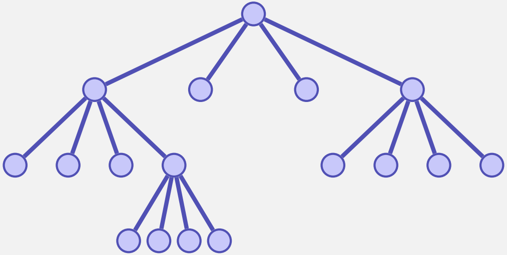
18. Quadtrees

Quadtrees, Collision Detection, Image Segmentation

Quadtree

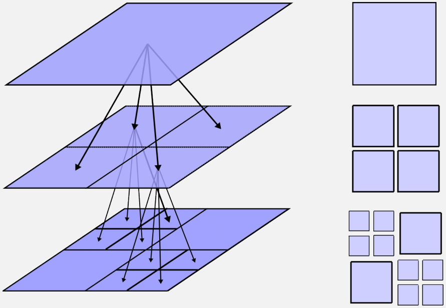
A quad tree is a tree of order 4.



... and as such it is not particularly interesting except when it is used for ...

Quadtree - Interpretation und Nutzen

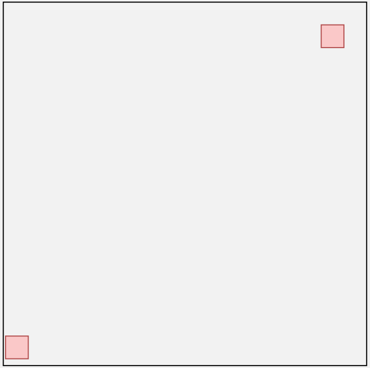
Separation of a two-dimensional range into 4 equally sized parts.



[analogously in three dimensions with an *octtree* (tree of order 8)]

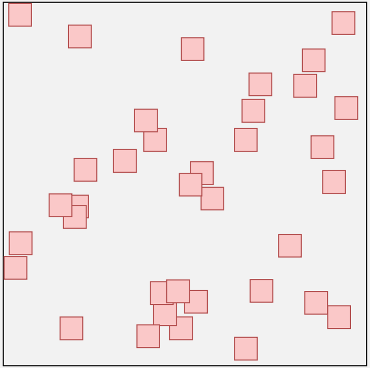
Example 1: Collision Detection

- Objects in the 2D-plane, e.g. particle simulation on the screen.
- Goal: collision detection



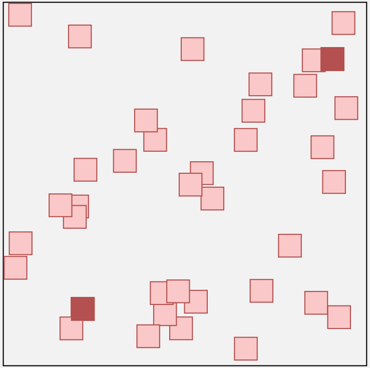
Idea

- Many objects: n^2 detections (naively)
- Improvement?



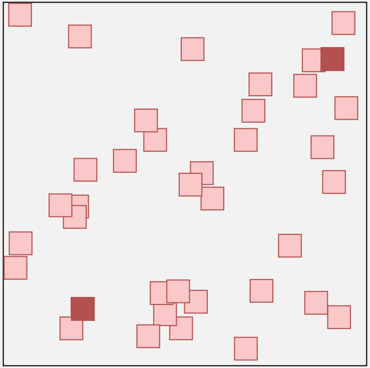
Idea

- Many objects: n^2 detections (naively)
- Improvement?
- Obviously: collision detection not required for objects far away from each other



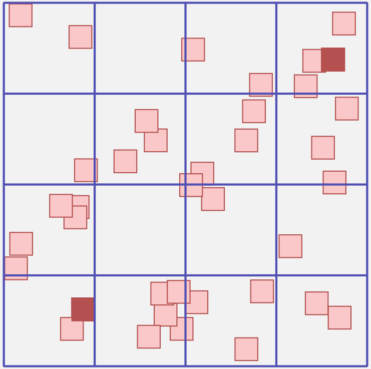
Idea

- Many objects: n^2 detections (naively)
- Improvement?
- Obviously: collision detection not required for objects far away from each other
- What is „far away”?



