Journal of Computer Applications
计算机应用,2013,33(2):563 -566
ISSN 1001-9081
CODEN JYIIDU
2013-02-01
http://www.joca.cn

文章编号:1001 -9081(2013)02 -0563 -04
doi:10.3724/SP.J.1087.2013.00563

一种具有自适应机制的闪存数据库索引结构

房俊华[1*],王翰虎[1,2],陈 梅[1],马 丹[1]
(1.贵州大学 计算机科学与信息学院,贵阳 550025; 2.贵州星辰科技开发有限公司,贵阳 550001)
(*通信作者电子邮箱 fangjunhua26@163.com)

摘 要:针对闪存数据库系统索引技术中基于日志更新策略存在的检索效率低、日志空间分配不合理及合并带来的高昂更新代价等问题,提出一种具有自适应机制的索引结构 LM-B+TREE。LM-B+TREE 将索引的更新缓冲页映射于传统 B+TREE 的相应节点,并根据闪存索引的读写负载及读写代价差异,动态地分配缓冲更新区,自适应地调整索引架构。实验证明 LM-B+TREE 能够动态地调整索引架构来适应索引的读写负载代价,在减少索引更新代价的同时,有效地提高了索引的查询性能。

关键词:闪存数据库;索引结构;缓冲更新;自适应机制;代价评估
中图分类号:TP311.13 **文献标志码**:A

Index structure with self-adaptive mechanism in flash-based database system

FANG Junhua[1*], WANG Hanhu[1,2], CHEN Mei[1], MA Dan[1]
(1. *College of Computer Science and Information, Guizhou University, Guiyang Guizhou 550025, China*;
2. *Guizhou Stars Technology Development Company Limited, Guiyang Guizhou 550001, China*)

Abstract: The log-based index update mechanism in flash-based database system has following shortage: low query efficiency, expensive update cost, unreasonable space allocation and merge for the log. In order to solve these problems, a new adaptive index structure named LM-B+TREE was proposed. LM-B+TREE can map the page for index update buffer into corresponding node of traditional B+ TREE. Furthermore, according to the read/write workload and read/write overhead, LM-B+TREE can dynamically maintain the update buffer and adjust the index frame adaptively. The experimental results show that LM-B+ TREE can dynamically adjust the index structure to adapt to the read-write workload, significantly reduce the overhead of index update and improve the query performance.

Key words: flash-based database; index structure; delayed update; self-adaptive mechanism; cost estimate

0 引言

近年来,随着闪存容量的不断攀升,它快速、节能、抗震等优势不管对于小型区域管理服务提供商(Management Service Provider,MSP)还是全球云基础架构即服务(Infrastructure As a Service,IAAS)提供商无疑都是一个福音。数据库管理系统是被人们所熟知的一种有效的数据管理工具,虽然传统的磁盘介质已被设计成为成熟的存储系统,但由于闪存固有的特性限制,这些技术却不能直接用在基于闪存介质的数据库上[1-2]。其中的一个重要的原因就是传统的索引结构直接用在闪存上会引起高昂的更新代价[3]。

B+树是众所周知的高效索引技术。Intel 将 B+树应用到闪存中提出了将闪存物理地址映射为逻辑地址的闪存转换层(Flash Translation Layer,FTL)方法[4],基于 FTL 的改进方法[5-6]使闪存提供了和磁盘相同的 I/O 接口,但在需要维护大量的映射表的同时导致更新操作引发大量的写闪存操作;日志更新[7]采取异地写入的方式更新索引记录,近年来,基于日志更新的改进方法[8-10]有效地降低了索引的更新代价,然而,新增的日志区却限制着索引的检索性能。

将 B+树应用到闪存数据库中,关键问题是要针对闪存特性,在兼顾索引级联更新[11]的同时提高索引的查询性能。本文基于传统 B+树读写操作的空间聚集性[12],根据闪存特性及页面读写负载情况,提出了一种更高性能的索引结构(Lower Merger & Higher Retrieval Base on B+ TREE,LM-B+TREE)。LM-B+TREE 从降低索引更新代价、提高索引检索性能的角度出发,针对 B+树各个层的不同节点,引入用于存储预读和预更新的预存页;提出了索引读写代价开销评估模型,引入预存页数量及合并的自适应机制来均衡索引的读写性能。

