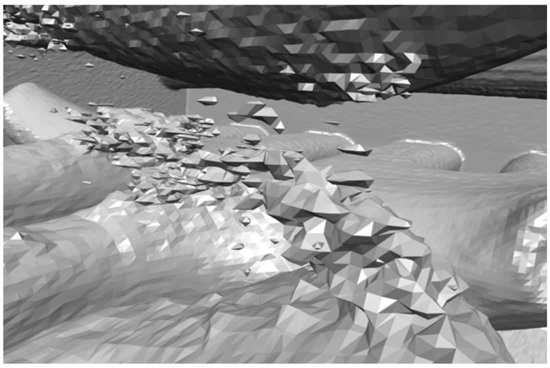


DEFECT TYPES

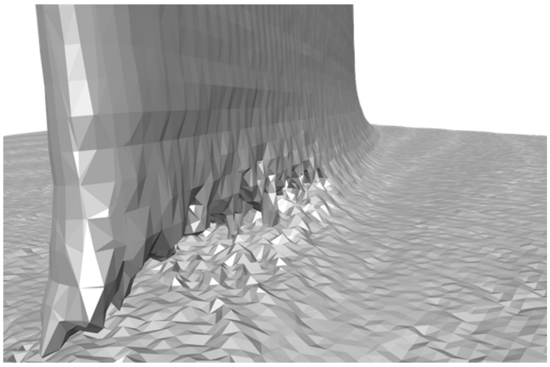
- Global topology
  - Topological noise
    - „Tiny spurious handles or tunnels"
    - „Tiny disconnected components"
    - „Unwanted cavities"
  - Orientation
    - „Incoherently oriented faces"

DEFECT TYPES

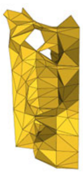
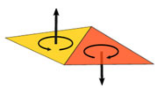


DEFECT TYPES



DEFECT TYPES

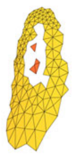
- Global topology
  - Topological noise
    - „Tiny spurious handles or tunnels"
    - „Tiny disconnected components"
    - „Unwanted cavities"
  - Orientation
    - „Incoherently oriented faces"

DEFECT TYPES

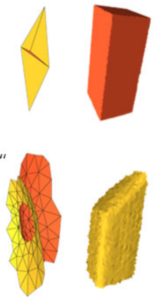
- Geometry
  - Holes
    - „Missing pieces within a surface"
      - e.g. due to occlusions during capturing
  - Gaps
    - „Missing pieces between surfaces"
      - e.g. due to inconsistent tessellation routines
  - Cracks / T-Junctions

  → Inherently ill-posed
  → Plausible geometry needs to be conceived

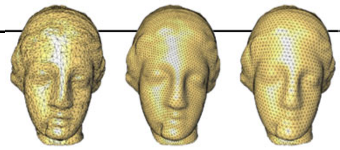



DEFECT TYPES

- Geometry
  - Degenerate elements
    - „Triangles with (near-)zero area"
  - Self-intersections
    - „Non-manifold geometric realization"
  - Sharp feature chamfering
    - „Aliasing artifacts due to sampling pattern"
  - Data noise
    - „Additive noise due to measurement imprecision"



DEFECT TYPES



- Geometry
  - Besides the absence of (near-)degeneracies, the general element quality is an important characteristic in several applications.

    → Conversion of meshes to meet such „continuous quality criteria" is the scope of „surface remeshing".

UPSTREAM APPLICATIONS

- Upstream applications (or *sources*) characterized by:
  - *Nature*
    - (physical) real-world data <-> (virtual) concepts
  - *Approach*
    - *... employed to convert data to polygon mesh*
- Both aspects can be the source of defects and flaws.

UPSTREAM APPLICATIONS

- Nature
  - *Designed*
    - Basic concept is an abstraction
    - Problems due to:
      - Inaccuracies in the modeling process
      - Inconsistencies in the description/representation
  - *Digitized*
    - Measurement of real-world phenomenon
    - Problems due to:
      - Measurement inaccuracies
      - Measurement limitations

UPSTREAM APPLICATIONS

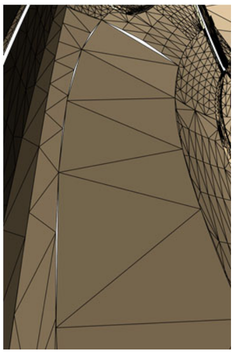
Nature	noise	holes	gaps	intersections	degeneracies	singularities	topolog. noise	aliasing
Digitized (physical)	X	X					X	X
Designed (virtual)			X	X	x	X		

UPSTREAM APPLICATIONS

- Approach
  - *Tessellation*
  - *Depth image fusion*
  - *Raster data contouring*
  - *Implicit function contouring*
  - *Reconstruction from points*
  - *Height field triangulation*
  - *Solid model boundary extraction*

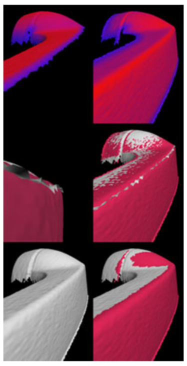
UPSTREAM APPLICATIONS

- Tessellation
  - *Gaps, Intersections*
    - *due to deviation from original curved surface*
  - *(Degeneracies)*
    - *depending on special case handling in tessellator*



UPSTREAM APPLICATIONS

- Depth image fusion
  - *Intersections*
  - *(Degeneracies, Singularities)*
    - *e.g. when using the popular Minolta V910 software*



from [Turk and Levoy 1994]

UPSTREAM APPLICATIONS

- Raster data contouring
  - *Singularities*
    - *due to ambiguous configurations*
  - *(Degeneracies)*
    - *If fixed pattern used*

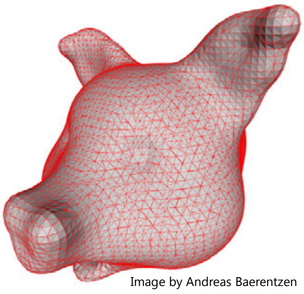


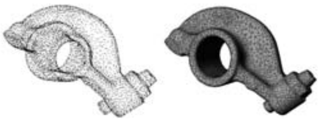
Image by Andreas Baerentzen

UPSTREAM APPLICATIONS

- Implicit function contouring
  - *Aliasing*
  - *(Topological noise) – if fixed pattern used*

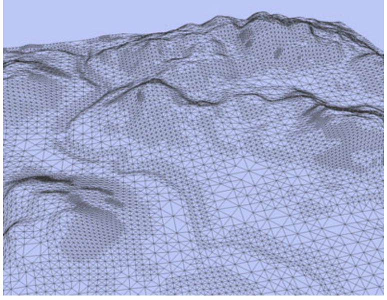
UPSTREAM APPLICATIONS

- Reconstruction from points
  - Methods available that do not introduce artifacts not already present in the data.
  - But many others might introduce
    - *(Holes)*
    - *(Gaps)*
    - *(Aliasing)*
    - *(Topological noise)*
  - *Even if certain sampling criteria are met that would allow for correct reconstruction in theory*



UPSTREAM APPLICATIONS

- Height field triangulation



UPSTREAM APPLICATIONS

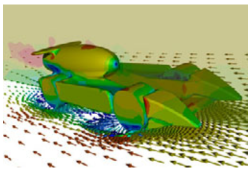
- Solid model boundary extraction
  - Singularities

UPSTREAM APPLICATIONS

Approach	noise	holes	gaps	intersections	degeneracies	singularities	topolog. noise	aliasing
Tessellation			**X**	**X**	x			
Depth image fusion				**X**	x	x		
Raster data contouring					x	**X**		
Implicit function contouring					x		x	**X**
Reconstruction from points		x	x				x	x
Height field triangulation								
Solid model boundary extract.						**X**		

DOWNSTREAM APPLICATIONS

- We consider prototypical requirements of a sample of the wide application spectrum
  - *Visualization*
  - *Modeling*
  - *Rapid Prototyping*
  - *Processing*
  - *Simulation*



DOWNSTREAM APPLICATIONS

Application Group	noise	holes	gaps	intersections	degeneracies	singularities	topolog. noise	aliasing
Visualization	x	**X**	x					x
Modeling		**X**	**X**		**X**	x	x	
Rapid Prototyping		**X**	**X**	**X**		**X**		
Processing	**X**	**X**	**X**	x	**X**	**X**	x	x
Simulation	**X**	**X**	**X**	**X**	**X**	**X**	**X**	x

REPAIR APPROACHES

- We distinguish between two types:
  - Local:
    - Handling defects individually by local modifications.
    - Low invasiveness, but only few guarantees.
  - Global:
    - Typically based on a complete remeshing.
    - High robustness, but often loss of detail.
    - More plausible ambiguity resolution possible.

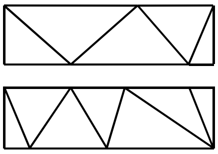
LOCAL APPROACHES
Gap closing

Gaps – Nature and origin

- Between connected components of a mesh; made of separated chains of edges
- Produced by tessellation, round-off, conversion errors, inaccurate trimming, …
- Usually long and narrow
- Most methods match gap boundaries by considering their spatial proximity

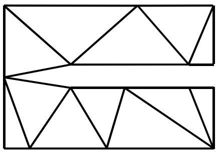
Proximity-based approaches

- Merge vertices within a prescribed distance [RW92]
  - to re-unite displaced but equivalent vertices.



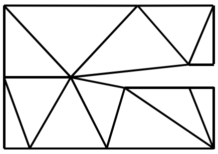
Proximity-based approaches

- Merge vertices within a prescribed distance [RW92]
  - to re-unite displaced but equivalent vertices.



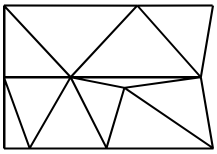
Proximity-based approaches

- Merge vertices within a prescribed distance [RW92]
  - to re-unite displaced but equivalent vertices.



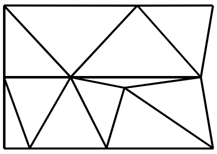
Proximity-based approaches

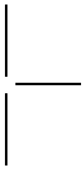
- Merge vertices within a prescribed distance [RW92]
  - to re-unite displaced but equivalent vertices.



Proximity-based approaches

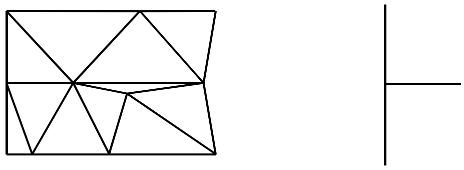
- Merge vertices within a prescribed distance [RW92]
  - to re-unite displaced but equivalent vertices.





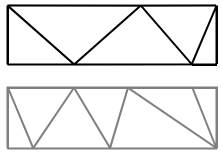
Proximity-based approaches

- Merge vertices within a prescribed distance [RW92]
  - to re-unite displaced but equivalent vertices.



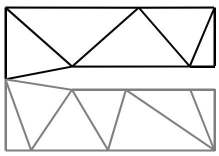
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



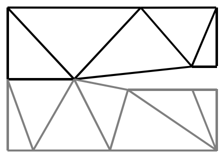
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



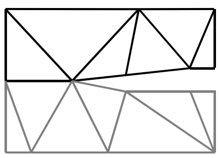
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



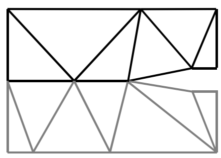
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



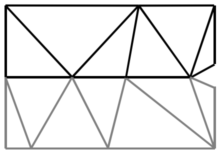
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



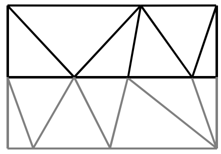
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



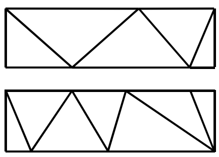
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



Proximity-based approaches

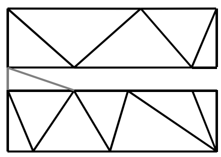
- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



“Stitching“

Proximity-based approaches

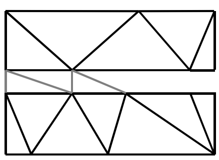
- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



“Stitching“

Proximity-based approaches

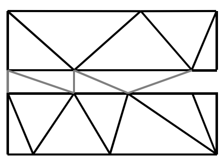
- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



“Stitching“

Proximity-based approaches

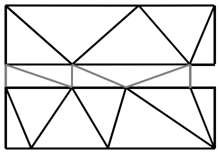
- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



“Stitching“

Proximity-based approaches

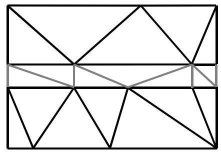
- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.



“Stitching“

Proximity-based approaches

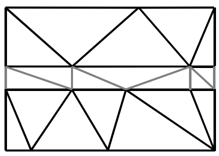
- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.

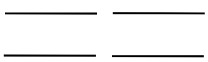


“Stitching“

Proximity-based approaches

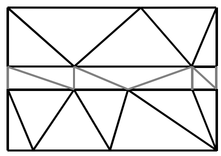
- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.
  - Start from closest pairs to resolve ambiguities.

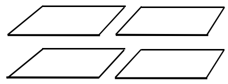




Proximity-based approaches

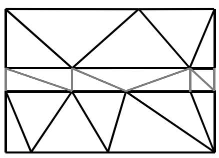
- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.
  - Start from closest pairs to resolve ambiguities.

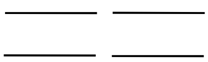




Proximity-based approaches

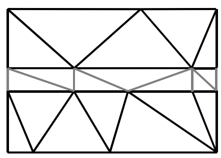
- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.
  - Start from closest pairs to resolve ambiguities.





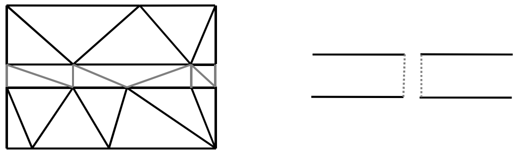
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.
  - Start from closest pairs to resolve ambiguities.



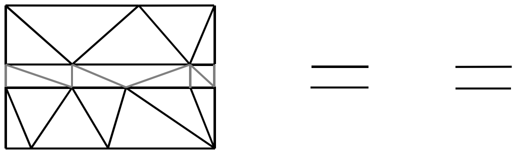
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.
  - Start from closest pairs to resolve ambiguities.



