Visión

Dr. Félix Calderon Solorio

22 de noviembre de 2021

Índice general

Introducción

1.1. Creación de Imágenes

El Puntillismo aparece por vez primera en 1884, encabezado por el pintor Neo - Impresionista Georges Seurat, y contando con entre sus más fieles seguidores tales como Henri-Edmond Cross y Vlaho Bokovac. Su procedimiento empleado por estos artistas, consistente en poner puntos de colores puros en vez de pinceladas sobre tela; logrará una técnica que fue el resultado de los estudios cromáticos llevados a cabo por Georges Seurat (1859 - 1891), pintor francés, quien en 1884 llegó a la división de tonos por la posición de toques de colores que, mirados a cierta distancia, crean en la retina las combinaciones deseadas [Wikipedia].

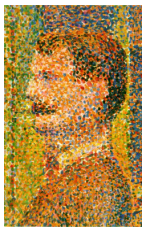


Figura 1.1: Imagen de puntos

1.1.1. Manejo de colores RGB

Definimos un pixel como la unidad básica de una imagen. Es un punto en cuyo valor se almacena la información concerniente a los colores. Los colores básicos que manejaremos son Rojo, Verde y Azul. Así pixel lo podemos ver como un arreglo $P = [P_r, P_g, P_b]$

En la siguiente figura podemos ver el como se almacena la información concerniente a los 32 bit o 4 Bytes

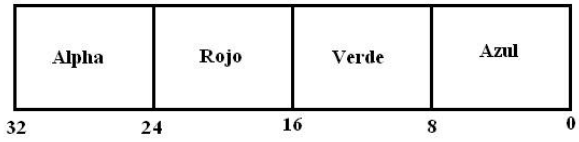


Figura 1.2: Formato de 32 bits para manejo de color

Una imagen discreta, la podemos definir como un arreglo bidimensional en el cual se almacena la información de cada punto de la imagen $256^3 = 16,777,216$ diferentes colores por pixel. Dado que la información se maneja en Bytes para almacenar un byte necesitaremos de 4 bytes, tres correspondientes a los colores Rojo, verde y Azul y un cuarto que tiene información del nivel de transparencia. En la cámaras es común hablar de su resolución en Megabytes, ¿Qué relación existe con esto?. Un megapixel (Mpx) equivale a 1 millón de píxeles, a diferencia de otras medidas usadas en la computación en donde se suele utilizar la base de 1024 para los prefijos, en lugar de 1000, debido a su conveniencia respecto del uso del sistema binario. Usualmente se utiliza esta unidad para expresar la resolución de imagen de cámaras digitales; por ejemplo, una cámara que puede tomar fotografías con una resolución de 2048 × 1536 píxeles se dice que tiene 3.1 megapixeles ($2048 \times 1536 = 3.145.728$).

Para el caso de una imagen de 700×525 pixeles en tres colores almacenada sin compresión tendremos que el tamaño de la imagen será igual a $700 \times 525 \times 3 \approx 1.1$ MB (Mega Bites).

En Forma matricial podemos hacer una imagen con colores verdes si definimos, un arreglo bidimensional donde ponemos un vector de colores {R,G,B}

$$\begin{pmatrix} \begin{pmatrix} 0 \\ 255 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 255 \\ 0 \end{pmatrix} \end{pmatrix}$$

Para este ejemplo si nuestro arreglo de imagen se llama A los colores almacenados en el renglón i columna j, la notación para referirnos a este punto es $A_{i,j}$

1.1.2. La escala de gris

En el caso de una imagen en tono de gris solamente se maneja una sola matriz de datos, así el color rojo será igual que el verde y el azul. La unión de los colores dará como resultado blanco y la ausencia dará negro. Nuestra imagen solamente tendrá colores que van desde el negro al blanco en 256 tonos diferentes.

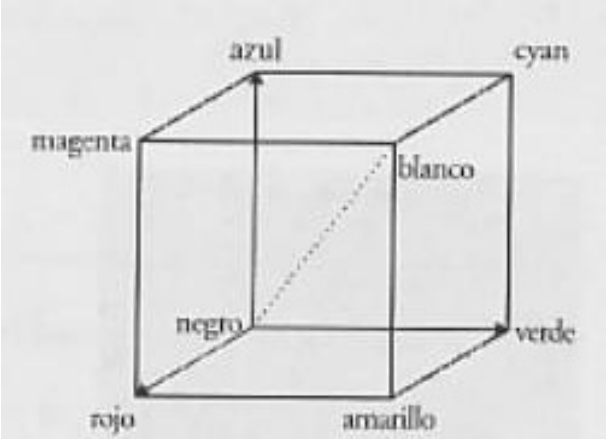


Figura 1.3: Cubo de colores y escala de gris

)

Utilizando imágenes en tono de gris, solamente será necesario almacenar la información concerniente a un canal de color, o un promedio de los tres canales de color. Esto reduce en 1/3 la información que debe ser almacenada.

1.1.3. Formato PPM para imágenes

Actualmente existen una infinidad de formatos para almacenamiento de imágenes. Los formatos más populares como el JPEG realizan la compresión de la imagen con perdida de información. Uno de los formatos mas simples es el Portable Pixel Map (PPM por sus siglas en ingles) y a continuación se muestra un ejemplo de este archivo.

```
P3
# P3 significa que los colores están en ASCII
# 3 columnas
# 2 renglones
# 255 para el color maximo
```

```
# El archivo termina con 6 tripletas RGB (18 valores enteros en ASCII)
3 2
255
255 0 128 32 255 0 64 32 255 25 55 10 255 11 5 0 0 0
```

La simpleza del formato PPM permite entender la información que se almacena en forma directa y escribir un código relativamente simple para escribir imágenes en este formato. Si escribimos en un archivo de texto lo mostrado en en las instrucciones anteriores, le damos la extensión PPM al archivo, tendremos una imagen con 6 píxeles a colores.

1.1.4. Interfase para desplegar imágenes en Java.

Ver Algoritmos y ejemplos en la página. ver inicial.java

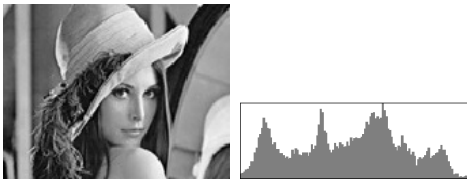
1.2. Operadores Sobre imágenes

1.2.1. Operadores puntuales sobre Imágenes

Una imagen discreta, la podemos definir como un arreglo bidimensional en el cual se almacena la información de cada punto de la imagen. Como definimos en la sección anterior cada unidad es un pixel y en el caso de operadores puntuales consideraremos que cada uno de estos se aplica sobre cada pixel de la imagen.

Linea Recta

Dada la imagen que se muestra en la figura 1.4(a) y su histograma 1.4(b) en tono de gris, podemos calcular una transformación lineal que nos permita cambiar el contraste de la imagen.



(a) ***Imagen original*** **(b)** ***Histograma***

Figura 1.4: Imagen original de Lena y su histograma en tono de gris

Consideremos el caso de la ecuación de una línea recta $y = mx + b$ donde m es la pendiente y b el cruce por cero.

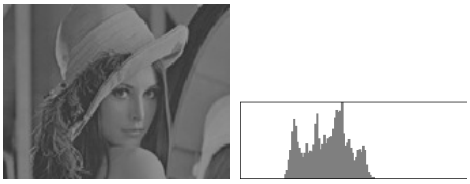
$$g_{i,j} = (f_{i,j} - f_{min}) * (g_{max} - g_{min})/(f_{max} - f_{min}) + g_{min};$$

donde el rango dinámico de la imagen dada es $f_{min} = 0$ y $f_{max} = 255$ y el rango de la nueva imagen es g_{min} y g_{max} los cuales fijaremos en el rango [10, 50]. La ecuación resultante es

$$g_{i,j} = \frac{8}{51} f_{i,j} + 10$$

$$\forall < i, j > \in \Omega$$

Así para una imagen dadas estas condiciones podemos cambiar la apariencia haciendo una proyección del espacio original de la imagen a un nuevo espacio dado por una línea recta. El resultado de la aplicación a cada punto se presenta en la figura 1.2.1. En la figura 1.5(b) podemos ver como la aplicación de este operador reduce el rango dinámico de la imagen reduciendo el rango del histograma en la misma medida.



(a) ***Imagen o*** **(b)** ***Histograma***

Figura 1.5: Imagen de Lena modificada y su histograma en tono de gris

En el caso de una imagen con mayor información esta lucirá como:

Cambio de color a tono de Gris

Otro ejemplo de operadores puntuales sobre imágenes es la conversión de colores RGB a escala de gris. Así para convertir la imagen a colores a tono de gris, simplemente sacamos el promedio de cada uno de los pixeles en la 3 bandas aplicando la operación

$$A_{i,j} = (R_{i,j} + G_{i,j} + B_{i,j})/\,3$$

Con esta operación tendremos una sola matriz con el promedio de los colores y cuando se crea la imagen a color las matrices R, G, B serán iguales a la matriz A.

Función Sigmoide

Otra manera de hacer esto es considerar una función diferente como la sigmoide, definida como

$$y = \frac{1}{1 + e^{-\beta x}}$$

donde β es un parámetro que controla la pendiente del escalón. Para hacer la aplicación de la función sigmoide, será necesario hacer un cambio al rango $[-1, 1]$ de la imagen original ya que el dominio y rango de la función sigmoide es $[-\infty, \infty]$ y $[0, 1]$ respectivamente.

Como luce una imagen en estas condiciones. En la figura 1.2.1, podemos ver la aplicación de la función sigmoide con $\beta = 100$ a la imagen original en la figura 1.4(a). En esta figura podemos ver la, la imagen procesada, el histogramas de la imagen y la gráfica de la sigmoide aplicada.

(a) ***Imagen o*** **(b)** ***Histograma*** **(c)** ***Sigmoide con*** $\beta = 100$

Figura 1.6: Imagen de Lena modificada por la función sigmoide

1.2.2. Operadores de Ventana sobre imágenes

El objetivo de un operador de ventana es reemplazar el valor de un pixel, por el contenido pesado en una ventana al rededor de esta. Así por ejemplo:

$$I_{i,j} = \sum_{k=i-nk}^{i+nk} \sum_{l=j-nk}^{j+nk} w_{k,l} I_{k,l.}$$

donde $w_{k,l}$ es un peso que puede tener varios valores con lo cual tenemos una posibilidad casi infinita de operadores. En la figura 1.7 tenemos un operador correspondiente a la media, donde cada uno de los pixeles es reemplazado por el promedio en una ventana de tamaño $(2n_k + 1) \times (2n_k + 1)$ y con $w_{k,l} = \frac{1}{(2n_k+1)^2}$.

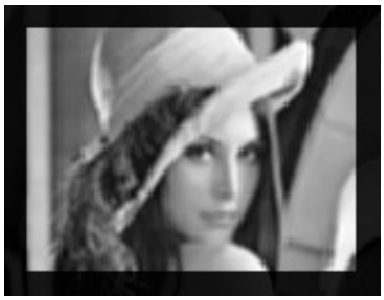


Figura 1.7: Lena aplicando un operador de ventana que simula la media

1.3. Transformaciones Geométricas

En esta sección básicamente analizaremos dos transformaciones geométricas en imágenes. Estas son la transformación Afín y la transformación proyectiva.

1.3.1. Transformación Afín

Una transformación afín es una transformación geométrica que esta constituida por translación, escalamiento, rotación y cizallamiento. Cada una de esas transformaciones es una transformación afín y se explican a continuación.

Traslación

Una translación la podemos hacer simplemente asumiendo que las nuevas coordenadas $\hat{x} = x + t_x$ $\hat{y} = y + t_y$ les sumamos un valor t_x o t_y según corresponda. En coordenadas homogéneas queda como

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

La Figura 1.8 muestra un ejemplo de una translación de un cuadro de color azul y el resultado del cuadro trasladado en color rojo. En este caso se aplico una translación dada

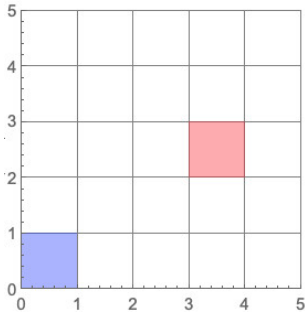


Figura 1.8: Aplicación de una translación a un cuadro

por la ecuación (1.1).

$$T = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.1}$$

Escalamiento

El escalamiento pude entenderse como hacer que una figura geométrica cambie su tamaño o cambie su escala. Un escalamiento en x lo podemos representar por como $\hat{x} = x\ s_x$ y en y como $\hat{y} = y\ s_y$. En coordenadas homogéneas se pude expresar como

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

La Figura 1.9 muestra un ejemplo de la aplicación de un escalamiento a un cuadro de color azul y el resultado del cuadro escalado en color rojo. En este caso se aplicó un escalamiento dada por la ecuación (1.2).

$$S = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.2}$$

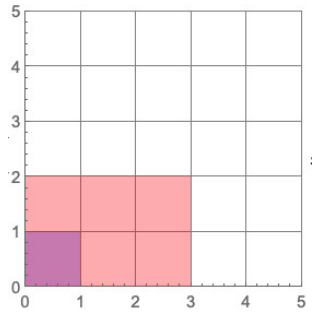


Figura 1.9: Aplicación de un escalamiento a un cuadro

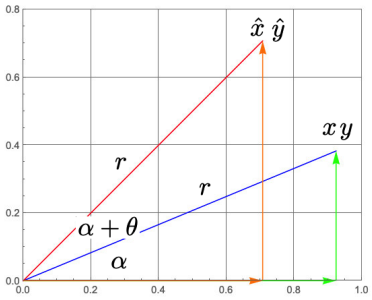


Figura 1.10: Coordenadas x, y girada un ángulo θ

Rotación

Consideremos el caso de un punto que rota respecto a un punto fijo tal como se muestra en la Figura 1.10. Las coordenadas x y y, en forma polar las podemos obtener como $x = r\ \cos(\alpha)$ y $y = r\ \sin(\alpha)$. Si consideramos que esta gira un ángulo θ entonces podemos representar esta rotación en forma polar como

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} r\ \cos(\alpha+\theta) \\ r\ \sin(\alpha+\theta) \end{pmatrix} = \begin{pmatrix} r\ \cos(\alpha)\cos(\theta) - r\ \sin(\alpha)\sin(\theta) \\ r\ \cos(\alpha)\sin(\theta) + r\ \sin(\alpha)\cos(\theta) \end{pmatrix}$$

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} x\ \cos(\theta) - y\ \sin(\theta) \\ x\ \sin(\theta) + y\ \cos(\theta) \end{pmatrix}$$

Una transformación de rotación puede ser definida como

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

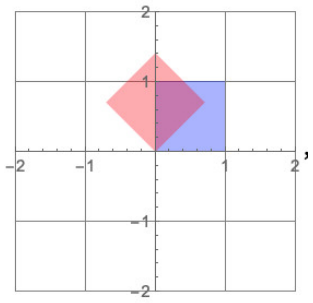


Figura 1.11: Aplicación de una rotación a un cuadro

y en coordenadas homogéneas puede representarse como

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$R(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

La Figura 1.11 muestra un ejemplo de la aplicación de una rotación de 45 grados ($\pi/4$ rad) a un cuadro de color azul y el resultado del cuadro rotado en color rojo. En este caso se aplicó una matriz de rotación dada por la ecuación (1.3).

$$R(\pi/4) = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.3}$$

Cizallamiento

El cizallamiento es una transformación dada por la matriz, donde c_x es el ángulo de cizallamiento respecto al eje x.

$$C_x = \begin{pmatrix} 1 & tg(c_x) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

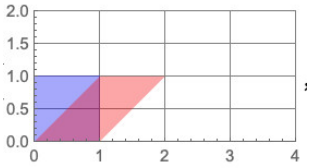


Figura 1.12: Aplicación de un cizallamiento a un cuadro

En caso de querer aplicar el cizallamiento en la dirección y, la matriz de transformación será.

$$C_y = \begin{pmatrix} 1 & 0 & 0 \\ tg(c_y) & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

La Figura 1.12 muestra un ejemplo de la aplicación de un cizallamiento de 45 grados ($\pi/4$ rad.) en la dirección de x a un cuadro de color azul y el resultado del cuadro cizallado en color rojo. En este caso se aplicó un cizallamiento dado por la ecuación (1.4).

$$C_x = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.4}$$

Una vez que unimos todas las transformaciones tenemos una composición dada por una Rotación R, un escalamiento S, un Cizallamiento C y una Translación T. Al conjunto de estas operaciones se le conoce como transformación afín.

$$T_{afin} = \begin{pmatrix} s_x\ cos(\theta) & -s_y\ sin(\theta) + s_x\ cos(\theta)\ tan(c_x) & t_x \\ s_x\ sin(\theta) & s_y\ cos(\theta) + s_x\ sin(\theta)\ tan(c_x) & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

1.3.2. Interpolación.

Todas las transformaciones aplicadas anteriormente son para un punto en el espacio de coordenadas $[x, y]$, pero qué sucede cuando hablamos de imágenes. Una imagen esta compuesta de colores en una posición con coordenadas enteras $[n, m]$. Si aplicamos una transformación afín las coordenadas enteras pasaran a ser coordenadas en números reales.

