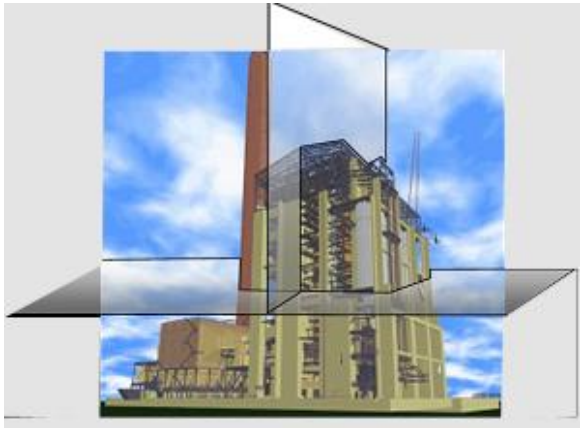
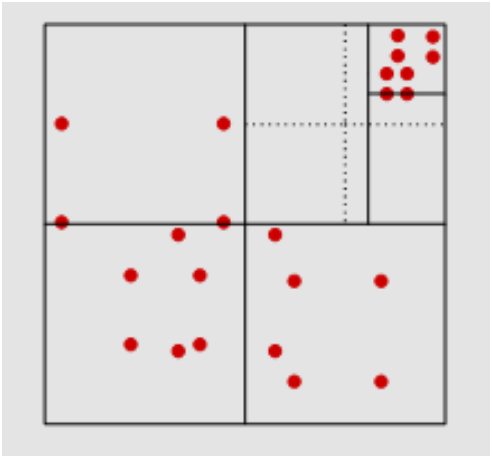
Bsp Trees for Low-Density Scenes

Neda Ezzati
University of Yazd
Spring 1391





Theorem12.5: for any set of n non-intersecting triangles in R^3 a BSP tree of size $O(n^2)$ exists. Moreover, there are congurations for which the size of any BSP is $\Omega(n^2)$.

The $\Omega(n^2)$ might give you the idea that BSP trees are useless in practice. Fortunately this is not the case: in many practical situations, BSP trees perform just fine. The problem is that certain inputs. We would like our analysis to reflect this: it should give different bounds for different types of input.(easy and diffcult input)

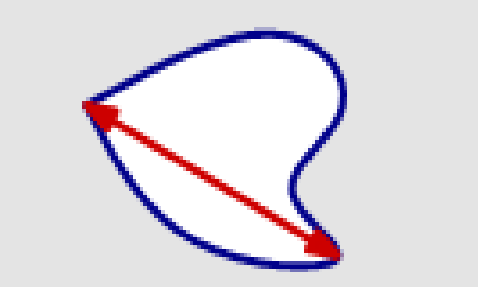
We must introduce another parameter, which distinguishes easy inputs from difficult ones. what are easy inputs?

Intuitively, easy inputs are inputs where the objects are relatively well separated, whereas difficult inputs have many objects packed closely together.

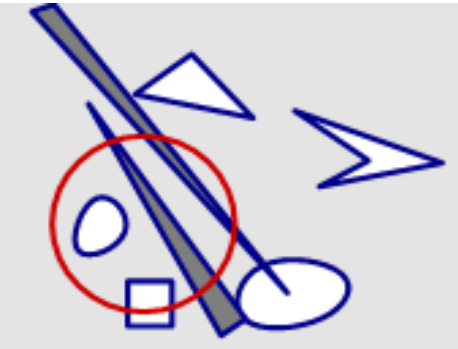
We can recognize type of input by parameter density of a scene.

Density of a scene:

density distribution parameter for sets of objects: density

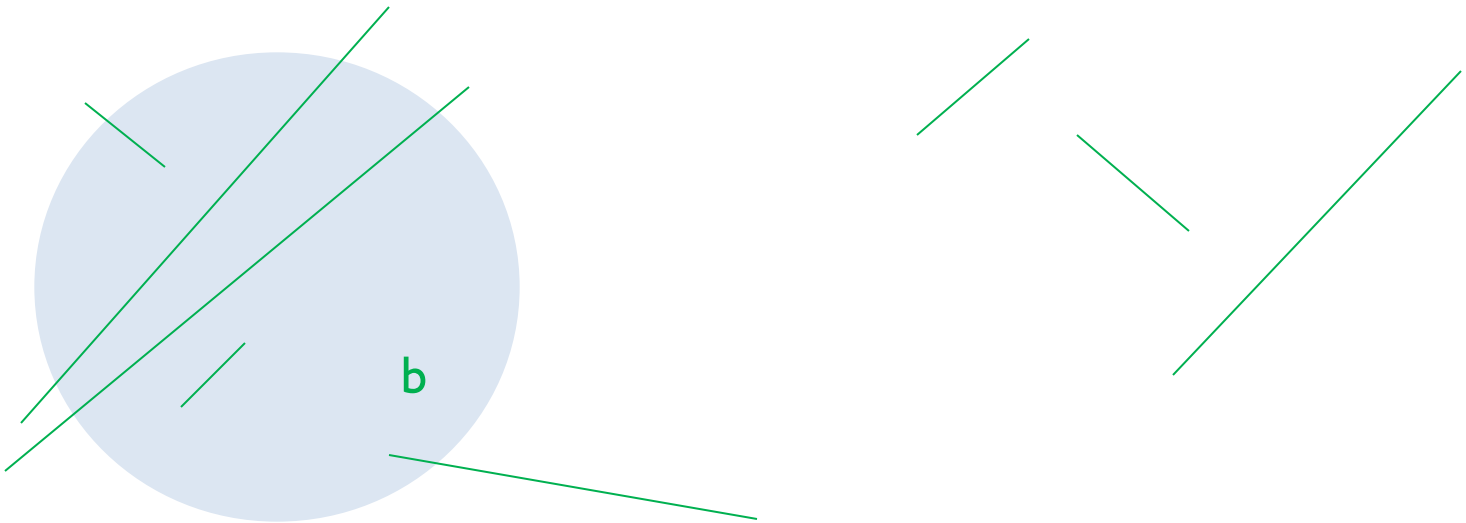


diam(o):= diameter of object o

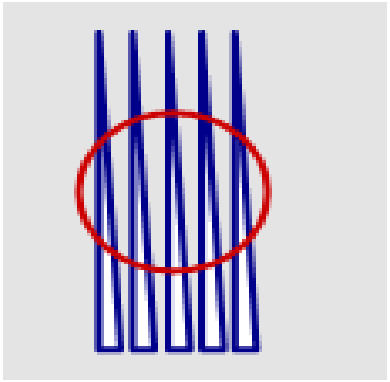


density of set S of objects:
minimum λ such that for any ball b :
#{$o \in S$:o intersects b, diam(o)$\geq$ diam(b)} $\leq \lambda$

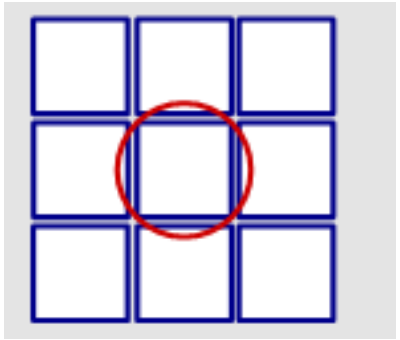
Examples:



A set of eight segments with density 3



Set of n disjoint triangles can have density n



Any set of n disjoint squares in the plane has density at most 9.

Result:

if the density is low then the objects are reasonably well separated, and if the density is high then there are regions with many objects close together.

Problem: we want to show that if the density is low then we can find a small BSP.

randomized algorithm of the previous section is not good, because even for inputs of low density, it may produce a BSP whose expected size is quadratic.

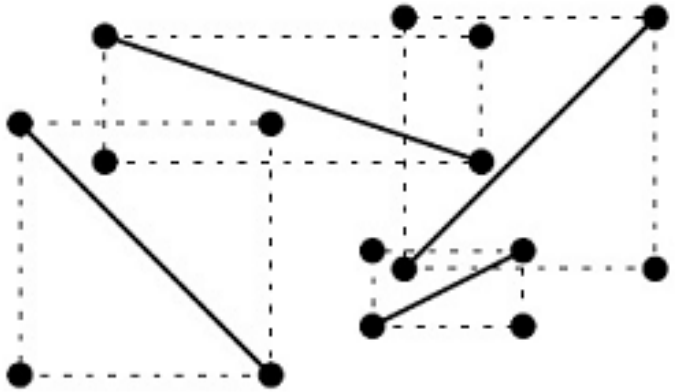
New Algorithm

The idea behind the algorithm:

Let S be a set of objects in R^2,S can contain segments, discs,triangles, etc. and let λ be the density of S.

each object $o \epsilon S$, a small set of points, we call them guards.

Let bb(o) denote the bounding box of o,that is the smallest axis aligned rectangle that contains o. The guards that we define for o are simply the four vertices of bb(o). Let G(S) be the multiset of 4n guards.



for any square δ, the number of objects intersecting δ is not much more than the number of guards inside δ .

The next lemma gives only an upper bound on the number of objects intersecting a square, not a lower bound: it is very well possible that a square contains Many guards without intersecting a single object.

Lemma 12.6

Any axis-parallel square that contains k guards from G(S) in its interior intersects at most k+4λobjects from S.

