Markov 控制转换多模块软件可靠性测评方法*)

覃志东　王洪亚　李继云　乐嘉锦

(东华大学计算机科学与技术学院嵌入式系统研究所　上海 201620)

摘　要　针对软件多模块化发展的趋势以及当前软件可靠性测评把软件当成一个整块进行处理的缺点，本文提供了一种 Markov 控制转换多模块软件的可靠性测评方法。方法的主要思想包括建立基于软件体系结构的可靠性模型，并依据该模型把系统级的可靠性指标优化分配到模块级，实现在模块级对整个软件系统可靠性的监控，并在系统级的测评中进行确认。文中针对系统级测评缺少失效数据的情况，重点阐述了如何基于最大熵原理求解系统可靠性参数分布函数，以实现对系统可靠性指标的验证。

关键词　软件可靠性，可靠性测试，经验贝叶斯方法，最大熵原理

Reliability Testing Method for Multi-modules Software with Markov Transfer of Control

QIN Zhi-dong　WANG Hong-ya　LI Ji-yun　LE Jia-jin

(Institute of Embedded Systems, School of Computer Science & Technology, Donghua University, Shanghai 201620, China)

Abstract　Due to the software multi-modules developing trend and the disadvantage of current software testing methods to treat the software system as a monolithic block, a reliability testing method is presented for the modular software with Markov transfer of control. The main idea of the testing method is that we can monitor and control the software reliability in the module level via allocating the reliability target of software system to the software modules based on the reliability model optimally, and validate the system reliability target by testing the software in system level again. According to the maximum entropy principle, the problem of how to estimate and demonstrate the system reliability with few failure data is solved emphatically.

Keywords　Software reliability, Reliability testing, Empirical bayesian statistics, Maximum entropy principle

1　引言

经过 35 年的发展，软件可靠性理论已经由单一的可靠性建模发展成为一门系统性的学科：软件可靠性工程(Software Reliability Engineering)[1]。但随着软件规模呈指数的快速增长，在安全关键领域的应用，其面临的问题与挑战越来越多[2,3]。

现代软件系统为了便于开发、集成和扩展，越来越趋于向多任务、多模块化方向发展[4]。各个任务模块是功能相对独立的软件。不同的软件体系结构，软件模块的执行流不一样，系统对每个软件模块可靠性的敏感性也不一样。特别是对于高可信软件，不同功能模块失效造成的后果不一样，因而对其要求的安全关键度也不一样[5]；从而，客观上，对实现不同软件功能的软件模块所要求的可靠性不一样。

然而，传统的软件可靠性建模和测评方法，如序惯测试、固定期测试、单风险序惯测试、经验贝叶斯测试[6,7]等方法，都是把软件作为一个整体来处理，不考虑软件内部结构及其相互作用关系。这样，在测试的过程中，很难于集中资源测试对可靠性要求较高的模块；同时，即便系统的可靠性达到要求，也很难保证单个模块的可靠性要求。

早在 20 世纪 70 年代末，Littlewood 就提出了按结构化、模块化的方式进行软件可靠性建模的思想[8]，Smidts 等也一贯倡导通过测试系统中各模块的可靠性达到对整个系统可靠性的评价，并指出这种方法更容易实施，而且可采用的软件测试策略将更加灵活[9]。

目前，基于软件体系结构进行软件可靠性建模的研究开展得如火如荼[10-12]，如何针对软件不同的体系结构建立相应的可靠性测评方法，是目前研究的焦点之一。而软件的体系结构众多，各有其特殊性，本文则针对其中的一类：Markov 控制转换多模块软件，基于最大熵原理，建立了一整套融合可靠性指标分配、模块测试、系统测试为一体的可靠性测评方法，并给出了实例。

2　软件系统模型

基于 Markov 控制转换的多模块软件系统模型如图 1 所示。Markov 控制转换描述了软件系统内部各个模块或者子系统之间对 CPU 资源的占有关系[13]。对应于每一类输入，软件内部都有一个模块执行路径或者是序列，使得软件系统的控制按照不同的方式转换。

由于控制转移的不均衡性，必然导致一些模块对整个软件系统可靠性影响较大。那么，针对此类软件的可靠性测试，必须识别出不同模块对系统可靠性的影响大小，对系统级的可靠性指标在模块级进行分配，在模块级进行设计、测试，然后系统集成，结合模块级的测试信息，再在系统级进行可靠性

