Wasserscheiden-Ansätze zur Bildsegmentierung II

Johannes Renfordt

renfordt@mathematik.uni-ulm.de

Seminar „Bildsegmentierung und Computer Vision“
im Wintersemester 2005/2006

Universität Ulm
Fakultät für Mathematik und Wirtschaftswissenschaften
Abteilung Stochastik

Übersicht

- Einführung
  - graphentheoretische Begriffe
  - Mosaike
- Praxis
  - 4er vs. 8er-Nachbarschaft
  - Vorbearbeitung: Glättungsfilter
  - Idee einer Nachbearbeitung
  - Implementierung
- Eigenschaften von Mosaiken

Einführung
Graphentheorie I

Die Graphentheorie ist ein hilfreiches Mittel zur Beschreibung von Bildern:

- sei E eine endliche Menge von Punkten oder Knoten
- sei $\Gamma \subseteq E \times E$ die Menge der Kanten mit
  - $(x, x) \in \Gamma \; \forall x \in E$
  - $(x, y) \in \Gamma \Leftrightarrow (y, x) \in \Gamma \; \forall x, y \in E$.

(E, Γ) heißt dann ein **ungerichteter Graph**

Weitere Definitionen:
sei (E, Γ) ein Graph, sei $X \subseteq E$.

- $\Gamma(x) := \{y \in E | (x, y) \in \Gamma\}$ heißt **Nachbarschaft** von $x \in E$
- Seien $x_0, x_n \in X$. Ein Pfad von x_0 nach x_n in X ist eine Sequenz $\pi = (x_0, x_1, \ldots, x_n)$ von Punkten (in X), so daß $x_{i+1} \in \Gamma(x_i)$, $i = 0, \ldots, n-1$.
- Seien $x, y \in X$. Die Punkte x und y heißen **in X verbunden**, falls ein Pfad von x nach y in X existiert.
- X heißt **zusammenhängend**, falls $\forall x, y \in X$ die Punkte x und y in X verbunden sind.

Im Folgenden sei der Graph (E, Γ) zusammenhängend.

Sei (E, Γ) ein Graph, sei $X \subseteq E$.

- $Y \subseteq E$ heißt **verbundene Komponente** oder **Zusammenhangskomponente**, falls
  - $Y \subseteq X$,
  - Y ist zusammenhängend

  und Y in diesen Eigenschaften maximal ist.
- Sei $\mathcal{F}(E)$ die Menge aller Abbildungen von E nach $\mathbb{Z}$. Eine Abbildung $F \in \mathcal{F}(E)$ heißt ein **Bild** und $F(x)$ wird die Höhe von x bezüglich F genannt, falls $x \in E$.

Mosaike
Minima-Erweiterung

Definition: Sei (E, Γ) ein Graph.
Sei X eine Teilmenge von E und $F \in \mathcal{F}(E)$ ein Bild. X heißt **Minima-Erweiterung** von F, falls

- jede verbundene Komponente von X genau ein Minimum von F von enthält und
- jedes lokale Minimum von F ist in einer verbundenen Komponente von X enthalten.

Das Komplement einer Minima-Erweiterung von F in E heißt **Trennungsmenge** von F.

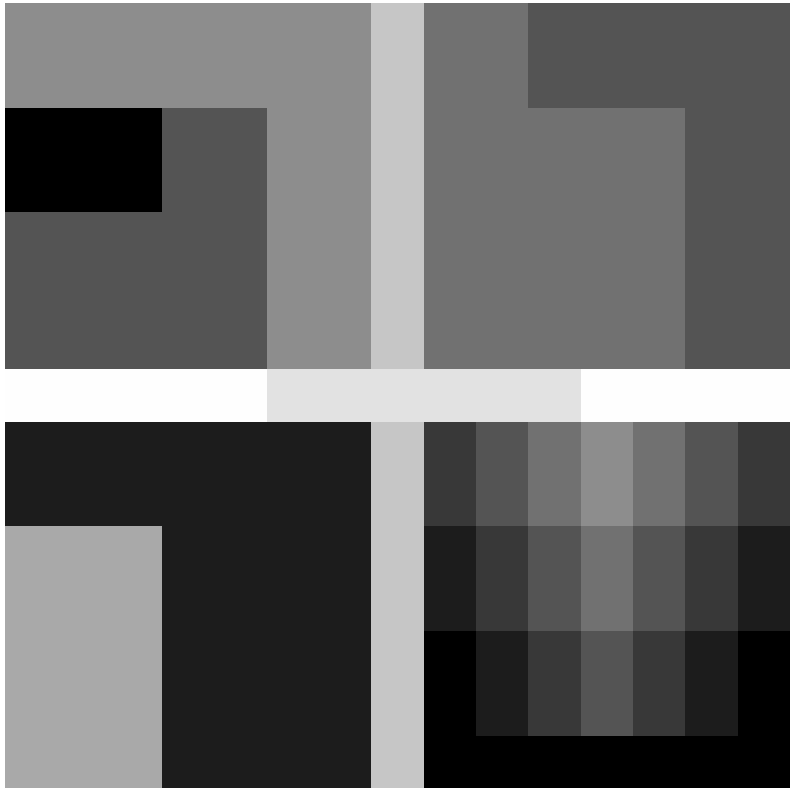
Mosaike
Definition Mosaik

Definition: Sei $F \in \mathcal{F}(E)$ und sei X eine Minima-Erweiterung von F. Das **Mosaik** von F bezüglich X ist ein Bild $F_X \in F(E)$, für das gilt:

- $F_X(x) = F(x)\ \forall x \notin X$ und
- $F_X(x) = \min\{F(y) | y \in C_x\}\ \forall x \in X$, wobei C_x die Zusammenhangskomponente angibt, die x enthält.

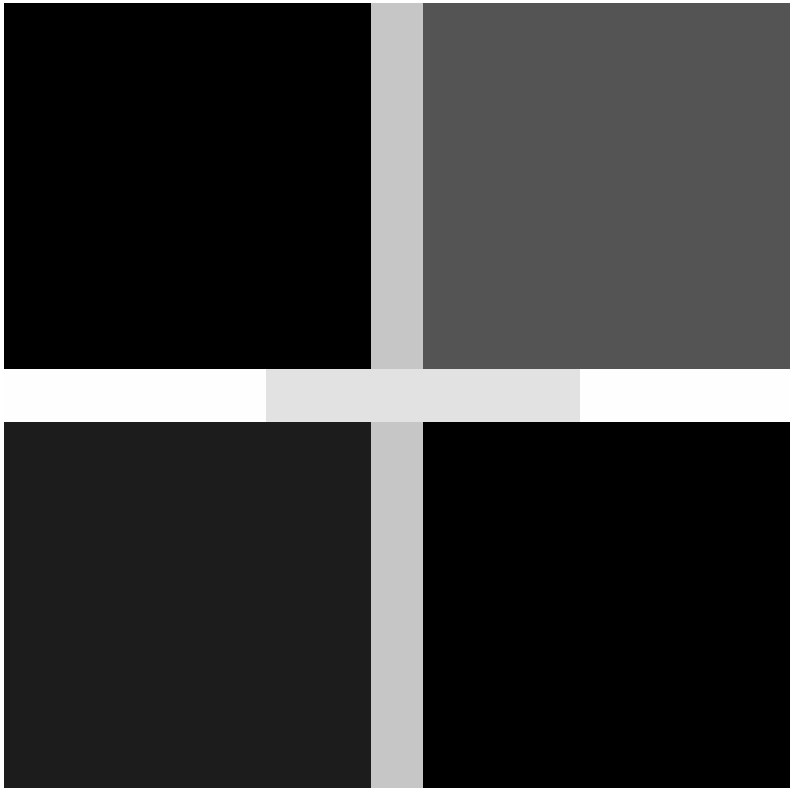
Mosaike
Beispiel

Beispiel für ein zu bearbeitendes Bild:



