Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computational Geometry 34 (2006) 58–74
Computational Geometry
Theory and Applications
www.elsevier.com/locate/comgeo

A new 2D tessellation for angle problems: The polar diagram

Clara I. Grima [a], Alberto Márquez [a], Lidia Ortega [b,*]

[a] *Departamento de Matemática Aplicada I, Universidad de Seville, Spain*
[b] *Departamento de Informática, Universidad de Jaén, Spain*

Received 17 November 2003; received in revised form 21 November 2005; accepted 24 November 2005

Available online 4 January 2006

Communicated by S. Akl

Abstract

The new approach we propose in this paper is a plane partition with similar features to those of the Voronoi Diagram, but the Euclidean minimum distance criterion is replaced for the minimal angle criterion. The result is a new tessellation of the plane in regions called *Polar Diagram*, in which every site is owner of a polar region as the locus of points with smallest polar angle respect to this site.

We prove that polar diagrams, used as preprocessing, can be applied to many problems in Computational Geometry in order to speed up their processing times. Some of these applications are the convex hull, visibility problems, and path planning problems.

1. Introduction

The solution to many important problems in Computational Geometry requires some kind of angle processing of the data input. We find an example in the computation of the convex hull for a set of sites using Jarvis's march [12]. This method obtains the convex hull by solving a simple angle problem: if a point is known to be an extreme point, the next site in the convex polygon can be obtained by computing $n-1$ polar angles, assuming a data input of n sites. In the worst case, Jarvis's march runs in $O(n^2)$. The question now is: is it possible to improve this classical technique avoiding these linear time searches?

In this paper we propose a new locus approach for problems processing angles, the *polar diagram*. Using this tessellation as preprocessing, exhaustive searches to find those sites with smallest angle become unnecessary. This information is intrinsic to the polar diagram data structure, and provides a similar solution to the given for the Voronoi diagrams in proximity problems [17]. For any position q in the plane (represented by a point), the site with smallest polar angle, is the owner of the region where q lies into. Using this new partition, we are able to solve an $O(n)$ search problem in an optimal $O(\log n)$ location operation.

* Corresponding author.
E-mail address: lidia@ujaen.es (L. Ortega).

0925-7721/$ – see front matter
doi:10.1016/j.comgeo.2005.11.004

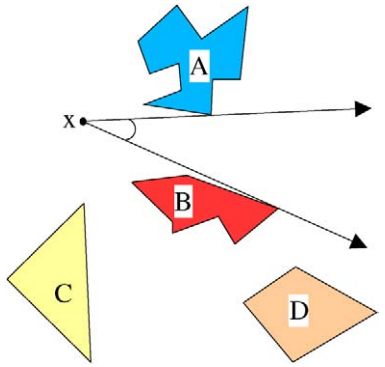


Fig. 1. Visibility angle from the point x.

As a result, the polar diagram principle can be used in some important problems requiring angle processing in Computational Geometry. Jarvis's march can be improved to become an optimal time process, as we see next. Visibility problems can take advantage of polar diagram principles as well. One of these visibility problems consists on finding the maximum visibility angle in any orthogonal direction, as illustrated in Fig. 1. This visibility problem is easy to be computed in linear time by performing an angular scanning using a clockwise and a counterclockwise criteria. However when this calculation is repetitive, it should be desirable to avoid all these exhaustive searches in order to obtain improved computation times. As an expected result, the polar diagram of a set of geometric objects becomes a powerful tool, avoiding the described angular sweeps.

This paper is structured as follows: in Section 2 we define the polar diagram of a set of sites and develop an optimal time algorithm for its construction. In Section 3, some visibility problems resolution, and the construction of the Convex Hull, justify the polar diagram calculation for a set of geometric objects. In Section 4 we study all these applications, and finally in Section 5 it is compared with some other classical methods that implement the convex hull to conclude that our proposal is one of the most robust and efficient techniques in the plane.

2. Definition and computation of the polar diagram

In this section we define the polar diagram of a set of sites as a plane partition called *polar regions*. There are at least two optimal methods for computing the polar diagram, following two of the most important paradigms in Computational Geometry: the Incremental and the Divide and Conquer. We describe in this paper the first one, that provides better computation times. The Divide and Conquer technique has been described in detail in [10].

2.1. Polar diagram definition

We introduce the polar diagram as a plane partition with similar features to those of the Voronoi diagram. In fact we see next, the polar diagram can be seen in the context of the generalized Voronoi diagram [17].

The polar angle of the point p with respect to s_i, denoted as $\text{ang}_{s_i}(p)$, is the angle formed by the positive horizontal line of p and the straight line linking p and s_i, as shown in Fig. 2. The polar angle has the restriction of being lower than π, thus p must be under the horizontal line defined by s_i.

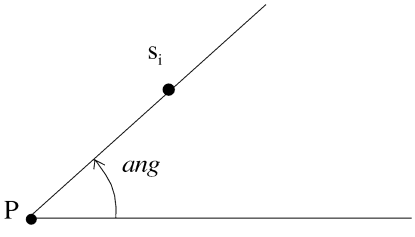


Fig. 2. Polar angle.

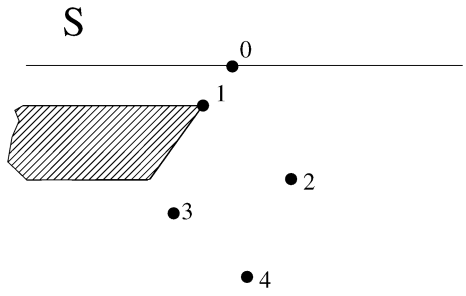


Fig. 3. Polar region of s_1.

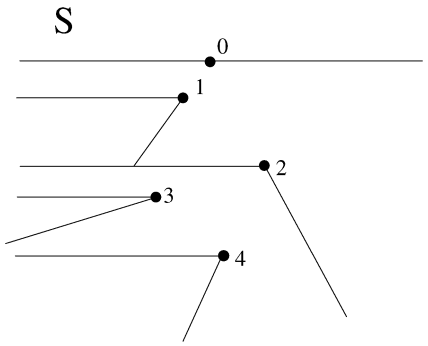


Fig. 4. Polar diagram of S.

Given a set S of n points in the plane, the locus of points with smaller positive polar angle with respect to $s_i \in S$ is called *polar region* of s_i, denoted $\mathcal{P}_S(s_i)$. Thus, $\mathcal{P}_S(s_i) = \{(x, y) \in E^2 \mid \text{ang}_{s_i}(x, y) < \text{ang}_{s_j}(x, y),\ \forall j \neq i\}$. The plane is divided into different regions in such a way that if the point $(x, y) \in E^2$ lies into $\mathcal{P}_S(s_i)$, it is known that s_i is the first site found performing an angular scanning starting from (x, y). We can draw an analogy between this angular sweep and the behavior of a radar [7].

