维普资讯 http://www.cqvip.com

用于移动计算的软构件模型及其实现*)

刘 波 卢炎生 徐丽萍
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 提出了一个可上下文感知的用于移动计算环境的构件模型 MCCM。首先讨论了用于移动计算环境的构件应具备的条件,然后提出了 MCCM 模型及其描述;接着给出了在 Linux 平台上的一个具体实现,并对这一实现进行了性能测试;最后同当前相关工作进行了对比,并指出了进一步的研究方向。

关键词 构件模型,移动计算,上下文感知,反射,中间件

Component Model for Mobile Computing

LIU Bo LU Yan-Sheng XU Li-Ping
(Department of Computer Science and Technology, Huazhong University of Sci. & Tec., Wuhan 430074)

Abstract A context-aware component model (MCCM) for mobile computing environment is described. First the characteristic of component model for mobile computing environment are considered. Then a context-aware component model for mobile computing environment is proposed, and description is specified. After that the implementation of MCCM on Linux platform is analyzed. Finally, conclusion and further research directions are given.

Keywords Component model, Mobile computing, Context-awareness, Reflection, Middleware

1 引言

在过去的十多年中,移动计算的硬件技术获得了快速的发展,目前最新的移动设备的处理能力已经接近 GHz 的水平,存储能力也得到了很大的提高。但是与之相对应的软件设计技术还停留在简单的基于过程和少数的基于对象的方法之上。与之相矛盾的是,个人移动电子产品更新换代快,要求较短的软件开发周期;同时,个人移动电子产品一般产量很大,对成本十分敏感,为了提高产品竞争力,必须控制好软件的开发和维护费用。因此,移动计算的软件工程的相对落后,严重地制约了移动计算技术的进一步发展。

反观桌面 PC 领域,软件设计技术曾经同样遇到过类似的问题。通过研究人员多年的研究,桌面软件的设计技术得到了迅速的发展,从最开始的基于过程到后来的基于对象、面向对象,再到目前的基于构件的软件设计技术,很好地解决了这一问题。目前主流的构件体系是 COM[1]、CORBA[2] 和 J2EE[3]。它们在桌面 PC 上获得了巨大的成功,极大地减轻了应用程序员的开发负担,提高了桌面软件的开发效率、可靠性和可维护性。虽然在 PC 和企业级应用上均获得了成功,但这些传统的构件体系不具备轻量级、可上下文感知等特点,并不适合移动计算的要求。因此,为了克服移动计算软件开发面临的困境,一种适用于移动计算环境的构件模型是必须的。

本文提出了一种用于移动计算的可上下文感知的构件模型 MCCM(Mobile Context-aware Component Model)。MCCM 模型具有轻量级、可上下文感知等特点,能较好地用于移动计算环境。

2 相关的工作

移动计算软件技术的研究主要集中在两方面:移动计算中间件技术的研究和用于移动计算的软构件技术的研究。

移动计算中间件技术的研究分为两个方向。一是通过改进传统的中间件以适应移动计算环境的需求。典型的代表有 ALICE[17]、Rover[18]、Mobiware[19] 和 L2imbo[20] 等。这种改进解决了移动中间件所需的轻量级、异步通信等需求,但是由于其侧重点在于解决如何从主干网提供服务给其它移动设备,因此并不适合 Ad-hoc[21] 环境。同时也不具备上下文感知功能,不适合上下文变化频繁的移动计算系统。另一方向是带上下文感知的反射式中间件。典型代表有 PKUAS[22]、ARTeMIS[23]、Compose[24]、OpenORB[25] 和 CARISMA[26] 等。这些中间件系统能较好地用于移动计算环境,为移动应用程序开发者提供一个统一界面,屏蔽了底层操作系统和硬件的细节,降低了移动应用的开发复杂度。但没有涉及到如何提高移动应用程序的开发效率和复用程度等方面的内容。

