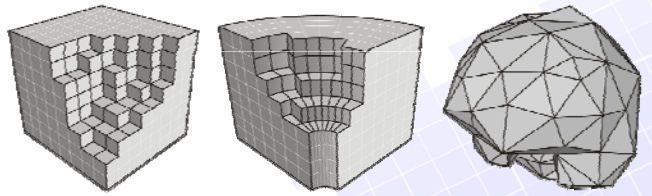
Filteroperationen Triangulierung

Andreas Kolb und Martin Lambers

computergraphik und multimedia systeme
universität siegen

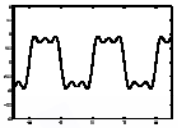
Review: Letzte Stunde

- **Gitterstrukturen in 2D und 3D**
  - uniforme Gitter
  - rectilineare Gitter
  - curvilineare Gitter
  - unstrukturierte Gitter



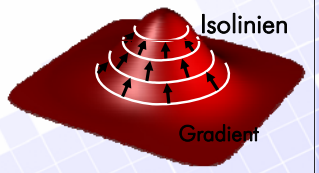
Review: Letzte Stunde

- **Abtasttheorem**
  - Kontinuierliche Signale lassen sich in harmonische Schwingungen zerlegen (Frequenzspektrum)
  - Bandbegrenzte Signale lassen sich aus diskreten Abtastwerten exakt rekonstruieren
  - Exakte Rekonstruktion erfolgt mit dem Sinc-Filter
  - Unterabtastung führt zu *Aliasing*-Effekten



Review: Letzte Stunde

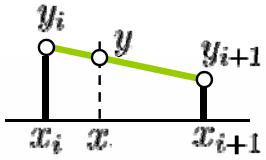
- **Differenzieren auf Gittern**
  - Gradientenvektor
    - zeigt in Richtung des größten Anstiegs
    - steht senkrecht auf den Isolinien (2D) bzw Isoflächen (3D)
  - Differenzieren auf Gittern
    - Curve/Surface Fitting
    - Finite Differenzen
      - Vorwärts, Rückwärts,
      - Zentrale Differenzen



Review: Lineare Interpolation

5

LINEAR:



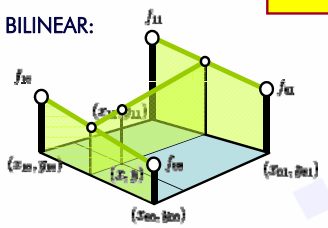
Lineares *Blenden*,

$$f(x) = \alpha y_{i+1} + (1-\alpha) y_i$$

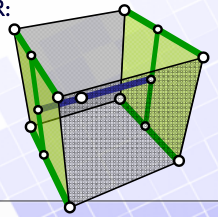
mit

$$\alpha = \frac{x - x_i}{x_{i+1} - x_i}$$

BILINEAR:



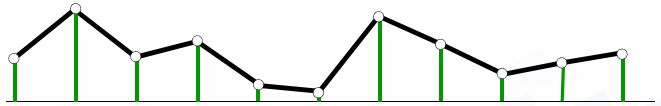
TRILINEAR:



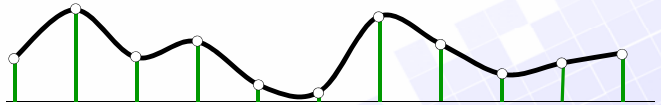
Glatte Interpolation

6

- (Stückweise) Lineare Interpolation ist C_0-*stetig*



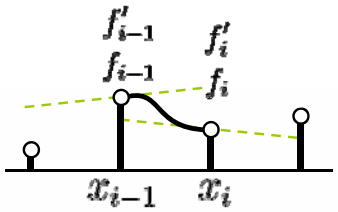
- Wie bekomme ich eine „glatte" Interpolation (C^1-*stetig*)?



- Lösung: z.B. stückweise kubische Interpolation

Catmull-Rom-Interpolation

7



1. Bestimme die Ableitungen f_i durch zentrale Differenzen

2. Bestimme eine (lokale) kubische Funktion, die sowohl die Funktionswerte f_i als auch die Ableitungen f'_i interpoliert

Hermite-Interpolation

Hermite Interpolation

8

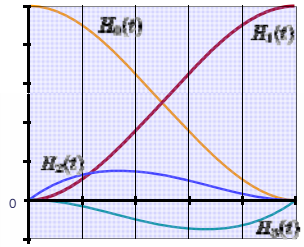
Gegeben: f_0, f_1, f'_0 und f'_1.

Gesucht: Polynom 3.Grades $f(t)$, so dass gilt

$$f(0) = f_0 \qquad f(1) = f_1 \qquad \frac{\partial}{\partial t} f(0) = f'_0 \qquad \frac{\partial}{\partial t} f(1) = f'_1$$

		$f(0)$	$f(1)$	$f'(0)$	$f'(1)$
$H_0(t)$	$(1-t)^2(2t+1)$	1	0	0	0
$H_1(t)$	$t^2(3-2t)$	0	1	0	0
$H_2(t)$	$-t^2(1-t)$	0	0	1	0
$H_3(t)$	$t(1-t)^2$	0	0	0	1

Hermite Interpolation



$f(0)$	$f(1)$	$f'(0)$	$f'(1)$

Das gesuchte Polynom zur lokalen Interpolation bei gegebenem f_0, f_1, f'_0 und f'_1 ist:

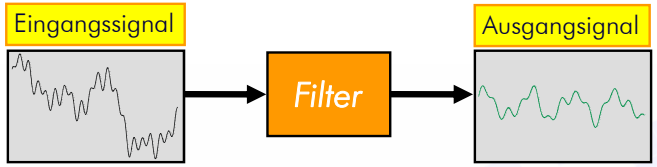
$$f(t) = f_0\,H_0(t) + f_1\,H_1(t) + f'_0\,H_2(t) + f'_1\,H_3(t)$$

Zusammenfassung Interpolation

- **Lokale Interpolationsverfahren**
  - linear, bilinear, trilinear in kartesischen Koordinaten
  - linear in baryzentrischen Koordinaten (Schwerpunktskoordinaten)
    - Dreieck, Tetraeder, Verallgemeinerbar auf beliebige n-Simplices
  - *Catmull-Rom Interpolation*
    - C^1-stetig
- **Was ich verschwiegen habe:**
  - Allgemeine Kurven (z.B. B-Splines)
    → Computergraphik II

Lineare Filter

Aus der Signalverarbeitung:



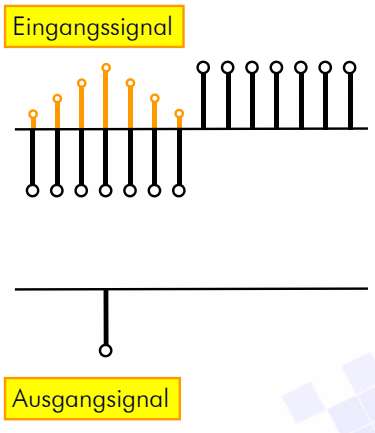
Ein linearer Filter schwächt bestimmte Frequenzen ab und läßt andere passieren.

