Republic of the Philippines
UNIVERSITY OF SOUTHEASTERN PHILIPPINES
Institute of Computing
Iñigo Street, Obrero, 8000 Davao City

- Exploring Novice Student’s C++ Programming Compilation Log Using Data Mining

- Papaya Defect Classification Using Artificial Neural Network

- Solid Waste Recognition Using Maximum Cross Correlation

- Tree Leaf Identification Through Digital Image Processing

PREFACE

The Institute of Computing (IC) Journal is a Research Journal encompassing the branches of Information Technology such as Programming, Web-based and Mobile Programming, Networking, Image Processing and the like. This journal, published twice a year, is a collection of researches conducted by the students of University of Southeastern Philippines – Institute of Computing (USeP-IC)

The main objective of this journal is to publish the researches of the students of the University of Southeastern Philippines – Institute of Computing. The journal aims to disseminate information to a larger community of experts whose expertise are in line with the disciplines and subject areas covered herein. All the papers included were reviewed by the subject experts before publication. The contents of this Journal can be accessed free of charge by anyone who finds interest on these researches.

This Journal will not be possible without the supervision of Editor- in-chief, Managing Editor and the Consultant/Dean of the Institute of Computing.

Table of Contents

Exploring Novice Student’s C++ Programming Compilation Log Using Data Mining

Randy Gamboa, Marlon Dizon, Rose Ann Joy Deluao, Mary Jude Mispeñas, Honey Lyn Toyorada

ABSTRACT

In this study, the researchers applied data mining in extracting information from students’ DevC++ Compilation Log. Dev C++ IDE was modified to capture students’ compilation logs, written in text files and forwarded to the server. Text files containing log information were sent to Dev C++ Log Miner; these were analyzed to generate visualization and interpretation of results. Visualizations and reports generated reliable information such as error count per compile, committed error count, compile time interval and character count per compile to determine novice students’ programming performance and challenges.

The data mining system gathered a total of one thousand five hundred forty three (1,543) errors committed by the students while solving the programming exercise. The most committed error ‘*’ was not declared in this scope with 25.86% means that the compilers could not identify the ‘*’in the program. Around fifteen percent of novice student performing the exercise takes longer time (approx. 250 – 300 second) to compile; five percent of them never attempted to compile nor perform the exercise; and the remaining 80 percent on the average are compiling within 50 to 150 seconds.

1. INTRODUCTION

Majority of novice students taking up computer programming subjects often find problems in coding, designing and in debugging their program using programming concepts. Also, compilation error messages generated by Integrated Development Environment (IDE) that are difficult to understand become another burden in their learning. This difficulty has made them frustrated with what they are solving and somehow affected their way of providing solution to a problem. The empirical study of Ebrahimi, Kopec and Schweikert [1] revealed that programming performance of students are mostly affected by language construct misconception, plan composition errors, programming environments and the inability to get assistance.

Data mining is a useful tool to discover new knowledge from a particular set of data which is utilized in this study. Data mining enables educational institutions to gain a more comprehensive, integrative and reflexive view of the impact of the technologies by obtaining a better understanding around issues of information use and access, ultimately leading to improved knowledge sharing and effective decision making. With the obtained information from data mining of common programming errors, this can help to know how teaching of programming can be done in an effective way or can help in designing a strategy that would help the student to better understand programming. Preventing students from committing common programming errors or letting them understand the meaning of these errors would help them think of a solution for a particular programming problem.

According to Ranjan and Khalil [2], examining programming errors will be useful to help instructors in managing their classes, understand their students' learning, and reflect on their teaching to support learner reflection and provide proactive feedback to learners.Several studies have been conducted discussing the programming behavior of programming students. One of these is in University of Kent in United Kingdom. The study explores first year students' programming behavior using BlueJ, an educational integrated development environment for programming in Java. The goal of the study according to Jadud[3] is to help the learning institution and the development environment for Java to match well with the behavior of students towards learning programming in Java. Examining compilation behavior enabled them to identify most common errors encountered by these students and how they program in supervised programming sessions.

Here in the Philippines, professors from Ateneo de Manila University conducted a study on what non-literal Java errors are mostly committed by the introductory class. As explained by Dy, et. al [4], non-literal errors are those compiler error messages that do not give the exact meaning of the error. One of the reasons why the study was conducted is the debugging difficulties the students encountered when the programming environment or compiler shows error mesproblem, rather it adds another problem to the program. Another purpose of the study is to gain idea on what part of the lesson should be focused in accordance to the most frequently committed error.

The concept of applying data mining in educational setting is relatively new. In Mindanao where insufficient fund for education and research is allotted, it is not surprising to know that the region is far behind when it comes to this matter. In Davao City, data mining is not popularly known, both in business and education. Only manual and written examinations are conducted among elementary and secondary institutions to further evaluate the performance of students. The results of these examinations serve as the basis of determining what particular area most of their students are good at or having difficulties with.

In the University of Southeastern Philippines – Institute of Computing, it was observed that a high number of students taking up introductory programming subjects like computer programming 1 were failing and were not promoted. Unfortunately, there is no study addressing this problem, hence the researchers were motivated to conduct this study. Generally, the researchers aim to create a data mining system that is capable of extracting information from a large set of collected compilation log data. Furthermore, this study aims to attain the following: 1) Apply the concept of data mining in the field of education; 2) Create an additional feature to DEV C++ IDE that is capable of generating compilation log file in text format; 3)Visualize and interpret collected data and 4) Provide a bridge between programming instructor and novice programmer.

2. LITERATURE REVIEW

2.1 Programming errors

Various studies about programming errors were already presented concerning more on the performance of students. A study from Bigrigget alia [5] constitutes the machine and the programmer activity towards conveying inaccuracy. They emphasize the importance of a compiler especially on how it can guide the programmer towards debugging. The study also insight factors that will help one user understand the error messages. In creating computer program, understanding different error messages is one of the factors that will guide the programmer. Hristovaet. al [6] categorized programming errors into three; syntax, semantic and logical error, and they defined it all as follows: syntax errors are those errors in the spelling, punctuation and order of words in the program that do not conform to the syntax of the programming language; semantic errors are those errors with valid programming structure with invalid logic, logical errors are caused by faulty reasoning. This type of error is a valid program but causes unexpected output. In the official website of Microsoft Developer Network, they have identified three kinds of programming errors, namely the compilation, run time and logic errors. According to them compilation errors are the errors prevent the program from running, run time error are errors encountered during the execution of the program and logic

2.2 Difficulties of novice programmer in understanding programming error message

Novice programmers usually experience a number of learning difficulties. These difficulties according to Pillay and Jugoo [7] were contributed by the poor planning and problem solving ability, lack of knowledge in programming language, lack of understanding in the application domain and lack of conceptualization in the execution of the program. In addition to that, are the difficulties of novice programmers in understanding the programming error as stated by Hristovaet. al [6]; understanding cryptic compiler message can be a daunting task for introductory students when they first start to write actual code. There are error messages that do not correspond to the problem or the compiler have used very technical messages that are hard to understand. This somehow affects their ability to provide solution, which leads to a wide range of misconceptions during the coding process. Even there are a lot of books and lecture slides which cover or discuss different types of errors, Histrovaet. al [6]; still observed that the problem in comprehending error messages still persist when the student does the actual coding.

2.3 Student's programming errors as an alternative way of weighing their performance

According to Stephen Edwards [7], a systematic strategy to be applied across the curriculum of the students learning programming will be much of use; strategies like providing student a rapid, concrete and useful feedback regarding their performance will be valuable enough to induce a cultural shift on how a student behaves. Novice student programmers believe that once the codes written compiles successfully, the codes are right and errors are gone. Even worse, students succeed at introductory programming courses without developing a broader view of the course. Ebrahimi, et alia, [1] said that investigating novices' problem with programming enables educators to understand students' obstacles and explore possible solutions. Most novice programming errors are related to: language construct misconception, plan composition errors, programming environment and the inability to get assistance. Some cases, errors are attributed with the lack of problem solving skills and poor understanding of programming concepts. In order to address this drawback, surveys are conducted among faculty members to identify what they consider to be the most important programming errors. Hristova, et alia, [6] categorized errors as syntax, semantic and logic errors. They insisted that preventing the following errors is considered essential for a solid foundation on learning how to program effectively. Marceau, et alia, [8] infer that error messages are arguably the most important point of contact between the programmer and the system. According to Johnson [9], programming errors might enhance the learning process of novice programmers.

Performance errors provide a unique opportunity for the teacher to understand the student’s confusions and misconceptions. Given such understanding, the teacher can then focus on remedying the student’s problem.

2.4 Parsing Text File

According to Flynn and Francis [10], parsing text involves identifying the spaces, punctuation, and other non-alphanumeric characters found in text documents, and separating the words from these other characters. Most programming and statistical languages contain character procedures that can be used to parse the text data. For instance, if the text data contains no punctuation and the words are separated by a space the algorithm for parsing words from spaces is: initialize an array to hold the words that are parsed. Search for the first space and record its position. Extract the string from the first position to the position before the next space. This is the first word. Find the next space and record its position. Use the positions of the previous and subsequent space to extract the second word. Repeat until all words are parsed. Text mining is an emerging technology that can be used to augment existing data in corporate databases by making unstructured text data available for analysis. Giles [11] provides a short introduction to text mining with a focus on insurance applications. One of the difficulties in getting started with text mining is acquiring the tools, i.e., the software for implementing text mining. With the enormous growth in digital text-based information, the efficiency and accuracy of search engines is a growing concern. In order to improve performance, novices and experts in information modeling need ways to study and evaluate various information retrieval (IR) models. To address this void in modeling software, we present an object oriented software environment called General Text Parser (GTP).

2.5 Knowledge Discovery Process and Data Mining

There has been a misconception regarding the meaning of data mining and knowledge discovery process. Some used these terms interchangeably, however there is a big difference between them. Knowledge discovery process is a process of discovering relationships and other descriptions from data, the whole process includes a collection of data until the visualization of result. Data mining refers only to the application algorithms for extracting patterns from data.

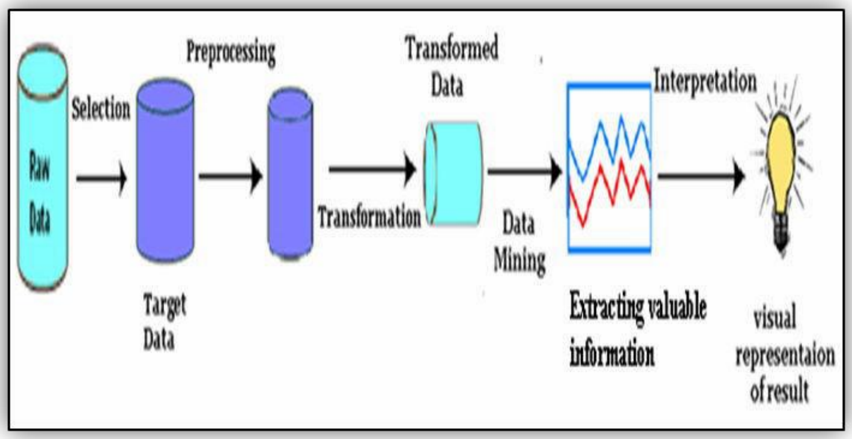


Figure 1. Knowledge Discovery Process

Figure 1 shows the process of knowledge discovery; the process starts with data selection. This step allows the selection of the relevant data that is suited to the chosen data mining task. Second phase, is the cleansing of data or preprocessing, concerned with treatment of corrupted data or creating strategies in dealing with missing data; it involves reduction of data to be analyzed. After the preprocessing is the core process: the data mining. Here, several algorithms can be employed, like neural networks, decision trees, association rules and genetic algorithms as enumerated by Han and Kamber [12], depending on the task chosen. Last is the interpretation of results of the discovered pattern aims to evaluate their utility and importance to the application domain. This phase includes interpreting the discovered patterns and possible visualization of the extracted patterns, removing redundant or irrelevant patterns and translating the useful ones into terms understandable by humans.

