Maintaining the Spatial Proximities of Objects under Motion

Zesheng Jia*[1], Anjali Jose*[2], Subhasree Methirumangalath[2], Jiju Peethambaran[1], Ramanathan Muthuganapathy[3]

[1]Saint Mary's University, Halifax, Canada,
[2]National Institute of Technology, Calicut, India ,
[3]Indian Institute of Technology, Madras, India,

Corresponding author: Jiju Peethambaran, jiju.poovvancheri@smu.ca

Abstract. We propose an algorithm to compute and maintain the spatial proximities of planar points continuously moving along predefined linear trajectories. The proximity information of moving points is captured and maintained via the well known Gabriel graph adapted for the kinetic setting, called kinetic Gabriel graph (KGG). Kinetic Gabriel Graph is built on top of the kinetic framework of Delaunay graph. Leveraging the positioning of the Delaunay circumcenters relative to the corresponding Delaunay triangles, we formulate 'Gabriel certificates' that determine whether or not an edge of a Delaunay triangle is Gabriel. Then we employ an edge tagging algorithm to maintain the set of all Gabriel edges from the Delaunay graph as the points move. The proposed algorithm has been evaluated using numerous test data, and various computational implications with respect to the topological events occurring in the data structure during the points' movement have been discussed. We also provide a conceptual demonstration of the practical potentials of the proposed algorithm in video based monitoring systems.

Keywords: Kinetic Gabriel graph, Gabriel graph, Delaunay graph, Proximity graph, Traffic monitoring.
DOI: https://doi.org/10.14733/cadaps.2021.368-382

1 INTRODUCTION

Spatial proximities of moving objects is a useful measure to predict and avoid object collisions in applications such as traffic control system [19], crowd simulation, and fluid flow rendering, among others. Depending on the application requirement, proximity graphs such as Delaunay graphs [5], Gabriel graphs [7], or Relative Neighborhood Graphs [18] 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

*Co-first authors

networks or computer graphics [9]. 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 [1, 2, 15] and experimental treatment [12, 16] of kinetic Delaunay graph can be found in the literature. Recently, Kerber et al. [10] described an algorithm to maintain the alpha shape [6] of a set of points moving along piecewise algebraic trajectories. The kinetic alpha shape is designed on top of Kinetic Delaunay triangulation. Inspired from [6], we design an algorithm to maintain Gabriel graph of a set of continuously moving points. Before we proceed further, we informally define the gabriel graph as follows. Gabriel graph of a set P of finite points in Euclidean plane consists of edges whose vertices are points from P and any closed disc for which one of these edges as its diameter contains no other points of P.

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. [3], effectively utilizes the temporal coherence of the moving objects. KDS is an algorithmic technique that maintains an attribute of interest as a group of objects move continuously. 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 that helps to find out when the structure will undergo a real combinatorial change. These information, referred to as *certificates*, are fundamental tests on the input objects with the additional property that as long as the outcomes of the certificates do not change, the combinatorial structure does not change. The main ingredient of an efficient KDS is a set of certificates that, on one hand, ensure the correctness of the configuration currently being maintained, and, on the other hand, are inexpensive to maintain as the points move.

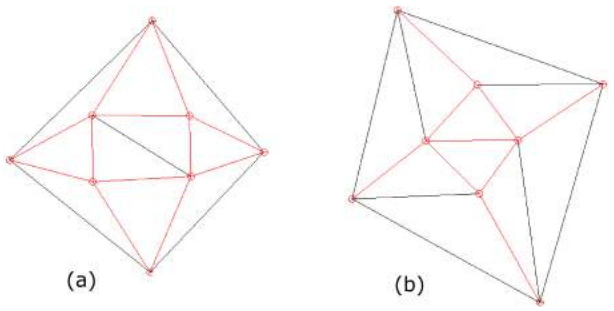


Figure 1: Gabriel graph (edges in red) at time (a) t=0 and (b) t >0 as all the points move in random directions along linear trajectories. Please note that the underlying Delaunay graph (black and red edges) is also shown for a better illustration of the concept.

In this paper, we consider the problem of maintaining the inter-object spatial proximities of a set of moving objects via a kinetic Gabriel graph, e.g., see Figure 1. We design the Gabriel certificate functions that determines whether or not a Delaunay edge is Gabriel. While identifying the Gabriel edges from Delaunay graph in non-kinetic setting relatively straightforward, kinetic setting of this problem involves accurate findings of the time of occurrences of Gabriel events, i.e., Gabriel to non-Gabriel transition of edges and vice versa, and hence, represents a challenge. Combining the Gabriel certificates with the existing kinetic Delaunay triangulation framework, we propose an edge tagging algorithm to maintain the Gabriel edges of moving objects. The algorithm has been evaluated using different test data and a conceptual application of this structure has been demonstrated. Furthermore, we discuss the computational implications with respect to the topological events occurring in the data structure during the points' movement. Specific contributions of this work are the following.

- **Gabriel certificates:** a robust certificate function for determining whether or not a Delaunay edge is Gabriel.
- **Edge tagging algorithm:** an edge tagging algorithm which computes various topological events and occurrence times, maintains the event queue, and performs timely updates on the Gabriel graph structure as the points move.

2 PRELIMINARIES AND BACKGROUND

In this section, we define the relevant geometric structures and describe the setting for the proposed kinetic algorithm.

2.1 Notations and Definitions

Let P be a set of n planar points in general position. Let $B(x, y, r)$ be a closed disk of radius r passing through two distinct points x and y and $d(x, y)$ be the Euclidean distance between the points x and y. The Gabriel graph of P, i.e. $GG(P)$ can be formally defined as follows.

