

## 基于自然语言处理的通用信息模型自动调试

项 炜<sup>1,2\*</sup>

(1. 乐山师范学院 计算机科学学院, 四川 乐山 614000; 2. 乐山师范学院 智能信息处理及应用实验室, 四川 乐山 614000)

(\* 通信作者电子邮箱 Lstcxw@163.com)

**摘 要:**通用信息模型(CIM)是工业界的一种公开标准,并已实现于很多产品中,大量的 bug 被发现和修复。为了减少人工查找错误根源所需的时间和精力,提出一种基于自然语言处理的方法对 CIM 的 bug 进行自动调试。首先使用最大熵模型对已解决 bug 的文档描述进行分词,然后基于构建的词典使用 simHash 找出那些重复性很大的已修复的 bug,最后使用文档处理的方法分析客户提供的 trace 找出问题所在和解决方法。实验结果取得了 87.5% 准确率,表明了该方法的有效性。

**关键词:**通用信息模型;自然语言处理;最大熵模型;调试;文档处理

**中图分类号:** TP391 **文献标志码:** A

### Automated debug for common information model defect using natural language processing algorithm

XIANG Wei<sup>1,2\*</sup>

(1. School of Computer Science, Leshan Normal University, Leshan Sichuan 614000, China;

2. Laboratory of Intelligent Information Processing and Application, Leshan Normal University, Leshan Sichuan 614000, China)

**Abstract:** Common Information Model (CIM) is an open industrial standard, which has been implemented in products of many companies. Meanwhile, there are lots of bugs being reported and fixed. In order to reduce the cost time and effort of finding the root cause, in this paper, a method to debug automatically was proposed based on natural language processing algorithm. It firstly segmented those sentences using maximum entropy model, then used simHash to find the most similar fixed bug based on specifically constructed dictionary, finally used text mining to find the root cause and solution via analyzing the trace provided by customer. The experimental result achieves 87.5% accuracy, which shows its effectiveness.

**Key words:** Common Information Model (CIM); natural language processing; maximum entropy model; debug; text processing

## 0 引言

随着计算机软件产品的复杂性越来越高,开发者用于发现 bug 的时间和精力也越来越多,所需的知识要求也越来越全面,这也使得人工发现和解决 bug 越来越不切实际。随着对产品的要求越来越高,调试已经成为软件开发中一个非常重要的部分,研究表明调试花费了程序员大量的时间。传统上的调试是一种人工过程,并且这个过程非常繁琐和具有挑战性。例如程序员为了查找错误根源去获取执行的状态,但是状态可能有很多种,程序员很难人工去确定。所以自动调试就显得越来越有意义,但是完全自动化也是不可能的,所以希望尽可能利用已有的信息来排除那些不相关的代码。这样程序员就可以集中精力去查找那些可能导致问题的关键代码,大大节省了程序员所需的时间和精力。

事实上,近些年越来越多的研究也开始关注自动调试,例如一些基于统计的方法<sup>[1-4]</sup>,该方法首先使用断言采样机制去部署软件和收集用户执行的相关信息,然后使用统计度量去发现那些与 bug 相关的断言。另外一些方法<sup>[5-9]</sup>通过收集代码运行时的控制流、数据流以及程序状态信息,然后使用统计分析的方法发现那些可疑代码。但是上面的这些方法的准确性与事先定义的由数据驱动模型非常相关,并且限制了处理的数据量大小,另外使用复杂模型很容易导致扩展性问

题。以上的方法只是从代码中去发现问题,并没有充分利用已有的一些信息,例如以前所解决的 bug 的信息。另外大型软件都会产生大量的 trace,程序员逐行查看这些文件是非常耗时耗力的,如何从这些 trace 中快速发现那些 bug 的错误根源需要更高效的手段。

本文提出用基于自然语言处理的方法对通用信息模型(Common Information Model, CIM)的 bug 进行自动调试,主要基于以下几个原因:1)很多 CIM 产品被公司所开发,并且已经使用了很长的时间,已经有很多的 bug 被开发者所解决,并且这些记录也已经被保存。但现实中仍然有很多已经被解决的 bug 被不同的客户所上报(他们产品中的 CIM 版本不同,可能前面已修复的 bug 没有更新到现在客户的 CIM),只是客户针对问题的描述有所不同,导致同样的 bug 要经过同一重复的流程,即经过不同版本的技术支持,而且这些流程很费时间,最终要由开发者去找出根本原因,严重影响了解决问题的效率。图1显示了数据中发现的客户上报的3个bug,本质上它们是一个相同的 bug,但是被不同的客户使用不同的描述上报。如何避免同样的 bug 被不同的客户重复上报是一个急需解决的问题,该问题的解决能使企业减少很多不必要的流程。2)当开发者对客户提交的 bug 进行人工查找问题时,主要通过分析客户收集的大量 trace 和一些相关的信息,但是这些信息是非常复杂的,如何高效地使用文档处理的方法从这

些信息中找出错误根源是非常有用的。为了尝试解决前面所提到的两个问题,本文首先使用最大熵模型(Maximum Entropy Model)<sup>[10]</sup>对文档进行词性标注和分词,然后使用 simHash<sup>[11]</sup>从那些已修复的 bug 中找出相似性很大的 bug,若没有相似的,最后使用文档处理的方法分析 trace 并找出错误根源和解决方法。

Description of PMR 14568,112,848:

Customer is running 5.1.0.6 (console : v5.1.07.03), encountered an issue where taking a configuration backup from GUI failed with CMMVC8000E and CMMVC5095E error. They tried to take the backup on Aug 11th, and provider log shows: 2011-08-11T11:45:34:563490 0xf5fedba0 ERROR API Error code:2 fail to open the backing up configuration file on cluster:/dumps/cimom/svc.config.backup.xml.zip

Customer succeeded in taking the backup once in four attempts, but provided svc snap seems to include a valid copy of config.xml.

Description of PMR 54435,999,738 :

C&ED SVC cluster version 5.1.0.3 , Onsite discover using GUI, backup configuration failure with error CMMVC8000E try to use CLI "svsconfig backup" can complete successfully. Try to re-start WAS for SVC in SVC console, symptom the same but error change to CMMVC5095E. View provider log 2011-10-31T00:58:05:571347 0xf57ffba0 ERROR API Error fail to open the backing up configuration file on cluster:/tmp/svc.config.backup.xml.zip Screen capture of failure have been upload under Other files: PRDSVC-Backup-error.doc Please review and check how to resolve, and can restart CIMON using cimon tools can resolve the problem?

Description of PMR 14951,6X2,760

SVC 5.1.0.6 Symptom " SVC Master Console displayed "CMMVC5095E" when they perform configuration on backup. (configuration was not backed up.) And they performed configuration change according to the message. (turned up CIMOM sensitivity to highest setting) Then configuration backup was performed, But this problem was not solve d. Recently, the ERROR occurred within the following time. 2010/10/12 12:00-24:00

图1 CIM 中同一 bug 的三种不同描述(下划线标注为重要信息)

## 1 相关研究

### 1.1 公共信息模型

公共信息模型(CIM)是工业界的一种公开标准,由分布式管理任务组(Distributed Management Task Force, DMTF)所开发,定义了 IT 环境中的受控元素如何被表示为一组通用的对象以及这些对象之间的关系。其目的是在不同的生产商和提供商之间为受控元素的管理提供一种一致的方法,是一个与具体实现无关的、用于描述管理信息的概念性模型,可用于所有 IT 相关管理领域。它主要由下面两部分构成:一部分是 CIM 规范,定义了与其他管理元素整合的语言和方法;另一部分是 CIM 模式,提供了设备、系统和应用程序的实际模型描述,包括了描述实体的类、属性、方法以及它们之间的关系。CIM 模型由核心模型、公共模型和扩展模型三层构成,它采用一种标准的格式来为不同平台和开发商的应用描述管理数据,以使数据能够在多种应用间共享。CIM 建模是一种通用方法,特定管理域的 CIM 建模是在核心模型和公共模型的基础上进行扩展。

CIM 已有很多开源实现,例如 OpenPegasus、OpenWBEM,已经被很多公司的设备所使用和嵌入到设备中,例如 IBM、EMC 等。Windows 系统中也自带了 CIM 的实现 Windows 管理规范。这些实现都提供了不同的接口,基于提供的接口, CIM Provider 开发者只需关心与设备相关的逻辑。图2为 CIM 开发框架图。另外这些开源实现也提供了大量 trace 文件以及不同的 trace 级别,这些都可用于发现 bug 的根本原因。

