A Training-by-Example Approach for Symbol Spotting from Raster Maps

Yao-Yi Chiang, Phokgoan Chioh, Sima Moghaddam

University of Southern California, Spatial Sciences Institute, 3616 Trousdale Parkway, AHF B55
Los Angeles, CA 90089-0374
Email: {yaoyic; chioh; khashkha}@usc.edu

1. Introduction

Graphic symbols in maps depict important and interesting geographic phenomena, such as wetlands (Figure 1). The descriptive metadata of these symbols can be found in map labels or keys; however, labels are only capable of displaying limited information (e.g., place names) and keys provide categorical information. For example, Figure 2 shows a group of unique buildings in a U.S. Geological Survey (USGS) topographic map but the map does not provide any information about these buildings (e.g., names). Figure 3 shows a scanned map of Baghdad, Iraq where most symbols are labeled with place names but retrieving and integrating further information (e.g., addresses) of these places from other sources requires additional efforts such as using the place names and locations to search on Wikipedia or DBpedia (a structured version of Wikipedia).

In this paper, we present a training-by-example approach for spotting graphic symbols in raster maps. We demonstrate that our approach efficiently enables automatic linkages between DBpedia records and locations in a map. Traditional document analysis techniques for spotting map symbols generally require a large amount of training datasets, the presence of map keys (e.g., Samet and Soffer, 1998), or ad-hoc preprocessing steps (e.g., image thresholding) (Chiang et al, 2014; Lladós et al., 2002). In contrast, our approach takes only one user-selected symbol example to extract the locations of all symbols that have similar graphical appearance to the example.

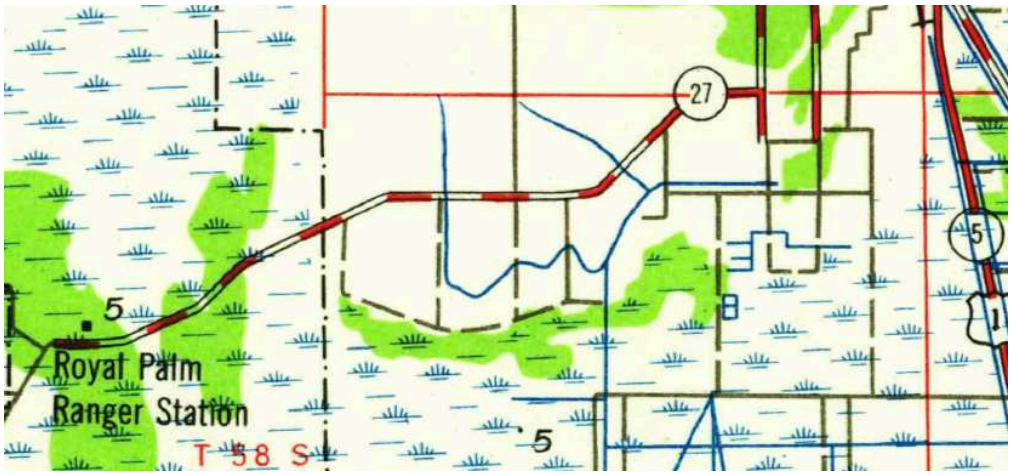


Figure 1: Wetlands in a historical USGS topographic map (Miami, Florida, circa 1958).

Figure 2: Buildings of the Park La Brea Apartment in a historical USGS topographic map (Hollywood, California, circa 1953) (left) and Google Earth imagery (right).

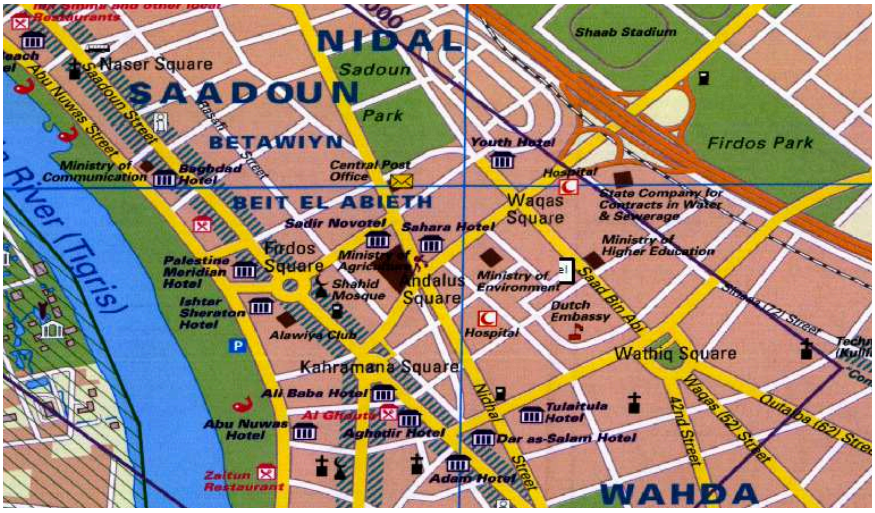


Figure 3: Symbols labeled with place names in a scanned Baghdad map.

