自适应系统软件传感器设计与实现

吴 斌 毛新军 董孟高 李学斯
(国防科学技术大学计算机学院 长沙 410073)

摘 要 随着Internet的普及应用,越来越多的软件系统运行在开放的环境中,需要感知和适应环境的变化。如何支持这类复杂软件系统的开发和运行已经成为当前软件工程面临的一项重要挑战。针对自适应系统与其驻留环境的交互问题,将自适应系统中的软件实体抽象为自主Agent,提出了自主Agent感知环境变化的软件传感器及其与环境的动态关联思想,给出了软件传感器的设计和实现。不同于已有研究,将软件传感器视为一类特殊的软件Agent。最后通过案例分析展示了上述思想和技术的可行性和有效性。

关键词 环境,自适应Agent,感知,软件传感器
中图法分类号 TP301 **文献标识码** A

Design and Implementation of the Software-sensor of Self-adaptive System

WU Bin MAO Xin-jun DONG Meng-gao LI Xue-si
(Department of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract Along with the popularization of Internet applications, more and more software systems operate in an open environment, and need to perceive and adapt the change of environment. How to support the development of such complex software systems has become a major challenge to the current software engineering. We discussed the problem between the adaptive system and its environment, and we abstracted software entity from adaptive system to Autonomous Agent. We put forward the thought of software sensor dynamic association with the environment, given the design and implementation of software sensor. Unlike existing research, we regarded software sensor as a kind of special software Agent. Finally, papers demonstrated the feasibility and validity of these ideas and techniques by case studies.

Keywords Environment, Self-adaptive Agent, Sense, Software sensor

1 引言

计算机技术应用领域的扩大和深化,使得软件系统的规模不断扩大,并呈现出新的复杂性特点,具体表现为:(1)环境复杂性,软件系统所驻留的环境变得日趋复杂,并具有开放、动态、不可控、无法事先确定等特点,如Internet。(2)系统复杂性,软件系统驻留在环境中,需要不断地调整自身的结构和行为以适应环境的变化,满足系统的设计目标,从而展示出自适应和自演化的复杂性特征。运行在Internet之上的软件系统的规模和复杂性越来越高,并呈现出以下新的形态和特点:面向组织而不是单一个人、自治性、分布和协同性、开放和动态性、自适应性等[1]。

为了适应开放的Internet环境,软件需要根据外部环境的变化按照既定的目标进行自适应和自演化,从而使系统能够在运行过程中更好地满足用户的需求。自适应软件是一类特殊的软件,这类软件能够在运行过程中实时收集系统的各种变化信息,并根据预先设定好的策略,在必要时对自身进行自动调整,以更好地为用户提供服务[2]。近年来,面向Agent软件工程受到了学术界和工业界的高度关注和重视,尤其是近两年来研究活跃,发展迅速[3]。软件实体是构成软件系统的基本元素[4],用Agent来抽象表示软件实体,但基于现有的Agent技术如Agent类、Agent类型构造的软件Agent结构和行为在其生命周期中不能动态地发生改变,无法满足自适应的开发要求。在这种情形下,引入了自适应Agent(SAgent)的概念,并使用自适应Agent作为开放的Internet环境下的软件实体。

自适应Agent在其生命周期中能够感知自身内部状态的改变及外部环境的变化,根据已有的策略采取一系列自适应行为可适应这些变化。我们对现有的Agent开发运行平台JADE(Java Agent DEvelopment Framework)[5]进行扩展和改进,开发了支撑自适应Agent开发和运行的平台:SADE(Self-adaptive Agent DEvelopment Framework)。SADE的关键技术包括动态绑定技术[6]和环境感知技术。

对环境的感知和响应处理是开发自适应系统要解决的重要问题,为了对环境进行有效的分类和描述,提出了环境Agent的概念。为了便于对客观物理世界的环境信息进行提取归纳,我们构造了软件传感器,作为一类特殊的Agent,其专门负责提取物理世界的信息并为上层环境Agent使用这些信

到稿日期:2009-09-16 返修日期:2009-12-10 本文受863国家重点基金项目(No. 2007AA01Z135)资助。
吴 斌(1982—),男,硕士生,主要研究方向为面向Agent的软件工程,E-mail:wubin_nudt@yahoo.com.cn;毛新军(1970—),男,博士生导师,主要研究方向为面向Agent的软件工程、分布计算技术等。

息提供查询服务。

我们针对网构软件的自适应、自演化等特性开展研究，着重关注如何提供有效的方法和工具来支持复杂自适应和自演化网构软件系统的分析、设计和构造。本文重点介绍 SADE 平台中软件传感器的设计和实现；第 2 节的内容是环境感知与软件传感器抽象；第 3 节详细介绍 SADE 平台及软件传感器的设计框架；第 4 节通过案例介绍软件传感器的使用方法；最后对全文进行小结。

2 环境感知与软件传感器抽象

软件实体的存在不是孤立的，它们总是存在于特定计算和(或者)物理环境中[7]。软件系统的主旨是服务于用户。因此，用户、软件系统、环境三要素之间的互动成为软件系统运行的核心。在经典的软件结构模型中，用户这一要素主要通过需求定义来体现，运行环境这一要素则主要通过各种辅助文档来说明，而相对固定的用户需求和环境支撑等要素则是作为一个全局性的前提和假设体现在软件开发、运行等各个阶段之中；为了使经典的软件模型及其开发与运行技术具有一定的灵活性，研究人员从不同的角度，来用了能够适应用户需求与环境变化的方法与技术来提升软件开发与软件系统的适应能力[8]。在传统的软件系统中，环境是通过辅助文档来说明的，单就运行态的软件系统而言，环境没有被显式地描述，而是隐含于系统中。在开放的 Internet 环境下，软件的运行环境是动态不可控的，如果软件能够感知外界环境的变化，并针对变化作出自适应行为以适应新的环境需要，软件的可用性就能有效增强。现在，众多学者认为，环境是一个可开发和设计的元素[9]，可以被显式地描述出来，甚至有观点认为环境是第一位的抽象[10,11]。

环境是一个相对概念，只有在指定参照物的情况下，环境才能够被描述和定义。在多 Agent 系统中，就整个系统而言，软件运行平台是它的环境，就系统中的某个 Agent 来说，它的环境是为它的存在提供基础，能够影响它行为的一切要素。

在多 Agent 系统中，把软件实体抽象为 Agent，Agent 要关注其它 Agent 的行为和状态，这类信息的获取是通过被关注的 Agent 主动向外发布自己的行为和状态信息而获得的，我们认为是动态环境信息。但有时系统的运行过程中 Agent 还需要关注其它信息，例如 CPU 和内存情况。我们采用主动查询的方式获取这类信息，并称这类信息为静态环境信息。采集的底层静态环境信息往往需要进行高层的抽象，并且采用统一的方式提供给需要此信息的 Agent。为了能让其它 Agent 有效使用这些信息，还要提供查询接口。我们把这些功能封装到软件传感器当中。软件传感器也是 Agent。

开发人员在利用 SADE 平台开发自适应系统时，需要描述和定义 Agent 的环境，为了使开发人员开发时能有一个统一和方便的描述方式，我们设计了一类特殊 Agent，称为环境 Agent，这类 Agent 负责与软件传感器交互，为需要查询环境信息的 Agent 提供查询服务。

由此在系统中的 Agent，按照功能来划分，有 3 种类型，如图 1 所示，一种是应用 Agent(Application Agent)，这是开发人员针对不同的应用设计实现的；第二种是环境 Agent (Environment Agent)，负责响应应用 Agent 的查询请求并向下查询软件传感器收集的环境信息；第三种是软件传感器(SoftSensor)，负责与物理环境打交道，收集环境信息，响应环境 Agent 的查询请求。

