

Bab 4

Filter Spasial

4.1 Pendahuluan

Sejauh ini kita telah membahas dasar-dasar Python dan modul ilmiahnya. Dalam bab ini, kita memulai perjalanan mempelajari pemrosesan gambar. Konsep pertama yang akan kita kuasai adalah penyaringan, yang merupakan inti dari kualitas gambar dan pemrosesan lebih lanjut.

Kita mengasosiasikan filter (seperti filter air) dengan penghilangan kotoran yang tidak diinginkan. Demikian pula, dalam pemrosesan gambar, filter menghilangkan kotoran yang tidak diinginkan yang mencakup noise. Dalam beberapa kasus, kotoran mungkin mengganggu secara visual dan dalam beberapa kasus dapat menyebabkan kesalahan dalam pemrosesan lebih lanjut. Beberapa filter juga digunakan untuk menghilangkan fitur tertentu dalam gambar dan menyorot yang lain. Misalnya, filter turunan pertama dan turunan kedua yang akan kita bahas digunakan untuk menentukan atau meningkatkan tepi dalam gambar.

Ada dua jenis filter: filter linear dan filter non-linear. Filter linear meliputi filter mean, Laplacian, dan Laplacian of Gaussian. Filter non-linear meliputi filter median, maksimum, minimum, Sobel, Prewitt, dan Canny.

Peningkatan citra dapat dilakukan dalam dua domain: spasial dan frekuensi. Domain spasial merupakan semua piksel dalam suatu citra. Jarak dalam citra (dalam piksel) sesuai dengan jarak nyata dalam mikrometer, inci, dst. Domain yang mencakup rentang transformasi Fourier suatu citra dikenal sebagai domain frekuensi citra. Kita mulai dengan teknik peningkatan citra dalam domain spasial. Kemudian

di dalam [Bab 7](#), "Transformasi Fourier," kita akan membahas peningkatan gambar menggunakan frekuensi atau domain Fourier.

Modul Python yang digunakan dalam bab ini adalah scikits dan scipy. Dokumentasi scipy dapat ditemukan di [\[Ilmu Pengetahuan 20c\]](#), dokumentasi scikits dapat ditemukan di [\[Ilmu Pengetahuan 20a\]](#), dan dokumentasi scipy ndimage dapat ditemukan di [\[Ilmu Pengetahuan 20d\]](#).

4.2 Penyaringan

Seperti halnya filter air yang menghilangkan kotoran, filter pemrosesan gambar menghilangkan fitur yang tidak diinginkan (seperti noise) dari gambar. Setiap filter memiliki kegunaan khusus dan dirancang untuk menghilangkan jenis noise atau untuk meningkatkan aspek tertentu dari gambar. Kami akan membahas banyak filter beserta tujuan dan pengaruhnya terhadap gambar.

Untuk penyaringan, digunakan filter atau masker. Biasanya berupa jendela persegi dua dimensi yang bergerak melintasi gambar yang hanya memengaruhi satu piksel pada satu waktu. Setiap angka dalam filter dikenal sebagai koefisien. Koefisien dalam filter menentukan efek filter dan akibatnya gambar keluaran. Mari kita pertimbangkan filter 3x3, F , diberikan dalam [Tabel 4.1](#).

TABEL 4.1:Filter berukuran 3x3.

F_1	F_2	F_3
F_4	F_5	F_6
F_7	F_8	F_9

Jika (aku_j) adalah piksel dalam gambar, kemudian sub-gambar di sekitar (aku_j) dengan dimensi yang sama dengan filter dipertimbangkan untuk penyaringan. Bagian tengah filter ditempatkan untuk tumpang tindih dengan (aku_j) . Piksel dalam sub-gambar dikalikan dengan koefisien yang sesuai dalam filter. Ini menghasilkan matriks dengan ukuran yang sama dengan filter. Matriks disederhanakan menggunakan persamaan matematika untuk mendapatkan satu nilai yang akan menggantikan

nilai piksel dalam (aku_j) dari gambar. Sifat pasti persamaan matematika bergantung pada jenis filter. Misalnya, dalam kasus filter rata-rata, nilai $F_{Saya=1}$ Bahasa Indonesia: Di mana N adalah jumlah elemen dalam filter. Citra yang difilter diperoleh dengan mengulang proses penempatan filter pada setiap piksel dalam citra, memperoleh nilai tunggal, dan mengganti nilai piksel dalam citra asli. Proses menggeser jendela filter di atas citra ini disebut konvolusi dalam domain spasial.

Mari kita perhatikan sub-gambar berikut dari gambar tersebut, $SAYA$, berpusat di (aku_j)

TABEL 4.2: Sub-gambar berukuran 3x3.

$SAYA(saya -1, j-1)$	$SAYA(saya -1, j)$	$SAYA(saya -1, j+1)$
$SAYA(aku, j -1)$	$SAYA(aku,j)$	$SAYA(aku,j+1)$
$SAYA(Saya+1, j-1)$	$SAYA(Saya+1, j)$	$SAYA(Saya+1, j+1)$

Konvolusi filter yang diberikan dalam [Tabel 4.1](#) dengan sub gambar di [Tabel 4.2](#) diberikan sebagai berikut:

$$\begin{aligned}
 SAYA_{baru}(aku_j) &= F_1 * SAYA(saya -1, j-1) + F_2 * SAYA(saya -1, j) + F_3 * SAYA(saya -1, j+1) \\
 &\quad + F_4 * SAYA(aku, j -1) + F_5 * SAYA(aku,j) + F_6 * SAYA(aku,j+1) \\
 &\quad + F_7 * SAYA(Saya+1, j-1) + F_8 * SAYA(Saya+1, j) + F_9 * SAYA(Saya+1, j+1)
 \end{aligned} \tag{4.1}$$

Di mana $SAYA_{baru}(aku_j)$ adalah nilai keluaran di lokasi (aku_j) . Proses ini harus diulang untuk setiap piksel dalam gambar. Karena filter memainkan peran penting dalam proses konvolusi, filter juga dikenal sebagai kernel konvolusi.

Operasi konvolusi harus dilakukan pada setiap piksel dalam gambar termasuk piksel pada batas gambar. Ketika filter ditempatkan pada piksel batas, sebagian filter akan berada di luar batas. Karena nilai piksel tidak ada di luar batas, nilai baru harus dibuat sebelum konvolusi. Proses pembuatan nilai piksel di luar batas ini disebut padding. Piksel yang diberi padding

dapat diasumsikan sebagai nol atau nilai konstan. Opsi padding lainnya seperti nearest neighbor atau reflect membuat piksel padding menggunakan nilai piksel dalam gambar. Dalam kasus nol, piksel padding semuanya nol. Dalam kasus konstan, piksel padding mengambil nilai tertentu. Dalam kasus reflect, piksel padding mengambil nilai baris atau kolom terakhir. Piksel padding hanya dipertimbangkan untuk konvolusi dan akan dibuang setelah konvolusi.

Mari kita perhatikan contoh untuk menunjukkan pilihan bantalan yang berbeda.[Gambar 4.1\(a\)](#) adalah gambar input 7x7 yang akan dikonvolusikan menggunakan filter 3x5 dengan pusat filter di (1 *Bahasa Indonesia:2*). Untuk menyertakan piksel batas untuk konvolusi, kami melapisi gambar dengan satu baris di atas dan satu baris di bawah serta dua kolom di kiri dan dua kolom di kanan. Secara umum, ukuran filter menentukan jumlah baris dan kolom yang akan dilapisi pada gambar.

- Tanpa bantalan: Semua piksel yang diberi bantalan diberi nilai nol ([Gambar 4.1\(b\)](#)).
- Padding konstan: Nilai konstan 5 digunakan untuk semua piksel yang diberi bantalan ([Gambar 4.1\(c\)](#)). Nilai konstan dapat dipilih berdasarkan jenis gambar yang sedang diproses.
- Tetangga terdekat: Nilai dari baris atau kolom terakhir ([Gambar 4.1\(d\)](#)) digunakan untuk bantalan.
- Mencerminkan: Nilai dari baris atau kolom terakhir ([Gambar 4.1\(e\)](#)) dipantulkan melintasi batas gambar.
- Membungkus: Dalam opsi bungkus yang diberikan di[Gambar 4.1\(f\)](#), baris (atau kolom) pertama setelah batas mengambil nilai yang sama dengan baris (atau kolom) pertama pada gambar dan seterusnya.

4.2.1 Filter Rata-rata

Dalam matematika, fungsi diklasifikasikan menjadi dua kelompok, yaitu linear dan non-linear. Fungsi f dikatakan linier jika

0	2	5	7	3	10	9
11	1	4	6	8	2	0
0	12	10	9	7	4	5
1	9	7	8	13	11	0
5	10	14	6	2	1	1
7	6	11	3	13	8	4
3	9	6	12	7	10	5

(a) Input 7x7

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	5	7	3	10	9	0	0	0
0	0	11	1	4	6	8	2	0	0	0	0
0	0	0	12	10	9	7	4	5	0	0	0
0	0	1	9	7	8	13	11	0	0	0	0
0	0	5	10	14	6	2	1	1	0	0	0
0	0	7	6	11	3	13	8	4	0	0	0
0	0	3	9	6	12	7	10	5	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

dengan itu

5	5	5	5	5	5	5	5	5	5	5	5
5	5	0	2	5	7	3	10	9	5	5	5
5	5	11	1	4	6	8	2	0	5	5	5
5	5	0	12	10	9	7	4	5	5	5	5
5	5	1	9	7	8	13	11	0	5	5	5
5	5	5	10	14	6	2	1	1	5	5	5
5	5	7	6	11	3	13	8	4	5	5	5
5	5	3	9	6	12	7	10	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5

(c) Padding dengan ac

0	0	0	2	5	7	3	10	9	9	9	10
0	0	0	2	5	7	3	10	9	9	9	9
11	11	11	1	4	6	8	2	0	0	0	0
0	0	0	12	10	9	7	4	5	5	5	5
1	1	1	9	7	8	13	11	0	0	0	0
5	5	5	10	14	6	2	1	1	1	1	1
7	7	7	6	11	3	13	8	4	4	4	4
3	3	3	9	6	12	7	10	5	5	5	5
3	3	3	3	9	6	12	7	10	5	5	5

dengan itu

2	0	0	2	5	7	3	10	9	9	9	10
1	11	11	1	4	6	8	2	0	0	2	0
12	0	0	12	10	9	7	4	5	5	4	0
9	1	1	9	7	8	13	11	0	0	11	1
10	5	5	10	14	6	2	1	1	1	1	1
6	7	7	6	11	3	13	8	4	4	8	4
9	3	3	9	6	12	7	10	5	5	10	5
9	3	3	3	9	6	12	7	10	5	5	10

(e) Padding dengan opsi refleksi.

5	10	3	9	6	12	7	10	5	3	9	10
9	10	0	2	5	7	3	10	9	0	2	0
0	2	11	1	4	6	8	2	0	11	1	1
5	4	0	12	10	9	7	4	5	0	12	0
0	11	1	9	7	8	13	11	0	1	9	1
1	1	5	10	14	6	2	1	1	5	10	5
4	8	7	6	11	3	13	8	4	7	6	4
5	10	3	9	6	12	7	10	5	3	9	5
9	10	0	2	5	7	3	10	9	0	2	0

(f) Bantalan dengan opsi bungkus.

GAMBAR 4.1:Contoh berbagai pilihan bantalan.

$$f(x+k) = \text{bahasa Indonesia} : f(x) + f(k)$$
(4.2)

Jika tidak, bersifat non-linier. Filter linier merupakan perluasan dari fungsi linier.

Contoh yang sangat baik dari filter linier adalah filter rata-rata. Koefisien filter rata-rata ([Tabel 4.1](#)) adalah angka 1. Untuk menghindari penskalaan intensitas piksel setelah penyaringan, seluruh gambar kemudian dibagi dengan jumlah piksel dalam filter; dalam kasus sub-gambar 3-kali-3, kami membaginya dengan 9.

Tidak seperti filter lain yang dibahas dalam bab ini, filter mean tidak memiliki fungsi modul `scipy.ndimage`. Namun, kita dapat menggunakan fungsi `convolve` untuk mencapai hasil yang diinginkan. Berikut ini adalah tanda tangan fungsi Python untuk `convolve`:

```
scipy.ndimage.filters.convolve(input, bobot)
```

Argumen yang diperlukan:

inputnya adalah ndarray numpy.

weights adalah ndarray yang terdiri dari koefisien 1 untuk filter rata-rata.

Argumen opsional:

mode menentukan metode untuk menangani batas array dengan padding. Pilihan yang berbeda adalah: constant, reflect, nearest, mirror, wrap. Lihat penjelasan di atas.

cval adalah nilai skalar yang ditentukan saat opsi mode bersifat konstan. Nilai default adalah 0,0.

origin adalah skalar yang menentukan origin filter. Nilai default 0 sesuai dengan origin filter.

yang titik asal (piksel referensi) berada di tengah. Dalam kasus 2D, titik asal = 0 berarti (0,0).

Pengembalian: output adalah ndarray

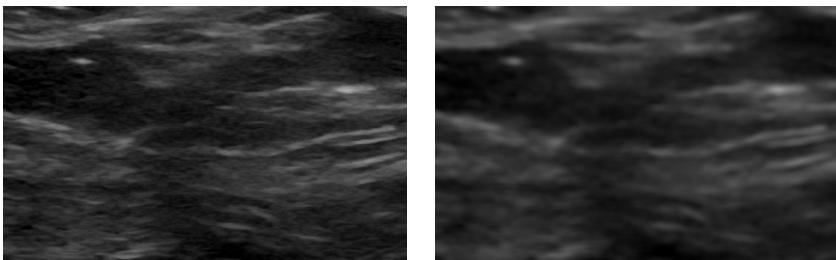
Program yang menjelaskan penggunaan mean filter diberikan di bawah ini. Filter (k) adalah array ndarray berukuran 5x5 dengan semua nilai = 1/25. Filter kemudian dikonvolusikan menggunakan fungsi "convolve" dari `scipy.ndimage.filters`.

```
impor cv2
impor numpy sebagai np
impor scipy.ndimage

# Membuka gambar menggunakan cv2.
a = cv2.imread('../Gambar/ultrasound_muscle.png')
# Mengonversi gambar ke skala abu-abu. a =
cv2.cvtColor(a, cv2.COLOR_BGR2GRAY)

# Menginisialisasi filter berukuran 5 kali 5.
# Filter dibagi 25 untuk normalisasi. k = np.ones((5,5))/
25
# melakukan konvolusi
b = scipy.ndimage.filters.convolve(a, k)
# Menulis b ke sebuah file. cv2.imwrite('../Figures/
mean_output.png', b)
```

Gambar 4.2(a) adalah gambar otot yang diambil dari USG. Perhatikan bahwa gambar tersebut mengandung noise. Filter mean berukuran 5x5 digunakan untuk menghilangkan noise. Outputnya ditunjukkan dalam Gambar 4.2(b). Filter rata-rata secara efektif menghilangkan noise tetapi dalam prosesnya mengaburkan gambar.



(a) Gambar masukan untuk filter rata-rata.

(b) Output yang dihasilkan dengan ukuran filter rata-rata (5,5).

GAMBAR 4.2:Contoh filter rata-rata.

Keuntungan dari filter rata-rata

- Menghilangkan kebisingan.
- Meningkatkan kualitas gambar secara keseluruhan, yaitu filter rata-rata mencerahkan gambar.

Kerugian dari filter rata-rata

- Dalam proses penghalusan, tepiannya menjadi kabur.
- Mengurangi resolusi spasial gambar.

Jika koefisien filter rata-rata tidak semuanya bernilai 1, maka filter tersebut adalah filter rata-rata tertimbang. Dalam filter rata-rata tertimbang, koefisien filter dikalikan dengan sub-gambar seperti pada filter tidak tertimbang. Setelah penerapan filter, gambar harus dibagi dengan bobot total untuk normalisasi.

