Matlab Code For Identification Of Graphics Objects In Aircraft Displays

Lakshmi Devi P
Assistant Professor
Channabasaveshwara Institute of Technology, Gubbi
Tumkur, Karnataka, India. *E-mail: lakshmi21devip@gmail.com*

Abstract— This paper is aimed at understanding, utilizing and improving the existing system and automating the graphics testing process. The paper involves developing an automation design for automating the graphics testing process involved in software testing of aircraft displays. The paper comes under software development; there are so many steps in software development like Requirement analysis, Design, Implementation, Testing and evolution. So testing is the one of the step which comes under software development. We are designing an automation tool in graphics testing. Graphic testing tests the display devices. The main motivation of this project is to save the time because current approach involves user to read each and every message box which gets popped up while executing script. This approach involves a lot of manual effort and a person has to physically present and do the test execution and also there is possibility of mistakes when a non-trained person working on it. So to make initial test set up as fast as possible, with no errors and with no manual effort we are developing an automation tool. The tool includes techniques of image comparison, optical character recognition and template matching. Our automation should be able to handle all kinds of text and digits. Implementing a design which is able to recognize characters which pops up from window and takes a decision of pass/fail based on the recognized characters, while doing testing automatically, for that we use optical character recognition and template matching is mainly used for object recognition. Image comparison is used to capture an image and do the executions process automatically.

Keywords- Image comparison, Automation design, Optical character recognition, template matching.

*****

I. INTRODUCTION

The concept of this paper comes under software development; there are so many steps in software development like Requirement analysis, Design, Implementation, Testing and evolution. So testing is the one of the step which comes under software development. There are so many testing's like graphics testing, integration testing, Performance testing, System testing and etc. We are designing an automation tool in graphics testing. Graphic testing tests the display devices. This is a graphics test automation that aims at automating the test execution process of tests related to graphics. It enables the user to capture images for the respective test cases and facilitate the execution of the test scripts. Automated test suite reduces the need for manual testing. Graphical tests have to be repeated often during development cycles to ensure quality. Every time source code is modified software tests should be re-executed. For each release of the software it may be tested on all supported operating systems. Manually repeating these tests is costly and time consuming. The net effect of the benefits listed above is that software development will become more predictable and repeatable. Automated Software Testing saves Time, Money and increases accuracy. Once created, automated tests can be run over and over again at no additional cost and they are much faster than manual tests. Automated testing can reduce the time to run repetitive tests from days to hours. A time savings that translates directly into cost savings. Testing is the one of the step which comes under the software development life cycle. Testing on a hardware target is very costly and hardware is unavailable in early phases of life cycle, so where graphical images are supposed to be tested using a hardware device, graphical software testing methodology is being proposed. Software testing is an integral and important phase of the software development life cycle. This part of the process ensures that defects are recognized as soon as possible. There are so many testing's like graphics testing, integration testing, Performance testing, System testing and etc. The tool is a graphics test automation tool that aims at automating the test execution process of test scripts that are developed for testing the graphics images based on Software Requirements.

Graphics Testing is used to test Display Units (DU's) or any console where images are displayed. Graphical user interface testing is a testing in which testing framework generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the program is correct.

A. Problem Statement

In an aircraft there will be two display units. One is for Pilot and another one is for Co-pilot. This is mainly for safety purpose. The current graphics testing is an intensive graphical human-machine interface testing. To perform a successful graphical testing, tester (who does graphical testing) should be aware of all the terminology and details about the particular aircraft display before performing a graphical testing. If the tester performs an unsuccessful testing, so aircraft display unit is displaying something else instead of displaying correct data, then it might impact safety of aircraft and pilot. So tester should be very serious and should have knowledge about the testing.

B. Existing system

Human inspection has been employed for the purpose of graphics testing for many years. But human inspection is expensive and prone to error. The tests need to be executed multiple times to ensure correctness.

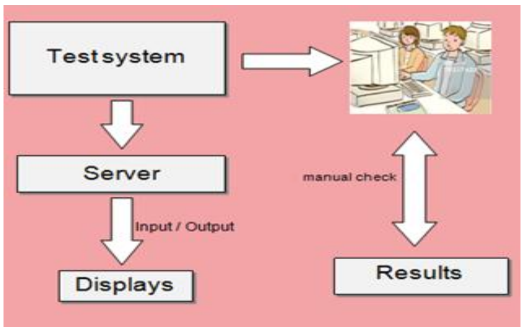


Figure1: Existing system operation.

