

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



Computer Vision
Lớp L01

Assignment 4: Transformations

Giảng viên hướng dẫn: Võ Thanh Hùng

Sinh viên: Đinh Vũ Hà - 2113269

Thành phố Hồ Chí Minh, Tháng 04/2024



Mục lục

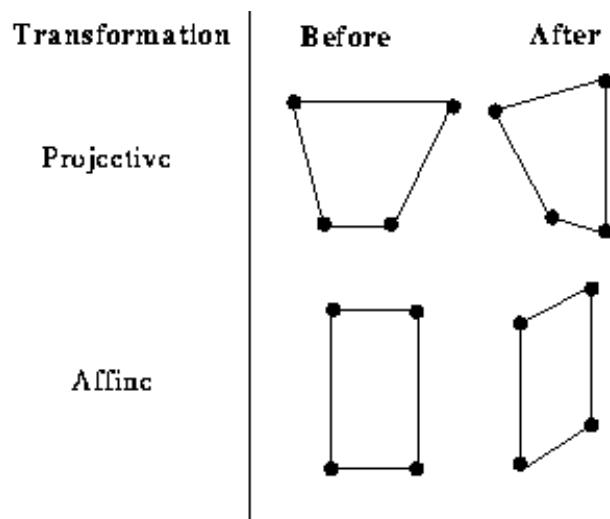
| | |
|---|----|
| 1 Yêu cầu | 2 |
| 2 Lý thuyết | 2 |
| 3 So sánh đánh giá affine và projective transformation | 4 |
| 4 Code | 7 |
| 5 Thực hiện các phép biến đổi | 7 |
| 6 Sử dụng projective transformation dán ảnh lên mặt phẳng cho trước | 14 |

1 Yêu cầu

- _ Viết các đoạn chương trình demo các phép biến đổi khác nhau đã học.
- _ So sánh đánh giá affine và projective transformation (ví dụ trực quan).
- _ Thủ nghiệm dán 1 hình chữ nhật/vuông lên một mặt phẳng chỉ định trước.

2 Lý thuyết

Trong bài tập lớn này, chúng ta làm quen với các phép biến đổi ảnh, có thể chia thành 2 nhóm chính đó là biến đổi affine (affine transformations) và biến đổi chiếu (perspective transformations).

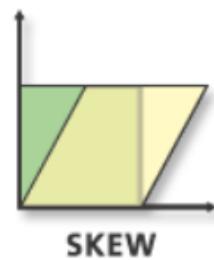


Affine, projective transformations

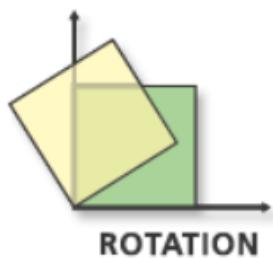
Affine transformation là một loại biến đổi hình học tuyến tính. Nó bao gồm các phép biến đổi như dịch chuyển (translation), co giãn (scaling), quay (rotation), cắt xéo (shearing) và phản chiếu (reflection). Các phép biến đổi này bảo toàn các đường thẳng và các đường song song. Các tỉ lệ trên hình được giữ nguyên.



DIFFERENTIAL SCALING



SKEW



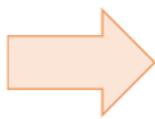
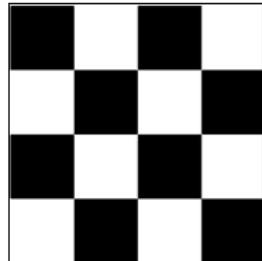
ROTATION



TRANSLATION

Một số affine transformations

Trong khi đó, projective transformation là một loại biến đổi hình học phi tuyến tính. Nó không bảo toàn các đường thẳng và song song, bao gồm các phép biến đổi như co giãn không đồng nhất. Projective transformation có thể biểu diễn một phần của không gian ba chiều trong không gian hai chiều, có thể biến đổi hình dạng của các đối tượng, cho phép biến đổi một hình ảnh từ một hệ tọa độ sang một hệ tọa độ khác một cách không tuyến tính.



