The Ball Pivoting Algorithm (BPA)

Michael H. Nguyen

Gliederung

- **Einführung**
  - Typische 3D-Dokumentation(3D-Modelling-Pipeline)
  - Anwendungsbeispiele (Archäologie, Bauforschung, Denkmalpflege)
  - Angestrebtes Ziel d. BPA
- **Grundlagen/Begriffsdefinitionen**
  - Interpolation
  - Meshing(allg.)
  - ...
- **Der Algorithmus (BPA)**
  - Sprachl. Beschreibung des Algorithmus
  - Implementation
- **Beispiele**
- **Fazit (Vor- und Nachteile des BPA)**
- **Quellen**

Einführung

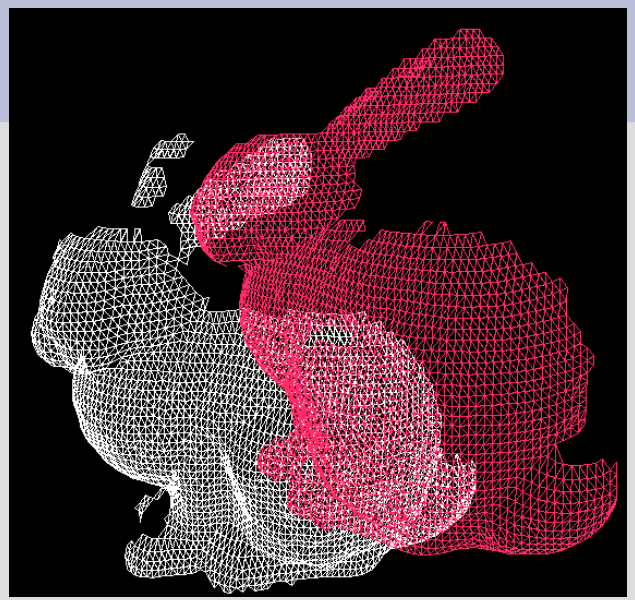
- Typische 3D-Dokumentation(3D-Modelling-Pipeline)
  - Scanning (Stereographic-System oder Laser-Range Scanner)
  - Data Registration (Anpassung mehrerer Abtastungsresultate → einem Koordinatensystem)
  - **Data Integration (Ball Pivoting)**
  - Model Conversion (Mesh decimation/optimization)

3D-modeling pipeline Scanning-Phase

- Klassische Messkammern
- Analoge und digitale Rotationskamera

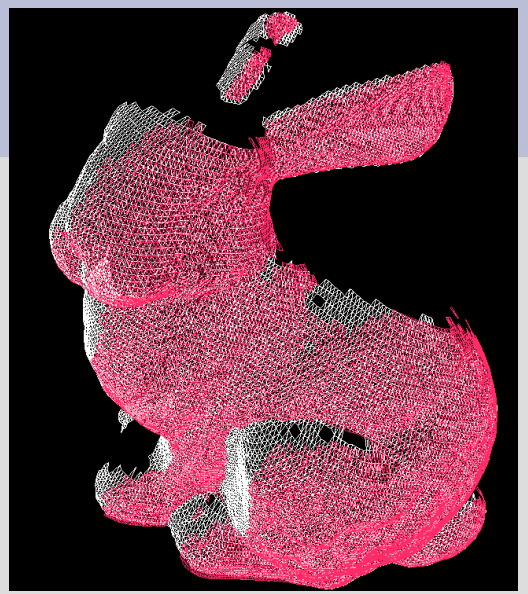
Figure 1. *Members of our team record exact 3D coordinates using a total station surveying instrument (upper left), capture video of an excavation using a scanner and digital camera (lower left), and scan a portion of the excavation site using a time-of-flight laser range scanner (right).*

3D-modeling pipeline



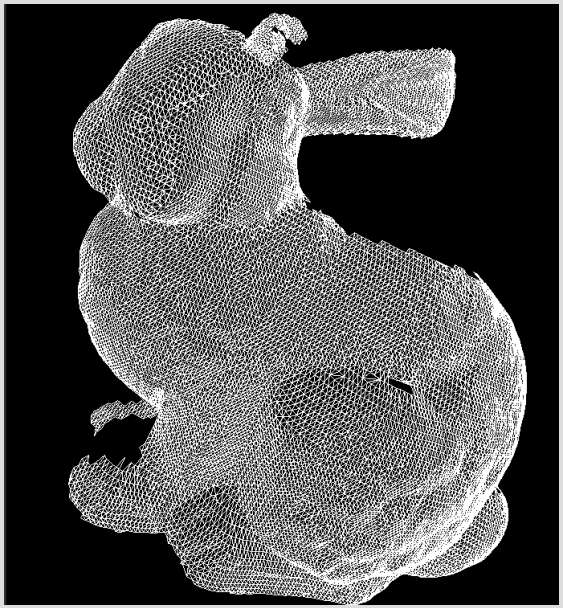
Range Image Acquisition

3D-to-3D Registration



Aligned meshes (Anpassung)

