Bab 5

Konvolusi dan Transformasi Fourier

Bab ini berisi konsep matematis yang melandasi teori pengolahan citra. Dua operasi matematis penting yang perlu dipahami dalam mempelajari pengolahan citra dijital adalah operasi konvolusi dan Transformasi Fourier. Konvolusi terdapat pada operasi pengolahan citra yang mengalikan sebuah citra dengan sebuah *mask* atau *kernel* (akan dijelaskan kemudian), sedangkan Transformasi Fourier dilakukan bila citra dimanipulasi dalam ranah (domain) frekuensi ketimbang dalam ranah spasial. Bagian pertama di dalam Bab 5 ini akan membahas konvolusi, dan bagian kedua akan membahas Transformasi Fourier.

5.1 Teori Konvolusi

Operasi yang mendasar dalam pengolahan citra adalah operasi **konvolusi**. Konvolusi 2 buah fungsi $f(x)$ dan $g(x)$ didefinisikan sebagai berikut:

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da \qquad (5.1)$$

yang dalam hal ini, tanda * menyatakan operator konvolusi, dan peubah (*variable*) *a* adalah peubah bantu (*dummy variable*).

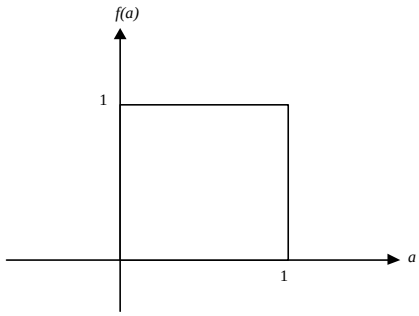
Untuk fungsi diskrit, konvolusi didefinisikan sebagai

$$h(x) = f(x) * g(x) = \sum_{a=-\infty}^{\infty} f(a)g(x-a) \qquad (5.2)$$

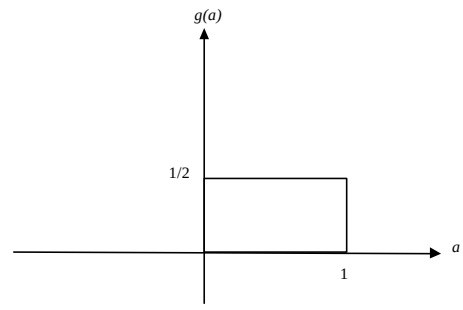
Pada operasi konvolusi di atas, $g(x)$ disebut **kernel konvolusi** atau **kernel penapis** (*filter*). Kernel $g(x)$ merupakan suatu jendela yang dioperasikan secara bergeser pada sinyal masukan $f(x)$, yang dalam hal ini, jumlah perkalian kedua fungsi pada setiap titik merupakan hasil konvolusi yang dinyatakan dengan keluaran $h(x)$.

Ilustrasi konvolusi adalah sebagai berikut. Misalkan fungsi $f(x)$ dan $g(x)$ diperlihatkan pada Gambar 5.1(a) dan 5.1(b). Langkah-langkah perhitungan hasil konvolusi ditunjukkan mulai dari Gambar 5.1(c) sampai 5.11(f). Hasil konvolusi ditunjukkan pada Gambar 5.1(g), yaitu:

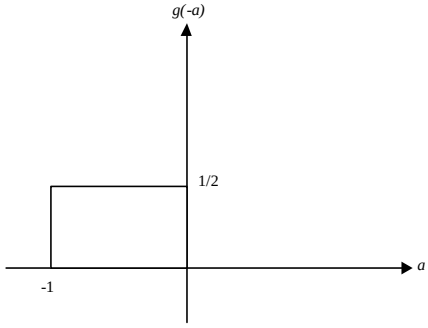
$$f(x) * g(x) = \begin{cases} x/2, & 0 \le x < 1 \\ 1 - x/2, & 1 \le x \le 2 \\ 0, & \textit{lainnya} \end{cases} \qquad (5.3)$$



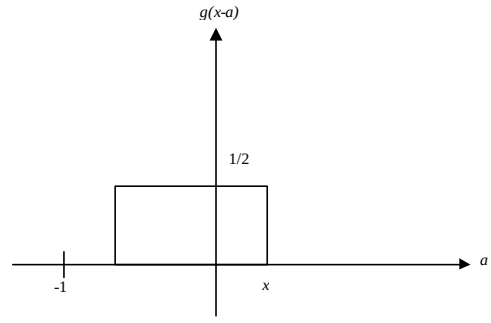
(a)