DEFINITION 1 *Gabriel Graph ($GG(P)$)*
The Gabriel graph of P is the graph whose vertex set is P and that has an edge between two vertices $x \in P$ and $y \in P$ if and only if there exists a disk $B(x, y, r)$ such that:

1. $r = \frac{d(x,y)}{2}$ *and*
2. x *and* y *are on the boundary of* $B(x, y, r)$*, and*
3. $B(x, y, r) \bigcap P \setminus x, y = \phi$.

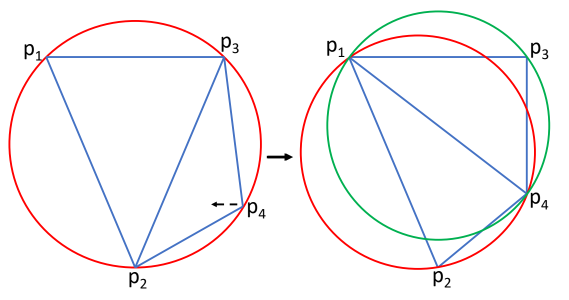
Variations of Gabriel graphs including locally Gabriel graph [11] and relaxed Gabriel graph [4] have been proposed in the literature. In Euclidean plane, different proximity graphs follow the relation: $MST \subseteq RNG \subseteq GG \subseteq Del(P)$[18], where MST, RNG and $Del(P)$ denote minimum spanning tree, relative neighborhood graph and Delaunay graph of a set of points P. In particular to $GG(P)$ and $Del(P)$, on relaxing the first condition of Gabriel graph (Definition 1), we obtain Delaunay graph (Definition 2).

DEFINITION 2 *Delaunay graph ($Del(P)$)*
The Delaunay graph of P is the graph whose vertex set is P and that has an edge between two vertices $x \in P$ and $y \in P$ if and only if there exists a disk $B(x, y, r)$ such that:

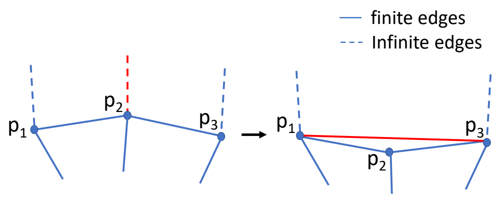
1. x *and* y *are on the boundary of* $B(x, y, r)$*, and*
2. $B(x, y, r) \bigcap P \setminus x, y = \phi$.

A Delaunay edge is called a **boundary edge** if it is incident to exactly one Delaunay triangle. Delaunay edges which are shared by two Delaunay triangles are referred to as **interior edges**. Gabriel graph can be extracted from a Delaunay graph in linear time. In Definition 3, we present a unique property of triangles in Delaunay triangulation, which is extensively used in our subsequent discussions.

DEFINITION 3 *Empty circumcircle property*
The circumcircle of any triangle in $Del(P)$ contains no points of P in its interior [13].



(a) Co-circularity



(b) Collinearity

Figure 2: 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 [16].

2.2 Kinetic Setting

Let each point $p_i \in P$ moves along a linear trajectory with a constant velocity. We assume that the coordinates of each point are linear functions of time.

$$p_i(t) = (x_i(t), y_i(t))$$

$$x_i(t) = x_i(t_0) + \Delta x_i.t$$

$$y_i(t) = y_i(t_0) + \Delta y_i.t$$

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.

In kinetic setting, a data structure is maintained right from its initial configuration at time $t = t_0$ through its final configuration at time $t = t_k$, by keeping a collection of *certificates*. The moment in time at which a certificate changes its sign is referred to as an *event*. First, future events of the data structure are computed and stored in a priority queue where the priority being the order in which they occur. A future event is detected by finding the smallest root of a non-zero certificate at time t with an additional constraint that the root is greater than t. A topological event may generate new events and invalidate some of the already computed events. In such cases, the new events are pushed into the priority queue and the invalid events are removed. Like in [10], we also assume that no two events occur at the same moment in time.

2.3 Flip Events

As the proposed KGG algorithm is built on top of kinetic Delaunay triangulation (KDT), we review the relevant certificate functions of KDT [16]. The main event that triggers a combinatorial change in the kinetic Delaunay triangulation is *flip event* [16] which happens due to either of the following two configurations.

- Co-circular: Four points of P lie on a circle that contains no other points of P

- Collinear: Three points lie on a line, and one of the half planes bounded by this line contains no points of P

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 [10]. 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 (Definition 3) is violated. The topological correctness of the local configuration involving the two triangles is then regained via edge flipping as shown in Figure 2. 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 2(a) shows an example of flip event in which the point p_4 triggers a topological event when it moves into the Delaunay circle of $\triangle p_1p_2p_3$ at time t_3. This leads to an edge flipping as shown in Figure 2 (a).

