ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
VNUHCM - UIT

Đồ án cuối kì

Môn: Nhập môn Thị giác máy tính (CS231)

Neon Effect with Computer Vision

Giảng viên hướng dẫn: TS. Nguyễn Vinh Tiệp

Sinh viên thực hiện đồ án: Nguyễn Minh Lý 20521592

TP.HCM, Ngày 6 tháng 1 năm 2023

Mục lục

Phần 1: Giới thiệu bài toán

1. Hiệu ứng neon là gì?

Neon là xu hướng màu lấy cảm hứng từ ánh đèn led. Màu neon là những gam màu chói, cực bắt mắt và nổi bật như: vàng sáng, xanh nõn chuối,...Đi kèm là những tông màu tối để giúp những gam màu này trở nên nổi bật và tạo ra cảm giác chói sáng.

Hiệu ứng này thường sử dụng trong các bảng hiệu, poster quảng cáo để gây nổi bật và chú ý tới người xem.

Hình 1: Ảnh minh họa hiệu ứng neon

2. Lý do thực hiện đề tài

Hiệu ứng neon xuất hiện ngày càng nhiều xung quanh ta, đặc biệt là các video top trending trên mạng xã hội. Trước đây, em chỉ tiếp cận nó với tư cách là người dùng vì chưa đủ kiến thức để hiểu sâu về nó. Sau khi được trang bị các kiến thức nền tảng nhất về xử lý ảnh, em muốn vận dụng những kiến thức đó để tìm hiểu và tự tạo ra hiệu ứng neon cho mình.

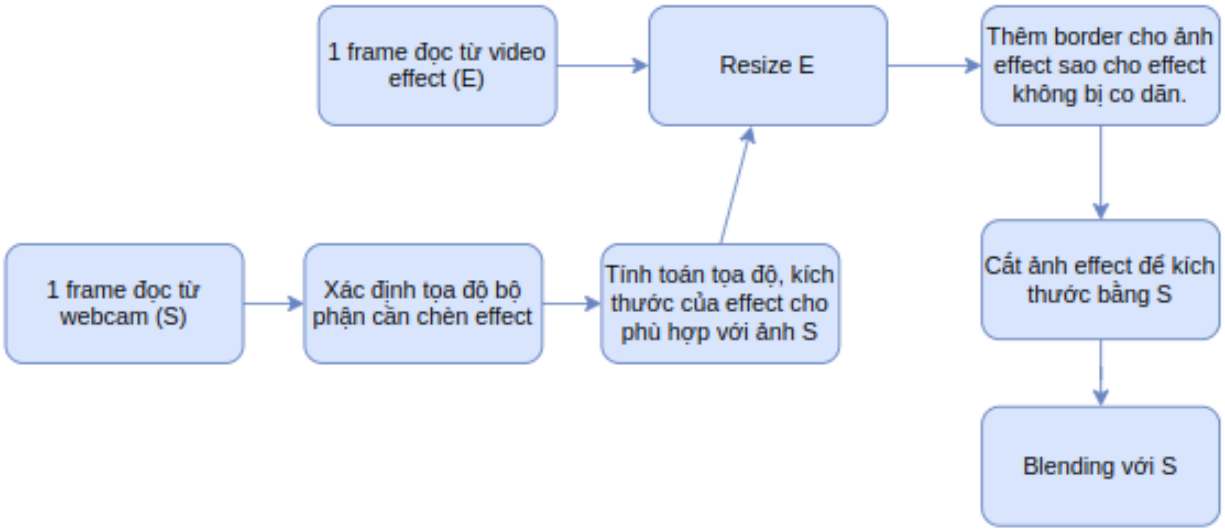
3. Thách thức của đề tài

Chủ đề Neon Effect không phải là mới xuất hiện gần đây, mà đã xuất hiện khá lâu trên mạng xã hội. Hiện có rất nhiều app hỗ trợ thêm hiệu ứng neon effect miễn phí và chất lượng. Tuy phổ biến là vậy, nhưng chắc do vấn đề bảo mật và bản quyền nên em không tìm thấy bài viết nào viết về các ý tưởng thực hiện và source code cho chủ đề này. Nên cách thực hiện trong đồ án là do vận dụng các kiến thức đã học để thực hiện.

Bên cạnh vấn đề hướng tiếp cận và phương pháp, vấn đề khó nhằn hơn là tìm và suy nghĩ ra hiệu ứng đẹp, bắt mắt. Tại quy cho cùng, dưới khía cạnh user họ cần một ứng dụng tốt, đẹp, hiệu quả chứ không quan tâm đến cách mà chúng ta làm.

