

International Journal of Computational Geometry & Applications
© World Scientific Publishing Company

RED-BLUE SEPARABILITY PROBLEMS IN 3D*

FERRAN HURTADO

*Departament Matemàtica Aplicada II
Universitat Politècnica de Catalunya
Jordi Girona 1, 08034 Barcelona, Spain
ferran.hurtado@upc.edu*

CARLOS SEARA†

*Departament Matemàtica Aplicada II
Universitat Politècnica de Catalunya
Jordi Girona 1, 08034 Barcelona, Spain
carlos.seara@upc.edu*

SAURABH SETHIA

*Department of Computer Science
Oregon State University
Corvallis, OR 97331-3202, USA
saurabh@cs.orst.edu*

Received (received date)
Revised (revised date)
Communicated by (Name)

In this paper we study the problems of separability of two disjoint point sets in 3D by multiple criteria extending some notions on separability of two disjoint point sets in the plane.

Keywords: Separability; slab; wedge; double wedge; prism; pyramid; tetrahedron; box.

1. Introduction

Let B and R be two disjoint sets of points in 3D classified as *blue* and *red* points, respectively. Let n be the number of points in $B \cup R$. We consider the sets of points in general position, i.e., there are no four points in a plane.

Let \mathcal{C} be a family of surfaces in 3D. The sets B and R are \mathcal{C} *separable* if there exists a surface $S \in \mathcal{C}$ such that every connected component of $\mathbb{R}^3 - S$ contains points only from B or from R . If S is a plane, we have *linear separability*. The decision problem of linear separability for two disjoint sets of objects (points, segments,

*Supported by projects MEC-MCYT-FEDER BFM2003-00368 and Gen Cat 2001SGR00224.

†Corresponding author.

2 Ferran Hurtado, Carlos Seara, Saurabh Sethia

polygons or circles) in 2D or (points, segments, polyhedra or spheres) in 3D can be solved in linear time^{9,15}.

In^{2,11,12} the authors study the separability of two disjoint point sets in the plane by the following criteria: *wedge separability*, *strip separability* and *double wedge separability*. A *wedge* is the union of two rays with common origin (the *apex*). A *strip* is the union of two parallel lines; their common slope is the *slope of the strip*. A *double wedge* is the union of two crossing lines, their intersection point is the *apex*. Optimal $\Theta(n \log n)$ time algorithms for deciding wedge, strip, and double wedge separability, as well as for computing the set of apices of separating wedges and double wedges, and the set of slope intervals of separating strips are described in^{2,11,12}. The set of apices and the set of slope intervals are partial descriptions of all separating wedges, double wedges, and strips. They have also shown how to find wedges and double wedges with maximum and minimum aperture angle, and the narrowest and the widest strips.

In this paper we study the separability of two disjoint point sets in 3D by extending the criteria above and giving solutions to various separability problems. While the separability criteria are extensions of their 2D counterpart, the techniques used in 3D do not extend from those used for 2D. For each separability criterion we consider the problem of deciding whether that particular separability is feasible, which is probably equivalent to finding one solution to the problem; in some cases, we also consider the problem of finding a representation of all feasible solutions. For some separability criteria we consider, the convex hull of either the red or blue points needs to be monochromatic. If it is so, we perform this check as a first step in our algorithms. It can be done in $O(n \log n)$ time by computing the convex hulls of the point sets in 3D¹⁶. In all our algorithms this time is dominated by other computations.

The paper is organized as follows. We consider the problem of computing a representation of all the feasible solutions for linear separability in Section 2. We study slice, wedge, and diwedge separability in Section 3 as the natural extensions in 3D for strip, wedge, and double wedge separability in 2D. We study the decision problem for prismatic, pyramidal, and dipyramidal separability, which also can be considered as extensions in 3D for strip, wedge, and double wedge separability criteria, but allowing a large number of planes in Section 4. Finally, we study some separability criteria defined by a constant number of planes in Section 5. We consider here a systematic approach to these problems from the deterministic viewpoint. A preliminary version of this work appeared in¹³; subsequently some researches undertook the task of studying some of these problems via a randomized approach¹.

2. Linear separability

The decision problem of linear separability for two disjoint point sets in 3D can be solved in linear time¹⁵. We look at the problem of computing a *representation* of

all the separating planes. By a representation we mean a region on the unit sphere defining the *set of directions of the separating planes* or a region in 3D defining the *locus of points of separating planes*. We compute these representations as follows.

- (1) In linear time compute a plane separating the two sets of points. Assume that this plane is the $y = 0$ plane such that the red points are above the plane and the blue points are below the plane.
- (2) In $O(n \log n)$ time compute the convex hulls of the two point sets, $CH(R)$ and $CH(B)$.
- (3) Compute the interior supporting planes or *extreme separating planes* between $CH(R)$ and $CH(B)$ (Figure 2). The extreme separating planes are supporting planes for $CH(R)$ and $CH(B)$ such that $CH(R)$ and $CH(B)$ lie in different half-spaces. These planes can be computed in $O(n)$ time using a projective transformation given by Davis in ⁷ which maps planes to planes, lines to lines, and preserves affine independence. The computation of the extreme separating planes is as follows: let $CH(R')$ and $CH(B')$ be the convex hulls of the red and blue points once the projective transformation has been done.

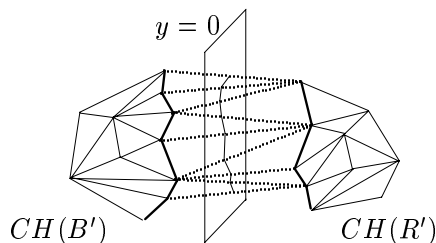


Fig. 1. Wrapping $CH(R')$ and $CH(B')$.

The extreme separating planes between $CH(R)$ and $CH(B)$ are the supporting planes between $CH(R')$ and $CH(B')$, which can be computed in $O(n)$ time by the wrapping algorithm of two line separable convex polyhedron given by Preparata and Hong ¹⁶ (Figure 1). The wrapping plane rotates over a supporting line between $CH(R')$ and $CH(B')$ and its normal vector has only one degree of freedom. As a new supporting line is bumped, the wrapping plane rotates over the new supporting line and so on.

- (4) *Set of directions of separating planes*: Notice that the extreme separating planes of $CH(R)$ and $CH(B)$ do not necessary have a common intersection point. However, the region on the unit sphere centered on the origin defined by the tips of the unit normal vectors of the extreme separating planes consists of two antipodal convex regions. This is so because each extreme separating plane defines two half-spaces: one containing the red convex hull (positive half-space) and the other containing the blue convex hull (negative half-space). Now we can

4 Ferran Hurtado, Carlos Seara, Saurabh Sethia

consider the planes parallel to the extreme separating planes passing through the origin. The intersection of their corresponding positive half-spaces gives a convex cone with apex on the origin, *the red convex cone*. Analogously, the intersection of their corresponding negative half-spaces gives a convex cone with apex on the origin, *the blue convex cone*, and opposite to the red convex cone. The intersection of these opposite cones with the unit sphere are two antipodal convex regions, which represent the regions on the unit sphere defined by the set of directions of the separating planes. These regions are not empty because of the linear separability of the sets. The boundary of these regions is formed by $O(n)$ circular arcs because the wrapping algorithm above produces $O(n)$ planes and, it can be computed in $O(n)$ time from the red and blue convex cones above. If we project a region on the plane $z = 1$ using rays from the origin, we will get a region bounded by either one convex polygon with $O(n)$ edges (Figure 2b) or two unbounded convex polygons with $O(n)$ edges, depending on whether the convex region is located in the north (or south) hemisphere or in between the two hemispheres.

