Apa itu Spatial Filtering?

sebelum ke penjelasan mengenai Spatial Filtering, seperti kulgram sebelumnya, saya menyediakan file jupyter notebook yang terintegrasi dengan Google Colab. Berikut link dari Google Colab yang akan digunakan.

https://colab.research.google.com/drive/1iYaA-9Sai1TuVavlO46sDI3nBlLHLL4k

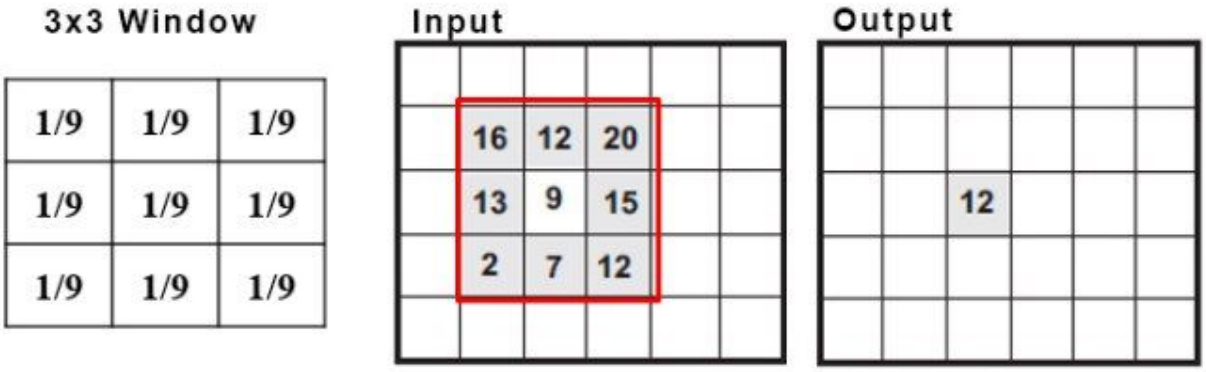
Spatial filtering adalah teknik pemrosesan gambar untuk mengubah intensitas piksel sesuai dengan intensitas piksel yang berdekatan atau bertetanggaan. Spatial filter* menggunakan gambar yang akan/telah ditransformasikan (terkonvolusi) berdasarkan pada kernel H yang memiliki tinggi dan lebar tertentu (x, y), yang mendefinisikan area dan berat piksel dalam gambar awal yang akan menggantikan nilai gambar .

Proses tersebut adalah untuk menggabungkan gambar input I (i, j) dengan fungsi filter H (x, y), untuk menghasilkan gambar yang difilter baru didenotasikan sebagai berikut:

$I'(i, j) = I(i, j) \circ H(x, y)$.

Operasi matematika yang digunakan pada spatial filter ini adalah perkalian dalam ruang frekuensi. Pemfilteran spasial dapat dikarakteristikkan sebagai operasi “geser-and-kali”: kernel bergeser ke atas gambar awal yang menghasilkan suatu mask dan mengalikan nilainya dengan nilai piksel yang sesuai dari gambar tersebut.

Hasilnya adalah nilai baru yang menggantikan nilai tengah di gambar baru. Ilustrasi mengenai spatial filter ini diilustrasikan pada gambar berikut:



Pada matriks paling kiri, diberikan sliding window berukuran 3x3 dengan nilai seperti yang tertera diatas. Gambar tengah anggap saja input. Kemudian hasil perkalian dari sliding window dengan input mengubah nilai tengah dari matriks input yang terkena sliding window (dari 9 menjadi 12). Setelah itu sliding window tersebut bergeser ke kanan satu persatu. Kemudian

bergeser ke bawah satu persatu. Proses ini umumnya dimulai dari nilai intensitas di pojok kiri bawah.

