Evaluating the Effectiveness of Metamorphic Testing on Edge Detection Programs

K. Y. Sim, D. M. L. Wong, and T. Y. Hii

***Abstract*—Programmer's errors in implementing edge detection algorithms could induce faults in edge detection programs. We study Sobel edge detection programs in C and evaluate the effectiveness of Metamorphic Testing technique in detecting faulty edge detection programs. We found that the fault detection effectiveness varies for different metamorphic relations used for testing. Contrary to common believe that faults in image processing programs can be detected with any non-trivial image as test input, our experiment results show that there exists subtle fault that can only be detected when images with certain properties are used as test inputs. Based on these results, we propose general guides for using metamorphic testing to detect faults in edge detection programs.**

***Index Terms*—Component, edge detection, metamorphic testing, software testing.**

I. Introduction

An edge in digital image can be defined as the boundary between two regions separated by two relatively distinct gray levels [1]. Hence, edge detection is the process of localizing the abrupt changes in the gray level of an image [2]. Edge detection is an important pre-processing step in many digital image processing and computer vision operations such as feature extraction and detection, image enhancement, compression, retrieval, watermarking, hiding, restoration and registration [3].

Well-known algorithms such as Sobel and Canny have been developed to perform edge detection [4], [5]. However, programmer's errors in implementing these algorithms could induce faults in edge detection programs. Conventional manual testing of edge detection programs suffers from two major shortfalls. Firstly, it relies on tester's subjective visual judgment to compare the input and output images and decide if the edge detection program is implemented correctly. Secondly, as manual testing is time consuming and labor intensive, only few standard test images such as "Lena" are used for testing. As a result of such unreliable testing approach, edge detection programs may contain undetected faults and produce deteriorated or incorrect output images.

Manuscript received September 13, 2012; revised November 26, 2012. This work was supported in part by Malaysian Governement's MOSTI ScienceFund 01-02-14-SF0005 and MOHE FRGS (FRGS/2/2010/TK/SWIN/02/03).

K. Y. Sim is with the School of Engineering, Computing and Science, Swinburne University of Technology (Sarawak Campus), Sarawak, Malaysia (e-mail: ksim@swinburne.edu.my).

M. L. D. Wong is with the Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, P.R. China (e-mail: Dennis.Wong@xjtlu.edu.cn).

T. Y. Hii is with the Faculty of Engineering, Multimedia University, Cyberjaya, Malaysia (e-mail: tun_tun88@hotmail.com).

In order to eliminate subjective human judgment in testing of image processing program, Mayer [6] introduced a statistical approach to judge the correctness of output images. However, this approach can only be used if the statistical distribution of the output images is known in advance. It assumes that such statistical distribution, if exists, is sufficient to judge the correctness of the output images. Hence, its application is limited.

In a subsequent study, Mayer and Guderlei [7] have proposed random [8] and binary [9] models to randomly generate a large quantity of binary images to test a Euclidian Distance Transformation [10] program. This is to overcome the reliance on few standard test images such as "Lena". Metamorphic testing technique [11] is adopted to generate additional test cases and detect faults in the test output images. Through metamorphic testing and model generated images, they have successfully automated the testing process for Euclidean Distance Transformation programs.

Even though edge detection is an important and widely used pre-processing operation in digital image processing, testing of edge detection programs has not received much attention. Any fault in its implementation will have significant impacts on subsequent operations. In this paper, we study the effectiveness of metamorphic testing technique in detecting faults in Sobel edge detection programs. This paper makes the following contributions.

1) We report the effectiveness of metamorphic testing in detecting faulty edge detection programs.
2) We proposed the use of real images from published image libraries as test inputs instead of model generated images. We argued and presented the advantages of real images over model generated images.
3) We showed that there exists subtle faults that can only be detected when images with certain properties are used as inputs.
4) We proposed general guides for the selection of images to effectively detect faults in edge detection program.

The remainder of this paper is organized as follows: Section II presents the concept of metamorphic testing technique. Types of faulty Sobel edge detection programs are presented in Section III. Section IV outlines the experiments to evaluate the effectiveness of metamorphic testing and present the results. Section V discusses the findings and concludes the paper.