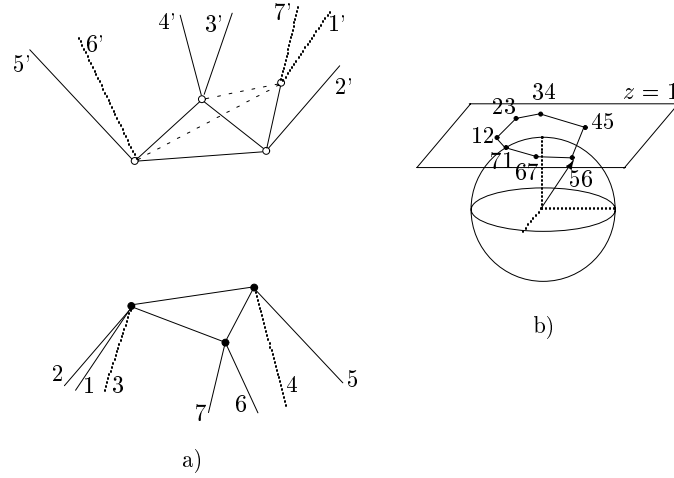


Fig. 2. a) Unbounded convex polygonal regions, b) set of directions of separating planes.

- (5) *Locus of points of separating planes*: The boundary of the locus of points of separating planes consists of two unbounded convex polygonal regions with $O(n)$ total complexity, which are formed by unbounded faces (defined by the extreme separating planes between $CH(R)$ and $CH(B)$) and some faces of $CH(R)$ and $CH(B)$ (these faces correspond to the faces of $CH(R')$ and $CH(B')$ belonging to $CH(R') \cup CH(B')$) (Figure 2a). Thus, using Davis algorithm ⁷ we can compute all these faces in linear time from $CH(R)$ and $CH(B)$. Once we have computed the boundary of the locus, we can decide in $O(\log n)$ time

whether a given plane is a separating plane, just checking whether the given plane intersects any of the two unbounded convex polygonal regions.

Lower bound: The problem of computing the boundary of the locus above has an $\Omega(n \log n)$ time lower bound due to the time lower bound for the 2D version of this problem. Moreover, the problem of computing the region defining the set of directions of separating planes has also an $\Omega(n \log n)$ time lower bound: take a blue point in the origin of the coordinate system and a set of red points in a half circle on a plane π parallel to the x - y -plane as it is shown in Figure 3. From the boundary of the region of directions of the separating planes for these point sets it is not difficult to compute the convex hull of the red points.

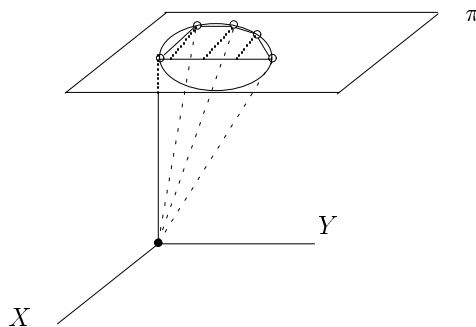


Fig. 3. Lower bound for the set of directions of separating planes.

Thus, by the discussion above, we have the following theorem.

Theorem 1. *Let B and R be two disjoint point sets in \mathbb{R}^3 . A representation of the set of directions of separating planes and the locus of points of separating planes can be computed in $\Theta(n \log n)$ time. Once we have pre-computed the locus, to decide whether a given plane separates the point sets can be done in $O(\log n)$ time.*

The problem of computing the maximum Euclidean distance between the sets or, in other words, the problem of computing maximum Euclidean distance between two parallel separating planes has been solved by Houle⁹ with the following theorem.

Theorem 2.⁹ *Given two point sets in \mathbb{R}^d , then a separating hyperplane which minimizes the orthogonal Euclidean distance between the hyperplane and the point sets may be found, or its non-existence determined, in $O(n)$ time.*

An obvious consequence of Theorem 2 is the following corollary.

Corollary 1. *Let B and R be two disjoint point sets in \mathbb{R}^3 . The maximum Euclidean distance between two parallel separating planes of the point sets can be computed in $O(n)$ time.*

3. Separability by two planes

In this section we study natural extensions for strip, wedge, and double wedge separability criteria in 2D, which involves exactly two separating planes. More concretely, we study slice, wedge, and diwedge separability as extension for strip, wedge, and double wedge separability, respectively.

3.1. Slice separability

A slice is defined as the space between two parallel planes (Figure 4). The normal vector to the planes of the slice is called the *slice direction*. Slice separability can be considered as a natural extension for strip separability in the plane. The slice separability problem asks: Is there a slice that contains all the red points but does not contain any blue point or vice versa?

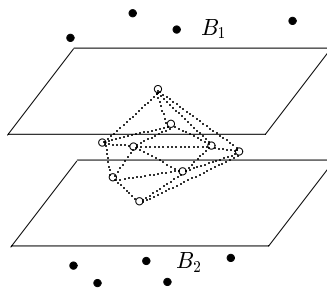


Fig. 4. Slice separability.

Before giving a solution to the problem, we address the following question. Suppose that there exists a slice separating R from B and the slice is given by two horizontal parallel planes. Let B_1 (B_2) be the set of blue points above (below) the slice. The sets B_1 and B_2 form a partition of B . The question now is, how many different partitions of B corresponding to slice separators can exist? We show that *the number of partitions is $\Theta(n^2)$ in the worst case.*

First, we show that there can be at most $\binom{n}{2}$ slice separators with different partitions as follows. Adding a red point only reduces the number of slice separators. So let's assume that there is only one red point r . Thus the slice separator is reduced to two planes passing through r . Hence the number of different partitions we get using planes passing through r is an upper bound. Any plane passing through r can be moved around without changing the partition until it bumps into two blue points. Thus the number of pairs of blue points is an upper bound on the number of different partitions.

Second, we show an example with $O(n^2)$ partitions doing the following construction. There is only one red point and it is at the origin of the coordinates

system. Half of the blue points are equally spaced on the segment $x = -1, z = 0, y = [-1, 1]$, and the other half of the blue points are equally spaced on the segment $x = 1, y = 0, z = [-1, 1]$ (Figure 5). Consider the set of points *mid-way* between successive blue points on these segments. There are $O(n)$ such mid-way points on each segment. Now consider any triangle T formed by one mid-way point from each segment and the origin. There are $O(n^2)$ such triangles. Let the plane containing T be π_1 . Two planes parallel to π_1 but infinitesimally away from π_1 in opposite directions would form a separating slice. And there would be a different partition for each triangle.

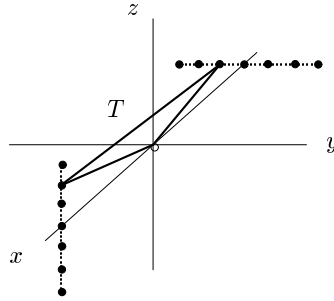


Fig. 5. $O(n^2)$ triangles.

Now let's see how we can decide the existence of a separating slice.

Theorem 3. *Let B and R be two disjoint point sets in \mathbb{R}^3 . Deciding whether the sets are slice separable can be done in $O(n^3)$ time. A representation of the set of slice directions of separating slices can be computed in $O(n^3 \log n)$ time and it has at most $\Theta(n^2)$ connected components with total complexity $\Theta(n^2)$.*

Proof. Assume that there exists a separating slice which contains R . Let u be its *slice direction*. Let B_1 (B_2) be the set of blue points above (below) the slice. Then a plane π_1 with normal vector u passing through any red point r is a separating plane of B_1 and B_2 . While keeping the point r on the plane, we can move u around with two degrees of freedom until it bumps into two blue points b_1 and b_2 . There are three exclusive possibilities: both points belong to B_1 , both belong to B_2 , or one belongs to B_1 and the other one belongs to B_2 . Therefore, if there exists a separating slice then, it can be found by the following algorithm.

- (1) Choose a red point $r \in R$.
- (2) For each pair of blue points b_1 and b_2 , compute the plane π_1 passing through b_1, b_2 , and r .
- (3) Considering the cases above, compute B_1 (B_2), the set of blue points above (below) π_1 .

