Quick Map Matching Using Multi-Core CPUs

Renchu Song† Wei Lu† Weiwei Sun† Yan Huang‡ Chunan Chen†
†School of Computer Science, Fudan University, Shanghai, China
{10300240022, 10300240009, wwsun, chenchunan}@fudan.edu.cn
‡Computer Science and Engineering Department, University of North Texas, Denton, Texas, USA
huangyan@unt.edu

ABSTRACT
The ACM SIGSPATIAL Cup 2012 is about map matching, a problem of correctly matching a sequence of GPS sampling points to the roads on a digital map. This paper describes one of the winning submissions of the competition. The approach applies multi-threading technology to map matching in order to reduce running time and we propose an improvement to the Hidden Markov Model (HMM) map matching algorithm.

Categories and Subject Descriptors
H.2.8 [**Database Applications**]: Spatial Databases and GIS.

General Terms
Algorithm, Performance

Keywords
Map matching, ACM SIGSPATIAL Cup 2012, multi-core CPUs, Hidden Markov Model.

1. INTRODUCTION
Map matching is a problem of correctly matching a sequence of GPS sampling points to the roads on a digital map [5]. Specifically, a road network is defined as a graph $G(V, E)$. Each note $v \in V$ is described by a latitude - longitude pair and each $r_i \in E$ is a road segment described by a polyline. Let $x_1, x_2, \ldots, x_n$ denote a trajectory with n GPS sampling points, and x_i stands for the i^{th} sampling point of the trajectory. The goal is to find a sequence of road segments $ans_1, ans_2, \ldots, ans_n$ where ans_i is the matching result of sampling point x_i.

Map matching is important for an in-vehicle navigation system to determine which road a vehicle is on and an essential step for many other applications and researches, including traffic flow analysis and the study of behaviors of drivers.

The ACM SIGSPATIAL Cup 2012 is about map matching. A set of vehicle trips represented by GPS trajectories at about one second sampling rate and their correct mapping to road segments were given. The competing programs were evaluated on a set of vehicle trips and road networks unknown to the participants and ranked quantitatively on mapping accuracy and speed.

We quickly realized that several existing map matching algorithms, such as Hidden Markov Model (HMM) map matching [1] [2] [3], Interactive Voting-based Map Matching (IVMM) [4] and ST-Matching [6], have already reached high matching accuracy. So we consider it a sensible strategy to put more emphasis on reducing running time.

This paper makes two contributions to the research of map matching. First, it combines map matching with multi-threading technology to reduce running time, which takes full advantage of multi-core CPU resources. Road segments can be indexed in parallel by different threads during the indexing stage. Similarly, during the matching stage, several trajectories can also be matched in parallel. Second, we propose an improvement to the HMM algorithm [3]. Due to the inevitable measurement errors of GPS devices, a trajectory segment may be closer to a side road even though a vehicle actually moves on a main road. So speed limit information is added into HMM algorithm to correct the special mismatching cases we found.

The rest of this paper is organized as follows: Our approach to the parallel building of a grid index and to the parallel matching of trajectories will be described in Section 2. In Section 3, the improvement in the HMM algorithm will be discussed in details. The performance of our program in running time and matching accuracy will be shown in Section 4 and we will conclude the paper in Section 5.

2. MATCHING QUICKLY
Multi-threading technology can be applied to both indexing and matching stages of our program. Retrieving road segments close to a GPS sampling point quickly is important in map matching. The cost of brute force search of millions of road segments in a large digital road network is prohibitive. We use a simple grid-based index to map road segments to grid cells. The non-hierarchical structure guarantees the independence of cells and allows for parallel index building and query. The sampling points in different trajectories are independent from each other so several trajectories can be matched at the same time by different threads.

