import matplotlib.pyplot as plt

import numpy as np

import cv2

%matplotlib inline

img1 = cv2.imread("./pizza.jpg")

gray1 = cv2.cvtColor(img1, cv2.COLOR\_RGB2GRAY)

gray1 = np.float32(gray1)

points1 = cv2.cornerHarris(gray1, 4, 7, 0.02)

points1 = cv2.dilate(points1, None)

pointer1 = 0.01 \* points1.max()

img1[points1 > pointer1] = [0, 255, 0]

plt.imshow(img1, cmap='gray')

img2 = cv2.imread("./cake.jpg")

gray2 = cv2.cvtColor(img2, cv2.COLOR\_RGB2GRAY)

gray2 = np.float32(gray2)

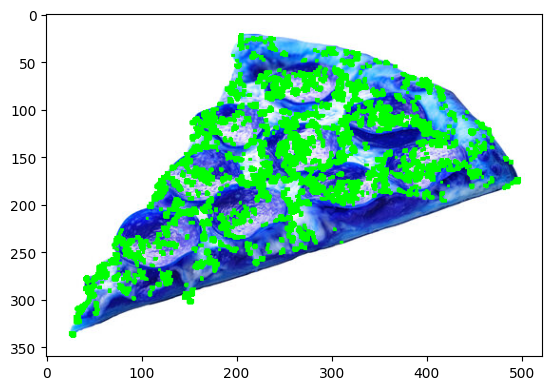
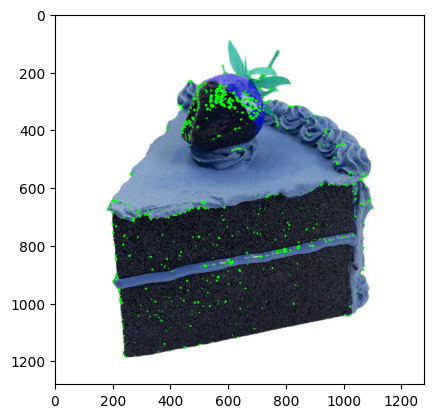
points2 = cv2.cornerHarris(gray2, 4, 7, 0.02)

points2 = cv2.dilate(points2, None)

pointer2 = 0.01 \* points2.max()

img2[points2 > pointer2] = [0, 255, 0]

plt.imshow(img2, cmap='gray')



import cv2

import numpy as np

import matplotlib.pyplot as plt

img = cv2.imread('/content/cat.jpeg')

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

plt.imshow(img)

plt.show()

transform\_img = cv2.resize(img, None, fx=0.2, fy=0.2)

plt.imshow(transform\_img)

plt.show()

r, c, \_ = img.shape

trans\_mat = np.float32([[1, 0, 150], [0, 1, 50]])

transform\_img = cv2.warpAffine(transform\_img, trans\_mat, (c, r))

plt.imshow(transform\_img)

plt.show()rot\_mat = cv2.getRotationMatrix2D((c/2, r/2), 60, 1)

transform\_img = cv2.warpAffine(transform\_img, rot\_mat, (c, r))

plt.imshow(transform\_img)

plt.show()

transform\_img = cv2.convertScaleAbs(transform\_img, alpha=1.8, beta=30)

plt.imshow(transform\_img)

plt.show()

sift = cv2.SIFT\_create()

kp1, des1 = sift.detectAndCompute(img, None)

kp2, des2 = sift.detectAndCompute(transform\_img, None)

bf = cv2.BFMatcher(cv2.NORM\_L2, crossCheck=True)

matches = bf.match(des1, des2)

matches = sorted(matches, key=lambda x: x.distance)

img\_result = cv2.drawMatches(img, kp1, transform\_img, kp2, matches, None, flags=cv2.DrawMatchesFlags\_NOT\_DRAW\_SINGLE\_POINTS)

plt.imshow(img\_result)

plt.show()

img\_d = cv2.imread('/content/dog.jpg')

img\_d = cv2.cvtColor(img\_d, cv2.COLOR\_BGR2RGB)

plt.imshow(img\_d)

plt.show()

sift = cv2.SIFT\_create()

kp1, des1 = sift.detectAndCompute(img, None)

kp2, des2 = sift.detectAndCompute(img\_d, None)

bf = cv2.BFMatcher(cv2.NORM\_L2, crossCheck=True)

