Geospatial Multi-agent System for Urban Search and Rescue

HEBA GABER, SAFAA AMIN, ABDEL-BADEEH M. SALEM
Faculty of Computer & Information Sciences
Ain Shams University, Cairo, EGYPT
eng.heba.gaber@gmail.com, safy11@hotmail.com, absalem@asunet.shams.edu.eg

Abstract: - Geospatial Multi-agent Systems (GMAS) are composed of heterogeneous agents. These agents are distributed in space and time, and act with incomplete information in an uncertain environment. GMAS have many applications including disaster recovery, military command and control, avionics and traffic control. This paper presents an implementation prototype for GMAS used to solve Multi Agent Path Planning (MAPP) problem in Urban Search and Rescue (USAR) domain. The proposed architecture incorporates a combination of online algorithms with standardized languages and well known technologies (ex. SQL Server Spatial, KML, GoogleEarth and CORBA) for solving the problem. This paper focuses on geographic data manipulation, storage and visualization.

Key-Words: - Multi-agent Systems, KML, Google Earth, SQL Server Spatial, Multi-agent Path Planning, RoboCup Rescue, Urban Search and rescue.

1 Introduction
In USAR domain after a large earthquake, buildings collapse, many civilians are buried in the collapsed buildings, fires are spreading, and it becomes difficult for rescue teams to pass roads because these are blocked by debris of buildings and something else [1, 2]. The emergency services are represented by police, fire and ambulance software agents. The domain incorporates general command and control issues and concepts between heterogeneous teams of agents. Rescue agents tasks include search, building situation awareness, blockage removal, extinguishing fires, medical assessment and intervention. Accordingly agents can be broadly categorized into types based on their capabilities (Fire Agents, Police Agents, Ambulance Teams…).

RoboCup Rescue Simulation Project is a now well established RoboCup competition intended to promote research on this very serious social problem [4]. The development of the RoboCup rescue simulator [5] offered a new practical domain for RoboCup and enables the application of research results achieved in RoboCup soccer competitions to a more sociably useful problem. Based on this simulator USARSim simulator was developed based on the 3D game environment robots behaviors. USARSim introduced as an in-between research tool where multi-agent and multi-robot systems can be studied in an artificial environment offering experimental conditions comparable to reality [6]. Moreover, MOAST (Mobility Open Architecture Simulation and Tools) 4D/RCR implementation of National Institute of Standard Technologies (NIST) on the top of USARSim is used for USAR simulation as shown in [7]. Takahashi shows in [8] implementing RoboCupRescue simulation using GoogleEarth Viewer. In [9] interfacing rescue simulation with Wearable Computing devices is presented as an attempt for implementing a realistic data integration model for rescue systems using GPS devices.

This work is addressing the issue of providing synchronous collaboration systems for geographically distributed teams working under time pressure to achieve a shared goal. It is using MAPP for USAR as an example of a domain requiring a team to navigate an area, perform physical tasks, monitor adherence to a schedule, respond to unexpected and changing situations, and re-plan if necessary. In our previous work [3] a combined technique for solving MAPP problem for rescue agents was proposed. This paper presents a discussion on the further use of GIS software for solving the problem through using SQL Server Spatial for geographic data manipulation and storage then visualizing the system results using KML format GoogleEarth 3D viewer.

The remaining sections of this paper are organized as follows: section 2 explains spatial data models and methods. Section 3 explores using SQL Server for spatial data manipulation; section 4 discusses the proposed architecture; section 5 presents the technique used for MAPP. Section 6 shows implementation results. Section 7 analyze and evaluates the implementation. Section 8 concludes the paper and highlights future work.
2 Spatial Data Models and Methods
Spatial data is used to represent points, lines, and areas on a surface. Most commonly, these elements relate to actual physical locations on Earth, so can be described a geospatial data. Spatial datasets enables building operational models of the real world based upon the field and object conceptions and the use of coordinate geometry to represent the object classes. These include: discrete sets of point locations; ordered sets of points (open sets forming composite lines or polylines, and closed, non-crossing sets, forming simple polygons); and a variety of representations of continuously varying phenomena. The problem with describing a location on a planetary surface is that planets are not flat. Earth is a very complex object that can be reasonably approximated by an oblate spheroid, a (slightly) flattened sphere. Spatial data can be modeled either using Geodetic or Planar Spatial Model.