In computational geometry, many predicates are evaluated by determining the sign of an algebraic or arithmetic expression on the coordinates of the primitive objects [8]. In the context of KDT, a transition at a co-circular configuration of any two adjacent Delaunay triangles are controlled through a certificate function formulated over incircle test [17]. Consider a triangle, $\triangle p_1p_2p_3$ and a point p_4. Let the coordinates of p_1, p_2, p_3 and p_4 are (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) respectively. To perform the incircle test, all the four points are first lifted onto a paraboloid, where each projected point $\hat{p}_i$ will get the coordinates $(x_i, y_i, {x_i}^2 + {y_i}^2)$. Then, the spatial location of p_4 with respect to the circumcircle of $\triangle p_1p_2p_3$ are determined by considering a plane through $\hat{p}_1, \hat{p}_2$ and $\hat{p}_3$ and then determining whether the point $\hat{p}_4$ lies above, below or on this plane. This is determined by evaluating the sign of the determinant given in Equation 1, which is zero when all the four projected points are co-planar in the lifted space.

$$I_c(t) = \begin{vmatrix} x_1(t) & y_1(t) & x_1(t)^2 + y_1(t)^2 & 1 \\ x_2(t) & y_2(t) & x_2(t)^2 + y_2(t)^2 & 1 \\ x_3(t) & y_3(t) & x_3(t)^2 + y_3(t)^2 & 1 \\ x_4(t) & y_4(t) & x_4(t)^2 + y_4(t)^2 & 1 \end{vmatrix} \tag{1}$$

Incircle test returns positive if p_4 lies inside the circle through p_1, p_2, p_3 where the points appear along the circle in counter-clockwise direction. However, if the points p_1, p_2, p_3 reverses their orientation (clockwise order), then the incircle test reverses the sign of its output, i.e., a positive sign will then corresponds to the exterior of the circle.

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 2 (b). Collinearity among the boundary points can be determined using the certificate function (orientation test) in Equation 2.

$$I_l(t) = \begin{vmatrix} x_1(t) & y_1(t) & 1 \\ x_2(t) & y_2(t) & 1 \\ x_3(t) & y_3(t) & 1 \end{vmatrix} \tag{2}$$

$I_l(t)$ is evaluated to 0 when all three points are collinear.

3 THE ALGORITHM

In this section, we describe the additional certificate functions and rules to extend KDT to Kinetic Gabriel Graph and present the edge tagging algorithm for maintaining KGG.

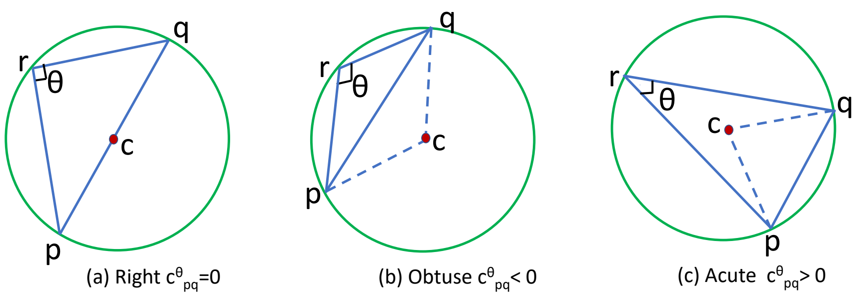

r
θ
q
c
p
(a) Right c^{θ}_{pq}=0
r
θ
q
c
p
(b) Obtuse c^{θ}_{pq}< 0
r
θ
q
c
p
(c) Acute c^{θ}_{pq}> 0

Figure 3: Signs of the angle certificate function with respect to the edge (pq) for different types of triangles.

Algorithm 1: KGG(P)

```
   Input: A set of planar points $P = \{p_0, p_1, .., p_n\}$ where each $p_i \in P$ moves along a linear trajectory
          with constant velocity
   Output: Continually updated Gabriel graph of $P$, $KGG(P)$
 1 Construct the Delaunay graph, $DT(P)$ and tag all the Gabriel edges;
 2 Initialize $KGG(P) = DT(P)$ and make it a global structure;
 3 Let $t_{curr}$ be the current time, $E$ be a temporary event variable;
 4 Let PQ be a global heap priority queue to hold future Gabriel and flip events, sorted in the ascending
     order of their time of occurrences;
 5 foreach adjacent triangle pair $T_i$, $T_j$ in KGG(P) do
 6     Compute the next future flip event $E_{T_{ij}} = \{T_{ij}, t_t\}$ at time $t_t$;
 7     if $E_{T_{ij}}$ exists & $t_t > t_{curr}$ then
 8         PQ.push($E_{T_{ij}}$);
 9     end
10 end
11 foreach internal edge $e_{ij}$ in KGG(P) do
12     UpdateGabriel($e_{ij}$ ,$t_{curr}$)(Algorithm 3);
13 end
14 while PQ not empty do
15     ProcessEevent($t_{curr}$) (Algorithm 2);
16 end
```

3.1 Angle and 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 circumcenters of the incident Delaunay triangles. We exploit the fact that the circumcenter of a right triangle lies on its longest edge and the circumcenters of acute and obtuse triangles lie in the interior and exterior of the triangles, respectively as illustrated in Figure 3. This immediately gives us a relation between the circumcenter location and the angle at the opposite vertex with respect to any edge of a triangle. Let pq and c be an edge and the circumcenter 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 circumcenter 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 obtuse 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 circumcenter c of a triangle, $\triangle pqr$ with respect to the edge pq is determined using the *angle certificate function* presented in Equation 3. In Equation 3, the coordinates of the circumcenter (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 3 (b)). A positive $\mathcal{C}^{\theta}_{pq}$ indicates an acute θ (Figure 3 (c))and a right angle at r evaluates $\mathcal{C}^{\theta}_{pq}$ to zero (Figure 3 (a)).

$$\mathcal{C}^{\theta}_{pq} = sign\begin{vmatrix} p_x(t) & p_y(t) & 1 \\ q_x(t) & q_y(t) & 1 \\ r_x(t) & r_y(t) & 1 \end{vmatrix} \times sign\begin{vmatrix} p_x(t) & p_y(t) & 1 \\ q_x(t) & q_y(t) & 1 \\ c_x(t) & c_y(t) & 1 \end{vmatrix} \quad (3)$$

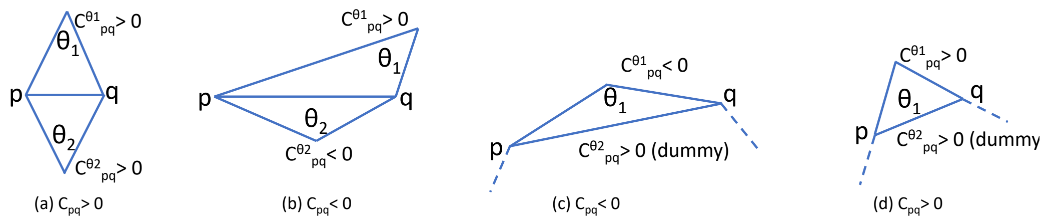


Figure 4: Gabriel certificates for interior and boundary Delaunay edges. (a) Gabriel edge, (b) Non-Gabriel edge and (c)-(d)Gabriel certificates for boundary edges.

