Bài thực hành chương 2 (phần 2): 26 - CANNY EDGE DETECTOR

I> Lý thuyết

1. Tìm hiểu về các bước của thuật toán Canny

a - Giải thích chi tiết từng bước:

Thuật toán phát hiện cạnh Canny là một phương pháp phổ biến bao gồm các bước sau:

1. **Giảm nhiễu (Noise Reduction):** Vì phát hiện cạnh rất nhạy cảm với nhiễu, bước đầu tiên là lọc nhiễu trong ảnh bằng bộ lọc Gaussian (Gaussian filter). Điều này giúp làm mượt ảnh và ngăn chặn các cạnh giả do nhiễu gây ra.
2. **Tính toán Gradient (Finding Intensity Gradients):** Ảnh đã làm mượt sau đó được lọc bằng hạt nhân Sobel theo cả chiều ngang và chiều dọc để lấy đạo hàm bậc nhất (G_x và G_y). Từ đó, độ lớn cạnh (Edge Gradient) và hướng (Direction) cho mỗi pixel được tính toán:
   - $Edge\_Gradient(G) = \sqrt{G_x^2 + G_y^2}$
   - $Angle(\theta) = \tan^{-1}(\frac{G_y}{G_x})$ Góc độ dốc được làm tròn đến một trong bốn góc đại diện: 0, 45, 90 hoặc 135 độ.
3. **Non-maximum Suppression (Triệt tiêu phi cực đại):** Bước này loại bỏ các pixel không phải là cực đại cục bộ theo hướng gradient. Thuật toán sẽ quét qua toàn bộ ảnh, tại mỗi pixel, nó kiểm tra xem liệu pixel đó có phải là cực đại cục bộ trong vùng lân cận của nó theo hướng gradient hay không. Nếu không, nó sẽ bị đặt về 0 (loại bỏ).
4. **Ngưỡng kép (Hysteresis Thresholding):** Bước này quyết định pixel nào thực sự là cạnh. Cần hai giá trị ngưỡng: `minVal` và `maxVal`.
   - Cạnh mạnh: Các pixel có gradient lớn hơn `maxVal` chắc chắn là cạnh.
   - Cạnh yếu: Các pixel có gradient nằm giữa `minVal` và `maxVal` được coi là cạnh yếu (tiềm năng).
   - Không phải cạnh: Các pixel có gradient nhỏ hơn `minVal` bị loại bỏ.
5. **Theo dõi cạnh (Edge Tracking by Hysteresis):** Các cạnh yếu được giữ lại chỉ khi chúng kết nối với các cạnh mạnh. Điều này dựa trên giả định rằng các cạnh nhiễu hoặc biến đổi màu nhỏ sẽ không kết nối với các đường viền mạnh.

b - So sánh với các thuật toán khác như Sobel, Laplacian:

Đặc điểm	Canny	Sobel	Laplacian
Nguyên lý	Dựa trên tối ưu hóa đa bước (Gradient + NMS + Thresholding).	Dựa trên đạo hàm bậc nhất (Gradient).	Dựa trên đạo hàm bậc hai (Zero-crossing).

Đặc điểm	Canny	Sobel	Laplacian
Độ chính xác	Cao, phát hiện cạnh mảnh (1 pixel), định vị tốt.	Trung bình, cạnh thường dày và thô.	Nhạy cảm với nhiễu cao, có thể tạo cạnh kép.
Nhiễu	Xử lý nhiễu tốt nhờ Gaussian filter tích hợp.	Nhạy cảm với nhiễu.	Rất nhạy cảm với nhiễu.
Độ phức tạp	Cao hơn (nhiều bước tính toán).	Thấp, tính toán nhanh.	Thấp.

2. Các tham số của thuật toán và ảnh hưởng của chúng

a - Sigma trong Gaussian filter:

- Tham số này xác định mức độ làm mượt của ảnh.
- **Sigma lớn:** Làm mượt nhiều hơn, loại bỏ nhiễu tốt hơn nhưng cũng làm mờ các cạnh chi tiết, có thể làm mất các cạnh yếu.
- **Sigma nhỏ:** Làm mượt ít hơn, giữ lại nhiều chi tiết hơn nhưng cũng giữ lại nhiều nhiễu hơn.

b - Ngưỡng thấp (minVal) và ngưỡng cao (maxVal):

- Quyết định độ nhạy của việc phát hiện cạnh.
- **Khoảng cách giữa hai ngưỡng lớn:** Ít cạnh nhiễu hơn nhưng có thể làm đứt đoạn các cạnh.
- **Ngưỡng thấp quá nhỏ:** Sẽ bắt được nhiều nhiễu là cạnh.
- **Ngưỡng cao quá lớn:** Có thể bỏ sót các cạnh mờ.
- Tỷ lệ thường dùng là 1:2 hoặc 1:3.

3. Ưu điểm và nhược điểm của Canny edge detector. Các ứng dụng thực tế

Ưu điểm:

- Khả năng phát hiện lỗi thấp (Low error rate).
- Định vị điểm cạnh tốt (Good localization).
- Phản hồi duy nhất cho một cạnh (Single edge response).

Nhược điểm:

- Phức tạp về mặt tính toán so với Sobel/Prewitt.
- Khó điều chỉnh tham số ngưỡng tự động cho mọi loại ảnh.

Các ứng dụng thực tế:

a - So sánh về hiệu suất:

- Canny thường chậm hơn Sobel do nhiều bước xử lý, nhưng cho kết quả chính xác hơn nhiều cho các tác vụ cần độ tin cậy cao.

b - Lĩnh vực phổ biến:

- Thị giác máy tính (Computer Vision): Nhận dạng đối tượng, trích xuất đặc trưng.
- Y tế: Phân tích ảnh X-quang, MRI để tìm biên khối u, cơ quan.
- Công nghiệp: Kiểm tra sản phẩm, phát hiện lỗi vết nứt.
- Xe tự lái: Phát hiện làn đường.

c - Ví dụ cụ thể:

- Phát hiện biển báo giao thông.
- Nhận dạng vân tay.
- Scan tài liệu (xác định biên giấy).

II> Bài tập thực hành

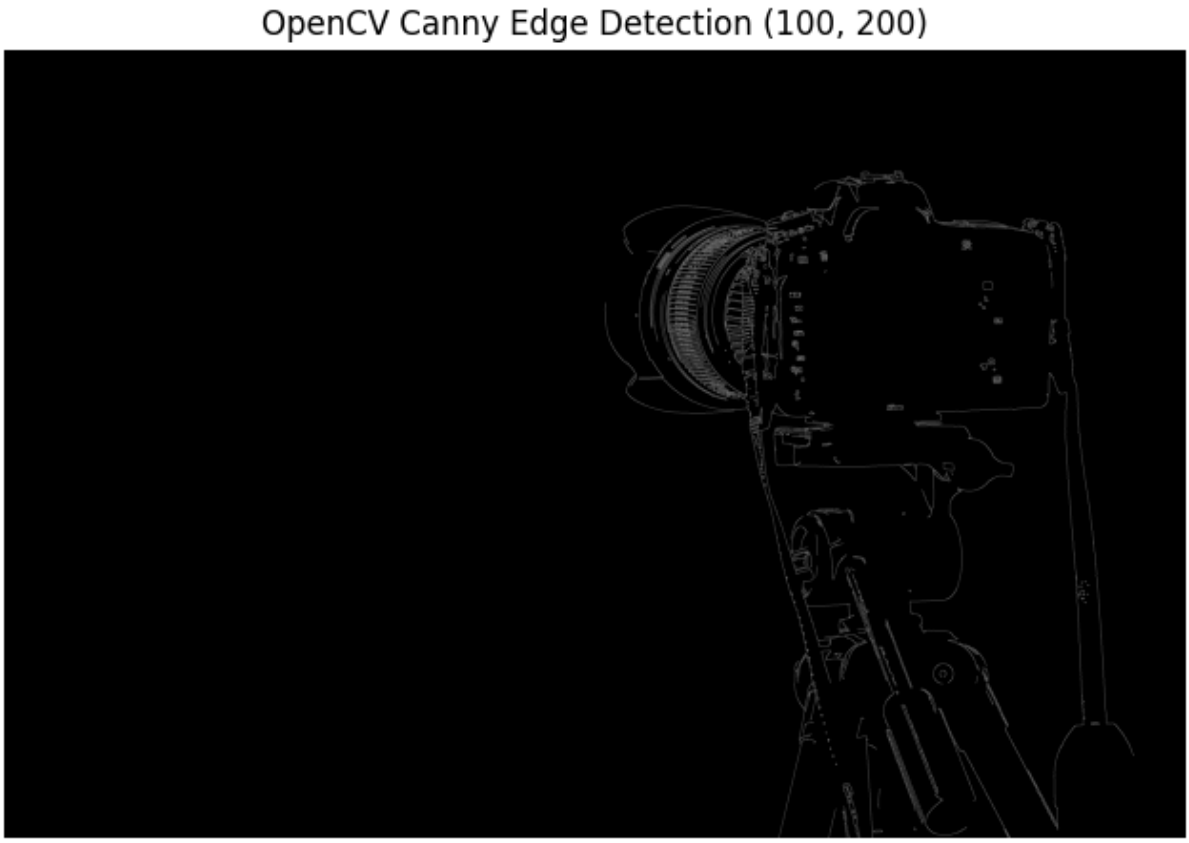
Chuẩn bị môi trường và tải ảnh

In [ ]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import feature, color, io
import os

