CN43-1258/ TP
ISSN 1007-130X

计算机工程与科学
COMPUTER ENGINEERING & SCIENCE

2009 年第 31 卷第 6 期
Vol. 31 ,No. 6 ,2009

文章编号:1007-130X(2009)06-0058-04

海量数据流上快速 Top-K 子序列匹配算法研究*

Research on the Fast Top-K Subsequence Matching Algorithm over Massive Data Streams

苏 亮,邹 鹏,贾 焰,杨树强

SU Liang, ZOU Peng, JIA Yan, YANG Shu-qiang

(国防科技大学计算机学院,湖南 长沙 410073)

(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

摘 要:数据流技术在金融分析、网络监控等诸多领域得到了广泛应用,而已有的子序列匹配算法主要针对静态序列,难于直接应用到海量、高速和连续的流数据。本文在动态时间规整技术的基础上,提出了一种新颖的 Top-KSM 算法,能渐进、实时地获取 Top-K 相似子序列。算法完全符合数据流"单遍扫描"的性能要求。大量的实验表明,与现有的 SPRING 算法相比,该算法具有更高的性能。

Abstract: Data stream techniques have been applied in wide fields such as financial analysis and network monitoring. The existing subsequence matching algorithms mainly focus on the static time series and are hardly used to massive, high-speed and continuous steam data. Based on the dynamic time warping technique, this paper provides a novel Top-KSM algorithm which can find the top-K similar subsequences in a progressive and real-time fashion. It completely fits the "single pass" requirement of data streams. Comprehensive experiments show that our algorithm has higher performance than the SPRING algorithm.

关键词:子序列匹配;动态时间规整;Top-K;数据流

Key words: subsequence matching; dynamic time warping; top-K; data stream

中图分类号: TP311

文献标识码: A

1 引言

随着硬件和网络技术的飞速发展,数据流技术已广泛应用在金融分析、网络监控等许多领域。这类应用所产生的流数据通常具有海量、高速和连续的特性,而存储资源相对有限,无法将所有的历史数据都存储到内存中,因此在数据流处理中需要更多地权衡精确性和实时性。时间序列的相似性匹配一直是一个研究热点,它是从时间序列数据集中找出与已知序列相似的序列或子序列。实际应用中极有可能受到机械故障、噪音等的影响,急需一种能同时处理噪音和时间偏移的处理机制。典型的相似性度量标准多为欧氏距离或其改进。但是,它要求序列的长度相等,无法处理数据在时间轴上的形变,同时抗噪声的能力非常弱[1]。因此,在数据流上渐进、实时地进行 Top-K 子序列匹配成为了一个极具价值和挑战性的问题。

针对欧氏距离的缺陷,研究者们提出了动态时间弯曲(Dynamic Time Warping,简称 DTW)度量,它能获得更高的识别率[2~6]。近年来,DTW 的研究主要有序列数据库的索引和 DTW 的界函数[7~9],它们关注的是全序列的匹配和索引而非子序列。

文献[10]首次提出数据流上基于 DTW 的子序列匹配问题,并给出了相应的算法——SPRING 算法,但它没有充分利用相似性阈值(ε),计算上存在较多的冗余。

本文在该算法的基础上提出了一种新颖高效的裁剪机制,更多地考虑数据流中最相似的前 K 个子序列的匹配问题(即 Top-K 子序列匹配问题),并给出了相应的算法——Top-KSM 算法(Top-K Subsequence Matching,简称 Top-KSM)。

大量真实和模拟数据实验表明,Top-KSM 算法与 SPRING 算法相比,在无任何算法精度损失的前提下,仅增加几个字节的空间开销,性能提高很明显。

* **收稿日期**:2008-01-04;**修订日期**:2008-07-16
基金项目:国家 863 计划资助项目(2006AA01Z451,2006AA10Z237);国家 973 计划资助项目(2005CB321804)
作者简介:苏亮(1978-),男,湖南衡山人,博士生,研究方向为分布计算和数据挖掘;邹鹏,教授,研究方向为操作系统和分布计算;贾焰,教授,研究方向为分布计算和数据库;杨树强,教授,研究方向为分布计算和数据库。
通讯地址:410073 湖南省长沙市国防科技大学计算机学院博士生队;Tel:13874969946;E-mail:suliangnudt@gmail.com
Address: School of Computer Science, National University of Defense Technology, Changsha, Hunan 410073, P. R. China