2.1 Grid Index on Earth Surface
During the matching process, we need to decide which road a GPS sampling point can be matched to. According to statistical analysis, the average GPS error is about 15 meters. That is to say, a GPS point is almost impossible to be on a road segment hundreds of meters away. In our algorithm, a GPS point can be matched to the road segments less than 50 meters away. This greatly reduces the

ACM SIGSPATIAL GIS '12, November 6-9, 2012. Redondo Beach, CA, USA

number of candidate road segments for each sampling point to match, and experiments show that this reduction of candidate road segments has no influence on matching accuracy. In order to get roads near a point efficiently, we build a grid index on the Earth's surface. We partition the given digital map by latitude and longitude. The partition makes the grid cells not less than 50 meters in both dimensions. Because the Earth is approximately a sphere, the southern cells are wider than the northern ones. This is shown in Figure 1.

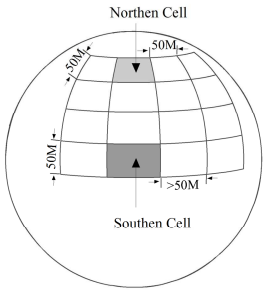

Figure 1. Grid cells on spherical surface.

A part of the road network and the associated grid index are illustrated by Figure 2. There are 8 road segments, named r_1 to r_8, and 12 cells, named A to L, in the view. Each road segment in the road network is inserted into the cell(s) it crosses, or rather each cell stores a set of road segments that cross the cell itself. For example, in Figure 2, the road r_5 will be inserted into the cell A, E and I because r_5 crosses these 3 cells on the map. Road r_1, r_2 and r_3 are stored in cell B for all of them cross the cell.

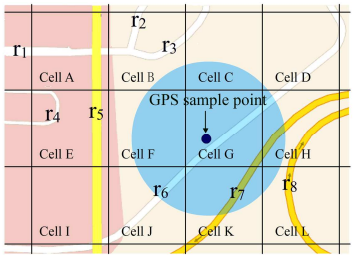

Figure 2. Part of a road network is indexed into several cells. There are 8 road segments and 12 grid cells on the map. A GPS sampling point (the blue dot) is in Cell G. The blue circle covers the land within 50 meters.

2.2 Building Index in Parallel

Each road segment can be inserted into one or more grid cells in parallel without obstructing others. Here multi-threading technology can be used to reduce the running time. Let N denote the number of threads used to build the grid index. All roads on a map are divided into N groups and each thread handles one group. Every road segment in each group is inserted into the grid cell(s) that it crosses.

Algorithm 1 describes the grid index building algorithm. Step in line 1 can be done through trivially dividing the array indice to the road segments into equal sized groups. Codes from line 3 to line 7 runs in parallel.

Algorithm 1 Grid Index Building

Input: road network G, thread number N
Output: grid index GI, where GI_j stands for the set of road segments that cross the j^{th} grid cell

1: Divide the road segments in G into N sets, let E_i denote the i^{th} set of the segments;
2: **for each** E_i
3: **new** thread(E_i)
4: **for each** road segment r **in** E_i
5: **for each** grid cell c **in** GetCrossCells(r)
6: $GI_c \leftarrow GI_c \cup \{r\}$;
7: **end** thread;
8: **return** GI;

Here GetCrossCells(r) is a function which returns the grid cells that road r crosses. This can be realized by analytic geometry algorithms easily. The details of indexing performance will be shown in Section 4 (Figure 5).

2.3 Querying Index Efficiently

Given a sampling point, the grid cell in which the point is located can be calculated directly by the geographic coordinates of the point. This grid cell and the eight grid cells around are going to be checked. If the distance between the sampling point and the grid cell is less than 50 meters, then the road segments stored in the cell will be checked. Each road segment in these cells that is less than 50 meters from the sampling point becomes a candidate road to which the sampling point may be matched. For example, there is a GPS sampling point in Figure 2. The highlighted circle in Figure 2 shows a region around the GPS sampling point, and road segments that cross this circle is the candidate road segments of this GPS sampling point. The sampling point is in cell G, and eight cells around are all less than 50 meters from the sampling point. So the road segments stored in the cell G and in the cells B, C, D, F, H, J, K and L (i.e. the road r_1, r_2, r_3, r_6, r_7 and r_8), are going to be checked. Finally, r_6 and r_7 are the candidates for the distances between the point and both r_6 and r_7 are less than 50 meters. Algorithm 2 shows how to get the candidate roads nearby a sampling point efficiently.