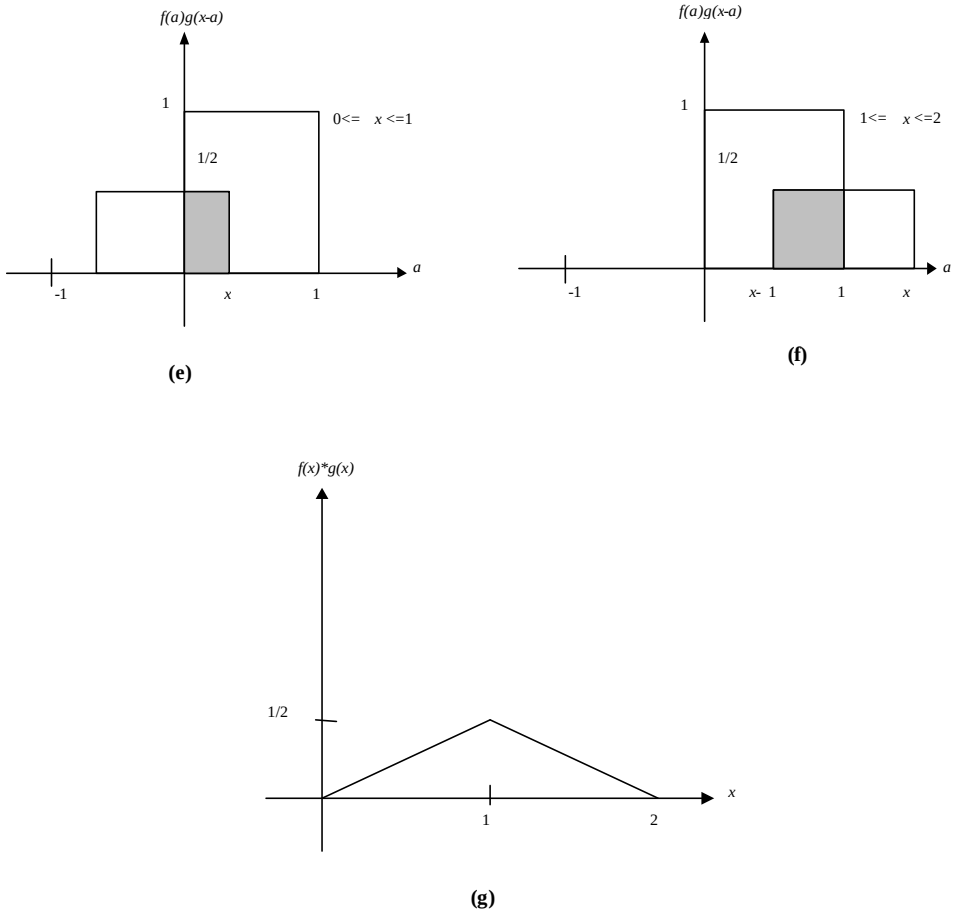
(b)



(c)



(d)



Gambar 5.1. *Ilustrasi proses konvolusi [GON77]*

Contoh ilustrasi konvolusi yang lain adalah dengaan fungsi delta. Ada dua macam fungsi delta: **delta Dirac** dan **delta Kronecker**.

Fungsi delta Dirac disebut juga fungsi denyut (impuls). Fungsi ini bernilai 0 untuk $x \neq 0$, dan "lebar" denyutnya sama dengan 1. Secara matematis fungsi delta Dirac definisikan sebagai

$$d(x) = 0, x \neq 0$$
$$\lim_{e \to 0} \int_{-e}^{e} d(x)dx = 1 \qquad (5.4)$$

Gambar 5.2 memperlihatkan bentuk fungsi delta Dirac.

Sifat-sifat fungsi delta Dirac:

$$1.\ \int_{-\infty}^{\infty} f(x')\delta(x-x')dx' = f(x) \tag{5.5}$$

$$2.\ \delta(ax) = \frac{\delta(x)}{|a|} \tag{5.6}$$

Fungsi delta Dirac adalah fungsi dengan daerah asal bilangan riil. Bila kita bekerja dengan fungsi diskrit, maka fungsi delta yang digunakan adalah fungsi delta Kronecker, yang didefinisikan sebagai

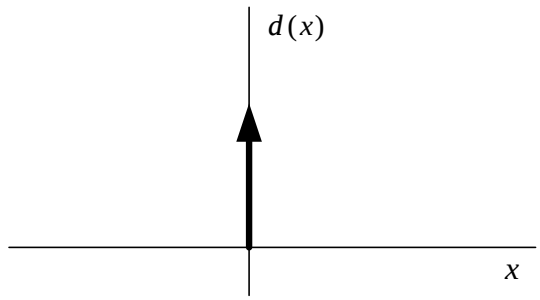
$$\delta(n) = \begin{cases} 0, & n \neq 0 \\ 1, & n = 0 \end{cases} \tag{5.7}$$

dengan sifat

$$\sum_{m=-\infty}^{\infty} f(m)\delta(n-m) = f(n) \tag{5.8}$$

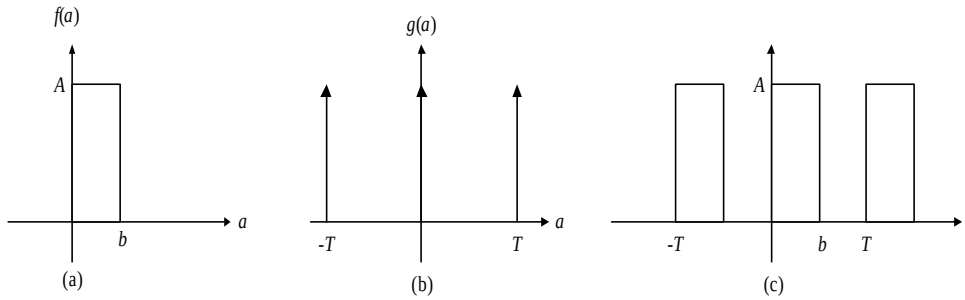
Bentuk dwimatra dari fungsi delta diperoleh dengan mengalikan bentuk satumatranya:

Dirac: $\delta(x,y) = \delta(x)\,\delta(y)$
Kronecker: $\delta(m,n) = \delta(m)\,\delta(n)$



Gambar 5.2. *Fungsi delta Dirac*

Hasil konvolusi fungsi $f(x)$ pada Gambar 5.3(a) dengan fungsi $g(x) = \delta(x + T) + \delta(x) + \delta(x - T)$ pada Gambar 5.3(b) ditunjukkan pada Gambar 5.3(c).

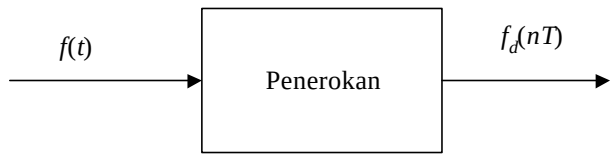


Gambar 5.3. *Konvolusi dengan fungsi impuls*

Salah satu penggunaan fungsi delta adalah melakukan penerokan (*sampling*) pada sinyal malar $f(x)$. Proses penerokan umumnya dilakukan pada periode yang tetap. Jika sinyal malar $f(t)$ diterok dengan periode tetap T, maka diperoleh serangkaian nilai diskrit $f_d(n)$:

$$f_d(n) = f(nT), \quad -\infty < n < +\infty$$

Proses penerokan ini ditunjukkan dengan Gambar 5.4.