### 1.2 词典创建

词典作为自然语言处理的一项基础资源,不仅对分词、命名实体识别、词义消歧等自然语言处理的底层技术有帮助,而

且被广泛应用于文本分类的应用层。但是词典中的词在不同的领域有些不同的语义<sup>[12-13]</sup>,所以针对不同的应用领域需要构建不同的词典,这样才能更准确地为后续的词性标注、分词以及文档处理提供更好的基础。针对特殊的领域(CIM),需要基于已有的语料构建一个针对性的词典,然后基于词典进行分词和一些自然语言处理。例如 CMMVC8000E 等词在其他文档分析领域可能没什么特殊的意义,但是在 CIM 中却有非常重要的意义,它能够很好地辅助发现 bug 的问题所在。图3表明了在本文中构建词典的流程方法,它主要使用收集的文档中出现的一些关键词来构建所需的词典。

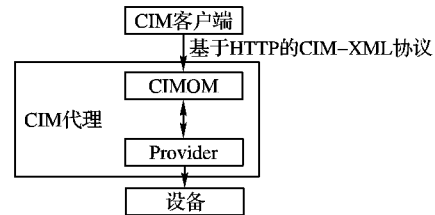


图2 CIM 框架

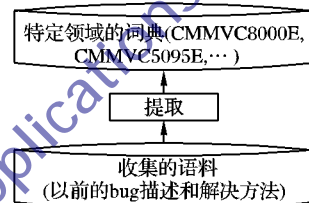


图3 构建词典的流程方法

### 1.3 最大熵法

最大熵模型<sup>[10]</sup>是一种机器学习方法,主要基于最大熵原理,可应用于自然语言处理的许多领域(如词性标注及分词、句子边界识别、浅层句法分析及文本分类等)。它的主要思想就是在只掌握关于未知分布的部分知识时,应该选取符合这些知识但熵值最大的概率分布。因为在这种情况下,符合已知知识的概率分布可能不止一个。最大熵原理的实质就是,在已知部分知识的前提下,关于未知分布最合理的推断就是符合已知知识最不确定或最随机的推断,任何其他的选择都意味着我们增加了其他的约束和假设,这些约束和假设根据我们掌握的信息无法做出。

最大熵模型的目的是为了估计拥有最大熵的概率分布  $p$ , 基于下面的公式:

$$H(p) = - \sum_{x \in A \times B} p(x) \ln p(x);$$

$$x = (a, b), \text{ where } a \in A \wedge b \in B \quad (1)$$

$$p^* = \arg \max_{p \in P} H(p) \quad (2)$$

其中:  $a$  为单词所属的词性,  $b$  为该词的上下文。更多细节请查看文献[2]。

最大熵模型的优点就是只需要集中精力选择特征,而且可以很灵活地选择、使用各种不同类型的特征,且特征容易更换。利用最大熵建模,一般也不需要做在其他方法建模中常常使用的独立性假设,参数平滑可以通过特征选择的方式加以考虑,无需专门使用常规平滑算法单独考虑。

### 1.4 simHash

simHash<sup>[11]</sup>主要用于比较两个文本的相似性,并用于去除从网络上爬取的重复文档。以往的方法大多是将文本分词之后,转化为特征向量距离的度量,比如常见的欧氏距离、海明距离或者余弦角度等。但这些方法的一个最大缺点就是无

法将其扩展到海量数据并且准确率也不是很高。例如,试想下随着客户所上报的 bug 越来越多,如果将待查找的每个 bug 描述都去和数据库里已有的每个已解决的 bug 计算一下余弦角度,其计算量相当大。

本文考虑采用为每一个 bug 描述文档通过哈希的方式生成一个指纹。传统的加密式哈希,比如 MD5,将原始内容尽量均匀随机地映射为一个签名值,原理上相当于伪随机数产生算法,输入内容哪怕只有轻微变化,哈希就会发生很大的变化。因此要设计一个哈希算法,对相似的内容产生的签名也相近,是更为艰难的任务,因为它的签名值除了提供原始内容是否相等的信息外,还能额外提供不相等的原始内容的差异程度的信息。很明显,前面所说的 MD5 等传统哈希无法满足我们的需求。相比而言, simHash 很好地解决了前面所提到的问题。

simHash 是局部敏感哈希的一种,最早由 Moses Charikar<sup>[1]</sup> 提出,它的输入是一个向量,输出是一个  $f$  位的签名值:

- 1) 将一个  $f$  维的向量  $V$  初始化为 0;  $f$  位的二进制数  $S$  初始化为 0;
- 2) 对每一个特征:用传统的哈希算法对该特征产生一个  $f$  位的签名  $b$ 。对  $k \in [1, f]$ :如果  $b$  的第  $k$  位为 1,则  $V$  的第  $k$  个元素加上该特征的权重;否则,  $V$  的第  $k$  个元素减去该特征的权重。
- 3) 如果  $V$  的第  $k$  个元素大于 0,则  $S$  的第  $k$  位为 1,否则为 0;
- 4) 输出  $S$  作为签名。

