Nama : Fadia Ramadhana
NIM : G64170026

Problem #2: Edge Detection

Gambar asli :

1.Implementasikan algoritma Canny Edge Detection pada gambar tersebut melalui kelima langkah* berikut

a. Noise reduction; Pada tahap ini lakukan teknik image convolution dengan Gaussian Kernel 5x5 dan 7x7. Tuliskan apa dampak dari perbedaan ukuran kernel yang digunakan. Sertakan gambar yang dihasilkan dari penggunaan kernel yang berbeda-beda tersebut

Noise reduction diperlukan karena pada tahap perhitungan gradien secara matematika akan dilakukan penurunan fungsi yang menyebabkan hasil deteksi tepi sangat sensitif terhadap noise. Salah satu cara untuk menghilangkan noise adalah dengan menerapkan filter Gaussian Blur yang bekerja dengan cara menghaluskan image. Gaussian blur memiliki beberapa kernel untuk melakukan konvolusi yaitu 3x3, 5x5, 7x7, dst. Semakin besar ukuran kernel, maka semakin blur pula gambar yang dihasilkan.
Pada image **IPB_Exam2.jpg** telah diterapkan Gaussian blur dengan dua jenis kernel yaitu 5x5 dan 7x7 menggunakan fungsi **cv2.GaussianBlur()** yang akan melakukan perhitungan dengan size kernel (2k+1)x(2k+1) :

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \leq i, j \leq (2k+1)$$

Sebelumnya, image tersebut telah dibaca dengan format grayscale. Output yang dihasilkan adalah sebagai berikut :

Gaussian Blur dengan kernel 5x5 :	**Gaussian Blur dengan kernel 7x7 :**
	

b. Gradient calculation;

Gradient calculation adalah step dimana akan mendeteksi intensitas dan arah tepi dengan menghitung gradien dari image menggunakan operator deteksi tepi. Tepi pada image berhubungan dengan perubahan intensitas piksel. Untuk mendeteksinya, cara termudah adalah dengan menerapkan filter yang berfokus pada perubahan intensitas piksel tersebut di kedua arah; horizontal (x) dan vertikal (y). Setelah image diterapkan proses smoothing, maka akan dihitung turunannya yang selanjutnya akan dilakukan konvolusi dengan kernel pada filter. Pada perhitungan ini, saya menggunakan filter Sobel yang memiliki kernel sebagai berikut :

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

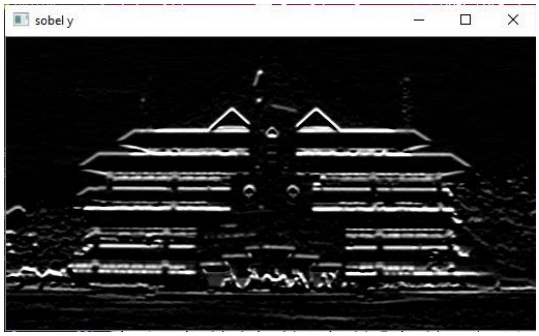
Kemudian, **magnitude** (intensitas gradien) **G** dan **kemiringan** (arah tepi) θ akan dihitung dengan rumus sebagai berikut :

$$|G| = \sqrt{I_x^2 + I_y^2},$$
$$\theta(x,y) = arctan\left(\frac{I_y}{I_x}\right)$$

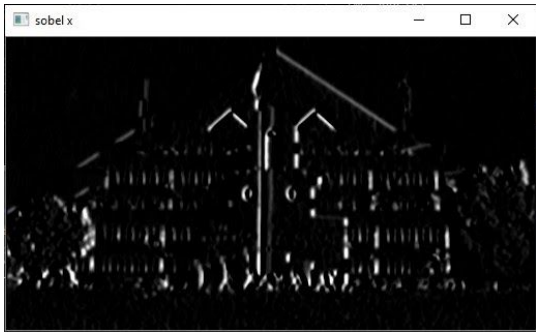
Pada OpenCV, tersedia fungsi **cv2.Sobel()** yang dapat digunakan untuk menerapkan konvolusi filter Sobel kepada sebuah image secara horizontal atau vertikal. Kemudian akan diperoleh output sebagai berikut :