Mosaike
Beispiel

Beispiel für ein zu bearbeitendes Bild:



Mosaike
Idee des Flutungs-Algorithmus

Die Grundidee hinter Flutungs-Algorithmen ist folgende:

- bohre Löcher an den Minimumstellen
- pumpe Wasser durch diese Löcher
- wenn so entstandene Seen sich verbinden, errichte an den Berührungspunkten Dämme

Mosaike
Ein konkreter Algorithmus I

Ein möglicher Flutungs-Algorithmus:

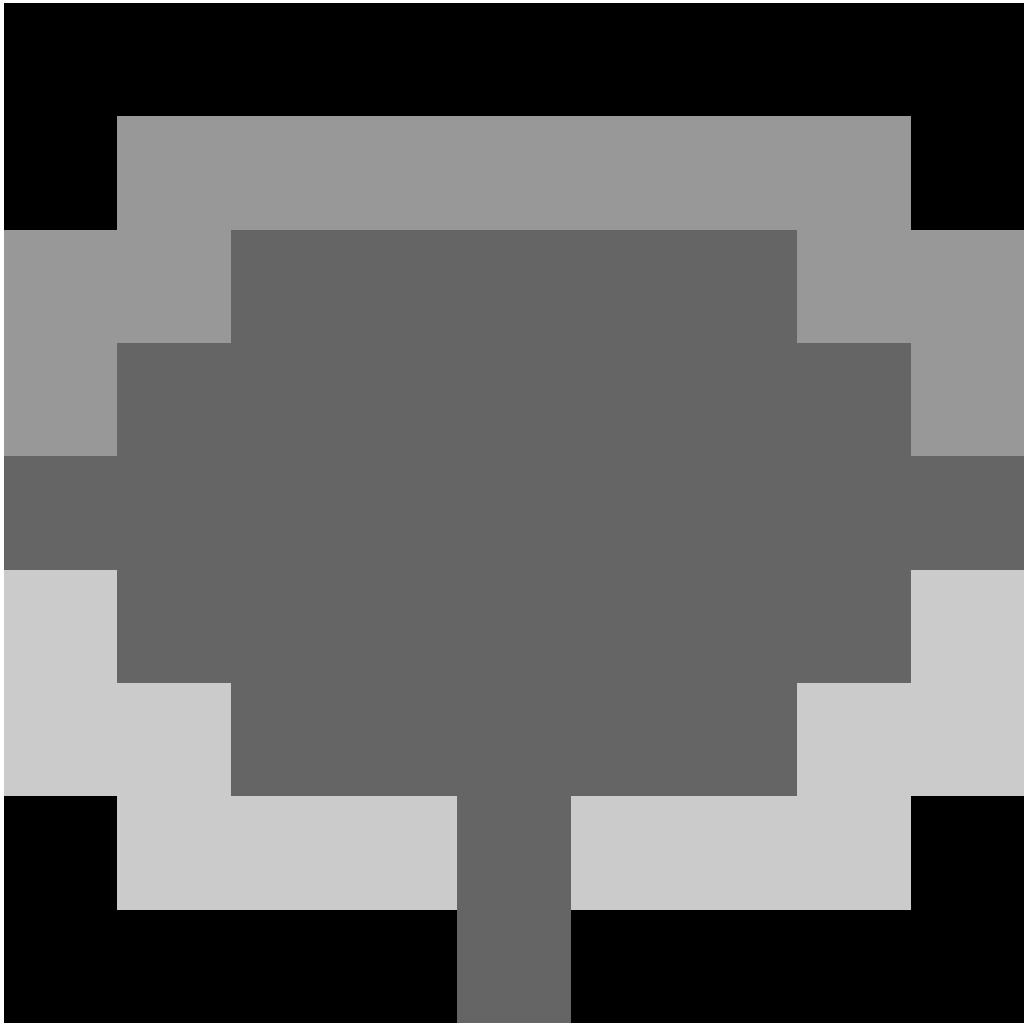
- 1. Weise jedem Minimum eine eindeutige Kennzeichnung zu;
- 2. Markiere jeden Punkt, der zu einem Minimum gehört mit der entsprechenden Markierung;
- 3. die Mengen Q und V sind leer;
- 4. Füge jeden nicht-markierten Nachbar der markierten Punkte der Menge Q hinzu;

Mosaike
Ein konkreter Algorithmus II

- 5. Wähle (und entferne) aus Q denjenigen Punkt x mit der geringsten Höhe und füge x der Menge V hinzu. Falls alle markierten Punkte aus der Nachbarschaft $\Gamma(x)$ von x dieselbe Markierung haben, dann
  - kennzeichne x mit dieser Markierung und
  - füge Q alle Punkte $y \in \Gamma(x)$ hinzu, für die gilt $y \notin Q \cup V$
- 6. Wiederhole Schritt 5. solange, bis die Menge Q leer ist.

