'National Conference on Recent Advances of Computational Intelligence Techniques in Science, Engineering and Technology'
International Journal of Scientific Research in Computer Science, Engineering and Information Technology | ISSN : 2456-3307 (www.ijsrcseit.com)

Development of Medical Device Application using QT/QML

Dr.I.Manimozhi[1], Ms.Neha[2]
[1]Associate Professor, [2]Assistant Professor
Department of Computer Science & Engineering, East Point College of Engineering & Technology, Banglore, Karnataka, India

ABSTRACT
The main objective here is to provide a solution for Post Development Automation for Medical Device applications. Image-based testing can help in solving problems observed while automating QT/QML applications wherein objects may not be accessible through locators. This solution can help the automation to step ahead and provide accurate results on medical devices on executing the scripts. Algorithms need to be generated for Image-based testing In this paper we will discuss the search of Sub Images in the Main Image with Scalability, Color Resolution, and partly excluded images. Images are not usually named with convenient text or element ID and it's displayed as the image, so the photos are needed to be searched by their pixel. It is essential to compare reference images with the main image and to locate the pattern which we are looking for it. The pattern can be a picture of anything Number, Character, logo, building, or just a duplicate of the same image, and the search operation is to retrieve matching images with different option (scale and color).
Keywords: QT/QM, Post Development Automation Medical Device Application

I. INTRODUCTION

The main objective is to identify the sub image from the main image which is captured as screenshot from the device .This sub image can be further used to validate tests. So firstly we need to design an algorithm for the same. Further based on Image, Image Resolution, Image Size, partly excluded images we can design algorithms respectively. Based on exclude image areas Adding excluded areas to the reference image and only part of the image should be compared with captured image in the screen. In real time scenarios we may come across that a particular image say for example, a submit button might remain same but only the text inside it may vary according to languages chosen by the user. In such situations, we can exclude the text part and take the remaining portion which can be compared with the referenced image. So a solution needs to be designed such that the captured image excluding a specific area can be used to carry out the comparison with the reference image.

Based on Image Resolution: Resolution of images may vary from device to device based on the quality, color tolerance, screen resolution etc. The quality and color tolerance can be handled efficiently by including tolerance values, so that the images captured as screenshots can be compared with the reference images and

Copyright: © the author(s), publisher and licensee Technoscience Academy. This is an open-access article distributed under the terms of the Creative Commons Attribution Non-Commercial License, which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited

provide an accurate result. So here our aim is to provide a one end solution to it by designing an algorithm for the same.

Based on Image Size: Images can vary in size when captured as screenshots in different devices. The images can be either bigger or smaller in size

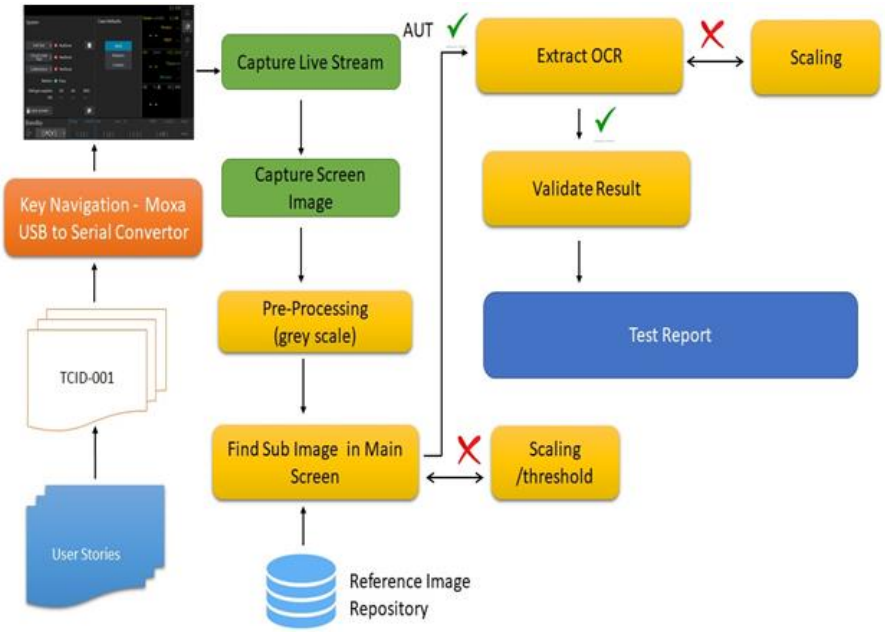
- Larger Image: An algorithm needs to be designed so that larger images that are captured as screenshots satisfies the condition when compared with the reference images during validation.
- Smaller Image: An algorithm needs to be designed so that smaller images that are captured as screenshots satisfies the condition when compared with the reference images during validation.