Sobel Filter (Horizontal) dengan kernel 5x5 :

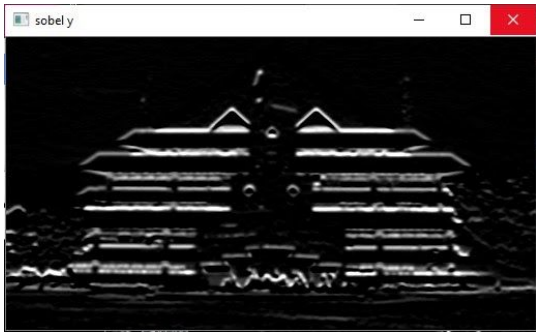
Sobel Filter (Vertikal) dengan kernel 5x5 :



Sobel Filter (Horizontal) dengan kernel 7x7 :



Sobel Filter (Vertikal) dengan kernel 7x7 :



Nantinya, setelah diterapkan filter secara horizontal maupun vertikal, kedua image akan digabungkan melalui perbandingan dengan menggunakan fungsi **cv2.bitwise_or()** dan diperoleh output sebagai berikut :

Sobel Filter dengan kernel 5x5 :

Sobel Filter dengan kernel 7x7 :

Magnitude (intensitas gradien) **G** dan **Kemiringan** (arah tepi) θ kernel 5*5 :

```
Gradient Calculation for Kernel 5*5 :
 [[ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 [ 0  1  3 ...  1  0  0]
 ...
 [ 8 15 14 ...  5 14 20]
 [ 6 12  9 ...  6 30 20]
 [ 0  6  8 ...  8 24  0]]
Theta for Kernel 5*5 :
 [[0.      0.      0.      ... 0.      0.      0.     ]
 [0.      0.      0.      ... 0.      0.      0.     ]
 [0.      0.785   1.249   ... 0.785   0.      0.     ]
 ...
 [1.57    0.567   0.5405  ... 1.373   1.166   1.57   ]
 [1.57    0.4636  0.11066 ... 0.785   0.661   1.57   ]
 [0.      0.      0.      ... 0.      0.      0.     ]]
```

Magnitude (intensitas gradien) **G** dan **Kemiringan** (arah tepi) θ kernel 7*7 :

```
Gradient Calculation for Kernel 7*7 :
 [[ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 [ 0  1  3 ...  1  0  0]
 ...
 [ 6  7 14 ...  6 10 10]
 [ 4  6  7 ...  5  7  8]
 [ 0  6  4 ...  2  6  0]]
Theta for Kernel 7*7 :
 [[0.     0.     0.     ... 0.     0.     0.    ]
 [0.     0.     0.     ... 0.     0.     0.    ]
 [0.     0.785  1.249  ... 0.785  0.     0.    ]
 ...
 [1.57   0.9507 0.9272 ... 1.57   1.326  1.57  ]
 [1.57   0.588  0.5405 ... 1.373  0.9507 1.57  ]
 [0.     0.     0.     ... 0.     0.     0.    ]]
```

c. Non-maximum suppression

Magnitude dari image memproduksi garis tepi dengan intensitas yang tebal. Idealnya, gambar final harus memiliki garis tepi yang tipis. Maka dari itu algoritma Non-Max Suppression dilakukan untuk menipiskan garis-garis tepi. Non maximum suppression bekerja dengan cara menemukan piksel dengan nilai maksimum pada tepi. Non maximum suppression tanpa interpolasi akan membagi 3x3 grid dari piksel menjadi 8 section sesuai besaran sudut. Contohnya, jika arah gradien jatuh diantara sudut -22.5 dan 22.5, maka kita akan menggunakan piksel yang jatuh diantara sudut tersebut (q dan r) sebagai nilai untuk perbandingan dengan nilai piksel p, contohnya seperti gambar berikut :

