FABRIC DEFECT DETECTION SYSTEM

Along with quantitative production quality production is also very much important in any industry. In textile industry sometime the clothes get defected during the manufacturing process due to various region. Previously the defect detection is done by human eye, in which error is possible and the process was slow because human eye cannot detect each and every error with accuracy and speed. With the increase in population and modern lifestyle of people the textile industry has to increase the production with correct quality check methods because a single defect can reduce the price of product from 45-60%.

So, to deal with this problem a computer based automatic fabric defect detection system was introduced, which will increase the rate of production of fabric and decrease the production of defect product. There is 23 different type of defect that can be there in a fabric e.g. shade variation, stains, knots, vertical & horizontal holes, needle line, broken or missing end, out of which 10 types of defect can be detected by the method we used in this project.

In this project we were be able to find the defect that is caused by holes and needle line defect by EDGE DETECTION method, with the help of CANNY FILTER and IMAGE PROCESING in OPEN CV 2 simulated in microcontroller (Raspberry pi).

INTRODUCTION

A fabric is a textile material that human being uses in their day to day life. The defect on the textile is a result of flaw in manufacturing process as well as the fabric inspection process. So, fabric inspection process or we can say fabric defect detection is very important for the textile industry, so that the product received by the customer is defect free. In the recent year of technological advancement automatic fabric defect detection is developed and it has sidelined the traditional process which is performed by humans. Generally, the automatic defect detection process involves three steps: image acquisition, defect detection and post processing of fabric.

WORKING

The main purpose of the project was to detect the defects in fabric as and when they occur. The basic idea that lead us to the project was the loss incurred by the fabric manufacturers due to trivial mistakes or lack of alertness.

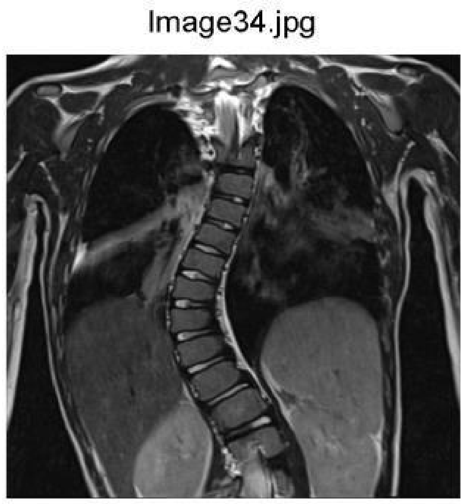
In our project we are viewing the cuts or the defects in fabric as a window of edge creation which needs to be detected by the camera it passes through. Our working model comprises of a conveyer belt type structure on which the fabric moves with a camera being at the center of it all to detect all kinds of fault occurring in the fabric. We are considering each frame of the image taken by the camera and converting it into grey scale for processing purposes. We then by the help of our microcontroller (Raspberry pi) process that image and pass it through canny filter to detect any edge present in the fabric. When no edges are detected, the motor continues to move as usual with the green LED still glowing while as soon as an edge is detected, our code asks the motor to stop and gives a red LED and a buzzer to let the people know of defects.

Code:

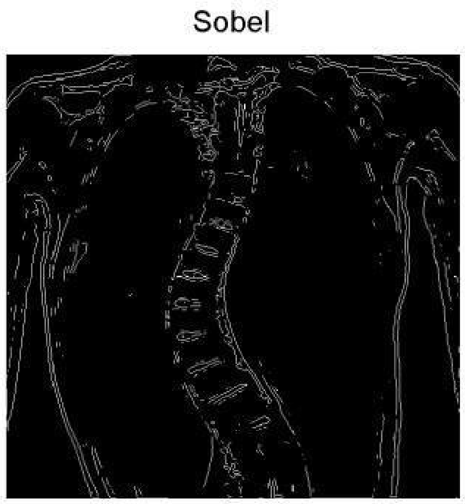
```
# import libraries of python OpenCV #

where its functionality resides import

cv2

import RPi.GPIO as GPIO

import time

from time import sleep

# np is an alias pointing to numpy library

import numpy as np

in1 = 15

in2 = 18 GPIO.setmode(GPIO.BOARD)

#GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(12,GPIO.OUT)
GPIO.setup(11,GPIO.OUT)
GPIO.setup(in1, GPIO.OUT)
GPIO.setup(in2, GPIO.OUT)

GPIO.output(in1, True)
GPIO.output(in2, True)
GPIO.output(11,GPIO.HIGH)
GPIO.output(12,GPIO.LOW)
# capture frames from a camera cap
= cv2.VideoCapture(0)
# loop runs if capturing has been initialized
while(1):
    # reads frames from a camera ret,
    frame = cap.read()
    # converting BGR to HSV
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) #
    define range of red color in HSV
    lower_red = np.array([30,150,50])
    upper_red = np.array([255,255,180])
    # create a red HSV colour boundary and #
    threshold HSV image
    mask = cv2.inRange(hsv, lower_red, upper_red) #
    Bitwise-AND mask and original image
    res = cv2.bitwise_and(frame,frame, mask= mask) #
    Display an original image
    cv2.imshow('Original',frame)
    # finds edges in the input image image and #
    marks them in the output map edges edges =
    cv2.Canny(frame,100,200)
    #to see if the matrix is all zeros
    canny_arr = np.array(edges)
    canny_flat = canny_arr.flatten()
    if np.count_nonzero(canny_flat)>50:
        print(np.count_nonzero(canny_flat))
        #print("LED on")
        GPIO.output(11,GPIO.LOW)
        GPIO.output(12,GPIO.HIGH)
        print("Damaged")
        print ("STOP")
        GPIO.output(in1, True)
        GPIO.output(in2, True)
        #print("stop")
        cv2.imwrite("Error.png",frame)
        break
    else:
        GPIO.output(11,GPIO.HIGH)
        GPIO.output(12,GPIO.LOW)
        print(np.count_nonzero(canny_flat))
        #print("LED off")
        GPIO.output(in2, False)
        GPIO.output(in1, False)
        print("Continue")
        #print("start")
    # Display edges in a frame
    cv2.imshow('Edges',edges)

    # Wait for Esc key to stop k
    = cv2.waitKey(5) & 0xFF if
    k == 27:
        GPIO.output(11,GPIO.LOW)
        GPIO.output(12,GPIO.LOW)
        GPIO.cleanup()
        break
# Close the window
```

```
cap.release()


# De-allocate any associated memory usage
cv2.destroyAllWindows()
```

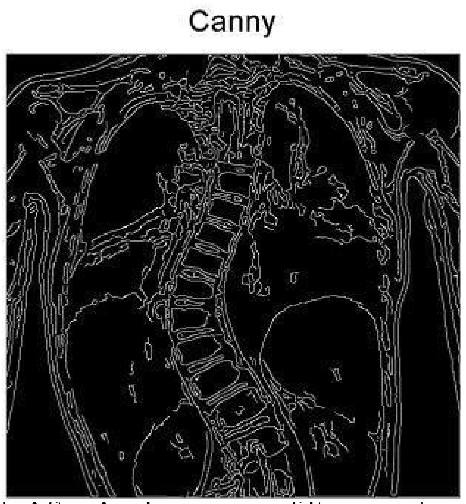
<u>Following are the images depicting outputs when a different edge detection is implemented on them:</u>



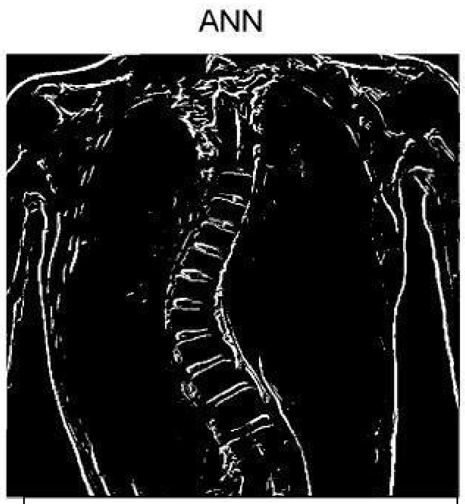
Original



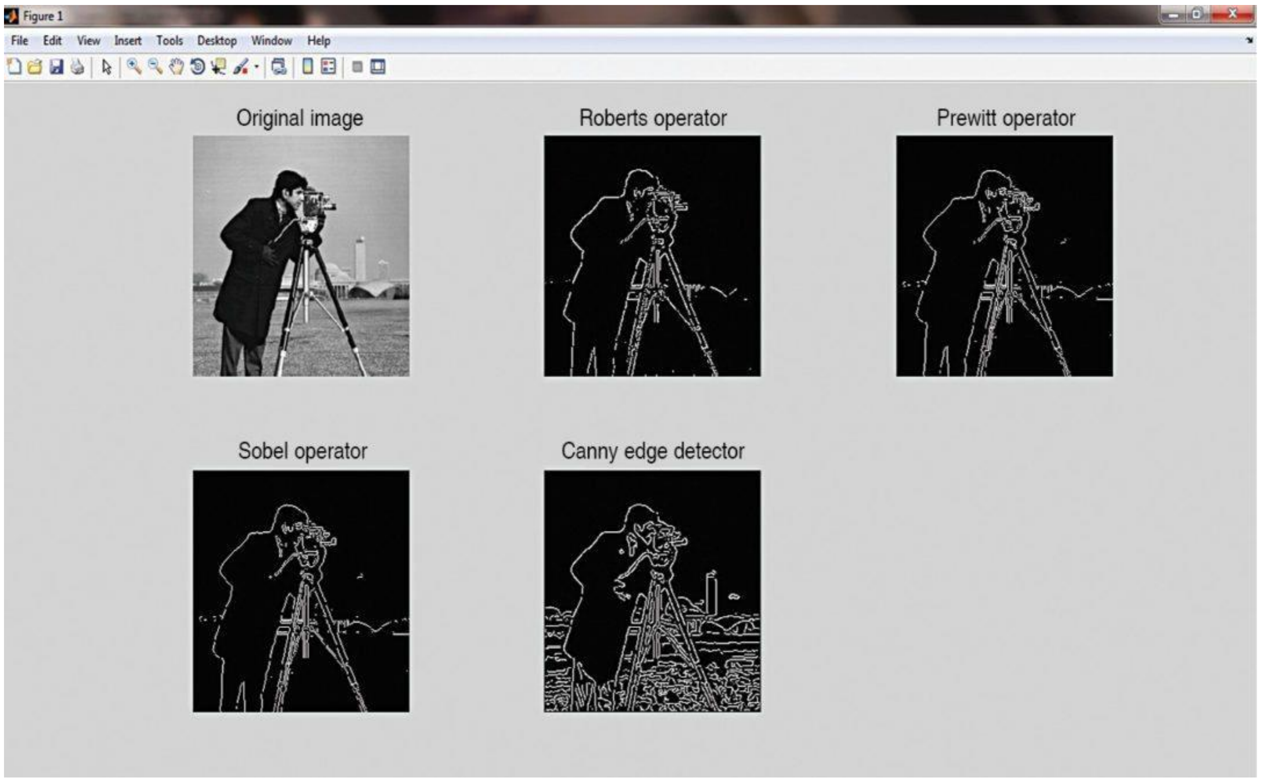
Applying sobel



After Applying canny filter



After Applying ANN filter



<u>Prototype of Fabric Defect detection System</u>

Video link of the working prototype
https://www.youtube.com/watch?v=JPnzcm21Uh8

Image of the working prototype: