Polar Diagram of Moving Objects

Mojtaba Nouri Bygi[*] Mohammad Ghodsi[†]

Abstract

Many important problems in Computational Geometry needs to perform some kind of angle processing. The Polar Diagram [4] is a locus approach for problems processing angles. Using this structure as preprocessing, one can eliminate exhaustive searches to find objects with smallest angle.

Handling data in change is a significant concept in Computer Science. One of the design and analysis tools used in the modeling of moving geometric objects is the kinetic data structure (or KDS) framework Kinetic Data Structure is a framework for maintaining a certain attribute of a set of objects while moving in a continuous manner.

In this paper, we use the notion of kinetic data structure to model the dynamic case of the Polar Diagram, i.e we maintain the the Polar Diagram of a set of continuously moving objects in the scene. We show that our proposed structure meets the main criteria of a good KDS.

1 Introduction

Although most of the Geometric problems have optimal solutions, most of them are only optimum in the worst case. If the size of result is small or we have to answer many instances, these solutions may not be suitable for us. For these reasons, algorithms that preprocess the scene and then answer to each query with a better performance are widely used in this field.

C. I. Grima et al. [4, 5] introduced the concept of the Polar Diagram. The Polar Diagram of the scene consisting of n objects is a partition of plane to *polar regions*. Each object creates a polar region representing the locus of points with common angular characteristics in a starting direction. If point p lies in the polar region of object o, we know that o is the first object found after performing an angular scanning from the horizontal line crossing p in counterclockwise direction. The computation of the Polar Diagram can be done using the Divide and Conquer or the Incremental methods, both working in $\Theta(n \log n)$, which is optimum. By using this tessellation as preprocessing, we can avoid other angular sweeps by locating a point into a polar region in logarithmic time [4].

Kinetic Data Structure is a framework for maintaining a certain attribute of a set of objects while moving in a continuous manner. For example, KDS has been used for maintaining the convex hull of moving objects, or the closest distance among moving objects. A KDS is mainly consists of two parts: a description of the attribute with some certificates such that as long as these certificates do not change, the attribute does not change. It is assumed that we can compute the failure time of each of these certificates. In such events that a certificate fails, the KDS must be updated. Until the next event, the KDS remains valid. See the survey by Guibas [3] for more background on KDSs and their analysis.

In this paper, we first propose an improved algorithm for computing the Polar Diagram of a set of line-segments or polygons. Then we use the notion of kinetic data structure to model the dynamic case of the Polar Diagram, i.e we maintain the Polar Diagram of a set of continuously moving objects in the scene. We Show that our proposed structure meets the main criteria of a good KDS.

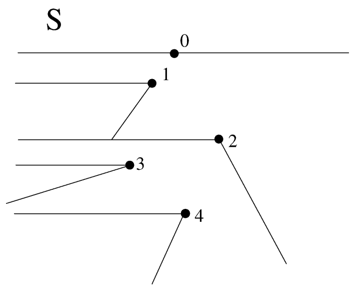

Figure 1: The Polar Diagram of a set of points in plane.

The rest of this paper is organized as follows: In section 2 we define our kinetic configuration for the Polar Diagram, and in section 2.2 we see what happens when the objects move in the plane.

[*]Department of Computer Engineering, Sharif University of Technology, P.O. Box 11365-9517, Tehran, Iran, nouribaygi@ce.sharif.edu

[†]Department of Computer Engineering, Sharif University of Technology, and IPM School of Computer Science (No. CS1382-2-02), P.O. Box 19395-5746, Tehran, Iran, ghodsi@sharif.edu

2 Kinetic Configuration

In this section we present a model for kinetic behavior of the Polar Diagram for different situations. Given a set of points moving continuously, we are interested in knowing at all times the Polar Diagram of the scene.

2.1 Proof Scheme

For simplicity of discussions, we assume that our objects are points in 2D. We state that each edge of the Polar Diagram is called a *polar edge*. We also define a *pivot* of an object to be the second object that lies on the polar edge passing through it, e.g., if Figure 1 the pivot of s_4 is s_2 and the pivot of s_2 is s_0.

We claim that if we have the sorted list of objects according to their y-coordinates, and the pivot of each object, we will have a unique Polar Diagram.

