|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **TRƯỜNG ĐIỆN – ĐIỆN TỬ**  logo_128  **BÁO CÁO BÀI TẬP LỚN**  **MÔN HỌC: XỬ LÝ ẢNH SỐ**  **Đề tài 5:** Tìm hiểu và xây dựng các bộ lọc xử lý ảnh  **Giảng viên hướng dẫn:** TS. Lê Thị Lan  **Sinh viên thực hiện:**   |  |  |  | | --- | --- | --- | | Họ và tên | MSSV | Lớp | | Hồ Sỹ Thạch | 20182769 | ET1 – 04 – K63 | | Trần Thị Hường | 20182570 | ET1 – 04 – K63 | | Nguyễn Quang Thu Phương | 20182725 | ET1 – 03 – K63 | | Trần Đức Trung | 20182838 | ET1 – 07 – K63 |   Hà Nội, 07-2022 |

**MỤC LỤC**

[DANH SÁCH HÌNH VẼ 1](#_Toc109478945)

[CHƯƠNG 1. KHÁI QUÁT VỀ XỬ LÝ ẢNH VÀ LỌC ẢNH 2](#_Toc109478946)

[1.1 Khái quát về xử lý ảnh 2](#_Toc109478947)

[1.1.1 Xử lý ảnh 2](#_Toc109478948)

[1.1.2 Các bước cơ bản trong xử lý ảnh 3](#_Toc109478949)

[1.1.3 Một số khái niệm cơ bản trong xử lý ảnh 3](#_Toc109478950)

[1.2 Lọc ảnh 4](#_Toc109478951)

[1.2.1 Khái niệm 4](#_Toc109478952)

[1.2.2 Nguyên tắc chung của lọc ảnh 4](#_Toc109478953)

[1.2.3 Các phép toán cơ bản 5](#_Toc109478954)

[CHƯƠNG 2. TÌM HIỂU MỘT SỐ BỘ LỌC VÀ ỨNG DỤNG 7](#_Toc109478955)

[2.1 Bộ lọc trung bình (Mean Filter) 7](#_Toc109478956)

[2.2 Bộ lọc Gaussian 7](#_Toc109478957)

[2.3 Bộ lọc Unsharp mask 8](#_Toc109478958)

[2.4 Bộ lọc trung vị (Median Filter) 8](#_Toc109478959)

[2.5 Bộ lọc Sobel 9](#_Toc109478960)

[2.6 Bộ lọc Laplacian 10](#_Toc109478961)

[2.7 Bộ lọc ứng dụng Fourier transform 10](#_Toc109478962)

[2.8 Bộ lọc hiệu ứng mùa đông 11](#_Toc109478963)

[2.9 Bộ lọc Wiener 11](#_Toc109478964)

[2.10 Bộ lọc ứng dụng Fourier transform 11](#_Toc109478965)

[CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM 13](#_Toc109478966)

[3.1 Lập trình một số bộ lọc làm mịn ảnh 13](#_Toc109478967)

[3.1.1 Bộ lọc trung bình (Mean Filter) 13](#_Toc109478968)

[3.1.2 Bộ lọc Gaussian 14](#_Toc109478969)

[3.1.3 Bộ lọc Unsharp mask 15](#_Toc109478970)

[3.1.4 Bộ lọc trung vị (Median Filter) 15](#_Toc109478971)

[3.2 Lập trình một số bộ lọc tìm biên ảnh 16](#_Toc109478972)

[3.2.1 Bộ lọc Sobel 16](#_Toc109478973)

[3.2.2 Bộ lọc Laplacian 17](#_Toc109478974)

[3.3 Một số bộ lọc khác 18](#_Toc109478975)

[3.3.1 Bộ lọc ảnh ứng dụng DFT 18](#_Toc109478976)

[3.3.2 Bộ lọc hiệu ứng mùa đông 19](#_Toc109478977)

[CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG ÁP DỤNG CÁC BỘ LỌC 20](#_Toc109478978)

[4.1 Chuẩn bị 20](#_Toc109478979)

[4.1.1 Tổng quan về lập trình giao diện với Python 20](#_Toc109478980)

[4.1.2 Tổng quan về thư viện Tkinter 20](#_Toc109478981)

[4.2 Phát triển ứng dụng 21](#_Toc109478982)

[CHƯƠNG 5. KẾT LUẬN 22](#_Toc109478983)

[TÀI LIỆU THAM KHẢO 23](#_Toc109478984)

# DANH SÁCH HÌNH VẼ

[Hình 1.1 Quá trình xử lý ảnh 2](#_Toc109478926)

[Hình 1.2 Các bước cơ bản trong xử lý ảnh 3](#_Toc109478927)

[Hình 1.3 Minh hoạ nhân ma trận ảnh 5](#_Toc109478928)

[Hình 1.4 Ví dụ phép tích chập 6](#_Toc109478929)

[Hình 2.1 Ví dụ kernel của bộ lọc trung bình 7](#_Toc109478930)

[Hình 2.2 Ví dụ kernel của bộ lọc Gausian 8](file:///C:\Users\Dell\Downloads\Report_XLAS.docx#_Toc109478931)

[Hình 3.1 Kết quả khi đưa ảnh qua bộ lọc trung bình 13](#_Toc109478932)

[Hình 3.2 Kết quả của hàm Gaussian so với ảnh gốc và của thư viện open cv 14](#_Toc109478933)

[Hình 3.3 Kết quả bộ lọc Unsharp mask với K=2(ảnh 2), K=1(ảnh 3) so với ảnh gốc 15](file:///C:\Users\Dell\Downloads\Report_XLAS.docx#_Toc109478934)

[Hình 3.4 Kết quả lọc trung vị so với thư viện medianBlur và code tự viết 15](#_Toc109478935)

[Hình 3.5 Kết quả hàm Sobel so với ảnh gốc của thư viện sobel và code tự viết theo phương thẳng đứng 16](#_Toc109478936)

[Hình 3.6 Kết quả hàm Sobel của thư viện sobel và code tự viết theo phương ngang 16](#_Toc109478937)

[Hình 3.7 Kết quả hàm Sobel của thư viện sobel và code tự viết theo cả 2 hướng 16](#_Toc109478938)

[Hình 3.8 Kết quả hàm Laplacian của thư viện laplacian và code tự viết 17](#_Toc109478939)

[Hình 3.9 Kết quả sau khi biến đổi Fourier 18](#_Toc109478940)