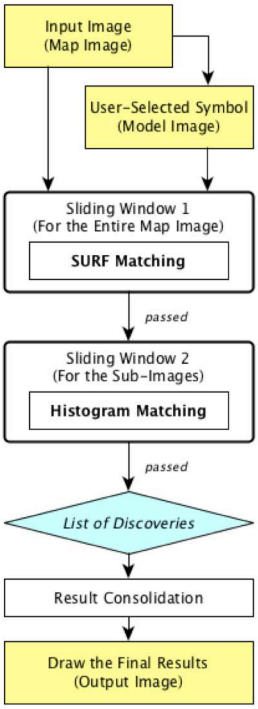


Figure 4: The SymbolRecognizer framework.

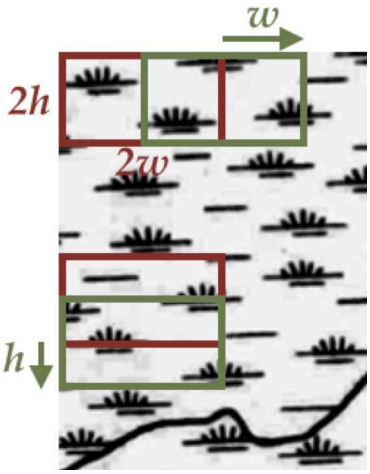


Figure 6: The SURF matching sliding window.

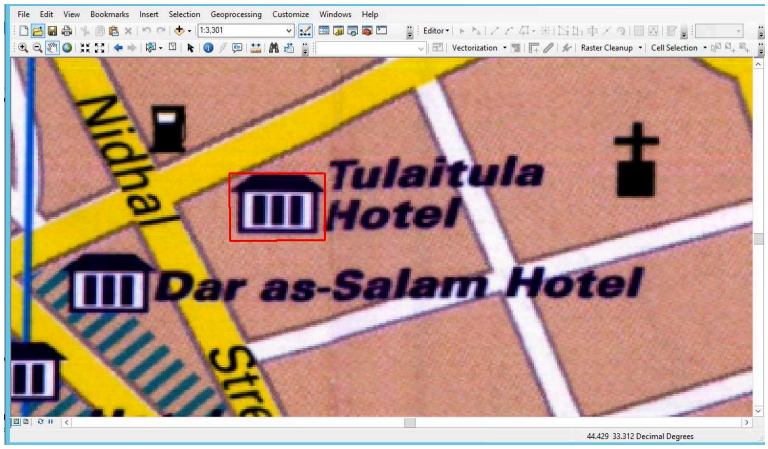


Figure 5: A user-selected symbol example.

2. Symbol Spotting

This section presents our symbol spotting approach called SymbolRecognizer (Figure 4). A model image is an image that covers a user-selected example in the input map (the red rectangle in Figure 5). The recognition task is to search the map for symbols that matches the model (i.e., target symbols). SymbolRecognizer utilizes a two-phase process: (1) Using the SURF (Speeded Up Robust Features) matching (Lowe, 1999; Bay et al., 2006) to efficiently identify the local regions (sub-images) where a target symbol might present and (2) Using pixel intensity distribution (with histogram matching) to verify the presence of a target symbol in each sub-image.

2.2 SURF (Speeded Up Robust Features) Matching

Considering a model image with width and height equal to *w* and *h* pixels, in the first phase, SymbolRecognizer uses a sliding window of the size equal to *2w* and *2h* pixels and moves *w* or *h* pixels in the horizontal or vertical direction to scan through the entire input map (Figure 6). The size of the sliding window guarantees that every target symbol is covered completely in at least one window (a sub-image). At each position of the sliding window, SymbolRecognizer detects the SURF features from the sub-image and compares the detected features with the SURF features of the model image. If the comparison result contains a high number of matched features, the sub-image is highly likely to contain a target symbol (see Lowe (1999) for details of this object recognition procedure) and is passed to the next phase.

2.2 Histogram Matching

The SURF matching is efficient and widely used to recognize real world objects in photography or videos, but map symbols have simpler shapes (than real world objects) and are relatively small, which can cause frequent false positives in the matching results. Therefore, SymbolRecognizer compares the pixel intensity distributions of the model image and each sub-image that passes the SURF matching to determine whether or not a target symbol presents and to extract the symbol location.

