INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 05, May 2024, pp: 298-310

e-ISSN : 2583-1062

Impact Factor: 5.725

BLOOD TYPE DETECTION USING IMAGE PROCESSING

Mrs. T. Ratnamala[1], Mohammed Yaseen Hussain[2], Danthalapally Kalyani[3], Kommana Ajay[4], Penugonda Aravind[5]

[1]Assisstant. Professor, CSE Dept, ACE Engineering College, Hyderabad, India.

[2,3,4,5]Student, CSE Dept, ACE Engineering College, Hyderabad, India.

ABSTRACT

Blood group is classification of blood based on the presence or absence of inherited antigenic substances on the surface of red blood cells. Depending on the blood group system, these antigens could be proteins, carbohydrates, glycoproteins or glycolipids. The abo system is the most important blood group system in human blood transfusion. Blood typing is the process of identifying a person's specific blood type. Every person should know their blood group and it is essential for the realization of a safe blood transfusion, so as to administer a blood type that is compatible with the type of receiver. The tests currently available require moving the person to the laboratory and collecting the samples and then manually detecting the blood type (after antigens are added to the samples). There may not be enough time to determine the blood type using this process and so the administered blood type-o negative (universal donor) is considered which provides less risk of incompatibility. During situations like transfusion of blood, donating blood, road disaster and other urgent situations, inquiring about the specific blood group in a short time span is very important. Pre-transfusion tests are performed manually by technician's analysts, which sometimes lead to the occurrence of human errors in procedures, reading and interpreting of results as a large number of samples are handled with the test. This process is repetitive to perform and it may lead to manual fault. The proposed system aims to develop an automated system which uses image processing techniques to perform blood tests based on abo and rh blood typing systems. The proposed system not only determines the blood type but also generates the patients' blood report. The main goal of the system is to provide outcomes in a short time span with greater exactness. Thus, the system allows us to determine the blood group of a person in an emergency situation. The system eliminates traditional transfusions based on the principle of universal donor and reducing transfusion reactions

1. INTRODUCTION

Blood group identification is the key step to ensure blood transfusion safety. In the case of emergency blood transfusion, rapid identification of the type of blood is essential, and directly related to the survival of the patient. In recent years, with the rapid development of image processing technology, the research and development of automatic recognition systems based on machine vision technology has become an urgent need in the field of blood group identification. The ABO blood group system is found and identified as the first human blood group system by Austria Rand Steiner in early nineteenth Century, according to the surface of red blood cells have no the distribution of specific antigen (agglutinogen) AlB, blood is divided into four types: A, B, AB and O. The blood type A, the red blood cells only contain agglutinogen A, the serum have anti B lectin; the blood type B, the red blood cells only contain agglutinogen B, the serum have anti A lectin; blood type AB, the red blood cells contain A, B two kinds of agglutinogen, have no anti-A and anti-B lectin hormone in serum; the blood type 0, the red blood cells contain A, B two kinds of agglutinogen, the serum have anti-A and anti-B lectin. The red blood cell which has agglutinogen A can agglutinate with Anti A lectin; anti-B lectin can agglutinate with B agglutination of red blood cells, based on this principle, ABO blood group identification can use red cell agglutination test, and then through the method of machine vision to determine blood type by whether the results obtained agglutination

2. OBJECTIVES

The main objective of this project is to develop a quick and efficient way to determine the blood type. In today's world where blood transfusion plays an important role in trauma and operational cases, it is really important to transfuse the correct blood type into the patient or else the blood can clump,which maybe fatal. The typical technique of testing is laboratory testing, and the results obtained may be inaccurate. To address this, the suggested approach of blood testing based on image processing requires less human participation and is very accurate.

Additionally, the proposed image processing approach not only enhances accuracy and reduces human error but also significantly expedites the blood typing process. Traditional laboratory testing methods often entail time-consuming procedures, including sample collection, transportation, and analysis, which can be particularly challenging in emergency situations where every minute counts. By leveraging image processing techniques, the time required for blood typing can be dramatically reduced, allowing for swift and precise determination of blood type directly at the point of care. This rapid turnaround time can be critical in emergency scenarios, enabling healthcare providers to promptly administer the appropriate blood transfusion without delay, thereby improving patient outcomes and potentially saving lives. Furthermore, the integration of image processing technology into blood typing procedures holds promise for scalability and accessibility, as it can be implemented using widely available digital imaging devices, such as smartphones or portable scanners, making it feasible for use in various healthcare settings, including remote or resource-limited environments.

