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

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

\*===\*===\*===\*===\*===\*



**BÁO CÁO**

**CHỦ ĐỀ: “CAMERA 360”**

**Môn học:** C338.K11 – Nhập môn Thị giác máy tính

**Giảng viên hướng dẫn:** ThS. Nguyễn Vinh Tiệp

**Sinh viên:** 17520342 – Trần Tấn Đạt

17520297 – Lê Duy Thành Công

17520449 – Nguyễn Thị Phương Hảo

Thành phố Hồ Chí Minh, ngày 01 tháng 01 năm 2020

**LỜI NÓI ĐẦU**

Môi trường Đại học là môi trường học tập khác biệt so với các cấp cơ sở trung học và phổ thông trung học. Sự năng động của Đại học gắn liền với giảng đường, gắn liền với từng môn học với kiến thức hàn lâm. Điều đó kéo theo sự năng động của mọi thứ kể cả sinh viên. Với việc học tập để hiểu và áp dụng kiến thức đã học vào thực tiễn là điều bắt buộc với sinh viên.

Nhập môn Thị giác máy tính là môn học như thế. Với kiến thức rộng lớn và xuyên suốt đòi hỏi sinh viên luôn phải nỗ lực tìm hiểu và không ngừng đào sâu. Nhưng đó cũng là một điều tốt tạo nên cách học và cách sinh hoạt hiệu quả cho bất kì ai.

Chủ đề để thực hiện đồ án trong môn học này rất nhiều, các chủ đề xoay quanh lĩnh vực Thị giác máy tính như: swap face, thực tại tăng cường, 360 panorama,... Với mục đích cung cấp cho sinh viên kiến thức cơ bản về Thị giác máy tính, các thuật toán dùng trong Thị giác máy tính như: lọc ảnh filter, làm mịn median filter, hough transform,...

Trong đó “Camera 360” sẽ được tìm hiểu kĩ trong đồ án này bởi nhóm sinh viên. Chủ đề này sẽ cho chúng ta hiểu rõ hơn về việc áp dụng kiến thức đã học để tạo ra sản phẩm, các kiến thức không cao siêu nhưng có thể làm ra những thứ rất hay và thú vị, điều đó cho thấy môn học này thực sự bổ ích và không quá đòi hỏi nhiều từ sinh viên.

Với chủ để này, nhóm sinh viên giải quyết các vấn đề:

**Bài toán Camera 360?!**

**Thuật toán để giải quyết bài toán?!**

**Thực nghiệm chạy chương trình?!**

**Kết quả thu được?!**

**Đánh giá và kết luận?!**

Sau quá trình học tập và đúc kết kiến thức điều quan trọng nhất là hiểu biết sâu về một lĩnh vực nào đó. Áp dụng những thứ đó vào học tập các môn học sau này. Không nhất thiết phải giỏi hết tất cả các chủ đề trong môn này nhưng mỗi thứ chỉ được một ít, sau đó rồi lại quên không áp dụng gì được.

**LỜI CẢM ƠN**

Nhóm sinh viên chúng em xin gửi lời cảm ơn chân thành và sâu sắc tới Thầy Nguyễn Vinh Tiệp. Suốt một học kì qua nhóm sinh viên chúng em đã nhận được sự chỉ bảo rất tận tình, tâm huyết từ thầy. Trải qua một kì học tập với nhiều kiến thức thu nhặt được từ môn “Nhập môn Thị giác máy tính”, chúng em có được kiến thức bổ ích về lý thuyết cũng như thực hành.

Trong quá trình học môn này, chúng em được nhận đồ án với chủ đề “Camera 360”. Đây là một thách thức với nhóm sinh viên chúng em, nhưng khi nhìn lại đây là một cơ hội rất tốt cho chúng em được chứng minh năng lực của mình, vận dụng những kiến thức đã được học vào thực tiễn. Hơn thế nữa, đây là dịp cho chúng em học hỏi kinh nghiệm, được lắng nghe những lời nhận xét chân thành từ Thầy. Đó là nền tảng cho chúng em có được nhiều kiến thức và trải nghiệm để học tập và làm việc sau này.

Nhóm sinh viên chúng tôi cũng gửi lời cảm ơn đến các bạn lớp CS231.K11, trong quá trình học tập và làm đồ án chúng tôi đã nhận được sự tư vấn chia sẻ rất quý báu về kiến thức và sự động viên rất chân thành từ các bạn. Có thể khẳng định rằng đó là nguồn động lực và là yếu tố rất cần thiết để chúng tôi hoàn thành đồ án này.

Để hoàn thành đồ án này là sự nỗ lực rất lớn của nhóm sinh viên chúng em. Chúng em đã cố gắng hết sức mình cho công việc, từ việc tìm kiếm nguồn tài liệu, tìm tòi thông tin trên các sách, báo, trang mạng cho đến những buổi họp nhóm thường xuyên để làm tốt đồ án. Đồ án này là sự tâm huyết, lòng nhiệt thành, sự bao dung của tất cả các thành viên trong nhóm. Nhưng dù đã nỗ lực hết sức vẫn không thể tránh được sai xót và khuyết điểm trong đồ án này. Nhóm sinh viên chúng em mong muốn nhận được sự nhận xét và đóng góp của Thầy cũng như các bạn để chúng em hoàn thiện và làm tốt hơn nữa.

Xin chân thành cảm ơn!

Thành phố Hồ Chí Minh, ngày 01 tháng 01 năm 2020.

**CHƯƠNG 0: GIỚI THIỆU VỀ ĐỒ ÁN**

1. **Đồ án.**

Chủ đề: Camera 360.

Môn học: CS231.K11

1. **Giảng viên và thành viên nhóm.**

Giảng viên hướng dẫn: Thầy Nguyễn Vinh Tiệp.

