**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**Khoa ĐIỆN - ĐIỆN TỬ**

**Bộ môn ĐIỆN TỬ-VIỄN THÔNG**

**------------o0o------------**



**BÁO CÁO BÀI TẬP LỚN**

**MÔN XỬ LÍ ẢNH**

**Đề tài: BIẾN ĐỔI HÌNH HỌC TRÊN ẢNH**

(GEOMETRIC TRANSFORMATIONS OF IMAGE)

***GVHD:* Hà Hoàng Kha**

***Lớp:* A02, Thứ 3 tiết 8, 9, 10 hàng tuần**

|  |  |  |
| --- | --- | --- |
| ***STT*** | ***Họ và tên*** | ***MSSV*** |
| **1** | **Trần Nhật Phương** | **1712736** |
| **2** | **Bùi Văn Quốc** | **1712846** |
| **3** | **Trần Trịnh Quang** | **1712803** |

*Tp. Hồ Chí Minh, ngày 16 tháng 06 năm 2020*

**I,Giới thiệu chung**

Xử lý ảnh (XLA) là đối tượng nghiên cứu của lĩnh vực thị giác máy, là quá trình biến đổi từ một ảnh ban đầu sang một ảnh mới với các đặc tính và tuân theo ý muốn của người sử dụng. Xử lý ảnh có thể gồm quá trình phân tích, phân lớp các đối tượng, làm tăng chất lượng, phân đoạn và tách cạnh, gán nhãn cho vùng hay quá trình biên dịch các thông tin hình ảnh của ảnh. Cũng như xử lý dữ liệu bằng đồ hoạ, xử lý ảnh số là một lĩnh vực của tin học ứng dụng. Xử lý dữ liệu bằng đồ họa đề cập đến những ảnh nhân tạo, các ảnh này được xem xét như là một cấu trúc dữ liệu và được tạo bởi các chương trình. Xử lý ảnh số bao gồm các phương pháp và kỹ thuật biến đổi, để truyền tải hoặc mã hoá các ảnh tự nhiên. Mục đích của xử lý ảnh gồm:

+ Biến đổi ảnh làm tăng chất lượng ảnh.

+ Tự động nhận dạng ảnh, đoán nhận ảnh, đánh giá các nội dung của ảnh.

Nhận biết và đánh giá các nội dung của ảnh là sự phân tích một hình ảnh thành những phần có ý nghĩa để phân biệt đối tượng này với đối tượng khác, dựa vào đó ta có thể mô tả cấu trúc của hình ảnh ban đầu. Có thể liệt kê một số phương pháp nhận dạng cơ bản như nhận dạng ảnh của các đối tượng trên ảnh, tách cạnh, phân đoạn hình ảnh,… Kỹ thuật này được dùng nhiều trong y học (xử lý tế bào, nhiễm sắc thể), nhận dạng chữ trong văn bản.

Xử lí ảnh có rất nhiều mảng và ứng dụng riêng cho từng phần, và nhóm em đã lựa chọn đề tài Geometric Transformation of Image (Biến đổi hình học trên ảnh) để làm bài tập lớn của nhóm. Trong đề tài này chúng em sẽ trình bày nội dung lý thuyết của việc biến đổi hình học trên ảnh, các code ứng dụng trong việc biến đổi hình học trên ảnh, và một phần ứng dụng của bọn em trong việc biến đổi ảnh nhằm tách khuôn mặt từ việc phát hiện khuôn mặt ra từ một tấm ảnh có nhiều người nhằm hỗ trợ cho việc nhận dạng người hỗ trợ trong mảng trí tuệ nhân tạo (AI).

Trong đề tài chúng em thực hiện thông qua ngôn ngữ lập trình Python và với thư viện OpenCV dùng cho Python. Trong lúc thực hiện còn nhiều sai sót mắc phải mong thầy xem xét và góp ý hộ nhóm em để bọn em có thể hoàn thiện hơn

**II, Nội dung lý thuyết**

***1, Mục đích***

Trong nội dung này chúng em sẽ đề cập đến các phép biến đổi hình học khác nhau cho hình ảnh như dịch chuyển một bức ảnh, xoay bức ảnh theo ý muốn hoặc biến đổi affine, v.v

Ngoài ra sẽ có nội dung code Python áp dụng cho từng phép biến đổi khác nhau.

***2, Các phép biến đổi***

Như bạn thấy trong đời sống hằng ngày trong lĩnh vực xử lí ảnh, có rất nhiều phép biến đổi hình học như phép dịch chuyển ảnh theo trục các trục tọa độ ngang và dọc hoặc thậm chí là phóng to thu nhỏ một bức ảnh, hay xoay ngang xoay dọc theo ý của người sử dụng.

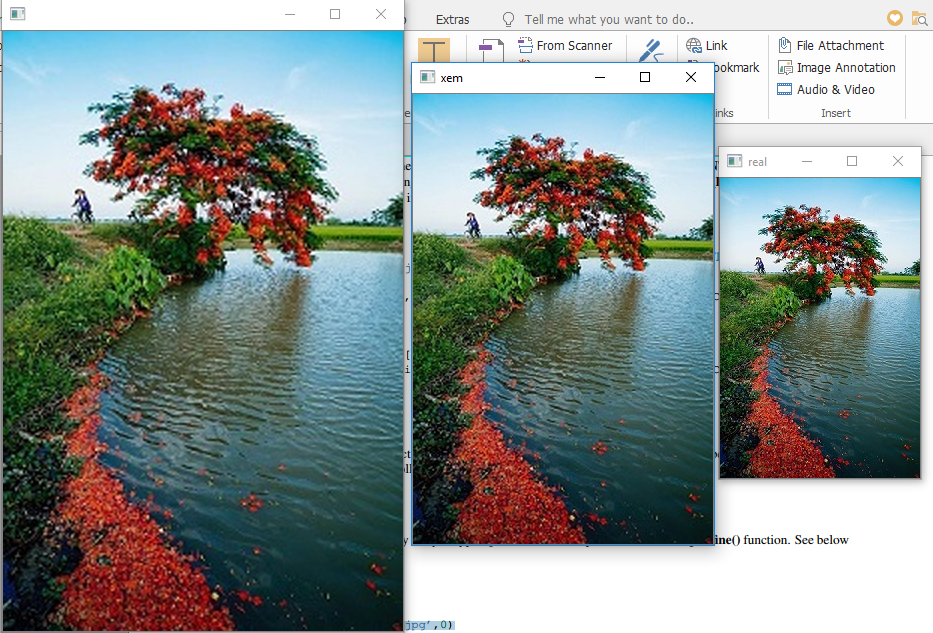
*a, Định kích thước*

Trong đời sống ngày nay với một bức ảnh ta có thể in nó ra với nhiều kích thước khác nhau, có người thích ảnh to, có người thích ảnh nhỏ, ngoài ra trong kĩ thuật có nhiều trường hợp chúng ta cần thay đổi kích thước của tấm ảnh thu được để có các kết quả chính xác cho từng chuẩn đoán ảnh.

Trong OpenCV được xử lý trong Python chúng ta có hàm cv2.resize() cho mục đích định kích thước cho ảnh. Kích thước của một hình ảnh có thể được chỉ định bằng tay hoặc bạn có thể chỉ định chúng thông qua hệ số tỷ lệ trong các thông số trong hàm resize. Đoạn code sau giúp ta có thể chỉnh sửa ảnh:

import cv2import numpy as npimg = cv2.imread(’messi5.jpg’)  
res = cv2.resize(img, None, fx=2, fy=2, interpolation = cv2.INTER\_CUBIC)  
#ORheight, width = img.shape[:2]  
res = cv2.resize(img, (2\*width, 2\*height), interpolation = cv2.INTER\_CUBIC)

Trong hàm trên các thông số fx hay width giúp ta chỉnh sửa chiều ngang của tấm ảnh to ra hay hẹp lại, còn fy và height giúp ta chỉnh sửa chiều dọc. Sau đây là ví dụ về một ảnh được thu nhỏ với các thông số khác nhau



Ở thông số cv2.INTER\_CUBIC, đây là thông số nội suy của ảnh. Có nhiều các thông số hay phương pháp nội suy khác nhau có thể được sử dụng. Các phương thức nội suy ưu tiên là cv2.INTER\_AREA để thu nhỏ và cv2.INTER\_CUBIC (chậm) & cv2.INTER\_LINEAR để thu phóng. Theo mặc định, phương pháp nội suy được sử dụng là cv2.INTER\_LINEAR cho tất cả thay đổi kích thước cho cả 2 mục đích.

