一种动态服务组合技术及其支撑系统*)

虞建杰　马晓星　吕　建

(南京大学计算机软件新技术国家重点实验室　南京大学计算机软件研究所　南京 210093)

摘　要　Web服务组合现已成为软件研究和实践热点之一。目前虽已出现多种Web服务组合模型和描述语言,但是面对开放分布多变的Internet网络环境,这些模型和语言在应对变化的动态性支持方面仍显不足,尤其是缺乏从协作全局对动态性的支持。在分析既有服务组合方式及其特点的基础上给出了一个解决方案,它通过一种动态协同模型综合了服务编排和服务编制的优点,在保持与主流技术和平台的兼容性的同时增强了对组合服务动态性的支持。还实现了相应的支撑系统,并在其上实现了一个应用示例。

关键词　服务组合,动态性,协作性,WS-CDL,BPEL

Dynamic Service Composition and its Supporting System

YU Jian-jie　MA Xiao-xing　LU Jian

(State Key Laboratory for Novel Software Technology, Institute of Computer Software, Nanjing University, Nanjing 210093, China)

Abstract　Web service composition has raised many interests in software research and practice. Now there are many Web service composition models and description languages, but most of them are not effectively adapt to the open dynamic and distributed network environment since they provide little support to the dynamic evolution, especially from the view of global collaboration. Analyzed the composition approaches applied nowadays and then presented a solution to the above problem. A dynamic coordination model that combines choreography and orchestration was proposed to enhance the support for dynamism, which is done without the sacrifice of the compatibility with mainstream platforms and technologies. Finally a supporting system was developed, with which an illustrative application was also implemented.

Keywords　Service composition, Dynamism, Collaboration, WS-CDL, BPEL

1　引言

Web服务是由URL标识的一个软件系统,其公共接口和绑定由XML定义和描述[1],通过开放协议(如WSDL,SOAP,UDDI)提供面向Internet应用的统一服务描述、发现、绑定和集成机制,成为了开放环境下实现互操作的一种主要方法。但是单个Web服务提供的功能有限,Web服务更大的价值在于服务的组合,将分布独立的Web服务组合成复杂的业务功能,以满足实际应用的需求才是Web服务真正的价值所在。近年来,Web服务组合研究成为了关注热点,服务组合模型和描述语言大量涌现,但是开放、分布、多变的网络环境对服务组合提出了更高的要求。一方面,网络环境的不断改变,服务的出现、更新、消亡、非功能性需求的变化等情况,都可能造成已经构建的服务组合应用无法继续使用,但停止原应用重新开发或部署往往会带来无法接受的延迟、代价和风险,因此实际应用需要组合服务具有动态适应的能力。另一方面,就服务组合的建模和控制技术而言,很多方面借鉴了工作流领域的成果,采用单一集中式的结构,即位于系统主导位置的服务组合引擎负责根据组合模型调度各基本服务依照指定的流程和条件执行,并进行相应的数据交换。但不同于传统工作流系统所应对的企业内部应用,Web服务组合面向开放的网络环境,参与组合的服务都是自治对等的实体,具有各自的功能、特性和要求,单一方式的描述和控制是不合理也不实用的,需要全面协作的组合模型才能更好地刻画实际应用。因此,开放的网络环境要求服务组合具有动态性,特别是从协作全局支持的动态性。本文通过分析比较现有的Web服务组合方式发现了多种服务组合模型和语言在这两方面的欠缺,并在此基础上给出了一个解决方案,通过一种动态协同模型综合服务编排和服务编制的优点,增强对组合服务动态性和全局协作性的支持,然后运用WS-CDL[2]和BPEL4WS[3](简称BPEL)两种服务组合语言予以实现,同时介绍、分析了其原型架构系统和可以进行的动态变化。

2　现有服务组合方式及其存在的问题

研究服务组合的动态性和全局协作性首先要明确在服务组合的设计运行过程中影响这些特性的因素。下面我们将分析服务组合的过程和相应的特性,并与现有的服务组合方式进行对比。

一个组合服务可以认为是多个活动(原子活动或结构化活动)的复合,它们以预先定义的顺序被调用,像一个整体一样执行[4]。在这个定义下,一个组合服务具有典型的业务流程的特点,所以构建一个组合服务一般有以下步骤:(1)一个流程模型指定多个活动之间的控制流和数据流;(2)查找并发现实际的应用服务,绑定到流程活动上;(3)组合服务作为整

