基于特征的领域构件高内聚控制*)

王映辉[1,2] 张世琨[1] 王立福[1]

(北京大学软件工程国家工程研究中心 北京100871)[1] (陕西师范大学计算机学院 西安710062)[2]

摘 要 构件的高内聚性是基于构件复用研究领域常见的话题,也是软件开发者一直追求的目标。无论是对软件体系结构的获取,还是对软件演化的探究,甚至遗留系统的现代化改造,都与高内聚性构件的确定有着不可分割的天然联系。通过特征跟踪矩阵,建立了领域构件与特征之间的联系;给出了基于特征的n维向量空间中领域构件向量的表示,进而获取了反应领域构件间密切关系的领域构件向量距离矩阵;基于领域构件向量距离矩阵,结合等级簇聚合算法建立了领域构件等级簇树;最后在领域构件等级簇树的基础上,提出了一种高内聚领域构件获取和控制策略。

关键词 领域工程,特征,跟踪矩阵,高内聚,领域构件

Variability Control of High-cohesion Domain Component Based on Feature

WANG Ying-Hui[1,2] ZHANG Shi-Kun[1] WANG Li-Fu[1]

(National Engineering Research Center for Software Engineering, Peking University, Beijing 100871)[1]

(School of Computer Science, Shanxi Normal University, Xi'an 710062)[2]

Abstract It's a general topic to achieve component cohesion in the research filed based on component reuse. High-cohesion of module (component is a special module) is always an aim of software developer. There is inherent relationship between All, which we get software architecture and explore software evolution and modernize legacy system, and high-cohesion component decision. The relationship between domain component and feature is established based on feature track matix. Expression of domain component vector within n-dimension space based feature is represented, then a distance matrix of domain component vector, which represent the relationship within domain components, is achieved. The domain component rank cluster tree is established combine rank cluster algorithm with the distance matrix of domain component vector. At last, the granularity decision strategy of high-cohesion domain component is demonstrated based on the domain component rank cluster tree.

Keywords Domain engineering, Feature, Track matrix, High-cohesion, Domain component

1 引言

目前,构件技术虽未能彻底解决软件变化性问题,但为软件"变化性"的控制奠定了良好的基础。构件对软件变化性的承载是通过构件组合来实现的,可见,如何准确确定构件之间的密切关系(内聚度),使组合更具有合理性,是构件技术解决软件变化性的关键问题之一。

领域工程的易复用性尽管为人们把握软件的共性与变化性提供了有效途径[1,2],然而,直到CMU/SEI (Software Engineering Institute/Carnegie Mellon University)提出面向特征的领域分析FODA(Feature-Oriented Domain Analysis)之后,特征才为人们揭示和把握需求特别是领域需求的变化开辟了一条崭新的途径。由于特征是跨越问题空间和解空间的一阶实体,并贯穿于软件的整个生命周期之中[3],因此基于特征进行构件(指领域构件,下文不再特别说明)组合的合理性研究是一种必然的想法。在利用构件的软件开发中,特征最终会以构件的形式沉淀下来并得以实现。构件与特征之间的这种天然联系,为根据特征及其之间的关系来确定构件之间的内聚关系并进行构件的合理组合提供了一条有效途径。

"内聚"是基于模块(构件也是一种特殊的模块)软件研究领域常见的话题,也是软件开发者追求的目标之一。自结构化时代到面向对象时代,人们一直在倡导着模块的高内聚性。其实,在以构件为基础的软件复用中,无论是对软件体系结构的获取[4],还是对软件演化的研究[5],甚至遗留系统的现代化改造[6,7],都会涉及构件的组合与分解。也就是说,需求和环境等因素的变化,带来的构件的重组和分解是常见之事,但如何组合才能满足"高内聚",对软件质量有着重要的影响。

内聚可分为高、中和低三类[8,9]。高内聚包括功能内聚、顺序内聚、逻辑内聚和时间内聚;中内聚包括过程内聚和通讯内聚;而低内聚包括偶然内聚。其中功能内聚是最高程度的内聚。内聚度提出的重要意义是要求人们通过设计力争做到高内聚,并能辨别出低内聚的模块。有能力通过修改提高模块的内聚度(如将其它内聚通过改造变为功能内聚),降低模块之间的耦合程度,从而提高模块的独立性。