Meshing



Single Mesh

Remove redundancy

Quelle: Ioannis Stamos

Anwendungsbeispiele

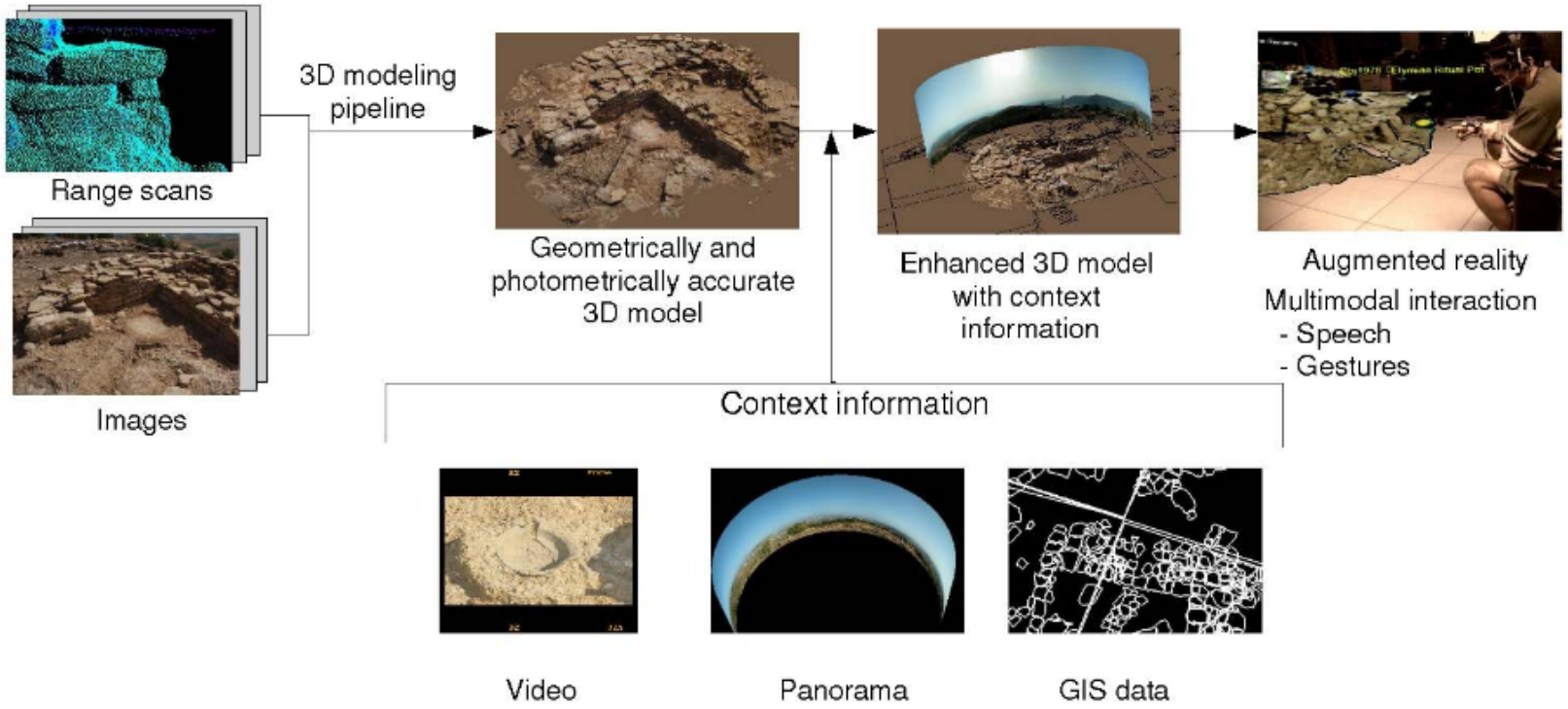


Figure 2. *Our 3D modeling and visualization pipeline. We start by building a textured 3D model using range scans and images, which we enhance with contextual information in the form of panoramic images, video, and GIS data. This context-rich model is then used as input to our multimodal augmented reality application.*

Quelle: IEEE Xplore

Angestrebtes Ziel

- Nicht-Ebene-Objekte und Freiformoberflächen 3-Dimensional dokumentieren
- Die Stärken der bisherigen Methoden (AlphaShapes) zu nutzen
- mit einer Linearen Zeitkomplexität (Zeit- und Speicherbedarf)
- Robusten Algorithmus

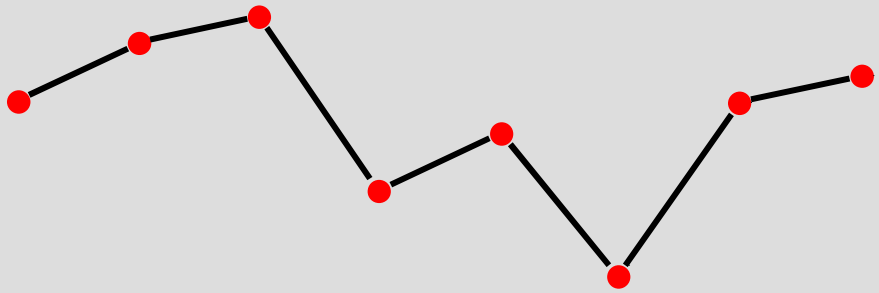
Grundlagen/Begriffsdefinitionen

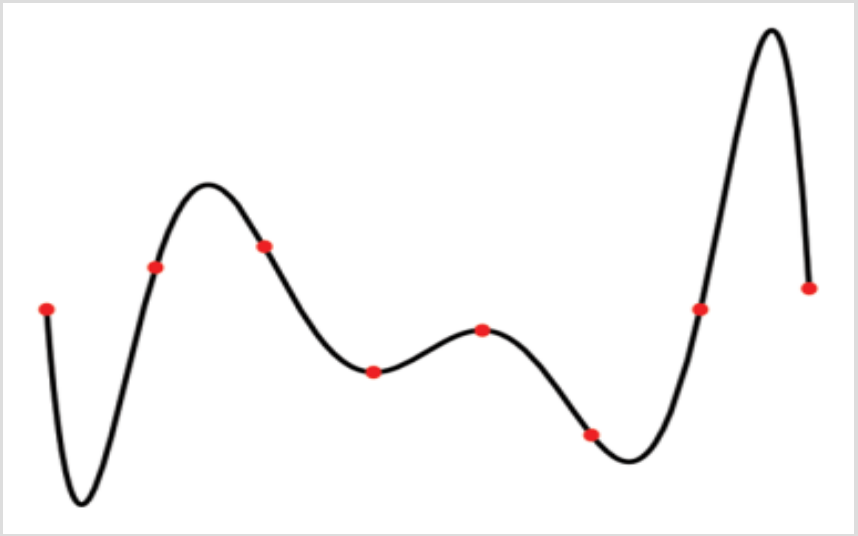
- **Punktwolke**: Unter einer Punktwolke wird in diesem Zusammenhang eine Menge ungeordneter Koordinaten (3-stellige Wertetupel) beschrieben, die Positionen oder Punkte im euklidischen R^3 repräsentieren. Diese Punkte bilden eine Teilmenge der Oberfläche eines dreidimensionalen Objektes.

Grundlagen/Begriffsdefinitionen

- **Range Image** = Eine Menge von 3D-Punkten
- **Interpolation** (Wiki)= Zu gegebenen diskreten Daten (z. B. Messwerten) soll eine kontinuierliche Funktion (die sogenannte Interpolante oder Interpolierende) gefunden werden, die diese Daten abbildet. Man sagt dann, die Funktion interpoliert die Daten.

Grundlagen/Begriffsdefinitionen

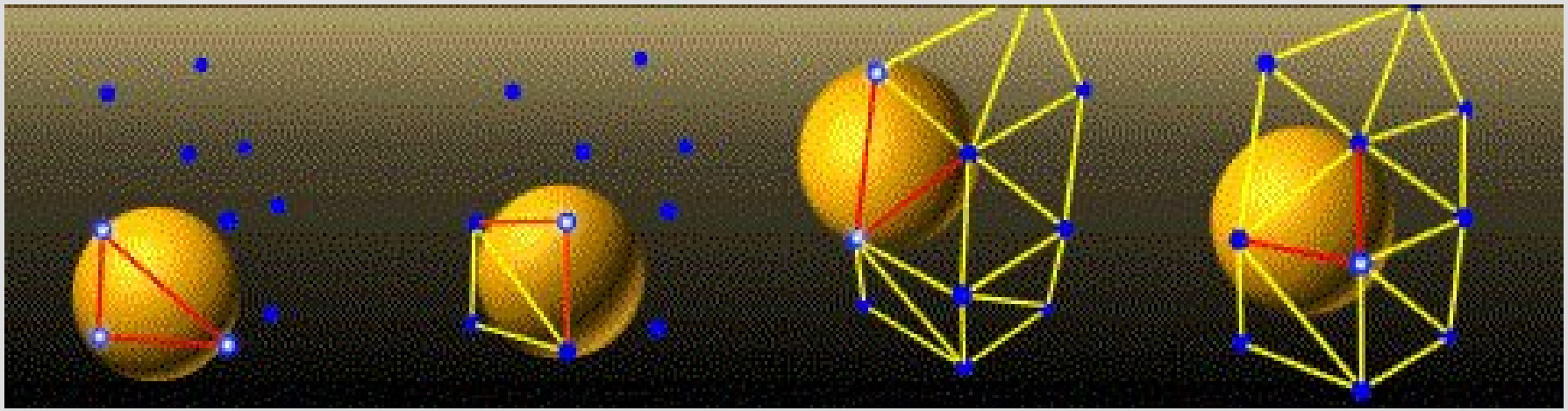




- Stückweise durchgeführte lineare Interpolation und
- Interpolationspolynom 7. Grades