*)本课题受上海市科技攻关项目(06DZ150003)，上海申通地铁集团课题(KY-07-025)，东华大学青年教师基金(112-10-0044056)资助。覃志东　博士，讲师，研究方向为实时计算与可信性计算；王洪亚　博士，副教授，研究方向为实时计算；李继云　博士，副教授，主要研究方向为实时计算、图像处理；乐嘉锦　教授，博士生导师，主要研究领域为数据库、可信性计算。

增长和验证测试。

多模块软件系统，特别是应用于关键领域的高可信软件，其诸多功能模块及其算法都是复用以前版本或者是相似系统的，这些软件模块在版本的更迭中已经相当成熟和稳定。即便是重新开发的软件模块，也存在某种程度的相似性和继承性。加之经历了软件模块级的覆盖测试、可靠性增长测试和集成测试，集成后的软件系统已经具备非常高的可靠性。在系统级的测试过程中，将面临两个重要的问题：

(1)采用何种方式对软件当前可靠性进行估计？由于软件可靠性增长模型受建模方法本身的局限性，它只适用于对失效率大于 10^{-3} 个/小时的范围，但评价结果的精度和稳定性都难以保证。

(2)如何制定合理的系统级可靠性验证测试方案，在保证验证测试结果可信性的条件下，尽量减少测试代价、缩短测试周期？由于此时软件系统本身已经具备很高的可靠性，软件可靠性增长测试阶段所能提供的失效数据很少，甚至无失效数据，那么就无法依据大量的失效数据利用经验贝叶斯方法去获得软件可靠性参数的先验分布。

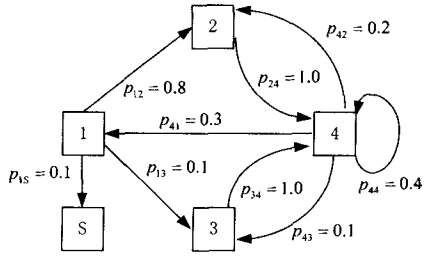


图 1　基于 Markov 控制转换的模块软件系统

这里我们利用开发过程中的模块级的测试信息、继承性以及系统体系结构信息，基于最大熵原理获得较优的软件可靠性参数的先验分布，实现了较优的系统级测评方案。

3　基于最大熵原理的系统级可靠性测评

3.1　最大熵原理

信息论的产生，形成了描述事物不确定性的概念—熵。熵的定义如下：

定义　设 Θ 是离散的，π 为 Θ 上的概率分布，以 $H(\pi)$ 表示 π 的熵，定义为 $H(\pi)=-\sum_{\Theta}\pi(\theta_i)\log\pi(\theta_i)$。(若 $\pi(\theta_i)=0$，定义 $\pi(\theta_i)\log\pi(\theta_i)$ 为零。)

由熵的定义可知，两个随机变量具有相同的分布时，它们的熵就相同，可见，熵只与随机变量的分布有关。因此，也称熵为某一分布的熵。某一分布的熵越大，表示该分布的不确定性越大。在确定软件系统可靠性参数的先验分布时，我们有一部分信息可以利用，如软件模块可靠性，而除此之外的部分就必须要求尽可能地采用无信息先验，使得先验分布的熵最大，以避免对系统可靠性产生冒进的推断结果。

3.2　先验分布参数的求解

假设在软件模块的测试过程中，我们获得了对软件模块 A_i 的失效概率 p_i 的估计值为 $\hat{p}_i$，那么，根据软件体系结构可靠性模型，可以得到此时软件系统的失效概率 p_s 的估计值为：

$$\hat{p}_s=F(\hat{p}_1,\hat{p}_2,\cdots,\hat{p}_n) \tag{1}$$

假设软件系统失效概率 p_s 的先验分布为

$$f(p_s)=Beta(a_0,b_0)=\frac{p_s^{a_0-1}(1-p_s)^{b_0-1}}{B(a_0,b_0)} \tag{2}$$

那么 p_s 的数学期望为：

$$\bar{p}_s=E(p_s)=\int_0^1 p_s Beta(a_0,b_0)\,\mathrm{d}p_s=\frac{a_0}{a_0+b_0} \tag{3}$$

而 p_s 的数学期望就是软件系统失效概率 p_s 的估计值，所以有：

$$\frac{a_0}{a_0+b_0}=F(\hat{p}_1,\hat{p}_2,\cdots,\hat{p}_n) \tag{4}$$

那么，在知道软件失效概率 p_s 的数学期望的情况下，求解其满足最大熵原则的先验分布，转化为求解下面的优化问题：

$$\begin{aligned}\mathrm{Max}\{H[f(p_s)]\}&=\mathrm{Max}\{-\int_0^1 Beta(a_0,b_0)\\&\qquad\ln(Beta(a_0,b_0))\,\mathrm{d}p_s\}=\\&\mathrm{Max}\{\ln B(a_0,b_0)-\frac{1}{B(a_0,b_0)}\int_0^1 p_s^{a_0-1}\\&(1-p_s)^{b_0-1}\cdot\ln[p_s^{a_0-1}(1-p_s)^{b_0-1}]\mathrm{d}p_s\}\end{aligned}$$

$$a_0>0\qquad b_0>0$$

$$\frac{a_0}{a_0+b_0}=F(\hat{p}_1,\hat{p}_2,\cdots,\hat{p}_n) \tag{5}$$

当软件系统失效概率 p_s 的先验分布确定后，就可以利用其来进行系统级的可靠性测评了。此时，需要根据对软件失效概率 p_s 的估计值 $\hat{p}_s$ 与系统可靠性设计指标 p_{s0} 的比较结果，判断是否进行系统级的可靠性增长测试。如果 $\hat{p}_s\leqslant p_{s0}$，则直接进行可靠性验证测试，利用 p_s 的先验分布计算出需要的验证测试用例数 N_0；反之，如果 $\hat{p}_s>p_{s0}$，就必须进行可靠性增长测试，并综合现场测试信息，得到 p_s 的后验分布为：

$$\begin{aligned}f(p_s\mid x,n,a_0,b_0)&=Beta(a_0+x,b_0+n-x)\\&=\frac{p_s^{a_0+x-1}(1-p_s)^{b_0+n-x-1}}{B(a_0+x,b_0+n-x)}\end{aligned} \tag{6}$$

其中，n 表示总测试用例数，x 表示发现的总失效数。

软件系统失效概率 p_s 的后验期望为：

$$\begin{aligned}\hat{p}'_s&=E(p_s\mid x,n,a_0,b_0)\\&=\int_0^1 p_s Beta(a_0+x,b_0+n-x)\,\mathrm{d}p_s\\&=\frac{a_0+x}{a_0+b_0+n}\end{aligned} \tag{7}$$

p_s 的后验期望就是软件失效概率的最小二次损失估计，用其作为软件系统当前失效概率的估计值。如果 $\hat{p}'_s\leqslant p_{s0}$，说明软件系统当前可靠性已经增长到预定的软件系统可靠性设计指标，软件可靠性增长测试可以结束。

在软件可靠性增长测试阶段获得的软件失效概率 p_s 的后验分布，已经综合了软件模块的可靠性信息和软件系统的测试信息，在接下来的软件可靠性验证测试阶段，就可以直接拿来作为失效概率 p_s 的先验分布，根据具体的可靠性验证指标去确定需要的验证测试用例数 N_0。

4　方法示例

4.1　基于软件体系结构的可靠性建模

如图 1 所示，软件模块之间的控制转换被描述成 Markov 链，那么，模块 i 向模块 j 的转换概率 p_{ij} 是独立于控制的，它是何种方式进入模块 i 的。假定系统有 n 个软件模块，其中模块 1 代表系统的初始状态，并且假定当软件系统成功完成执行时，软件系统的控制由控制所在模块 i 以概率 p_{iS} 转向一个终止模块 S(如操作系统)，则转换概率满足如下关系式：

$$p_{iS}+\sum_{j=1}^{n}p_{ij}=1 \tag{8}$$

上述转换概率是基于完全可靠的软件系统的。但是，我们所研究的软件系统不是完全可靠的。每个软件模块都存在缺陷，当缺陷触发时，软件就要失效。以 r_i 表示软件模块的运行可靠性，即控制转换到模块 i 后，系统失效的概率为 $1-r_i$。为了对这个系统建模，需要假定一个失效状态 F。由于每个软件模块都不是绝对可靠，因此，每个软件模块都可以直接进入状态 F。显然，状态 F，S 与其它的 n 个状态是不一样的，是 Markov 链的吸收状态。于是，对该不可靠系统建模的 Markov 链具有 $n+2$ 个状态，并且其转换矩阵 Q 的元素 q_{ij} 满足如下关系：

$$\begin{cases} q_{ij}=r_i p_{ij} & i=1,2,\cdots,n;\quad j=1,2,\cdots,n \\ q_{iF}=1-r_i & i=1,2,\cdots,n \\ q_{FF}=q_{SS}=1 & \\ q_{ij}=0 & \text{其它} \end{cases} \tag{9}$$

