UJIAN AKHIR PENGOLAHAN CITRA

PENERAPAN OPERATOR TEPI UNTUK PENGOLAHAN CITRA

Pendahuluan

Pengolahan Citra merupakan salah satu ilmu yang berkembang paa teknologi informasi. Deteksi tepi pada citra adalah salah satu point dalam pengolahan citra, yang memiliki tujuan untuk melakukan identifikasi perubahan dalam piksel di sekitar area citra. Tepi citra dapat memberikan informasi mengenai batas antara objek, perubahan tekstur, dan perubahan lain yang dapat digunakan dalam berbagai sistem atau aplikasi pengenalan pola, segmentasi objek, dan paling sering dalam melakukan rekonstruksi citra.

Dalam paper ini, penulis mencoba melakukan deteksi tepi pada citra dengan menggunakan beberapa algoritma yang umum digunakan, seperti Sobel, LoG, Canny, dan Prewitt. Algoritma yang akan digunakan ini merupakan beberapa algoritma yang akan menjadi percobaan dalam deteksi tepi menggunakan google colabs dan algoritma tersebut sudah memberikan hasil yang baik dibeberapa aplikasi.

Terdapat beberapa alasan dalam memilih algoritma deteksi tepi yang sesuai. Kecepatan komputasi, akurasi deteksi tepi, dan kemampuan dalam menangani *noise* dan variasi dalam citra merupakan beberapa alasan yang penting untuk di pikirkan. Oleh karena itu, dalam percobaan ini, penulis bertujuan untuk membandingkan dan menganalisis kinerja dari beberapa algoritma deteksi tepi yang umum digunakan. Melalui paper ini, penulis berharap dapat memberikan pemahaman yang lebih baik tentang kelebihan dan kelemahan dari masing masing algoritma deteksi tepi pada citra yang di gunakan.

Tinjauan Pustaka

Deteksi tepi pada citra telah menjadi topik yang luas dalam bidang pengolahan citra. Beberapa algoritma deteksi tepi yang digunakan adalah Sobel, Prewitt, dan Canny. Dalam tinjauan ini, penulis akan membahas mengenai prinsip kerja dari masing masing algoritma.

Algoritma Sobel merupakan salah satu algoritma yang popular dalam melakukan deteksi tepi pada pengolahan citra. Algoritma ini menggunakan operasi konvulisi dengan *kernel* sobel yang terdiri dari 2 matriks 3x3 yang merupakan *kernel* untuk melakukan perhitungan gradien *horizontal* dan *kerner* untuk bagian gradien *vertical*. Terdapat kelebihan dan kelemahan dalam algoritma ini, kelebihannya adalah kemampuannya dalam menangani *noise* pada citra dan menghasilkan tepi yang tajam. Untuk kelemahannya algoritma ini cendrung sensitif terhadap *noise* dan kurang efektif dalam mendeteksi tepi yang tipis.

Algoritma Prewitt juga merupakan salah satu algoritma populer dalam melakukan deteksi tepi pada pengolahan citra. Seperti algortima sobel, algoritma ini menggunakan operasi konvolusi dengan kernel Prewitt yang berbeda untuk gradien *horizontal* dan *vertical*. Kelebihan dari algoritma ini adalah kemampuannya dalam mendeteksi tepi yang tipis dan menghasilkan tepi yang cukup tajam. Sedangkan untuk kelemahannya algoritma ini kurang efektif dalam menangani *noise* dan tidak sebagus sobel dalam mengasilkan tepi.

Algoritma Canny merupakan algoritma deteksi tepi yang simple dan sering digunakan. Algoritma ini memiliki beberapa tahap, termasuk dengan penerapan filter *Gaussian* untuk

melakukan penghalusan citra, perhitungan gradien menggunakan operator sobel, penekanan non maksimum untuk memperoleh tepi yang tipis dan penambatan histeris untuk menghubungkan tepi yang saling berhubungan. Untuk kelebihan algoritma ini adalah kemampuannya dalam mendeteksi tepi dengan baik. Sedangkan untuk kelemahannya algoritma ini membutuhkan komputasi yang lebih rumit.