长期以来,人们根据模块的功能,将模块内各成分紧密地联系起来,构成一个完整的有机体,无疑是增大内聚度的合理、自然而可靠的方法。事实上,在结构化软件开发中,这种功能联系法是各种增大内聚方法中最好的方法。但在面向对象技术中,由于一个对象可以看作抽象数据类型封装的模块,包括完成多个功能的操作,也包括数据,这种模块的内聚性不同于功能内聚,对内聚的判定方法需进一步研究[10]。为此,本文通过跟踪矩阵,建立了构件与特征之间的关系,给出了n维向量空间中构件向量变量的表示和构件向量构成的距离矩阵,进而建立了构件等级簇树,并在此基础上描述了基于构件等级簇树的高内聚构件确定策略,使构件能在更为合理的基础上满足领域共性和变化性的需要。

*)本文得到国家"八六三"高技术研究发展计划项目(2001AA113171),国家"九七三"重点基础研究发展规划项目(2002CB312006)和国家博士后基金项目(20040350251)资助。王映辉 博士后,教授,研究方向为软件工程,软件演化。张世琨 博士,教授,研究方向为软件工程和软件体系结构。王立福 博士,教授,博士生导师,研究领域为软件工程。

2 相关概念

本文的研究定位在领域工程范围内，同时涉及到许多相关的概念，如领域/领域工程、特征/特征模型、FODA 等，而且这些概念在不同的研究文献中略有不同，为了能尽量比较精确地呈现和表达本文的思想，首先对这些概念作以阐述。

2.1 领域和领域工程

领域(domain)是指具有相似和相近软件需求的应用系统所覆盖的功能区域。领域知识的内聚性和相对稳定性[1,2]为软件复用活动提供了可供复用的软件资产和潜在的经济利益，使得特定领域的软件复用相对容易成功。对领域实施软件工程，是获取共性和控制变化性，并实现软件复用的有效途径。具体来说，领域工程(domain engineering)是进行软件体系结构和可复用资产(构件和其它副产品)进行系统化设计的过程，进而利用这些可复用资产来构造一簇相关的应用或子系统，从而达到复用的目的。领域工程自领域分析开始，经过领域设计和领域实现，最终获得领域体系结构 DSSA(Domain Specific Software Architecture)和构件等可复用资产，为应用工程中组装和形成新的系统提供支撑。领域工程被公认为是可复用软件资产生产的主要技术手段，也是系统化复用成功的关键[11]。

2.2 特征和特征模型

特征(feature)是 Davis 在1982年首次提出的，并认为特征是从用户角度对系统的感知，用特征对系统需求规约进行模块化组织是一种非常自然的手段[12]。随后，针对领域，基于特征的各种方法的研究层出不穷，并获得了许多领域特征模型[2,13~16]。可以说，特征是为领域工程的复用者提供的，领域特征模型已被人们认为是把握领域共性(commonality)和变化性(variability)的主要手段。

对特征的定义在领域工程研究中不尽相同，本文采用文[17]中的多视角定义：从需求规约的组织结构角度来看，特征提供了一种对需求的分割和组织方式，即以特征作为需求空间内的一阶实体，系统具有的特征及其相互关系构成了系统的需求空间；从需求的内涵来看，一个特征体现了系统具有的某种能力或特点，反应了需求获取的参与者对系统的某种要求或理解；从需求的类型上看，一个特征可能是一种功能性的需求，或是对系统质量属性的要求，或者是外部环境对系统的某种约束条件。

此外，特征按内容可分为功能特征、质量特征、环境特征和表示特征。特征的多样性反应了特征对需求变化的支撑。

特征及其特征之间的关系构成了特征模型。特征模型通过使用抽象和细化的机制，对领域中不同应用的所有特征进行分类，既体现了领域共性和变化性的需求，也蕴涵了对领域知识的复用，是对用户需求的规约。领域特征模型通过记录领域中一组相对稳定的特征及其特征之间的关系来描述整个领域的需求。特征模型作为需求规约的有效模型，与 Use Case 模型之间也存在着密切的关系，具体参阅文[2]。