8 *Ferran Hurtado, Carlos Seara, Saurabh Sethia*

- (4) By linear programming compute a separating slice (if it exists) defined by two parallel planes such that one separates B_1 from $R \cup B_2$ and, the other one separates $R \cup B_1$ from B_2 .

To compute a representation of the set of slice directions of all the feasible solutions we proceed as follows. Once a separating slice partitioning B into B_1, B_2 is founded; we do the following.

- (4.1) Compute $CH(B_1)$, $CH(B_2)$, $CH(R \cup B_1)$, and $CH(R \cup B_2)$.
 (4.2) Compute the region on the unit sphere formed by the set of directions of the separating planes between $CH(B_1)$ and $CH(R \cup B_2)$, as we did in Section 2. Proceed analogously for computing the set of directions of the separating planes between $CH(R \cup B_1)$ and $CH(B_2)$. We obtain two regions with $O(n)$ complexity and bounded by at most two convex chains. The projection of each region on the plane $z = 1$ is either a convex polygon or two unbounded convex polygons. Thus there are at most four convex polygonal chains.
 (4.3) Compute the (non empty) intersection of the two regions in $O(n)$ time. Its boundary corresponds to parallel planes which are always touching $CH(R)$ and some blue point.

Thus, in additional $O(n \log n)$ time, we can compute the convex region on the unit sphere defined by the set of slice directions of all the feasible solutions for each *good partition*. The total time complexity of the algorithm is $O(n^3 \log n)$.

Notice that different partitions give disjoint convex regions of slice directions; since otherwise there is a slice direction shared by two different partitions which differ in at least a point b and then, b has to lie in different half-spaces at the same time. Therefore, the region on the unit sphere of the set of slice directions of the feasible solutions is formed by at most $O(n^2)$ connected components, each one with at most linear complexity. Nevertheless, we show next that the total complexity of this region is $\Theta(n^2)$.

First, notice that our argument for the lower bound on the number of different partitions shows that there exist examples such that the locus has $\Omega(n^2)$ connected components, each one with constant complexity. We now argue that the size of the locus cannot be more than $O(n^2)$. The connected components of the region have to be bounded by some edges and vertices. A vertex corresponds to a solution with two points (red or blue) incident on either of the two slice plains (one point on each plain is allowed). Since there are only $O(n^2)$ pairs of points, there can only be $O(n^2)$ vertices. Hence the total complexity of the region is at most $O(n^2)$. \square

3.1.1. *The narrowest and the widest separating slices*

The width of a separating slice is defined by the distance between its parallel planes. If the sets are slice separable, it is natural to ask about the narrowest and the widest separating slices.

The widest separating slice: The widest separating slice depends on the sets of blue points B_1 and B_2 , above and below the slice, respectively and on the slice directions for this partition. Suppose that we have computed the region D on the plane $z = 1$ corresponding to the slice directions for this partition. The region D can consist of at most four convex polygons. Let these components of D be d_1, d_2, d_3, d_4 . To compute the widest separating slice, we can do the following.

- (4.4) Apply Theorem 2⁹ to the point sets B_1 and B_2 but introducing the set of (at most $O(n)$) linear constraints on the normal vector \tilde{h} of the solution plane h which are given by d_i (Figure 6). We do this for each d_i separately. Notice that doing this, we still have a quadratic minimization problem in three variables and $O(n)$ constraints, which can be solved in $O(n)$ time and space. Therefore, we compute the widest separating slice for each partition and for each d_i and maintain the widest by doing the same computation for each partition generated by a separating slice (at most $O(n^2)$).

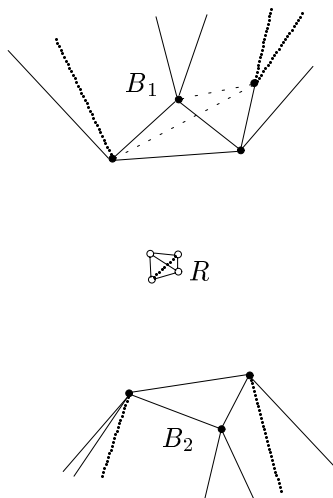


Fig. 6. The widest slice.

Thus, we can compute the widest separating slice in total $O(n^3 \log n)$ time. However, once we have pre-computed the at most $O(n^2)$ convex regions on the unit sphere (with $O(n^2)$ total complexity), we can proceed as follows. For each convex region, we take a direction on this region and we make a parallel sweep with this direction obtaining the first and last red points and classifying the blue points into B_1 and B_2 , depending on whether a blue point is located before the first red point or after the last red point. Then we apply the step 4.4 to obtain the widest separating slice. As a consequence we have the following theorem.

10 Ferran Hurtado, Carlos Seara, Saurabh Sethia

Theorem 4. *Let B and R be two disjoint point sets in \mathbb{R}^3 . The widest separating slice can be computed in $O(n^3)$ time, if we have pre-computed the set of slice directions of the separating slices.*

The narrowest separating slice: The narrowest separating slice is given by two parallel planes passing through an antipodal facets pair of $CH(R)$ such that its normal vector belongs to the set of slice directions of separating slices. Thus, the width of the narrowest slice is the width of $CH(R)$ restricted to the set of slice directions of separating slices.

Houle and Toussaint¹⁰ gave an algorithm for computing the width of a point set or a convex polyhedron in 3D which runs in $O(n^2)$ time in the worst case. There is also an algorithm by Chazelle et al.⁶ which runs in $O(n^{8/5+\epsilon})$ deterministic time for any $\epsilon > 0$. We will use a variation of the former algorithm to solve the problem.

Houle and Toussaint's algorithm only consider parallel planes of support passing through an antipodal vertex-face pair or an antipodal edge-edge pair of $CH(R)$, because other antipodal facet pairs can be reduced to these two ones. In our case, other possible pairs of facets can be consider since we have the constraints of blue points. The algorithm is as follows.

- (4.5) Assume that we have computed the set of (at most $O(n^2)$) convex polygons on the plane $z = 1$ corresponding to slice directions, which has $O(n^2)$ total complexity. Then, in $O(n^2 \log n)$ time, we preprocess these polygons in an $O(n^2)$ -vertices planar subdivision such that we can do point-location in $O(\log n)$ time. Each edge of a convex polygon corresponds to a supporting plane of $CH(R)$ which defines a separating slice (Figure 6).
- (4.6) For each edge e of the convex polygons, we do the following. Assume that e contains the information of the corresponding supporting plane π_e . In $O(\log n)$ time we compute the other supporting plane of $CH(R)$ which is parallel to π_e , compute the width defined by these parallel planes, and maintain the minimum width.
- (4.7) Apply Houle and Toussaint's algorithm as follows. For each antipodal vertex-face pair or antipodal edge-edge pair of $CH(R)$, using point-location check whether the direction of the plane defined by the current pair belongs to the set of slice directions. In the affirmative case, compute the corresponding width and maintain the minimum.

Theorem 5. *Let B and R be two disjoint point sets in \mathbb{R}^3 . The narrowest separating slice can be computed in $O(n^2 \log n)$ time, if we have pre-computed the set of slice directions of the separating slices.*

3.2. Wedge separability

A natural extension for wedge separability in the plane is the separability by a 3D wedge which is defined as follows. Two intersecting planes divide the 3D space into four quadrants. Any one quadrant is called a wedge. The *aperture angle* of the

wedge is the angle of its planes (Figure 7). The wedge separability problem is the following: Is there a wedge that contains R but does not contain any blue point or vice versa?