1 LM-B+TREE 索引

1.1 闪存数据库索引

闪存是一种非易失性的存储介质,较磁盘具有更快的读取速度。此外,和磁盘相比其还具有自身的物理特性:读写速度不同、不可覆盖写及磨损平衡等[13-14]。针对闪存的特性,基于日志更新的纯闪存索引技术直接根据闪存物理地址进行数据访问,决定了它具有更快的存取速度[15]。为延续 B+树索引结构的高性能,研究者们提出许多改进的结构,包括将闪存块的最后一个页作为延迟更新的日志页;在内存中维护一棵日志树来减少写的代价;将闪存块划分为两部分,通过组提交、更新合并及多级延迟来提高更新性能等。

然而,上述基于日志更新的索引技术,日志区是孤立于索

收稿日期:2012-08-20;**修回日期**:2012-10-04。 **基金项目**:贵阳市 2010 年工业科技攻关项目([2010]筑科工合同字第 28 号)。
作者简介:房俊华(1985 -),男,河南周口人,硕士研究生,主要研究方向:闪存数据库系统; 王翰虎(1946 -),男,贵州遵义人,教授,CCF 高级会员,主要研究方向:数据库系统、分布式系统; 陈梅(1964 -),女,贵州贵阳人,教授,主要研究方向:软件工程; 马丹(1977 -),女,贵州贵阳人,副教授,主要研究方向:多媒体数据库。

引结构的定值大小,这样的区域在索引的合并更新及检索方面都存在着缺陷。与以往的工作相比,本文提出的 LM-B + TREE 索引结构,根据闪存页读写负载情况,将索引的缓冲操作逻辑映射于 B + 树特定节点,同时在系统运行过程中,LM-B + TREE 根据索引影响因子的状况,动态调整预存页大小及合并时机。

1.2 LM-B + TREE 索引结构

如图 1 所示,LM-B + TREE 由一棵 B + 树和若干预存页构成,索引节点的更新首先根据页面负载代价,判定是否更新在预存页中,若是,预存页未满时直接写入预存页,如果预存页满,迭代更新至下一级预存页,直至更新到叶子节点的父节点;若更新至缓存页产生较大代价,则合并更新至 B + 树索引。查询时,LM-B + TREE 首先根据关键字查找相应的预存页。

预存页与日志页的不同在于:1)数量上,LM-B + TREE 根据 B + 树的各个层节点读写的负载情况,动态划分预存页对应的节点数;2)存储内容上,日志页仅仅缓冲了对索引更新操作的日志记录,预存页考虑到闪存读写粒度特性,使用经典的最近最少使用(Least Recently Used,LRU)算法,在存储了更新记录的同时存放了经常使用的索引记录,更新记录具有更高的存储级别;3)写缓存页时,插入和删除直接写入预存页,更新若出现范围跨度时则需先删除后写入更新对应的预存页内容。

如图 1 所示,一个 3 阶的 LM-B + TREE,a、b 为经分裂后无效的预存页,cd、ef、g 为现存有效预存页,其中预存页 ef、g 分别对应节点 B 的子节点 E、F 和 G,cd 对应 C、D。假设 1 个块 15 页,对于传统的索引结构,检索操作需提前扫描整个日志区,因此检索至少需要进行 6 次读操作,LM-B + TREE 检索操作最少时可只检索 B + 树(3 次),即使在需要扫描预存页的情况下,LM-B + TREE 此时也只需额外读取与之相关的一个缓存页(C、D 读取 cd,E、F 读取 ef,G 读取 g);初始状态只有 a 缓存了整个索引的更新,更新操作使预存页 a 满时,若合并,a 写入会引起甚至包括所有节点重写的高合并代价,故选择分裂为 b、cd。如图 2 所示,预存页不仅缓存了对索引节点的更新,同时也缓存了检索频率高的预读索引记录。

