BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC ĐIỆN LỰC
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO KẾT THÚC HỌC PHẦN

NHẬP MÔN XỬ LÝ ẢNH

VẼ VÀ TÔ MÀU HÌNH HỌC

Giảng viên hướng dẫn	**:Đào Nam Anh**
Sinh viên thực hiện	**:Nguyễn Tiến Thanh**
	Phạm Ngọc Huyên
Ngành	**:Công nghệ thông tin**
Chuyên ngành	**:Công nghệ phần mềm**
Lớp	**:D17CNPM6**
Khóa	**:2022 - 2027**

Hà Nội, tháng 11 năm 2025

LỜI CẢM ƠN

Trong bài báo cáo chuyên đề này, chúng em xin gửi lời cảm ơn sâu sắc và lòng biết ơn chân thành đến những người đã đóng góp và hỗ trợ chúng em trong quá trình nghiên cứu và viết bài báo cáo này.

Đầu tiên, chúng em muốn bày tỏ lòng biết ơn đến giảng viên hướng dẫn của chúng em, thầy Đào Nam Anh. Sự tận tâm và kiến thức sâu rộng của thầy đã giúp định hình và chỉ dẫn cho chúng em trong quá trình nghiên cứu. Những ý kiến xây dựng và hướng dẫn của thầy đã cung cấp cho chúng em một cơ sở vững chắc để phát triển ý tưởng và triển khai nghiên cứu của mình.

Chúng em cũng muốn gửi lời cảm ơn đến các thành viên trong ban giảng dạy và các chuyên gia đã dành thời gian để đọc và đánh giá bài báo cáo chuyên đề của chúng em. Sự đánh giá chuyên môn và phản hồi xây dựng từ thầy cô đã giúp chúng em nắm bắt những khía cạnh quan trọng và cải thiện chất lượng của bài viết.

Một lần nữa, chúng em xin chân thành cảm ơn tất cả những người đã đóng góp và hỗ trợ chúng em trong quá trình nghiên cứu và viết bài báo cáo chuyên đề này. Sự giúp đỡ của các bạn đã làm cho dự án này trở thành hiện thực và mang lại những kết quả mà chúng em tự hào.

Chúng em xin chân thành cảm ơn!

PHIẾU CHẤM ĐIỂM

Danh sách sinh viên tham gia:

Họ và Tên	Nội dung thực hiện	Điểm	Chữ ký	Ghi chú
Nguyễn Tiến Thanh 22810310382	Vẽ hình, tạo logo			
Phạm Ngọc Huyên 22810310376	Vẽ hình , tạo tranh			

Danh sách giảng viên chấm thi:

Họ và Tên	Chữ ký	Ghi chú

MỤC LỤC

Trang

DANH MỤC HÌNH ẢNH, BẢNG BIỂU

Trang

PHẦN MỞ ĐẦU

Trong kỷ nguyên số, xử lý ảnh đóng vai trò nền tảng trong nhiều lĩnh vực công nghệ từ thị giác máy tính, nhận dạng đối tượng cho đến thiết kế đồ họa. OpenCV (Open Source Computer Vision Library) là một thư viện mạnh mẽ, cung cấp bộ công cụ toàn diện để thao tác và xử lý hình ảnh, giúp người học dễ dàng tiếp cận và thực hành các kỹ thuật cơ bản. Đề tài "Vẽ và Tô Màu Hình Học Cơ Bản" được lựa chọn nhằm mục đích làm quen và nắm vững các hàm cơ bản trong OpenCV để vẽ (Draw) và tô màu (Color) các hình học cơ bản như đường thẳng, hình chữ nhật, hình tròn, elip, và đa giác. Đề tài này không chỉ là bài thực hành kỹ thuật, mà còn là bước khởi đầu quan trọng để hiểu rõ cách OpenCV thao tác trên ma trận ảnh, từ đó tạo nền tảng vững chắc cho việc tiếp cận các thuật toán xử lý ảnh nâng cao hơn.

Báo cáo này sẽ tập trung vào việc thực hành và củng cố kiến thức về hệ màu BGR mà OpenCV sử dụng, cũng như cách thức thao tác với các kênh màu để tạo ra các hiệu ứng tô màu và gradient. Nội dung chính sẽ trình bày chi tiết quá trình sử dụng các hàm vẽ hình cơ bản, thêm text với cv2.putText(), và ứng dụng sáng tạo các kỹ thuật đã học để xây dựng các hình ảnh phức tạp. Cụ thể, sản phẩm trực quan được tạo ra sẽ bao gồm tranh phong cảnh đơn giản, và các thiết kế logo/biểu tượng chuyên nghiệp, có kèm theo hiệu ứng viền cho text.

CHƯƠNG 1: TỔNG QUAN VỀ XỬ LÝ ẢNH

1.1 Định nghĩa và mục tiêu của xử lý ảnh

1.1.1 Định nghĩa

Xử lý Ảnh (Image Processing) là một lĩnh vực thuộc khoa học máy tính và kỹ thuật, chuyên thực hiện các phép toán và thuật toán trên hình ảnh kỹ thuật số (Digital Image). Mục đích chính là biến đổi, cải thiện chất lượng ảnh hoặc trích xuất thông tin có ý nghĩa từ ảnh.