Phần 2: Hướng tiếp cận

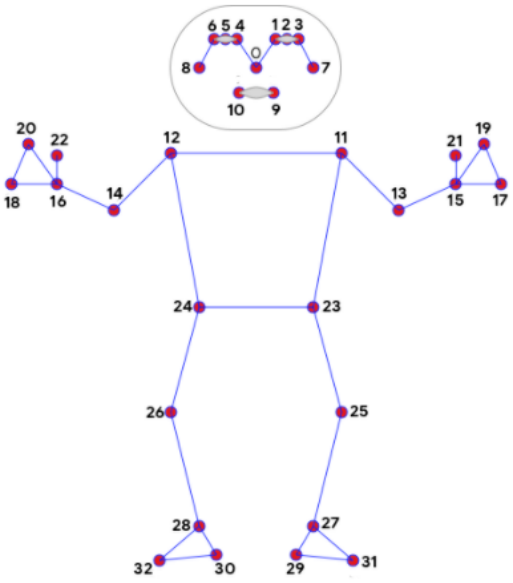
1. Phương pháp chung giải quyết bài toán



Hình 2: Pipeline bài toán

a. Pose detect

Để xác định tọa độ các bộ phận trên cơ thể, em sử dụng Mediapipe, một thư viện cung cấp bởi Google. Với mỗi bức ảnh input đưa vào thuật toán, model sẽ trả về 33 điểm tọa độ (hình dưới) cho mỗi người trong ảnh.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Hình 3: 33 bộ phận trên cơ thể mà Mediapipe có thể phát hiện.

b. Resize effect

Để resize ảnh E, em sử dụng hàm resize có sẵn trong opencv. Opencv hỗ trợ nhiều kĩ thuật “Interpolation” cho việc resize, mỗi kĩ thuật sử dụng cho các trường hợp khác

nhau. Trong bài toán này, các hiệu ứng đa phần cần resize nhỏ hơn, nên em sử dụng cv2.INTER_AREA theo đề nghị của opencv để giảm việc ảnh bị vỡ.

Bảng 1: Kết quả thực nghiệm resize ảnh với các interpolation khác nhau, thu nhỏ 5 lần so với ảnh gốc.

Interpolation Method	Result
cv2.INTER_AREA (Time process: 2.3ms)	
cv2.INTER_CUBIC (Time process: 0.79ms)	
cv2.INTER_LANCZOS4 (Time process: 3.05ms)	
cv2.INTER_LINEAR (Time process: 0.28ms)	

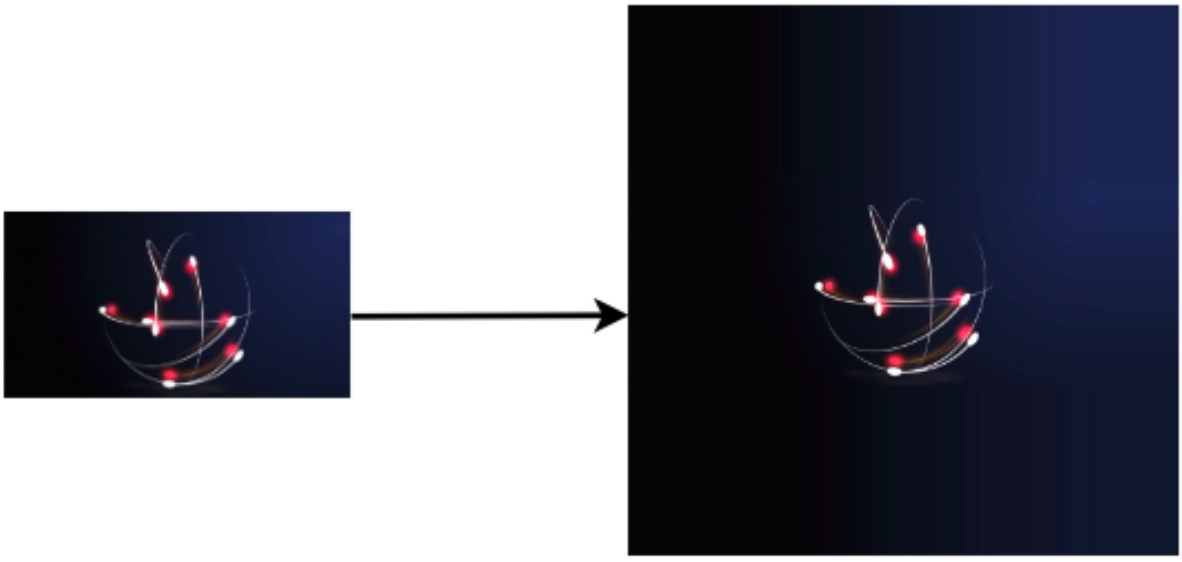
cv2.INTER_NEAREST (Time process: 0.14ms)	

c. Thêm border và cắt hợp lý cho ảnh effect

Sau khi resize, ảnh E có kích thước khác với ảnh S, do đó em sử dụng 2 kĩ thuật sau để có ảnh E cùng shape với ảnh S:

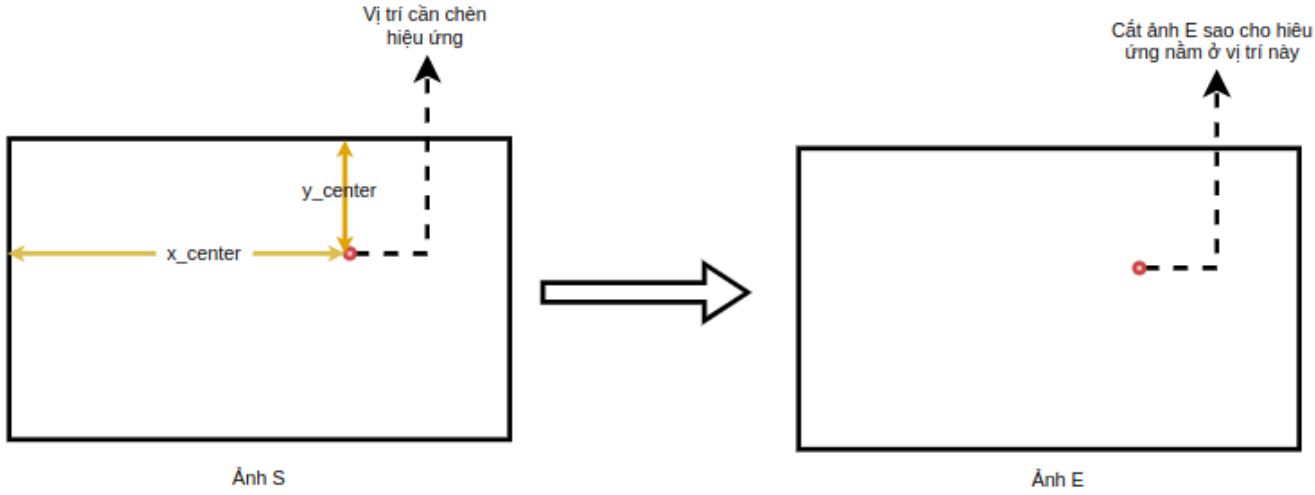
- Thêm border cho ảnh E
  Mục tiêu: ảnh đầu ra có kích thước 2000x2000, tâm hiệu ứng ở vị trí 1000x1000. Để làm được điều này, em sử dụng hàm copyMakeBorder của opencv với borderType là cv2.BORDER_REPLICATE và truyền các tham số:

Tham số	Chức năng
top=`1000 - effect.shape[0]//2`	Mở rộng về phía trên top pixcel
bot=`1000 - effect.shape[0]//2`	Mở rộng về phía dưới bot pixcel
left=`1000 - effect.shape[1]//2`	Mở rộng về phía trái left pixcel
right=`1000 - effect.shape[1]//2`	Mở rộng về phía phải right pixcel



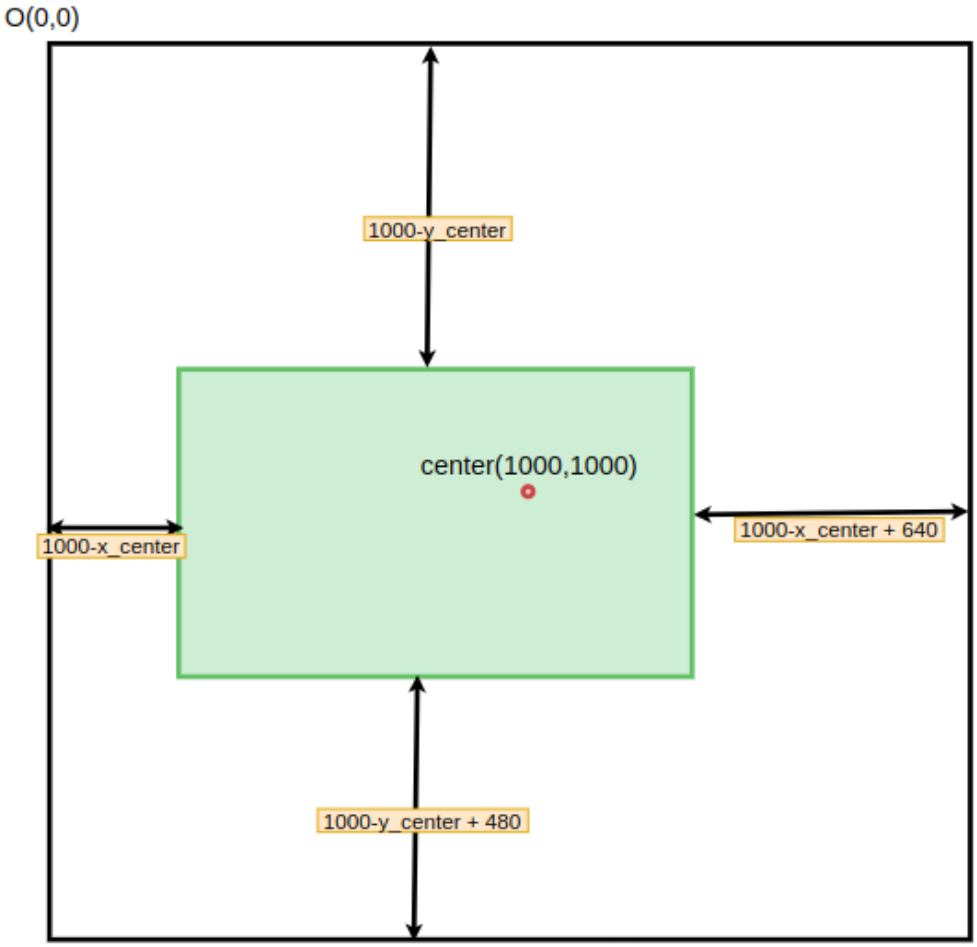
Hình 4: Minh họa việc thêm border cho ảnh