对于如图 1 所示软件模型，其转换矩阵 Q 如下：

$$Q=\begin{bmatrix} 0 & p_{12}r_1 & p_{13}r_1 & 0 & p_{1S}r_1 & 1-r_1 \\ 0 & 0 & 0 & r_2 & 0 & 1-r_2 \\ 0 & 0 & 0 & r_3 & 0 & 1-r_3 \\ p_{41}r_4 & p_{42}r_4 & p_{43}r_4 & p_{44}r_4 & 0 & 1-r_4 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ S \\ F \end{matrix} \tag{10}$$

$$\begin{matrix} 1 & 2 & 3 & 4 & S & F \end{matrix}$$

软件系统成功执行，也就是软件系统从初始状态成功转移到吸收状态 S。所以，软件系统的运行可靠性也就是系统从初始状态转移到吸收状态 S 的概率，可以由下式求解：

$$R_s=\sum_{i=1}^{n}(I_n-\hat{Q})_{1i}^{-1}r_i p_{iS} \tag{11}$$

其中，I_n 为 n 阶单位矩阵，$\hat{Q}$ 为 n 阶非吸收状态转移概率矩阵。

4.2 软件可靠性分配

针对图 1 所示的软件模型，我们在假定其它软件模块的运行可靠性为 0.999 的情况下，对软件系统可靠性 R_s 与各个软件模块可靠性 r_i 之间的关系进行了数值分析，结果如图 2 所示。可知，软件模块 4 对软件系统可靠性的影响最大，而模块 3 则最小，在可靠性分配时，必须优先考虑软件模块 4 的可靠性。

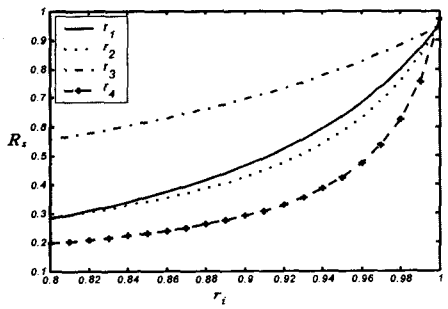


图 2 软件系统对各个软件模块可靠性的敏感性

假如软件的可靠性验证指标是 $(p_0,c)=(0.005,0.9)$，开发方依其制定的软件系统可靠性设计指标为：$p_{s0}=0.004$。那么根据式(11)，并综合权衡软件系统可靠性敏感性分析结果、软件模块功能重要度、复杂性等，把软件系统的可靠性设计指标分配到各个软件模块。假如这里分配的结果是：$p'_1=p'_2=0.00008$，$p'_3=0.0003$，$p'_4=0.00003$。这些值将要作为软件开发过程中质量跟踪的依据，并且是软件模块可靠性增长测试停止的依据。

4.3 系统级的可靠性测评方法

由于软件模块级的运行剖面的开发以及软件模块接口可靠性估计存在一定的误差，并且由于问题的复杂性，我们不能精确地通过误差分析而得到系统真实的可靠性水平，所以还需要进行系统级的可靠性测试，利用系统级的测试数据进一步对软件系统可靠性进行估计、验证。但是，经过精心的可靠性设计和管理，加之严格的模块级的可靠性增长测试、集成测试，此时的软件系统可靠性已经达到非常高的水平，所以，系统级的可靠性测试失效数据非常少。在利用 Bayes 方法进行系统级可靠性推断时，需要利用软件模块测试和集成测试信息以确定软件系统失效概率的先验分布。

假设经过软件模块测试和集成测试后，各个软件模块的失效概率减小到 $\hat{p}_1=0.000081$，$\hat{p}_2=0.000082$，$\hat{p}_3=0.00027$，$\hat{p}_4=0.000038$。由式(11)可以估计出软件系统运行失效概率为 $\hat{p}_s=F(\hat{p}_1,\hat{p}_2,\cdots,\hat{p}_n)=0.0042$。根据式(5)，可以计算出满足最大熵原则的软件失效概率 p_s 的先验分布超参数为：$a_0=1$，$b_0=237.0952$。

由于 $0.0042>0.004$，系统的可靠性还未达到预定的可靠性设计指标，因此还需要进行系统级的可靠性增长测试。在增长测试阶段，当测试到第 250 个测试用例时，发现了一个失效，排错后，须判断是否应该继续测试下去。而由式(7)得：

$$\hat{p}'_s=\frac{a_0+x}{a_0+b_0+n}=\frac{1+1}{1+237.0952+250}=0.0041> 0.004 \tag{12}$$

