MINJIE CHEN

Processing of Maps and GPS Trajectories in Location-based Applications

Publications of the University of Eastern Finland.
Dissertations in Forestry and Natural Sciences
No. 81

Academic Dissertation
To be presented by permission of the Faculty of Science and Forestry for public examination in the Louhela auditorium at the University of Eastern Finland, Joensuu, on September, 21th, 2012, at 12 o'clock noon.

School of Computing

Kopijyvä Oy Joensuu, 2012
Editors: Prof. Pertti Pasanen and Prof. Pekka Kilpeläinen

Distribution:
Eastern Finland University Library / Sales of publications
P.O.Box 107, FI-80101 Joensuu, Finland
tel. +358-50-3058396
http://www.uef.fi/kirjasto

ISBN: 978-952-61-0880-3 (printed)
ISSNL: 1798-5668
ISSN: 1798-5668
ISBN: 978-952-61-0881-0 (PDF)
ISSN: 1798-5676 (PDF)

Author's address: University of Eastern Finland
School of Computing
FINLAND
email: mchen@cs.joensuu.fi

Supervisors: Professor Pasi Fränti, Ph.D.
University of Eastern Finland
School of Computing
FINLAND
email: franti@cs.joensuu.fi

Professor Mantao Xu, Ph.D.
Shanghai Dianji University
School of Electronical Engineering
CHINA
email: xumt@sdju.edu.cn

Reviewers: Professor Ioan Tabus, Ph.D
Tampere University of Technology
Department of Signal Processing
FINLAND
email: tabus@cs.tut.fi

Professor Sébastien Lefèvre, Ph.D
VALORIA Research Laboratory in Computer Science
Université de Bretagne-Sud
FRANCE
email: sebastien.lefevre@univ-ubs.fr

Professor Pierre-François Marteau, Ph.D
VALORIA Research Laboratory in Computer Science
Université de Bretagne-Sud
FRANCE
email: Pierre-Francois.Marteau@univ-ubs.fr

Opponent: Professor Susanto Rahardja, PhD, IEEE Fellow
Institute for Infocomm Research
1 Fusionopolis Way, #21-01
Connexis (South Tower)
SINGAPORE
email: rsusanto@i2r.a-star.edu.sg

ABSTRACT:

This thesis consists of two main parts:

In the first part, we study filtering algorithms for raster map images. Raster maps are one type of color-mapped images, where each color represents a different class of semantic map object. They are mostly used at the client side with no additional rendering cost. Firstly, a multi-layer filtering algorithm is proposed by transforming the problem of color-mapped image denoising into the binary domain. Secondly, we present a statistical filtering algorithm that extends the solution dealing with additive Gaussian noise and mixed Gaussian-impulsive noise. Later, we focus on the problem of optimized context selection using a novel context-based voting method to identify the noisy pixels.

The second part of the thesis is dedicated to the simplification and compression of vector maps and Global Positioning System (GPS) trajectory. Firstly, a bottom-up multi-resolution polygonal approximation algorithm with linear time complexity is proposed for the GPS trajectory simplification. This gives a joint optimization on the two proposed error criteria; local integral square synchronous Euclidean distance (LSSD), and integral square synchronous Euclidean distance (ISSD). Secondly, we study the problem of lossy vector map compression in Geographic Information Systems (GIS). New compression algorithms are designed by combining point reduction and a quantization process. We propose a fast solution as well as an optimized codebook selection strategy. We also extend the problem for lossy compression of GPS trajectories under maximum synchronous Euclidean distance (SED). In the proposed algorithm, speed and direction changes are used in the encoding process instead of the differential coordinates used in vector map compression. Line simplification and quantization are combined in order to seek an optimized approximated trajectory for compression.

Keywords: Raster Map Image, Image denoising, Statistical Filtering, Vector Map Compression, GPS Trajectory Simplification, GPS Trajectory Compression.

AMS Mathematics Subject Classification: 93E11, 94A08, 68U10, 68P30, 68W99
Universal Decimal Classification: 004.021, 004.932, 510.5, 912.43
Library of Congress Subject Headings: Digital maps; Image processing; Data compression (Computer science); Image compression; Electronic noise; Noise control; Filters (Mathematics); Algorithms; Location-based services; Global Positioning System; Geographic information systems

Acknowledgments

The work presented in this thesis was carried out at the School of Computing, University of Eastern Finland, Finland, during the years 2008-2012.

I am glad to have had the opportunity to study in the Speech and Image Processing Unit at the School of Computing, University of Eastern Finland.

I am thankful to my supervisors, Prof. Pasi Fränti and Prof. Mantao Xu for their support and guidance over the years. I would also like to express my gratitude to Prof. Juha Alho and Dr. Ville Hautamäki for the useful suggestions and discussions during my research work. I would also like to thank my colleagues for the help during my PhD studies.

I am thankful to Professor Ioan Tabus, Professor Sébastien Lefèvre and Professor Pierre-François Marteau, the reviewers of the thesis, for their constructive feedback and comments. I would also thank Professor Susanto Rahardja for acting as my opponent.

I would thank my parents who have always encouraged me to follow my own paths.

This research has been supported by the East Finland Graduate School in Computer Science and Engineering (ECSE), Tekniikan edistämissäätiö (TES), MOPSI project and Nokia foundation. All their support is gratefully acknowledged.

Joensuu Aug 7th 2012 Minjie Chen

List of abbreviations

CRF	Conditional Random Field
CT	Context-Tree Modeling
DCT	Discrete Cosine Transform
DUDE	Discrete Universal Denoiser
ECPNN	Entropy-constrained Pair-wise Nearest Neighbor
ECVQ	Entropy-constrained Vector Quantization
GIS	Geographic Information Systems
GPS	Global Positioning System
LBS	Location-based service
MAP	Maximum a posteriori
MRF	Markov Random Field
OMP	Orthogonal Matching Pursuit
RSDP	Reduced Search Dynamic Programming
SVD	Singular Value Decomposition
VQ	Vector Quantization

LIST OF ORIGINAL PUBLICATIONS

Denoising of Raster Map Images

[P1] **M. Chen**, M. Xu and P. Fränti, "Multi-layer Filtering Approach for Map Images", *IEEE Int. Conf. on Image Processing* (ICIP'09), Cairo, Egypt, 3953-3956, 2009.

[P2] **M. Chen**, M. Xu, P. Fränti, "Adaptive Context-tree-based Statistical Filtering of Raster Map Images Denoising", *IEEE Trans. on Multimedia*, 13(6), 1195-1207, 2011.

[P3] **M. Chen**, M. Xu, P. Fränti, "Adaptive Filtering of Raster Map Images Using Optimal Context Selection", *IEEE Int. Conf. on Image Processing* (ICIP'11), 77-80, Brussels, Belgium, 2011.

Compression and Simplification of GPS Trajectory and Vector map

[P4] **M. Chen**, M. Xu, P. Fränti, "A Fast $O(N)$ Multi-resolution Polygonal Approximation Algorithm for GPS Trajectory Simplification", *IEEE Trans. on Image Processing*, 21(5), 2770–2785, 2012.

[P5] **M. Chen**, M. Xu, P. Fränti, "Fast Dynamic Quantization Algorithm for Vector Map Compression", *IEEE Int. Conf. on Image Processing* (ICIP'10), Hong Kong, China, 4289-4292, 2010.

[P6] **M. Chen**, M. Xu, P. Fränti, "Optimized entropy-constrained Vector Quantization of Lossy Vector Map Compression", *IEEE Int. Conf. on Pattern Recognition* (ICPR'10), Istanbul, Turkey, 722-725, 2010.

[P7] **M. Chen**, M. Xu, P. Fränti, "Compression of GPS Trajectories", *IEEE Data Compression Conference* (DCC'12), 62-71, Snowbird, USA, 2012.

The original publications are included at the end of this thesis by permission of their copyright holders. Throughout the overview, these papers will be referred to as [**P1**]–[**P7**]. The contributions of the authors of these papers to this dissertation can be summarized as follows: In [**P1**] the author extends the work of the multi-layer method for compression in [88] to the filtering problem. In [**P2**, **P3**] our work starts from the statistical filtering for impulsive noise presented in [4, 89], and it is generalized for different noise models. [**P2**] is a journal version of paper [21] and [**P3**] is an extension of that work. In [**P4**] we start our work from the multi-resolution framework in [57, 75] for vector maps. A priority queue structure [30] is used during the stage of graph construction. Later, new error measures are proposed for GPS trajectories. For the proposed multi-resolution algorithm, the proof of the time complexity is also given. [**P5, P6**] are the extensions on the early work of vector map compression in [2, 3, 58, 60, 102]. Later we modified the scheme to fit the problem of GPS trajectory compression in [**P7**]. In all these papers, the principal author did the algorithm development, coding and wrote the paper. The second and third authors performed the paper revision.

Contents

1. Introduction

With the rapid development of wireless communication, mobile computing technologies, location-based services have become increasingly important research areas; especially concerning data types, such as GPS trajectories, vector maps, and raster maps.

1.1 GPS TRAJECTORY

Over recent decades many spatial trajectories have been collected by geo-positional mobile devices, which represent the mobility of a variety of moving objects, such as: people, vehicles, animals, and natural phenomena, in both indoor and outdoor environments (Fig. 1.1). To explore the broad applications, much systematic research and development in the new computing technologies has investigated the storage, preprocessing, retrieving, and mining of these spatial trajectories [70].

Recently, the pre-processing of spatial trajectories has become an important research topic that has attracted extensive attention. For example, reducing the data size of the trajectories is important to alleviate storage and communication overheads, as well as the computational workload on the server. Given that a GPS trajectory consists of a full series of time-stamped location points, the simplification process aims at generating its approximated trajectory by reducing the number of points with negligible errors. This can be considered as a polygonal approximation problem, which has been studied in computational geometry for many years [8].

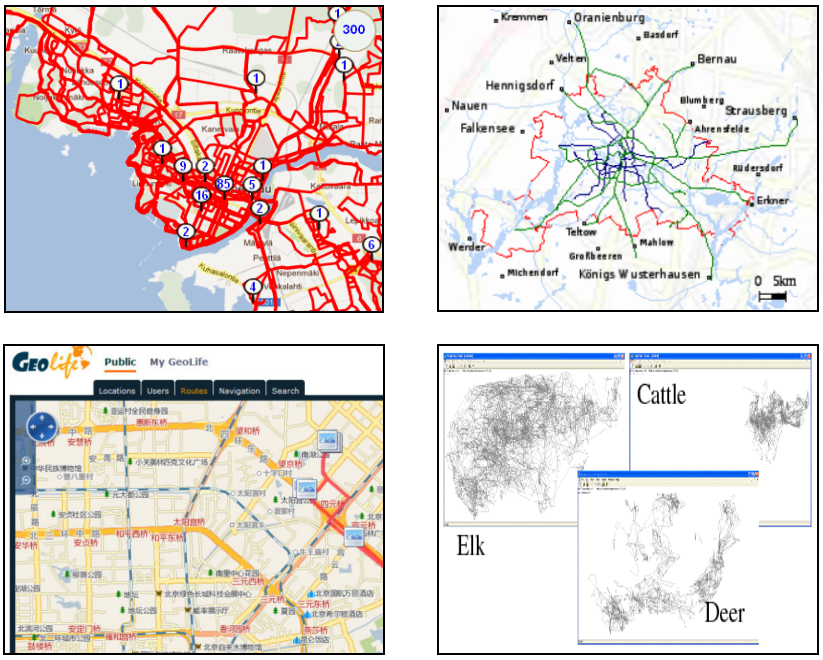

Fig. 1.1. *GPS trajectory dataset in MOPSI (up-left), Berlin MOD Cycling dataset (up-right), Geolife dataset (bottom-left) and animal movement analysis (bottom-right)*

If the encoding process is also considered, the problem becomes a topic in data compression, where the coding algorithm will also be integrated [97]. The difference between simplification and compression is, in the simplification process, the reduced data points are saved directly with a fixed bit-length, which is required to support the trajectory queues in the database. Meanwhile, these reduced trajectories can also be used for improving the efficiency of the visualization process, where spatial trajectories with different reduction rates are displayed to fit maps with different scales. On the other hand, when data compression techniques are used, a better compression ratio is achieved for the spatial trajectory data, which is appropriate for data storage.

1.2 VECTOR MAP FORMAT

Vector map is another important data type in location-based systems. Fig. 1.2 is a vector-based collection of Earth data at various levels of detail in a *Geographic Information System* (GIS). Vector maps are more compact and suitable for large databases providing both excellent flexibility for display and compact size for storage. Because of the large size of digital maps, their data often need to be compressed for map database storage and transmission to remote users [3, 58, 59, 62, **P5**, **P6**].

However, the main disadvantage of vector maps is their complex data structure, which can result in a higher cost during the visualization process.

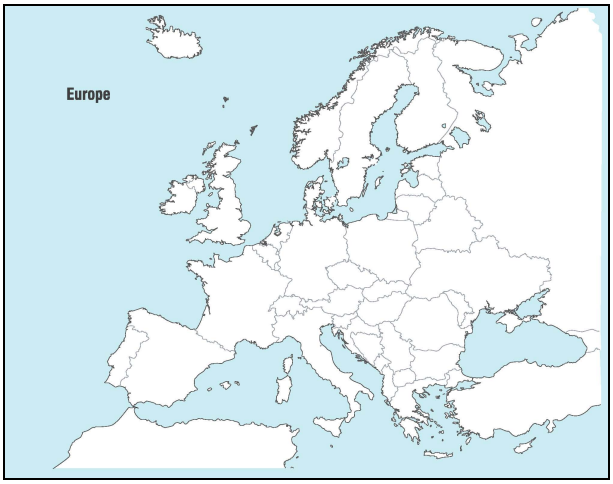

Fig. 1.2. *Example of vector map of Europe*

1.3 RASTER MAP FORMAT

In order to save on the visualization cost, raster formats are mostly used at the client-side because no additional processing is needed. In raster formats, the images are stored as a regular grid of pixel colors in which each color represents a different class of semantic map object. Raster maps often consist of repeated pixel-level detailed structures and sharp edges, but lack the smooth color transitions that are typical

of photographic images. They are mostly converted from a vector database by a vector-raster conversion, or from old maps using a digitization process, see Fig. 1.3.

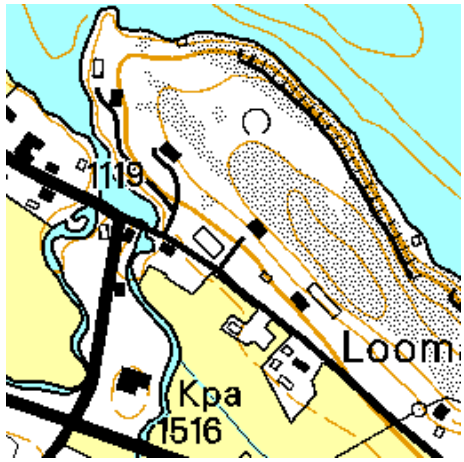

Fig. 1.3. *Example of raster map*

1.4 STUCTURE OF THE THESIS

The rest of the thesis will be organized as follows: we will describe two new filtering algorithms for raster map images in Section 2. A simplification algorithm for GPS trajectories will be presented in Section 3. Subsequently, compression of vector maps will be discussed in Section 4. In Section 5, compression algorithms for GPS trajectories will be offered. A summary of the contribution will be drawn in Section 6 and conclusions in Section 7.

2. Filtering of Raster Map Images

Raster maps have become an increasingly important source in geographic information systems. They do not require any additional processing and are suitable for delivery to multimedia applications.

However, image degradation may occur in the digitization process or during the vector-raster conversion, which results in mismatch and false recognition of important semantic map objects. Image denoising is therefore needed for accurate conversion of older maps into a raster format. Denoising can also be crucial for the later stage of raster map analysis, when extracting the semantic content (roads, contours and river) from a map [23, 47, 52, 71].

A great variety of noise removal techniques have been investigated for color image processing. However, most noise removal algorithms are developed specifically for only one type of noise model. For instance, to eliminate impulsive noise, a number of denoising algorithms have been developed by firstly identifying the potential noisy pixels in a color image and then employing a class of vector median filters over those pixels. Noisy pixels can be detected either by classifying each pixel directly in RGB color space [114] or by setting some statistical rules in terms of the variation within the local neighborhood [14, 15, 73, 100, 103]. However, these approaches need a training dataset or prior knowledge for constructing the statistical rules.

For additive Gaussian noise, a number of denoising algorithms have been proposed by selecting an optimal linear combination of a few basis elements in pixel-wise or block-wise order. For example, *wavelet denoising* [90] is proposed based on a local Gaussian scale mixture model in an overcomplete oriented pyramid representation. In a *non-local means filter* [11], the concept of locality in a bilateral filter is extended to the entire image. The *dictionary-based method* (K-SVD) is proposed in [35] by assuming that the image patches are sparse

representable. In K-SVD, *singular value decomposition* (SVD) and *orthogonal matching pursuit* (OMP) are used in dictionary learning and the denoising step, respectively. Based on the assumption that similar patches share similar dictionaries, K-SVD framework is further improved by a *non-local sparse model* (NLSM) [74]. Another patch redundancy-based framework, BM3D [28], adopts a hybrid approach of grouping similar patches and performing collaborative filtering in a DCT domain. *Markov random fields* (MRF) [95] or *conditional random field* (CRF) [7] are also applied to denoising natural images, where a training process is used to learn the parameters of the model from example images. In [18], *patch-based locally optimal wiener filtering* (PLOW) is proposed where patch redundancies are considered to improve the denoising performance on both geometric and photometric properties. Recently, *sparsity-based image denoising* [33] has been presented to unify both local and non-local properties in a natural image. It is formulated as a double-header L_1-optimization problem where the regularization involves both dictionary learning and structural clustering.

However, these algorithms are limited to the cases where the true signal can be approximated by a linear combination of a few basis elements and therefore, they are mostly designed for denoising continuous tone images and do not work well for color-mapped images.

On the other hand, raster maps have different properties, such as: complicated spatial structures, one-pixel thin lines, textured areas, dashed and dotted lines, text, and symbols. The problem of false filtering exists with most filters designed for photographic imagery when processing these kinds of spatial structures. This is because these filters consider local intensity variation as noise, but ignore repeated patterns in the entire image. High variance regions including written text, symbols, and textured backgrounds lack uniformity, but their presence is vital for the readability of the map. Examples of such structures are shown in Fig. 2.1. Filtering of raster map images can be chosen as a case study of a more general class of color-mapped image denoising problem with a discrete number of output colors.

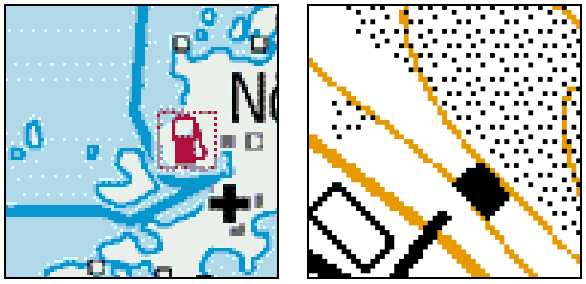

Fig. 2.1. *Examples of complicated structures that are treated as noise by most filters.*

2.1 MULTI-LAYER FILTERING

Problems with filtering of raster map images can be partially solved by applying an ordering criterion in local regions to only those pixels that are identified as (or assumed to be) noise or outliers [14, 15, 73, 100, 103]. However, nonlinear filters designed for conventional color images may eliminate the most useful edge information and detailed structures when they are applied to raster map images. Even though statistical modeling approaches [64, 89, 108] have made significant progress by learning image structure and preserving the repetitive structures, these methods have high memory consumption and computational expense.

Thus, a multi-layer image filtering approach is proposed in [**P1**]. The use of multi-layer decomposition was originally presented in [38] for image compression. Instead of using an order-statistics filter for color vectors, the image I is first decomposed into a series of binary layers. Suppose (i, j) denotes a given pixel of the raster map image I, and that the number of colors N for I is very limited, e.g., $N << 256$. The color for the given pixel $I(i, j)$ can be uniquely determined by a binary vector $\mathbf{x}_L$:

$$\mathbf{x}_L = (L_1(i,j),\ldots L_N(i,j)), \text{where } L_t(i,j) = \begin{cases} 1, \text{if } t = I(i,j) \\ 0, \text{ otherwise} \end{cases} \tag{2.1}$$

$L_t(i,j)$ can be treated as the value of pixel (i, j) of t^{th} binary layer. Therefore, any binary filtering algorithm, such as a morphological filter [10] or statistical filter [108], can be applied to each binary layer separately.

Once all the layers have been filtered, the resulting binary images $\{L^*_t \mid t = 1,\ldots, N\}$ are merged to reconstruct a raster map image I_G in a similar format of the input map image I. However, for a given pixel (i, j), there might be a number of layers L_C such that:

$$L_C = \{ L^*_t \mid L^*_t(i, j) = 1, 1 \le t \le N\} \quad (2.2)$$

which indicates that the resulting color $I_G(i, j)$ must be selected among L_C according to some criteria. In other words, for each pixel (i, j), N binary layers must be ordered for the sake of merging the filtered binary layers.

For example, a graph-based algorithm [65] has been developed for ordering these binary layers, which is used in the compression of raster maps. The algorithm seeks an optimal ordering of the binary layers by using the minimum spanning tree over a compression cost matrix. However, the construction of a compression cost matrix often incurs a huge computational cost. A simple way is to prioritize each color according to its frequency or occurrence. Namely, the higher the occurrence of one color, the lower priority it will be assigned. A demonstration of a multi-layer map image filtering algorithm by global color frequency can be found in Fig. 2.2.

Merging the filtered binary images using the same ordering criterion will result in the problem of damaging the important disconnected semantic objects. For example, in Fig. 2.2, blue color is assigned with a higher priority than white by using a global layer ordering scheme. As a result, it will cause the island inside the sea region to become corrupted after the merging step.

To overcome this difficulty, a region-based ordering scheme is proposed. An image segmentation operation is firstly conducted to segment the raster map image into several distinct regions. After that, different color priorities can be set according to the color occurrence in different segmented regions. Here, instead of performing image segmentation on the input raster map, a multi-layer image segmentation algorithm is applied, which first extracts large-size initial regions from all the filtered binary layers L^*. Those candidate regions are then refined via dilation, filling holes, and connected component labeling, sequentially. A mask image is then constructed by adding those initial candidate regions one by one, according to two region-based features: the ratio of object pixels and the percentage of

unlabeled pixels. Once the mask image is obtained, unlabeled pixels are assigning with the label of their nearest segmented region. Figure 2.3 illustrates an example of the merging process on filtered binary images by using the proposed region-based layer ordering scheme.

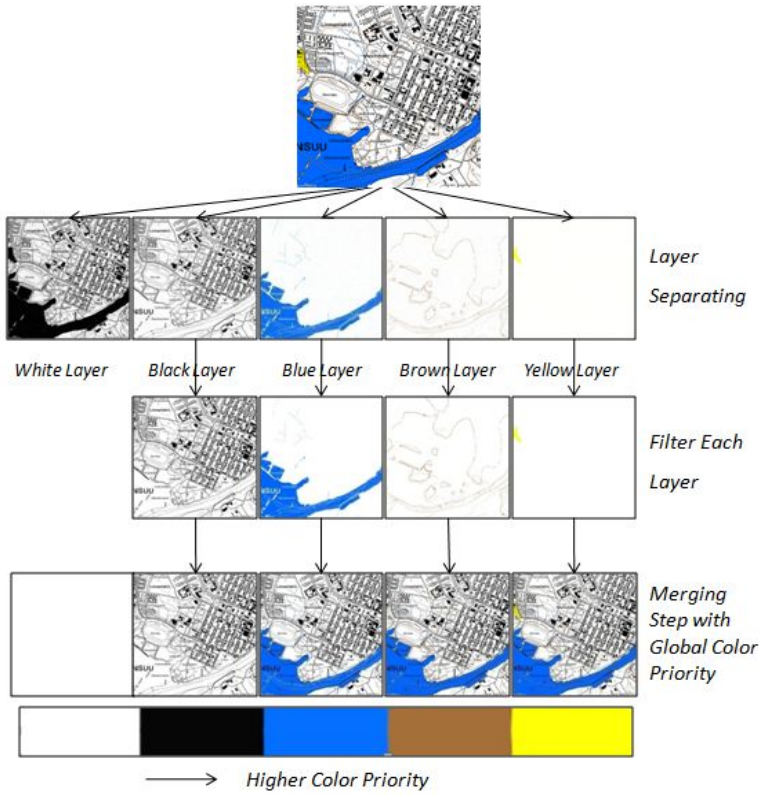

***Fig.* 2.2.** *Multi-layer framework using global layer ordering*

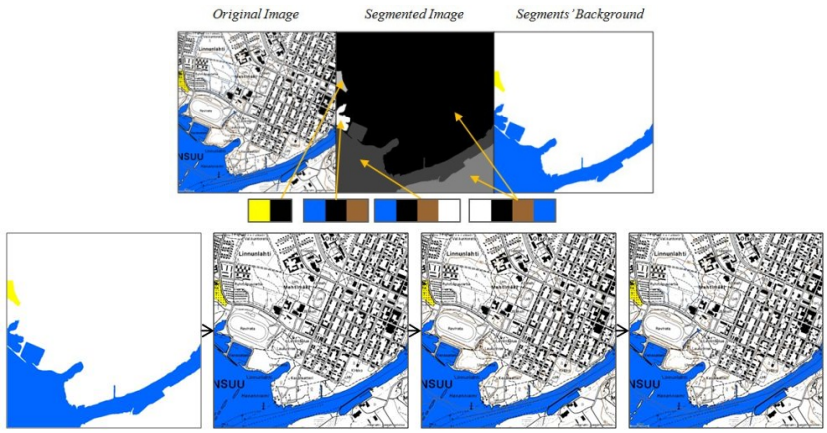

***Fig.* 2.3.** *Merging the filtered binary layers by region-based layer ordering*

2.2 STATISTICAL FILTERING OF RASTER MAP IMAGES

Although the multi-layer method is computationally efficient, the layer-ordering scheme does not work well when the number of colors increases. Therefore, statistical filtering is also considered in raster map image denoising. The main idea of the statistical filter follows an assumption that the image signal originates from a universal source. Hence, if the conditional probability $P(I(x)\,|\,\mathbf{c})$ in context $\mathbf{c}$ is less than a predefined value, the current pixel can be treated as noise and then replaced by the most probable color in the context.

A pioneering work in the art of statistical filtering is *discrete universal denoising* (DUDE) [108] for binary data filtering with a known noise channel. It comprises two steps: counting statistics for all context patterns encountered (analysis step), and denoising by utilizing the conditional probability in a local context (denoising step). This method is applicable in denoising binary images if the noise level δ can be reliably estimated. Namely, if the conditional probability of the current pixel x in context $\mathrm{P}(I(x)\,|\,\mathbf{c})$ is lower than $2\delta(1-\delta)$, it is considered to be noise and replaced by its complementary value. This kind of context-based approach can also be extended to the denoising problem with an unknown channel using the min–max criterion [42].

In contrast to the denoising algorithms that incorporate a prior model, statistical filtering is based on an unsupervised learning paradigm. Patterns that are frequently presented in the image are detected and considered as important image structures that should be preserved, whereas pixels that seldom appear in their context are treated as noise and can be filtered out. This allows filtering with preservation of borders and regular structures regardless of their size and variance. Three examples of context and their corresponding statistical distributions are demonstrated in Fig. 2.4. Domination of the most probable color can be observed in the first two examples (left and middle), but not in the final example (right).

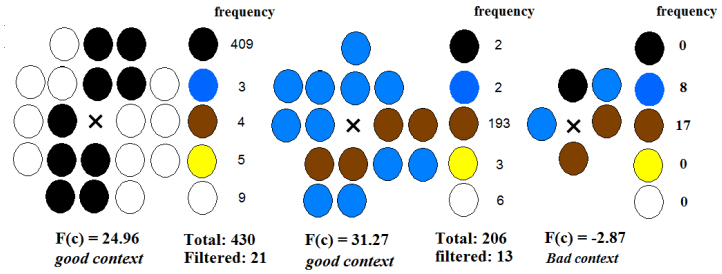

Fig. 2.4. *Examples of context distribution: the pixels in the left and middle contexts are filtered using the dominant color, whereas no filtering is done for the pixel in the context on the right.*

2.2.1 Context Tree Modeling

In DUDE, the memory allocation for learning the patterns grows exponentially with the size of the context template (number of the pixels in the context), which makes it of limited use in practice. Moreover, the conditional probability estimation becomes inaccurate when the contexts have rare appearance, which is known as the *context dilution problem*. To circumvent this problem, *context-tree modeling* [64, 85, 89] is applied by pruning redundant nodes of the context tree.

The classical context-tree modeling technique has been widely-used in the field of data compression [94, 107] with a time complexity of $O(kN)$, where N is the length of the data sequence and k is the depth of the context tree. The tree is built by estimating the count statistics via a sequential traversal of the image pixel by pixel. Each node of the context tree represents a single context by storing the count statistics of each color appearing for the current pixel relative to this context. As not all possible contexts are presented in the image, memory is allocated only for the actual pixel combinations appearing in the image.

In image compression, all pixels must be encoded regardless of the reliability of their contexts; moreover, one can keep track of the compression performance. Poor probability estimation leads to a longer code length and thus a large file size. Optimal pruning is done on each node of the tree in order to achieve the highest overall compression rate. For instance, a dynamic programming pruning technique was proposed to improve the context selection in [85], whereas universal context modeling was employed in [111].

There are other spanning criteria, such as maximum likelihood [43] and minimum coding cost [4, 85]. Adaptive context selection has also been extended for denoising gray-scale images [109] by using a *minimum description length* (MDL) guided criterion, where an optimal balance between the variance and bias of the errors in fitting a 2D *piecewise autoregressive* (PAR) model is found.

In our solution [**P2**], the spanning of the tree is terminated if the frequency of the context on a given node is less than a predefined threshold T. According to our experiments, there are only 50,000–100,000 contexts for a context template with 20 pixels in a 16 color map image, which is far below 16^{20}. An example of context-tree modeling is shown in Fig. 2.5.

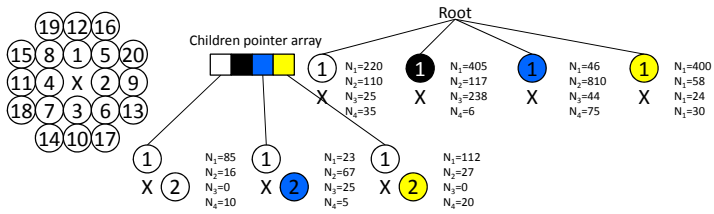

Fig. 2.5. *Part of the context tree (first two levels) and its context template*

2.2.2 Impulsive Noise

In this sub-section, a filtering algorithm for impulsive noise is presented for a raster map image, where a clean image is corrupted over an *M-ary Symmetric Channel* during the transmission.

In the filtering process for impulsive noise, conditional probability estimation plays a crucial role. Inaccurate conditional probability estimation can cause either a lack of detection of a noisy pixel or the addition of new noise. In contrast to image compression, several challenges exist when statistical filtering is applied for image filtering. Firstly, in practice, it is difficult to estimate the noise level from a single image. Secondly, in statistical filtering, the contexts themselves may include a significant number of noisy pixels. If the neighborhood pixels are contaminated by erroneous colors, the particular context would appear infrequently in the image, which causes an inaccurate

estimation of its conditional probability distribution. Thirdly, a proper decision rule for the filtering is a non-trivial design problem.

For the improvement of the statistical filter, we discuss the following three design problems:

a) Decision Rule for Pixel Denoising

Suppose we have a map image with impulsive noise that is generated by transmitting a clean image **X** over *an M-ary Symmetric Channel*. The optimal decision rule in *discrete universal denoising* (DUDE) [108] is essentially a *MAP estimator*, which is:

$$u_0 = \arg\max_{I(x)\in\{1,\ldots,M\}} P(I(x)|\mathbf{c}) \tag{2.3}$$

where **c** is the context of the pixel x. In image filtering, the current pixel x will be replaced by u_0, which is the value with the highest probability if the decision rule in Eq. (2.4) is met:

$$\frac{(M-1)^2(1-\delta)}{\delta((1-\delta)M-1)} P(I(x)=x_0|\mathbf{c}) - \frac{(M-1)}{((1-\delta)M-1)} P(I(x)=u_0|\mathbf{c}) < 1,\ x_0 = 1,2,\ldots,M \tag{2.4}$$

Note that this decision rule is designed for the count statistics collected on the noisy image. If the statistical distribution is collected on a clean image, the decision rule is:

$$P(I(x)=x_0|\mathbf{c}) < \delta/(M-1),\ x_0 = 1,2,\ldots,M \tag{2.5}$$

b) Context-Merging Strategy

Although DUDE follows a so-called "*asymptotic optimality*" property, it requires an infinite sequence of data source for estimating all the conditional distributions of the contexts, which is not realistic in practice. In particular, when the context of the pixel is contaminated by erroneous colors, the context can appear infrequently and its associated conditional probability would be far from its true distribution. In order to alleviate this problem, context-tree modeling is used by terminating the tree spanning with different criteria.

In [64, 89], a pruning step is used to remove those contexts with a frequency less than a predefined threshold T. After pruning, the statistics of its parent node are used for the probability estimation instead. The main challenge for the pruning step is that the tree is constructed in a fixed order, and that the noisy pixels may appear anywhere in the tree, not just in the leaf nodes. Thus, in many cases a clean pixel may also be removed from the context.

To this end, in [**P2**], we present a *context-merging* strategy for those infrequent contexts that are expected to be contaminated. For each context **c**, we first construct a set of sub-contexts:

$$S(\mathbf{c}) = \{\mathbf{z}_i \mid \mathbf{z}_i = \{\mathbf{c} / x_i\}\}_{i=1}^{k} \tag{2.6}$$

by removing the i^{th} element from the original context **c,** where $i = 1, \ldots, k$, and $\mathbf{z}_i$ is the sub-context. We sum up all the count statistics of the sub-contexts as the estimated distribution, if the frequency of the context **c** is lower than T:

$$P^*(I(x) = x_0 \mid x \in \mathbf{c}) = \begin{cases} P(I(x) = x_0 \mid x \in S(\mathbf{c})), \\ \quad \text{if } \sum_{i=1}^{M} n_{\mathbf{c}}(i) < T \\ P(I(x) = x_0 \mid x \in \mathbf{c}), \\ \quad \text{otherwise} \end{cases} \tag{2.7}$$

The idea of this *context-merging* strategy is that the sub-context will appear much more frequently in the image if the noisy pixel is removed from the context, whereas removing a clean pixel will not greatly change the statistics. After the *context-merging* operation, only noise-free sub-contexts become dominant, which serves as a good estimation of the conditional probability in the summation of all the sub-context distributions. An example can be seen in Fig. 2.6.

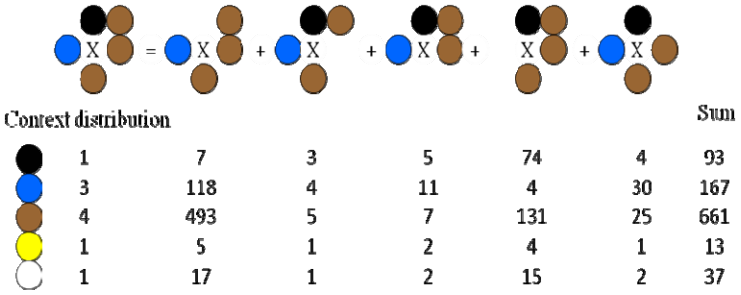

Fig. 2.6. *Example of context-merging strategy. A more reliable context distribution (93, 167, 661, 13, 37) is obtained instead of the estimation (4, 30, 25, 1, 2) obtained by the pruning operation. Colors with low probability (yellow and white) are replaced by the dominant color (brown).*

The computational complexity of the merging process is calculated as follows: for each infrequent context, the statistical distributions of M^2 (M is the size of the color palette) similar contexts are identified by tree traversal on the constructed context-tree, whereas the conditional probability estimation is calculated by summing up all the statistical distributions of the sub-contexts. Suppose that we have an infrequent context **c** with k pixels in context, the time complexity of the tree traversal is:

$$\sum_{i=1}^{k}((k-i)+(i-1)M)=O(k^2M) \tag{2.8}$$

c) Noise Level Estimation

In order to improve the filtering robustness under different noise levels, an estimation of the noise level δ of impulsive noise is needed. This can be estimated either in terms of the min–max criterion [42] or by using some image context metrics [115]. However, those solutions conduct the noise estimation in terms of the filtering results for each noise level, which is computationally expensive. A more practical estimate of δ in [84] is the minimized conditional probability in the contexts with "sufficient frequency". In a similar manner, the noise level δ is estimated here on the noisy image directly as:

$$\delta=1-\max_{\forall y,p(\mathbf{c})>10^{-2}}P(I(y)\,|\,\mathbf{c}) \tag{2.9}$$

where $P(\mathbf{c})$ is the probability of context $\mathbf{c}$.

2.2.3 Additive Gaussian Noise

For completeness, we study the statistical filtering in the case of additive Gaussian noise as well. In this case, the filtering of raster map images can be considered as a continuous-input–finite-output problem. For a noisy image $\mathbf{Y}$, the problem is defined as finding a denoised color-mapped image $\mathbf{Z}$ with M colors. As the size of the color palette is limited for raster map images, color quantization [63, 77, 113] can be efficiently applied if the color components are easily separable.

After color quantization, color space is partitioned into several regions, in which each color vector $\mathbf{Y}(y)$ is represented by its centroid $\mathbf{m}_{I(y)}$. As some color components can overlap, misclassification is inevitable in color quantization (see the quantized image in Fig. 2.7). Therefore, a novel iterative fusion algorithm is proposed in [**P2**], by calculating the distance from a pixel to the centroids in the color palette and the conditional probability relative to its context:

$$\begin{aligned} I(y) &= \arg\min_{I(y)\in\{1,\ldots,M\}}(-\log_2 g(\mathbf{Y}(y)\,|\,\mathbf{m}_{I(y)}) \\ &\qquad -\log_2 P(I(y)\,|\,\mathbf{c})) \\ g(\mathbf{Y}(y)\,|\,\mathbf{m}_{I(y)}) &= \exp(-\frac{\|\,\mathbf{Y}(y)-\mathbf{m}_{I(y)}\,\|^2}{2\sigma^2}) \end{aligned} \tag{2.10}$$

where σ is the variance of the additive Gaussian noise and y is the current pixel in the noisy image $\mathbf{Y}$. This fusion filter can be considered as a specific form of the energy function in a *Markov random field* [19, 104], which is derived by replacing the neighborhood similarity with conditional probability in the context.

After the fusion process, the color palette and the estimated noise variance σ are re-estimated. The fusion and estimation processes are performed iteratively. An example of the fusion result is shown in Fig. 2.7.

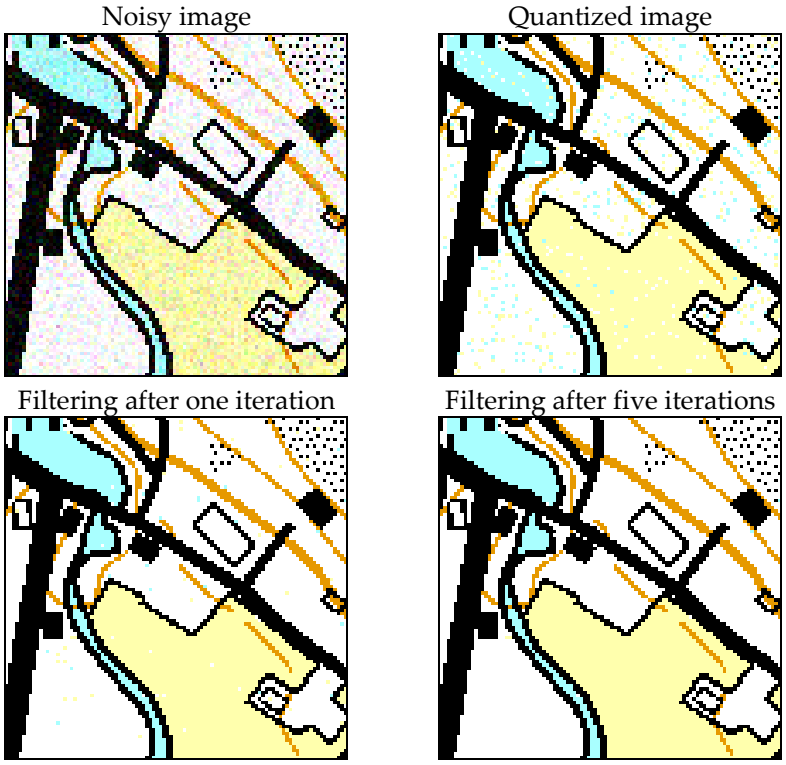

Fig. 2.7 *Filtering example (fragment from Image #1-26) of the fusion process for additive Gaussian noise.*

2.2.4 Mixed Gaussian-Impulsive Noise

To denoise an image with mixed Gaussian-impulsive noise, a straightforward approach is to apply two filters successively: one for the impulsive noise and the other for the Gaussian noise. For example, an algorithm based on *fuzzy peer group* [80] combines a statistical method for impulsive noise detection with replacement by an averaging operation to smooth out Gaussian noise.

In a similar manner, the proposed statistical filtering can also be extended to the problem of denoising mixed Gaussian-impulsive noise. The extension in [**P2**] combines both the case of the statistical filtering for impulsive noise, and the case of the fusion process for additive Gaussian noise. Namely, if the DUDE decision rule is met, the current pixel is identified as impulsive noise and then replaced by the color

with the maximum conditional probability. Otherwise, the fusion process is applied.

2.2.5 Time Complexity

In this sub-section, we give the time complexity of the proposed statistical filter for raster map image.

In general, the context-tree construction leads to a time complexity of $O(kN)$, where k is the size of the context and N is the number of the pixels in the image. Any context can have a maximum of NM/T child nodes, where M is the size of the color palette and T is the frequency threshold for *context merging*. In *context merging*, because the time complexity of every merging process is $O(k^2M)$ in Eq. (2.8), the total time complexity of the *context-merging* process is $O(k^2M \cdot NM/T)$. Additionally, the noise estimation procedure has a time complexity of $O(M)$ in which all the contexts with a frequency higher than $p(\mathbf{c}) = 0.01$ are extracted by the tree traversal process. In the denoising step, the DUDE decision rule is applied to determine whether a pixel is filtered or not. As the conditional probability of all contexts is pre-calculated, the filtering procedure has a time complexity of $O(N)$. As a result, the total time complexity for denoising impulsive noise is $O(k^2M^2 \cdot N/T)$.

For additive Gaussian noise, the clustering-based color quantization step has a time complexity of $O(MN)$. Context-tree construction and *context merging* have the same complexity as impulsive noise filtering. In the fusion procedure, the cost function of Eq. (2.10) needs to be calculated for each pixel for all the colors in the color palette, and thus it leads to a time complexity of $O(MN)$. The total time complexity of denoising additive Gaussian noise is therefore $O(k^2M^2 \cdot N/T)$.

In the case of the mixed noise, either the DUDE decision rule based statistical filtering, or the fusion process is applied and no additional cost is incurred. The time complexities are summarized in Table 2.1.

Table 2.1 Time Complexity of the Proposed Statistical Filter

Impulsive noise		Additive Gaussian noise	
Step	Complexity	Step	Complexity
Context-tree modeling	$O(kN)$	Color quantization	$O(MN)$
Context-merging	$O(k^2M^2 \cdot N/T)$	Context-tree modeling	$O(kN)$
Noise level estimation	$O(M)$	Context-merging	$O(k^2M^2 \cdot N/T)$
Statistical filtering	$O(N)$	Fusion procedure	$O(MN)$
Total	$O(k^2M^2 \cdot N/T)$	Total	$O(k^2M^2 \cdot N/T)$

2.3 ADAPTIVE FILTERING USING OPTIMIZED CONTEXT SELECTION

Several context-based approaches have been developed using fixed context templates [108], context tree modeling [89], or a context-merging strategy [**P2**]. However, these algorithms fail to reveal the local geometrical structures when the underlying contexts are contaminated by more than one noisy pixel. To address this problem, in [**P3**] we propose a novel context-based voting method to identify the possible noisy pixels, and these detected noisy pixels are excluded in the context selection for conditional probability estimation.

For example, when the context of a given pixel is contaminated by erroneous colors, it will be credited to a "*wrong*" context with rare appearance, which causes inaccurate estimation of the context distribution. This motivates us to investigate a criterion for context classification.

In [**P3**] all the contexts are categorized into three groups: *good*, *uncertain* or *bad* according to a *context efficiency function*:

$$\mathrm{F}(\mathbf{c}) = \log_2\left(\frac{P(\mathbf{c})}{P_E(\mathbf{c})}\right) + k\sum_x P(x|\mathbf{c})\log\left(\frac{P(x|\mathbf{c})}{P(x)}\right) \tag{2.11}$$

where: $P_E(\mathbf{c}) = \Pi_i P(y_i)$, $P(\mathbf{c})$ is the probability of a given context $\mathbf{c}$ and $P_E(\mathbf{c})$ is the estimated probability of $\mathbf{c}$, y_i is the color of i^{th} element in a

given context **c**, and $P(y_i)$ the probability of color y_i in the image. $P_E(\mathbf{c})$ is computed by assuming all elements in the context are mutually independent.

In the sense of image compression, the first term on the right-hand side of (2.11) can be interpreted as the difference of the context code length achieved according to the actual context probability and the expected context probability. Higher values for this term indicate that context **c** is a repetitive structure, and thus it can be used as a direct filter when a *dominant color* exists. The second term is the so-called *Kullback-Leiber distance* between conditional probability and color probability of the entire image. Larger distances imply that more bits can be saved when context **c** is used in coding.

Those frequent contexts associated with a dominant color (e.g., conditional probability > 90%), are termed as good context, on which filtering can be directly applied. Those contexts with rare appearance, which include noise elements, are defined as *bad* contexts because estimation of conditional probability under such contexts is inaccurate. Accordingly, all the contexts are categorized into three groups:

- good: $F(\mathbf{c}) > T_{max}$, with a dominant color
- bad: F(c) < Tmin
- uncertain: otherwise

Once all the contexts have been categorized into these three classes, the context tree is processed by identifying the good and bad nodes. The offspring nodes of any good or bad node will be removed in the tree pruning using top-to-bottom tracing. Three context examples are shown in Fig. 2.4. Two *good* contexts with *dominant colors* of black (left) and brown (middle) can be directly used for filtering, while the bad context on the right contains a noisy pixel. Fig. 2.9 shows an example of good and bad context distribution in a test image. The unmarked pixels belong to an uncertain context.

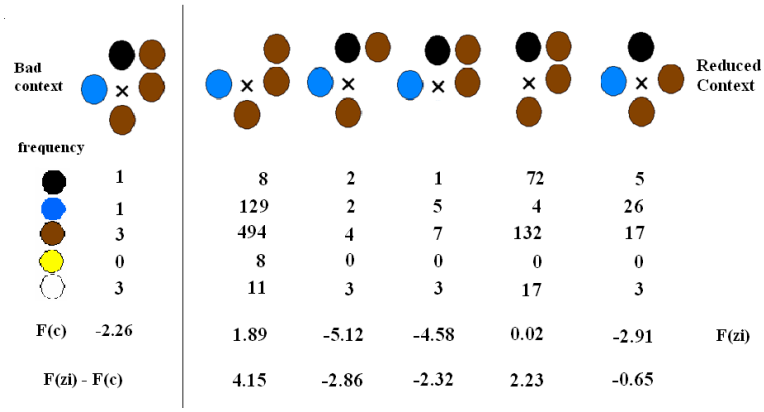

Fig. 2.8 *Bad context and its reduced context, black pixel is possible a noisy pixel with high $F(z_i)$ - $F(c)$ difference*

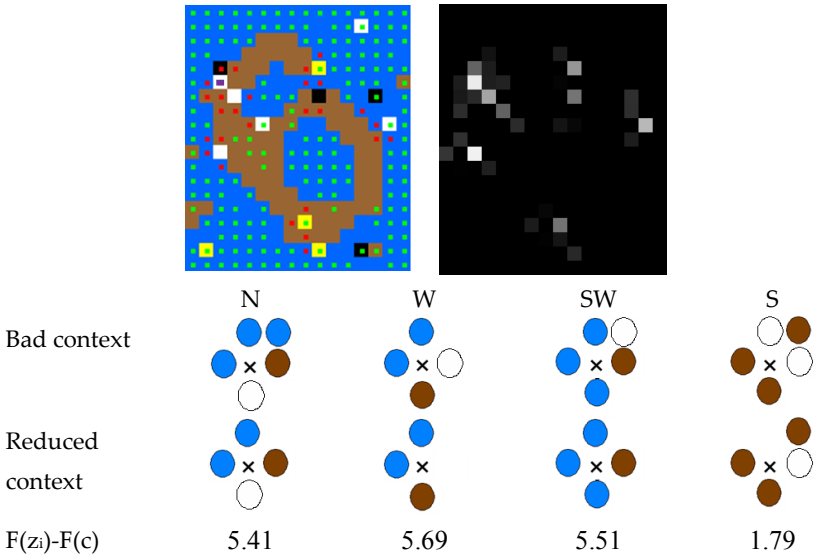

Fig. 2.9 *Sample image with good and bad context demonstrated in red and green colors (top left), its voting image (top right). Voting example for the white pixel labeled with purple with accumulated $F(z_i)$-$F(c)$ value 18.30 (bottom).*

As most of the bad contexts contain noisy pixels in themselves, they are seldom used to estimate a statistical model. However, they can be very useful in detection of noisy pixels when a noisy pixel is not isolated from most of the inherited geometrical structures. Given a bad context **c**, a set of sub-contexts is constructed in a similar way with Eq. (2.6), where $\mathbf{z}_i$ is the reduced-size context after removal of the i^{th} pixel.

If the removed pixel x_i is a noisy pixel, it is expected that the reduced-size context will have a higher efficiency: $F(\mathbf{z}_i) > F(\mathbf{c})$, such that each pixel in $\mathbf{c}$ can be assigned with a meaningful value $F(\mathbf{z}_i) - F(\mathbf{c})$. The higher the difference, the more likely x_i is to be a noisy pixel. Figure 2.8 gives an example of bad context and its reduced context.

A voting image $\mathbf{R}$ is then constructed according to the following rule: if the context $\mathbf{c}$ is detected as bad context, the accumulated voting score of every other pixel x_i in the same context $\mathbf{c}$ can be updated by:

$$R(x_i) = R(x_i) + F(\mathbf{z}_i) - F(\mathbf{c}) \tag{2.12}$$

Intuitively, we may conclude that most of the contexts containing noisy pixels may be detected as bad contexts. If the noisy pixel is removed from the bad context, the reduced-size context will have much better context efficiency. Namely, the accumulated voting score according to Eq. (2.12) is significantly higher than those of its neighborhood pixels. In this sense, the noisy pixel can then be detected by finding the high peak points in the voting image. An example of a voting scheme is shown in Fig. 2.9.

Once the voting image has been obtained, an adaptive context-based filter is applied in two manners. Firstly, if the context $\mathbf{c}$ is good, statistical filtering is applied directly by the decision rule in Eq. (2.4). Secondly, if x is detected as a noisy pixel in the voting scheme and its context $\mathbf{c}$ is not good, the context is re-selected adaptively excluding those noise pixels using a 3×3 context template. Statistical distribution of the adaptive context is collected and the DUDE framework is then applied. An example of the adaptive context selection can be seen in Fig. 2.10 and a filtering example is given in Fig. 2.11.