Gambar 5.4. *Proses penerokan*

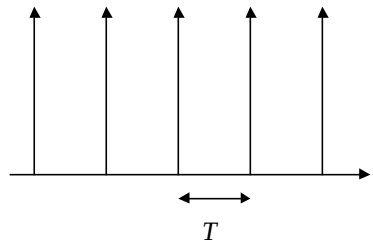
Secara matematis, proses penerokan dinyatakan sebagai perkalian sinyal malar $f(t)$ dengan fungsi penerok berupa rentetan sinyal delta sejarak T satu sama lain (Gambar 5.5). Fungsi penerok itu dapat dinyatakan sebagai

$$s(t) = \sum_{-\infty}^{\infty} d(t - nT) \tag{5.9}$$

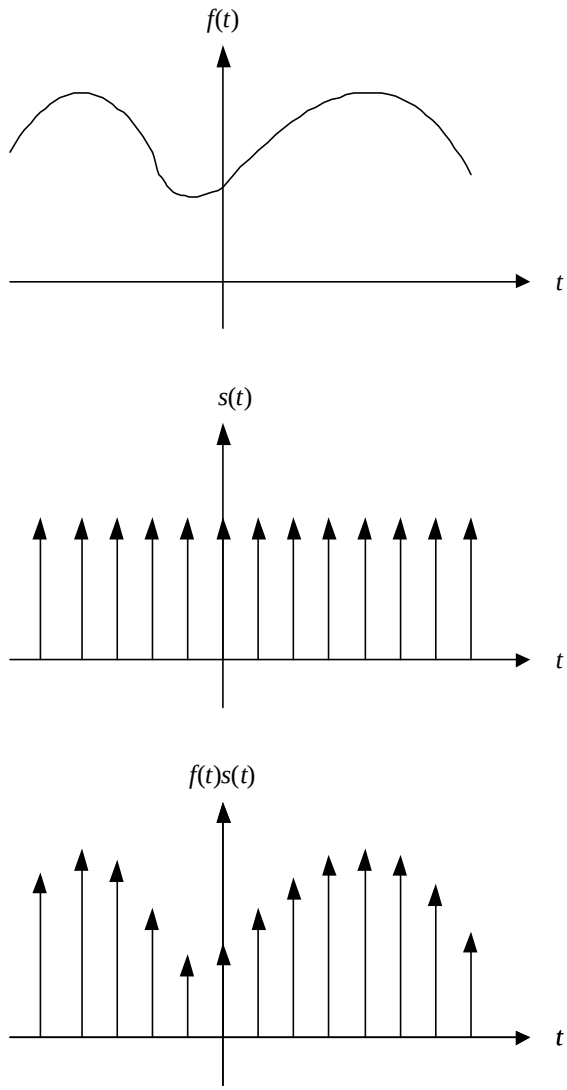
Dengan demikian,

$$f_d(t) = f(t)s(t) = f(t)\sum_{-\infty}^{\infty} d(t - nT) = \sum_{-\infty}^{\infty} f(t)d(t - nT) \tag{5.10}$$

Ilustrasi grafis proses penerokan ditunjukkan pada Gambar 5.6



Gambar 5.5. *Fungsi penerok s*



Gambar 5.6. *Ilustrasi grafis proses penerokan*

5.1 Konvolusi Pada Fungsi Dwimatra

Untuk fungsi dengan dua peubah (fungsi dua dimensi atau dwimatra), operasi konvolusi didefinisikan sebagai berikut:

a) untuk fungsi malar

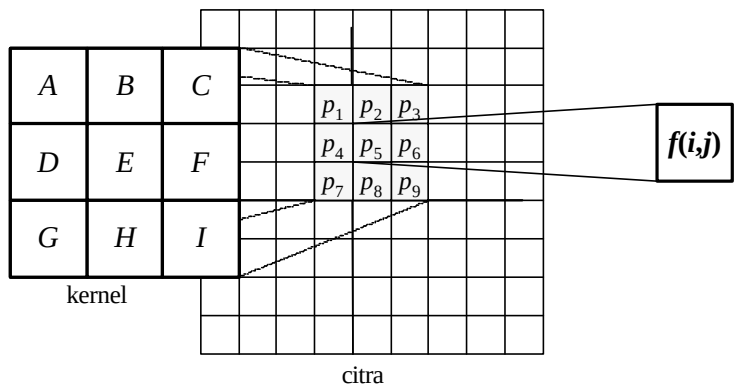
$$h(x,y) = f(x,y) * g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(a,b)g(x-a,y-b)dadb \qquad (5.11)$$

b) untuk fungsi diskrit

$$h(x,y) = f(x,y) * g(x,y) = \sum_{a=-\infty}^{\infty}\sum_{b=-\infty}^{\infty} f(a,b)g(x-a,y-b) \qquad (5.12)$$

Fungsi penapis $g(x,y)$ disebut juga *convolution filter*, *convolution mask*, *convolution kernel*, atau *template*. Dalam ranah diskrit kernel konvolusi dinyatakan dalam bentuk matriks (umumnya 3×3, namun ada juga yang berukuran 2×2 atau 2×1 atau 1×2). Ukuran matriks ini biasanya lebih kecil dari ukuran citra. Setiap elemen matriks disebut koefisien konvolusi.

Ilustrasi konvolusi ditunjukkan pada Gambar 5.7.



$$f(i,j) = A\,p_1 + B\,p_2 + C\,p_3 + D\,p_4 + E\,p_5 + F\,p_6 + G\,p_7 + H\,p_8 + I\,p_9$$

Gambar 5.7 *Ilustrasi konvolusi [JAI95]*

Operasi konvolusi dilakukan dengan menggeser *kernel* konvolusi *pixel* per *pixel*. Hasil konvolusi disimpan di dalam matriks yang baru.

Contoh 5.1. Misalkan citra $f(x, y)$ yang berukuran 5×5 dan sebuah *kernel* atau *mask* yang berukuran 3×3 masing-masing adalah sebagai berikut:

$$f(x, y) = \begin{bmatrix} 4 & 4 & 3 & 5 & 4 \\ 6 & 6 & 5 & 5 & 2 \\ 5 & 6 & 6 & 6 & 2 \\ 6 & 7 & 5 & 5 & 3 \\ 3 & 5 & 2 & 4 & 4 \end{bmatrix} \qquad g(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \bullet 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

(Keterangan: Tanda • menyatakan posisi (0, 0) dari *kernel*)

Operasi konvolusi antara citra $f(x, y)$ dengan *kernel* $g(x, y)$,

$f(x, y) * g(x, y)$

dapat diilutrasikan sebagai berikut:

(1) Tempatkan *kernel* pada sudut kiri atas, kemudian hitung nilai *pixel* pada posisi (0, 0) dari *kernel*:

4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

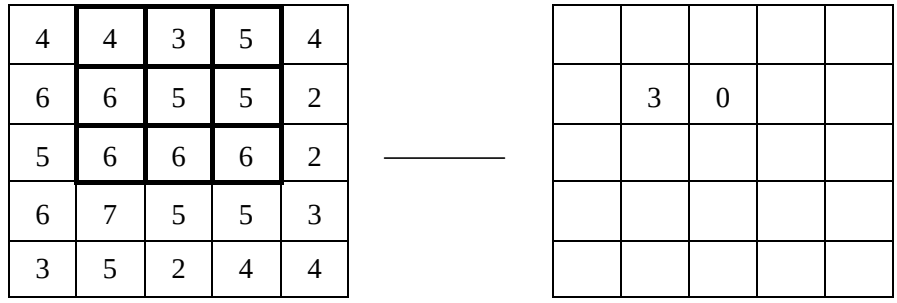
———

	3			

Hasil konvolusi = 3. Nilai ini dihitung dengan cara berikut:

$(0 \times 4) + (-1 \times 4) + (0 \times 3) + (-1 \times 6) + (4 \times 6) + (-1 \times 5) + (0 \times 5) + (-1 \times 6) + (0 \times 6) = 3$

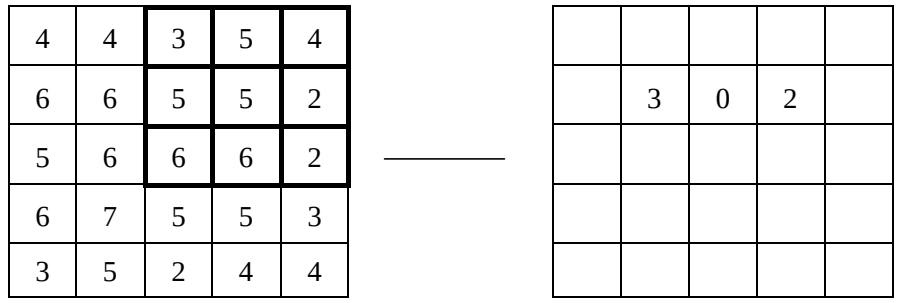
(2) Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (0, 0) dari *kernel*:



Hasil konvolusi = 0. Nilai ini dihitung dengan dengan cara berikut:

$(0 \times 4) + (-1 \times 3) + (0 \times 5) + (-1 \times 6) + (4 \times 5) + (-1 \times 5) + (0 \times 6) + (-1 \times 6) + (0 \times 6) = 0$

(3) Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (0, 0) dari *kernel*:



Hasil konvolusi = 2. Nilai ini dihitung dengan cara berikut:

$(0 \times 3) + (-1 \times 5) + (0 \times 4) + (-1 \times 5) + (4 \times 5) + (-1 \times 2) + (0 \times 6) + (-1 \times 6) + (0 \times 2) = 2$

(4) Selanjutnya, geser *kernel* satu *pixel* ke bawah, lalu mulai lagi melakukan konvolusi dari sisi kiri citra. Setiap kali konvolusi, geser *kernel* satu *pixel* ke kanan:

(5)

4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

———

	3	0	2	
	0			

Hasil konvolusi = 0. Nilai ini dihitung dengan cara berikut:

$(0 \times 6) + (-1 \times 6) + (0 \times 5) + (-1 \times 5) + (4 \times 6) + (-1 \times 6) + (0 \times 6) + (-1 \times 7) + (0 \times 5) = 0$

(6)

4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

———

	3	0	2	
	0	2		

Hasil konvolusi = 2. Nilai ini dihitung dengan cara berikut:

$(0 \times 6) + (-1 \times 5) + (0 \times 5) + (-1 \times 6) + (4 \times 6) + (-1 \times 6) + (0 \times 7) + (-1 \times 5) + (0 \times 5) = 2$

(7)

4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

———

	3	0	2	
	0	2	6	

Hasil konvolusi = 6. Nilai ini dihitung dengan cara berikut:

$$(0 \times 5) + (-1 \times 5) + (0 \times 2) + (-1 \times 6) + (4 \times 6) + (-1 \times 2) + (0 \times 5) + (-1 \times 5) + (0 \times 3) = 6$$

Dengan cara yang sama seperti di atas, maka *pixel-pixel* pada baris ketiga dikonvolusi sehingga menghasilkan:

	3	0	2	
	0	2	6	
	6	0	2	

■

Sebagai catatan, jika hasil konvolusi menghasilkan nilai *pixel* negatif, maka nilai tersebut dijadikan 0, sebaliknya jika hasil konvolusi menghasilkan nilai *pixel* lebih besar dari nilai keabuan maksimum, maka nilai tersebut dijadikan ke nilai keabuan maksimum (ingat operasi *clipping*).