Theorem 6. *Let B and R be two point sets in \mathbb{R}^3 . Deciding whether the sets are wedge separable can be done in $O(n^3)$ time.*

Proof. Consider a separating wedge with a fixed aperture angle θ . We can move the two planes in parallel to themselves until they touch a red point of $CH(R)$. Let π_1, π_2 be the two planes and r_1, r_2 the respectively touched red points. Project r_2 on π_1 and construct the circle showed in Figure 7. Now rotate plane π_2 around the circle keeping the touching point r_2 on it (in this way we maintain the aperture angle θ) until π_2 bumps into a new (red or blue) point. Now do the same with plane π_1 . Thus we have two points on each of the planes of the wedge. The two planes can now be rotated simultaneously about the lines passing through their respective two points such that the aperture angle between them remains θ . We can do this until one of the planes bumps into a new red or blue point. Thus at least one of the planes will have three points on it of which at least one would be red.

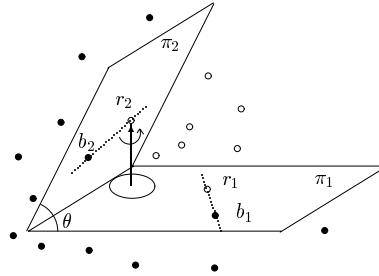


Fig. 7. Wedge separability.

Notice that there is still one degree of freedom as we rotate the other plane around the line defined by the two points until it bumps into a new point; but doing this, the value θ will be increased or decreased and then, the two planes of the wedge will have three points on it of which at least one will be from $CH(R)$. This observation leads to the following algorithm.

- (1) For each pair of points compute the line l passing through them.
- (2) Assuming a hierarchical representation of $CH(R)$, compute the at most two supporting planes between l and $CH(R)$ in $O(\log n)$ time. For each of these planes do the following.
 - (2.1) Let the plane under consideration be the plane π_1 of the wedge. Assume that π_1 is the plane $z = 0$ such that $CH(R)$ is above π_1 . Compute the set B_1 of blue points above π_1 .

12 Ferran Hurtado, Carlos Seara, Saurabh Sethia

- (2.2) In $O(n)$ time compute (if it exists) a separating plane between B_1 and $CH(R)$. It will be the second plane of the wedge.

This gives an $O(n^3)$ time algorithm for deciding the wedge separability. \square

3.2.1. Wedges with maximum and minimum aperture angle

Suppose that B and R are wedge separable but not line or slice separable. It is natural to ask about the separating wedges with maximum and minimum aperture angle. A wedge with minimum aperture angle is related to slice separability (a slice can be considered as a wedge with aperture angle 0°) and a wedge with maximum aperture angle is related to linear separability (if the aperture angle is close to 180° then, we have a good approximation to linear separability).

By the proof of Theorem 6 we know that if B and R are wedge separable, then there exists a wedge with minimum (or maximum) aperture angle that has one of its planes π_1 passing through three points, at least one of which is red. The intersection line of the planes of a wedge is called the *wedge-axis*. The next procedure first computes a region in the plane π_1 , which is the locus of points of wedge-axes of separating wedges that share π_1 , where π_1 produces the partition B_1 , $B_2 = B - B_1$; and second it computes the wedges with maximum and minimum aperture angle for this situation. The procedure is as follows.

- (2.3) If π_1 produces a wedge solution, B_1 is line separable from R . Since B and R are not slice separable, there exists no plane parallel to π_1 separating B_1 from R . In $O(n \log n)$ time compute the boundary, $\mathcal{B}(B_1, R)$, of the region of points of separating planes between B_1 and R (see Section 2), which is formed by two unbounded convex polygonal regions with $O(n)$ complexity.
- (2.4) Compute $\mathcal{B}(B_1, R) \cap \pi_1$, which is the locus of points of wedge-axes of the separating wedges with one of its half-planes contained in π_1 and such that π_1 produces the partition B_1, B_2 . The boundary of $\mathcal{B}(B_1, R) \cap \pi_1$ is formed by two concave polygonal chains, P_1 and P_2 (Figure 8).
- (2.5) The wedge-axis of the separating wedge with minimum (maximum) aperture angle either contains one of the at most $O(n)$ edges of P_1 (P_2) or the wedge-axis only passes through a vertex p in P_1 (P_2); otherwise we can *close* (*open*) the separating wedge decreasing (increasing) the aperture angle. In the second case, the interior supporting line between $CH(B_1)$ and $CH(R)$ which defines vertex p is perpendicular to the wedge-axis and, as we rotate the second plane of the wedge around this supporting line, the aperture angle of the wedge always increases or always decreases; so the wedge-axis corresponding to a vertex p can be computed in constant time. Thus, the separating wedges with maximum and minimum aperture angle can be computed in $O(n \log n)$ time.

To compute the separating wedges with maximum and minimum aperture, we do the steps above maintaining the maximum and the minimum aperture angle.

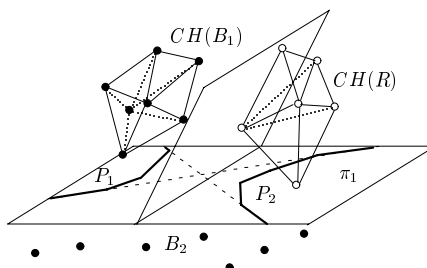


Fig. 8. Two concave polygonal chains.

Theorem 7. *Let B and R be two point sets in \mathbb{R}^3 . Computing the separating wedges with maximum and minimum aperture angle can be done in $O(n^3 \log n)$ time.*

3.2.2. Wedges with fixed aperture angle

A natural problem is to decide whether there exists a separating wedge with a fixed aperture angle θ , $0^\circ < \theta < 180^\circ$. Notice that if we have pre-computed the separating wedges with maximum and minimum aperture angle, the problem is not solved because it may be that B and R are not wedge separable for all possible values of aperture angle between the minimum and the maximum.

Nevertheless, the problem can be solved in $O(n^3 \log n)$ time by the algorithm in Theorem 7 with the following additional computation in step 2.5: check whether θ is in between the minimum and the maximum aperture angles obtained with P_1 , P_2 . The existence of a separating wedge with aperture angle θ is guaranteed by the continuity of the solutions. We can also consider the problem of reporting a solution (if it exists). The following alternative $O(n^3 \log n)$ time algorithm solves this problem.

- (2.4') Assume that π_1 is the plane $z = 0$ with normal vector $(0, 0, 1)$ such that R and B_1 are above π_1 . Compute the circle c on the unit sphere defined by the directions of planes which form an angle θ with vector $(0, 0, 1)$.
- (2.5') Compute the convex region on the unit sphere formed by the directions of separating planes between R and B_1 (see Section 2). Compute the intersection of this convex region with circle c and report a separating wedge with aperture angle θ if the intersection is not empty. To report a solution, choose a direction of the intersection and compute a separating plane with this direction in $O(n)$ time.

For $\theta = 90^\circ$ we solve the problem in $O(n^3)$ time as follows: let π_1 be the plane $z = 0$ with normal vector $(0, 0, 1)$ and the sets R and B_1 are above π_1 . Compute plane π_2 in linear time by projecting B_1 and R on π_1 and computing a line (if it exists) which separates the projected blue points from the projected red points.

14 Ferran Hurtado, Carlos Seara, Saurabh Sethia

Theorem 8. *Let B and R be two point sets in \mathbb{R}^3 . Computing a separating wedge with fixed aperture angle can be done in $O(n^3 \log n)$ time.*

3.3. Diwedge separability

A natural extension for double wedge separability in the plane is the separability by a 3D *diwedge* which is defined as follows. Two intersecting planes divide the 3D space into four quadrants. The union of a pair of opposite quadrants is called a *diwedge*. The line of intersection of the planes of the diwedge is called as the *diwedge-axis*. We define the *aperture angle* of a diwedge to be the bigger of the two aperture angles defined by the planes of the diwedge (Figure 9). The diwedge separability problem is the following: Is there a diwedge that contains R but does not contain any point from B or vice versa? We can consider that each opposite quadrant has at least one point, otherwise it will be a particular case of wedge separability.

