

图像区域增长程序的蜕变测试框架

蒋超¹, 黄松^{1,2}, 惠战伟^{1,2}

(1. 陆军工程大学 指挥控制工程学院, 江苏 南京 210007;

2. 全军军事训练软件测评中心, 江苏 南京 210007)

摘要: 图像区域增长是图像处理技术研究的热点问题, 是图像分割的重要手段。图像处理软件的测试面临测试判定难题, 当前其结果判定主要由领域专家依赖其经验完成, 其准确性无法保证并且效率低下。蜕变测试技术有效缓解了测试判定问题, 并且已成功应用于诸多领域, 但是当前蜕变测试过程存在一定的随意性和盲目性。为了提高测试效率、规范测试流程, 提出了适用于图像区域增长程序的蜕变测试框架。该框架包括原始测试用例的自动生成、区域增长程序自动执行、测试结果自动判定和测试有效性度量等模块, 并且给出了每个模块的具体实现方法, 具有很好的实践指导意义。通过实际软件测试工程项目, 验证了该测试框架的合理性和有效性, 同时规范了测试流程, 提高了测试效率, 保证了测试质量。

关键词: 图像区域增长; 测试框架; 蜕变关系; 蜕变测试

中图分类号: TP31

文献标识码: A

文章编号: 1673-629X(2019)04-0063-05

doi: 10.3969/j.issn.1673-629X.2019.04.013

Metamorphic Testing Framework for Image Region Growth Applications

JIANG Chao¹, HUANG Song^{1,2}, HUI Zhan-wei^{1,2}

(1. Institute of Command Control Engineering, Army Engineering University of PLA, Nanjing 210007, China;

2. PLA Military Software Testing and Evaluation Center, Nanjing 210007, China)

Abstract: Image region growth is a hot issue in image processing technology and also an important means of image segmentation. The testing of the image processing software confronts with the test oracle problem. At present, the result validation mainly depends on the experience of the domain experts, and its accuracy cannot be guaranteed and the efficiency is comparatively low. The metamorphic testing method effectively alleviates the oracle problem, and has been successfully applied to many fields, but the current testing process has certain randomness and blindness. In order to improve test efficiency and standardize the testing process, we propose a metamorphic testing framework that is suitable for image region growth program. This framework includes automatic generation of original test cases, automatic execution of image region growth program, automatic validation of test results, and testing effectiveness metrics. The specific implementation method of each module is given, which has great practical guidance significance. Through the actual software test project, the rationality and effectiveness of the test framework are verified. At the same time, the test flow is standardized, the efficiency of the test is improved and the quality of the test is guaranteed.

Key words: image region growth; test framework; metamorphic relation; metamorphic testing

0 引言

图像区域生长是图像分割的常用方法之一^[1], 是指将图像中成组的像素或区域发展成更大区域的过程。从种子点的集合开始, 区域增长是通过将与每个种子点具有相似属性如颜色、灰度、纹理等的相邻像素合并到此区域的过程。

软件测试是保证软件可靠性和软件质量的重要手段, 是评估软件能否满足设计需求的过程。当前, 图像处理软件的测试更多的是由领域专家手动完成, 依赖其经验来判断输出结果是否正确, 这种测试方法效率低下并且无法保证测试充分性和正确性。

传统软件测试面临的一大难题是预期输出结果很

收稿日期: 2018-05-14

修回日期: 2018-09-19

网络出版时间: 2018-12-20

基金项目: 国家自然科学基金(61702544); 中国博士后科学基金项目(2016M603031); 江苏省自然科学基金(BK20160769, BK20141072)

作者简介: 蒋超(1988-), 男, 硕士研究生, 研究方向为蜕变测试; 黄松, 教授, 博导, 研究方向为软件测试、关键软件质量评估; 惠战伟, 博士后, 通信作者, 研究方向为软件测试、故障定位。

网络出版地址: <http://cnki.net/kcms/detail/61.1450.TP.20181220.1049.050.html>

难甚至无法获取,即测试判定问题(test oracle problem)^[2]。为了解决此问题,Meyer 等提出利用统计的方法来验证图像处理程序的正确性^[3],Jameel 等提出利用支持向量机的方法来构建程序测试判定^[4],Din 等提出利用机器学习的方法来构建程序测试判定并成功将此方法应用于衍射图像处理软件测试中^[5]。