This principle achieves an optimal processing time for Jarvis's March or Gift Wrapping method in $2D$ [12], as we see next. Angular scanning operations to find sites with the smallest polar angle are then avoided. The exhaustive search is replaced by locating a point in a polar region, similar to the Voronoi principle of proximity approach. Fig. 3 shows a set of sites in the plane, the marked area is the polar region of point s_1.

All $s_i \in S$ create a polar region and these n regions divide the plane defining a tessellation we have called *polar diagram* of S, denoted as $\mathcal{P}(S)$.

To summarize, we construct a plane tessellation with the polar angle criterion; actually, the polar diagram constructs a partition of the lowest half-plane. The boundary is the horizontal line crossing the top most site of S. Fig. 4 depicts the polar diagram of a set of points in the plane and the final division constructed using the smallest polar angle criterion. Each polar region, as observed in the figure, is constructed using two different half-lines or polar edges. There is always a horizontal edge starting at each s_i, $\{(x, y_i),\ x < x_i\}$, and another oblique polar edge with the gradient of the straight line crossing s_i and s_k, with $s_i \in \mathcal{P}_S(s_k)$.

In the generalized Voronoi diagrams framework, the polar diagram can be considered as one of these plane partitions. When a point lies in a Voronoi region, it is known the nearest site to this point. The minimal Euclidean distance criterion is replaced with the smallest polar angle in order to obtain a new plane division. Actually, the polar diagram can be considered a Voronoi generalization applying the following rule, with $p \in E^2$:

$$\partial(p, s_i) = \begin{cases} 1, & \text{if } \text{ang}_{s_i}(p) \leqslant \text{ang}_{s_j}(p) \text{ for all } j \neq i, \\ 0, & \text{in other case} \end{cases}$$

being $\text{ang}_{s_i}(p)$ the polar angle between p and s_i, and being ∂ the assignment rule.

2.2. Incremental algorithm

There are at least two optimal approaches for the polar diagram construction, the Incremental and the Divide and Conquer methods, both working in optimal time. However some tests carried out, show that the Incremental or Plane Sweep algorithm [7,10] described in this section, is always faster.

In this paradigm, a horizontal straight line sweeps the plane from top to bottom computing each point $s_i \in S$, being S a set of points in the plane. The polar region of s_i is built according to the following lemma:

Lemma 1. *Let S'_i denote the set of processed points when point s_i is reached, $S'_i = S'_{i-1} \cup s_i$. If $s_i \in \mathcal{P}_S(s_k)$, $s_k \in S'_{i-1}$, then the polar region of s_i is the angular sector defined by the horizontal half-line to its left, $\{(x, y_i),\ x < x_i\}$ and the half-line defined by s_i and s_k, which does not contain s_k.*

Proof. Let $\mathcal{P}_S(s_i)$ be the region described in the lemma. Let x be any point inside this region. The segment joining x with any point $s_q \in S'_i$ always intersects the horizontal polar edge of s_i. Thus, the polar angle of x with respect to s_q is always greater than the polar angle of x with respect to s_i. As a result, x can not lie into another polar region by the given definition, because s_i is the first site found in an angular scanning starting from x (see Fig. 5). □

Lemma 1 is the key to compute the polar diagram using the Incremental method. Algorithm 1 describes the process: S is sorted from top to down obtaining the sequence $\{s_0, s_1, \ldots, s_{n-1}\}$. The polar region of s_i is computed when $\mathcal{P}_S(s_0), \mathcal{P}_S(s_1), \ldots, \mathcal{P}_S(s_{i-1})$ have been already processed according to last lemma.

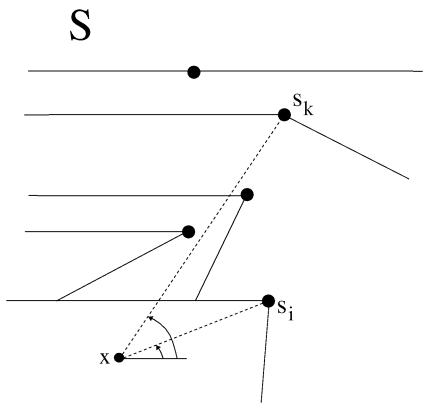


Fig. 5. Polar angle of x is lower with respect to s_i that with respect to any other point in S'.

```
 Input: A set S of n points in E^2
Output: P(S)
BEGIN
  1. Sort S by decreasing order, obtaining
     S = {s_0, s_1, ..., s_{n-1}}
  2. Let be S' = {s_0}
  3. S = S - {s_0}
  4. WHILE (S ≠ ∅) DO
     (a) Let s_i be the highest y-coordinate point in S
     (b) Do S' = S' ⋃ {s_i} and S = S - {s_i}
     (c) Construct P_S(s_i) according to Lemma 1
     (d) Discard all edges inside P_S(s_i)
     END-WHILE
  END
```

Algorithm 1. Incremental.

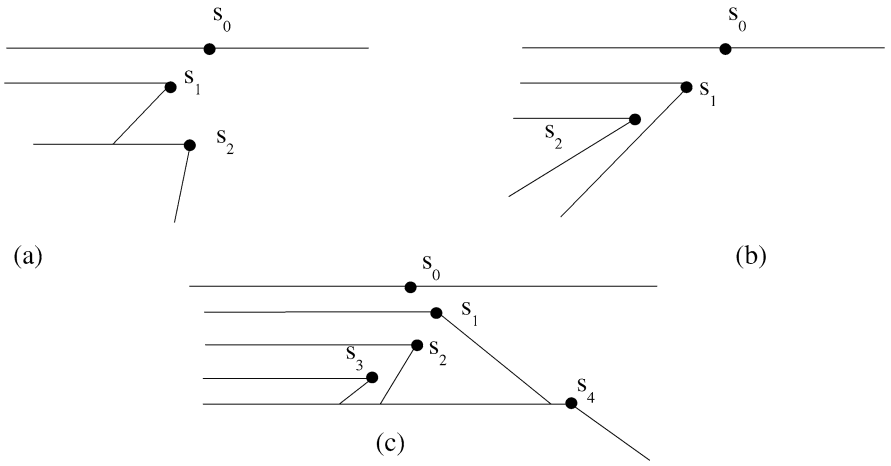


Fig. 6. New points positions after the Incremental algorithm processing.