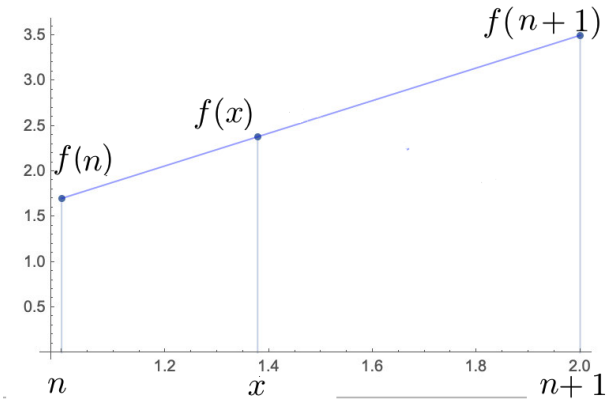


Figura 1.13: Interpolación lineal en una dimensión

Interpolación en 1 dimensión

La Figura 1.13 muestra el ejemplo con una señal en el intervalo $[1, 2]$ donde $f(1) = 1.7$ y $f(2) = 3.5$ y queremos encontrar los valores para una $f(1.378)$. Para nuestros propósito suponemos que $n \in \mathbb{N}$ es un número entero y dado que no tenemos ninguna información a priori los puntos entre $1.0 < x < 2.0$ tiene un comportamiento lineal tal como se muestra en la Figura 1.13.

Utilizando la ecuación de la línea recta tenemos dos puntos $P_1 = [n, f(n)]$ y $P_2 = [n + 1, f(n + 1)]$, la ecuación de la linea recta esta dada por

$$f(x) - f(n) = \frac{f(n+1) - f(n)}{n + 1 - n}(x - n)$$

Considerando que $x = n + \alpha$ tenemos

$$f(x) - f(n) = (f(n + 1) - f(n))\alpha$$

$$f(x) = f(n + 1)\alpha + f(n)(1 - \alpha)$$

Para el ejemplo $x = 1.378$, tenemos $n = 1$ y $\alpha = 0.378$ por lo tanto $f(1.378) = f(1)(1 - \alpha) + f(2)\alpha = 1.7 \times (1 - 0.378) + 3.5 \times 0.378 = 2.3804$,

Interpolación en 2 dimensiones

Para realizar la interpolación en dos dimensiones hacemos la aplicación de una interpolación en cada una de la dimensiones x y y. Para este caso tenemos cuatro puntos $P_1 = [n, m, f(n, m)]$, $P_2 = [n, m + 1, f(n, m + 1)]$ $P_3 = [n + 1, m, f(n + 1, m)]$ y $P_4 = [n + 1, m + 1, f(n + 1, m + 1)]$.

Vamos a suponer que tenemos que calcular el valor de la función en dos dimensiones $f(u, v)$ donde $u \in \mathbb{R}$ y $v \in \mathbb{R}$ entonces tenemos un valor $u = n + \alpha$ y $v = m + \beta$ donde $n \in \mathbb{N}$ y $m \in \mathbb{N}$ y podemos escribir:

$$f(u, m) = f(n, m)(1 - \alpha) + f(n + 1, m)\alpha$$

$$f(u, m + 1) = f(n, m + 1)(1 - \alpha) + f(n + 1, m + 1)\alpha$$

y finalmente

$$f(u, v) = f(u, m)(1 - \beta) + f(u, m + 1)\beta$$

$$\begin{aligned} f(u, v) = f(n, m)(1 - \alpha)(1 - \beta) + f(n + 1, m)\alpha(1 - \beta) \\ + f(n, m + 1)(1 - \alpha)\beta + f(n + 1, m + 1)\alpha\beta \end{aligned}$$

La función de interpolación bilineal en Java es

```
static public double bilineal(double f[][], double x0, double y0)
  {
   int ii, jj, nr, nc;
   double a, b, y;

   ii = (int) y0;
   jj = (int) x0;

   a = y0 - ii;
   b = x0 - jj;

   nr = f.length;
   nc = f[0].length;

   if (ii < 0 || jj < 0 || ii > nr - 2 || jj > nc - 2)
```

```
 return 0;
if ( (a == 0) && (b == 0))
 return f[ii][jj];

if (ii == nr - 1 && jj == nc - 1)
 return f[ii][jj];

if (b == 0 || jj == nc - 1) {
 y = (1 - a) * f[ii][jj] + a * f[ii + 1][jj];
 return y;
}

if (a == 0 || ii == nr - 1) {
 y = (1 - b) * f[ii][jj] + b * f[ii][jj + 1];
 return y;
}

y = (1 - a) * (1 - b) * f[ii][jj] + (1 - a) * b * f[ii][jj + 1] +
  a * (1 - b) * f[ii + 1][jj] + a * b * f[ii + 1][jj + 1];
return y;
}
```

1.3.3. Problemas con la transformación Afín

Para una imagen origen I_1 las coordenadas del renglón r y la columna c son números enteros. Al aplicar una transformación T dará como resultado coordenadas en el renglón r_n y columna c_n que son números reales y deben ser asignadas a coordenadas enteras en la imagen destino I_2. La Figura 1.14 muestra este problema donde el punto en coordenadas $r = 2$ y $c = 2$ de la imagen $I1$, para una cierta transformación T, corresponde al punto $r_n = 3.7$ y $c_n = 2.5$ de la imagen $I2$. Esto significa que tenemos que poner el color en el pixel de la imagen origen $I_1[2, 2]$ en un pixel que no existe en la imagen destino $I_2[3.7, 2.5]$. Una manera de resolver el problema es truncar las coordenadas aplicando la operación piso y entonces hacer $I_2[\lfloor 3.7 \rfloor, \lfloor 2.5 \rfloor] = I_2[3, 2] \equiv I_1[2, 2]$.

Consideremos el caso de una matriz de rotación dada por:

$$R = \begin{pmatrix} 0.951057 & -0.309017 & 0. \\ 0.309017 & 0.951057 & 0. \\ 0. & 0. & 1. \end{pmatrix}$$

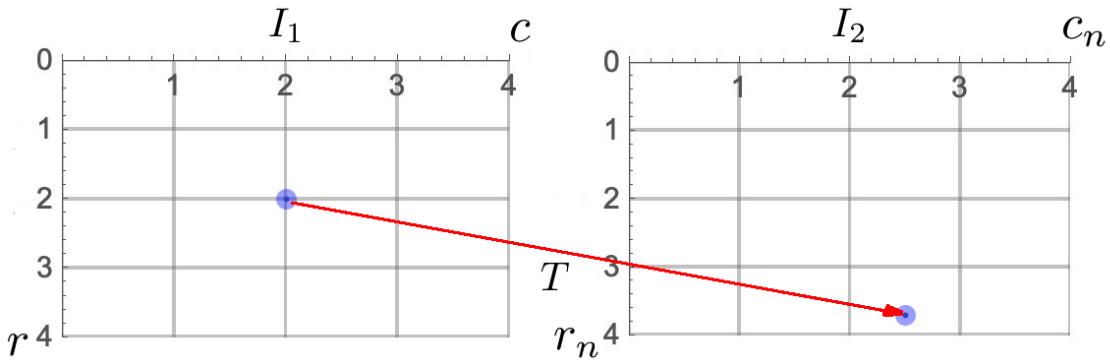


Figura 1.14: Puntos correspondientes en imágenes I_1 e I_2

y las coordenadas las podemos calcular

$$\begin{bmatrix} c_n \\ r_n \end{bmatrix} = \begin{bmatrix} 0.951057 & -0.309017 & 0. \\ 0.309017 & 0.951057 & 0. \end{bmatrix} \begin{bmatrix} c \\ r \end{bmatrix}$$

de acuerdo con lo expuesto al aplicar la transformación, para cada uno de los puntos de la imagen origen I_1 tendrán su equivalente en I_2. Todo esto se puede realizar mediante el siguiente código y la imagen destino I_2 resultante se muestra en la Figura 1.15. Note los puntos negros en la Fig. 1.15, generados por el truncamiento.

```
for(r=0; r<Nr; r++)
  for(c=0; c<Nc; c++) {
      cn = T[0]*c + T[1]*r + T[2];
      rn = T[3]*c + T[4]*r + T[5];

      I2[(int) rn][(int) cn] = I1[r][c];
  }
```

¿Cómo corregir este efecto?. Una manera de hacerlo es en lugar de barrer todos los puntos de la imagen origen I_1, barreremos todos los puntos de la imagen destino I_2 y las coordenadas correspondientes en la imagen origen las calculamos utilizando la transformación inversa. Las coordenadas nuevas que no sean enteras las redondearemos como el piso de los valores reales $I_2[r, c] \equiv I_1[\lfloor r_n \rfloor, \lfloor c_n \rfloor]$. En la Figura 1.16 se muestra un ejemplo de cómo las coordenadas enteras en la imagen destino $I2$ vienen de unas coordenadas reales en la imagen origen I_1.

Figura 1.15: Lena aplicando una transformación de rotación $R(18°)$

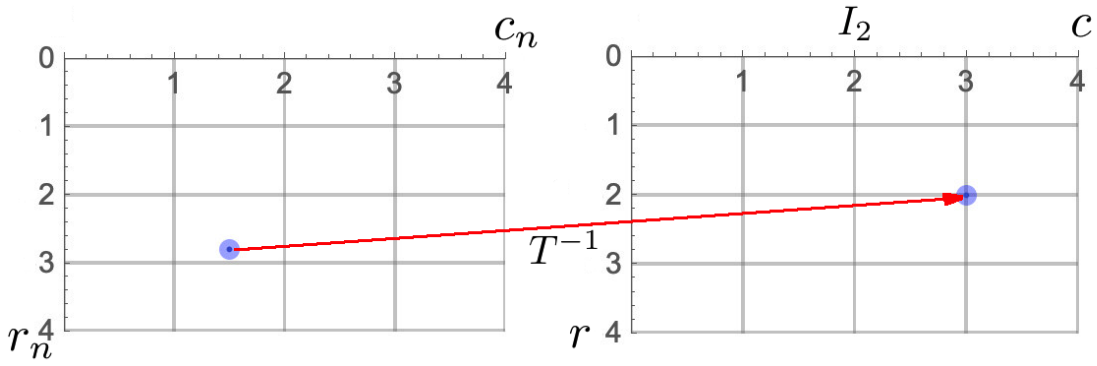


Figura 1.16: Puntos correspondientes en imágenes I_1 e I_2 aplicando la transformación inversa

El código para hacer esto es el que se muestra a continuación.

```
for(r=0; r<Nr; r++)
  for(c=0; c<Nc; c++) {
      cn = Tinv[0]*c + Tinv[1]*r + Tinv[2];
      rn = Tinv[3]*c + Tinv[4]*r + Tinv[5];


      I2[r][c] = I2[rn][cn]
  }
```

El código para la transformación afín resume todo lo antes dicho agregando la interpolación bilineal para evitar hacer el truncamiento de las coordenadas. Este código se muestra a continuación:

```
 static public void afin(double f[][], double g[][], double T[])
{
   int r, c, Nr, Nc;

   double rn, cn;

   Nr = f.length;
   Nc = f[0].length;


   for (r = 0; r < Nr; r++)
    for (c = 0; c < Nc; c++) {

     cn = T[0]*c  + T[1]*r  + T[2];
     rn = T[3]*c  + T[4]*r  + T[5];
     I2[r][n] = bilineal(I1, cn, rn);

    }
  }
```

La Figura 1.17 podemos ver la diferencia entre la aplicación de la transformación inversa con interpolación 1.17 (a) y la aplicación de la transformación directa 1.17 (b).

Ejemplo

Considere una imagen I_1 dada como

$$I_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \\ 19 & 20 & 21 & 22 & 23 & 24 \end{bmatrix}$$

y la Transformación

(a) ***Aplicando*** $R(-18°)$

(b) ***Aplicando*** $R(18°)$

Figura 1.17: Transformación Afín. (a) Aplicando transformación inversa e interpolación. (b) aplicando la transformacion directa

$$T = \begin{bmatrix} \frac{3}{2} & 0 & 0 \\ 0 & \frac{3}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T^{-1} = \begin{bmatrix} \frac{2}{3} & 0 & 0 \\ 0 & \frac{2}{3} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Vamos a Determinar la imagen I_2 considerando la transformación T y como referencia los puntos de la Imagen I_1. Para este caso tenemos los resultados en la Tabla 1.1.

Ahora vamos a Determinar la imagen I_2 considerando la transformación T^{-1} y como referencia los puntos de la Imagen I_2. Para este caso las coordenadas quedan como 1.2.

Finalmente en cada uno de los casos la imagen I_2 para los tres casos queda como:

Cuadro 1.1: Puntos calculados con la transformación T y con referencia en la imagen I_1

r	c	r_n	c_n	$I_2(\lfloor r_n \rfloor, \lfloor c_n \rfloor) = I_1(r, c)$
0	0	0.000000	0.000000	$I_2(0,0) = I_1(0,0)$
0	1	0.000000	1.500000	$I_2(0,1) = I_1(0,1)$
0	2	0.000000	3.000000	$I_2(0,3) = I_1(0,2)$
0	3	0.000000	4.500000	$I_2(0,4) = I_1(0,3)$
0	4	0.000000	6.000000	$I_2(0,6) = I_1(0,4)$
0	5	0.000000	7.500000	$I_2(0,7) = I_1(0,5)$
1	0	1.500000	0.000000	$I_2(1,0) = I_1(1,0)$
1	1	1.500000	1.500000	$I_2(1,1) = I_1(1,1)$
1	2	1.500000	3.000000	$I_2(1,3) = I_1(1,2)$
1	3	1.500000	4.500000	$I_2(1,4) = I_1(1,3)$
1	4	1.500000	6.000000	$I_2(1,6) = I_1(1,4)$
1	5	1.500000	7.500000	$I_2(1,7) = I_1(1,5)$
2	0	3.000000	0.000000	$I_2(3,0) = I_1(2,0)$
2	1	3.000000	1.500000	$I_2(3,1) = I_1(2,1)$
2	2	3.000000	3.000000	$I_2(3,3) = I_1(2,2)$
2	3	3.000000	4.500000	$I_2(3,4) = I_1(2,3)$
2	4	3.000000	6.000000	$I_2(3,6) = I_1(2,4)$
2	5	3.000000	7.500000	$I_2(3,7) = I_1(2,5)$
3	0	4.500000	0.000000	$I_2(4,0) = I_1(3,0)$
3	1	4.500000	1.500000	$I_2(4,1) = I_1(3,1)$
3	2	4.500000	3.000000	$I_2(4,3) = I_1(3,2)$
3	3	4.500000	4.500000	$I_2(4,4) = I_1(3,3)$
3	4	4.500000	6.000000	$I_2(4,6) = I_1(3,4)$
3	5	4.500000	7.500000	$I_2(4,7) = I_1(3,5)$

Cuadro 1.2: Puntos calculados con la transformación inversa y con referencia en la imagen 2

r	c	r_n	c_n	$I_2(r,c) = I_1(\lfloor r_n \rfloor, \lfloor c_n \rfloor)$
0	0	0.000000	0.000000	$I_2(0,0) = I_1(0,0)$
0	1	0.666667	0.000000	$I_2(0,1) = I_1(0,0)$
0	2	1.333334	0.000000	$I_2(0,2) = I_1(1,0)$
0	3	2.000001	0.000000	$I_2(0,3) = I_1(2,0)$
0	4	2.666668	0.000000	$I_2(0,4) = I_1(2,0)$
0	5	3.333335	0.000000	$I_2(0,5) = I_1(3,0)$
1	0	0.000000	0.666667	$I_2(1,0) = I_1(0,0)$
1	1	0.666667	0.666667	$I_2(1,1) = I_1(0,0)$
1	2	1.333334	0.666667	$I_2(1,2) = I_1(1,0)$
1	3	2.000001	0.666667	$I_2(1,3) = I_1(2,0)$
1	4	2.666668	0.666667	$I_2(1,4) = I_1(2,0)$
1	5	3.333335	0.666667	$I_2(1,5) = I_1(3,0)$
2	0	0.000000	1.333334	$I_2(2,0) = I_1(0,1)$
2	1	0.666667	1.333334	$I_2(2,1) = I_1(0,1)$
2	2	1.333334	1.333334	$I_2(2,2) = I_1(1,1)$
2	3	2.000001	1.333334	$I_2(2,3) = I_1(2,1)$
2	4	2.666668	1.333334	$I_2(2,4) = I_1(2,1)$
2	5	3.333335	1.333334	$I_2(2,5) = I_1(3,1)$
3	0	0.000000	2.000001	$I_2(3,0) = I_1(0,2)$
3	1	0.666667	2.000001	$I_2(3,1) = I_1(0,2)$
3	2	1.333334	2.000001	$I_2(3,2) = I_1(1,2)$
3	3	2.000001	2.000001	$I_2(3,3) = I_1(2,2)$
3	4	2.666668	2.000001	$I_2(3,4) = I_1(2,2)$
3	5	3.333335	2.000001	$I_2(3,5) = I_1(3,2)$

Referencia	I_2
I_1 con T	$\begin{bmatrix} 1.0000 & 2.0000 & 0.0000 & 3.0000 & 4.0000 & 0.0000 \\ 7.0000 & 8.0000 & 0.0000 & 9.0000 & 10.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 13.0000 & 14.0000 & 0.0000 & 15.0000 & 16.0000 & 0.0000 \end{bmatrix}$
I_2 con T^{-1}	$\begin{bmatrix} 1.0000 & 1.0000 & 7.0000 & 13.0000 & 13.0000 & 19.0000 \\ 1.0000 & 1.0000 & 7.0000 & 13.0000 & 13.0000 & 19.0000 \\ 2.0000 & 2.0000 & 8.0000 & 14.0000 & 14.0000 & 20.0000 \\ 3.0000 & 3.0000 & 9.0000 & 15.0000 & 15.0000 & 21.0000 \end{bmatrix}$
I_2 con T^{-1} interpolada	$\begin{bmatrix} 1.0000 & 1.6667 & 2.3333 & 3.0000 & 3.6667 & 4.3333 \\ 5.0000 & 5.6667 & 6.3333 & 7.0000 & 7.6667 & 8.3333 \\ 9.0000 & 9.6667 & 10.3333 & 11.0000 & 11.6667 & 12.3333 \\ 13.0000 & 13.6667 & 14.3333 & 15.0000 & 15.6667 & 16.3333 \end{bmatrix}$

ver (afin.java)

1.3.4. Transformación Log Polar

La geometría log polar en imágenes fue motivada por la estructura de la retina de algunos sistema de vision y esta tiene algunas propiedades de compresión. Cuando comparamos el sistema usual de coordenadas cartesianas en imágenes, las imágenes log - polar permiten muestreo mas rápido en los sistema de visión artificial sin reducción en el tamaño de la vista y la parte centra de la retina (fovea).

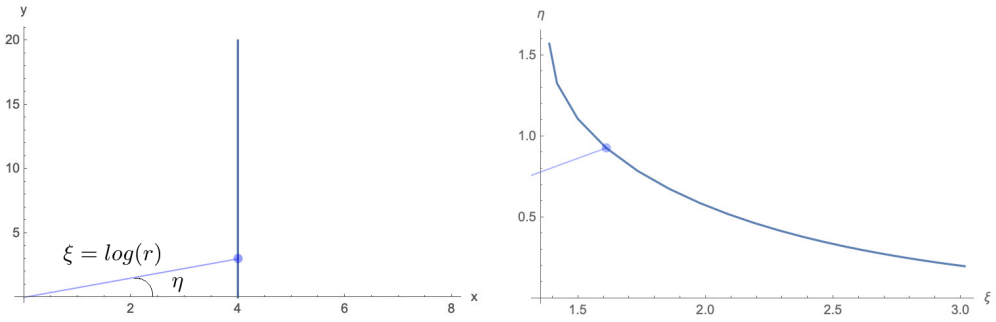
La transformación log-polar es un mapeo de puntos en el plano cartesiano (x, y) a puntos en el plano log-polar (ξ, η). La Figura 1.18 (a) muestra un conjunto de puntos en linea recta a los que se les aplica la transformación log-polar y la Figura 1.18 (b) muestra como quedan representados esos puntos. Los valores graficados en la Figura 1.18 se presentan en la tabla 1.3. La manera de llevar a cabo la transformación es mediante:

$$\xi = ln\left(\sqrt{x^2 + y^2}\right)$$

$$\eta = tg^{-1}\left(\frac{y}{x}\right)$$

Aplicando esto tenemos que una imagen luce como la figura 1.19

ver (logpolar.java)



(a) (b)

Figura 1.18: Interpolación lineal en una dimensión

n	x_n	y_n	ξ_n	η_n
1	4.	0.	1.38629	0
2	4.	1.	1.41661	0.2450
3	4.	2.	1.49787	0.4626
4	4.	3.	1.60944	0.6435
5	4.	4.	1.73287	0.7854
6	4.	5.	1.85679	0.8961
7	4.	6.	1.97562	0.9828
8	4.	7.	2.08719	1.0517
9	4.	8.	2.19101	1.1071
10	4.	9.	2.28736	1.1526
11	4.	10.	2.3768	1.1903
12	4.	11.	2.45999	1.2220
13	4.	12.	2.53759	1.2490
14	4.	13.	2.61018	1.2723
15	4.	14.	2.67829	1.2925
16	4.	15.	2.7424	1.3102
17	4.	16.	2.8029	1.3258
19	4.	17.	2.86016	1.3397
20	4.	18.	2.91447	1.3521
21	4.	19.	2.96612	1.3633
22	4.	20.	3.01534	1.3734

Cuadro 1.3: Transformación de puntos de coordenadas x, y a coordenadas polares ξ, η

Figura 1.19: Lena aplicando una transformación log polar

```
static public void LogPolar (double datos[][], double res[][]) {   int nr, nc,
i, j, ren, col;
  double alfa, r, x, y;
  nr = datos.length;
  nc = datos[0].length;
  ren = res.length;
  col = res[0].length;
  System.out.println (ren + + col);
  double rmax = Math.sqrt (ren*ren + col*col)/2.0;
  for (i = 0; i < ren; i++)
   for (j = 0; j < col; j++) {
    alfa = (2.0*Math.PI*i)/(ren - 2);
    r = Math.exp (j*Math.log (rmax)/col);
    x = r*Math.cos (alfa) + nc/2.0;
    y = r*Math.sin (alfa) + nr/2.0;
    res[i][j] = bilineal (datos, x, y);
   }
}
```

La transformación log-polar inversa se calcula con el código

```
static public void InvLogPolar (double datos[][], double[][] res)
{
 int nr, nc, i, j, ii, jj, ren, col;
 double alfa, r;
 double fac = 65.0;
 nr = datos.length;
 nc = datos[0].length;
```

```
 ren = res.length;
 col = res[0].length;
 double phimax = Math.log (Math.sqrt (nr*nr + nc*nc)/2.0);
 for (i = 0; i < ren; i++)
  for (j = 0; j < col; j++) {
   ii = ren/2 - i;
   jj = col/2 - j;
   alfa = (Math.atan2 (ii, jj)/Math.PI + 1.0)*(double) (nr - 2)/2.0;
   r = Math.log (Math.sqrt (ii*ii + jj*jj))/phimax*nc;
   res[i][j] = bilineal (datos, r, alfa);
  }
}
```

1.3.5. Transformación Proyectiva

Una transformación proyectiva es la generalización de las transformaciones $R^2 \rightarrow R^2$, en la que las líneas paralelas no son transformadas necesariamente en tales. Se puede expresar de la forma

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} A & t \\ v^T & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

donde v^T es la diferencia entre la transformación proyectiva y la afín. Note para el caso de la transformación proyectiva la coordenada $\hat{z}$ no necesariamente es unitaria lo cual implica que la imagen transfromada se fue a un plano con coordenadas diferentes a esta. Una manera de regresar al sistema de coordenadas es dividir entre $\hat{z}$. Así la matriz de transformación queda

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{pmatrix} = \frac{1}{\hat{z}} \begin{pmatrix} A & t \\ v^T & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Esta ecuación es conocida también como homógrafía. La aplicación a una imagen de una transformación

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1e-3 & 0 & 1 \end{pmatrix}$$

Aplicando esta transformación tenemos una proyección de la imagen de Lena que se muestra en la figura 1.20

Figura 1.20: Transformación Proyectiva en la imagen de Lena

El código Java para hacer esta transformación es:

```
static public void proyeccción (double f[][], double g[][], double T[])
{
 int i, j, nr, nc;
 double in, jn, kn;
 nr = f.length;
 nc = f[0].length;
 for (i = 0; i < nr; i++)
  for (j = 0; j < nc; j++) {
  // System.out.println (T[0]);
  jn = T[0]*j + T[1]*i + T[2];
  in = T[3]*j + T[4]*i + T[5];
  kn = T[6]*j + T[7]*i + 1.0;
  jn /= kn;
  in /= kn;
  g[i][j] = bilineal (f, jn, in);
  }
 }
```

Parámetros a partir de puntos correspondientes

En esta sección vamos a calcular los parámetros de una homografía, la cual representa un modelo proyectivo, dada por la ecuación

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} A & t \\ v^T & k \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$$

donde el vector $P_1 = [x_1, y_1, z_1]^T$ es un punto en el espacio tridimensional, el vector $P_2 = [x_2, y_2, z_2]^T$ es la proyección en un plazo con $z_2 = 1$y ambos puntos son correspondientes $P_1 \longleftrightarrow P_2$

El problema consiste en determinar el vector $H = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8]^T$ a partir de un conjunto de puntos correspondientes. Una manera de resolver este problema es utilizando mínimos cuadrados, sin embargo tiene problema de estabilidad numérica por lo que es preferible utilizar el método siguiente.

Dado los puntos y que estos son correspondientes, entonces podemos decir que $P_2 \times HP_1 = 0$, dado que ambos vectores on paralelos. Esto no lleva a la siguiente representación.

$$\begin{pmatrix} 0^T & -z_{2,i}P_{1,i}^T & y_{2,i}P_{1,i}^T \\ z_{2,i}P_{1,i}^T & 0^T & -x_{2,i}P_{1,i}^T \end{pmatrix} \begin{pmatrix} H_1 \\ H_2 \\ H_3 \end{pmatrix} = 0$$

donde $H_1 = [h_1, h_2, h_3]^T$, $H_2 = [h_4, h_5, h_6]^T$ y $H_3 = [h_7, h_8, h_9]^T$.

La solución del sistema de ecuaciones puede llevarse a cabo utilizando la descompisición en valores singulares SVD. La descomposición en valores singulares calcula tres matrices USV = SVD(A) donde U y V son matrices y S una matriz diagonal. La solución del sistema es el último vector de la matriz V (ver Harley-Zisserman pp 90).

Ejemplo

Dados los puntos P1 = { [10.0 10.0 1.0], [10.0 50.0 1.0], [50.0 50.0 1.0], [50.0 10.0 1.0], [25.0 25.0 1.0]} y P2 = {[8.9108, 10.89108, 1.0], [4.95049, 46.53465, 1.0], [39.04761, 48.57142, 1.0], [42.85714, 14.28571, 1.0], [20.48780, 25.36585, 1.0], determinar la Homografía correspondiente. La siguiente imagen muestra los puntos P1 a la izquierda y los puntos P2 a la derecha.

Figura 1.21: Puntos correspondientes

De acuerdo con la ecuación

$$\begin{pmatrix} 0^T & -z_{2,i}P_{1,i}^T & y_{2,i}P_{1,i}^T \\ z_{2,i}P_{1,i}^T & 0^T & -x_{2,i}P_{1,i}^T \end{pmatrix} \begin{pmatrix} H_1 \\ H_2 \\ H_3 \end{pmatrix} = BH = 0$$

Tenemos

$$\mathrm{B} = \begin{pmatrix} 0. & 0. & 0. & -10. & -10. & -1. & 108.911 & 108.911 & 10.891 \\ 10. & 10. & 1. & 0. & 0. & 0. & -89.109 & -89.109 & -8.911 \\ 0. & 0. & 0. & -10. & -50. & -1. & 465.347 & 2326.73 & 46.535 \\ 10. & 50. & 1. & 0. & 0. & 0. & -49.505 & -247.525 & -4.95 \\ 0. & 0. & 0. & -50. & -50. & -1. & 2428.57 & 2428.57 & 48.571 \\ 50. & 50. & 1. & 0. & 0. & 0. & -1952.38 & -1952.38 & -39.048 \\ 0. & 0. & 0. & -50. & -10. & -1. & 714.286 & 142.857 & 14.286 \\ 50. & 10. & 1. & 0. & 0. & 0. & -2142.86 & -428.571 & -42.857 \\ 0. & 0. & 0. & -25. & -25. & -1. & 634.146 & 634.146 & 25.366 \\ 25. & 25. & 1. & 0. & 0. & 0. & -512.195 & -512.195 & -20.488 \end{pmatrix}$$

Haciendo la SVD(B) tenemos que la matriz v es

$$v = \begin{pmatrix} -0.008799 & -0.01804 & 0.4915 & 0.0774 & -0.4276 & -0.5010 & -0.07493 & 0.3712 & -0.4178 \\ -0.006549 & -0.001930 & 0.6014 & -0.7039 & 0.06790 & 0.3676 & -0.009472 & -0.02489 & 0.04643 \\ -0.0001982 & -0.0003293 & 0.01553 & -0.01201 & -0.01818 & 0.007560 & 0.7942 & -0.3909 & -0.4641 \\ -0.008647 & -0.002579 & -0.5090 & -0.7008 & -0.03908 & -0.4922 & 0.005096 & 0.05909 & -0.04643 \\ -0.01059 & 0.0176 & -0.3700 & -0.07726 & -0.4068 & 0.6088 & -0.06827 & 0.3757 & -0.41782 \\ -0.0002489 & 0.0002550 & -0.01247 & -0.01494 & -0.02256 & -0.005127 & -0.5945 & -0.6558 & -0.4642 \\ 0.7018 & 0.7119 & 0.01084 & -0.00820 & -0.01566 & -0.01418 & 0.00008070 & 0.0008968 & -0.0004646 \\ 0.7119 & -0.7018 & -0.01073 & -0.007749 & -0.01538 & 0.01458 & -0.0002675 & 0.0004953 & 2.6997e-7 \\ 0.01740 & -0.000201 & -0.001353 & 0.02651 & 0.8025 & -0.01339 & -0.07222 & 0.3660 & -0.4642 \end{pmatrix}$$

Tomando la última columna de la matriz v y normalizando respecto a $v_{9,9}$ tenemos que la Homografía es

$$\mathrm{H} = \begin{pmatrix} 0.900036 \\ -0.100016 \\ 0.999866 \\ 0.100021 \\ 0.899986 \\ 0.99995 \\ 0.00100087 \\ -5.8153e-7 \\ 1. \end{pmatrix}$$

Procesamiento de Señales

2.1. Definiciones.

Representaremos a una señal discreta x como un arreglo de tamaño N con muestras $[x[0], x[1], \cdots, x[k], \cdots x[N-1]]$ y una señal continua $\hat{x}$ como una función del tiempo $\hat{x}(t)$ donde $t \in \mathbb{R}$ es un número real que representa el tiempo en segundos. Esta señal discreta resulta multiplicar una señal continua $f(t)$ por un tren de pulsos $\delta(t - nT)$ representados por

$$x\left[n\right] = f\left(t\right)\delta(t - nT)$$

La función $\delta(t)$ es la delta de dirac la cual toma el valor unitario cuando su argumento es cero (ver Figura 2.1).

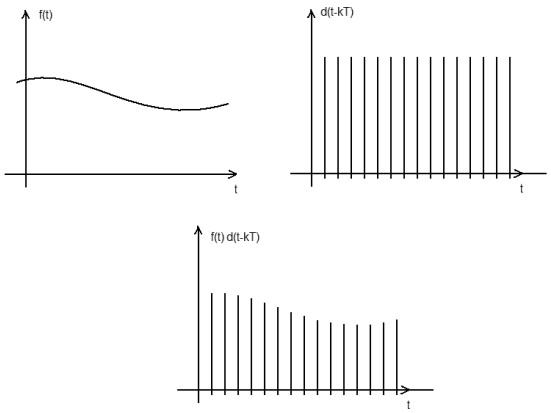


Figura 2.1: Multiplicación de una función por un tren de pulsos

Otra manera de muestrear un señal, es sustituyendo el tiempo t por un múltiplo de T

mediante

$$x[n] = f(nT) \tag{2.5}$$

donde $n \in \mathbb{N}$ es un número entero y T es conocido como periodo de muestreo. La función $f : \mathbb{R} \to \mathbb{R}$ es una función unidimensional.

