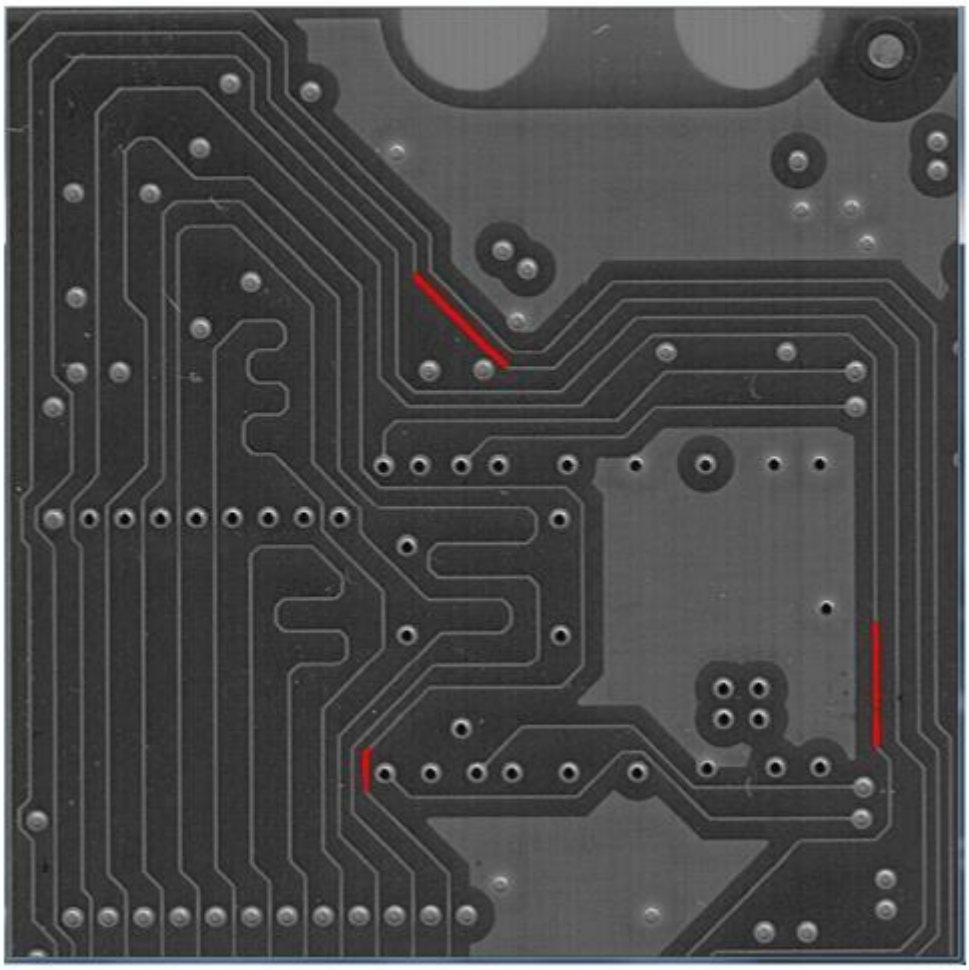
INSTITUTO FEDERAL
São Paulo
Câmpus Birigui

PROCESSAMENTO DIGITAL DE IMAGENS

Carlos Vinicius Baggio Savian, BI3002217

Desafio Stanford

Exercício 1) Verificação de defeitos em placas: Basicamente realizando uma operação de subtração entre uma imagem de uma placa sem defeito com uma placa com defeito é possível encontrar defeitos no processo de fabricação.



```
import cv2
import numpy as np
def carregar_imagens(placa_sem_defeito_path, placa_com_defeito_path):
placa_sem_defeito = cv2.imread(placa_sem_defeito_path)
placa_com_defeito = cv2.imread(placa_com_defeito_path)
return placa_sem_defeito, placa_com_defeito
def converter_para_tons_de_cinza(imagem):
return cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
def encontrar_diferencas(imagem1, imagem2):
return cv2.absdiff(imagem1, imagem2)
def destacar_diferencas(diferenca, limite=30):
_, diferenca_binaria = cv2.threshold(diferenca, limite, 255,
cv2.THRESH_BINARY)
return diferenca_binaria
def encontrar_contornos(imagem_binaria):
contornos, _ = cv2.findContours(imagem_binaria, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
return contornos
def desenhar_contornos(imagem, contornos):
imagem_com_contornos = imagem.copy()
cv2.drawContours(imagem_com_contornos, contornos, -1, (0, 0, 255), 2)
return imagem_com_contornos
if __name__ == "__main__":
placa_sem_defeito_path = 'pcbCroppedTranslated.png'
placa_com_defeito_path = 'pcbCroppedTranslatedDefected.png'
placa_sem_defeito, placa_com_defeito =
carregar_imagens(placa_sem_defeito_path, placa_com_defeito_path)
if placa_sem_defeito is None or placa_com_defeito is None:
print("Erro ao carregar as imagens.")
else:
placa_sem_defeito_gray =
converter_para_tons_de_cinza(placa_sem_defeito)
placa_com_defeito_gray =
converter_para_tons_de_cinza(placa_com_defeito)
diferenca = encontrar_diferencas(placa_sem_defeito_gray,
placa_com_defeito_gray)
limite = 30
diferenca_binaria = destacar_diferencas(diferenca, limite)
contornos = encontrar_contornos(diferenca_binaria)
placa_com_defeito_com_contornos =
desenhar_contornos(placa_com_defeito, contornos)
cv2.imshow('Placa sem defeito', placa_sem_defeito)
cv2.imshow('Placa com defeito', placa_com_defeito_com_contornos)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Exercício 2) Detecção de movimento: A partir de um vídeo, ao realizar a subtração do fundo da cena sem nenhuma pessoa é possível detectar movimentos.

(Detectando)


```
import cv2
# Inicializar a captura de vídeo
cap = cv2.VideoCapture('output.avi') # Substitua 'seu_video.mp4' pelo
caminho do seu vídeo
# Inicializar o modelo de subtração de fundo
fgbg = cv2.createBackgroundSubtractorMOG2()
while True:
# Ler o próximo quadro do vídeo
ret, frame = cap.read()
if not ret:
break
# Aplicar a subtração de fundo no quadro atual
```

```
fgmask = fgbg.apply(frame)
# Aplicar uma operação de limiarização para segmentar os objetos em
movimento
threshold = 50 # Ajuste esse valor conforme necessário
_, thresh = cv2.threshold(fgmask, threshold, 255, cv2.THRESH_BINARY)
# Encontrar contornos dos objetos em movimento
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
# Desenhar retângulos ao redor dos objetos em movimento
for contour in contours:
if cv2.contourArea(contour) > 100: # Ajuste essa área mínima
conforme necessário
x, y, w, h = cv2.boundingRect(contour)
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
# Mostrar o quadro resultante
cv2.imshow('Detecção de Movimento', frame)
# Pressione a tecla 'Esc' para sair do loop
if cv2.waitKey(30) & 0xFF == 27:
break
# Liberar a captura de vídeo e fechar todas as janelas
cap.release()
cv2.destroyAllWindows()
```