Masalah timbul bila *pixel* yang dikonvolusi adalah *pixel* pinggir (*border*), karena beberapa koefisien konvolusi tidak dapat dapat diposisikan pada *pixel-pixel* citra (efek "menggantung"), seperti contoh di bawah ini:

4	4	3	5	4	?
6	6	5	5	2	?
5	6	6	6	2	?
6	7	5	5	3	
3	5	2	4	4	

Masalah "menggantung" seperti ini selalu terjadi pada *pixel-pixel* pinggir kiri, kanan, atas, dan bawah. Solusi untuk masalah ini adalah [SID95]:

1. *Pixel-pixel* pinggir diabaikan, tidak di-konvolusi. Solusi ini banyak dipakai di dalam pustaka fungsi-fungsi pengolahan citra. Dengan cara seperti ini, maka *pixel-pixel* pinggir nilainya tetap sama seperti citra asal. Gambar 5.8 memperlihatkan hasil konvolusi pada Contoh 5.1, yang dalam hal ini nilai *pixel-pixel* pinggir sama dengan nilai *pixel* semula.
2. Duplikasi elemen citra, misalnya elemen kolom pertama disalin ke kolom M+1, begitu juga sebaliknya, lalu konvolusi dapat dilakukan terhadap *pixel-pixel* pinggir tersebut.
3. Elemen yang ditandai dengan "?" diasumsikan bernilai 0 atau konstanta yang lain, sehingga konvolusi *pixel-pixel* pinggir dapat dilakukan.

Solusi dengan ketiga pendekatan di atas mengasumsikan bagian pinggir citra lebarnya sangat kecil (hanya satu *pixel*) relatif dibandingkan denagn ukuran citra, sehingga *pixel-pixel* pinggir tidak memperlihatkan efek yang kasat mata.

4	4	3	5	4
6	3	0	2	2
5	0	2	6	2
6	6	0	2	3
3	5	2	4	4

Gambar 5.8 *Pixel-pixel pinggir (yang tidak diarsir) tidak dikonvolusi (dari Contoh 5.1)*

Algoritma konvolusi citra $N \times M$ dengan dengan *mask* atau *kernel* yang berukuran 3×3 ditunjukkan pada Algoritma 5.1. *Pixel* yang dikonvolusi adalah elemen (i, j). Delapan buah *pixel* yang bertetangga dengan *pixel* (i, j) diperlihatkan pada Gambar 5.9.

i-1, j-1	i-1, j	i-1, j+1
i, j-1	i,j	i, j+1
i+1, j-1	i+1,j	i+1, j+1

Gambar 5.9 *Pixel-pixel pinggir (yang tidak diarsir) tidak dikonvolusi (dari Contoh 5.1)*

```
void konvolusi(citra Image, citra ImageResult, imatriks Mask,
               int N, int M)
/* Mengkonvolusi citra Image yang berukuran N ´ M dengan mask 3 ´ 3.
Hasil konvolusi disimpan di dalam matriks ImageResult.
 */
{ int i, j;

  for (i=1; i<=N-3; i++)
    for(j=1; j<=M-3; j++)
      ImageResult[i][j]=
          Image[i-1][j-1]*Mask[0][0] +
          Image[i-1][j+1]*Mask[0][1] +
          Image[i-1][j]*Mask[0][2] +
          Image[i][j-1]*Mask[1][0] +
          Image[i][j]*Mask[1][1] +
          Image[i][j+1]*Mask[1][2] +
          Image[i+1][j-1]*Mask[2][0] +
          Image[i+1][j]*Mask[2][1] +
          Image[i+1][j+1]*Mask[2][2];
}
```

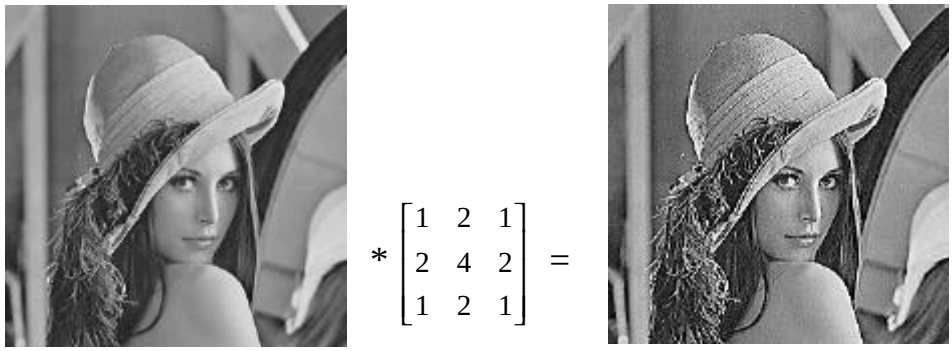
Algoritma 5.1. *Konvolusi citra dengan sebuah mask yang berukuran 3 ´ 3.*

Anda dapat melihat bahwa operasi konvolusi merupakan komputasi pada aras lokal, karena komputasi untuk suatu *pixel* pada citra keluaran melibatkan *pixel-pixel* tetangga pada citra masukannya.