[Hình 3.10 Sau khi lọc thông số cao và chuyển về hình ảnh từ miền tần số 18](#_Toc109478941)

[Hình 3.11 Kết quả sau khi đi qua bộ lọc hiệu ứng mùa đông 19](#_Toc109478942)

[Hình 4.1: Thư viện Tkinter 20](#_Toc109478943)

[Hình 4.2 Giao diện của ứng dụng bộ lọc ảnh 21](#_Toc109478944)

# KHÁI QUÁT VỀ XỬ LÝ ẢNH VÀ LỌC ẢNH

## Khái quát về xử lý ảnh

### Xử lý ảnh

Con người thu nhận thông tin qua các giác quan, trong đó thị giác đóng vai trò quan trọng nhất. Những năm trở lại đây với sự phát triển của phần cứng máy tính, xử lý ảnh và đồ hoạ đó phát triển một cách mạnh mẽ và có nhiều ứng dụng trong cuộc sống. Xử lý ảnh và đồ hoạ đóng một vai trò quan trọng trong tương tác người máy.

Quá trình xử lý nhận dạng ảnh là một quá trình thao tác nhằm biến đổi một ảnh đầu vào để cho ra một kết quả mong muốn. Kết quả đầu ra của một quá trình xử lý ảnh có thể là một ảnh “tốt” hơn, hoặc một kết luận.

Ảnh “tốt hơn”

XỬ LÝ ẢNH

Ảnh

Kết luận

Hình 1.1 Quá trình xử lý ảnh

Do đó mục tiêu của xử lý ảnh có thể chia ra làm 3 hướng như sau:

- Xử lý ảnh ban đầu để cho ra một ảnh mới tốt hơn theo một mong muốn của người dung (ví dụ: ảnh nhiễu cần phải lọc nhiễu).

- Phân tích ảnh để thu nhận một thông tin nào đó giúp cho giúp cho việc phân loại và nhận biết ảnh.

- Từ ảnh đầu vào mà có những nhận xét, kết luận ở mức cao hơn, sâu hơn. (ví dụ: ảnh một tai nạn giao thông phác họa hiện trường).

### Các bước cơ bản trong xử lý ảnh

Diagram

Description automatically generated

Hình 1.2 Các bước cơ bản trong xử lý ảnh

Đầu tiên thu nhận ảnh. Để thu nhận ảnh ta cần có các thiết bị nhu nhận ảnh bao gồm camera, scanner các thiết bị thu nhận này có thể cho ảnh đen trắng. Sau đó là Tiền xử lý, ảnh sẽ được cải thiện về độ tương phản, khử nhiễu, khôi phục ảnh, nắn chỉnh hình học… Với mục đích làm cho chất lượng ảnh trở nên tốt hơn nữa, chuẩn bị cho các bước xử lý phức tạp kế tiếp sau đó.

Phân đoạn ảnh là tách một ảnh đầu vào thành các vùng thành phần để biểu diễn phân tích, nhận dạng ảnh. Đầu ra ảnh sau phân đoạn chứa các điểm ảnh của các vùng ảnh (ảnh đã phân đoạn) cộng với mã liên kết với các vùng lân cận.

Ảnh sau khi số hóa sẽ được lưu vào bộ nhớ, hoặc chuyển sang các khâu tiếp theo để phân tích. Nếu lưu trữ ảnh trực tiếp từ các ảnh thô, đòi hỏi dung lượng bộ nhớ cực lớn và không hiệu quả theo quan điểm ứng dụng và công nghệ. Thông thường, các ảnh thô đó biểu diễn lại theo các đặc điểm của ảnh như: biên ảnh, vùng ảnh.

Nhận dạng ảnh là quá trình xác định ảnh. Quá trình này thường thu được bằng cách so sánh với mẫu chuẩn đã được lọc (hoặc lưu) từ trước. Nội suy là phán đoán theo ý nghĩa trên cơ sở nhận dạng.

### Một số khái niệm cơ bản trong xử lý ảnh

#### Điểm ảnh

Điểm ảnh (Pixel) là một phần tử của ảnh số tại tọa độ (x, y) với độ xám hoặc màu nhất định. Kích thước và khoảng cách giữa các điểm ảnh được chọn thích hợp sao cho mắt người cảm nhận về sự liên tục về không gian và mức xám (hoặc màu) của ảnh số gần như thật. Mỗi phần tử trong ma trận được gọi là một phần tử ảnh.

#### Độ phân giải ảnh

Độ phân giải (Resolution) của ảnh là mật độ điểm ảnh được ấn định trên một ảnh số được hiển thị.

Có ba cách để biểu thị độ phân giải của ảnh:

* Biểu thị bằng số lượng điểm ảnh theo chiều dọc và theo chiều ngang của ảnh (ví dụ: 1024×768)
* Biểu thị bằng tổng số điểm ảnh trên một tấm ảnh (ví dụ: 960.000 pixel)
* Biểu thị bằng số lượng điểm ảnh có trên 1 inch (ppi) hoặc số chấm(dot) có trên 1 inch (dpi)

#### Mức xám của ảnh

Mức xám của điểm ảnh là cường độ sáng của nó được gán bằng giá trị số tại điểm đó.

Các thang giá trị mức xám thông thường: 16, 32, 64, 128, 256 (mức 256 là mức phổ dụng). Mức xám dùng 1 byte biểu diễn: 28= 256 mức (tức là từ 0 đến 255).

- **Ảnh đen trắng**: là ảnh có hai màu đen, trắng (không chứa màu khác) với mức xám ở các điểm ảnh có thể khác nhau.

- **Ảnh nhị phân**: Ảnh chỉ có 2 mức đen, trắng phân biệt tức dùng 1 bit mô tả 21 mức khác nhau. Nói cách khác mỗi điểm ảnh của ảnh nhị phân chỉ có thể là 0 hoặc 1.

- **Ảnh màu**: Trong hệ màu RGB (Red, Green, Blue) để tạo nên thế giới màu, người ta thường dùng 3 byte để mô tả mức màu, khi đó các giá trị màu: 28\*3=224≈16,7 triệu màu.

## Lọc ảnh

### Khái niệm

Một hệ thống dùng để làm biến dạng sự phân bố tần số của các thành phần tín hiệu theo các chỉ tiêu gọi đã cho được gọi là bộ lọc số.