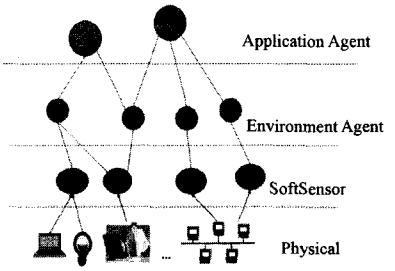

图 1 环境感知系统整体框架

系统中应用 Agent 的环境可分为动态和静态两部分，动态部分的环境可视为该应用 Agent 所关注的其它应用 Agent 的集合，感知的方式是订阅(SUBSCRIBE)方式；静态部分的环境由环境 Agent 来抽象表示，不同的环境 Agent 负责向上层应用 Agent 反馈不同的环境信息，感知的方式是探测(SENSE)方式。这样做的好处在于，对于应用 Agent，感知动态和静态的环境可以采用统一的描述。采用订阅方式，环境信息是由信息源主动发布的，应用 Agent 如果想获得此信息就需要进行订阅。采用探测方式时，信息源不会主动向外发布信息，当应用 Agent 需要获取环境信息时采用主动查询的方式向环境 Agent 发送查询请求，环境 Agent 再向下发送请求到软件传感器，由软件传感器跟实际的物理环境打交道，并向上反馈环境信息。

实现时用 EnvironmentContext 类来描述应用 Agent 所要感知的环境，每一个 EnvironmentContext 类生成的对象表示一个环境描述，一个环境描述用于刻画应用 Agent 要感知的一个 Agent 的内容。示例如图 2 所示。

```
EnvironmentContext envContext=new EnvironmentContext(Enviro-
                              mentContext. AGENT_TYPE,
                              "SystemResource", Environment-
                              Context,SENSE);
envContext. addContextScope(EnvironmentContext. PROPERTY);
envContext. addConstraint(EnvironmentContext. PROPERTY,"CPU
_LOAD");
addEnvironmentContext(envContext);
```

图 2 EnvironmentContext 描述示例代码

上述代码表示应用 Agent 要感知的环境是一个名字叫做“SystemResource”的环境 Agent，感知方式是 SENSE 方式，感知的内容是内存总量“MEM_TOTAL”和 CPU 占用率“CPU_LOAD”。

为了统一概念模型，系统中把软件传感器抽象为 Agent，即软件传感器也是 Agent。软件传感器跟实际的物理环境打交道，负责收集静态环境的信息，并响应上层环境 Agent 的感知请求。

3 软件传感器框架设计

SADE(Self-adaptive Agent Development Environment)平台是在 JADE(Java Agent Development Framework)平台的基础上开发的。JADE 是目前应用较广的一个多 Agent 系统开发运行平台，该平台为具有生命周期特性的 Agent 提供了运行环境，同时 JADE 作为软件框架为开发多 Agent 系统提

供了便利。在 JADE 中，Agent 通过 Behaviour 对象来执行业务；Agent 之间通过 ACLMessage 进行通信。

SADE 保留了 JADE 的以上特点，如图 3 所示，同时引入了 Role 的概念。Role 被视为系统中一类自适应 Agent 的行为和功能的实现单元。自适应 Agent 和 Role 之间是动态绑定的关系[12]。动态绑定关系是指自适应 Agent 在其生命周期内可以动态地加入或者退出某个 Role，所绑定的 Role 可以处于激活状态也可以处于非激活状态，一个自适应 Agent 实例在某一个时刻可以绑定多个 Role。通过动态绑定操作，自适应 Agent 实例可以改变自身内部结构，实现不同的业务[13]。

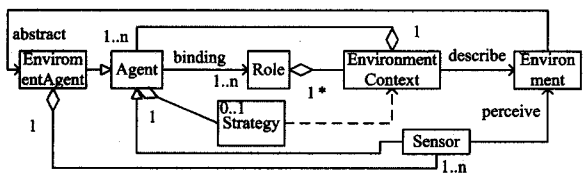

图 3　SADE 框架

软件传感器具备以下 4 个功能：

(1)软件传感器自身向 DF 的注册和注销；

(2)软件传感器对环境 Agent 注册和注销请求的处理；

(3)软件传感器对环境 Agent 查询环境状态请求的处理；

(4)软件传感器对特定环境状态信息的探测。

感知不同的环境信息，需要不同功能的软件传感器。SADE2.0 平台为开发人员提供设计软件传感器的框架，并实现两个常用的软件传感器 CPUSensor 和 MemSensor，如图 4 所示。

