ABSTRACT

Self-Checked Metamorphic Testing of Monte Carlo Simulation

by Tong Wu

March, 2011

Chair: Dr. Junhua Ding

Major Department: Computer Science

Photon propagation in biological tissues can be modeled with Monte Carlo simulations numerically. However, testing a such program is difficult due to the unknown character of the test oracles. Although approaches based on Beer-Lambert law, van de Hulst's table or Radiative Transfer Equation (RTE) can be used for testing the Monte Carlo modeling programs, these approaches are only applied to the programs that are designed for homogeneous media. A rigorous way for testing the Monte Carlo modeling programs for heterogeneous media is needed.

Metamorphic testing, as an effective approach for testing systems that do not have test oracles, is one of possible supplementary approaches to test a Monte Carlo modeling program for heterogeneous media. In metamorphic testing, instead of verifying the correctness of a test output, the satisfaction of a metamorphic relation of the test outputs is checked. If a violation of the metamorphic relation is found, the system implementation must have some faults. However, checking only the metamorphic relations is not good enough to ensure the testing quality. Randomly or accidently generated incorrect outputs may satisfy a metamorphic relation as well. Therefore, it is necessary to provide a systematic approach to measure the test effectiveness of a metamorphic testing, to choose metamorphic relations, and to generate test input data.

In this thesis, we propose a new approach called self-checked metamorphic testing. In our new approach, the original metamorphic testing is extended with the evaluation of the adequacy

of testing coverage criteria to measure the quality of a metamorphic testing, to guide the creation of metamorphic relations, to generate testing inputs, and to investigate the found exceptions. The effectiveness of this approach has been demonstrated through testing a parallel Monte Carlo modeling program we developed for simulating photon propagation in human skins.

This thesis contains three parts of work. In first part, the enhanced Monte Carlo modeling program was used to preliminarily study the relationship between the height of the collection lens and the contrast values of the reflectance image of the system. In second part, the homogenous part of the Monte Carlo program was validated with van de Hulst's table method, which compares the simulation results with the calculated values on van de Hulst's table. The third and the main part of the thesis is applying the self-checked metamorphic testing approach to test the Monte Carlo modeling program.

Self-Checked Metamorphic Testing of Monte Carlo Simulation

A Thesis

Presented To

The Faculty of the Department of Computer Science

East Carolina University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Tong Wu

March, 2011

Self-Checked Metamorphic Testing of Monte Carlo Simulation

by

Tong Wu

APPROVED BY:

DIRECTOR OF THESIS:_____
Junhua Ding, PhD

COMMITTEE MEMBER:_____
Xin-Hua Hu, PhD

COMMITTEE MEMBER:_____
M. Nasseh Tabrizi, PhD

COMMITTEE MEMBER:_____
Sergiy Vilkomir, PhD

CHAIR OF THE DEPARTMENT OF COMPUTER SCIENCE:

_____
Karl Abrahamson, PhD

DEAN OF THE GRADUATE SCHOOL:

_____
Paul J. Gemperline, PhD

DEDICATION

I would like to dedicate this work to my parent, Fenghua Dai and Xueshun Wu, my beloved wife, Kangqing Chen, and my baby daughter, Katharine Wu.

Thank you for all the love, understanding, supports, and encouragement.

Without any of you, I could do nothing.

ACKNOWLEDGEMENTS

First of all, I wish to send my tremendous gratitude to my advisor, Dr. Junhua Ding, for his incredible guidance, training, and influence. He helps me to absorb knowledge from multiple research areas and teaches me to solve the problem like a software engineer. His positive attitude towards unknown mysteries always encourages me to overcome the frustrations in my research work.

I would also like to thank my committee members, Dr. Xin-Hua Hu, Dr. Nasseh Tabrizi, and Dr. Sergiy Vilkomir. Dr. Hu have been very generous and patient to answer all kinds of simple physics questions from me for a very long time. Dr. Tabrizi provides me such a wonderful opportunity to study in this graduate program. And almost all my knowledge about software engineering and software testing comes from Dr. Tabrizi and Dr. Vilkomir's courses.

I must thank Dr. Jun-Qing Lu for helping me manage my account on the computing cluster and providing technical support in parallel program and UNIX system.

Finally, I want to give my thanks to all my friends at East Carolina University, in United States, and in China for their great help and supports.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

For some software systems, it is impossible or very difficult to create test oracles to validate the systems. Generally speaking, this kind of software is hard to test with normal testing methods and is called "non-testable" software. Metamorphic testing is an effective approach which is designed to test "non-testable" systems. However, there is an obvious drawback of metamorphic testing, which prevents this creative technique to be more practical for the real world. Thus, to say that metamorphic testing itself could not ensure the quality of metamorphic relations which are used in the testing. Randomly or accidentally generated incorrect outputs may satisfy a metamorphic relation as well. Therefore, the testing quality could not be guaranteed by checking only metamorphic relations.

A major theme of this thesis is to propose a self-checked metamorphic testing approach, which integrates structural testing into metamorphic testing. We hope this approach could not only detect subtle defects in system implementation, but also further verify the metamorphic testing results with structure coverage information. A real world computational program, which generates reflectance images by simulating light transportation in heterogeneous media with Monte Carlo model, has been used as a case study to investigate the effectiveness of our new approach. After that, a small simulation experiment has been done with the verified code. The results are used to clarify the relationship between the height of the collection lens and the contrast values of the image of the system.

In chapter two, the background knowledge and the hypothesis of our new approach has been presented. Basic information related to our topic, like software testing, metamorphic testing, structural testing, test coverage criteria, Monte Carlo model, and the simulation of light transportation, has been introduced. Following that, it is our further discussion about the

metamorphic testing and the hypothesis of our new approach. What is our new idea? Comparing with the original metamorphic relations, what is the improvement or enhancement of our new approach? These questions will also be answered in this chapter.

The implementation of the Monte Carlo code and the experimental details of the case study are introduced in chapter three. In this chapter, there are parts present the complete structure of the code, the hardware environments to execute the code, the sample code to implement our new approach, and the method applied to validate the Monte Carlo simulation code for homogeneous media and heterogeneous media. Various metamorphic relations are also discussed in this chapter. The input file for each metamorphic relation is specified. Meanwhile, the details to carry on the experiment to identify the relationship between the height of the collection lens and the contrast value of the image are also introduced in this chapter.

In chapter four, all the experiment results of the case study are presented. The results are further investigated. The improvements and limitations of our new approach are discussed in this chapter thoroughly. At the end of the chapter, we talk about the future work that we think will provide a better understanding and deeper evaluation to our new approach.

CHAPTER 2: BACKGROUND AND HYPOTHESIS

Photon propagation in biological tissues has been an interesting topic for a long time. Different models were established to mimic this transportation. Within them, Monte Carlo modeling has been widely used to simulate various scenarios because of its nature of balancing between algorithm accuracy and computing complexity. However, testing of a Monte Carlo program modeling photon propagation in biological tissues is difficult due to the unknown character of the test oracles.

Even though software testing has developed very rapidly in this decade, testing a "non-testable" program remains as a challenge. In order to address this problem, a new testing method called self-checked metamorphic testing, which is based on the conventional metamorphic testing methodology, has been proposed in this thesis.

In this chapter, all the necessary background knowledge concerning self-checked metamorphic testing and Monte Carlo simulation of photon transportation in biological tissues will be introduced. Also, our hypothesis on the new testing method will be presented as well.

Software Testing

In 1979, Glenford Myers defined software testing as "The process of executing a program or system with the intent of finding errors." in his classic book*, The Art of Software Testing* [1]. At the time that his book was written, this definition probably was the best available and concluded perfectly the facts of software testing at seventies. Nowadays, software testing is no longer an alias for software debugging. It has become a complicated and systematic process which is involved in all the stages of the software development lifecycle. The goal of today's

software testing is not only revealing the defects of the system under test, but also measuring and improving the quality of the software being tested.

Software testing has attracted more and more attentions from various software developers as well as the researchers in computer science field. On one hand, the pervasiveness of software has kept increasing over the decades. As with some obvious instances like personal computers and smart mobile phones, software is too behind almost every gadget and device we use today at home or at work. For example, televisions, digital watches, and microwave ovens or any other kitchen equipment all have embedded software. On the other hand, more and more software has been applied to mission critical situations where single failure will cause unacceptable results. There are increasingly stringent requirements for software reliability, usability, stability, performance, and security all the time. All these facts discussed above certainly demand the rapid development of software testing methods, processes, and tools.

Taxonomy of Testing Techniques

Software testing is a complicated process and it possibly contains many technical and non-technical areas, such as documentation and management. Meanwhile, there are various types of testing techniques as well as the ways to divide these techniques. Mathur presented a framework in his book to classify the different types of testing techniques[2]. In this framework, five classifiers are used to categorize testing techniques according to different features of the testing techniques.

Based on the source of test case generation, there are several categories of testing techniques. Black-box testing usually is a testing method that generates test cases from functional requirements of the system and no knowledge of code or the structure of the program is needed. On the other hand, white-box testing is defined as generating test cases from internal

structure of the code directly. Gray-box testing is the combination of black-box and white-box testing. Model-based specification testing generates test cases according to formal model of the system and interface testing make test cases from component's interface.

According to the phase of SDLC (System Development Life Cycle) in which testing is applied, Testing techniques could be classified as unit testing, integration testing, system testing, acceptance testing, and regression testing.

Testing techniques also could be divided by the goal of specific testing activity. The existence of a variety of goals of software testing leads to different types of testing methods, for instances, functional testing, security testing, robustness testing, vulnerability testing, GUI testing and so on.

The other two classifiers in Mathur's framework are artifacts under test and test process models. Testing techniques such as client-server testing, compiler testing, operating system testing, and design testing are categorized by the artifact being tested. At the same time, waterfall model testing, V-model testing, spiral testing, and agile testing are divided by the classifier − test process models.

Even though testing techniques have been classified under different categories, they are interrelated when they are used to test any system in the real world. For example, If you test a single module of a software system, the testing technique obviously belongs to unit testing. Meanwhile, if you generate your test cases according to the system requirements, the testing technique also could be classified under black-box testing.

Black-Box Testing and White-Box Testing

Black-box testing and white-box testing probably are the most common terms when you talk about the testing techniques. As we discussed above, black-box testing generates test cases only according to the requirements. The structure of the code or the implementation of the program remains a black-box for the testers. Testing techniques like boundary value analysis, equivalence partition testing, random testing, and ad hoc testing are all belong to black-box testing.

On the other hand, white-box testing generates test cases with the aid of code structure information. Testing techniques like coverage testing, date-flow testing, and path testing are part of white-box testing. Unlike the black-box testing, white-box testing seldom generates test cases only from structural information of the code. Only if the testers give certain input to force the program undergoes certain path. Otherwise, test cases are still generated from requirement, the code is an additional artifact for test case generation.

Test Coverage Criteria

Goodenough and Gerhart pointed out in their early research that the key question for software testing is to answer "what is a test criterion?", which means software testing needs to define a criterion to constitute an adequate test [61][62]. When a test set T is used to test a program P against a selected criterion C, the test set T will be considered adequate when it satisfies criterion C. Otherwise, it will be considered as inadequate. For example, if criterion C is all-statement coverage criterion, test set T will be considered adequate when all the statements of program P are covered at least once during the execution of test set T.

Zhu *et al*. [58] discussed in their paper that adequacy of coverage criteria could act as different roles in software testing. First, it could act as stopping rules, which means testing could be stopped if coverage criteria are adequate. Second, adequacy of coverage criteria could act as

measurements, which gives a degree evaluation to different test sets instead of simple good or bad. At last, adequacy of coverage criteria could be used as test case generators. As guidelines, different test coverage criteria will require different test cases to be generated. For example, in the white-box or structural testing, testers will generate test cases by selecting specific sequence of statements or branches according to different testing criteria.

There are numerous test coverage criteria as well as the ways to categorize these criteria. In the same paper, Zhu and his coworkers presented one possible framework to group basic coverage criteria [58]. This framework was constructed based on the combination of the underlying test approach for test coverage criteria and the information source of the test coverage criteria. All the test coverage criteria are first divided into three groups: structural testing coverage criteria, fault-based testing coverage criteria, and error-based testing coverage criteria. Structural testing coverage criteria focus on the coverage of a certain set of structural elements in the program or the specification. Fault-based testing coverage criteria are adequacy criteria to measure the fault detecting ability of test sets. Error-based testing coverage criteria ask test cases to check the program in error-prone points.

For each group of test coverage criteria, they could be further divided into two sub-groups: specification-based and program-based. For instance, a structural testing coverage criterion could belong to specification-based structural testing coverage criteria or program-based structural testing coverage criteria. The specification-based criteria evaluate test sets with identified features of the specification or the requirements of the software program, while the program-based criteria require to check if certain structural elements of the program has been exercised thoroughly. There are too much information about the categories of test coverage criteria and we could not cover all of them in this section. So, only information which

directly relates our research work, thus to say, program-based structural testing coverage criteria, will be discussed in the following paragraphs. Please refer to ref. 58 and references over there for the details about other test coverage criteria.

There are also two sub-categories for program-based structural testing coverage criteria: control flow coverage criteria and data flow coverage criteria. These two sub-categories are related and somehow different. Both based on the flow-graph model of program structure, control flow coverage criteria focus on the structural elements of a program such as statements or branches and data flow coverage criteria are interested in values that associated with variables and how the associations effect the program execution. Again, Due to the large volume of information, data flow coverage criteria like all definitions criterion, all uses criterion, and all definition-use-paths criterion will not be discussed in this thesis. But control flow coverage criteria will undergo detailed discussion since it relates to our research work directly.

The main test coverage criteria for control flow coverage criteria include: *function coverage criterion*, *statement coverage criterion*, *branch coverage criterion*, *path coverage criterion, decision coverage criterion*, *condition coverage criterion, decision/condition coverage criterion,* and *multiple condition coverage criterion* [3].

Function coverage criterion requires all the interested functions are covered at least once during the execution of the test set.

Statement coverage criterion requires all the statements of the program are covered at least once during the execution of the test set.

Branch coverage criterion requires branches from all decision points are executed at least once during the execution of the test set.

Path coverage criterion requires to all possible routes of the part of the code being tested are executed at least once during the execution of the test set.

Decision coverage criterion is also known as branch decision coverage. We say a decision is covered if all possible outcomes of the decision have been taken. For example, a `if` or `while` statement has two possible outcomes, namely, `true` or `false`. This decision is considered being covered if both situations have been executed by some test input of the program under test. Decision coverage criterion requires all decisions are exercised at least once during the execution of the test set.

Condition coverage criterion has some similarity as decision coverage. Like decision coverage, a condition is considered covered if all possible outcomes of the condition have been taken. The difference is a decision could be composed with multiple conditions which are connected by logical operators, such as `and` , `or`, and `xor`. Condition coverage criterion requires all conditions are exercised at least once during the execution of the test set.

Decision/condition coverage criterion requires that both decision coverage and condition coverage criterion are satisfied at the same time. It should be noted that decision coverage or condition coverage criterion could be satisfied individually without satisfying the other criterion.

Multiple condition coverage criterion requires all possible combinations of each single condition to occur at least once during the execution of the test set. For K conditions, there are 2K possible combinations.

Some of the coverage criteria we discussed above are connected. For instance, the adequacy of branch coverage criterion subsumes the adequacy of statement coverage criterion. It means a test set must satisfy statement coverage criterion if it satisfied the branch coverage

criterion. However, this does not mean branch coverage criterion has higher fault revealing ability and the statement coverage criterion is redundant. Different test coverage criteria are selected for different circumstances and have different advantages. Statement coverage criterion is used to check the test cases in a higher level. Evaluating the adequacy of statement coverage criterion will give a quick feedback for updating the test cases. Then these test cases could be further evaluated in detail with branch coverage criterion, which will provide more precise coverage information for the code. The following diagram (Figure 2.1) shows the subsumes relationship among various control-flow based test adequacy criteria.

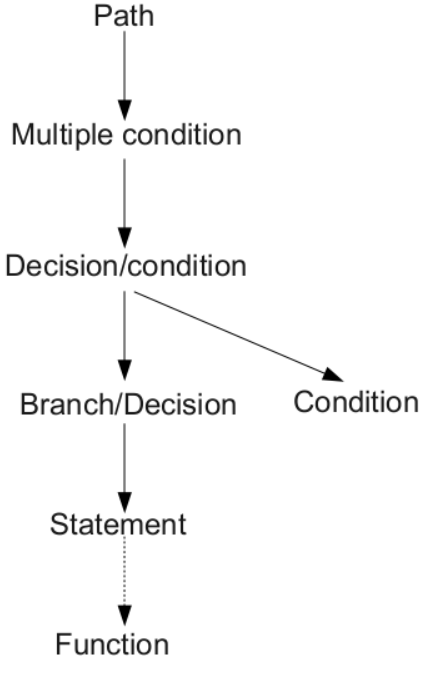

Figure 2.1. The subsumes relationship among various adequacy criteria.

In order to collect the coverage information, extra efforts need to be put into the program under testing. one simple way is testers implement extra code into the source code of the program manually and use the extra code to collect the coverage information. But this approach

is low-effective and error-prone. Since checking control flow criteria to get certain level of confidence of the software program is so common, a large number of automatic control flow coverage measurement tools have been developed in recent years. For example, an automatic tool called ATAC could measure statement, decision, and condition coverage for C programs [63]. JUnit is a coverage measurement tool which is designed for Java code. It could measure statement coverage and decision coverage [64]. Another commercial software named Clover also could provide information about statement, branch, and method (function) coverage for Java code [65].

Every piece of coverage information in the testing process needs some trade-off to be collected. More precise coverage information probably means more test cases, more testing time, and higher testing cost. For example, the adequacy of condition/decision coverage of a decision with 10 conditions needs 2 test cases. Meanwhile, the adequacy of Multiple condition coverage for the same decision needs 1024 test cases. Sometimes path coverage criteria even could not be checked due to the infinite possible paths of the program. Different coverage criteria have different focuses on the code structure. It depends on the testers' experiences to decide which type of coverage information should be examined in order to gain confidence for a specific program. It is the art of testing that balancing the cost of testing and how much confidence could gain by selecting appropriate test coverage criteria.

Non-testable Software Systems

Non-testable software system, which is another name for the system with no test oracle available. Weyuker defined two situations for "no test oracle available" in her paper [59]. One situation is an oracle doesn't exist for a program. The other situation is an oracle is potentially available, but the efforts to get the oracle are impractical. Systems such as numerical and

scientific computing system and machine learning systems [4][5] are typical "non-testable" software systems. For a “non-testable” system, it is not practical to know the correct output of an arbitrary input to the system under test [6].

In her paper, Weyuker also tried to summarize some possible solutions for testing "non-testable" software programs since giving up testing is not an option. The first solution is called pseudo-oracle or dual coding. A second programmer will be asked to write the code independently and two pieces of code will serve as the test oracle for each other. Identical sets of input will be executed for both programs and the results will be compared. Of cause, the overhead of this method is huge. It almost doubles the work for coding and testing. Another alternative is to use technique to narrow the possible results and simply accept plausible results while always keep in mind that the correctness of the output has not been approved yet. The last choice includes using limited and simplified input which the correct out put already have been known. Maybe these data could only test part of the code or the system under certain situation, but it is better than abandoning the efforts of testing completely.

We think another possible approach might be to simplify the whole system under test instead of picking up special circumstances. A testing technique called model based software testing could be used to model the non-testable system under test [60]. A certain characteristic of the non-testable system could be abstracted from the system itself and be modeled by some software models like finite state machines, statecharts, and Markov chains. The simplified model probably will have test oracle available. Instead of testing the whole original non-testable system, the model or the certain characteristic of the non-testable will be under test.

Metamorphic Testing

Metamorphic testing is a general technique for testing systems that do not have test oracles through creating testing cases based on metamorphic relations and checking the predictable relations among test outputs [4]. It is an effective approach for testing "non-testable" software systems. Instead of checking the correctness of outputs, metamorphic testing checks the metamorphic relations among test outputs. If a violation of the metamorphic relation is found, the system under test must have some faults [7]. For example, in a metamorphic testing, if test input x to a system under test generates an output $f(x)$, the metamorphic relation is used to create a transformation function t, which is applied to the input x to create another test input $t(x)$. The transformation then allows us to predict the relation between the output $f(x)$ and the output $f(t(x))$ based on the value of $f(x)$ and the metamorphic relation [8]. If the test outputs do not satisfy the metamorphic relation (*i.e.*, the relation between $f(x)$ and $f(t(x))$ is not as expected), then the system under test must have some faults.

The procedure of metamorphic testing could be concluded as follows:

(1). Identify metamorphic relations. A metamorphic relation is the relationship among the test input data and could be used to create additional test inputs based on existing one and to predict the relations among the test outputs [9].

(2). Create metamorphic test input data. Based on identified metamorphic relations, additional test inputs are generated through transforming existing test inputs.

(3). Conduct testing. Input data are executed, and their outputs are checked against the metamorphic relations. If a violation of the metamorphic relation is found, then faults must exist in the system; otherwise, more test inputs might be needed or new metamorphic relations have to be identified to test the system.

In order to test "non-testable" systems, metamorphic testing was first proposed in [4] and further investigated in [6]. Metamorphic testing has been used for testing "non-testable" systems such as bioinformatics systems [10], machine learning systems [8][11]. and online service systems [12]. Identifying metamorphic properties is one of the most challenge and important work in metamorphic testing. Six types of metamorphic relations that apply in general to machine learning systems were classified in [13], and some specific metamorphic relations extended from the general metamorphic relations for machine learning applications were discussed in [8]. A guideline for selecting good metamorphic relations that are good at detecting program faults was discussed in [9] through case studies. Test case generation based on identified metamorphic relations is another important task in metamorphic testing. Testing input data for some systems or test output data generated from those systems could be very complex. Therefore, it is very difficult to manually create test input data or to interpret test output data in metamorphic testing. Automatically generating metamorphic test inputs and comparing test outputs based on metamorphic relations is important to improve the performance and effectiveness of the metamorphic testing. One framework for automatically generating test input data and checking the metamorphic relations was discussed in [14]. Another automated metamorphic testing was discussed in [11], which demonstrated the approaches for automatically generating test inputs and checking metamorphic relations as well. In addition, an approach called heuristic metamorphic testing is presented in [11] for reducing false positives and to address some cases of non-determinism in test outputs. Although metamorphic testing can be applied to individual component, most of the work focuses on system testing by considering the properties of entire systems [4][10][11] since the output of an individual component is even more difficult to be decided in a "non-testable" program. In order to detect more errors in metamorphic

testing, integration approach of symbolic evaluation and metamorphic testing was discussed in [15].

Monte Carlo Simulation

Monte Carlo method, which is named after the famous Monte Carlo casino in Monaco by Stanislaw Ulam and John von Neumann, is a computerized mathematical technique that uses random number and probability statistics to solve problems. It is a general and stochastic method and is used to describe a big number of approaches. Any computational approach involves the algorithm that contains random numbers and repeated sampling belongs to Monte Carlo method. However these approaches probably use the same pattern to get the final results.

1. Look for the range of the possible inputs for the problem.

2. Generate random inputs within the range by applying a specified probability distribution.

3. Conduct the computation of each single input.

4. Summarize individual computational results to make the final result.

In order to remain the accuracy of the results, Monte Carlo methods, as a stochastic method, require a huge number of repeated calculations even for a simple question. This characteristic makes Monte Carlo methods severely depend on the power of computational tools. When computers become more and more powerful these days, Monte Carlo methods have had wide applications in a variety of fields, such as physical sciences [16][17], engineering [18][19], computational biology [20][21], applied statistics [22][23], finance [24][25], and business [26][27].

Modeling of Photon Transportation in Biological Tissues

The topics of photon transportation in biological tissues have been attracted intensive interests of researchers from scientific , engineering, and medical field [28][29][30]. For a long time, it has been realized that visible and near-infrared light has the ability to penetrate into biological tissues such as human skins [31][32][33]. This characteristic gives us the potential to develop noninvasive methods for diagnosis of skin illnesses by analyzing the reflected light signals. Compared to the traditional imaging methodologies like x-ray tomography, magnetic-resonance imaging, and ultrasound imaging, the new methods will provide benefits such as higher spatial resolution, lower instrumentation costs, and more safety to patients. Right now, the biggest barrier to implement this idea is the lack of accurate and fast modeling tools to quantitatively analyze the reflectance image data.

Methods Used to Model Photon Propagation

Monte Carlo modeling: Monte Carlo modeling, due to its nature of balancing between algorithm accuracy and computing complexity, has been widely used to simulate light transportation in either homogeneous or heterogeneous media [34][35]. The Monte Carlo method involves the stochastic techniques in which random numbers with desired probability distributions are used to model the turbid media that are characterized by the specific boundary value defined with optical properties of the system under investigation. The statistical property of Monte Carlo algorithm enables easy adaptation for problems with irregularly shaped structures and boundaries, which, on the other hand, is much difficult for numerical approaches instead. However, as a general statistical approach, Monte Carlo modeling normally requires to trace a large number of photons in order to achieve reasonable variance, which needs to consume a lot

of computational time and set the barrier for practical applications of this method. Fortunately, this problem could be solved by conducting a parallel computing algorithm since the tracings of individual photons are independent processes according to radiative transfer theory [36].

Maxwell equations: There are two other major approaches that are also applied to simulate light propagation in turbid media. One approach is to solve the electromagnetic wave distribution of the scatted light based on Maxwell equations with the consideration of optical heterogeneity of the media and the scales of wavelength [37]. However, the practicality of this approach is very limited. Because of the computational complexities of this approach, it would require very long computational time to obtain the solution with acceptable fineness when the size of the medium under investigation is much larger than wavelength. It is almost impossible to solve Maxwell equations with realistic boundary conditions. Even with methods like finite-difference time-domain (FDTD) approximation to greatly simplify the original Maxwell equations, the computational cost of this approach still remains high [38].

Radiative transfer equation: The other type of approaches, radiative transfer equation (RTE), has also been used to model photon transportation inside a biological tissues [39]. RTE is an equation which models the radiation energy transferring inside a tissue mathematically. The basic idea of RTE is the incident light loses its energy by medium absorption and scattering away from the light. Meanwhile, the light gains energy from other light sources (other external light source or the scattered part from the original incident light) which scatter towards the incident light. Any time, the current radiance of the incident light could be calculated according the energy loss and gain.

The complexity of RTE makes it almost impossible to be solved with realistic boundary conditions. Variety of algorithms are used to reduce the complexity of the original formula of

RTE. Within them, Diffusion theories is a well established and generally accepted approach which is used to solve the original radiative transfer equation with reasonable approximation in a lot of cases. It has attracted intense interests due to its potential to make closed-form solution very fast [40][41]. But in either case of homogeneous or heterogeneous media, the diffusion theories could lead to significant errors in the regions where the photon intensity is strongly anisotropic or optical properties change dramatically comparing with the neighboring regions, for instances, calculating light distributions with the diffusion theories near interfaces of modest or large index mismatch [42] and at small source detector distances [43]. Within these regions, the basic assumption of diffusion theories would be violated and the errors are unavoidable.

In conclusion, numerical methods like Maxwell equation, RTE, and the approximation methods of RTE are lack of flexibility when simulate the photon transportation in biological tissues. Only very simple systems or systems with certain restrictions could be solved with these methods. Otherwise, the computational cost is unacceptably high or errors will be introduced into the results. On the same time, Monte Carlo methods, as a statistical approach, are more flexible. As long as the sampling volume is good enough, they could simulate a lot more situations with reasonably accurate results than the numerical methods could do.

Validation of Monte Carlo Program Modeling Photon Propagation in Biological Tissue

Just like other scientific computing algorithms, it is very difficult to test the Monte Carlo simulation of interaction between lights and turbid media. First of all, we could not predict the results of the simulation. In fact, that is the knowledge we try to learn from the modeling. On the other hand, according to the discussion above, it is also very hard to use other algorithms to yield the suitable test oracles and indicate the expected output for a certain set of input. It is either very time-consuming or even worse in most cases − no alternative algorithm is available to generate

reliable test oracles for the modeling, especially, for the systems with irregular-shaped elements or complex boundary conditions, which probably happens all the time while modeling the real world problem.

Still, there are several approaches could be applied to validate the Monte Carlo simulation for very simple scenarios. In this section, three methods, Beer-Lambert law, van de Hulst's table, and radiative transfer Equation (RTE) would be introduced.

Beer-Lambert law: Beer-Lambert law [44], which is also known as Beer's law or Beer-Lambert-Bouguer law, is widely used to calculate the absorbance or the transmittance while light passes through homogeneous and transparent media. The law itself could be described with the following equation:

$$I_c(l) = I_c(0)\exp(-\tau) \qquad \text{(eq. 2.1)}$$

where $I_c(0)$ and $I_c(l)$ are the intensity of the incident light and the transmitted light, respectively; τ is the optical depth of the medium. For a homogeneous medium, τ could be expressed as the product of the attenuation coefficient μ_t and the thickness of the medium l. From eq. 2.1, the collimated transmittance $T_c(l)$ could be easily derived as:

$$T_c(l) = I_c(l)/I_c(0) = \exp(-\mu_t l) \qquad \text{(eq. 2.2)}$$

Where $_t$ is the sum with absorption coefficient $_a$ and scattering coefficient $_s$. Equation 2.2 could be reorganized as:

$$T_c(l) = I_c(l)/I_c(0) = \exp(-(\ _a + \ _s)l) \qquad \text{(eq. 2.3)}$$

To test the Monte Carlo simulation code, four parameters, $I_c(0)$, $_a$, $_s$ and l could be varied independently or combinatorially while the other input parameters of the code remain

unchanged. Equation 2.3 is used to generate test oracles for the Monte Carlo code. The actual results of the collimated transmittance T_c will be compared with the expected values coming from the calculation of Beer-Lambert law and the results will be evaluated for the testing purpose.

One thing we need to draw the attention here is the intensity of the light attenuates exponentially within media. As a result, the selected testing values for μ_t and the thickness of the slab couldn't be very large, otherwise, the error percentage between the result of Monte Carlo simulation and the result of Beer-Lambert law calculation will increase dramatically because of the statistical nature of Monte Carlo simulation.

Although Beer-Lambert law is a straightforward validation approach, it is an over-simplified model to mimic the light propagation within biological tissues. Because this model does not consider the situation for light scattering which happens all the time when light travels in the biological tissues.

Van de Hulst table: The heart of solving light scattering within a medium is to re-define the photon direction after each scattering event. Phase function $p(\hat{s}, \hat{s}')$ is used to describes the amount of photons scattered from the propagation direction $\hat{s}$ into direction $\hat{s}'$. There are a number of ways to possibly normalize the phase function, but the most natural treatment is to think the phase function as a probability distribution. As a result, this normalization condition requires the integral of the phase function over all angles to equal to unity.

$$1/(4\pi)\int_{4\pi} p(\hat{s},\hat{s}')d\omega = 1 \qquad \text{(eq. 2.4)}$$

where $d\omega$ is a differential solid angle in the $\hat{s}'$ direction.

In 1941, Henyey and Greenstein proposed a scattering phase function for particles in atmosphere [45]. Lately, Henyey–Greenstein phase function or some linear combinations of it have been widely adopted for calculating light scattering in biological tissues.

In order to help non-specialists who want to use the results from variant scattering theories but don't want to do the calculation themselves or spend excessive time to search the literature, H. C. van de Hulst, in 1980, published the extensive results of radiation by repeated scattering under different circumstances [46]. For the purpose of validating the Monte Carlo code, the calculated results were selected according to the scenario needed to be simulated. For example, Table 35 in [46] contains the computational results of light propagation in finite layers with different optical parameters by using Henyey–Greenstein phase function. Then, the optical parameters used to do the calculation are selected as the input parameters of the system, which Monte Carlo code tries to simulate. The calculated results will served as the test oracle, the simulation results will be compared with the calculated results to validate the Monte Carlo program.

RTE and its approximation methods: Another alternative approach to validate the Monte Carlo code involves calculation of Radiative Transfer Equation (RTE). As we discussed in previous section, The original RTE is so complex and it almost impossible to be solved with realistic boundary conditions. Different simplified algorithms based on RTE , such as diffusion approximation, are used to reduce the complexity of the original formula of RTE. At the same time, depending on the optical system under investigation, simplified boundary conditions are also selected for the purpose of generating closed-form solutions for the RTE. After the solution has been generated, the optical parameters absorption coefficient $_a$, scattering coefficient $_s$, anisotropy g, and refractive index *n* and the phantom position parameters are used to calculate

the reflectance image and the image serves as the expected values for the Monte Carlo code under testing.

Overall, these three methods have very big limitations when they are used to validate Monte Carlo code. They can only be applied for testing the Monte Carlo program that simulates photon propagation in a very simple environments such as homogeneous media. In practical, test oracles for the Monte Carlo program for heterogeneous media and media with other real boundary conditions are absent. Therefore, traditional verification methods cannot be applied to test a Monte Carlo program for heterogeneous media.

Hypothesis of Self-Checked Metamorphic Testing

Limitation of Metamorphic Testing

Although the violation of metamorphic relation exposes faults in a system, it cannot prove the absence of the faults. Satisfaction of a metamorphic relation cannot improve the confidence to the system under test. A metamorphic relation might be accidently satisfied even though the test output is incorrect. In addition, a metamorphic relation might be too weak to capture some subtle errors in a system. For example, if y is the test output of test input x, we expect test outputs of input data $(x+c)$ and $(x–c)$ both are y *(i.e.,* the metamorphic relation of the test outputs is *equal*). At many cases, test outputs from a defected system may satisfy above metamorphic relation (*such as the system has a systematic shift of output values*). Additional example likes: if z is the test output of test input x, we expect the test output r_1 of test input $(x+b)$ is larger than z, and the test output r_2 of test input $(x–b)$ is smaller than z *(*another popularly used metamorphic relation). One fault in the implementation may cause one of the modules for calculating the output of x, $(x+b)$, and $(x–b)$ was not executed (such as the missed module is used

to adjust output values), but above relation may still hold. Since metamorphic testing is an input to output approach, it has no ability to prevent all these false-positive results. Obviously, above issues can be resolved if a set of perfect metamorphic relations and test inputs are identified in the testing. However, identifying a perfect metamorphic relation set or generating a perfect test input set is an extreme challenge work. The effectiveness of metamorphic testing is highly dependent on the quality of identification of metamorphic relations and the generation of test inputs. Although there are some general frameworks for choosing metamorphic relations, complex techniques and domain knowledge are required to identify metamorphic relations. Rigorous approaches needed to ensure the quality of identified metamorphic relations and to validate test inputs and outputs.

Self-Checked Metamorphic Testing

One of the possible approaches to improve the original metamorphic testing method is to combine metamorphic testing with structural testing or other white box testing methods. In this thesis, we present a new testing method called *self-checked metamorphic testing*.

What is self-checked metamorphic testing: Self-checked metamorphic testing is a metamorphic testing approach extended with evaluation of the adequacy of testing coverage criteria during the test process. In addition to checking the metamorphic relations, several test coverage criteria are also examined. The test coverage data are used as a reference to examine the quality of test cases and metamorphic relations. Meanwhile, the test coverage data are also used as general criteria to evaluate the quality of the metamorphic testing. The code for checking the coverage is instrumented in the system under test and the test coverage will be automatically checked during the testing so that no extra steps or executions are needed to perform this new approach.

Advantages of self-checked metamorphic testing: We assume the combination of traditional metamorphic testing and structural coverage information would have following advantages.

1) Act as criteria to evaluate the identification of metamorphic relations. The information coming from white-box coverage testing could serve as general criteria to evaluate the quality of metamorphic relations. We do realize that there are a lot of possible criteria, like fault-revealing rate, could be used to evaluate a metamorphic relation. Meanwhile, the code coverage could also be a suitable candidate. The rationale behind this assumption pretty straightforward. If you want to reveal the faults in code, your test cases at least need to cover the code. A metamorphic relation r with 80% code coverage is probably better than another metamorphic relation r' with 10% code coverage. Although it has limitations, our approach provides a way to measure the quality of metamorphic relations. In order to satisfy the adequacy of some test coverage criteria, we may need to identify some additional metamorphic relations, and we may also remove some metamorphic relations due to the redundancy among them. Therefore, evaluating test coverage criteria provides a criterion for selecting metamorphic relations (*to answer the question do we have enough metamorphic relations*).

2) Act as criteria to evaluate the generation of test cases. Through checking test coverage, the necessary coverage information will be provided to testers and serve as the reference to evaluate test cases. For example, a test case t and its follow-up test cases satisfy a metamorphic relation r appropriately. All the test cases undergo a similar path, which covers 40% of the code, during the execution. From the coverage information, we may know simply increasing the number of test cases for metamorphic relation r might not improve the fault revealing possibility since they all cover the same piece of code.

Additional metamorphic relations need to be identified to further test the system. In another scenario, if all the test cases have different structural coverage but only cover 40% of the code, then, more test cases for metamorphic relation *r* might be needed. Evaluating the test coverage criteria gives us a criterion to select test cases (*to answer the question do we have enough test cases*).

3) Possibly increase fault-revealing rate. The white-box coverage testing could provide further verification for the metamorphic test cases and decrease the possibility of false positive results coming from only metamorphic testing. By checking the coverage of each test case, more information will be used to decide if the test case is successful or not. For instance, a function *f* is critical in the code and it should be called by all the test cases. If one test case and its follow-up test case satisfied a metamorphic relation but one of the test cases didn't call *f* during execution, some faults in the implementation will be revealed. There is possibility that combing metamorphic testing with coverage testing could increase the fault-revealing rate.

4) Help improve the test plan in the future. The white-box coverage information could be used as a reference to help testers make the decision for the future testing plan. For metamorphic testing, satisfaction of a metamorphic relation could provide little useful information either for the system under test or the test procedure itself. Testers could not answer questions like whether the metamorphic testing is good enough or we need more test cases or more metamorphic relations. The white-box coverage information could be used as a reference to help testers make the decision for the future testing plan. For instance, some test cases hold metamorphic relation *r* and all these test cases have 100% code coverage. Then the coverage information could be used as the evidence if the tester want to stop the testing process. On the other hand, if hundreds of test cases are

examined but only have 40% code coverage. From the coverage information, we will know more testing work, such as creating more test cases, identifying more metamorphic relations or introducing other testing methods, needs to be done in order to guarantee the correctness of the whole program.

From discussion above, we can make a summary about our new self-checked metamorphic testing method. As long as the possibility to increase the fault revealing rate, the biggest improvement of simply involving structural coverage information with original metamorphic testing is having the possibility to make metamorphic testing more practical and efficient. The coverage information could serves as the criteria to evaluate the quality of metamorphic relations, test cases, even the whole testing process. Meanwhile, structural coverage information also could act as a guideline to select metamorphic relations, create test cases, and make the test plan. According to your resource, budget, and time, redundant metamorphic relations and test cases with lower possibility to reveal faults could be put aside or tested later. The code for calculating the coverage is instrumented in the source code. Upon the system's execution, the coverage information is collected together with the metamorphic testing outputs. There is little extra efforts are required to perform the self-checked metamorphic testing comparing with the original metamorphic testing methods.