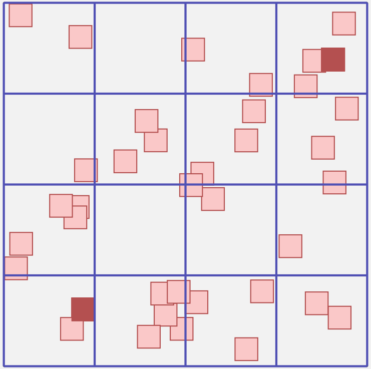
Idea

- Many objects: n^2 detections (naively)
- Improvement?
- Obviously: collision detection not required for objects far away from each other
- What is „far away”?
- Grid ($m \times m$)



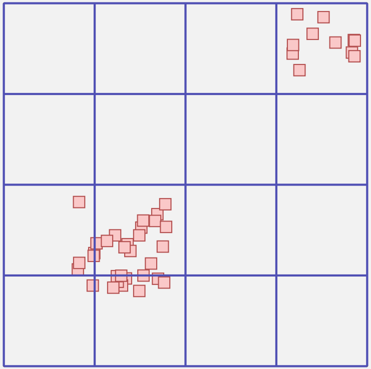
Idea

- Many objects: n^2 detections (naively)
- Improvement?
- Obviously: collision detection not required for objects far away from each other
- What is „far away”?
- Grid $(m \times m)$
- Collision detection per grid cell



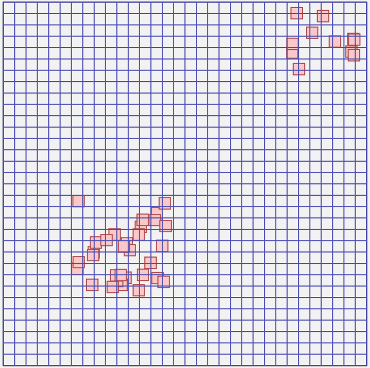
Grids

- A grid often helps, but not always
- Improvement?



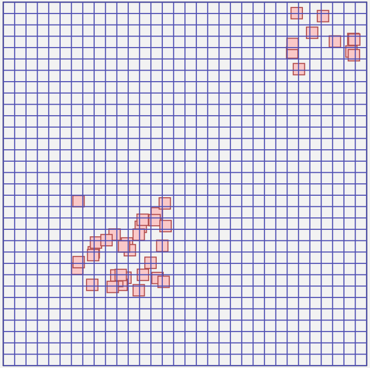
Grids

- A grid often helps, but not always
- Improvement?
- More finegrained grid?



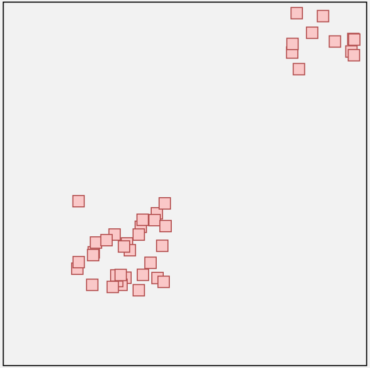
Grids

- A grid often helps, but not always
- Improvement?
- More finegrained grid?
- Too many grid cells!



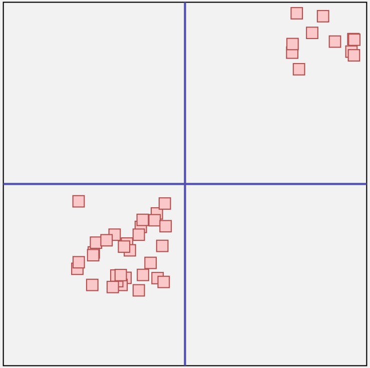
Adaptive Grids

- A grid often helps, but not always
- Improvement?