2.6 Educational Data Mining

As described by Baker [13], educational data mining is the area of scientific inquiry for making discoveries within a unique kind of data that comes from educational settings. Data mining contributed a lot to improve the educational standards of every institution. Heiner, et alia. [14] stated that educational data mining is the process of converting raw data from educational systems to useful information that can be used to inform design decisions and answers research questions. Further, data mining encompasses a wide range of research techniques using traditional options such as data queries and automatic logging. Data mining can help solve the learning styles or strategies to be an effective approach in teaching. This assertion was seconded by Merceron and Kalina [15] which stated that there seems to be a lot of potential for education and can bring their students learning and reflect on their teaching; also, it supports learner reflection and provides proactive feedback to learners. Baker [13] identified these general categories in educational data mining, the prediction, clustering, relationship mining discovery and distillation of data for human judgment. One of the common used of prediction is to predict student educational outcome, it can be based on their previous score or grade. Clustering, the goal here is to naturally group variables which naturally shared commonality. The third category is relationship mining, which goal is to discover relationships between variables; for example, a particular instructional strategy which generates high scorer student during the exam. And lastly is distillation of data for human judgment, which simply means is to visualize the information. With the application of data mining in an educational setting, this can help in determining the common learning styles or strategies that are effective. As said by Britos, et alia, [14], data mining can be an effective way of discovering new knowledge from data sets of educational processes, data generated by learning systems or experiments. All these discovered information can be used to improve adaption and personalization of the student involved and to help the teachers discover the students' misunderstanding causes: a very promising issue to explore as a new diagnosis tool area.

2.7 Data Mining and Statistics

According to Kuonen [16], data mining is like statistics because they both deal with "learning from data" or "turning data into information". Statistics includes planning for the collection of data, data management, drawing inferences and presentation of results. Berry and Linoff [17], said that, "data mining is the process of exploration and analysis, by automatic or semiautomatic means, of large quantities of data in order to discover meaningful patterns and rules". From the "Insightful Miner 3.0 User Guide" it is said that, data mining is simply the application of statistics to reveal patterns and trends in a very large data sets. Kuonen [16] concluded that data mining can be learned from statistics. Data mining and statistics will grow toward each other in the near future because data mining will not become knowledge discovery without statistical thinking. Statistics will not be able to succeed on massive and complex datasets without data mining approaches.

2.8 Data Collection Mechanism

To collect data from every student while they are performing laboratory exercise is one of the most essential part of this study. As described by Cheong Vee, Meyer and Mannock [18], they automated data collection in their study with the help of a compiler; storing the "snapshots" of student programs at every compilation. The resulting interaction log allows the developer to explore the behavior of students while they perform a particular programming task. Moreover, Dy, et. al. [4] described their own mechanism to collect data from students by connecting each computer in a network, followed by installing BlueJ. BlueJ is an integrated development environment, used to send each logs of each compilation to a central server. These logs contained the source code, error and line number, if any,

2.9 Data Visualization and Presentation

Another area of interest within educational data mining is the distillation of data for human judgments. Baker [13] laid down the methods in this area of educational data mining which are information visualization methods. Data is displayed in ways that enable a human being to easily identify well-known patterns that are nonetheless difficult to formally express. Raw data and algorithm results can be visualized through tables and graphics such as graphs and histograms, as well as through a more specific technique such as symbolic techniques, Merceron and Kalina [15]. It aims to present the data in ways that it may be obvious for human eye. Cheong Vee, Meyer and Mannock [20] also added the use of both statistical and non-statistical techniques to analyze data that provide insights in helping students learn programming.

3. DESIGN AND METHODOLOGY

Figure 2 shows the analytical framework of the study, every DEV C++ IDE have this compilation logger feature that starts automatically as the DEV C++ IDE is opened for the session. A text file containing compilation logs is generated after every session. The compilation logger feature will be responsible of writing all generated error messages to the created text file. This compilation log file generated is automatically sent to a local server and fed to a data mining system. This data mining system, analyzes and digs in information from these collected data, and visualizes this information.

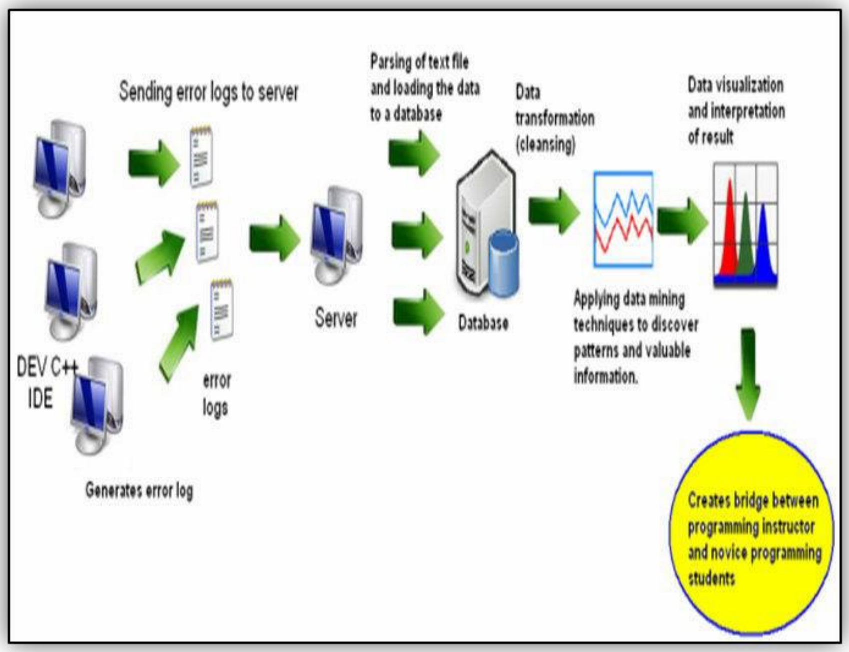
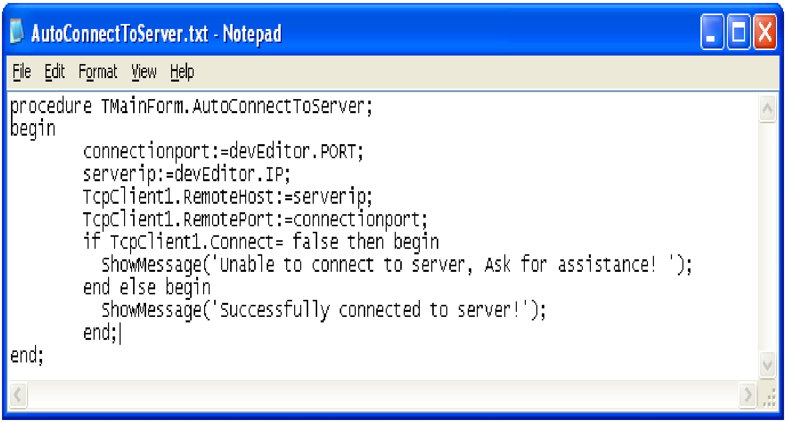

Sending error logs to server
Parsing of text file and loading the data to a database
Data transformation (cleansing)
Data visualization and interpretation of result
DEV C++ IDE
error logs
Server
Database
Applying data mining techniques to discover patterns and valuable information.
Generates error log
Creates bridge between programming instructor and novice programming students

Figure 2. Analytical Framework

The methodology used in this study consisted of five stages: modification of the Dev C++ 5.1.1.1 Beta Version, creation of Dev C++ Log Miner that will act as server system, the installation of the modified Dev C++, system implementation during the laboratory exercise of Computer Programming 1 students and the Data Analysis for generating summaries and report.

3.1 Modifying Dev C++ 5.1.1.1 Beta Version

Dev C++ is an integrated environment for computer programming 1 in the University of Southeastern Philippines – Institute Of Computing. Researchers modified the Dev C++ and added a function that will capture the compilation log information of the novice student. The compilation log information sent consists of compilation time, number of characters, total number of errors and the corresponding code of the error. The changes in the Dev C++ IDE were installed in all computer units. The feature added was build using the Delphi programming language. Shown in figure 3 is the added code for automatic connection during start up of the modified Dev C++ IDE. This function was added to automatically establish a connection between the client and the server during start up of the modified Dev C++ IDE.



```
procedure TMainForm.AutoConnectToServer;
begin
        connectionport:=devEditor.PORT;
        serverip:=devEditor.IP;
        TcpClient1.RemoteHost:=serverip;
        TcpClient1.RemotePort:=connectionport;
        if TcpClient1.Connect= false then begin
          ShowMessage('Unable to connect to server, Ask for assistance! ');
        end else begin
          ShowMessage('Successfully connected to server!');
        end;
end;
```

Figure 3. Code for Automatic Connection

3.2 Creating the Dev C++Log Miner

Dev C++ Log miner was created using the Java language and utilizes MySQL component of the WAMP package to create database for the data gathered. JFreeChart, an open source Java library, was also used for displaying charts. Instructors need to register to the system to be able to create classes and programming sessions. When Dev C++ Log Miner starts, client computers connect to the server by configuring the connection setting; specifying the Dev C++ Log Miners' IP Address and connection port. The Dev C++ Log Miner receives generated text file from the modified Dev C++ IDE when students submit their logs. Gathered text files were temporarily stored as it is in the assigned directory. When the session ends, and the server is stopped, the contents of the text files are then transferred into the database for analysis. The Dev C++ Log Miner analyzes the compilation log information received from clients and produces reports; visualization and interpretation of results.

3.3 Installation of the modified Dev C++ IDE

At least 30 functional units in the computer laboratory were used for deployment of the system. These units were connected via local area network and were installed with the same operating system and the modified Dev C++developed to track the compilation activities of the students. The modified Dev C++ connection settings were properly configured to connect to the server. Configuration settings such as IP address and connection port were specified before the implementation.

3.4 System Implementation

After the successful installation of the modified Dev C++ IDE, system implementation begins. The system was implemented during the computer programming 1 class of one of the IC instructors. Students were instructed to use the modified Dev C++ IDE installed in the units. A laboratory exercise prepared by the instructor was given to the students and thirty minutes was allotted for the exercise. After thirty minutes, the instructor evaluated the students' program to give corresponding points as to what they have created. Remarks were categorized as perfect points, partial points and failed. Once program has been evaluated, students were instructed on how to submit their logs; first is to key in their ID number, then their corresponding remarks. Dev C++ prompts the students if they have successfully submitted their logs to the Dev C++ Log Miner system.

3.5 Data Analysis

After all the compilation logs were sent to the server, contents were parsed for data analysis. Specifically, the content was parsed according to compilation time, number of characters, number of errors and the corresponding code of errors. After all the required information was parsed, this forms a large set of collected compilation log information which will be used to generate different summaries and report.

The following summaries that can be generated:

3.5.1 Error Count per Compile

The error count per compile showed the frequency of the errors generated between compilation events. From the individual view, instructor can identify which students are struggling on solving their programming errors. From the class view, instructor can identify which type of errors the students were struggling in relation to the given laboratory exercise.

3.5.2 Committed Error Count

The committed error count shows the total number of a particular error encountered. From the individual view, students' error will be easily identified and what particular error they are struggling to solve. From the class view, instructor can classify which errors are mostly encountered.

3.5.3 Time interval

The time interval shows the interval between the compilation events of the students. From the individual view, students are as-

3.5.4 Character Count per Compile

The character count per compile shows the number of characters between compilation events of the students. From the individual view, students are easily identified if they are working their laboratory exercise because every compilation sequence event, the number of characters they made will be shown. From the class view, instructor can easily identify those students who are working very hard to create a running program.

4. RESULTS AND DISCUSSION

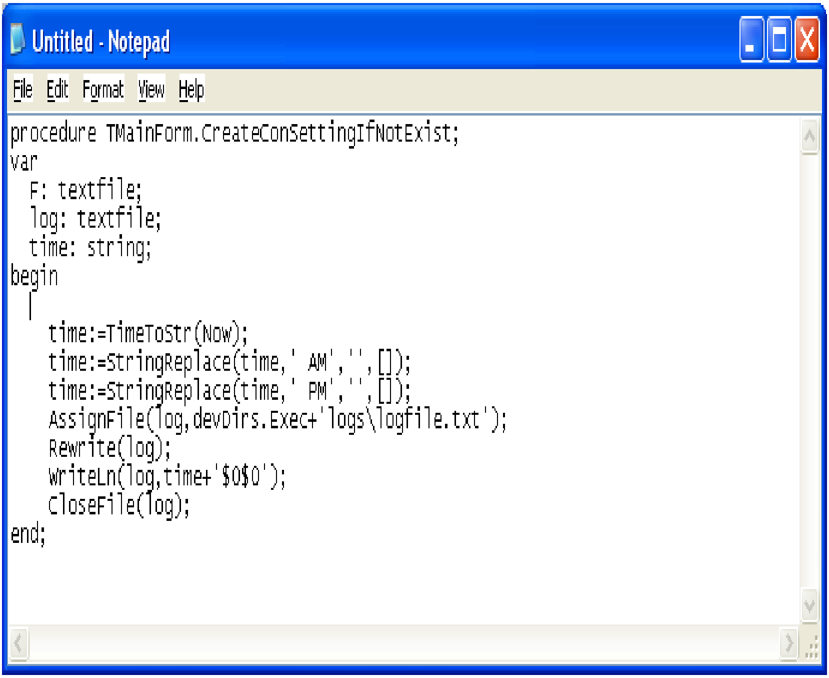
One of the major advantages of using Dev C++ IDE is that, it is an open-sourced software wherein developers can easily modify its functionalities. The researchers used the Dev C++ 5.1.1.1 beta version by Orwell. This version of Dev C++ was modified in order to gather compilation log information.

4.1 Data Mining in Education

The concept of data mining was applied to assess the programming skills of students taking up computer programming 1 of the University of Southeastern Philippines – Institute Of Computing. Researchers ascertained that data mining can be used to gather reliable data from an educational setting, and be helpful in improving the institutions educational standards. The process of data mining in the scenario starts by gathering compilation logs from students doing a particular programming exercise using the Dev C++ IDE. Compilation logs were written on a text file and then sent to the server. The DevC++LogMiner had analyzed all the data collected producing visualization and interpretation of the result.

4.2 Generating Compilation Log

The researchers added a function that can get the compiled time, the character count, the total errors committed and the error itself. The feature was created in order to write the compilation log information to a text file. The researchers provided a list of error messages with its corresponding code that will served as the source of what particular error code is to be written on the text file. The process was used to map the generated error message to the list provided. During the initial run of the modified Dev C++ IDE, the program have automatically generated a log file containing the time of opening the program, zero character count and zero error count.



```
procedure TMainForm.CreateConSettingIfNotExist;
var
  F: textfile;
  log: textfile;
  time: string;
begin

    time:=TimeToStr(Now);
    time:=StringReplace(time,' AM','',[]);
    time:=StringReplace(time,' PM','',[]);
    AssignFile(log,devDirs.Exec+'logs\logfile.txt');
    Rewrite(log);
    WriteLn(log,time+'$0$0');
    CloseFile(log);
end;
```

Figure 4. Code for Initializing Log File

Shown in Figure 4 is the code to get the information between the initial run of Dev C++ up to the first compilation event. As students complete the programming exercise and submit their logs, the modified Dev C++ IDE automatically send the generated text file to the server. The researchers applied the client-server system in this study, wherein the client Dev C++ IDE 5.1.1.1 beta version was modified to connect to a server and be able to send the generated compilation log. DevC++ IDE was modified to check the connection on the server every time it runs, and will connect if the server is available and ready to accept connection from clients.

4.3 Visualization and Interpretation of Results

The data mining system, Dev C++ Log Miner allowed the researchers to gather compilation logs of every student performing the laboratory exercises. One laboratory session was performed to test the reliability and efficiency of the system. The data mining system gathered a total of one thousand five hundred forty three (1,543) errors committed by the students while solving the programming exercise.

4.3.1 Committed Errors

Shown in Table 1 are the top 10 committed errors of student's performing the machine problems. The presentation was limited to top 10 errors so that programming instructor can focus on to these particular errors, and be able to address novice students' difficulties.

Error Message	Percentage
'*' was not declared in this scope	25.86%
expected '*' before '*' token	14.00%
candidates are:	10.17%
no match for '*' in '*'	7.91%
'*' does not name a type	4.93%
expected declaration before '*' token'	4.67%
expected unqualified-id before '*' token	4.54%
invalid conversion from '*' to '*' [-fpermissive]	3.82%
expected constructor, destructor, or type conversion before '*' token	2.66%
expected primary-expression before '*' token	2.46%

Table 1. Top 10 Committed Errors

The most committed error '*' was not declared in this scope with 25.86% means that the compiler could not identify the '*'in the program. This implies that students do not have either enough knowledge about initializing variables or forgot to declare it. Also, they could not determine what particular header file was to be used for the program. Another error which is '*' does not name a type with 4.93%, was often committed whenever a keyword or variable was called outside the scope of any function. This means that students do not have enough idea about the scope of a particular variable declared within a function. This was also committed whenever there was an excess of curly braces "{}". Lastly, the error expected primary-expression before '*' token with 2.46% is committed if logical operators were misused. Example: if(var1 !== var2). Knowing these percentages, the instructor now had an idea on what particular concepts, or programming structure is to be discussed in order to address students' difficulties.

4.3.2 Average Compile Time Interval

Figure 5 presented the Average Compile Time Interval of the novice students during their laboratory exercise. By analyzing these time intervals, programming instructors are given an idea about the ability of novice students to fix errors. The x-axis represents the ID number of the participants or respondents while the y-axis implies the average time interval in seconds. The longer average compile time interval a novice student takes implies that he is having difficulty with fixing errors.

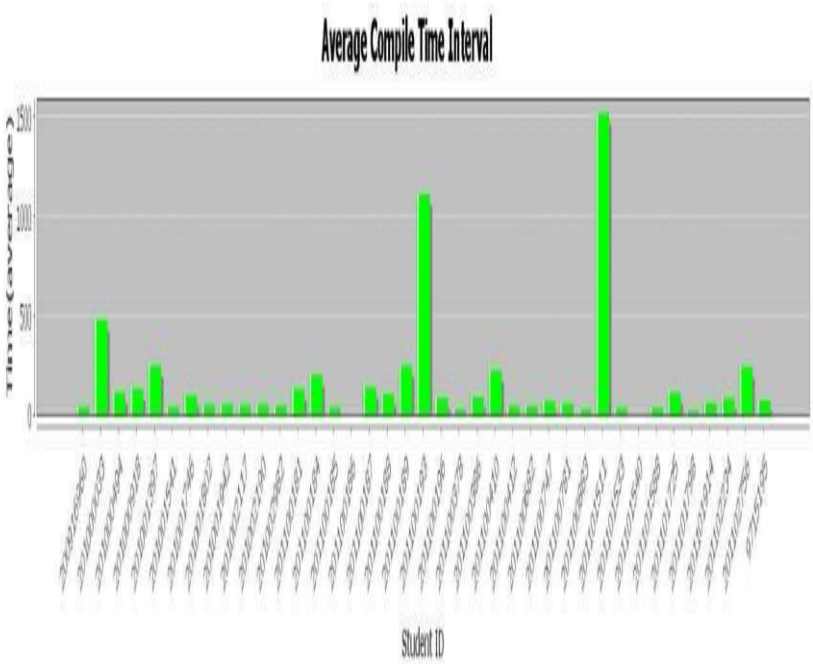


Figure 5. Average Compile Time Interval

As shown in Figure 6, around fifteen percent of novice student performing the exercise takes longer time (approx. 250 – 300 second) to compile; five percent of them never attempted to compile nor perform the exercise; and the remaining 80 percent on the average are compiling within 50 to 150 seconds. Since 80 percent and majority of novice students are compiling at a seemingly acceptable time, this means that most of the students are working very hard on their given exercise, attempting to fix committed errors; are having deeper knowledge about the program constructs, and understanding of error messages. Provided by this information, instructor can easily identify those students who are working on their machine problems and those who did not even tried to fix their errors.

4.3.3 Compile Count

Figure 6 shows the number of compilation events performed by the novice students. The x-axis represents the ID number of the student and the y- axis corresponds to their compile count.

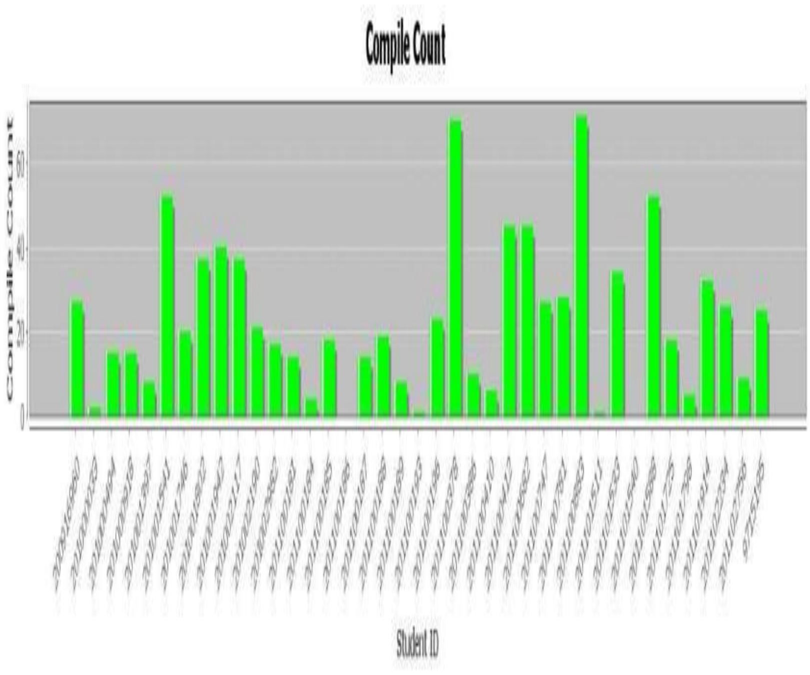


Figure 6. Compile Count

Analyzing this graph gives the programming instructor a hint about performance or activity of the student. If we are to look closely on Figure 6, it can be observed that students 2011-00186 and 2011-01540 neither attempted to compile any codes, nor worked on the exercise given. This rate allows the programming instructor to think of disciplinary actions for the following students. Thirty seven percent are presumed to be working as they are compiling regularly.

4.3.4 Error Count per Compile

Shown in figure 7 is the error count per compile of every student's performing the laboratory exercise. The x-axis of the graph represents the compilation sequence of every student, while the y-axis represents the number of errors committed.

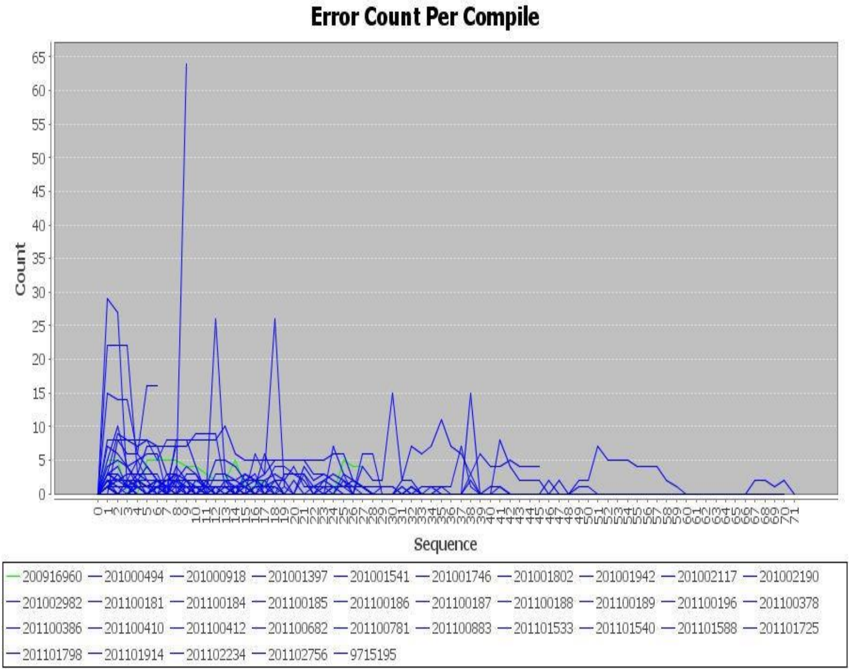


Figure 7. Error Count per Compile

If we assess the graph in Figure 7, notice that from compilation sequence 1 to 20 of students, numerous errors were committed. However as they progress, lesser errors were committed; and presumed that students were able to fixed the errors accordingly.

4.3.5 Character Count per Compile

Shown in Figure 8 is the character count per compile of every student's performing the exercise. The x-axis of the graph represents the compilation sequence of every student, while the y-axis represents the character count.