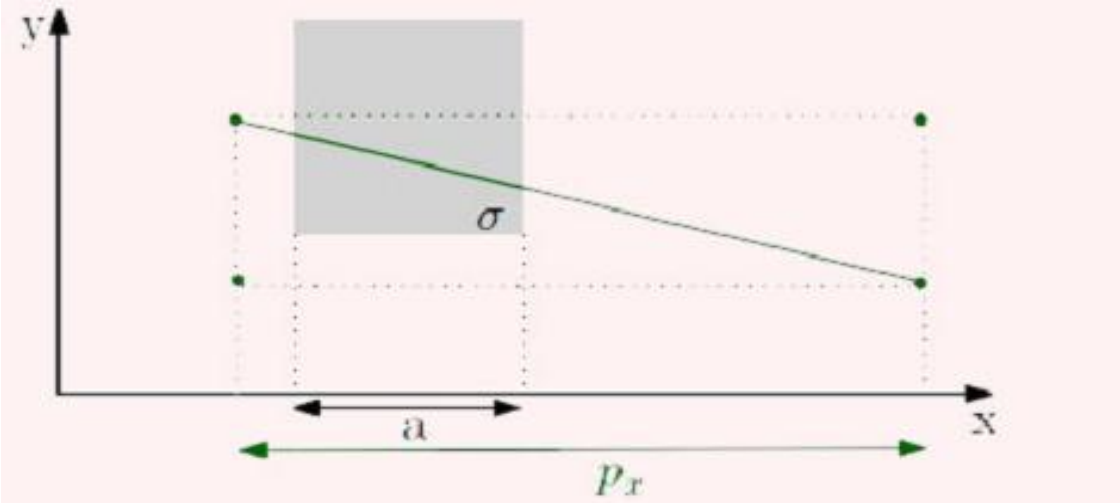
Proof:

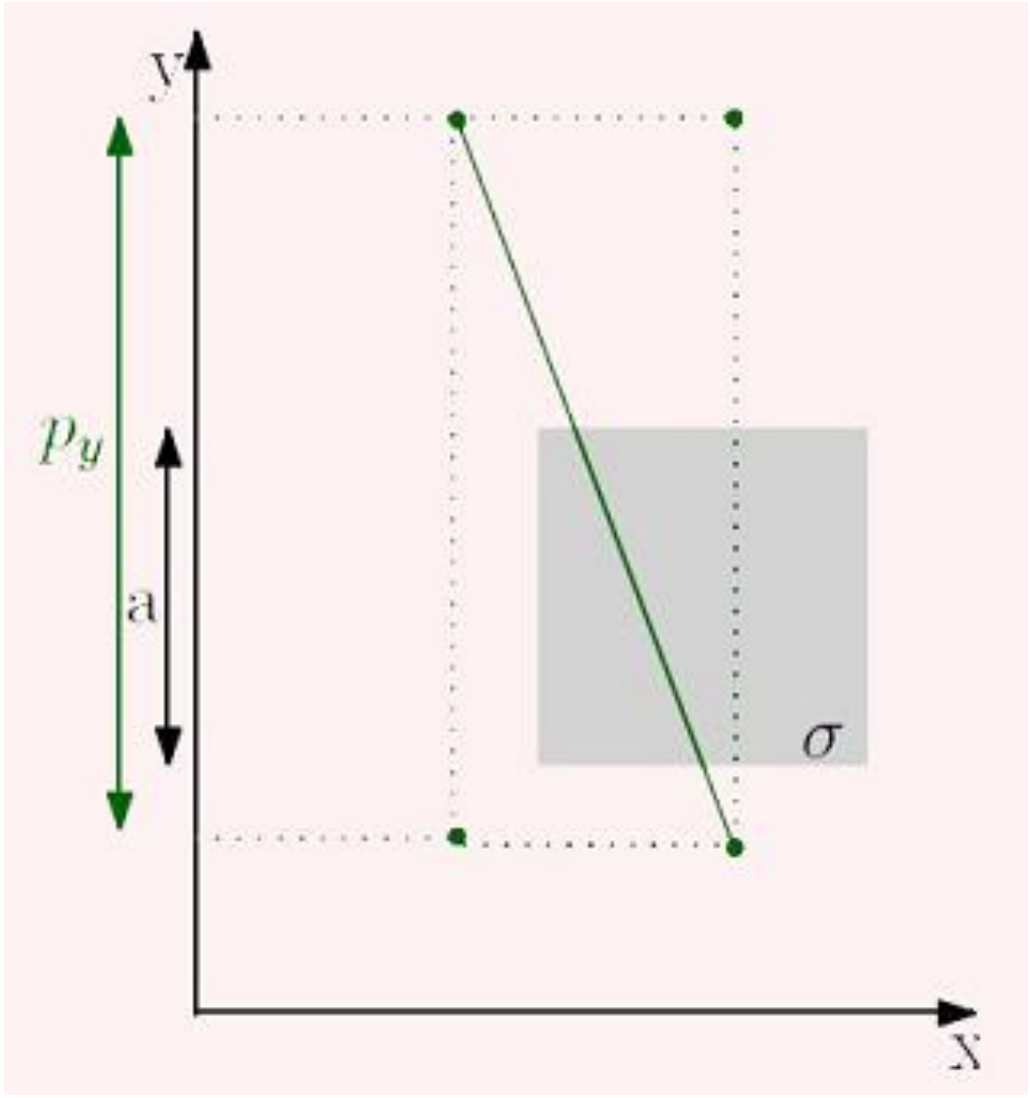
Obviously there are at most k objects that have a guard Inside δ.