How to perform self-checking metamorphic testing: The steps of our new approach are described as follows:

1) Define metamorphic relations. Guided by experience discussed in [7][8][13] and many other papers, we first identify the metamorphic relations based on behaviors of a given problem, and then conduct metamorphic testing on the system to check these relations. The initial metamorphic relations may need to come form the domain knowledge. For

instance, In the Monte Carlo program simulating photon propagation, different configurations of the simulation are carefully set as the test inputs and the relations among the outputs of those configurations are established according to physical knowledge.

2) Generate metamorphic testing data. As soon as the metamorphic relations are identified, new test inputs can be generated through transforming existing test inputs based on metamorphic relations and other techniques. The initial test inputs may have to be created based on the characteristics of the system under test.

3) Select test coverage criteria. Based on the program structure and domain knowledge, proper structural coverage targets such as all functions and all branches are chosen.

4) Instrument code for checking test coverage criteria. Code for checking the adequacy of test coverage criteria is manually instrumented in the program under test. For function coverage, a checking statement is added immediately before the first statement in the function to be checked. For branch coverage, a checking statement is added immediately before the first statement in each branch that is decided by a decision point. Every decision point in the program has to be processed with above approach.

5) Conduct testing. Based on testing results, If structural information shows inadequate coverage, new metamorphic relations might be created and new test cases might be generated. Under that circumstance, the above 5 steps are repeated until we are satisfied with the results.

Overall, in spite of the rapid development of software testing, to test a "non-testable" software program is remaining as a fairly challenge. Due to its simple algorithm, accurate simulating results, and acceptable computational cost, Monte Carlo model has been widely used to simulate photon propagation in biological tissues. Unfortunately, the Monte Carlo programs

are typical "non-testable" software programs. It is difficult to validate and verify the correctness of the code by traditional testing techniques.

Metamorphic testing is one of the approaches which try to test "non-testable" software program. Metamorphic testing does not check the correctness of outputs. Alternatively, it checks the metamorphic relations among test outputs. But metamorphic testing has its weakness as well as its advantages. In order to improve metamorphic testing method, we propose a new approach called self-checked metamorphic testing. The structural coverage information is collected together with the metamorphic outputs. This information is used not only as general criteria to evaluate metamorphic relations and test cases, but also as general guideline to instruct the creations and selections of metamorphic relations and test cases. Through our discussion, we think our new approach could possibly let us conduct the metamorphic testing with more controllable and effective way, which will make metamorphic testing more practical for testing the real life "non-testable" programs.

CHAPTER 3: IMPLEMENTATION AND EXPERIMENTAL DETAILS

A parallel Monte Carlo simulation code, which models photon propagation in either homogeneous or heterogeneous media, was used as a case study to investigate the ability of our proposed new method - self-checked metamorphic testing. In order to help us to understand the structural information of the code, the implementation of the Monte Carlo simulation was learned carefully. And the experimental details about testing this program with various testing methods which include self-checked metamorphic testing will be introduced in this chapter as well.

Implementation of Monte Carlo Simulation

Monte Carlo simulation has been wide accepted as an accurate tool to modeling the photon propagation in biological tissues [47] - [51]. The statistical nature of Monte Carlo simulations enable them to adapt to various algorithms very quickly [52] [53].

In this thesis, we conduct an experimental study on a real-world "non-testable" program, a parallel Monte Carlo modeling program, to evaluate self-checked metamorphic testing method. This program is used for accurate and efficient modeling of reflectance images from turbid tissue phantoms. The draft code was developed in Biomedical Laser Laboratory at East Carolina University. Discovered bugs of the program were fixed before the testing. Only the necessary changes were made on the code since our goal is to examine our new approach by testing this program, instead of creating high quality software.

Algorithm of Monte Carlo Program under Testing

The algorithm of the parallel Monte Carlo program is under the framework of RTE and Fresnel's equations. The basic idea of simulating photon transportation in biological tissues with

Monte Carlo modeling is tracking along each photon's trajectory after it enters the media. For each photon, it could be scattered or absorbed when travel within a biological tissue. it will also undergo reflection or transmission on the boundary of different optical regions inside the tissue. The direction and position of the photon will change constantly. The possibility that each kind of events happens is decided by the random numbers and the environment that the photon presents. All important parameters should be tracked until the photon dies or exits the tissue. Aggregating the information of enough volume of incident photons will provide useful outputs for the system such as reflectance images.

As an example, Figure 3.1 shows the total trajectory of one photon through a homogenous medium as calculated with Monte Carlo simulation by Wang and Jacques [54]. The asterisk indicates the actual position where photon escaped from the medium.

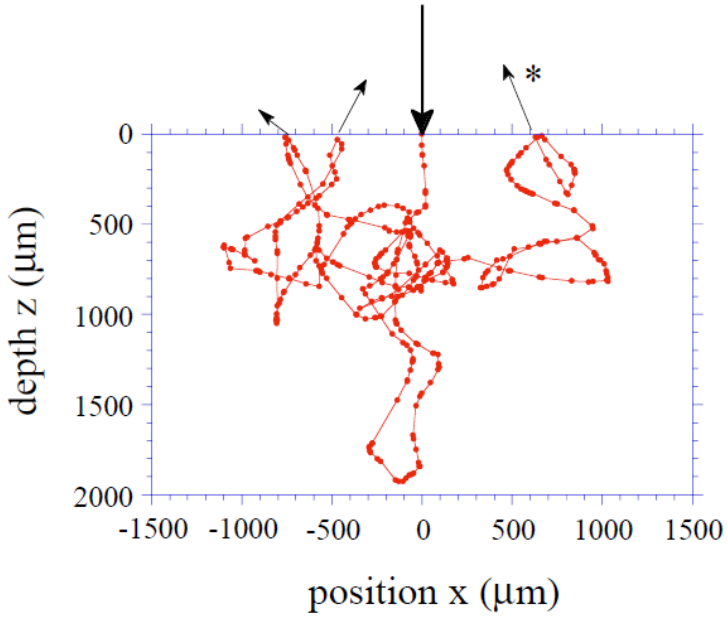

Figure 3.1. The trajectory of one photon through a homogeneous medium [54].

A more detailed algorithm about photon tracking of the parallel Monte Carlo modeling program is demonstrated in figure 3.2 with a control flow chart.

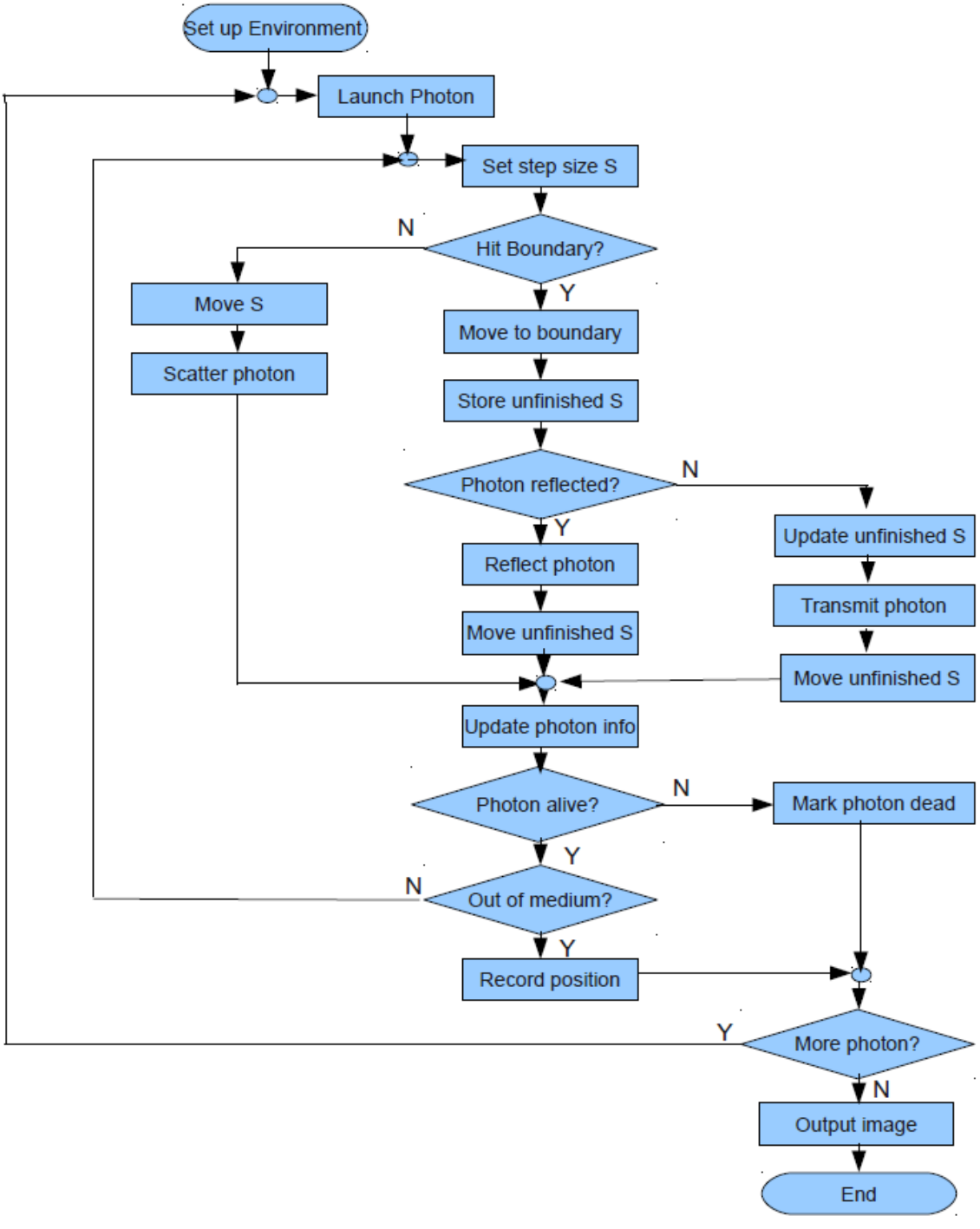

Figure 3.2. Control flow chart of the photon tracking.

The optical parameters of homogenous or heterogeneous phantom were set up by reading input file. Photon launch will record the initial photon position and direction according to incident light profile. A life path-length of the tracked photon in the region where it first comes into the phantom is also determined by the random number and the absorption coefficient (μ_a) in this step. Life length means the largest length a photon could travel before it is absorbed.

A step size S will also be decided randomly. Step size means the distance a photon could travel between two scattering events. The program will check if the photon hits the boundary of different optical regions when the photon moves S. If the photon does not hit the boundary, move the photon with full step size. Then the photon will be scattered and redirected. The direction also be determined randomly. Otherwise, if the photon hits the boundary, move the photon to the boundary. The unfinished S will be stored. On the interface, two possible events, reflection and transmission, could possibly happen. What actual occurs also depends on the random number and the optical properties of the different regions. If photon be reflected, the unfinished S will be moved towards the new direction. If transmission happens, the unfinished S will be renormalized at first. Then, the photon will be moved along the new transmitted direction with updated unfinished S distance.

After each possible movement we talked above. The photon information includes position, direction, current life path-length, and current environment will be updated. Then, photo information will be used to decide the photon status. If the current life path-length equals or is smaller than zero, the current photon will be marked as a dead photon and the track of the next photon will be started. Otherwise, the position of the photon will be checked to see if it is outside the medium we interested. If the photon still inside the phantom, a new step size will be created and the whole procedure will be repeated. The procedure will be cycled until either the photon is dead or the photon escapes from the phantom. The position where the photon comes

out of the phantom will be recoded. All the photons will be tracked following the same strategy. After that, the image will be generated.

Parallelization & Random Number Generator

In order to get satisfied results, Monte Carlo simulations require sampling large volume of photons, which demands more resource and higher computational cost. Fortunately, this barrier could be overcame by introducing parallel computing methodology into the algorithm. The independent tracking of photons makes Monte Carlo simulations ideal for parallel computing. The program employed Message Passing Interface (MPI) for parallelization of the sequential Monte Carlo code [55]. The total tracked photons are divided and grouped for distribution among different processing elements (PE). The results from each PE are aggregated to generate the code output.

the quality of the random numbers is essential to ensure the accuracy of the Monte Carlo simulation results. Random numbers used in the simulations need to be generated independently and uniformly between 0 and 1 to describe the random events. According to the nature of the light tissue interaction, there are different types of random events, for instances, scattering, absorption, and reflection from or refraction through an interface in the photon tracking process. Each type of random events should be assigned a unique sequence of random numbers in order to ensure the randomness for the Monte Carlo modeling. Therefore, random number generator needs to maintain several independent random number sequences in the Monte Carlo simulation at the same time. A large volume of random numbers, say 100 or more in most cases here, are required to track a single photon in the simulation. As a consequence, a long sequence period of random numbers are necessary to guarantee a satisfied simulation results. Meanwhile, in order to keep a high performance of the parallel code, it is preferable to generate random numbers locally

on each PE so that data communication and PE idle time will be minimized. The key issue of this implementation is to have a good random number generator to manage the coexisting random number sequences on different PEs with minimal correlations when conduct the parallel computation.

A new random number generator, which is modified from an existing one (Ran4) [56], was used in the parallel computation to address the requirements discussed above. For detailed description and discussion about the parallelization of the Monte Carlo simulation and random number generator, please consult ref [57] and the references within the paper.

Program Structure

The Monte Carlo simulation program has been developed using Fortran 90 with the Intel MPI library. The program contains five source files. *Monte_main.f90* is the main program including the code calling the MPI functions; *Monte_go.f90* includes the subroutines and functions to check if the photons hit the different optical boundaries in the turbid medium and record current photon status and position. *Monte_sub.f90* is the module for all utility subroutines that do the calculation for the simulation. *Monte_io.f90* is the file for input/output subroutines; *Monte_define.f90* contains all the definitions for objects and constants. There are about 40 subroutines or functions in the program, and the total lines of Fortran 90 code is about 1600.

Experimental Details For Testing

All the experiments conducted in this thesis focus on testing the correctness of the parallel Monte Carlo code. No non-functional requirements, like reliability and performance, are evaluated in our experiments.

The biological model this Monte Carlo program simulated is demonstrated in the Figure 3.3.

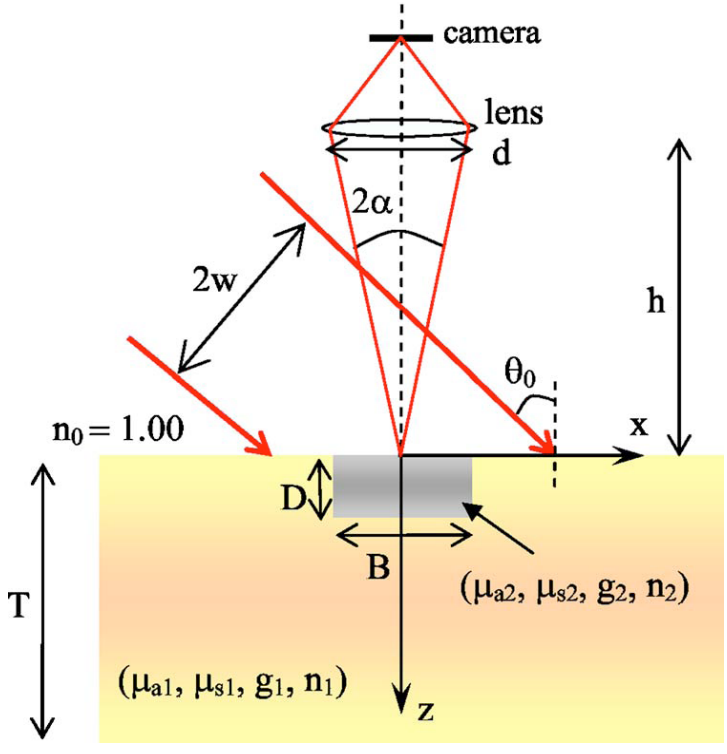

Figure 3.3. The Configuration of the incident light beam, turbid phantom, and optical system [57].

Both homogeneous and heterogeneous tissue phantoms have been considered in our simulation. The set of optical parameters were used to characterize different regions of heterogeneous phantoms includes the refractive index (n), absorption coefficient (μ_a), scattering coefficient (μ_s), and anisotropy factor (g). A uniformly distributed incident photo beam of diameter $2w$ was used to illuminated the phantom surface with a direction of (θ_0, 0). A round collection lens having diameter d and height h was applied to collect the reflected photons from the phantom and send the photons to the camera which will make the final reflectance image. For

the heterogeneous tissue, a cylinder region of height *D*, diameter *B* and optical parameters as $_{a2}$, $_{s2}$, g_2, and n_2 was embedded in a semi-infinite host medium which has optical parameters as $_{a1}$, $_{s1}$, g_1, and n_1. The thickness *T* of the host was infinite along z axis. For the homogenous system, the physical parameters of the embedded cylinder were selected as *D* = 0.0 mm, *B* = 0.0mm, while the optical parameters were kept the same as the optical parameters of phantom.

Experimental Setup

Each test case was carried on a computing cluster of 4 nodes (PowerEdge 1750, Dell) with a total of 8 process elements (Xeon 3.06 GHZ CPU, Intel). Intel MPI library was used as message passing interface. While other parameters are changed according to each test case. A two dimension (32 X 10) array was hard coded into the input file as the random number generator seeds and used for all test cases. For detailed numbers of the array, please check appendix A of this thesis.

Validation of the Monte Carlo Code for Homogenous Media

For the purpose of validating our Monte Carlo code for homogenous media, the reflected light signals were analyzed in form of angle-resolved distribution on the surface. Even through the spatially resolved distribution is needed to produce the reflectance image. The results were compared with the data from table 35 of ref [46]. Table 35 contains the computational results of light propagation in finite layers with different optical parameters by using Henyey–Greenstein phase function. There are very similar scenarios that our Monte Carlo code tries to simulate.

Different sets of incident light deflection angle θ_0, absorption coefficient $_a$, scattering coefficient $_s$, anisotropy g, and slab thickness *l* were selected according to table 35 of Van de

Hulst's book [46]. The incident light azimuthal angle φ remains as [0, 2π]. these sets of data were used as the inputs for the Monte Carlo code. The other input parameters remain unchanged. The reflectance *R* from Monte Carlo simulation was converted into an angle-resolved bidirectional form $R_b(\theta_i, \theta_0)$ by averaging R and multiplying by $1/(\cos\theta_i \sin\theta_i \Delta\theta)$ while θ_i is $[(i-1)\Delta\theta, i\,\Delta\theta]$, with $\Delta\theta = \pi/(2M)$ and i = 1, 2,...M (M = 30 in our validations). Then the function $R_b(\theta_i, \theta_0)$ could be compared with the bidirectional reflection function $R_b(\mu, \mu_0)$ from van de Hulst's table [46]. Very similar procedure could also be applied to compare the bidirectional transmission function T_b from Monte Carlo simulation and from van de Hulst's table [46].

To mimic this homogenous system, the physical parameters of the embedded cylinder were selected as *D* = 0.0 mm, *B* = 0.0mm, while the optical parameters were kept the same as the optical parameters of phantom. The field of view was set as 41.2 mm X 41.2 mm, which covered evenly with 201 X 201 grid cells. To convert the simulation results to angle-resolved form, the reflected or transmitted angle was divided by 30, which makes the $\Delta\theta = 3^o$.

Two test sets (test set HM-V-1 and test set HM-V-2) with different phantom optical parameters were selected to perform this validation. Each test set contains four test case. The detailed test cases were shown in Table 3.1 and Table 3.2. In order to save some space and improve the readability, we only present here the parameters which will affect the validation. Other parameters, like the ones we discussed in the above paragraph, remain unchanged in all test cases and will be neglected from the table. The output for both bidirectional reflection and transmission function will be checked versus exit angle. The results will be compared with the calculated results from ref. 46.