untuk coding yang digunakan, adalah sebagai berikut:

```
def convolve(image, kernel):
  # grab the spatial dimensions of the image, along with
  # the spatial dimensions of the kernel
  (iH, iW) = image.shape[:2]
  (kH, kW) = kernel.shape[:2]

  # allocate memory for the output image, taking care to
  # "pad" the borders of the input image so the spatial
  # size (i.e., width and height) are not reduced
  pad = (kW - 1) // 2
  image = cv2.copyMakeBorder(image, pad, pad, pad, pad,
    cv2.BORDER_REPLICATE)
  output = np.zeros((iH, iW), dtype="float32")
  # loop over the input image, "sliding" the kernel across
  # each (x, y)-coordinate from left-to-right and top to
  # bottom
  for y in np.arange(pad, iH + pad):
    for x in np.arange(pad, iW + pad):
      # extract the ROI of the image by extracting the
      # *center* region of the current (x, y)-coordinates
      # dimensions
      roi = image[y - pad:y + pad + 1, x - pad:x + pad + 1]

      # perform the actual convolution by taking the
      # element-wise multiplicate between the ROI and
      # the kernel, then summing the matrix
      k = (roi * kernel).sum()

      # store the convolved value in the output (x,y)-
      # coordinate of the output image
      output[y - pad, x - pad] = k
  # rescale the output image to be in the range [0, 255]
  output = rescale_intensity(output, in_range=(0, 255))
  output = (output * 255).astype("uint8")

  # return the output image
  return output
```

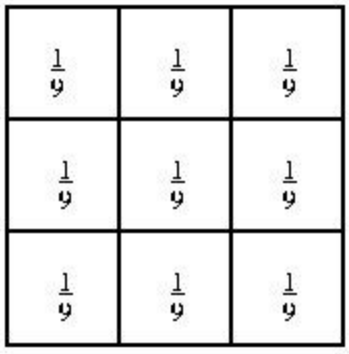
* code ini udah ada di google colab yang saya bagikan

Pada kulgram ini saya akan menjelaskan metode-metode spatial filtering dan kegunaannya, yang terdiri atas:
> Bluring Filter (Mean FIlter, Low Pass Filter)
> Sharpening Filter (High Pass Filter)
> Noise Reduction Filter (Median Filter)
> Edge Detection (Sobel Filter)

Kita mulai dari Bluring Filter,
Metode bluring filter yang akan saya jelaskan pertama adalah Mean Filter.

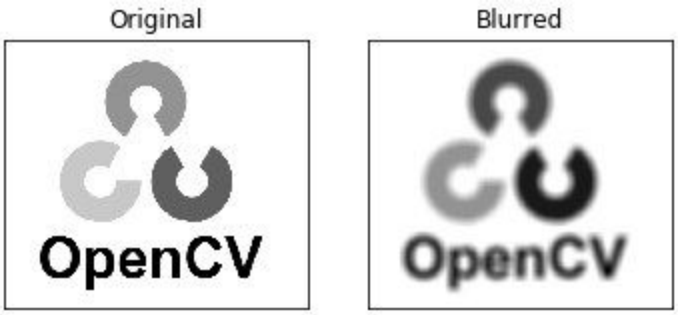
Ide mengenai mean filtering hanyalah untuk mengganti setiap nilai piksel dalam gambar dengan nilai mean (rata-rata) dari tetangganya, termasuk dirinya sendiri. Metode ini memiliki efek menghilangkan nilai piksel yang tidak mewakili lingkungan mereka. filter ini berbasi kernel, yang mewakili bentuk dan ukuran lingkungan yang akan diambil sampelnya saat menghitung rata-rata.

Seringkali kernel 3 × 3 persegi digunakan, seperti yang ditunjukkan pada Gambar dibawah, meskipun kernel yang lebih besar (mis. 5 × 5 kotak) dapat digunakan untuk menghaluskan yang lebih parah.