4.2.2 Filter Median

Fungsi yang tidak memenuhi Persamaan 4.2 adalah non-linier. Filter median adalah salah satu filter non-linier yang paling populer. Jendela geser dipilih dan ditempatkan pada gambar pada posisi piksel (aku_j).

Semua nilai piksel di bawah filter dikumpulkan. Median dari nilai-nilai ini dihitung dan ditetapkan ke (*aku*) pada gambar yang difilter. Misalnya, perhatikan subgambar 3x3 dengan nilai 5, 7, 6, 10, 13, 15, 14, 19, 23. Untuk menghitung median, nilai-nilai disusun dalam urutan menaik, sehingga daftar baru adalah: 5, 6, 7, 10, 13, 14, 15, 19, dan 23. Median adalah nilai yang membagi daftar menjadi dua bagian yang sama; dalam kasus ini adalah 13. Jadi piksel (*aku*) akan ditetapkan 13 pada gambar yang difilter. Filter median paling umum digunakan untuk menghilangkan derau salt-and-pepper dan derau impuls. Derau salt-and-pepper dicirikan oleh bintik-bintik hitam dan putih yang terdistribusi secara acak pada suatu gambar.

Berikut ini adalah fungsi Python untuk filter median:

```
scipy.ndimage.filters.median_filter(input, ukuran=Tidak Ada,  
footprint=Tidak ada, mode='pantulkan', cval=0.0, asal=0)
```

Argumen yang diperlukan:

input adalah gambar masukan sebagai ndarray.

Argumen opsional:

Ukuran dapat berupa skalar atau tupel. Misalnya, jika gambarnya 2D, ukuran = 5 menyiratkan bahwa filter 5-kali-5 dipertimbangkan. Atau, ukuran juga dapat ditentukan sebagai ukuran=(5,5).

footprint adalah array boolean dengan dimensi yang sama dengan ukuran kecuali ditentukan lain. Piksel dalam gambar input yang sesuai dengan titik-titik pada footprint dengan nilai benar dipertimbangkan untuk penyaringan.

mode menentukan metode untuk menangani batas array dengan padding. Opsinya adalah: konstan, pantulkan, terdekat, cerminkan, bungkus.

Pengembalian: gambar keluaran sebagai ndarray.

Kode Python untuk filter median diberikan di bawah ini:

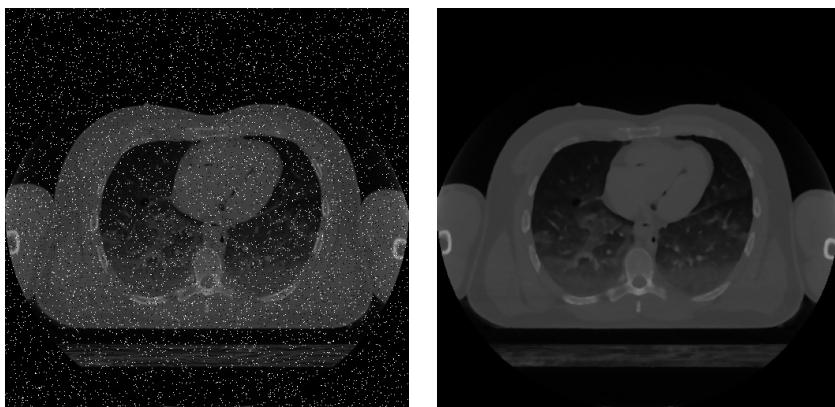
```
impor cv2
impor scipy.ndimage

# Membuka gambar dan mengubahnya menjadi skala abu-abu. a = cv2.imread('..//Figures/ct_saltandpepper.png')
# Mengonversi gambar ke skala abu-abu. a =
cv2.cvtColor(a, cv2.COLOR_BGR2GRAY)
# Melakukan filter median.
b = scipy.ndimage.filters.median_filter(a, ukuran=5)
# Menyimpan b sebagai median_output.png di folder
Gambar cv2.imwrite('..//Figures/median_output.png', b)
```

Pada kode di atas, *ukuran*=5 mewakili filter (topeng) berukuran 5x5. Gambar di[Gambar 4.3\(a\)](#) adalah irisan CT abdomen dengan noise salt-and-pepper. Gambar dibaca menggunakan 'cv2.imread' dan ndarray yang dikembalikan diteruskan ke fungsi filter median. Output dari fungsi filter median disimpan sebagai file 'png'. Gambar output ditampilkan dalam [Gambar 4.3\(b\)](#). Filter median secara efisien menghilangkan noise garam dan merica.

4.2.3 Filter Maksimal

Filter ini meningkatkan titik terang dalam gambar. Dalam filter ini, nilai maksimum dalam sub-gambar menggantikan nilai pada (*aku*). Fungsi Python untuk filter maksimum memiliki argumen yang sama dengan filter median yang dibahas di atas. Kode Python untuk filter maksimum diberikan di bawah ini.



(a) Gambar masukan untuk filter median.

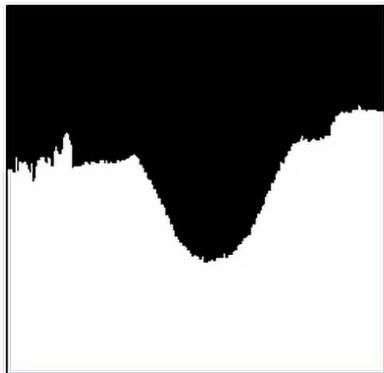
(b) Output yang dihasilkan dengan ukuran filter = (5,5).

GAMBAR 4.3:Contoh filter median.

```
impor scipy.misc  
impor scipy.ndimage  
dari scipy.misc.pilutil impor Gambar
```

```
# membuka gambar dan mengubahnya menjadi skala  
abu-abu a = Image.open('..../Figures/wave.png').convert('L')  
# melakukan filter maksimum  
b = scipy.ndimage.filters.maximum_filter(a, ukuran=5)  
# b diubah dari ndarray menjadi gambar b =  
scipy.misc.toimage(b)  
b.simpan('..../Gambar/maxo.png')
```

Gambar di[Gambar 4.4\(a\)](#) adalah citra masukan untuk filter maks. Citra masukan memiliki batas hitam tipis di sebelah kiri, kanan, dan bawah. Setelah penerapan filter maks, piksel putih telah tumbuh dan karenanya tepi tipis pada citra masukan digantikan oleh piksel putih pada citra keluaran seperti yang ditunjukkan pada[Gambar 4.4\(b\)](#).



(a) Gambar masukan untuk filter maks.



(b) Gambar keluaran filter maks.

GAMBAR 4.4:Contoh filter maks.

4.2.4 Filter Min

Filter ini digunakan untuk meningkatkan titik tergelap dalam gambar.

Dalam filter ini, nilai minimum sub-gambar menggantikan nilai pada (*aku*).

Fungsi Python untuk filter minimum memiliki argumen yang sama dengan filter median yang dibahas di atas. Kode Python untuk filter min diberikan di bawah ini.

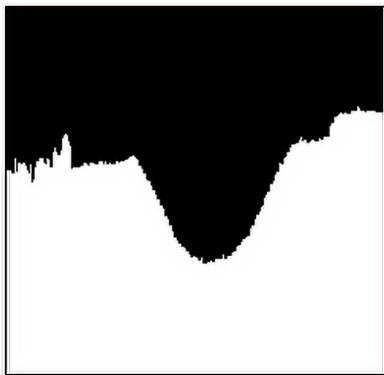
```
impor cv2
impor scipy.ndimage

# membuka gambar dan mengubahnya menjadi skala abu-abu
a = cv2.imread('..//Figures/wave.png')

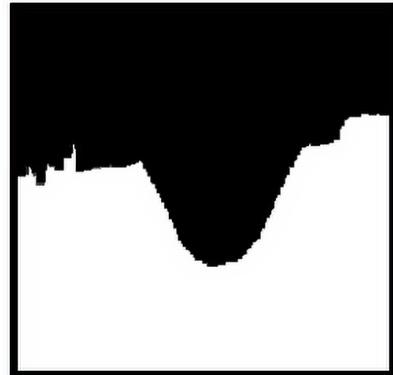
# melakukan filter minimum
b = scipy.ndimage.filters.minimum_filter(a, ukuran=5)
# menyimpan b sebagai mino.png
cv2.imwrite('..//Figures/mino.png', b)
```

Setelah penerapan filter min ke gambar input diAngka
4.5(a), gambar masukan ini
piksel hitam tebal

n di dalam



(a) Gambar masukan untuk filter min.