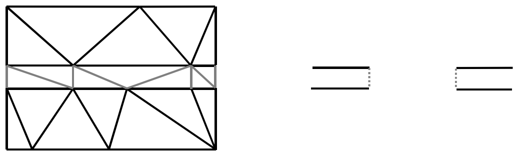
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.
  - Start from closest pairs to resolve ambiguities.



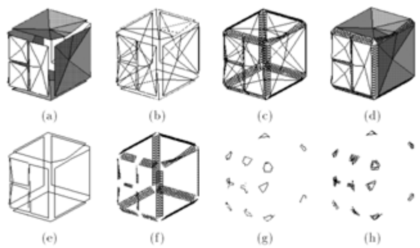
Proximity-based approaches

- Progressively “zip” pairs of boundary edge chains [SM95], [BK97]
  - Better control over topology.
  - Start from closest pairs to resolve ambiguities.



Advanced Gap Closing

- Several gaps may cross and meet
  - globally optimal matching of (parts of) boundary curves [BS95] instead of greedy



Negative Gaps

- Consider also “negative gaps”, i.e. overlapping patches, by clipping and merging [TL94]

Negative Gaps

- Consider also “negative gaps”, i.e. overlapping patches, by clipping and merging [TL94]

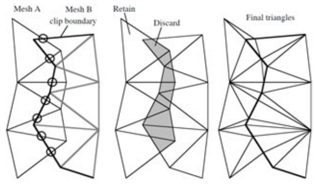
Negative Gaps

- Consider also "negative gaps", i.e. overlapping patches, by clipping and merging [TL94]

- general problem for „stitching"
- problem for „zippering" if overlap larger than triangles

Negative Gaps

- Consider also "negative gaps", i.e. overlapping patches, by clipping and merging [TL94]

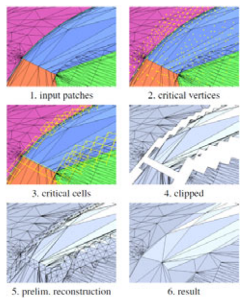


Issues of pairwise boundary stitching

- pairwise processing does not introduce singular edges
- some gaps remain when only reasonably resolvable into a non-manifold mesh
- Thus, some methods allow to produce non-manifolds to be able to close all the gaps [BNK02]
- Dynamic selection of zipping or stitching depending on gap width [PMR05]

Advanced Gap Closing

- Gaps may not be bounded by boundary edges
  - More general detection and resolution needed
  - Hybrid approach [BK05]
    - Remeshing in voxels surrounding the gaps
    - Output guaranteed intersection-free



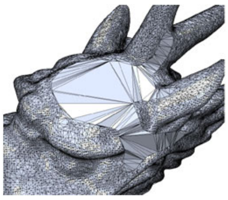
Summary Table – Gap Closing

Algorithm	Input requirements	Parameters	Potential new flaws
[Rock and Wozny 1992]	Very small gaps	Gap width	Intersections, degen., singularities
[Sheng and Meier 1995]	-	Gap width	Intersections, degen.
[Barequet and Kumar 1997]	-	Gap width	Intersections, degen.
[Turk and Levoy 1994]	Overlap	Gap threshold	Intersections, degen.
[Borodin et al. 2002]	-	-	Intersections, degen., singularities
[Patel et al. 2005]	-	-	Intersections, degen., singularities
[Bischoff and Kobbelt 2005]	-	Gap width, resolution	Degeneracies

LOCAL APPROACHES
Hole Filling

Hole Filling

- Early methods detect holes by looking for closed loops of boundary edges
- These "simple" holes can be patched by triangulating their boundary loops



3D Polygon Triangulation

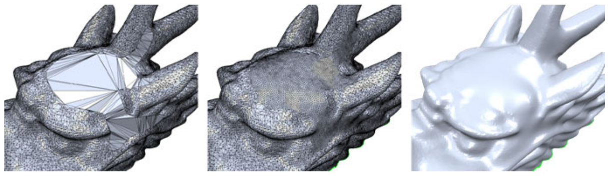
- Heuristics:
  - Minimal area, minimal dihedral angles
- Greedy triangulation [BW92; MD93; VPK05; RW97]
- Find optimum by Dynamic Programming [BS95; Lie03]

- Too coarse for large holes
- Some 3D polygons cannot be triangulated without self-intersections

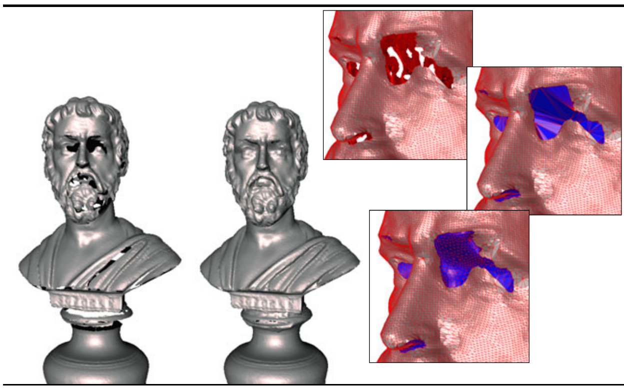
Beyond Triangulation

- For large holes, insert additional vertices within the triangulation while trying to:
  - meet Delaunay criterion [PS96]
  - reproduce the sampling density and achieve normal continuity [Lie03]
  - consider internal angles, dihedral angles, and areas [WWP10]
- Dynamic programming rather inefficient for very large holes (e.g. in high-res scans)

Beyond Triangulation



Beyond Triangulation



Beyond Triangulation

- For large holes, insert additional vertices within the triangulation while trying to:
  - meet Delaunay criterion [PS96]
  - reproduce the sampling density and achieve normal continuity [Lie03]
  - consider internal angles, dihedral angles, and areas [WWP10]
- Dynamic programming rather inefficient for very large holes (e.g. in high-res scans)

Other approaches

- Advancing front with Poisson equation [ZGL07]
- Radial Basis Functions [BPB06]
- NURBS fitting [KSI*07]
- Curvature energy minimization [Lev03; PMV06]
- Moving Least Squares projection [WO07; TC04]

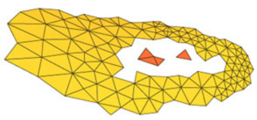
- Often robustness issues due to required boundary region parameterizations, hole boundary flattenings, control point setup, etc.

Self-intersections

- When additional vertices are inserted, methods may try to create intersection-free patches
  - [TC04] – After each triangle insertion check for intersections. Might fail in producing the complete patch.
  - [WLG03] – Randomized optimization by simulated annealing. Less failures, but still no guaranteed convergence to any plausible result.

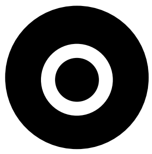
Issues

- Potentially new intersections
- Holes might have complex topologies
  - The algorithms cited consider one loop at a time. Not suitable for e.g. holes with “islands”

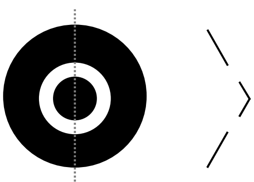
Issues

- Potentially new intersections
- Holes might have complex topologies
  - The algorithms cited consider one loop at a time. Not suitable for e.g. holes with “islands”



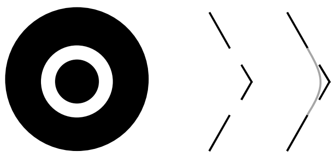
Issues

- Potentially new intersections
- Holes might have complex topologies
  - The algorithms cited consider one loop at a time. Not suitable for e.g. holes with “islands”



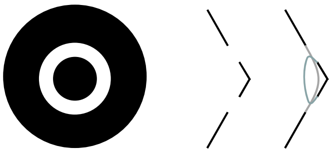
Issues

- Potentially new intersections
- Holes might have complex topologies
  - The algorithms cited consider one loop at a time. Not suitable for e.g. holes with “islands”



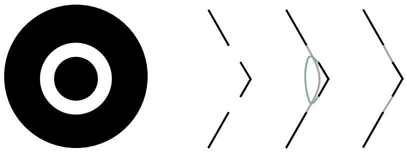
Issues

- Potentially new intersections
- Holes might have complex topologies
  - The algorithms cited consider one loop at a time. Not suitable for e.g. holes with "islands"



Issues

- Potentially new intersections
- Holes might have complex topologies
  - The algorithms cited consider one loop at a time. Not suitable for e.g. holes with "islands"



Exploit the volume

- Build a Constrained Delaunay Tetrahedralization
  - Input required to be free of self-intersections, singularities and degeneracies
- Using graph-cut techniques, proper facets of tetrahedra are selected to fill holes with multiple boundaries [PR05]
- Guarantee: intersection-free output

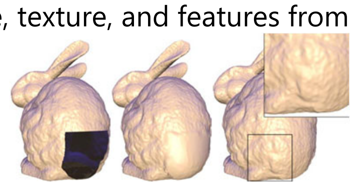
Summary Table – Hole Filling

Algorithm	Input requirements	Parameters	Intersect.-free
[Bøhn and Wozny 1992]	-	-	
[Mäkelä and Dolenc 1993]	-	-	
[Roth and Wibowoo 1997]	Roughly planar hole boundaries	-	
[Varnuska et al. 2005]	-	-	
[Barequet and Sharir 1995]	-	-	
[Liepa 2003]	-	-	
[Pfeifle and Seidel 1996]	-	-	
[Tekumalla and Cohen 2004]	-	-	X
[Wagner et al. 2003]	-	Sim. Anneal. Param.	X
[Podolak and Rusinkiewicz 2005]	No degen., intersect., singular.	-	X

LOCAL APPROACHES
Mesh Completion

Beyond smooth patches

- Reproducing morphological details can lead to more plausible patches
- *Mesh completion* algorithms attempt to solve this problem
- Copy structure, texture, and features from intact parts



from [Sharf et al. 2004]

Types of completion algorithms

- Mainly two classes:
  - Modification of template shapes
    - Use available geometry to select a proper "template" from a DB and adapt the latter
  - Creation of the missing geometry
    - Use available geometry to create detailed patches that complete the missing parts

Template-based completion

- Often require user suggestions to start the alignment, e.g. correspondences, feature markers, …
- Useful just for objects that can be clearly classified into one of few categories
  - human head scans [BV99; KHYS02; BMVS04]
  - bodies [ACP03; ASK*05]
  - teeth [KHYS02; SK02]

Template-based completion

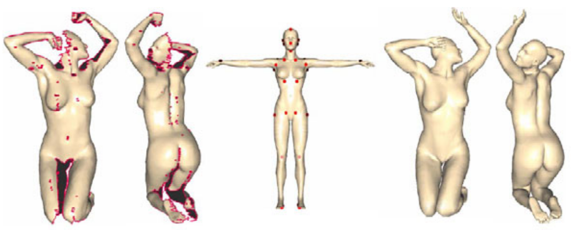


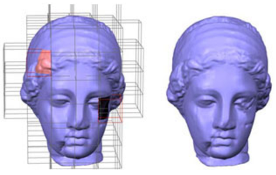
Image courtesy of Kraevoy and Sheffer

Inter vs Intra-shape similarities

- The necessary patches can be
  - copied from other parts of the same model [SACO04]
    - Useful if textures and features shall be replicated
  - synthesized according to the geometry of a set of meshes of the same class as the input [PMG*05]
    - To achieve correct global structure and topology

Patch-based Completion

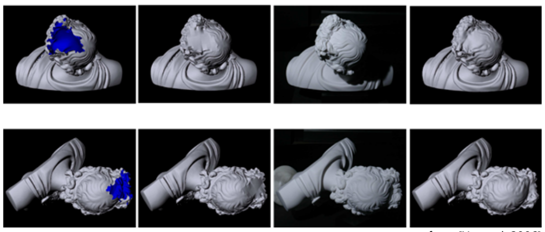
- Shape similarity measure
  - To find best region or patch to copy into hole region.
- Selection strategy:
  - Evaluate for a set of discrete locations/ orientations/scales [SACO04]
  - Evaluate on a per point basis [BSK05; BF05]



from [Sharf et al. 2004]

Photo-based Completion

- Infer information from additional photos
  - Shape-from-shading technique [XGR*06]
  - Photo-consistency measure [BWS*10]



from [Xu et al. 2006]

Creation of missing geometry

- Patches can be easily copied, but it's hard to merge them continuously with mesh
- Many algorithms work on point-based representations instead [SACO04; BSK05; BF05; PGSQ06; XZM*07]
  - Need to appropriately sample if input is mesh
  - Need to triangulate the resulting patches

Summary Table – Mesh Completion

Algorithm	Input requirements	Parameters	Potential new flaws
[Sharf et al. 2004]	- (point-based)	Resolution	(topo. noise, alias.)
[Bendels et al. 2005]	- (point-based)	Scale levels	(topo. noise, alias.)
[Breckon and Fisher 2005]	- (point-based)	Window Size	(topo. noise, alias.)
[Park et al. 2006]	- (point-based)	Resolution	(topo. noise, alias.)
[Xiao et al. 2005]	- (point-based)	Several ...	(topo. noise, alias.)
[Pauly et al. 2005]	-	Model database, keywords	Degeneracies, intersections
[Xu et al. 2006]	Roughly planar hole boundaries	Calibrated images	Degeneracies, intersections

LOCAL APPROACHES
Degeneracy and Self-Intersection Removal

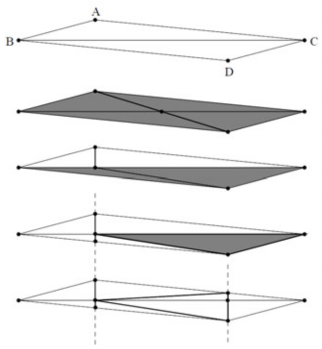
Types of input

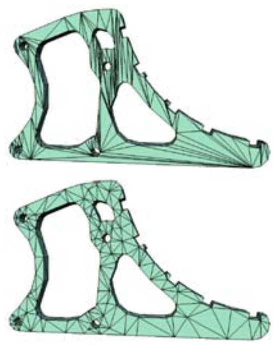
- Degeneracies (or near-degeneracies) are often the source of instabilities
- Algorithms that fix them might need to use robust geometric predicates
- Two types of input
  - Tessellated CAD models
  - Digitized models

Slicing CAD models

- Needle-like triangles are simply removed by collapsing the edge opposite to the degenerate corner
- Caps can be split into needles
  - To avoid loops, [BK01] employ a slicing technique
- When done, iterative edge collapses can simplify the model while removing all the needles

Slicing technique [BK01]



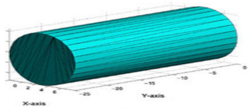


Treating raw digitized meshes

- Here we can count on a rather uniform and dense sampling [Att10]
- Needles can still be removed by collapsing the opposite edge
- Caps can be resolved by “swapping” the edge opposite to the flat corner
- Guaranteed to converge for exact degeneracies, not for near-degeneracies

Manifold meshes

- Keeping the mesh manifold while removing the needles might be impossible for non-exact degeneracies
- E.g. a long and thin cylinder might be tessellated with only triangles with angles below the given threshold



Self-intersections

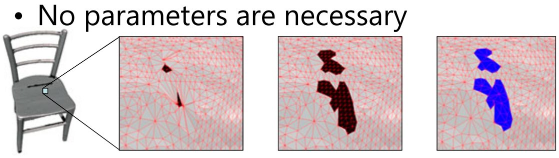
- Two problems
  - Detection
  - Resolution
- All-with-all intersection tests lead to quadratic complexity → unaffordable
- Need spatial subdivision to reduce the search space

Local remeshing

- [BK05] use a voxel grid to locate self-intersections efficiently
- If a voxel contains intersecting triangles, the surface within the voxel is remeshed
- Same process to locate and fix small gaps
- Modifications occur only near the flaws, thus the approach is local
- Useful to fix tessellated CAD patches with approximated trimming curves

Removing triangles

- In digitized meshes intersecting triangles are small and can be simply removed, and the resulting holes filled
- Approach used in [Att10], where several repairing tasks are performed in sequence
- No parameters are necessary



Robustness issues

- Finite precision may be insufficient to represent the intersection points
- [CK10] use an intermediate BSP representation
  - Fast and robust
- [GHH*03] use arbitrary precision arithmetic
  - More precise
  - Slower; requires more resources

Summary Table – Degeneracy and Self-Intersection Removal

Algorithm	fixes:	Input requirements	Parameters	GS	Accuracy
[Botsch and Kobbelt 2001]	D	manifold	Thr. angle		approx.
[Attene 2010]	D, S, H	-	Thr. angle		approx.
[Bischoff and Kobbelt 2005]	S, G	manifold	Tolerance, gap width	X	approx.
[Campen and Kobbelt 2010]	S	no boundary, no degeneracies	-	X	exact
[Granados et al. 2003]	S	-	-	X	exact

D = degenerate faces
S = self-intersections
H = holes
G = gaps

GS = guaranteed success

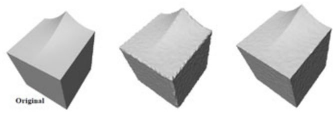
LOCAL APPROACHES
Sharp Feature Restoration

Interactive approach

- In [KB03], for each corrupted sharp edge, the user draws a “fishbone” structure (spine and orthogonal ribs)
- A tessellation of this structure replaces the original chamfer with a sharp patch
- Useful also to model arbitrary profiles to be swept along the edge

The EdgeSharpener approach

- Detect smooth regions by analyzing the dihedral angle at mesh edges [AFRS05]
- Create sharp features as intersections of planar extrapolations of smooth regions
- Suitable for meshes interpolating points of feature-insensitive sampling patterns
- Automatic

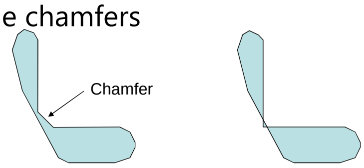


Other automatic approaches

- In their hole-filling algorithm [CC08] include a sharpness-dependent filter to reconstruct features
- In [Wan06] both sharp features and smooth blends between smooth regions are reconstructed
  - To differentiate between these two cases, the user is required to specify a parameter

Newly-introduced flaws

- All the methods discussed “add” or “remove” material” to reconstruct the features → potential self-intersections
- The EdgeSharpener method might produce degenerate triangles while splitting the chamfers



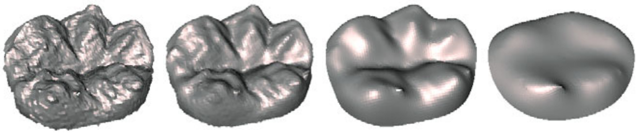
Summary Table – Sharp Features

Algorithm	Input requirements	Parameters	Potential new flaws
[Kobbelt and Botsch 2003]	manifold	interactive	self-intersections
[Attene et al. 2005]	manifold, no degeneracies	-	self-intersections, degeneracies
[Chen and Cheng 2008]	manifold, no degeneracies	-	self-intersections
[Wang 2006]	no noise, no degeneracies	two thresholds	self-intersections

LOCAL APPROACHES
Mesh Denoising

Laplacian smoothing

- Iterative algorithm
- For each iteration, compute the eventual position of each vertex as the center of mass of the neighbors
- Tends to “shrink” the shape



Non-shrinking methods

- λ/μ algorithm [Tau95] - a modification of the Laplacian smoothing
- For each iteration, two sub-iterations are performed
  - One inward diffusion, controlled by λ
  - One outward diffusion, controlled by μ
- Alternatively, [VMM99] push vertices toward original surface after each Laplacian iteration

Morphology-aware denoising

- Previous algorithms smooth everything
- If morphological features are important, [FDCO03] propose to use a bilateral filter as done in image processing
  - User need to set two parameters
- [JDD03] propose an alternative approach which is non-iterative
  - Can treat polygon soups as well

Denoising and sharpening

- In both [HP04] and [SRML07], smooth regions are denoised while potential sharp edges are actually sharpened
- For the case of mechanical/man-made objects, [FYP10] propose a specific approach that is more accurate

Newly introduced flaws

- All the described methods move vertices to new positions
- Some of them reconstruct sharp features, thus add material to the object
- Typically, no controls are performed to check that these modifications do not produce self-intersections or degeneracies

Summary table - Denoising

Algorithm	Properties	Input requirements	Parameters
[Taubin 1995]	N	closed manifold	λ, μ, n
[Fleishman et al. 2003]	N, F	manifold	σ_c, σ_s, n
[Jones et al. 2003]	N, F	-	σ_noise
[Hildebrandt and Polthier 2004]	N, S	manifold	λ, r
[Fan et al. 2010]	N, S	manifold	-n

N = noise removal
F = feature preservation
S = feature sharpening

All these methods might introduce degeneracies and self-intersections

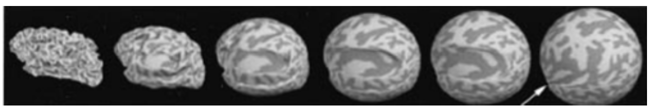
LOCAL APPROACHES
Topology Correction

Types of approaches

- Based on prior knowledge of the topology
  - e.g. Reconstruction of human cortex from MRI [XPR*02]: known to be genus 0
- Involving user interaction
  - [SLS*07] – Ask the user to resolve possible ambiguities
- Based on threshold parameters
  - [GW04] – Remove all handles smaller than a threshold size

Genus-0 surfaces

- [FLD01] inflate the input mesh (brain cortex) by alternating steps of Laplacian smoothing and radial projection (spherical parameteriz.)
- Folds are replaced by disk-like patches, and the parameterization is reversed
- Other methods: [SL01], [HXBNP02] (genus-N)



Removing small handles

- [ESV97] roll a sphere of radius α over the mesh and fill up all the regions that are not accessible to the sphere
- This removes tiny handles and tunnels, but also spoils concave edges and is unsuitable for meshes with boundary
- Appears to be extremely difficult to implement; robustness issues may arise

Topological Noise Removal

- [GW04] use a wavefront traversal to find if the mesh has local handles or tunnels (user-defined size)
- Non-separating cuts are identified and the mesh is cut and sealed along them
- [AF06] propose an accelerated method for digitized meshes which exploits wavefront splitting points

Newly introduced flaws

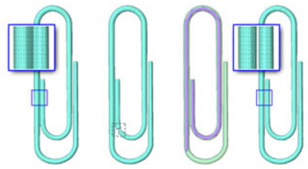
- All these methods add or remove material
- Typically, no checks are performed that these modifications do not produce self-intersections
- Other algorithms exploit explicit definition of the volume to avoid this problem
  - If you have a mesh, voxelization modifies it everywhere, so these must be considered global approaches

Voxel-based topology correction

- [WHDS04] use Reeb graphs to locate handles in voxelized shapes
- If handle is small (measured by short non-separating cycles), volumetric data is processed to remove it
- If model has numerous handles, topology-sensitive carving [SV03] is faster though less precise

Voxel-based topology correction (2)

- For huge voxelizations (e.g. 4096^3), [ZJH07] use discrete curve skeletons
- [JZH07] make it possible to actually edit the topology of an object so as to make it equivalent to that of a given target shape



Summary table - topology correction

Algorithm	Input requirem.	Parameters	Potential new flaws
[El-Sana and Varshney 1997]	no boundary	radius	self-intersect., aliasing
[Guskov and Wood 2001]	oriented manifold	threshold	self-intersections
[Fischl et al. 2001]	oriented manifold	(→ 0 handles)	self-intersections
[Attene and Falcidieno 2006]	-	threshold	self-intersections
[Shattuk and Lehay 2001]	no large holes	(→ 0 handles)	(aliasing)
[Han et al. 2002]	no large holes	(→ 0 handles)	(aliasing)
[Szymczak and Vanderhyde 2003]	no large holes	threshold	(aliasing)
[Wood et al. 2004]	no large holes	threshold	(aliasing)
[Zhou et al. 2007]	no large holes	two thresholds	(aliasing)
[Ju et al. 2007]	no large holes	target „shape"	(aliasing)

GLOBAL APPROACHES

- Approaches discussed so far are local
  - remove single defects (holes, singularities, self-intersections, ...) mainly individually.
- Absence of individual defects not required for their own sake:
  - part of greater requirement for manifoldness.
- Achieving this by sequential local operations is extremely difficult:
  - new defects can be introduced
  - ambiguities are hard to resolve in a local manner.

GLOBAL APPROACHES

- Global repair methods can be advantageous in this regard.
  - consider mutual relation of defects for better ambiguity arbitration.
  - possibly employ intermediate volumetric representation:
    - Guarantees that the result is a manifold surface of some solid.
    - Disadvantage: often complete conversion and remeshing necessary → invasive, loss of detail

GLOBAL APPROACHES

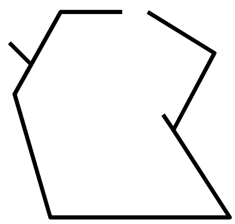
- Intermediate volumetric representation
  → the repair task boils down to deciding which parts of the volume are inside and outside.
- We can group the global methods by how this decision is performed and by their input requirements
  - Input without significant gaps and holes
  - Input with normal or orientation information
  - Arbitrary input

GLOBAL APPROACHES

- Input without significant gaps and holes
  - Rasterization into voxel grid representation,
  - Determination of inside/outside volume by flood-filling,
    - from given seed points [OSD97]
    - from a point at infinity [ABA02]
  - Reconversion to polygon mesh by contouring.
  - Not possible for gaps or holes beyond voxel size.

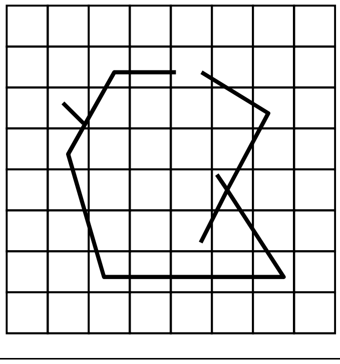
GLOBAL APPROACHES

Input mesh



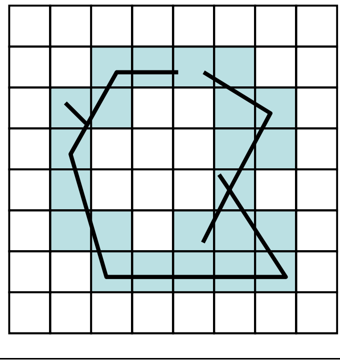
GLOBAL APPROACHES

Voxelization



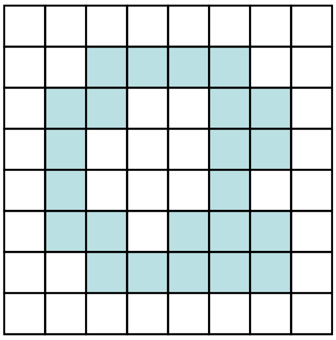
GLOBAL APPROACHES

Voxelization



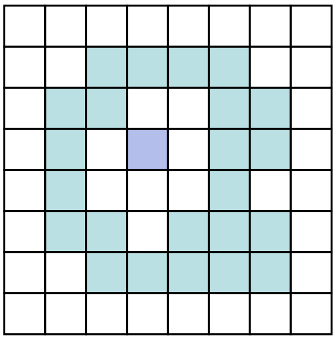
GLOBAL APPROACHES

Voxelization



GLOBAL APPROACHES

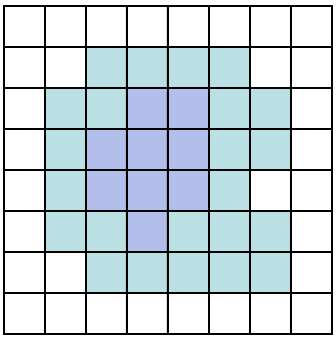
Interior seed



[OSD97]

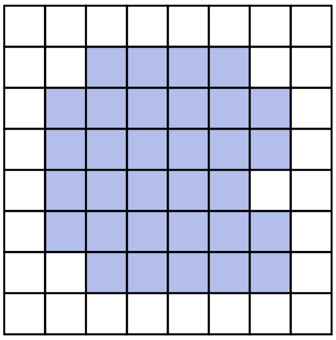
GLOBAL APPROACHES

Flooding



[OSD97]

