Abstract—Tracking, monitoring and management of moving objects become increasingly important in modern geospatial application. An important example of such applications is monitoring and management of a dynamic scene captured by a network of videos or sensors for ensuring for example the security of the people during a public event, or for leading emergency or rescue teams in real environment. While current GIS are widely used to represent and manage spatiotemporal phenomena, their applications for 3D dynamic environments are very limited due to the 2D static nature of most of existing spatial data structures. In this paper, we first present a review of important requirements for efficient management of a dynamic scene. Next, we present a 3D kinetic data structure based on Delaunay tetrahedralization and Voronoi diagram that allows representation of static and moving objects and their interactions in a 3D dynamic environment, simultaneous tracking of a large number of moving objects, detection of the important events and collisions and other interesting analysis of a dynamic scene. Finally, we discuss other potentials and limitations of the proposed model and suggest some new research avenues regarding the improvement of the proposed model.

Keywords— Moving objects; dynamic scene surveillance; kinetic data structures; Voronoi diagram; GIS

I. INTRODUCTION

Tracking, monitoring and management of moving objects become increasingly important in modern geospatial application. An important example of such applications is monitoring and management of a dynamic scene captured by a network of videos or sensors for different purposes including public security, leading emergency and rescue teams or planning and monitoring a military operation. These applications are inherently dynamic and three dimensional where in addition to the representation and management of a 3D environment itself, we need to represent and manage several moving objects, analyze their trajectories, detect their possible collision with other moving objects and to predict and detect important events, etc.

While current GIS are widely used to represent and manage spatiotemporal phenomena, their applications for the representation and management of 3D dynamic environments, such as a dynamic scene, are very limited due to the 2D static nature of most of existing spatial data structures used in those systems. For an efficient representation and management of a 3D dynamic scene, we need spatial data structures that not only are adapted rapidly to the local changes occurring in such an environment but also allow us to perform different types of spatiotemporal queries and analysis on the moving objects within the dynamic environment.

There have been several attempts to use different types of hierarchical data structure to manage moving objects within a dynamic scene. These include KD trees, BSP trees, and the octree data structures etc. [8], [13]. Generally, using these approaches, the scene is partitioned into smaller regions and the objects are stored into the leaf node of the trees. Within a static scene, where the position of objects does not change with respect to the time, finding an object within the scene is a relatively simple task. This implies that it is possible to make a tree traversal operation and find the required object stored in the tree. In dynamic scene, however, the tree structure cannot be static and need to be changed in the presence of one or more moving objects. This will need a reconstruction of the whole data structure for most of the changes occurring within the dynamic scene.

The Octree data structure has been widely applied in three dimensional spaces compared to the BSP tree and KD tree. The data structure allows an irregular partition of the space. The data structure is usually more adaptive to the reality of the scene compared to the other regular tessellations of the space (e.g. 3D raster) and requires less storage memory where the environment is relatively homogeneous. For dynamic scene management, dynamic octree data structure has been proposed by several authors [24], [10]. However, it is still difficult to reconstruct only the part of the data structure where changes are occurred. It is also difficult to manage several changes in different parts of a dynamic scene using the data structure. Reference [10] proposes several assumptions prior to the application of such data structure for the management of moving objects within a dynamic scene, however these assumptions add more complexity to the data structure and its application for the management of a dynamic scene. Detection and management of important and vital events within a scene is also difficult by such a data structure.

In order to overcome these problems, in this paper, we present a 3D kinetic data structure based on Delaunay tetrahedralization and Voronoi diagram that allows
representation of static and moving objects and their interactions in a 3D dynamic environment. The kinetic spatial data structure proposed here, allows simultaneous tracking of several moving objects in the scene. This is essential to an efficient monitoring and management of complex dynamic scenes where a large number of moving objects should be monitored simultaneously. The data structure allows also the detection of the important events that are defined based on some spatial and temporal criteria within the dynamic scene. This helps the dynamic scene manager to better focus on the important events and to respond adequately and timely to the situation present in the scene. Analysis and visualization of trajectories of the moving objects and their topological relations in a 3D environment is another important possibility that the data structure offers to the user. The proposed data structure may also help to study the dynamic behavior of the moving objects within the dynamic scene.

The remainder of this paper is organized as follows: in section 2.1 we present the foundations of the proposed kinetic data structure and then through sections 2.2 to 2.4, we describe how such a data structure can be adapted to the management of moving objects in a 3D dynamic environment. Next, in section 2.5, we present several complexities related to the data structure and its maintenance. In section 3, the potentials and limitations of the data structure for the monitoring and management of a dynamic scene are presented and discussed. Finally conclusions and future works are presented in the section 4.

II. PROPOSED APPROACH BASED ON 3D VORONOI DIAGRAM

In this section, we present the foundation of the proposed kinetic data structure for monitoring and management of moving objects in a 3D dynamic environment. The data structure is based on Delaunay tetrahedralization and its dual 3D Voronoi diagram. A formal definition of such data models are presented in the following section.

A. Definition

First let consider a dynamic scene filled with a number of moving objects. Let $S(t)$ be the set of moving objects in R3 and each object is moving along a trajectory in such a way that the trajectory and speed of each object can change over time. A Voronoi diagram (VD) for the point set $S(t)$ for a given time, is constructed by partitioning the 3D space into regions with one region for each point, so that all the points of the space within the region generated by the point $p_i(t) \in S(t)$ are closer to $p_i(t)$ than any other point in $S(t)$ (Fig.1a); that is:

$$V_{p_i(t)} = \{ x \in \mathbb{R}^3 | \|x - p_i(t)\| \leq \|x - p_j(t)\|, \forall p_j \in S(t) \}$$

The 3D Delaunay Triangulation of $S(t)$ is the tetrahedralization of $S(t)$ such that there are no points of $S(t)$ inside the circumsphere of any tetrahedron (Fig.1b). This resulting tetrahedralization is unique for the point set $S(t)$, when the points are in general position, i.e. when no five or more of the given points are co-spherical [2].

3D Delaunay triangulation and 3D Voronoi diagram of a set of points are dual structures. The duality between VD and DT structures is based on specific correspondences between geometric elements of the two data structures. In 3D space, each Delaunay tetrahedron becomes a Voronoi vertex, a Delaunay edge becomes a Voronoi face, each Delaunay triangular face becomes a Voronoi polyhedron and vice versa. Based on the duality between two data structures, if one data structure is constructed, the other can be easily derived from the first one.

B. Moving objects management

Having extracted objects from a dynamic scene, we can construct Voronoi diagram for those objects for a given time. For the sake of simplicity, in this paper, we address the problem for the moving objects represented as moving points in 3D space. However, Voronoi diagram can be extended to include more complex objects (e.g. line segments, polygons and volumes) that may exist in a dynamic scene. Let’s consider that we have constructed a 3D Delaunay tetrahedralization on a set of moving points. Point movement changes the configuration of the tetrahedra having the moving point as one of their vertexes. Thus, following any movement, the data structure must be updated. A kinetic data structure benefits from the fact that if the location of a point is changed without any topological changes (topological event), its spatial relationships with the neighboring points does not need to be updated. In fact, the main idea in kinetic data structure is that despite of the continuous movement of the points, the data structure maintained by local updates when topological events occur [1], [22], [4], [3], [19]. In contrast, in a static data structure, the whole data structure, even the unchanged areas is rebuilt in each time-step.
C. One moving point management

To represent continuous motion of a point within a dynamic scene using a kinetic data structure, we need not only to allow the point movement within the dynamic environment but also we need to maintain the validity of the data structure itself [22], [8], [14]. For this purpose, we need to detect all the topological events on the trajectory of the moving point, move the point to its new positions, one by one on its trajectory and finally, update the data structure. For the sake of simplicity, the evolution of adjacency relationships for a single moving point is shown in Fig. 2 in a 2D kinetic VD.

The detection of the topological events could be used as a fundamental criterion for monitoring and surveillance of a 3D dynamic scene that will be discussed later in this paper.

In a 3D DT, a topological event occurs when a point moves in or out of the circumsphere of a tetrahedron (Fig. 3). This must be detected to preserve the empty circumsphere criterion [22] and then the topological modification must be done locally to maintain the validity of the data structure [25].

In a kinetic DT, to find the topological event of a moving point, only the spatial information of the tetrahedra having the moving point as one of their vertexes and their neighbors are used and the remaining tetrahedra in the mesh do not need to be tested. This can be computed using a well-known predicted test [6] to preserve the Delaunay empty circumsphere criterion. Since in a kinetic data structure, the position of points are time dependent, then, the value of the determinant will be time dependent as well:

\[
\begin{vmatrix}
    p_x(t) & p_y(t) & p_z(t) & p^2_x(t) + p^2_y(t) + p^2_z(t) & 1 \\
    a_x(t) & a_y(t) & a_z(t) & a^2_x(t) + a^2_y(t) + a^2_z(t) & 1 \\
    b_x(t) & b_y(t) & b_z(t) & b^2_x(t) + b^2_y(t) + b^2_z(t) & 1 \\
    c_x(t) & c_y(t) & c_z(t) & c^2_x(t) + c^2_y(t) + c^2_z(t) & 1 \\
    d_x(t) & d_y(t) & d_z(t) & d^2_x(t) + d^2_y(t) + d^2_z(t) & 1 \\
\end{vmatrix} = 0
\]  

Figure 3. When the location of the point is changed without any topological changes (cases (a) and (b)) the spatial relationship does not need to be updated. A topological event occurs when a point moves in or out of the circumsphere of a tetrahedron (cases (c) and (d)).

However, the cost of generating, computing and updating the predicate function is very expensive, especially when dealing with simultaneous moving of the points on complex trajectories as seen in a dynamic scene. For example, a quadratic trajectory of a point in a 3D space results in a degree eight predicate function. As described in [6], the computational cost can be reduced by minimizing the degree of the predicate function.

To minimize the degree of the function, we assume that only one point is allowed to move at a time on a linear trajectory. Therefore, one row of the predicate determinant must be allowed to vary linearly. Equation 2 shows the predicted function for a moving point in 3D Delaunay triangulation. According to this equation, a topological event for point \( p \) occurs when \( p \) moves in or moves out of the circumsphere of the tetrahedron \( abcd \), i.e. the value of the predicate function is 0.

\[
\begin{vmatrix}
    a_x(t) & a_y(t) & a_z(t) & a^2_x(t) + a^2_y(t) + a^2_z(t) & 1 \\
    b_x(t) & b_y(t) & b_z(t) & b^2_x(t) + b^2_y(t) + b^2_z(t) & 1 \\
    c_x(t) & c_y(t) & c_z(t) & c^2_x(t) + c^2_y(t) + c^2_z(t) & 1 \\
    d_x(t) & d_y(t) & d_z(t) & d^2_x(t) + d^2_y(t) + d^2_z(t) & 1 \\
\end{vmatrix} = 0
\]
The priority of each point is defined based on the moving points are organized with respect to a given priority criterion. The priority of each point is defined based on the increasing value of the simulation time, (red numbers). In this example for event #10, the point c is the next point to be moved, because:

\[ t_{\text{simulation}}^c < t_{\text{simulation}}^a < t_{\text{simulation}}^b \]

To facilitate the management of the topological events, we used a priority queues data structure by organizing the moving points based on the increasing value of \( t_{\text{simulation}} \). Therefore, the first member of the queue which has the smallest simulation time is processed first i.e. the moving point is first moved to its new location and a local update in the mesh is carried out in the data structure for the moving point and its neighbors. For this purpose, the dynamic operation flip23 or flip32 are used (Fig.5). The flip23 or the face-to-edge flip operator converts two neighbor tetrahedra to three tetrahedra. The flip32 or the edge-to-face flip operator converts three neighbor tetrahedra to two in order to guarantee the Delaunay empty circumsphere criterion [12], [25].

Following the topological changes in the data structure, \( t_{\text{simulation}} \) is computed and updated for the points involved in this operation. As a result, the priorities of some of the moving points may change. This occurs because, when a point moves, the related circumspheres and event times of the neighboring points change. The above process is reiterated until the end of the simulation process.

\[ p_i(t) \quad p_j(t) \quad p_k(t) \quad p_i^2(t)+p_j^2(t)+p_k^2(t) \]

\[ a_x \quad a_y \quad a_z \quad d_x^2+d_y^2+d_z^2 \quad 1 \]

\[ b_x \quad b_y \quad b_z \quad h_x^2+h_y^2+h_z^2 \quad 1 \]

\[ c_x \quad c_y \quad c_z \quad c_x^2+c_y^2+c_z^2 \quad 1 \]

\[ d_x \quad d_y \quad d_z \quad d_x^2+d_y^2+d_z^2 \quad 1 \]

\[ \frac{p_i(t) \cdot p_j(t) \cdot p_k(t) \cdot p_i^2(t)+p_j^2(t)+p_k^2(t)}{a_x, a_y, a_z, d_x^2+d_y^2+d_z^2} = 0 \]
Algorithm: 3D kinetic Delaunay data structure