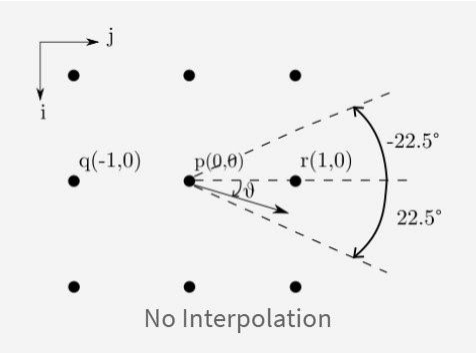
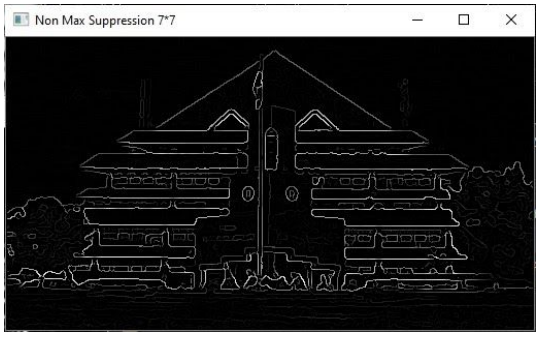


Image **IPB_Exam2.jpg** yang telah kita cari magnitude (intensitas gradien) dan arah dari sudutnya selanjutnya akan diterapkan algoritma non max suppression. Output yang didapat adalah sebegai berikut :

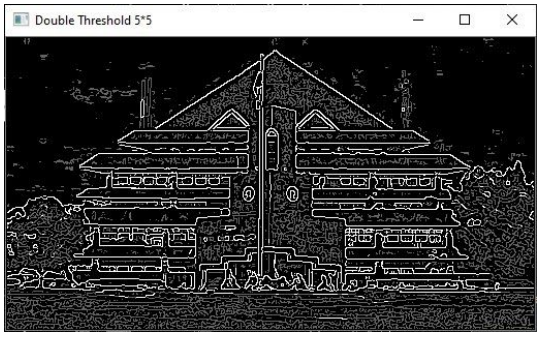
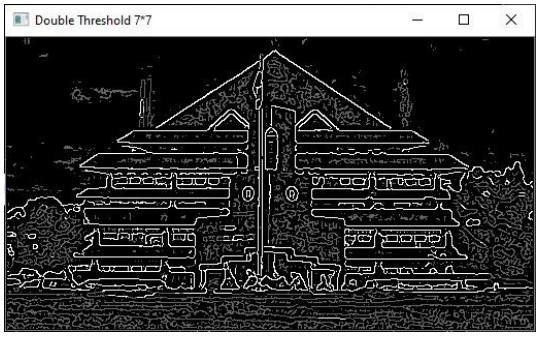
Non max suppression dengan kernel 5x5 :	**Non max suppression dengan kernel 7x7 :**
	

d. Double threshold; Pada tahap ini, ujikan gambar dengan berbagai nilai threshold. Tuliskan apa dampak dari menurunkan dan menaikkan nilai low threshold dan high threshold.

Setelah melalui proses non maximum suppression, output yang didapat belumlah sempurna, beberapa tepi mungkin tidak terdeteksi dan dapat dilihat hasil yang didapat masih memiliki noise. Untuk mengatasi masalah ini, kita akan melakukan **Double Thresholding.** Proses ini akan mengidentifikasi piksel-piksel ke dalam tiga jenis yaitu : strong, weak, dan non-relevant :

- Strong piksel adalah piksel yang memiliki intensitas sangat tinggi sehingga piksel-piksel ini pasti akan masuk ke dalam tepi final
- Weak piksel adalah piksel yang memiliki nilai intensitas yang tidak cukup untuk menjadi strong piksel namun tidak cukup kecil untuk dikelompokkan sebagai non-relevant piksel pada deteksi tepi
- Non relevant piksel adalah piksel yang tidak termasuk ke dalam kelompok piksel tepi