*)本文工作受到国家自然科学基金(60403014),教育部新世纪人才项目(NCET-07-0419),江苏省自然科学基金(BK2006712)和江苏省六大人才高峰项目的支持。**虞建杰**　硕士研究生,研究方向为Internet软件技术、软件Agent技术;**马晓星**　博士,研究方向为Internet软件技术、软件体系结构;**吕　建**　教授,博士生导师,研究方向为软件自动化、并行程序形式化方法、面向对象语言。

体发布，使其能被客户发现并使用；(4)组合服务应用期间，一个协同实体(如流程执行引擎)解析流程模型并管理控制流和数据流。

根据以上步骤可知，影响一个组合服务动态性的主要是步骤(1)和(2)。一方面，创建流程模型时的自动化程度对组合服务的动态变化有重要影响。根据自动化程度，一般可以分成3种流程创建方式：手动(manual)、半自动(semi-automatic)、全自动(automatic)。硬编码和使用一些图形化编辑器创建流程属于手动方式，较为繁杂且不灵活，形成的组合服务动态性差。模型驱动的服务组合方法[4]属于半自动方式，它将组合逻辑和组合规范分离开，在更抽象的层次使用模型语言(如UML)将组合服务模型化，然后映射到具体规范(如BPEL)上，进而再关联到实际应用。模型驱动方法建立在UML等基础上，不依赖于特定的平台和规范，适用范围广，但需要多个映射子系统，较为复杂且不易实现。全自动的服务组合方式要求在没有(或极少)人为干预的情况下从应用目标自主产生一个服务组合方法，人工智能和形式化逻辑知识被广泛地应用于此，而OWL-S[5]的出现更为自动的服务组合增加了可能性。这种自动服务组合方式具有很强的适应性，能满足不断变化的业务需求。但是，如何准确高效地自动组合、如何评价组合方案与实际目标的吻合度等一系列问题都亟待深入研究，是非常复杂和困难的工作。

另一方面，查找和发现运用到服务组合中的具体服务可以在设计阶段或运行时刻进行，前者是静态绑定，后者可以在运行时根据一些特定的标准(如功能性、QoS)进行选择。静态(early)绑定在设计阶段即确定具体服务，运行效率高，但运行时刻无法改变，不适应灵活多变的应用。延迟(late)绑定使得具体服务可以动态调用，但需要建立相应的协同工作层，应用一些查找匹配机制等，使用延迟绑定在适应性、容错和可靠性上更具优势。

全局协作性应该主要从执行控制方面进行分析。从执行和控制的角度看，Web服务组合方式有编制(Orchestration)和编排(Choreography)两种[6]。编制通常表示一个可执行业务流程，流程中必须描述所有的交互动作(包括活动顺序、可能的事件等)，编制从一个实体的视角集中描述所有控制。而编排更注重协作性，刻画多方实体之间的公共消息交换，每个实体只明确自己的交互和行为，没有任何一个实体具有全局视图或控制。因此，编制展现了一个集中式的局部视图，它是可执行的业务流程；而编排描述了协作式的全局视图，从整体刻画整个应用的交互，但现阶段编排语言多是规约形式。

根据之前的分析，可将Web服务组合方式分为12个种类(每个种类之间的界限并不完全严格)，如图1所示。

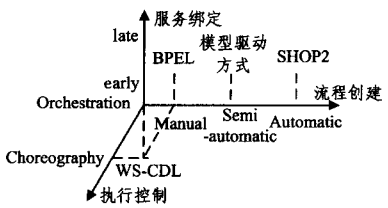


图1 Web服务组合方式分类

许多Web服务组合模型和描述语言都可归于上面的类型。比较具有代表性的如微软和IBM联合推出的BPEL4WS语言，它是一种基于编制的静态服务组合方式；Oracle公司提出的WS-CDL是一种编排规约。同时，在Web服务组合的系统研发方面也已经出现了一些原型系统和商业产品，如IBM发布的BPWS4J类库[7]是基于BPEL语言实现的Web服务组合支撑系统；文献[8]介绍了一个模型驱动服务组合方法的系统框架；SHOP2[9]是一个利用HTN[10]方法实现的自动Web服务组合系统。但是这些系统多是基于编制实现的，缺乏全局协作性。所以，为了实现兼具动态性和协作性的服务组合方式，下面将提出一种把服务编排和服务编制相结合的组合方法。