Input: a DT $S(t=t_0)$; trajectories and speed for $S(t)$

Output: a DT$(t=t_{final})$

Find the closest topological event
Sort the dynamic priority queue for moving points

Repeat
   1) Get first member of the queue
   2) Move it to its closest topological event
   3) Local update the data structure
   4) Update the trajectory and the attributes (e.g. Speed) for the moving point and its neighbors
   5) Update the priority queue

Until the simulation time is equal the given value;

E. Complexities related to generation and maintenance of the proposed kinetic data structure

When discussing a kinetic data structure, the initial assumption is that the point set $S(t)$ is in general position. This assumption, however, is no more valid when objects start to move within a dynamic scene. As a result, when implementing the kinetic data structure complicated and degenerate cases usually arise during the movement of points and must be taken into account. In general, in the kinetic data structure, the degeneracies occur when the distribution of points creates an ambiguity in the construction of DT$(S(t))$. For DT in $R^3$, this ambiguity occurs when five or more points lie on the same sphere, called co-spherical points and consequently the DT$(S(t))$ for this set of points is not unique.

1) Co-spherical points

Let assume the points of the set $S(t)$ at $t=0$ are in general position, thus, a co-spherical case in the kinetic data structure occurs when (Fig. 6):

1. The point moves in a Real tetrahedron circumsphere; a Real tetrahedron is a tetrahedron that is incident to the face of the tetrahedra containing $p$ but outside of them; consequently we have two locations for the topological events with respect to this circumsphere.

2. The point moves out from an Imaginary tetrahedron circumsphere; an Imaginary tetrahedron is a tetrahedron formed by the neighbors of $p$ and would exist if the moving point was moved out of its circumsphere; in this case we have only one location for the topological event with respect to the circumsphere.

3. The point trajectory is tangential with the circumsphere of the given tetrahedron; then we have two equal locations for a move-in and a move-out event.

As we can observe from these points, the point movement creates an ambiguity in the data structure that is discussed in the following paragraphs. Depending on the position of the moving point with respect to its neighbors, different solutions may be proposed to solve this ambiguity in the structure. Here, we discuss two possible configurations in which five points are co-spherical:

**Configuration (I): There are five points co-spherical in which no four points are coplanar**

When the moving point lies on circum-sphere of a real or imaginary tetrahedron (cases (1) and (2)), depending on the points configuration, the tetrahedralization is done by two or three tetrahedra as seen in Fig. 6 and one or more flips are required to replace one tetrahedralization by another. However, since every tetrahedron cannot be flipped, a flippability test must be used. According to [11], a flip23 is always possible when the points are in the configuration (6a) and as a result there are no unflippable deadlock configurations that can occur. The tetrahedra in configuration (6b) are flippable if there is the third tetrahedron to fill the convex hull of the five points. In this case, a flip32 is possible; otherwise, no flip is performed. Therefore, to update the data structure when a point moves in a real tetrahedron circumsphere or moves out from an imaginary tetrahedron circumsphere or the point trajectory is tangential with a circumsphere, one flip32 or flip23 or no flip is required. In the tangential trajectory case, it is easier to solve the ambiguity. Here we simply ignore the first topological event and take in turn the second one. As seen, there is a solution for this type of co-spherical points and this configuration is not a problematic case for kinetic data structure.

![Figure 6. Optimizing the tetrahedra when the five points are co-spherical in which no four points are coplanar (configuration (I))](image)
In this configuration, the moving point lies not only on the same sphere but also on the same plane of other three points. Therefore, two ways of tetrahedralization are possible (Fig. 7). The tetrahedra are flippable if there is a vertex on the opposite side of the coplanar point [11], otherwise no flip is performed.

As we discussed in this section, the co-spherical and co-planer configurations of five points could be resolved with the strategies explained here. In a real application, it is possible that we have more co-spherical or co-planer points that makes the problem more complex and which goes beyond the scope of this paper.

Another important complexity that must be mentioned here is the round off errors in inexact computations. This problem is due to the limited precision of the computers that complicates the implementation of the geometric algorithms and has a significant role in the robustness of such algorithms as discussed in the following section.

In fact, using the floating-point arithmetic the situations close to degeneracy positions become more difficult to decide on the correct topological relations. To solve this problem, an exact computation must be used to find the topological event. In this method, a high precision computation is carried out which is computationally expensive. To reduce the cost of exact computation, [23] proposes using the exact computation approach only where there is some doubt on the accuracy of the rounded result and otherwise an inexact computation would be more efficient for general computation purposes.