为了缓解测试判定问题,香港中文大学的 Chen T Y 等提出了蜕变测试技术(metamorphic testing, MT)^[6]。该技术针对程序预期输出难以获取的问题,将检查程序的实际输出是否符合预期输出转变为检查程序的多个执行结果之间是否满足某特性关系,这种关系被称为“蜕变关系”。蜕变测试以原始输入为基础,以蜕变关系为判断准则,当测试用例之间满足一定的输入关系时,检查输出之间是否符合相应的蜕变关系,若不满足,则可以认为软件存在缺陷,这就为缓解

测试判定问题提供了一种有效途径。

当前蜕变测试的研究热点是蜕变关系的构造及其在不同领域的应用^[7],对测试过程的标准规范研究不是很多,这就造成了测试过程的随意性和盲目性。为了减少人为因素对测试过程的干扰,规范蜕变测试流程,文中提出了一种适用于图像区域增长程序的蜕变测试框架,并对其中的相关模块的实现方法进行介绍。

1 图像区域增长程序的蜕变测试框架

针对图像区域增长程序的特点,提出了由蜕变关系构造、原始测试用例的自动生成、衍生测试用例自动生成、区域增长程序自动执行、测试结果自动判定、测试有效性度量和蜕变关系优化等模块组成的蜕变测试框架,如图 1 所示。

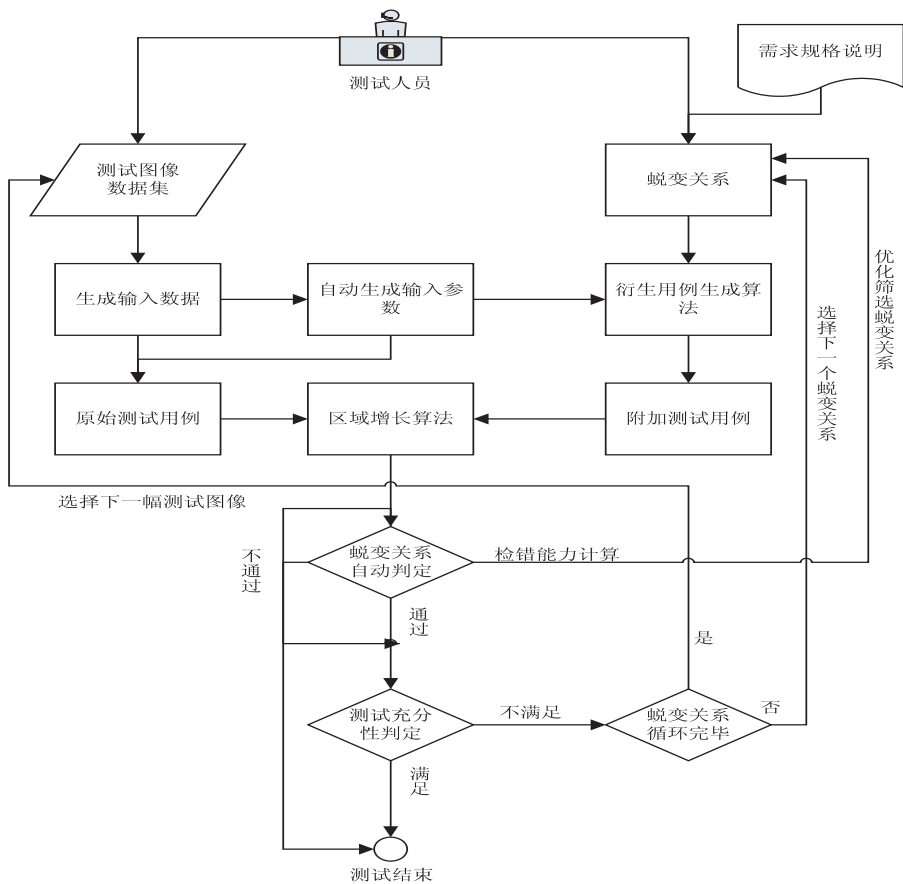


图 1 图像区域增长程序蜕变测试框架

1.1 蜕变关系构造

蜕变测试的结果可以分为以下四类,如表 1 所示。通常第一种与第二种情况是经常遇到的,第一种情况对于程序的正确性有一定参考价值,第二类能够发现程序中存在的错误。因此,有效的蜕变关系应当使得第一种和第二种情况尽可能多的出现。在测试过程中,蜕变关系的构造是实施蜕变测试的关键环节,其质量的优劣直接影响到蜕变测试的性能和效率,也是影响自动化蜕变测试的重要因素。