Sinh viên thực hiện:

17520342 – Trần Tấn Đạt

17520644 – Lê Duy Thành Công

17520449 – Nguyễn Thị Phương Hảo

1. **Thời gian hoàn thành.**

Đồ án hoàn thành và chỉnh sửa lần cuối:

Thời gian: 23h59 ngày 08 tháng 01 năm 2020.

Địa điểm: Đại học Quốc gia Thành phố Hồ Chí Minh, Q. Thủ Đức, TP. Hồ Chí Minh.

**CHƯƠNG 1: GIỚI THIỆU VỀ CHỦ ĐỀ**

* + - 1. **Camera 360.**

Camera 360 là một sản phẩm kết hợp camera trên điện thoại thông minh với hệ thống xử lí hình ảnh để cuối cùng cho ra những tấm ảnh được chỉnh sửa theo ý muốn người dùng. Các tấm ảnh đã được xử lí cho qua việc lọc hoặc thêm bớt các sticker để cho ra kết quả. Camera 360 khác với camera 360 độ là sản phẩm có thể ghi hình ở góc rộng đến 360 độ.

Camera 360 đã không còn quá xa lạ với chị em phụ nữ vì họ luôn luôn cần đẹp trong mọi trường hợp, hoặc họ sẽ có những tấm ảnh để sống ảo với bạn bè. Việc phát triển ứng dụng hứa hẹn một thị trường rất tiềm năng và không lỗi mode.

Khi phát triển camera 360 nhóm đã theo hai hướng để đáp ứng chức năng cơ bản nhất của ứng dụng gồm:

Lọc ảnh: sử dụng các bộ lọc filter như: làm mịn, khảm mờ, tăng độ sáng tối, tạo ảnh màu sắc, …

Gắn sticker: gồm các chức năng để tạo ra bức ảnh lung linh, xinh động như: tạo phông nền tuyết, gắn mặt nạ, gắn sticker, …

* + 1. **Cải tiến.**

Nhóm thực hiện chủ yếu trên ảnh vì hình ảnh cho chất lượng sắc nét hơn và đơn giản hơn. Tuy nhiên sau lần thuyết trình trên lớp với góp ý của thầy và các bạn nhóm đã cải tiến để có thể chạy trên webcam một số chức năng: gắn sticker và đổi màu trên bức ảnh.

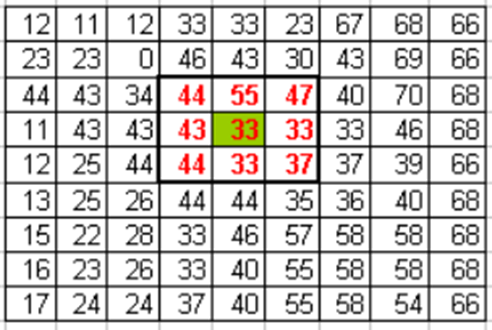
Do nhóm làm nhiều chức năng và các chức năng khá đơn giản nên nhóm không làm video demo. Chỉ cần cho ảnh vào và chạy là được.

**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

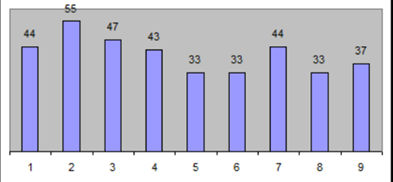
1. **Bộ lọc median.**

Lọc Median hay còn gọi là lọc trung vị khác với bộ lọc trung bình ở chỗ nó không xảy ra bất cứ phép tính toán gì, nó chỉ đơn giản là chỉ ra đâu là giá trí nằm ở vị trí trung bình trong các giá trị đầu vào được thiết lập bởi khung lọc rồi lấy giá trị đó thế vào tâm của khung lọc trong ảnh gốc để tạo ra giá trị ảnh mới.

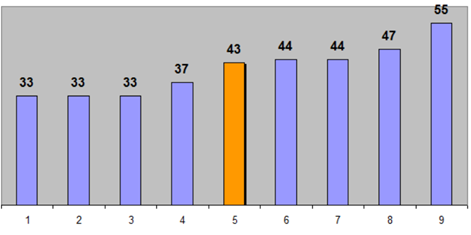
Với nhân chập 3x3 thì với một tập điểm ảnh như sau:



Giả sử vùng dữ liệu ảnh hiện có như trên thì khi sử dụng nhân 3x3 của lọc trung vị ta có các giá trị của nhân là:



Sau khi sắp xếp theo độ lớn tăng dần thì ta có



Nên giá trị của điểm ảnh sau khi lọc trung vị là 43

1. **Bộ lọc Gauss.**

Bộ lọc Gauss được cho là bộ lọc hữu ích nhất, được thực hiện bằng cách nhân chập ảnh đầu vào với một ma trận lọc Gauss sau đó cộng chúng lại để tạo thành ảnh đầu ra.

Ý tưởng chung là giá trị mỗi điểm ảnh sẽ phụ thuộc nhiều vào các điểm ảnh ở gần hơn là các điểm ảnh ở xa. Trọng số của sự phụ thuộc được lấy theo hàm Gauss (cũng được sử dụng trong quy luật phân phối chuẩn).