Quá trình xử lý ảnh bao gồm 3 giai đoạn chính:

- **Nhập ảnh (Image Acquisition)**: Chuyển đổi dữ liệu quang học (analog) thành định dạng số (digital).

- **Thao tác/Biến đổi (Manipulation)**: Áp dụng các thuật toán lên ma trận ảnh.

- **Xuất ảnh/Kết quả (Output)**: Hiển thị ảnh đã xử lý hoặc thông tin trích xuất.

1.1.2 Phân loại và Phạm vi

Xử lý ảnh thường được phân loại theo mục tiêu chính của nó.

Bảng 1.1: Phân loại xử lý ảnh

Phân loại	Mô tả
Xử lý Ảnh Mức thấp (Low-Level Processing)	Các thao tác đầu vào và đầu ra đều là ảnh. Ví dụ: khử nhiễu (noise removal), tăng cường độ tương phản (contrast enhancement)
Xử lý Ảnh Mức trung bình (Mid-Level Processing)	Đầu vào là ảnh, đầu ra là các thuộc tính (attributes) trích xuất được từ ảnh. Ví dụ: phân đoạn ảnh (image segmentation), trích xuất đường biên (edge detection).
Xử lý Ảnh Mức cao (High-Level Processing / Computer Vision)	Chuyên dùng để "hiểu" và diễn giải nội dung của ảnh. Ví dụ: nhận dạng đối tượng (object recognition), mô tả cảnh (scene description).

1.2 Nền tảng Kỹ thuật và Biểu diễn Ảnh

1.2.1 Cấu trúc Ảnh Số (Digital Image Structure)

Ảnh số được biểu diễn dưới dạng ma trận M x N.

- **Pixel (Phần tử Ảnh)**: Mỗi phần tử (i, j) trong ma trận là một pixel, đại diện cho độ sáng hoặc màu sắc tại tọa độ đó.

- **Độ sâu Bit (Bit Depth)**: Xác định số lượng giá trị cường độ sáng hoặc màu sắc có thể có của một pixel.

+ Ảnh Grayscale 8-bit: Có $2^8 = 256$ mức độ xám (từ 0 đến 255).

+ Ảnh màu 24-bit: Mỗi kênh màu sử dụng 8 bit, tổng cộng 2^8 x 2^8 x 2^8 ≈ 16.7 triệu màu.

1.2.2 Hệ màu (Color Models)

Hệ màu là phương pháp toán học để mô tả các màu sắc.

- **Hệ màu RGB (Red, Green, Blue)**: Là hệ màu cộng (Additive Color Model). Ba kênh màu được kết hợp để tạo ra màu sắc cuối cùng. Ví dụ: trong Python, giá trị pixel được biểu diễn dưới dạng tuple (R, G, B).

- **Hệ màu BGR (Blue, Green, Red) trong OpenCV**:

+ Quy ước của OpenCV: Trong thư viện OpenCV, ảnh màu được lưu trữ dưới dạng ma trận 3 kênh, nhưng theo thứ tự BGR (Blue, Green, Red), không phải RGB tiêu chuẩn.

+ Lý do: Quy ước này bắt nguồn từ lịch sử phát triển của OpenCV và cách sắp xếp dữ liệu trong các định dạng ảnh phổ biến.

+ Thao tác bắt buộc: Khi hiển thị ảnh BGR bằng các thư viện khác (như Matplotlib trong Notebook), cần sử dụng hàm chuyển đổi không gian màu (cv2.cvtColor(ảnh, cv2.COLOR_BGR2RGB)) để đảm bảo màu sắc được hiển thị chính xác.

1.3 Thư viện OpenCV: Nền tảng Vẽ và Thao tác Ảnh

1.3.1 OpenCV và NumPy

Trong môi trường lập trình Python, OpenCV thường sử dụng NumPy arrays để biểu diễn ma trận ảnh. Điều này cho phép tận dụng các phép toán ma trận hiệu suất cao của NumPy cho các tác vụ xử lý ảnh.

1.3.2 Chức năng Vẽ Hình học (Drawing Functions)

Các hàm vẽ hình (cv2.line, cv2.rectangle, cv2.circle, v.v.) là các công cụ thao tác trực tiếp lên ma trận pixel để:

- **Tạo ảnh tổng hợp/Sơ đồ**: Xây dựng các hình ảnh đồ họa cơ bản từ đầu.

- **Trực quan hóa kết quả thuật toán**: Ví dụ: vẽ các đường thẳng biểu thị các tính năng (feature) đã trích xuất.

- **Thông số kỹ thuật**: Các hàm này yêu cầu các thông số cơ bản như Tọa độ (điểm bắt đầu/kết thúc, tâm), Màu sắc (dưới dạng tuple BGR), và Độ dày (hoặc giá trị -1 để tô đầy hình).

1.3.3 Hàm Thao tác Văn bản

- **cv2.putText()**: Hàm quan trọng để thêm văn bản vào ảnh.

- **Tham số chính**: Bao gồm font chữ (OpenCV cung cấp nhiều font như FONT_HERSHEY_SIMPLEX), kích thước, màu sắc (BGR), và độ dày nét. Kỹ thuật vẽ viền (outline) được thực hiện bằng cách vẽ cùng một văn bản hai lần với độ dày và màu sắc khác nhau.