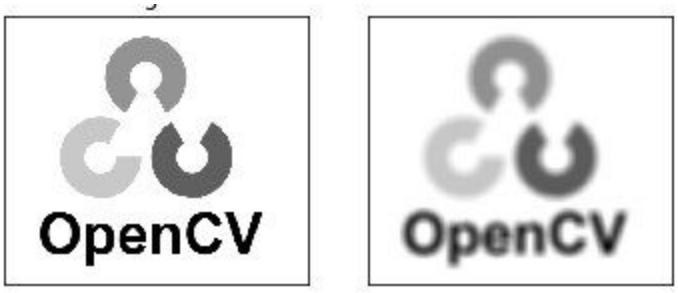
Berikut code yang digunakan untuk mean filtering,

```
# set kernel
ksize = 15
kernel = np.ones((ksize, ksize), dtype="float") * (1.0 / (ksize * ksize))

#read image
!curl -o opencv_logo.png https://i.stack.imgur.com/ez8QV.png
img = cv2.imread('opencv_logo.png', cv2.IMREAD_GRAYSCALE)

#convolve
convoleOutput = convolve(img, kernel)
opencvOutput = cv2.filter2D(img, -1, kernel)

#show
plt.subplot(121),plt.imshow(img, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(opencvOutput, cmap='gray'),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()
```



Hasilnya adalah sebagai berikut,
Fungsi Mean Filter, terdapat pada library OpenCV. Berikut adalah code nya.

```
#membaca gambar
!curl -o opencv_logo.png https://i.stack.imgur.com/ez8QV.png
img = cv2.imread('opencv_logo.png', cv2.IMREAD_GRAYSCALE)

# menambah efek blur
blur = cv2.blur(img,(15,15))

# menampilkan hasil
plt.subplot(121),plt.imshow(img, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur, cmap='gray'),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()
```



Menghasilkan gambar yang sama,
Lanjut ke metode yang kedua yaitu Low Pass Filtering.
low pass filter cenderung mempertahankan informasi frekuensi rendah dalam suatu gambar sekaligus mengurangi informasi frekuensi tinggi. Pixel di tengah diganti dengan penjumlahan piksel lainnya. Ini diulang untuk setiap piksel dalam gambar. Contohnya adalah array yang dibagi oleh kernel, seperti kernel 3 per 3 berikut:

0	+1/8	0
+1/8	+1/2	+1/8
0	+1/8	0

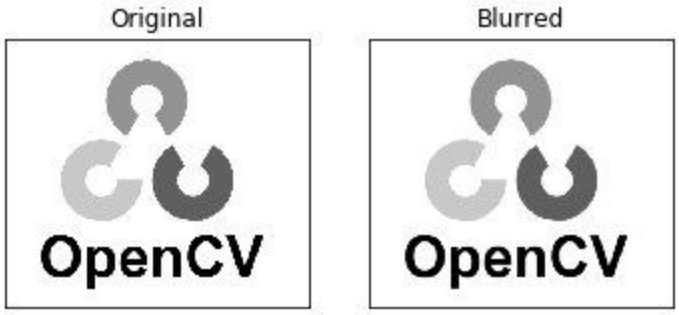
Masing-masing dari empat piksel di atas, di bawah, kiri, dan kanan tengah berkontribusi masing-masing 1/8. Ini akan memiliki efek yang lebih halus. Dengan memilih filter low-pass

yang berbeda, kita dapat memilih yang memiliki noise smoothing yang cukup, tanpa mengaburkan gambar terlalu banyak. Kita juga bisa membuat kernel lebih besar. Contoh di atas adalah 3x3 piksel dengan total sembilan. Kita bisa menggunakan 5x5 dengan mudah, atau bahkan lebih. Satu-satunya masalah dengan menggunakan kernel yang lebih besar adalah jumlah perhitungan yang dibutuhkan menjadi sangat besar.

