CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

Evaluation of Metamorphic Relations of Image Processing Operations using Mutation Testing

by

Fakeeha Jafari

A dissertation submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Computing
Department of Computer Science

2024

Evaluation of Metamorphic Relations of Image Processing Operations using Mutation Testing

By

Fakeeha Jafari

(DCS 163001)

Dr. Radu Prodan, Profeessor

University of Klagenfurt, Austria

(Foreign Evaluator 1)

Dr. Seifedine Kadry, Professor

Noroff University College, Norway

(Foreign Evaluator 2)

Dr. Aamer Nadeem

(Research Supervisor)

Dr. Abdul Basit Siddiqui

(Head, Department of Computer Science)

Dr. Muhammad Abdul Qadir

(Dean, Faculty of Computing)

DEPARTMENT OF COMPUTER SCIENCE

CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

ISLAMABAD

2024

Alhamdolilah

CAPITAL UNIVERSITY OF SCIENCE & TECHNOLOGY
ISLAMABAD

Expressway, Kahuta Road, Zone-V, Islamabad
Phone:+92-51-111-555-666 Fax: +92-51-4486705
Email: info@cust.edu.pk Website: https://www.cust.edu.pk

CERTIFICATE OF APPROVAL

This is to certify that the research work presented in the dissertation, entitled "**Evaluation of Metamorphic Relations of Image Processing Operations using Mutation Testing**" was conducted under the supervision of **Dr. Aamer Nadeem**. No part of this dissertation has been submitted anywhere else for any other degree. This dissertation is submitted to the **Department of Computer Science, Capital University of Science and Technology** in partial fulfillment of the requirements for the degree of Doctor in Philosophy in the field of **Computer Science.** The open defence of the dissertation was conducted on **May 03, 2024**.

Student Name : Fakeeha Jafari (DCS163001)

The Examination Committee unanimously agrees to award PhD degree in the mentioned field.

Examination Committee :

(a) External Examiner 1: Dr. Tamim Ahmed Khan,
Professor
Bahria University, Islamabad

(b) External Examiner 2: Dr. Yasir Hafeez
Professor
UIIT, PMAS-AAU, Rawalpindi

(c) Internal Examiner : Dr. Sabeen Masood
Assistant Professor
CUST, Islamabad

Supervisor Name : Dr. Aamer Nadeem
Professor
CUST, Islamabad

Name of HoD : Dr. Abdul Basit Siddiqui
Associate Professor
CUST, Islamabad

Name of Dean : Dr. Muhammad Abdul Qadir
Professor
CUST, Islamabad

AUTHOR'S DECLARATION

I, **Fakeeha Jafari** (**Registration No. DCS163001**), hereby state that my dissertation titled, '**Evaluation of Metamorphic Relations of Image Processing Operations using Mutation Testing'** is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/ world.

At any time, if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my PhD Degree.

(Fakeeha Jafari)

Dated: 03 May, 2024 Registration No: DCS163001

PLAGIARISM UNDERTAKING

I solemnly declare that research work presented in the dissertation titled "**Evaluation of Metamorphic Relations of Image Processing Operations using Mutation Testing**" is solely my research work with no significant contribution from any other person. Small contribution/ help wherever taken has been duly acknowledged and that complete dissertation has been written by me.

I understand the zero-tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled dissertation declare that no portion of my dissertation has been plagiarized and any material used as reference is properly referred/ cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled dissertation even after award of PhD Degree, the University reserves the right to withdraw/ revoke my PhD degree and that HEC and the University have the right to publish my name on the HEC/ University Website on which names of students are placed who submitted plagiarized dissertation.

(**Fakeeha Jafari**)

Dated: 03 May, 2024 Registration No: DCS163001

List of Publications

It is certified that following publication(s) have been made out of the research work that has been carried out for this dissertation:-

1. **Jafari, F.**, Nadeem, A., and Zaman, Q.u. "Evaluation of Metamorphic Testing for Edge Detection in MRI Brain Diagnostics," *Appl. Sci*, vol. 12, no. 17, pp. 8684, 2022.

2. **Jafari, F.**, Nadeem, A. "Measuring Effectiveness of Metamorphic Relations for Image Processing Using Mutation Testing," *J.Imaging*, vol. 10, no. 4, pp. 87, 2024.

(Fakeeha Jafari)
Registration No: DCS 163001

Acknowledgement

I want to start by expressing my gratitude to Allah, the most Gracious and Merciful, for all of His blessings bestowed upon me and for providing the strength I needed to complete this task.

I want to sincerely thank my supervisor, Professor Dr. Aamer Nadeem, for his intellectual guidance and suggestions. He is an incredibly motivating professor who helped me out with my PhD studies. He gave me moral and technical support throughout this difficult journey, and kept me motivated. His incisive feedback encourages me to sharpen my abilities and critical reasoning in order to reach my research objectives. Without his invaluable assistance, I would not be able to finish this dissertation. I can't express enough gratitude for the support he gave me and the belief he had in my ability to complete this difficult and tiring expedition. I consider it a great honour to have finished my PhD research under his guidance. My appreciation also extends to my CSD members, Dr. Mudassir Sindhu, and Dr.Qamar-uz-Zaman for valuable suggestions and guidance. Additionally, I want to thank my friends who supported and encouraged me throughout this journey.

I dedicate this work to my parents who supported and stood by me throughout this journey. I would like to thank my husband who supported me financially and morally to pursue my PhD study, without him I would not be able to accomplish this work. I would also thank my sisters for their support, wise counsel and sympathetic ears. I would also thank my brother in law for being a great support throughout this journey. I also thank my father in law and mother in law for their support. In addition, a special note to my kids, they sacrificed a lot,thank you for your love and support.

Finally, I want to express my gratitude for the chance to study at the esteemed Capital University of Science and Technology. This opportunity has given me the chance to learn from and work with some of the most talented people.

(Fakeeha Jafari)

Abstract

Testing of an intricate plexus of advanced software system architecture of image processing applications is quite challenging due to the absence of test oracle. Metamorphic testing is a popular technique to improve the test oracle problem. Effectiveness of metamorphic testing is dependent on metamorphic relations (MRs), a necessary property of system under test. One of the important metrics to evaluate MRs is their fault detection capability as a low fault detection rate shows that the faults in the system under test are not fully identified.

Even though, various metamorphic testing approaches have been applied to evaluate IPAs, but there has been no substantial work performed for the evaluation of metamorphic testing itself. In this work, we have evaluated effectiveness of metamorphic testing on edge detection and morphological image operations of MRI images for the evaluation of metamorphic relations of image processing operations using mutation testing. The existing techniques of edge detection and morphological image operations for the evaluation of MRs are not comprehensive for two reasons; firstly, the source test cases are generated randomly from an incomplete sample population which is missing many image properties leading to a false positive error in testing results and secondly, a very smaller number of mutants is generated through very few mutation operators which have affected their fault detection capability to kill significant number of mutants.

A solution is proposed for the generation of source test cases by combining both black-box testing and white-box testing techniques. In the black-box testing technique, source test cases are divided into five classes based on image properties through strong equivalence class testing: image resolution, image bit depth, image horizontal dimension, image vertical dimension, and image type (T1-weighted images, T2-weighted images and flair-type images). In white-box testing, the selected test cases are further checked through code coverage to ensure complete coverage.

In our proposed framework we have used nine applicable mutation operators and determine the effectiveness of these operators. By using these applicable operators, we have ensured that all the possible numbers of mutants are generated for

our optimal fault detection rate. We have also proposed six new MRs for dilation and erosion operation. The fault detection capability of six newly proposed MRs and four existing edge detection MRs is determined through mutation testing.
We have compared the results of our proposed framework with the results of existing techniques. Results of evaluation of the proposed framework of four MRs of edge detection show an improvement in all the respective MRs especially in MR_1 and MR_4 with a fault detection rate of 76.54% and 69.13% respectively which is 32% and 24% improved than the existing technique. The fault detection rate of MR_2 and MR_3 is also improved by 1%. Similarly, results of dilation and erosion are compared with the existing technique. Results show that out of 8 MRs, the fault detection rate of four MRs are improved than the existing technique. In proposed framework, MR_1 is improved by 39%, MR_4 is improved by 0.5%, MR_6 is improved by 17%, and MR_8 is improved by 29%. While comparing our proposed MRs with the existing MRs of dilation and erosion operations, we have come to the conclusion that the proposed MRs complement the existing MRs effectively as the proposed MRs are are able to find those faults which are not identified by the existing MRs.

Contents

List of Figures

List of Tables

Abbreviations

AOD	Arithmetic Operator Deletion
AOR	Arithmetic Operator Replacement
COI	Conditional Operator Insertion
IPAs	Image Processing Applications
RIL	Reverse Iteration Loop
FDR	Fault Detection rate
IUT	Implementation Under Test
MR	Metamorphic Relation
MT	Metamorphic Testing
OIL	One Iteration Loop
ROR	Relational Operator Replacement
SDL	Statement Deletion
SIR	Slice Index Remove
SUT	System Under Test
ZIL	Zero Iteration Loop

Symbols

δ	Dilation operation
ε	Erosion Operation
$\oplus$	Dilation operation
c	Complement of image
$\ominus$	Erosion operation
θ	Threshold Value
s	Structuring Element
mt	Mutant

Chapter 1

Introduction

The importance of image processing applications (IPAs) is growing fast in our daily lives [1]. IPAs utilize algorithms to analyze the characteristics of an image using various methods and techniques. Digital images can be rotated, scaled, translated, and sheared by using geometric transformations. Also, in binary and grayscale images, different morphological operations such as erosion, dilation, skeletonization, opening and closing operations are used that are further used for filtering, thinning, and pruning of the images [2].

Nowadays, IPAs are widely used in safety, and mission critical systems such as medical radiology, biometric systems, surveillance systems etc. [1]. Testing of the software used in these critical systems is vital to ascertain the credibility of the results produced by these systems. Software testing is a common method to test and verify the quality of IPAs software [3]. In software testing, an oracle is a mechanism that ascertains that whether a software has successfully executed for a test case or not [4]. The Software is run for a specific test case, and the result (actual output) is compared with the anticipated result (expected output). If the output differs from what was anticipated, the program is said to be faulty [3]. In real life scenarios, the term "oracle problem" occurs when the oracle is either unavailable or it is very difficult or impossible to verify the test result of a specific test case [5]. Testing of IPAs is especially challenging due to the test oracle problem.

Among many solutions of test oracle problem, Metamorphic Testing (MT) is the

most popular testing technique that tackles the oracle problem in software testing of IPAs [6]. MT was first proposed by Chen at al. in 1998 [7]. Instead of focusing on ensuring that each distinct output is verified, MT looks at the relationships between the inputs and outputs of various iterations of the program under test. These relationships are called Metamorphic Relations (MRs), and they are the necessary properties of the intended program's functionality [8]. Additionally, source test cases are needed to check the functionality of SUT [9]. The source test cases are generated through traditional test case generation techniques such as random test case generation, coverage criteria based etc. From these source test cases, a set of new test cases known as follow-up test cases are constructed using MRs [10]. Afterwards, the source and follow-up test cases are given to the SUT. After the execution, if the output results of source and follow-up test cases obtained from SUT does not satisfy the output relation, then it shows that the program is faulty [11].

1.1 Motivation

Software testing is a crucial technique that is frequently used during development to find faults in a program [12]. This typically involves choosing a set of input data (test cases), running the program under test, and then examining the corresponding output data [13]. Test case consists of test value, action, or event and expected result [14]. The expected result is a predetermined outcome of the test cases, and the test case only passes if the actual result matches the expected result [15]. The test failure occurs if the actual result is different from the expected result [16]. The mechanism used to determine whether a test has succeeded or failed is known as the (test) oracle [17]. In many programs such as IPAs, search engines, simulators, or compilers, test oracle is not often well defined and the expected results are not obvious. This is the well-known test oracle problem. When the chosen test case fails to yield the expected output, oracle problem occurs [18]. It is common to refer to programs that have the Oracle Problem as "untestable," as testing is hardly ever beneficial when it is impossible to detect a program's output

as (in)correct [17].

A quarter of a century ago, MT was first developed by Tsong Yueh Chen to address the oracle problem. It is becoming more and more recognised as a helpful testing method by academics and business professionals [19]. In many domains, MT has shown to be effective for creating test cases and finding errors. Figure 1.1 shows the applications of MT to specific problem domains. Figure 1.1 shows twelve

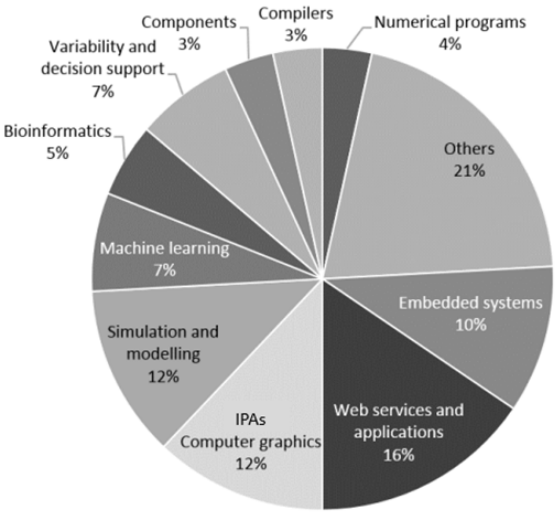

FIGURE 1.1: Metamorphic Testing Application Domains [20].

different application domains in MT. It is observed that the most popular domain is web services and applications (16%) followed by computer graphics (12%) and simulation and modeling (12%). In computer graphics, image processing (IP) is a crucial area because it can be extremely challenging to determine whether an IP system's behaviour satisfies its stated requirements or not [21]. Additionally, IPAs are becoming more and more popular for tasks like surveillance, medicine, bio-metrics etc,.

In MT, MRs are the intrinsic properties of the software being tested [22]. The MRs used for testing have a significant impact on how well MT works. According to studies, the effectiveness of MRs is related to their fault detection rate. Therefore, a good method for calculating the MR fault detection rate can somewhat increase the efficacy of MT [23]. The higher the fault detection rate, the higher is the chances to reveal faults [24]. For the detection of different faults, the effectiveness of various MRs varies because every MR is not capable to detect faults or for bug

manifestation. It's essential to use MRs with high fault-detection effectiveness to save time and resources [25].

Early research on MT has primarily focused on how to use this novel testing strategy to solve oracle problems using MRs. Efforts have been made to increase the effectiveness of failure detection in MT through evaluation of MRs and test case diversity. Studies in this thesis concentrate on evaluating the fault detection rate of MRs.

1.2 Problem Statement

The problem this thesis work addressing are as follows:

In existing literature, the method of generating random test cases is widely used because of its simplicity and un-biased nature. However, generating the test cases using this method (randomly) could lead to inadequate representation of parametric values because the sample population is not all-inclusive and complete itself.

In existing techniques, the evaluation of MRs is performed through mutation testing using a very few mutation operators such as ROR and LOR. This evaluation is not comprehensive because the fewer the mutation operators, the fewer the mutants generated which are not quite sufficient for comprehensive testing to measure the fault detection rate of MRs.

In the field of IP, the MRs are not adequate. Also, the fault detection rate of some of the MRs are very low and low fault detection rate implies the presence of the hidden bugs/faults in the program which are not detected.

1.3 Scope

MT is typically a highly and simply automatable testing approach to address the oracle problem [26]. MT is dependent on the fault detection rate of an MR. The

higher the fault detection rate, the higher the fault detection capability. The primary focus of this thesis work is to establish a comprehensive criterion to assess the effectiveness of an MR. Also the scope of our newly proposed framework includes the evaluation of the effectiveness of currently available MRs of IP operations such as edge detection, dilation, erosion, etc, using the use of newly developed criteria. Additionally, an assessment is performed in this thesis work for the adequacy of all currently available MRs collectively and to propose new MRs if needed.

1.4 Objectives

The core objectives of this thesis are as follows:

1. To improve the evaluation method of fault detection rate of MRs in MT.
   One of the objective of this research is to assess the effectiveness of fault detection rate of MRs in MT. In existing literature, to assess the fault detection rate of MRs, very few mutation operators (ROR, LOR) are used in mutation testing for the generation of mutants. These smaller number of mutants will affect the fault detection capability to kill significant number of mutants. Additionally, source test cases are generated randomly and random test case generation could lead to unfair distribution of parametric values. This unfair distribution, being non-comprehensive, may provide inaccurate results of the evaluation process.

2. To propose new MRs in the field of IP.
   In the existing literature, there are not enough MRs in the field of IP for comprehensive testing.

3. To compare fault detection effectiveness of existing MRs and proposed MRs in MT.
   The last objective is to evaluate the fault detection rate of existing MRs using our proposed methodology. Additionally, the proposed MRs are compared with the existing MRs for any improvement.

1.5 Research Questions

The research questions of this thesis are as follows:

1. How to assess and improve the evaluation of fault detection rate of MRs for MT ?

   RQ1 focuses to measure the evaluation of fault detection rate of MRs for MT. In existing techniques, the evaluation of MRs is performed through mutation testing using a very few mutation operators such as ROR and LOR. This evaluation is not comprehensive because the fewer the mutation operators, the fewer the mutants generated which are not quite sufficient for comprehensive testing to accurately measure the fault detection rate of MRs. We have used all possible nine mutation operators to evaluate the MRs of edge detection and dilation and erosion operations. This RQ is addressed in Chapter 6.

2. How to create new MRs for MT ?

   RQ2 emphasis on the creation of new MRs in the field of IP as the existing MRs are not adequate for comprehensive testing. In existing literature, there are eight MRs for dilation and erosion operations. We have created six new MRs for dilation and erosion operations as well. Additionally, some more MRs are constructed after composing the composite MRs. This RQ is addressed in Chapter 5.

3. How do existing and proposed MRs compare with respect to fault detection effectiveness in MT ?

   RQ3 emphasis on the evaluation of existing MRs as well as the evaluation of proposed MRs as the evaluation of existing MRs is not comprehensive with respect to their fault detection rates. The evaluation is performed on both the edge detection MRs and dilation and erosion MRs.The fault detection rates of proposed MRs and existing MRs are evaluated through mutation testing. Lastly, the effectiveness of proposed MRs and existing MRs is compared and analyzed. This RQ is addressed in Chapter 6.

1.6 Contributions

The contributions of this thesis work are listed below:

1. Currently, the method of generating random test cases is widely used because of its simplicity and un-biased nature. However, generating the test cases using this method (randomly) could lead to inadequate representation of parametric values because the sample population is not all-inclusive and complete itself. The contribution of my thesis work is to define an all-inclusive sample population ready for a true sample to be selected from with every parametric value having equal probability of selection. Source test cases are generated through a systematic way to ascertain that the generated test cases are truly random from an all-inclusive sample population. The proposed method uses strong equivalence class testing along with code coverage for the generation of source test cases.

2. In existing techniques, the evaluation of MRs is performed through mutation testing using a very few mutation operators such as ROR and LOR. This evaluation is not comprehensive because the fewer the mutation operators, the fewer the mutants generated which are not quite sufficient for comprehensive testing to accurately measure the fault detection rate of MRs . All possible nine mutation operators to evaluate the MRs of edge detection and dilation and erosion operations.

3. For dilation and erosion operations, we have proposed six new MRs (four general and two specific). The effectiveness of proposed MRs is determined through mutation testing.

4. In addition, we have constructed new MRs by composing the MRs. In the field of IP, composition has not been performed to-date.

5. For medical images such as MRI, CT scan, ultrasound, x-rays etc., MT has not been applied to-date. We have studied the implications of MT applied on MRI brain images. It has been suggested that which of the MRI image

(T1 weighted images, T2 weighted images, and flair images) is more useful for the diagnostic purpose.

1.7 Thesis Organization

The thesis' remaining chapters are organised as follows:
Chapter 2 presents a brief introduction about the challenges in testing IPAs such as oracle problem. Different methods have been discussed to alleviate the oracle problem. The chapter also discusses MT in detail, a method to handle oracle problem in IPAs is discussed in detail.
Chapter 3 presents the review of literature for MT in IPAs. It includes the papers related to the evaluation of MRs to improve the effectiveness of MT, the use of machine learning for output evaluation of images, and the enhancements in MT.
Chapter 4 presents the proposed framework to evaluate the MRs of IP operations such as edge detection and morphological image operations (dilation and erosion). The chapter also discusses the existing and proposed MRs of IP operations.
Chapter 5 discusses the results related to evaluation of MRs and mutation operators. The chapter also discusses the composition results in detail. The comparison of proposed framework with the existing techniques are also discussed in detail.
Chapter 6 derives from our published work where an improved canny edge detection algorithm is used to evaluate the metamorphic relations of edge detection. Furthermore, the output comparison is made using structure similarity image measure.
Chapter 7 concludes this research and summarises the main contributions made by the research. Finally, the proposed work's future prospects are discussed.

Chapter 2

Background

It is a challenging and expensive task to evaluate the output of those programs that involve a huge amount of data. For example, many aspects of our daily lives depend on IPAs, including biometrics, surveillance, and medical imaging etc,. These IPAs handle large amount of data and produce complex outputs [27]. As we all know that the software systems are error prone and their failure results in massive disaster [28]. So, the reliability and quality of these systems can be measured through testing [29].

2.1 Testing of Image Processing Applications

Software testing is a fundamental approach to determine the bugs in implementation under test (IUT) [30]. Testing is an evaluation process that determines whether the system is able to meet its specification or not. A test case in software testing is a written document with a set of test data, preconditions, anticipated outcomes, and post-conditions created for a particular test scenario in order to verify compliance with a specified requirement [31]. Expected result is the best result that can be achieved once a test case has been executed. For example, in the case of a calculator application, if you enter 4+8 and the actual result is 10, your test case fails because the expected result obtained for the same test case

is 12 [32]. Figure 2.1 shows the whole process of software testing. In case of IP, if we apply the operations (edge detection, dilation, erosion, segmentation, image region growth etc,.) on the images then we get the original output and we need expected output for comparison to see whether the original output generated is correct or not.

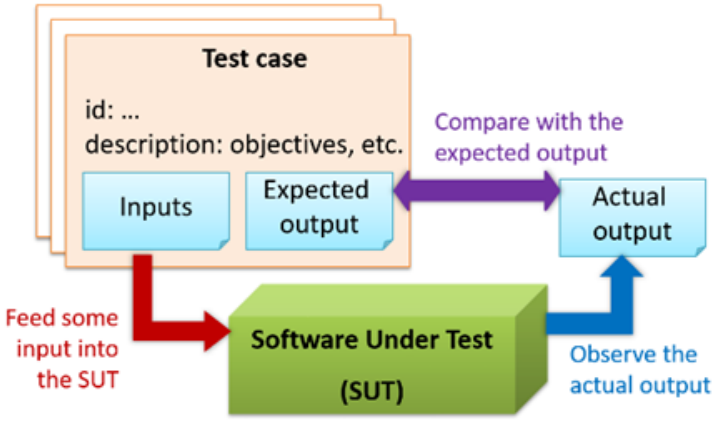

FIGURE 2.1: Process of Software Testing.

Figure 2.1 shows that every test case has its expected and actual results compared, and any discrepancy is labelled as a defect. Testing IPAs requires a lot more time and resources than testing traditional software because these applications are typically tested manually. As test inputs, those images are used whose expected output is already determined [33]. For example, in IP, edge detection is an operation which is used to compute the edges of an image. If we want to check whether the output edges computed by the edge detection operator is correct or not then we do not have a reference image (expected output) for comparison. This is the well known oracle problem where the expected results are not obvious.

2.2 Challenges in Testing Image Processing Applications

Testing of IPAs is a very challenging task due to its complications and visual semantics. Some of the challenges of testing the IPAs are given below:

2.2.1 Generation of Test Cases

Dijkstra emphasised that testing reveals faults, not their absence [34]. In software testing, generation of test cases is a vital and tedious task [35]. Finding test inputs that reflect the unexpected performance behaviour in the program under test is necessary for fault detection through testing, but this can be quite challenging [36]. The first challenge is to generate test cases (images) that are significant enough to explore individual parts of program under test and to check the presence of bugs in the implementation [30].

In literature, multiple methods have been discussed for the generation of input images e.g. random input generation, Boolean model, combinatorial techniques, symbolic evaluation method etc. Random input generation is considered unbiased and less complex as compared to other input generation methods or techniques. However, generating the test data randomly could lead to unfair distribution of parametric values. Also, the sample population is not comprehensive. Therefore, sample selected is also not true representation of sample population.

In literature, the image libraries, from where the test cases are selected, have images with same attribute values. For example, in one library, all the images which are used for testing have same resolution of 96dpi and in another library all the images have same horizontal and vertical dimensions. This unfair distribution, being non-comprehensive, may provide inaccurate results of the testing process.

2.2.2 Evaluation of Output Images

The second challenge is to correctly evaluate the output images. In literature, several image quality measures are used that provide accurate and close to perceived quality of the tested image such as peak signal to noise ratio (PSNR), root mean square error (RMSE), signal to noise ratio (SNR), structure similarity index measure (SSIM), and feature similarity index measure (FSIM) etc, [37]. The last two measures are comparatively new measures as compared to the others. Since subjective human evaluation is inconvenient, time-consuming, and expensive in

practise, these are substitute methods for quantifying the quality of visual images [38]. However, it is quite challenging to correctly evaluate the output images pixel by pixel [39]. For example, it is difficult to perform a pixel by pixel comparison of two images which seem alike visually but are in fact different [27]. When pixel by pixel comparison is not possible due to large pixel differences then it is difficult to compare the images at structure as well as at the semantic level. [40].

2.2.3 Oracle Problem

The third challenge in testing IPAs is the test oracle problem. In the field of IP, test oracle (a decision, whether a test case passes or fails) is not often well defined and the expected results are not obvious. This is the well-known test oracle problem. Test oracle problem is a mechanism to determine the ability to distinguish between correct and incorrect behavior of the system under test for a given input [41]. An oracle problem occurs when it is challenging to get the expected results from a chosen test case [27]. Because of oracle problem, testers use input images that can be handcrafted or images with well-defined expected output results [42]. The concept of oracle problem is depicted in Figure 2.2.

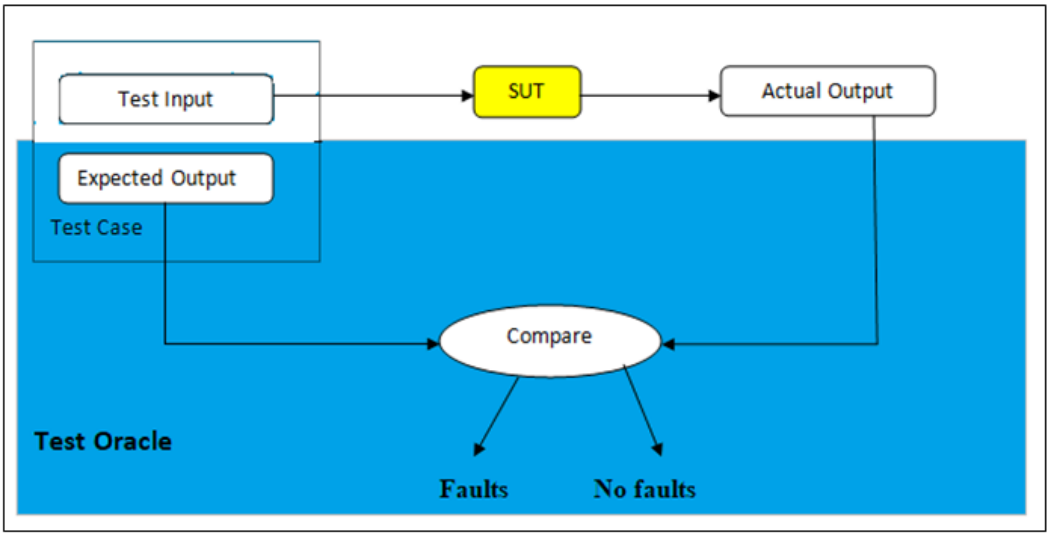

FIGURE 2.2: Test Oracle Problem.

In IPAs, there are different methods to alleviate the oracle problem. We have discussed these methods in the next section.

2.3 Methods to Alleviate Oracle Problem

Some of the methods to alleviate oracle problem are given below:

• According to Davis and Weyuker (1981), in pseudo-oracle, we can have several implementations working on the same issue and verifying one another. All the outputs of the implementations are compared, and if one of them differs, it may be a sign that the algorithm has flaws. But this method is time consuming as it is difficult to make a program of same nature and it is also quite probabilistic that different people can commit the same error [43]. Therefore, in literature this method is not recommended for the testing purpose.

• Partial oracle is another method to address the oracle problem. In this method, there may still be a small number of unique and straightforward input values for which the output values are known or can be easily determined in the absence of an oracle. The software being tested uses this group of special input values as a partial oracle. However, it has been demonstrated that the failure detection capability of this small range of special and simple input values is extremely limited [44].

• Metamorphic Testing (proposed by Chan et al.) is a useful strategy for solving the oracle problem. The testing method verifies all the outputs of the system rather than checking the individual outputs [45]. Metamorphic relations can be checked between source and follow up test cases to verify the property of system under test (SUT) [46].

The survey indicates that MT is an effective and efficient way to address the IPAs' oracle problem rather than using pseudo-oracle or partial oracle. As a result, we are using MT to solve the IPA's oracle problem.

2.4 Metamorphic Testing

Currently, the most effective method to handle test oracle problem in IPAs is Metamorphic testing (MT) [47]. MT is a software testing paradigm which identifies the properties of the SUT, called metamorphic relations, to either verify the

test results, or to generate new test cases (follow-up test cases) [48]. Instead of only constructing the follow-up test cases from source test cases, MT verifies the test results by checking the relationship between the outputs of source and follow-up test cases [49]. Figure 2.3 shows the process of metamorphic testing.