Table 1. Geodetic vs Planar model

<table>
<thead>
<tr>
<th>Definition</th>
<th>Geodetic Spatial Model</th>
<th>Planar Spatial Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>Accurate</td>
<td>Converting geographical data from a sphere to a flat surface always results in some distortion of features.</td>
</tr>
<tr>
<td>Variation</td>
<td>Calculated earth radius Conversion method</td>
<td></td>
</tr>
<tr>
<td>Calculations</td>
<td>Working with an ellipsoid and taking planetary curvature into account</td>
<td>A projection is created to flatten the geographical objects on the spheroid.</td>
</tr>
<tr>
<td>Models</td>
<td>Airy 1830 ellipsoid used in the United Kingdom’s Ordnance Survey geographic system, and the WGS84 ellipsoid used by the world’s GPS solutions</td>
<td>Mercator projection, the Peters projection, and the Lambert Conformal Conic projection</td>
</tr>
<tr>
<td>Application</td>
<td>Big geographical areas</td>
<td>Small geographical areas such as individual countries, states, and towns, or for non-projected spatial surfaces such as interior floor plans.</td>
</tr>
</tbody>
</table>

In general the choice of which geometry to use will depend on how you plan to use the data. To describe the positions of points in space, every spatial reference system is based on an underlying coordinate system. A coordinate reference is a conventional and widely accepted way of describing the position of a point from a given origin, in a given dimension. A set of n coordinates, such as (1, 2, 3... n), can therefore be used to describe the position of a point from an origin in n-dimensional space [15]. As shown in Table [1], using Geodetic Spatial model provides more accuracy though being harder in calculation. Also it suits large scale domain as the proposed one in this paper.

3 SQL Server Spatial
SQL Server Spatial is used in this implementation for geographic data storage and manipulation. With the introduction of spatial support in SQL Server, number of barriers that, until now, have prevented mainstream developers from using spatial data was reduced because:
1- Spatial data-types are included as a core component of the SQL Server database, requiring no additional components to be installed or configured.
2- Spatial operations are integrated into the existing functionality of the SQL Server Database, allowing developers to continue working within a familiar development environment using existing tools such as SQL Server Management Studio.
3- Existing SQL Server databases can be easily enriched by adding spatial data fields to their existing structure—there is no need to migrate data onto a new platform.
4- The new geometry spatial data-type conforms to accepted industry-wide standards set by the Open Geospatial Consortium (OGC).

SQL Server Spatial provides a comprehensive set of spatial methods with high performance spatial indexes. It supports three main types of geometry that can be used to represent spatial information: Points, LineStrings, and Polygons. Also it provides standardized methods used mainly for:
1- Retrieving Properties of Geometry Objects.
2- Constructing Geometry Objects.
3- Returning WKT, WKB and GML.
4- Querying Validity, Instance Type, and Geometry Collection Information.
5- Determining the Relationship between Two Geometries.
6- Creating New Geometries.

4 Proposed Architecture
In this work a simplified version of USAR implementation is presented, the main focus was given for solving MAPP problem between the contributing agents. Fig.1 shows a schematic view of the system’s architecture. The system architecture consists of several communicating modules:
1. GIS server provides initial condition of the disaster space. This implementation uses RobocupRescue simulator dataset for Kobe, Japan for simulating GIS data retrieval.

2. Kernel, a centralized server responsible for data/information storage and retrieval, coordination between heterogeneous system components, managing resources, detecting conflicts and resolving them. The Kernel is composed of many modules:
   a. Knowledge Base: contains domain specific information related to the domain, e.g. for MAPP in USAR when two agents' paths conflict priority may be assigned according to distance to target location or according to agent type, i.e. police forces have higher priority than fire fighters.
   b. Global Map: integrated map consisting of initial conditions and updated incrementally with disaster information and data reported by the operating agents.
   c. Global Path Planner: identifies the ideal path from the agent's current position to its target location. Global path planning typically ignores local transient obstacles.
   d. CPAD mission controller: responsible for conflicts detection and resolution between agents’ paths.

3. The Viewer server used to monitor the situation information using the data provided from the Kernel. It helps system operators to investigate a variety of plans for MAS coordination.

4. Heterogeneous Agents that dynamically apply to update the required databases, or to satisfy other requests from a particular application. Agent is composed of these modules:
   a. Local Map: consisting of initial conditions provided by the Kernel initially and updated with local sensory data.
   b. Local Path Planner: concerned with moving the agent along the planned path taking into account the obstacles ignored by the global plan.
   c. Sensory Data: collected by the local sensors and used for local planning.

Spatial information and agents actions are stored in SQL database server. This information are used to update agents’ information and used for global planning. At certain intervals, rescue agents report some important information discovered during its explorations to the Kernel. These updates are used by the Kernel for re-planning agents’ missions and paths if necessary. Also system operators monitoring the system can force the Kernel to re-plan the path whenever necessary.

In the proposed architecture, distributed object computing is implemented using CORBA interfaces [14]. Geographic data processing needed for local and global path planning, i.e. calculating distances, detecting conflicts, planning a collision free path from initial to target location is implemented using SQL Server Spatial. Moreover, this implementation allows the system operators to visualize the situation evolution using KML file format and GoogleEarth is used for 3D visualization environment.