1. **Bộ lọc song phương.**

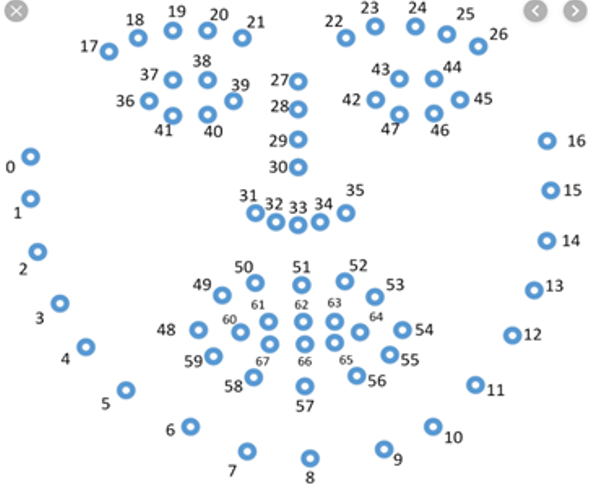
Đây là một bộ lọc hiệu quả cao trong việc loạt bỏ nhiễu mà vẫn giữ lại được các đường viền (cạnh) trong ảnh.

Như chúng ta đã biết, bộ lọc Gauss quyết định giá trị một điểm ảnh bằng cách lấy trung bình theo hàm Gauss các giá trị điểm ảnh xung quanh điểm đó. Hàm trọng số Gauss chỉ phụ thuộc vào khoảng cách trong không gian so với điểm ảnh trung tâm, không quan tâm đến sự tương quan giữa mức xám của điểm trung tâm với các điểm xung quanh đó. Nó cũng không quan tâm rằng điểm ảnh trung tâm có nằm tại một đường biên trong ảnh không, vì thế làm nhòe luôn các đường biên trong ảnh.

Bộ lọc Bilateral cũng sử dụng một bộ lọc Gauss với khoảng cách đến điểm trung tâm, đảm bảo chỉ có các điểm ở gần tham gia vào giá trị của điểm ảnh trung tâm. Tuy vậy nó sử dụng thêm một hàm Gauss cho mức xám, đảm bảo chỉ các điểm ảnh có mức xám tương đồng với điểm ảnh trung tâm tham gia vào quá trình làm mịn. Vì thế bộ lọc Bilateral bảo toàn được các đường biên trong ảnh bởi vì điểm ảnh ở biên có sự thay đổi về mức xám rất rõ ràng. Hơn nữa, thay vì hoạt động trên các kênh màu một cách riêng rẽ như bộ lọc trung bình hay bộ lọc Gauss, bộ lọc Bilateral thi hành việc đo đạc màu sắc có chủ đích trong không gian màu CIE-Lab , làm mượt màu và bảo toàn các biên theo hướng phù hợp hơn với nhận thức con người.

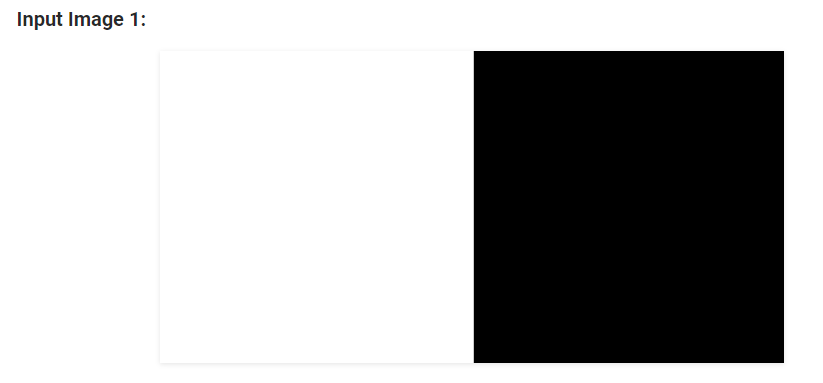
Tuy vậy, bộ lọc Bilateral cũng có nhược điểm là chậm hơn các bộ lọc khác.

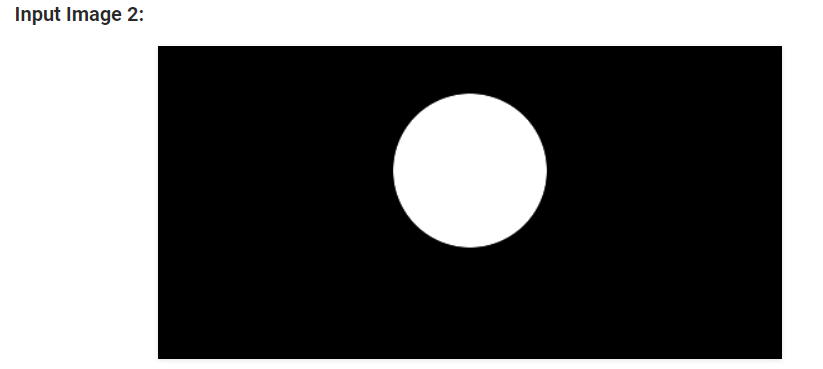
1. **Facial Landmark.**



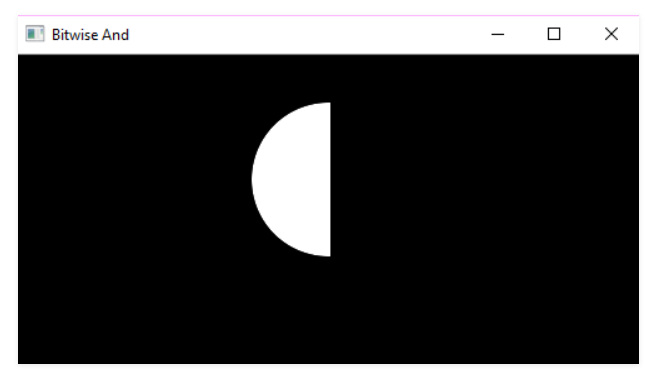
# **Bitwise Operations on Binary Images.**

Cho 2 bức ảnh sau, ta lần lượt thực hiện các phép toán trên 2 bức ảnh này sẽ ra được các kết quả sau:

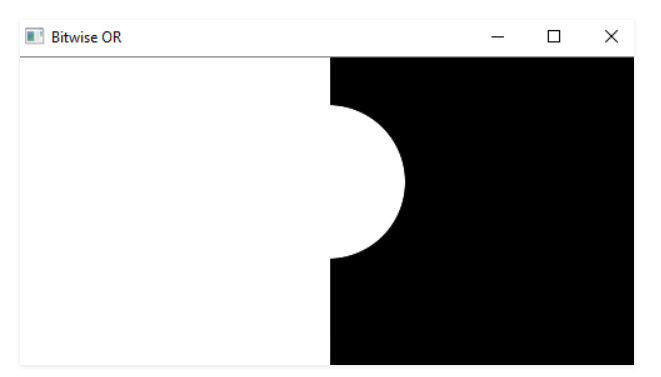




* + - * 1. **Phép Bitwise AND operation on Image:**



* + - * 1. **Phép Bitwise OR operation on Image:**

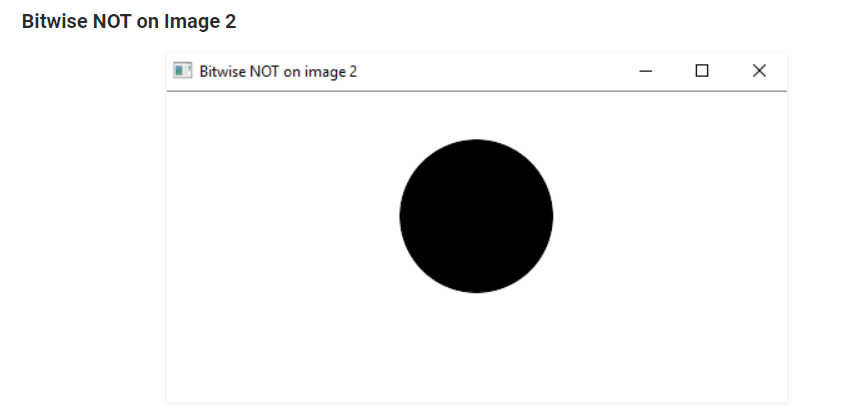


* + - * 1. **Phép Bitwise XOR operation on Image:**



* + - * 1. **Phép Bitwise NOT operation on Image:**





1. **Phân ngưỡng ảnh.**

Nếu pixel có giá trị lớn hơn giá trị ngưỡng thì nó được gán 1 giá trị (thường là 1), ngược lại nhỏ hơn giá trị ngưỡng thì nó được gán 1 giá trị khác (thường là 0).

**THRESH\_BINARY**

* + Nếu giá trị pixel lớn hơn ngưỡng thì gán bằng maxval
  + Ngược lại bằng gán bằng 0.

**THRESH\_BINARY\_INV**

* + Nếu giá trị pixel lớn hơn ngưỡng thì gán bằng 0
  + Ngược lại bằng gán bằng maxval

**THRESH\_TRUNC**

* + Nếu giá trị pixel lớn hơn ngưỡng thì gán giá trị bằng ngưỡng
  + Ngược lại giữ nguyên giá trị

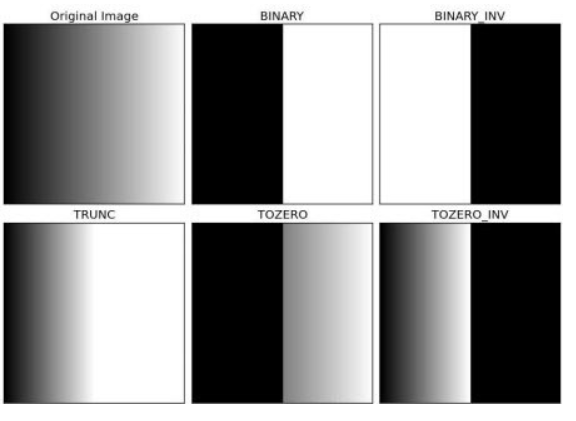
**THRESH\_TOZERO**

* + Nếu giá trị pixel lớn hơn ngưỡng thì giữ nguyên giá trị
  + Ngược lại gán bằng 0

**THRESH\_TOZERO\_INV**

* + Nếu giá trị pixel lớn hơn ngưỡng thì gán giá trị bằng 0
  + Ngược lại giữ nguyên

Ví dụ: với ngưỡng 127.



1. **Gamma correction.**

Gamma correction  có thể được sử dụng để điều chỉnh độ sáng của hình ảnh bằng cách sử dụng phép biến đổi phi tuyến giữa các giá trị đầu vào và giá trị đầu ra được ánh xạ:

O=I^(1/G)

O là giá trị pixel đầu ra

I là giá trị pixel đầu vào

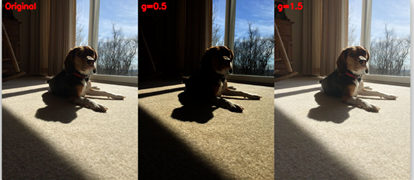
G là giá trị gama

Thay đổi giá trị gama sẽ tạo ra nhiều hiệu ứng

Giá trị gamma  *<1* sẽ dịch chuyển hình ảnh về phía cuối phổ tối hơn

giá trị gamma  *> 1* sẽ làm cho hình ảnh trông sáng hơn.