2.2. Señales básicas de tiempo continuo.

A continuación se hace una descripción de algunas de las principales señales que estaremos utilizando y que son importantes para el desarrollo del curso.

2.2.1. Señales Sinusoidal

Definimos una señal continua x sinusoidal como

$$x(t) = sin(wt + \phi)$$

donde el tiempo t está en segundo, la w es definida cómo la velocidad angular la cual se expresa en rad/seg y ϕ es un ángulo de fase en radianes.

Para esta señal senoidal podemos definir la frecuencia fundamental f expresada en ciclos por segundo o Hertz como

$$f = \frac{w}{2\pi}$$

y el periodo T (segundos) como el inverso de la frecuencia f

$$T = \frac{1}{f} = \frac{2\pi}{w}$$

El periodo de la señal lo podemos interpretar como el tiempo que tarda en repetirse ciclo completo de la señal, es decir, una señal con periodo T volverá a ser la misma cada T segundos $x(t + T) = x(t)$.

La señal coseno puede expresarse como una señal seno desfasada un ángulo $\phi = \pi/2$.

$$x(t) = sin(wt + \pi/2) = cos(wt)$$

2.2.2. Exponencial compleja

La señal exponencial de tiempo continuo es de la forma

$$x\left(t\right) = Ce^{at}$$

donde C y a son, en general números complejos. Dependiendo de los valores de estos parámetros, la exponencial puede adoptar varias características diferentes. Una clase de exponenciales complejas que nos interesa es

$$x\left(t\right) = Me^{jwt}$$

donde $w = 2\pi f$ es definida como la velocidad angular en radianes por segundo y f es la frecuencia en ciclos por segundo o Hertz. Utilizando la relación de Euler esta señal puede expresarse como

$$x\left(t\right) = M\cos\left(wt\right) + jMsen\left(wt\right)$$

La Figura 2.2 muestra una exponencial compleja con frecuencia angular $w = 1$, la gráfica en rojo corresponde a la parte real y la gráfica en azul corresponde a la parte imaginaria.

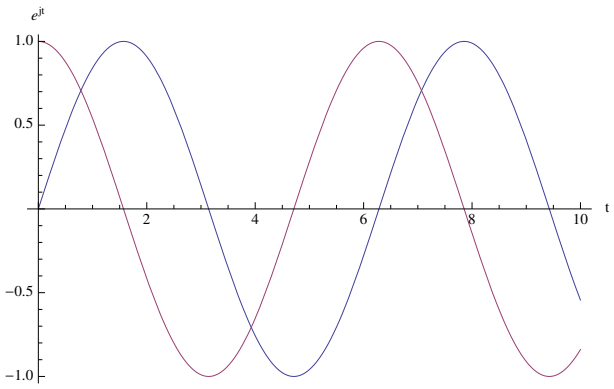


Figura 2.2: Exponencial Compleja

Una propiedad importante de esta señal, es su periodicidad. Una señal es periódica si cada intervalo de tiempo T, tenemos un repetición de la señal. Para verificar esta propiedad hacemos

$$x(t) = x(t+T)$$

$$\begin{aligned} e^{jwt} &= e^{jw(t+T)} \\ e^{jwt} &= e^{jwt}e^{jwT} \end{aligned}$$

para que esta ecuación se cumpla debemos tener que

$$e^{jwT} = 1$$

Existen dos posibilidades para que esta condición se cumpla: la primera cuando $w = 0$, la cual es periódica para cualquier valor de T, pero si $w \neq 0$, entonces tenemos que existe un valor T al cual llamamos periodo fundamental y esta dado por la ecuación

$$T = \frac{2\pi}{w}$$

2.2.3. Función impulso unitario

La función impulso unitario se define como

$$\delta(t) = \begin{cases} 1, & \text{Si } t = 0 \\ 0, & \text{en caso contrario} \end{cases}$$

La Figura 2.3 muestra la función $\delta(t-2)$

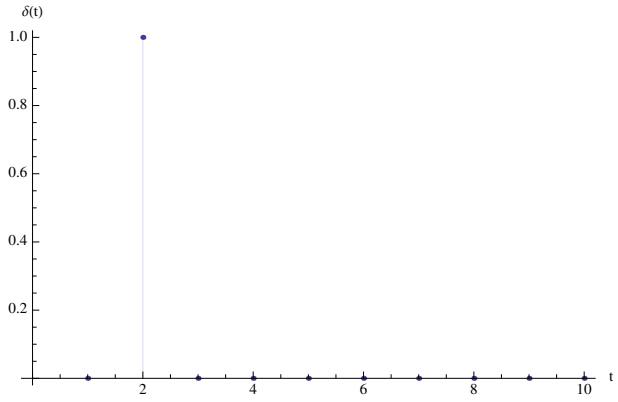


Figura 2.3: Función impulso unitario

2.2.4. Función escalón unitario

Otra señal de interés es la función escalón unitario la cual esta dada por

$$u(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases}$$

La Figura 2.4 muestra la función $u(t-3)$

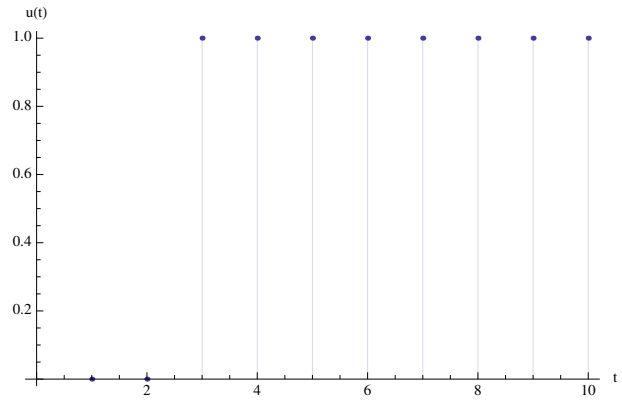


Figura 2.4: Función escalon unitario

La función impulso unitario de tiempo continuo está relacionada con el escalón unitario por la ecuación

$$u(t) = \int_{-\infty}^{t} \delta(\tau)\, d\tau$$

2.3. Señales Periódicas

Una señal es periódica si tiene la propiedad de tener un valor positivo T para el cual

$$x(t) = x(t+T) \quad \forall t$$

en este caso diremos que la señal es periódica con periodo T.

Para señales discretas se deberá cumplir que

$$x[n] = x[n+\hat{T}]$$

donde $\hat{T}$ es un número entero. Es muy importante enfatizar que la señal discreta deberá tener periodos que sean números enteros.

Cualquier señal x que sea igual a la suma de dos señales periódicas, x_1 y x_2, con periodos fundamentales T_1 y T_2 respectivamente, será periódica si se cumple la siguiente relación:

$$\frac{T_1}{T_2} = \frac{m}{n}$$

y el periodo se calcula como $T = nT_1 = mT2$ para n y m enteros, en caso contrario se considera que la señal no es periódica.

2.3.1. Ejemplos

Calcular el periodo de la señal en el caso de que esta es periódica

a) $x(t) = cos(25\pi t)$

b) $x(t) = sen(5t) + cos(6t)$

c) $x(t) = sen(20t) - sen(4t)$

d) $x(t) = sen^2(2\pi t)$

Inciso a)

Dado que tenemos una señal simple

$$\begin{aligned} x(t) &= x(t+T) \\ cos(25\pi t) &= cos(25\pi(t+T)) \\ cos(25\pi t) &= cos(25\pi t)cos(25\pi T) - sen(25\pi t)sen(25\pi T) \end{aligned}$$

por lo tanto $cos(25\pi T) = 1$ y $sin(25\pi T) = 0$: el valor que hace esto es $25\pi T = 2\pi$ por lo tanto $T = \frac{2}{25}$

Otra manera de resolver el problema es haciendo $w = 25\pi$ y de la definición del periodo tenemos que

$$T = \frac{2\pi}{w}$$
$$T = \frac{2\pi}{25\pi} = \frac{2}{25}$$

Inciso b)

Los periodos fundamentales de cada una de las señales es $T_1 = \frac{2\pi}{5}$ y $T_2 = \frac{2\pi}{6}$, de acuerdo con la formula tenemos

$$\frac{T_1}{T_2} = \frac{m}{n} = \frac{\frac{2\pi}{5}}{\frac{2\pi}{6}} = \frac{6}{5}$$

Podemos notar que tenemos la razón de dos números enteros por lo tanto la señal es periódica y tendrá periodo $T = nT_1 = mT_2$ lo cual nos da

$$T = nT_1 = 5\frac{2\pi}{5} = 2\pi$$

Inciso c)

Los periodos fundamentales de cada una de las señales es $T_1 = \frac{2\pi}{20}$ y $T_2 = \frac{2\pi}{4}$, de acuerdo con la formula tenemos

$$\frac{T_1}{T_2} = \frac{m}{n} = \frac{\frac{2\pi}{20}}{\frac{2\pi}{4}} = \frac{4}{20} = \frac{1}{5}$$

Podemos notar que tenemos la razón de dos números enteros por lo tanto la señal es periódica y tendrá periodo $T = nT_1 = mT_2$ lo cual nos da

$$T = nT_1 = 5\frac{2\pi}{20} = \frac{\pi}{2}$$

$$T = mT_2 = 1\frac{2\pi}{4} = \frac{\pi}{2}$$

Incido d)

Aplicado la identidad

$$x(t) = sen^2(2\pi t) = \frac{1}{2} - \frac{cos(4\pi t)}{2}$$

tenemos que la frecuencia fundamental es $w = 4\pi$ por lo tanto el periodo es $T = \frac{2\pi}{4\pi} = 0.5$

2.3.2. Ejemplo

Considere un señal muestreda x_m donde cada elemento se calcula como $x_m[n] = x(nT_m)$ y la señal $x(t)$ esta dada como $x(t) = cos(0.2\pi t)$. Calcular el periodo de la señal muestreada con periodos $T_1 = 1/10$, $T_2 = 2/10$ y $T_3 = 3/10$.

Para la señal x_1 muestreada con periodo T la podemos escribir como

$$x_m[n] = x(T_m n) = cos(0.2\pi T_m n) = cos\left(\frac{2\pi}{T_0} T_m n\right)$$

donde el periodo de muestreo de la señal continua es $T_0 = \frac{2\pi}{w} = \frac{2\pi}{0.2\pi} = 10$ y el periodo de muestreo de la nueva señal es

$$\hat{T}_m = \frac{T_0}{T_m}$$

de acuerdo con esto tendremos diferentes periodos de muestreo para la nueva señal dados como $\hat{T}_1 = 10/(1/10) = 100$, $\hat{T}_2 = 10/(2/10) = 50$ y $\hat{T}_3 = 10/(3/10) = 100/3$ para las señales x_1, x_2 y x_3 respectivamente.

Es importante enfatizar que la señal con muestreo $T_3 = 3/10$ no es una señal discreta periódica.

2.4. Señales pares e impares

Podemos decir que una señal es par si es idéntica a su reflexión alrededor del origen, esto es

$$x(-t) = x(t)$$

y que es impar si

$$x(-t) = -x(t)$$

Una característica importante de cualquier señal, es que esta pude ser representada por la suma de una señal par y una señal impar

$$x\,(t) = \mathcal{P}(x(t)) + \mathcal{I}(x(t))$$

donde la parte par la calculamos

$$\mathcal{P}(x(t)) = \frac{1}{2}\left[x\,(t) + x(-t)\right]$$

y la impar por

$$\mathcal{I}(x(t)) = \frac{1}{2}\left[x\left(t\right) - x(-t)\right]$$

ver [**?**]

2.4.1. Ejemplos.

(a) Demostrar que si $x\left[n\right]$ es discreta e impar entonces

$$\sum_{n=-\infty}^{\infty} x[n] = 0$$

Demostración: si, $x[n]$ es impar, entonces

$$x[n] = -x[-n]$$

es decir

$$x[n] + x[-n] = 0$$

considerando

$$\sum_{n=-\infty}^{\infty} x\left[n\right] = \sum_{n=-\infty}^{-1} x\left[n\right] + x\left[0\right] + \sum_{n=1}^{\infty} x\left[n\right]$$

reordenando

$$\begin{aligned}
\sum_{-\infty}^{\infty} x\left[n\right] &= x\left[0\right] + \sum_{n=-\infty}^{-1} x\left[n\right] + \sum_{n=1}^{\infty} x\left[n\right] \\
&= x\left[0\right] + \sum_{n=1}^{\infty} x\left[-n\right] + \sum_{n=1}^{\infty} x\left[n\right] \\
&= x\left[0\right] - \sum_{n=1}^{\infty}(x\left[n\right] + x\left[-n\right])
\end{aligned}$$

con $x\left[0\right] = 0$ y sustituyendo la definición de señal par en la ecuación anterior tenemos:

$$\begin{aligned}\sum_{-\infty}^{\infty} x[n] &= 0 + \sum_{n=1}^{\infty} 0 \\ \sum_{-\infty}^{\infty} x[n] &= 0\end{aligned}$$

(b) Comprobar, si $x_1[n]$ es impar y $x_2[n]$ es par, entonces

$$x_1[n] \times x_2[n] \Longrightarrow impar$$

Comprobación: Hacemos que

$$z[n] = x_1[n] \times x_2[n]$$

y

$$z[-n] = x_1[-n] \times x_2[-n]$$

Además, sabemos que

$$x_1[n] = -x_1[-n]$$

y

$$x_2[n] = x_2[-n]$$

sustituyendo las ecuaciones tenemos

$$z[n] = -x_1[-n] \times x_2[-n]$$

resulta

$$z[n] = -z[-n]$$

Lo cual implica que $z[n]$ es impar, i.e. $x_1[n] \times x_2[n]$ es impar.

(c) Considere que $x[n]$ es una señal con parte par $\mathcal{P}(x[n])$ y parte impar $\mathcal{I}(x[n])$, demostrar que

$$\sum_{n=-\infty}^{\infty} x^2[n] = \sum_{n=-\infty}^{\infty} P^2(x[n]) + \sum_{n=-\infty}^{\infty} I^2(x[n])$$

Demostración: Hacemos

$$\begin{aligned}\mathcal{P}\left(x\left[n\right]\right) &= x_p\left[n\right] \\ \mathcal{I}\left(x\left[n\right]\right) &= x_i\left[n\right]\end{aligned}$$

y

$$\begin{aligned}\sum_{n=-\infty}^{\infty} x^2\left[n\right] &= \sum_{n=-\infty}^{\infty}\left(x_p\left[n\right]+x_i\left[n\right]\right)^2 \\ &= \sum_{n=-\infty}^{\infty}\left(x_p^2\left[n\right]+2x_p\left[n\right]x_i\left[n\right]+x_i^2\left[n\right]\right)\end{aligned}$$

sustituyendo el término $x_p\left[n\right]x_i\left[n\right]$ por $z\left[n\right]$ y reordenando

$$\sum_{n=-\infty}^{\infty} x^2\left[n\right] = \sum_{n=-\infty}^{\infty}\left(x_p^2\left[n\right]+x_i^2\left[n\right]\right)+2\sum_{n=-\infty}^{\infty} z\left[n\right]$$

Como $z\left[n\right]$ es el producto de una señal par y una impar, usamos el resultado del inciso b) y, concluimos que $z\left[n\right]$ es impar. Ahora, usando el resultado del inciso a) sabemos que

$$2\sum_{n=-\infty}^{\infty} z\left[n\right] = 0$$

Si sustituimos este resultado en la ecuación tenemos que

$$\begin{aligned}\sum_{n=-\infty}^{\infty} x^2\left[n\right] &= \sum_{n=-\infty}^{\infty}\left(x_p^2\left[n\right]+x_i^2\left[n\right]\right) \\ \sum_{n=-\infty}^{\infty} x^2\left[n\right] &= \sum_{n=-\infty}^{\infty} x_p^2\left[n\right]+\sum_{n=-\infty}^{\infty} x_i^2\left[n\right]\end{aligned}$$

Sistemas

3.1. Introducción

Un sistema se puede ver como cualquier proceso que produce una transformación de señales. Un sistema tiene una señal de entrada y una señal de salida la cual estó relacionada con la entrada a través de la transformación del sistema.

$$y(t) = T\left[x(t)\right]$$

donde $x(t)$ es la señal de entrada, $y\left(t\right)$ es la señal de salida y $T\left[\right]$ es la transformación del sistema.

Entre los sistemas podemos tener interconexiones serie y paralelo dadas como los mostrados en la figura 3.1.

Sistemas con y sin memoria.