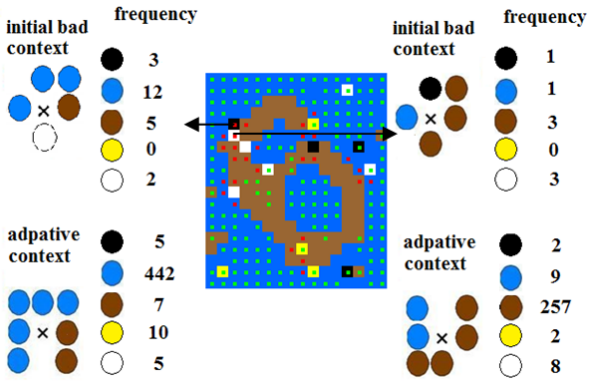

Fig. 2.10 *Example of adaptive context selection. For noise pixels (black and white, with high voting value), a new context in 3 × 3 region excluding surrounding noise pixels is selected, statistical information is collected for new contexts, black pixel is correctly changed to blue while white pixel changed to brown.*

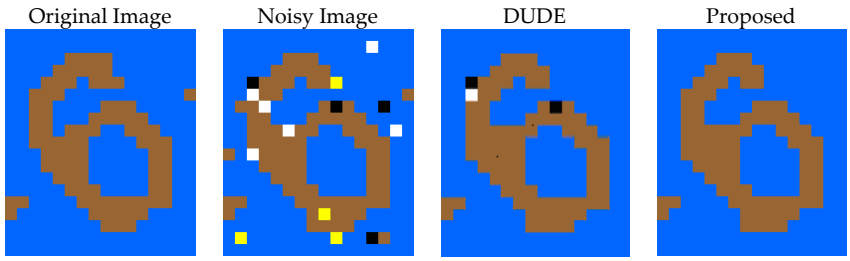

Fig. 2.11 *Example of voting-based adaptive filtering*

2.4 SUMMARY

In conclusion, in [**P1**] we have proposed a multi-layer approach filtering algorithm for raster map images. The proposed method provided a solution for processing map images in a binary domain. It has lower computation costs and memory consumption comparing with statistical methods.

In [**P2**] we have proposed a statistical filtering algorithm dealing with map images distorted by impulsive noise, additive Gaussian noise, and mixed Gaussian-impulsive noise. The proposed filter incorporates an information fusion process that exploits both the color distribution in RGB space, and the conditional probabilities of a given pixel in a local context. It operates with no prior knowledge of the

properties of the noise and aims at maximal preservation of repetitive structures of the image. This is an essential property for raster map images and is expected to generalize to other types of color-indexed imagery as well.

In [**P3**] we have extended the algorithm in [**P2**] and proposed an adaptive filtering algorithm using optimized context selection, which is designed via a novel voting-based noise estimation scheme.

The proposed context-based filter can be viewed as a pilot study to restore the raster map image distortion caused by uncertain noise. This algorithm can also be applied to other problems, such as image segmentation and color quantization.

3. *Simplification of GPS Trajectories*

Spatial-temporal data exist in many areas, such as: geographic information systems (GIS), location-based services (LBS), computer graphics and computational geometry. The processing and analysis of spatial-temporal data has been discussed widely in the literature [70].

In this chapter, we discuss the problem of GPS trajectory simplification in location-based services.

3.1 BACKGROUND

Location-acquisition technologies, such as geo-positioning mobile devices, enable users to obtain their location and record travel experiences by a number of time-stamped trajectories. In location-based web services, users can record, upload, visualize and share those trajectories [40, 41, 112]. Therefore, people are more likely to find the travel routes that interest them and acquire reference knowledge facilitating their travel from other trajectories.

However, GPS devices usually record far more data points than necessary and these redundant data points will decrease the performance of the data collection. For example, if data are collected at 10 second intervals, a calculation in [79] shows that without any compression, 100 Mb of storage capacity is required to store just 400 objects for a single day. Moreover, these redundant GPS trajectories will also cause a longer uploading/downloading time to the mobile service providers. The dense representation will also place a heavy burden on a web browser in client-side rendering of these trajectories. In some cases, web browsers may even run out of memory and crash. Therefore, a fast polygonal approximation algorithm is needed for the trajectories simplification task, i.e., conducting multiple GPS trajectory

simplifications corresponding to different map scales beforehand, such that the trajectories can be visualized more efficiently.

This can be considered as a polygonal approximation problem in 2-dimensional space. Given a polygonal curve $P = (p_1, \ldots, p_n)$ with N points, the problem of polygonal approximation is to seek an ordered subset P':

$$P' = (p_{i_1}, p_{i_2}, \ldots, p_{i_m}) \tag{3.1}$$

as an approximation of P, where $1 = i_1 < \ldots < i_m = N$. A polygonal approximation can be categorized into two classes of sub-problems:

a) *min-ε problem*: given N-points, polygonal curve P, and integer M, approximate a polygonal curve P' with at most M points, which has the minimum approximation error.

b) *min-# problem*: given N-points, polygonal curve P, and error tolerance ε, approximate a polygonal curve P' with the minimum number of points within the error tolerance ε.

In [**P4**] we present a fast $O(N)$ time polygonal approximation algorithm for the GPS trajectory simplification. The proposed method applies a joint optimization for both *min-# approximation* and *min-ε approximation*. The main contributions are summarized as follows:

First, we extend the *local integral square error* criterion (LISE) and the *integral square error* criterion (ISE) [26, 86, 92] and derive two new error measures for the GPS trajectory simplification problem, called *local integral square synchronous Euclidean distance* (LSSD) and *integral square synchronous Euclidean distance* (ISSD). The main advantage of LSSD and ISSD is that time information is also considered in the approximation process. Meanwhile, they have the same properties with LISE and ISE, i.e., they can be obtained efficiently in $O(1)$ time after pre-calculating all the accumulative terms within $O(N)$ time.

Second, for each resolution (map scale), an initial approximated curve is constructed by a combination of a priority queue structure [29-31] and a stopping search criterion [55], which leads to $O(N^2/M)$ time complexity and $O(N)$ space complexity. However, using a stopping search criterion will cause a trade-off in the optimality.

Third, we extend the reduced search algorithm and employ two fine-tune strategies to minimize both the number of output points M and the approximated error ε, which lead to time complexities

$O(WN^2/M)$ and $O(N^2/M)$, respectively where W is the width of the bounding corridor.

The algorithm is implemented by a *bottom-up multi-resolution* approach and it achieves a linear time and space complexity. The proposed GPS trajectory simplification algorithm can be used in a real-time application for the rendering process of the GPS trajectories on the map.

3.2 ERROR MEASURE: FROM LISE TO LSSD

The primary goal of the GPS trajectory simplification techniques is to reduce the data size without compromising the precision. Thus, there is a need to find appropriate error measures for use in the algorithms and performance evaluation.

In polygonal approximation, different error criteria have been defined, such as: *tolerance zone*, *parallel-strip*, *uniform measure, minimum height* and *minimum width* [17, 26, 50, 78, 86, 105].

However, Meratnia [79] indicated that such algorithms were not suitable for GPS trajectory because both spatial and temporal information should be considered. Therefore, the errors are measured through distances between pairs of temporally synchronized positions, which are called the *synchronous Euclidean distance* (SED). In synchronized Euclidean distance, the continuous nature of moving objects necessitates the inclusion of temporal, as well as spatial properties of the objects.

In addition to the two error measures described above, other error functions are also considered in some literature. For example, position, speed and orientation information are all used in the *threshold-guided algorithm* [91]. In [16] a new distance function called *spatial join* was proposed, which was bounded for spatial-temporal queries. In the area of shape matching, Fréchet distance [5, 6] also takes the continuity of shapes into account with a time complexity $O(MN)$. Note that topological and geometric properties are also considered as important constraints in the simplification process in GIS applications. In [36] simple Detours (SD) heuristic was proposed, where no new (Steiner) vertices would be introduced after the approximation process.

However, in most algorithms, in order to calculate the approximated error of line segment $\overline{p_i p_j}$, at least j - i distance calculations are needed. In [30] the calculation process was solved in dual space by a priority queue structure, which achieved the best processing time $O(\log N)$ with a preprocessing time $O(N\log N)$.

In order to improve the computational efficiency, two error measures, which are called *integral square error* (ISE) and *local integral square error* (LISE) [26, 27, 86, 92], are jointly used in this paper for approximating polygonal curves. The main advantage of the integral square error criteria is that the approximation error $\delta(P_i^j)$ can be obtained efficiently in $O(1)$ time after pre-calculating all the accumulative terms within $O(N)$ time (see equation 3.6) [27, 86]. An example of calculating ISE and LISE is illustrated in Fig. 3.1.

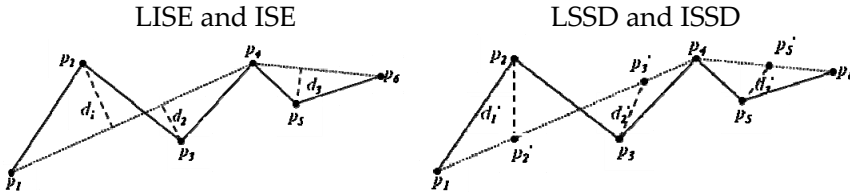

Fig. 3.1. *An example of calculating ISE, LISE, LSSD and ISSD. Given P = (p_1, p_2, p_3, p_4, p_5, p_6), and the approximated curve P'= (p_1, p_4, p_6), where p_2', p_3' and $p_5'$are the approximated temporally synchronized position. ISE is estimated as $d_1^2+d_2^2+d_3^2$ and LISE is estimated as $d_1^2+d_2^2$ (left). Meanwhile, ISSD is estimated as $d_1'^2+d_2'^2+d_3'^2$ and LSSD is estimated as $d_1'^2+d_2'^2$(right).*

Although the LISE and ISE criteria are computational efficient, time information is not considered. Therefore, for the simplification of the GPS trajectories, we extend LISE and ISE criteria and derive two new error measures, called *local integral square synchronous Euclidean distance* (LSSD) and *integral square synchronous Euclidean distance* (ISSD), which have the same properties as LISE and ISE:

$$f_{ISSD}(P') = \sum_{j=1}^{M-1} \delta_{SED2}(P_{i_j}^{i_{j+1}}) \tag{3.1}$$

$$f_{LSSD}(P') = \max_{1 \le j < M} \delta_{SED2}(P_{i_j}^{i_{j+1}}) \tag{3.2}$$

$$\begin{aligned} &\text{where } \delta_{\text{SED2}}(P_i^j) = \sum_{i<k<j} SED^2(p_k, p_k^{'}) \\ &= (c_1^2 + c_3^2)(j-i-1) + (c_2^2 + c_4^2)(S_{t^2}^{j-1} - S_{t^2}^i) \\ &+2(c_1c_2 + c_3c_4)(S_t^{j-1} - S_t^i) + (S_{x^2}^{j-1} - S_{x^2}^i) \\ &+(S_{y^2}^{j-1} - S_{y^2}^i) - 2c_1(S_x^{j-1} - S_x^i) - 2c_3(S_y^{j-1} - S_y^i) \\ &-2c_2(S_{xt}^{j-1} - S_{xt}^i) - 2c_4(S_{yt}^{j-1} - S_{yt}^i) \end{aligned} \tag{3.3}$$

Here $c_1 = \dfrac{x_i t_j - x_j t_i}{t_j - t_i}$, $c_2 = \dfrac{x_j - x_i}{t_j - t_i}$, $c_3 = \dfrac{y_i t_j - y_j t_i}{t_j - t_i}$, $c_4 = \dfrac{y_j - y_i}{t_j - t_i}$

S_x, S_y, S_t, S_{x2}, S_{y2}, S_{t2}, S_{tx} and S_{ty} are the accumulated sums of x, y and t on the GPS trajectory respectively:

$$\begin{aligned} &S_x^i = \sum_{j=1}^{i} x_j,\ S_y^i = \sum_{j=1}^{i} y_j,\ S_t^i = \sum_{j=1}^{i} t_j,\ S_{x^2}^i = \sum_{j=1}^{i} x_j^2, \\ &S_{y^2}^i = \sum_{j=1}^{i} y_j^2,\ S_{t^2}^i = \sum_{j=1}^{i} t_j^2,\ S_{tx}^i = \sum_{j=1}^{i} t_j x_j,\ S_{ty}^i = \sum_{j=1}^{i} t_j y_j \end{aligned} \tag{3.4}$$

Computation of the above approximation errors $\delta_{\text{SED2}}(P_i^j)$ also takes $O(1)$ time with an $O(N)$ time accumulated sum pre-calculation. The proof of the LSSD/ISSD calculation can be seen in [**P4**]. LSSD and ISSD criteria are used for the GPS trajectory simplification.

3.3 NEAR-OPTIMAL POLYGONAL APPROXIMATION

Optimal polygonal approximation methods are generally implemented by incrementally constructing a *directed acyclic graph* (DAG), and therefore inevitably suffer a computational cost limitation of $O(N^2)$ at minimum[17, 20, 26, 45, 50, 78, 86, 98, 105]. An advance achieved in [1] is to combine an iterative graph algorithm and a divide-and-conquer approach, which offers the best time and space complexity of $O(N^{4/3+\delta})$ by using the L_1 *metric*, where $\delta > 0$ is an arbitrarily small constant. Later, the graph-based framework has been

significantly re-organized and optimized by using *priority queues* dynamically [30].

In a real-time application, quadratic time complexity is too high, and therefore, most applications utilize a class of heuristic methods in order to achieve near-linear time complexity. A set of well-known heuristic algorithms are the *split* and *merge* approaches [34, 48, 87]. The split algorithms divide the segment causing the biggest deviation, and the merge algorithms merge the pair of segments with least deviation. The classic *Douglas–Peucker* split algorithm [34] can be implemented in $O(N\log N)$ time on average. Later Hershberger [48] showed that it can be implemented in $O(N\log^* N)$ time, where log* denotes the iterated logarithm function. Respectively, Pikaz [87] proposed a merging algorithm with $O(N\log N)$ time complexity. These heuristic methods are of low time complexity, but may lead to an undesirable approximation result.

In the GPS trajectory simplification, a number of algorithms have also been well studied and developed. In [79] a trajectory simplification algorithm is implemented greedily by a so-called *opening window* approach. *Synchronous Euclidean distance* is also defined and applied by incorporating the time dimension instead of the original perpendicular distance. In [91] the parameters including coordinates, speed, and orientation are all considered in calculating the safe area of the next point, which is called the *threshold-guided algorithm*. Indeed, all these algorithms solve the min-# problem in a greedy manner, of which the time complexity is $O(N^2)$. The *STTrace sampling algorithm* [91] is also implemented using a bottom-up strategy where the synchronous Euclidean distance is minimized in each step. In [68] a *generic remote trajectory simplification protocol* (GCTS) combined optimal and heuristic algorithms [50, 79], which allowed a trade-off of computational complexity against reduction efficiency. Recently, a new simplification algorithm SQUISH [82] was proposed based on the priority queue data structure, which preserved speed information at a much higher accuracy. In [22] a *trajectory simplification* algorithm is proposed, where different point headcounts are assigned in terms of the product of the average heading change and the distance of each segment. After that, the min-ε problem is solved in each segment by using a local weighting process in $O(N\log M)$ time. However, as the

distances of neighborhood points are used instead of the perpendicular distance in the simplification procedures, the algorithm is not robust when the sampling frequency is non-uniform. Moreover, the error of the GPS signals may also cause the inaccurate estimation both in the trajectory segmentation and the local weighting process. Performance evaluations are made for several traditional trajectory simplification algorithms in [81]. However, it is noted in [82] that there is not one algorithm that always outperforms other approaches in all situations.

Note that except for the linear approximation of GPS trajectories, non-linear methods are also considered in the literature, such as: Bézier curves [24], spline interpolation [53], clothoid [67] and Chebyshev polynomials [83].

3.3.1 Min-# Initialization

For the *min-# problem*, Imai and Iri's graph-based approach [50] comprises two essential steps: constructing DAG and shortest path search by *breadth-first traversal* (BFT). In order to construct DAG, $N(N-1)/2$ approximation errors are calculated for every pair of vertices, which is called as *edge tests* here. Thus, the time complexity for the min-# problem is $O(N^2)$ using the LISE or LSSD criterion. In **[P4]** we revisit two computationally efficient improvements for the min-# problem.

The first improvement is to reduce the computational cost of DAG construction by maintaining two *priority queue* structures, of which one corresponds to vertices that can be reached via $k-1$ links, and the other corresponding to vertices reachable via k links [29-31]. The reason for this is that there is no need to construct the graph G explicitly, and that only edges visited by the BFT should be included. Although the priority-queue based search is not able to completely mitigate the worst case time complexity, it turns out in the experiments that a number of edge tests were saved in the practical implementations.

The second improvement is to apply a stopping criterion in the iterative shortest path search, which is very efficient in the case of low error tolerance. For example, a good stopping criterion has been proposed for *tolerance zone criterion* [20] by maintaining two

intersection cones. An alternative solution has also been proposed in [29-31] by verification in dual space. Both of the implementations hold the optimality for solving the min-# problem. To pursue the best possible computational cost savings for the LISE/LSSD criteria, a simple stopping criterion is applied in the edge tests by utilizing a preset *high threshold,* e.g., two times the given tolerance [55]. Edge tests for the subsequent vertices in the un-visited vertices set will be omitted once the approximation error becomes larger than a given high threshold. Applying stopping criteria achieves a significantly improved time complexity of $O(N^2/M)$, but the optimality is not guaranteed. To overcome this difficulty, we extend our effort in improving the robustness of the stop search criterion. Instead of using a fixed high threshold, we adopt the approximation error tolerance of the next coarser resolution as the high threshold in the multi-resolution implementation; see Section 3.4 for more detailed discussion.

We combine both the advantage of the priority queue structure and the stopping criterion to achieve the most computationally efficient implementation in the initialization of the min-# problem. Accordingly, the output of the initialization of the min-# problem is a tree structure; see Fig. 3.3 (left). The proposed initialization algorithm for solving the min-# problem under the LISE/LSSD criteria leads to an expected time complexity of $O(N^2/M)$ and a space complexity of $O(N)$, respectively.

3.3.2 Fine-tuning the Initial Approximation

For the GPS trajectory simplification, the optimal algorithm provides the best reduction efficiency, but causes the highest overhead, while solutions based on heuristics lower the computational overhead at the cost of worsening reduction rates. A compromise between the optimal and heuristic solutions is the *reduced search dynamic programming* [55, 56, 61].

The algorithm uses a *bounding corridor* surrounding a reference curve or initialized curve in the state space (Fig. 3.2), followed by a limited search for the minimum cost path. This idea is presented and known as the Sakoe-Chiba band [96], which has been used extensively in *Dynamic Time Wrapping* (DTW) approaches dealing with the similarity calculation of time-series [32].

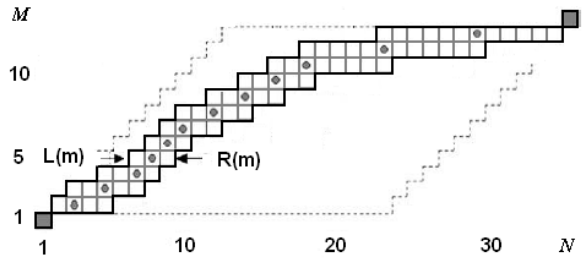

Fig. 3.2. *An example of the reduced search dynamic programming with N = 35 and M = 14, where the width of the bounding corridor W = 2. Reference curve is represented with the gray circles while the full state space is surrounded by the dashed line.*

In reduced search dynamic programming, if the initialized curve is evenly distributed in the state space, the time complexity for RSDP is ideally $O(W^2N^2/M^2)$. In [**P4**] we also prove that the expected time complexity is still achievable as $O(W^2N^2/M^2)$, even if the precondition of even distribution is not satisfied. In particular, if the number of vertices for the approximated curve is proportional to that of the input curve, namely, $M = N/c$, a linear time complexity can be achievable for RSDP. This will be later shown to be an important property when selecting *bottom-up* approaches for the multi-resolution case. However, the main difficulty with RSDP is that a large corridor bound and many iterations are needed in order to achieve a desirable solution when the approximated curve is poorly initialized, which causes a high computational cost.

To the benefit of best computational efficiency, the initialization in section 3.3.1 for the min-# problem is a compromise of the optimality for minimizing the number of vertices. In order to mitigate the limited optimality, we minimize the number of vertices based on the initialized curve, so that a better result can be achieved. The reduced search algorithm can be utilized for minimizing the number of vertices. To speed up the procedure, we exploit a new fine-tune method at a time complexity of $O(WN^2/M)$ instead, which is achieved by lifting the vertex position in the output tree structure after the initialization step.

A graphical illustration is demonstrated in Fig. 3.3 of lifting the vertex position: starting from vertex p_1 with 0 links, at each iteration, edge tests are performed to verify whether the approximation error is less than the given tolerance between the currently processed vertices

with k links, and those target vertices with $\{k + 2, \ldots, k + W + 1\}$ links. An example is given in Fig.3.3 (left) when the width of the bounding corridor is $W = 2$. The process of updating the tree structure can be done recursively as shown in Fig. 3.3 (right).

The proposed fine-tune algorithm provides the following advantages over the original reduced search approach for the min-# problem. Firstly, calculation of the approximated errors between any pair of vertices with adjacent number of links is unnecessary and can be omitted. Secondly, once the tree structure is updated by the lifting operations, edge tests for those lifted vertices are also avoided.

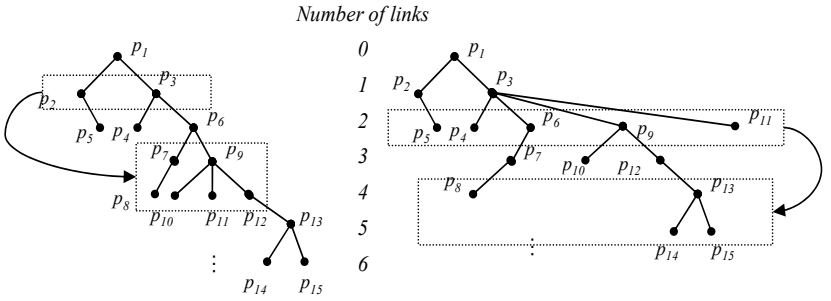

Fig. 3.3 *An example of reducing the number of output vertices with bounding corridor $W = 2$: the target vertices with 1 links (left, which is a typical example after the initialization step) and target vertices with 2 links after tree structure updated (right).*

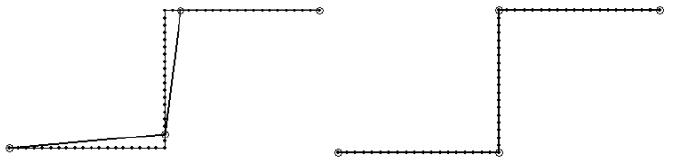

Fig. 3.4. *An example of equivalent solutions in min-# approximation, where both approximated curves meet the error tolerance $\varepsilon = 2$ and have same output $M = 4$.*

The proposed algorithm for the output vertex reduction under the LSSD criterion has an expected time complexity of $O(WN^2/M)$ and space complexity of $O(N)$, respectively.

However, an *equivalent solution problem* may exist even after the min-# problem is solved by the LSSD criterion. In other words, given an error tolerance ε, a number of solutions for the min-# approximation can be achieved with the same number of output

vertices M, but they lead to distinct approximation performance, see Fig. 3.4.

Hence, after the tree structure is updated, additional post-processing is performed in order to identify the best possible curve P', among these equivalent solutions with the minimum ISSD. This can be can be solved by dynamic programming in terms of the following recursive expression:

$$\begin{aligned} &D(p_j) = \min(D(p_i) + \delta_{SED2}(P_i^j)), 1 \le i < j \\ &A(p_j) = \arg\min_i (D(p_i) + \delta_{SED2}(P_i^j)), 1 \le i < j \\ &s.t.\ \delta_{SED2}(P_i^j) < \varepsilon, L(p_j) = L(p_i) + 1 \end{aligned} \tag{3.5}$$

where $A(p_j)$ is the parent vertex of p_j and $D(p_j)$ is the accumulated ISSD. The minima are found in $O(N^2/M)$ time and no iterations are needed. The above minimization offers significant improvement (theoretically W^2 time faster) over the original RSDP that has a time complexity of $O(W^2N^2/M)$.

In summary, a polygonal approximation algorithm for GPS trajectory simplification is proposed by joint optimization of both the min-# approximation using the LSSD criterion, and the min-ε approximation using the ISSD criterion. The proposed algorithm has been introduced as a three step procedure: initialization of the min-# problem, minimizing the number of output vertices, and minimizing integral square error. The proposed algorithm has $O(N^2/M)$ expected time complexity and $O(N)$ space complexity. An example of the proposed algorithm is shown in Fig. 3.5. The improvement of time complexity is also summarized in Table 3.1.

Table 3.1 Summary of the proposed near-optimal polygonal approximation algorithm, where * represents the initial curve is equally partitioned

Step	Time complexity		Improvements and contributions
	RSDP	Proposed	
I	$O(N^2/M)$	$O(N^2/M)$	Combine priority queue structure to reduce the computation cost. Proof is given.
II	$O(W^2N^2/M)$*	$O(WN^2/M)$	Time complexity reduced. Proof is given.
III	$O(W^2N^2/M)$*	$O(N^2/M)$	Time complexity reduced. Proof is given.

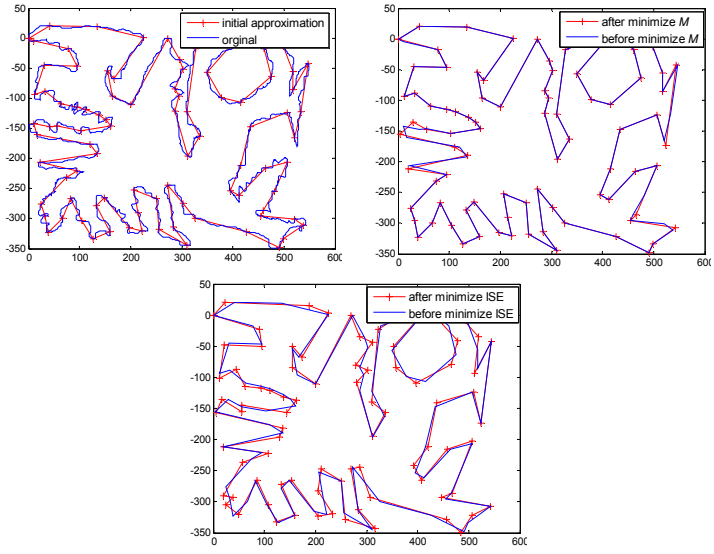

Fig. 3.5. *An example of the proposed polygonal approximation. Artificial curve in [43] is used with ε = 1500 and the optimal solution is M_{opt} = 86. Initial approximated curve is obtained with M′ = 91 (up-left). Approximated curve (M = 86) is obtained after reducing number of output vertices with $f_{ISE}(P') = 1.04 \cdot 10^5$ (up-right). The final solution is obtained by minimizing ISE with $f_{ISE}(P') = 4.88 \cdot 10^4$ (bottom).*

3.4 LINEAR TIME MULTI-RESOLUTION POLYGONAL APPROXIMATION

In order to improve the computational efficiency, a bottom-up multi-resolution polygonal approximation approach is proposed with linear time complexity in **[P4]**. The error criterion is the joint optimization on both the min-# approximation using the LSSD criterion and the min-ε approximation using the ISSD criterion, as in Section 3.3.

A multi-resolution polygonal approximation can be applied for scalable representation and compression of vector maps in GIS [12, 13]. For solving the min-ε problem, two heuristic approaches: split (top-down), and merge (bottom-up) are known with a time complexity of $O(NlogN)$. It is important to mention that split and merge are applied locally and can often result in undesirable approximation results in the later hierarchy process.

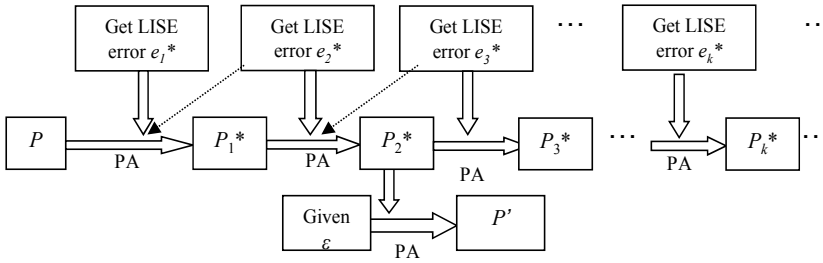

Fig. 3.6. *Workflow of the proposed bottom-up multi-resolution method. Error tolerance of coarser resolution is selected as high threshold for polygonal approximation, which is labeled by dashed line in the figure. In this example, with* $e_2^* < \varepsilon < e_3^*$ *the approximation of* P_3^* *and* P_4^**,… can be skipped.*

An *optimal split algorithm* (OSA) was proposed in [57], where the optimal approximation at the higher resolution level is achieved using the result of a lower (previous) resolution level. This provides resolution hierarchy in an order (1→2→4→….), but at a cost of $O(N^2)$ time complexity.

In [75] a bottom-up multi-resolution algorithm for the min-ε problem is proposed with near-linear time complexity. The min-ε problem is solved using the fine resolution as input for approximating the corresponding coarser resolution iteratively ($N \rightarrow N/2 \rightarrow N/4 \rightarrow \ldots$). For each scale, RSDP is also incorporated. As an ISE criterion is used,

the approximation error between two vertices in any resolution level can be calculated in a constant time, according to the pre-calculating cumulative summation of x, y, x^2, y^2 and xy in the original curve, see Section 3.2.

Although the bottom-up approach [75] is computationally efficient, this approach can only solve the min-ε problem. In practice, in order to progressively display the GPS trajectory data, we need to approximate a number of approximated results with corresponding error tolerance for each resolution (map scale), which is considered as a min-# problem. Moreover, the reduced search algorithm is a fine-tune method that needs an initial curve beforehand. If the curve is poorly initialized, a number of iterations must be needed to find the near-optimal result by the reduced search algorithm.

In [**P4**], given error tolerance ε, a joint optimization for both the min-# approximation using the LSSD criterion and the min-ε approximation using the ISSD criterion is solved in linear time. The underlying algorithm consists of three sequential procedures in each resolution:

I. Error tolerance initialization. Initialize $\log_c N$ error tolerances $\{e_1^*, e_2^*, e_3^*, \ldots\}$ $(e_1^* < e_2^* < e_3^* \ldots)$.
II. Initial curve approximation. A number of polygonal curves $\{P_1^*, P_2^*, \ldots, P_k^*\}$ are approximated based on a bottom-up multi-resolution approach with corresponding error tolerance $\{e_1^*, e_2^*, e_3^*, \ldots\}$. The algorithms of Section 3.3 are used for approximating the curve of each resolution.
III. Final approximation. A polygonal approximation is conducted under the given error tolerance ε by selecting the most suitable input curve amongst those approximated curves $\{P_1^*, P_2^*, \ldots, P_k^*\}$.

In step I, the error tolerances $e_1^*, e_2^*, e_3^* \ldots$ $(e_1^* < e_2^* < e_3^* \ldots)$ are estimated according to the LISE/LSSD error criterion:

$$
\begin{aligned}
e_k^* &= \frac{1}{N/c^k - 1} \sum_{1 \le j < N/c^k} \sum_{i_j < m < i_{j+1}} \delta_{SED2}(p_m, \overline{p_{i_j} p_{i_{j+1}}}) \\
i_j &= \frac{N-1}{N/c^k - 1} \cdot (j-1) + 1
\end{aligned}
\tag{3.6}
$$

where c is a parameter in the proposed multi-resolution approach. The above estimation can be viewed as the average LISE/LSSD error for all approximated segments when the curve is equally partitioned. The approximated curve under the error tolerance e_k^* has the property $M_k \approx N/c^k$, where M_k is the number of output vertices in the k^{th} resolution.

In step II, the *bottom-up multi-resolution* algorithm is applied to estimate the approximated curves P_1^*, P_2^*, P_3^*, … under the corresponding error tolerances e_1^*, e_2^*, e_3^*, …. Here, e_{k+1}^* is used as the high threshold in the approximation procedure of resolution k. The curve achieved in the previous finer resolution is used as the input of polygonal approximation in the next coarser resolution ($N_{k+1} = M_k$), and the algorithms from Section 3.3 are applied. As the optimality of these initial approximation results is not significantly compromised, the step of minimizing the number of vertices can be omitted.

In step III, given error tolerance ε, a polygonal approximation is conducted to get the final approximation result by selecting the most suitable inputs curve P_k^* among those approximated curves in step II, such that:

$$k = \arg\max_k (e_k^* < \varepsilon) \tag{3.7}$$

The workflow of the proposed algorithm is presented in Fig. 3.6. As the time complexity of the approximation process is $O(N_k^2/M_k)$ on each resolution, both the time complexity and the space complexity of the proposed bottom-up multi-resolution algorithm are $O(N)$.

An application of the proposed approximation algorithm for the GPS trajectory simplification is demonstrated in Fig. 3.7. The sample route with 575 vertices is visualized in different map scales with 44, 13, 6 vertices correspondingly in our MOPSI system. As a suitable error tolerance is selected for each resolution, the visualization of the GPS trajectory is not compromised by the reduced data, whereas the rendering time is greatly reduced.

3.5 SUMMARY

In conclusion, in [**P4**] we have proposed a fast $O(N)$ time polygonal approximation algorithm for the GPS trajectory simplification, by a

joint optimization on both LSSD and ISSD criteria, which is effective and very computationally efficient. The proposed method is designed by a bottom-up multi-resolution approach with a linear time complexity. In each resolution, a near-optimal polygonal approximation algorithm is exploited, which has a time complexity of $O(N^2/M)$. Both the theoretical analysis and the experimental tests have demonstrated that the proposed method has made significant progress in solving polygonal approximation in a real-time application. Moreover, the proposed polygonal approximation algorithm and fine-tune strategy can also be extended and exploited to other error criteria.

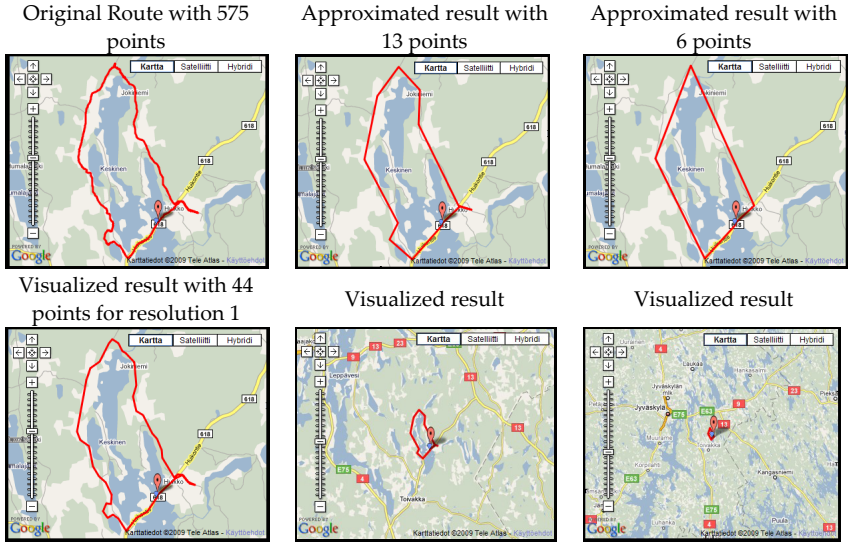

Fig. 3.7. *Example of the GPS trajectory simplification by the proposed algorithm.*

4. Compression of Vector Maps

A vector map can be formulated as a 2-dimensional vector sequence $P = (p_1, p_2, \ldots, p_n)$. It embraces a number of objects such as waypoints, routes and areas. A sample curve and the distribution of the corresponding differential coordinates are shown in Fig. 4.1.

A vector map consists of a large number of geographic objects, which cause a high data storage cost. Therefore, a variety of compression algorithms has been studied and developed [2, 3, 58, 60, 72, 101, 102, 110]. Existing algorithms are based on two strategies: *polygonal approximation* and *quantization*.

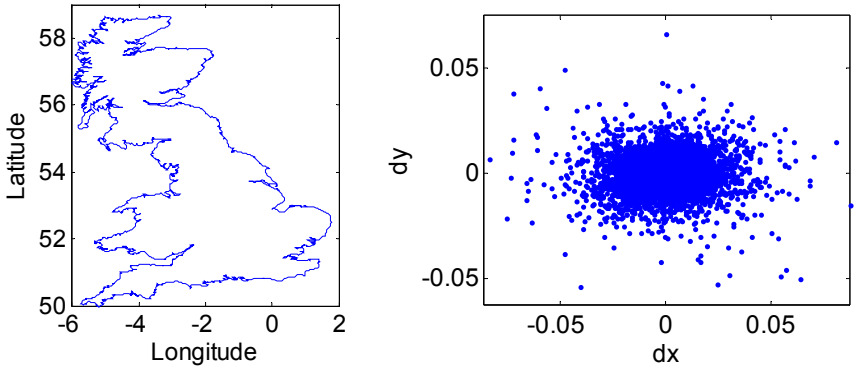

Fig.4.1. *Test map Britain with 10,910 points (left), and the prediction residuals.*

In polygonal approximation, the number of points in the vector map is reduced so that the polygonal curve is represented in a coarser resolution. Polygonal approximation can be achieved either by fast heuristic methods [9, 76], or by graph-based methods [86, 98].

On the other hand, most quantization-based approaches calculate the differential coordinates of adjacent data points as the prediction error, and these residual vectors are then quantized using different quantization strategies, such as: *product uniform quantization* [72],

product scalar quantization [2], and *vector quantization* with fixed-size codebook [102].

A better rate-distortion performance is achieved later by combining both the advantage of polygonal approximation and prediction error quantization. For instance, *reference line* method [3] first identifies a series of reference lines by polygonal approximation. For the remaining points, the prediction errors are then estimated according to their nearest reference lines, followed by *product scalar quantization* in a similar manner to [2]. Likewise, in [110], a number of data points were first reduced by the *Visvalingam-Whyatt* algorithm to preserve a consistent topology, and were subsequently quantized and encoded by a clustering-based method.

4.1 FAST DYNAMIC QUANTIZATION

In [58], d*ynamic quantization* (DQ) is proposed by performing a joint optimization on both polygonal approximation and vector quantization. For a given quantization level l, product uniform quantization is employed in the joint optimization by a given *Lagrangian* parameter λ. Thus, a rate-distortion curve corresponding to the given quantization level l, will be constructed by traversing different *Lagrangian* parameters λ.

A common practice for data compression encompasses three essential procedures: prediction, quantization of the residual vectors, and entropy coding. In vector map compression, the prediction procedure calculates the differential coordinates of adjacent points as a prediction error, instead of using the absolute coordinates. After these residual vectors are quantized in the coding process, we assumed that they obeyed an empirical distribution of a random variable, e.g., *uniform distribution*, *geometric distribution*, *negative binomial distribution*, or *Poisson distribution*.

To avoid quantization error propagation, the prediction is done using closed-loop prediction:

(4.1)

where: Q is a 2-D *product uniform quantizer*, $\mathbf{v}_i$ is the residual vector, and p_{i-1}^r is the estimated position of the previous point. For a given quantization level l, the product uniform quantizer is formulated as:

$$Q(\mathbf{v}_i) = [\mathbf{v}_i / l] \cdot l = ([\Delta x_i / l] \cdot l, [\Delta y_i / l] \cdot l) \tag{4.2}$$

Thus, the relation between the quantization error D_0 and quantization level l is solved, which gives $D_0 = l^2/6$.

Coding $Q(\mathbf{v}_i)$ is equivalent to coding an integer vector $\mathbf{q} = ([\Delta x_i/l], \Delta y_i/l])$, which can be encoded by probability distributions of q_x and q_y.

$$r(\mathbf{v}_i) = -\log_2 f(q_{xi}) - \log_2 f(q_{yi}) \tag{4.3}$$

Note that the codebook itself also needs to be encoded and transmitted to the decoder. Thus, a large codebook is intractable in order to achieve a desirable coding efficiency. In [**P5**] we use a single-parameter *geometric distribution* to model $|q_x|$ and $|q_y|$:

$$f(|q_x|) = (1 - p_x)^{|q_x|} p_x \tag{4.4}$$

where parameters p_x and p_y are approximated by their maximum likelihood estimation. Therefore, no codebook is needed and the code length achieved by arithmetic coding can be written as:

$$\begin{aligned} r(\mathbf{v}_i) = &-(|q_{xi}| \log_2(1 - p_x) + \log_2(p_x)) + 2 \\ &-(|q_{yi}| \log_2(1 - p_y) + \log_2(p_y)) \end{aligned} \tag{4.5}$$

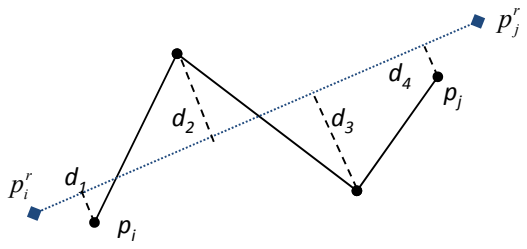

Fig. 4.2. *Poly-line (p_i,…, p_j) (solid line) is approximated by line segment (p_i^r, p_j^r) (dot line) in the approximation process.*

Meanwhile, polygonal approximation is also embedded into the closed-loop framework in dynamic quantization. Suppose that a polygonal curve (p_i, … , p_j) is approximated by line segment (p_i^r, p_j^r), local integral square error (LISE) [26, 86] is used as the approximated error of the approximation process, see Fig. 4.2.

$$e_2(p_i^r, p_j^r) = \sum_{k=i}^{j} d^2(p_k, (p_i^r, p_j^r)) \tag{4.6}$$

The approximation error of Eq. (4.6) can be calculated in constant time by pre-computing the accumulated sums of x^2, x, xy, y^2 and y [26, 86]. In this way, a joint optimization of polygonal approximation and prediction error quantization can be considered as minimizing the following cost function:

$$J = E_2 + \lambda R = \sum_{m=1}^{M} (e_2(p_{i_m}^r, p_{i_{m+1}}^r) + \lambda \cdot r(p_{i_m}^r, p_{i_{m+1}}^r)) \tag{4.7}$$

where M is the number of points output by polygonal approximation. This problem can be solved by finding the shortest path on a weighted directed acyclic graph (DAG). Suppose J_i is the minimum weighting sum from p_1 to p_i on G, and A is an array used for backtracking operation, then the recursive formulation of the problem is:

$$\begin{aligned} J_i &= \min_{\{1 \le k \le i-1\}} (J_k + e_2(p_k^r, p_i^r) + \lambda r(p_k^r, p_i^r)), J_1 = 0 \\ A_i &= \arg\min_{\{1 \le k \le i-1\}} (J_k + e_2(p_k^r, p_i^r) + \lambda r(p_k^r, p_i^r)) \end{aligned} \tag{4.8}$$

In the existing approach [58], all the rate-distortion curves with respect to each of the quantization levels are calculated so that the best quantizer is the *lower envelope* of the set of curves. However, the computation cost of the entire set of rate-distortion curves is hugely time-consuming. For example, a unique rate-distortion curve can be constructed by considering 30-40 *Lagrangian* parameter λ. It would lead to over 300 minimization processes of Eq. (4.7) by selecting different quantization levels.

To overcome this difficulty, an *error balance principle* is proposed in [60] based on an assumption that the quantization errors are equal to

the approximation error of *polygonal approximation*. Thus, an optimal number of points M, can be identified in the min-ε polygonal approximation using binary search for a given quantization level l. However, in practice, the time complexity for the *min-ε polygonal approximation*s is equal to $O(N^2)$ as well [98].

In order to reduce the computational cost, in [**P5**] a *fast dynamic quantization* (FDQ) *algorithm* is proposed. For a given quantization level l, its corresponding optimal *Lagrangian* parameter λ is derived, which is:

$$\lambda \approx \frac{1}{6} l^2 \ln 2 \tag{4.9}$$

Thus, only one rate-distortion curve needs to be constructed. An example is shown in Fig. 4.3.

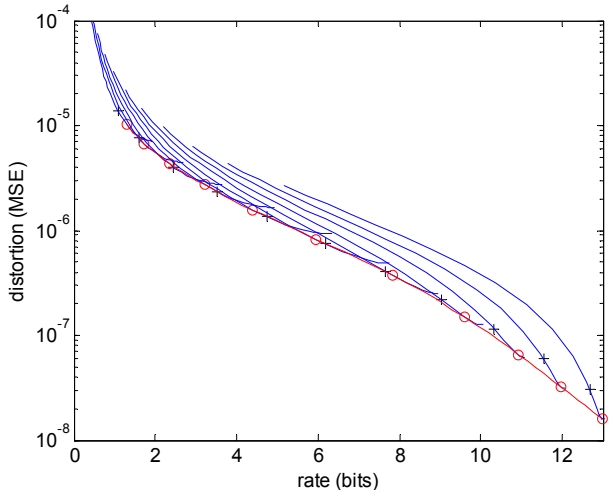

Fig. 4.3. *Rate-distortion curve for quantization step q_k=0.01/2^k, where k = 0, 1/2, 1,… 5 (from left to right), black '+' is the position when error balance principle is applied, red 'o' is the proposed. The red line is the rate-distortion curve when optimal λ is selected.*

The shortest path algorithm on a weighted DAG takes $O(N^2)$ time at minimum [17, 20, 26, 45, 50, 78, 86, 98, 105]. This can be further improved by incorporating a specific search criterion:

$$\sqrt{\frac{e_2(p_k^r, p_i^r)}{(i-k)}} > \tau \sqrt{\frac{e_2(p_{A_i}^r, p_i^r)}{(i-A_i)}} \tag{4.10}$$

where $(p_{A_i}^r, p_i^r)$ is the shortest path so far and p_k^r is the current point. For a target point p_i^r, the search for the shortest path will terminate the weight calculation before point p_k if Eq. (4.10) is satisfied. The proposed method can also be applied for the entropy-constrained problem, in which the vector map is compressed under a certain bit-rate. The result can be obtained by several iterations of the proposed algorithm using binary search on the quantization level *l*. Visualization performance for different compression bit-rate is shown in Fig. 4.4.

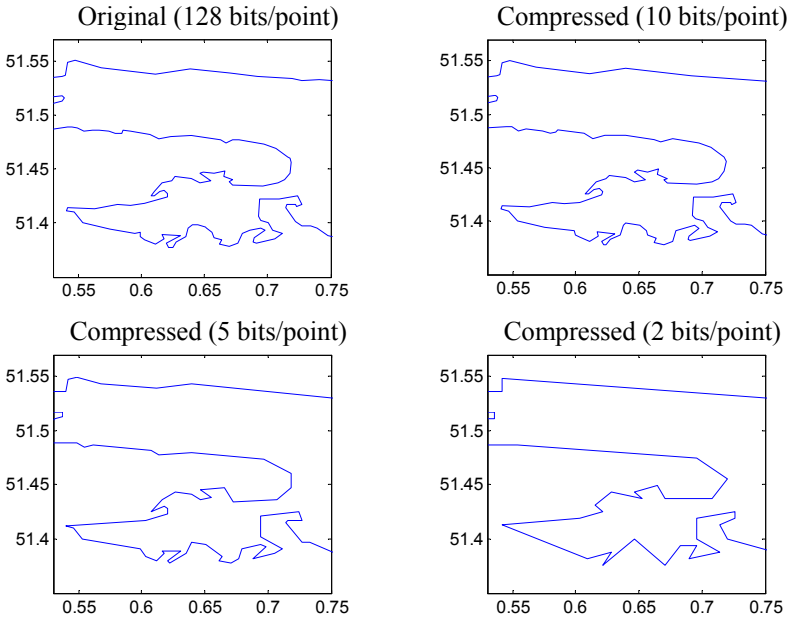

Fig. 4.4*. Performance under different bit-rate on a fragment of the testing UK map with 10911 points.*

4.2 OPTIMIZED VECTOR QUANTIZATION

Quantization plays an important part in lossy vector map compression. In **[P6]** we proposed an entropy-constrained vector quantization to jointly optimize the size and the structure of the

codebook. In order to lower the distortion to a desirable level, we exploit two-level design strategy, where the vector quantization codebook is designed only for the most common vectors and the remaining (outlier) vectors are coded by uniform quantization.

In vector quantization, we quantize the residual vectors by minimizing mean square error under a constraint that the average bit-rate does not exceed c:

$$\min D, s.t. R < c, where\ D = \sum_{i=1}^{N} (\| \boldsymbol{v_i} - Q(\boldsymbol{v_i}) \|^2) \tag{4.11}$$

where: $\boldsymbol{v}_i$ is the residual vector and $Q(\boldsymbol{v}_i)$ is its quantized form.

In entropy-constrained vector quantization (ECVQ), the constraint minimization procedure can be solved by converting it as an unconstrained optimization problem [25], formulated as $J = D + \lambda R$. For each Lagrangian parameter λ, it has a corresponding point on the rate-distortion curve. However, using a fixed-size codebook does not solve the problem efficiently, as the prediction error for vector data may vary in different cases. In order to find a better combination of Lagrangian parameter λ and size of codebook k, a high computation cost is needed for ECVQ, and thus, it is suitable only for off-line processes.

In order to improve the computational efficiency, *entropy-constrained pair-wise nearest neighbor for vector quantization* (ECPNN-VQ) has been proposed in [66]. It merges the pair of clusters that result in the smallest increase in distortion and largest decrease in bit-rate. Here, the increased distortion after merging two clusters i and j can be calculated by:

$$\Delta D = \frac{n_i + n_j}{n_i n_j} \| \boldsymbol{c_i} - \boldsymbol{c_j} \|_2^2 \tag{4.12}$$

where: n_i and n_j are the number of vectors in cluster i and j, respectively, and $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ are their centroid vectors. Thus, the change of the bit-rate can be calculated as:

$$\Delta R = -n_i \log(n_i / n) - n_j \log(n_j / n) \\ -(-(n_i + n_j)\log((n_i + n_j)/n) + r_q) \tag{4.13}$$

where: r_q is the code length of one quantized vector in the codebook, n is the number of residual vector.

In every merge step, the pair of clusters with the minimum -$\Delta D/\Delta R$ is merged. This can also be considered as searching the minimum slope in the rate-distortion curve. It therefore guarantees the optimality of each merge step. As in classic ECVQ frameworks, λ is interpreted as the slope of the line supporting the operational rate-distortion curve, and it is approximated in ECPNN-VQ as:

$$\lambda \approx -(D^{n+1}-D^n)/(R^{n+1}-R^n) \tag{4.14}$$

As a merge-based clustering method, ECPNN-VQ will stop reducing the size of the codebook when a given bit-rate constraint is met. The time complexity of ECPNN-VQ is $O(\tau N^2)$, the same as the traditional PNN algorithm [39].

After ECPNN-VQ, the cost of each residual vector $\boldsymbol{v}_i$ in cluster j can be formulated as:

$$J_{ij} = \| \boldsymbol{v}_i - \boldsymbol{c}_j \|_2^2 + \lambda(-\log_2(n_j / n) + r_q / n_j) \tag{4.15}$$

Meanwhile, the cost of uniform quantization can be calculated as: $J_0 = \Sigma_i(D_0 + \lambda r_{i0})$. In **[P5]** we have derived the optimal relation between quantization level l and *Lagrangian* parameter λ as:

$$l \approx \sqrt{6\lambda / \ln 2} \tag{4.16}$$

In our method, ECPNN is used to initialize the codebook. For a given bit constraint c, λ is first approximated on the rate distortion curve by Eq. (4.15). Then, a so-called *outlier cluster* is created with quantization level l by Eq. (4.16). Residual vectors are repartitioned to the clusters with minimum cost J by:

$$Q(\boldsymbol{v}_i) = \boldsymbol{c}_j, j = \arg\min_j(J_{ij}), j = 0,1..,k \tag{4.17}$$

where $j = 0$ is the outlier cluster.

A centroid step is followed in order to update the codebook. Parameters p_x and p_y for the geometric distribution are also updated. Fig. 4.5 shows an example of the codebook design. We can observe that several clusters have been moved completely to the outlier cluster, and that the size of the main codebook is reduced from 78 to 30.