5 Combined Technique for MAPP

MAPP problem can only be solved by efficiently integrating data of many observers into a single consistent view. The input to this problem is a graph. Graph edges represent the open roads at the beginning of the disaster. As the time goes by, some roads will be opened so this graph is a dynamic graph. Path planning is typically performed on one agent at a time, and broken into at least two tasks. The first, is concerned with global path planning, and identifies the ideal path from the agent's current position to its target location. Global path planning typically ignores local transient obstacles, such as other moving objects. The second, local task is concerned with moving the agent along the planned path at a reasonable speed and taking into account the obstacles ignored by the global plan. In addition to local and global planner a coordination technique
should be combined with these algorithms to manage conflicts between their paths. As suggested in the previous work [3], MAPP problem can be solved using the combination of four online algorithms:

A* algorithm: Deterministic algorithm for global planning. A* is a directed breadth-first search and combines the advantages of uniform-cost and greedy searches using a fitness function [13].

\[ F(n) = g(n) + h(n), n \in N \]  

\[ g(n) = \text{cost of getting from the initial node to n.} \]
\[ h(n) = \text{the estimate, according to the heuristic function, of the cost of getting from n to the goal node.} \]

E-GAP algorithm: for task allocation and resources management. An allocation matrix \( A \) is used to represent task allocation, where \( a_{ij} \) is 1 if task \( j \) is allocated to agent \( i \) and 0 otherwise [11]. An optimum solution to the problem is given by matrix \( A^* \), which maximizes the system reward given the agents’ capabilities, as shown in

\[ A^* = \arg\max_{A} \sum_{i=1}^{n} \sum_{j=1}^{m} K_{ij} \times a_{ij} \]

Potential Field method (PFM): PFM is used for local planning. In PFM agent is represented as an object moving under the influence of the potential created by the goals and the obstacles in the environment. While the target location exerts a force that attracts the particle \( \Box_{att} \), the obstacles exert repulsive forces \( \Box_{rep} \). At each time \( t_0 \), the motion is computed to follow the direction of the artificial force induced by the sum of both potentials [12]. As shown in (3) the net force is the summation of both attractive and repulsive forces.

\[ \Box (q_t) = \Box_{att}(q_t) + \Box_{rep}(q_t) \]

CPAD algorithm: to predict conflicts between agents’ paths and to assign priorities to agents’ according to their planned paths. CPAD is an approach for MAPP problem which combines the use of checkpoints (i.e., areas where motion conflict is likely) and dynamic priorities assignment [10].

6 MAPP for USAR Implementation

The system operation starts immediately after the disaster in USAR environment. The Kernel follows the following procedure for initiating the system:

1- Receive the initial environment conditions from GIS server (buildings, roads,...).
2- Receive agents’ locations, types and capabilities.
3- Send global information to each agent according to its location.
4- Receive the disaster information from the disaster simulators.
5- Send the initial environments conditions, agents’ and disaster’s locations to the Viewer.

After collecting the initial data, the Kernel starts global planning according to the available information at each time step. The global planning starts by assigning missions on teams of agents according to their suitability for the mission then tasks are distributed among team members based on a set of predefined rules variant to the application domain. In this system task allocation is performed using E-GAP algorithm [11]. And since the main objective is solving MAPP, therefore the execution cost mainly depends on the distance between the agent and the target location. A* algorithm is used for global path planning considering buildings as obstacles and roads as edges. Conflicts between planned paths are detected by CPAD algorithm as shown in Fig.3. For the conflicting paths the priorities are assigned according to predefined rules specific to the domain. Agents recognize the surrounding circumstances using their local sensors. The Kernel provides them with global data integrated from various locations beyond their local capabilities. The procedure for the agents is:

1- Send the initial location information to the Kernel, its capabilities and local sensory data.
2- Receive the initial condition data beyond its local capabilities from the Kernel.
3- Request the target mission.
4- Receive the allocated task, the global plan and the predicted conflicts.
5- Execute the global plan and submit its actions to the Kernel.
6- The Kernel updates its information and accordingly updates the viewer.

Agents perform local planning using PFM. As shown in Fig.2, A* algorithm for global planning can be implemented using SQL statements and geography types. The algorithm depends mainly on SQL Server Spatial operations and types. Examples:

1- Geography.STIntersects(Geography): the result determines whether two geography objects intersect.
2- Geography.STDistance(Geography): calculates the distance between two objects.

7 Analysis and Evaluation

USAR systems has been implemented using different tools and many USAR simulators are available; the main objective of this work is to implement an extensible architecture based on a set of standardized and well known technologies and
Recent Researches in Computer Science

standards such as CORBA, SQL, OCG standards, KML and GoogleEarth. This system is presented by implementing a combined technique for MAPP on USAR agents. This techniques combines online algorithms as (A*, CPAD and Potential Fields).