Projective transformations



3 So sánh đánh giá affine và projective transformation

Dựa vào lý thuyết trên, ta có thể dễ dàng đưa 2 kiểu transformations lên bàn cân so sánh:

| Affine | Projective |
|--|---|
| Là biến đổi hình học tuyến tính | Là phép biến đổi hình học không nhất thiết phải tuyến tính |
| Giữ nguyên tính chất song song giữa các đường | Không nhất thiết giữ tính chất song song giữa các đường |
| Giữ nguyên tỉ lệ giữa các chi tiết với nhau | Không cần giữ nguyên tỉ lệ |
| * Phù hợp với các bài toán biến đổi góc nhìn 2 chiều | * Phù hợp xử lý các bài toán về thay đổi góc nhìn 3 chiều ("uốn" các vật thể từ hình dạng này về hình dạng mong muốn) |

Ở hàng cuối của cột so sánh, ta có ví dụ về sự khác nhau trong ứng dụng của 2 kiểu biến đổi này.

Ta có ảnh của 1 tòa nhà như sau:



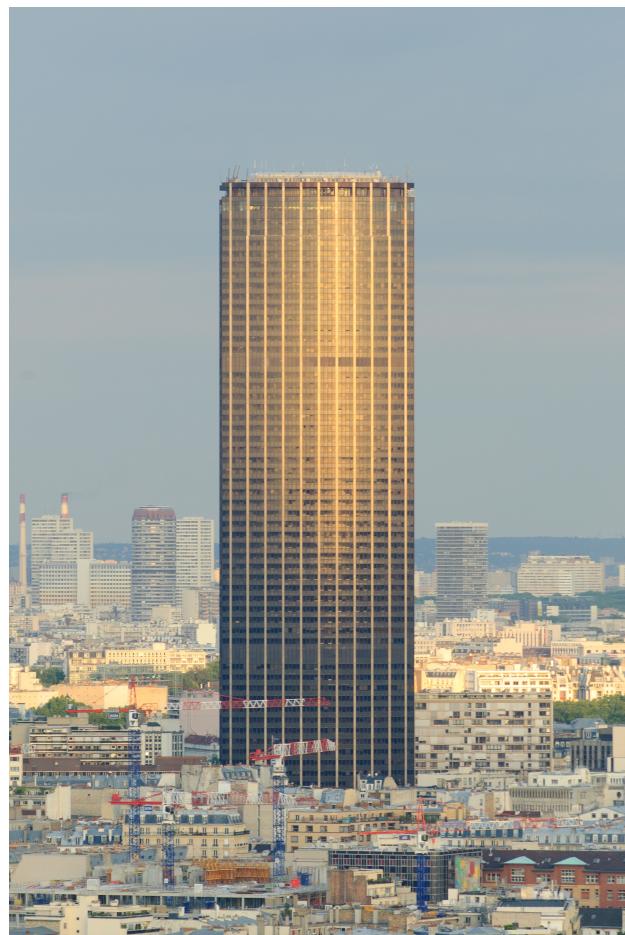
Ảnh 1 tòa nhà



Ở góc độ này, ta có thể thấy toàn cảnh tòa nhà được bao phủ bởi 1 hình như sau:



Có thể thấy, hình trên là một hình thang. Nói cách khác, theo góc độ này, chúng ta sẽ nhận định rằng tòa nhà là 1 vật thể hình thang theo góc nhìn 2D. Với Affine transformation, ta sẽ giữ nguyên nhận định này, bởi vì nó giữ nguyên tỉ lệ các chi tiết. Tuy nhiên với Projective transformation, ta có thể đưa góc nhìn về góc như thế này (ảnh minh họa):



Ảnh trên cũng là tòa nhà đó, tuy nhiên thay vì chúng ta nhìn tòa nhà như hình thang, ta có thể biến đổi nó để đưa về dạng hình chữ nhật như góc nhìn trong ảnh trên (hình trên hình chỉ mang tính chất minh họa về góc nhìn, không phải kết quả chính xác) bằng Projective transformation, đúng với cấu trúc của tòa nhà trên thực tế. Điều này Affine Transformation không thể làm, vì không thể biến đổi 1 hình thang thành 1 hình chữ nhật được (đã vi phạm nguyên tắc song song của các đường). Qua ví dụ này, ta thấy được điểm khác biệt giữa 2 phép biến đổi này.



