Chapter X

**Experimental Study**

To test and support our theoretical approach to compute spatial alarms in the obstructed space to be efficient, we have done extensive experiments using both real and synthetic data-sets. In this chapter, we are going to present the validation and comparison of the proposed approach against the naïve approach regarding various effective parameters.

**X.1 Experiment Setup**

In this section, the detailed setup of the experiments is described as per the following sub-sections.

X.1.1 Data Set Used

We have used both real and synthetic data-sets to evaluate our solution. In case of real data-sets, we have used obstacles and point of interests (POIs) of Germany. The obstacle set has 30674 minimum bounded rectangles (MBRs) of railway lines (rrlines). In our experiment, we assume obstacles to be presented by MBRs, but our algorithm can handle any type of obstacles. The POI set has 76999 MBRs of hypsography data (hypsogr). We assume data-points to be endpoints of the hypsography data. In this way, the POIs and obstacles are in the same plane which allows us to simulate a real-life scenario. We do not allow intersections between POIs and obstacles, neither do we allow duplicate POIs or obstacles. In case of synthetic data-sets, we have generated the obstacles and POIs from the real datasets following a uniform distribution. Before using in the real experiment, we have always normalized both real and synthetic dataset in a 10*,*000 *∗* 10,000 grid.  
In our experiments, we have assumed only one type of POIs. However, our approach also can handle multiple types of POI.

Prior to the run of the main algorithms, two separate R-trees are built to store the POIs and the obstacles respectively from the used data-sets.

|  |  |  |
| --- | --- | --- |
| Parameter | Values | Default |
| Client’s Velocity(v) Range (km/hour) | 0~40, 0~100, 40~120 | 0~40 |
| Alarm Range, r (meter) | 40, 60, 100, 150, 200 | 150 |
| Synthetic data-set size |  |  |

X.1.2 Sample Query Generation

We have simulated the movement of the client randomly in the runtime. During the experiments, we have assumed that any movement-path of the client can be synthesized as piecewise-linear, i.e., a set of directed straight lines. In the explicit form of any straight line, y=mx+c, for any client position (x, y), we have randomized the slope m giving some value of c each time. The direction of the client along this new straight line is also randomized to be either in forward or backward along the path, with a bias towards the forward direction, as the real world user-movement with a definite source and destination usually proposes.

After determining the client’s new position along this path, in the naïve approach, the algorithm 1 directly queries the server for POIs and obstacles within the client’s alarming range. Alternately, in the main approach, the algorithm 6 checks the region-crosses and queries the server if necessary.

The client’s velocity is also randomized within a certain range to give a new position of the client along the current direction in the next iteration, which again generates a new server-query accordingly.

Meanwhile, the direction of the client is predicted in the main approach from the latest set of piecewise linear paths in the client’s movement history. This prediction procedure has already been described in the section <Algorithm>.

X.1.3 Measurement of the performance parameters

In our experiments, we vary the following query parameters:

(i) the velocity range of the client,

(ii) the alarm range, r

(iii) the size of data sets (synthetic data set).

Table 1.1 summarizes the parameter values used in our experiments. In all experiments, we estimate I/O accesses and the query processing time to measure the efficiency of our algorithms. In each set of experiments, we run the experiment for 100 queries and present the average result.

X.1.4 Implementation language and tool

The project of our experiment is implemented using C++ language and have been compiled, debugged and tested using Microsoft Visual Studio 2015 Enterprise edition with a full version student-license from DreamSpark.

X.1.5 PC configuration

We have run our experiments in three PCs of the following configurations:

1. Intel Core i5 2.9 GHz (Quad Core), 12 GB RAM

2. Intel Core i5 2.3 GHz (Quad Core), 4 GB RAM

3. AMD FX 6100 3.3 GHz (Hexa Core), 8 GB RAM

The average of these multiple runs is taken to measure and compare the performance of both the naïve approach and our approach.

**X.2 Effect of the Parameters**

(In each of these sections, you have to describe graph. In addition to describing the trend of the curve and doing comparative analysis, try to go into more deep analysis like for varying from a to b it increase is x%, the rate of decrease increases with the increase of ... etc.

For every trend and pattern you have to argue on why it is happening like this.)

**X.3 Summary of the Experiments**

(At the end write a summary paragraph of your experimental result that should include the following information:

Give some statistics in experimental results like our approach is on average x times faster than the naive one in terms of (e.g., computational time)

)

* 3 graphs: cumulative run-time, I/O access, #server queries

X. 3.1 Client’s Velocity Range:

(bar-chart of 3 ranges’(x-axis) for 2 algorithms(bars))

X.3.2 Alarm Range

(Computation )