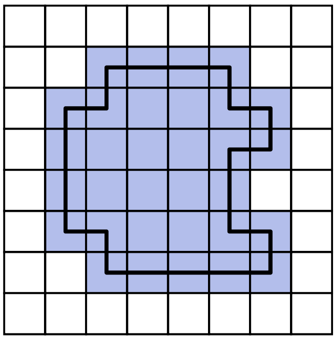
GLOBAL APPROACHES



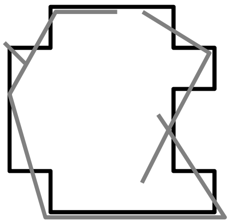
[OSD97]

GLOBAL APPROACHES

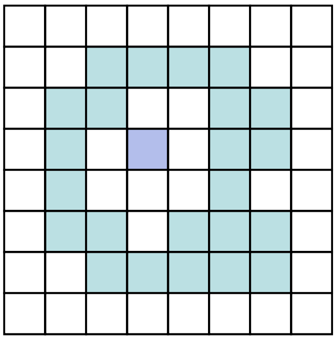
Boundary
extraction



GLOBAL APPROACHES

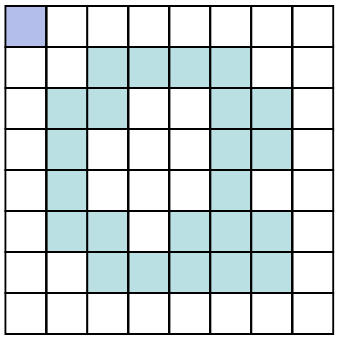


GLOBAL APPROACHES



GLOBAL APPROACHES

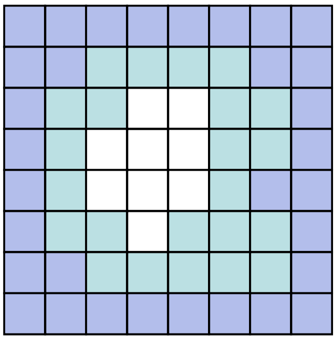
Exterior seed



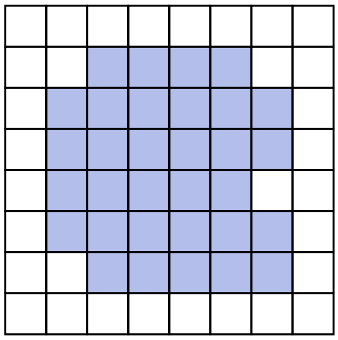
[ABA02]

GLOBAL APPROACHES

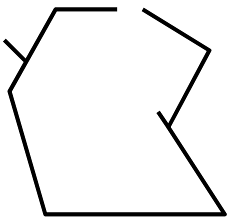
Flooding



GLOBAL APPROACHES

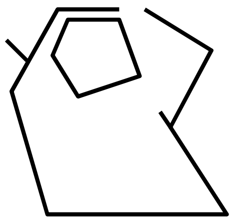


GLOBAL APPROACHES

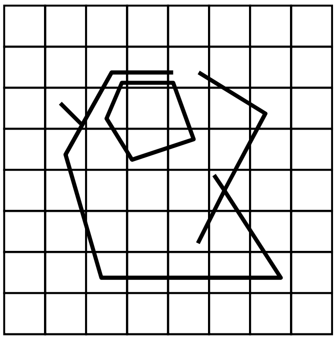


GLOBAL APPROACHES

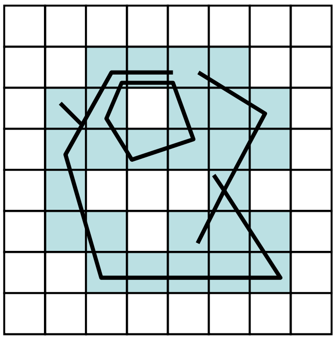
Internal void



GLOBAL APPROACHES

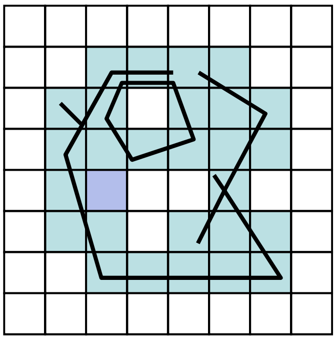


GLOBAL APPROACHES



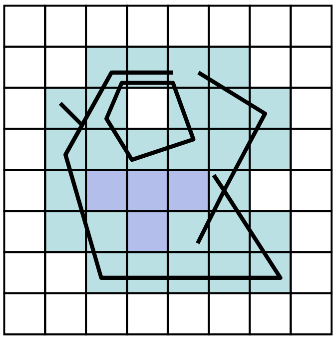
GLOBAL APPROACHES

Interior seed



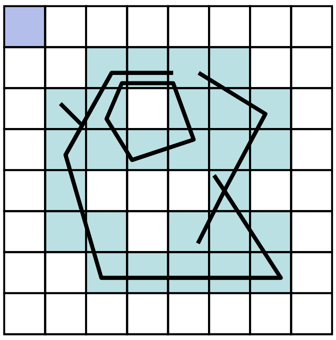
GLOBAL APPROACHES

Void preserved

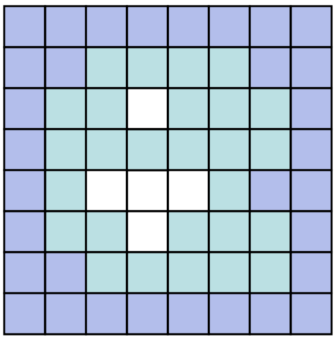


GLOBAL APPROACHES

Exterior seed

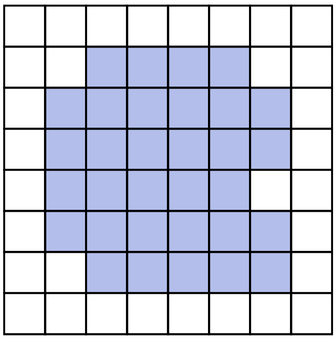


GLOBAL APPROACHES



GLOBAL APPROACHES

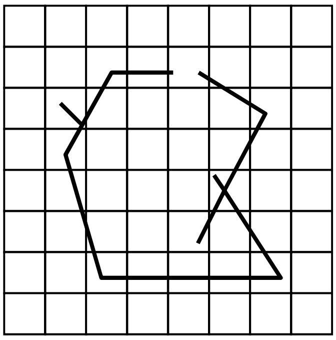
Void lost



GLOBAL APPROACHES

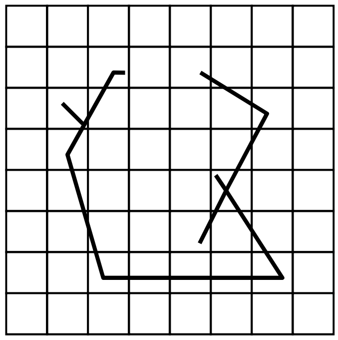
- Input without significant gaps and holes
  - Rasterization into voxel grid representation,
  - Determination of inside/outside volume by flood-filling,
    - from given seed points [OSD97]
    - from a point at infinity [ABA02]
  - Reconversion to polygon mesh by contouring.
  - Not possible for gaps or holes beyond voxel size.

GLOBAL APPROACHES

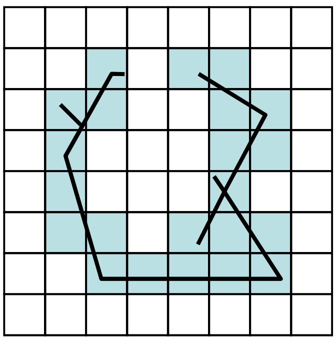


GLOBAL APPROACHES

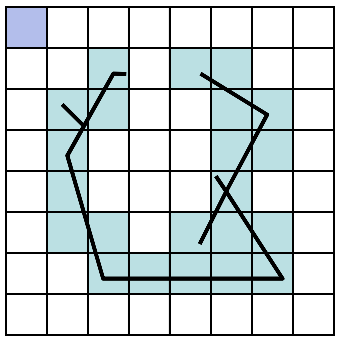
Larger hole



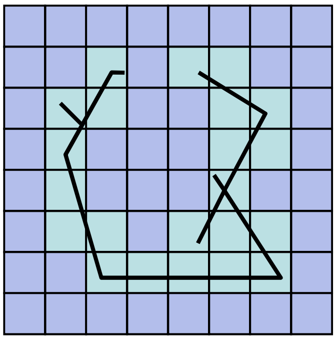
GLOBAL APPROACHES



GLOBAL APPROACHES



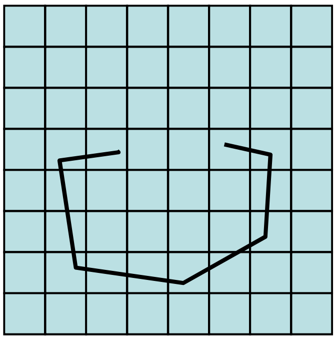
GLOBAL APPROACHES



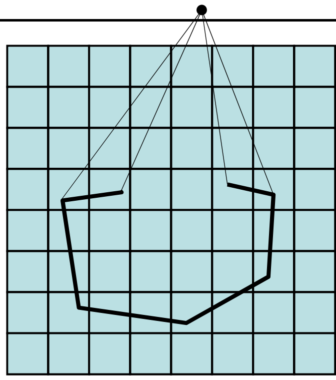
GLOBAL APPROACHES

- Input with orientation information
  - often available due to the acquisition process (e.g. line-of-sight of the laser scanner).
  - Inside/outside decision by:
    - Line-of-sight carving [CL96]
      - + line-of-light carving [FIMK07]
    - Diffusion-based propagation [DMGL02]
      - + feature sensitivity [GLWZ06] [Mas04]
    - Surface area minimization [SI03] [SI08]

GLOBAL APPROACHES

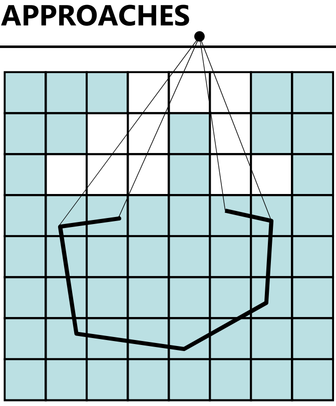


GLOBAL APPROACHES



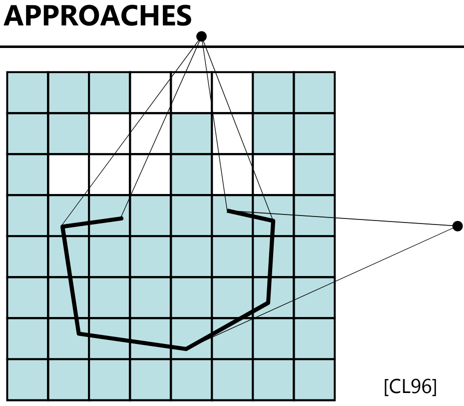
[CL96]

GLOBAL APPROACHES



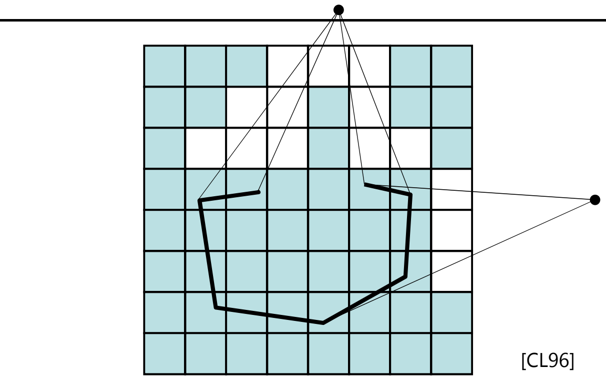
[CL96]

GLOBAL APPROACHES



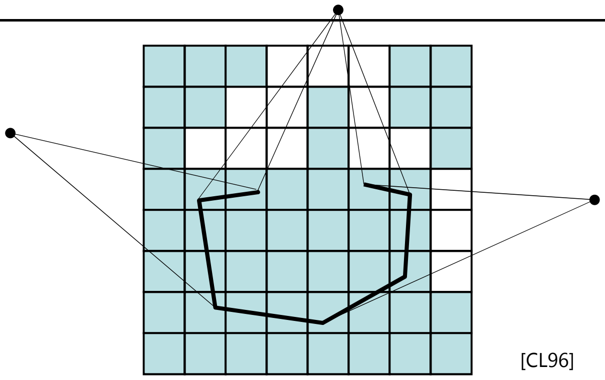
[CL96]

GLOBAL APPROACHES



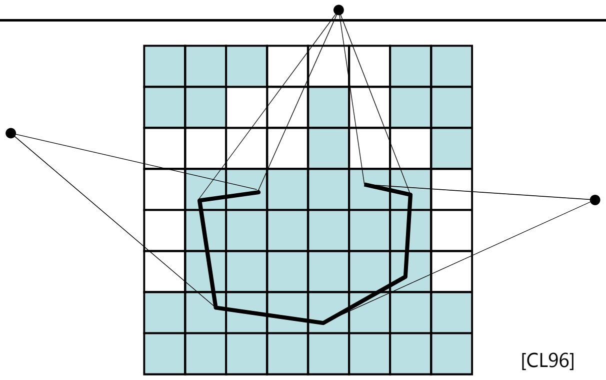
[CL96]

GLOBAL APPROACHES



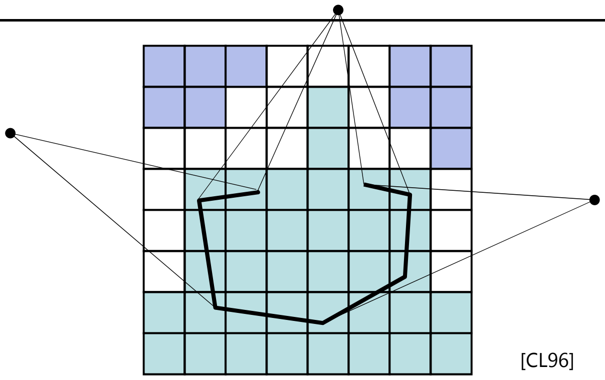
[CL96]