3) Collision problem between the moving objects:
Within a dynamic scene, there is always the possibility of collision between two or more moving objects. Collision between two points happens when a moving point \( p_i(t) \) meets another point of the mesh on its trajectory (Fig. 9). In the other word, a collision happens when there exists a point at the exact location where \( p_i(t) \) was supposed to be moved.

The collision detection methods commonly are studied in the fields of computational geometry, physically-based modeling, geometric modeling; robotics, computer graphics and computer simulation [21], [11], [15]. A comprehensive review of collision detection optimization methods in application to dynamic simulation can be found in [3]. These methods typically have two stages. The first stage identifies the pairs of points that can potentially be in contact at a given time by maintaining for every point a list of neighbors. The second stage decides for every pair defined by the neighborhood whether or not there is actually contact. This is done by testing the distance between two points \( p_i(t) \) and \( p_j(t) \in S \), against a tolerance. If the distance is smaller than the tolerance, there is collision detection (3):

![Figure 7. Optimizing the tetrahedra when the five points are co-spherical in which four points are coplanar configurations (II). The tetrahedra are flippable if there is a vertex on the opposite side of the coplanar point.](image)

![Figure 8. An example of errors caused by inexact computation of intersections.](image)
For collision detection, several neighborhood techniques have been used. In a straightforward approach each pair of points is considered to be neighbors. Therefore, this gives an O(n²) algorithm which is computationally very expensive. In the proposed data structure, neighborhood is clearly defined and maintained using the O(n) Delaunay and Voronoi diagram. Thus collision checks are in the worst case.

Following the collision detection stage, the central question to be answered is: “What should be done on the colliding objects?”

Form geometrical points of view, the collisions can be handled by deleting one point and reinserts it after second object along its new trajectory. However, in some applications, the trajectories of points are described by physical models such as partial differential equations in the simulation of a fluid flow, therefore, the reactions are defined according to the physical model, as a result, the trajectories, velocity and other attributes of points change. Then, in some application the collisions can be handled by merge one point and updating the physical attribute and trajectory of second one.

In the case of the monitoring and management of a dynamic scene, depending on the type of application, the detection of the collision and avoidance of the collision could be the objective of the monitoring. In other cases, we may not intervene to avoid the collision but only a collision alert may be the objective of the application. Here, the collision can represent an important event to be reported within the scene. Hence, given the maintenance of the topological kinetic data structure during the evolution of a dynamic scene, the surveillance, detection and report of collision events are possible.

![Figure 9. Collision case a) Deleting one point and reinserting it next to the second object along the trajectory, b) Deleting one point and updating the physical attribute and trajectory of second one.](image)

\[
\left| p_i(t) - p_j(t) \right| = \left( \sum_{k=1}^{3} \left| p_{ik}(t) - p_{jk}(t) \right|^2 \right)^{1/2} \leq \text{Tolerance},
\]

\[
\forall p_i(t), p_j(t) \in S(t)
\]  

III. POTENTIALS AND LIMITATIONS OF THE PROPOSED KINETIC DATA STRUCTURE FOR THE SURVEILLANCE AND MONITORING OF A DYNAMIC SCENE

A. Tracking of moving objects and analysis of their trajectories

The trajectory of a moving object can be represented by a set of line segments in three-dimensional space. In fact, a trajectory represents the geometric and temporal information that can be of key importance for some analyses. Trajectories can be analyzed in their spatial, temporal or spatiotemporal dimension by projecting them into the corresponding subspaces [16]. Analysis of the object trajectories offers the possibility of extracting information on the behavior of the moving objects and their interactions with the environment. These interactions are closely related to proximity problems. The proposed data structure based on the Voronoi diagram provides a clear definition of the proximity of the moving objects within a dynamic scene throughout the monitoring time. This can help to track people and mobile devices in the geometric models of the dynamic scene and more importantly to help recognize what activities they are carrying out.