Berikut contoh code untuk Low pass filtering.

```
# set kernel
kernel = np.array((
  [0, 1/8, 0],
  [1/8, 1/2, 1/8],
  [0, 1/8, 0]), dtype="float")

#read image
!curl -o opencv_logo.png https://i.stack.imgur.com/ez8QV.png
img = cv2.imread('opencv_logo.png', cv2.IMREAD_GRAYSCALE)

#convolve
convoleOutput = convolve(img, kernel)
opencvOutput = cv2.filter2D(img, -1, kernel)

#show
plt.subplot(121),plt.imshow(img, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(opencvOutput, cmap='gray'),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()
```

Berikut hasilnya



Disini saya menggunakan kernel ukuran 3x3. Dapat dilihat, gambar menjadi semakin halus. Penghalusan ini terlihat kecil dikarenakanukuran kernel yang saya gunakan tergolong kecil. Sayangnya untuk membuat kernel ukuran besar, diperlukan coding lagi. Namun mohon maaf saya belum sempat membuat code tersebut. Lalu, Saya, sayangnya, tidak menemukan penggunakan filter ini menggunakan Library OpenCV. Bila teman2 menemukan, mungkin bisa sharing-sharing nanti.

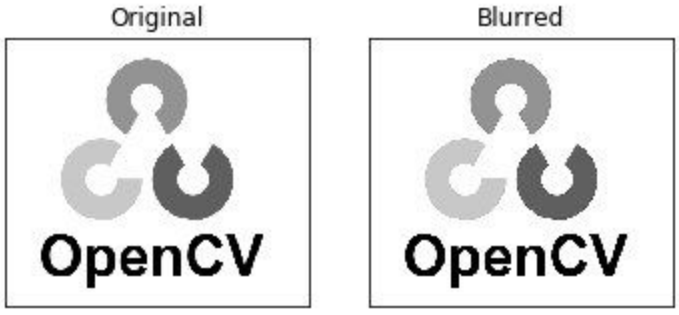
Lanjut ke metode Spatial Filter lainnya, yaitu Sharpening Filter.
Metode yang digunakan, salah satunya, adalah kebalikan dari Low Pass Filter yaitu High Pass Filter. Filter high-pass dapat digunakan untuk membuat gambar tampak lebih tajam.

filter ini menekankan detail halus pada gambar - persis kebalikan dari filter low-pass. Pemfilteran high-pass bekerja dengan cara yang persis sama dengan pemfilteran low-pass; hanya menggunakan kernel konvolusi yang berbeda. Pada contoh di bawah ini, perhatikan tanda minus untuk piksel yang berdekatan. Jika tidak ada perubahan intensitas, tidak ada yang terjadi. Tetapi jika satu pixel lebih terang dari tetangga terdekatnya, ia akan terdongkrak. Berikut Kernel yang digunakan untuk ukuran 3x3:

0	-1/4	0
-1/4	+2	-1/4
0	-1/4	0

Berikut contoh code nya:

```
[ ]  # set kernel
     kernel = np.array((
       [0, -1/4, 0],
       [-1/4, 2, -1/4],
       [0, -1/4, 0]), dtype="float")

     #read image
     !curl -o opencv_logo.png https://i.stack.imgur.com/ez8QV.png
     img = cv2.imread('opencv_logo.png', cv2.IMREAD_GRAYSCALE)

     #convolve
     convoleOutput = convolve(img, kernel)
     opencvOutput = cv2.filter2D(img, -1, kernel)

     #show
     plt.subplot(121),plt.imshow(img, cmap='gray'),plt.title('Original')
     plt.xticks([]), plt.yticks([])
     plt.subplot(122),plt.imshow(opencvOutput, cmap='gray'),plt.title('Blurred')
     plt.xticks([]), plt.yticks([])
     plt.show()
```