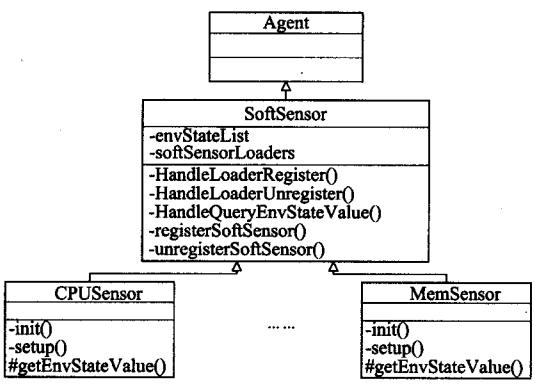

图 4　软件传感器继承关系类图

软件传感器被设计成一类特殊的 Agent，在软件传感器基类中实现了每个软件传感器都必须具备的注册注销操作，并制定了子类 Sensor 必须重载的方法规范。基类中的 HandleLoaderRegister 和 HandleLoaderUnregister 是负责响应环境 Agent 注册注销的行为。HandlerQueryEnvStateValue 是负责响应环境 Agent 对所需环境信息查询的行为。

利用平台提供的框架，可以快速方便地开发出软件传感器。开发人员在开发新的软件传感器时只需继承 SoftSensor 基类，并重载对应的功能函数。在 SoftSensor 基类中保存有环境状态的链表，开发人员需要在软件传感器子类中填入相关信息以充实环境状态链表的内容，链表中的项由 EnvState 类的对象组成。EnvState 类描述了环境状态的名字和值。设计 CPUSensor 时初始化代码如图 5 所示。

```
static final String SERVICES_ONE="CPU_LOAD";
static final String SERVICES_TWO="CPU_THREAD";
Private void init(){
    EnvState. serOne=new EnvState();
    SerOne. sEnvStateName=SERVICES_ONE;
    SerOne. objValue=0;
    ServicesList. add(serOne);
    ServicesList. add(new EnvState(SERVICES_TWO,
0));
}
public CPUSensor(){
    super();
    init();
}
```

图 5　CPUSensor 初始化代码

子类中必须实现 getEnvStateValue 方法，该方法入口参数是环境状态的名字，返回值是环境状态的值。由于客观世界的复杂性和多样性，软件传感器感知的数据类型也必然是多种多样的，因此，返回值的类型是 Java 语言中的基础数据类型 Object。

4　软件传感器的使用及案例分析

软件传感器的使用对基于 SADE 平台的应用系统来说是透明的，开发人员只需在角色中描述要感知的环境，系统运行时就会自动根据开发人员的描述加载相应的环境 Agent，之后由环境 Agent 加载所需的软件传感器。

案例说明：我们开发了模拟机器人清理垃圾的系统。系统中 GarbagerManageAgent 定时产生垃圾，垃圾的类型有 BrickGarbage 和 DustGarbage。PowerfulAgent 是具有自适应特性的 Agent，它会找到当前离它最近的垃圾，并且根据垃圾的类型，分别加入 PorterCleaner 和 VacuumCleaner 角色来清理垃圾，当它发现没有垃圾时加入 MusicPlayer 角色。

场景描述：机器人 PowerfulAgent 在绑定了角色 Robot 后，由于角色 Robot 的环境描述中有 GarbageManger 角色，PowerfulAgent 就将感知绑定 GarbageManager 角色的 Agent，也就是 GarbageManagerAgent。GarbageManagerAgent 在场地中随机放置垃圾，模拟现实中的垃圾生成，并将这些环境信息向外发布。Robot 角色订阅了关于垃圾的信息，它将在收到这些信息后判断离自己最近垃圾的位置并移动到垃圾所在位置，绑定能够清理垃圾的角色 VacuumCleaner 和 PorterCleaner，并把垃圾清除。同时 PowerfulAgent 能够感知当前环境中的静态信息，即当前的功耗，这里用 CPU 的占用率来模拟功耗的大小，当功耗过大，也就是当清理垃圾的 PowerfulAgent过多时它将加入一个功耗较小的角色。当所有的垃圾都清理完时，PowerfulAgent 还会加入 MusicPlayer 角色用来播放音乐。

情况分析：

系统中存在 3 种类型的 Agent，一类是应用 Agent，包括垃圾管理者 GarbageManagerAgent 和机器人 PowerfulAgent；一类是环境 Agent，这里用到的是 SystemResourceAgent；第三类是软件传感器 SoftSensor，系统中使用了 CPUSensor 和 MemSensor，分别用于探测 CPU 的状态和内存的状态。

