FI MU

Faculty of Informatics
Masaryk University

Linear Binary Space Partitions and Hierarchy of Object Classes

by

Petr Tobola
Karel Nechvíle

FI MU Report Series **FIMU-RS-2003-02**

 April 2003

Linear Binary Space Partitions and Hierarchy of Object Classes

Petr Tobola, Karel Nechvíle [1]

Department of Software Systems and Communications
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno
Czech Republic
{ptx,kodl}@fi.muni.cz

Abstract: We consider the problem of constructing binary space partitions for the set $\mathcal{P}$ of d-dimensional objects in d-dimensional space. There are several classes of objects defined for such settings, which support design of effective algorithms. We extend the existing the de Berg hierarchy of classes [8] by the definition of new classes derived from that one and we show desirability of such an extension. Moreover we propose a new algorithm, which works on generalized λ-low density scenes [20] (defined in this paper) and provides BSP tree of linear size. The tree can be constructed in $O(n \log^2 n)$ time and space, where n is the number of objects. Moreover, we can trade-off between size and balance of the BSP tree fairly simply.

keywords: BSP, tree, partitioning, object, class, hierarchy

1 Introduction

In the past, much attention has been dedicated to the development of algorithms, whose goal is to construct the smallest possible BSP trees. Initially, several heuristic methods were developed (for example [2, 9, 15, 16]), which however could create a tree of excessive size under unfavourable circumstances ($\Omega(n^2)$ on the plane and $\Omega(n^3)$ in $\mathbb{R}^3$ space). The first provable bounds were obtained by Paterson and Yao [13, 12]. They showed [13] that the optimal size of BSP in the worst case is $\Theta(n^2)$ in $\mathbb{R}^3$ space and $O(n \log n)$ on a plane. The next result from these authors [12] was optimal sized BSP

[1]Support was provided by the grants GACR 201/98/K041, Grant Agency of Czech Republic and MSM 143300004, Ministry of Education, Czech Republic.

algorithm for the set of orthogonal objects with $\Theta(n^{3/2})$ in $\mathbb{R}^3$ space in the worst case and $\Theta(n)$ on a plane in the worst case.

It was observed that many practical scenes behave reasonably and enable effective processing. Several attempts to specify object sets of such scenes have been made to this time. Recently, de Berg et al. [7] have investigated the common properties of the realistic scenes where effective algorithms can be used. A hierarchy of the known object classes (see Figure 1) has been composed and matched by them with realistic input models. This could simplify the design of algorithms, which are provably efficient.

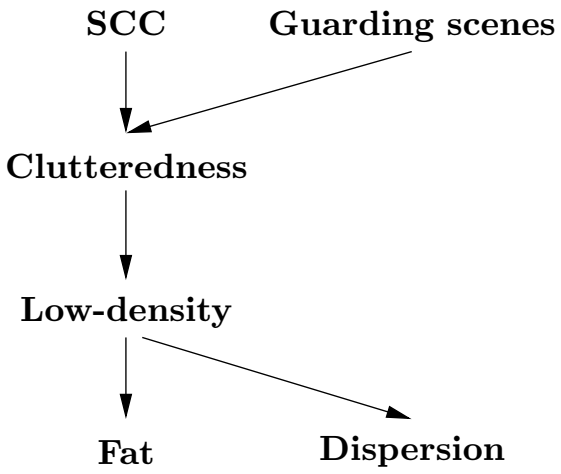

Figure 1: de Berg Hierarchy of object models.

Although the de Berg hierarchy includes a large number of scenes, there still remains a large spectrum of simple and potentially practical scenes, which do not match it. We will outline a few particular examples of such scenes and then we will try to emphasize their significant common features.

A picket fence includes many long and thin pickets standing in a line. This scene does not fit within the de Berg hierarchy because the pickets are located tightly one after another. Similar results can be obtained if we consider a shoal of long and thin fishes, radiator ribs, suits in a wardrobe, books in a library, a cluster of fluorescent lamps, or a thick forest.

It seems that the real world scenes frequently contain groups of similar objects, which usually appear in (isolated) clusters and these objects usually differ only in their position (and possibly in scale ratio). Moreover, it is very probable that the described similarity holds for many objects, which lie near to each other. The correctness of this presupposition has been shown by

results of practical implementation. We are going to describe the practical results exactly in the forthcoming paper.

Although the picket fence does not form a set of fat objects, we can transform it to the set of fat objects by simple linear scale transformation. Figure 2 shows an example of such transformation. The picket fence is substituted by a set A of long and thin rectangles. The linear scale transformation along y-axis transforms the set A to the set B of squares (i.e. fat objects).

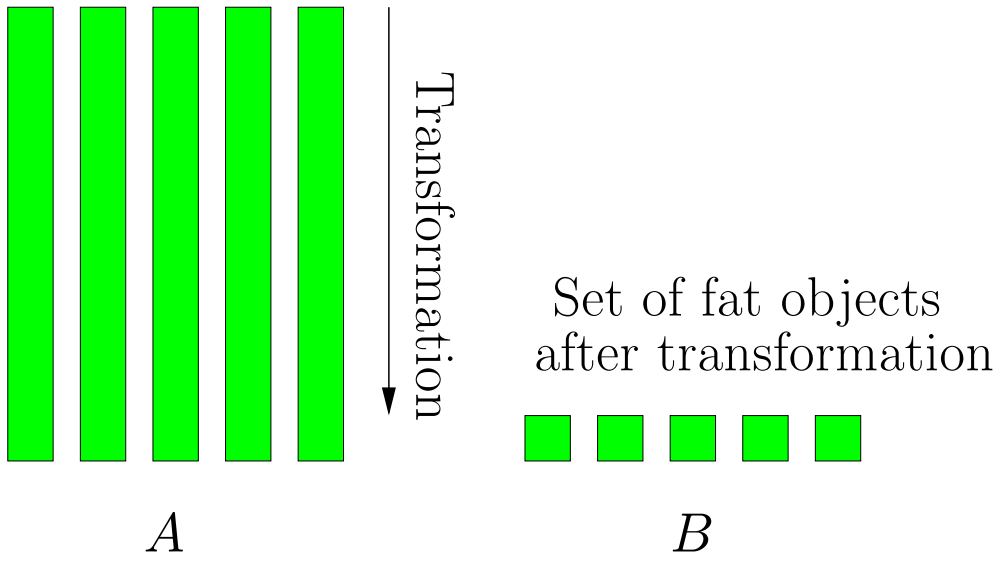

Figure 2: The set A of long thin rectangles is transformed by linear scale transformation onto the set B of squares.

In the following considerations, we come out from the proposed idea of linear transformation. We extend some object classes of the existing de Berg hierarchy by the inclusion of linear transformation.

This paper is organized as follows: In Section 2, we give some necessary definitions needed in the rest of the paper. Section 3 presents the BSP construction method and description of the algorithm. The proofs of algorithm effectivity follows. Section 4 includes concluding remarks.

We have to note that we are going to propose a new paper which will be devoted to practical comparison of the method proposed here and the de Berg algorithm [7].

2 Preliminary

In the following sections, we will use some definitions and denotations introduced here. We start with description of neighborhoods, then we explore the de Berg hierarchy of object classes and we conclude with a useful tool – the Cutting Lemma.