Suppose there are n points in the scene. For our proof scheme, we maintain two kinds of information about the scene: we maintain the vertically sorted list of objects, and for each object its current pivot. As we will show shortly, these data is sufficient for the uniqueness of our polar data, i.e. only if one of these conditions change, the polar structure of the scene will change.

So we will have two kinds of certificates: $n-1$ certificates will indicate the sorted list of objects. For instance, if the sorted list of objects is $s_{i_0}, s_{i_1}, \ldots, s_{i_{n-1}}$, we need the certificates 1.

$$\begin{aligned} s_{i_0} &< s_{i_1} \\ s_{i_1} &< s_{i_2} \\ &\ldots \\ s_{i_{n-2}} &< s_{i_{n-1}} \end{aligned} \tag{1}$$

For stating the pivot of each object, we need n more certificates, each indicating a object and its pivot in the Polar Diagram. In total, our proof scheme consists of $2n-1$ certificates.

2.2 Events and Event Handling

Once we have a proof system, we can animate it over time as follows. As stated before, each condition in the proof is called a certificate. A certificate fails if the corresponding function flips its sign. It is also called an event happens if a certificate fails. All the events are placed in a priority queue, sorted by the time they occur. When an event happens, we examine the proof and update it. An event may or may not change the structure. Those events that cause a change to the structure are called *exterior events* and those not *interior events*. When the motion of an object changes, we need to reevaluate the failure time of the certificates that involve that object (this is also called *rescheduling.*).

As there are two kinds of certificates in our proof scheme, it is obvious that there must be two kinds of event:

- **pivot event**, when three objects, which one of them is pivot of another one, become collinear.
- **horizontal event**, when two objects have a same y-coordinate (have a same horizontal level)

In the former case, we must update the certificates relating to sorted sequence of two neighbor points, which is at most three certificates (two, if one of the points is a boundary point, i.e. top most or button most points). In the latter case, one certificate becomes invalid and another certificate (indicating the new pivot of the object) is needed. As we will show, other certificates will remain still.

Lemma 1 *When an event is raised, the objects above the object(s) which raised the event do not change their polar structures.*

Proof: From the incremental method used for the construction of the Polar Diagram of a set of points [4] we know that there is no need to know about the state of objects below a object to determine its pivot object, so when an object change its state, it will not affect the above objects.

We can also say that an angular sweep that starts from the horizontal direction would never intersect any objects below this initial horizontal line (by definition, the top most object has no pivot). □

Pivot event:

First, we consider the simplest case, i.e. when the lowest object is moving. Figures 2 and 3 show these cases, where s_2 is moving. In Figure 2, s_0 is the pivot of s_2. While s_2 is moving left, the line segment s_0s_2 is coincide with the object s_1 (note that there may be other objects between s_0 and s_2, but we are only interested in s_1). At the moment that three objects s_0, s_1, and s_2 become collinear, the s_1 will occlude s_0 from s_2 and it no longer can be its pivot. From now on, s_1 becomes the new pivot of s_2. Similarly, in Figure 3, s_1 is the pivot of moving object s_2. When three objects s_0, s_1, and s_2 become collinear (again, there may be other objects between each pair of these objects, but we are not interested in them), s_2 needs to change its pivot which becomes s_0.

As we assumed that no other object other than s_2 is moving, form lemma 1 we know that there will be no change in other objects, so at this event, only one certificate becomes invalid and it must be replaced by another certificate indicating the new pivot of the moving object. It is clear that upon occurring this event, the processing of the event and changing of proof scheme can be done in $O(1)$ and $O(\log n)$, respectively (we need

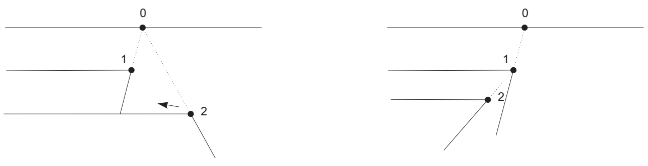

Figure 2: A pivot event. As s_2 moves left, s_0, s_1 and s_2 become collinear.