# Đường dẫn ảnh (Giả sử ảnh nằm ở thư mục Session_1)
image_path = 'Images/image_1.jpg'

if not os.path.exists(image_path):
    print(f"Warning: Image not found at {image_path}. Please check the path.")
else:
    # Đọc ảnh bằng OpenCV
    img_cv = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Đọc ảnh để dùng với Scikit-image (đọc dạng màu rồi chuyển xám)
    img_ski = io.imread(image_path)
    img_ski_gray = color.rgb2gray(img_ski)

    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(img_cv, cmap='gray')
    plt.title('Original Image (Grayscale)')
    plt.axis('off')
    plt.show()
```

1. Thực hiện thuật toán Canny bằng các thư viện

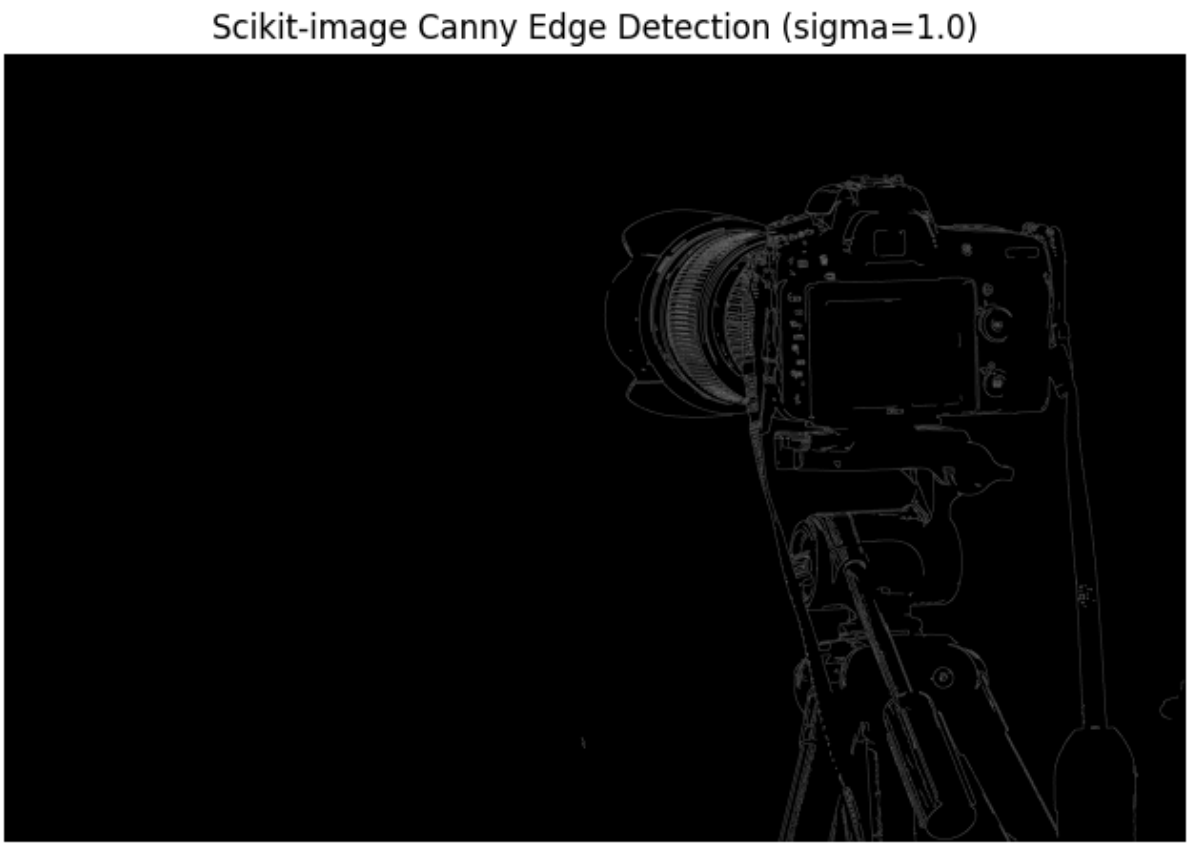
a) OpenCV: `cv2.Canny`

In [3]:
```
# OpenCV Canny
# Tham số mặc định ví dụ: threshold1=100, threshold2=200
edges_cv = cv2.Canny(img_cv, 100, 200)