$C^{\theta_1}_{pq}$	$C^{\theta_2}_{pq}$	$C_{pq} = sign(C^{\theta_1}_{pq}) \times sign(C^{\theta_2}_{pq})$
> 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 4. A Delaunay edge is never shared by two obtuse triangles as it violates the empty circumcircle 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. If its sign is positive, we tag the edge as Gabriel. More specifically, a non-Gabriel to Gabriel transition of edge pq occurs at time t where t is the greatest of the roots of the two associated positive angle certificates, i.e., $C^{\theta_1}_{pq}$ and $C^{\theta_2}_{pq}$. Similarly, a Gabriel to non-Gabriel transition of pq occurs at a time t where t is the smallest of the roots of the two associated negative angle certificates. To make the algorithmic steps consistent, we use dummy certificate (refer to Figure 5) functions corresponding to the infinite triangles of the boundary edges. The dummy certificate functions are always set to a positive quantity.

Algorithm 2: ProcessEvent(t_{curr})

```
Input: The current time, $t_{curr}$.
Output: Updated Gabriel graph and the event queue.
$E \leftarrow$ root(PQ);
if $E$ is a flip event corresponding to $T_{ij}$ then
    Let $E.T_i = p_1p_2p_3$ and $E.T_j = p_1p_4p_2$ be the two adjacent triangles involved;
    if $E.T_i$ is invalid or $E.T_j$ is invalid then
        Discard $E$ and exit;
    end
    Swap the common edge of $E.T_i$ and $E.T_j$, i.e., $E.T_i = p_1p_4p_3$ and $E.T_j = p_2p_3p_4$;
    foreach triangle $N$ sharing a common edge with $E.T_i$ do
        if a nearest future flip event $E_{in} = \{T_{in}, t_{in}\}$ at time $t_{in}$ exists between $E.T_i$ and N then
            PQ.push($E_{in}$);
        end
    end
    foreach triangle N(N $\neq T_i$) sharing a common edge with $E.T_j$ do
        if a nearest future flip event $E_{jn} = \{T_{jn}, t_{jn}\}$ at time $t_{jn}$ exists between $E.T_j$ and N then
            PQ.push($E_{jn}$);
        end
    end
    foreach edge $e_{ij}$ in the quadrilateral $p_1p_2p_3p_4$ including the diagonal do
        UpdateGabriel($e_{ij}$ ,$t_{curr}$)(Algorithm 3);
    end
end
else if $E$ is Gabriel event corresponding to edge $e_{ij}$ then
    if $e_{ij}$ is non-Gabriel & $E.t_{ij} \leq t_{curr}$ then
        Tag $e_{ij}$ as Gabriel;
    end
    else if $e_{ij}$ is Gabriel & $E.t_{ij} \leq t_{curr}$ then
        Tag $e_{ij}$ as non-Gabriel;
    end
    UpdateGabriel($e_{ij}$ ,$t_{curr}$)(Algorithm 3);
end
```

3.2 Kinetic Gabriel Graph Algorithm

Initialization. The pseudo-code for maintaining KGG for moving points is presented in Algorithm 1. Initially, (at time t=t_0) all the edges of the Delaunay triangulation are tagged (Gabriel or non-Gabriel) based on the Gabriel certificates. Kinetic Gabriel graph (KGG) is then initialized to the edge tagged Delaunay triangulation. All the events from the initial KGG are computed and stored in a heap based priority queue called *event queue* where the events are ordered according to their times of occurrence. The time of occurrence (t_t in line 6 of Algorithm 1) of an event is obtained through solving the corresponding certificate function.

Event handling. During the simulation, the algorithm retrieves the next event from the root of the event queue and handles it using the event processing routine. The algorithm continues this process (refer to the lines 14-17 of Algorithm 1) until the event queue is empty. The function for the processing of various events, i.e., flip and Gabriel events is presented in Algorithm 2. A flip event results in five new flip certificate updates (lines 7-16, Algorithm 2). Further, it also 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. All the new events as a result of these updates are pushed into the queue in the order of their occurrences. Similarly, when an edge undergoes Gabriel to non-Gabriel or vice versa transition (lines 21-27 of Algorithm 2), the corresponding Gabriel tag of the edge is updated (Algorithm 3). Figure 5 presents a minimal illustration of the algorithm.