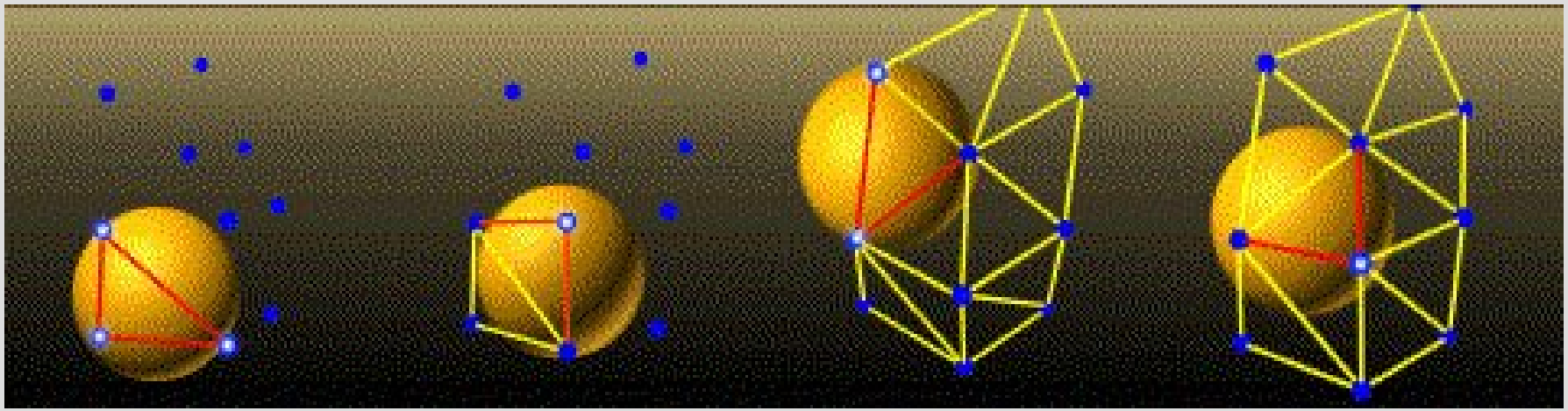
Grundlagen/Begriffsdefinitionen

- **BPA** (allg) ist ein Algorithmus (erstmal vorgestellt in Bernardini et al. '99), BPA erzeugt mit Hilfe einer Kugel ein Dreiecksnetz aus einer Punktwolke, bei der die Oberflächennormalen zu jedem Punkt bekannt sind.



Grundlagen/Begriffsdefinitionen

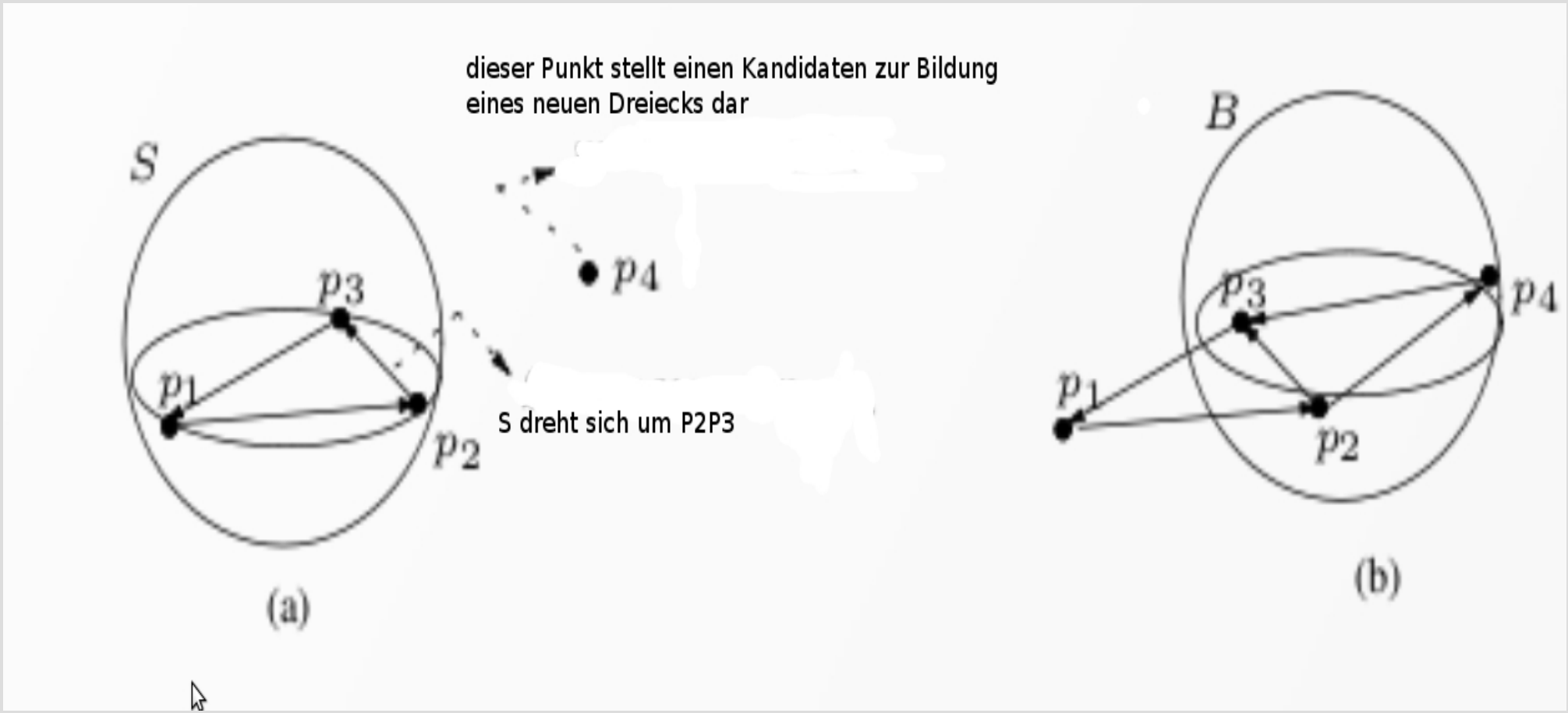
- **Interpolationstechniken**
  - **Sculpting-based (3D-Delaunay-Triangulation)**
  - **Region-growning (BPA)**