4 Code

Link tới thư mục drive của Assignment: https://drive.google.com/drive/folders/1dIIcSdjAVP73jxHxfUhYusp=drive_link

Trong đó: Thư mục Transformations chứa các file liên quan tới mục 5, thư mục HomographyEditImages liên quan tới mục 6 của bài báo cáo

5 Thực hiện các phép biến đổi

Các bước thực hiện phép biến đổi trong phần này nằm ở file Transformations.py. Ta có file ảnh gốc (husky.jpg) sẽ được biến đổi dựa trên các phép biến đổi.



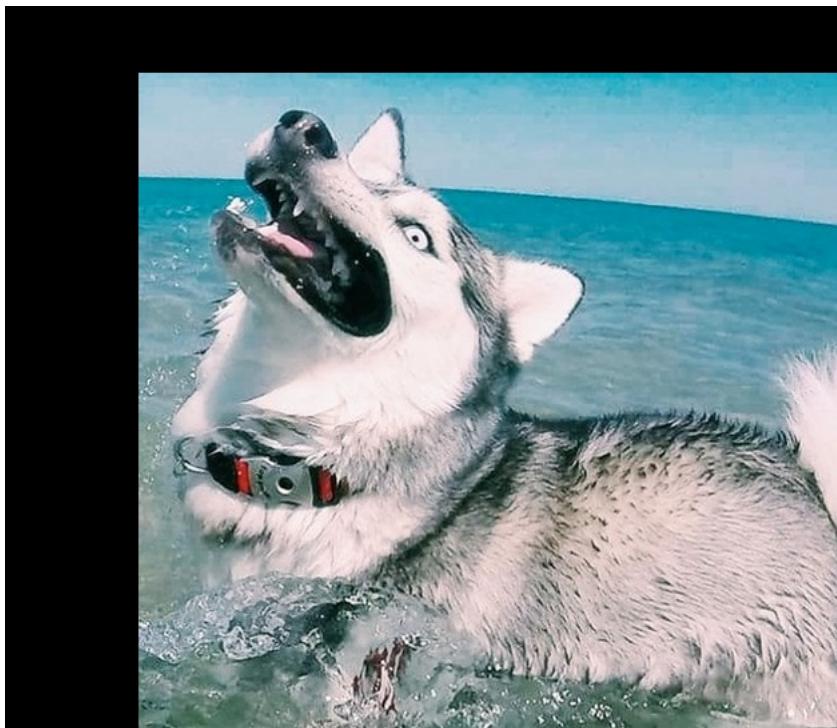
Ảnh gốc

- Phép translation: Thực hiện xê dịch ảnh qua phải theo trục x 100 đơn vị (ở đây là pixel) và dịch xuống dưới theo trục y 50 đơn vị (pixel). Ta có code và kết quả:



```
6 # translation
7 T= np.float32([[1, 0, 100], [0, 1, 50]])
8 translation = cv.warpAffine(img, T, (cols, rows))
9 cv.imshow('translation', translation)
10 cv.imwrite('husky_translation.jpg', translation)
11 cv.waitKey(0)
12 cv.destroyAllWindows()
```

Kết quả:



Kết quả phép translation

- Phép scaling: Thực hiện tăng kích thước theo chiều x là 1.5 và chiều y là 0.5. Ta có code và kết quả:



```
15 # scaling
16 scaling = cv.resize(img, None, fx=1.5, fy=0.5, interpolation=cv.INTER_CUBIC)
17 cv.imshow('scaling', scaling)
18 cv.imwrite('husky_scaling.jpg', scaling)
19 cv.waitKey(0)
20 cv.destroyAllWindows()
```

Kết quả:



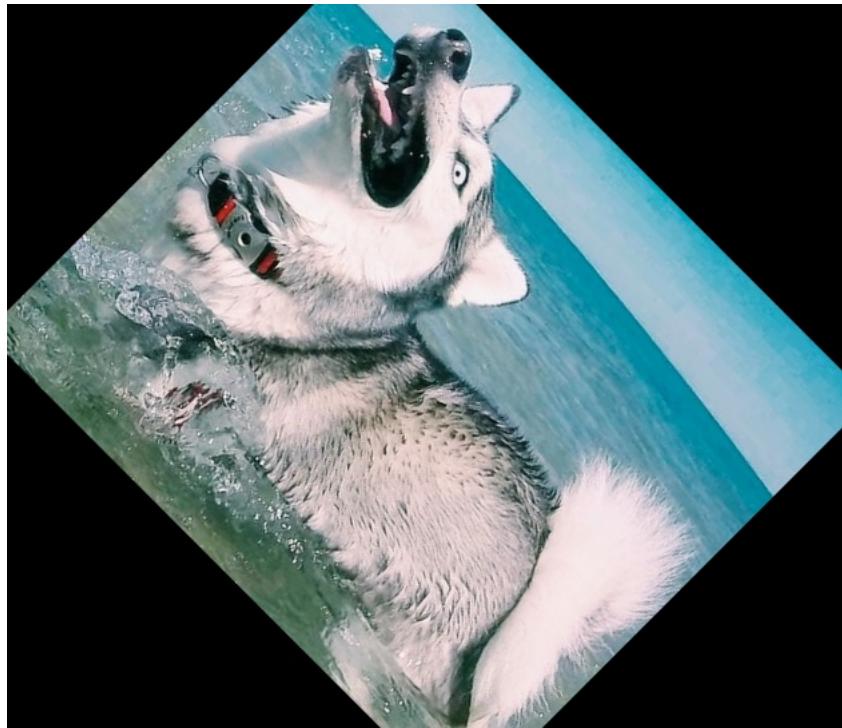
Kết quả phép scaling

- Phép rotation: Thực hiện xoay ảnh theo chiều kim đồng hồ 1 góc 45 độ. Ta có code và kết quả:

```
22 # rotation
23 rotation = cv.warpAffine(img, cv.getRotationMatrix2D((cols/2, rows/2), -45, 0.8), (cols, rows))
24 cv.imshow('rotation', rotation)
25 cv.imwrite('husky_rotation.jpg', rotation)
26 cv.waitKey(0)
27 cv.destroyAllWindows()
```



Kết quả:



Kết quả phép rotation

- Phép phản chiếu (gương): Thực hiện làm ảnh phản chiếu như khi ta soi gương. Ta có code và kết quả:

```
29  # mirror
30  M = np.float32([[-1, 0, cols], [0, 1, 0], [0, 0, 1]])
31  mirror = cv.warpPerspective(img, M, (int(cols), int(rows)))
32  cv.imshow('mirror', mirror)
33  cv.imwrite('husky_mirror.jpg', mirror)
34  cv.waitKey(0)
35  cv.destroyAllWindows()
```



Kết quả:



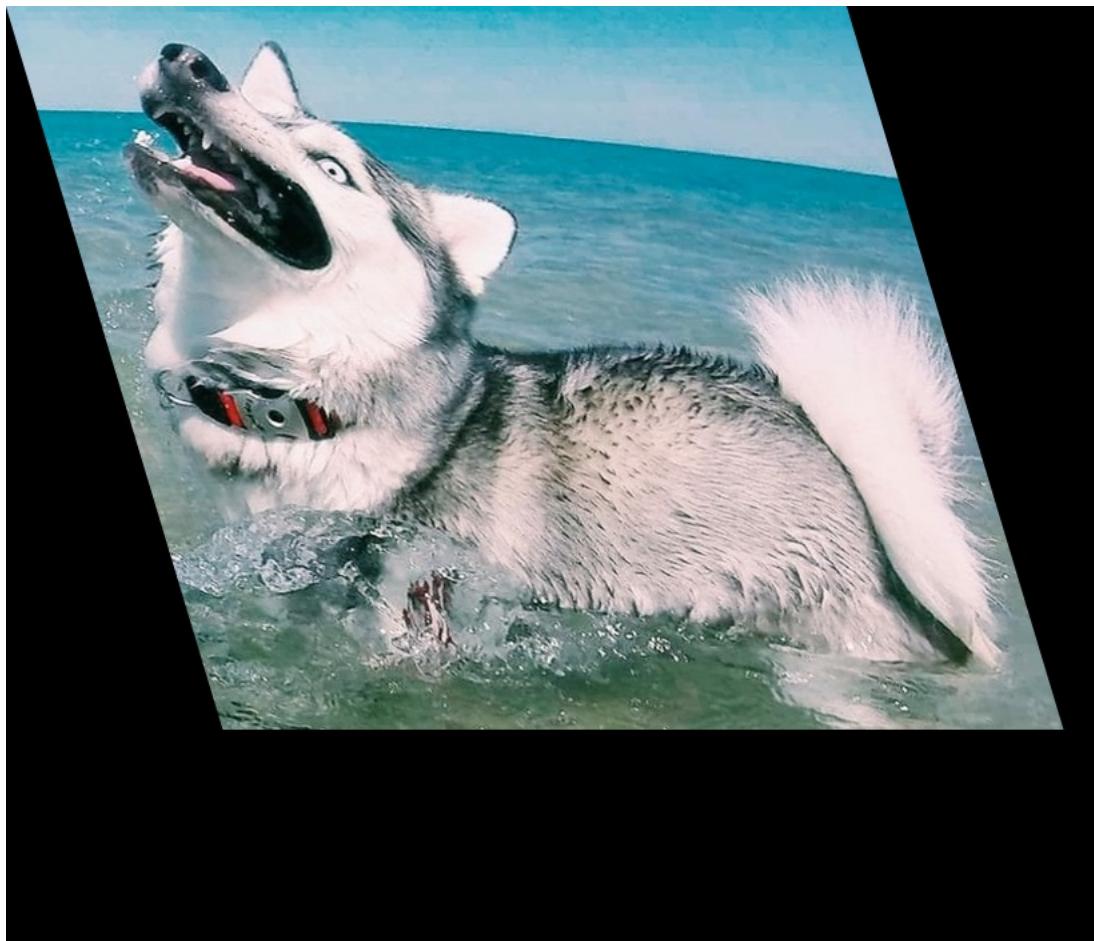
Kết quả phép mirror

- Phép cắt xéo ảnh (theo chiều x): Thực hiện làm ảnh xéo theo chiều x. Ta có code và kết quả:

```
37 # shear in x-axis
38 Sx = np.float32([[1, 0.3, 0], [0, 1, 0], [0, 0, 1]])
39 shearx = cv.warpPerspective(img, Sx, (int(cols*1.3), int(rows*1.3)))
40 cv.imshow('shearx', shearx)
41 cv.imwrite('husky_shearx.jpg', shearx)
42 cv.waitKey(0)
43 cv.destroyAllWindows()
```



Kết quả:



Kết quả phép shearx

- Phép cắt xéo ảnh (theo chiều y): Thực hiện làm ảnh xéo theo chiều y. Ta có code và kết quả:

```
45 # shear in y-axis
46 Sy = np.float32([[1, 0, 0], [0.3, 1, 0], [0, 0, 1]])
47 sheary = cv.warpPerspective(img, Sy, (int(cols*1.3), int(rows*1.3)))
48 cv.imshow('sheary', sheary)
49 cv.imwrite('husky_sheary.jpg', sheary)
50 cv.waitKey(0)
51 cv.destroyAllWindows()
```



Kết quả:



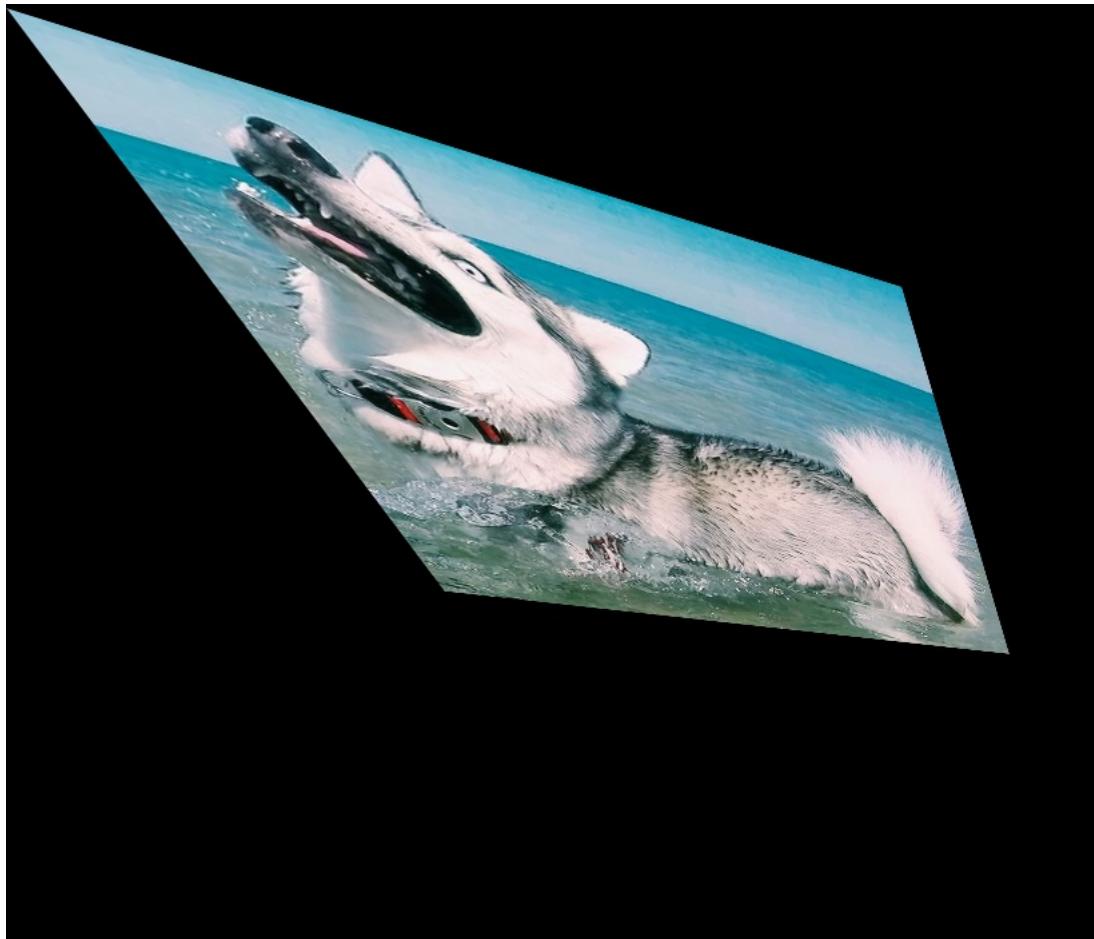
Kết quả phép sheary

- Projective transformation: Thực hiện làm ảnh biến đổi. Ta có code và kết quả:

```
53 # projective transformation
54
55 P = np.float32([[ 1.6, 0.9, 1.3], [ 0.5, 1.2, 3.6], [0.0008, 0.0009, 1]])
56 projective = cv.warpPerspective(img, P, (int(cols*1.3), int(rows*1.3)))
57 cv.imshow('projective', projective)
58 cv.imwrite('husky_projective.jpg', projective)
59 cv.waitKey(0)
60 cv.destroyAllWindows()
```



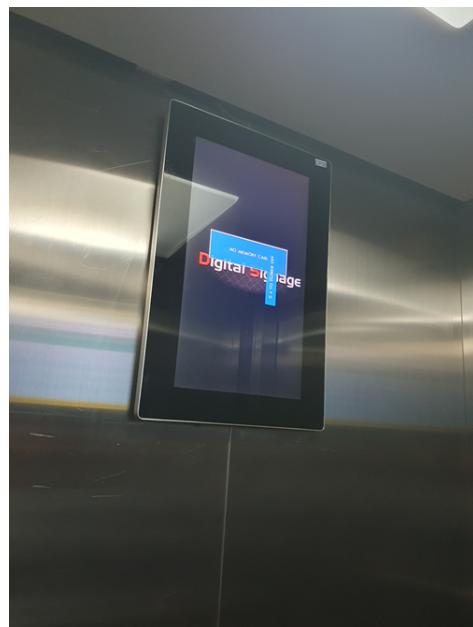
Kết quả:



Kết quả phép projective transformation

6 Sử dụng projective transformation dán ảnh lên mặt phẳng cho trước

Trong phần này, ta sẽ tiến hành dán ảnh hình chữ nhật/ hình vuông lên mặt phẳng chỉ định trước. Trước tiên ta có các ảnh gốc ban đầu và ảnh hình chữ nhật mà ta sẽ dán:



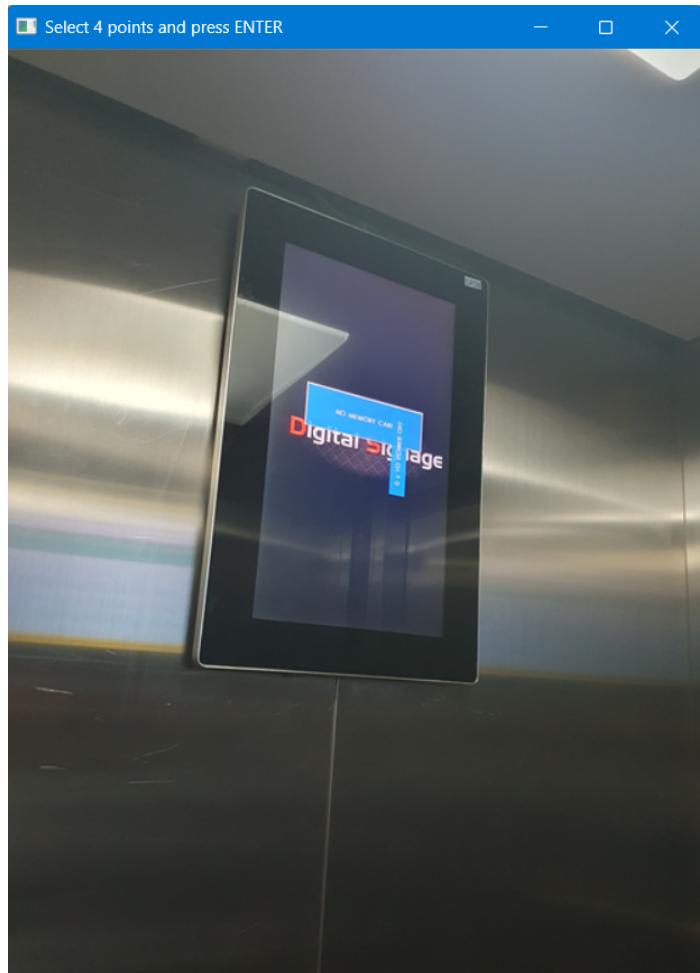
Ảnh ban đầu



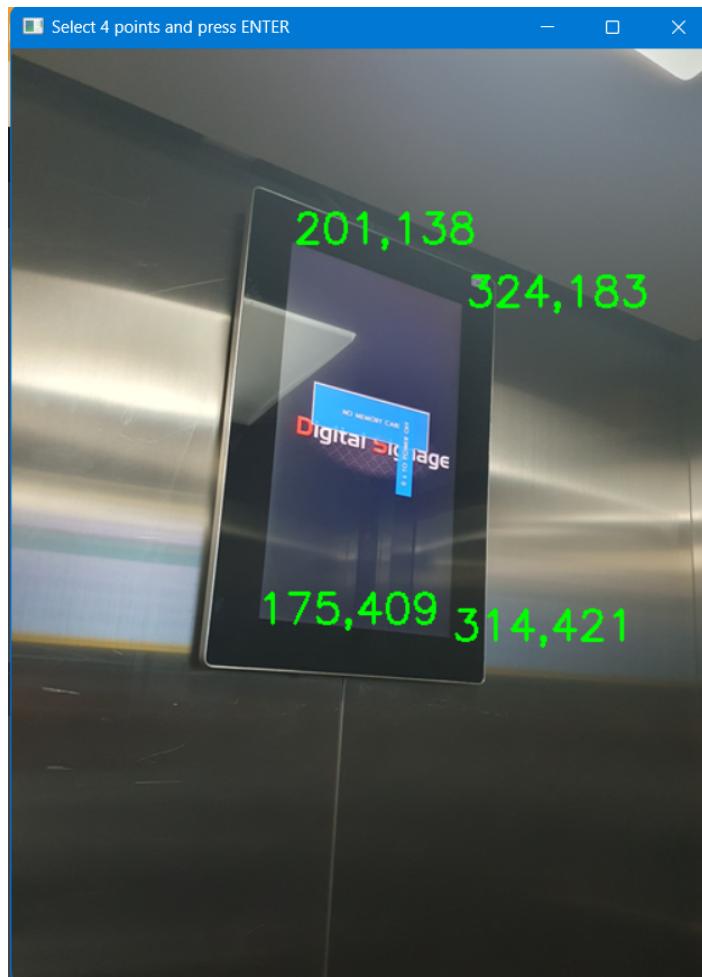
Ảnh sẽ được dán vào



Trước tiên, chúng ta chạy file findobject.py của trong thư mục homography trước. Khi chạy file trên, màn hình sẽ hiện ra như thế này để ta thực hiện chọn 4 điểm cắt:

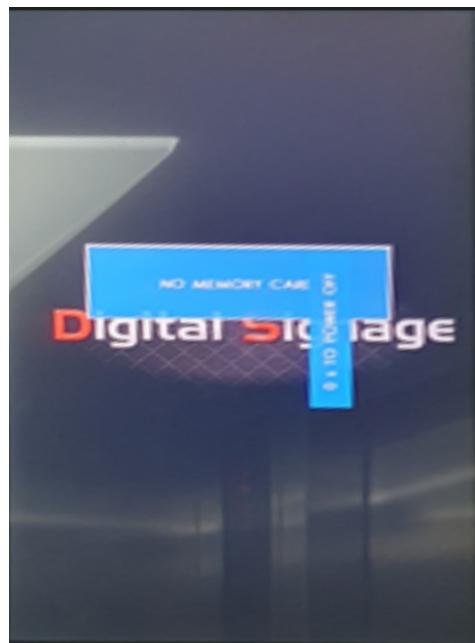