3 一种兼具动态性和协作性的服务组合方式

从上一节的分析我们可以得出，具有最好动态性和协作性的服务组合方式应该采用全自动的流程创建、具体服务的延迟绑定和编排的执行控制方式。但是全自动的方式多采用人工智能等知识，较为复杂，故暂不予研究。同样，模型驱动等半自动的方式也不予考虑。我们提出一种较为实用的组合方式，选择手动的流程创建，支持延迟绑定，同时采用编排的方式协作控制整个应用，主体框架如图2所示。

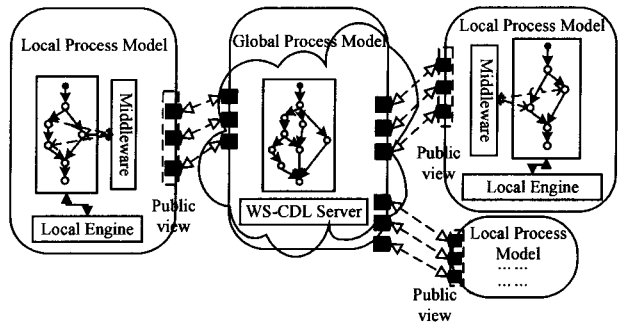


图2 兼具动态性和协作性的服务组合方式

在这种服务组合方式中，我们采用编排语言描述整个应用的全局协作模型，同时采用编制语言描述每一个参与方实体的局部流程模型。在运行时刻，每个实体按自身的局部流程模型执行，但实体之间的交互调用不再是之前已经确定的，而是由协作模型动态解释执行。若协作模型动态改变，则实体之间的交互也可以动态改变，因此具有动态性。另外，协作模型描述的是各个参与实体之间的公共消息交换，需要将各个实体的外部可见流程、条件和约束等组成的外部视图进行全面的刻画，以满足每一方的要求。每个参与实体自身单一对外的描述控制共同协作，组成了逻辑上具体应用的整体流程控制和约束限制，协作性即体现于此。

从实际应用层面理解，协作模型是将服务组合系统中各参与方达成的约定和总体需求变成了一个可以操作并且运行时刻可以动态变化的实体。一般开发服务组合系统时，系统开发者首先需要明确整个系统和各参与方的需求和限制，形成书面或其他形式的约定，并按此进行系统的开发。但这样的约定往往是静态的，确定之后不可更改，同时在系统运行时不再或很少应用这些信息。这样不仅限制了应用动态演化的能力，而且不能很好地对系统进行全面掌控。采用协作模型对约定信息进行全面动态的刻画是解决这个问题的一个可行的思路。另外，这里的协作模型是逻辑上的全局概念，在实现时可以根据性能等要求在物理上实现为集中式或分布式。

4 基于WS-CDL和BPEL的服务组合支撑系统

实现上述兼具动态性和协作性的服务组合方式，首先要

确定采用的编排和编制语言，我们选择 WS-CDL 和 BPEL。WS-CDL 是 W3C 最新定义的从全局视角描述服务组合行为的 XML 语言，目前还是草案，但正努力朝标准迈进，代表了编排语言方面较新的研究成果。BPEL 现今已广泛应用，成为了事实上的标准，而且 BPEL2.0 很可能成为 OASIS 的官方标准。因此我们选择这两种代表性的语言进行系统开发，保持与主流技术和平台的兼容性。关于 WS-CDL 和 BPEL 的详细内容可以参考文献[2,3]。下面将介绍基于这两种语言实现的服务组合支撑系统。

4.1 WS-CDL 与 BPEL 的关联

在动态协作的服务组合方式中，WS-CDL 描述的协作模型需要对 BPEL 描述的局部模型的对外交互调用进行动态解释，因此需要建立 WS-CDL 与 BPEL 之间的关联。分析 WS-CDL 与 BPEL 语言的特点，并借鉴文献[11]的工作，我们发现 WS-CDL 描述可以为每一个参与方映射生成 BPEL 局部抽象描述，此处称之为存根(stub)。具体的映射规则如表 1 所示。