Lemma 2. *The polar diagram of a set of sorted sites in the plane can be computed in linear time.*

Proof. In Fig. 6, s_1 can only belongs to $\mathcal{P}_S(s_0)$. Nevertheless, s_i, $i \neq 0$, $i \neq 1$, can be placed in different positions:

1. The case of point s_2, depicted in Fig. 6(a), is computed in constant time on account of the intersection between the oblique polar edge of s_1 and the horizontal one of s_2. Consequently, this point is known to be in the same region of s_1.
2. The case shown in Fig. 6(b) can be computed in constant time as well. The reason is just the opposite. There is no intersection between the last two polar edges processed, so s_2 is in the polar region of s_1.
3. We show a general case in Fig. 6(c). The polar region of s_4 is obtained after computing three intersections. Thus, in the worst case we need a linear running time. But we observe that once a point have been discarded, it is not processed again. Actually, the behavior of this algorithm is similar to the Graham's scan for the calculation of the convex hull [5], where a stack is used as local data structure. The candidate to be the first point seen in an angular sweep, is the top of the stack. If this candidate intersects in the way seen above, it will be taken out of the stack and never processed again. In the worst case, a candidate is processed only twice, for a push and a pop operation. Consequently, the compensated times of all these operations provide a linear time process. □

Theorem 1. *Given a set S of n points in the plane, the polar diagram of S, $\mathcal{P}(S)$, can be computed in $\Theta(n \log n)$ time using the Incremental method.*

Proof. We can solve a sorting operation is $O(n \log n)$ time and Lemma 1 ensures this time complexity after computing all polar regions. Finally, we know this is a lower bound because as we will see in following sections, it is possible to calculate the convex hull of a set of points in linear time using the polar diagram as a preprocessing. It is well known that the convex hull is an $\Theta(n \log n)$ problem. □

3. Polar diagram of geometric objects

As mentioned before, plane partitions have lots of applications fields for real problems. However, tessellation generators can not be considered point objects in most of realist representations. Thus, line segments, polygons and circles are chosen to model scenes in application fields like Proximity problems, Path Planning and Visibility or Illumination problems. Classical examples of tessellation in the plane are the Voronoi diagram or the trapezoidal maps. We now propose the polar diagram of geometric objects as a new plane partition with similar characteristics to the polar diagram of points [8,9].

The polar diagram definition for a set of geometric objects is similar to the given for points in the plane. Let $O = \{o_0, o_1, \ldots, o_{n-1}\}$ be a set of geometric objects, the polar region associated to o_i, $\mathcal{P}_O(o_i)$ is the locus of points

with smaller polar angle with respect to o_i than with respect to any other object of O, in a positive angular scanning starting from the zero angle: $\mathcal{P}(o_i) = \{(x, y) \in E^2 \mid \text{ang}_{o_i}(x, y) < \text{ang}_{o_j}(x, y),\ \forall j \neq i\}$, with $\text{ang}_{o_i}(x, y)$ being the positive polar angle of any (x, y) with respect to o_i.

Even though polar regions of points and geometric objects are really similar, there is a significant difference: polar regions associated to points are not bounded areas, but this property is not necessary true for geometric objects.

3.1. Polar diagram of line segments and polygons

Polygons and line segments have an important similarity when their polar diagrams are computed. In both cases, their polar edges start from extreme points or vertices, as the following lemma ensures:

Lemma 3. *Every polar edge associated to the polar diagram of a set of segments or a set of polygons in the plane, is contained into the polar diagram associated to the set of points made up of their line segments endpoints or polygons vertices respectively.*

Proof. As a matter of fact, any angular sweep starting from a point (x, y) always finds an endpoint. It could be possible to find two extreme points at the same time, but never a middle point of a line segment or a polygon edge. □

The polar diagram of sets of line segments and polygons has some common properties to the polar diagram of points. We can see an example of this plane partition for line segments in Fig. 7. However polar edges can intersect in this case with other segments, constructing bounded regions as well. An example of polar diagram polygons is illustrated in Fig. 8, where polar edges are observed to be associated only to some vertices.

This result gives us a method to calculate the polar diagram of segments and polygons. Firstly, the polar diagram of a set of distinguished points is computed using the Incremental method studied in Section 2, and then we add the following restrictions to discard some edges or portions of them:

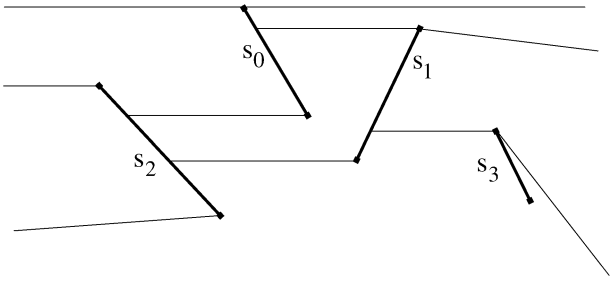


Fig. 7. Example of polar diagram of a set of segments.

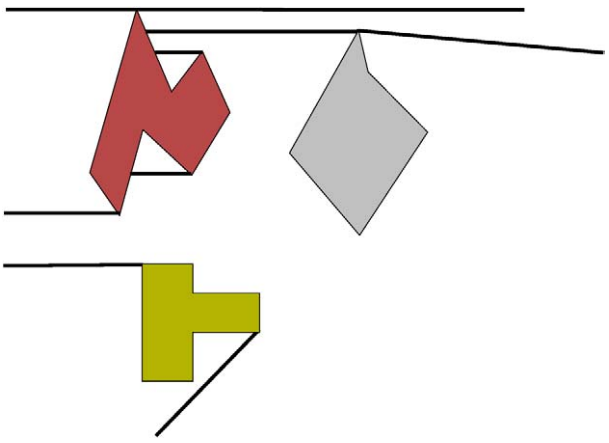


Fig. 8. Example of polygons polar diagram.

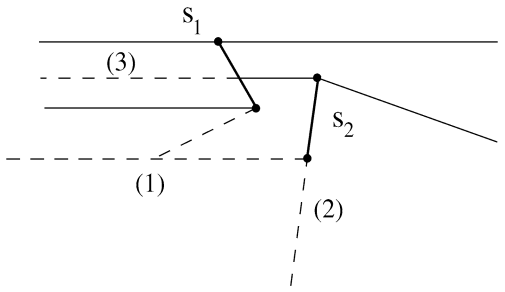


Fig. 9. Discarded edges in a polar diagram of line segments.

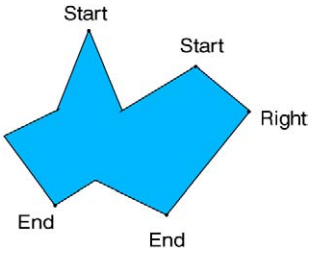


Fig. 10. Processed vertices.

1. If there is any obstacle to the right of an endpoint, the polar oblique edge is discarded.
2. A polar edge is eliminated if it splits two sectors of the same polar region.
3. If a portion of edge lies in another polar region, it must be discarded.
4. If a polar edge lies in the object it belongs (in polygons), it is discarded as well.

Fig. 9 shows an example of excluded polar edges (with dotted lines) in a polar diagram of a set of line segments. Those rules described above have been numbered according to these restrictions.