Konvolusi berguna pada proses pengolahan citra seperti:
- perbaikan kualitas citra (*image enhancement*)
- penghilangan derau
- mengurangi erotan
- penghalusan/pelembutan citra
- deteksi tepi, penajaman tepi
- dll

Sebagai contoh, Gambar 5.9 memperlihatkan konvolusi citra Lena dengan penapis Gaussian untuk mempertajam tepi-tepi di dalam citra. Penapis Gaussian adalah sebuah *mask* yang berukuran 3×3:

$$g(x,y) = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



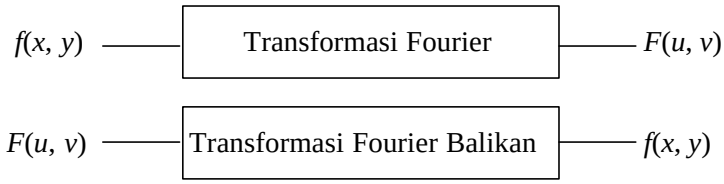
(a) Citra Lena semula (b) Citra Lena sesudah konvolusi

Gambar 5.10 *Konvolusi citra Lena dengan penapis Gaussian untuk mempertajam gambar.*

Karena konvolusi dilakukan per *pixel* dan untuk setiap *pixel* dilakukan operasi perkalian dan penjumlahan, maka jelas konvolusi mengkonsumsi banyak waktu. Jika citra berukuran $N \times N$ dan *kernel* berukuran $m \times m$, maka jumlah perkalian adalah dalam orde N^2m^2. Sebagai contoh jika citra berukuran 512 $\times$ 512 dan kernel berukuran 16×16, maka ada sekitar 32 juta perkalian yang dibutuhkan. Ini jelas tidak cocok untuk proses yang *real time* tanpa perangkat keras yang *dedicated*.

Satu cara mengurangi waktu komputasi adalah mentransformasi citra dan *kernel* ke dalam ranah frekuensi (dengan menggunakan Transformasi Fourier – akan diuraikan di upabab 5.2), selanjutnya konvolusi dilakukan dalam ranah waktu. Keuntungan utama dari penggunaan ranah frekuensi adalah proses konvolusi dapat diterapkan dalam bentuk perkalian langsung.

Proses perubahan fungsi dari ranah ranah spasial ke ranah frekuensi dilakukan melalui **Transformasi Fourier**. Sedangkan perubahan fungsi dari ranah frekuensi ke ranah spasial dilakukan melalui **Transformasi Fourier Balikan** (*invers*).



Dengan demikian, operasi konvolusi dua buah fungsi dalam ranah frekuensi menjadi:

$$h(x, y) = f(x, y) * g(x, y) \leftrightarrow H(u, v) = F(u, v)\, G(u, v)$$

5.2 Transformasi Fourier

Transformasi Fourier merupakan transformasi paling penting di dalam bidang pengolahan sinyal (*signal processing*), khususnya pada bidang pengolahan citra.

Umumnya sinyal dinyatakan sebagai bentuk plot amplitudo versus waktu (pada fungsi satu matra) atau plot amplitudo versus posisi spasial (pada fungsi dwimatra). Pada beberapa aplikasi pengolahan sinyal, terdapat kesukaran melakukan operasi karena fungsi dalam ranah waktu/spasial, misalnya pada operasi konvolusi di atas. Operasi konvolusi dapat diterapkan sebagai bentuk perkalian langsung bila fungsi berada dalam ranah frekunsi.

Transformasi Fourier adalah kakas (*tool*) untuk mengubah fungsi dari ranah waktu/spasial ke ranah frekuensi. Untuk perubahan sebaliknya digunakan Transformasi Fourier Balikan. Intisari dari Transformasi Fourier adalah menguraikan sinyal atau gelombang menjadi sejumlah sinusoida dari berbagai frekuensi, yang jumlahnya ekivalen dengan gelombang asal.

Di dalam pengolahan citra, transformasi Fourier digunakan untuk menganalisis frekuensi pada operasi seperti perekaman citra, perbaikan kualitas citra, restorasi citra, pengkodean, dan lain-lain. Dari analisis frekuensi, kita dapat melakukan perubahan frekuensi pada gambar. Perubahan frekuensi berhubungan dengan spektrum antara gambar yang kabus kontrasnya samapi gambar yang kaya akan rincian visualnya. Sebagai contoh, pada proses perekaman citra mungkin terjadi pengaburan kontras gambar. Pada gambar yang mengalami kekaburan kontras terjadi perubahan intensitas secara perlahan, yang berarti kehilangan informasi frekuensi tinggi. Untuk meningkatkan kualitas gambar, kita menggunakan penapis frekuensi tinggi sehingga *pixel* yang berkontras kabur dapat dinaikkan intensitasnya.

5.3 Transformasi Fourier Malar

Transformasi Fourier malar (kontinu) untuk satu peubah:

$$\Im\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)e^{-i2pux}du \qquad (5.13)$$

Transformasi Fourier Balikan untuk satu peubah:

$$\Im^{-1}\{F(u)\} = f(x) = \int_{-\infty}^{\infty} F(u)e^{i2pux}du \qquad (5.14)$$

yang dalam hal ini,

i = imaginer = $\sqrt{-1}$
u adalah peubah frekuensi

Baik transformasi Fourier maupun Transformasi Fourier Balikan keduanya dinamakan **pasangan transformasi Fourier**.

Untuk $f(x)$ real, $F(u)$ adalah fungsi kompleks dan dapat dituliskan sebagai:

$$F(u) = R(u) + iI(u) = |F(u)|e^{if(u)} \qquad (5.15)$$

Amplitudo atau $|F(u)|$ disebut **spektrum Fourier** dari $f(x)$ dan didefinisikan sebagai:

$$|F(u)| = \sqrt{R^2(u) + I^2(u)} \qquad (5.16)$$

Sudut fase spektrum,

$$\Theta(u) = \tan^{-1}[\frac{I(u)}{R(u)}] \qquad (5.17)$$

menyatakan pergeseran fase atau sudut fase dari setiap frekuensi u.