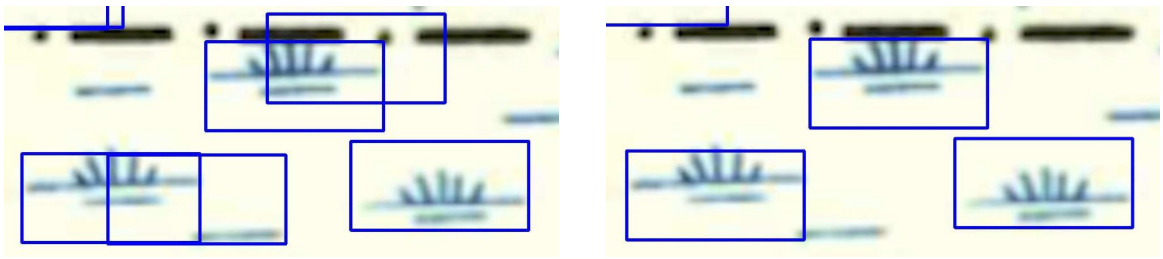
For each sub-image that passes the first phase, SymbolRecognizer uses the model image to scan from the top-left corner and moves *one* pixel in the horizontal or vertical directions (i.e., Sliding Window 2 in Figure 4). Each scanning position records a similarity score calculated using the correlation of the grayscale histogram of the model image (H^{model}) and the grayscale histogram of the overlapping image patch (the overlapping area between the model image and the sub-image) (H^{patch}). The correlation is defined as follows:

$$Similarity\ Score\ = \frac{\sum_{i=0}^{255}(H_i^{model} - \overline{H^{model}})(H_i^{patch} - \overline{H^{patch}})}{\sqrt{\sum_{i=0}^{255}(H_i^{model} - \overline{H^{model}})^2 \sum_{i=0}^{255}(H_i^{patch} - \overline{H^{patch}})^2}}$$

SymbolRecognizer uses an empirically set threshold of 90% on the similarity score to filter out the sub-images that do not contain a target symbol and to locate the symbol location. If none of the scanning positions in a sub-image has a similarity score higher than 90%, the sub-image does not contain a target symbol; otherwise, the scanning position that has the highest similarity score (in a sub-image) is the detected location of a target symbol.

2.3 Result Consolidation

A target symbol can be detected in overlapping sub-images during the SURF matching since the sliding window can cover a symbol more than once (Figure 7(a)). To consolidate the results, if overlapping sub-images contain multiple target symbols, SymbolRecognizer keeps only the target symbol with the highest histogram matching score (Figure 7(b)).



(a) Before consolidation. (b) After consolidation.

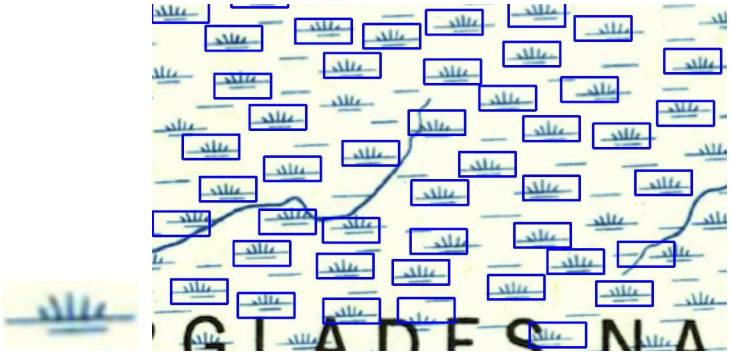
Figure 7: Result consolidation for overlapping sub-images.

3. Preliminary Results and Discussion

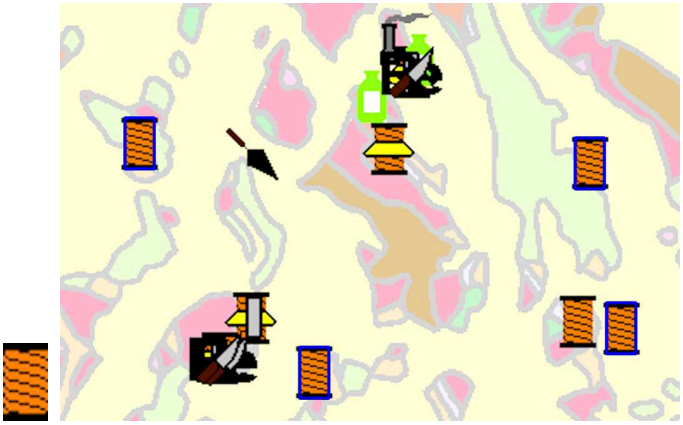
We implemented SymbolRecognizer in our map processing system, Strabo, as an Esri ArcMap plugin and tested the plugin with maps from four sources (Figure 3, Figure 8, and Table 1). For each test map, the user selected one sample symbol and Strabo automatically processed the sample to find other symbols in the map.

The results showed promising extraction precision (with only a few false positives). The USGS Hollywood map had the lowest extraction precision since the target symbols (the Park La Brea apartment buildings) are in different orientations. Although the SURF matching is rotation invariant, the histogram matching results could be compromised if the image patch did not cover the entire symbol of different orientations in the sub-image. All other test maps that contain symbols in the same orientation had more than 97% extraction precision.

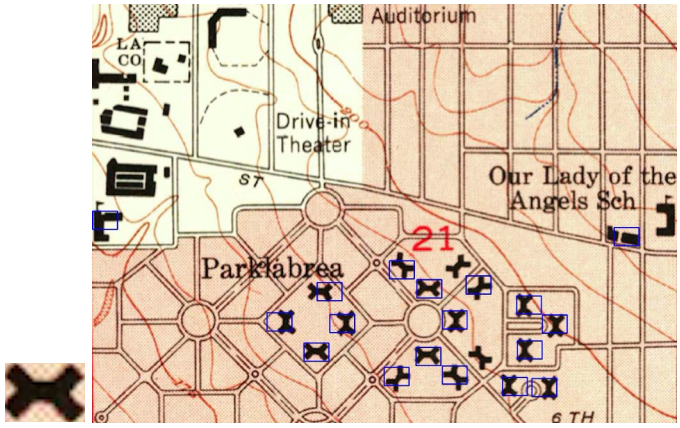
Considering the extraction recall, significantly overlapped features were the main cause of true negatives. Figure 9 shows two examples of overlapping symbols in the USGS Mine and Mineral map. The overlapping symbol to the right was detected because only a small portion of the symbol was overlapped by another symbol. The symbol to the left (Figure 9) was not detected since the entire symbol was almost covered by other symbols. The USGS Mine and Mineral map contains 12 (out of 25) significantly overlapped symbols and hence the extraction recall was the lowest among the test maps. All other test maps had more than 83% extraction recall.



(a) A historical USGS topographic map (Miami, Florida, 1958).



(b) The USGS Mine and Mineral Processing Plant Locations map.



(c) A historical USGS topographic map (Hollywood, California, circa 1953).

Figure 8: Model images (left) and sample results where blue rectangles are the recognized locations (right).

Table 1. Recognition Results.

Source	Image Size (pixels)	# of Target Symbols	Precision	Recall
USGS Miami (1958)	409x438	87	97.33%	83.91%
USGS Mine and Mineral	2465x2150	25	100%	48%
USGS Hollywood (1953)	554x396	18	88.89%	88.89%
Gecko Maps, Baghdad	5104x2616	17	100%	88.23%

Figure 9: Examples of overlapping symbols.

Figure 10 shows the Baghdad map with the identified symbols linked with DBpedia URIs. Once the symbols were identified, Strabo queried the DBpedia SPARQL endpoint to retrieve the nearest DBpedia entries to individual symbol locations. These entries had various DBpedia types such as Museum, Embassy, School, and Hotel. Since the identified symbols represented places in the same category, Strabo first detected the most popular category among the retrieved entries and only linked a symbol to DBpedia if the closest entry of the symbol was in the popular category. In this test area, the most popular category is Hotel and there were only four hotel entries on DBpedia.