	HM-V-1-T1	HM-V-1-T2	HM-V-1-T3	HM-V-1-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.00	1.00	1.00	1.00
$_{a1}$ (mm^{-1})	0.50	0.50	0.50	0.50
$_{s1}$ (mm^{-1})	0.95	0.95	0.95	0.95
g_1	0.75	0.75	0.75	0.75
d (mm)	0.80	0.80	0.80	0.80
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.00	1.00	1.00	1.00
$_{a2}$ (mm^{-1})	0.50	0.50	0.50	0.50
$_{s2}$ (mm^{-1})	0.95	0.95	0.95	0.95
g_2	0.75	0.75	0.75	0.75
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.00	0.00	0.00	0.00
r	0.00	0.00	0.00	0.00
h	0.00	0.00	0.00	0.00
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface)				
total number	$1.26*10^7$	$1.26*10^7$	$1.26*10^7$	$1.26*10^7$
θ (°)	0.00	45.5730	72.5424	84.2608
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00

Table 3.1. input data for test set HM-V-1.

	HM-V-2-T1	HM-V-2-T2	HM-V-2-T3	HM-V-2-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.00	1.00	1.00	1.00
$_{a1}$ (mm^{-1})	4.00	4.00	4.00	4.00
$_{s1}$ (mm^{-1})	6.00	6.00	6.00	6.00
g_1	0.50	0.50	0.50	0.50
d (mm)	0.10	0.10	0.10	0.10
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.00	1.00	1.00	1.00
$_{a2}$ (mm^{-1})	4.00	4.00	4.00	4.00
$_{s2}$ (mm^{-1})	6.00	6.00	6.00	6.00
g_2	0.50	0.50	0.50	0.50
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.00	0.00	0.00	0.00
r	0.00	0.00	0.00	0.00
h	0.00	0.00	0.00	0.00
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface)				
total number	$1.26*10^7$	$1.26*10^7$	$1.26*10^7$	$1.26*10^7$
θ (°)	0.00	45.5730	72.5424	84.2608
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00

Table 3.2. input data for test set HM-V-2.

Self-Checked Metamorphic Testing

To perform self-checked metamorphic testing, two structural test coverage criteria, *function coverage* and *branch coverage*, were checked for all the subroutines and functions of the program. Statements for evaluating the test coverage criteria are instrumented into the program under test so that test coverage information can be generated whenever a test input is executed. A sample code was shown in Figure 3.4.

```
subroutine HitOutSideCy(Pos_p,Dir,s,r,Reg,Hit,s_go,s_left)
IMPLICIT NONE
real(8), intent(in) :: Pos_p(3), Dir(3), s
real(8), intent(in) :: r, Reg(2)
real(8), intent(out) :: s_go, s_left
integer(1), intent(out) :: Hit
real(8) :: Pos(3), Pos2(3), dist(2),t1,t2,temp
     output%fc(1)=output%fc(1) +1
!change coordinates
Pos=Pos_p
if ((cy%Pos(1).NE.0.0).OR.(cy%Pos(2).NE.0.0)) then
  output%branch(1)=output%branch(1) +1
Pos(1)=Pos_p(1)-cy%Pos(1)
Pos(2)=Pos_p(2)-cy%Pos(2)
        ...
```

Figure 3.4. Sample code for implementation of checking coverage information

Two global arrays are used to store the coverage information. The statement to evaluate function coverage is the first executable statement of each function. The statement to evaluate branch coverage was coded right after each decision point. The number of functions and

branches was grouped by different modules of the program. Since *monte_main* module only have several function calls and *monte_define* module only contains all the definitions for the variables, both will not affect the results of the simulation. So, only module *monte_go*, *monte_sub*, and *monte_io* have been checked with structural coverage information. For module *monte_go*, there are 14 functions and 166 branches. Module *monte_sub* has 14 functions and 56 branches; while *monte_io* module has 13 function and 30 branches.

According ref [57], The contrast C was defined as:

$$C = (Rc - Rp) / (Rc + Rp) \qquad \text{(eq. 3.1)}$$

Where *Rc* is the averaged reflectance over a circle of 3 mm radius which centered at the origin in the image and *Rp* is the averaged reflectance over a concentric ring with 5 and 11 mm as the inner and outer radiuses which also centered at the origin in the image. Meanwhile, in order to reduce the variance in the simulation, averaged reflectance *R(x,0)* was calculated by averaged the photon density along the *y* axis over three rows of grid cells on each side of the *x* axis [57].

The metamorphic relations selected in this research were referred the results in [57]. We assumed these relations are correct so that we could perform the metamorphic testing. 5 metamorphic relations (MR) are selected for the testing. Some of these metamorphic relations are easily to be validated by physics theories or experiments, but some of them are difficult to be validated. For each metamorphic relation, at least two test sets with different inputs were examined to lower the possibility that satisfies the metamorphic relations accidentally by special inputs.

For semi-infinite (only consider $z \geq 0$) heterogeneous media, which the parallel Monte Carlo code simulated, phantoms thickness is infinite ($T \rightarrow \infty$). In simulations, T was set to

100mm. For all the test cases in this study, we used the following parameters: w = 12.5 mm, d = 25 mm, FOV = 41.2 mm X 41.2 mm, and 201 X 201 grid cells on the surface. The details for metamorphic relations and test sets are presented below. Just like the test cases for homogenous media validation, only parameters related to phantom, cylinder, and incident light are displayed in the input data tables. The structural information was collected according to the modules.

Metamorphic relation 1 (MR1): Metamorphic relation 1 is summarized as: **Contrast C value decreases when refractive index n2 value increases.** The input data for the two test sets (HT-MR1-1 and HT-MR1-2) are shown in the Table 3.3 and Table 3.4.

	HT-MR1-1-T1	HT-MR1-1-T2	HT-MR1-1-T3	HT-MR1-1-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.40	1.40	1.40	1.40
$_{a1}$ (mm^{-1})	0.30	0.30	0.30	0.30
$_{s1}$ (mm^{-1})	5.50	5.50	5.50	5.50
g_1	0.80	0.80	0.80	0.80
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.36	1.40	1.44	1.48
$_{a2}$ (mm^{-1})	0.15	0.15	0.15	0.15
$_{s2}$ (mm^{-1})	6.00	6.00	6.00	6.00
g_2	0.80	0.80	0.80	0.80
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.375	0.375	0.375	0.375
r	4.00	4.00	4.00	4.00
h	0.75	0.75	0.75	0.75
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	30.00	30.00	30.00	30.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	0.00	0.00	0.00	0.00

Table 3.3. input data for test set HT-MR1-1.

	HT-MR1-2-T1	HT-MR1-2-T2	HT-MR1-2-T3	HT-MR1-2-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.40	1.40	1.40	1.40
$_{a1}$ (mm^{-1})	0.30	0.30	0.30	0.30
$_{s1}$ (mm^{-1})	5.50	5.50	5.50	5.50
g_1	0.80	0.80	0.80	0.80
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.38	1.42	1.46	1.50
$_{a2}$ (mm^{-1})	2.00	2.00	2.00	2.00
$_{s2}$ (mm^{-1})	4.00	4.00	4.00	4.00
g_2	0.80	0.80	0.80	0.80
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.375	0.375	0.375	0.375
r	4.00	4.00	4.00	4.00
h	0.75	0.75	0.75	0.75
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	0.00	0.00	0.00	0.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	0.00	0.00	0.00	0.00

Table 3.4. input data for test set HT-MR1-2.

Metamorphic relation 2 (MR2): Metamorphic relation 2 is summarized as: **Contrast C value decreases when anisotropy factor g_2 value increases.** The input data for the two test sets (HT-MR2-1 and HT-MR2-2) are shown in the Table 3.5 and Table 3.6.

	HT-MR2-1-T1	HT-MR2-1-T2	HT-MR2-1-T3	HT-MR2-1-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.40	1.40	1.40	1.40
$_{a1}$ (mm^{-1})	0.30	0.30	0.30	0.30
$_{s1}$ (mm^{-1})	5.50	5.50	5.50	5.50
g_1	0.80	0.80	0.80	0.80
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.40	1.40	1.40	1.40
$_{a2}$ (mm^{-1})	0.02	0.02	0.02	0.02
$_{s2}$ (mm^{-1})	4.00	4.00	4.00	4.00
g_2	0.10	0.30	0.60	0.90
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.375	0.375	0.375	0.375
r	4.00	4.00	4.00	4.00
h	0.75	0.75	0.75	0.75
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	30.00	30.00	30.00	30.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	0.00	0.00	0.00	0.00

Table 3.5. input data for test set HT-MR2-1.

	HT-MR2-2-T1	HT-MR2-2-T2	HT-MR2-2-T3	HT-MR2-2-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.40	1.40	1.40	1.40
$_{a1}$ (mm^{-1})	0.30	0.30	0.30	0.30
$_{s1}$ (mm^{-1})	5.50	5.50	5.50	5.50
g_1	0.80	0.80	0.80	0.80
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.40	1.40	1.40	1.40
$_{a2}$ (mm^{-1})	1.20	1.20	1.20	1.20
$_{s2}$ (mm^{-1})	6.00	6.00	6.00	6.00
g_2	0.20	0.40	0.70	1.00
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.375	0.375	0.375	0.375
r	4.00	4.00	4.00	4.00
h	0.75	0.75	0.75	0.75
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	0.00	0.00	0.00	0.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	0.00	0.00	0.00	0.00

Table 3.6. input data for test set HT-MR2-2.

Metamorphic relation 3 (MR3): Metamorphic relation 1 is summarized as: **Contrast C value increases when albedo α_2 value increases,** where $\alpha = \mu_s/(\mu_s + \mu_a)$. The input data for the two test sets (HT-MR3-1 and HT-MR3-2) are shown in the Table 3.7 and Table 3.8.

	HT-MR3-1-T1	HT-MR3-1-T2	HT-MR3-1-T3	HT-MR3-1-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.40	1.40	1.40	1.40
$_{a1}$ (mm^{-1})	0.30	0.30	0.30	0.30
$_{s1}$ (mm^{-1})	5.50	5.50	5.50	5.50
g_1	0.80	0.80	0.80	0.80
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.40	1.40	1.40	1.40
$_{a2}$ (mm^{-1})	0.20	1.00	2.50	5.00
$_{s2}$ (mm^{-1})	5.00	5.00	5.00	5.00
g_2	0.80	0.80	0.80	0.80
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.375	0.375	0.375	0.375
r	4.00	4.00	4.00	4.00
h	0.75	0.75	0.75	0.75
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	30.00	30.00	30.00	30.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	0.00	0.00	0.00	0.00

Table 3.7. input data for test set HT-MR3-1.

	HT-MR3-2-T1	HT-MR3-2-T2	HT-MR3-2-T3	HT-MR3-2-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.50	1.50	1.50	1.50
$_{a1}$ (mm^{-1})	0.82	0.82	0.82	0.82
$_{s1}$ (mm^{-1})	5.50	5.50	5.50	5.50
g_1	0.80	0.80	0.80	0.80
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.50	1.50	1.50	1.50
$_{a2}$ (mm^{-1})	0.02	0.50	1.20	2.00
$_{s2}$ (mm^{-1})	3.00	3.00	3.00	3.00
g_2	0.80	0.80	0.80	0.80
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.375	0.375	0.375	0.375
r	4.00	4.00	4.00	4.00
h	0.75	0.75	0.75	0.75
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	0.00	0.00	0.00	0.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	0.00	0.00	0.00	0.00

Table 3.8. input data for test set HT-MR3-2.

Metamorphic relation 4 (MR4): Metamorphic relation 1 is summarized as: **For each pixel along the x axis P(x,0) on the image, the averaged reflectance R(x,0) will decrease if the numerical aperture (NA) decreases,** where NA = sinα. The input data for the two test sets (HT-MR4-1 and HT-MR4-2) are shown in the Table 3.9 and Table 3.10.

	HT-MR4-1-T1	HT-MR4-1-T2	HT-MR4-1-T3	HT-MR4-1-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.50	1.50	1.50	1.50
$_{a1}$ (mm^{-1})	0.20	0.20	0.20	0.20
$_{s1}$ (mm^{-1})	4.00	4.00	4.00	4.00
g_1	0.80	0.80	0.80	0.80
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.50	1.50	1.50	1.50
$_{a2}$ (mm^{-1})	2.00	2.00	2.00	2.00
$_{s2}$ (mm^{-1})	3.00	3.00	3.00	3.00
g_2	0.80	0.80	0.80	0.80
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.375	0.375	0.375	0.375
r	4.00	4.00	4.00	4.00
h	0.75	0.75	0.75	0.75
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	0.00	0.00	0.00	0.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	0.00	3.35	12.50	46.65

Table 3.9. input data for test set HT-MR4-1.

	HT-MR4-2-T1	HT-MR4-2-T2	HT-MR4-2-T3	HT-MR4-2-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.50	1.50	1.50	1.50
$_{a1}$ (mm^{-1})	0.20	0.20	0.20	0.20
$_{s1}$ (mm^{-1})	4.00	4.00	4.00	4.00
g_1	0.80	0.80	0.80	0.80
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.50	1.50	1.50	1.50
$_{a2}$ (mm^{-1})	0.15	0.15	0.15	0.15
$_{s2}$ (mm^{-1})	4.00	4.00	4.00	4.00
g_2	0.80	0.80	0.80	0.80
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.375	0.375	0.375	0.375
r	4.00	4.00	4.00	4.00
h	0.75	0.75	0.75	0.75
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	0.00	0.00	0.00	0.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	1.094	7.217	21.65	70.89

Table 3.10. input data for test set HT-MR4-2.

Metamorphic relation 5 (MR5): Metamorphic relation 1 is summarized as: **For each pixel along the x axis P(x,0) on the image, the averaged reflectance R(x,0) will decrease if the incident light angle θ_0 increases.** Both semi-finite homogeneous medium and semi-finite heterogeneous phantom were tested for metamorphic relation 5. The input data of the two test sets (HM-MR5-1 and HM-MR5-2) for semi-finite homogeneous medium are shown in the Table 3.11 and Table 3.12. The input data of the two test sets (HT-MR5-1 and HT-MR5-2) for semi-finite heterogeneous phantom are shown in the Table 3.13 and Table 3.14.

	HM-MR5-1-T1	HM-MR5-1-T2	HM-MR5-1-T3	HM-MR5-1-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.40	1.40	1.40	1.40
$_{a1}$ (mm^{-1})	0.20	0.20	0.20	0.20
$_{s1}$ (mm^{-1})	4.00	4.00	4.00	4.00
g_1	0.80	0.80	0.80	0.80
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.40	1.40	1.40	1.40
$_{a2}$ (mm^{-1})	0.2	0.2	0.2	0.2
$_{s2}$ (mm^{-1})	4.00	4.00	4.00	4.00
g_2	0.80	0.80	0.80	0.80
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.00	0.00	0.00	0.00
r	0.00	0.00	0.00	0.00
h	0.00	0.00	0.00	0.00
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	0.00	15.00	45.00	75.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	0.00	0.00	0.00	0.00

Table 3.11. input data for test set HM-MR5-1.

	HM-MR5-2-T1	HM-MR5-2-T2	HM-MR5-2-T3	HM-MR5-2-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.20	1.20	1.20	1.20
$_{a1}$ (mm^{-1})	0.05	0.05	0.05	0.05
$_{s1}$ (mm^{-1})	3.00	3.00	3.00	3.00
g_1	0.60	0.60	0.60	0.60
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.20	1.20	1.20	1.20
$_{a2}$ (mm^{-1})	0.05	0.05	0.05	0.05
$_{s2}$ (mm^{-1})	3.00	3.00	3.00	3.00
g_2	0.60	0.60	0.60	0.60
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.00	0.00	0.00	0.00
r	0.00	0.00	0.00	0.00
h	0.00	0.00	0.00	0.00
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	5.00	30.00	60.00	85.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	0.00	0.00	0.00	0.00

Table 3.12. input data for test set HM-MR5-2.

	HT-MR5-1-T1	HT-MR5-1-T2	HT-MR5-1-T3	HT-MR5-1-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.40	1.40	1.40	1.40
$_{a1}$ (mm^{-1})	0.20	0.20	0.20	0.20
$_{s1}$ (mm^{-1})	4.00	4.00	4.00	4.00
g_1	0.80	0.80	0.80	0.80
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.40	1.40	1.40	1.40
$_{a2}$ (mm^{-1})	1.20	1.20	1.20	1.20
$_{s2}$ (mm^{-1})	6.00	6.00	6.00	6.00
g_2	0.80	0.80	0.80	0.80
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.375	0.375	0.375	0.375
r	4.00	4.00	4.00	4.00
h	0.75	0.75	0.75	0.75
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	0.00	0.00	0.00	0.00
x0	0.00	0.00	0.00	0.00
y0	0.00	15.00	45.00	75.00
height (mm)	0.00	0.00	0.00	0.00

Table 3.13. input data for test set HT-MR5-1.

	HT-MR5-2-T1	HT-MR5-2-T2	HT-MR5-2-T3	HT-MR5-2-T4
Parameters for phantom (d is the thickness of the phantom)				
n_1	1.20	1.20	1.20	1.20
$_{a1}$ (mm^{-1})	0.05	0.05	0.05	0.05
$_{s1}$ (mm^{-1})	3.00	3.00	3.00	3.00
g_1	0.60	0.60	0.60	0.60
d (mm)	100	100	100	100
Parameters for cylinder ((x, y, z) is the center position of the cylinder, r = B/2, h = D)				
n_2	1.20	1.20	1.20	1.20
$_{a2}$ (mm^{-1})	1.20	1.20	1.20	1.20
$_{s2}$ (mm^{-1})	4.00	4.00	4.00	4.00
g_2	0.60	0.60	0.60	0.60
x	0.00	0.00	0.00	0.00
y	0.00	0.00	0.00	0.00
z	0.375	0.375	0.375	0.375
r	4.00	4.00	4.00	4.00
h	0.75	0.75	0.75	0.75
Photon profile (θ is the incident angle of photons, (x0, y0) is the position of the center of the incident beam on the surface, height is the height of collection lens)				
total number	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$	$1.13*10^8$
θ (º)	5.00	30.00	60.00	85.00
x0	0.00	0.00	0.00	0.00
y0	0.00	0.00	0.00	0.00
height (mm)	0.00	0.00	0.00	0.00

Table 3.14. input data for test set HT-MR5-2.

Relationship Between Image Contrast and Height of Collection Lens

It is very interesting to see the relationship between image contrast and the height of collection lens. This information may help physicists build light collection system when they conduct real experiments to obtain reflectance images.

The parallel Monte Carlo program was used to simulate the relationship between image contrast and the height of collection lens. The biological system the program tried to model is identical to the system used for testing purpose. The hardware environment of running this experiment is unchanged as testing experiments. The input data are shown in Figure 3.15. All the symbols remain the same physical definitions as they are in the testing section.

Phantom					Cylinder									Light profile			
n1	μ_{a1} mm^{-1}	μ_{s1} mm^{-1}	g_1	d (mm)	n_2	μ_{a2} mm^{-1}	μ_{s2} mm^{-1}	g_2	x	y	z	r	h	Total number	θ (°)	x0	y0	height (mm)
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	0
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	1
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	2
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	3
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	4
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	5
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	6
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	7
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	8
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	9
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	12
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	15
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	18
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	21
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	24
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	30
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	40
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	50
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	70
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	100
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	130
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	150
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	170
1.50	0.20	4.00	0.80	100	1.50	2.00	4.00	0.80	0.00	0.00	0.375	4.00	0.75	$1.26*10^7$	30.00	0.00	0.00	200

Table 3.15. input data for contrast and height relationship experiment.

Overall, the algorithm and the implementation of the parallel Monte Carlo modeling program was introduced with details in this chapter. The configuration of the optical system under simulation and the experimental procedures for both Van de Hulst table validation and self-checking metamorphic testing are all presented as well. Test cases were grouped as test sets based on the metamorphic relations. The input data for each test case were shown with tables for clear reading.

