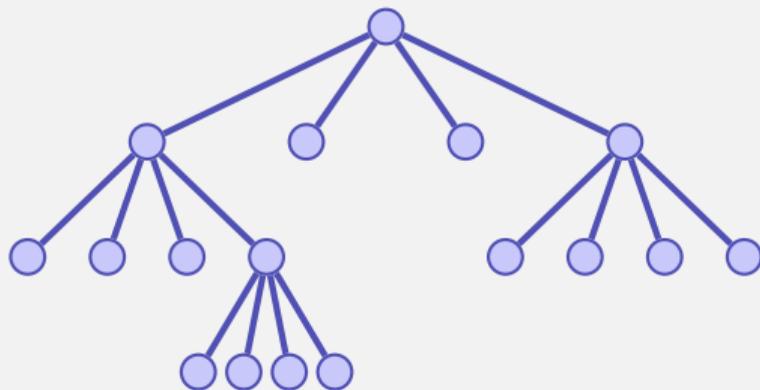


# 18. Quadrees

Quadrees, Kollisionsdetektion, Bildsegmentierung

# Quadtree

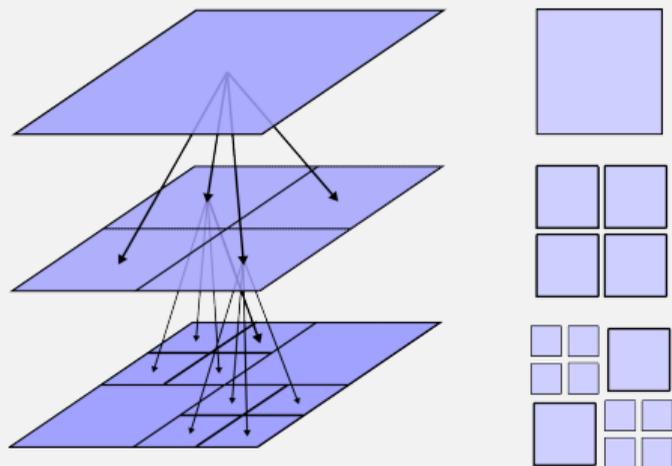
Ein Quadtree ist ein Baum der Ordnung 4.



... und ist als solcher nicht besonders interessant, ausser man verwendet ihn zur...

# Quadtree - Interpretation und Nutzen

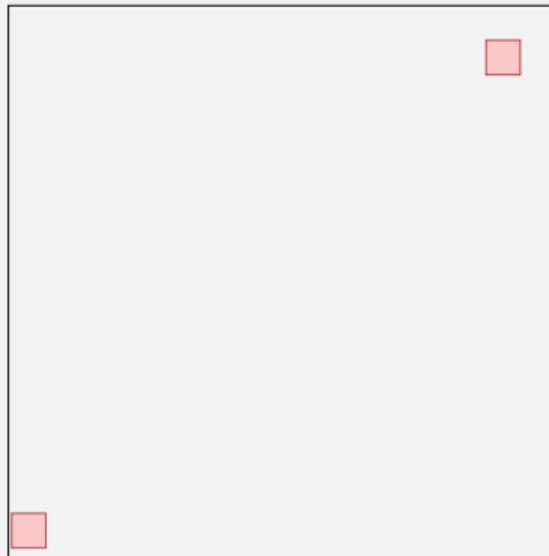
Partitionierung eines zweidimensionalen Bereiches in 4 gleich grosse Teile.



[Analog für drei Dimensionen mit einem *Octtree* (Baum der Ordnung 8)]

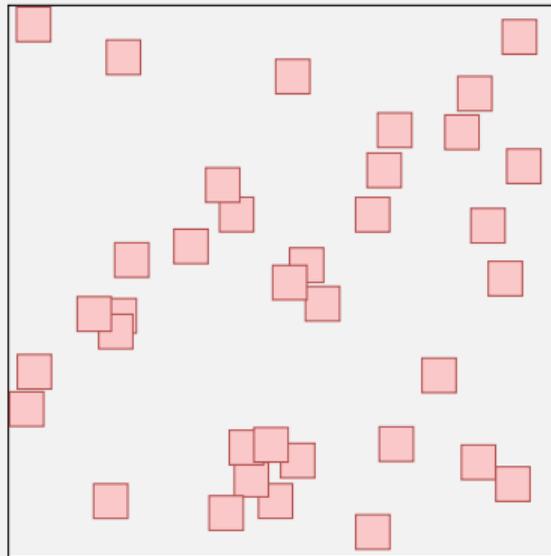
# Beispiel 1: Erkennung von Kollisionen

- Objekte in der 2D-Ebene, z.B. Teilchensimulation auf dem Bildschirm.
- Ziel: Erkennen von Kollisionen