Double threshold membutuhkan dua buah threshold yaitu high dan low threshold. High threshold berfungsi untuk mengidentifikasi strong piksel (intensitas > high threshold). Sedangkan low threshold berfungsi untuk mengidentifikasi non-relevant piksel (intensitas < low threshold). Semua piksel yang jatuh di antara kedua threshold ini nantinya akan diberi label sebagai weak piksel.
Image yang telah melewati proses non maximum suppression selanjutnya akan diterapkan algoritma Double Threshold dengan low threshold ratio adalah 0.09 dan high threshold ratio adalah 0.13. Untuk mendapatkan nilai ini saya melakukan tune in secara manual. Nantinya dari ratio ini kita akan mendapatkan nilai dari low threshold dan high threshold. Untuk nilai piksel weak di set 100 dan strong piksel di set 255. Kemudian diperoleh output sebagai berikut :

Double Threshold kernel 5x5 :	**Double Threshold kernel 7x7 :**
	

Yang berwarna putih jelas adalah strong piksel, sedangkan yang berwarna abu-abu membayang adalah weak piksel.

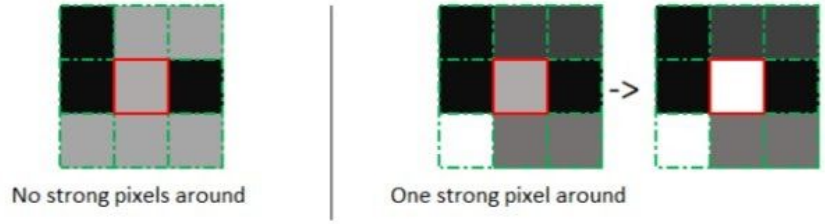
Seberapa banyak piksel yang diklasifikasikan kedalam weak piksel atau strong piksel sangat bergantung pada nilai low dan high threshold. Contohnya dengan mengurangi nilai low threshold menjadi 0.1 diperoleh hasil sebagai berikut :

Dapat dilihat bahwa semakin banyak piksel yang diklasifikasikan menjadi weak piksel karena low threshold yang terlalu rendah. Begitu pula jika menaikkan nilai high threshold, misalnya menjadi 0.7 diperoleh hasil sebagai berikut :

Dapat dilihat bahwa semakin sedikit piksel yang diklasifikasikan sebagai strong piksel karena high threshold yang terlalu tinggi.

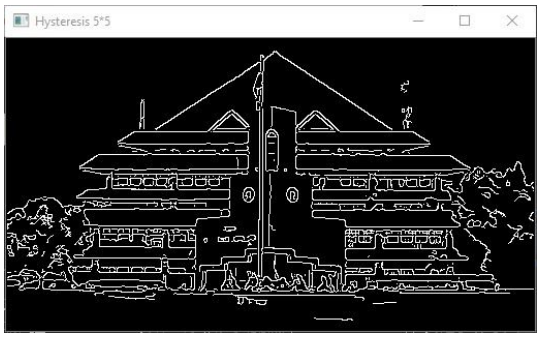
e. Edge Tracking by Hysteresis.

Piksel-piksel yang jatuh di antara kedua threshold ini (weak piksel) selanjutnya akan diidentifikasi apakah piksel tersebut cukup mumpuni untuk dikelompokkan sebagai strong piksel atau sebagai non relevant piksel melalui bantuan algoritma **Hysteresis.** Sebuah weak piksel akan ditransformasi menjadi strong piksel jika dan hanya jika setidaknya ada salah satu dari piksel yang berada di dekatnya diidentifikasi sebagai strong piksel. Weak piksel yang tidak memiliki hubungan apapun dengan strong piksel selanjutnya akan di removed.

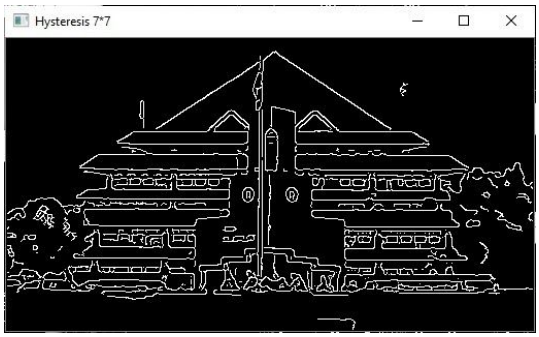


Hasil yang diperoleh dari tahapan ini adalah sebagai berikut :