CHAPTER 4: RESULTS AND DISCUSSION

In this chapter, we present the simulation results for both validation of the Monte Carlo code for homogenous media and self-checked metamorphic testing as well as the structural information of each test case for the latter. At the end of experimental results section, the results of the change of the collection lens height against image contrast are demonstrated. All the results will be evaluated and discussed carefully in the discussion section. Conclusion and future work will also be made to close this chapter.

Experimental Results

Validation of the Monte Carlo Code for Homogenous Media

This validation is based on Table 35 on Van de Hulst's book [46]. The parameters are selected according to the table value. For easy comparison , the simulation results of each test cases and the calculated values from Table 35 are plotted together on the same diagram. The diagram would undergo a visual inspection.

Results for test set HM-V-1: this test set contains four test cases. The results of individual test case are shown in Figure 4.1 to Figure 4.4 respectively.

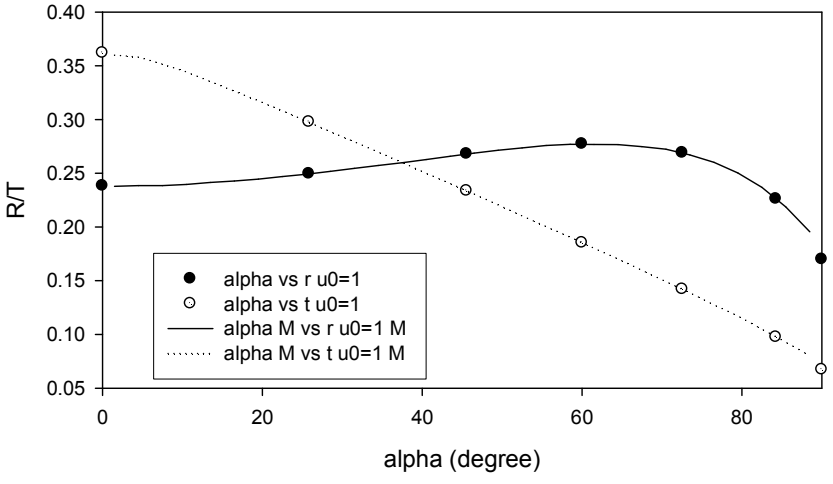

Figure 4.1. Simulation results of test case HM-V-1-T1. The lines are the bidirectional reflection / transmission functions calculated by the simulation while the symbols are the corresponding values from RTE calculation of Table 35 [46].

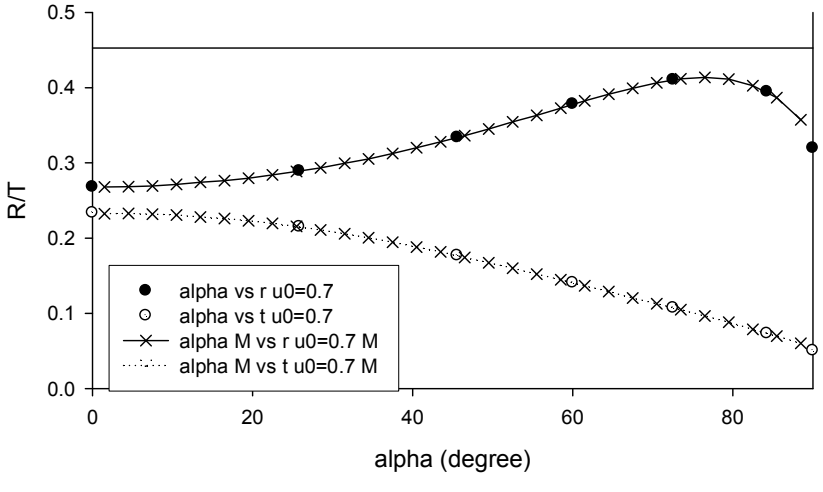

Figure 4.2. Simulation results of test case HM-V-1-T2. The lines are the bidirectional reflection / transmission functions calculated by the simulation while the symbols are the corresponding values from RTE calculation of Table 35 [46].

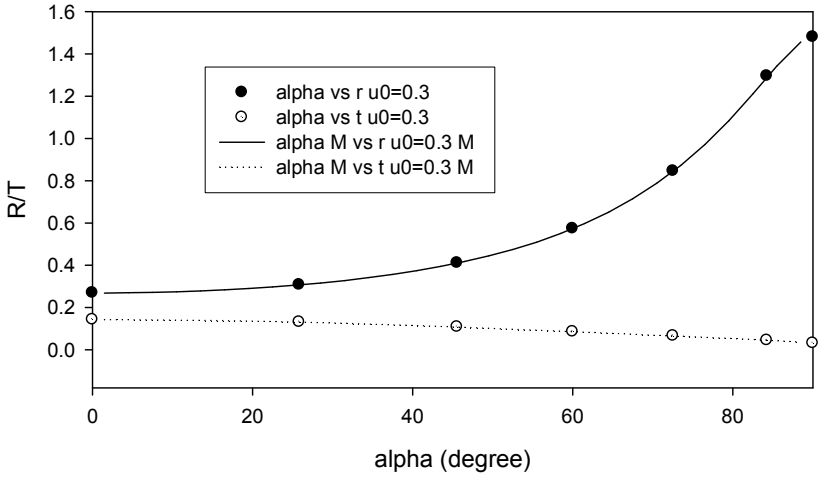

Figure 4.3. Simulation results of test case HM-V-1-T3. The lines are the bidirectional reflection / transmission functions calculated by the simulation while the symbols are the corresponding values from RTE calculation of Table 35 [46].

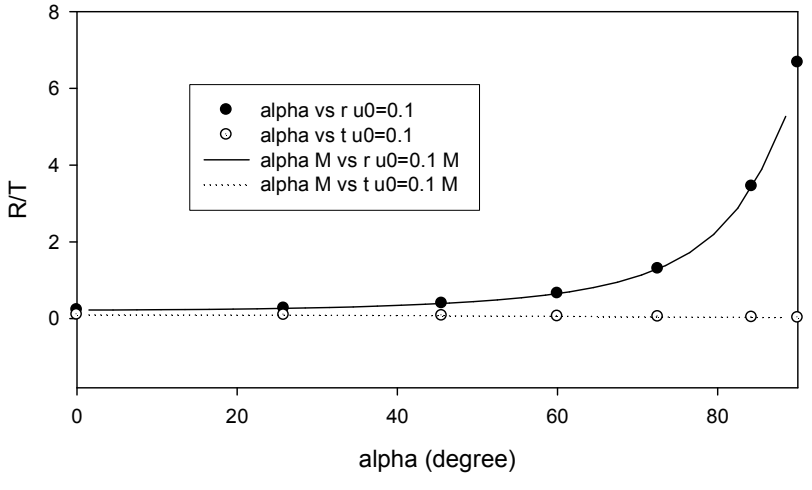

Figure 4.4. Simulation results of test case HM-V-1-T4. The lines are the bidirectional reflection / transmission functions calculated by the simulation while the symbols are the corresponding values from RTE calculation of Table 35 [46].

Results for test set HM-V-2: this test set also contains four test cases. The results of individual test case are shown in Figure 4.5 to Figure 4.8 respectively.

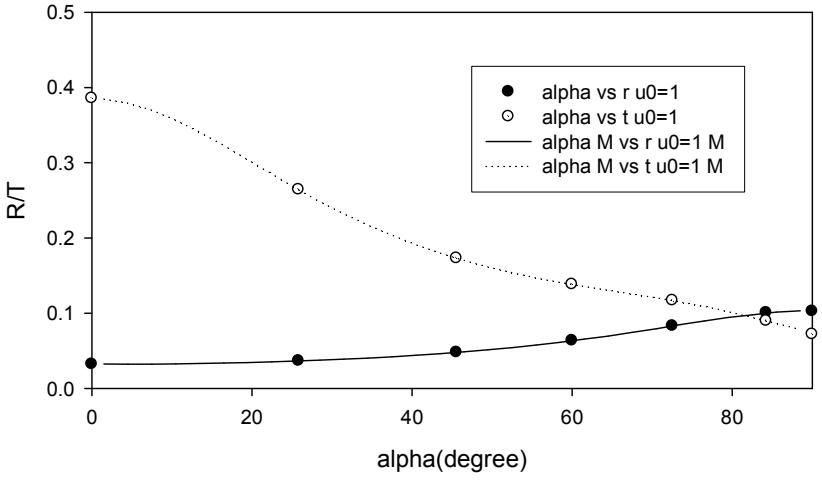

Figure 4.5. Simulation results of test case HM-V-2-T1. The lines are the bidirectional reflection / transmission functions calculated by the simulation while the symbols are the corresponding values from RTE calculation of Table 35 [46].

.

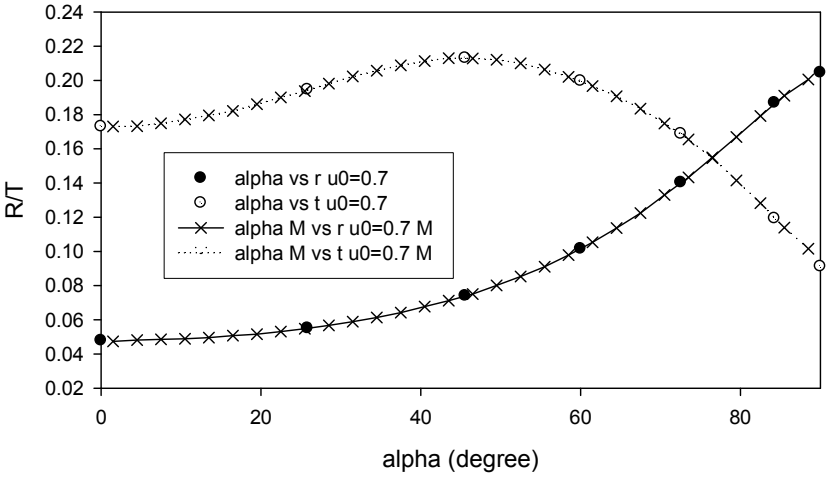

Figure 4.6. Simulation results of test case HM-V-2-T2. The lines are the bidirectional reflection / transmission functions calculated by the simulation while the symbols are the corresponding values from RTE calculation of Table 35 [46].

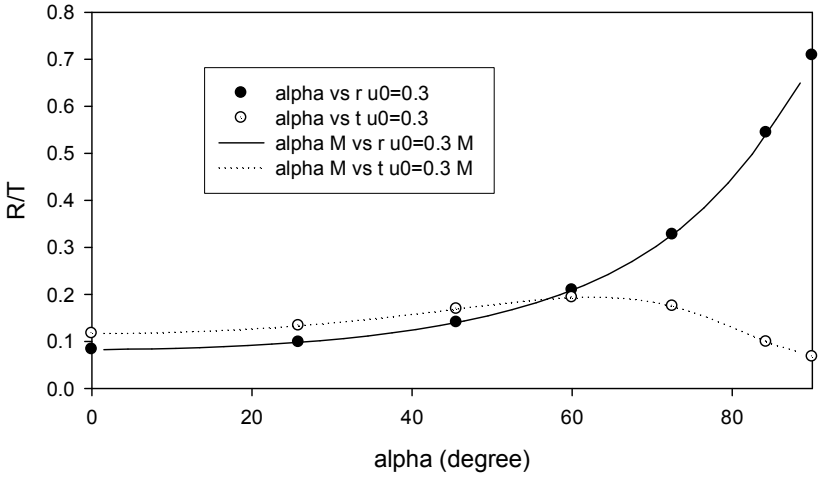

Figure 4.7. Simulation results of test case HM-V-2-T3. The lines are the bidirectional reflection / transmission functions calculated by the simulation while the symbols are the corresponding values from RTE calculation of Table 35 [46].

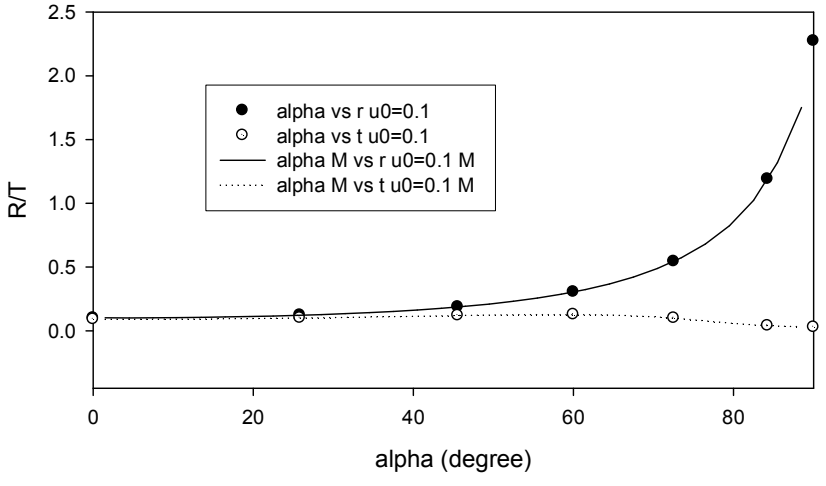

Figure 4.8. Simulation results of test case HM-V-2-T4. The lines are the bidirectional reflection / transmission functions calculated by the simulation while the symbols are the corresponding values from RTE calculation of Table 35 [46].

Self-Checked Metamorphic Testing

Five metamorphic relations (MR1-5) have been selected to perform self-checked metamorphic testing. Except MR5, each metamorphic relation contains two test sets. MR5 has four test sets instead. Each test set made with four test cases. Totally, 5 metamorphic relations, 12 test sets, and 48 test cases have been examined.

The simulation results are summarized with structural coverage information. Meanwhile, the optical parameter which is related to the metamorphic relation is presented again with the simulation results for easy comparison. Although the numeric results of the simulations could be compared directly, the results are plotted for an easier and quicker checking.

For the structural information within the tables, F-go, F-sub, and F-io means the coverage of all functions or routines in module *monte_go*, *monte_sub* and *monte_io* respectively. B-go,

B-sub, and B-io means the coverage of all branches in module *monte_go*, *monte_sub* and *monte_io* respectively. For both function and branch coverage information, the actual number of functions and branches being covered for each test case is presented as well as the percentage in the corresponding module.

Simulation results of MR1: MR1 states as contrast value decreases when n_2 value increases. There are two test sets are used to examine MR1. Each test set contains four test cases.

For test set HT-MR1-1, the results of individual test case are shown in Table 4.1. Also, the results of contrast C are plotted against n_2 and shown in Figure 4.9. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	**n_2**	**Contrast C**	**F-go**	**F-sub**	**F-io**	**B-go**	**B-sub**	**B-io**
HT-MR1-1-T1	1.36	0.19589	14 100%	14 100%	13 100%	126 75.9%	34 60.7%	27 90%
HT-MR1-1-T2	1.40	0.175588	14 100%	14 100%	13 100%	124 74.7%	34 60.7%	26 86.7%
HT-MR1-1-T3	1.44	0.162571	14 100%	14 100%	13 100%	127 76.5%	34 60.7%	26 86.7%
HT-MR1-1-T4	1.48	0.148713	14 100%	14 100%	13 100%	127 76.5%	34 60.7%	26 86.7%

Table 4.1. Simulation results of test set HT-MR1-1.

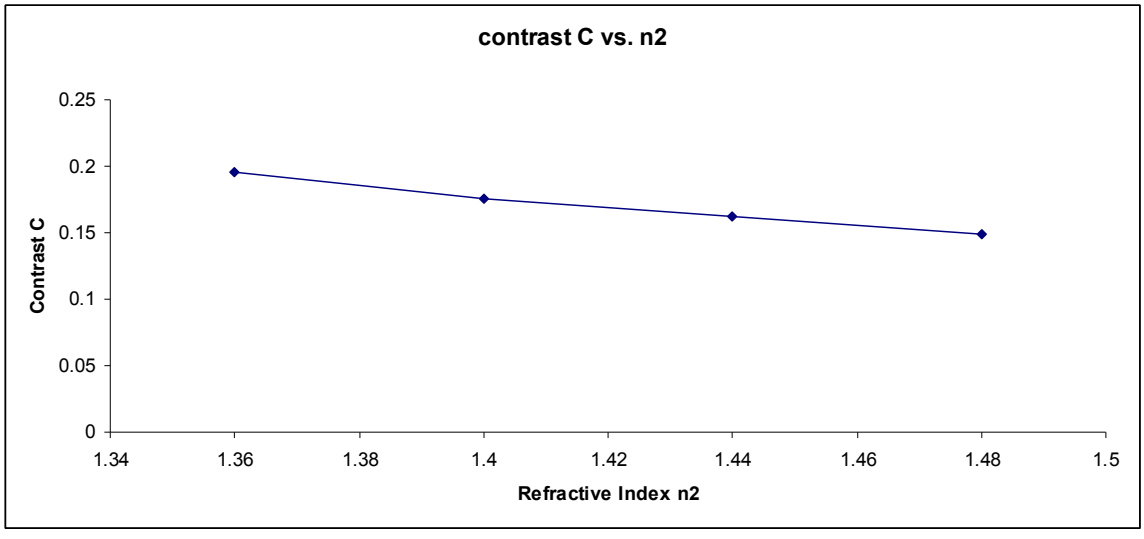

Figure 4.9. Results of contrast C plotted against n_2 for test set HT-MR1-1.

For test set HT-MR1-2, the results of individual test case are shown in Table 4.2. Also, the results of contrast C are plotted against n_2 and shown in Figure 4.10. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	**n_2**	**Contrast C**	**F-go**	**F-sub**	**F-io**	**B-go**	**B-sub**	**B-io**
HT-MR1-1-T1	1.38	-0.794871	14 100%	14 100%	13 100%	127 76.5%	34 60.7%	26 86.7%
HT-MR1-1-T2	1.42	-0.802847	14 100%	14 100%	13 100%	129 77.7%	34 60.7%	26 86.7%
HT-MR1-1-3	1.46	-0.817642	14 100%	14 100%	13 100%	129 77.7%	34 60.7%	26 86.7%
HT-MR1-1-T4	1.50	-0.82838	14 100%	14 100%	13 100%	129 77.7%	34 60.7%	26 86.7%

Table 4.2. Simulation results of test set HT-MR1-2.

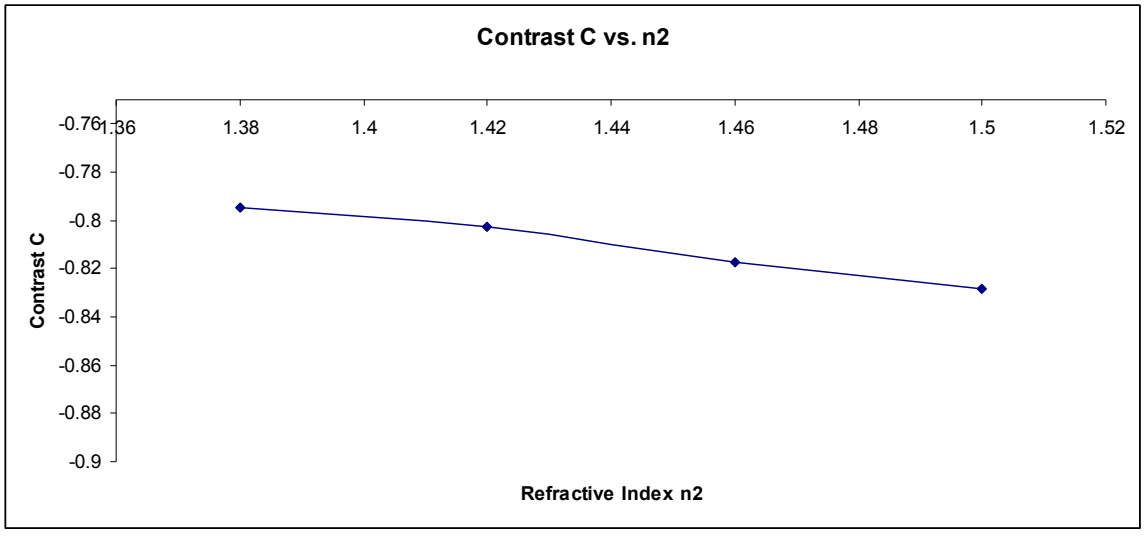

Figure 4.10. Results of contrast C plotted against n_2 for test set HT-MR1-2.