*b, Dịch chuyển*

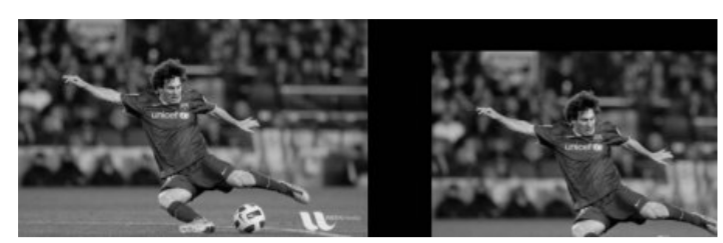
Dịch chuyển ảnh tức là ta di chuyển ảnh theo chiều dọc ngang (2 chiều) theo ý muốn. Ta cũng có thể xem dịch chuyển là thay đổi vị trí của đối tượng trong ảnh đến vị trí mới mà ta muốn đặt. Giả sử ta dịch chuyển ảnh theo hướng (x, y) theo không gian 2 chiều thì ta có thể tạo ra ma trận biến đổi ảnh M như sau:

Ta có thể xét ví dụ dưới đây để biến ma trận trên thành một mảng Numpy dạng np.float32. Xét ví dụ dưới đây với chiều không gian là (100, 50). Và hàm ta sử dụng là hàm cv2.warpAffine

import cv2import numpy as npimg = cv2.imread(’messi5.jpg’,0)  
rows,cols = img.shape  
M = np.float32([[1,0,100], [0,1,50]])  
dst = cv2.warpAffine(img,M,(cols,rows))  
cv2.imshow(’img’,dst)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

Lưu ý: Đối với đối số thứ 3 của hàm cv2.warpAffine () là kích thước của hình ảnh đầu ra, phải ở dạng (chiều rộng, chiều cao). Ở đây ta xét chiều rộng = chiều ngang và chiều cao = chiều cao của ảnh gốc.

Ta có kết quả thu được như hình sau:



*c, Xoay ảnh*

Trong nhiều trường hợp ảnh ta thu được sẽ bị ngược hoặc nằm ngang so với chiều dọc mà ta muốn thu được. Chính vì thế mà ta cần tới việc xoay hình ảnh lại theo một góc nào đó để có thể thu được ảnh theo chiều mà ta muốn có. Ở đây ta có thể xét một ma trận cho việc xoay hình ảnh theo một góc mà ta muốn như sau:

Đó là trên mặt lý thuyết còn đối với OpenCV, module này cung cấp cho ta khả năng xoay theo tỷ lệ nhất định:

Với: và :

Để tìm ma trận biến đổi này, OpenCV cung cấp một hàm, cv2.getRotationMatrix2D. Kiểm tra ví dụ dưới đây để xoay hình ảnh 90 độ so với trung tâm mà không có tỷ lệ.

img = cv2.imread(’messi5.jpg’,0)  
rows,cols = img.shape  
M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)  
dst = cv2.warpAffine(img,M,(cols,rows))

Xem kết quả:



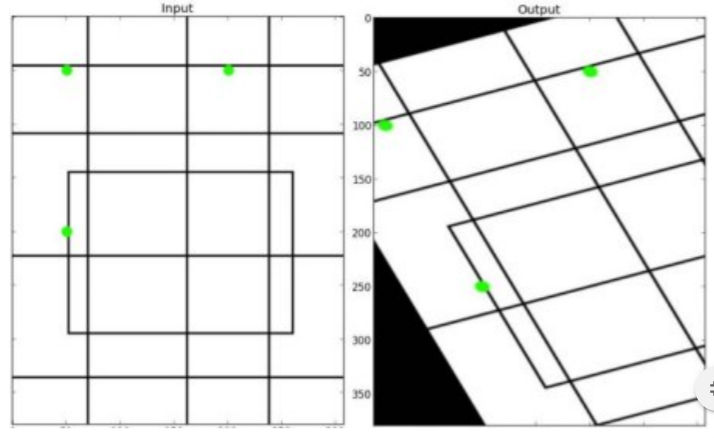
*d, Biến đổi Affine*

Trong phép biến đổi affine, tất cả các đường song song trong ảnh gốc sẽ vẫn song song trong ảnh đầu ra. Để tìm ma trận biến đổi, chúng ta cần ba điểm từ hình ảnh đầu vào và vị trí tương ứng của chúng trong hình ảnh đầu ra. Sau đó, cv2.getAffineTransform sẽ tạo ra một ma trận 2x3 sẽ được chuyển đến cv2.warpAffine.

Kiểm tra ví dụ bên dưới và cũng xem xét các điểm tôi đã chọn (được đánh dấu màu Xanh lục):

img = cv2.imread(’drawing.png’)  
rows,cols,ch = img.shape  
pts1 = np.float32([[50,50],[200,50],[50,200]])  
pts2 = np.float32([[10,100],[200,50],[100,250]])  
M = cv2.getAffineTransform(pts1,pts2)  
dst = cv2.warpAffine(img,M,(cols,rows))  
plt.subplot(121),plt.imshow(img),plt.title(’Input’)  
plt.subplot(122),plt.imshow(dst),plt.title(’Output’)  
plt.show()

Xem kết quả:



**Phối cảnh chuyển đổi**

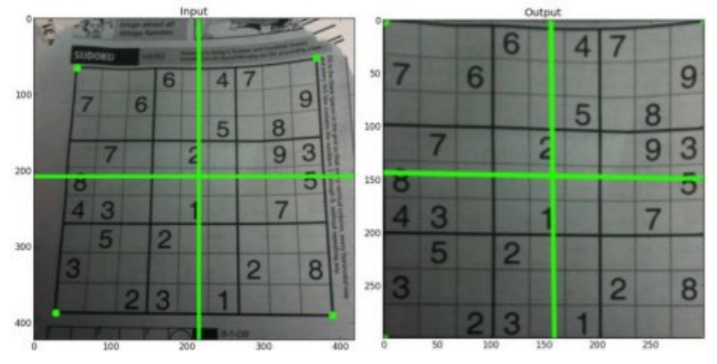
Để chuyển đổi phối cảnh, bạn cần một ma trận biến đổi 3x3. Các đường thẳng sẽ vẫn thẳng ngay cả sau khi chuyển đổi. Để tìm ma trận biến đổi này, bạn cần 4 điểm trên hình ảnh đầu vào và các điểm tương ứng trên hình ảnh đầu ra. Trong số 4 điểm này, 3 trong số đó không nên cộng tuyến. Sau đó, ma trận biến đổi có thể được tìm thấy bởi hàm cv2.getPers perspectiveTransform. Sau đó áp dụng cv2.warpPers perspective với ma trận biến đổi 3x3 này.

Xem mã dưới đây:

img = cv2.imread(’sudokusmall.png’)  
rows,cols,ch = img.shape  
pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])  
pts2 = np.float32([[0,0],[300,0],[0,300],[300,300]]) M = cv2.getPerspectiveTransform(pts1,pts2)  
dst = cv2.warpPerspective(img,M,(300,300))  
plt.subplot(121),plt.imshow(img),plt.title(’Input’)  
plt.subplot(122),plt.imshow(dst),plt.title(’Output’)  
plt.show()

plt.show()

Kết quả:



III, Ứng dụng

Về phần ứng dụng mảng này, nhóm chúng em đã viết một chương trình dùng để biến đổi ảnh và sắp xếp theo từng phần. Mỗi phần là một chức năng riêng, cộng với các trackbar tương ứng để chỉnh sửa lại các thông số theo ý thích của bản thân. Ngoài ra, còn có một mục ứng dụng giúp phát hiện được khuôn mặt trong ảnh và cắt khuôn mặt ra để giúp cho việc nhận dạng khuôn mặt dễ hơn