目前在用于移动计算的软构件技术上的研究还比较少。有一些用于嵌入式系统的构件模型,如 PBO[27]、PECOS[28]、SEESCOA[29] 和 KOALA[30] 等。这些构件模型虽然考虑了用于移动计算所需的轻量级条件,但并没有考虑移动计算环境下的其它各种需求。另外,英国 Lancaster 大学的 ReMMoC[31] 系统,虽然提出了一个用于移动计算的构件模型,但只限用于建构移动计算中间件,并未涉及建构移动应用程序。

3 一种用于移动计算的软构件模型—MCCM

3.1 移动计算软构件的设计需求

由于移动计算环境的特殊性,使得许多传统的软构件模

*)预研基金项目(10104010201)。**刘 波** 博士研究生,研究方向为构件化软件、中间件、移动计算;**卢炎生** 教授,博士生导师,主要研究方向为特种数据库、构件化软件、中间件、软件测试、移动计算;**徐丽萍** 博士,副教授,主要研究方向为构件化软件、中间件、移动计算。

维普资讯 http://www.cqvip.com

型不再适用。用于移动计算的构件模型面临着一些不同于传统构件模型所面临的特殊需求。

轻量级[4]。移动应用程序运行在资源相对紧张的移动设备上，如小容量内存、低速处理器、有限的电池容量等。目前传统的构件体系(COM、CORBA、J2EE)非功能性服务很多，导致体积庞大、运行开销昂贵，显然不适合移动环境。用于移动计算的构件必须是轻量级的。

上下文感知[4]。不同于固定系统，移动系统运行在一个上下文变化很频繁的环境中。不稳定的网络带宽、服务可用性的变化、用户所处的不同情境、电池的剩余电量等，都对应用程序的服务质量有很大的影响。为了满足用户的需求，用于移动计算的构件必须能够动态适应上下文状态的变化，尽可能地给用户提供高质量的服务。

可接受用户干预[5]。移动设备的用户需求和偏好千差万别。有些用户可能为了延长电池的使用时间而愿意接受视觉效果较差的黑白图像。而有些用户则可能更愿意消耗较多的电量来获得较好的彩色视觉效果。因此，构件应能根据用户的不同需求做出适当的反应。

动态定制。对于不同的上下文环境，用于移动计算的构件应该有不同的指令集合进行处理，每一个指令集合对应处理一种或多种上下文环境。由于上下文环境状态的多样性，有可能导致指令集合数目庞大。在资源相对稀有的移动设备上，将所有的指令集合静态地装入或卸出内存是不合适的。因此，用于移动计算的构件必须有运行时动态定制功能。

带惯性的上下文感知。构件在对上下文环境的变化进行适应时必须保持惯性。即这种适应不能只是从上下文环境到构件行为的简单函数映射，还必须同时考虑当前的构件行为。仍然以图像质量和电池电量为例。当电池电量降低，为减少电池消耗，图像从32位真彩色经由16位色、256色最后到黑白图像，比直接从32位真彩色立刻切换到黑白图像更容易让用户接受。无需上下文感知的任务无额外开销。移动计算中同样包含有无需上下文感知的任务，如数值计算等。为了保证构件模型有较广的适用范围，模型必须支持无需上下文感知的任务，并且无需上下文感知的任务无额外开销(相对上下文感知部分)。

本文提出的MCCM模型很好地满足了上面提到的各种需求。

3.2 理论基础

3.2.1 反射技术

为了实现具有上下文感知的自适应能力，MCCM构件采用了反射技术。

反射技术最早由Smith于20世纪80年代初提出，并用于设计更开放、可扩展的程序设计语言系统3{Lisp[6,7]。其主要思想是提供一个元接口，以支持可查性(inspection)和可适应性(adaptation)。为了提供对自身的可查性，需要有一个具体化(reification)[8]的过程。具体化有效地将内部表示的某些方面显露出来，实现从基层到元层的映射。与之相对应的是吸收(absorption)过程，使基层能够根据元层的改变做出相应的调整，这一过程就是反射过程。其中基层和元层是因果相连(causally connected)[9]的。这里所谓因果相连是指，对其中任意一方的改变都会引起另一方的改变(如图1)。因此，上层应用可通过对元层的访问实现对底层的可查性和可适应性。