II. Metamorphic Testing

Metamorphic testing was first coined by Chen et al. [11] as a way to generate new test inputs and detect faults in program under test even if the correct expected output is unknown.

Metamorphic testing is a property based testing technique. In metamorphic testing, the necessary properties for correct program implementation are first identified as metamorphic relations. Base on existing test inputs (known as source test inputs), new test inputs (known as follow-up test inputs) are generated based the metamorphic relations. If the outputs of source test inputs and follow-up test inputs violate the metamorphic relation, then we can conclude that the program under test is faulty.

To define a metamorphic relation (MR), let,

$I_s = \{T_1, T_2, \ldots, T_k\}$ be a set of test inputs to a function f, where $k \geq 1$. I_s is known as the source test inputs.

$O_s = \{f(T_1), f(T_2), \ldots, f(T_k)\}$ be the set of outputs correspond to I_s.

$S = \{f(Ts_1), f(Ts_2), \ldots, f(Ts_m)\}$ be a subset of O_s where $m \geq 0$.

$I_f = \{T_{k+1}, T_{k+2}, \ldots, T_n\}$ be another set of test inputs to f, where $n \geq k+1$. I_f is know the as the follow up test inputs.

$O_f = \{f(T_{k+1}), f(T_{k+2}), \ldots, f(T_n)\}$ be the corresponding set of outputs for I_f.

$R_I(T_1, T_2, \ldots, T_k, f(Ts_1), f(Ts_2), \ldots, f(Ts_m), T_{k+1}, T_{k+2}, \ldots, T_n)$ be the test input relation.

$R_O(T_1, T_2, \ldots, T_n, f(T_1), f(T_2), \ldots, f(T_n))$ be the test output relation.

If there exists a relation R_I among I_s, S and I_f, and another relation R_O among I_s, I_f, O_s and O_f such that R_O must be satisfied whenever R_I is satisfied, then, an MR can be defined as:

MR: If $R_I(T_1, T_2, \ldots, T_k, f(Ts_1), f(Ts_2), \ldots, f(Ts_m), T_{k+1}, T_{k+2}, \ldots, T_n)$, then $R_O(T_1, T_2, \ldots, T_n, f(T_1), f(T_2), \ldots, f(T_n))$.

To illustrate the concept of metamorphic testing, consider the following example. Suppose that a program has been written to compute the sine function $f(x) = \sin(x)$. Let $T_1 = 36.5°$ be a test input to the program and $f(T_1) = 0.5948$ is the program output. We can use the trigonometry identity $\sin(x) = -\sin(x+180°)$ to derive the metamorphic relation. By using $T_1 = 36.5°$ as the source test input, a follow up test input $T_2 = 36.5° + 180°$ (that is $216.5°$) can be generated based on the trigonometric identity as metamorphic relation. If the program output for $f(T_2)$ is not the negation of $f(T_1)$ (that is, -0.5948) then we can conclude that the program under test is faulty. From this example, it can be seen metamorphic testing can detect the presence of fault even though the correct expected output is not known beforehand (we do not know beforehand if 0.5948 is the correct answer for $\sin(36.5°)$). This is particularly useful for many image processing programs because the correct expected output is unknown and not available beforehand for output verification.

We define the following metamorphic relations (MRs) to detect possible faults in edge detection program. Let *Im* be at the input image, $E(Im)$ be the corresponding output image of edge detection program,

- MR1: $C(E(Im)) = E(C(Im))$

where $C(.)$ is a 90° counter-clockwise rotation. The output of edge detection followed rotation should be the same as rotation followed by edge detection for input image *Im*.

- MR2: $Mx(E(Im)) = E(Mx(Im))$

where $Mx(.)$ is the reflection at the ordinate. The output of edge detection followed by reflection at the ordinate should be the same as reflection at the ordinate followed by edge detection for input image *Im*.

- MR3: $My(E(Im)) = E(My(Im))$

where $My(.)$ is the reflection at the abscissa. The output of edge detection followed by reflection at the abscissa should be the same as reflection at the abscissa followed by edge detection for input image *Im*.