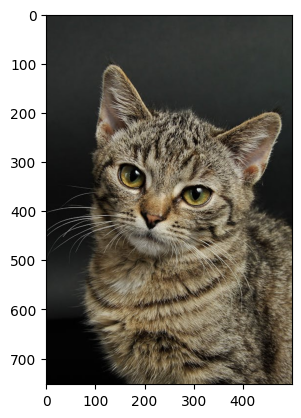
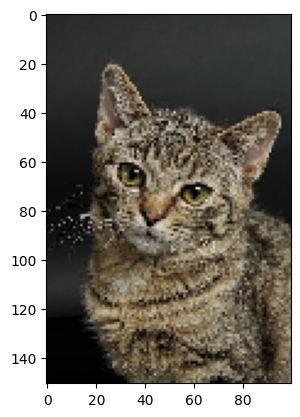
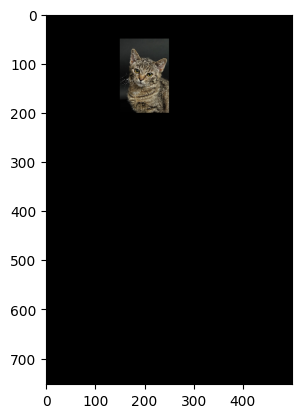
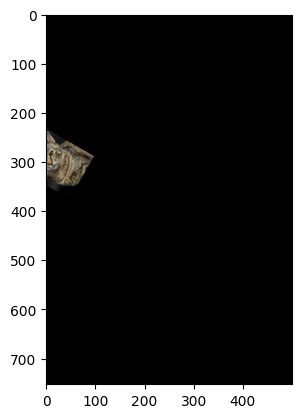
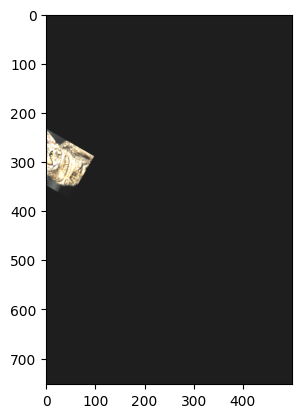
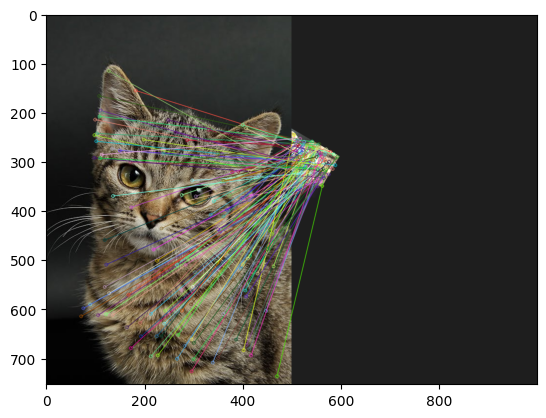
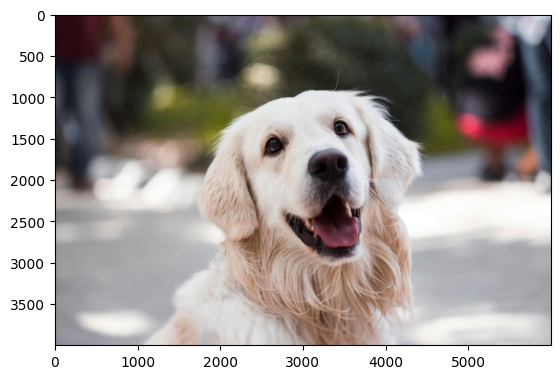
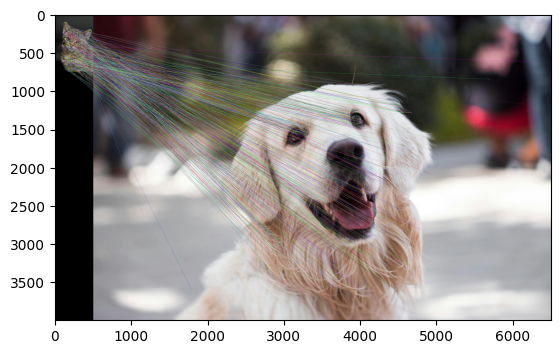
matches = bf.match(des1, des2)

matches = sorted(matches, key=lambda x: x.distance)

img\_result = cv2.drawMatches(img, kp1, img\_d, kp2, matches, None, flags=cv2.DrawMatchesFlags\_NOT\_DRAW\_SINGLE\_POINTS)

plt.imshow(img\_result)

plt.show()



import tensorflow as tf

from tensorflow.keras import layers, models

def vgg16(input\_shape):

model = models.Sequential()

model.add(layers.Conv2D(64, (3, 3), activation='relu', input\_shape=input\_shape))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2), strides=2))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2), strides=2))

model.add(layers.Conv2D(256, (3, 3), activation='relu'))

model.add(layers.Conv2D(256, (3, 3), activation='relu'))

model.add(layers.Conv2D(256, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2), strides=2))

model.add(layers.Conv2D(512, (3, 3), activation='relu'))

model.add(layers.Conv2D(512, (3, 3), activation='relu'))

model.add(layers.Conv2D(512, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2), strides=2))

model.add(layers.Conv2D(512, (3, 3), activation='relu'))

model.add(layers.Conv2D(512, (3, 3), activation='relu'))

model.add(layers.Conv2D(512, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2), strides=2))

model.add(layers.Flatten())

model.add(layers.Dense(4096, activation='relu'))

model.add(layers.Dense(4096, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

return model

from google.colab import files

uploaded = files.upload()

import tensorflow as tf

import numpy as np

train\_x = []

train\_y = [0,0,1,1,0]

for img\_dir in uploaded.keys():

image = tf.io.read\_file(img\_dir)

image = tf.image.decode\_jpeg(image, channels=3)

image = tf.image.resize(image, (224, 224))

image = image / 255.0

image = image.numpy()

train\_x.append(image)

train\_x = np.array(train\_x)

train\_y = np.array(train\_y)

input\_size = (224, 224, 3)

vgg16\_model = vgg16(input\_size)

vgg16\_model.summary()

train\_model = vgg16\_model.fit(

train\_x,

train\_y,

epochs=5,

batch\_size=2,

)