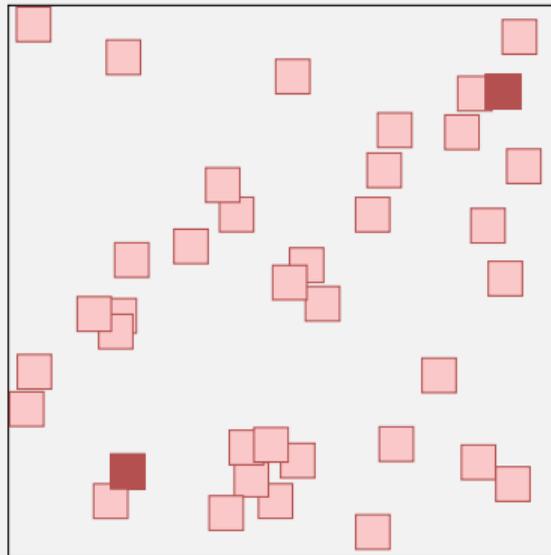
# Idee

- Viele Objekte:  $n^2$  Vergleiche (naiv)
- Verbesserung?



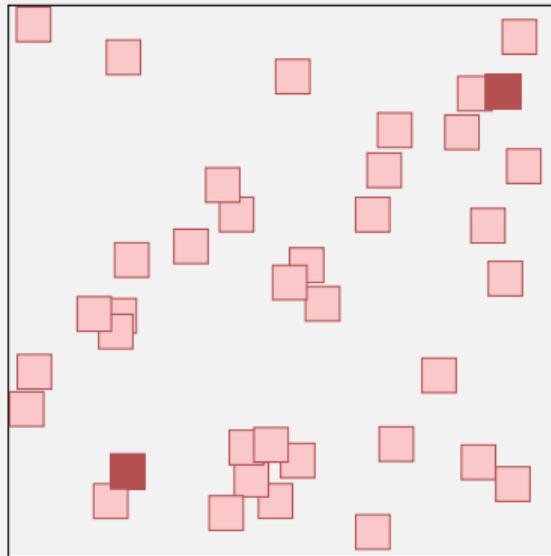
# Idee

- Viele Objekte:  $n^2$  Vergleiche (naiv)
- Verbesserung?
- Offensichtlich: keine Kollisionsdetektion für weit entfernte Objekte nötig.



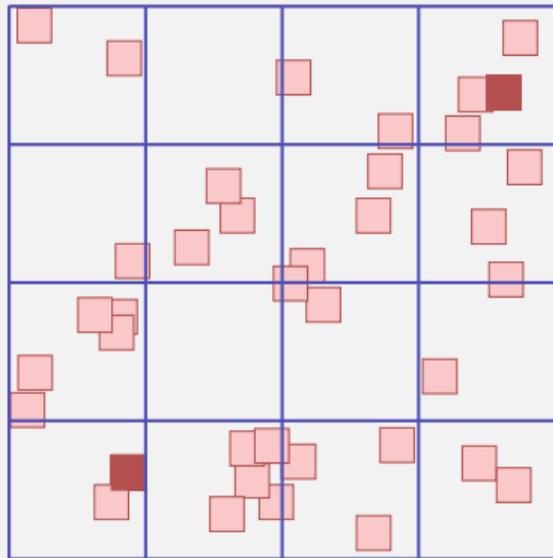
# Idee

- Viele Objekte:  $n^2$  Vergleiche (naiv)
- Verbesserung?
- Offensichtlich: keine Kollisionsdetektion für weit entfernte Objekte nötig.
- Was ist „weit entfernt“?



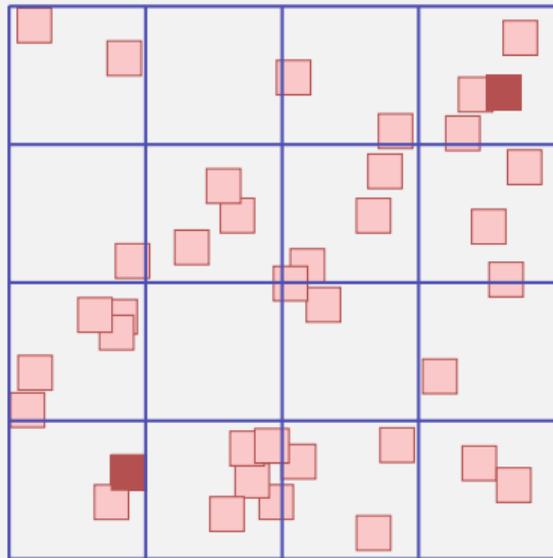
# Idee

- Viele Objekte:  $n^2$  Vergleiche (naiv)
- Verbesserung?
- Offensichtlich: keine Kollisionsdetektion für weit entfernte Objekte nötig.
- Was ist „weit entfernt“?
- Gitter ( $m \times m$ )