3. PROBLEM STATEMENT

The current methods of blood typing predominantly rely on labor-intensive laboratory procedures, presenting several challenges in terms of accuracy, efficiency, and accessibility. Traditional techniques involve manual sample collection, processing, and analysis, which are prone to human error and time-consuming processes. Moreover, the need for specialized equipment and trained personnel limits the availability of blood typing services, particularly in remote or underserved regions. Consequently, there is a pressing need for innovative approaches to blood typing that can mitigate these challenges and ensure rapid, accurate determination of blood type, especially in critical situations such as trauma and surgical cases where timely transfusions are crucial for patient survival.

Inaccuracies in blood typing can have serious consequences, including incompatible transfusions that may lead to life-threatening complications such as hemolytic reactions. The existing laboratory-based methods, while generally reliable, are not infallible and may yield erroneous results due to various factors such as sample contamination, procedural errors, or misinterpretation of test outcomes. These inaccuracies pose a significant risk to patient safety, highlighting the urgency for more robust and foolproof blood typing techniques. Furthermore, the logistical complexities associated with laboratory testing, including sample transportation and processing delays, exacerbate the problem by prolonging the time required to ascertain blood type compatibility and administer appropriate transfusions.

4. LITERATURE SURVEY

The ABO blood group system and the Rh D blood group system are the most important blood group systems used for determining the blood group of a person and the test used for determining the blood group is blood typing. The blood groups are defined by the presence or absence of a specific antigen on the surface of a red blood cell. There are four ABO blood groups: A, B, AB and O. They refer to the presence of different antigens on the red blood cells and antibodies in the blood. Blood group means you have neither antigen present on the surface of RBC and antibodies A and B in the blood, but blood group AB means you have both the A and B antigens present and no antibodies in the blood. Blood group A has antigen A present on the surface and antibody B in the blood, while blood group B has antigen B present on the surface and antibody A in the blood.

Referring to the Rh D blood group system, one more antigen called Rh D is involved while determining the blood group. If D antigen is present on the red blood cells of a person then he/she is Rh D positive, while one who does not have D antigen on the red blood cells is Rh negative. While having a blood transfusion, blood grouping is very important. If there is any incompatibility while transfusing blood, it can be fatal causing intravenous clumping in the patient’s blood. Antigens on the red blood cells in the blood of the person receiving blood can be attacked by the antibodies produced in the blood due to incompatibility. Naturally occurring antibodies are not present in the blood of a person having blood group O, hence person with blood group O can safely donate blood to a person with any other blood group.Similarly, a person with blood group AB can receive blood from a person having any other blood group safely due to absence of antibodies in the blood. A person with a positive blood group can be given either Rh D positive or Rh D negative blood, but a person with a negative blood group can only receive blood from a person with Rh D negative blood. Hence, a person with O -ve blood group is a universal donor whereas one with AB +ve blood group is a universal receiver.

There is a scope for determining blood group and the software developed is by using image processing techniques. Three samples of blood are taken on a slide, each mixed with reagent anti-A, anti-B and anti-D respectively. 6 After some time, agglutination occurs and the result is interpreted according to the occurrence of agglutination. The agglutination reaction is the occurred reaction between the antibody and the antigen, indicating the presence of a particular antigen. of the occurrence of agglutination determines the blood group of the patient. Thus, the software developed based on image processing techniques allows the detection of agglutination on the slide through an image captured after mixing specific reagents and consequently the blood group of the patient is determined. Few papers have suggested methods in order to identify the blood groups. An adaptive method of Hough transform techniques has been presented to analyze blood samples by Keerthana et al.. Image processing techniques such as thresholding, morphological operations, attribute extraction by ScaleInvariant Feature Transform (SIFT) algorithm, and classification using Support Vector Machine (SVM) have been proposed for the identification of blood groups by Pathan et al. Different systems have been reviewed and their outputs evaluated to detect the class of blood sample using MATLAB calculations to assist researchers in their work by Jogi et al. Sathiyan et al. employed image processing techniques such as Region of Interest (RoI), picture cropping, gray-scale segmentation, contouring or edge-detection algorithms, and HSL (hue, saturation, lightness) average value to quickly determine the patient's blood type. Pavithra provided a survey on current developments in blood group detection based on several strategies such as thresholding, histogram, and cluster conditions. Morphological operations have been used for classifying the blood regions based on statistical features such as area, shape, size of the segmented regions by Dantiet. The third and final stage involves the production of the OFSNN model, where the optimized features and the novel index synergistically contribute to the development of a robust neural network architecture. Existing literature emphasizes the significance of model interpretability and the need for methods that go beyond conventional accuracy metrics. While neural networks have exhibited remarkable performance in various applications, their "black-box" nature often hinders their adoption in domains where transparency is paramount. The OFSNN model, emerging from the proposed framework, integrates the benefits of optimal feature selection with a neural network structure, promising not only improved predictive accuracy but also enhanced transparency and interpretability. Sometimes the human eye may give us an inaccurate result, but if we detect the blood group using image processing technology then the small error in the results that are calculated and given by humans is reduced. Using image processing technology, we can give the best result as this technology is growing faster and faster. This method can quickly and accurately classify the blood group.