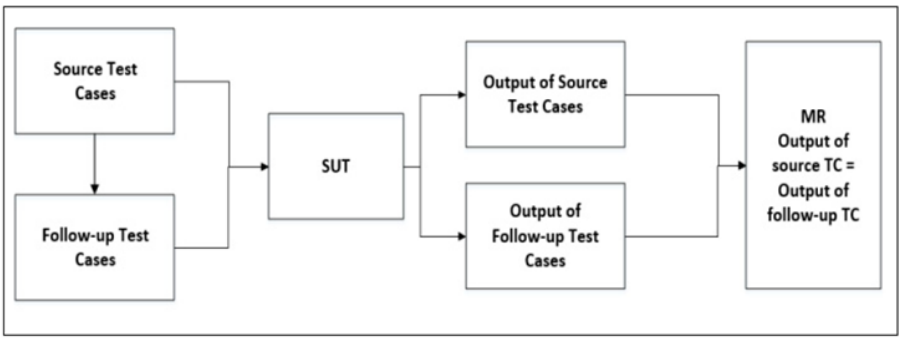

FIGURE 2.3: Process of Metamorphic Testing.

The steps in MT process are given below:

• Based on the domain knowledge about the proposed algorithm, one or multiple metamorphic relations can be identified.

• Source test cases are generated after MRs have been identified using some conventional test generation techniques, like random generation through random model or Boolean model, structural or program based test generation techniques, behavioral or specification based, symbolic evaluation method, combinatorial techniques and fault based test generation techniques etc,.

• The source test cases are then transformed by an MR into new test cases called follow-up test cases [50].

• Now run both the source and follow-up test cases to determine whether their results correspond to the change that the MR has projected.

• During the testing process, if there is an MR violation, then it shows that the SUT is faulty [51].

2.4.1 Example of Metamorphic Testing

The concept of MT can be explained through an example in which a sin function of a program can be computed. The metamorphic relation in this case is the

property of sin function given below:

$$\sin(x) = \sin(180 - x)$$

According to the above MR, x is the source test case for any arbitrary value e.g., 57.3. After the execution of sin function, suppose we got the output 0.8415. The verification of this output is not easy if we do not have the test oracle. In MT, follow up test cases can be generated which is (180-x) in case of above MR. The follow-up test case is given to SUT for execution. Suppose the output produced is 0.8402. The results of the source and the follow-up test cases can now be compared. As we see, both the outputs have different results and are not satisfying the relevant MR. It shows that the SUT is faulty [20] [52].

2.5 Metamorphic Testing In Image Processing Applications

As discussed earlier, IPAs are more difficult to test than traditional software because of their complexity and the visual semantics that are involved. In the field of IP, there are several algorithms such as sobel edge detection, canny edge detection, robert's algorithm to calculate the edges of the algorithm. Figure 2.4 shows the original image and the image after applying the edge detection algorithm. If we need to verify the accuracy of the output edges calculated by the edge detection algorithm we do not have a reference image for comparison. This is well known oracle problem in IPAs. As an example, consider Figure 2.4 where edges are calculated by edge detection algorithm. But, we do not have a reference image for comparison that whether the output is correct or incorrect.

MT is used in IPAs to alleviate the oracle problem. In IPAs, MT entails transforming the input images and determining whether the resultant image preserves specific anticipated characteristics or not. For example, if an IP algorithm is designed to calculate the edges of the images, MT helps ensure that the edge detection algorithm is invariant to certain transformations and consistently produces

accurate results, even if the input undergoes changes. This method (MT) uses expected relationships between input and output transformations to help validate the accuracy and dependability of IP algorithms.

FIGURE 2.4: (a) Input Image Before edge Detection (b) Input Image After Computing the Edges.

2.6 Metamorphic Relations

Metamorphic Relations (MRs) are the properties of the functionality of the software which involves multiple executions of the software [53]. These properties may include inequalities, periodicity properties, convergence properties, subsumption relationships, and other properties. The MR relates two or more inputs with their expected outputs after execution of each property of the target program [42]. Domain experts study the problem related to the target function and then formulate the MRs respectively. The primary function of MR is to produce and validate test results in the absence of a test oracle [54].

In the field of IP, there are different operations such as edge detection, dilation, erosion, image region growth, segmentation, enhancement, restoration, and Euclidean distance transform etc to perform specific operations. In existing literature, the authors have presented various MRs (both general and specific) which are derived from the properties of these operations. Some of the IP operations and their relative MRs are given in Table 2.1

TABLE 2.1: IP Operations and their Relative MRs

IP Operations	**MRs**
Edge Detection	MR_1 : Counter clock-wise rotation at 90 degree
	$C(E(Im)) = E(C(Im))$
	MR_2 : Reflection at the ordinate
	$M_x(E(Im)) = E(M_x(Im))$
	MR_3 : Reflection at abscissa
	$M_y(E(Im)) = E(M_y(Im))$
	MR_4 : Image Transpose
	$T(E(Im) = E(T(Im))$
Erosion and Dilation	R_1: Reflection at the ordinate
	$Ref_{ord}(Output(I)) = Output(Ref_{ord}(I)$
	R_2: Reflection at abscissa
	$Ref_{abs}(Output(I)) = Output(Ref_{abs}(I))$
	R_3: Duality
	$\delta_s(I) = \varepsilon_s(I^c)$
	$\varepsilon_s(I) = \delta_s(I^c)$
	R_4: Non Inverses
	$\delta_s(\varepsilon_s(I)) \neq I \neq \varepsilon_s(\delta_s(I))$
	R_5: Image objects size changes
	$Size_{obj}(\delta_s(I)) \geq Size(I) \& Pix_{list}(I) \subset Pix_{list}(\delta_s(I))$
	R_6: Number of objects in an image changes
	$Number_{obj}(\delta_s(I)) \leq Number_{obj}(I)$
	R_7: Commutative
	$\delta_s(I) = I \oplus S = S \oplus I = \delta_I(S)$
	$\varepsilon_s(I) = \varepsilon_I(S)$
	R_8: Translation Invariance
	$\delta_{s+x}(I) = \delta_s(I) + x$

TABLE 2.1: IP Operations and their Relative MRs

IP Operations	**MRs**
Image Region Growth	MR_1: $(r, r_f)\|(r(I, I') = (I' = Reftsp))) \Rightarrow (rf(I), f(I)) = (RG(r) = Reftsp(RG(I)))$
	MR_2: $\{(r, r_f)\|(r(I, I') = (I' = I * k, graythd' = graythd * k)) \Rightarrow (r_f(f(I), f(I')) = (RG(I) = RG(I')))\}$
	MR_3: $\{(r, r_f)\|(r(I, I') = (I' I_{seed1 \to seed2})) \Rightarrow$ (r$_f$($f(I)$, f(I')) = (RG (I') = RG (I)))$\}$(seed 2 $\in RG(I)$)
	MR_4: $\{(r, r_f)\|(r(I, I') = (I' I_{seed1 \to seed2})) \Rightarrow (r_f(f(I), f(I')) = (RG(I') \cap RG(I) = \phi))\}$ $(seed2 \in RG(I))$
Euclidean Distance Transform	R_1 : 90 degree Rotation (counter clock-wise) $C(D(A)) = D(C(A))$
	R_2: Reflection at the Ordinate $M_X(D(A)) = D(M_X(A))$
	R_3: Reflection at the Abscissa $M_y(D(A)) = D(M_y(A))$
	R_4: Image Transpose $T(D(A)) = D(T(A))$
	R_5: Image Enlargement Let$D(A) = (d_{i,j})$and$D(E(A)) = (e_{i,j})$.then $3d_{i,j} = e_{3i+2,3j+2}$for each$0 \leq i \leq nyand0 \leq j \leq n_x$
	R_6: Image Intersection $D(A \cap B) = min(D(A), D(B))$
	R_7: Image Union $D(A \cup B) \geq max(D(A), D(B))$

2.6.1 Generation of Source and Follow-up Test Cases Using MR

In IP, edge detection is a very meticulous process that serves as a chief tool for the detection of edges in the image with variations in its luminosity or incoherence [55]. The authors in [56] have presented four MRs for the edge detection operation. We have used one of the MR (counter clock-wise rotation at 90 degree) presented by Sim et al. and shows how source and follow-up test cases are generated using an MR. The MR is given below:

$$MR : C(E(Im)) = E(C(Im))$$

In above MR,

C(.) is the counter clock-wise rotation at 90 degree.
E is the edge detection operation and also the SUT of the above given MR.
Im is the image as well as the source test case.
C(Im) is the follow-up test case of the above given MR.
For input image Im, the output of edge detection followed by counter clock-wise rotation at 90 degree should be equal to the output of counter clock-wise rotation at 90 degree followed by edge detection.

2.6.2 Evaluation of Metamorphic Relations

The evaluation of MR is performed by comparing the outputs of source and follow-up test cases, generated after the execution of implementation under test. As each MR consists of more than one input, so, multiple executions are required to check the satisfaction of MR [57]. If the relation holds between the outputs of source and follow-up test cases then it show a bug free SUT else the SUT is faulty [58]. However, if the MR satisfies all the test cases, then it is too weak to find the violation [55].
Consider the same MR depicted in section 2.5.1 to show the evaluation process of

MR. Figure 2.5 shows the original and follow-up test case generated for the MR of edge detection i.e., counter clock-wise rotation at 90 degree.

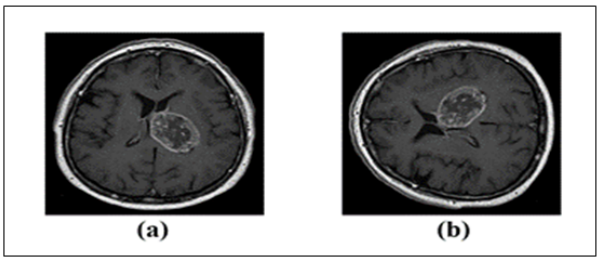

FIGURE 2.5: (a) Source Test Case; (b) Follow-up Test Case .

Both the test cases (source and follow-up) are applied to SUT as inputs. Here, edge detection is the SUT. So, after the execution of SUT, the outputs of both the test cases are observed. Figure 2.6 shows the output results of source and follow-up test cases.

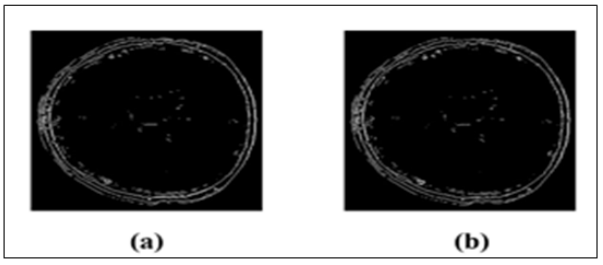

FIGURE 2.6: (a) Output of source test case E(Im); (b) output of follow-up test case E(C(Im)).

Afterwards, counter clock-wise rotation is applied to E(Im). Figure 2.7 shows the output of both the test cases after balancing the MR.

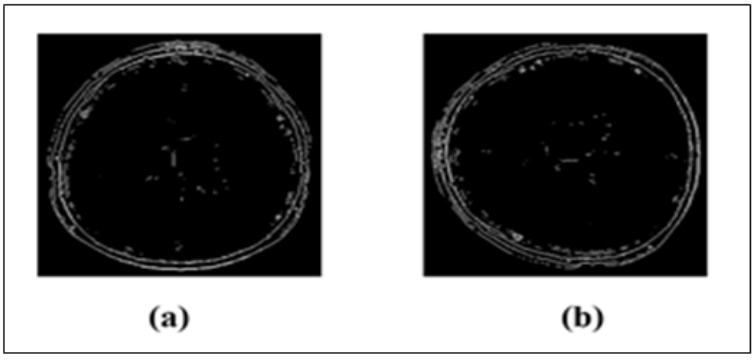

FIGURE 2.7: (a) C(E(Im); (b) E(C(Im)).

Now, the outputs of both the test cases are compared for MR violation. If we

obtain a full black image after pixel by pixel comparison, then it shows that the outputs of both the test cases are similar and there is no MR violation. However, if both the outputs of both the test cases are different then it shows an MR violation.

2.6.3 Fault Detection Rate of Metamorphic Relations

MT is dependent on the fault detection rate of the MR. The higher the fault detection rate, the higher the fault detection capability. Let a program P have a set of test cases T, and R be a metamorphic relation for P. Let t' denote follow-up test case of t w.r.t R, and P(t) denote the output of P on test case t. A test case t is said to satisfy R if metamorphic relation R holds between P(t) and P(t'). Metamorphic relation R is said to be satisfiable w.r.t. T if all test cases in T satisfy R, otherwise R is said to be violative w.r.t. T. The fault detection rate (FDR) of an MR with respect to a program P is the ratio of the size of the MR's set of violative source inputs to the size of the MR's set of source inputs [59]. If R is satisfiable for a given program P and a given test set T, then it means either R has 0 or low fault detection rate or there is no bug in P. If R is violative for a given program P and a given test set T, then it means R has high fault detection rate.

In existing literature, the FDR of MRs is calculated using mutation testing. Mutation testing is a software testing technique which is used to identify the faults in a program by changing the program [60]. It is a white box testing method (code based testing) where the aim is to inject artificial changes based on real faults to assess the quality of test suites in order to reveal faults [61]. It is a process that produces semantic program variants by changing the program's syntax. These program variants (faults) are called mutants. Each mutant contains a single fault. When a test case can tell the difference between the behaviour of the original program and the mutant program, we say the mutant has been killed [62]. The process of mutation testing is given in Figure 2.8. According to the above diagram, there are many steps that are involved in the mutation testing process but the three main steps of mutation testing process are as follows: [63]:

• Selection of mutation operators that is relevant to the faults.

• Generation of mutants

• Executing the original program and analysing each mutant produced using the test cases will help distinguish between different mutants.

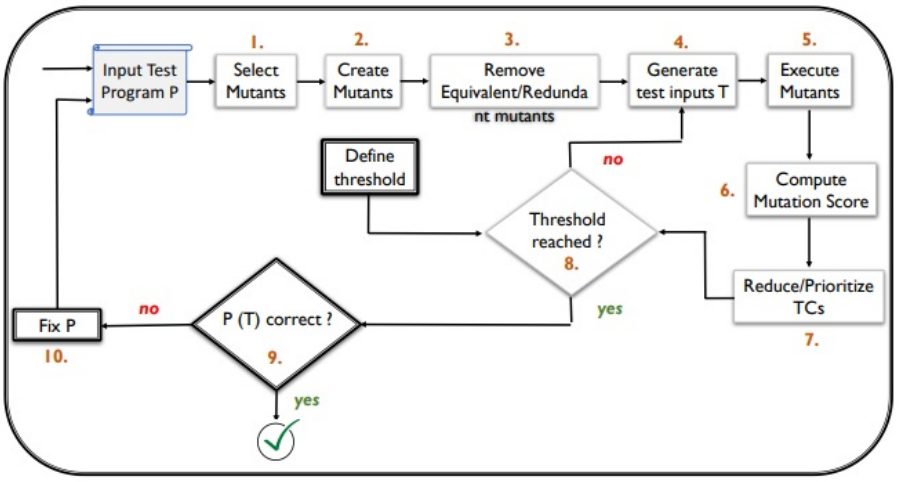

FIGURE 2.8: Process of Mutation Testing [63]

After the execution of test cases on the mutated programs, mutation score is computed. If the behaviour of the modified program (mutated) differs from the behaviour of the original program, then the mutant is killed; otherwise, it remains alive [64]. The mutant program is equivalent if it behaves in the same way as the original program [63]. The mutation score is calculated by excluding the equivalent mutants. The formula to calculate the mutation score is given in Equation 2.1.

$$\text{Mutation Score} = \frac{\text{Number of killed mutants} \times 100}{\text{Total no. of mutants}} \tag{2.1}$$

In MT, using mutation testing, the source and follow-up test cases are given to the original program for execution. The outputs of both the test cases are recorded for comparison. The same two test cases are given to the mutated program for execution. The outputs of these two test cases are also recorded for comparison. The mutant is said to be alive if the outputs from the original and mutated programs satisfy their respective MRs; otherwise, the mutant is killed. Afterwards, mutation score is calculated to check the FDR of each MR. If the mutation score is near to 1 then it shows that the MR is strong else the MR is weak enough to

find the violation.

The effectiveness of mutation testing technique is computed using the mutation score formula. If the mutation score is low then it shows that the mutation testing is not effective [65]. In mutation testing, mutation operator is the key component [66]. The process of creating mutants involve injecting faults through a variety of predefined rules called mutation operators that help to measure the adequacy of a test suite detecting these faults [67]. For example, an arithmetic mutant operator changes the arithmetic programming language operator from '+' to '-', '*', and '/' etc., [62]. The basic five mutation operators proposed by [68] are given in Table 2.2.

TABLE 2.2: Basic Five Mutation Operators Proposed by Offutt et al. [64]

Operator	Description	Mutation Operators
ABS	Absolute Value Insertion	{(e,0),(e,abs(e)), (e,-abs(e))}
AOR	Arithmetic Operator Replacement	{((a op b), a), ((a op b), b), (x, y) — x, y ∈{+, -, ×, /, %} ∧ x 6= y}
LCR	Logical Connector Replacement	{((a op b), a), ((a op b), b), ((a op b) , false), ((a op b), true), (x, y) — x, y ∈ {&, —, ∧, &&, ——} ∧ x 6= y}
ROR	Relational Operator Replacement	{((a op b), false), ((a op b), true), (x, y) — x,y ∈{>, >=, <=, ==, !=} ∧ x 6= y}
UOI	Unary Operator Insertion	{(cond, !cond), (v, -v), (v, v), (v, –v), (v, v–), (v, ++v), (v, v++)}

2.7 Summary

In this chapter we have covered the background knowledge of the basic concepts and terminologies used in our research. This chapter lists the concepts of software testing, oracle problem, some of the challenges are also discussed to alleviate the oracle problem, metamorphic testing, and evaluation of metamorphic testing.

The concepts of mutation testing is also discussed. These concepts are useful to understand the problem focused in this thesis.

Chapter 3

Literature Review

We have divided the papers selected for literature review into three categories. The first category of papers covers the topics where MRs are evaluated to improve the effectiveness of MT. The second category discusses the use of machine learning for output evaluation of images. The third and last category discusses the enhancements of MT.

3.1 Evaluation of Metamorphic Relations

In this category, the authors have selected different IP operators such as edge detection, image region growth, dilation and erosion, Euclidean distance transform and used their properties as MRs. The effectiveness of these MRs are checked through mutation testing.

3.1.1 Evaluating Effectiveness of MT on Edge Detection Programs

The authors in [56] have studied Sobel edge detection program in C programming language in order to examine the efficiency of MT. Here, MT is used to spot the bugs in edge detection programs. The authors have presented four MRs for edge

detection. All the four MRs are general and can be applicable to most of the IP operations. The metamorphic relations described in this paper are as follows:

$$MR_1 : \text{Counter clock-wise rotation at 90 degree}$$

$$C(E(Im)) = E(C(Im))$$

Where,
C: Counter clock-wise rotation at 90 degree
E: Edge detection
Im: Input image

$$MR_2 : \text{Reflection at the ordinate}$$

$$Mx(E(Im)) = E(Mx(Im))$$

Where,
M_x: Reflection at the ordinate

$$MR_3 : \text{Reflection at abscissa}$$

$$My(E(Im)) = E(My(Im))$$

Where,
M_y: Reflection at abscissa

$$MR_4 : \text{Image Transpose}$$

$$T(E(Im) = E(T(Im))$$

Where,
T: Image transpose

To conduct the experiments, collections of images are needed for the generation of test inputs. Unlike model generated images, camera captured images (real images)

from published image libraries are selected randomly. MT is used to evaluate the fault detection capability of MRs. Two types of faults are seeded into the program of Sobel edge detection. First fault is the Single operator fault in which a single fault is seeded at a time. In single operator faults two types of operators are used i.e., logical operators (AND, OR, NOT) and the relational operators ($<, >, \leq, \geq, =, \neq$). The stride implementation fault constitutes the second error. A failure in the stride implementation occurs when the image is processed up to the visible horizontal width rather than the "stride" width. Stride width is the image data array's real horizontal dimension. For efficiency, when an image is stored, the horizontal dimension will be padded until the next multiple of four. Fault detection effectiveness is measured against every MR using mutation testing. In experiments, thirty one (31) faulty programs are used where 30 programs are related to single operator fault and one is related to stride implementation fault. It is observed that relation MR_2 is the most effective MR in terms of fault detection having a mutation score of 77% trailed by MR_3 with a mutation score of 68%. The fault detection ratio of MR_1 and MR_4 is 45% each. It is also observed that stride implementation fault is the subtle fault and is detected only when certain features of input images are used.

3.1.2 Addreessing Test Oracle Problem in IPAs

the authors in [27] have discussed the oracle problem in IPAs and used the properties of IP operations as MRS. The authors have studied some specific and general properties of dilation and erosion operations. The effectiveness of each MR is analyzed through mutation testing. The author's approach is given below:

- Original test cases are selected from image libraries.
- After the test case selection, the properties of IP operations are used as MRs.
- Follow up test cases are generated through source test cases and MRs.
- The outputs of source and follow up test cases are calculated.
- Verification of MRs can be done after the output generation of source and follow up test cases.

•The SUT is considered faulty if there is any violation of MR. The process of testing can be repeated after debugging.

• If there is no violation of MR then the testing process can be tested against the stopping criteria.

• The testing process is considered complete, if the stopping criteria can be successfully met, otherwise, the testing objectives can be achieved by redefining the MRs.

The authors have presented eight metamorphic relations; two are generic and the remaining six are specific to the morphological operations (dilation and erosion). The MRs are given below:

R_1: Reflection at the ordinate

$$Ref_{ord}(Output(I)) = Output(Ref_{ord}(I)$$

Where,

I: Input image

Output: Image output

Ref_{ord}: Reflection at the ordinate

R_2: Reflection at the abscissa

$$Ref_{abs}(Output(I)) = Output(Ref_{abs}(I))$$

Where,

Ref_{abs}: Reflection at abscissa

R_3: Duality

$$\delta_s(I) = \varepsilon_s(I^c)\varepsilon_s(I) = \delta_s(I^c)$$

Where,

δ_s: Dilation with structuring element s

ε_s: Erosion with structuring elemnent s

c: Image complement

R_4: Non Inverses

$$\delta_s(\varepsilon_s(I)) \neq I \neq \varepsilon_s(\delta_s(I))$$

R_5: Image objects size changes

$$Size_{obj}(\delta_s(I)) \geq Size(I) \& Pix_{list}(I) \subset Pix_{list}(\delta_s(I))$$

Where,

$Size_{obj}$: Size of object

Pix_{list}: List of pixels

R_6: Number of objects in an image changes

$$Number_{obj}(\delta_s(I)) \leq Number_{obj}(I)$$

Where,

$Number_{obj}$: Number of objects

R_7: Commutative

$$\delta_s(I) = I \oplus S = S \oplus I = \delta_I(S)\varepsilon_s(I) = \varepsilon_I(S)$$

Where,

$\oplus$: Dilation operation

S: Structuring element

R_8: Translation Invariance

$$\delta_{s+x}(I) = \delta_s(I) + x$$

Where,

x: 2D factor

To conduct the experiments, input images are selected randomly. Single operator faults are seeded into the Mex C code deliberately. The authors have used only

relational operators and have generated 33 mutants against each MR. The number of mutants that have been killed is indicated by the mutation score. The mutant is said to be killed if an MR is able to detect the bug. It is observed that the mutation score of R_3 i.e. (duality) is highest i.e. 97%. The second-highest mutation score is 73 percent for Relation R_7 (commutative property). On the other hand, R_1 (reflection at the ordinate) has the lowest (15 percent) mutation score. One interesting aspect is that, the mutants can be killed by only specific images while other images failed to kill the mutants. Therefore, it is concluded that for bug identification, instead of using general input images such as Lena, specific images are needed.

3.1.3 MT of Image Region Growth Programs in IPAs

The authors in [46] have worked on image region growth program and applied MT on image region growth program. Mutation testing is used to find the effectiveness of MRs. The MRs presented in this paper are given below:

MRs of Geometric Properties:

MR_1 :

$$(r, r_f)|(r(I, I') = (I' = Reftsp))) \Rightarrow$$
$$(rf(I), f(I)) = (RG(r) = Reftsp(RG(I)))$$

Where,

I: Input image

I': Image transpose

(r, r_f): Starting seed points

MRs of Numerical Property:

MR_2 :

$$\{(r, r_f)|(r(I, I') = (I' = I * k, graythd' = graythd * k)) \Rightarrow$$
$$(r_f(f(I), f(I')) = (RG(I) = RG(I')))\}$$

Where,

graythd: threshold value

k: adjacent pixels

MRs of Algorithmic Properties

MR_3 :

$$\{(r, r_f)|(r(I, I') = (I' = I_{seed1 \to seed2})) \Rightarrow$$
$$(r_f(f(I), f(I')) = (RG(I') = RG(I)))\}$$
$$(seed2 \in RG(I))$$

MR_4 :

$$\{(r, r_f)|(r(I, I') = (I' = I_{seed1 \to seed2})) \Rightarrow$$
$$(r_f(f(I), f(I')) = (RG(I') \cap RG(I) = \phi))\}$$
$$(seed2 \in RG(I))$$

In this paper, MT is applied to test the aerospace IP software. A total of five test cases are generated using segmental symbolic evaluation method. The original program is implemented in C language and is executed sequentially with three mutant programs. The program is said to be faulty if MR violation can be seen after the validation of output relations. The authors have drawn some conclusions after performing the experiment: first, if a certain MR is used to test the mutant program then the same MR and test case can be used to test the original program as well. In this way the results satisfy the corresponding MR while the correctness of the original program can also be confirmed. Second, The experiment also confirms the effectiveness of the MT technology in resolving the oracle issue in the region growth program. Finally, each metamorphic relation's error detecting ability is different from the other relation.

3.1.4 Models for Random Input Generation

The authors in [69] have proposed an idea to randomly generate the test data (images) using two models i.e. random model and Boolean model. Euclidean

An IP operation called Euclidean distance transform turns a binary image into a greyscale image. The implementation of Euclidean distance transform is used to compare the results of both the models using mutation testing. To evaluate the test results, the Euclidean distance transform also identifies the MRs, necessary properties, and special values in addition to the random test data. The above criteria's are also compared through mutation testing. The MR proposed in this paper is given below:

R_1: 90 degree Rotation (counter clock-wise)

$$C(D(A)) = D(C(A))$$

Where,
A: Input image
C: Counter clockwise rotation at 90 degree
D: Euclidean distance transform

R_2: Reflection at the Ordinate

$$M_X(D(A)) = D(M_X(A))$$

Where,
M_X: Reflection at ordinate

R_3 Reflection at Abscissa

$$M_y(D(A)) = D(M_y(A))$$

Where,
M_y: Reflection at abscissa

R_4: Image Transpose

$$T(D(A)) = D(T(A))$$

Where,

T: Image transpose

R_5: Image Enlargement

$$\text{Let} D(A) = (d_{i,j}) \text{and} D(E(A)) = (e_{i,j}).\text{then}$$
$$3d_{i,j} = e_{3i+2,3j+2} \text{for each} 0 \leq i \leq nyand 0 \leq j \leq n_x$$

Where,

E: Image enlargement

R_6: Image Intersection $D(A \cap B) = min(D(A), D(B))$

Where,

B: Input image

min: minimum distance

R_7: Image Union

$$D(A \cup B) \geq max(D(A), D(B))$$

Where,

Max: Maximum distance

The implementation is done in java whereas mujava tool is used for mutation testing. The mutation operators used in mutation testing are: AOR and ROR, variables or constants replacement. Instead of only comparing the test data generating models using mutation testing, the size of the image as well as the proportion of the pixels in the images are also analyzed. The image sizes used in this paper are: 10 x 10, 20 x 20, 30 x 30, 50 x 50 and 60 x 60. The percentage of black pixels in the chosen Boolean model is 10%, 20%, 50%, 80% and 90% respectively. Whereas additionally 99% in random model. The relations R_1 to R_7, along with necessary properties is examined through random model with size of image 30 x

30 and 99% of black pixel proportion. The MRs in Boolean model is analyzed with image size 30 x 30 and 90% black pixel proportion respectively. The study's findings indicate that the percentage of black pixels has the greatest impact on the killing rate of mutants. It is only necessary to choose an image size that is sufficiently large, or at least 30. The number of mutants killed increases with the percentage of black pixels. Results show that the best metamorphic relations are those that exchange coordinates, specifically R_1 and R_4. Only when used in conjunction with the relation R_1, S_1 kills 1218 mutants. So, The most effective combination for distance transforms appears to be R_1 and S_1.

3.1.5 Testing Imaging Software Automatically

In this paper [29], the authors have presented an approach that test the imaging software automatically. The authors have proposed several models such as Boolean model, grayscale Boolean model, random model and grayscale random model to generate the test data randomly. Implementation of Euclidean distance transform, labeling of connected components, and Lipschitz are used to evaluate the testing approach. The implementation is done in java whereas mujava tool is used for mutation testing. The mutation operators used in mutation testing are: AOR or ROR operators, variables and constants replacement. MRs are used to evaluate the test results. Mutation analysis is used to compare oracle solutions and test data generation models. The authors have used the same metamorphic relations as described in their previous work i.e. [69].