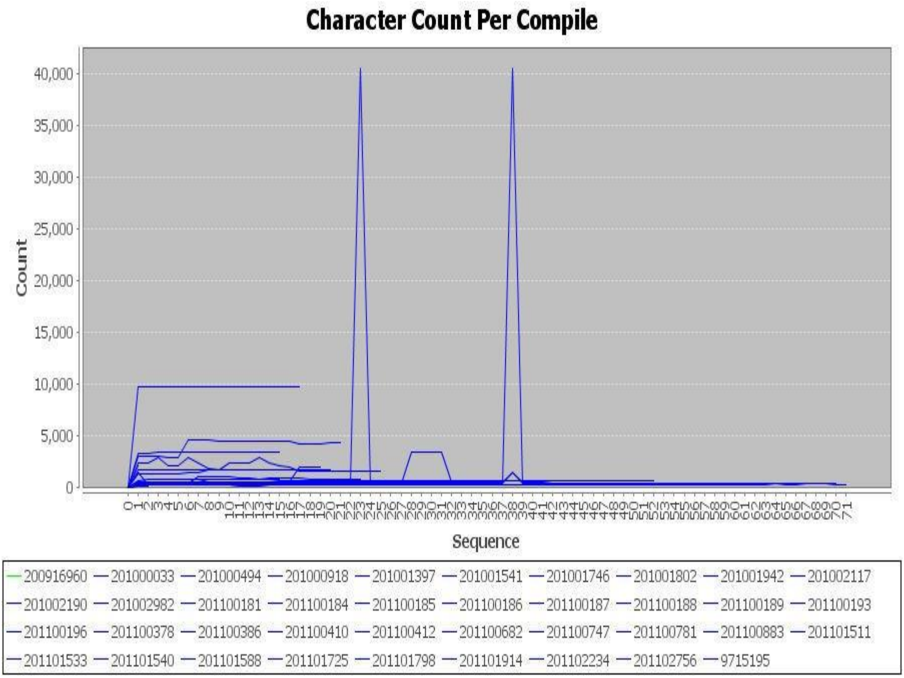


Figure 8. Character Count per Compile

Evaluating the graph shown in Figure 8, a noticeable two compilation events have compiled about 40,000 characters while the others were regularly compiling less than 5,000 characters. This would mean that a particular student is either coding the laboratory exercise manually (that is creating it without loops, etc.), copying and pasting code snippets, or compiling other source codes and files.

Analyzing the graph for character count per compile will give the instructor an idea if students were really working on the exercise; and not just repetitively compiling the same source code.

4.3.6 Success Compile Rate

Success compile rate gives the instructor an idea if students were able to fix committed errors. Figure 9 shows the success compile rate of every student performing the laboratory exercise. The x-axis represents the students ID while the y-axis represents the percentage

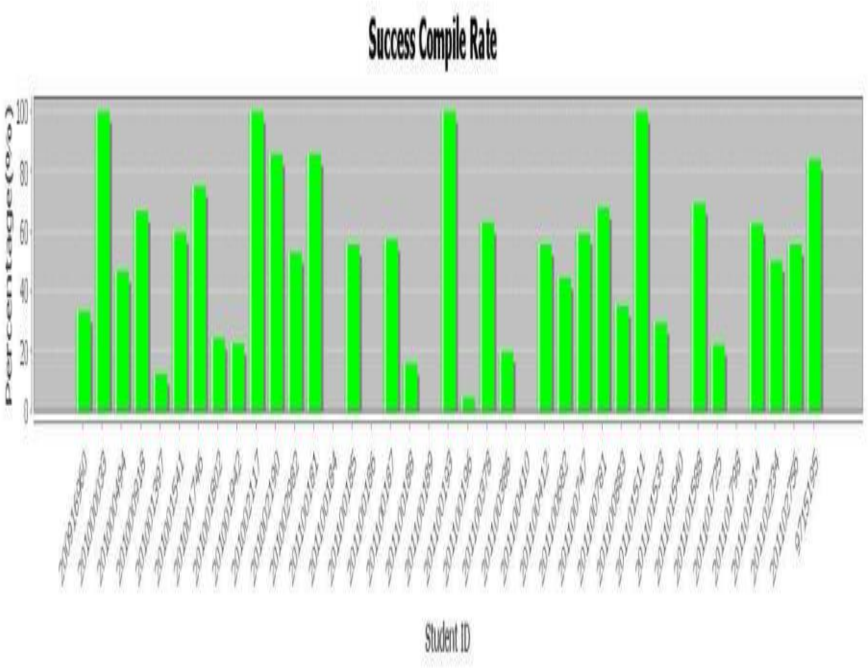
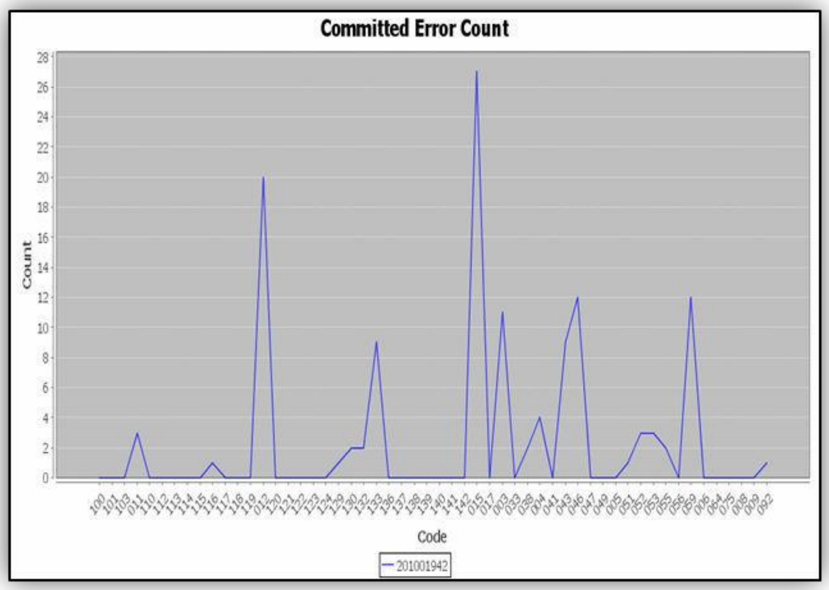


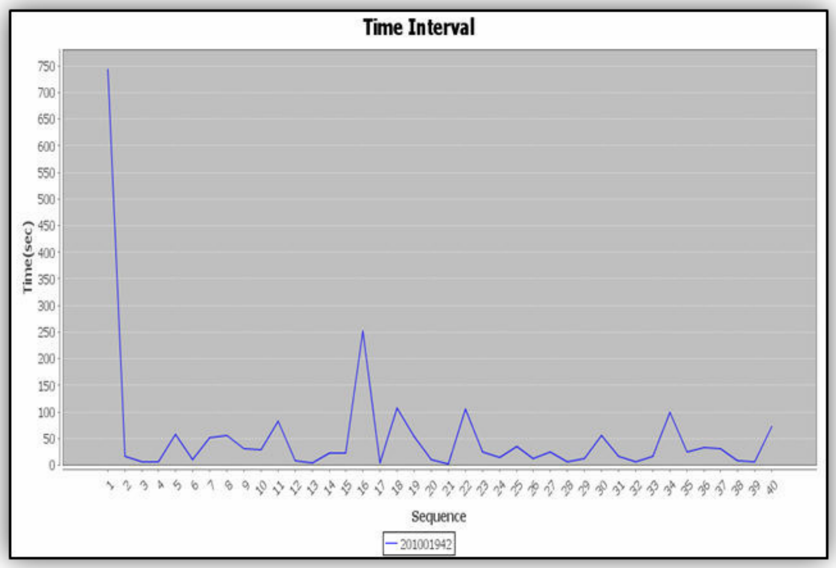
Figure 9. Success Compile Rate

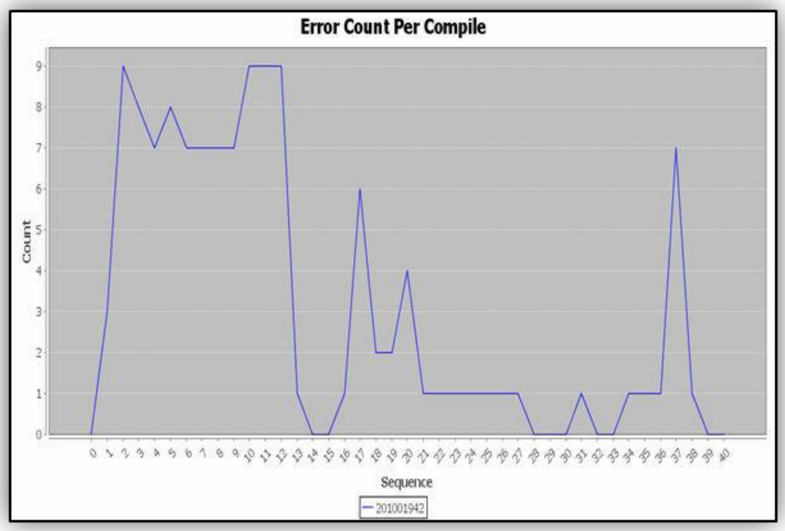
Analyzing the graph shown in Figure 9, roughly forty percent of students had reached a fifty percent and above success compile rate. It implies that only forty-eight percent of the novice students are able to address or fixed their errors. The remaining fifty-two percent of the class were still having difficulty in fixing errors, giving them unsuccessful compiles.

4.3.7 Individual Views of Results

Shown in figure 10 are the individual views of results. Programming instructor was given this choice of view to extend the instructor’s assessment of one student.


Committed Error Count
Count
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
100
101
103
011
110
112
113
114
115
116
117
118
119
012
120
121
122
123
124
129
130
132
133
136
137
138
139
140
141
142
015
017
003
033
038
004
041
043
046
047
049
005
051
052
053
055
056
059
006
064
075
008
009
092
Code
201001942


Time Interval
Time(sec)
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
Sequence
201001942



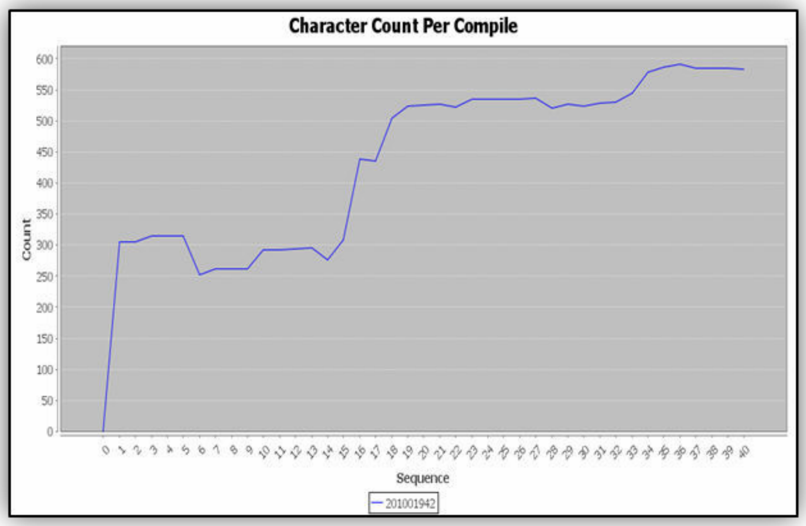


Figure 10. Individual Views of Results

Individual view of results gives the instructor an idea about the performance of a particular student. Instructor will be informed of the pursuit or activity of the student during the exercise.

5. SUMMARY AND CONCLUSIONS

This study basically addresses or helps novice students and programming instructors to understand the difficulties in learning Computer Programming 1. We targeted one of the contributing factors that causes these difficulties; the compilation error messages. We added a feature on Dev C++ IDE to collect the required information on the compilation activities of each student during laboratory exercises. The feature added contains function implemented in Delphi an object-oriented derivative of Pascal that gathers all the compilation log information

Generally, the system was able to extract information from the collected generated text file that contains the compilation log information of every student participant. In extracting the information; the researcher used the data mining approach that provides a summarization of the useful information. This information were successfully classified according to total number of errors committed, character count, compile time interval and the success compilation rate of the subject.

The system was able to provide visualization of records of the students' most frequently committed errors generated, as well as the time they spent between compilation events. The data shows that most novice student programmers encountered similar errors in their programming exercises. Thus, it is an indication that they are having complexity to avoid some particular errors. With this information, instructors can focus on the specific topic to help the students be aware and overcome their difficulty. The results on average compile time interval shows another data about the students' compilation behavior. Their determination to overcome faults on constructing a running program, have a high percentage on generating successful compilation that can be useful to the instructor to monitor those students who are serious on the course.

We already presented some broad analyses applicable for this system; thus, the researchers recommended this study to be tested to two or more sessions to evaluate further the students' compilation log information. The researchers also recommends to have a university server for collecting large error logs and that error logs must be encrypted for security purposes so no one can perform modifications or alterations. The study also needs to be tested on linux operating system. The compilation logs must be automatically sent to the server every compilation made by the novice students and enable the expert system to rank the novice students depending on their marks obtained and analyze the students' performance every after the programming session.

6. REFERENCES

[1] A. Ebrahimi, Novice Programmer Errors: Language Constructs and Plan Composition. Internal Journal of Human-Computer Studies, Volume 41, Issue 4, pages 457-480, October 1994.

[2] Ranjan and Khalil, Conceptual Framework of Data Mning Process in Management Education in India: AN Institutional Perspective. Retrieved from http:// scialert.net/ abstract/?doi=itj.2008.16.23, 2008.

[3] Jadud, M. A First Look at Novice Compilation Behavior. Computer Science Education Vol. 15, No. 1, pp. 25 – 40, March 2005.

[4] Dy, etaliaia. Development of a Processor That Detects Non-Literal Java Errors. Proceeding of the 10th Koli Calling International Conference on Computing Education Research, Pages 118-133, 2010.

[5] Bigrigg,M. et aliaia. An Evaluation of the Usefulness of Compiler Error Messages.ICES Carnegie Mellon University Technical Report 04-9-03, 2003.

[6] Hristova, M., et alia. Identifying and Correcting Java Programming Errors for Introductory Computer Science Students.SIGCE'03 Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, 2003.

[7] Edwards, S. Rethinking Computer Science education from a Test-First Perspective. Virginia Tech, Department of Computer Science, VA. Proceeding OOPSLA '03 Comapaning of the 18th Annual CAN Sigplan conference on Object-oriented Programming, Systems, Languages, and Application, pages 148-155, 2003.

[8] Marceau G., et alia. Measuring the Effectiveness of Error Messages Designed for Novice Programmers. Proceeding of the 42nd ACM Technical Symposium on Computer Science Education, Pages 499-504, 2011.

[9] Johnson, W. Understanding and Debugging Novice Programs. USC/ Information Science Institute, CA. Journal Artificial Intelligence-Special issue on Artificial Intelligence and Learning Environments, Volume 42, Issue 1, pages 51-97, 1990.

[10] Flynn, L.Text Mining Handbook. Retrieved from: www casact.org/pubs/

[11] Giles, J. .GTP General Text Parser Software for Text Mining.Retrieved from: web.eecs.utk.edu /~berry /papers02/GTPchap.pdf, 2002.

[12] Han and Kamber , Data Mining: Concepts and Techniques. Retrieved from http://www.cs.wmich.edu /~yang/teach/cs595/han/ch01.pdf, 2000.

[13] Baker, R. Data Mining for Education.Carnegie Mellon University, Pittsburg, Pennsylvania, USA. Retrieved from: http://users.wpi.edu/~rsbaker/ Encyclopedia% 20Chapter%20Draft%20v10%20-fw.pdf, 2008

[14] Heiner, et alia. Educational Dataming. Retrieved from http://nth.wpi.edu/ pubs_ and_grants/papers/2007/AIED-ED M/AIED-EDM_proceeding_full2.pdf, 2007

[15] Merceron and Kalina. Educational Data Mining: a Case Study Retrieved from ftp://ftp.cs.pitt.edu/web/ projects/nlp/conf/aied2005/FAIA125-0467.pdf, 2005.

[16] Kuonen, D., A Statistical Perspective of Data Mining .CRM Zine (Vol. 48).CRM Today. 2005.

[17] Berry, G., GTP (General Text Parser) Software for Text Mining. University Of Tennessee knoxville, USA. Retrieved from: -http://citeseerx.ist.psu.edu/ viewdoc/ download?doi=10.1.1.77.4888&rep=rep1&type=pdf, 2011.

[18] Cheong Vee, Meyer and Mannock. Understanding novice errors and error paths in Object-oriented programming through log analysis Retrieved from http:// www. educationaldatamining.org/ITS2006EDM/ngcheongveecr.pdf, 2006

PAPAYA DEFECT CLASSIFICATION USING ARTIFICIAL NEURAL NETWORK

Cristina E. Dumdumaya, Ronnie Kris M. Dominguez Lorraine F. Donasco, Mildred O. Juaton, Karl Oliver G. Nonesa

ABSTRACT

Non-destructive and unbiased evaluation procedures are necessary in domains where high quality standards are highly expected like in the area of agriculture. This study explored the application of computer vision in agriculture through the development of an inspection system to detect and classify the surface defects of solo papaya fruits. The process is divided into three phases: image pre-processing, network training and disease classification.

In phase one, the useful features of the input papaya image are extracted. These features are used as input to the neural network. Phase two involves the initialization and training of the neural network using a set of images of two papaya diseases namely: Anthracnose and Ringspot virus using Scale Conjugate Gradient (SCG) back propagation algorithm. Phase three is the classification of papaya diseases.

The classifier was tested in several cycles to refine the accuracy of the system in detecting and classifying papaya surface defects. The results of the tests showed that the system performed the classification of healthy, defective-anthracnose, defective-ringspot and unclassified diseases with an average accuracy level of 83.5%. This study demonstrates that image processing and neural network are efficient tools for non-invasive quality inspections of agricultural products.

I. BACKGROUND OF THE STUDY

Computer vision has become one of the prominent solutions to enhance quality in several domains; whether in manufacturing, commercial or agricultural fields, automated inspection system is seen as an innovative solution to ensure quality of products. In agriculture, traditional quality assurance methods are performed by human experts based on subjective multifactor interpretation of quality parameters such as size, color, shape and the presence of defects or diseases. Human sensors are expensive and slow and quite often manual inspection results to inconsistency of quality standards. More so, optimal defect detection is complicated by the fact that biologically formed products do not have ideal quality templates. It exhibits non-standard and non-uniform characteristics and the acceptable defect limits periodically change (Majidi, 2003).

Papaya is a highly valued agricultural product yet said to be one of the most sensitive fruits to produce. It is prone to post-harvest and pre-harvest deformities. Post-harvest blemishes are often caused by improper handling while pre-harvest defects are usually caused by viruses and fungi, insects, climate change, or too much exposure to chemicals (Gross, 2003). These problems degrade production volume and product quality. Two of the most widespread diseases of papaya are anthracnose and ringspot. Papaya anthracnose is developed due to rainy and humid climates. Symptoms include water soaked sunken spots one-quarter to one inch in diameter. The center of the defect turns pinkish when the fungus produces spores. The flesh beneath the spots becomes soft and watery and spreads to the entire fruit. Ringspot is a disease caused by Papaya Ring Spot Virus (PRSV). The virus, transmitted by species of aphids and spreads fast among plants, is considered a limiting factor for commercial papaya production in many areas of the world. It reduced crop production in Hawaii, the Caribbean, Brazil, Southeast Asia and other papaya growing countries. The physical damage brought by the disease caused post-harvest losses in papaya (Stice et. al., 2009).

Philippines is one of the world's largest papaya producers. From 1999 to 2004, the country produced an average of 77,000 tons, making it the world's 14th largest papaya exporter. In 2003, Philippine papaya export to Japan was restricted because of fruit flies found in the fruits (McGregor and McGregor, 2009). Philippine papaya which made inroads into the world market suffered because of the adverse effects of PRSV and other diseases. The foregoing scenario motivated the researchers to develop a papaya disease classifier utilizing computer vision.

II. SIGNIFICANCE OF THE STUDY

Visual inspection system is a computer-based method used for quality control. This scheme offers an alternative solution for recognition and analysis of meaningful patterns in various domains. In this research, the use of computer vision to detect and classify papaya defects was explored. The potential beneficiaries of this study are the following:

Papaya Fruit Industry

The result of the study will provide an accurate method of papaya defect detection and classification. Automation of the process reduces error and variations therefore improving quality, productivity and profitability.

USEP- Institute of Computing and College of Agriculture

Agriculture is a major thrust of the university. This study hopes to introduce the use of computer technology in crop quality control, production and management. Further, the output of this study anticipates a possible linkage between technology and agriculture to further strengthen the research potential of the university.

Students

The field of agriculture is an enormous and very interesting area for research. Researches in this field bring great impact not only to the researchers but to a larger set of the society. This study hopes to encourage the students and future researchers to explore the integration of computer technology and applied sciences.

III. OBJECTIVES

The generally objective of this study is to create an automated inspection system that is capable of detecting and classifying papaya surface defects. Further, this study aims to attain the following:

- To apply the concept of computer vision in the field of agriculture.
- To use digital image processing algorithms and Artificial Neural Network in detecting and classifying papaya fruit defects.
- To test the accuracy of the system in detecting and classifying papaya defects.

IV. SCOPE AND LIMITATION

The visual inspection system for papaya defect classification was created using the image processing functions and Neural Network Toolbox of Matlab 2010. It runs in a 32-bit, Microsoft XP operating system in a stand-alone environment. The system only accepts cropped images of the papaya fruit defects in JPEG format. The input image must have a minimum size of 18x20 pixels and a maximum size of 500x500 pixels. The classification of papaya diseases is only based on surface defects and limited only to Anthracnose and Ringspot diseases.

V. RELATED LITERATURE AND STUDIES

Several applications using computer vision algorithms have been developed and utilized to derive unbiased evaluations of plant traits and diseases. These methods were studied to improve farm yield and crop quality. Some of the literatures and works related to the study are discussed in this section.Moncada

Image processing, description, shape and shape discrimination. Input image was acquired by scanning the grains using a flatbed scanner. The image background was extracted through thresholding and the grain's shape signatures were described using Fourier transformation. The shape signature is defined using one dimensional function describing the two-dimensional shape boundary. Shape discrimination was achieved using morphological algorithm. Morphological method is a model-based approach which processes binary and gray scale images based on shapes. In this procedure, the obtained binary and gray scale images served as the input and output where each pixel value in the output image was based on the corresponding input pixel and its neighbors. By choosing the neighborhood shape appropriately, a morphological operation sensitive to the specific shapes of the input image can be constructed (Elbehiery et. al.,2007).

Sannakkiet. Al.(2011)developed an expert system for grading plant leaf diseases by means of image analysis and Fuzzy Logic. Possible infected leaf regions were identified using K-means clustering algorithm while diseases grading was done using fuzzy logic based on a predefined scoring scale. Triangular membership function was used to define the variables and rules of the fuzzy logic component which takes Percent Infection as the input and outputs the disease grade.The K-means clustering technique partitions n observations into k clusters where each observation belongs to the cluster with the nearest mean(Valliammal,2012)*.*

Another approach for plant diseases classification was implemented using an Artificial Neural Network(ANN). The methodology utilized texture feature analysis to detect the infected areas of the leaf. This was done by converting the RGB input image to Hue Intensity Saturation (HIS) format and analyzing the green pixel intensity by comparing it to a predetermined threshold value. The unhealthy regions were extracted by removing the mostly green colored pixels. The extracted areas were partitioned into equally sized patches. Each partition was analyzed and only those useful segments were utilized for computation of the texture features using color co-occurrence methodology. The infected area was characterized using gray level co-occurrence methodology which statistically measures neighboringgray level dependence. This derived the co-occurrence features such as contrast, energy, local homogeneity, shade and prominence. Disease recognition was done by calculating minimum distance criterion and using supervised learning methods for classification and regression through Support Vector Machines (SVMs) (Arivazhagan **et. al., 2012).**Riyadi et.al. (2007) also used ANN and image analysis in estimating papaya weight. Shape characteristics analysis, using excess green ExGcolor filter, was employed to extract the object from both its shadow the background. Region filling technique using morphological algorithm was utilized to remove the noise affecting the image quality. Finally, back propagation algorithm of the multilayer perceptron (MLP) ANN model was used for training and testing. The system successfully implemented the papaya grading system by yielding an accuracy rating of more than 90%. Rokunuzzamanet. al. (2005) sorted tomatoes using machine vision. Defect sorting was based on color image processing using Hue Saturation Value (HSV) significant value. The hue matrix was separated from HSV matrix. Two types of tomato defects were detected: blossom end rot and cracks. The ANN component was trained with images which were grouped into two categories: (1) images recognized as calyx images and (2) good images of fresh tomatoes. The neural network used back propagation technique and set the target for the defective images as ‘1’ and non-defective images as ‘0’.

VI. TECHNICAL BACKGROUND

The following tools were used in the implementation of the project.

Software

MATLAB – is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB has a variety of toolboxes that can be used for algorithm development, modeling, simulation and prototyping, data analysis, exploration and visualization, image processing, and application development. The Graphical User Interface and the defect detection and classification functionalities of the application were all developed using MATLAB. The ANN training of the system used the *Scaled Conjugate Gradient Backpropagation* (SCG) algorithm in MATLAB’s Neural Network Toolbox.

Hardware

- Personal Computer - a desktop or laptop computer used for the development and testing of the papaya defect classifier.

Minimum Requirements

- Running Windows XP Service Pack 3 or later
- Any Intel or AMD x86 processor supporting SSE2 instruction set
- At least 2 GB RAM
- Disk Space: 1 GB for MATLAB only, 3–4 GB for a typical installation

VII. DESIGN AND METHODOLOGY

Figure 4 illustrates the conceptual framework of the papaya defect classification system. Prior to the classification process is the network training process. A selected set of validated images were loaded into the system for neural network training. The network was initialized based on the input weights and trained to create the output weights using the Scale Conjugate Gradient (SCG) Back propagation algorithm wherein input neurons are multiplied to the weights then added to the bias before proceeding to transfer function (tansig) to produce a single output defects.

In the classification stage, the preprocess function extracts the features of the input image. The extracted features are used as input to the neural network. The neural network computes the extracted feature, also called as vectors, following the formula **output= f (∑weight*input + bias)** and returns an output value for classification. The return value was then compared to the dataset value to output the corresponding classification.

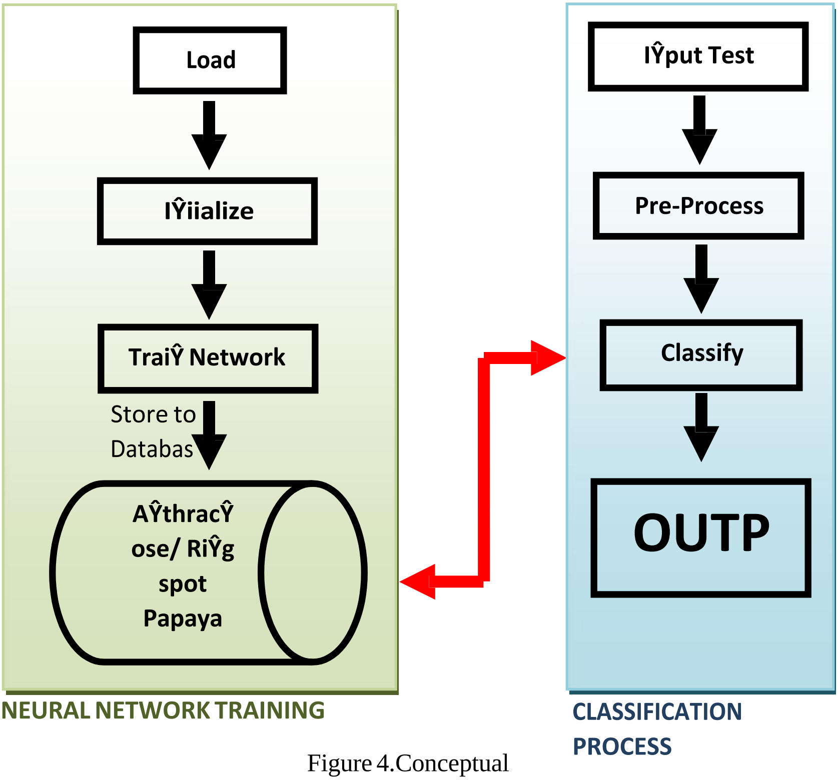


Figure 4.Conceptual

Methodology

The methodology used in this study consisted of four stages: Data gathering, Design, Development and Training and Testing.

A. *Data Gathering*

In the data gathering process, the researchers collected solo papaya fruit samples from the different public markets in Davao City. The samples were brought to a plant pathologist for evaluation and classification. The samples were also downloaded from the internet to comprise the dataset of ringspot disease. The downloaded images were likewise presented to the pathologist for evaluation and approval. The approved images of anthracnose and ringspot infected papayas were divided into two groups. One for neural network training, while the other for testing. Unqualified data were considered as *unclassified*. These data were also used for system testing together with the approved images.

B. Design

The design phase set up the structure of the Papaya Defect Classification System. A user interface was created, properly laying out the controls, visualizations and displays to give an easy and friendly environment to the user. Also, the System Algorithm was put into blueprint to guide the researchers in developing the system. The algorithm design of Papaya Defect Classification System is shown in Figure 5.

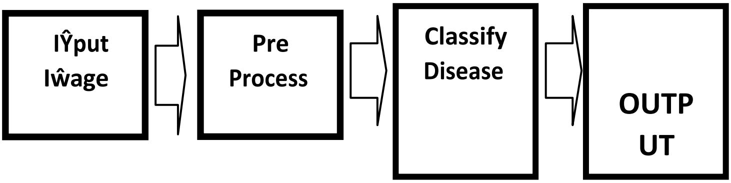


Figure5.System Algorithm Design

- *Input Image.* The papaya image to be tested.
- *Pre Process.* This process extracts the features of the input image by converting it into binary type then threshold to minimize the variance of the black and white pixels. It collects any non-zero values and feed it to the neural network for computation and returns a value.
- *Classify Disease.* It classifies the returned value by comparing it to the dataset value stored in the neural network database to output the result.
- *Output.* Display the output of the system which is either *non defective*, *defective – Anthracnose, defective – Ringspot,* or *defective – unclassified.*

C. Development and Training

Development stage covers all the activities undertaken from the time at which a potential requirement is identified until the resulting system is fully implemented. The development of Papaya Defect Classification System includes two phases: Training and Coding.

- Training

The feedforward neural network was trained using supervised learning Using the feedforward neural network algorithm, the network initializes the input weights and computes for the output weights which are the base output for classifying the disease during implementation. The neural network training phase is comprised of three steps:

- *Create Training Database.* A function that will store the image value of the loaded images. The database was initialized with two types of papaya images, non-defective and defective (diseased) papaya images. There were92 Anthracnose and 40 Ringspot papaya images loaded in the database. These images are used as the training dataset for the network and basis for classification.
- *Initialize Network.* This step sets up the three layered neural network model which is; composed of 132 input from the dataset, weights (number of weights depends on the input), and 1 output neuron. A simple neural network was created with a ***tansig*** transfer function as the hidden layer neuron and a ***purelin*** transfer function as the output neuron. The training dataset was used to train "classification engine" for recognition purpose. In this part, feed-forward backpropagation Neural Network was used.
- *Train Network.*There are several training algorithms for the Neural Network model in MATLAB which have a variety of different computation and storage. The training vector and testing vector from the database are used to match the input accepted by the neural network. The input or the training vectors are also used to train the network, while the testing vectors or the targets are used to evaluate the performance of the network. The researchers used Scale Conjugate Gradient Backpropagation (SCG) algorithm in the network. SCG is a second order Conjugate Gradient Algorithm which helps reduce target functions of several variables. It avoids time consuming line-search per iteration using a step size mechanism; hence it is faster than other second order algorithms. SCG algorithm, or “trainscg” in MATLAB, can train any network as long as its weight, net input, and transfer functions have derivative functions. Training only stops when it encounters any of these conditions:

  a. The maximum number of epochs (repetitions) is reached.

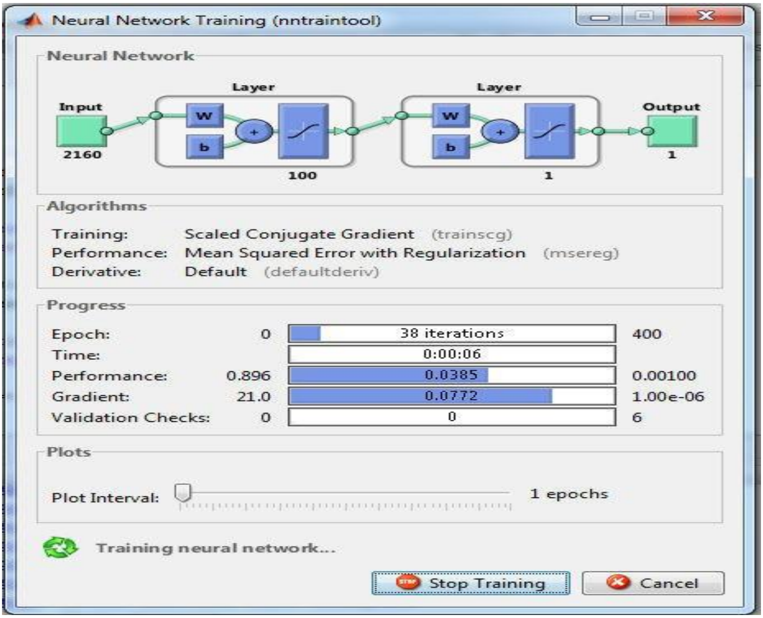
  b. The maximum amount of time is exceeded

  c. Performance is minimized to the goal.

  d. The performance gradient falls below “min_grad”.

e. Validation performance has increased more than max_fail times since the last time it decreased (when using validation)

Figure 6 shows the training process of the neural network.



Algorithms

Training: Scaled Conjugate Gradient (trainscg)
Performance: Mean Squared Error with Regularization (msereg)
Derivative: Default (defaultderiv)

Progress

```
Epoch:               0      38 iterations   400
Time:                       0:00:06
Performance:     0.896      0.0385          0.00100
Gradient:         21.0      0.0772          1.00e-06
Validation Checks:   0      0               6
```

Plots

Plot Interval: 1 epochs

Training neural network...

Figure 6. Neural Network Training Process

- Coding

Functions to read, pre-process and classify the image were developed using the programming environment of MATLAB. The Image processing functions included (1) binarization where the input image was converted to binary image, (2) Conversion to grayscale image, (3)Conversion to binary image, and (4) Resizing, where the binarized image wasmagnified 5 times. Resize function is important when the size of the character is not standard, especially in the preprocessing phase. The Classification function which test the pre-processed input data with the feedforward neural network but very low accuracy level for the unclassified category which means that more unclassified papayas were falsely classified. Some papaya images that

defective – Ringspot, or defective – unclassified.

D. Testing

System testing is a series of tests which primary purpose is to verify that system elements have been properly integrated and the functions specified are properly performed. During the system testing, the second set of approved images of healthy, anthracnose-infected and ringspot-infected papaya were tested and analyzed. Test images were loaded into the system for classification. Each test result was tabulated to evaluate and refine the accuracy of the system.

VIII. RESULTS AND DISCUSSIONS

A Papaya Defect Classification System was developed using MATLAB. Feedforward neural network was used to detect and classify papaya diseases using scale conjugate gradient algorithm. Trained dataset images were used as basis for disease classification of papaya. The neural network detects and classifies the disease by (1) computing the weighted sum of inputs to the neuron, (2) add the bias to the sum, (3) feed the sum as an input to the activation function of the neuron and (4) output of the activation function defined to be the output of the neuron that was represented with the formula, **output= f (∑weight*input + bias)**. The generated output weight was compared to the output weight stored in the network database. If the generated output weight is equivalent to the output weight from any of the two diseases stored in the neural network database, then the system outputs the corresponding classification of the disease.

A selected set of pre-classified images were loaded to the neural network for training. The system was trained to classify two papaya diseases, namely anthracnose and ringspot. The system cannot classify anthracnose disease on its later stage. It is also incapable of detecting and classifying ringspot on its early stage. The presence of other diseases or materials on the image results to false classification. Image lightning and brightness are also factors that affect the detection of disease attributes. The system can identify healthy papaya if it was totally blemish free.