Combination of cluster and patch			Result as a blood group
C	P	C	A
P	C	C	B
C	C	C	AB
P	P	P	O

5. PROPOSED SYSTEM

Firstly, the Blood samples are mixed with four different reagents namely anti-A, anti-B,anti-D and Control are taken on a slide. After some time, agglutination may or may not occur. After the occurrence of agglutination, the slide containing 4 samples of blood mixed with four reagents is captured as an image and uploaded to the software developed to process the image. The software reduces the chances of false detection of a blood group. Image processing techniques used for blood group detection are :

- Pre-processing techniques
- Color Information Extraction
- Automatic Threshold Segmentation
- Object Segmentation using Ni-Black Algorithm
- Morphological Analysis
- Feature Extraction

6. HARDWARE AND SOFTWARE REQUIREMENTS

6.1 HARDWARE REQUIREMENTS:

- Processor – Pentium IV
- RAM – 4 GB (min)
- Hard Disk – 20 GB
- Key Board – Standard Windows Keyboard
- Mouse – Two or Three Button Mouse
- Monitor – SVGA

6.2 SOFTWARE REQUIREMENTS:

- Operating system – Windows 7, 8, 10, 11, Mac OS
- Coding Language – Python
- Back-End – Python.
- IDE: PyCharm, Visual Studio
- Framework: Tkinter, OpenCV

7. PACKAGES USED

Detecting blood types using image processing typically involves analyzing images of blood samples or blood typing cards to identify the presence of specific antigens on red blood cells. Various image processing techniques can be employed for this purpose, often relying on computer vision and machine learning algorithms. Here are some common modules or libraries used in blood type detection using image processing:

OpenCV (Open Source Computer Vision Library):

OpenCV is a popular open-source library for computer vision and image processing tasks. It provides various functions and modules for image manipulation, feature detection, object recognition, and more. In blood type detection, OpenCV can be used for tasks such as image preprocessing, segmentation, feature extraction, and pattern recognition.

scikit-image:

scikit-image is a collection of algorithms for image processing in Python. It provides a range of functions for tasks such as image filtering, segmentation, feature extraction, and morphology. scikit-image can be used alongside OpenCV for more advanced image processing tasks in blood type detection.

NumPy and SciPy:

NumPy and SciPy are fundamental libraries for numerical computing and scientific computing in Python, respectively. They provide efficient data structures and functions for mathematical operations, linear algebra, statistics, and signal processing. These libraries are often used in

conjunction with image processing libraries like OpenCV and scikit-image for handling image data and performing various computations.

Machine Learning Libraries (e.g., TensorFlow, PyTorch):

Machine learning algorithms, particularly deep learning models, can be utilized for blood type detection from images. Frameworks like TensorFlow and PyTorch provide tools for building, training, and deploying machine learning models, including convolutional neural networks (CNNs)

for image classification tasks. These libraries enable the development of custom models tailored to

the specific requirements of blood type detection.

Image Segmentation Algorithms:

Image segmentation is a crucial step in blood type detection, where the regions of interest (e.g., blood cells or typing card patterns) are separated from the background or other irrelevant objects in the image. Various segmentation algorithms, such as thresholding, edge detection,

region growing, and watershed segmentation, can be implemented using the before mentioned libraries to isolate relevant features for further analysis.

Feature Extraction Techniques:

Once the regions of interest are segmented, feature extraction techniques are applied to extract discriminative features from the image data. These features can include shape descriptors, texture features, color histograms, or any other relevant characteristics that help distinguish

different blood types.