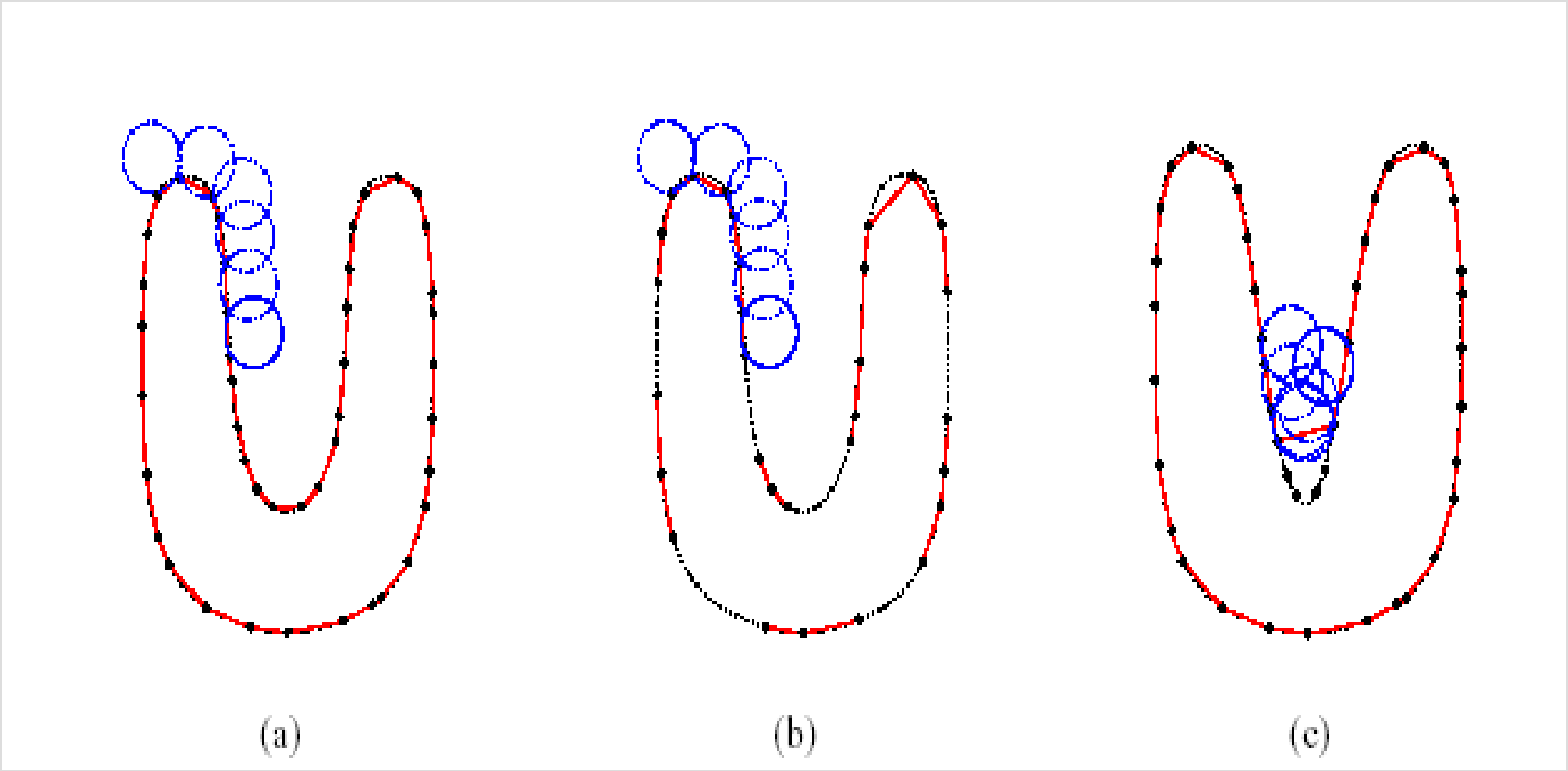
Der Algorithmus (BPA)

- Zu jeder Randkante eines Startdreieck wird eine gedachte Kugel gebildet, die einem festen Radius r besitzt
- diese Kugel rotiert nun um die Kante und streift dabei einen Punkt der Punktwolke.
- Ob sich dieser Punkt nun zur Erzeugung eines Dreieck eignet, wird bei einem Vergleich der Normale mit der Normale der Kantenendpunkte ermittelt.

Der Algorithmus (BPA)

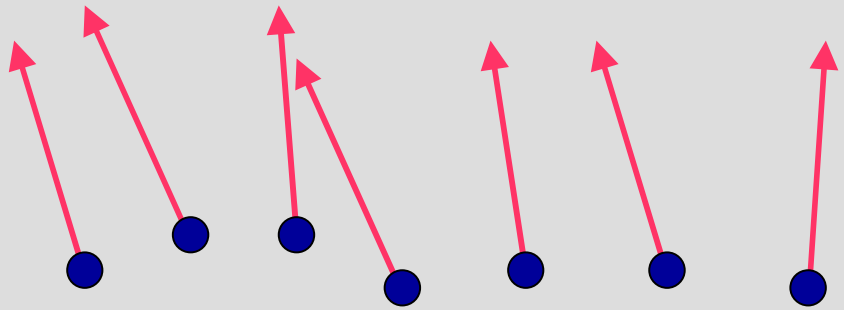


BPA (in 2D)