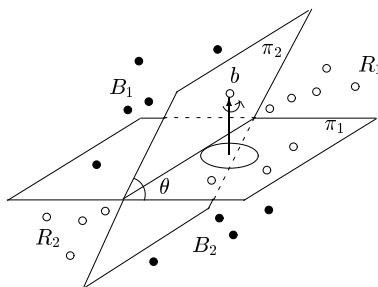


Fig. 9. Diwedge separability.

Theorem 9. *Let B and R be two sets of points in \mathbb{R}^3 . Deciding whether the sets are diwedge separable can be done in $O(n^4)$ time.*

Proof. Assume that the sets are not line separable. Consider a separating diwedge with fixed aperture angle θ formed by the planes π_1 and π_2 , where π_1 is a horizontal plane. We can move π_2 in parallel to itself until it bumps into a red or blue point b . Project b on π_1 and construct the circle showed in Figure 9. Rotate π_2 around the circle keeping the point b on it (maintaining the aperture angle θ) until π_2 bumps into a new (red or blue) point.

Proceeding analogously with plane π_2 , we get two points on each of the planes of the diwedge. Rotate the two planes simultaneously about the lines passing through their respective two points such that the aperture angle between them remains θ . We do this until one of the planes bumps into a new (red or blue) point. Thus, if the sets are diwedge separable, there exists an equivalent solution (same bipartition

of the point sets) such that one of the planes is defined by three points. This leads to the following algorithm.

- (1) For each plane, π_1 , defined by three points, do the following.
 - (1.1) In $O(n)$ time, classify the points as: R_1 (B_1) red (blue) points above π_1 , and R_2 (B_2) red (blue) points below π_1 .
 - (1.2) The second plane, π_2 , (if it exists) is a plane which separates $B_1 \cup R_2$ from $R_1 \cup B_2$. It can be obtained by linear programming in $O(n)$ time.

This gives an $O(n^4)$ time algorithm for deciding diwedge separability. Notice that as we rotate a plane around the diwedge-axis, the rotational order of the colored subsets of points has to be red/blue/red/blue, since we have assumed that the sets are not line separable. \square

3.3.1. Diwedges with maximum, minimum and fixed aperture angle

According to the definition of aperture angle, the values of the maximum and minimum aperture angles are in between 90° and 180° . By the proof of Theorem 9 we know that one of the planes of the separating diwedge with maximum aperture angle has three points on it. Assume that π_1 is this (horizontal) plane and let R_1 and B_1 (R_2 and B_2) be the set of red and blue points above (below) π_1 , respectively. From this point we can proceed analogously as the algorithm in Theorem 7 with evident changes as follows.

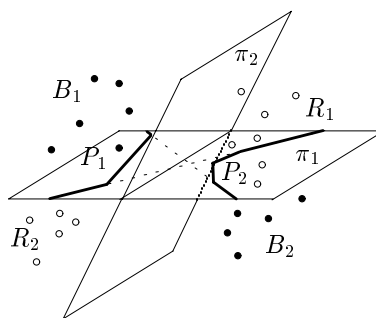


Fig. 10. Two concave polygonal chains.

If π_1 produces a solution, i.e., $B_1 \cup R_2$ is line separable from $R_1 \cup B_2$, then the concave polygonal chains P_1 and P_2 define the locus of points of diwedge-axes of the separating diwedges with one of its planes contained in π_1 and such that π_1 produces the partition $B_1 \cup R_1, B_2 \cup R_2$ (Figure 10). The diwedge-axis of the separating diwedge with maximum aperture angle either contains one of the edges of P_1 or P_2 or it passes through a vertex p in P_1 or P_2 . If M_0 is the maximum

aperture angle for the wedge as in Theorem 7, then the maximum aperture angle for a separating diwedge is $\max\{M_0, 180^\circ - M_0\}$. Thus, we can compute the separating diwedge with maximum aperture angle for this situation.

The minimum aperture angle of a separating diwedge is either 90° (π_2 is perpendicular to π_1) or bigger than 90° . The first case is easy and in the second case we proceed as above. The existence of a separating diwedge with fixed aperture angle θ , $90^\circ \leq \theta < 180^\circ$ and the problem of reporting a solution can be solved similarly as in Theorem 7 with the evident changes.

Theorem 10. *Let B and R be two point sets in \mathbb{R}^3 . A separating diwedge with maximum, minimum or fixed aperture angle can be computed in $O(n^4 \log n)$ time.*

4. Separability by a large number of concurrent planes

In this section we study other separability criteria which can be consider also as extensions for slice, wedge and double wedge separability in the plane, but allowing a large number of planes. More precisely, we extend the concepts above to prismatic, pyramidal and dipyramidal separability, respectively. In all the cases, we will show algorithms for solving the decision problems.

4.1. Prismatic separability

A prism is defined as the space swept by a convex polygon when it is moved along a line perpendicular to its plane; the direction of this line is called the *prism direction*. The prismatic separability problem asks: Is there an infinite prism which contains all red points but none of blue points or vice versa?

Theorem 11. *Let B and R be two sets of points in \mathbb{R}^3 . Deciding prismatic separability and computing a representation of the set of prism directions of separating prisms can be done in $O(n^3)$ time. The representation has at most $O(n^2)$ connected components with $O(n^2 \alpha(n))$ total complexity.*

Proof. A solution for prismatic separability can be described by the *prism direction*. The presence of a blue point rules out a certain set of direction vectors. This set is given by the dipyramid (two symmetrical pyramids having a common apex) whose apex is the blue point and whose planes are tangent to the red convex hull (Figure 11). Each pyramid of this dipyramid represents a set of directions which cannot be the solution, otherwise the point will be contained in the prism. On the unit sphere these directions occupy two convex regions. We get two such convex regions for every blue point. The complement of the union of all these regions is the set of direction vectors that correspond to a solution. We project the direction vectors on the unit sphere to two planes at $z = 1$ and $z = -1$. The convex regions are now (possibly unbounded) convex polygons. There are $O(n)$ convex polygons, each of size $O(n)$.

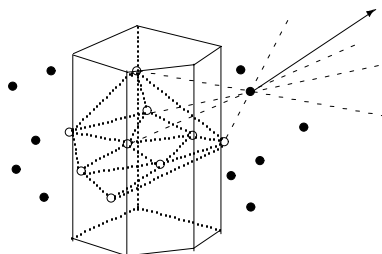


Fig. 11. Prismatic separability.

These polygons can be computed in $O(n)$ time each, taking a total of $O(n^2)$ time. This complexity would be dominated by the time to compute the complement of the union of these polygons. One way to compute the complement of the union of these polygons is by computing their arrangement. To compute the arrangement we use Chazelle and Edelsbrunner's algorithm⁵ with $O(m \log m + k)$ running time, where m is the total number of segments and k is the size of the arrangement. In our case m is $O(n^2)$ and k is $O(n^3)$. Thus we can compute the arrangement and hence the union (representation of prism directions of separating prisms) in $O(n^3)$ time with the following algorithm.

- (1) For each blue point, compute the dipyrmaid whose apex is the blue point and whose planes are tangent to $CH(R)$.
- (2) Compute set of directions on the unit sphere represented by each pyramid of these dipyrmaid. Project the direction vectors on the unit sphere to two planes at $z = 1$ and $z = -1$, obtaining $O(n)$ (possible unbounded) convex polygons, each of size $O(n)$.
- (3) Compute the complement of the union of these polygons by computing their arrangement, and hence the union in $O(n^3)$ time. The sets B and R are prismatic separable if, and only if, the complement of the union is not empty.

Now we consider the question of knowing how many different regions (connected components) corresponding to prism directions of separating prisms can exist. The number of different regions is the number of connected components in the complement of the union. In³ the following result by Katona¹⁴ and Kovalev is mentioned: *the common exterior (complement of the union) of K compact convex bodies in the plane has at most $\binom{K-1}{2} + 1 = O(K^2)$ components*, which in our case says that the number of connected components is at most $O(n^2)$.