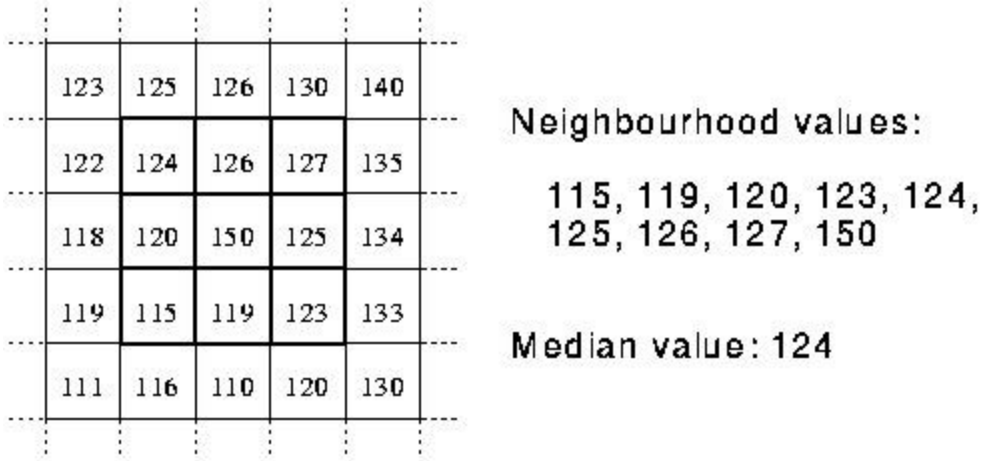
Berikut hasilnya,
Dapat dilihat, gambar tersebut terlihat semakin tajam. BIla ukuran kernelnya diperbesar, maka hasilnya akan tajam. Namun mohon maaf, saya belum sempat membuat code untuk membuat kernel dapat fleksibel.

Lanjut ke Metode Spatial Filtering lainnya, yaitu Noise Reduction Filter menggunakan Median Filter.

Filter median biasanya digunakan untuk mengurangi noise pada gambar, agak seperti filter rata-rata. Namun, sering melakukan pekerjaan yang lebih baik daripada filter mean dalam menjaga detail berguna dalam gambar. Seperti filter mean, filter median mempertimbangkan setiap piksel pada gambar secara bergantian dan melihat tetangga terdekatnya untuk memutuskan apakah itu mewakili lingkungannya atau tidak.

Alih-alih hanya mengganti nilai piksel dengan rata-rata nilai piksel tetangga, ia menggantikannya dengan median nilai-nilai tersebut. Median dihitung dengan terlebih dahulu menyortir semua nilai piksel dari nilai intensitas tetangganya ke dalam urutan numerik dan kemudian mengganti piksel yang sedang dipertimbangkan dengan nilai piksel tengah.

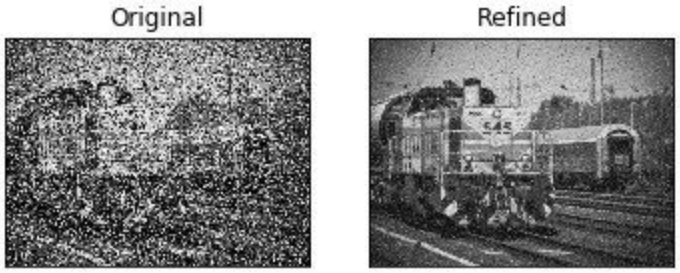
Jika nilai intensitas tetangga yang dipertimbangkan mengandung angka genap, rata-rata dari dua nilai piksel tengah. Gambar berikut menggambarkan contoh median filter



Berikut contoh code median filter menggunakan OpenCV:

```
#membaca gambar
!curl -o train.png https://i.stack.imgur.com/J13Wn.jpg
img = cv2.imread('train.png', cv2.IMREAD_GRAYSCALE)

# menambah efek blur
blur = cv2.medianBlur(img,5)

# menampilkan hasil
plt.subplot(121),plt.imshow(img, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur, cmap='gray'),plt.title('Refined')
plt.xticks([]), plt.yticks([])
plt.show()
```

Dan berikut hasilnya:



diberikan suatu gambar yang diberi noise, kemudian noise tersebut dapat terkurangi berkat median filter.