CHƯƠNG 2: THỰC HÀNH VẼ VÀ TÔ MÀU HÌNH HỌC VỚI OPENCV

2.1 Thiết lập môi trường và nền tảng cơ bản

2.1.1 Import thư viện và chuẩn bị

Các thư viện chính được nhập khẩu để hỗ trợ xử lý ảnh và hiển thị kết quả:

- **cv2 (OpenCV)**: Thư viện chính chứa các hàm xử lý ảnh và vẽ hình.
- **numpy**: Được sử dụng để tạo và thao tác trên ma trận ảnh kỹ thuật số.
- **matplotlib.pyplot**: Dùng để hiển thị ảnh trong môi trường Notebook.

```
# Import các thư viện cần thiết
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Thiết lập để hiển thị ảnh trong notebook
%matplotlib inline

# Hàm hỗ trợ hiển thị ảnh
def hien_thi_anh(anh, tieu_de="Ảnh", figsize=(10, 8)):
    """
    Hàm hiển thị ảnh với matplotlib
    OpenCV sử dụng BGR, cần chuyển sang RGB để hiển thị đúng màu
    """
    plt.figure(figsize=figsize)
    # Chuyển từ BGR sang RGB
    anh_rgb = cv2.cvtColor(anh, cv2.COLOR_BGR2RGB)
    plt.imshow(anh_rgb)
    plt.title(tieu_de, fontsize=16)
    plt.axis('off')
    plt.show()

print("Đã import thành công!")
print(f"OpenCV version: {cv2.__version__}")
```

```
Đã import thành công!
OpenCV version: 4.12.0
```

Hình 2.1: Import thư viện

2.1.2 Khởi tạo Canvas và Hệ màu

Trước khi tiến hành vẽ, việc khởi tạo một không gian làm việc (canvas) và định nghĩa bảng màu là bước thiết yếu.

- **Khởi tạo Canvas**: Một ma trận NumPy được tạo ra để mô phỏng một bức tranh trắng kích thước 800 x 600 pixel, với 3 kênh màu (BGR) và kiểu dữ liệu np.uint8 (8-bit, giá trị từ 0 đến 255).

- **Định nghĩa Bảng màu BGR**: Do OpenCV mặc định sử dụng hệ màu BGR (Blue, Green, Red), các màu cơ bản được định nghĩa theo thứ tự này.

```
# Tạo canvas trắng (800x600 pixels)
chieu_cao, chieu_rong = 600, 800
canvas = np.ones((chieu_cao, chieu_rong, 3), dtype=np.uint8) * 255

# Định nghĩa bảng màu
MAU_DO = (0, 0, 255)       # Đỏ
MAU_XANH_LA = (0, 255, 0)  # Xanh lá
MAU_XANH_DUONG = (255, 0, 0)  # Xanh dương
MAU_VANG = (0, 255, 255)   # Vàng
MAU_CAM = (0, 165, 255)    # Cam
MAU_TIM = (255, 0, 255)    # Tím
MAU_HONG = (203, 192, 255)  # Hồng
MAU_DEN = (0, 0, 0)        # Đen

print("Đã khởi tạo canvas và bảng màu!")
```

```
Đã khởi tạo canvas và bảng màu!
```

Hình 2.2: Khởi tạo Canvas và Hệ màu

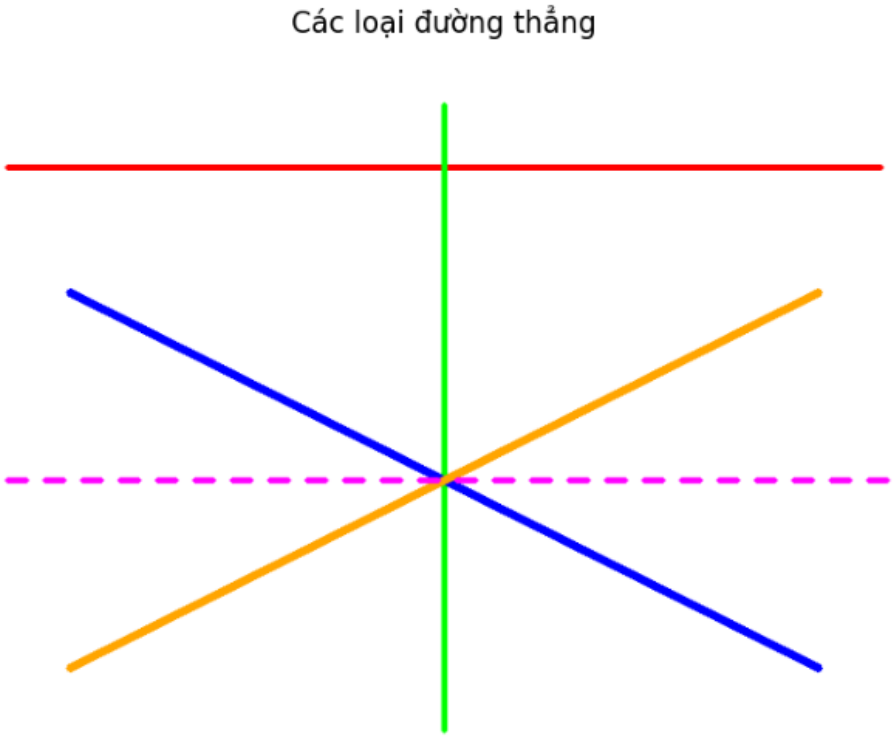
2.2 Vẽ các hình học cơ bản

2.2.1 Vẽ đường thẳng (cv2.line)

Hàm cv2.line() là công cụ cơ bản để tạo ra các đoạn thẳng, được định nghĩa bởi tọa độ của điểm đầu và điểm cuối.