C. Proposed system

Image Comparison techniques are used to compare the new images to the existing data set. A database is created with pre-existing/required images to compare the obtained image. Image comparison and OCR techniques can be employed to automate the process of Graphics testing. OCR techniques are used to capture the characters in image displayed. The automation test process in turns consists of two blocks. First block consists of techniques which we will be using and implementing in our paper like image comparison, optical character recognition and template matching. Second block is a message box which contains a message for automatic pass/fail statement. Our algorithms checks and writes the result instead of user manually to see the result. This is our novelty. So pass/fail asking window will not be displayed, they will be in fact, that is stored in summary file for further verification.

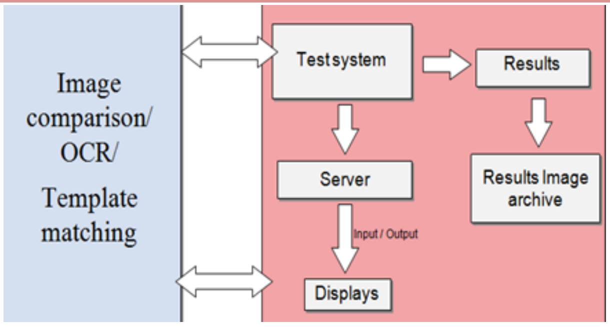


Figure2: Automation Proposed Design

D. Objective

- To automate complete graphics testing by using the techniques of
  - Image comparison
  - OCR and
  - Template matching.
- Tool to capture and populate reference image database with all data.
- Tool to run random suite of Graphics tests, compare image and click Pass/Fail.
- Involve Advanced Technique for inputs on improving the solution and algorithms.
- To eliminate a lot of manual effort and manual errors.
- To save huge amount of time.
- Implement character recognition techniques thereby automating more scenarios.
- To reduce the effort and space required for image database creation.
- Summary image archive files for offline comparison and records.

II. IMAGE COMPARISON

In this phase the image is captured from the display by running the script, bitmap and coordinates are fetched, new image is captured and the two images are compared. The resultant images are stored in the database for future comparison and reporting purpose. The execution stops automatically after all the test cases in the test script file are run.The captured images are compared against the pre-existing images in the database pixel by pixel. OpenCV libraries are used to efficiently compare the images. An error margin can be specified to account for minor errors. The algorithm converts the images into matrices and compares the matrices thus generated. After the error margin bits are accounted for, the algorithm decides if the comparison has passed or failed.

OpenCV libraries are used to efficiently compare the images. It contains a large collection of image processing functions, to solve a computational challenge. An error margin can be specified to account for minor errors. The algorithm converts the images into matrices and compares the matrices thus generated. After the error is accounted for, the algorithm decides if the comparison has passed or failed.

A. Image comparison algorithm:

Double cvNorm (const CvArr* **arr1**, const CvArr* **arr2**=NULL, int **norm_type**=CV_L2, const CvArr* **mask**=NULL)

The function above can be used to compute the norm of an array and also a variety of relative distance norms if two arrays are provided. There is an intuitive interpretation of the norms as a Euclidean distance in a space of dimension equal to the number of pixels in an image.

Description: The function slides through image, compares the overlapped patches of size against template using the specified method and stores the comparison results in result. Here are the formulae for the available comparison methods (denotes image, template, result). The summation is done over template and/or the image patch:

- method=CV_TM_SQDIFF

$$R[x,y]=\sum_{x',y'}(T(x',y')-I(x+x',y+y'))2$$

- method=CV_TM_SQDIFF_NORMED

$$R[x,y]=\frac{\sum_{x',y'}(T(x',y')-I(x+x',y+y'))2}{\sqrt{\sum_{x',y'}T(x',y')2*\sum_{x',y'}I(x+x',y+y')2}}$$