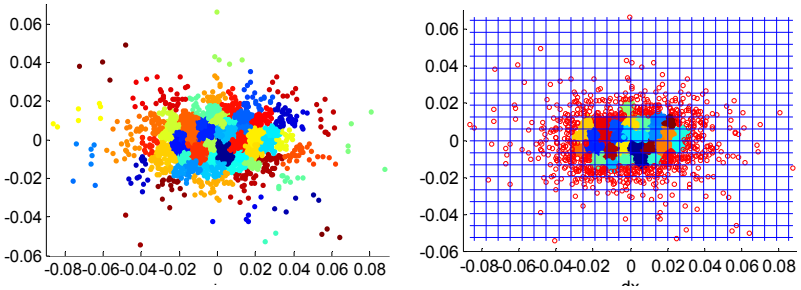

Fig. 4.5. *Demonstration of the outlier selection (5 bits/point constraint). ECPNN with MSE= $8.7 \cdot 10^{-6}$ and codebook size 78(left). The proposed two-level codebook with MSE= $6.9 \cdot 10^{-6}$ and size 30 (right). Outliers are marked as 'o'. Grid size for uniform quantization is also labeled. The testing curve is the UK map with 10911 points.*

However, selecting the quantized vector according to the closed-loop framework in Eq. (4.1) cannot guarantee optimality. This is because each point is quantized to the quantized position with minimum distortion, during the cost function minimization in the encoding procedure. In **[P6]** we keep more candidates (t = 8 in our implementation) in each quantization step, and the optimal solution is found by a dynamic programming process in the state space of size $n \cdot t$, where n is the number of the points of the vector map. Suppose that there are t best solutions recorded for encoding from p_1 to p_i, with the corresponding costs $L_{i,1}$, $L_{i,2}$, …, $L_{i,t}$, and $p^r_{i,1}, p^r_{i,2}, \ldots, p^r_{i,t}$ are the approximating positions for p_i. Based on a combination of k quantized vectors and t best solutions for p_i, in the next step, $k \cdot t$ solutions are tested for approximating p_{i+1} and t best solutions $p^r_{i+1,1}, p^r_{i+1,2}, \ldots, p^r_{i+1,t}$ are saved with minimum costs $L_{i+1,1}$, $L_{i+1,2}$,…$L_{i+1,t}$. In the end, backtracking is used to find the quantized vectors from $p^r_{n,1}$ with minimum cost $L_{n,1}$. The time complexity of the proposed approach is $O(ktn \cdot \log kt)$ in total.

In the next iteration, the residual vectors can be updated after the approximated curve has been constructed. Given bit-rate constraint c, λ is also updated by a binary search in the next iteration.

4.3 SUMMARY

In summary, we have made two significant improvements on the problem of lossy compression for vector map data.

Firstly, a fast dynamic quantization (FDQ) algorithm is proposed by deriving the optimal relationship between given quantization level l and Lagrangian parameter λ, which greatly saves on computational cost.

Secondly, a two-level strategy has been exploited and employed to optimize the codebook design. Vector quantization codebook is designed only for the most common vectors, and the remaining vectors (outliers) are coded by additional bits using uniform quantization. Additionally, instead of using a conventionally greedy approach in the quantization process, a dynamic programming method is utilized to improve the quantized vector selection.

5. Compression of GPS Trajectory

Over recent years, enormous numbers of GPS trajectories, which record users' spatial and temporal information, have been collected by geo-positioning mobile phones providing plenty of test datasets[1]. The massive volume of trajectory data brings about a heavy burden both for network transmission and data storage.

A number of compression algorithms have been proposed by reducing the number of points in the trajectory data. These *line simplification* solutions were discussed in Section 3. Except for the simplification process by polygonal approximation, semantic meaning of the GPS trajectories has also been considered during the compression process in urban areas in [99], whereas a trajectory compression algorithm with network constraints has been developed in [51].

However, in these algorithms, the compression is achieved only by the approximation process and these methods lack a rigorous analytical approach on the encoding procedure of the reduced trajectories. Without further compression, 12 bytes are used for saving each point including: latitude, longitude, and timestamp. In [54], arithmetic coding is also considered in the encoding process, but only a simple predefined prediction model is used, which may not be robust in a real application.

[1] These datasets are:
BerlinMOD dataset: http://dna.fernuni-hagen.de/secondo/BerlinMOD/BerlinMOD.html
Geolife Dataset: http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/
MOPSI Dataset: http://cs.joensuu.fi/sipu/MOPSI_GPS Traj_TXTv1.zip

5.1 PROPOSED COMPRESSION ALGORITHM

Therefore, a coding algorithm for GPS trajectories with latitude, longitude and timestamp information is considered. A lossy compression algorithm is proposed in [**P7**] for GPS trajectories under maximum synchronous Euclidean distance (max SED). Suppose $p_j = (x_j, y_j, t_j)$ is the point on the original GPS trajectory and p_j' is its synchronized approximated position. The distortion of the whole trajectory is calculated by its maximum synchronized Euclidean distance:

$$SED(P, P') = \max_{1 \le i \le n}(SED(p_i, p_i')) \tag{5.1}$$

In contrast to the existing trajectory compression algorithm, we achieve two significant improvements.

Firstly, in vector map compression, differential coordinates are used directly in the encoding process. However, in GPS trajectories they are inconsistent if different reduction rates are applied after the simplification (approximation) process. Meanwhile, speed and direction changes are more robust variants, even if a simplification (approximation) step is applied with different reduction rates in different segments; therefore they are used in the proposed algorithm.

Secondly, line simplification and quantization are combined during the approximation process in order to seek the approximated trajectory for compression.

An approximation example is demonstrated in Fig. 5.1. Suppose we want to approximate a sub-trajectory $P_{i_k}^{i_{k+1}}$ by line segment $\overline{p_{i_k}' p_{i_{k+1}}'}$, where p_{ik}' is the approximated position of p_{ik} in the previous step. If time interval $\Delta t(k)$ is known, the speed of the line segment is:

$$spd(k) = dist(p_{i_k}', p_{i_{k+1}}) / \Delta t(k) \tag{5.2}$$

Given max SED tolerance ε, we assume the quantization error of point p_{ik+1}' is $\gamma\varepsilon$ at maximum, thus the quantized level for speed can be set as:

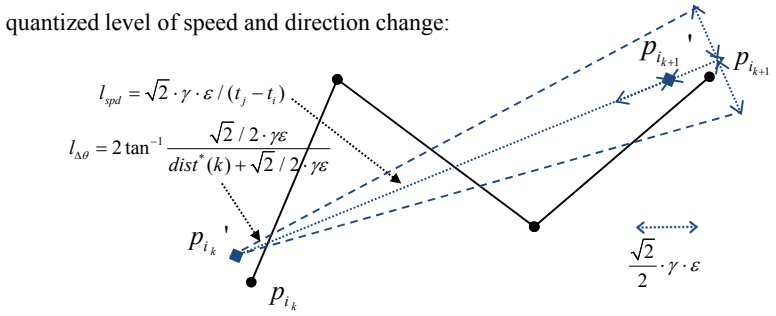

Fig. 5.1. *An example of the quantization process for the GPS trajectory*

$$l_{spd}(k) = \sqrt{2} \cdot \gamma \cdot \varepsilon / \Delta t(k) \tag{5.3}$$

Here γ is a parameter as the ratio of the quantized error and the total SED, which is set as γ = 0.5 by our experiment. Thus, the quantized speed can be calculated as:

$$spd^*(k) = [spd(k) / l_{spd}(k)] \cdot l_{spd}(k) \tag{5.4}$$

Meanwhile, we can get the direction change $\Delta\theta(k)$ with a value between $-\pi$ and π, where a negative value represents the direction change in a clockwise direction.

Given the quantized speed $spd^*(k)$, the quantization level for the direction change can be estimated as:

$$l_{\Delta\theta}(k) = 2\tan^{-1} \frac{\sqrt{2}\gamma\varepsilon / 2}{spd^*(k) \cdot \Delta t(k) + \sqrt{2}\gamma\varepsilon / 2} \tag{5.5}$$

Thus, the quantized direction change is:

$$\Delta\theta^*(k) = [\Delta\theta(k) / l_{\Delta\theta(k)}] \cdot l_{\Delta\theta(k)} \tag{5.5}$$

Based on the quantized speed and direction change $spd^*(k)$ and $\Delta\theta^*(k)$, the quantized position $p_{ik+1}' = \{x_{ik+1}', y_{ik+1}', t_{ik+1}'\}$ can be approximated as:

$$
\begin{aligned}
x_{i_{k+1}}' &= x_{i_k}' + spd^*(k) \cdot \Delta t(k) \cdot \cos(\Delta\theta^*(k) + \theta^*(k-1)) \\
y_{i_{k+1}}' &= y_{i_k}' + spd^*(k) \cdot \Delta t(k) \cdot \sin(\Delta\theta^*(k) + \theta^*(k-1)) \\
t_{i_{k+1}}' &= t_{i_{k+1}}
\end{aligned}
\tag{5.6}
$$

A greedy solution is used for the trajectory approximation. Starting from the first point, the furthest point is found with an approximated SED less than the given error tolerance.

Note that when the input of the GPS trajectory is latitude and longitude in WGS84 format, a *Mercator projection* is needed as a preprocessing step so that the distance can be calculated directly.

In the encoding process, we need to encode both the differential coordinates and time difference (Δx, Δy and Δt). Suppose $p_{ik}' = (x_{ik}', y_{ik}', t_{ik}')$ and $p_{ik+1}' = (x_{ik+1}', y_{ik+1}', t_{ik+1}')$ are two neighboring points in the approximated trajectory P'. Firstly, the time difference is encoded by the following probability:

$$
\begin{aligned}
& p(\Delta t(k) = t_{i_{k+1}} - t_{i_k}) = \frac{p + \delta_t / rtsp_{max}}{1 + \delta_t} \\
& \text{where } p = \mathbf{r_t}(\Delta t(k) / min_{tsp}) / \sum_{s=1}^{rtsp_{max}} \mathbf{r_t}(s), \\
& rtsp_{max} = \max(\Delta t(q)) / tsp_{min}, q = 1, 2, ..., m-1.
\end{aligned}
\tag{5.7}
$$

where: tsp_{min} is the minimum sampling time on the GPS trajectory (1s in most cases) and δ_t is a bias factor ($\delta_t = 0.01$), vector $\mathbf{r_t}$ is initialized as a zero vector with size $rtsp_{max} \times 1$.

After $\Delta t(k)$ has been encoded, vector $\mathbf{r_t}$ is updated by:

$$
r_t(s) = \begin{cases} 1 + \mu_t r_t(s), & s = \Delta t_k / tsp_{min} \\ \mu_t r_t(s), & \text{else} \end{cases}
\tag{5.8}
$$

where μ_t is a forgetting factor, which gives higher influence on the recently encoded time intervals with μ_t = 0.995 in this paper. The reason that we use a forgetting factor is because of the possible multiple transportation models in the GPS trajectory. A higher reduction rate can possibly be achieved for the segments with slower moving speed (e.g., walking) compared with faster moving segments

(e.g., car). Thus, it will be helpful to improve the coding performance if a forgetting factor is used.

The speed value is then predicted by $spd_{pred}(k)$ and $\sigma_{spdpred}^2(k)$, as:

$$spd_{pred}(k) = n_{c1}\sum_i spd^*(i)\cdot(\Delta t(i)\cdot\exp(-\frac{1}{2}\cdot(\frac{t(k)-t(i)}{w_t})^2))$$

$$\sigma_{spdpred}{}^2(k) = n_{c2}\sum_i \Delta t(i)\cdot((spd^*(i)-spd_{pred}(k))^2 \quad (5.9)$$

$$+\frac{\gamma^2\varepsilon^2}{6\Delta t^2(i)}+\frac{\sigma_{GPS}^2}{2\Delta t^2(i)}, \text{where } t(i)\geq t(k)-d\cdot\Delta t(k)$$

where: n_{c1} and n_{c2} are normalized values for the weighting factors, while w_t , d and σ_{GPS} are parameters with w_t = 20, d = 4 and σ_{GPS} = 5. The second and third terms of $\sigma_{spdpred}^2(k)$ are the variance of the quantization procedure and the GPS error, respectively.

The probability is then estimated by assuming the speed has a *Gaussian* distribution:

$$p(spd^*(k)) = \frac{p+\delta_{spd}\ / nlv_{spd}(k)}{1+\delta_{spd}} \quad (5.10)$$

where: p has a Gaussian distribution with mean $spd_{pred}(k)$ and variance $\sigma_{spdpred}^2(k)$, bias factor δ_{spd} is set as 0.01.

Adaptive arithmetic coding is also used directly for direction change in [**P7**]. However, GPS signals are not always accurate and a quantization step will also cause errors. Therefore, the encoded and true distributions of the direction change are not always same. Here, Bayes' theorem is applied to improve the probability estimation for the direction changes. Suppose $P(\Delta\theta_0)$ is the distribution of direction change when the signal is clean, $P(\Delta\theta_k)$ is the predicted distribution for p_k, we have:

$$P(\Delta\theta_k) = \sum_b P(\Delta\theta_k \mid \Delta\theta_0 = b) \cdot P(\Delta\theta_0 = b)$$

$$\text{where } P(\Delta\theta_k \mid \Delta\theta_0 = b) = -\frac{1}{a^2}(\Delta\theta_k - b)\,\mathrm{sgn}(\Delta\theta_k - b) + \frac{1}{a}, \quad b - a \le \Delta\theta_k \le b + a \tag{5.11}$$

$$a = \tan^{-1}\left(\frac{\gamma\varepsilon}{\sqrt{2}spd(k)\Delta t(k)}\right)$$

After p_k is encoded, posterior probability $P(\Delta\theta_0 \mid \Delta\theta_k)$ is estimated by:

$$P(\Delta\theta_0 \mid \Delta\theta_k) = \frac{P(\Delta\theta_k \mid \Delta\theta_0) \cdot P(\Delta\theta_0)}{P(\Delta\theta_k)} \tag{5.12}$$

$P(\Delta\theta_0)$ is then updated:

$$P(\Delta\theta_0) = \mu_{\Delta\theta} \cdot P(\Delta\theta_0) + P(\Delta\theta_0 \mid \Delta\theta_k) \tag{5.13}$$

From our experiment, we set 180 levels uniformly distributed between $-\pi$ and π for $P(\Delta\theta_0)$.

Given the estimated probabilities, the time difference and speed and direction changes are encoded by arithmetic coding. In the compressed file, a 192 bits fixed-length header is used to save the parameters of the trajectory. These values include: start position of x and y (30 bits + 30 bits), time (32 bits), tsp_{min} (8 bits), $rtsp_{max}$ (16 bits), m (24 bits), spd_{max} (32 bits), and a scaling factor of *Mercator projection* (20 bits).

Complexity analysis is given in Table 5.1. Here τ_1, τ_2 and τ_3 are constant values, which are not related to the size of the GPS trajectory data. An example of the compression result at different tolerances can be seen in Fig. 5.2. In the experiments, the compression algorithm is evaluated by KB/h, which tells us the average storage cost under a given time duration.

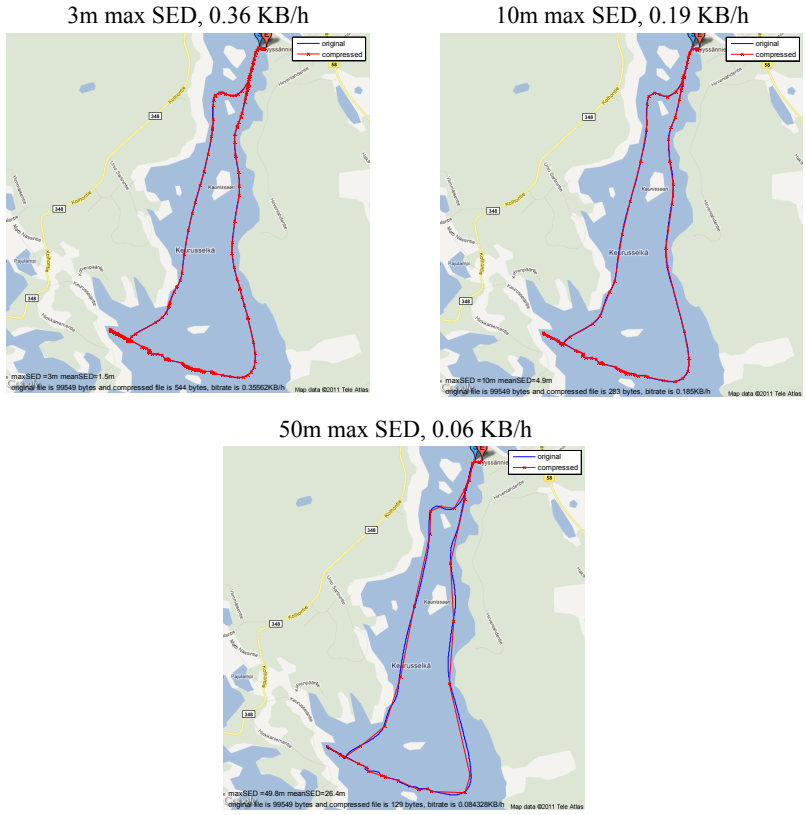

Fig. 5.2. *Compression example of the proposed GPS trajectory compression (GTC) algorithm.*

Table 5.1 Summary of the Expected Time Complexity of the Proposed GPS Trajectory Compression Algorithm

Step			Time Complexity
I.	Approximate Trajectory		$O(n^2/m)$
II.	Encoding Process	Time	$O(m \cdot \tau_1)$, $\tau_1 = rtsp_{max}$
		Speed	$O(m \cdot \tau_2), \tau_2 = \frac{spd_{\max}}{\varepsilon} \overline{\Delta t}$
		Direction Change	$O(m \cdot \tau_3), \tau_3 = nlv_{\Delta\theta}$
III.	Decoding Process		Same as in the encoding process

5.2 FILTERING GPS TRAJECTORY BEFORE COMPRESSION

GPS trajectories are never perfectly accurate due to sensor noise and other factors. Filtering is therefore needed when the trajectory data is particularly noisy. Therefore, filtering can be considered beforehand on the GPS trajectories to smooth the noise and potentially decrease the error in the measurements. A summary of the GPS trajectory preprocessing techniques can be found in [70].

The noise of GPS trajectories is usually modeled by adding unknown, random Gaussian noise, assumed to be drawn from a two-dimensional Gaussian probability density, with a zero mean and a diagonal covariance matrix. Some filters, such as the Kalman filter [44] and particle filters [49], require both the measurement model and the dynamic model. In reality, the parameters of these models are difficult to estimate and may change from one segment to another.

Therefore, we propose a filtering algorithm for GPS trajectories based on its feature properties. Our proposed algorithm has two steps: outlier removal and filtering with speed consistency. Prior information, such as a road network is not needed in the proposed algorithm.

Firstly, for those points with unreliable speed and speed change, they are selected as outlier points and removed. Secondly, the trajectory is smoothed using *ridge regression* with a smooth speed regularized term. It can be considered as an optimization problem for both the position distortion and the speed change.

From our experiment, if a filtering algorithm is performed beforehand, the bit-rate can be reduced by around 30%, 20%, and 15% for 1m, 3m, and 10m SED, respectively. Meanwhile, if a higher tolerance is set, the bit-rate will not be changed even if a filtering operation is used.

5.3 SUMMARY

In summary, we have addressed the problem of spatial-temporal data compression, particularly the lossy compression of GPS trajectories with sets of (x, y, t) records. In the proposed algorithm,

both data reduction and quantization are considered in the approximation process.

There are several immediate extensions of our present work. First, we plan to extend the compression for online application. Second, improvement of the approximation and encoding processes will be considered, e.g., using dynamic programming to seek an approximated trajectory with minimum coding cost, and improve the probability estimation by extended Kalman filtering. Third, applying a hierarchy of compression stages is an interesting idea for further investigation. Finally, the coding scheme of multiple similar GPS trajectories can be jointly considered.

6. Summary of the Contributions

6.1 CONTRIBUTIONS OF THE THESIS

In [**P1**] a multi-layer filtering algorithm is proposed for raster map images by transforming discrete denoising into binary domain. It consists of three intuitive image operators: layer decomposition, binary image filtering, and layer merging. Experimental results show that the proposed method has a lower computational cost and memory consumption than other statistical filters, and it is efficient for denoising raster map images. From our experiment, it achieves a similar error rate to that of Discrete *Universal Denoising* (DUDE), but with a lower computational cost.

In [**P2**] we extended the statistical filter to a specific continuous-input-and-finite-output problem, in which the map images are corrupted by an additive Gaussian noise. The extended method iteratively conducts a fusion procedure based on the probability distribution of pixels' intensity in RGB space and their conditional probabilities in the local contexts. It can also be considered as an energy minimization model similar to the *Markov random field*, but the neighborhood similarity is replaced by a conditional probability of the local context. It is also extended for filtering mixed Gaussian-impulsive noise. The proposed raster map filtering algorithm achieves better results than the previous denoising algorithms for photographic images, see Tables 6.1-6.3. It can also be applied in color quantization processes for noisy images.

In [**P3**] we focus on the *optimized context selection* in filtering raster map images. The reason is that identifying and excluding the possible noisy pixels in the context template is of great importance in achieving a desirable filtering result. To solve the above problems, we propose a

novel context-based voting method to detect the possible noisy pixels. From our experiment, the error rate is improved by 5% compared with [**P2**] for filtering impulsive noise. The proposed voting strategy can also be extended and used in other image analysis problem, such as texture analysis.

In [**P4**] a fast $O(N)$ multi-resolution GPS trajectory simplification algorithm is proposed, with both experimental results and theoretical analysis. The proposed simplification algorithm is based on a joint optimization of two new error measures, *local integral square synchronous Euclidean distance* (LSSD), and *integral square synchronous Euclidean distance* (ISSD). The time complexity is reduced from $O(N^2)$ as in the TD-TR algorithm [34] to $O(N)$, while our proposed method also achieves a better approximation result. The result is summarized in Table 6.4.

In [**P5**] we propose a fast dynamic quantization algorithm for lossy compression of vector maps with latitude and longitude information. The underlying algorithm first identifies an optimal Lagrangian multiplier λ value for each quantization step l, and then constructs only one rate-distortion curve for encoding the differential coordinates. Experimental results show that the proposed method is 20 times faster than the previous dynamic quantization algorithm and achieves a similar or better compression performance.

In [**P6**] a two-level strategy has been exploited to optimize the codebook design in vector map compression. To reduce the high coding cost of a large codebook in vector quantization, the codebook is designed only for the most common vectors, and the rest (called outliers) are coded by uniform quantization. A dynamic programming method is applied to improve the quantized vector selection in a closed-loop framework, instead of using a conventional greedy approach. According to our experiments, the proposed algorithm achieves approximately a 5% improvement compared with [**P5**]. The result is summarized in Table 6.5.

In [**P7**] we exploit the problem of lossy compression for GPS trajectories with latitude, longitude and timestamp information under a given error tolerance, i.e., maximum synchronous Euclidean distance (max SED). In the proposed algorithm, speed and direction changes are used in the encoding process instead of differential coordinates. Line simplification and quantization are combined in the encoding process in order to seek the approximated trajectory for compression. From our experiments, the bit-rate of the proposed algorithm is 0.39 KB/h for 3m accuracy, and with only 35% of the storage cost of the TD-TR+LZMA algorithm, see Table 6.6. In comparison with the GPX file the proposed algorithm achieves over 100:1 compression ratio, see Table 6.7.

6.2 SUMMARY OF RESULTS

In this sub-section, Table 6.1-6.7 are listed to summarize the experimental results in [P1] - [P7].

Table 6.1 Statistical filtering for impulsive noise of Set #1

Error rate (%)	Noise level		
	$\delta = 0.05$	$\delta = 0.10$	$\delta = 0.20$
PGF [3]	1.40	3.56	10.9
CT [19]	1.04	7.83	19.9
DUDE [11]	0.94	1.95	5.04
MUD **[P1]**	0.93	2.12	5.46
CTM **[P2]**	0.65	1.31	3.35
OCTM **[P3]**	**0.62**	**1.20**	**3.08**

Table 6.2 Statistical filtering for Gaussian noise of Set #1

PSNR(dB)	Noise level		
	$\sigma = 15$	$\sigma = 25$	$\sigma = 35$
BLS-GSM [5]	26.8	24.4	22.2
NLM [6]	26.9	24.9	22.7
BM3D [8]	29.1	26.2	23.9
CTM **[P2]**	**59.4**	**47.1**	**42.7**

Table 6.3 Statistical filtering for mixed Gaussian-impulsive noise of Set #1

PSNR(dB)	Noise level		
	δ = 0.03, σ= 15	δ = 0.05, σ= 25	δ = 0.10, σ= 35
FPGA [32]	22.0	19.5	16.4
PGF [3] +BM3D [8]	22.5	19.8	16.3
CTM **[P2]**	**25.6**	**23.6**	**20.3**

Table 6.4 Result of GPS Trajectory Simplification of MOPSI Dataset

	RMSE	MAE	MEDE	MAXE
f_{LSSD} = 50, N/M = 8.02				
D-P [46]	4.51	2.38	1.32	39.0
TD-TR [34]	1.82	1.41	1.23	**4.61**
MRPA[**P4**]	**1.61**	**1.23**	**1.05**	5.88
f_{LSSD} = 2000, N/M = 25.1				
D-P [46]	13.8	8.39	5.08	81.1
TD-TR [34]	6.85	5.55	4.82	**17.7**
MRPA[**P4**]	**5.96**	**4.76**	**4.07**	23.9
f_{LSSD} =10^5, N/M = 79.4				
D-P [46]	42.0	29.0	19.9	173.3
TD-TR [34]	26.7	21.6	18.3	**70.5**
MRPA[**P4**]	**22.9**	**18.5**	**15.8**	79.2

Table 6.5 Vector map compression for UK map

MSE (x10^{-7})	2 bit/point	6 bit/point	10 bit/point
CBC [74]	N/A	78	20
Ref Line [75]	N/A	450	6
DFQ [**P5**]	70	16	1.3
OVQ [**P6**]	55	10	1.3

Table 6.6 Result of GPS trajectory compression

	Geolife (KB/h)		MOPSI (KB/h)	
	SED = 3m	SED = 10m	SED = 3m	SED = 10m
TD-TR + LZMA [34]	0.95	0.53	1.94	1.06
GTC [**P7**]	0.39	0.19	0.75	0.35
Filtered Data + GTC [**P7**]	0.31	0.16	0.46	0.26

Table 6.7 Size of the compressed GPS trajectories

	Geolife (KB)	MOPSI (KB)
Original GPX file	426,100	75,573
Compressed GPX file	36,514	14,466
Compressed SED = 3m	1,215	258
Compressed SED = 10 m	576	120

7. Conclusions

We have studied raster map filtering, GPS trajectory simplification, lossy compression of vector map and GPS trajectories.

In raster map filtering, both the multi-layer method and statistical method are proposed. In the multi-layer method, the problem of color image filtering is converted into binary domain. Therefore it can achieve a higher computational efficiency. However, even if the local color priority scheme is used, this method still has a limitation when processing map images with higher number of colors (>10 colors). Later, extensions are made for statistical filtering algorithms on processing additive Gaussian noise and mixed Gaussian-impulsive noise. A novel voting-based context selection scheme is also developed to exclude the noisy pixels in the context template. It has a lower error rate at the expense of high computational cost. In future work, we will consider extending the methodology for processing gray-scale images and medical images.

For vector data and GPS trajectories, both simplification and lossy compression algorithms are presented. Two new error measures are proposed for spatial-temporal data and the proposed simplification algorithm has a linear time complexity. It can be used to improve the scheme of map visualization as well as a point reduction solution during server-client communication. In future work, a joint simplification solution for multiple GPS trajectories will also be considered, where a fast trajectory clustering [113] algorithm may be a possible solution for real-time application.

Meanwhile, the lossy compression algorithms for both vector map and GPS trajectories are also discussed. As a general compression solution, the proposed algorithm uses no prior information during the encoding process, and each GPS trajectory is processed separately. In the future, history information will also be considered, and a joint compression scheme for multiple GPS trajectories can be used. Road networks can also be valuable information during the compression process to improve both the compression ratio and the accuracy of the trajectories.

References

1. P.K. Agarwal, K.R. Varadarajan, "Efficient algorithms for approximating polygonal chains", *Discrete Comput. Geom*. 23, 273–291, 2000.

2. A. Akimov, A. Kolesnikov and P. Fränti, "Coordinate quantization in vector map compression", *IASTED Conference on Visualization, Imaging and Image Processing* (*VIIP'04*), 748-753, 2004.

3. A. Akimov, A. Kolesnikov and P. Fränti, "Reference line approach for vector data compression", *IEEE Int. Conf. on Image Processing (ICIP'04)*, vol. 2, 1891-1894, 2004.

4. A. Akimov, A. Kolesnikov, and P. Fränti, "Lossless compression of color map images by context tree modeling", *IEEE Trans. Image Process.*, vol. 16, no. 1, 114–120, 2007.

5. H. Alt, L.J. Guibas, "Handbook of Computational Geometry", pp. 121-153, 1999.

6. H. Alt, C. Knauer, C. Wenk, "Matching polygonal curves with respect to the Fréchet distance", STACS, LNCS , 63-74, 2001.

7. A. Barbu, "Training an active random field for real-time image denoising", *IEEE Trans. Image Process.*, vol. 18, no. 11, 2451–2462, 2009.

8. M. Berg, O. Cheong, M. Kreveld, *Computational geometry: algorithms and applications*, 3rd edition, Springer, 2008.

9. P. Bhowmick, B.B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties", *IEEE Trans. on Pattern Anal. Mach. Intell.*, 29(9), 1590 – 1602, 2007.

10. N. Bouaynaya, M. Charif-Chefchaouni, D. Schonfeld, "Theoretical foundations of spatially-variant mathematical morphology Part I: Binary Images", *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(5), 823-836, 2008.

11. A. Buades, B. Coll, and J.M. Morel, "A non-local algorithm for image denoising", in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, vol. 2, 60–65, 2005.

12. C. Le, T. Ebrahimi, M. Kunt, "Progressive content-based shape compression for retrieval of binary images", *Computer Vision and Image Understanding*, 71(2), 198-212, 1998.

13. B.P. Buttenfield, "Transmitting vector geospatial data across the Internet", *Proc. GIScience, LNCS*, vol. 2478, 51-64, 2002.

14. J. Camarena, V. Gregori, S. Morillas, A. Sapena, "Fast detection and removal of impulsive noise using peer groups and fuzzy metrics", *Journal of Visual Communication and Image Representation*, 19(1), 20-29, 2008.

15. J. Camarena, V. Gregori, S. Morillas, A. Sapena, "Two-step fuzzy logic-based method for impulse noise detection in colour images", *Pattern Recognition Letters*, 31(13), 1842-1849, 2010.

16. H. Cao, O. Wolfson, G. Trajcevski, "Spatio-temporal data reduction with deterministic error bounds", *VLDB Journal*, 15(3), 211-228, 2006.

17. W.S. Chan, F. Chin, "On approximation of polygonal curves with minimum number of line segments or minimum error", *Lecture Notes in Computer Science,* vol.650, 378-387, 1992.

18. P. Chatterjee, P. Milanfar, "Patch-based near-optimal image denoising", *IEEE Trans. on Image Process.*, 21(4), 1635 -1649, 2012.

19. C. Chaux, A. Jezierska, J-C. Pesquet, H. Talbot, "A Spatial Regularization Approach for Vector Quantization", *Journal of Mathematical Imaging and Vision*, 41(1), 23-38, 2011.

20. D. Chen, O. Daescu, "Space-efficient algorithms for approximating polygonal curves in two dimensional space", *Computing and Combinatorics*, vol.1449, 45-55, 1998.

21. M. Chen, M. Xu, P. Fränti, "Statistical filtering of raster map images", *IEEE Int. Conf. on Multimedia & Expo* (ICME'10), Singapore, 394-399, July 2010.

22. Y. Chen, K. Jiang, Y. Zheng, C. Li, N. Yu, "Trajectory Simplification Method for Location-Based Social Networking Services", *ACM GIS workshop on Location-based social networking services*, 33-40, 2009.

23. Y. Chen, Y. Liu, Z. Fu, and R. Wang, "Automatic extracting residential areas from color scanned topographical maps," *Image Signal Process.*, 2009.

24. J.W. Choi, G.H. Elkaim, "Bézier curves for trajectory guidance", World Congress on Engineering and Computer Science (ECECS'08), 625-630, Oct. 2008.

25. P. Chou, T. Loolabaugh and R.M. Gray, "Entropy-constrained vector quantization" , *IEEE Trans. on Acoustics, Speech, Signal Processing*, 37(1), 31–42,1989.

26. K.L Chung, W.M Yan, W.Y. Chen, "Efficient algorithms for 3-D polygonal approximation based on LISE criterion", *Pattern Recognition* 35, 2539-2548, 2002.

27. K.L. Chung, P.H. Liao, J.M. Chang, "Novel efficient two-pass algorithm for closed polygonal approximation based on LISE and curvature constraint criteria", *Journal of Visual Communication and Image Representation* 19(4), 219-230, May 2008.

28. K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering", *IEEE Trans. Image Process.*, vol. 16, no. 8, 2080–2095, 2007.

29. O. Daescu, "New results on path approximation", *Algorithmica* 38(2), 131–143, 2004.

30. O. Daescu, N. Mi, "Polygonal chain approximation: A query based approach", *Computational Geometry*, 30(1), 41–58, 2005.

31. O. Daescu, N. Mi, C.S. Shin, A. Wolff, "Farthest-point queries with geometric and combinatorial constraints", *Computational Geometry*, 33 (3), 174–185, 2006.

32. H. Ding, G. Trajcevski, P. Scheuermann. X. Wang, E. J. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures", *Proceedings of the VLDB*, 1(2), 1542-1552, 2008.

33. W. Dong, X. Li, L. Zhang and G. Shi, "Sparsity-based Image Denoising via Dictionary Learning and Structural Clustering", *IEEE Conference on Computer Vision and Pattern Recognition*, 457-464, 2011.

34. D.H. Douglas, T.K. Peucker, "Algorithm for the reduction of the number of points required to represent a line or its caricature", *The Canadian Cartographer*, 10 (2), 112-122, 1973.

35. M. Elad and M. Aharon. "Image denoising via sparse and redundant representations over learned dictionaries", *IEEE Trans. Image Process.*, vol. 15, no. 12, 3736–3745, 2006.

36. R. Estkowski, J. Mitchell, "Simplifying a polygonal subdivision while keeping it simple", *Symposium on Computational Geometry*, 40-49, 2001.

37. D. Eu, G.T. Toussaint, "On approximation polygonal curves in two and three dimensions", *Graphical Models, and Image Processing*, 56(3), 231-246, 1994.

38. S. Forchhhammer, O. Jensen, "Content layer progressive coding of digital maps", *IEEE Trans. on Image Process.*, 11(12), 1349-1356, 2002.

39. P. Fränti, T. Kaukoranta, D.-F. Shen and K.-S. Chang, "Fast and memory efficient implementation of the exact PNN", *IEEE Trans. on Image Process.*, 9 (5), 773-777, 2000.

40. P. Fränti, A. Tabarcea, J. Kuittinen, V. Hautamäki, "Location-based search engine for multimedia phones", *IEEE Int. Conf. on Multimedia & Expo* (ICME'10), Singapore, 558-563, July 2010.

41. P. Fränti, J. Chen, A. Tabarcea, "Four aspects of relevance in location-based media: content, time, location and network", Int. Conf. on Web Information Systems & Technologies (WEBIST'11), Noordwijkerhout, Netherlands, 413-417, May 2011.

42. G. Gemelos, S. Sigurjonsson, and T. Weissman, "Algorithms for discrete denoising under channel uncertainty", *IEEE Trans. Signal Process.*, vol. 54, no. 6, 2263–2276, 2006.

43. G. Gimel'farb, "Adaptive context for a discrete universal denoiser", *Structural, Syntactic, and Statistical Pattern Recognition, IAPR Int. Workshops, SSPR 2004 and SPR*, 477–485, 2004.

44. M.S. Grewal, L.R. Weill, A. P. Andrews, Chapter 8 in Global Positioning Systems, inertial Navigation and Integration. Second Edition, John Wiley & Sons, 2007.

45. A. Gribov, E. Bodansky, "A new method of polyline approximation", *Proc. of International Conference on Structural, Syntactic and Pattern Recognition, LNCS*, vol. 3138, 504-511, 2004.

46. M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Cluster validity methods", *ACM SIGMOD Record*, 2002.

47. T. Henderson and T. Linton, "Raster map image analysis," in *Int. Conf. Document Analysis and Recognition* (ICDAR'09), 376–380, 2009.

48. J. Hershberger, J. Snoeyink, "Cartographic line simplification and polygon CSG formulae in $O(n \log^* n)$ time", *5th International Workshop on Algorithms and Data Structures*, 93–103, 1997.

49. J. Hightower, G. Borriello, “Particle filters for location estimation in ubiquitous computing, a case study”, *Int. Conf. on Ubiquitous Computing*, 88–106, 2004.

50. H. Imai, M. Iri, "Polygonal approximations of a curve-formulations and algorithms", *Computational Morphology*, 71-86, Amsterdam, 1988.

51. G. Kellaris, N. Pelekis and Y. Theodoridis, "Trajectory Compression under Network Constraints", *Lecture Notes in Computer Science*, Vol. 5644, 392-398, 2009.

52. A. Khotanzad and E. Zink, “Contour line and geographic feature extraction from USGS color topographical paper maps,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(1), 18–31, 2003.

53. M. Koegel, W. Kiess, M. Kerper, M. Mauve, “Compact Vehicular Trajectory Encoding”, *IEEE Vehicular Technology Conference* (VTC ’11), 1-5, May 2011.

54. M. Koegel, M. Mauve, ”On the Spatio-Temporal Information Content and Arithmetic Coding of Discrete Trajectories”, *International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Copenhagen, Denmark, December 2011.

55. A. Kolesnikov, P. Fränti, "A fast near-optimal min-# polygonal approximation of digitized curves", *ACIT’2002*, 418-422, 2002.

56. A. Kolesnikov, P. Fränti, "Reduced-search dynamic programming for approximation of polygonal curves", *Pattern Recognition Letters*, 24 (14), 2243-2254, October 2003.

57. A. Kolesnikov, P. Fränti, X. Wu, "Multi-resolution Polygonal Approximation of Digital Curves", *17th International Conference on Pattern Recognition (ICPR’04*), vol.2, 855-858, 2004.

58. A. Kolesnikov, "Optimal encoding of vector data with polygonal approximation and vertex quantization", *SCIA'05*, LNCS, vol. 3540, 1186–1195. 2005.

59. A. Kolesnikov, A. Akimov, "Distortion-constrained compression of vector maps", SAC, 8-12, 2007.

60. A. Kolesnikov, "Optimal algorithm for lossy vector data compression", *Int. Conf. on Image Analysis and Recognition* (ICIAR'07), LNCS 4633, 761-771, 2007.

61. A. Kolesnikov, "Fast Algorithm for ISE-bounded Polygonal Approximation", *IEEE International Conference on Image Process.*, 1013-1015, 2008.

62. A. Kolesnikov, "Fast algorithm for error-bounded compression of digital curves", *IEEE International Conference on Image Processing*, 1453-1456, 2010.

63. P. Kopylov and P. Fränti, "Color quantization of map images", *IASTED Conf. Visualization, Imaging, and Image Processing*, 837–842, 2004.

64. P. Kopylov and P. Fränti, "Filtering of color map images by context tree modeling", *IEEE Int. Conf. Image Process.*, vol. 1, 267–270, 2004.

65. P. Kopylov and P. Fränti, "Compression of map images by multilayer context tree modeling", *IEEE Trans. on Image Process.*, 14(1), pp.1-11, 2005.

66. F. Kossentini, "A fast PNN design algorithm for entropy - constrained residual vector quantization", *IEEE Trans. on Image Process.*, vol.7, 1045-1050, 1998.

67. L. Labakhua, U. Nunes, R. Rodrigues, F.S. Leite, "Smooth trajectory planning for fully automated passengers vehicles - spline and clothoid based methods and its simulation", *Informatics*

in Control Automation and Robotics, Lecture Notes in Electrical Engineering, 15(2), 169-186,2008.

68. R. Lange, T. Farrel, F. Dürr, K. Rothermel, "Remote real-time trajectory simplification", *IEEE International Conference on Pervasive Computing and Communications*, 1-10, 2009.

69. J.G. Lee, J. Han, K.Y. Whang, "Trajectory clustering: a partition-and-group framework", *ACM SIGMOD Int. Conf. on Management of Data*, 593-604, New York, USA, 2007.

70. W.C. Lee and J. Krumm, "Chapter 1: Trajectory Preprocessing", in Book *Computing with Spatial Trajectories*, Springer, 2011.

71. S. Leyk and R. Boesch, "Colors of the past: color image segmentation in historical topographic maps based on homogeneity," *Geoinfomatica*, vol. 14, 1–21, 2010.

72. Z. Li, and S. Oppenshaw, "A natural principle for the objective generalization of digital maps", *Cartography and Geographical Information Systems*, vol. 20, 19-29, 1993.

73. R. Lukac, "Adaptive vector median filtering", *Pattern Recognition Letters*, vol. 24, 1889–1899, 2003.

74. J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-Local Sparse Models for Image Restoration," *IEEE International Conference on Computer Vision*, 2272 - 2279, 2009.

75. P.F. Marteau, G. Ménier, "Speeding up simplification of polygonal curves using nested approximations", *Pattern Analysis and Application*, 2008.

76. A. Masood, "Optimized polygonal approximation by dominant point deletion", *Pattern Recognition,* 41 (1), 227-239, 2008.

77. D. Mavridis and N. Papamarkos, "Color quantization using principal components for initialization of Kohonen SOFM", *IEEE Int. Conf. Image Process.*, 1633–1636, 2009.

78. A. Melkman, J. O'Rourke, "On polygonal chain approximation", *Computational Morphology*, 87-95, Amsterdam, 1988.

79. N. Meratnia, R.A. de By, "Spatiotemporal compression techniques for moving point objects", *Proceedings of the Extending Database Technology*, 765-782, 2004.

80. S. Morillas, V. Gregori, and A. Hervas, "Fuzzy peer groups for reduction mixed Gaussian-impulsive noise from color images", *IEEE Trans. Image Process.*, vol. 18, no. 7, 1452–1466, 2009.

81. J. Muckell, J.H. Hwang, C.T. Lawson, S.S. Ravi, "Algorithms for compressing GPS trajectory data: an empirical evaluation", *SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 402-405, 2010.

82. J. Muckell, J.H. Hwang, V. Patil, C.T. Lawson, F. Ping , S.S. Ravi, "SQUISH: an online approach for GPS trajectory compression", *International Conference on Computing for Geospatial Research & Applications*, 1-8, 2011.

83. J. Ni, C.V. Ravishankar, "Indexing Spatio-Temporal Trajectories with Efficient Polynomial Approximations," *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, 663–678, May 2007.

84. E. Ordentlich, G. Seroussi, S. Verdú, M. Weinberger, and T. Weissman, "A discrete universal denoiser and its application to binary images," *IEEE Int. Conf. Image Process.*, 117–120, 2003.

85. E. Ordentlich, M.J. Weinberger, and T. Weissman, "Multi-directional context sets with applications to universal denoising and compression", *IEEE Int. Symp. Inform. Theory*, 1270–1274, 2005.

86. J.C. Perez, E. Vidal, "Optimum polygonal approximation of digitized curves", *Pattern Recognition Letter,* vol. 15, 743–750, 1994.

87. A. Pikaz, I. Dinstein, "An algorithm for polygonal approximation based on iterative point elimination", *Pattern Recognition Letters*, 16 (6): 557-563, 1995.

88. A. Podlasov, E. Ageenko, P. Fränti, "Morphological reconstruction of semantic layers in map images", *Journal of Electronic Imaging*, 15 (1), 013016, January-March 2006.

89. A. Podlasov, P. Kopylov, and P. Fränti, "Statistical filtering of raster map images using a context tree model", *Int. Conf. Signal-Image Technology & Internet-based Systems*, 467–474, 2007.

90. J. Portilla, V. Strela, M. Wainwright, and E. P. Simoncelli, "Image denoising using a scale mixture of Gaussians in the wavelet domain", *IEEE Trans. Image Process.*, vol. 12, no. 11, 1338–1351, 2003.

91. M. Potamias, K. Patroumpas, T. Sellis, "Sampling Trajectory Streams with Spatiotemporal Criteria", *Proceedings of the Scientific and Statistical Database Management* (SSDBM), 275-284, 2006.

92. B.K. Ray, K.S. Ray, "A non-parametric sequential method for polygonal approximation of digital curves", *Pattern Recognition Letter*. 15, 161–167, 1994.

93. M.D. Reavy and C.G. Boncelet, "BACIC: a new method for lossless bi-level and grayscale image compression", *IEEE Int. Conf. on Image Processing*, vol.2, 282-285, 1997.

94. J. Rissanen, "A universal data compression system", *IEEE Trans. Inf. Theory*, vol. 29, no. 5, 656–664, 1983.

95. S. Roth and M.J. Black, "Fields of experts: A framework for learning image priors", *International Journal of Computer Vision*, vol. 82, no.2, 205-229, 2009.

96. H. Sakoe, S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition", *IEEE Trans. on Acoustics, Speech and Signal Processing*, 26(1), 43-39, 1978.

97. D. Salomon, *Data Compression, The Complete Reference*, 4th edition, Springer, 2006.

98. M. Salotti, "Optimal polygonal approximation of digitized curves using the sum of square deviations criterion", *Pattern Recognition*, 35(2), 435-443, 2002.

99. F. Schmid, K.F. Richter and P. Laube, "Semantic Trajectory Compression", *Lecture Notes in Computer Science*, vol. 5644, 411-416, 2009.

100. S. Schulte, S. Morillas, V. Gregori, E.E. Kerre, "A New Fuzzy Color Correlated Impulse Noise Reduction Method", *IEEE Trans. on Image Process.*, 16(10), 2565-2575, 2007.

101. G.M. Schuster and A.K. Katsaggelos, "An optimal polygonal boundary encoding scheme in the rate-distortion sense", *IEEE Trans. on Image Processing*, vol.7, 13-26, 1998.

102. S. Shekhar, S. Huang, Y. Djugash, J. Zhou, "Vector map compression: a clustering approach", *10th ACM Int. Symp. Advances in Geographic Inform*, 74-80, 2002.

103. B. Smolka and A. Chydzinski, "Fast detection and impulsive noise removal in color images", *Real-Time Imaging*, vol. 11, no. 5–6, 389–402, 2005.

104. R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov random fields with smoothness-based priors", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, 1068–1080, 2008.

105. G.T. Toussaint, "On the complexity of approximating polygonal curves in the plane", *IASTED International Symposium on Robotics,* Switzerland, 59-62, 1985.

106. Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, "Image quality assessment: From error visibility to structural similarity", *IEEE Trans. Image Process.*, vol. 13, no. 4, 600–612, 2004.

107. M. Weinberger, J. Rissanen, and R. Arps, "Application of universal context modeling to lossless compression of gray-scale images", *IEEE Trans. Image Process.*, vol. 5, no. 4, 575–586, 1996.

108. T. Weissman, E. Ordentlich, G. Seroussi, S. Verdru, and M. Weinberger, "Universal discrete denoising: Known channel", *IEEE Trans. Inf. Theory*, vol. 51, no. 1, 5–28, 2005.

109. X. Wu, G. Zhai, X. Yang, and W. Zhang, "Adaptive sequential prediction of multidimensional signals with applications to lossless image coding ", *IEEE Trans. on Image Process.*, vol.20, no.1, 36-42, 2011.

110. B. Yang and Q. Li, "Efficient compression of vector data map based on a clustering model", *Geo-spatial Information Science*, 12(1), 13-17, 2009.

111. J. Yu and S. Verd´u, "Schemes for bidirectional modeling of discrete stationary sources", *IEEE Trans. Inform. Theory*, vol. 52, no. 11, 4789–4807, 2006.

112. Z. Yu, X. Zhou, "Computing with Spatial Trajectories", *Springer*, 2011.

113. Q. Zhao, V. Hautamäki, I. Kärkkäinen, and P. Fränti, "Random swap EM algorithm for finite mixtures models in image segmentation", *IEEE Int. Conf. Image Processing*, 2397–2400, 2009.

114. H. Zhou and K. Mao, "An impulsive noise color image filter using learning-based color morphological operation", *Digit. Signal Process.*, vol. 18, 406–421, 2008.

115. X. Zhu, P. Milanfar, "A no-reference image content metric and its application to denoising", *IEEE Int. Conf. Image Processing*, 1145–1148, 2010.

Paper P1

M. Chen, M. Xu and P. Fränti, "Multi-layer Filtering Approach for Map Images", *IEEE Int. Conf. on Image Processing* (ICIP'09), Cairo, Egypt, 3953-3956, 2009.

Multi-layer Filtering Approach for Map Images

Minjie Chen[1], Mantao Xu[1], Pasi Fränti[1,2]
1. University of Joensuu, Finland; 2.Fudan University, Shanghai, China

ABSTRACT

Raster map image is an important set of color images with similar patterns and textures presented in a limited number of colors. Manipulating this class of color images using conventional image filters may lead to a severe over-filtering problem, by which important details structures are over eliminated or degraded. Even if the statistical based algorithms have been recognized as a set of most efficient filters, their exponentially high memory consumption and computational cost can make them intractable in practice. To solve this operational difficulty, this work proposed a novel multi-layer image filtering approach transforming map image filtering into binary domain. It consists of three intuitive image operators: layer decomposition, binary image filters and layer merging. Experimental results demonstrated that the new proposed approach is very efficient for filtering of raster map image.

Index Terms—Digital image processing, filtering.

1. INTRODUCTION

A class of widely used color images in geosciences is raster map images that are encoded in a regular grid of pixel colors arrayed in rows and columns. In contrast to the conventional color images (e.g., photographic images), raster map images do not allow smooth continuous tone changes in colors but merely display pixel level repetitive detail structures, which are essential for interpreting map objects. This is because each color in the image represents a different semantic map object. However, in many applications, the color information of map images may be distorted by lossy compressions, vector-to-raster conversion, or during the digitization process itself. This color distortion may lead to a severe false recognition of map objects.

A common way to address this problem is to apply image filtering before further processing. The main challenge in color image filtering is that most of color images contain multivariate data with color correlations, which make the design of image filters very difficult in practice. A proven set of intuitive color image filters is to apply gray-scale image filters for each channel of color images separately. However, this approach does not take into account the necessary color correlations, which results in a production of false colors and edge degradation. To overcome this difficulty, many filters have been studied using nonlinear approaches. For instance, a class of efficient color image filters can be designed using a set of ordering criterions or order-statistics for color vectors [3]. However, in most cases, the set of ordering criterions must be defined in terms of the distance between two color vectors, which is usually not appropriate for our case.

This problem can be partly revealed by applying the ordering criterions only to the pixels that are identified (or assumed) as noises or outliers by using *adaptive vector median* [2] or *fast peer group filter* [4]. However, nonlinear filters designed for conventional color images may eliminate the most useful edge information (e.g., thin edges containing important information) when they are applied to map images. This is because each color in map images represents a distinct class of map objects, which forces the conventional filters to over-smooth the more detailed structures. Even though statistical modeling approaches [8, 9] have made significant progress by learning image structure and preserving the repetitive structures, their memory consumption and computational expense are exponentially high.

In this paper, we propose a multi-layer image filtering approach. Instead of using an order-statistics filter for color vectors (e.g., the adaptive vector median filter), the image I is first decomposed into a series of binary layers. For a given pixel (i,j), amongst N number of layers of $\{L_t(i,j)|\ t = 1,\ldots N\}$ there exists only one layer t where the pixel belongs to: $L_t(i,j)= 1$. Then each binary layer is processed separately by a binary image filtering method. In this way, we avoid the problem of dealing with color distances. Instead, the problem is reduced back into a simpler (binary) domain of which set operations (as in morphological filters), or rank-order operations can be applied.

For the sake of reconstructing the resulting filtered color image I^* from the set of binary layers $\{L^*_t \mid t = 1,\ldots N\}$, the layers must be ordered according to the color priority. Namely, for a given pixel (i,j), its output color in the filtered image, $I_G(i,j)$ must be selected from those layers such that $\{L^*_t(i,j)|\ L^*_t(i,j) \neq 0,\ t = 1,\ldots N\}$. For this reason, prioritizing the colors is needed. We apply region-based ordering where the image is first divided into segments having different background color, and then deriving the color propriety for each segment locally.