- **Cú pháp**: cv2.line(ảnh, điểm_đầu, điểm_cuối, màu, độ_dày)

- **Kỹ thuật Đứt nét**: Hiệu ứng đường đứt nét được mô phỏng bằng cách lặp lại việc vẽ các đoạn thẳng ngắn, ngắt quãng, chứng minh khả năng kiểm soát từng pixel của OpenCV.



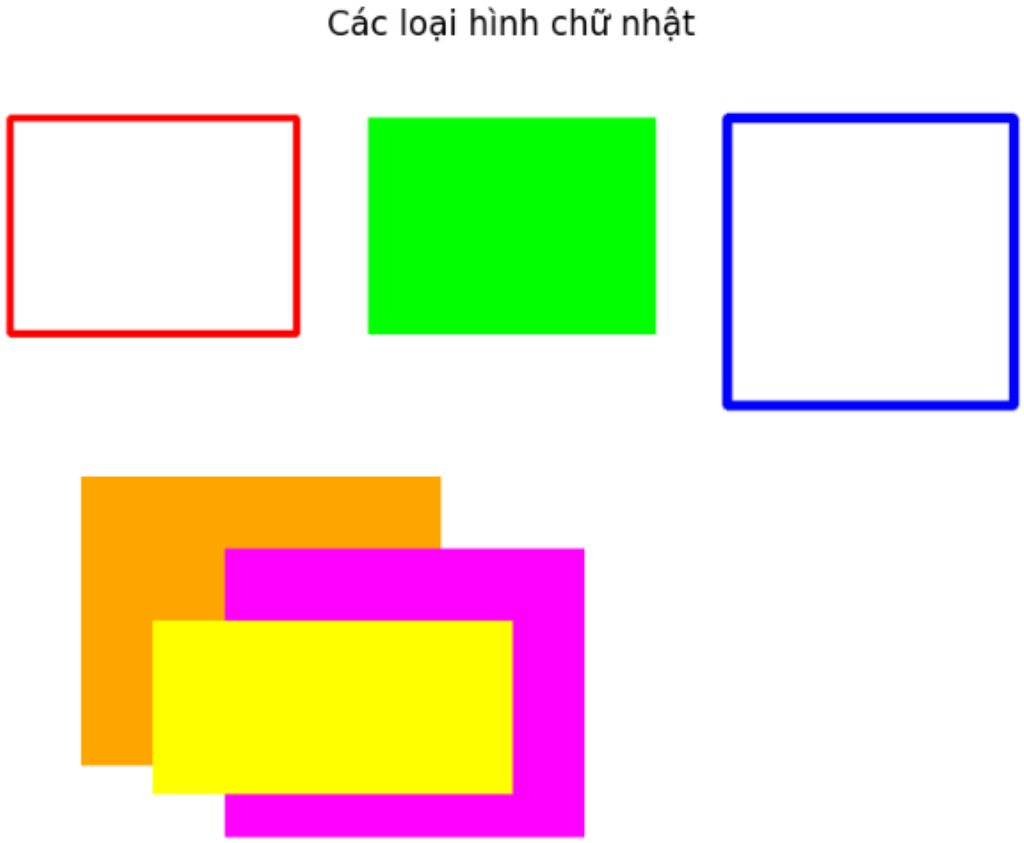
Hình 2.3: Các loại đường thẳng

2.2.2 Vẽ Hình chữ nhật (cv2.rectangle)

Hình chữ nhật được định nghĩa thông qua hai điểm tọa độ: góc trên trái và góc dưới phải.

- **Cú pháp**: cv2.rectangle(ảnh, (x_trên_trái, y_trên_trái), (x_dưới_phải, y_dưới_phải), màu, độ_dày)

- **Tô đầy (Fill)**: Khả năng tô đầy hình (bằng cách đặt độ_dày = -1) được khai thác để tạo ra các khối màu cơ bản, cần thiết cho việc kết hợp hình sau này.



