Dự án phát hiện cạnh ảnh sử dụng toán tử Laplace

Phát biểu bài toán:

Bài toán phát hiện cạnh trong ảnh sử dụng toán tử Laplace là một kỹ thuật quan trọng trong lĩnh vực xử lý ảnh nhằm tìm ra các cạnh – những vùng có sự thay đổi đột ngột về cường độ sáng trong ảnh. Mục tiêu của dự án là xây dựng một hệ thống hoặc thuật toán có khả năng nhận diện các cạnh này bằng cách áp dụng toán tử Laplace, một toán tử vi phân bậc hai, để tính đạo hàm bậc hai của cường độ sáng tại mỗi pixel trong ảnh. Kết quả mong muốn là một ảnh mới, trong đó các cạnh được làm nổi bật để phục vụ cho các ứng dụng như phân tích ảnh, nhận diện đối tượng hoặc xử lý hình ảnh y khoa.

Bài toán phát hiện cạnh trong xử lý ảnh nhằm mục đích xác định các vùng trong ảnh có sự thay đổi đột ngột về cường độ sáng (intensity), thường tương ứng với biên giới hoặc cạnh của các đối tượng. Toán tử Laplace được sử dụng như một phương pháp để phát hiện các cạnh này thông qua việc tính toán đạo hàm bậc hai của cường độ ảnh.

Cụ thể:

- **Đầu vào**: Một ảnh xám (grayscale image) được biểu diễn dưới dạng ma trận I(x,y) I(x, y) I(x,y), trong đó I(x,y) I(x, y) I(x,y) là giá trị cường độ sáng tại vị trí pixel (x,y)(x, y)(x,y).
- **Mục tiêu**: Tìm các điểm trong ảnh nơi có sự thay đổi đột ngột về cường độ, tức là các cạnh, bằng cách áp dụng toán tử Laplace.
- **Phương pháp**: Sử dụng toán tử Laplace để tính đạo hàm bậc hai của ảnh, từ đó phát hiện các điểm mà tại đó đạo hàm bậc hai đổi dấu (zero-crossing), biểu thị vị trí của cạnh.
- **Đầu ra**: Một ảnh nhị phân hoặc ảnh đánh dấu các cạnh, trong đó các pixel thuộc cạnh được xác định.

Toán tử Laplace:

Toán tử Laplace cho một hàm liên tục f(x,y) f(x, y) f(x,y) được định nghĩa là:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Trong không gian rời rạc của ảnh số, toán tử Laplace được xấp xỉ bằng cách sử dụng mặt nạ (mask) tích chập. Một mặt nạ Laplace phổ biến là:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

hoặc phiên bản mở rộng với các hướng chéo:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Các bước thực hiện:

1. **Tiền xử lý**: Nếu cần, làm mịn ảnh đầu vào bằng bộ lọc Gaussian để giảm nhiễu, vì toán tử Laplace rất nhạy với nhiễu.
2. **Áp dụng toán tử Laplace**: Tích chập ảnh với mặt nạ Laplace để tính đạo hàm bậc hai tại mỗi pixel.
3. **Phát hiện zero-crossing**: Xác định các điểm mà giá trị đạo hàm bậc hai đổi dấu (từ dương sang âm hoặc ngược lại), đây là các vị trí của cạnh.
4. **Tạo ảnh đầu ra**: Đánh dấu các pixel thuộc cạnh trên ảnh kết quả.

Đặc điểm:

- Toán tử Laplace là một phương pháp đẳng hướng (isotropic), nghĩa là nó phát hiện cạnh mà không phụ thuộc vào hướng.
- Nhược điểm: Nhạy với nhiễu, vì vậy thường cần kết hợp với bước làm mịn trước khi áp dụng.

Hướng giải quyết

1. Về toán tử Laplace
    - Toán tử Laplace là một công cụ toán học dùng để phát hiện các điểm có sự thay đổi đột ngột trong hàm số, trong trường hợp này là cường độ sáng của ảnh.

- Nó hoạt động bằng cách tính đạo hàm bậc hai, giúp làm nổi bật các vùng chuyển tiếp mạnh trong ảnh.

2. Áp dụng toán tử Laplace lên ảnh:
    - Sử dụng một ma trận kernel Laplace (thường là ma trận 3x3) để thực hiện phép tích chập với ảnh gốc, qua đó tính toán đạo hàm bậc hai tại mỗi pixel.
    - Một kernel Laplace phổ biến có dạng:

```
text
```

```
0  1  0
1 -4  1
0  1  0
```

    Kernel này phát hiện cạnh theo cả hướng ngang và dọc.

3. Xử lý ảnh kết quả:
    - Sau khi áp dụng toán tử Laplace, ảnh thu được sẽ cho thấy các vùng có giá trị pixel cao tại những nơi cường độ sáng thay đổi mạnh.
    - Áp dụng ngưỡng (threshold) để phân loại các pixel thành cạnh hoặc không phải cạnh.
4. Tinh chỉnh kết quả
    - Để giảm nhiễu và tăng độ chính xác, có thể làm mịn ảnh trước khi áp dụng Laplace bằng bộ lọc Gaussian. Phương pháp này được gọi là Laplacian of Gaussian (LoG).

Thuật toán phát hiện cạnh ảnh sử dụng toán tử Laplace

Code:

```
"""
@file laplace_demo.py
@brief Sample code showing how to detect edges using the Laplace operator
"""
import sys
import cv2 as cv
def main(argv):
# [variables]
# Declare the variables we are going to use
ddepth = cv.CV_16S
kernel_size = 3
window_name = "Laplace Demo"
# [variables]
# [load]
imageName = argv[0] if len(argv) > 0 else 'lena.jpg'
src = cv.imread(cv.samples.findFile(imageName), cv.IMREAD_COLOR) # Load an image
# Check if image is loaded fine
if src is None:
print ('Error opening image')
print ('Program Arguments: [image_name -- default lena.jpg]')
return -1
# [load]
# [reduce_noise]
# Remove noise by blurring with a Gaussian filter
src = cv.GaussianBlur(src, (3, 3), 0)
# [reduce_noise]
# [convert_to_gray]
# Convert the image to grayscale
src_gray = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
# [convert_to_gray]
# Create Window
cv.namedWindow(window_name, cv.WINDOW_AUTOSIZE)
# [laplacian]
# Apply Laplace function
dst = cv.Laplacian(src_gray, ddepth, ksize=kernel_size)
# [laplacian]
# [convert]
# converting back to uint8
abs_dst = cv.convertScaleAbs(dst)
# [convert]
# [display]
cv.imshow(window_name, abs_dst)
cv.waitKey(0)
# [display]
return 0
if __name__ == "__main__":
main(sys.argv[1:])
```

1. Đọc ảnh đầu vào

Mô tả: Ảnh được đọc từ tệp được chỉ định thông qua tham số dòng lệnh (argv[0]) hoặc sử dụng ảnh mặc định lena.jpg nếu không có tham số nào được cung cấp. Hàm cv.imread() từ thư viện OpenCV được sử dụng để tải ảnh dưới dạng ảnh màu (RGB).
Kiểm tra lỗi: Nếu ảnh không được tải thành công (do đường dẫn sai hoặc tệp không tồn tại), chương trình sẽ in thông báo lỗi và kết thúc với mã lỗi -1.
Ý nghĩa: Đây là bước đầu tiên để lấy dữ liệu ảnh cần xử lý.

2. Làm mịn ảnh để giảm nhiễu
   Mô tả: Ảnh đầu vào được làm mịn bằng bộ lọc Gaussian với kích thước kernel 3x3 thông qua hàm cv.GaussianBlur(). Tham số (3, 3) xác định kích thước kernel, và 0 là độ lệch chuẩn mặc định của Gaussian.
   Mục đích: Giảm nhiễu trong ảnh trước khi áp dụng toán tử Laplace, giúp hạn chế việc phát hiện các cạnh giả do nhiễu gây ra.
   Ý nghĩa: Bước tiền xử lý này rất quan trọng để cải thiện độ chính xác của thuật toán phát hiện cạnh.
3. Chuyển đổi ảnh sang thang độ xám
   Mô tả: Ảnh màu (RGB) được chuyển đổi sang ảnh xám (grayscale) bằng hàm cv.cvtColor() với mã màu cv.COLOR_BGR2GRAY.
   Mục đích: Đơn giản hóa dữ liệu ảnh từ ba kênh màu (RGB) xuống còn một kênh duy nhất, vì toán tử Laplace chỉ cần xử lý trên một kênh cường độ sáng.
   Ý nghĩa: Việc này giúp giảm độ phức tạp tính toán và phù hợp với yêu cầu của toán tử Laplace.
4. Áp dụng toán tử Laplace
   Mô tả: Toán tử Laplace được áp dụng lên ảnh xám bằng hàm cv.Laplacian(). Các tham số bao gồm:
   - src_gray: Ảnh xám làm đầu vào.
   - ddepth = cv.CV_16S: Độ sâu của ảnh đầu ra, sử dụng định dạng 16-bit có dấu để tránh tràn số khi tính đạo hàm.
   - ksize = 3: Kích thước kernel 3x3, xác định vùng lân cận được sử dụng để tính đạo hàm bậc hai.

   Nguyên lý: Toán tử Laplace tính đạo hàm bậc hai của cường độ sáng, giúp phát hiện các vùng có sự thay đổi đột ngột (các cạnh) trong ảnh.
   Ý nghĩa: Đây là bước cốt lõi của thuật toán, xác định vị trí các cạnh dựa trên sự thay đổi độ sáng.

5. Chuyển đổi kết quả về định dạng uint8
   Mô tả: Kết quả từ hàm Laplace (dạng 16-bit có dấu) được chuyển đổi sang định dạng 8-bit không dấu (uint8) bằng hàm cv.convertScaleAbs(). Hàm này lấy giá trị tuyệt đối và điều chỉnh phạm vi giá trị để phù hợp với hiển thị (0-255).

Mục đích: Định dạng uint8 là yêu cầu chuẩn để hiển thị ảnh trên màn hình.
Ý nghĩa: Bước này đảm bảo kết quả có thể được trực quan hóa một cách chính xác

6. Hiển thị kết quả
   Mô tả: Một cửa sổ hiển thị có tên "Laplace Demo" được tạo bằng cv.namedWindow(), và ảnh kết quả (abs_dst) được hiển thị bằng cv.imshow(). Chương trình chờ người dùng nhấn phím bất kỳ (cv.waitKey(0)) trước khi đóng.
   Ý nghĩa: Cho phép người dùng xem kết quả cuối cùng của thuật toán, tức là ảnh với các cạnh đã được phát hiện.