Si la salida de un sistema para cada valor de la variable independiente depende solo de la entrada en ese mismo instante de tiempo se dice que el sistema no tiene memoria.

$$y(t) = Rx(t)$$

Un ejemplo de un sistema con memoria es

$$y[n] = \sum_{k=-\infty}^{n} x[k]$$

3.1.1. Sistemas inversos.

Decimos que un sistema es invertible si dada una transformación T podemos encontrar la transformación T^{-1} tal que:

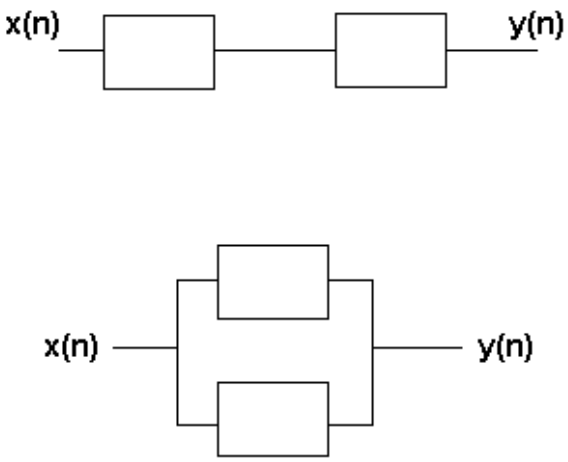


Figura 3.1: Sistemas serie y paralelo

$$\begin{aligned} y(t) &= T[x(t)] \\ x(t) &= T^{-1}[y(t)] \end{aligned}$$

Un ejemplo de un sistema que no es invertible es $y\left(t\right) = 0$.

3.1.2. Sistemas causales.

Un sistema es causal si su salida en cualquier instante de tiempo depende sólo de los valores en el tiempo presente y en el pasado. Estos sistemas también son llamados no anticipativo.

$$\begin{aligned} y(t) &= x(t-1) \\ y(t) &= \int_{-\infty}^{t} x(\tau)d\tau \end{aligned}$$

3.1.3. Estabilidad.

Un sistema es llamado estable si se produce una salida acotada para una entrada acotada. Una señal es acotada si

$$|x(t)| \leq M < \infty$$

y la señal de salida $y(t) = T[x(t)]$, es acotada si

$$|y(t)| \leq N < \infty$$

Ejemplo. Consideremos la secuencia de los números de Fibonacci donde $x = [1, 2, 3, 5, 8, 13, ...]$ donde cada elemento se calcula como $x[k] = x[k-1] + x[k-2]$ sí $k \geq 0$ y 0 cero en caso contrario. En este caso tenemos un sistema no acotado ya que la sucesión crece indefinidamente para valores de $k \in \mathbb{N}$.

Un ejemplo de señal estable lo tenemos en la sucesión

$$x[n] = \sum_{k=1}^{n} \frac{1}{k^2} = 1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \cdots + \frac{1}{(n-1)^2} + \frac{1}{n^2}$$

para esta sucesión podemos ver que

$$x[n] - x[n-1] = \frac{1}{n^2}$$

si tomamos el límite podemos verificar que la sucesión converge en un valor estable

$$\lim_{n\to\infty} [x[n] - x[n-1]] = \lim_{n\to\infty} \frac{1}{n^2} = 0$$

3.2. Invariancia en el tiempo.

Para que un sistema sea invariante en el tiempo se debe cumplir que para un desplazamiento en la señal de entrada se produzca el mismo desplazamiento en la señal de salida.

$$T[D^n[x(t)]] = D^n[T[x(t)]]$$

Ejemplo. Considere la señal $y(t) = sen[x(t)]$

$$\begin{aligned} D^k[x(t)] &= x(t-k) \\ &= sen[x(t-k)] \\ D^k y(t) &= sen[x(t-k)] \end{aligned}$$

3.2.1. Linealidad

La característica principal de los sistemas lineales es

$$T\left[\alpha x(t) + \beta y(t)\right] = T\left[\alpha x(t)\right] + T\left[\beta y(t)\right]$$

esta propiedad es conocida como el principio de superposición.

Ejemplo. Considere el sistema $y(t) = mx(t) + b$. Que valores debe tener el sistema para ser lineal.

Consideremos dos señales $x_1(t)$ y $x_2(t)$

$$\begin{aligned} y_1(t) &= mx_1(t) + b \\ y_2(t) &= mx_2(t) + b \\ y_1(t) + y_2(t) &= m\left(x_1(t) + x_2(t)\right) + 2b \end{aligned}$$

Si aplicamos la transformación a la suma de $x_1(t)$ y $x_2(t)$ tendremos

$$y_3(t) = m\left(x_1(t) + x_2(t)\right) + b$$

note que $y_3(t)$ es diferente de $y_1(t) + y_2(t)$, la única posibilidad es que la constante b sea igual a cero.

3.3. Ejemplos

3.3.1. Ejemplo

Demuestre que $y[n] = x[n] - x[n-1]$ es invariante en el tiempo.

Solución. Sea, $v[n] = D^{n_0}\left\{T[x[n]]\right\}$ y $\hat{v}(n) = T\left\{D^{n_0}[x(n)]\right\}$. Un sistema es invariante si $v(n) = \hat{v}(n)$

Para nuestro ejemplo $v[n] = D^{n_0}[x[n] - x[n-1]] = x[n-n_0] - x[n-n_0-1]$ por otro lado $\hat{v}(n) = T[x[n-n_0]] = x[n-n_0] - x[n-n_0-1]$, como $v[n] = \hat{v}[n]$ el sistema es invariante en el tiempo.

Nota: En este ejemplo, hacemos una aproximación de la derivada utilizando diferencias finitas

$$y(t) = \frac{dx}{dt} = \lim_{h \leftarrow 0} \frac{x(t) - x(t-h)}{h}$$

En forma discreta podemos expresarla considerando $h = 1$ como:

$$y[n] \approx \frac{x(n) - x(n-1)}{1} = x(n) - x(n-1)$$

3.3.2. Ejemplo

Demuestre que $y[n] = \sum_{k=n-n_0}^{n} x[k]$ es invariante en el tiempo.

Solución. En éste caso, $v[n] = D^{n_0}\left[\sum_{k=k_0}^{k_f} x[k]\right] = \sum_{k=k_0}^{k_f} x[k-n_0]$ y $\hat{v}[k] = T[D^{n_0}[x[k]]] = \sum_{k=k_0}^{k_f} x[k-n_0]$. Como $v[n] = \hat{v}[k]$ el sistema es invariante en el tiempo.

3.3.3. Ejemplo

Dada la sucesión

$$y[n] = \frac{1}{4}y[n-1] + x[n] \tag{3.6}$$

1. Probar que es un sistema Lineal invariante en el Tiempo (LIT) y
2. Determinar la salida $y[n]$ si $x[n] = \delta(n-1)$

Comenzaremos por dar una solución de la recurrencia, haciendo sustituciones sucesivas

$$\begin{aligned} y[n] &= x[n] + \frac{1}{4}y[n-1] \\ &= x[n] + \frac{x[n-1]}{4} + \frac{1}{4}y[n-2] \\ &= x[n] + \frac{x[n-1]}{4} + \frac{x[n-2]}{4^2} + \frac{1}{4^2}y[n-3] \\ &= x[n] + \frac{x[n-1]}{4} + \frac{x[n-2]}{4^2} + \frac{x[n-3]}{4^3} + \cdots + \frac{x[1]}{4^{n-1}} + \frac{x[0]}{4^n} \end{aligned}$$

donde de manera general la solución de la recursión es

$$y(n) = \sum_{k=0}^{n} \frac{x[n-k]}{4^k} \tag{3.7}$$

Linealidad

Por otra parte tenemos que verificar que el sistema dado por 3.6 es Lineal, primero calculamos utilizando 3.7

$$T[\alpha x_1[n]] + T[\beta x_2[n]] = \alpha \sum_{k=0}^{n} \frac{x_1[n-k]}{4^k} + \beta \sum_{k=0}^{n} \frac{x_2[n-k]}{4^k}$$

y segundo utilizando 3.7 calculamos

$$\begin{aligned} T[\alpha x_1[n] + \beta x_2[n]] &= \sum_{k=0}^{n} \frac{\alpha x_1[n-k] + \beta x_2[n-k]}{4^k} \\ &= \alpha \sum_{k=0}^{n} \frac{x_1[n-k]}{4^k} + \beta \sum_{k=0}^{n} \frac{x_2[n-k]}{4^k} \end{aligned}$$

lo cual demuestra que el sistema es lineal.

Invariancia en el tiempo

En este caso tenemos que mostrar $D^{n_0}\left[T\left[x[n]\right]\right] = T\left[D^{n_0}\left[x[n]\right]\right]$, aplicando la solución de la recursión dada por 3.7, tenemos

$$D^{n_0}\left[T\left[x[n]\right]\right] = D^{n_0}\left[\sum_{k=0}^{n} \frac{x[n-k]}{4^k}\right] = \sum_{k=0}^{n} \frac{x[n-n_0-k]}{4^k}$$

Si aplicamos la traslación a la señal x tenemos

$$D^{n_0}\left[x[n]\right] = x[n-n_0], x[n-1-n_0], x[n-2-n_0], \cdots, x[-n_0],$$

y posteriormente calculamos

$$T\left[D^{n_0}\left[x[n]\right]\right] = \sum_{k=0}^{n} \frac{x[n-n_0-k]}{4^k}$$

De lo anterior podemos concluir que el sistema es invariante en el tiempo.

Respuesta al impulso

Para determinar la salida $y[n] = \frac{1}{4}y[n-1] + x[n]$ y suponiendo que $y[-1] = 0$, se obtiene la siguiente tabla al evaluar recursivamente:

n	$x[n]$	$y[n]$
0	1	1
1	0	$\frac{1}{4}$
2	0	$\frac{1}{4^2}$
3	0	$\frac{1}{4^3}$

Se observa que para cualquier n la salida ceró $y[n] = \left(\frac{1}{4}\right)^n$, note que esta solución puede verificarse utilizando la solución de la recursión.

3.3.4. Ejemplo

Considere un sistema $y = Ax$ donde x y y son vectores y A es una matriz, es un sistema lineal e invariante a translación.

$$\begin{bmatrix} y[0] \\ y[1] \\ \vdots \\ y[N-1] \end{bmatrix} = \begin{bmatrix} A[0,0] & A[0,1] & \cdots A[0,N-1] \\ A[1,0] & A[1,1] & \cdots A[1,N-1] \\ \vdots & \vdots & \cdots \vdots \\ A[N-1,0] & A[N-1,1] & \cdots A[N-1,N-1] \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

Linealidad

Consideremos que x esta dado por la suma de dos vectores x_1 y x_2

$$\hat{y} = A(x_1 + x_2)$$

$$\hat{y} = Ax_1 + Ax_2 = y_1 + y_2$$

El sistema es lineal.

Invariancia a translación

Para el sistema dado podemos calcular un elemento de la señal y haciendo

$$y[n] = A[n,0]x[0] + A[n,1]x[1] + A[n,2]x[2] + \cdots + A[n,N-1]x[N-1] = y[n] = A_n x$$

Si aplicamos el operador de traslación $D^{n_0}[x]$ a la señal x tenemos

$$D^{n_0}[x] = x[0 - n_0] + x[1 - n_0] + \cdots + x[N - 1 - n_0]$$

y multiplicamos por la matriz A

$$y'[n] = A[n,0]x[0 - n_0] + A[n,1]x[1 - n_0] + \cdots + A[n,N-1]x[N - 1 - n_0] = A_n D^{n_0}[x]$$

Si aplicamos la transformación a la señal y tenemos $D^{n_0}[y]$

$$y[n-n_0] = A[n-n_0,0]x[0-n_0] + A[n-n_0,1]x[1-n_0] + \cdots + A[n-n_0,N-1]x[N-1-n_0]$$

$$y[n-n_0] = A_{n-n_0}D^{n_0}[x]$$

Para que el sistema sea invariante a traslación tenemos que $y[n-n0] = y'[n]$ lo que significa que

$$y'[n] = y[n-n_0]$$
$$A_n D^{n_0}[x] = A_{n-n_0}D^{n_0}[x]$$

El sistema será invariante a translación si y solo si los renglones n y $n-n_0$ de la matriz A son iguales $A_n = A_{n-n_0}$.

Convolución

4.1. Correlación

Consideremos dos señales x y y, dados como $x = [x[0], x[1], x[2], \cdots, x[N-1]]$ $y = [y[0], y[1], \cdots, x[0], \cdots, x[N-1], \cdots, y[M-1]]]$, donde x es un subconjunto de los valores de la señal y .

Comenzaremos por utilizar como medida de similitud el producto escalar de vectores, así el ángulo entre estas dos señales lo podemos calcular como

$$cos(\theta) = \frac{a^T b}{|a||b|} = \frac{\sum_{k=0}^{N-1} a[k]b[k]}{\sqrt{\sum_{k=0}^{N-1}(a[k])^2}\sqrt{\sum_{k=0}^{N-1}(b[k])^2}}$$

donde θ es el ángulo entre los dos vectores. Note que cuando los dos vectores son iguales tendremos que el ángulo es cero, el valor máximo será 1 y en caso de que sean totalmente diferentes será -1.

Si consideramos

$$r = |a||b|cos(\theta) = ab^T = \sum_{k=0}^{N-1} a[k]b[k]$$

el valor de r será máximo cuando el vector b sea igual o múltiplo del vector $b = \alpha a$.

Para nuestras señales x y y tomamos un subconjunto de la señal y de tamaño N igual a $\hat{y}_n = [y[n], y[n+1], y[n+2], \cdots, y[n+N-1]] = y[n : n+N-1]$ y tratamos de averiguar la posición n de la señal y que empata mejor con la señal x. Para ello buscamos cual producto escalar $x^T \hat{y}_n$ es máximo considerando diferentes valores $n \in [0, 1, 2, \cdots]$.

El producto de $x^T \hat{y}_n$ lo podemos calcular como

$$r_{x\hat{y}_n} = \sum_{k=0}^{N} x[k]\hat{y}_n[k]$$

dado que $\hat{y}$ es un subconjunto de y, podemos calcular el producto $x^T \hat{y}_n$ en las n posiciones posibles de la serie con la siguiente expresión

$$r_{xy}[n] = \sum_{k=0}^{N} x[k]y[k+n]$$

La cross–correlación de las señales discretas x y y es la secuencia $r_{xy}(n)$ definida por

$$r_{xy}[n] = \sum_{k=-\infty}^{\infty} x[k]\, y[k+n] \tag{4.8}$$

y en el caso de que las señales sean continuas como

$$r_{xy}(t) = \int_{-\infty}^{\infty} x(\tau)\, y(\tau + t)\, d\tau$$

La función de correlación suministra una medida de la similitud o inter–dependencia entre las funciones x y y.

4.1.1. Propiedades

Conmutación

Demostrar que la función de correlación y auto–correlación no son conmutativas, es decir

$$r_{xy}(t) = r_{yx}(-t)$$
$$r_{xx}(t) = r_{xx}(-t)$$

para ello hacemos

$$r_{xy}(t) = \int_{-\infty}^{\infty} x(\tau)\, y(\tau + t)\, d\tau$$

y sustituimos $\widehat{\tau} = \tau + t$

$$r_{yx}(-t) = \int_{-\infty}^{\infty} x(\widehat{\tau} - t)\, y(\widehat{\tau})\, d\widehat{\tau}$$

en el caso de la auto–correlación tenemos

$$r_{xx}(t) = \int_{-\infty}^{\infty} x(\tau)\, x(\tau + t)\, d\tau$$

y sustituimos $\widehat{\tau} = \tau + t$

$$r_{xx}(-t) = \int_{-\infty}^{\infty} x\left(\widehat{\tau} - t\right) x\left(\widehat{\tau}\right) d\widehat{\tau}$$

Lo cual significa que la correlación no es conmutativa.

Invariancia a traslación

Es la correlación invariantes a traslación.

Comenzamos por calcular $x(t-a) = D^a[x(t)]$

$$r_{xy}(t) = \int_{-\infty}^{\infty} x\left(\tau - a\right) y\left(\tau + t\right) d\tau$$

si sustituimos

$$\widehat{\tau} = \tau - a$$
$$\tau = \widehat{\tau} + a$$
$$\int_{-\infty}^{\infty} x\left(\widehat{\tau}\right) y\left(\widehat{\tau} + (t+a)\right) d\tau = r_{xy}\left(t + a\right)$$

Si aplicamos $r_{xy}(t-a) = D^a[r_{xy}(t)]$ dado lo anterior la correlación no es invariante a traslación.