表 1 蜕变测试结果分析

序号	输入关系	输出关系	测试结论
1	满足	满足	不能得出结论
2	满足	不满足	程序存在错误
3	不满足	满足	无实际意义
4	不满足	不满足	无实际意义

Murphy 等针对数值计算函数提出了 6 条通用蜕变关系^[8],如表 2 所示。6 类常见蜕变关系可以作为构造蜕变关系的依据,但更需要针对不同待测程序的特性构造不同的蜕变关系。针对图像处理软件的特点,可以从以下几个方面着手构建蜕变关系:

(1)几何属性。对原始输入图像做几何变换可以得到新的输入图像,这样并未影响图像的内部布局,因

此两幅图像的生长结果应当保持相应的几何关系。

(2)算法特性。给定输入图像后,区域增长程序的输出受控制参数的影响。Chen 等提出了基于策略选择的系统构建蜕变关系的方法^[9],即改变一个输入参数,控制其他输入参数,查看输出结果的变化是否符合预期。

表 2 常见数值蜕变关系

序号	关系名称	输入变换
1	加法	输入值增加一个常数
2	乘法	输入值乘以一个常数
3	变序	改变输入元素顺序
4	取反	对每一个输入元素取反
5	增加元素	输入数据增加一个元素
6	减少元素	输入数据删除一个元素

1.2 原始测试用例自动生成

原始测试用例生成是影响测试效率的关键环节,现有的测试用例生成技术主要有特殊值选取、随机值选取和迭代生成等^[10]。目前应用最广泛的是随机值选取和特殊值选取两种方法^[11]。特殊值选取与待测程序和测试人员相关度很大,且生成的测试用例数量有限,难以保证测试充分性。随机值选取的测试效果优于特殊值,且可视为无偏的,但是测试过程带有一定的盲目性。迭代生成的方法是将每次产生的衍生用例作为下一次测试的原始用例,但这种方法容易造成状态爆炸问题,实际使用范围有限。对于图像区域增长程序,原始测试用例包含图像数据和程序控制参数。

原始输入图像的常见来源有两种,一是从标准测试数据集中选取,二是由测试人员自动生成。从测试图像数据集选取也可分为随机选取和启发式选取。标准数据集存在一个缺点,即通用的测试数据很难涵盖待测程序的特定功能,因此就产生了根据需求自动生成图像的方法。最直接的图像生成方法就是随机生成或者依据概率生成,这种简单的方法的弊端就是生成的图像脱离现实,参考价值不足。Tahir 等提出了利用符号执行技术生成测试图像^[12],Matthew 等提出利用程序性噪声生成测试图像^[13],Hu 等提出利用模板生成灰度图像^[14],这些方法为图像自动生成提供了很好的借鉴。

在给定输入图像后,种子点的选择和阈值设定将直接影响图像区域生长效果。文献[15]介绍了种子点的自动选取方法,生长阈值和生长个数阈值采用 Otsu 方法全局最佳自动生成。

1.3 衍生测试用例自动生成

根据蜕变关系的定义,可以将蜕变关系分解为输

入关系、程序函数关系和输出关系。衍生测试用例是由原始测试用例根据输入关系生成的,此项工作可以由计算机自动完成。值得注意的是,图像处理软件中,测试用例包含图像数据和控制参数,衍生用例的生成需要同时对两者进行处理。

1.4 测试结果自动判定

图像的输入数据可以视为函数 $f(x,y)$,其中 (x,y) 对应图像像素点坐标, $f(x,y)$ 就是其对应像素点上的属性值,因此区域增长后生成的图像依然可以看作一个函数。根据蜕变关系中的输出关系,原始测试用例和衍生测试用例的输出是否满足蜕变关系就可以由计算机来自动判定,提高了判定的效率和准确性。

1.5 测试充分性判定

蜕变测试过程中,不满足蜕变关系意味着程序错误,然而输出之间满足蜕变关系并不能保证程序正确。如果程序是正确的,那么测试终止条件和测试充分性就是需要考虑的问题。为此测试框架引入了测试充分性判定。