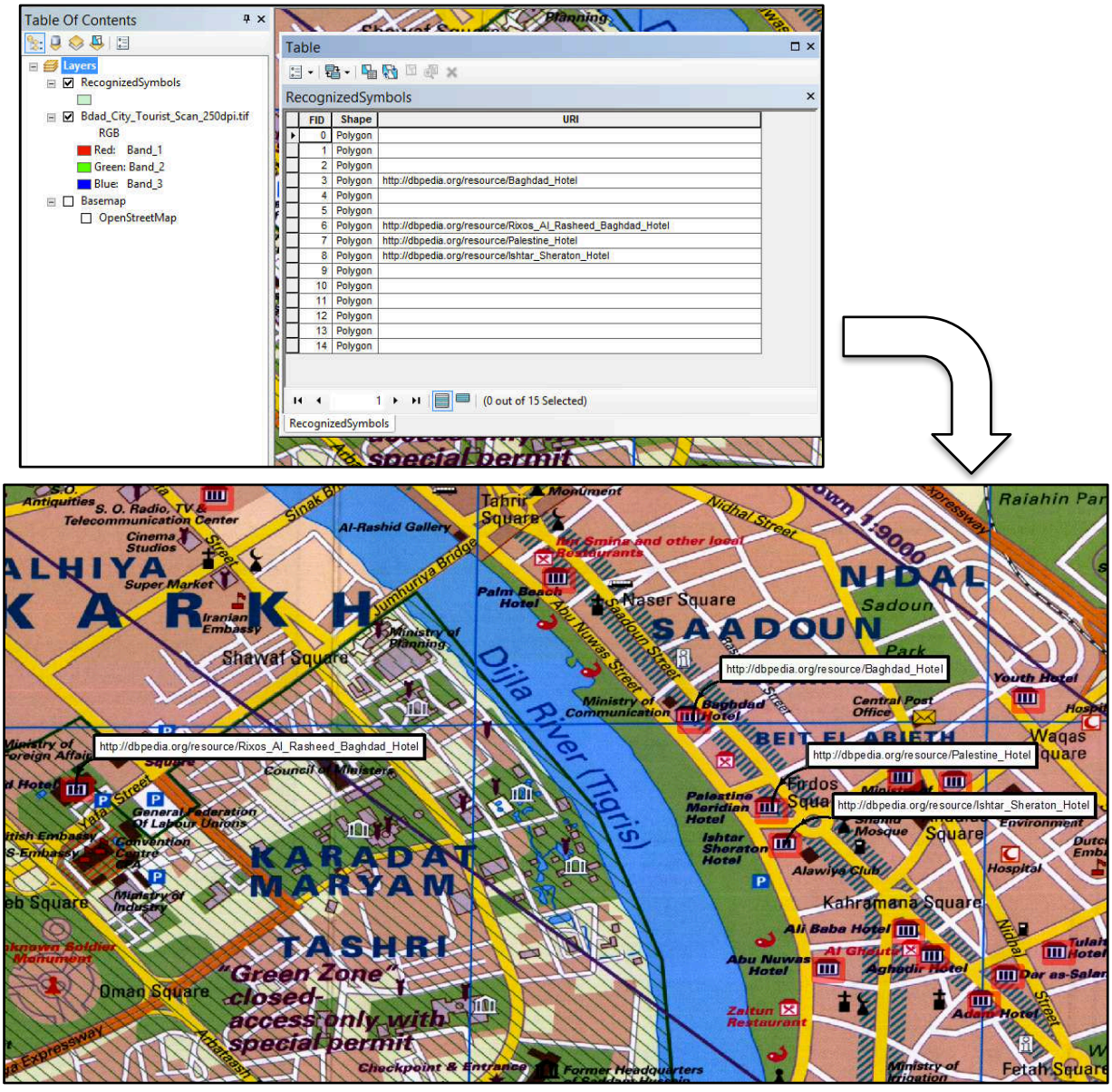


Figure 10: Automatic linkages between map locations and DBpedia records.

4. Summary and Outlook

We presented a training-by-example approach for symbol spotting from raster maps. Our approach requires very little user effort and can handle various types of maps and symbols. We plan to test on more symbol types and further investigate automatic methods to link the extracted symbol locations to other sources.

References

Bay, H., Tuytelaars, T., and Gool, L. V., 2006, SURF: Speeded up robust features. In the Proceedings of the 9th *ECCV*, pages 404–417.

Chiang, Y.-Y., Leyk, S., and Knoblock, C. A., 2014, A survey of digital map processing techniques. *ACM Computing Surveys*. doi: 10.1145/2557423, in press.

Lladós, J., Valveny, E., Sánchez, G., Martí, E., 2002, Symbol recognition: Current advances and perspectives. In *GREC*, pages 104–127.

Lowe, D. G., 1999, Object recognition from local scale-invariant features. In ICCV, vol. 2, pages 1150–1157,

Samet, H. and Soffer, A., 1998, Magellan: Map acquisition of geographic labels by legend analysis. *IJDAR*, 1(2): 89–101.

Location Prediction With Sparse GPS Data

Ayush Jaiswal[1], Yao-Yi Chiang[2], Craig A. Knoblock[3], Liang Lan[4]

[1]National Institute of Technology Calicut, India
Email: ayush_bcs10@nitc.ac.in

[2]Spatial Sciences Institute, University of Southern California, USA
Email: yaoyic@usc.edu

[3]Department of Computer Science and Information Sciences Institute, University of Southern California, USA
Email: knoblock@isi.edu

[4]Noah's Ark Lab, Huawei Technologies
Email: lan.liang@huawei.com

1. Introduction

Predicting the next location of a user from their movement history is useful in building intelligent applications that can continuously assist users without explicit user-input. Data collected by applications on consumer-grade mobile devices, such as GPS data, can have missing records (e.g., due to the application crashing) and the sensor sampling frequency needs to be kept low so that it does not drain the mobile battery. Thus, there can be a significant time gap between each pair of recordings. Our work in this paper focuses on predicting the next location of a mobile user using such sparse GPS data, collected at a very low frequency of once every 10 minutes. To give an example of dense data, Krumm and Horvitz (2005, 2006) use data collected once in every six seconds.

Figure 1: Movement patterns may be disjoint. The blue and the red points were recorded on two different days.

Sparseness in GPS data makes finding patterns in a user's movement history difficult. Moreover, the low sampling rate might capture movement patterns that are along the same path but are disjoint (Figure 1). Losses in GPS connection and imperfect behavior of the data collection application further increase the sparseness of the data. We tackle the problem of sparseness by interpolating user movements using a routing service.

Location prediction can be viewed as a classification problem in which the possible next locations are discrete classes, but GPS data is continuous in nature. Hence, we use a grid over the region where the data is centered, and map the points to grid-blocks. Another possible method of location abstraction is mapping points to the nearest mapped addresses according to maps such as Baidu Maps and OpenStreetMap. This is known as reverse geocoding. This

approach depends significantly on the accuracy and the amount of address information available for the region where the data is collected. With insufficient address information, such as in our case, using reverse geocoding results in a lot of repetition in location-IDs as many points map to a single location-ID. This leads to loss of movement information.

We also discuss the results of using four different Markov models for the prediction task on the sparse and the processed data.

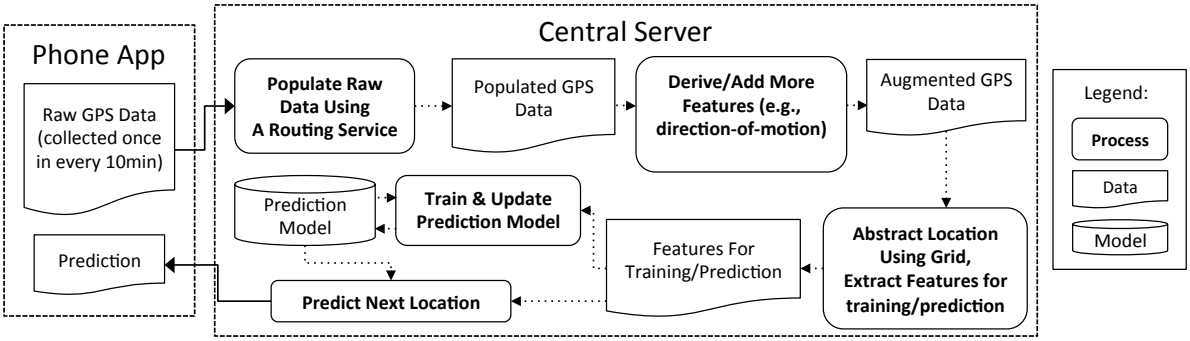


Figure 2: Location Prediction System for Sparse GPS Data.

2. Next Location Prediction

Figure 2 shows the overall workflow of our approach. The sparse GPS data is populated using a routing service to produce a dense set of user movement history, additional features (such as direction-of-motion, described later) are added, and the points are abstracted to locations using a grid. The resulting features are given as inputs to the prediction model.

2.1 Dealing with Sparseness

Our approach uses a routing service to find the shortest path between every consecutive pair of points and uses the route returned to fill the gap between the pair with dummy points. The underlying assumption is that people tend to take the shortest path between any two places that are near one another, especially when they are separated by just 10 minutes in time.

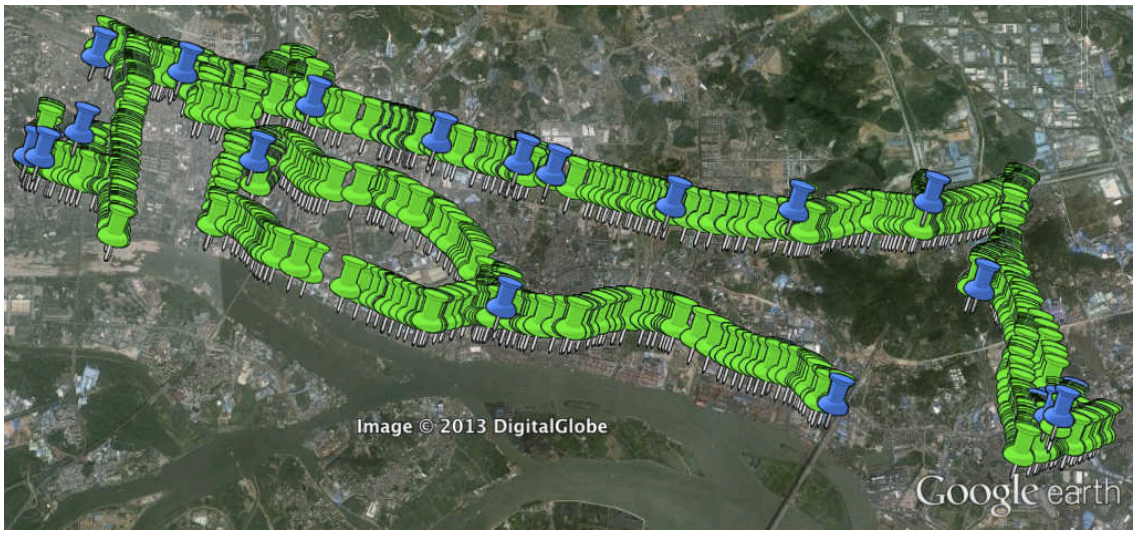


Figure 3: The blue points are original points in the data while the green ones were added using the routing service.

For example, Figure 3 shows how our system populated some of the data that we work on. The interpolated points filled in using the routing service complete the original path very

elegantly. We use the Google Directions API[1] to get the shortest driving route between consecutive pairs of points.