Hysteresis kernel 5x5 :



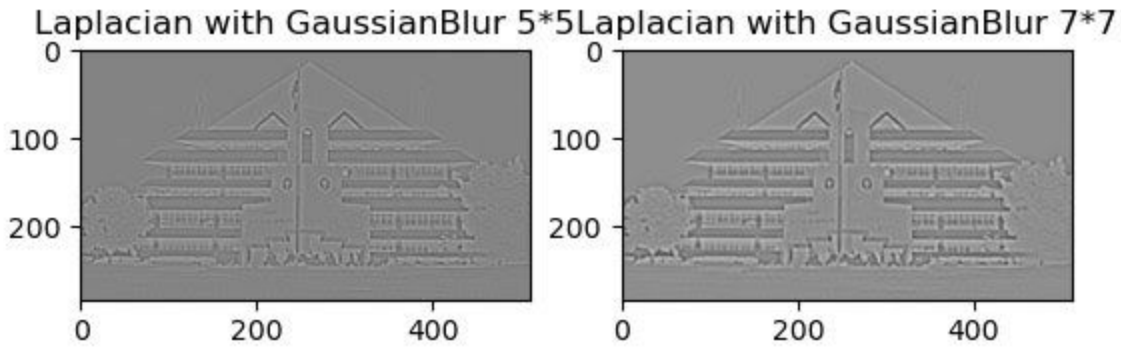
Hysteresis kernel 7x7 :



2. Setelah melakukan pendeteksian tepi dengan algoritme Canny, bandingkan hasil yang didapat dengan algoritme edge detection yang berbeda: Sobel/Prewitt/Laplacian (cukup pilih salah satu algoritme sebagai pembanding). Jelaskan algoritme mana yang menurut Anda paling baik dari teknik-teknik tersebut berdasarkan kriteria: good detection, good localization, single response constraint.

Algoritme edge detection : Laplacian

Output :



Ada tiga kriteria untuk pendeteksian tepi yang optimal, yaitu:

- Good Detection : detektor yang optimal harus dapat meminimalisir peluang dari false positive (mendeteksi tepi yang salah akibat noise) dan juga false negative (tepi aktual yang tidak terdeteksi)
- Good Localization : tepi yang terdeteksi harus sedekat mungkin dengan tepi yang sebenarnya
- Single response constraint : detektor harus dapat mengembalikan hanya satu poin untuk setiap tepi atau biasa disebut dengan proses non-maximum suppression

Laplacian operator adalah sebuah operator yang memanfaat turunan kedua dan biasa digunakan untuk deteksi tepi. Jika dibandingkan dengan edge detector yang memiliki basis turunan pertama, Laplacian operator menghasilkan hasil yang lebih baik pada edge localization. Meskipun begitu, karena memiliki metode turunan kedua, Laplacian sangat sensitif terhadap noise. Menurut saya, algoritme Canny edge detection dapat menyediakan metode yang lebih baik dan reliabel karena pendekatannya sendiri berdasarkan tiga objektif dasar yaitu *good detection*, *good localization*, dan *minimal response*. Canny mendefinisikan kriteria untuk deteksi dan lokalisasi tepi serta menambahkan batasan bahwa operator harus dapat menyediakan sebuah respon untuk setiap tepi. Good detection pada canny juga berjalan seiring dengan rendahnya peluang terhadap false positive atau false negative edge

Sumber referensi :

https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

http://justin-liang.com/tutorials/canny/

https://github.com/StefanPitur/Edge-detection---Canny-detector/blob/master/canny.py

https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html

https://github.com/abhilas0/edge_detection/blob/master/edge_detection.ipynb

https://www.researchgate.net/publication/336125919_COMPARATIVE_STUDY_AMONG_SOBEL_PREWITT_AND_CANNY_EDGE_DETECTION_OPERATORS_USED_IN_IMAGE_PROCESSING

https://www.cse.unr.edu/~bebis/CS791E/Notes/EdgeDetection.pdf