从表 1 所示的映射规则可以看到 WS-CDL 与 BPEL 语言的语法结构和语义有着很好的对应关系(此处不研究 BPEL 语言的图连接结构)。WS-CDL 描述的每一部分可以映射成它包含的一个或多个参与方的 BPEL 存根的组成部分，这表明这种映射是合理的。同时，这种映射构建了两种语言之间的关联，是实现动态协作的服务组合方式所必需的。这种映射方式的关键在于将 WS-CDL 描述的定义声明和流程控制准确定位到相应的一个或多个 BPEL 存根中，参考文献[11]，我们能够合理地予以实现。上述映射方式涵盖了 WS-CDL 和 BPEL 的大部分语法结构，状态和事务暂时没有考虑，但随着研究工作的继续，它们也将逐步加入。

表 1 WS-CDL 到 BPEL 的映射规则

	WS-CDL →		BPEL
类型定义&声明	relationshipType		partnerLinkType
	roleType		role&portType
	variableDefinitions&informationType		variables
	relationship		partnerLink
控制部分	基本活动	Interaction	exchange(action="request" role="fRole") → invoke
			exchange(action="request" role="tRole") → receive
			exchange(action="response" role="tRole") → reply
		assign	assign
		noAction	empty
		silentAction	sequence(empty)*
	结构化活动	sequence	sequence
		choice	switch
		parallel	flow
		workunit	scope
		Workunit	workunit(repeat="true") → while
			workunit(block="true") → receive*
			workunit(guard="true") → switch*

注：带“*”的项表示一种参考映射方式。

4.2 WS-CDL 与 BPEL 相结合的服务组合方式

WS-CDL 与 BPEL 相结合的服务组合方式具有动态性和协作性的本质在于与通常的 BPEL 服务组合方式所不同的服务调用方式。一般 BPEL 中的组合服务与其“伙伴(partner)”的交互行为可以归结为：partner.portType(params)，调用方式如图 3(上)所示，这要求在调用发生前 partner 必须有具体的值，因此在组合服务部署前需要静态绑定具体服务。这就限制了运行时刻的动态适应性，而且完全由组合服务方控制的调用可能不满足交互方或应用整体的要求。而 WS-CDL 刻画了应用的整体交互和约束，根据 4.1 节的表述，WS-CDL 可以为每一个参与方生成 BPEL 存根。因此我们不把 partner 静态绑定到具体服务，而绑定到特定的中间件服务，在运行时刻将调用交由 BPEL 存根解释执行，如图 3(下)所示。这种结合的方式使得调用的具体服务可以在运行时刻动态确定，并且可由 BPEL 存根对调用的约束进行检查，以满足应用的整体要求，使动态性和协作性得以提高。

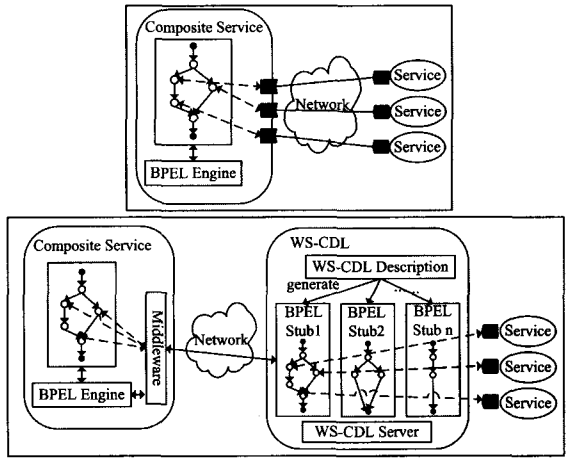


图 3 BPEL 静态服务调用(上)和 WS-CDL 与 BPEL 结合的服务调用(下)

4.3 基于 WS-CDL 和 BPEL 的服务组合支撑系统原型系统总体结构

基于前面的分析，本文实现了一个基于 WS-CDL 和 BPEL 的服务组合支撑系统，其总体结构如图 4 所示。

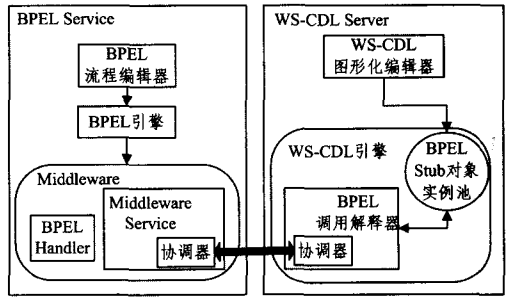


图 4 基于 WS-CDL 和 BPEL 的服务组合支撑系统

图 4 描述了原型系统的主要组成部分：

(1)WS-CDL 图形化编辑器提供给客户简单的图形界面，用于编写应用的 WS-CDL 描述。

(2)WS-CDL 引擎根据 WS-CDL 描述产生 BPEL 存根对象存放于实例池中。在运行时刻收到 BPEL 组合服务对外的调用时，从实例池中找到合适的 BPEL 存根对象，并运用 BPEL 调用解释器根据 BPEL 存根对象解释调用，进而执行