GLOBAL APPROACHES



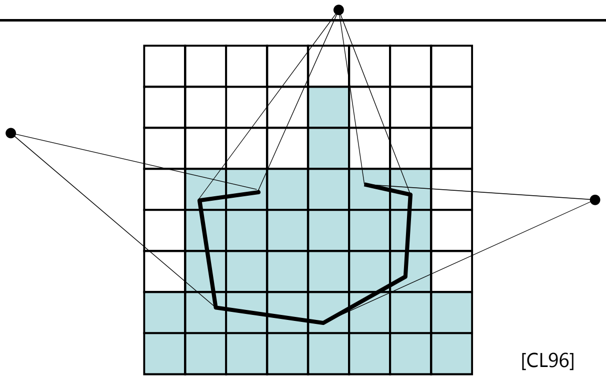
[CL96]

GLOBAL APPROACHES



[CL96]

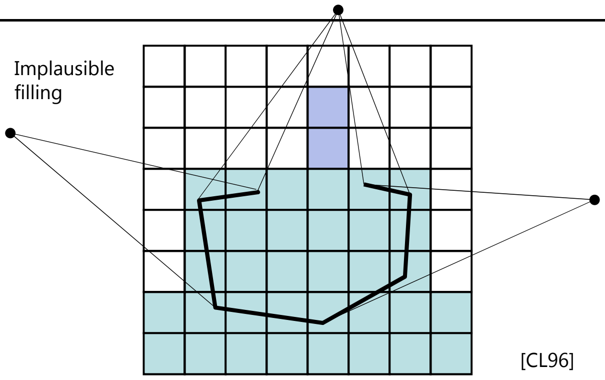
GLOBAL APPROACHES



[CL96]

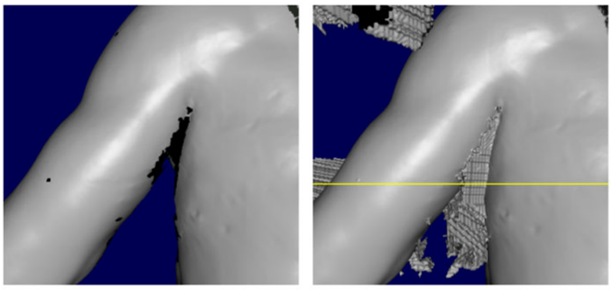
GLOBAL APPROACHES

Implausible filling



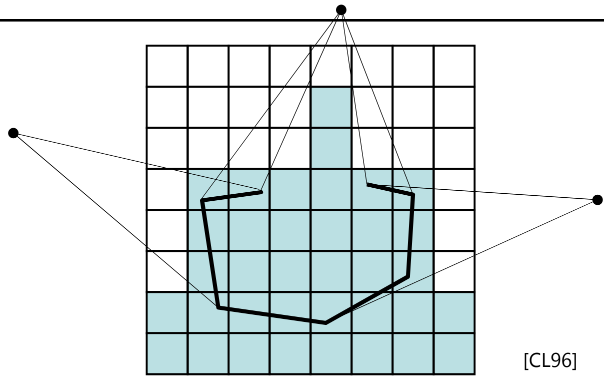
[CL96]

GLOBAL APPROACHES



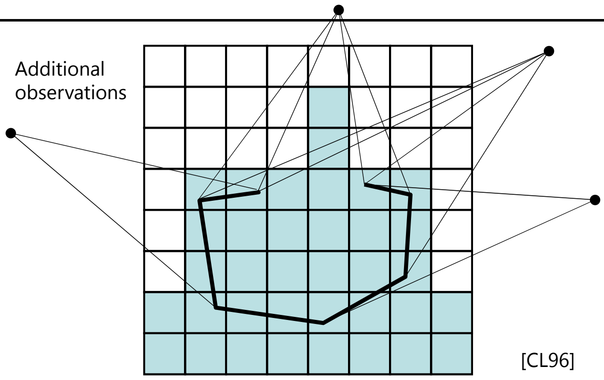
from [Davis et al. 2002]

GLOBAL APPROACHES



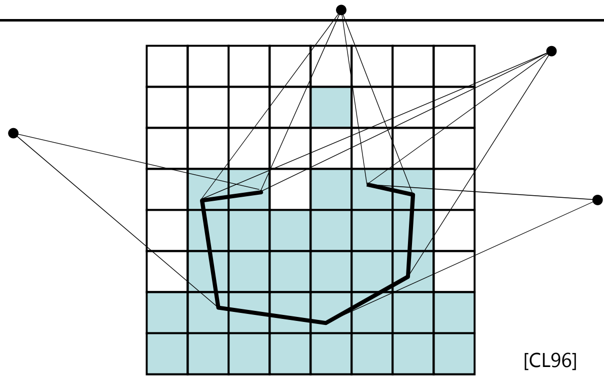
[CL96]

GLOBAL APPROACHES



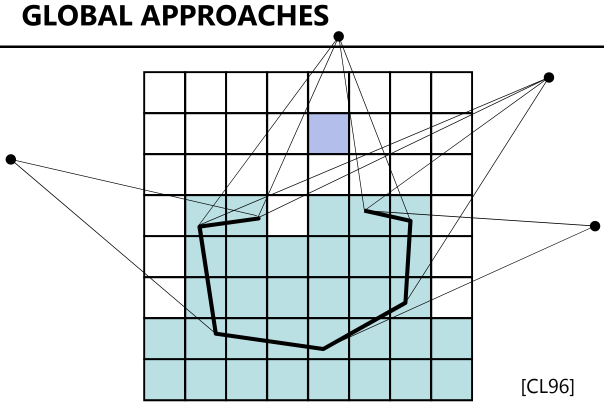
[CL96]

GLOBAL APPROACHES



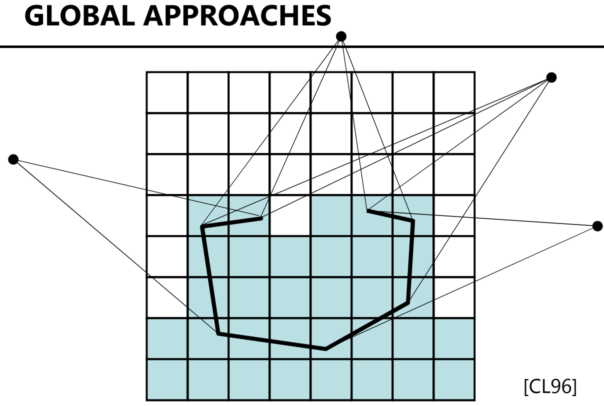
[CL96]

GLOBAL APPROACHES



[CL96]

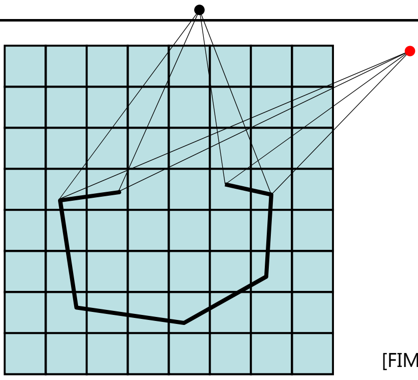
GLOBAL APPROACHES



[CL96]

GLOBAL APPROACHES

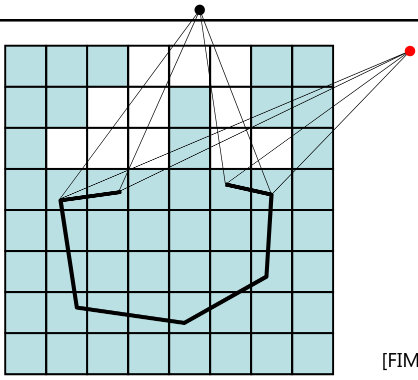
Exploit
line-of-light



[FIMK07]

GLOBAL APPROACHES

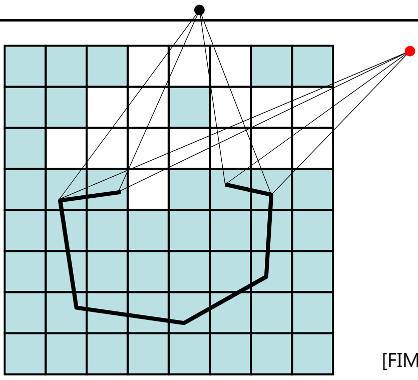
Exploit
line-of-light



[FIMK07]

GLOBAL APPROACHES

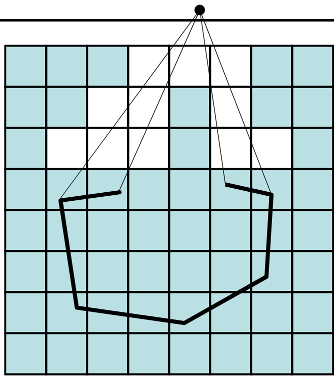
Exploit
line-of-light



[FIMK07]

GLOBAL APPROACHES

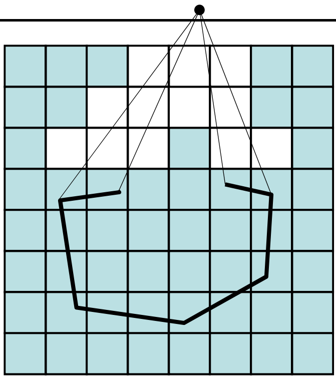
Discrete
area
minimization



[SI08]

GLOBAL APPROACHES

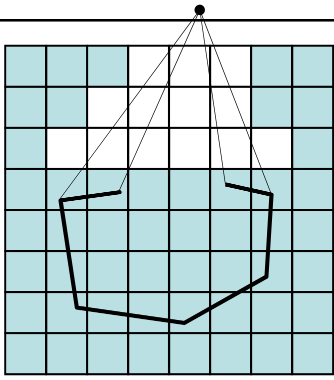
Discrete
area
minimization



[SI08]

GLOBAL APPROACHES

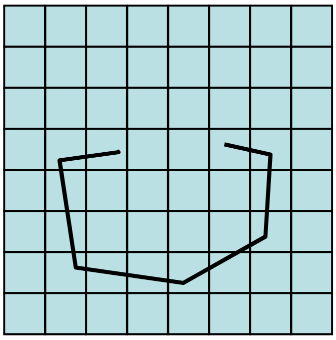
Discrete
area
minimization



[SI08]

GLOBAL APPROACHES

Without
cam/light
information

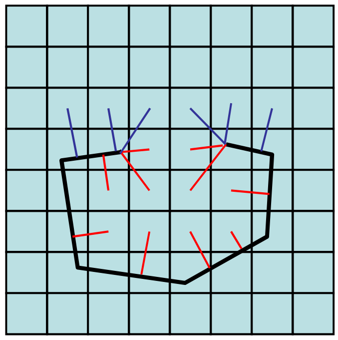


[SI03]

GLOBAL APPROACHES

Without
cam/light
information

Orientation
of nearest
surface

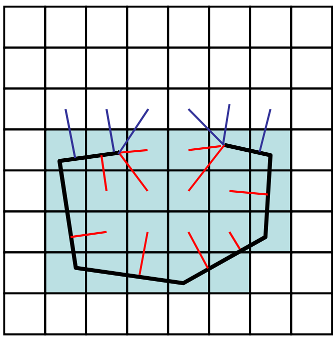


[SI03]

GLOBAL APPROACHES

Without
cam/light
information

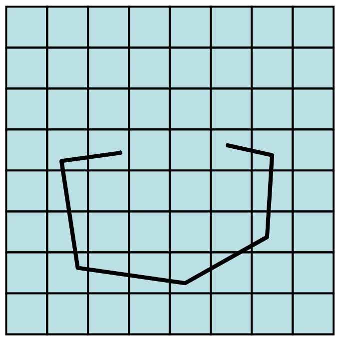
Orientation
of nearest
surface



[SI03]

GLOBAL APPROACHES

Without
cam/light
information

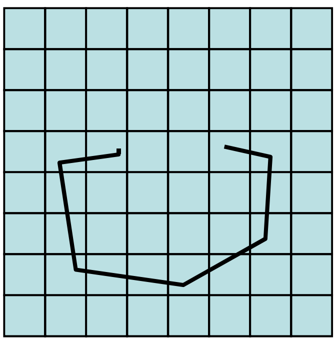


[SI03]

GLOBAL APPROACHES

Without
cam/light
information

Unclean
boundaries

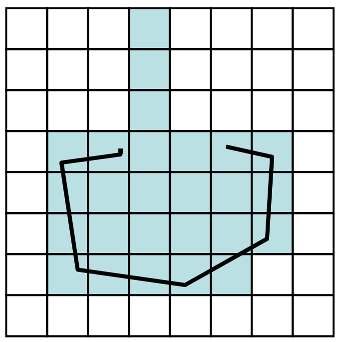


[SI03]

GLOBAL APPROACHES

Without
cam/light
information

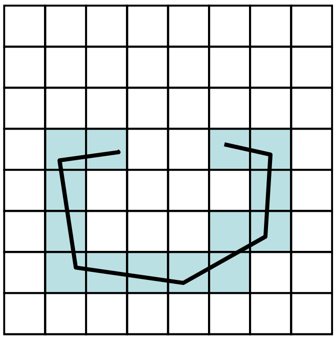
Unclean
boundaries



[SI03]

GLOBAL APPROACHES

Diffusion

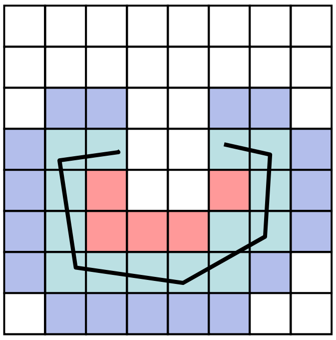


[DMGL02]

GLOBAL APPROACHES

Diffusion

Local distance field

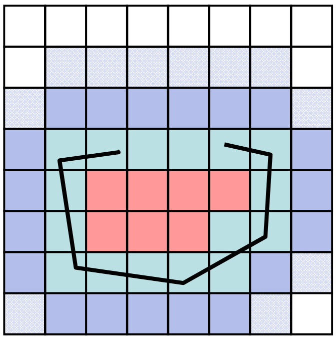


[DMGL02]