Dengan mengingat kesamaan Euler,

$$e^{\pm ix} = \cos(x) \pm i\sin(x) \qquad (5.18)$$

maka pasangan transformasi Euler dapat juga ditulis sebagai

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-i2pux}dx = \int_{-\infty}^{\infty} f(x)\{\cos(2pux) - i\sin(2pux)\}dx \qquad (5.19)$$

$$f(x) = \int_{-\infty}^{\infty} F(u)e^{i2pux}du = \int_{-\infty}^{\infty} F(u)\{\cos(2pux) + i\sin(2pux)\}du \qquad (5.20)$$

Transformasi Fourier untuk fungsi dengan dua peubah adalah

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-i2p(ux+uy)}dudv \qquad (5.21)$$

sedangkan Transformasi Fourier Balikannya adalah

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{i2p(ux+uy)}dudv \qquad (5.22)$$

yang dalam hal ini, x dan y adalah peubah spasial, sedangkan u dan v adalah peubah frekuensi.

Spektrum Fourier dari fungsi dua peubah:

$$|F(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)} \qquad (5.23)$$

sedangkan sudut fasenya:

$$\Theta(u,v) = \tan^{-1}[\frac{I(u,v)}{R(u,v)}] \qquad (5.24)$$

Sifat-sifat Transformasi Fourier

Jika $f(t) \leftrightarrow F(u)$ dan $g(t) \leftrightarrow G(u)$, maka sifat-sifat Transformasi Fourier dirumuskan di dalam Tabel 5.1.

5.4 Transformasi Fourier Diksrit

Pada pengolahan sinyal dengan komputer digital, fungsi dinyatakan oleh himpunan berhingga nilai diskrit. Transformasi Fourier Diskrit (TFD) ditujukan bagi persoalan yang tidak menghasilkan solusi transformasi Fourier dalam bentuk fungsi malar.

Tabel 5.1 *Sifat-sifat Transformasi Fourier*

Sifat	Ranah Waktu	Ranah Frekuensi
1. Kelanjaran	$af(t)+bg(t)$	$aF(u)+bG(u)$
2. Penskalaan	$f(at)$	$\frac{1}{\lvert a\rvert}F(u/a)$
3. Pergeseran	$f(t-a)$	$F(u-a)$
4. Modulasi	$e^{i2pat}f(t)$	$F(u)e^{-i2pua}$
5. Konyugasi	$f^*(t)$	$F^*(-u)$
6. Konvolusi	$h(t)=f(t)*g(t)$	$H(u)=F(u)G(u)$
7. Perkalian	$h(t)=f(t)g(t)$	$H(u)=F(u)*G(u)$
8. Diferensiasi	$\frac{d^n f(t)}{dt^n}$	$(i2pu)^n F(u)$
9. Simetri	$F(t)$	$f(-u)$
10. Hasil kali dalam	$\int_{-\infty}^{\infty} f(t)g^*(t)dt$	$\int_{-\infty}^{\infty} F(u)G^*(u)du$

Bila $f(x)$ yang menerus dibuat diskrit dengan mengambil N buah terokan (*sampling*) sejarak Δx, yaitu himpunan nilai $\{f(x_0), f(x_0 + \Delta x), f(x_0 + 2\Delta x), \ldots, f(x_0 + (N\text{-}1)\Delta x)\}$.

Jadi,

$$f_x = f(x_0 + x\,\Delta x), \quad x = 0, 1, 2, \ldots, N-1$$

Pasangan Transformasi Fourier Diskrit untuk fungsi dengan satu peubah:

$$F_u = \frac{1}{N}\sum_{x=0}^{N-1} f_x e^{-i2pux/N} \qquad , u = 0, 1, 2, \ldots, N-1 \qquad (5.25)$$

$$f_x = \sum_{u=0}^{N-1} F_u e^{i2pux/N} \qquad , x = 0, 1, 2, \ldots, N-1 \qquad (5.26)$$

Dengan mengingat kesamaan Euler, pasangan Transformasi Fourier Diskrit dapat ditulis dalam bentuk

$$F_u = \frac{1}{N}\sum_{x=0}^{N-1}[\, f_x \cos(2pux/N) - i\ f_x \sin(2pux/N)] \qquad (5.27)$$

$$f_x = \sum_{u=0}^{N-1}[F_u \cos(2pux/N) + i\, F_u \sin(2pux/N)] \qquad (5.28)$$

Interpretasi dari TFD adalah sebagai berikut: TFD mengkonversi data diskrit menjadi sejumlah sinusoida diskrit yang frekuensinya dinomori dengan u = 0, 1, 2, …, N – 1, dan ampiltudonya diberikan oleh $F(u)$.

Faktor 1/N pada persamaan $F(u)$ adalah faktor skala yang dapat disertakan dalam persamaan $F(u)$ atau dalam persamaan $f(x)$, tetapi tidak kedua-duanya.

Contoh 5.2. [MEN89] Diketahui fungsi sinyal $f(t)$ dengan hasil penerokan ke dalam nilai-nilai diskrit sebagai berikut (N = 4):

$x_0 = 0.5, \quad f_0 = 2$
$x_1 = 0.75, f_1 = 3$
$x_2 = 1.0, \quad f_2 = 4$
$x_3 = 1.25, f_3 = 4$