Giá trị gamma của  *G = 1* sẽ không ảnh hưởng đến hình ảnh đầu vào:



**CHƯƠNG 3: CÀI ĐẶT CHƯƠNG TRÌNH**

1. **Lọc mụn.**
2. **Input và output.**

|  |  |
| --- | --- |
| Input: 1 tấm ảnh cận mặt có mụn | https://lh5.googleusercontent.com/Fpq8ZF_zZtdR4V-jbzMsc4MRdYbclD-Sw7yed11aFZ0wEO8ZvDqFAhh6R9_mXoeElpWCEmtfPPgux-1MOprkLYUznXIrVQYDSp-Q52INg7canRSdhncoaZy0hocGI_t4RaiOZrR5 |
| Output: tấm ảnh sau khi làm mờ mụn (mức độ mờ mụn phụ thuộc vào việc chọn ma trận để nhân chập). |  |

1. **Quá trình thực hiện.**

Áp dụng bộ lọc median ta được:

|  |  |  |
| --- | --- | --- |
| https://lh3.googleusercontent.com/9-QTtavxoe0T4HQmT_Z2U7W3gbpsW1AZjJNHGS718T6ztt9nunDp0UCYxt1e_XyibSLfPMGJenbhB8-vNgSLSUt2aN-ffkKRuP9XPhNpGPfpAAbOzd7MBFDuv_G-0HLdUrbunwRf | https://lh6.googleusercontent.com/qkqqcyD9m5N22kL9knVW1QMq1zNIXto-QeQGTSRHv5cfo6LK3NqO_M2aGxvZBlESFQErCMkfVn-1gxrM63tugZYFYy1JrWVCS8Euny3UhZ_KgPXFaUTQ77BjDgISrJkKymjtQ2sY | https://lh4.googleusercontent.com/WEngVSnEmbrC-owvrYc8KaDI-uhXTJQYbRnNxghOhx1PPnh4M2PBUTlvWd7mo9D_2WybNRJOSE6li4s08h2XHYk2znBdF86pOgALDQ0J4lKpuOLwwIUiYJOZ4IaLPSXQCHA-WyyF |
| Nhân chập 5x5 | Nhân chập 7x7 | Nhân chập 9x9 |

Sử dụng bộ lọc median ta thấy rằng nó làm mờ mụn nhỏ nhưng đối với mụn lớn thì khó làm mờ được.

Với kich thước kernel càng lớn thì mụn sẽ càng mờ đi, tuy nhiên lúc đó nó sẽ làm nhòe đi các vùng khác khiến ảnh mất tự nhiên. Để giải quyết vấn đề đó, nhóm đã tìm tòi và thấy được việc sử dụng giải pháp mới là sử dụng **bộ lọc song phương** sẽ cho ra kết quả tốt hơn.



**2. Đeo mặt nạ.**

1. **Input, output.**

|  |  |
| --- | --- |
| Input: 1 tấm ảnh mặt người (.jpg) và 1 tấm ảnh mặt nạ (.jpg) | https://lh4.googleusercontent.com/9o0yGI-CCZbu-RCJ2ttEf6-S0OR--R4XPVkd29EIp_8SJzL9dy2ua37-9yhKx_9v5w9h2hjXdb-_VmPUF22Ub4qFuZ5uwhn93GfNNTHWrdrAmxMIcn59kmsOmSQTZaKkVHTitdv8https://lh3.googleusercontent.com/6mUtoQtmVxSx4EldbXFkaj8T_LQ2VWwK33vofZXnjMu1_P-H6CaNSGw9jjaTmeQKS-7pVZYLsFcUnvy3o8Sut-ResxRQm2r5UYdGzWoDzYAUcVmfpXfnMB2LcE5Fk1loaTPHVkW_ |
| Output: tấm ảnh người đã ghép mặt nạ | https://lh5.googleusercontent.com/pkoKQcviv22FC5SZ1-Y4-ZHQ8LtFG4VfBGTjP1XqkTrbR64-EK6TXYXghtYd1ItrleCHcW9CNA2KrD_rwyoxK2sjESw0KcEPbss0KEEKMcB-78oEyS4l4eBqAaJ-iAj_2KW1VSJ6 |

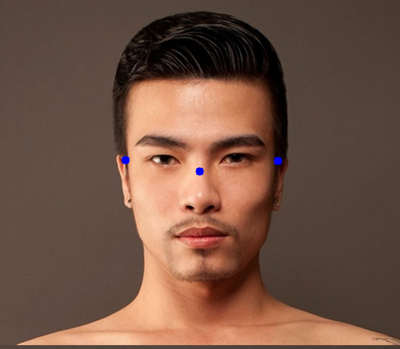
1. **Quá trình thực hiện.**

Các bước thực hiện:

**Bước 1:** Detect khuôn mặt:

Sử dụng thư viện Dlib xác định khuôn mặt bằng landmark.

Xác định 3 điểm chính trên khuôn mặt:



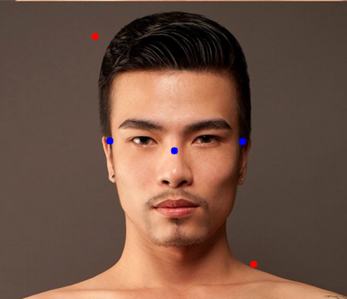
Điểm Left: 0 Điểm Center: 27 Điểm Right: 16

Tiến hành tính toán chiều rộng, chiều cao của khuôn mặt. Chiều rộng: Left - Right

Vì landmark không detect hết các điểm phía trên trán, điểm cao nhất của nó chỉ ở lông mày nên để tính được chiều cao khuôn mặt để sao cho áp mặt nạ không bị lệch,tụi em sẽ ước lượng tỉ lệ chiều cao với chiều rộng.

Sau nhiều lần thực nghiệm với nhiều bức ảnh thì được kết quả ước lượng: Chiều cao = 1.3\*chiều rộng

Từ đó, xác định vị trí Top-left, Bottom-right



**Bước 2:** Xử lí ảnh mặt nạ: làm cho nền (vùng trắng quanh ảnh mặt nạ trong suốt).

Chuyển về ảnh xám.

Resize ảnh mặt nạ về kích thước vùng mặt đã được tính toán, để áp vô cho đúng.

Chuyển ảnh mặt nạ về ảnh nhị phân (sử dụng**threshold**)



Tiếp theo, ta đảo ngược lại hình ảnh mặt nạ vừa rồi: sử dụng mặt nạ nghịch đảo để lấy phần còn lại của hình ảnh. Viền bao xung quanh giờ đã không còn màu nữa.



Chỉ lấy có nên trắng bên ngoài vùng mặt nạ từ hình ảnh mặt nạ.



**Bước 3:** Đặt mặt nạ vô đúng vị trí trên khuôn mặt.

Bôi đen phần mặt nạ trong khuôn mặt vừa detect được(ROI)



Đặt mặt nạ vào ROI và sửa đổi hình ảnh chính



Tuy nhiên lúc này có một vấn đề: khi xoay ảnh, chiếc mặt nạ không quay theo?

Nhóm tiếp tục tìm cách giải quyết vấn đề này và cuối cùng ra được cách giải quyết trong chức năng sticker.

**3. Gắn Sticker**

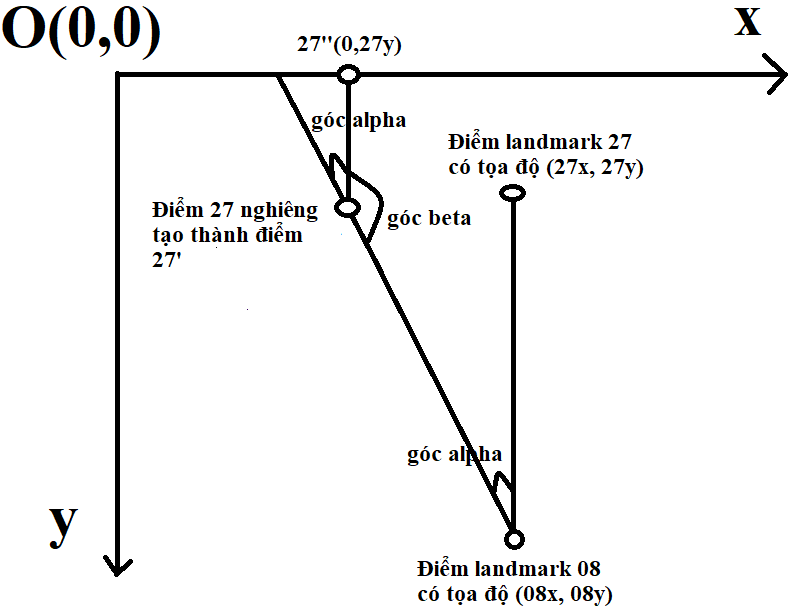
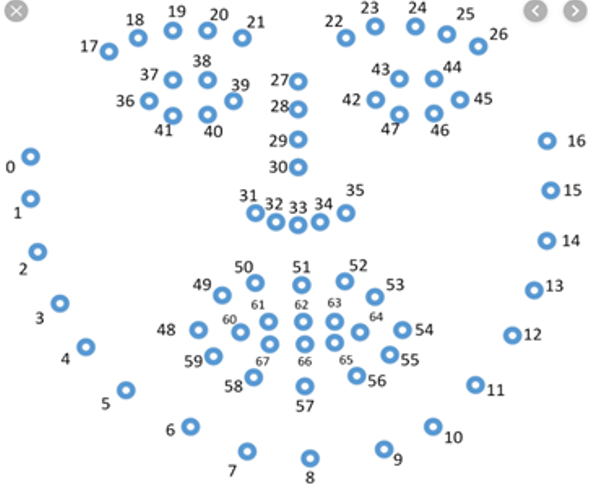
**a. Input và output.**

|  |  |
| --- | --- |
| Input: 1 tấm ảnh mặt người (.jpg) và 1 tấm sticker (.jpg) |  |
| Output: tấm ảnh người đã ghép sticker |  |

1. **Quá trình thực hiện.**

Quá trình thực hiện giống với cách thực hiện gắn mặt nạ. Ở đây có cải tiến thêm về việc xoay mặt nạ theo mặt người.

Nhóm đã thử nhiều hàm xoay mặt nạ có sẵn nhưng chưa thực sự hài lòng với kết quả, vì lí do đó nhóm quyết định tính thủ công bằng cách sau:

Như ta thấy trên hình có 68 điểm landmark, từ điểm 27 đến điểm 8 sẽ tạo ra một đường thẳng đứng. Giả sử khi gương mặt bị nghiêng thì đường thẳng này sẽ nghiêng một góc alpha, làm điểm 27 nghiêng thành điểm 27’. Điểm 27’ sẽ chiếu lên trục Ox tạo thành điểm 27’’ nằm trên trục Ox. Lúc này điểm 27’’ sẽ có tọa độ 27’’(0,27y)

Ta đã tìm ra tọa độ 3 điểm 08, 27’ và 27’’ và tính toán sẽ ra được góc beta. Lấy 180 – beta = alpha, góc vừa tính là góc nghiêng bên trái. Nếu gương mặt nghiêng bên phải thì ta lấy 360-alpha vừa tìm thì ra được góc cần tìm, ta xoay ảnh mặt nạ theo góc đó thì mặt nạ và khuôn mặt sẽ khớp nhau.

Có một thông số gọi là **sai số khách quan** (sskq) được cộng thêm vào nếu cần để cho ra mặt nạ khớp hoàn toàn với gương mặt, điều đó sẽ cho ra bức ảnh thật sự hoàn hảo. Sai số khách quan cần có vì gương mặt khi bị nghiêng nó không nghiêng bình thường mà nó sẽ bị khuất đi một vùng nào đó như phần má, làm gương mặt không được đều. Thêm một lý do là các sticker thiết kế chủ yếu cho gương mặt thẳng đứng nên áp vào gương mặt nghiêng sẽ không tự nhiên nên cần có sskq. Nhóm đã tính toán và ước lượng sskq thường dao động từ 10-15 sẽ cho ra kết quả tốt nhất.