Linealidad

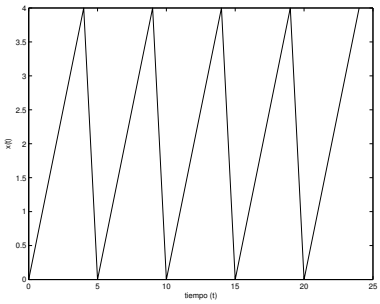
Pero, Será lineal la correlación?

$$r_{xy}\left(t\right) = \int_{-\infty}^{\infty} \left[x_1\left(\tau\right) + x_2\left(\tau\right)\right] y\left(\tau + t\right) d\tau$$
$$= \int_{-\infty}^{\infty} x_1\left(\tau\right) y\left(\tau + t\right) d\tau + \int_{-\infty}^{\infty} x_2\left(\tau\right) y\left(\tau + t\right) d\tau$$
$$= r_{x_1 y} + r_{x_2 y}$$

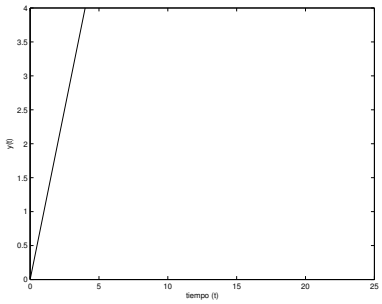
si es una transformación lineal la correlación.

4.1.2. Ejemplo

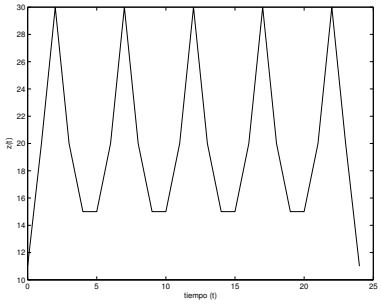
Suponga una señal $x = [0, 1, 2, 3, 4, 0, 1, 2, 3, 4, \ldots]$ (Fig. 5.5(a)) y una señal $y = [0, 1, 2, 3, 4]$ (Fig. 4.1(b)). La correlación entre ambas señales, se muestra en la Fig. 5.5(b). Note que los máximos se obtienen cuando la señal y empata perfectamente con x.



(a) $x(t)$



(b) $y(t)$



(c) $z(t)$ ***correlación entre*** $x(t)$ ***y*** $y(t)$

Figura 4.1: Correlación

4.1.3. Correlación en dos dimensiones

Si tenemos dos señales discretas en dos dimensiones, como es el caso de imágenes, la correlación se representa como

$$r_{xy}[n,m] = \sum_{k=-n_k}^{n_k} \sum_{l=-n_l}^{n_l} x[k,l] * y[k+n, l+m]$$

4.2. Representación de señales en términos de impulsos.

La función impulso unitario, puede utilizarse para construir una clase amplia de señales. Para ilustrar como funciona consideremos que tenemos una señal discreta dada por x. Si queremos ver un parte de la señal en cierto instante de tiempo multiplicamos la señal de

entrada por un impulso en el instante que deseamos analizar, así por ejemplo:

$$x[-1]\,\delta(n+1) = \begin{cases} x[-1] & \text{si n=-1} \\ 0 & \text{si no} \end{cases}$$

$$x[0]\,\delta(n+0) = \begin{cases} x[0] & \text{si n=0} \\ 0 & \text{si no} \end{cases}$$

$$x[1]\,\delta(n-1) = \begin{cases} x[1] & \text{si n=1} \\ 0 & \text{si no} \end{cases}$$

por lo tanto la suma suma de estos términos me da la señal x. Matemáticamente podemos expresar esta suma como

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\,\delta(n-k)$$

esta ecuación es llamada la propiedad de escudriñamiento del impulso unitario.

4.3. Convolución

Podemos extender el concepto sustituyendo la función impulso por cualquier otra función, así obtenemos la expresión de la convolución para dos señales discreta como.

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\,h[n-k]$$

note que también es un sistema LIT. La convolución la podemos representar por $y = x * h$

4.3.1. Propiedades

Conmutación

La convolución es conmutativa es decir

$$f_1 * f_2 = f_2 * f_1$$

demostración, para probar la propiedad conmutativa hacemos

$$f_1 * f_2 = \sum_{k=-\infty}^{\infty} f_1[k]\,f_2[n-k]$$

haciendo el cambio de variable $\widehat{k} = n - k$ tenemos

$$\sum_{\hat{k}=-\infty}^{\infty} f_1\left[n-\widehat{k}\right] f_2\left[\widehat{k}\right] = \sum_{\hat{k}=-\infty}^{\infty} f_2\left[\widehat{k}\right] f_1\left[n-\widehat{k}\right] = f_2 * f_1$$

Asociativa

La propiedad asociativa consiste en

$$(f_1 * f_2) * f_3 = f_1 * (f_2 * f_3)$$

para demostrar la propiedad asociativa hacemos $f_1 * f_2 = g$ y $f_2 * f_3 = h$

$$g * f_3 = f_1 * h$$

puesto que

$$g[n] = \sum_{k=-\infty}^{\infty} f_1[k] f_2[n-k]$$

se tiene que

$$g[n] * f_3(n) = \sum_{j=-\infty}^{\infty} g[j] f_3[n-j]$$

$$= \sum_{j=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} f_1[k] f_2[j-k] \right) f_3[n-j]$$

sustituyendo $l = j - k$ y cambiando el orden de las sumatorias

$$= \sum_{k=-\infty}^{\infty} f_1[k] \sum_{l=-\infty}^{\infty} f_2[l] f_3[n-(l+k)]$$

$$= \sum_{k=-\infty}^{\infty} f_1[k] \sum_{l=-\infty}^{\infty} f_2[l] f_3[(n-k)-l]$$

$$= \sum_{k=-\infty}^{\infty} f_1[k] h[n-k]$$

$$= f_1 * h$$

Invariancia a traslación

Para probar que la convolución es un sistema invariante a traslación hacemos la convolución de la señal discreta x trasladada un valor k_0

$$y[n] = \sum_{k=-\infty}^{\infty} x\left[k - k_0\right] h\left(n - k\right)$$

haciendo $\widehat{k} = k - k_0$ tenemos:

$$\begin{aligned} &\sum_{\widehat{k}=-\infty}^{\infty} x\left(\widehat{k}\right) h\left[n - \left(\widehat{k} + k_0\right)\right] \\ &\sum_{\widehat{k}=-\infty}^{\infty} x\left[\widehat{k}\right] h\left[\left(n - k_0\right) - \widehat{k}\right] \\ &= y[n - k_0] \end{aligned}$$

Linealidad

Para probar que se trata de un sistema lineal, calculamos la convolución de la suma de dos señales x_1 y x_2

$$\begin{aligned} y[n] &= \sum_{k=-\infty}^{\infty} \left(\alpha x_1\left[k\right] + \beta x_2\left[k\right]\right) h\left[n - k\right] \\ &= \alpha \sum_{k=-\infty}^{\infty} x_1\left[k\right] h\left[n - k\right] + \beta \sum_{k=-\infty}^{\infty} x_2\left[k\right] h\left[n - k\right] \\ &= y_1[n] + y_2[n] \end{aligned}$$

4.3.2. Sucesión útil

Una sucesión que nos será especialmente útil para realizar los cálculos de convolución y correlación es la sucesión geométrica

$$s_N = \sum_{n=0}^{N} a^n = 1 + a + a^2 + ... + a^N$$

para esta serie podemos ver

$$\begin{aligned} s_N &= 1 + a(1 + a + a^2 + ... + a^{N-1}) \\ &= 1 + as_{N-1} \end{aligned}$$

que pasa si multiplicamos s_N por $(1 - a)$

$$\begin{aligned} (1 - a)\, s_N &= \left(1 + a + a^2 + ... + a^N\right) - \\ &= \left(a + a^2 + a^3 ... + a^{N+1}\right) \\ &= 1 - a^{N+1} \end{aligned}$$

de lo cual concluimos que

$$s_N = \frac{1 - a^{N+1}}{1 - a}$$

Esta serie será convergente en el caso de que $|a| < 1$ y divergente en el caso de que $|a| > 1$. Lo cual lo podemos verificar haciendo

$$\lim_{N \to \infty} a^N = 0$$

y el valor de convergencia lo podemos calcular con

$$s_N = 1 + as_{N-1}$$

en el lómite $s_N = s_{N-1} = r$

$$\begin{aligned} r &= 1 + ar \\ r(1 - a) &= 1 \\ r &= \frac{1}{1 - a} \end{aligned}$$

En el caso de que $|r| = 1$, la serie converge al valor de

$$\begin{aligned} s_N = \sum_{n=0}^{N} 1^n &= 1 + 1 + 1^2 + ... + 1^N \\ &= N + 1 \end{aligned}$$

y en el caso de $|r| > 1$ la serie diverge

$$\lim_{N \to \infty} a^N \neq 0$$

4.3.3. Ejemplos

Ejemplo 1

Considere una señal discreta x cuyos elementos se calculan como $x[n] = \alpha^n$ y un kernel dado por $h\,[n] = \beta^n$. Calcular la convolución de estas dos señales.

$$y\,[n] = \sum_{k=0}^{N-1} \alpha^k \beta^{(n-k)}$$
$$y\,[n] = \sum_{k=0}^{N-1} \alpha^k \beta^n \beta^{-k}$$
$$y\,[n] = \beta^n \sum_{k=0}^{N-1} \left(\frac{\alpha}{\beta}\right)^k$$
$$y\,[n] = \beta^n \frac{1 - \left(\frac{\alpha}{\beta}\right)^N}{1 - \frac{\alpha}{\beta}}$$

si quisieramos calcular la correlación hacemos

$$y\,[n] = \sum_{k=0}^{N-1} \alpha^k \beta^{(n+k)}$$
$$y\,[n] = \sum_{k=0}^{N-1} \alpha^k \beta^n \beta^k$$
$$y\,[n] = \beta^n \sum_{k=0}^{N-1} (\alpha\beta)^k$$
$$y\,[n] = \beta^n \frac{1 - (\alpha\beta)^N}{1 - (\alpha\beta)}$$

Ejemplo 2

Mostrar que la convolución de una señal impulso unitario con un kernel cualquiera es el mismo kernel

$$\delta[n] * h[n] = h[n]$$

prueba

Por definición tenemos

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

para nuestro caso sustituimos $x[n]$ por la función impulso $\delta[n]$ dando lugar a

$$y[n] = \sum_{k=-\infty}^{\infty} \delta[k]h[n-k]$$

recordemos que la función impulso, será igual a 1 solo cuando su argumento es cero, por lo cual

$$y[n] = \cdots + 0 \times h[n-1] + 1 \times h[n] + 0 \times h[n+1] + \cdots$$
$$y[n] = h[n]$$

Por esta propiedad al kernel se le conoce también como respuesta al impulso.

Ejemplo 3

Mostrar que dado $y[n] = x[n] * h[n]$ podemos hacer $y'[n] = x[n] * h'[n]$ donde h' es la derivada del kernel h

Por definición tenemos

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] * h[n-k]$$

y que la derivada de una función discretizada puede ser representada por $y'[n] = y[n] - y[n-1]$ tenemos

$$y[n] - y[n-1] = \sum_{k=-\infty}^{\infty} x[k] * h[n-k] - \sum_{k=-\infty}^{\infty} x[k] * h[(n-1)-k]$$
$$y[n] - y[n-1] = \sum_{k=-\infty}^{\infty} x[k] * [h[n-k] - h[(n-1)-k]]$$
$$y'[n] = \sum_{k=-\infty}^{\infty} x[k] * h'[n-k]$$

4.4. Convolución en dos dimensiones

La convolución en dos dimensiones de una señal x con un kernel h se define como

$$y[r,c] = \sum_{k=-n_k}^{n_k} \sum_{l=-n_l}^{n_l} h[k,l]x[r-k, c-l]$$

El algoritmo 1 muestra el pseudo código para hacer la implementación de la convolución en dos dimensiones. Para este algoritmo tenemos que el número de operaciones que se debe realizar es $N_r \times N_c \times (2N_k+1) \times (2N_l+1)$. En el caso extremos de que $N_r = N_c = (2n_k+1) = (2n_l+1) = N$ tendremos en total N^4 lo cual significa que tenemos un algoritmo de complejidad $O(N^4)$

Resultado: y
para $r \leftarrow 0$ **a** N_r **hacer**
  para $c \leftarrow 0$ **a** N_c **hacer**
    $suma \leftarrow 0$;
    para $k \leftarrow -N_k$ **a** N_k **hacer**
      para $l \leftarrow -N_l$ **a** N_l **hacer**
        si $r-k \geq 0$ y $c-l \geq 0$ y $r-k < N_r$ y $c-l < N_c$ **entonces**
          $suma \leftarrow suma + h[k,l]x[r-k, c-l]$;
        fin
      fin
      $y[r,c] = suma$
    fin
  fin
fin

Algoritmo 1: Convolución en dos dimensiones

4.5. Kerneles Separables

Si partimos de la definición de la convolución

$$y[r,c] = \sum_{k=-n_k}^{n_k} \sum_{l=-n_l}^{n_l} h[k,l]x[r-k, c-l]$$

y suponemos que el kernel $h[k,l]$ puede representarse por el producto de dos kerneles unidimensionares $h[k,l] = h_1[k] \times h_2[l]$, podemos escribir la convolución como:

$$y[r,c] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x[k,l]h[r-k,c-l]$$

$$y[r,c] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x[k,l](h_1[r-k] \times h_2[c-l])$$

Reorganizando términos tenemos

$$y[r,c] = \sum_{k=-\infty}^{\infty} h_1[r-k] \sum_{l=-\infty}^{\infty} x[k,l]h_2[c-l]$$

Podemos hacer la convolución por renglones mediante

$$\hat{y}[r,c] = \sum_{l=-\infty}^{\infty} x[k,l]h_2[c-l] \equiv \sum_{l=-\infty}^{\infty} x[k,c-l]h_2[l]$$

y después por columnas

$$y[r,c] = \sum_{k=-\infty}^{\infty} \hat{y}[k,l]h_1[r-k] \equiv \sum_{k=-\infty}^{\infty} \hat{y}[k-r,l]h_1[k]$$

El algoritmo 2 muestra el pseudo código para hacer la implementación de la convolución en dos dimensiones utilizando kerneles separables. Para este algoritmo tenemos que el número de operaciones que se debe realizar es $N_r \times N_c \times (2n_l+1) + N_r \times N_c \times (2n_k+1)$. En el caso extremos de que $N_r = N_c = (2n_k+1) = (2n_l+1) = N$ tendremos en total $2N^3$ lo cual significa que tenemos un algoritmo de complejidad $O(N^3)$. Este algoritmo 2 es un orden

de magnitud más rápido que el algoritmo 1.

Resultado: y
para $r \leftarrow 0$ **a** N_r **hacer**
  para $c \leftarrow 0$ **a** N_c **hacer**
    $suma \leftarrow 0$;
    para $l \leftarrow -N_l$ **a** N_l **hacer**
      si $c - l \geq 0$ *y* $c - l < N_c$ **entonces**
        $suma \leftarrow suma + h_2[l]x[r, c - l]$;
      fin
      $\hat{x}[r, c] = suma$
    fin
  fin
fin
para $r \leftarrow 0$ **a** N_r **hacer**
  para $c \leftarrow 0$ **a** N_c **hacer**
    $suma \leftarrow 0$;
    para $k \leftarrow -N_k$ **a** N_k **hacer**
      si $r - k \geq 0$ *y* $r - k < N_r$ **entonces**
        $suma \leftarrow suma + h_1[k]\hat{x}[r - k, c]$;
      fin
      $y[r, c] = suma$
    fin
  fin
fin

Algoritmo 2: Convolución en dos dimensiones Desacoplada

4.6. Algunos kerneles interesantes

4.6.1. Suavizadores

Caja

Para implementar este suavizador, en una dimensión, utilizamos una señal dada como $h = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]$

En dos dimensiones, podemos implementar este kernel de manera separada si hacemos la siguiente operación

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

En la figura 4.2(a) y 4.2(d), se presentan gráficamente, los kerneles para suavizadores de caja en una y dos dimensiones

Binomial

Este kernel esta basado en los coeficientes binomiales. Una manera fácil de calcularlo es utilizar un triángulo de Pascal, de la siguiente forma.