(b) Gambar keluaran filter min.

GAMBAR 4.5:Contoh filter min.

4.3 Deteksi Tepi menggunakan Derivatif

Tepi adalah sekumpulan titik dalam gambar yang intensitasnya berubah antara satu sisi titik tersebut dengan sisi lainnya. Dari kalkulus, kita tahu bahwa perubahan intensitas dapat diukur dengan menggunakan turunan pertama atau kedua. Pertama, mari kita pelajari bagaimana perubahan intensitas memengaruhi turunan pertama dan kedua dengan mempertimbangkan gambar sederhana dan profilnya yang sesuai. Metode ini akan menjadi dasar penggunaan filter turunan pertama dan kedua untuk deteksi tepi. Pembaca yang tertarik juga dapat merujuk ke [MH80] Bahasa Indonesia: [Mar72] Bahasa Indonesia: [PK91] Dan [Rob77].

Gambar 4.6(a) adalah citra masukan dalam skala abu-abu. Sisi kiri citra berwarna gelap sedangkan sisi kanan berwarna terang. Saat melintasi dari kiri ke kanan, di persimpangan antara dua wilayah, intensitas piksel

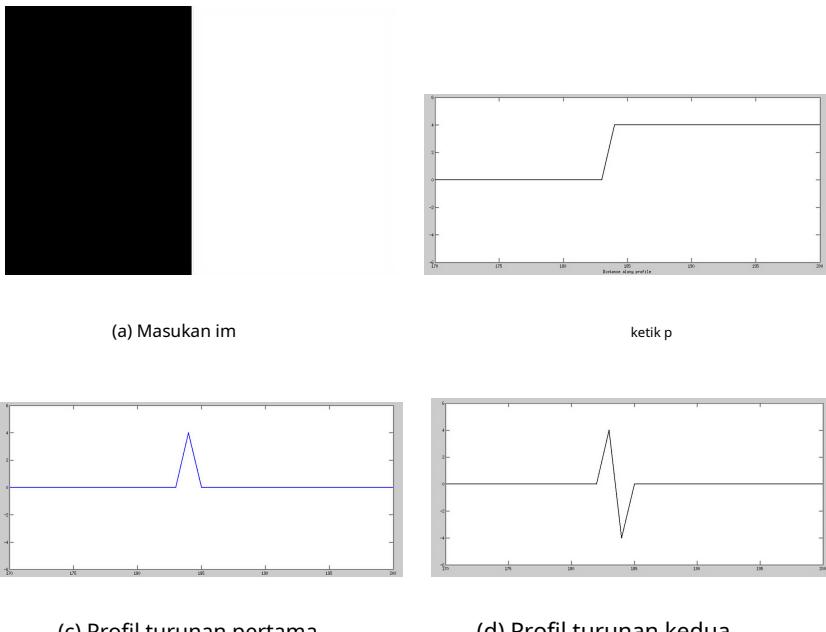
berubah dari gelap ke terang. [Gambar 4.6\(b\)](#) adalah profil intensitas di seluruh penampang horizontal gambar masukan. Perhatikan bahwa pada titik transisi dari daerah gelap ke daerah terang, ada perubahan intensitas pada profil. Jika tidak, intensitasnya konstan di daerah gelap dan terang. Untuk kejelasan, hanya daerah di sekitar titik transisi yang ditampilkan dalam profil intensitas ([Gambar 4.6\(b\)](#)), turunan pertama ([Gambar 4.6\(c\)](#)), dan turunan kedua ([Gambar 4.6\(d\)](#)) profil. Di wilayah transisi, karena profil intensitas meningkat, turunan pertama bernilai positif, sementara bernilai nol di wilayah gelap dan terang. Turunan pertama memiliki nilai maksimum atau puncak di tepi. Karena turunan pertama meningkat sebelum tepi, turunan kedua bernilai positif sebelum tepi. Demikian pula, karena turunan pertama menurun setelah tepi, turunan kedua bernilai negatif setelah tepi. Selain itu, turunan kedua bernilai nol di wilayah gelap dan terang karena turunan pertama yang sesuai bernilai nol. Di tepi, turunan kedua bernilai nol. Fenomena turunan kedua yang mengubah tanda dari positif sebelum tepi menjadi negatif setelah tepi atau sebaliknya dikenal sebagai zero-crossing, karena mengambil nilai nol di tepi. Gambar masukan disimulasikan di komputer dan tidak memiliki noise. Namun, gambar yang diperoleh akan memiliki noise yang dapat memengaruhi deteksi zero-crossing. Selain itu, jika intensitas berubah dengan cepat di profil, tepi palsu akan terdeteksi oleh zero-crossing. Untuk mencegah masalah karena noise atau intensitas yang berubah dengan cepat, gambar diproses terlebih dahulu sebelum penerapan filter turunan kedua.

4.3.1 Filter Turunan Pertama

Suatu gambar bukanlah suatu fungsi kontinu dan karenanya turunannya dihitung menggunakan pendekatan diskrit dan bukan menggunakan fungsi. Untuk tujuan pembelajaran, mari kita lihat definisi gradien fungsi kontinu dan kemudian memperluasnya ke kasus diskrit. Jika $f(x, y)$ adalah fungsi kontinu, maka gradien dari f sebagai vektor diberikan oleh

$$\nabla f = \begin{bmatrix} F_x \\ F_y \end{bmatrix} \quad (4.3)$$

Fkamu



GAMBAR 4.6: Suatu contoh dari zero-crossing.

Di mana $F_x = \frac{\partial f}{\partial x}$ dikenal sebagai turunan parsial dari f mengenai X , (perubahan f sepanjang arah horizontal) dan $F_{kamu} = \frac{\partial f}{\partial y}$ diketahui sebagai turunan parsial dari f mengenai $kamu$, (perubahan f sepanjang arah vertikal). Untuk rincian lebih lanjut, lihat [Sch04]. Besarnya gradien adalah besaran skalar dan diberikan oleh

$$|\nabla f| = \sqrt{F_x^2 + F_{kamu}^2} \quad (4.4)$$

Di mana $Bahasa Indonesia: Bahasa Indonesia: z$ adalah norma dari z .

Untuk tujuan komputasi, kami akan menggunakan versi sederhana dari gradien yang diberikan oleh Persamaan 4.5 dan sudut yang diberikan oleh Persamaan 4.6.

$$|\nabla f| = \sqrt{F_x^2 + F_{kamu}^2} \quad (4.5)$$

$$\theta = \tan^{-1} \frac{F_{kamu}}{F_x} \quad (4.6)$$

4.3.1.1 Saringan Sobel

Salah satu filter turunan pertama yang paling populer adalah filter Sobel. Filter atau topeng Sobel digunakan untuk menemukan tepi horizontal dan vertikal seperti yang diberikan dalam [Tabel 4.3](#).

TABEL 4.3: Masker Sobel untuk tepi horizontal dan vertikal.

- 1	- 2	- 1
angka 0	angka 0	angka 0
1	2	1

- 1	angka 0	1
- 2	angka 0	2
- 1	angka 0	1

Untuk memahami bagaimana penyaringan dilakukan, mari kita pertimbangkan subgambar berukuran 3-kali-3 yang diberikan dalam [Tabel 4.4](#) dan kalikan subgambar dengan topeng Sobel horizontal dan vertikal. Output yang sesuai diberikan dalam [Tabel 4.5](#).

TABEL 4.4: Subgambar berukuran 3x3.