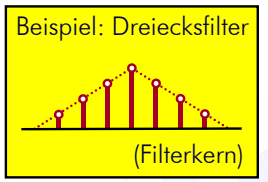
Tiefpass-Filter: läßt nur niedrige Frequenzen passieren

Hochpass-Filter: läßt nur hohe Frequenzen passieren

Bandpass-Filter: läßt nur einen bestimmten Frequenz-*bereich* durch.

Lineare Filter

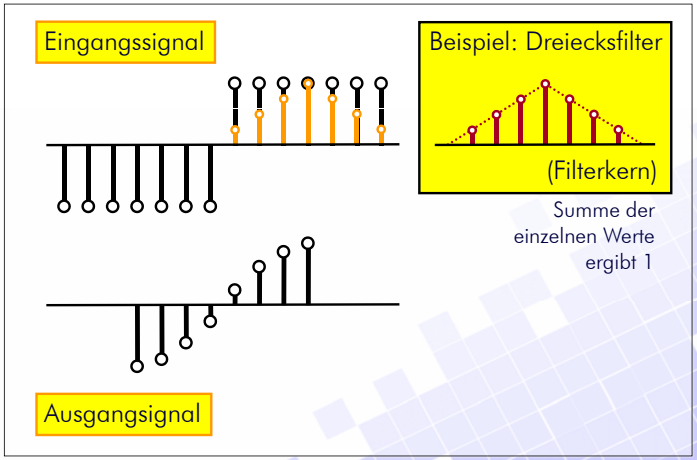




Summe der einzelnen Werte ergibt 1

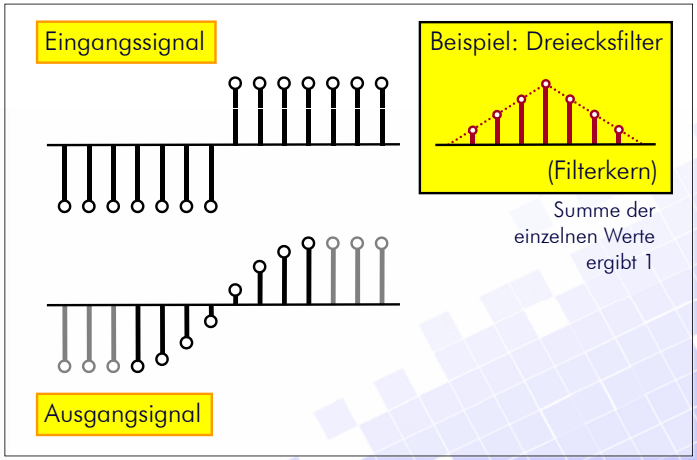
Lineare Filter

13



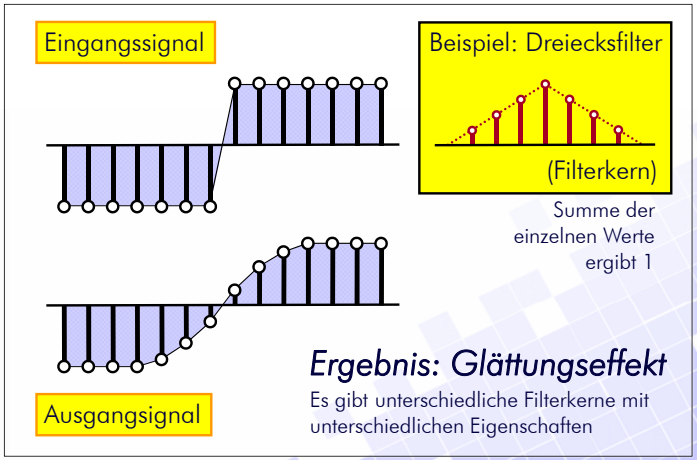
Lineare Filter

14



Lineare Filter

15

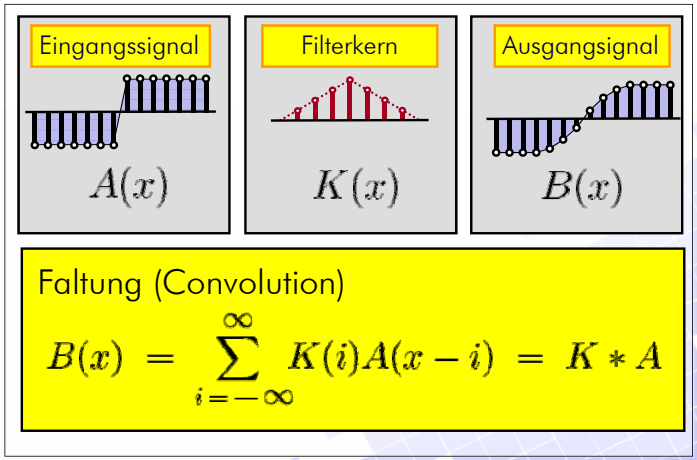


Ergebnis: Glättungseffekt

Es gibt unterschiedliche Filterkerne mit unterschiedlichen Eigenschaften

Lineare Filter

16



Faltung (Convolution)

$$B(x) = \sum_{i=-\infty}^{\infty} K(i)A(x-i) = K * A$$

Lineare Filter

17

Faltung (Convolution)

$$B(x) = \sum_{i=-\infty}^{\infty} K(i)A(x-i) = K * A$$

Rechenregeln für die Faltung:

Kommutativgesetz:

$$(A * B) = (B * A)$$

Assoziativgesetz:

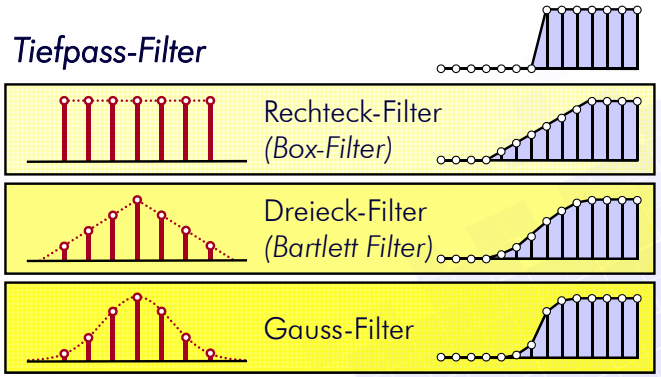
$$(A * B) * C = A * (B * C)$$

Glättungsfilter

18

Zweck: Glätten von scharfen Kanten, Rauschunterdrückung

Tiefpass-Filter



Kantenfilter

19

Zweck: Erkennen von Kanten *Hochpass-Filter*

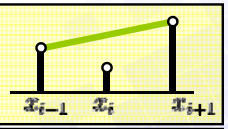
Kantendetektion:

$|f'(x)| = \max$

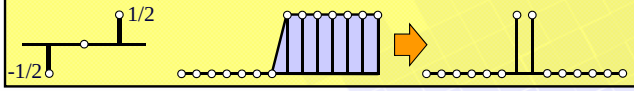
$\Rightarrow$ Kante in $f(x)$

Zentrale Differenzen:

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$$

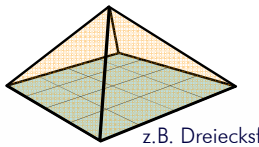


Linearer Filter, der zentrale Differenzen berechnet:



Lineare Filter in 2D

20



z.B. Dreiecksfilter

Eingangssignal	$A(x,y)$
Filterkern	$K(x,y)$
Ausgangssignal	$B(x,y)$

2D Faltung:

$$B(x,y) = K * A = \sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} K(i,j)A(x-i,y-j)$$

Faltung in 3D und n-dim. Raum analog

Separierbare Filter

21

- n-dim. lineare Filter, die sich durch Hintereinanderschalten mehrerer 1D-Filter berechnen lassen (→ schnellere Auswertung: k*n statt k^n),
- d.h. lineare Filter, die sich als Tensorprodukt darstellen lassen:

$$K(x,y) = K_1(x) * K_2(y)$$

Beispiel: 2D Gauss

$$\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} = \begin{bmatrix}1\\2\\1\end{bmatrix} \begin{bmatrix}1&2&1\end{bmatrix}$$

Beispiel: Sobel

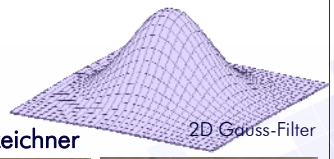
$$\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix} = \begin{bmatrix}1\\2\\1\end{bmatrix} \begin{bmatrix}-1&0&1\end{bmatrix}$$

2D Glättungsfilter

22

Tensorprodukt der 1D Varianten:

- Boxfilter
- Dreiecksfilter (Pyramide)
- Gauss-Filter



2D Gauss-Filter

Beispiel Bildbearbeitung: Weichzeichner

Original | Gauss Filter, 5x5 | Gauss-Filter 9x9

Zusammenfassung

23

Lineare Filter:

- Filterkern wird mit Signal ***gefaltet***
- Linearer Filter entfernen bestimmte Frequenzen aus dem Signal
- Glättungsfilter:
  - Box-, Dreieck-, Gauss-Filter
  - Tiefpassfilter: hohe Frequenzen werden entfernt
- Kantenfilter:
  - Hochpass-Filter: niedrige Frequenzen werden abgeschwächt

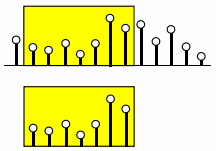
Nicht-Lineare Filter

24

Lineare Filter berechnen den Wert des Ausgangssignals als *Linearkombination* der Werte des Eingangssignals

- *Nicht-lineare Filter* verwenden hier beliebige nicht-lineare Funktionen $f(x_{i-k}, \ldots, x_i, \ldots, x_{i+k})$

Beispiel: *Der Median-Filter* ***(Rangordnungsoperation)***



1. Wähle alle Samples innerhalb des Filterkerns

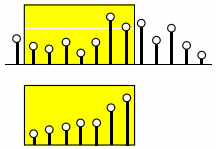
Nicht-Lineare Filter

25

Lineare Filter berechnen den Wert des Ausgangssignals als *Linearkombination* der Werte des Eingangssignals

- *Nicht-lineare Filter* verwenden hier beliebige nicht-lineare Funktionen $f(x_{i-k}, \ldots, x_i, \ldots, x_{i+k})$

Beispiel: *Der Median-Filter (Rangordnungsoperation)*



1. Wähle alle Samples innerhalb des Filterkerns
2. *Sortiere* alle Samples nach ihrem Wert

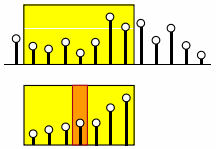
Nicht-Lineare Filter

26

Lineare Filter berechnen den Wert des Ausgangssignals als *Linearkombination* der Werte des Eingangssignals

- *Nicht-lineare Filter* verwenden hier beliebige nicht-lineare Funktionen $f(x_{i-k}, \ldots, x_i, \ldots, x_{i+k})$

Beispiel: *Der Median-Filter (Rangordnungsoperation)*



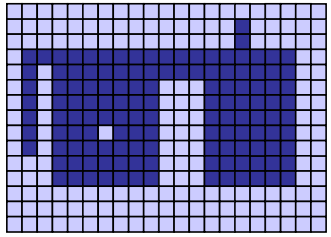
1. Wähle alle Samples innerhalb des Filterkerns
2. *Sortiere* alle Samples nach ihrem Wert
3. Wähle das ***mittlere*** Sample als Ausgangswert

Nichtlineare Filter

27

- *Morphologische Operatoren* für binäre Daten (0,1)-Werte

3x3-Filterkern K



Dilatation:
Pixelwert = 1, wenn in der Filternachbarschaft ein Pixelwert 1 vorkommt

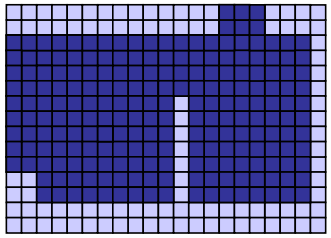
□ =0 (Hintergrund)
■ =1 (Objekt)

Nichtlineare Filter

28

- *Morphologische Operatoren* für binäre Daten (0,1)-Werte

3x3-Filterkern K



Dilatation:
Pixelwert = 1, wenn in der Filternachbarschaft ein Pixelwert 1 vorkommt

□ =0 (Hintergrund)
■ =1 (Objekt)

Dilatation erweitert den Rand des „Objekts“

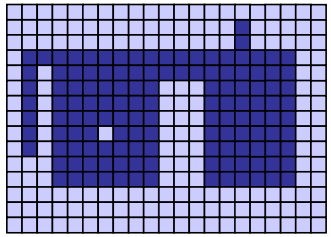
$$A \oplus K$$

Nichtlineare Filter

29

- *Morphologische Operatoren*
  für binäre Daten (0,1)-Werte

3x3-Filterkern K



Erosion:
Pixelwert = 0, wenn in der Filternachbarschaft ein Pixelwert 0 vorkommt

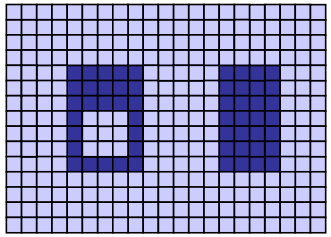
=0 (Hintergrund)
=1 (Objekt)

Nichtlineare Filter

30

- *Morphologische Operatoren*
  für binäre Daten (0,1)-Werte

3x3-Filterkern K



Erosion:
Pixelwert = 0, wenn in der Filternachbarschaft ein Pixelwert 0 vorkommt

Erosion verringert den Rand des „Objekts“

$$A \ominus K$$

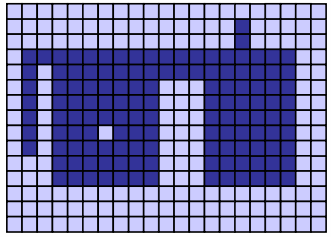
=0 (Hintergrund)
=1 (Objekt)

Nichtlineare Filter

31

- *Morphologische Operatoren*
  für binäre Daten (0,1)-Werte

3x3-Filterkern K



Opening (Öffnung):
Hintereinanderschaltung von *Erosion* und *Dilatation*

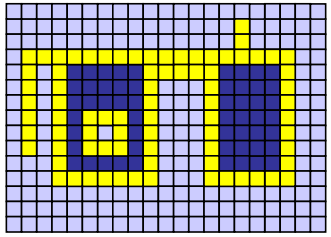
=0 (Hintergrund)
=1 (Objekt)

Nichtlineare Filter

32

- *Morphologische Operatoren*
  für binäre Daten (0,1)-Werte

3x3-Filterkern K



Opening (Öffnung):
Hintereinanderschaltung von *Erosion* und *Dilatation*

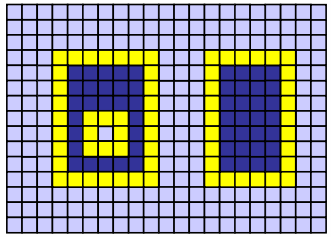
=0 (Hintergrund)
=1 (Objekt)

Nichtlineare Filter

33

- *Morphologische Operatoren*
  für binäre Daten (0,1)-Werte

3x3-Filterkern K



Opening (Öffnung):
Hintereinanderschaltung von
Erosion und *Dilatation*

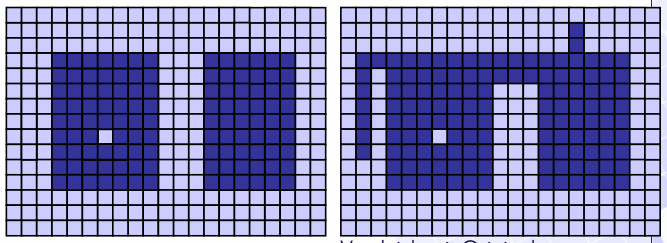
□ =0 (Hintergrund)
■ =1 (Objekt)

Nichtlineare Filter

34

- *Morphologische Operatoren*
  für binäre Daten (0,1)-Werte

3x3-Filterkern K



Vergleich mit Original

□ =0 (Hintergrund)
■ =1 (Objekt)

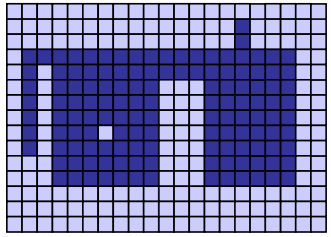
$$A \bigcirc K = (A \ominus K) \oplus K$$

Nichtlineare Filter

35

- *Morphologische Operatoren*
  für binäre Daten (0,1)-Werte

3x3-Filterkern K



Closing (Schließung):
Hintereinanderschaltung von
Dilatation und *Erosion*

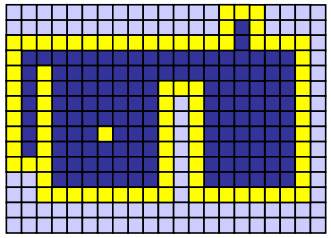
□ =0 (Hintergrund)
■ =1 (Objekt)

Nichtlineare Filter

36

- *Morphologische Operatoren*
  für binäre Daten (0,1)-Werte

3x3-Filterkern K



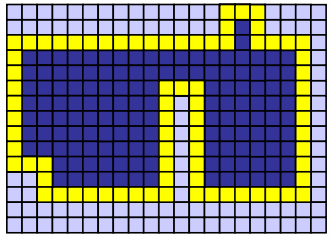
Closing (Schließung):
Hintereinanderschaltung von
Dilatation und *Erosion*

□ =0 (Hintergrund)
■ =1 (Objekt)

Nichtlineare Filter

- *Morphologische Operatoren*
  für binäre Daten (0,1)-Werte

3x3-Filterkern K



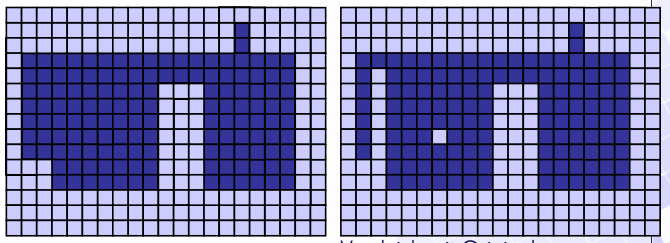
Closing (Schließung):
Hintereinanderschaltung von
Dilatation und *Erosion*

□ =0 (Hintergrund)
■ =1 (Objekt)

Nichtlineare Filter

- *Morphologische Operatoren*
  für binäre Daten (0,1)-Werte

3x3-Filterkern K



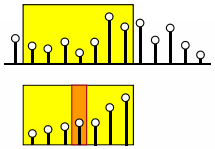
Vergleich mit Original

□ =0 (Hintergrund)
■ =1 (Objekt)

$$A \circ K = (A \oplus K) \ominus K$$

Nichtlineare Filter

- *Rangordnungsoperationen*

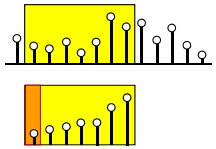


1. Wähle alle Samples innerhalb des Filterkerns
2. *Sortiere* alle Samples nach ihrem Wert

Median-Filter: Wähle mittleres Element.

Nichtlineare Filter

- *Rangordnungsoperationen*



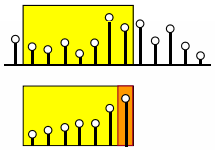
1. Wähle alle Samples innerhalb des Filterkerns
2. *Sortiere* alle Samples nach ihrem Wert

Median-Filter: Wähle mittleres Element.

Grauwert-Erosion: Wähle erstes Element.

Nichtlineare Filter

- *Rangordnungsoperationen*



1. Wähle alle Samples innerhalb des Filterkerns
2. ***Sortiere*** alle Samples nach ihrem Wert

Median-Filter: Wähle mittleres Element.

