DEVELOPMENT OF ATTENDANCE FOLIO DETECTION USING IMAGE PROCESSING

Prof. Amit D. Bhagure, Akshay Sunil Jejurkar[1], Rutvik Bhagwan Patil[2], Mahesh Bhagwan Satpute[3], and Gaurav Somnath Sonawane[4]

1Student, K. K. Wagh Institute of Engineering Education and Research, Nashik, Maharashtra, India
akshayjejurkar2002@gmail.com, rutvikpatil938@gmail.com, msatpute825@gmail.com, gauraveducation2025@gmail.com

Abstract. The project proposes an accurate, timely and cost-effective OMR sheet evaluation system for attendance marking used in education system based on a low-priced web camera that can evaluate any format of folio paper efficiently. The OMR sheet is also designed and developed using several iteration techniques. The selected region of the sheet with all marked entries on the specified paper will be used as a template image in the matching process to extract the answered region of the student answer script. Then the cropped region of the sheet is matched with the template image to recognize the present and absent students. The Results obtained using different formats of MCQ papers are presented in this report. The main objective of this project is to develop Image processing based Optical Mark Recognition sheet scanning system. This system can be used in many entrance exams which used OMR sheet in competitive exams which comprise of MCQs. Summation of the present and absent students & displaying of total attendance will be also implemented. The implementation is done using Python or Java programming languages. We also have used GUI for storing the key for all the correct answers.

Keywords: OMR, Computer vision, Image processing, MCQ, Machine learningand OpenCV.

1. Introduction

An Optical Mark Reader (OMR) Sheet is a type of security document that features pre-printed paper with circular bubbles, timing tracks, and sensors. These bubbles are filled by individuals taking the exam and the timing tracks assist in the OMR Sheet's reading. The OMR Sheet for an exam includes personal details such as the candidate's name, date of birth, father's name, roll number, category, address, mobile number, email, date of the exam, barcode/QR code, and spaces for the candidate's answers.

In modern technology, candidates can mark their answers with ticks instead of filling in the bubbles, and the scanning software accepts both types of marks. OMR Sheets are used for administering exams at all levels of education, from kindergarten to professional exams.

Recently OMR Sheet has been used for election and attendance purpose.

Optical Mark Recognition (OMR), also called mark sensing, is a technique to sense the presence or absence of marks by recognizing their depth (darkness) on sheet. A mark is a response position on the questionnaire sheet that is filled with pencil or ballpoint pen. The way of marking is simple to everyone, and OMR device can process mark information on sheets rapidly. Thus, OMR has been widely used as a direct input device for data of censuses and surveys and is fit for handling discrete data, whose values fall into a limited number of values.

In the field of education, OMR technique is often used to process objective questions in the examination, such as College Boards Scholastic Aptitude Test (SAT), the Graduate Record Examination (GRE) in the United States, and the College English Test (CET) in China. However, there are a few distinct drawbacks which limit the application of OMR technology.

2. Literature Survey

Here, various papers describing various techniques for OMR detection system are studied**.**

Paper1-OMR Sheet Evaluation by Web Camera Using Template Matching Approach. Nalan Karunanayake, Department of Electrical & Computer Engineering Sri Lanka Institute of Information Technology Malabe, Sri Lanka nalan.k@sliit.lk

Summary: -

Optical Mark Recognition (OMR) is a process of capturing marked data such as crosses, ticks, and filled regions on printed papers like multiple-choice examinations (MCQs) and surveys.

Paper 2- OMR Sheet Recognition. Lakshay Bansal, Aditya Sood, Harsh Wani Under the guidance of Prof. Swarnalatha P. VIT University, Vellore, Chennai, Tamil Nadu, India.

Summary: -

A computer vision approach to evaluate OMR sheets using OPENCV libraries is presented in this work. The traditional method of using scanners for capturing OMR sheet images is replaced by using a webcam and a custom program.

Paper 3- Research on OMR Recognition Based on Convolutional Neural Network TensorFlow Platform. Y. Ju, X. Wang, and X. Chen, "Research on OMR Recognition Based Convolutional Neural Network TensorFlow Platform," 2019 11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), 2019, pp. 688-691, doi: 10.1109/ICMTMA.2019.00157.