当选定一幅图像和一个蜕变关系后,自动执行测试流程,执行完毕后,计算程序路径覆盖率,若不满足覆盖率要求,则依次使用下一个蜕变关系,若蜕变关系使用完毕,则使用下一幅图像,直至覆盖率满足测试要求,整改测试过程结束。考虑到部分程序有一定冗余,部分路径可能永远不能覆盖,若到达一定测试阶段,测试覆盖率不满足测试的要求,但也不再增长,亦可结束测试。

1.6 蜕变关系优化

不同的蜕变关系其检错能力也不一样。在测试过程中,为了提高测试效率,需要不断优化测试过程,蜕变关系优化就是选取一些检错能力强的蜕变关系,剔

除检错能力弱的蜕变关系。总的来讲,有效的蜕变关系会涉及到程序的核心功能,程序执行路径也会尽量不同。对于正确的程序,可以在待测程序中植入变异,计算各个蜕变关系的检错能力,优化蜕变关系,以提高测试效率。

2 实例验证

某雷达图像处理软件是一款用于识别雷达图像中重点目标的软件,作者所在的军用软件测评中心参与了该软件的定型测试,整个测试的过程按照上述框架完成。

2.1 蜕变关系构造

根据需求规格说明以及程序源代码,从几何属性、数值属性和算法特性三个方面构造了 14 条蜕变关系。

数值属性方面,对整个图像的灰度值做整体增减和缩放,那么区域生长后的结果应当保持不变。

几何属性方面,在确定起始种子点、生长阈值和生长个数阈值的情况下,原始输入图像绕坐标轴旋转或沿对角线转置可以得到新的输入图像,两幅图像的生长结果保持相应的几何关系。

算法特性方面,可以依次对种子点位置、生长阈值和生长个数阈值做调整,那么生长结果应当保持一定的关系。

2.2 测试用例生成

由于真实雷达图像涉密,测试过程中采用标准测试数据集来完成软件测试工作。Pascal voc^[16]是专门用于目标检测、图像分割的标准测试数据集,采取随机选取图像的方法,控制参数由前面介绍的方法自动确定。

2.3 测试充分性判定

测试目标设定的测试覆盖率为 100%。因此,在没有检测到错误的情况下,区域增长功能测试必须达到设定的覆盖率才能停止。程序测试覆盖率变化如图 2 所示。随着测试图像的增加,到第九幅图像的时候,覆盖率达到 100%,满足测试需求。

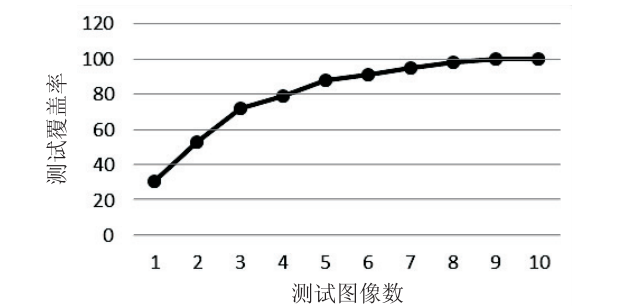


图 2 程序蜕变测试覆盖率

2.4 蜕变关系优化

为了检测不同蜕变关系的检错能力,可以在源

程序中依次植入三个变异,不满足蜕变关系就意味着检测到错误。Milu^[17]是一款针对 C 程序的变异算子生成工具,用该工具植入的变异如表 3 所示,这些都是开发过程中常见的错误,不会使输出结果出现较大的变化。

表 3 变异构造列表

变异	原始正确语句	植入变异后的语句
M1	for(k=0;k<4;k++)	for(k=0;k<3;k++)
M2	iCol>=0	iCol>0
M3	iRow=uiCurrH+nDctH[k]	iRow=uiCurrH+nDctW[k]

表 4 所示为各个蜕变关系的检错能力,所有的变异都被检测出来,证明了该方法的有效性,但是不同蜕变关系的检错能力不一样。为了提高测试效率,测试过程中筛选错误检测率在 50% 以上的蜕变关系,精简了蜕变关系集,节约了测试时间。

表 4 蜕变关系检错能力 %

MRs	Mutant1	Mutant2	Mutant3	Average
MR1	100	60	80	80
MR2	100	100	100	100
MR3	100	100	100	100
MR4	0	0	0	0
MR5	0	0	0	0
MR6	100	100	100	100
MR7	100	100	100	100
MR8	0	10	30	13.3
MR9	80	0	20	33.3
MR10	0	80	70	50
MR11	10	20	0	10
MR12	0	30	0	10
MR13	0	0	0	0
MR14	0	0	0	0

3 结束语

近年来,图像处理软件在生活中扮演日益突出的作用,因此软件的正确性和可靠性需要得到保证。为了缓解测试判定问题和推动测试过程的自动化标准化,提出了图像区域增长程序的蜕变测试框架。该框架从三个层面解决了此问题:蜕变测试的同时也产生了测试用例,相比于随机测试,该方法生成了更多的测

试数据;在缺乏测试判定的情况下,通过检查蜕变关系有效缓解了测试判定问题;该方法结合测试充分性判定,规范了测试过程,确保了软件测试质量。

参考文献:

- [1] ANGELINA S, SURESH L P, VENI S H K. Image segmentation based on genetic algorithm for region growth and region merging [C]//International conference on computing, electronics and electrical technologies. Kumaracoil, India: IEEE, 2012: 970–974.
- [2] BARR E T, HARMAN M, MCMINN P, et al. The oracle problem in software testing: a survey [J]. IEEE Transactions on Software Engineering, 2015, 41(5): 507–525.
- [3] MAYER J, GUDERLEI R. On random testing of image processing applications [C]//International conference on quality software. Beijing, China: IEEE, 2006: 85–92.
- [4] JAMEEL T, LIN Mengxiang, LIU Chao. Automatic test oracle for image processing applications using support vector machines [C]//IEEE international conference on software engineering and service science. Beijing, China: IEEE, 2015: 1110–1113.
- [5] KANEWALA U, BIEMAN J M, BEN-HUR A. Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels [J]. Software Testing Verification & Reliability, 2016, 26(3): 245–269.
- [6] CHEN T Y. Metamorphic testing: a simple method for alleviating the test oracle problem [C]//IEEE/ACM international workshop on automation of software test. Florence, Italy: IEEE, 2015: 53–54.
- [7] SEGURA S, FRASER G, SANCHEZ A B, et al. A survey on metamorphic testing [J]. IEEE Transactions on Software Engineering, 2016, 42(9): 805–824.
- [8] KANEWALA U, BIEMAN J M. Using machine learning techniques to detect metamorphic relations for programs without test oracles [C]//IEEE international symposium on software reliability engineering. Pasadena, CA, USA: IEEE, 2013: 1–10.
- [9] CHEN T Y, POON P L, XIE X. METRIC: metamorphic relation identification based on the category-choice framework [J]. Journal of Systems & Software, 2016, 116: 177–190.
- [10] 董国伟, 郭涛, 张普含, 等. 基于路径分析和迭代蜕变测试的 Bug 检测方法研究 [C]//信息安全漏洞分析与风险评估大会. 出版地不详: 中国信息安全测评中心, 2013: 211–224.
- [11] 董国伟, 徐宝文, 陈林, 等. 蜕变测试技术综述 [J]. 计算机科学与探索, 2009, 3(2): 130–143.
- [12] JAMEEL T, LIN Mengxiang, LI He, et al. Test image generation using segmental symbolic evaluation for unit testing [C]//15th IEEE/ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing. Las Vegas, NV, USA: IEEE, 2014: 1–6.
- [13] PATRICK M, CASTLE M D, STUTT R O J H, et al. Automatic test image generation using procedural noise [C]//IEEE/ACM international conference on automated software engineering. Singapore: IEEE, 2016: 654–659.
- [14] HU Shuilan, CHEN Shaoqiang. A contour generation algorithm for grayscale images using templates [C]//International congress on image and signal processing. Dalian, China: IEEE, 2015: 253–257.
- [15] 刘俊, 马燕, 陈坤. 一种改进的基于区域生长的彩色图像分割方法 [J]. 计算机应用与软件, 2012, 29(11): 288–291.
- [16] 杨扬, 李善平. 分割位置提示的可变形部件模型快速目标检测 [J]. 自动化学报, 2012, 38(4): 540–548.
- [17] JIA Yue, HARMAN M. MILU: a customizable, runtime-optimized higher order mutation testing tool for the full c language [C]//Testing: academic & industrial conference – practice and research techniques. Windsor, UK: IEEE, 2008: 94–98.