Lọc số ảnh (Filter) có ý nghĩa quan trọng trong việc tạo ra các hiệu ứng trong ảnh, 1 số hiệu ứng nhờ sử dụng các bộ lọc làm mờ (Blur), làm trơn (Smooth), ...

### Nguyên tắc chung của lọc ảnh

Nguyên tắc chung của các phương pháp lọc là cho ma trận ảnh nhân với một ma trận lọc (Kernel). Việc nhân ảnh với ma trận lọc giống như việc trượt ma trận lọc theo hàng trên ảnh và nhân với từng vùng của ảnh, cộng các kết quả lại tạo thành kết quả của điểm ảnh trung tâm.

A picture containing shoji, crossword puzzle, building

Description automatically generated

Hình 1.3 Minh hoạ nhân ma trận ảnh

Hình 1.3, ma trận đầu vào I được nhân với ma trận lọc (phần xám ở hình trái) để tạo thành ma trận đầu ra O.

### Các phép toán cơ bản

#### Phép tích chập(convolusion)

Đối với phép lọc ảnh, phép tích chập giữa ma trận lọc (kernel) và ảnh cho ra kết quả ảnh đã được xóa nhiễu. Ta có công thức tích chập giữa hàm ảnh f(x, y) và bộ lọc k(x, y) (kích thước mxn) như sau:

k(x,y) \* f(x,y) = (1.1)

Phép tích chập được hình dung thực hiện bằng việc dịch chuyển ma trận kernel lần lượt qua tất cả các điểm ảnh trong ảnh, bắt đầu từ góc bên trái trên của ảnh. Và đặt điểm neo của kernel tương ứng tại điểm ảnh đang xét. Ở mỗi lần dịch chuyển, thực hiện tính toán kết quả mới cho điểm ảnh đang xét bằng công thức tích chập.

Ta định nghĩa kernel là một ma trận vuông kích thước k\*k trong đó k là số lẻ. k có thể bằng 1, 3, 5, 7, 9, … Ví dụ kernel kích thước 3\*3

A picture containing text, clock

Description automatically generated

Kí hiệu phép tích chập (\*), kí hiệu Y =X\*W

Với mỗi phần tử trong ma trận X lấy ra một ma trận có kích thước bằng kích thước của kernel W có phần tử làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận A. Sau đó tính tổng các phần tử của phép tính của ma trận A và ma trận W, rồi viết vào ma trận kết quả Y.

A picture containing white

Description automatically generated

Hình 1.4 Ví dụ phép tích chập

Với các phần tử ở viền ngoài như , thêm giá trị 0 ở viền ngoài ma trận X và ma trận Y thu được sẽ bằng kích thước ma trận X ban đầu.

Mục đích của phép tính convolution trên ảnh là làm mở, làm nét ảnh; xác định các đường; … Mỗi kernel khác nhau thì sẽ phép tính convolution sẽ có ý nghĩa khác nhau.

#### Phép tương quan chéo(cross-correlation)

Kí hiệu phép tương quan chéo (⊗), G = F ⊗ K. Ta có công thức tính tương quan chéo:

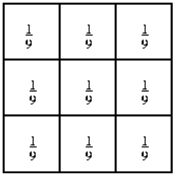
Cả Convolution và cross-correlation đều là bộ lọc tuyến tính. Sự khác biệt duy nhất mà chúng ta thấy được qua công thức trên là: convolution lật ngược kernel.

# TÌM HIỂU MỘT SỐ BỘ LỌC VÀ ỨNG DỤNG

## Bộ lọc trung bình (Mean Filter)

Lọc trung bình là một phương pháp đơn giản, trực quan và dễ thực hiện để làm mịn hình ảnh, tức là giảm mức độ thay đổi cường độ giữa một pixel và pixel tiếp theo. Nó thường được sử dụng để giảm nhiễu trong hình ảnh.

Ý tưởng của lọc trung bình chỉ đơn giản là thay thế mỗi giá trị pixel trong một hình ảnh bằng giá trị trung bình (‘trung bình’) của các hàng xóm của nó, bao gồm cả chính nó. Điều này có tác dụng loại bỏ các giá trị pixel không đại diện cho môi trường xung quanh chúng. Lọc trung bình thường được coi là bộ lọc tích chập. Giống như các phép chập khác, nó dựa trên một hạt nhân, đại diện cho hình dạng và kích thước của vùng lân cận được lấy mẫu khi tính giá trị trung bình. Thường sử dụng nhân hình vuông 3 × 3, mặc dù các nhân lớn hơn (ví dụ hình vuông 5 × 5) có thể được sử dụng để làm mịn nghiêm trọng hơn. (Lưu ý rằng một nhân nhỏ có thể được áp dụng nhiều lần để tạo ra một hiệu ứng tương tự nhưng không giống như một lần truyền đơn với một nhân lớn.



Hình 2.1 Ví dụ kernel của bộ lọc trung bình

## Bộ lọc Gaussian

Bộ lọc Gaussian là bộ lọc có đáp ứng xung là một hàm Gauss (hoặc một giá trị gần đúng với nó, vì một đáp ứng Gaussian thực sự sẽ có đáp ứng xung vô hạn). Bộ lọc Gaussian có các đặc tính là không có đầu vào hàm bước quá mức trong khi giảm thiểu thời gian tăng và giảm. Hành vi này được kết nối chặt chẽ với thực tế là bộ lọc Gaussian có độ trễ nhóm nhỏ nhất có thể. Nó được coi là bộ lọc miền thời gian lý tưởng, cũng như sinc là bộ lọc miền tần số lý tưởng. Những đặc tính này rất quan trọng trong các lĩnh vực như máy hiện sóng và hệ thống viễn thông kỹ thuật số.

A picture containing logo

Description automatically generatedTable

Description automatically generated+, Công thức để tạo ra bộ lọc Gauss:

Hình 2.2 Ví dụ kernel của bộ lọc Gausian

## Bộ lọc Unsharp mask

Bộ lọc Unsharp mask là một kỹ thuật làm sắc nét hình ảnh, lần đầu tiên được thực hiện trong chụp ảnh phòng tối, nhưng hiện nay thường được sử dụng trong phần mềm xử lý ảnh kỹ thuật số. Tên của nó bắt nguồn từ thực tế là kỹ thuật sử dụng hình ảnh âm bản bị mờ, hoặc "không rõ nét", để tạo ra một mặt nạ của hình ảnh gốc. Mặt nạ unsharp sau đó được kết hợp với hình ảnh dương bản gốc, tạo ra hình ảnh ít mờ hơn so với hình ảnh gốc. Hình ảnh thu được, mặc dù rõ ràng hơn, có thể thể hiện chủ thể của hình ảnh kém chính xác hơn.

Một quy trình đã được sử dụng trong nhiều năm bởi ngành in và xuất bản để làm sắc nét hình ảnh bao gồm việc trừ một phiên bản unsharp (được làm mịn) của hình ảnh từ hình ảnh gốc. Quy trình này, được gọi là Masking Unsharp, bao gồm các bước sau:

1. Làm mờ hình ảnh gốc.

2. Trừ hình ảnh mờ khỏi bản gốc (chênh lệch kết quả được gọi là mặt nạ.)

3. Thêm mặt nạ vào bản gốc.

Công thức cộng ảnh:

f(x,y) -

g(x, y) = f(x, y) + k \* (x, y)

## Bộ lọc trung vị (Median Filter)

Lọc trung vị là một kĩ thuật lọc phi tuyến, nó khá hiệu quả đối với hai loại nhiễu: nhiễu đốm (speckle noise) và nhiễu muối tiêu (salt-pepper noise). Chức năng cơ bản của nó là thiết lập giá trị của các điểm với các mức xám khác nhau thành giá trị có vẻ như gần giống với các điểm lân cận.

Ý tưởng chính của thuật toán lọc Trung vị như sau:

B1: Sử dụng một ma trận lọc (ma trận 3x3) quét qua lần lượt từng điểm ảnh của ảnh đầu vào.

B2: Tại vị trí mỗi điểm ảnh lấy giá trị của các điểm ảnh tương ứng trong vùng 3x3 của ảnh gốc “lấp” vào ma trận lọc.

B3: Sau đó sắp xếp các điểm ảnh trong ma trận này theo thứ tự (tăng dần hoặc giảm dần tuỳ ý).

B4: Cuối cùng, gán điểm ảnh nằm chính giữa (trung vị) của dãy giá trị điểm ảnh đã được sắp xếp ở trên cho giá trị điểm ảnh đang xét của ảnh đầu ra.

## Bộ lọc Sobel

Ngoài việc làm mịn ảnh, một số bộ lọc còn có tác dụng tìm biên của ảnh. Đường biên là một loại đặc trưng cục bộ tiêu biểu trong phân tích, nhận dạng ảnh. Đường biên giúp phân đoạn các vùng trong ảnh (xám, màu). Do đó sử dụng các vùng ảnh để tìm đường biên.

Bộ lọc Sobel là bộ lọc tuyến tính được sử dụng trong các thuật toán phát hiện cạnh để tạo ra một hình ảnh nhấn mạnh các cạnh. Sobel ước tính độ lớn và hướng của gradient tại mỗi pixel trong hình ảnh thang độ xám. Các điểm ảnh có độ lớn cao tạo thành một cạnh. Bộ lọc Sobel dựa trên việc biến đổi hình ảnh bằng một bộ lọc nhỏ, có thể phân tách và có giá trị nguyên theo hướng ngang và dọc.

Bộ lọc Sobel sử dụng hai ma trận kernel 3 × 3 tích chập với hình ảnh gốc để tính toán giá trị gần đúng của các đạo hàm - một cho các thay đổi theo chiều ngang và một cho các thay đổi theo chiều dọc. Nếu chúng ta xác định A là hình ảnh nguồn và Gx, Gy là hai hình ảnh mà tại mỗi điểm chứa các xấp xỉ đạo hàm ngang và dọc tương ứng, các phép tính như sau:

= A \* ; = A \*

Tại mỗi điểm trong hình ảnh, các xấp xỉ gradient thu được có thể được kết hợp để cung cấp độ lớn gradient:

Thuật toán Sobel có thể được tóm tắt trong bốn bước:

* Chuyển hình ảnh thành thang độ xám
* Chuyển đổi hình ảnh màu xám với bộ lọc Sobel-x
* Chuyển đổi hình ảnh màu xám với bộ lọc Sobel-y
* Tính toán độ lớn của gradient

## Bộ lọc Laplacian

Để khắc phục hạn chế và nhược điểm của phương pháp Gradient, trong đó sử dụng đạo hàm riêng bậc nhất người ta nghĩ đến việc sử dụng đạo hàm riêng bậc hai hay toán tử Laplace. Phương pháp dò biên theo toán tử Laplace hiệu quả hơn phương pháp toán tử Gradient trong trường hợp mức xám biến đổi chậm, miền chuyển đổi mức xám có độ trải rộng. Laplacian của một hình ảnh làm nổi bật các khu vực thay đổi nhanh chóng về cường độ và do đó có thể được sử dụng để phát hiện cạnh. Toán tử Laplace trong không gian hai chiều được viết như sau:

Toán tử Laplace dùng một số kiểu ma trận lọc khác nhau nhằm tính gần đúng đạo hàm riêng bậc hai. Ma trận lọc sẽ được sử dụng là:

Để thấy rõ việc xấp xỉ đạo hàm riêng bậc 2 trong không gian 2 chiều với ma trận lọc Kernel làm ví dụ, ta có thể tính gần đúng như sau:

Nhược điểm của kỹ thuật này rất nhạy với nhiễu, do vậy đường biên thu được thường kém ổn định. Để khắc phục nhược điểm này người ta mở rộng toán tử Laplace – dùng hàm Gauss (Laplace of Gauss) để giảm nhiễu cho ảnh (làm trơn ảnh).

## Bộ lọc ứng dụng Fourier transform

Ta đã biết, trong ảnh xám, vùng ảnh mà giá trị giữa các Pixel thay đổi đột ngột là vùng ảnh tần số cao, ví dụ như biên hoặc nhiễu. Ngược lại, vùng ảnh mà giá trị các Pixel thay đổi rất ít là vùng ảnh có tần số thấp, ví dụ như vùng nền (Background).

Giả sử, ta có ma trận kết quả biến đổi DFT của 1 ảnh xám, hệ số DC được dịch vào giữa. Do các thành phần tần số thấp ở gần tâm ma trận, ta có thể thực hiện các bộ lọc tần số thấp bằng cách loại bỏ các hệ số ở cách xa tâm.

Ngược lại với lọc thông thấp. Lọc thông cao sẽ loại bỏ các hệ số ở gần tâm và giữa lại các hệ số cách xa tâm. Với kích thước hình tròn càng lớn thì biên của đối tượng trong ảnh càng hiện rõ. Tuy nhiên, bộ lọc này không lọc được nhiễu.

Ta có công thức biến đổi Fourier:

F(u,v) =

Ảnh Fourier có thể được chuyển đổi lại thành miền không gian:

f(u,v) =

## Bộ lọc hiệu ứng mùa đông

Bộ lọc này thường làm trắng các phần tối nhất của hình ảnh, chủ yếu là đường, cây, núi và các đối tượng địa lý phong cảnh khác, ....

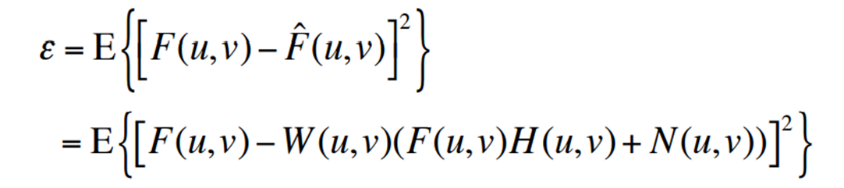
Sử dụng phương pháp tăng “Độ sáng” của HLS.

Kỹ thuật này không hoạt động tốt đối với hình ảnh tối.

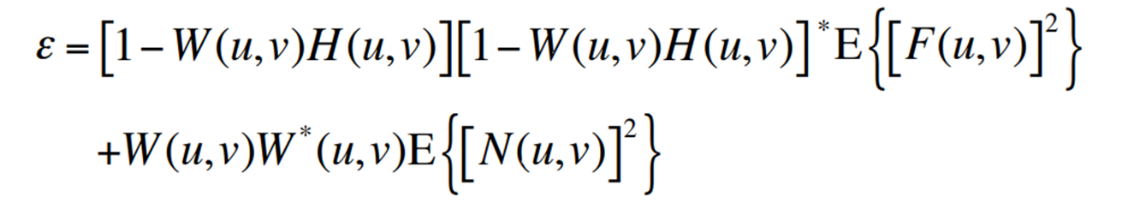
## Bộ lọc Wiener

Bộ lọc wiener là bộ lọc nổi tiếng trong thế giới được Norbert Wiener (ông là nhà toán học nổi tiếng trên cả phương diện toán học thuần túy và toán học ứng dụng, được coi là cha đẻ của ngành Điều khiển học, là người đi tiên phong trong nghiên cứu về quá trình ngẫu nhiên và quá trình nhiễu) đề xuất trong những năm 1940 và được công bố vào năm 1949. [1]

Bộ lọc Wiener dùng để khôi phục ảnh gốc từ các ảnh đã xuống cấp do bị nhiễu phá hủy. Bộ lọc này dùng để làm cự tiểu bình phương sai số giữa ảnh gốc và ảnh đang có. Ảnh sau khi qua một đáp ứng nào đó thì bị nhiễu và khác với ban đầu, theo lí thuyết ta xây dựng một đáp ứng ngược lại để khôi phục ảnh tuy nhiên thực tế có nhiễu nên ta phải sử dụng kĩ thuật lọc Wiener cho các hệ thống có nhiễu. Bộ lọc thường bộ lọc thông thấp, năng lượng của ảnh thường tập trung ở vùng tần số thấp.



Giả thiết nhiễu độc lập với tín hiệu, ta có



## Bộ lọc ứng dụng Fourier transform

Ta đã biết, trong ảnh xám, vùng ảnh mà giá trị giữa các Pixel thay đổi đột ngột là vùng ảnh tần số cao, ví dụ như biên hoặc nhiễu. Ngược lại, vùng ảnh mà giá trị các Pixel thay đổi rất ít là vùng ảnh có tần số thấp, ví dụ như vùng nền (Background).

Giả sử, ta có ma trận kết quả biến đổi DFT của 1 ảnh xám, hệ số DC được dịch vào giữa. Do các thành phần tần số thấp ở gần tâm ma trận, ta có thể thực hiện các bộ lọc tần số thấp bằng cách loại bỏ các hệ số ở cách xa tâm.

Ngược lại với lọc thông thấp. Lọc thông cao sẽ loại bỏ các hệ số ở gần tâm và giữa lại các hệ số cách xa tâm. Với kích thước hình tròn càng lớn thì biên của đối tượng trong ảnh càng hiện rõ. Tuy nhiên, bộ lọc này không lọc được nhiễu.

Ta có công thức biến đổi Fourier:

F(u,v) =

Ảnh Fourier có thể được chuyển đổi lại thành miền không gian:

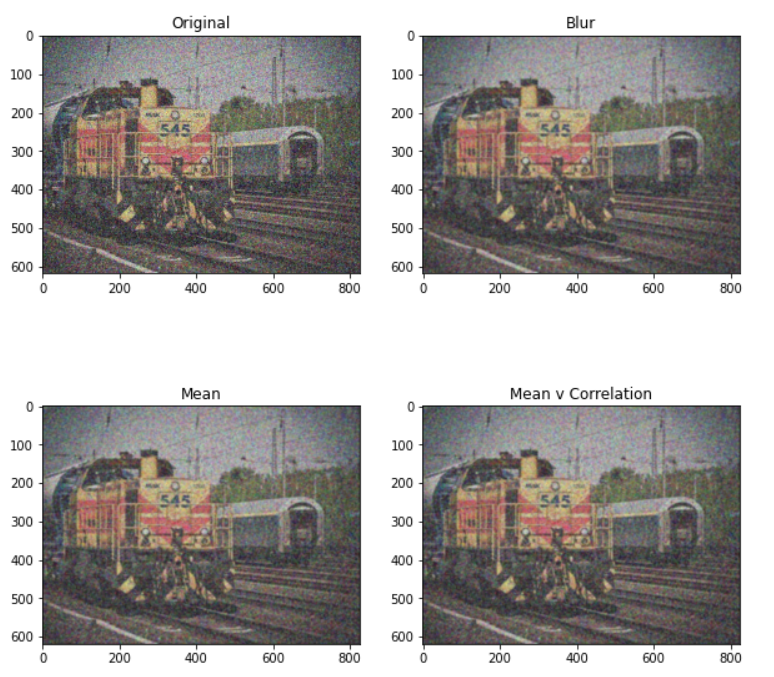
f(u,v) =

# KẾT QUẢ THỰC NGHIỆM

## Lập trình một số bộ lọc làm mịn ảnh

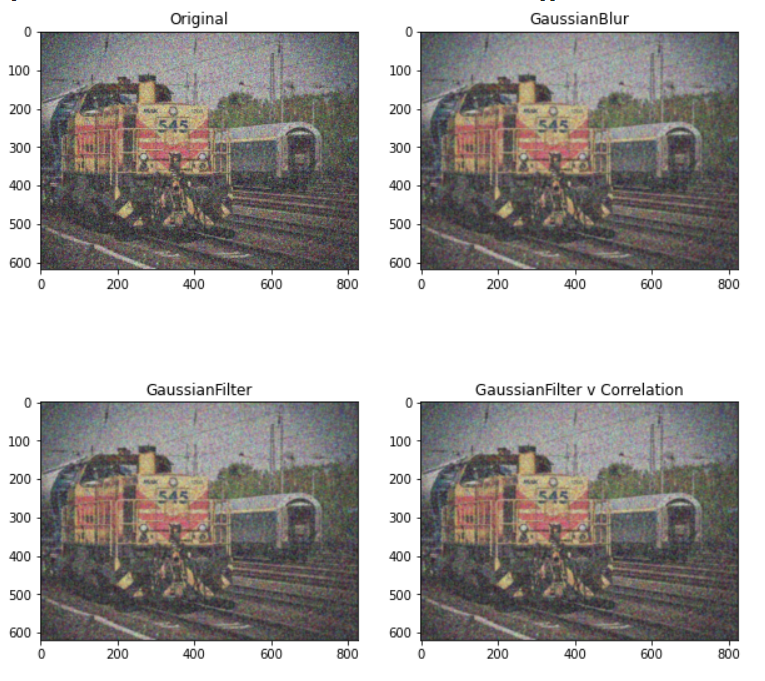
### Bộ lọc trung bình (Mean Filter)

Kết quả bộ lọc trung bình ở ảnh 3 và ảnh 4 khá giống với kết quả của thư viện opencv blur, làm mờ được nhiễu muối tiêu so với ảnh gốc, tuy nhiên ảnh không sắc nét bằng ảnh gốc.



Hình 3.1 Kết quả khi đưa ảnh qua bộ lọc trung bình

### Bộ lọc Gaussian



Hình 3.2 Kết quả của hàm Gaussian so với ảnh gốc và của thư viện open cv

Kết quả bộ lọc trung bình ở ảnh 3 và ảnh 4 khá giống với kết quả của thư viện opencv Gaussian blur, làm mờ ảnh muối tiêu so với ảnh gốc, tuy nhiên ảnh không sắc nét bằng ảnh gốc

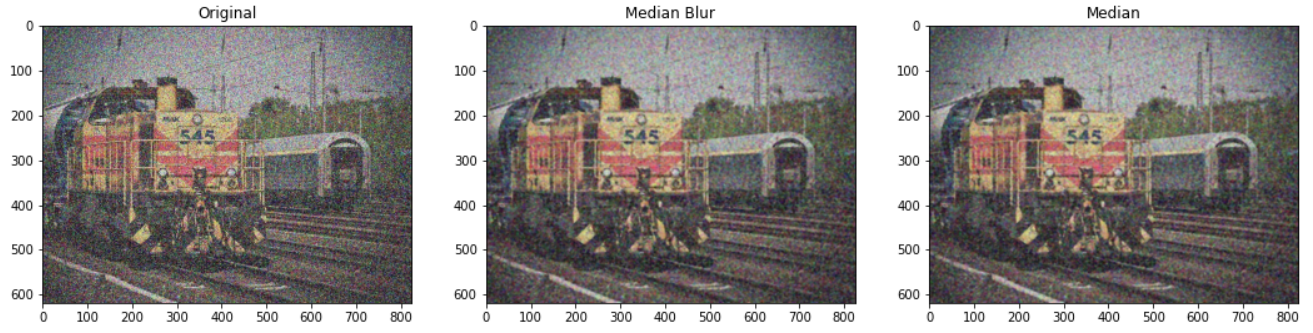
### Chart Description automatically generatedBộ lọc Unsharp mask

Hình 3.3 Kết quả bộ lọc Unsharp mask với K=2(ảnh 2), K=1(ảnh 3) so với ảnh gốc

Kết quả với k=1 và k = 2 đều làm sắc nét ảnh hơn so với ảnh gốc, các đường nét nhìn được rõ hơn. Tuy nhiên k=2 thì làm tốt nhiệm vụ hơn với k = 1 và làm cho các điểm ảnh sắc nét hơn ảnh gốc khá nhiều.

### Bộ lọc trung vị (Median Filter)

Chúng ta sử dụng ảnh đầu vào bị nhiễu muối tiêu:



Hình 3.4 Kết quả lọc trung vị so với thư viện medianBlur và code tự viết

Có thể thấy rõ, với việc lọc trung vị, nhiễu muối tiêu đã được loại bỏ tốt hơn nhiều so với lọc trung bình hay lọc Gauss.

## Lập trình một số bộ lọc tìm biên ảnh

### Bộ lọc Sobel

A collage of a person

Description automatically generated with medium confidence

Hình 3.5 Kết quả hàm Sobel so với ảnh gốc của thư viện sobel và code tự viết theo phương thẳng đứng

Text

Description automatically generated with low confidence

Hình 3.6 Kết quả hàm Sobel của thư viện sobel và code tự viết theo phương ngang

A picture containing text

Description automatically generated

Hình 3.7 Kết quả hàm Sobel của thư viện sobel và code tự viết theo cả 2 hướng

Bộ lọc Sobel chỉ xét với kernel = 1, đối với xét theo cả 2 hướng đường biên của ảnh rất rõ nét.

### Bộ lọc Laplacian

A person looking at the camera

Description automatically generated with low confidence

A picture containing text, tree

Description automatically generated

Hình 3.8 Kết quả hàm Laplacian của thư viện laplacian và code tự viết

Kết quả trên cho thấy qua sử dụng hàm laplacian cho ảnh tìm ra đường biên của ảnh, độ rõ nét khác nhau đối với từng ma trận kernel, trong đó rõ nhất là kernel bằng K2:

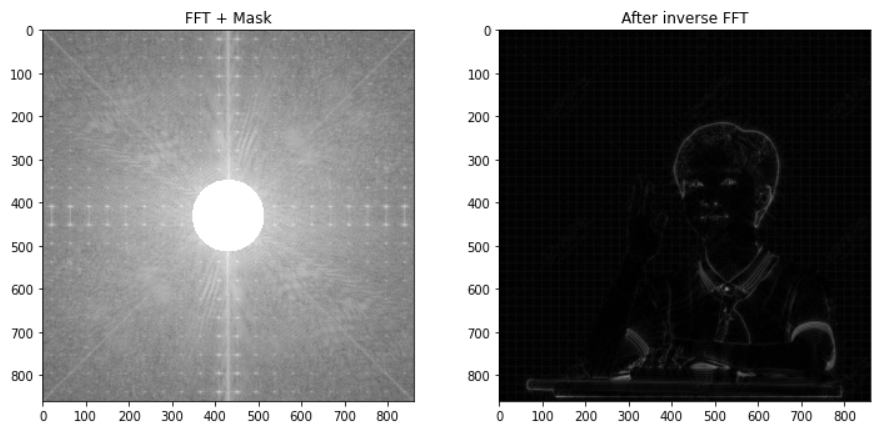
## Một số bộ lọc khác

### Bộ lọc ảnh ứng dụng DFT



Hình 3.9 Kết quả sau khi biến đổi Fourier

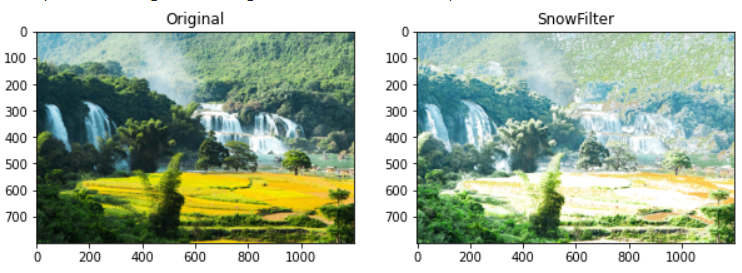
Ta đã biết, trong ảnh xám, vùng ảnh mà giá trị giữa các Pixel thay đổi đột ngột là vùng ảnh tần số cao, ví dụ như biên hoặc nhiễu. Ngược lại, vùng ảnh mà giá trị các Pixel thay đổi rất ít là vùng ảnh có tần số thấp, ví dụ như vùng nền (Background).



Hình 3.10 Sau khi lọc thông số cao và chuyển về hình ảnh từ miền tần số

Ở đây ta lọc thông cao lấy ra các viền lúc mà thay đổi cường độ mức sáng đột ngột (viền). Các thành phần tần số thấp ở gần tâm ma trận còn tần số cao ở xa tâm. Vì vậy kích thước hình tròn càng lớn thì biên của đối tượng trong ảnh càng hiện rõ.

### Bộ lọc hiệu ứng mùa đông



Hình 3.11 Kết quả sau khi đi qua bộ lọc hiệu ứng mùa đông

Bộ lọc làm thay đổi hiệu ứng trong ảnh thành hiệu ứng mùa đông

# XÂY DỰNG ỨNG DỤNG ÁP DỤNG CÁC BỘ LỌC

## Chuẩn bị

### Tổng quan về lập trình giao diện với Python

Thế giới thay đổi, thứ từng quan trọng của ngày hôm qua thì hôm nay chưa chắc đã cần tới. Thời đại tất cả mọi thứ đều chuyển lên web, thì web/app mobile trở thành giao diện để tương tác với người dùng, chứ không phải các phần mềm có giao diện chạy trên máy tính như trước kia. Giờ đây người ta: nghe nhạc trên web, xem film trên web, chơi game trên web, soạn thảo văn bản trên web… khó còn ứng dụng nào không đưa lên web nữa. Vậy nên về mặt “sự nghiệp”, có vẻ như bạn nên đầu tư vào kỹ năng làm web thay vì học để tạo một app trên desktop như cách đây chục năm.

Dĩ nhiên, GUI không ngay lập tức biến mất, vẫn có nhu cầu sử dụng, vẫn có người dùng, vẫn có hàng tá thư viện đồ hoạ tồn tại từ lâu (và vẫn tiếp tục phát triển), vẫn có những game mà chỉ chơi được trên máy tính do yêu cầu về hiệu năng mà web không đáp ứng nổi (như Half-Life/ đế chế / đua xe …).

Python hỗ trợ không ít các thư viện làm GUI app, có thể kể tới: Qt, WxWidgets, Tkinter, Kivy (làm cả app mobile), …

Qt là nền tảng phát triển ứng dụng dùng trong công nghiệp, hỗ trợ mọi hệ điều hành phổ biến, và rất “xịn”. Nếu có nhu cầu làm ứng dụng desktop với Python, hãy đầu tư vào Qt để có một sản phẩm đẳng cấp, không kém bất kỳ nền tảng nào khác.

Tk là hệ thống thư viện đồ hoạ đơn giản, dễ dùng, chạy trên cả 3 hệ điều hành phổ biến: Windows, Ubuntu, OSX/MacOS và điều quan trọng nhất: thư viện tkinter đi kèm mọi bộ cài Python, nên muốn dùng không cần phải cài đặt gì thêm.

### Tổng quan về thư viện Tkinter



Hình 4.1: Thư viện Tkinter

Tkinter là thư viện GUI tiêu chuẩn cho Python. Python khi kết hợp với Tkinter cung cấp một cách nhanh chóng và dễ dàng để tạo các ứng dụng GUI. Tkinter cung cấp giao diện hướng đối tượng mạnh mẽ cho bộ công cụ GUI Tk.