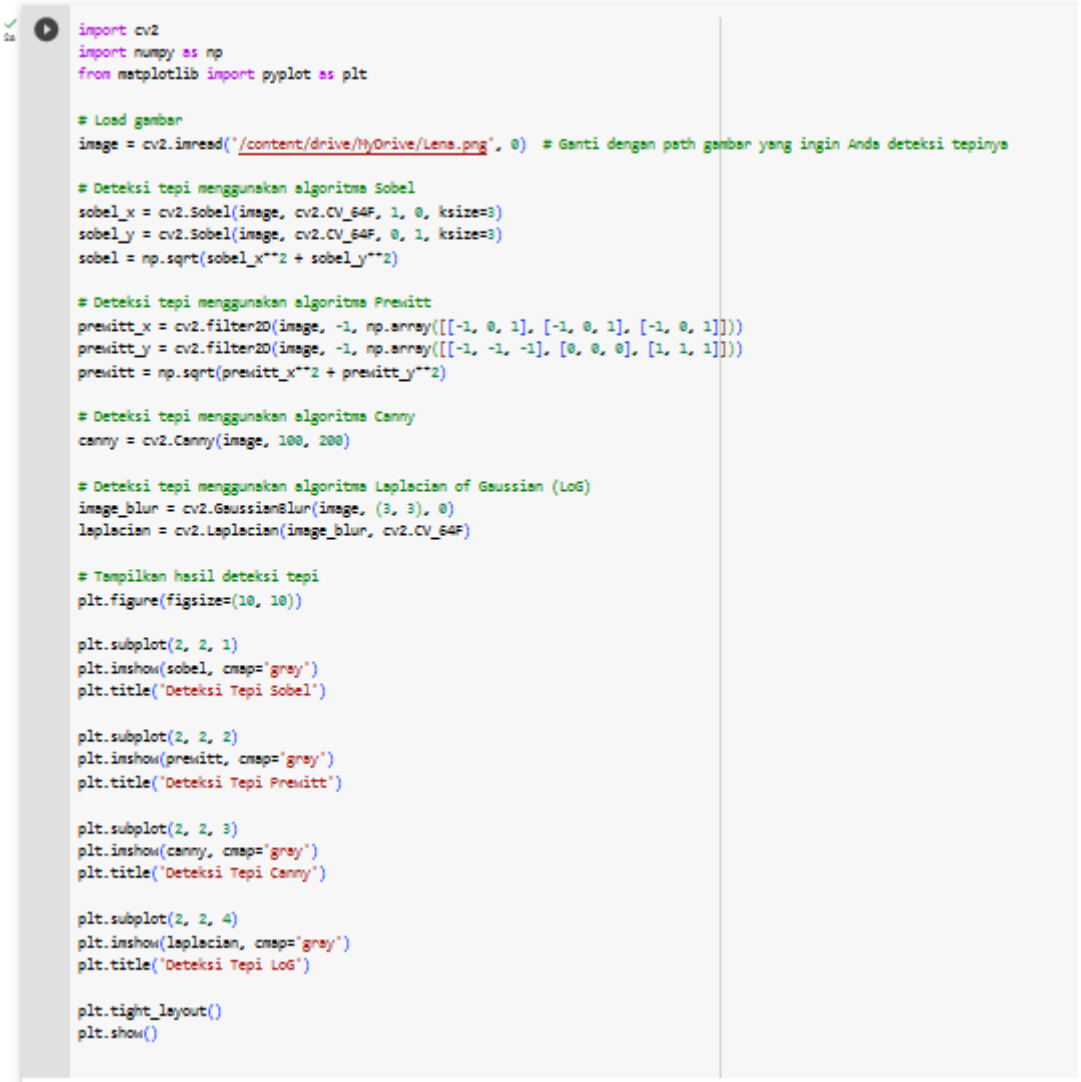
Algortima LoG (*Laplacian of Gaussian*) merupakan penggabungan antara filter *Gaussian* dengan operator *Laplacian* untuk mendeteksi tepi. Filter *Gaussian* digunakan untuk menghaluskan citra, kemudian operator *Laplacian* digunakan untuk menghaluskan citra yang telah di haluskan. Untuk kelebihan dari algortima LoG adalah kemampuannya dalam mendeteksi tepi dengan presisi yang tinggi. Sedangkan untuk kelemahannya algoritma ini cengerung menghasilkan banyak respons tepi yang palsu.