BPA

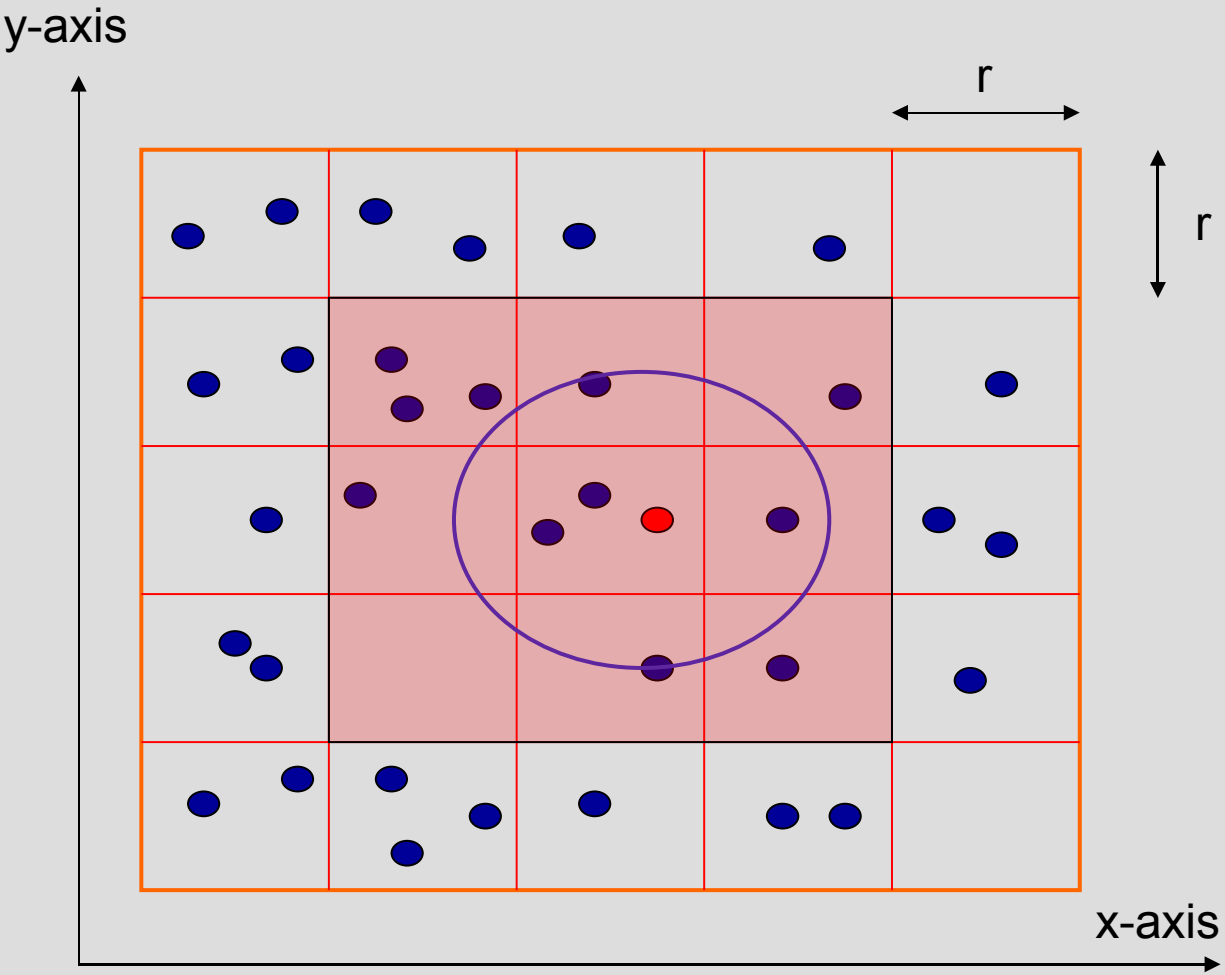
- Input: 3D-Punkten (Punktwolke) und Oberflächennormalen von Range-Images
- Output: Dreiecksnetz (Triangular mesh)



BPA-Preprocessing

- Um die möglichen Schnittpunkte mit dieser Kugel schnell finden zu können, wird die Punktwolke in einem Vorverarbeitungsschritt in ein dreidimensionales Zellraster der Kantenlänge 2r einsortiert.
  Vorteil des Zellrasters: konstanter Zugriff auf die Punkten

Räumlicher Datensatz



Schnelle Suche der Punkten mit dem Kreis-Raidus
r

[9 in diesem Bild –
27 in 3D

Implentierung des BPAs

- Front F repräsentiert die Sammlung von verketteten Listen, die Kanten beinhalten (dynamische Datenstrukturen)
- Implementierung der Punkte mit Hilfe einer verketteten Liste (diese werden mit bucket-sort organisiert)

  Vorteile :???

Vorteile d. Dynamischen Datenstruktur (BPA)

- Datenstruktur, die während ihrer Laufzeit geändert werden kann (vgl. Arrays)
- Einfache Überprüfung, ob der Punkt oder die Kante schon benutzt worden ist

Auswahl des Dreiecks (seed-selection)

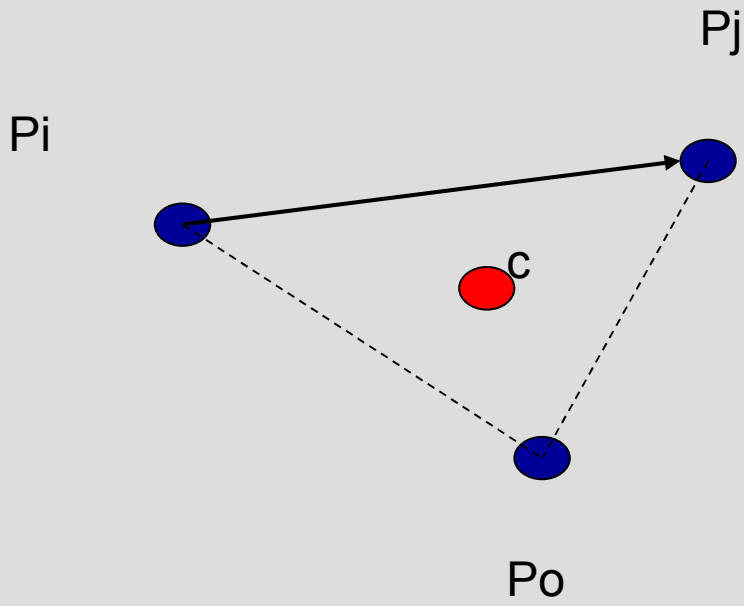
- Wähle einen Punkt P1, der noch nicht benutzt wurde
- Prüfe ob die Punkten P2 und P3 in der Nachbarschaft von P1 liegen
- Bilde ein Dreieck mit P1,P2,P3 (Triangulierung)
- Überprüfen der Oberflächennormalen der Punkten
- Teste ob eine Kugel mit einem Radius r alle 3 Punkte berührt (!! nur diese 3)
- Stoppe, wenn das ”gültige Dreieck” gefunden wurde

”Ball-Pivoting”

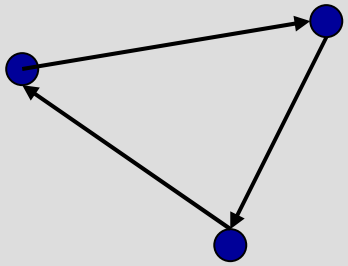
- Starte mit einem Dreieck (Pi, Pj, Po) und einer Kugel mit dem Radius r
- e(i,j) = die ”Pivoting” Kante (aktive)
- Das Pivoting ist praktisch gesehen eine kontinuierliche Bewegung der Kugel, dabei muss d. Kugel stets Pi und Pj berühren
- Wärend d. Bewegung kann die Kugel einen neuen Punkt Pk berühren => neues Dreieck (Pi,j,k)
- Falls die Kugel keinen neuen Punkt berührt, dann wird die Kante als boundary markiert

Implentierung des BPAs

- Edge (Pi, Pj)
  - Edge: “Active”, “Boundary”, or “Frozen”



