Component Testing for VsImaging Library Using Pixel Comparison Technique

M.Zulfahmi Toh, Abbas Saliimi Lokman and K.S.K Ibrahim

Faculty of Computer Systems & Software Engineering,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak
26300, Gambang, Kuantan, Pahang
Email: {zulfahmi,abbas,saimah}@ump.edu.my

Abstract. Testing the image similarity between two images is a non-trivial task. Image is not a quantitative data input and output. Image contains several complex properties that can be evaluated. In the present paper, properties like height, length and pixel between the two image are compare to get the similarly of the component testing from the VSImaging library image output with the expected image from the library to validate the output image are match the criteria of the expected output image. Furthermore these paper will explain the automatic unit testing of the VSImaging component will be conducted.

Keywords: Component Testing, Image Testing, Unit Test, Image Pixel Evaluation

1 Introduction

Component testing is a process of verifying results produced by a software component. Component in this context is a collection of processing algorithm that is used to process a specific input in order to get a specific output or result. Component testing is essential when programmers or system developers need to validate their algorithm in order to proof its correctness in term of producing an expected result. In this paper, we discussed a technique used to test a commercial component named VSDP in attempt to validate its produced results. The testing's result then was used to produce a Software Test Result (STR) report for client validation purpose.

Vision System Development Platform (VSDP) is a Commercial off the Shelf (COTs) component or a library that consists of intelligent processing algorithm such as neural network, fuzzy logic and so on. Developed by Center of Artificial Intelligent and Robotic (CAIRO), Universiti Teknologi Malaysia (UTM), VSDP can be used by any system developer who requires its processing power. Several sample systems that uses VSDP are Vision Plate Recognition System, Wood Recognition System and Semi Conductor Inspection System.

There are a lot of library components within VSDP (VsCore, VsImaging, VsMath and so on). For our proposed testing technique, we will focus on VsImaging

component that contains a set of functions for computer vision including almost all common image processing algorithms such as Color Filter, HSL Filter, YCbcCrFilter and so on. The proposed technique will be used to test images that are the results of all 12 main functions in VsImaging. Detail discussion will be on the Pixel Comparison Technique section of this paper.

2 Literature

2.1 Unit Testing

In stated in IEEE standard [1], Unit testing is to verify the individual source code or individual function are working properly in the computer programming sub-function or component. Unit testing is the important approach to enables the high quality in software development due to its efficiency [2].

Unit is the smallest piece of software. Unit testing is concerned with the low level structure of software program code and function. Beside, unit testing also is the software testing process of validating a smallest block of a software before proceed to integration and system testing. Unit testing will ensure software units are behaves exactly in the way it supposed to be. In the unit testing, the individual of source code, class and function or procedure are tested separately before integrated into module to test it between modules.

Objective of the unit testing is to separate and validate each individual part of the software unit, code, etc. Unit testing will help the software developer identify the error and bug as early in the development cycle thus can reduce the risk and effort in the system and integration testing level.

2.2 Component Based Software Testing

Component based software development main idea is to build new software product by reusing readily available parts rather than by developing erverthing from scratch [4]. The study had been proving that the software reuse has a higher profit in the investment than develop software from scratch. Component based software development can reduce the development time and cost. Even thought the component based software developments have advantages, it also has some drawback.

2.3 Component Based Test Strategy

Before VSImaging component testing begins, the component test activities and strategy plan shall be specified. The component test strategy shall specify the techniques to be employed in the design of test cases and the rationale for their choice [5]. The activities such as specify the criteria for test completion and the rationale for the implementation shall be address in the component based test strategies. The

component test strategy shall document the test process that will be used for component testing and the document must define the activities that need to be performed.

2.4 Black Box Testing (Functional Testing)

Functional testing is also known as black box testing. The software testers test the function in the software by entering the appropriate input and examining the output result. Functional testing normally will be applied in the acceptance test which ignores the internal mechanism of the system. Functional test is very useful to verify the software compliance against software's initial requirement specification and software's design document. Testers who test the software do not need knowledge of any specific programming language to execute the testing. The importance part to be remember is, the test need to be done from the user point of view.

In the component based testing, black box testing has a very important role to assure a component that can work property or not [6]. Before the tester can execute the component's black box testing. Tester need to require some information of the component, which can execute the test case. Components are normally an executable file or libraries that do not have any graphical interface which tester can give input for testing.

