الجلسة السادسة

Feature Extraction

Template matching

مطابقة القوالب

1.1 خوارزمية مطابقة القوالب Template Matching

- لها عدة طرق:
- Correlation
- Normalized Correlation
- Squared Difference
- Normalized Squared Difference

يمكن تنفيذها من خلال تابع cv2.matchTemplate(gray_img, template, method) حيث نمرر له الصورة الرمادية والقالب المراد كشفه ضمنها والطريقة المراد مطابقة القوالب وفقاً لها (correlation, Squared difference).

تستخدم هذه الخوارزمية لكشف مناطق محددة ضمن الصورة.

1.2 التنفيذ العملي:

نفذ الكود التالي:

```
import cv2
import numpy as np
# Load the original and template images
img1 = cv2.imread('original.png', 1)
template = cv2.imread('template.png', 0)
```

قراءة الصورة الأصلية وصورة القالب

```
# Convert the images to grayscale
gray_img = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
```

تحويل الصورة للتدرج الرمادي والاحتفاظ بالنسخة الملونة من أجل العرض النهائي للنتيجة

```
# Get the dimensions of the template
w, h = template.shape[::-1]

# Perform template matching using different methods
methods = [cv2.TM_CCOEFF, cv2.TM_CCOEFF_NORMED, cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED]

for method in methods:
    img2=img1.copy()
    # Apply template matching
    result = cv2.matchTemplate(gray_img, template, method)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
    # Choose the appropriate threshold based on the method
    if method in [cv2.TM_CCOEFF, cv2.TM_CCOEFF_NORMED]:
        threshold = max_val-0.05
        print(threshold)
        # Find all occurrences above the threshold
        loc = np.where(result >= threshold)
        for pt in zip(*loc[::-1]):
            # Draw a rectangle around the matched region
            cv2.rectangle(img2, pt, (pt[0] + w, pt[1] + h), (0, 255, 0), 2)
    else:
        print(threshold)
        threshold = min_val*2
        loc = np.where(result <= threshold)
        for pt in zip(*loc[::-1]):
            # Draw a rectangle around the matched region
            cv2.rectangle(img2, pt, (pt[0] + w, pt[1] + h), (0, 255, 0), 2)
    # Display the matched image
```

الطرق المستخدمة هي correlation, squared difference بشكلهما الأساسي وبالنسخة normalized

تطبيق تحويل هاف من خلال تابع matchTemplate الجاهز في مكتبة OpenCV
وسنقوم بتنفيذ حلقة في كل تكرار منها سنقوم بتطبيق عملية مطابقة القوالب باستخدام طريقة مختلفة وعرض النتيجة

من ناتج مطابقة القوالب نأخذ القيم العظمى والصغري ومواقع كل منها في الصورة (تمثل القيمة العظمى والصغرى مركز المنطقة التي تم العثور فيها على القالب في الصورة) لذلك نحتاج لإحاطة المنطقة بمستطيل مركزه في هذه النقطة العظمى أو الصغرى حسب الطريقة المستخدمة

في حال استخدام طريقتي correlation نستخدم القيمة العظمى بحيث نحتفظ بجميع نتائج المطابقة التي تكون قيمة correlation عندها أكبر من max_value-0.05
ثم ننفذ حلقة يتم فيها رسم مستطيل حول المناطق التي كانت قيمة correlation عندها أكبر أو تساوي العتبة