B. Querying on the moving objects and their neighbors in 3D environment

Queries are frequently used in GIS. There are several types of queries including location queries, range queries, nearest neighbor queries and k-nearest neighbor queries. The Voronoi diagram defines adjacency relations between objects. Two objects in a Voronoi diagram are adjacent if their respective Voronoi cells share a common face. Hence, it is easy to collect k-nearest neighbors for the object of interest using its adjacent cells in the Voronoi diagram. These queries become more complex when dealing with a 3D dynamic scene. Since, in such environment, objects move and their relations change continuously (Fig. 10). Therefore, the queries must respect the time dimension. These queries aim at determining the state of moving objects in the past, present and future and their relative position with respect to each other and other static objects within the dynamic scene. A typical example of such queries is ‘report the closest ship in the sea that is moving towards another ship in difficulties’. Another type of query aims at determining the moving objects that will intersect a query window referred to as a predicative spatio-temporal window query (Kollios et al., 1999).

We may also be interested in answering queries on the moving objects and their neighbors in 3D environment such as:

a) What were the k-nearest neighbors of an object of interest in a specific moment in the past?

b) What are the closest neighbors of the object at the present time?
c) Which objects will be closer to the object of interest at a given time in the future?

Querying on the past position of objects can be done using the history of the trajectory of each object. This requires keeping track of the trajectories of the moving objects and the evolution of the topological relations between objects. Considering the number of the topological events, this may not be feasible. An alternative solution would be to return all objects to their position in a required time in the past and rebuild the part of the topology in a given region. There is still room for further investigation on this topic.

Querying nearest neighbors at present time can directly be done by the topological model. We assume that the update operation in the topological model is efficient enough and the query operation at present time includes latest changes in the neighborhood of the object of interest.

In the case of querying nearest neighbors of a moving point at future time, we need information on the trajectories of moving objects and their velocity vectors. This knowledge is necessary for the prediction of the position of the moving objects in the future. One solution for this kind of query is to allow the topological data structure to change until the moving objects arrive to their predicted position and then proceed with the computation of nearest neighbors as suggested for queries at present time. The second solution will be a reconstruction of the topology for the predicted position of the objects in the future and making the nearest neighbor queries on the model.

C. Event detection and management

During the monitoring of a dynamic scene, several kinds of events may be of interest for the users of the system. For example, the rapprochement of a particular object to a point of interest, possibility of collision between two or more moving objects, their agglomeration or dispersal and visibility of a particular object could be considered as an important event within a dynamic scene. Hence, in order to support the user in his task, these events should be detected within the dynamic scene automatically based on some spatial and temporal criteria. The kinetic data structure proposed in this paper is an event based approach for the representation of a dynamic scene. The events can not only be defined based on as the topological changes, but also we can consider the so-called trajectory events that consist of direction or property change of a moving object. These events can be combined with other spatial and spatiotemporal query operators as criteria for the kinetic data structure. Therefore, this could help to detect or predict those kinds of event in the dynamic scene and respond to it adequately and timely.

F. Limitations of the proposed approach

As mentioned earlier in this paper, the proposed data structure can currently manage only object represented by points. However, within a dynamic scene in the reality, we have often more complex objects that can be represented by lines, polygons and volumes. Although some work has been done on the Voronoi diagram for complex objects on the plane, application of such a model for a 3D dynamic environment with more realistic representation of the scene is still a very complex task and needs further investigations on the different aspects of the problem.

In addition, in this paper, we did consider all the aspects of the problem related to the real-time monitoring of a dynamic scene using a sensor or video network. The integration of data sources, processing and extraction of the moving objects from those data sources and then, introducing these information the proposed kinetic data structure, represent many complex challenges where the efficiency of the methods and tools applied for those task are very important in order to make the whole system applicable to a real world problem.

IV. CONCLUSIONS AND PERSPECTIVES

Dynamic scene surveillance and management is becoming increasingly important for many modern geospatial applications. Efficient monitoring and management of dynamic scene needs a spatial data structure that could mange the motion of moving objects within the scene. In this paper, we proposed a 3D kinetic data structure for representation and analysis of a dynamic scene. The data structure can efficiently adapt itself to the movement of the objects in the scene. This is due to the fact that the maintenance of the structure is based on a completely local operation. The kinetic spatial data structure proposed here, allows simultaneous tracking and management of several moving objects in the scene. This is essential to an efficient monitoring and management of a complex dynamic scene where a large number of moving objects should be
monitored simultaneously. The data structure allows also the detection of important events, analysis and visualization of the trajectories of the moving objects and the studying of their dynamic behavior.

Although, the data structure have many potentials for the surveillance and monitoring of a dynamic scene, but its application to a more complex real world application context is still limited and yet needs further investigations.

ACKNOWLEDGEMENT

This research work has been made possible thanks to a grant from the Natural Science and Engineering Research Council of Canada (NSERC).

REFERENCES


