1. Kỹ thuật cài đặt

Ứng dụng được xây dựng bằng Python sử dụng thư viện OpenCV để thực hiện phát hiện cạnh ảnh bằng toán tử Laplacian. Giao diện người dùng được triển khai bằng Streamlit, hỗ trợ tải ảnh lên, điều chỉnh tham số, xem kết quả và tải xuống ảnh đã xử lý.

Các bước xử lý ảnh:

1. **Tải ảnh lên**: Người dùng chọn một ảnh từ thiết bị (định dạng PNG, JPG, JPEG, BMP, WEBP).
2. **Chuyển đổi định dạng**: Ảnh từ PIL được chuyển sang mảng NumPy để dùng với OpenCV. Nếu ảnh có kênh alpha (RGBA), ta chuyển về BGR để xử lý đúng:

```
image = np.array(image)
if len(image.shape) == 3 and image.shape[2] == 4:
    image = cv.cvtColor(image, cv.COLOR_RGBA2BGR)
elif len(image.shape) == 2:
    image = cv.cvtColor(image, cv.COLOR_GRAY2BGR)
```

3. **Điều chỉnh kích thước ảnh**: Hàm resize_with_aspect_ratio() giúp giữ nguyên tỷ lệ ảnh trong khi thay đổi chiều rộng theo giá trị do người dùng chọn:

```
def resize_with_aspect_ratio(image, width=None, height=None,
inter=cv.INTER_AREA):
    dim = None
    (h, w) = image.shape[:2]

    if width is None and height is None:
        return image

    if width is None:
        r = height / float(h)
        dim = (int(w * r), height)
    else:
        r = width / float(w)
        dim = (width, int(h * r))

    return cv.resize(image, dim, interpolation=inter)
```

4. **Làm mờ ảnh**: Áp dụng bộ lọc Gaussian để giảm nhiễu:

```
blurred = cv.GaussianBlur(image, (3, 3), 0)
```

5. **Chuyển ảnh sang thang độ xám**: Dùng cv.cvtColor() để dễ dàng áp dụng toán tử Laplacian:

```
gray = cv.cvtColor(blurred, cv.COLOR_BGR2GRAY)
```

6. **Áp dụng toán tử Laplacian**: Dùng cv.Laplacian() với kích thước kernel do người dùng chọn:

```
laplacian = cv.Laplacian(gray, cv.CV_16S, ksize=kernel_size)
```

7. **Chuyển đổi kết quả về uint8**: Dùng cv.convertScaleAbs() để đảm bảo ảnh có thể hiển thị đúng:

```
abs_laplacian = cv.convertScaleAbs(laplacian)
```

8. **Hiển thị kết quả**: Ảnh gốc và ảnh sau khi phát hiện cạnh được hiển thị song song trong giao diện Streamlit.
9. **Tải xuống ảnh**: Người dùng có thể tải ảnh kết quả về thiết bị bằng cách chuyển đổi ảnh OpenCV thành định dạng PIL:

```
buf = io.BytesIO()
Image.fromarray(abs_laplacian).save(buf, format='PNG')
st.download_button(
    label="Tải Xuống Kết Quả Phát Hiện Cạnh",
    data=buf.getvalue(),
    file_name="edge_detection.png",
    mime="image/png"
)
```

---

2. Các vấn đề gặp phải

- **Xử lý ảnh có kênh alpha (RGBA)**: Một số ảnh có kênh trong suốt, cần chuyển đổi sang BGR để đảm bảo xử lý đúng.
- **Lỗi khi tải ảnh lớn**: Giới hạn kích thước ảnh tải lên giúp tối ưu hiệu suất.
- **Hiệu suất trên hình ảnh lớn**: Khi kernel Laplacian quá lớn, thời gian xử lý tăng lên đáng kể.

- **Định dạng ảnh không phù hợp**: Một số ảnh đầu vào có thể không được hỗ trợ, gây lỗi khi hiển thị hoặc xử lý.

---

3. Khuyến khích cải thiện

- **Bổ sung bộ lọc khác**: Hỗ trợ thêm Sobel, Canny để so sánh kết quả.
- **Xử lý ảnh theo thời gian thực**: Cho phép chụp ảnh từ webcam và áp dụng Laplacian ngay lập tức.
- **Tùy chỉnh nâng cao**: Cho phép người dùng điều chỉnh mức làm mờ Gaussian hoặc độ tương phản trước khi áp dụng Laplacian.