



جامعة دمشق

كلية الهندسة المعلوماتية

الرؤية الحاسوبية

تقرير إنجاز مشروع

Jigsaw Solver

تقديمة:

أسامه يوسف بازو

حمزة محمد المحروس

عليا ماجد المسوتي

ياسين راتب عبد المهدى

18-12-2023

المرحلة الأولى:

حل المرحلة بدون استخدام الـ `hint`:

تم في البداية بناء `class Piece` وذلك لتمثيل القطع الموجودة في الصورة حيث يمتلك الـ `class` `attributes` التالية:

- `pieceNum`: رقم القطعة الحالية.
- `Size_vertical`: طول القطعة الأفقي.
- `Size_horizontal`: طول القطعة العمودي.
- `pieceChn`: عدد الفنوات اللونية في القطعة.
- `pieceStart`: لتحديد رقم موقع البداية للقطعة الأفقي والعمودي.
- `pieceData`: المعلومات التي تحتويها القطعة (قيم البسكلات).
- `pieceTotal`: عدد القطع الكلي.
- `sideUp`: مصفوفة تحتوي على قيم البسكلات الموجودة في الطرف العلوي للقطعة.
- `SieRight`: مصفوفة تحتوي على قيم البسكلات الموجودة في الطرف اليمني للقطعة.
- `sideDown`: مصفوفة تحتوي على قيم البسكلات الموجودة في الطرف السفلي للقطعة.
- `sideLeft`: مصفوفة تحتوي على قيم البسكلات الموجودة في الطرف اليساري للقطعة.
- `Sides`: مصفوفة تحتوي على جميع الحواف التي تحتويها القطعة.
- `neighbors`: مصفوفة تحتوي على أرقام القطع المجاورة للقطعة الحالية.
- `differences`: مصفوفة تحتوي على قيم الفروق بين القطعة ومجاوراتها.

أسلوب الحل:

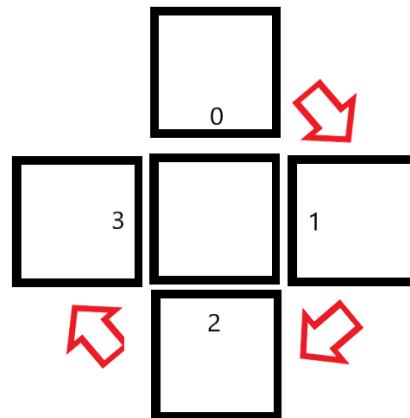
قمنا في البداية بأخذ عدد الصدوف والأعمدة من المستخدم، ومن ثم قمنا بتقسيم الصورة لمصفوفة من القطع بناء على المدخلات، حيث قمنا بتخزين القطع في الصورة في مصفوفة من الـ `class Piece`.

تقوم فكرة الخوارزمية على إيجاد أفضل القطع المجاورة للقطعة الحالية.

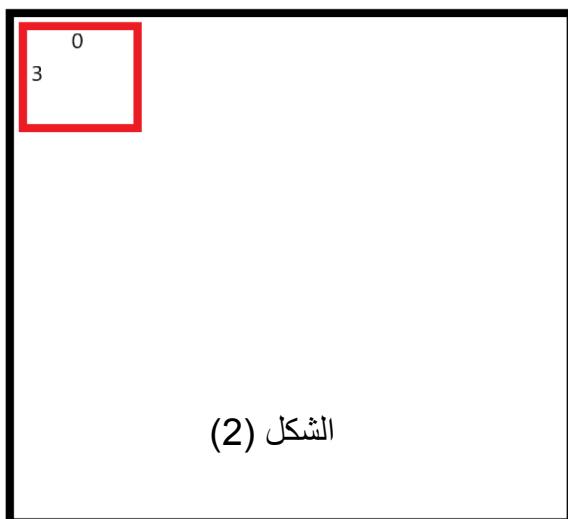
حيث نقوم بإيجاد أفضل المجاورات عن طريق حساب فروق القيم اللونية لكل بكسل مع المقابل له من القطعة الأخرى، فتكون القطعة هي المجاورة لقطعة أخرى عند حافة ما عندما يكون هذا الفرق أقل ما يمكن.

حيث نقوم بالمرور على حافة القطعة من الجهة العليا في البداية و الدوران مع عقارب الساعة حيث تم الاصطلاح على أن الـ `index` التالية لمصفوفة التجاور: 0 الحافة العليا، 1 الحافة اليمنى، 2 الحافة السفلى، 3 الحافة اليسرى.

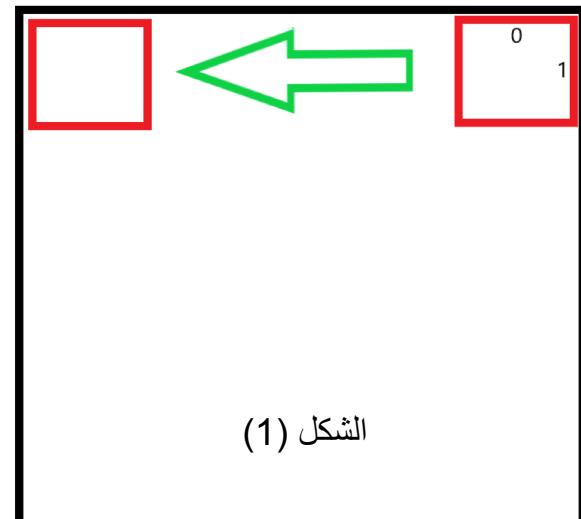
كما في الصورة الموضحة تاليا:



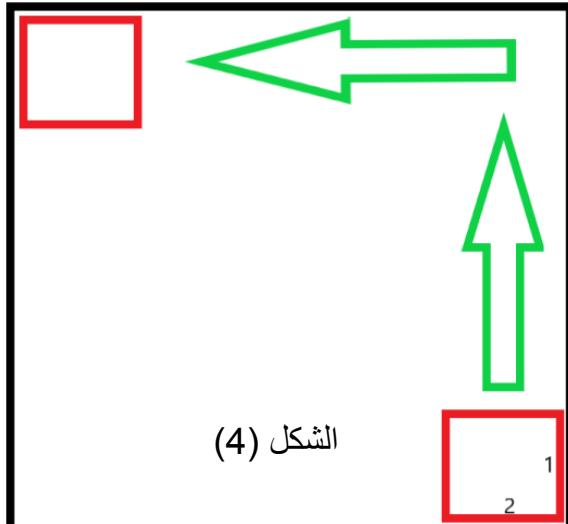
عند القيام بتجمیع القطع في النهاية نقوم بالتحقق من القطع في الزوايا؛ أي أننا نقوم بإيجاد القطع الأربع التالية:



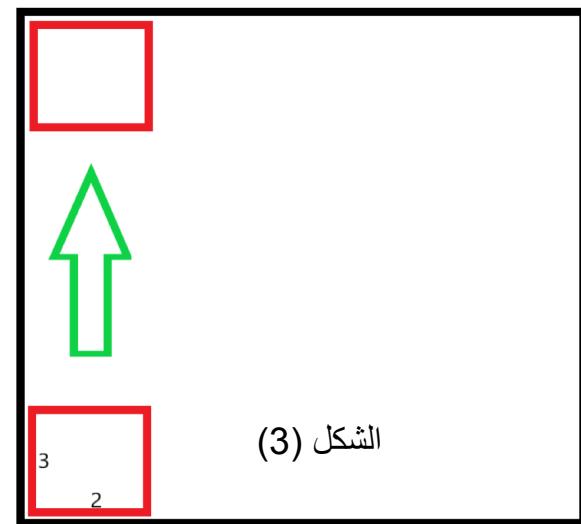
الشكل (2)



الشكل (1)



الشكل (4)



الشكل (3)

بعد القيام بإيجاد محاورات جميع القطع نبدأ بتركيب الصورة الناتجة، يتم التركيب من قطعة البداية والتي تقع في الزاوية العليا اليسرى، حيث تكون عملية إيجاد هذه القطعة هي المفتاح الرئيسي الذي سيتم عليها تركيب بقية الصورة، لذلك لضمان إيجاد القطعة نقوم بإيجاد قطع الزوايا الأربع المشكلة للصورة النهائية كما هو موضح في الصورة السابقة.

تكون قطع الزوايا تملك قيم مميزة للفروقات عند حواف معينة وفق البنية التي تم اعتمادها لتمثيل القطع، ومنه نقوم بأخذ القطعة ذات الحواف التي تملك أكبر قيم ومنها نتجه لقطعة البداية.

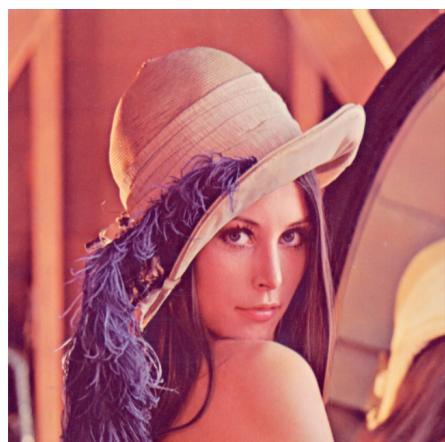
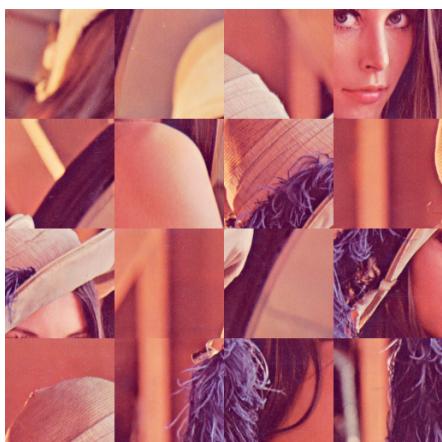
في حال كانت تقع في الزاوية العليا اليمنى كما في الشكل(1) نقوم بالاتجاه يسارا لإيجاد قطعة البداية، في حال كانت قطعة البداية كما في الشكل(2) نقوم بتباعية الصورة بشكل طبيعي، في حال كانت القطعة في الزاوية السفلى اليمنى نقوم بالإتجاه للأعلى ومن ثم لليسار للوصول إلى قطعة البداية، في حال كانت القطعة في الزاوية السفلى اليسرى نقوم بالإتجاه للأعلى للوصول إلى قطعة البداية.

النتائج:

كانت النتائج كالتالي عند القيام بالتطبيق على مختلف الأمثلة:

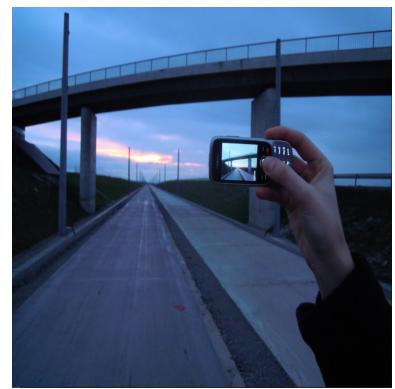
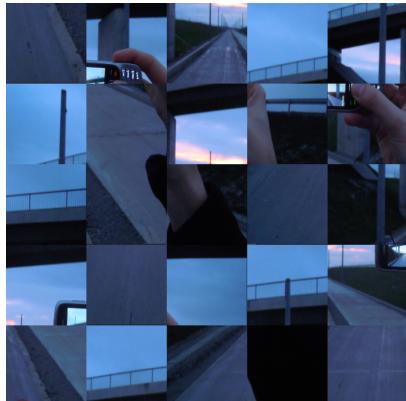
الصورة تتتألف من 25 قطعة بأبعاد 5,5:

- الدقة: %100
- الزمن المستغرق: 0.52 ثانية.
- النتيجة:



الصورة تتتألف من 25 قطعة بأبعاد 5,5:

- الدقة: %100
- الزمن المستغرق: 0.78 ثانية.
- النتيجة:



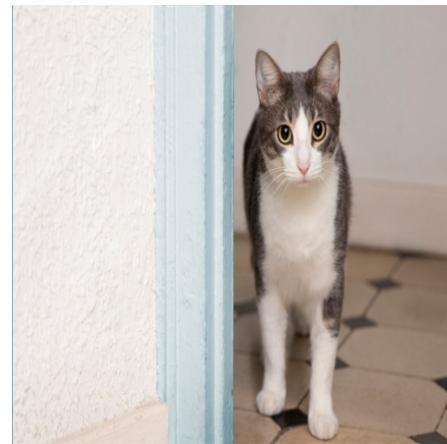
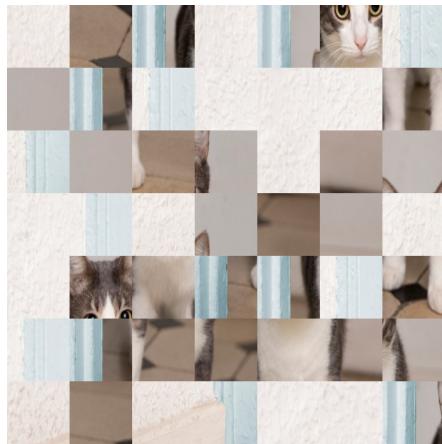
الصورة تتتألف من 36 قطعة بأبعاد 6,6:

- الدقة: %100
- الزمن المستغرق: 0.71 ثانية.
- النتيجة:



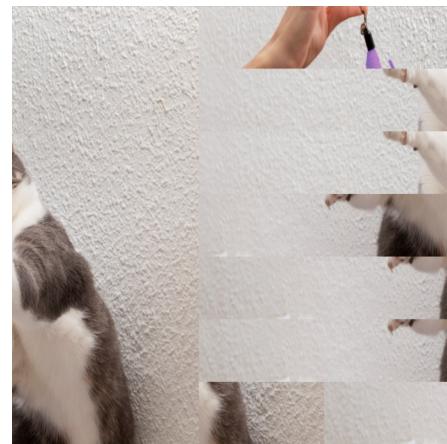
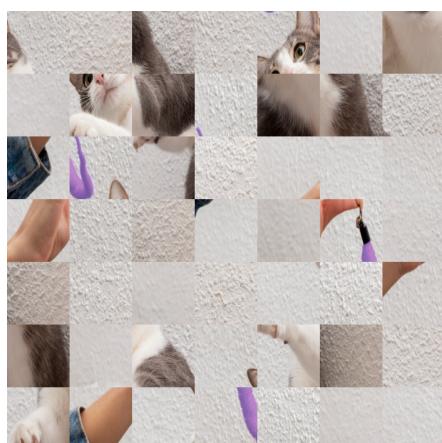
الصورة تتتألف من 49 قطعة بأبعاد 7,7:

- الدقة: %100
- الزمن المستغرق: 1.88 ثانية.
- النتيجة:



الصورة تتتألف من 49 قطعة 7,7:

- الدقة: %57
- الزمن المستغرق: 1.88 ثانية.
- النتيجة:



عيوب الخوارزمية:

إحدى العيوب الواضحة في الخوارزمية هو في حال كانت الصورة تميز بقيم لونية متقاربة وخاصة في الصورة المقطعة، حيث يؤدي ذلك إلى إيجاد قطع مجاورة خاطئة نتيجة لفروقات اللونية بين الحواف في القطع، فتصبح الصورة النهائية خاطئة.

حل المرحلة مع استخدام الـ **hint**

في هذه المرحلة يكون لدينا بالإضافة لصورة الـ **Puzzle** الصورة الأصلية سنستخدمها كـ **hint**. بداية نجري عملية **resize** لكلا الصورتين ليصبحوا بنفس الحجم ثم نختبر طريقتين في هذه المرحلة وسيتم ذكر كل الطريقتين تالياً:

1- طريقة خوارزميات الـ **SIFT** والـ **ORB**:

تم اعتماد خوارزميات الـ **SIFT** والـ **ORB** لمعرفة مكان تموير كل قطعة من قطع الـ **Puzzle** بداية نقوم بتقسيم الـ **Puzzle** للحصول على كل قطعة على حدا، وبعدها نقوم بإجراء عملية استخراج الـ **features** الموجودة في القطعة وإيجاد التطابق بينها وبين صورة الـ **hint** ذلك لمعرفة مكان تموير هذه القطعة ضمن الـ **Grid**.

خلال هذه العملية نقوم بعملية إسقاط لمعرفة مكان تجمع الـ **features** الخاصة بالقطعة الحالية وعن طريقة موقع هذه الـ **features** نقوم باكتشاف مكان القطعة الحالية، وبالمرور على جميع القطع نقوم بمعرفة مكان كل قطعة وبالتالي نقوم بتكوين الـ **Puzzle** المحلول حسب الموقع المكتشف لكل قطعة.

النتائج:

هذه الطريقة لم تكون ذات دقة عالية فمن طريقها لم يتم اكتشاف جميع القطع فتبقي بعض القطع غير معروفة الموقع بسبب قلة الـ **features** بها أو توزع الـ **features** بأماكن عددة في الصورة الـ **hint**.

الصورة تتكون من 12 قطعة بأبعاد 4,3:

- الدقة: %66.66
- الزمن المستغرق: 0.48 ثانية.

2- طريقة الفروق اللونية لبكسلات كل قطعة:

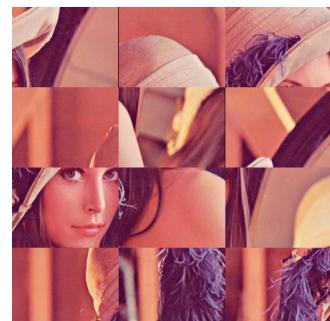
في هذه الطريقة نقوم بقطع puzzle والصورة الأصلية hint إلى قطع من قياس قطعة الـ puzzle وبعدها نقوم بعملية مقارنة لكل قطعة من قطع الـ puzzle مع قطع الصورة الـ hint والقطعة ذات أقل فرق تكون هي القطعة المقابلة لقطعة الـ puzzle الحالية ونقوم بحفظ مكانها بالصورة الـ hint وننتقل للقطعة التالية وهكذا حتى المرور على جميع القطع.

النتائج:

هذه الطريقة كانت ذات نتائج أفضل مقارنة بسابقتها وهي ذات دقة عالية فعن طريقها تم حل puzzles مؤلفة من 120 قطعة و 144 قطعة.

الصورة تتتألف من 12 قطعة بأبعاد 3,4:

- الدقة: %100
- الزمن المستغرق: 0.03 ثانية.



الصورة تتتألف من 120 قطعة بأبعاد 10,12:

- الدقة: %100
- الزمن المستغرق: 0.29 ثانية.



الصورة تتتألف من 144 قطعة بأبعاد 9,16

- الدقة: %100
- الزمن المستغرق: 0.39 ثانية.



المرحلة الثانية:

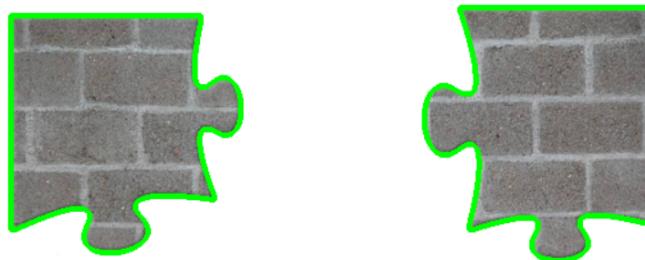
حل المرحلة بدون استخدام الـ `hint`:

قمنا بدايةً باستخراج قطع الـ `puzzle` من الصورة المعطاة كالتالي:

- ايجاد اللون الأكثر تكرارا في الصورة.
- بناء `mask` يأخذ القيمة 0 عند اللون الأكثر تكرارا، القيمة 1 عدا ذلك.
- تطبيق عملية الـ `opening` على الـ `mask` لإزالة الـ `noise`.
- تطبيق عملية الـ `closing` على الـ `mask`. يصبح الماسك بالشكل:



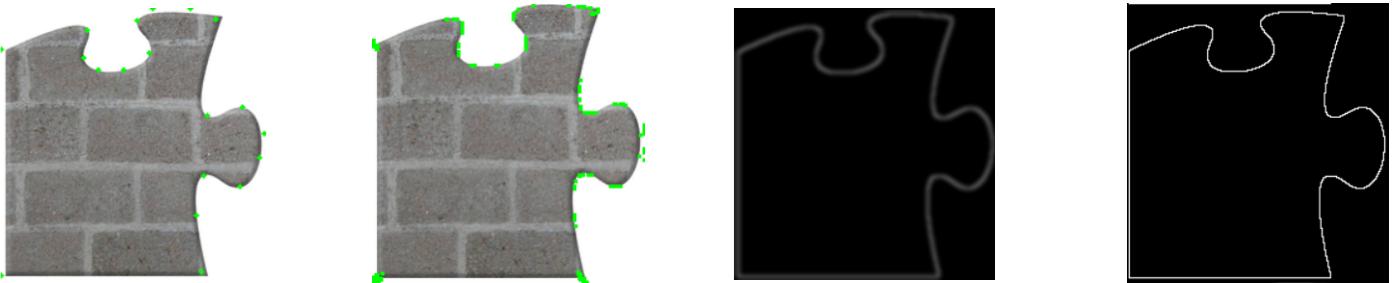
- إيجاد حواف كل قطعة عن طريق تطبيقتابع الـ `cv.findContours` على `mask` الصورة أعلاه. نتيجة الـ `contours` على الصورة الأساسية (الـ `cv.findContours` ملونة بالأحمر):



من أجل كل حافة من الحواف أعلاه (أي من أجل كل قطعة `puzzle`) نطبق العمليات التالية:

- إيجاد الزوايا الأربع للقطعة:

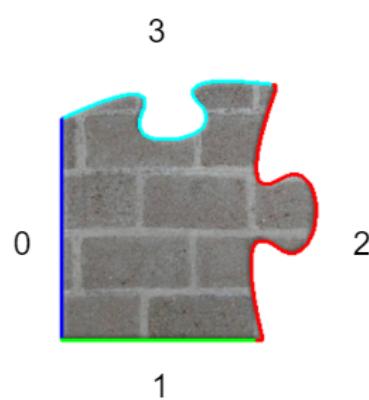
نوجد من `mask` القطعة الحالية حواف الماسك (من خلال طرح الـ `eroded mask` من `mask`) و نستخرج منه نقاط الحواف و نرتيبها مع عقارب الساعة (بالاعتماد على الزاوية التي تشكلها النقطة مع مركز القطعة). يستخدم الـ `Gaussian Blur` على حواف الـ `mask` و نستخدم التابع `cv.cornerHarris` عليه فينتج لدينا عدد كبير من الزوايا، نقل العدد من خلال إبقاء نقطة واحدة في كل منطقة (بناء على قيمة `threshold`) كما يلي:



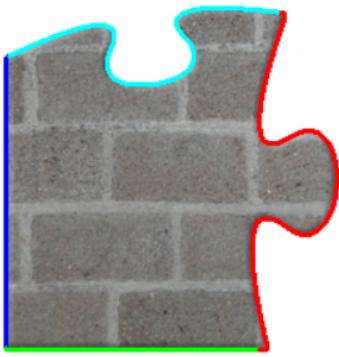
ومن ثم نختار 4 نقاط منها وفقا للشروطين: أن تكون أضلاع الشكل الناتج متساوية إلى أكثر حد، وأن تعطي أكبر مساحة. تم اختبار منهجين لحساب المساحة التي تشكلها النقاط الأربع و هما:

- مساحة الرباعي الذي تشكله النقاط
 - مساحة الشكل الناتج عن تقاطع الرباعي الذي تشكله النقاط مع mask قطعة الـ puzzle
- و بعد التجريب تم اعتماد المنهج الثاني.
- تحديد نوع كل حافة من حواف القطعة (رأس، حفرة، مستقيم):

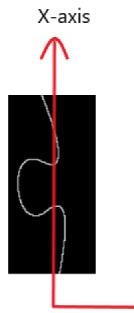
نوجد الحواف الأربع بعد تحديد أنساب أربع نقاط لزوايا و تكون الحواف مرتبة كالتالي:



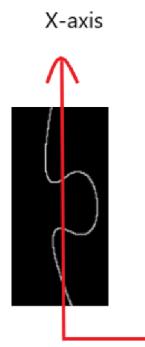
الخطوة التالية تتضمن في تحديد طبيعة كل حافة في الإطار حيث من الممكن أن تكون إما: رأس، حفرة او خط مستقيم. نقوم بتطبيق عمليات الانسحاب أو التدوير وذلك لجعل الحافة تتطبق على محور الـ X كما هو موضح في الصورة:



الشكل(1)



الشكل(2)



الشكل(3)

الشكل (2) هو تدوير و انسحاب للحافة السماوية العليا و الشكل (3) هو تدوير و انسحاب للحافة الحمراء اليمنى. نوجد القيم الدنيا والعليا على المحور y لمصفوفة نقاط الحافة بعد المحاذاة و نحدد شكل الحافة وفق القواعد التالية:

- إذا كان الفرق بين أكبر وأصغر قيمة أقل من 20 نصنف الحافة على أنها مستقيمة
- إذا كانت القيمة الصغرى أقل من 25- نصنف الحافة على أنها حفرة
- فيما عدا ذلك نصنفها على أنها نتوء

نبدأ بعملية ترتيب القطع وفق التالي:

- إيجاد قطع الزوايا الأربع:

نوجد القطعة التي تمثل الزاوية العليا اليسارية من خلال البحث عن القطعة التي تكون حافتها العليا اليسارية مستقيمة و نوجد الزوايا الأخرى بطريقة مماثلة.

- إيجاد القطع الأخرى:

نعتمد في إيجاد القطع البقية على القطعة اليسارية و العليا للقطعة المراد إيجادها، حيث يتوجب أن تكون الحافة العليا للقطعة الحالية معاكسة للحافة السفلية للقطعة العليا و الحافة اليسرى للقطعة الحالية معاكسة للحافة اليمنى للقطعة اليسارية، و من بين جميع القطع التي تحقق الشرطين السابقين نختار القطعة التي حوافها لها تطابق لوني أكبر مع حواف القطعة العليا و اليسارية، لحساب هذا التطابق تم استخدام التابع `cross_correlation` على القنوات اللونية الثلاث وذلك بعد تحويلها من LAB إلى BGR.



Similarity Score: 103.61459750344706



Similarity Score: 148.19887509142566

النتائج:

أحجية تتالف من 6 قطع أبعادها (2,3):



- الدقة: %100
- الزمن المستغرق: 3.50 ثانية



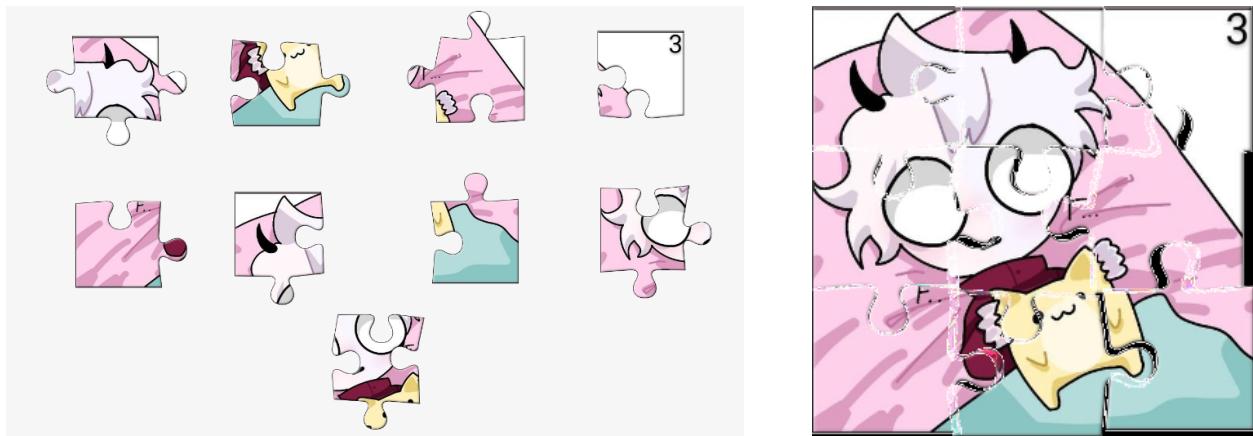
أحجية تتتألف من 6 قطع أبعادها (3,2) خلفيتها لون بسيط:

- الدقة: %100
- الزمن المستغرق: 5.86 ثانية.



أحجية تتتألف من 9 قطع أبعادها (3,3) خلفيتها لون بسيط:

- الدقة: %100
- الزمن المستغرق: 3.39 ثانية.



أحجية تتتألف من 16 قطعة أبعادها (4,4) خلفيتها لون بسيط:

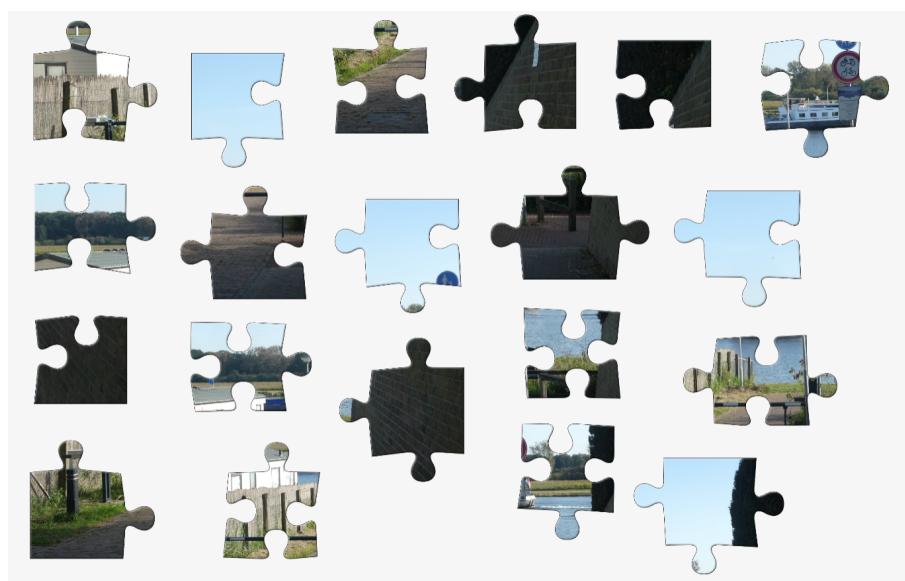
- الدقة: %100
- الزمن المستغرق: 15.76 ثانية.

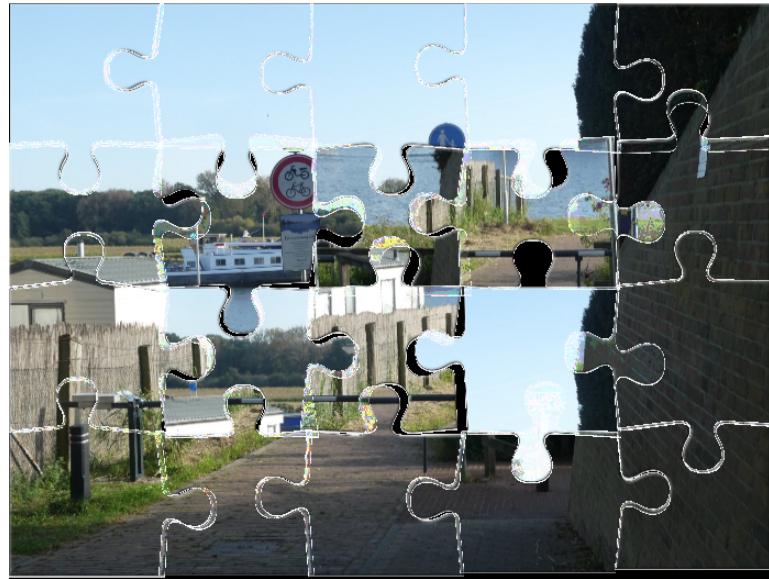




أحجية تتكون من 20 قطعة أبعادها (4,5) خلفيتها لون بسيط:

- الدقة: %70
- الزمن المستغرق: 15.75 ثانية.





حل المرحلة مع استخدام الـ **:hint**

قمنا بالبداية باستخراج القطع ولدينا الصورة الأصلية:

إعداد (**BFMatcher** و **SIFT**)

:SIFT_create

تُستخدم لإنشاء **SIFT**، والذي يكتشف النقاط المهمة في الصور (الميزات).

مطابق الميزات الذي يقارن النقاط بين صورتين : **(BFMatcher (Brute-Force matcher))**

استخراج الميزات **:(extract_features)**

تستخدم هذه الدالة لاكتشاف النقاط المهمة وصفاتها في الصورة باستخدام **SIFT**.

مطابقة الميزات **:(match_features)**

يتم تكرار كل زوج من التطابقات (حيث كل عنصر في الـ **matches** يحتوي على أقرب تطابقين).

في كل تكرار ، يتم التحقق من مدى قرب التطابق الأول **m** مقارنة بالتطابق الثاني **n**

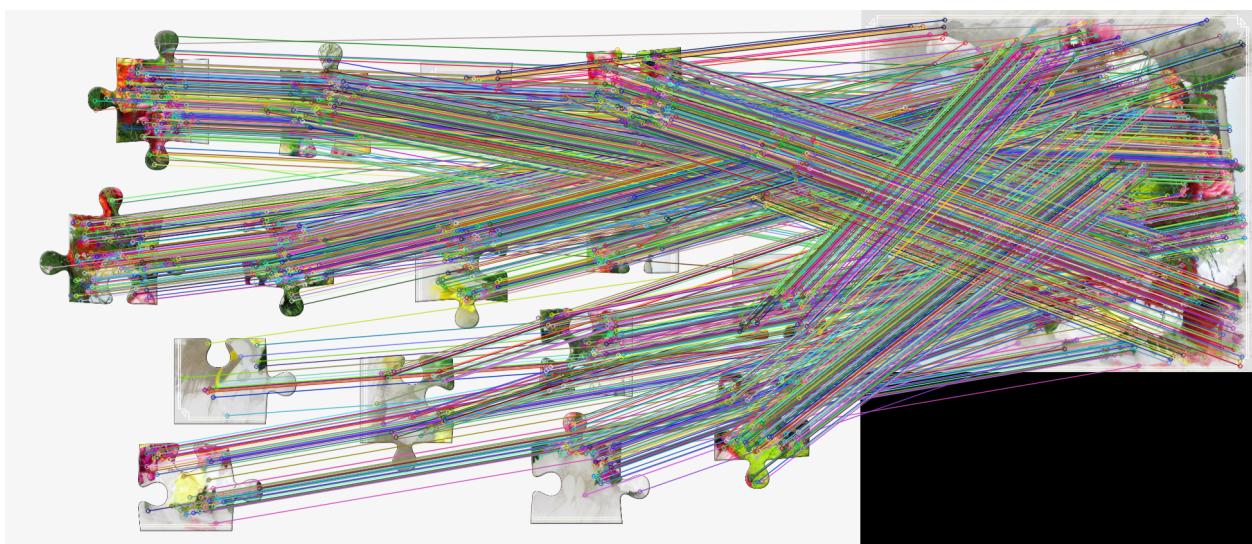
إذا كانت المسافة للتطابق الأول أقل من 0.75 أضعاف المسافة للتطابق الثاني، يُعتبر التطابق جيداً ويتم إضافته إلى `good_matches`.

هذه الدالة تعتبر جزءاً أساسياً في عملية مطابقة الميزات بين صورتين، حيث تسمح بتحديد التطابقات بين الصورتين.

رسم التطابقات :`(draw_matches)`

تُستخدم هذه الدالة لرسم خطوط توضح المطابقات بين النقاط المهمة في الصورتين.

ونرى النتيجة كما هو موضح في الصورة:



تحديد مكان قطعة من الأحجية :`(find_piece_placement_and_draw_matches)`

تحل هذه الدالة قطعة من الأحجية والصورة الكاملة لتحديد مكان القطعة عن طريق المطابقات.

أيضاً لدينا دالة :`find_homography`

الهدف منها هو إيجاد تحويل هندسي بين قطعة من الأحجية والصورة المرجعية. هذا التحويل يساعد في تحديد كيفية تناسب قطعة الأحجية داخل الصورة المرجعية. العملية تشمل عدة خطوات أساسية:
 يتم تحديد النقاط المتطابقة بين قطعة الأحجية والصورة المرجعية. هذه النقاط تمثل الأماكن المتماثلة في كلا الصورتين.

باستخدام هذه النقاط المتطابقة، يتم إجراء عملية تسمى "التحويل الهندسي" أو Homography. هذا التحويل يحدد كيف يمكن تحويل قطعة الأحجية لتطابق بدقة مع موقعها الصحيح في الصورة الكاملة.

يتم تقييم جودة التحويل الهندسي بناءً على عدد النقاط التي تتطابق بشكل جيد بين الصورتين (تُسمى inliers).

إذا كان عدد هذه النقاط يلبي معياراً محدداً (مثل أن يكون أكبر من عدد معين)، يتم اعتبار التحويل الهندسي دقيقاً.

إذا كان التحويل يلبي المعايير المحددة، يتم معرفة مصفوفة التحويل التي تشير إلى النقاط. إذا لم يكن التحويل كافياً أو لم يتم العثور على تطابقات كافية، يتم إرجاع قيمة فارغة أو None.

مثال:

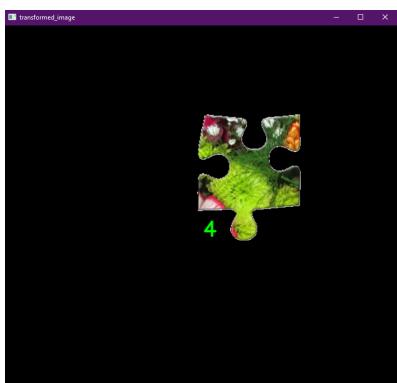
Piece 14: Homography matrix is

`[[1.28074100e+00 -9.74336851e-03 1.27890029e+02]`

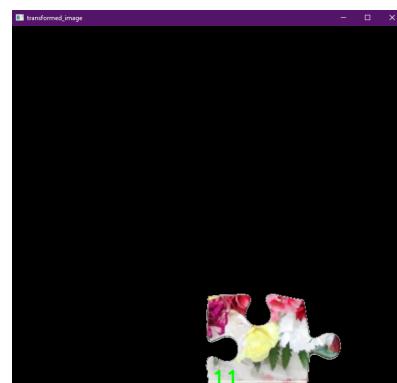
`[-3.43279919e-04 1.26442488e+00 2.81090234e+02]`

`[1.04022567e-06 -3.23299093e-05 1.00000000e+00]]`

إذا تم العثور على تحويل هندسي مناسب، يتم استخدامه لمعرفة موقع القطعة من خلال `cv2.warpPerspective` فيصبح لدينا صورة موجود بها موقع كل قطعة أي لكل قطعة يوجد صورة يوضح موقعها بالنسبة للصورة المرجعية كالتالي:



الشكل (2)

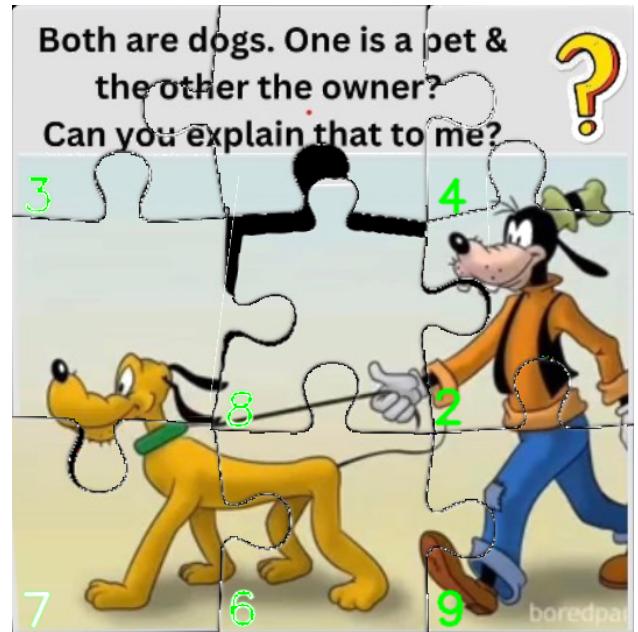


الشكل (1)

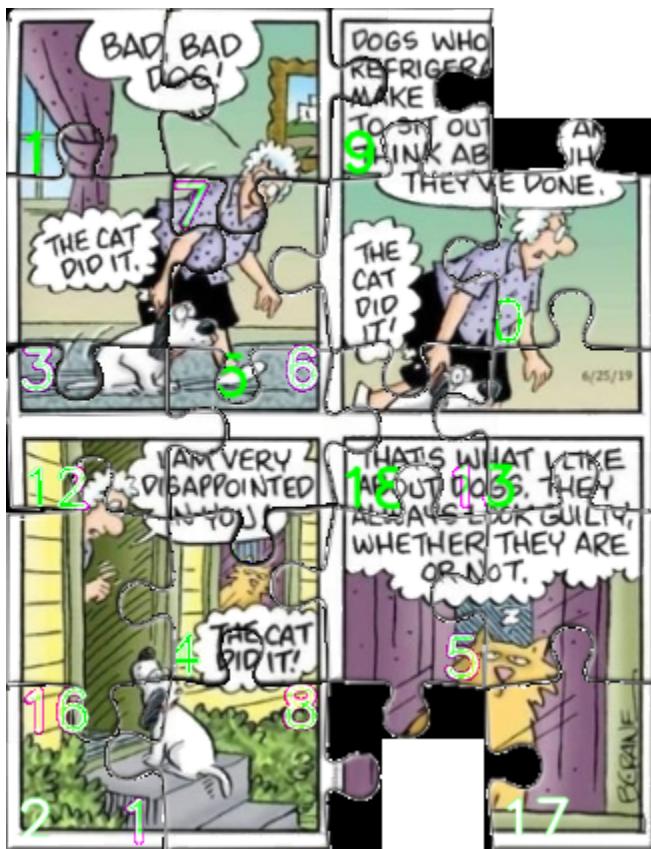
وفي النهاية عند مطابقة هذه الصور مع بعضها البعض وهنا لدينا أحجية عبارة عن 16 قطعة نلاحظ
النتيجة التالية:



وأيضاً النتائج التالية عند التجربة على أحجيات 9 قطع:



عندما تم زيادة عدد القطع أكثر من ذلك نلاحظ النتيجة التالية على أحجية مكونة من 20 قطعة:



نلاحظ هنا قد تم معرفة 18 قطعة من أصل 20.

أخيرا التجربة أيضا على أحجية من 24 قطعة قد تم معرفة 17 قطعة أيضاً:



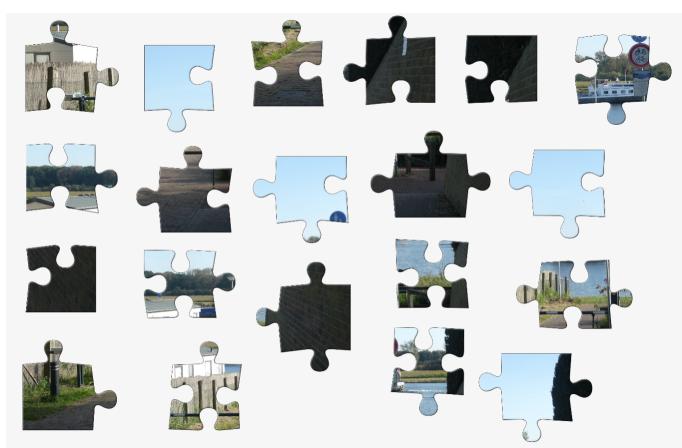
يمكن معرفة الدقة من خلال عدد ال Matrix التي كانت صحيحة:

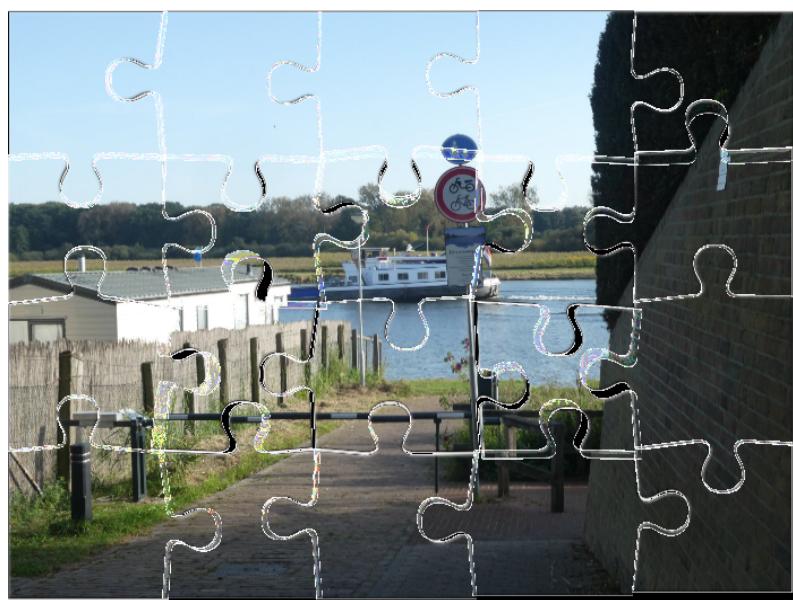
if matrix is not None:

```
puzzle_pieces_transformed.append(transformed_image)  
print("Number of pieces transformed:", len(puzzle_pieces_transformed))
```

النتائج النهائية بعد الإعتماد على شكل القطع و الصورة المساعدة في ترتيب القطع:

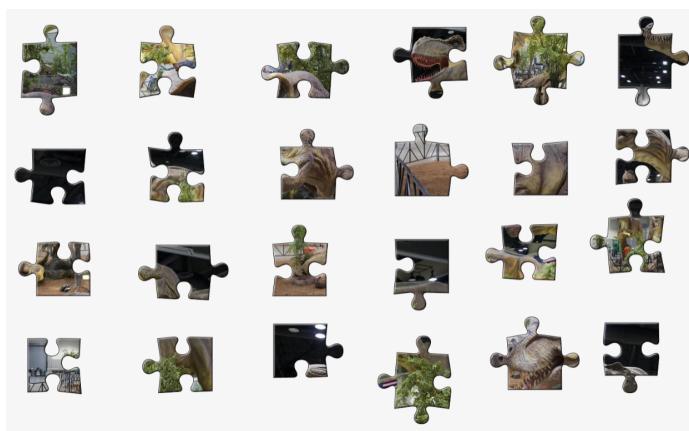
أحجية تتكون من 20 قطعة أبعادها (4,5) خلفيتها لون بسيط:

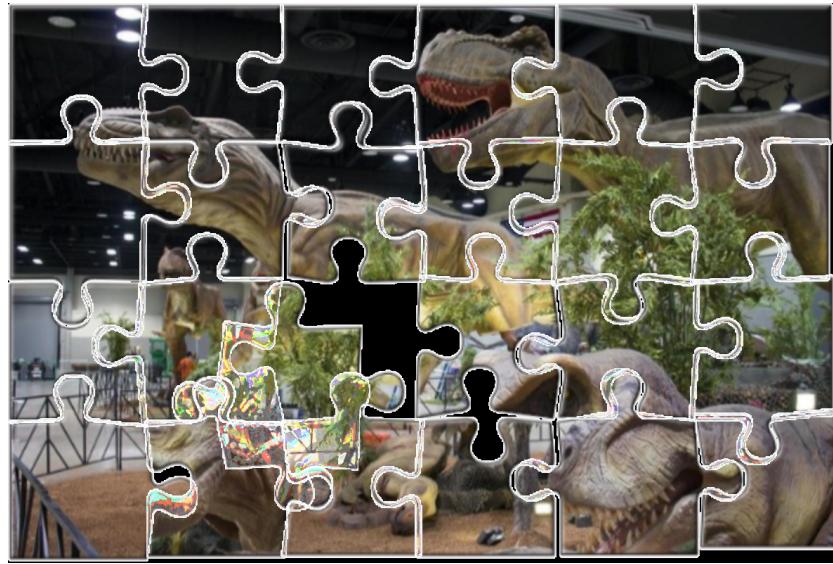




- الدقة: %100
- الزمن المستغرق: 16.10 ثانية

أحجية تتالف من 24 قطعة أبعادها (4,6) خلفيتها لون بسيط:





• الدقة: 83.33%
• الزمن المستغرق: 50.10 ثانية