OpenCV. РОБОТА із ЗОБРАЖЕННЯМИ

Файл: CV_Image_09_001

Детектори кутів. FAST алгоритм

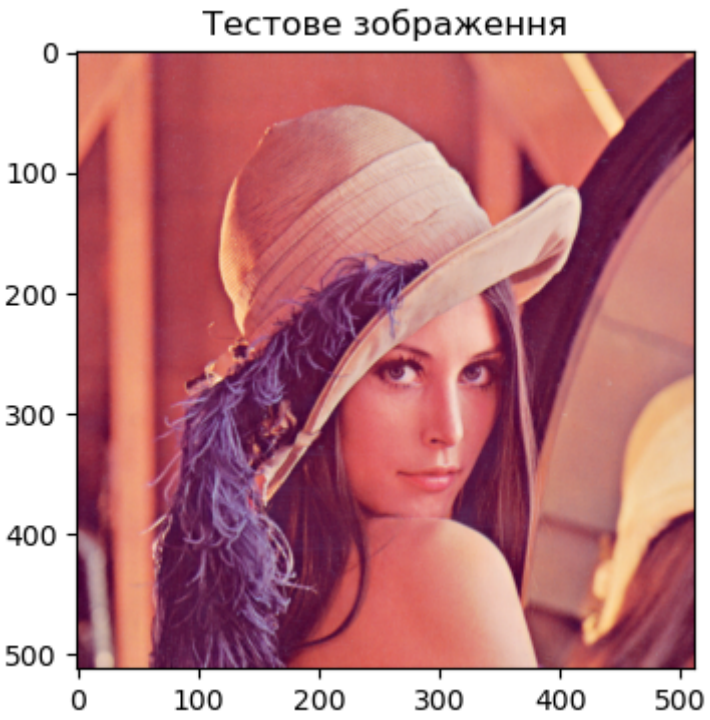
[FAST Algorithm for Corner Detection]

```
# Завантаження бібліотек
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import skimage.io as io
import time
```

Завантаження зображення

```
## %%script false
# Завантаження зображення
path = './IMAGES/'
filename = 'Lenna.png'
# filename = 'Face_1_300_X_400.jpg'
test_img_ = io.imread(path+filename)
#  Визначення стркутури та розміру зображення
print('---- TEST IMAGE  -----')
print('IMAGE SHAPE', test_img_.shape, 'IMAGE SIZE', test_img_.size)
irows_num = test_img_.shape[0]  # кількість рядків
iclms_num = test_img_.shape[1]  # кількість колонок
print('ROWS NUMBER', irows_num, 'CLMS NUMBER', iclms_num)
fig, ax = plt.subplots(figsize=(4, 4))
plt.imshow(test_img_)
plt.title("Тестове зображення")
plt.show()
```

```
---- TEST IMAGE  -----
IMAGE SHAPE (512, 512, 3) IMAGE SIZE 786432
ROWS NUMBER 512 CLMS NUMBER 512
```




```
test_img = cv.cvtColor(test_img_, cv.COLOR_BGR2RGB)
# Wait for a key press and close the window
while True:
    cv.imshow('Display window', test_img)
    if cv.waitKey(1) == ord('q'):
        break
cv.destroyAllWindows()
```

Рамка

```
# I am giving  big random numbers for x_min and y_min because if you initialize
them as zeros whatever coordinate you go minimum will be zero
x_min,y_min,x_max,y_max=36000,36000,0,0

def coordinat_chooser(event,x,y,flags,param):
    global go , x_min , y_min, x_max , y_max

    # when you click the right button, it will provide coordinates for variables
    if event==cv.EVENT_RBUTTONDOWN:

        # if current coordinate of x lower than the x_min it will be new x_min ,
same rules apply for y_min
        x_min=min(x,x_min)
        y_min=min(y,y_min)


         # if current coordinate of x higher than the x_max it will be new x_max ,
same rules apply for y_max
        x_max=max(x,x_max)
        y_max=max(y,y_max)

        # draw rectangle
```

```
        cv.rectangle(test_img,(x_min,y_min),(x_max,y_max),(0,255,0),1)


    """
        if you didn't like your rectangle (maybe if you made some misscliks),
reset the coordinates with the middle button of your mouse
        if you press the middle button of your mouse coordinates will reset and
you can give a new 2-point pair for your rectangle
    """
    if event==cv.EVENT_MBUTTONDOWN:
        print("reset coordinate  data")
        x_min,y_min,x_max,y_max=36000,36000,0,0
```

ОБИРАЄМО BOX

```
cv.namedWindow('coordinate_screen')
# Set mouse handler for the specified window, in this case, "coordinate_screen"
window
cv.setMouseCallback('coordinate_screen',coordinat_chooser)

while True:
    cv.imshow("coordinate_screen",test_img) # show only first frame

    k = cv.waitKey(5) & 0xFF # after drawing rectangle press ESC
    if k == 27:
        cv.destroyAllWindows()
        break

print (x_min,y_min,x_max,y_max)
```

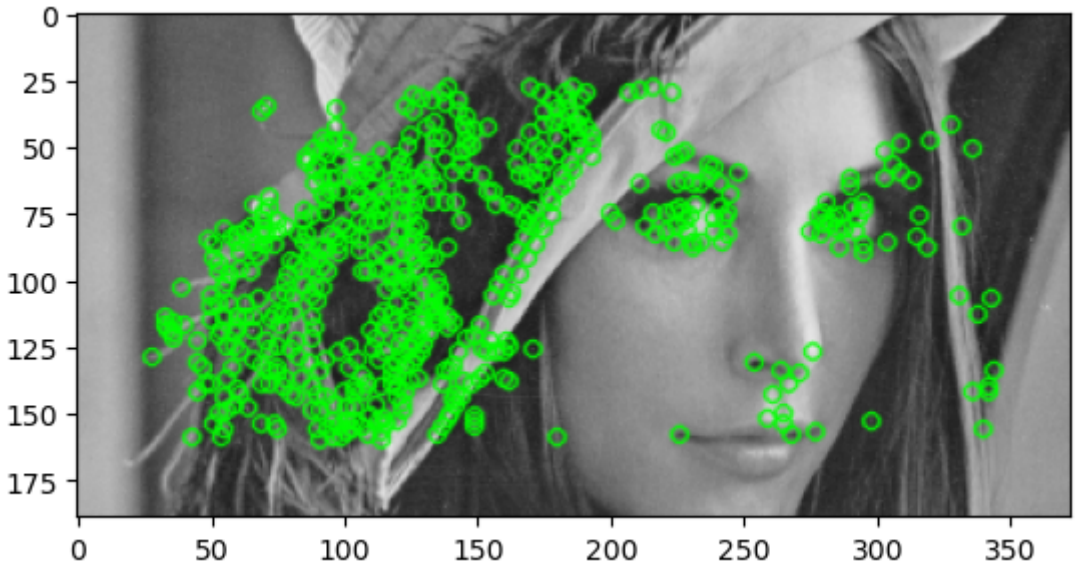
```
41 191 414 380
```

3 зображення беремо прямокутник інтересу (BOX)

```
# take region of interest ( take inside of rectangle )
image_box=test_img[y_min:y_max,x_min:x_max]

# convert roi to grayscale, SIFT Algorithm works with grayscale images
image_box_gray=cv.cvtColor(image_box,cv.COLOR_BGR2GRAY)
#image_box_gray_ = cv.cvtColor(image_box_gray, cv.COLOR_BGR2RGB)

# Wait for a key press and close the window
while True:
    cv.imshow('Display window', image_box_gray)
    if cv.waitKey(1) == ord('q'):
        break
cv.destroyAllWindows()
```

```
# Initialize the FAST detector and BRIEF descriptor extractor
fast = cv.FastFeatureDetector_create(threshold=20)
brief = cv.xfeatures2d.BriefDescriptorExtractor_create()
```

```
# detect keypoints
keypoints_1 = fast.detect(image_box_gray, None)
# descriptors
keypoints_1, descriptors_1 = brief.compute(image_box_gray, keypoints_1)

# draw keypoints for visualizing
keypoints_image = cv.drawKeypoints(image_box_gray, keypoints_1, outImage=None,
color=(0, 255, 0))
# display keypoints
plt.imshow(keypoints_image,cmap="gray")
```



Відео

Відеофайли з free stock сервісу [PIXABAY]

```
# Path to video
path = './IMAGES/'
filename = '01_Bird_.mp4'
video = cv.VideoCapture(path + filename)

# read only the first frame for drawing a rectangle for the desired object
ret,frame = video.read()
test_frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB)

# Wait for a key Q press and close the window
while True:
    cv.imshow('Display window', test_frame)
    if cv.waitKey(1) == ord('q'):
        break
cv.destroyAllWindows()
```

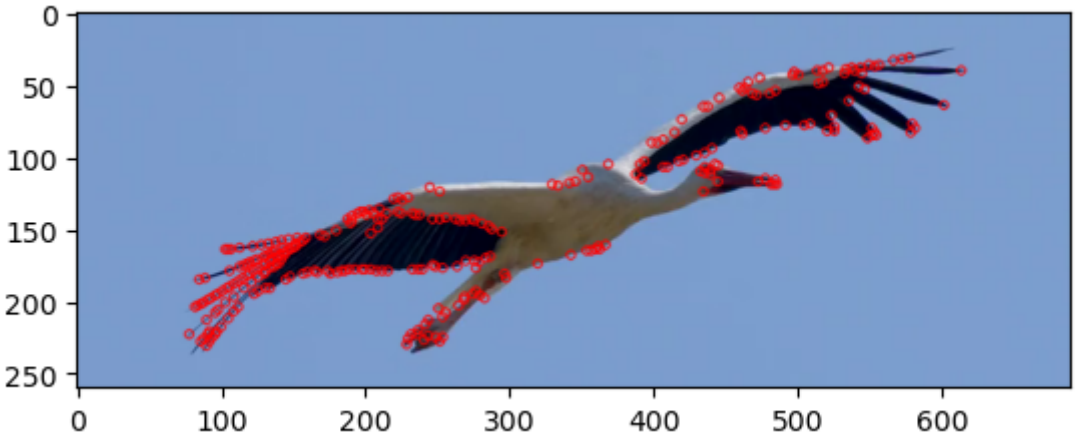
Перший кадр відофрагменту

```
x_min,y_min,x_max,y_max=36000,36000,0,0

cv.namedWindow('video_frame')
# Set mouse handler for the specified window, in this case, "coordinate_screen"
window
test_img = test_frame
cv.setMouseCallback('video_frame',coordinat_chooser)

while True:
    cv.imshow("video_frame",test_frame)

    k = cv.waitKey(5) & 0xFF # after drawing rectangle press ESC
    if k == 27:
        cv.destroyAllWindows()
        break

print (x_min,y_min,x_max,y_max)
```

```
281 310 971 570
```

Перший кадр відофрагменту з означеним прямокутником інтересу

```
# take region of interest ( take inside of rectangle )
roi_image=test_frame[y_min:y_max,x_min:x_max]

# convert roi to grayscale, SIFT Algorithm works with grayscale images
roi_gray=cv.cvtColor(roi_image,cv.COLOR_BGR2GRAY)

# Initialize the FAST detector and BRIEF descriptor extractor
fast = cv.FastFeatureDetector_create(threshold=20)
brief = cv.xfeatures2d.BriefDescriptorExtractor_create()

# detect keypoints
keypoints_1 = fast.detect(roi_gray, None)
# descriptors
keypoints_1, descriptors_1 = brief.compute(roi_gray, keypoints_1)

# draw keypoints for visualizing
keypoints_image = cv.drawKeypoints(roi_image, keypoints_1, outImage=None, color=
(255, 0, 0))
# display keypoints
plt.imshow(keypoints_image,cmap="gray")
```




```
# matcher object
bf = cv.BFMatcher()

# Variables for FPS calculation
frame_count = 0
start_time = time.time()

while True :
    # reading video
    ret,frame=video.read()

    if ret:
          # convert frame to gray scale
        frame_gray=cv.cvtColor(frame,cv.COLOR_BGR2GRAY)


        # Detect keypoints using FAST
        keypoints_2 = fast.detect(frame_gray, None)

        # Compute descriptors using BRIEF
        keypoints_2, descriptors_2 = brief.compute(frame_gray, keypoints_2)

        """
        Compare the keypoints/descriptors extracted from the
        first frame(from target object) with those extracted from the current
frame.
        """
        if descriptors_2 is  not None:
            matches =bf.match(descriptors_1, descriptors_2)


            for match in matches:

                # queryIdx gives keypoint index from target image
                query_idx = match.queryIdx

                # .trainIdx gives keypoint index from current frame
                train_idx = match.trainIdx
```

```
                # take coordinates that matches
                pt1 = keypoints_1[query_idx].pt

                # current frame keypoints coordinates
                pt2 = keypoints_2[train_idx].pt

                # draw circle to pt2 coordinates , because pt2 gives current frame
coordinates
                cv.circle(frame,(int(pt2[0]),int(pt2[1])),5,(255,0,0),-1)

        # Calculate and display FPS
        frame_count += 1
        elapsed_time = time.time() - start_time
        fps = frame_count / elapsed_time
        cv.putText(frame, f"FPS: {fps:.2f}", (10, 30), cv.FONT_HERSHEY_SIMPLEX, 1,
(0, 255, 0), 3)


        cv.imshow("coordinate_screen",frame)


        k = cv.waitKey(5) & 0xFF # after drawing rectangle press esc
        if k == 27:
            cv.destroyAllWindows()
            break
    else:
        break

video.release()
cv.destroyAllWindows()
```