2.3 FODA

FODA(面向特征的领域分析)是将特征模型首次引入领域工程中的领域分析方法[13]。自此之后，各种面向特征的领域研究仅仅是对特征模型的结构做了一些剪裁、修改和扩充而已。也就是说，FODA 是以领域特征模型为核心的领域分析方法，它集中于对特征的收集、表示和分簇。同一问题域中的不同应用基于特征可以进行区分和比较。在开发可复用资产(软件体系结构和构件等)时，不仅要明确特征之间的关系，以及特征和可复用资产之间的关系，而且要明确对不同的应用哪些特征是必选的、哪些是可选的。这些内容在文[2]有较详尽的描述。

FODA 的过程由三个阶段组成[13]：上下文分析(context analysis)、领域建模(domain modeling)和体系结构建模(architecture modeling)。上下文分析的目的是确定领域范围；领域建模是提供应用表现出的共性和差异的分析，并产生一些领域模型；体系结构建模是提供领域的高层设计，如 DSSA 等。

2.4 构件及其内聚性

构件是系统中可以明确辨识的构成成分和结构单元，是软件功能设计和实现的承载体。所以，从系统的构成上看，任何在系统运行中能承担一定功能、发挥一定作用的软件体都可看成构件。

构件可分为原子构件和复合构件。原子构件是不可再分的基本单元；复合构件是由一组关系密切和相互协作的成员(可以是原子构件或复合构件)构件通过封装而成。可见，构件可大可小，可以组合或分解。小构件包括类构件和类簇构件等，而对象也是具有内部结构和行为能力的构件。在复杂的系统设计实现中，需要建立这种具有内部结构和行为能力的构件。其实，在面向对象的软件方法中，构件的基本形式就是对象，而它在不同的设计和运行环境、以及用于不同的目的时，又可表现为控件(control)、库、包、设计模式、框架、甚至 SA(Software Architecture)等不同多种形式。从程序设计角度看，构件可以看作是模块、类、对象或一个相关功能的集合等。本文中的构件是特指在领域工程范围内，基于面向对象技术的构件。

组成构件的各元素之间具有一定的关系，关系的密切程度就是构件的内聚性。构件的内聚性是信息隐蔽和局部化概念的自然扩展。构件的组合与分解是软件开发中常见的现象。也就是说，将关系较为密切的构件进行组合(体现高内聚性)，可增强基于构件开发的软件的可理解性、可测试性、可靠性和可维护性等质量属性。构件的高内聚往往意味着构件之间的底耦合，但实践表明内聚更为重要，应将注意力集中到提高构件的内聚程度上。

3 特征与构件关系的建立

既然构件是软件功能和实现的承载体，则与特征之间存在着天然的联系。一般情况下，构件所含的功能越多，构件的内部结构相对越复杂，同时构件的内聚度也相对偏小。可以说，特征模型对需求的规约最终会通过构件或构件的组合来体现。本节的主要任务是在特征模型的基础上，通过跟踪矩阵来建立构件与特征之间的跟踪关系。

3.1 面向特征的三级模型

需求工程将需求分为三类：业务需求、用户需求和功能需求。其中功能需求中记录的功能则是构成系统的基本元素，是实现业务需求和用户需求的载体。这三类需求都可能是特征出现的地方。文[17]以这三类需求为基础，考虑有效捕捉领域共性和变化性的因素，引入了需求的行为特征层，从而构造了一种特征模型，其主体部分如图1所示。

在图1中，服务(service)、功能(function)和行为特征(behavior)通过 WPA (Whole-Part Association)组成了特征模型的主体部分。除此之外，还有 Use Case、质量和约束，但都可以通过特征模型的主体部分来体现，它们之间关系细节参见文[17]。

此外，WPA 作为特征的基本组织方式，采用部分相对整体的可选性，可有效地再现特征的变化性。以特征作为需求空

间基本模块的观点和领域工程是一致的，这为我们建立特征与领域构件之间的关系提供了依据。

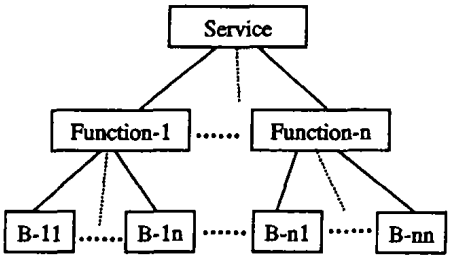

图1 特征模型的核心结构

3.2 特征跟踪矩阵

一个软件系统可由若干个构件通过 SA 或 DSSA 进行装配而成，软件系统呈现给外界的特征是通过构件来实现的。特征构成的特征集(区域)是对问题空间的有效分割和划分，这种划分为承载特征的构件的划分奠定了基础。其实，特征之间的耦合与紧密程度，反应了实现它的构件之间的耦合与紧密程度，加之特征的易获取性，为人们确定高内聚的构件提供了一条更为方便的途径。

更为重要的是，由于特征可以直接反应领域的共性和可变性，因此为构件的演化和 SA 的演化、乃至软件的演化与其波及效应分析提供有效的支持。下面首先通过示例来阐明什么是特征跟踪矩阵。

例如，一个起初由 $C1$、$C2$、$C3$、$C4$和 $C5$五个初始构件(可以是最简单的对象或功能非常单一的构件)组成的系统，实现的特征共有 $B1$、$B2$、$B3$三个。该系统的初始构件与行为特征之间的关系假设如图2所示。

	B1	B2	B3
C1	1	0	0
C2	1	1	0
C3	0	0	1
C4	1	0	0
C5	1	1	0

C1,C2,…,C5:Component

B1,B2,B3:behavior character

图2 特征跟踪矩阵

在图2中，位置为1时，说明当前行的构件承载或参与了当前列对应特征的实现，否则为0。将这种矩阵称为特征跟踪矩阵，并命名为矩阵 T。

可见同一特征可由若干个构件共同协同来实现，不同的特征也可映射到同一构件片段之上。

需要说明的是，在 FDD(Feature-Driven Development)方法[15]中，对贯穿于整个软件生命周期过程的特征给出了详尽的描述，并在设计阶段以一种很自然的方式，给出了拥有这些特征的构件(以类的形式)的确定方法。可以看出，构件与系统行为特征之间的关系无论在领域工程的需求阶段，还是实现阶段都是容易建立的，而且具有很好的一致性。这不仅为问题空间和解空间之间的平滑过渡提供了一致的词汇表达，而且为 SA 或 DSSA 与构件内聚程度的合理选择提供了支撑。本文是在假定特征与承载特征的初始构件(如简单的对象等)之间的关系已初步给定的前提下，根据需求的变化将关系密切的构件进一步组合并确保组合后构件的高内聚度。有关特征与实现特征的初始构件之间关系的具体确定过程，可参阅文[15]中的相关内容。

图2中仅仅是在特征模型的行为特征层面上记录了特征与初始构件间的关系。其实，这种关系的建立也可在特征模型的其它层面上进行。当然，建立了初始构件与行为特征间的关系之后，构件与其它层面间的关系也可根据特征模型容易确定。

4 高内聚领域构件控制

下面继续以特征为导线，在特征跟踪矩阵的基础上，通过建立能反应构件之间耦合程度距离矩阵，进而获取初始构件等级簇树，并给出具有高内聚性的构件组合控制策略。

4.1 构件间的距离矩阵

设 n 维空间上的向量 $V_1=(x_{11},x_{12},\cdots,x_{1n})$和 $V_2=(x_{21},x_{22},\cdots,x_{2n})$，则此两向量之间的距离：

$$D=\sqrt{\sum_{i=1}^{n}(x_{i1}-x_{i2})^2}$$

例如对于图2来说，将特征跟踪矩阵 T 中的每一行作为三维空间中的向量(称为构件向量)，并根据以上向量间距离计算公式，可计算各个构件向量之间的距离，从而形成一个矩阵(称为初始构件间的距离矩阵)，如图3所示。其中 $V1$，$V2$，$V3$，$V4$和 $V5$，分别代表初始构件 $C1$，$C2$，$C3$，$C4$和 $C5$对应的构件向量，且 $V1=(1,0,0)$，$V2=(1,1,0)$，$V3=(0,0,1)$，$V4=(1,0,0)$，$V5=(1,1,0)$，并将图3中的矩阵记为 $M_{5\times5}$。

	V1	V2	V3	V4	V5
V1	0				
V2	1	0			
V3	$\sqrt{2}$	$\sqrt{3}$	0		
V4	0	1	$\sqrt{2}$	0	
V5	1	0	$\sqrt{3}$	1	0

图3 由图2所得的距离矩阵

可以看出，n 维空间上的维度数实质上是领域中的特征数。在这个空间上，可以通过计算构件向量之间的距离来判断构件间的相近关系。也就是说，距离越近的构件，合并的可能性越大，且合并后的新构件的内聚性越好。

4.2 构件等级簇树的建立

有了以上初始构件间的距离矩阵，则可以根据等级簇聚合算法获得具有等级关系的初始构件等级簇树。

首先介绍等级簇聚合算法[18]。假设在 n 维空间上有向量 $k_1,k_2,\cdots,k_m$ 构成的距离矩阵为 $M_{m\times m}$，该算法将 $M_{m\times m}$作为输入，最终输出一棵具有等级关系的树。具体步骤描述如下：

①将每一个变量 $k_i(i=1,2,\cdots,m)$初始化为一个簇，并组成一个簇集合 C；

②如果簇集合 C 的长度为1，转④；否则继续下步；

③以簇中变量间的平均距离作为簇间的距离，合并距离最近的两个或多个簇为一个新簇，更新集合 C，并转②；

④结束。

例如，根据图3建立的初始构件等级簇树如图4所示。其中纵向虚线上所标的数字表示构件簇之间的距离；横向上表示不同层次的构件簇的划分。

结合图3中构件间的距离矩阵，图4中以初始构件为基础，形成等级簇树的形成过程是：

首先，由于 $V1$和 $V4$、$V2$和 $V5$之间的距离分别为0，所以分别构成构件簇 $V14$和 $V25$。其次，分别计算 $V14$、$V25$和 $V3$相互之间的距离，结果是：$V14$和 $V25$之间的距离为(1+1+1

+1)/4=1.0;$V14$和$V3$之间的距离为$(\sqrt{2}+\sqrt{2})/2=\sqrt{2}$;$V25$和$V3$之间的距离为$(\sqrt{3}+\sqrt{3})/2=\sqrt{3}$。由此可得,$V14$和$V25$之间的距离最近,所以合并成为构件簇$V1245$。最后,将$V1245$与$V3$合并为一个构件簇$V12345$,且其间的距离为$(2\sqrt{2}+2\sqrt{3})/4=1.57$。

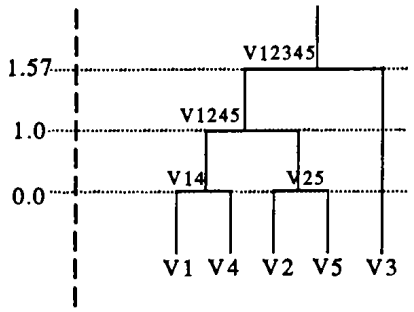

图4 构件等级簇树

但要注意的是,虽然在以上的简单例子中没有出现距离相近的构件簇存在多种组合的情况,但在等级簇的聚合算法中,这种现象经常出现。本文对此问题的处理办法是,可根据领域经验知识选取更加符合实际需求的组合即可。当然,也有其它的确定策略,望参阅文[19]。

4.3 高内聚构件变化控制的描述

通过构件等级簇树,构件之间的相似、相近或密切程度完全被"记录"到了其中,这样可以直接利用构件等级簇树来获取具有高内聚的构件,并确保了其它组合后的新构件的内聚度不会高于此。

有了构件等级簇树,则确定具有高内聚性构件变得非常简单。具体来说,首先根据对软件或SA抽象表达层次的需要,可在相应的横向层次上进行切分,从而形成由初始构件构成的构件簇,最后将各个构件簇进行组合和封装,形成新的、具有较合理的内聚度的构件。

显而易见,这样确定的构件,不但各自具有较高的内聚性,而且它们之间保持了合理的低偶合性。

例如在图4中,如果在横向标识为1.0的方向上进行构件的获取,则可得到由V1、V2、V4和V5这些初始构件组成的新构件,而V3构件自身保持不变。可以看出,在横向标识为1.0的方向上,这种组合获得的构件的内聚性是最好的。

5 相关研究

内聚的概念早在20世纪70年代由Stevens等人[9]提出,建议和主张无须精确确定内聚的级别,只要在设计中力争高内聚即可。按照他们的观点将内聚进行了分类[9,20],并给出了增大模块内聚的定性策略和评价准则,诸如使模块规模小而精的方法,尽量采用功能性联系法,慎用偶然性联系法、逻辑性联系法和顺序性联系法,以及将其它内聚转换为功能内聚以提高模块的内聚度等。80年代,具有典型意义的成果是,人们在程序切片概念[21]的基础上,利用源程序或控制流图或数据流图,探讨切片之间的关系,并研究出了多种高内聚模块的判定准则和度量方法[22,23]。这些方法的共同特点是只有等程序完成之后,基于功能进行模块内聚度判定的。到了90年代,基于软件度量学,人们大量开展了面向对象系统的内聚度度量[24~28]研究,这些度量都是基于访问或共享实例变量、或者通过类中各种方法与属性以及它们之间关系的定量度量,并提出了优良度量准则所具有的一些性质[26]等。实际上,在软件再工程,特别是遗留系统的现代化改造[29]研究中,人们自觉不自觉地对内聚进行着"间接"的研究和利用。诸如文[30]中,作者阐述了通过收集分散在遗产系统中与某个可复用特征相关的代码并将其封装为一个构件的过程,隐藏地利用了"内聚"的思想。

进入21世纪,随着软件规模的扩大和应用的进一步普及,人们对软件质量和性能提出了更高的要求,特别是在软件质量评价方面更追求科学性和准确性。而"高内聚"是获得软件独立性、可理解性、可移植性、易维护性和易复用性等诸多软件属性的基础,因此,内聚度度量成了软件度量的重要组成部分[31]。

结束语 首先要说明的是,本文对初始构件如何获取没有给出任何的策略,且对特征在构件中如何分配也没有作更多的说明,只是假设在设计初期阶段,已经初步建立了最简单初始构件(如简单的对象)与特征之间基本关系的前提下,在面向对象的领域工程范围内,利用特征给出了一种基于这种初始构件的高内聚性领域构件获取方法,使得获取的构件在内聚性上具有较好的合理性。

基于特征之间的关系确定了构件在组合与变化时的内聚关系,这不仅使人们能在分析和设计阶段对提前关注到内聚,而且为遗留系统的现代化改造给予了支持,也为特征的验证提供了一条可行途径。此外,由于特征能较好地反映领域的共性和变化性,显然,从需求变化到特征变化直到软件演化,这种传递也刚好能通过特征跟踪矩阵、构件间的距离矩阵和构件等级簇树来记载和表达,为SA演化乃至软件演化中的变化跟踪提供了一种有效途径。另外,本文虽然表面上只是建立了构件与特征之间的关系,并给出了高内聚领域构件的确定方法,实质上也给出一种构件内聚度相对大小的数值确定模型,为构件内聚度的定量研究提供了一条有效途径。