GLOBAL APPROACHES

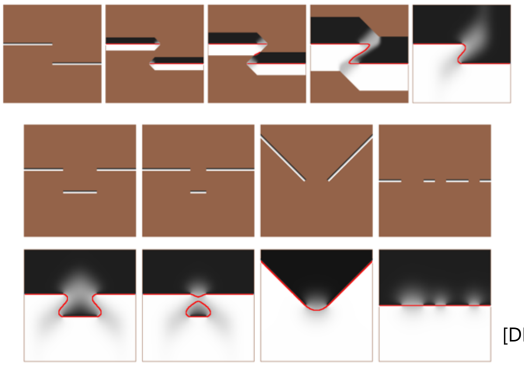
Diffusion

Local distance field



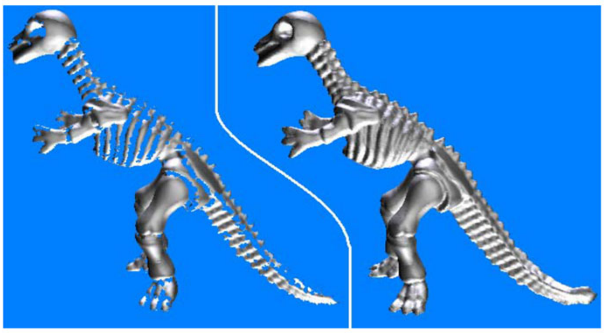
[DMGL02]

GLOBAL APPROACHES



[DMGL02]

GLOBAL APPROACHES



from [Guo et al. 2006]

GLOBAL APPROACHES

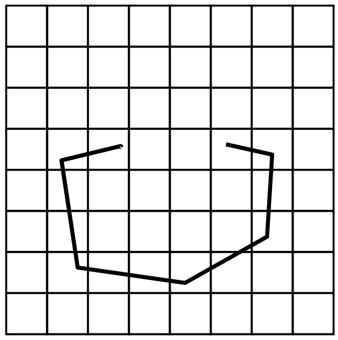
- Arbitrary Input
  - Various inside/outside decision principles:
    - Parity Counting
    - Ray Stabbing
    - Boundary Loop Patching
    - Morphology & Flooding
    - Graph Cutting
    - Membrane Shrinking
    - Optimization of global consistency functional

GLOBAL APPROACHES

- Parity Counting / Ray Stabbing
  - Consider intersections of rays with the object [NT03].
  - Combine findings from multiple ray directions to be less affected by holes and larger gaps [NT03].

GLOBAL APPROACHES

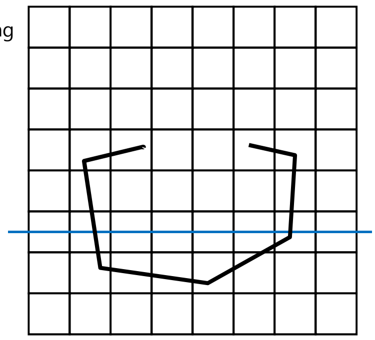
Ray Stabbing



[NT03]

GLOBAL APPROACHES

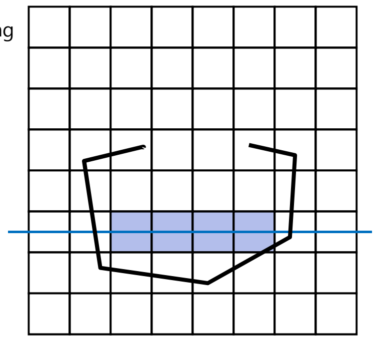
Ray Stabbing



[NT03]

GLOBAL APPROACHES

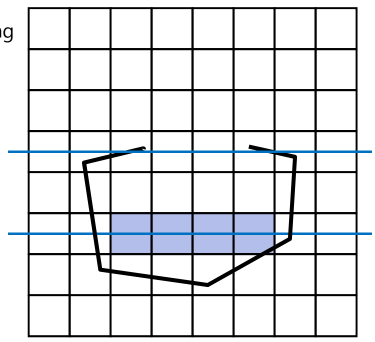
Ray Stabbing



[NT03]

GLOBAL APPROACHES

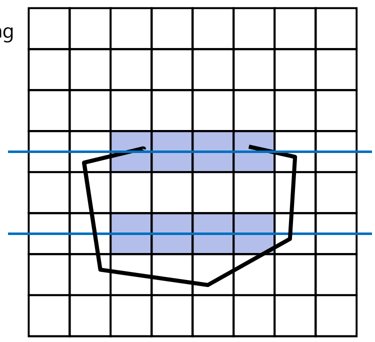
Ray Stabbing



[NT03]

GLOBAL APPROACHES

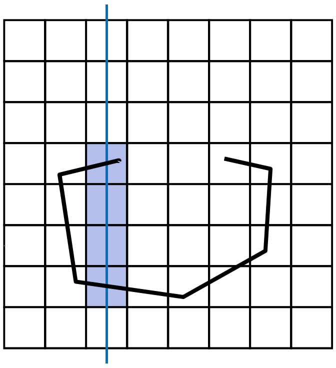
Ray Stabbing



[NT03]

GLOBAL APPROACHES

Ray Stabbing



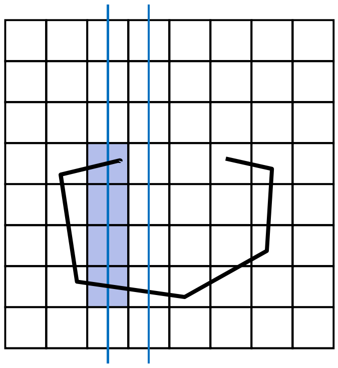
[NT03]

GLOBAL APPROACHES

Ray Stabbing

Problem:
holes
-> voting by several ray directions

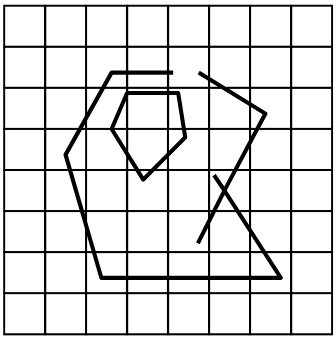
Varying plausibility of fillings



[NT03]

GLOBAL APPROACHES

Parity Counting

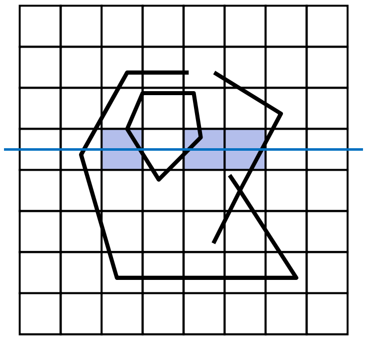


[NT03]

GLOBAL APPROACHES

Parity Counting

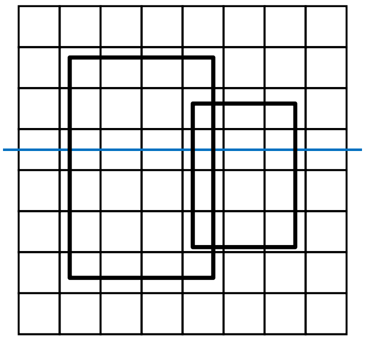
-> handles voids



[NT03]

GLOBAL APPROACHES

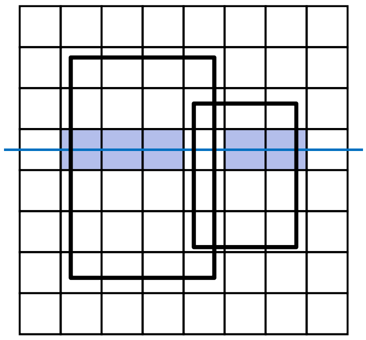
Parity Counting



[NT03]

GLOBAL APPROACHES

Parity Counting

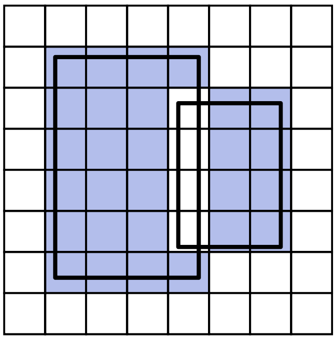


[NT03]

GLOBAL APPROACHES

Parity
Counting

-> also
artifact voids

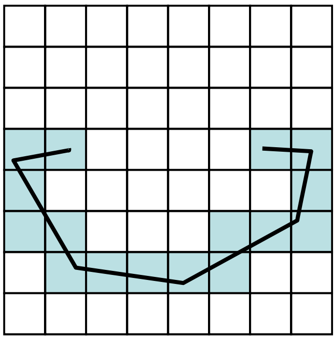


[NT03]

GLOBAL APPROACHES

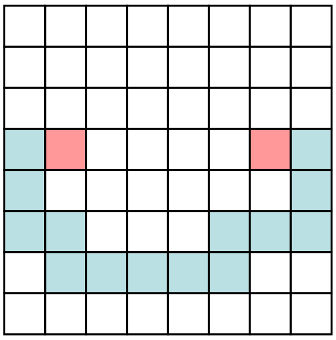
- Boundary Loop Patching
  - Detect holes and islands in rasterized version
  - Add patches (using XOR to prevent intersections)

GLOBAL APPROACHES



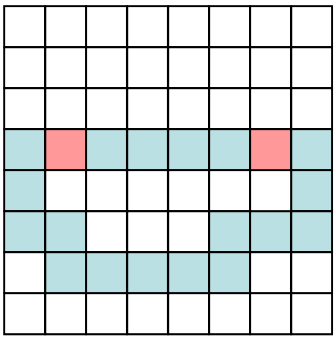
[Ju04]

GLOBAL APPROACHES



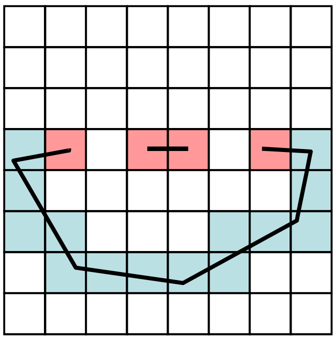
[Ju04]

GLOBAL APPROACHES



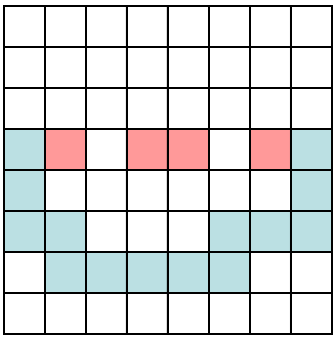
[Ju04]

GLOBAL APPROACHES



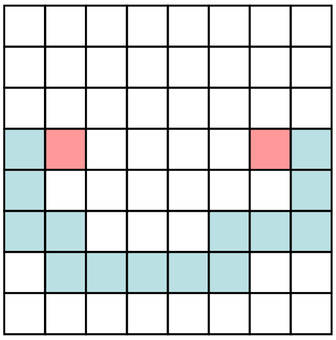
[Ju04]

GLOBAL APPROACHES



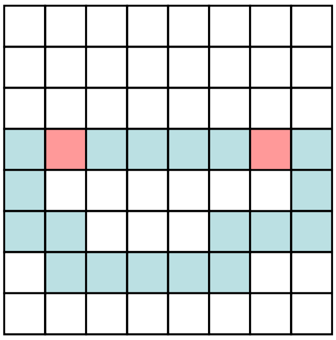
[Ju04]

GLOBAL APPROACHES



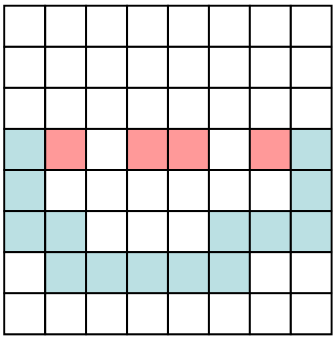
[Ju04]

GLOBAL APPROACHES



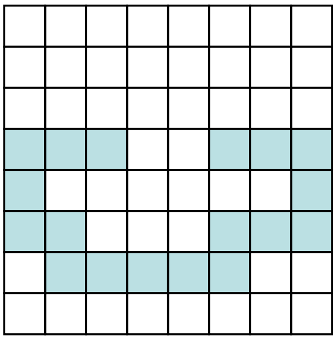
[Ju04]

GLOBAL APPROACHES



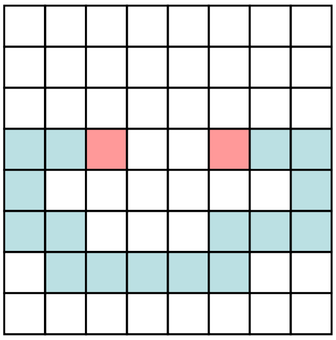
[Ju04]

GLOBAL APPROACHES



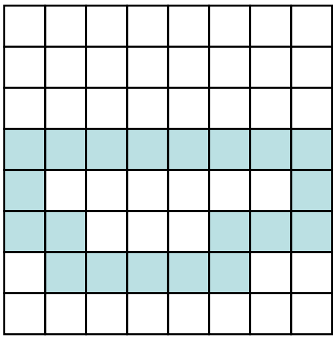
[Ju04]

GLOBAL APPROACHES



[Ju04]

GLOBAL APPROACHES

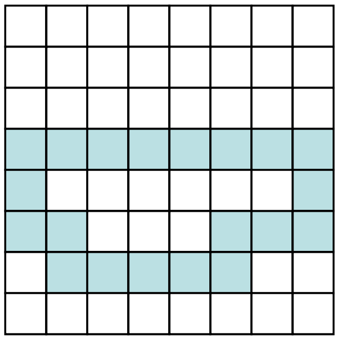


[Ju04]

GLOBAL APPROACHES

Does not work for gaps

Intersections in the input can lead to non-manifold output

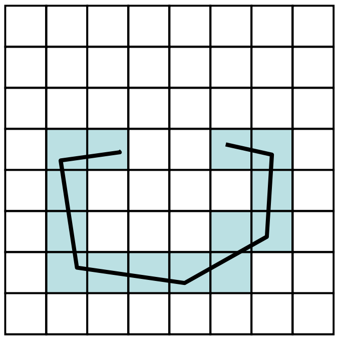


[Ju04]

GLOBAL APPROACHES

- Morphology
  - Does not rely on explicit hole boundary loop detection.
  - Closing operations to fill holes and gaps, flood-filling to determine outside [BPK05].
  - Improve final surface smoothness using graph-cut [HK06].

