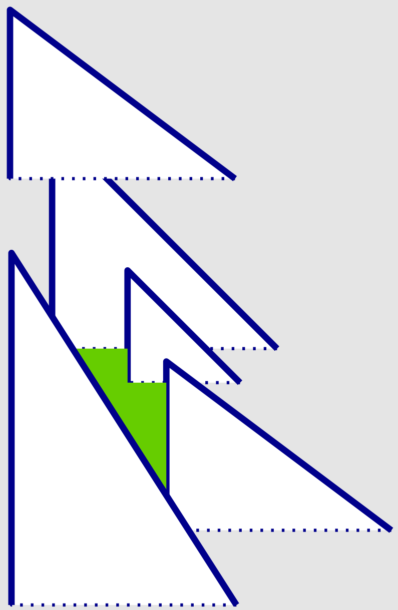
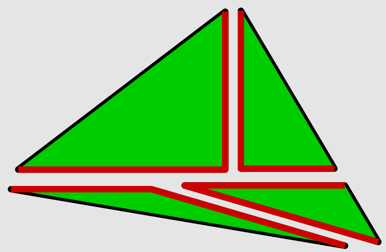
TU/e

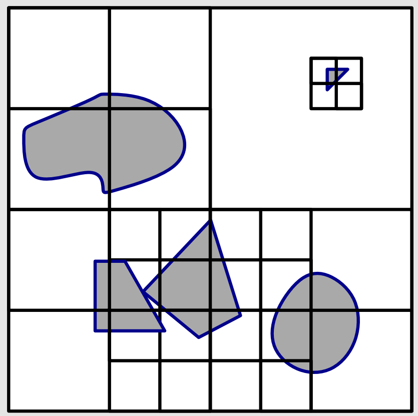
Computational Geometry
for
Fat Objects and Low Density Scenes

Mark de Berg

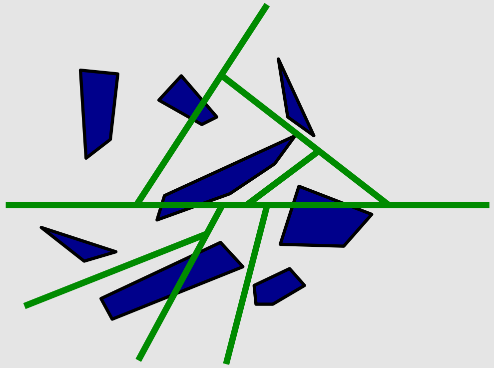
TU Eindhoven



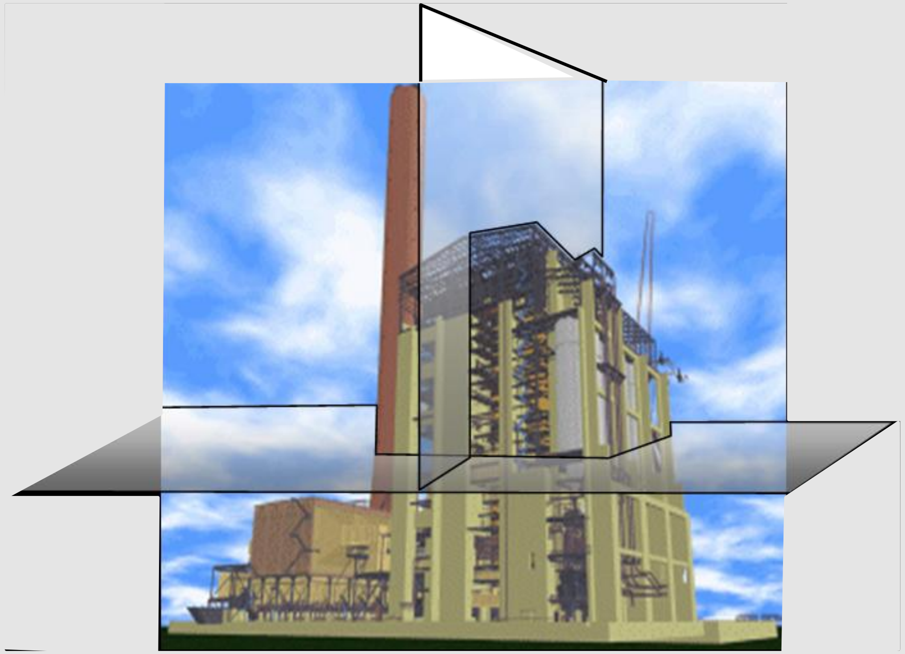




binary space partitions (BSPs): recursive subdivision of space until in each region there is only one object (or: constant number of objects)

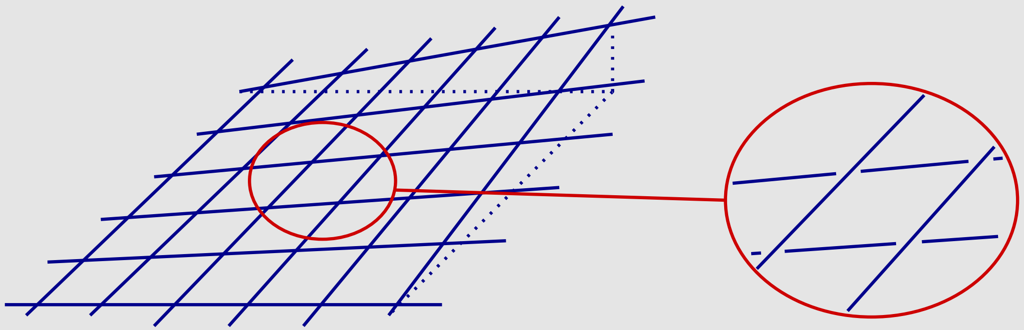


Practice: BSPs are used



12,748,510 triangles

Theory: worst-case size $\Theta(n^2)$ (Paterson-Yao '90)



- analysis not only as function of input size n,
  but also as function of certain geometry-describing parameters

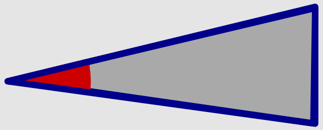
- leads (hopefully) to

  – better prediction of when algorithms are efficient in practice

  – simpler algorithms, designed with practical inputs in mind

Shape parameter for individual objects:fatness

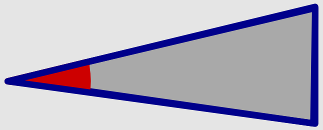
fat triangles



δ-fat triangle:
minimum angle at least δ

Shape parameter for individual objects:fatness

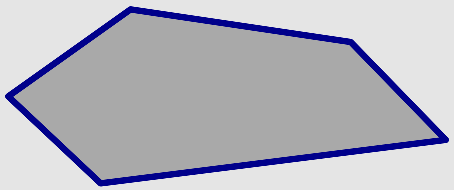
fat triangles



δ-fat triangle:
minimum angle at least δ

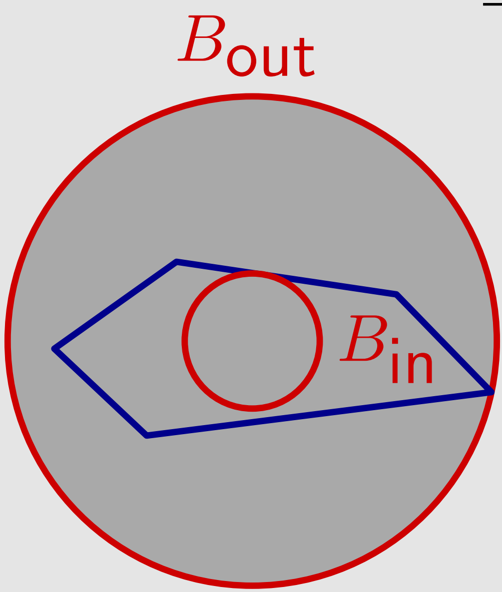
β-fat convex objects

definition 1



$$\frac{\mathrm{vol}(o)}{\mathrm{diam}^2(o)} \geq \beta$$

definition 2

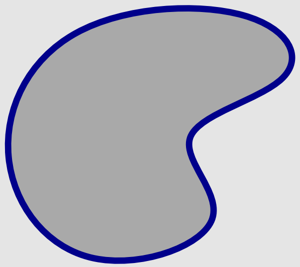


There are concentric balls $B_{\mathrm{in}} \subset o \subset B_{\mathrm{out}}$ such that

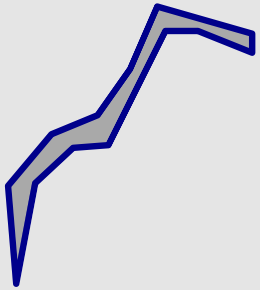
$$\frac{\mathrm{diam}(B_{\mathrm{in}})}{\mathrm{diam}(B_{\mathrm{out}})} \geq \beta$$

Definitions: fat objects (2)

Non-convex objects

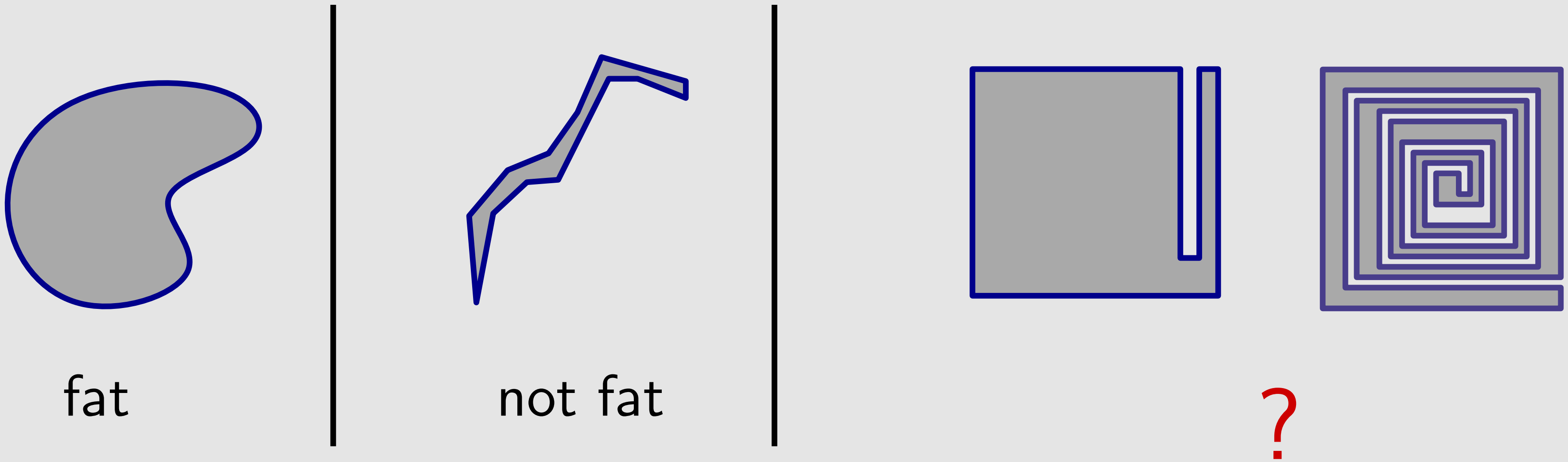


fat

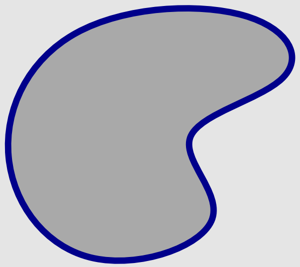


not fat

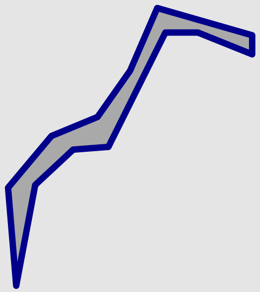
Non-convex objects



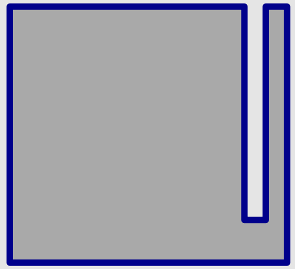
Non-convex objects

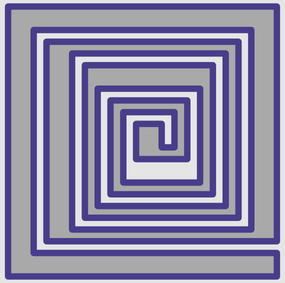


fat



not fat





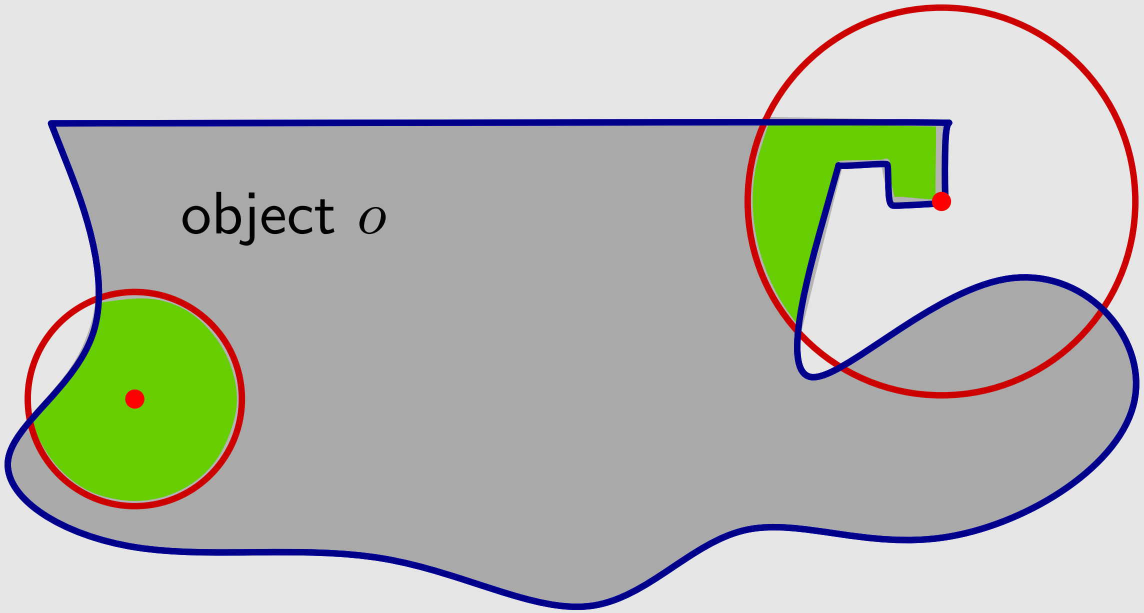
?

fat objects should not have skinny parts

$\Longrightarrow$

these objects are not fat

(locally) γ-fat object

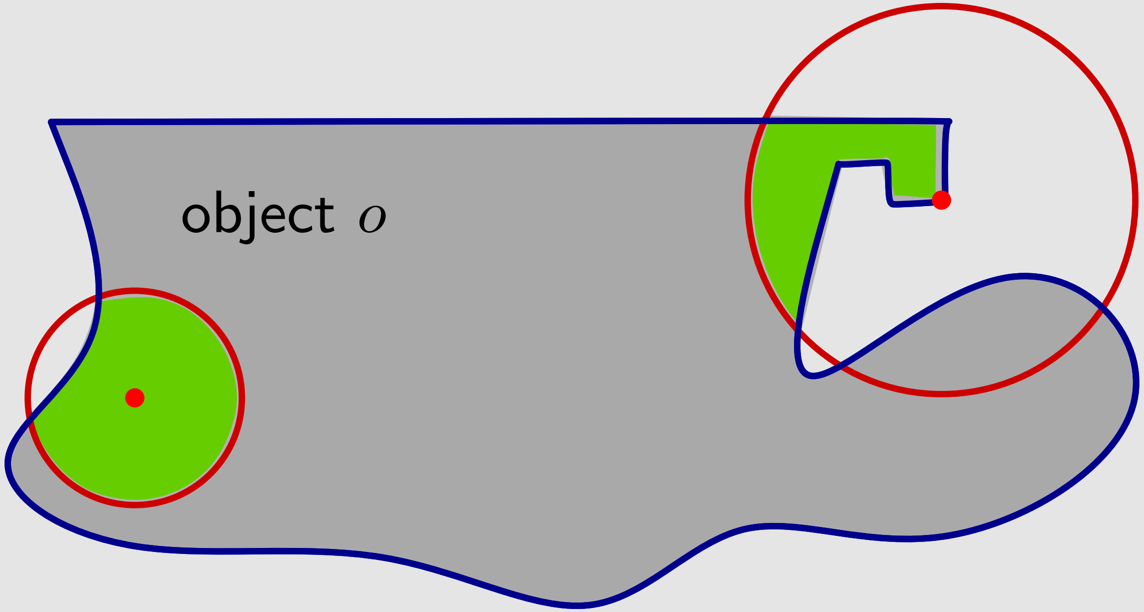


For any ball B with center in o and not containing o in its interior, we have:

$$\text{vol}(\underbrace{B \sqcap o}_{\boxed{\text{component of } B \cap o \text{ containing center}}}) \geq \gamma \cdot \text{vol}(B)$$

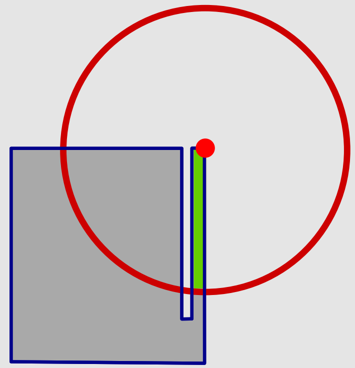
Definitions: fat objects (3)

(locally) γ-fat object



For any ball B with center in o and not containing o in its interior, we have:

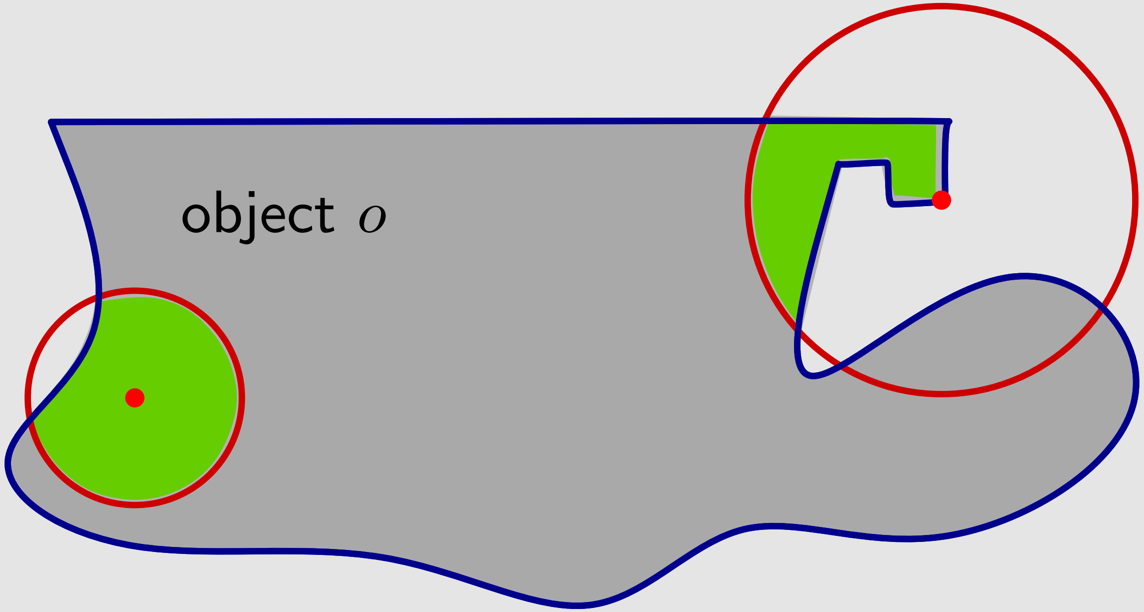
$$\mathrm{vol}(\underbrace{B \sqcap o}_{\text{component of } B \cap o \text{ containing center}}) \geq \gamma \cdot \mathrm{vol}(B)$$



not locally fat

Definitions: fat objects (3)