Summary: -

The optical mark reader (OMR) recognition challenges and the complexity of virtual operations have led to increased human labour. To tackle these issues, a model architecture based on convolutional neural network (CNN) and TensorFlow platform has been proposed to improve OMR recognition technology.

Paper4-Image processing using python. Muhammad Arif Ridoy, International Journal of Scientific & Engineering Research Volume 9, Issue 3, March-2018 ISSN 2229-5518

Summary: -The scikit-image library has become increasingly popular for image processing tasks. Developed in Python, it aims to be user-friendly, efficient, and applicable to various scenarios. This research paper discusses the design choices made for the library's application programming interface (API).

3. Proposed Methodology

In this project, this system has been developed utilizing the Python programming language and several native libraries. It can be implemented on any operating system (OS) since the entire system has been programmed in Python, which is platform independent.

Developed Algorithm Flowchart

An algorithm has been developed to solve Detailed flowchart of algorithm has been illustrated in Fig.

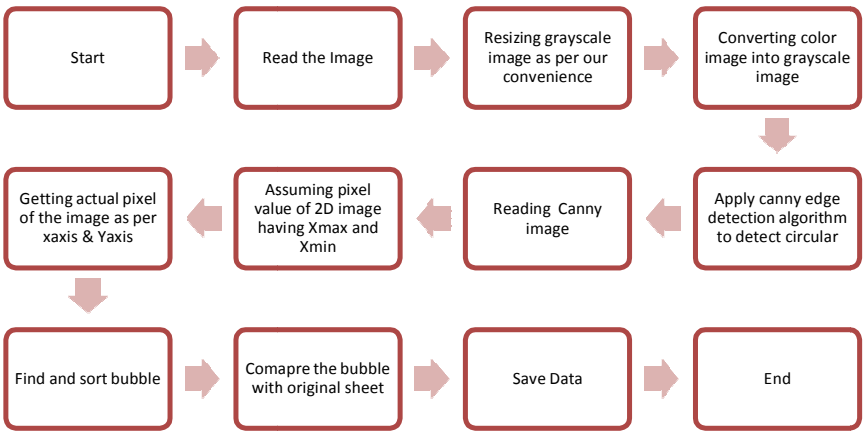


Fig.3.1 Algorithm Flowchart

Data extraction and analysis of sheet

In the concept of OMR sheet for image processing, several steps are involved to ensure accurate data extraction and analysis.

The OMR technique involves a series of five fundamental steps:

1. Sheet design: Construct the layout of the sheet.

2. Template creation: Generate a reference template for the marked areas.
3. 2D transformations: Apply transformations to the scanned image to ensure proper alignment and sizing.
4. Image processing: Employ image processing techniques to enhance the visibility of attendance markings.
5. Template matching: Utilize the template to determine the presence or absence of marks.

Design of Sheet

Although there are numerous pre-designed OMR forms available for use, designing your own forms can be straightforward. The fundamental principles involve creating a form that satisfies the following criteria:

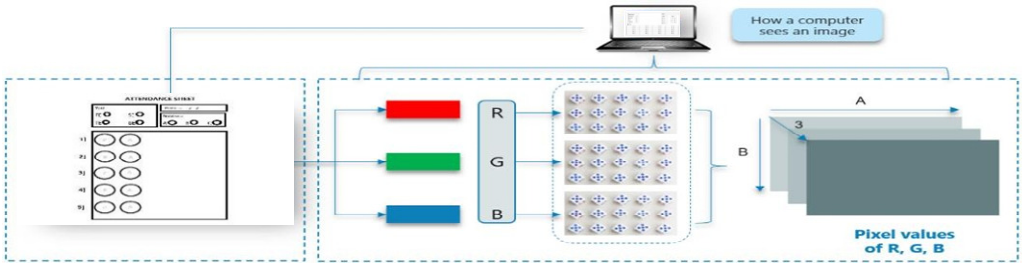


Fig.3.2 RGB to Grayscale image

It is simple for the software to swiftly align, shift, and adjust the scale for efficient readability. It is straightforward for the software to eliminate unimportant portions to simplify processing. Here as a simple example: -

When designing the form, the primary objective is to ensure that it is easily processed in the future. After scanning the form, it is critical to ensure that it is appropriately scaled and aligned with the template to match the scanned image. To aid in this process, I have inserted two timing circles, each half an inch in diameter, in the top-left and bottom-right margins at a fixed position. These circles will be simple to locate later on, and all template coordinates will be based on their location.