Hình 2.4: Các hình chữ nhật

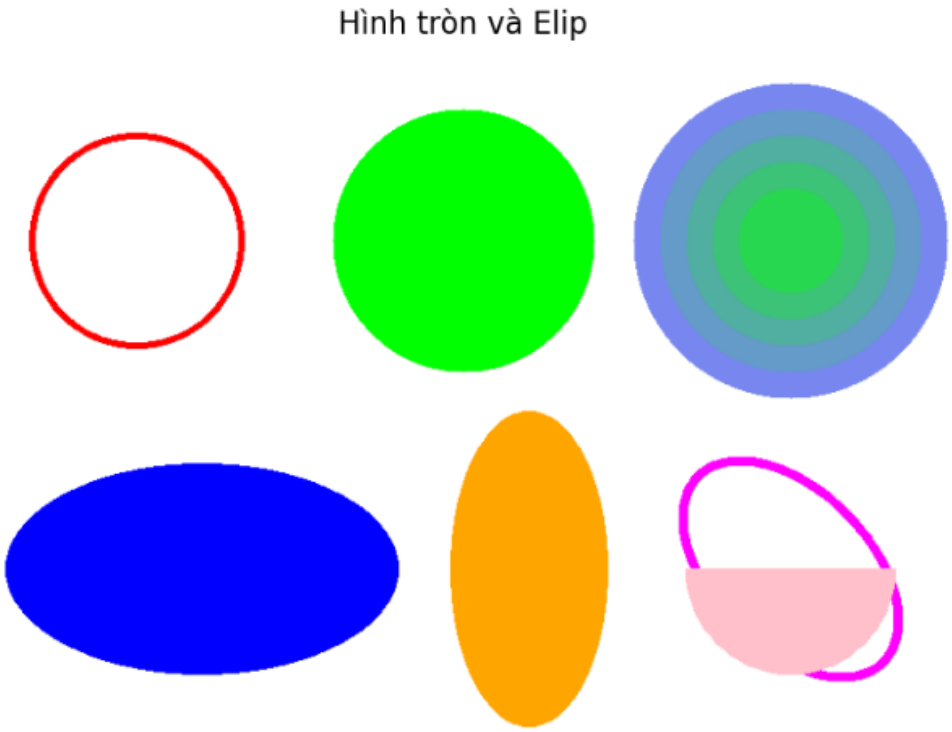
2.2.3 Vẽ Hình tròn và Hình Elip (cv2.circle và cv2.ellipse)

Các hàm vẽ hình tròn và elip cho phép tạo ra các hình cong, đóng góp vào sự đa dạng của các yếu tố đồ họa.

- **Hình tròn (cv2.circle)**: Định nghĩa bằng tâm và bán kính.

- **Hình Elip (cv2.ellipse)**: Là hàm vẽ hình cong phức tạp hơn, được định nghĩa bằng: tâm, chiều dài trục (trục dài, trục ngắn), góc xoay, và góc bắt đầu/kết thúc.

Tham số góc bắt đầu/kết thúc cho phép chỉ vẽ cung tròn (một phần của elip), ví dụ: chỉ vẽ nửa elip (từ 0 đến 180 độ).



Hình 2.5: Hình tròn và Elip

2.2.4 Vẽ Đa giác (cv2.polylines và cv2.fillPoly)

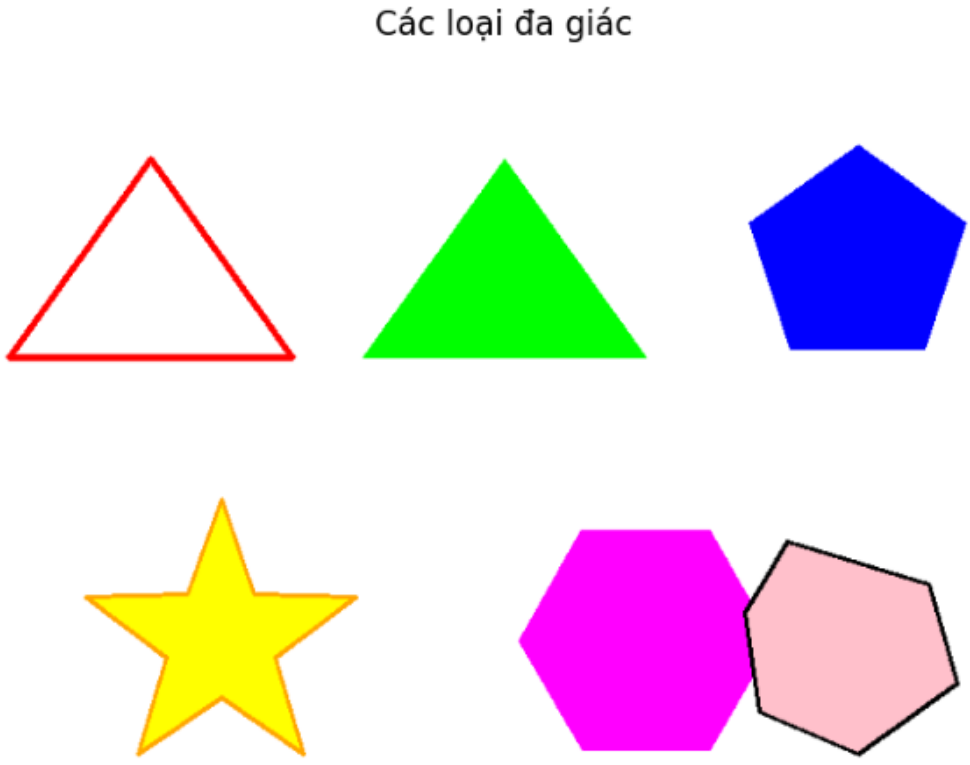
Để vẽ các hình có số cạnh tùy ý (không phải hình chữ nhật hay hình tròn), ta sử dụng kỹ thuật đa giác.

- **Định nghĩa Đỉnh**: Các đỉnh của đa giác được lưu trữ trong một mảng NumPy (np.array).

- **Vẽ viền (cv2.polylines)**: Kết nối các đỉnh đã cho để tạo viền hình.

- **Tô đầy (cv2.fillPoly)**: Lấp đầy vùng không gian bên trong đa giác.

- **Tạo Ngôi sao và Lục giác**: Các hình phức tạp như ngôi sao 5 cánh được tạo ra bằng cách tính toán tọa độ các đỉnh một cách toán học (dựa trên góc) và sau đó áp dụng cv2.fillPoly().



Hình 2.6: Các loại đa giác

CHƯƠNG 3: ỨNG DỤNG THIẾT KẾ

3.1 Kỹ thuật tô màu và chồng lớp hình học

Việc kết hợp nhiều hình học với các khối màu khác nhau là chìa khóa để tạo ra các vật thể có chiều sâu và phức tạp hơn hình dạng ban đầu.

3.1.1 Tô màu kỹ thuật chuyển màu (Gradient)

Kỹ thuật tạo hiệu ứng chuyển màu (gradient) được phân tích chi tiết trong sản phẩm tranh phong cảnh:

- **Nguyên lý**: Thay vì chỉ vẽ các khối màu solid, một vòng lặp được sử dụng để vẽ hàng loạt đường thẳng ngang (cv2.line) với độ dày là 1 pixel, từ tọa độ y = 0 đến y = 400 (vùng trời).

- **Công thức Màu**: Giá trị của các kênh màu BGR được tính toán dựa trên chỉ số hàng hiện tại (i):

mau_troi = (255 - i//2, 200 - i//3, 100)

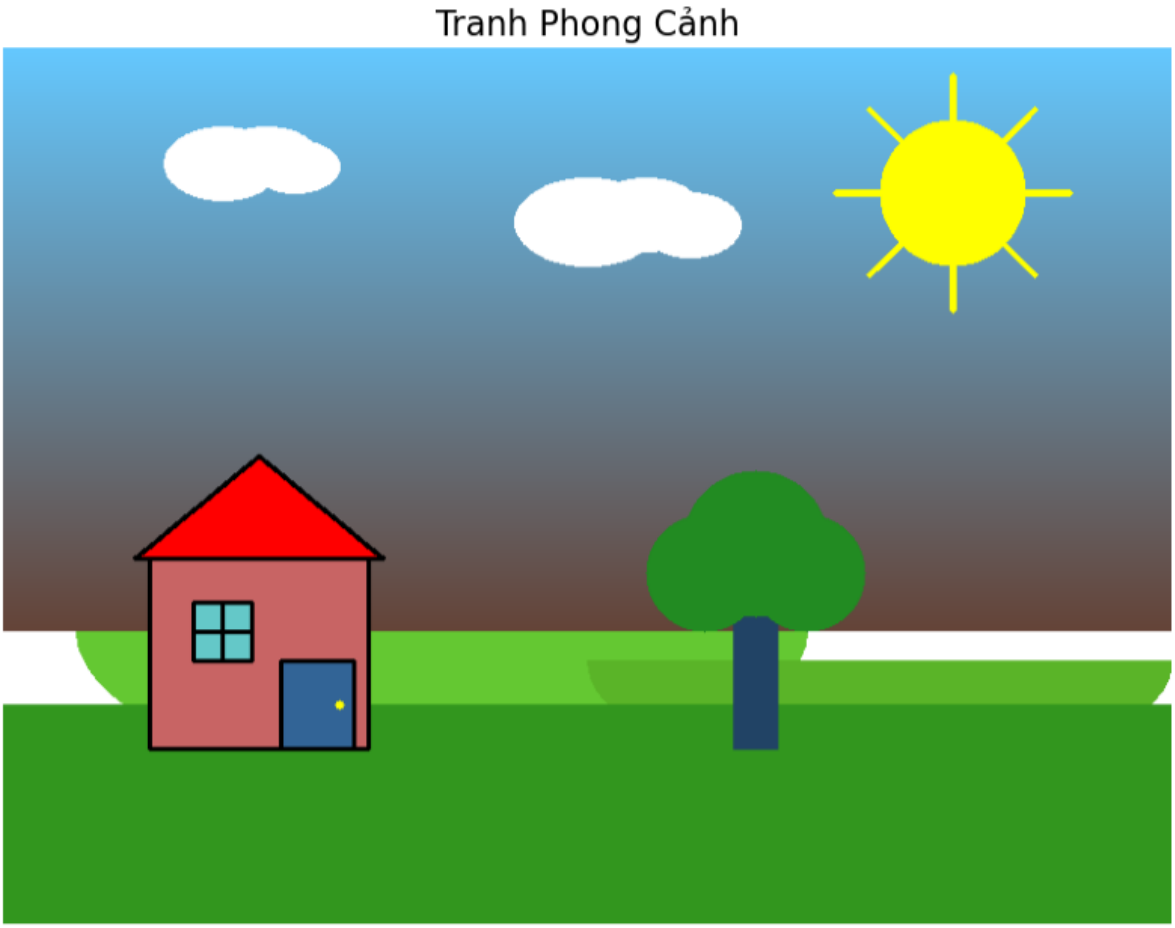
Sự thay đổi tuyến tính này giúp màu sắc chuyển đổi mượt mà từ sắc xanh sáng phía trên (khi i nhỏ) sang sắc xanh đậm hơn hoặc có pha màu khác ở phía dưới (khi i lớn), mô phỏng hiệu ứng bầu trời.