Algorithm 2 GetCandidateRoads(G,z);

Input: grid index GI, sampling point z;
Output: the list of roads around z within 50 meters L

1: $L \leftarrow \emptyset$;
2: $c \leftarrow$ GetCell(z); //get the grid cell in which z is located
3: $L \leftarrow L \cup GI_c$;
4: **for each** grid cell c' around c; //get 8 grid cells around c
5: **if** distance(z, c') <= 50.0
6: $L \leftarrow L \cup GI_{c'}$
7: **for each** road segment r **in** L
8: **if** distance(z, r) > 50.0
9: $L \leftarrow L - \{z\}$;
10: **return** L;

2.4 Matching in Parallel

A GPS trajectory consists of a series of GPS sampling points. Since any two points from two different GPS trajectories have no relationship with each other, multi-threading technology can be applied to the matching stage of the program as well. Let N denote the number of threads the program uses and M denote the number of trajectories to be matched. All trajectories to be matched are divided into N groups and each thread matches the trajectories in one group. Figure 3 illustrates the multi-threading map matching where $N = 4$ and $M = 8$. Rectangle labelled Ti represents the i^{th} trajectory. The width of a rectangle indicates the number of sampling points in the corresponding trajectory. Generally, a trajectory containing more sampling points takes more time to finish matching. So the width of a rectangle also indicates the matching time of the corresponding trajectory.

When only one thread is used, the eight trajectories form a long queue. It takes time t_1 to finish matching all trajectories. When 4 threads are used, each queue has only 2 trajectories and the

matching time is greatly reduced to t_2. The reduction of matching time in our experiment is shown in Section 4 (Figure 6).

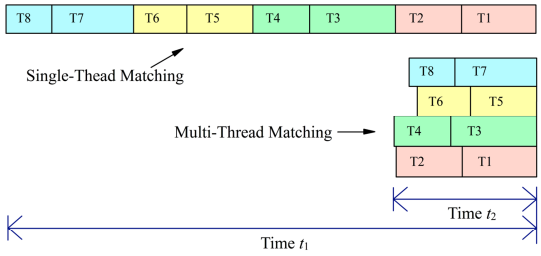

Figure 3. Eight trajectaries can be divided into 4 groups when 4 threads are used. The matching time reduces from t_1 to t_2.

Algorithm 3 describes the parallel matching algorithm. The Hidden Markov Model (HMM) based matching algorithm on line 5 is based on previous work in [3], and our improvements to this algorithm is emphasized in Section 3.

Algorithm 3 Parallel Matching

Input: road network G, thread number N, GPS trajectories $L_1, L_2, \dots, L_m$
Output: matching result routes $A_{L_1}, A_{L_2}, \dots, A_{L_m}$
1: Divide the GPS trajectories of into N sets, let S_i denote the i^{th} set of the trajectories;
2: **for each** S_i
3: **new** thread(S_i)
4: **for each** GPS trajectory L **in** S_i
5: $A_L \leftarrow$ HMM Map Matching(G, L);
6: **end** thread;
7: **return** $A_{L_1}, A_{L_2}, \dots, A_{L_m}$

3. IMPROVEMENT ON HMM MAP MATCHING

Paul Newson and John Krumm present a map matching algorithm based on HMM [3] which achieves very good matching accuracy in their experiments. However, when we used their original HMM map matching algorithm on the training data and synthetic data, some special mismatching cases were found.

Let $p(x_t|r_i)$ be the *measurement probability* [3] of matching a sampling point x_t to a road segment r_i. When calculating $p(x_t|r_i)$, the original HMM map matching algorithm only considers the spatial distance between x_t and r_i. Due to the inevitable GPS measurement errors, a vehicle traveling on the main road may log a sequence of GPS sampling points that are closer to the side road. In this situation, the original algorithm may mismatch these points to the side road, as illustrated in Figure 4. Here r_1 is the main road, while r_2 is a parallel side road, maybe linked to another road (the verticle road in Figure 4).

