Title:
Maintaining the Spatial Proximities of Objects under Motion

Authors:
Zesheng Jia[1], will.jia.sheng@gmail.com, Saint Mary's University, Halifax, Canada
Anjali Jose[1], anju.jp17@gmail.com, National Institute of Technology, Calicut, india
Subhsree Methirumangalath, subhasree.rajiv@gmail.com, National Institute of Technology, Calicut, india
Jiju Peethambaran, jiju.poovvancheri@smu.ca, Saint Mary's University, Halifax, Canada
Ramanathan Muthuganapathy, emry01@gmail.com,Indian Institute of Technology, Madras, India

Keywords:
Computational geometry, Gabriel graph, Kinetic Delaunay triangulation, Collision detection

DOI: 10.14733/cadconfP.2020.248-252

Introduction:

Spatial proximities of moving objects is a useful measure to predict and avoid object collisions in applications such as traffic control system, crowd simulation, and fluid flow rendering, among others. Depending on the application requirement, proximity graphs such as Delaunay graphs, Gabriel graphs, or Relative Neighborhood Graphs can be used to compute the geometric proximities of a set of objects. Proximity graphs have been extensively studied in computational geometry and have several applications in GIS, wireless networks or computer graphics[3]. However, the past research on proximity graphs mainly focused on stationary points with limited attention given to these structures for moving objects. An exception being the Delaunay graph, which has been decently studied in the kinetic setting. A few work addressing the theoretical bounds, e.g., [1] and experimental treatment [4] of kinetic Delaunay graph can be found in the literature. We consider the problem of maintaining the inter-object spatial proximities of a set of moving objects via a kinetic Gabriel graph.

In general, there are two approaches for handling kinetic data (i.e, moving data) - time discretization approach and continuous movement approach. The traditional time-discretization paradigm, in which a problem involving moving objects is discretized into static instances, is inadequate in many applications as it ignores temporal coherence and non-uniform nature of motion. By temporal coherence, we expect that the structure does not change too much between two consecutive time steps and hence recomputing the structure from scratch at each time step is wasteful. Kinetic data structure(KDS) introduced by Basch et al. [2], effectively utilizes the temporal coherence of the moving objects. Under KDS setting, the motion trajectories of the moving points is known apriori which are given as functions of time. The combinatorial description of the geometric structures (e.g., convex hull or Delaunay graph) is referred to as the *configuration function* of the system, which changes at discrete times when certain events happen among the moving objects. KDS maintains not only the combinatorial structure itself but also some additional information called *certificates* that helps to find out when the structure will undergo a real combinatorial change.

[1]Co-first author

Kinetic Gabriel Graph:

Let each point $p_i \in P$ moves along an algebraic trajectory with a constant velocity. We assume that the coordinates of each point are linear functions of time, i.e., $p_i(t) = (x_i(t), y_i(t))$, $x_i(t) = x_i(t_0) + \Delta x_i t$, $y_i(t) = y_i(t_0) + \Delta y_i t$. The initial position of a point at time t_0 is represented by $p_i(0) = (x_i(t_0), y_i(t_0))$ and the velocity vector $v_i = (\Delta x_i, \Delta y_i)$. We also assume that the trajectories do not meet each other, i.e., $p_i(t) \neq p_j(t)$ for $i \neq j$ and all t for which both trajectories exist.

Flip events A combinatorial change in the kinetic Delaunay triangulation is *flip event* [4] occurs due to either co-circularity, i.e., four points of P lying on a circle that contains no other points of P, or collinearity, i.e., three points of P lying on a line, and one of the half planes bounded by this line contains no points of P.

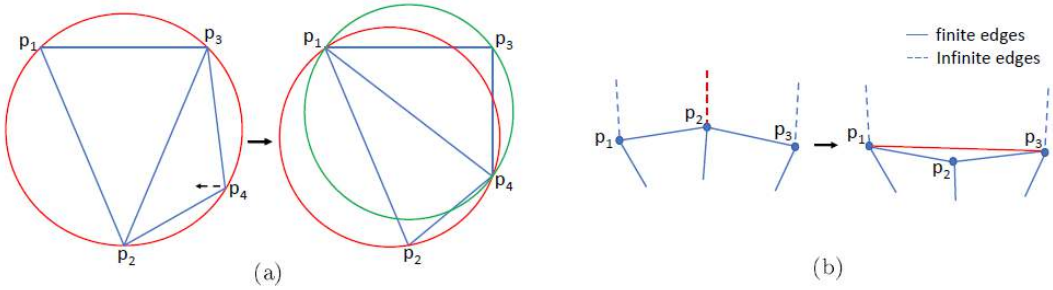


Fig. 1: Edge flipping due to different configurations: (a)co-circularity-the point p_4 moves into the circumcircle of $\triangle\, p_1p_2p_3$ thereby, violating the definition and induces an edge flipping. (b) due to collinearity: three collinear boundary points along with infinite vertex results in co-circular degeneracy results in an edge flipping. Dashed edges represent the infinite edges [4].