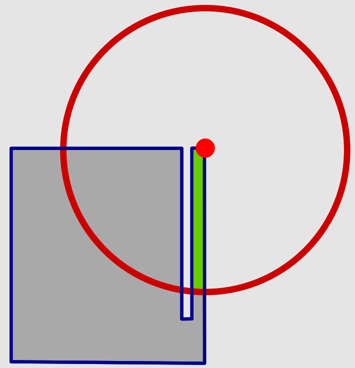
(locally) γ-fat object



For any ball B with center in o and not containing o in its interior, we have:

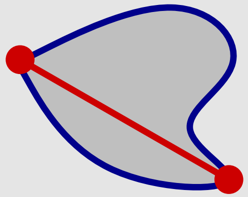
$$\mathrm{vol}(\underbrace{B \sqcap o}_{\text{component of } B \cap o \text{ containing center}}) \geq \gamma \cdot \mathrm{vol}(B)$$

fatness of o: largest γ such that o is γ-fat



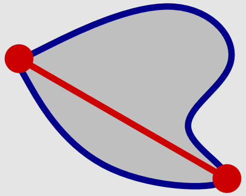
not locally fat

- distribution parameter for sets of objects: density



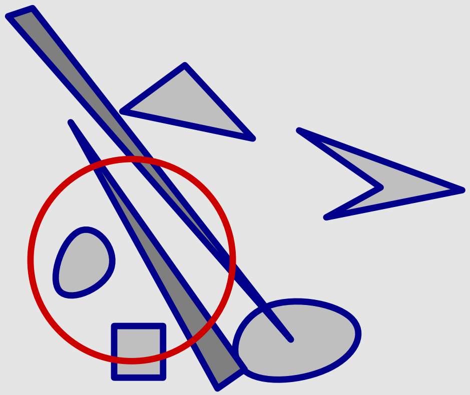
$\text{diam}(o) :=$ diameter of o

- distribution parameter for sets of objects: density



diam(o) := diameter of o

van der Stappen '94



density of set S of objects:

minimum λ such that for any ball b :

$\#\{o \in S : o \text{ intersects } b, \text{diam}(o) \geq \text{diam}(b)\ \}| \leq \lambda$

Hope: density will in practice be a (small) constant.

density of set S of objects:

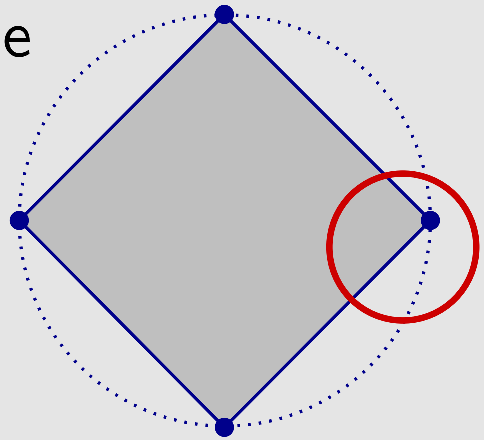
minimum λ such that for any ball b :

$$\#\{o \in S : o \text{ intersects } b,\ \text{diam}(o) \geq \text{diam}(b)\ \}| \leq \lambda$$

Does it makes sense?

object $=$ facet of polytope

refining the object does not increase the density significantly



Hope: density will in practice be a (small) constant.

density of set S of objects:

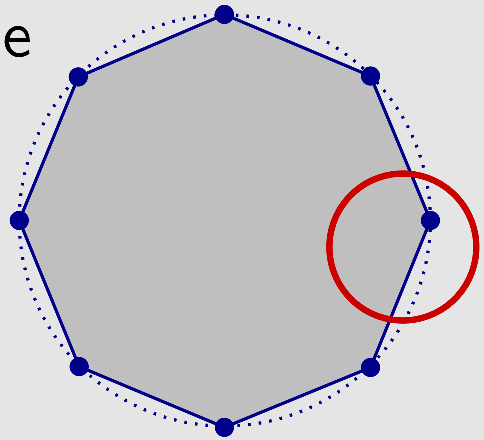
minimum λ such that for any ball b :

$$\#\{o \in S : o \text{ intersects } b,\ \text{diam}(o) \geq \text{diam}(b)\ \}| \leq \lambda$$

Does it makes sense?

object = facet of polytope

refining the object does not
increase the density significantly



Hope: density will in practice be a (small) constant.

density of set S of objects:

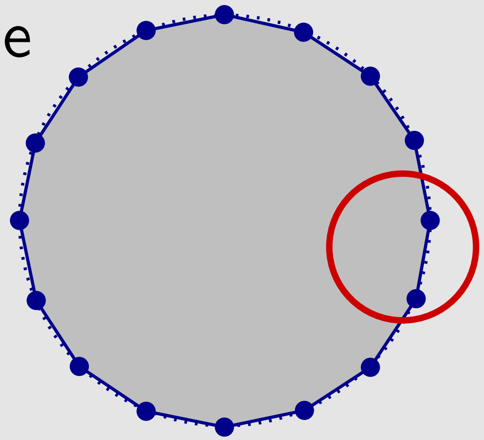
minimum λ such that for any ball b :

$$\#\{o \in S : o \text{ intersects } b, \text{diam}(o) \geq \text{diam}(b) \ \}| \leq \lambda$$

Does it makes sense?

object $=$ facet of polytope

refining the object does not
increase the density significantly



geometry-describing parameters: density (cont'd)

Hope: density will in practice be a (small) constant.

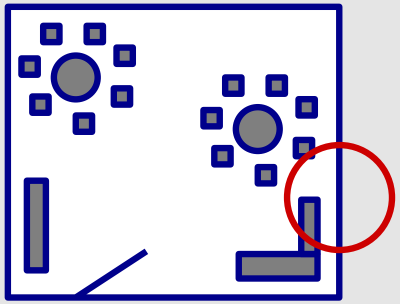
density of set S of objects:

minimum λ such that for any ball b :

$$\#\{o \in S : o \text{ intersects } b, \text{diam}(o) \geq \text{diam}(b)\ \}| \leq \lambda$$

Does it makes sense?

polyhedral model of a building



increasing number of rooms does not increase density significantly

Hope: density will in practice be a (small) constant.

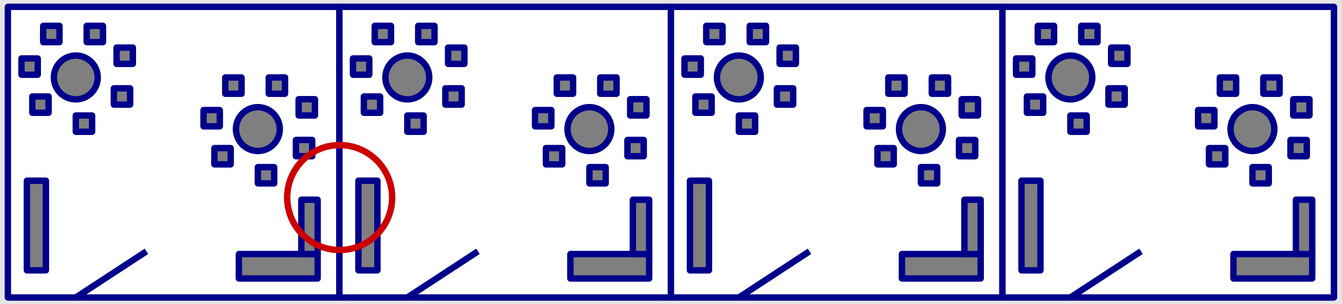
density of set S of objects:

minimum λ such that for any ball b :

$\#\{o \in S : o$ intersects b, diam$(o) \geq$ diam(b) $\}| \leq \lambda$

Does it makes sense?

polyhedral model of a building



increasing number of rooms does not increase density significantly

van der Stappen '94

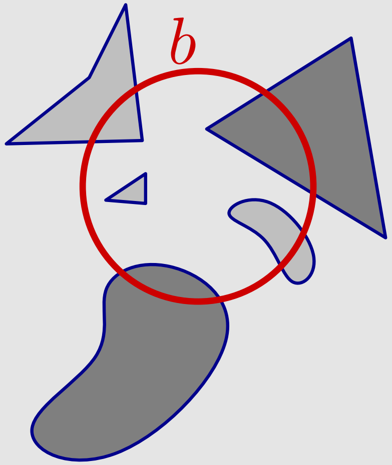
A set S of disjoint β-fat objects has density $O(1/\beta)$.

van der Stappen '94

A set S of disjoint β-fat objects has density $O(1/\beta)$.

Proof: we must show that for any ball b:

$$\#\{o : o \text{ intersects } b,\ \text{diam}(o) \geq \text{diam}(b)\} = O(1/\beta)$$

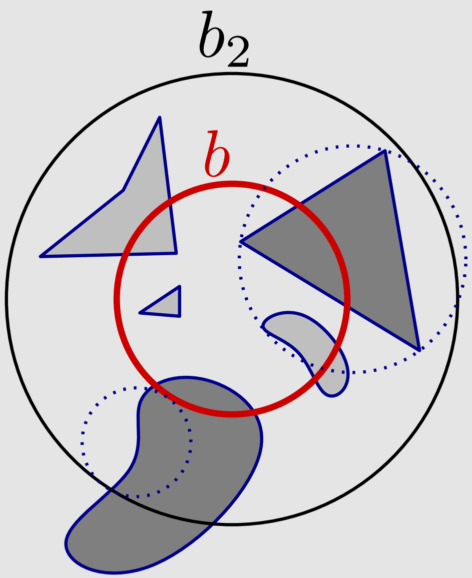


van der Stappen '94

A set S of disjoint β-fat objects has density $O(1/\beta)$.

Proof: we must show that for any ball b:

$$\#\{o : o \text{ intersects } b,\ \text{diam}(o) \geq \text{diam}(b)\} = O(1/\beta)$$



- each object o that we must count covers a fraction $\Omega(\beta)$ of the ball b_2
- objects are disjoint

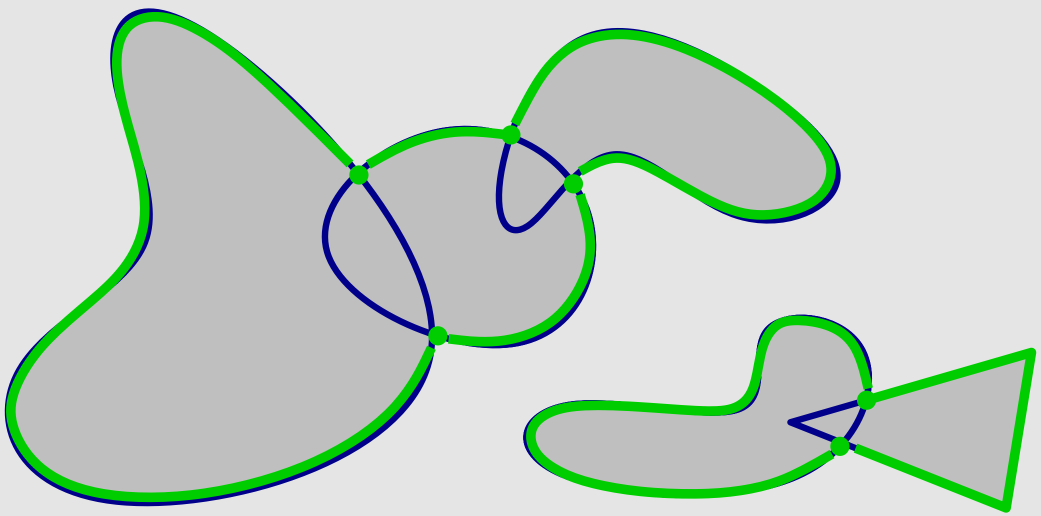
■

S: n locally γ-fat objects

E: "edges" of boundary of union of S

dB '05

The set E has density $O(1/\gamma)$.

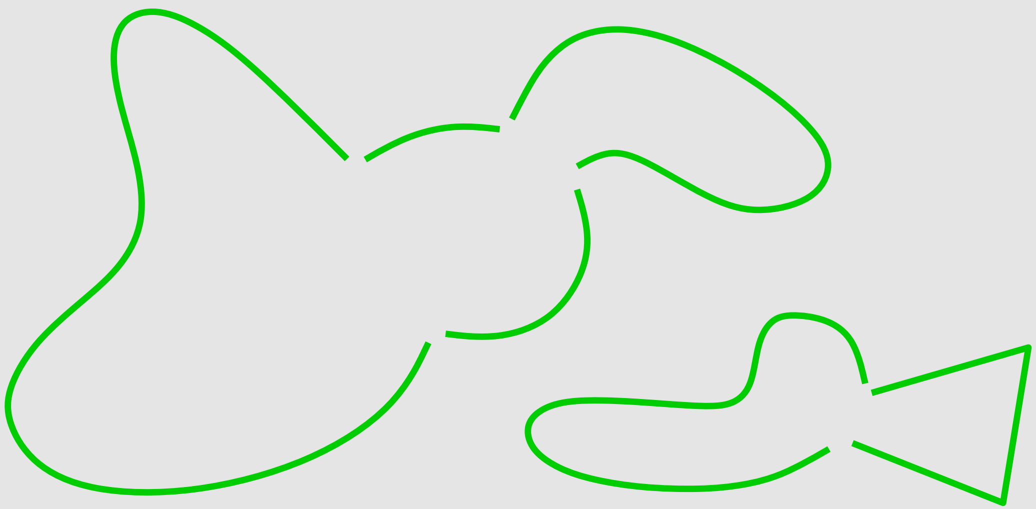


S: n locally γ-fat objects

E: “edges” of boundary of union of S

dB ’05

The set E has density $O(1/\gamma)$.

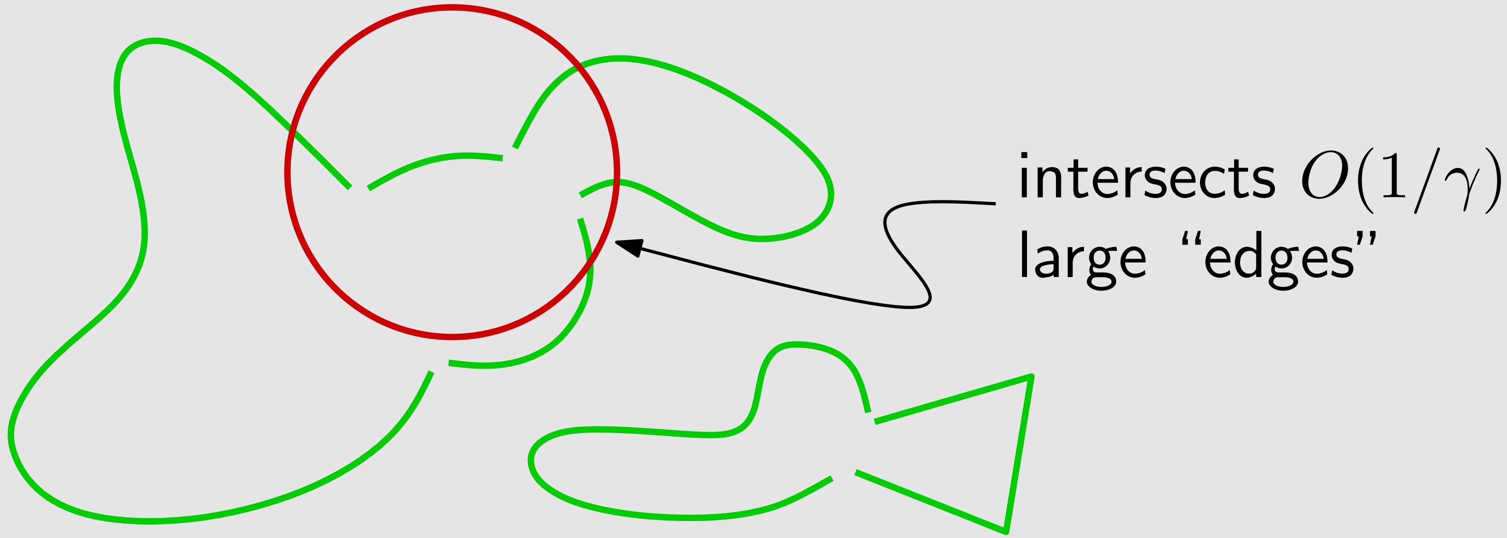


S: n locally γ-fat objects

E: "edges" of boundary of union of S

dB '05

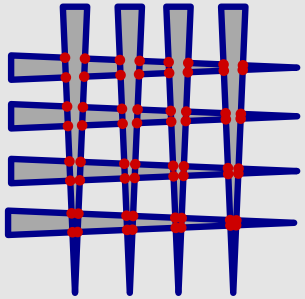
$\boxed{\text{The set } E \text{ has density } O(1/\gamma).}$



- combinatorics
  - overview of results
  - recent result: union of fat triangles
- algorithms and data structures
  - overview of results
  - some tools and examples

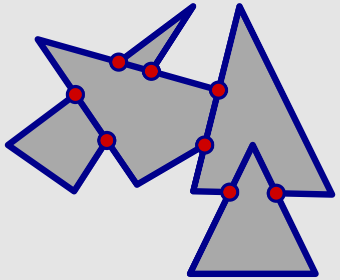
Combinatorics

for Fat Objects and Low Density Scenes



n arbitrary triangles:

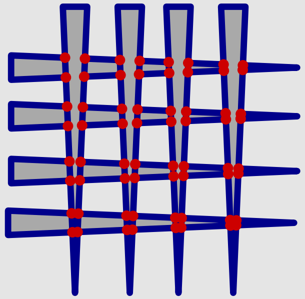
union complexity can be $\Theta(n^2)$



[Matoušek *et al.* '94, Pach-Tardos '02]

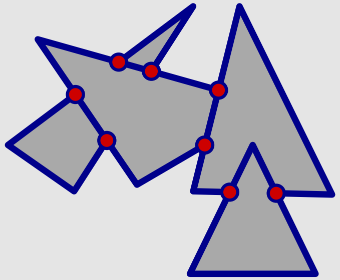
n fat triangles: complexity is $O(n \log\log n)$

Conjecture: $\Theta(n\alpha(n))$



n arbitrary triangles:

union complexity can be $\Theta(n^2)$



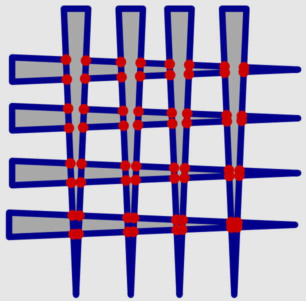
[Matoušek *et al.* '94, Pach-Tardos '02]

n fat triangles: complexity is $O(n \log \log n)$

Conjecture: $\Theta(n\alpha(n))$

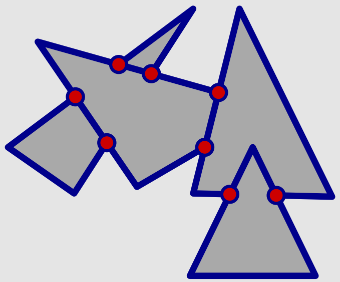
Ezra-Aronov-Sharir (SODA 11): $O(n2^{\alpha(n)} \log^* n)$