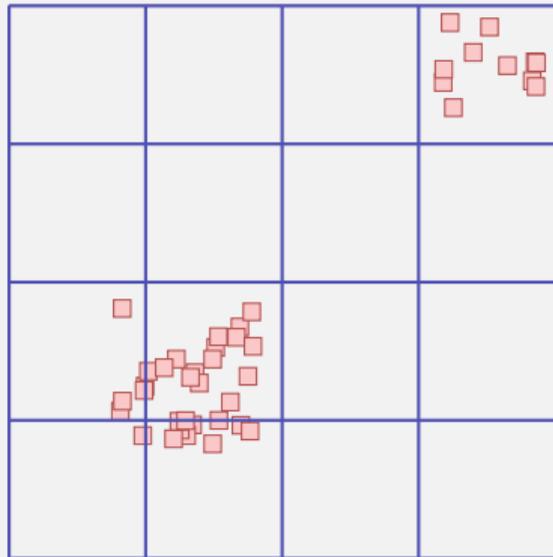
# Idee

- Viele Objekte:  $n^2$  Vergleiche (naiv)
- Verbesserung?
- Offensichtlich: keine Kollisionsdetektion für weit entfernte Objekte nötig.
- Was ist „weit entfernt“?
- Gitter ( $m \times m$ )
- Kollisionsdetektion pro Gitterzelle



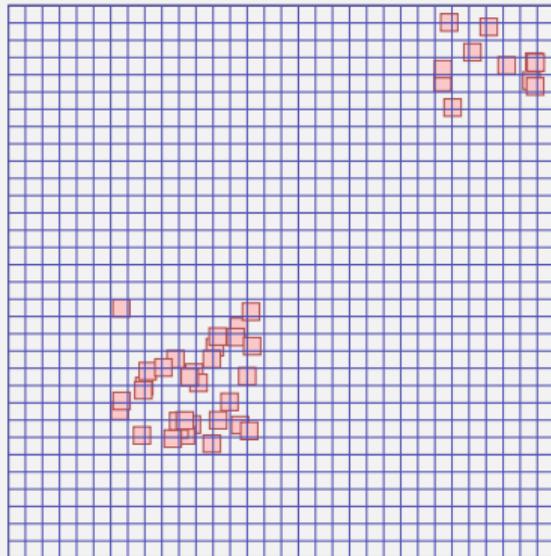
# Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?



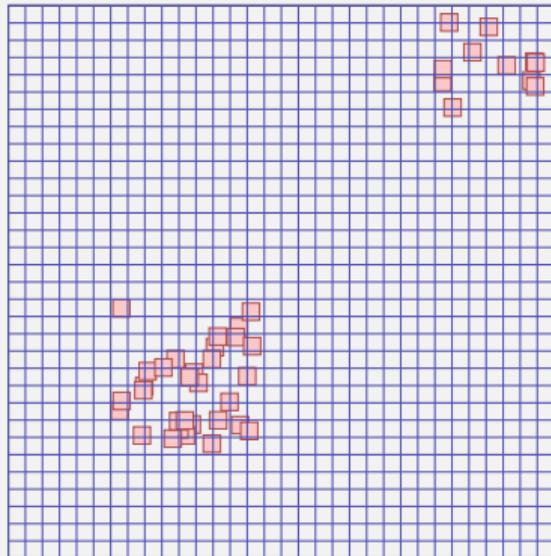
# Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?
- Gitter verfeinern?



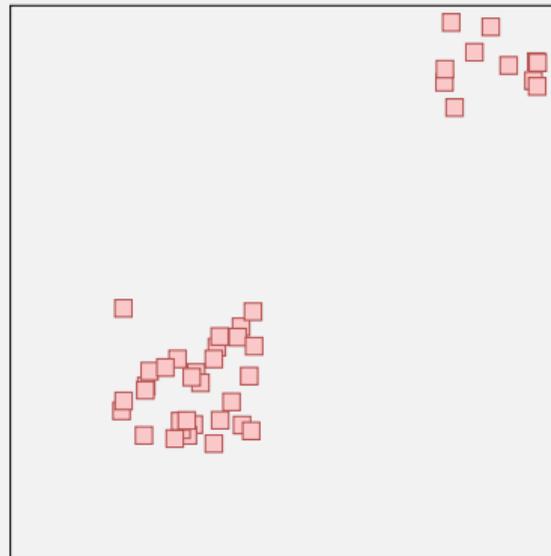
# Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?
- Gitter verfeinern?
- Zu viele Gitterzellen!



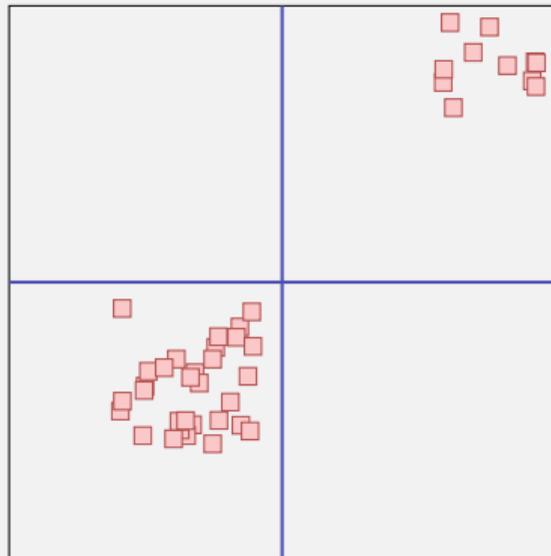
# Adaptive Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?