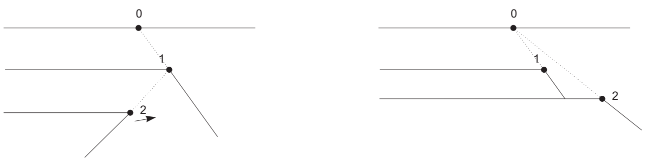

Figure 3: A pivot event. As s_2 moves right, s_0, s_1 and s_2 become collinear.

to find the corresponding certificate in the certificates list).

Now we see what happens to the second lowest object (see Figures 4 and 5, where s_2 is moving right). In Figure 4, s_1 is the pivot of s_2, and also the pivot of the lower object s_3. While moving, there will be a time that s_2 occlude the lower object s_3 from its pivot. In Figure 4 it is when the objects s_1, s_2 and s_3 become collinear. At this time, although there is no change in polar structure of moving object s_2, there is a change in the lower object s_3, and we must update the proof scheme accordingly. If s_2 continues its motion, there will be a pivot event (see Figure 5) that its polar structure is changing.

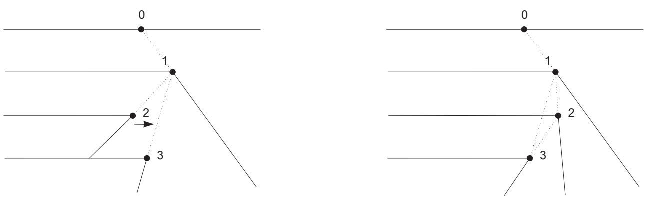

Figure 4: While moving, s_2 can change the pivot of each of its below objects by occluding their initial pivots.

Lemma 2 *The changes in the structure of an object caused by moving an above object, would not cause any other changes in other objects.*

Proof: The Structure of each object is determined by the first object that encountered by an angular sweep. As we assumed that no other objects is moved, this encountered object would not change. □

From above discussions, we can deduce that if an object is moving in the scene and there are k other objects below it, there can be up to k pivot events changing the structure of below objects, and one pivot event changing its own structure. Each of these events can be processed

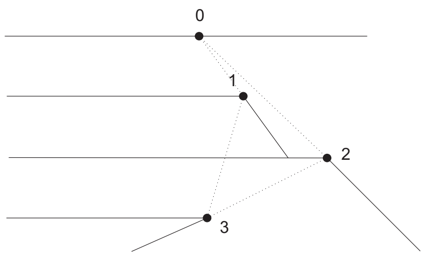

Figure 5: For each moving object, there is one pivot event when its own pivot will change.

in $O(1)$ time and the change in proof scheme can be done in $O(\log n)$.

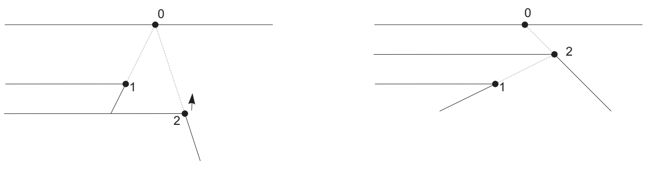

Figure 6: When two objects s_1 and s_2 lay on a same horizontal level, a horizontal event is occurred and the polar structure will change.

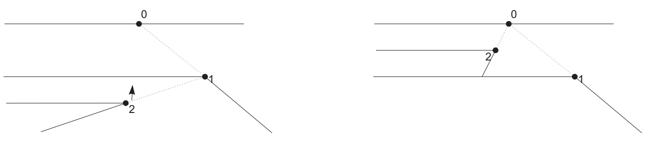

Figure 7: In a horizontal event, only one of the objects will change its pivot.

Horizontal event:

In these events, one of the situations of Figures 6 and 7 will happen. As we can see, only one of the objects will change its pivot (set it to the third object). This change of configuration is equal to changing three or four certificates in proof scheme: one for a change in one of the object's pivot, and three or two for change in vertical order of objects.

Now we show that no more changes is needed. Assume that in a small interval before and after the horizontal event, no other pivot events would occur. From lemma 1 we know that there would be no change in the above objects. What about the below objects? We can see that for a change in the pivot of an object, there must be an occlusion between the objects and its previous pivot, and it means that three objects must lay on a same line, i.e. we need a pivot event (see Figure 8).