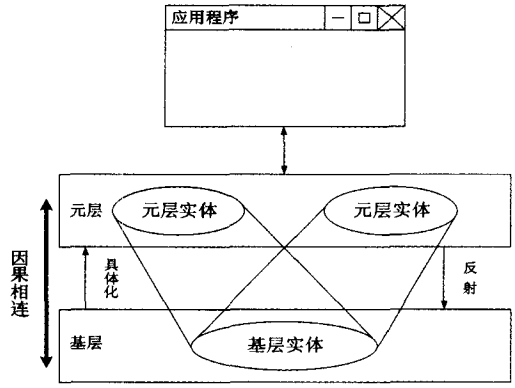

图1 具体化与反射

3.2.2 Moore机

Moore机[10]是一种对应处理输入串的过程中经过的每一个状态，都有一个字符输出的有穷状态自动机。

Moore机是一个六元组：

$M=(Q,\Sigma,\Delta,\delta,\lambda,q_0)$，其中：

Q：状态的非空有穷集合。任意q属于Q，称为M的一个状态(state)。

Σ：输入字母表(input alphabet)。

Δ：输出字母表(output alphabet)。

δ：状态转移函数(transition function)。$\delta:Q\times\Sigma\rightarrow Q$，对任意(q, a)属于$Q\times\Sigma$，$\delta(q,a)=p$表示M在状态q读入字符a，将状态变为p，并将读头向右移动一格指向下一个字符。

λ：为输出函数。对任意q属于Q，$\lambda(q)=a$表示M在状态q时输出为a。

q_0：M的开始状态(initial state)。q_0属于Q。

3.3 MCCM模型

MCCM是一种用于移动计算的构件模型，采用多层反射[9]机制实现可上下文感知的移动系统(如图2)。其中MCCM构件构成应用层和元元层，操作系统和(或)反射式中间件构成基层和元层。元元层、元层和基层分别因果相连。因为元元层、元层和基层是因果相连的，所以MCCM构件对元元层的访问和修改可以感知上下文环境，并同时在应用层、中间件和(或)操作系统层做出相应的适应动作。同样，反射式中间件[9,11,25,13]层也可以感知上下文环境并做出适应动作。这样，系统便同时实现了应用层和中间件层的反射。

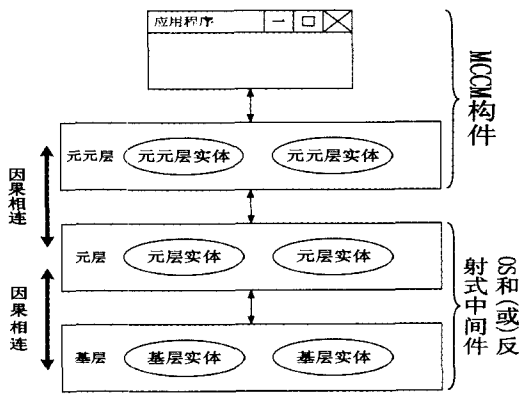

图2 使用MCCM构件的移动系统

MCCM构件结构如图3。MCCM构件由一个普通功能集和一个上下文感知功能集构成。功能集均可为空。每一个上下文感知功能由一个上下文感知适配器和一个动作集组

维普资讯 http://www.cqvip.com

成。上下文感知适配器由用户偏好配置和本功能感兴趣的上下文状态集组成。由上下文感知适配器来决定使用动作集中的哪个动作。当动作集为空时，其所对应的上下文感知功能称为平凡上下文感知功能。

每一非平凡上下文感知功能在任一时刻都与其所含动作集中一动作逻辑相连，这一动作称为活跃动作。每当上下文感知功能感兴趣的上下文状态发生变化时，上下文适配器根据用户偏好配置、上下文状态及当前活跃动作得出下一活跃动作。这样产生的适应上下文变化的活跃动作参考了前一活跃动作，是具有惯性的。