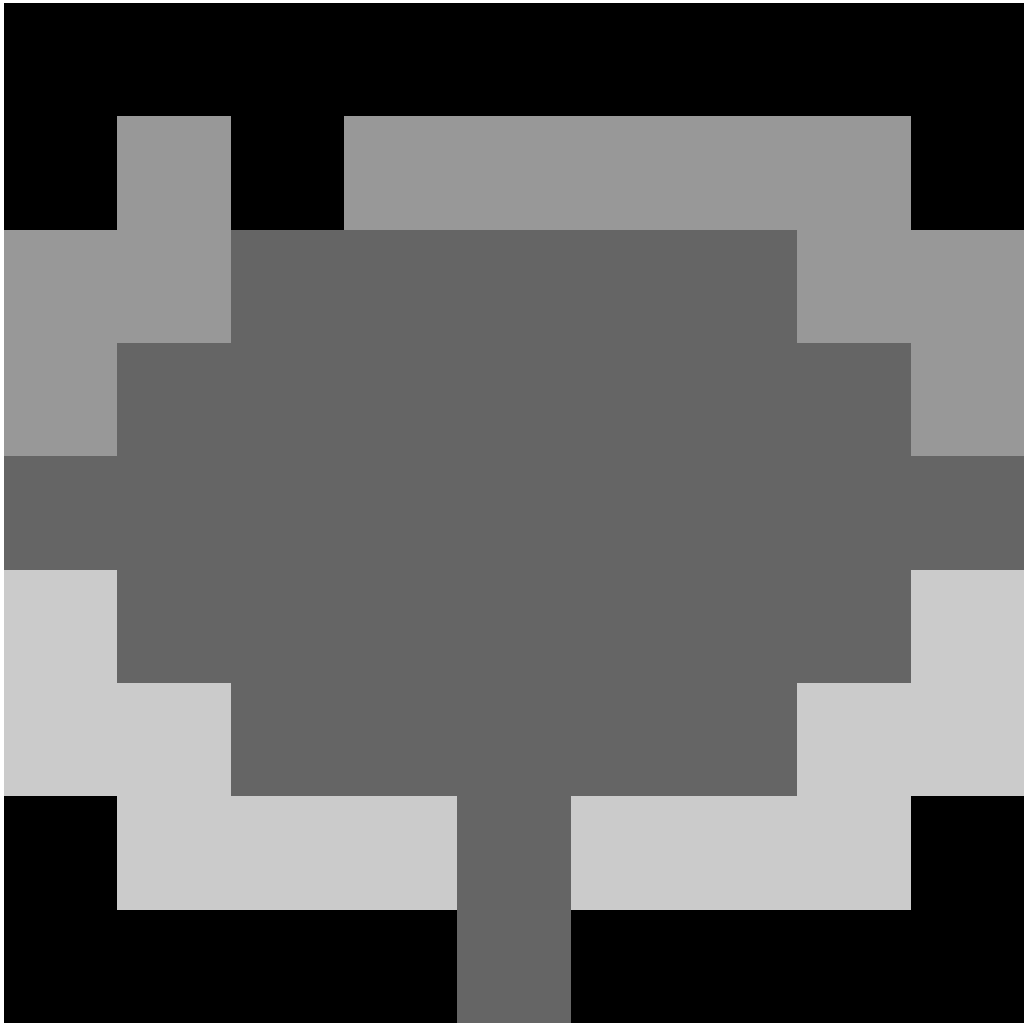
Mosaike
Beispiel

Beispiel, 9×9 Pixel



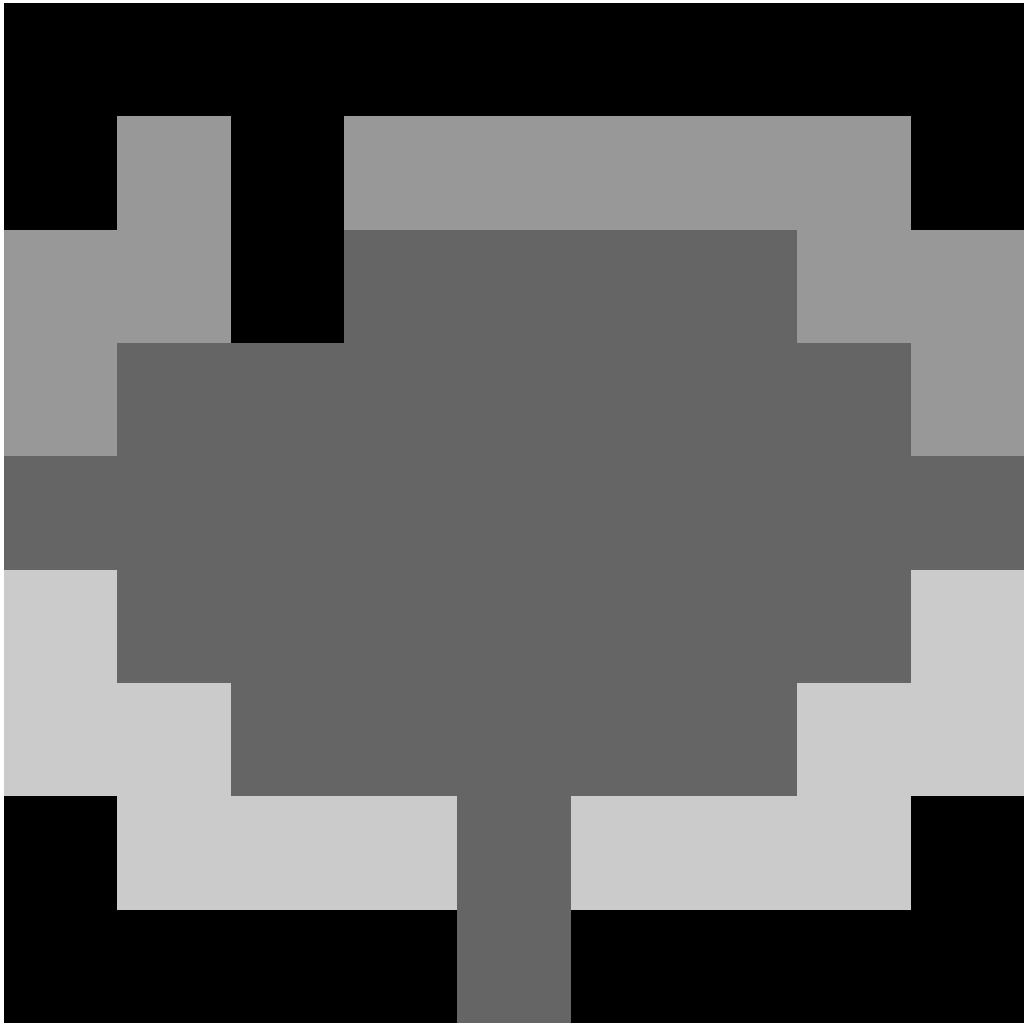
Mosaike
Beispiel

Beispiel, 9×9 Pixel



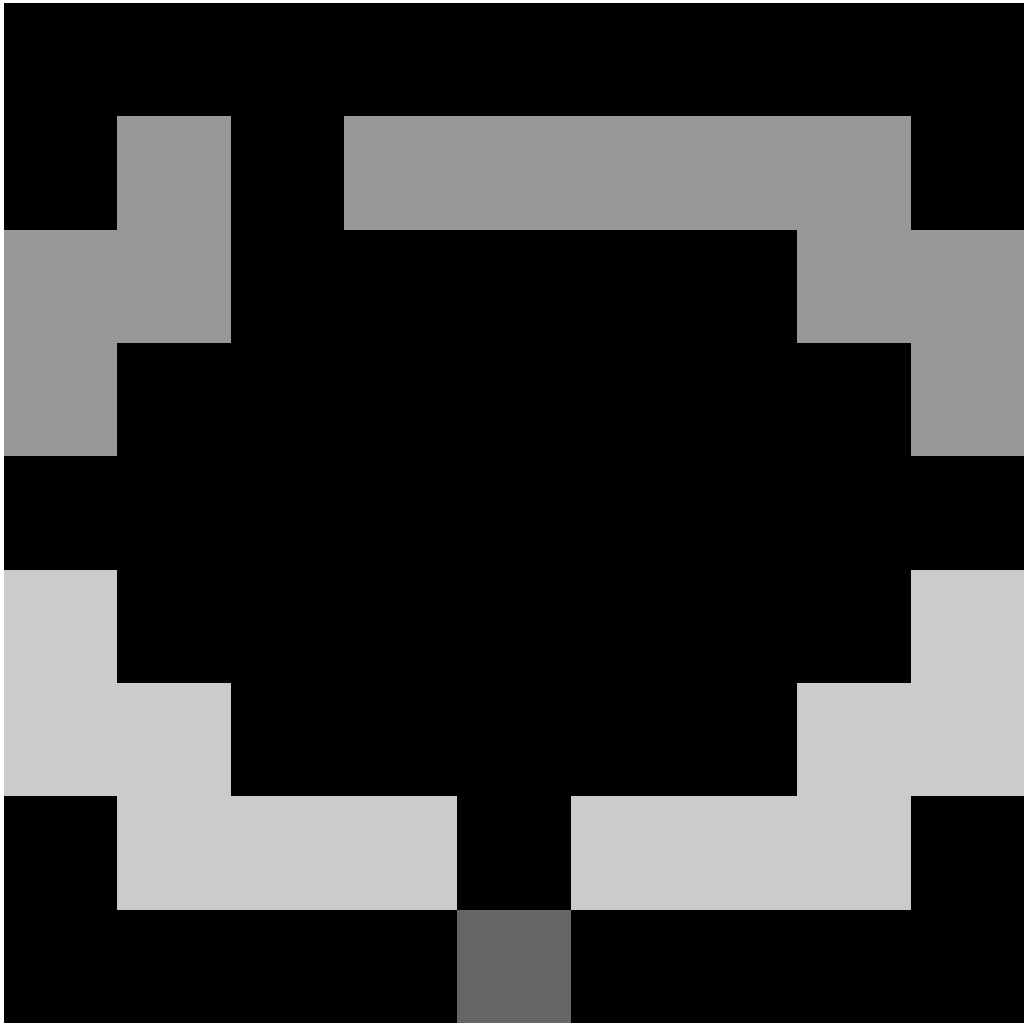
Mosaike
Beispiel

Beispiel, 9×9 Pixel



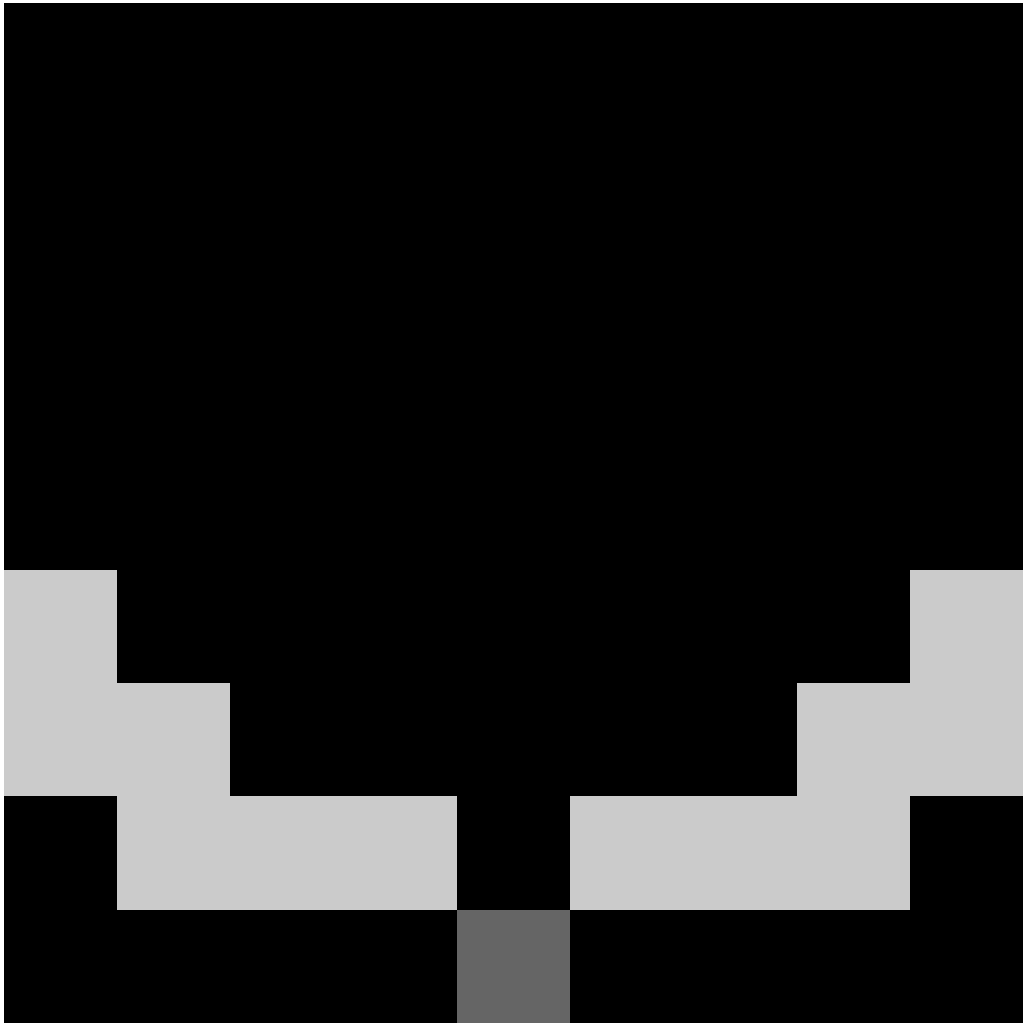
Mosaike
Beispiel

Beispiel, 9×9 Pixel