Lanjut ke metode Spatial Filtering terakhir yaitu Edge Detection.
Disini saya akan membahas Sobel Filter. Operator Sobel, kadang-kadang disebut operator Sobel-Feldman atau Sobel filter, digunakan dalam pemrosesan gambar dan visi komputer, khususnya dalam algoritma deteksi tepi di mana ia menciptakan gambar yang menekankan tepi.

Filter ini menggunakan dua kernel 3 × 3 yang dikonvolusikan dengan gambar asli untuk menghitung perkiraan turunan - satu untuk perubahan horisontal, dan satu untuk vertikal. Gx dan Gy adalah dua kernel yang masing-masing berisi perkiraan turunan vertikal dan horizontal, kernel tersebut adalah sebagai berikut:

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

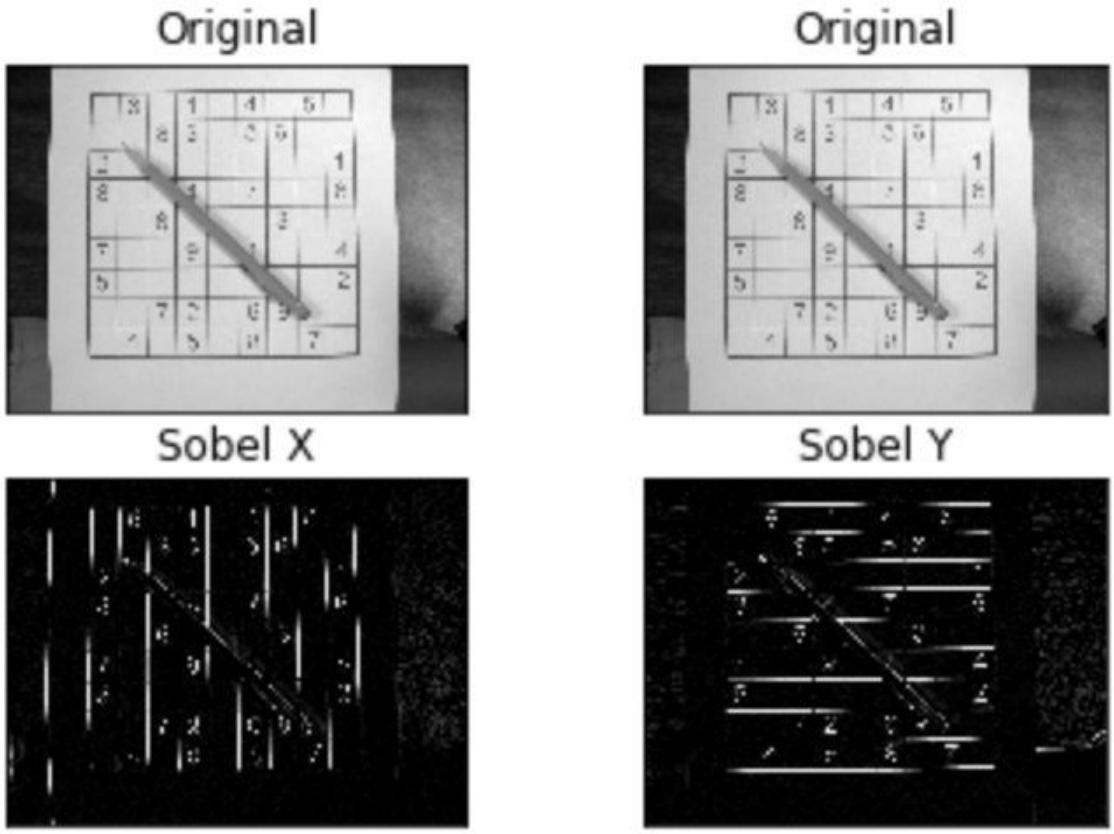
Gy

Gx digunakan untuk mengambil garis vertikal, dan Gy digunakan untuk mengambil garis horisontal,