These are degenerate configurations, whose effect on the combinatorial structure (Delaunay triangulation) can be investigated by examining the non-degenerate local configurations at the preceding and following moments of the degeneracies. Consider a point moving into the interior of a Delaunay circle defined by three other points. At a time instant, the four points becomes co-circular. Right after the co-circular configuration, the empty circumcircle property is violated. The topological correctness of the local configuration involving the two triangles is then regained via edge flipping as shown in Figure 1. Further, the KDT algorithm updates the certificates of the neighboring triangle pairs in the boundary of the quadrilateral formed by the four points that triggered the event, i.e. five new certificate functions must be computed. Figure 1(a) shows an example of flip event.

Since it is convenient to deal with only triangular faces in many applications such as incremental Delaunay construction, the convex hull edges of the triangulation is connected to a fictitious vertex called *infinite vertex*, via *infinite edges*. Three boundary points in the collinear configuration together with the *infinite vertex* give rise to the co-circular degeneracy, which is resolved by an edge flipping as shown in Figure 1 (b).

Gabriel certificates Kinetic Gabriel graph is maintained via tagged Delaunay edges. Each Delaunay edge is equipped with a *Gabriel flag* that indicates whether or not the edge belongs to the *GG*. Edges are tagged based on the locations of the circum-centers of the incident Delaunay triangles. We exploit the fact that the circum-center of a right triangle lies on its longest edge and the circum-centers of acute and obtuse triangles lie in the interior and exterior of the triangles, respectively as illustrated in Figure 2. This immediately gives us a relation between the circum-center location and the angle at the opposite

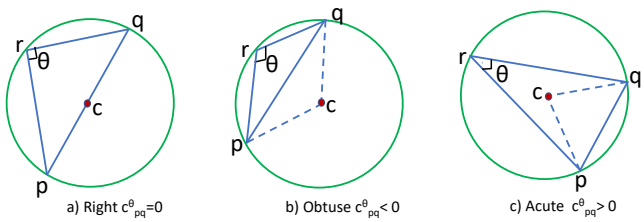


Fig. 2: Signs of the angle certificate function with respect to the edge (pq) for different types of triangles.

vertex with respect to any edge of a triangle. Let pq and c be an edge and the circum-center of a Delaunay triangle $\triangle pqr$. Let θ be the angle opposite to the edge pq. For any edge pq of a triangle, if the circum-center c and the vertex opposite to pq lies on the same half plane bounded by the line pq, then θ is an acute angle. If c and the opposite vertex r lie on either side of pq, then θ is an obstuse angle. We capture this configuration using a predicate function indirectly defined over the angle (θ) opposite to the edge under consideration. The spatial location of the circum-center c of a triangle, $\triangle pqr$ with respect to the edge pq is determined using the *angle certificate function* presented in Equation 2.1. In Equation 2.1, the coordinates of the circum-center (c_x, c_y) can be obtained from the coordinates of the triangle vertices. If both, r and c lie on either side of the edge pq, then the orientation tests will return different signs. Consequently, $\mathcal{C}^{\theta}_{pq}$ will be evaluated to negative quantity implying an obtuse θ (see Figure 2 (b)). A positive $\mathcal{C}^{\theta}_{pq}$ indicates an acute θ (Figure 2 (c))and a right angle at r evaluates $\mathcal{C}^{\theta}_{pq}$ to zero (Figure 2 (a)).

$$\mathcal{C}^{\theta}_{pq} = sign \begin{vmatrix} p_x(t) & p_y(t) & 1 \\ q_x(t) & q_y(t) & 1 \\ r_x(t) & r_y(t) & 1 \end{vmatrix} \times sign \begin{vmatrix} p_x(t) & p_y(t) & 1 \\ q_x(t) & q_y(t) & 1 \\ c_x(t) & c_y(t) & 1 \end{vmatrix} \tag{2.1}$$

