# LAMPIRAN

Program Spyder, Coding Haar cascade classifier dan Interface

```
"""
Match setup
"""
import cv2
import numpy as np
from matplotlib import pyplot as plt
import os, errno
from xml.dom import minidom
import os

threshold = 0.87 #set threshold
resultsDirectory = 'results'
sourceDirectory = os.fsencode('sourceImages')
templateDirectory = os.fsencode('templates')
detectedCount = 0
Detected_BadProduct = 0



"""
    Realtime setup
"""
import PIL
from PIL import Image,ImageTk
```

```
import pytesseract
from tkinter import *
import tkinter as tk
import ResizeImage
from tkinter import filedialog
"""
    Variable
"""
frames = ''
reverence_name = ''
flow = 'cap1'

def save_source(event):
    global framester
    global reverence_name
    global flow

    filename = "temp/result.jpg"
    scale_percent = 200 # percent of original size
    width = int(frames.shape[1] * scale_percent / 100)
    height = int(frames.shape[0] * scale_percent / 100)
    dim = (width, height)
    # resize image
    resized = cv2.resize(frames, dim, interpolation = cv2.INTER_AREA)
    cv2.imwrite(filename, resized)
```

```
    match(filename)


def load_image_source(event):
    link = filedialog.askopenfilename()
    match(link)


def show_frame():
    global frames
    st, frame = cap.read()
    if st:
        frame = cv2.flip(frame, 1)
        frames = frame
        cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
        img = PIL.Image.fromarray(cv2image)
        imgtk = ImageTk.PhotoImage(image=img)
        lmain.imgtk = imgtk
        lmain.configure(image=imgtk)
        lmain.after(10, show_frame)
    else:
        print("Camera False")


def match(image_1):
    global detectedCount
    global Detected_BadProduct
```

```
img_rgb = cv2.imread(image_1)
img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)

for templateFile in os.listdir(templateDirectory):
    templateFilename = os.fsdecode(templateFile)
    template = cv2.imread('templates/'+templateFilename,0)
    w, h = template.shape[::-1]
    res = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED)
    loc = np.where( res >= threshold)

    if (len(loc[0])):
        detectedCount = detectedCount + 1
        for pt in zip(*loc[::-1]):
            cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)
            Detected_BadProduct = Detected_BadProduct + 1

        print ('detected positive ' + str(detectedCount))
        print ('detected bad product ' + str(Detected_BadProduct))
    labelCnn.config(text=str(detectedCount))
cv2.imwrite('temp/result.jpg', img_rgb)
# resized = cv2.resize('temp/result.jpg', (352, 288))
gmb = ResizeImage.center("temp/result.jpg", 652, 588)
lbl_reverence.config(image=gmb)
lbl_reverence.image = gmb
```

```
  # cv2.waitKey(0)


"""
   Video setup
"""
width, height = 320, 320
print(cv2.CAP_DSHOW)
cap = cv2.VideoCapture(1 + cv2.CAP_DSHOW)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)


"""
   GUI setup
"""
root = tk.Tk()
root.wm_title("PCB DEFECT DETECTION  USING HAAR CASCADE")
root.geometry("1035x600") # 1475x630 1366x768


"""
   Realtime view
"""
leftFrameRealtime = tk.Frame(root, width=365, height=672, bg="#b8e2ff")
leftFrameRealtime.place(x=0, y=0)
lmain = tk.Label(leftFrameRealtime)
lmain.place(x=5, y=5)
```

```
labelRealtime = tk.Label(leftFrameRealtime, text="REALTIME", font = ("Arial", 14), bg="#b8e2ff")

labelRealtime.place(x=135, y=295)


labelRealtime = tk.Label(leftFrameRealtime, text="Capture Object", font = ("Arial", 14), bg="#b8e2ff")

labelRealtime.place(x=5, y=335)

Bsource = tk.Button(leftFrameRealtime, text="Capture", font = ("Arial", 12), height=1, width=9, bg="white", fg="black")

Bsource.place(x=180, y=333)

Bsource.bind("<Button-1>", save_source)


labelRealtime = tk.Label(leftFrameRealtime, text="Search Object", font = ("Arial", 14), bg="#b8e2ff", fg="grey")

labelRealtime.place(x=5, y=375)

BsearchSource = tk.Button(leftFrameRealtime, text="Search", font = ("Arial", 12), height=1, width=9, bg="grey", fg="white")

BsearchSource.place(x=180, y=373)

BsearchSource.bind("<Button-1>", load_image_source)


labelNotice = tk.Label(leftFrameRealtime, text="", font = ("Arial", 14), bg="#b8e2ff", fg="red")

labelNotice.place(x=5, y=650)



"""
```

Label

```
"""
labelDeteksi = tk.Label(leftFrameRealtime, text="Hasil Deteksi", font = ("Arial", 14), bg="#b8e2ff")
labelDeteksi.place(x=5, y=445)
labelCnn = tk.Label(leftFrameRealtime, text="...", font = ("Arial", 14), bg="#b8e2ff")
labelCnn.place(x=125, y=445)



"""
    Show image source.jpg
"""
rightFrameSource = tk.Frame(root, width=1105, height=640, bg="#b8e2ff")
rightFrameSource.place(x=370, y=0)
gmb_conv = ResizeImage.center("temp/first.png", 652, 588)
lbl_reverence = tk.Label(rightFrameSource, image = gmb_conv, width = 652, height = 588, bd = 0, bg="#8fd1ff")
lbl_reverence.place(x = 10, y = 5)



def main():
    show_frame()
    root.mainloop()

if __name__ == "__main__":
    main()
```