berikut contoh code nya ;

```
# set kernel
kernelX = np.array((
  [-1, 0, 1],
  [-2, 0, 2],
  [-1, 0, 1]), dtype="float")

kernelY = np.array((
  [1, 2, 1],
  [0, 0, 0],
  [-1, -2, -1]), dtype="float")

#read image
!curl -o sudoku.png https://cdn.instructables.com/F86/PKFR/1XLEPD7QXJF/F86PKFR1XLEPD7QXJF.LARGE.jpg
img = cv2.imread('sudoku.png', cv2.IMREAD_GRAYSCALE)

#convolve
opencvOutputX = cv2.filter2D(img, -1, kernelX)
opencvOutputY = cv2.filter2D(img, -1, kernelY)

#show
plt.subplot(221),plt.imshow(img, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(222),plt.imshow(img, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(223),plt.imshow(opencvOutputX, cmap='gray'),plt.title('Sobel X')
plt.xticks([]), plt.yticks([])
plt.subplot(224),plt.imshow(opencvOutputY, cmap='gray'),plt.title('Sobel Y')
plt.xticks([]), plt.yticks([])
plt.show()
```

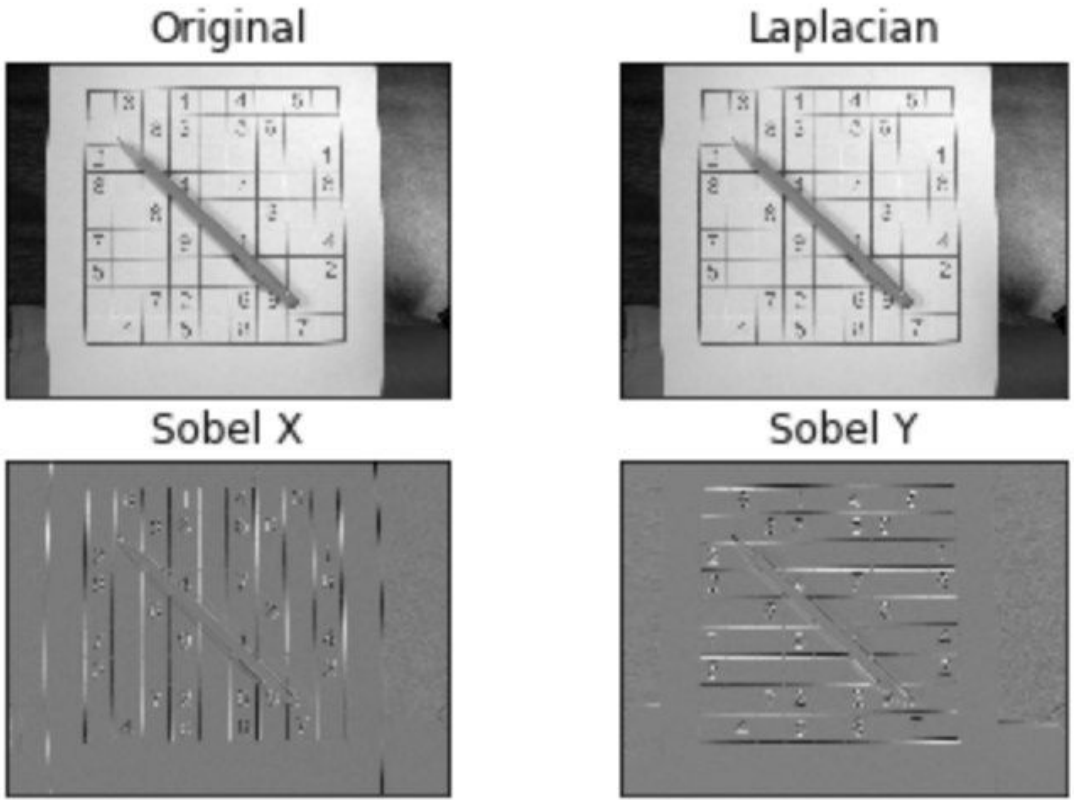
Library OpenCV telah menyediakan fungsi khusus untuk filter Sobel,