Berikut adalah code google colabs dari deteksi tepi pada gambar lena dengan menggunakan beberapa algoritma

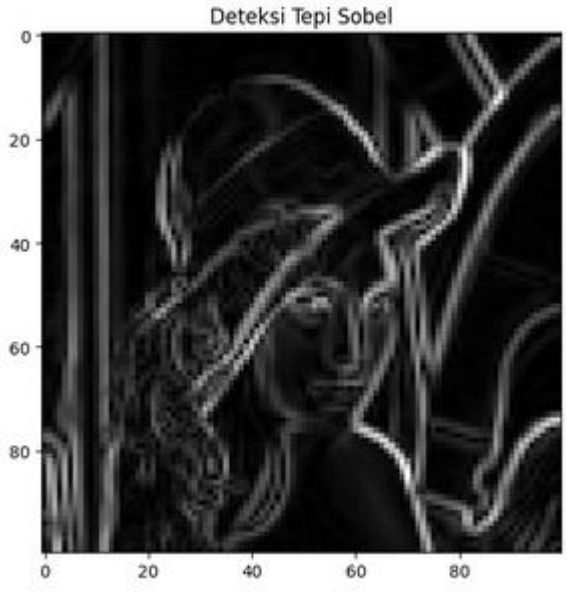
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load gambar
image = cv2.imread('/content/drive/MyDrive/Lena.png', 0)

# Deteksi tepi menggunakan algoritma Sobel
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
sobel = np.sqrt(sobel_x**2 + sobel_y**2)

# Deteksi tepi menggunakan algoritma Prewitt
prewitt_x = cv2.filter2D(image, -1, np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]]))
prewitt_y = cv2.filter2D(image, -1, np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]]))
prewitt = np.sqrt(prewitt_x**2 + prewitt_y**2)

# Deteksi tepi menggunakan algoritma Canny
canny = cv2.Canny(image, 100, 200)

# Deteksi tepi menggunakan algoritma Laplacian of Gaussian (LoG)
image_blur = cv2.GaussianBlur(image, (3, 3), 0)
laplacian = cv2.Laplacian(image_blur, cv2.CV_64F)

# Tampilkan hasil deteksi tepi
plt.figure(figsize=(10, 10))

plt.subplot(2, 2, 1)
plt.imshow(sobel, cmap='gray')
plt.title('Deteksi Tepi Sobel')

plt.subplot(2, 2, 2)
plt.imshow(prewitt, cmap='gray')
plt.title('Deteksi Tepi Prewitt')

plt.subplot(2, 2, 3)
plt.imshow(canny, cmap='gray')
plt.title('Deteksi Tepi Canny')
```

```
plt.subplot(2, 2, 4)
plt.imshow(laplacian, cmap='gray')
plt.title('Deteksi Tepi LoG')