对于面向对象方法,对象作为抽象数据类型封装了完成多个功能的操作,功能内聚判定方法不再适应于它。基于特征的领域构件高内聚控制方法较好地解决了这一问题。

进一步的研究包括:如何基于构件等级簇树,获取构件内聚度度量的数学模型;如何将本文的高内聚领域构件确定策略融合到SA演化研究中,并如何与FDD方法相结合等等。

致谢 非常感谢北京大学杨芙清院士提供良好的博士后研究条件。

参考文献

1 Guillermo A, Ruben P D. Domain analysis concepts and research directions. In: R Prieto Diaz and G. Arango, eds. Domain Analysis and Software System Modeling. Los Alamitos, California: IEEE Computer Society Press, 1991
2 Griss M L, Favaro J, d'Alessandro M. Integrating feature modeling with the RSEB. In: Devanbu P, Poulin J, eds. Proc. of the 15th Intl. Conf. on Software Reuse. Victoria: IEEE Computer Society, 1998. 76~85
3 Turner CR, Fuggetta A, Lavazza L, Wolf A L. A conceptual basis for feature wngineering. Journal fo Systems and Software, 1999, 49(1): 3~15
4 Len B, Paul C, Rick K. Software architecture in practice. 2nd ed. New York: Pearson Education, Inc., 2003
5 王映辉,刘瑜,王立福. 基于不动点转移的SA动态演化模型. 计算机学报,2004,27(11):1451~1456
6 Robert C S, Daniel P, Grace A L, Modernizing legacy system. New York: Pearson Education, Inc., 2003
7 Mehta A, Heineman GT. Evolving legacy system features into fine-grained components. In: Proc. of the 24th Intl. Conf. on System Engineering. ACM, 2002. 417~427
8 张海藩. 软件工程导论. 北京:清华大学出版社,2002
9 Stevens P, Myers J, Constantine L. Structure design. IBM System Journal, 1974, 132(2): 115~139
10 弓惠生. 模块内聚性的度量方法. 计算机研究与发展,1997,34(8): 571~576
11 Ruben P D. Status Report: Software Reusability. IEEE Software, 1993, 10(3): 61~66

12 Davis A M. The design of a family of application-oriented requirements languages. Computer,1982,15(5):21～28
13 Kang K C, Cohen S G, Hess J A, et al. Feature-Oriented domain analysis (FODA) feasibility study:[Technical Report, CMU/SEI-90-TR-21]. Pittsburgh: Carnegie Mellon University, Software Engineering Institute,1990. 1～52
14 Jacobson I, Christeron M, Jonsson P, Overgaard G. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1992. 123～159
15 Stephen R P,John M F. A practical guide to feature-driven development. Prentice-Hall Inc. ,2002
16 Chastek G, Donohoe P, Kang KC, Thiel S. Product line analysis: a practical introduction:[Technical Report, CMU/SEI-2001-TR-001]. Pittsburgh: Carnegie Mellon University, Software Engineering Institute,2001. 1～42
17 张伟,梅宏. 一种面向特征的领域模型及其建模过程. 软件学报,2003,14(8):1345～1356
18 Wiggert T A. Using clustering algorithms in legacy systems remodularization. In:IEEE ed. Proc. of the Fourth Working Conference of Reverse Engineering. Washington:IEEE,1997. 33～43
19 Siff M,Reps T. Identifying modules via concept analysis. In:IEEE ed. , Proc. of Intl. Conf. Software Maintenance. Washington: IEEE,1997. 170～179
20 Yourdon E,Constantine L L. Structure Design:Fundamentals of a Discipline of Computer Program and Systems Design. Yourdon Press,1978
21 Weiser D. Program slicing. In:Proc. 5th Conf Software Engineering,California,1981. 439～449
22 Emerson J. A discriminant metrics for module cohesion. In:Proc. 7th Conf Software Engineering,Florida,1984. 294～303
23 Ott M,Thuss J. The relationship between slices and module cohesion. In:Proc. 11th Conf. Software Engineering,California,1989. 198～204
24 Bieman J M,Kang B K. Cohesion and reuse in an objected-oriented system. In: Proc. Of the Symposium on Software Reusability (SSR'95),Seattle,WA,1995. 259～262
25 Chae H S,Kwon Y R. A cohesion measure for classes in objected-oriented system. In: Proc. of 5th International Software Metrics Symposium,Bethesda,MD,1998. 158～166
26 Briend L C,Daly J W. A unified framework for coupling measurement in object-oriented systems. Empirical Software Engineering, 1998,3(1): 65～117
27 李心科,刘宗田,潘飚,刑大红. 一个面向对象软件度量工具的实现和度量实验研究. 计算机学报,2000,23(11):1220～1225
28 陈振强,徐宝文. 一种基于依赖性分析的类内聚度度量方法. 软件学报,2003,14(11):1849～1856
29 Robert C S,Daniel P,Grace A L. Modernizing Legacy Systems: Software Techologies,Engineering Processes and Business Practices. Addison-Wesley,2003
30 Mehta A,Heineman G T. Evolving legacy system features into fine-grained components. In: ACM ed. , Proc. of the 24th Intl. Conf. on Software Engineering. ACM,2002. 417～427
31 Stephen H K. Metrics and Models in Software Quality Engineering. Addison-Wesley,2002