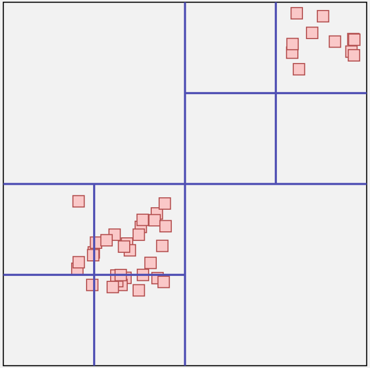
Adaptive Grids

- A grid often helps, but not always
- Improvement?
- *Adaptively* refine grid



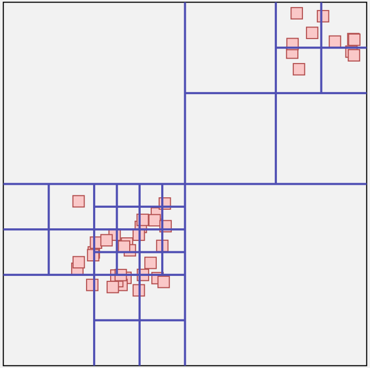
Adaptive Grids

- A grid often helps, but not always
- Improvement?
- *Adaptively* refine grid



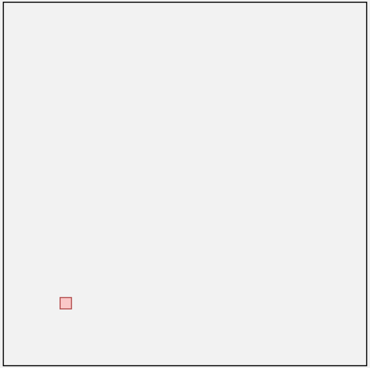
Adaptive Grids

- A grid often helps, but not always
- Improvement?
- *Adaptively* refine grid
- Quadtree!



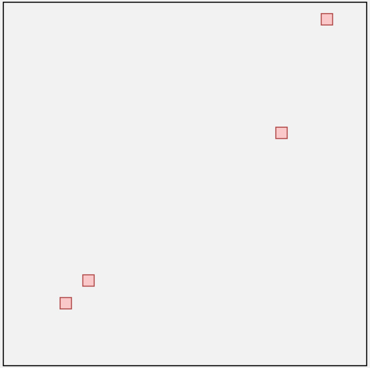
Algorithm: Insertion

- Quadtree starts with a single node



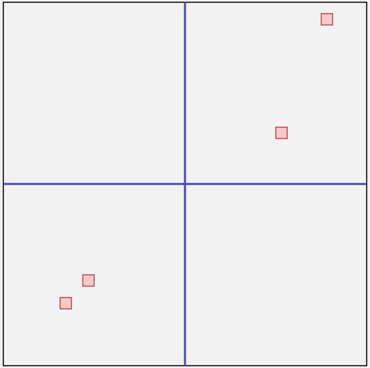
Algorithm: Insertion

- Quadtree starts with a single node
- Objects are added to the node. When a node contains too many objects, the node is split.



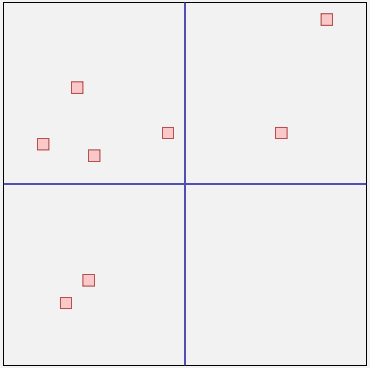
Algorithm: Insertion

- Quadtree starts with a single node
- Objects are added to the node. When a node contains too many objects, the node is split.



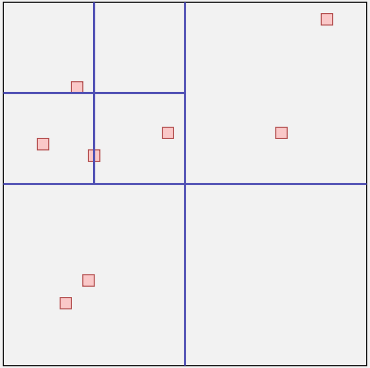
Algorithm: Insertion

- Quadtree starts with a single node
- Objects are added to the node. When a node contains too many objects, the node is split.



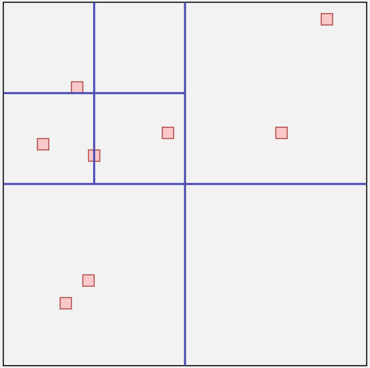
Algorithm: Insertion

- Quadtree starts with a single node
- Objects are added to the node. When a node contains too many objects, the node is split.



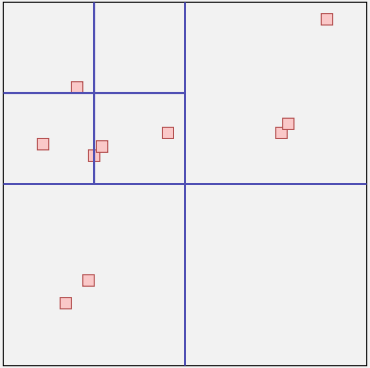
Algorithm: Insertion

- Quadtree starts with a single node
- Objects are added to the node. When a node contains too many objects, the node is split.
- Objects that are on the boundary of the quadtree remain in the higher level node.

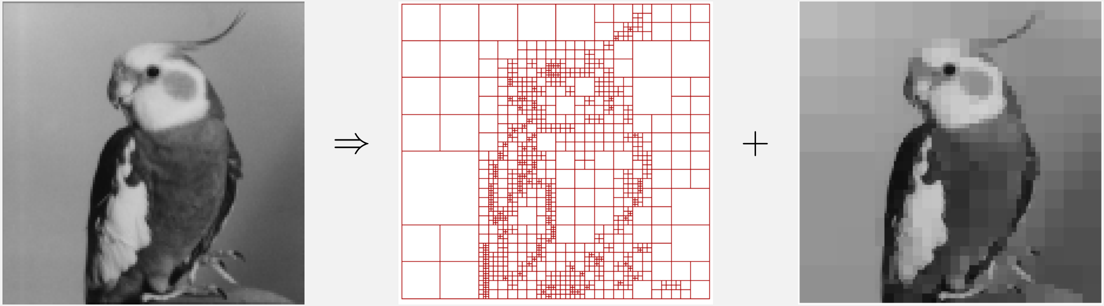


Algorithm: Collision Detection

- Run through the quadtree in a recursive way. For each node test collision with all objects contained in the same or (recursively) contained nodes.

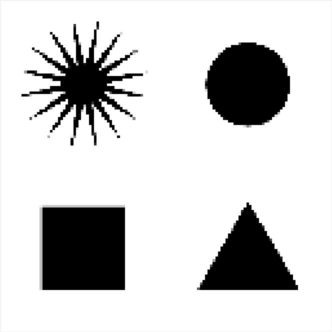


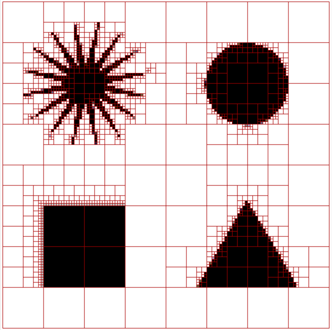
Example 2: Image Segmentation



(Possible applications: compression, denoising, edge detection)