Fat objects: bounds on the union complexity



n arbitrary triangles:

union complexity can be $\Theta(n^2)$

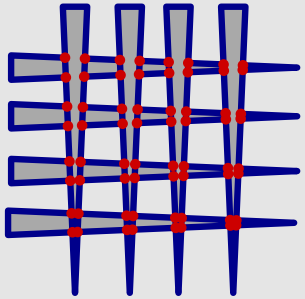


[Matoušek *et al.* '94, Pach-Tardos '02]

n fat triangles: complexity is $O(n \log \log n)$

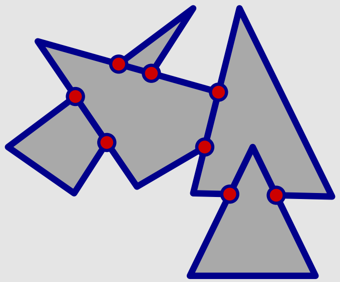
Conjecture: $\Theta(n\alpha(n))$

Ezra-Aronov-Sharir + dB: $O(n \log^* n)$



n arbitrary triangles:

union complexity can be $\Theta(n^2)$

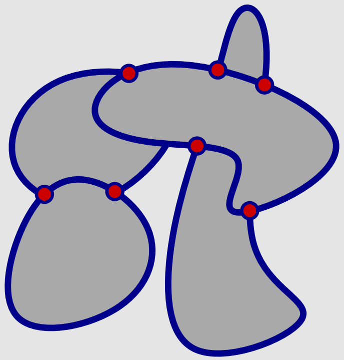


[Matoušek *et al.* '94, Pach-Tardos '02]

n fat triangles: complexity is $O(n \log\log n)$

Conjecture: $\Theta(n\alpha(n))$

Ezra-Aronov-Sharir + dB: $O(n \log^* n)$



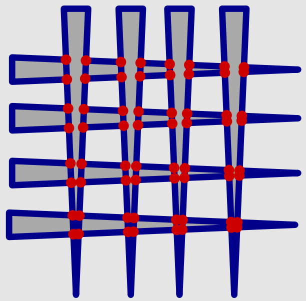
Efrat-Sharir '97, Efrat-Katz '98, Efrat '05, dB '05, dB'10

n fat (possibly non-convex and/or curved) objects:

complexity is $O(\lambda_{s+2}(n) \log n)$

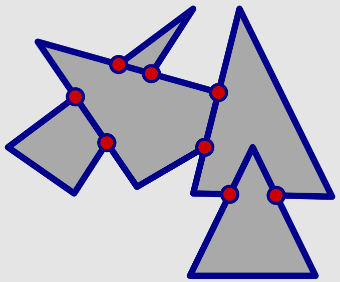
Conjecture: $O(\lambda_{s+2}(n))$

Fat objects: bounds on the union complexity



n arbitrary triangles:

union complexity can be $\Theta(n^2)$

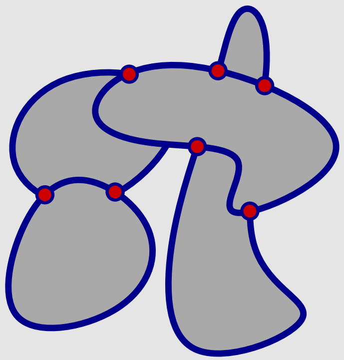


[Matoušek *et al.* '94, Pach-Tardos '02]

n fat triangles: complexity is $O(n \log \log n)$

Conjecture: $\Theta(n\alpha(n))$

Ezra-Aronov-Sharir + dB: $O(n \log^* n)$



Efrat-Sharir '97, Efrat-Katz '98, Efrat '05, dB '05, dB'10

n fat (possibly non-convex and/or curved) objects:

complexity is $O(\lambda_{s+2}(n) \log n)$

$O(n2^{O(\log^* n)})$

Union complexity: objects in 3D

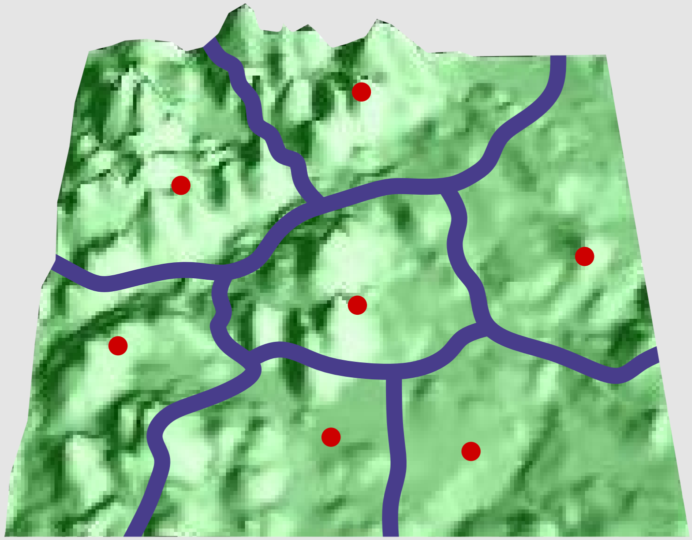
- worst case: $\Theta(n^3)$
- Pach *et al.* '03: arbitrarily oriented equal-sized cubes: $O(n^{2+\varepsilon})$
- Ezra-Sharir '07: fat tetrahedra: $O(n^{2+\varepsilon})$
- Aronov *et al.* '04: κ-round objects: $O(n^{2+\varepsilon})$
- fat convex polyhedra: open

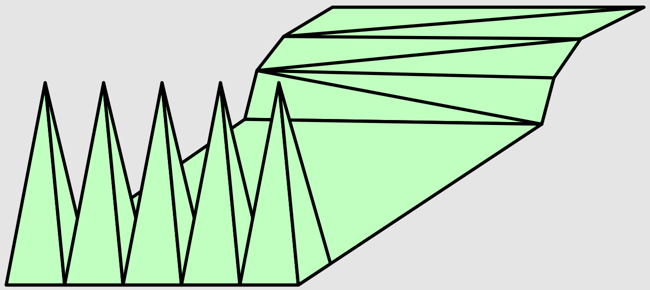
decompositions of (non-convex) polyhedra into tetrahedra

- worst case: $\Theta(n^2)$
- dB-Gray '08: fat polyhedra with fat faces: $O(n)$

complexity of Voronoi diagrams on terrains
Moet *et al.* '06, Aronov-dB-Thite '08

- terrain with n triangles, m sites
- worst-case: $\Omega(n^2)$ even for two sites
- on realistic terrain: $O(n + m\sqrt{n})$





complexity of visibility maps of terrains
Moet *et al.* '06, dB-Haverkort-Tsirogiannis '09

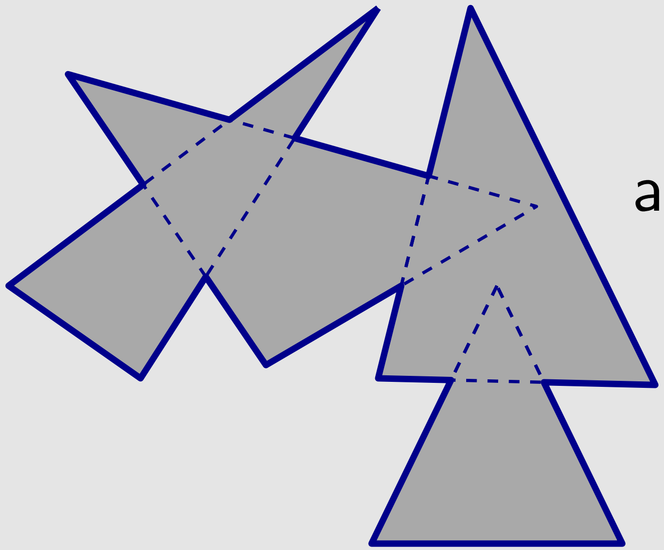
- worst case: $\Theta(n^2)$
- realistic terrain: $\Theta(n\sqrt{n})$
- realistic terrain with noise: $\Theta(n)$

river networks: $O(n^2)$ complexity on realistic terrains, instead of $O(n^3)$

The union complexity of γ-fat triangles

(and of locally γ-fat curved objects)

(joint work with Boris Aronov, Esther Ezra, and Micha Sharir)



all angles at least γ

basic tool: Merging Lemma

Let A and B be two sets of locally γ-fat objects in $\mathbb{R}^2$. Then

$$UC(A \cup B) \;=\; O((1/\gamma) \cdot (UC(A) + UC(B))),$$

where UC = union complexity

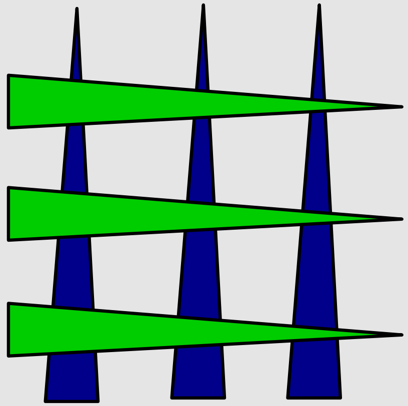
basic tool: Merging Lemma

Let A and B be two sets of locally γ-fat objects in $\mathbb{R}^2$. Then

$$UC(A \cup B) \;=\; O((1/\gamma) \cdot (UC(A) + UC(B))),$$

where UC = union complexity

Note: not true for non-fat objects



basic tool: Merging Lemma

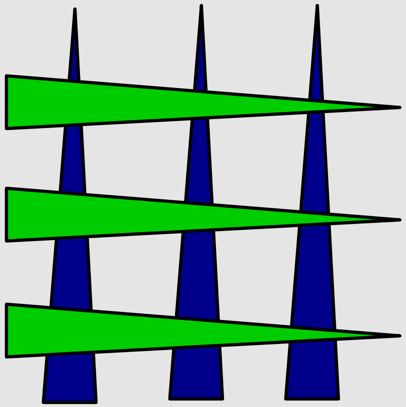
Let A and B be two sets of locally γ-fat objects in $\mathbb{R}^2$. Then

$$UC(A \cup B) \;=\; O((1/\gamma) \cdot (UC(A) + UC(B))),$$

where UC = union complexity

(alternative for polygonal objects: Combination Lemma)

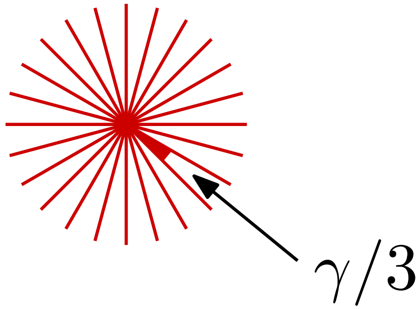
Note: not true for non-fat objects



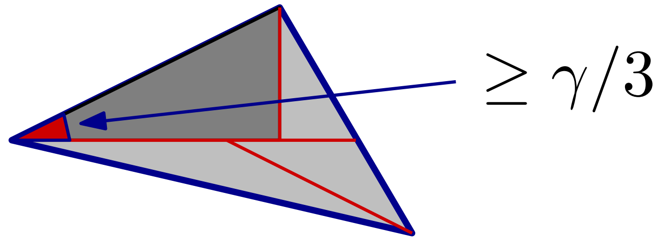
another tool

- cover each triangle by subtriangles with canonical shape

$6\pi/\gamma$ canonical directions



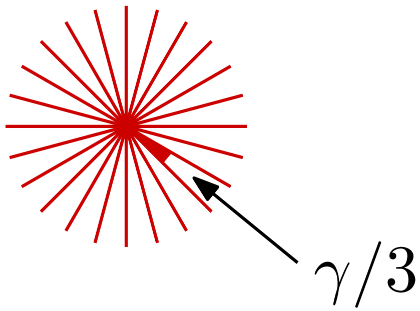
subtriangles have two edges with canonical directions



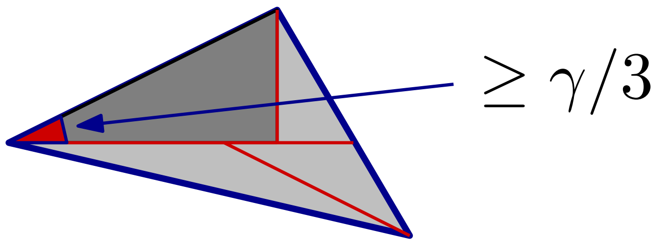
another tool

- cover each triangle by subtriangles with canonical shape

$6\pi/\gamma$ canonical directions



subtriangles have two edges with canonical directions

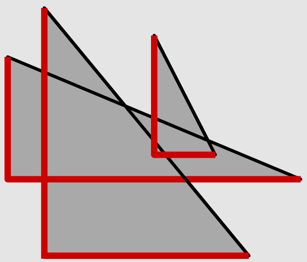


T: orginal set of n triangles $\Longrightarrow$ $T_1, T_2, \ldots$: $O(1/\gamma^2)$ sets of subtriangles

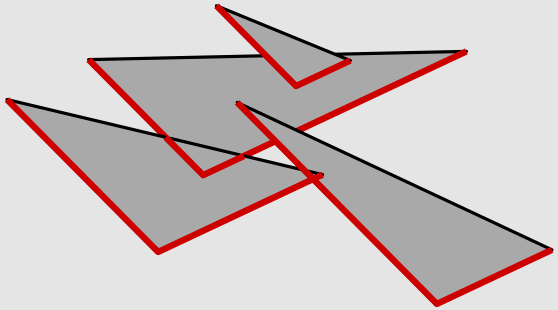
within each set all subtriangles are fat and use the same canonical directions

(in fact, $O(1/\gamma)$ sets suffice in this problem)

T_1:



T_2:



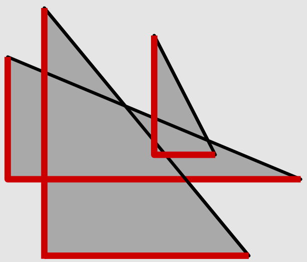
$\vdots$

each vertex of union of T is

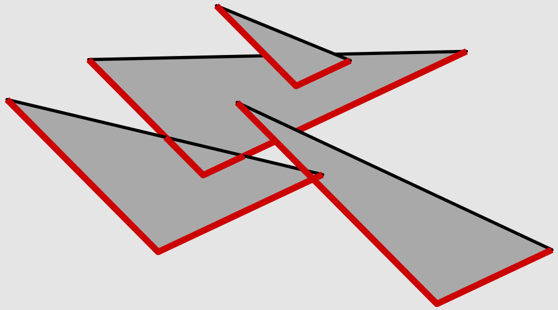
- a vertex of some union $\bigcup T_i$
- a vertex of some union $\bigcup (T_i \cup T_j)$

union complexity of fat triangles (cont'd)

T_1:



T_2:



•
•
•

each vertex of union of T is

- a vertex of some union $\bigcup T_i$
- a vertex of some union $\bigcup (T_i \cup T_j)$

Merging Lemma

$$\sum_{i,j} UC(T_i \cup T_j)$$

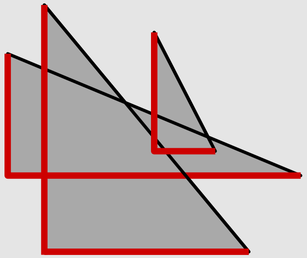
$$= \sum_{i,j} O(UC(T_i) + UC(T_j))$$

$$= \ (\text{ \# subsets }) \cdot \sum_i O(UC(T_i))$$

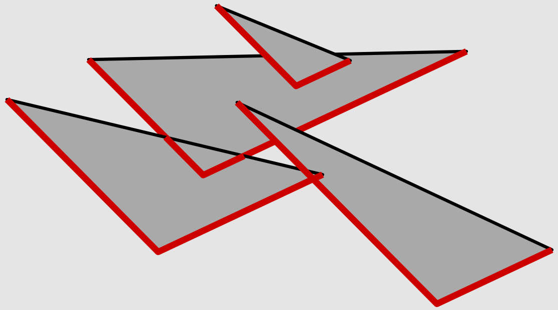
$$= O(1/\gamma^2) \cdot O(\textstyle\sum_i UC(T_i))$$

union complexity of fat triangles (cont'd)

T_1:



T_2:



$\vdots$

each vertex of union of T is

- a vertex of some union $\bigcup T_i$
- a vertex of some union $\bigcup (T_i \cup T_j)$

Merging Lemma

$$\sum_{i,j} UC(T_i \cup T_j)$$

$$= \sum_{i,j} O(UC(T_i) + UC(T_j))$$

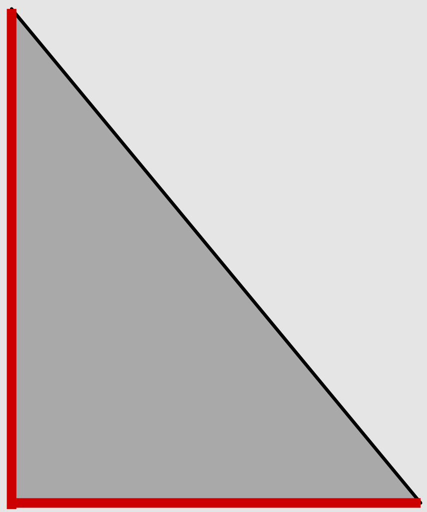
$$= (\text{ \# subsets }) \cdot \sum_i O(UC(T_i))$$

$$= O(1/\gamma^2) \cdot O(\sum_i UC(T_i))$$

bound for fat canonical triangles carries over to fat triangles

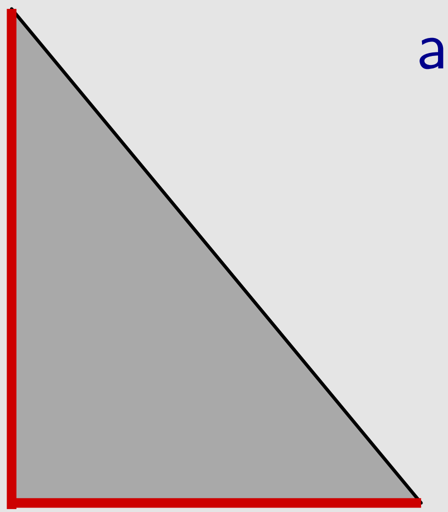
New problem: bound union complexity of n canonical triangles

- each triangle has a horizontal bottom edge
- each triangle has vertical left edge
- angles at top corner and right corner are at least γ
  (in fact, can get angles arbitrary close to 45°)



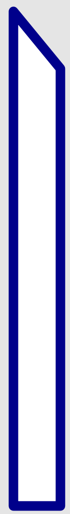
New problem: bound union complexity of n canonical triangles

- each triangle has a horizontal bottom edge
- each triangle has vertical left edge
- angles at top corner and right corner are at least γ
  (in fact, can get angles arbitrary close to 45°)



a further reduction:

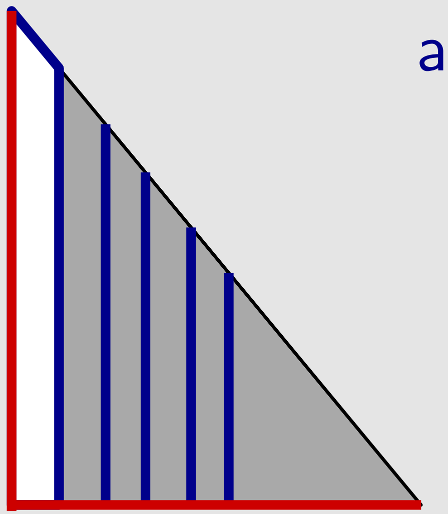
- cover triangles by towers:
  fat triangle on top of long
  (but still fat) rectangle



union complexity of fat triangles (cont'd)

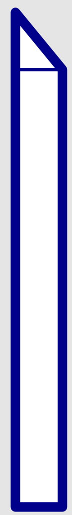
New problem: bound union complexity of n canonical triangles

- each triangle has a horizontal bottom edge
- each triangle has vertical left edge
- angles at top corner and right corner are at least γ
  (in fact, can get angles arbitrary close to 45°)