(上接第184页)

噪声的 cameraman 图像进行边缘检测。检测结果如图4。其分别是加噪图像应用 WFCE、Sobel 算子、Canny 算子和模糊竞争算法得到的边缘图像。图4(c)大尺度下的边缘有较少的噪声点,并且细节保留得很好。这正是小波多尺度的优点所在。Sobel 算子检验出的边缘噪声少,可细节丢失了很多;图4(e)、图4(f)轮廓清晰但噪声干扰太严重。由此可见,小波多尺度模糊竞争算法(WFCE)的实验结果是理想的,保留细节的同时对噪声有较好的抑制功能。

结论 针对图像边缘不同的灰度变化方式,为细化边缘和提高图像的抗噪性,本文提出了小波多尺度模糊竞争边缘检测算法(WFCE)。算法融合小波多尺度和模糊理论,利用竞争法则有效地选取边缘点。算法对于图像边缘有理想的检测效果,同时有较高的抗噪性。实验证明,算法有效、可行。

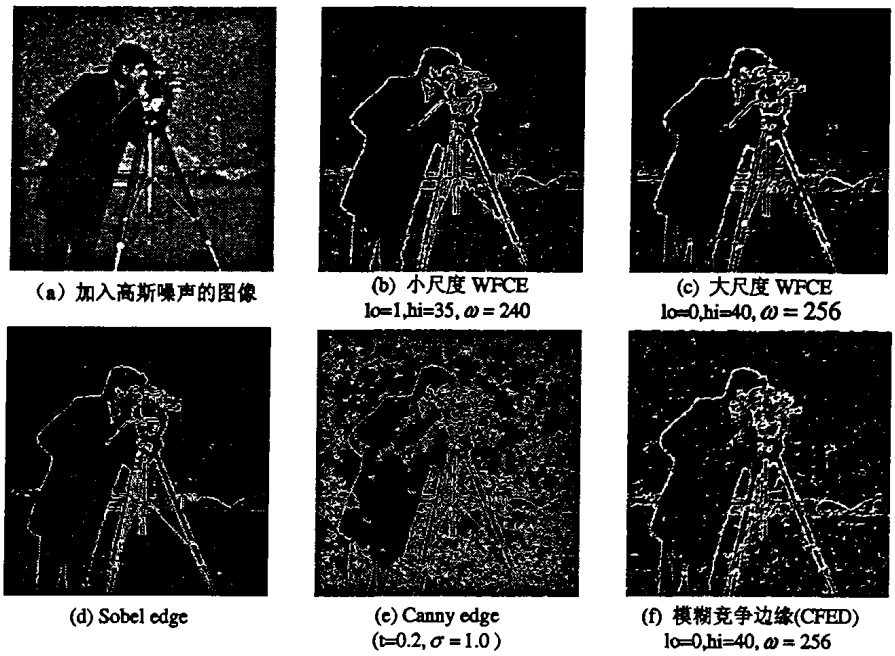

图4 含噪 cameraman 的 cameraman 图像边缘

参 考 文 献

1 Mike H,Sudeep S,Thomas S,et al. Comparison of Edge Detectors: A Methodology and Initial Study[J]. Computer Vision and Image Understanding,1998,69(1):38～54
2 Canny J A. Computational approach to edge detection[J]. IEEE Trans PAMI, 1986,8(6):679～698
3 Mallat S, Zhong S. Characterization of signals from multi-scale edges[J]. IEEE Trans PAMI,1992,14(9):710～732
4 杨丹,张小洪. 基于小波多尺度积的边缘检测算法[J]. 计算机科学,2004,31(1):133～135
5 Lu Siwei,Wang Ziqing,Shen Jun. Neuro-fuzzy synergism to the intelligent system for edge detection and enhancement[J]. Pattern Recognition,2003,36:2395～2409
6 Liang L R,Looney C G. Competitive Fuzzy Edge Detection[J]. Applied soft computing,2003,3:123～137
7 Looney C G. Radial basis Function Link nets and fuzzy reasoning [J]. Neurocomputing,2002,48:489～509