实际的调用操作。

(3)BPEL 流程编辑器和引擎用于 BPEL 流程的描述和执行,本文采用开源性开发工具 ActiveBPEL。

(4)Middleware 是存在于每一个参与方 BPEL 组合服务端的中间件,作为 BPEL 组合服务与 WS-CDL 服务器交互的中间实体。BPEL 组合服务对外的服务调用都绑定到此中间件,从而调用请求都转发到 Middleware,然后由 Middleware 将请求发送到 WS-CDL 服务器进行动态解释。另外,Middleware 需通过 BPEL Handler 给请求附加 WS-CDL 和参与方的名称信息,使得在 WS-CDL 服务器端能准确找到相应的 BPEL 存根对象。

(5)协调器是根据 WS-Coordination 规范设计的协调框架,系统的交互都是在此协调框架内进行的。

4.4 动态演化

使用基于 WS-CDL 和 BPEL 的服务组合支撑系统开发应用,在运行时刻如果网络环境或客户需求发生改变,则只需要对 WS-CDL 的描述进行修改,即可产生对服务调用的重新解释,适应新的变化。

对 WS-CDL 描述的修改需要满足一定的条件,这也表明能够进行一定限度的动态演化。修改 WS-CDL 描述要遵循的条件是改变后的 WS-CDL 描述产生的各 BPEL 存根对象的视图要包含当前正在运行的各个参与方 BPEL 组合服务的外部视图,即不影响当前正在运行的各服务。此条件可以通过形式化方法论证,此处不详述。

5 实例分析

下面我们通过一个实例说明基于 WS-CDL 和 BPEL 组合服务方式的动态性和协作性的应用效果。实例采用一个客户贷款服务的场景,客户向银行贷款,银行根据贷款额度必要时先附加风险评估再确定是否同意贷款。场景示意图如图 5 所示。

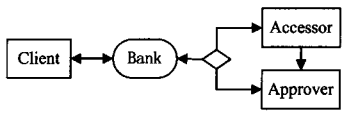


图 5 客户贷款服务场景

首先用 BPEL 描述 Bank,Accessor,Approver 的流程场景。篇幅所限,仅简单描述 Bank 的 BPEL 流程框架大体如下:

```
〈process name="LoanApproval" ……"〉
  〈partnerLinks〉 ……〈/partnerLinks〉
  〈variables〉 …… 〈/variables〉
  〈sequence〉
    〈receive createInstance="yes" ……"/〉
    〈switch name="IsamountLT10000"〉
      〈case. condition="bpws:getVariable('loanRequest','a-
       mount')&amp;gt;=10000" 〉 ……〈/case〉
       〈case condition=……〉
         〈switch name="Ishighrisk"〉
           〈case. condition=bpws:getVariable('riskAssess-
            ment','risk')=='low'"〉 ……〈/case〉
            〈case condition=……〉……〈/case〉
         ……
       〈reply……/〉
  〈/sequence〉
〈/process〉
```