plt.tight_layout()
plt.show()
```

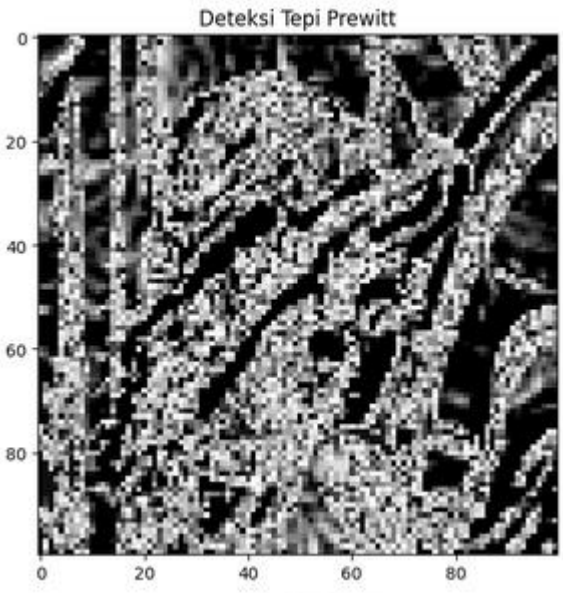
atau dapat mengakses link berikut:
https://colab.research.google.com/drive/12gjFJfZRvYbWWWtEejRRoJz7nSu8i_p8?usp=sharing



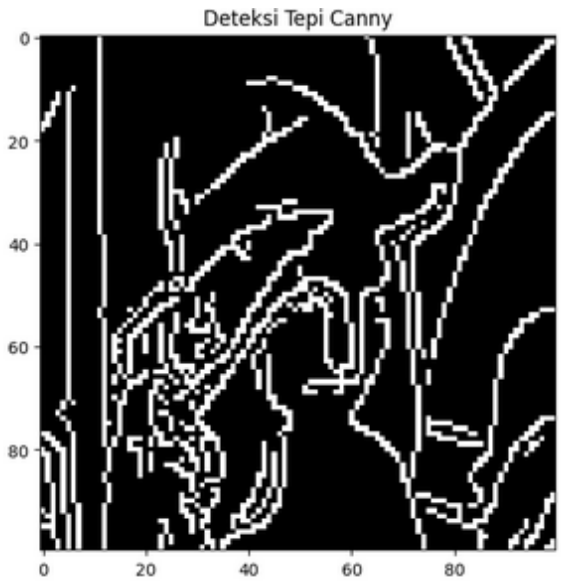
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load gambar
image = cv2.imread('/content/drive/MyDrive/Lena.png', 0)  # Ganti dengan path gambar yang ingin Anda deteksi tepinya

# Deteksi tepi menggunakan algoritma Sobel
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
sobel = np.sqrt(sobel_x**2 + sobel_y**2)

# Deteksi tepi menggunakan algoritma Prewitt
prewitt_x = cv2.filter2D(image, -1, np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]]))
prewitt_y = cv2.filter2D(image, -1, np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]]))
prewitt = np.sqrt(prewitt_x**2 + prewitt_y**2)

# Deteksi tepi menggunakan algoritma Canny
canny = cv2.Canny(image, 100, 200)

# Deteksi tepi menggunakan algoritma Laplacian of Gaussian (LoG)
image_blur = cv2.GaussianBlur(image, (3, 3), 0)
laplacian = cv2.Laplacian(image_blur, cv2.CV_64F)

# Tampilkan hasil deteksi tepi
plt.figure(figsize=(10, 10))

plt.subplot(2, 2, 1)
plt.imshow(sobel, cmap='gray')
plt.title('Deteksi Tepi Sobel')

plt.subplot(2, 2, 2)
plt.imshow(prewitt, cmap='gray')
plt.title('Deteksi Tepi Prewitt')

plt.subplot(2, 2, 3)
plt.imshow(canny, cmap='gray')
plt.title('Deteksi Tepi Canny')

plt.subplot(2, 2, 4)
plt.imshow(laplacian, cmap='gray')
plt.title('Deteksi Tepi LoG')

plt.tight_layout()
plt.show()
```



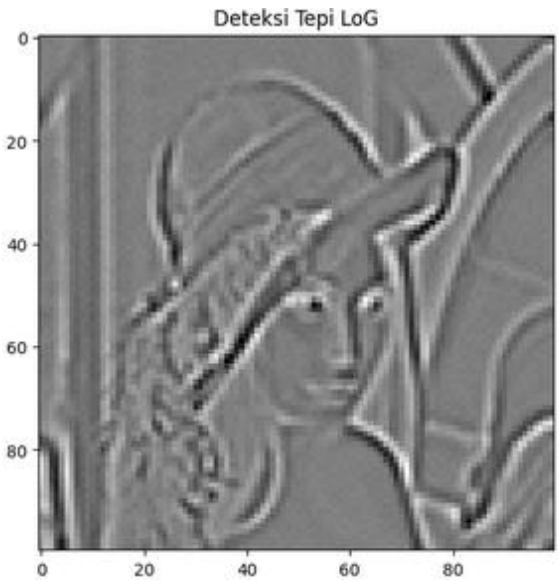
Hasil 1 Deteksi Sobel



Hasil 2 Deteksi Prewitt



Hasil 3 Deteksi Canny



Hasil 4 Deteksi LoG

Kesimpulan

Dalam percobaan ini, penulis telah mencoba dan membandingkan ke-empat algoritma deteksi tepi pada citra, yaitu Sobel, Prewitt, Canny, dan LoG (*Laplacian of Gaussian*). Berhasarkan hasil evaluasi, didapatkan bahwa setiap algoritma memiliki kelebihan dan kelemahannya masing masing. Seperti algoritma sobel dan prewitt yang menghasilkan tepi yang cukup tajam tetapi kedua algoritma ini sensitive terhadap *noise* pada citra. Untuk algoritma canny menghasilkan tepi yang presisi dan peka terhadap *noise* namun memerlukan komputasi yang lebih rumit. Sedangkan untuk algoritma LoG menghasilkan tepi yang lebih presisi dengan menggabungkan filter *Gaussian* dan dan operator *Laplacian* namun menghasilkan respons dari tepi palsu.