3.1.2 Chồng lớp hình học (Geometric Layering)

Các vật thể phức tạp được xây dựng bằng cách chồng nhiều hình đơn giản, được tô đầy (thickness = -1), lên nhau theo thứ tự từ xa đến gần, hoặc từ nền đến chi tiết.

- **Vẽ mặt trời**: Là một hình tròn tô đầy. Tuy nhiên, các tia sáng được thêm vào bằng cách tính toán tọa độ của 8 đường thẳng (cv2.line) tỏa ra từ tâm, tạo ra hiệu ứng ánh sáng rực rỡ.

- **Vẽ ngôi nhà**: Kết hợp hình chữ nhật (thân nhà), tam giác (mái nhà) và hình chữ nhật nhỏ hơn (cửa sổ, cửa ra vào). Thứ tự vẽ mái nhà sau thân nhà là quan trọng để đảm bảo chi tiết được chồng lớp đúng.



Hình 3.1: Tranh phong cảnh

3.2 Kỹ thuật chèn và hiệu ứng text

Việc chèn văn bản không chỉ là đặt chữ lên ảnh mà còn bao gồm các kỹ thuật đồ họa để làm nổi bật thông điệp.

3.2.1 Sử dụng đa dạng font chữ

OpenCV cung cấp một bộ các font chữ HERSHEY khác nhau, mỗi font có một hằng số riêng. Notebook đã minh họa các font như:

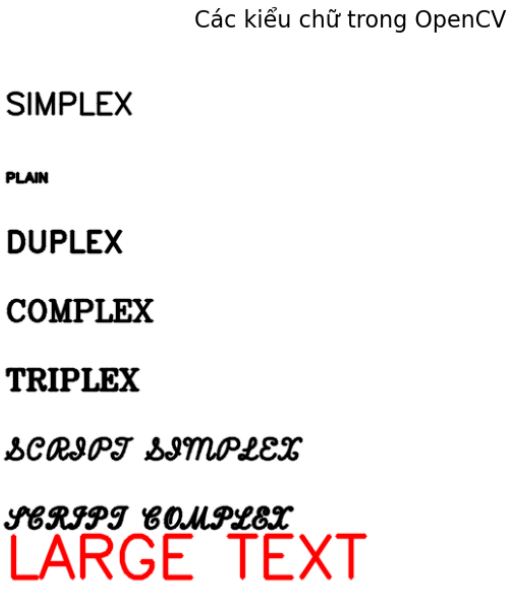
- FONT_HERSHEY_SIMPLEX

- DUPLEX

- COMPLEX

- SCRIPT_SIMPLEX.

Sự khác biệt giữa chúng nằm ở độ phức tạp của nét vẽ(ví dụ: SIMPLEX đơn giản, TRIPLEX có ba nét vẽ).



Hình 3.2: Các kiểu chữ trong OpenCV

3.2.2 Tạo hiệu ứng viền (Outline Text)

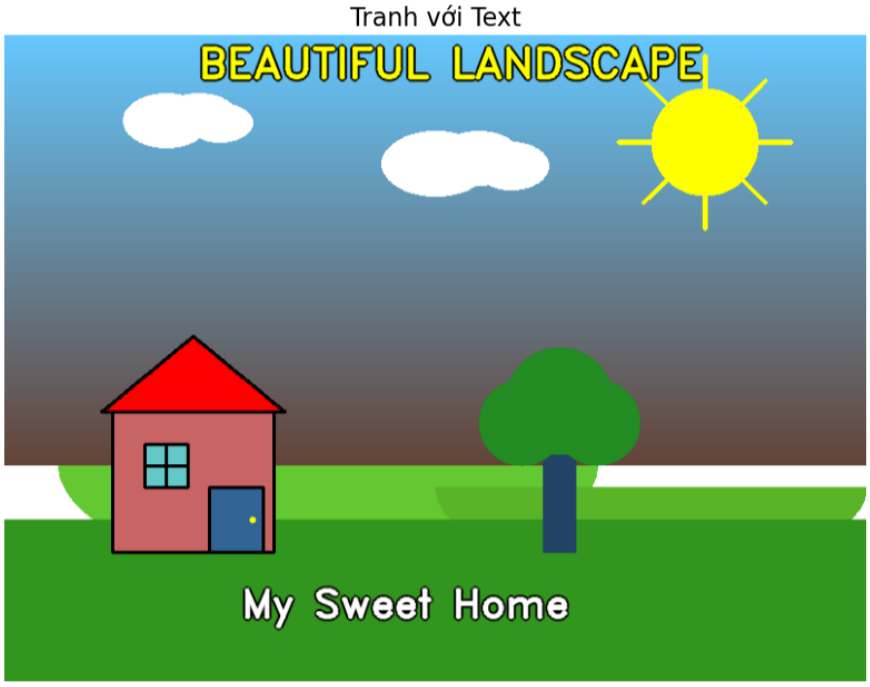
Đây là kỹ thuật nâng cao để cải thiện khả năng đọc của văn bản, đặc biệt khi nền ảnh có màu sắc hỗn hợp.