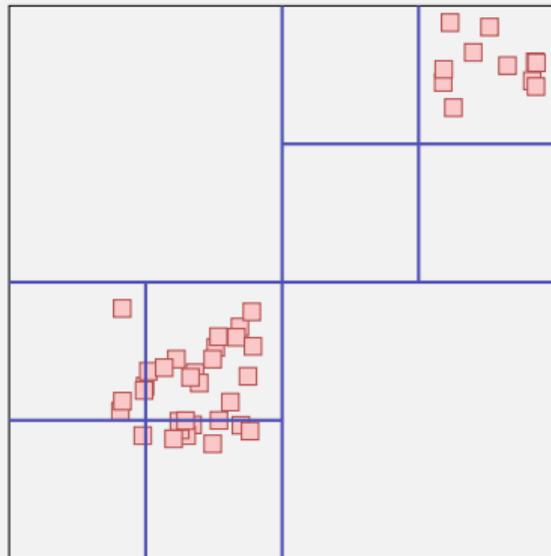
# Adaptive Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?
- Gitter *adaptiv* verfeinern!



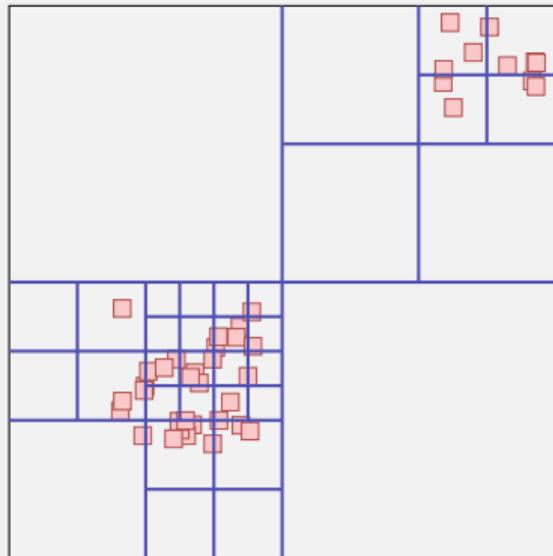
# Adaptive Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?
- Gitter *adaptiv* verfeinern!



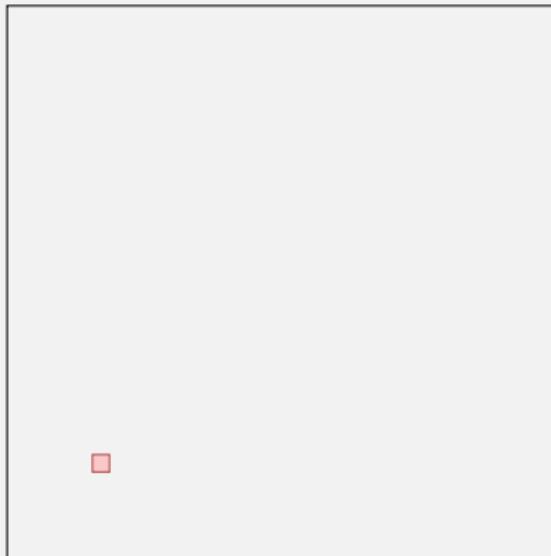
# Adaptive Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?
- Gitter *adaptiv* verfeinern!
- Quadtree!



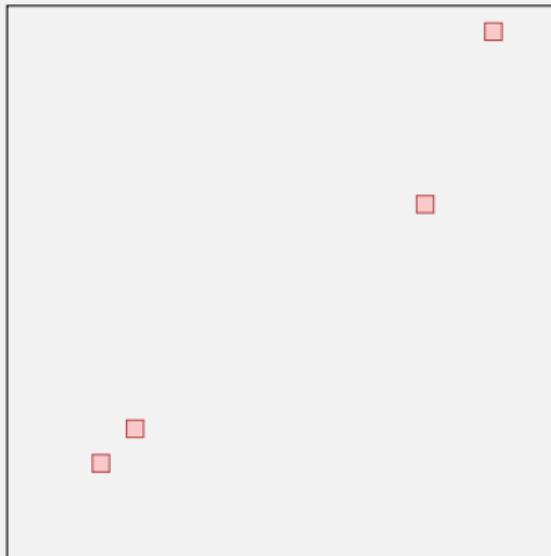
# Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten



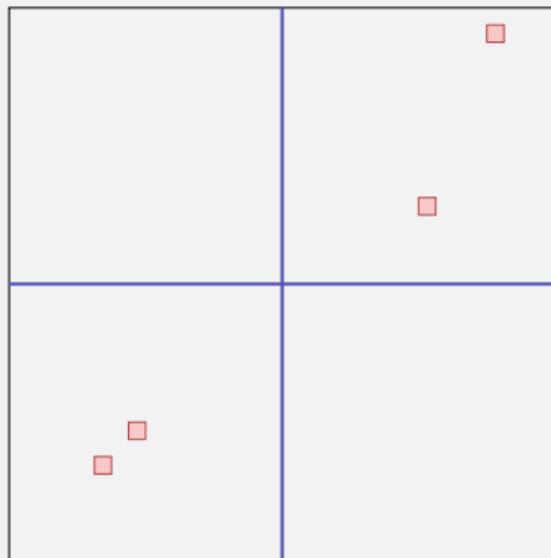
# Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten
- Objekte werden zu dem Knoten hinzugefügt. Wenn in einem Knoten zu viele Objekte sind, wird der Knoten geteilt.