- method=CV_TM_CCORR

$$R[x,y]=\sum_{x',y'}(T(x',y')*I(x+x',y+y'))$$

- method=CV_TM_CCORR_NORMED

$$R[x,y]=\frac{\sum_{x',y'}(T(x',y')-I(x+x',y+y'))}{\sqrt{\sum_{x',y'}T(x',y')2*\sum_{x',y'}I(x+x',y+y')2}}$$

- method=CV_TM_CCOEFF

$$R[x,y]=\sum_{x',y'}(T(x',y')*I(x+x',y+y'))$$

Where

$$T'(x',y')=T(x',y')-1/(w*h)*\sum_{x'',y''}T(x'',y'')$$

$$I'(x+x',y+y')=I(x+x',y+y')-$$

$$1/(w*h)*\sum_{x'',y''}I(x+x'',y+y'')$$

- method=CV_TM_CCOEFF_NORMED

$$R[x,y]=\frac{\sum_{x',y'}(T(x',y')*I(x+x',y+y'))}{\sqrt{\sum_{x',y'}T(x',y')2*\sum_{x',y'}I(x+x',y+y')2}}$$

After the function finishes the comparison, the best matches can be found as global minimums (when CV_TM_SQDIFF was used) or maximums (when CV_TM_CCORR or CV_TM_CCOEFF was used) using the minMaxLoc() function. In case of a color image, template summation in the numerator and each sum in the denominator is done over all of the channels and separate mean values are used for each channel. That is, the function can take a colour template and a colour image. The result will still be a single-channel image, which is easier to analyse.

Where: *x* is the template gray level image.
x' is the average grey level in the template image.
y is the source image section.
y' is the average grey level in the source iHUmage.

There are two phase in image comparison:
-Image capturing phase
-Image execution phase.

B. Parameters:

1. Character pointer variable holding first Image path: Char*
2. Character pointer variable holding second Image path: Char*
3. Types of array normalization: integer values as below.

CV_C	1
CV_L1	2
CV_L2	4
CV_NORM_MASK	7
CV_RELATIVE	8
CV_DIFF	16
CV_MINMAX	32
CV_DIFF_C	(CV_DIFF | CV_C)
CV_DIFF_L1	(CV_DIFF | CV_L1)
CV_DIFF_L2	(CV_DIFF | CV_L2)
CV_RELATIVE_C	(CV_RELATIVE | CV_C)
CV_RELATIVE_L1	(CV_RELATIVE | CV_L1)
CV_RELATIVE_L2	(CV_RELATIVE | CV_L2)

Provide other than the above values to choose the default norm: "CV_DIFF_L1 (CV_DIFF | CV_L1)".

mask – it must have the same size as src1 and CV_8UC1 type.

Description: The function slides through image, compares the overlapped patches of size against template using the specified method and stores the comparison results in result.

FOR LETTER CROP:

```
%function lines%
%function letter_in_a_line
function [fl re space]=letter_crop(im_texto)
% Divide letters in lines
im_texto=clip(im_texto);
num_filas=size(im_texto,2);

%figure,imshow(im_texto);
%title('line sent in the function letter');
for s=1:num_filas
   s;
   sum_col = sum(im_texto(:,s));
   if sum_col==0
      k = 'true';
      nm=im_texto(:,1:s-1); % First letter matrix
      %figure,imshow(nm);
      %title('first letter in the function letter_in_a_line');
      %pause(1);
      rm=im_texto(:,s:end);% Remaining line matrix
      %figure,imshow(rm);
      %title('remaining letters in the function letter_in_a_line');
      %pause(1);
      fl = clip(nm);
      %pause(1);
      re=clip(rm);
      space = size(rm,2)-size(re,2);
      %*-*-*Uncomment lines below to see the result*-*-*-*-
            %subplot(2,1,1);imshow(fl);
            %subplot(2,1,2);imshow(re);
     break
   else
      fl=im_texto;%Only one line.
      re=[ ];
      space = 0;
   end
end
function img_out=clip(img_in)
[f c]=find(img_in);
img_out=img_in(min(f):max(f),min(c):max(c));
```

FOR LINES CROP:

```
%function lines%
%
function [fl re]=lines_crop(im_texto)
im_texto=clip(im_texto);
num_filas=size(im_texto,1);
for s=1:num_filas
   if sum(im_texto(s,:))==0
      nm=im_texto(1:s-1, :); % First line matrix
      %pause(1);
      rm=im_texto(s:end, :);% Remain line matrix
      %pause(1);
      fl = clip(nm);
      pause(1);
      re=clip(rm);
      %*-*-*Uncomment lines below to see the result*-*-*-*-
            %subplot(2,1,1);imshow(fl);
            %subplot(2,1,2);imshow(re);
      break
   else
      fl=im_texto;%Only one line.
      re=[ ];
   end
end
%subplot(3,1,1);imshow(im_texto);title('INPUT IMAGE')
%subplot(3,1,2);imshow(fl);title('FIRST LINE')
%subplot(3,1,3);imshow(re);title('REMAIN LINES')
function img_out=clip(img_in)
[f c]=find(img_in);
img_out=img_in(min(f):max(f),min(c):max(c));
```

MAIN FUNCTION:

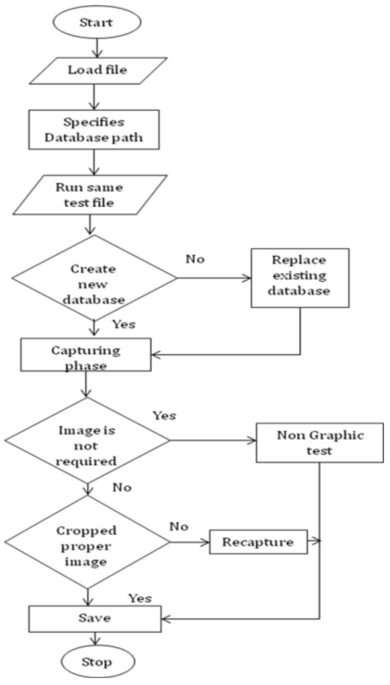
```
function[a, b]=letter(temp)
for i=1:67
d=dir(['let\',num2str(i),'\*.jpg']);
for j=1:length(d)
      img=imread(['let\',num2str(i),'\',d(j).name]);
   img=im2bw(img);
   cor(j)=corr2(img,temp);
end
fcor(i)=max(cor);
end
[a,b]=max(fcor);
a
end
```

TO READ A LETTER:

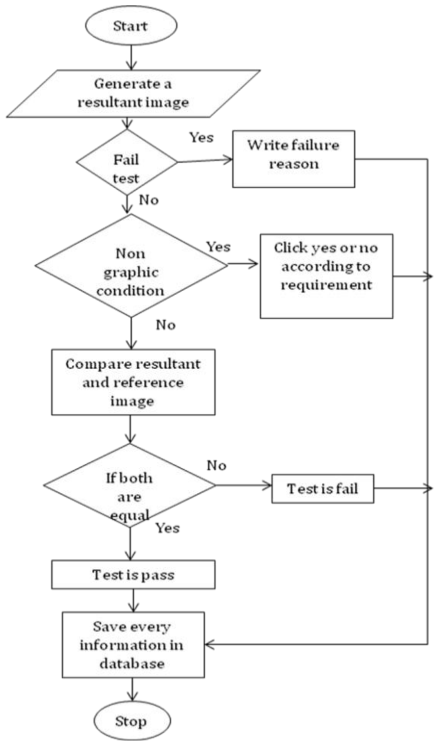
```
function [a,letter]=read_letter(imagn)
%Computes the correlation between template and input image
%and its output is a string containing the letter.
%Size of 'imagn' must be 42 x 24 pixels
%Example:
% imagn=imread('D.bmp');
% letter=read_letter(imagn)
% Load_Data_Name=['template_english.mat'];
% load(Load_Data_Name);
% comp=[];
%load template_english
% for n=1:62
%     sem=corr2(template_en{1,n},imagn);
%     comp=[comp sem];
% end
% vd=find(comp==max(comp));
[a,vd]=my_main(imagn);
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-
if vd>0
vd=vd(1);
if vd==1
   letter='1';
elseif vd==2
   letter='2';
elseif vd==3
   letter='3';
elseif vd==4
   letter='4';
elseif vd==5
   letter='5';
elseif vd==6
   letter='6';
elseif vd==7
   letter='7';
elseif vd==8
   letter='8';
elseif vd==9
   letter='9';
elseif vd==10
   letter='A';
elseif vd==11
   letter='a';
elseif vd==12
   letter='B';
elseif vd==13
   letter='b';
elseif vd==14
   letter='C';
elseif vd==15
   letter='c';
elseif vd==16
   letter='D';
elseif vd==17
   letter='d';
elseif vd==18
   letter='E';
elseif vd==19
   letter='e';
elseif vd==20
   letter='F';
elseif vd==21
   letter='f';
elseif vd==22
   letter='G';
elseif vd==23
   letter='g';
elseif vd==24
   letter='H';
elseif vd==25
   letter='h';
elseif vd==26
```

```
    letter='I';
    %*-*-*-*-%
elseif vd==27
    letter='i';
elseif vd==28
    letter='J';
elseif vd==29
    letter='j';
elseif vd==30
    letter='K';
elseif vd==31
    letter='k';
elseif vd==32
    letter='L';
elseif vd==33
    letter='l';
elseif vd==34
    letter='M';
elseif vd==35
    letter='m';
elseif vd==36
    letter='N';
elseif vd==37
    letter='n';
elseif vd==38
    letter='O';
elseif vd==39
    letter='o';
elseif vd==40
    letter='P';
elseif vd==41
    letter='p';
elseif vd==42
    letter='Q';
elseif vd==43
    letter='q';
elseif vd==44
    letter='R';
elseif vd==45
    letter='r';
elseif vd==46
    letter='S';
elseif vd==47
    letter='s';
elseif vd==48
    letter='T';
elseif vd==49
    letter='t';
elseif vd==50
    letter='U';
elseif vd==51
    letter='u';
elseif vd==52
    letter='V';
    % ####
elseif vd==53
    letter='v';
elseif vd==54
    letter='W';
elseif vd==55
    letter='w';
elseif vd==56
    letter='X';
elseif vd==57
    letter='x';
elseif vd==58
    letter='Y';
elseif vd==59
    letter='y';
elseif vd==60
    letter='Z';
elseif vd==61
    letter='z';
elseif vd==62
    letter='0';
elseif vd==63
    letter='%';
elseif vd==64
    letter=':';
elseif vd==65
    letter='-';
elseif vd==66
    letter='+';
elseif vd==67
    letter='.';
end
else
    letter='#';
end
```

III. FLOW CHART OF CAPTURING PHASE



IV. FLOW CHART OF EXECUTION PHASE


Start
Generate a resultant image
Fail test
Yes
Write failure reason
No
Non graphic condition
Yes
Click yes or no according to requirement
No
Compare resultant and reference image
If both are equal
No
Test is fail
Yes
Test is pass
Save every information in database
Stop

V. TEMPLATE MATCHING

Template matching is a technique in digital image processing for finding small parts of an image which match a template image. It is developed as an answer to object recognition. Template matching algorithm has the characteristics of high speed and real time.

- cvMatchTemplate(imgOriginal, imgTemplate,imgResult, CV_TM_CCORR_NORMED);

This instruction does all the sliding and correlation mathematics using imgOriginal (the source), imgTemplate (the template) and puts the correlation map into imgResult. The calculations used for determining the correlation map is the last parameter, CV_TM_CCORR_NORMED. To determine the maximum point in the correlation, we use another OpenCV function: cvMinMaxLoc. This function returns the minimum and maximum values and their locations.

Now max_loc holds the point we are interested in: the point with maximum correlation. We'll just put a rectangle there, and also print out the actual value of correlation. Correlation value shows that how much template image is matched to an original image, and finally displays the modified original image. Correlation is a measure of the degree to which two variables agree, not necessary in actual value but in general behavior. The two variables are the corresponding pixel values in two images, template and source.

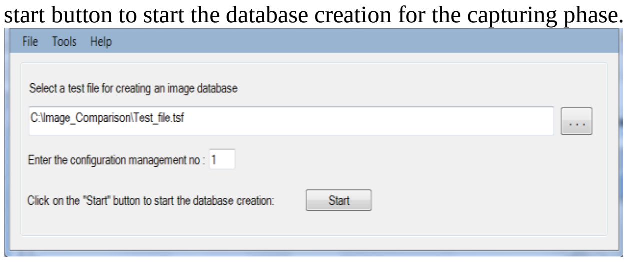
VI. OPTICAL CHARACTER RECOGNITION

One of the most frequent tasks in computer vision and image processing is the recognition of an image or an object in the image. Among these tasks, Optical character recognition (OCR) is a popular research topic. OCR aims at enabling computers to recognize optical symbols without human intervention. OCR is the mechanical or electronic conversion of scanned images of handwritten, typewritten or printed text into machine-encoded text. OCR is an image to text conversion. It is sometimes called as "Intelligent character reader" (ICR).

VII. RESUTS

A.Image comparison

It enables the user to capture images for the respective test cases and facilitate the execution of the test scripts. Tool asks to upload a test file as shown in the figure below and respective configuration management version of the test file is entered. After that click on the start button to start the database creation for the capturing phase.



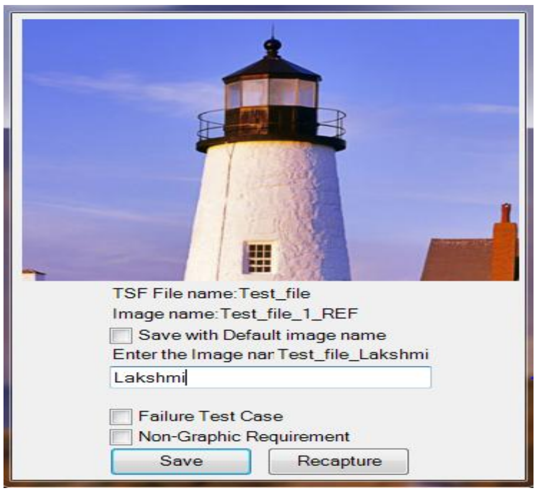
Select a test file for creating an image database

C:\Image_Comparison\Test_file.tsf

Enter the configuration management no : 1

Click on the "Start" button to start the database creation:

Figure3: Image comparison tool



TSF File name:Test_file

Image name:Test_file_1_REF

☐ Save with Default image name

Enter the Image nar Test_file_Lakshmi

Lakshmi

☐ Failure Test Case

☐ Non-Graphic Requirement

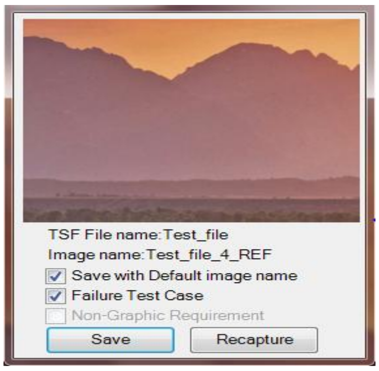
Figure4: Captured image preview

The above preview image shows the Test file name and tool asks whether it needs to save with default name or would like to enter any name. By default it saves Test_file_1_REF if it is first reference image or Test_file_2_REF if it is second reference image and so on. Entered image name saves like Test_file_name and specified_name, by specifying the x co-ordinate and y co-ordinate pixel values in the database.

B. Failure test case

After selecting failure test case condition, automatically non-graphic condition is disabled, because as explained in the image comparison part, for failure test condition there is an image but that is not the required image which is displaying on the aircraft display unit, and for Non-graphic condition, no need to select an image. So obviously for failure test case there will be some image to capture so automatically non-graphic condition is disabled as shown in the figure below.

In any of the condition if the captured image is not proper then it can be recaptured by selecting 'Recapture' option.



TSF File name:Test_file

Image name:Test_file_4_REF

☑ Save with Default image name

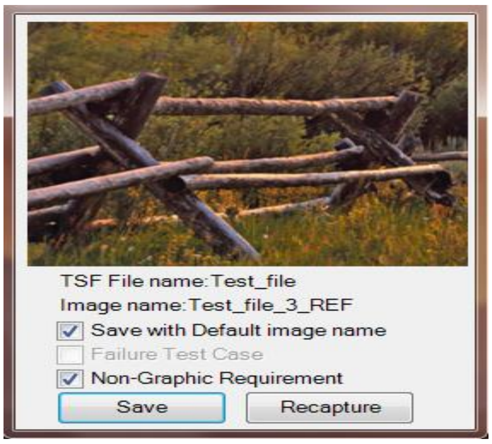
☑ Failure Test Case

☐ Non-Graphic Requirement

Figure5: Selecting Failure test case

C. Non-Graphic condition

After selecting the non-graphic condition, then automatically failure test case option is disabled. Because for non graphic condition no need to consider any image at all then there is no point in saying that this image is wrong image. So tool disables the failure test case option while performing non-graphic condition as shown in the figure below.



TSF File name:Test_file

Image name:Test_file_3_REF

☑ Save with Default image name

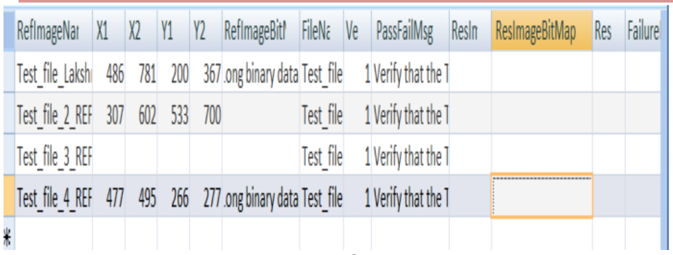
☐ Failure Test Case

☑ Non-Graphic Requirement

Figure6: Selecting Non-graphic condition

D. Database created after capturing phase

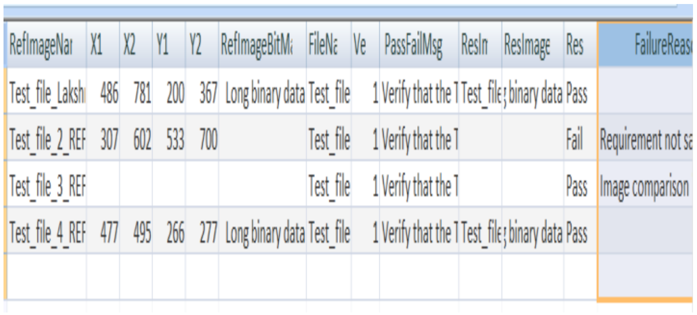
After performing capturing phase successfully, tool automatically stores all the information about the performed test file name, captured images X co-ordinate and Y co-ordinate pixel values, pass-fail messages and also saves the reference image name for further verification.



	RefImageNar	X1	X2	Y1	Y2	RefImageBit!	FileNa	Ve	PassFailMsg	ResIn	ResImageBitMap	Res	Failure
	Test_file_Lakshi	486	781	200	367	ong binary data	Test_file	1	Verify that the T				
	Test_file_2_REF	307	602	533	700		Test_file	1	Verify that the T				
	Test_file_3_REF						Test_file	1	Verify that the T				
	Test_file_4_REF	477	495	266	277	ong binary data	Test_file	1	Verify that the T				
*													

Figure7: Database of capturing phase.

E. Execution phase
Automatically it captures and compares with the created database for particular test cases according to the requirement. If the captured and saved images are in the database for a particular test case then it automatically clicks and says that test case is pass and writes every details in database as a summary file for further verification. If the reference and resultant images are not same then it automatically clicks No by saying that it is a fail test case. Automatic clicking yes or no option performs for every test case in a test file and information has been written automatically in the database as shown in the figure below.



RefImageNar	X1	X2	Y1	Y2	RefImageBitMa	FileNa	Ve	PassFailMsg	ResIn	ResImage	Res	FailureReas
Test_file_Lakshi	486	781	200	367	Long binary data	Test_file	1	Verify that the T	Test_file	binary data	Pass	
Test_file_2_REF	307	602	533	700		Test_file	1	Verify that the T			Fail	Requirement not sa
Test_file_3_REF						Test_file	1	Verify that the T			Pass	Image comparison
Test_file_4_REF	477	495	266	277	Long binary data	Test_file	1	Verify that the T	Test_file	binary data	Pass	

Figure8: Stored database information.

F. Optical Character Recognition Results
The input image may be colored or grayscale image, if it is colored image then it converts into a gray scale image. The optical character recognition has been checked for all the cropped aircraft display images. The input image may be series of characters, character followed by numbers, character followed by symbols and input image may consists of many number of lines. The tool is able to recognize.

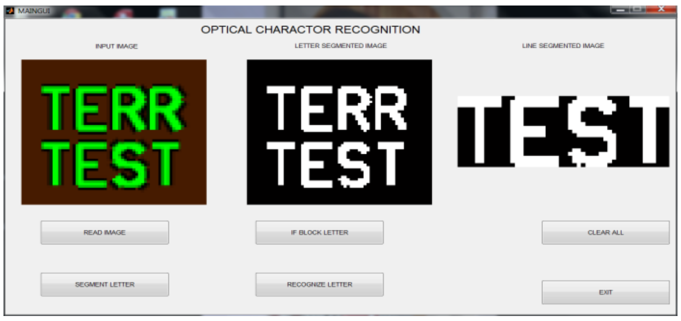
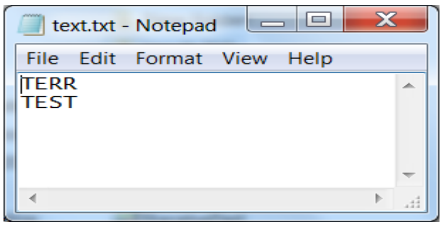


Figure9: Line segmented image



TERR
TEST

Figure10: Recognized characters

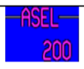
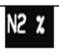
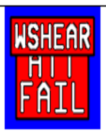
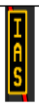
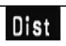
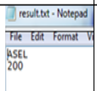
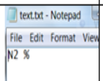
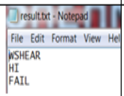
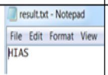
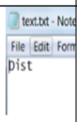
Parameter	Image1	Image2	Image3	Image4	Image5	Average
Input Image						
Result	 result.txt - Notepad File Edit Format ASEL 200	 text.txt - Notepad File Edit Format View N2 %	 result.txt - Notepad File Edit Format View Hel WSHEAR HI FAIL	 result.txt - Notepad File Edit Format View HIAS	 text.txt - Note File Edit Form Dist	
Accuracy	1	1	0.769	0.50	1	0.8538
Time taken	0.39	0.34	0.46	0.43	0.35	0.394

Table1: Time taken and Accuracy rate of characters.

VIII. CONCLUSION & FUTURE WORK

Overall the paper has been a success with the entire paper requirement. The automation tool has been used three popularly used image processing and computer vision techniques to deal with the problem of manual testing for the identification of graphics objects in aircraft display units.
The design could be effectively deployed in various platforms using image comparison, OCR and template matching techniques to reduce the manual efforts required for the graphics testing process with very less amount of time and with no errors and saves huge amount of time. Coming up with an Optical Character Recognition algorithm which works for aircraft display image and it is still an open problem in computer vision. One more challenge is to obtain an algorithm which can be applied for any kind of real time data to perform automated testing. It may be a Moving, rolling, blinking and even if the size of the DU varies between the Image Capturing phase and the execution phase, the tool has to perform automated testing with same accuracy.

REFERENCES

[1] Huizhong Chen, Sam S. Tsai, Georg Schroth, David M. Chen, Radek Grzeszczuk and Bernd Girod, “Robust text detection in natural images with edge enhanced maximally stable extremal regions”.

[2] Boris Epshtein, Eyal Ofek, Yonatan Wexler, “Detecting text in natural scenes with stroke width transform”.

[3] M. Szmurlo, Masters Thesis, Oslo, May 1995,(users.info.unicaen.fr/~szmurlo/papers/masters/master.thesis.ps.gz)

[4] Lukas Neumann and Jiri Matas, “A method for text localization and recoginition in real world images”.

[5] http://www.aishack.in/2010/02/installing-and-getting-opencv-running/.

[6] Artificial neural network based character recognition using backpropagation by Madhup Shrivastava

[7] Dinesh Dileep., “ A Feature extraction technique based on character geometry for character recognition”, IEEE.

[8] Artificial Intelligence and cognitive science ©2006,NilsJ.NilssonStanfordAI Lab http://ai.stanford.edu/~nilsson.

[9] Unicode Optical Character Recognition by Daniel Admassu, 23 Aug 2006

[10] Using Neural Networks to Create an Adaptive Character Recognition System
© 2002, Alexander J. Faaborg Cornell University, Ithaca NY