- **Kỹ thuật**: Thay vì chỉ gọi cv2.putText() một lần, kỹ thuật này yêu cầu gọi hai lần tại cùng một tọa độ:

+ Lớp Viền (Outline Layer): Vẽ text bằng màu tương phản (thường là đen) với độ dày lớn hơn (ví dụ: 5 pixels).

+ Lớp Chính (Fill Layer): Vẽ text bằng màu mong muốn (ví dụ: trắng hoặc vàng) với độ dày nhỏ hơn (ví dụ: 2 pixels).

- **Hiệu quả**: Lớp text chính sẽ lấp đầy phần trung tâm của lớp viền dày, tạo ra đường viền sắc nét xung quanh, giúp text nổi bật trên mọi nền ảnh.



Hình 3.3: Tranh với text

3.3 Ứng dụng thiết kế logo

Các ví dụ về logo và biểu tượng thể hiện tính ứng dụng thực tế của việc kết hợp linh hoạt các hàm vẽ hình.

3.3.1 Logo và biểu tượng mạng xã hội

Bảng 3.1: Logo và biểu tượng mạng xã hội

Sản phẩm	Kỹ thuật tổng hợp	Phân tích
Logo TechVision	cv2.circle (nền) + cv2.rectangle (chữ "T") + cv2.putText	Logo áp dụng nguyên tắc thiết kế tối giản, sử dụng hình học cơ bản (hình tròn, chữ nhật) để tạo ra biểu tượng dễ nhận biết.
Logo Morning Coffee	cv2.fillPoly (thân cốc) + cv2.ellipse (quai cốc, đĩa lót) + cv2.polylines (hơi nước)	Sử dụng hình thang (fillPoly) để tạo thân cốc và tận dụng tham số cung tròn của cv2.ellipse để vẽ quai cốc, mô phỏng đối tượng 3D.
Biểu tượng (Icon)	cv2.circle, cv2.rectangle, cv2.fillPoly cho các icon: Heart, Camera, Share	Minh họa khả năng tạo ra các icon đơn giản (biểu tượng hóa), sử dụng các khối màu solid để đạt được hiệu ứng phẳng (Flat Design).

Logo TechVision

Hình 3.4: Logo Techvision

Logo Morning Coffee

Hình 3.5: Logo Morning Coffee

Hình 3.6: Biểu tượng mạng xã hội

3.3.2 Logo Game (Legend of Warriors)

Logo này là ứng dụng phức tạp nhất, yêu cầu sự kết hợp chính xác của nhiều kỹ thuật:

- **Khiên (Shield)**: Kết hợp nửa elip (phần trên của khiên) và đa giác (phần dưới hình chữ V) bằng cách sử dụng cv2.ellipse và cv2.fillPoly.

- **Kiếm (Sword)**: Được tạo ra bằng các đa giác nhỏ (lưỡi kiếm) và đường thẳng (thanh đỡ, chuôi), sau đó được xoay và đặt đối xứng để tạo cảm giác về quyền lực và chiến đấu.

- **Text Viền**: Kỹ thuật text viền được áp dụng để làm nổi bật tên thương hiệu "LEGEND" trên nền màu xanh của khiên.

Hình 3.7: Logo game Legend of Warriors

PHẦN KẾT LUẬN

Đề tài "Vẽ và Tô Màu Hình Học Cơ Bản" đã hoàn thành xuất sắc các mục tiêu đề ra, cung cấp cái nhìn toàn diện và thực tế về nền tảng xử lý ảnh bằng thư viện OpenCV trong môi trường Python. Thông qua việc thực hành vẽ các hình học cơ bản như đường thẳng, hình chữ nhật, hình tròn, elip, và đa giác, chúng ta đã nắm vững các hàm cốt lõi của OpenCV và cách chúng thao tác trực tiếp lên ma trận pixel. Đặc biệt, báo cáo đã làm rõ sự cần thiết của việc hiểu hệ màu BGR của OpenCV và kỹ thuật tô đầy tạo nền tảng vững chắc cho mọi công việc xử lý ảnh sau này.

Các kỹ thuật nâng cao hơn như tạo gradient màu bằng cách lặp lại cv2.line(), chồng lớp hình học để xây dựng các vật thể phức tạp, và tạo hiệu ứng viền (outline) cho văn bản đã được triển khai thành công, minh chứng cho tính linh hoạt của thư viện. Những sản phẩm ứng dụng như tranh phong cảnh, logo công nghệ, và biểu tượng game không chỉ là minh họa trực quan mà còn thể hiện rằng OpenCV là một công cụ mạnh mẽ, không chỉ dừng lại ở phân tích dữ liệu mà còn mở rộng sang lĩnh vực thiết kế đồ họa cơ bản (basic iconography). Thành công của đề tài này là bước đệm quan trọng, chuẩn bị kiến thức và kỹ năng cần thiết để tiếp tục nghiên cứu và triển khai các thuật toán thị giác máy tính phức tạp hơn trong tương lai.

DANH MỤC TÀI LIỆU THAM KHẢO

1. Giáo trình môn học Nhập môn xử lý ảnh
2. Slide môn học Nhập môn xử lý ảnh