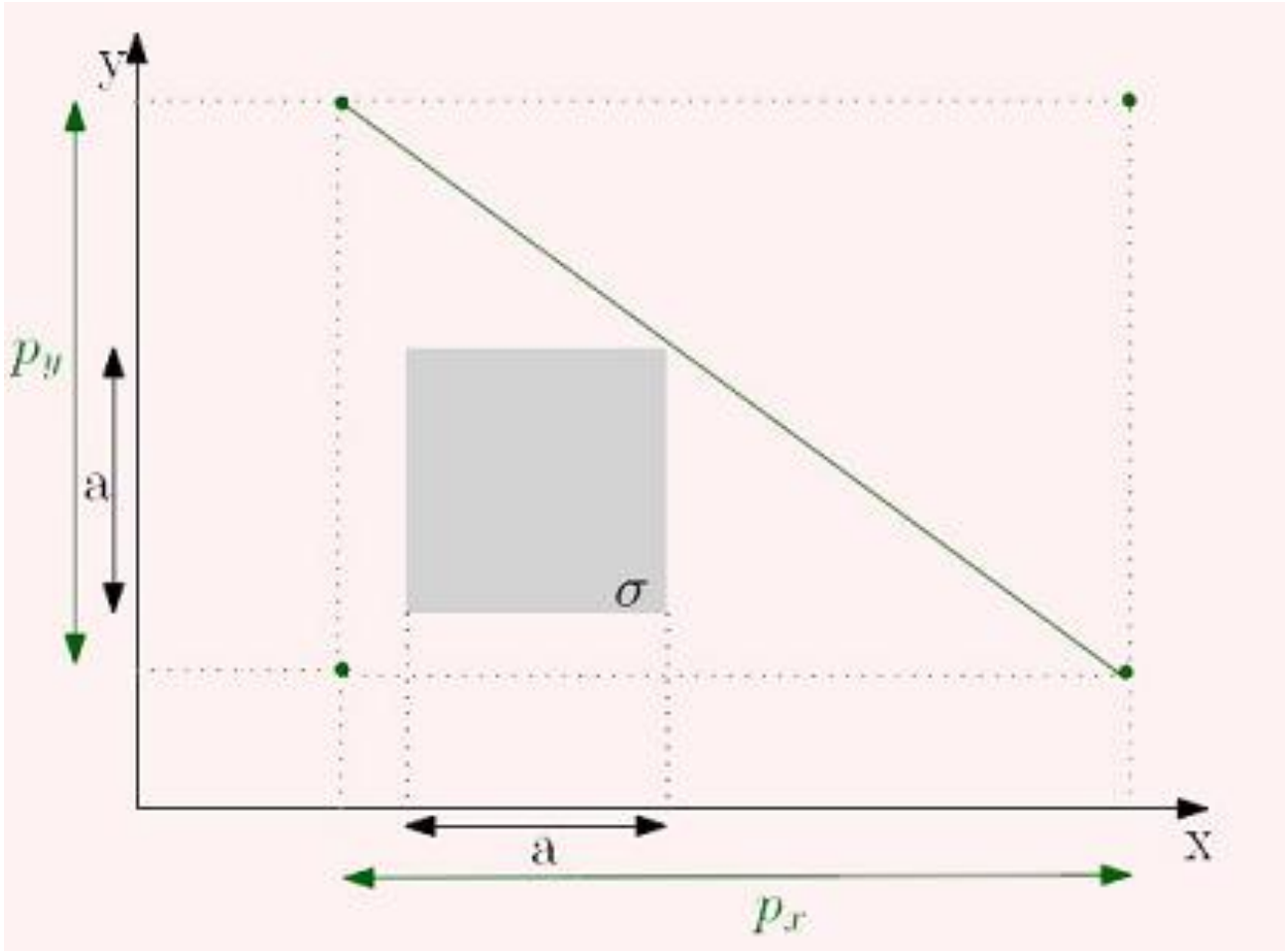
The density of $\acute{s}$ is at most $\lambda, s \subset s$. We have to show that at most 4λobjects from $\acute{s}$ can intersect δ.

Three states for an object $o \in \acute{s}$ intersects δ

1) The projection of bb(o) onto the x-axis contains the projection of s onto the x-axis
2) The projection of bb(o) onto the y-axis contains the projection of s onto the y-axis
3) Both of 1 and 2




y
p_y
a
σ
x

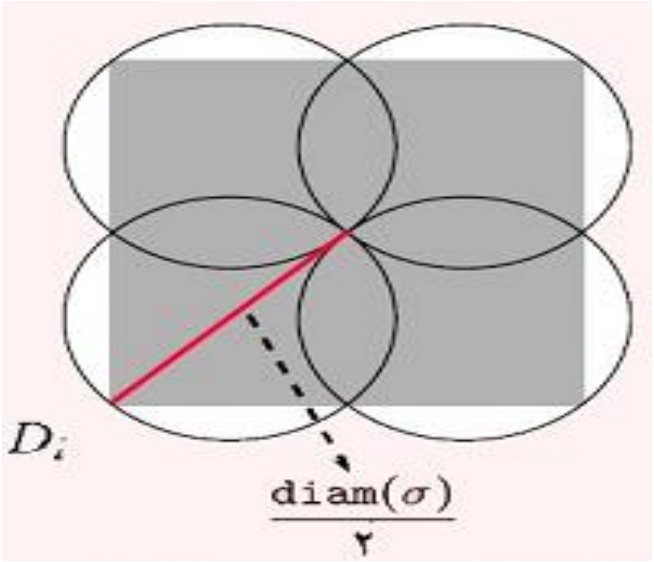

y
p_y
a
σ
a
X
p_x

Proof (continue)

For three states we have $\text{diam}(o) \geq \text{diam}(\delta) / \sqrt{2}$

Now cover s with four discs $D_1, \ldots, D_4$ of diameter $\text{diam}(\delta) / \sqrt{2}$

The object o intersects at least one of these discs, D_i . We charge o to D_i .



We have:

$$\text{diam}(o) \geq \text{diam}(\delta) / \sqrt{2} > \text{diam}(\delta) / 2 = \text{diam}(D_i)$$

Since $\acute{S}$ has density at most λ, each D_i is charged at most λ times.Hence, δ intersects at most 4λ objects from $\acute{S}$.

Adding the at most k objects not in $\acute{S}$.

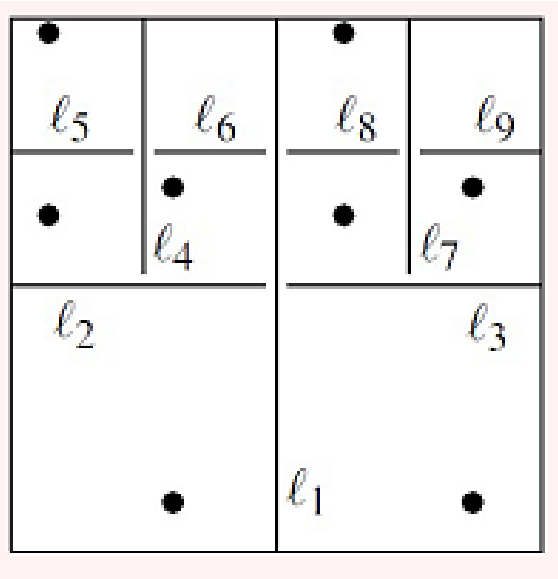
We find that δ intersects at most k+4λ objects from S.

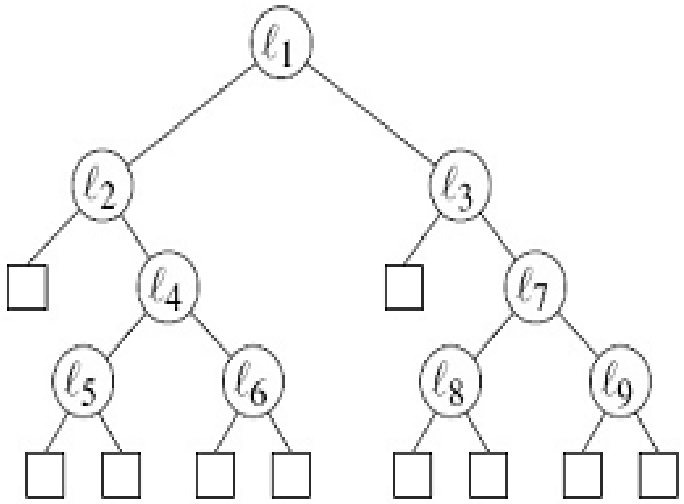
Lemma 12.6 suggests the following two-phase algorithm to construct a BSP.

In the frst phase, we recursively subdivide U into squares until each square contains at most one guard in its interior.

By Lemma 12.6, each leaf region in the resulting subdivision intersects only a few objects at most $1+4\lambda$

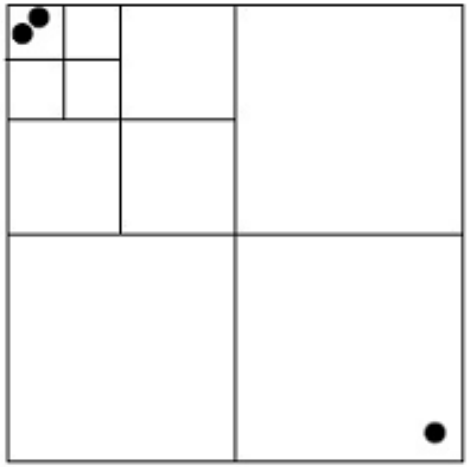
Second phase of the algorithm then partitions each leaf region further, until all objects are separated.





one problem in new algorithm

The number of leaf regions can be very large.This happens for example when two guards lie very close together near a corner of the initial square U.Therefore we modify the frst phase of the algorithm to guarantee it producesa linear number of leaf regions.