每一普通功能简单地对应一个动作。

MCCM 构件的普通功能集和上下文感知功能集均为空，或普通功能集为空而上下文感知功能集中的元素均为平凡功能时，称为空构件。空构件没有任何逻辑功能。

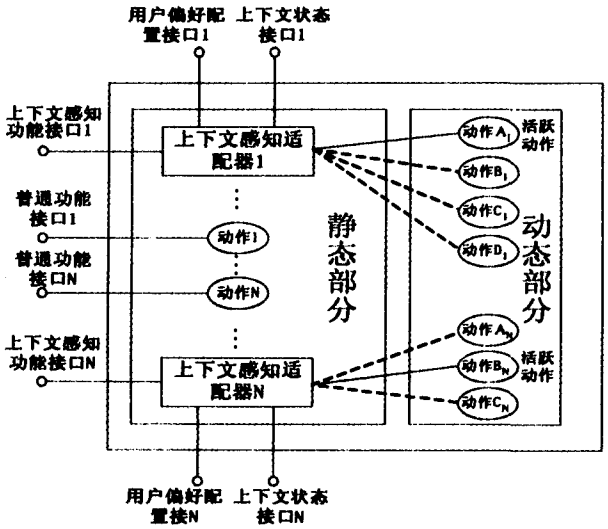

图 3　MCCM 构件结构

MCCM 构件对应的 BNF 描述：

MCCM 构件::=普通功能集 上下文感知功能集

普通功能集::=普通功能 普通功能集|ε

普通功能::=动作

上下文感知功能集::=上下文感知功能 上下文感知功能集|ε

上下文感知功能::=上下文感知适配器 动作集

动作集::=动作 动作集|ε

上下文感知适配器::= 用户偏好配置 感兴趣的上下文状态集

上下文状态集::=上下文状态集 上下文状态 | 上下文状态

因为上下文感知功能的动作变迁必须保持惯性，所以不能简单地用从上下文状态集到动作集的函数映射对 MCCM 构件的动作变迁进行描述。本文采用 Moore 机进行描述。

每个非平凡的上下文感知功能都可用一个 Moore 机进行描述：

$M=(Q,\Sigma,\Delta,\delta,\lambda,q_0)$，其中：

Q：非平凡上下文感知功能对应的动作集，作为 Moore 机的状态集。

Σ：感兴趣的上下文状态集，作为 Moore 机的输入符号集。

Δ：同 Q，动作集既是状态集也是输出符集。

δ：用户偏好配置，作为 Moore 机的状态转移函数。

λ：恒等映射。转换为什么状态就输出什么状态。

q_0：非平凡上下文感知功能的初始动作。

3.4　一个 MCCM 例子

假设某 MCCM 构件提供了一个带上下文感知的定时提醒功能，在某一指定的时间提醒用户。该功能感兴趣的上下文因素包括电池剩余电量和外界的物理噪音。剩余电量和外界噪音均分为高中低三种状态。该功能可能以三种方式之一提醒用户：响铃、振动和响铃且振动。当用户启动该构件的定时提醒功能时，构件会提醒用户输入用户偏好配置，或确定使用缺省用户偏好配置。当该功能启动后，用户仍然可以通过构件提供的相应接口对用户偏好配置进行动态修改。在本例中：