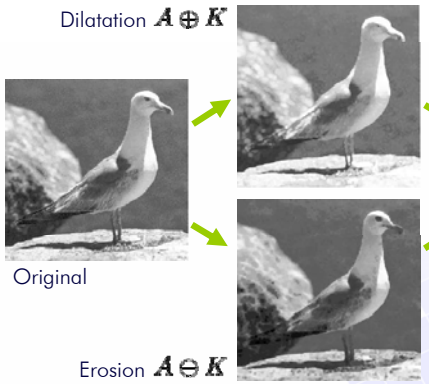
Grauwert-Erosion: Wähle erstes Element.

Grauwert-Dilatation: Wähle letztes Element.

Nichtlineare Filter

- *Morphologische Operationen* für Grauwerte

Dilatation $A \oplus K$



Kantenerkennung

Original

Erosion $A \ominus K$

$$\frac{A \oplus K}{A \ominus K}$$

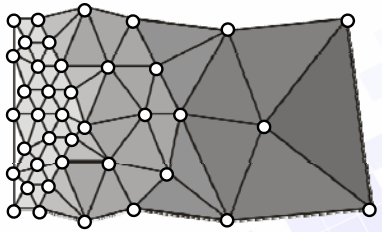
Zusammenfassung

Nicht-Lineare Filter:

- Keine ***Faltung,*** sondern beliebige nichtlineare Verknüpfung
- Rangordnungsoperationen:
  - Median-Filter:
    Rauschunterdrückung ohne Kantenglättung
  - Morphologische Operationen
    Dilatation, Erosion, Opening, Closing

Unstrukturierte Gitter

- Gegeben eine ***Punktewolke***, d.h. eine ungeordnete Menge an Vertices
- Wie sieht eine geeignete Gittersturktur dazu aus ?



Triangulierung

Triangulierung

Definition:

Eine ***Triangulierung*** einer Punktmenge

$$\mathbf{P} = \{P_i \mid i = 0, \ldots N-1\} \subseteq \mathbb{R}^2/\mathbb{R}^3$$

besteht aus

- ✗ den Punkten $\mathbf{P}$
- ✗ den Kanten

$$\mathbf{E} \subseteq \{\overline{P_iP_j} \mid 0 \le i,j < N, i \neq j\}$$

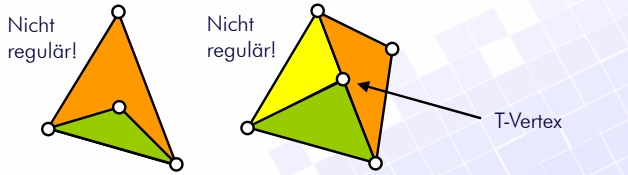
- ✗ und den Flächen (Dreiecken)

$$\mathbf{F} \subseteq \{\triangle(P_iP_jP_k) \mid 0 \le i,j,k < N, i \neq j\}$$

Triangulierung

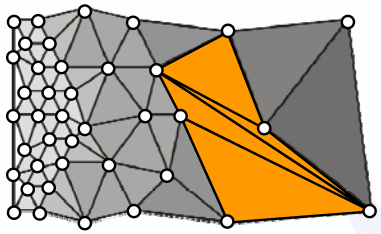
Eine Triangulierung ist ***regulär***, wenn gilt

- jeder Vertex ist Endpunkt mind. zweier Kanten
- jede Kante ist Teil von einem oder zwei Dreiecken
- Zwei Dreiecke schneiden sich { nicht ODER in einem Punkt ODER in einer Kante



Triangulierungen

- Sind alle regulären Triangulierungen auch „gute“ Triangulierungen?



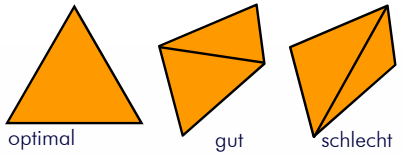
Triangulierung ist regulär

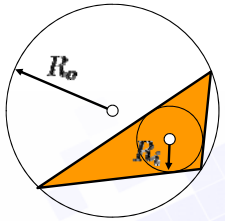
„schlechte“ Triangulierung: Dreiecke sind zu lang und spitz

Lange, spitze Dreiecke sollten vermieden werden, da sie bei Interpolation zu unschönen Ergebnissen führen!

Triangulierungen

- Was sind „gute“ Dreiecke?





Faustregeln:

- ✗ Optimal sind gleichseitige Dreiecke
- ✗ Der kleinste Winkel im Dreieck sollte maximal sein.
- Das Verhältnis zwischen Inkreis und Umkreis sollte maximal sein: $\frac{R_i}{R_o} = \max$

Systematische Lösung in 2D

49

Wie finde ich eine *optimale Triangulierung?*

- **Delaunay-Triangulierung**
  - *Definition* über Voronoi-Diagramm
  - *Eigenschaften* der Delaunay-Triangulierung
  - *Algorithmen* zur Bestimmung der Delaunay-Triangulierung
- **Voronoi-Diagramm**
  - Hilfreich bei der exakten Definition

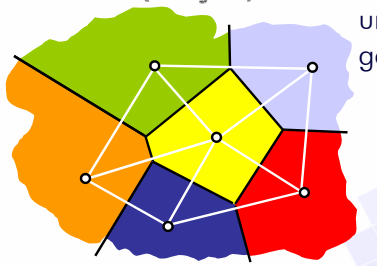
Voronoi-Tessellierung

50

Definition der *Delaunay-Triangulierung*

Kante $\overline{P_iP_j}$ $\Leftrightarrow$ Voronoi-Zellen von P_i und P_j haben eine gemeinsame Kante

Dreieck $\triangle(P_iP_jP_k)$ $\Leftrightarrow$ Voronoi-Zellen von P_i, P_j und P_k haben einen gemeinsamen Eckpunkt

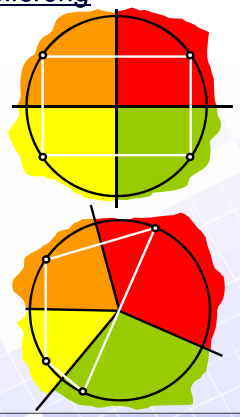


Voronoi-Tessellierung

51

Definition der *Delaunay-Triangulierung*

- *Ausnahmefall:*
  4 Punkte liegen auf einem Kreis *(und innerhalb des Kreises liegt kein weiterer Punkt)*
- In diesem Fall ergibt die Delaunay-Triangulierung kein Dreieck, sondern ein Viereck.
- Dieses Viereck kann durch Hinzunahme einer beliebigen Diagonale trianguliert werden



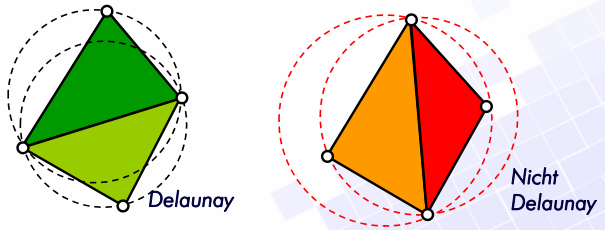
Delaunay-Triangulierung

52

- Eigenschaften der Delaunay-Triangulierung

Umkreis-Kriterium (CC):
Der Umkreis eines beliebigen Dreiecks enthält keinen weiteren Vertex

Notwendige und Hinreichende Bedingung!