R_1: 90 degree Rotation (counter clock-wise)

$$R(O(A)) = O(R(A))$$

Where,

A: Input image

R: Rotation at 90 degree

O: operators

R_2: Reflection at the Ordinate

$$M_X(O(A)) = O(M_X(A))$$

M_X: Reflection at ordinate

R_3 Reflection at Abscissa

$$M_y(O(A)) = O(M_y(A))$$

M_y: Reflection at abscissa

R_4: Image Transpose

$$(O(A))T = O(AT)$$

Where,
T: Image transpose The four MRs listed above are universal and apply to various IP operators O(.). the MRs specific to Euclidean distance transform is given below:

R_5: Image Enlargement

$$LetD(A) = (di, j)andD(E(A)) = (ei, j).Then$$
$$3di, j = e3i + 2, 3j + 2foreach0 \leq i < nyand0 \leq j < nx$$

R_6: Image Intersection

$$D(A \cap B) = min(D(A), D(B))$$

R_7: Image Union

$$D(A \cup B) \geq max(D(A), D(B))$$

These metamorphic relations are used for the generation of follow up test cases as well for the evaluation of test results. The authors presume some indications about

the test generation models that both Boolean and random models are equally effective to generate the test images. Since, it is easy and simple to generate random binary images so they should be preferred. Furthermore, it has been found that random binary images are more sensitive to the percentage of black pixels than the Boolean model. In greyscale models, Boolean grayscale model is preferable over random grayscale model. The MRs are used for the generation of follow up test cases as well for the evaluation of test results. Relation R_1 and R_4 is considered the best for killing maximum number of mutants. Relation R_1 has killed 1211 mutants in EuclidDT, 459 in Connected C8, and 825 in Lipschitz implementation. While combining the MRs, only in case of EuclidDT, combining relation R_1 and R_8 gives better results than R_1 otherwise all the other combined relations are not effective than R_1. S_1 is more effective than R_1 by killing 1218 mutants. All of the mutants are killed when R_1 and S_1 are combined.

3.1.6 Evaluation of Partial Oracles using MT

In this paper [42], the authors have presented an approach to assess the testing strategies using mutation testing which determine the test inputs and evaluate the partial oracles. In this approach, MRs are used as partial oracles for graphic (imaging) software. Random image generation for imaging software is quite non trivial task. Input values in imaging software are complex because of their different sizes and the color depth. So, the authors have divided the assessment of testing strategy into two parts. First, determine the input values carefully. For the selection of suitable input values, random greyscale model is protracted for the creation of color images. Second, evaluate the quality of partial oracle. The complete process is given below:

- Create mutants using appropriate implementation under test (IUT)
- Choose any model for generating input values.
- Using the generated mutants and the original implementation as the perfect oracle, determine the input domains that are the most effective. Apply the input values that have been determined to be appropriate to all mutants, then check the

results using the partial oracle that will be assessed.

The authors have presented four MRs in this paper. Metamorphic relations used in this paper are given below:

R_1: Add a constant value to each component of the color values.

R_2: Using a constant, multiply each component of the colour values.

R_3: Take transpose of each color component.

R_4: Image enlargement with zero padding.

In this paper, authors have used two image sizes i.e., 128 x 10 and 10 x 128 for input data. The implementation should be done in object oriented language preferably in java in order to apply class based mutants. This implementation generates 463 conventional/traditional mutants and 120 class-based mutants. It is observed from the results that class-based mutants should be preferred over traditional mutants because they represent structural faults. It is preferable to use object oriented mutants along with traditional ones in order to improve the reliability of testing strategies while using partial oracles. Furthermore, for making the best testing strategy, earlier determination of input values are also very important. Results show that the effectiveness of MRs is difefrent with regard to the employed mutants. For traditional mutants, the relation R2 is rather ineffective, but it works fine for class-based mutants. However, the relation R3 kills all traditional mutants while being obviously insufficient for class-based mutants.

3.1.7 Evaluation of Partial Oracles

An approach in this paper [70], uses a partial oracle to automatically test a jpeg2000 encoder. Jepeg2000 encoder is used as an example for a system that consists of several integrated components. The paper focuses on the possible improvements regarding efficiency and effectiveness of partial oracles. Mutation testing is used to evaluate the effectiveness of partial oracle. A mutant must meet three requirements in order to be killed. (1) The implemented code which is mutated should be grasped and executed (2) the state of program should be

changed by the mutation and (3) the output must show the propagated change. The workflow of the system after taking a colored image (uncompressed) as its input (through random model) is given below:

• RGB (red, green, blue) are the three colour components that make up the colour image.
• The color values of color components are shifted.
• Red, green and blue color components are transformed into YCbCr components.
• Two dimensional wavelet transformations are used to decompose the color components.

In this paper, partial oracle is used to avoid the oracle problem. The oracles which are chosen for this system are given below:

R_1: Add a constant value to each component of the color values.
R_2: Using a constant, multiply each component of the colour values.
R_3: Take transpose of each color component.
R_4: Image enlargement with zero padding.
R_5: reverse the color values of the image

In this paper, the authors have used partial oracle to investigate the efficiency of a complete system as well as the fault revealing capability in each transformation. It is observed that traditional mutants throw more exceptions (due to illegal array indexes) at run time than the mutants generated by class. The approximate ratio is 40%: 20%. Among all the MRs, the relation R_3 is regarded as the best with a 9 mutation score (in case of traditional mutants). The above mentioned oracle is not good or appropriate for killing class-based mutants with a mutation score of about 58%. Hence, combining the partial oracles will produce better results. Combining partial oracle gives effective results for unit testing as well as for integration testing.

3.1.8 Framework for Evaluating MT

The authors in this paper [71] have proposed a framework to evaluate the effectiveness of MT. They have developed an iterative method to check the adequacy

of MRs which is controlled by MT adequacy evaluation. The effectiveness of MT is checked through code coverage criterion (function coverage, branch coverage, and definition-usage coverage), mutation analysis and testing MRs using mutation tests. The proposed framework is explained through an IP program that is used to create a 3D model of a biological cell. The effectiveness of proposed scheme is demonstrated through a case study that tests an intricate Monte Carlo program. The MRs used in this research is given below:

MR_1 Inclusive: Each cluster of mitochondria contains one or more mitochondria. The program draws an outline in each section of mitochondrial cluster and then constructs a three-dimensional biological cell by joining the lines of similar mitochondrial cluster in the neighboring sections. By adding a new mitochondrial cluster in an original image section, the number of clusters should be increased by 1. Therefore, an increase in the volume of mitochondria is expected.
MR_2 Exclusive: In the second MR, remove a cluster of mitochondria from the original image section. As discussed earlier, multiple sections contain the same cluster therefore; the cluster should be removed from all the sections of the image. Now the processed or reconstructed image does not contain the removed cluster. Hence, a 1 cluster decreased should be seen. Because of the above decrease in the cluster, the expected volume should also be decreased.
MR_3 Multiplicative: The same cluster of mitochondria is appeared in adjacent image section; therefore, every section of the image is increased by the same percentage. The mitochondrial clusters should remain same in the reconstructed image but the volume may be increased.
MR_4 Additive: Add a mitochondrial cluster into an image section. With this addition, the original cluster of mitochondria extends in multiple image sections. The original cluster spreads alongside the artificial one because it is connected to the new cluster. The number of clusters in in the processed image remains constant but the volume of mitochondria may increase.
MR_5 Permutative: Change the positions of two clusters simultaneously in all the sections of the image. In the reconstructed image, the cluster's amount remains constant whereas the volume of the cluster also remains constant.

MR_6 Invertive: Select a cluster of mitochondria that exists in multiple adjacent sections. Then, rotate all the images to 180 degree altogether. In the reconstructed image, the cluster's amount remains constant whereas the volume of the cluster also remains constant.
MR_7 Lengths: In the same image section, mitochondria with a small cluster may appear, but a larger mitochondrial cluster may appear in 1 or more neighboring sections of the image. By adding a new mitochondria in a single image section then the processed image should contain the newly added mitochondria and it is anticipated that the mitochondria's volume will increase. On the other hand, it is significant to remember that if the same mitochondrion is added to multiple image sections then the reconstructed image is processed on the basis of multiple sections of the image.
MR_8 Shapes: The cluster gains additional mitochondria with various shapes. The processed image should contain these new constructed mitochondria's as expected.
MR_9 Locations: Add new mitochondria in different locations i.e. near nuclear or cell membrane. The processed image should recognize these new constructed mitochondria's as expected.

Test adequacy evaluation is required in MT in order to produce suitable tests for evaluation purposes. If sufficient test cases are not generated through existing MRs then there is a need to develop more MRs, so that new test cases can be developed. The steps involved in test adequacy evaluation are given below:

• Generation of initial tests and MRs: A set of MRs are identified based on the knowledge of domain experts. Source test cases are generated using test generation technique such as random generation, category based, combinatorial technique etc. source test cases can also be generated for each MR accordingly.
• Evaluation: When the SUT passes all the tests, then the quality of MR is checked by the tests created for mutations of MRs. Mutation testing and the coverage criterion are used to verify the test's effectiveness. A test should kill or weakly kill a mutant.
• MRs Refinement: If the test case does not fulfill the coverage criterion against the expected threshold value, it means new MRs is required. The MR is very weak

if it can satisfy a mutation test. Therefore, MRs should be refined, or new MRs should be generated that are sufficiently strong to find the violation.

The proposed framework is validated by testing scientific software system i.e. an IP program implemented in Matlab and a Monte Carlo simulation program . An iterative test adequacy evaluation is done for the refinement of test cases and MRs. The outcomes of MT on the Monte Carlo program show that test coverage results can be used to evaluate the calibre of tests and MRs, as well as a guide for choosing suitable MRs and developing tests. If more complex test coverage criteria, like state transition coverage or path coverage, are taken into consideration, the results of the coverage analysis may be more useful in evaluating the quality of MRs and their tests. It is observed that the five MRs selected for the Monte Carlo program testing are found to be effective after being tested with mutation tests. Additionally, it has been found that the suggested approach is helpful for testing other scientific software. We have summarized some of the important parameters used in the papers. The summary of this section is given in Table 3.2

TABLE 3.1: Summary of Section 3.1

Ref Papers	SUT	Image Generation Method	Testing Method	Mutation operators	Prog. Lang
2013 [56]	Sobel Edge Detection Program	Randomly selected	Mutation Testing	LOR, ROR	C
2015 [27]	Dilation, Erosion Program	Randomly Selected	Mutation Testing	ROR	Mex C

TABLE 3.1: Summary of Section 3.1

Ref Papers	SUT	Image Generation Method	Testing Method	Mutation operators	Prog. Lang
2018 [46]	Image Region Growth program	Segmental symbolic evaluation method	Mutation Testing	Milu tool	C
2006 [69]	Euclidean distance transform	Random binary model & Boolean model	Mutation Testing	AOR, ROR, replacement of variables, constants	Java
2007 [29]	Euclidean distance transform, lipschitz	Random model, Boolean model,	Mutation Testing	AOR, ROR, replacement of variables, constants	Java
2009 [42]	Program with image processing application	Random binary model	Mutation Testing	Traditional and class based operators	Java
2011 [70]	Program with image processing application	Random binary model	Mutation Testing	Traditional and class based operators	Java

TABLE 3.2: Summary of Section 3.1

Ref Papers	SUT	Image Generation Method	Testing Method	Mutation operators	Prog. Lang
2017 [56]	Monte Carlo Program, 3D structure reconstruct program	Random, Category based Selection	Mutation Testing, Structural Testing	AOR, COR, Constant addition, CRP, SDL	Mtlab

3.2 Metamorphic Testing and Machine Learning

In this category, the authors have automated the test oracle using a classifier in order to avoid the manual effort. For this purpose, several correct and incorrect images are required for training and then classify the correct and incorrect output images correctly. In some cases the correct output images are given to MT for further checking and in some cases the results of classifiers are compared with MT and statistical test oracle in terms of classification error.

3.2.1 Mechanism to Automate Test Oracle using SVM

In this paper [72], the authors have proposed a framework that automates the test oracle by using support vector machine (SVM). Some correct and incorrect output images are required along with their labels for training the model that is responsible for the classification of valid and invalid output images. The steps involved in this scheme are given below:

- Training Data Availability: SVM training requires output images and their

associated labels. Here, ground truth (GT) images are used to train support vector machine.

• Training: The main step of this scheme is to train data that involves data pre-processing to make it a valid input for the machine learning algorithms. In IPAs, different types of features are used to interpret the images i.e. Hough features, binary features, statistical features, wavelet-based features, etc. Careful selection of features is required in order to represent a certain class.

• Testing: Using some of the feature extraction technique as discussed in training phase, features are extracted from the output images. The class for this feature vector is determined that either the image is passed or failed. A comparative study is made by the authors to compare their scheme with the MT oracle and the statistical oracle (contains parameters like mean and standard deviation of the images) for morphological operators such as dilation and erosion. 35 versions of dilation program are created in C language by seeding the errors intentionally one at a time.

For the evaluation purpose, the output images are generated from the generated versions of image dilation program. It is concluded that some of the output images give correct results while the other versions give incorrect results. The output images are categorized into four categories i.e. valid output image, little variation from valid output images, no change with respect to the input image and fully invalid image. Two labels are used for the output images i.e. passed and failed. 50% of the images are used to train SVM. Different features (wavelet features, binary features, hough features, statistical features) of dilated images are used to analyze their effectiveness.

Experimental result shows that the wavelet-based features are the best with 2.69% classification error. As discussed earlier that the results of proposed scheme are compared with metamorphic test oracle and statistical oracle. 17% classification error is observed for statistical oracle. MR5 presented in [27] is used in this paper. This MR is specific to image dilation operator and produced 11% classification error. Therefore, it is concluded that SVM produced better results in term of lowest classification error than statistical oracle as well as metamorphic test oracle.

3.2.2 Framework of Automatic Testing of IPAs

In this paper [30], the authors have proposed an automatic testing process for IPAs. The framework consists of: integration of test image generation, execution of test cases, measures the effectiveness of test cases, stopping criteria and evaluation of test output. The authors have checked their approach through experiments and come to the conclusion that the proposed framework helps to reduce the manual work and gives effective output results through automatic input generation. The steps of proposed framework are given below:

(1) At the start, numbers of test cases are generated using symbolic evaluation method. (2) Source code is analyzed statically to specify the interested code parts such as branches, functions, loops, conditions etc. The interested code parts of IUT are appended by introducing analysis code to generate instrumented code version IUT'. When IUT' is executed, the analysis code identifies the executed code areas during a test run. During this process, if any unexpected program behavior is observed then it is considered as bug. (3) The correctness of output images are evaluated using MT and machine learning. The applications where outputs of different algorithms are available then we use SVM based classifier else test oracle is generated by MT.

The above three steps are implemented using Matlab. A mechanism is proposed where binary classifier is generated using SVM. Output images along with their labels are required to train the classifier. Here, ground truth (GT) images are used for training purpose. Features are extracted from images produced by IUT and then sent to the classifier. The classifier refers the feature vector to its closest class. Accuracy of classifier is checked through its classification error. In MT based test oracle, general property of image smoothing algorithms is used as MR to evaluate the correctness of output images. Smoothing property states that if the images contains noise then after applying the smoothing operation the variance in the output images decreases. The variance of a given image is given below:

$$\sigma = \frac{\sum (I_{rc} - I')^2}{(r*c)-1}$$

Where, $I^{'}$ is the mean of the pixel values and I_{rc} is the value of current pixel. To demonstrate the machine learning based test oracle, the authors have used implementation of twenty different edge detection algorithms. Results show that canny edge detection produces accurate result as compare to other edge detection algorithms. The framework integrates a mechanism to evaluate the correctness of output images in the absence of test oracle.

3.2.3 Identification of Failures in Mesh Simplification Programs using MT

In this paper [73], the authors have suggested a testing method that combines the pattern classification technique and MT. Mesh simplification is an algorithm that transforms a polygon into another with fewer faces, edges and vertices. During testing, the output of graphical images faces test oracle problem. C4.5 classifier is used to label the test cases as passed and failed. The outputs from the passed test also include failures because the classifier is statistical in nature. If the input data cannot reveal failures by statistical classifier then the test cases along with their test outputs can be pipelined to an analytical MT component for additional testing. Three MRs that are proposed are: (1) After each iteration, verification is done to check the size of bounding box. (2) In the input file (.PLY), change (reverse) the order of polygon vertexes. (3) The rendered image can be changed by changing the input values.

Four algorithms of mesh simplification (implemented in Java) are used as subject program. The trained images for the passed class are obtained by executing a set of 44 3D polygonal models in each reference system. A subset is selected randomly as source test cases. Similarly, the trained images for the failed class are obtained by generating the mutants from the reference system (total 3,060 mutants are generated).The authors have generated follow-up test cases and then check to see if failures are revealed using the implementation to exercise MT. The implementation results show that 29.4% failure causing test cases of Melax and 34.1% for Quadric are detected. The summary of section 3.2 is given in Table 3.3

TABLE 3.3: Summary of Section 3.2

Ref Papers	SUT	Image Generation Method	Testing Method	Programing Language	Classifier
2015 [72]	Dilation Program	Randomly selected	Mutation Testing	C	SVM
2016 [30]	Dilation & Erosion Program	Segmental symbolic evaluation method	Mutation Testing	Matlab	SVM
2007 [73]	Mesh & simplificati-on Program of image rendering	Randomly selected	Mutation Testing	Java	C4.5

3.3 Enhancements of Metamorphic Testing

In this category, the authors have used mutation testing as well as structural testing (branch coverage, path coverage, def-use coverage) to make MT more effective.

3.3.1 Self Checked MT Approach

A method of self-checked testing has been suggested [74] for the detection of subtle faults in the implementation. To guarantee the quality of testing, the suggested strategy incorporates structural testing into MT. By evaluating a cellular IP program, the efficacy of MT is investigated. The program is implemented in Fortran 90 language which is later verified by test coverage criterion. The steps involved in this approach are given below:

Defining MRs:

The first step is to identify the MRs and then MT is applied on the SUT for the verification of the relations. The MRs used in this paper are given below:

• R_1: Add mitochondria to the processed image. By adding new mitochondria,

the expected output result shows new mitochondria along with the original one. So, it is anticipated that the mitochondria's volume will increase.

• R_2: Add mitochondria in adjacent images. By adding new mitochondria in adjacent images, the expected output result shows new mitochondria along with the original one. So, it is anticipated that the mitochondria's volume will increase.

• R_3: Different shapes of mitochondria are added into the images.The mitochondrial structure should be constructed as anticipated without altering the structure of the original mitochondria by the addition of new mitochondria with various shapes. So, it is anticipated that the mitochondria's volume will increase.

• R_4: Add mitochondria to different locations of the images. By adding new mitochondria in different locations of the images, it is anticipated that the mitochondria's volume will increase.

• R_5: Remove some of the mitochondria from the images. After removing the mitochondria from the processed image, those mitochondria will not appear in the output. So, it is anticipated that the mitochondria's volume will decrease.

Generate Data for MT:

After the identification of MRs, source test cases are generated. Initially, 36 images of B16 cell are chosen as test inputs.

Selection of Coverage Criteria:

Statement coverage, function coverage, and definition-use coverage criteria is used to check the coverage of the code.

Code Instrumentation:

A code line is instrumented manually in SUT to test the coverage criterion. For example, each statement is followed by a checking statement to check the statement coverage. A checking statement is added right before the function's first statement to verify the function coverage. In def-use coverage, the statement of definition is followed by a checking statement, and another checking statement is added after each paired use statement.

An un-testable IP program is used to recreate or rebuild a 3D structure of a biological cell. The processing of each image results in the identification of the nucleus, cytoplasm, and mitochondria. The volume of a cell, its nucleus, cytoplasm, and

mitochondria is calculated using a 3D structure file. The reconstructed image is compared with the original image through pattern recognition component. The three-coverage criterion discussed earlier, are used to check the coverage of the code. 100% coverage criterion is observed against each test input. Furthermore, the effectiveness of proposed approach is validated by adding mutants into the source code. It has been noted that the coverage information kills the mutants as soon as the coverage time of each criterion is taken into account for each test input.

3.3.2 Application of MT for Testing Scientific Software

In this ppaer [75], the authors have presented a method for MR development and refinement. They have used discrete dipole approximation program (ADDA) program implemented in Fortran and C++ to assess MT's effectiveness. Due to the absence of test oracle, testing of ADDA is a tedious task. Testing of ADDA can be done using some special inputs like sphere scatterer. As output, ADDA contains thousands of datum items. On the inputs and outputs of ADDA, MR identification is very challenging. So, the images that are produced from the ADDA output are used as MRs.

MR_1: The first MR describes the relation between the sphere size and the textual pattern of an image (diffraction image) that is generated from the output of ADDA. Image pattern changes by changing the sphere size. Similarly, the lines of the pattern become slimmer whenever the sphere's size increases.

MR_2: As discussed earlier, there is a relation between sphere size and the diffraction image pattern. Therefore, if the shape of the scatterer is regular i.e. sphere or ellipsoid, then an association occur between the shape and the pattern. On the other hand, if the shape of the scatterer is irregular then the pattern of diffraction image is affected by changing the shape of the scatterer. The relationship between the sphere's shape and the textural pattern of the diffraction image is still unknown.

MR_3: The third MR explains how the scatterer's orientation and the diffraction

image's textural pattern relate to one another. For a sphere (at any orientation), the pattern of diffraction image should remain same. According to this MR, when changing the orientation, the diffraction image pattern should also change.

MR_4: This MR states the association between the scatter's refractive index and the images pattern. The MR describes the relationship between the scatterer's refractive index and the textural pattern of the diffraction image. The values of refractive index have diverse patterns. The relation exists between the homogeneous and heterogeneous scatterer. For all diploes, the scatterer which is homogeneous has same refractive index while the heterogeneous scatterer has not similar values for refractive index.

MR_5: The final MR discusses the relationship between the diffraction image pattern and the irregular morphology of heterogeneous scatterers. By changing the shape, orientation and size of cell, the image textual pattern also changes.

In this paper, statement coverage is used to check the effectiveness of tests. The effectiveness of MT is validated through mutation testing. The authors have applied mutation testing to only one module of ADDA program. The mutants are manually instrumented into the program rather than creating mutants with a tool. Total 20 mutants are instrumented in ADDA program. Out of 20, 17 mutants are killed due to exception handling or by crashing a program. The remaining 3 mutants are killed through MRs. It is observed that the MRs defined in this program is weak because the test output relation is unknown. Therefore, more MRs is required to adequately test the ADDA program. the summary of this section is given in Table 3.4.

3.4 Research Gaps

After the review of literature, we have identified the following gaps:

1. In existing literature, numerous methods have been used to generate the test cases such as random binary model, grayscale random model, Boolean model, and grayscale Boolean model. The method of random generation of test cases

TABLE 3.4: Summary of Section 3.3

Ref Papers	SUT	Image Generation Method	Testing Method	Mutation Operators	Programing language
2010 [74]	Cellular IP program	Randomly selected	Mutation Testing & Structural testing	AOR, SDL	Fortran 90
2016 [75]	Discrete Diploe Approxima-tion Program	Random selected	Mutation Testing & Structural testing	ABS, ROR	Fortran & C++

is widely used because of its perceived un-biased nature. However, generating the random test data could lead to unfair distribution of parametric values. Also, the sample population is not comprehensive. Therefore, sample selected is also not true representation of sample population. In literature, the image libraries, from where the test cases are selected, have images with same attribute values. For example, in one library all the images are of same resolution i.e., 96dpi. This unfair distribution, being non-comprehensive, may provide inaccurate results of the testing process. So, there should be a systematic way to generate the test cases to ensure completeness of all properties with equal chances to be selected in the sample for evaluation.

2. In existing techniques, MR evaluation is done through mutation testing using mutation operators. This evaluation is not comprehensive as only a few selected mutation operators are used to measure the FDR of MRs. The resultant mutants generated through these fewer mutation operators are also quite low which makes the comprehensive testing impossible.

3. In existing literature, no work has been done to gauge the effectiveness of mutation operators to find out which operator is more valuable to generate maximum number of mutants and through which operator maximum number of mutants can be killed.

4. In literature, in the field of IP, not enough MRs have been proposed for comprehensive testing. We need more relevant MRs for further studies.

3.5 Summary

This chapter discusses the evaluation methods of MRs to improve the effectiveness of MT. The MRs related to IP operations are discussed in details. In the evaluation methods, generally, the source test cases are generated randomly and for the evaluation of MRs, mutation testing is used. Moreover, integration of MT with machine learning algorithms are also discussed where machine learning is used for the output evaluation of images.

Chapter 4

A Framework for Evaluation of Metamorphic Relations

In this chapter, we have discussed our proposed framework to evaluate the MRs of IP operations such as edge detection an dilation and erosion operations. In proposed framework, source test cases are selected through black box testing (strong equivalence class testing) and white box testing (code coverage) techniques. For the evaluation of MRs, mutation testing is used by seeding mutants into the program. The FDR of MRs is calculated through mutation score. In the end, new MRs are constructed by composing the MRs.

4.1 Proposed Framework for MR Evaluation

The steps of proposed framework are given below whereas the flowchart of proposed framework is shown in Figure 4.1.

1. Generation of Source Test cases
2. Test Case Adequacy through Equivalence Class Testing and Code Coverage
3. Generation of Follow-up Test Cases

4. Evaluation of Metamorphic Relations

5. Composition of Metamorphic Relations

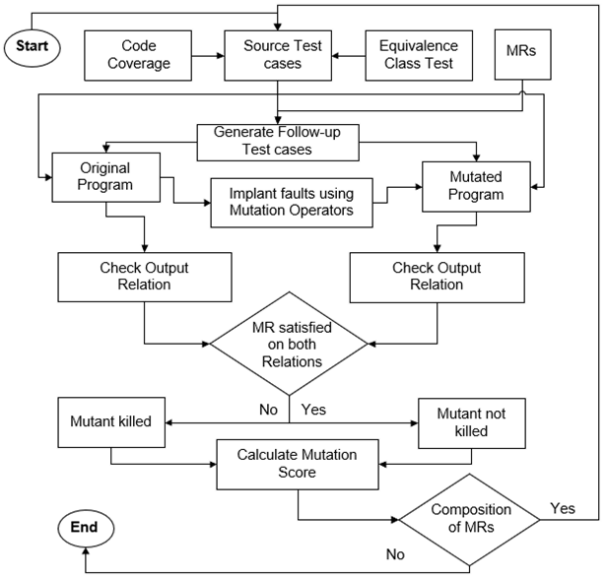

FIGURE 4.1: Proposed Framework for MR Evaluation.

4.1.1 Generation of Source Test Cases

Software test cases are defined as the set of circumstances under which a tester must test and determine whether the SUT accurately produced the expected result [76]. In the domain of software testing, a set of test cases are determined according to some testing techniques such as white box testing, black box testing, error based testing etc., [7]. The software is then tested using these test cases to find out the internal structure and function of the system along with some of its common faults [77]. For example, let us suppose that we have a program p(x) having a function f(x) on domain d. T shows the set of input values or test cases; T = $(t_1, t_2, , \ldots, t_n) \subset$ d. The program p can be tested by running p on T. The output values of the program $p(t_1), p(t_2), p(t_3)...p(t_n)$ are then verified against the expected results $f(t_1), f(t_2), f(t_3)\ldots, f(t_n)$. If $p(t_i) = f(t_i)$ then it shows that test case t_1 is successful test case but if $p(t_i) \neq f(t_i)$ then it means that t_1 is a failure causing test case. The mechanism that decides that whether $p(t_i) = f(t_i)$ for i= 1, 2, 3,. . . , n is called the oracle [8].

In MT, after the identification of MRs, the next step is to generate the test cases known as source test cases. In literature, the source test cases are generated using some traditional test case generation techniques such as random test generation through random model or Boolean model, structural or program based test generation techniques, behavioral or specification based, symbolic evaluation method, combinatorial techniques and fault based test generation techniques etc., [69], [29], [42] or through some tool i.e., EvoSuite (it generates source test cases automatically through coverage criterion) [6].

4.1.2 Test Case Adequacy through Equivalence Class Testing and Code Coverage

After the generation of source test cases, the next step is to check the adequacy of test cases. Adequacy criterion describes the selection of test cases and determines that whether the test suite is adequate during the testing process or not [78]. Adequacy of test cases can be checked either through black box testing techniques (boundary value analysis, equivalence partitioning, state transition testing etc,.) or through white box testing techniques (branch testing, control flow testing, loop testing, data flow testing etc,.) [79].

In this research, first the test cases are generated through equivalence class testing. In equivalence class testing, the program's input and output domains are partitioned into finite number of classes in such a way that all the cases in each partition follow the same functionality [80]. We have divided our dataset into five distinct classes based on the properties of images such as horizontal dimension, vertical dimension, bit depth, resolution, and type of image (T1 weighted image, T2 weighted image, and flair image).

Further, the test cases selected through equivalence class testing is checked through program coverage criterion (branch coverage, statement coverage). Code coverage shows the completeness of testing and can be used to evaluate the effectiveness of the test [81]. If the test suite does not fulfill the coverage criterion against the expected threshold value (suppose the expected threshold value is to achieve 100

percent coverage criterion), then it means new test cases are required. The reason of creating new test cases is to improve the coverage criteria, not achieved through previous test cases.

We have selected 95 test cases through equivalence class testing. Afterwards, the adequacy of source test cases is checked through program coverage (statement coverage and branch coverage). If the test cases (accumulatively) do not achieve 100% coverage then we need more test cases to achieve 100% branch coverage.

4.1.3 Generation of Follow-up Test Cases

The purpose of MT is to extract valuable information from the successful test cases that do not reveal faults [19]. MT introduced the generation of new test cases (aka follow-up test cases) from the existing test cases (aka source test cases) using MRs [82]. The follow-up test cases are generated by applying transformation to the source test cases [83]. Suppose we have a program P that does not have a test oracle. Let test suite of source test cases (STC) and source test cases (stc) are defined as: STC = $\{stc_1, stc_2, stc_3 \ldots..stc_n\}$ with n stc. A follow-up test case (ftc) is generated by applying transformation against each source test case using an MR. Hence, the test suite of follow-up test cases after transforming the source test cases using MR be $FTC_{MR} = \{ftc_1, ftc_2, ftc_3 \ldots..ftc_n\}$ [84]. The program P is executed with the source test cases $\{stc_1, stc_2, stc_3 \ldots..stc_n\}$ and follow-up test cases $\{ftc_1, ftc_2, ftc_3 \ldots..ftc_n\}$. The outputs of source and follow-up test cases are recorded. The outputs of both the source and follow-up test cases are compared to check the satisfaction of relevant MR. If MR is violated then it shows that the program P is faulty otherwise P passes the test [36].

As discussed earlier, we have used the MRs of edge detection and morphological image operation. The follow-up test cases of edge detection are discussed in [85]. Here, we will discuss the follow-up test cases generated through source test cases and MRs. The follow-up test cases of proposed MRs are given below:

$$MR_1 : C(\delta_s(Im)) = \delta s(C(Im))$$
$$MR_2 : C(\varepsilon_s(Im)) = \varepsilon s(C(Im))$$

In MR_1 and MR_2, C(Im) is the follow-up test case.

$$MR_3: T(\delta_s(Im)) = \delta s(T(Im))$$
$$MR_4: T(\varepsilon_s(Im)) = \varepsilon s(T(Im))$$

In MR_3 and MR_4, T(Im) is the follow-up test case.

$$MR_5: (A \oplus B) \oplus C = (A \oplus C) \oplus B$$

Where $(A \oplus C)$ is the follow-up test case.

$$MR_6: Trans(\varepsilon_s(Im)) = \varepsilon_s(Trans(Im))$$

Where, Trans(Im) is the follow-up test case of the above MR.

4.1.4 Evaluation of Metamorphic Relations

Evaluation of MRs is an indication of their fault detection capabilities. The greater the fault detection capabilities, the greater the abilities to detect more faults from a program. We have checked the fault detection capability of these MRs through mutation testing. In mutation testing, Mutation operators play an important role to generate the mutants. Previously, very few mutation operators are used which even did not highlight the effectiveness of these operators. We have used nine mutation operators to check that which operator is most effective to generate the maximum number of mutants and through which operator maximum number of mutants can be killed. Our work is relevant to [56] and [27] so we have compared the mutation operators used in our proposed framework with these two techniques. The mutation operators used in the existing techniques ([56] , [27]) and in proposed framework are given in Table 4.1.

If the output of source and follow-up test cases satisfies the relation then mutation testing can be performed by seeding the faults into the original program to check for MR violation. The process to check the original and mutated program

TABLE 4.1: Mutation Operators Used in Existing Techniques and in Proposed Framework

Mutation Operators in Existing Techniques	Mutation Operators in Proposed Framework
ROR- Relational operator replacement	AOD- Unary arithmetic operator deletion
LOR- Logical operator replacement	AOR- Arithmetic operation replacement
	LOR- Logical operator replacement
	ROR- Relational operator replacement
	OIL- One iteration loop
	RIL- Reverse iteration loop
	SIR- Slice index remove
	SDL- Statement deletion
	ZIL – Zero iteration loop

is explained through an example. Suppose we have two test cases t_1 (source test case) and t_1' (follow-up test case) that have to be tested under the original program p. The outputs of test t_1 and t_1' can be recorded as r_1 and r_1'. Afterwards, the same test cases t_1 and t_1' can be executed on the modified program $p^{'}$ with mutant M. The outputs should be saved as r_2 and r_2'. The mutant 'M' is not killed if both (r_1, r_1') and (r_2, r_2') satisfy their related MR [46]. Otherwise the mutant is killed. After mutation testing, mutation score is to be calculated. Mutation score indicates the fault detection capability of each MR. In existing literature, i.e., [56] , [27], [71], [46], all the MRs are evaluated by seeding faults manually in the code or through a tool that calculates the mutation score automatically through traditional mutation testing approach. We have seeded the faults manually and checked the relation manually on both the programs (original and mutated) for the calculation of mutation score.

Mutation score indicates the fault detection capability (FDC) of MR. For the calculation of mutation score; we have examined the mutants manually and have removed all the equivalent mutants. The formula of fault detection rate is given in 4.1

$$\text{FDR} = \frac{\text{Number of killed mutants} \times 100}{\text{Total no. of mutants - No. of equivalent mutants}} \tag{4.1}$$

According to the formula, if the mutation score is 1 then it shows that MR is strong (high fault detection rate) whereas a 0 score would show MR is weak (low

fault detection rate). We can also say that if the mutation score is near to 0, the MR is weak enough to find the violation.

4.1.5 Composition of Metamorphic Relations

If the MR satisfies the mutation tests for all the cases, then it shows that the MR is too weak to find the violation. Therefore, MRs should be refined or new MRs should be generated that are strong enough to find the violations. Composition of metamorphic relation is an idea proposed in [28] for the construction of MRs. The concept is to merge multiple MRs into a single MR. The composited relation contains all the properties of the original MR. Suppose, we have two MRs: MR_1 and MR_2. The relation MR_2 is compositable to MR_1 if and only if for any source test case of MR_1, its corresponding follow up test case is always used as a source test case for MR_2. This process of composition reduces the number of test cases (cost effective) and embeds several properties into a single MR [86]. After the composition of metamorphic relations, new source test cases and follow up test cases are generated to check the strength of MR through metamorphic testing process. According to [59], fault detection rate of MR_{12} is not smaller than MR_1 or MR_2, and MR_{12} is very likely to be at least equal to the maximum of MR_1 and MR_2. Now we consider the two MRs of edge detection to see how composition of two MRs take place.

$$MR_1 : C(E(Im)) = E(C(Im))$$
$$MR_2 : T(E(Im)) = E(T(Im))$$

Where,

Im: Source test case

C: Counter clock-wise rotation at 90 degree

T: Transposition

E: Edge detection Program

After composing the above two MRs, the resultant composed MR is given below:

$$MR_{12} : E(T(C(Im))) = T(E(C(Im)))$$

We have composed the four MRs of edge detection proposed in [56]. First, we have composed two MRs and make 12 new MRs. Then we have composed three MRs and make 24 new MRs and in the end we have composed four MRs and make 24 new MRs. The detail is given in Chapter 6, section 6.5.

Now we discuss the pros of using our framework. In literature, mostly, the source test cases are generated randomly and there was no systematic way to generate the test cases.. In our framework, we have proposed a systematic way to generate the source test cases using equivalence class partitioning and code coverage. We have constructed new MRs by using composition. The composed MRs contain the properties of component MRs.

4.2 Summary

This chapter discusses our proposed framework for the evaluation of MRs. Each and every step of the proposed framework is discussed in detail such as generation of source and follow-up test cases, test case adequacy through strong equivalence class testing, evaluation of MRs, and composition of MRs.

Chapter 5

Proposed Metamorphic Relations

This chapter is divided into two sections. The first section describes several IP operations and the IP operations which we have considered in our research. The second section discusses the proposed MRs for dilation and erosion and the existing MRs for edge detection and dilation and erosion operations in detail.

5.1 Image Processing Operations

As discussed earlier, there are several IP operations such as edge detection, morphological image operations, image segmentation, image enhancement, euclidean distance transform, image region growth, image restoration etc,. to perform specific tasks on the images. The details of some of the common IP operations are given below:

5.1.1 Edge Detection

In IP, edge detection plays a vital role for identifying the immediate changes in grayscale images [87]. The goal of spotting abrupt changes in picture brightness is to record significant occurrences and modifications to the world's characteristics. The edge detection technique is divided into two parts i.e,. first derivative edge

detection (Sobel, Canny, Roberts) and second derivative edge detection (LoG) [88]. Some of the common first derivative edge detection algorithms are as follows:

- Canny Edge Detection: Canny edge detection was created by John Canny in 1983. It is designed to identify the low intensity edges. The steps of canny edge detection algorithm is given below: [89]

  1. Image smoothing using Gaussian filter.
  2. Image smoothing using Gaussian filter.
  3. Calculates the de-noising image's gradient's magnitude and direction.
  4. Uses non-maxima suppression in accordance with gradient direction to produce a unilateral edge response.
  5. In the end, performs double thresholding to identify and connect the edges.

- Sobel Edge Detection: In 1970, sobel edge detection algorithm was first proposed by Sobel [90]. A noise-free image's edges can be quickly and precisely detected by the Sobel operator [91]. The Sobel operator calculates the edges in two directions i.e,. horizontal direction and vertical direction. When X filter is applied on the image, it will highlight the vertical edges whereas the Y filter highlights the horizontal edges of an image [92].

- Prewitt Edge Detection: Prewitt edge detection algorithm was created by Prewitt in 1970. A suitable method for determining the magnitude and orientation of an edge is the Prewitt edge detector. There are only 8 possible orientations for the Prewitt operator. This gradient-based edge detector has estimates for eight directions in the 3x3 neighbourhood. Calculations have been made for all eight convolution masks. Afterward, one convolution mask is chosen, specifically the one with the largest module [93].

- Roberts Edge Detection: In Roberts operator, a 2 by 2 convolution kernel is used. G_x is a straightforward kernel whereas G_y is rotated by 90 degree. When using Robert's cross operator, at each point, pixel value corresponds to the input image's current absolute magnitude [94].

5.1.2 Morphological Image Operations

Morphological operations explain how an image interacts with a structuring element S [95]. In comparison to the input image, the structural element is typically a small binary image, either with zero or one values [96]. Erosion, dilation, opening, closure, filtering, skeletonization, top-hat, edge-off, and watershed transforms are useful methods of morphological image processing [97]. Dilation and erosion are the main morphological operations that increases or decreases the region of the image according to the structuring element [2]. Figure 5.1 shows probing an image with the structuring element.

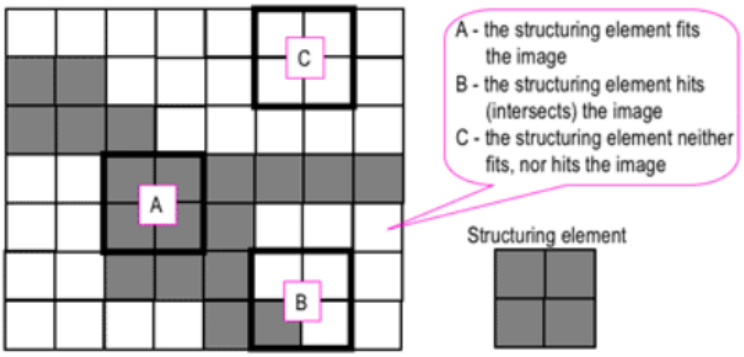

FIGURE 5.1: Probing an Image with Structuring Element [98].

According to Figure 5.1, a structuring element is said to fit an image if each of its pixels is set to 1, and the corresponding pixel in the image is also set to 1.If at least one of the pixels in a structuring element is set to 1 and the corresponding pixel in the image is also 1, then the structuring element also "hits" or "intersects" the image.

5.1.3 Image Segmentation

Image segmentation is a crucial and challenging step in the field of IP. It has grown in popularity in the area of image understanding [99]. Segmentation plays a vital role in a broad range of applications such as medical imaging, surveillance (video), navigation system etc,. In image segmentation, the entire image is divided into a number of regions that share some characteristics such as colour, grayscale, spatial

texture, and geometric shapes etc., [100]. Some of the types of image segmentation are as follows:

- Region Based Segmentation: For object detection and recognition, region-based segmentation techniques are effective tools. The goal of region based segmentation is to separate the various entities in the image by dividing it into homogeneous zones [101].
- Edge Based Segmentation: Edge-based segmentation uses edge detection operators to locate edges in an image. These edges identify the locations of grey-level discontinuities in the image [102].
- Thresholding: Thresholding is the main technique for image segmentation that converts a greyscale image into a binary image using a single threshold value. The most crucial step in this procedure is selecting the threshold value (T), pixels whose intensities are above the threshold value for the foreground region, and all other pixels in the background region [103].
- Clustering: Clustering is the process of assembling homogeneous data into groups based on analogy criteria. The primary clustering segmentation algorithm is K-means clustering, in which each component of the dataset can only belong to one cluster at a time [103].

5.1.4 Image Reconstruction

In the field of IP, image reconstruction is a common technique [104]. Reconstruction of an image is the process of creating a two- or three-dimensional image from fragmented or insufficient data, such as radiation readings obtained during a medical imaging study. To create a readable and usable image or to sharpen an image to make it useful, a mathematical formula may need to be applied when using certain imaging techniques. For instance, image reconstruction can assist in creating a three-dimensional image of the body from a collection of individual camera images during CT scanning [105].

5.1.5 Euclidean Distance Transform

One of the important operation of IP is Euclidean distance transform D(A), which translated a binary image A into a real-valued grayscale image. Each grayscale pixel has a distance from the nearest white binary pixel (pixel having 0 value) associated with it [69]. Figure 5.2 shows the original image and its distance transform.

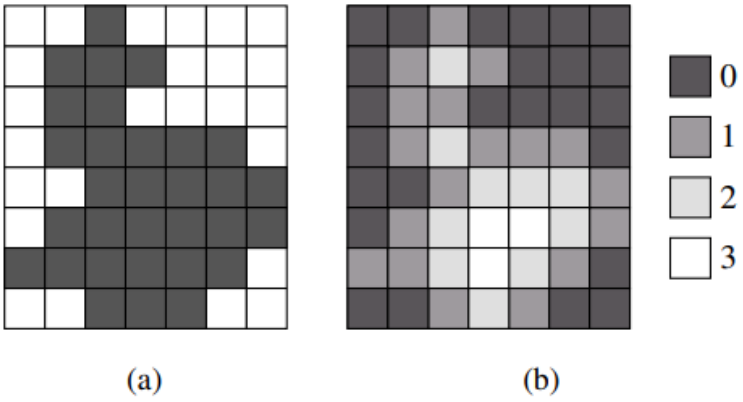

FIGURE 5.2: (a) Original Image (b) Distance Transform [69].

There are many more IP operations but we have covered only a few. In MT literature, the authors have used some of the operations such as edge detection, euclidean distance transform, image region growth, dilation and erosion, and image reconstruction and used the properties of these operations as MRs. In this research, we have considered only two operations of IP i.e,. edge detection (Sobel and Canny) and morphological image operations (dilation and erosion).

5.2 Metamorphic Relations

In the field of software testing, MT is the practical solution to oracle problem. In MT, the foremost step is the identification of MRs [106]. These properties include various forms such as equalities, inequalities, periodicity properties, convergence constraints, subsumption relationships etc., [107]. The MRs can be identified based on the guidance provided by the experiences ([69], [27]) and the domain

knowledge in the specific field such as IP, web applications, networks, machine learning etc. So, based on the domain knowledge about the proposed algorithm or the SUT functionality, one or multiple MRs can be identified. The significant role of MR is to generate test cases as well as to verify the test results in the absence of test oracle [108]. An MR defines a logical connection between a source test case with its observed output [109]. If there is a discrepancy between how the output changes and what the anticipated MR implies, it is termed a violation of the MR [110]. If an MR is violated then it shows the presence of bugs in the SUT [107]. The effectiveness of MT is dependent on the selection of MRs with high FDR. The higher the FDR of the MR, the higher is the fault detection capability.

5.3 Existing Metamorphic Relations

In existing literature, different MRs have been proposed in the field of IP such as MRs for edge detection operation, dilation and erosion operations, image region growth, and euclidean distance transform. In this research, we have used the existing MRs of edge detection presented in [56] and dilation and erosion MRs presented in [27] for the evaluation purpose because we have used a case study of MRI brain images and in diagnosis process, edge detection and dilation and erosion operations are used as pre-processing step. Furthermore, we have proposed six new MRs for dilation and erosion operations and evaluated them using our proposed framework.

5.3.1 MRs for Edge Detection

The MRs of edge detection [56] is given in Table 5.1.

According to Table 5.1,
Im is the source test case
E is the edge detection program
C(.) is the counter clock wise rotation at 90 degree

TABLE 5.1: MRs for Edge Detection

MR	Mathematical Property
MR_1 : Counter clock wise rotation at 90^o	$C(E(Im)) = E(C(Im))$
MR_2 : Transposition	$T(E(Im)) = E(T(Im))$
MR_3 : Reflection at the ordinate	$M_x(E(Im)) = E(M_x(Im))$
MR_4 : Reflection at abscissa	$M_y(E(Im)) = E(M_y(Im))$

T(.) is the transpose of an image Im

$M_x(.)$ is the reflection at the ordinate

$M_y(.)$ is the reflection at abscissa.

5.3.2 Existing MRs for Dilation and Erosion

The MRs of dilation and erosion [27] are given in 5.2.

According to Table 5.2,

I: Input image

Ref_{ord}: Reflection at the ordinate

Ref_{abs}: Reflection at abscissa

Output: Output of the image after applying dilation and erosion operations

δ_s: Dilation operation with structuring element s

ε_s: Erosion operation with structuring element s

c: Complement of an image I. In image processing, complement means the background of the image

$Size_{obj}$: Size of image object. Size of image object increases in dilation whereas decreases in erosion

Pix_{list}: Pixel list

$Number_{obj}$: Number of objects. Number of objects decreases in dilation whereas increases in erosion

$\oplus$: Dilation operation

x: 2D factor

TABLE 5.2: Existing MRs for Dilation and Erosion Operations

MR	Mathematical Property
R_1 : Reflection at the ordinate	$Ref_{ord}(Output(I)) = Output(Ref_{ord}(I)$
R_2 : Reflection at abscissa	$Ref_{abs}(Output(I)) = Output(Ref_{abs}(I))$
R_3 : Duality	$\delta_s(I) = \varepsilon_s(I^c)$ $\varepsilon_s(I) = \delta_s(I^c)$
R_4 : Non-inverses	$\delta_s(\varepsilon_s(I)) \neq I \neq \varepsilon_s(\delta_s(I))$
R_5: Size of image object changes	$Size_{obj}(\delta_s(I)) \geq Size(I)$ and $Pix_{list}(I) \subset Pix_{list}(\delta_s(I))$
R_6 : No. of objects in image changes	$Number_{obj}(\delta_s(I)) \leq Number_{obj}(I)$
R_7 : Commutative	$\delta_s(I) = I \oplus S = S \oplus I = \delta_I(S)$ $\varepsilon_s(I) \neq \varepsilon_I(S)$
R_8 : Translation invariance	$\delta_{s+x}(I) = \delta_s(I) + x$

5.3.3 Proposed MRs for Dilation and Erosion

We have proposed six new MRs for dilation and erosion operation (four general and two specific). The details of these MRs are given below:

5.3.3.1 Counter Clock Wise Rotation at 90 degree

$$MR_1\text{: } C(\delta_s(Im)) = \delta_s(C(Im))$$
$$MR_2\text{: } C(\varepsilon_s(Im)) = \varepsilon_s(C(Im))$$

Where,

Im is the input Image

C(.) is the counter clockwise rotation at 90 degree

δ_s is the dilation operation with structuring element s

ε_s is the erosion operation with structuring element s.

The image output of counter-clock wise rotation at 90 degree followed by morphological operations should be similar to image output of morphological operations followed by counter-clock wise rotation at 90 degree.

5.3.3.2 Transposition

$$MR_3: T(\delta_s(Im)) = \delta_(T(Im))$$
$$MR_4: T(\varepsilon_s(Im)) = \varepsilon_s(T(Im))$$

Where,

Im is the input Image

T(.) is the transpose of an image

δ_s is the dilation operation with structuring element s

ε_s is the erosion operation with structuring element s.

The image output of transposition followed by dilation and erosion should be similar to image output of dilation and erosion followed by transposition.

5.3.3.3 Enhanced Associative Property

$$MR_5: (A \oplus B) \oplus C = (A \oplus C) \oplus B$$

Where,

A is the input image

B and C are the structuring elements

$\oplus$ is the dilation operation.

Image dilated with structuring element B and then dilated with structuring element C should give same results when we first dilate the image with structuring element C and then dilated with structuring element B. This property is specific to dilation as erosion does not fulfill this property.

5.3.3.4 Image Translation

A translation operation moves an image in either the x- or y-direction, or both, by a predetermined number of pixels [111]. The MR of image translation is given below:

$$MR_6: Trans(\varepsilon_s(Im)) = \varepsilon_s(Trans(Im))$$

Where,
Im is the input image
Trans(.) is the image translation
ε_s is the erosion operation with structuring element s.

The output of image translation followed by erosion should be similar to the output of erosion followed by image translation. This property is not applicable on dilation operation.
The tabular representation of proposed MRs are given in Table 5.4

TABLE 5.3: Proposed MRs for Dilation and Erosion Operations

MR	Mathematical Property
MR_1 : Counter clock wise rotation at 90^o (for dilation)	$C(\delta_s(Im)) = \delta_s(C(Im))$
MR_2 : Counter clock wise rotation at 90^o (for erosion)	$C(\varepsilon_s(Im)) = \varepsilon_s(C(Im))$
MR_3 : Transposition (for dilation)	$T(\delta_s(Im)) = \delta_(T(Im))$
MR_4 : Transposition (for erosion)	$T(\varepsilon_s(Im)) = \varepsilon_s(T(Im))$
MR_5 : Enhanced Associative Property	$(A \oplus B) \oplus C = (A \oplus C) \oplus B$

TABLE 5.4: Proposed MRs for Dilation and Erosion Operations

MR	Mathematical Property
MR_6 : Image Translation	$Trans(\varepsilon_s(Im)) = \varepsilon_s(Trans(Im))$

5.4 Summary

This chapter discusses several IP operations such as edge detection, morphological image operations, euclidean distance transform etc,. in detail. The existing MRs of edge detection and morphological image operations (dilation and erosion) are discussed. The MRs proposed for the dilation and erosion operations are also discussed in detail.

Chapter 6

Evaluation of Proposed MRs

This chapter is divided into six sections. The first section discusses the experimental data used in our experiments. The second section discusses the detailed discussion on the results related to the evaluation of mutation operators. The third section discusses the results related to the evaluation of Existing and proposed MRs. The fourth section describes the comparison results of our proposed MRs with the existing MRs as well as the comparison results of our proposed framework with the existing techniques. the fifth sections shows the composition results in detail, and the last section discusses the evaluation of MRs using SSIM.

6.1 Experiment Design

In this section, we have discussed the details about SUT used for our experiment; dataset, original test cases, and coverage criterion used.

6.1.1 Subject Program

In this section, we will discuss the subjeect programs we have used for our experiments. The subject program in this research consists of the following:

• Sobel edge detection program

- Dilation and erosion programs
- Improved Canny edge detection program

6.1.1.1 Sobel Edge Detection

Sobel is a convolution process that goes from the designated window to the target image. In the Sobel convolution with a 3x3 window, the predicted gradient should be exactly at the center of the window in the image [112]. The horizontal and vertical direction templates G_x and G_y are calculated for the image matrix's convolution. As a result, we obtain the gradients in the X and Y direction as an approximation of the brightness difference [113]. On the basis of arrangements of adjacent pixels, the gradient size determined by the Sobel operator is:

$$|G| = \sqrt{G_x^2 + G_y^2}$$
$$|G| = |G_x| + |G_y|$$

The input and output of Sobel edge detection program after calculating horizontal and vertical edges is given in Figure 6.1

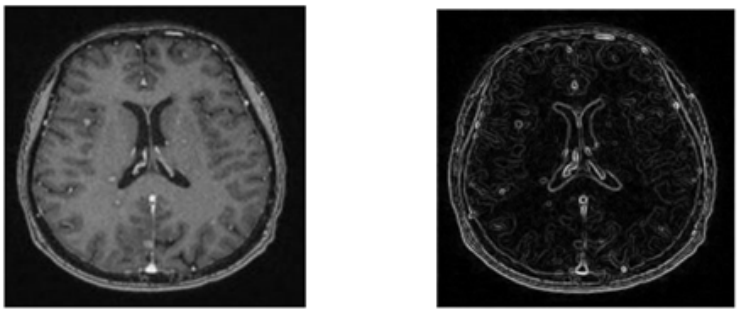

FIGURE 6.1: (a) Input Image (b) Output of Edge Detection.

6.1.1.2 Dilation and Erosion

- Dilation: The dilation process increases the size of an object. The type and shape of the structuring element determine how much it grows. The dilation of an image A with structuring element B is defined as: [114]

$$A \oplus B = \{Z(B) \cap A \neq \emptyset\}$$

• Erosion: The erosion operation is a complement to the dilation operation when considering the operation effect. The object loses size as a result of that erosion operation [115]. The erosion of an image A with structuring element B is defined as: [114]

$$A \ominus B = \{Z|(B)_z \subseteq A\}$$

The sample inputs of MRI brain images and their expected output images after performing dilation operation and erosion operation are shown in Figure 6.2.

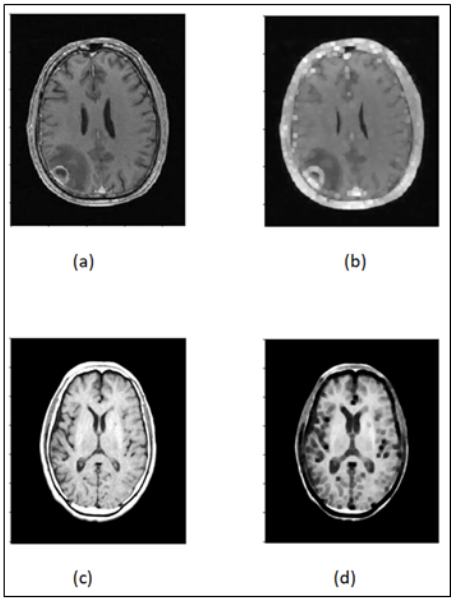

FIGURE 6.2: (a) Input Image (b) Output of Dilation (c) Input Image (d) Output of Erosion.

6.1.1.3 Improved Canny Edge Detection

The improved cannay edge detection algorithm [55] consists of seven steps for identification of brain tumor. We have implemented our algorithm of detection of edges of the soft tissues in the MRI of a brain image by using first five of the seven steps given below:

• Apply a fast local Laplacian filter on the original image for the enhancement of

contrast and texture.

• Convert the image into a grayscale image.

• Apply K-means clustering and fuzzy C-Means clustering.

• Apply traditional Canny edge detection to identify the edges in the MRI of a brain image.

• Apply median filter to smooth out the lines detected in step four.

The input and output of Sari 's improved edge detection algorithm using our dataset is given in Figure 6.3

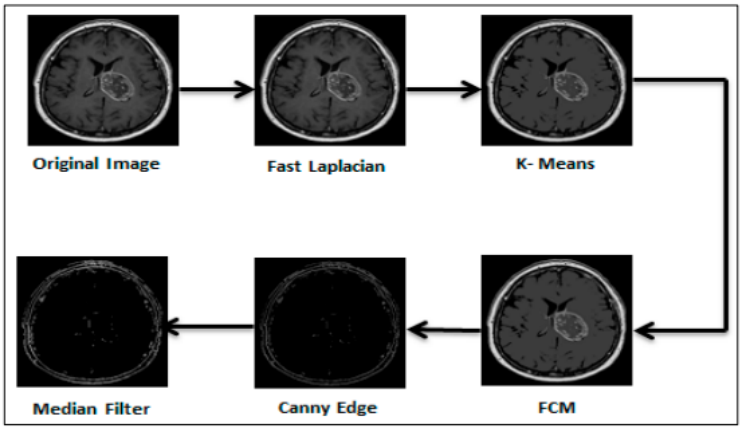

FIGURE 6.3: Output of Improved Canny Edge Detection.

6.1.2 Source Code

As discussed earlier, we have used four source codes (Sobel edge detection, dilation, erosion, and improved canny edge detection) to validate our experiments. The detail of each source code is given below:

• Sobel Edge Detection Program:

We have used a well-structured code of Sobel edge detection written in Python version 3.8.3 for our implementation. The code consists of 41 statements and 10 branches.

• Dilation and Erosion Programs:

We have used a well-structured code of Dilation and Erosion written in Python version 3.8.3 for our implementation. The code consists of 46 statements and 12

branches.

• Improved Canny edge Detection program:

The algorithms of all five steps mentioned above are taken from GitHub or GeeksforGeeks and are consolidated in a single Python file which has 347 statements and 110 branches. The sources of above given codes are given in Table 6.1.

TABLE 6.1: Sources of Source Codes

Operation	Source
Sobel Edge Detection	https://towardsdatascience.com/edge-detection-in-python-a3c263a13e03
Dilation	https://python.plainenglish.io/image-dilation-explained-easily-e085c47fbac2
Erosion	https://medium.com/analytics-vidhya/2d-convolution-using-python-numpy-43442ff5f381
Fast Local Laplacian	https://github.com/qq491577803/fast-local-laplacian-filter/blob/main/myFastLocalLaplacian.py
K-Means	https://docs.opencv.org/3.4/d1/d5c/tutorial_py_kmeans_opencv.html
Canny Edge Detection	https://www.geeksforgeeks.org/implement-canny-edge-detector-in-python-using-opencv/
Median Filter	https://codereview.stackexchange.com/questions/191974/median-filter-implementation-in-python

6.1.3 Dataset

In this research, we have used the dataset of MRI brain images. A diversified collection of images in jpg format, as source test cases, is taken from kaggle: https://www.kaggle.com/datasets/abhranta/brain-tumor-detection-mri?resource=download for our study. The dataset consists of 3000 images (T1 weighted images, T2 weighted images and flair images) with 1500 images having brain tumor and 1500 images with no brain tumor. The basic three types of images, used as test cases are shown in Figure 6.4.