II. LITERATURE REVIEW

Background: Testing of Medical Device Applications (QT/QML) using Squish Automation tools helps to build test scripts which provides an accurate and efficient way of logging the test results. During development stage most Automation testing tools copying plugin/agents/packages in the targeted systems to run the test scripts But post development copying agents file into targeted systems is restricted due to security reason.

Problem Statement: To run automation and identify the regression issues during post development environment is challenging part in the medical devices. Third party software packages are restricted to copying in the medical devices during the post development and unable to run the automation scripts without third party plugin or agents in the QT/QML based targeted setup environment.

III. SYSTEM ARCHITECTURE



Approach: Image and text recognition is the backbone of automating the desktop applications. Image recognition is finding one image within another image. Reference image defined during the wireframe design time and actual image which is capture as a screenshot of the Real time application when the automation flow is running and validated against with reference image. Image based validation is the visual verification of rendering image on the screen for GUI application and its suitable for medical application.

Visual testing in medical software programs is a exceptional assurance activity for visible aspects of the utility's consumer interface and it's additionally referred to as Visual Validation Testing. Verifying the best information and content material are getting displayed on the utility's the front stop. Additionally, it also validates the format and appearance of each visual element present on the consumer interface and the complete UI itself.
Image search in the screenshot and extract the text, compare with reference text then follow the validation. Image with icon or object based search will not accurate based on resolution, color, image quality etc.. in ordered to avoid the complexity. text area image search algorithm required to avoid problems.

IV. IMPLEMENTATIONS

Algorithm to find Sub Image on Main Image

Consider a screenshot captured from a device as shown in "Fig 1". We need to identify a sub image "Fig 2" from the captured screenshot.

Algorithm:

Step 1: Begin:

Step 2: Input the source image (Original image)

Image src = ImageIO.read (new
File ("srcfile.jpg")

2.1: Define value of x- coordinate
Define value of y – coordinate
Define height
Define width

Step 3: Input the destination image file wherein the cropped image needs to be copied.

Step 4: Paint the desired part of the source image (src) onto the Destination image file (dst) using Graphics object

Step 5: Write the buffered image out from the Destination image file to a new file using ImageIO.

ImageIO . write (dst ,"png", new File("final_dst_file")

Step 6: Stop

Obtain text area image from the wireframe with scaled 1:1 - Load image, convert to grayscale, Gaussian blur, and adaptive threshold

Simple Thresholding became defined with extraordinary kinds of thresholding techniques. Another Thresholding technique is Adaptive Thresholding. In Simple Thresholding, a international fee of threshold was used which remained regular during. So, a steady threshold cost won't help within the case of variable lights conditions in different areas. Adaptive thresholding is the approach in which the brink value is calculated for smaller areas

cv2.ADAPTIVE_THRESH_MEAN_C: Threshold Value = (Mean of the neighbourhood area values – constant value). In other words, it is the mean of the blockSize×blockSize neighborhood of a point minus constant.

cv2.ADAPTIVE_THRESH_GAUSSIAN_C: Threshold Value = (Gaussian-weighted sum of the neighbourhood values – constant value). In other words, it is a weighted sum of the blockSize×blockSize neighborhood of a point minus constant.

Combine adjacent text. We create a rectangular structuring kernel then dilate to form a single contour

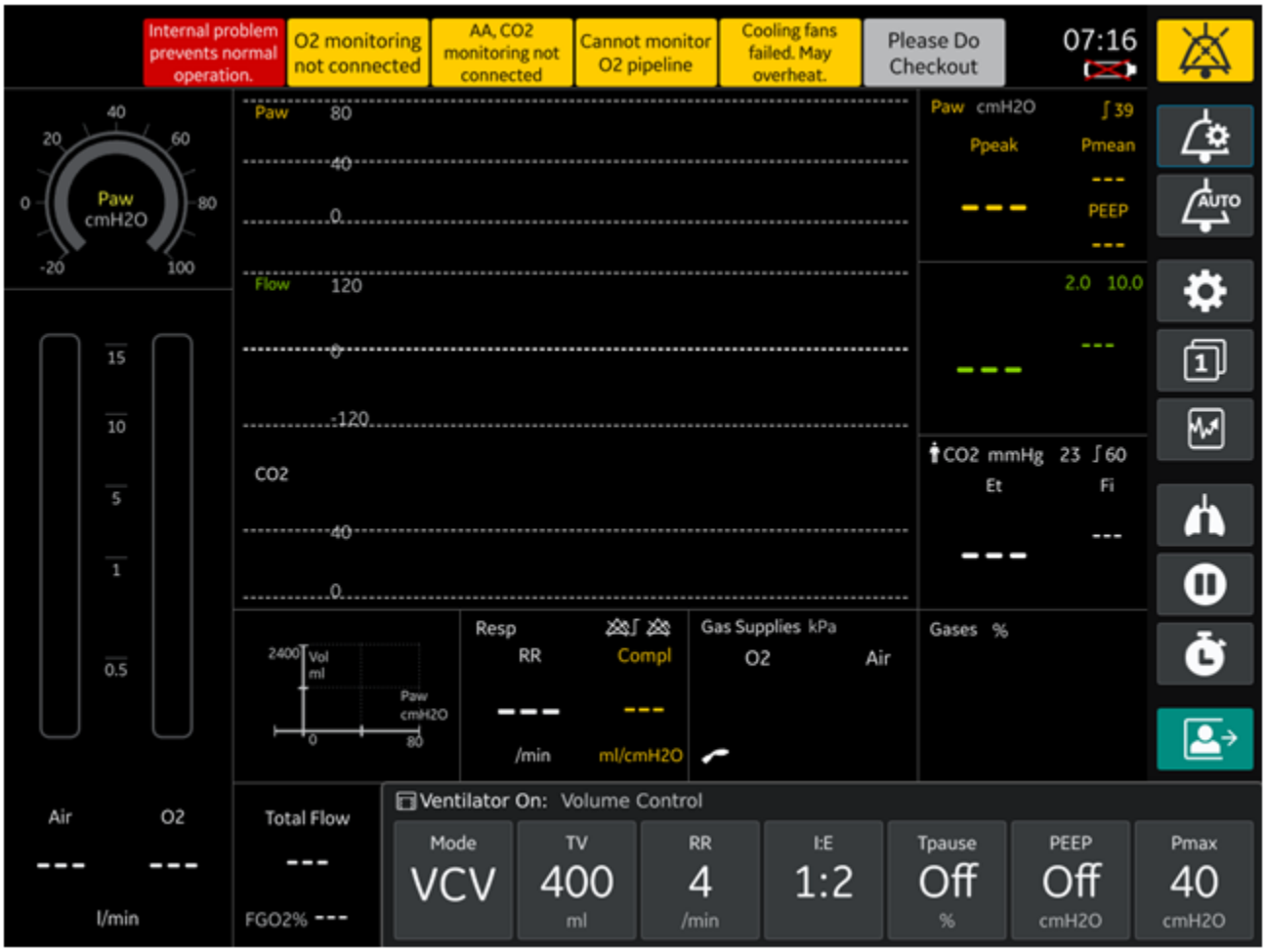


Fig 1: Image captured as screenshot from Wireframe Design

Algorithm to Upscale/Downscale of Reference image : Images needs to be upscaled/Downscaled according to Main image the pixels can be calculated and the exact position of the image can be located.

Algorithm:

Step1: Begin:

Step2: Input Source image (src)

2.1: Input the Captured Image (Screenshot from the device ass source Image)

Step3: Input Reference image to fins the sub image in the source image

Step4: Sub-image not detected then rescaling factor should be carried either upscaled /downscaled.

Step5: Search sub-image with incremental/decremental with scale factor

Step6: Re-Search Sub-Image – loops continues till target found

Step 8: Stop

Width and height of the image measured with help of sub image detection in the main image followed with Key navigation will be occurred. Again validation reference image search in the main image and extract text and validated with expected result .

Upscaled and downscaled mechanism will be taken both image search and OCR extraction

Automation Flow Chart:

Algorithm for dynamic text based on exclude image areas: Consider an image which remains fixed always and only a sub image of this image might vary frequently. In such situations the sub image can be cropped and the excluding area can be taken as the reference image to compare with the image obtained during real time.

We design an algorithm such that the sub image is detected from the original image and further that part is cropped from the original image. The image obtained after this is the reference image which is used for comparison.

Procedure1: Image cropping

Step 1: Begin:

Step 2: Input the source image (Original image)

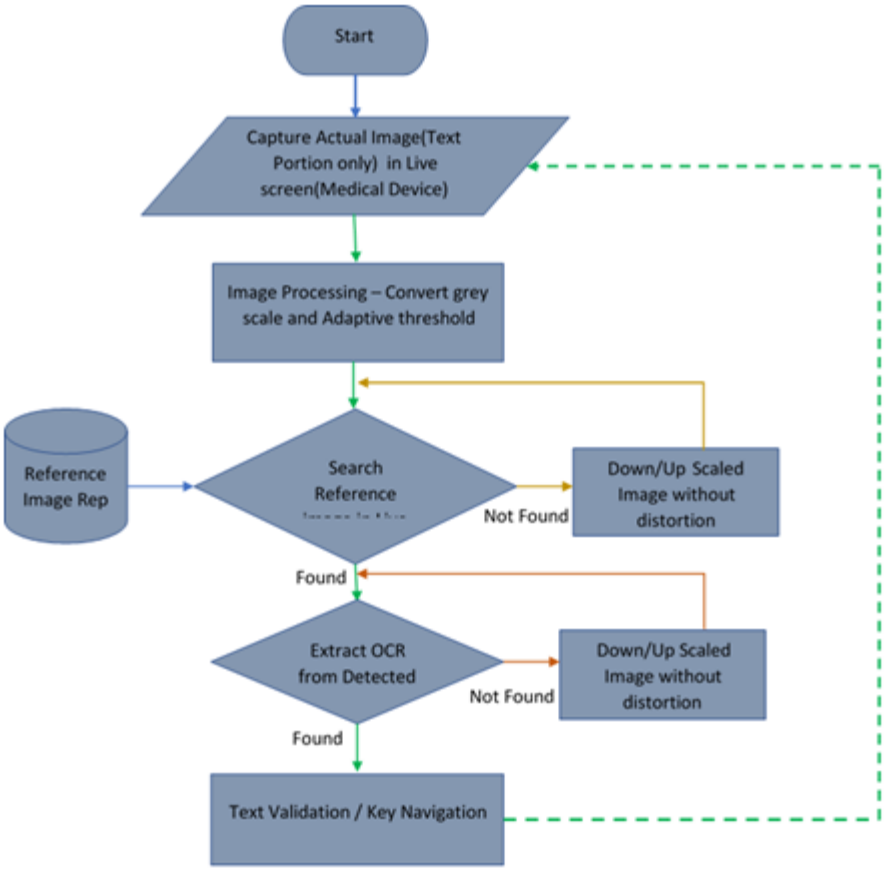
Image src = ImageIO.read (new File ("srcfile.jpg")

2.1: Define value of x- coordinate

Define value of y – coordinate

Define height

Define width



Algorithm:

Step 3: Input the destination image file wherein the cropped image needs to be copied.

BufferedImage dst= new BufferedImage (w, h, BufferedImage.TYPE_INT_ARGB)

Note: The width and height should be same as the source image

Step 4: Paint the desired part of the source image (src) onto the Destination image file (dst) using Graphics object dst.getGraphics().drawImage(src, 0,0,w,h,x,y,x+w,y+h,null)
Step 5: Write the buffered image out from the Destination image file to a new file using ImageIO.
ImageIO . write (dst ,"png", new File("final_dst_file")
Step 6: Stop
Procedure 2: Locate the image from the screenshot captured.
Algorithm findSubimage (BufferedImage Im1, BufferedImage Im2)
Step 1: Begin
Step 2: Input Im1 that is the screenshot captured from the Medical device
Input Im2 that is the cropped image "final_dst_file" obtained in Procedure1
2.1: Get the width and height of Im1 and Im2
w1 ← Im1. getWidth ()
h1 ← Im1. getHeight ()
w2 ← Im2. getWidth ()
h2 ← Im2. getHeight ()
assert (w2 < = w1 && h2 < = h1)
Step 3: Keep track of best position found
bestX ← 0
bestY ← 0
lowestDiff←Double.POSITIVE_INFINITY
Brute force search method through the whole image
for (int x =0 ; x < w1-w2 ; x++) do
for (int y =0 ; y < h1-h2 ; y++) do
double comp ← compareImages(Im1.getSubimage(x,y,w2,h2),Im2)
if (comp < lowestDiff) then
bestX ← x
bestY ← y
lowestDiff ← comp
print (lowestDiff)
return new int[] {bestX, bestY}

Step 4 : Determine how different two identically sized regions are Function compareImages (BufferedImage Im1, BufferedImage Im2)
assert(Im1.getHeight() == Im2.getHeight() && Im1.getWidth() == Im2.getWidth())
variation ← 0.0
for (int x = 0 ; x < Im1.getWidth() ; x++) do
for (int y = 0 ; y < Im1.getHeight() ; y++) do
variation += compareARGB(Im1.getRGB(x,y) ,Im2.getRGB(x,y)) / Math.sqrt(3)
return variation / (Im1.getWidth() * Im1 . getHeight())
end function
Step 5 : Calculate the difference between two ARGB Colours

```
function   compareARGB  (int rgb1 , int rgb2)
        r1 ← ((  rgb1 >> 16) & 0XFF)/255.0
        r2 ← ((  rgb2 >> 16) & 0XFF)/255.0
        g1 ← ((  rgb1 >> 8) & 0XFF)/255.0
        g2 ← ((  rgb2 >> 8) & 0XFF)/255.0
        b1 ← ((  rgb1  &  0XFF) / 255.0
        b2 ← ((  rgb2  &  0XFF) / 255.0
        a1 ← ((  rgb1 >> 24) & 0XFF)/255.0
        a2 ← ((  rgb1 >> 24) & 0XFF)/255.0
return a1 * a2 * Math.sqrt((r1-r2) * (r1 – r2 ) + (g1 –g2 ) * (g1 –g2 ) + (b1 –b2 ) * (b1 –b2 ))

end function

Step 6:   Stop
end algorithm
```

V. RESULTS

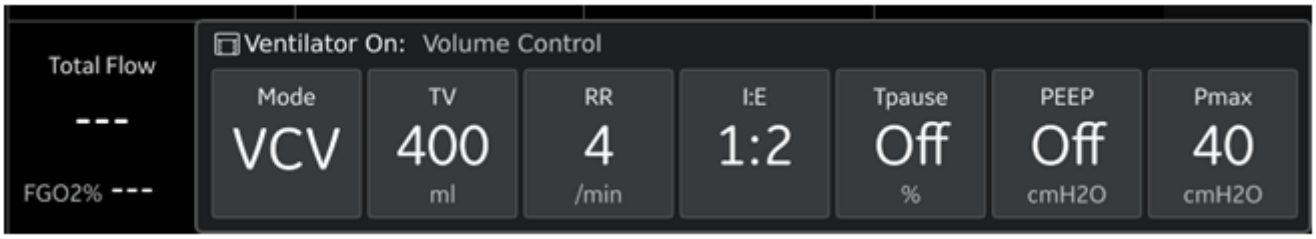


Fig 4: Screenshot obtained from the medical device application

Fig 4.1: Locate the Submit button from the screenshot in Fig 4 which is the value to be located with the text "400" being cropped off from the button (Src Image)

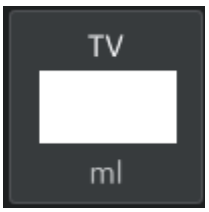


Fig 4.2: Submit button after cropping the text inside the button

Consider figure Fig 4 wherein the screenshot is captured from the medical device.

Consider the figure Fig 4.1 from which the Submit button whose text needs to be cropped off is located and then Consider figure Fig 4.2 which is the Submit button obtained after cropping of the text which is stored in the final_dst_file according to Algorithm in Procedure 1.

Procedure 2: The figure Fig 4.2 is compared with Fig 4.1 according to Algorithm of Procedure2 and thereby the subimage Fig 4.2 is located within the image in Fig 4.1

Algorithm based on Image Resolution: Color resolution of the medical application may vary from one device to another and therefore an algorithm should be designed to calculate the percentage of similarity between two images pixel wise. This percentage of similarity can be then compared with a tolerance value to judge the similarity

Algorithm:

compareImage (File fileA, File fileB)

Step1: Begin

Step2: Input the reference image file and the image file obtained from the screenshot which needs to be compared with the reference image.

Step3: Initialize the percentage value to

Float percentage = 0

3.1: Initialize the threshold value

3.2: Initialize the tolerance value

3.3: Calculate buffer data from both image files and store the value in a integer variable for further calculation

```
BufferedImage BufIm_A = ImageIO.read(fileA)
DataBuffer  DbufIm_A =  BufIm_A.getData().getDataBuffer()
Integer sizeA = DbufIm_A.getSize()
BufferedImage BufIm_B = ImageIO.read(fileB)
DataBuffer  DbufIm_B =  BufIm_B.getData().getDataBuffer()
Integer sizeB = DbufIm_B.getSize()
```

Step 4: Initialize a count variable count to 0

4.1: Compare data objects

```
If (sizeA == sizeB)
    for (int i = 0;          i < sizeA ; i++)
        if (DbufIm_A.getElem(i) ==
          (DbufIm_B.getElem(i) )
          count  += 1
        endif
    end for
   percentage = ( count  * 100 )/sizeA
 Else
   "Images are not of same size"
Return percentage
```

Step 5: Compare the percentage of similarity with the tolerance value

```
If (percentage <= threshold)
```

"Images are similar"
Else if (percentage + tolerance <=threshold)
"Images are similar"
Else if (percentage - tolerance <=threshold)
"Images are similar"
Else
"Images are not similar"

Step 6: End

Fig 5. Screenshot from Device

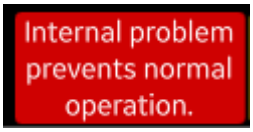


Fig 5.1. Reference Image

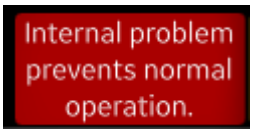


Fig 5.2. Image to be compared from Device

According to the Algorithm Figure Fig 5.1 and Figure Fig 5.2 are compared and based on the percentage of similarity the Images are found to be similar or not.
the image from the screenshot.

VI. CONCLUSION

The primary aim of this paper presentation was to find a solution to execute regression issues and layout verification with Automation scripts in medical device application using Image based Validation. We have listed down the scenarios that arise while automating and have tried to figure out solutions to the same. The idea presented here to execute the test scripts in post development environment without copying any third-party files/packages/agents in the targeted devices.

VII. REFERENCES

[1]. http://tech-algorithm.com/articles/nearest- neighbor-image-scaling
[2]. Richard Szeliski (Microsoft Research). Computer Vision: Algorithms and Applications szeliski.org/Book

[3]. Digital Image Processing Algorithms and Applications by I.Pitas
[4]. Principles of Digital Image Processing Burger, W. (et al.) (2019)
[5]. Saugat Bhattacharyy Kunal Pal "IoT-Based Applications in Healthcare Devices published in Journal of HealthCare Engineering, Hindawi, 2021
[6]. Jennifer C Goldsack " Verificaton,Analytical.Validation and Clinical Validation" published in Digitl Medicine.
[7]. A Disciplined Approach to User Interface Development Using Qt (QML) for Medical Devices, published in Medcuity software2020
[8]. S M Riyazul "The Internet of Things for Health Care: A Comprehensive Survey" published in IEEE access 2018