Simulation results of MR2: MR2 states as contrast value decreases when g_2 value increases. There are two test sets are used to examine MR2. Each test set contains four test cases.

For test set HT-MR2-1, the results of individual test case are shown in Table 4.3. Also, the results of contrast C are plotted against g_2 and shown in Figure 4.11. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	g_2	Contrast C	F-go	F-sub	F-io	B-go	B-sub	B-io
HT-MR2-1-T1	0.1	0.557547	14 100%	14 100%	13 100%	124 74.7%	34 60.7%	26 86.7%
HT-MR2-1-T2	0.3	0.513292	14 100%	14 100%	13 100%	124 74.7%	34 60.7%	26 86.7%
HT-MR2-1-T3	0.6	0.391112	14 100%	14 100%	13 100%	124 74.7%	34 60.7%	26 86.7%
HT-MR2-1-T4	0.9	0.0907334	14 100%	14 100%	13 100%	125 75.3%	34 60.7%	26 86.7%

Table 4.3. Simulation results of test set HT-MR2-1.

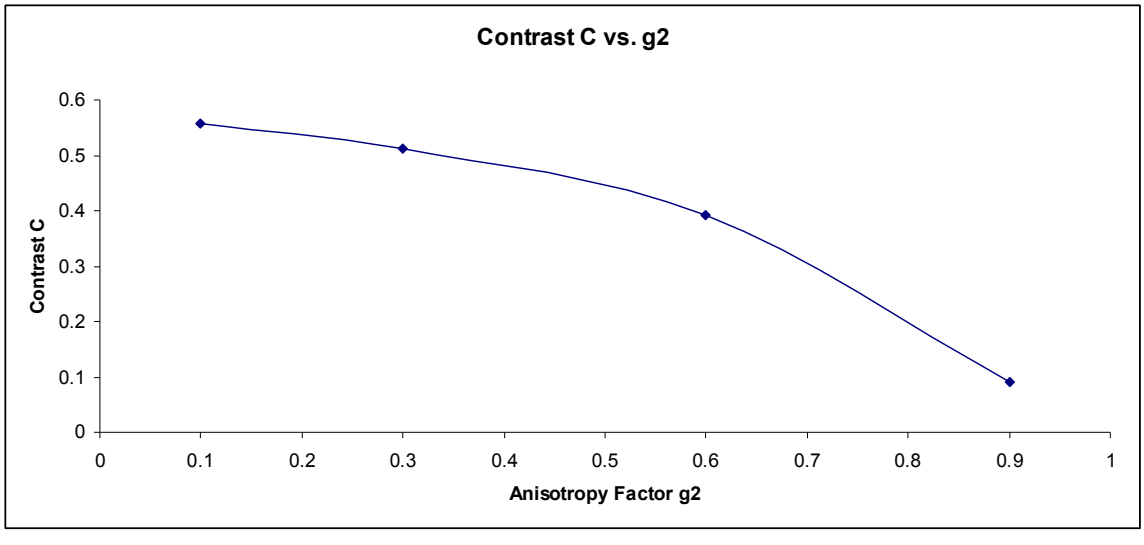

Figure 4.11. Results of contrast C plotted against g_2 for test set HT-MR2-1.

For test set HT-MR2-2, the results of individual test case are shown in Table 4.4. Also, the results of contrast C are plotted against g_2 and shown in Figure 4.12. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	g_2	Contrast C	F-go	F-sub	F-io	B-go	B-sub	B-io
HT-MR2-2-T1	0.2	0.085616	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%
HT-MR2-2-T2	0.4	-0.042526	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%
HT-MR2-2-T3	0.7	-0.362704	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%
HT-MR2-2-T4	1.0	-0.767007	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%

Table 4.4. Simulation results of test set HT-MR2-2.

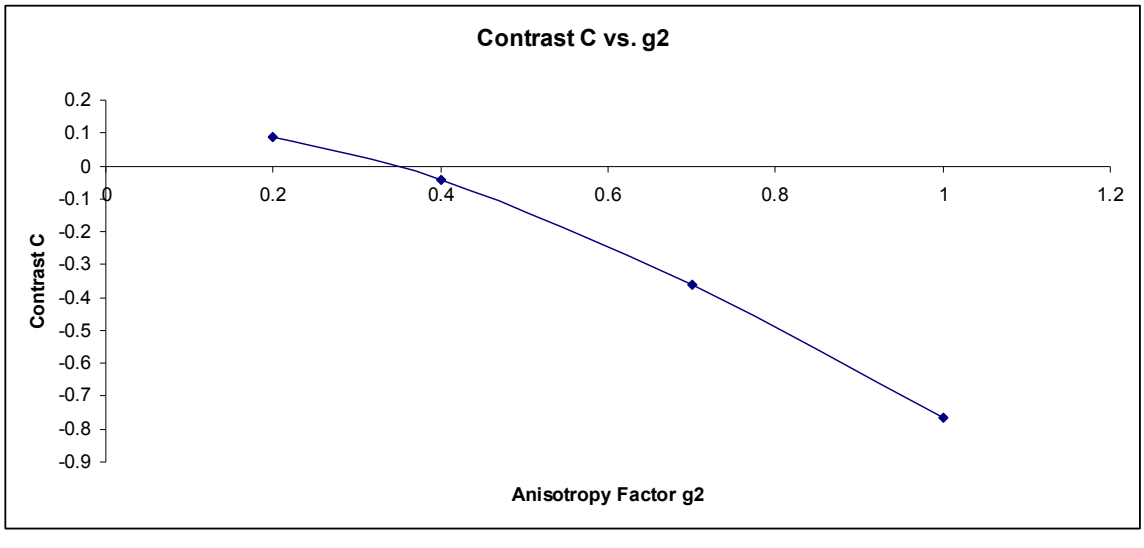

Figure 4.12. Results of contrast C plotted against g_2 for test set HT-MR2-2.

Simulation results of MR3: MR3 states as contrast value increases when albedo α_2 value increases. There are two test sets are used to examine MR3. Each test set contains four test cases.

For test set HT-MR3-1, the results of individual test case are shown in Table 4.5. Also, the results of contrast C are plotted against α_2 and shown in Figure 4.13. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	a_2	α_2	**Contrast C**	**F-go**	**F-sub**	**F-io**	**B-go**	**B-sub**	**B-io**
HT-MR3-1-T1	0.02	0.9615	-0.0820255	14 100%	14 100%	13 100%	124 74.7%	34 60.7%	26 86.7%
HT-MR3-1-T2	1.00	0.8333	-0.637966	14 100%	14 100%	13 100%	124 74.7%	34 60.7%	26 86.7%
HT-MR3-1-T3	2.50	0.6666	-0.904051	14 100%	14 100%	13 100%	124 74.7%	34 60.7%	26 86.7%
HT-MR3-1-T4	5.00	0.50	-0.961337	14 100%	14 100%	13 100%	125 75.3%	34 60.7%	26 86.7%

Table 4.5. Simulation results of test set HT-MR3-1.

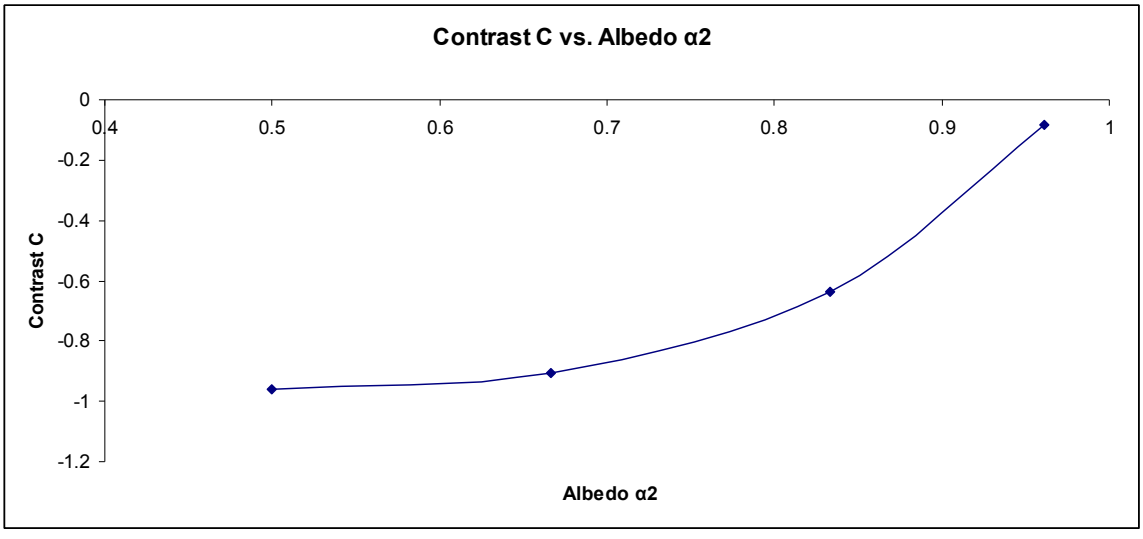

Figure 4.13. Results of contrast C plotted against α_2 for test set HT-MR3-1.

For test set HT-MR3-2, the results of individual test case ware shown in Table 4.6. Also, the results of contrast C are plotted against α_2 and shown in Figure 4.14. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	$_{a2}$	α_2	**Contrast C**	**F-go**	**F-sub**	**F-io**	**B-go**	**B-sub**	**B-io**
HT-MR3-2-T1	0.02	0.9933	0.185817	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%
HT-MR3-2-T2	0.50	0.8500	-0.247233	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%
HT-MR3-2-T3	1.20	0.7430	-0.639169	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%
HT-MR3-2-T4	2.00	0.60	-0.82205	14 100%	14 100%	13 100%	125 75.3%	37 66.1%	26 86.7%

Table 4.6. Simulation results of test set HT-MR3-2.

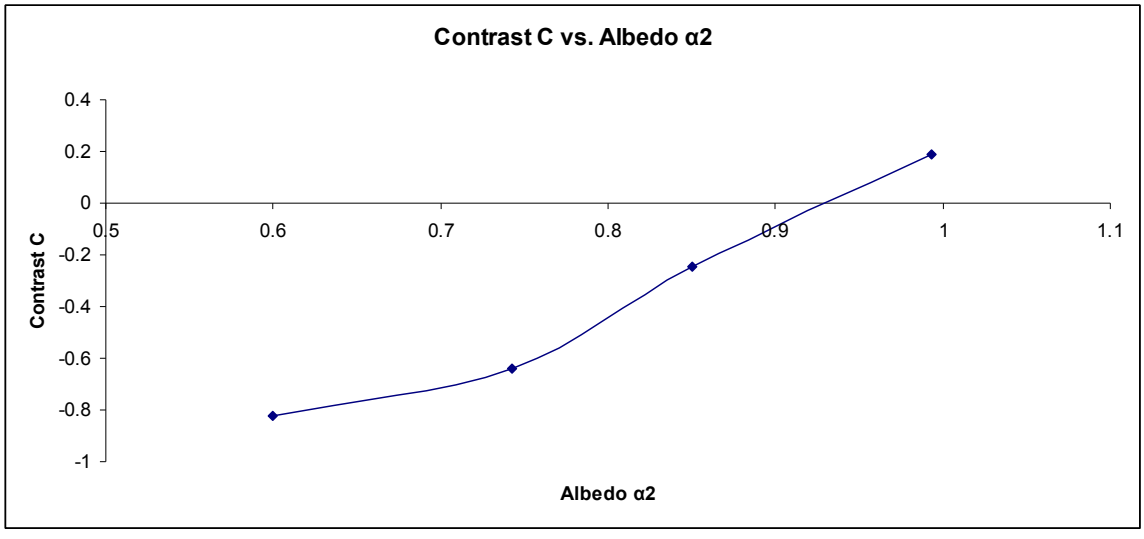

Figure 4.14. Results of contrast C plotted against α_2 for test set HT-MR3-2.

Simulation results of MR4: MR4 states as: for each pixel along the x axis P(x,0) on the image, will decrease if the numerical aperture (NA) decreases. There are two test sets are used to examine MR4. Each test set contains four test cases.

For test set HT-MR4-1, the structural coverage information of individual test case is shown in Table 4.7. The numeric results of the averaged reflectance R(x,0) are not shown in this thesis due to the large volume of data. The results of R(x,0) are plotted against *x* and shown in Figure 4.15. In order to get a clearer view and easy comparison with the results of ref [57], the diagram is plotted with a logarithmic y axis. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	Height (mm)	NA	F-go	F-sub	F-io	B-go	B-sub	B-io
HT-MR4-1-T1	0.00	1.000	14 100%	14 100%	13 100%	122 73.5%	34 60.7%	26 86.7%
HT-MR4-1-T2	3.35	0.966	14 100%	14 100%	13 100%	122 73.5%	34 60.7%	26 86.7%
HT-MR4-1-T3	12.50	0.707	14 100%	14 100%	13 100%	122 73.5%	34 60.7%	26 86.7%
HT-MR4-1-T4	46.65	0.259	14 100%	14 100%	13 100%	122 73.5%	34 60.7%	26 86.7%

Table 4.7. Coverage information of test set HT-MR4-1.

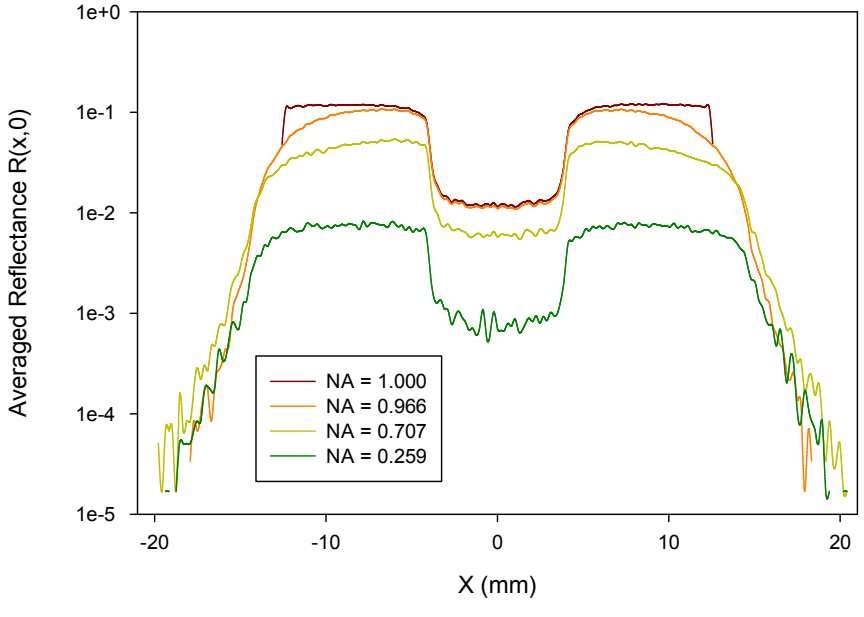

Figure 4.15. the averaged reflectance R(x,0) plotted against *x* for test set HT-MR4-1.

For test set HT-MR4-2, the structural coverage information of individual test case is shown in Table 4.8. The numeric results of the averaged reflectance R(x,0) are not shown in this

thesis due to the large volume of data. The results of R(x,0) are plotted against *x* and shown in Figure 4.16. In order to get a clearer view and keep the uniform format of the output for MR4, the diagram is plotted with a logarithmic y axis. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	**Height (mm)**	**NA**	**F-go**	**F-sub**	**F-io**	**B-go**	**B-sub**	**B-io**
HT-MR4-2-T1	0.00	0.996	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%
HT-MR4-2-T2	3.35	0.866	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%
HT-MR4-2-T3	12.50	0.500	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%
HT-MR4-2-T4	46.65	0.174	14 100%	14 100%	13 100%	124 74.7%	37 66.1%	26 86.7%

Table 4.8. Coverage information of test set HT-MR4-2.

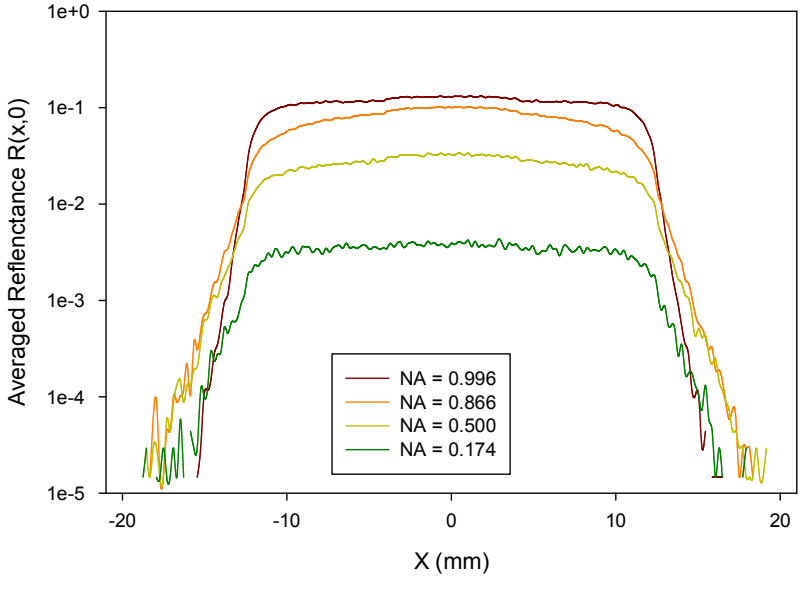

Figure 4.16. the averaged reflectance R(x,0) plotted against *x* for test set HT-MR4-2.

Simulation results of MR5: MR5 states as: For each pixel along the x axis P(x,0) on the image, the averaged reflectance R(x,0) will increase if the incident light angle θ_0 increases. There are four test sets are used to examine MR4. Two of them (HM-MR5-1 and HM-MR5-2) are for semi-finite homogeneous medium and the other two (HT-MR5-1 and HT-MR5-2) are for semi-finite heterogeneous phantom. Each test set contains four test cases.

For test set HM-MR5-1, the structural coverage information of individual test case is shown in Table 4.9. The numeric results of the averaged reflectance R(x,0) are not shown in this thesis due to the large volume of data. The results of R(x,0) are plotted against *x* and shown in Figure 4.17. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	**θ_0 (°)**	**F-go**	**F-sub**	**F-io**	**B-go**	**B-sub**	**B-io**
HM-MR5-1-T1	0	11 78.6%	14 100%	13 100%	56 33.7%	35 62.5%	26 86.7%
HM-MR5-1-T2	15	11 78.6%	14 100%	13 100%	55 33.1%	32 57.1%	26 86.7%
HM-MR5-1-T3	45	11 78.6%	14 100%	13 100%	55 33.1%	32 57.1%	26 86.7%
HM-MR5-1-T4	75	11 78.6%	14 100%	13 100%	55 33.1%	32 57.1%	26 86.7%

Table 4.9. Coverage information of test set HM-MR5-1.