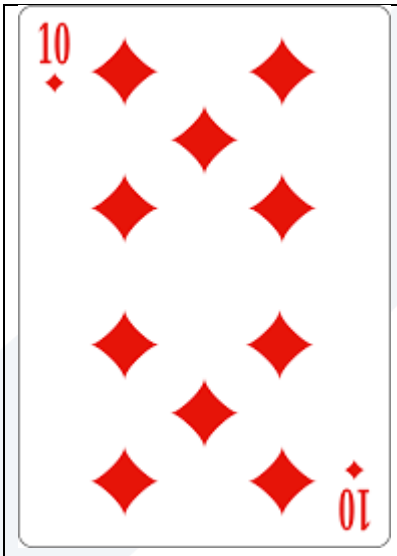
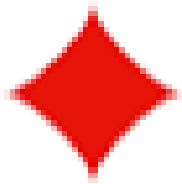
في حال استخدام طريقتي squared difference نستخدم القيمة الصغرى بحيث نحتفظ بجميع نتائج المطابقة التي تكون قيمة correlation عندها أقل من min_val*2
ثم ننفذ حلقة يتم فيها رسم مستطيل حول المناطق التي كانت قيمة squared difference عندها أقل أو تساوي العتبة

```
cv2.imshow('Matched Image', img2)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

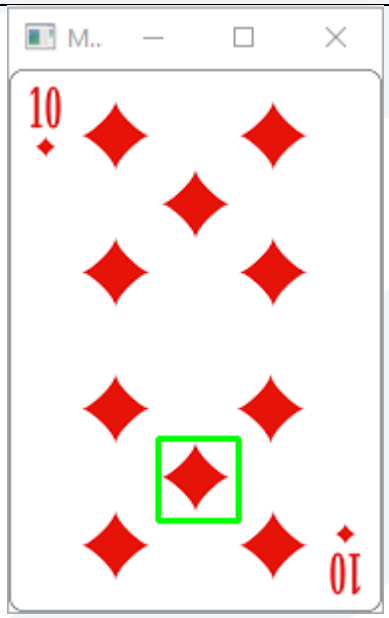
عرض الصورة الأصلية الملونة وفوقها أماكن تواجد القالب

تغيير العتبات يغير النتيجة، قد يقل ظهور بعض المناطق التي لا تمثل شكل القالب عند تقليل قيمة العتبة لكن قد تحذف أيضاً بعض المناطق الصحيحة.

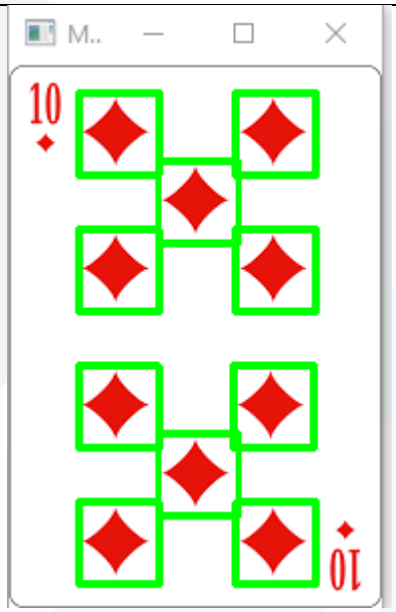
الصورة الأصلية مع القالب

 Original Image	 Template

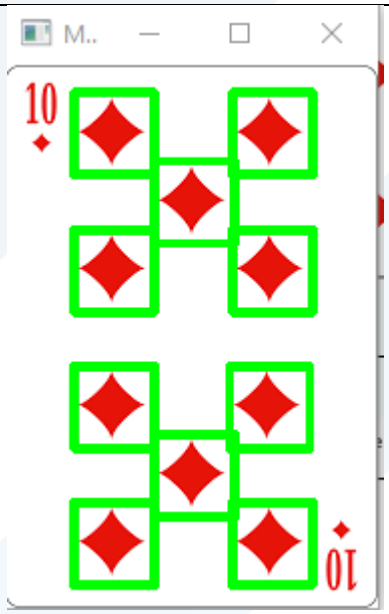
نتيجة التنفيذ:



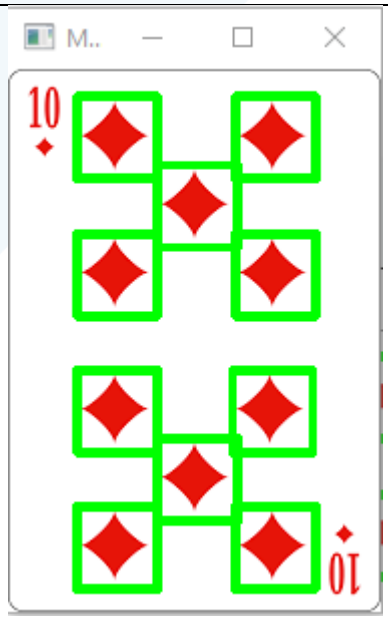
correlation



Normalized correlation



Squared difference



Normalized squared difference

مثال ثاني:

سنستخدم الصورة الأصلية والقالبين المبينين في الشكل

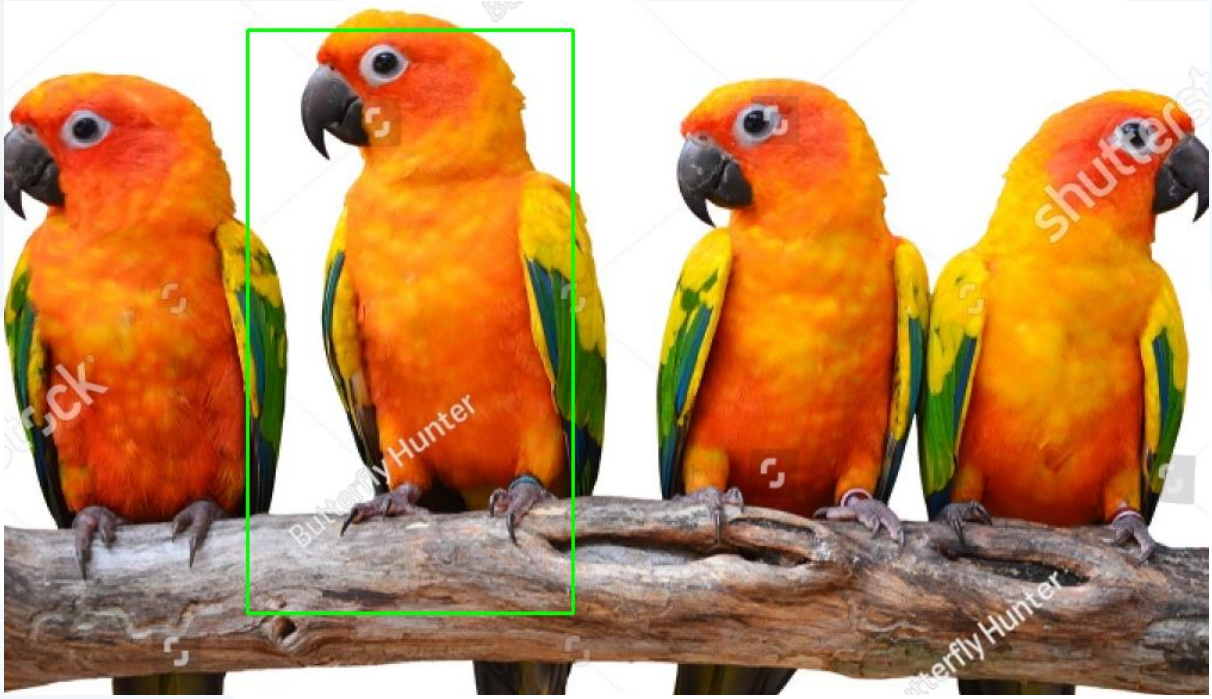
Original image

First template

Second template

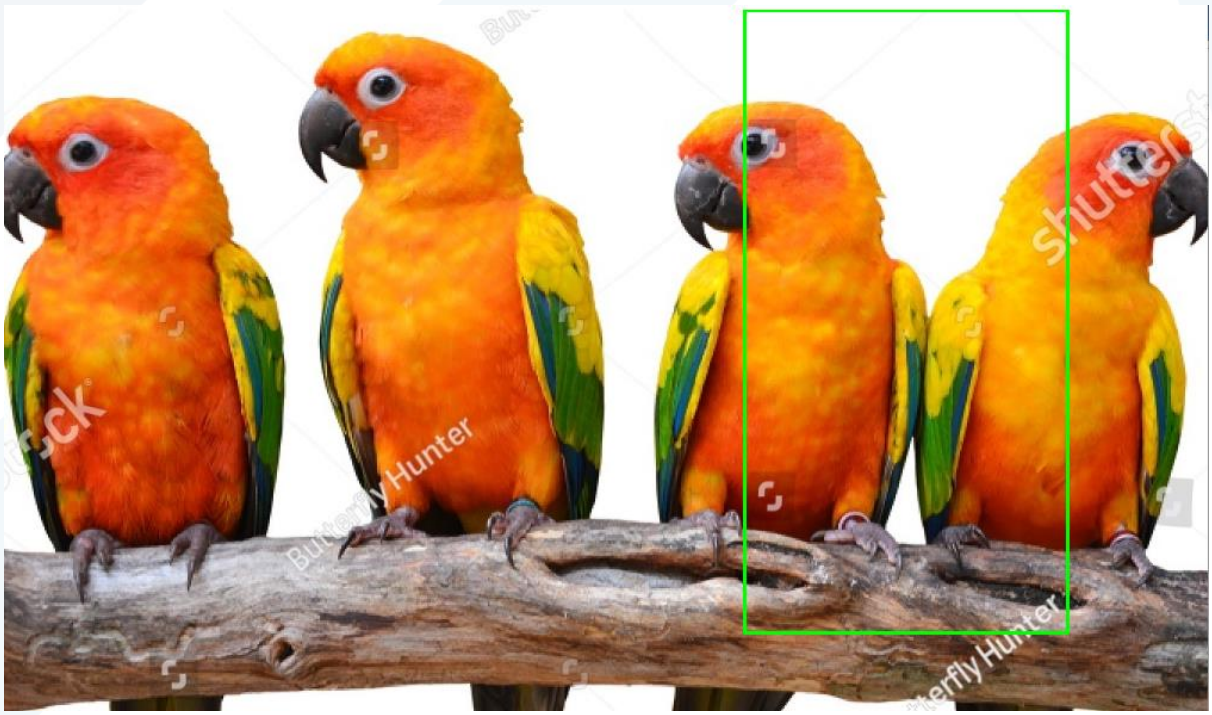
نتيجة تنفيذ الكود باستخدام القالب الأول:

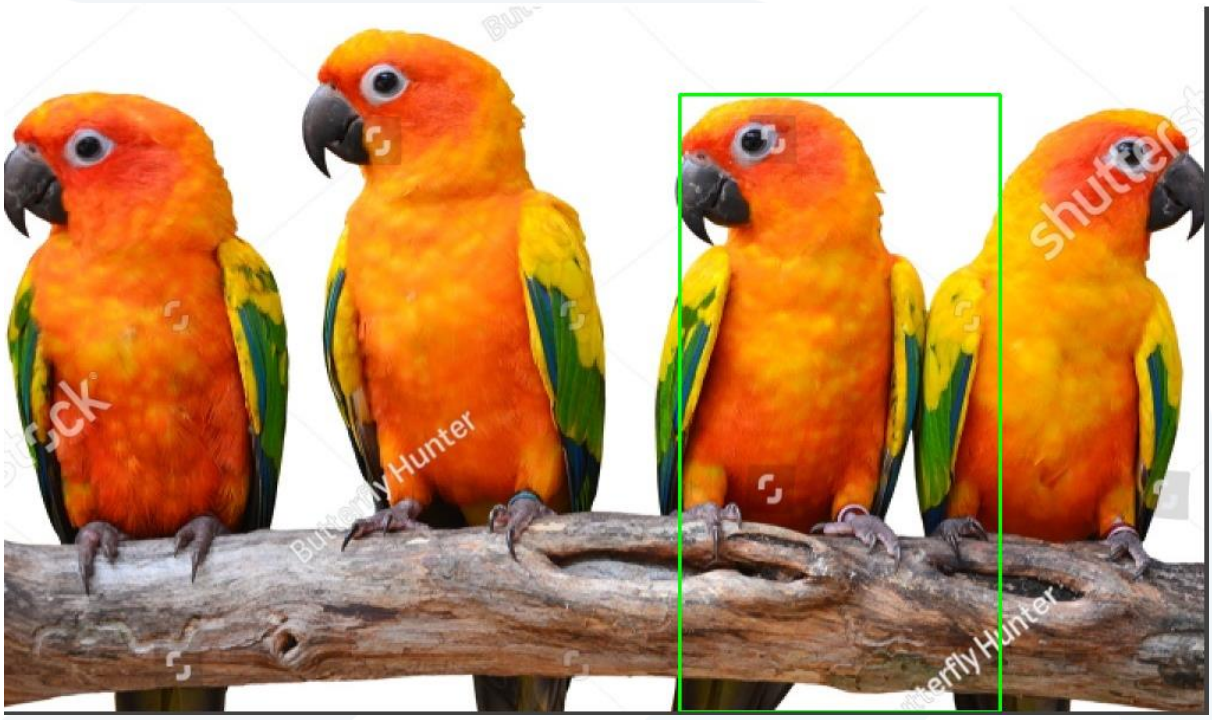

Butterfly Hunter
shutter


Butterfly Hunter
shutter

بالنسبة للقالب الأول جميع الطرق كانت جيدة.

والآن نستخدم القالب الثاني:





كانت طريقة Squared Difference أفضل في حالة القالب الثاني.