- Cắt ảnh để hiệu ứng ở đúng vị trí.
  Mục tiêu: cắt ảnh E để kích thước bằng với kích thước của ảnh S và hiệu ứng chính ở đúng vị trí cần chèn so với ảnh gốc.



Hình 5: Minh họa vị trí cắt hiệu ứng

Với ảnh S có shape wxh: 640x480, ta xác định tọa độ cần cắt như ảnh bên dưới:



Hình 6: Minh họa cách cắt ảnh effect

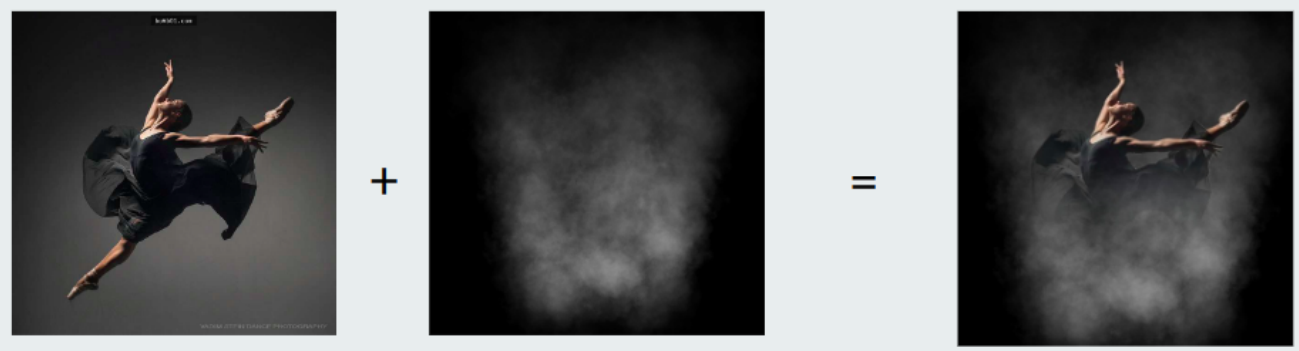
d. Blending image

Blending là kĩ thuật "trộn" 2 ảnh với nhau để tạo ra ảnh mới với sự có mặt của 2 ảnh trước. Trong đồ án này, em sử dụng kĩ thuật Alpha blending để trộn ảnh S và ảnh E với nhau.

Công thức của alpha blending:

```
dst = image * alpha + effect * (1-alpha)
```

Ví dụ:



Hình 7: Xử dụng alpha blending để tạo ra ảnh có hiệu ứng đẹp.

2. Điểm mạnh điểm yếu và hướng phát triển

Điểm mạnh:

- Detect pose từ Mediapipe nhanh, kỹ thuật blending image đơn giản, giúp cho code chạy nhanh, có thể đưa vào các ứng dụng realtime.
- Nhiều effect đẹp trên internet. Ta chỉ cần chọn lựa hiệu ứng ưng ý nhất.

Điểm yếu:

- Khó custom nội dung effect theo ý mình (do effect là video có sẵn), muốn custom lại cần có kiến thức thật vững về Photoshop.

Hướng phát triển:

- Xây dựng trang web giao diện người dùng cho phép user upload video hiệu ứng bất kì, chọn các custom phù hợp để không cần sửa code.

3. Video demo

https://www.youtube.com/watch?v=kS-PNAT-Cp4&t

Source code: https://github.com/lynguyenminh/CS231.Neon-Effect.git

Phần 3: Nguồn tham khảo

[1].https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d

[2].https://www.geeksforgeeks.org/python-opencv-cv2-copymakeborder-method/

[3].https://docs.opencv.org/3.4/d5/dc4/tutorial_adding_images.html