显然，软件的当前可靠性水平仍然没有达到预定的可靠性设计指标，需继续测试下去。这个过程一直持续到执行第 1529 个测试用例时发现第 6 个失效，此时有：

$$\hat{p}'_s=\frac{a_0+x}{a_0+b_0+n}=\frac{1+6}{1+237.0952+1529}\leqslant 0.004 \tag{13}$$

则软件可靠性增长测试可以停止。而软件失效概率 p_s 的后验分布超参数为：$a_0=7$，$b_0=1760.1$。

在可靠性验证测试阶段，软件失效概率 p_s 的先验分布为可靠性增长测试阶段获得的后验分布：

$$f(p_s)=Beta(7,1760.1)=\frac{p_s^6(1-p_s)^{1759.1}}{B(7,1760.1)} \tag{14}$$

第一次无失效验证可靠性指标 $(p_0,c)=(0.005,0.9)$，所需要的测试用例数由下式计算：

$$\int_0^{p_0}\frac{p^{a_0+j-1}(1-p)^{b_0+N_{j+1}-j-1}dp}{B(a_0+j,b_0+N_{j+1}-j)}\geqslant c \tag{15}$$

计算结果为：$N_0=338$。那么当软件无失效地执行完这些测试用例，则系统可靠性达标，可以接受该软件。

结束语 本文建立的 Markov 控制转换多模块软件可靠性测评方法具有以下优点。

(1)根据体系结构软件可靠性模型，可以识别出对软件系统可靠性影响比较大的软件模块，对该类模块，分配较高的可靠性指标，在模块级测试时，可以集中资源对这些模块进行测试，可以保证整个系统的可靠性快速增长。

(2)把可靠性指标分配到模块并进行模块级测试，可以避免系统达到很高的可靠性而模块的可靠性却很低的情况出现。特别是在安全关键领域，有利于减小安全关键事故的风险。

(3)基于最大熵原理，综合模块级的测试信息，获得系统

级的软件可靠性参数的经验贝叶斯分布，一方面，有利于减少系统级的测试用例量，实现对一些高可靠性指标的验证；另一方面，在对先验分布的估计时，减小了冒进操作，有利于确保测评结果的可信性。

本文的工作，可以为多模块软件的测评实践提供理论支持。

参 考 文 献

[1] Lyu M R. Handbook of software reliability engineering. McGraw Hill and IEEE Society Press, 1996

[2] Lyu M R. Software reliability engineering: a roadma // Proceeding of 29th Int. conference on software engineering. Minneapolis, 2007: 153-170

[3] 陈火旺，王戟，董威. 高可信软件工程技术. 电子学报，2003，31(12A)：1933-1938

[4] 霍夫迈斯特，王千祥. 实用软件体系结构. 电子工业出版社，2004

[5] Leveson N G. Safetware, system safety and computers. Addison Wesley, 1995

[6] 覃志东，雷航，桑楠，等. 安全关键软件可靠性验证测试方法研究. 航空学报，2005，26(3)：334-339

[7] 覃志东，雷航，桑楠，等. 连续执行软件可靠性验证测试方法. 计算机科学，2005，32(6)：202-205

[8] Littlewood B. A reliability model for systems with Markov structure. Appl. Statist., 1975, 24 (2): 172-177

[9] Smidts C, Sova D. An architectural model for software reliability quantification: sources of data. Reliability Engineering and System Safety, 1999, 64(2): 279-290

[10] Gokhale S S, Trivedi K S. Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework. IEEE Transactions on reliability, 2006, 55(4): 578-590

[11] Cheung L, Golubchik L, Medvidovic N, et al. Identifying and addressing uncertainty in architecture-level software reliability modeling // IEEE International parallel and distributed processing symposium. Miami, 2007: 1-6

[12] Jung Hua lo. Software reliability estimation for modular software systems and its applications // The 3rd international conference on information technology: research and education. Hsinchu, 2005: 312-316

[13] Siegrist K. Reliability of systems with Markov transfer of control. IEEE Transactions on software engineering, 1988, 14(7): 1049-1053

(上接第 240 页)