## 2 本文方法

在本文中提出了一种基于自然语言处理的方法对 CIM 的 bug 进行自动调试。首先使用最大熵模型对有待解决 bug 的文档描述进行分词,然后基于构建的词典使用 simHash 找出那些重复性很大的已修复的 bug,若没有比较相似的,再使用文档处理的方法分析客户提供的 trace 找出问题所在和解决方法。若数据比较大时,使用关联规则算法挖掘一些规则。以下为本文所提出方法的详细步骤。

- 1) 首先收集以前已解决 bug 的描述文档和解决方法作为训练样本,去除停用词并构建针对 CIM 领域的词典
- 2) 针对客户新提交的 bug 使用最大熵法进行分词和词性标注,然后基于已构建的词典提取特征,并使用 simHash 计算新 bug 与收集的已修复 bug 的相似性,找出最相似的那个 bug,若该相似值大于设定的阈值,则把这个最相似的 bug 的根本原因和解决方法反馈给客户,否则继续 3)。
- 3) 使用文档处理分析客户提交的 trace。本文中主要以 CIM provider 的 trace 文件作为例子,同样的方法也可用于 openpegasus 等其他模块。图 4 是客户提供的 trace 文件的一个例子,主要包括 ProviderLog 和 ProvierTrace 两个文件。
  - a) 首先定义一些与错误和异常相关的单词,例如 error, exception, failed 等;
  - b) 从文档中去除一些不必要的详细信息,只保留那些调用、时间片以及调用函数的返回值;
  - c) 从 ProviderLog 文件提取 error 信息和它相关的时间片;
  - d) 从 ProvierTrace 文件中提取上面时间片 1 秒内的调用,及调用函数的返回值;

e) 分析前面步骤提取的信息获取一些模式,例如 sendCommand; returnValue, log; description;

f) 基于上面步骤获得的调用返回值和 log 描述,然后向客户提供 bug 的解决方法。若数据比较大,可以训练关联规则,自动使用训练得到的规则获取解决方法。

ProviderLog file:

```
2011-09-27T08:43:52:913053 0xf7684b90 ERROR CPA User superuser
not found in FreeMap
2011-09-27T08:43:52:913291 0xf7684b90 ERROR CPA User superuser
not found in FreeMap
2011-09-27T08:55:38:417218 0xf56f5b90 ERROR API Failed to copy
iostats dump file /dumps/iostats/Nv_stats_78G032B-2_110927_084520
2011-09-27T08:55:38:417414 0xf56f5b90 ERROR API Failed to get
performance statistics file for IBMTSSVC_PerformanceStatisticsService
```

ProviderTrace file

```
2011-09-27T08:55:38:288260 PerformanceStatisticsServiceGlue.cpp :
IBMTSSVC_PerformanceStatisticsServiceInvokeMethod:1426 ENTER
GetStatisticsCollection
2011-09-27T08:55:38:288341 0xf56f5b90 TRACE CIM glue/ibmtssvc/
PerformanceStatisticsServiceGlue.cpp:doGetStatisticsCollection:606
ENTER
include/hwlayer/connection/Connection.h:sendCommand:287 RETURN:
SendResult <ReturnCode = Success (0), NativeError = 0CommandResult
= 0>
2011-09-27T08:55:38:417083 0xf56f5b90 TRACE CPA glue/cpa/cpalog.
cpp:~enter_exit:812 EXIT
2011-09-27T08:55:38:417098 0xf56f5b90 TRACE CPA include/hwlayer/
ConnectionManager.h:sendCommand:157 RETURN:0
2011-09-27T08:55:38:417128 0xf56f5b90 TRACE CPA include/hwlayer/
Cluster.h:sendCommand:1060 RETURN:Success (0)
2011-09-27T08:55:38:417159 0xf56f5b90 TRACE CPA include/hwlayer/
Cluster.h:sendCommand:1048 RETURN:38359
2011-09-27T08:55:38:417206 0xf56f5b90 TRACE LOG Failed to copy
iostats dump file /dumps/iostats/Nv_stats_78G032B-2_110927_084520
2011-09-27T08:55:38:417238 0xf56f5b90 TRACE API src/hwlayer/api/
PerfStatsAPI.cpp:copyPerfStatsFile:165 RETURN:0
2011-09-27T08:55:38:417324 0xf56f5b90 TRACE API glue/cpa/cpalog.
cpp:~enter_exit:812 EXIT
2011-09-27T08:55:38:417342 0xf56f5b90 TRACE API src/hwlayer/api/
PerfStatsAPI.cpp:getPerfStatsFiles:103 RETURN:0
2011-09-27T08:55:38:417404 0xf56f5b90 TRACE LOG Failed to get
performance statistics file for IBMTSSVC_PerformanceStatisticsService
2011-09-27T08:55:38:417428 0xf56f5b90 TRACE API src/ibmtssvc/
PerformanceStatisticsServiceProvider.cpp:doGetStatisticsCollection:147
RETURN:4
```