Also, the following results by Aronov and Sharir³ can be applied: i) *the maximum complexity of the entire common exterior (the complement of the union) of K convex polygons in the plane with a total of N edges is $\Theta(N\alpha(K) + K^2)$* , ii) *the maximum number of edges bounding any single face of the arrangement of K convex polygons with a total of N edges is $O(N\alpha(K))$, the same bound holds for*

18 Ferran Hurtado, Carlos Seara, Saurabh Sethia

any face of the arrangement. In our case K is $O(n)$ and N is $O(n^2)$, therefore the complexity of the complement of the union is $O(n^2\alpha(n) + n^2) = O(n^2\alpha(n))$, and any connected component of the complement of the union has at most $O(n^2\alpha(n))$ complexity. Therefore, the representation of the set of prism directions of separating prisms is formed by at most $O(n^2)$ connected components with total complexity $\Theta(n^2\alpha(n))$. \square

Open problem. A still open problem is how can we find a minimum (in number of faces) separating prism? The minimum prism has at least three faces, hence this must address the question of deciding triangular prismatic separability which will be consider in Section 5. We can compute the minimum separating prism for a given prism direction u , which is equivalent to compute the minimum (in number of edges) convex polygon separating the projected red and blue points on a plane with normal vector u ; this problem can be solved in $\Theta(n \log n)$ optimal time ^{2,8}.

4.2. Pyramidal separability

Join all the vertices of a convex polygon to a point in space to get a pyramid. The convex polygon is called the base of the pyramid while the point in space is called its apex. An infinite pyramid is one whose base is at infinity. The pyramidal separability problem asks the question: Is there an infinite pyramid that contains all the red points but none of the blue points or vice versa?

Theorem 12. *Let B and R be two sets of points in \mathbb{R}^3 . Deciding pyramidal separability and computing a region in \mathbb{R}^3 formed by the apices of separating pyramids can be done with a randomized algorithm whose expected running time is $O(n^3 \log^2 n)$. The region of apices has at most $O(n^3)$ connected components and $O(n^3 \log n)$ total complexity.*

Proof. A solution to pyramidal separability can be stated by the apex point of the pyramid. The presence of a blue point rules out a certain region in 3D where the solution can not be there. Consider the dipyrmaid subtended by the blue point on the red convex hull (Figure 12). The pyramid of this dipyrmaid that does not contain the red convex hull can not contain the solution. Also if we consider the blue point as a light source, the shadow of the red convex hull can not contain the solution. The complement of the union of these ruled out unbounded convex polyhedra for all the blue points gives the region defined by the set of apices of all feasible solutions. There are $O(n)$ 3D unbounded convex polyhedra each of complexity $O(n)$. The complexity of computing the union of these unbounded convex polyhedra would dominate.

One way to compute the union is by the following result from Aronov, Sharir and Tagansky ⁴: *the number of vertices, edges and faces of the union of K convex polyhedra in three dimensional space, having a total of N faces is $O(K^3 + NK \log K)$.*

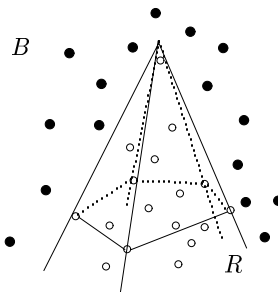


Fig. 12. Pyramidal separability.

This bound is almost tight in the worst case, as there exists collections of polyhedra with $\Omega(K^3 + KN\alpha(K))$ union complexity. They also describe a rather simple randomized incremental algorithm for computing the boundary of the union in $O(K^3 + KN \log K \log N)$ expected time. In our case, K is $O(n)$ and N is $O(n^2)$; therefore the complexity of the union of the unbounded convex polyhedra above is $O(n^3 + n^3 \log n) = O(n^3 \log n)$ and, the boundary of the union can be computed in $O(n^3 \log^2 n)$ expected time. The complement of the union is the region of apices of separating pyramids. The point sets B and R are pyramidal separable if, and only if, the complement of the union is not empty.

Now we consider the question of knowing how many different regions (connected components) corresponding to apices of separating pyramids can exist. It is clear that this number is the number of connected components of the complement of the union of the convex polyhedra above. A result by Katona¹⁴ says that *the complement of the union of K convex polyhedra in d -space has at most $O(K^d)$ connected components*, which in our case say that the number of regions of apices of separating pyramids is at most $O(n^3)$. \square

In⁴, to obtain an efficient deterministic algorithm for constructing the union of convex polyhedra is presented as an open problem. Next theorem shows a very simple (but with high complexity) deterministic algorithm for solving the decision problem of pyramidal separability.

Theorem 13. *Let B and R be two sets of points in \mathbb{R}^3 . Deciding whether the sets are pyramidal separable can be done in $O(n^7)$ time.*

Proof. A solution for pyramidal separability can be described by the *apex* of the pyramid, because for any solution we can bring the faces of the pyramid closer until each face touches an edge of $CH(R)$. And for a given apex the planes passing through the apex and touching red edges of the convex hull are unique. Assume that the point sets are pyramidal separable by a pyramid P with apex p such that each face of P contains an edge $CH(R)$. Now, we move p with three degrees of freedom until a face of P bumps into a new red or blue point q_1 defining the plane π_1 of a face f_1 containing a red edge. If we want the face f_1 to keep touching the

20 Ferran Hurtado, Carlos Seara, Saurabh Sethia

point q_1 , we move the pyramid by moving its apex on the plane π_1 until some other face of the pyramid bumps into another red or blue point q_2 , defining a plane π_2 of other face f_2 which will contain a red edge of $CH(R)$. Let l_{12} be the intersecting line of the planes π_1 and π_2 . We can still move the pyramid moving the apex on the line l_{12} until some other face f_3 of the pyramid bumps into a new red or blue point q_3 , defining a plane π_3 of a third face f_3 which will contain a red edge of $CH(R)$. Let l_{23} be the intersecting line of the planes π_2 and π_3 . Now, the intersecting point of the lines l_{12} and l_{23} is the apex of a possible separating pyramid. We can check all such apices and, if none of them is a solution then, we would be sure that no solution exists. Thus, we get the following algorithm.

- (1) Compute three planes π_1 , π_2 and π_3 given by three edges of $CH(R)$ and three red or blue points, respectively. This gives $O(n^6)$ candidates.
- (2) Compute the intersecting lines of two different pairs of planes. The apex of the possible separating pyramid is the intersection point of these lines (if it exists). In $O(n)$ time compute the cone defined by this apex and $CH(R)$, and check whether or not the cone contains blue points.

This gives an $O(n^7)$ time algorithm for deciding pyramidal separability. \square

4.3. Dipyramidal separability

Dipyramidal separability has the same definition as pyramidal separability except that now we have two symmetrical pyramids having the same apex. Assume that the red points R are inside the possible separating dipyrmaid which produces a partition of R (Figure 13).

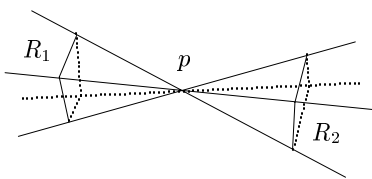


Fig. 13. Dipyramidal separability.

Theorem 14. *Let B and R be two point sets in \mathbb{R}^3 . Deciding whether the sets are dipyramidal separable can be done in $O(n^8 \log n)$ time.*

Proof. Assume that there exists a separating dipyrmaid with apex p . It is easy to see that the following conditions satisfy: i) any plane defining a face of the dipyrmaid must lie on a red point r_i , because we can always *close* the dipyrmaid until each face bumps into a red point; ii) every plane defining a face can be made to lie on

two points, since the line $r_i p$ is contained in the face then we can rotate the plane on the line $r_i p$ until the plane bumps a new point (red or blue); iii) the apex point has three degrees of freedom and we can move the apex until at least three planes have three points each of them (at least one of them from R for each plane). These conditions gives the following algorithm.