Transformasi Fourier Diskrit adalah sebagai berikut:

$$F_0 = \frac{1}{4}\sum_{x=0}^{3} f_x e^{-i.0.2px/4} = \frac{1}{4}\sum_{x=0}^{3} f_x e^{0} = \frac{1}{4}\sum_{x=0}^{3} f_x = \frac{1}{4}(f_0 + f_1 + f_2 + f_3) = 3.25$$

$$F_1 = \frac{1}{4}\sum_{x=0}^{3} f_x e^{-i.1.2px/4} = \frac{1}{4}(f_0 e^{0} + f_1 e^{-ip/2} + f_2 e^{-ip} + f_3 e^{-i3p/2})$$

$$= \frac{1}{4}(2 + 3[\cos(p/2) - i\sin(p/2)] + 4[\cos(p) - i\sin(p)] + 4[\cos(3p/2) - i\sin(3p/2)])$$

$$= \frac{1}{4}(2 + 3[0 - i] + 4[-1 - 0] + 4[0 + i]) = \frac{1}{4}(-2 - i)$$

$$F_2 = \frac{1}{4}\sum_{x=0}^{3} f_x e^{-i.2.2px/4} = \frac{1}{4}(2e^{0} + 3e^{-ip} + 4e^{-i2p} + 4e^{-i3p})$$

$$= \frac{1}{4}(1 + 0i) = -\frac{1}{4}$$

$$F_3 = \frac{1}{4}(2 + i)$$

Spektrum Fouriernya:

$$|F_0| = 3.25$$

$$|F_1| = \sqrt{(1/2)^2 + (1/4)^2} = \sqrt{1/4 + 1/16} = \sqrt{5/16} = \frac{1}{4}\sqrt{5}$$

$$|F_2| = \frac{1}{4}$$

$$|F_3| = \frac{1}{4}\sqrt{5}$$ ■

Algoritma TFD dan algoritma TFD Balikan ditunjukkan masing-masing pada Algoritma 5.2 dan Algoritma 5.3.

```
void TFD(int N)
/* Melakukan Transformasi Fourier Diskrit untuk N buah data masukan.
Hasil transformasi disimpan di dalam array R dan I. Array R menyimpan
bagian riil, dan array I menyimpan bagian bagian imajiner. Kedua array
ini dideklarasikan sebagai peubah global. Data masukan disimpan di dalam
array f[0] s/d f[N-1]
*/
{
  int j, k;
  double tetha;

  for (j=0; j<N; j++)
  {
    R[j]=0.0;
    I[j]=0.0;
  }
  for (k=0; k<=N; k++)
     for (j=0; j<=N-1; j++)
     {
       tetha= k*2*3.14*j/(double)N;
       R[k]=R[k]+(f[j]*cos(tetha))/(double)N;
       I[k]=I[k]-(f[j]*sin(tetha))/(double)N;
     }
}
```

Algoritma 5.2. *Transformasi Fourier Diskrit*

```
void TFD_balikan(int N)
/* Melakukan Transformasi Fourier Diskrit Balikan untuk N buah data
masukan. Masukan disimpan di dalam array R dan I. Array R menyimpan
bagian riil, dan array I menyimpan bagian bagian imajiner. Kedua array
ini dideklarasikan sebagai peubah global. Data keluaran disimpan di dalam
array fReal[0] s/d fReal[N-1] dan array fImag[1] s/d fImag[N-1].
*/
{
  int j, k;
  double tetha, epsilon = 1E-12;

  for (j=0; j<N; j++)
  {
    fReal[j]=0;
    fImag[j]=0;
  }

  for (k=0; k<N1; k++)
  {
    for (j=0; j<N; j++)
    {
     tetha=k*2*3.14*j/(double)N;
     fReal[k]=fReal[k]+(R[j]*cos(tetha)–I[j]*sin(tetha));
     fImag[k]=fImag[k]+(I[j]*cos(tetha)+ R[j]sin(tetha));
    }
    if (fImag[k] < epsilon) fImag[k]=0;
  }
}
```

Algoritma 5.3. *Transformasi Fourier Diskrit Balikan*

Citra digital adalah fungsi diskrit dalam ranah spasial, dengan dua peubah, *x* dan *y*. Pada fungsi diskrit dengan dua peubah dan berukuran $N \times M$, pasangan Transformasi Fourier Diskritnya adalah:

$$F_{u,v} = \frac{1}{NM} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f_{x,y} e^{-i2p\,(ux/N + vy/M)} \quad , u \text{ dan } v = 0, 1, 2, \ldots, N-1 \qquad (5.29)$$

$$f_{x,y} = \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F_{u,v} e^{i2p\,(ux/N + vy/M)} \quad , x \text{ dan } y = 0, 1, 2, \ldots, N-1 \qquad (5.30)$$

atau

$$F_{u,v} = \frac{1}{N} \sum_{x=0}^{N-1} f_{x,y} e^{-i2pux/N} \; \frac{1}{M} \sum_{y=0}^{M-1} f_{x,y} e^{-i2pvy/M} \qquad (5.31)$$

$$f_{x,y} = \sum_{u=0}^{N-1} F_{u,v} e^{i2pux/N} \sum_{v=0}^{M-1} F_{u,v} e^{i2pvy/M} \qquad (5.31)$$

untuk $u, x = 0, 1, \ldots, N-1$ dan $v, y = 0, 1, \ldots, M-1$.

Algoritma TFD dan balikannya dapat diterapkan untuk fungsi diskrit dwimatra. Mula-mula transformasi dilakukan dalam arah x (dengan nilai y tetap). Kemudian, hasilnya ditransformasikan lagi dalam arah y.

Algoritma TFD tidak bagus untuk N yang besar karena komputasinya memakan waktu yang lama. Kompleksitas waktu algoritmanya adalah $O(N^2)$. Algoritma yang dikenal cepat untuk menghitung transformasi Fourier diskrit adalah FFT (*Fast Fourier Transform*). Algoritma FFT mempunyai kompleksitas waktu $O(N^2 \log N)$. Jadi, untuk $N = 50$, TFC kira-kira 10 kali lebih cepat daripada TFD, untuk $N = 1000$ sekitar 100 kali lebih cepat. Algoritma FFT tidak dibahas di dalam buku ini.