F_1	F_2	F_3
F_4	F_5	F_6
F_7	F_8	F_9

TABEL 4.5: Output setelah mengalikan sub-gambar dengan masker Sobel.

$- f(Bahasa Indonesia)$	$2F_2$	$- f(Bahasa Indonesia)$
angka 0	angka 0	angka 0
F_7	$2F_8$	F_9

$- f(Bahasa Indonesia)$	F_3
$- 2F_4$	$2F_6$
$- f(Bahasa Indonesia)$	F_8

Sejak F_x adalah turunan parsial dari f di dalam x arah, yang merupakan perubahan f sepanjang arah horizontal, turunan parsial dapat diperoleh dengan mengambil perbedaan antara baris ketiga dan baris pertama di topeng horizontal, jadi $F_x = (F_7 + 2F_8 + F_9) + (-f(Bahasa Indonesia)) \cdot 2F_2 - F_1$. Juga, F_{kamu} adalah turunan parsial dari f di dalam $kamu$ arah, yang merupakan perubahan f dalam arah vertikal; turunan parsial dapat diperoleh dengan mengambil perbedaan antara kolom ketiga dan kolom pertama dalam topeng vertikal, jadi $F_{kamu} = (F_3 + 2F_6 + F_9) + (-f(Bahasa Indonesia)) \cdot 2F_4 - F_7$.

Menggunakan F_x dan F_y , gradien diskrit pada F_5 (Persamaan 4.7) dapat dihitung.

/f. Bahasa Indonesia = Bahasa Indonesia; f₁+2f₂+f₃-f₄-f₅ (Bahasa Indonesia); Bahasa Indonesia; + /f₁+2f₂+f₃-f₄-f₅ (Bahasa Indonesia); -2f₂-f₅ (Bahasa Indonesia); Bahasa Indonesia;

Fitur penting dari filter Sobel adalah:

- Jumlah koefisien pada gambar topeng adalah 0. Ini berarti bahwa piksel dengan skala abu-abu konstan tidak terpengaruh oleh filter turunan.
- Efek samping dari filter turunan adalah terciptanya noise tambahan. Oleh karena itu, koefisien +2 dan -2 digunakan pada gambar topeng untuk menghasilkan penghalusan.

Berikut ini adalah fungsi Python untuk filter Sobel:

```
scipy.ndimage.sobel(gambar)
```

Argumen yang diperlukan:

gambar adalah ndarray dengan satu atau tiga saluran.

Pengembalian: output adalah ndarray.

Kode Python untuk filter Sobel diberikan di bawah ini.

```
impor cv2  
dari scipy impor ndimage
```

```
# Membuka gambar.  
a = cv2.imread('..Angka/cir.png')  
# Mengonversi a ke skala abu-abu.  
a = cv2.cvtColor(a, cv2.WARNA_BGR2GRAY)
```

```
# Melakukan filter Sobel. b =
ndimage.sobel(a)
# Menabung b.
cv2.imwrite('../Gambar/sobel_cir.png', b)
```

Seperti yang dapat dilihat dalam kode, gambar 'cir.png' dibaca menggunakan cv2.imread. Ndarray 'a' kemudian diteruskan ke fungsi scipy.ndimage.sobel untuk menghasilkan gambar Sobel edge enhanced yang kemudian ditulis ke file.

4.3.1.2 Filter Pra-Witt

Filter turunan pertama populer lainnya adalah Prewitt [Sebelumnya70] Masker untuk filter Prewitt diberikan dalam [Tabel 4.6](#).

TABEL 4.6:Masker Prewitt untuk tepi horizontal dan vertikal.

- 1	- 1	- 1
angka 0	angka 0	angka 0
1	1	1

- 1	angka 0	1
- 1	angka 0	1
- 1	angka 0	1

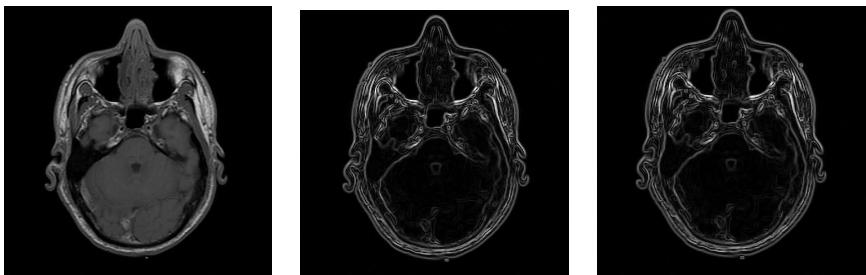
Seperti halnya filter Sobel, jumlah koefisien dalam Prewitt juga 0. Oleh karena itu, filter ini tidak memengaruhi piksel dengan skala abu-abu konstan. Namun, filter ini tidak mengurangi noise seperti filter Sobel.

Untuk Prewitt, argumen fungsi Python mirip dengan argumen fungsi Sobel.

Mari kita perhatikan sebuah contoh untuk mengilustrasikan efek penyaringan gambar menggunakan Sobel dan Prewitt. Gambar di [Gambar 4.7\(a\)](#) adalah potongan CT tengkorak manusia di dekat daerah hidung. Output dari filter Sobel dan Prewitt diberikan dalam [Gambar 4.7\(b\)](#) Dan [4.7\(c\)](#). Kedua filter berhasil membuat gambar tepi.

Filter Sobel dan Prewitt yang dimodifikasi sedikit dapat digunakan untuk mendeteksi satu atau beberapa jenis tepi. Filter Sobel dan Prewitt untuk mendeteksi tepi diagonal diberikan dalam [Tabel 4.7](#) Dan [4.8](#).

Untuk mendeteksi tepi vertikal dan horizontal untuk filter Sobel dan Prewitt, kami akan menggunakan filter dari modul skimage.



(a) Potongan melintang tengkorak manusia.

(b) Keluaran Sobel.

(c) Keluaran Prewitt.

GAMBAR 4.7:Contoh untuk Sobel dan Prewitt.

TABEL 4.7:Masker Sobel untuk tepi diagonal.

angka 0	1	2
- 1	angka 0	1
- 2	- 1	angka 0

- 2	- 1	angka 0
- 1	angka 0	1
angka 0	1	2

TABEL 4.8:Masker Prewitt untuk tepi diagonal.

angka 0	1	1
- 1	angka 0	1
- 1	- 1	angka 0

- 1	- 1	angka 0
- 1	angka 0	1
angka 0	1	1

- Fungsi filter.sobel v menghitung tepi vertikal menggunakan filter Sobel.
- Fungsi filter.sobel h menghitung tepi horizontal menggunakan filter Sobel.
- Fungsi filter.prewitt v menghitung tepi vertikal menggunakan filter Prewitt.
- Fungsi filter.prewitt h menghitung tepi horizontal menggunakan filter Prewitt.

Misalnya, untuk deteksi tepi vertikal, gunakan prewitt.v dan definisi fungsi Python adalah:

dari skimage impor filter

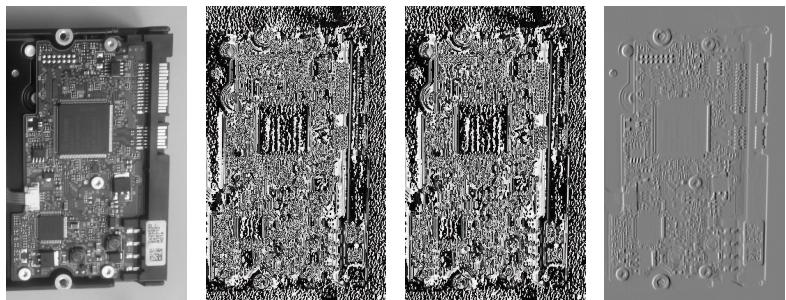
```
# Input ke filter.prewitt_v harus berupa array
numpy.filters.prewitt_v(image)
```

Gambar 4.8 adalah contoh pendektsian tepi horizontal dan vertikal menggunakan filter Sobel dan Prewitt. Filter vertikal Sobel dan Prewitt telah menyempurnakan semua tepi vertikal, sedangkan filter horizontal yang sesuai menyempurnakan tepi horizontal dan filter Sobel dan Prewitt biasa menyempurnakan semua tepi.