Denotation 2.1: Let p be an object in $\mathbb{R}^d$. The volume of p is denoted $\mathrm{vol}(p)$. The *bounding box* of p (further $\mathrm{bb}(p)$) is the smallest axis-aligned box that completely contains p. The *bounding hypersphere* of p (further $\mathrm{meb}(p)$) denotes the minimal enclosing ball of p and $\varrho_{meb}(p)$ denotes the radius of the minimal enclosing ball. The *bounding hypercube* of p (further $\mathrm{mec}(p)$) denotes the minimal enclosing hypercube of p and $\varrho_{mec}(p)$ denotes the side length of the minimal enclosing hypercube.

Now, we define the neighborhood of axis aligned hyperrectangles in d-dimensional space. We define the neighborhood of any d-dimensional hyperrectangle as a union of $2d$ neighborhood parts. Each such part is represented by a d-dimensional blunted pyramid. Moreover, we use an array of d functions $f = \langle f_1, ..., f_d \rangle$ to parameterize height of blunted pyramids for any direction. δ is the parameter used in the functions $f_i \in f$.

Definition (Rectangle neighborhood) 2.2: Let r be a hyperrectangle in $\mathbb{R}^d$ with edge vectors $\vec{e_1}, ..., \vec{e_d}$. W.l.o.g. we can suppose that $\vec{e_i} \parallel$ i-axis, otherwise we can select an appropriate relative coordinate system. A two dimensional example is depicted in Figure 3. A point C is the center of the hyperrectangle r. Then $(\delta, f, e_i^{\{1,-1\}}, r)$*-neighborhood of hyperrectangle* r, we will mark it $\Omega(\delta, f, e_i^{\{1,-1\}}, r)$, it is a union of set of points defined as follows:

- $\Omega(\delta, f, e_i^1, r) = \bigcup_{c_i, c_j} (C + \sum_{j=1}^{d; j \neq i} (\pm c_j \vec{e_j}) + c_i \vec{e_i})$
- $\Omega(\delta, f, e_i^{-1}, r) = \bigcup_{c_i, c_j} (C + \sum_{j=1}^{d; j \neq i} (\pm c_j \vec{e_j}) - c_i \vec{e_i})$

where $c_i \in \langle \frac{1}{2}, ..., \frac{1}{2} + f_i \rangle$, $c_j \in \langle 0, ..., f_j \rangle$. $f_k = f_k(\delta, r)$ is non-negative above unlimited function increasing with δ and $f = \langle f_1, ..., f_d \rangle$. The (δ, f, r)*-neighborhood of hyperrectangle* r (we will mark it $\Omega(\delta, f, r)$) is a union

$$\Omega(\delta, f, r)) = \bigcup_{i=1}^{d} \Omega(\delta, f, e_i^{\{1,-1\}}, r)$$

.

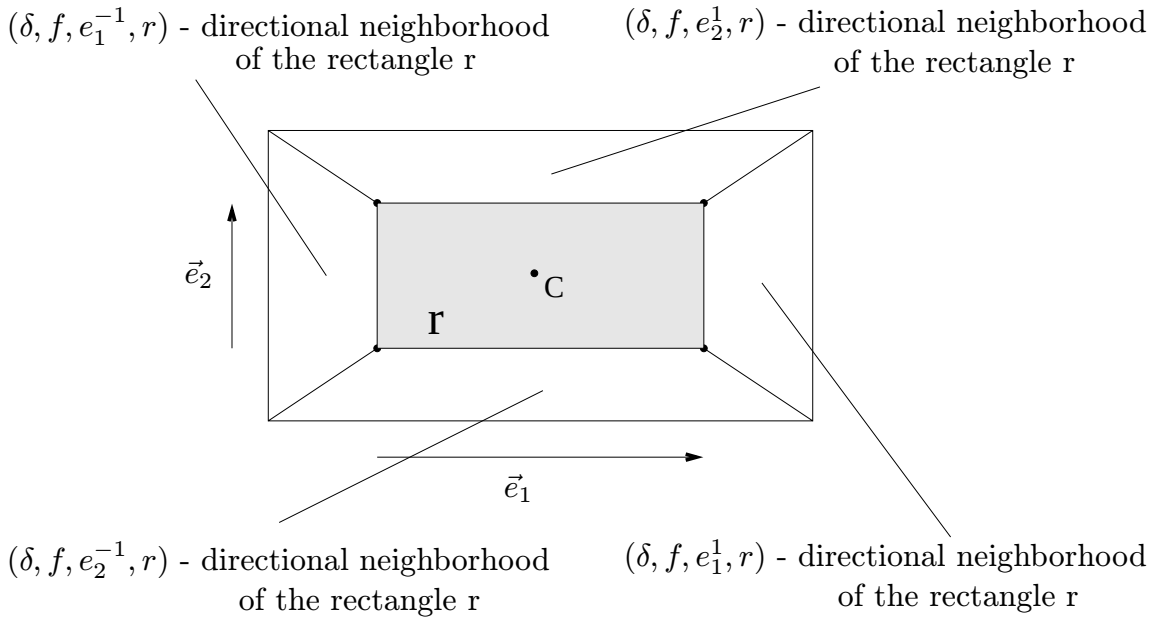

Figure 3: The neighborhood of the rectangle r formed by four blunted cones.

The proposed definition is general and a little bit complicated. Hence, we select two special cases, which are necessary for the remaining chapters – the *simple neighborhood* and the *extended neighborhood*. You can see an example in Figure 4.

Definition (Simple/Extended neighborhood) 2.3:

1. If $f_i = \delta$; $i \in \langle 1, ..., d \rangle$, then we call the neighborhood $\Omega(\delta, f, r)$ *simple neighborhood* and sign it $\Omega_s(\delta, r)$.

2. Let i be the coordinate with maximal edge length $|\vec{e_i}|$. If $f_i = \frac{|\vec{e_j}|}{|\vec{e_i}|}(1+2\delta)$ $i \in \langle 1, ..., d \rangle$, then we call the neighborhood $\Omega(\delta, f, r)$ as an *extended neighborhood* and sign it $\Omega_e(\delta, r)$.

We have already mentioned the de Berg hierarchy of object classes. Our new results are based on this hierarchy and hence we give a bit more detailed description of some its basic classes here.

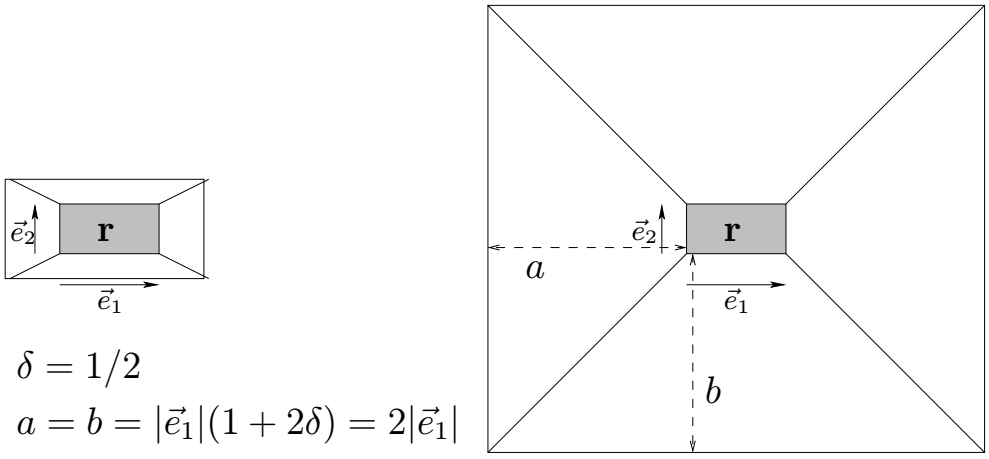

Figure 4: Simple and Extended hyperrectangle neighborhood.

The discrepancy between high worst-case running time of many algorithms and their good practical performance led to study scenes with special properties. The definition of *fatness* was one of the first attempts to determine sets of objects for which we can design effective algorithms. Informally, an object is fat if it does not have any skinny and long parts.

The fatness has been used in many areas of computational geometry (range searching [11] robot motion planing [19], or computing depth order [1]). Recently, de Berg et al. have introduced a hierarchy of object classes [8] where the *fatness* (defined below) represents the most special class. Other models of this hierarchy are the *low density*, the *clutteredness*, and the *simple-cover complexity*.

Definition (Fatness) 2.4: Let $p \subseteq \mathbb{R}^d$ be an object and let β be a constant such that $0 \leq \beta \leq 1$. Define $\mathrm{U}(p)$ as the set of all balls centered inside p which boundary intersects p. We say that the object p is *β-fat* if for all balls $B \in U(p), vol(p \cap B) \geq \beta vol(B)$. The *fatness* of p is defined as the maximal β for which p is β-fat. (Figure 5a)

The low-density model has been introduced by van der Stappen [20]. The low-density forbids any hypercube C to be intersected by many objects which side length of minimal-enclosing-hypercube is at least as large as side length of C.

Definition (Low-density) 2.5: Let $\mathcal{P} = \{p_1, ..., p_n\}$ be a set of objects in $\mathbb{R}^d$ and let $\lambda \geq 1$ be a parameter. We call $\mathcal{P}$ a λ-*low density* scene if for any hypercube C, the number of objects $p_i \in \mathcal{P}$ with $\varrho_{mec}(p_i) \geq$ radius(C) that intersect C is at most λ. The *density* of $\mathcal{P}$ is defined to be the smallest λ for which $\mathcal{P}$ is a λ-low-density scene (Figure 5b).

The model of clutteredness has been introduced by de Berg [7]. Intuitively, the scene is uncluttered if any long and thin object is either alone or surrounded by reasonable number of other smaller objects.

Definition (Clutterredness) 2.6: Let $\mathcal{P}$ be a set of objects in $\mathbb{R}^d$ and let $\kappa \geq 1$ be a parameter. We call $\mathcal{P}$ a κ-*cluttered scene*, if any hypercube which interior does not contain a vertex of one of the bounding boxes of the objects in $\mathcal{P}$ is intersected by at most κ objects in $\mathcal{P}$. The *clutter factor* of a scene is the smallest κ for which it is κ-cluttered (Figure 5c).

The last model of de Berg hierarchy is simple-cover complexity introduced by Mitchell et al. [10]. Given a scene $\mathcal{P}$, we call a ball δ-*simple* if it intersects at most δ objects in $\mathcal{P}$.

Definition (Simple-cover complexity) 2.7: Let $\mathcal{P}$ be a set of objects in $\mathbb{R}^d$, and let $\delta > 0$ be a parameter. The δ-*simple-cover* for $\mathcal{P}$ is a collection of δ-simple balls whose union covers the bounding box of $\mathcal{P}$. We say that $\mathcal{P}$ has *(s, δ)-simple-cover* complexity if there is a δ-simple-cover for $\mathcal{P}$ of cardinality sn (Figure 5d).

Recently, de Berg et al. has proposed the model of *Guarding scenes* [4, 5]. It is direct extension of *Clutteredness*. Moreover, the model of *Dispersion* [14, 21] can be also included into the hierarchy. We omit detailed description of these models because they concern our work only marginally. The whole hierarchy is depicted in Figure 1.

Here, we are ready to introduce the extension of the de Berg class hierarchy. It is based on the idea of scaling mentioned above.

Definition 2.8: Let $\mathcal{C}$ be a class of objects, R be an arbitrary convex region. We call the class $\mathcal{C}$ *local* iff it holds for any set of objects $P | P \in \mathcal{C}$ that $(P \cap R) \in \mathcal{C}$.

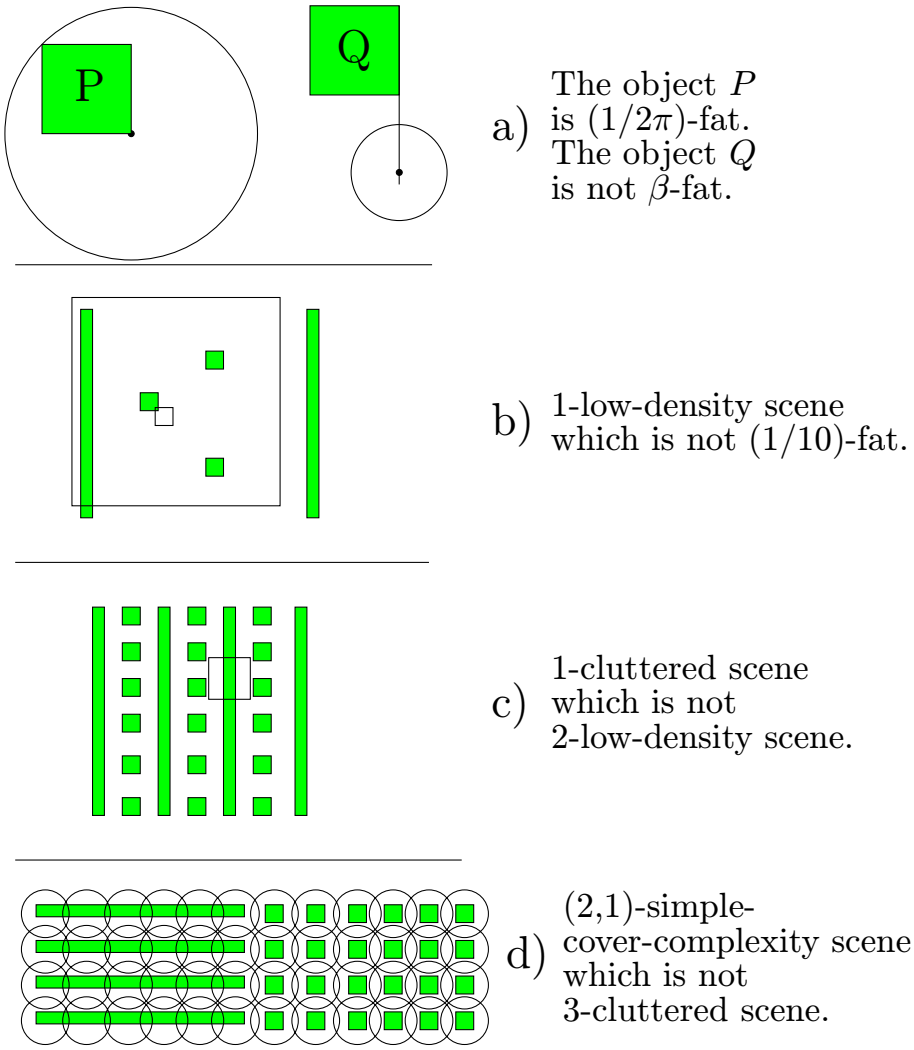

Figure 5: Object classes.

The locality markedly simplifies the work with particular scenes. For example, the classes *Low density* and *Uncluttered scenes* are local conversely to the classes *SCC* and *Guarding scenes*. This can be the reason of troubles with analysis of the last two mentioned classes.

Definition 2.9: Let $\mathcal{C}$ be a local class of objects. We define the extension of $\mathcal{C}$ called *locally-balanceable-*$\mathcal{C}$ (LB-$\mathcal{C}$) as follows:

Let $\mathcal{P} = \{p_1, ..., p_n\}$ be a scene (i.e. set of objects). We say that $\mathcal{P} \in$LB-$\mathcal{C}$ iff $\forall(p_i \in \mathcal{P})$: the intersection $\Omega_e(\delta, bb(p_i)) \cap \mathcal{P}$ can be reduced by linear scale transformations along some coordinate axes to $\mathcal{C}$ scene.

The proposed BSP algorithm exploits properties of the λ-*low-density* scenes and of the new class *LB-λ-low-density*, which is an extension of that one.

We shortly describe dependences inside the extended class hierarchy. First, it is clear that $\forall(\mathcal{C}) : \mathcal{C} \subseteq$ LB-$\mathcal{C}$ because of the fact that each LB-$\mathcal{C}$ is generated by the $\mathcal{C}$. The inequality is sharp in the case of local classes of de Berg hierarchy (Figure 2 illustrates an example of LB-Low density scene that lies outside the de Berg hierarchy). Second, $\forall(\mathcal{C}, \mathcal{C}') :$ LB-$\mathcal{C} \subset$ LB-$\mathcal{C}'$ iff $\mathcal{C} \subset \mathcal{C}'$.

Finally, we give the cornerstone of the algorithm - the Cutting Lemma. It was introduced by Tobola and Nechvile in [18] and it is essential for designing the algorithm and for proving size of resulting BSP trees. The Cutting lemma concerns two sets of segments on a plane. The simplified idea of the Cutting lemma looks as follows.

Let us have two identical sets of segments S and B on the plane. Then we shift the whole set B in arbitrary direction. It seems probable that for any direction we can select a line l cutting at least as many segments from the set B as segments from the set S. The Cutting lemma generalizes and proves this idea.

Proposition (Cutting lemma) 2.10: *Let S, B be non-empty sets of segments on the plane which fulfil the following conditions:*

1. $n = |S| \leq |B| = n + k; k \geq 0$ *is an integer.*
2. *There is such injective mapping* $\sigma : I \to J; I = \{1, ..., n\}, J = \{1, ..., n+k\}$ *and real constant* α*, that the following is true for all* $i \in I$*:* $(|s_i| \leq \alpha|b_{\sigma(i)}|) \wedge (s_i \parallel b_{\sigma(i)})$*, where* $|s_i|$ *means the length of segment* $s_i \in S$ *and* $|b_{\sigma(i)}|$ *means the length of segment* $b_{\sigma(i)} \in B$*.*

Furthermore, let v be an arbitrary non-zero vector such that $\exists(s_i) : s_i \nparallel v$ and p be a line parallel with v. Then: $\exists(p) : |p \cap S| \leq \alpha|p \cap B| \geq 1$.

3 The BSP method

Now, we are going to describe the algorithm for creating a linear BSP tree. The algorithm exploits a set of hyperrectangles to form a BSP tree.

The set of the hyperrectangles $R = \{r_1, ..., r_n\}$ is a set of bounding boxes of the original objects $\mathcal{P} = \{p_1, ..., p_n\}$ with constant complexity in $\mathcal{R}^d$ space. At first, we build up d pairs of auxiliary sets of segments B_i, $S_i | i \in \{1, ..., d\}$ as follows:

Let us project the set $\{r | r \in R\}$ of original hyperrectangles onto the i-th coordinate axis (further i-axis). The set of segments S_i contains the projected rectangles: $S_i = \{s | s = Proj_i(r), r \in R\}$. For the sake of simplicity, we will suppose that the endpoints are in general position (i.e. no two endpoints have the same x-coordinate).

The degenerate cases can be simply solved by lexicographical ordering of the points of the original hyperrectangles. Each endpoint of $s \in S_i$ can be considered as a projection of the unique point p_{max} $(p_{min}) \in r$ maximal (minimal) in the standard lexicographical ordering.

The $\Omega_s(\delta, r)$ neighborhood belonging to r is a part of hyperrectangle enclosing r. Let us project the set $\{\Omega_s(\delta, r) | r \in R\}$ onto the i-axis. We get the set of segments. Let us split each segment $Proj_i(\Omega_s(\delta, r))$ onto two parts by subtraction of the $Proj_i(r)$ from that one. We get two resulting segments: b_1 with lower coordinates and b_2 with higher coordinates associated with the segment s, as you can see in Figure 6. It follows from the definition of $\Omega_s(\delta, r)$ that $|b_1| = |b_2| = \delta|s|$. Moreover, the segments b_1, b_2 form $\Omega_s(\delta, s)$ neighborhood of the segment s. The set B_i is an unification of all segments b_1 and b_2 generated by the set $\{\Omega_s(\delta, r) | r \in R\}$. The degenerate cases are treated by lexicographical ordering as well.

The total number of $s \in S = \cup_{i \in \{1,...,d\}} S_i$ segments is dn and the total number of $b \in B = B \cup_{\{1,...,d\}} B_i$ segments is $2dn$, where d means the dimension of space.

Denotation 3.1: We call the segment $s \in S_i'$ *bounded* iff both segments b_1, b_2 associated with s have been eliminated from the set B_i'. We call the hyperrectangle r *bounded* iff all segments s associated with r are *bounded*. We call the hyperrectangle *bounded* along i-axis iff the segment $s \in S_i'$ associated with r is *bounded*.