plt.figure(figsize=(8, 6))
plt.imshow(edges_cv, cmap='gray')
plt.title('OpenCV Canny Edge Detection (100, 200)')
plt.axis('off')
plt.show()
```



b) Scikit-image: `skimage.feature.canny`

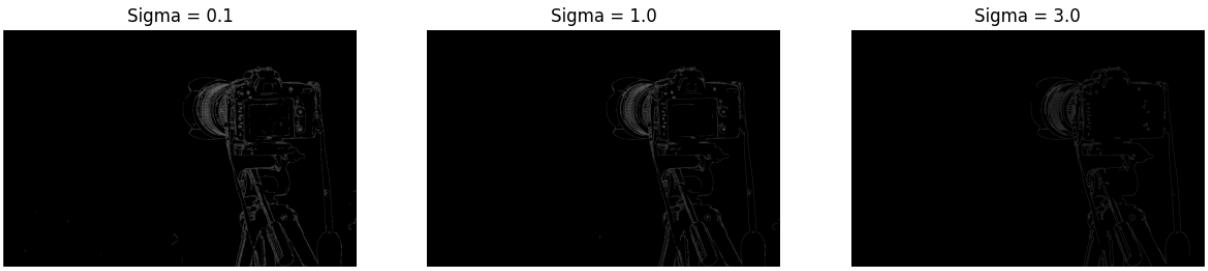
In [4]:
```
# Scikit-image Canny
# Sigma mặc định là 1.0
edges_ski = feature.canny(img_ski_gray, sigma=1.0)

plt.figure(figsize=(8, 6))
plt.imshow(edges_ski, cmap='gray')
plt.title('Scikit-image Canny Edge Detection (sigma=1.0)')
plt.axis('off')
plt.show()
```



2. Thay đổi các tham số và quan sát kết quả

a) Thay đổi Sigma (Scikit-image)

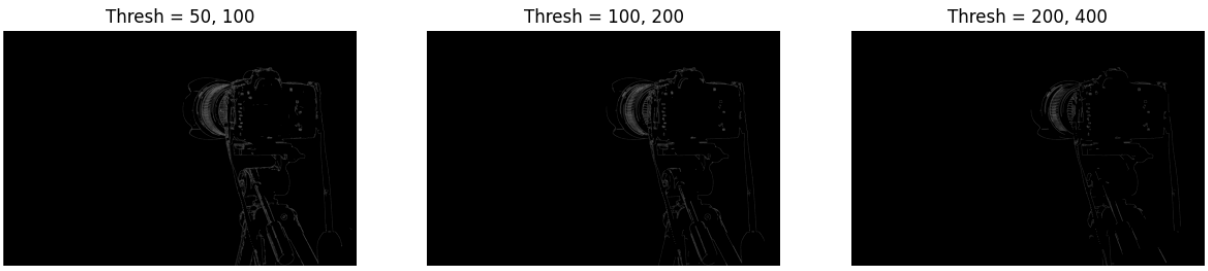
In [5]:
```
sigmas = [0.1, 1.0, 3.0]
plt.figure(figsize=(15, 5))

for i, sig in enumerate(sigmas):
    edges = feature.canny(img_ski_gray, sigma=sig)
    plt.subplot(1, 3, i + 1)
    plt.imshow(edges, cmap='gray')
    plt.title(f'Sigma = {sig}')
    plt.axis('off')
plt.show()
```



Nhận xét: Sigma càng lớn, ảnh càng được làm mượt nhiều hơn trước khi phát hiện cạnh, dẫn đến ít chi tiết hơn (chỉ các cạnh rất mạnh mới được giữ lại).

b) Thay đổi Ngưỡng (OpenCV)

In [6]:
```
thresholds = [(50, 100), (100, 200), (200, 400)]
plt.figure(figsize=(15, 5))