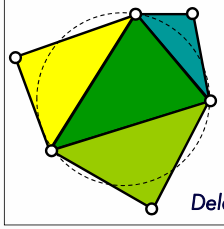
Delaunay-Triangulierung

53

- Eigenschaften der Delaunay-Triangulierung

Lokales Umkreis-Kriterium (LCC):
Für alle inneren Kanten: Der Umkreis eines Dreiecks enthält nicht den dritten Vertex des benachbarten Dreiecks

Notwendige und hinreichende Bedingung!



Leichter zu zeigen als globales Umkreis Kriterium, da nur die benachbarten Dreiecke untersucht werden müssen!

Delaunay

Algorithmen

54

Algorithmen zur Bestimmung der Delaunay Triangulierung einer gegebenen Punktemenge

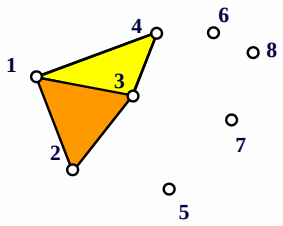
- *Flipping Algorithmus*
- *Inkrementeller Algorithmus*
- *Divide & Conquer-Algorithmus*

Delaunay-Triangulierung

55

Flipping Algorithmus

1. *Bestimme eine **beliebige (reguläre)** Triangulierung*



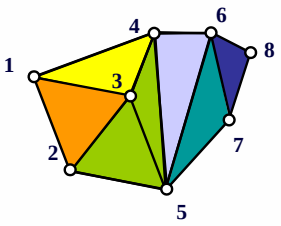
- Ordne alle Vertices lexikographisch (z.B. sortiere zuerst nach x- dann nach y-Koord.)
- Beginne mit den ersten 3 Vertices
- Füge schrittweise jeweils einen neuen Vertex hinzu. Verbinde den neuen Vertex mit allen sichtbaren Kanten

Delaunay-Triangulierung

56

Flipping Algorithmus

1. *Bestimme eine **beliebige (reguläre)** Triangulierung*



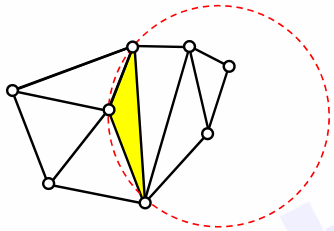
- Ordne alle Vertices lexikographisch (z.B. sortiere zuerst nach x- dann nach y-Koord.)
- Beginne mit den ersten 3 Vertices
- Füge schrittweise jeweils einen neuen Vertex hinzu. Verbinde den neuen Vertex mit allen sichtbaren Kanten

Delaunay-Triangulierung

Flipping Algorithmus

2. *„Flippe“ alle Kanten, die das lokale Umkreiskriterium (LCC) verletzen*

- Bestimme eine Liste aller Kanten, die LCC verletzen

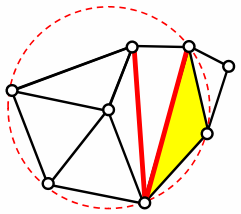


Delaunay-Triangulierung

Flipping Algorithmus

2. *„Flippe“ alle Kanten, die das lokale Umkreiskriterium (LCC) verletzen*

- Bestimme eine Liste aller Kanten, die LCC verletzen

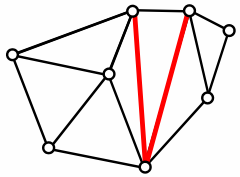


Delaunay-Triangulierung

Flipping Algorithmus

2. *„Flippe“ alle Kanten, die das lokale Umkreiskriterium (LCC) verletzen*

- Bestimme eine Liste aller Kanten, die LCC verletzen
- Entnehme die erste Kante der Liste und „flippe“ sie.

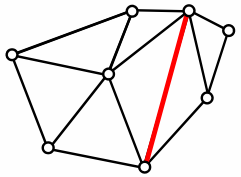


Delaunay-Triangulierung

Flipping Algorithmus

2. *„Flippe“ alle Kanten, die das lokale Umkreiskriterium (LCC) verletzen*

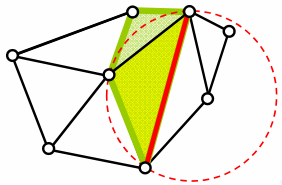
- Bestimme eine Liste aller Kanten, die LCC verletzen
- Entnehme die erste Kante der Liste und „flippe“ sie.



Delaunay-Triangulierung

Flipping Algorithmus

2. *„Flippe“ alle Kanten, die das lokale Umkreiskriterium (LCC) verletzen*

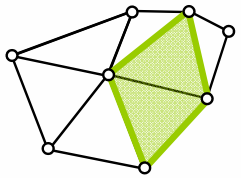


- Bestimme eine Liste aller Kanten, die LCC verletzen
- Entnehme die erste Kante der Liste und „flippe“ sie.
- Überprüfe alle 4 Kanten des betroffenen Vierecks und aktualisiere die Liste

Delaunay-Triangulierung

Flipping Algorithmus

2. *„Flippe“ alle Kanten, die das lokale Umkreiskriterium (LCC) verletzen*

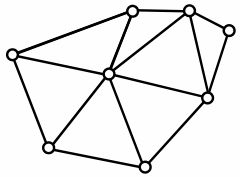


- Bestimme eine Liste aller Kanten, die LCC verletzen
- Entnehme die erste Kante der Liste und „flippe“ sie.
- Überprüfe alle 4 Kanten des betroffenen Vierecks und aktualisiere die Liste
- Weiter mit nächster Kante

Delaunay-Triangulierung

Flipping Algorithmus

2. *„Flippe“ alle Kanten, die das lokale Umkreiskriterium (LCC) verletzen*



- Bestimme eine Liste aller Kanten, die LCC verletzen
- Entnehme die erste Kante der Liste und „flippe“ sie.
- Überprüfe alle 4 Kanten des betroffenen Vierecks und aktualisiere die Liste
- Weiter mit nächster Kante

Algorithmen

Algorithmen zur Bestimmung der Delaunay Triangulierung einer gegebenen Punktemenge

- *Flipping Algorithmus*
- *Inkrementeller Algorithmus*
- *Divide & Conquer-Algorithmus*

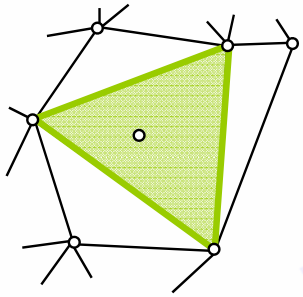
Vorteile: sehr einfach
Nachteil: benötigt initiale Triangulierung