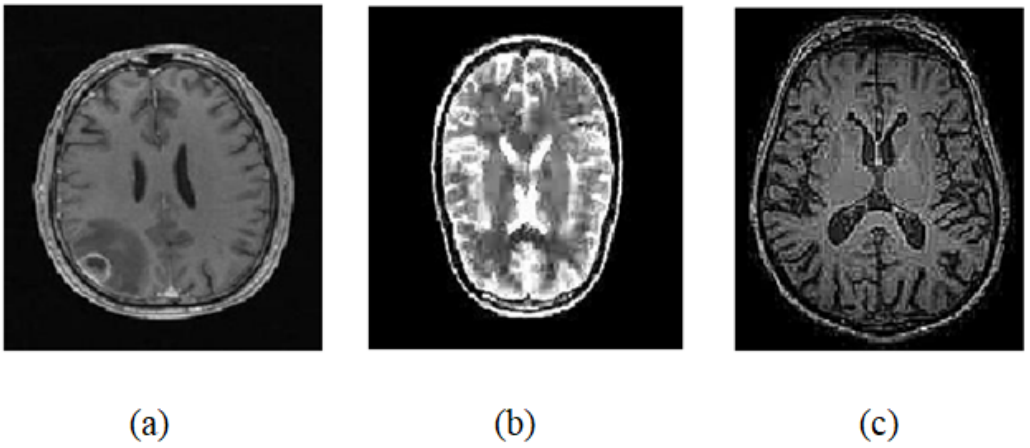

FIGURE 6.4: (a) T1 weighted image (b) T2 weighted image (c) Flair Image.

The dataset comprises of 1,500 images with no brain tumor and 1,500 images with brain tumor with multiple properties such as horizontal and vertical dimensions, resolution in dpi, image type (T1 weighted, T2 weighted, Flair), and bit depth. We have discussed the range against each property as well as the number of images in the dataset against each property. The total number of image against each category is described in Table 6.2

TABLE 6.2: Classification of Dataset.

Properties	Range	No. of images
Horizontal Dimension	h1: 1 - 300	2081
	h2: 301 - 650	623
	h3: 651+	296

TABLE 6.2: Classification of Dataset.

Properties	Range	No. of images
Vertical Dimension	v1: 1 - 350	1488
	v2: 351 - 700	1192
	v3: 701+	320
Resolution	r1: 1 - 90 dpi	28
	r2: 91 - 99 dpi	2905
	v3: 100 - 450 dpi	67
Bit Depth	b1: 8	106
	b2: 24	2894
Type of Image	t1: T1 weighted images	1223
	t2: T2 weighted images	781
	t3: Flair iamges	996

Table 6.2 shows the selected five image properties (attributes) along with their ranges (classes) and 3000 images (1500 having brain tumor and 1500 having no brain tumor) against each class.

6.1.4 Source Test Cases

We have selected source test cases using strong equivalence class testing and grouped the properties of MRI images into five attributes: horizontal dimension, vertical dimension, bit depth, resolution, and the image type. Each attributes is further divided into multiple classes as shown in Table 6.3. Each attribute has three classes except "bit depth", as it has only two classes. We have generated the source test cases automatically, by writing a program in python. The program will make thee folders of these attributes and put the images into their relevant

class. For example, when we execute the program it will create two folders for the attribute "bit depth". One folder contains 8 bit depth images whereas the second folder contains the 24 bit depth images.

TABLE 6.3: Classes Using Strong Equivalence Class Testing.

Attributes	Classes
Horizontal Dimension	h_1: 1 – 300
	h_2: 301 – 650
	h_3: 651+
Vertical Dimension	v_1: 1 – 350
	v_2: 351 – 700
	v_3: 701+
Resolution	r_1: 1 – 90 dpi
	r_2: 91 – 99 dpi
	r_3: 100 – 450 dpi
Bit Depth	b_1: 8
	b_2: 24
Type of Image	t_1: T1 weighted images
	t_2: T2 weighted images
	t_3: Flair images

According to Table 6.3, "Horizontal Dimension" is a class which is further divided into three sub classes. "Vertical Dimension" is divided into three classes as well. "Resolution" and "Type of Image" has three sub classes each. "Bit Depth" is a class which is further divided into two sub classes.

As discussed earlier, we have used strong equivalence class testing for the generation of source test cases. In strong equivalence class testing, we will make all possible combination from these classes for the generation of source test cases. The total numbers of classes generated through these combinations are: 3 x 3 x 3 x 2 x 3 = 162. Out of 162 classes, only 95 classes (33-T1, 29-T2, 33-Flair) are obtained with a few with 8-bit depth value and resolution between 1 to 90 range.

There are few images in each image type (T1, T2, flair) with 8-bit depth as well as resolution between the range 1 to 90 that are selected. All missing 67 classes with 8-bit depth value and resolution having range from 1 to 90 are unavailable. The results of 8-bit depth value or lower show either very dark (T1 and flair) images or very bright (T2) images which make it very difficult to detect the lesions accurately.

6.1.5 Code Coverage

The adequacy of these 95 test cases is checked through white box testing which validates code coverage for branch coverage and statements coverage. The test suite should cover 100 percent branch coverage for the initialization of our proposed testing process; otherwise, new test cases would be required to complete the branch coverage to 100 percent. Our test suite covers 100 percent for branch coverage and statement coverage respectively. The summary of code coverage is given in Table 6.4

Table 6.4: Code Coverage Summary.

	Sobel Edge	Dilation	Erosion	Improved canny
No. of Test Cases	95	95	95	95
Total No. of Statements	41	46	46	341
No. of Covered Statements	41	46	46	341
Statement Coverage (%)	100%	100%	100%	100%
Total No. of Branches	10	12	12	110
No. of Covered Branches	10	12	12	110
Branch Coverage (%)	100%	100%	100%	100%

As shown in Table 6.4, 95 test cases (accumulative) cover 100% statement coverage as well as branch coverage in all the three programs. So, we do not need more test cases for our test suite.

6.2 Effectiveness of Mutation Operators

In this section, we have assessed the effectiveness of mutation operators by calculating the percentage of mutants generated by each mutation operator and the number of mutants killed by each operator. Mutation score of each operator shows the FDR of each mutation operator. The formula of mutation score is described in Chapter 2. The formula to calculate the FDR of each mutation operator is depicted in Equation 2.1. The percentage of generated mutants is calculated by the formula given in Equation 6.1.

$$\text{V} = \frac{\text{No. of mutants generated by V operator} \times 100}{\text{Total no. of mutants generated by all the operators}} \tag{6.1}$$

Where,

V: any mutation operator

6.2.1 Effectiveness of Mutation Operators used in Edge Detection

Table 6.6 shows the effectiveness of mutation operators in terms of mutants generated and mutants killed by using each operator for edge detection operation. Each operator shows the total number of mutants it has generated and the number of mutants killed through each of the operator.

TABLE 6.5: Effectiveness of Mutation Operators used in Edge Detection

Mutation Operators	No. of Mutants Generated	% of Generated Mutants	No.of Killed Mutants	FDC of Mutation Operations
AOD	6	3.70%	MR_1: 6 MR_2: 6 MR_3: 6 MR_4: 6	MR_1: 100% MR_2: 100% MR_3: 100% MR_4: 100%

TABLE 6.5: Effectiveness of Mutation Operators used in Edge Detection

Mutation Operators	No. of Mutants Generated	% of Generated Mutants	No.of Killed Mutants	FDC of Mutation Operations
AOR	98	60.49%	MR_1: 73 MR_2: 74 MR_3: 62 MR_4: 62	MR_1: 74.48% MR_2: 75.51% MR_3: 63.26% MR_4: 63.26%
COI	2	1.23%	MR_1: 2 MR_2: 2 MR_3: 2 MR_4: 2	MR_1: 100% MR_2: 100% MR_3: 100% MR_4: 100%
ROR	30	18.51%	MR_1: 22 MR_2: 22 MR_3: 22 MR_4: 22	MR_1: 73.33% MR_2: 73.33% MR_3: 73.33% MR_4: 73.33%
OIL	2	1.23%	MR_1: 1 MR_2: 1 MR_3: 1 MR_4: 1	MR_1: 50% MR_2: 50% MR_3: 50% MR_4:50%
RIL	2	1.23%	MR_1: 1 MR_2: 2 MR_3: 0 MR_4: 0	MR_1: 50% MR_2: 100% MR_3: 0% MR_4: 0%
SIR	4	2.46%	MR_1: 4 MR_2: 4 MR_3: 4 MR_4: 4	MR_1: 100% MR_2: 100% MR_3: 100% MR_4: 100%

TABLE 6.6: Effectiveness of Mutation Operators used in Edge Detection

Mutation Operators	No. of Mutants Generated	% of Generated Mutants	No.of Killed Mutants	FDC of Mutation Operations
SDL	16	9.87%	MR_1: 13 MR_2: 13 MR_3: 13 MR_4: 13	MR_1: 81.25% MR_2: 81.25% MR_3: 81.25% MR_4: 81.25%
ZIL	2	1.23%	MR_1: 2 MR_2: 2 MR_3: 2 MR_4: 2	MR_1: 100% MR_2: 100% MR_3: 100% MR_4: 100%

In the existing technique [56], a total of 31 mutants have been generated by using only two mutation operators, i.e., ROR, and LOR. In our proposed framework, we have employed nine mutation operators to generate a total of 162 mutants. It is observed that AOR operator has generated the highest number of mutants i.e., 98 mutants and killed 74 mutants in MR2 and 62 mutants in MR3 and MR4 respectively. ROR operator has generated 30 mutants and killed 22 mutants in each MR followed by SDL operator that has generated 16 mutants and killed 13 mutants in each MR. Though the mutation operators such as AOD, COI, SIR, and ZIL have achieved 100% mutation score in all respective MRs but on the other hand they have generated very less number of mutants i.e., 6, 2, 4, and 2 respectively. The percentage of mutants killed and mutants generated by each mutation operator is depicted in Figure 6.5 and Figure 6.6.

It is observed from Figure 6.5 and Figure 6.6 that the mutation operators such as AOD, COI, ZIL, and SIR have shown 100% mutation score in all the four MRs. But, their percentage with respect to generated mutants is very low i.e., 3.70%, 1.23%, 2.46%, and 1.23% respectively. The effectiveness is dependent on two factors i.e., mutation score and number of mutants generated. The operator

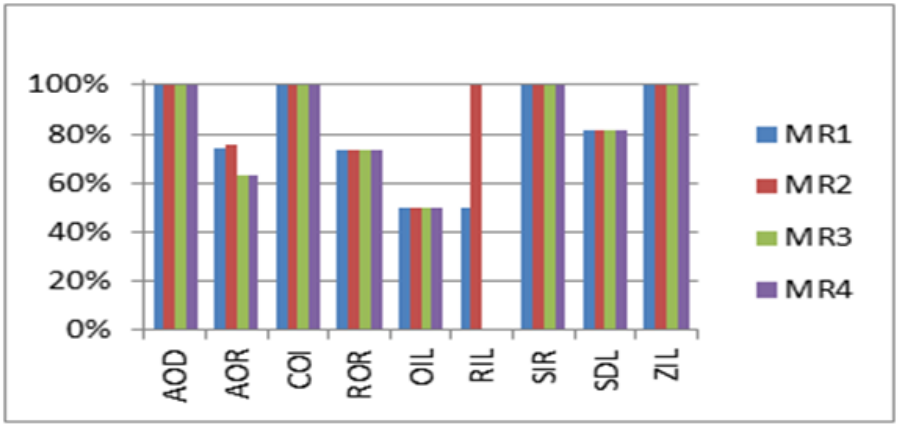

FIGURE 6.5: Percentage of Killed Mutants Used in Edge Detection.

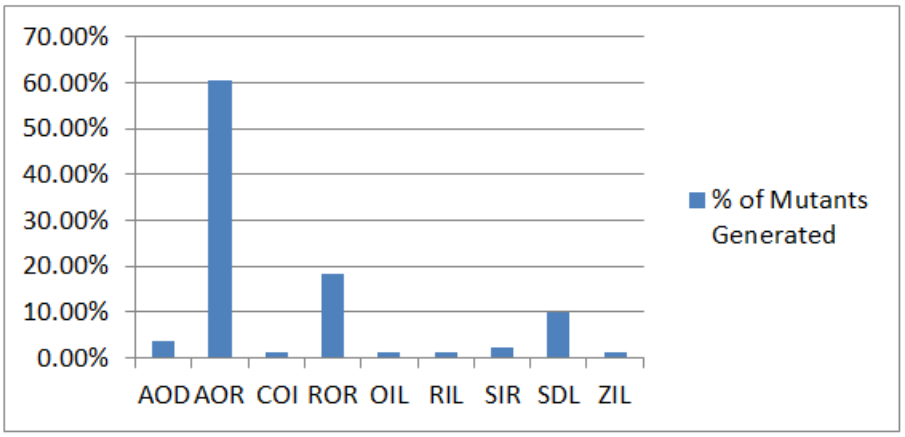

FIGURE 6.6: Percentage of Generated Mutants used in Edge Detection.

that scores high percentage in one of the two factors and scores very low in other is not as effective as the one having moderate scores in both the factors. So, AOR operator is the most effective operator because its lowest mutation score is 63% (MR_4) and the highest mutation score is 74% (MR_1) whereas its percentage to generate mutants is 60.49%. RIL is the least effective operator because its lowest mutation score is 0% (MR_4) and the highest mutation score is 100% (MR_2) whereas its percentage to generate mutants is only 1.23%. SDL operator has achieved a good mutation score of 81.25% followed by ROR having a mutation score of 73.33% against each MR whereas their percentage to generate mutants is 9.87% and 18.51% respectively. The effectiveness of SDL and ROR is almost similar because the mutation score of SDL is 12% higher than ROR whereas the percentage of ROR operator in terms of mutation generation is 10% higher than the SDL operator.

6.2.2 Effectiveness of Mutation Operators used in Dilation and Erosion Operations (Proposed Framework)

Table 6.7 shows the FDC (mutation score) and percentage of mutants generated by each mutation operator used in dilation and erosion operation. We have calculated the effectiveness of mutation operators used in our proposed MRs only.

TABLE 6.7: Effectiveness of Mutation Operators used in Dilation and erosion

Mutation Operators	No. of Mutants Generated	% of Generated Mutants	No.of Killed Mutants	FDC of Mutation Operations
AOR	95	73.07%	MR1: 51 MR2: 50 MR3: 51 MR4: 53 MR5: 56 MR6: 56	MR1: 53.68% MR2: 52.63% MR3: 53.68% MR4: 55.78% MR5: 58.94% MR6: 58.94%
COI	4	3.07%	MR1: 1 MR2: 1 MR3: 1 MR4: 2 MR5: 1 MR6: 2	MR1: 25% MR2: 25% MR3: 25% MR4: 50% MR5: 25% MR6: 50%
ROR	20	15.38%	MR1: 3 MR2: 3 MR3: 3 MR4: 6 MR5: 3 MR6: 5	MR1: 15% MR2: 15% MR3: 15% MR4: 30% MR5: 15% MR6: 25%

TABLE 6.7: Effectiveness of Mutation Operators used in Dilation and erosion

Mutation Operators	No. of Mutants Generated	% of Generated Mutants	No.of Killed Mutants	FDC of Mutation Operations
OIL	1	0.76%	MR1: 1 MR2: 0 MR3: 1 MR4: 1 MR5: 0 MR6: 0	MR1: 100% MR2: 0% MR3: 100% MR4: 100% MR5: 0% MR6: 0%
RIL	1	0.76%	MR1: 0 MR2: 0 MR3: 0 MR4: 0 MR5: 0 MR6: 1	MR1: 0% MR2: 0% MR3: 0% MR4: 0% MR5: 0% MR6: 100%
SIR	2	1.53%	MR1: 1 MR2: 1 MR3: 1 MR4: 1 MR5: 1 MR6: 2	MR1: 50% MR2: 50% MR3: 50% MR4: 50% MR5: 50% MR6: 100%
SDL	6	4.61%	MR1: 1 MR2: 1 MR3: 1 MR4: 1 MR5: 1 MR6: 1	MR1: 16.6% MR2: 16.6% MR3: 16.6% MR4: 16.6% MR5: 16.6% MR6: 16.6%

TABLE 6.7: Effectiveness of Mutation Operators used in Dilation and erosion

Mutation Operators	No. of Mutants Generated	% of Generated Mutants	No.of Killed Mutants	FDC of Mutation Operations
ZIL	1	0.76%	MR1: 1	MR1: 100%
			MR2: 1	MR2: 100%
			MR3: 1	MR3: 100%
			MR4: 1	MR4: 100%
			MR5: 1	MR5: 100%
			MR6: 1	MR6: 100%

In the literature, the authors [27] have used only one mutation operator for the generation of mutants and created only 33 mutants whereas, we have used eight mutation operators and produced 130 mutants in total. It is observed from Table 6.7 that AOR has generated maximum number of mutants, i.e., 95 and killed more than 50 mutants in each MR which is the second highest mutation score. ROR operator has generated 20 mutants but it does not kill a significant number of mutants. OIL, RIL, and ZIL operators have produced only one mutant though many of the MRs achieves a mutation score of 100%. The percentage of mutants killed and mutants generated by each mutation operator is depicted in Figure 6.7 and Figure 6.8.

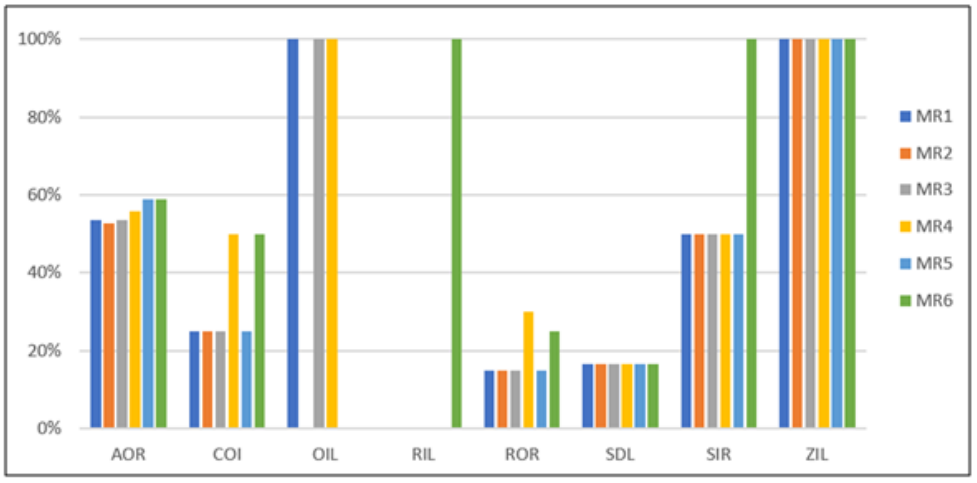

FIGURE 6.7: Percentage of Killed Mutants Used in Dilation and Erosion.

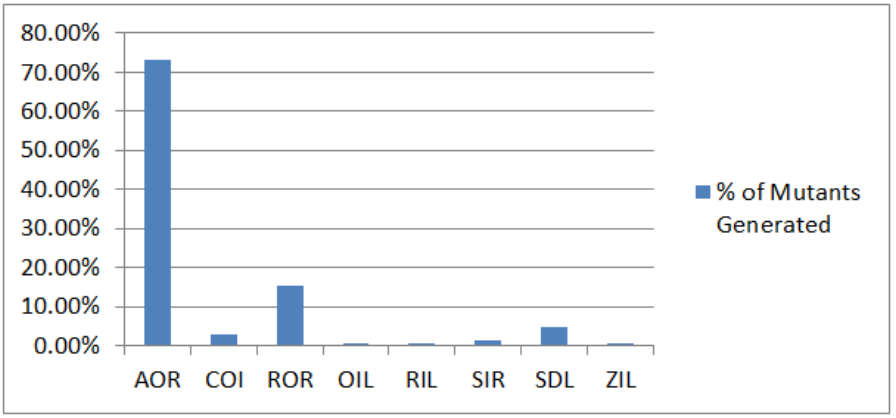

FIGURE 6.8: Percentage of Generated Mutants used in Dilation and Erosion.

It is observed from Figure 6.7 and Figure 6.8 that AOR operator is the most effective operator because it has generated maximum number of mutants i.e., 95 and its fault detection rate is greater than 50% against each MR. Though FDC of ZIL operator is 100% but it has generated only one mutant and having a percentage of 0.76. ROR has a better percentage to generate the mutants i.e., 15% and also a mutation score lies between 15 to 30%.

It is concluded from this section that AOR operator is the most effective operator in terms of mutants killed and mutants generated in both the subject programs of edge detection and dilation and erosion.

6.3 Effectiveness of Metamorphic Relations

The effectiveness of MRs is determined through mutation testing. FDR depicted in Equation 4.1 shows the effectiveness of each MR. We have assessed the effectiveness of existing MRs (edge detection and dilation and erosion) and proposed MRs (dilation and erosion).

6.3.1 Effectiveness of Edge Detection MRs

Fault detection rate (mutation score) defines the strength of each MR. FDR is calculated through mutation testing. The FDR of edge detection MRs is given in

Table 6.8.

TABLE 6.8: Fault Detection Rate of Edge Detection MRs

MR	Total No. of Mutants	No. of Killed Mutants	FDR(%)
MR_1	162	124	76.54%
MR_2	162	126	77.77%
MR_3	162	112	69.13%
MR_4	162	112	69.13%

We have generated a total of 162 mutants for edge detection manually by introducing one fault at a time. It is observed from Table 6.8 that MR_2 has killed maximum number of mutants i.e., 126 followed by MR_1 which has killed 124 mutants. MR_3 and MR_4 have killed same number of mutants i.e., 112 mutants each. Figure 6.9 shows the FDR (in percentage) of each MR.

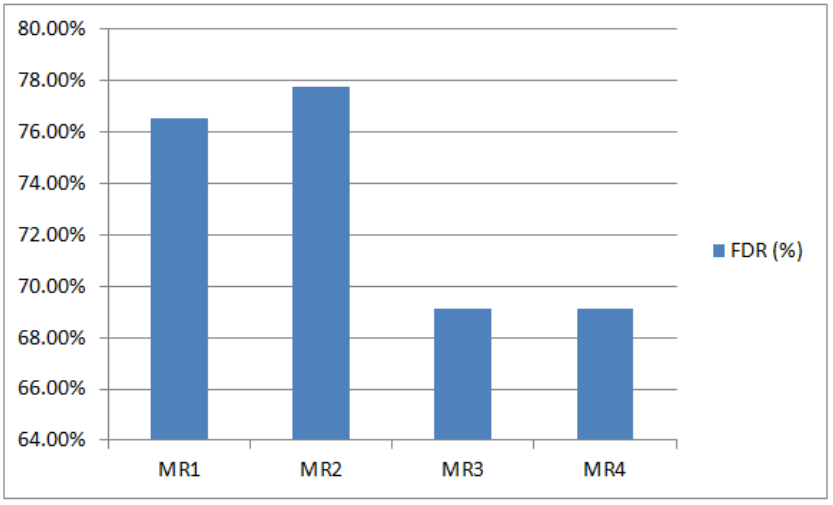

FIGURE 6.9: Graphical Representation of FDR of Edge Detection MRs.

Figure 6.9 shows that MR_2 is the most effective MR having FDR of 77.77% followed by MR_1 with FDR of 76.54%. MR_3 and MR_4 have same FDR of 69.13% each.

6.3.2 Effectiveness of Dilation and Erosion MRs

We have also assessed the effectiveness of morphological image operations (dilation and erosion).The FDR of existing operations of dilation and erosion using our proposed framework are given in Table 6.9.

TABLE 6.9: Fault Detection Rate of Dilation and Erosion MRs

MR	Total No. of Mutants	No. of Killed Mutants	FDR (%)
R_1	130	70	53.84%
R_2	130	68	52.30%
R_3	130	60	46.15%
R_4	130	67	51.53%
R_5	130	52	40.00%
R_6	130	45	34.61%
R_7	130	68	52.30%
R_8	130	65	50.00%

We have generated a total of 130 mutants for dilation and erosion operation. Table 6.9 shows that R_1 has killed maximum number of mutants i.e., 70 followed by R_2 and R_7 with 68 killed mutants each. R_6 has killed least number of mutants i.e., 45. The FDR of dilation and erosion MRs are depicted in Figure 6.10

After calculating the mutation score, it is observed from Figure 6.10 that R_1 has the highest FDR of 53.84% thus making R_1 most effective MR. R_2 and R_7 have second highest FDR of 52.30% each. R_6 has the lowest FDR of 34.61% thus

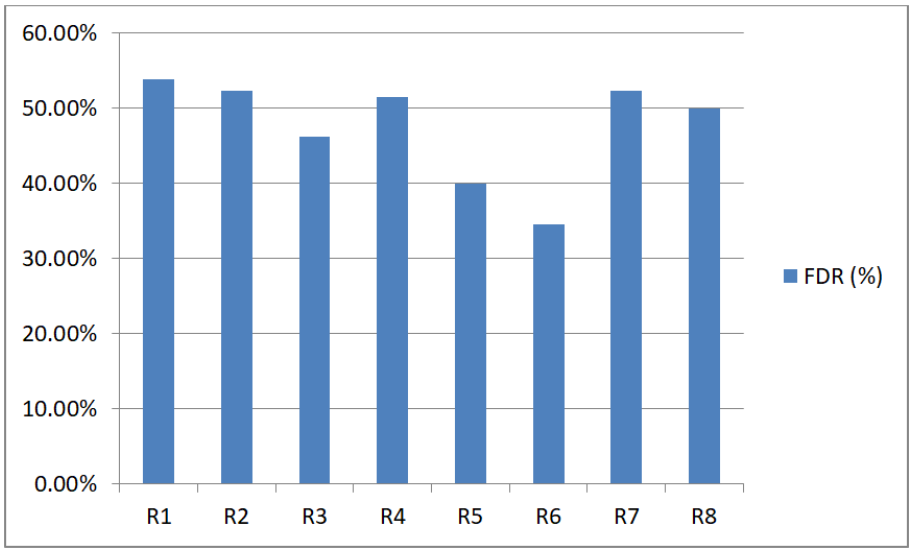

FIGURE 6.10: Graphical Representation of FDR of Dilation and Erosion MRs.

making this MR least effective. The FDR of remaining MRs are neither too high nor too low thus making them to identify some of the faults effectively.

6.3.3 Effectiveness of Proposed MRs

For the dilation and erosion operation, we have suggested six MRs (two specific and four general). We have also assessed the effectiveness of our proposed MRs using our proposed framework. The FDR of proposed MRs are given in Table 6.10.

TABLE 6.10: Fault Detection Rate of Proposed MRs

MR	Total No. of Mutants	No. of Killed Mutants	FDR(%)
MR_1	130	59	45.38%
MR_2	130	57	48.46%
MR_3	130	59	43.84%
MR_4	130	65	50.00%

TABLE 6.10: Fault Detection Rate of Proposed MRs

MR	Total No. of Mutants	No. of Killed Mutants	FDR(%)
MR_5	130	63	48.46%
MR_6	130	68	52.30%

Table 6.10 shows that we have generated a total of 130 mutants. It is observed that MR_6 has the highest FDR by killing 68 mutants followed by MR_4 that has killed 65 mutants. MR_1 and MR_3 have killed 59 mutants each. The FDR of MR_2 is the lowest because it has killed 57 mutants. The graphical representation of FDR of proposed MRs is shown in Figure 6.11.

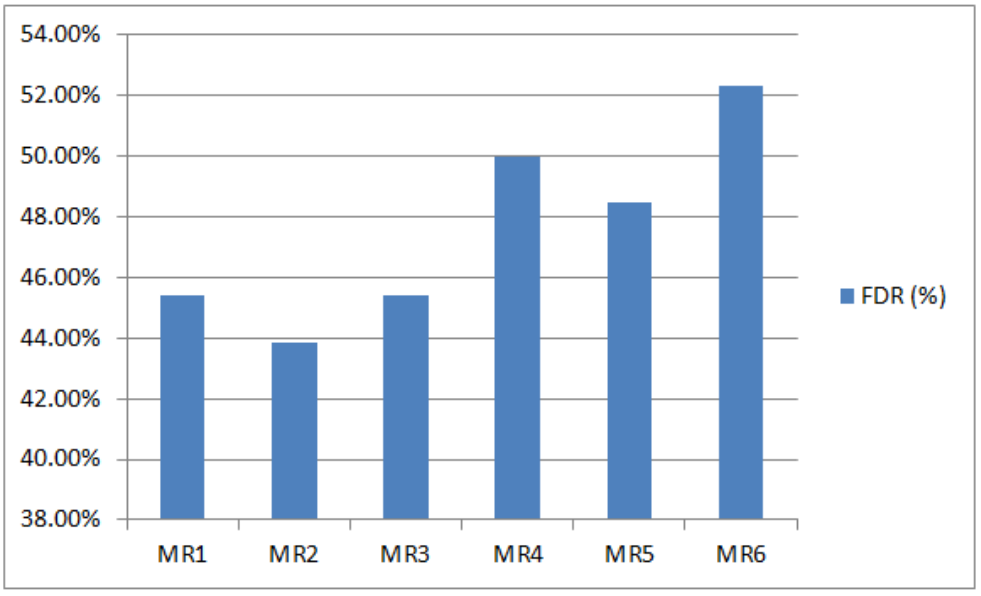

FIGURE 6.11: Graphical Representation of FDR of Proposed MRs.

It is observed from Figure 6.11 that the FDR of our proposed MRs are neither too high nor too low but are significant enough to find the violations in all the respective MRs. However, the most effective MR among the proposed MRs is MR_6 (image translation) which has the highest FDR of 52.30% followed by MR_4 (transposition in erosion operation) with FDR of 50%. MR_2 (counter clock-wise rotation at 90 degree in erosion operation) is the least effective with FDR of 43.84%.

6.4 Comparison of Proposed Framework with Existing Techniques

In this section, we have compared the results of our proposed framework with the existing techniques. The proposed framework is compared with the existing techniques where edge detection and dilation and erosion MRs are used. We have compared the results of our proposed framework with [56] and [27]. Table 6.11 shows the statistics of existing techniques and proposed framework.

TABLE 6.11: Statistics of Existing Techniques and Proposed Framework

Ref Papers	SUT	No. of Test cases	Mutation operators	No. of Mutants Generated	Lang.
[56]	Edge Detection	30 test cases are selected randomly.	LOR, ROR	31	C
[27]	Dilation and Erosion	Randomly Selected	ROR	33	Mex C
Proposed Framework	Edge Detection and Dilation and Erosion	95 MRI brain images are selected	AOD, AOR, ROR, COI, ZIL, SIR, SDL, OIL, RIL	162: Edge detection 130: dilation and erosion	Python

According to the statistics given in Table 6.11, Sim et al. have selected 30 images as source test cases from different image libraries given in [56]. The images used by the authors are very limited and not diverse in nature. The Kodak site has 24 images and the image compression site has only 15 images. All the images

have same bit depth of 24 and resolution of 96dpi. All the images have only two dimensions such as 768 by 512 or 512 by 768 (Kodak site) and a single dimension of 700 by 525. Jameel et al. have not mentioned the source as well as the number of test cases selected for their experiments. We have used the data set of MRI brain images taken from Kaggle.com. The complete source is given section 5.1.3. The dataset compromises of 3000 images having diverse image properties. From 3000 images, we have selected 95 images using equivalence class testing and code coverage.

The authors [56] have used two mutation operators and generated only 31 mutants whereas the authors in [27] have used only one operator and generated 33 mutants. Whereas, we have used nine mutation operators in edge detection program and eight operators in dilation and erosion program to generate 162 and 130 mutants respectively.

6.4.1 Comparison Results of Edge Detection

We have compared the results of our proposed framework for thee evaluation of MRs with [56]. The FDR results are compared against each MR.The comparison results are given in Table 6.12.

Table 6.12: Comparison of Existing Technique and Proposed Framework

MR	FDR of Existing Technique	FDR of Proposed Framework
MR_1	45%	76.54%
MR_2	77%	77.77%
MR_3	68%	69.13%
MR_4	45%	69.13%

Table 6.12 shows that in existing technique [56] MR_2 has the highest FDR followed by MR_3. Whereas in proposed framework MR_2 has the highest FDR followed by MR_1. In existing technique, the FDR of MR_1 and MR_4 are same i.e., 45% whereas in proposed framework the FDR of MR_3 and MR_4 are same i.e., 69.13%. The graphical representation of both the techniques are depicted in Figure 6.12.

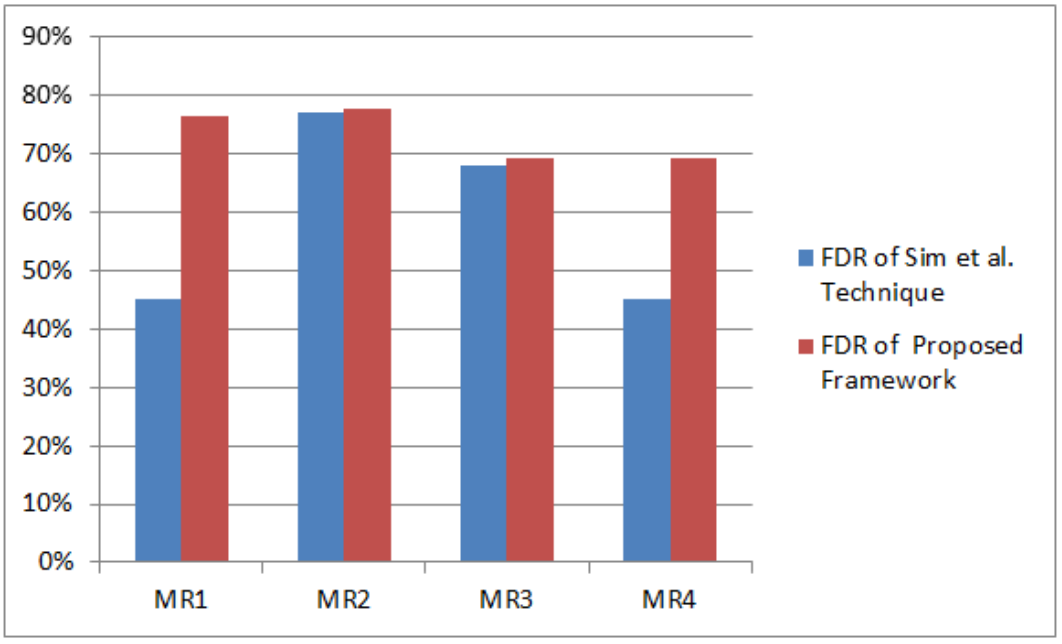

FIGURE 6.12: FDR of Existing Technique and Proposed Framework.

Figure 6.12 shows that in existing technique, MR_1 and MR_4 have achieved FDR of 45% each. The FDR of MR_2 is highest, i.e., 77% followed by MR_3 with FDR of 68%. In proposed framework, the FDR of MR_1 and MR_4 is far better than the existing technique, i.e., 76.54% and 69.13% respectively. The FDR of MR_2 and MR_3 is also improved by 1%.

6.4.2 Comparison Results of Dilation and Erosion

The authors in this paper [27], have evaluated eight MRs of dilation and erosion operations. We have also evaluated the same eight MRs using our proposed framework. The FDR results are checked against each MR. The effectiveness of each MR is checked using it's FDR value. The comparison results of existing MRs of dilation and erosion are given in Table 6.13.

TABLE 6.13: Comparison of Existing Technique and Proposed Framework

MRs	FDR of Existing Technique	FDR of Proposed Framework
R_1	15%	53.84%
R_2	58%	52.30%
R_3	97%	46.15%
R_4	51%	51.53%
R_5	58%	40%
R_6	18%	34.61%
R_7	73%	52.30%
R_8	21%	50%

Table 6.13 shows that out of eight MRs, four MRs i.e., R_1, R_4, R_6, and R_8 have improved FDR using our proposed framework. The FDR of R_1 (53.84%), R_6 (34.61%), and R_8 (50%) are far improved than the existing technique having FDR of 15%, 18%, and 21% respectively. In existing technique, R_2, MR_3, R_5, and R_7 have high FDR i.e., 58%, 97%, 50% and 73% because they have used only one mutation operator and consider only one type of faults. The comparison results are depicted in Figure 6.13

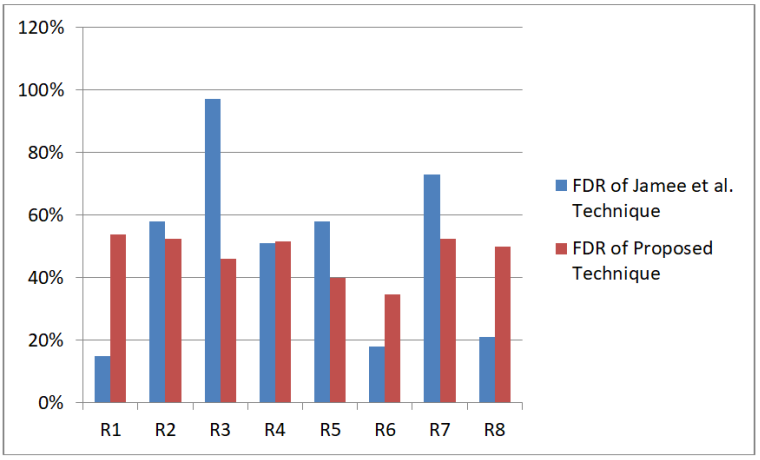

FIGURE 6.13: Graphical Representation of FDR of Existing Technique and Proposed Framework.

It is observed from Figure 6.13 that in existing technique, R_3 has the highest FDR of 97% followed by R_7 having FDR of 73%. The FDR of $R_1, R_6, and R_8$ is very low i.e., 15%, 18%, and 21% respectively. In proposed framework, the FDR of all the MRs are moderate, neither too high nor too low thus making them effective to find the violations in all respective MRs. In proposed framework, R_1 is improved by 39%, R_4 is improved by 0.5%, R_6 is improved by 17%, and R_8 is improved by 29%.

6.4.3 Comparison Results of Proposed MRs with Existing MRs for Dilation and Erosion

In this section, we have compared the results of our proposed MRs with the existing MRs of dilation and erosion operations. As described earlier, we have proposed six new MRs for morphological image operations (dilation and erosion) whereas there are eight existing MRs of dilation and erosion operations in literature. By using mutation testing, we have used eight mutation operators for the evaluation of dilation and erosion MRs and have generated 130 mutants in total whereas in existing technique, the authors have generated only 33 mutants against each mutation operator. The number of mutants which are generated against each mutation operator is depicted in Table 6.15.

TABLE 6.14: Mutants Generated against each Mutation Operator

Mutation Operators	**No. of Mutants Generated**	**Mutant Labels**
AOR	95	$mt_1, mt_2, mt_3, mt_4, mt_5, ..., mt_{95}$
COI	4	$mt_{96}, mt_{97}, mt_{98}, mt_{99}$
ROR	20	$mt_{100}, mt_{101}, mt_{102}, ...mt_{119}$

TABLE 6.15: Mutants Generated against each Mutation Operator

Mutation Operators	**No. of Mutants Generated**	**Mutant Labels**
OIL	1	mt_{120}
RIL	1	mt_{121}
SIR	2	mt_{122}, mt_{123}
SDL	6	$mt_{124}, mt_{125}, ..., mt_{128}, mt_{129}$
ZIL	1	mt_{130}

Table 6.15 shows 130 mutants ($mt_1, mt_2, ..., mt_{130}$) generated by eight mutation operators. The mutant is denoted by "mt". Now Table 6.16 shows the existing MRs and the mutants killed by each MR.

TABLE 6.16: Mutants Killed by Existing MRs

MRs	**Mutants Killed**
R_1	$mt_{11}, ..., mt_{30}, mt_{36}, mt_{37}, mt_{39}, mt_{40}, ..., mt_{43}, mt_{45}, mt_{47},$ $..., mt_{49}, mt_{54}, mt_{76}, ..., mt_{85}, mt_{87}, ..., mt_{90}, mt_{94}, ..., mt_{97},$ $mt_{99}, mt_{101}, mt_{102}, mt_{104}, ..., mt_{107}, mt_{109}, mt_{116}, mt_{117},$ $mt_{118}, mt_{120}, mt_{122}, mt_{123}, mt_{125}, ..., mt_{130}$
R_2	$mt_1, ..., mt_5, mt_{21}, ..., mt_{30}, mt_{37}, mt_{38}, mt_{40}, mt_{44}, mt_{46},$ $..., mt_{55}, mt_{77}, ..., mt_{81}, mt_{83}, ..., mt_{96}, mt_{98}, mt_{99}, mt_{101},$ $mt_{102}, mt_{104}, mt_{106}, mt_{107}, mt_{109}, mt_{116}, ..., mt_{118}, mt_{120},$ $mt_{121}, mt_{122}, mt_{125}, ..., mt_{130}$

TABLE 6.16: Mutants Killed by Existing MRs

MRs	**Mutants Killed**
R_3	$mt_{16}, ..., mt_{30}, mt_{76}, ..., mt_{81}, mt_{83}, ..., mt_{91}, mt_{93}, ..., mt_{99}, mt_{101}, mt_{102}, mt_{104}, ..., mt_{107}, mt_{109}, mt_{111}, mt_{112}, mt_{113}, mt_{116}, ..., mt_{117}, mt_{118}, mt_{120}, ..., mt_{123}, mt_{125}, ..., mt_{130}$
R_4	$mt_1, ..., mt_5, mt_{21}, ..., mt_{30}, mt_{37}, mt_{38}, mt_{40}, mt_{44}, mt_{46}, ..., mt_{55}, mt_{77}, ..., mt_{81}, mt_{83}, ..., mt_{96}, mt_{98}, mt_{99}, mt_{101}, mt_{102}, mt_{104}, mt_{106}, mt_{107}, mt_{109}, mt_{116}, mt_{117}, mt_{118}, mt_{121}, mt_{122}, mt_{125}, ..., mt_{130}$
R_5	$mt_{16}, ..., mt_{30}, mt_{77}, mt_{78}, mt_{80}, mt_{81}, mt_{83}, ..., mt_{88}, mt_{90}, mt_{94}, mt_{96}, ..., mt_{99}, mt_{101}, mt_{102}, mt_{104}, ..., mt_{107}, mt_{109}, mt_{111}, mt_{112}, mt_{113}, mt_{116}, mt_{117}, mt_{118}, mt_{120}, ..., mt_{123}, mt_{125}, ..., mt_{130}$
R_6	$mt_{16}, ..., mt_{30}, mt_{77}, mt_{78}, mt_{80}, mt_{81}, mt_{83}, ..., mt_{88}, mt_{90}, mt_{94}, mt_{96}, ..., mt_{99}, mt_{101}, mt_{102}, mt_{104}, ..., mt_{107}, mt_{109}, mt_{111}, mt_{112}, mt_{113}, mt_{116}, mt_{117}, mt_{118}, mt_{120}, ..., mt_{123}, mt_{125}, ..., mt_{130}$
R_7	$mt_1, ..., mt_5, mt_{21}, ..., mt_{30}, mt_{37}, mt_{38}, mt_{40}, mt_{44}, mt_{46}, ..., mt_{55}, mt_{77}, ..., mt_{81}, mt_{83}, ..., mt_{96}, mt_{98}, mt_{99}, mt_{101}, mt_{102}, mt_{104}, mt_{106}, mt_{107}, mt_{109}, mt_{116}, mt_{117}, mt_{118}, mt_{120}, mt_{121}, mt_{122}, mt_{125}, ..., mt_{130}$
R_8	$mt_1, ..., mt_5, mt_{21}, ..., mt_{30}, mt_{37}, mt_{38}, mt_{40}, mt_{44}, mt_{46}, ..., mt_{55}, mt_{77}, ..., mt_{81}, mt_{83}, ..., mt_{96}, mt_{98}, mt_{99}, mt_{101}, mt_{102}, mt_{104}, mt_{106}, mt_{107}, mt_{109}, mt_{116}, mt_{117}, mt_{118}, mt_{122}, mt_{125}, ..., mt_{129}$

Table 6.16 shows the existing MRs of dilation and erosion and the mutants killed by these MRs respectively. To summarize Table 6.16, we have generated another table, Table 6.17, that shows the statistics of total number of mutants killed by these MRs as well as total number of alive mutants(the mutants that are not killed by any of the MR). Hence, it is concluded from the above table that the ratio of killed mutants are greater than the alive mutants. The alive mutants can be killed by some other MRs or we need some new mutation operators that will identify the faults not identified by the respective mutation operators used in the existing literature.

TABLE 6.17: Killed and Alive Mutants in Existing MRs

Mutants Killed	**Mutants Alive**
$mt_1, ..., mt_5, mt_{11}, ..., mt_{30}, mt_{36}, ...,$ $mt_{55}, mt_{76}, ..., mt_{99}, mt_{101}, mt_{102},$ $mt_{104}, ..., mt_{107}, mt_{109}, mt_{111}, ...,$ $mt_{113}, mt_{116}, ..., mt_{123}, mt_{125}, ...,$ mt_{130}	$mt_6, ..., mt_{10}, mt_{31}, ..., mt_{35}, mt_{56},$ $..., mt_{75}, mt_{100}, mt_{103}, mt_{108}, mt_{110},$ $mt_{114}, mt_{115}, mt_{124}$

After calculating the number of mutants killed by the existing MRs of dilation and erosion, we have also calculated the number of mutants killed by the proposed MRs. Table 6.19 shows the proposed MRs (dilation and erosion) and the mutants killed by each of the MR.

TABLE 6.18: Mutants Killed by Proposed MRs

MRs	**Mutants Killed**
MR_1	$mt_1, ..., mt_5, mt_{11}, ..., mt_{20}, mt_{36}, ..., mt_{55}, mt_{76}, mt_{77},$ $mt_{79}, ..., mt_{82}, mt_{84}, ..., mt_{87}, mt_{89}, ..., mt_{92}, mt_{94}, ..., mt_{96},$ $mt_{105}, mt_{111}, mt_{116}, mt_{120}, mt_{123}, mt_{129}, mt_{130}$
MR_2	$mt_1, ..., mt_5, mt_{11}, mt_{12}, ..., mt_{19}, mt_{36}, ...mt_{55}, mt_{76}, mt_{77},$ $mt_{79}, ..., mt_{82}, mt_{84}, ..., mt_{87}, mt_{89}, ..., mt_{91}, mt_{94}, ..., mt_{96},$ $mt_{105}, mt_{111}, mt_{116}, mt_{123}, mt_{129}, mt_{130}$

TABLE 6.19: Mutants Killed by Proposed MRs

MRs	**Mutants Killed**
MR_3	$mt_1, ..., mt_5, mt_{11}, ..., mt_{20}, mt_{36}, ..., mt_{55}, mt_{76}, mt_{77}, mt_{79}, ..., mt_{82}, mt_{84}, ..., mt_{87}, mt_{89}, ..., mt_{92}, mt_{94}, ..., mt_{96}, mt_{105}, mt_{111}, mt_{116}, mt_{120}, mt_{123}, mt_{129}, mt_{130}$
MR_4	$mt_8, ..., mt_{45}, mt_{76}, mt_{79}, ..., mt_{82}, mt_{84}, ..., mt_{87}, mt_{89}, ..., mt_{92}, mt_{94}, ..., mt_{96}, mt_{99}, mt_{101}, mt_{102}, mt_{105}, mt_{111}, mt_{116}, mt_{118}, mt_{120}, mt_{122}, mt_{129}, mt_{130}$
MR_5	$mt_{16}, ..., mt_{55}, mt_{76}, mt_{77}, mt_{79}, ..., mt_{83}, mt_{85}, ..., mt_{87}, mt_{89}, ..., mt_{93}, mt_{95}, mt_{96}, mt_{102}, mt_{105}, mt_{116}, mt_{123}, mt_{129}, mt_{130}$
MR_6	$mt_3, mT_6, mt_8, ..., mt_{45}, mt_{76}, mt_{77}, mt_{79}, ..., mt_{82}, mt_{84}, ..., mt_{87}, mt_{89}, ..., mt_{92}, mt_{94}, ..., mt_{96}, mt_{99}, mt_{101}, mt_{102}, mt_{105}, mt_{111}, mt_{116}, mt_{121}, ..., mt_{123}, mt_{129}, mt_{130}$

The above table shows the number of killed mutants by each MR. But, it is difficult to see the number of alive mutants from the above table. So, the statistics of Table 6.19 is summarized in Table 6.20 which shows the number of killed mutants as well as alive mutants in proposed MRs.

TABLE 6.20: Killed and Alive Mutants in Existing MRs

Mutants Killed	**Mutants Alive**
$mt_1, ..., mt_6, mt_8, ..., mt_{10}, ..., mt_{55}, mt_{76}, mt_{77}, mt_{79}, ..., mt_{87}, mt_{89}, ..., mt_{96}, mt_{99}, mt_{101}, mt_{102}, mt_{105}, mt_{111}, mt_{116}, mt_{118}, mt_{120}, ..., mt_{123}, mt_{129}, mt_{130}$	$mt_7, mt_{56}, ..., mt_{75}, mt_{78}, mt_{88}, mt_{97}, mt_{98}, mt_{100}, mt_{103}, mt_{104}, mt_{106}, ..., mt_{110}, mt_{112}, ..., mt_{115}, mt_{117}, mt_{119}, mt_{124}, ..., mt_{128}$

Table 6.17 and 6.20 shows the number of mutants killed and mutants alive by existing MRs and proposed MRs. In mutation testing, we have used eight mutation operators for the evaluation of dilation and erosion MRs. In AOR operator, we have used six type of faults such as addition (+), subtraction (-), multiplication (*), division (/), exponent/power (**), and floor division (//). It is observed from Table 6.17 and 6.20 that there are nine arithmetic faults ($mt_6, mt_8, mt_9, mt_{10}, mt_{31}, mt_{32}, mt_{33}, mt_{34}, mt_{35}$) which are identified by the proposed MRs and are not identified by any of the existing MR. Among these faults, eight faults ($mt_8, mt_9, mt_{10}, mt_{31}, mt_{32}, mt_{33}, mt_{34}, mt_{35}$) are identified by MR_4, five faults ($mt_{31}, mt_{32}, mt_{33}, mt_{34}, mt_{35}$) are identified by MR_5, and nine faults ($mt_6, mt_8, mt_9, mt_{10}, mt_{31}, mt_{32}, mt_{33}, mt_{34}, mt_{35}$) are identified by MR_6. So, we can say that MR_4, MR_5, and MR_6 are more effective operators as compared to MR_1, MR_2, and MR_3 because they have identified the additional faults not identified by the any of the existing MRs.

We have observed the presence of alive mutants in both existing and proposed MRs. It is also observed that by combining both the MRs, the total number of alive mutants was reduced. The remaining alive mutants are: $mt_7, mt_{56}, mt_{57}, ..., mt_{75}$, $mt_{100}, mt_{103}, mt_{108}, mt_{110}, mt_{114}, mt_{115}, mt_{124}$. Hence, it is concluded that the proposed MRs of dilation and erosion complement the existing MRs effectively because the proposed MRs are able to detect those faults which are not identified by any of the existing MRs of dilation and erosion operations.

6.5 Composition of Metamorphic Relations

Composition of MRs is done for the construction of new MRs. The composited relation contains all the properties of the original MR [28]. Composition of MR is cost effective as it reduces the number of executions considerably. For example, if we use MR_1 and MR_2 for testing the program P, then we need four test cases (two source test cases and two follow-up test cases). After composing the two MRs we generate and execute three test cases (one common source test case plus two follow-up test cases).

For the composition of MRs, we have selected four MRs of edge detection [56]. The composition results after combining the two MRs are given in Table 6.21

TABLE 6.21: Composition of Two MRs

S.No	MR_x	MR_y	MR_{xy}
1	MR_1: 76.54	MR_2: 77.77	MR_{12}: 77.77
2	MR_1: 76.54	MR_3: 69.13	MR_{13}: 69.13
3	MR_1: 76.54	MR_4: 69.13	MR_{14}: 69.13
4	MR_2: 77.77	MR_1: 76.54	MR_{21}: 76.54
5	MR_2: 77.77	MR_3: 69.13	MR_{23}: 69.13
6	MR_2: 77.77	MR_4: 69.13	MR_{24}: 69.13
7	MR_3: 69.13	MR_1: 76.54	MR_{31}: 76.54
8	MR_3: 69.13	MR_2: 77.77	MR_{32}: 77.77
9	MR_3: 69.13	MR_4: 69.13	MR_{34}: 69.13
10	MR_4: 69.13	MR_1: 76.54	MR_{41}: 76.54
11	MR_4: 69.13	MR_2: 77.77	MR_{42}: 77.77
12	MR_4: 69.13	MR_3: 69.13	MR_{43}: 69.13

It is observed from Table 6.21 that at one execution (MR_{12}), the value of composite

MR is equal to the value of greatest component MR and when we reverse the execution i.e., MR_{21}, then the value of composite MR is equal to the value of lowest component MR. We have also composed three MRs. The process of composing three MRs is given below:

Composed MR (MR_1 and MR_2):

$$MR_{12} : T(E(C(Im))) = E(T(C(Im)))$$

Third MR:

$$MR_3 : M_x(E(Im)) = E(M_x(Im))$$

After Composition:

$$MR_{123} : M_x(E(T(C(Im)) = E(M_x(T(C(Im))$$

Where,

T(C(Im)) = follow-up test case of MR_{12} as well as source test case of MR_3.

$M_x(T(C(Im))$ = follow-up test case of MR_3.

We have four MRs of edge detection as discussed earlier. So, we have made all possible combinations to composed the MRs by composing the three MRs manually. We have made 24 new MRs from composing three MRs. The results are given in Table 6.24.

TABLE 6.22: Composition of Three MRs

S.No	MR_{xy}	MR_z	MR_{xyz}
1	MR_{12}: 77.77	MR_3: 69.13	MR_{123}: 69.13
2	MR_{12}: 77.77	MR_4: 69.13	MR_{124}: 69.13
3	MR_{13}: 69.13	MR_2: 77.77	MR_{132}: 77.77

TABLE 6.23: Composition of Three MRs

S.No	MR_{xy}	MR_z	MR_{xyz}
4	MR_{13}: 69.13	MR_4: 69.13	MR_{134}: 69.13
5	MR_{14}: 69.13	MR_2: 77.77	MR_{142}: 77.77
6	MR_{14}: 69.13	MR_3: 69.13	MR_{143}: 69.13
7	MR_{21}: 76.54	MR_3: 69.13	MR_{213}: 69.13
8	MR_{21}: 76.54	MR_4: 69.13	MR_{214}: 69.13
9	MR_{23}: 69.13	MR_1: 76.54	MR_{231}: 76.54
10	MR_{23}: 69.13	MR_4: 69.13	MR_{234}: 69.13
11	MR_{24}: 69.13	MR_1: 76.54	MR_{241}: 76.54
12	MR_{24}: 69.13	MR_3: 69.13	MR_{243}: 69.13
13	MR_{31}: 76.54	MR_2: 77.77	MR_{312}: 77.77
14	MR_{31}: 76.54	MR_4: 69.13	MR_{314}: 69.13
15	MR_{32}: 77.77	MR_1: 76.54	MR_{321}: 76.54
16	MR_{32}: 77.77	MR_4: 69.13	MR_{324}: 69.13
17	MR_{34}: 69.13	MR_1: 76.54	MR_{341}: 76.54

TABLE 6.24: Composition of Three MRs

S.No	MR_{xy}	MR_z	MR_{xyz}
18	MR_{34}: 69.13	MR_2: 77.77	MR_{342}: 77.77
19	MR_{41}: 76.54	MR_2: 77.77	MR_{412}: 77.77
20	MR_{41}: 76.54	MR_3: 69.13	MR_{413}: 69.13
21	MR_{42}: 77.77	MR_1: 76.54	MR_{421}: 76.54
22	MR_{42}: 77.77	MR_3: 69.13	MR_{423}: 69.13
23	MR_{43}: 69.13	MR_1: 76.54	MR_{431}: 76.54
24	MR_{43}: 69.13	MR_2: 77.77	MR_{432}: 77.77

Table 6.24 shows that the value of composite MR (MR_{xyz}) is not less than the lowest value of component MR (MR_z) and the composite MR (MR_{xy}). For example, in case of MR_{214}, the value of composite MR is 69.13 which is the lowest value among the component MR (MR_4) and the composite MR (MR_{21}) but not less than the lowest MR i.e., MR_4.

In the end, we have also composed four MRs. The process of composing four MRs is given below:

Composed MR:

$$MR_{123} : M_x(E(T(C(Im)) = E(M_x(T(C(Im))$$

Fourth MR:

$$MR_4 : M_y(E(Im)) = E(M_y(Im))$$

After Composition:

$$MR_{1234} : M_y(E(M_x(T(C(Im)) = E(M_y(M_x(T(C(Im))$$

Where

$M_x(T(C(Im)) =$ follow-up test case of MR_{123} as well as source test case of MR_4.

$M_y(M_x(T(C(Im)) =$ follow-up test case of MR_4.

We have made 24 new MRs from composing four MRs. The results are given in Table 6.25.

TABLE 6.25: Composition of Four MRs

S.No	MR_{xyz}	MR_w	MR_{xyzw}
1	MR_{123}: 69.13	MR_4: 69.13	MR_{1234}: 69.13
2	MR_{124}: 69.13	MR_3: 69.13	MR_{1243}: 69.13
3	MR_{132}: 77.77	MR_4: 69.13	MR_{1324}: 69.13
4	MR_{134}: 69.13	MR_2: 77.77	MR_{1342}: 77.77
5	MR_{142}: 77.77	MR_3: 69.13	MR_{1423}: 69.13
6	MR_{143}: 69.13	MR_2: 77.77	MR_{1432}: 77.77
7	MR_{213}: 69.13	MR_4: 69.13	MR_{2134}: 69.13
8	MR_{214}: 69.13	MR_3: 69.13	MR_{2143}: 69.13
9	MR_{231}: 76.54	MR_4: 69.13	MR_{2314}: 69.13
10	MR_{234}: 69.13	MR_1: 76.54	MR_{2341}: 76.54

TABLE 6.25: Composition of Four MRs

S.No	MR_{xyz}	MR_w	MR_{xyzw}
11	MR_{241}: 76.54	MR_3: 69.13	MR_{2413}: 69.13
12	MR_{243}: 69.13	MR_1: 76.54	MR_{2431}: 76.54
13	MR_{312}: 77.77	MR_4: 69.13	MR_{3124}: 69.13
14	MR_{314}: 69.13	MR_2: 77.77	MR_{3142}: 77.77
15	MR_{321}: 76.54	MR_4: 69.13	MR_{3214}: 69.13
16	MR_{324}: 69.13	MR_1: 76.54	MR_{3241}: 76.54
17	MR_{341}: 76.54	MR_2: 77.77	MR_{3412}: 77.77
18	MR_{342}: 77.77	MR_1: 76.54	MR_{3421}: 76.54
19	MR_{412}: 77.77	MR_3: 69.13	MR_{4123}: 69.13
20	MR_{413}: 69.13	MR_2: 77.77	MR_{4132}: 77.77
21	MR_{421}: 76.54	MR_3: 69.13	MR_{4213}: 69.13
22	MR_{423}: 69.13	MR_1: 76.54	MR_{4231}: 76.54
23	MR_{431}: 76.54	MR_2: 77.77	MR_{4312}: 77.77
24	MR_{432}: 77.77	MR_1: 76.54	MR_{4321}: 76.54

It is observed from Table 6.25 that the value of composite MR (MR_{xyzw}) is not less than the lowest value of component MR (MR_w) and the composite MR (MR_{xyz}). For example, in case of MR_{214}, the value of composite MR_{4321} is 76.54 which is the lowest value among the component MR_1 and the composite MR_{432} but not less than the lowest MR i.e., MR_1. So, we have two observations from Table 6.21, 6.24, and 6.25. First, the result of composed MR is not less than the lower value of component MR and second, after the execution, the composite MR has always the value of the MR on second placeholder. Hence, it can be concluded that we have to choose that composite MR which has highest value of the MR at the second place.

6.6 MR Evaluation using SSIM

This section is derived from our published framework [85], where an improved canny edge edge detection algorithm is used to evaluate the MRs of edge detection operation using MRI brain images. We have selected all four MRs of edge detection [56], from the literature of MT. Instead of testing the conventional edge detection programs, we have assessed the accuracy of these MRs on an improved canny edge detection program, proposed for MRI brain diagnostics [55]. Conventional edge detection algorithms are already tested using the conventional software testing techniques. However, the edge detection algorithms that are proposed by medical researchers themselves are not tested by any software testing technique. So, our primary concern is to validate these type of un-tested edge detection algorithms. The proposed framework for the evaluation of MRs using SSIM is given in next section.

6.6.1 Proposed Framework

For our proposed framework, we have selected the Sari's edge detection algorithm because our data set comprises of MRI brain images and amongst all the articles, this is the latest research article to detect brain tumor in MRI images. The

flowchart of the proposed framework is shown in Figure 6.14. The basic steps of proposed framework are similar to the steps depicted in Figure 4.1 for the evaluation of MRs except the evaluation process is different. Previously in Chapter 4, the evaluation of MR is performed using mutation testing where in this methodology the evaluation is performed through SSIM, structure similarity measure that will verify the correctness of images. Here, we have discussed the issue narrated in section 2.2.2 about the evaluation of output images as it is difficult to perform a pixel by pixel comparison of two images which seem alike visually but in fact are different.

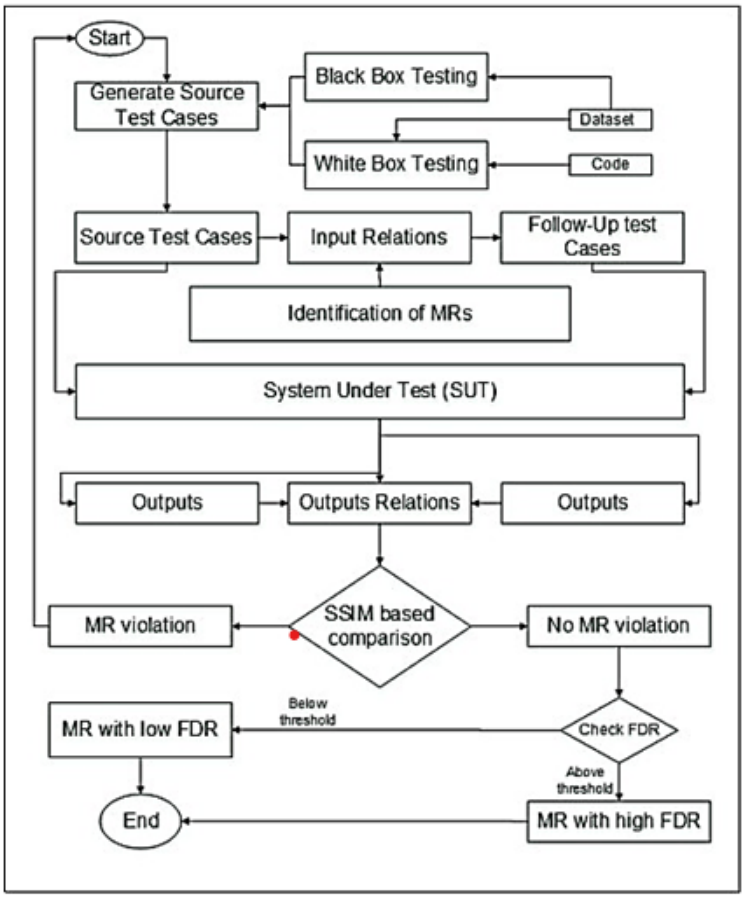

FIGURE 6.14: Flowchart of Proposed Framework.

6.6.1.1 Generation of Source Test Cases

The first step is to generate source test cases to test the algorithm from the selected dataset of MRI of brain. We have performed MT on an improved algorithm of edge

detection (as the algorithm does not have built-in functions such as conventional edge detection algorithms Canny, Sobel, Prewitt, Robert, etc.) that shows the fault detection capability of MR in terms of satisfying each relation. A solution is proposed for the generation of source test cases by using strong equivalence class testing and code coverage criterion. The classes and sub classes using strong equivalence class testing are shown in Table 5.2. The selected test cases are further checked through code coverage to ensure complete coverage. The coverage results are shown in Table 6.4.

6.6.1.2 Identification of Metamorphic Relations

In MT, testers define MRs which are used to generate new test cases (referred as follow-up test cases) from the available test cases (referred as original/source test cases) [116]. The key role of MR is to generate new test cases and to verify test results in the absence of a test oracle [22]. For verification of test results, there are only two possible outcomes: a high FDR or a low FDR. Greater FDR shows higher fault detection capabilities and vice versa. We have selected four MRs [56], and we have assessed the FDR of these four MRs through an edge detection program using edge detection as a SUT.

6.6.1.3 Generation of Follow-Up Test Cases

Follow-up test cases are generated from source test cases using MRs [82]. After the generation of source test cases through our proposed criterion, follow-up test cases are generated through source test cases and MRs. Source test cases and follow-up test cases are given to an edge detection program used as SUT to generate outputs, respectively. In the MT process, first, the source test cases are given to the original program. The outputs of source test cases are recorded as O1. Then, the follow-up test cases are given to the same original program. The outputs of follow-up test cases are also recorded as O2. The outputs of both source and follow-up test cases are compared and if (O1, O2) satisfy their related MR for all the test cases then it shows that the related MR is satisfiable.

6.6.1.4 SSIM Based Output Comparison

Image quality assessment is an important parameter of assessing the quality between two images. Usually, MSE (mean square error) and PSNR (peak signal-to-noise ratio) are used to assess the quality of images by giving absolute errors. However, these two measures are not normalized, and therefore, it is difficult to understand them. Recently, two new metrics, SSIM and FSIM (feature similarity index measure), have been developed to check the structure and feature similarity between two images [117].

SSIM calculates the similarity of two iamges which is a value betweeen +1 and -1. We have used SSIM to compare the output of source and follow-up test cases because SSIM compares the image based on luminance, contrast, and structure, respectively. We are using a dataset of MRI brain images where correct identification of luminance, contrast and structure helps in the identification of edges and lesions which helps in the correct diagnosis process. SSIM has also become a default measure in the field of IP [118].

If the value of SSIM is 0, then both images are different, but if the value of SSIM is 1, then the images are exactly similar. To check the satisfaction of MR, we have compared the outputs of all the source and follow-up test cases and set a threshold value of 0.95. If the value of SSIM is below this threshold values, then MR is in violation. Afterwards, FDR is calculated for each MR. The formula to calculate the FDR using SSIM based comparison is given in Equation 6.2.

$$\text{FDR} = \frac{\text{Number of test cases violating MR} \times 100}{\text{Total number of test cases}} \tag{6.2}$$

Where,

MR: Metamorphic Relation

6.6.2 Results and Discussion

We have checked the outputs of source and follow-up test cases for each MR against the three image types: T1-weighted images, T2-weighted images, and flair images

by using SSIM. The SSIM value of each MR on 33 test cases (of 95 total test cases) of T1-weighted images are shown in Table 6.27 as below:

TABLE 6.26: SSIM Value of T1 Weighted Images

Test Cases (TC)	MR_1	MR_2	MR_3	MR_4
TC1	0.93	0.99	0.98	0.99
TC2	0.96	0.96	1	0.96
TC3	0.98	0.99	0.98	0.93
TC4	0.95	0.98	0.97	0.92
TC5	0.99	0.99	0.99	0.99
TC6	0.81	0.99	0.95	0.91
TC7	0.92	0.98	0.93	0.94
TC8	0.99	1	0.98	0.92
TC9	0.99	1	0.98	0.96
TC10	0.93	0.98	0.94	0.84
TC11	0.83	0.82	1	0.82
TC12	0.92	0.97	0.93	0.95
TC13	0.98	0.94	0.94	0.94

TABLE 6.26: SSIM Value of T1 Weighted Images

Test Cases (TC)	MR_1	MR_2	MR_3	MR_4
TC14	0.79	0.97	0.97	0.89
TC15	0.89	0.96	0.95	0.84
TC16	0.94	0.98	0.92	0.9
TC17	0.94	0.82	0.91	0.89
TC18	0.98	0.99	0.98	0.97
TC19	0.94	0.98	0.94	0.95
TC20	0.98	0.99	0.96	0.95
TC21	0.96	0.97	0.97	0.96
TC22	0.87	0.99	0.87	0.9
TC23	0.95	0.98	0.84	0.93
TC24	0.88	0.94	0.72	0.78
TC25	0.98	0.99	0.89	0.94
TC26	0.95	0.97	0.96	0.92
TC27	0.95	0.99	0.96	0.92

TABLE 6.27: SSIM Value of T1 Weighted Images

Test Cases (TC)	MR_1	MR_2	MR_3	MR_4
TC28	0.98	0.98	0.96	0.98
TC29	0.93	0.97	0.9	0.87
TC30	0.91	0.98	0.95	0.88
TC31	0.97	1	0.96	0.94
TC32	0.94	0.98	0.95	0.88
TC33	0.96	0.97	0.97	0.94

According to Table 6.27, if the value of SSIM is equal to 1, then both the outputs of source and follow-up test cases are exactly similar. If the value of SSIM is equal to 0, then both the outputs of source and follow-up test cases are exactly dissimilar. The lower the values of SSIM, the more dissimilar the images are. We have set a threshold value for comparison because we did not get exact match for the images. Our reasoning for not getting an exact match is because the MRs are designed for conventional edge detection algorithms and our algorithm consists of many steps other than edge detection. Therefore, it is a high probability that the images may loose their contrast and luminance after processing.

We know that the relation in MT satisfies when output of both source and follow-up test cases are same. As we did not get the exact match between the outputs of source and follow-up test cases therefore, we need test cases that would satisfy the MR for calculating meaningful results. We have set the threshold value to 0.95 because the SSIM value greater than and equal to 0.95 shows the similarity of output images closest to 1. If the SSIM value is less than the given threshold

value, then the MR does not satisfy the relation for that test case. The FDR of MR is calculated by using the formula given in 6.1.

Let's suppose threshold is denoted by θ. The total number of test cases that satisfy the MR against θ value 0.95, and the FDR for all the MRs for T1 weighted images are shown in Table 6.28.

TABLE 6.28: Fault Detection Rate of T1 Weighted Images.

MR	$\theta = 0.95$	FDR
MR_1	15	54.54%
MR_2	29	12.12%
MR_3	21	36.36%
MR_4	12	63.63%

As shown in Table 6.28, when the value of θ is set to 0.95, the FDR of MR_4 is the highest (63.63%) followed by MR_1 (54.54%) by violating the MR on more than 50 percent test cases. The FDR of MR3 is 36.36% which is neither too high nor too low to identify the faults. MR_2 has the lowest (12.12%) FDR value. Hence it is concluded that MR2 has the lowest FDR and is not a recommendable MR to identify faults in T1 weighted images. MR_4 has the highest FDR value and is considered best to identify faults in T1 weighted images.MR_1 and MR_3 have also high FDR and are recommendable for this type of images. The SSIM value of T2 weighted images are shown in Table 6.30.

TABLE 6.29: SSIM Value of T2 Weighted Images

Test Cases (TC)	MR_1	MR_2	MR_3	MR_4
TC1	0.97	0.99	0.94	0.94

TABLE 6.29: SSIM Value of T1 Weighted Images

Test Cases (TC)	MR_1	MR_2	MR_3	MR_4
TC2	0.94	0.89	0.97	0.93
TC3	0.75	0.75	0.93	0.92
TC4	0.92	0.76	0.76	0.76
TC5	0.92	0.99	0.92	0.96
TC6	0.94	0.84	0.96	0.95
TC7	0.98	0.99	0.99	0.97
TC8	0.93	0.96	0.95	0.93
TC9	0.88	0.99	0.95	0.88
TC10	0.85	0.99	0.93	0.96
TC11	0.95	0.97	0.91	0.85
TC12	0.97	0.99	0.94	0.93
TC13	0.97	0.86	1	0.91
TC14	0.97	0.99	0.9	0.97
TC15	0.96	0.88	0.87	0.95

TABLE 6.30: SSIM Value of T2 Weighted Images

Test Cases (TC)	MR_1	MR_2	MR_3	MR_4
TC16	0.95	0.96	0.96	0.91
TC17	0.95	0.98	0.96	0.89
TC18	0.93	0.95	0.94	0.87
TC19	0.98	0.86	0.98	0.85
TC20	0.9	0.96	0.93	0.9
TC21	0.95	0.97	0.97	0.93
TC22	0.94	0.98	0.96	0.84
TC23	0.91	0.83	0.93	0.93
TC24	0.86	0.98	0.92	0.9
TC25	0.83	0.97	0.95	0.91
TC26	0.98	0.98	0.97	0.95
TC27	0.96	0.98	0.98	0.96
TC28	0.95	0.97	0.96	0.92
TC29	0.95	0.98	0.98	0.94

Table 6.30 shows that we have 29 test cases in the category of T2 weighted images. The test cases that satisfy the MR against the θ value 0.95 is depicted in Table 6.31.

TABLE 6.31: Fault Detection Rate of T2 Weighted Images.

MR	$\theta = 0.95$	FDR
MR_1	15	48.27%
MR_2	21	27.58%
MR_3	16	44.82%
MR_4	8	72.41%

Table 6.31 shows that considering θ as 0.95, FDR of MR_4 is the highest that is 72.41% followed by $MR_1, MR_3, and MR_2$ with FDR values 48.27%, 44.82%, and 27.58% respectively. Hence it is determined that all the MRs are useful for T2 weighted images when the θ is set to 0.95. The SSIM values of flair type images are given in Table 6.34.

TABLE 6.32: SSIM Value of Flair Images

Test Cases (TC)	MR_1	MR_2	MR_3	MR_4
TC1	0.98	0.97	0.97	0.98
TC2	0.95	0.96	0.98	0.97
TC3	1	0.99	1	0.99
TC4	0.99	0.99	0.99	0.98

TABLE 6.32: SSIM Value of Flair Images

Test Cases (TC)	MR_1	MR_2	MR_3	MR_4
TC5	0.84	0.96	0.94	0.94
TC6	0.96	0.97	0.97	0.95
TC7	0.97	0.99	0.95	0.97
TC8	0.98	0.99	0.99	0.98
TC9	0.91	0.96	0.91	0.89
TC10	0.96	0.98	0.98	0.92
TC11	0.98	0.99	0.92	0.97
TC12	0.99	1	0.99	0.97
TC13	0.98	1	0.98	0.97
TC14	0.84	0.98	0.96	0.97
TC15	0.99	0.97	0.98	0.99
TC16	0.87	0.97	0.93	0.76
TC17	1	1	1	0.98
TC18	0.96	0.98	0.97	0.97

TABLE 6.33: SSIM Value of Flair Images

Test Cases (TC)	MR_1	MR_2	MR_3	MR_4
TC19	1	0.99	1	0.99
TC20	0.98	0.99	0.98	0.98
TC21	0.99	0.99	0.99	0.99
TC22	0.97	0.98	0.97	0.95
TC23	0.94	0.91	0.91	0.98
TC24	0.95	0.98	0.87	0.93
TC25	0.99	0.99	0.99	0.99
TC26	0.97	1	0.94	0.92
TC27	0.99	0.96	0.98	0.99
TC28	0.96	0.98	0.96	0.96
TC29	0.92	0.96	0.77	0.83
TC30	0.94	0.97	0.95	0.92
TC31	0.91	0.98	0.97	0.96
TC32	0.96	0.99	0.97	0.97

TABLE 6.34: SSIM Value of Flair Images

Test Cases (TC)	MR_1	MR_2	MR_3	MR_4
TC33	0.92	0.99	0.99	0.94

The test cases that satisfy the MR against θ value 0.95 for flair type images with their FDR is depicted in Table 6.35.

TABLE 6.35: Fault Detection Rate of Flair Type Images.

MR	$\theta = 0.95$	FDR
MR_1	24	27.27%
MR_2	32	3.03%
MR_3	25	24.24%
MR_4	24	27.27%

Table 6.35 shows that MR_2 has the lowest FDR value of 3.03% whereas FDR of MR_1, MR_3, and MR_4 is 27.27%, 24.24%, and 27.27% respectively. Results show that like T1 and T2 weighted images, FDR of MR_4 is highest and MR_2 is lowest.Considering θ as 0.95, MR_2 has lowest FDR and is not recommendable for flair type images. FDR of MR_1, MR_3, and MR_4 is neither too high nor too low thus making them useful to identify faults. Figure 6.15 shows the statistics of first MR (counter clock-wise rotation at 90 degree) for all the types of images.

In Figure 6.15, when the θ is set to 0.95, the capability of MR_1 to detect faults is high for T1 and T2 weighted images by violating 18 and 14 test cases respectively. On the other hand, flair type images violate only 9 test cases. Hence it is concluded that MR_1 is more suitable for T1 and T2 weighted images rather than flair type images.

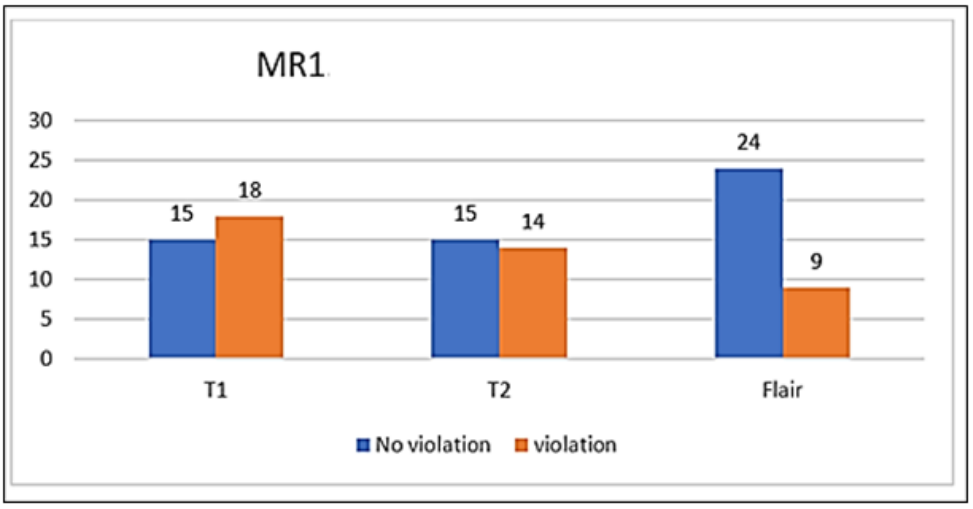

FIGURE 6.15: No. of Test Cases Violating MR_1 For T1, T2, and Flair Images.

Now, we consider the second MR (transpose of an image) and check the FDR of MR_2 on all three types of MRI images. Figure 6.16 shows the graphical representation of MR_2.

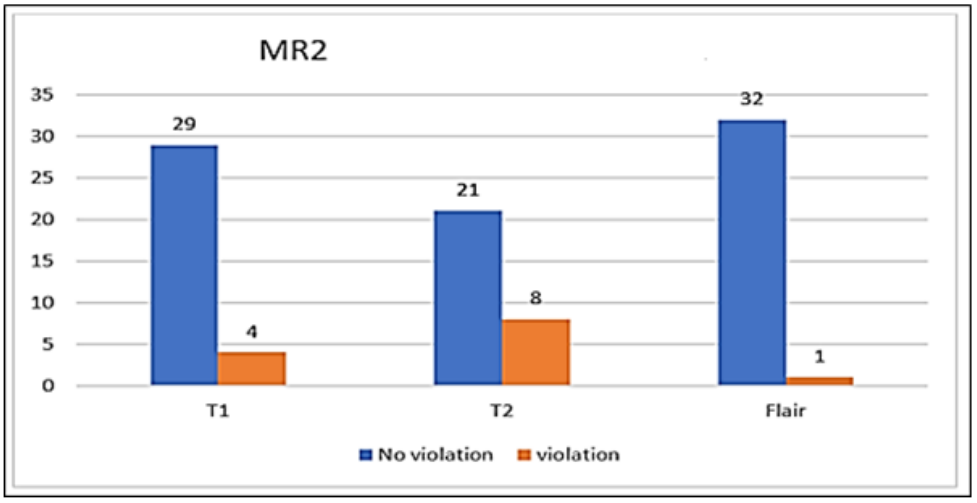

FIGURE 6.16: No. of Test Cases Violating MR_2 For T1, T2, and Flair Images.

Figure 6.16 shows that MR_2 has lowest capability to identify faults for T1 and flair type images. T1 images satisfy the relation on 29 test cases whereas the flair type images satisfy the relation on 32 test cases respectively. MR_2 is relatively better MR to identify faults in T2 weighted images by satisfying 21 test cases. It is concluded that MR_2 is recommendable for only T2 weighted images by violating the MR on 8 test cases. Now consider the third MR which is reflection at the ordinate. The result of MR_3 is given in Figure 6.17.

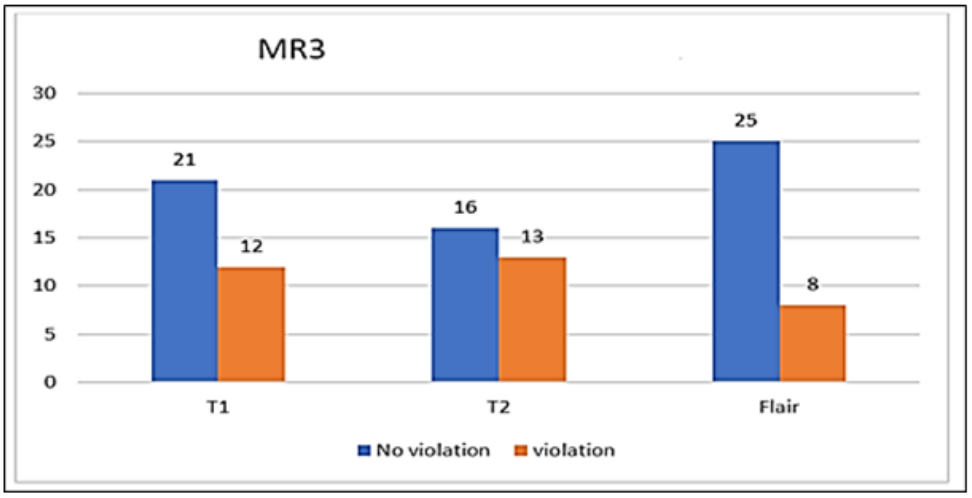

FIGURE 6.17: No. of Test Cases Violating MR_3 For T1, T2, and Flair Images.

Figure 6.17 shows that the capability of MR_3 to detect faults is low for flair type images by violating only 8 test cases whereas the capability of MR_3 to detect faults is high for T1 and T2 weighted images by violating 12 and 13 test cases respectively. It is concluded that MR_3 is recommended for all the categories of images. Fault detection capability of this MR is low for flair type images while comparing with T1 and T2 weighted images but it is still able to detect faults. The last MR is reflection at abscissa. Figure 6.18 shows the statistics of MR_4.

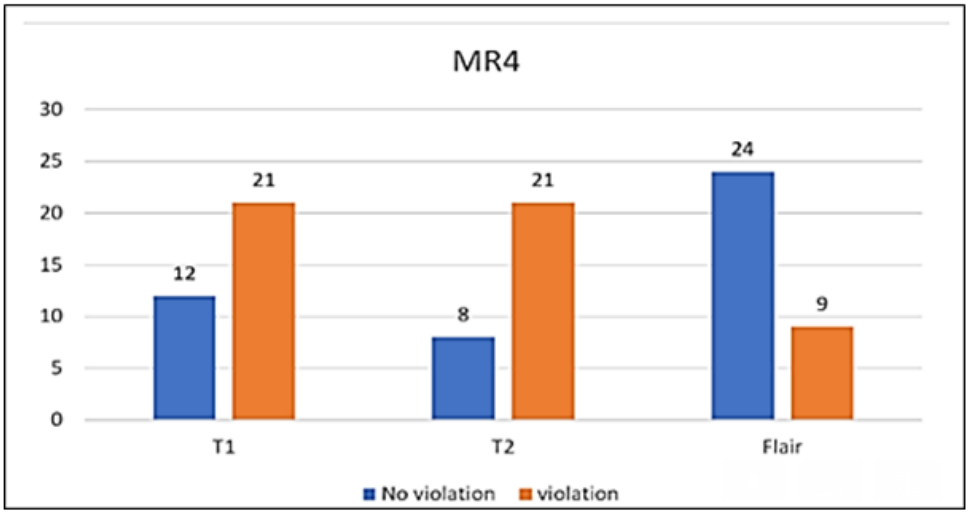

FIGURE 6.18: No. of Test Cases Violating MR_4 For T1, T2, and Flair Images.

Figure 6.18 shows that the FDR of MR_4 is very high for T1 and T2 weighted images when θ is set to 0.95. Both of the image types violate the relation on 21 test cases each. At the same θ value flair type images violate the relation on 9

test cases which is neither too low nor too high. Hence it is concluded that MR_4 is useful for all the three types of images but highly recommended for T1 and T2 weighted images. The FDR of all four MRs against all the three type of images is shown in Figure 6.19.

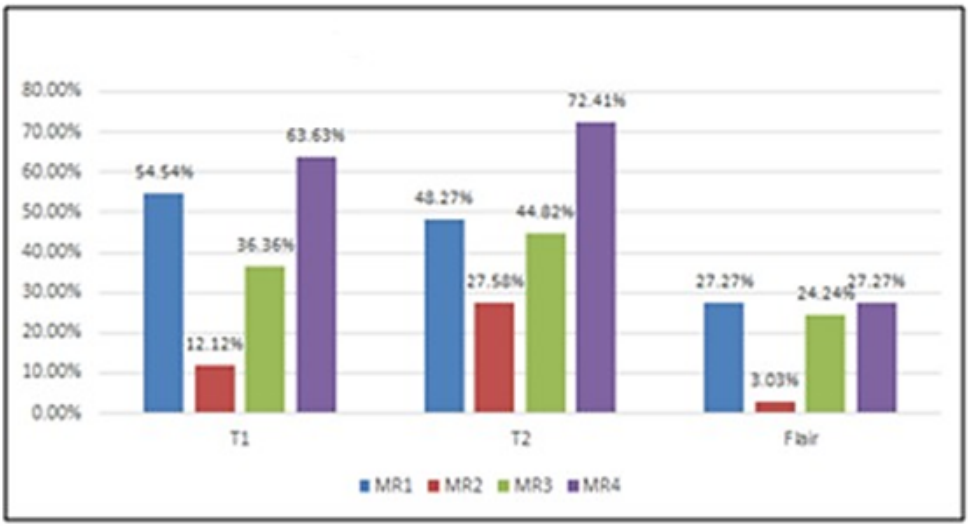

FIGURE 6.19: FDR of MRs on T1, T2 and Flair Type Images.

It is concluded from Figure 6.19 that for each image type, MR_4 is considered best among all the MRs by achieving highest FDR. The FDR of MR_4 for T1, T2, and flair type images are 63.63%, 72.41%, and 27.27% respectively. On the other hand, FDR of MR_2 is considered lowest in all three types of images with FDR value 12.12%, 27.54%, and 3.03% respectively. The FDR of MR_1 and MR_3 are neither too high nor too low but they are able to find the faults. Hence, it is observed that for T1, T2, and flair images, MR_4 should be preferred to enhance the credibility of MRI diagnostics. On the other hand, MR_2 produced low FDR and is not suggested for the diagnostics purpose specially in flair type images.

6.7 Threats to Validity

In this section, we have discussed the threats to validity of our experiment.
We have performed our experiment on a single dataset of MRI brain images. In this way, we cannot generalized our results. So, more datasets should be used to

make the results more generalized.

In this dissertation, we have performed testing on a particular code of python. It is recommended to perform testing on other languages as well. The reason is that the mutation operators used in mutation testing are language dependent. The mutation operators used in python may not be used in other programming languages such java, C++, C sharp etc,. So, codes written in different languages may reveal those faults which are not identified by the mutation operators used in python language.

6.8 Summary

In this chapter, MR evaluation results are discussed in detail. We have discussed the details about SUT used for our experiments, dataset, original test cases, and coverage criterion used. Effectiveness of mutation operators as well as effectiveness of MRs are discussed in detail. A comparison is made between the existing techniques of edge detection and morphological image operations with the proposed framework. Moreover, the effectiveness of proposed MRs over existing MRs are also highlighted.

Results show that AOR operator is the most effective operators in terms of mutants generated and mutants killed. Results of evaluation of the proposed framework of four MRs of edge detection show an improvement in all the respective MRs. Similarly, the dilation and erosion MRs have also shown improvement in four MRs using our proposed framework. It is observed after comparing the proposed MRs with the existing MRs that the proposed MRs are able to detect those faults which are not identified by any of the existing MRs.

Chapter 7

Conclusion and Future Work

This chapter will wrap up our dissertation by summarising this research from the angles of the earlier discussed research questions, aims and objectives. Future work ultimately provides guidance for how to continue this work.

7.1 Answers to Research Questions

This research is initiated with a number of research questions to be answered. The answers to these research questions are as follows:

1. RQ1: How to improve the evaluation of fault detection rate of MRs for MT ?

   In literature, the evaluation of fault detection rate of MRs is ascertained through mutation testing. For this evaluation, we need test cases. Test case selection strategies are developed to reveal the better faults detection. Some of the traditional test case generation techniques are random test generation through random model or Boolean model, behavioral or specification based, symbolic evaluation method, and combinatorial techniques etc,. In literature, the test cases are generated randomly and there is no systematic way to ascertain that the generated test cases are actually random and have

diversity to represent all different type of properties or full coverage. If the sample is not a full representation of the population then we would get biased results affecting the final outcome.