2 相关概念与问题定义

2.1 动态时间规整

标准的DTW算法通常是运用动态规划的思想寻找两序列之间最小的匹配路径。假定长度分别为 m 和 n 的两实数序列 R 和 Q:$R = \langle r_1, r_2, \cdots, r_m \rangle$, $Q = \langle q_1, q_2, \cdots, q_n \rangle$,它们之间的DTW距离定义为:

$$D(R,Q) = \begin{cases} 0, \text{if } m = n = 0 \\ \infty, \text{if } m = 0 \text{ or } n = 0 \\ dist(r_1, q_1) + \min\{D(Rest(R), Rest(Q)), \\ D(Rest(R), Q), D(R, Rest(Q))\}, \text{otherwise} \end{cases} \quad (1)$$

$$dist(r_i, q_j) = \begin{cases} |r_i - q_j|, \text{if } r_i, q_j \text{ not gap} \\ |r_i - q_{j-1}|, \text{if } q_j \text{ is a gap} \\ |r_{i-1} - q_j|, \text{if } r_i \text{ is a gap} \end{cases} \quad (2)$$

其中,$Rest(R) = \langle r_1, r_2, \cdots, r_{m-1} \rangle$, $Rest(Q) = \langle q_1, q_2, \cdots, q_{n-1} \rangle$。计算两序列的DTW距离值需要构造一个 $m \times n$ 的时间归整矩阵,利用递归公式(1)以行或列的顺序填充该矩阵,最后矩阵 (m, n) 元素中的值即为两序列的DTW值。通常,归整路径 P 可以描述为:$W = \langle p_1, p_2, \cdots, p_k, \cdots, p_L \rangle$,其中,$\max(m, n) \leq L \leq m + n - 1$ 且 p_1 为矩阵(1,1)位置,p_L 为矩阵(m, n)位置。显然,标准DTW算法的时间复杂度为 $O(mn)$,由于算法只需保存矩阵中的当前列和其前一列(或者当前行或其前一行),从而DTW算法的空间复杂度为 $O(\min(m, n))$。DTW计算的本质在于允许插入空值(gap),将两条不同长度的时间序列扩展成长度相等的序列,并定义实数值与空值之间的距离,以便实数值与空值作比较,从而实现算法对数据中的局部时间偏移和噪音的容忍。

2.2 问题定义

数据流上的子序列匹配问题就是在数据流 S 上快速实时监控与查询序列 Q 相似的子序列。数据流 S 可描述为半无限的数值序列 $s_1, s_2, \cdots, s_n, \cdots$,给定序列 $S = \langle s_1, s_2, \cdots, s_n \rangle$和 $Q = \langle q_1, q_2, \cdots, q_m \rangle$,$S[t_s, t_e]$表示从时刻 t_s 开始、t_e 结束的子序列,Q 为已知的查询序列。可将该问题分解为多个子问题:“最佳匹配”、“范围匹配”、“最优范围匹配”和“Top-K匹配”。

定义1 最佳匹配(Best-Match Query):假定序列 S 和 Q 的长度分别为 n 和 m,在所有的可能序列 $S[t, j]$中寻找与 Q 距离最小的子序列 $S[t_s, t_e]$,即 $D(S[t_s, t_e], Q) \leq D(S[t, j], Q)$,$t = 1, \cdots, n; j = t, \cdots, m$。

定义2 范围匹配(Range Query):给定查询序列 Q 和相似性阈值 ε,从数据流中查找出所有子序列,使其DTW距离满足 $D(Q, R) \leq \varepsilon$。

定义3 最优范围匹配(Optimal Range Query):给定数据流 S 在时刻 t 的长度为 n,查询序列 Q 的长度为 m,相似性阈值为 ε,输出满足如下要求的所有子序列 $S[t_s, t_e]$:

(1) $D(S[t_s, t_e], Q) \leq \varepsilon$;

(2)如果任意两子序列 $S[t_s, t_e]$和$(S[t_s', t_e']$都满足条件(1),则对应的两个时间区间$[t_s, t_e]$和$[t_s', t_e']$满足$[t_s, t_e] \cap [t_s', t_e'] = \varnothing$。

定义4 Top-K匹配(Top-K Query):在 S 上监控DTW距离最小的前 K 个最优范围匹配子序列。

由于流数据 S 具有无限不可预测的特性,在此情景下“最佳匹配”意义并不大,因为不能确保后续流数据中的子序列是否为最佳匹配的子序列,因此“范围匹配”更适合于数据流场景。不过值得注意的是,当查询序列 Q 与 S 的子序列匹配时,可能存在与“局部最小”的范围匹配子序列重叠的其它子序列。因此,在标准的“范围匹配”中,我们需要附加第二个条件,其目的是排除这些大量重叠的子序列。这些匹配有如下不良影响:(1)它们将潜在地给用户输出大量冗余的子序列;(2)将降低算法的速度,迫使算法追踪这些冗余的子序列。即便是排除了重叠子序列的最优范围匹配,由于数据流的潜在无限性,通常我们更关注的是最相似的 K 个子序列。由此,Top-K匹配问题是一个更具价值的研究问题,本文后续章节中仅讨论 Top-K匹配问题。

3 TopKSM算法

以往的研究主要集中在如何通过限制矩阵中元素的计算区域来提高最优路径的搜索效率,而文献[7]提出的SPRING算法是针对数据流上子序列的匹配。该文设计了一个STWM(Subsequence Time Warping Matrix,简称STWM)矩阵,该矩阵改进了原有的标准DTW计算矩阵,在每个矩阵元素中增加了一个起始位置,表明当前位置的子序列中与被匹配序列最相似序列的起始时刻。另一方面,STWM矩阵在每个新数据到达后只需计算矩阵中一列元素上的距离,因而每个新到的数据只需计算 m 次距离,其空间复杂度为 $O(m)$,与标准的DTW算法一致。STWM矩阵中的 $D(S[t_s, t_e], Q)$和 $sp(t, i)$ 计算公式如下(其中 $D(S[t_s, t_e], Q)$的起点位置为 $t_s = sp(t_e, m)$):

$$D(S[t_s:t_e], Q) = d(t_e, m) = \min(d(t, m))$$

$$d(t, i) = |S_t - Q_i| + d_{best}$$

$$d_{best} = \min(d(t, i-1), d(t-1, i), d(t-1, i-1)) \quad (3)$$

$$sp(t, i) = \begin{cases} sp(t, i-1), \text{if } d(t, i-1) = d_{best} \\ sp(t-1, i), \text{if } d(t-1, i) = d_{best} \\ sp(t-1, i-1), \text{if } d(t-1, i-1) = d_{best} \end{cases} \quad (4)$$

$$d(t, 0) = 0, d(0, i) = \infty (t = 1, \cdots, n; i = 1, \cdots, m)$$

分析上述STWM矩阵计算公式可得,SPRING算法主要的计算开销为STWM矩阵的计算,距离计算公式越复杂(如L2范数或者Lp范数),计算开销就越大,查询序列 Q 越长,计算开销也越大。该算法中,如果 $\forall i,\ d(t, i) \geq d_{\min} \vee sp(t, i) > t_e$,则输出匹配结果。直观上,如果 $\min(d(t, i-1), d(t-1, i-1), d(t-1, i)) > \varepsilon$,又 $dist(s_t, q_i) \geq 0$,那么可根据公式(2)计算出 $d(i, j) > \varepsilon$。因而,我们可以充分利用公式(2)并选取适当相似性阈值 ε 来减少许多计算量。上述思想可表述为以下定理:

定理1 如果 $\forall i, 1 \leq i \leq m, d(t-1, i) > \varepsilon$ 且 $d(t, j) > \varepsilon$,那么 $\forall k, j+1 \leq k \leq m, d(t, k) > \varepsilon$。

证明 假设 $k = j + 1$, $d(t, k) = d(t, j+1) = dist(s_t, q_{j+1}) + \min(d(t, j), d(t-1, j+1), d(t-1, j))$,$dist$ 是一个

距离计算函数(比如 Lp 范数), $\forall x, y, dist(x, y) \geq 0$。又已知 $d(t, j) > \varepsilon, 1 \leq j, j+1 \leq m$ 时, $d(t-1, j) > \varepsilon, d(t-1, j+1) > \varepsilon$,所以 $\min(d(t, j), d(t-1, j+1), d(t-1, j)) > \varepsilon$,因而 $d(t, k) > \varepsilon, k > j+1$ 时同理可证。□

定理 2 已知 $1 \leq h \leq m$,如果 $\forall i, h \leq i \leq m, d(t-1, i) > \varepsilon$ 且 $d(t, h) > \varepsilon$,那么 $\forall k, h+1 \leq k \leq m, d(t, k) > \varepsilon$。

证明 假设 $k = h+1$, $d(t, k) = d(t, h+1) = dist(s_t, q_{h+1}) + \min(d(t, h), d(t-1, h+1), d(t-1, h)), dist(s_t, q_{h+1}) \geq 0$,又已知 $d(t-1, h) > \varepsilon, d(t-1, h+1) > \varepsilon$,则 $\min(d(t, h), d(t-1, h+1), d(t-1, h)) > \varepsilon$,所以 $d(i, h+1) > \varepsilon$,同理可证 $h+2 \leq k \leq m$ 时 $d(t, k) > \varepsilon$。□

定理 3 已知 $1 \leq l \leq m, j \leq l$,如果 $\forall i, 1 \leq i \leq l, d(t-1, i) > \varepsilon, d(t-1, l) \leq \varepsilon$ 且 $d(t, j) > \varepsilon$,那么 $\forall k, j < k < l, d(t, k) > \varepsilon$。

证明 假设 $k = j+1, d(t, k) = d(t, j+1) = dist(s_t, q_{j+1}) + \min(d(t, j), d(t-1, j), d(t-1, j+1)), dist(s_t, q_{j+1}) \geq 0$,又已知 $d(t-1, j) > \varepsilon, d(t-1, j+1) > \varepsilon$ 且 $d(t, j) > \varepsilon$,则 $\min(d(t, j), d(t-1, j), d(t-1, j+1)) > \varepsilon$,所以 $d(t, j+1) > \varepsilon$,同理可证 $j+1 \leq k \leq l$ 时, $d(t, k) > \varepsilon$。□

有定理 1 作保证,可以减少大量的冗余计算,并能得到与 SPRING 算法相同的运行结果。在任意时刻 t,根据定理 2 和定理 3,只要记录上界(h)和下界(l),它们分别是距离矩阵的第 t 列中最后一个小于或者等于 ε 的位置和第一个小于或者等于 ε 的位置。

首先定义上下界的数据结构 struct Bound,然后给出 Top KSM 的算法描述。

```
struct Bound {
  unsigned long lowest;
  unsigned long highest;
};
Bound priBound,curBound;
```

其中, *priBound. lowest* 和 *priBound. highest* 分别表示 STWM 中前一列的第一个小于或者等于 ε 和最后一个小于或者等于 ε 的元素的位置,算法为其赋初值 0。Top KSM 算法详细介绍如下:

Top KSM 算法 Top KSM (s_t, ε, k)

输入: s_t 为 t 时刻数据流上的一个新数据,匹配阈值 ε, k 为最关注的子序列个数;

输出:满足要求的 K 个子序列 *SortedList*。

1 *curBound. lowest* = 0; *curBound. hightest* = 0; i = 1;

2 *SortedList*[1…k] ←nil,包含 k 个元素的数值都赋空值,其中保存 Top-K 个子序列。

3 while $i \leq m$ do

4 利用公式(3)和公式(4)以及定理 1,计算 $d(t, i)$, $sp(t, i)$;

5 if $d(t, i) > \varepsilon$ then

6 根据当前计算指示 i 处在 *priBound. lowest* 和 *priBound. highest* 所划分区间的位置,决定下一个计算单元的值,即变量 i 的值。

7 else

8 i + +;

9 if $d_{\min} \leq \varepsilon$ then

10 if $\forall i, d(t, i) \geq d_{\min} \vee sp(t, i) > t_e$ then

11 根据 $D([p_s, p_e], Q)$ 的值将子序列 $S[p_s, p_e]$ 与 *SortedList* 中最后一个元素进行比较,决定是否对 *SortedList*[k] 进行替换,如果 $D([p_s, p_e], Q)$ 小,则替换,并对 *SortedList* 按升序重新排序。

12 for i = 1 to m do

13 if $sp(t, i) \leq p_e$ then $d(t, i) = \infty$;

14 if $d(t, m) \leq \varepsilon \wedge d(t, m) < d_{\min}$ then

15 $d_{\min} = d(t, m)$; $p_s = sp(t, m)$; $p_e = t$;

16 $D^{t-1} = D^t$; $SP^{t-1} = SP^t$; *priBound* = *curBound*;

假设查询序列 Q 的长度为 m,在每一时刻 t 算法计算 D^t 中的值,在最坏情况下需计算 m 个元素的值。因此,算法在每新到一个流数据的情况下,最坏时间复杂度为 O(m),与 SPRING 算法相同。达到最坏时间复杂度的情况是:相似性阈值 ε 很大或者查询序列 Q 的长度为 1。但实际应用中,通常不会选择很大的相似性阈值 ε 或查询序列 Q 的长度为 1,因此 Top KSM 算法要比 SPRING 算法效率高。该算法仅增加 4 个上下界变量,因而其空间复杂度也和 SPRING 算法相同,都为 O(m)。

4 实验结果与分析

本节给出相关的实验结果与分析。我们的实验主要分为两部分:有效性测试和性能测试。测试环境为 P4 2 GHz、1 G 内存的 Windows XP,所有算法采用 Visual C++ 6.0 设计。测试数据集包括两个真实数据集,一个为文献[7]中使用到的 Sensor 数据集,另一个为股票数据集。从上述分析可知,影响 Top KSM 算法效率的主要因素有维数(d)、相似性阈值(ε)、查询效率长度(m)和参数 k。有效性测试采用文献[10]中的 Sensor 数据集(该数据集中包括温度、湿度、光照强度和电压数据等四项,本测试中仅选择温度数据项)。表 1 为 Top KSM 算法找到的四个匹配的子序列。

表 1 Top KSM 算法查找出来的匹配的子序列

数据集	查询序列		匹配序列		
	长度	ε	起点	终点	距离
温度	100	20	889	993	18.346
			2 400	2 493	4.089
			14 231	15 228	12.694
			23 056	23 170	8.466

性能测试中,采用股票数据集将 Top KSM 算法与 SPRING 算法进行比较。该股票数据集包括最高价、最低价、成交量和均价。图 1 显示了查询序列的长度对算法性能的影响,SPRING 算法的查询时间随着查询序列的长度的增长成线性上升,而 Top KSM 算法基本上与查询序列的长度没太大影响,性能保持得相对稳定。

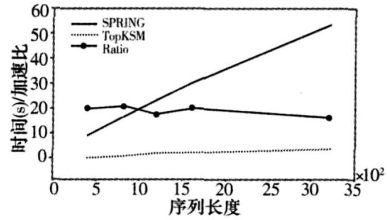

图 1 查询序列长度对性能的影响

表 2 列出了其它几个参数对算法的影响结果。该实验表明,相似性阈值(ε)对 Top KSM 算法的性能影响较大。这也符合该算法的设计原理,通常在实际的数据流应用中,对参数 ε 不会取得特别大,从而 Top KSM 算法总体上能保持至少 15 倍以上的性能提高,甚至可高达 40 多倍。参数 k 对算法效率影响不是很明显,因为两算法都是先计算出"最优范围匹配"的子序列,再将其保存到 *SortedList* 中,由于参数 k 相对较小,因此数据在 *SortedList* 中处理的开销也非常小。最后,与图 1 展示的结果相似,查询序列的长度 m 对 SPRING 算法具有很大的影响,但对 Top KSM 算法影

响很小,因此两算法的加速比一直保持比较高。大量的实验表明,Top KSM 算法具有更高的性能。

表 2　三个参数对性能的影响

k	ε	m	SPRING(s)	Top KSM (s)	加速比
10	60	500	1.312	0.036	36.444
40	60	500	1.463	0.052	28.135
10	100	500	1.524	0.063	24.190
40	100	500	1.836	0.067	27.403
10	150	1 000	2.975	0.069	43.116
40	150	1 000	3.527	0.091	38.758
10	200	1 000	3.878	0.161	24.087
40	200	1 000	4.101	0.218	18.812

5　结束语

本文研究数据流上快速的 Top- K 子序列匹配问题,设计了一种新颖的界限策略,极大地减少了 STWM 矩阵中的冗余计算。理论分析和实验表明,Top KSM 算法在不损失任何算法精度的前提下,仅增加几个字节的空间开销,算法性能提高极为显著。该算法更适合于海量大规模数据流的子序列匹配。

参考文献:

[1] Agrawal R, Faloutsos C, Swami A N. Efficient Similarity Search in Sequence Databases[C] //Proc of FODO'93, 1993:69-84.

[2] Berndt D J, Clifford J. Using Dynamic Time Warping to Find Patterns in Time Series[C] //Proc of KDD'94 Workshop, 1994:359-370.

[3] 翁颖钧,朱仲英. 基于动态时间弯曲的时序数据聚类算法的研究[J]. 计算机仿真, 2004, 21(3):37-41.

[4] Chen S-C, Kashyap R L. A Spatio Temporal Semantic Model for Multimedia Presentations and Multimedia Database Systems[J]. IEEE Trans on Knowledge and Data Engineering, 2001, 13(4):607-622.

[5] 安镇宙,杨鉴. 一种新的基于并行分段裁剪的 DTW 算法[J]. 计算机工程与应用, 2007, 43(15):35-38.

[6] 陈当阳,贾素玲. 时态数据的趋势序列分析及其子序列匹配算法研究[J]. 计算机研究与发展, 2007, 44(3):516-520.

[7] Kim S-W, Park S, Chu W W. An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases[C] //Proc of ICDE'01, 2001:607-614.

[8] Keogh E J. Exact Indexing of Dynamic Time Warping[C] // Proc of VLDB'02, 2002:406-417.

[9] Zhou M, Wong M H. Boundary-Based Lower-Bound Functions for Dynamic Time Warping and Their Indexing[C] // Proc of ICDE'07, 2007:1307-1311.

[10] Sakurai Y, Faloutsos C, Yamam M. Stream Monitoring Under Time Warping Distance[C] //Proc of ICDE'07, 2007:1046-1055.

(上接第 54 页)

$$R\% = \frac{n_1}{N \times M} \times 100\%$$

其中,n_1 是正确分割的像素数。边缘准确性指分割结果边缘与原始图像纹理边缘的吻合程度,以及边缘邻近区域分割结果的准确性;区域一致性则指原始图像中纹理特性相对一致的区域在分割结果中是否呈现为统一的区域。本文的分割效果和分割正确率分别如图 1 和表 1 所示。由此可知,本文提出的算法在分割正确率、边缘准确性和区域一致性方面,均比 K 均值算法有较明显的改善。

表 1　两种方法分割正确率比较　%

算法	纹理图 1	纹理图 2
K 均值算法	92.37	90.18
本文算法	95.61	92.75

6　结束语

本文提出了一种基于小波变换和蚁群算法的纹理分割新方法。为了获得更好的分割效果,我们采用了小波包分析提取纹理的特征。更重要的是提出了具有模糊聚类能力的蚁群算法模型。实验证明,将蚁群算法应用到纹理分割领域是一次有效的尝试。

参考文献:

[1] Jain A K, Farrokhnia F. Unsupervised Texture Segmentation using Gabor Filters[J]. Pattern Recognition, 1991, 23(12): 1167-1186.

[2] Wang Jingwen. Multiwavelet Packet Transforms with Application to Texture Segmentation[J]. Electronics Letters, 2002, 38(18):1021-1023.

[3] Unser M. Texture Classification and Segmentation Using Wavelet Frames[J]. IEEE Trans on Image Processing, 1995, 4(11):1549-1560.

[4] Zeng Xiang-yan, Chen Yen-wei, Nakao Z, et al. A New Texture Feature Based on PCA Pattern Maps and Its Application to Image Retrieval[J]. IEICE Trans on Information and System, 2003, 80(5):929-936.

[5] Song A, Ciesielski V. Texture Analysis by Genetic Programming[C] //Proc of IEEE Congress on Evolutionary Computation, 2004:2092-2099.

[6] Dorigo M, Maniezzo V, Colorni A. Ant System: Optimization by a Colony of Cooperating Agents[J]. IEEE Trans on Systems, 1996, 26(1):29-41.

[7] Dorigo M, Caro G D, Gambardella L M. Ant Algorithms for Discrete Optimization [J]. Artificial Life, 1999, 5(2):137-172.

[8] 韩彦芳, 施鹏飞. 基于蚁群算法的图像分割方法[J]. 计算机工程与应用, 2004, 18(5):5-7.

[9] 刘传才,杨静宇. 一种新的图像纹理表示方法[J]. 计算机学报, 2001, 24(11):1202-1209.

[10] 李波,覃征,石美红. 利用小波变换和 FCM 算法进行多特征纹理分割[J]. 计算机工程, 2005, 31(24):148-150.

[11] Li Chunmao, Wang Lingzhi, Wu Shunjun. Ant Colony Fuzzy Clustering Algorithm Applied to SAR Image Segmentation[C] //Proc of the Int'l Conf on Radar, 2006:1-4.

[12] Haralick R M, Shanmugam K, Dinstein I. Textural Features for Image Classification[J]. IEEE Trans on System, Man and Cybernetics, 1973, 3(6):610-621.

[13] Ng L S, Nixon M S, Carter J N. Texture Classification Using Combined Feature Sets[C] //Proc of the IEEE Southwest Symp on Image Analysis and Interpretation, 1998: 103-108.