- (1) Choose any triple of points and assume that π_1 is a horizontal plane containing these points. Plane π_1 splits R into R_1 (red points above π_1) and R_2 (red points below π_1).
- (2) The other two planes can be obtained by choosing any pair of points and finding the supporting planes to R_1 or R_2 passing through these points. The intersection of these three planes is the candidate apex point p . So the number of candidates is $O(n^7)$ because there are $O(n^3)$ chooses for the first plane and $O(n^2)$ chooses for the second and third planes. Basically for $O(n^7)$ candidates we spend at most $O(n \log n)$ time per candidate to compute the apex. Once we have the apex p , we do the following.

- (2.1) Compute $R'_1 = CH(R_1 \cup p)$, $R'_2 = CH(R_2 \cup p)$.
- (2.2) Compute the interior supporting planes between R'_1 and R'_2 in $O(n)$ time using the algorithm by Davis ⁷. Note that R'_1 and R'_2 are separable by any of the three planes above (each one pass through point p). The interior supporting planes define two opposite cones C_1 and C_2 .
- (2.3) For any blue point $b \in B$, check whether the half-line \overrightarrow{pb} intersects either R'_1 or R'_2 in $O(\log n)$ time per blue point. In this way we can check that there are not blue points inside the cones C_1 and C_2 , otherwise we choose another candidate.

Since the steps (2.1), (2.2) and (2.3) can be performed in additional $O(n \log n)$ time, the total running time of the algorithm is $O(n^8 \log n)$. \square

5. Separability by a constant number of planes

In this section we study some particular separability criteria which involve three to six planes. More precisely, we consider triplane, triangular prism, tetrahedral and box separability.

5.1. Triplane separability

Three intersecting planes divide the 3D space into eight octants. The triplane separability problem asks the question: Are there three planes such that each of the octants they define has points of only one color?

Theorem 15. *Let B and R be two sets of points in \mathbb{R}^3 . Deciding whether the sets are triplane separable can be done in $O(n^7)$ time.*

Proof. Assume that the sets are separable by three planes such that the octants they define have points (at least one) of only one color. It is clear that we can move

22 Ferran Hurtado, Carlos Seara, Saurabh Sethia

around the normal vector of each one of the planes until each plane touch three (red or blue) points of the sets. To decide triplane separability we proceed as follows.

- (1) For each pair of planes define each one by three points of the sets ($O(n^6)$ candidates), and label the four quadrants they define by 1, 2, 3, 4. Let B_i (R_i) be the subsets of B (R) located in quadrant i , where $i = 1, 2, 3, 4$; these subsets can be computed in $O(n)$ time.
- (2) Select one of the constant number of combinations of colors for each octant. Then, in $O(n)$ time, check whether the union of the corresponding subsets B_i or R_i that have to be on each half-space defined by the third plane are line separable. In the affirmative case, we obtain the third plane.

The time complexity of the algorithm is $O(n^7)$, since we have $O(n^6)$ candidates for the first and second planes, and the third plane can be computed in $O(n)$ time. \square

A particular case is to decide separability by three planes perpendicular to the axes such that the octants have points of only one color. This problem can be solved in $O(n^3)$ time as follows.

- (1) In $O(n \log n)$ time sort the points by x -coordinate, by y -coordinate and by z -coordinate. There are $\binom{n}{2}$ pairs of planes defined by the points sorted by the x -coordinate (first plane) and the y -coordinate (second plane).
- (2) For each pair of planes, there are n planes defined by the z -coordinate. Using the z -coordinate order, we check in constant time per update/delete of a point, whether the partition given by the three planes has monochromatic octants.

5.2. Triangular prismatic separability

The triangular prismatic separability problem asks the question: Is there an infinite triangular prism which contains all the red points but none of the blue points or vice versa?

Theorem 16. *Let B and R be two sets of points in \mathbb{R}^3 . Deciding whether the sets are triangular prismatic separable can be done in $O(n^5)$ time.*

Proof. Let the three planes forming the prism be π_1, π_2, π_3 . Also let the lines of intersections between them be l_{12}, l_{23}, l_{31} . Move all three planes in parallel towards the red convex hull until they touch red points r_1, r_2, r_3 . Move each plane keeping them in contact with r_i and about the line passing through r_i and parallel to the prism direction, until they each touch a (red or blue) point. Let these points be q_1, q_2, q_3 , respectively.

Infinitesimally rotate π_1 about r_1q_1 , this will change the direction of lines of intersection l_{12} and l_{31} . Rotate π_2 about r_2q_2 so that π_2 becomes parallel to l_{31} . This will get the three lines of intersection to be parallel again. We can thus rotate π_1 and π_2 simultaneously such that l_{12}, l_{23}, l_{31} remain parallel. Notice that π_3

remains unmoved during these rotations. Eventually either π_1 or π_2 will hit a point. Without loss of generality let's say π_1 hits a point q_4 . Now with π_1 stationary move π_2 and π_3 simultaneous like we moved π_1 and π_2 before. Eventually one of π_2 and π_3 , will hit another point. Without loss of generality let's say π_2 hits another point q_5 . Thus, if there is a solution, we can move the planes around to get a solution with π_1 and π_2 both incident on three points each (at least one of which belongs to $CH(R)$). This leads to the following algorithm.

- (1) We choose all combinations of two pairs of points to be on π_1 and π_2 . There are $O(n^4)$ such combinations. For a given q_1 and q_4 we can find two candidates for π_1 , these are the planes passing through q_1q_4 and tangent to $CH(R)$. Similarly we can find two candidates for π_2 . Thus for each choice of q_1, q_2, q_4, q_5 , there are four candidate pairs of π_1, π_2 .
- (2) For each candidate pair we find the line of intersection between π_1 and π_2 . Planes π_1, π_2 divide the space into four wedges. We remove all blue points that do not lie in the same wedge as R .
- (3) Now we project all points on a plane perpendicular to l_{12} . We then find whether there is a line separating the projected red and the blue points. If there is one then the plane passing through this line and parallel to l_{12} will be plane π_3 with π_1, π_2, π_3 forming a triangular prism that separates B and R .

The complexity of this algorithm is $O(n^5)$, because it runs the linear separability algorithm in the plane $O(n^4)$ times. \square

5.3. Tetrahedral separability

The tetrahedral separability problem asks the question: Is there a tetrahedron that contains all the red points but none of the blue points or vice versa?

Theorem 17. *Let B and R be two sets of points in \mathbb{R}^3 . Deciding whether the sets are tetrahedral separable can be done in $O(n^7)$ time.*

Proof. Suppose there exists a separating tetrahedron. We can bring the four faces f_1, f_2, f_3 and f_4 closer until they touch $CH(R)$ at red points r_1, r_2, r_3 and r_4 , respectively. If we want the face f_1 to keep touching r_1 , we move around the normal vector of f_1 on r_1 until f_1 bumps into a point b_1 (red or blue). Now, rotate f_1 around the line passing through r_1 and b_1 until the face f_1 bumps into a new point b'_1 (red or blue). Analogously we can proceed with the other faces. This ensures that the faces of the tetrahedron are defined by three points, given the next algorithm.

- (1) Determine the plane of a face f_1 as follows: take a pair of points and compute the line l_1 passing through these points. Assuming a hierarchical representation of $CH(R)$, in $O(\log n)$ time compute the two supporting planes between l_1 and $CH(R)$. Take one of these planes as a plane π_1 of the face f_1 . Proceed analogously for two other faces f_2, f_3 . We get a constant number of candidates for a selection of six points. This gives a total number of $O(n^6)$ possible candidates.

24 Ferran Hurtado, Carlos Seara, Saurabh Sethia

- (2) For each candidate, if the three planes intersect in a point forming an infinite pyramid P , then in $O(n)$ time compute the set B_1 of blue points inside P and compute (if it exists) a separating plane between B_1 and $CH(R)$. It will be the fourth plane of the face f_4 of the tetrahedron.