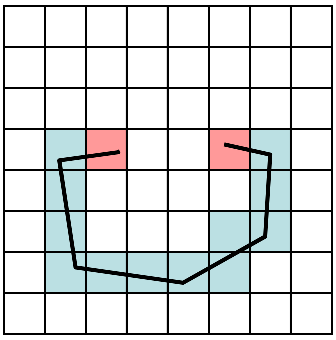
GLOBAL APPROACHES



[BPK05]

GLOBAL APPROACHES

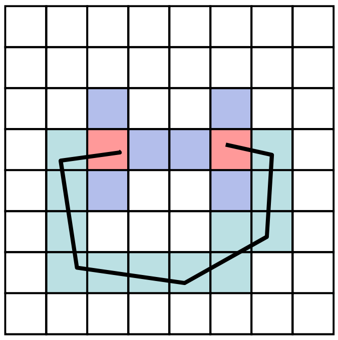
Dilation



[BPK05]

GLOBAL APPROACHES

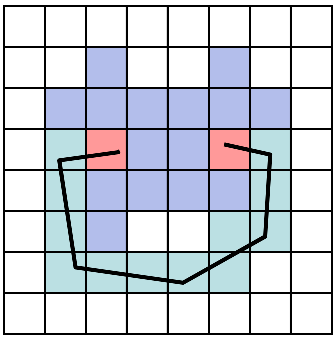
Dilation



[BPK05]

GLOBAL APPROACHES

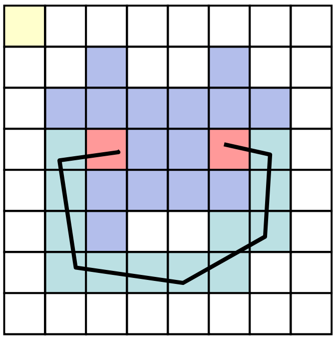
Dilation



[BPK05]

GLOBAL APPROACHES

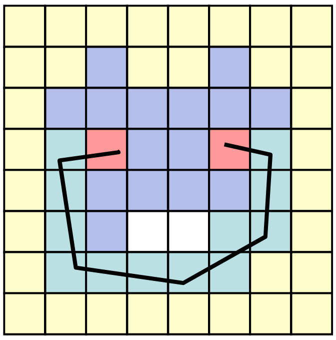
Flooding



[BPK05]

GLOBAL APPROACHES

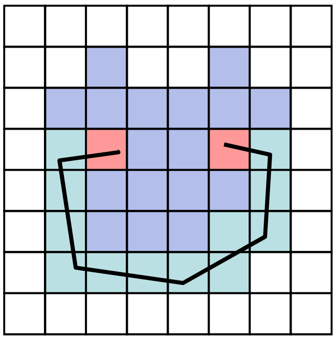
Flooding



[BPK05]

GLOBAL APPROACHES

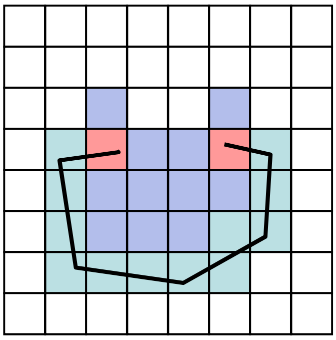
Flooding



[BPK05]

GLOBAL APPROACHES

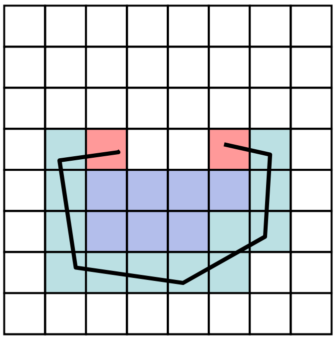
Erosion



[BPK05]

GLOBAL APPROACHES

Erosion



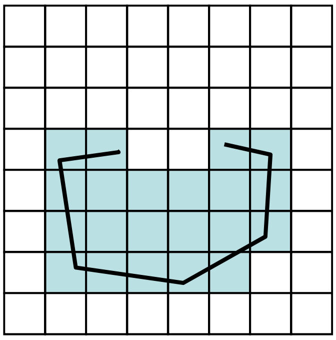
[BPK05]

GLOBAL APPROACHES

Result

Finally smoothing applied to hole region

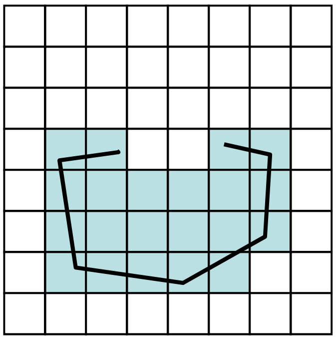
Alternative: graph-cut approach to find „nice" fillings [HK06]



GLOBAL APPROACHES

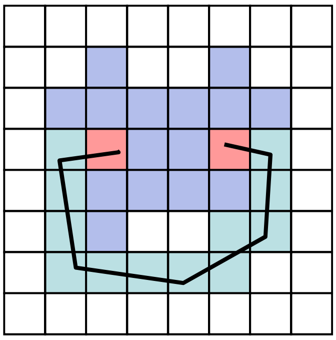
Result

Problems:
- fills cavities
- outer hull



GLOBAL APPROACHES

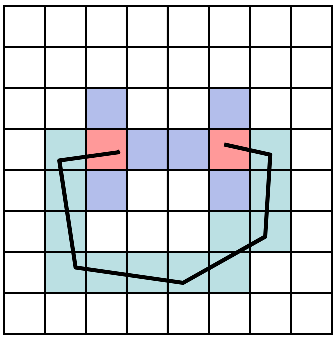
OR: Topology-preserving erosion



[BK05]

GLOBAL APPROACHES

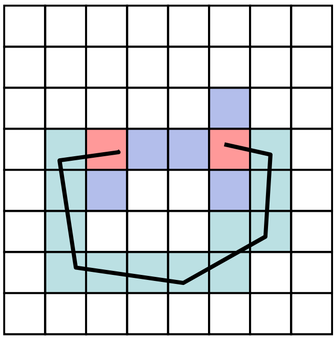
OR: Topology-preserving erosion



[BK05]

GLOBAL APPROACHES

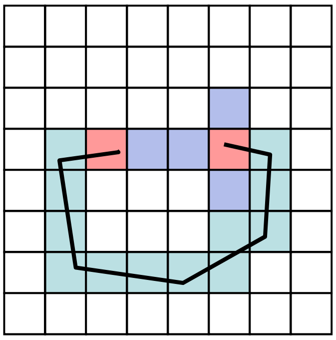
OR: Topology-preserving erosion



[BK05]

GLOBAL APPROACHES

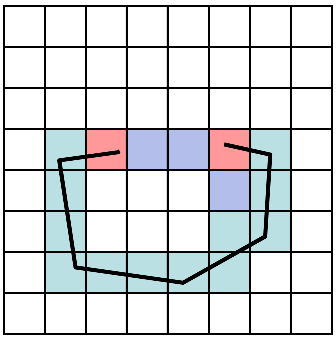
OR: Topology-preserving erosion



[BK05]

GLOBAL APPROACHES

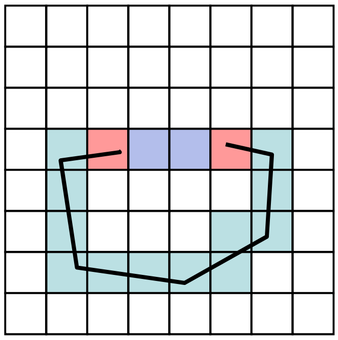
OR:
Topology-
preserving
erosion



[BK05]

GLOBAL APPROACHES

OR:
Topology-
preserving
erosion

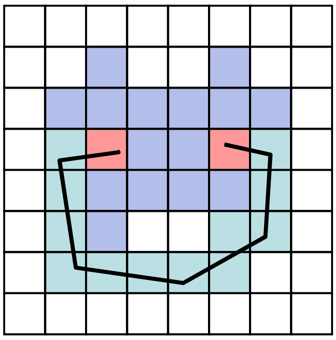


[BK05]

GLOBAL APPROACHES

OR:
Topology-
preserving
erosion

Problem:
amount of
dilation

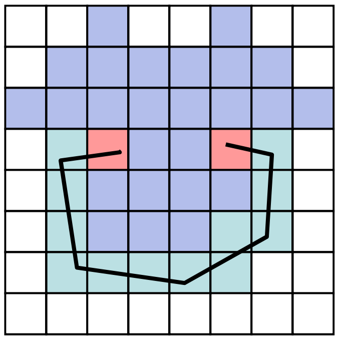


[BK05]

GLOBAL APPROACHES

OR:
Topology-
preserving
erosion

Problem:
amount of
dilation

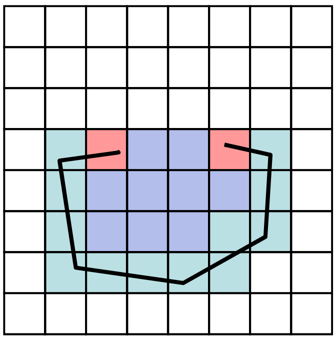


[BK05]

GLOBAL APPROACHES

OR:
Topology-
preserving
erosion

Problem:
amount of
dilation

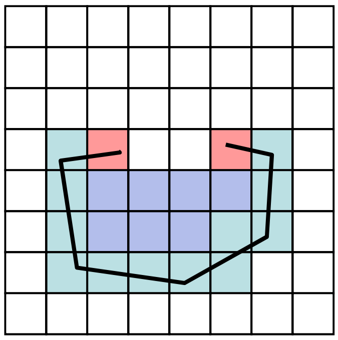


[BK05]

GLOBAL APPROACHES

OR:
Topology-
preserving
erosion

Problem:
amount of
dilation

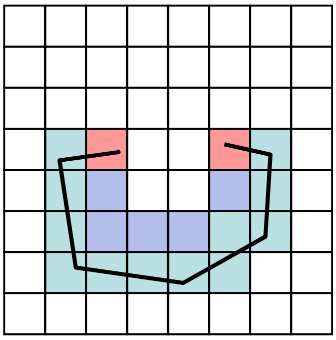


[BK05]

GLOBAL APPROACHES

OR:
Topology-preserving erosion

Problem: amount of dilation

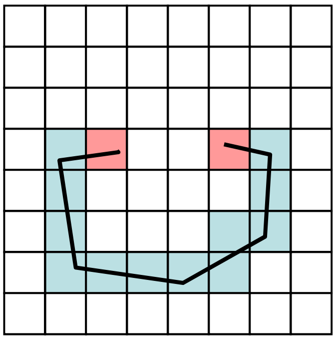


[BK05]

GLOBAL APPROACHES

OR:
Topology-preserving erosion

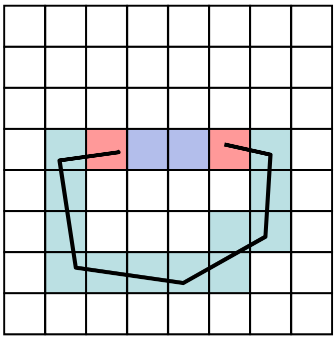
Problem: amount of dilation



[BK05]

GLOBAL APPROACHES

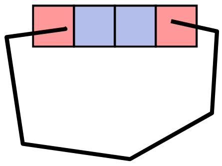
Variant: hybrid, structure-preserving



[BK05]

GLOBAL APPROACHES

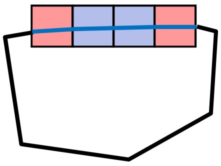
Variant: hybrid, structure-preserving



[BK05]

GLOBAL APPROACHES

Variant: hybrid, structure-preserving



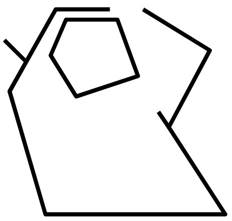
[BK05]

GLOBAL APPROACHES

- Global Optimization [MF97]
  - Surface-aligned volumetric representation (BSP-based)
  - Determine optimal inside/outside labels for cells such that the output surface maximally conforms with the (partial) input.
  - The geometry of the resulting hole-filling patches is rather random and can be unpleasing in case of larger holes.

GLOBAL APPROACHES

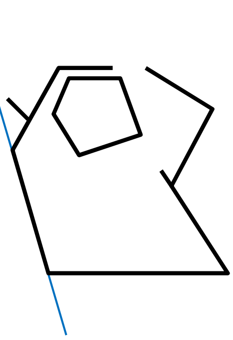
Build aligned
space
decomposition



[MF97]

GLOBAL APPROACHES

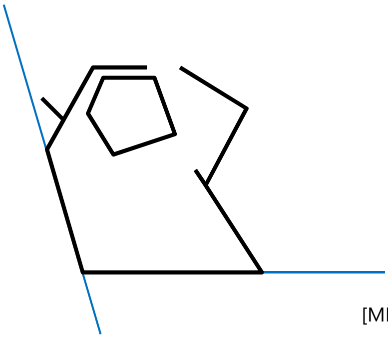
Build aligned
space
decomposition



[MF97]

GLOBAL APPROACHES

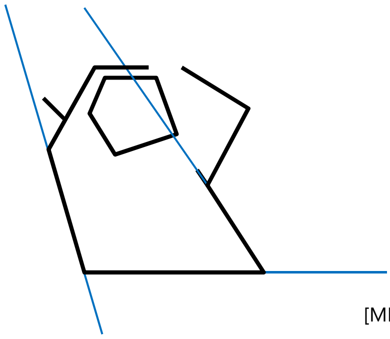
Build aligned
space
decomposition



[MF97]

GLOBAL APPROACHES

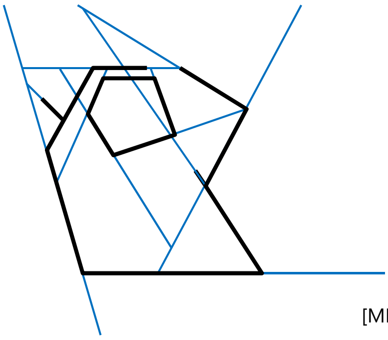
Build aligned
space
decomposition



[MF97]

GLOBAL APPROACHES

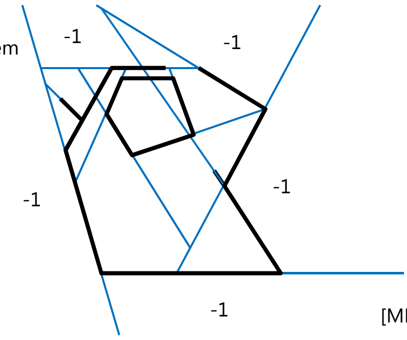
Build aligned
space
decomposition



[MF97]

GLOBAL APPROACHES

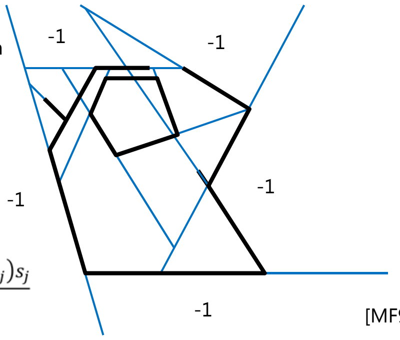
Setup global
Equation system



[MF97]

GLOBAL APPROACHES

Setup global
Equation system

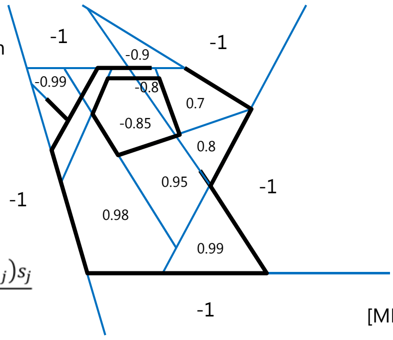


$$s_i = \frac{\sum_j (t_{i,j} - o_{i,j}) s_j}{A_i}$$

[MF97]

GLOBAL APPROACHES

Solve global
Equation system

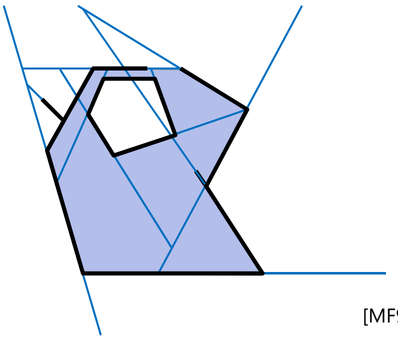


$$s_i = \frac{\sum_j (t_{i,j} - o_{i,j}) s_j}{A_i}$$

[MF97]

GLOBAL APPROACHES

Extract
pos./neg.
interface



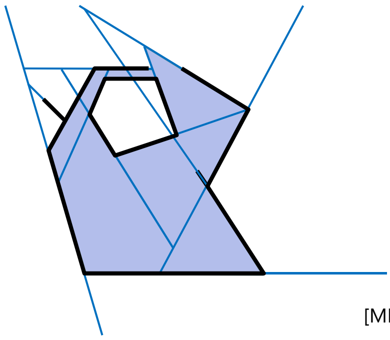
[MF97]

GLOBAL APPROACHES

Hole fillings
unstable

Input geometry
is preserved

Handles voids



[MF97]

GLOBAL APPROACHES

Possible variations:

- Smoothing of hole fillings
  - constrained to cells to avoid intersections
- Local cell decomposition at holes
  - Requires absence of intersections and singularities in input
- Incorporation of user constraints
  - to interactively correct output topology

[PR05]

Summary table – global approaches

Algorithm	**Input requirem.**	**Signing method**
[Oomes et al. 1997]	no significant holes/gaps	flood-filling
[Andújar et al. 2002]	no significant holes/gaps	flood-filling
[Curless and Levoy 1996]	oriented range meshes	line-of-sight
[Furukawa et al. 2007]	oriented range meshes	line-of-sight/light
[Davis et al. 2002]	oriented	normals + diffusion
[Sagawa and Ikeuchi 2008]	oriented	normals + area minimization
[Nooruddin and Turk 2003]	-	parity counting, ray stabbing
[Ju 2004]	(no significant gaps)	hole patching + parity counting
[Bischoff et al. 2005]	-	morphology + flood-filling
[Hornung and Kobbelt 2006]	-	morphology + graph-cut
[Murali and Funkhouser 1997]	(no significant holes)	global sign optimization

REPAIRING WORKFLOWS
An example for raw digitized meshes

Raw digitized meshes

- We can assume that:
  - Samples are rather uniformly spaced
  - Model is densely sampled (opposed to sparse tessellated NURBS)
- What is the typical input?
  - An indexed face set, possibly non manifold, self-intersecting, with degenerate faces, holes, topological noise, …
- How do I fix all these defects?
  - Global approach -> unnenessary distortion also where the model has no defects (defects are sparse !)
    - Filters out sharp features
    - Low distortion requires too many triangles

A repairing pipeline

- Sequence of local approaches
- Assumes that the input is a raw digitized mesh
- Creates a valid watertight polyhedral surface
- Works in two successive phases:
  - Topology reconstruction
  - Geometry correction

Topology reconstruction: step 1

- Convert the indexed face set to a simplicial complex

1) Triangulate non-triangular facets while loading (only simply connected, but this is normal in raw digitized meshes)
2) Create a (initially empty) list L of edges and, for each triangle <i,j,k> insert in L its three bounding edges <i,j>, <j,k> and <k,i>
3) Sort L lexicographically, i.e. if e_1 = <i,j> and e_2 = <k,n>
   $e_1 \leq e_2$ iff $i < k$ OR ($i = k$ AND $j \leq n$)
4) Two triangles are adjacent iff they induce consecutive edges in the sorted list L.

Topology reconstruction: step 2

- Convert the simplicial complex to a valid triangle mesh (i.e. manifold and oriented)

1) Run the cut&stitch algorithm (Gueziec et al., 2001)
   - Duplicate singular vertices and edges
2) Orient the mesh consistently
   1) Select a 'seed' triangle (e.g. the topmost one) and orient it
   2) Propagate the orientation to neighboring triangles
   3) Cut the mesh at non-consistently oriented pairs of triangles

Topology reconstruction: step 3

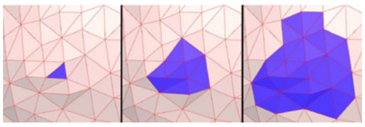
- Convert the manifold and oriented mesh to a single watertight mesh

1) If the mesh could not be oriented (i.e. cuts were necessary) this phase cannot take place -> failure
2) Otherwise, delete all the smallest connected components and fill the holes using Liepa's algorithm
   1) Smallest components are computed by counting their triangles (we assumed that the sampling is rather uniform)
   2) Smallest components include possible 'isles'
   3) The patches used to fill the holes may intersect other parts of the surface

Simplicial Neighborhoods

- For the “geometry correction” phase, we make use of the notion of simplicial neighborhood

1) The simplicial neighborhood N(t) is the set of all the simplexes which share at least a vertex with the triangle ‘t’
2) The i’th order simplicial neighborhood Ni(t) is defined as N(N(...N(N(t))...)), with ‘i’ nested levels

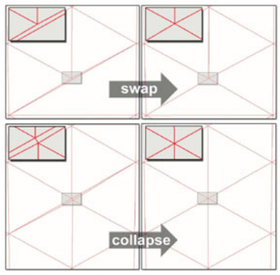


Geometry correction: step 1

- Remove (nearly) degenerate triangles

Require: A combinatorial manifold $\mathcal{M}$ and an integer threshold $max\_iterations$
Ensure: A combinatorial manifold $\mathcal{M}'$ and a status notice (success/failure)

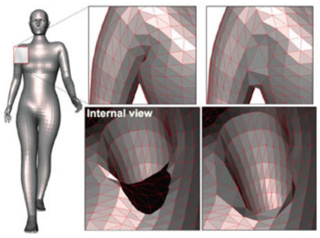
$\mathcal{M}' := \mathcal{M}$
Let S be the set of all the triangles of $\mathcal{M}'$
for $k = 1$ to $max\_iterations$ **do**
  Run the swap/collapse algorithm within S
  Let T be the set of degeneracies in S untreatable due to topological constraints
  if $T = \emptyset$ **then**
    terminate with success /* $\mathcal{M}'$ *is degeneracy free* */
  end if
  Let R be the union of the $k^{th} - order$ simplicial neighborhoods of the t_is $\in T$
  Remove R from $\mathcal{M}'$
  Remove possible disconnected components from $\mathcal{M}'$
  Patch the remaining gaps with a new set P of triangles
  $S := P$
end for



Geometry correction: step 2

- Remove intersecting triangles

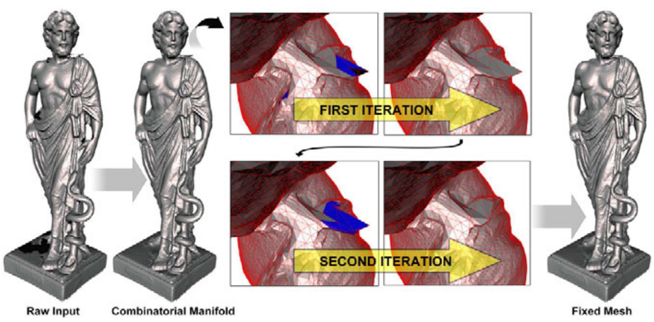
$\mathcal{M}' := \mathcal{M}$
Let S be the set of all the triangles of $\mathcal{M}'$
Let G be a uniform 100^3 voxel grid tightly enclosing $\mathcal{M}'$
for $k = 0$ to $max\_iterations$ **do**
  Let H be the set of voxels intersecting at least a triangle of S
  Check for triangle-triangle intersections within each voxel of H
  Let T be the set of intersecting triangles detected above
  if $T = \emptyset$ **then**
    terminate with success /* $\mathcal{M}'$ *is not self-intersecting* */
  end if
  Let R be the union of the $k^{th} - order$ simplicial neighborhoods of all $t \in T$
  Remove R from $\mathcal{M}'$
  Remove possible disconnected components from $\mathcal{M}'$
  Patch the remaining gaps with a new set P of triangles
  $S := P$
end for



Geometry correction: iteration

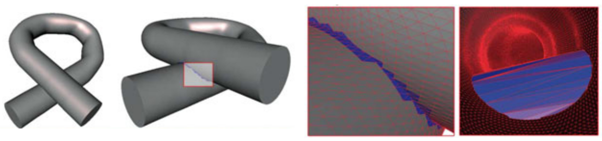
- While patching holes to remove self-intersections, new degenerate or nearly degenerate triangles may appear
- So, after step 2 we check for degeneracies and, if any, we repeat steps 1 and 2, until no more degeneracies are left
- This is guaranteed to converge only when exact degeneracies are removed

Example



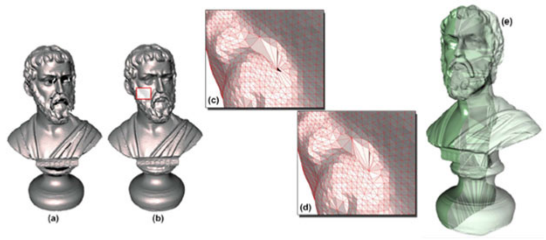
“Pathological” cases

- The method is not guaranteed to succeed in all the cases
- We have run it on hundreds of digitized models, it never failed -> good heuristics
- We had to synthesize a specific model to make it fail



Applications

- The fixed model can be converted to an explicit solid (i.e. a tetrahedral mesh). This is required e.g. for simulation.



DISCUSSION & OUTLOOK

- Widely varying hardness of repair tasks depending on the defects involved.
  - e.g. consistently orienting faces is easily formalized and solved,
  - but filling complex holes or plausibly removing intersections requires a non-trivial approach and intricate case-by-case study due to ambiguities.
- Despite the vast number of existing techniques, there is room for future investigation, especially for the *hard* cases.

DISCUSSION & OUTLOOK

- Minimally Invasive, but Guaranteeing and Global
  - Hybrid methods that are as accurate as local methods but ensure correctness like global ones.

DISCUSSION & OUTLOOK

- Minimally Invasive, but Guaranteeing and Global
  - Hybrid methods that are as accurate as local methods but ensure correctness like global ones.
- High-Level Interaction incorporating Meta-Knowledge
  - Missing data always implies ill-posedness, automatic heuristics have limits → exploit qualified knowledge of the user through intuitive interaction metaphors.

DISCUSSION & OUTLOOK

- Minimally Invasive, but Guaranteeing and Global
  - Hybrid methods that are as accurate as local methods but ensure correctness like global ones.
- High-Level Interaction incorporating Meta-Knowledge

from [Sharf et al. 2007]

DISCUSSION & OUTLOOK

- Minimally Invasive, but Guaranteeing and Global
  - Hybrid methods that are as accurate as local methods but ensure correctness like global ones.
- High-Level Interaction incorporating Meta-Knowledge
  - Missing data always implies ill-posedness, automatic heuristics have limits → exploit qualified knowledge of the user through intuitive interaction metaphors.
- Vertical Integration to Repair Workflows
  - Local methods often treat one defect type → suitable sequencing of methods necessary.

AVAILABLE REPAIR TOOLS

- Several repair tools are freely available
  - Implementing one or multiple of the covered methods.
- Up-to-date information and references to these tools are available at:

http://www.meshrepair.org

ACKNOWLEDGEMENTS

- Support to this work was given by:
  - **EU FP7 project VISIONAIR (contract N. 262044)**
  - CNR project "Multimodal and Multidimensional Content & Media" – activity N. ICT.P10.009
  - MIUR-PRIN Project N. 2009B3SAFK_002
  - DFG Cluster of Excellence UMIC (German Research Foundation grant DFG EXC 89)