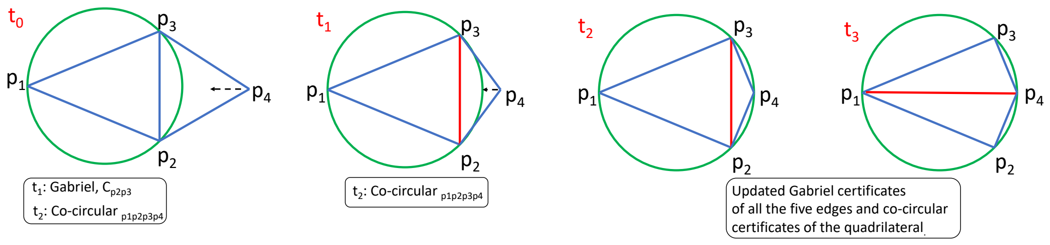


Figure 5: An illustration of the event processing and topological updates : Events are popped of the event queue and processed according to their time of occurrence (t_1 & t_2). Each event processing calls for an edge flipping or edge tag update in the graph. All the new events are pushed to the event queue at t_3. Blue lines indicate Gabriel edges. The time instants follow the order, $t0 < t1 < t2 < t3$.

Solving the certificates. The count and multiplicity of the roots of the equation corresponding to the cocircular event depends on which points are moving and their velocities. Here, we are only interested in finding the future events. Thus, we do not have to search for all roots of the equation. We have to find only those roots which are greater than or equal to the current time. 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 $I_c(t) = 0$ ($I_l(t) = 0$ for boundary configurations, see Equation 2) needs to be solved where *I_ c(t)* (Equation 1 represents the determinant corresponding to the incircle test matrix. The method adopted to solve this equation is Sturm Sequences[14] which enjoys the advantage that it gives the count of real roots in any interval [a,b] and their multiplicities.

The polynomial corresponding to the angle certificate (Equation 3) is a product of two polynomials of degree two (assuming linear trajectories) and therefore, can be of degree up to 4. For triangulation data structure with consistently oriented triangles, the angle certificates are primarily controlled by the respective circumcenter positions and hence, the roots of the angle certificates are found by solving the degree two polynomial formulated over the collinearity check of the circumcenter with the edge vertices. Gabriel certificate is the product of the signs of two angle certificates. However, the event time of a Gabriel to non-Gabriel transition is taken as the root of the negative angle certificates and the event time of a non-Gabriel to Gabriel transition is taken as the largest root of the two corresponding positive angle certificates (refer to Algorithm3). So, in effect, the KGG part of the algorithm needs to solve only degree 2 polynomials and a few checks on certificate signs.

4 RESULTS AND DISCUSSION

The proposed algorithm for KGG is implemented in C# using Visual Studio 2017. The input is given as a set of x, y coordinates of points along with their directional vectors. Points can be stationary (with velocity zero) or moving.

Algorithm 3: UpdateGabriel(e_{ij}, t_{curr})

```
Input: The edge $e_{ij}$ & the current time, $t_{curr}$.
Output: Updated tag for $e_{ij}$.
Compute new angle certificates $C^{\theta_i}_{e_{ij}}$ and $C^{\theta_j}_{e_{ij}}$ ;
Let $C^{\theta_i}_{e_{ij}}$ and $C^{\theta_j}_{e_{ij}}$ occur at times $t_i$ and $t_j$, respectively;
if $e_{ij}$ is non-Gabriel then
    if $max(t_i, t_j) \leq t_{curr}$ then
        Tag $e_{ij}$ as Gabriel;
    end
    else
        $E = \{e_{ij}, max(t_i, t_j)\}$;
        PQ.push($E$);
    end
end
else
    Let $C^{\theta_i}_{e_{ij}}$ be the negative angle certificate;
    if $t_i \leq t_{curr}$ then
        Tag $e_{ij}$ as non-Gabriel;
    end
    else
        $E = \{e_{ij}, t_i)\}$;
        PQ.push($E$);
    end
end
```

4.1 Sample Results

Shape Samples. Example kinetic Gabriel graphs for sets of points sampled from human and giraffe shapes are shown in Figures 6 and 7, respectively. In Figure 6, all the points sampled from the hand part of the shape as highlighted in the green box moves upward with a constant velocity. KGGs at various time instants have ben showed in Figure 6. As the points move, the spatial proximities of the moving points with the stationary points change and consequently, the Gabriel events occur thereby leading to updated Gabriel graphs. In Figure 7, we set the linear trajectory movements to the points sampled from the frontal leg of the giraffe shape. Gabriel graphs of the giraffe shape samples at various time instants show the changed Gabriel status of the edges within and in the premise of the frontal leg of the giraffe as highlighted in the green box. However, the topological structures of the stationary points are minimally affected in both the examples.

Random Points. Figure 8 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 8(a). When t>0, the points with non-zero velocity move and the corresponding Gabriel graphs are shown in Figures 8(b)-8(d). As the points move, the graph maintains the Delaunay property and the edges satisfying Gabriel condition are highlighted (in red color). Figure 9 shows an example of Gabriel graph maintained when all the input points move.

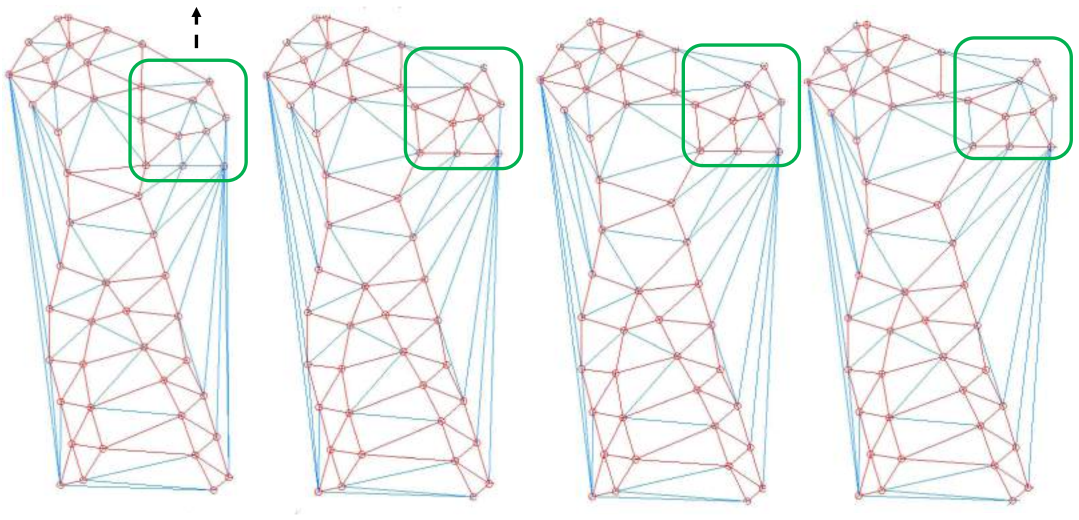


Figure 6: Gabriel graph (edges in red) at time $t = 0$ (left), $t = 1000ms$ (middle left), $t = 2000ms$ (middle right) and $t = 3000ms$ (right). The samples along the hand part of the body shape move up along linear trajectories. The underlying Delaunay graph (blue and red edges) is also shown.

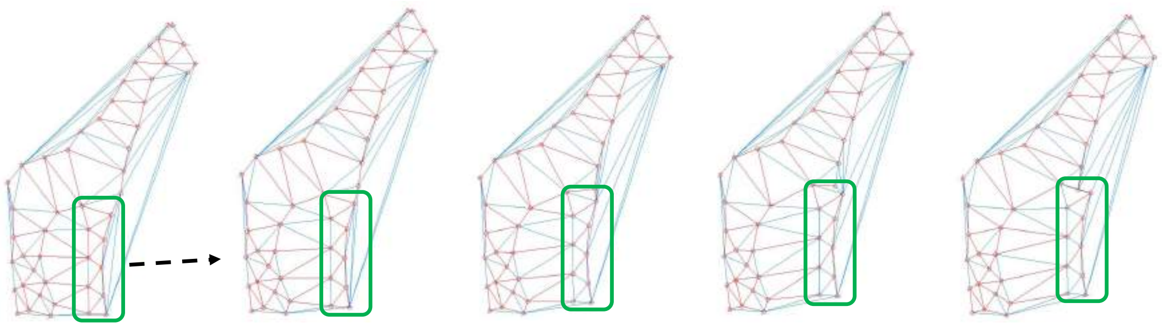
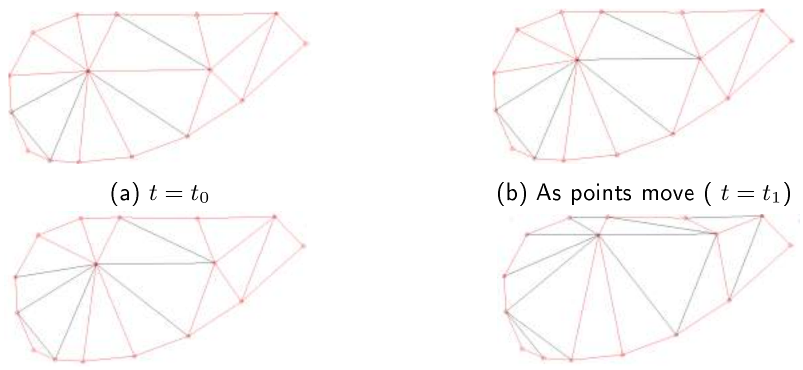


Figure 7: Gabriel graph (edges in red) at times (left to right) $t = 0$, $t = 1000ms$, $t = 2000ms$, $t = 3000ms$ and $t = 4000ms$. The samples along the front legs of the giraffe shape move up along linear trajectories. The underlying Delaunay graph (blue and red edges) is also shown.

4.2 Topological Events

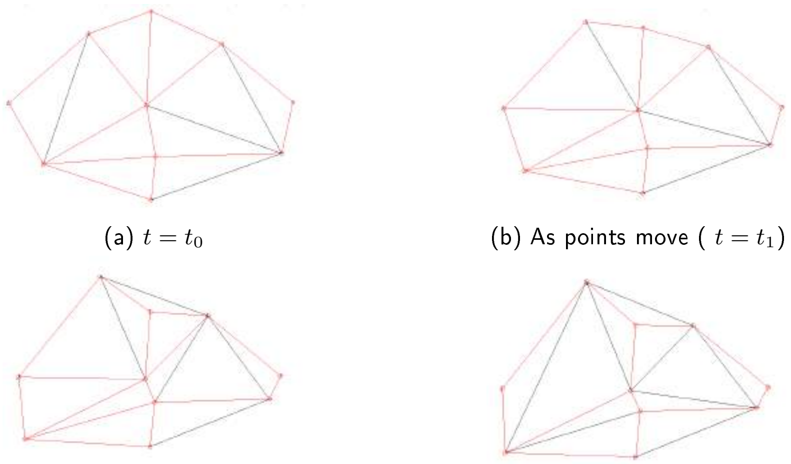
Figure 10(a) represents the relation between number of points and number of events. Here, all the points are moving with linear trajectories. The Gabriel events except those triggered by cocircular events are counted. As the number of moving points increases, the number of events increases drastically. Also, we can see that a considerable number of Gabriel events occur which are not triggered by cocircular events. Figure 10(b) represents the number of executed and discarded Gabriel and cocircular events. This result was obtained from a randomly chosen 100 points among which a certain percentage were moving along random linear trajectories. The experiment was conducted for an interval of 10 seconds. Events are discarded when the triangles involved



(a) $t = t_0$ (b) As points move ($t = t_1$)

(c) $t = t_2$ (d) $t = t_3$

Figure 8: Kinetic Gabriel graph of 16 points, where two points in the middle move in linear trajectories. Red and black lines indicate the Gabriel and non-Gabriel edges of the corresponding Delaunay graph, respectively. The time instants follow the order, $t_3 > t_2 > t_1 > t_0$. The outward edges represent edges connected to the infinite vertex.