$Q=\Delta=\{$响铃，振动，响铃且振动$\}$

$\Sigma=\{$(高噪音，高电量)，(高噪音，中电量)，(高噪音，低电量)，(中噪音，高电量)，(中噪音，中电量)，(中噪音，低电量)，(低噪音，高电量)，(低噪音，中电量)，(低噪音，低电量)$\}$

q_0 设为响铃，λ 为 $Q\rightarrow\Delta$ 的恒等映射。

一个可能的用户偏好如表 1。

表 1　一个可能的用户偏好

电量＼噪音	高	中	低
高	振动	振动且响铃	振动且响铃
中	振动	响铃	响铃
低	振动	响铃	响铃

表 1 表示在相应上下文状态下用户所期望的构件的行为方式。但是在出现某一上下文状态时，构件行为方式有可能不会马上转变为用户所期望的行为方式。在该例中，表现为提高电量消耗的行为转变只能逐步进行，即不能直接从响铃转变为响铃且振动，必须先转变为振动方式。

为了便于表示，对上面 Σ 中的元素从 1 到 9 进行编号。状态转换函数 δ 表示如图 4。

4　MCCM 模型在 Linux 平台上的实现

为了实现一个二进制构件，首先要考虑两个问题[14]：第一，该构件在目标操作系统平台上以哪种形式存在；第二，构件和客户程序之间如何进行通讯。本文在 Linux 平台上用 C^{++} 语言实现了对 MCCM 构件模型的支持。

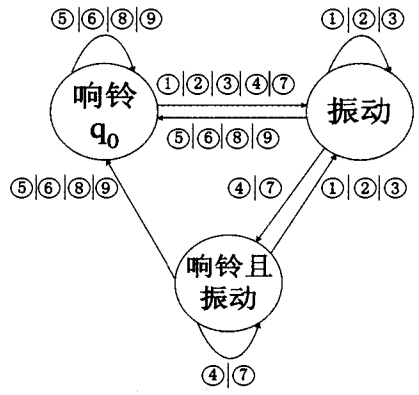

图 4　状态转换函数

本文采用共享对象(shared object)和静态库相结合的方式实现 MCCM 构件。其中上下文感知适配器和与普通功能

维普资讯 http://www.cqvip.com

接口静态相连的动作以静态库的形式存在，MCCM 构件用户可以将 MCCM 构件静态链接进客户程序，并直接使用。与上下文感知功能接口相关的动作以共享对象的形式存在，上下文感知适配器可以根据前一活跃动作、当前上下文状态以及运行时管理策略来决定动作集中动作的动态装入和卸出。

为了使客户程序能够方便高效地与 MCCM 构件通讯，本文给每一动作分配一个全球唯一标志(Globle Unique ID)，并提供一个管理动作的动态库。该动态库提供了对每个动作的注册功能，注册信息包括动作在文件系统中的位置、GUID、可读功能描述等。同时提供了对动作的查询、动态装入和卸出等函数接口。客户程序使用 MCCM 构件时只需设置好用户偏好配置，就可以调用上下文感知功能接口和普通接口。上下文感知适配器会自动根据通过上下文状态接口从操作系统或反射式中间件传入的上下文状态信息，利用管理动作的动态库查询动作注册信息并动态装入或卸出相关动作。这样客户程序调用的始终是当前的活跃动作，而应用程序员完全不用考虑上下文感知的相关问题。反射式中间件向应用层(MCCM 构件)提供上下文状态信息的方法在文[15, 16] 中进行了讨论。本文的实现采用异步事件通知的方式。

5 MCCM 模型的性能测试

本节主要对比测试使用 MCCM 构件在时间和空间上的开销，以及当上下文状态发生改变时，活跃动作发生变迁所需的时间开销。

作者制作了一个 MCCM 构件，包含一个上下文感知功能和一个普通功能。上下文感知功能名为 Greet，根据用户位置在办公室还是在家决定输出“Hello, boss!”还是“Hello, darling!”。普通功能名为“HelloWorld”，简单地输出“Hello world!”。

不用构件的无上下文感知功能的普通 C^{++} 程序主要由一个类组成，这个类提供了与上面对应的三个成员函数，分别为“GreetBoss”、“GreetDarling”和“HelloWorld”。