Nevertheless, this calculation process can be improved because not all vertices generate a polar edge. There are not edges associated to reflex vertices and neither to those in the left side of a polygon. Only the following vertices, illustrated in Fig. 10, have to be taken into account:

Start: Vertex v_i is a Start vertex if v_{i-1} and v_{i+1} have lower y-coordinate.
End: Vertex v_i is an End vertex if v_{i-1} and v_{i+1} have greater y-coordinate.
Right: Vertex v_i is a Right vertex if being a convex vertex, v_{i-1} has lower y-coordinate and v_{i+1} has greater y-coordinate than v_i.

As a result, we obtain an intuitive method for the polar diagram construction of sets of line segments and polygons. Even so, this two-steps technique is not necessary and a more efficient solution can be given by calculating the final set of polar edges at the same time that the Incremental is processed, deciding at each step if horizontal or oblique edges are discarded or not.

Theorem 2. *The polar diagram of a set of segments or polygons in the plane can be computed in* $\Theta(n \log n)$*, being* n *the number of line segments or polygons vertices.*

Proof. The polar diagram of the set of sites made up with the $2n$ segments endpoints can be computed in $O(n \log n)$ time according to Theorem 1. All these discarding operations need an additional $O(\log n)$ time to search neighbors to left and right of every endpoint. We work in a similar way in the polygons case, taking into account that n is the number of vertices. Any additional computation time to discover intersections does not alter the efficiency obtained, $O(n \log n)$, that is optimal as we proved. □

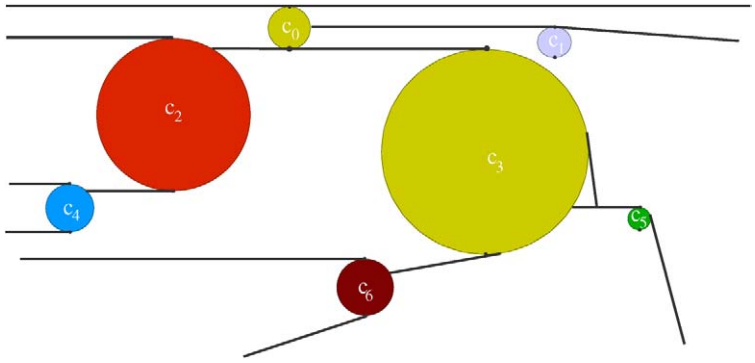


Fig. 11. Example of circles polar diagram.

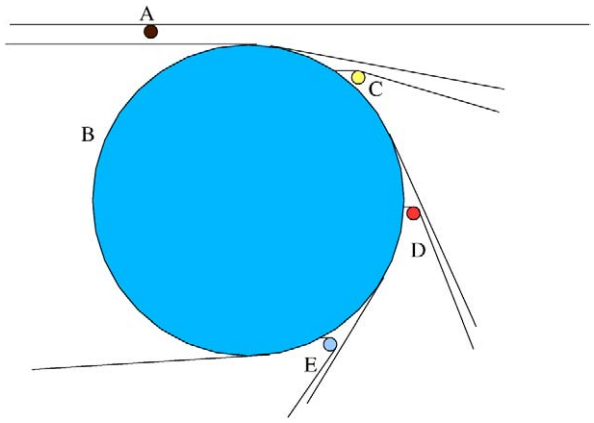


Fig. 12. Circle B has O(n) oblique edges.

3.2. Polar diagram of circles

Even when the polar diagram definition for a set of circles coincides with the one given for any set of geometric objects, we observe that there are no distinguished points to process, therefore it is not possible to know in advance where to focus the search, the number of final polar edges, and neither the position where they start from. An example of circles polar diagram is shown in Fig. 11.

It is straightforward to see that there is always a horizontal edge starting from the North pole in every circle. Sometimes there is another horizontal edge in the South pole if another obstacle is located just to its right. But the only certainty about oblique edges is that they start at some place in the right portion of a circle. Fig. 12 shows an example of polar diagrams in which circle B generates O(n) oblique edges. It could give us the impression that the resulting number of edges is quadratic. However the following result rules out this supposition.

Lemma 4. *The maximum number of polar edges associated to the polar diagram of a set of circles is* $3n - 3$.

Proof. Suppose all circles are reduced to point objects. It is straightforward to probe that the result is a planar graph, with a linear number of edges associated. Adding a point in the infinite, it is straightforward to prove using Euler's formula that this planar graph has as much $3(n + 1) - 6$ edges. □

Thus, we replace the study of significant endpoints with a set of $2n - 1$ horizontal strips. They are created by throwing horizontal infinite lines at each North and South pole. The set of $2n$ poles are sorted obtaining the sequence of points $\{p_0, \ldots, p_{2n-1}\}$. Every strip f_i is the locus of points included into the horizontal edges throw $[p_i, p_{i+1})$.

A strip contains only one pole, a North or a South pole. As much, the number of polar edges processed in every strip is two, a horizontal and an oblique edge. As mentioned before, there is always a horizontal edge starting from

a North pole, and another one starting from the South if an obstacle to the right is found. Oblique edges exist only in the right most position in a strip. Specifically, in the right portion of the most right circle, what reduces the edges search dramatically. Again, an additional search in $O(\log n)$ time to find the right most circle, or the one placed to the left, should be necessary. According to these premises we can give the following result:

Theorem 3. *The polar diagram of a set of n circles in the plane can be computed in an* $\Theta(n \log n)$ *time.*

To sum up, the polar diagram can be considered an optimal preprocessing of the plane to solve angle problems in Computational Geometry.

4. Polar diagram applications

The properties of the polar diagram make this tessellation an interesting preprocessing for angle problems in Computational Geometry. Some polar diagram applications we report here are the Convex Hull of a set of points and objects in the plane, Visibility problems in a horizontal direction and its generalization to any other direction, with interesting repercussions in applications like Path Planning and Collision Detection [11,16].

4.1. Convex hull

The convex hull is one of the most useful tools in Computational Geometry with applications in several scientific fields. Many optimal solutions have been proposed but, even when this is an $\Theta(n \log n)$ time problem, some non-optimal methods of construction, as Jarvis's March, can be considered as practical techniques [12].

In addition, the convex hull has been computed not only in the plane or in the space, but in different surfaces as we find in [3,4] and specially in [6]. Its computation is interesting for a set of objects [8,9], and there exist parallel methods for a performance improvement [1].

We conceive the polar diagram as a new plane preprocessing that can be used for the computation of the convex hull of points and geometric objects in the plane. We propose this new approach as a Jarvis's March improvement, in which an angular sweep to find the first site with the smallest positive polar angle becomes unnecessary. These exhaustive searches make Jarvis's March be a quadratic method in the worst case. In this section we see that using the polar diagram as a preprocessing, the computational cost of Javis's march can be improved to only linear time.