Algorithm

if The set $\bigcup_{j=1}^{d} S_j$ is not empty **then**
begin

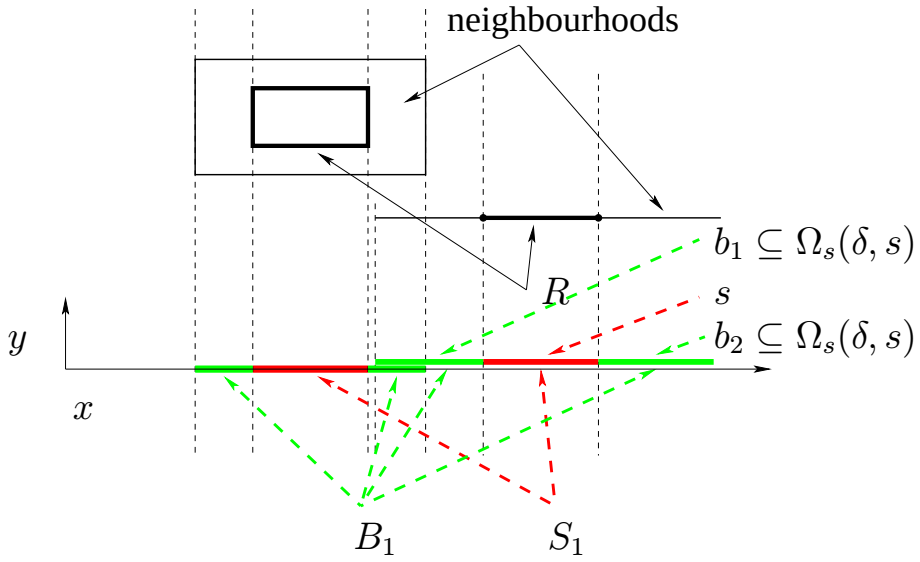

Figure 6: The sets S_1 and B_1.

(1) Eliminate all segments $\{s \in S_j |\ j \in \{1, ..., d\} \wedge$ s belongs to a *bounded* hyperrectangle$\}$ from the set S_j;
(2) **if** We cannot select an i such that a line l fulfilling the Cutting lemma conditions for the sets S_i, B_i can be found **then**
    // $\forall(l) : \frac{|l \cap B|}{|l \cap S|} \leq \delta$
    Select a line l with maximal value of $\frac{|l \cap B|}{|l \cap S|}$;
(3) **else** Select an appropriate line l according to the Cutting lemma;
(4) Select a hyperplane p that contains the line l. The hyperplane p should be perpendicular to the i axis;
(5) Eliminate all segments $b \in B_j | b \cap p \neq \emptyset$ from the sets $B_j | j \in \{1, ..., d\}$;
(6) Use p as the splitting hyperplane for the set R onto subregion R^{p^-} and R^{p^+};
(7) Split the sets $S_j \cup B_j | j \in \{1, ..., d\}$ using p so that the resulting sets $S_j^{p^-}$ and $S_j^{p^+}$ correspond to R^{p^-} and R^{p^+} (see fig. 7);
(8) recurse on the sets $S_j^{p^-}, B_j^{p^-} | j \in \{1, ..., d\}$ and R^{p^-};
(9) recurse on the sets $S_j^{p^+}, B_j^{p^+} | j \in \{1, ..., d\}$ and R^{p^+};
end
else
(10) Apply autopartition on the resulting set;

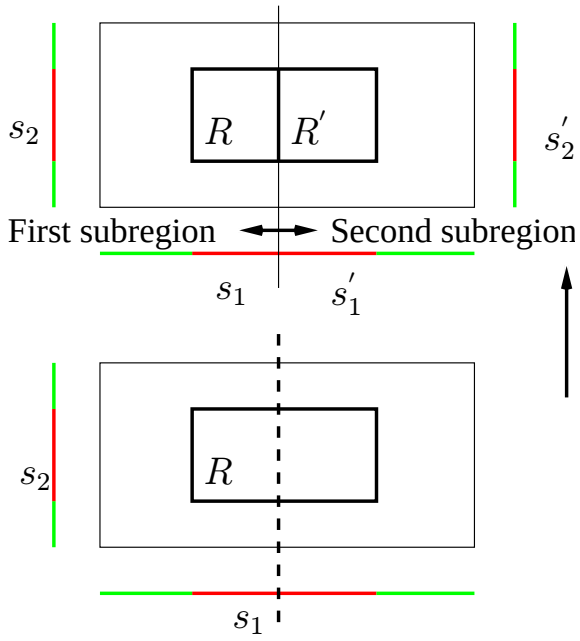

Figure 7: The rectangle R and sets S and B are split into two parts.

Lemma 3.2: *Let $\mathcal{P} \subset \mathbb{R}^d$ be the λ-low density scene, d be the constant. The proposed algorithm provides linear BSP tree for the scene $\mathcal{P}$.*

Proof: In the following text, we assume that δ is an appropriate constant[2].

Lemma 3.3: *Let $R = \{r_1, ..., r_n\}$ be the set of bounding hyperrectangles of the objects $\{p_1, ..., p_n\} \in \mathcal{P}$ in $\mathbb{R}^d$ space. Then each $\Omega_s(\delta, r_i)$ can be intersected by at most $\lambda\lceil 1 + 2\delta\rceil^d$ objects p_j such that $\varrho_{mec}(p_j) > \varrho_{mec}(p_i)$.*

Proof: Each bounding hyperrectangle r_i is covered by one hypercube $\varrho_{mec}(p_i)$. Furthermore, any $\Omega_s(\delta, r_i)$ can be covered by at most $\lceil 1 + 2\delta\rceil^d$ hyperrectangles r_i and hence by at most $\lceil 1 + 2\delta\rceil^d$ hypercubes $\varrho_{mec}(p_i)$. It follows from the definition of λ-low density that each $\Omega_s(\delta, r_i)$ can be intersected by at most $\lambda\lceil 1 + 2\delta\rceil^d$ objects p_j with $\varrho_{mec}(p_j) > \varrho_{mec}(p_i)$. □

Observation 3.4: Let p be a hyperplane perpendicular to i-axis and r be a hyperrectangle. Then the following cases can occur (see Figure 8):

[2]We have tested different constants δ in practical implementation. The results come in the forthcoming paper

1. The hyperplane p misses both the hyperrectangle r and its neighborhood $\Omega_s(\delta, r)$.

2. The plane p intersects the hyperrectangle r and the neighborhoods $\Omega_s(\delta, e_j^{\{1,-1\}}, r)|(j \in \{1, ..., d\} \wedge j \neq i)$.

3. The plane p intersects the neighborhoods $\Omega_s(\delta, e_j^{\{1,-1\}}, r)|(j \in \{1, ..., d\} \wedge j \neq i)$ and $\Omega_s(\delta, e_i^1, r)$

4. The plane p intersects the neighborhoods $\Omega_s(\delta, e_j^{\{1,-1\}}, r)|(j \in \{1, ..., d\} \wedge j \neq i)$ and $\Omega_s(\delta, e_i^{-1}, r)$

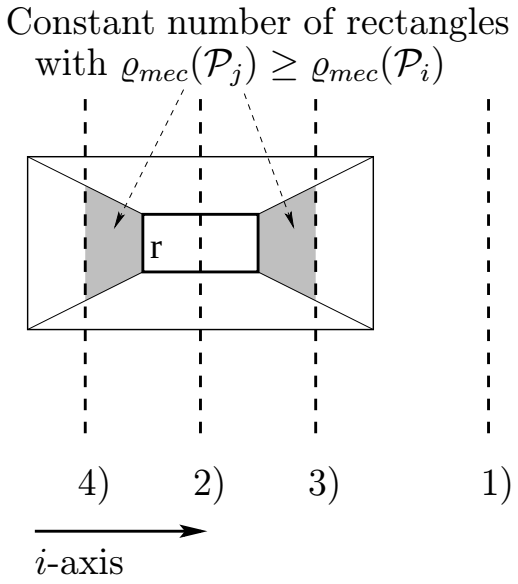

Figure 8: Intersections of r by hyperplanes perpendicular to the i-axis.