Gx digunakan untuk mengambil garis vertikal, dan Gy digunakan untuk mengambil garis horisontal, Dan berikut code Filter SObel menggunakan opencv.

```
#read image
!curl -o sudoku.png https://cdn.instructables.com/F86/PKFR/1XLEPD7QXJ
img = cv2.imread('sudoku.png', cv2.IMREAD_GRAYSCALE)

sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=3)
sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=3)

plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,2),plt.imshow(img,cmap = 'gray')
plt.title('Laplacian'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,3),plt.imshow(sobelx,cmap = 'gray')
plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,4),plt.imshow(sobely,cmap = 'gray')
plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])

plt.show()
```

Hasilnya sebagai berikut:



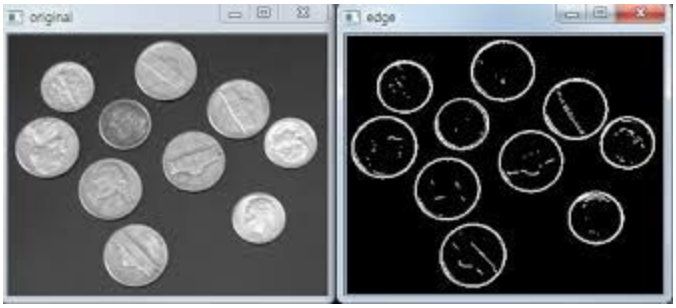
Note : Tambahan untuk belajar

Untuk blurring filter, terdapat metode lain seperti Gaussian Blur yang menggunakan Kernel Gaussian. Lalu untuk Sharpening, dapat menggunakan metode Unsharp. Untuk metode Edge detection, terdapat metode Laplacian. Karena keterbatasan waktu, silahkan temen-temen bisa meng-eksplor metode-metode tersebut. Terimakasih

Sesi tanya jawab

Dari @ bren jhonatan
saya mau tanya sedikit tentang edge detection, penasaran aja sih contoh edge detection, dalam kasus apa foto itu di butuhkan filter pakai edge detection ya?

Jawaban
Pertanyaan bagus. Edge Detection biasanya digunakan untuk mengambil informasi dari garis tepinya. Contoh saya beri gambar berikut:
Maaf saya sedang mencari beberapa tugas ketika saya kuliah dulu, tapi belum nemu, Jadi kemarin saya pernah membuat tugas menghitung panjang daun (berdasarkan panjang piksel) jadi setelah mendapatkan edge detection, nanti ada proses lagi namanya Thining.
Peng-kurusan bahasa indonesianya, dari hasil thining tersebut didapatkan garis, dan garis tersebut bisa dihitung panjangnya. Selain itu, edge detection dapat digunakan untuk object detection,



itu contohnya, dari situ bisa didapatkan mana coin dan mana bukan. Selain itu, proses ini bisa menghitung diameter dari masing2 koin tersebut

Dari @ admaja putra
Bisa di bilang setiap metode bergantung pada sifat cirta yang akan di proses ya mas, mungkin kalau di liat dari contoh metode median filtering paling gampang saya pahami karena dari contoh input dan output yang di hasilkan sdah cukup menggambarkan. Nah semisal kita menerima banyak data set gambar yang akan di proses, dengan tingkatan noise yang berbeda, pemilihan filternya nanti seperti apa ya mas?

Jawaban
Nah ini memang kelemahan median filter. Noise2 tiap gambar kan bervariasi, tapi kernel yang digunakan fixed. Ada banyak metode lainnya yang bisa digunakan untuk mengurangi noise. Beberapa diantaranya ada yang menggunakan Deep Learning, dimana nantinya bisa fleksible dalam berbagai image. Untuk penggunaan median filter ini, terdapat batasan yaitu kondisi noise dataset images dianggap mirip. Misal nih, ada dataset yang diambil dari kamera X. Kondisi noise nya tiap frame itu anggepannya sama tuh, karena menggunakan satu kamera.