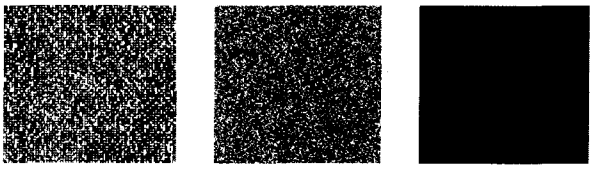
方程及参数我们选用 Logistic 映射来产生两个混沌序列，其仿真参数为 $(x_1, x_2, \mu_1, \mu_2) = (0.3, 0.6, 3.87, 3.95)$，各子块 DFRFT 的 x, y 方向阶数 px, py 由这两混沌序列确定。图 3(a)为分块 DFRFT 后的幅值图像。

(2) 对分块 DFRFT 后的图像进行混沌置乱仿真

再用 Logistic 映射产生一个混沌序列，其仿真参数为：$(x_3, \mu_3) = (0.32, 3.91)$，用来置乱分块 DFRFT 后的图像。图 3(b)为再进行混沌置乱后得到的双重加密图像。

(3) RGB 分量映射存储加密图像数据仿真

双重加密图像的复数数据经 RGB 分量映射存储为一幅混乱的图。此图即可用来传输至接收方。图 3(c)为存储成的加密图像。

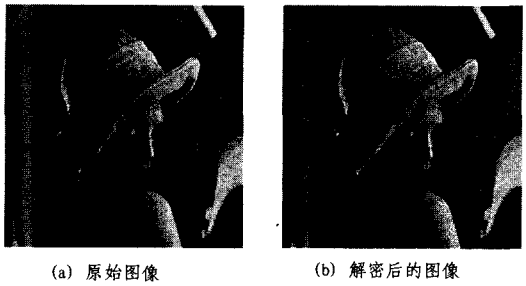


(a) DFRFT 后的幅值图　(b) 混沌置乱后　(c) RGB 分量映射后

(a)DFRFT 后的幅值图　(b)混沌置乱后　(c)RGB 分量映射后

图 3　仿真结果图

(4) 接收方解密出原图仿真



(a) 原始图像　(b) 解密后的图像

图 4　原始图像与解密后的图像

接收方对收到的混乱彩色图进行 RGB 分量逆映射即可得到双重加密图像的数据，然后对其用三个混沌序列的密钥解密得出原灰度图像。图 4(a)，4(b)分别为原始图像与解密后的图像。

结束语　本文利用混沌序列的特性，对灰度图像先进行分块 DFRFT，然后再混沌置乱，实现了图像的双重加密。进一步，为了实现图像的加密传输，故把加密图像数据映射到彩色位图的 R, G, B 分量中，形成一幅混乱的彩色图像，这样可用它来进行传输。计算机仿真结果表明，本算法实现容易，密钥空间大，安全性强，可应用于实际传输中，因而具有了较强的实用性。

参 考 文 献

[1] Nikolaids, Athanasios. Asymptotically optimal detection for additive water-marking in the DCT and DWT domains. IEEE Trans. on Image Processing, 2003, 12(5): 563-571

[2] Mendlovic D, Ozakatas H M. Fractional Fourier transforms and their optical implementation: I. J. Opt. Soc. Am., 1993, A10: 1875-1881

[3] Mendlovic D, Zalevsky Z, Dorsch R G, et al. New signal representation based on the frac-tional Fourier transform. J. Opt. Soc. Am., 1995, A12: 2424-2431

[4] 何俊发，李俊，王红霞，等. 不对称离散分数傅里叶变换实现数字图像的加密变换. 光学技术，2005，31(3)：410-412

[5] Ozaktas H M, Arikan O, Kutay M A, et al. Digital computation of the fractional Fourier transform. IEEE Trans. on Signal Processing, 1996, 44(9): 2141-2150

[6] Candan C, Kutay C A, Ozaktas H M. The discrete fractional Fourier transform. IEEE Trans. on Signal Processing, 2000, 48(5): 1329-1337

[7] 王银花，柴晓冬，周成鹏，等. 基于混沌序列和分数傅里叶变换的图像加密技术. 计算机技术与发展，2006，16(9)：213-215

[8] 于凤芹，姚旭辉，曹家麟. 分数阶傅里叶变换的若干问题. 江南大学学报，2002，1(4)：349-353

[9] 张峰. 基于分数阶 Fourier 变换的 chirp 类数字水印算法研究. 郑州大学硕士学位论文. 2006

[10] 陶然，齐林，王越. 分数阶 Fourier 变换的原理与应用. 北京：清华大学出版社，2004

[11] 张可，王典洪. 基于 Logistic 混沌序列的图像空域复合加密研究. 计算机与现代化，2005(1)：66-69