When cut (3) or cut (4) is used, then the boundaries between adjacent neighborhoods are intersected (Figure 8). The shaded part of $\Omega_s(\delta, e_i^1, r)$ (or $\Omega_s(\delta, e_i^{-1}, r)$) bounded by the plane p and the hyperrectangle r forms a blunt pyramid. The hyperrectangle r is clearly separated from the surrounding space when each of $\Omega_s(\delta, e_k^{\{1,-1\}}, r)$ is crossed by a perpendicular hyperplane. It follows from the above considerations that the part of space belonging to the $\Omega_s(\delta, r)$ can be intersected by at most $\lambda\lceil 1+2\delta\rceil^d$ objects with $\varrho_{mec}(\mathcal{P}_j) \geq \varrho_{mec}(\mathcal{P}_i)$.

Cuts (3) and (4) perpendicular to i-axis correspond to crossing a segment $b_{\{1,2\}} \in B_i$ by a line $l = Proj_j(p)|j \neq i$, p is the hyperplane perpendicular to i-axis. Similarly, each cut (2) perpendicular to i-axis corresponds to crossing a segment $s \in S_i$. We call cuts (3) and (4) *effective cuts*.

It follows from the definition of sets B_i and S_i that $\forall b_{\{1,2\}} \in B_i : |s| \leq 1/\delta |b_{\{1,2\}}|$. Moreover, $n = |S_i| \leq |B_i| = 2n$ and $s \parallel b_{\{1,2\}}$. The assumptions of the Cutting lemma are satisfied for the initial sets S_i and B_i and thus we can find such a line $l = Proj_j(p)|j \neq i$ that $|l \cap S_i| \leq 1/\delta |l \cap B_i|$.

In the first part of the algorithm (steps (1) - (9)) we use only splitting hyperplanes perpendicular to some i-axis. At least one segment $b_{1,2} \in B_i$ is treated in each pass of the algorithm and no quite new segment $b_{1,2} \in B$ can arise in any new subregion. Hence, the algorithm finishes after a finite number of steps.

Now, let us suppose that the assumptions of the Cutting lemma are satisfied during each step (2) of the algorithm. We show that the algorithm provides linear BSP in this case. The opposite case will be discussed afterward.

It is clear that each segment $b \in B_i$ has been crossed by a line l (and therefore discarded) during the run of our algorithm. Each such line l corresponds to the splitting hyperplane p, which goes through a $\Omega_s(\delta, r)$. Therefore, each segment $s \in S_i$ is *bounded* and each hyperrectangle r (or fragment of the hyperrectangle) finds itself in a subset of the original space separated by splitting hyperplanes from the rest of the original space. Each such subset of the original space can consist of the number of the simple hyperrectangular cells corresponding to BSP nodes.

Let $\mathcal{M}$ be the maximal set of objects p_j such that each p_j lies in different simple cell and p_j has minimal diameter of $\varrho_{mec}(p_j)$ of all objects in the same cell. It follows from Lemma 3.3 that each $\Omega_s(\delta, r_j)$ can be crossed by at most $\lambda \lceil 1 + 2\delta \rceil^d$ of other objects p_k with $\varrho_{mec}(p_j) \leq \varrho_{mec}(p_k)$. Moreover, each p_j has been separated from the neighborhood by hyperplanes going through the $\Omega_s(\delta, r_j)$. Because δ and d are constants, the number of p_k objects (or object fragments) contained in this cell is at most constant. If we use autopartition for this set of objects, we obtain a BSP of constant complexity. We will show that the total number of simple hyperrectangular cells is linear.

Let us select an arbitrary fixed $i \in \{1, ..., d\}$. If the objects are split only by the cuts perpendicular to i-axis and no object is split by a cut perpendicular to another axis, then at most $O(2n\frac{1}{\delta})$ objects (hyperrectangles) are split by this cut in total (we have to cross $2n$ segments $b \in B_i$ and we

use only *effective cuts* according the Cutting lemma. The $O(2n\frac{1}{\delta})$ boundary follows from that Lemma). Hence, we have at most $O(n+2n\frac{1}{\delta}) = O((1+\frac{2}{\delta})n)$ objects. Now, we extend our consideration to the remaining dimensions. It is easy to show by induction that the total number of cells is at most $O((1+\frac{2}{\delta})^d n)$.

Together, the overall size of the resulting BSP tree is bounded by

$$\begin{aligned} & O(\lambda(1+2\delta)^d(1+\tfrac{2}{\delta})^d n) \quad = \\ = \quad & O(\lambda(5+\tfrac{2}{\delta}+2\delta)^d n \quad = \quad O(n) \end{aligned}$$

It still remains to explore the case when the conditions of the Cutting lemma fail. This can happen only if every set S_i contains a *bounded* segment s (in opposite case, there is at least one segment $b_{\{1,2\}}$ associated with s and the Cutting lemma conditions are fulfilled).

Let us suppose that this event occurs in $R' \subset R^d$. Then R' forms a hyper-rectangle, which contains a set of objects $\mathcal{P}'$. Each of R' sides is contained in a splitting hyperplane. Let u_i be the longest edge of R', S_i^b be the set of *bounded* segments parallel with u_i and let $\mathcal{P}_i'^b$ be the set of objects from which the segments S_i^b are generated. It follows from Lemma 3.3 that the number of objects in $\mathcal{P}_i'^b$ is bounded by $\lambda\lceil 1+2\delta\rceil^d$. If we remove the objects $\mathcal{P}_i'^b$ from the set $\mathcal{P}'$ then the conditions of the Cutting lemma are fulfilled and we can find a splitting hyperplane p. Any such splitting hyperplane can cut at most $\lambda\lceil 1+2\delta\rceil^d$ extra objects. Because at least one segment $b \in B$ is crossed by the line l, the ratio between crossed segments $s \in S_i$ and $b \in B_i$ is at most $\frac{1}{\delta} + \lambda\lceil 1+2\delta\rceil^d$ i.e. constant. If we use the last expression in the previous part instead of $\frac{1}{\delta}$ then we obtain the following bound of BSP size.

$$O(\lambda(1+2\delta)^d(1+\frac{2}{\delta}+2\lambda(1+2\delta)^d)^d n) = O(n)$$

□

The constant in the previous lemma looks nasty but this result is overrated. Violation of the conditions of the Cutting lemma occurs very rarely in practical scenes and the real constant is very low.

Lemma 3.5: *If the above algorithm works for a local class $\mathcal{C}$, then it also works for the class LB-$\mathcal{C}$.*

Proof: As long as a cut according to the Cutting lemma can be found, the algorithm works well. Problems can appear only in the case when the conditions of Cutting lemma fail. It was shown in Lemma 3.2 that every set S_i contains at least one *bounded* segment s in this case.

Let $\mathcal{R}$ be a BSP region corresponding to the subtree node for which the lemma conditions fail, S^b be the set of all *bounded* segments associated with $\mathcal{R}$. The region $\mathcal{R}$ is formed by a hyperrectangle because it is bounded by two axis-perpendicular hyperplanes for any i-axis.