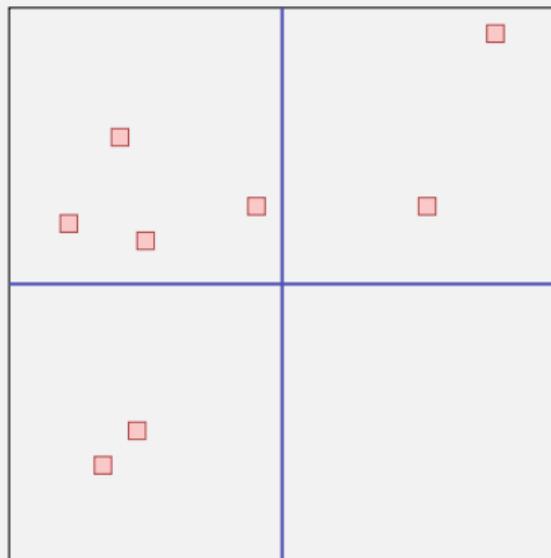
# Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten
- Objekte werden zu dem Knoten hinzugefügt. Wenn in einem Knoten zu viele Objekte sind, wird der Knoten geteilt.



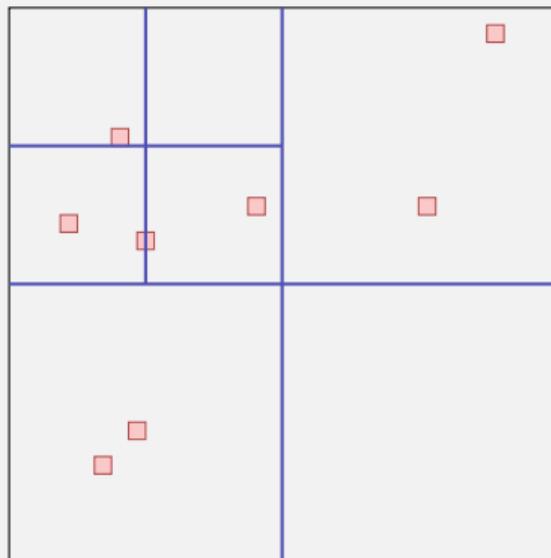
# Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten
- Objekte werden zu dem Knoten hinzugefügt. Wenn in einem Knoten zu viele Objekte sind, wird der Knoten geteilt.



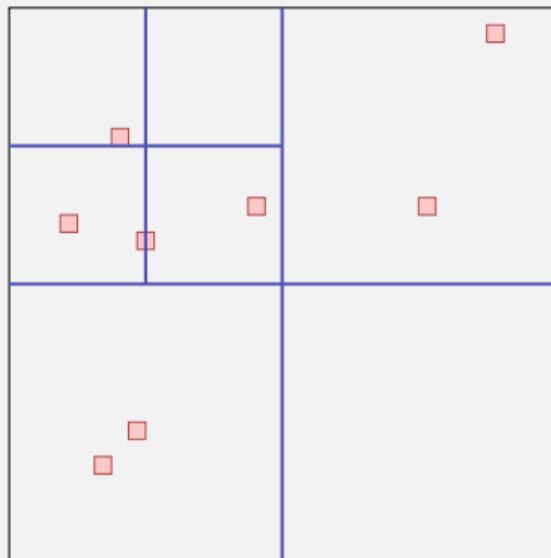
# Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten
- Objekte werden zu dem Knoten hinzugefügt. Wenn in einem Knoten zu viele Objekte sind, wird der Knoten geteilt.



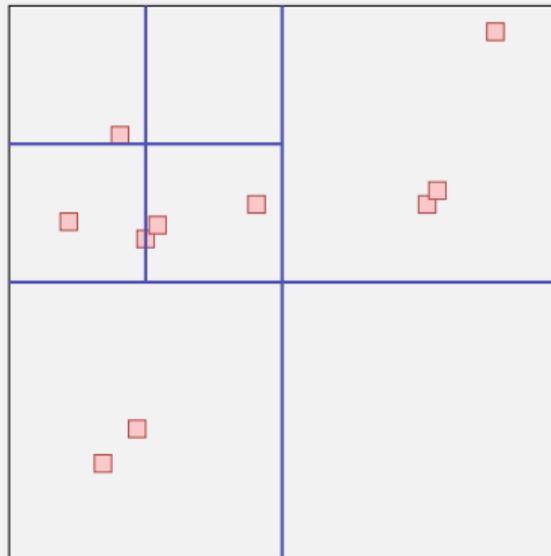
# Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten
- Objekte werden zu dem Knoten hinzugefügt. Wenn in einem Knoten zu viele Objekte sind, wird der Knoten geteilt.
- Objekte, die beim Split auf dem Rand zu liegen kommen, werden im höher gelegenen Knoten belassen.

