import cv2 as cv

import matplotlib

import matplotlib.pyplot as plt

import numpy as np

from google.colab.patches import cv2\_imshow

img=cv.imread('dhoni.jpg')

rows,cols=img.shape[:2]

src = np.float32([[0,0],[cols-1,0],[0,rows-1],[cols-1,rows-1]])

dst = np.float32([[0,0],[cols-1,0],[int(0.33\*cols),rows-1],[int(0.66\*cols),rows-1]])

perspective\_matrix=cv.getPerspectiveTransform(src,dst)

out=cv.warpPerspective(img,perspective\_matrix,(cols,rows))

cv2\_imshow(img)

cv2\_imshow(out)

Graphical user interface, text, application, chat or text message

Description automatically generated

A person wearing a blue uniform

Description automatically generated with low confidence

A screenshot of a video game

Description automatically generated

#Affine transformation

pts1=np.float32([[50,50],[200,50],[50,200]])

pts2=np.float32([[10,100],[200,50],[100,250]])

M=cv.getAffineTransform(pts1,pts2)

img\_afftran=cv.warpAffine(img,M,(cols,rows))

import math

#vertical wave

img\_output=np.zeros(img\_afftran.shape,dtype=img.dtype)

for i in range(rows):

  for j in range(cols):

    offset\_x=int(25.0\*math.sin(2\*3.14\*i/180))

    offset\_y=0

    if j+offset\_x < rows:

      img\_output[i,j]=img\_afftran[i,(j+offset\_x)%cols]

    else:

      img\_output[i,j]=0

cv2\_imshow(img\_output)

Graphical user interface, text, application

Description automatically generated

A picture containing text, doing, male

Description automatically generated

#horizontal wave

img\_output=np.zeros(img\_afftran.shape,dtype=img.dtype)

for i in range(rows):

  for j in range(cols):

    offset\_x=0

    offset\_y=int(25.0\*math.cos(2\*3.14\*i/180))

    if j+offset\_y < cols:

      img\_output[i,j]=img\_afftran[(i+offset\_y)%rows,i]

    else:

      img\_output[i,j]=0

cv2\_imshow(img\_output)

Graphical user interface, text, application

Description automatically generated

Background pattern

Description automatically generated with medium confidence

import cv2

img=cv2.imread('BW.jpg',0)

cv2\_imshow(img)

kernel=np.ones((5,5),np.uint8)

erosion=cv2.erode(img,kernel,iterations=1)

cv2\_imshow(erosion)

A picture containing text

Description automatically generated

A lion lying down

Description automatically generated with medium confidence

dilation=cv2.dilate(img,kernel,iterations=1)

cv2\_imshow(dilation)



cv2\_imshow(img)

kernel=np.ones((3,3),np.uint8)

A lion lying down

Description automatically generated with low confidence

img=cv.imread('A.webp')

cv2\_imshow(img)

kernel=np.ones((3,3),np.uint8)

opening=cv2.morphologyEx(img,cv2.MORPH\_OPEN,kernel)

cv2\_imshow(opening)

closing=cv2.morphologyEx(img,cv2.MORPH\_CLOSE,kernel)

cv2\_imshow(closing)

Graphical user interface

Description automatically generated with medium confidence

A picture containing text

Description automatically generated

import cv2

from google.colab.patches import cv2\_imshow

# Read the original image

img = cv2.imread('dhoni.jpg')

# Display original image

cv2\_imshow(img)

# Convert to graycsale

img\_gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Canny Edge Detection

edges = cv2.Canny(img\_gray, 10,100) # Canny Edge Detection

# Display Canny Edge Detection Image

cv2\_imshow(edges)

# find the contours in the edged image

contours, hierarchy = cv2.findContours(edges, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE)

#Find Number of Contours

print("Number of Contours is: "+str(len(contours)))

#draw yellow border around the contours

cv2.drawContours(img, contours, 0, (0, 230, 255), 6)

cv2.drawContours(img, contours, 2, (0, 230, 255), 6)

#show the image with Contours

cv2\_imshow(img)

Text

Description automatically generated

Shape

Description automatically generated

img = cv2.imread('CC.jfif')

# Display original image

cv2\_imshow(img)

# Convert to graycsale

img\_gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

ret, thresh = cv2.threshold(img\_gray, 255, 255, cv2.THRESH\_BINARY\_INV)

cv2\_imshow(thresh)

contours, hierarchy = cv2.findContours(image=thresh, mode=cv2.RETR\_TREE,

                                       method=cv2.CHAIN\_APPROX\_SIMPLE)

# draw all contours on the original image

cv2.drawContours(img, contours=contours, contourIdx=-1,

                 color=(0, 255, 0), thickness=2, lineType=cv2.LINE\_AA)

cv2\_imshow(img)

Graphical user interface, text

Description automatically generated

Shape

Description automatically generated

img = cv2.imread('CC.jfif',0)

img2=img.copy()

template=cv2.imread('dhoni.jpg',0)

cv2\_imshow(template)

w,h = template.shape[::-1]

#ALl 6 methods for comparision in list

methods=['cv2.TM\_CCOEFF','cv2.TM\_CCOEFF\_NORMED','cv2.TM\_CCORR','cv2.TM\_CCORR\_NORMED','cv2.TM\_SQDIFF','cv2.TM\_SQDIFF\_NORMED']

for meth in methods:

    img = img2.copy()

    method = eval(meth)

    # Apply template Matching

    res = cv2.matchTemplate(img,template,method)

    min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(res)

    # If the method is TM\_SQDIFF or TM\_SQDIFF\_NORMED, take minimum

    if method in [cv2.TM\_SQDIFF, cv2.TM\_SQDIFF\_NORMED]:

        top\_left = min\_loc

    else:

        top\_left = max\_loc

    bottom\_right = (top\_left[0] + w, top\_left[1] + h)

    cv2.rectangle(img,top\_left, bottom\_right, 255, 2)

    plt.subplot(121),plt.imshow(res,cmap = 'gray')

    plt.title('Matching Result'), plt.xticks([]), plt.yticks([])

    plt.subplot(122),plt.imshow(img,cmap = 'gray')

    plt.title('Detected Point'), plt.xticks([]), plt.yticks([])

    plt.suptitle(meth)

    print("  ")

    plt.show()

Text

Description automatically generated



Graphical user interface, website

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence