**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

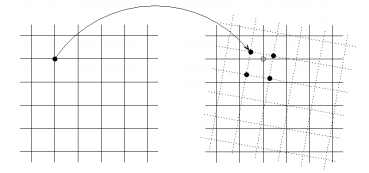
Ngoài ra sẽ có nội dung code Python áp dụng cho từng phép biến đổi khác nhau.

***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.

Trong một phép biến đổi không gian, mỗi điểm (x, y) của ảnh A được ánh xạ tới một điểm (u, v) trong một hệ tọa độ mới.

u = f1(x, y)

 v = f2(x, y)

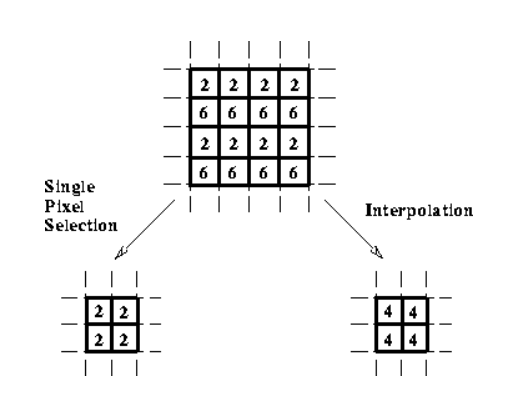
Ánh xạ từ tọa độ (x, y) đến (u, v). Một mảng ảnh số có một lưới ẩn được ánh xạ tới các điểm riêng biệt trong miền mới. Những điểm này có thể không rơi vào các điểm lưới trong miền mới.

*a, Định kích thước*

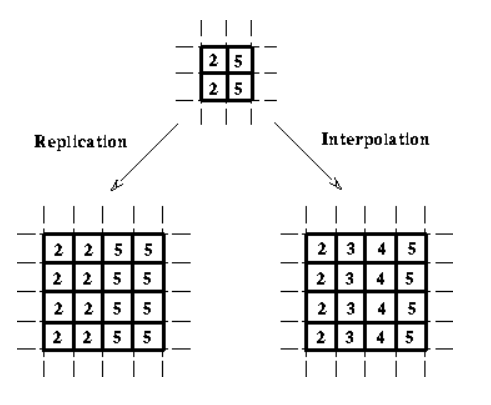
Mô tả: Toán tử tỷ lệ thực hiện một phép biến đổi hình học có thể được sử dụng để thu nhỏ hoặc thu phóng kích thước của hình ảnh (hoặc một phần của hình ảnh). Giảm hình ảnh, thường được gọi là lấy mẫu con, được thực hiện bằng cách thay thế (của một nhóm các giá trị pixel bằng một giá trị pixel được chọn tùy ý từ trong nhóm này) hoặc bằng cách nội suy giữa các giá trị pixel trong vùng lân cận cục bộ. Thu phóng hình ảnh đạt được bằng cách nhân rộng pixel hoặc bằng cách nội suy. Chia tỷ lệ được sử dụng để thay đổi giao diện trực quan của hình ảnh, để thay đổi lượng thông tin được lưu trữ trong biểu diễn cảnh hoặc dưới dạng bộ xử lý cấp thấp trong chuỗi xử lý hình ảnh nhiều giai đoạn hoạt động trên các tính năng của một tỷ lệ cụ thể. Mở rộng quy mô là một trường hợp đặc biệt của chuyển đổi affine.

-Cách hoạt động: Chia tỷ lệ nén hoặc mở rộng hình ảnh dọc theo hướng tọa độ. Vì các kỹ thuật khác nhau có thể được sử dụng để lấy mẫu và thu phóng, lần lượt từng thảo luận.

Hình 1 minh họa hai phương pháp lấy mẫu phụ. Trong lần đầu tiên, một giá trị pixel trong vùng lân cận cục bộ được chọn (có thể ngẫu nhiên) là đại diện cho môi trường xung quanh. (Phương pháp này đơn giản về mặt tính toán, nhưng có thể dẫn đến kết quả kém nếu vùng lân cận lấy mẫu quá lớn.) Phương pháp thứ hai nội suy giữa các giá trị pixel trong vùng lân cận bằng cách lấy mẫu thống kê (như giá trị trung bình) của các giá trị cường độ cục bộ.



Một hình ảnh (hoặc các vùng của một hình ảnh) có thể được phóng to thông qua sao chép pixel hoặc nội suy. Hình 2 cho thấy cách nhân rộng pixel chỉ đơn giản thay thế từng pixel ảnh gốc bằng một nhóm pixel có cùng giá trị (trong đó kích thước nhóm được xác định bởi hệ số tỷ lệ). Ngoài ra, nội suy các giá trị của các pixel lân cận trong ảnh gốc có thể được thực hiện để thay thế từng pixel bằng một nhóm pixel mở rộng. Hầu hết các triển khai cung cấp tùy chọn tăng kích thước thực tế của hình ảnh gốc hoặc giữ lại chúng và chỉ cần phóng to một phần hình ảnh trong ranh giới hình ảnh cũ.

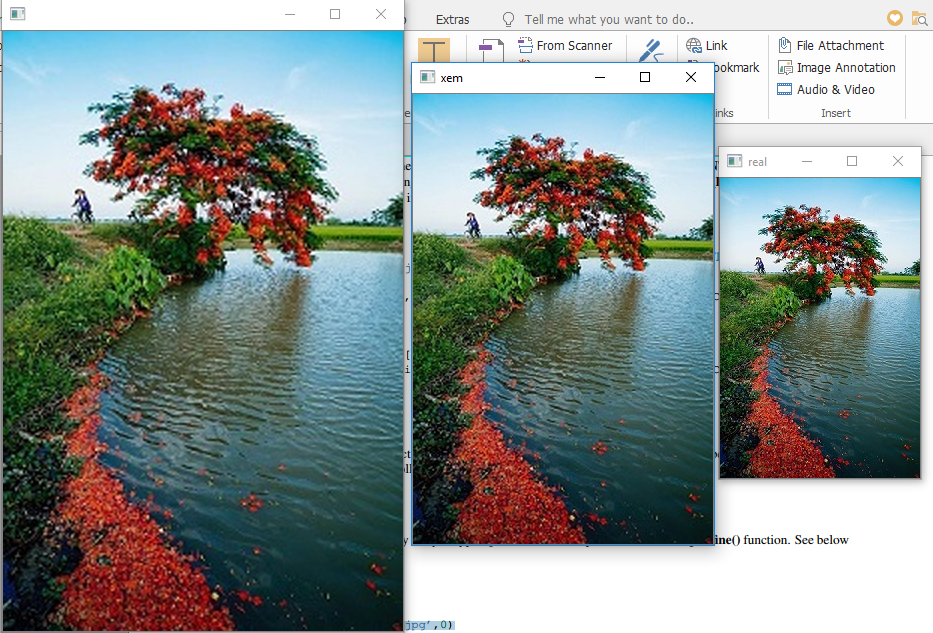


Trong đời sống ngày nay với một bức ảnh ta có thể in nó ra với nhiều kích thước khác nhau, có người thích ảnh to, có người thích ảnh nhỏ, ngoài ra trong kĩ thuật có nhiều trường hợp chúng ta cần thay đổi kích thước của tấm ảnh thu được để có các kết quả chính xác cho từng chuẩn đoán ảnh.

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*

-Mô tả: Toán tử dịch thực hiện một phép biến đổi hình học ánh xạ vị trí của từng thành phần hình ảnh trong một hình ảnh đầu vào thành một vị trí mới trong một hình ảnh đầu ra, trong đó kích thước của hai hình ảnh thường là, nhưng không nhất thiết phải giống nhau. Trong phép dịch, một phần tử hình ảnh nằm ở (x1, y1) trong bản gốc được chuyển sang vị trí mới (x2, y2) trong hình ảnh đầu ra tương ứng bằng cách dịch chuyển nó thông qua phép dịch do người dùng chỉ định (βx, βy). Việc xử lý các yếu tố gần các cạnh hình ảnh khác nhau khi thực hiện. Phép dịch được sử dụng để cải thiện trực quan hóa hình ảnh, nhưng cũng có vai trò là bộ tiền xử lý trong các ứng dụng. Phép dịch là một trường hợp đặc biệt của chuyển đổi affine.

-Cách thức hoạt động: Toán tử dịch thực hiện một phép biến đổi của biểu mẫu:

Phương trình: x2 = x1 + βx

Phương trình: y2 = y1 + βy

Vì kích thước của hình ảnh đầu vào được xác định rõ, hình ảnh đầu ra cũng là một không gian riêng biệt của kích thước hữu hạn. Nếu tọa độ mới (x2, y2) nằm ngoài hình ảnh, toán tử dịch thường sẽ bỏ qua chúng, mặc dù, trong một số triển khai, nó có thể liên kết các điểm tọa độ cao hơn với các điểm thấp hơn để bọc kết quả xung quanh vào không gian hiển thị của bức hình. Hầu hết các triển khai điền vào các khu vực hình ảnh trong đó một hình ảnh đã được thay đổi với các pixel đen.

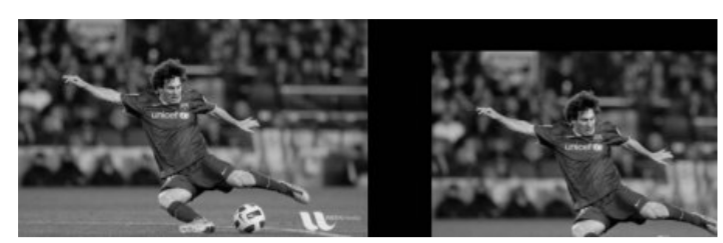
Dịch chuyển ảnh tức là ta di chuyển ảnh theo chiều dọc ngang (2 chiều) theo ý muốn. Ta cũng có thể xem dịch chuyển là thay đổi vị trí của đối tượng trong ảnh đến vị trí mới mà ta muốn đặt. Giả sử ta dịch chuyển ảnh theo hướng (x, y) theo không gian 2 chiều thì ta có thể tạo ra ma trận biến đổi ảnh M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

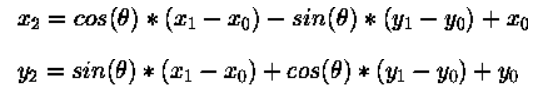
Ta có kết quả thu được như hình sau:



*c, Xoay ảnh*

Mô tả: Toán tử xoay thực hiện một phép biến đổi hình học ánh xạ vị trí (x1, y1) của một thành phần hình ảnh trong một hình ảnh đầu vào vào một vị trí (x2, y2) trong một hình ảnh đầu ra bằng cách xoay nó qua một góc θ do người dùng chỉ định. Trong hầu hết các triển khai, các vị trí đầu ra (x2, y2) nằm ngoài ranh giới của hình ảnh đều bị bỏ qua. Xoay thường được sử dụng nhất để cải thiện hình ảnh trực quan của hình ảnh, mặc dù nó có thể hữu ích như một bộ xử lý trước trong các ứng dụng có liên quan đến các toán tử định hướng. Xoay là một trường hợp đặc biệt của chuyển đổi affine.

-Cách thức hoạt động: Toán tử xoay thực hiện một phép biến đổi của biểu mẫu:



Trong đó (x0, y0) là tọa độ của tâm quay (trong ảnh đầu vào) và θ là góc quay với góc quay theo chiều kim đồng hồ có góc dương. (Lưu ý ở đây là chúng ta đang làm việc trong tọa độ hình ảnh, do đó trục y đi xuống. Công thức xoay tương tự có thể được xác định khi trục y đi lên.) Thậm chí nhiều hơn toán tử dịch, thao tác xoay tạo ra các vị trí đầu ra (x2, y2) không vừa với ranh giới của hình ảnh (như được xác định bởi kích thước của hình ảnh đầu vào ban đầu). Trong các trường hợp như vậy, các phần tử đích đã được ánh xạ bên ngoài hình ảnh bị bỏ qua bởi hầu hết các triển khai. Các vị trí pixel trong đó hình ảnh đã được xoay thường được lấp đầy bằng các pixel đen.

Thuật toán xoay, không giống như thuật toán được sử dụng bởi dịch thuật, có thể tạo ra tọa độ (x2, y2) không phải là số nguyên. Để tạo cường độ của các pixel tại mỗi vị trí số nguyên, các phương pháp phỏng đoán khác nhau (hoặc kỹ thuật lấy mẫu lại} có thể được sử dụng. Ví dụ: hai phương pháp phổ biến bao gồm:

Cho phép mức cường độ tại mỗi vị trí pixel nguyên để giả sử giá trị của hàng xóm không nguyên gần nhất (x2, y2)

Tính mức cường độ tại mỗi vị trí pixel nguyên dựa trên mức trung bình có trọng số của n giá trị không nguyên gần nhất. Trọng số tỷ lệ thuận với khoảng cách hoặc pixel chồng lấp của các hình chiếu gần đó.

Phương pháp sau tạo ra kết quả tốt hơn nhưng tăng thời gian tính toán của thuật toán.

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:

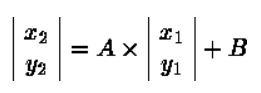


*d, Biến đổi Affine*

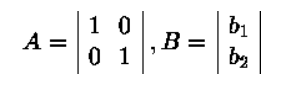
-Mô tả: Trong nhiều hệ thống hình ảnh, hình ảnh được phát hiện có thể bị biến dạng hình học do các bất thường phối cảnh giới thiệu trong đó vị trí của máy ảnh đối với cảnh làm thay đổi kích thước rõ ràng của hình dạng cảnh. Áp dụng một phép biến đổi affine cho một hình ảnh bị biến dạng đồng đều có thể sửa cho một loạt các biến dạng phối cảnh bằng cách chuyển đổi các phép đo từ tọa độ lý tưởng sang các tọa độ thực sự được sử dụng. (Ví dụ: điều này rất hữu ích trong hình ảnh vệ tinh trong đó cần có bản đồ mặt đất chính xác về mặt hình học.)

Phép biến đổi affine là một lớp quan trọng của phép biến đổi hình học 2 chiều tuyến tính, ánh xạ các biến (ví dụ: các giá trị cường độ pixel nằm ở vị trí Eqn: (x1, y1) trong ảnh đầu vào) thành các biến mới (ví dụ (x2, y2) trong ảnh đầu ra) kết hợp các thao tác dịch, xoay, chia tỷ lệ và / hoặc cắt (tức là chia tỷ lệ không đồng nhất theo một số hướng).

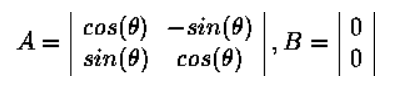
 -Cách hoạt động: Phép biến đổi affine chung thường được viết theo tọa độ đồng nhất như dưới đây:



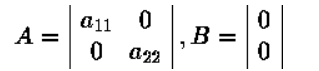
Bằng cách chỉ xác định ma trận B, phép chuyển đổi này có thể thực hiện dịch thuần túy:



Xoay thuần sử dụng ma trận A và được định nghĩa là (đối với các góc dương là xoay theo chiều kim đồng hồ):



Tương tự, tỷ lệ thuần túy là:



(Lưu ý rằng một số phép biến đổi affine khác nhau thường được kết hợp để tạo ra phép biến đổi kết quả. Thứ tự các phép biến đổi xảy ra rất có ý nghĩa vì một bản dịch theo sau một phép quay không nhất thiết phải tương đương với phép đảo ngược.)

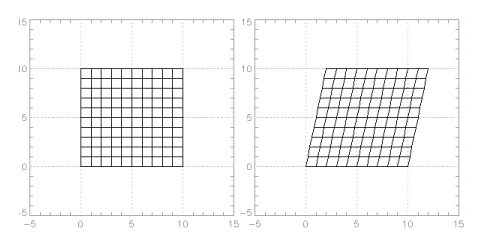
Do phép biến đổi affine chung được xác định bởi 6 hằng số, nên có thể xác định phép biến đổi này bằng cách chỉ định vị trí hình ảnh đầu ra mới Eqn: (x2, y2) của bất kỳ cặp tọa độ hình ảnh đầu vào nào trong ba cặp (x1, y1) . (Trong thực tế, nhiều điểm hơn được đo và phương pháp bình phương tối thiểu được sử dụng để tìm biến đổi phù hợp nhất.)

Một phép biến đổi affine là phép biến đổi bảo toàn cộng tuyến (collinearity) (tức là, tất cả các điểm nằm trên một dòng ban đầu vẫn nằm trên một dòng sau khi chuyển đổi) và tỷ lệ khoảng cách (ví dụ: điểm giữa của đoạn thẳng vẫn là trung điểm sau khi biến đổi).Nói chung, một phép biến đổi affine là một thành phần của phép quay, bản dịch, phóng đại và kéo.

u = c11x+ c12y + c13

v = c21x + c22y + c23

c13 và c23 ảnh hưởng đến bản dịch, c11 và c22 ảnh hưởng đến độ phóng đại và sự kết hợp ảnh hưởng đến việc quay và kéo.

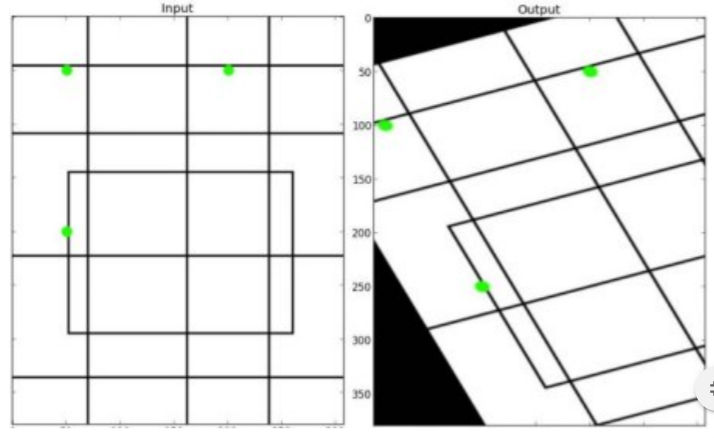


Trong phép biến đổi affine, tất cả các đường song song trong ảnh gốc sẽ vẫn song song trong ảnh đầu ra. Để tìm ma trận biến đổi, chúng ta cần ba điểm từ hình ảnh đầu vào và vị trí tương ứng của chúng trong hình ảnh đầu ra. Sau đó, cv2.getAffineTransform sẽ tạo ra một ma trận 2x3 sẽ được chuyển đến cv2.warpAffine.

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**

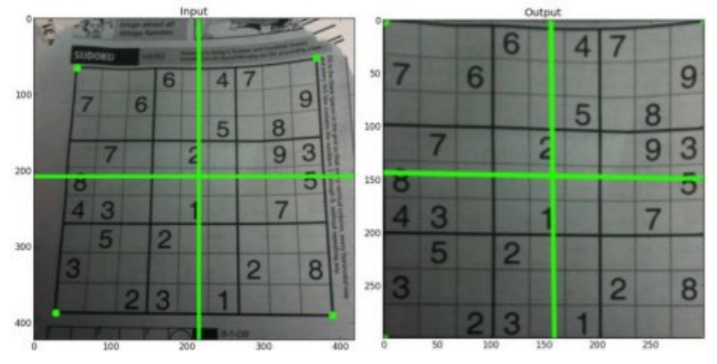
Để chuyển đổi phối cảnh, bạn cần một ma trận biến đổi 3x3. Các đường thẳng sẽ vẫn thẳng ngay cả sau khi chuyển đổi. Để tìm ma trận biến đổi này, bạn cần 4 điểm trên hình ảnh đầu vào và các điểm tương ứng trên hình ảnh đầu ra. Trong số 4 điểm này, 3 trong số đó không nên cộng tuyến. Sau đó, ma trận biến 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:

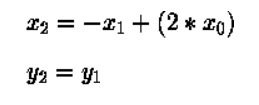


e. Phép phản chiếu:

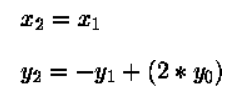
-Mô tả: Toán tử phản chiếu biến đổi hình học một hình ảnh sao cho các thành phần hình ảnh, tức là các giá trị pixel, nằm ở vị trí (x1, y1) trong ảnh gốc được phản ánh về trục hình ảnh do người dùng chỉ định hoặc điểm hình ảnh thành một vị trí mới (x2, y2) trong một hình ảnh đầu ra tương ứng . Sự phản chiếu chủ yếu được sử dụng như một sự trợ giúp cho hình ảnh trực quan, nhưng có thể được sử dụng như một toán tử tiền xử lý theo cách tương tự như xoay. Phản xạ là một trường hợp đặc biệt của biến đổi affine.

-Cách hoạt động: Sự phản chiếu có thể được thực hiện về một trục hình ảnh hoặc một điểm trong hình ảnh. Trong trường hợp trước đây, một số phép biến đổi thường được sử dụng là như sau:

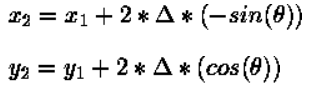
Phản ánh về trục dọc của x0 trong hình ảnh đầu vào:



Phản ánh về một trục ngang của tọa độ y0­:



Phản ánh về một trục được định hướng theo bất kỳ hướng tùy ý θ và đi qua (x0, y0)

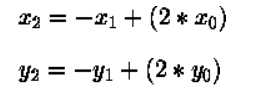


trong đó:



Lưu ý rằng nếu (x0, y0) không ở giữa hình ảnh đầu vào, một phần của hình ảnh sẽ được phản ánh ra khỏi phạm vi có thể nhìn thấy của hình ảnh. Hầu hết các triển khai điền vào các khu vực hình ảnh trong đó các pixel đã được phản chiếu với các pixel đen.

Từ cuộc thảo luận này, dễ dàng thấy rằng sự phản chiếu ngang và dọc về một điểm (x0, y0) trong hình ảnh đầu vào được đưa ra bởi:



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