在奔Ⅲ800 处理器、256MB 内存的 PC 机上对 MCCM 进行了测试(当前最新的移动设备的处理能力和内存空间均可以达到这个水平)，测试结果如表 2。

表 2 测试结果

	文件大小(字节)	运行时间(微秒)
普通程序	7965	GreetBoss: 35 GreetDarling:35 HelloWorld: 35
使用 MCCM 构件的程序	主程序:5072 动作 1:3619 动作 2:3623	Greet:37 HelloWorld:35

在空间开销方面，由于在某一时刻只需载入一个动作的共享对象文件，因此使用 MCCM 构件的内存的额外开销约为 9.1%。使用 MCCM 构件导致的外存额外开销约为 54.6%。在本例中，逻辑功能很简单，只是输出一条简短的语句，而在实际应用当中，构件的逻辑功能相对于支持构件运行的固定部分应当是十分复杂的，所以在实际应用当中，额外的空间开销应当远低于本例中的测量值。内存的额外开销甚至有可能为负值，因为 MCCM 构件可以把大量暂时处于非活跃状态的动作卸出内存。

在时间开销方面，MCCM 构件的普通功能表现出了不用构件时同样的效率。这表明 MCCM 构件模型保证了无需上下文感知的任务无额外开销。MCCM 构件的上下文感知功能比不用构件时效率下降了 5.7%。这主要是因为 MCCM 构件模型利用了 C^{++} 对象模型中的 vtable 机制，造成了额外一层间接寻址所引起的。同样，在实际的较复杂应用中，逻辑功能所消耗的时间会远高于这一层间接寻址所消耗的时间，因此实际的时间开销应低于本例中的测量值。

最后，模拟操作系统和(或)反射式中间件向 MCCM 构件报告上下文状态的改变，并测量了构件切换活跃动作所需的时间。多次测量表明：如果切换至的动作是第一次成为活跃动作，大概需要 25300 微秒；如果不是第一次则只需要约 185 微秒。这是因为第一次成为活跃动作时，操作系统必须到外存进行读取。如果曾经成为活跃动作被装入过内存，即便后来被其他动作替代并“卸出”内存，由于 Linux 操作系统良好的缓存机制，仍然可以保持较好的性能。由于实际应用中的构件逻辑比较复杂，各动作的共享对象文件比较大，所以实际应用中的切换操作所需时间应该高于作者的测量值。

结论和今后的工作 本文提出的 MCCM 模型具有轻量级、可上下文感知等特点，可以根据用户的不同偏好和上下文状态的变化来动态调节自身的行为，能根据需要动态地装入或卸出各种动作，能够较好地用于移动计算环境。

光有一个构件模型，还不足以成为一个完整的移动应用开发框架。下一步的工作包括开发新的或改进现有的移动计算中间件[5, 15, 16]以提供上下文状态信息给 MCCM 构件，使移动中间件系统和 MCCM 构件有机地结合起来，形成一套完整的移动应用开发框架。

MCCM 构件本身还有一些有待加强的地方。计算机系统的安全性是非常重要的，对于移动计算系统更是如此，而 MCCM 模型还没有这方面的考虑。动作集中动作的运行时管理策略也是一个值得继续研究的问题，何时装入或卸出一个动作、装入或卸出哪个动作、保留多少动作在内存当中等，这些都将很大程度地影响 MCCM 构件的性能。一个较好的运行时管理策略不仅要依赖于用户偏好、硬件资源，同样也要受上下文状态的影响。另外，增强用户偏好配置接口的友好性也是今后的一个研究方向。

参 考 文 献

1 Pan Aimin. COM principle and application [M]. Beijing : Tshinghua University Publisher, 1999

2 The Object Management Group. The Common Object Request Broker Architecture and Specification. [EB/OL]. http://www.omg.org, 1998-10-10/1999-06-20

3 Sun Microsystems Inc. JavaTM 2 Platform (Enterprise Edition) [EB/OL]. http://java.sun.com/j2ee/

4 Mascolo C, Capra L, Emmerich W. Middleware for Mobile Computing(A Survey). In: Networking 2002 Tutorial Papers2002, vol2497 of LNCS, Springer. 20～58

