CONVOLUCIÓN

1 Introducción matemática

Es un concepto matemático de gran utilidad en procesado de imágenes (suavización, diferenciación)

La convolución de dos funciones $f(x)$ y $g(x)$, $f(x) * g(x)$ se define con la integral

$f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha) g(x - \alpha) d\alpha$

Ejemplo:

$f(\alpha) = \begin{cases} 1 & \text{if } |\alpha| \leq 1 \\ 0, & \text{otherwise.} \end{cases}$

$g(\alpha) = \begin{cases} \frac{1}{2} & \text{if } 0 \leq \alpha \leq 1 \\ 0, & \text{otherwise.} \end{cases}$

Observe que

- $g(-\alpha)$ es la función $g(\alpha)$ reflejada sobre el eje vertical
- $g(x - \alpha)$ es $g(-\alpha)$ desplazada a la derecha una distancia x
- Para un valor dado de x, $f(\alpha)g(x - \alpha)$ integrado sobre todos los valores de α es el área de solape de las dos funciones, ya que $f(\alpha)$ tiene altura 1

Si ahora consideramos x desde $-\infty$ a $+\infty$,

- para $x \leq -1$ o $x \geq 2$, no hay solape
- para x desde -1 a 0, el área de solape se incremeta rápidamente de 0 a $\frac{1}{2}$
- para x desde 0 a 1, el área de solape se mantiene constante e igual a $\frac{1}{2}$
- para x desde 1 a 2, el área de solape se decremeta rápidamente de $\frac{1}{2}$ a 0

Por tanto

$f(x) * g(x) = \begin{cases} \frac{x+1}{2} & \text{if } -1 \leq x \leq 0 \\ \frac{1}{2}, & \text{if } 0 \leq x \leq 1 \\ 1 - \frac{x}{2}, & \text{if } 1 \leq x \leq 2 \\ 0, & \text{otherwise.} \end{cases}$

La convolución de funciones 2D se define como

$$h(x, y) = f(x, y) * g(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(u, v) g(x - u, y - v) du dv$$

2 Transformadas de Fourier y Convolución

Teorema de la Convolución: Si $f(x)$ y $g(x)$ son dos funciones con transformadas de Fourier $F(u)$ y $G(u)$, la transformada de Fourier de la convolución $f(x) * g(x)$, es el producto de las transformadas de Fourier de las dos funciones, $F(u)G(u)$

Por tanto puede usarse **deconvolución** para compensar una imagen imperfecta debido a algún efecto del sistema de adquisición:

- Se hace la transformada de Fourier de la imagen imperfecta
- Se hace la transformada de Fourier de la función que describe el efecto del sistema
- Se divide la primera por la segunda para obtener la transformada de Fourier de la imagen ideal
- Se hace la transformada de Fourier inversa para recuperar la imagen ideal

3 Introducción computacional

Un **borde** es un cambio abrupto en la intensidad de una imagen. Suponga que quiera determinar bordes verticales en una imagen. Puede calcular las derivadas de la imagen en dirección horizontal; las derivadas con un valor alto, positivo o negativo, seran bordes verticales. Si la intensidad fuese una función continua, tendría que calcular

$$\frac{\partial F(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{F(x+\Delta x, y) - F(x,y)}{\Delta x}$$

Las imágenes digitalizadas son funciones de variable discreta. Δx no puede hacerse arbitrariamente pequeño. Si la unidad de medida es el pixel tenemos

$\Delta x = 1$

y una aproximación a la derivada es:

$$\left.\frac{\partial F(x,y)}{\partial x}\right|_{x=j, y=i} \approx f(i, j+1) - f(i,j)$$

Por ejemplo, en C, para computar esta aproximación a lo largo de la fila i tendríamos:

```
for (j = jstart; j <= jend; j++) h[i][j] = f[i][j+1] - f[i][j];
```

Considere una máscara g con valores $g[0] = 1$ y $g[1] = -1$. Podría considerar la operación anterior como: situar la máscara en cada posición j a lo largo de la fila i, multiplicar lo que hay bajo la máscara por las entradas de la máscara, sumar los resultados, *i.e.*,

```
for (j = jstart; j <= jend; j++)
    h[i][j] = g[0] * f[i][j+1] + g[1] * f[i][j];
```

Es más general: podría cambiar los valores de g sin cambiar el código.

En el proceso de añadir más generalización, podría pensarse en una definición *centrada* de la derivada:

$$\left.\frac{\partial F(x,y)}{\partial x}\right|_{x=j,y=i} \approx f(i,j+1) - f(i,j-1)$$

Con una máscara con tres entradas ($g[-1] = 1$, $g[0] = 0$, $g[1] = -1$) podríamos computar la derivada anterior y dejar abierta la posibilidad de cambiar los valores de g:

```
for (j = jstart; j <= jend; j++)
    {
      h[i][j] = 0;
      for (b = bstart; b <= bend; b++)
          h[i][j] += g[b] * f[i][j-b];
    }
```

Hemos conseguido más generalidad: se puede elegir los vecinos horizontales a combinar y con que peso. La próxima generalización es combinar también pixels por encima de i, j, y para toda la imagen, no sólo una fila:

```
for (i = istart; i <= iend; i++)
for (j = jstart; j <= jend; j++)
    {
      h[i][j] = 0;
      for (a = astart; a <= aend; a++)
      for (b = bstart; b <= bend; b++)
          h[i][j] += g[a][b] * f[i-a][j-b];
    }
```

El código anterior es el cómputo de la convolución de f y g, *i.e.*,

$$h(i,j) = \sum_{a=a_{start}}^{a_{end}} \sum_{b=b_{start}}^{b_{end}} g(a,b) f(i-a, j-b) \tag{1}$$

A la convolución de una señal con una máscara g también se le llama *filtrado* de la señal con la máscara. A la máscara también se le llama *función de propagación de punto*. En efecto, sea la señal un *impulso* δ centrado en el origen,

$$f(i,j) = \delta(i,j) = \begin{cases} 1 & \text{if } i = j = 0 \\ 0, & \text{otherwise.} \end{cases}$$

Al aplicar la convolución, ecuación 1,

$$h(i,j) = g(i,j)$$