1,1 1,2,1 1,3,3,1 1,4,6,4,1 1,5,10,10,5,1 1,6,15,20,15,6,1

Dado $b_1 = [1, 1]$ podemos calcular

$$\begin{aligned} b_2 &= b_1 * b_1 \\ b_3 &= b_2 * b_1 \\ b_4 &= b_3 * b_1 \\ &\vdots \\ b_{k+1} &= b_k * b_1 \end{aligned}$$

Para su implementación este kernel es normalizado para que su suma de 1. En dos dimensiones de manera separable se puede implementar como

$$B_3 = b_3 * b_3$$

Note que este kernel es recursivo.

$$\frac{1}{4}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \frac{1}{4}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} = \frac{1}{16}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Gaussiano

Este kernel es creado utilizando una campana de Gauss. Par calcularla utilizamos la expresión

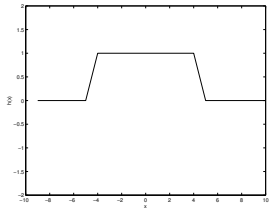
$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

donde: μ es la media de la distribución y σ es la varianza. En nuestro caso la media la consideraremos cero y la varianza la utilizaremos como una estimación del tamaóo del kernel.

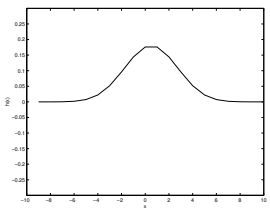
En dos dimensiones calcularemos nuestro kernel de manera desacoplada haciendo $G(x, y) = g(x) * g(y)$ o mediante la función

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp^{\left(-\frac{(x-\mu_x)^2}{2\sigma^2} - \frac{(y-\mu_y)^2}{2\sigma^2}\right)}$$

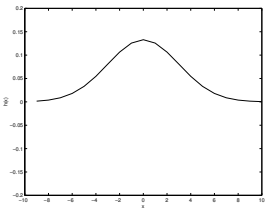
En la figura 4.2(c) se muestra el kernel Gaussiano en una dimensión y 4.2(f) en dos dimensiones.



(a) ***Caja en 1d***

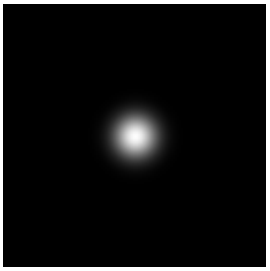


(b) ***Binomial 1d***



(c) ***Gaussiano 1d***

(d) ***Caja 2d***



(e) ***Binomial 2d***

(f) ***Gaussiano 2d***

Figura 4.2: Kerneles para suavizado de señales en una y dos dimensiones

4.6.2. Derivadas

Derivada en x y y

La derivada de una función esta definida como

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x) - f(x-h)}{h}$$

considerando un incremento unitario, que para el caso discreto es el más pequeño y que la señal fue capturada con una frecuencia de muestreo adecuada tenemos que la derivada la podemos aproximar por

$$\frac{df(x)}{dx} \approx f(x) - f(x-h)$$

De la expresión anterior podemos ver que el kernel de derivadas seró $d(n) = [1 - 1]$, asó la derivada la calcularemos como $y(n) * d(n)$.

Para calcular las derivadas de orden N, solamente debemos recordar que se debe aplicar sucesivamente la derivada de orden uno, con ello damos lugar a la siguiente familia de kerneles.

$$+1,\text{-}1 \quad +1,\text{-}2,+1 \quad +1,\text{-}3,+3,\text{-}1 \quad +1,\text{-}4,+6,\text{-}4,\text{-}1$$

Derivadas de Gaussianas

Un problema que presenta el cólculo de las derivadas utilizando convolución es que son amplificadores de ruido. Esto quiere decir que si la señal presenta ruido, la señal resultante de la convolución, el ruido será más notorio. Una manera de eliminar el ruido de una señal es aplicar un suavizador, el mós apropiado es el suavizado Gaussiano [**?**]. Así pues, para eliminar el ruido, es deseable primero convolucionamos con un kernel gaussiano $h[n] = g(n)$ y luego aplicamos un kernel de derivadas de la siguiente forma

$$y[n] = (x[n] * h[n]) * d[n] = x(n) * (d[n] * g(n)) = x[n] * \left.\frac{dg(x)}{dx}\right|_{x=n}$$

Resulta que tanto $h[n]$ y $d[n]$ son aproximaciones de una Gaussiana y una derivada respectivamente, y que ambos dan lugar a un nuevo kernel, pero es más inteligente calcular este kernel como la derivada de una función gaussiana. El kernel de derivada Gaussiano es

$$\frac{dg(x)}{dx} = \frac{-x}{\sigma^2}\left[\frac{1}{\sqrt{2\pi}\sigma}\exp^{\frac{-x^2}{2\sigma^2}}\right]$$

El kernel de segunda derivada es

$$\frac{d^2g(x)}{dx} = \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right]\left[\frac{1}{\sqrt{2\pi}\sigma}\exp^{\frac{-x^2}{2\sigma^2}}\right]$$

(a) ***Primer derivada 1d*** **(b)** ***Segunda derivada 1d***

Figura 4.3: Kerneles de derivadas de gaussianas con $\sigma = 10$

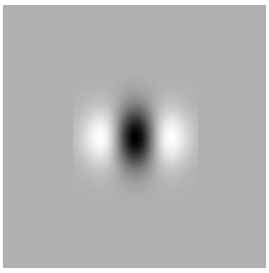
En dos dimensiones, la implementación de estos kerneles lo haremos de manera separable de la siguiente manera. Así la derivada en la dirección x será

$$\begin{aligned} g(x,y) &= g(x) * g(y) \\ \frac{\partial g(x,y)}{\partial x} &= \frac{\partial g(x)}{\partial x} * g(y) \\ \frac{\partial g(x,y)}{\partial y} &= g(x) * \frac{\partial g(y)}{\partial y} \end{aligned}$$

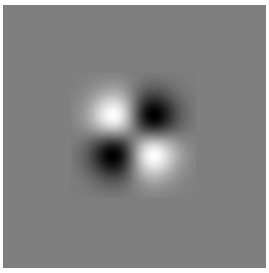
Las segundas derivadas se calcularán como:

$$\begin{aligned} \frac{\partial^2 g(x,y)}{\partial x^2} &= \frac{\partial^2 g(x)}{\partial x^2} * g(y) \\ \frac{\partial^2 g(x,y)}{\partial x \partial y} &= \frac{\partial g(x)}{\partial x} * \frac{\partial g(y)}{\partial y} \\ \frac{\partial^2 g(x,y)}{\partial y^2} &= g(x) * \frac{\partial^2 g(y)}{\partial y^2} \end{aligned}$$

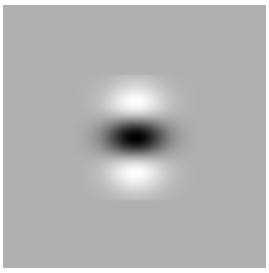
En la figuras 4.4(a), 4.4(b) y 4.4(c), se presentan los kerneles de derivadas de Gaussianas en dos dimensiones.



(a) *Kernel* $\frac{\partial^2 g(x,y)}{\partial x^2}$ *en 2d*



(b) *Kernel* $\frac{\partial^2 g(x,y)}{\partial x \partial y}$ *en 2d*



(c) *Kernel* $\frac{\partial^2 g(x,y)}{\partial y^2}$ *en 2d*

Figura 4.4: Kerneles de derivadas de gaussianas con $\sigma = 10$ en dos dimensiones

Polinomios de Hermite

Considerando una función Gaussiana $g(x)$ las derivadas de esta función dan origen a un conjunto de polinmio denominados de Hermite. La ecuación (4.9) muestra la función Gaussiana y sus primeras seis derivadas. Si factorizamos la gaussiana en cada una de sus derivadas estos polinomios resultantes son los denominados polinomios de Hermite.

$$
\begin{aligned}
g(x) \\
\frac{dg(x)}{dx} &= g(x)\frac{\mu - x}{\sigma^2} \\
\frac{d^2 g(x)}{dx^2} &= g(x)\frac{(x-\mu)^2 - \sigma^2}{\sigma^4} \\
\frac{d^3 g(x)}{dx^3} &= g(x)\frac{3\sigma^2(x-\mu) - (x-\mu)^3}{\sigma^6} \\
\frac{d^4 g(x)}{dx^4} &= g(x)\frac{3\sigma^4 - 6\sigma^2(x-\mu)^2 + (x-\mu)^4}{\sigma^8} \\
\frac{d^5 g(x)}{dx^5} &= g(x)\frac{10\sigma^2(x-\mu)^3 + 15\sigma^4(\mu - x) - (x-\mu)^5}{\sigma^{10}} \\
\frac{d^6 g(x)}{dx^6} &= g(x)\frac{-15\sigma^6 - 15\sigma^2(x-\mu)^4 + 45\sigma^4(x-\mu)^2 + (x-\mu)^6}{\sigma^{12}}
\end{aligned}
\tag{4.9}
$$

Laplaciano

El Laplaciano resulta de sumar las segundas derivadas en x y y de una señal bidimensional y esta dado por la siguiente expresión

$$\Delta = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2}$$

Este kernel lo podemos construir sumando los kerneles de segundas derivadas de la siguiente forma

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

De manera robusta lo podemos calcular utilizando derivadas de Gaussiana, lo cual da lugar al kernel denominado "Mexican Hat". Este kernel se utiliza para determinar bordes en una imagen y podemos ver este kernel en la figura 4.5

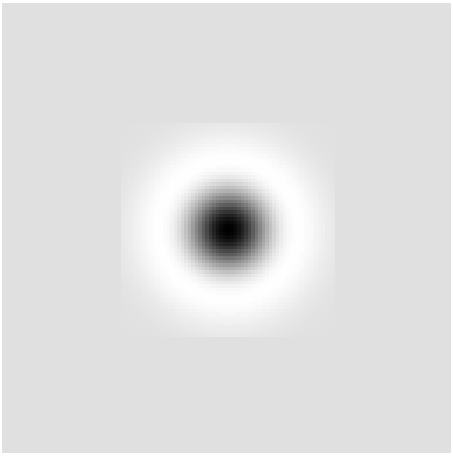


Figura 4.5: Laplaciano. Mexican Hat

4.7. Convolución con Imágenes Integrales

Definimos un pixel de la Imagen Integral $H[r,c]$ como la suma de todos los valores de la imagen original x arriba y a la izquierda de las coordenadas (r,c). Esto pude escribirse como:

$$H[r,c] = \sum_{k=0}^{r} \sum_{l=0}^{c} x[k,l] \tag{4.10}$$

Sin embargo, el calculo de la imagen integral utilizando (4.10), resulta en un algoritmo de orden $O(N^4)$ lo cual es igual de costoso de una convolución.

Para mejorar el desempeño del calculo de la imagen integral podemos reordenar las sumatorias y aplicando propiedades de la asociatividad de la suma escribir (4.10) como:

$$\begin{aligned} H[r,c] &= \sum_{k=0}^{r}\sum_{l=0}^{c} x[k,l] \\ H[r,c] &= \sum_{k=0}^{r-1}\sum_{l=0}^{c} x[k,l] + \sum_{l=0}^{c} x[r,l] \\ H[r,c] &= \sum_{k=0}^{r-1}\sum_{l=0}^{c} x[k,l] + \sum_{l=0}^{c-1} x[r,l] + x[r,c] \\ H[r,c] &= \sum_{k=0}^{r-1}\sum_{l=0}^{c} x[k,l] + \left(\sum_{k=0}^{r}\sum_{l=0}^{c-1} x[k,l] - \sum_{k=0}^{r-1}\sum_{l=0}^{c-1} x[k,l]\right) + x[r,c] \end{aligned} \tag{4.11}$$

Aplicando la definición de imagen integral (4.10) podemos escribir la recurrencia para calcular cada uno de los elementos de la imagen integral.

$$H[r,c] = H[r-1,c] + H[r,c-1] - H[r-1,c-1] + x[r,c]$$

En caso de no existir $H[r-1,c]$, $H[r,c-1]$ o $H[r-1,c-1]$ se remplazán por ceros.

Cálculo de la convolución con una caja utilizando imágenes integrales

La Figura fig:ImagenIntegral muestra el ejemplo las sumas en cuadro puntos de una caja de tamaño $(r_1 - r_0) \times (c_1 - c_0)$. La Figura fig:ImagenIntegral a) representa la imagen integral $H[r_1, c_1]$, la Fig. fig:ImagenIntegral b) la imagen integral $H[r_0 - 1, c_1]$, la Fig. fig:ImagenIntegral c) $H[r_1, c_0 - 1]$ y la Fig. fig:ImagenIntegral d) $H[r_0 - 1, c_0 - 1]$. Con estas cuatro valores es posible calcular la suma de los píxeles alrededor de una caja de tamaño $(r_1 - r_0) \times (c_1 - c_0)$. Para ello aplicamos

$$y[n,m] = H[r_1,c_1] - H[r_0-1,c_1] - H[r1,c_0-1] + H[r_0-1,c_0-1] \tag{4.12}$$

Mediante (4.11) tenemos una manera de calcular la imagen integral en un tiempo cuadrático y la convolución con una caja la podemos hacer mediante (4.12) en un tiempo cuadrático. Todo esto puede ser escrito como el algoritmo 3.

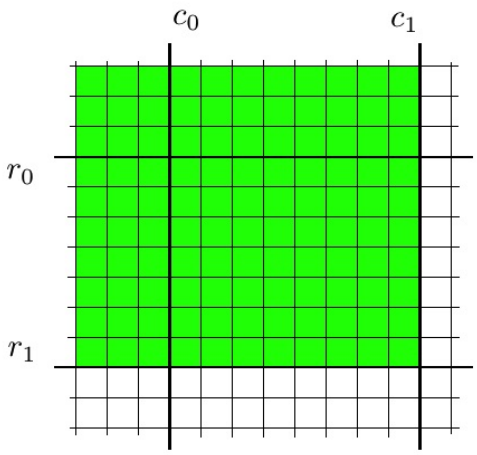
En la Figura 4.7 se muestra del lado izquierdo en amarillo la caja con la que se quiere hacer la convoulución de la imagen y a la derecha las cuatro imágenes integrales involucradas.

Note que independientemente del tamaño de la caja siempre son cuatro las operaciones

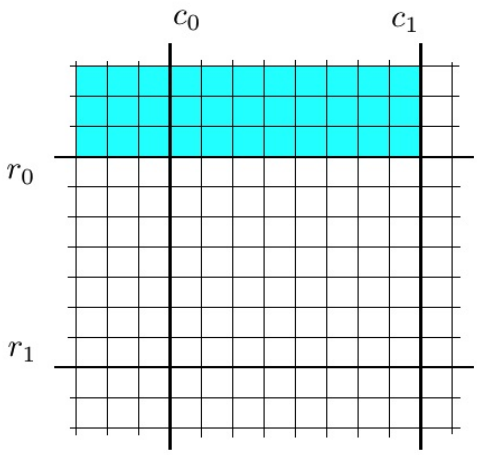
que deben realizarse.

Resultado: y
Cálculo de la imagen Integral ;
para $r \leftarrow 0$ **a** N_r **hacer**
  para $c \leftarrow 0$ **a** N_c **hacer**
    $H[r, c] \leftarrow I[r, c]$;
    si $r > 0$ **entonces**
      $H[r, c] \leftarrow H[r, c] + H[r - 1, c]$;
    fin
    si $c > 0$ **entonces**
      $H[r, c] \leftarrow H[r, c] + H[r, c - 1]$;
    fin
    si $r > 0$ *y* $c > 0$ **entonces**
      $H[r, c] \leftarrow H[r, c] - H[r - 1, c - 1]$;
    fin
  fin
fin
Cálculo de la convolución ;
para $r \leftarrow 0$ **a** N_r **hacer**
  para $c \leftarrow 0$ **a** N_c **hacer**
    si $r_1 \geq N_r$ **entonces**
      $r_1 = N_r - 1$;
    fin
    si $c_1 \geq N_c$ **entonces**
      $c_1 = N_c - 1$;
    fin
    $s_0 \leftarrow H[r_1, c_1]$, $s_1 \leftarrow 0$, $s_2 \leftarrow 0$ y $s_1 \leftarrow 0$;
    si $r_0 > 0$ **entonces**
      $s_1 \leftarrow H[r_0 - 1, c_1]$;
    fin
    si $c_0 > 0$ **entonces**
      $s_2 \leftarrow H[r_1, c_0 - 1]$;
    fin
    si $r_0 > 0$ *y* $c_0 > 0$ **entonces**
      $s_3 \leftarrow H[r0 - 1, c_0 - 1]$;
    fin
    $y[r, c] \leftarrow s_0 - s_1 - s_2 + s_3$
  fin
fin

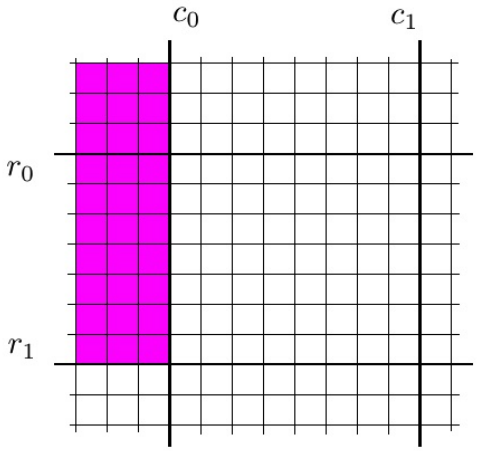
Algoritmo 3: Convolución en dos dimensiones con una caja de tamaño $(r_1 - r_0) \times (c_1 - c_0)$. Para la convolución se utilizan Imágenes Integrales.