代码说明：

策略中环境感知相关代码如图 6 所示，在 PowerfulAgent所使用的策略中，加入感知静态信息的语句。该 Agent 关注名叫 SystemResource 的环境 Agent 中的 CPU_LOAD 属性，看属性值是否大于 70%，如果大于，则加入“Sleeper”角色。

```
//感知静态环境信息
private class Rule_7 extends CyclicBehaviour{
    MessageTemplate mt = MessageTemplate. MatchPerformative
(ACLMessage. ENVIRONMENT_CHANGE);
        public void action(){
        ACLMessage msg=myAgent. receive(mt);
            if (msg! =null){
                if (checkEnvMsg(new AgentBehaviourMessage
                    ("PowerfulAgent","","CleanGarbage"),msg,
                    agent)){
                    String[][] args=new String[1][4];
                    args[0][0]="SystemResource";
                    args[0][1]="CPU LOAD";
                    args[0][2]="int";
                    args[0][3]=">70%";
                    String[][] adaptation_args=new String[2][2];
                    adaptation_args[0][0]="quit";
                    adaptation_args[0][1]="Robot";
                    adaptation_args[1][0]="join";
                    adaptation_args[1][1]="Sleeper";
                    agent. addBehaviour (new CheckEnvironment-
                    State(args,adaptation_args,0));
                }
            }else{
                this. block();
        }
    }
}
```

图 6　策略文件中感知静态环境信息部分代码

角色中环境上下文描述，采用三元组 EnvironmentContext=〈type，name，sense_mode〉。相关代码如图 7 所示，该角色在初始化时针对所关注的环境定义了两个环境上下文(EnvironmentContext)对象，即 cxt1 和 cxt2，cxt1 表示要感知的环境采用的是订阅(SUBSCRIBE)方式，类型是角色类型(ROLE_TYPE)，角色名字是 GarbageManager，对该角色中名叫“GridInfo”的属性(PROPERTY)进行关注；cxt2 表示要感知的环境采用的是感知(SENSE)方式，类型是 Agent 类型(AGENT_TYPE)，也就是说感知的是一个环境 Agent，该环境 Agent 的名字叫“SystemResource”，对该 Agent 中名叫“CPU_LOAD”的属性(PROPERTY)进行关注。

```
    public void init(){
    EnvironmentContext cxt1 = new EnvironmentContext(Environ-
                              mentContext. ROLE_TYPE,"Gar-
                              bageManager ", EnvironmentCon-
                              text. SUBSCRIBE);
    cxt1. addContextScope(EnvironmentContext. PROPERTY);
    cxt1. addConstraint(EnvironmentContext. PROPERTY,"GridIn-
fo");
    addEnvironmentContext(cxt1);
    EnvironmentContext cxt2 = new EnvironmentContext(Environ-
                              mentContext. AGENT_TYPE,
                              " SystemResource ", Environment-
                              Context. SENSE);
    cxt2. addContextScope(EnvironmentContext. PROPERTY);
    cxt2. addConstraint(EnvironmentContext. PROPERTY,"CPU_
LOAD");
    addEnvironmentContext(cxt2);
}
```

图 7　角色文件中对所要感知的环境进行描述的代码

案例运行结果：

如图 8 所示，PowerfulAgent 把 GarbageManagerAgent 作为它的环境，GarbageManagerAgent 将当前的垃圾存在状况信息向外发布，PowerfulAgent 绑定的角色 Robot 负责订阅这些信息同时根据信息寻找离自己最近的垃圾，在移动到该垃圾处时根据垃圾的种类绑定角色 VacuumCleaner 或者 PorterCleaner 进行垃圾的清理。