Two modications for improve algorithm

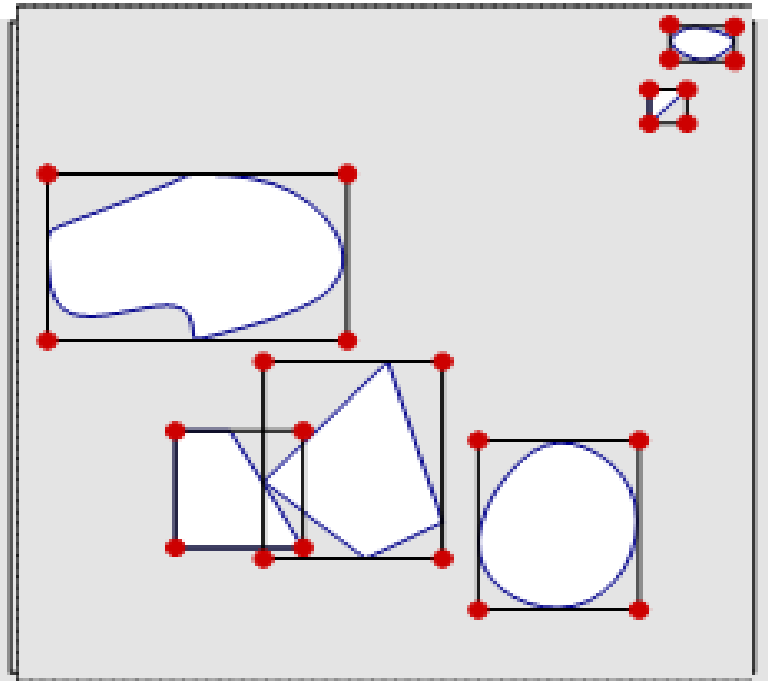
The frst modication is that we stop when the region contains k or fewer guards, for some suitable parameter $K \geq 1$.

The second modication is: Consider the four quadrants of δ. If at least two of them contain more than k guards in their interior, then we proceed as before by applying a

quadtree split.

If there is exactly one quadrant, say δ , with more than k guards in it, we perform a shrinking step.

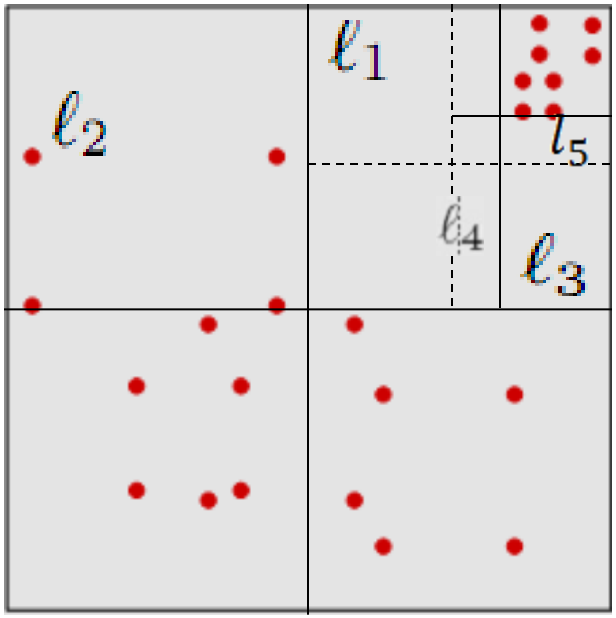
Step 1: replace objects by vertices of bounding boxes (=guards)