Tiếp theo, ta click chuột vào ảnh, chọn 4 vị trí cần cắt để tiến hành cắt vật thể ra khỏi ảnh (tương ứng với 4 tọa độ được viết bằng màu xanh trong hình dưới):



Tọa độ 4 điểm cần cắt

Sau khi chọn xong, ta sẽ thu về được ảnh đã được cắt ra từ 4 điểm được chọn, đây chính là lời giải cho bài toán đã nêu trên mục 3 nói ở trên. Ảnh được lưu lại dưới tên file res-1.jpg như sau:



Hình ảnh file res-1.jpg

Qua bước trên, ta đã lấy được vật thể cần cắt và đưa vật thể từ hình tứ giác (không vuông) về dạng hình chữ nhật. Tiếp theo, muốn dán ảnh chữ nhật cần được dán vào ở trên, ta chạy file run.py. Và ta có kết quả:

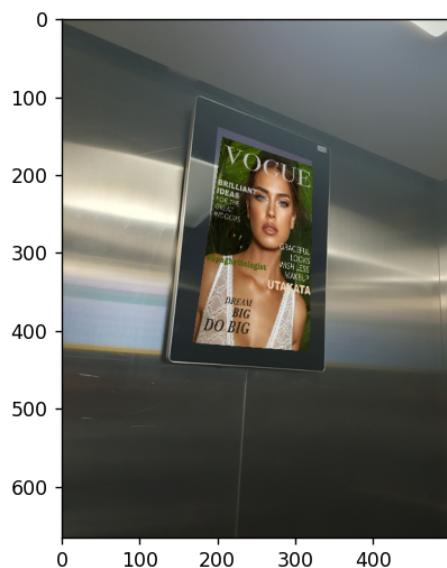


Ảnh cần được dán vào



Figure 1

– □ ×



Kết quả sau khi thực hiện dán hình