(a) $t = t_0$ (b) As points move ($t = t_1$)

(c) $t = t_2$ (d) $t = t_3$

Figure 9: Kinetic Gabriel Graph where all points are moving : $t_3 > t_2 > t_1 > t_0$

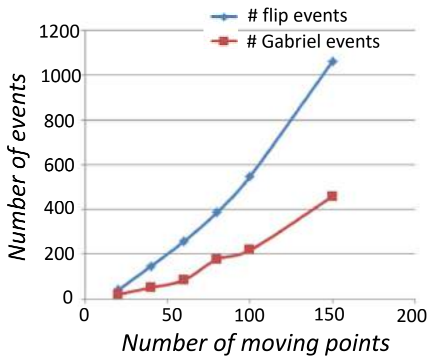
in the event no longer exist. As we can see, the number of discarded events is always greater than the number of executed events. Table 2 represents the number of certificate functions solved and processed during a time period of 10 seconds where all the input points have a non-zero velocities. As the size of 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.

4.3 A Potential Application

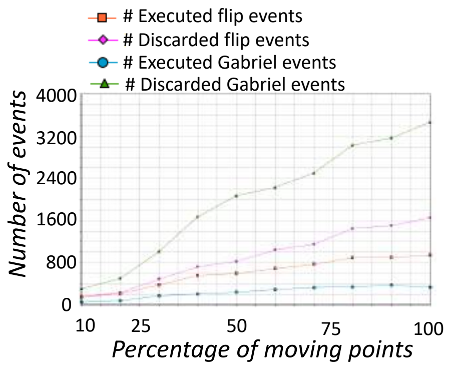
KGG can be used in video based collision avoidance systems. Let us consider the real time video feed from a traffic camera at a road intersection. Proximity information of the vehicles approaching the intersection can

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

Table 2: Number of moving points and number of certificates solved



(a) Delaunay Vs. Gabriel events



(b) Executed Vs. Discarded events

Figure 10: Summary of various events including executed and discarded appeared in the kinetic Gabriel graphs of different point sets with varying sizes.

be captured via Gabriel graph. The vehicles in each frame can be detected using a robust object detection algorithm from computer vision. 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 the directional vector 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. Figure 11 showcases a few frames of a traffic video with the corresponding Gabriel graphs overlapped. Please note that this is a conceptual prototype, and to realize this practically, many factors need further investigation, e.g., how to account for non-uniform velocities and non-linear paths of vehicles.

5 CONCLUSIONS AND FUTURE WORK

We presented an algorithm to maintain Kinetic Gabriel graph of points moving along linear trajectories with uniform velocities. The main contribution is the design of a new certificate function for determining Gabriel edges from Delauny graphs. While this is quite straightforward for stationary points, the kinetic setting demands a more elegant solution to accurately finding the times at which topological changes to the Gabriel graph occurs. We leverage the spatial locations of Delaunay circumcenters relative to the Delaunay triangles to formulate the Gabriel certificate function. The current algorithmic design can still be improved in terms of speed optimization. One way of improving the performance would be finding the redundant and/or obsolete events ahead, so that the polynomials for the same need not be solved, thereby saving considerable computational

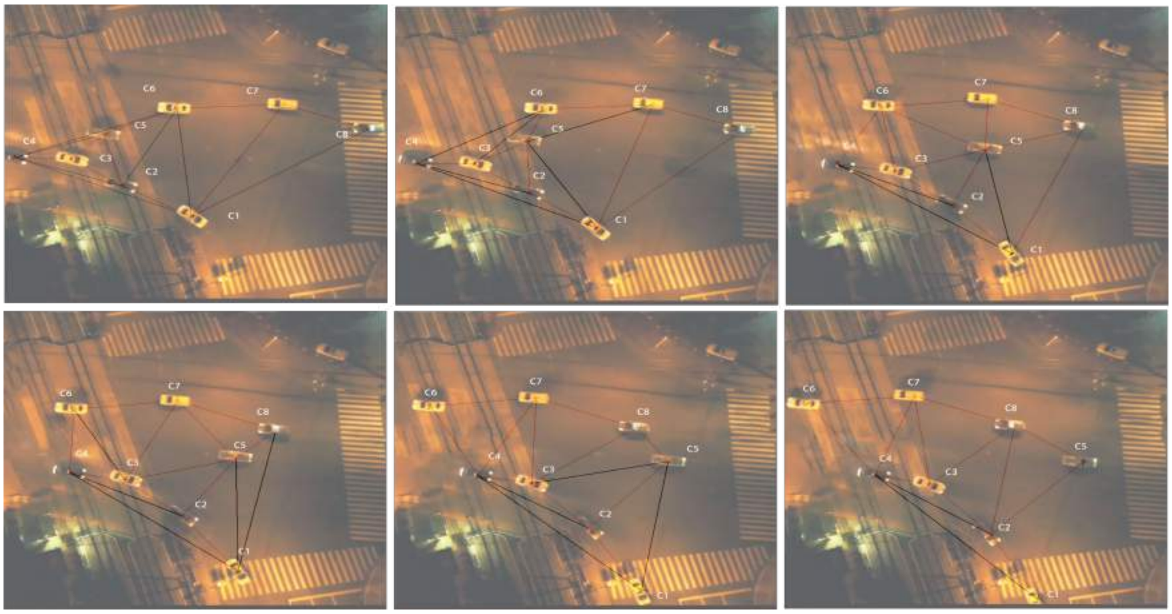


Figure 11: An illustration of how KGG can be used in video based collision avoidance systems. Kinetic Gabriel graph of cars moving at an intersection overlaid on a set of corresponding images. Red edges indicate cars in potential collision zone. When used with an appropriate threshold, the proposed data structure is useful in similar collision avoidance applications in kinetic sceneraio.

time. Further extensions aside from kinetic Gabriel graph in 3D include addressing other types of point movements such as quadratic or piece-wise linear trajectories and non-uniform velocities. As illustrated in the discussion, adapting the algorithm for the collision avoidance applications is another promising direction to pursue.

ACKNOWLEDGEMENTS

We would like to thank Thomas Vomacka for providing the implementation of Kinetic Delaunay triangulation [18].

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. http://doi.org/https://doi.org/10.1007/s00454-015-9729-3.

[2] Agarwal, P.K.; Wang, Y.; Yu, H.: A two-dimensional kinetic triangulation with near-quadratic topological changes. Discret. Comput. Geom., 36(4), 573–592, 2006. http://doi.org/10.1007/s00454-006-1266-7.

[3] Basch, J.; Guibas, L.J.; Hershberger, J.: Data structures for mobile data. Journal of Algorithms, 31(1), 1 – 28, 1999. ISSN 0196-6774. http://doi.org/https://doi.org/10.1006/jagm.1998.0988.

[4] Bose, P.; Cardinal, J.; Collette, S.; Demaine, E.D.; Palop, B.; Taslakian, P.; Zeh, N.: Relaxed gabriel graphs. In In Proc. Canadian Conf. on Computational Geometry, 169–172, 2009.

[5] Delaunay, B.: Sur la sphere vide. Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk, 7, 793–800, 1934.

[6] Edelsbrunner, H.; Kirkpatrick, D.; Seidel, R.: On the shape of a set of points in the plane. Information Theory, IEEE Transactions on, 29(4), 551 – 559, 1983. ISSN 0018-9448. http://doi.org/10.1109/TIT.1983.1056714.

[7] Gabriel, R.K.; Sokal, R.R.: A new statistical approach to geographic variation analysis. Systematic Zoology, 18(3), 259–278, 1969. http://doi.org/https://doi.org/10.2307/2412323.

[8] Guibas, L.; Russel, D.: An empirical comparison of techniques for updating delaunay triangulations. In Proceedings of the Twentieth Annual Symposium on Computational Geometry, SCG '04, 170–179. ACM, New York, NY, USA, 2004. ISBN 1-58113-885-7. http://doi.org/https://doi.org/10.1145/997817.997846.

[9] Jaromczyk, J.W.; Toussaint, G.T.: Relative neighborhood graphs and their relatives. Proceedings of the IEEE, 80(9), 1502–1517, 1992. ISSN 0018-9219. http://doi.org/10.1109/5.163414.

[10] Kerber, M.; Edelsbrunner, H.: 3d kinetic alpha complexes and their implementation. In P. Sanders; N. Zeh, eds., Proceedings of the 15th Meeting on Algorithm Engineering and Experiments, ALENEX 2013, New Orleans, Louisiana, USA, January 7, 2013, 70–77. SIAM, 2013. http://doi.org/10.1137/1.9781611972931.6.

[11] Li, X.Y.; Calinescu, G.; Wan, P.J.: Distributed construction of a planar spanner and routing for ad hoc wireless networks. In Proceedings. 21^{st} Joint Conference of the IEEE Computer and Communications Societies, vol. 3, 1268–1277 vol.3, 2002. ISSN 0743-166X. http://doi.org/10.1109/INFCOM.2002.1019377.

[12] Manhaes de Castro, P.M.; Tournois, J.; Alliez, P.; Devillers, O.: Filtering relocations on a delaunay triangulation. Computer Graphics Forum, 28(5), 1465–1474, 2009. http://doi.org/10.1111/j.1467-8659.2009.01523.x.

[13] O'Rourke, J.: Computational Geometry in C. Cambridge University Press, New York, NY, USA, 2nd ed., 1998. ISBN 0521640105. http://doi.org/https://doi.org/10.1017/CBO9780511804120.

[14] Ralston, A.; Rabinowitz, j.a., Philip: A first course in numerical analysis. New York : McGraw-Hill, 2d ed ed., 1978. ISBN 0070511586. http://digitool.hbz-nrw.de:1801/webclient/DeliveryManager?pid=2332893&custom_att_2=simple_viewer.

[15] Rubin, N.: On kinetic delaunay triangulations: A near-quadratic bound for unit speed motions. J. ACM, 62(3), 2015. ISSN 0004-5411. http://doi.org/10.1145/2746228.

[16] Russel, D.: Kinetic Data Structures in Practice. Ph.D. thesis, Stanford, CA, USA, 2007. AAI3253531.

[17] Shewchuk, J.R.: Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. Discrete & Computational Geometry, 18(3), 305–363, 1997. http://doi.org/https://doi.org/10.1007/PL00009321.

[18] Toussaint, G.T.: The relative neighbourhood graph of a finite planar set. Pattern Recognition, 12(4), 261 – 268, 1980. ISSN 0031-3203. http://doi.org/https://doi.org/10.1016/0031-3203(80)90066-7.

[19] Vomacka, T.: Delaunay triangulation of moving points. In Proceedings of the 12th Central European Seminar on Computer Graphics, 67–74, 2008.