4.3.1.3 Filter Cerdik

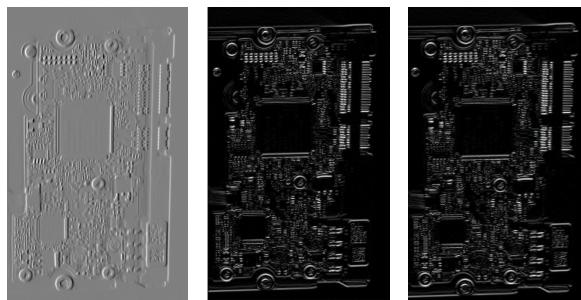
Filter populer lainnya untuk deteksi tepi adalah filter Canny atau detektor tepi Canny[Can86] Filter ini menggunakan tiga parameter untuk mendekksi tepi. Parameter pertama adalah deviasi standar, k_1 , untuk filter Gaussian. Parameter kedua dan ketiga adalah nilai ambang batas, T_1 Dan T_2 . Filter Canny dapat dijelaskan dengan langkah-langkah berikut:

1. Filter Gaussian digunakan pada gambar untuk penghalusan.
2. Properti penting dari piksel tepi adalah piksel tersebut akan memiliki besaran gradien maksimum dalam arah gradien. Jadi, untuk setiap piksel, besaran gradien (berikan) ∇ pada Persamaan 4.5 dan arah yang sesuai, $\theta = \text{coklat kecokelatan-1}$ F_x *F kamu Balas Saya* dihitung.
3. Pada titik tepi, turunan pertama akan memiliki nilai minimum atau maksimum. Ini berarti bahwa besarnya (nilai absolut) gradien gambar pada titik tepi adalah maksimum. Kita akan menyebut titik-titik ini sebagai piksel punggungan. Untuk mengidentifikasi titik tepi dan menghilangkan yang lain, hanya puncak punggungan yang dipertahankan dan piksel lainnya diberi nilai nol. Proses ini dikenal sebagai penekanan non-maksimal.



(a) Masukan gambar.

(b) Keluaran dari Sobel.

tp
tt.TPU
berdetak kencang

(e) Keluaran vertikal Prewitt.

(f) Keluaran dari horisontal Sobel.

(g) Keluaran dari horisontal Prewitt.

GAMBAR 4.8:Keluaran dari filter Sobel dan Prewitt vertikal, horizontal, dan reguler.

4. Dua ambang batas, ambang batas rendah dan ambang batas tinggi, kemudian digunakan untuk membatasi tepian. Nilai piksel tepian membantu mengklasifikasikan piksel tepian menjadi lemah dan kuat. Piksel tepian dengan nilai lebih besar dari ambang batas tinggi diklasifikasikan sebagai piksel tepian kuat, sedangkan piksel tepian antara ambang batas rendah dan ambang batas tinggi disebut piksel tepian lemah.

5. Pada langkah terakhir, piksel tepi lemah dihubungkan 8 dengan piksel tepi kuat.

Fungsi Python yang digunakan untuk filter Canny adalah:

cv2.Canny(gambar)

Argumen yang diperlukan:

input adalah gambar input sebagai ndarray

Pengembalian: output adalah ndarray.

Kode Python untuk filter Canny diberikan di bawah ini. Kode tersebut tidak memerlukan banyak penjelasan.

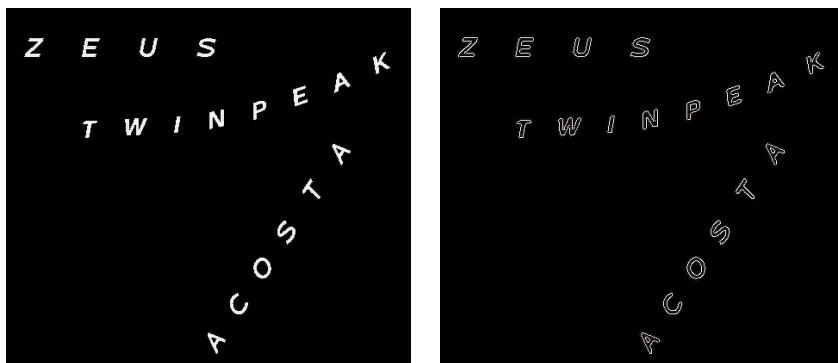
```
impor cv2
```

```
# Membuka gambar.  
a = cv2.imread('..../Gambar/maps1.png')  
# Melakukan filter tepi Canny. b =  
cv2.Canny(a, 100, 200)  
# Menabung b.  
cv2.imwrite('..../Gambar/canny_output.png', b)
```

Gambar 4.9(a) adalah peta simulasi yang terdiri dari nama-nama fitur geografis Antartika. Filter tepi Canny digunakan pada gambar masukan ini untuk mendapatkan hanya tepi huruf seperti yang ditunjukkan pada Gambar 4.9(b). Perhatikan bahwa tepi karakter ditandai dengan jelas pada output.

4.3.2 Filter Turunan Kedua

Seperti yang ditunjukkan oleh namanya, pada filter turunan kedua, turunan kedua dihitung untuk menentukan tepinya. Karena memerlukan penghitungan turunan dari gambar turunan, maka secara komputasi lebih mahal dibandingkan dengan filter turunan pertama.



(a) Gambar masukan untuk filter Canny.

(b) Keluaran filter Canny.

GAMBAR 4.9:Contoh filter Canny.

4.3.2.1 Filter Laplacian

Salah satu filter turunan kedua yang paling populer adalah Laplacian. Laplacian dari fungsi kontinu diberikan oleh:

$$\nabla^2 F = \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2}$$

Di mana $\nabla^2 F$ adalah turunan parsial kedua dari F di dalam x arah perwakilan membentuk perubahan $\frac{\partial f}{\partial x}$ sepanjang arah horizontal dan $\frac{\partial^2 F}{\partial y^2}$ adalah turunan parsial kedua dari F di dalam y arah perwakilan membentuk perubahan $\frac{\partial f}{\partial y}$ sepanjang arah vertikal. Untuk detail lebih lanjut, lihat [Eva10] Dan [GT01]. Laplacian diskrit yang digunakan untuk pemrosesan gambar memiliki beberapa versi. Masker Laplacian yang paling banyak digunakan diberikan dalam Tabel 4.9.

TABEL 4.9:Topeng Laplacian.

angka	1	angka
- 1	4	- 1
angka	- 1	angka

- 1	- 1	- 1
- 1	8	1
- 1	- 1	- 1

Fungsi Python yang digunakan untuk Laplacian beserta argumennya adalah sebagai berikut:

```
scipy.ndimage.filters.laplace(input, output=Tidak Ada,  
mode='pantulkan', cval=0.0)
```

Argumen yang diperlukan:

input adalah gambar input sebagai ndarray

Argumen opsional:

mode menentukan metode untuk menangani batas array dengan padding. Pilihan yang berbeda adalah: konstan, pantulkan, terdekat, cermin, bungkus.

cval adalah nilai skalar yang ditentukan saat opsi untuk mode bersifat konstan. Nilai default adalah 0,0.

origin adalah skalar yang menentukan origin filter. Nilai default 0 sesuai dengan filter yang origin (piksel referensi)-nya berada di tengah. Dalam kasus 2D, origin = 0 berarti (0,0).