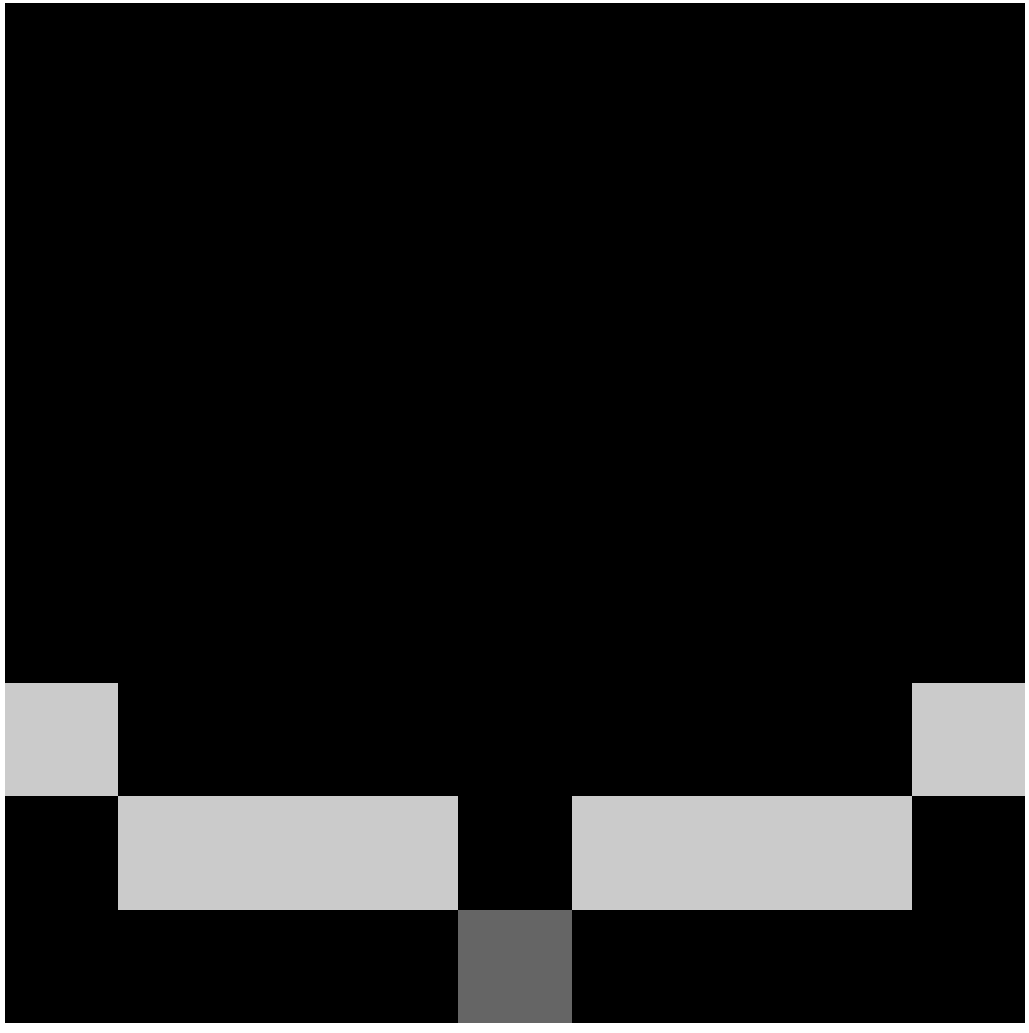
Mosaike
Beispiel

Beispiel, 9×9 Pixel



Mosaike
Beispiel

Beispiel, 9×9 Pixel



Übersicht

- Einführung
  - graphentheoretische Begriffe
  - Mosaike
- Praxis
  - 4er vs. 8er-Nachbarschaft
  - Vorbearbeitung: Glättungsfilter
  - Idee einer Nachbearbeitung
  - Implementierung
- Eigenschaften von Mosaiken