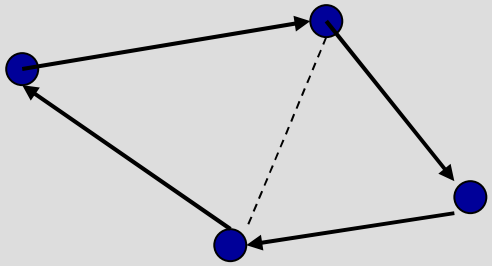
Beispiel



Active edge

Punkt aus Front-F

Beispiel

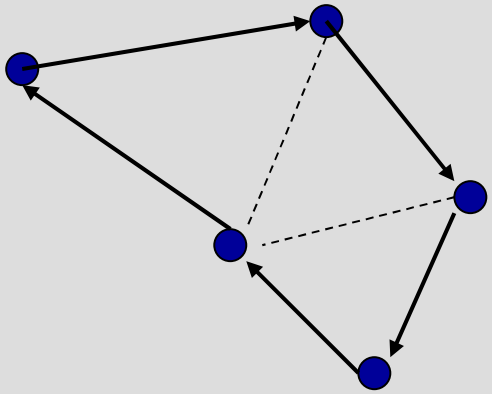


Active edge

Punkte aus fron-F

Kugel dreht sich (pivoting) um die active Kante

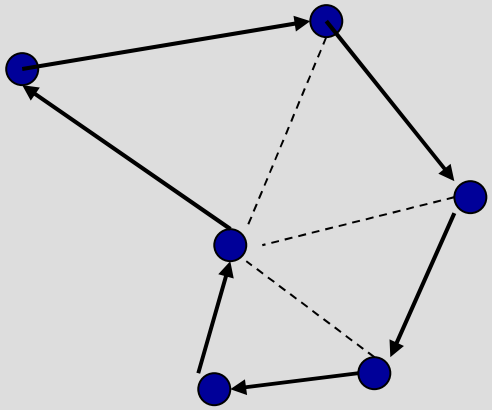
Beispiel



Active edge

Punkte aus front-F

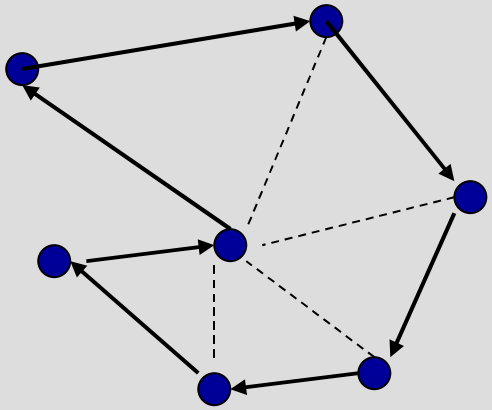
Beispiel



Active edge

Punkte aus front-F

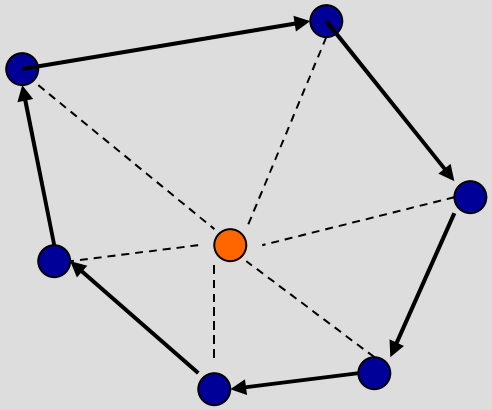
Beispiel



Active edge

Punkte aus front-F

Beispiel



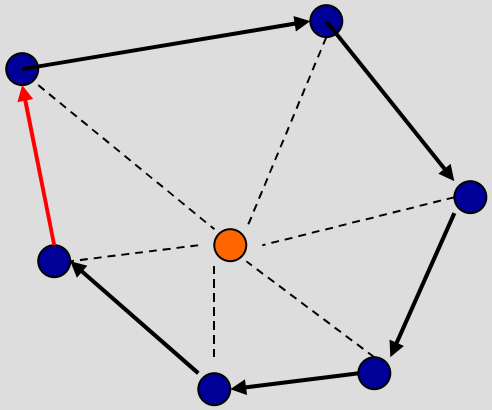
Active edge

Punkte aus F

Interner Punkt

Beispiel

Boundary edge



Active edge

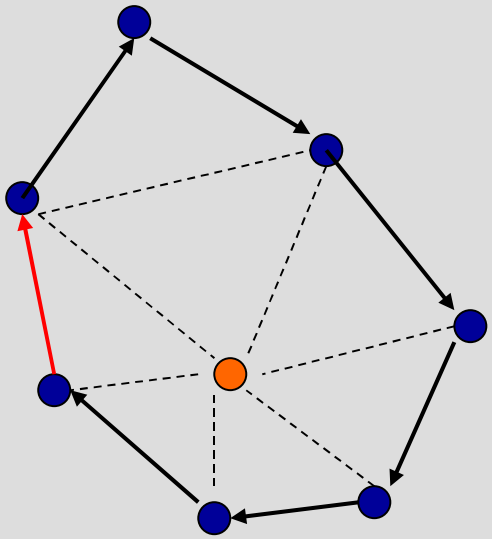
Punkte aus F

Interner Punkt (intenal point)