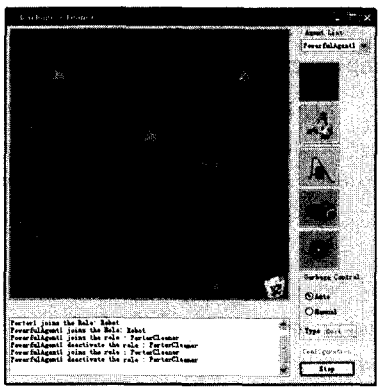

图 8　垃圾收集案例运行示意图

结束语　本文从开发自适应系统入手，通过研究，解决了自适应 Agent 感知环境的问题，创造性地提出了软件传感器概念，并使用软件传感器来感知环境。为了使开发人员能够以统一的方式对环境进行描述，提出并实现了环境 Agent，环境 Agent 为应用 Agent 提供环境信息，屏蔽底层技术细节，使得开发人员可以方便地通过描述 Agent 感兴趣的环境而获得环境的信息。本文提供了软件传感器的开发框架，开发人员可以根据需要开发针对特定领域的软件传感器，并对提出的概念和技术构架进行了编码实现，和策略编辑编译等子系统一起，形成 SADE2.0 平台。

参 考 文 献

[1] 毛新军. 面向主体的软件开发[M]. 北京：清华大学出版社，2005

[2] Zhao Jiangming, Mao Xinjun, Wang Ji. Developing Multi-agent Systems with Dynamic Binding Mechanism[C]//Proc. of IEEE/WIC/ACM 6th Intelligent Agent Technology. 2006

[3] 毛新军，常志明，王戟，等. 面向 Agent 的软件工程：现状与挑战[J]. 计算机研究与发展，2006，43(10)：1782-1789

[4] 吕建，马晓星，陶先平，等. 网构软件的研究与进展[J]. 中国科学，2006，36(10)：1037-1080

[5] http://jade. tilab. com/

[6] 王千祥，申峻嵘，梅宏. 自适应软件初探[J]. 计算机科学，2004，31(10)：168-171

[7] Viroli M, Ricci A, Zambonelli F. Engineering the Environment of Multiagent Systems[J]. jaamas. tex, 2005, 12: 21

(下转第 293 页)

本文方法的识别率相同。人脸图像的频谱特性虽然具有与光照无关的特性，但是它保留的人脸信息量较少，如果选有主元特征作为人脸特征其识别率相对不处理、统计归一化的识别率高，但是主元特征将人脸之间细微的可区分的信息忽略了，因此相对最小非零特征的可区分度小，利用人脸图像的相频特性选用最小特征向量作为人脸特征。

对人脸库(200个人脸样本)进行人脸识别仿真。验证了人脸的光照对不同人脸识别算法的影响，而未考虑人脸姿态表情变化、年龄和人脸附属物对人脸识别的影响。本文将训练库中的人脸样本进行光照变化得到人脸测试样本。部分测试样本如图5所示。

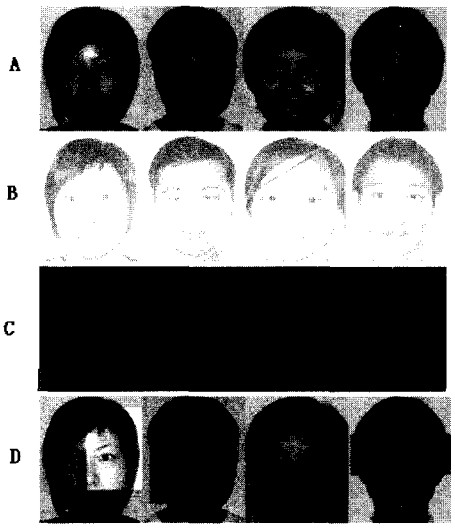

图5 部分测试样本

测试集分别采用直方图归一化处理、不处理、统计归一化处理以及本文方法的人脸识别率如表4所列。

表4 人脸识别率

人脸 \ 识别率	直方图	不处理	统计归一	本文方法	
				主元特征	本文特征
A	1	1	1	1	1
B	1	0.5	0.6	0.8	1
C	1	0.4	0.65	0.8	1
D	0.6	0.4	0.5	0.75	1

对于人脸整体光照变化的测试集，直方图归一化方法和本文方法的识别率相同。但是对于人脸局部光照变化的样本，本文能够完全识别而直方图归一化的识别率仅仅只有0.6。

结束语 传统人脸识别要求人脸图像与人脸训练库的光照差异不大。这一要求限制了人脸识别的应用。本文为了降低人脸识别对环境条件的要求，克服光照对人脸识别的影响，分析了人脸图像相频特性与光照无关。利用相频特性这一特性提出了基于相频特性的人脸识别，在保留了人脸的可区分性的前提下，克服了光照对人脸识别的影响。人脸图像的相频特性之间可区分性的信息量相对较小，因此本文选用最小非零特征向量作为人脸特征。通过对人脸整体光照变亮、变暗和人脸局部区域光照变化的实验，仿真结果表明，对于整体光照变化人脸，运用直方图归一化和本文方法的识别率相同，对于局部光照变化的人脸，本文识别算法比直方图归一化的识别率高。因此本文人脸识别算法对光照具有鲁棒性。

参考文献

[1] Kim T, Kittler J, Cipolla R. Learning Discriminative Canonical Correlations for Object Recognition with Image Sets[J]. Lecture Notes in Computer Scinence, 2006, 3953: 251

[2] Cai D, He X, Hu Y, et al. Learning a spatially smooth subspace for face recognition[C]// 2007 IEEE Conference on Computer Vision and Pattern Recognition. Vols 1-8, 2007: 650-656

[4] Lei Z, Chu R F, He R, *et al. Face recognition by discriminant analysis* with Gabor tensor representation[J]. Advances in Biometrics, Proceedings, 2007, 4642: 87-95

[5] Kim T, Kittler J, Cipolla R. Discriminative Learning and Recognition of Image Set Classes Using Canonical Correlations[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29(6): 1005-1018

[6] Wang X, Tang X. A Unified Framework for Subspace Face Recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26(9): 1222-1228

[7] Xie X, Lam K. An efficient illumination normalization method for face recognition[J]. Pattern Recognition Letters, 2006, 27(6): 609-617

[8] Liu D, Lam K, Shen L. Illumination invariant face recognition [J]. Pattern Recognition, 2005, 38(10): 1705-1716

[9] Xie X, Lam K. Face recognition under varying illumination based on a 2D face shape model[J]. Pattern Recognition, 2005, 38(2): 221-230

[10] Belhumeur P, Kriegman D. What is the Set of Images of an Object Under All Possible Lighting Conditions[Z]: 270-277

[11] Georghiades A, Belhumeur P, Kriegman D. From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose[J]. IEEE Trsnsactions on Pattern Analysis and Machine Intellgence, 2001: 643-660

[12] Tao D, Li X, Wu X, et al. General Tensor Discriminant Analysis and Gabor Features for Gait Recognition[J]. IEEE Trsnsactions on Pattern Analysis and Machine Intellgence, 2007: 1700-1715

[13] 冯海亮，王丽，李见为. 一种新的用于人脸识别的特征提取方法[J]. 计算机科学，2009，36(6)：294-296

[14] 赵建民，朱信忠，江小辉. 基于改进型LBP特征的人脸识别方法研究[J]. 计算机科学，2009，36(8)：276-280

[15] 黄非非，李见为，王玮，等. 结合小波分析和LBP算子的人脸描述与识别[J]. 重庆工学院学报：自然科学版，2009，23(1)：102-108

(上接第155页)

[8] 吕建，马晓星，陶先平，等. 面向网构软件的环驱动模型与支撑技术研究[J]. 中国科学，2008，38(6)：864-900

[9] Platon E, Mamei M, Sabouret N, et al. Mechanisms for Environments in Multi-Agent Systems[J]. si-env-mecha. tex, 2006, 21: 23

[10] Gouaich A, Michel F. Towards a Unified View of the Environment(s) within Multi-Agent Systems[Z]. Informatica, 2005: 423-432

[11] Weyns D, Omicini A, Odell J. Environment as a first class abstraction in multiagent systems[Z]. Auton Agent Multi-Agent Syst DOI 10.1007/s10458-006-0012-0

[12] Mao Xinjun, Chang Zhiming, Shan Lijun, et al. The Dynamic Casteship Mechanism for Modeling and Designing Adaptive Agents[C]// Proc. of the International Workshop on Agent-Oriented Software Development Methodology. 2006: 639-644

[13] 常志明，毛新军，齐治昌. 基于Agent的网构软件构件模型及其实现[J]. 软件学报，2008，19(5)：1113-1124