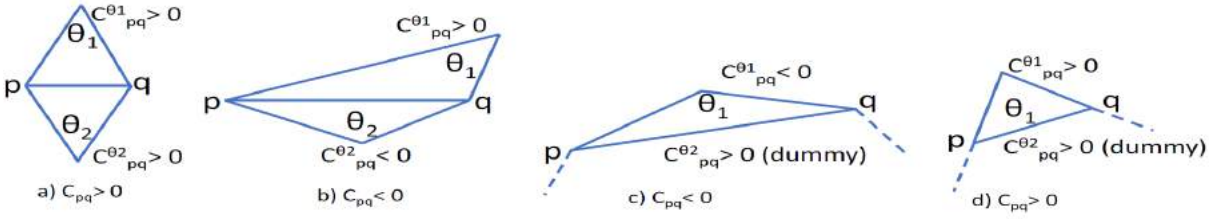


Fig. 3: Gabriel certificates for interior and boundary Delaunay edges. (a) Gabriel edge, (b) Non-Gabriel edge and (c)-(d)Gabriel certificates for boundary edges.

$C^{\theta_1}_{pq}$	$C^{\theta_2}_{pq}$	$C_{pq} = sign(C^{\theta_1}_{pq}) \times sign(C^{\theta_2}_{pq})$
> 0	> 0	Gabriel
< 0	> 0	Non-Gabriel
> 0	< 0	Non-Gabriel
< 0	< 0	Not valid

Table 1: Relation between the angle and Gabriel certificates for an interior Delaunay edge pq

Each *interior* edge in the triangulation has two angle certificates corresponding to its incident triangles. Depending on the signs of the two angle certificates, four cases arise as shown in Table 1 and Figure 3.

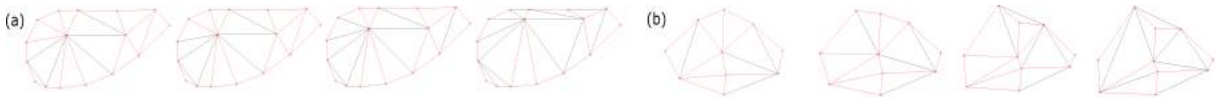


Fig. 4: (a) KGG of 16 points, where only two points in the center moving. (b)KGG where all points are moving : $t_3 > t_2 > t_1 > t_0$

A Delaunay edge is never shared by two obtuse Delaunay triangles as it violates the empty circum-circle property and hence we discard that case (fourth row in Table 1). We consider the remaining three cases. To tag the edges as Gabriel, for each internal edge, we use the signs of its two angle certificates. The product of the signs of the two angle certificates of an edge is referred to as Gabriel certificate(C_{pq}). Depending on the sign of this certificate, we tag the edges as Gabriel or non-Gabriel as per the rules given in Table 1. Implications of these rules will be discussed in the presentation or extended version.

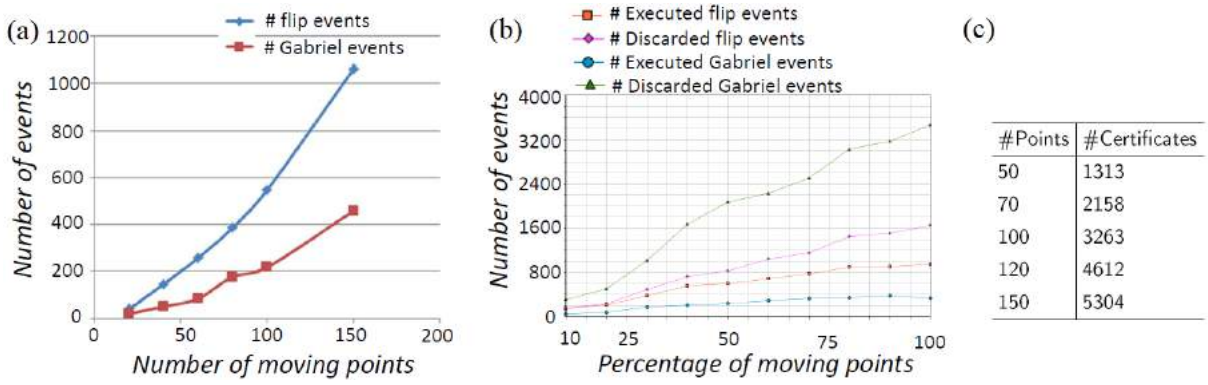
The Algorithm Initially, (at time t=t_0) all the edges of the Delaunay triangulation are tagged (Gabriel or non-Gabriel) based on the Gabriel certificates. The algorithm then searches for the next event. A flip event results in the update of five angle and Gabriel certificates corresponding to the diagonal and the quadrilateral edges of the triangles involved in the flip. The functions are again pushed into the priority queue depending upon the time of their occurrences. Similarly, when there is a change in the Gabriel certificate, the corresponding Gabriel tag of the edge is updated.