Mosaike

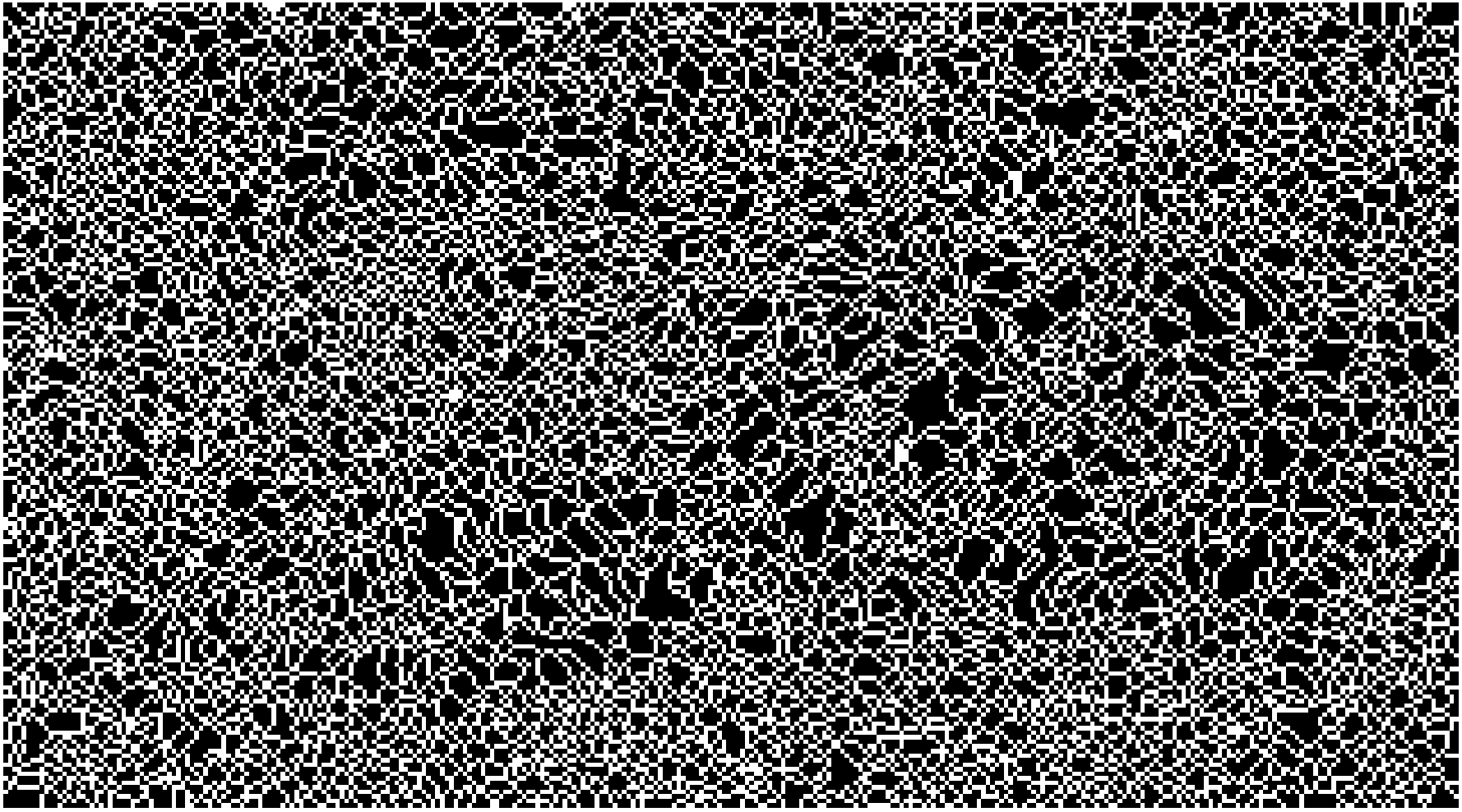
Beispiele I

Original, 320 x 177 Pixel, 256 Graustufen

Mosaike

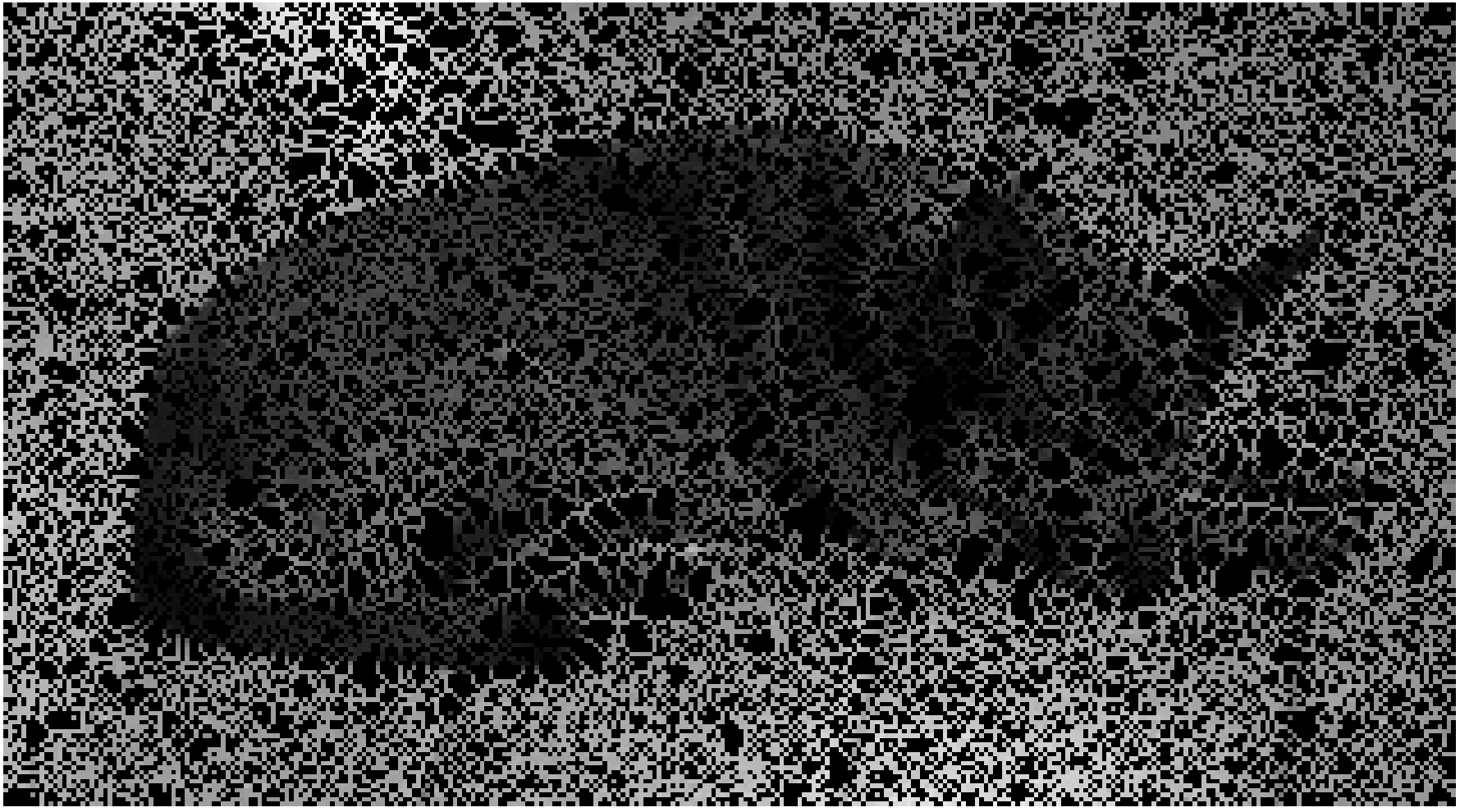
Beispiele II

normale Wasserscheidentransformation ...



Mosaike
Beispiele III

... und das dazugehörende Mosaik (4er-Nachbarschaft)



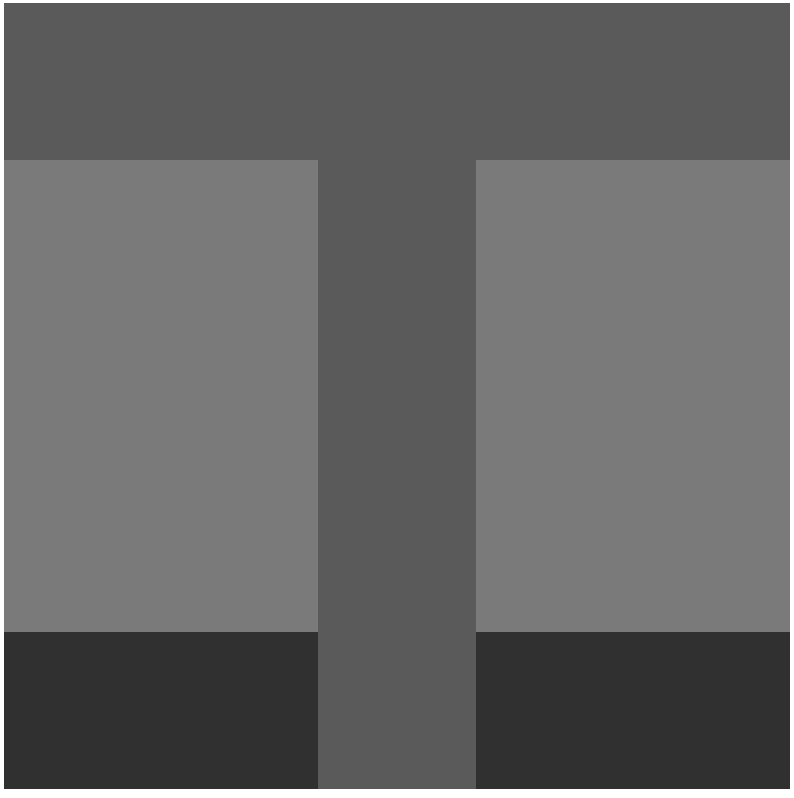
Mosaike
Problem Übersegmentierung

- das gezeigte Bild hat lediglich 56.640 Bildpunkte, ist also recht klein
- jedoch existieren 4.408 lokale Minima, die zudem auch noch aus mehreren Punkten bestehen können!
- auf dem Mosaik existieren Stellen, an denen die Trennungslinien nur unwesentlich heller (also nur wenig höher) als die Zusammenhangskomponenten sind
- Verbesserungen sollten also möglich sein

Mosaike

Problem Nicht-Eindeutigkeit

Oft ist es von der genauen Implementierung abhängig, welches Minimum-Gebiet zuerst ein Plateau oder ein bisher nicht zugängliches Gebiet für sich erschließt:



8er-Nachbarschaft
Idee

Idee: Übergang zu 8er Nachbarschaften, d.h. für jedes Pixel zählen als Nachbarn nicht mehr nur horizontale und vertikale, sondern auch diagonal benachbarte Pixel

- liegt ein Minimum bei 8er-Nachbarschaft vor, ist es auch ein Minimum bei 4er-Nachbarschaft
- die Umkehrung gilt i.a. nicht

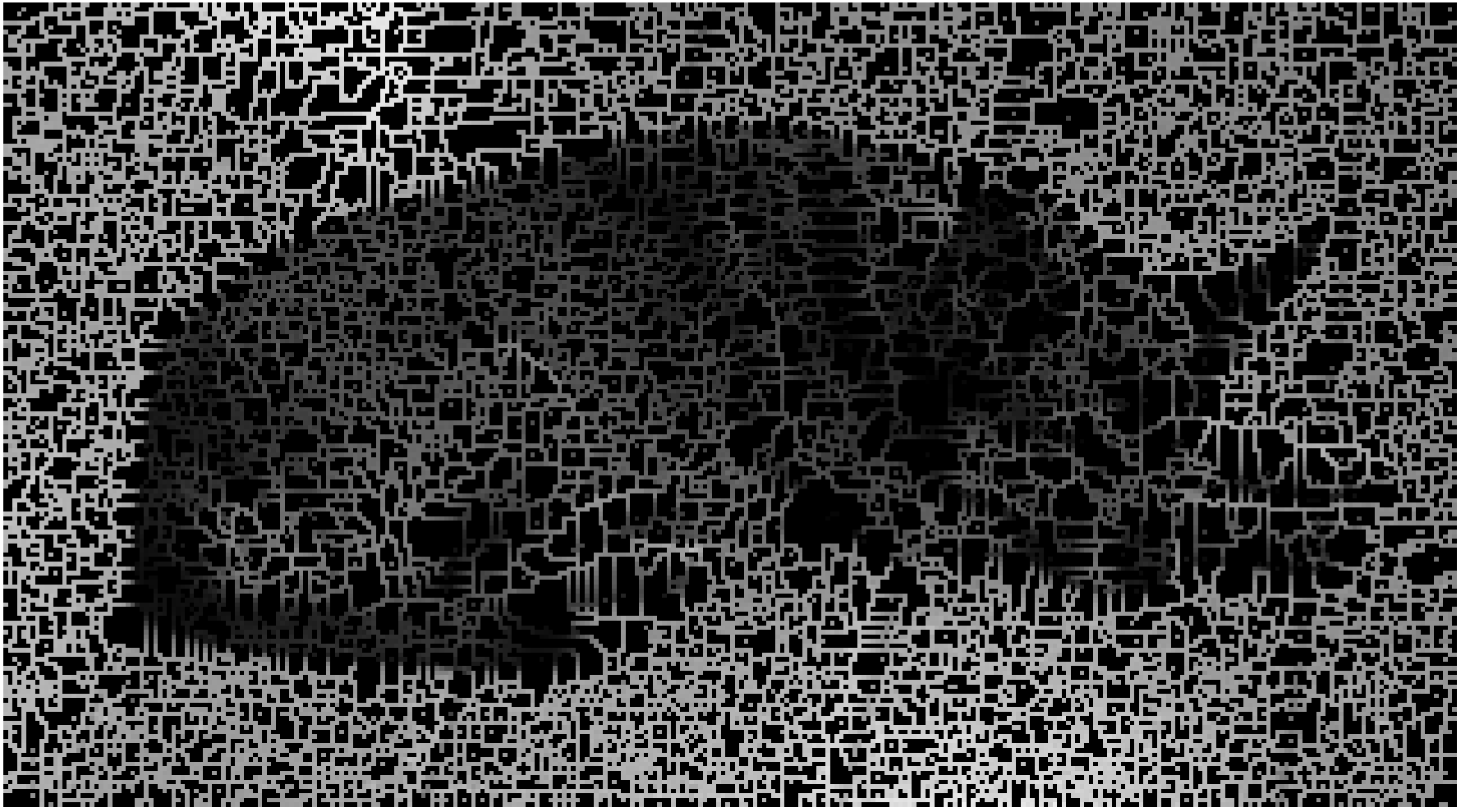
$\Rightarrow$ bei der Verwendung von 8er-Nachbarschaften verringert sich tendenziell die Anzahl der Minima

In der praktischen Anwendung ist diese Verringerung beträchtlich - und der Übergang zu 8er-Nachbarschaften ist leicht zu implementieren!