2.2 Features and Prediction Models

We use Markov models to predict the next grid-block the user will be in, as illustrated in Figure 4. Markov models help in describing sequences of events and their associated probabilities. Cheng et al. (2003) explain how Markov models can be used for location prediction. We employ four different Markov models to test four hypotheses for location prediction from sparse GPS data:

- order-1 Markov model (O1MM): predict the next location of the user based on *their last known location*
- order-2 Markov model (O2MM): predict the next location based on *their two last known locations*
- order-2 Markov model with fallback on order-1 Markov model (FMM): try predicting with O2MM, and when it fails to make a prediction, use O1MM
- order-1 Markov model with direction-of-motion feature (O1MMD): we use the direction-of-motion between every consecutive pair of points. The directions that we employ are: North, North-East, East, South-East, South, South-West, West, North-West, and *stationary*. This feature removes the need of keeping track of multiple previous locations as it captures the information contained in them.

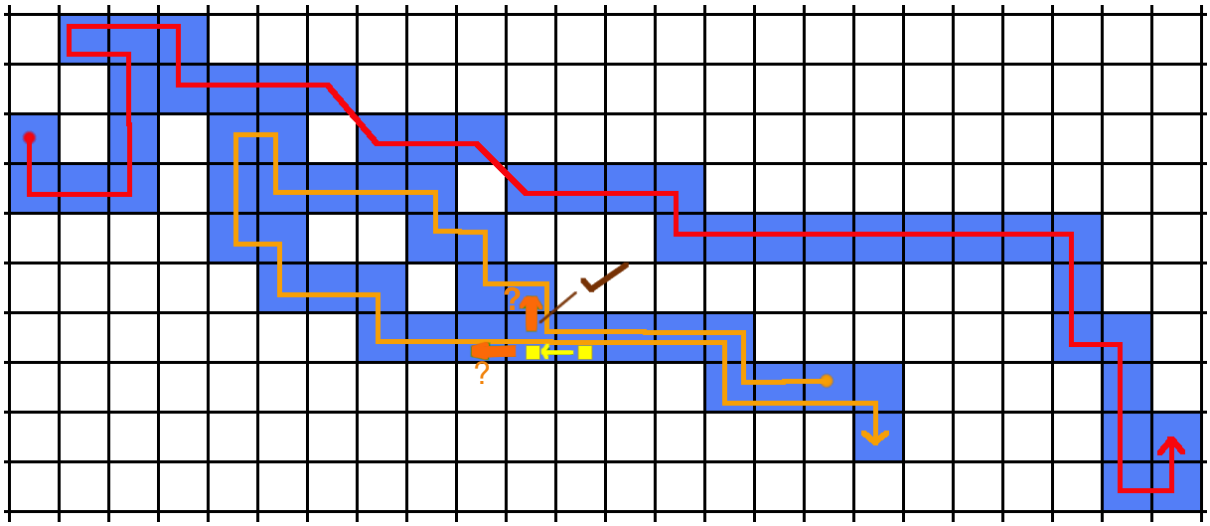


Figure 4: Predicting the next grid-block the user will be in. The model has learnt user movement patterns from day-1 (red line) and day-2 (orange line). On day-3 (yellow squares), it predicts the next location of the user in the upward direction (as learnt from the previous day).

3. Experiments and Results

Our data were collected by a user in Shenzhen, China over a 24 day period. On average, it has 14 GPS points in a day. We used the aforementioned Markov models for the task of location prediction on both the original data and the data resulting from the application of our processing steps. We calculated the average prediction accuracies using two experiment settings: the leave-one-day-out cross-validation setting (L1CV) uses the data from a particular day as test data and data from all other days as training data, and the sequential data

[1] http://developers.google.com/maps/documentation/directions

setting (SEQ) that uses data from a particular day as test data and data from only the days in the movement history before that day as training data. While cross-validation is a general approach to comparing the accuracies of machine learning models, SEQ is closer to how we would want the prediction to work in real world settings. A correct prediction is one that matches the next observed grid-block of the user. Our accuracy measure is the fraction of predictions that are correct.

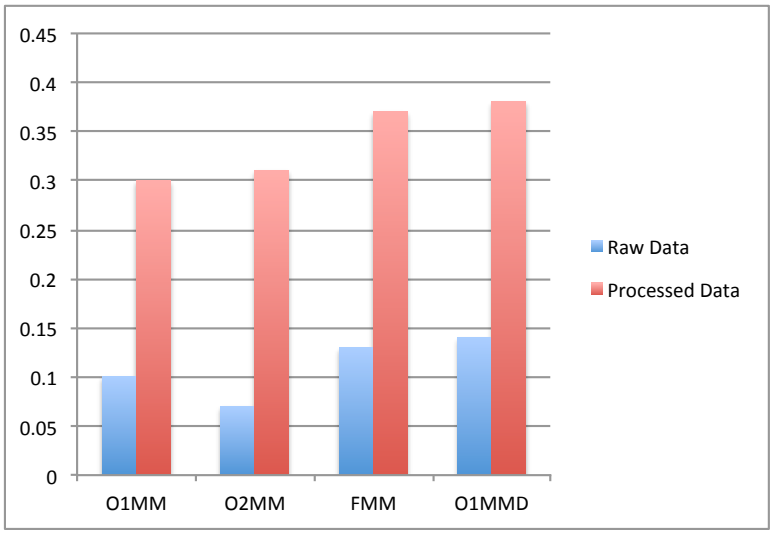


Figure 5: Average SEQ accuracy.

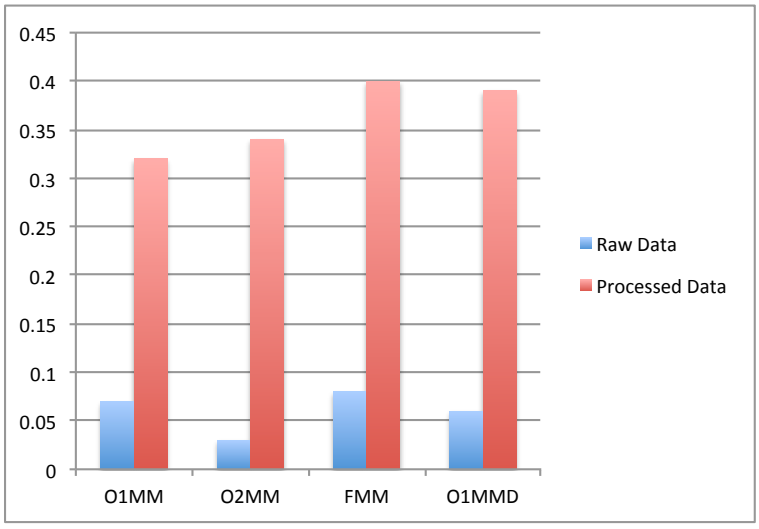


Figure 6: Average L1CV accuracy.

Figures 5 & 6 summarize our results. O1MMD and FMM performed almost equally well and better than the other models on the processed data. The desired order of accuracies should be O1MM ≤ O2MM ≤ FMM as the ones to the right make use of more information about the user's history, but we do not find this order in case of sparse data as O2MM could not learn many patterns because of the sparseness. In general, the prediction models were unable to learn patterns in the user's movements from the sparse GPS data. Solving the problem of sparseness improves their prediction accuracies. The overall accuracies appear low because of significant randomness in the movement patterns of the user whose data we used. It has been found that randomness in a user's movement patterns reduces the accuracy of prediction models (Anagnostopoulos et al. 2009). Such randomness is inevitable in the movements of real users.

4. Related Work

Krumm and Horvitz (2006) use grid-based location abstraction to predict the destination of the user from partial trajectories. Our work is different from theirs as we predict the user's next location, and our data is much more sparse than theirs. While their data is collected once every 6 seconds, ours is collected once every 10 minutes. Gao et al. (2012) report that Hierarchical Pitman-Yor language gives a higher accuracy as compared to Markov models. Anagnostopoulos et al. (2009) implemented location prediction using decision trees, k-nearest neighbor, and ensemble learning algorithms, and found that ensemble learning algorithms performed the best among them. The methods proposed in this past work cannot be applied directly to sparse data, such as ours, as the machine learning algorithms used in them will be unable to learn patterns effectively. Our processing steps interpolate the sparse data and improve location prediction on such data.

5. Discussion and Future Work

This paper presented an approach for location prediction using sparse user movement history. We showed that by exploiting an online routing service, we made location prediction possible on sparse movement data. We plan to build an intelligent method for automatically generating the dynamic grid size specific to a dataset and to incorporate other sensor data on mobile phones into the location prediction framework.

Acknowledgements

We thank the USC Viterbi School of Engineering for providing summer fellowship to Ayush Jaiswal, and we thank Huawei Technologies Co. Ltd. for a gift that supported the project and for providing the data.

References

Christine Cheng, Ravi Jain and Eric van den Berg, 2003, Location prediction algorithms for mobile wireless systems. *Wireless Internet Handbook, 245-263, CRC Press, Inc. Boca Raton, FL, USA*.

Huiji Gao, Jiliang Tang and Huan Liu, 2012, Mobile location prediction in spatio-temporal context. *Proceedings of the Mobile Data Challenge by Nokia Workshop in conjunction with the International Conference on Pervasive Computing, Newcastle, U. K*.

John Krumm and Eric Horvitz, 2005, The Microsoft Multiperson Location Survey. *Microsoft Research (MSR-TR-2005-103): Redmond, WA USA*.

John Krumm and Eric Horvitz, 2006, Predestination: Inferring Destinations from Partial Trajectories. Proceedings of the *8th International Conference on Pervasive and Ubiquitous Computing (UbiComp) 243-260*.

Theodoros Anagnostopoulos, Christos Anagnostopoulos, Stathes Hadjiefthymiades, Miltos Kyriakakos and Alexandros Kalousis, 2009, Predicting the Location of Mobile Users: A Machine Learning Approach. *ICPS '09 Proceedings of the 2009 International Conference on Pervasive Services, 65-72*.