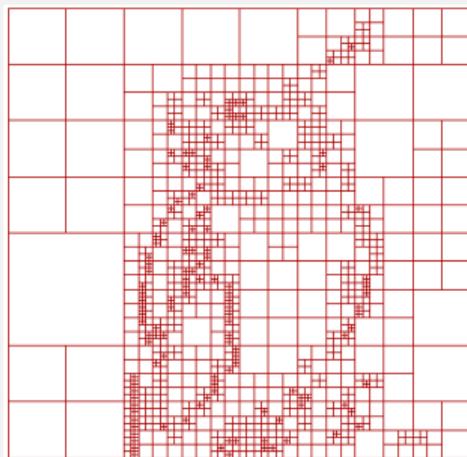


# Algorithmus: Kollisionsdetektion

- Durchlaufe den Quadtree rekursiv. Für jeden Knoten teste die Kollision der enthaltenen Objekte mit Objekten im selben Knoten oder (rekursiv) enthaltenen Knoten.

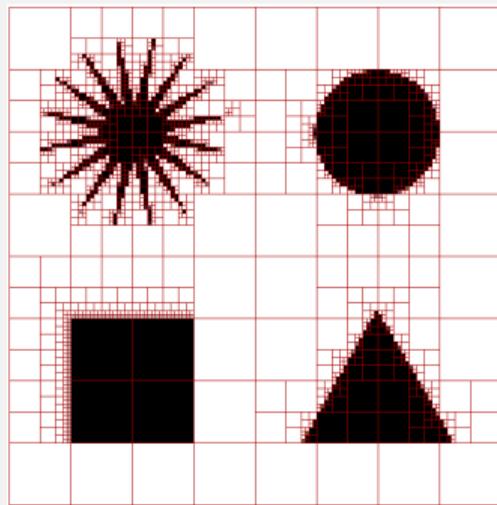
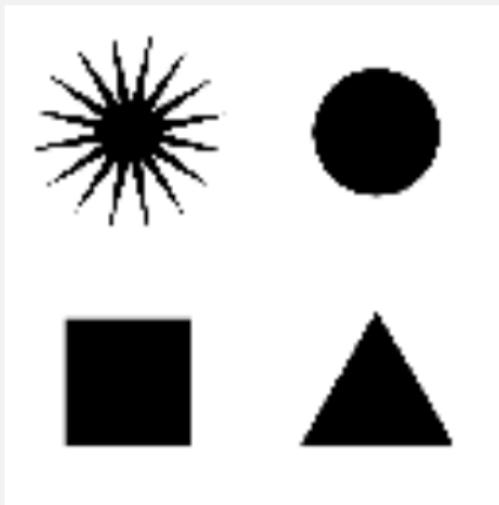


## Beispiel 2: Bildsegmentierung



(Mögliche Anwendungen: Kompression, Entrauschen, Kantendetektion)

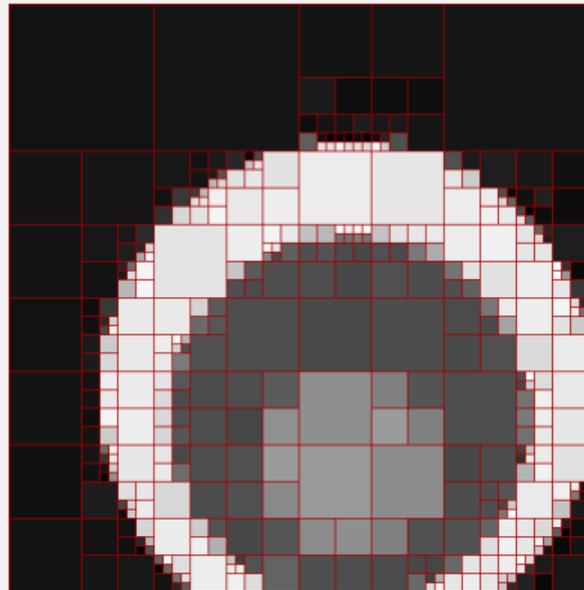
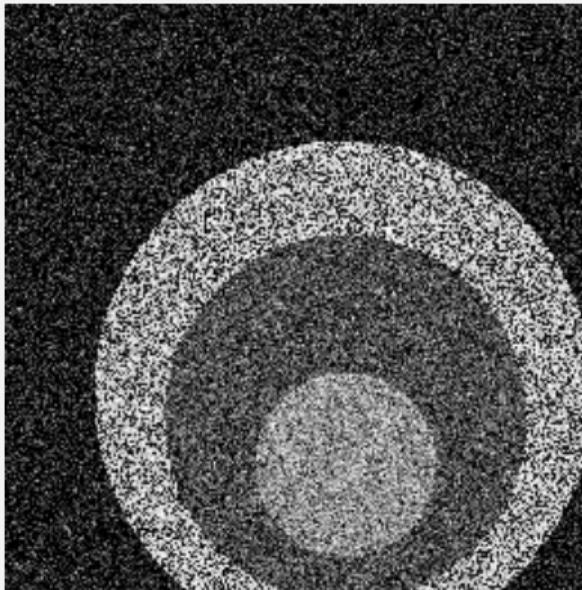
# Quadtree auf Einfarbenbild



Erzeugung des Quadtree ähnlich wie oben: unterteile Knoten rekursiv bis jeder Knoten nur Pixel einer Farbe enthält.

# Quadtree mit Approximation

Wenn mehr als zwei Farbewerte vorhanden sind, wird der Quadtree oft sehr gross.  $\Rightarrow$  Komprimierte Darstellung: *approximiere* das Bild stückweise konstant auf Rechtecken eines Quadtrees.



# Stückweise konstante Approximation

(Graustufen-)Bild  $z \in \mathbb{R}^S$  auf den Pixelindizes  $S$ .<sup>27</sup>

Rechteck  $r \subset S$ .

Ziel: bestimme

$$\arg \min_{x \in r} \sum_{s \in r} (z_s - x)^2$$

---

<sup>27</sup>Wir nehmen an, dass  $S$  ein Quadrat ist mit Seitenlänge  $2^k$  für ein  $k \geq 0$

# Stückweise konstante Approximation

(Graustufen-)Bild  $z \in \mathbb{R}^S$  auf den Pixelindizes  $S$ .<sup>27</sup>

Rechteck  $r \subset S$ .

Ziel: bestimme

$$\arg \min_{x \in \mathbb{R}} \sum_{s \in r} (z_s - x)^2$$

Lösung: das arithmetische Mittel  $\mu_r = \frac{1}{|r|} \sum_{s \in r} z_s$

---

<sup>27</sup>Wir nehmen an, dass  $S$  ein Quadrat ist mit Seitenlänge  $2^k$  für ein  $k \geq 0$

# Zwischenergebnis

Die im Sinne des mittleren quadratischen Fehlers beste Approximation

$$\mu_r = \frac{1}{|r|} \sum_{s \in r} z_s$$

und der dazugehörige Fehler

$$\sum_{s \in r} (z_s - \mu_r)^2 =: \|z_r - \mu_r\|_2^2$$

können nach einer  $\mathcal{O}(|S|)$  Tabellierung schnell berechnet werden:  
Präfixsummen!

# Welcher Quadtree?

## Konflikt

- *Möglichst nahe an den Daten*  $\Rightarrow$  kleine Rechtecke, grosser Quadtree. Extremer Fall: ein Knoten pro Pixel. Approximation = Original
- *Möglichst wenige Knoten*  $\Rightarrow$  Grosse Rechtecke, kleiner Quadtree. Extremfall: ein einziges Rechteck. Approximation = ein Grauwert

# Welcher Quadtree?

Idee: wähle zwischen Datentreue und Komplexität durch Einführung eines Regularisierungsparameters  $\gamma \geq 0$

Wähle Quadtree  $T$  mit Blättern<sup>28</sup>  $L(T)$  so, dass  $T$  folgenden Funktion minimiert

$$H_\gamma(T, z) := \gamma \cdot \underbrace{|L(T)|}_{\text{Anzahl Blätter}} + \underbrace{\sum_{r \in L(T)} \|z_r - \mu_r\|_2^2}_{\text{Summierter Approximationsfehler aller Blätter}} .$$

---

<sup>28</sup>hier: Blatt = Knoten mit Nullkindern,

# Regularisierung

Sei  $T$  ein Quadtree über einem Rechteck  $S_T$  und seien  $T_{ll}, T_{lr}, T_{ul}, T_{ur}$  vier mögliche Unterbäume und

$$\hat{H}_\gamma(T, z) := \min_T \gamma \cdot |L(T)| + \sum_{r \in L(T)} \|z_r - \mu_r\|_2^2$$

Extremfälle:

$\gamma = 0 \Rightarrow$  Originaldaten;

$\gamma \rightarrow \infty \Rightarrow$  ein Rechteck

# Beobachtung: Rekursion

- Wenn der (Sub-)Quadtree  $T$  nur ein Pixel hat, so kann nicht aufgeteilt werden und es gilt

$$\hat{H}_\gamma(T, z) = \gamma$$

- Andernfalls seien

$$M_1 := \gamma + \|z_{S_T} - \mu_{S_T}\|_2^2$$

$$M_2 := \hat{H}_\gamma(T_{ll}, z) + \hat{H}_\gamma(T_{lr}, z) + \hat{H}_\gamma(T_{ul}, z) + \hat{H}_\gamma(T_{ur}, z)$$