8er-Nachbarschaft

Beispiel

nur noch 2835 Minima mit 8er-Nachbarschaften



Vorbearbeitung
Idee: Glätten I

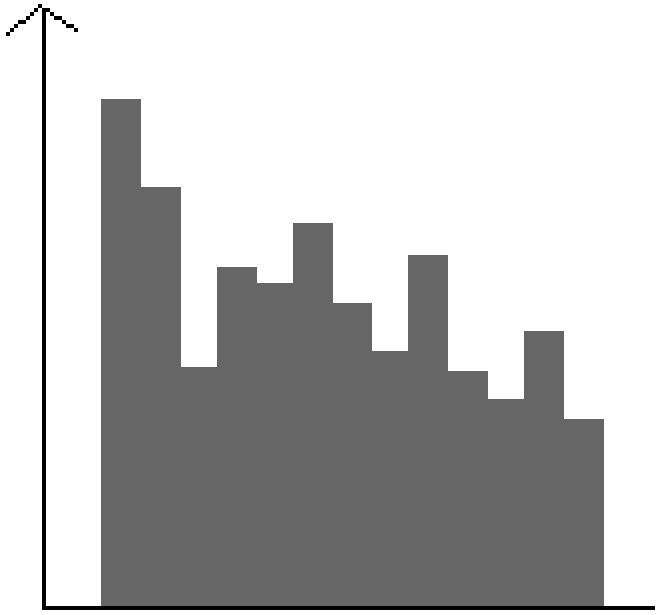
Vor der Segmentierung kann das Bild geglättet werden. Beispiele für Glättungsverfahren (*Filter*):

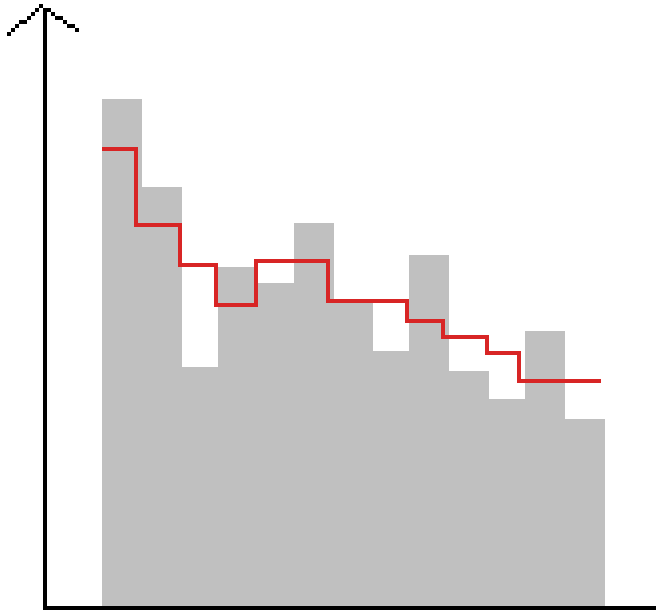
- Mittelwertfilter
- Medianfilter
- Gaußfilter

Dabei wird eine Umgebung von $n \times m$ Pixeln um das zu modifizierende Pixel benutzt

Vorbearbeitung
Idee: Glätten II

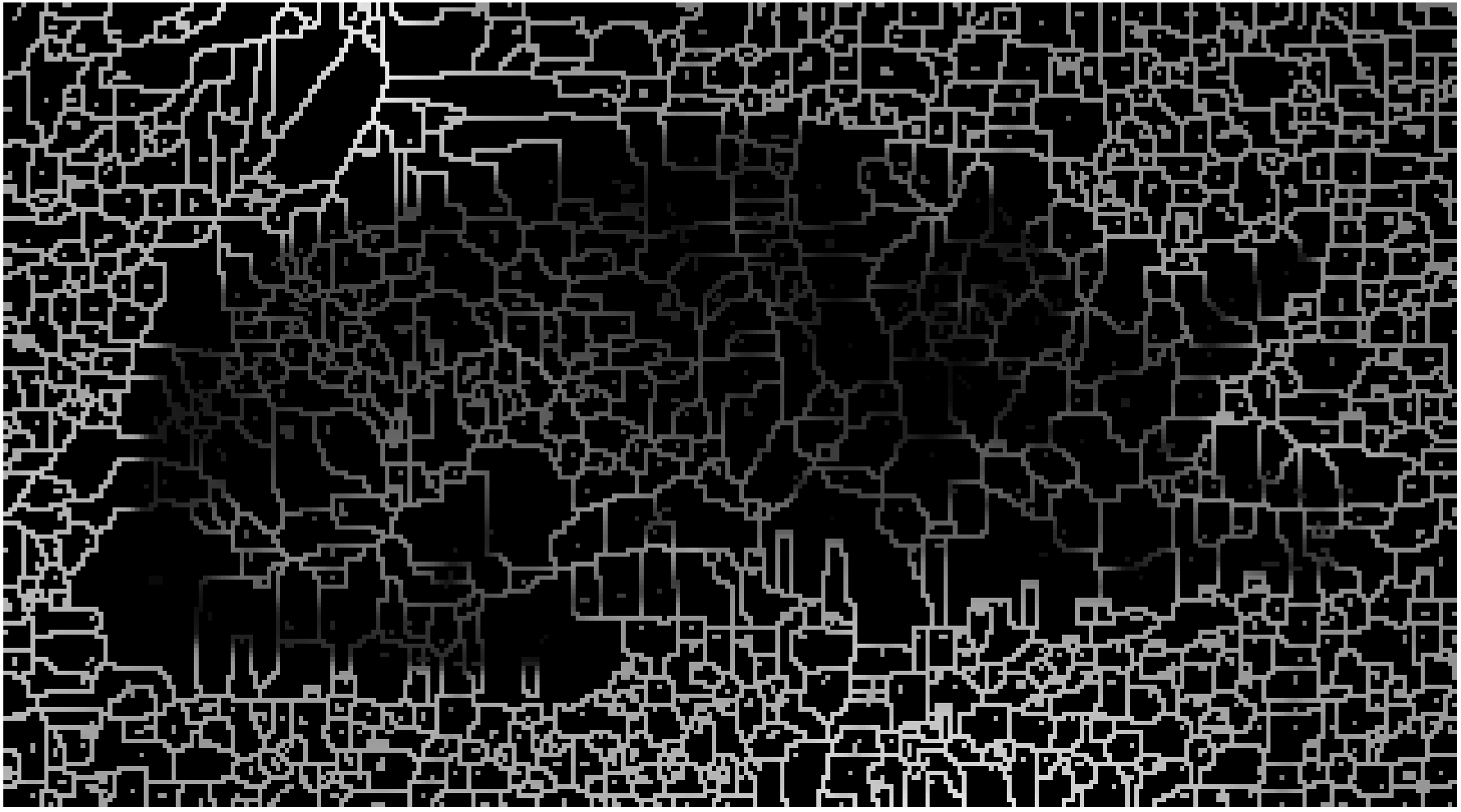
Motivierendes Beispiel:





Vorbearbeitung
Idee: Glätten III

geglättet wurde mit einem 3×3-Mittelwertfilter



Vorbearbeitung
Idee: Glätten IV

- durch die Glättung und Verwendung der 8er-Nachbarschaft konnte die Anzahl der Minima von 4.408 auf 840 verringert werden!
- die Umgebung zum Glätten sollte nicht zu groß sein
- Die Wahl des Filters ist nicht unproblematisch:
  - ein Medianfilter kann Objekte, die nur aus ihren eigenen Konturen bestehen, im Extremfall vollständig eliminieren
  - ein Mittelwertfilter reagiert sensibel auf Ausreißer

Nachbearbeitung
Idee

Oft hat das entstandene Mosaik noch wesentlich mehr Zusammenhangskomponenten, als man sich erhofft.

Eine Idee kann sein:

Vereinige paarweise die Minima, deren Trennungsmengen nur unwesentlich höher als die minimalen Höhen der jeweiligen Minima sind.

Problematisch bleibt aber die Frage, ab welcher Höhendifferenz tatsächlich zwei Minima vereinigt werden.