Let $\mathcal{M}$ be set of all original hyperrectangles from which the bounded segments $s_i \in S^b$ can be derived and $A \in \mathcal{M}$ be the hyperrectangle with maximal length of the longest side among all hyperrectangles from the set $\mathcal{M}$ (we mark this length L). Without lost of generality (w.l.o.g.), we can suppose that the mentioned longest side is parallel to the x-axis (see the Figure 9). We show that the hyperrectangle $\mathcal{R}$ is completely covered by the neighborhood $\Omega_e(\delta, A)$. This statement is clearly true iff all cuts which form the boundary of $\mathcal{R}$ go through the $\Omega_e(\delta, A)$.

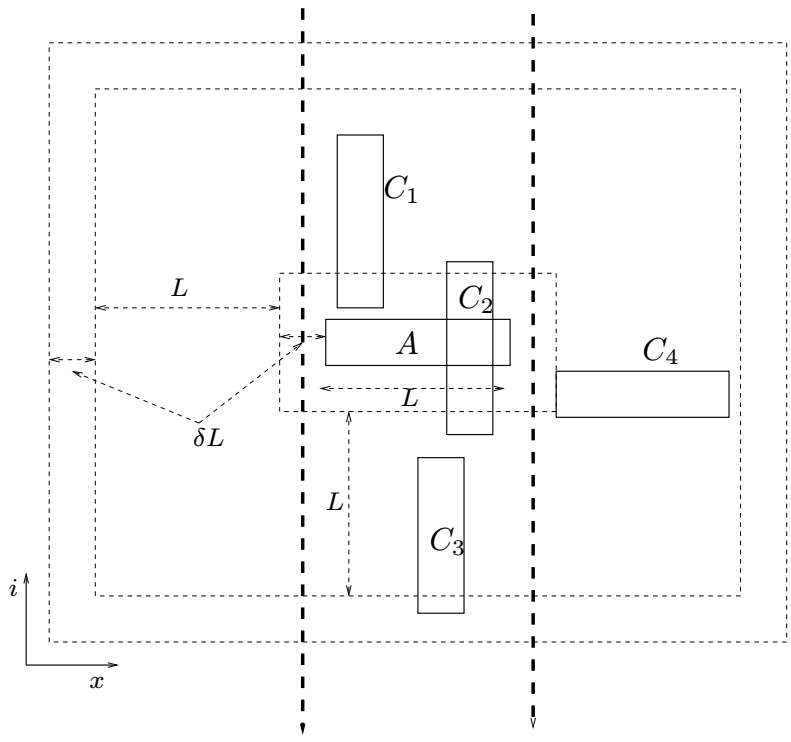

Figure 9: The two dimensional example.

It results from the definition of *extended neighborhood* that the thickness of $\Omega_e(\delta, A)$ is at least $2\delta L + L$ in arbitrary direction. In addition to this

fact, A is separated from the rest of the scene by two cuts perpendicular to a coordinate axis (w.l.o.g. we can suppose that it is the x axis) and the cuts go through the neighborhood $\Omega_s(\delta, A) \subseteq \Omega_e(\delta, A)$. It is clear that the region $\mathcal{R}$ is covered by the $\Omega_e(\delta, A)$ iff there are two similar cuts for any remaining coordinate axes which go through $\Omega_e(\delta, A)$.

Let i-axis ($i \neq x$) be an arbitrary coordinate axis and $C \in \mathcal{M}$ be a hyperrectangle, which is *bounded* by two cuts perpendicular to the i-axis. We can distinguish the next cases:

1. $A = C$. Then the cuts which form the boundary of $\mathcal{R}$ go clearly through $\Omega_e(\delta, A)$.

2. $A \neq C$. Then:

   (a) There are points $u \in A$ and $v \in C$ such that $|u, v|_i \leq \delta L$, where $|u, v|_i$ denotes the distance of points u,v on the i-axis (see Fig. 9, rectangles C_1 (or C_2), A). We use the fact that C is *bounded* by two cuts perpendicular to i coordinate axis and this cuts go through the *simple neighborhood* of C. Let w be a point inside of C with its simple neighborhood. Then $|u, w|_i \leq |u, v|_i + L_1 + \delta L_1 \leq 2\delta L + L$. From the previous considerations, both the cuts go through the extended neighborhood of A.

   (b) The points $u \in A$ and $v \in C$ such that $|u, v|_i \leq \delta$ do not exist (see Fig. 9, rectangles C_3, A). Then $\forall (u \in A, v \in C) |u, v|_i > \delta$. Nevertheless, there is a cut perpendicular to i-axis and it goes through the *simple neighborhood* of C. Such a cut must separate the rectangle A and the rectangle C. Hence, A and C cannot belong to the set $\mathcal{M}$ at the same time – it is a contradiction.

We have already shown that region $\mathcal{R}$ is covered by the $\Omega_e(\delta, A)$. It follows directly from definition 2.9 and the conditions of this lemma that the region $\mathcal{R}$ can be reduced by linear scale transformations along some coordinate axes to a $\mathcal{C}$ scene.

It remains to show that the algorithm works identically for a scene $\mathcal{P}$ and for it's counterpart $\mathcal{P}'$ created by linear scale transformations of $\mathcal{P}$. However, this fact follows in a straightforward fashion from the observation that the order of endpoints of the set of segments $\{S_i' \in \mathcal{P}'\} \cup \{B_i' \in \mathcal{P}'\}$ is identical to that one of the set $\{S_i \in \mathcal{P}\} \cup \{B_i \in \mathcal{P}\}$. Hence, the splitting line l' of the set $\mathcal{P}'$ has identical behavior as the splitting line l of the set $\mathcal{P}$. $\square$

Corollary 3.6: *The above algorithm works for the class LB-low-density.*

The method presented above provides us with a pseudo algorithm for creating a linear BSP tree. If we opt for brute force approach in implementation of the algorithm it could behave very inefficiently and it could violate space and time bounds proclaimed above.

The main difficulty is to find the splitting line according to the Cutting lemma. We suggest two ways how to achieve the $O(n \log^2 n)$ time bound.

The first way is to use the *tandem search* [6] technique. To avoid spending too much time in case of an unbalanced split, we start walking simultaneously from both ends, one step at a time. This way we find the smaller subsets from (R^{p^+}, $S_j^{p^+}$, $B_j^{p^+}$) and (R^{p^-}, $S_j^{p^-}$, $B_j^{p^-}$) (lines (6) and (7) of the algorithm) in time proportional to the size of this subset. This technique leads to the $O(n \log^2 n)$ time and besides it works in $O(n)$ space. Unfortunately, this method tends to create as unbalanced BSP tree as possible. Because the balance criterion is very important in many practical tasks, we suggest another method for solving this problem.

The second way is to use the segment trees discovered by Bentley [3]. We will maintain the set of segments B_i and S_i in a segment tree along with some extra data. Using these trees, we will be able to select the splitting plane effectively according to the Cutting lemma. This technique is described in detail in [17]. The advantage of this technique lies in the possibility of controlling the search of the cutting line l. Hence, we can trade-off simply the balance and size of the resulting BSP tree. The slight drawback of this method is the $O(n \log^2 n)$ space requirement.