图 4 客户提供 trace 文件的例子(下划线标注了该 trace 的重要性)

为了提供一个更直观的描述,图 5 给出了本文所提出方法的流程。

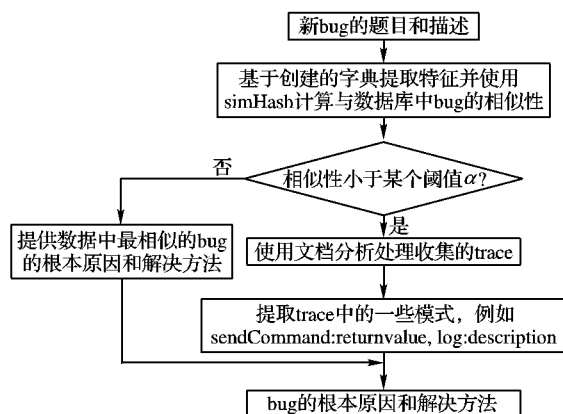


图 5 算法流程

## 3 实验结果和讨论

本文收集了已经修复的 103 个 bug,其中 80 个作为训练样本,其他 23 个作为测试样本,其中有 8 个 bug 在训练样本有重复 bug 存在。实验也考虑了该算法中的相似性阈值  $\alpha$  的影响,表 1 显示了  $\alpha$  参数对实验准确率(8 个重复 bug 找出的数量)的影响。针对那些未在训练样本中出现的另外 15 个 bug,使用文档处理的方法分析大 trace 文件,从中提取一些模

式,通过这些模式可以更快地发现 bug 的根本原因。表2给出了从文档处理提取的模式发现 bug 根本原因的数目,该结果也受前面参数  $\alpha$  的影响。

表1  $\alpha$  与找出重复 bug 数量的关系

$\alpha$	准确率	$\alpha$	准确率
0.95	5/8 = 62.5%	0.85	7/8 = 87.5%
0.90	6/8 = 75.0%	0.80	7/8 = 87.5%

表2  $\alpha$  与从文档处理提取的模式发现问题原因个数的关系

$\alpha$	准确率	$\alpha$	准确率
0.95	11/18 = 61.1%	0.85	14/16 = 87.5%
0.90	12/17 = 70.5%	0.80	14/16 = 87.5%

上述的实验结果表明,本文的方法取得了很好的效果,主要由于以下几方面的原因:1) 充分利用了已修复的 bug 信息描述,减少了重复 bug 出现的可能性;2) 使用文档处理的方法从文档中去除一些无用的信息,只保留那些对发现问题有很大帮助的 trace,可以大大减轻程序员的负担。另外本文方法的效果也可以从时间方面来体现,但是由于没有相关的数据,本文通过分析来表明该算法的有效性。本文的算法中,若新提交的 bug 在已修复的 bug 数据库中出现,则可以非常顺利和迅速地获得解决方法。若未在数据库中出现,程序员也只需要分析那些通过文档处理获得的关键 trace,而不是从大文件中去寻找那些不是很相关的 trace,避免了程序员去逐行查看那些大文件,同时也减小了那些无关 trace 导致错误方向的可能性。

#### 4 结语

本文提出了一种基于自然语言处理的方法对 CIM 的 bug 进行自动调试。首先使用最大熵模型对已解决 bug 的文档描述进行分词,然后基于构建的词典使用 simHash 找出那些重复性很大的已修复的 bug,若没有比较相似的,再使用文档处理的方法分析客户提供的 trace 找出问题所在和解决方法。该方法充分利用已有的一些信息来帮助发现 bug 的原因和解决方法,大大减小了解决重复 bug 的可能性,另外也减少了人工查找根本原因和解决方法的时间。