5 Capra L, Emmerich W. CARISMA: Context-aware reflective middleware system. IEEE Transactions on Software Engineering, 2003, 29(10): 929～945

6 Smith B C. Reflection and semantics in a procedural language. MIT Laboratory of Computer Science, USA: [Technical Report 272]. 1982

7 Smith B C. Reflection and semantics in Lisp. In: Proceedings of the 14th Annual ACM Symposiumon Principles of Programming Languages, Salt Lake City, 1984. 23～35

8 Friedman D. Reification: Reflection without meta-physics. In: Proceedings of the 1984 ACM Symposium on LISP and Functional Programming, Texas, 1984. 348～355

维普资讯 http://www.cqvip.com

9 胡海洋,马晓星,陶先平,等. 反射中间件的研究与进展. 计算机学报,2005,28(9): 1407～1420
10 Hopcroft J E, Ullman J D. Introduction to Automata Theory, Languages, and Computation. Wesley Publishing Company,1979
11 Blair G S, Coulson G, Robin P, et al. An architecture for next generation middleware. In: Proceedings of the Middleware'98, LakeDistrict, UK, 1998. 191～206
12 Coulson G, Blair G S, Clarke M, et al. The design of a highly configurable and reconfigurable middleware platform. Journal of ACM Distributed Computing,2002,15(2): 109～126
13 Fabio K, Fabio C, Gordon B, et al. The case for reflective middleware. Communications of the ACM,2002,45(6): 33～38
14 潘爱民,徐辉. Linux平台下的组件对象模型及其实现. 小型微型计算机系统,2004,25(9):1707～1711
15 Chan T S, Chuang Siu-Nam. MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing. IEEE Transactions on Software Engineering, 2003,29(12): 1072～1085
16 Welling G, Badrinath B. An Architecture for Exporting Enviroment Awareness to Mobile Computing. IEEE Transactions on Software Engineering, 1998,24(5): 391～400
17 Haahr M, Cunningham R, Cahill V. Supporting CORBA Applications in a Mobile Environment (ALICE). In: 5th Int Conf. on Mobile Computing and Networking (MobiCom). ACM Press, August 1999
18 Joseph A D, Tauber J A, Kaashoek M F. Mobile Comuting with the Rover Toolkit. IEEE Transactions on Computers, 1997, 46 (3)
19 Angin O, Campbell A, Kounavis M, et al. The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking. Personal Communications Magazine, Special Issue on Adapting to Network and Client Variability. IEEE Computer Society Press, August 1998
20 Davies N, Friday A, Wade S, et al. L2imbo: A Distributed Systems Platform for Mobile Computing. ACM Mobile Networks and Applications (MONET), Special Issue on Protocols and Software Paradigms of Mobile Networks, 1998, 3(2)
21 Perkins C. Ad-hoc Networking. Addison-Wesley, Jan. 2001
22 黄罡,王千祥,梅宏,等. 基于软件体系结构的反射式中间件研究. 软件学报,2003,14(11):1819～1826
23 陶先平,吕建,马晓星. ARTeMIS: 基于 agent 的多模式协同中间件. 南京:南京大学软件新技术国家重点实验室:[技术报告 BG2001012]. 2003
24 Nalini V, Mayur D, Shivajit M, et al. Design & implementation of a composable reflective middleware framework. In: Proceedings of the IEEE International Conference on Distributed Computer System, Arizona, 2001. 644～653
25 Coulson G, Blair G S, Clarke M, et al. The design of a highly configurable and reconfigurable middleware platform. Journal of ACM Distributed Computing, 2002, 15(2): 109～126
26 Capra L, Emmerich W. CARISMA: Contextaware reflective middlware system. IEEE Transactions on Software Engineering, 2003, 29(10): 929～945
27 Wall A, Norstr C. A Component Model for Embedded Real-Time Software Product-Lines. In: 4th IFAC Conference on Field Bus Systems and Their Applications, Nancy(France), 2001
28 Genβler T, Nierstrasz O, Schönhage B. Components for Embedded Software The PECOS Approach. In: Proc International Conference on Compilers, Architectures and Synthesis for Embedded Systems, 2002
29 Vandewoude Y, Berbers Y. Run-time evolution for embedded component-oriented systems. In: Werner B, ed. Proceedings International Conference on Software Maintenance, IEEE Computer Society, 2002. 242～245
30 van Ommering R. Building Product Populations with Software Components. International Conference on Software Engineering (ICSE2002), Orlando, Florida, US, May2002
31 Clarke M, Blair G S, Coulson G, et al. An Efficient Component Model for the Construction of Adaptive Middleware. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware'2001), Heidelberg, Germany, November 2001. 160～178