Ball pivoting around active edge
No pivot found

Beispiel

Boundary edge



Active edge

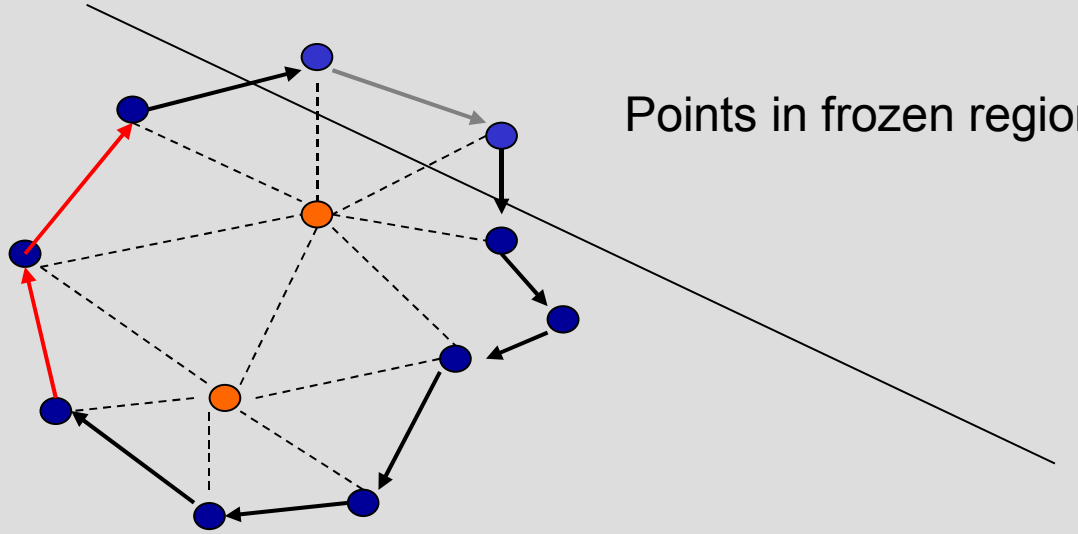
Point on front

Internal point

Implentierung des BPAs

Boundary edge

Frozen edge



Points in frozen region

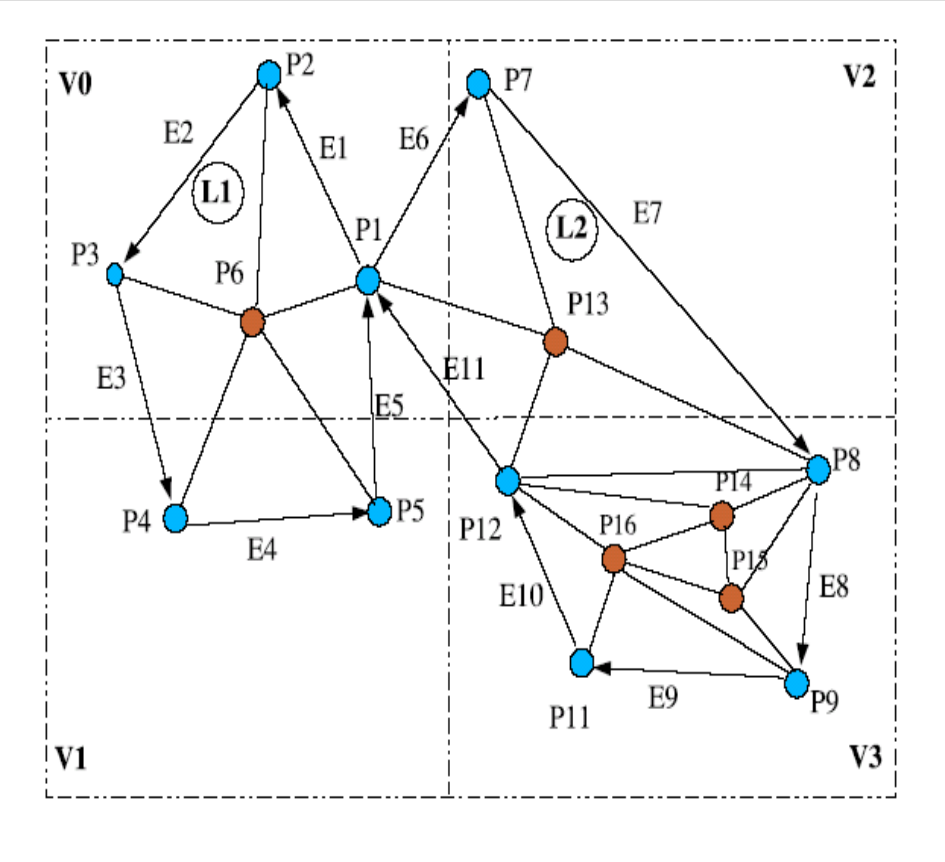
Active Kante

Ball pivoting around active edge

Punkte aus front-F

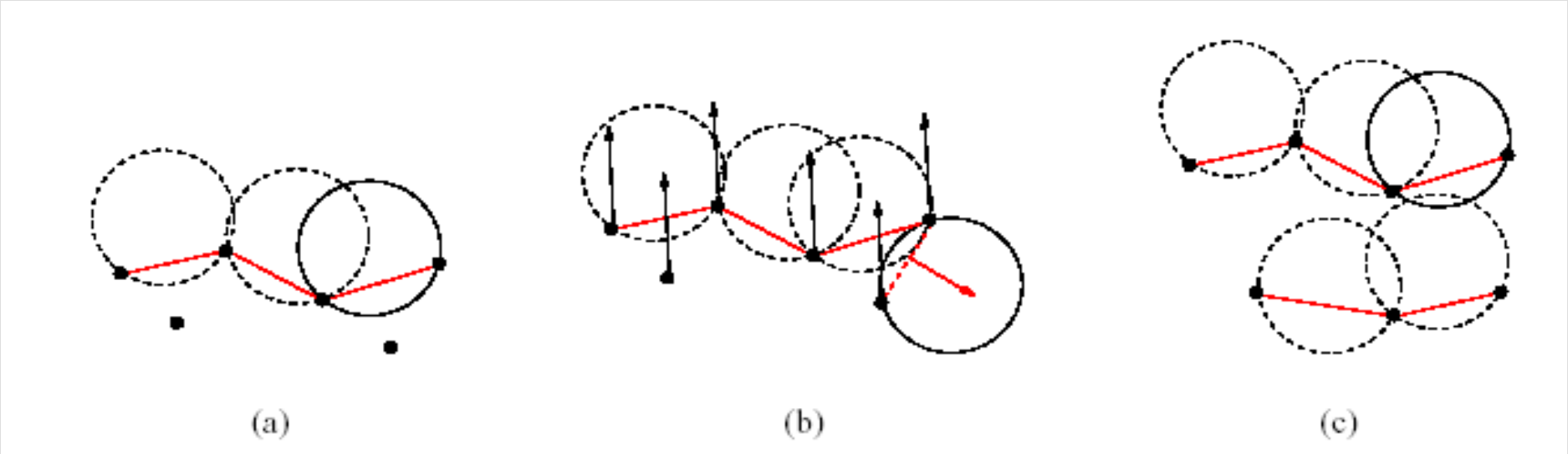
Internal point

Beispiel



- 13 Punkten (P1, …)
- 4 voxels (V0, …)
- 11 Kanten (E1, …)
- Two loops: L1, L2

Noisy data



Skelett eines BPAs

```
1.  while (true)

2.      while (e_(i,j) = get_active_edge(F))
3.          if (σ_k = ball_pivot(e_(i,j)) &&
                    (not_used(σ_k) || on_front(σ_k)))
4.              output_triangle(σ_i, σ_k, σ_j)
5.              join(e_(i,j), σ_k, F)
6 .             if (e_(k,i) ∈ F) glue(e_(i,k), e_(k,i), F)
7 .             if (e_(j,k) ∈ F) glue(e_(k,j), e_(j,k), F)
8 .         else
9 .             mark_as_boundary(e_(i,j))

10.     if ((σ_i, σ_j, σ_k) = find_seed_triangle())
11.         output_triangle(σ_i, σ_j, σ_k)
12.         insert_edge(e_(i,j), F)
13.         insert_edge(e_(j,k), F)
14.         insert_edge(e_(k,i), F)
15.     else
16.         return
```

Join & Clue Operationen

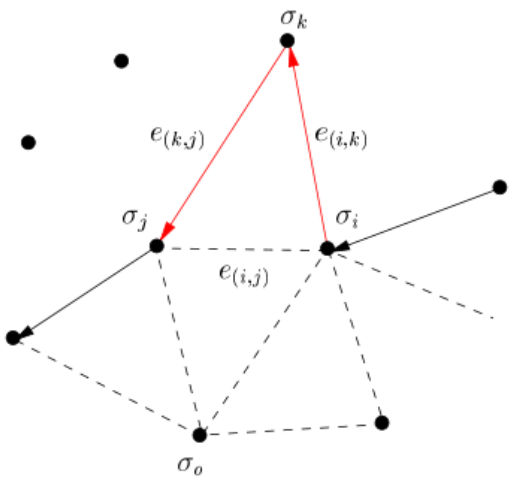


Fig. 6. A *join* operation simply adds a new triangle, removing edge $e_{(i,j)}$ from the front and adding the two new edges $e_{(i,k)}$ and $e_{(k,j)}$.