**4. Làm sáng da**

1. **Input, output**

Input: 1 tấm ảnh muốn tăng sáng

Output: 1 tấm ảnh đã làm sáng

1. **Quá trình thực hiện**

Nhóm áp dụng phương pháp nomolize tuy nhiên ảnh sáng không tự nhiên. Sau khi tìm hiểu nhóm đã tìm được phương pháp làm sáng mới được cho là tốt hơn và cho ra tấm ảnh thực sự tự nhiên, đó là phương pháp **Gamma correction**.

Đây là kết quả của phương pháp này với gama=2



**5. Thêm nền tuyết**

1. **Input và output.**

|  |  |
| --- | --- |
| Input:  2 bức ảnh, 1 là ảnh nền cần ghép phông, 1 bức ảnh chưa phông cần ghép (có nền đen, xanh hoặc xanh dương) |  |
| Output: 1 bức ảnh đã được ghép phông vào |  |

1. **Quá trình thực hiện.**

Áp dụng phương pháp **Coluor threshold** là ngưỡng màu, dùng để phần biết giữa nền xanh (xanh dương hoặc đen) so với vật thể cần ghép vào 1 bức ảnh nào đó**.**

Cách thực hiện:

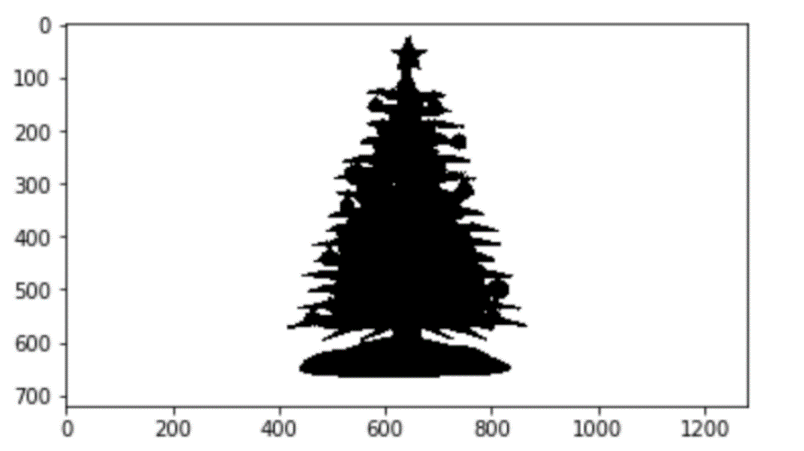
**Bước 1:** Tạo 1 ngưỡng màu cần tách: chọn màu xanh dương

* lower\_blue = np.array([0, 0, 100]) ##[R value, G value, B value]   
  upper\_blue = np.array([120, 100, 255]).

**Bước 2:** Tạo 1 ảnh mức xám mà đã được tách phông ở trên nhờ dùng hàm inRange() của opencv.

                mask = cv2.inRange(image\_copy, lower\_blue, upper\_blue)

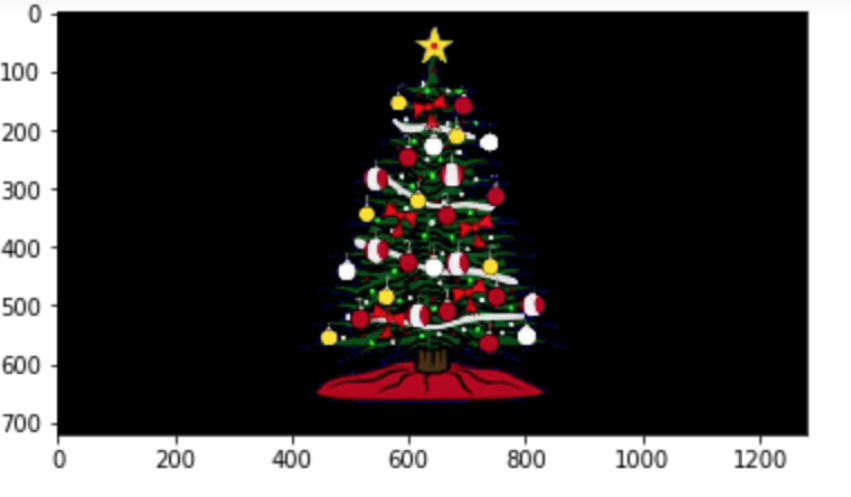
plt.imshow(mask, cmap='gray').



**Bước 3:** Tạo bức ảnh từng bức ảnh mask, những nơi trong ảnh mask khác màu đen thì đặt lại màu đen.

                masked\_image = np.copy(image\_copy)

                masked\_image[mask != 0] = [0, 0, 0]



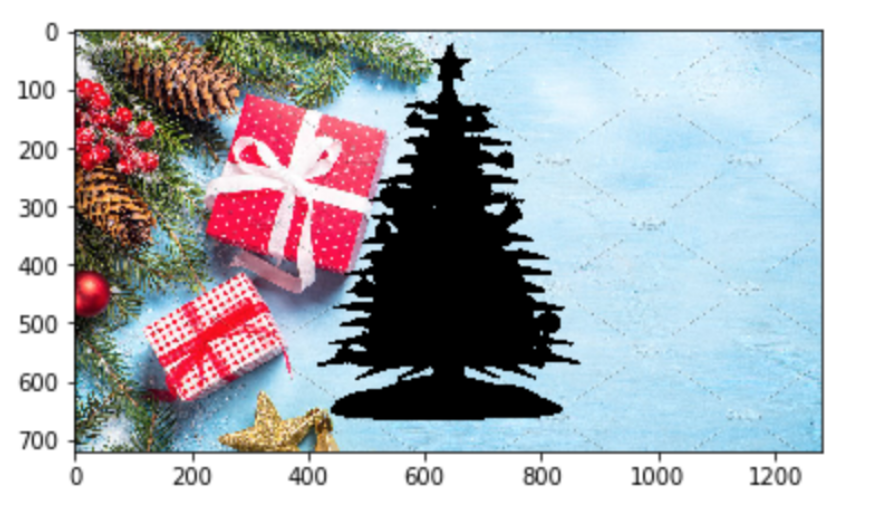
**Bước 4:** Load ảnh cần ghép nền vào, resize bức ảnh lại cho cùng với kích thước của ảnh phông. Sau đó nhưng nơi trên ảnh cần ghép nền sẽ set thành màu đen nếu ở đó mask là màu đen, bước này để xem liệu phông cần ghép vào có đúng vị trí, phù hợp hay không.