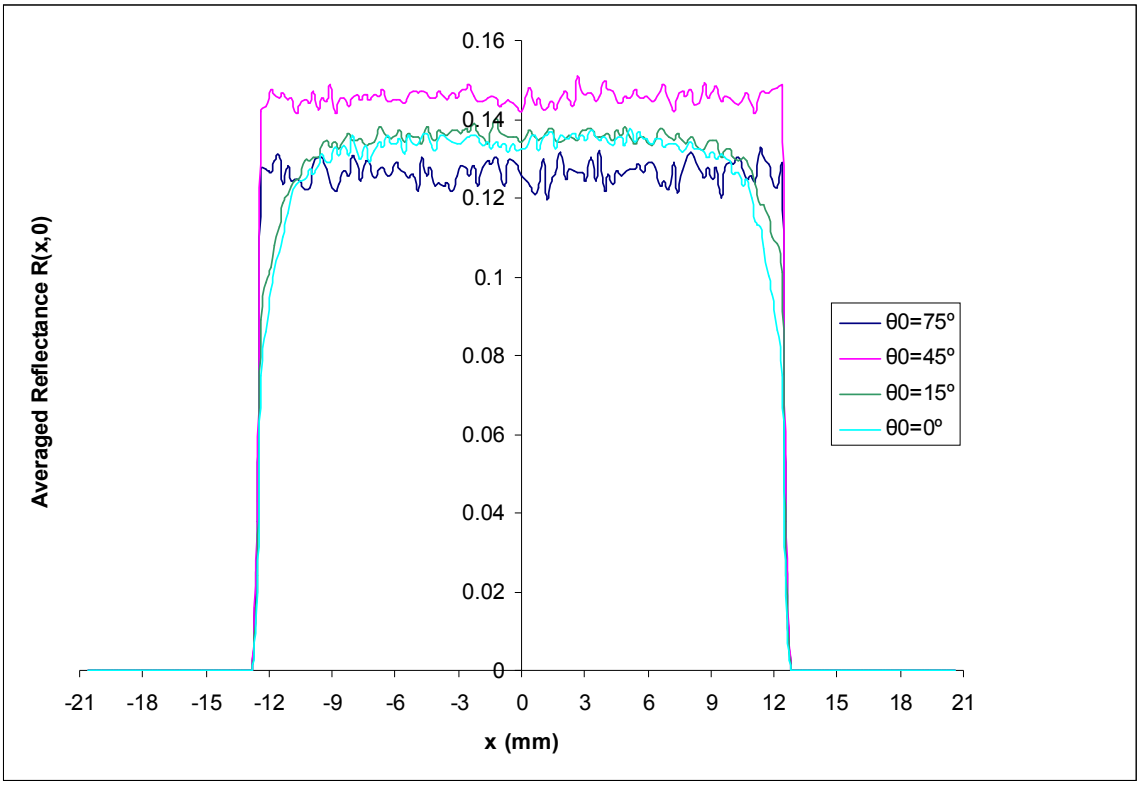

Figure 4.17. the averaged reflectance R(x,0) plotted against *x* for test set HM-MR5-1.

For test set HM-MR5-2, the structural coverage information of individual test case is shown in Table 4.10. The numeric results of the averaged reflectance R(x,0) are not shown in this thesis due to the large volume of data. The results of R(x,0) are plotted against *x* and shown in Figure 4.18. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	θ_0 (°)	F-go	F-sub	F-io	B-go	B-sub	B-io
HM-MR5-2-T1	5	11 78.6%	14 100%	13 100%	55 33.1%	32 57.1%	26 86.7%
HM-MR5-2-T2	30	11 78.6%	14 100%	13 100%	55 33.1%	32 57.1%	26 86.7%
HM-MR5-2-T3	60	11 78.6%	14 100%	13 100%	55 33.1%	32 57.1%	26 86.7%
HM-MR5-2-T4	85	11 78.6%	14 100%	13 100%	55 33.1%	32 57.1%	26 86.7%

Table 4.10. Coverage information of test set HM-MR5-2.

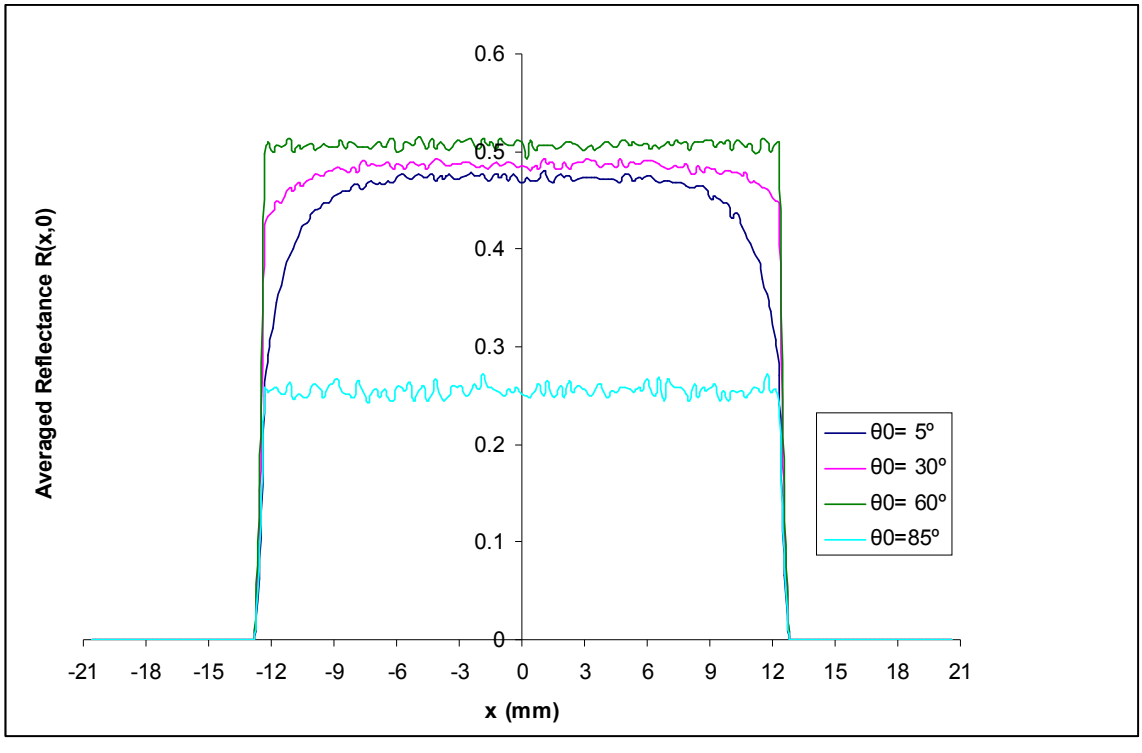

Figure 4.18. the averaged reflectance R(x,0) plotted against *x* for test set HM-MR5-2.

For test set HT-MR5-1, the structural coverage information of individual test case is shown in Table 4.11. The numeric results of the averaged reflectance R(x,0) are not shown in this thesis due to the large volume of data. The results of R(x,0) are plotted against *x* and shown in Figure 4.19. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	**θ_0 (°)**	**F-go**	**F-sub**	**F-io**	**B-go**	**B-sub**	**B-io**
HT-MR5-1-T1	0	14 100%	14 100%	13 100%	124 73.5%	37 66.1%	26 86.7%
HT-MR5-1-T2	15	14 100%	14 100%	13 100%	124 74.7%	34 60.7%	26 86.7%
HT-MR5-1-T3	45	14 100%	14 100%	13 100%	124 74.7%	34 60.7%	26 86.7%
HT-MR5-1-T4	75	14 100%	14 100%	13 100%	124 74.7%	34 60.7%	26 86.7%

Table 4.11. Coverage information of test set HT-MR5-1.

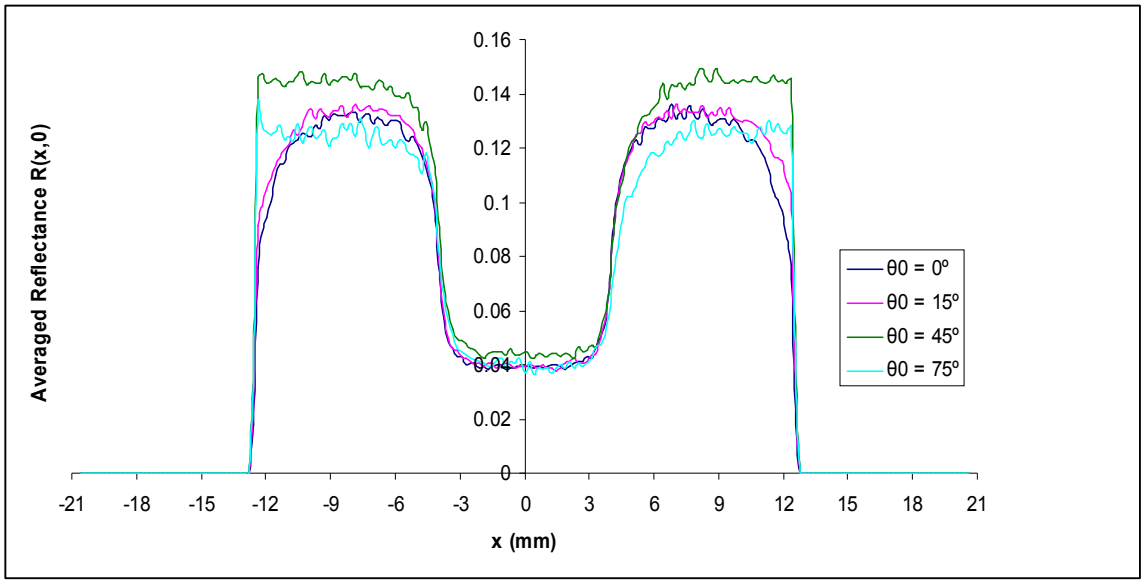

Figure 4.19. the averaged reflectance R(x,0) plotted against *x* for test set HT-MR5-1.

For test set HT-MR5-2, the structural coverage information of individual test case is shown in Table 4.12. The numeric results of the averaged reflectance R(x,0) are not shown in this thesis due to the large volume of data. The results of R(x,0) are plotted against *x* and shown in Figure 4.20. The diagram is visually inspected to decide if the metamorphic relation is satisfied or not.

Test Case	θ_0 (°)	F-go	F-sub	F-io	B-go	B-sub	B-io
HT-MR5-2-T1	5	14 100%	14 100%	13 100%	123 74.1%	36 64.3%	26 86.7%
HT-MR5-2-T2	30	14 100%	14 100%	13 100%	122 73.5%	34 60.7%	26 86.7%
HT-MR5-2-T2	60	14 100%	14 100%	13 100%	122 73.5%	34 60.7%	26 86.7%
HT-MR5-2-T2	85	14 100%	14 100%	13 100%	122 73.5%	34 60.7%	26 86.7%

Table 4.12. Coverage information of test set HT-MR5-2.

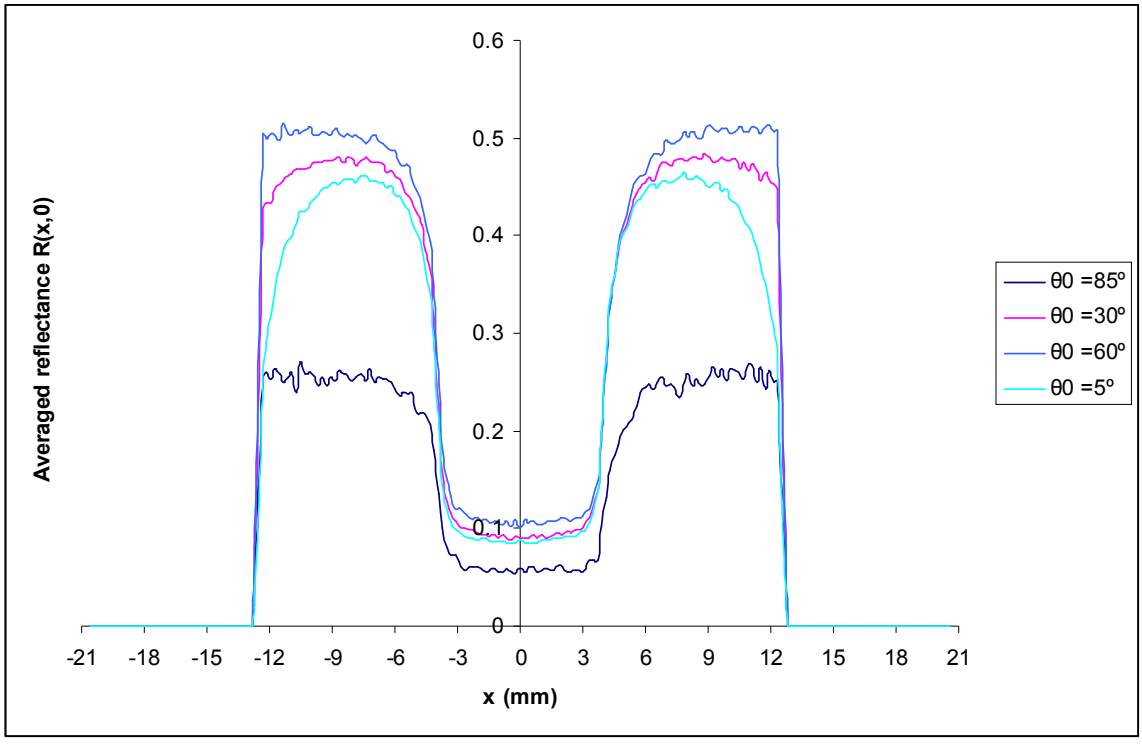

Figure 4.20. the averaged reflectance R(x,0) plotted against *x* for test set HT-MR5-2.

Relationship Between Image Contrast and Height of Collection Lens

For the experiment to examine the relationship between contrast C and height of the collection lens, the simulation results are shown in Table 4.13. The total reflected photons, the averaged photon number of the center circle, and the averaged photon number of the outer ring are also presented in the table. The results of contrast C are plotted against *height* and shown in Figure 4.21. The diagram is visually evaluated for possible pattern or conclusion.

Height (mm)	Total Photon Number	Ave. No_{center}	Ave. No_{ring}	Contrast C
0	535381	10.6449	50.8768	-0.653947
1	506495	10.6077	49.2203	-0.645393
2	467428	10.4621	45.3566	-0.62514
3	429382	10.1947	41.2406	-0.603593
4	393975	9.79941	37.38	-0.58459
5	361832	9.29421	33.924	-0.569894
6	332464	8.67311	30.8182	-0.560759
7	305989	8.08618	28.0929	-0.552991
8	281974	7.48886	25.6397	-0.547891
9	260063	6.89747	23.4425	-0.54532
12	206223	5.39079	18.2041	-0.543055
15	166004	4.24963	14.3858	-0.543919
18	135128	3.30609	11.565	-0.555367
21	111451	2.66568	9.43768	-0.559514
24	92929	2.15453	7.79502	-0.566909
30	66419	1.47103	5.48663	-0.577149
40	40776	0.884101	3.30557	-0.577962
50	27135	0.580981	2.18979	-0.580635
70	14223	0.307578	1.14336	-0.576029
100	7035	0.15156	0.569218	-0.579454
130	4121	0.0846954	0.337226	-0.598525
150	3070	0.0624071	0.252532	-0.603688
170	2366	0.0460624	0.195414	-0.618493
200	1746	0.0326895	0.143078	-0.628038

Table 4.13. Results of contrast *C* vs. collection lens height change

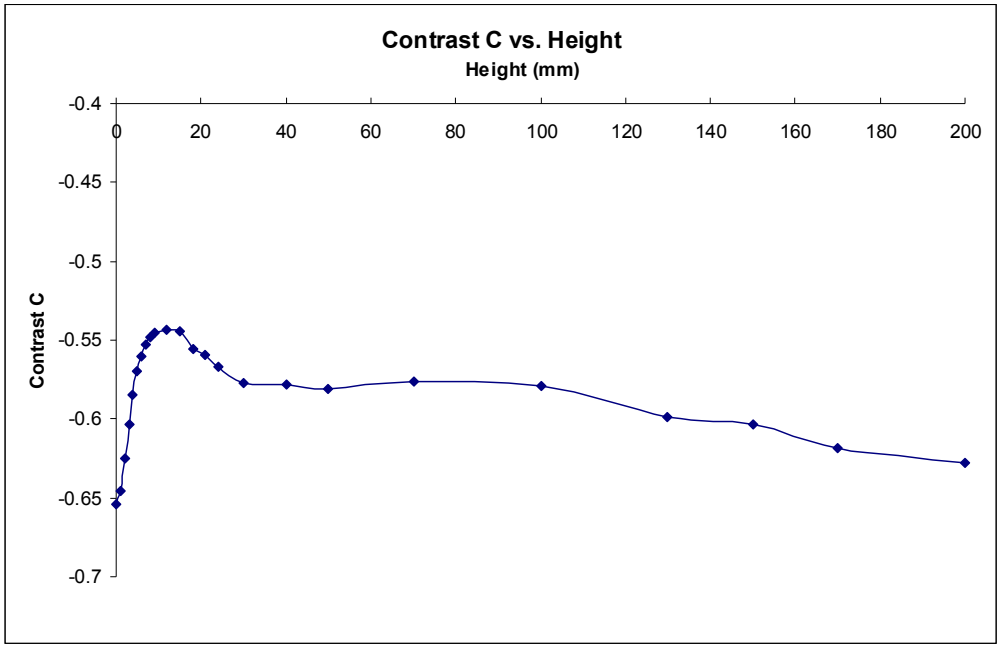

Figure 4.21. Results of contrast C against height of collection lens.

Discussion

Validation of the Monte Carlo Code for Homogenous Media

According to the testing results and diagrams of the two test sets and eight test cases, both simulated bidirectional reflection and transmission functions have very good agreements with the calculated results from van de Hulst's table [46]. These results show that our Monte Carlo program has passed the validation with van de Hulst's table method. We should have more confidence to at least the part of the program code which handles homogenous media. Meanwhile, these results also provide a good start point to validate the whole Monte Carlo simulation program.

Self-Checked Metamorphic Testing

For metamorphic relation MR1 to MR4, simple visual inspections can show that all the inputs and outputs fit the corresponding metamorphic relation very well for the whole eight test sets and 32 test cases. No faults were revealed.

For metamorphic relation MR5, we found there was no simple linear change between averaged reflectance *R(x,0)* and *x*, which is different to MR5. MR5 was reported as a property in reference [57]. Therefore, further investigation is needed.

In order to locate the problem, we experimented several different test sets but we could not find any test set that was satisfied MR5. However, MR5 could be satisfied for some special test cases, such as $\theta_0 = 45^0$, 75^0, and 85^0 when other configuration parameters remain the same as test set HT-MR5-1. The result further confirms that it is important to evaluate the quality of the test cases and their corresponding metamorphic relations in order to minimize the false positive results. From physical knowledge and the configuration of the simulation, we know that if MR5 holds for heterogeneous media, then it should hold for homogenous media as well. Since the Monte Carlo modeling program handling the homogenous media is a subset of the program for handling the heterogeneous media, we tested MR5 for homogeneous media with test sets HM-MR5-1 and HM-MR5-2 to look at whether MR5 is satisfied in the special cases.

Comparing the test coverage results in Table 4.9 and Table 4.11, it is not difficult to found that the program for the homogenous media is much simpler than the program for the heterogeneous media. Since Figure 4.17 and 4.19 have the same patterns, we can narrow the problematic code to the program for homogenous media. Several experienced programmers independently inspected the program for homogenous media (less than 500 lines of code), we could not find the error. From physics theory, it is difficult to tell whether MR5 should hold or not. We could not find the original version of the code that was experimented for the results in

reference [57], we cannot conclude the correctness of MR5. One way to check the validity of MR5 is to build a real experimental environment to test MR5. However, building the experimental environment that is identical to our Monte Carlo simulation configuration is very complex.

The Monte Carlo modeling program discussed in this thesis is a "non-testable" program, which is hard to find test oracles. Therefore, metamorphic testing is used to test the program. We identified 5 metamorphic relations for the testing. Some of these metamorphic relations (like MR4) are easy to be validated by physics theories or experiments, but some of them (such as MR5) are difficult to be validated. We selected these metamorphic relations based on the experimental results discussed in reference [57]. Several test sets were created for each metamorphic relation, and each test set includes at least 4 test cases. For each test case (except those for MR5 for homogeneous media), all functions are easily covered. However, no any test case even test set covers 100% branches. The source code of the Monte Carlo simulation program was underwent a static inspection with the help of coverage information from self-checked metamorphic testing. Most of the code, which has not been covered by the test cases, is used to handle extreme system configurations and the errors happening during the simulation. In order to cover 100% branches in the program, special test cases that will cause simulation error are needed.