- Zur Generierung des Dreiecks und Modifikation der front-F

Join & Clue Operationen

- Sei Pk ein von der Kugel erfasste Punkt(not_used/not in mesh):
  - Triangulierung der Punkten Pi, Pj, Pk
  - e(Pi,Pj) wird aus front-F entfernt und e(Pi,Pk) + e(Pk,Pj) hinzugefügt

Join & Clue Operationen

- Sei Pk ein von der Kugel erfasste Punkt, der ein Teil von ”mesh” ist:
- 1.Fall Pk ist ein interner Punkt (internal point)
  - keine Triangulierung der Punkten Pi, Pj, Pk
  - e(Pi,Pj) wird als boundary markiert
- 2.Fall Pk ist ein Punkt aus front-F
  - Überprüfung der Oberflächennormalen
  - Ausführen der join-Operation und gibt ein neues Dreieck(Pi,Pk,Pj) aus
  - Problem: Theoretisch könnte 1-2 Kanten mit entgegengesetzter Richtungen hinzugefügt werden
  - Lösung: clue-Operation → entfernt diese Kanten wieder aus front-F

Join & Clue Operationen

- einfaches Beispiel: wenn e(Pi,Pk) durch join-Operation in F hinzugefügt wurde
  - Und e(Pk,Pi) E front-F dann werden e(Pi,Pk) und e(Pk,Pi) durch clue wieder entfernt

Out-of-Core

- Datenregionen in “Slices” aufzuteilen
- Effizient-Steigerung(Speicher)
- Kein limit in size of input erforderlich
- Für größere Scans zwingend erforderlich (aufgrund der beschränkten Hardware)

run-out-of-core()

```
for (i=0; i<k; ++i){
   current_slice=i;
   if (i>0) unload_slice(i-1);
   load_slice(i);
   if (i>0) move_frozen_to_active();
   run_bpa();
   save_current_mesh();
}
```

end run-out-of-core()

Fazit

- Vorteile:
  - Effizienter und Robuster Algorithmus
  - Models scenes of any geometric type
  - Models scenes of any size (out of core)
  - Flexibel:
    - Größere Kugel: grob(kleinere Objekte)
    - Kleinere Kugel: schärfer, mehr details (größere Objekte)
- Nachteile:
  - Kugel-Radius r passt sich nicht an die lokale Dichte
  - Empfindlich gegen ungenauer/verfälschte Oberflächennormale/Daten (noisy data)
  - Lücken (holes) können dadurch entstehen kleinere Lücken deuten auf ungenaueren Oberflächennormalen oder unterschiedliche Dichte hin größere = fehlende/fehlerhafte Daten Ein hole-filling-algorithm ist zwingend erforderlich

Beispiel

Dataset	# Pts	# Scans	ρ	# Slices	# Triangles	Mem. Usage	I/O Time	CPU Time
Clean	11K	1	4	-	22K	4	1.2secs	1.8secs
Bunny	361K	10	0.3, 0.5, 2	-	710K	86	4.5secs	2.1
Dragon	2.0M	71	0.3, 0.5, 1	-	3.5M	228	22secs	10.1
Buddha	3.3M	58	0.2, 0.5, 1	-	5.2M	325	48secs	16.9
Out of core								
Dragon	2.1M	1452	0.3, 0.5, 1	23	3.5M	137	1.0	19.8
Buddha	3.5M	1122	0.2, 0.5, 1	24	5.2M	155	2.1	26.8
Pietà	7.2M	770	1.5, 3, 6	24	14M	180	2.5	28.5

Fig. 10. Summary of results. *# of Pts* and *# of Scans* are the original number of data points and range images respectively. ρ lists the radii of the pivoting balls, in *mm*. Multiple radii mean that multiple passes of the algorithm, with increasing ball size, were used. *# Slices* is the number of slices into which the data is partitioned for out-of-core processing. *# of Triangles* is the number of triangles created by BPA. *Mem. Usage* is the maximum amount of memory used at any time during mesh generation, in MB. *I/O Time* is the time spent reading the input binary files; it also includes the time to write the output mesh, as an indexed triangle set, in binary format. *CPU Time* is the time spent computing the triangulation. All times are in minutes, except where otherwise stated. All tests were performed on a 450MHz Pentium II Xeon.

Beispiele

Beispiele

Beispiele

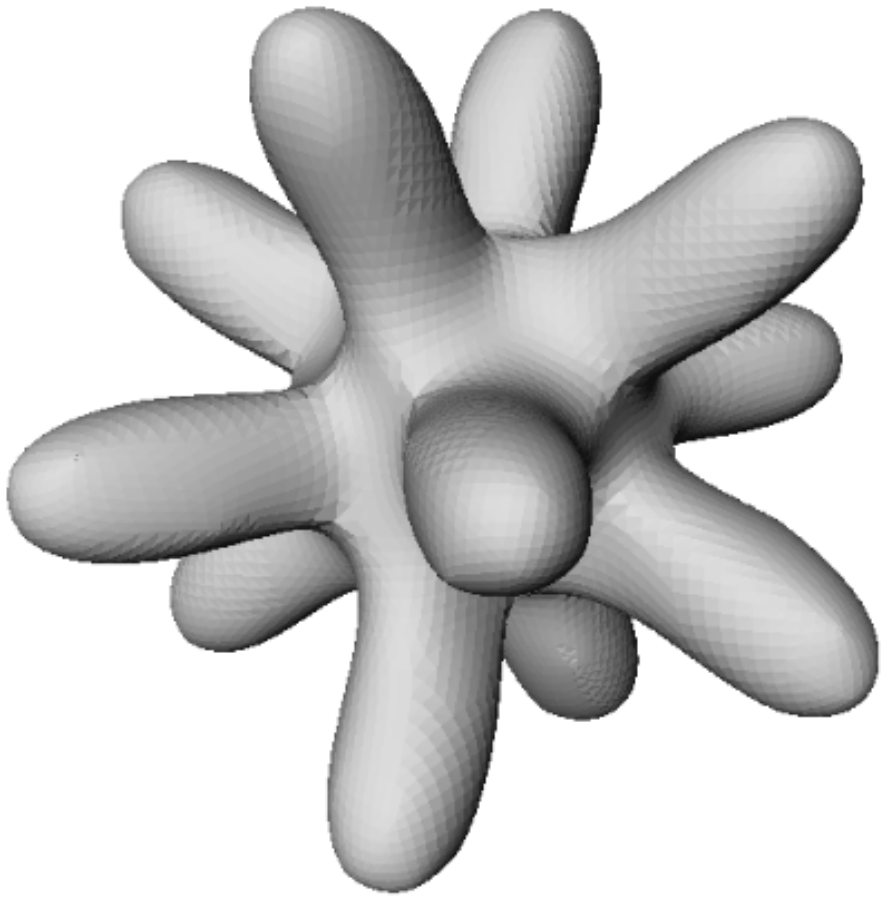


Fig. 9. Results. “Clean” data computed from an analytical surface.

Quellen/Literatur

- The Ball-Pivoting Algorithm for Surface Reconstruction(Bernardini)
- The Ball Pivoting Algorithm – Vortrag von Ioannis Stamos
- A Topological Framework for Advancing Front Triangulation
  - Von ESDRAS MEDEIROS, LUIZ VELHO , HELIO LOPES
- Von Punktwolken zu Dreiecksnetzen
  - Diplomarbeit von Stefan Preuß
  - Universität Karlsruhe (TH)