Implementing A* algorithm and conflict detection algorithms using SQL server spatial gives the following advantages over procedural languages from the implementation point of view:

1- Enhances application performance because the operations are done directly on the stored geographic data using SQL queries.
2- Operations can be done on set of data at once.
3- Processing can be done on geographic data directly without transform- manipulate retransform data from geodetic to planar data.

While implementing the viewer Google Earth was the best choice for a 3D visualization tool. Google submitted KML (formerly Keyhole Markup Language) to the OGC to be evolved within the OGC consensus process. KML Version 3.0 will be an adopted OpenGIS implementation specification that will have been harmonized with relevant OpenGIS specifications that comprise the OGC standards baseline. The objectives for this standards work [16]:

- International standard language for expressing geographic annotation and visualization.
- KML is aligned with international best practices and standards.
- OGC and Google will work collaboratively to engage KML implementer community.

The advantage of KML format is that it's easily readable. GoogleEarth provides rich user environment with Paths and Placemarks can be drawn on a Google Map it reveals the system implementer from implementing the viewer details and maps. GoogleEarth imagery and all GoogleEarth functionality are also available to the system operators.

8 Conclusion and Future Work

This work presents integration between well known standardized technologies and online algorithms together into a single GMAS system for solving MAPP problem for rescue agents. During the development of this system many factors were taken into account to ensure that the architecture was both well-defined and standardized. Moreover, scientific standards and conventions for units, coordinate systems, and interfaces were used whenever possible. The future work is considering a more complex scenario with larger number of teams and by including other teams as ambulance teams. The proposed model supposed that the final goal is to reach the assigned target whereas teams of agents as ambulance teams and police forces should plan their path back to deliver the injured civilians to the hospitals and the refugees to the mitigation centers.

```
AStarLoop:
if exists (select * from @OpenSet)
begin
    select @x = (select top 1 op.nodeID
    from @OpenSet op inner join
    @solutionMap sol on op.NodeID= sol.NodeID
    order by f_TotalDistance asc)
    select @xGeo= nodeGeo from @Nodes where nodeId= @x
    if(@x =@goalNodeId)
        goto ConstructPath
    delete from @OpenSet where NodeID = @x
    insert into @ClosedSet values(@x)
    delete from @neighbor_nodes
    insert into @neighbor_nodes(nodeID)
    select aNodeId from
    (select roads.headNodeId NodeId
    from @Roads roads
    where roadGeo.STIntersects(@xGeo)=1)
    union
    select roads.tailNodeId NodeId from @Roads
    where roadGeo.STIntersects(@xGeo)=1
    a where aNodeId <> @x and aNodeId not in (select NodeId from @ClosedSet)
DECLARE db_cursor CURSOR FOR
SELECT neigh.nodeId , NodeGeo FROM @neighbor_nodes
neigh inner join @Nodes nodes on neigh.nodeId = nodes.NodeId
OPEN db_cursor
FETCH NEXT FROM db_cursor INTO @curNodeId, @curGeo
WHILE @@FETCH_STATUS = 0
BEGIN
    set @tentative_g_score = (select
    map.g_DistanceToInitial from @solutionMap map where
    map.NodeID= @x) + @xGeo.STDistance(@mission)
    set @tentative_is_better = 1
    insert into @solutionMap(NodeID)
    values(@curNodeId)
    insert into @solutionMap(NodeID)
    values(@curNodeId)
    end
    else if @tentative_g_score < (select
    map.g_DistanceToInitial from @solutionMap map where
    map.NodeID = @curNodeId)
    set @tentative_is_better = 1
    else
        set @tentative_is_better =0
        if(@tentative_is_better = 1)
        begin
            update @solutionMap set ParentNodeId = @x
            where NodeID = @curNodeId and (ParentNodeId is
            null or (@select f_TotalDistance from
            @solutionMap where NodeID= @x) < (select
            f_TotalDistance from @solutionMap where
            NodeID=(select ParentNodeId from @solutionMap
            where NodeID= @curNodeId))))
            update @solutionMap set
            g_DistanceToInitial =@tentative_g_score ,
            h_DistanceToGoal = @curGeo.STDistance(@mission),
            f_TotalDistance = @tentative_g_score +
            @curGeo.STDistance(@mission)
            where NodeID = @curNodeId
        end
        FETCH NEXT FROM db_cursor INTO @curNodeId, @curGeo
        END
        goto AStarLoop
        end
```

Fig. 2. A* algorithm using SQL Server Spatial
References


[12] Nancy Amato, “Potential Field Methods”, course notes, Fall 04, Univ. of Padova.


Fig. 3. Displaying Path Planning and conflict detection results using 3D Google Earth