In VSDP VSImaging component test, tester is providing with the user manual which can be referring when executing the test case. The VSImaging user manual provide the information about all the function structure but not the inner code of the working function, tester had been provided with the recommended input to be used to execute the test case. This recommended input such as numeric value and overlay image need to be follow because these stated input values already apply to the expected images that will be used in the testing activities.

2.5 Image Pixel

Images in the digital world are a computer file that contains graphical information. Pixel or picture element is the basic building block of all digital images or a single point in a digital image. The pixel is an image are transformed into a color space and are indexed into a lookup table. The indexed values will be the threshold value to be compared while detecting or comparing the skin pixel [8]. Pixel exists in the reference grid form inside the picture. More grid exist the more quality picture will be. The use of reference grid derived from aerial photography for a pixel by pixel comparison with classified images can yield conservative estimates of the classification accuracy [7].

3 Pixel Comparison Technique

For the testing of VsImaging component, we developed a new algorithm called VsImaging Pixel Comparison Technique or VSIPC. Inputs for VSIPC are two

distinctive images which are Expected Image and Actual Image. Expected image are images gathered from VsImaging library (created using other professional image editing tool) and actual image are output images from VsImaging component processes. Detail on this will be discussed later in Experimental Setup section of this paper. VSIPC algorithm steps are defined as follows:

1. Validate both images' width and height. If not equal, go to step 11
2. Make new variable i as equal to 0
3. If variable i is less than image width, go to step 6 OR if variable i is more or equal than image width, prepare a success message ("Passed") then go to step 11
4. Make new variable j as equal to 0
5. If variable j is less than image height, go to step 8 OR if variable j is more or equal than image height, go to step 10
6. Get the color value of an expected and actual image's pixel
7. Compare the color value between expected and actual image's pixel
8. If the color value did not matched, prepare an error message ("Failed") then go to step 11 OR if the color value is matched, go to step 9
9. Add the value 1 to variable j then go to step 5
10. Add the value 1 to variable i then go to step 3
11. Output result and end of process

In the first step, validation process will be done to make sure that both images (expected and actual images) are same in regard to its width, height and size. If both images are validated, actual image will go to the looping process to determine the similarity on each pixel's color value. Two nested looping process will be done in the next step (step 2 until step 10). The main looping if for the width and the sub looping is for the height. Both maximum pixel of image's width and height will be defined as the maximum value for each looping respectively. Through this process, the image's pixel will be analysed one-by-one vertically from the first pixel located at top-left corner of the image. Step 9 is where the pixel's color value will be compared and an error message is prepared if the color value is not equal. Because the accuracy is important in this test, one unmatched pixel is equivalent to a 100% matching failure. As such, when step 7 encounters an unmatched result, algorithm will go straight to step 11 and print the error message thus ending the matching process.

Following is VSIPC algorithm written in Visual Studio C# Language:

```
Public static void AreEqual(Bitmap expected, Bitmap
actual)
  if(!ex[expected.Width.Equals(actual.Width)){
    HandleFail(*ImageAssert.AreEqual – Width not
    equal.", String.Empty, 1, 2);
  }
  if(!expected.Height.Equals(actual.Height)){
    HandleFail(*ImageAssert.AreEqual – Height not
    equal.",String.Empty, 1, 2);
  }
```

```
Boolean equal = true;
for (int i=0; i<expected.Width; i++){
  for (int j=0; j<expected.Height; j++){
    Color expectedBit = expected.GetPixel(i,j);
    Color actualBit = actual.GetPixel(i,j);

    if (!expectedBit.Equals(actualBit)){
      equal = false;
      HandleFail(*ImageAssert.AreEqual – Image Not
      Equal", String.Empty, i, j);
      continue;
    }
  }
}
```

4 Experimental Setup

4.1 Test Items

VSDP's VsImaging library has 12 main functions in total. Those functions are Color Filter, HSL Filter, YCbcCr Filter, Binarization, Morphology, Convolution, Edge Detectors, Noise Generation, Two Source Filters, Other, Transform and ImageExt. Within these 12 main functions, there are a total of 93 sub functions and because of its distinctive processing nature (each function runs different processes), all 93 functions are required to be test. Some of those 93 functions are: Blue Filter, Cyan Filter, Brightness Correction, Extract Cb Channel, Bayer Ordered Dithering, Top-Hat Operator, Matrix of Blur Filter, Canny Edge Detector, Additive Noise Filter, Add Pixel Values, Jitter Filter and Resize Nearest Neighbour.

4.2 Tool

Visual Studio 2008 Professional Edition is being used as a test tool. Visual Studio (VS) is identified to be suitable tool because of several reasons. That reasons being: 1) Its feature unit testing tools that able to call the method of a class and passing suitable parameter and data to verify that the returned value is what tester expects. 2) VS allow tester to manually code the unit test, thus making us able to incorporate VSIPC algorithm into the test. 3) VS provide a tester-friendly interface that is a working space for tester to create the test project and easily view what function need to be test. Following Fig. 1. shows the VS working space.

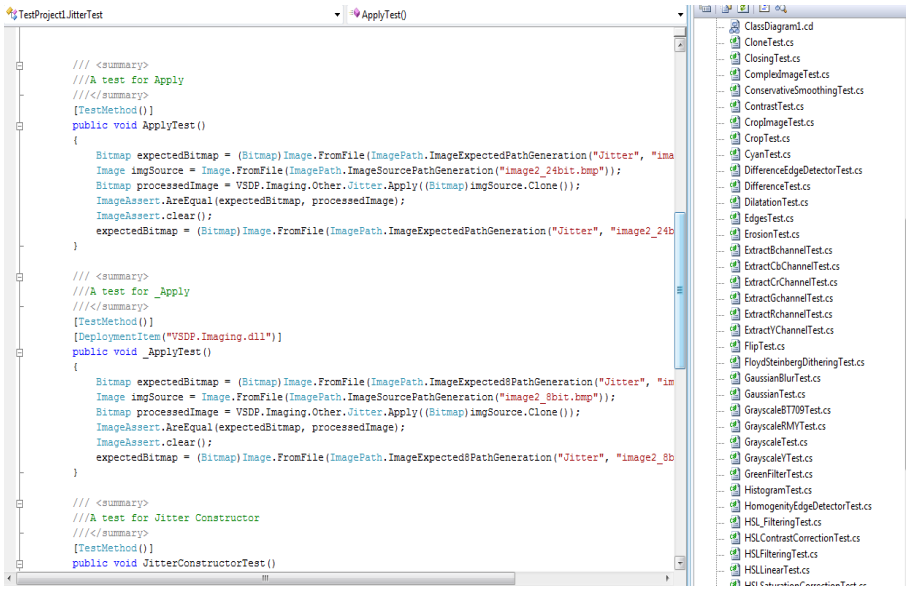

Fig. 1. Visual Studio 2008 Professional Edition working space.

4.3 Aims

There are two aims within this experiment. The first aim is to test all 93 functions in VsImaging by doing a comparison between two images (expected and actual image) in order to validate the correctness of the function's process. The second aim is to prove the effectiveness of VSIPC by analyzing the results produced by this test.

Comparing two images, Inputs for VSIPC are two distinctive images which are Expected and Actual Image. While expected image are images gathered from VsImaging library (created using other professional image editing tool), an actual image are output images from VsImaging component processes. As such, 93 VsImaging functions are being executed in order to get 93 actual images that will be used in this comparison process.

Proving the effectiveness of VSIPC, Two hypotheses are made in regard to this aims. First hypothesis is h1: The effectiveness of VSIPC is arguable if all test results are passed/positive. This is because VSIPC might be wrong in the sense that a 100 percent success rate is highly illogical. If an actual 100 percent success rate is achieved, VSIPC algorithm will be checked first before further analysis is done in order to confirm that the 100 percent success rate is justly because all 93 VsImaging functions are producing very accurate results. Second hypothesis is h2: The effectiveness of VSIPC is justified if test results are mixed (passed and failed). Mixed

test results can justify the effectiveness of VSIPC because it proved that VSIPC was able to detect the inaccuracy in results producing by some of VsImaging functions.

5 Result

Result of this test will be discussed in regard to previously discussed aims.

Comparing two images, Among all 93 VsImaging functions, only 76 are tested. This is because 17 functions are identified to be not ready for the test due to unavailability of the expected images (not yet prepared by the VSDP team). Therefore, the aim to test all 93 VsImaging functions is only 81.72% achieved.

Proving the effectiveness of VSIPC, Fig 2 show 76 VsImaging functions are tested. Among those functions, 71 have passed and 5 have failed the test. This is a 93.42% success rate. Based on this numbers, following conclusion can be made; h1: The effectiveness of VSIPC is arguable if all test results are passed/positive is not supported while h2: The effectiveness of VSIPC is justified if test results are mixed (passed and failed) is supported. As such, further analysis need not to be done because it is justified that VSIPC was able to detect the inaccuracy in results producing by some VsImaging functions. Although the percentage is small (6.58%), it is enough to say that VSIPC is effectively comparing two images that seems to be the same if only viewed using the naked eyes. 93.42% of success rate can be viewed as the accuracy of VsImaging in producing its results.

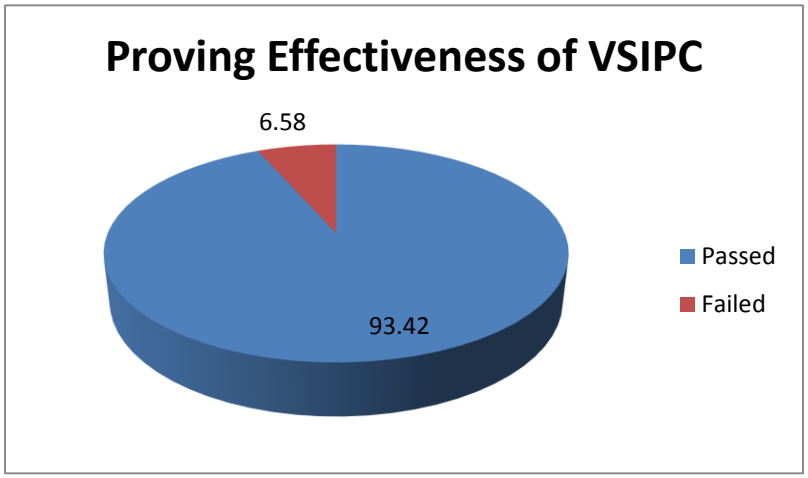

Fig.2: Effectiveness of the Algorithm

6 Conclusion

In this paper we have discussed a component based testing done to VSDP VsImaging Library using newly developed technique named VsImaging Pixel Comparison Technique or VSIPC. VSIPC is proofed to accurately measure the similarity between two images (expected and actual images) by comparing the color value of all pixels representing the images. Each pixel is tested using two nested looping processes that analyzed all the pixels one-by-one in vertical order from the first pixel located at top-left corner of the image. From the total of 93 functions in VsImaging, 71 functions have passed the test while only 5 functions are failed. Balance 17 functions are not tested because of the unavailability of the expected images to be tested. Results from the test have been used to prepare the STR document for client validation and approval. Based on the STR, client can improved on failed functions in order to get a success result later in the near future.

7 Acknowledgements

We would like to thank Universiti Malaysia Pahang (UMP) for the support grant for this research no. RDU 130616: Entitle Investigation a Genetic Algorithm Based Strategy for Sequence Covering Array Construction

References

1. IEEE Std. 610.12-1990. Standard Glossary of Software Engineering Terminology, IEEE, 1990.
2. Bin Xu, Toward Efficient Calloborative Component-Based Software Unit Testing Via Extend E-CARGO Model- Based Activity Dependence Identification.Proceedings of the 2009 International Symposium on Inteligent Ubiquitous Computing and Education,pp.172-175
3. Ravinder Kumar, Karambir Singh, A Literature Survey on Black Box Testing in Component Based Software Engineering.
4. Hans – Gerhard Gross, "Component Based Software testing with UML", Springer, Kaiserslautern, Germany, 2005.
5. "Standard for Software Component Testing", British Computer Specialist Interest Group in Software Testing, 2001.
6. Furqan Nasser,Shafiq ur Rehman ,Khalid Hussain, "Using Meta-Data Technique for Component Based Black Box Testing", 2010 6th International Conference on Emerging Technologies, IEEE, 2010, pp 276-28.
7. D.L. Verbyla, T.O Hammond, "Conservative bias in classification accuracy assessment due to pixel by pixel comparison of classified images with reference grids", INT.J.Remote Sensing, 1995, vol. 16, No. 3, P 581-587.
8. S.H.D.S Jitvinder, S.S.S. Ranjit, K.C.Lim,A.J.Salim, "Image Pixel Comparison Using Block Based Positioning Subtraction Technique for Motion Estimation." 2011 Fifth Asia Modeling Symposium.