- MR4: $T(E(Im) = E(T(Im))$

where $T(.)$ is a transposition. The output of edge detection followed by transposition should be the same as transposition followed by edge detection for input image *Im*.

These metamorphic relations are identified and adapted from [7] based on the necessary property for correct implementation of edge detection program. As all the metamorphic relations are defined as identities, the fault detection process can be automated by performing pixel to pixel comparison to the output images. If the output images are not identical, then the metamorphic relation is violated and we can conclude that the edge detection program under test is faulty. Hence, subjective human visual judgment can be eliminated from the testing process.

III. Faulty Edge Detection Programs

A collection of faulty edge detection programs that contain known faults are needed to evaluate the fault detection effectiveness of the metamorphic testing techniques presented in the last section. In this section, we present two categories of faulty edge detection programs used in this study.

A. Programs with Seeded Single Operator Fault

A single operator fault is seeded into the edge detection program implementing Sobel algorithm in *C* programming language. Two subtypes of operators are used here, namely, the logical operators (AND, OR, NOT) and the relational operators (>, >=, <, <=, ==, !=). For each occurrence of these operators, the operator is replaced with another operator from the same subtypes to create a faulty edge detection program. For example, an AND operator in the program will be replaced with an OR operator to create a faulty program. In this study, one faulty program contains only one fault. Programs with one fault are general harder to detect than programs with multiple faults. Through error seeding, 30 versions faulty edge detection programs have been produced.

B. Program with Stride Implementation Fault

Wrong interpretation or incorrect implementation of algorithm specifications can results in faulty program too. In a related study [12], we have encountered a stride implementation fault in the early stage of Sobel algorithm implementation in C. A stride implementation fault occurs when the image is processed up to the visible horizontal width instead of the "stride" width. The stride width is the actual horizontal dimension of the image data array. When an image is saved, the horizontal dimension of the image will be padded until the closest multiple of four for efficiency purpose. To the best of our knowledge, it has not been reported in any testing literature.

IV. Experiments

Experiments are conducted to evaluate the effectiveness of metamorphic testing in detecting the faulty edge detection programs. The fault detection effectiveness, E, is defined in (1).

$$E = \frac{\text{Number of Detected Faulty Programs}}{\text{Total Number of Faulty Programs}} \times 100\% \quad (1)$$

The fault detection effectiveness is evaluated for each metamorphic relation presented in the previous section. A faulty program is said to be detected by an MR if the metamorphic relation is violated.

A. Test Inputs

To conduct metamorphic testing, a collection of images need to be selected or generated as source test inputs. We propose the use of real images (that is, camera captured images) from published image libraries as test inputs as opposed to model generated images proposed in [7].

Unlike model generated images, real images do not require special tools to generate. Hence, it is a more practical and accessible choice for program testers. Real images also can have higher complexity and diversity in image content. They are more unlikely to suffer from systematic bias caused by parameter settings in image generating models. Furthermore, a large collection of real images are available through publicly accessible image libraries[1].

For the purpose of experimentation, a total of 30 images of different format (BMP, JPEG, PNG) have been sampled from the image libraries in Table I. The 30 images selected are used as the test inputs to test each of the faulty programs. If any of the images trigger a violation of the metamorphic relations, the faulty program is said to be detected.

B. Results

A total of 31 faulty programs (30 faulty programs with single operator fault plus one program with stride implementation fault) have been used in the experiment to evaluate the fault detection effectiveness of metamorphic testing. More particularly, the fault detection effectiveness of each metamorphic relation is recorded. The experiment results are presented in Table II. The overall fault detection effectiveness (combining all metamorphic relations) is presented in Table III.

From Table II, it could be observed that MR2 has the highest fault detection effectiveness (77%), followed by MR3. MR1 and MR4 have the lowest fault detection effectiveness. However, the fault detection effectiveness improves significantly when all the four metamorphic relations are used in testing. As shown in Table III, 90% of faulty programs violate at least one of the four metamorphic relations.

[1] A list of image library websites are available at http://www.cs.cmu.edu/~cil/v-images.html at the time of writing

TABLE I: The Collection of 30 Images Used As Test Inputs.

Image Library	Number of Images Sampled
http://www.hlevkin.com/TestImages/classic.htm	5
http://www.imagecompression.info/test_images/	10
http://r0k.us/graphics/kodak/	15

TABLE II: Fault Detection Effectiveness for Each Metamorphic Relation.

Metamorphic Relations	MR1	MR2	MR3	MR4
No. of Undetected Faulty Programs (no violation to this MR)	17	7	10	17
No. of Detected Faulty Programs (violation of this MR)	14	24	21	14
Total No. of Faulty Program	31	31	31	31
E, Fault Detection Effectiveness	45%	77%	68%	45%

TABLE III: Overall Fault Detection Effectiveness

No. of Undetected Faulty Programs (No Violation to Any MR)	3
No. of Detected Faulty Programs (Violation to at Least One MR)	28
Total No. of Faulty Programs	31
E, Fault Detection Effectiveness	90%

1) Observations on the detection of faulty programs with a single operator fault.

For the 30 faulty programs with a single operator fault, it was found that the faults that can be detected by MR1 and MR4 can also be detected by MR2 and MR3, but not vice-versa. This suggests MR1 and MR4 are redundant in detecting edge detection program with a single operator fault. However, this observation may not hold for programs with other types of faults.

Upon further investigation, it was found that either all or none of the 30 images used as test inputs will cause violation to an MR for a particular faulty program. This may suggest that any of the 30 images is as good as the others in detecting the faulty program. This observation reinforces that general believe that fault in image processing program can be detected with any non-trivial image as test input.

2) Observations on the detection of the faulty program with stride implementation fault.

Contradictory to the observation made on programs with single operator fault, the faulty program with stride implementation fault can only be detected by MR1 and MR4 but not MR2 and MR3. Furthermore, as shown in Table IV, the stride implementation fault can only be detected by five out of the 30 images used as test inputs when tested with MR1 and MR4. Upon further examination, it was found that the five images that detects this faulty program have a horizontal widths that are not multiple of four. This

observation defies the general believe that fault in image processing application can be detected with any non-trivial image as test input. It also implies that fault detection by metamorphic testing also rely on the properties of the input images.

TABLE IV: DETECTION OF FAULTY PROGRAM WITH STRIDE IMPLEMENTATION ERROR

Metamorphic Relations	MR1	MR2	MR3	MR4
No. of Test Images that Violate the MR	5	0	0	5
No. of Test Images that do not Violate the MR	25	30	30	25
Total No. of Test Images	30	30	30	30
Detection Rate for Test Inputs	17%	0%	0%	17%

V. DISCUSSIONS

The results of the experiments show that metamorphic testing is very effective in detecting faults in the implementations of edge detection algorithm. It can detect up to 90% of the faulty edge detection programs under study. This is encouraging finding for testing of edge detection programs as metamorphic testing can be conducted automatically without relying subjective human visual judgment.

Even though some metamorphic relations have higher fault detection effectiveness than the others, the metamorphic relations with lower fault detection effectiveness are not redundant. This is because there are certain faulty programs that can only be detected by these metamorphic relations. Furthermore, the experiment results also show that the stride implementation fault is a subtle fault that can only be detected when images with certain property are used as test inputs.

From the above observations, we propose the following general guides in using metamorphic testing to detect faults in edge detection programs.

1) All metamorphic relations identified should be used to test edge detection programs because the metamorphic relations may work in complementary to detect different faults that may exist in the edge detection program. As each MR has different fault detection effectiveness, we recommend that testing is conducted with MR with higher fault detection effectiveness first followed by the lower ones. Such prioritization will improve the probability of detecting more faults earlier in the testing process.
2) Real images can be used as test inputs for metamorphic testing instead of model generated images. A large collection of real images are publicly accessible to testers and are more unlikely to suffer from constraints that exist on model generated images.
3) A variety of real images should be selected as test inputs. Images with different dimensions, color depth and formats should be used as test inputs. This is because there may exist certain subtle faults that can only be detected by images with certain properties.