From the test results, we believe the test coverage data can be used to evaluate the quality of test cases and metamorphic relations and to act as a guideline to select metamorphic relations and create test cases. In our research work, all test cases in Table 4.9 and Table 4.11 are used to examine MR5. But obviously, the test cases in Table 4.11 have more branch coverage than those in Table 4.9. We could say test cases in Table 4.11 have better quality. From branch coverage

point of view, all the test cases in Table 4.9 are redundant. They probably won't reveal any new faults since they didn't cover any new branches of the program (certainly, they are useful to test MR5). If more complex test coverage criteria such as state transition coverage or path coverage are considered, then the coverage data can be even more useful.

Due to the large sampling volume of the total photon number, it is not a surprise that the coverage data for all the test cases in each test set are very similar. This information may indicate simply increasing the number of test cases for each metamorphic relation we investigated probably have little help to find more faults because no new branches are covered.

The test results for MR5 showed that some bugs must exist in the Monte Carlo modeling program, which also demonstrated the effectiveness of the metamorphic testing. Through analyzing the metamorphic testing results and test coverage information, we found that MR1 to MR4 might not be needed. Not only because they didn't reveal any faults, but also because they have very similar function and branch coverage as test cases for MR5. However, if we consider state transition coverage or all path coverage, then we may found MR1 to MR5 or even more metamorphic relations are needed since state transitions and all paths are much more complicated and each test case probably will undergo different state or path.

The testing results successfully shows that the testing coverage information could guide us to select metamorphic relations and create test cases. Through testing MR5 for both homogenous and heterogeneous media, we could narrow the range of potential problems in the program via analyzing the test coverage information after the metamorphic testing results revealed possible faults.

Overall, we demonstrated in this work with a case study that the combination of metamorphic testing and structural testing has its unique benefits. Comparing with original

metamorphic testing method, even requiring slightly additional work, our approach could provide more helpful information to evaluate and select test cases or metamorphic relations and control the whole testing procedure.

Relationship Between Image Contrast and Height of Collection Lens

As the last part of our experiments, Simulation results show that the height change of the collection lens does not have to much affect on the image contrast according to the simulation results and the diagram. The contrast range changes form -0.653 to -0.543. Especially when the height is over 40 mm, the change of image contrast is really slow. The experiment stopped at the height of 200 mm because the total reflected photons that the collection lens could collect is really small compared to the total reflected photons, which will cause big variance during the simulations. Although, this conclusion has no conflict with any physics theory as far as we know, the results remain questionable since the correctness of the Monte Carlo code needs further verification. In order to get more reliable conclusion, this experiments will be carried on again for more data points and configuration files of the system after we fix the bugs of our current version of program.

Summary and Future Work

Metamorphic testing as an effective testing technique has been widely used for testing systems that do not have test oracles. However, checking only the metamorphic relations among outputs is not good enough to ensure the testing quality. A guideline for creating metamorphic relations and generating test cases is important to improve the quality of metamorphic testing. In this thesis, the metamorphic testing is extended with checking adequacy of test coverage criteria: function coverage and branch coverage. The adequacy of the test coverage criteria is chosen as

the metamorphic testing requirements and it is also served as a guideline for selecting metamorphic relations, generating test cases, and finding errors in the program. The effectiveness of our approach has been investigated through testing a Monte Carlo modeling program. In addition to checking 5 metamorphic relations, the adequacy of function coverage criterion and branch coverage criterion are automatically evaluated during the metamorphic testing.

The case study demonstrated that our approach could be conducted with a more controllable and efficient manner than the original metamorphic testing method. But it also raised several questions regarding the quality of the metamorphic testing. In the future, we will compare the effectiveness of our approach to other metamorphic testing approaches and investigate approaches for generating test cases for metamorphic testing through analyzing program structures.

REFERENCES

[1] G. J. Myers., "The Art of Software Testing", Wiley, 1979.

[2] A. P. Mathur "Foundations of Software Testing", Pearson Education, 2008.

[3] M. Pezzè, and M. Young, “Software Testing and Analysis: Process, Principles, and Techniques”, Wiley, 2007.

[4] T. Y. Chen, S. C. Cheung, and S. Yiu, “Metamorphic testing: a new approach for generating next test cases”, Tech. Rep. HKUST-CS98-01, Dept. of Computer Science, Hong Kong Univ. of Science and Technology, 1998.

[5] A. Gotleib, and B. Botella, “Automated metamorphic testing”, In Proc. of 27th Annual International Computer Software and Applications Conference, pp. 34-40, 2003.

[6] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, “Fault-based testing without the need fo oracles”. Information and Software Technology, 44(15), pp. 923 – 931, 2002.

[7] C. Murphy, K. Shen, and G. Kaiser, “Using JML runtime assertion checking to automate metamorphic testing in applications without test oracles”, In Proc. of the 2nd IEEE International Conference on Software Testing, Verication and Validation (ICST), 2009.

[8] X. Xie, J. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, “Application of metamorphic testing to supervised classifiers”, In Proc. of the 9th International Conference on Quality Software (QSIC), 2009.

[9] T. Y. Chen, D. H. Huang, T. H. Tse, and Z. Q. Zhou, “Case studies on the selection of useful relations in metamorphic testing”, In Proc. of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering, pp. 569-583, 2004.

[10] T. Y. Chen, J. W. K. Ho, H. Liu, and X. Xie. “An innovattive approach for testing bioinformatics programs using metamorphic testing”, BMC Bioinformatics, pp. 10-24, 2009.

[11] C. Murphy, K. Shen, and G. Kaiser. “Automatic system testing of program without test oracles”. In Proc. of the 2009 ACM Inter. Symposium of Software Testing and Analysis (ISSTA), 2009.

[12] W. K. Chan, S. C. Cheung, and K. R. Leung, “A metamorphic testing approach for online testing of service-oriented software applications”, International Journal of Web Services Research 4, 1, pp. 60 – 80, 2007.

[13] C. Murphy, G. Kaiser, L. Hu, and L. Wu. “Properties of machine learning applications for use in metamorphic testing”. In Proc. of the 20th Int. conference on software engineering and knowledge engineering (SEKE), pp. 867–872, 2008.

[14] A. Gotleib, and B. Botella, “Automated metamorphic testing”, In Proc. of 27th Annual International Computer Software and Applications Conference, pp. 34-40, 2003.

[15] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, “Semi-proving: an integrated method based on global symbolic evaluation and metamorphic testing”. In Proc. of the Inter. Symposium on Software Testing and Analysis (ISSTA), pp. 191-195, 2002.

[16] H. T. MacGillivray, R. J. Dodd, "Monte-Carlo simulations of galaxy systems", *Astrophysics and Space Science*, Volume 86, Number 2 / September, 1982.

[17] S. A. Baeurle, "Multiscale modeling of polymer materials using field-theoretic methodologies: a survey about recent developments", Journal of Mathematical Chemistry 46 (2): 363–426 2009.

[18] R. H. Swendsen, and J. Wang, "Replica Monte Carlo simulation of spin-glasses", Physical Review Letters 57 (21) 1986.

[19] G. H. Bird, "Monte-Carlo simulation in an engineering context", Progress in Astronautics and Aeronautics, pp. 239-255, 1981.

[20] M. Milik, and J. Skolnick, "Insertion of peptide chains into lipid membranes: an off-lattice Monte Carlo dynamics model", Proteins 15 (1): pp. 10-25, 1993.

[21] C. Forastero, L. Zamora, D. Guirado, and A. Lallena, "A Monte Carlo tool to simulate breast cancer screening programmes", Physics in Medicine and Biology 55 (17)pp. 5213, 2010.

[22] S. S. Sawilowsky, and G. C. Fahoome, "Statistics via Monte Carlo Simulation with Fortran", Rochester Hills, 2003.

[23] S. Sawilowsky, "You think you've got trivials?", Journal of Modern Applied Statistical Methods, 2(1), 218-225 2003.

[24] D. Vose, “Risk Analysis, A Quantitative Guide,” Second Edition, pp. 13, John Wiley & Sons, 2000.

[25] P. Glasserman, "Monte Carlo methods in financial Engineering", Springer, 2004.

[26] S. Chib and E. Greenberg, "Markov chain Monte Carlo simulation methods in econometrics", Econometric Theory vol. 12, pp. 409-431, 1996.

[27] P. P. Boyle, "Options: a Monte Carlo approach", Journal of Financial Economics Vol 4(3), pp. 323-328, 1977.

[28] S. R. Arridge, M. Schweiger, M. Hiraka, and D. T. Delpy, "A finite element approach for modelling photon transport in tussue", Medical Physicas, 20 (2) pp. 299-309, 1993.

[29] M. S. Patterson, B. Chance, and B. C. Wilson, "Time resolved reflectance and transmittance for the non-invasive measurement of tissue optical properties", Applied Optics, Vol. 28 (12) pp. 2331-2336 1989.

[30] A. Kienle, L. Lilge, M. S. Patterson, R. Hibst, R. Steiner, and B. C. Wilson, "Saptially resolved absolute diffuse reflectance meansurements for noninvasive determination of the optical scattering and absorption coefficients of biolobical tissue" Applied Optics, Vol. 35 (13) pp. 2304-2314, 1996.

[31] R. R. Anderson and J. A. Parrish, "The optics of human skin," Journal of Investigative Dermatology, Vol.77, pp. 13–19, 1981.

[32] G. Zonios, J. Bykowski, and N. Kollias, "Skin melanin, hemoglobin, andlight scattering properties can be quantitatively assessed in vivo usingdiffuse reflectance spectroscopy," Journal of Investigative Dermatology, Vol.117, pp. 1452–1457, 2001.

[33] S. L. Jacques, J. C. Ramella-Roman, and K. Lee, "Imaging skin pathologywith polarized light," Journal of Biomedical Optics, Vol 7, pp. 329–340, 2002.

[34] S.A. Prahl, M. keijzer, S. L. Jacques, and A. J. Welch "A Monte Carlo model of light propagation in tissue", Proceedings of SPIE IS Vol. 5 pp. 102 – 111, 1989.

[35] M. Keijzer, S. L. Jacques, S. A. Prahl, and A. J. Welch, "Light distributions in artery tissue: Monte Carlo simulations for finite-diameter laser beams." Lasers in Surgery and Medicine, Vol. 9 pp. 148 – 154, 1989.

[36] C. Chen, J. Q. Lu, K. Li, S. Zhao, R. S. Brooks, and X. H. Hu, "Numerical study of reflectance imaging using a parallel Monte Carlo method." Medical Physics 34, pp. 2939-2947, 2007.

[37] A. Ishimaru, "Wave propagation and scattering in random media", Vol. 1, Academic Press, New York, 1978.

[38] D. Weston, "Electromagnetic Compatibility (Electrical and Computer Engineering)", 2nd edition, CRC Press, Boca Raton, 2001.

[39]S. Chandrasekhar, "Radiative Tranfer", Dover Publications Inc., 1960.

[40] R. A. J. Groenhuis, J. J. Ten Bosch, and H. A. Ferwerda, "Scattering and absorption of turbid materials determined from reflection measurements. 1: Theory," Applied Optics 22, pp. 2456-2462, 1983.

[41] T. J. Farrell, M. S. Patterson, and B. C. Wilson, "A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the non-invasive determination of tissue optical properties in vivo," Medical Physics 19, pp. 879-888, 1992.

[42] S. R. Arridge, "Optical tomography in medical imaging", Inverse Problems, 15:R41–93, 1999.

[43] J.M. Kaltenbach, and M. Kaschke "M. Frequency- and time-domain modeling of light transport in random media." Proc SPIE ISll:65–86, 1993.

[44] J. D. J. Ingle and S. R. Crouch, "Spectrochemical analysis", Prentice Hall, New Jersey, 1988.

[45] L. Henyey and J. Greenstein, ``Diffuse radiation in the galaxy," Astrophysics. Journal, vol. 93, pp. 70-83, 1941.

[46] H. C. van de Hulst, "Multiple light scattering: tables, formulas, and applications", Vol. 1 & 2. Academic Press, New York, 1980.

[47] B. C. Wilson and G. Adam, “AMonte Carlo model for the absorption and flux distributions of light in tissue,” Medical Physics, 10, pp. 824–830, 1983.

[48] S. A. Prahl, M. Keijzer, S. L. Jacques, and A. J. Welch, “A Monte Carlo Model of Light Propagation in Tissue,” SPIE Institute Series 5, pp. 102–111, 1989.

[49] L. Wang, S. L. Jacques, and L. Zheng, “MCML-Monte Carlo modeling of light transport in multi-layered tissues,” Computer Methods and Programs Biomedicine, 47, pp. 131–146, 1995.

[50] Z. Song, K. Dong, X. H. Hu, and J. Q. Lu, “Monte Carlo simulation of converging laser beams propagating in biological materials,” Applied Optics, 38, pp. 2944–2949, 1999.

[51] J. Q. Lu, X. H. Hu, and K. Dong, “Modeling of the rough-interface effect on a converging light beam propagating in a skin tissue phantom,” Applied Optics, 39, pp. 5890–5897, 2000.

[52] P. K. Yalavarthy, K. Karlekar, H. S. Patel, R. M. Vasu, M. Pramanik, P. C. Mathias, B. Jain, and P. K. Gupta, “Experimental investigation of perturbation Monte-Carlo based derivative estimation for imaging lowscattering tissue,” Optical Express, 13, pp. 985–997, 2005.

[53] K. Mosegaard, and M. Sambridge, “Monte Carlo analysis of inverse problems,” Inverse Problem. 18, R29-R54, 2002.

[54] L. Wang, and S. L. Jacques, "Monte Carlo modeling of light transport in multi-layered tissues in standard C", 1992-1998. Download as 177-page manual in pdf format.

[55] M. Pezzè, and M. Young, “Software Testing and Analysis: Process, Principles, and Techniques”, Wiley, 2007.

[56] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, "Numerical Recipes in Fortran 77—The Art of Scientific Computing", 2nd ed. Cambridge University Press, New York, 1992.

[57] C. Chen, J.Q. Lu, K. Li, S. Zhao, R.S. Brock, X.H. Hu, “Numerical study of reflectance imaging using a parallel Monte Carlo method”, Medical Physics, vol. 34, pp. 2939-2948, 2007.

[58] H. Zhu, P. A. Hall, and J. H. May. “Software unit test coverage and adequacy”. ACM Comput. Surv. vol. 29, 4, pp. 366-427, 1997.

[59] E. Weyuker, "On testing non-testable programs". The Computer Journal, 25(4), pp 465-470.

[60] I. K. El-Far and J. A. Whittaker "Model-based software testing," Encyclopedia on Software Engineering, Wiley, 2001.

[61] J. B. Goodenough and S. L. Gerhart "Toward a theory of test data selection." IEEE Transactions on Software Engineering, SE-3 (June), 1975.

[62] J. B. Goodenough and S. L. Gerhart "Toward a theory of testing: data selection criteria." Current trends in programming methodology, vol. 2, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[63] J. R. Horgan and S. London, "ATAC: a data flow coverage testing tool for C." In Proceedings of Symposium on Assessment of Quality Software Development Tools, Los Alamitos, pp. 2-10. 1992.

[64] V. Massol and T. Husted, "JUnit on Action." Greenwich, CT: Manning Publications Co. 2003.

[65] R. Jameson and G. Gaskell "About code coverage." Clover Documentation Home, 2010. < http://confluence.atlassian.com/display/CLOVER/Clover+Documentation+Home>

APPENDIX: RANDOM SEEDS FOR RANDOM NUMBER GENERATOR

-444703364 -19639514 -568072461 -225949868 -267947251 -692669395

-340047949 -815934769 -548782126 -999447305

-615432348 -681277161 -370413557 -579806873 -439924310 -84079061

-314217311 -460769831 -931583353 -961636407

-791937037 -379481018 -702739913 -760365010 -933380108 -454355150

-365078387 -457354380 -335197430 -58862165

-921812971 -831796018 -546571152 -529823117 -683332324 -441828297

-393239551 -450688882 -655531055 -360311171

-738207246 -502812884 -444880205 -640526499 -212559864 -353250455

-591525204 -412219061 -391904207 -548512814

-176266144 -709471393 -694567240 -209069404 -839238240 -153606363

-119746617 -901609825 -627314788 -261769570

-405706213 -428892365 -621310131 -379818370 -628784600 -675644650

-38128797 -5583939 -699080138 -597344848

-935469699 -304617367 -794821080 -783328650 -133772748 -699213328

-458597955 -297405679 -397183954 -49277997

-916904440 -189653748 -956843448 -680845751 -207132730 -727509129

-869867350 -49162489 -413628895 -571057494

-410270207 -193431156 -522590349 -461095127 -607198945 -478384381

-934236519 -693180453 -655212946 -700857233

-893649531 -682223224 -880142207 -567828712 -629887849 -554841986

-264449166 -650106410 -837585099 -962288259

-57891305 -302764401 -172956141 -794210651 -370476826 -121047113

-160300339 -982987782 -371608032 -750518232

-352868132 -541673854 -979746897 -59182593 -575147779 -450753941

-872855257 -552673243 -425253158 -739993045

-813166497 -150872976 -271447259 -602869086 -451424827 -715882948

-237880308 -400073516 -594663371 -431873386

-9861301 -697898482 -252329347 -50268804 -43895325 -892841608 -645831250

-198788520 -565738572 -634265956

-138890882 -378373001 -875741900 -415374860 -27185123 -273102470

-966887421 -625201023 -716542397 -803026340

-202765219 -860011605 -737305988 -304998677 -312685048 -254769296

-664931212 -733362802 -511311448 -83881007

-198721743 -853655131 -136518742 -874367172 -12862575 -865603478

-870381026 -375885482 -776401212 -945462788

-603792479 -593562913 -11756687 -15009499 -383967288 -232350371

-9927305 -9876463 -489345476 -915942460

-272187925 -496552450 -893897966 -767950390 -683115968 -804871744

-137009892 -419857806 -185904451 -601987424

-198814268 -899769175 -199138067 -970844939 -92842462 -908397543

-818755827 -753669626 -700635405 -253560579

-15273927 -821629161 -298723012 -990082593 -35338324 -231894318

-430166049 -793871775 -982708802 -873450806

-746785677 -644910384 -661442576 -788861692 -612395481 -239312564

-890321724 -919957207 -806637749 -513400710

-445096432 -817974341 -284408590 -438658534 -608540361 -49754484

-734908212 -844721502 -703567655 -732650649

-931814578 -660227556 -469224285 -498311303 -15759818 -78384075

-687323592 -367752882 -484963719 -422226586

-465994342 -341970618 -64781123 -213963332 -16354934 -640815410

-346111968 -620801332 -114612821 -961369999

-418649468 -289725896 -988334938 -643492288 -190074589 -190886570

-166034739 -731277264 -664855567 -72059239

-846221418 -341193569 -582791682 -320035577 -586918472 -843869499

-155612576 -193893180 -365373892 -553407968

-525152496 -534079018 -423496257 -960098600 -57581090 -173900248

-191116311 -904812334 -140044457 -291983921

-202647358 -727113217 -515511752 -726631767 -367568039 -170792814

-422451526 -569205750 -566772800 -857963513

-672137468 -309290160 -333951480 -411953208 -631451165 -994295491

-855975707 -631789929 -823008314 -335755139

-838118445 -838496045 -432906596 -744565783 -717634421 -439790857

-490249987 -234412955 -673948632 -680203850