We now observe in Fig. 13(a) the relationship between the polar diagram and Jarvis's march. This classical technique begins with s_6, the lowest y-coordinate site which it is known to be in the convex hull. The following point, s_5, is found in linear time by computing a counterclockwise angular scanning. The rest of points in Jarvis's march are found following the same strategy until the right convex hull portion is completed. The process works in a similar way to obtain the left portion. Nevertheless, the successor site in Jarvis's march is just the one with smallest positive polar angle, an information given by the polar diagram. We know that s_5 is the following site in Jarvis's march because s_6 belongs to its polar region, that is, s_6 lies in $\mathcal{P}_S(s_5)$, information available in constant time. But processing Jarvis's

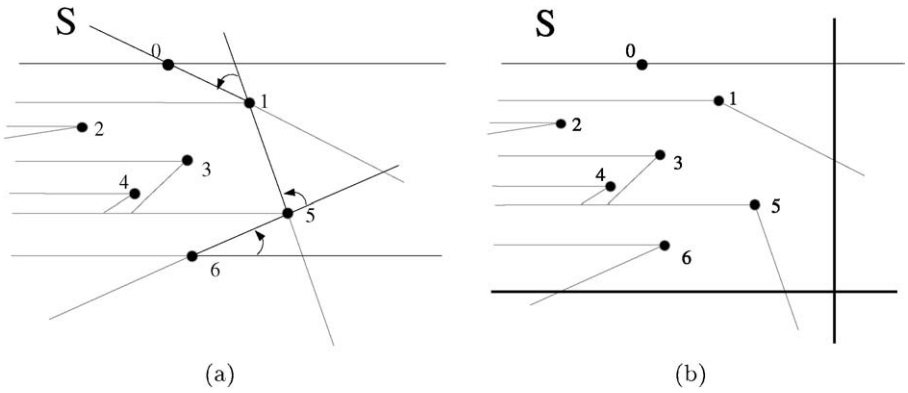


Fig. 13. Polar diagram and Jarvis's march relationship.

march after the polar diagram construction is not really necessary. In Fig. 13(b) it is straightforward to see that if we placed the set of points in the second quadrant, all those oblique edges intersecting with the coordinate axes start from sites belonging to the right portion of the convex hull, this is, all infinite oblique edges.

Lemma 5. *The convex hull of a set of n points in the plane can be computed, using the polar diagram as a preprocessing, in* $O(k)$ *time, being k the number of sites in the convex hull.*

Proof. We remove an oblique edge when it intersects with a horizontal one. It is straightforward to realize that the local data structure used, a stack, finally contains those edges not intersecting with others. The straight lines containing these infinite edges connect two consecutive points in the convex hull because they leave the rest of points to the left, just Jarvis's march principle. □

Nevertheless, a plane preprocessing is not the more accurate method to compute the convex hull, even more when this is the only final aim. For instance, Voronoi diagrams can be used to obtain the convex hull of their set of generator points in linear time, because all those sites lying in unbounded Voronoi regions, belong to the convex hull. However, any of the direct methods to compute convex hulls are faster than Voronoi diagrams. The very important difference with polar diagrams is that this new method does not need to perform a real plane partition into polar regions to obtain the convex hull, it suffices to follow the Incremental technique using a local stack as only data structure. Oblique and horizontal edges construction can be avoided in order to obtain a better running time.

The result is a very efficient method described in Algorithm 2, based in the Incremental technique for the polar diagram construction of Algorithm 1, but avoiding the tessellation process. The set S of sites is sorted by y-coordinate in decreasing order. The *stack* is used as local data structure to maintain the candidate to be the first site found in the described angular scanning. The *intersects* function detects if there is an intersection between the oblique edge starting from s_b (with the gradient of the line segment $s_a s_b$) and the horizontal edge associated to s_i; if so, s_b is totally discarded and taken out of the stack.

The set of sites that remains in the stack when the process is finished, corresponds to the right portion of the convex hull, $CH_r(S)$. The left convex chain, $CH_l(S)$ can be computed using a similar method but considering the smallest negative polar angle criterion. However, it suffices to change each abscissa of the set of points with its symmetrical, $x = -x$, and applying Algorithm 2 again. Finally, $CH_r(S)$ and $CH_l(S)$ are joined into an only convex chain after discarding the repeated top most and bottom most sites, as described in Fig. 14.

We conclude that the polar diagram is a valid preprocessing for the convex hull computation of a set of points in the plane. This result can be generalized to geometric objects. In Fig. 15(a) and (b) we observe again that all these infinite

```
 INPUT: A set of n points
OUTPUT: The right portion of CH(S)
BEGIN
   sort(S,n)
   push (stack, 0)
   push (stack, 1)
   for i ← 2 TO n-1 do
      a ← pop(stack)
      b ← pop(stack)
      while Intersects (S[b],S[a],S[i]) AND NOT empty(stack) do
         a ← pop(stack)
         b ← pop(stack)
      end_while
      push (stack,a)
      push (stack i)
   end_for
   CHr ⇐ stack
END
```

Algorithm 2. Right_CH (S, N, CHr).

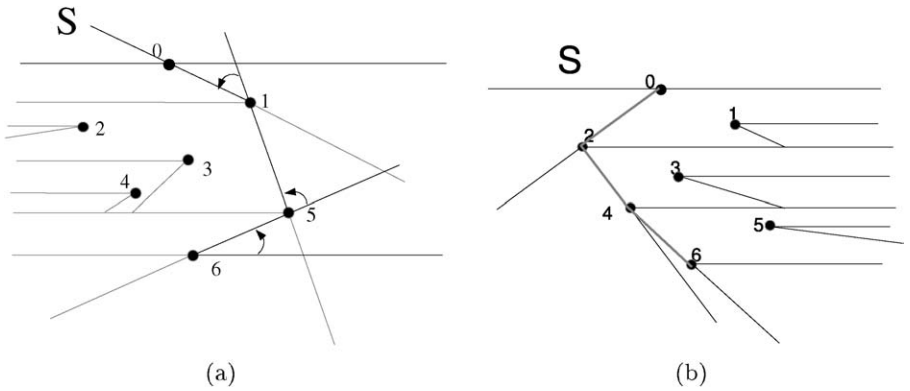


Fig. 14. $CH(S) = CH_r(S) + CH_l(S) - 0 - 6$.