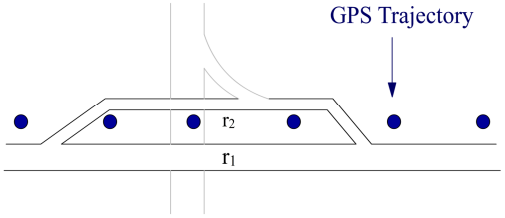

Figure 4. This shows an example of the *main road – side road* situation.

This kind of flyover is common in authentic network, some of the examples are shown in Section 4 (Figure 8).

Intuitively, a vehicle is more likely to travel on a main road with higher speed constraints, rather than a side road. Based on this observation, we consider the speed limits of road segments when calculating the *measurement probability*. Let $MaxSpeed_i$ denote the speed limit of r_i, we re-formulate the *measurement probability* as

$$p'(x_t|r_i) = p(x_t|r_i) * MaxSpeed_i \quad (1)$$

Figure 8 illustrates one of the experiments we did to verify our improvement, and the matching results show that our algorithm can handle the side road cases correctly.

In addition, when a route becomes circuitous, for any two sampling points x_t and x_{t+1} on this route, the shortest path distance between x_t and x_{t+1} may be much larger than the Euclidean distance between them. In this case, the transition probability corresponding to that route will become very small, which triggers *HMM breaks* [3]. The more frequently that *HMM breaks* occur, the less efficient the matching algorithm will be. An alternative to lower the break occurs is to enlarge the parameter β, which measures the tolerance of non-direct routes [3]. An appropriately amplification of β can effectually reduce the break cases, with inappreciable influence on the matching accuracy. We did experiments and found the optimal values of β in different sampling period, which is shown in Table 1.

4. EXPERIMENT PERFORMANCE

We ran our program on the training data of ACM SIGSPATIAL Cup 2012 on a 2.40GHz – Intel® Core™ i5 CPU. The sampling period of all original data is 1 second. We processed the data and generated low-sampling-rate test files in order to test our algorithm in the same situation.

The accuracy of our algorithm at different sampling periods is shown in Figure 5. It achieved satisfactory matching accuracy at both high and low sampling rates. The mismatching percentage for a certain sampling rate equals to the number of mismatching points divided by the number of all points at this sampling rate.

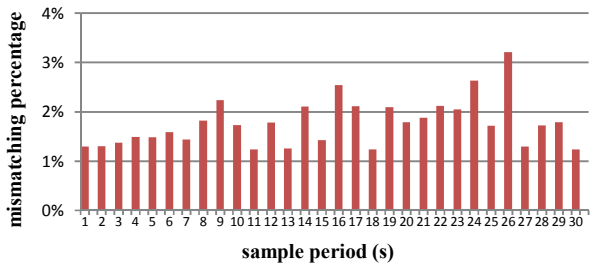

Figure 5. Mismatching percentage at different sampling periods.

As shown in Figure 6, index building time decreases greatly when multi-thread technology is applied to our program. The road network of Washington State, USA is used in the test. There are 1,283,539 road segments in this road network. It takes the program 0.37s to build the grid index when 4 threads are used.

The reduction of matching time using multi-core CPUs is quite similar to the indexing. Test files are divided into several groups and each thread deals with one of these groups. If the matching time using one thread is set to 100%, 2-thread matching time is 65% and 4-thread matching time is 43%.

Our program reaches its best performance when all cores in the CPU are fully used, as shown in Figure 7.

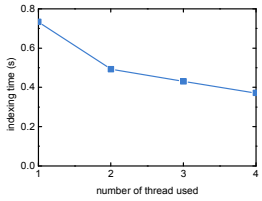

Figure 6. Indexing time using 1 to 4 thread(s)

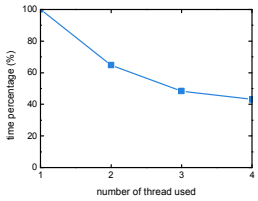

Figure 7. Matching time using 1 to 4 thread(s) with respect to 1 thread

We obtain the optimal values of β from training data by considering both matching accuracy and efficiency for different sampling rates. With the value of β in Table 1, our algorithm handles the HMM break discussed in Section 3 both correctly and efficiently.

Table 1. Value of β at different sampling periods (S = *Sampling Period*)

S	β	S	β	S	β
1	0.49037673	11	4.67319795	21	12.55107715
2	0.82918373	12	5.41088180	22	15.82820829
3	1.24364564	13	6.47666590	23	17.69496773
4	1.67079581	14	6.29010734	24	18.07655652
5	2.00719298	15	7.80752112	25	19.63438911
6	2.42513007	16	8.09074504	26	25.40832185
7	2.81248831	17	8.08550528	27	23.76001877
8	3.15745473	18	9.09405065	28	28.43289797
9	3.52645392	19	11.09090603	29	32.21683062
10	4.09511775	20	11.87752824	30	34.56991141

The special mismatching cases discussed in Section 3 can often be found in an authentic road network. Figure 8 shows some examples of this situation. The original HMM map matching algorithm matches the a series of GPS sampling points to the side road (marked with green line), while our improvement to the original HMM map matching algorithm rectified these mismatching cases.

Figure 8. The map of the special cases discussed in Section 3.

5. SUMMARY

Map matching can benefit from multi-threading technology because the processing of road segments can be separated from each other during indexing and each sample trajectory is independent during matching. In this paper, our algorithm is designed to run on a multi-core CPU. Multi-thread technology is applied in order to greatly shorten the running time. We improved the Hidden Markov Model map matching algorithm. When special cases of road network are considered, our improvement makes the matching result more accurate.

In order to evaluate the performance of our program, several experiments are conducted. The results show that our modification to the original Hidden Markov Model map matching algorithm does correct some special mismatch cases. And a dramatic reduction in running time is achieved when resources of a multi-core CPU are fully used.

To the best of our knowledge, our work is the first map matching solution that has put much emphasis on running time. We would like to extend our work in a few directions. First, multi-thread technology does reduce the indexing time greatly. However we believe the design of the indexing algorithm can be improved to reduce running time further. Second, a digital map, such as a OpenStreet map produced by crowd sourcing and used in this competition, has many errors (for example, wrong linking relationship between two different roads). These errors on digital road network often cause a long sequence of mismatching in current map matching algorithms. We plan to do research on more robust map matching algorithms to survive these defective digital maps. Our future work will explore further in these circumstances.

6. ACKNOWLEDGMENTS

We would like to thank ACM SIGSPATIAL Cup 2012 organizers Mohamed Ali, Travis Rautman, John Krumm and Ankur Teredesai for the interesting competition. This work is partially supported by the National Natural Science Foundation of China (NSFC) under grant No. 61073001 and the National Science Foundation (NSF) under grant No. IIS-1017926.

7. REFERENCES

[1] Hummel, B., Map Matching for Vehicle Guidance, in Dynamic and Mobile GIS: Investigating Space and Time, J. Drummond and R. Billen, Editors. 2006, CRC Press: Florida.

[2] Krumm, J. and Letchner, J. and Horvitz, E., Map Matching with Travel Time Constraints, in Society of Automotive Engineers (SAE) World Congress, 2007.

[3] Newson, P. and Krumm, J., Hidden Markov Map Matching Through Noise and Sparseness, Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS), 2009.

[4] Yuan, J. and Zheng, Y. and Zhang, C. and Xie, X. and Sun, G.Z., An Interactive-Voting Based Map Matching Algorithm, 11th International Conference on Mobile Data Management (MDM), 2010.

[5] http://depts.washington.edu/giscup/home. [cited 2012].

[6] Lou, Y. and Zhang, C. and Zheng, Y. and Xie, X. and Wang, W. and Huang, Y., Map-Matching for Low-Sampling-Rate GPS Trajectories, Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS), 2009.