Implementierung
Allgemein

- Objektorientiertes Design, hier: C++
- als Grundlagen bieten sich die Klassen *pixel* und *image* an
- Anhand des Algorithmus wird ein Satz Funktionen für die Klassen geschrieben
- Entscheidung für ein Grafikformat, hier: PGM (*Portable Greymap*) in binärer Darstellung

Implementierung
Portable Greymap

Im Kopf der PGM-Datei werden durch Whitespace getrennt angegeben:

- Magische Zahl
  - P2 ASCII oder
  - P5 binäre Darstellung
- Breite und Höhe
- maximaler Grauwert

Ein maximaler Grauwert von 255 und die binäre Darstellung passen sehr gut zueinander: dann belegt jedes Pixel genau ein Byte

Implementierung
Klasse image

Mögliche Schnittstelle der Klasse image:

```
namespace images {

        class  image {
        public:
                image(char*,int);
                void  print ();
                void  markMinima();
                void  watershed();
                void  printMinima();
                void  printWatershed();
                void  printMosaic();
                void  smooth(int);
        };
}
```

Implementierung
Klasse pixel

Mögliche Schnittstelle der Klasse pixel:

```
namespace images {

        class  pixel  {
        public:
                minimum min;
                pixel ( int );
                void  print ();
                inline   int  getValue () {
                        return  grauwert;
                }
                void  setNeighbour(pixel*);
                list <pixel*>* getNeighbours();
                void  saveSmoothing(int);
                void  setSmoothing();
        };
}
```

Implementierung
Minima finden

Im ersten Schritt des Algorithmus heißt es: weise jedem Minimum eine eindeutige Markierung zu.
Wie kann man die Minima im Bild finden?

- jedes Pixel ist ein potentielles Minimum
- vergleiche jedes Pixel mit seinen Nachbarn
  - kein Minimum
  - Minimum
  - noch nicht entscheidbar
- wiederholt: betrachte Nachbarn der Nicht-Minima und markiere sie ggf. auch als Nicht-Minima
- alle jetzt noch potentiellen Punkte sind Teile von Minima, die mehr als ein Pixel umfassen

Implementierung
Implementierung der Warteschlange

In der Menge Q werden die abzuarbeitenden Pixel abgelegt.
Aus Q wird jeweils das Pixel mit dem kleinsten Grauwert entnommen

$\Rightarrow$ Implementierung durch eine sortierte verkette Liste bietet sich an:

```
typedef struct {
            priorList * next;
            pixel * p;
} priorList ;
```

Implementierung
Problem: Rechenzeit

Leider ist die Sache nicht unproblematisch:

- der mathematische Algorithmus ist effektiv, in der direkten Umsetzung aber leider nicht sonderlich effizient, denn
- bei sehr großen Bildern wird die Liste der noch abzuarbeitenden Punkte sehr lang - das Einsortieren an der richtigen Stelle kann sehr aufwendig sein
- eine kleine Erleichterung schafft die Überprüfung, ob ein Pixel noch in die Liste einsortiert werden muß oder nicht

Hier ist es angebracht, einen Container zu implementieren, der die Schnittstellen einer Liste anbietet, intern aber effizienter organisiert ist.

Implementierung
Problem: Arbeitsspeicher

Die zu bearbeitenden Bilder können nicht beliebig groß sein:

- der verfügbare Arbeitsspeicher ist in aller Regel sehr begrenzt
- ein ca. 4 Millionen Pixel großes Bild braucht zur Laufzeit etwa 680 bis 725 MB Arbeitsspeicher
- reicht der physikalische Speicher nicht aus, wird Speicher ausgelagert $\rightarrow$ sehr zeitintensiv

Übersicht

- Einführung
  - graphentheoretische Begriffe
  - Mosaike
- Praxis
  - 4er vs. 8er-Nachbarschaft
  - Vorbearbeitung: Glättungsfilter
  - Idee einer Nachbearbeitung
  - Implementierung
- Eigenschaften von Mosaiken

Eigenschaften von Mosaiken

Warum überhaupt Mosaike?

Mosaike enthalten u.a. die Information der Höhe der Trennungsmengen zwischen den einzelnen Minima.

Das Ziel bei einer Transformation ist es, die wesentlichen Bildeigenschaften zu erhalten. Darauf basiert der Begriff der Separation, der im Folgenden eingeführt werden wird.

Eigenschaften von Mosaiken
Paßhöhe

Definition: Sei $F \in \mathcal{F}(E)$, sei $\pi = (x_0, \ldots, x_n)$ ein Pfad im Graphen (E, Γ). Weiterhin sei die Höhe des Pfades durch $F(\pi) := \max\{F(x_i) | i = 0, \ldots, n\}$ definiert.

Seien x und y zwei Punkte aus E. Dann heißt $F(x, y) := \min\{F(\pi) | \pi \in \Pi(x, y)\}$ die **Paßhöhe** (bezüglich F) zwischen x und y, wobei $\Pi(x, y)$ die Menge aller Pfade zwischen x und y angibt.

Sind X und Y zwei Teilmengen von E, dann wird die Paßhöhe zwischen X und Y (bezüglich F) durch $F(X, Y) := \min\{F(x, y) | x \in X, y \in Y\}$ definiert.

Eigenschaften von Mosaiken
Separation I

Definition: Sei $F \in \mathcal{F}(E)$ ein Bild und seien $x, y \in E$.

Falls $F(x,y) > \max\{F(x), F(y)\}$ gilt, dann heißen x und y **separiert** (bezüglich F).

Sind x und y bezüglich F getrennt und gibt k die Paßhöhe $F(x,y)$ an, dann werden x und y auch als **k-separiert** bezüglich F bezeichnet.

Sei $G \in \mathcal{F}(E)$ und gelte $G \leq F$. Falls $\forall x, y \in E$ aus x und y sind k-separiert bezüglich F folgt, daß x und y auch k-separiert bezüglich G sind, dann heißt G eine **Separation**.

Eigenschaften von Mosaiken
Separation II

Bemerkungen:

- nicht jedes durch Wasserscheidentransformation erzeugte Mosaik ist eine Separation
- die Eigenschaft Separation anhand der Definition nachzuweisen, ist für größere Bilder sehr aufwendig

$\Rightarrow$ finde eine Äquivalenz zur Separation

Eigenschaften von Mosaiken
Definitionen

Definitionen:

- Sei $F \in \mathcal{F}(E)$. Die Menge $F_k := \{x \in E | F(x) \geq k\}$, $k \in \mathbb{Z}$, wird **Oberabschnitt** von F genannt.
- Sei $X \subseteq E$. Der Punkt $x \in X$ heißt **einfach** bezüglich F, wenn x zu genau einer Zusammenhangskomponente der Trennungsmenge benachbart ist.

Eigenschaften von Mosaiken
Ausdünnung

Definitionen:

Sei $F \in \mathcal{F}(E)$, $x \in E$ und $k = F(x)$.

- Der Punkt x heißt **zerstörbar** bezüglich F, wenn x einfach bezüglich F_k ist.
- Das Bild $G \in \mathcal{F}(E)$ wird **Ausdünnung** von F genannt, falls gilt:
  - $F = G$ oder
  - G kann aus F erzeugt werden, indem nach und nach die zerstörbaren Punkte entfernt werden.

Eigenschaften von Mosaiken

Theorem

Damit kann folgende Äquivalenz formuliert werden:

Theorem: Sei $F \in \mathcal{F}(E)$, $X \subseteq E$ eine Minima-Erweiterung von F und F_X das Mosaik von F bezüglich X. Dann gilt:

F_X ist eine Separation von F genau dann, wenn F_X eine Ausdünnung von F ist

Ende

Vielen Dank für die Aufmerksamkeit!