Consider fig. of the OMR sheet, upon first glance you will see color of image. The image is broken down into 3 color-channels which is Red, Green and Blue. Each of these color channels are mapped to the image's pixel. Then, the computer recognizes the value associated with each pixel and determine the size of image. However, for black-white images, there is only one channel, and the concept is the same.

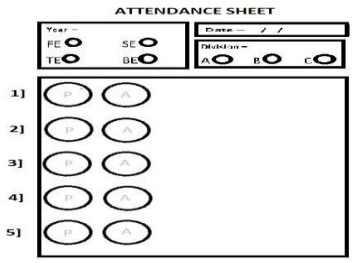


Fig 3.3 Layout of attendance sheet

When designing the form, the primary objective is to ensure that it is easily processed in the future. After scanning the form, it is critical to ensure that it is appropriately scaled and aligned with the template to match the scanned image. To aid in this process, I have inserted two timing circles, each half an inch in diameter, in the top-left and bottom-right margins at a fixed position. These circles will be simple to locate later on, and all template coordinates will be based on their location.

Process of Computer Read an Image

3.6 Source Code

1.Pro.py (main code)

```
import numpy as np
import cv2
import utlis
from tkinter import messagebox
import pandas as pd
import datetime
import csv
################################
path=("sheet8.jpeg")
widthImg = 300
heightImg = 300
img=cv2.imread(path)
questions = 5
choices = 5
ans = [0,0,0,0,0]
webcamFeed = True
cameraNo = 0
#######################
cap = cv2.VideoCapture(ipCam)
cap.set(10,150)
while True:
   if webcamFeed: succes, img = cap.read()
   else: img = cv2.imread(path)
   #PREPROCESSING
   img = cv2.resize(img,(widthImg,heightImg))
   imgContours = img.copy()
   imgFinal = img.copy()
   imgBiggestContours = img.copy()
   imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
   imgBlur = cv2.GaussianBlur(imgGray,(5,5),1)
   imgCanny = cv2.Canny(imgBlur,10,50)
 try:
 #FINDING ALL CONTOURS
      contours, hierarchy =
cv2.findContours(imgCanny,cv2.RETR_EXTERNAL,cv2.CHAI
N_APPROX_NONE)
      cv2.drawContours(imgContours,contours,-1,(0,255,0),10)
      #FIND RECTANGLES
      rectCon = utlis.rectCountour(contours)
      biggestContour = utlis.getCornerPoints(rectCon[0])
      gradePoints = utlis.getCornerPoints(rectCon[0])
      #print(biggestContour)
      if biggestContour.size != 0 and gradePoints.size !=0:
         cv2.drawContours(imgBiggestContours,biggestContour,-
1,(0,255,0),20)
         cv2.drawContours(imgBiggestContours,gradePoints,-
1,(255,0,0),20)
         biggestContours = utlis.reorder(biggestContour)
         gradePoints = utlis.reorder(gradePoints)
         pt1 = np.float32(biggestContours)
         pt2 =
np.float32([[0,0],[widthImg,0],[0,heightImg],[widthImg,hei
ghtImg]])
         matrix = cv2.getPerspectiveTransform(pt1,pt2)
         ImgWarpColored =
cv2.warpPerspective(img,matrix,(widthImg,heightImg))
         ptG1 = np.float32(gradePoints)
         ptG2 = np.float32([[0,0],[325,0],[0,150],[325,150]])
         matrixG = cv2.getPerspectiveTransform(ptG1,ptG2)
         ImgGradeDisplay =
cv2.warpPerspective(img,matrixG,(325,150))
         #cv2.imshow("Grade",ImgGradeDisplay)
         #APPLY THRESHOLD
         imgWarpGray =
cv2.cvtColor(ImgWarpColored,cv2.COLOR_BGR2GRAY
)
         imgThresh =
cv2.threshold(imgWarpGray,150,255,cv2.THRESH_BINA
RY_INV)[1]
         boxes= utlis.splitBoxes(imgThresh)
      #cv2.imshow("Test",boxes[2])
      #print(cv2.countNonZero(boxes[1]),cv2.countNonZer
o(boxes[2]))
      # GETTING NO ZERO PIXEL VALUES OF EACH