Tạo một ứng dụng GUI bằng Tkinter là một nhiệm vụ dễ dàng. Tất cả bạn cần làm là thực hiện các bước sau:

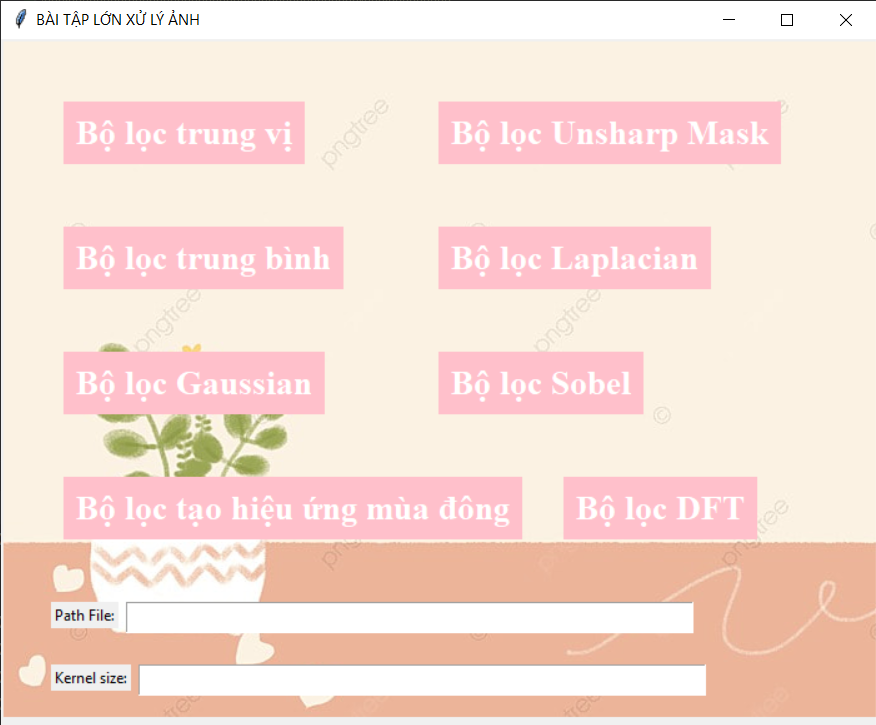
1. Nhập mô-đun Tkinter.
2. Tạo cửa sổ chính của ứng dụng GUI.
3. Thêm một hoặc nhiều tiện ích được đề cập ở trên vào ứng dụng GUI.
4. Nhập vòng lặp sự kiện chính để thực hiện hành động đối với từng sự kiện được kích hoạt bởi người dùng.

## Phát triển ứng dụng

Ứng dụng bộ lọc ảnh được xây dựng bằng công cụ Android studio với sự hỗ trợ của thư viện Tkinter cung cấp các hàm tạo giao diện GUI.

Xây dựng các action cho các chức năng lấy ảnh từ thiết bị, hiên thị hình ảnh, các button chuyển đổi các bộ lọc, reset bộ lọc, lưu ảnh và giao diện cho ứng dụng

Ứng dụng bộ lọc ảnh chỉ gồm duy nhất một giao diện đầy đủ các chức năng chính: Lấy ảnh từ thiết bị, hiên thị hình ảnh, các button chuyển đổi các bộ lọc, lưu ảnh



Hình 4.2 Giao diện của ứng dụng bộ lọc ảnh

# KẾT LUẬN

Trong quá trình tìm hiểu tài liệu và thực hiện bài tập lớn dưới sự định hướng của cô hướng dẫn chúng em đã đạt được một số kết quả như sau:

* Tìm hiểu được một cách tổng quan các vấn đề về bộ lọc. Phép tương quan và tích chập đều được sử dụng trong bộ lọc tuyến tính. Thông thường, chúng cho kết quả khác nhau. Tuy nhiên với những kernel đặc biệt, chúng lại cho kết quả tương đồng, nên với các bộ lọc sử dụng kernel đó thì ảnh đầu ra với hai phép toán này hoàn toàn giống nhau.
* Có rất nhiều bộ lọc ảnh với nhiều mục đích sử dụng khác nhau: làm mịn, làm mờ, lọc các loại nhiễu, làm nổi biên, sắc nét,…
* Bằng cách hiểu bản chất của từng bộ lọc, em chạy thử nghiệm đối với một số bộ lọc.
* Xây dựng một ứng dụng áp dụng các bộ lọc ảnh dựa theo các bộ lọc đã trình bày.

Ngoài ra, trong quá trình tìm hiểu chúng em cũng tự tích lũy thêm cho mình các kiến thức về toán học, về kỹ thuật lập trình, … Tuy mới chỉ là bước đầu, nhưng những kết quả này sẽ giúp ích cho chúng em trong những tìm hiểu sau này để thu được những kết quả tốt hơn. Chúng em xin chân thành cảm ơn cô Lê Thị Lan đã hướng dẫn chúng em hoàn thành bài tập lớn.

# TÀI LIỆU THAM KHẢO

1. *https://github.com/ultralytics/yolov5*, last accessed at 14:57 on June 24, 2021.
2. Qing Chen, El-Sawah A., Joslin C., et al. (2005). *A dynamic gesture interface for virtualenvironments based on hidden markov models*. IEEE, 109–114, 109–114.
3. Chen X. and Koskela M. (2013). *Online RGB-D gesture recognition with extreme learning machines*. ACM Press, 467–474, 467–474.
4. Doan H.-G., Vu H., and Tran T.-H. (2017). *Dynamic hand gesture recognition from cyclical hand pattern*. IEEE, 97–100, 97–100.
5. Burges C.J.C. (1998). *A Tutorial on Support Vector Machines for Pattern Recognition. Data Min Knowl Discov*, 2(2), 121–167.
6. Gkioxari G., Girshick R., and Malik J. (2015). *Contextual Action Recognition with R\*CNN*. IEEE, 1080–1088, 1080–1088.
7. Cheron G., Laptev I., and Schmid C. (2015). *P-CNN: Pose-Based CNN Features for Action Recognition*. IEEE, 3218–3226, 3218–3226.
8. Simonyan K. and Zisserman A. *Two-Stream Convolutional Networks for Action Recognition in Videos.* 9.
9. Ji S., Xu W., Yang M., et al. (2013). 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Trans Pattern Anal Mach Intell*, 35(1), 221–231.

[10] Bishop C.M. (2006), *Pattern recognition and machine learning*, Springer, New York.

[11] Becker S. and Lecun Y. (1989). Improving the convergence of back-propagation learning with second-order methods. Proc 1988 Connect Models Summer Sch San

Mateo.

[12] Brox T., Bruhn A., Papenberg N., et al. (2004). High Accuracy Optical Flow Estimation Based on a Theory for Warping. *Computer Vision - ECCV 2004*. Springer Berlin Heidelberg, Berlin, Heidelberg, 25–36.

[13] *https://github.com/AlexeyAB/darknet,* last accessed at 22:35 on June 17, 2021.

[14]*https://www.kaggle.com/phamdinhkhanh/convolutional-neural-network-p1,* last accessed at 15:31 on June 14, 2021.

[15] *https://blog.roboflow.com/yolov5-improvements-and-evaluation/,* last accessed at 17:31 on June 18, 2021.