Theorem 3 *Each of the events in the kinetic Polar Diagram of a set of points takes $O(\log n)$ time to process and causes has $O(1)$ changes in proof scheme.*

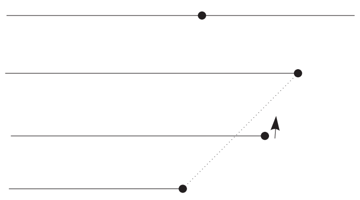
Figure 8: Only upon occurring a pivot event the structure of other objects will change.

Proof: For horizontal events, we need to update at most three certificates, we just need to find these certificates in the proof scheme and replace them with the new ones, which takes $O(\log n)$ time. We also need to update one pivot certificate with the same cost. The same thing is holds for pivot events, which we need to find and update $O(1)$ pivot certificates. □

Theorem 4 *The initial event list can be built in $O(n \log n)$ time, using a suitable event queue.*

Proof: As there are $O(n)$ certificates in our proof scheme, and for each moving object. we can find the first certificate that it will violates by a simple $O(\log n)$ search, the proof is straightforward. □

3 KDS Evaluation

In this section we evaluate our kinetic model according to the criteria of a *good* KDS. Similar to other algorithms, a good KDS should take small space, small initialization cost, and efficient update time. In KDS, an update may happen in two cases. One is when a certificate fails and an event happens. The other is when the motion of an object changes. In first case, we need to update the certificate set, and in the second case we must recompute the failure times for all the certificates that involve that object. These requirements induce the following quality measurements for KDSs [2].
Compactness the size of the proof.
Responsiveness the time to process an event.
Locality the number of certificates that a single object involves in.

Another crucial efficiency factor of a KDS is the number of events processed. This factor determines how many times we need to stop to check our proof and structure. This factor is expressed by efficiency:
Efficiency the number of events processed.

Now, we consider each of the above criteria in our kinetic model.
Compactness The structure clearly takes linear space. As we stated in Section 2.1, for a set of n point objects, the proof scheme consists of $n-1$ certificates for sorted vertical order of objects and n certificates for maintaining the pivots of each object, so in total, our proof scheme have $2n-1$ certificates.
Responsiveness $O(\log n)$ for processing an event as there are $O(1)$ certificates need to reschedule. Each reschedule takes $O(\log n)$ time.
Locality Each object is involved in at most three certificates.
Efficiency All the events are exterior – the ordering changes once a horizontal event happens, or the pivot of an object changes once a pivot event happens. The number of events is bounded by $O(n^2)$ as any two points can exchange their ordering only constant number of times for constant degree algebraic motions, and any point is a potential candidate for being the pivot of another point.

4 Conclusion and Future Work

In this paper we studied the concept of the Polar Diagram, which is a new locus approach for problems processing angles, and KDS, which is a structure that maintains a certain attribute of a set of continuously moving objects among moving objects. We used KDS to model the behavior of a the Polar Diagram when our scene is dynamic, i.e. we maintain the Polar Diagram of a set of continuously moving objects. We showed that our proposed structure meets the main criteria of a good KDS.

Following our defined model for the kinetic Polar Diagram, we can use it in direct applications of the Polar Diagram to maintain the computed attributes. For example, we can use the kinetic Polar Diagram for maintaining the convex hull of a set of moving objects with a very low cost.

References

[1] J. Basch, L. J. Guibas, and J. Hershberger. *Data structures for mobile data.* In Proc. 8th ACM-SIAM Sympos. Discrete Algorithms, pages 747–756, 1997.

[2] J. Basch. *Kinetic data structures.* PhD thesis, Stanford University, Palo Alto, California, 1999.

[3] L. Guibas. *Kinetic data structure: a state of art report.* In Proc. 3rd Workshop Algorithmic Found. Robot., pages 191–209, 1998.

[4] C.I. Grima, A. Márquez, and L. Ortega. *A new 2D tessellation for angle problems: The polar diagram.* In Computational Geometry, Theory and Application, 34(2): 58-74, 2006.

[5] C.I. Grima, A. Márquez, and L. Ortega. *A locus approach to angle problems in computational geometry.* In Proc. of 14th European Workshop in Computational Geometry, Barcelona, 1998.