for i, (t1, t2) in enumerate(thresholds):
    edges = cv2.Canny(img_cv, t1, t2)
    plt.subplot(1, 3, i + 1)
    plt.imshow(edges, cmap='gray')
    plt.title(f'Thresh = {t1}, {t2}')
    plt.axis('off')
plt.show()
```

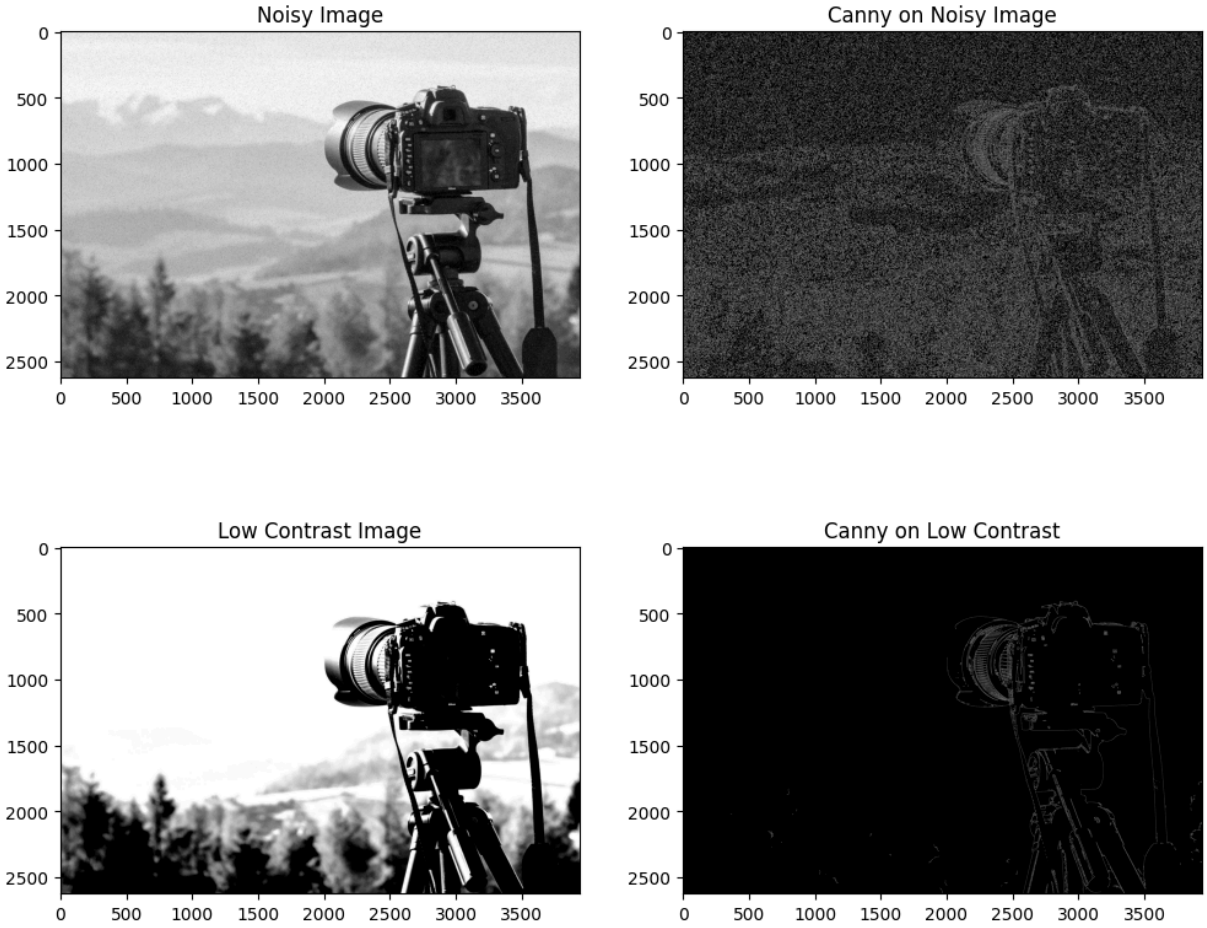


Nhận xét: Ngưỡng càng cao, thuật toán càng khắt khe, chỉ các cạnh có độ tương phản rất cao mới được hiển thị. Ngưỡng thấp giúp bắt được nhiều chi tiết nhưng kèm theo nhiễu.

3. Áp dụng Canny cho các loại ảnh khác nhau

Chúng ta sẽ giả lập các điều kiện ảnh khác nhau từ ảnh gốc để kiểm tra: Ảnh nhiễu, Ảnh độ tương phản thấp.

In [7]:
```
from skimage.util import random_noise
from skimage import exposure