Delaunay-Triangulierung

65

Inkrementeller Algorithmus

1. *Beginne mit einem Dreieck (3 beliebige Vertices)*
2. *Füge nacheinander weitere Vertices hinzu*



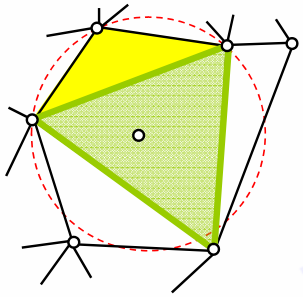
- Finde ein Dreieck, das den hinzugefügten Vertex enthält.
- Prüfe Umkreis der benachbarten Dreiecke. Markiere die Dreiecke die LCC verletzen

Delaunay-Triangulierung

66

Inkrementeller Algorithmus

1. *Beginne mit einem Dreieck (3 beliebige Vertices)*
2. *Füge nacheinander weitere Vertices hinzu*



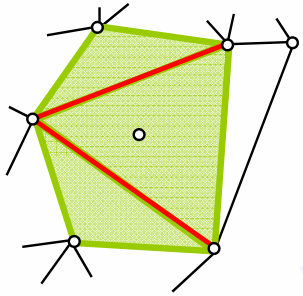
- Finde ein Dreieck, das den hinzugefügten Vertex enthält.
- Prüfe Umkreis der benachbarten Dreiecke. Markiere die Dreiecke die LCC verletzen

Delaunay-Triangulierung

67

Inkrementeller Algorithmus

1. *Beginne mit einem Dreieck (3 beliebige Vertices)*
2. *Füge nacheinander weitere Vertices hinzu*



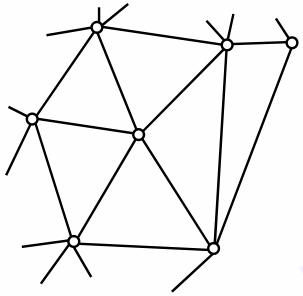
- Finde ein Dreieck, das den hinzugefügten Vertex enthält.
- Prüfe Umkreis der benachbarten Dreiecke. Markiere die Dreiecke die LCC verletzen
- Entferne alle inneren Kanten der markierten Region

Delaunay-Triangulierung

68

Inkrementeller Algorithmus

1. *Beginne mit einem Dreieck (3 beliebige Vertices)*
2. *Füge nacheinander weitere Vertices hinzu*



- Finde ein Dreieck, das den hinzugefügten Vertex enthält.
- Prüfe Umkreis der benachbarten Dreiecke. Markiere die Dreiecke die LCC verletzen
- Entferne alle inneren Kanten der markierten Region
- Verbinde neuen Vertex mit allen Randkanten der Region

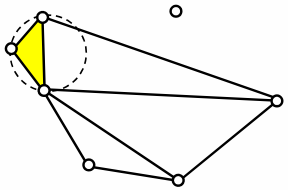
Delaunay-Triangulierung

Inkrementeller Algorithmus

1. *Beginne mit einem Dreieck (3 beliebige Vertices)*
2. *Füge nacheinander weitere Vertices hinzu*

Sonderfall: Externer Vertex

- Prüfe Umkreis aller Dreiecke. Markiere die Dreiecke die LCC verletzen



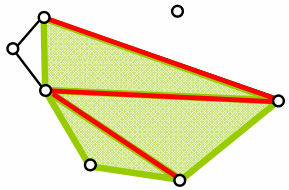
Delaunay-Triangulierung

Inkrementeller Algorithmus

1. *Beginne mit einem Dreieck (3 beliebige Vertices)*
2. *Füge nacheinander weitere Vertices hinzu*

Sonderfall: Externer Vertex

- Prüfe Umkreis aller Dreiecke. Markiere die Dreiecke die LCC verletzen
- Entferne alle Kanten der markierten Region, die LCC verletzen



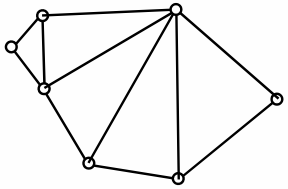
Delaunay-Triangulierung

Inkrementeller Algorithmus

1. *Beginne mit einem Dreieck (3 beliebige Vertices)*
2. *Füge nacheinander weitere Vertices hinzu*

Sonderfall: Externer Vertex

- Prüfe Umkreis aller Dreiecke. Markiere die Dreiecke die LCC verletzen
- Entferne alle Kanten der markierten Region, die LCC verletzen
- Verbinde neuen Vertex mit allen Randkanten der Region



Algorithmen

Algorithmen zur Bestimmung der Delaunay Triangulierung einer gegebenen Punktemenge

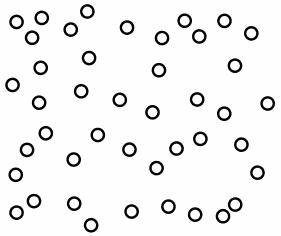
- *Flipping Algorithmus*
- *Inkrementeller Algorithmus*
- *Divide & Conquer-Algorithmus*

Vorteile: keine initiale Triangulierung
Nachteil: etwas komplizierter als *Flipping*

Delaunay-Triangulierung

Divide and Conquer-Algorithmus

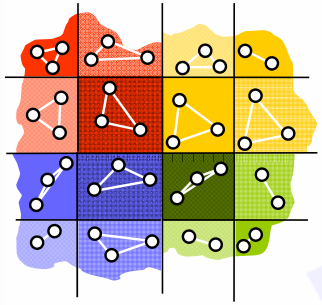
Schritt 1 (Teilen): *Teile die Menge der Dreiecke solange, bis die Triangulierung trivial wird.*



Delaunay-Triangulierung

Divide and Conquer-Algorithmus

Schritt 1 (Teilen): *Teile die Menge der Dreiecke solange, bis die Triangulierung trivial wird.*

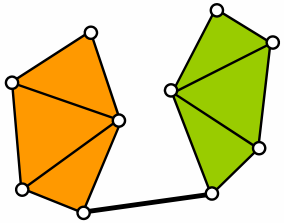


- Zerlege die Menge in Teilmengen
- Fahre solange fort bis jede Teilmenge 2 oder 3 Vertices enthält.
  (Achte darauf, dass die Teilmengen annähernd quadratische Form haben)
- Bestimme (triviale) Triangulierung der Teilmengen