Quadtree on Monochrome Bitmap

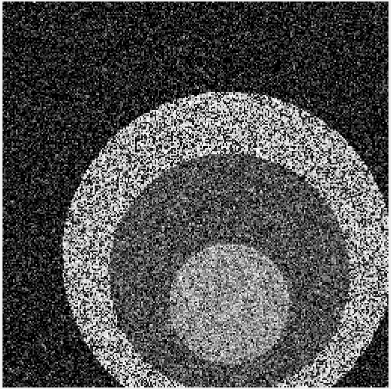


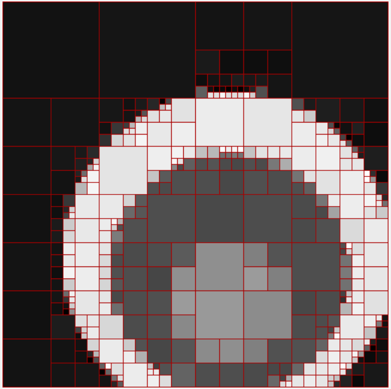


Similar procedure to generate the quadtree: split nodes recursively until each node only contains pixels of the same color.

Quadtree with Approximation

When there are more than two color values, the quadtree can get very large. ⇒ Compressed representation: *approximate* the image piecewise constant on the rectangles of a quadtree.





Piecewise Constant Approximation

(Grey-value) Image $z \in \mathbb{R}^S$ on pixel indices S. [36]

Rectangle $r \subset S$.

Goal: determine

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

[36] we assume that S is a square with side length 2^k for some $k \geq 0$

Piecewise Constant Approximation

(Grey-value) Image $z \in \mathbb{R}^S$ on pixel indices S. [36]

Rectangle $r \subset S$.

Goal: determine

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

Solution: the arithmetic mean $\mu_r = \frac{1}{|r|} \sum_{s \in r} z_s$

[36] we assume that S is a square with side length 2^k for some $k \geq 0$

Intermediate Result

The (w.r.t. mean squared error) best approximation

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

and the corresponding error

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

can be computed quickly after a $\mathcal{O}(|S|)$ tabulation: prefix sums!

Which Quadtree?

Conflict

- *As close as possible to the data* ⇒ small rectangles, large quadtree . Extreme case: one node per pixel. Approximation = original
- *Small amount of nodes* ⇒ large rectangles, small quadtree Extreme case: a single rectangle. Approximation = a single grey value.

Which Quadtree?

Idea: choose between data fidelity and complexity with a regularisation parameter $\gamma \geq 0$

Choose quadtree T with leaves[37] $L(T)$ such that it minimizes the following function

$$H_\gamma(T, z) := \gamma \cdot \underbrace{|L(T)|}_{\text{Number of Leaves}} + \underbrace{\sum_{r \in L(T)} \|z_r - \mu_r\|_2^2}_{\text{Cummulative approximation error of all leaves}} .$$

[37] here: leaf: node with null-children

Regularisation

Let T be a quadtree over a rectangle S_T and let $T_{ll}, T_{lr}, T_{ul}, T_{ur}$ be the four possible sub-trees and

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

Extreme cases:
$\gamma = 0 \Rightarrow$ original data;
$\gamma \to \infty \Rightarrow$ a single rectangle

Observation: Recursion

- If the (sub-)quadtree T represents only one pixel, then it cannot be split and it holds that

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

- Let, otherwise,

$$\begin{aligned} M_1 &:= \gamma + \|z_{S_T} - \mu_{S_T}\|_2^2 \\ M_2 &:= \widehat{H}_\gamma(T_{ll}, z) + \widehat{H}_\gamma(T_{lr}, z) + \widehat{H}_\gamma(T_{ul}, z) + \widehat{H}_\gamma(T_{ur}, z) \end{aligned}$$

then

$$\widehat{H}_\gamma(T, z) = \min\{\underbrace{M_1(T, \gamma, z)}_{\text{no split}}, \underbrace{M_2(T, \gamma, z)}_{\text{split}}\}$$