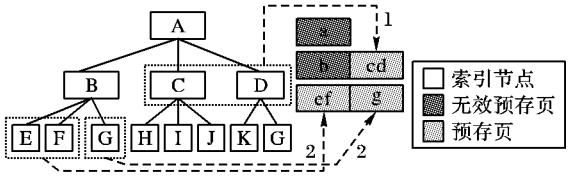

图 1 LM-B + TREE 整体架构

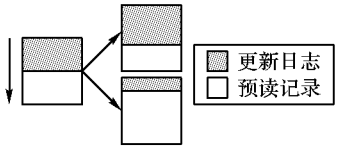

图 2 预存页结构

2 LM-B + TREE 的关键算法设计

LM-B + TREE 使用预存页缓冲索引更新记录和频繁使用的索引记录,根据闪存页的读写负载,通过比较分裂预存页与合并操作代价,自适应地分配预存页的数量。

2.1 相关概念定义

索引操作的请求到达时,LM-B + TREE 首先估算该操作直接更新至索引节点与缓冲在预存页的代价关系。关系可表示为

$$Z = (BUFFER\_C(Y) + R_{BUFFER\_C(Y)}) - (MERGE\_C(Y) + R_{MERGE\_C(Y)}) \quad (1)$$

当 $Z \geqslant 0$ 时,合并;当 $Z < 0$ 时,缓存。表 1 是算法描述中要使用的符号含义。

表 1 各个变量定义

符号	含义
C_W	写闪存页时间
C_R	读闪存页时间
F_R	对节点读操作概率
F_W	对节点写操作概率
Y	一个预存页对应的索引节点个数
F_C	当前操作记录在预读页中的概率
$BUFFER\_C(Y)$	预存页有 Y 个索引节点,未合并代价
F_F	预存页分裂概率,即预存页是否写满
$MERGE\_C(Y)$	预存页有 Y 个索引节点,合并代价
F_M	合并时 B + 树索引架构调整的概率
F_S	预存页存储记录与索引节点的相关系数
T_M	B + 树索引架构调整时牵涉页的相关系数
$R_{(OC)}$	经过(OC)操作代价后,索引检索代价
Z	总操作代价

2.2 索引操作开销评估

不合理的索引扇出度会浪费存储空间,LM-B + TREE 以闪存页为非叶节点,若干个页为叶子节点。假设闪存页大小为 2 KB,一个非叶子索引为 8 B,它可以存储 256 个这样的索引项,叶节点存储了数据记录及指向下个节点的指针,若数据记录平均 50 B,一般 5 个闪存页作为一个叶子节点。不合并情况下:对索引的操作有只读和更新两种情况。当只读时,操作将会首先读取预存页,命中即结束,未命中则读取预存页对应的相关 B + 树索引节点,其操作代价表示为

$$BUFFER\_C(Y)_R - C_R + (1 - F_C) * C_R * Y \quad (2)$$

当操作为更新操作时,LM-B + TREE 需判断此更新操作写入预存页是否会导致预存页的分裂,操作代价为

$$BUFFER\_C(Y)_U = C_W * (1 - F_F) + 2 * F_F * C_R = C_W * (1 + F_F) + C_R \quad (3)$$

综合式(2)、(3),不合并时索引操作的总代价为

$$BUFFER\_C(Y) = F_R * BUFFER\_C(Y)_R + F_F * BUFFER\_C(Y)_U = C_R + (1 - F_C) * C_R * Y + F_R * (1 - F_C) * C_R * Y \quad (4)$$

合并情况下:若将预读页立即合并到 B + 树索引,需考虑合并过程 B + 树索引是否发生自身架构的调整,若合并时 B + 树架构不发生调整,操作代价为

$$MERGE\_C(Y)_W = C_R + C_R * Y + Y * C_W * (1 + F_S) \quad (5)$$

若发生调整,操作代价为

$$MERGE\_C(Y)_T = (T_M * Y * F_S) * C_R + T_M * Y * F_S(1 + F_S) * C_W \quad (6)$$

综合式(5)、(6),合并时索引操作的总代价为

$$MERGE\_C(Y) = (1 - F_M) * MERGE\_C(Y)_W + F_M * MERGE\_C(Y)_T \quad (7)$$

2.3 自适应机制

为兼顾索引的读写代价,LM-B + TREE 动态地分配预存

页的数量,自适应地选择预存页的合并。预存页写入通过 B + 树索引架构是否调整来评估:

写操作且未调整:将式(4)、(5)代入式(1)化简得

$$Z = C_W + C_R + (C_W + C_R - Y * F_S * C_W) * F_F + [(-1 - F_C) * C_R - C_W] * Y * F_S \quad (8)$$

写操作且调整:将式(4)、(6)代入式(1),化简得

$$Z = C_W + (C_W + C_R - Y * F_S * C_W) * F_F + [(1 - F_C) * C_R - (C_R + C_W) * T_M * C_R] * Y * F_S \quad (9)$$

由于闪存的读写速度 C_R、C_W 为定值,预存页内记录的命中概率采用 LRU,在此不予考虑,式(9)、(10)可以表示为:

$$Z = A + (A - B * Y * F_S) * F_F + P * (A - B) * Y * F_S = G(F_F, Y * F_S) \quad (10)$$

$$Z = B + (A - B * Y * F_S) * F_F - [P' * (A - B) - A * (A - B) * T_M] * Y * F_S = H(F_F, Y * F_S, T_M) \quad (11)$$

由式(10)、(11)可知,索引架构的调整及预存页的大小划分与索引页是否分裂、预存页存储映射节点个数、与节点记录相关系数及原索引架构调整所需要调整的页数相关。在索引的运行过程中,LM-B + TREE 根据索引架构的情况,动态地选择代价最小的操作方式指导索引的运行,操作过程如 2.4 节算法。

2.4　索引操作算法

对索引的操作可分为查询与更新,算法 1 描述了 LM-B + TREE 的操作流程。

算法 1

```
LM_BPLUSTREE (operator, K)
    /* operator 表示请求操作类型,K 表示请求操作关键字 */
1)  If (operator 是查询操作)
2)    If (K 有相关预存页)                          //读取预读页
3)      If(预存页命中) 返回索引记录
        Else 读取该关键字节点 返回结果
        End if
      Else 读取该关键字节点 返回结果
      End if
    End if
5)  If (operator 是更新操作)
6)    If(K 有预存页 && 未满)排序写入预存页
7)    Else If ((BUFFER_C(Y) + R_BUFFER_C(Y)) -
        (MERGE_C(Y) + R_MERGE_C(Y)) ≥ 0)  //比较负载代价
8)      For each [该节点的预存页]                  //合并操作
          {读所属关键字的预存页更新至其子节点
          If (B + 树发生调整)
            将架构调整牵涉到的节点预存页合并
          End if
          If ((BUFFER_C(Y') + R_BUFFER_C(Y')) -
            (MERGE_C(Y') + R_MERGE_C(Y')) > 0)
            B + 树层 + 1
          End if}
9)    Else 分裂预存页 写入更新记录 更新映射
      End if
    End if
```

3　实验结果和分析

通过实验来验证提出索引结构的性能,实验中操作系统为 Windows XP,编程环境是 VC2008,编程语言是C ++。计算机硬件配置为:Intel 2.8 GHz CPU,2 GB 内存。闪存页的大小为 2 KB,块由 64 个页组成,芯片的读、写、擦除延迟分别为 40 μs、200 μs 和 1.5 ms。测试中的数据源来自 TPC-C 基准测试,通过改变 TPC-C 的参数以获得不同的查询/更新频率。

图 3 显示了通过三种类型的索引分别进行 30 万、50 万、70 万次随机更新操作的平均更新时间代价。实验结果显示 BFTL 的平均更新时间代价最大,BFTL 虽通过逻辑映射提供了与磁盘相同的接口但造成大量的随机写操作。在三次随机更新的过程中,BFTL 分别写闪存 231 356、352 715 和 513 982 次,IPL B + TREE 分别写 23 856、40 139 和 56 128 次,而 LM-B + TREE只需要写 6 626、10 276 和 15 799 次。相对于 IPL B + TREE,LM-B + TREE 考虑了节点分裂及索引结构调整带来的更新代价。

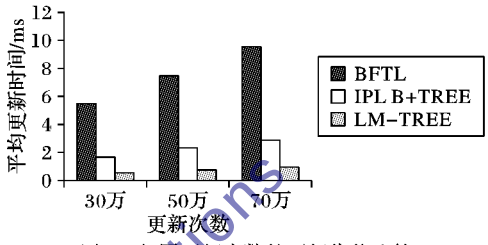

图 3　相同更新次数的更新代价比较

闪存数据库中,不同更新比率的操作反映着系统的实用性。图 4、5 是数据库进行 50 万次不同更新比率情况下,索引的平均更新及平均更新读取时间代价。相对于 BFTL 和 IPL B + TREE,LM-B + TREE 的性能更稳定,这直接得益于 LM-B + TREE架构及自适应模式。LM-B + TREE 将用于缓冲更新的预存页动态地映射到索引各个层的不同节点上,通过自适应机制兼顾了索引结构的变化和预存页的大小,同时页内使用 LRU 算法预存频繁使用的索引记录,这在降低更新代价的同时,提高了索引的检索性能。

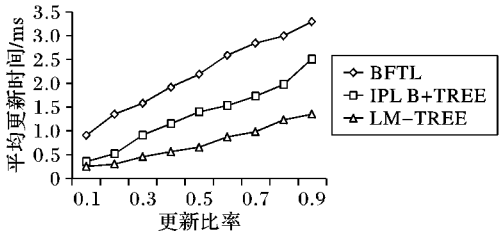

图 4　不同更新比率下的更新代价比较

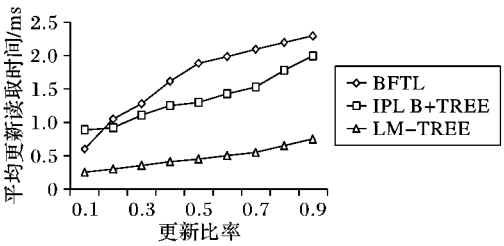

图 5　不同更新比率下的更新和读取代价比较

图 6 给出了用 postmark 数据集测试三个索引分别进行 30 万、50 万、70 万次的操作时闪存介质的擦除次数,本文将产生一个垃圾块视为一次擦除操作。结果表明LM-B + TREE 使索引根据影响因子(F_F、F_M 及 T_M)动态地选择小代价模式操作,从而减少垃圾页的产生,延长了闪存的寿命。

范围查询是索引性能的重要体现。图 7 是存储着 100 万条数据记录的闪存数据库进行 50 万次不同范围跨度的范围查询的实验结果。实验结果表明定值查询需提前扫面日志区的 IPL B + TRRR 在范围查询时更可能反复地扫面日志区,导

致其性能最差;LM-B + TREE 将预存页逻辑地映射于与之相关的索引节点,使查询有针对性地读取预存页,同时在预存页内存储着经常使用的索引记录,这也在一定程度上降低了检索的代价。

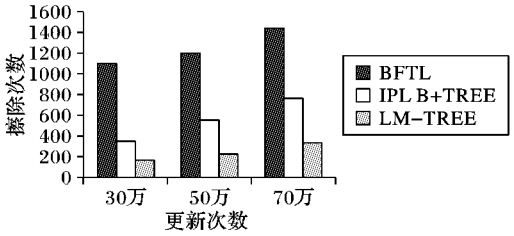

图 6 闪存擦除次数比较

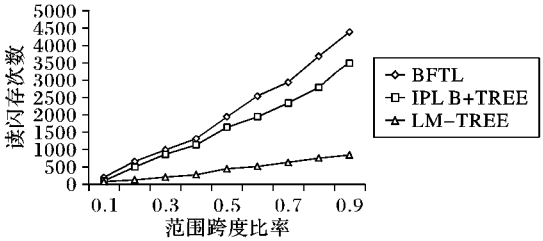

图 7 范围查询实验结果

4 结语

闪存作为一种新型的存储设备,相比磁盘具有很大的性能优势,然而它的一些特性也制约着索引性能。本文基于传统 B + 树操作的空间聚集性,根据闪存特性及页面读写负载情况,提出了一种更高性能的索引结构 LM-B + TREE。它动态地分配缓冲更新区,解决了不合理的日志空间带来的索引检索性能下降问题;根据闪存索引读写负载及读写代价差异,考虑索引架构调整,引入索引操作代价评估模型,使缓冲区的大小及合并自适应于闪存索引的代价开销,从而有效地提高索引的查询性能,减少索引的更新代价。下一步工作将对索引操作代价的评估模型加以完善,进一步明确各影响因子对其操作代价的影响。

参考文献:

[1] BAI S, LIAO X F. A parallel flash translation layer based on page group-block hybrid-mapping method[J]. IEEE Transactions on Consumer Electronics, 2012, 58(2): 441 – 449.

[2] 王立, 王跃清, 王翰虎, 等. 闪存数据库系统中一种高效的自适应存储模式[J]. 计算机应用, 2011, 31(5): 1400 – 1403.

[3] ROH H, PARK S, KIM S, *et al*. B + -tree index optimization by exploiting internal parallelism of flash-based solid state drives[J]. Proceedings of the VLDB Endowment, 2011, 5(4): 286 – 297.

[4] Intel-Corporation. Understanding the Flash Translation Layer (FTL) specifications [EB/0L]. [2012 – 06 – 26]. http://www.cse.ust.hk/~yjrobin/reading_list.html.

[5] WU C-H, KUO T-W, CHANG L P. An efficient B-tree layer implementation for flash-memory storage systems [J]. ACM Transactions on Embedded Computing Systems, 2007, 6(3): Article No. 19.

[6] TSAI Y-L, HSIEH J-W, KUO T-W. Configurable NAND flash translation layer [C]// Proceedings of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing. Piscataway: IEEE, 2006, 1: 118 – 127.

[7] LEE S-W, MOON B. Design of flash-based DBMS: an in-page logging approach [C]// SIGMOD'07: Proceeding of the ACM SIGMOD International Conference on Management of Data. New York: ACM, 2007: 55 – 66.

[8] BITYUTSKIY A B. JFFS3 design issues [EB/OL]. (2005 – 10 – 24) [2012 – 05 – 15]. http://www.linux-mtd.infradead.org/doc/JFFS3design.pdf.

[9] KIM G-J, BACK S-C, LEE H-S, *et al*. LGeDBMS: a small DBMS for embedded system with flash memory [C]// VLDB'06: Proceeding of the 32nd International Conference on Very Large Databases. New York: ACM, 2006: 1255 – 1258.

[10] 周大, 梁智超, 孟小峰. HF-Tree: 一种闪存数据库的高更新性能索引结构[J]. 计算机研究与发展, 2010, 47(5): 832 – 840.

[11] BYUN S, HUH M, HWANG H. An index rewriting scheme using compression for flash memory database systems [J]. Journal of Information Science, 2007, 33(4): 398 – 415.

[12] XU C, SHOU L D, CHEN G, *et al*. Update migration: an efficient B + tree for flash storage [C]// DASFAA'10: Proceedings of the 15th International Conference on Database Systems for Advanced Applications. Berlin: Springer-Verlag, 2010: 276 – 290.

[13] RYE Y. A flash translation layer for NAND flash-based multimedia storage devices [J]. IEEE Transactions on Multimedia, 2011, 13(3): 563 – 572.

[14] LEE S-W, MOON B, PARK C, *et al*. A case for flash memory SSD in enterprise database applications [C]// SIGMOD'08: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2008: 1075 – 1086.

[15] 孟小峰, 金培权, 曹巍, 等. 闪存数据库研究进展及发展趋势[J]. 中国科学基金: 学科进展与展望, 2012(3): 142 – 145.

(上接第 546 页)

[8] NIEMEIJER M, van GINNEKEN B, STAAL J, *et al*. Automatic detection of red lesions in digital color fundus photographs [J]. IEEE Transactions on Medical Imaging, 2005, 24(5): 584 – 592.

[9] QUELLEC G, LAMARD M, JOSSELIN P M, *et al*. Optimal wavelet transform for the detection of microaneurysms in retina photographs [J]. IEEE Transactions on Medical Imaging, 2008, 27(9): 1230 – 1241.

[10] ZHANG B, WU X, YOU J, *et al*. Hierarchical detection of red lesions in retinal images by multiscale correlation filtering [C]// Medical Imaging 2009: Computer-Aided Diagnosis, SPIE 7260. Lake Buena Vista, FL: SPIE, 2009: 72601L.

[11] MIZUTANI A, MURAMATSU C, HATANAKAY, *et al*. Automated microaneurysm detection method based on double ring filter in retinal fundus images [C]// Medical Imaging 2009: Computer-Aided Diagnosis, SPIE 7260. Lake Buena Vista, FL: SPIE, 72601N.

[12] HANSEN L K, SALAMON P. Neural network ensembles [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990, 12(10): 993 – 1001.

[13] 张东波, 尚星宇. 病变视网膜图像的血管骨架提取方法研究[J]. 电子测量与仪器学报, 2011, 25(9): 749 – 755.

[14] SPENCER T, OLSON J A, McHARDY K C, *et al*. An image-processing strategy for the segmentation of microaneurysms in fluorescein angiograms of the ocular fundus [J]. Computers and Biomedical Research, 1996, 29(4): 284 – 302.

[15] NIEMEIJER M, GINNEKEN B, CREE M J, *et al*. Retinopathy online challenge: automatic detection of microaneurysms in digital color fundus photographs [J]. IEEE Transactions on Medical Imaging, 2010, 29(1): 185 – 195.