Pengembalian: output adalah ndarray

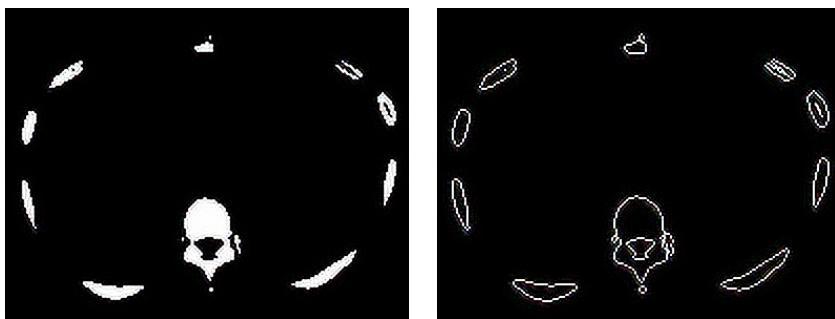
Kode Python untuk filter Laplacian diberikan di bawah ini. Laplacian dipanggil menggunakan fungsi laplace scipy beserta mode opsional untuk menangani batas array.

```
impor cv2  
impor scipy.ndimage
```

```
# Membuka gambar.
```

```
a = cv2.imread('..../Figur/imagefor_laplacian.png')  
# Melakukan filter Laplacian.  
b = scipy.ndimage.filters.laplace(a,mode='pantulkan')  
cv2.imwrite('..../Figures/laplacian_new.png',b)
```

Gambar hitam putih di[Gambar 4.10\(a\)](#) adalah irisan CT yang tersegmentasi dari tubuh manusia di sepanjang tulang rusuk. Berbagai gumpalan dalam gambar adalah tulang rusuk. **tepi artefak.**

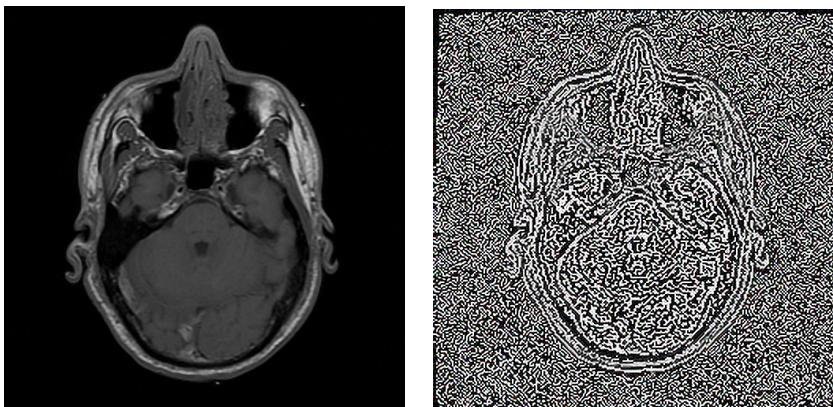


(a) Gambar masukan untuk Laplacian

(b) Keluaran Laplacian

GAMBAR 4.10:Contoh filter Laplacian.

Seperti yang dibahas sebelumnya, filter turunan menambahkan noise ke gambar. Efeknya akan semakin kuat ketika gambar turunan pertama dibedakan lagi (untuk memperoleh turunan kedua) seperti pada kasus filter turunan kedua. [Gambar 4.11](#) menampilkan efek ini. Gambar di[Gambar 4.11\(a\)](#) adalah gambar MRI dari pemindaian otak. Karena ada beberapa sisi pada gambar masukan, filter Laplacian melakukan segmentasi berlebihan pada objek (menciptakan banyak sisi) seperti yang terlihat pada gambar keluaran,[Gambar 4.11\(b\)](#) Hal ini menghasilkan gambar yang bising tanpa tepi yang jelas.



(a) Gambar masukan

(b) Gambar keluaran

GAMBAR 4.11:Contoh lain dari filter Laplacian.

4.3.2.2 Filter Laplacian atau Gaussian

Untuk mengimbangi efek noise dari Laplacian, fungsi penghalusan, Gaussian, digunakan bersama dengan Laplacian. Sementara Laplacian menghitung zero-crossing dan menentukan tepinya, Gaussian menghaluskan noise yang disebabkan oleh turunan kedua.

Fungsi Gaussian diberikan oleh

$$G(r^2) = \frac{e^{-r^2}}{\pi R^2} \quad (4.8)$$

Di mana $R^2 = x^2 + y^2$ dan k kita adalah simpangan baku. Konvolusi gambar dengan Gaussian akan menghasilkan penghalusan gambar. k ita menentukan besarnya penghalusan. Jika k besar maka akan ada lebih banyak penghalusan, yang menyebabkan tepi tajam menjadi kabur. Nilai yang lebih kecil k kita menghasilkan lebih sedikit penghalusan.

Konvolusi Laplacian dengan Gaussian dikenal sebagai Laplacian dari Gaussian dan dilambangkan dengan LoG. Karena Laplacian adalah turunan kedua, ekspresi LoG dapat diperoleh dengan mencari turunan kedua dari G mengenai R , yang menghasilkan

$$\frac{(R_2 - \sigma_2)}{\sqrt{2k\pi^2}}$$
 DQLR Bahasa Indonesia: Bahasa Inggris: $\frac{1}{\sqrt{2k\pi^2}}$
(4.9)

Masker LoG atau filter berukuran 5-kali-5 diberikan dalam [Tabel 4.10](#).

TABEL 4.10:Laplacian dari topeng Gaussian

angka	angka	- 1	angka	angka
angka	- 1	- 2	- 1	angka
- 1	- 2	16	- 2	- 1
angka	- 1	- 2	- 1	angka
angka	angka	- 1	angka	angka

Berikut ini adalah fungsi Python untuk LoG:

```
scipy.ndimage.filters.gaussian_laplace(masukan,
                                         sigma, keluaran=Tidak Ada, mode='pantulkan', cval=0.0)
```

Argumen yang diperlukan:

input adalah gambar masukan sebagai ndarray.

sigma nilai titik mengambang adalah deviasi standar Gaussian.

Pengembalian: output adalah ndarray

Kode Python di bawah ini menunjukkan implementasi filter LoG. Filter dipanggil menggunakan fungsi Gaussian Laplace dengan sigma 1.

```
impor cv2
impor scipy.ndimage

# Membuka gambar.
a = cv2.imread('..../Gambar/vhuman_t1.png')
# Melakukan Laplacian dari Gaussian.
```

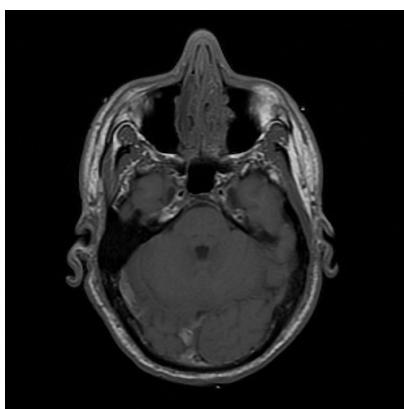
```
b = scipy.ndimage.filters.gaussian_laplace(a, sigma=1,  
    mode='pantulkan')  
cv2.imwrite('../Gambar/log_vh1.png', b)
```

Gambar 4.12(a) adalah gambar masukan dan Gambar 4.12(b) adalah output setelah penerapan LoG. Filter LoG mampu menentukan tepi lebih akurat dibandingkan dengan Laplacian saja (Gambar 4.11(b)). Namun, intensitas latar depan yang tidak seragam telah menyebabkan terbentuknya gumpalan (sekelompok piksel yang terhubung).

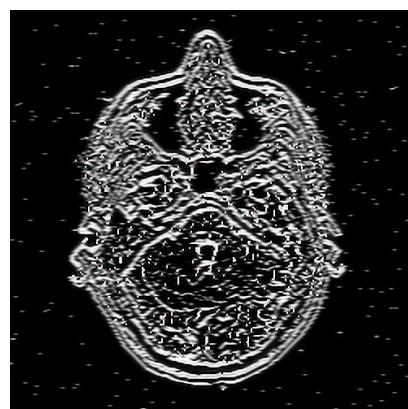
Kerugian utama LoG adalah biaya komputasi karena dua operasi, Gaussian diikuti oleh Laplacian, harus dilakukan. Meskipun LoG melakukan segmentasi objek dari latar belakang, ia melakukan segmentasi berlebihan pada tepi efek spaghetti)

kamar mandi

4.12



(a) Gambar masukan untuk LoG



(b) Keluaran filter LoG

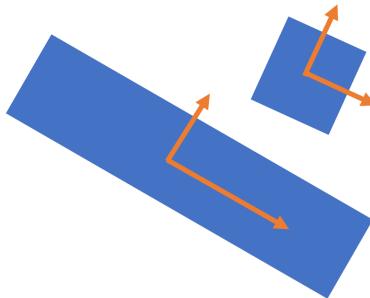
GAMBAR 4.12: Contoh LoG.

