جامعة دمشق
كلية الهندسة المعلوماتية
السنة الخامسة
قسم الذكاء الصنعي
الرؤية الحاسوبية

مشروع عملي مقرر الرؤيا الحاسوبية 2023

Photo Editor using Hand Gestures

تقديم الطالبات:

ايمان الغدير

تسنيم عجاج

راما ريحاوي

رغد الحلبي

المهندس المشرف:

م. أيمن خوالدة

الخميس 2022/12/22

الشق البرمجي:

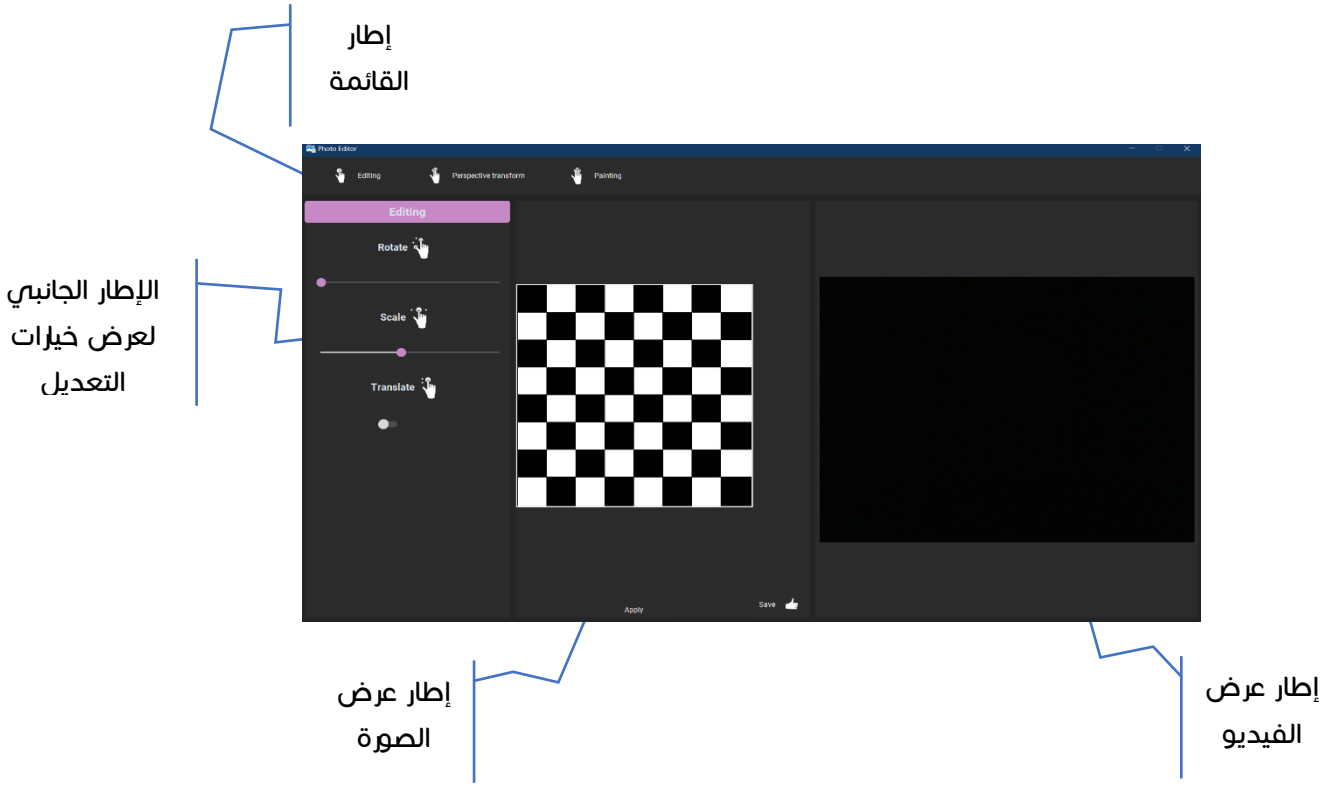
استخدمنا Tkinter في بناء الواجهة

splash_screen:

تابع لتهيئة وعرض الواجهة التعريفية لمدة قصيرة، وبعد انتهاء الزمن يتم استدعاء upload_action لتحميل الصورة التي سيتم العمل عليها.

__init__:

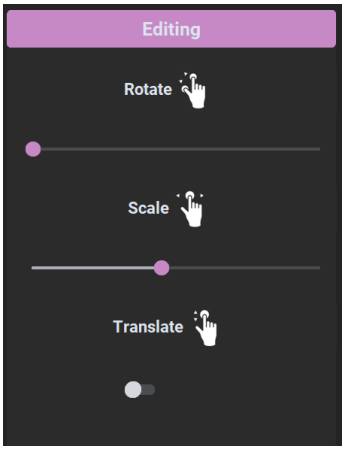
تابع الـ initialize لتهيئة الواجهة الرئيسية، خصائص الواجهة بالإضافة لـ frames.



Editing_action:

يُنفذ عند اختيار editing ويهيئ الإطار الجانبي لخيارات التعديل.

فيه slider التدوير من 0 لـ 360 درجة والتحجيم من 10% لـ 200% وswitchg لتفعيل التحريك



translation_action:

تابع التحريك، عند الضغط بالماوس على الصورة يستدعي تابع start_translation يأخذ فيه إحداثيات النقرة،

وعند تحريك الماوس يتفعل تابع translation لتطبيق التحريك على الصورة.

يعني التحريك بشكل أساسي أننا نغير الصورة عن طريق إضافة/طرح إحداثيات X وY من أجل القيام بذلك، نحتاج إلى إنشاء مصفوفة تحويل، كما هو موضح على النحو التالي:

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

rotate_action:

تابع الدوران، وهو أيضًا شكل من أشكال التحويل، ويمكننا تحقيقه باستخدام مصفوفة التحويل التالية:

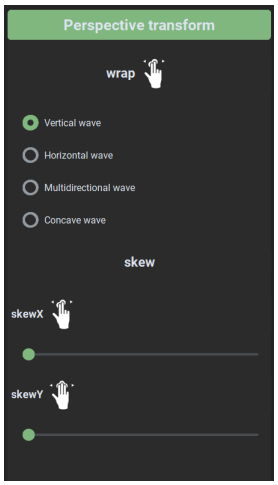
$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

scale_action:

التحجيم، تغيير حجم الصورة تصغير أو تكبير.

Perspective_action:

تهيئة لخيارات wrap وskew



Painting_action:

يهيئ للرسم، فيه زر اختيار اللون، slider لسماكة الفرشاية بالرسم، تفعيل المحي وslider لسماكة الممحاة.

عند الضغط على الصورة يتفعل تابع start_draw وعند التحريك draw.

start_draw يأخذ إحداثيات النقرة وهم إحداثيات البداية

draw:

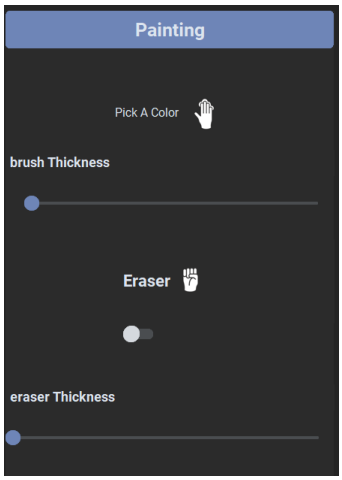
يتم الرسم على الـ canvas وعلى الصورة

الـ canvas image صورة سوداء بحجم الصورة التي سنرسم عليها.

نأخذ threshold binary inverse تصبح بيضاء والرسم بالأسود عليها (لو مرسوم اسود يصبح ابيض كأنه غير مرسوم أي يمحي)

الـ bitwise_or بين canvas والصورة النتيجة الصورة عليها اللون المرسوم، وهنا يتم عرض النتيجة وحفظها

الـ bitwise_and يتحول اللون المرسوم للون الأسود للمحي.



apply_action: لحفظ التغيير

save: للحفظ بالمكان والاسم المطلوب

الشق الذكي:

تابع open_camera ضمن Frontend class:

هو التابع المسؤول عن تسلسل عمليات معالجة الفيديو.

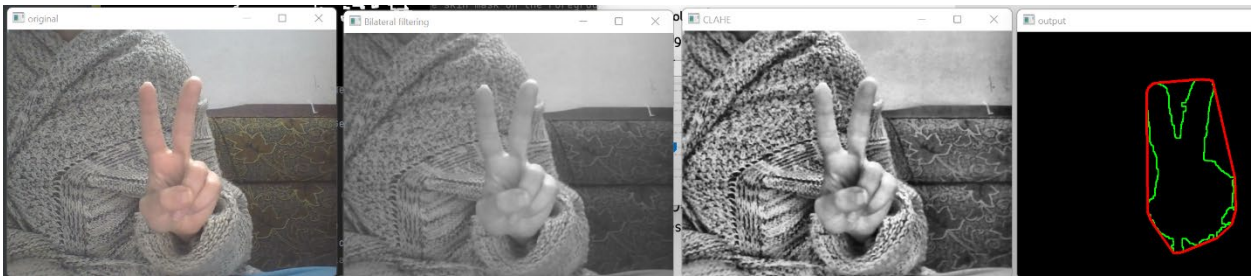
1- التحويل إلى النظام اللوني YCrCb والذي ثبت أنه الأفضل في التعامل مع البيئات ذات ظروف الإضاءة المختلفة من خلال معالجة مركبة الـ y وهو ما سنتطرق له لاحقاً.
2- تطبيق الـ skinModel لكشف المناطق التي تحتوي على لون الجلد.
3- تطبيق الـ nonFixedForegroundModel على نتيجة skinModel السابق لكشف المناطق المتحركة واعتبارها كـ foreground.
4- تطبيق تابع mergeTwoModels لدمج نتائج المعالجة لكلا الـ models.
5- تطبيق الـ preparePreGesturesProcessing لكشف الـ features من اليد contouring والتعرّف على الأصابع وعددها وتحديد الإجراء المناسب translate, rotate, scale.
6- تطبيق implementGesture تبعاً لـ gesture التي تمّ التعرّف عليها.

تابع skinModel ضمن Processing class:

<u>خطوات الـ Preprocessing:</u>

بعد تجريب الكثير من الخيارات وترتيب العمليات بأشكال مختلفة توصّلنا إلى أنّ النتائج التالية هي الأفضل:

1- معالجة الإضاءة ودقّة الكاميرا من خلال تابع getMergedImageAfterEditingY:
   1. من المعروف أنّ كاميرات الحواسيب عامّةً لا تعطي دقّة جيدة بما يكفي وتسبّب ضجيجاً بشكل كبير ولذلك عزلنا بدايةً مركّبة الـ Y وتطبيق bilateral filter عليها وهو blur filter له تأثير smoothness على الـ frame ولكن مع المحافظة على الـ edges.
   2. ومن ثمّ طبّقنا Contrast Limited Adaptive Histogram Equalization (CLAHE) على مركّبة y لمعالجة تباين الصورة الناتج عن ظروف الإضاءة المختلفة ومن ثمّ تمّ دمج قنوات الصورة المختلفة.



2- ومن ثمّ وباعتبار أنّ المسألة تتضمن بيئات مختلفة بالإضاءة ومن أجل نتائج أفضل أتحنا إمكانيّة للمستخدم أن يعدّل على قيم Y,Cr,Cb ضمن min-max ranges مع تثبيت قيم بدائيّة معيّنة لهذه الـ ranges تعبّر عادة عن القيم اللونية للجلد.

خطوات الـ Processing:

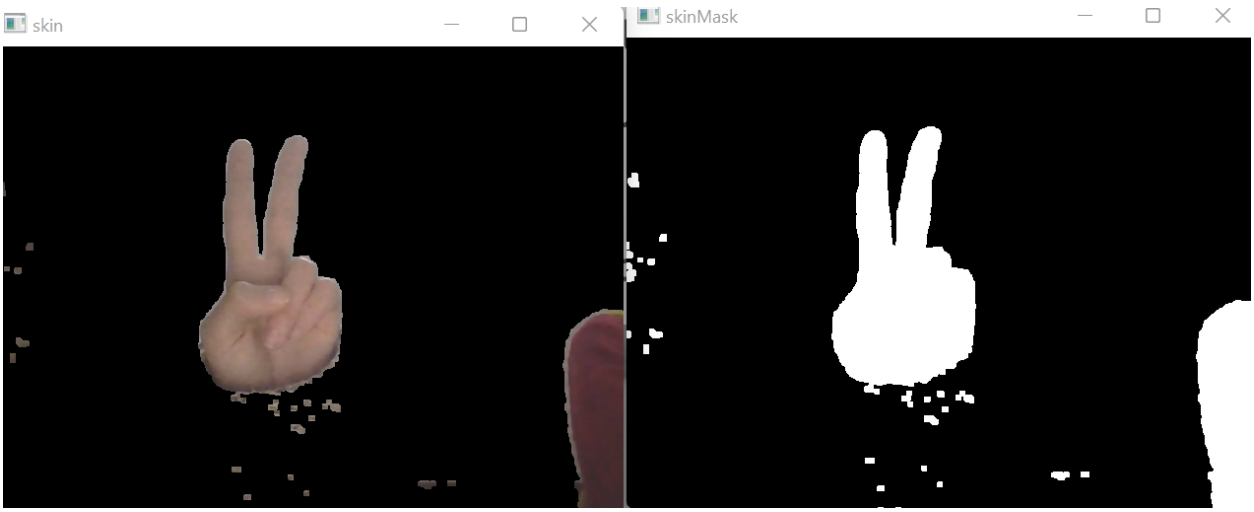
1- الحصول على skin لوحدها ضمن الصورة من خلال تابع maskSkin:
   1. من القيم اللونية السابقة يتمّ بناء صورة جديدة تحتوي فقط على القيم اللونية المطلوبة أما بقيّة أجزاء الصورة ستكون باللون الأسود وهي التي يمكن للمستخدم التعديل بقيم Y, Cr, Cb حتى يصبح الـ skinMask مناسباً.

ملاحظات:

قمنا بدايةً بجعل عمليّة تعديل القيم محصورة فقط ببداية البرنامج ومن ثمّ يتم تثبيتها ولكن افترضنا أنّ ظروف الإضاءة من الممكن أن تتغير ببساطة وبهذا يمكن للمستخدم حريّة التعديل متى شاء.

خطوات الـ Preprocessing:

1- طبّقنا التحويلات المورفولوجية بالترتيب التالي بعد تجريب العديد من الخيارات:
   Close then Dilation then Open the BilateralFilter

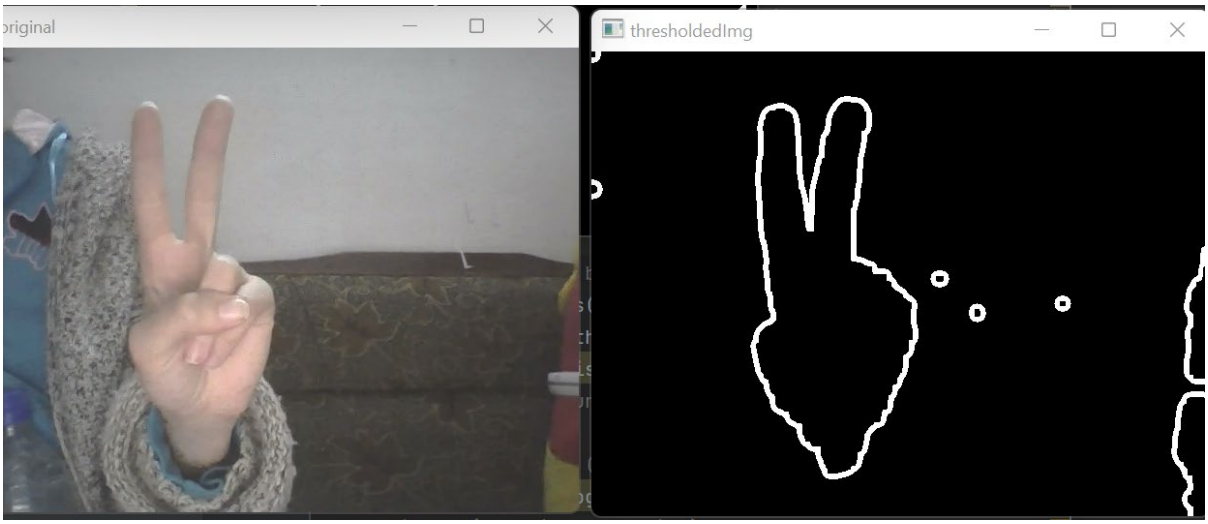


تابع notFixedForegroundBackgroundModel ضمن Processing class:

خطوات الـ Preprocessing:

1- معالجة الإضاءة:

1. على الرغم من معالجة هذه المسألة سابقاً إلا أننا ارتأينا أنّه من الأفضل هنا التعامل مع binary image لمعالجة الإضاءة بالطريقة الأمثل، ولذلك طبّقنا adaptiveThresholding الغاوسي.



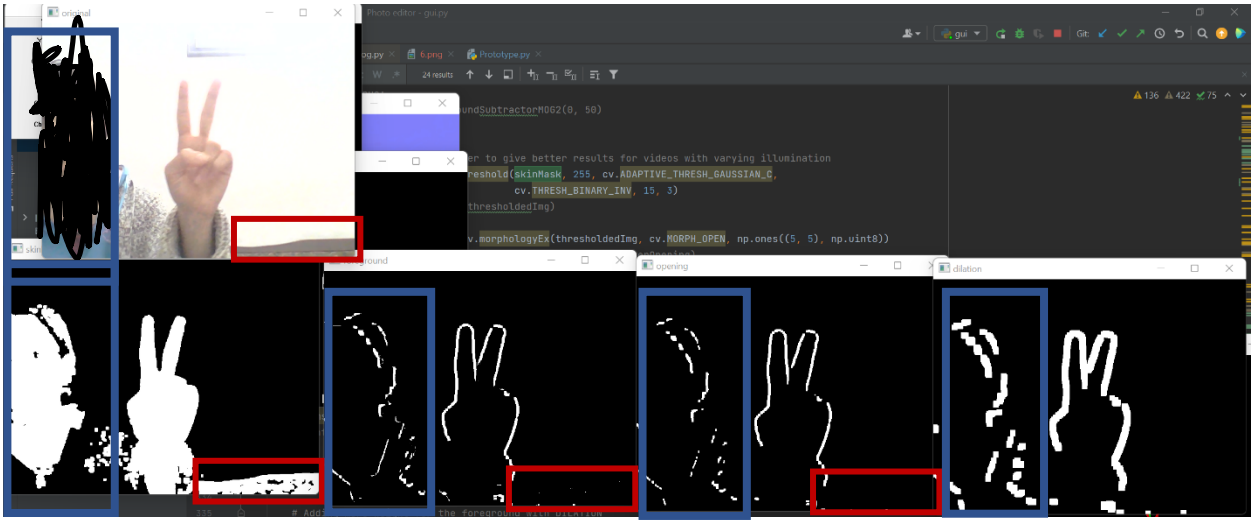
2. ومن المعروف أنّ العمليّة السابقة سوف تسفر عن ضجيج كبير يجعل من القيم اللونية التي ستعتبر بيضاء أحيانا وسوداء أحيانا أخرى تبدو كما لو أنّها عناصر متحركة لذلك طبقنا التحويل المورفولوجي open للحصول على نتيجة أفضل.

خطوات الـ Preprocessing:

1- أخيراً طبقنا Mixture Of Gaussians Background subtractor مع عدم تحديد الـ shadows وتطبيق عملية bitwise-and على الـ thresholded image مع الـ mask الناتج عن الـ subtractor وبهذا حصلنا على حدود اليد فقط (دون تعبئة) ولن تظهر العناصر التي تمتلك لوناً آخر يمثل لون الجلد طالما أنّه غير متحرّك فهو يمثّل الخلفية.

خطوات الـ Postprocessing:

1- بما أننا حصلنا فقط على حدود اليد فقد طبقنا التحويليلن المورفولوجيين open then dilate وذلك لتوسيع حجم الحدود حتى يتم التقاطها بسهولة في المراحل اللاحقة.



ملاحظات:

1- جرّبنا تطبيق تابع interpolate لاستيفاء خطوط اليد التي تمّ رسمها بالشكل الصحيح كما يمكن تطبيق canny للعثور على الـ edges وذلك للحصول على نتائج أفضل، لكن لم يسعفنا الوقت لتجريب نتائج هاتين العمليتين.
2- إنّ الاعتماد على طريقة فصل الخلفية باستخدام MOG2 رغم محاسنها تتضمّن العديد من المساوئ في robustness لـ model لذلك من الممكن تحديد تطبيقه كلّ (عدد معين من الفريمات) أو العودة إلى الطرق التقليدية لفصل الخلفية مثل اعتبار أول فريم هي الخلفية وحذفها دائماً ولكن هذا يجعل البيئة fixed ولا يمكن أن يدخل أي عنصر جديد لها ولا حتّى تغيير المستخدم لمكانه، أو أخذ كل الصورة السابقة للصورة الحالية كخلفية ولكن هذا الحل يسبب كثيراً من الضجيج بسبب التغييرات الطفيفة بالصورة الناتجة عن تعديل مستوى الإضاءة بشكل تلقائي من قبل الكاميرا.
3- من أجل تطبيق عمليّة tracking جرّبنا تطبيق خوارزمية **camshift** ولكن كانت النتائج أسوأ وذلك لأنها تعتمد فقط على اللون في عمليّة tracking ولم نجد الوقت الكافي لتحسين نتائجها فقمنا بإلغائها.

تابع preparePreGesturesProcessing ضمن Processing class

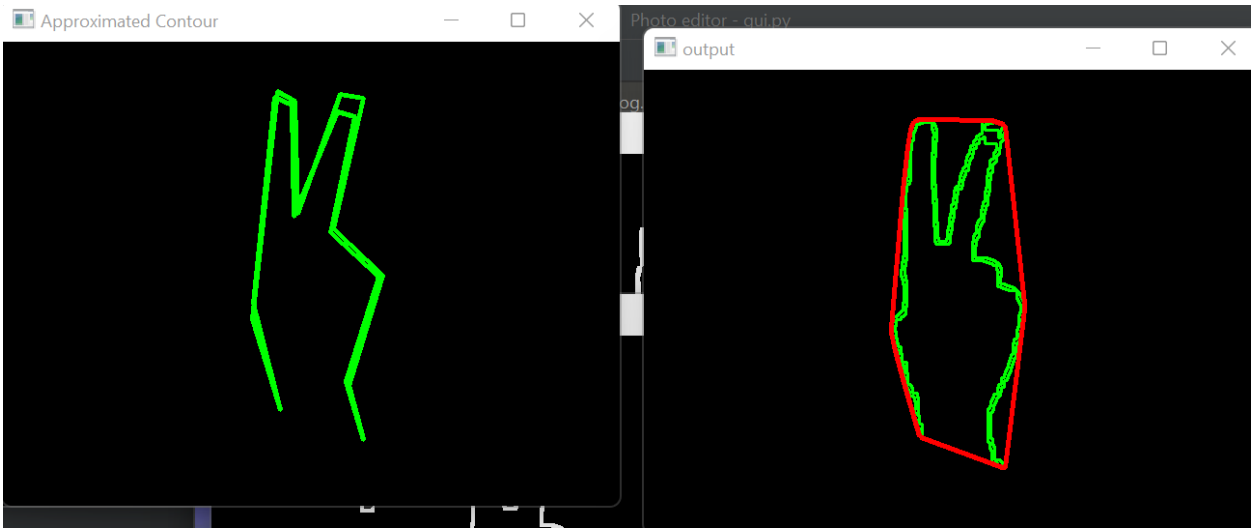
1- الحصول على features لتمييز اليد من خلال تابع getMaxCon:

اعتمدنا على حساب contours لصورة المعالجة وافتراض أنّ اليد تحمل الـ max contour والمفترض ألا نعاني من مشكلة ظهور الوجه ضمن الفيديو طالما أنّ الرأس لا يتحرك، لكن ستظهر بعض المشاكل الناتجة عن الـ learning rate في تابع MOG2 بالمقابل.

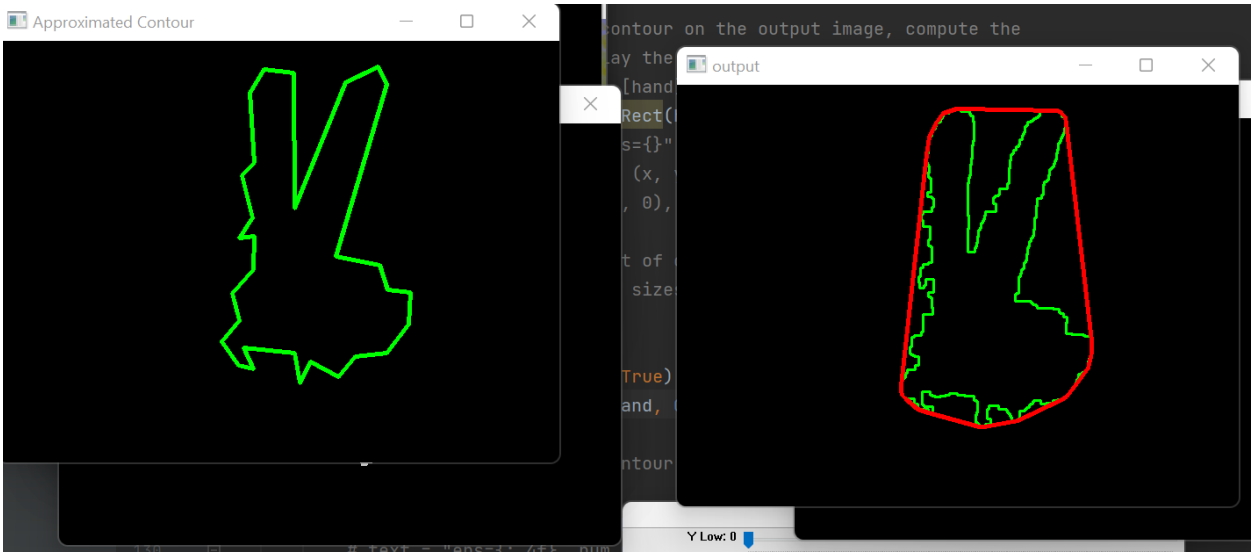
<u>ملاحظات:</u>

جرّبنا تطبيق تابع approxPolyDP لتقريب الـ contour المرسوم حول اليد إلى مجموعة أضلاع للتخلص من التعرّجات الزائدة ويمكن من خلاله الحصول على نتائج أفضل.

عند حذف الخلفيّة من خلال MOG في كل فريم ومع معدل تعلم 4:



دون حذف الخلفيّة:

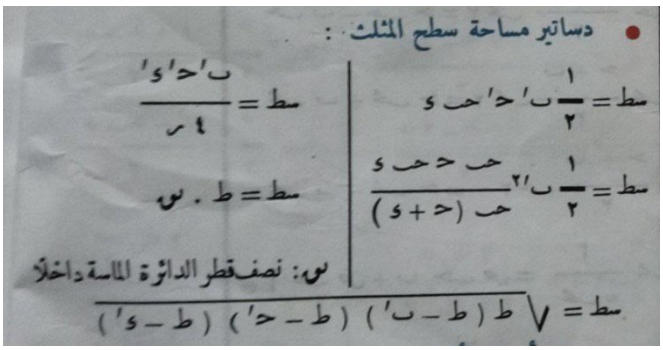


2- ومن ثمّ نطبّق تابع findFingers وبدايةً نحن بحاجة لرسم الـ convexHull ثم حساب الـ convexity defects للحصول على المثلثات الواقعة بين الـconvexHull والـcontour، حيث نأخذ احداثيات المثلثات الناتجة من الـcontours ونقوم بحساب أطوال أضلاعه والزاوية المقابلة لضلع المثلث الواقع على الـhull. وفي حال كانت الزاوية أقل من 90 درجة يتم اعتبارها متبوعة باصبع وزيادة عدد الاصابع واحداً.

3- نطبق تابع centroid نحسب مركز الـcontour باستخدام تابع الـ cv.moments.

4- نطبق تابع findFarPoint لايجاد ابعد نقطة عن المركز والتي تمثل رأس الاصبع، والتي ستمكننا من تتبع حركة الاصبع للقيام بعملية الرسم او المحي او غيرها، لايجادها نستخرج قيمة الـ s لكافة المثلثات ونحسب المسافة بينها وبين مركز اليد الناتج من التابع السابق، ونأخذ أطول مسافة بينها ونحصل لى موقعها من الـ contour (بإمكاننا الحصول على رؤوس الاصابع كافة أيضا من الـdefects من خلال مركبة الـ s التي تقابل رؤوس الاصابع).

القوانين المستخدمة لحساب الزوايا:



$$a = \sqrt{s * \left(\frac{s}{2} - a\right) * \left(\frac{s}{2} - b\right) * \left(\frac{s}{2} - c\right)}$$

$$angel = \frac{Acos\sqrt{b^2 + c^2 - a^2}}{2bc}$$

1- بعد الحصول على ال features السابقة أصبح بإمكاننا تحديد الوضعية عن طريق ربطها بعدد معين من الاصابع، وهذا ما يقوم به تابع recognizeGesture الذي يقوم بتفعيل الوضعية عن طريق عدد الاصابع التي تم الحصول عليه سابقاً.

تابع implementGesture ضمن Frontend class:

يأخذ ال features التي تم استنتاجها بالتابع السابق، ويستخدمها بتحديد دخل التوابع المسؤولة عن تحقيق العمليات المرتبطة بالوضعية المفعلة.

1- في حال كانت الوضعية المفعلة بعدد اصابع 2 (الذي تم استنتاجه من تابع findFingers)، فالعملية هي translation فيقوم بتفعيل الزر المسؤول عن تفعيل خيار translate ثم أخذ احداثيات ال prev farthest point وطرحها من احداثيات farthest point وإرسال النقطة الجديدة كبارامتر لتابع translation الذي تم شرحه سابقا لاستخدامهما ضمن مصفوفة translate.

رابط المشروع على ال github:

Photo-Editor-using-Hand-Gesture

المراجع:

Papers:

[1] Sci-Hub | Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. Multimedia Tools and Applications, 74(8), 2687–2715 | 10.1007/s11042-013-1501-1

[2] Kinect Sensor-Based Long-Distance Hand Gesture Recognition and Fingertip Detection with Depth Information (hindawi.com)

[3] J. Imaging | Free Full-Text | Hand Gesture Recognition Based on Computer Vision: A Review of Techniques (mdpi.com)

[4] Hand Gesture Recognition Using OpenCv and Python | Request PDF (researchgate.net)

[5] (PDF) Appearance based and Gesture Independent Model for Natural Human Hand Detection and Tracking (researchgate.net)

[6] (PDF) Hand gestures recognition with improved skin color segmentation in human-computer interaction applications (researchgate.net)

[7] Motion segmentation and pose recognition with motion history gradients | IEEE Conference Publication | IEEE Xplore

Codes:

[1] Face Detection Using Skin Tone Threshold(RGB-YCrCb): Python Implementation. | by Mahmoud Harmouch | The Startup | Medium

[2] Finger Detection and Tracking using OpenCV and Python | by Amar Prakash Pandey | Medium

[3] Finger Detection and Tracking using OpenCV and Python | Amar Prakash Pandey (amarpandey.me)

[4] Motion Detection Techniques (With Code on OpenCV) | by Safa Abbes | Medium

[5] Detecting Motion with OpenCV — Image Analysis for Beginners | by Mike Huls | Towards Data Science

[6] Sadaival/Hand-Gestures (github.com)

[7] Drawing With Fingers (Python + OpenCV) - YouTube

tools

[1] OpenCV Contour Approximation - PyImageSearch

[2] c++ - OpenCV2 skin segmentation with back projection - Stack Overflow

[3] YCrCb Values For Various Colors (tvone.com)

[4] OpenCV: cv::VideoCapture Class Reference

[5] OpenCV: Back Projection

[6] OpenCV: Histogram - 4 : Histogram Backprojection

[7] image - What the function calcHist() give us - Stack Overflow

[8] Parallel Processing in Python - GeeksforGeeks

[9] Track objects with Camshift using OpenCV - GeeksforGeeks

[10] Computer Vision 2023 lectures