  background\_image = cv2.imread('images/treeBackground.jpg')

  background\_image = cv2.cvtColor(background\_image, /cv2.COLOR\_BGR2RGB)

crop\_background = background\_image[0:720, 0:1280]

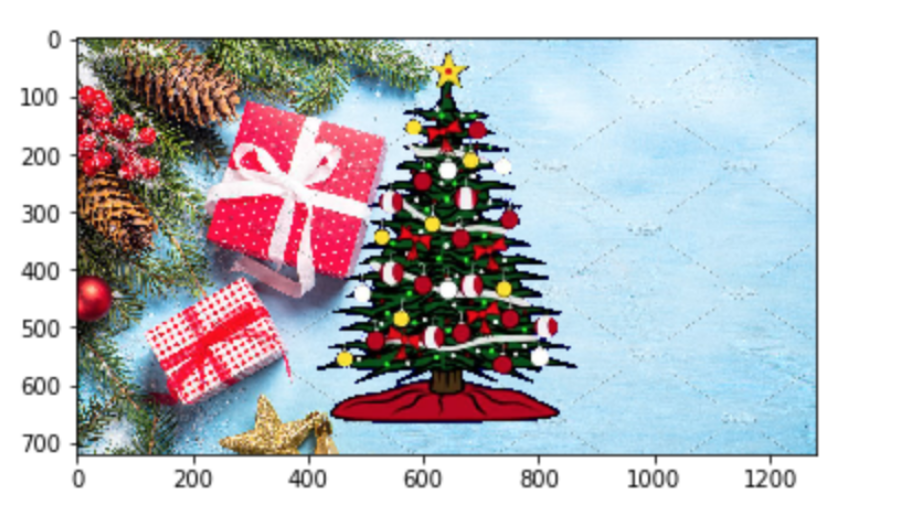
crop\_background[mask == 0] = [0, 0, 0]



**Bước 5:** Ghép ảnh:

final\_image = crop\_background + masked\_image

plt.imshow(final\_image)



**6. Làm cong ảnh**

1. **Input và output**

Input: 1 tấm ảnh

Output: 1 tấm ảnh cong theo sóng dọc, 1 tấm ảnh cong theo sóng ngang

1. **Quá trình thực hiện**

Như ta đã biết sóng cơ dao động theo một đồ thị hình sin Dựa vào tính chất đó, tụi em áp dụng vào việc làm cong ảnh theo đường sóng. mỗi điểm ban đầu sẽ thay đổi 1 khoảng: độ dời: delta=(2\*pi\*w/lamda).

**7. Thay đổi màu trên bức ảnh.**

1. **Input và output**

|  |  |
| --- | --- |
| Input: Bức ảnh RGB bình thường | https://lh3.googleusercontent.com/HGLoqrzYIOJA0eYSOPMLLs_dJm8I0sylWBZuMfZCo4DXCFCoviyL400kbjVYptmHpdCUvMxaTtX4F0nYOLz80LiznuBQKr2hs7v40WciL_zz6altQ-En-jj0UqKK1EStSctsml3phttps://lh6.googleusercontent.com/T0DwExeaycVr2guzAFzWuf5MyaeGA2vPY_qtoU8r0UJOYJbtW5TSP6nG030JyO666zqfJ6QT0ZMdgsknRt5n4tHfi82l_XAXL5E70EObnAVAAKKCdZtXpdlIaQtCSnuL-56jw6tF |
| Output: Bức ảnh được thay đổi nền theo 3 kiểu: mức xám, ảm bản, theo màu sắc | https://lh5.googleusercontent.com/hnSxcASI3NWAxuthtBlXx2Amt764clrB-pprXYjldkajgBWX1fbcMtsVi03R7EnsZSAZ55V6hqsHuTmdULqYblOXDrDRhdqP-Ej1rMCSX7J-0uDox8OrHExIBahBTyQBAntsR8Ks |

1. **Quá trình thực hiện:**

**Ảnh mức xám:**

Gọi hàm từ thư viên opencv:

gray = cv2.cvtColor(img3, cv2.COLOR\_BGR2GRAY)

**Ảnh âm bản:**

img1 = cv2.imread("chipu.jpg")

cv2.imshow("Pic", img1)

# Nagative Image

img\_not = cv2.bitwise\_not(img1)

**Ảnh theo màu sắc tùy thích (màu đỏ):**

Cách thực hiện: tạo 1 bức ảnh màu đỏ, trộn ảnh đỏ với ảnh gốc theo 1 tỉ lệ tùy chọn:

target\_image = cv2.imread('chipu.jpg')

h, w, \_ = target\_image.shape

# create an image with a single color (here: red)

red\_img = np.full((h, w, 3), (0, 0, 255), np.uint8)

cv2.imshow('red', red\_img)

used\_img  = cv2.addWeighted(target\_image, 0.8, red\_img, 0.2, 0)

cv2.imshow('b',used\_img)