VI. CONCLUSION

In conclusion, our study has shown that metamorphic testing is effective in detecting faulty edge detection programs. Our experiment results have shown that even though the fault detection effectiveness of metamorphic relations varies, they are complementary in detecting different faulty edge detection programs. Contrary to common believe that faults in image processing program can be detected with any non-trivial image as test input, we have found that there exists subtle fault that can only be detected when images with certain properties are used as test inputs. Finally we have outlined general guides for using metamorphic testing to detect faults in edge detection programs.

ACKNOWLEDGMENT

This work is supported via Malaysian Government MOSTI ScienceFund 01-02-14-SF0005 and MOHE FRGS (FRGS/2/2010/TK/SWIN/02/03).

REFERENCES

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Reading: MA Addison Wesley, 2 nd ed., 2002.
[2] G. Markus, A. E. E. Kwae, and R. K. Mansur, "Edge detection in medical images using a genetic algorithm," *IEEE Trans. on Medical Imaging*, vol. 17, pp. 469-474, 1998.
[3] D. Ziou and S. Tabbone, "Edge detection techniques - an overview," *Intl. J. Pattern Recognition and Image Analysis*, vol. 8, pp. 537-559, 1998.
[4] S. E. Umbaugh, *Computer Vision and Image Processing: A Practical Approach Using CVIPtools*, Upper Saddle, NJ, Prentice Hall, 1998.
[5] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679-714, 1986.
[6] J. Mayer, "On testing image processing applications with statistical methods," in *Proc. Software Engineering 2005 (SE 2005)*, vol. P-64 of Lecture Notes in Informatics, Gesellschaft f ¨ur Informatik e.V., K ¨ollen Druck+Verlag GmbH, Bonn, 2005, pp. 69-78.
[7] J. Mayer and R. Guderlei, "On random testing of image processing applications," in *Proc. Sixth International Conference on Quality Software (QSIC'06)*, IEEE Computer Society, 2006, pp. 85-92.
[8] K. N. King and A. J. Offutt, "A fortran language system for mutation-based software testing," *Software Practice and Experience*, vol. 21, pp. 685-718, 1991.
[9] D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic geometry and applications*, John Wiley and Sons, Chichester, 1995.
[10] P. Danielsson, "Euclidean distance mapping," *Computer Graphics and Image Processing*, vol. 14, pp. 227–248, 1980.
[11] T. Chen, S. Cheung, and S. Yiu, "Metamorphic testing: a new approach for generating next test cases," *Technical report, Technical Report HKUST-CS98-01*, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, 1998.
[12] N. S. Chong, M. L. D. Wong, K. Y. Sim, B. M. Goi, and H. C. Ling, "Hardware accelerated Sobel edge detection using common GPU parallel computing platform," in *Proc. Conf. on Electrical and Electronic Technology, 4th World Engineering Congress (CEET-WEC2010)*, Kuching, Malaysia, August 2010.

K. Y. Sim received his BEng (Hons) from the National University of Malaysia in 1999 and Masters of Computer Science from University of Malaya, Malaysia in 2001. He is currently a Senior Lecturer and the Associate Head for Program Development and Accreditation at the School of Engineering, Computing and Science, Swinburne University of Technology, Sarawak Campus, Malaysia. His research interests include software testing and for embedded system testing. Mr. Sim is a member of IEEE and IEEE Computer Society.

M. L. Dennis Wong received his BEng (Hons) in Electronics and Communication Engineering and his PhD from the Department of Electrical Engineering and Electronics, University of Liverpool, Merseyside, U.K. in 1999 and 2004 respectively. He is currently an Associate Professor at the Department of Electrical and Electronic Engineering, Xi'an Jiaotong Liverpool University (XJTLU), Suzhou, Jiangsu Province, China. Prior to his current appointment, he was a Lecturer, a Senior Lecturer, later the Associate Head of School (Engineering) at the School of Engineering, Computing and Science, Swinburne University of Technology (Sarawak Campus), Malaysia from 2004 to 2011. His research interests include statistical signal processing and pattern classification, machine condition monitoring, and VLSI Design for digital signal processing. Dr. Wong is a Senior Member of the IEEE and a Member of the IET.

T. Y. Hii received his Bachelor of Engineering (Hons) from Multimedia University, Malaysia in 2011.