Trial and error method was used to adjust the system's performance. During the first cycle of tests, a set of 44 papaya images were used. This included 18 healthy, 4 unclassified, 12 with Anthracnose and 10 Ringspot Virus. Result of the tests is shown in Table 2. Out of the 18 healthy input images 16 were correctly classified resulting to an 88% accuracy rating. Classification of Anthracnose and PRSV defects was 100% while unclassified defects only got an accuracy rating of 25%. And outputs one of the following: *non defective, defective –*

Cycle 1			
	Total	**Correct**	**Accuracy (%)**
Healthy	18	16	88%
Unclassified	4	1	25%
Anthracnose	10	10	100%
PRSV	12	12	100%

On the second cycle, 100 papaya images were tested consisting 30 healthy, 16 unclassified, 36 with Anthracnose and 18 Ringspot Virus. Table 3 shows that 28 of the 30 healthy images were correctly recognized, 13 of 16 unclassified images were wrongly classified resulting to a very low accuracy rating of 18%. 31 out of 36 with Anthracnose were properly classified giving an 86% performance rating while PRSV had an accuracy rating of 61%.

Cycle 2			
	Total	**Correct**	**Accuracy (%)**
Healthy	30	28	88%
Unclassified	16	3	18%
Anthracnose	36	31	86%
PRSV	18	11	61%

Table 3. Cycle 2 Test Results

For the third cycle, 108 papaya images were tested. The set of images consisted of 34 healthy, 17 unclassified, 39 with Anthracnose and 18 Ringspot Virus. The performance of the system in classifying defects is revealed in Table 4.

Table 4. Cycle 3 Test Results

Cycle 3			
	Total	**Correct**	**Accuracy (%)**
Healthy	34	31	91%
Unclassified	17	14	82%
Anthracnose	39	35	89%
PRSV	18	13	72%

The first two test results were used as basis for adjusting the parameters of the system to increase the performance level of the defect classifier. The output weight iteration of healthy and unclassified characteristics were fine tuned which increased the accuracy of the system on third test cycle. As a result, healthy papaya accuracy rating increased to 91%, unclassified disease accuracy level increased to 82%, anthracnose test accuracy rate rose to 89% and PRSV increased to 72% .

Only three cycles of testing were made to verify the accuracy of the system. In the first two testing cycles, high classification rates were observed for some categories captured the early stage anthracnose were not detected because of unclear feature that cannot be extracted through thresholding. The presence of unnecessary materials in the images, like moisture and other particles also affected the feature extraction process resulting to misclassification of the input images. Very dark green papaya surface was erroneously recognized as anthracnose because its color values were almost similar to anthracnose color characteristic.

The existence of overlapping defect classes, which should be recognized as unclassified, was falsely classified to the closest matched disease. Results also showed that in comparison with other error percentages ringspot performance level was higher. This is attributed to the imbalance of ANN training images. The researchers used fewer images of ringspot in training the network than the other classes.

IX. CONCLUSIONS AND RECOMMENDATIONS

This study addresses the inconsistency of manual quality assurance process in detecting and classifying defects of agricultural products, specifically papaya fruits. Based on the results of the study, the Papaya Defect Classifier was able to perform the classification of the papaya diseases, specifically anthracnose and ringspot diseases in a consistent and non-intrusive manner.

Using the feedforward neural network, the papaya surface defects were efficiently classified following supervised learning rule. Results showed that visual inspection system could effectively classify papaya diseases through normalizing the set parameters on the neural network. Thus it has been proven that through computer vision or image processing, artificial neural networks could be an effective medium to strengthen quality inspection mechanisms for agricultural products like the solo papaya variety fruits.

Based on the results of the study, the following recommendations are made:

- The use of Feed forward Neural Network will be more effective in classifying diseases if sufficient images are trained into the neural network.
- It is necessary to add a function that checks the type of the input image before loading it up for classification since the system does not classify a grayscale input image.
- To avoid false classification results, it is recommended that image to be tested must be cleaned-up first by removing unnecessary materials before image capture for a better input image quality.
- A crop functionality is advised to be implemented to allow selection a particular area of the papaya surface in cases when an input papaya image contains two or more diseases.
- It is also recommended that additional classes of papaya defects and diseases be included as classification parameters to further improve the system and reduce false classification of surface defects.
- For future implementation of the system, this study recommends to integrate a data acquisition hardware, like a high definition camera to increase the pixel per inch quality of images, for better image processing and for real time processing and classification.
- Lastly, the researchers recommend to replicate the study to further

a. REFERENCES

Arivazhagan, S. A. S., NewlinShebiah, , R. andVarthini, S. V.(2013) Detection of unhealthy region of plant leaves and classi_cation of plant leaf diseases using texture features. AgriculturaL Engineering International:CIGR Journal. Vol. 15, No.1.pages 211-{217. Retrieved from http://www.cigrjournal.org

.

Elbehiery, H., Hefnawy, A. and Elewa, M. (2007.)"Surface Defects Detection for Ceramic Tiles Using Image Processing and Morphological Techniques", Proceedings of World Academy of Science, Engineering and Technology, vol 5, pp 158-162, ISSN 1307-6884

Gross Dick, (2003). Papaya": A tantalising taste of the Tropics. Maricopa County Master Gardener Volunteer information, University of Arizona Cooperative Extension. Retrieved August 5, 2011, from http://www.arpapress.com/Volumes/Vol5Issue3/IJRRAS_5_3_15.pdf

Matzkanin George, (2006). Selecting a Nondestructive Testing Method: Visual Inspection. Retrieved August 10, 2011 from :http://www.aws.org/itrends/2007/04/it200704/it0407-15.pdf

Majidi B. and Moshiri, B. (2003).. Industrial Assessment of Horticultural Products' QualityUsing Image Data Fusion. Retrieved August 12, 2011from http://isif.org/fusion/proceedings/fusion03CD/regular/r179.pdf.

McGregor, K. and McGregor, A. (2009). The market for papaya from Fiji and other Pacific Islands – Japan Study. Retrieved August 1, 2011 from http://old.asean.or.jp/eng/trade/event/guide/2.pdf

Mossler, Mark and Crane, Jonathan(2002) Florida Crop/Pest Management Profile: Papaya. Institute of Food and Agricultural Sciences, University of Florida.Retrieved July 16, 2011 from http://edis.ifas.ufl.edu/pdffiles/PI/PI05300.pdf.

Moncada, F.M. and Mariano, V.Y. (2008). Grain Classification and Grading Based on Fourier Descriptor. Retrieved June 14, 2011 from http://www.ics.uplb.edu.ph/node/273

Narendra V G, and Hareesh K S(2010). Prospects of Computer Vision Automated Grading and Sorting Systems in Agricultural and Food Products for Quality Evaluation.International Journal of Computer Applications (0975 – 8887) Volume 1 – No. 4.

Pernezny, K. and Litz, R. E. (1993).Some Common Diseases of Papaya in Florida. Institute of Food and Agricultural Sciences (IFAS) University of Florida. **R**etrieved July 14, 2011 from http://university.uog.edu/cals/

Peña, Jorge E. and Johnson, Freddie.(2006) Insect Management in Papaya. Institute of Food and Agricultural Sciences, University of Florida.Retrieved July 14, 2011 fromhttp://edis.ifas.ufl.edu/IG074.

Pitas, Ioannis (2000) Digital Image Processing Algorithms and Applications. Wiley and Sons, Inc.ISBN 0-471-377739-2.

Prasad Babu, M.S. and SrinivasaRao, B. (n.d.) . “Leaves Recognition using Back propagation Neural Network- Advice for pest & disease control on crops”. Retrieved from: http://www.bharathgramvikas.net/agriculture/web/leafrecgnition.pdf

Riyadi, S.,Ashrani A. Abd. Rahni, Mohd., Mustafa, M. and Hussain, A (2007). Shape Characteristics Analysis for Papaya Size Classification. The 5th Student Conference on Research and Development –SCOReD 2007, 11 -12 December 2007, Malaysia. Retrieved June 14, 2011 from **riyadi**.staff.umy.ac.id/files/2010/07/Slamet-SCOReD07-pdfexpress.pdf

Rokunuzzaman, Md., Akhter, S., and Afzulpurkar, N. V. (2007). Automatic Defect Sorter for Tomatoes Using Machine Vision. Retrieved August 8, 2011, from www.aciar2005.ait.ac.th/CD/F-104.pdf

Sannakki,S., Rajpurohit, V, Nargund, V, Kumar, A and Yallur, P.(2011) Leaf disease grading by machine vision and fuzzy logic. International Journal of Computer Technology and Applications, 2:1709-1716, Retrieved from http://www.ijcta.com/documents/volumes/vol2issue5/ijcta2011020576.pdf

Stice, Kyle, McGregor, Andrew, Kumar, Sant and Prasad, Vinesh(2009). The market for papaya from Fiji and other Pacific Islands – New Zealand Study. Retrieved August 13, 2011 fromhttp://www.spc.int/lrd/index.php?op- tion=com_docman&amp;task=doc_download&amp;gid=51&amp;Itemid=254

Texas Plant Disease Handbook: Papaya. A&M University - Department of Plant Pathology & Microbiology. Retrieved January 15, 2011, from http://plantdiseasehandbook.tamu.edu/fruit-crops/papaya/

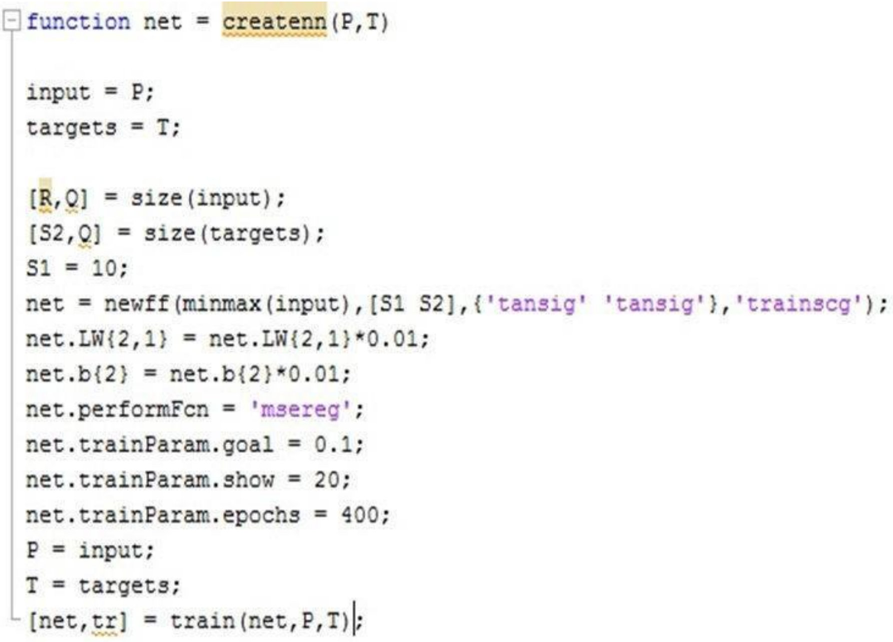
Unay D. and Gosselin, B. (2006). Automatic defect segmentation of Jonagold apples on multi-spectral images. Retrieved January 5, 2011 from http://staff.eng.bahcesehir.edu.tr/~unay/Unay_DefectSegmentation_PBT06.pdf

Valliammal N. and Geethalakshmi S.N. (2012). Plant Leaf Segmentation Using Non Linear K means Clustering. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 1, pp. 212-218. ISSN

Zhou, Lili, Paull, Robert E. and Jung Chen, Nancy. (n.d.)Papaya. Department of Tropical Plant and Soil Sciences, University of Hawaii at Manao, Honolulu, HI. Retrieved July 14, 2011 from http://www.ba.ars.usda.gov/hb66/101papaya.pdf.

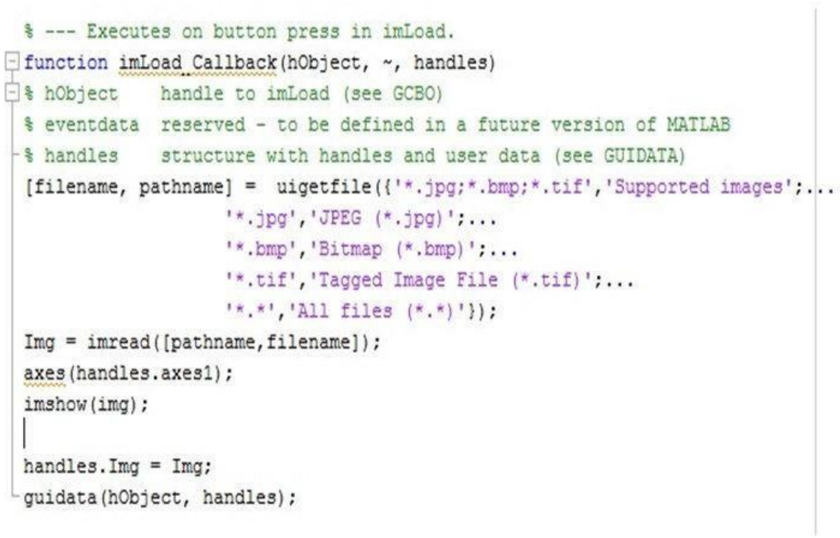
b. RELEVANT SOURCE CODES

- *Create Network*

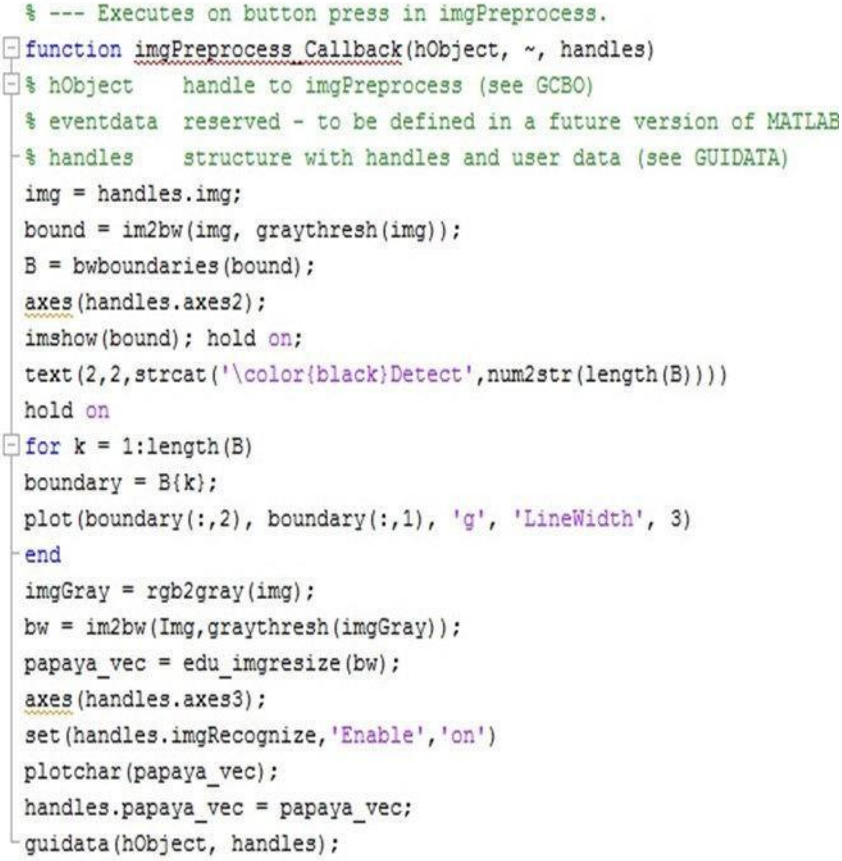


```
function net = createnn(P,T)

input = P;
targets = T;

[R,Q] = size(input);
[S2,Q] = size(targets);
S1 = 10;
net = newff(minmax(input),[S1 S2],{'tansig' 'tansig'},'trainscg');
net.LW{2,1} = net.LW{2,1}*0.01;
net.b{2} = net.b{2}*0.01;
net.performFcn = 'msereg';
net.trainParam.goal = 0.1;
net.trainParam.show = 20;
net.trainParam.epochs = 400;
P = input;
T = targets;
[net,tr] = train(net,P,T);
```

- *Pre-process Input Image*

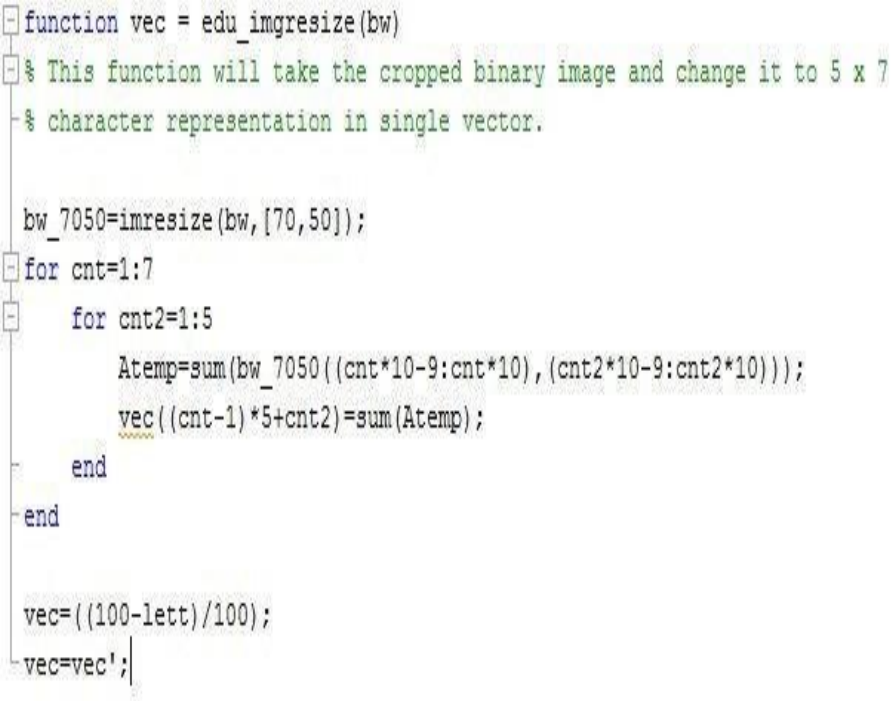


```
% --- Executes on button press in imLoad.
function imLoad_Callback(hObject, ~, handles)
% hObject    handle to imLoad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename, pathname] =  uigetfile({'*.jpg;*.bmp;*.tif','Supported images';...
                   '*.jpg','JPEG (*.jpg)';...
                   '*.bmp','Bitmap (*.bmp)';...
                   '*.tif','Tagged Image File (*.tif)';...
                   '*.*','All files (*.*)'});
Img = imread([pathname,filename]);
axes(handles.axes1);
imshow(img);

handles.Img = Img;
guidata(hObject, handles);
```



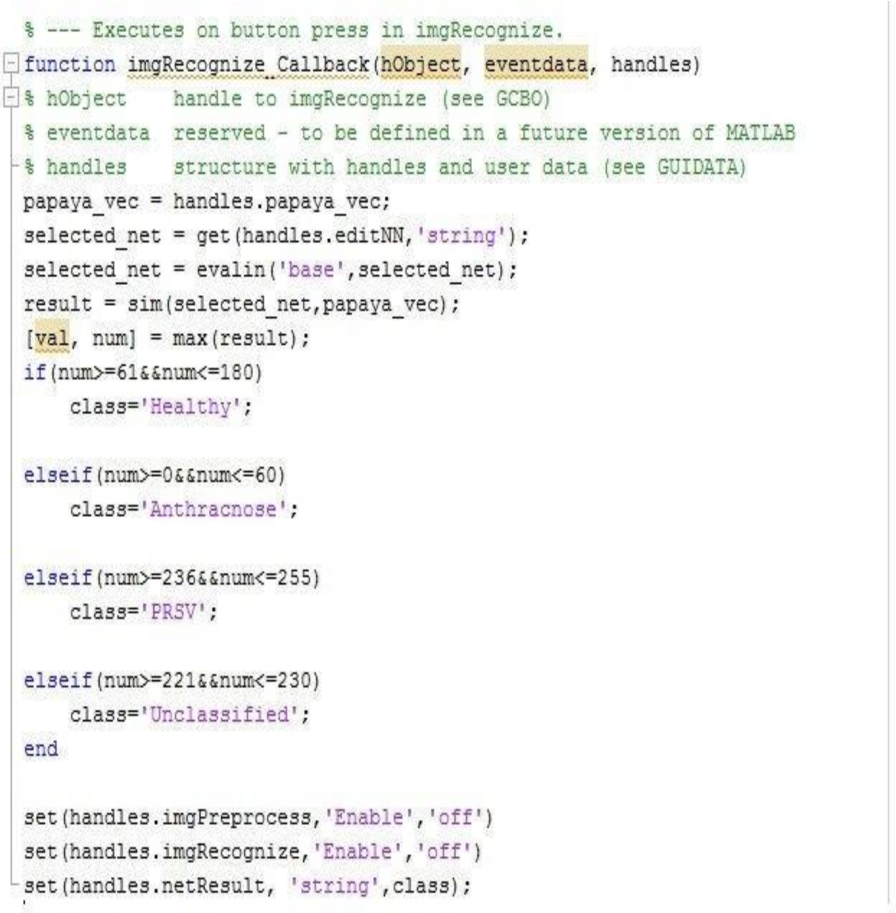
```
% --- Executes on button press in imgPreprocess.
function imgPreprocess_Callback(hObject, ~, handles)
% hObject    handle to imgPreprocess (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
img = handles.img;
bound = im2bw(img, graythresh(img));
B = bwboundaries(bound);
axes(handles.axes2);
imshow(bound); hold on;
text(2,2,strcat('\color{black}Detect',num2str(length(B))))
hold on
for k = 1:length(B)
boundary = B{k};
plot(boundary(:,2), boundary(:,1), 'g', 'LineWidth', 3)
end
imgGray = rgb2gray(img);
bw = im2bw(Img,graythresh(imgGray));
papaya_vec = edu_imgresize(bw);
axes(handles.axes3);
set(handles.imgRecognize,'Enable','on')
plotchar(papaya_vec);
handles.papaya_vec = papaya_vec;
guidata(hObject, handles);
```

- *Resize Image*



```
function vec = edu_imgresize(bw)
% This function will take the cropped binary image and change it to 5 x 7
% character representation in single vector.

bw_7050=imresize(bw,[70,50]);
for cnt=1:7
    for cnt2=1:5
        Atemp=sum(bw_7050((cnt*10-9:cnt*10),(cnt2*10-9:cnt2*10)));
        vec((cnt-1)*5+cnt2)=sum(Atemp);
    end
end

vec=((100-lett)/100);
vec=vec';
```

RecognizeDefect Class



```
% --- Executes on button press in imgRecognize.
function imgRecognize_Callback(hObject, eventdata, handles)
% hObject    handle to imgRecognize (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
papaya_vec = handles.papaya_vec;
selected_net = get(handles.editNN,'string');
selected_net = evalin('base',selected_net);
result = sim(selected_net,papaya_vec);
[val, num] = max(result);
if(num>=61&&num<=180)
    class='Healthy';

elseif(num>=0&&num<=60)
    class='Anthracnose';

elseif(num>=236&&num<=255)
    class='PRSV';

elseif(num>=221&&num<=230)
    class='Unclassified';
end

set(handles.imgPreprocess,'Enable','off')
set(handles.imgRecognize,'Enable','off')
set(handles.netResult, 'string',class);
```

SOLID WASTE RECOGNITION USING MAXIMUM CROSS CORRELATION

Tamara Cher R. Mercado, Hazel Ann N. Martinez, Iñaki O. Narciso, Hazel Joy B. Sanchez

Abstract

The continuous consumption of resources tends to produce more wastes as different countries face a common environmental problem: the enormous growth in quantity of solid waste materials. With plastics and other wastes employed in human lives and the increasing pressure being placed on landfills, the need for segregation and recycling has assumed increasing importance today.

In this paper, the researcher applied the concept of digital image processing and object-recognition in the field of solid waste management by building a system which detects and recognizes solid non-deformed non-biodegradable objects according to object classes: bottles, batteries, bottle caps, and cup noodle plastics. An experimental research approach was employed which was conducted in five phases: image acquisition, system development, testing and experimentation, interpretation of the results, and verification and validation. Feature matching and maximum cross correlation were the algorithms employed in the development of the system.

For each of the four object classes, 500 test images were used which consists of 400 positive test images and 100 negative images. Positive test images contained objects of interest from the four valid object classes, while 100 negative images contained random scenes and do not have objects coming from the four valid object classes.

After executing the experiments on 400 positive test images, the results obtained were as follows: the system yielded 88% detection rate on the bottle class, 87% on the battery class, 85% on the bottle cap class, and 91% on the cup noodle class; while experiments on the negative dataset of images yielded 10% false acceptance rate.

The result of this study showed that feature-based matching and maximum cross correlation can be used to recognize objects in a digital image. The high percentage of object match indicates that the system developed can be used as a classification tool in segregating solid wastes.