Delaunay-Triangulierung

Divide and Conquer-Algorithmus

Schritt 2 (Vereinen): *Vereinige die einzelnen Teilmengen schrittweise.*

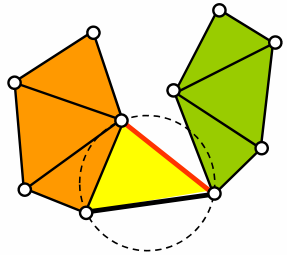


- Beginne mit der „untersten“ Kante

Delaunay-Triangulierung

Divide and Conquer-Algorithmus

Schritt 2 (Vereinen): *Vereinige die einzelnen Teilmengen schrittweise.*



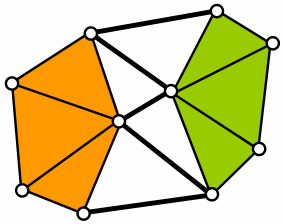
- Beginne mit der „untersten“ Kante
- Bilde mit einer der nächsten „untersten“ Kanten ein Dreieck, das LCC nicht verletzt. (*Dies ist nicht immer möglich!* → *Sonderfall*)

Delaunay-Triangulierung

77

Divide and Conquer-Algorithmus

Schritt 2 (Vereinen): *Vereinige die einzelnen Teilmengen schrittweise.*



- Beginne mit der „untersten“ Kante
- Bilde mit einer der nächsten „untersten“ Kanten ein Dreieck, das LCC nicht verletzt. (*Dies ist nicht immer möglich! → Sonderfall*)
- Fahre so fort

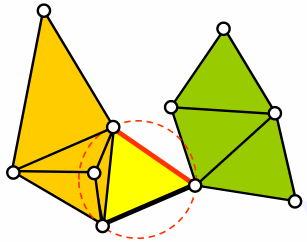
Delaunay-Triangulierung

78

Divide and Conquer-Algorithmus

Schritt 2 (Vereinen): *Vereinige die einzelnen Teilmengen schrittweise.*

Sonderfall: Einfügen einer Kante nicht möglich



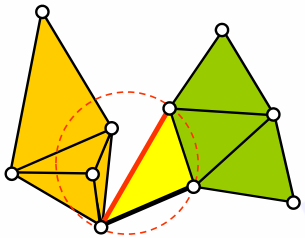
Delaunay-Triangulierung

79

Divide and Conquer-Algorithmus

Schritt 2 (Vereinen): *Vereinige die einzelnen Teilmengen schrittweise.*

Sonderfall: Einfügen einer Kante nicht möglich



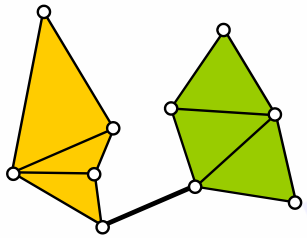
Delaunay-Triangulierung

80

Divide and Conquer-Algorithmus

Schritt 2 (Vereinen): *Vereinige die einzelnen Teilmengen schrittweise.*

Sonderfall: Einfügen einer Kante nicht möglich



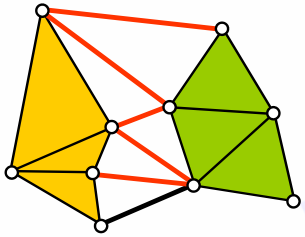
- *Grund:* Es existiert bereits eine Kante, die LCC verletzt!
- Bestimme die Kante, die LCC verletzt.
- Entferne diese Kante und fahre fort

Delaunay-Triangulierung

81

Divide and Conquer-Algorithmus

Schritt 2 (Vereinen): Vereinige die einzelnen Teilmengen schrittweise.



Sonderfall: Einfügen einer Kante nicht möglich

- *Grund:* Es existiert bereits eine Kante, die LCC verletzt!
- Bestimme die Kante, die LCC verletzt.
- ✕ Entferne diese Kante und fahre fort

Zusammenfassung Algorithmen

82

- *Flipping Algorithmus*
  - Komplexität $o(n^2)$
  - sehr einfacher erfordert initiale Triangulierung
- *Inkrementeller Algorithmus*
  - Komplexität $o(n^2)$
- *Divide & Conquer-Algorithmus*
  - Komplexität $o(n \log n)$
  - relativ kompliziert
- Nicht besprochen: *(Plane-Sweep Algorithmus)*
  - Komplexität $o(n \log n)$
  - sehr kompliziert

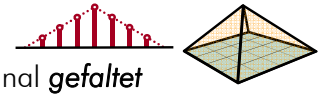
Verwenden bei kleinen Netzen (optimal für $n < 1000$)

Verwenden bei großen Netzen (optimal für $n > 100000$)

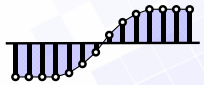
Zusammenfassung

83

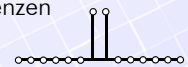
Lineare Filter:



- *Filterkern* wird mit Signal *gefaltet*
- Lineare Filter entfernen bestimmte *Frequenzen* aus dem Signal
- Glättungsfilter:
  - Box-, Dreieck-, Gauss-Filter
  - *Tiefpassfilter:* hohe Frequenzen werden entfernt



- Kantenfilter:
  - *Hochpass-Filter:* niedrige Frequenzen werden abgeschwächt

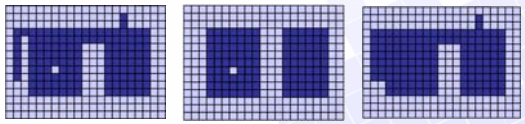


Zusammenfassung

84

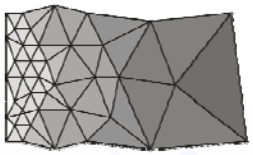
Nicht-Lineare Filter:

- Keine *Faltung,* sondern beliebige nichtlineare Verknüpfung
- Rangordnungsoperationen:
  - Median-Filter: Rauschunterdrückung ohne Kantenglättung
  - Morphologische Operationen Dilatation, Erosion, Opening, Closing

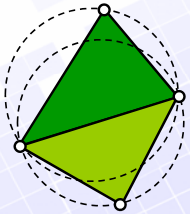


Zusammenfassung

Triangulierungen:



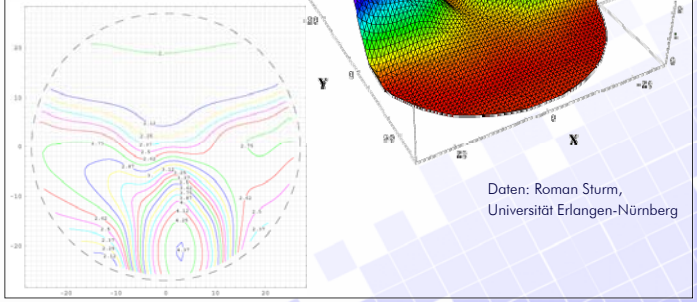
- *Reguläre Triangulierungen*
- *Delaunay-Triangulierung*
  - Definition über Voronoi-Diagramm
  - globales und lokales Umkreis-Kriterium
- *Algorithmen zur Triangulierung:*
  - Flipping-Algorithmus
  - Inkrementeller Algorithmus
  - Divide & Conquer Algorithmus



Nächste Stunde

2D Skalarfelder

- Isolinien
- Surface Plots



Daten: Roman Sturm, Universität Erlangen-Nürnberg