Pattern Recognition and Classification:

Finally, pattern recognition and classification algorithms are employed to identify the blood type based on the extracted features. This may involve training a machine learning model (e.g., SVM, random forest, CNN) on a labeled dataset of blood type images to predict the blood type of new

samples based on their extracted features.

By integrating these modules and techniques, a comprehensive blood type detection system using image processing can be developed, allowing for automated analysis and interpretation of blood type information from digital images.

8. ALGORITHM

The image processing is handled by OpenCV(open-source computer vision(library).It is a complete suite of tools written and developed in C, C++& python for both image and video processing and analysis.

NumPy offers comprehensive mathematical functions, random number generators,linear algebra routines and fourier transforms and more.

Matplotlib is a comprehensive library for creating static animatedinteractive visualization in python.

Tkinter is the de facto way in python to create graphical userinterfaces(GUI's)and is included in all standard python distributions. It is the only Framework built into the python standard library.

Pillow(PIL) library contains all basic image processing functionalities and also provides support for opening, manipulating and saving many different image file formats.

The Python Imaging Library (PIL expansion) is the de facto image processing package for the Python programming language. It includes lightweight image processing tools that help with image editing, creation, and saving. Pillow was announced as a future replacement for PIL. Pillow is compatiblewith a wide range of image file types, including BMP, PNG, JPEG, and TIFF. By providing new file decoders, the library supports the addition of support for future formats.

The Niblack thresholding technique seeks to improve performance,particularly for microscopic images. It is a local thresholding algorithm that adjuststhe threshold based on the local mean and standard deviation over a given windowsize around each pixel.