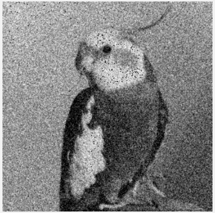
Algorithmus: Minimize(z,r,γ)

```
Input: Image data $z \in \mathbb{R}^S$, rectangle $r \subset S$, regularization $\gamma > 0$
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 $r_{ll}$,$r_{lr}$,$r_{ul}$,$r_{ur}$
    $m_1 \leftarrow \mathrm{Minimize}(z, r_{ll}, \gamma)$; $m_2 \leftarrow \mathrm{Minimize}(z, r_{lr}, \gamma)$
    $m_3 \leftarrow \mathrm{Minimize}(z, r_{ul}, \gamma)$; $m_4 \leftarrow \mathrm{Minimize}(z, r_{ur}, \gamma)$
    $m' \leftarrow m_1 + m_2 + m_3 + m_4$
else
    $m' \leftarrow \infty$
if $m' < m$ then $m \leftarrow m'$
return $m$
```

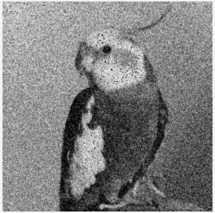
Analysis

The minimization algorithm over dyadic partitions (quadtrees) takes $\mathcal{O}(|S| \log |S|)$ steps.

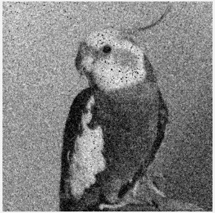
Application: Denoising (with addditional Wedgelets)



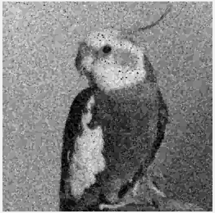
noised



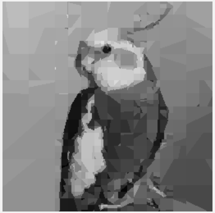
$\gamma = 0.003$



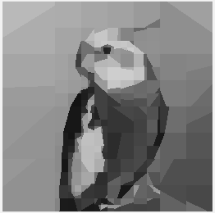
$\gamma = 0.01$



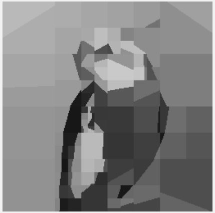
$\gamma = 0.03$



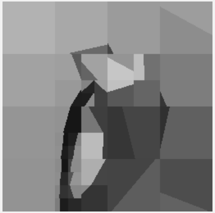
$\gamma = 0.1$



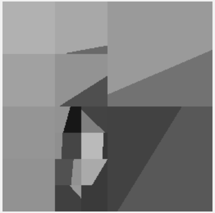
$\gamma = 0.3$



$\gamma = 1$

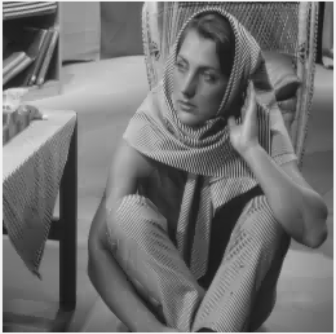


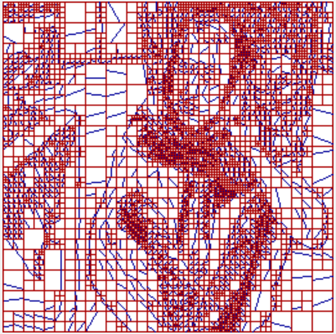
$\gamma = 3$



$\gamma = 10$

Extensions: Affine Regression + Wedgelets





Other ideas

no quadtree: hierarchical one-dimensional modell (requires dynamic programming)