Dann

$$\hat{H}_\gamma(T, z) = \min\left\{\underbrace{M_1(T, \gamma, z)}_{\text{kein Split}}, \underbrace{M_2(T, \gamma, z)}_{\text{Split}}\right\}$$

# Algorithmus: Minimize( $z, r, \gamma$ )

**Input** : Bilddaten  $z \in \mathbb{R}^S$ , Rechteck  $r \subset S$ , Regularisierung  $\gamma > 0$

**Output** :  $\min_T \gamma |L(T)| + \|z - \mu_{L(T)}\|_2^2$

**if**  $|r| = 0$  **then return** 0

$m \leftarrow \gamma + \sum_{s \in r} (z_s - \mu_r)^2$

**if**  $|r| > 1$  **then**

    Split  $r$  into  $r_{ll}, r_{lr}, r_{ul}, r_{ur}$

$m_1 \leftarrow \text{Minimize}(z, r_{ll}, \gamma)$ ;  $m_2 \leftarrow \text{Minimize}(z, r_{lr}, \gamma)$

$m_3 \leftarrow \text{Minimize}(z, r_{ul}, \gamma)$ ;  $m_4 \leftarrow \text{Minimize}(z, r_{ur}, \gamma)$

$m' \leftarrow m_1 + m_2 + m_3 + m_4$

**else**

$m' \leftarrow \infty$

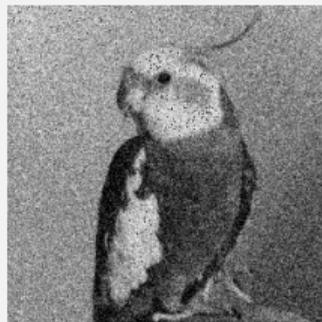
**if**  $m' < m$  **then**  $m \leftarrow m'$

**return**  $m$

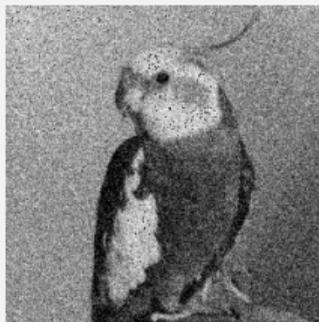
# Analyse

Der Minimierungsalgorithmus über dyadische Partitionen (Quadtree) benötigt  $\mathcal{O}(|S| \log |S|)$  Schritte.

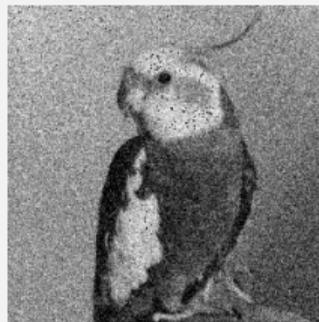
# Anwendung: Entrauschen (zusätzlich mit Wedgelets)



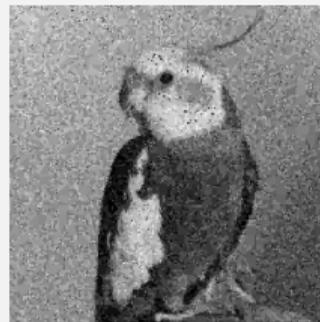
noised



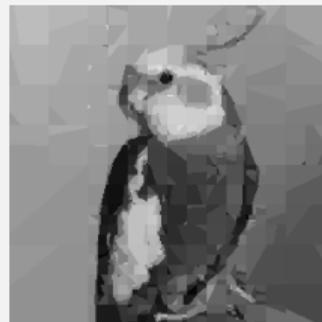
$\gamma = 0.003$



$\gamma = 0.01$



$\gamma = 0.03$



$\gamma = 0.1$



$\gamma = 0.3$



$\gamma = 1$

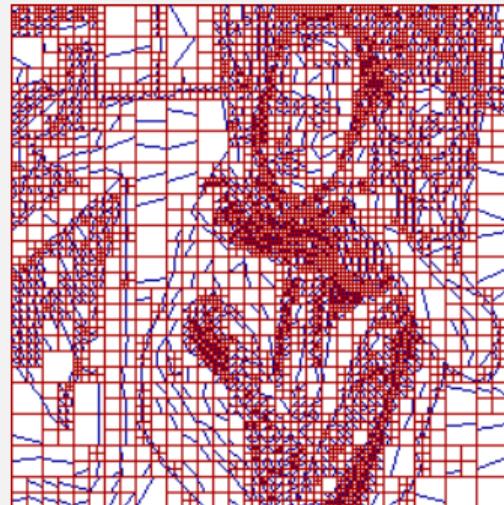


$\gamma = 3$



$\gamma = 10$

# Erweiterungen: Affine Regression + Wedgelets



# Andere Ideen

kein Quadtree: hierarchisch-eindimensionales Modell (benötigt Dynamic Programming)