9. SOURCE CODE

```
from ast import Break
from tkinter import *
from tkinter import messagebox, filedialog
from PIL import Image, ImageTk
import cv2
import numpy as np
blood = [False, False, False, False]
q = 1
f = 0
v = 0
p1 = ''
p2 = ''
p3 = ''
p4 = ''
fname = ''
class Detect(Frame):
def __init__(self, master=None):
Frame.__init__(self, master)
self.master = master
self.init_window()
def init_window(self):
self.img = img = PhotoImage(file='C:\\Users\\YASEEN\\OneDrive\\Desktop\\Blood-group-detection-using-python-ML\\blood_logo.png')
self.imglbl = Label(self, image=img)
self.imglbl.place(x=0, y=0)
# Code Segment for Icon and title
self.configure(background='powder blue')
self.pack(fill=BOTH, expand=1)
self.master.title("Blood Group Detection System")
# self.master.iconbitmap('Blood.ico')
e5 = Button(self, text="Choose Image", command=self.imgload)
e5.place(x=10, y=120)
l5 = Label(self, text="Blood group detector",
fg="red", font=("Helvetica", 23))
l5.place(x=630, y=15)
def Close(self, event=None):
root.destroy()
def message(self, q):
messagebox.showinfo("Result", q+"Confirmed")
```

```
def start1(self):
self.start(p1, "Anti A")
def imgload(self):
print("Image copied to environment")
global blood
global q
global f
global v
global p1
global p2
global p3
global p4
global fname
global pid
blood = [False, False, False, False]
q = 1
f = 0
v = 0
p1 = ''
p2 = ''
p3 = ''
p4 = ''
fname = ''
global pid
global pname
#global v
v += 1
s = filedialog.askopenfilename()
x = ""
# print(s)
i = len(s)-1
# print(i)
while s[i] != '/':
x += s[i]
# print(x)
i -= 1
# print(i)
#global p1
p = x[::-1]
self.p = Image.open(x[::-1])
r = self.p.resize((500, 225), Image.ANTIALIAS)
i = ImageTk.PhotoImage(r)
l = Label(self, image=i)
l.Image = i
l.place(x=165, y=80)
print(" ")
irt = 1
```

```
if irt == 1:
# print("aaaaaaaaaaaaaaaaaaa")
import time
start_time = time.time()
img = cv2.imread(s)
x = 0
y = 0
h = img.shape[0]
w = 274
crop_img = img[y:y+h, x:x+w]
#cv2.imshow("cropped", crop_img)
# cv2.waitKey()
im1name = "A1.jpg"
cv2.imwrite(im1name, crop_img)
p1 = im1name  # x[::-1]
self.p = Image.open(im1name)
r = self.p.resize((300, 425), Image.ANTIALIAS)
i = ImageTk.PhotoImage(r)
l = Label(self, image=i)
l.Image = i
l.place(x=165, y=330)
print(i)
# print(im1name)
# if v == 4:
#self.ep.configure(relief=RAISED, fg='green', command=self.start1)
x = 300
y = 0
h = img.shape[0]
w = 274
crop_img = img[y:y+h, x:x+w]
#cv2.imshow("cropped", crop_img)
# cv2.waitKey()
im1name = "A2.jpg"
cv2.imwrite(im1name, crop_img)
#global p2
p2 = im1name  # x[::-1]
self.p = Image.open(im1name)
r = self.p.resize((300, 425), Image.ANTIALIAS)
i = ImageTk.PhotoImage(r)
l = Label(self, image=i)
l.Image = i
l.place(x=465, y=330)
# if v == 4:
#self.ep.configure(relief=RAISED, fg='green', command=self.start1)
x = 590
y = 0
h = img.shape[0]
```

```
w = 274
crop_img = img[y:y+h, x:x+w]
#cv2.imshow("cropped", crop_img)
# cv2.waitKey()
im1name = "A3.jpg"
cv2.imwrite(im1name, crop_img)
#global p3
p3 = im1name  # x[::-1]
self.p = Image.open(im1name)
r = self.p.resize((300, 425), Image.ANTIALIAS)
i = ImageTk.PhotoImage(r)
l = Label(self, image=i)
l.Image = i
l.place(x=765, y=330)
x = 890
y = 0
h = img.shape[0]
w = 274
crop_img = img[y:y+h, x:x+w]
#cv2.imshow("cropped", crop_img)
# cv2.waitKey()
im1name = "A4.jpg"
cv2.imwrite(im1name, crop_img)
p4 = im1name  # x[::-1]
self.p = Image.open(im1name)
r = self.p.resize((300, 425), Image.ANTIALIAS)
i = ImageTk.PhotoImage(r)
l = Label(self, image=i)
l.Image = i
l.place(x=1065, y=330)
# if v == 4:
#self.ep.configure(relief=RAISED, fg='green', command=self.start1)
self.start(p1, "Anti A")
self.start(p2, "Anti B")
self.start(p3, "Anti D")
self.start(p4, "Anti H")
self.check()
def process1(self, p, r):  # Obtainting the Grreen image
img = cv2.imread(p)
gi = img[:, :, 1]
cv2.imwrite("p1"+r+".png", gi)
return gi
def process2(self, p, r):  # Obtaining the threshold
gi = self.process1(p, r)
_, th = cv2.threshold(gi, 0, 255, cv2.THRESH_OTSU)
cv2.imwrite("p2"+r+".png", th)
def process3(self, p, r):  # Obtaining Ni black image
```

```
img = cv2.imread('p2'+r+'.png', 0)
th4 = cv2.adaptiveThreshold(
img, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 11, 14)
cv2.imwrite("p3"+r+".png", th4)
def process4(self, r):  # Morphology: fill holes
gi = cv2.imread('p3'+r+'.png', cv2.IMREAD_GRAYSCALE)
th, gi_th = cv2.threshold(gi, 220, 255, cv2.THRESH_BINARY_INV)
gi_floodFill = gi_th.copy()
h, w = gi_th.shape[:2]
mask = np.zeros((h+2, w+2), np.uint8)
cv2.floodFill(gi_floodFill, mask, (0, 0), 255)
gi_floodFill_inv = cv2.bitwise_not(gi_floodFill)
gi_out = gi_th | gi_floodFill_inv
cv2.imwrite('p4'+r+'.png', gi_out)
def process5(self, r):  # Morphing To eliminate small objects
img = cv2.imread('p4'+r+'.png')
kernel = np.ones((5, 5), np.uint8)
open = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
close = cv2.morphologyEx(open, cv2.MORPH_CLOSE, kernel)
cv2.imwrite('p5'+r+'.png', close)
def process7(self, r):  # Histogram
img = cv2.imread('p5'+r+'.png', 0)
img2 = cv2.imread('p1'+r+'.png', 0)
mask = np.ones(img.shape[:2], np.uint8)
hist = cv2.calcHist([img2], [0], mask, [256], [0, 256])
min = 1000
max = 0
n = 0
s = 0
ss = 0
for x, y in enumerate(hist):
if y > max:
max = y
if y < min:
min = y
s += y
n += 1
mean = s/n
for x, y in enumerate(hist):
ss += (y-mean)**2
ss /= n
sd = abs(ss)**0.5
print(r, "-", sd, "\n")
if sd < 580:
return 1
else:
return 0
```

```
def start(self, p, r):
global blood
print(p)
self.process1(p, r)
self.process2(p, r)
self.process3(p, r)
self.process4(r)
self.process5(r)
a = self.process7(r)
print(a, " - ", r)
if a == 1:
if r == "Anti A":
blood[0] = True
elif r == "Anti B":
blood[1] = True
elif r == "Anti D":
blood[2] = True
elif r == "Control":
blood[3] = True
# if (blood == [True,True,False,True])
def check(self):
# if blood[0] is True and blood[1] is True and blood[2] is False and blood[3] is True:
if (blood == [True, True, False, True]):
self.message("Bombay blood")
elif (blood == [False, False, True, False]):
self.message("O+")
elif (blood == [False, False, False, False]):
self.message("O-")
elif (blood == [True, False, True, False]):
self.message("A+")
elif (blood == [True, False, False, False]):
self.message("A-")
elif (blood == [False, True, True, False]):
self.message("B+")
elif (blood == [False, True, False, False]):
self.message("B-")
elif (blood == [True, True, True, False]):
self.message("AB+")
if (blood == [True, True, False, False]):
self.message("AB-")
def stp_full(self, event=None):
root.attributes("-fullscreen", False)
root.geometry("1020x720")
if __name__ == "__main__":
root = Tk()
root.attributes("-fullscreen", True)
app = Detect(root)
```

```
root.bind("<Escape>", app.stp_full)
root.bind("<Delete>", app.Close)
root.mainloop()
```

10. OUTPUT

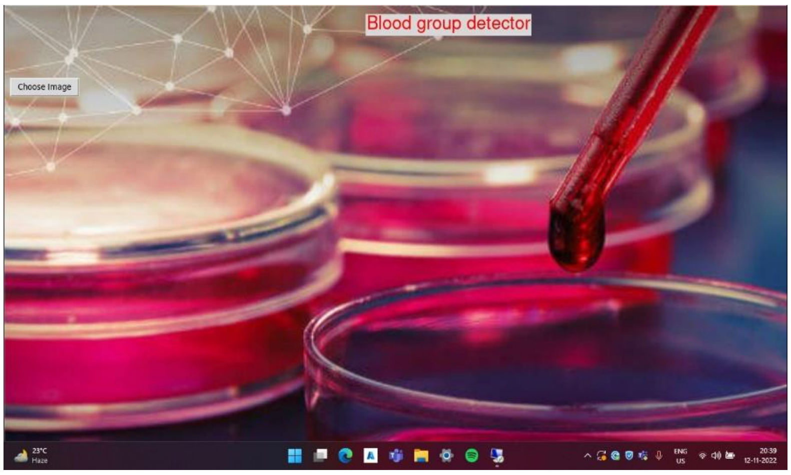
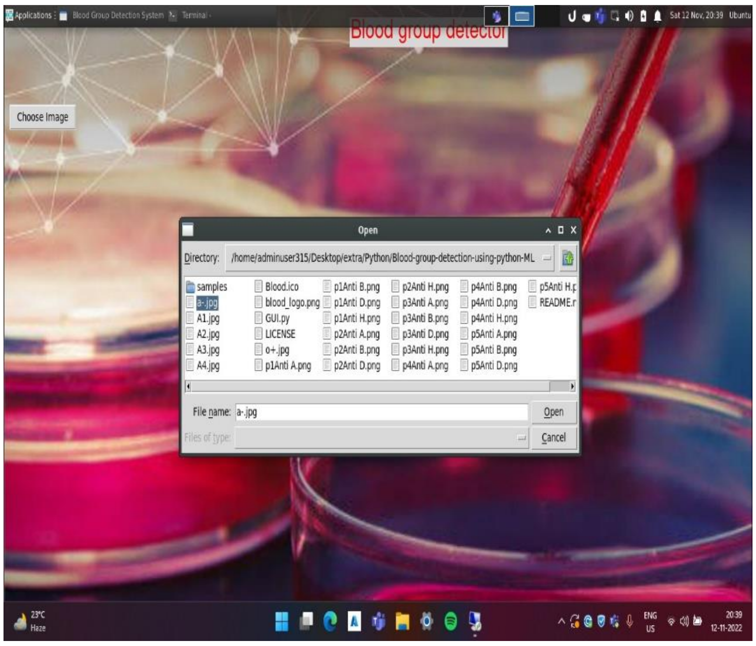


Fig 10.1 Opening Screen



```
Directory: /home/adminuser315/Desktop/extra/Python/Blood-group-detection-using-python-ML

samples     Blood.ico       p1Anti B.png  p2Anti H.png  p4Anti B.png  p5Anti H.p
a-.jpg      blood_logo.png  p1Anti D.png  p3Anti A.png  p4Anti D.png  README.r
A1.jpg      GUI.py          p1Anti H.png  p3Anti B.png  p4Anti H.png
A2.jpg      LICENSE         p2Anti A.png  p3Anti D.png  p5Anti A.png
A3.jpg      o+.jpg          p2Anti B.png  p3Anti H.png  p5Anti B.png
A4.jpg      p1Anti A.png    p2Anti D.png  p4Anti A.png  p5Anti D.png

File name: a-.jpg                                   Open
Files of type:                                      Cancel
```

Fig 10.2 Selecting Sample Slide

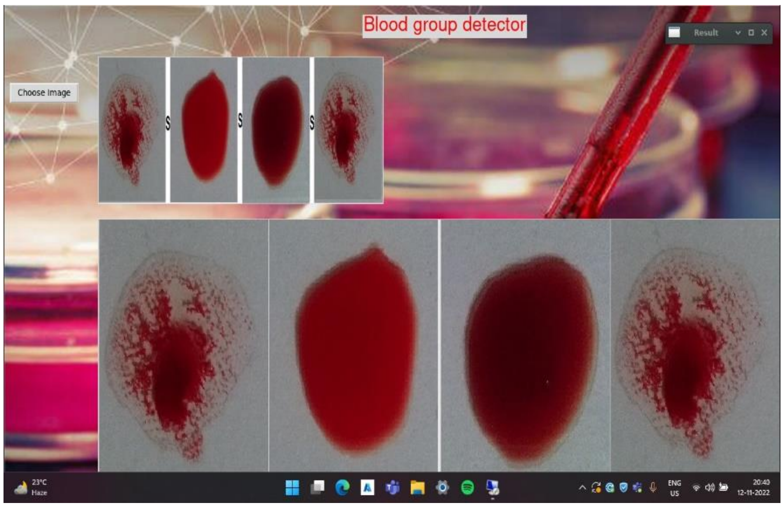


Fig 10.3 Processed Images

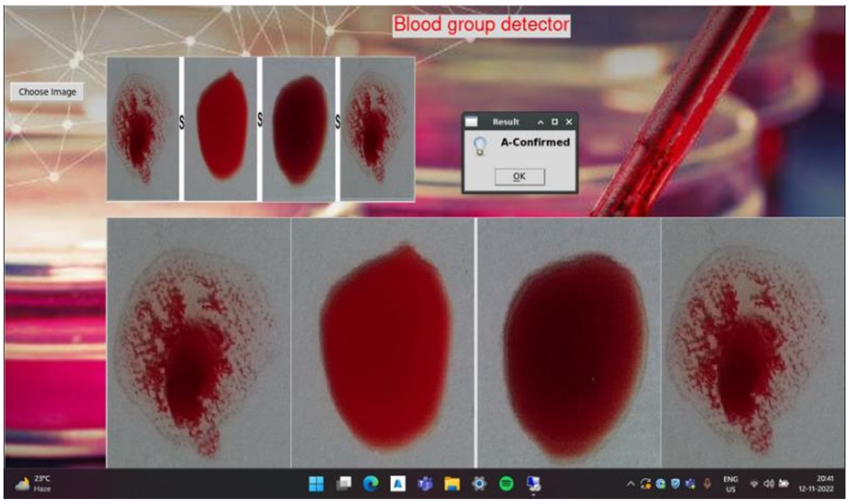


Fig 10.4 Blood Group Confirmed

11. TESTING

Testing the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity, but quality of the data used that matters for testing. Testing is aimed at ensuring that the system was accurate and efficient before live operation commands.

The main objective of testing is to uncover a host of errors, systematically

and with minimum effort and time. Stating formally, we can say, testing is a process

of executing a program with intent of finding an error.

- A successful test is one that uncovers an as yet undiscovered error.
- A good test case is one that has probability of finding an error, if it exists.
- The test is inadequate to detect possibly present errors.
- The software more or less confirms the quality and reliable standards.

Levels of Testing:

Unit Testing: Conduct thorough testing of individual modules and components in isolation to verify their functionality as intended. Ensure that each unit of code, including algorithms, feature extraction methods, and data preprocessing techniques, generates the expected output.

Integration Testing: Evaluate the interaction between different system modules and components to confirm seamless integration. Verify that various components collaborate effectively, detecting and resolving any issues related to data flow and communication between them.

Functional Testing: Validate that the system adheres to specified functional requirements. Test core functionalities such as phishing website detection algorithms, feature selection techniques, and model evaluation procedures.

User Interface (UI) Testing: Assess the usability and intuitiveness of the system's interface. Ensure that users can easily navigate through the application, input necessary data, and interpret detection results without encountering usability issues.

Performance Testing: Evaluate the system's responsiveness, scalability, and resource utilization under diverse conditions. Test the application's capability to handle large datasets and complex computations efficiently while maintaining optimal performance.

Regression Testing: Verify that new updates or modifications do not negatively impact existing functionalities. Re-run previously conducted tests to confirm that any changes have not introduced new errors or compromised the functionality of existing features.

Security Testing: Assess the system's resilience against unauthorized access, data breaches, and other security threats. Implement robust security measures to safeguard sensitive user information processed by the application.

Cross-Browser and Cross-Platform Testing: Ensure compatibility with various web browsers and operating systems. Test the system's functionality on different platforms to ensure consistent performance and user experience across diverse environments.

Adaptive Learning and Continuous Improvement Testing: Evaluate the system's ability to adapt and improve over time. Validate that adaptive learning mechanisms effectively enhance the system's detection capabilities based on new phishing tactics and evolving threats.

User Acceptance Testing (UAT): Engage end-users, cybersecurity experts, and stakeholders in the testing process to gather feedback on the system's usability and effectiveness. Ensure that the system aligns with user expectations and effectively meets their requirements for phishing threat detection.

12. CONCLUSION

A fast accurate and robust blood group judgment method is proposed for the rapid and accurate identification of blood types in the case of emergency transfusion. A large number of experiments show that this method is an efficient and an effective way to detect the occurrence of agglutination and consequently the blood group of the patient in a short interval of time meeting the needs of an automated rapid blood type analyzer. The system would achieve high percentage of sensitivity and specificity which will be useful in determining the blood group in emergency situations. The software developed in image processing is cross compatible across different operating systems making it a reliable and trust-worthy tool for analysis.

The potential future of our project can be classified into the following sections: The Developed Image Processing software heavily relies on CPU, for processing the images, this can be efficiently carried out by introducing the GPU and enabling it for image processing. Present Software is not equipped with any sort of database to store the information about the tested sample, having a database will enable it for further functioning.

The stored information can be taken forward with major blood banks in the city to provide the service and availability of blood to a wider community. Having this information will also entitle us to generate reports and the results help us in rapid transfusions.

13. FUTURE SCOPE

The potential future of our project can be classified into the following sections:

The Developed Image Processing software heavily relies on the CPU, for processing the images, this can be efficiently carried out by introducing the GPU and enabling it for image processing.

Present Software is not equipped with any sort of database to store the information about the tested sample, having a database will enable it for further functioning.

The stored information can be taken forward with major blood banks in the city to provide the service and availability of blood to a wider community.

Having this information will also entitle us to generate reports and the results help us in rapid transfusions.

14. REFERENCES

[1] Yue-fang Dong, Wei-wei Fu, Zhe Zhou, Nian Chen, Min Liu and Shi Chen, "ABO blood group detection based on image processing technology," 2017 2nd International Conference on Image, Vision and Computing (ICIVC), 2017, pp. 655-659, doi: 10.1109/ICIVC.2017.7984637.

[2] Ankita Dalvi, Hanu Kumar Pulipaka, “Determination of Blood group using Image processing“, International Journal of Scientific & Engineering ResearchVolume 9, Issue 3, March-2018 ISSN 2229-5518. https://doi.org/10.1016/j.proeng.2010.09.155.

[3] G., Ravindran & Titus, Joby & Manikandan, Pravin & Pitchai, Pandiyan. (2017). Determination and Classification of Blood Types using Image Processing Techniques. International Journal of Computer Applications. 157. 975-8887. 10.5120/ijca2017912592.

[4] "NON INVASIVE BLOOD GROUP DETECTION ", International Journal of Emerging Technologies and Innovative Research (www.jetir.org | UGC and issn Approved), ISSN:2349-5162, Vol.6, Issue 5, page no. pp139-143, May 2019, Available at : http://www.jetir.org/papers/JETIRCU06029.pdf