BOX
         myPixelVal = np.zeros((questions,choices))
         countC=0
         countR=0
         for image in boxes:
            totalPixels = cv2.countNonZero(image)
            myPixelVal[countR][countC] = totalPixels
            countC +=1
            if (countC == choices):countR +=1 ;countC=0
 # FINDING INDEX VALUES OF THE MARKING
         myIndex = []
                for x in range (0,questions):
 arr = myPixelVal[x]
            #print("arr",arr)
            myIndexVal = np.where(arr==np.amax(arr))
            #print(myIndexVal[0])
            myIndex.append(myIndexVal[0][0])
         #print (myIndex)
         k1 = []
         k2 = []
text = { }
         for i in range(1, len(myIndex)+1):
```

```
            kk = ""
            if myIndex[i-1] == 1:
                kk = "Absent"
            else:
                kk = "Present"
            k1.append(kk)
            k2.append(i)

            print("Student " + str(i) + " is " + str(kk))
        for key, value in zip(k2, k1):
            text[key] = value
        print(text)
        df = pd.DataFrame(list(text.items()), columns=["Roll
No", "Attendance"])
        df.to_excel('attendance_report.xlsx', index=False)
        messagebox.showinfo('Attendance System', 'Attendance
report generated successfully.')
        # GRADING
        grading =[]
        for x in range (0,questions):
            if ans[x] == myIndex[x]:
                grading.append(1)
            else:
                grading.append(0)
        #print(grading)
        score = (sum(grading)/questions)*100 #FINAL GRADE
        print("present student =",int(score/20))
        print("Absent student =",int((100-score)/20))
        # DISPLAYING ANSWERS
        imgResult = ImgWarpColored.copy()
        imgResult =
utlis.showAnswers(imgResult,myIndex,grading,ans,questions,ch
oices)
imgRawDrawing = np.zeros_like(ImgWarpColored)
          imgRawDrawing =
utlis.showAnswers(imgRawDrawing,myIndex,grading,ans,questi
ons,choices)
invMatrix = cv2.getPerspectiveTransform(pt2,pt1)
 imgInvWarp =
cv2.warpPerspective(imgRawDrawing,invMatrix,(widthImg,heig
htImg))

        # DISPLAYING GRADE
        imgRawGrade = np.zeros_like(ImgGradeDisplay)
        cv2.putText(imgRawGrade,str(int(score))+"%",(50,100),c
v2.FONT_HERSHEY_SIMPLEX,2,(255,0,0),6)
        invMatrixG = cv2.getPerspectiveTransform(ptG2,ptG1)
        imgInvGradeDisplay =
cv2.warpPerspective(imgRawGrade,invMatrixG,(widthIm
g,heightImg))

        imgFinal =
cv2.addWeighted(imgFinal,1,imgInvWarp,1,0)
        imgFinal =
cv2.addWeighted(imgFinal,1,imgInvGradeDisplay,1,0)
    imgBlank = np.zeros_like(img)
    imageArray = ([img,imgGray,imgBlur,imgCanny],
            [imgContours,imgBiggestContours,ImgWarp
Colored,imgThresh ],
            [imgResult,imgRawDrawing,imgInvWarp,im
gFinal])

  except:
    imgBlank = np.zeros_like(img)
    imageArray = ([img,imgGray,imgBlur,imgCanny],
            [imgBlank,imgBlank,imgBlank,imgBlank ],
            [imgBlank,imgBlank,imgBlank,imgBlank])


  lables = [["Original","Gray","Blur","Canny"],
      ["contours","Bigger Con","Warp","Threshold"],
      ["Result","Raw Drawing","Inv Warp","Final"]]

  imgStacked = utlis.stackImages(imageArray,0.6,lables)

  cv2.imshow("Final Result",imgFinal)
  cv2.imshow("Stacked Images",imgStacked)
  cv2.waitKey(0)

  if cv2.waitKey(1) & 0xFF == ord('s'):
    break
    cv2.imwrite("FinalResult.jpg",imgFinal)
    cv2.waitKey(300)
```

Utilis.py (sub code)

```
import cv2
import numpy as np
from tkinter import messagebox
import pandas as pd
def stackImages(imgArray,scale,lables=[]):
  rows = len(imgArray)
  cols = len(imgArray[0])
  rowsAvailable = isinstance (imgArray[0],list)
  width = imgArray[0][0].shape[1]
  height = imgArray[0][0].shape[0]
  if rowsAvailable:
    for x in range ( 0, rows):
      for y in range(0, cols):
        imgArray[x][y]= cv2.resize(imgArray[x][y],(0, 0),
None, scale, scale)
        if len(imgArray[x][y].shape) == 2: imgArray[x][y]=
cv2.cvtColor( imgArray[x][y], cv2.COLOR_GRAY2BGR)
    imageBlank = np.zeros((height, width, 3), np.uint8)
    hor = [imageBlank]*rows
    hor_con= [imageBlank]*rows
    for x in range(0, rows):
      hor[x] = np.hstack(imgArray[x])
      hor_con[x] = np.concatenate (imgArray[x])
    ver = np.vstack(hor)
    ver_con = np.concatenate(hor)
  else:
    for x in range(0, rows):
      imgArray[x] = cv2.resize(imgArray[x],(0, 0), None, scale,
scale)
      if len(imgArray[x].shape) == 2: imgArray[x] =
cv2.cvtColor(imgArray[x], cv2.COLOR_GRAY2BGR)
    hor= np.hstack(imgArray)
    hor_con= np.concatenate(imgArray)
    ver = hor
  if len(lables) != 0:
    eachImgWidth= int(ver.shape[1] / cols)
    eachImgHeight= int(ver.shape[0] / rows)
    #print(eachImgHeight)
    for d in range(0, rows):
      for c in range (0,cols):
        cv2.rectangle(ver, (c*eachImgWidth,
eachImgHeight*d),(c*eachImgWidth+len(lables[d][c])*13+27,3
0+eachImgHeight*d),(255,255,255),cv2.FILLED)
        cv2.putText(ver,lables[d][c], (eachImgWidth*c+10,
eachImgHeight*d+20),cv2.FONT_HERSHEY_COMPLEX,0.7,(
255,0,255),2)
  return ver
def rectCountour(coutours):
  rectCon = []
  for i in coutours:
    area= cv2.contourArea(i)
    #print("Area",area)
    if area>50:
      peri = cv2.arcLength(i,True)
      approx = cv2.approxPolyDP(i,0.02*peri,True)
      #print("Corner Points",len(approx))
      if len(approx)==4:
        rectCon.append(i)
  rectCon = sorted(rectCon,key=
cv2.contourArea,reverse=True)
  return rectCon
def getCornerPoints(cont):
  peri = cv2.arcLength(cont, True)
  approx = cv2.approxPolyDP(cont,0.02 * peri,True)
  return approx
def reorder(myPoints):
  myPoints = myPoints.reshape((4,2))
  myPointsNew = np.zeros((4,1,2),np.int32)
  add = myPoints.sum(1)
  #print(myPoints)
  #print(add)
  myPointsNew[0] = myPoints[np.argmin(add)] #[0,0]
  myPointsNew[3] = myPoints[np.argmax(add)] #[w,h]
  diff = np.diff(myPoints,axis=1)
  myPointsNew[1] = myPoints[np.argmin(diff)] #[w,0]
  myPointsNew[2] = myPoints[np.argmax(diff)] #[h,0]
  #print(diff)
  return myPointsNew
def splitBoxes(img):
  rows =np.vsplit(img,5)
  boxes=[]
for r in rows:
    cols=np.hsplit(r,5)
    for box in cols:
      boxes.append(box)
      #cv2.imshow("split",box)
  return boxes
def
showAnswers(img,myIndex,grading,ans,questions,choices):
  secW = int(img.shape[1]/questions)
  secH = int(img.shape[0]/choices)
```

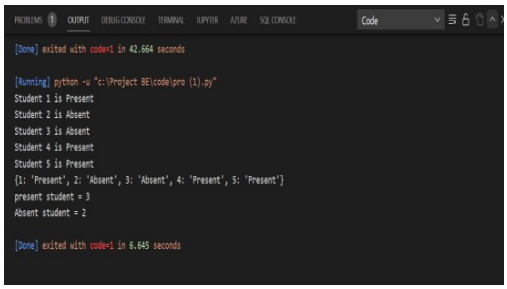
```
for x in range (0,questions):
        myAns = myIndex[x]
        cX = (myAns*secW)+secW//2
        cY = (x*secH) + secH//2
        if grading[x] ==1:
            myColor = (0,255,0)
        else:
            myColor = (0,0,255)
            correctAns = ans[x]
            cv2.circle(img,((correctAns*secW)+secW//2,(x*secH)+
secH//2),20, (0,255,0), cv2.FILLED)
        cv2.circle(img,(cX,cY),20, myColor, cv2.FILLED)
    return img
def generate_report(text):
    lines = text.split('\n')
    headers = ['Roll No.', 'Attendance']
    data = []
    for line in lines:
        if line.startswith('Roll No'):
            continue
        parts = line.split()
        if len(parts) < 3:
            continue
        name = ' '.join(parts[:-2])
        roll_no = parts[-2]
        attendance = parts[-1]
        data.append([name, roll_no, attendance])
    df = pd.DataFrame(data, columns=headers)
    df.to_excel('attendance_report.xlsx', index=False)
    messagebox.showinfo('Attendance System', 'Attendance
report generated successfully.')
    messagebox.showinfo('Attendance System', 'Attendance
report generated successfully.')
```

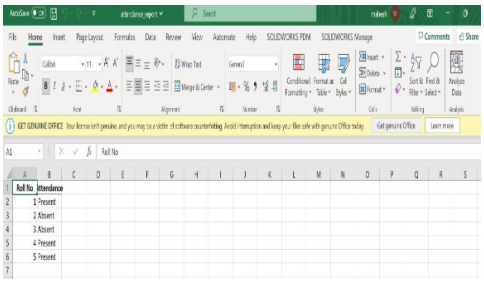
3.8 Result

Output

By employing the mobile camera, we capture an image that is subsequently uploaded to the source code. The program then scans the image and compiles an attendance report in an Excel sheet, indicating the presence or absence of each student based on their respective roll numbers. Additionally, the output image displays the total number of students present and absent in the class.

Terminal





Excel Sheet Report

Here, we create report in excel sheet. In excel sheet have two columns. First column contains roll number of student and second column contain attendance of student.

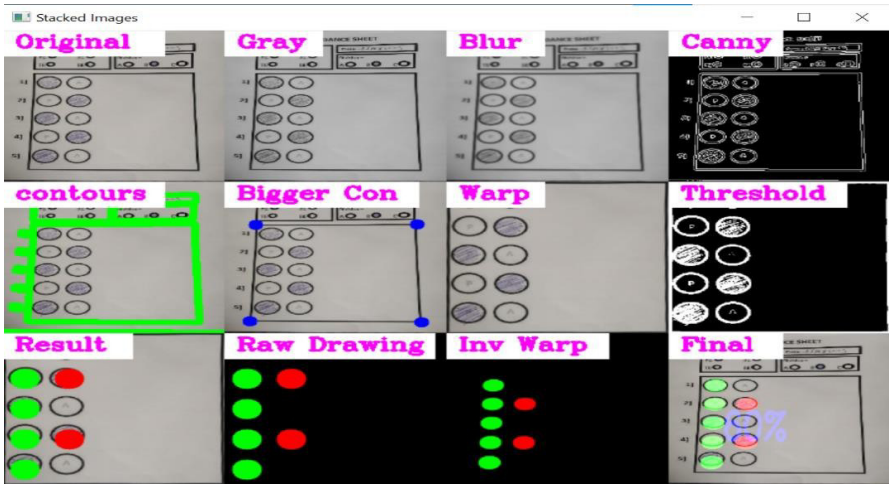


Fig. Stacked image result

Results and Graph:

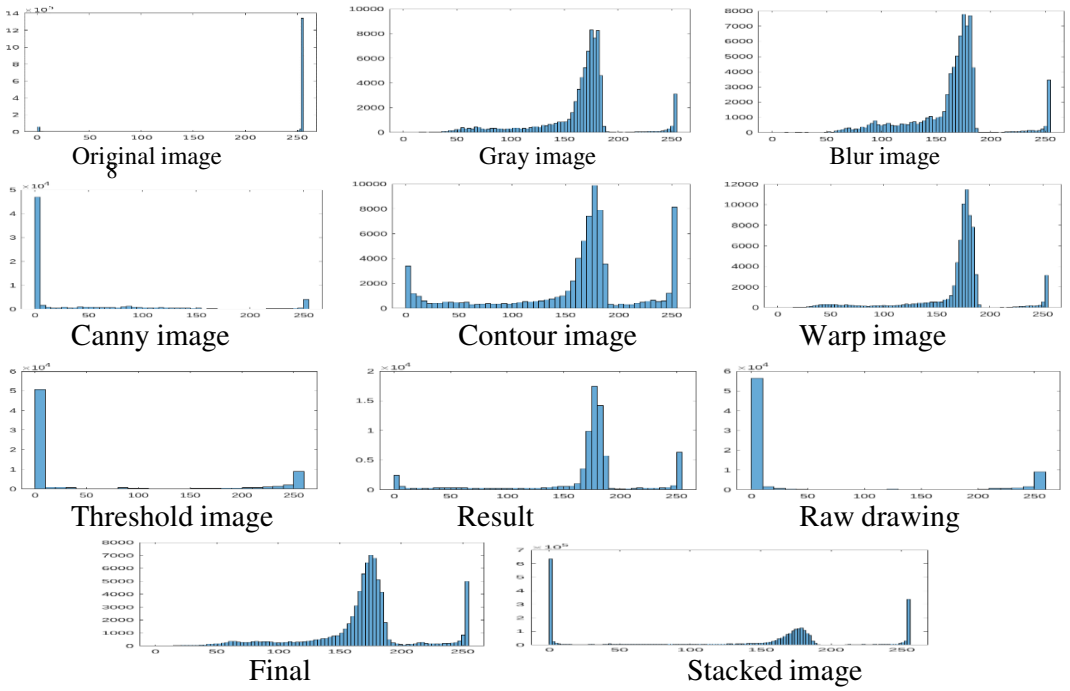


Fig. Histogram of each image processing

4. Future Scope

1. **Automation in Education:** OMR sheet detection technology has found extensive usage in the field of education, particularly for assessing multiple-choice examinations.
2. **Healthcare:** OMR sheet detection technology has the potential to be utilized in the healthcare sector for the purpose of gathering and analyzing data. It can aid in collecting data on patients' medical history, symptoms, and treatment strategies, and then scrutinize the data to uncover patterns and trends.
3. **Elections:** OMR sheet detection technology is applicable in election procedures for scanning and tallying votes. By doing so, the technology would alleviate the burden on election officials and enhance the precision of the results.

Overall, OMR sheet detection technology has several potential future applications in various industries.

5. Conclusion

The purpose of this project is to create an application that utilizes OMR (Optical Mark Recognition) technology to extract attendance information from students using a regular scanner. The user can then print out as many copies as needed, distribute them to the relevant parties, and scan the completed sheets. The scanned image files will serve as input to the software, which will process them, extract the attendance values, and manipulate the data according to the user's instructions. Finally, the data will be stored in a database for future reference.

References

1. Journal Article

1) N. Karunanayake, “OMR sheet evaluation by Web camera using Template Matching approach.
2) Lakshay Bansal et al. “OMR sheet Recognition”.
3) Xichang Wang et al. “Research on OMR Recognition Based on Convolutional Neural Network Tensorflow Platform”.
4) Muhammad Arif Ridoy et al. “Image processing using python”.
5) Abdullah erdal tümer et al. “An image processing oriented optical mark recognition and evaluation system”.