#### 参考文献:

- [1] LIBLIT B. The cooperative bug isolation project[EB/OL]. [2003-10-09]. <http://www.cs.wisc.edu/cbi/>.
- [2] WOOD M. A dynamic approach to statistical debugging: building program specific models with neural networks[D]. Georgia: Georgia Institute of Technology, 2007.
- [3] ZHENG A X, JORDAN M I, LIBLIT B, *et al.* Statistical debugging: simultaneous identification of multiple bugs[C]// Proceedings of the 23rd International Conference on Machine Learning. New York: ACM Press, 2006: 1105-1112.
- [4] ZHENG A, JORDAN M, LIBLIT B, *et al.* Statistical debugging of sampled programs[C]// Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2004: 501-510.
- [5] LIU C, YAN X F, FEI L. Sober: statistical model-based bug localization[J]. Symposium on the Foundations of Software Engineering, 2006, 30(5): 286-295.
- [6] LIBLIT B, NAIK M, ZHENG A X, *et al.* Scalable statistical bug isolation[C]// PLDI '05: Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation. New York: ACM Press, 2005: 15-26.
- [7] HANGAL S, LAM M S. Tracking down software bugs using automatic anomaly detection[C]// Proceedings of the 24th International Conference on Software Engineering. Washington, DC: IEEE Computer Society, 2002: 291-301.
- [8] LIU C, FEI L, YAN X, HAN J, *et al.* Statistical debugging: a hypothesis testing-based approach[J]. IEEE Transactions on Software Engineering, 2006, 10(3): 831-848.
- [9] ANDRZEJEWSKI D, MULHERN A, LIBLIT B, *et al.* Statistical debugging using latent topic models[C]// 18th European Conference on Machine Learning. Berlin: Springer-Verlag, 2007: 6-17.
- [10] BERGET A L, DELLA PIETRA V J, DELLA PIETRA S A. Maximum entropy approach to natural language processing[J]. Computational Linguistics, 1998, 22(1): 39-71.
- [11] CHARIKAR M S. Similarity estimation techniques from rounding algorithms [C]// Processing of the 34th Annual ACM Symposium on Theory of Computing. New York: ACM, 2002: 10-19.
- [12] 王健,冀明辉,林鸿飞,等. 基于上下文环境和句法分析的蛋白质关系抽取[J]. 计算机应用, 2012, 32(4): 1074-1077.
- [13] 谈文蓉,符红光,刘莉,等. 一种基于贝叶斯分类与机读词典的多义词排歧方法[J]. 计算机应用, 2006, 26(6): 1389-1391.
- [14] 吴朝福,胡占义. PnP问题的线性求解算法[J]. 软件学报, 2003, 14(3): 682-688.
- [15] SHI X F, DIAO C Y, LU D M. A camera extrinsic parameters calibration using 3 point-targets[C]// 孙立峰,杨士强,戴国强. 和谐人机环境2006. 北京:清华大学出版社, 2007: 139-150.
- [16] 付强,全权,蔡开元. 基于自由运动的一维标定物的多摄像机标定[C]// 中国自动化学会控制理论专业委员会: D卷. 北京:北京航空航天大学出版社, 2011: 5023-5028.
- [17] HARTLEY R I. Estimation of relative camera position for uncalibrated cameras [C] // Proceedings of Second European Conference on Computer Vision. London: Springer-Verlag, 1992: 579-587.
- [18] 吴朝福. 计算机视觉中的数学方法[M]. 北京: 科学出版社, 2008.
- [19] TRIGGS B, ZISSERMAN A, SZELISKI R. Vision algorithms: theory and practice[M]. Berlin: Springer-Verlag, 2000.
- [9] FISCHLER M A, BOLLES R C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography[J]. Communication of the ACM, 1981, 24(6): 381-395.
- [10] 张灵飞,陈刚,叶东,等. 基于一维标定物和改进进化策略的相机标定[J]. 光学学报, 2009, 29(11): 3136-3141.
- [11] 郭环,华玉爱. 用零点定理理解不等式[J]. 山东轻工业学院学报, 2000, 14(2): 74-76.
- [12] HARTLEY R, ZISSERMAN A. Multiple view geometry in computer vision [M]. Cambridge, UK: Cambridge University Press, 2000.
- [13] 唐虹,怀时卫,石峰. 基于图像的摄像机焦距参数的算法研究[J]. 计算机仿真, 2011, 28(8): 237-241.
- [14] MAYBANK S J, FAUGERAS O D. A theory of self-calibration of a moving camera[J]. International Journal of Computer, 1992, 8(2): 123-151.

(上接第1427页)