2. MULTI-LAYER DECOMPOSITION OF IMAGES

The use of multi-layer decomposition was originally proposed in [5] for image compression. We extend this

framework to the filtering of map image. The basic idea behind the multi-layer approach is illustrated in Fig. 1. The image decomposition is straightforward, and the following filtering can be performed using any binary image filtering method. The last step is to merge the layers back to a color image, which requires some attention.

Here, (i,j) denotes a pixel of a image I with limited number of colors, i.e. $N << 256$. The output color for the pixel, $I(i,j)$, can be uniquely determined by a binary vector, $\mathbf{x}_L$,

$$\mathbf{x}_L = (L_1(i,j),\ldots L_N(i,j)) \tag{1}$$

where each component x_k such that:

$$L_t(i,j) = \begin{cases} 1, \text{if } t = I(i,j) \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

Here, $L_t(i,j)$ can be treated as the value of pixel (i,j) for t^{th} binary layer. Clearly, any filter applicable to binary images can be performed on each of the N number of binary layers, e.g., median filter, morphological filter [7] and the statistical filter [8]. Once each of binary layers is processed, the resulting binary layers, $\{L^*_t \mid t = 1,\ldots N\}$, are merged to reconstruct the output color image, I_G. However, for a given pixel (i,j), there might be several layers L_C where the pixel is set to 1: $\{ L^*_t \mid L^*_t(i,j) = 1, 1<t<N\}$. Since the resulting only one color can be assigned to each pixel, it must be selected from the set L_C according to some criterion. We make the selection based on color priority so that the output color is selected as the color layer with the highest priority:

$$I_G(i,j) = \begin{cases} \arg\max_t \sum_t L^*_t(i,j) = 1 \\ 1, \text{ otherwise} \end{cases} \tag{3}$$

In other words, the decomposed binary layers must be ordered for the sake of merging the filtered binary images using (3). In image compression, graph based algorithm [10] has been applied to find optimal ordering of the binary layers using minimum spanning tree algorithm based on a compression cost matrix. However, the construction of the cost matrix have huge computational cost, which may not be feasible for filtering, and our criterion should be based on maximizing information in the image but not minimizing it.

One useful heuristic is to order the layers according the frequency of object pixels appearing on it. The higher is the occurrence of a color, the lower priority is assigned to it. The idea is that frequent colors represent either background or other large objects, whereas infrequent colors represent finer details, which are more likely to be more important for the quality of the image. The ordering of the colors (binary layers) is the same for every pixel, and is therefore denoted as global layer ordering. An example of this is shown in Fig. 1. The drawback of this approach is that the importance of the colors can be different in different regions. For example, the dominant (background) color that should be given the lowest priority is usually white in the map shown in Fig. 1, but in water areas it is blue and in some places it can be yellow (fields). To overcome this problem, we propose a region based algorithm for ordering the binary layers to localize the choice of color priority.

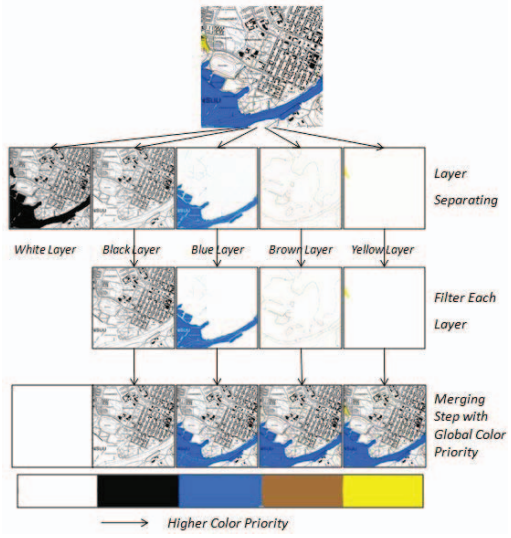

Fig. 1 *Multi-layer framework using global layer ordering*

3. REGION BASED ORDERING OF THE LAYERS

Merging the filtered binary images using global ordering results in a severe problem of damaging important disconnected structures. For example, the island inside the lake region in Fig. 1 disappeared as a result of the merging because the global layer ordering scheme assigns the *blue* color with higher priority than the *white* color, thus destroys the small island.

A region based ordering scheme is proposed for merging the binary layers L^* as follows. A multi-layer based image segmentation operation was firstly conducted to segment the raster map image into several distinct regions, $S = \{s_k \mid \cup s_k = I, k = 1,\ldots K,$ and $s_k \cap s_l = \phi, \forall\ 1<k, l < K\}$. After this preliminary segmentation, the color priority for each image pixel (i,j) is calculated according to the color occurrence in its region S_k, where $(i,j) \in S_k$. The main idea of exploiting this region based color ordering scheme is to incorporate the statistical features behind those disconnected semantic regions into image filters. For the sake of image segmentation, the conventional gradient-based segmentation algorithms in [6] are seemingly not applicable due to an inadequate number of colors for extracting the edge information. Hence, instead of performing image segmentation on the input raster map, a multi-layer based image segmentation algorithm is applied. After a dilation, hole fillings and region labeling operation on all the filtered binary layers L^*, a set of large-sized candidate or initial regions (M number of candidate regions) are extracted from all these layers. For simplicity of implementation, those

refined candidate regions are maintained in a list of size M, $R = \{R_m \mid, m = 1,\ldots M\}$ in a descending order according to their pixel size. These candidate regions in the list are evaluated one by one according to two region based features (f_1 and f_2). The first feature (f_1) is the number of object pixels in the candidate region relative to the size of the region:

$$f_1(R_m) = \frac{\sum_{(i,j)\in R_m} L^*_{t(m)}(i,j)}{|R_m|} \quad (4)$$

where R_m is the candidate region in the $t(m)^{\text{th}}$ layer. The second feature (f_2) is the percentage of labeled pixels in R_m so far:

$$f_2(R_m) = \frac{|S_M(i,j) \neq 0 \cap (i,j) \in R_m|}{|R_m|} \quad (5)$$

where S_M is a label mask image for the image segmentation S such that $s_k = \{(i,j) \mid S_M(i,j) = k, (i,j) \in I\}$. The pseudocodes of the underlying image segmentation algorithm can be found in Fig. 3.

Once the label mask image has been obtained using the algorithm in Fig. 3, the remaining non-labeled pixels ($S_M(i,j)$=0) are processed by distance transform algorithm. It assigns these pixels the same label as their nearest segmented region.

For the final region s_k in S, its color frequency is calculated, and the local color ordering within this region is determined by sorting the color from lowest to highest frequency. A demonstration of the segmentation results is given in Fig. 2, by showing the background color (the lowest priority color) of each segmented region.

The overall algorithm is demonstrated in Fig. 4 by showing the resulting regions, and their corresponding color priorities. The merging process is also demonstrated where the image is composed step-by-step. At first step, the background colors are added for each region, and the remaining colors then one by one.

Fig. 2 *Background color for each segmented region*

```
INPUT R ← the M-sized list of candidate regions
      L* ← the N layers of filtered binary images
OUTPUT: S_M ← label mask image
FUNCTION MultiLayerSegmentation (R*, L*) RETURN S_M
S_M ← 0;
r ← 1;
FOR m = 1 to M
    f1 ← calculate ratio of object pixel according to (4)
    f2 ← calculate percentage of labeled pixel according to (5)
    IF AcceptanceCriterion (f1, f2) is TRUE
        FOR every pixel (i,j) ∈ R_m
            S_M(i,j) ← r
        END FOR
        r ← r + 1
    END IF
END FOR

PROCEDURE AcceptanceCriterion (f1, f2) REUTRN newSeg
newSeg ← FALSE
IF f1 > 0.52 and f2 < 0.001
   newSeg ← TRUE
ELSE IF f1 > 0.6 and f2 < 0.005
   newSeg ← TRUE
ELSE IF f1 > 0.8
   newSeg ← TRUE
ELSE
   newSeg ← FALSE
END
```

Fig. 3 *Pseudocodes of multi-layer image segmentation algorithm*

4. RESULT AND DISCUSSION

We have evaluated the proposed multi-layer filtering algorithm using a set of six map images from the database provided by [1]. These images are presented by 5-16 colors, and they are of different spatial resolutions. Some of them include quantization noise whereas others are converted from vector origin. For evaluating the performance of the image filters, we artificially distort those images by adding impulsive and content-dependent noise as described in [9].

As a performance comparison, four alternative filters for color images are studied. *Adaptive vector median* (AVM) [2] and *color morphological* (MM) [3] are the filters designed for general color image filtering, while *fast peer group filter* (FPGF) [4] and *context-tree filter* (CT) [9] are tailored methods for color-index image. We measure the mean color distance (ΔE) in L*a*b color space. Although this measure does not match completely how human see the image, it gives rough idea about the relative performance of the filters.

For implementation of the proposed multi-layer image filtering approach, two binary filters, *spatially-variant morphology* [7] and *discrete universal denoising* [8] are incorporated into the framework of filtering each decomposed binary layers. We termed them as MSM and MUD in Table 1 respectively.

The objective experiment results obtained by using the six raster map image filters are summarized reported in Table 1 and subjective comparison in Fig. 5. It turns outs from the results that the proposed multi-layer method can work more efficiently than the other four filters when image is converted from vector origin with less color number (Image#1, 2).When those map images are generated by quantization of scanning maps with more colors, some meaningless layers may exist because of the inaccurate quantization, which make the frequency-based ordering does not work.

Fig. 4 *An example of merging of filtered binary images (right) using the region based layer ordering (left)*

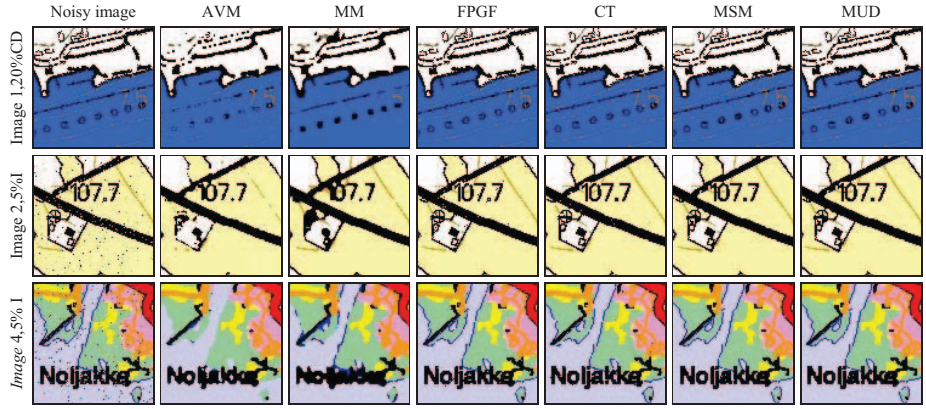

Fig. 5 *Performance comparison of the six image filters tested in the experiments*

Table 1 the filter efficiency for 5% impulsive noise (I) and 20% content-dependent noise (CD)

ΔE	Image	1	2	3	4	5	6
	Colors	5	5	9	10	16	16
I	AVM	3.47	5.81	8.98	7.83	1.68	6.18
	MM	4.80	4.16	12.0	10.0	2.60	12.8
	FPGF	0.95	1.00	1.77	1.45	0.87	3.35
	CT	0.77	0.95	1.54	1.97	0.83	2.24
	MSM	0.99	1.21	3.10	2.56	1.67	8.09
	MUD	0.68	0.89	1.87	1.69	1.06	3.88
CD	AVM	4.14	5.58	9.26	8.07	1.89	7.21
	MM	4.61	4.31	12.0	9.96	2.73	12.5
	FPGF	1.99	1.70	3.51	3.35	1.42	5.02
	CT	2.08	1.91	3.15	3.66	1.17	3.56
	MSM	2.03	2.02	4.52	4.05	2.05	9.01
	MUD	2.04	1.81	3.37	3.23	1.39	5.22

5. CONCLUSION

We have proposed a multi-layer approach for filtering algorithm for raster map images. This proposed method provided a solution for processing map image in binary domain. It also has lower computation cost and memory consumption comparing to statistical method. Experimental results have validated that the proposed algorithm is very efficient for filtering raster map images.

Future work can be done in two aspects: refining the proposed filtering algorithm in denoising the true color images and optimizing the region based layer ordering using shape analysis.

6. REFERENCES

[1] National Land Survey of Finland, (http://www.nls.fi).
[2] R. Lukac, "Adaptive vector median filtering", *Pattern Recognition Letters* 24, pp. 1889-1899, 2003.
[3] G. Louverdis, M. Vardavoulia, I. Andreadis, P. Tsalides, "A new approach to morphological color image processing", *Pattern Recognition* 35(8), pp. 1733–1741,2002.
[4] B. Smolka, A. Chydzinski, "Fast detection and impulsive noise removal in color images", *Real-Time Imaging*, 11, pp.389-402,2005.
[5] S. Forchhhammer, O. Jensen, "Content Layer Progressive Coding of Digital Maps", *IEEE Trans. on Image Processing*, 11(12), pp.1349-1356, 2002.
[6] J. Angulo, J. Serra, "Modelling and segmentation of colour images in polar representations", *Image and Vision Computing* 25, pp. 475-495, 2007.
[7] N. Bouaynaya, M. Charif-Chefchaouni, D. Schonfeld, "Theoretical Foundations of Spatially-Variant Mathematical Morphology Part I: Binary Images", *IEEE Trans. on Pattern Anal. and Machine Intelligence*, 30(5), pp.823-836, 2008.
[8] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdru, and M. Weinberger, "Universal discrete denoising: Known channel," *IEEE Trans. Inform. Theory*, 51(1), pp.5-28, 2005.
[9] P. Kopylov and P. Fränti, "Filtering of color map images by context tree modeling", *IEEE Int. Conf. on Image Processing (ICIP'04)*, Singapore, vol. 1, pp. 267-270, 2004.
[10] P. Kopylov and P. Fränti, "Compression of map images by multilayer context tree modeling", *IEEE Trans. on Image Processing*, 14(1), pp.1-11, 2005.

Paper P2

M. Chen, M. Xu, P. Fränti, "Adaptive Context-tree-based Statistical Filtering of Raster Map Images Denoising", *IEEE Trans. on Multimedia*, 13(6), 1195-1207, 2011.

Adaptive Context-Tree-Based Statistical Filtering for Raster Map Image Denoising

Minjie Chen, *Student Member, IEEE*, Mantao Xu, and Pasi Fränti, *Senior Member, IEEE*

***Abstract*—Filtering of raster map images is chosen as a case study of a more general class of palette-indexed images for the denoising problem of images with a discrete number of output colors. Statistical features of local context are analyzed to avoid damage to pixel-level patterns, which is frequently caused by conventional filters. We apply a universal statistical filter using context-tree modeling via a selective context expansion capturing those pixel combinations that are present in the image. The selective context expansion makes it possible to use a much larger spatial neighborhood, with a feasible time and memory complexity, than fixed-size templates. We improve the existing context-tree approaches in two aspects: Firstly, in order to circumvent the context contamination problem, a context-merging strategy is applied where multiple similar contexts are considered in the conditional probability estimation procedure. Secondly, we study a specific continuous-input-finite-output problem in which the map images are corrupted by additive Gaussian noise. Performance comparisons with competitive filters demonstrate that the proposed algorithm provides robust noise filtering performance and good structure preservation in all test cases without any *a priori* information on the statistical properties of the noise.**

***Index Terms*—Context-tree modeling, raster map image, statistical filtering.**

I. Introduction

GEOGRAPHICAL map images are classified into two formats: raster and vector. Vector format is suitable for large databases, providing excellent flexibility for display and compact storage size. A raster image, on the other hand, is encoded in a regular grid of pixel colors in which each color represents a different class of semantic map object. It consists of pixel-level detailed structures and sharp edges but lacks smooth color transitions that are typical for photographic images. It does not require any additional image processing and is suitable for delivery to multimedia applications as such.

Manuscript received October 31, 2010; revised March 08, 2011 and June 29, 2011; accepted August 15, 2011. Date of publication August 30, 2011; date of current version November 18, 2011. The work of M. Chen was supported by East Finland Graduate School in Computer Science and Engineering (ECSE), MOPSI project EU (EAKR), Tekniikan edistämissäätiö (TES), and Nokia Scholarship. The work of M. Xu was supported by China NSF (Grant No 61072146) and Shanghai SCST (Grant No 10PJ1404400). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Zicheng Liu.

M. Chen and Pasi Fränti are with the School of Computing, University of Eastern Finland, Joensuu, Finland (e-mail: mchen@cs.joensuu.fi; franti@cs.joensuu.fi).

M. Xu is with the School of Electrical Engineering, Shanghai Dianji University, Shanghai, China (e-mail: xumt@sdju.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2011.2166538

Raster maps are an important source of geospatial information due to popularized *geographic information systems*. A large number of topographic maps have been collected during the last century, and they contain geographic features that can be used in the verification of map-image-guided navigation applications. Advances in digital information technology have brought the conversion of these older physical documents into digitized representations. However, the digitization of maps may incur additional noise and errors in the digital versions. Aging of paper archived for a long period also leads to additional random color variations. This kind of image degradation results in mismatch and false recognition of important semantic map objects. Image denoising is therefore needed for accurate conversion of these older maps into raster format. This preprocess can be crucial for the later raster map analysis step, when extracting the semantic content (roads, contours, river) on a map [31]–[34]. Image denoising can also be applied as a preprocessing when converting a raster map into a vector format.

A great variety of noise removal techniques have been investigated for color image processing. However, most noise removal algorithms are developed for only one type of noise model specifically. For instance, to eliminate impulsive noise, a number of denoising algorithms have been developed by first identifying the potential noisy pixels in the color image and then employing a class of vector median filters over those detected noisy pixels. Noisy pixels can be detected either by classifying each pixel directly in RGB color space [1] or by setting some statistical rules in terms of the variation in the local neighborhood [2], [3]. However, these approaches need a training dataset or prior knowledge for constructing the statistical rules. A multi-layer approach was recently proposed [4] to solve the problem in the binary domain.

For additive Gaussian noise, a number of state-of-the-art denoising algorithms have been proposed by selecting an optimal linear combination of a few basis elements in pixel-wise or block-wise order. These pioneer denoising techniques include *wavelet denoising* [5], *non-local means* [6], *dictionary-based method* (K-SVD) [7], *block-matching and 3-D filtering* [8], *Markov random fields*, and *active random fields* [9], [10]. However, they are limited to the case when the true signal can be approximated by a linear combination of a few basis elements, and therefore, they are designed for denoising continuous tone images and do not apply well for palette-indexed images.

Raster map images contain a number of complicated spatial structures such as one-pixel thin lines, textured areas, dashed and dotted lines, text, and symbols. The problem of false filtering exists with most filters designed for photographic imagery when processing these kinds of spatial structures. This is

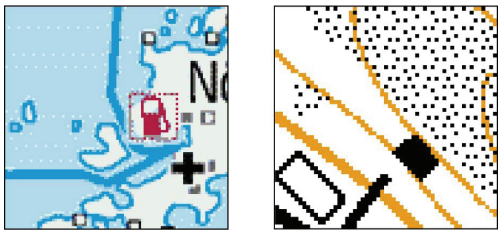

Fig. 1. Examples of complicated structures that are treated as noise by most filters.

because these filters consider local intensity variation as noise but ignore repeated patterns in the entire image. On the other hand, high variance regions including written text, symbols, and textured background lack uniformity but their presence is vital for the readability of the map. Examples of such structures are shown in Fig. 1.

A pioneer work in the art of statistical filtering is the so-called *discrete universal denoising* (DUDE) [11] in design of a filter for binary data with a known noise channel, which comprises two steps: counting statistics for all context patterns encountered (*analysis step*), and denoising by utilizing the conditional probability in a local context (*denoising step*). This method is applicable in denoising of binary images if the error probability δ can be reliably estimated. Namely, if the conditional probability of the current pixel x in its context $P(I(x)\,|\,\mathbf{c})$ is lower than $2\delta(1-\delta)$, it is considered as noise and replaced by its complementary value.

This kind of context-based approach can be extended to the denoising problem with an unknown channel using the min-max criterion [12]. In contrast to the previous algorithms [1]–[10] that incorporate a prior model, statistical filtering is based on an unsupervised learning paradigm. Patterns that are frequently presented in the image are detected and considered as important image structures that should be preserved. On the contrary, pixels that appear seldom in their context are treated as noise and can be filtered out. This allows filtering with preservation of borders and regular structures regardless of their size and variance.

However, the memory allocation for learning the patterns grows exponentially with the number of pixels within the pattern (context), which makes it of limited use in practice. Moreover, the conditional probability estimation becomes inaccurate for those contexts with rare appearance, which is known as the *context dilution problem*. To circumvent this problem, *context-tree modeling* [13], [14], [19] has been applied by pruning redundant nodes of the context tree. This can be done according to different criteria, such as fixed frequency [13], maximum likelihood [15], or the coding cost of the model [14], [16]. Adaptive context selection has also been extended for denoising gray-scale images [17] by using a *minimum description length* (MDL) guided criterion aimed at finding an optimal balance between the variance and bias of the errors in fitting a 2-D *piecewise autoregressive* (PAR) model to input images.

In this paper, we propose a new adaptive context-tree-based statistical filtering algorithm for raster map images. In contrast to the existing context-tree-based methods, we achieve two significant improvements described below.

Firstly, although the context-tree-based algorithms are efficient for the images with a smaller number of noisy pixels, the contexts themselves will embrace a significant number of noisy pixels when noise level is increased in the image. Including noisy pixels (outliers) in the surrounding contexts will make it difficult to estimate a good conditional probability distribution for context models. We call this the *context contamination problem*, when the contexts themselves contain a number of outliers. For those infrequently appearing contaminated contexts, we present a *context-merging* strategy in order to improve conditional probability estimation. In our method, multiple similar contexts are considered jointly in the conditional probability estimation procedure.

Secondly, we extend the algorithm to deal with map images with additive Gaussian noise or mixed Gaussian-impulsive noise. The extended method iteratively conducts a fusion procedure according to the probability distribution of pixels' intensity in the RGB space as well as their conditional probabilities of contexts.

The rest of the paper is organized as follows. The proposed method is introduced in Section II, experimental results are reported in Section III, and finally, conclusions are drawn in Section IV. A preliminary version of this paper has been presented at ICME [30].

II. Proposed Method

For concreteness, a noise-free map image can be formulated as follows: A clean map image $\mathbf{X}$ has M colors (size of color palette) such that the alphabet $\mathrm{A} = \{1, 2, \ldots, M\}$ includes all possible index values in the image. For any pixel $x \in \mathbf{X}$ with index value $I(x)$, its corresponding color in RGB space is

$$\mathbf{X}(x) = \mathbf{m}_{I(x)} = \left[m_{I(x)}(r), m_{I(x)}(g), m_{I(x)}(b)\right] \quad (1)$$

according to the color palette $\mathrm{CP} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_M\}$. Impulsive noise is often produced in the transmission of the clean image signal $I(x)$ over *an M-ary Symmetric Channel* with transition matrix $\mathbf{\Pi}$:

$$\prod(i,j) = \begin{cases} 1-\delta, & i = j \\ \delta/(M-1), & i \neq j \end{cases} \quad (2)$$

where $\Pi(i,j)$ denotes the probability of output index j when the input index is i, and δ is the noise level. The performance of denoising impulsive noise can be evaluated using the following measures:

1) *Error rate*: the percentage of different pixel values between the noise-free and filtered images.
2) *False acceptance rate* (FA): the percentage of noise to survive after filtering (efficiency of noise removal).
3) *False rejection rate* (FR): the probability of an original (clean) image pixel being filtered (amount of corruption imposed by the filter).

Since the clean image $\mathbf{X}$ is also an RGB image, the corresponding noisy image $\mathbf{Y}$ can be produced by adding the same

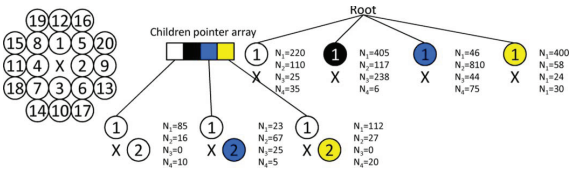

Fig. 2. Part of the context tree (first two levels) and its context template.

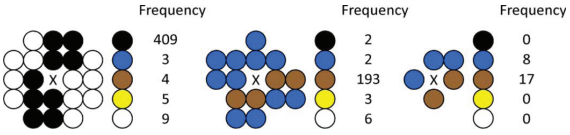

Fig. 3. Examples of context distribution: the pixels in the left and middle contexts are filtered using the dominant color whereas no filtering is done for the pixel in the context on the right.

additive Gaussian noise with variance σ to each color channel independently:

$$\mathbf{Y}(x) = \mathbf{X}(x) + \boldsymbol{\varepsilon},\ \text{where}\ \boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2\mathbf{I}) \tag{3}$$

where $\mathbf{Y}(x)$ and $\mathbf{X}(\mathrm{x})$ are the RGB color vectors of pixel x. The quality of denoising of the image with Gaussian noise can be evaluated by:

1) PSNR: Peak signal-to-noise ratio in RGB color space.
2) SSIM: Structural similarity index [21].

A. Context-Based Statistical Filter

Suppose that the clean image $\mathbf{X}$ contains M colors and M is small. A context $\mathbf{c} = \{x_1, \ldots, x_k\}$ can be defined as a set of k pixels. A sample 20-pixel context template is shown in Fig. 2(left), where the current pixel x is marked with "×". For simplicity, a context $\mathbf{c}$ of pixel x is denoted as $x \in \mathbf{c}$. The context $\mathbf{c}$ can be associated with a vector $n_{\mathbf{c}} = (n_{\mathbf{c}}(1), \ldots, n_{\mathbf{c}}(M))$ called a *vector of statistics* for the current pixel x, where $n_{\mathbf{c}}(i)$ is the count statistics of index value i. After the *vectors of statistics* have been collected for every context of the image, the conditional probability of every pixel appearing in its context is estimated as

$$P(I(x) = j \,|\, x \in \mathbf{c}) = \frac{n_{\mathbf{c}}(j)}{\sum_{i=1}^{M} n_{\mathbf{c}}(i)}. \tag{4}$$

For simplicity, we denote the conditional probability in the context $\mathbf{c}$ as $P(I(x) \,|\, \mathbf{c})$.

Statistical filtering can be performed in a two-pass procedure, of which the first pass is the estimation of conditional probabilities (statistical context modeling) and the second pass is the filtering of the noisy pixels. The main idea of the statistical filter follows an assumption that the image signal originates from a universal source. Hence, if the conditional probability $P(I(x) \,|\, \mathbf{c})$ in context $\mathbf{c}$ is less than a predefined value, the current pixel can be treated as noise and then replaced by the most probable color in the context. Three examples of contexts and their corresponding statistical distributions are demonstrated in Fig. 3. Domination of the most probable color can be observed in the first two examples (left and middle) but not in the last example (right).

B. Context-Tree Modeling

In practical implementation, we optimize memory allocation using *context-tree modeling*. The classical *context-tree modeling* technique has been widely used in the field of data compression with a time complexity of $O(kN)$, where N is the length of the data sequence and k is the depth of the context tree. The tree is built by estimating the count statistics via a sequential traversal of the image pixel by pixel. Each node of the context tree represents a single context by storing the count statistics of each color appearing for the current pixel relative to this context. Since not all possible contexts are present in the image, memory is allocated only for the actual pixel combinations appearing in the image. In our implementation, the spanning of the tree is terminated if the frequency of the context on a given node becomes less than a predefined threshold value T. According to our experiments, there are only 50 000–100 000 contexts for a 20-pixel context template in a 16 color map image, which is far below 16^{20}. An example of context-tree modeling is shown in Fig. 2, where each node represents a single context including the count statistics of each color for the current pixel in respect to the context. An example of the context distribution is shown in Fig. 3.

C. Statistical Filtering by Context-Tree Modeling

Context-tree modeling has been extensively studied in the problem of image compression [28], [29]. In image compression, all pixels must be encoded regardless of the reliability of their contexts. Moreover, one can keep track of the compression performance. Poor probability estimates only lead to a longer code length and thus a large file size. Optimal pruning of a context tree can be done on each node in order to achieve the highest overall compression performance. For instance, a dynamic programming pruning technique was proposed to improve context selection in [14], whereas universal context modeling was employed in [18].

However, conditional probability estimation plays a crucial role in image denoising. Wrongly estimated conditional probability can cause either a lack of detection of a noisy pixel by the algorithm or changing of a clean pixel, causing new noise. In contrast to image compression, several challenges exist when applying statistical context modeling for image denoising. Firstly, the distribution of noise is seldom known, and therefore, it is hard to estimate in practice. Secondly, the contexts themselves may include a significant number of noisy pixels and lead to a so-called *context contamination problem*. If the neighborhood pixels were contaminated by erroneous colors, the particular context would appear infrequently in the image. This causes an inaccurate estimation of the conditional probability distribution. Thirdly, a proper decision rule for the filtering is not trivial to design.

For the improvement of the statistical filter, we will discuss the following three design problems:

a) how to determine the decision rule for filtering;
b) how to calculate the conditional probability of the infrequent contexts;
c) how to estimate the noise level of the image.

a) Decision Rule for Pixel Denoising: Suppose we have a map image with impulsive noise generated by transmitting a clean image $\mathbf{X}$ over *an M-ary Symmetric Channel*. The optimal decision rule in DUDE [11] is essentially a *MAP estimator*, which is

$$u_0 = \arg \max_{I(x)\in\{1,\ldots,M\}} P(I(x)\,|\,\mathbf{c}) \tag{5}$$

where $\mathbf{c}$ is the context of the pixel x. In image filtering, the current pixel x will be replaced by u_0, which is the index value of the highest probability if the decision rule in (6) is met:[1]

$$\frac{(M-1)^2(1-\delta)}{\delta((1-\delta)M-1)}P(I(x)=x_0\,|\,\mathbf{c}) - \frac{(M-1)}{((1-\delta)M-1)}P(I(x)=u_0\,|\,\mathbf{c}) < 1, \quad x_0 = 1,2,\ldots,M. \tag{6}$$

b) Improve Conditional Probability Estimation by a Context-Merging Strategy: Although DUDE follows a so-called "*asymptotic optimality*" property, it requires an infinite sequence of data source for estimating all the conditional distributions of contexts, which is not realistic in practice. In particular, when the context of the pixel is contaminated by erroneous colors, the context can appear infrequently and its associated conditional probability would be far from its true distribution. In order to alleviate this problem, *context-tree modeling* is used by terminating the tree spanning if the frequency of the context becomes less than a given threshold T. In Fig. 5(left), the number of noisy pixels in the context is compared between a fixed-template context and the *context-tree modeling*. No more than two noisy pixels are observed in most contexts regardless of the noise level.

A pruning step is added to remove those contexts as in [13] and [19] if the frequency of a context is less than a predefined threshold T. After pruning, the statistics of its parent node are used for the probability estimation instead. The main challenge for the pruning is that the tree is constructed in a fixed order, and the noisy pixels may appear anywhere in the tree, not just in the leaf nodes. Thus, a clean pixel may also be removed from the context in many cases.

To this end, we will present a *context-merging* strategy for those infrequent contexts that are expected to be contaminated. For each context $\mathbf{c}$, we first construct a set of sub-contexts:

$$S(\mathbf{c}) = \{\mathbf{z}_i\,|\,\mathbf{z}_i = \{\mathbf{c}/x_i\}\}_{i=1}^{k} \tag{7}$$

by removing the ith element from the original context $\mathbf{c}$, where $i = 1,\ldots,k$, and $\mathbf{z}_i$ is the sub-context. Without loss of generality, we sum up all the *vectors of statistics* of the sub-contexts as the estimated distribution of the original context $\mathbf{c}$ if the frequency of the context is lower than T. See (8) at the bottom of the page. The idea of this *context-merging* strategy is that the sub-context will appear much more frequently in the image if the noisy pixel is removed from the context. On the contrary, removing a clean pixel will not change the statistics much. For example, in Fig. 4 it is difficult to conclude anything from the original context as it is so infrequent. However, by analyzing the statistics of the sub-contexts we can see that the black pixel is the noisy one and should be removed. After this *context-merging* operation, only noise-free sub-contexts become dominant in the summation of all the sub-context distributions, which serves as a good estimation of the conditional probability for the context model.

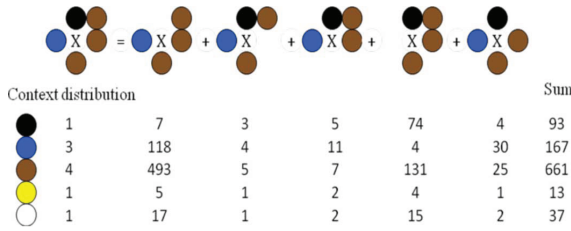

Fig. 4. Example of context-merging strategy. A more reliable context distribution (93, 167, 661, 13, 37) is obtained instead of the estimation (4, 30, 25, 1, 2) obtained by the pruning operation. Colors with low probability (yellow and white) are replaced by the dominant color (brown).

The effect of the *context-merging* strategy is evaluated in Fig. 5(middle and right) when the contexts contain different numbers of noisy pixels. We can observe that the proposed *context-merging* strategy achieves a lower *error rate* and *false acceptance rate* compared to the previous context-pruning method. Even after the *context-merging* process, some sub-contexts are still corrupted by one noisy pixel but they have become frequent enough to be useful. Those contexts can be considered statistically significant for the filtering and they mostly appear in the background region. That is why the proposed *context-merging* strategy retains its efficiency even when two noisy pixels appear in the context.

Performance comparisons are also made for three approaches in Fig. 6(left): a fixed-template context algorithm with different context sizes k, a context-pruning algorithm with different thresholds T, and a *context-merging* algorithm with different thresholds T. We can observe that a lower *false acceptance rate* is achieved at the cost of an increased *false rejection rate* when a greater T is used. In other words, more noisy pixels are filtered out with a trade-off that more clean pixels are replaced by a wrong color. Moreover, it is observed in Fig. 6(middle)

[1]This decision rule is designed for the count statistics collected on the noisy image. For a clean image, the decision rule is $P(I(x) = x_0\,|\,\mathbf{c}) < \delta/(M-1), x_0 = 1,2,\ldots,M$ instead.

$$P*(I(x)=x_0\,|\,x\in\mathbf{c}) = \begin{cases} P(I(x)=x_0\,|\,x\in S(\mathbf{c})), & \text{if } \sum_{i=1}^{M} n_{\mathbf{c}}(i) < T \\ P(I(x)=x_0\,|\,x\in\mathbf{c}), & \text{otherwise} \end{cases} \tag{8}$$

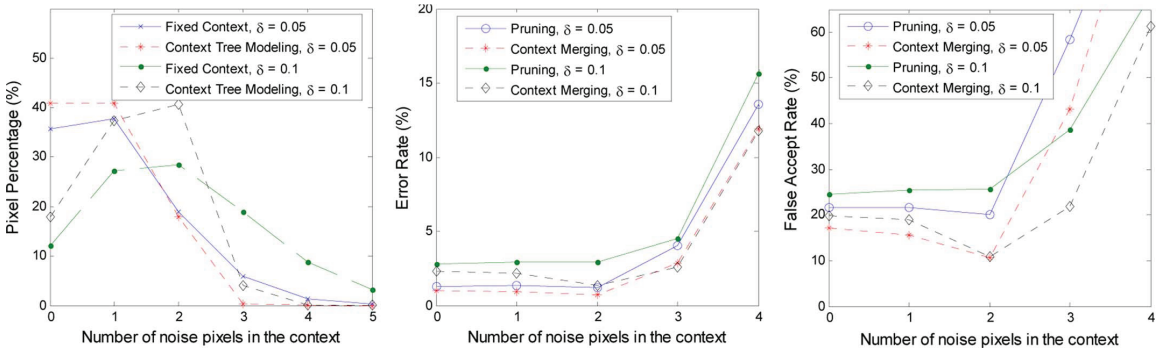

Fig. 5. Comparison of the denoising performance when different numbers of noisy pixels are included in the context. The numbers of noisy pixels in the context are compared for a fixed-template context and context-tree modeling, respectively (left). The *error rate* (middle) and *false acceptance rate* (right) are also evaluated, respectively. Image #1-26 is used is this example with $T = 128$.

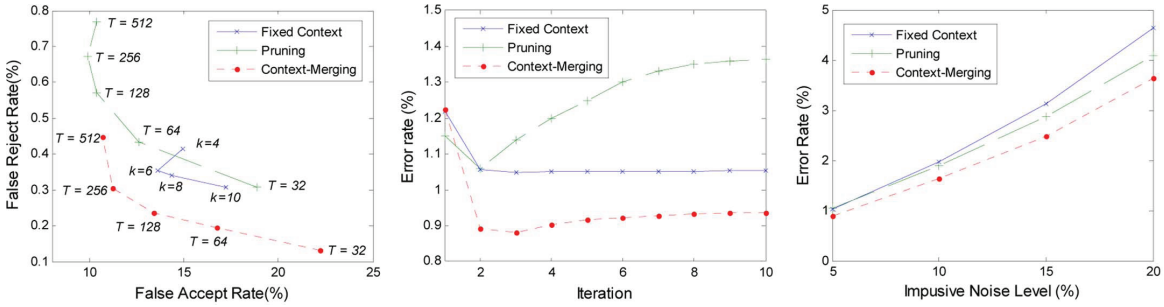

Fig. 6. *False acceptance rate* (FA), *false rejection rate* (FR), and *error rate* are evaluated on different conditional probability estimation algorithms for filtering impulsive noise. The performances are compared by selecting different thresholds T (left), different iterations (middle), and different noise levels (right). Image #2-05 is used as the test image by adding 5% impulsive noise (left and middle), while $T = 128$ and $k = 8$ are used in the middle and on the right.

that further image degradation is not caused even after running several iterations of the proposed statistical filtering algorithm.[2] Image denoising performance is evaluated with different noise levels in Fig. 6(right).

The computational complexity of the merging process is calculated as follows: for each infrequent context, the statistical distributions of M^2 (M is the size of the color palette) similar contexts are identified by tree traversal on the constructed context tree whereas the conditional probability estimation is calculated by summing up all the statistical distributions of the sub-contexts. Suppose that we have an infrequent context $\mathbf{c}$ with k elements; the time complexity of the tree traversal is

$$\sum_{i=1}^{k}((k-i)+(i-1)M) = O(k^2M). \tag{9}$$

c) Noise Level Estimation: In order to improve the filtering robustness under different noise levels, an estimation of noise level δ is needed. It can be estimated either in terms of the min-max criterion [12] or by using image context metrics [22]. However, those solutions conduct the noise estimation in terms of the filtering results for each noise level, which is computationally expensive. A more practical estimate of δ in [23] is the minimized conditional probability in the contexts with "sufficient frequency". In a similar manner, the noise level δ is estimated here on the noisy image directly as

$$\delta = 1 - \max_{\forall y, p(\mathbf{c})>10^{-2}} P(I(y)\,|\,\mathbf{c}) \tag{10}$$

where $P(\mathbf{c})$ is the probability of context $\mathbf{c}$. The performance of the proposed noise level estimation is evaluated on 65 test images, and the results are summarized in Fig. 7. We can observe that the noise level estimation algorithm is reasonably accurate for all tested noise levels.

D. Filtering Additive Gaussian Noise

For completeness, we study the statistical filtering in case of additive Gaussian noise as well. In this case, filtering of raster map images can be considered as a continuous-input-finite-output problem. For a noisy image $\mathbf{Y}$, the problem is defined as finding a denoised palette-indexed image $\mathbf{Z}$ with M colors, which needs an estimation for the color palette. Since the size of the color palette is limited for raster map images, color quantization [20], [24], [25] can be efficiently applied if the color components are well separable [see Fig. 8, (left)]. In the following, several approaches to this problem are addressed.

a) Color Palette Estimation: The nature of the color palette estimation problem is shown in Fig. 8, where five source colors exist in the map image. Some of the colors are

[2]The denoising result of the previous iteration is only used for context pixels. The conditional probability estimation is still based on the noisy input.

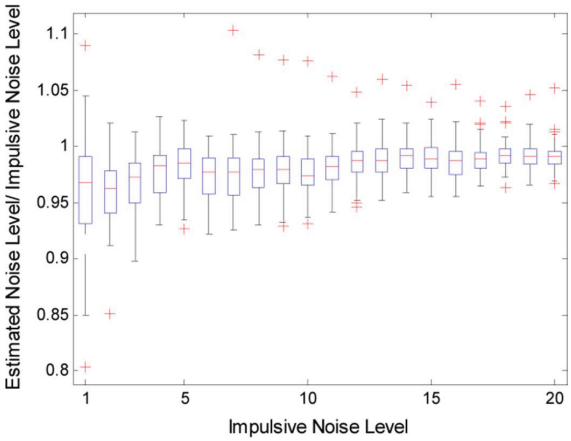

Fig. 7. Performance of noise level estimation on 65 test images.

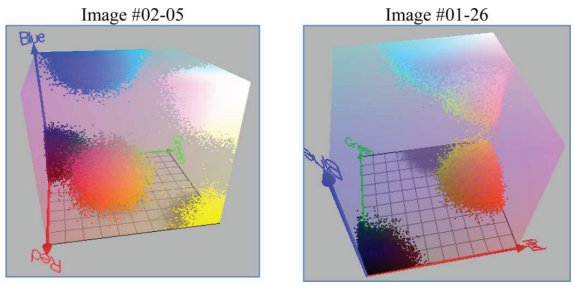

Fig. 8. Distribution in RGB color space for two map images corrupted by Gaussian noise with $\sigma = 25$.

Definitions:
Y: Noisy image
Z: Denoised image
$I(\mathbf{Z})$: Index value of image **Z**
CP = ($\mathbf{m}_1$, $\mathbf{m}_2$, …, $\mathbf{m}_M$) color palette where
$\mathbf{m}_i = [m_i(r), m_i(g), m_i(b)], \forall z \in \mathbf{Z}, I(z) \in \{1,..,M\}$,
Input: Y
$I(\mathbf{Z})$, **CP** ← Conduct color quantization according to (13)
σ ← Estimate the variance according to (16)
For T iterations DO:
Given $I(\mathbf{Z})$, update $P(I(z)|\mathbf{c})$ according to (8), for $\forall z \in \mathbf{Z}$
Update $I(\mathbf{Z})$ by fusion procedure according to (14)
Update **CP** and σ according to (15) and (16)
End-For
Output: Z, CP.

Fig. 9. Pseudo code for filtering Gaussian noise.

located near the borders of the RGB color cube. In the noisy images, the colors are spread to form a Gaussian distribution around the source colors. Colors near the border have been truncated, causing the distributions to be one-sided Gaussians. In k-means clustering, this would cause inaccurate estimation of the representative color. We therefore apply a k-medians algorithm where the median value is used (for each color channel separately) to estimate the palette color instead of the mean.

b) Size of the Color Palette M: In color quantization, the size of the color palette can be determined by using a variety of criteria [26] such as the *F-test ratio*, *Bayesian information criterion* (BIC), and *minimum description length* (MDL). For simplicity, we assume that the additive Gaussian noise model shares

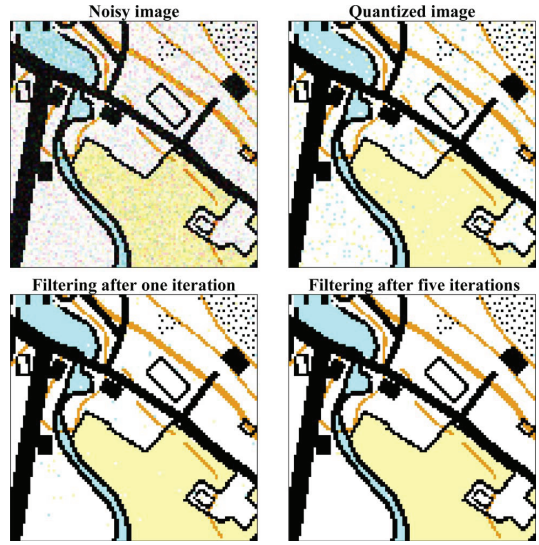

Fig. 10. Filtering example (fragment from Image #1-26) of the fusion process for additive Gaussian noise.

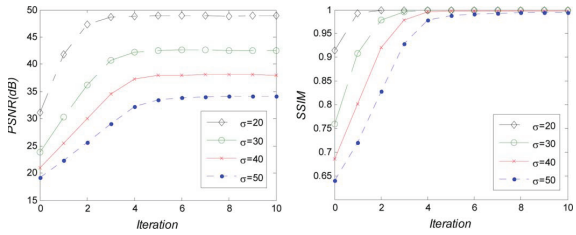

Fig. 11. PSNR and SSIM of the proposed fusion process under different noise levels (#1-26). PSNR and SSIM at iteration 0 are the results of the pre-quantization step.

the same covariance matrix $\sigma^2\mathbf{I}$ ($\mathbf{I}$ is a unit matrix) for each RGB color component and the possible size of the color palette for raster map images is limited to 2–16. This is the operative range for which we expect the algorithm to work well. Accordingly, the size of the color palette is determined as follows:

$$M = \arg \min_{m \in \{2,\ldots,16\}} f(m) \tag{11}$$

$$f(m) = \frac{1}{\sigma^2}\left(\sum_{i=1}^{m} n(i) \sum_{j=1}^{3} (\sigma_{ij} - \sigma)^2\right) \Big/ \sum_{i=1}^{m} n(i) \tag{12}$$

where $n(i)$ is the frequency of the colors on each component for the color palette, and σ_{ij} is the variance of the ith component in color channel j.

c) Iterative Fusion Process With Conditional Probability in Context: After color quantization, RGB color space is partitioned into several regions, in which each color vector $\mathbf{Y}(y)$ is represented by its centroid:

$$I(y) = \arg \min_{I(y) \in \{1,\ldots,M\}} \left(\left\|\mathbf{Y}(y) - \mathbf{m}_{I(y)}\right\|^2\right), \quad y \in \mathbf{Y}. \tag{13}$$

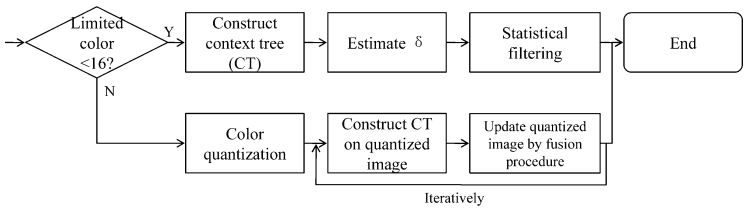

Fig. 12. Workflow of the proposed adaptive context-based method.

Since some color components can overlap (Fig. 8, right), misclassification is inevitable in color quantization (see the quantized image in Fig. 10). To overcome this problem, a novel iterative fusion algorithm is proposed by calculating the distance from a pixel to its color component in the color palette and the conditional probability relative to its context is then applied:

$$I(y) = \arg \min_{I(y)\in\{1,\ldots,M\}} \left(-\log_2 g\left(\mathbf{Y}(y)\,|\,\mathbf{m}_{I(y)}\right) - \log_2 P(I(y)\,|\,\mathbf{c})\right)$$

$$g\left(\mathbf{Y}(y)\,|\,\mathbf{m}_{I(y)}\right) = \exp\left(\frac{-\left\|Y(y)-\mathbf{m}_{I(y)}\right\|^2}{2\sigma^2}\right) \quad (14)$$

where σ is the variance of the additive Gaussian noise and y is the current pixel in the noisy image $\mathbf{Y}$. This fusion filter can be considered as a specific form of the energy function in a *Markov random field* [27], which is derived by replacing the neighborhood similarity with conditional probability in the context.

After the fusion process, the color palette and the estimated noise variance σ are re-estimated as

$$\mathbf{m}_i = \text{median}(\{\mathbf{Y}(y)\,|\,y\in\mathbf{Y}, I(y)=i\}) \quad (15)$$

$$\sigma = \frac{1}{3|\mathbf{Y}|}\sum_{y\in\mathbf{Y}}\left\|\mathbf{Y}(y)-\mathbf{m}_{I(y)}\right\|^2. \quad (16)$$

The fusion and the estimation processes are performed iteratively. The pseudo code of the algorithm is described in Fig. 9, and an example of the fusion result is shown in Fig. 10.

Performance comparisons (PSNR and SSIM) are made for different noise levels in Fig. 11. It is observed that the proposed fusion process is effective and very robust for different noise levels. To sum up, the workflow of the proposed adaptive context-tree-based statistical filtering is summarized in Fig. 12.

E. Process Noisy Image With Mixture Noise

To denoise an image with mixed Gaussian-impulsive noise, a straightforward approach is to apply two filters successively: one for impulsive noise and another one for Gaussian noise, respectively. For example, a *fuzzy peer group* [35] combines a statistical method for impulsive noise detection with replacement by an averaging operation to smooth out Gaussian noise.

In a similar manner, the proposed statistical filtering can be extended to the problem of denoising the mixed noise, as outlined in Fig. 13. The proposed extension combines both the case of the statistical filtering for impulsive noise and the case of the fusion process for additive Gaussian noise. Namely, if the DUDE decision rule is met, the current pixel is identified as impulsive noise and then replaced by the color with the maximum conditional probability. Otherwise, the fusion process is applied.

Definitions:
Y: Noisy image
Z: Denoised image
$I(\mathbf{Z})$: Index value of image **Z**
CP=($\mathbf{m}_1$, $\mathbf{m}_2$, …, $\mathbf{m}_M$) color palette where
$\mathbf{m}_i = [m_i(r), m_i(g), m_i(b)]$, $\forall z \in \mathbf{Z}, I(z) \in \{1,..,M\}$,
Input: Y
$I(\mathbf{Z})$, **CP** ← Conduct color quantization according to (13)
σ, δ ← Estimate the noise level according to (10) and (16)
For T iterations DO:
Given $I(\mathbf{Z})$, update $P(I(z)|\mathbf{c})$ according to (8), $\forall z \in \mathbf{Z}$
IF DUDE decision rule (6) is met
Update $I(z)$ by the color with maximum conditional probability
Else
Update $I(z)$ by fusion procedure according to (14), $\forall z \in \mathbf{Z}$
End
Update **CP** , δ, and σ according to (10), (15), and (16)
End-For
Output: Z, CP.

Fig. 13. Pseudo code for filtering mixture noise.

F. Computational Analysis

In general, the context-tree construction leads to a time complexity of $O(kN)$, where k is the size of the context and N is the number of the pixels in the image. Any context can have a maximum of NM/T child nodes, where M is the size of the color palette and T is the frequency threshold for *context merging*. In *context merging*, since the time complexity of every merging process is $O(k^2M)$ in (9), the total time complexity of the *context-merging* process is $O(k^2M \cdot NM/T)$. Additionally, the noise estimation procedure has a time complexity of $O(M)$ in which all contexts with a frequency higher than $p(\mathbf{c}) = 0.01$ are extracted by the tree traversal process. The DUDE decision rule is applied to determine whether or not a pixel is filtered. As the conditional probability of all contexts is pre-calculated, the filtering procedure has a time complexity of $O(N)$. As a result, the total time complexity for denoising impulsive noise is $O(k^2M^2 \cdot N/T)$.

For filtering additive Gaussian noise, the clustering-based color quantization step has a time complexity of $O(MN)$. Context-tree construction and *context merging* have the same complexity as in impulsive noise filtering. In the fusion procedure, the cost function of (14) needs to be calculated on each pixel for all the colors in the color palette, and thus, it leads to a time complexity of $O(MN)$. The total time complexity of denoising additive Gaussian noise is therefore $O(k^2M^2 \cdot N/T)$.

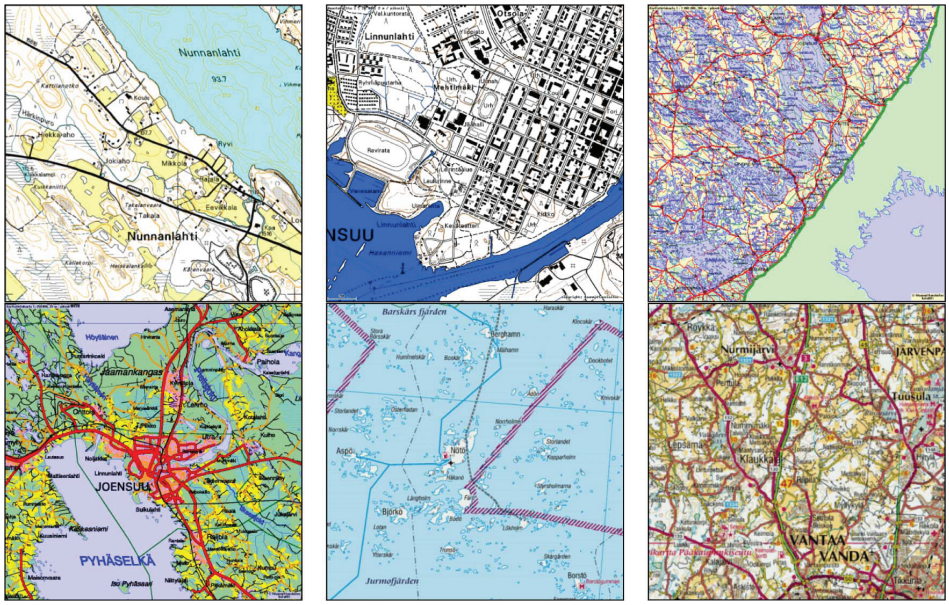

Fig. 14. Sample images from the test set: #1-26, #2-05, #3-03, #4-04, #5-01, and #5-02.

TABLE I
TIME COMPLEXITY OF THE PROPOSED ALGORITHM

Impulsive noise		Additive Gaussian noise	
Step	Complexity	Step	Complexity
Context tree modeling	$O(kN)$	Color quantization	$O(MN)$
Context-merging	$O(k^2M^2 \cdot N/T)$	Context tree modeling	$O(kN)$
Noise level estimation	$O(M)$	Context-merging	$O(k^2M^2 \cdot N/T)$
Statistical filtering	$O(N)$	Fusion procedure	$O(MN)$
Total	$O(k^2M^2 \cdot N/T)$	Total	$O(k^2M^2 \cdot N/T)$

In case of the mixed noise, either the DUDE decision rule-based statistical filtering or the fusion process is applied, and no additional cost is incurred. The time complexities are summarized in Table I.

III. EXPERIMENTS

We evaluate the proposed adaptive context-based statistical filtering algorithm (ACS) on a set of map images chosen from the *Finnish National Land Survey* (http://cs.joensuu.fi/sipu/images/mapset.zip); see Table II and Fig. 14. The images have different types (topographic, roadmap) and scales. To test the performance, we artificially distort the images with impulsive noise, with additive Gaussian noise, and with mixed Gaussian-impulsive noise.

A. Parameter Adjustment

In order to choose the appropriate adjustment of the denoising parameters, we analyzed the performance with different selections of parameters. These include the threshold T for the *context merging*, the number of iterations of filtering impulsive noise, and the number of iterations of the fusion procedure.

TABLE II
DESCRIPTIONS OF THE TEST IMAGES

	Scale	Number of images	Image size	Number of colors
Test set #1	1 : 8 000	50	1024 × 1024	5
Test set #2	1 : 20 000	5	1024 × 1024	5
Test set #3	1 : 800 000	4	1024 × 1024	9
Test set #4	1 : 100 000	4	1024 × 1024	10
Test set #5	1 : 250 000	2	800 × 800	16

TABLE III
COMPARISONS OF THE ERROR RATE (TEST SET #1) WHEN CONDITIONAL PROBABILITY ESTIMATION OF A LOCAL CONTEXT IS PERFORMED SEPARATELY OR JOINTLY ON THE WHOLE SET

	Error Rate (%)	
	Single image	Whole test set
$\delta = 0.01$	0.21	0.18
$\delta = 0.05$	0.65	0.58
$\delta = 0.10$	1.31	1.16
$\delta = 0.20$	3.35	3.01

First, we investigate how to select the threshold T. The best selection of T depends on both the number of colors and the noise level of the input image. When the number of colors increases, a larger T should be selected but the exact value of the threshold was found not to be critical for the performance of the algorithm. In our experiments, we set $T = 128$ for test sets #1 and #2 and $T = 256$ for test sets #3–5.

From our experiment, an appropriate selection of the number of iterations in the filtering of impulsive noise can be adopted

TABLE IV
ERROR RATE, FALSE ACCEPTANCE RATE, AND FALSE REJECTION RATE FOR FILTERING IMPULSIVE NOISE

		Error rate (%)					False acceptance rate (%)					False rejection rate (%)				
		AVM	PGF	CT	DUDE	ACS	AVM	PGF	CT	DUDE	ACS	AVM	PGF	CT	DUDE	ACS
$\delta = 0.05$	Set#1	8.65	1.40	1.04	0.94	**0.65**	45.7	26.9	15.6	13.0	7.33	6.70	0.06	0.27	0.30	0.30
	Set#2	4.19	1.26	0.99	0.99	**0.78**	9.89	24.7	14.7	16.5	10.2	3.89	0.02	0.27	0.17	0.28
	Set#3	11.8	3.26	2.41	2.09	**1.59**	47.1	49.3	23.9	36.9	27.5	9.89	0.83	1.28	0.26	0.22
	Set#4	6.55	1.21	1.42	1.34	**0.82**	12.1	20.4	10.5	21.4	11.8	6.25	0.20	0.94	0.28	0.24
	Set#5	9.27	6.02	4.36	2.66	**2.02**	43.6	39.9	25.9	48.3	34.6	7.46	4.23	3.23	0.25	0.30
$\delta = 0.10$	Set#1	10.9	3.56	7.83	1.95	**1.31**	46.3	34.9	77.7	14.5	8.05	6.95	0.07	0.07	0.55	0.56
	Set#2	5.06	3.07	5.01	2.14	**1.57**	11.1	30.5	49.0	18.2	10.7	4.39	0.03	0.13	0.35	0.55
	Set#3	14.1	5.96	4.28	4.14	**3.14**	48.1	52.6	33.1	36.5	26.2	10.3	0.78	1.07	0.55	0.58
	Set#4	7.14	2.45	2.63	2.66	**1.51**	12.9	22.5	17.7	21.4	10.8	6.50	0.22	0.96	0.57	0.48
	Set#5	11.4	7.94	5.89	5.01	**3.85**	44.7	42.0	29.8	44.9	31.5	7.67	4.16	3.23	0.58	0.78
$\delta = 0.20$	Set#1	15.7	10.9	19.9	5.04	**3.35**	48.3	54.3	99.5	21.0	11.7	7.59	0.10	2e-3	1.05	1.28
	Set#2	7.47	9.20	19.5	5.31	**3.49**	14.9	15.8	97.6	23.2	13.3	5.62	0.03	9e-3	0.82	1.05
	Set#3	19.3	12.5	15.6	9.63	**6.84**	51.1	59.8	76.7	44.1	28.4	11.4	0.74	0.31	1.01	1.45
	Set#4	8.65	6.07	12.1	6.67	**3.39**	15.0	29.1	58.8	29.3	13.0	7.05	0.30	0.44	1.02	0.98
	Set#5	16.1	12.8	10.4	10.4	**7.79**	47.5	47.4	41.2	48.0	31.8	8.25	4.09	2.68	1.02	1.80

TABLE V
PSNR AND SSIM FOR FILTERING ADDITIVE GAUSSIAN NOISE

		PSNR					SSIM				
		GSM	NLM	BM3D	ARF	ACS	GSM	NLM	BM3D	ARF	ACS
$\sigma = 15$	Set#1	26.8	26.9	29.1	26.2	**59.4**	0.956	0.963	0.987	0.949	**0.999**
	Set#2	27.1	24.9	29.9	25.2	**70.1**	0.944	0.938	0.952	0.939	**0.999**
	Set#3	26.3	23.8	25.6	24.5	**40.8**	0.957	0.946	0.948	0.946	**0.998**
	Set#4	27.1	25.9	27.1	25.2	**51.5**	0.941	0.931	0.945	0.933	**0.999**
	Set#5	26.3	26.0	27.1	26.2	**28.5**	0.881	0.886	0.911	0.872	**0.952**
$\sigma = 25$	Set#1	24.4	24.9	26.2	24.2	**47.1**	0.943	0.944	0.978	0.942	**0.999**
	Set#2	24.0	23.5	26.5	23.3	**59.2**	0.919	0.918	0.941	0.924	**0.999**
	Set#3	23.7	22.4	23.7	23.0	**27.0**	0.919	0.904	0.928	0.929	**0.975**
	Set#4	24.6	24.1	25.5	23.8	**33.4**	0.874	0.881	0.923	0.902	**0.988**
	Set#5	25.1	24.3	26.1	25.1	**27.7**	0.846	0.827	0.899	0.860	**0.926**
$\sigma = 35$	Set#1	22.2	22.7	23.9	22.3	**42.7**	0.898	0.912	0.964	0.919	**0.998**
	Set#2	21.6	21.6	23.9	21.5	**48.4**	0.869	0.893	0.926	0.893	**0.999**
	Set#3	21.1	20.7	21.6	21.3	**25.7**	0.823	0.857	0.859	0.860	**0.968**
	Set#4	21.5	22.0	22.9	21.8	**31.3**	0.717	0.821	0.783	0.771	**0.979**
	Set#5	22.1	22.6	23.9	22.0	**25.4**	0.700	0.766	0.796	0.766	**0.912**

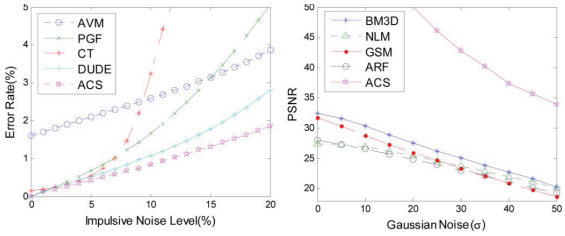

Fig. 15. Performance comparison with different noise levels (Image #01-26 is used).

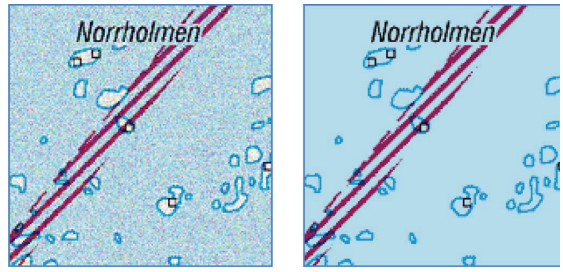

Fig. 16. Worst case example of the denoising result for image #05-01: noisy image (left) and the denoised image (right). Noise has been eliminated; text and other details are preserved. However, due to the sub-optimal color palette estimation, the light blue and white colors have been merged, which leads to the loss of essential information.

as follows: two iterations are conducted for images of impulsive noise with noise levels smaller than 10% and three iterations are conducted for images with noise levels greater than 10%. For denoising additive Gaussian noise, five iterations provide a good compromise between denoising performance and computational complexity (see Fig. 11). For denoising mixed Gaussian-impulsive noise, five iterations of the fusion process are conducted for Gaussian noise denoising, and two iterations

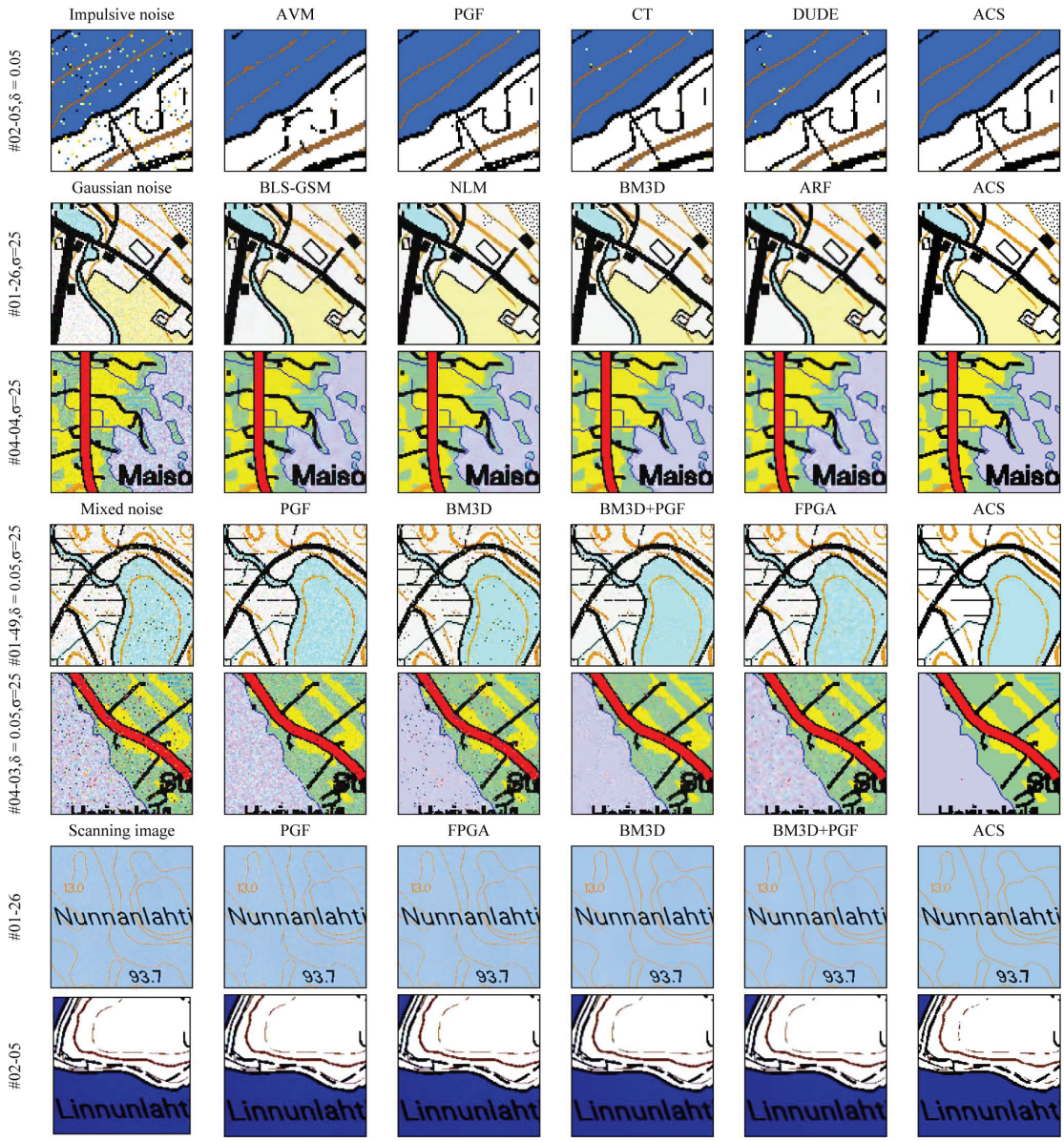

Fig. 17. Visual performance comparison.

of DUDE decision rule-based statistical filtering are performed for detecting and denoising impulsive noise.

Test set #1 consists of 50 topographic images of size 1024×1024, which all use the same types of notations to represent topographic information and include the same patterns. We therefore test whether the denoising performance will improve when the conditional probability estimation is estimated on the entire test set using the same conditional probability of each context for all the images. From our experiments, the algorithm performance improves by 10% compared with the case of conditional probability estimation on a single image (see Table III). To sum up, if the type of image is known, the context modeling can be trained and better performance is achieved. Nevertheless, this approach is not used in the following experiments.

B. Objective Evaluation

a) Impulsive Noise: First, we compare the proposed algorithm with four alternative filters: *adaptive vector median* (AVM) [2], *fast peer group filter* (PGF) [3], *context-tree filter*

TABLE VI
PSNR AND SSIM FOR FILTERING MIXED NOISE

		PSNR					SSIM				
		PGF	BM3D	BM3D +PGF	FPGA	ACS	PGF	BM3D	BM3D +PGF	FPGA	ACS
$\delta = 0.03, \sigma = 15$	Set#1	21.4	20.2	22.5	22.0	**25.6**	0.769	0.786	0.940	0.901	**0.977**
	Set#2	21.3	19.8	2201	22.9	**23.7**	0.772	0.759	0.906	0.891	**0.978**
	Set#3	21.4	20.9	22.0	23.0	**25.4**	0.779	0.798	0.902	0.906	**0.970**
	Set#4	22.7	21.9	23.8	25.4	**25.8**	0.710	0.800	0.898	0.860	**0.967**
	Set#5	21.4	22.3	23.7	24.8	**25.9**	0.652	0.739	0.868	0.821	**0.945**
$\delta = 0.05, \sigma = 25$	Set#1	18.0	17.7	19.8	19.5	**23.6**	0.635	0.685	0.867	0.788	**0.966**
	Set#2	18.1	17.2	19.4	19.5	**21.7**	0.669	0.670	0.830	0.750	**0.967**
	Set#3	17.9	18.5	19.7	20.7	**23.0**	0.654	0.707	0.817	0.785	**0.952**
	Set#4	19.0	19.1	21.3	21.4	**24.4**	0.598	0.666	0.780	0.716	**0.961**
	Set#5	18.3	19.9	21.9	21.5	**22.7**	0.523	0.640	0.774	0.673	**0.914**
$\delta = 0.10, \sigma = 35$	Set#1	14.8	14.3	16.3	16.4	**20.3**	0.499	0.514	0.649	0.649	**0.925**
	Set#2	15.0	13.7	16.1	16.1	**18.7**	0.572	0.532	0.654	0.637	**0.929**
	Set#3	15.3	15.3	16.6	17.6	**19.6**	0.544	0.558	0.625	0.672	**0.895**
	Set#4	16.4	15.2	17.6	18.1	**21.5**	0.505	0.479	0.564	0.594	**0.920**
	Set#5	16.2	16.3	18.4	18.5	**19.8**	0.425	0.474	0.547	0.551	**0.895**

(CT) [13], [19], and *discrete universal denoising* (DUDE) [11], [14] by using images corrupted by impulsive noise. The performance is measured by the *error rate* (%), *false acceptance rate* (%), and *false rejection rate* (%) (see Table IV). Experiments show that the algorithms based on statistical filtering (CT, DUDE, ACS) achieve better noise reduction performance (lower *false acceptance rate*) and better preservation of image details than the best of the conventional filters (AVM, PGF). This is because raster map images consist of pixel-level detailed structures, sharp edges, and repetitive patterns whereas those traditional algorithms are based on *a priori* assumption that the images have smooth color transitions.

The context-tree filter with a predefined filtering threshold lacks robustness in denoising images with different noise levels. However, both the noise estimation and DUDE decision rule have been exploited here to achieve robust performance in the proposed statistical filter. Since the conditional probability estimation is further improved by the *context-merging* strategy, the proposed algorithm achieves better performance than DUDE, which has been the best solution for universal discrete denoising problems so far.

b) Additive Gaussian Noise: Four state-of-the-art filters for denoising Gaussian noise are also evaluated: *wavelet denoising using Gaussian scale mixtures* (BLS-GSM) [5], *non-local means* (NLM) [6], *block matching and 3-D filtering* (BM3D) [8], and *active random fields* (ARF) [10]. Their performance comparisons are summarized in Table V using the *peak signal-to-noise ratio* (PSNR) and *structural similarity index* (SSIM) and are visually demonstrated in Fig. 17. It can be observed that the proposed method achieves both visually and numerically better results than the comparative filters. The difference is remarkable for test sets #1–4. This is because the comparative algorithms are designed with *a priori* assumption of smooth color transitions in the images. Moreover, raster map images have a very limited output and this is not considered in those algorithms.

We also compared the performances between the images with different noise levels as shown in Fig. 15. It can be observed that the proposed algorithm is also robust to the noise level. However, the performance is less impressive for image #05-01 than for the other images. This is because most color components become more overlapped and those color components lack separation in color space when its number increases. In the color palette estimation of Section II-D, those overlapped color components are merged into the same output color. This problem is demonstrated in Fig. 16 for #05-01, in which the number of output colors is reduced to nine, and it degrades the denoising performance.

c) Mixed Noise: We compare the performance using images with mixed Gaussian-impulsive noise against PGF,[3] BM3D, and *Fuzzy Peer Group Averaging* (FPGA). An additional experiment was carried out by using BM3D as a first step for denoising Gaussian noise followed by PGF as a second step for denoising impulsive noise. The results are reported in Table VI and Fig. 17. Both the numerical results and visual examinations show that the proposed algorithm is superior to all the comparative algorithms in denoising images with mixed Gaussian-impulsive noise.

d) Real-World Examples: We performed additional tests by printing and re-scanning two selected images from set #1 and #2; see Fig. 17 for scanning image. The resulting images have slightly different colors than their original ones and Gaussian type of noise and blurred contours also appear. Among the other filters, PGF cannot remove the Gaussian-type of noise in these real examples, and BM3D causes blurring effect around the contours. The proposed method (ACS) preserves the larger structures in Set #2 very well but some of the noisy thin structures are broken. For the image from set #1, thin structures are mostly well restored. No false colors or blurring effects appear in the output image either, but discontinuation of thin contours appears at places.

[3]DUDE and the CT algorithm can only be used for discrete denoising problems and cannot be applied when the input is a continuous-tone image.

IV. Conclusion

We have proposed a statistical filtering algorithm dealing with map images distorted by impulsive noise, additive Gaussian noise, and mixed Gaussian-impulsive noise. The proposed filter incorporates an information fusion process which exploits both the color distribution in RGB space and the conditional probabilities of a given pixel in a local context. It operates with no prior knowledge of the properties of the noise and aims at maximal preservation of repetitive structures of the image. This is an essential property for raster map images and is expected to generalize to other types of palette-indexed imagery as well. It can also be viewed as a pioneer study to attack distortion caused by unknown noise types. Experiments with different noise types and spatial image resolutions show that the proposed filter provides robust and reliable filtering performance and good structure preservation ability.

We also investigate the *context contamination* problem in conditional probability estimation in statistical filtering and a *context-merging* strategy is proposed to improve the estimation accuracy for those infrequent contexts.

The proposed algorithm can also be used for other types of color palette images such as engineering drawings, schemes, comic books, and similar art imagery. Raster map images have the typical properties (sharp edges and repeated patterns) that also exist in other kind of images, and were therefore selected here as a typical but challenging case study for evaluating the efficiency of the proposed algorithm.

Future work can be done to extend the proposed filtering method to continuous-tone images in addition to color indexed images. More theoretical analysis is needed on how to select the optimal threshold in context merging.

Acknowledgment

The authors would like to thank the reviewers and the editor for their valuable comments and suggestions, which have been very useful in improving the technical content and the presentation of the paper. The authors also would like to thank Prof. S. Morillas for providing the code of his algorithm.

References

[1] H. Zhou and K. Mao, "An impulsive noise color image filter using learning-based color morphological operation," *Digit. Signal Process.*, vol. 18, pp. 406–421, 2008.

[2] R. Lukac, "Adaptive vector median filtering," *Pattern Recognit. Lett.*, vol. 24, pp. 1889–1899, 2003.

[3] B. Smolka and A. Chydzinski, "Fast detection and impulsive noise removal in color images," *Real-Time Imaging*, vol. 11, no. 5–6, pp. 389–402, 2005.

[4] M. Chen, M. Xu, and P. Fränti, "Multi-layer filtering approach for map images," in *Proc. IEEE Int. Conf. Image Processing*, 2009, pp. 3953–3956.

[5] J. Portilla, V. Strela, M. Wainwright, and E. P. Simoncelli, "Image denoising using a scale mixture of Gaussians in the wavelet domain," *IEEE Trans. Image Process.*, vol. 12, no. 11, pp. 1338–1351, Nov. 2003.

[6] A. Buades, B. Coll, and J. M. Morel, "A non local algorithm for image denoising," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2005, vol. 2, pp. 60–65.

[7] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[8] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[9] S. Roth and M. J. Black, "Fields of experts: A framework for learning image priors," in *Proc. IEEE Int. Conf. Computer Vision*, 2005, pp. 860–867.

[10] A. Barbu, "Training an active random field for real-time image denoising," *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2451–2462, Nov. 2009.

[11] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdú, and M. Weinberger, "Universal discrete denoising: Known channel," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 5–28, Jan. 2005.

[12] G. Gemelos, S. Sigurjonsson, and T. Weissman, "Algorithms for discrete denoising under channel uncertainty," *IEEE Trans. Signal Process.*, vol. 54, no. 6, pp. 2263–2276, Jun. 2006.

[13] P. Kopylov and P. Fränti, "Filtering of color map images by context tree modeling," in *Proc. IEEE Int. Conf. Image Processing (ICIP'04)*, 2004, vol. 1, pp. 267–270.

[14] E. Ordentlich, M. J. Weinberger, and T. Weissman, "Multi-directional context sets with applications to universal denoising and compression," in *Proc. 2005 IEEE Int. Symp. Inform. Theory (ISIT'05)*, 2005, pp. 1270–1274.

[15] G. Gimel'farb, "Adaptive context for a discrete universal denoiser," in *Proc. Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR Int. Workshops, SSPR 2004 and SPR*, 2004, pp. 477–485.

[16] A. Akimov, A. Kolesnikov, and P. Fränti, "Lossless compression of color map images by context tree modeling," *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 114–120, Jan. 2007.

[17] G. Zhai, X. Wu, X. Yang, and W. Zhang, "MDL context modeling of images with application to denoising," in *Proc. IEEE Int. Conf. Image Processing*, 2009, pp. 3845–3848.

[18] J. Yu and S. Verdú, "Schemes for bidirectional modeling of discrete stationary sources," *IEEE Trans. Inf. Theory*, vol. 52, no. 11, pp. 4789–4807, Nov. 2006.

[19] A. Podlasov, P. Kopylov, and P. Fränti, "Statistical filtering of raster map images using a context tree model," in *Proc. Int. Conf. Signal-Image Technology and Internet-Based Systems (IEEE-SITIS)*, 2007, pp. 467–474.

[20] Q. Zhao, V. Hautamäki, I. Kärkkäinen, and P. Fränti, "Random swap EM algorithm for finite mixtures models in image segmentation," in *Proc. IEEE Int. Conf. Image Processing*, 2009, pp. 2397–2400.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[22] X. Zhu and P. Milanfar, "A no-reference image content metric and its application to denoising," in *Proc. IEEE Int. Conf. Image Processing*, 2010, pp. 1145–1148.

[23] E. Ordentlich, G. Seroussi, S. Verdú, M. Weinberger, and T. Weissman, "A discrete universal denoiser and its application to binary images," in *Proc. IEEE Int. Conf. Image Processing*, 2003, pp. 117–120.

[24] P. Kopylov and P. Fränti, "Color quantization of map images," in *Proc. IASTED Conf. Visualization, Imaging, and Image Processing (VIIP'04)*, 2004, pp. 837–842.

[25] D. Mavridis and N. Papamarkos, "Color quantization using principal components for initialization of Kohonen SOFM," in *Proc. IEEE Int. Conf. Image Processing*, 2009, pp. 1633–1636.

[26] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Cluster validity methods," in *ACM SIGMOD Record*, 2002.

[27] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov random fields with smoothness-based priors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 1068–1080, Jun. 2008.

[28] J. Rissanen, "A universal data compression system," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 5, pp. 656–664, Sep. 1983.

[29] M. Weinberger, J. Rissanen, and R. Arps, "Application of universal context modeling to lossless compression of gray-scale images," *IEEE Trans. Image Process.*, vol. 5, no. 4, pp. 575–586, Apr. 1996.

[30] M. Chen, M. Xu, and P. Fränti, "Statistical filtering of raster map images," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME'10)*, 2010, pp. 394–399.

[31] S. Leyk and R. Boesch, "Colors of the past: Color image segmentation in historical topographic maps based on homogeneity," *Geoinfomatica*, vol. 14, pp. 1–21, 2010.

[32] A. Khotanzad and E. Zink, "Contour line and geographic feature extraction from USGS color topographical paper maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 1, pp. 18–31, Jan. 2003.
[33] Y. Chen, Y. Liu, Z. Fu, and R. Wang, "Automatic extracting residential areas from color scanned topographical maps," *Image Signal Process.*, 2009.
[34] T. Henderson and T. Linton, "Raster map image analysis," in *Proc. Int. Conf. Document Analysis and Recognition*, 2009, pp. 376–380.
[35] S. Morillas, V. Gregori, and A. Hervas, "Fuzzy peer groups for reduction mixed Gaussian-impulsive noise from color images," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1452–1466, Jul. 2009.

Minjie Chen (S'10) received the B.Sc. and M.Sc. degrees in biomedical engineering from Shanghai Jiaotong University, Shanghai, China, in 2003 and 2007, respectively. Since 2008, he has been pursuing the Ph.D. degree in computer science at the University of Eastern Finland, Joensuu, Finland.

His research interests include image denoising and compression, spatial-temporal data compression, and medial image analysis.

Mantao Xu received the B.Sc. degree in mathematics from Nankai University, Tianjin, China, in 1991, the M.Sc. degree in applied mathematics from Harbin Institute of Technology, Harbin, China, in 1997, and the Ph.D. degree in computer science from the University of Joensuu, Joensuu, Finland, in 2005.

He served as a Research Lab Manager with Kodak Health Group and Carestream Health Inc., Global R&D Center, Shanghai, China, from 2005 to 2010. He is now a Research Professor at the School of Electric Engineering, Shanghai Dianji University, Shanghai, China. His research interests include medical image analysis, multimedia technology, and pattern recognition.

Pasi Fränti (SM'08) received the M.Sc. and Ph.D. degrees in science from the University of Turku, Turku, Finland, in 1991 and 1994, respectively.

Since 2000, he has been a Professor of computer science at the University of Eastern Finland, Joensuu, Finland. He has published 56 journal and 128 peer review conference papers, including 10 IEEE transaction papers. His research interests include clustering algorithms, vector quantization, lossless image compression, voice biometrics, and location-based systems. He has supervised 14 Ph.D. students and is currently the head of the East Finland doctoral program in Computer Science and Engineering (ECSE).

Dr. Fränti serves as an Associate Editor for *Pattern Recognition Letters*.

Paper P3

M. Chen, M. Xu, P. Fränti, "Adaptive Filtering of Raster Map Images Using Optimal Context Selection", *IEEE Int. Conf. on Image Processing* (ICIP'11), 77-80, Brussels, Belgium, 2011.

Adaptive Filtering of Raster Map Images Using Optimal Context Selection

Minjie Chen[1], *Mantao Xu*[2], *Pasi Fränti*[1]
[1]University of Eastern Finland, Finland [2]Shanghai Dianji University, China

ABSTRACT

Filtering of raster map images or more general class of palette-indexed images is considered as a discrete denoising problem with finite color output. Statistical features of local context are used to avoid damages of some specific but frequently occurring contexts caused by conventional filters. Several context-based approaches have been developed using either fixed context templates or context tree modeling. However, these algorithms fail to reveal the local geometrical structures when the underlying contexts are also contaminated. To address this problem, we propose a novel context-based voting method to identify the possible noisy pixels, which are excluded in the context selection and optimization. Experimental results show that the proposed context based filtering outperforms all other existing filters both for impulsive and Gaussian additive noise.

Index Terms *—Nonlinear filters, context modeling*

1. INTRODUCTION

Raster map images are commonly encoded in a regular grid of pixel colors arrayed in rows and columns, in which each color represents a different class of semantic map object. It consists of pixel level detailed structures and sharp edges but lacks smooth color transitions that are typical for photographic images. It does not require any additional image processing procedure and is therefore suitable for the delivery to the multimedia applications. However, such images can be degraded in image acquisition (digitization) process and this color degradation may lead to severe false recognition of important semantic map objects. Hence, image filtering is needed before publishing the image. There are several technical challenges for designing suitable filter for this kind of images.

A great variety of noise removal techniques have been extensively investigated for color image processing. The multi-layer approach in [1] converts the problem into binary domain whereas other approaches try to work in the color domain. However, these algorithms are usually developed in terms of specifically noise model, such as *impulsive noise* [2-3] and *additive Gaussian noise* [4-7]. In the case of *impulsive noise*, noisy pixels could be detected if suitable statistical rules based on the local variation were designed. However, these rules are always based on presumable knowledge, which inevitably incur a significant misclassification error. For *additive Gaussian noise,* most of filters are designed by selecting an optimal linear combination of a few basis elements, either pixel-wise or block-wise. These methods assume that the true signal can be approximated by a linear combination of few basis elements and designed only for continuous tone images. They are not applicable for map images in general, because raster map images need a finite color output. Moreover, the repeatable geometrical structures inherited in map images haven't been considered in these methods. In other words, the literature lacks of methods that is capable to filter this kind of images properly.

A pioneer work in the art of statistical filtering is the so-called *discrete universal denoising* (DUDE) [8] for filtering binary data with a known noisy channel. It consists of two steps: counting statistics for all context patterns encountered, and denoising by utilizing the conditional probability of local context. This method is applicable in denoising of binary image if the error probability δ can be estimated presumably. Namely, for a given pixel x, if the conditional probability in the surrounding context $P(x|\mathbf{c})$ is lower than $2\delta(1-\delta)$, it would be treated as noise pixel and replaced by the complementary value. However, the memory allocation for running this algorithm grows exponentially with the size of the fixed context, which makes the implementation of this algorithm intractable. To circumvent this memory allocation problem, the context tree modeling [9, 10] has been applied by pruning redundant nodes of contexts.

In practice, albeit the above algorithms are very efficient for the input images with a few number of noise pixels, the contexts themselves will embrace a significant number of noise pixels when noise level is increased for the input image. Even refining the contexts adaptively using the pruning algorithm [10-12] was not able to remedy this open problem completely. Obviously, including noise pixels or outliers in the surrounding contexts will make it extremely difficult to estimate a good conditional probability distribution for context modeling. Thus, the underlying noise pixels in the selected contexts must be first detected and then need to be excluded in the modeling.

The motivation of this work is to attack the problem by removing detected outliers from the context by classifying the context according to its filtering efficiency. Each problematic context is processed by removing one pixel in turn. In case of significant change in the self-entropy of the context, we conclude that the removed pixel is noisy one.

The work was supported by MOPSI project EU (EAKR), Natural Science Foundation of China (Grant No 61072146), Shanghai Committee of Science and Technology (Grant No 10PJ1404400) and Nokia scholarship.

The modified context is then used in the filtering if it remains statistically significant in order to separate valid values from noisy ones. In this way, the context-based filtering is significantly improved.

2. PROPOSED METHOD

2.1 Context Tree Modeling

The classical context tree modeling technique has been widely used by data compression community with linear time complexity. A context tree is built by estimating the count statistics via a sequential traversal of the image pixel-by-pixel. Each node of the context tree represents a single context by storing the count statistics over each color for the current pixel relative to the node of context. Since not all possible contexts are present in the image, memory is only allocated for the actual number of pixel combinations. In our implementation, the spanning of tree is terminated once the frequency of the context on a given node is less than a predefined value.

2.2 Context Efficiency Validation

In image compression, all pixels must be encoded regardless of the reliability or probability of the context surrounding that pixel. One keeps track of how well it performs. In case of bad probability estimate, the coding of that pixel just takes more space. Optimal pruning of a context tree is always done on each of possible node in order to achieve a highest overall compression performance.

In image filtering, however, the main challenge is that the distribution of noise data is seldom known, and thus no evaluation can be done on how well filtering works. The critical issue is how to find some "meaningful" local contexts and filter only the pixels with lower conditional probability in local contexts. In DUDE [8], this decision rule is, in essence, a *MAP estimator*, which can be formulated as follows: for a local context $\mathbf{c}$, the output of filtering x, equals to u_0, is the index value with highest conditional probability when equation (2) is met.

$$u_0 = \arg\max_{x \in A} P(x|\mathbf{c}) \tag{1}$$

$$\frac{(M-1)^2(1-\delta)}{\delta((1-\delta)M-1)} P(x = x_0|\mathbf{c}) - \frac{(M-1)}{((1-\delta)M-1)} P(x = u_0|\mathbf{c}) < 1 \tag{2}$$

Albeit DUDE has a so-called "*asymptotic optimality*" property, it demands an infinite sequence of data source for estimating all the conditional distributions of local contexts. This is of course not realistic in practice. In particular, when the context of a given pixel is contaminated by erroneous colors, the count corresponding to the symbol will be credited to the "wrong" context with rare appearance, which also causes inaccurate estimation of the context distribution.

This scenario motivates us to investigate a criterion for context classification. Those frequently contexts associated with a *dominant color* (conditional probability>90%), are termed as *good context*, on which filtering can be applied directly. Those rare appearance contexts, which include noise elements, we define as *bad* context because estimation of conditional probability under such contexts is inaccurate. In order to improve this estimation, a voting scheme is applied to find those noise pixels based in the *bad* contexts. An accurate conditional probability can be estimated on a new context excluding those noisy pixels. Here, all the contexts are categorized into three groups: *good*, *uncertain* or *bad* according to a so-called *context efficiency function*:

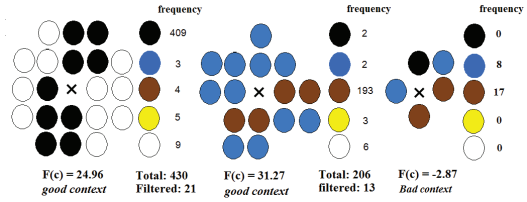

Fig. 1 *Examples of context classifications*

$$\mathrm{F}(\mathbf{c}) = \log_2\left(\frac{P(\mathbf{c})}{P_E(\mathbf{c})}\right) + k\sum_x P(x|\mathbf{c}) \log\left(\frac{P(x|\mathbf{c})}{P(x)}\right) \tag{3}$$

where $P_E(\mathbf{c}) = \Pi_i P(y_i)$, $P(\mathbf{c})$ is the probability of a given context $\mathbf{c}$ and $P_E(\mathbf{c})$ is the estimated probability of $\mathbf{c}$, y_i is the color of i^{th} element in a given context $\mathbf{c}$, and $P(y_i)$ the probability of y_i. $P_E(\mathbf{c})$ is computed by assuming all elements in the context are mutually independent.

In a sense of image compression, the first term in the right hand side of (3) can be interpreted as the difference of the context code length achieved according to the actual context probability and the expected context probability. Higher value for this term indicates that context $\mathbf{c}$ is a repetitive structure, and thus, it can be used as a direct filter when *dominant color* exists. The second term is the so-called *Kullback-Leiber distance* between conditional probability and color probability of the entire image. Larger distance implies that more bits can be saved when context $\mathbf{c}$ is used in coding.

To this end, all contexts are categorized into three groups accordingly: *good:* $F(\mathbf{c}) > T_{max}$ with a *dominant color*, *bad*: $F(\mathbf{c}) < T_{min}$ and *uncertain*: otherwise. Once all contexts have been categorized into these three classes, the context tree is processed by identifying the *good* and *bad* nodes. The offspring nodes of any *good* or *bad* node will be removed in the tree pruning using top-to-bottom tracing. T_{max}, T_{min} and k are adopted as values of 3, -0.5 and 0.5 in this work. Three context examples are shown in Fig. 1. Two *good* contexts are with *dominant colors* of black (left) and brown (middle) both having the conditional probability over 90%, which can be used for filtering directly, while the right one is a *bad* context containing noise pixel in it. Fig. 3 shows an example of *good* and *bad* context distribution in a test image. The unmarked pixels belong to an *uncertain* context.

2.3 Voting Image

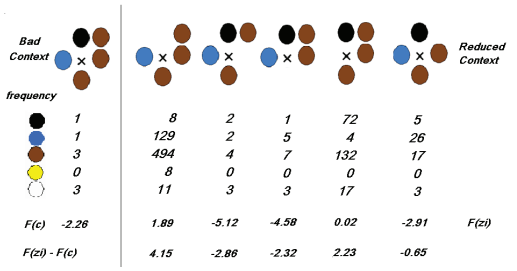

Fig. 2 *Bad context and its reduced context ,black pixel is possible a noise pixel with high F(**z**$_i$) - F(**c**) difference*

Since most of *bad* contexts contain noise pixels in themselves, they are seldom used to estimate a statistical model. However, they can be very useful in detection of noise pixels – except the case when a noisy pixel is isolated from most of inherited geometrical structures. Given an *bad* context **c**, a set of sub-contexts,

$$S(\mathbf{c}) = \{\mathbf{z}_i \mid \mathbf{z}_i = \mathbf{c} / \{x_i\}\}_{i=1}^{k} \quad (4)$$

are constructed by removing i^{th} pixel x_i from the context **c**, i = 1, …k, where $\mathbf{z}_i$ is the reduced-size context after removal of the i^{th} pixel. If the removed pixel x_i is a noise, it is expected that the reduced-size context will have a higher efficiency: F($\mathbf{z}_i$) > F(**c**), such that each pixel in **c** can be assigned with a meaningful value F($\mathbf{z}_i$) - F(**c**). The higher the difference, the more likely x_i is a noise pixel. Fig. 2 gives an example of *bad* context and its *reduced context*.

A *voting image R* is constructed according to the following rule: if the surrounding context **c** for a given pixel x is detected as *bad*, the accumulated voting score of each other pixel x_i in the same context **c** can be updated by:

$$\mathrm{R}(x_i) = \mathrm{R}(x_i) + \mathrm{F}(\mathbf{z}_i) - \mathrm{F}(\mathbf{c}) \quad (5)$$

Intuitively, we may conclude that most of contexts containing a noise pixel may be detected as *bad* contexts. If the noise pixel is removed from the *bad* context, the reduced-size context may have much better context efficiency. Namely, the accumulated voting score according to (5) is significantly higher than those of its neighborhood pixels. In this sense, the noise pixel can then be detected by finding the high peak points in the voting image. An example of a voting scheme is shown in Fig. 4.

2.4 Adaptive Context Selection

Once the voting image has been obtained, an adaptive context-based filter is applied in two manners. Firstly, if the surrounding context **c** in the context tree is *good*, there always exist one *dominant color* in the context, and eventually the value of the pixel x is replaced by the *dominant color* in that particular context if (2) meets. Secondly, if x is detected as a noise pixel in voting scheme and its surrounding context **c** is not labeled as *good*, the context is re-selected adaptively excluding those noise pixels using a 3×3 context template. Statistical distribution

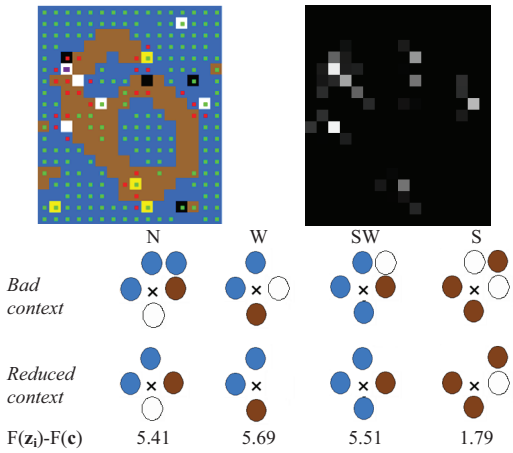

Fig. 3 *Sample image with good and bad context demonstrated in red and green colors (top left), its voting image (top right). Voting example for the white pixel labeled with purple with accumulated F(**z**$_i$)-F(**c**) value 18.30 (bottom).*

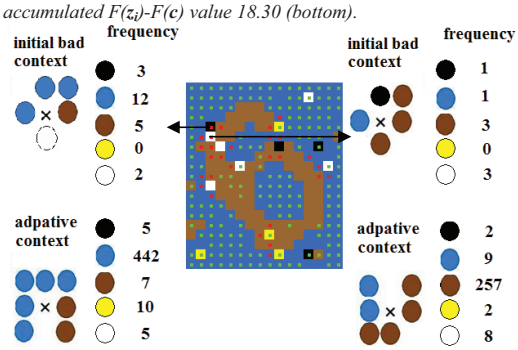

Fig. 4 *Example of adaptive context selection. For noise pixels (black and white, with high voting value), a new context in 3×3 region excluding surrounding noise pixels are selected, statistical information are collected for new contexts, black pixel is correctly changed to blue while white pixel changed to brown.*

of this adaptive context is collected and the DUDE framework is then applied. An example of adaptive context selection can be seen in Fig. 4. In order to improve the filtering robustness under different noise level, an estimation of δ is needed. Here, it is estimated by the minimum conditional probability occurred for contexts with "sufficient frequency", which is formulated as:

$$\delta = 1 - \max_{\forall x, p(\mathbf{c}) > 10^{-2}} P(x \mid \mathbf{c}) \quad (6)$$

The filtering can also be applied iteratively. Contrary to conventional filters, the images will not be degraded after applying several iterations.

2.5 Extension for Additive Gaussian Noise

During the map digitization, as limited color output is desired, a color quantization process can be applied for

reducing the number of colors. However, the noise will be incurred during the scanning process, which causes the possible overlapping in color space for some color components. Thus, we need to avoid misclassifying pixels caused by conventional color quantization. A novel iterative algorithm is proposed in [13] to optimize both the estimation of the indexed image and its color palette. For each pixel x, both the distance between RGB color vector to its corresponding component in the color palette, and its conditional probability of local context (estimated in Section 2.2-2.4) are taken into account as follows:

$$I(x) = \arg\min_{x=(1..M)}(-\log_2 f(\mathbf{y}_x \mid \mathbf{M}_x) - \log_2 P(x \mid \mathbf{c}))$$
$$f(\mathbf{y}_x \mid \mathbf{M}_x) = \exp(-\| \mathbf{y}_x - \mathbf{m}_x \|^2 / 2\sigma^2) \quad (7)$$

where $\mathbf{M}_x$ is 3-D *Gaussian distribution* with mean $\mathbf{m}_x$ and covariance matrix $\sigma^2\mathbf{I}$. This formula is similar to the energy function in *Markov random fields*, but the term *neighborhood homogeneity* is replaced by the conditional probability of the local context. The color components in color palette and the mean variance are updated following with the minimization step, see [13] for more details.

3. EXPERIMENTS

We evaluate the proposed *adaptive context-based filtering algorithm* (ACF) on a set of images from *National Land Survey of Finland*. For testing the performance of the filter, we artificially distort those images by adding impulsive or additive Gaussian noise. For comparison, four alternative filters [2, 3, 8, 10] are investigated for *impulsive noise* and four for *Gaussian noise* [4-7]. Performance comparisons are demonstrated in Fig. 5. It can be observes that the proposed filtering algorithm achieves significantly better numerical and visual quality.

4. CONCLUSION

We have proposed an efficient adaptive filtering using optimal context selection, which is designed via a novel voting-based noise estimation method. The proposed context-based filter can be viewed as a pilot study to conquer the raster map image distortion caused by uncertain noises. This algorithm can also be applied in other problem domains, such as image segmentation and color quantization.

5. REFERENCES

[1] M. Chen, M. Xu and P. Fränti, "Multi-layer filtering approach for map images", *IEEE Int. Conf. on Image Processing (ICIP'09)*, pp. 3953-3956, 2009.

[2] R. Lukac, "Adaptive vector median filtering", *Pattern Recognition Letters* 24, pp. 1889-1899, 2003.

[3] B. Smolka, A. Chydzinski, "Fast detection and impulsive noise removal in color images", *Real-Time Imaging*, 11(5-6), pp. 389-402, 2005.

[4] J. Portilla, V. Strela, M. Wainwright and E. P. Simoncelli, "Image denoising using a scale mixture of Gaussians in the wavelet domain," *IEEE Trans. on Image Proc,* 12(11), pp. 1338–1351, 2003.

[5] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising", *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, vol. 2, pp. 60-65,2005.

[6] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering", *IEEE Trans. on Image Proc*, 16(8), pp.2080–2095, 2007.

[7] A. Barbu. "Learning Real-Time MRF Inference for Image Denoising", *CVPR 2009.*

[8] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdru, and M. Weinberger, "Universal discrete denoising: Known channel," *IEEE Trans. Inform. Theory*, 51(1), pp.5-28, 2005.

[9] P. Kopylov and P. Fränti, "Filtering of color map images by context tree modeling", *IEEE Int. Conf. on Image Processing (ICIP'04)*, Singapore, vol. 1, pp. 267-270, October 2004.

[10] E. Ordentlich, M. J. Weinberger, and T. Weissman, "Multi-directional context sets with applications to universal denoising and compression", *In Proc. of the 2005 IEEE Intl. Symp. on Inform. Theory, (ISIT'05)*, pp. 1270-1274, Sept. 2005.

[11] J. Yu and S. Verd´u, "Schemes for bidirectional modeling of discrete stationary sources", *IEEE Trans. Inform. Theory*, 52(11), pp.4789–4807,2006.

[12] A. Akimov, A. Kolesnikov, P. Fränti, "Lossless compression of color map images by context tree modeling", *IEEE Trans. on Image Processing*, 16(1), pp. 114-120, 2007.

[13] M. Chen, M. Xu and P. Fränti, "Statistical filtering of raster map images", *IEEE Int. Conf. on Multimedia & Expo (ICME'10)*, pp. 394-399, 2010.

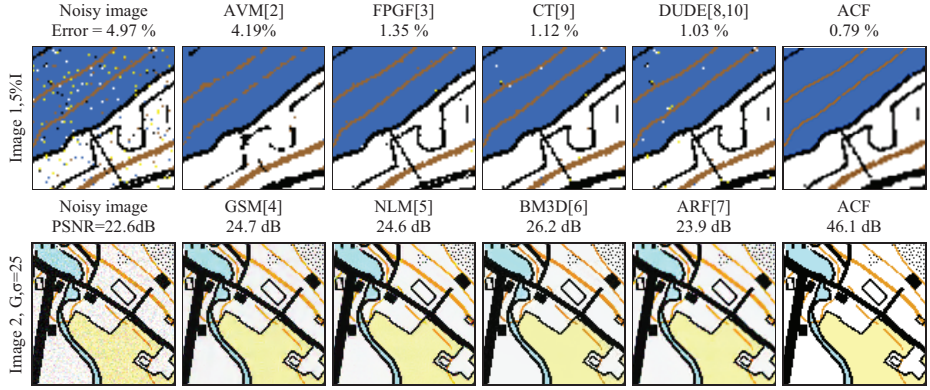

Fig. 5 *Performance comparison*

Paper P4

M. Chen, M. Xu, P. Franti, "A Fast $O(N)$ Multi-resolution Polygonal Approximation Algorithm for GPS Trajectory Simplification", *IEEE Trans. on Image Processing*, 21(5), 2770 – 2785, 2012.

A Fast $O(N)$ Multiresolution Polygonal Approximation Algorithm for GPS Trajectory Simplification

Minjie Chen, *Student Member, IEEE*, Mantao Xu, and Pasi Fränti, *Senior Member, IEEE*

***Abstract*—Recent advances in geopositioning mobile phones have made it possible for users to collect a large number of GPS trajectories by recording their location information. However, these mobile phones with built-in GPS devices usually record far more data than needed, which brings about both heavy data storage and a computationally expensive burden in the rendering process for a Web browser. To address this practical problem, we present a fast polygonal approximation algorithm in 2-D space for the GPS trajectory simplification under the so-called integral square synchronous distance error criterion in a linear time complexity. The underlying algorithm is designed and implemented using a bottom–up multiresolution method, where the input of polygonal approximation in the coarser resolution is the polygonal curve achieved in the finer resolution. For each resolution (map scale), priority-queue structure is exploited in graph construction to construct the initialized approximated curve. Once the polygonal curve is initialized, two fine-tune algorithms are employed in order to achieve the desirable quality level. Experimental results validated that the proposed algorithm is fast and achieves a better approximation result than the existing competitive methods.**

***Index Terms*—Geographic information systems (GISs), global positioning system trajectory simplification (GPS TS), polygonal approximation, priority queue, reduced search dynamic programming (RSDP).**

I. Introduction

LOCATION-ACQUISITION technologies, such as geopositioning mobile devices, enable users to obtain their locations and record travel experiences by a number of time-stamped trajectories. In the location-based Web services, users can record, then upload, visualize, and share those trajectories [34]. Therefore, people are more likely to find the travel routes that interest them and acquire reference knowledge facilitating their travel from other's trajectories. However, these GPS devices usually record far more data points than necessary, and these redundant data points will decrease the performance of the data collection. For example, if data are collected at 10-s intervals, a calculation in [32] shows that, without any compression, 100 Mb is required to store just 400 objects for a single day. Moreover, these redundant GPS trajectories will also cause a longer uploading/downloading time to the mobile service providers. The dense representation will also bring about a heavy burden for a Web browser when rendering these trajectories on the client side. In some cases, Web browsers may even get out of memory and crash. From our experiment, it takes approximately 1 s for rendering 1000 points on the map. Therefore, a fast polygonal approximation algorithm is needed for the trajectory simplification (TS) task, i.e., multiple GPS TSs are conducted corresponding to different map scale beforehand such that the trajectories can be efficiently visualized.

In recent years, polygonal approximation in 2-D space has attracted a considerable interest with a great deal of applications such as geographic information systems (GISs), computer graphics and data compression. Given a polygonal curve $P = (p_1, \ldots, p_n)$, the problem of polygonal approximation is to seek a set of ordered points P' (a subset of P), i.e.,

$$P' = (p_{i_1}, p_{i_2}, \ldots, p_{i_m}) \tag{1.1}$$

as an approximation of P, where $1 = i_1 < \ldots < i_m = N$. Polygonal approximation can be categorized into two classes of subproblems.

1) *min-ε problem*: Given N-vertices polygonal curve P and integer M, approximate a polygonal curve P' with the minimum approximation error with at most M vertices.
2) *min-# problem*: Given N-vertices polygonal curve P and error tolerance ε, approximate a polygonal curve P' with the minimum number of vertices within the error tolerance ε.

For polygonal approximation, there exist different solutions, which vary in reduction efficiency and computational overhead. For example, an optimal algorithm provides the best reduction efficiency but causes the highest overhead $O(N^2) - O(N^2 \log N)$ [1]–[5], [10]–[13], [15], whereas solutions based on heuristics lower the computational overhead at the cost of reduced reduction rates $O(N \log N)$ [7]–[9]. A compromise between the optimal and heuristic solutions is the *reduced search dynamic programming* (RSDP) [17], [18], [23]. The algorithm uses a *bounding corridor* surrounding a reference curve to limit the search space during the minimizing process. In different applications, different error criteria have been defined [1]–[5].

For the GPS TS, since both spatial and temporal information should be considered, a number of heuristic methods have also

Manuscript received May 11, 2011; revised November 10, 2011 and January 02, 2012; accepted January 09, 2012. Date of publication January 31, 2012; date of current version April 18, 2012. The work of M. Chen was supported in part by Tekniikan edistämissäätiö, under a Nokia Scholarship, under MOPSI Project EU (EAKR). The work of M. Xu was supported by the National Natural Science Foundation of China under Grant 61072146, and the Shanghai Committee of Science and Technology, China, under Grant 10PJ1404400. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ferran Marques.

M. Chen and P. Fränti are with the School of Computing, University of Eastern Finland, 80101 Joensuu, Finland (e-mail: mchen@cs.joensuu.fi; franti@cs.joensuu.fi).

M. Xu is with the School of Electrical Engineering, Shanghai Dianji University, Shanghai 200240, China (e-mail: xumt@sdju.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2012.2186146

been proposed with different error measures, such as TS [31], *top–down time ratio* (TD-TR) [32], *Open Window* (OW) [32], *threshold-guided algorithm* [33], *STTrace* [33], *spatial join* [35], Spatial QUalIty Simplification Heuristic (SQUISH) [37], and *generic remote* TS (GRTS) [38]. Performance evaluations are made for several traditional TS algorithms in [36]. In these algorithms, the performance is measured on the reduction rate by the line simplification process. It is noted in [37] that there is not one algorithm that always outperforms other approaches in all situations. In the GPS TS, the reduced data points are mostly directly saved with a fixed bit length, which is required to support both the rendering process and the effective trajectory queues in database. On the other hand, when data compression techniques are used, a better compression ratio is achieved for the GPS trajectory data [41], which is appropriate for data storage.

In this paper, we present a fast $O(N)$ time polygonal approximation algorithm for the GPS TS. The proposed method applies a joint optimization for both *min-# approximation* using the *local integral square synchronous Euclidean distance* (LSSD) criterion and *min-ε approximation* using the *integral square synchronous Euclidean distance* (ISSD) criterion.

The proposed GPS TS algorithm is implemented in a real-time application for the rendering process of the GPS trajectories on the map.[1]

II. Related Work

In this section, we will review the related work in the GPS TS in several aspects, such as error measures, approximation of the polygonal curves, fine-tune solutions by reduced search, and multiresolution polygonal approximation. The contributions of this paper are also summarized at the end of each subsection.

A. Error Measures

The primary goal of the GPS TS techniques is to reduce the data size without compromising much of its precision. Thus, there is a need to find appropriate error measures in algorithms and performance evaluation.

In polygonal approximation, different error criteria have been defined, such as *tolerance zone, parallel strip, uniform measure, minimum height*, and *minimum width* [1]–[5]. Later, Meratnia and de By [32] indicated that such algorithms were not suitable for GPS trajectory since both spatial and temporal information should be considered. Therefore, the errors were measured through distances between pairs of temporally synchronized positions, called *synchronous Euclidean distance* (SED).

The definition can be formulated as follows:

$P_i^j = (p_i, \ldots, p_j)$ is the subcurve of P, and $\overline{p_i p_j}$ is the line segment between p_i and p_j (an approximated edge in P'). For each point $p_k = (x_k, y_k)$ with time t_k $(i < k < j)$ on the original GPS trajectory, its approximated temporally synchronized position $p'_k = (x'_k, y'_k)$ can be calculated as

$$x'_k = x_i + \frac{t_k - t_i}{t_j - t_i} \cdot (x_j - x_i) \tag{2.1}$$

$$y'_k = y_i + \frac{t_k - t_i}{t_j - t_i} \cdot (y_j - y_i). \tag{2.2}$$

[1]Two datasets are considered, which are MOPSI dataset (http://cs.joensuu.fi/mopsi) and geolife dataset [34].

After the approximated position p'_k is determined, SED is calculated by

$$\mathrm{SED}\,(p_k, p'_k) = \sqrt{(x_k - x'_k)^2 + (y_k - y'_k)^2}. \tag{2.3}$$

In SED, the continuous nature of moving objects necessitates the inclusion of temporal and spatial properties.

Except for the aforementioned error measures, other error functions were also considered in some literatures. For example, position, speed, and orientation information were all used in the *threshold-guided algorithm* [33]. In [35], a new distance function called *spatial join* was proposed, which was bounded for spatial-temporal queries. In the area of shape matching, Fréchet distance [39] also took the continuity of shapes into account with a time complexity $O(MN)$, where M and N are the number of points correspondingly [40].

However, in most algorithms, in order to calculate the approximated error of the line segment $\overline{p_i p_j}$, at least $j - i$ distance calculations are needed. In [15], the calculation process was solved in dual space by a priority-queue structure, which achieved the best processing time $O(\log N)$ with a preprocessing time $O(N \log N)$.

In this paper, we further study the cost-effective spatiotemporal error measures, which can be computed in constant time. Namely, we extend *local integral square error* (LISE) criterion and *integral square error* (ISE) criterion [4]–[6] and derive two new error measures for the GPS TS problems, called LSSD and ISSD. LSSD and ISSD have the same properties with LISE and ISE, i.e., they can be computed efficiently in $O(1)$ time after precalculating all the accumulative terms within $O(N)$ time, whereas temporal information is also considered meanwhile. The further discussion of the error measures will be made in Section III.

B. Polygonal Approximation: Optimal and Heuristic Methods

Optimal polygonal approximation algorithms are mostly implemented by incrementally constructing a *directed acyclic graph* (DAG) and therefore inevitably suffer a computational cost limitation of $O(N^2)$ at the minimum [1]–[5], [10], [11], [13], [30]. An advance achieved by Agarwal and Varadarajan [12] is to combine an iterative graph algorithm and a divide-and-conquer approach, which offers the best time and space complexity of $O(N^{4/3+\delta})$ by using the L_1 *metric*, where $\delta > 0$ is an arbitrarily small constant. Later, the graph-based framework has been significantly reorganized and optimized by using two *priority queues* dynamically [15]. Albeit this approach was not proven to reduce the time complexity in theory, it provided remarkable improvement in the processing time in practice.

In real-time application, quadratic time complexity maybe too high, and therefore, most applications utilized a class of heuristic methods in order to achieve near-linear time complexity. A set of well-known heuristic algorithms are *split* and *merge* approaches [7]–[9]. The split algorithms divide the segment causing the biggest deviation, whereas the merge algorithms combine the pair of segments with the least deviation. The classic *Douglas–Peucker* (*D–P*) split algorithm [7] can be implemented in $O(N \log N)$ time on average,

while its worst case time complexity is $O(N^2)$. Later, Hershberger and Snoeyink [8] showed that it can be implemented in $O(N \log * N)$ time, where $\log *$ denotes the iterated logarithm function. Respectively, Pikaz and Dinstein [9] proposed a merging algorithm with $O(N \log N)$ time complexity. These heuristic methods are of low time complexity but may lead to an undesirable approximation result. Note that topological and geometric properties are also considered as an important constraint in the simplification process in GIS applications. In [44], *simple-detour heuristic* was proposed, where no new vertices would be introduced after the approximation process.

In the GPS TS, a number of algorithms have been also well studied and developed, and most of them are heuristic methods. In [32], a TS algorithm is greedily implemented by a so-called *opening-window* approach. SED is also defined and applied by incorporating the time dimension, instead of the original perpendicular distance. In [33], the parameters including coordinates, speed, and orientation are all considered in calculating the safe area of the next point, which is called as the *threshold-guided algorithm*. Indeed, all these algorithms solve the min-# problem in a greedy manner, the time complexity of which is $O(N^2)$. The *STTrace sampling algorithm* [33] is also implemented using a bottom–up strategy where the SED is minimized in each step. In [38], GRTS *protocol* combines optimal and heuristic algorithms [1], [32], which allows a tradeoff between the computational complexity and the reduction efficiency. Recently, a new simplification algorithm SQUISH [37] has been proposed based on the priority-queue data structure, which preserves speed information at a much higher accuracy. In [31], TS algorithm is proposed, where different point headcounts are assigned in terms of the product of the average heading change and the distance of each segment. After that, the min-ε problem is solved in each segment by using a local weighting process in $O(N \log M)$ time. However, as the distances of neighborhood points are used instead of the perpendicular distance in the simplification procedures, the algorithm is not robust when the sampling frequency is not uniform.

Graph-based methods can achieve a better approximation result than those heuristic ones but at a higher computational cost. Therefore, in the initialization process of the proposed solution, graph-based methods are used and further speeded up by both a novel priority-queue structure and a stopping search criterion, which leads to $O(N^2/M)$ time complexity and $O(N)$ space complexity. Here, N and M are the number of the points for the input and output GPS trajectories, respectively. However, using a stopping search criterion will cause a tradeoff of the optimality. This will be introduced in Section IV.

C. Fine Tune by Reduced Search

For the GPS TS, optimal algorithms provide the best reduction efficiency but cause the highest overhead, whereas solutions based on heuristics lower the computational overhead at the cost of worse reduction rates. A compromise between the optimal and heuristic solutions is the RSDP [17], [18], [23]. The algorithm uses a *bounding corridor* surrounding a reference curve or a initialized curve in the state space, followed by a limited search for the minimum cost path. This idea is presented and known as Sakoe–Chiba band [42], which has been extensively used in *dynamic time wrapping* approaches dealing with the similarity calculation of time series [43].

If the initialized curve is evenly distributed in the state space, the time complexity for RSDP is ideally $O(W^2N^2/M^2)$, where W is the width of the bounding corridor. We will also prove that the expected time complexity for RSDP is still achievable as $O(W^2N^2/M^2)$ even if the precondition of even distribution is not satisfied. In particular, if the number of vertices for the approximated curve is proportional to that of the input curve, namely, $M = N/c$, a linear time complexity can be achievable for the RSDP. This will be later shown to be an important property when selecting *bottom–up* approaches for the multiresolution case. However, the main difficulty of the RSDP is that a large corridor bound and many iterations are needed in order to achieve a desirable solution when the approximated curve is poorly initialized, which causes a high computational cost.

In this paper, we extend the RSDP and employ two fine-tune algorithms to minimize both the number of output points M and the approximated error ε, which leads to a time complexity $O(WN^2/M)$ and $O(N^2/M)$ correspondingly. The fine-tune algorithms are speeded up by lifting the vertex position in the tree structure, also solving the *equivalent solution problem*. This will be discussed in Section V.

In Sections III–V, the U.K. map with 10 911 points (see Fig. 13) will be selected as an example to demonstrate the proposed algorithm.

D. Multiresolution Polygonal Approximation

Multiresolution polygonal approximation can be applied for scalable representation and compression of vector maps in the GIS [19], [20]. For solving the min-ε problem, two heuristic approaches, i.e., split (top–down) and merge (bottom–up), are known with a time complexity of $O(N \log N)$. Split and merge are locally applied and can often result in undesirable approximation results in the later hierarchy process.

The *optimal split algorithm* is proposed in [21], where the optimal approximation at the higher resolution level is achieved using the result of the lower (previous) resolution level. This provides resolution hierarchy in a sequential order $(1 \rightarrow 2 \rightarrow 4 \rightarrow \ldots.)$ but at a cost of $O(N^2)$ time complexity.

In [22], a bottom–up multiresolution algorithm for the min-ε problem is proposed with near-linear time complexity. The min-ε problem is solved using the fine resolution as input for approximating the corresponding coarser resolution iteratively $(N \rightarrow N/2 \rightarrow N/4 \rightarrow \ldots.)$. For each scale, the simplified RSDP is also incorporated. As the ISE criterion is used, the approximation error between two vertices in any resolution level can be calculated in a constant time according to the precalculating cumulative summations in the original curve (see Section III).

Although the bottom–up approach [22] is computationally efficient, this approach can only solve the min-ε problem. In practice, in order to progressively display the GPS trajectory data, we need to approximate a number of approximated results with corresponding error tolerance for each resolution, which is considered as a min-# problem. Moreover, the reduced search algorithm is a fine-tune method, which needs an initial curve beforehand. If the curve is not well initialized, a number of iterations are needed to obtain the near-optimal result.

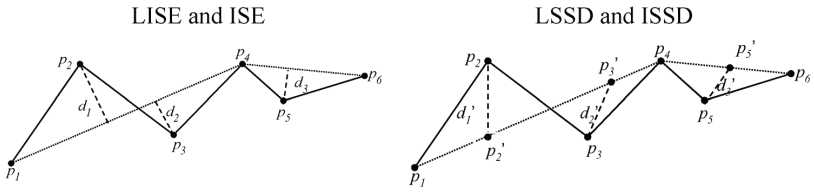

Fig. 1. Example of calculating ISE, LISE, LSSD, and ISSD. Given $P = (p_1, p_2, p_3, p_4, p_5, p_6)$ and the approximated curve $P' = (p_1, p_4, p_6)$, where p'_2, p'_3, and p'_5 are the approximated temporally synchronized position. (Left) ISE is estimated as $d_1^2 + d_2^2 + d_3^2$, and LISE is estimated as $d_1^2 + d_2^2$. Meanwhile, (Right) ISSD is estimated as $d_1'^2 + d_2'^2 + d_3'^2$, and LSSD is estimated as $d_1'^2 + d_2'^2$.

In this paper, a *bottom–up multiresolution* approach is proposed with linear time and space complexities, which implements the algorithms in Sections III–V for each intermediate resolution. This will be discussed in Section VI.

III. Error Measure: From LISE to LSSD

In order to improve the computational efficiency, two error measures, which are called ISE and *local* ISE [4]–[6], are jointly used for approximating polygonal curves, i.e.,

$$f_{\mathrm{ISE}}(P') = \sum_{j=1}^{M-1} \delta\left(P_{i_j}^{i_{j+1}}\right) \tag{3.1}$$

$$f_{\mathrm{LISE}}(P') = \max_{1\le j<M} \delta\left(P_{i_j}^{i_{j+1}}\right) \tag{3.2}$$

where error δ can be calculated by

$$\begin{aligned}\delta\left(P_i^j\right) &= \sum_{i<k<j} d^2(p_k, \overline{p_i p_j}) \\ &= \frac{1}{a_{ij}^2 + b_{ij}^2} \sum_{i<k<j} (a_{ij}\cdot x_k + b_{ij}\cdot y_k + c_{ij})^2 \\ &= \Big((j-i-1)\cdot c_{ij}^2 + a_{ij}^2 \cdot \left(S_{x^2}^{j-1} - S_{x^2}^{i}\right) \\ &\quad + b_{ij}^2 \cdot \left(S_{y^2}^{j-1} - S_{y^2}^{i}\right) + 2\cdot a_{ij}b_{ij}\cdot \left(S_{xy}^{j-1} - S_{xy}^{i}\right) \\ &\quad + 2\cdot a_{ij}\cdot c_{ij}\cdot \left(S_{x}^{j-1} - S_{x}^{i}\right) \\ &\quad + 2\cdot b_{ij}\cdot c_{ij}\cdot \left(S_{y}^{j-1} - S_{y}^{i}\right)\Big) / \left(a_{ij}^2 + b_{ij}^2\right). \end{aligned} \tag{3.3}$$

Here, d is the perpendicular distance from p_k to $\overline{p_i p_j}$. $a_{ij} = (y_j - y_i)$, $b_{ij} = (x_i - x_j)$, $c_{ij} = y_i x_j - x_i y_j$, and S_x, S_y, S_{x2}, S_{y2}, S_{xy} are the accumulated sums of the x and y coordinates on curve P, respectively, i.e.,

$$\begin{aligned} S_x^i &= \sum_{j=1}^{i} x_j \quad S_y^i = \sum_{j=1}^{i} y_j \quad S_{x^2}^i = \sum_{j=1}^{i} x_j^2 \\ S_{y^2}^i &= \sum_{j=1}^{i} y_j^2 \quad S_{xy}^i = \sum_{j=1}^{i} x_j y_j \quad i = 1, \ldots, N. \end{aligned} \tag{3.4}$$

The main advantage of the ISE criterion is that the approximation error $\delta(P_i^j)$ can be efficiently obtained in $O(1)$ time after precalculating all the accumulative terms within $O(N)$ time [see (3.3)] [4], [16]. An example of calculating ISE and LISE is illustrated in Fig. 1.

Although LISE and ISE criteria are computationally efficient, time information is not considered. For the simplification of the GPS trajectories, we extend LISE and ISE criteria and derive two new error measures, called LSSD and ISSD, which have the same properties with LISE and ISE, i.e.,

$$f_{\mathrm{ISSD}}(P') = \sum_{j=1}^{M-1} \delta_{\mathrm{SED2}}\left(P_{i_j}^{i_{j+1}}\right) \tag{3.5}$$

$$f_{\mathrm{LSSD}}(P') = \max_{1\le j<M} \delta_{\mathrm{SED2}}\left(P_{i_j}^{i_{j+1}}\right). \tag{3.6}$$

Here

$$\begin{aligned}&\delta_{\mathrm{SED2}}\left(P_i^j\right) \\ &\quad = \sum_{i<k<j} \mathrm{SED}^2\left(p_k, p'_k\right) \\ &\quad = \left(c_1^2 + c_3^2\right)(j-i-1) + \left(c_2^2 + c_4^2\right)\left(S_{t^2}^{j-1} - S_{t^2}^{i}\right) \\ &\qquad + 2(c_1c_2 + c_3c_4)\left(S_t^{j-1} - S_t^i\right) \\ &\qquad + \left(S_{x^2}^{j-1} - S_{x^2}^{i}\right) + \left(S_{y^2}^{j-1} - S_{y^2}^{i}\right) \\ &\qquad - 2c_1\left(S_x^{j-1} - S_x^i\right) - 2c_3\left(S_y^{j-1} - S_y^i\right) \\ &\qquad - 2c_2\left(S_{xt}^{j-1} - S_{xt}^i\right) - 2c_4\left(S_{yt}^{j-1} - S_{yt}^i\right) \end{aligned} \tag{3.7}$$

Here

$$\begin{aligned} c_1 &= \frac{x_i t_j - x_j t_i}{t_j - t_i} \quad c_2 = \frac{x_j - x_i}{t_j - t_i} \\ c_3 &= \frac{y_i t_j - y_j t_i}{t_j - t_i} \quad c_4 = \frac{y_j - y_i}{t_j - t_i}. \end{aligned}$$

S_x, S_y, S_t, S_{x2}, S_{y2}, S_{t2}, S_{tx}, and S_{ty} are the accumulated sums of x, y, and t on the GPS trajectory, respectively, i.e.,

$$\begin{aligned} S_x^i &= \sum_{j=1}^{i} x_j \; S_y^i = \sum_{j=1}^{i} y_j \; S_t^i = \sum_{j=1}^{i} t_j \; S_{x^2}^i = \sum_{j=1}^{i} x_j^2 \\ S_{y^2}^i &= \sum_{j=1}^{i} y_j^2 \; S_{t^2}^i = \sum_{j=1}^{i} t_j^2 \; S_{tx}^i = \sum_{j=1}^{i} t_j x_j \; S_{ty}^i = \sum_{j=1}^{i} t_j y_j. \end{aligned} \tag{3.8}$$

The computation of the aforementioned approximation errors $\delta_{\mathrm{SED2}}(P_i^j)$ also takes $O(1)$ time with an $O(N)$ time accumulated sum precalculation. The proof of the LSSD and ISSD calculation is shown in the Appendix.

In the following sections, ISE and LISE criteria will be used for the approximation of the polygonal curves, whereas LSSD and ISSD criteria will be used for the GPS TS.

IV. Min-# Initialization for GPS TS

For the *min-# problem*, Imai and Iri's graph-based approach [1] comprises two essential steps, i.e., constructing a DAG and the shortest path search by *breadth-first traversal* (BFT). In order to construct a DAG, $N(N-1)/2$ approximation errors are calculated for every pairs of vertices, and thus, the time complexity for initializing the solution for the min-# problem is $O(N^2)$ if the LISE or LSSD criterion is applied.

In this paper, we revisit two computationally efficient improvements for the min-# problem. The first improvement is to reduce the computational cost of the DAG construction by maintaining two *priority-queue* structures [15], [29]. The reason is that there is no need to construct graph G explicitly and only edges visited by the BFT are included. For simplicity, we define a term, i.e., the *number of links* $L(p_i)$, to denote the minimum number of line segments to connect the starting vertex p_1 to p_i under a given error tolerance ε, i.e.,

$$L(p_i) = \min\left(L(p_k)\right) + 1 \quad \text{s.t. } \delta\left(P_k^i\right) < \varepsilon, 1 \le k < i \quad (4.1)$$

where the initial condition is set as $L(p_1) = 0$. Suppose all the vertices with k links are first identified by the shortest path search, which is maintained by a priority queue $V_{L(k)}$ in the descending order. The next search will be performed on the remaining unvisited vertices set S_u by testing if they have an edge connecting with vertices in $V_{L(k)}$ (i.e., approximation error lower than a given tolerance ε), which is called as *edge tests* here. These connected vertices will be removed from unvisited vertices set S_u and enqueued in the priority queue $V_{L(k+1)}$. Supposing two vertices $p_a, p_b \in V_{L(k)}$ with $a < b$, if $\exists p_c \in S_u$ and $\delta(P_b^c) < \varepsilon$, then p_c will be removed from S_u such that the edge test between p_a and p_c can be avoided. Moreover, edge tests are also avoided for the vertices with the same number of links. After all the unvisited points have been tested between $V_{L(k)}$ and S_u, in the next step, the vertices in $V_{L(k+1)}$ will be used as the starting points for edge tests. The shortest path search will be terminated when the last vertex p_n is connected to p_1. Albeit the priority-queue-based search is not able to mitigate the worst case time complexity, it turns out that a number of edge tests are greatly saved.

The second improvement is to apply a stopping criterion in the shortest path search, which is efficient in the case of low error tolerance. For example, a good stopping criterion has been proposed for the *tolerance zone criterion* [11] by maintaining the intersection of two cones. An alternative solution has been also proposed in [15] and [28] by verification in dual space. Both of the implementations hold the optimality for solving the min-# problem. To pursue the best computational cost savings as possible for LISE/LSSD criteria, a simple stopping criterion is applied in edge tests by utilizing a preset *high threshold*, e.g., two times of a given tolerance [17]. Edge tests for the subsequent vertices in the unvisited vertices set will be omitted once the approximation error becomes larger than a given high threshold. Applying a stopping criterion leads to a significant improvement to a time complexity of $O(N^2/M)$ but the optimality is not guaranteed. To overcome this difficulty, we extend our effort in improving the robustness of the stop search criterion. Instead of using a fixed high threshold, we adopt the error tolerance of the next coarser resolution as a high threshold in the multiresolution implementation, the robustness of which has been validated by experiments; see Section VI for additional discussion.

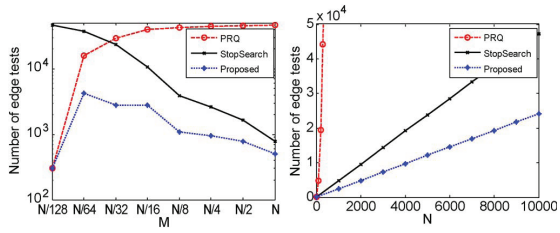

Fig. 2. Number of *edges tests* for solving the *min-# problem* (left) under different error tolerance and (right) with different number of input vertices for U.K. map (Curve II). In the left figure, the resulting number of output vertices M is shown in the x-axis instead of the given error tolerance.

We combine both the advantage of the priority-queue structure and the stopping criterion to achieve the most computationally efficient implementation in the initialization of the min-# problem. Accordingly, the output is a tree structure [see Fig. 5(left)]. The pseudocode is given in Fig. 3. Both the theoretical proof and the experiments are given for the complexity analysis of the proposed initialization algorithm.

Theorem 1: The proposed initialization algorithm for solving the min-# problem under the LISE/LSSD criterion leads to an expected time complexity of $O(N^2/M)$ and a space complexity of $O(N)$, respectively.

Proof: See Appendix.

In the graph-based initialization algorithm, the main bottleneck is the cost of edge tests (calculating the edge approximation errors, line 22 of Algorithm I) during graph construction. In order to evaluate the computational improvement achieved by the proposed algorithm, the number of edge tests is calculated and treated as an indicator of the computational efficiency in Fig. 2. Here "PRQ" represents the previous graph-based polygonal approximation algorithm using the priority-queue structure [15], [29], and "StopSearch" is the stopping criterion using a predefined high threshold [17]. It can be observed that the proposed algorithm is able to combine the computational advantages of both two algorithms.

V. Fine Tuning the Initial Approximated Result

As a stopping criterion is incorporated in Algorithm I (line 27) to reduce the computational cost in the initial approximation process, the optimality is not guaranteed. Thus, two fine-tune algorithms are introduced in this section in order to improve the approximation performance. Both the number of vertices and the ISE/ISSD are minimized.

A. Minimizing the Number of Vertices

To the benefit of best computational efficiency, the initialization in Algorithm I for the min-# problem is a compromise of the optimality for minimizing the number of vertices. In order to mitigate the limited optimality, we need to minimize the number of vertices based on the initialized curve so that a better result can be achieved. The reduced search algorithm (RSDP) can be utilized for minimizing the number of vertices, but it

ALGORITHM I, MIN-# INITIALIZATION

```
1.  INPUT
2.  P={p1, p2,...pN}: original polygonal curve
3.  th: LISE /LSSD error tolerance
4.  hth: high threshold of error tolerance
5.  OUTPUT
6.  T : Tree structure
7.
8.  V1 ← {1}
9.  V2 ← Ø
10. Su ← {2,...,N}
11. REPEAT
12.    REPEAT
13.       maintainPRQ()
14.    UNTIL V1 = Ø
15.    V1 ← V2
16.    V2 ← Ø
17. UNTIL pN ∈ V1
18.
19. Procedure maintainPRQ()
20. ind1 ← dequeue(V1)
21. FOR ind2 = Su (1) TO Su (end)
22.    dist ← δ(P_ind1^ind2)
23.    IF dist ≤ th
24.       Su ← { Su \ ind2 }
25.       V2 ← enqueue(ind2)
26.       Update T by ind1.child ← ind2, ind2.father ← ind1
27.    ELSE IF (dist> hth)
28.       break
29.    ENDIF
30. ENDFOR
31. RETURN V1, V2, Su, TT
```

Fig. 3. Pseudocode of min-# initialization.

ALGORITHM II, MINIMIZING THE NUMBER OF OUTPUT VERTICES

```
1.  INPUT
2.  P={p1,p2,...pN}: original polygonal curve
3.  T: tree structure
4.  W: width of bounding corridor
5.  th: LISE /LSSD error tolerance
6.  OUTPUT
7.  T: updated tree structure
8.
9.  PRQ1 ← {1}
10. REPEAT
11.    PRQ2  ← child nodes of all vertices in PRQ1
12.    Vtar{k} ← Ø, k ∈ {0, 1, 2, ..., W}
13.    Vtar {0} ← PRQ2
14.    FOR k = 1 TO W
15.       Vtar {k} ← child nodes of all vertices in Vtar {k-1}
16.    ENDFOR
17.    FOR k = W TO 1
18.       FOR ind1 = PRQ1(1) TO PRQ1(end)
19.          FOR ind2 = Vtar {k}(1) TO Vtar {k}(end)
20.             dist← δ(P_ind1^ind2)
21.             IF dist ≤ th
22.                Vtar {k}  ← { Vtar {k} \ind2 }
23.                PRQ2  ← enqueue(ind2)
24.                Update T by ind1.child ← ind2, ind2.father ← ind1
25.             ENDIF
26.          ENDFOR
27.       ENDFOR
28.    ENDFOR
29.    PRQ1 ← PRQ2
30.    PRQ2 ← Ø
31. UNTIL pN ∈ PRQ1
```

Fig. 4. Pseudocode of minimizing the number of vertices.

leads to $O(W^2N^2/M)$ time complexity. To speed up the procedure, we exploit a new fine-tune method at a time complexity of $O(WN^2/M)$ instead, which is achieved by lifting the vertex position in the output tree structure after the initialization step in Algorithm I. The pseudocode is given in Fig. 4.

A graphical illustration is demonstrated in Fig. 5 on lifting vertex position; starting from vertex p_1 with 0 links, at each iteration (lines 11–30 in Algorithm II), edge tests are performed to verify if the approximation error is less than the given tolerance between the currently processed vertices with k links and those target vertices with $\{k+2, \ldots, k+W+1\}$ links. An example is given in Fig. 5 (left) when the width of the bounding corridor is $W = 2$. Supposing p_2 and p_3 are the vertices with one link, all the vertices with three links (p_7 and p_9) and four links (p_8, p_{10}, p_{11}, and p_{12}) are chosen as the target vertices for edge tests. If the connected edge exists, the tree structure is updated by lifting the target vertices (lines 22–24). The process of updating the tree structure can be recursively done [see Fig. 5 (right)]. The proposed fine-tune algorithm provides the following advantages over the original reduced search approach for the min-# problem. First, the calculation of the approximated errors between any pair of vertices with adjacent number of links is unnecessary and can be omitted. Second, once the tree structure is updated by the lifting operations, edge tests for those lifted vertices are also avoided.

Theorem 2: The proposed algorithm for the output vertex reduction under the LISE/LSSD criterion has an expected time complexity of $O(WN^2/M)$ and a space complexity of $O(N)$, respectively. The original RSDP method has an expected time complexity of $O(W^2N^2/M)$.

Proof: See Appendix.

Intuitively, the fine-tune algorithm can be also iteratively done. However, since the graph-based method has already achieved an ideal initial approximation, according to our experiments, optimal results can be derived in most cases by setting $W = 2$ with one iteration. The main bottleneck here is also the number of edge tests (line 20 in Algorithm II). In Fig. 6, the actual time cost is evaluated by calculating the number of edge tests against the three parameters, i.e., the width of the bounding corridor W, the number of output vertices M, and the number of input vertices N. To further demonstrate the efficiency of the proposed fine-tune algorithm, we also evaluated the performance when the initialization step is skipped and the original polygonal curve is selected as input directly. We can observe that the optimal result is achieved with less than five iterations by the proposed fine-tune algorithm and the number of edge tests is much less than the RSDP, which is shown in Fig. 7.

B. Minimizing the Global Integral Square Error

After the number of vertices is reduced by the LISE/LSSD criterion, a so-called *equivalent solution problem* may still exist. In other words, given an error tolerance ε, a number of solutions for the min-# approximation can be achieved with the same number of output vertices M, but they lead to distinct approximation performance (see Fig. 9). Hence, an additional postprocessing step based on the ISE/ISSD criterion is needed in order

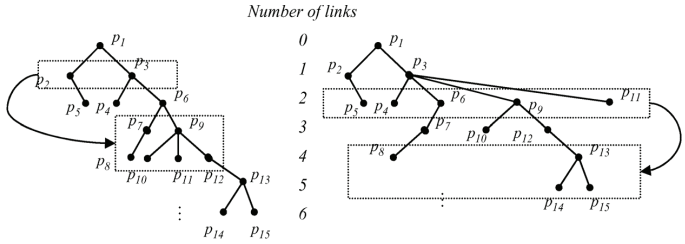

Fig. 5. Example of reducing the number of output vertices with a width of bounding corridor $W = 2$: (left) the target vertices with one link and (right) the target vertices with two links after tree structure updated given $\delta(P_3^9) < \varepsilon, \delta(P_3^{11}) < \varepsilon$. (Left figure) Typical example after the initialization step for Algorithm I.

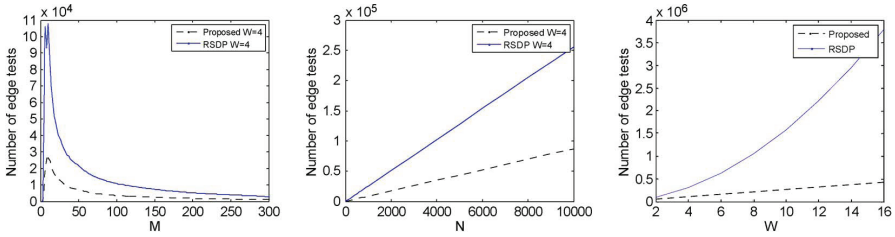

Fig. 6. Number of edge tests in minimizing the number of output vertices. (Left) Different error tolerance, (middle) different number of input vertices, and (right) different width of bounding corridor W are tested on U.K. map (Curve II). (Left figure) The resulting number of output vertices M is shown in the x-axis instead of the given error tolerance. (Left and right figures) The input polygonal curve is the U.K. map with $N = 10911$.

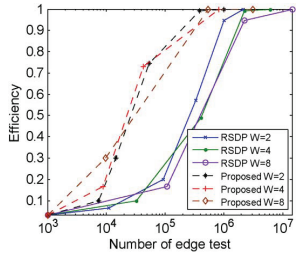

Fig. 7. Performance comparisons of the proposed fine-tune algorithm and RSDP when the original curve is selected as the initial curve directly. U.K. map (Curve II) is tested with $\varepsilon = 0.01$.

to find the best approximation result among these equivalent solutions, which can be also considered as a min-ε problem. The pseudocode is shown in Fig. 11.

After executing Algorithm II, which effectively updates the tree structure, additional postprocessing is performed to identify the best possible curve P' with the minimum ISE/ISSD, i.e.,

$$P' = \arg\min f_{\text{ISE/ISSD}}(P') \quad \text{s.t. } f_{\text{LISE/LSSD}}(P') < \varepsilon \tag{5.1}$$

This can be solved by dynamic programming in terms of the following recursive expression:

$$\begin{aligned} D(p_j) &= \min\left(D(p_i) + \delta\left(P_i^j\right)\right), 1 \le i < j \\ A(p_j) &= \arg\min_i \left(D(p_i) + \delta\left(P_i^j\right)\right), 1 \le i < j \\ &\text{s.t. } \delta\left(P_i^j\right) < \varepsilon, \quad L(p_j) = L(p_i) + 1 \end{aligned} \tag{5.2}$$

where $A(p_j)$ is the parent vertex of p_j and $D(p_j)$ is the accumulated ISE/ISSD.

Theorem 3: The minimization of the global ISE/ISSD under the constraint of the LISE/LSSD has an expected time complexity of $O(N^2/M)$ and a space complexity of $O(N)$.

Proof: See Appendix.

From Theorem 3, the minima can be found in $O(N^2/M)$ time, and no iterations are needed. The aforementioned minimization offers a significant improvement (theoretically W^2 time faster) over the original RSDP that has a time complexity of $O(W^2N^2/M)$. In Fig. 10, the histograms of the approximated LISE are plotted before and after the fine-tune step. As the ISE is the sum of the LISE for all the approximated segments, we can observe that the ISE is significantly reduced, whereas the LISE has not increased after the fine-tune process.

C. Summary of the Near-Optimal Approximation Algorithm

The polygonal approximation algorithm for the joint optimization of both the min-# approximation using the LISE/LSSD criterion and the min-ε approximation using the ISE/ISSD criterion has been introduced as a three-step procedure, i.e., the initialization of the min-# problem, minimizing the number of output vertices, and minimizing the ISE/ISSD. Proof has been given that the proposed algorithm has expected time complexity of $O(N^2/M)$ and space complexity of $O(N)$, and experiment results have demonstrated that the practice is consistent with the theoretical analysis. An example of the proposed algorithm is shown in Fig. 8. The improvement of the time complexity is also summarized in Table I.

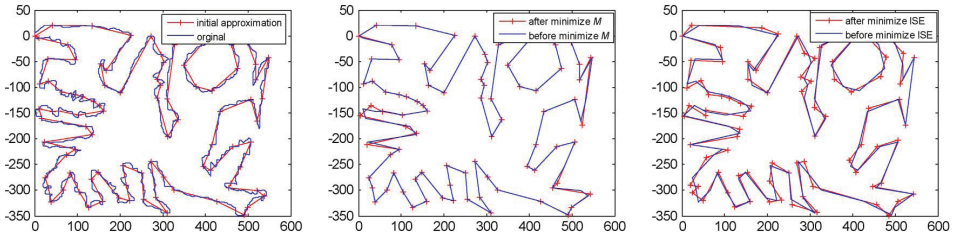

Fig. 8. Example of the proposed polygonal approximation. Curve I [25] is used with $\varepsilon = 1500$, and the optimal solution is $M_{\text{opt}} = 86$. (Left) Initial approximated curve is obtained with $M'' = 91$. (Middle) Approximated curve ($M = 86$) is obtained after reducing number of output vertices with $f_{\text{ISE}}(P') = 1.04 \cdot 10^5$. (Right) The final solution is obtained by minimizing ISE with $f_{\text{ISE}}(P') = 4.88 \cdot 10^4$.

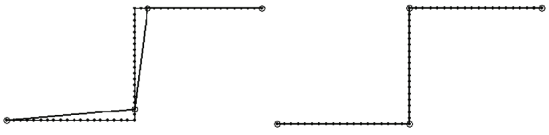

Fig. 9. Example of equivalent solutions in *min-# approximation*, where both approximated curves meet the error tolerance $\varepsilon = 2$ and have same output $M = 4$.

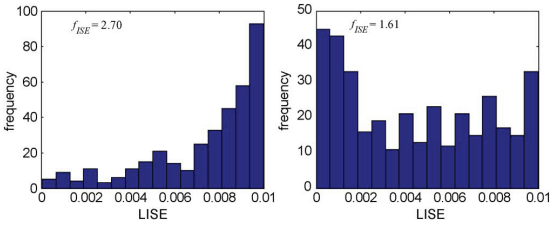

Fig. 10. LISE distribution of all the approximated edges with $\varepsilon = 0.01$ for Curve II. The best approximation result (right) with $f_{\text{ISE}}(P') = 1.61$ is found from all the *equivalent solutions*, which is much lower than result after Algorithm II (left) with $f_{\text{ISE}}(P') = 2.70$. Both approximation results have $M = 364$.

VI. Linear-Time Multiresolution Polygonal Approximation Method

In order to further improve the computational efficiency, in this section, a *bottom–up multiresolution polygonal approximation* approach is proposed by implementing Algorithms I and III in Sections III–V in each map scale, which achieves linear time and space complexity. Given an error tolerance ε, a joint optimization for both the min-# approximation using the LISE/LSSD criterion and the min-ε approximation using the ISE/ISSD criterion is solved. The underlying algorithm consists of three sequential procedures.

1) Error tolerance initialization. Initialize $\log_c N$ error tolerances $\{e_1^*, e_2^*, e_3^*, \ldots\}(e_1^* < e_2^* < e_3^* \ldots)$.
2) Initial curve approximation. A number of polygonal curves $\{P_1^*, P_2^*, \ldots, P_k^*\}$ are approximated based on the bottom–up multiresolution approach with corresponding error tolerance $\{e_1^*, e_2^*, e_3^*, \ldots\}$. Algorithms I and III are used for approximating the curve of each resolution.
3) Final approximation. A polygonal approximation is conducted under the given error tolerance ε by selecting

ALGORITHM III, FIND BEST SOLUTION USING INTEGRAL SQUARE ERROR CRITERION

```
1.  INPUT
2.  P={p1,p2,...pN} ← original polygonal curve
3.  T ← tree structure
4.  th ← LISE/LSSD error tolerance
5.  OUTPUT
6.  P' ← approximated curve
7.
8.  E ← {0, ∞, ∞, ....., ∞}, N x 1 vector storing the approximated error
9.  A ← {0, 0, 0, ....., 0}, N x 1 vector for backtracking
10. H ← {0, 0, 0, ....., 0}, M x1 vector
11. V1 ← {1}
12. M ← 1
13. REPEAT
14.   V2 ← child nodes of all the vertices in V1
15.   FOR ind1 = V1(1) TO V1(end)
16.     FOR ind2 = V2(1) TO V2(end)
17.       dist ← δ(P_ind1^ind2)
18.       IF (E(ind1) + dist < E(ind2)) && (dist ≤ th)
19.         A(ind2) ← ind1
20.         E(ind2) ← E(ind1) + dist
21.       ENDIF
22.     ENDFOR
23.   ENDFOR
24.   V1 ← V2
25.   M ← M + 1
26. UNTIL E(N)!= ∞
27. //Backtracking
28. H(M) ← N
29. FOR m = M TO 2 DO
30.   H(m-1) ← A(H(m))
31. ENDFOR
32. P' ← P(H)
```

Fig. 11. Pseudocode of minimizing ISE.

TABLE I
SUMMARY OF THE PROPOSED POLYGONAL APPROXIMATION ALGORITHM. $*$ REPRESENTS THAT THE INITIAL CURVE IS EQUALLY PARTITIONED

Step	Time Complexity		Improvements and Contributions
	RSDP	Proposed	
I	$O(N^2/M)$	$O(N^2/M)$	Combine priority queue and stopping criterion to reduce the computation cost. Proof is given.
II	$O(W^2N^2/M)*$	$O(WN^2/M)$	Time complexity is reduced. Proof is given.
III	$O(W^2N^2/M)*$	$O(N^2/M)$	Time complexity is reduced. Proof is given.

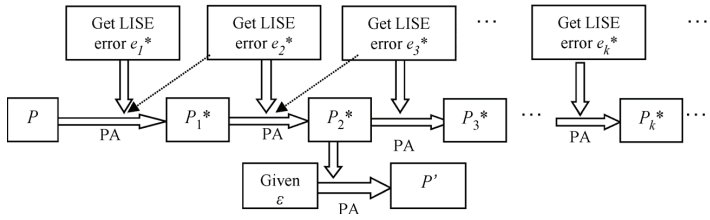

Fig. 12. Workflow of the proposed bottom–up multiresolution method. Error tolerance of coarser resolution is selected as high threshold for polygonal approximation, which is labeled by dashed line in the figure. In this example, if $e_2^* < \varepsilon < e_3^*$, then the approximation of P_3^* and $P_4^*, \ldots$ can be skipped.

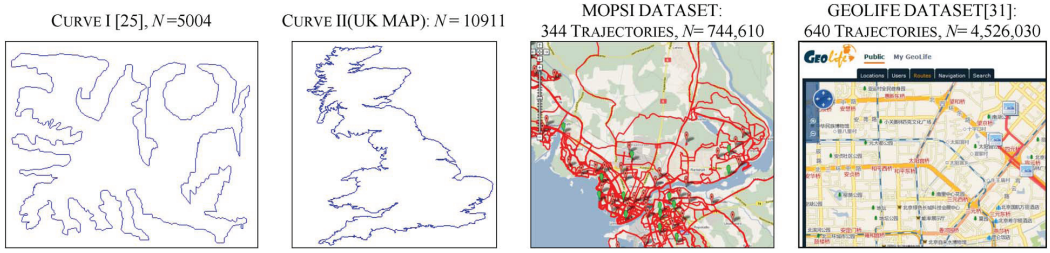

Fig. 13. Testing data in the experiments.

the most suitable input curve among those approximated curves $\{P_1^*, P_2^*, \ldots, P_k^*\}$.

In step 1, the error tolerances $e_1^*, e_2^*, e_3^* \ldots (e_1^* < e_2^* < e_3^* \ldots)$ are estimated according to the LISE/LSSD error criterion, i.e.,

$$e_k^* = \frac{1}{N/c^k - 1} \sum_{1 \le j < N/c^k} \delta\left(P_{i_j}^{i_{j+1}}\right)$$
$$i_j = \frac{N-1}{N/c^k - 1} \cdot (j-1) + 1. \tag{6.1}$$

Here $c > 1$ is a parameter to control the number of intermediate scale. For example, if $c = 2$, in each scale, the number of points will be around $N \to N/2 \to N/4 \to \ldots$. The aforementioned estimation can be viewed as the average LISE/LSSD error for all approximated segments when the curve is equally partitioned. The approximated curve under the error tolerance e_k^* has property $M_k \approx N/c^k$, where M_k is the number of output vertices in the kth resolution. Note that there are less intermediate scales when a larger c is selected, thus achieving a better reduction rate at the cost of a higher computational cost. When $c \to \infty$, there are no intermediate scales, and it is exactly the approximation algorithm that we described with $O(N^2/M)$ time complexity (Algorithms I–III).

In step 2, a bottom–up multiresolution algorithm is applied to estimate the approximated curves $P_1^*, P_2^*, P_3^*, \ldots$ under the corresponding error tolerances $e_1^*, e_2^*, e_3^*, \ldots$. Here, e_{k+1}^* is used as the high threshold in the approximation procedure of resolution k. The approximated result achieved in the previous finer resolution is used as the input of polygonal approximation in the next coarser resolution $(N_{k+1} = M_k)$, where Algorithms I and III are applied in each approximation. Since the optimality of these initial approximation results is not significantly compromised, the step of minimizing the number of vertices described in Algorithm II can be omitted.

In step 3, given an error tolerance ε, a polygonal approximation is conducted to obtain the final approximation result by selecting the most suitable input P_k^* among those approximated curves in step 2 such that

$$k = \arg\max_k \left(e_k^* < \varepsilon\right). \tag{6.2}$$

The workflow of the proposed algorithm is presented in Fig. 12. As the time complexity of the approximation process is $O(N_k^2/M_k)$ on each resolution, we have the following theorem:

Theorem 4: Both the time complexity and the space complexity of the proposed bottom–up multiresolution algorithm are $O(N)$.

Proof: See Appendix

Corollary 4.1: Given $0 < \varepsilon_1 < \varepsilon_2 < \ldots < \varepsilon_R$ as the R number of error tolerances, its corresponding approximated curves can be also constructed in linear time.

Proof: As the approximated curve for error tolerance ε_i can be used as the input for approximating the curve with error tolerance ε_{i+1}, the total time complexity is $O(N + M_1 + M_2 + \ldots) = O(N)$. □

VII. EXPERIMENTS

In order to evaluate the performance of the proposed *multiresolution polygonal approximation algorithm*, two polygonal curves are used as a test case. Curve I is an artificial curve used in [25] with 5004 vertices; curve II is the U.K. map contour with 10 911 vertices. For the GPS TS algorithm, two datasets are used, which are the MOPSI dataset and the Geolife dataset [31]. The graphical presentations are shown in Fig. 13.

TABLE II
COMPARISON OF THE EFFICIENCY AND THE PROCESSING TIME ($c = 2$)

CURVE I	M_{opt}	EFFICIENCY			TIME COST (MS)			
		SPLIT[7]	MERGE[9]	PROPOSED	SPLIT[7]	MERGE[9]	MRPA	OPTIMAL[1]
$\varepsilon_1 = 1$	824	0.68	0.81	**0.85**	7	3	6	847
$\varepsilon_2 = 100$	193	0.66	0.74	**0.78**	6	4	6	791
$\varepsilon_3 = 10^4$	49	0.68	0.71	**0.77**	4	4	7	794
CURVE II	M_{opt}	EFFICIENCY			TIME COST (MS)			
		SPLIT[7]	MERGE[9]	PROPOSED	SPLIT[7]	MERGE[9]	MRPA	OPTIMAL[1]
$\varepsilon_1 = 10^{-4}$	1986	0.71	0.83	**0.86**	18	11	15	3699
$\varepsilon_2 = 10^{-2}$	364	0.66	0.72	**0.75**	14	12	17	3678
$\varepsilon_3 = 1$	72	0.66	0.70	**0.75**	11	13	17	3592

TABLE III
EFFICIENCY AND PROCESSING TIME FOR CURVES I AND II WHEN DIFFERENT c IS SELECTED

CURVE I	M_{opt}	EFFICIENCY			TIME COST (MS)		
		$c = 1.5$	$c = 2$	$c = 4$	$c = 1.5$	$c = 2$	$c = 4$
$\varepsilon_1 = 1$	824	0.82	0.85	0.87	7	6	9
$\varepsilon_2 = 100$	193	0.74	0.78	0.81	8	6	11
$\varepsilon_3 = 10^4$	49	0.72	0.77	0.79	8	7	11
CURVE II	M_{opt}	EFFICIENCY			TIME COST (MS)		
		$c = 1.5$	$c = 2$	$c = 4$	$c = 1.5$	$c = 2$	$c = 4$
$\varepsilon_1 = 10^{-4}$	1986	0.84	0.86	0.91	18	15	20
$\varepsilon_2 = 10^{-2}$	364	0.75	0.75	0.77	21	17	21
$\varepsilon_3 = 1$	72	0.76	0.75	0.80	22	17	22

TABLE IV
PERFORMANCE OF GPS TS BY SED

RESOLUTION 1: $f_{LSSD} = 50$	MOPSI DATASET(744,610 POINTS) AVERAGE $N/M = 8.02$				GEOLIFE DATASET(4,526,030 POINTS) AVERAGE $N/M = 10.1$			
METHOD	RMSE	MAE	MEDE	MAXE	RMSE	MAE	MEDE	MAXE
D-P	4.51	2.38	1.32	39.0	10.7	4.13	1.12	134.1
TD-TR	1.82	1.41	1.23	**4.61**	1.89	1.47	1.28	**4.91**
OW	1.89	1.45	1.23	5.33	1.99	1.53	1.30	5.85
STTrace	4.37	2.67	1.60	21.1	4.93	3.16	2.07	26.0
TS	24.0	11.9	3.64	132.8	42.3	16.6	3.58	363.3
MRPA	**1.61**	**1.23**	**1.05**	5.88	**1.46**	**1.06**	**0.83**	6.51
RESOLUTION 2: $f_{LSSD} = 2000$	MOPSI DATASET AVERAGE $N/M = 25.1$				GEOLIFE DATASET AVERAGE $N/M = 29.5$			
METHOD	RMSE	MAE	MEDE	MAXE	RMSE	MAE	MEDE	MAXE
D-P	13.8	8.39	5.08	81.1	52.3	22.2	6.73	416.6
TD-TR	6.85	5.55	4.82	**17.7**	7.48	6.04	5.24	**19.40**
OW	7.40	5.86	4.89	21.1	8.59	6.80	5.73	25.59
STTrace	33.9	19.9	8.67	132.1	39.3	24.4	14.6	169.2
TS	82.7	48.7	20.9	316.2	200.4	98.6	29.0	1090.0
MRPA	**5.96**	**4.76**	**4.07**	23.9	**5.60**	**4.19**	**3.27**	29.0
RESOLUTION 3: $f_{LSSD} = 10^5$	MOPSI DATASET AVERAGE $N/M = 79.4$				GEOLIFE DATASET AVERAGE $N/M = 109.6$			
METHOD	RMSE	MAE	MEDE	MAXE	RMSE	MAE	MEDE	MAXE
D-P	42.0	29.0	19.9	173.3	154.8	80.4	32.6	867.1
TD-TR	26.7	21.6	18.3	**70.5**	28.9	23.2	19.4	**84.8**
OW	29.5	23.4	19.2	82.1	33.8	26.7	22.1	103.8
STTrace	198.9	131.6	72.4	559.4	251.2	160.9	96.5	871.4
TS	270.1	181.3	106.8	763.5	691.0	399.6	179.7	2733.5
MRPA	**22.9**	**18.5**	**15.8**	79.2	**21.4**	**15.7**	**11.7**	115.6

A. Performance for Artificial Polygonal Curve and Vector Map

For the min-# problem, the performance of polygonal approximation is evaluated by its *efficiency* [26], [27], which is defined as

$$\text{efficiency} = \frac{M_{\text{opt}}}{M}. \tag{7.1}$$

Here M_{opt} is the result of the optimal solution.

In Table II, efficiency and computational cost are evaluated under different error tolerance. It can be observed that the proposed bottom–up multiresolution approach has a lower time cost and its performance is better than that of the two fast heuristic methods, i.e., split [7] and merge [9].

In Table III, we compare the performance when parameter c varies. For larger c, better performance is achieved at higher time cost. We can observe that the least time cost is achieved

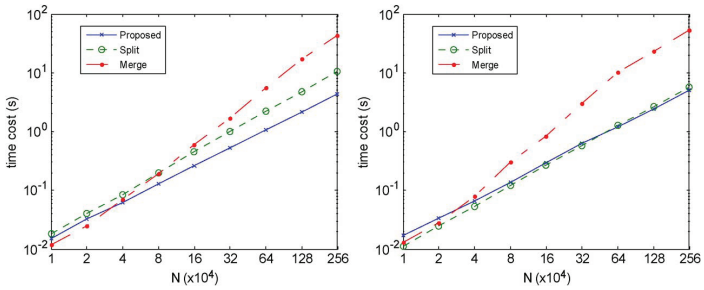

Fig. 14. Processing time cost is plotted for different number of input vertices for curve II. (Left) Low and (right) high error tolerances are both tested.

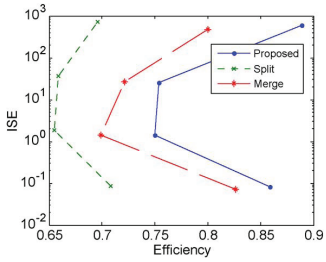

Fig. 15. *Efficiency* and ISE for different error tolerances $\varepsilon = 10^{-4}, 10^{-2}, 1$, and 100.

when $c = 2$, which is in accordance with the theoretical analysis.

In Fig. 14, time cost is also analyzed in comparison with the split and merge algorithms when the size of the input curve N increases. Both the low- and high-error-tolerance cases are tested in the experiment. We can observe that the time cost of the proposed algorithm linearly increases in both cases and it achieves better result than the two comparative heuristic algorithms when the number of input vertices increases.

As the proposed approximation algorithm is a joint optimization for both the min-# approximation using the LISE criterion and the min-ε approximation using the ISE criterion, in Fig. 15, a comparison is made on the ISE and the efficiency of the approximated curve by using different error tolerances. We can observe that the proposed algorithm has achieved both higher efficiency (less number of output vertices) and equal or less ISE compared with the competitive algorithms.

B. Performance Evaluation for GPS TS

The performance of the proposed GPS TS algorithm is tested on two datasets, which are the MOPSI dataset with 344 trajectories and 744 610 points, and the Geolife dataset with 640 trajectories and 4 526 030 points. The root mean square error, the average error, the median error, and the maximum error are all calculated in order to evaluate the efficiency of the proposed algorithm under the SED. In Table V, we also compare these error measures for the GPS trajectories with walking and no-walking segments. We can observe that, although the same LSSD error tolerance is used, walking trajectories can have less distortion with more detailed information comparing with no-walking segments.

TABLE V
PERFORMANCE OF PROPOSED GPS TS ALGORITHM FOR DIFFERENT TRANSPORTATION MODES UNDER SED (IN MOPSI DATASET)

RESOLUTION 1: $f_{LSSD} = 50$,					
	RMSE	MAE	MEDE	MAXE	*N*/*M*
WALKING	1.54	1.20	1.06	5.79	9.38
NO-WALKING	1.71	1.19	0.88	6.29	4.92
RESOLUTION 2: $f_{LSSD} = 2000$					
	RMSE	MAE	MEDE	MAXE	*N*/*M*
WALKING	5.25	4.26	3.68	21.1	32.6
NO-WALKING	8.23	6.27	5.07	33.2	12.9
RESOLUTION 3: $f_{LSSD} = 10^5$					
	RMSE	MAE	MEDE	MAXE	*N*/*M*
WALKING	17.9	14.6	12.5	60.8	119.7
NO-WALKING	34.3	27.4	23.5	128.1	35.9

TABLE VI
TIME COST OF THE TS

	TIME COST (S)	
	MOPSI	GEOLIFE
D-P[7]	1.83	12.9
TD-TR[32]	1.95	13.1
OW[32]	25.4	320.8
STTrace[33]	1160.1	20589
TS[31]	0.85	6.8
Proposed	1.48	10.2

The proposed polygonal approximation algorithm is also compared with other GPS TS algorithms with the same number of approximated points. These competitive algorithms are the *D–P* algorithm [7], *TD-TR* [32], *OW* [32], *STTrace* [33], and *TS* [31]. The results are shown in Table IV, where SED is considered as the error measure. We can observe that the proposed algorithm yields the minimum distortion than other solutions. The time cost of the TS is also summarized in Table VI. It follows from our experiment that the time cost of the proposed algorithm is higher than the TS algorithm [31]. This is because the constant factor in the proposed algorithm is larger than other solutions, which comes from the LISE/LSSD calculation and the graph structure maintenance. For example, based on our experiment, in Fig. 14, when $N > 10\,000$, the proposed solution will have less time cost than the split or merge algorithm. Note that the proposed solution also achieves a better approximation performance than those fast solutions.

An application of the proposed approximation algorithm for the GPS TS is demonstrated in Fig. 16 over a sample route with 575 vertices, where the GPS trajectory is visualized in different

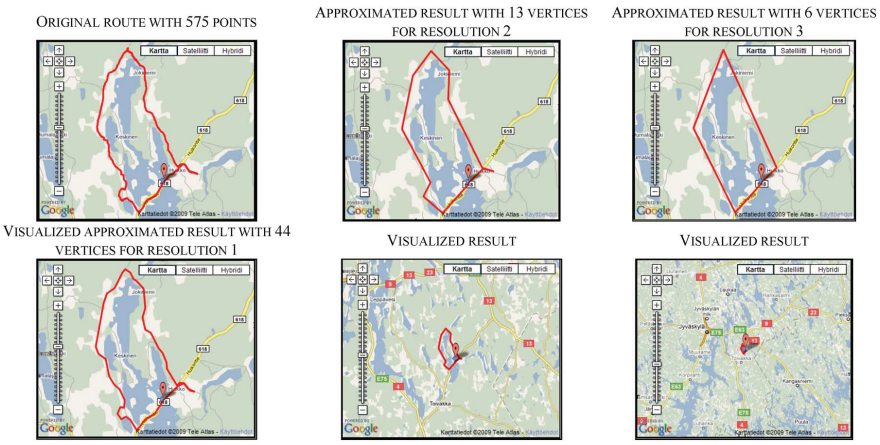

Fig. 16. Example of the GPS TS by the proposed algorithm.

map scales with 44, 13, and 6 vertices correspondingly. As the suitable error tolerance is selected for each resolution, the visualization of the GPS trajectory is not compromised by the reduced data, whereas the rendering time is greatly reduced. The code and the testing dataset can be seen on http://cs.joensuu.fi/sipu/GPSTS.htm.

VIII. Conclusion

We have proposed a fast $O(N)$ time polygonal approximation algorithm for the GPS TS by a joint optimization on both the LSSD and ISSD criteria, which is effective and computationally efficient. The proposed method has been designed by the bottom–up multiresolution approach. In each resolution, a near-optimal polygonal approximation algorithm has been exploited, which has a time complexity of $O(N^2/M)$. Both the theoretical analysis and the experimental tests have demonstrated that the proposed method had made a significant progress in solving the GPS TS problem in a real-time application. Moreover, the proposed polygonal approximation algorithm and fine-tune strategy in Algorithms II and III can be also extended and exploited to other error criteria.

There are several potential extensions of our paper. For example, in our future work, topology properties, road network information, and the similarity of the multiple GPS trajectories can be also considered in the approximation process.

Appendix

Proof of the LSSD in (3.7):

For the sake of the computational efficiency of the SED, we extend the LISE criterion and derive a new error measure, called LSSD, where

$$\delta_{\mathrm{LSSD}}\left(P_i^j\right) = \sum_{i<k<j} \mathrm{SED}^2\left(p_k, p_k'\right)$$

where p_k' is the approximated position at time t_k if subcurve P_i^j is approximated by edge $\overline{p_i p_j}$ [see the definition in (2.3)]. Thus

$$\begin{aligned}
&\delta_{\mathrm{LSSD}}\left(P_i^j\right) \\
&= \sum_{i<k<j}\left(x_i\cdot\frac{t_j-t_k}{t_j-t_i}+x_j\cdot\frac{t_k-t_i}{t_j-t_i}-x_k\right)^2 \\
&\quad+\sum_{i<k<j}\left(y_i\cdot\frac{t_j-t_k}{t_j-t_i}+y_j\cdot\frac{t_k-t_i}{t_j-t_i}-y_k\right)^2 \\
&= \sum_{i<k<j}\left(\frac{x_it_j-x_jt_i}{t_j-t_i}+\frac{x_j-x_i}{t_j-t_i}\cdot t_k-x_k\right)^2 \\
&\quad+\sum_{i<k<j}\left(\frac{y_it_j-y_jt_i}{t_j-t_i}+\frac{y_j-y_i}{t_j-t_i}\cdot t_k-y_k\right)^2 \\
&= c_1^2(j-i-1)+c_2^2\sum_{i<k<j}t_k^2+\sum_{i<k<j}x_k^2 \\
&\quad+2c_1c_2\sum_{i<k<j}t_k-2c_1\sum_{i<k<j}x_k \\
&\quad-2c_2\sum_{i<k<j}t_kx_k+c_3^2(j-i-1) \\
&\quad+c_4^2\sum_{i<k<j}t_k^2+\sum_{i<k<j}y_k^2+2c_3c_4\sum_{i<k<j}t_k \\
&\quad-2c_3\sum_{ikj}y_k-2c_4\sum_{ikj}t_ky_k \\
&= \left(c_1^2+c_3^2\right)(j-i-1)+\left(c_2^2+c_4^2\right)\left(S_{t^2}^{j-1}-S_{t^2}^i\right) \\
&\quad+2(c_1c_2+c_3c_4)\left(S_t^{j-1}-S_t^i\right) \\
&\quad+\left(S_{x^2}^{j-1}-S_{x^2}^i\right)+\left(S_{y^2}^{j-1}-S_{y^2}^i\right) \\
&\quad-2c_1\left(S_x^{j-1}-S_x^i\right)-2c_3\left(S_y^{j-1}-S_y^i\right) \\
&\quad-2c_2\left(S_{xt}^{j-1}-S_{xt}^i\right)-2c_4\left(S_{yt}^{j-1}-S_{yt}^i\right)
\end{aligned}$$

where

$$c_1 = \frac{x_i t_j - x_j t_i}{t_j - t_i} \quad c_2 = \frac{x_j - x_i}{t_j - t_i}$$
$$c_3 = \frac{y_i t_j - y_j t_i}{t_j - t_i} \quad c_4 = \frac{y_j - y_i}{t_j - t_i}$$

where S_x, S_y, S_t, S_{x2}, S_{y2}, S_{t2}, S_{tx}, and S_{ty} are the accumulated sums of x, y, and t on the GPS trajectory, respectively.

The computation of the aforementioned approximation error $\delta_{\mathrm{LSSD}}(P_i^j)$ takes $O(1)$ time with an $O(N)$ time accumulated sum precalculation.

Proof of Theorem 3:

Suppose that, under an error tolerance ε, curve P with N vertices can be approximated by curve P' with M vertices. The number of vertices with k links is $n_k, k = 0, 1, \ldots, M-1$. In total, the $2N$ space is needed to record the accumulated errors and the backtracking vector; thus, it has a space complexity $O(N)$.

As every node is only visited once in the tree traversal step with $O(N)$ in total, the main bottleneck is the cost on edge tests, which can be calculated as follows:

$$f = \sum_{i=1}^{M-1} n_{i-1} \cdot n_i \quad \text{s.t.} \quad \sum_{i=0}^{M-1} n_i = N, n_0 = 1$$
$$n_{M-1} = 1 \quad n_i \geq 1 \quad i = 0, \ldots, M-1.$$

Suppose that M vertices are first selected with the number of links from 0 to $M-1$, respectively. For the remaining $N-M$ vertices, if the number of links of every vertices is randomly distributed under a *multinominal distribution*, then we have

$$(u_1, u_2 \ldots, u_{M-2}) \sim \text{Mult}\left(U; \left(\frac{1}{M-2}, \frac{1}{M-2}, \ldots, \frac{1}{M-2}\right)\right)$$

where $u_i = n_i - 1, i = 1, 2, \ldots, M-2$ and the corresponding statistical properties of u_i $(i = 1, 2, \ldots, M-2)$ can be formulated as follows:

$$\begin{aligned}
E(u_i) &= \frac{N-M}{M-2} \\
\text{var}(u_i) &= (N-M) \cdot \frac{1}{M-2} \cdot \left(1 - \frac{1}{M-2}\right) \\
\text{cov}(u_i, u_j) &= E(u_i u_j) - E(u_i)E(u_j) \\
&= -(N-M) \cdot \frac{1}{M-2} \cdot \frac{1}{M-2} \\
E(u_i u_j) &= \text{cov}(u_i, u_j) + E(u_i)E(u_j) \\
&= \frac{-N + M + N^2 + M^2 - 2NM}{(M-2)^2} \\
&= O(N^2/M^2) \\
E\left(u_i^2\right) &= E^2(u_i) + \text{var}(u_i) \\
&= \frac{(N-M)^2 + (N-M) \cdot (M-3)}{(M-2)^2} \\
&= O(N^2/M^2).
\end{aligned}$$

Thus, the expected time complexity, i.e.,

$$\begin{aligned}
E(f) &= n_1 + n_{m-2} + (M-3)E(n_1 n_2) \\
&= 1 + E(u_1) + 1 + E(u_{M-2}) \\
&\quad + (M-3)E\left((1+u_1)(1+u_2)\right) \\
&= 2 + \frac{N-M}{M-2} \cdot 2 + (M-3)\left(1 + 2 \cdot E(u_1) + E(u_1 u_2)\right) \\
&= 2 + \frac{N-M}{M-2} \cdot 2 + (M-3) \cdot \frac{N^2 - 5N + M + 4}{(M-2)^2} \\
&= O(N^2/M).
\end{aligned}$$

To sum up, the expected time complexity is $O(N^2/M)$ and space complexity $O(N)$. □

Proof of Theorem 1:

As the output of the *min-# initialization* is a tree structure, $2N$ space is needed in order to record all the parent and child nodes on the tree, and its space complexity is $O(N)$.

The time complexity of the min-# initialization mainly consists of two parts, i.e., the number of edge tests and the maintenance cost of two priority queues. The cost of edge tests can be calculated in a similar manner as in Theorem 3, i.e.,

$$f = \sum_{i=0}^{M-2} n_i \cdot \left(\frac{2 \cdot \left(\sum_{j=0}^{i} n_j\right)}{(i+1)}\right)$$
$$\text{s.t.} \quad \sum_{i=0}^{M-1} n_i = N, n_0 = 1, n_i \geq 1, i = 0, \ldots, M-1$$
$$(u_1, u_2 \ldots, u_{M-1}) \sim \text{Mult}\left(U; \left(\frac{1}{M-1}, \frac{1}{M-1}, \ldots, \frac{1}{M-1}\right)\right)$$

where $u_i = n_i - 1$, $i = 1, 2, \ldots, M-1$, and

$$E(f) = 2(M-1) \cdot \left[\frac{1}{i+1} E\left(n_i^2\right) + \frac{i}{i+1} E(n_j n_j)\right].$$

From Theorem 3, we have

$$E(n_i n_j) = O(N^2/M^2), E\left(n_i^2\right) = O(N^2/M^2).$$

Thus, $E(f) = O(N^2/M)$, as

$$\begin{aligned}
E(n_i) &= 1 + E(u_i) \\
&= 1 + \frac{N-M}{M-2}, \qquad i = 1, \ldots, M-2.
\end{aligned}$$

The cost of maintaining the priority queues is

$$E(g) = E\left(\sum_{i=0}^{M-1} n_i \cdot \log_2(n_i)\right) = 1 + \sum_{i=1}^{M-1} E\left(n_i \cdot \ln(n_i)\right) / \ln 2.$$

Suppose a linear function is constructed as follows:

$$y_i = \ln\left(E(n_i)\right) + \frac{1}{E(n_i)} \cdot \left(n_i - E(n_i)\right).$$

The constructing function has property $\ln(n_i) \le y_i$, and thus

$$\begin{aligned} E(g) &\le 1+\frac{1}{\ln 2}\sum_{i=1}^{M-1} E\left(n_i\cdot(\ln(E(n_i))+\frac{1}{E(n_i)}\cdot(n_i-E(n_i))\right), \\ &\qquad i=1,\ldots,M-2 \\ &\le 1+\underbrace{\frac{1}{\ln 2}\left(-1+\ln\frac{N-1}{M-1}\right)\cdot(N-1)}_{O(N\log(N/M))} \\ &\quad +\underbrace{\frac{1}{\ln 2}\frac{(M-1)^2}{N-1}E\left(n_2^2\right)}_{O(N)} \\ E(g) &= O\left(N\log(N/M)\right). \end{aligned}$$

Thus, the min-# initialization has an expected time complexity of $O(N^2/M)$ and a space complexity of $O(N)$. □

Proof of Theorem 2:

First, we give the proof of the time complexity for simplified RSDP method. Suppose the initial approximated curve $P' = (p_{i_1}, p_{i_2}, \ldots, p_{i_M})$, where $i_1, i_2, \ldots, i_M$ are the indexes on the curve, s.t.

$$n_k = i_{k+1} - i_k, k = 1, \ldots, M-1.$$

The number of edges tests of RSDP is

$$\begin{aligned} f &= \sum_{i=1}^{M-1}\left(\left(1+\sum_{j=i-W/2}^{i+W/2-1} n_j\right)\cdot\left(1+\sum_{j=i-W/2+1}^{i+W/2} n_j\right)\right) \\ &\text{s.t.} \quad \sum_{i=1}^{M-1} n_i = N-1, n_i \ge 1, i = 1, \ldots, M-1. \end{aligned}$$

Let us define $u_i = n_i - 1$, $i = 1, 2, \ldots, M-1$, and assume that curve P' is randomly initialized as in Theorem 3 such that u_i has the following property:

$$\begin{aligned} &(u_1, u_2 \ldots, u_{M-1}) \\ &\qquad \sim \text{Mult}\left(U; \left(\frac{1}{M-1}, \frac{1}{M-1}, \ldots, \frac{1}{M-1}\right)\right). \end{aligned}$$

The expected time complexity is therefore estimated as

$$\begin{aligned} E(f) &= (M-1)\cdot\left(\left(1+\sum_{j=i-W/2}^{i+W/2-1} n_j\right)\right. \\ &\qquad \left.\cdot\left(1+\sum_{j=i-W/2+1}^{i+W/2} n_j\right)\right) \\ &= (M-1)\cdot\left(1+\sum_{j=i-W/2}^{i+W/2-1} n_j + \sum_{j=i-W/2+1}^{i+W/2} n_j\right. \\ &\qquad \left.+\sum_{j=i-W/2}^{i+W/2-1} n_j \cdot \sum_{j=i-W/2+1}^{i+W/2} n_j\right) \\ &= (M-1)\cdot\left(\left(1+2\cdot W\cdot E(n_j)+(W-1)\cdot E\left(n_j^2\right)\right.\right. \\ &\qquad \left.\left.+(W^2-W+1)\cdot E(n_i n_j)\right)\right. \end{aligned}$$

According to Theorem 3, we have

$$\begin{aligned} E(n_j) &= O(N/M) \\ E(n_i^2) &= O(N^2/M^2) \\ E(n_i n_j) &= O(N^2/M^2). \end{aligned}$$

Thus, $E(f) = O(W^2N^2/M^2)$. □

On the other hand, the proposed reduced search method is achieved by lifting the vertex position in the output tree structure in the initialization. The memory cost of maintaining a tree structure is $O(N)$. Likewise, the cost of number of edges tests is calculated as

$$\begin{aligned} f &= \sum_{i=0}^{M-3} n_i \cdot \left(\sum_{j=i+2}^{i+W+1} n_j\right) \\ E(f) &= (M-2)W\cdot E(n_i n_j) = O(WN^2/M). \end{aligned}$$

As $E(n_i n_j) = O(N^2/M)$, we have $E(f) = O(WN^2/M)$ Thus, it has an expected time complexity of $O(WN^2/M)$ and a space complexity of $O(N)$. □

Proof of Theorem 4:

From Theorems 1–3, the space complexity of the near-optimal polygonal approximation algorithm is $O(N)$. An additional cost is the precalculated sums, which also takes the $O(N)$ space. As we do not need to record all the information of the intermediate scales, the total space complexity is $O(N)$.

The time complexity of the proposed bottom–up multiresolution algorithm mainly consists of three parts, i.e., the error tolerance initialization (step 1), the initial curve approximation (step 2), and the final approximation (step 3). As the approximation error between two vertices can be calculated in constant time, the time cost of step 1 can be calculated as follows:

$$\begin{aligned} \sum_{k=1}^{\log_c N}\frac{N}{c^k} &= \frac{\frac{N}{c}\left(1-\left(\frac{1}{c}\right)^{\log_c N}\right)}{1-\frac{1}{c}} \\ &= \frac{\frac{N}{c}\left(1-\frac{1}{N}\right)}{1-\frac{1}{c}} = \frac{N-1}{c-1} = O(N). \end{aligned}$$

In step 2, the time complexity of the proposed polygonal approximation method is $O(N_k^2/M_k)$. As the number of input and output vertices obeys equation $M_k = N_k/c$ for each resolution, the time complexity can be estimated by

$$\begin{aligned} \sum_{k=0}^{\log_c N-1}\frac{(N/c^k)^2}{N/c^{k+1}} &= \sum_{i=0}^{\log_c N}\left(\frac{N}{c^{k-1}}\right) = N\cdot\frac{c\left(1-\frac{1}{c^{\log_c N}}\right)}{1-\frac{1}{c}} \\ &= (N-1)\cdot\frac{c^2}{c-1} = O(N). \end{aligned}$$

Since the proposed polygonal approximation algorithm (Algorithms I–III) has time complexity of $O(N_k^2/M_k)$, the computational cost of step 3 can be written as $O(cN_k)$, where the value of the parameter is always $c > 1$.

To sum up, the proposed multiresolution polygonal approximation has a time complexity of $O(N)$. □

ACKNOWLEDGMENT

The authors would like to thank the reviewers and the editor for their valuable comments and suggestions, which have been very useful in improving the technical content and the presentation of this paper. The authors would also like to thank Prof. J. Alho and Dr. V. Hautamäki for the useful discussion during this paper.

REFERENCES

[1] H. Imai and M. Iri, "Polygonal approximations of a curve-formulations and algorithms," in *Computational Morphology*. Amsterdam, The Netherlands: North Holland, 1988, pp. 71–86.

[2] G. T. Toussaint, "On the complexity of approximating polygonal curves in the plane," in *Proc. IASTED*, Lugano, Switzerland, 1985, pp. 59–62.

[3] A. Melkman and J. O'Rourke, "On polygonal chain approximation," in *Computational Morphology*. Amsterdam, The Netherlands: North Holland, 1988, pp. 87–95.

[4] J. C. Perez and E. Vidal, "Optimum polygonal approximation of digitized curves," *Pattern Recognit. Lett.*, vol. 15, no. 8, pp. 743–750, Aug. 1994.

[5] K.-L. Chung, W.-M. Yan, and W.-Y. Chen, "Efficient algorithms for 3-D polygonal approximation based on LISE criterion," *Pattern Recognit.*, vol. 35, no. 11, pp. 2539–2548, Nov. 2002.

[6] B. K. Ray and K. S. Ray, "A non-parametric sequential method for polygonal approximation of digital curves," *Pattern Recognit. Lett.*, vol. 15, no. 2, pp. 161–167, Feb. 1994.

[7] D. H. Douglas and T. K. Peucker, "Algorithm for the reduction of the number of points required to represent a line or its caricature," *Can. Cartographer*, vol. 10, no. 2, pp. 112–122, 1973.

[8] J. Hershberger and J. Snoeyink, "Cartographic line simplification and polygon CSG formulae in $O(n \log * n)$ time," in *Proc. 5th Int. Workshop Algorithms Data Struct.*, 1997, pp. 93–103.

[9] A. Pikaz and I. Dinstein, "An algorithm for polygonal approximation based on iterative point elimination," *Pattern Recognit. Lett.*, vol. 16, no. 6, pp. 557–563, Jun. 1995.

[10] W. S. Chan and F. Chin, "On approximation of polygonal curves with minimum number of line segments or minimum error," in *Proc. ISAAC*, 1992, vol. 650, Lecture Notes in Computer Science, pp. 378–387.

[11] D. Chen and O. Daescu, "Space-efficient algorithms for approximating polygonal curves in two dimensional space," in *Proc. Comput. Combinatorics*, 1998, vol. 1449, pp. 45–55.

[12] P. K. Agarwal and K. R. Varadarajan, "Efficient algorithms for approximating polygonal chains," in *Proc. Discrete Comput. Geom.*, 2000, vol. 23, pp. 273–291.

[13] M. Salotti, "Optimal polygonal approximation of digitized curves using the sum of square deviations criterion," *Pattern Recognit.*, vol. 35, no. 2, pp. 435–443, Feb. 2002.

[14] D. Eu and G. T. Toussaint, "On approximation polygonal curves in two and three dimensions," *Graph. Models Image Process.*, vol. 56, no. 3, pp. 231–246, May 1994.

[15] O. Daescu and N. Mi, "Polygonal chain approximation: A query based approach," *Comput. Geom.*, vol. 30, no. 1, pp. 41–58, Jan. 2005.

[16] K. L. Chung, P. H. Liao, and J. M. Chang, "Novel efficient two-pass algorithm for closed polygonal approximation based on LISE and curvature constraint criteria," *J. Vis. Commun. Image Represent.*, vol. 19, no. 4, pp. 219–230, May 2008.

[17] A. Kolesnikov and P. Fränti, "A fast near-optimal min-# polygonal approximation of digitized curves," in *Proc. ACIT*, 2002, pp. 418–422.

[18] A. Kolesnikov and P. Fränti, "Reduced-search dynamic programming for approximation of polygonal curves," *Pattern Recognit. Lett.*, vol. 24, no. 14, pp. 2243–2254, Oct. 2003.

[19] B. P. Buttenfield, "Transmitting vector geospatial data across the Internet," in *Proc. GIScience—LNCS*, 2002, vol. 2478, pp. 51–64.

[20] C. Le Buhan Jordan, T. Ebrahimi, and M. Kunt, "Progressive content-based shape compression for retrieval of binary images," *Comput. Vis. Image Understand.*, vol. 71, no. 2, pp. 198–212, Aug. 1998.

[21] A. Kolesnikov, P. Fränti, and X. Wu, "Multiresolution polygonal approximation of digital curves," in *Proc. 17th ICPR*, 2004, vol. 2, pp. 855–858.

[22] P. F. Marteau and G. Ménier, "Speeding up simplification of polygonal curves using nested approximations," *Pattern Anal. Appl.*, vol. 12, no. 4, pp. 367–375, Oct. 2009.

[23] A. Kolesnikov, "Fast algorithm for ISE-bounded polygonal approximation," in *Proc. IEEE Int. Conf. Image Process.*, 2008, pp. 1013–1016.

[24] G. Papakonstantinou, P. Tsanakas, and G. Manis, "Parallel approaches to piecewise linear approximation," *Signal Process.*, vol. 37, no. 3, pp. 415–423, Jun. 1994.

[25] M. Salotti, "An efficient algorithm for the optimal polygonal approximation of digitized curves," *Pattern Recognit. Lett.*, vol. 22, no. 2, pp. 215–221, Feb. 2001.

[26] P. L. Rosin, "Techniques for assessing polygonal approximations of curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 6, pp. 659–666, Jun. 1997.

[27] P. L. Rosin, "Assessing the behavior of polygonal approximation algorithms," *Pattern Recognit.*, vol. 36, no. 2, pp. 505–518, Feb. 2003.

[28] O. Daescu, N. Mi, C. S. Shin, and A. Wolff, "Farthest-point queries with geometric and combinatorial constraints," *Comput. Geom.*, vol. 33, no. 3, pp. 174–185, Feb. 2006.

[29] O. Daescu, "New results on path approximation," *Algorithmica*, vol. 38, no. 1, pp. 131–143, Oct. 2003.

[30] A. Gribov and E. Bodansky, "A new method of polyline approximation," in *Proc. Int. Conf. Struct., Syntactic Pattern Recognit.—LNCS*, 2004, vol. 3138, pp. 504–511.

[31] Y. Chen, K. Jiang, Y. Zheng, C. Li, and N. Yu, "Trajectory simplification method for location-based social networking services," in *Proc. ACM GIS Workshop Location-Based Social Netw. Serv.*, 2009, pp. 33–40.

[32] N. Meratnia and R. A. de By, "Spatiotemporal compression techniques for moving point objects," in *Proc. Extending Database Technol.*, 2004, pp. 561–562.

[33] M. Potamias, K. Patroumpas, and T. Sellis, "Sampling trajectory streams with spatiotemporal criteria," in *Proc. SSDBM*, 2006, pp. 275–284.

[34] Z. Yu and X. Zhou, *Computing With Spatial Trajectories*. New York: Springer-Verlag, 2011.

[35] H. Cao, O. Wolfson, and G. Trajcevski, "Spatio-temporal data reduction with deterministic error bounds," *VLDB J.*, vol. 15, no. 3, pp. 211–228, Sep. 2006.

[36] J. Muckell, J. H. Hwang, C. T. Lawson, and S. S. Ravi, "Algorithms for compressing GPS trajectory data: An empirical evaluation," in *Proc. SIGSPATIAL Int. Conf. Adv. Geograph. Inform. Syst.*, 2010, pp. 402–405.

[37] J. Muckell, J. H. Hwang, V. Patil, C. T. Lawson, F. Ping, and S. S. Ravi, "SQUISH: An online approach for GPS trajectory compression," in *Proc. Int. Conf. Comput. Geospatial Res. Appl.*, 2011, pp. 1–8.

[38] R. Lange, T. Farrel, F. Dürr, and K. Rothermel, "Remote real-time trajectory simplification," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2009, pp. 1–10.

[39] H. Alt and L. J. Guibas, *Handbook of Computational Geometry*. Amsterdam, The Netherlands: North Holland, 1999, pp. 121–153.

[40] H. Alt, C. Knauer, and C. Wenk, "Matching polygonal curves with respect to the Fréchet distance," in *Proc. STACS—LNCS*, 2001, pp. 63–74.

[41] M. Chen, M. Xu, and P. Fränti, "Compression of GPS trajectories," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, 2012, pp. 62–71.

[42] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-26, no. 1, pp. 43–49, Feb. 1978.

[43] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *Proc. VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, Aug. 2008.

[44] R. Estkowski and J. Mitchell, "Simplifying a polygonal subdivision while keeping it simple," in *Proc. Symp. Comput. Geom.*, 2001, pp. 40–49.

Minjie Chen (S'10) received the B.Sc. and M.Sc. degrees in biomedical engineering from Shanghai Jiaotong University, Shanghai, China, in 2003 and 2007, respectively. Since 2008, he is currently working toward the Ph.D. degree in computer science at the University of Eastern Finland, Joensuu, Finland.

His research interests include image denoising and compression, spatial-temporal data compression, and medial image analysis.

Mantao Xu received the B.Sc. degree in mathematics from Nankai University, Tianjin, China, in 1991, the M.Sc. degree in applied mathematics from Harbin Institute of Technology, Harbin, China, in 1997, and the Ph.D. degree in computer science from the University of Joensuu, Joensuu, Finland, in 2005 respectively.

From 2005 to 2010, he was a Research Laboratory Manager with Kodak Health Group and Carestream Health Inc., Global Research and Development Center, Shanghai, China. He is currently a Research Professor with the School of Electric Engineering, Shanghai Dianji University, Shanghai. His research interests include medical image analysis, multimedia technology, and pattern recognition.

Pasi Fränti (SM'08) received the M.Sc. and Ph.D. degrees in science from the University of Turku, Turku, Finland, 1991 and 1994, respectively.

Since 2000, he has been a Professor of computer science with the University of Eastern Finland, Joensuu, Finland. He has published 57 journals and 130 peer-review conference papers, including ten IEEE transaction papers. He has supervised 15 Ph.D. students and is currently the head of the East Finland doctoral program in Computer Science and Engineering. His research interests include clustering algorithms, vector quantization, lossless image compression, voice biometrics, and location-based systems.

Dr. Fränti serves as an associate editor for Pattern Recognition Letters.

Paper P5

M. Chen, M. Xu, P. Fränti, "Fast dynamic quantization algorithm for vector map compression", *IEEE Int. Conf. on Image Processing* (ICIP'10), Hong Kong, China, 4289-4292, 2010.

FAST DYNAMIC QUANTIZATION ALGORITHM FOR VECTOR MAP COMPRESSION

Minjie Chen[1], Mantao Xu[2], Pasi Fränti[1]

[1]School of Computing, University of Eastern Finland
[2]School of Electrical Engineering, Shanghai Dianji University, China

ABSTRACT

Vector map compression can be solved by incorporating both data reduction (polygonal approximation) and quantization of the prediction errors, which is the so-called dynamic quantization. This straightforward solution is to calculate all the rate-distortion curves with respect to each of the quantization levels such that the best quantizer is the lower envelope of the set of curves. But computing an entire set of rate-distortion curves is computationally expensive. To solve this problem, we propose a fast algorithm first estimates an optimal Lagrangian parameter λ for each given quantization level *l* and thus only one rate-distortion curve is achievable for constructing the optimal quantizer of prediction errors. An experimental result demonstrates that proposed algorithm reduces the computational complexity significantly without compromising its rate-distortion performance.

Index Terms— Data compression, Computational geometry

1. INTRODUCTION

Vector maps embrace a number of geographic information or objects such as waypoints, routes and areas. Those geographic objects can be represented with a sequence of points in a given coordinate system. However, encoding and achieving the geographic objects in a map image may require expensive data storage and processing time. In order to reduce this computational cost, a variety of algorithms has been studied and developed [1-8]. Existing algorithms have been explored via two classes of strategies: *polygonal approximation* and *quantization*.

The main advantage of polygonal approximations is the high compression rates, which can be achievable either by the fast heuristic methods in [9, 10] or by the graph-based methods in [11, 12]. The number of points in the vector map is reduced by *polygonal approximations* such that the polygonal curve can be represented in a coarser resolution. But they quite often incur a high image distortion. On the other hand, the quantization-based approaches calculate the differential coordinates of adjacent data points as the prediction error and then the residual vectors are quantized using different quantization strategies including *product uniform quantization* [2], *product scalar quantization* [3], and *vector quantization* with fixed-size codebook [4]. However, they often lead to less distortion error with the limited compression gains. A pioneer solution is to combine both the advantage of polygonal approximations and prediction error quantization to achieve the best rate-distortion performance. For instance, the previous *reference line* method [5] first identified a series of references lines by using polygonal approximation, prediction errors are then estimated for the remaining points according to their nearest reference lines followed by *product scalar quantization* in a similar manner to [3]. Likewise in [8], a number of data points were first reduced by *Visvalingam-Whyatt* algorithm, to preserve a consistent topology, and then were quantized and encoded by a clustering-based method.

Motivated by the previous progress made by polygonal approximation and polygonal quantization, a so-called *dynamic quantization* (DQ) in [6] was sincerely investigated. The *dynamic quantization* algorithm performs a joint optimization using both polygonal approximation and vector quantization via *dynamic programming*. For a given quantization level *l*, a naive *product uniform quantization* is employed in the joint optimization using a *Lagrangian* parameter λ. Traversing different *Lagrangian* parameter λ will construct a rate-distortion curve that depends on the quantization level *l*. The optimal solution is selected according to the *lower envelope* of these curves. However, a main challenge for the joint optimization is its expensive computational cost. To overcome this difficulty, the *error balance principle* was proposed in [7] based on a strict assumption that the total quantization error equals to the error for *polygonal approximation* without quantization. For a given quantization level *l*, an optimal number of points *M* can be identified in the *min-ε polygonal approximation* problem by using binary search. However, in practice, the time complexity for *min-ε polygonal approximation*s equals to $O(N^2)$ [12] as well, which is still intractable for real-time application.

In this work, we have proposed a fast algorithm for vector map compression. For a given quantization level *l*, an optimal *Lagrangian* parameter λ was first estimated and the vector map is compressed by solving a shortest path problem in a directed acyclic graph with cost function $J = E + \lambda R$, where *E* is the distortion for the approximation curve and R is the coding cost. Moreover, the algorithm is further

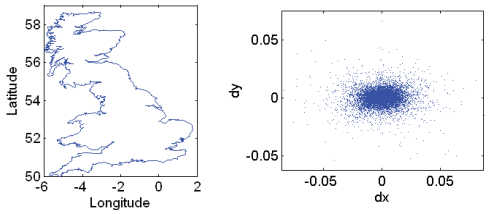

Fig. 1, *Test map Britain with 10,910 points* (*left*)*, and corresponding prediction residuals* (*right*).

improved by using a specific search criterion. The structure of this presentation is organized as follows: section 2 describes the improved *dynamic quantization* algorithm; the experimental results are reported in section 3 and finally a conclusion is drawn in section 4.

2. PROPOSED METHOD

2.1 Prediction and encoding

Vector map compression can be formulated as a data compression problem for a 2-dimensional vector sequence $P = (p_1, p_2, \ldots, p_n)$. A common practice for data compression encompasses the three essential procedures: *prediction*, *quantization* of the residual vectors and *entropy coding*. The prediction procedure calculates the differential coordinates of adjacent points as a prediction error instead of using the absolute coordinates for data quantization. It can be assumed that the prediction errors obey a distribution of random variable empirically, e.g., uniform distribution or geometric distribution. An example of polygonal curve can be observed in fig. 1, where the resulting differential coordinates obeys a geometric distribution.

To avoid quantization error propagations, the prediction must be done in a way of closed-loop prediction:

$$p_i^r = Q(\mathbf{v}_i) + p_{i-1}^r \tag{1}$$

where Q is a two-dimensional *product uniform quantizer* $\mathbf{v}_i$ is the residual vector and p_{i-1}^r is the estimation of the previous point. For a given quantization level l, the *product uniform quantizer* is formulated as:

$$Q(\mathbf{v}_i) = [\mathbf{v}_i / l] \cdot l = ([\Delta x_i / l] \cdot l, [\Delta y_i / l] \cdot l) \tag{2}$$

Obviously, coding Q $(\mathbf{v}_i)$ is equivalent to coding an integer vector $\mathbf{q} = ([\Delta x_i/l], [\Delta y_i/l])$, which can be encoded by probability distributions of q_x and q_y.

$$r(\mathbf{v}_i) = -\log_2 f(q_{xi}) - \log_2 f(q_{yi}) \tag{3}$$

where the codebook itself must be also encoded and transmitted to the decoder. But a large-sized codebook is intractable in order to achieve a desirable coding efficiency. An intuitive solution is to adopt a single-parameter *geometric distribution* to model $|q_x|$ and $|q_y|$:

$$f(|q_x|) = (1 - p_x)^{|q_x|} p_x \tag{4}$$

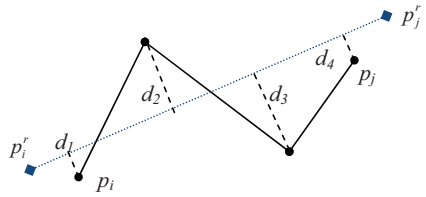

Fig. 2, *Poly-line* $\{p_i, \ldots, p_j\}$ *(solid line) is approximated by* $\{p_i^r, p_j^r\}$ *(dot line)with approximating error* $e_2(p_i^r, p_j^r) = d_1^2 + d_2^2 + d_3^2 + d_4^2$

where p_x can be approximated by using *maximum likelihood estimation*. Thus, the code length led by an arithmetic coding according to the *geometric distribution* is written as

$$\begin{aligned} r(\mathbf{v}_i) = &-(|q_{xi}| \log_2(1 - p_x) + \log_2(p_x)) + 2 \\ &-(|q_{yi}| \log_2(1 - p_y) + \log_2(p_y)) \end{aligned} \tag{5}$$

where no codebook is needed.

We should mention when the data has different property, *uniform distribution*, *negative binomial distribution* or *Poisson distribution* can also be considered instead of *geometric distribution*.

2.2 Dynamic quantization

In *dynamic quantization, polygonal approximation* is embedded into the closed-loop framework by using dynamic programming. Suppose that a poly-line $\{p_i, \ldots, p_j\}$ is approximated by line segment $\{p_i^r, p_j^r\}$, the approximation error can be defined as the sum of square distances from vertices p_k ($i \le k \le j$) to $\{p_i^r, p_j^r\}$ in fig. 2:

$$e_2(p_i^r, p_j^r) = \sum_{k=i}^{j} d^2(p_k, \{p_i^r, p_j^r\}) \tag{6}$$

This approximation error in (6) can be calculated in a constant time [11] by pre-computing the accumulated sum for curve coordinates x^2, x, xy, y^2 and y. The *dynamic quantization* becomes a joint optimization of polygonal approximation and prediction error quantization, which minimizes the cost function:

$$J = E_2 + \lambda R = \sum_{m=1}^{M} (e_2(p_{i_m}^r, p_{i_{m+1}}^r) + \lambda \cdot r(p_{i_m}^r, p_{i_{m+1}}^r)) \tag{7}$$

where M is the number of points output by polygonal approximation. The minimization problem can be solved by the shortest path search on a weighted directed acyclic graph (DAG) or *dynamic programming*. Suppose J_i is the minimum weighting sum from p_1 to p_i on G, A is an array used for backtracking operation, the recursive equation can be defined by:

$$J_i = \min_{\{1 \le k \le i-1\}} (J_k + e_2(p_k^r, p_i^r) + \lambda r(p_k^r, p_i^r)), J_1 = 0 \tag{8}$$

$$A_i = \arg\min_{\{1 \le k \le i-1\}} (J_k + e_2(p_k^r, p_i^r) + \lambda r(p_k^r, p_i^r)) \tag{9}$$

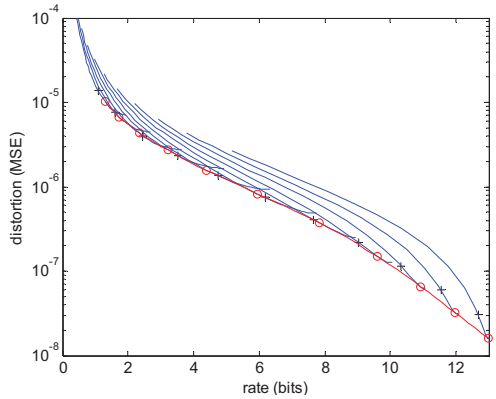

Fig. 3, *Rate-distortion curve for quantization step $q_k=0.01/2^k$, where k=0, 1/2,1,..., 5 (from left to right), black '+' is the position when error balance principle is applied, red 'o' is the proposed(λ is selected by (10)).The red line is the rate-distortion curve when optimal λ is selected with $q_k=0.01/2^k$, where k=0, 1/5, 2/5,..., 5.*

Existing approach intuitively calculates all the rate-distortion curves with respect to each of the quantization levels such that the best quantizer is the *lower envelope* of the set of curves. This method computes an entire set of rate-distortion curves which is hugely time-expensive.

To resolve this operational problem, a *fast dynamic quantization (FDQ) algorithm* is proposed. We have proved, for each quantization level l, one optimal *Lagrangian* parameter λ can be estimated as (see appendix):

$$\lambda \approx \frac{1}{6} l^2 \ln 2 \qquad (10)$$

Eventually, only one dynamic quantization needs to be conducted for a given quantization level l. However, by traversing different quantization level l, a unique rate-distortion curve can be constructed. An example can be found in Fig. 3. In the figure, it was also illustrated that the pervious methods have investigated different quantization levels by considering 30-40 number of λs for each l, which leads to 300 iterations of minimization of (7).

2.3 Search criterion

The shortest path algorithm on a weighted DAG takes $O(N^2)$ time. This can further be improved by incorporating a specific search criterion:

$$\sqrt{\frac{e_2(p_k^r, p_i^r)}{(i-k)}} > \tau \sqrt{\frac{e_2(p_{A_i}^r, p_i^r)}{(i-A_i)}} \qquad (11)$$

where $\{ p_{A_i}^r , p_i^r \}$ has a shortest path so far and p_k^r is the current testing point. Namely, for a target point p_i^r, the shorted path search will terminate weight calculation before point p_k once if equation (11) is satisfied. The experimental testing in this presentation has revealed that the processing time can be reduced by more than 95% with τ =40. Pseudo code of proposed algorithm is shown in Fig. 4.

The proposed method can also be applied for entropy-constrained problem, in which we compress the vector map data under a certain bit-rate. The result can then be obtained by several iterations of the algorithm using bisection search on the quantization level l.

```
INPUT   l                          ← quantization step
        P= (p1, p2,..., pn)        ← 2-D vector sequence
OUTPUT: P' = ( p1r , p2r ..., pnr ) ← approximation curve
        F                          ← compression file
FUNCTION VectorMapEncoding (P, l) RETURN P' ,F
N       ←   Number of point for curve P;
J       ←   array with N elements, J(1)=0, J(2:N)=∞
λ       ←   1/6*l^2*ln2
p1r     ←   p1
A       ←   backtracking array with N elements
FOR i = 1 TO N
    FOR k = i-1 TO 1
        ptmp← pkr + Q(pi - pkr )
        Jtmp ← Jk + e2(pkr , ptmp) + λr(pkr , ptmp)
        IF Jtmp < Ji
                Ji   ← Jtmp
                Ai   ← k
                pir  ← ptmp
        ELSEIF (11) met
                BREAK; // Stop searching criterion
        END IF
    END FOR
END FOR
P'      ←   Backtracking on A and do sub-sampling on P'
F       ←   Encoding on P'  using arithmetic coding by (5)
```

Fig. 4 *Pseudo code of fast dynamic quantization method*

3. EXPERIMENTS

The proposed *fast dynamic quantization* algorithm (FDQ) is evaluated on a 10,910-point vector map representing the contour of Britain (Fig. 1). We compare the performance with the previous *dynamic quantization* (DQ) algorithm [7], and with several other approaches as well: *clustering-based method* (CBC) [4], and *reference line method* (RL) [5]. The distortion is measured here by mean squared error (MSE). The corresponding rate-distortion curves are plotted in Fig. 5. It can be observed that the proposed algorithm achieves significantly better rate-distortion result than the other approaches in this work considered and can be also comparable with the DQ algorithm. The computational cost for solving the entropy-constrained problem can be reduced within 1 second by using the proposed dynamic quantization. In the experiments, the proposed algorithm only takes 5% of the time as the previous approaches do. The proposed algorithm is also applicable to variable-resolution compression problem for a real-time application. The visualization performance in the decoder for different compression bit-rate can be found in fig. 6.

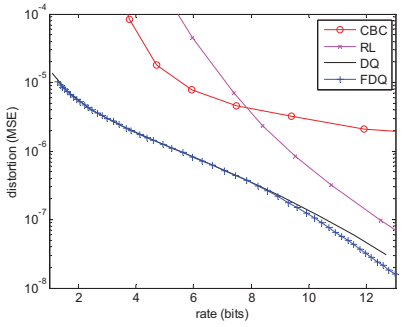

Fig. 5, *Performance comparison*

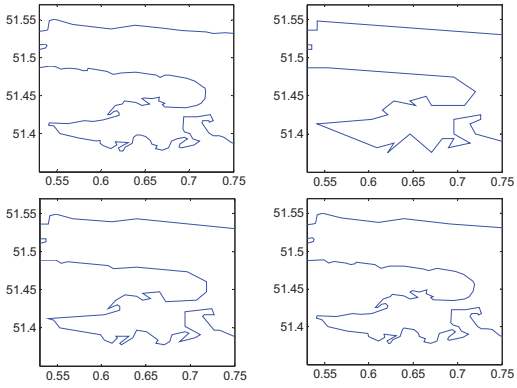

Fig. 6, *Performance under different bit-rate on a fragment of the test curve. Original 128 bits/point (top-left), 2 bits/point (top-right),5 bits/point (bottom-left),10 bits/point(bottom-right)*

4. CONCLUSION

We have proposed a fast *dynamic quantization* algorithm for lossy compression of vector map. The underlying algorithm first identified an optimal *Lagrangian* multiplier λ value for each quantization step *l* and then constructed only one rate-distortion curve for design of predicted-error quantizer. In addition, a powerful searching criterion was exploited for the sake of speeding up the dynamic quantization.

Experimental results have shown that the proposed method is twenty times faster than the previous dynamic quantization algorithm but achieves a similar or better compression performance. Future work can be considered in the following perspectives:

1. The dynamic quantization can be improved by combining vector quantization and uniform product quantization.
2. Lattice VQ can be used instead of uniform quantization.
3. Linear prediction can be considered to improve the prediction of the residual vectors.

5. REFERENCES

[1] G. M. Schuster and A. K. Katsaggelos, "An optimal polygonal boundary encoding scheme in the rate-distortion sense", *IEEE Trans. on Image Processing*, vol.7, pp. 13-26, 1998.

[2] Z. Li, and S. Oppenshaw, "A natural principle for the objective generalization of digital maps", *Cartography and Geographical Information Systems*, vol. 20, pp. 19-29, 1993.

[3] A. Akimov, A. Kolesnikov and P. Fränti, "Coordinate quantization in vector map compression", *IASTED Conference on Visualization, Imaging and Image Processing* (*VIIP'04*), pp. 748-753, 2004.

[4] S. Shekhar, S. Huang, Y. Djugash, J. Zhou, "Vector map compression: a clustering approach", *10th ACM Int. Symp. Advances in Geographic Inform*, pp.74-80, 2002.

[5] A. Akimov, A. Kolesnikov and P. Fränti, "Reference line approach for vector data compression", *IEEE Int. Conf. on Image Processing (ICIP'04)*, vol. 2, pp. 1891-1894, 2004.

[6] A. Kolesnikov, "Optimal encoding of vector data with polygonal approximation and vertex quantization", *SCIA'05*, LNCS, vol. 3540, 1186–1195. 2005.

[7] A. Kolesnikov, "Optimal algorithm for lossy vector data compression", *Int. Conf. on Image Analysis and Recognition* (ICIAR'07), LNCS 4633, pp. 761-771, 2007.

[8] B. Yang and Q. Li, "Efficient compression of vector data map based on a clustering model", *Geo-spatial Information Science*, 12(1), 13-17, 2009.

[9] A. Masood, "Optimized polygonal approximation by dominant point deletion", P*attern Recognition,* 41 (1), 227-239, 2008.

[10] P. Bhowmick and B. Bhattacharya, "Fast polygonal approximation of digital curves using relaxed straightness properties", *IEEE Trans. on PAMI,* 29 (9), 1590-1602, 2007.

[11] J. C. Perez, E. Vidal, "Optimum polygonal approximation of digitized curves", *Pattern Recognition Letters,* 15, 743–750, 1994.

[12] M. Salotti, "Optimal polygonal approximation of digitized curves using the sum of square deviations criterion", *Pattern Recognition* 35, 435–443, 2002.

APPENDIX: PROOF OF (10)

The cost function is defined as: $J= E + \lambda R$. In uniform product quantization, for a given quantization step l, mean square error E can be calculated by:

$$\int_0^{l/2} \frac{1}{l} x^2 dx + \int_{l/2}^{l} \frac{1}{l} (l-x)^2 = \frac{1}{2} E$$

where $E=l^2/6$. For residual vector $\boldsymbol{v}_i$, after uniform product quantization, if it is estimated by *geometric distribution*,

$$f(|q_x|) = (1-p_x)^{|q_x|} p_x, f(|q_y|) = (1-p_y)^{|q_y|} p_y$$

The coding length can be approximated as:

$$r(\boldsymbol{v}_i) = -(\log_2((1-p_x)^{|q_{xi}|} \cdot (p_x)) + -(\log_2(1-p_y)^{|q_{yi}|} \cdot p_y) + 2$$

The maximum likelihood estimation of *geometric distribution* is:

$$p_x = 1/(\frac{1}{n}\sum_{i=1}^{n} |q_{xi}| + 1) \approx \frac{1}{|\overline{\Delta x_i}|/l + 1} = \frac{|\overline{\Delta x_i}|}{|\overline{\Delta x_i}| + l}$$

$$\therefore R = \frac{1}{n}\sum_{i=1}^{n} r(\boldsymbol{v}_i) \approx -(\frac{|\overline{\Delta x_i}|}{l} \log_2 \frac{|\overline{\Delta x_i}|}{|\overline{\Delta x_i}| + l} + \log_2 \frac{l}{|\overline{\Delta x_i}| + l}) +$$

$$-(\frac{|\overline{\Delta y_i}|}{l} \log_2 \frac{|\overline{\Delta y_i}|}{|\overline{\Delta y_i}| + l} + \log_2 \frac{l}{|\overline{\Delta y_i}| + l}) + 2$$

Define $r_x = |\overline{\Delta x_i}|/l, r_y = |\overline{\Delta y_i}|/l$,

$$R \approx -(r_x \log_2 \frac{r_x}{r_x+1} + \log_2 \frac{1}{r_x+1}) + -(r_y \log_2 \frac{r_y}{r_y+1} + \log_2 \frac{1}{r_y+1})) + 2$$

$$\text{As } r_x, r_y >> 1, r_x \cdot \log_2 \frac{r_x}{r_x+1} \approx \frac{1}{\ln 2}, \log_2 \frac{1}{r_x+1} \approx \log_2 \frac{1}{r_x} = -\log_2 r_x,$$

$$\therefore R \approx -(1/\ln 2 - \log_2 r_x) + -(1/\ln 2 - \log_2 r_y)) + 2$$

$$\approx -(1/\ln 2 - \log_2 |\overline{\Delta x_i}| + \log_2 l) + -(1/\ln 2 - \log_2 |\overline{\Delta y_i}| + \log_2 l)) + 2$$

$$\partial J_0 / \partial l \approx \frac{l}{3} - \frac{\lambda}{l \ln 2} - \frac{\lambda}{l \ln 2} \approx \frac{l}{3} - \frac{2\lambda}{l \ln 2}$$

By setting $\partial J_0 / \partial l = 0$, We got $\lambda \approx 1/6 \cdot l^2 \ln 2$ □

Paper P6

M. Chen, M. Xu, P. Fränti, "Optimized Entropy-constrained Vector Quantization of Lossy Vector Map Compression", *IEEE Int. Conf. on Pattern Recognition* (ICPR'10), Istanbul, Turkey, 722-725, 2010.

2010 International Conference on Pattern Recognition

Optimized entropy-constrained vector quantization of lossy vector map compression

Minjie Chen[1], Mantao Xu[2], Pasi Fränti[1]
[1]*University of Eastern Finland, Finland;* [2]*Shanghai Dianji University, China*
E-mail: {mchen,franti}@cs.joensuu.fi, mantao.xu@gmail.com

Abstract

Quantization plays an important part in lossy vector map compression, for which the existing solutions are based on either a fixed size open-loop codebook, or a simple uniform quantization. In this paper, we proposed an entropy-constrained vector quantization to optimize both the structure and size of the codebook at the same time using a closed-loop approach. In order to lower the distortion to a desirable level, we exploit two-level design strategy, where the vector quantization codebook is designed only for most common vectors and the remaining (outlier) vectors are coded by uniform quantization.

1. Introduction

Vector maps consist of geographic information such as waypoints, routes and areas, which can be represented as a sequence of points in a given coordinate system. To reduce the archive space and transmission time, a variety of compression algorithms have been investigated and developed for compressing vector maps [2-6]. Existing lossy compression algorithms use two different strategies: *polygonal approximation* and *quantization-based method*.

In *polygonal approximation,* the number of points is reduced and the curve represented by a coarser approximation [1]. In *quantization-based method*, differential coordinates of subsequent sampling points are considered as the prediction error and these residual vectors are quantized using various methods, such as *uniform quantization* [2], *product scalar quantization* [3] *and vector quantization* with fixed size codebook [4]. In [5], *reference line* method first identified a series of references lines by polygonal approximation and then estimated prediction errors for the remaining points according to their nearest reference lines followed by *product scalar quantization* in a similar manner to [3]. In [6], *dynamic quantization* was studied where the curve approximation was performed by taking into consideration vector quantization of the approximation line segments.

In this paper, we propose three concrete improvements for the *quantization-based method*. Firstly, all the previous methods use a fixed size codebook, whereas *entropy-constrained pair-wise nearest neighbor for vector quantization* (ECPNN-VQ) is used that optimizes the size of the codebook as well. As a merge-based clustering method, ECPNN-VQ will stop reducing the size of the codebook when a given bit-rate constraint is met.

Secondly, the codebook is further optimized where the codebook of vector quantization is only applied for most common vectors and the rest (*outliers*) are processed by uniform quantization. In contrary to typical image compression, vector data has a wide dynamic range that can vary significantly from a dataset to another. In conventional approaches, a large-size codebook has therefore been required in order to achieve lower distortion. In practice, however, the code length and cost of the codebook itself must also be taken into account, which is ignored in the existing vector quantization methods. In specific, large-size codebooks are required when high bit rate is desired, and it is difficult to achieve a desirable compression performance due to the additional code length of the codebook. To attack this problem, an additional *"outliers"* cluster is designed for vectors that differ too much from the majority of the vectors. Those vectors with significantly high cost in the rate-distortion sense are selected as *"outliers"* and coded by a separate escape codeword. The underlying quantization method leads to a robust compression performance both for high and low bit-rates.

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.182

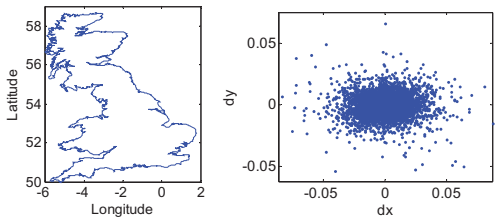

Fig. 1, Test map Britain with 10,910 points (left), and the prediction residuals with MSE =1.8·10^{-4}.

Finally, the close-loop structure in the encoding step was improved by dynamic programming algorithm. The remainder of the paper is organized as follows. The proposed method is introduced in section 2; Section 3 reports the corresponding experimental results; and conclusions are drawn in section 4.

2. Proposed Method

In vector map compression, we want to compress a given 2-D vector sequence: P= (p_1, p_2,…, p_n), under given bit-rate constraint c. General compression procedure contains three steps: *prediction*, *quantization* of the residual vectors and *entropy coding*. Differential coordinates of subsequent sampling points are used as the prediction error in prediction step. A sample test curve and the distribution of the corresponding differential coordinates for which the quantization codebook is designed, are shown in Fig. 1.

2.1 Initial the codebook in vector quantization

In vector quantization, we want to quantize the residual vectors by minimizing mean square error under a constraint that the average bit-rate does not exceed c:

$$\min D, s.t. R < c, where\ D = \sum_{i=1}^{N} (\| \boldsymbol{v}_i - Q(\boldsymbol{v}_i) \|^2) \qquad (1)$$

$\boldsymbol{v_i}$ is the residual vector and $Q(\boldsymbol{v_i})$ is its quantized form.

In entropy-constrained vector quantization (ECVQ), the problem can be solved by a *Lagrangian minimization procedure* by converting it as an unconstrained optimization problem [7], formulated as $J = D + \lambda R$. For each Lagrangian parameter λ, it has a corresponding point on the rate-distortion curve. However, unlike image coding, prediction error for vector data varies for different case, and thus, using a fixed size (k) in codebook design does not solve the problem efficiently. In order to find a better combination of λ and k, ECVQ can be applied but at the cost of higher time complexity, which makes it suitable only off-line.

Entropy-constrained pair-wise nearest neighbor for vector quantization (ECPNN-VQ) has been proposed in [8]. It merges the pair of cluster that results in the smallest increase in distortion and largest decrease in rate. The increased distortion after merging two clusters i and j can be calculated by:

$$\Delta D = \frac{n_i + n_j}{n_i n_j} \| \boldsymbol{c}_i - \boldsymbol{c}_j \|_2^2 \qquad (2)$$

n_i and n_j are the number of vectors in cluster i and j, respectively, and $\boldsymbol{c_i}$ and $\boldsymbol{c_j}$ are their centroid vectors. The change in bit-rate can be calculated as:

$$\begin{aligned} &\Delta R = -n_i \log(n_i / n) - n_j \log(n_j / n) \\ &-(-(n_i + n_j) \log((n_i + n_j) / n) + r_q) \end{aligned} \qquad (3)$$

r_q is the code length of one quantized vector in codebook, n is the number of residual vector.

In every merge step, the pair of clusters with minimum $-\Delta D/\Delta R$ is merged. This can also be considered as searching the minimum slope in the rate-distortion curve, and thus, it guarantees the optimality of each merge step. Since in classic ECVQ framework, λ is interpreted as the slope of the line supporting the operational rate-distortion curve, and therefore, in ECPNN-VQ, it is approximated by:

$$\lambda \approx -(D^{n+1} - D^n)/(R^{n+1} - R^n). \qquad (4)$$

The time complexity of ECPNN-VQ is $O(\tau N^2)$, the same as that of the traditional PNN algorithm [9].

2.2 Optimize quantization by outlier cluster

After ECPNN-VQ, the cost of each residual vector i in cluster j can be formulated as:

$$J_{ij} = \| \boldsymbol{v}_i - \boldsymbol{c}_j \|_2^2 + \lambda(-\log_2(n_j / n) + r_q / n_j) \qquad (5)$$

If a very high accuracy is needed, a large-sized codebook is intractable in achieving a desirable coding efficiency. Better compression performance can be achieved by uniform quantization because no overhead is needed for storing the codebook. Therefore, we apply two-level codebook so that the most common vectors are coded by vector quantization using the optimized codebook, while the outlier vectors are coded by uniform quantization. In uniform quantization, given quantization level l, residual vector $\boldsymbol{v_i}$ is quantized as $Q(\boldsymbol{v_i}) = ([\Delta x_i/l]\cdot l, [\Delta y_i/l]\cdot l)$, where $q_{xi} = [\Delta x_i/l]$, $q_{yi} = [\Delta y_i/l]$ are integer value be coded.

The mean square error D_0 can be calculated by:

$$\int_0^{l/2} \frac{1}{l} x^2 dx + \int_{l/2}^{l} \frac{1}{l} (l - x)^2 = \frac{1}{2} D_0 \qquad (6)$$

We have $D_0 = l^2/6$.

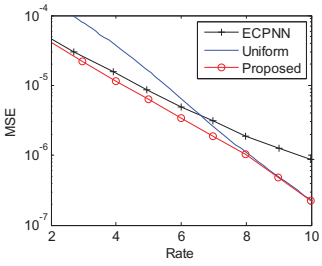

Fig.2, Comparison of the vector quantization of the residual vectors

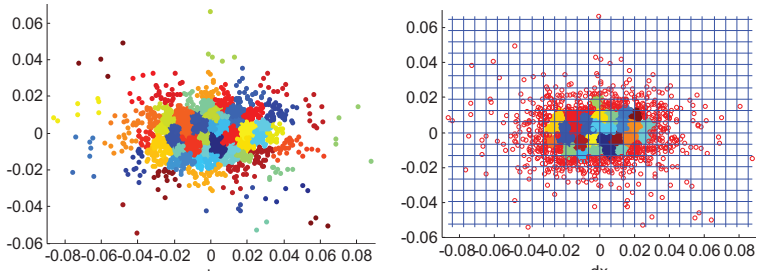

Fig. 3, Demonstration of the outlier selection (5 bits/point constraint). ECPNN with MSE= $8.7\cdot10^{-6}$ and codebook size 78(left). The proposed two-level codebook with MSE= $6.9\cdot10^{-6}$ and size 30 (right). Outliers are marked as 'o'. Grid size for uniform quantization is also labeled.

We observe that the $|q_x|$ and $|q_y|$ can be described by *geometric distribution*:

$$f(|q_x|) = (1-p_x)^{|q_x|} p_x \tag{7}$$

where p_x is the parameter, which is approximated by:

$$p_x = 1/(\frac{1}{n}\sum_{i=1}^{n}|q_{xi}|+1) \tag{8}$$

The coding length for uniform quantization is:

$$\begin{aligned} r_{i0} = &-(|q_{xi}|\log_2(1-p_x)+\log_2(p_x)) - \log_2(n_0/n)+2 \\ &-(|q_{yi}|\log_2(1-p_y)+\log_2(p_y)) \end{aligned} \tag{9}$$

where n_0 is the number of vectors used for uniform quantization. The cost of uniform quantization is calculated by: $J_0 = \Sigma_i(D_0 + \lambda r_{i0})$. By setting $\partial J_0 / \partial l = 0$, we got the optimal quantization level:

$$l \approx \sqrt{6\lambda / \ln 2} \tag{10}$$

In our method, ECPNN is used to initialize the codebook. For a given bit constraint c, λ is first approximately on the rate distortion curve by (4). *"Outlier cluster"* is then created with quantization level l by (10). Residual vectors are repartitioned to the clusters with minimum cost J by:

$$Q(\boldsymbol{v}_i) = \boldsymbol{c}_j, j = \arg\min_j(J_{ij}), j = 0,1..,k \tag{11}$$

A centroid step is followed in order to update the codebook. Parameters p_x and p_y for the *geometric distribution* are also updated. Fig.3 shows an example of the codebook design. We can observe that several vectors and some clusters have completely been moved to the *"outlier cluster"*, and the size of the main codebook is reduced from 78 to 30.

The corresponding rate-distortion curve of the two-layer quantization step in Fig.2 shows better rate-distortion performance than the corresponding one-level ECPNN, or the uniform quantization under different bit-rate conditions. We should mention when the data has different property, *uniform distribution*, *negative binomial distribution* or *Poisson distribution* can also be considered instead of *geometric distribution*.

2.3. Encoding by closed-loop with dynamic programming

After prediction and quantization of the residual vectors, we compress the vector data by entropy encoding. In order to avoid error propagation, the prediction must take into account the quantization effect of the previous point by using closed-loop prediction:

$$\boldsymbol{v}_i = p_i - p_{i-1}^r \tag{12}$$

$$p_i^r = Q(\boldsymbol{v}_i) + p_{i-1}^r \tag{13}$$

here p_i^r is the approximated point after entropy coding. The total cost can be calculated by:

$$L_n = \sum_{i=1}^{n} J_i, where\ J_i = \| p_i^r - p_i \|_2^2 + \lambda r_i \tag{14}$$

where r_i is the coding length for p_i.

Selecting the quantized vector according to (13) cannot guarantee optimality during minimizing the cost function in this encoding procedure. In our method, we keep more possible candidates (t=8 in our implementation) in each step and the optimal solution is found by a dynamic programming process in the state space of size $n\cdot t$. Suppose that t is the best solution recorded for encoding from p_1 to p_i, with the corresponding costs $L_{i,1}, L_{i,2}, \ldots L_{i,t}$, $p_{i,1}^r, p_{i,2}^r, ..., p_{i,t}^r$ is the approximating points for p_i. Based on a combination of k quantized vector and t best solutions for p_i, $k\cdot t$ solutions are tested for approximating p_{i+1} and t best solutions $p_{i+1,1}^r, p_{i+1,2}^r, ..., p_{i+1,t}^r$ is saved with minimum costs $L_{i+1,1}$, $L_{i+1,2}, \ldots L_{i+1,t}$. In the end, backtracking is done to find the quantized vectors from $p_{n,1}^r$ with minimum cost $L_{n,1}$. The time complexity of proposed approach is O($ktn\cdot log\ kt$).

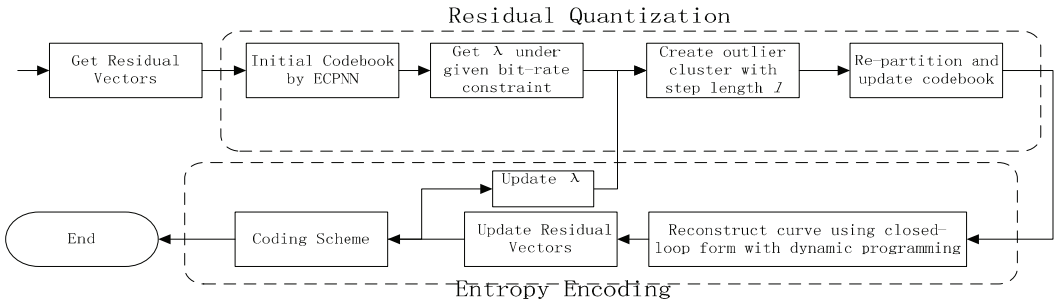

Fig. 4 Workflow of the proposed method

The residual vectors can be updated by (12) after the approximated curve has been constructed. Given bit-rate constraint c, λ is updated by a binary search in the next iteration.

3. Experiments

We evaluated the proposed algorithm with optimal codebook design (OCVQ) on a 10,911-point vector map representing the contour of Britain (Fig. 1). For comparison, two alternative methods are investigated in the experimental tests: *Clustering-based method* (CBC) [4] and the *reference line method* (RL) [5]. The distortion is measured by mean squared error (MSE). We further integrate our method into *Dynamic quantization* (DQ) [6], where integral square error (ISE) is used as the error measure. The corresponding rate-distortion curves are plotted in Fig. 5. The proposed algorithm compares favorable with the existing approach.

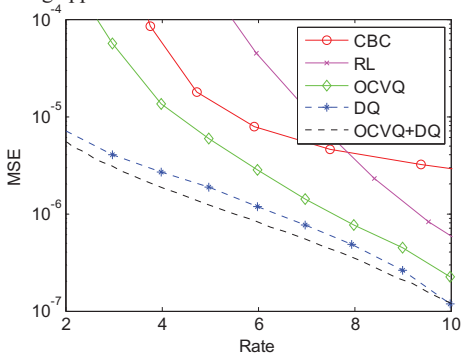

Fig. 5, Performance comparison

4. Conclusion

We propose a lossy compression algorithm for vector map data under a certain bit-rate constraint. In a comparison to the previous clustering-based method, a two-level strategy has been exploited and employed to optimize the codebook design. Vector quantization codebook is designed only for most common vectors, and the remaining vectors (outliers) are coded by additional bits using uniform quantization. Additionally, a dynamic programming method is utilized to improve the quantized vector selection in closed-loop framework, instead of using a conventionally greedy approach.

References

[1] G. M. Schuster and A. K. Katsaggelos, "An optimal polygonal boundary encoding scheme in the rate-distortion sense, *IEEE Trans. on Image Processing*, vol.7, pp. 13-26, 1998.

[2] Z. Li, and S. Oppenshaw, "A natural principle for the objective generalization of digital maps", *Cartography and Geographical Information Systems*, vol. 20, pp. 19-29, 1993.

[3] A. Akimov, A. Kolesnikov and P. Fränti, "Coordinate quantization in vector map compression", *Proc. of IASTED Conference on Visualization, Imaging and Image Processing* (*VIIP'04*), pp. 748-753, 2004.

[4] S. Shekhar, S. Huang, Y. Djugash, J. Zhou, "Vector map compression: a clustering approach", *Proc. 10th ACM Int. Symp. Advances in Geographic Inform*, pp.74-80, 2002.

[5] A. Akimov, A. Kolesnikov and P. Fränti, "Reference line approach for vector data compression", *IEEE Int. Conf. on Image Processing (ICIP'04)*, vol. 2, pp. 1891-1894, 2004.

[6] A. Kolesnikov, "Optimal algorithm for lossy vector data compression", *Int. Conf. on Image Analysis and Recognition* (ICIAR'07), LNCS 4633, pp. 761-771, 2007.

[7] P. Chou, T. Loolabaugh and R. M. Gray, "Entropy-constrained vector quantization" , *IEEE Trans. on Acoustics, Speech, Signal Processing*, 37(1), pp. 31–42,1989.

[8] F. Kossentini, "A fast PNN design algorithm for entropy - constrained residual vector quantization", *IEEE Trans. on Image Processing*, vol.7, pp.1045-1050, 1998.

[9] P. Fränti, T. Kaukoranta, D.-F. Shen and K.-S. Chang, "Fast and memory efficient implementation of the exact PNN", *IEEE Transactions on Image Processing*, 9 (5), 773-777, May 2000.

Paper P7

M. Chen, M. Xu, P. Fränti, "Compression of GPS Trajectories",
Data Compression Conference (DCC'12), 62-71, Snowbird, USA,
2012.

Compression of GPS trajectories

Minjie Chen[1], Mantao Xu[2] and Pasi Franti[1]

[1] *School of Computing, Univ. of Eastern Finland, Finland*
{mchen, franti }@cs.joensuu.fi

[2] *Shanghai Dianji University, China*
xumt@sdju.edu.cn

***Abstract**:* Enormous amounts of GPS trajectories, which record users' spatial and temporal information, are collected by geo-positioning mobile phones in recent years. The massive volumes of trajectory data bring about heavy burdens for both network transmission and data storage. To overcome these difficulties, a number of compression algorithms have been proposed by reducing the number of points in the trajectory data. But these algorithms lack a rigorous investigation on how to encode the reduced trajectories. In this paper, we propose an algorithm that optimizes both the trajectory simplification and the coding procedure using the quantized data. The underlying algorithm is also compared with the existing methods across 640 trajectories from *Microsoft Geolife dataset* using s*ynchronous Euclidean distance* (SED) as the error metrics. Experimental results show that the proposed method saves 60% of compression cost against the current state of the art compression algorithms.

1. Introduction

Location-acquisition technologies have generated a great deal of enthusiasms in the global mobile market in recent years. For example, the Location Based Services (LBS) enables the end-users of GPS mobile phones to obtain their locations, and therefore record travel experiences by a number of time-stamped trajectories. However, most of LBS applications bring about an enormous volume of data to the end-users as well as incur a large amount of redundant storage and a longer uploading/downloading time to mobile service providers. For example, if data is collected at 10 second intervals, a calculation in [1] shows that without any compression, 100 Mb of storage capacity is required to store the GPS trajectories of 400 users for a single day in server side.

To overcome this difficulty, a number of GPS trajectory compression algorithms have been studied in literature. In these algorithms, an approximation technique called *line simplification* has been treated as an active research topic in data reduction of the GPS trajectories, amongst which the Douglas-Peucker algorithm [2] is the most popular one. The algorithm divides the line segment with biggest deviation at each step until the approximated error is smaller than a given error tolerance.

Later, Meratnia [1] indicated that such algorithms were not suitable for GPS trajectory since both spatial and temporal information should be considered. Thus, the so-called TD-TR algorithm was developed, where *synchronous Euclidean distance* was used instead of the perpendicular distance in the Douglas-Peucker algorithm. Similarly,

[1] The research is partially supported by Tekniikan edistämissäätiö (TES), Nokia Scholarship, East Finland Graduate School in Computer Science (ECSE), National Natural Science Foundation of China (Grant No 61072146) and Shanghai Committee of Science and Technology, China (Grant No 10PJ1404400).

1068-0314/12 $26.00 © 2012 IEEE
DOI 10.1109/DCC.2012.14

Opening Window [1] algorithm was also proposed to solve the line simplification problem sequentially in a greedy manner. Another solution [3] was the *threshold-guided algorithm* via estimating the safe area of the next point using the position, speed and orientation information. *STTrace* sampling algorithm was also implemented using a bottom-up strategy such that the synchronous Euclidean distance was minimized in each step. In [4], a new distance-function called *spatial join* was proposed, which was bounded for spatial-temporal queries.

Recently, a new simplification algorithm SQUISH [11] has been proposed based on the priority queue data structure that preserves the speed information at a much higher accuracy. A multi-resolution simplification algorithm MRPA has also been designed with linear time complexity in [12]. Semantic meanings of the GPS trajectories are also considered during the compression process in urban area in [15] whereas trajectory compression algorithm with network constraint has been developed in [14]. Performance evaluations are also made for several traditional trajectory simplification algorithms [10]. However, there is not one algorithm that always outperforms other compression approaches in all situations [11].

In these algorithms, the performance is measured only on the reduction rate by the line simplification process. The reduced data points are saved directly, which is useful to support the effective trajectory queues in database. However, they lack of a rigorous analytical approach on the encoding procedures of the reduced trajectories. Namely, without further compression after line simplification, 12 bytes are needed at minimum for encoding each point including latitude, longitude and timestamp information. On the other hand, when data compression techniques are used, we can achieve a better compression ratio for the spatial trajectory data, which is appropriate for data storage.

To circumvent the above problem, a quantization technique can be applied to further improve the encoding procedure of these reduced GPS trajectories. The quantization approach has been analytically investigated in the so-called *vector map compression* problem [5-8]. In these algorithms, differential coordinates of adjacent data points are used as the prediction error and the residual vectors are then quantized using a variety of quantization strategies, including uniform quantization, product scalar quantization [5] and vector quantization with fixed-size codebook [6]. Amongst them, a pioneer solution in [7] combines both the advantage of line simplification and quantization via dynamic programming and hence the terminology of *dynamic quantization* (DQ). Likewise, the quantized vectors can be further compressed by arithmetic coding based on the assumption that the resulting quantized differential coordinates obeys a geometric distribution [8]. In all these methods, timestamp information is not considered.

In this paper, we consider the problem of lossy compression for GPS trajectories with latitude, longitude and timestamp information, under a given error tolerance, i.e., synchronous Euclidean distance. In contrast to the existing algorithms, we achieve two significant improvements described below. Firstly, speed and direction changes are incorporated in the encoding process instead of using the differential coordinates in the previous methods. Secondly, line simplification and quantization are combined in the encoding procedures in order to seek the approximated trajectory for compression.

The rest of the paper is organized as follows. The proposed GPS trajectory compression (GTC) algorithm is introduced in Section II, experimental results are reported in Section III, and finally, conclusions and discussions are drawn in Section IV.

2. Proposed GPS Trajectory Compression Algorithm

2.1. Synchronized Euclidean Distance

In this paper, synchronized Euclidean distance [1] is used as the error metrics for evaluating the distortion of the approximated (compressed) GPS trajectories. Here, the error is measured through distances between pairs of temporally synchronized positions, one on the original and one on the approximated trajectories, which can be formulated as follows:

Suppose $P = \{p_1, p_2, \ldots, p_n\} = \{(x_1, y_1, t_1),\ldots,(x_n, y_n, t_n)\}$ and $P' = \{p_{i1}', p_{i2}', \ldots, p_{im}'\} = \{(x_{i1}', y_{i1}', t_{i1}'), \ldots, (x_{im}', y_{im}', t_{im}')\}$ are the original and the corresponding approximated GPS trajectories with $i_1 < i_2 < \ldots < i_m$, $i_1 = 1$, $i_m = n$ and $m \le n$. For each point $p_j = (x_j, y_j, t_j)$ on the original trajectory, its approximated synchronized position can be calculated as:

$$x_j' = x_{i_k}' + \frac{t_j - t_{i_k}'}{t_{i_{k+1}}' - t_{i_k}'} \cdot (x_{i_{k+1}}' - x_{i_k}'), \text{ where } t_{i_k}' \le t_j \le t_{i_{k+1}}' \tag{1}$$

$$y_j' = y_{i_k}' + \frac{t_j - t_{i_k}'}{t_{i_{k+1}}' - t_{i_k}'} \cdot (y_{i_{k+1}}' - y_{i_k}'), \text{ where } t_{i_k}' \le t_j \le t_{i_{k+1}}' \tag{2}$$

After p_i' is determined, synchronized Euclidean distance is calculated by:

$$SED(p_i, p_i') = \sqrt{(x_i - x_i')^2 + (y_i - y_i')^2} \tag{3}$$

In order to evaluate the distortion of the whole trajectory, maximum synchronized Euclidean distance is used, which is defined as:

$$SED(P, P') = \max_{1 \le i \le n}(SED(p_i, p_i')) \tag{4}$$

In synchronized Euclidean distance, the continuous nature of moving objects necessitates the inclusion of temporal as well as spatial properties of moving objects.

2.2. Approximate GPS trajectory with given error bound

We consider both the line simplification and the quantization in the approximation process. In vector map compression, differential coordinates are used in the encoding process. However, for GPS trajectories, because of the inconsistency of the differential coordinates after the approximation process, the coding efficiency may be reduced if differential coordinates are used directly. Meanwhile, speed and direction will be more consistent even if a simplification (approximation) step is applied with different reduction rate in different segments.

An approximation example is demonstrated in Fig. 1, suppose we want to approximate a sub-trajectory $P_{i_k}^{i_{k+1}}$ by line segment $\overline{p_{i_k}' p_{i_{k+1}}'}$, where p_{i_k}' is the approximated position in previous step. If time interval $\Delta t(k)$ is known, the speed of the line segment is:

$$spd(k) = d(p_{i_k}', p_{i_{k+1}}) / \Delta t(k) \tag{5}$$

Given maximum SED tolerance ε, we assume the quantization error of point $p_{i_{k+1}}'$ is $\gamma\varepsilon$ at maximum, thus the quantized level for speed can be set as:

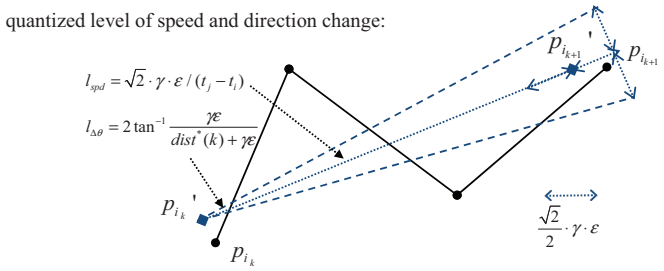

Fig. 1: An example of the quantization process for the GPS trajectory

$$l_{spd}(k) = \sqrt{2} \cdot \gamma \cdot \varepsilon / \Delta t(k) \tag{6}$$

Here γ is a parameter as the ratio of the quantized error and the total SED on the target point, which is set as γ = 0.5 by our experiment. Thus, the quantized speed for approximated segment can be calculated as:

$$spd^*(k) = [spd(k) / l_{spd}(k)] \cdot l_{spd}(k) \tag{7}$$

Meanwhile, we can get the direction change $\Delta\theta(k)$ with a value between $-\pi$ and π, where negative value represents the direction change in clockwise.

Given the quantized speed $spd^*(k)$, the quantization level for the direction change can be estimated by:

$$l_{\Delta\theta}(k) = 2\tan^{-1} \frac{\sqrt{2}\gamma\varepsilon / 2}{spd^*(k) \cdot \Delta t(k) + \sqrt{2}\gamma\varepsilon / 2} \tag{8}$$

Thus, the quantized direction change is:

$$\Delta\theta^*(k) = [\Delta\theta(k) / l_{\Delta\theta(k)}] \cdot l_{\Delta\theta(k)} \tag{9}$$

Based on the quantized speed and direction change $spd^*(k)$ and $\Delta\theta^*(k)$, the quantized position $p_{i_{k+1}}' = \{x_{i_{k+1}}', y_{i_{k+1}}', t_{i_{k+1}}'\}$ can be approximated as:

$$\begin{aligned} x_{i_{k+1}}' &= x_{i_k}' + spd^*(k) \cdot \Delta t(k) \cdot \cos(\Delta\theta^*(k) + \theta^*(k-1)) \\ y_{i_{k+1}}' &= y_{i_k}' + spd^*(k) \cdot \Delta t(k) \cdot \sin(\Delta\theta^*(k) + \theta^*(k-1)) \\ t_{i_{k+1}}' &= t_{i_{k+1}} \end{aligned} \tag{10}$$

A greedy solution is used for the trajectory approximation in this paper. Start from the first point, the farthest point is found with an approximated SED less than the given error tolerance $SED(P_{i_k}^{i_{k+1}}, \overline{p_{i_k}' p_{i_{k+1}}'}) \le \varepsilon$ for every line segment $\overline{p_{i_k}' p_{i_{k+1}}'}$. The pseudocode can be seen in Fig. 2.

Note that when the input of GPS trajectory is latitude and longitude in WGS84 format, *Mercator projection* is needed as a preprocessing step so that the distance can be calculated directly.

ALGORITHM I, APPROXIMATION OF GPS TRAJECTORY

INPUT

$P = \{p_1, p_2, \ldots, p_n\}$: original trajectory

ε: SED error tolerance

OUTPUT

$P' = \{p_{i1}', p_{i2}', \ldots, p_{im}'\}$

$i \leftarrow 1$

$j \leftarrow 2$

$p_{i1}' \leftarrow p_1$

$m \leftarrow 2$

FOR $i = 1$ **TO** $i = n - 1$ **DO**

  IF $j > n$

    $p_{im}' \leftarrow p_n'$

    BREAK

  END

  $d \leftarrow SED(P_i^j, \overline{p_i' p_j'})$

  IF $d \le \varepsilon$

    $j \leftarrow j + 1$

  ELSE

    $p_{im}' \leftarrow p_{j-1}'$

    $m \leftarrow m + 1$

    $i \leftarrow j - 1$

    $j \leftarrow i + 1$

  END

END

Fig. 2: Pseudocode of proposed GPS trajectory approximation process

2.3. Encoding Process on the Approximated Trajectory

In the encoding process, we need to encode both the differential coordinates and time difference (Δx, Δy and Δt). Suppose p_{i_k}' and $p_{i_{k+1}}'$ are two neighbor points in the approximated trajectory P'. Firstly, the time difference is encoded by the following probability:

$$p(\Delta t(k) = t_{i_{k+1}} - t_{i_k}) = \frac{p + \delta_t / rtsp_{max}}{1 + \delta_t}$$
$$\text{where } p = \mathbf{r_t}(\Delta t(k) / min_{tsp}) / \sum_{s=1}^{rtsp_{max}} \mathbf{r_t}(s), rtsp_{max} = \max(\Delta t(q)) / tsp_{min}, q = 1, 2, ..., m-1. \quad (11)$$

tsp_{min} is minimum sampling time internal on the GPS trajectory (1s in most cases) and δ_t is a bias factor ($\delta_t = 0.01$), vector $\mathbf{r_t}$ is initialized as a zero vector with size $rtsp_{max} \times 1$.

After $\Delta t(k)$ has been encoded, vector $\mathbf{r_t}$ is updated by:

$$r_t(s) = \begin{cases} 1 + \mu_t r_t(s), & s = \Delta t_k / tsp_{min} \\ \mu_t r_t(s), & \text{else} \end{cases} \quad (12)$$

where μ_t is a forgetting factor[13] which gives higher influence on the recently encoded time intervals with $\mu_t = 0.995$ in this paper. The reason that we use a forgetting factor is that the possible multi-model in the GPS trajectory. A higher reduction rate can possibly be achieved for the segments with slower moving speed (e.g., by walking) comparing

with fast moving segments (e.g., by car). Thus, it will be helpful to improve the coding performance if a forgetting factor is used.

The speed value is then predicted by $spd_{pred}(k)$ and $\sigma_{spdpred}^2(k)$, which is:

$$\begin{aligned} spd_{pred}(k) &= n_{c1}\sum_i spd^*(i)\cdot(\Delta t(i)\cdot\exp(-\frac{1}{2}\cdot(\frac{t(k)-t(i)}{w_t})^2)), t(i)\geq t(k)-d\cdot\Delta t(k) \\ \sigma_{spdpred}^2(k) &= n_{c2}\sum_i \Delta t(i)\cdot((spd^*(i)-spd_{pred}(k))^2+\frac{\gamma^2\varepsilon^2}{6\Delta t^2(i)}+\frac{\sigma_{GPS}^2}{2\Delta t^2(i)} \end{aligned} \tag{13}$$

n_{c1} and n_{c2} are normalized values for the weighting factors, while w_t , d and σ_{GPS} are parameters with $w_t = 20$, $d = 4$ and $\sigma_{GPS} = 5$. The second and third term of $\sigma_{spdpred}^2(k)$ are the variance of the quantization procedure and the GPS error correspondingly.

The probability is then estimated by assuming the speed has a *Gaussian* distribution:

$$p(spd^*(k)) = \frac{p+\delta_{spd}/nlv_{spd}(k)}{1+\delta_{spd}} \tag{14}$$

where p has a Gaussian distribution with mean $spd_{pred}(k)$ and variance $\sigma_{spdpred}^2(k)$, bias factor δ_{spd} is set as 0.01.

For the direction changes, the encoding process has a similar form with the encoding process for time difference, which is:

$$\begin{aligned} &p(\Delta\theta^*(k)) = \frac{p+\delta_{\Delta\theta}/(2nlv_{\Delta\theta}^0+1)}{1+\delta_{\Delta\theta}}, \\ &\text{where } p = \begin{cases} \sum_{s=s_a(k)}^{s_b(k)}\mathbf{r}_{\Delta\theta}(s)/\sum_{s=-nlv_{\Delta\theta}}^{nlv_{\Delta\theta}}\mathbf{r}_{\Delta\theta}(s), & nlv_{\Delta\theta}^0 < nlv_{\Delta\theta}(k) \\ \frac{\mathbf{r}_{\Delta\theta}([\Delta\theta^*(k)/l_{\Delta\theta}^0])}{(s_d(k)-s_c(k)+1)}/\sum_{s=-nlv_{\Delta\theta}}^{nlv_{\Delta\theta}}\mathbf{r}_{\Delta\theta}(s), & \text{else} \end{cases} \\ &s_a(k) = [(\Delta\theta^*(k)-\frac{1}{2}l_{\Delta\theta}(k))/l_{\Delta\theta}^0], s_b(k) = [(\Delta\theta^*(k)+\frac{1}{2}l_{\Delta\theta}(k))/l_{\Delta\theta}^0] \\ &s_c(k) = [(\Delta\theta^*(k)-\frac{1}{2}l_{\Delta\theta}^0)/l_{\Delta\theta}(k)], s_d(k) = [(\Delta\theta^*(k)+\frac{1}{2}l_{\Delta\theta}^0)/l_{\Delta\theta}(k)] \\ &l_{\Delta\theta}^0 = \pi/nlv_{\Delta\theta}, nlv_{\Delta\theta}(k) = \lceil\pi/l_{\Delta\theta}(k)\rceil \end{aligned} \tag{15}$$

where the size of vector $\mathbf{r}_{\Delta\theta}$ is $(1+2lv_{\Delta\theta}) \times 1$ and $nlv_{\Delta\theta}^0$ is set as 180 from our experiment.

After $\Delta\theta(k)$ has been encoded, vector $\mathbf{r}_{\Delta\theta}$ is updated by:

$$r_{\Delta\theta}(s) = \begin{cases} \frac{1}{s_b(k)-s_a(k)+1}+\mu_{\Delta\theta}r_{\Delta\theta}(s), & nlv_{\Delta\theta}^0 < nlv_{\Delta\theta}(k) \text{ and } s_a(k)\leq s\leq s_b(k) \\ 1+\mu_{\Delta\theta}r_{\Delta\theta}(s), & nlv_{\Delta\theta}^0 \geq nlv_{\Delta\theta}(k) \text{ and } s=[\Delta\theta^*(k)/l_{\Delta\theta}^0] \\ \mu_{\Delta\theta}r_{\Delta\theta}(s) & \text{else} \end{cases} \tag{16}$$

where forgetting factor $\mu_{\Delta\theta}$ is set as 0.995 here.

The probabilities of the time difference, speed and direction change are encoded by arithmetic coding. The pseudocode can be seen in Fig. 3.

Note that in the compressed file, a 192 bits fixed-length head file is used to save the parameters of the trajectory. These values includes start position of x (30 bit), y (30 bit), time(32 bit), tsp_{min} (8 bit), $rtsp_{max}$ (16 bit), m (24 bit), spd_{max} (32 bit) and scaling factor of *Mercator projection* (20 bit).

ALGORITHM II, ENCODING PROCESS OF THE APPROXIMATED TRAJECTORY

INPUT
$P = \{p_1, p_2, \ldots, p_n\}$: original trajectory
ε: SED error tolerance
OUTPUT
Encoding file

P' ← Calculate the approximated trajectory by Figure 2.
Calculate parameters $rtsp_{max}$, tsp_{min}, spd_{max}
Write head file
FOR k**= 1 TO** k **= m - 1 DO**
  Encode $\Delta t(k)$ by (11)
  Update $\mathbf{r_t}$ by (12)
  Predict $spd_{pred}(k)$ and $\sigma_{spdpred}^2(k)$ by (13)
  Encode $spd^*(k)$ by (14)
  Encode $\Delta\theta^*(k)$ by (15)
  Update $\mathbf{r_\theta}$ by (16)
END

Fig. 3: Pseudocode of the encoding process of the proposed GPS trajectory compression algorithm

2.4. Complexity Analysis

In this section, we give the complexity analysis for each step of the proposed algorithm, which is listed the Table I. Note that τ_1, τ_2 and τ_3 are constant values, which are not related to the size of the trajectory data.

TABLE I: Summary of the Expected Time Complexity of the Proposed GPS Trajectory Compression Algorithm

STEP		TIME COMPLEXITY
I. Approximated Trajectory		$O(n^2/m)$
II. Encoding Process	Time Difference	$O(m\cdot\tau_1)$, $\tau_1 = rtsp_{max}$
	Speed	$O(m\cdot\tau_2)$, $\tau_2 = \frac{spd_{\max}}{\varepsilon}\overline{\Delta t}$
	Direction Change	$O(m\cdot\tau_3)$, $\tau_3 = nlv_{\Delta\theta}$
III. Decoding Process		Same as the encoding process

3. Experiments

In order to evaluate the performance of the proposed compression method, we use *Microsoft Geolife dataset* [8] for testing purpose. It includes 640 trajectories with 4,526,030 points, which has a sampling rate between 1s to 5s with different transportation

mode such as walking, bus, car, airplane or a multimodal. The code is written in Matlab and all the experiments have been performed on Intel Pentium-4 3.0 GHz CPU running Windows XP with 2G RAM.

The compression performances (KB/hour) are evaluated for different error tolerance: 1m, 3m, 10m, 30m and 100m maximum synchronous Euclidean distance (SED). The proposed GPS trajectory compression algorithm (GTC) is compared with the state-of-art method TD-TR [1][2]. LZMA (7-zip) is used to further compress the reduced trajectory of TD-TR algorithm[3]. We also evaluate the performance of vector map compression algorithm (VMC) [8] for compressing the position information, while the time information is encoded in the same way as in GTC. We can observe in Fig. 4 (left) that the coding cost of the proposed algorithm is only about 35%-40% compared with TD-TR, and it is consistent on different tolerance level. Note that for an uncompressed GPS trajectory with 1s sampling frequency, if twelve bytes were allocated for each point, the total storage cost would be 42.2 KB/hour. Thus, the proposed method achieves a compression ratio more than 100:1 for a 10m max SED. We also evaluate the reduction rate for different approximation algorithms in Fig. 4 (right). Experiments show that although we design a different reduction and approximation algorithm for the encoding purposes, the reduction rate is still similar.

Time costs of the encoding and decoding process are calculated and shown in Fig. 5 (left). The decoding cost reduces when maximum tolerance increases, because more points are reduced already in the approximation step. For the encoding process, since a $O(n^2/m)$ time complexity is needed for the approximation step, the time cost will slightly increase with the high tolerance case.

We evaluate the bit-rate for each component on the reduced trajectory: time difference, speed and direction change. They are compared with the coding cost using differential coordinates [8], which is given in Fig. 5 (right). We observe that the bit-rate of time difference increases when maximum SED increases. This is because the time difference varies more when the trajectory has a higher reduction rate. However, the bit-rate of speed and direction change will not increase even if a higher tolerance is used. This is because we select a higher quantized level when the given tolerance increases. We also notice that a lower bit-rate is needed when speeds and direction changes are considered instead of the differential coordinates, especially for lower error tolerance.

For the given max SED, its corresponding mean or median SED are also evaluated, which is around 50% of the maximum SED for all the tolerance levels in our experiment.

Note that GPS trajectories are never perfectly accurate, due to sensor noise and other factors. Many filtering algorithms are proposed which are summarized in [16]. From our experiment, if a filtering algorithm is performed beforehand, the bit-rate can be reduced around 30%, 20%, 15% for 1m, 3m, 10m maximum SED correspondingly. Meanwhile, if higher tolerance is set, bit-rate will not be changed even if a filtering operation is used.

[2] Various GPS compression algorithms reported in [10] are all based on the line simplification. There are only around 10%-20% differences on the reduction rate in all these methods. Thus, here TD-TR is selected as a typical example in our experiments to evaluate these types of solutions.

[3] We use a similar evaluation method with a commercial software on: http://www.droyd.org/gps-trajectory-compression.

Further information such as proof of the time complexity, details of the experiment result and the matlab code can be seen on http://cs.joensuu.fi/~mchen/GPSTrajComp.htm.

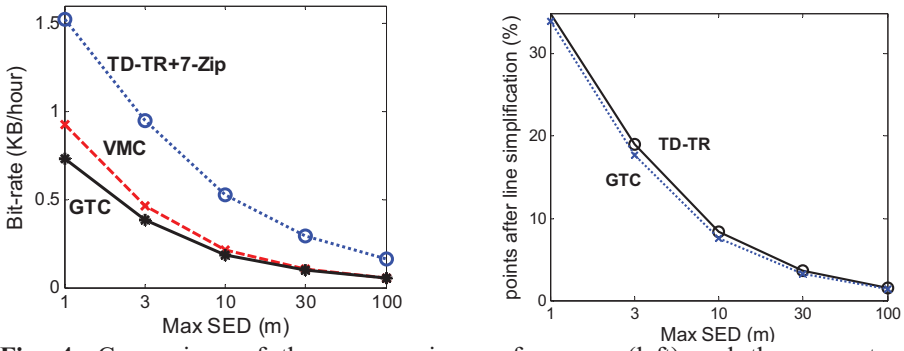

Fig. 4: Comparison of the compression performance (left) and the percentage of remaining points after approximation process.

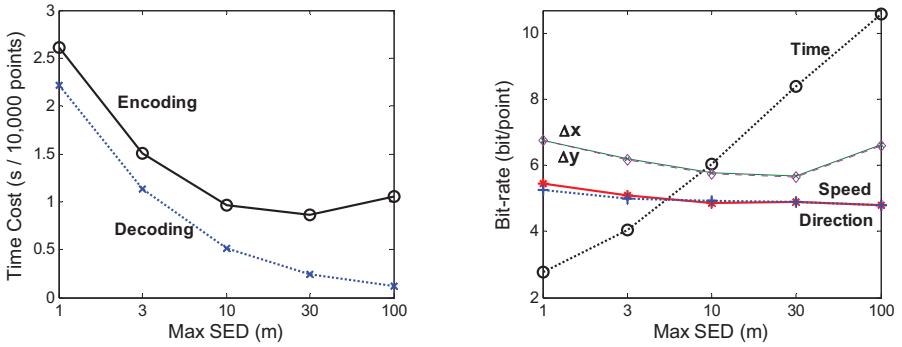

Fig. 5: Time cost of the encoding and decoding process (left), bit-rate of each reduced point for time, differential coordinates (VMC), speed and direction (GTC). (right)

4. Conclusion

In this paper, we have addressed the problem of spatial-temporal data compression, particularly the compression of GPS trajectories with sets of (x, y, t) records. In the proposed algorithm, both data reduction and quantization are considered in the approximation process. Experimental tests demonstrate that the proposed method makes a significant improvement comparing with the state-of-the-art TD-TR algorithm.

There are several immediate extensions of our present work. Firstly, we plan to extend the compression for online application. Also, improvement of approximation and encoding process by dynamic programming will also be considered. Finally, applying a hierarchy of compression stages is an interesting idea for further investigation.

References

[1] N. Meratnia and R. A. de By. "Spatiotemporal Compression Techniques for Moving Point Objects", *Advances in Database Technology*, vol. 2992, 551–562, 2004.

[2] D. H. Douglas, T. K. Peucker, "Algorithm for the reduction of the number of points required to represent a line or its caricature", *The Canadian Cartographer*, 10 (2), 112-122, 1973.

[3] M. Potamias, K. Patroumpas, T. Sellis, "Sampling Trajectory Streams with Spatiotemporal Criteria", *Scientific and Statistical Database Management* (SSDBM), 275-284, 2006.

[4] H. Cao, O. Wolfson, G. Trajcevski, "Spatio-temporal data reduction with deterministic error bounds", *VLDB Journal*, 15(3), 211-228, 2006.

[5] A. Akimov, A. Kolesnikov and P. Fränti, "Coordinate quantization in vector map compression", *IASTED Conference on Visualization, Imaging and Image Processing* (*VIIP'04*), 748-753, 2004.

[6] S. Shekhar, S. Huang, Y. Djugash, J. Zhou, "Vector map compression: a clustering approach", *ACM Int. Symp. Advances in Geographic Inform*, 74-80, 2002.

[7] A. Kolesnikov, "Optimal encoding of vector data with polygonal approximation and vertex quantization", *SCIA'05*, LNCS, vol. 3540, 1186–1195. 2005.

[8] M. Chen, M. Xu and P. Fränti, "Fast dynamic quantization algorithm for vector map compression", *IEEE Int. Conf. on Image Processing,* 4289-4292, September 2010."

[9] Y. Chen, K. Jiang, Y. Zheng, C. Li, N. Yu, "Trajectory Simplification Method for Location-Based Social Networking Services", *ACM GIS workshop on Location-based social networking services*, 33-40, 2009.

[10] J. Muckell, J. H. Hwang, C. T. Lawson, S. S. Ravi, "Algorithms for compressing GPS trajectory data: an empirical evaluation", *SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 402-405, 2010.

[11] J. Muckell, J. H. Hwang, V. Patil, C. T. Lawson, F. Ping , S. S. Ravi, "SQUISH: an online approach for GPS trajectory compression", *International Conference on Computing for Geospatial Research & Applications*, 1-8, 2011.

[12] M. Chen, M. Xu and P. Fränti, "A Fast *O*(*N*) Multi-resolution Polygonal Approximation Algorithm for GPS Trajectory Simplification", *IEEE Transactions on Image Processing* (in press).

[13] M. D. Reavy and C. G. Boncelet, "BACIC: a new method for lossless bi-level and grayscale image compression", *IEEE Int. Conf. on Image Processing*, vol.2, pp. 282-285, 1997.

[14] G. Kellaris, N. Pelekis and Y. Theodoridis, "Trajectory Compression under Network Constraints", *Lecture Notes in Computer Science*, Vol. 5644, pp.392-398, 2009.

[15] F. Schmid, K. F. Richter and P. Laube, "Semantic Trajectory Compression", *Lecture Notes in Computer Science*, Vol. 5644, pp.411-416, 2009.

[16] W. Lee, J. Krumm, "Chapter 1: Trajectory Preprocessing", in Book "Computing with Spatial Trajectories", Springer, 2011.