In proposed framework, the evaluation of fault detection rate of MRs for MT is improved by two methods. In the first method we have defined an all-inclusive sample population ready for a true sample to be selected from with every parametric value having equal probability of selection. The proposed method has used strong equivalence class testing along with code coverage for the generation of source test cases. In the second method, we have used all possible nine mutation operators to evaluate the MRs of edge detection and morphological image operations (dilation and erosion) in order to improve the fault detection rate of MRs rather than using a few mutation operators. Results of evaluation of the proposed framework of four MRs of edge detection show an improvement in all the respective MRs especially in MR_1 and MR_4 having FDR of 76.54% and 69.13% respectively which is 32% and 24% improved than the existing technique. The FDR of MR_2 and MR_3 is also improved by 1%. Similarly, the results of erosion and dilation operations show that out of 8 MRs, the FDR of four MRs are improved than the existing technique. In proposed framework, MR_1 is improved by 39%, MR_4 is improved by 0.5%, MR_6 is improved by 17%, and MR_8 is improved by 29%.

2. RQ2: How to create new MRs for MT ?

   In the field of IP, there are four general MRs which are applicable to almost all the IP operations. These general MRs are counter clock-wise rotation at 90 degree, transposition, reflection at the ordinate, and reflection at abscissa. The authors [69] have applied these four MRs on euclidean distance transform, The authors in [56] have applied these MRs on edge detection operation whereas the authors in [27] have applied two MRs i.e,. reflection at the ordinate and reflection at abscissa on dilation and erosion operations. An intriguing question arises that why they left the two (rotation at 90 degree and transposition) MRs?. We have decided to explore that option and apply these relations on dilation and erosion operations. In this way we have

proposed four general MRs for dilation and erosion operations.

Associative property $(A \oplus B) \oplus C = A \oplus (B \oplus C)$ is specific to dilation operation. We have changed the order of associative property $((A \oplus B) \oplus C = (A \oplus C) \oplus B)$ and checked whether the new arrangement satisfies the dilation operation or not. This result leads us to propose a new MR for dilation operation.

Image translation is an operation of IP. We have checked this operation on both erosion and dilation operations. Image translation satisfies on only erosion operation and we have proposed a specific MR for erosion operation.

Additionally, we have performed composition of MRs for the construction of new MRs. Composition of MR is cost effective as it reduces the number of executions considerably. Also, the value of composite MR (MR_{xy}) is not less than the lowest value of component MR (MR_x) and (MR_y). It is also observed that after the execution, the composite MR has always the value of the MR on second placeholder. Hence, it can be concluded that we have to choose that composite MR which has highest value of the MR at the second place.

3. RQ3: How effective are the proposed MRs in comparison with the existing MRs ?

   To ascertain the effectiveness of MRs, we have compared the total number of mutants killed by existing and proposed MRs. In mutation testing, we have used eight mutation operators (AOR, COI, ROR, RIL, OIL, SDL, SIR, and ZIL) and generated 130 mutants from these operators. The effectiveness of existing and proposed MRs is evident by the number of mutants killed by each MR. Among the results, we have selected those faults that were not detected by the eight existing MRs. But were identified by the proposed MRs.

   Following are the nine AOR faults ($mt_6, mt_8, mt_9, mt_{10}, mt_{31}, mt_{32}, mt_{33}$, mt_{34}, mt_{35}) which are identified by the proposed MRs but are not identified by any of the existing eight MRs. Among these faults, eight faults ($mt_8, mt_9, mt_{10}, mt_{31}, mt_{32}, mt_{33}, mt_{34}, mt_{35}$) are identified by MR_4, five faults

($mt_{31}, mt_{32}, mt_{33}, mt_{34}, mt_{35}$) are identified by MR_5, and nine faults (mt_6, $mt_8, mt_9, mt_{10}, mt_{31}, mt_{32}, mt_{33}, mt_{34}, mt_{35}$) are identified by MR_6. So, the proposed MRs complement the existing MRs effectively as the proposed MRs are are able to find those faults which are not identified by the existing MRs.

7.2 Conclusion

Testing of IPAs, of course, is a challenging task because of the absence of test oracle. Metamorphic testing is an effective method to deal with the applications with a test oracle problem. Metamorphic relations play an important role in metamorphic testing. A metamorphic relation relates two or more inputs with their expected outputs after execution of the properties of the target program. Properties of different IP operations can also be used as metamorphic relations.

In our proposed framework, random source test cases are selected through the strong equivalence class testing technique (black-box testing), and then, the adequacy of the selected source test cases is verified through code coverage criteria (white-box testing).

MT is widely used to handle the test oracle problem of the IPA as the related and relevant MRs can identify the faults in the SUT used in the MT. However, every MR is not suitable for bug manifestation. In the proposed framework, we have proposed six new MRs of morphological image operations (dilation and erosion). The fault detection rate of newly proposed algorithm along with existing MRs of edge detection and dilation and erosion is determined through mutation testing. A total of nine mutation operators are used to increase the total number of mutants that improves the fault detection rate of MRs. The effectiveness of mutation operators is also determined that which operator is more effective to kill maximum number of mutants. AOR is considered the best operator in both the subject programs as it generates maximum number of mutants. We have compared the results of our proposed approach with the existing techniques of edge detection and morphological image operations. Our results demonstrate that the mutation score of all the MRs of edge detection has improved whereas the MRs

of dilation and erosion has shown improvement in four MRs (out of 8). While comparing our proposed MRs with the existing MRs of dilation and erosion operations, we have come to the conclusion that the proposed MRs complement the existing MRs effectively as the proposed MRs are are able to find those faults which are not identified by the existing MRs.

7.3 Future Directions

In future, the proposed framework for the evaluation of MRs can be strengthened by using more coverage criterion for improved coverage. The future work involving coverage criterion may include multiple condition coverage (MCC) where every combination of conditions' outcomes is tested at least once in a decision, modified condition/decision coverage (MCDC) where a decision's potential outcomes are determined by each condition contained within the decision, all def-use (definition-usage) coverage where all-def coverage is attained when all defs of any variable are covered and all-uses coverage is attained when a path from each def to each use of that def has been exercised etc. The proposed framework can also be strengthened by using additional mutation operators that will cover the fault types not used in the proposed framework such as logical connector replacement (LCR), break continue replacement (BCR), constant replacement(CRP), classmethod decorator insertion (CDI) etc,.

In the proposed framework, we have ascertained the effectiveness of edge detection MRs on Sboel edge detection algorithm and Canny edge detection algorithm. In future, the effectiveness of these edge detection MRs can also be checked on other edge detection algorithms such as Roberts, Prewitt, fuzzy logic methods etc,. A comparison can be made to determine which of the edge detection algorithm is most effective for fault identification.

Bibliography

[1] C. Bolchini, L. Cassano, A. Mazzeo, and A. Miele, "Usability-based cross-layer reliability evaluation of image processing applications," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6, IEEE, 2021.

[2] A. Raid, W. Khedr, M. El-Dosuky, and M. Aoud, "Image restoration based on morphological operations," *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, vol. 4, no. 3, pp. 9–21, 2014.

[3] F. U. Rehman and C. Izurieta, "An approach for verifying and validating clustering based anomaly detection systems using metamorphic testing," in *2022 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pp. 12–18, IEEE, 2022.

[4] A. Memon, I. Banerjee, and A. Nagarajan, "What test oracle should i use for effective gui testing?," in *18th IEEE International Conference on Automated Software Engineering, 2003. Proceedings.*, pp. 164–173, IEEE, 2003.

[5] D. Seca, "A review on oracle issues in machine learning," *arXiv preprint arXiv:2105.01407*, 2021.

[6] P. Saha and U. Kanewala, "Fault detection effectiveness of source test case generation strategies for metamorphic testing," in *Proceedings of the 3rd International Workshop on Metamorphic Testing*, pp. 2–9, 2018.

[7] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases," *arXiv preprint arXiv:2002.12543*, 2020.

[8] Z. Q. Zhou, S. Xiang, and T. Y. Chen, "Metamorphic testing for software quality assessment: A study of search engines," *IEEE Transactions on Software Engineering*, vol. 42, no. 3, pp. 264–284, 2015.

[9] S. Segura, J. Troya, A. Durán, and A. Ruiz-Cortés, "Performance metamorphic testing: Motivation and challenges," in *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER)*, pp. 7–10, IEEE, 2017.

[10] Q.-H. Luu, H. Liu, T. Y. Chen, and H. L. Vu, "Testing ocean software with metamorphic testing," in *2022 IEEE/ACM 7th International Workshop on Metamorphic Testing (MET)*, pp. 23–30, IEEE, 2022.

[11] T.-P. Hong, C.-C. Chiu, J.-H. Su, and C.-H. Chen, "Applicable metamorphic testing for erasable-itemset mining," *IEEE Access*, vol. 10, pp. 38545–38554, 2022.

[12] J. G. Adigun, L. Eisele, and M. Felderer, "Metamorphic testing in autonomous system simulations," in *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 330–337, IEEE, 2022.

[13] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software testing techniques: A literature review," in *2016 6th international conference on information and communication technology for the Muslim world (ICT4M)*, pp. 177–182, IEEE, 2016.

[14] Rahul, "What Is Software Testing - Overview, Process, Importance and Terms — technotrice.com." https://technotrice.com/what-is-software-testing/. [Accessed 27-May-2023].

[15] "The expected results of the software are: - Manual testing — careerride.com." https://www.careerride.com/question-29-Manual-testing. [Accessed 27-May-2023].

[16] K. Sugali, "Software testing: Issues and challenges of artificial intelligence & machine learning," *International Journal of Artificial Intelligence and Applications*, vol. 12, no. 1, pp. 101–112, 2021.

[17] E. J. Weyuker, "On testing non-testable programs," *The Computer Journal*, vol. 25, no. 4, pp. 465–470, 1982.

[18] G. Jahangirova, "Oracle problem in software testing," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 444–447, 2017.

[19] T. Y. Chen and T. Tse, "New visions on metamorphic testing after a quarter of a century of inception," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1487–1490, 2021.

[20] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A survey on metamorphic testing," *IEEE Transactions on software engineering*, vol. 42, no. 9, pp. 805–824, 2016.

[21] M. Lindvall, A. Porter, G. Magnusson, and C. Schulze, "Metamorphic model-based testing of autonomous systems," in *2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)*, pp. 35–41, IEEE, 2017.

[22] P. Saha and U. Kanewala, "Fault detection effectiveness of metamorphic relations developed for testing supervised classifiers," in *2019 IEEE International conference on artificial intelligence testing (AITest)*, pp. 157–164, IEEE, 2019.

[23] A. Ma, S. Yan, and X. Yang, "Calculation method of metamorphic relational complexity for numerical computation programs based on scale complexity,"

in *International Conference on Signal Processing, Computer Networks, and Communications (SPCNC 2022)*, vol. 12626, pp. 588–593, SPIE, 2023.

[24] Y. Cao, Z. Q. Zhou, and T. Y. Chen, "On the correlation between the effectiveness of metamorphic relations and dissimilarities of test case executions," in *2013 13th International Conference on Quality Software*, pp. 153–162, IEEE, 2013.

[25] M. Asrafi, H. Liu, and F.-C. Kuo, "On testing effectiveness of metamorphic relations: A case study," in *2011 fifth international conference on secure software integration and reliability improvement*, pp. 147–156, IEEE, 2011.

[26] X. Xie, J. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Application of metamorphic testing to supervised classifiers," in *2009 Ninth International Conference on Quality Software*, pp. 135–144, IEEE, 2009.

[27] T. Jameel, M. Lin, and L. Chao, "Test oracles based on metamorphic relations for image processing applications," in *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 1–6, IEEE, 2015.

[28] H. Liu, X. Liu, and T. Y. Chen, "A new method for constructing metamorphic relations," in *2012 12th international conference on quality software*, pp. 59–68, IEEE, 2012.

[29] R. Guderlei and J. Mayer, "Towards automatic testing of imaging software by means of random and metamorphic testing," *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, no. 06, pp. 757–781, 2007.

[30] T. Jameel, L. Mengxiang, and L. Chao, "A framework of automatic testing of image processing applications," in *2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pp. 312–317, IEEE, 2016.

[31] "Test Case — tutorialspoint.com." https://www.tutorialspoint.com/software_testing_dictionary/test_case.htm. [Accessed 02-Jun-2023].

[32] "What is expected outcome in software testing? — educative.io." https://www.educative.io/answers/what-is-expected-outcome-in-software-testing. [Accessed 02-Jun-2023].

[33] J. Mayer, "On testing image processing applications with statistical methods," *Conferencee on Software Engineering*, pp. 69–78, 2005.

[34] S. Segura, D. Towey, Z. Q. Zhou, and T. Y. Chen, "Metamorphic testing: Testing the untestable," *IEEE Software*, vol. 37, no. 3, pp. 46–53, 2018.

[35] G. Fraser and A. Arcuri, "Evosuite: On the challenges of test case generation in the real world," in *2013 IEEE sixth international conference on software testing, verification and validation*, pp. 362–369, IEEE, 2013.

[36] S. Segura, J. Troya, A. Durán, and A. Ruiz-Cortés, "Performance metamorphic testing: A proof of concept," *Information and Software Technology*, vol. 98, pp. 1–4, 2018.

[37] M. H. Rasheed, H. N. Fadhel, and M. M. Siddeq, "Novel methods to measure the quality of 2d images compression techniques," *Research Square Platform LLC*, 2023.

[38] M. I. Jaafar, S. W. Nawawi, and R. A. Rahim, "Improving measurement bias of structural similarity index (ssim) using absolute difference equation," *Applications of Modelling and Simulation*, vol. 6, pp. 10–19, 2022.

[39] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–27, 2018.

[40] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1120–1154, 2018.

[41] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE transactions on software engineering*, vol. 41, no. 5, pp. 507–525, 2014.

[42] R. Just and F. Schweiggert, "Evaluating testing strategies for imaging software by means of mutation analysis," in *2009 International Conference on Software Testing, Verification, and Validation Workshops*, pp. 205–209, IEEE, 2009.

[43] J. Chen, Y. Wang, Y. Guo, and M. Jiang, "A metamorphic testing approach for event sequences," *Plos one*, vol. 14, no. 2, p. e0212476, 2019.

[44] J. Bell, C. Murphy, and G. Kaiser, "Metamorphic runtime checking of applications without test oracles," *CrossTalk*, vol. 28, no. 2, pp. 9–13, 2015.

[45] M. Jiang, T. Y. Chen, F.-C. Kuo, D. Towey, and Z. Ding, "A metamorphic testing approach for supporting program repair without the need for a test oracle," *Journal of systems and software*, vol. 126, pp. 127–140, 2017.

[46] C. Jiang, S. Huang, and Z.-w. Hui, "Metamorphic testing of image region growth programs in image processing applications," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 70–72, IEEE, 2018.

[47] M. Pu, C. Y. Chong, and M. K. Lim, "Robustness evaluation in hand pose estimation models using metamorphic testing," *arXiv preprint arXiv:2303.04566*, 2023.

[48] H. Spieker and A. Gotlieb, "Adaptive metamorphic testing with contextual bandits," *Journal of Systems and Software*, vol. 165, p. 110574, 2020.

[49] C.-A. Sun, H. Dai, H. Liu, and T. Y. Chen, "Feedback-directed metamorphic testing," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 1, pp. 1–34, 2023.

[50] Z. Ying, A. Bellotti, D. Towey, T. Y. Chen, and Z. Q. Zhou, "Using metamorphic relation violation regions to support a simulation framework for

the process of metamorphic testing," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1722–1727, IEEE, 2022.

[51] U. Kanewala, J. M. Bieman, and A. Ben-Hur, "Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels," *Software testing, verification and reliability*, vol. 26, no. 3, pp. 245–269, 2016.

[52] X. Lin, M. Simon, and N. Niu, "Exploratory metamorphic testing for scientific software," *Computing in science & engineering*, vol. 22, no. 2, pp. 78–87, 2018.

[53] M. Jiang, T. Y. Chen, and S. Wang, "On the effectiveness of testing sentiment analysis systems with metamorphic testing," *Information and Software Technology*, vol. 150, no. 1, p. 106966, 2022.

[54] L. Jin, Z. Ding, and H. Zhou, "Evaluation of chinese natural language processing system based on metamorphic testing," *Mathematics*, vol. 10, no. 8, p. 1276, 2022.

[55] C. A. Sari, W. S. Sari, and H. Rahmalan, "A combination of k-means and fuzzy c-means for brain tumor identification," *Scientific Journal of Informatics*, vol. 8, no. 1, pp. 76–83, 2021.

[56] K. Sim, D. Wong, and T. Hii, "Evaluating the effectiveness of metamorphic testing on edge detection programs," *International Journal of Innovation, Management and Technology*, vol. 4, no. 1, pp. 6–10, 2013.

[57] C.-A. Sun, A. Fu, P.-L. Poon, X. Xie, H. Liu, and T. Y. Chen, "Metric ˆ{+}+: A metamorphic relation identification technique based on input plus output domains," *IEEE Transactions on Software Engineering*, vol. 47, no. 9, pp. 1764–1785, 2019.

[58] H. Liu, F.-C. Kuo, D. Towey, and T. Y. Chen, "How effectively does metamorphic testing alleviate the oracle problem?," *IEEE Transactions on Software Engineering*, vol. 40, no. 1, pp. 4–22, 2013.

[59] K. Qiu, Z. Zheng, T. Y. Chen, and P.-L. Poon, "Theoretical and empirical analyses of the effectiveness of metamorphic relation composition," *IEEE Transactions on software engineering*, vol. 48, no. 3, pp. 1001–1017, 2020.

[60] M. Ojdanic, E. Soremekun, R. Degiovanni, M. Papadakis, and Y. Le Traon, "Mutation testing in evolving systems: Studying the relevance of mutants to code evolution," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 1, pp. 1–39, 2023.

[61] F. Tambon, V. Majdinasab, A. Nikanjam, F. Khomh, and G. Antonio, "Mutation testing of deep reinforcement learning based on real faults," *arXiv preprint arXiv:2301.05651*, 2023.

[62] M. Papadakis, M. Kintis, J. Zhang, Y. Jia, Y. Le Traon, and M. Harman, "Mutation testing advances: an analysis and survey," in *Advances in Computers*, vol. 112, pp. 275–378, Elsevier, 2019.

[63] B. Tekinerdogan, H. G. Gurbuz, C. Catal, and N. P. Er, "Test suite assessment of safety-critical systems using safety tactics and fault-based mutation testing," *Authorea Preprints*, 2023.

[64] V. H. Durelli, R. S. Durelli, S. S. Borges, A. T. Endo, M. M. Eler, D. R. Dias, and M. P. Guimarães, "Machine learning applied to software testing: A systematic mapping study," *IEEE Transactions on Reliability*, vol. 68, no. 3, pp. 1189–1212, 2019.

[65] M. B. Kusharki, S. Misra, B. Muhammad-Bello, I. A. Salihu, and B. Suri, "Automatic classification of equivalent mutants in mutation testing of android applications," *Symmetry*, vol. 14, no. 4, p. 820, 2022.

[66] Q. Hu, L. Ma, X. Xie, B. Yu, Y. Liu, and J. Zhao, "Deepmutation++: A mutation testing framework for deep learning systems," in *2019 34th*

IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 1158–1161, IEEE, 2019.

[67] P. Delgado-Pérez, A. B. Sánchez, S. Segura, and I. Medina-Bulo, "Performance mutation testing," *Software Testing, Verification and Reliability*, vol. 31, no. 5, p. e1728, 2021.

[68] A. J. Offutt, A. Lee, G. Rothermel, R. H. Untch, and C. Zapf, "An experimental determination of sufficient mutant operators," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 5, no. 2, pp. 99–118, 1996.

[69] J. Mayer and R. Guderlei, "On random testing of image processing applications," in *2006 Sixth International Conference on Quality Software (QSIC'06)*, pp. 85–92, IEEE, 2006.

[70] R. Just and F. Schweiggert, "Automating unit and integration testing with partial oracles," *Software Quality Journal*, vol. 19, no. 4, pp. 753–769, 2011.

[71] J. Ding and X.-H. Hu, "Application of metamorphic testing monitored by test adequacy in a monte carlo simulation program," *Software Quality Journal*, vol. 25, no. 3, pp. 841–869, 2017.

[72] T. Jameel, L. Mengxiang, and L. Chao, "Automatic test oracle for image processing applications using support vector machines," in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 1110–1113, IEEE, 2015.

[73] W. Chan, J. C. Ho, and T. Tse, "Piping classification to metamorphic testing: An empirical study towards better effectiveness for the identification of failures in mesh simplification programs," in *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, vol. 1, pp. 397–404, IEEE, 2007.

[74] J. Ding, T. Wu, J. Q. Lu, and X.-H. Hu, "Self-checked metamorphic testing of an image processing program," in *2010 Fourth International Conference*

on Secure Software Integration and Reliability Improvement, pp. 190–197, IEEE, 2010.

[75] J. Ding, D. Zhang, and X.-H. Hu, "An application of metamorphic testing for testing scientific software," in *Proceedings of the 1st International Workshop on Metamorphic Testing*, pp. 37–43, 2016.

[76] R. Ibrahim, A. A. B. Amin, S. Jamel, and J. A. Wahab, "Epit: A software testing tool for generation of test cases automatically," *arXiv preprint arXiv:2007.11197*, 2020.

[77] S. I. Khaleel and R. Anan, "A review paper: optimal test cases for regression testing using artificial intelligent techniques," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 2, pp. 1803–1816, 2023.

[78] V. Arnicane, "Complexity of equivalence class and boundary value testing methods," *Int J Comput Sci Inform Techn*, vol. 751, pp. 80–101, 2009.

[79] M. E. Khan and F. Khan, "A comparative study of white box, black box and grey box testing techniques," *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 6, pp. 12–15, 2012.

[80] S. Nidhra and J. Dondeti, "Black box and white box testing techniques-a literature review," *International Journal of Embedded Systems and Applications (IJESA)*, vol. 2, no. 2, pp. 29–50, 2012.

[81] K. Burr and W. Young, "Combinatorial test techniques: Table-based automation, test generation and code coverage," in *Proc. of the Intl. Conf. on Software Testing Analysis & Review*, Citeseer, 1998.

[82] Z. Zhou, Z. Zheng, T. Y. Chen, J. Zhou, and K. Qiu, "Follow-up test cases are better than source test cases in metamorphic testing: A preliminary study," in *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*, pp. 69–74, IEEE, 2021.

[83] J. Ayerdi, S. Segura, A. Arrieta, G. Sagardui, and M. Arratibel, "Qos-aware metamorphic testing: An elevation case study," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pp. 104–114, IEEE, 2020.

[84] A. Arrieta, "Multi-objective metamorphic follow-up test case selection for deep learning systems," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1327–1335, 2022.

[85] F. Jafari, A. Nadeem, and Q. u. Zaman, "Evaluation of metamorphic testing for edge detection in mri brain diagnostics," *Applied Sciences*, vol. 12, no. 17, p. 8684, 2022.

[86] A. Arrieta, "On the cost-effectiveness of composite metamorphic relations for testing deep learning systems," in *2022 IEEE/ACM 7th International Workshop on Metamorphic Testing (MET)*, pp. 42–47, IEEE, 2022.

[87] S. Israni and S. Jain, "Edge detection of license plate using sobel operator," in *2016 international conference on electrical, electronics, and optimization techniques (ICEEOT)*, pp. 3561–3563, IEEE, 2016.

[88] M. Yasir, M. S. Hossain, S. Nazir, S. Khan, and R. Thapa, "Object identification using manipulated edge detection techniques," *Science*, vol. 3, no. 1, pp. 1–6, 2022.

[89] R. Song, Z. Zhang, and H. Liu, "Edge connection based canny edge detection algorithm," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 8, no. 6, pp. 1228–1236, 2017.

[90] N. Mathur, S. Mathur, and D. Mathur, "A novel approach to improve sobel edge detector," *Procedia Computer Science*, vol. 93, pp. 431–438, 2016.

[91] R. Tian, G. Sun, X. Liu, and B. Zheng, "Sobel edge detection based on weighted nuclear norm minimization image denoising," *Electronics*, vol. 10, no. 6, p. 655, 2021.

[92] A. Asmaidi, D. S. Putra, M. M. Risky, *et al.*, "Implementation of sobel method based edge detection for flower image segmentation," *Sinkron: jurnal dan penelitian teknik informatika*, vol. 3, no. 2, pp. 161–166, 2019.

[93] G. M. H. Amer and A. M. Abushaala, "Edge detection methods," in *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*, pp. 1–7, IEEE, 2015.

[94] "Concept of Edge Detection - Javatpoint — javatpoint.com." https://www.javatpoint.com/dip-concept-of-edge-detection. [Accessed 27-Jun-2023].

[95] L. Najman and H. Talbot, *Mathematical morphology: from theory to applications*. John Wiley & Sons, 2 ed., 2013.

[96] M. Goyal, "Morphological image processing," *IJCST*, vol. 2, no. 4, p. 59, 2011.

[97] R. A. Lotufo, R. Audigier, A. V. Saúde, and R. C. Machado, "Morphological image processing," in *Microscope image processing*, pp. 75–117, Elsevier, 2023.

[98] "Morphological Image Processing — cs.auckland.ac.nz." https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm. [Accessed 20-Apr-2023].

[99] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3523–3542, 2021.

[100] X. Liu, L. Song, S. Liu, and Y. Zhang, "A review of deep-learning-based medical image segmentation methods," *Sustainability*, vol. 13, no. 3, p. 1224, 2021.

[101] L. Lalaoui and T. Mohamadi, "A comparative study of image region-based segmentation algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 6, pp. 198–206, 2013.

[102] M. Habijan, D. Babin, I. Galić, H. Leventić, K. Romić, L. Velicki, and A. Pižurica, "Overview of the whole heart and heart chamber segmentation methods," *Cardiovascular Engineering and Technology*, vol. 11, no. 24, pp. 725–747, 2020.

[103] K. Ramesh, G. K. Kumar, K. Swapna, D. Datta, and S. S. Rajest, "A review of medical image segmentation algorithms," *EAI Endorsed Transactions on Pervasive Health and Technology*, vol. 7, no. 27, pp. e6–e6, 2021.

[104] S. Mirjalili, J. Song Dong, A. S. Sadiq, and H. Faris, "Genetic algorithm: Theory, literature review, and application in image reconstruction," *Nature-Inspired Optimizers: Theories, Literature Reviews and Applications*, vol. 811, pp. 69–85, 2020.

[105] "What Is Image Reconstruction? (with pictures) — easytechjunkie.com." https://www.easytechjunkie.com/what-is-image-reconstruction.htm. [Accessed 28-Jun-2023].

[106] L. Fu, Y. Lei, M. Yan, L. Xu, Z. Xu, and X. Zhang, "Metafl: Metamorphic fault localisation using weakly supervised deep learning," *IET Software*, vol. 17, no. 2, pp. 137–153, 2023.

[107] X. Lin, M. Simon, and N. Niu, "Hierarchical metamorphic relations for testing scientific software," in *Proceedings of the International Workshop on Software Engineering for Science*, pp. 1–8, 2018.

[108] T. Y. Chen, P.-L. Poon, and X. Xie, "Metric: Metamorphic relation identification based on the category-choice framework," *Journal of Systems and Software*, vol. 116, no. 2, pp. 177–190, 2016.

[109] D. Sobania, M. Briesch, P. Röchner, and F. Rothlauf, "Mtgp: Combining metamorphic testing and genetic programming," in *Genetic Programming: 26th European Conference, EuroGP 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings*, pp. 324–338, Springer, 2023.

[110] E. Altamimi, A. Elkawakjy, and C. Catal, "Metamorphic relation automation: Rationale, challenges, and solution directions," *Journal of Software: Evolution and Process*, vol. 35, no. 1, p. e2509, 2023.

[111] "Translate an Image Using imtranslate Function - MATLAB amp; Simulink — mathworks.com." https://www.mathworks.com/help/images/translate-an-image.html. [Accessed 17-May-2023].

[112] I. G. Prahmana and K. A. B. Sitepu, "Identification identification of land and water centella asiatica leaf herbal plants using digital imagery with the sobel edge detection algorithm," *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, vol. 2, no. 2, pp. 48–52, 2023.

[113] R. A. AS and S. Gopalan, "Comparative analysis of eight direction sobel edge detection algorithm for brain tumor mri images," *Procedia Computer Science*, vol. 201, pp. 487–494, 2022.

[114] M. A. Kumar, N. S. Goud, R. Sreeram, and R. G. Prasuna, "Image processing based on adaptive morphological techniques," in *2019 International Conference on Emerging Trends in Science and Engineering (ICESE)*, vol. 1, pp. 1–4, IEEE, 2019.

[115] R. Srisha and A. Khan, "Morphological operations for image processing: understanding and its applications," *NCVSComs-13*, vol. 13, pp. 17–19, 2013.

[116] A. C. Barus, T. Y. Chen, F.-C. Kuo, H. Liu, and H. W. Schmidt, "The impact of source test case selection on the effectiveness of metamorphic testing," in *Proceedings of the 1st International Workshop on Metamorphic Testing*, pp. 5–11, 2016.

[117] U. Sara, M. Akter, and M. S. Uddin, "Image quality assessment through fsim, ssim, mse and psnr—a comparative study," *Journal of Computer and Communications*, vol. 7, no. 3, pp. 8–18, 2019.

[118] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli, "Image quality assessment: Unifying structure and texture similarity," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 5, pp. 2567–2581, 2020.