(上接第 248 页)

本算法具有较强的抗干扰能力,这是因为干扰在图像中表现为孤立的像素点,而本算法利用到大量的像素点的空间排列及统计信息,所以孤立的像素点对本算法结果影响不大,故本算法具有较强的抗干扰能力。甚至在背景较干净的条件下,可以不用作滤波前处理。

改进后的 SSDA 算法,进一步提高匹配精度,并且大大减少了算法的运算时间,完全能满足系统对精度与速度的要求。同时,算法也具有较强的抗干扰能力。

结束语 半导体后工序生产设备是非常复杂的机电一体化系统。除借鉴国外已有的技术外,必须自主研究和解决其中的关键技术问题。本文着重研究芯片识别中所采用的关键技术——模板匹配,讨论分析目前常用的基于灰度的快速模板匹配技术,即 FFT 相关算法、序贯相似检测算法(SSDA)、幅度排序相关算法、多辨率塔型结构算法。本文改进了自适应门限序列的 SSDA 算法,将匹配过程分为两步进行,第一步为粗匹配,第二为精匹配,大大提高了定位的精度与速度。经过改造而成的自动焊机,可以做到一人多机,在生产效率上有较大幅度的提高。

参 考 文 献

1 胡跃明,等. 半导体生广后工序设备中的若干视觉检测与伺服系统设计技术[J]. 广东自动化与信息工程,2003,24(1):7～9
2 王延凤,何惠阳,孙宝玉,等. 全自动金丝球焊机的 CAD/CAE 设计研究[J]. 光学精密工程,2002,10(5):466～470
3 鲍苏苏,等. 多步进电机控制算法研究[J]. 东华大学学报,2003,29(5):83～85
4 刘立峰,汤建华. 全自动金丝球焊机焊接定位系统设计研究[J]. 光学精密工程,2002,10(6):70～74
5 姜凯. 全自动金丝球焊机中芯片图像实时定位方法研究[D]:[中国科学院硕士研究生论文]. 2003
6 王耀南,等. 计算机图像处理与识别技术[M]. 高等教育出版社,2001
7 何斌. Visual C++数字图像处理[M]. 人民邮电出版社,2001
8 Banks S. Signal Processing, Image Processing and Pattern Recognition[M]. Englewood Cliffs, NJ: Prentice Hall, 1996
9 Cha Sung-Hyuk. Efficient Algorithms for Image Template and Dictionary Matching. Journal of Mathematical Imaging and Vision [J]. 2000, 12: 81～90
10 Castleman K R. Digital Image Processing[M]. Prentice Hall, 1996
11 郑南宁. 计算机视觉与模式识别[M]. 国防工业出版社,1998
12 贾云得. 机器视觉[M]. 科学出版社,2002
13 Zhang Tianxu, Sang Nong, Wang Guoyou, et al. An effective mothod for identifying small objects on a complicated background [J]. Artificial Intelligence in Engineering, 1996(10)
14 米长为,刘晓利,徐明友,等. 基于 SSDA 的改进算法[J]. 弹箭与制导学报,2004,24(1):85～87
15 李俊山,等. 图像匹配技术研究[J]. 微电子学与计算机,2002(2)