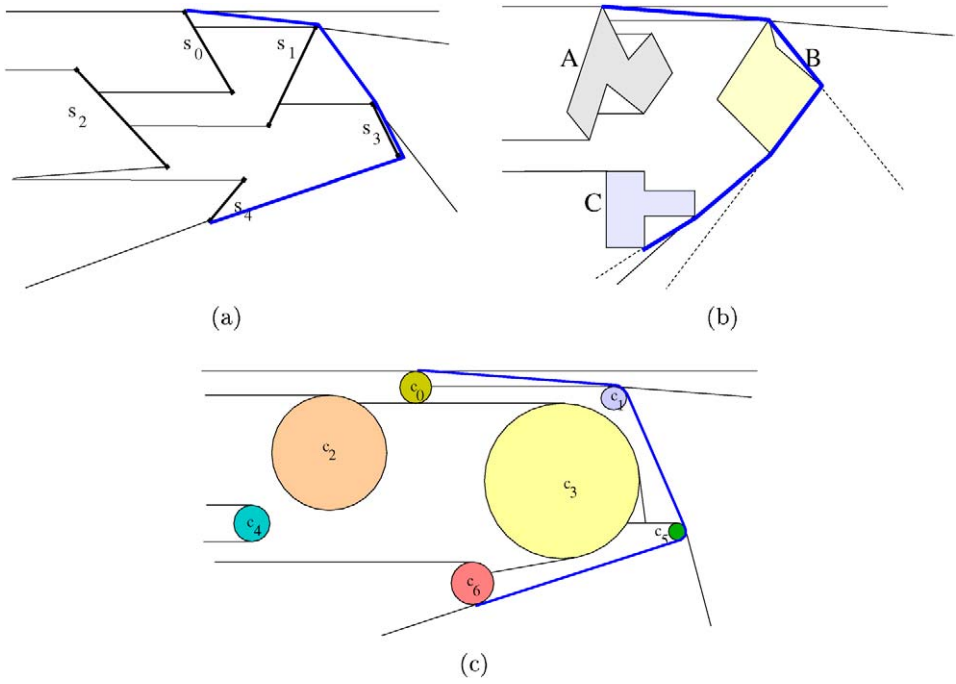


Fig. 15. Convex hull right portion in geometric objects.

oblique edges start from sites in the right portion of the convex hull. In line segments and polygons we should take into account all those infinite edges discarded in the Incremental algorithm in order to avoid additional calculations, due to fact that these removed places can be sites belonging to the convex hull.

In the polar diagram of circles, however, there are not oblique polar edges discarded in the construction process, so that the right portion of the convex hull is straightforward obtained from their polar diagram, as it is depicted in Fig. 15(c).

Theorem 4. *The convex hull of a set of objects, line segments, polygons and circles, can be computed in linear time using the polar diagram as a preprocessing.*

Again, if execution time for the computation of the convex hull of a set of objects is an important restriction, the tessellation process can be avoided to obtain improved running times, as described for a set of points.

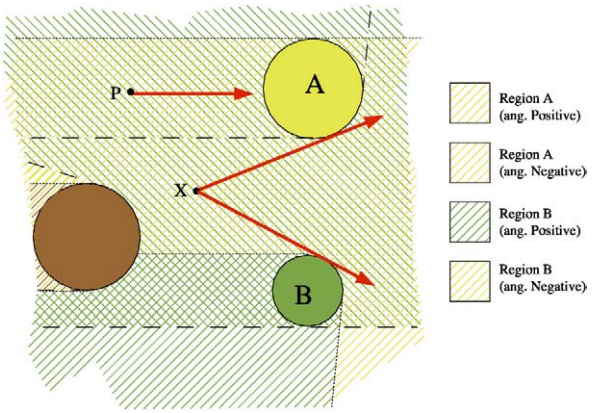


Fig. 16. Point x lies in the polar regions of A and B.

4.2. Visibility problems

Visibility problems are some of the most important topics in Computational Geometry. Their advances have been applied to the Art Gallery [2], Illumination [14], Collision Detection [13] and even the Path Planning problem [15].

One of the simplest visibility problems consists on finding the maximum visibility angle in any orthogonal direction from a point position x, that is, the angular sector we can draw from x including no objects, as shown in Fig. 16. It is possible to see that there exists an open visibility angle looking from point x in the East direction. A simple solution to this problem can be computed in linear time by performing two angular sweeps from the angle zero, one in counterclockwise direction to reach object A, and another clockwise scan to find B. The problem comes when this process has to be repetitive. Then, it should be desirable to avoid searches in linear time in order to obtain improved computation times.

In fact, after the polar diagram definition, it is straightforward to understand how polar diagrams are useful for the described visibility problem resolution. This tessellation is known to be able to avoid angular sweeps by locating a point into a polar region. It is easy to see that a positive angular scanning (starting from angle zero) can be avoided in order to find object A, due to the polar diagram principle: point x lies into A polar region.

The polar diagram is constructed using two criteria: the starting angle and the sweep direction. To find object A we use the positive polar angle criterion in East direction (zero angle and counterclockwise direction), and denoted as East+ polar diagram. Nevertheless, this plane partition is not valid to find object B, in fact, this object is found by a clockwise scanning. The solution comes by changing the construction criterion of the polar diagram: the smallest positive polar angle by the smallest negative polar angle. As a conclusion, any starting and sweeping direction can be used as construction rules of the polar diagram.

In Fig. 16 it has been superimposed the two East polar diagrams according to the positive and negative angle criteria. The visibility problem solution is given by a simple result:

- When a point lies in regions associated to different objects, as point x does, it always means that the visibility angle is different from null, this is, there is an open visibility angle in the chosen polar diagram direction.
- When a point lies in regions belonging to the same object, the visibility angle is null.

We find in Fig. 16 examples of both situations. Even when point p has a null visibility angle, the information provided can be enormously important because we do know the only object obstructing a trajectory in the specified direction. Depending on the visibility application, the polar diagram can help to detect obstacles when a mobile object is moving into a scene. To sum up, we find a close relationship between polar regions and visibility because they are able to know if there exists an obstacle-free path.

Theorem 5. *Given a set of n geometric objects in the plane, line segments, polygons and circles, the polar diagram can be used as a preprocessing to find the visibility angle in any orthogonal direction in* $O(\log n)$ *time.*

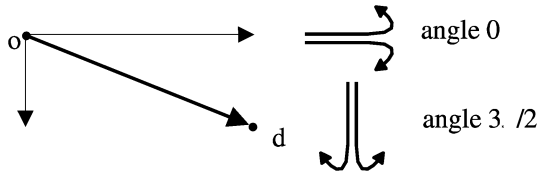


Fig. 17. Orthogonal vector decomposition.

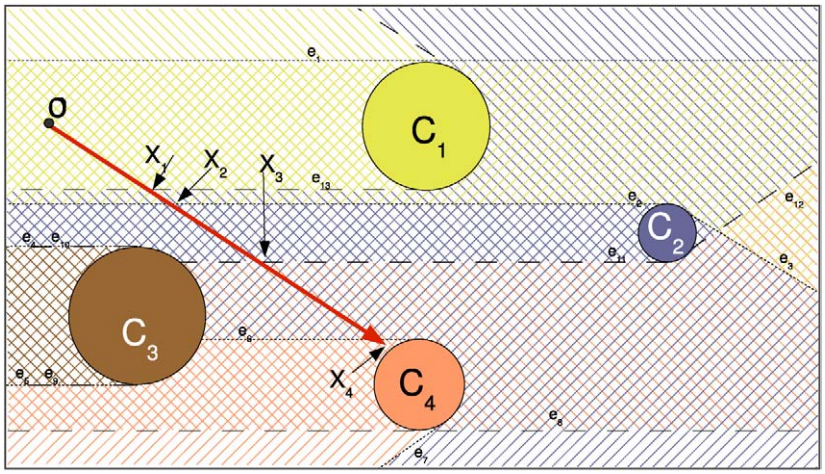


Fig. 18. Generalized visibility problem resolution.

Proof. The followed strategy for visibility resolution in the East direction can be extended to any other orthogonal direction. It is suffices to calculate the pairs of polar diagrams by changing the starting angle to $\pi/2$, π or $3\pi/2$, instead of the 0 angle. The angular search is replaced by locating a point into a polar region in logarithmic time. This result can be extended to polygons, circles and line segments because of the similarity of plane partition. □

The question now is whether the polar diagram is able to provide visibility information about nonorthogonal directions. In real problems, visibility does not depend on a given coordinate system. Fig. 17 shows the vector $\vec{d}$, representing the visibility direction from where an observer, located in position o, is looking at the scene. The decomposition of this vector gives two orthogonal vectors in East and South directions. If we study the angular sweeps that can be avoided using the East and South pairs of polar diagrams, we reach the conclusion that it is possible to obtain visibility information enough in direction $\vec{d}$.

Thus, generalized visibility problems can be solved using together a pair of polar diagrams. Let us observe Fig. 18, where East+ and East− polar diagrams of a set of circles have been superimposed. Suppose that the scene is observed from point o in the given direction $\vec{d}$. Point o lies in $\mathcal{P}_S(o)$ in both polar diagrams, what implies that while the vector crosses the rectangular area formed by o, object C_1 and x_1, the only possible object to be seen is C_1; however it is so distant to be visible (at least directly visible). When the vector crosses from x_1 to x_2 the visibility angle is not null, what means that no objects can be found, as explained before. Again, C_2 is not in the trajectory of $\vec{d}$ when it goes from x_2 to x_3. The last area until point x_4 is reached, belongs again to an open visibility region and no object is visible. But just after crossing x_4, $\vec{d}$ founds C_4, that is checked for proximity. After finding the intersection, the first visible object in the given direction is found.

The computation time for this generalized visibility problem resolution is different to the given in Theorem 5 because we solve a set of local visibility problems. Each of them is solved in $O(\log n)$ time, what implies that the process runs in $O(k \log n)$ time, if k polar regions are finally crossed by vector $\vec{d}$.

5. Convex hull comparative tests

This paper introduces the polar diagram as a new method to compute the convex hull and some other visibility problems. However a novel technique for this classical calculation should not only be interesting, but efficient.

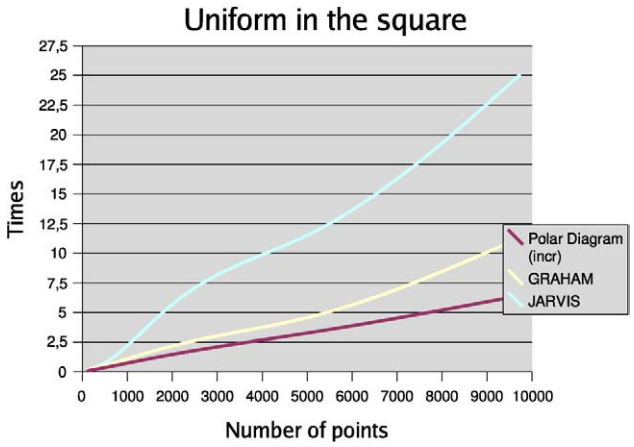
Many classical algorithms to implement the convex hull are optimal methods. Constants associated to their recurrence equations are not always easy to compute. In order to compare some of these algorithms, we have developed under the same conditions, two of the most important and efficient methods. We choose Graham's scan [5] because of its data structure similarity with the one we propose, and the Jarvis's march, whose performance improvement is the aim of this paper. Even when this is a quadratic algorithm, it is known to have a practical time behavior. Some others classical methods have been implemented as well, although we discard Quickhull and the Divide and Conquer paradigm for their poor response in time.

The sets of points have been generated randomly and following different distributions, Normal and Uniform, in the Square and in the Circle as it is shown in Tables 1, 2, 3 and 4. We use the same example for all algorithms tested, our aim is to know the behavior of them under the same conditions. The size of the set of points goes from 100 to 10.000, in order to value the algorithms behavior in infinite. The total number of tests is 72, divided into four tables depending on the distribution. We add in every case a table including the running times and comparative graphs of the polar diagram algorithm with Jarvis's and Graham's methods.

As it is possible to observe, running times associated to the Incremental technique for the polar diagram construction are the fastest, with independence of the size of the set of points or distribution. We have to emphasize that both, right and left portions of the convex hull have been processed, running Algorithm 2 twice according to the explanations in Section 4.1. Similarities with Graham's scan are evident, both methods graphs have the same shape. Although, Incremental method for the polar diagram calculation is always faster. In fact, their differences seem to be greater when the size of the points set grows.

Conclusions about our principal aim are significant. Jarvis's march is even more than three times slower than our method. We prove that the polar diagram can improve this classical algorithm significantly. Processing times are more

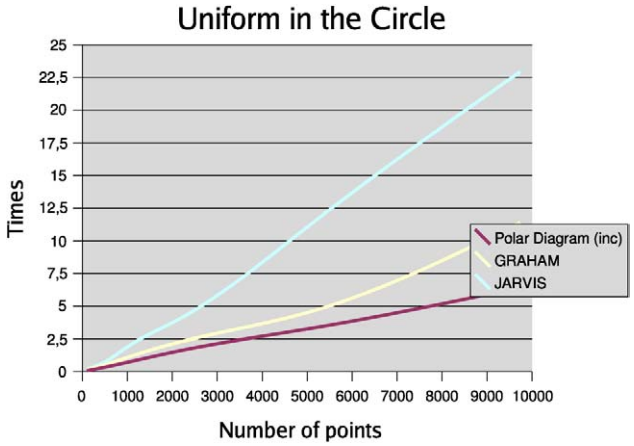
Table 1
Uniform in the square



Unif. square	100	500	1000	2000	5000	10000
DP (Incremental)	0.070	0.355	0.740	1.465	3.250	6.600
Graham	0.135	0.540	1.065	2.175	4.550	11.750
Jarvis	0.165	0.755	2.220	5.665	11.50	25.95

Unif. square	100	500	1000	2000	5000	10000
Graham/Incremental	1.92	1.52	1.43	1.48	1.4	1.78
Jarvis/Incremental	2.35	2.12	2.08	3.86	3.53	3.93

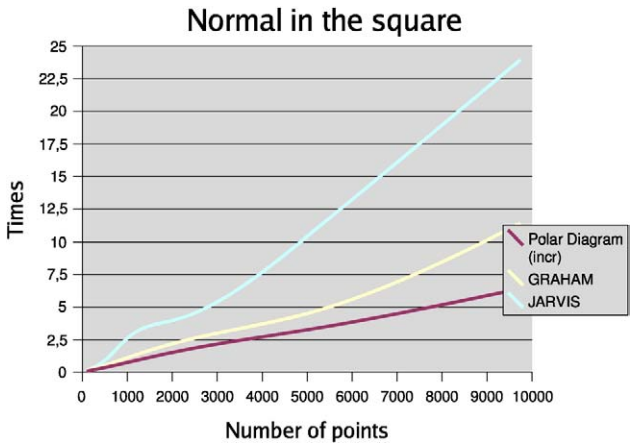
Table 2
Uniform in the circle



Unif. circle	100	500	1000	2000	5000	10000
DP (Incremental)	0.070	0.340	0.715	1.480	3.250	6.600
Graham	0.135	0.530	1.090	2.120	4.500	11.85
Jarvis	0.195	0.740	1.895	3.725	11.05	23.50

Unif. circle	100	500	1000	2000	5000	10000
Graham/Incremental	1.92	1.55	1.52	1.43	1.38	1.79
Jarvis/Incremental	2.78	2.17	2.65	2.51	3.4	3.56

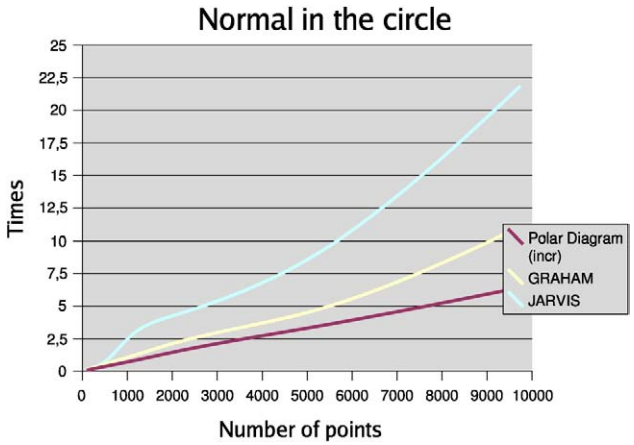
Table 3
Normal in the square



Normal square	100	500	1000	2000	5000	10000
DP (Incremental)	0.070	0.350	0.735	1.510	3.250	6.600
Graham	0.135	0.530	1.105	2.180	4.500	11.85
Jarvis	0.140	0.905	2.600	3.960	10.40	24.70

Normal square	100	500	1000	2000	5000	10000
Graham/Incremental	1.92	1.51	1.50	1.47	1.38	1.79
Jarvis/Incremental	2	2.58	2.35	2.67	3.20	3.74

Table 4
Normal in the circle



Normal circle	100	500	1000	2000	5000	10000
DP (Incremental)	0.070	0.355	0.700	1.440	3.300	6.600
Graham	0.125	0.535	1.050	2.110	4.500	11.50
Jarvis	0.115	0.725	2.460	4.240	8.550	22.70

Normal circle	100	500	1000	2000	5000	10000
Graham/Incremental	1.78	1.50	1.5	1.46	1.36	1.76
Jarvis/Incremental	1.64	2.04	3.51	2.94	2.59	3.43

or less similar with independence of the distribution for Graham's scan and polar diagram methods. Jarvis's march has a stronger dependence on the number of points in the convex hull, obtaining better results with a normal distribution in the circle.

To summarize, the polar diagram tessellation and the technique developed for its construction, can be considered an important new approach in Computational Geometry because of their robust behavior, optimal running time and the lots of applications in angle problems.

6. Conclusions and open problems

We can conclude that the polar diagram performs a new partition of the plane that certainly can improve some angle problems performance. We provide robust and efficient methods for this tessellation construction, obtaining important results for a set of points and for a set of geometric objects as well.

We consider the polar diagram applications as an open problem with several applications to some other visibility problems. Actually, Collision Detection and the Path Planning problem can be dealt with the polar diagram in order to know in advance those obstacles to whom a mobile object can intersect [11,16] and the Visibility Map. Our immediate aims are to lay the foundations for the $3D$ and $2-1/2D$ polar diagram in order to obtain solutions to three-dimensional problems.

References

[1] S.G. Akl, K.A. Lyons, Parallel Computational Geometry, Prentice-Hall, 1993.
[2] V. Chvátal, A combinatorial theorem in plane geometry, J. Combin. Theory, Ser. B. (1975).
[3] J.C. Dana, C.I. Grima, A. Márquez, Convex hull in non-planar surfaces, in: Proc. 13th European Workshop on Computational Geometry, University of Wuerzburg, Germany, 1997.
[4] J.C. Dana, C.I. Grima, A. Márquez, Envolvente convexa en superficies no planas, in: Proc. VII Encuentros en Geometría Computacional, Polytechnic University of Madrid, Spain, 1997 (in Spanish).
[5] R.L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, Inform. Process. Lett. 1 (1972) 132–133.

[6] C.I. Grima, A. Márquez, Computational Geometry on Surfaces, Kluwer Academic Publishers, 2001.
[7] C.I. Grima, A. Márquez, L. Ortega, A locus approach to angle problems in computational geometry, in: 14th European Workshop in Computational Geometry, Barcelona, Spain, 1998.
[8] C.I. Grima, A. Márquez, L. Ortega, Diagrama polar de objetos geométricos, in: IX Congreso Español de Informática Gráfica, University of Jaén, Spain, 1999 (in Spanish).
[9] C.I. Grima, A. Márquez, L. Ortega, Polar diagrams of geometric objects, in: 15th European Workshop in Computational Geometry, Antibes, France, 1999.
[10] C.I. Grima, A. Márquez, L. Ortega, Un preprocesamiento para problemas de ángulos en Geometría Computacional, in: VIII Congreso Español de Informática Gráfica, Ourense, Spain, 1998 (in Spanish).
[11] C. Grima, A. Márquez, L. Ortega, Motion planning and visibility problems using polar diagrams, in: Annual Conference of the European Association for Computer Graphics, EG'2003, University of Granada, Spain, 2003.
[12] R.A. Jarvis, On the identification of the convex hull of a finite set of points in the plane, Inform. Process. Lett. 2 (1973) 18–21.
[13] P. Jiménez, F. Thomas, C. Torras, 3D collision detection: a survey, Computer Graphics 25 (2001).
[14] V. Klee, Is every polygonal region illuminated from some point? Amer. Math. Monthly (1969).
[15] J.C. Latombe, Robot Motion Planning, Academic Publishers, Boston, 1991.
[16] L. Ortega, C. Grima, A. Márquez, Colisiones entre objetos usando Diagramas Polares, in: XIII Congreso Español de Informática Gráfica CEIG'03, University of La Coruña, Spain, 2003 (in Spanish).
[17] A. Okabe, B. Boots, K. Sugihara, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, John Wiley and Sons, 1992.