然后用 WC-CDL 编辑器编写 WS-CDL 描述。篇幅所限,仅示部分框架,见图 6。

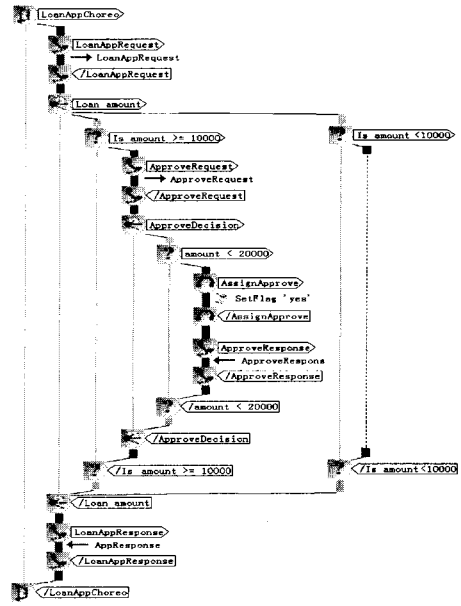


图 6 客户贷款场景 WS-CDL 描述

根据 WS-CDL 描述可以生成 Bank 的 BPEL 存根对象,结构如图 7 所示。

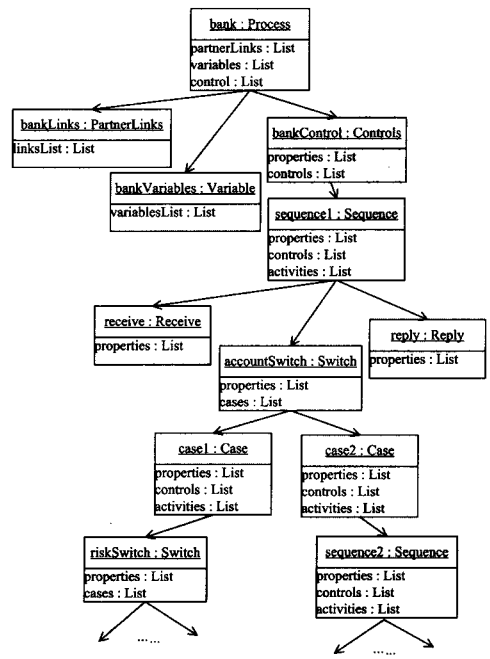


图 7 Bank 的 BPEL stub 对象

在运行时刻,Bank 流程对外的服务调用可以通过遍历 Bank 的 BPEL 存根对象,从 invoke 对象的属性中获得实际的服务地址,并可以通过对象所含有的属性进行检查与限制。这只是对调用动态解释的最简单方式,也可使用其他的解释方式,在以后的研究中将逐步深入。

上述贷款实例可以进行多种动态演化。如图 8(上)所示,在运行时刻修改实际的 Accessor 服务,只要修改 WS-CDL 描述中 Accessor 的绑定即可;若一个服务同时包含 Accessor 与 Approver 服务,则应用场景结构可以简化成图 8

(下)所示结构,但并不影响 Bank 服务的应用。

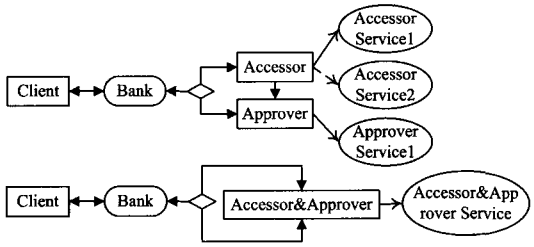


图 8 应用服务绑定的变化(上)和应用整体结构的变化(下)

结束语 本文通过分析现有的 Web 服务组合方式的种类和特点,讨论了多种服务组合方式在动态性和协作性方面的不足,进而提出了一种兼具动态性和协作性的服务组合方式。在此基础上结合 WS-CDL 和 BPEL 对此服务组合方式予以实现,并介绍了其原型支撑系统,最后通过一个客户贷款实例展示了其应用效果。当然,对照完整的 SOA 应用的需求,本文工作尚不全面,还有许多工作,尤其是关于状态、事务等特性的考虑需要进一步深入研究。

参 考 文 献

[1] Web Service Definition Language (WSDL) Specification[EB/OL]. 2002. http://www.w3.org/TR/wsdl

[2] World Wide Web Consortium W3C. Web Services Choreography Description Language Version 1.0 (WS-CDL)

[3] BEA Systems, IBM, Microsoft, SAPAG, and Siebel Systems. Business Process Execution Language for Web Services (BPEL4WS) Version1.1. 2003. http://www.ibm.com/developerworks/library/ws-bpel/

[4] Fluegge M, Garcia dos Santos I J, Tizzo N P. Challenges and Techniques on the Road to Dynamically Compose Web Services //6th International Conference on Web Engineering (ICWE). 2006

[5] OWL-S1.1Release[EB/OL]. http://www.daml.org/services/owl-s/1.1/

[6] PelzC. Web services orchestration and choreography. IEEE Computer, 2003, 36(8): 46-52

[7] BPWS4J. http://www.alphaworks.ibm.com/

[8] Bart O, Yang J, Papazoglou M P. A Framework for Business Rule Driven Web Service Composition // Jeusfeld M A, Pastor Ó, Eds. ER 2003 Workshops. LNCS 2814. Berlin Heidelberg: Springer-Verlag, 2003: 52-64

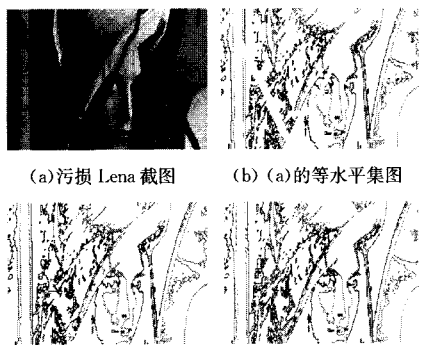
[9] Wu Dan, Sirin E, Hedler J. Automatic Web Services Composition Using SHOP2. [EB/OL]. http://www.mindswap.org/papers/ICAPS03-SHOP2.pdf

[10] Sirin E, Parsia B, Wu Dan. HTN Planning for Web Service Composition Using [EB/OL]. http://www.mindswap.org/papers/SHOP-JWS.pdf

[11] Mendling J, Hafner M. From Inter-Organizational Workflows to Process Execution: Generating BPEL from WS-CDL // Meersman R, Tari Z, Herrero P, eds. Proceedings of OTM 2005 Workshops, LNCS 3762. 2005: 506-515

(上接第 213 页)

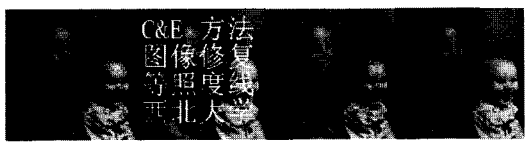
500 次的结果,(d)图是迭代 5000 次的结果。图示参数:$K=0.9$,步长 $\Delta t=0.1$,N 是迭代次数。



(a)污损 Lena 截图 (b)(a)的等水平集图

(c)C&E 方法 (d)Lena 原图的等水平集图

图 3 等水平集修复



(a)原始照片 (b)"涂鸦"照片 (c)$N=500$ (d)$N=5000$

图 4 C&E 方法修复被"涂鸦"的照片

(a)污损 Lena 图像 (b)CDD 方法 (c)C&E 方法

参数:$K=0.9$,$\Delta t=0.1$,$N=7000$

图 5 C&E 和 CDD 方法比较

接着,我们在相同的迭代次数下,对比了 C&E 和 CDD 方法,图 5(a)是带签字的 Lena 图像;(b)是 CDD 方程修复的结果;(c)是 C&E 模型修复的结果。

值得指出的是,CDD 方法经过足够的迭代次数以后能得到在视觉上较图 5(b)稍好一些的修复效果,但最终效果不及 C&E 方法。

结束语 本文从图像形态学的观点出发,分析了等照度线曲率在控制图像修复中的重要性,提出了将曲率驱动与边缘停止相结合的非线性扩散模型(C&E 模型)应用于图像修复,并讨论了有关这一模型的构造和数值实现。由于 C&E 模型同时强调了保护小曲率和大梯度的重要性,实验结果表明其应用于非纹理数字图像修复性能良好。

参 考 文 献

[1] Sapiro G, Caselles V. Histogram Modification via Differential Equations [J]. Journal of Differential Equations, 1997, 135(2): 238-268

[2] Caselles V, et al. Shape Preserving Local Contrast Enhancement [C]// Proceedings of the 1997 International Conference on Image Processing (ICIP'97). Washington: IEEE Computer Society, 1997: 314-317

[3] Bertalmio M, Sapiro G, Caselles V, et al. Image inpainting [C]// Proc. of the ACM SIGGRAPH 2000. New Orleans, USA, 2000 (7): 417-424

[4] Fadili M J, Starck J L. Em algorithm for sparse representation-based image inpainting[J]// Proc. of the 2005 IEEE Int'l Conf. on Image Processing. 2005, 2: 61-64

[5] Patwardhan K A, Sapiro G, Bertalmio M. Video Inpainting of Occluding and Occluded Objects [J] // Proc. IEEE ICIP'05. 2005, 2: 69-72

[6] Chan T F, Shen J. Non-texture Inpainting by Curvature-driven Diffusions (CDD)[J]. Journal of Visual Communication and Image Representation, 2001, 12: 436-449

[7] 祝轩,周明全,耿国华,等. 一种新的三阶非线性扩散图像平滑方法及其算法实现[J]. 计算机科学, 2007, 34(12): 227-230