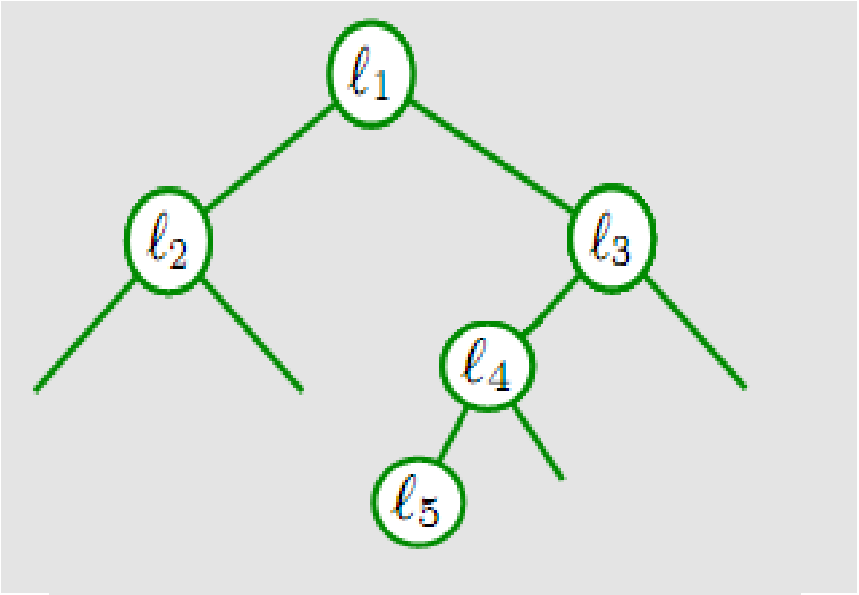


Step 2: construct BSP for resulting set of points

- if at least two quadrants contain points: split into quadrants

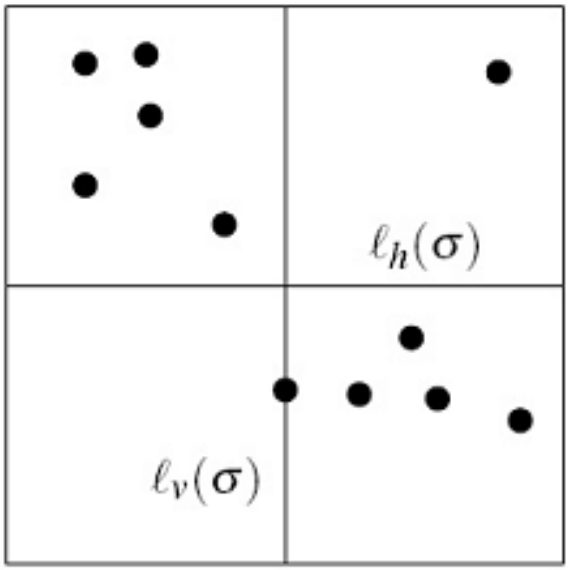
- if exactly one quadrant contains points: shrink



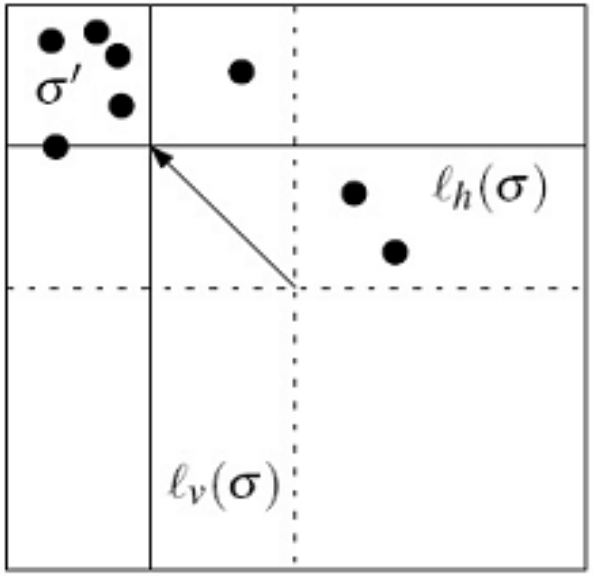


continue recursively until interior of all cells empty

We shrink δ by moving its bottom right corner diagonally to the north-west until at least k guards are outside the interior of δ.



quadtree split



shrinking step

Algorithm PHASE1(σ, G, k)

Input. A region σ, a set G of guards in the interior of σ, and an integer $k \geqslant 1$.

Output. A BSP tree $\mathcal{T}$ such that each leaf region contains at most k guards.

1. **if** card$(G) \leqslant k$
2. **then** Create a BSP tree $\mathcal{T}$ consisting of a single leaf node.
3. **else if** exactly one quadrant of σ contains more than k guards in its interior
4. **then** Determine the splitting lines $\ell_v(\sigma)$ and $\ell_h(\sigma)$ for a shrinking step, as explained above.
5. **else** Determine the splitting lines $\ell_v(\sigma)$ and $\ell_h(\sigma)$ for a quadtree split, as explained above.
6. Create a BSP tree $\mathcal{T}$ with three internal nodes; the root of $\mathcal{T}$ stores $\ell_v(\sigma)$ as its splitting line, and both children of the root store $\ell_h(\sigma)$ as their splitting line.
7. Replace each leaf μ of $\mathcal{T}$ by a BSP tree $\mathcal{T}_\mu$ computed recursively on the region corresponding to μ and the guards inside that region.
8. **return** $\mathcal{T}$

Lemma 12.7

PHASE1(U,G(S),k) produces a BSP tree with O(n/k) leaves, where each leaf region intersects at most k+4λ objects.

Proof: first part

(PHASE1(U,G(S),k) produces a BSP tree with O(n/k) leaves)

- number of leaves = number of internal nodes + 1

- N(m): maximum number of internal nodes in a BSP tree created

- by PHASE1(δ,G,k) when card(G) = m.

- If m$\leq$ k, no splits are performed, and so N(m) = 0 in this case.

- Otherwise, a quadtree split or a shrinking step is applied to . This results in three internal nodes, and four regions in which we recurse.

- $m_1, \ldots, m_4$: the numbers of guards in the four regions,

I :={i : 1$\leq$ i$\leq$4 and mi > k}

$$N(m) \leq 3 + \sum_{i \in I} N(m_i)$$

We will prove by induction that N(m)$\leq$max(0, (6m/k)-3).

This is obviously true for m≤k, and so we now assume that m > k.