At the end it should be noted that the algorithm works well even if the non-overlapping condition is not satisfied. The resulting BSP tree is always correct.

Theorem 3.7: *Let $\mathcal{P}$ be a set of objects in $\mathbb{R}^d$, $P \in$ LB-λ-low-density. Then the linear size BSP tree can be constructed in $O(n \log^2 n)$ time and $O(n \log^2 n)$ space. Moreover, we can trade-off between balance and size of the resulting tree.*

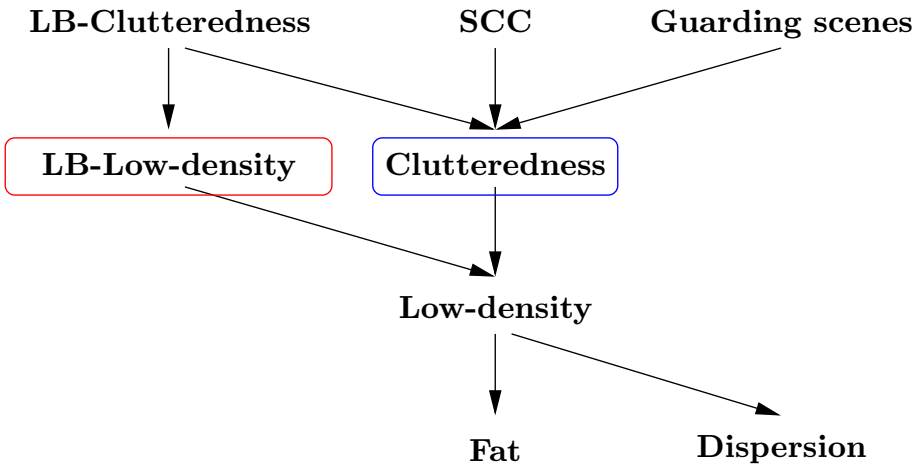

Figure 10: The full hierarchy of recent object classes. The red oval denotes domain of our algorithm and the blue oval denotes domain of de Berg's algorithm.

4 Conclusion

In this paper, we have proposed the extension of the existing de Berg hierarchy of object classes by a set of so called *locally balanceable* (LB) classes. The full class hierarchy is depicted in Figure 10.

Moreover, we have presented a new BSP construction algorithm, which works provably well for sets of objects with LB-λ-low density. In this case it provides linear BSP trees and it runs in $O(n \log^2 n)$ time and space. This method is very simple.

The de Berg's algorithm works for the uncluttered scenes which are more general then λ-low density scenes. However, our algorithm covers some scenes, which fall outside the definition of uncluttered scenes. Moreover, the Cutting lemma provides our algorithm with certain input sensitivity.

The following step is to compare the real performance of the discussed algorithms by practical implementation. Moreover, it would be interesting to explore the definition of LB-$\mathcal{C}$ scenes and try to refine it to be more powerful.

References

[1] P. Agarwal, M. Katz, and M. Sharir. Computing depth order and related problems. *Comput. Geom. Theory Appls.*, 5:187–206, 1995.

[2] John Milligan Airey. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-space Subdivision and Potentially Visible Set Calculations*. Ph.D. Thesis, Dept. of Computer Science, University of North Carolina, Chapel Hill, 1990.

[3] J. L. Bentley. Solutions to Klee's rectangle problems. Technical report, Carnegie-Mellon Univ., Pittsburgh, PA, 1977.

[4] M. de Berg, H. David, M. Katz, M. Overmars, F. van der Stappen, and J. Vleugels. Guarding scenes against invasive hypercubes. In *Proc. 2nd Workshop on Algorithm Engineering*, pages 110– 120, 1998.

[5] M. de Berg, M. J. Katz, M. Overmars, A. F. van der Stappen, and J. Vleugels. Models and motion planning. In *In Proc. 6th Scand Workshop Algorithm Theory, Lecture Notes Comput. Sci., Springer-Verlag*, 1998.

[6] M. de Berg, M. Overmars, and O. Schwarzkopf. Computing and verifying depth orders. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 138–145, 1992.

[7] Mark de Berg. Linear size binary space partitions for uncluttered scenes. *Algorithmica*, 28(3):353–366, 2000.

[8] Mark de Berg, Matthew J. Katz, A. Frank van der Stappen, and Jules Vleugels. Realistic input models for geometric algorithms. In *Symposium on Computational Geometry*, pages 294–303, 1997.

[9] H. Fuchs, Z. M. Kedem, and B. Naylor. On visible surface generation by a priori tree structures. *Comput. Graph.*, 14(3):124–133, 1980. Proc. SIGGRAPH '80.

[10] Joseph S. B. Mitchell, David M. Mount, and Subhash Suri. Query-sensitive ray shooting. *International Journal of Computational Geometry and Applications*, 7(4):317–347, 1997.

[11] Mark H. Overmars and A. Frank van der Stappen. Range searching and point location among fat objects. In *European Symposium on Algorithms*, pages 240–253, 1994.

[12] M. Paterson and F. Yao. Optimal binary space partitions for orthogonal objects. *J. Algorithms*, 13:99–113, 1992.

[13] M. S. Paterson and F. F. Yao. Efficient binary space partitions for hidden-surface removal and solid modeling. *Discrete Comput. Geom.*, 5:485–503, 1990.

[14] Ph. Pignon. *Structuration de l' espace pour une planification hi'erarchis'ee des trajectoires de robots mobiles.* Ph.D. thesis, LAAS-CNRS and Universit'e Paul Sabatier de Toulouse, 1993. Rapport LAAS N o 93395.

[15] S. Teller. *Visibility Computations in Densely Occluded Polyhedral Environments.* Ph.D. thesis, Computer Science Div., Univ. of California, Berkeley, 1992.

[16] William C. Thibault and Bruce F. Naylor. Set operations on polyhedra using binary space partitioning trees. *Computer Graphics*, 21(4):153–162, 1987.

[17] Petr Tobola and K. Nechvile. Linear BSP trees for sets of hyperrectangles with low directional density. In V. Skala, editor, *WSCG 2001 Conference Proceedings*, pages 237–244, 2001.

[18] Petr Tobola and Karel Nechvile. Linear BSP tree in the plane for set of segments with low directional density. In V. Skala, editor, *WSCG'99 Conference Proceedings*, pages 297–304, 1999.

[19] A. van der Stappen, D. Halperin, and M. Overmars. The complexity of the free space for a robot moving amidst fat obstacles. *Computational Geometry: Theory and Applications*, 3:353–373, 1993.

[20] A. van der Stappen and M. Overmars. Motion planning amidst fat obstacles. In *In Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 31–40, 1994.

[21] J. Vleugels. *On fatness and fitness — realistic input models for geometric algorithms.* Ph.D. thesis, Dept. Comput. Sci., Univ. Utrecht, Utrecht, The Netherlands, 1997.

Publications in the FI MU Report Series are in general accessible via WWW and anonymous FTP:

`http://www.fi.muni.cz/informatics/reports/`
`ftp ftp.fi.muni.cz (cd pub/reports)`

Copies may be also obtained by contacting:

**Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic**