Moppy – Mobile Object Position Prediction Application

Łukasz Rosikiewicz
Logos sp. z o.o.
ul. Rogalińskiego 16
60-267, Poznań, Poland
Email: Lukasz.Rosikiewicz@logos.com.pl

Mikołaj Morzy
Poznan University of Technology
ul. Piotrowo 2
60-965 Poznan, Poland
Email: Mikolaj.Morzy@put.poznan.pl

***Abstract*—In this paper we present Moppy — an application for mobile object position prediction. Moppy is an easy to use, intuitive application with interactive graphical user interface and result visualization. The main purpose of Moppy is to predict an unknown position of a mobile object based on frequent trajectories discovered from historical movement data. In this paper we present an overview of Moppy's features, including input data formats, mining algorithms employed by Moppy, and visualization of prediction results. Some research issues pertaining to the development of Moppy are also briefly discussed in this paper.**

I. Introduction

Mobile devices capable of wireless communication are becoming ubiquitous in our everyday lives. Combined with recent advances in positioning technology, many location-based services are becoming available for casual end-users of cellular phones, personal digital assistants, or vehicles equipped with positioning sensors and tracking systems. Most systems record positions of all tracked mobile objects, thus enabling to construct paths of moving objects. Due to several reasons (e.g., power loss or communication failure), an exact location of a moving object can remain unknown for a certain period of time. Hence, a method is required to predict a possible position of a moving object in case the exact position of the object is not known. There are numerous applications of position prediction, e.g., context and location-based advertisement, traffic management, way-finding, etc.

Predicting the location of a moving object can be a difficult task. The amount of data to be analyzed precludes most traditional prediction methods, e.g., methods developed within machine learning community. Another limitation is the requirement for the prediction algorithm to work online, producing position predictions instantly. This requirement severely limits the amount of computations allowed for the prediction algorithm. Furthermore, the performance of the prediction method should remain independent (or almost independent) of the number of moving objects monitored by the positioning system. We also expect that the prediction module produces a "white box" prediction model where each prediction is somehow "explained" to the user, e.g., in the form of human-readable rules. Given that the model does not drop below a certain accuracy threshold, we strongly prefer a model that predicts faster, probably at the expense of some accuracy. This assumption is driven by practical requirements of location-based services, where the prediction speed is of crucial importance. Many solutions proposed in the literature so far do not fulfill this requirement. As an example, consider simulation which is widely used for location prediction. Simulation is capable of producing fairly accurate results, but requires sophisticated models of moving objects interactions and movement environment (area topology, routing schemes, unexpected barriers, etc.). Such complex models may be computationally too expensive for successful deployment in mobile environment. The second assumption driving our research and development efforts is the necessity to use historical data of past object movements for prediction. The raw data collected from moving objects contains knowledge about typical behavior of moving objects. This knowledge can be used to predict future positions of a moving object, assuming that the object adheres to typical movement patterns (and most objects do).

In this paper we demonstrate Moppy, a new tool for predicting positions of moving objects in urban environments. Moppy employs data mining techniques to historical data of moving objects to discover frequent movement patterns expressed as frequent trajectories, following the ideas expressed in [1] and [4]. Based on discovered patterns Moppy predicts the probable position of a moving object, for which a partial movement path is known. Frequent patterns discovered in historical data are matched against the known part of a moving object's trajectory in order to predict the location of the mobile object.

The paper is organized as follows. Section II describes the general idea behind Moppy. In Section III we present the architecture of the application, we enumerate its main features, and we describe Moppy's interface. Finally, Section IV presents a case study of Moppy's demonstration.

II. General Idea

Figure 1 presents the general idea of mobile object position prediction using Moppy. Moving object positions are measured using wireless positioning technology and recorded in the database. Unfortunately, raw position data are expressed in terms of continuous domains of geographical coordinates. In order to mine raw data it must undergo the discretization process. Moppy imposes a user-defined grid of rectangular cells

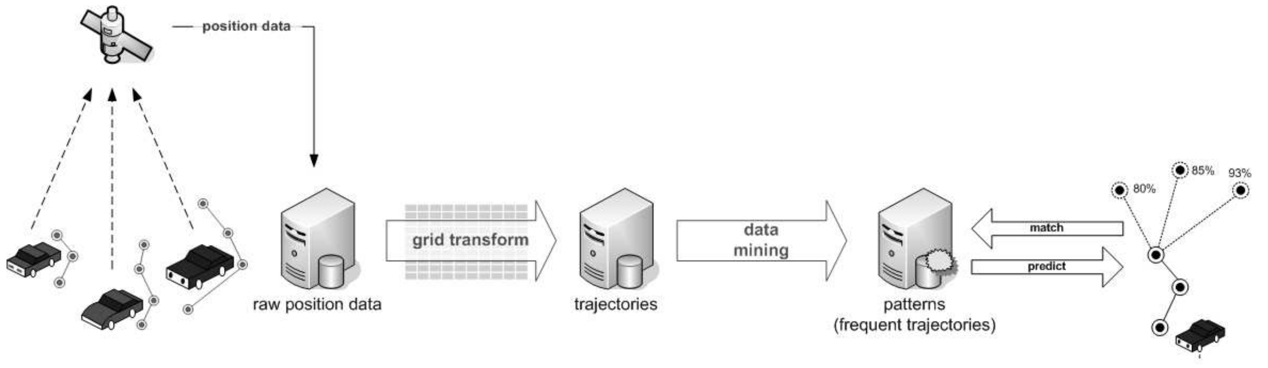

Fig. 1. Position prediction using Moppy

on the movement area. Moving object paths and positions are transformed, using this grid, into trajectories. Each trajectory is an ordered list of grid edges traversed by a moving object path. Thus, a moving object path expressed using continuous coordinates is transformed into a sequence of discrete edges. Next, a data mining algorithm is applied to trajectory data to find frequent trajectories that describe movement patterns exhibited by a significant number of moving objects. These patterns (frequent trajectories) are stored in the database using an optimized physical structure of the *MFP-Tree*, which is a modified *FP-Tree* [5] capable of storing sequences. The above mentioned steps are computationally expensive, but can be performed periodically and off-line.

The prediction of a mobile object position is performed on-line. Figure 2 presents an overview of the prediction process. For a given vehicle, let us suppose that a partial path of the vehicle is known. Suddenly, the tracking device of the vehicle stops sending positioning signal. In order to predict the possible location of the vehicle, the known path of the vehicle, referred to as the query path, is transformed into the query trajectory. Frequent trajectories are read off the MFP-Tree structure and matched against the query trajectory using a matching algorithm. All matching algorithms are implemented as plugins and can be freely replaced. The matching algorithm returns the set of possible positions of the moving vehicle. Furthermore, each prediction has a probability estimate associated with it. The results are visualized in Moppy.

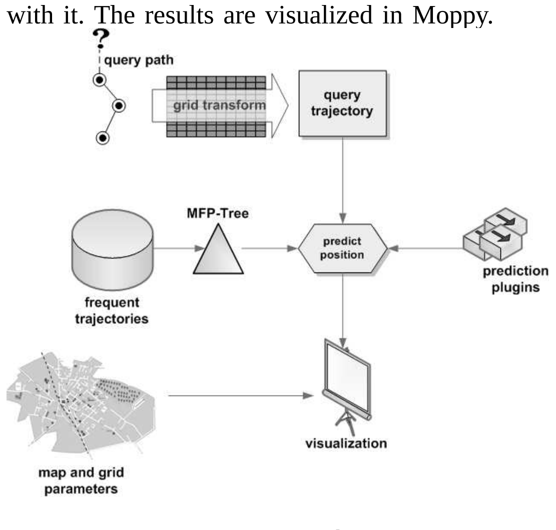

Fig. 2. Prediction process

III. Architecture, Features, and User Interface

Moppy's architecture follows the modular design pattern, which results in a flexible and extensible application. The use of plugin modules allows to easily extend the basic framework with new solutions. Input and import modules can be added to the application to acquire data from various sources. Position prediction algorithms are also pluggable, thus enabling to test new algorithms using Moppy's transformation and discovery engines. Most programming is done against interfaces, modularizing the internal architecture of the application. Moppy is written in C# and runnable on Microsoft .NET 2.0 Framework, which greatly attributes to overall efficiency and speed of execution. Below we briefly describe the most important modules. Moppy consists of the following modules:

- Import module – this module allows loading map files stored in `*.node` and `*.edge` formats [2], as well as using mobile objects paths stored in `*.mpf` and `*.mof` file formats. It is possible to transform TIGER/Lines files and Shape files into format used by the application with external programs.
- Transformation module – this module is responsible for the transformation of mobile object paths acquired from raw data into trajectories expressed in terms of grid cells and edges according to user-defined parameters. The transformation step can be perceived as a discretization of the continuous domain of moving object positions into a discrete domain of grid edges. Simplification resulting from this step allows us to employ an efficient data mining algorithm for the discovery of frequent trajectories. As the result of this step mobile object paths stored in `*.mpf` or `*.mof` formats are transformed into the trajectory format. Obviously, the discretization results in a loss of precision in the measurement of mobile object positions. The transformation process is governed by grid parameters supplied by the user. If the grid is too coarse, then the discovered patterns might be too general to accurately describe individual moving objects. On the other hand, if the grid is too dense, the patterns might not be apparent in trajectory data. In our experiments we have used a 100×100 grid covering the city of Oldenburg.
- Discovery engine – the purpose of this module is to employ data mining algorithm to the transformed trajectory data. Moppy implements of *Traj-PrefixSpan* and *Traj-PrefixSpan Neighbour* algorithms [6] that are modifica-

tions of the well-known *PrefixSpan* algorithm [7]. Thorough description of both algorithms and accompanying physical structures is beyond the scope of this paper. Discovery engine finds frequent trajectories of mobile objects that are further used by the prediction module.

- Prediction module – this module uses plugins that implement algorithms of position prediction based on frequent trajectories found by the discovery engine. The use of plugins allows the user to test a number of different prediction algorithms based on frequent trajectories. Users can freely write and use new plugins that implement different matching strategies. Currently, Moppy ships with three plugins: *Whole Matcher*, *Last Matcher* and *Longest Last Matcher*. Moppy produces a probabilistic prediction model, where each prediction has an accuracy estimate associated with it. Depending on map characteristics, parameters of the transformation grid, and available historical data it is possible to achieve prediction quality rate of over 90%. Prediction algorithms are very fast, the mean prediction time for a queried mobile object is less than one second [6].
- Graphical user interface (GUI) with visualization of prediction results.

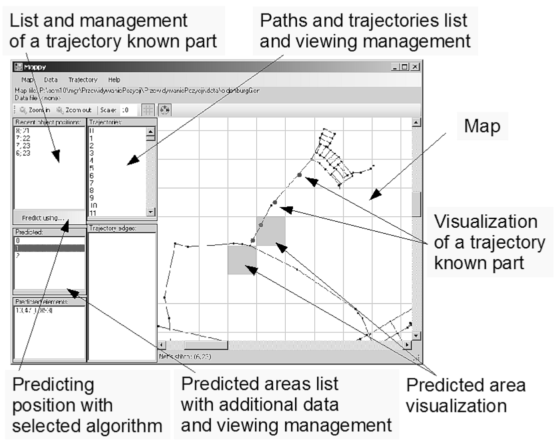
List and management of a trajectory known part
Paths and trajectories list and viewing management
Map
Visualization of a trajectory known part
Predicting position with selected algorithm
Predicted areas list with additional data and viewing management
Predicted area visualization

Fig. 3. Moppy GUI

Moppy is a highly usable and intuitive desktop application. It has a graphical user interface (GUI) for performing all operations. A screenshot of Moppy is presented in Figure 3. Moppy allows to load a map, set grid parameters and visualize a grid on top of the map, investigate movement paths loaded from a file, transform movement paths into trajectories, investigate trajectories and frequent trajectories. The map of the movement area can be easily zoomed in and out. After loading movement object data and setting discretization grid parameters, data mining algorithm for discovering frequent trajectories can be run. Discovered patterns can be displayed on the map or can be exported to a disk file for future use. Moppy allows users to quickly verify discovered patterns by matching patterns against a user-defined query path. A user can construct a query path by simply double-clicking on a map to indicate subsequent positions of a moving object. The results of the position prediction are displayed as a list of possible locations of the moving object, ordered by prediction accuracy. Furthermore, possible locations of a moving object are indicated on the map using colored rectangles. The color of the rectangle informs about the probability of the prediction in this area making the interpretation of results more intuitive.

Moppy can be run either in graphical mode or in console mode. Running Moppy in console mode is useful when the user needs to perform a number of operations using a batch script. Console mode offers similar functionality to the graphical mode. Loading, transformation, and the discovery of frequent trajectories can be run from a script. Console mode also allows to predict the location of a moving object given the query movement path. This feature allows to incorporate Moppy as a prediction service into larger applications.

IV. Demonstration

The outline of the demonstration of Moppy is the following. We will present an overview of the architecture of Moppy and we will briefly discuss the process of frequent trajectories discovery. Next, we will demonstrate the position prediction using Moppy, with the emphasis on the usability of different plugins. We will present application's graphical user interface: viewing maps, movement paths, movement trajectories and discovered frequent trajectories. Using an example map of Oldenburg we will demonstrate a few test cases of how to predict the position of a mobile object and how to work with the results in Moppy's user interface. We will also show how using different matching algorithms influences the quality of received results. Our experiments are conducted on synthetic data sets created using Network-based Generator of Moving Objects [3]. For more information please visit Moppy's homepage, which is located at http://www.icpnet.pl/~rosa/moppy/. The page contains additional screenshots of the application and the documentation of the project. Moppy is freely available for download (Microsoft .NET 2.0 Framework required) both as binary and source code. The authors welcome any questions that may arise and will be happy to provide additional explanations.

References

[1] S. Brakatsoulas, D. Pfoser, and N. Tryfona. Modeling, storing, and mining moving object databases. In *IDEAS*, pages 68–77. IEEE Computer Society, 2004.

[2] T. Brinkhoff. Description of the format of the network files. http://www.fh-oow.de/institute/iapg/personen/brinkhoff, 2000.

[3] T. Brinkhoff. Network-based generator of moving objects. http://www.fhoow.de/institute/iapg/personen/brinkhoff/generator/, 2005.

[4] G. Gidófalvi and T. B. Pedersen. Spatio-temporal rule mining: Issues and techniques. In A. M. Tjoa and J. Trujillo, editors, *DaWaK*, volume 3589 of *Lecture Notes in Computer Science*, pages 275–284. Springer, 2005.

[5] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *SIGMOD Conference*, pages 1–12. ACM, 2000.

[6] M. Morzy and Łukasz Rosikiewicz. Mining frequent trajectories of moving objects for location prediction. Technical Report RA-01/07, Poznan University of Technology, Institute of Computing Science, 2007.

[7] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *ICDE*, pages 215–224. IEEE Computer Society, 2001.