A guard can be in the interior of at most one region, which means that $\sum_{i \in I} m_i \leq m$

If at least two quadrants of σ contain more than k guards, then card(I) ≥2 and we have

$$N(m) \leq 3 + \sum_{i \in I} N(m_i) \leq 3 + \sum_{i \in I} \left(\left(\frac{6m_i}{k}\right) - 3\right)$$

$$\leq 3 + \left(\sum_{i \in I} \left(\frac{6m_i}{k}\right)\right) - card(I).3 \leq \frac{6m}{k} - 3$$

If none of the quadrants contains more than k guards, then the four regions are all leaf regions and N(m)= 3. Together with the assumption m > k, this implies N(m)≤ (6m/k)−3. The remaining case is where exactly one quadrant contains more than k guards. In this case we do a shrinking step. Because of the way a shrinking step is performed, a shrunk quadrant contains fewer than m−k guards and the other resulting regions contain at most k guards. Hence, in this case we have

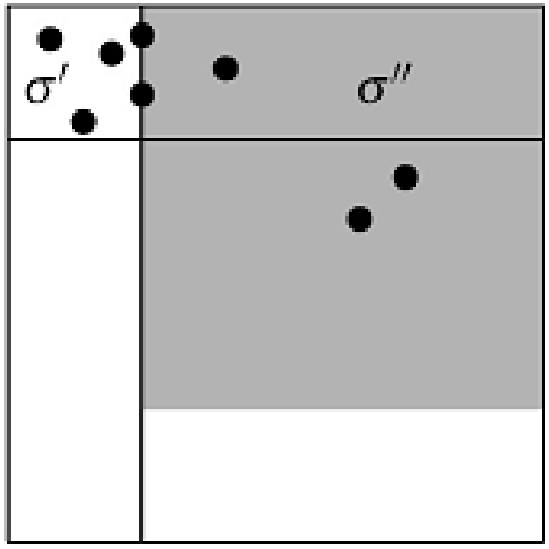
$$N(m) \leq 3 + N(m-k) \leq 3 + \left(\frac{6(m-k)}{k} - 3\right) \leq \frac{6m}{k} - 3$$

Proof: second part

(each leaf region intersects at most k + 4λ objects)

If a leaf region is a square: Lemma 12.6 implies that it intersects k + 4λ objects.

leaf regions are not square: a non-square leaf region must have been produced in a shrinking step.We use this step and prove it.



How can we use Lemma 12.7 to construct BSP tree?

the larger k is, the fewer leaf regions we will have. On the Other hand, a larger k will also mean more objects per leaf region.

Setting k := λ will do this. the number of leaf regions will decrease by a factor λ while the maximum number of objects per leaf region only goes from 1+4λ to 5λ.

One problem: we do not know , the density of the input scene, and so we cannot use it as a parameter in the algorithm.

Solution:

We guess a small value for λ, say $\lambda = 2$. Then we run PHASE1 with our guess as the value of k, and we check whether each leaf region in the resulting BSP tree intersects at most 5k objects. If so, we proceed with the second phase of the algorithm; otherwise, we double our guess and try again. This leads to the following algorithm.

Algorithm LOWDENSITYBSP2D(S)

Input. A set S of n objects in the plane.

Output. A BSP tree $\mathcal{T}$ for S.

1. Let $G(S)$ be the set of $4n$ bounding-box vertices of the objects in S.
2. $k \leftarrow 1$; *done* $\leftarrow$ **false**; $U \leftarrow$ a bounding square of S
3. **while not** *done*
4. **do** $k \leftarrow 2k$; $\mathcal{T} \leftarrow$ PHASE1$(U, G(S), k)$; *done* $\leftarrow$ **true**
5. **for** each leaf μ of $\mathcal{T}$
6. **do** Compute the set $S(\mu)$ of object fragments in the region of μ.
7. **if** card$(S(\mu)) > 5k$ **then** *done* $\leftarrow$ **false**
8. **for** each leaf μ of $\mathcal{T}$
9. **do** Compute a BSP tree $\mathcal{T}_\mu$ for $S(\mu)$ and replace μ by $\mathcal{T}_\mu$.
10. **return** $\mathcal{T}$

Theorem 12.8

for any set S of n disjoint line segments in the plane, there is a BSP of size O(nlog λ), where λ is the density of S.

Proof:

When the while-loop ends in line 7, we have at most k objects at each leaf region.

k^*denote the value of k when we get to line 8. according to Lemma 12.1, each tree $\mathcal{T}_\mu$ has (expected) size O(k^* log k^*) when 2DRANDOMBSP is used in line 9.

there are O(n/k^*) leaf regions, the total size of the BSP tree is

O(nlog k^*). Since $k^* \leq 2\lambda$, this proves the theorem

Effciency of The algorithm for a set of segments

The algorithm that we have just described works very well for segments in the plane: it produces a BSP whose size is O(nlogn) in the worst case and O(n) when the density of the input is a constant.

What happens when we apply this approach to a set of triangles in R^3 ?

Theorem 12.9:

For any set S of n disjoint triangles in R^3 , there is a BSP of size O(n λ), where λ is the density of S.

End