4.4 Filter Pendeksi Bentuk

4.4.1 Filter Prancis

Filter Frangi^[AFFV98]digunakan untuk mendeksi objek seperti bejana dalam gambar. Kita akan memulai pembahasan dengan ide dasar filter Frangi sebelum membahas matematika di baliknya. Gambar 4.4.1 berisi dua objek. Salah satu objek memanjang dalam satu arah tetapi tidak pada arah yang lain, sedangkan objek kedua hampir berbentuk persegi. Panah ortogonal digambar agar proporsional dengan panjang sepanjang arah tertentu. Perbedaan geometris kualitatif ini dapat diukur dengan menemukan nilai eigen untuk kedua objek ini. Untuk objek yang memanjang, nilai eigen akan lebih besar ke arah panah yang lebih panjang dan lebih kecil ke arah panah yang lebih kecil. Di sisi lain, untuk objek persegi, nilai eigen sepanjang arah panah yang lebih panjang mirip dengan nilai eigen sepanjang arah panah yang lebih panjang.

filter menghitung nilai eigen o
bukannya menghitung eigen



GAMBAR 4.13: Ilustrasi filter Frangi.

Untuk mengurangi noise akibat turunan, citra dihaluskan dengan konvolusi. Umumnya, digunakan Gaussian smoothing. Dapat ditunjukkan bahwa mencari turunan citra konvolusi yang dihaluskan Gaussian adalah

setara dengan mencari turunan Gaussian yang dikonvolusikan dengan gambar. Kita akan menentukan turunan kedua Gaussian menggunakan rumus di bawah ini, dimana G_{kita} adalah Gaussian.

$$G_{kita} = \begin{bmatrix} \frac{\partial^2 G_{kita}}{\partial x^2} & \sigma^2_{\text{grammata}} \\ \frac{\partial^2 G_{kita}}{\partial x \partial y} & \frac{\partial^2 G_{kita}}{\partial y^2} \end{bmatrix} \quad [\text{Basa Indonesia}] \quad (4.10)$$

Selanjutnya kita akan menentukan turunan kedua lokal (Hessian) dan nilai eigennya. Untuk gambar 2D, akan ada dua nilai eigen (λ_1 dan λ_2) untuk setiap koordinat piksel. Nilai eigen kemudian diurutkan dalam urutan menaik. Sebuah piksel dianggap sebagai bagian dari struktur tubular atau seperti pembuluh jika $\lambda_1 \approx 0$ sementara $|\lambda_2| > |\lambda_1|$ Bahasa Indonesia:

Untuk gambar 3D, akan ada tiga nilai eigen (λ_1 Bahasa Indonesia: $\lambda_1, \lambda_2, \lambda_3$) untuk setiap koordinat voxel. Nilai eigen kemudian diurutkan dalam urutan menaik. Sebuah voxel dianggap sebagai bagian dari struktur tubular atau seperti pembuluh jika $\lambda_1 \approx 0$ sementara λ_2, λ_3 memiliki nilai absolut tinggi yang hampir sama dan memiliki tanda yang sama. Bejana yang terang akan memiliki nilai positif untuk λ_1 dan λ_2 sedangkan pembuluh darah yang lebih gelap akan memiliki nilai negatif untuk λ_1 dan λ_2 .

Dalam kode di bawah ini, kami akan menunjukkan penggunaan filter Frangi secara terprogram. Gambar pertama kali dibuka dan diubah menjadi skala abu-abu. Gambar diubah menjadi array numpy menggunakan fungsi np.array sehingga dapat dimasukkan ke filter Frangi. Terakhir, panggilan dilakukan ke filter Frangi yang terletak di modul skimage.filters. Output dari fungsi frangi kemudian disimpan ke dalam sebuah berkas.

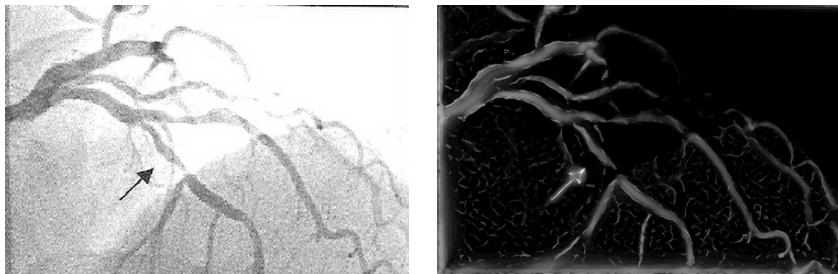
```
impor cv2
impor numpy sebagai np

impor numpy sebagai np
dari PIL impor Gambar
dari skimage.filters impor frangi

img = cv2.imread('..//Gambar/angiogram1.png')
```

```
img1 = np.asarray(img)
img2 = frangi(img1, black_ridges=True)
img3 = 255*(img2-np.min(img2))/(np.max(img2)-np.min(img2))
cv2.imwrite('../Gambar/frangi_output.png', img3)
```

Gambar di [Gambar 4.14\(a\)](#) adalah input ke filter Frangi dan gambar di [Gambar 4.14\(b\)](#) adalah keluaran dari filter Frangi. Citra masukan adalah angiogram yang dengan jelas memperlihatkan beberapa pembuluh darah yang ditingkatkan oleh kontras. Citra keluaran hanya berisi piksel yang berada di dalam pembuluh darah. Kontras menyatakan bahwa



(a) Gambar masukan untuk filter Frangi

(b) Output filter Frangi (gambar disempurnakan untuk visualisasi)

GAMBAR 4.14: Contoh filter Frangi.

4.5 Ringkasan

- Filter rata-rata menghaluskan gambar sekaligus mengaburkan bagian tepi gambar.
- Filter median efektif dalam menghilangkan noise salt-and-pepper.

- Filter turunan pertama yang paling banyak digunakan adalah Sobel, Prewitt dan Canny.
 - Baik Laplacian maupun LoG merupakan filter turunan kedua yang populer. Laplacian sangat sensitif terhadap noise. Dalam LoG, Gaussian menghaluskan gambar sehingga noise dari Laplacian dapat dikompensasi. Namun, LoG mengalami efek spaghetti.
 - Filter Frangi digunakan untuk mendeteksi struktur seperti pembuluh.
-

4.6 Latihan

1. Tulis program Python untuk menerapkan mean filter pada gambar dengan noise salt-and-pepper. Jelaskan outputnya, termasuk kemampuan mean filter untuk menghilangkan noise.
2. Jelaskan seberapa efektif filter rata-rata dalam menghilangkan derau salt-andpepper. Berdasarkan pemahaman Anda tentang filter median, dapatkah Anda menjelaskan mengapa filter rata-rata tidak dapat menghilangkan derau salt-andpepper?
3. Dapatkah filter maks atau filter min digunakan untuk menghilangkan noise salt-and-pepper?
4. Periksa dokumentasi scipy yang tersedia di <http://docs.scipy.org/doc/scipy/reference/ndimage.html>. Identifikasi fungsi Python yang dapat digunakan untuk membuat filter khusus.
5. Tulis program Python untuk memperoleh selisih Laplacian Gaussian (LoG). Kode semu untuk program tersebut adalah sebagai berikut:
 - (a) Bacalah gambarnya.

- (b) Terapkan filter LoG dengan asumsi deviasi standar 0,1 dan simpan gambar sebagai im1.
 - (c) Terapkan filter LoG dengan asumsi deviasi standar 0,2 dan simpan gambar sebagai im2.
 - (d) Temukan perbedaan antara kedua gambar tersebut dan simpan gambar yang dihasilkan?
6. Dalam bab ini, kita telah membahas beberapa filter spasial. Identifikasi dua filter lainnya dan bahas propertinya.