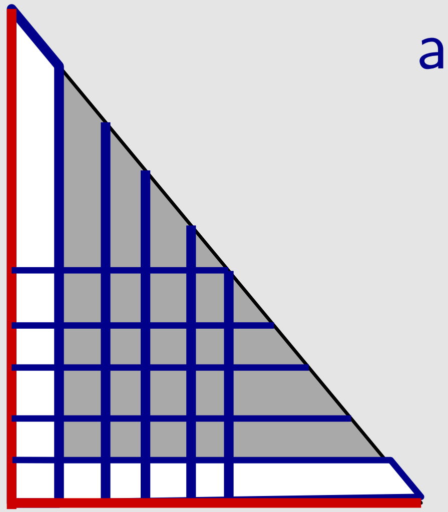
a further reduction:

- cover triangles by towers:
  fat triangle on top of long
  (but still fat) rectangle



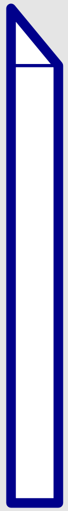
New problem: bound union complexity of n canonical triangles

- each triangle has a horizontal bottom edge
- each triangle has vertical left edge
- angles at top corner and right corner are at least γ
  (in fact, can get angles arbitrary close to 45°)



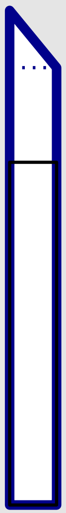
a further reduction:

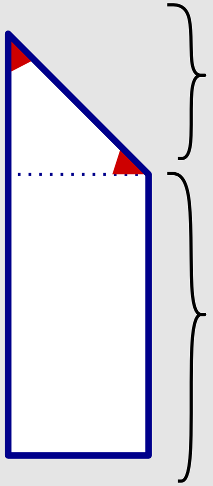
- cover triangles by towers:
  fat triangle on top of long
  (but still fat) rectangle



New problem: bound union complexity of n vertical towers

assume w.l.o.g.:



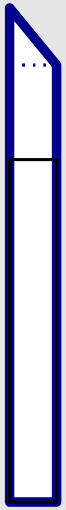


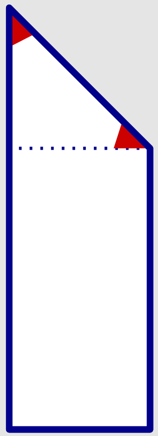
TU/e

union complexity of fat triangles (cont'd)

New problem: bound union complexity of n vertical towers

assume w.l.o.g.:



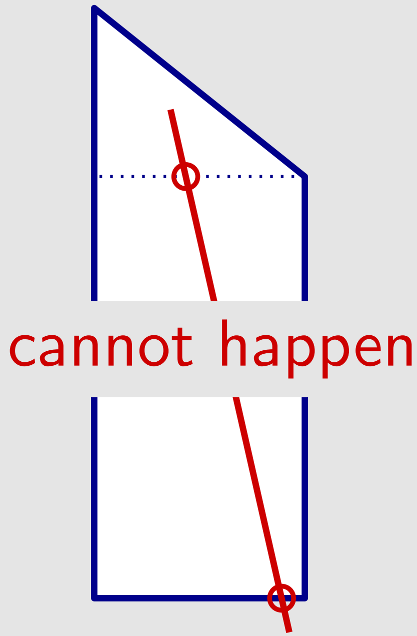


triangular top:
angles between 44° and 46°

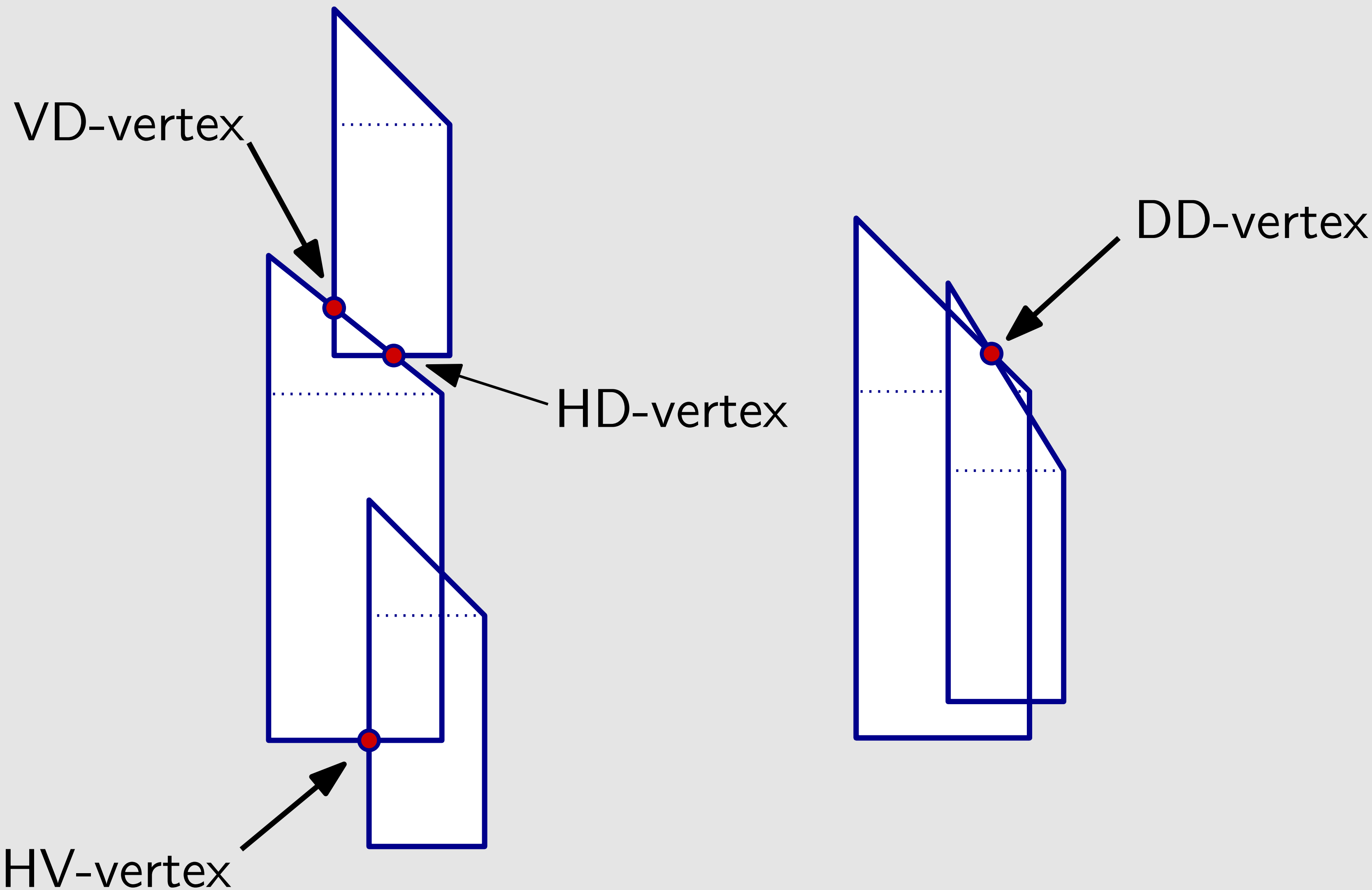
rectangular base:
height $= 2 \times$ width

crucial property:

diagonal edge of one tower cannot simultaneously intersect top edge and bottom edge of base of another tower



The union complexity of towers



Lemma: The number of HV-, HD, and VD-vertices is $O(n)$.

Lemma: The number of HV-, HD, and VD-vertices is $O(n)$.

Proof:

HV-vertices

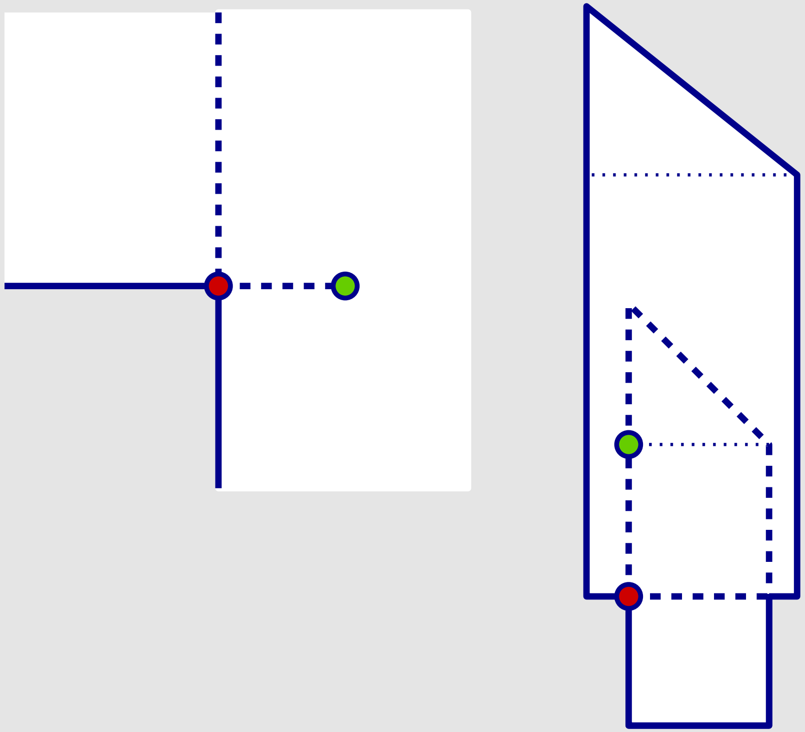
HD-vertices

Lemma: The number of HV-, HD, and VD-vertices is $O(n)$.

Proof:

HV-vertices

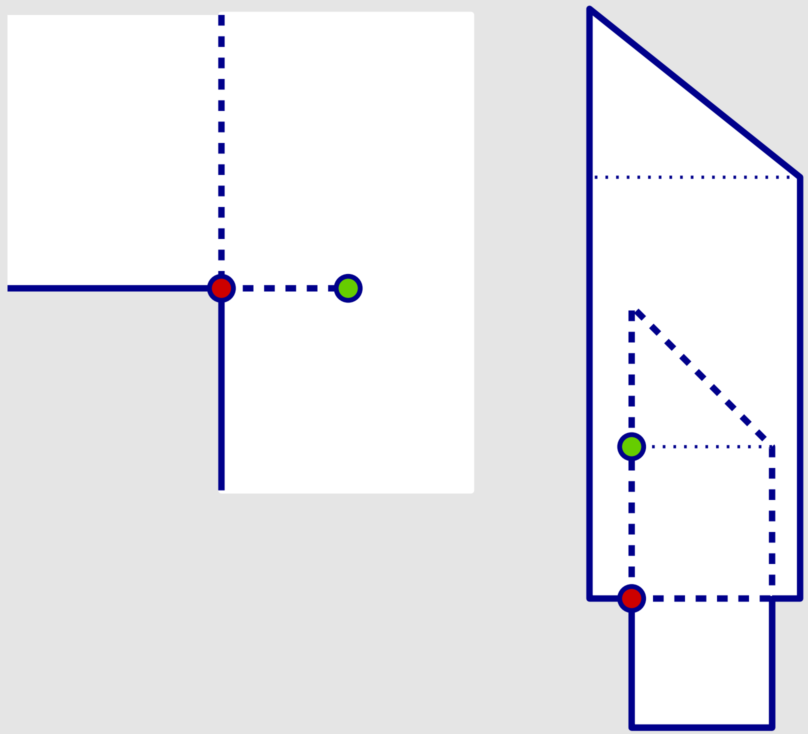
HD-vertices



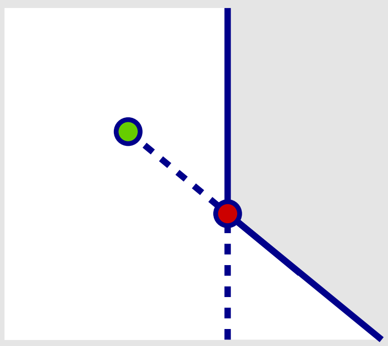
Lemma: The number of HV-, HD, and VD-vertices is $O(n)$.

Proof:

HV-vertices



HD-vertices



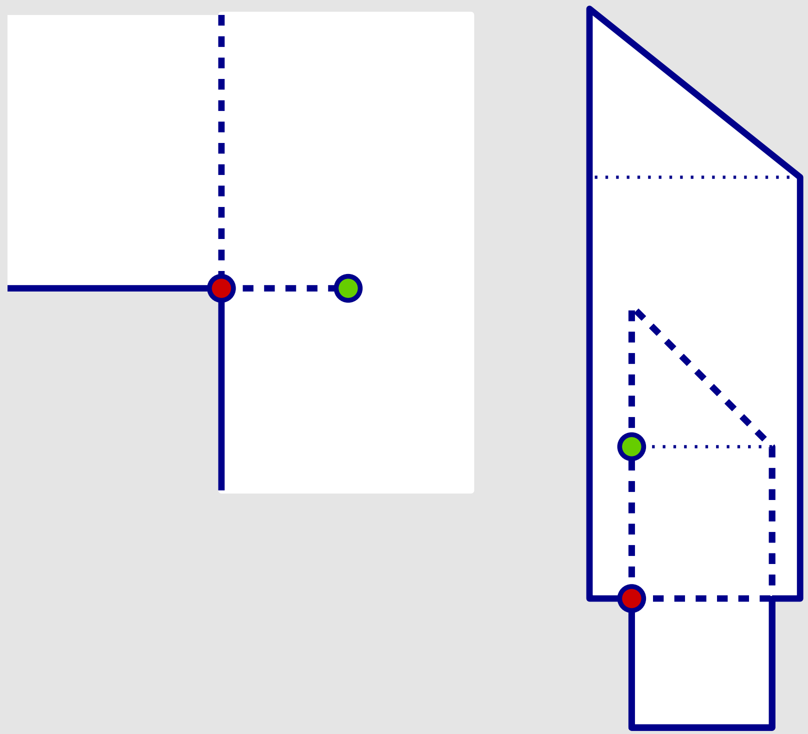
TU/e

union complexity of fat triangles (cont'd)

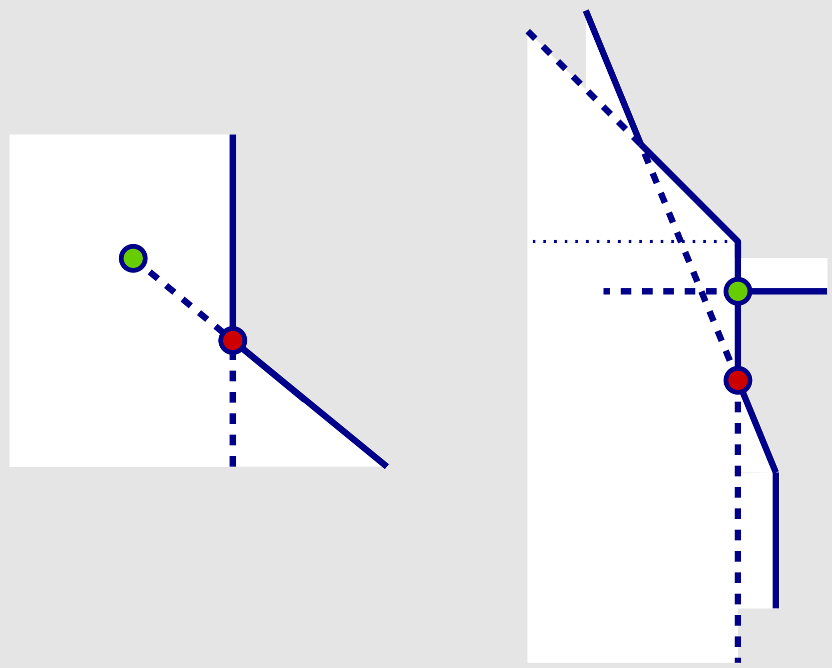
Lemma: The number of HV-, HD, and VD-vertices is $O(n)$.

Proof:

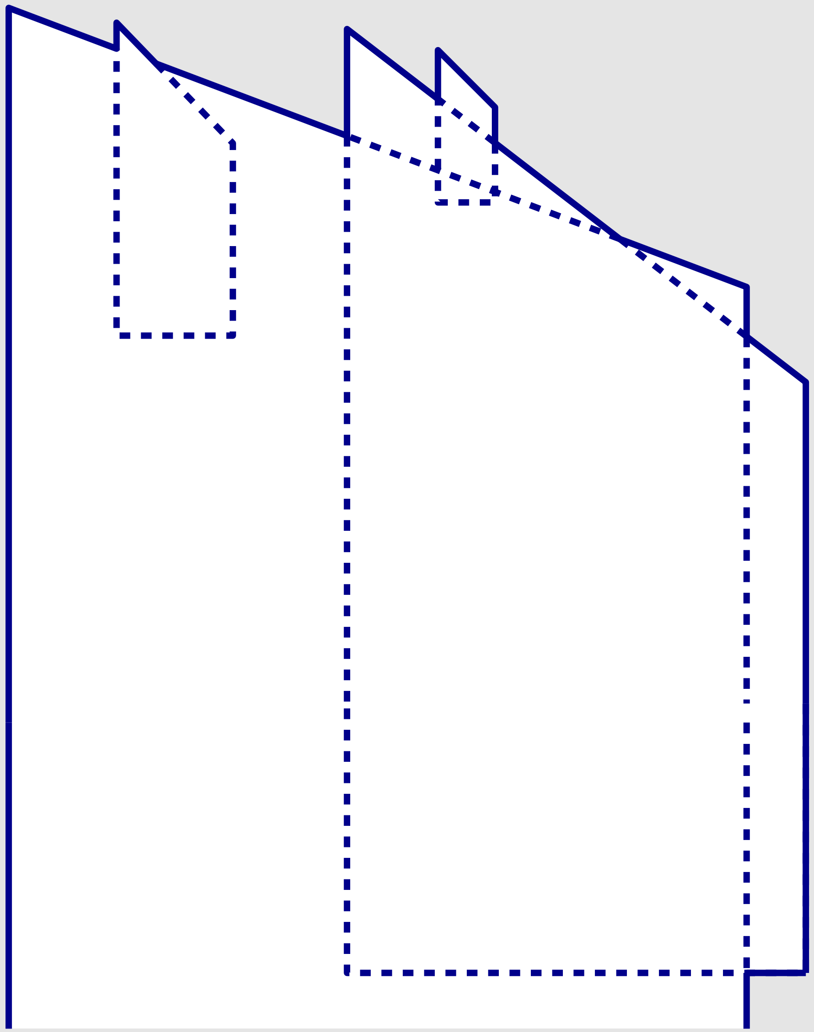
HV-vertices



HD-vertices

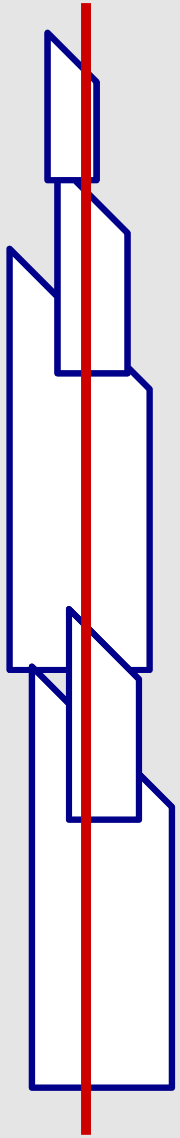


It remains to bound the number of DD-vertices



- upper-envelope complexity gives $\Omega(n\alpha(n))$ lower bound
- upper-envelopes connection also gives upper bound ??

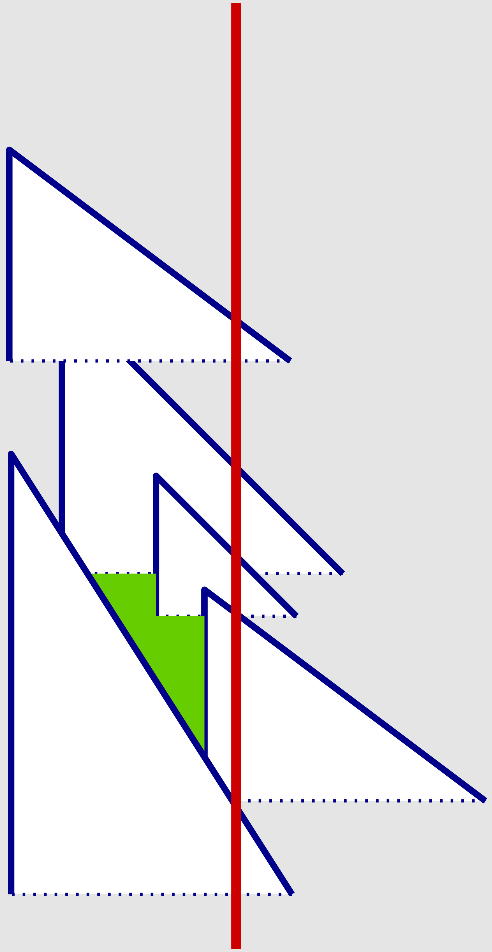
Lemma: Union complexity of n towers stabbed by a vertical line is $O(n)$.



Recall: we only need to worry about DD-vertices

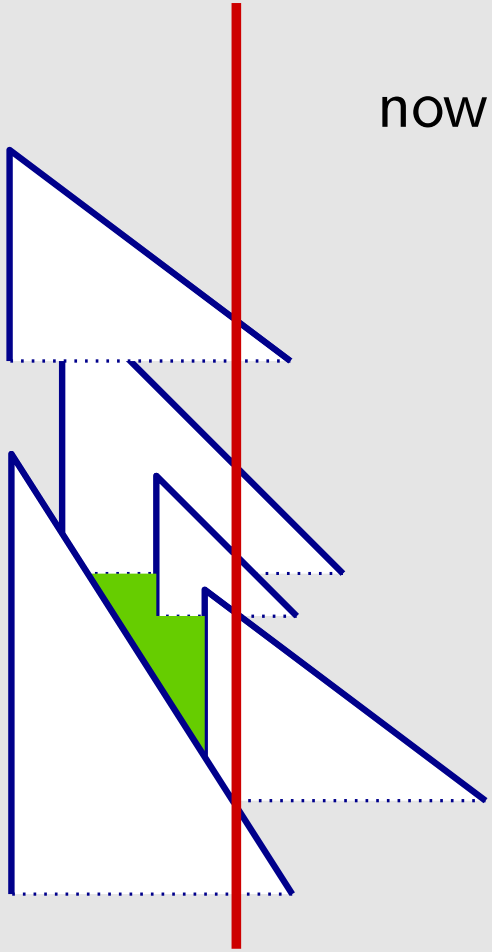
Lemma: Union complexity of n towers stabbed by a vertical line is $O(n)$.

Proof: only consider triangular top of each tower



Lemma: Union complexity of n towers stabbed by a vertical line is $O(n)$.

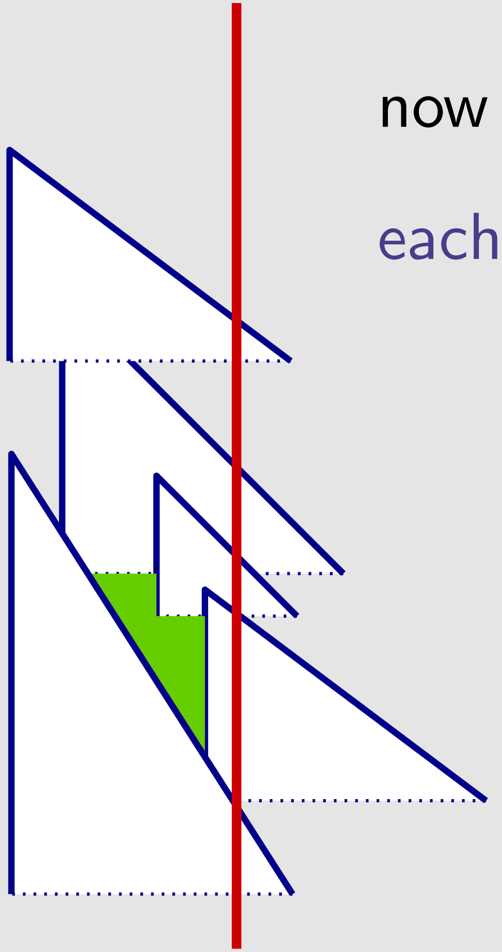
Proof: only consider triangular top of each tower



now consider impact of rectangular bases:

Lemma: Union complexity of n towers stabbed by a vertical line is $O(n)$.

Proof: only consider triangular top of each tower

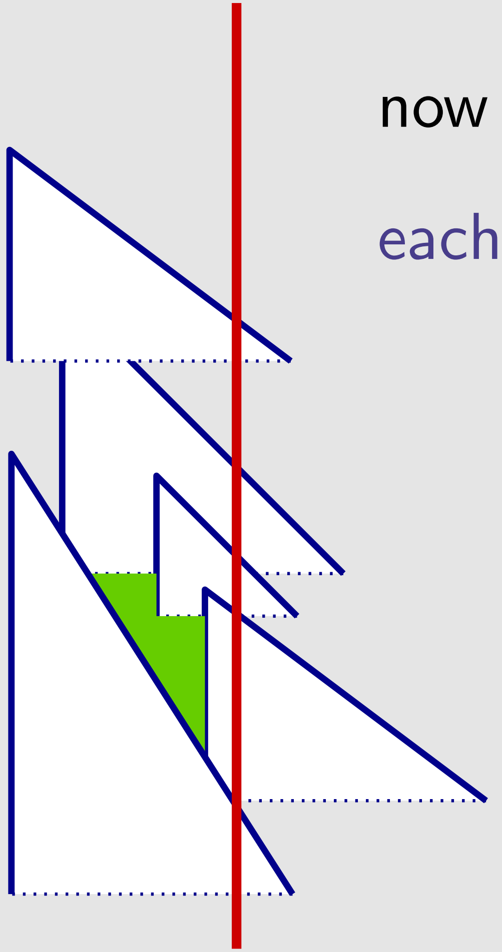


now consider impact of rectangular bases:

each hole must be covered by a base

Lemma: Union complexity of n towers stabbed by a vertical line is $O(n)$.

Proof: only consider triangular top of each tower



now consider impact of rectangular bases:

each hole must be covered by a base

$\Longrightarrow$ single cell: $O(n\alpha(n))$

[in fact, special type: $O(n)$]

Theorem: Union complexity of n fat triangles is $O(n \log^* n)$.

Proof:

1. Reduce problem to bounding union complexity of towers.
2. Prove that number of HV-, HD-, and VD-vertices is $O(n)$.
3. Prove that union complexity is $O(n)$ if triangles are stabbed by vertical line.
4. Use a clever scheme based on interval trees and intersection-sensitive cuttings to get a recursion that solves to $O(n \log^* n)$.

$\blacksquare$

Theorem: Union complexity of n fat triangles is $O(n \log^* n)$.

Proof:

1. Reduce problem to bounding union complexity of towers.
2. Prove that number of HV-, HD-, and VD-vertices is $O(n)$.
3. Prove that union complexity is $O(n)$ if triangles are stabbed by vertical line.
4. Use a clever scheme based on interval trees and intersection-sensitive cuttings to get a recursion that solves to $O(n \log^* n)$.

$\blacksquare$

Similar proof gives $O(n 2^{O(\log^* n)})$ bound for locally fat curved objects.

Algorithms and Data Structures
for Fat Objects and Low Density Scenes

overview of results: algorithms and data structures

data structures: storage for $O(\text{polylog } n)$ query time in 3D

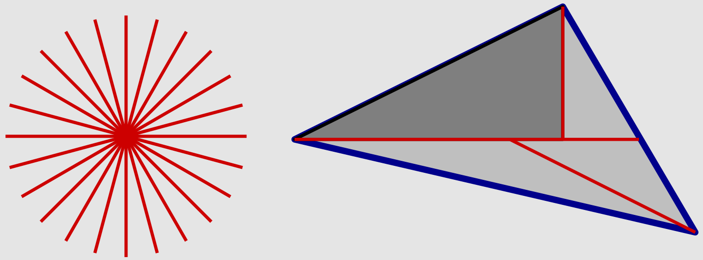
		general		
BSP trees		n^2	n	
point location		$n \log n$	n	constant density
intersection searching	approx	—	n	
	exact	n^4	n^3	
ray shooting	arbitrary	n^4	n^2	constant fatness
	vertical	n^2	$n \log^2 n$	

algorithms (some assumptions omitted)

f-DOF motion planning	n^f	$n \log n$	constant density
depth orders	$n^{4/3}$	$n \log^3 n$	
hidden-surface removal	$n^{4/3}$	n polylog n	constant fatness
kinetic collision detection	n^2	n polylog n	

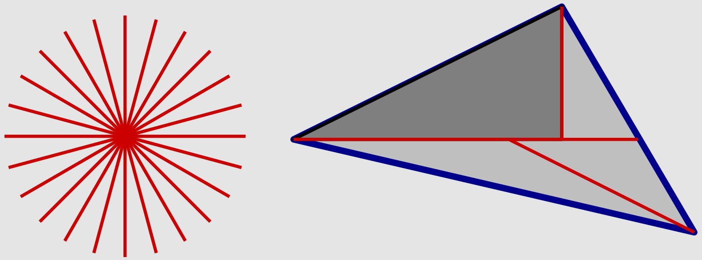
fat objects:

- cover objects by simpler objects using canonical directions



canonical directions can be handled efficiently fat objects:

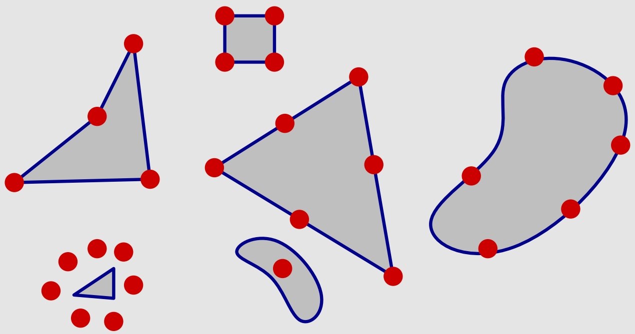
- cover objects by simpler objects using canonical directions



canonical directions can be handled efficiently

sets with low density

- replace objects by carefully chosen points ("guards")



guards "represent" distribution of objects

Example: how to find and use guarding points

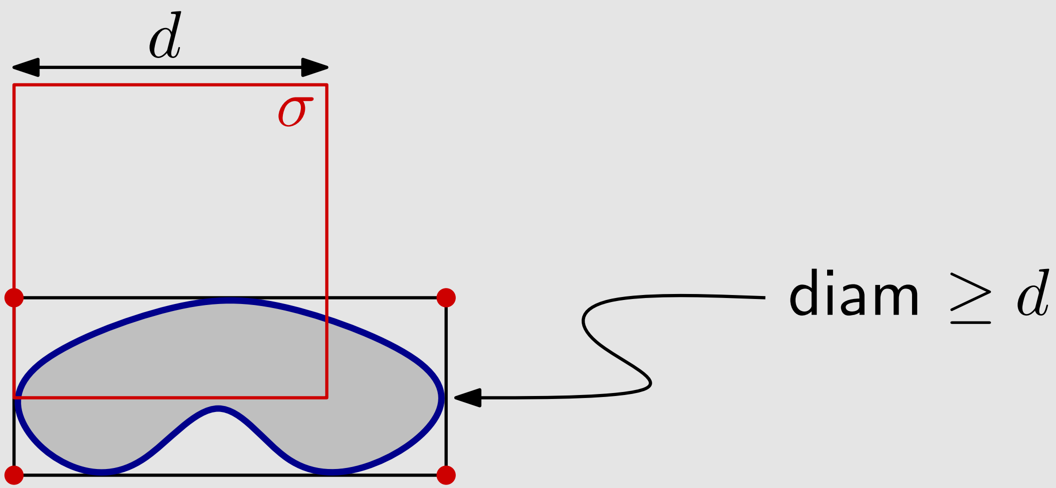
S: set of n objects

λ: density of S

G: set of $4n$ bounding-box vertices of the objects in S

Any square not containing bounding-box vertex intersects $\leq 4\lambda$ objects.

Proof:



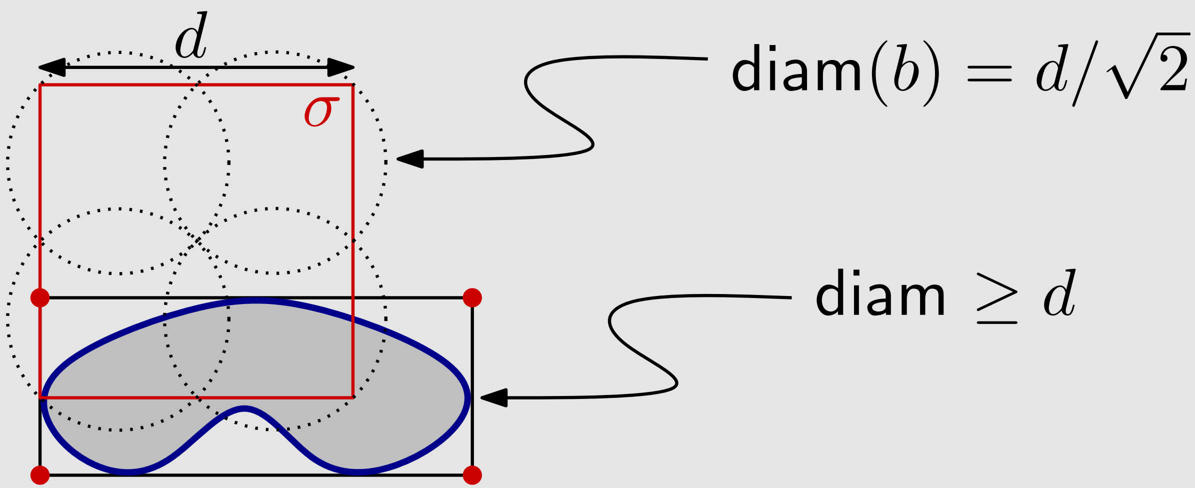
S: set of n objects

λ: density of S

G: set of $4n$ bounding-box vertices of the objects in S

Any square not containing bounding-box vertex intersects $\leq 4\lambda$ objects.

Proof:



$\blacksquare$

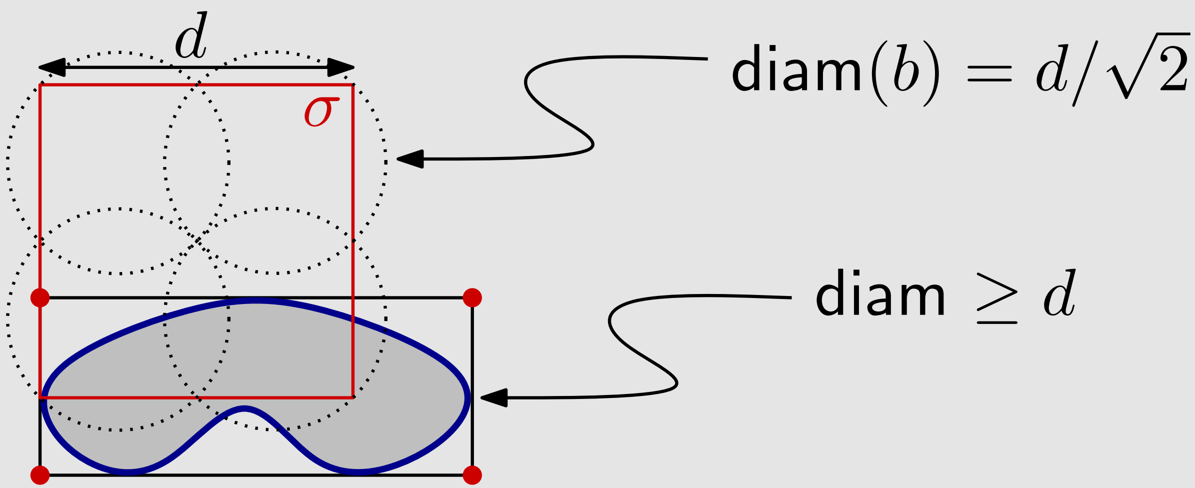
S: set of n objects

λ: density of S

G: set of $4n$ bounding-box vertices of the objects in S

Any square not containing bounding-box vertex intersects $\leq 4\lambda$ objects.

Proof:

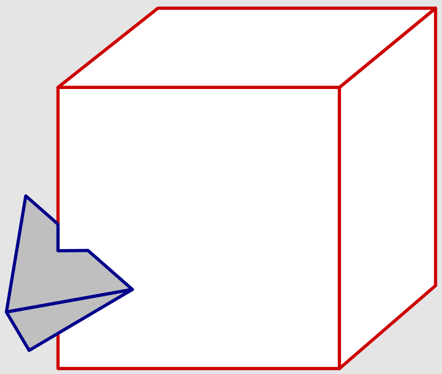


■

Note: similar statement holds in higher dimensions.

Guarding points with respect to squares / cubes (cont'd)

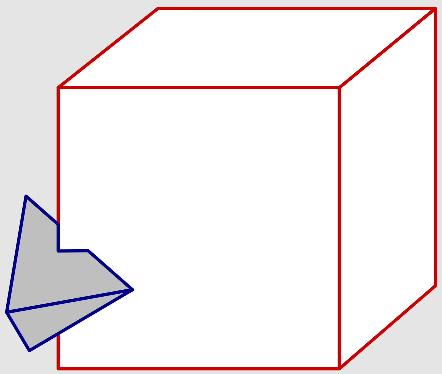
Using vertices of polyhedral object as guards does not work.



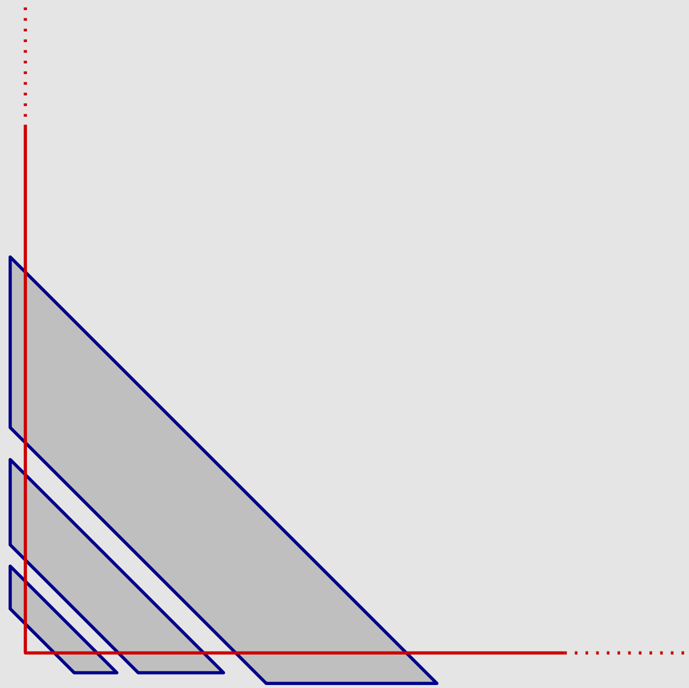
not in 3D . . .

Guarding points with respect to squares / cubes (cont'd)

Using vertices of polyhedral object as guards does not work.



not in 3D ...



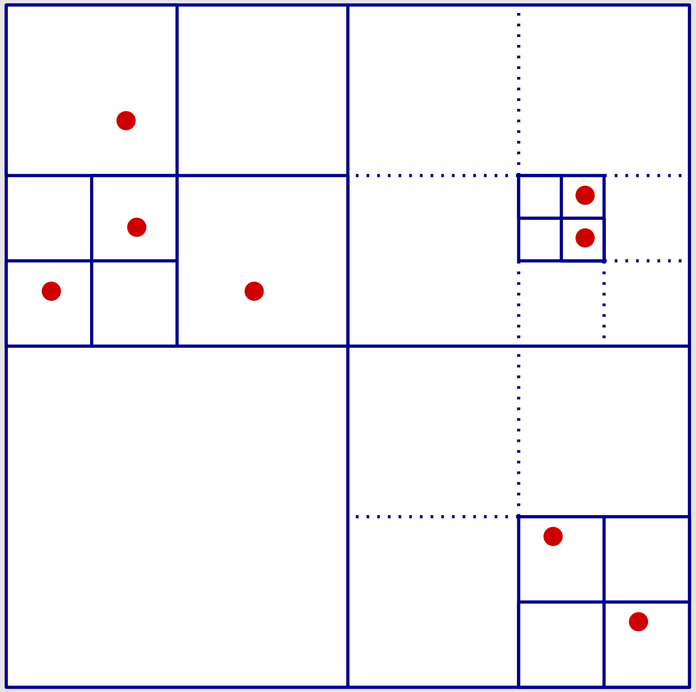
...not even in 2D

Compressed quadtrees for low-density scenes (based on dB-Haverkort-Thite-Toma '10)

- size of compressed quadtree is linear
- works in any dimension
- can be used to do point location
- can be used to do map overlay
- can be used to obtain linear size BSP
- can be used to do approximate range searching (range = convex region; data = low-density set)
- can be made I/O-efficient

Compressed quadtrees for point sets:

compress paths with only one non-empty subtree into single nodes



number of cells: $O(n)$

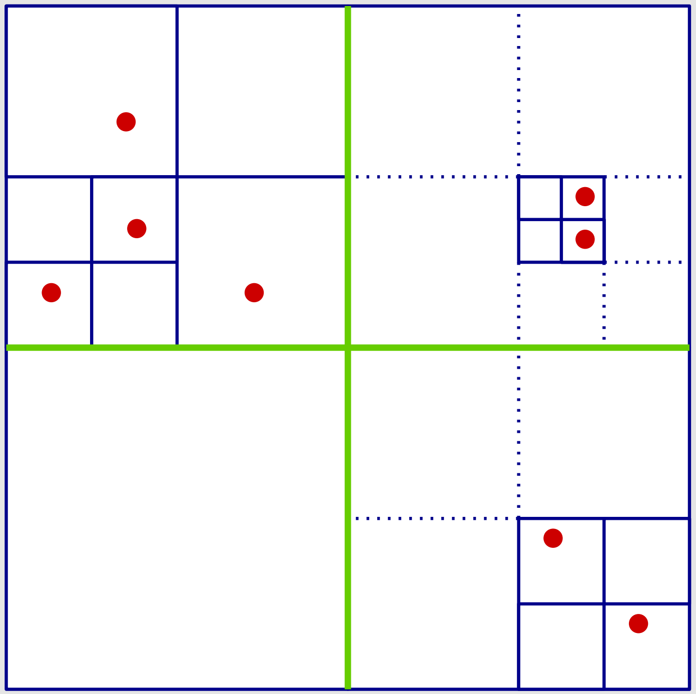
Recursively partition square

- points in more than one quadrant: split into quadrants
- all points in same quadrant: split into
  - smallest quadtree square containing all the poins
  - donut (contains no points)

Stop when zero or one point left

Compressed quadtrees for point sets:

compress paths with only one non-empty subtree into single nodes



number of cells: $O(n)$

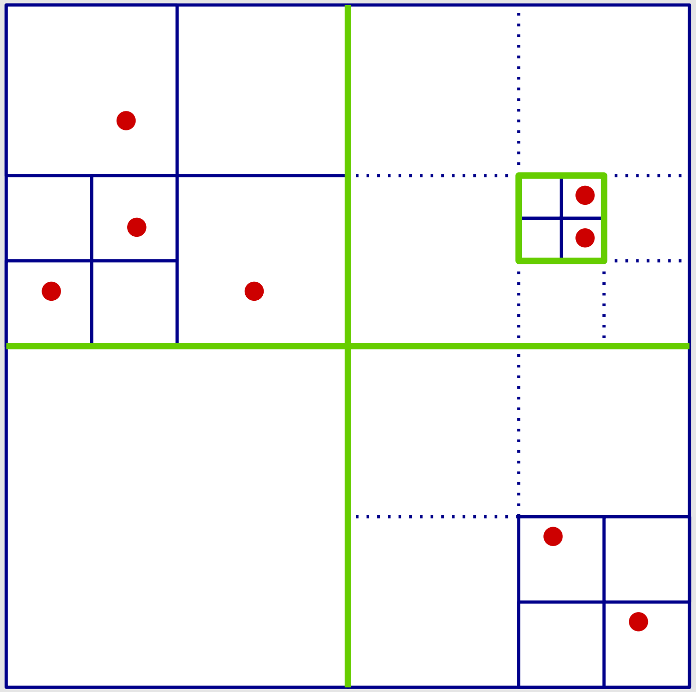
Recursively partition square

- points in more than one quadrant: split into quadrants
- all points in same quadrant: split into
  - smallest quadtree square containing all the poins
  - donut (contains no points)

Stop when zero or one point left

Compressed quadtrees for point sets:

compress paths with only one non-empty subtree into single nodes



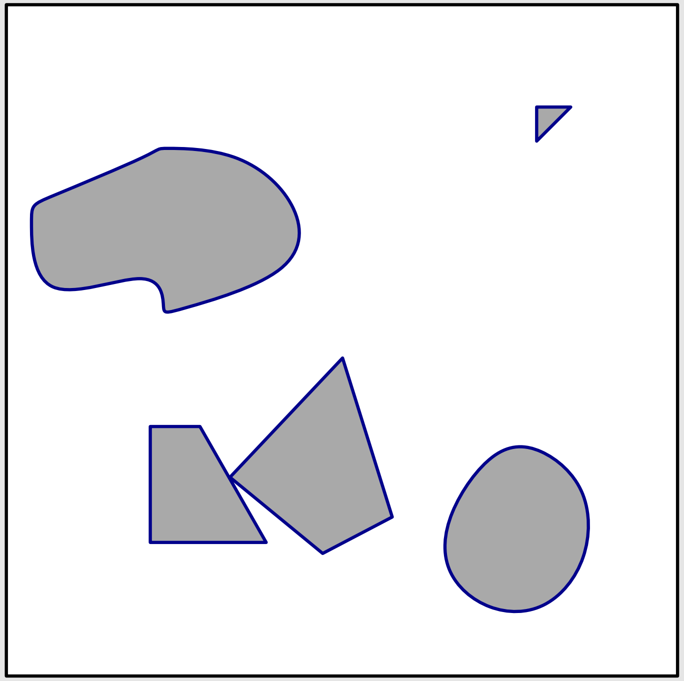
number of cells: $O(n)$

Recursively partition square

- points in more than one quadrant: split into quadrants
- all points in same quadrant: split into
  - smallest quadtree square containing all the poins
  - donut (contains no points)

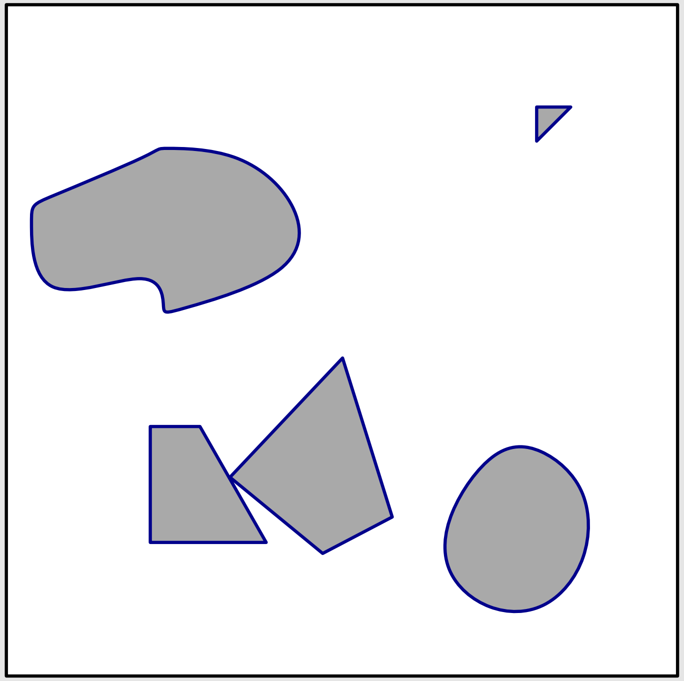
Stop when zero or one point left

Compressed quadtrees for low-density scenes



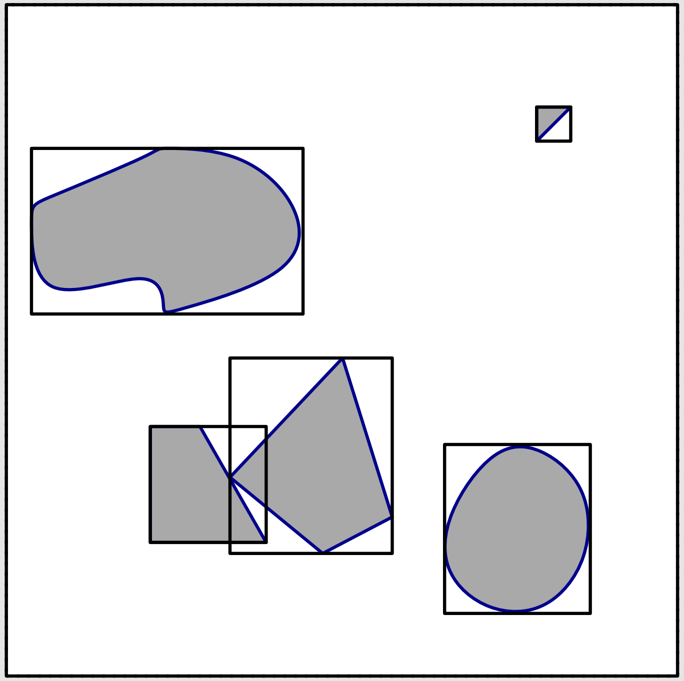
Compressed quadtrees for low-density scenes

Step 1: replace objects by vertices of bounding boxes



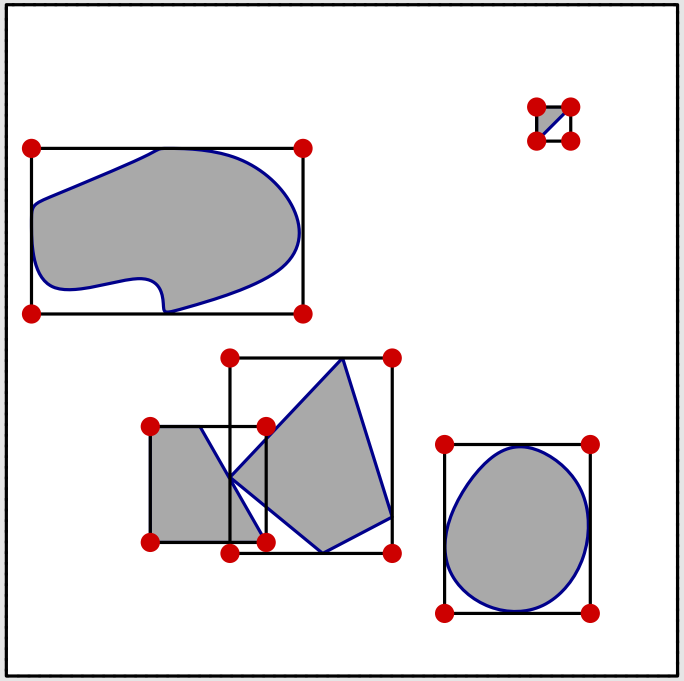
Compressed quadtrees for low-density scenes

Step 1: replace objects by vertices of bounding boxes



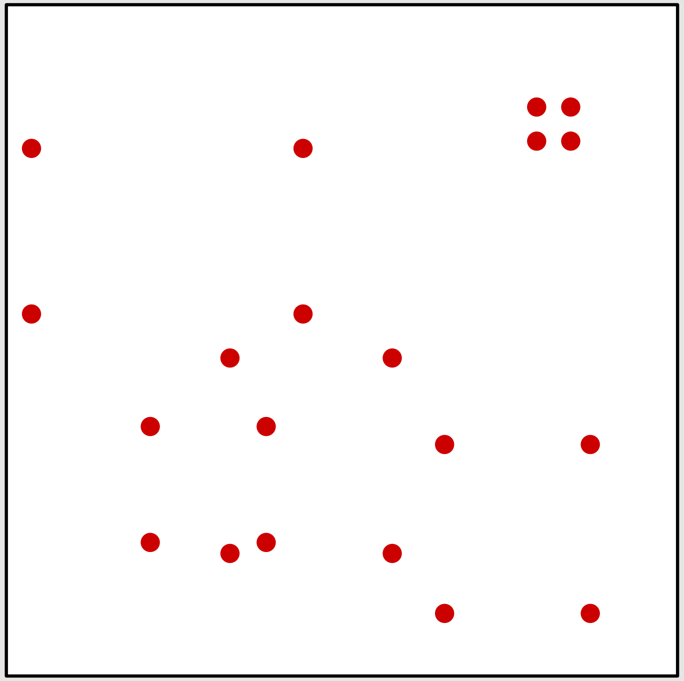
Compressed quadtrees for low-density scenes

Step 1: replace objects by vertices of bounding boxes



Compressed quadtrees for low-density scenes

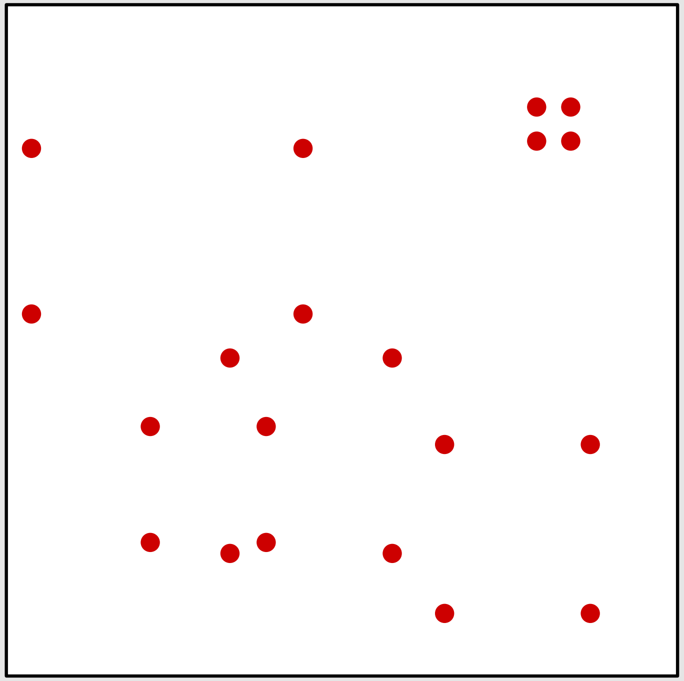
Step 1: replace objects by vertices of bounding boxes



Compressed quadtrees for low-density scenes

Step 1: replace objects by vertices of bounding boxes

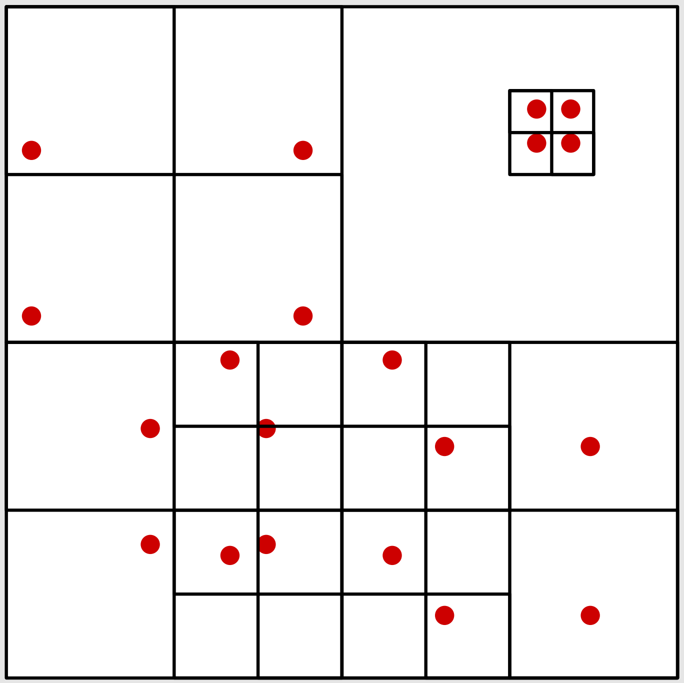
Step 2: construct compressed quadtree for resulting set of points



Compressed quadtrees for low-density scenes

Step 1: replace objects by vertices of bounding boxes

Step 2: construct compressed quadtree for resulting set of points

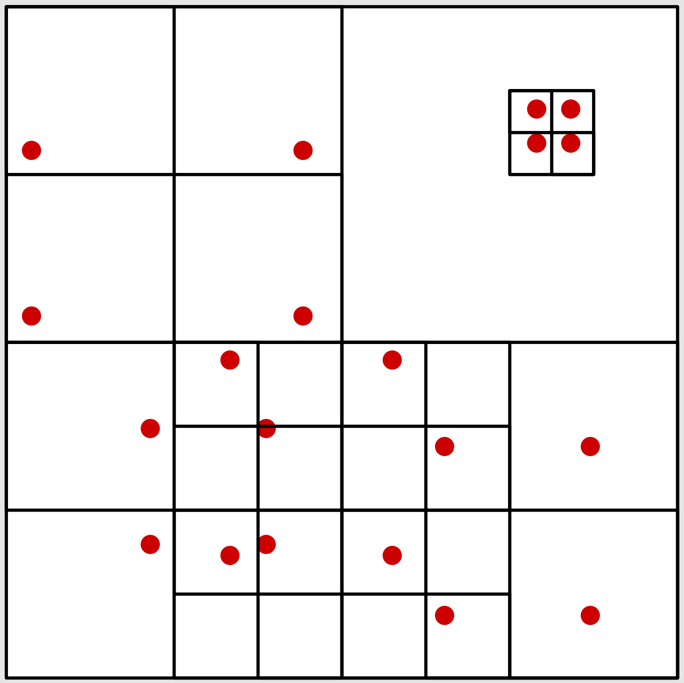


Compressed quadtrees for low-density scenes

Step 1: replace objects by vertices of bounding boxes

Step 2: construct compressed quadtree for resulting set of points

. . . and put objects back

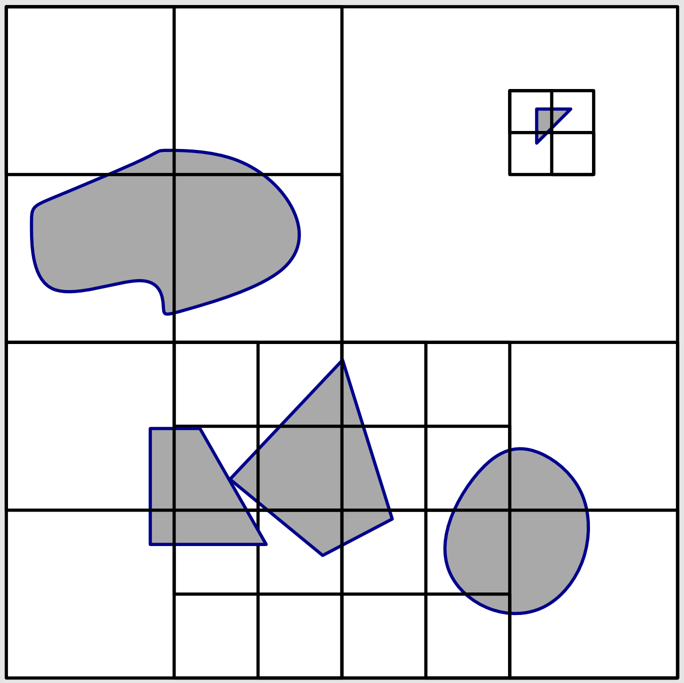


Compressed quadtrees for low-density scenes

Step 1: replace objects by vertices of bounding boxes

Step 2: construct compressed quadtree for resulting set of points

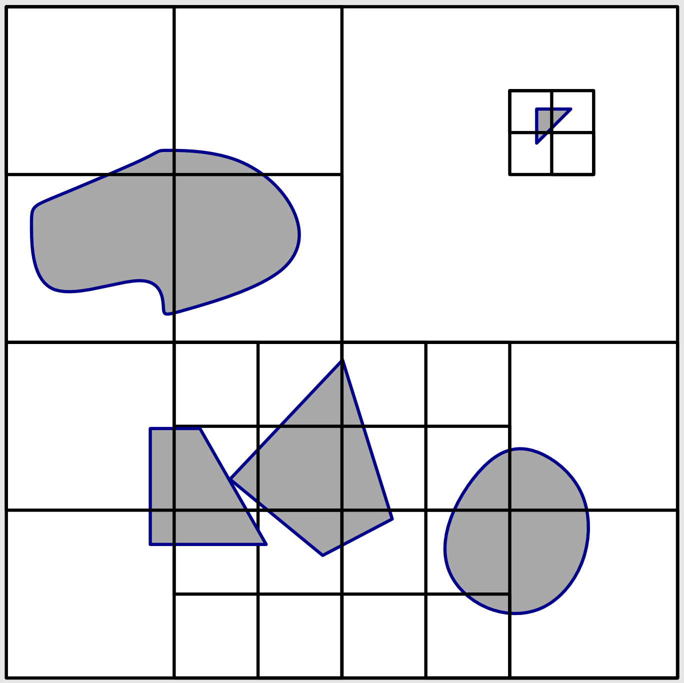
. . . and put objects back



Compressed quadtrees for low-density scenes ($\lambda =$ density)

Step 1: replace objects by vertices of bounding boxes

Step 2: construct compressed quadtree for resulting set of points



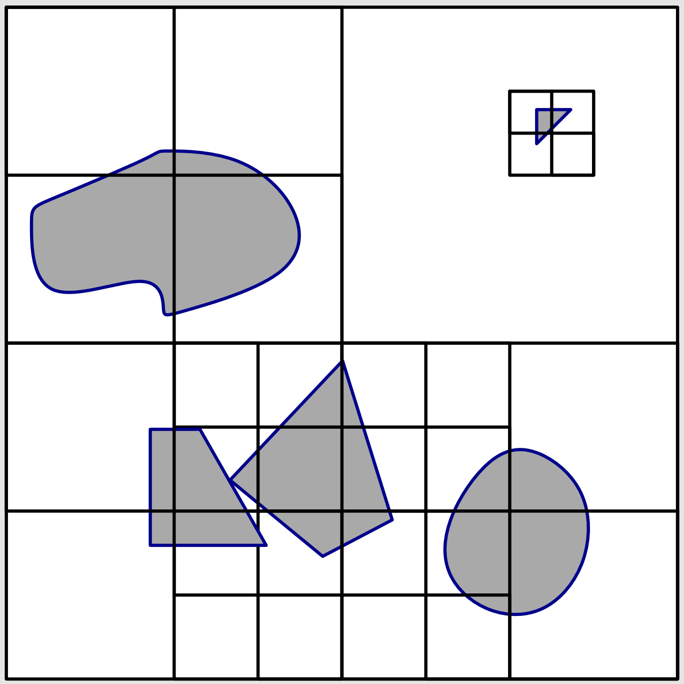
. . . and put objects back

- number of cells: $O(n)$
- number of objects per cell: $O(\lambda)$

Compressed quadtrees for low-density scenes (λ = density)

Step 1: replace objects by vertices of bounding boxes

Step 2: construct compressed quadtree for resulting set of points



. . . and put objects back

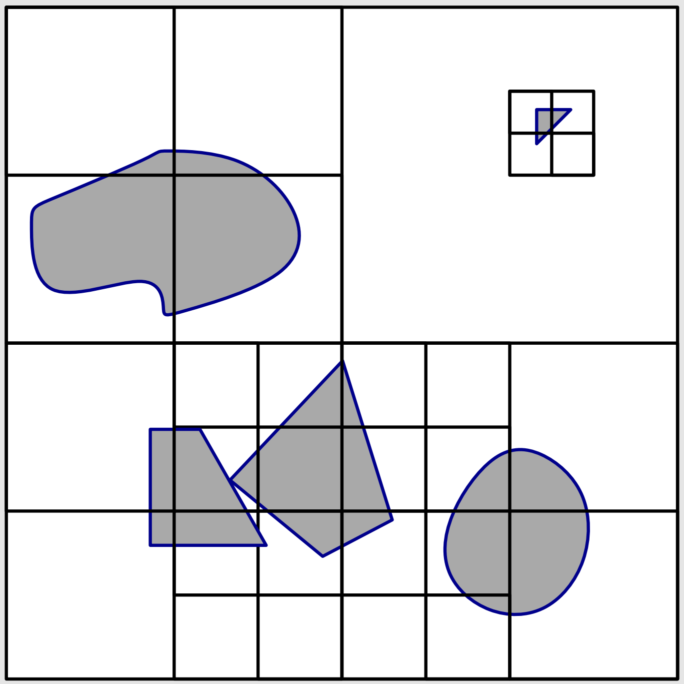
- number of cells: $O(n)$
- number of objects per cell: $O(\lambda)$

Any square not containing bounding-box vertex intersects $\leq 4\lambda$ objects.

Compressed quadtrees for low-density scenes (λ = density)

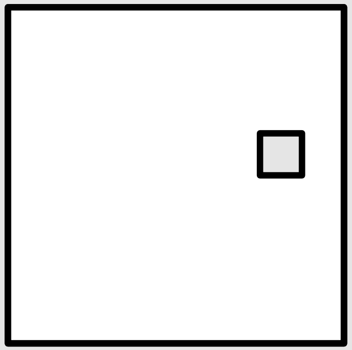
Step 1: replace objects by vertices of bounding boxes

Step 2: construct compressed quadtree for resulting set of points



... and put objects back

- number of cells: $O(n)$
- number of objects per cell: $O(\lambda)$



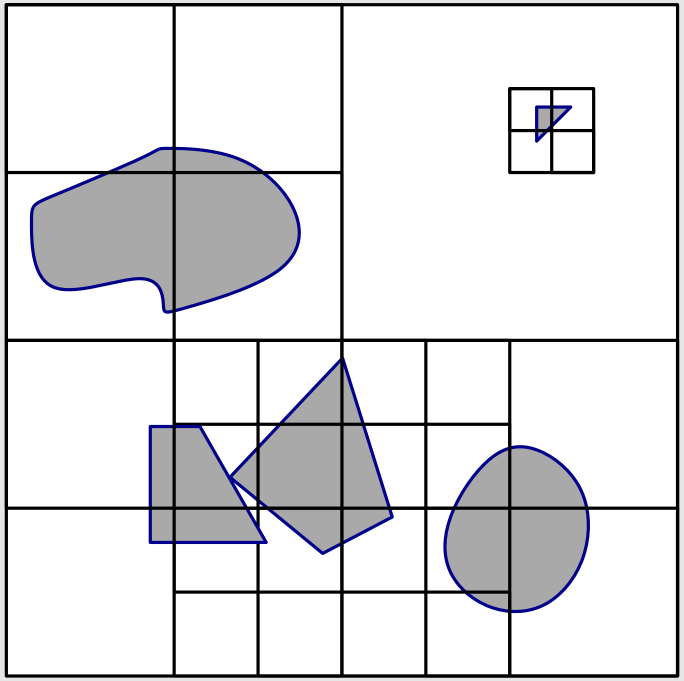
donut can be covered by six squares

Any square not containing bounding-box vertex intersects $\leq 4\lambda$ objects.

Compressed quadtrees for low-density scenes ($\lambda =$ density)

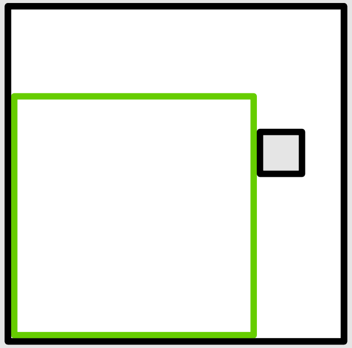
Step 1: replace objects by vertices of bounding boxes

Step 2: construct compressed quadtree for resulting set of points



. . . and put objects back

- number of cells: $O(n)$
- number of objects per cell: $O(\lambda)$



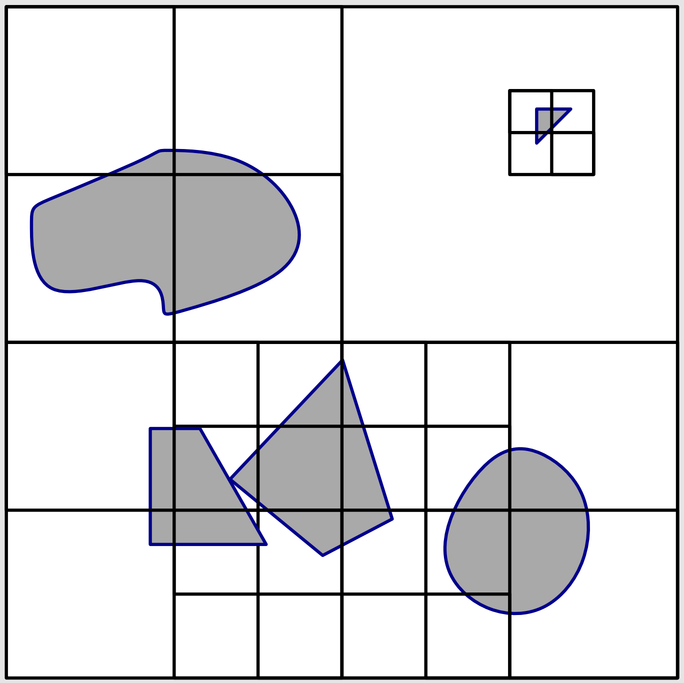
donut can be covered by six squares

Any square not containing bounding-box vertex intersects $\leq 4\lambda$ objects.

Compressed quadtrees for low-density scenes ($\lambda =$ density)

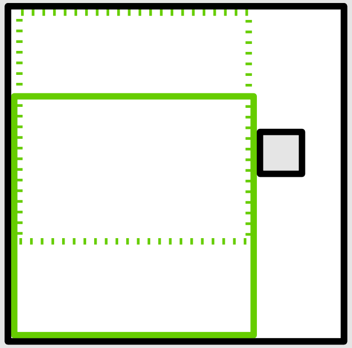
Step 1: replace objects by vertices of bounding boxes

Step 2: construct compressed quadtree for resulting set of points



. . . and put objects back

- number of cells: $O(n)$
- number of objects per cell: $O(\lambda)$



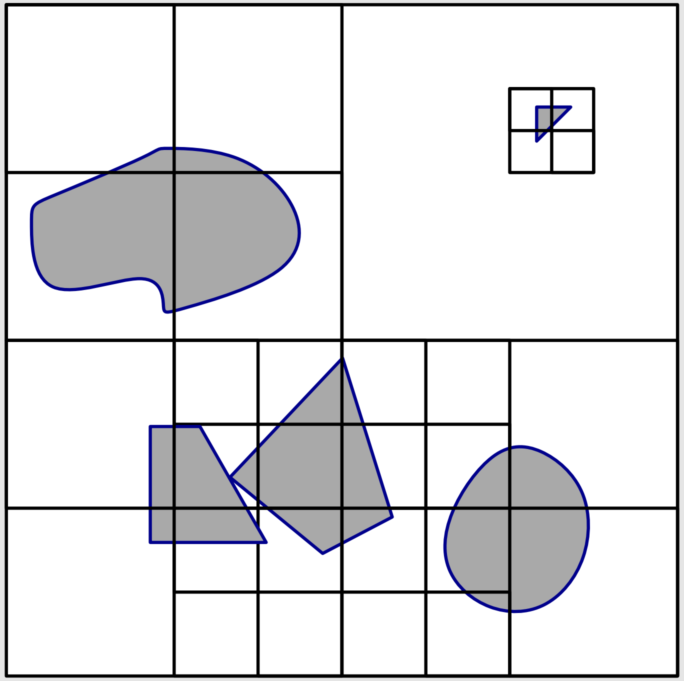
donut can be covered by six squares

Any square not containing bounding-box vertex intersects $\leq 4\lambda$ objects.

Compressed quadtrees for low-density scenes ($\lambda =$ density)

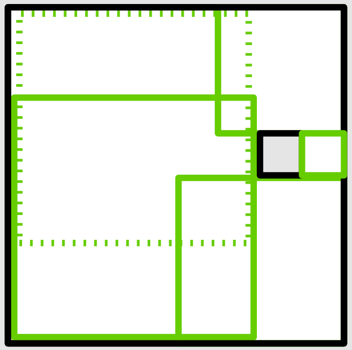
Step 1: replace objects by vertices of bounding boxes

Step 2: construct compressed quadtree for resulting set of points



. . . and put objects back

- number of cells: $O(n)$
- number of objects per cell: $O(\lambda)$



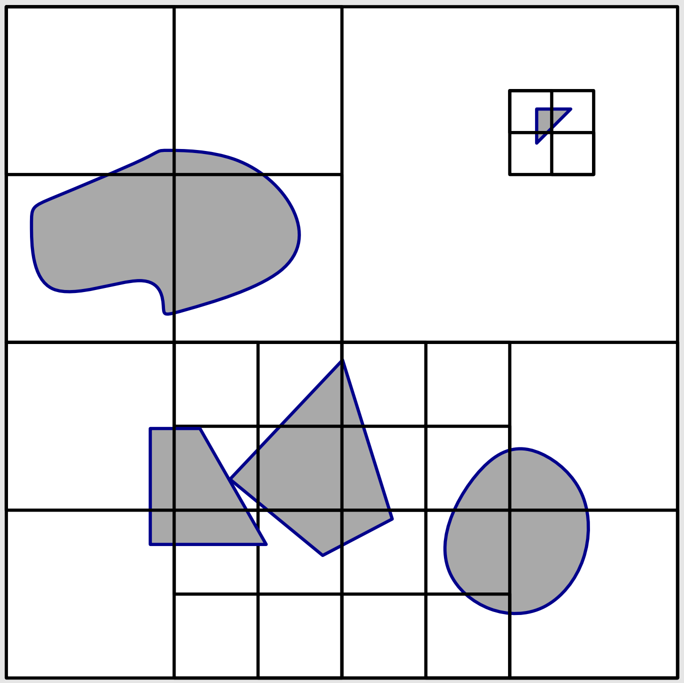
donut can be covered by six squares

Any square not containing bounding-box vertex intersects $\leq 4\lambda$ objects.

Compressed quadtrees for low-density scenes

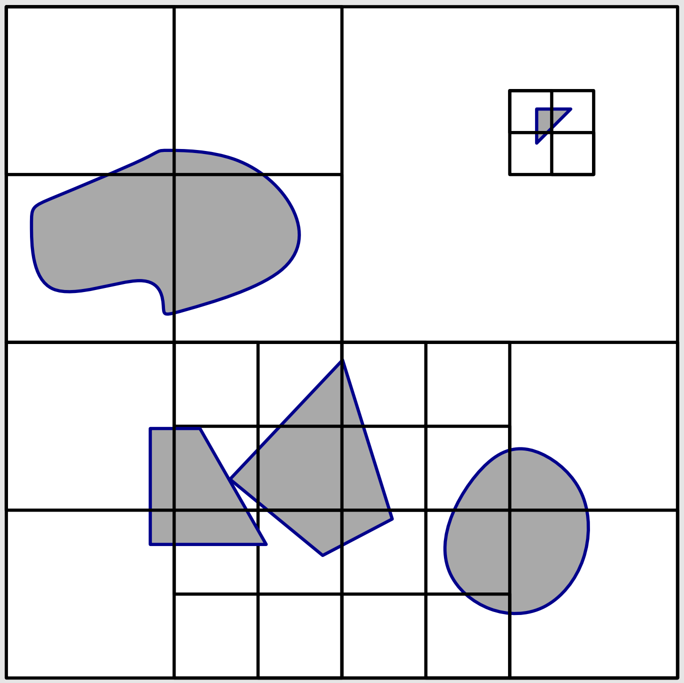
- number of cells: $O(n)$
- number of objects per cell: $O(\lambda)$

$O(\lambda n)$ fragments



Compressed quadtrees for low-density scenes

$$\left.\begin{array}{l}\bullet\ \text{number of cells: } O(n) \\ \bullet\ \text{number of objects per cell: } O(\lambda)\end{array}\right\}\ O(\lambda n) \text{ fragments}$$

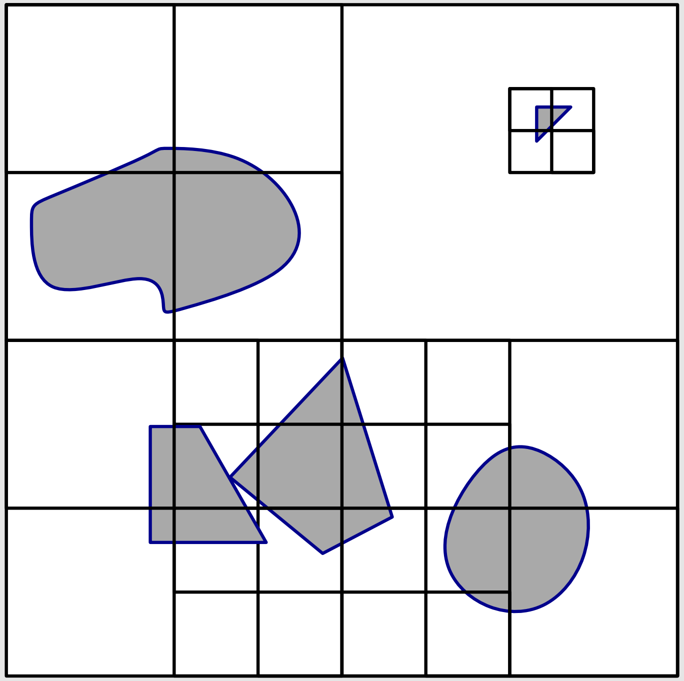


Reducing the number of fragments:

- sort bb-vertices in Z-order
- only keep every λ-th bb-vertex

Compressed quadtrees for low-density scenes

$$\left.\begin{array}{l}\bullet\ \text{number of cells: } O(n)\\[1em] \bullet\ \text{number of objects per cell: } O(\lambda)\end{array}\right\}\ O(\lambda n) \text{ fragments}$$



Reducing the number of fragments:

- sort bb-vertices in Z-order

NW	NE
SW	SE

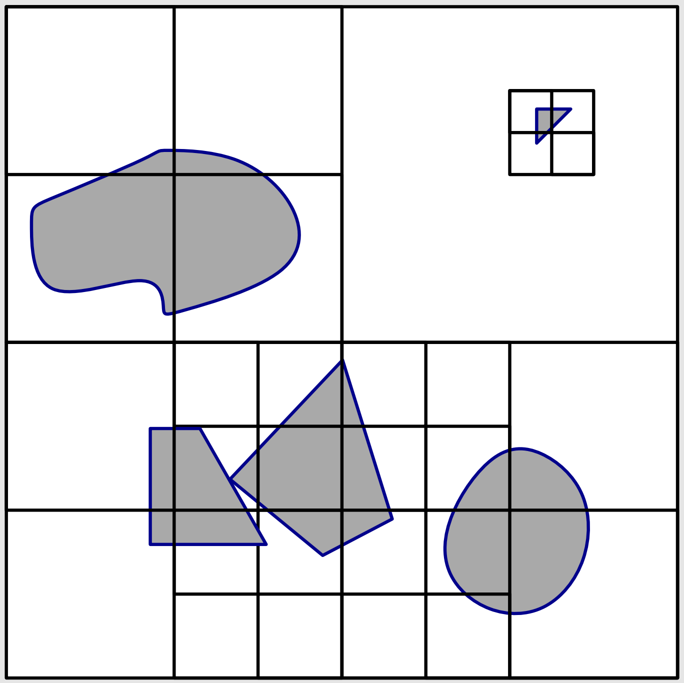
NW $\prec$ NE $\prec$ SW $\prec$ SE

- only keep every λ-th bb-vertex

Compressed quadtrees for low-density scenes

- number of cells: $O(n)$
- number of objects per cell: $O(\lambda)$

$\Big\}$ $O(\lambda n)$ fragments



Reducing the number of fragments:

- sort bb-vertices in Z-order

NW	NE
SW	SE

NW $\prec$ NE $\prec$ SW $\prec$ SE

- only keep every λ-th bb-vertex

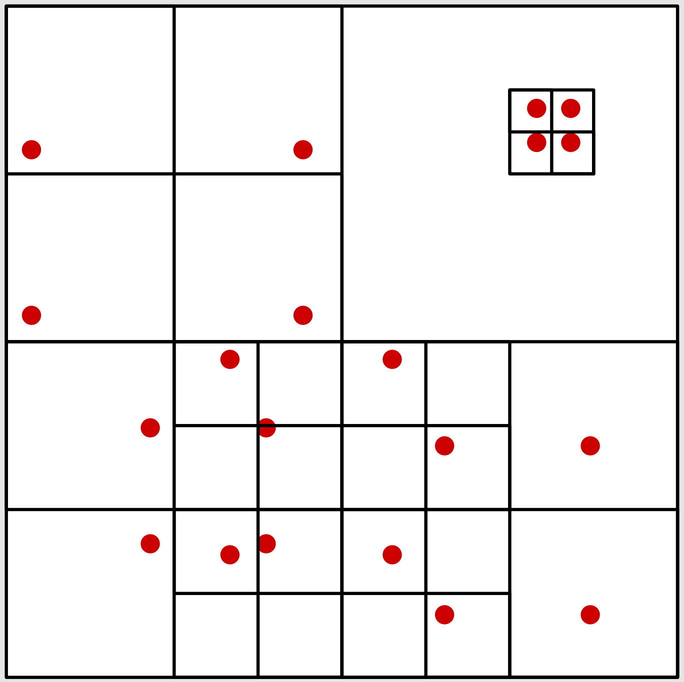
at most $2\lambda - 1$ bb-vertices per cell $\Longrightarrow$ still $O(\lambda)$ objects / cell

Compressed quadtrees for low-density scenes

- number of cells: $O(n)$
- number of objects per cell: $O(\lambda)$
- but depth can be linear

Compressed quadtrees for low-density scenes

- number of cells: $O(n)$
- number of objects per cell: $O(\lambda)$
- but depth can be linear

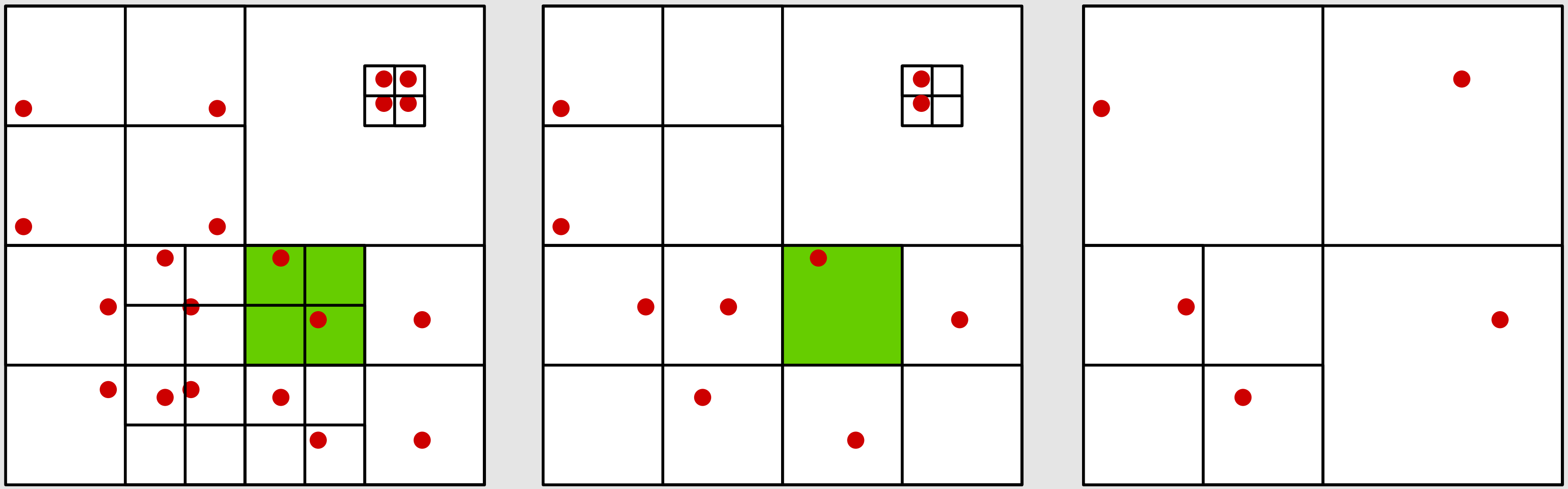


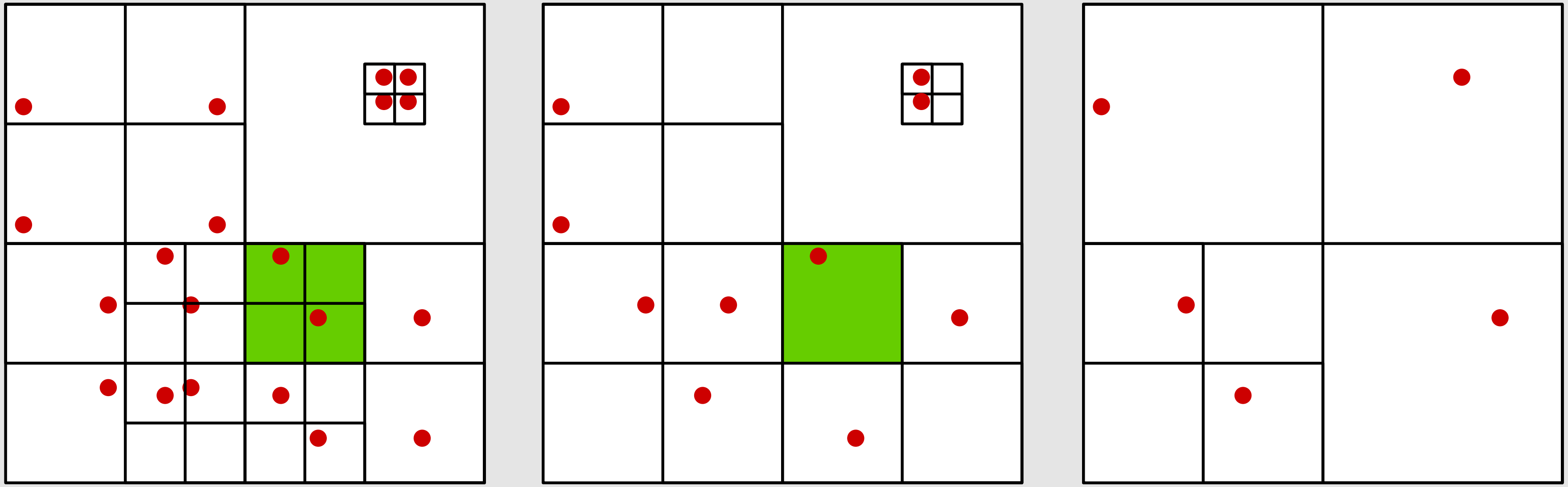
sort bb-vertices in Z-order:

- $P_0 = P = p_1, p_2, p_3, p_4, p_5, \ldots, p_{4n}$
- $P_1 = p_1, p_3, p_5, \ldots$
- $P_2 = p_1, p_5, \ldots$

Construct hierarchy of compressed quadtrees

Compressed quadtrees for low-density scenes





every cell has $O(1)$ children

number of levels is $O(\log n)$

$\Longrightarrow$ point location in $O(\log n)$ time

S: set of n objects in $\mathbb{R}^d$

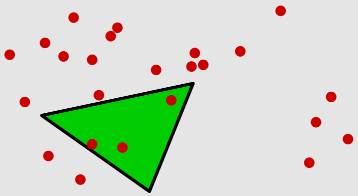
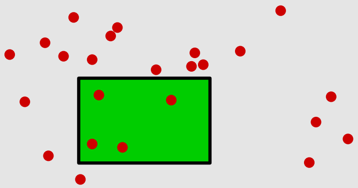
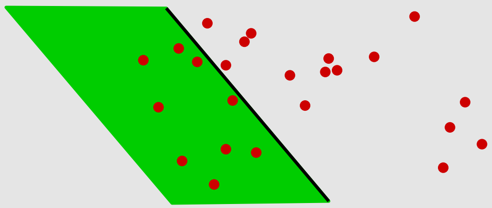
λ: density of S

Theorem: There is a compressed quadtree hierarchy of $O(\log n)$ depth and $O(n/\lambda)$ size where any leaf region intersects $O(\lambda)$ objects.

Example: how to use canonical directions

Using canonical directions: exact range searching

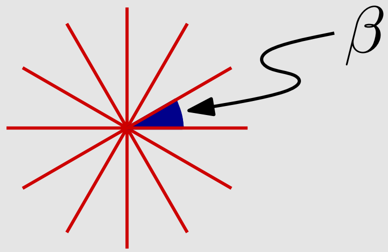
2D range searching: report all points inside a query range

	range	storage	query time
	triangle	n^2	$\log n + k$
	axis-parallel rectangle	$n \log n$	$\log n + k$
	halfplane	n	$\log n + k$

what if the range is a *fat* triangle?

Using canonical directions: exact range searching

range is β-fat triangle: all angles $\geq \beta$

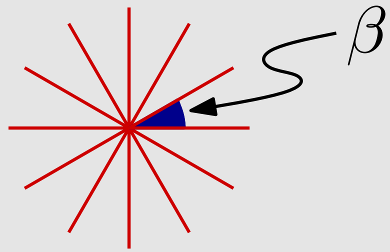


canonical directions: $0, \beta, 2\beta, \ldots, 2\pi - \beta$

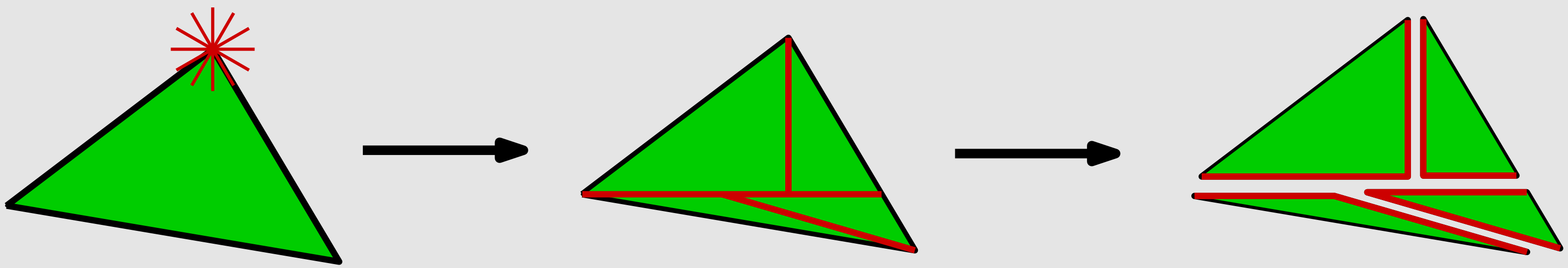
Using canonical directions: exact range searching

range is β-fat triangle: all angles $\geq \beta$

canonical directions: $0, \beta, 2\beta, \ldots, 2\pi - \beta$



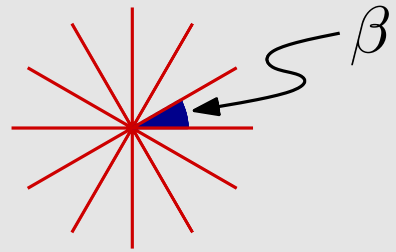
partition query triangle into four sub-triangles using canonical directions



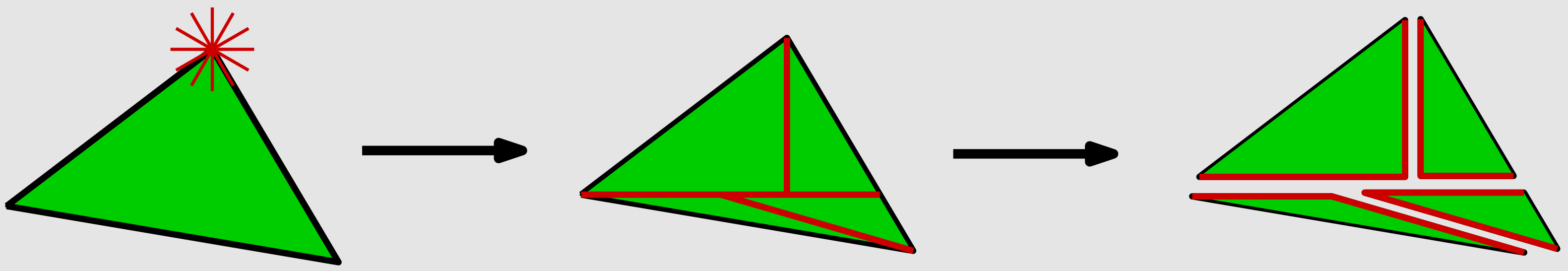
Using canonical directions: exact range searching

range is β-fat triangle: all angles $\geq \beta$

canonical directions: $0, \beta, 2\beta, \ldots, 2\pi - \beta$

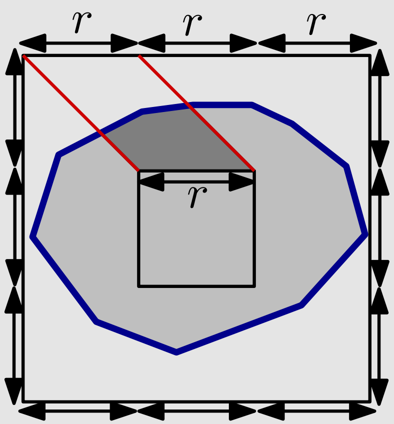


partition query triangle into four sub-triangles using canonical directions



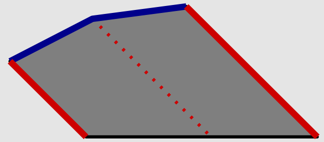
subtriangle is intersection of an "axis-parallel" rectangle and half-plane

$\Longrightarrow$ combine halfplane and rectangle structures into single structure:
storage $O(n \log^2 n)$, query time $O(\log^3 n)$



[Aronov-dB-Gray '06]

cover β-fat convex object by "towers" using $O(1/\beta^2)$ canonical directions



Improved bounds for

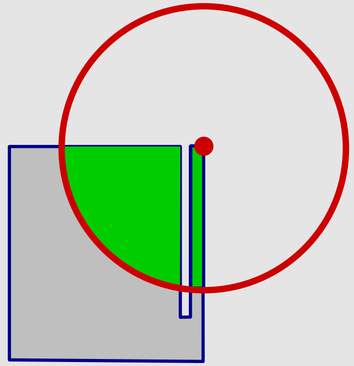
- ray shooting: storage $O(n^{2+\varepsilon})$, query $O(\text{polylog } n)$
- intersection searching: storage $O(n^{3+\varepsilon})$, query $O(\text{polylog } n + k)$
- range searching: storage $O(n \text{ polylog } n)$, query $O(\text{polylog } n + k)$

geometry-sentive analysis

- analysis not only as function of input size n,
  but also as function of certain geometry-describing parameters
- by now, nice theory that leads to
  - algorithms with much better scale-up behavior
    ... if geometry parameters are indeed constant
  - often simpler, more practical algorithms
  - better prediction of efficiency in practice ??

Why local fatness ?

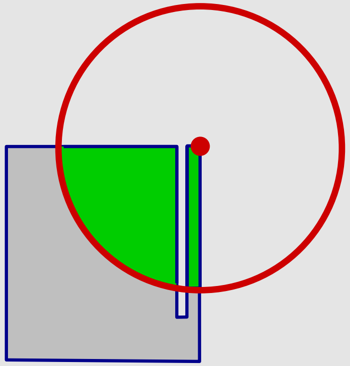
Standard fatness definition:
For any disk D with center in o, we have:



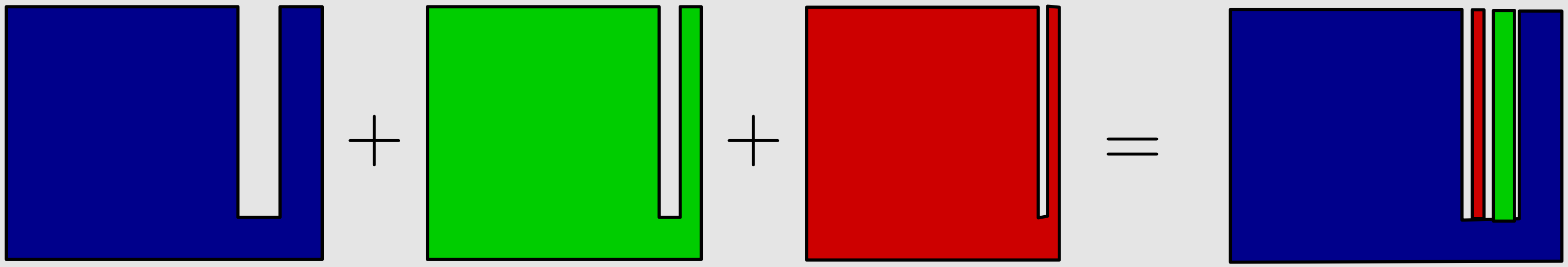
$$\text{area}(D \cap o) \geq \gamma \cdot \text{area}(D)$$

Why local fatness ?

Standard fatness definition:
For any disk D with center in o, we have:

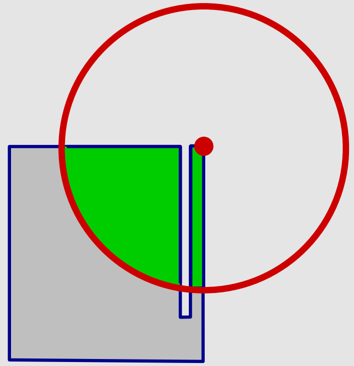


$$\text{area}(D \cap o) \geq \gamma \cdot \text{area}(D)$$



Why local fatness ?

Standard fatness definition:
For any disk D with center in o, we have:



$$\text{area}(D \cap o) \geq \gamma \cdot \text{area}(D)$$