This gives an $O(n^7)$ time algorithm for deciding tetrahedral separability. \square

5.4. Box separability

The box separability problem asks the question: Is there an orthogonal box which contains all the red points but none of the blue points or vice versa?

Theorem 18. *Let B and R be two sets of points in \mathbb{R}^3 . Deciding whether the sets are box separable can be done in $O(n^7)$ time.*

Proof. Suppose there exists a separating box. We can bring the six faces f_i , $i = 1, \dots, 6$, closer until they touch $CH(R)$ into red points r_i , $i = 1, \dots, 6$, respectively. The six faces can be moved while maintaining contact with these red points. As we move face f_1 with two degrees of freedom, other faces can be moved simultaneously to maintain the box shape until one of the faces bumps into another point (red or blue). Now one of the faces is incident on two points. Without loss of generality, let this be face f_1 . Now keeping f_1 incident to those two points we can rotate it about the line joining the two points. Other faces can be moved simultaneously to maintain the box shape until one of the faces bumps into another point (red or blue). Depending on which face bumps into another point, there are three cases: 1) face f_1 bumps into a third point, 2) face opposite to face f_1 bumps into a second point, 3) face adjacent to face f_1 bumps into a second point.

In all three cases some of the faces can be rotated with one degree of freedom until one of them bumps into a new point (red or blue). Thus from a solution we can obtain an equivalent solution such that each face of the separating box is touching a red point and three other points (red or blue) are incident on the faces of the box. Without loss of generality there can be three cases:

- (1) Face f_1 is incident to three points (at least one belongs to $CH(R)$) and another face is incident to two points (at least one belongs to $CH(R)$).
- (2) There are three faces incident to two points each (at least one belongs to $CH(R)$) and there is a pair of opposite faces among these three faces.
- (3) There are three faces incident to two points each (at least one belongs to $CH(R)$) and these three faces share a common vertex.

Testing for the first case is easier. For every pair of points p , q find planes passing through p and q and tangent to $CH(R)$. There can be at most two such planes per pair, which can contain one of the faces of the box. For each of these planes find the slice parallel to these planes and touching the red convex hull. Now consider only the red and blue points between the slice planes. Project all these points on

one of the slice planes and find the answer to the question: Is there a rectangle that separates the red and the blue points? If there is such a separating rectangle there is a separating box. Answering this rectangular separability question can be done optimally in $\Theta(n \log n)$ time^{2,8}.

Testing for the second case is similar to the first case. Consider all combinations of two pairs of points p, q and r, s . With at least one point from $CH(R)$ in each pair. We are looking for a solution such that p, q are incident on one face and r, s are incident on the opposite face. The normal to these two faces can be found by finding the cross-product of vectors \vec{pq} and \vec{rs} . We find two planes with this normal touching the red convex hull and follow the algorithm for case 1.

Testing for the third case takes the most time. For this we consider all combinations of three pairs of points p, q, r, s , and u, v . For each combination we check if there is a separating box with three mutually adjacent faces incident to the three pairs of points. This can be done in linear time because there are only $O(1)$ such boxes per combination. We proceed as follows: Compute the equation of the plane passing through points p and q . Its normal vector, \vec{a} , depends on a parameter λ_1 which is bounded by some values, because the plane can not intersect $CH(R)$. Analogously, compute the equation of the plane passing through r and s ; its normal vector, \vec{b} , depends on a bounded parameter λ_2 . Do same computation for the plane passing through u and v , with normal vector, \vec{c} , depending on a bounded parameter λ_3 . The orthogonality condition for these planes imply the following equations with the dot-product of the normal vectors: $\vec{a} \cdot \vec{b} = 0$, $\vec{a} \cdot \vec{c} = 0$, and $\vec{b} \cdot \vec{c} = 0$. The solution of this equations system let us to know the planes of the three adjacent faces. We can compute the faces parallel to these ones which form the orthogonal box. Then in additional linear time we check that the box does not contain blue points. Thus, the third case checks $O(n^6)$ combinations in $O(n)$ time each and hence it takes $O(n^7)$ total time. \square

A particular case is to decide whether exists an orthogonal box with the faces parallel to the coordinate planes, which can be solved in $O(n)$ time as follows: (i) sweeping with a plane with normal vector the direction of the x -axis, compute in $O(n)$ time the first and the last red points and the corresponding planes with this normal vector passing through these red points; (ii) proceed analogously sweeping with planes with normal vector the direction of the y -axis and the z -axis, respectively. We get six perpendicular planes forming an orthogonal box. In $O(n)$ time check that this box contains no blue points.

Conclusions

A disadvantage of the deterministic approach is that it leads to high complexities. For example, pyramidal separability is decided in $O(n^7)$ time. We have also obtained for this problem an $O(n^3 \log^2 n)$ randomized algorithm and, as mentioned in the introduction, the nondeterministic approach has also been considered later for some other of the problems¹.

There are many more interesting questions on separability in 3D as compared to 2D. We have shown algorithms for some of them. Other possible extensions for slice, wedge and diwedge separability are the cylindrical, conical and diconical separability, which are defined by an infinite cylinder, by an infinite circular cone or by a pair of symmetrical infinite circular cones having the same apex, respectively and such that they separate the red and blue points.

References

1. P. K. Agarwal, B. Aronov, V. Koltun, Efficient algorithms for bichromatic separability, *Proceedings 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, (2004) pp. 675–683.
2. E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara, S. S. Skiena, Some lower bounds on geometric separability problems, *11th Fall Workshop on Computational Geometry*, (2001) submitted to IJCGA.
3. B. Aronov, M. Sharir, The common exterior of convex polygons in the plane, *Computational Geometry: Theory and Applications*, 8 (1997) pp. 139–149.
4. B. Aronov, M. Sharir, B. Tagansky, The union of convex polyhedral in three dimensions, *SIAM J. Comput.*, **26** 6 (1997) pp. 1670–1688.
5. B. Chazelle, H. Edelsbrunner, An optimal algorithm for intersecting line segments, *Journal of ACM*, **39** (1992) pp. 1–54.
6. B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, Diameter, width, closest line pair, and parametric searching, *Proceedings of the eighth Annual Symposium on Computational Geometry*, (1992) pp. 120–129.
7. G. Davis, Computing separating planes for a pair of disjoint polytopes, *Proceedings of the first Annual Symposium on Computational Geometry*, (1985) pp. 8–14.
8. H. Edelsbrunner, F. P. Preparata, Minimum polygonal separation, *Information and Computation*, **77** (1988) pp. 218–232.
9. M. E. Houle, Algorithms for weak and wide separation of sets, *Discrete Applied Mathematics*, **45** (1993) pp. 139–159.
10. M. E. Houle, G. T. Toussaint, Computing the width of a set, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **10** 5 (1988) pp. 761–765.
11. F. Hurtado, M. Noy, P. A. Ramos, C. Seara, Separating objects in the plane with wedges and strips, *Discrete Applied Mathematics*, **109** (2001) pp. 109–138.
12. F. Hurtado, M. Mora, P. A. Ramos, C. Seara, Separability by two lines and by nearly straight polygonal chains, *Discrete Applied Mathematics*, **144** (2004) pp. 110–122.
13. F. Hurtado, C. Seara, S. Sethia, Red-blue separability problems in 3D, *Third International Workshop on Computational Geometry and Applications*, Lecture Notes in Computer Science, Springer-Verlag, 2669 (2003) pp. 766–775.
14. G. O. Katona, On a problem of L. Fejes Tóth, *Stud. Sci. Math. Hung.*, **12** (1977) pp. 77–80.
15. N. Megiddo, Linear-time algorithms for linear programming in \mathbb{R}^3 and related problems, *SIAM Journal on Computing* **12** (1983) pp. 759–776.
16. F. P. Preparata, S. J. Hong, Convex hulls of finite sets of points in two and three dimensions, *Communications of the ACM* **20** (1977) pp. 87–93.