# 1. Ảnh nhiễu (Salt & Pepper)
noisy_img = random_noise(img_ski_gray, mode='s&p', amount=0.05)
noisy_img_uint8 = (noisy_img * 255).astype(np.uint8)

# 2. Ảnh độ tương phản thấp
low_contrast_img = exposure.rescale_intensity(img_ski_gray, in_range=(0.3, 0.7))
low_contrast_img_uint8 = (low_contrast_img * 255).astype(np.uint8)

# Áp dụng Canny
edges_noisy = cv2.Canny(noisy_img_uint8, 100, 200)
edges_low_contrast = cv2.Canny(low_contrast_img_uint8, 50, 150) # Cần giảm ngưỡng c

plt.figure(figsize=(12, 10))

plt.subplot(2, 2, 1); plt.imshow(noisy_img, cmap='gray'); plt.title('Noisy Image')
plt.subplot(2, 2, 2); plt.imshow(edges_noisy, cmap='gray'); plt.title('Canny on Noi

plt.subplot(2, 2, 3); plt.imshow(low_contrast_img, cmap='gray'); plt.title('Low Con
plt.subplot(2, 2, 4); plt.imshow(edges_low_contrast, cmap='gray'); plt.title('Canny

plt.show()
```



Đánh giá:

- Với ảnh nhiễu, Canny dễ bắt nhầm nhiễu thành cạnh nếu không tăng sigma (làm mượt) hoặc tăng ngưỡng.
- Với ảnh độ tương phản thấp, gradients nhỏ nên cần giảm ngưỡng để phát hiện được cạnh, nhưng cũng dễ bị nhiễu hơn.

4. Kết hợp Canny với các kỹ thuật khác (Mở rộng)

Ví dụ: Sử dụng Canny để tìm viền, sau đó tìm Contours (nhận dạng hình dạng).

In [8]:
```
# Tìm contours từ ảnh Canny
contours, _ = cv2.findContours(edges_cv, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE

# Vẽ contours lên ảnh gốc (chuyển sang BGR để vẽ màu)
img_contours = cv2.cvtColor(img_cv, cv2.COLOR_GRAY2BGR)
cv2.drawContours(img_contours, contours, -1, (0, 255, 0), 1)

plt.figure(figsize=(8, 6))
plt.imshow(cv2.cvtColor(img_contours, cv2.COLOR_BGR2RGB))
plt.title(f'Detected {len(contours)} contours using Canny')
plt.axis('off')
plt.show()
```

III> Các câu hỏi mở rộng

1. Làm thế nào để đánh giá chất lượng của các cạnh được phát hiện bởi Canny?

- **Định lượng:** Sử dụng các số liệu như Precision, Recall, F-measure so với Ground Truth (ảnh biên chuẩn do con người gán nhãn). Sử dụng thang đo như FOM (Figure of Merit) của Pratt.
- **Định tính:** Kiểm tra khả năng kết nối (continuity) của các cạnh, độ mảnh (thinness) của cạnh (lý tưởng là 1 pixel), và khả năng loại bỏ nhiễu.

2. Có những phương pháp nào khác để cải thiện hiệu suất của Canny?

- **Tự động xác định ngưỡng (Otsu's method):** Thay vì chọn ngưỡng thủ công, dùng thống kê của ảnh để chọn ngưỡng `min` và `max` động.
- **Adaptive Canny:** Áp dụng các tham số khác nhau cho từng vùng của ảnh dựa trên đặc tính cục bộ.
- **Tiền xử lý tốt hơn:** Sử dụng Bilateral Filter thay vì Gaussian để giữ biên tốt hơn trong khi lọc nhiễu.

3. Canny có thể được sử dụng để phát hiện các cạnh trong ảnh màu không? Nếu có, thì như thế nào?

- Có. Thông thường người ta chuyển ảnh màu sang xám (Grayscale) trước khi áp dụng Canny.

- Tuy nhiên, có thể áp dụng Canny trên từng kênh màu (R, G, B) riêng biệt và sau đó gộp kết quả lại (dùng phép OR logic hoặc lấy max gradient) để tăng độ chính xác ở những nơi có sự thay đổi màu sắc nhưng độ sáng không đổi.

4. Làm thế nào để áp dụng Canny cho các video?

- Video là chuỗi các khung hình (frames). Ta áp dụng thuật toán Canny cho từng frame riêng lẻ.
- Để ổn định kết quả giữa các frame (tránh cạnh nhấp nháy), có thể sử dụng thông tin từ frame trước để làm mượt hoặc điều chỉnh ngưỡng động.