Solving the certificates. Maintaining the Gabriel graph of moving points involves solving equations corresponding to the co-circular, angle and Gabriel certificates. For solving polynomial corresponding to co-circular event (degree up to 4), the equation $det(I(t)) = 0$ needs to be solved where I represents the incircle test matrix. The method adopted to solve this equation is Sturm Sequences which enjoys the advantage that it gives the count of real roots in any interval [a,b] and their multiplicities. The KGG part of the algorithm need to solve only degree 2 polynomials and a few checks on certificate signs (further discussion on this will be included in the presentation/extended version of the paper.)

Results and Discussion:

Figure 4(a) showcases an example of kinetic Gabriel graph for a set of 16 points. The input set consists of static as well as kinetic points. Initially, at time t=0, all points have velocity zero and a static Gabriel graph is shown in Figure 4(a), first column. When t>0, the points with non-zero velocity move and the corresponding Gabriel graphs are shown in the columns 2-4 of Figure 4(a). As the points move, the graph maintains the Delaunay property and the edges satisfying Gabriel condition are highlighted (in red color). Figure 4(b) shows an example where all the input points are moving.

Topological Events Figure 5(a) represents the relation between number of points and number of events. Here, all the points are moving with linear velocity. The Gabriel events except those triggered by cocircular events are counted. As the number of moving points increase, the number of events increase drastically. Also, we can see that a considerable number of Gabriel events occur which are not triggered by cocircular events. Figure 5(b) represents the number of executed and discarded Gabriel and cocircular events . This result was obtained from randomly chosen 100 points among which a certain percentage are assigned random velocity. The experiment was conducted for an interval of 10 seconds. Events are discarded when the triangles involved in the event no longer exist. As we can see, the number of discarded events is always greater than number of executed events. Figure 5(c) represents the number of certificate functions solved and processed during a time period of 10 seconds where all the input points have a



#Points	#Certificates
50	1313
70	2158
100	3263
120	4612
150	5304

Fig. 5: Summary of various events including executed and discarded appeared in the kinetic Gabriel graphs of different point sets with varying sizes.

Fig. 6: An illustration of how KGG can be used in video based monitoring system. Kinetic Gabriel graph of cars moving at an intersection overlaid on a set of corresponding images. Red edges indicate cars in potential collision zone. When used with an appropriate threshold, the proposed data structure is useful in similar collision avoidance applications in kinetic scenario.

non-zero velocity. As the point set increases, number of Delaunay triangles increases which results in a monotonic increase in the number of certificates, both co-circular and Gabriel.

In video based intersection traffic monitoring and collision avoidance systems, proximity information of the vehicles approaching the intersections can be captured via Gabriel graph. The vehicles in each frame can be detected using a robust object detection algorithm and the velocity vectors of the objects in a frame can be determined by inter-frame object mappings (using the nearest neighbors and the directional vector of the previous frame). An object's directional vector gets updated only when it considerably deviates from the previous one, e.g., if the vehicle turns left or right. Any two vehicles connected by a red edge (Gabriel) with an appropriate threshold are in collision zone and hence, call for evasive actions. Fig. 6 showcases a few frames of a traffic video with the corresponding Gabriel graphs overlapped.

References:

[1] Agarwal, P.K., Kaplan, H., Rubin, N., Sharir, M.: Kinetic voronoi diagrams and delaunay triangulations under polygonal distance functions. Discrete & Computational Geometry **54**, 871–904 (2015). https://doi.org/https://doi.org/10.1007/s00454-015-9729-3

[2] Basch, J., Guibas, L.J., Hershberger, J.: Data structures for mobile data. Journal of Algorithms **31**(1), 1 – 28 (1999). https://doi.org/https://doi.org/10.1006/jagm.1998.0988

[3] Jaromczyk, J.W., Toussaint, G.T.: Relative neighborhood graphs and their relatives. Proceedings of the IEEE **80**(9), 1502–1517 (Sep 1992). https://doi.org/10.1109/5.163414

[4] Russel, D.: Kinetic Data Structures in Practice. Ph.D. thesis, Stanford, CA, USA (2007)