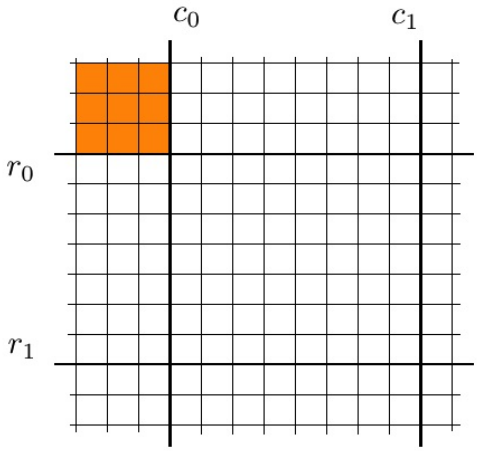
a) $H[r_1, c_1] = \sum_{k=0}^{r_1} \sum_{l=0}^{c_1} x[k, l]$



b) $H[r_0 - 1, c_1] = \sum_{k=0}^{r_0-1} \sum_{l=0}^{c_1} x[k, l]$

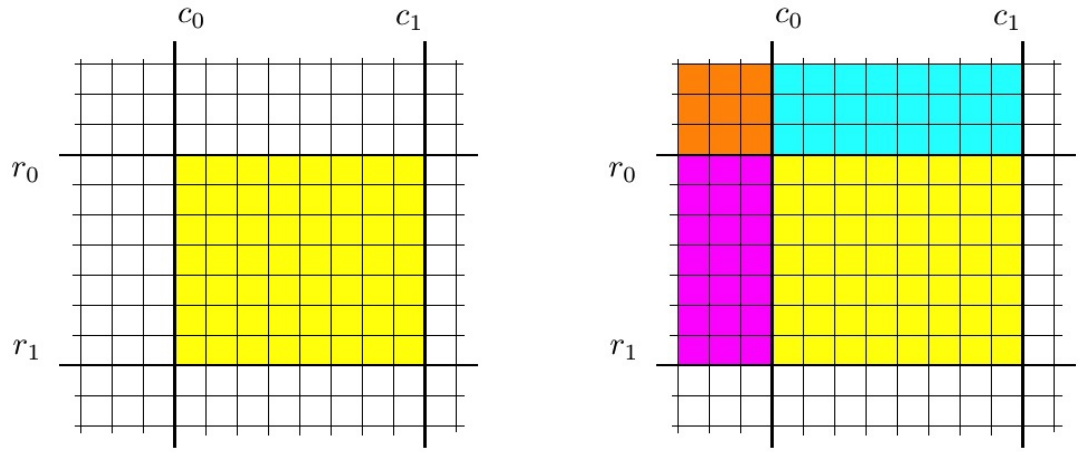


c) $H[r_1, c_0 - 1] = \sum_{k=0}^{r_1} \sum_{l=0}^{c_0-1} x[k, l]$



d) $H[r_0 - 1, c_0 - 1] = \sum_{k=0}^{r_0-1} \sum_{l=0}^{c_0-1} x[k, l]$

Figura 4.6: Ejemplos de cálculo de algunos valores de imágenes integrales



a) Área a calcular b) Traslape de regiones

Figura 4.7: Cálculo de la convolución con una caja utilizando imágenes integrales

4.7.1. Ejemplo

Dada la imagen x igual a

$$x = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

La imagen integral para este caso es:

$$H = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 \\ 4 & 8 & 12 & 16 & 20 & 24 & 28 \\ 5 & 10 & 15 & 20 & 25 & 30 & 35 \end{bmatrix}$$

Finalmente la convolución es:

$$y = \begin{bmatrix} 4 & 6 & 6 & 6 & 6 & 6 & 4 \\ 6 & 9 & 9 & 9 & 9 & 9 & 6 \\ 6 & 9 & 9 & 9 & 9 & 9 & 6 \\ 6 & 9 & 9 & 9 & 9 & 9 & 6 \\ 4 & 6 & 6 & 6 & 6 & 6 & 4 \end{bmatrix}$$

4.8. Respuesta de Sistemas lineales invariantes en el tiempo a exponenciales complejas.

La importancia de las exponenciales complejas en el estudios de sistemas LTI proviene del hecho, de que la respuesta de un sistema LTI a una entrada exponencial compleja es la misma exponencial compleja modificada solo en amplitud.

$$T\left[z^n\right] = H(z)z^n$$

donde el factor complejo de la amplitud $H(z)$ seró en general una función de la variable compleja s. Esto lo podemos mostrar haciendo

$$y(n) = \sum_{k=-\infty}^{\infty} h\left(k\right) x\left(n-k\right)$$

$$y(n) = \sum_{k=-\infty}^{\infty} h\left(k\right) z^{(n-k)}$$

$$y(n) = \sum_{k=-\infty}^{\infty} h\left(k\right) z^n z^{-k}$$

$$y(n) = z^n \sum_{k=-\infty}^{\infty} h\left(k\right) z^{-k}$$

$$y(n) = H(z)z^n$$

donde

$$H(z) = \sum_{k=-\infty}^{\infty} h\left(k\right) z^{-k}$$

hemos demostrado que cualquier exponencial compleja es una función característica de un sistema LIT.

Si la entrada de un sistema LIT de tiempo discreto se presenta como una combinación de exponenciales complejas, esto es, si

$$x\left(n\right) = \sum_{k} a_k z_k^n$$

entonces la salida es

$$y(n) = \sum_k a_k H(z_k) z_k^n$$

Transformada de Fourier

5.1. Representación de señales periódicas.

Recordemos que la exponencial compleja $e^{j(2\pi/N)n}$ es periódica con periodo N. A partir de esta podemos crear una familia de exponenciales complejas también con periodo N/k dadas por

$$\phi_k(n) = e^{jk(2\pi/N)n}$$

$$\phi_k(n) = cos(\frac{2\pi k}{N}n) + jseno(\frac{2\pi k}{N}n) \tag{5.13}$$

donde k es un número entero que representa una frecuencia fundamental. Todas estas exponenciales complejas tienen frecuencias que son múltiplos de las misma frecuencia fundamental $2\pi/N$. Así cuando $k = 0$ llamaremos a este el componente de CD de la señal (constante), cuando $k = 1$ como armónico de frecuencia $2\pi k/N$ fundamental, con $k = 2$ tenemos $4\pi k/N$ segundo armónico y así sucesivamente para todos los valores k.

Con esto queremos llegar a hacer la representación de una señal periódica como la combinación lineal de exponenciales complejas, dadas como (5.14)

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\phi_k(n)$$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{jk(2\pi/N)n} \tag{5.14}$$

donde la ecuación (5.14) es conocida como la Transformada Discreta Inversa de Fourier (TDIF) de $x[n]$ y cuyos coeficientes son $X[k]$.

Para determinar los coeficientes X de la TDIF procedemos

$$\begin{aligned}
x[0] &= \frac{1}{N}\left(X[0]\phi_0(0) + X[1]\phi_1(0) + X[2]\phi_2(0) + ... + X[N-1]\phi_{N-1}(0)\right)\\
x[1] &= \frac{1}{N}\left(X[0]\phi_0(1) + X[1]\phi_1(1) + X[2]\phi_2(1) + ... + X[N-1]\phi_{N-1}(1)\right)\\
x[2] &= \frac{1}{N}\left(X[0]\phi_0(2) + X[1]\phi_1(2) + X[2]\phi_2(2) + ... + X[N-1]\phi_{N-1}(2)\right)\\
&\vdots\\
x[N-1] &= \frac{1}{N}\left(X[0]\phi_0(N-1) + X[1]\phi_1(N-1) + ... + X[N-1]\phi_{N-1}(N-1)\right)
\end{aligned}$$

En forma matricial

$$\frac{1}{N}\begin{bmatrix} \phi_0(0) & \phi_1(0) & \dots & \phi_{N-1}(0) \\ \phi_0(1) & \phi_1(1) & \dots & \phi_{N-1}(1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(N-1) & \phi_1(N-1) & \dots & \phi_{N-1}(N-1) \end{bmatrix}\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

donde $\phi_k[n]$ se calcula utilizando (5.13), podemos representar en forma compacta esta ecuación como :

$$\frac{1}{N}\Phi X = x \tag{5.15}$$

Para el sistema de ecuaciones ponemos calcular la Transformada Discreta de Fourier X resolviendo 5.15

$$X = N\Phi^{-1}x$$

5.2. Cálculo de la Transformada Discreta de Fourier

Si bien, la Transformada Discreta de Fourier (TDF) se puede resolver a partir de un sistema de ecuaciones (5.15) como $X = \Phi^{-1}x$ no es del todo correcto, además de requerir la inversa de un sistema de ecuaciones.

Una alternativa es multiplicar, ambos lados de la ecuación (5.14), por $e^{-jr(2\pi/N)n}$

$$x[n]e^{-jr(2\pi/N)n} = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{jk(2\pi/N)n}e^{-jr(2\pi/N)n}$$

y sumar para todos los N términos de la serie

$$\sum_{n=0}^{N-1} x[n]e^{-jr(2\pi/N)n} = \frac{1}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1} X[k]e^{jk(2\pi/N)n}e^{-jr(2\pi/N)n}$$

$$\sum_{n=0}^{N-1} x[n]e^{-jr(2\pi/N)n} = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\sum_{n=0}^{N-1} e^{j(k-r)(2\pi/N)n}$$

$$\sum_{n=0}^{N-1} x[n]e^{-jr(2\pi/N)n} = \frac{1}{N}\sum_{k=0}^{N-1} X[k]s_N$$

Para la resolver s_N de la ecuación anterior

$$s_N = \sum_{n=0}^{N-1} e^{j(k-r)(2\pi/N)n} = \sum_{n=0}^{N-1} a^n$$

con $a = e^{j(k-r)(2\pi/N)}$

La solución de s_N en el caso de $(k = r)$ es:

$$s_N = \sum_{n=0}^{N-1} e^{j(0)(2\pi/N)n} = \sum_{n=0}^{N-1} 1 = N$$

y en caso de que $(k \neq)$ haremos uso de la sucesión dada por (5.16)

$$s_N = \sum_{n=0}^{N-1} a^n = \frac{1-a^N}{1-a} \tag{5.16}$$

$$s_N = \frac{1-e^{j(k-r)(2\pi/N)N}}{1-e^{j(k-r)(2\pi/N)}} = \frac{1-e^{j(k-r)(2\pi)}}{1-e^{j(k-r)(2\pi/N)}} = \frac{1-1}{1-e^{j(k-r)(2\pi/N)}} = 0$$

Por tanto si escogemos valores de r igual que k, tendremos que esta suma es igual a N y cero de lo contrario, por lo que tenemos

$$X[r] = \sum_{n=0}^{N-1} x[n]e^{-jr(2\pi/N)n}$$

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-jk(2\pi/N)n}$$

$$X[k] = \sum_{n=0}^{N-1} x[n]\phi_r^*(n)$$

En lo general representaremos la transformada Discreta de Fourier como (5.17)

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-jk(2\pi/N)n} \tag{5.17}$$

En forma matricial lo podemos expresar (5.17) como

$$\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} \phi_0^*(0) & \phi_1^*(0) & \dots & \phi_{N-1}^*(0) \\ \phi_0^*(1) & \phi_1^*(1) & \dots & \phi_{N-1}^*(1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0^*(N-1) & \phi_1^*(N-1) & \dots & \phi_{N-1}^*(N-1) \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

$$X = \Phi^* x$$

donde $\phi_k^*(n)$ es el valor conjugado de $\phi_k(n)$ y se calcula utilizando (5.13) cambiando el signo de la parte imaginaria.

5.2.1. Implementación de la Transformada de Fourier

A partir de las ecuaciones de la TDF (5.17) y de la TDIF (5.14), es necesario construir una matriz Φ^* y Φ respectivamente. Matricialmente la TDF se calcula con $X = \Phi^* x$ y TDIF como $x = \frac{1}{N}\Phi X$. Note que la diferencia es un signo en la parte imaginaria de las exponenciales complejas y la división entre N. La implementación genérica de la TDF y TDIF se muestra en el siguiente código denominado DFT_1D:

```
void funciones::DFT_1D(float *entrada_r, float *entrada_i,
        float * salida_r, float * salida_i, int signo, int N){

    int n, k;
    float **Ar, **Ai
    Ar = funciones::Matriz(N,N);
    Ai = funciones::Matriz(N,N);
    float w = 2.0*(3.1415926)/(float) N, suma_r, suma_i;

    float factor;

    factor = signo == 1 ? 1.0/(double) N : 1.0 ;

    for(n=0; n<N; n++){
```

```
            for(k=n; k<N; k++) {
                Ar[n][k] = cos(w*n*k);
                Ar[k][n] = Ar[n][k];
                Ai[n][k] = signo*sin(w*n*k);
                Ai[k][n] = Ai[n][k];
            }
        }

        for(k=0; k<N; k++) {
            suma_r = 0;
            suma_i = 0;

            for(n=0; n<N; n++){
                suma_r += (Ar[k][n]*entrada_r[n] - Ai[k][n]*entrada_i[n]);
                suma_i += (Ar[k][n]*entrada_i[n] + Ai[k][n]*entrada_r[n]);
            }

            salida_r[k] = suma_r*factor;
            salida_i[k] = suma_i*factor;
        }
    }
```

Para evitar confusiones, de manera particular, para el calculo de la TDF utilizaremos la función

```
void funciones::Fourier(float *entrada_r, float *entrada,
        float *salida_r, float * salida_i, int N){
     DFT_1D(entrada_r, entrada_i, salida_r, salida_i, -1, N);
}
```

y para la TDIF utilizamos

```
 void funciones::Inversa_Fourier(float *entrada_r, float * entrada_i,
        float * salida_r, float * salida_i, int N){
     DFT_1D(entrada_r, entrada_i, salida_r, salida_i, 1, N);
}
```

Se puede comprobar los resultados de estas funciones en Mathematica haciendo Fourier[x, FourierParameters$\rightarrow \{1, -1\}$] y InverseFourier[X, FourierParameters$\rightarrow \{1, -1\}$].

5.2.2. Ejemplo

Determinar la TDF X para una señal $x = [1, 4, 1, 1]^T$

De acuerdo con la ecuación (5.17), tenemos que calcular la TDF X, haciendo simplemente la multiplicación:

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ 4 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ -j3 \\ -3 \\ j3 \end{bmatrix}$$

Si nuestra formulación es consistente la transformada Inversa de Fourier de X, nos debe dar el vector original x

$$\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \begin{bmatrix} 7 \\ -j3 \\ -3 \\ j3 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 1 \\ 1 \end{bmatrix}$$

5.2.3. Ejemplo

Encontrar los coeficientes de la TDF para la señal mostrada en la Fig. 6.16(a), cuyos elementos son $x = [1, 1, 1, 0, 0, 0, 0, 1, 1]^T$.

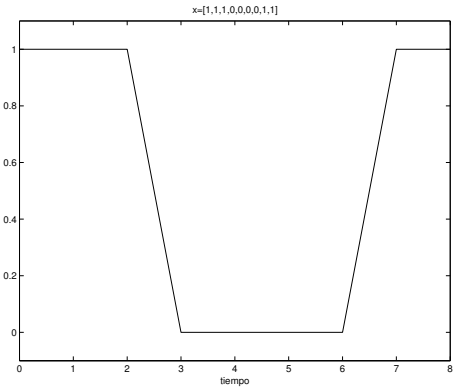


Figura 5.1: Señal a descomponer en Fourier

Para encontrar los coeficientes de Fourier de la señal x, es necesario calcular :

$$X = \Phi^* x$$

donde Φ es la matriz de exponenciales complejas (5.13) y N es la cantidad de muestras de la señal, en este caso es 9. Así la matriz es:

$$\Phi^* = \begin{pmatrix} \phi_0(0) & \phi_1(0) & \phi_2(0) & \phi_3(0) & \phi_4(0) & \phi_5(0) & \phi_6(0) & \phi_7(0) & \phi_8(0) \\ \phi_0(1) & \phi_1(1) & \phi_2(1) & \phi_3(1) & \phi_4(1) & \phi_5(1) & \phi_6(1) & \phi_7(1) & \phi_8(1) \\ \phi_0(2) & \phi_1(2) & \phi_2(2) & \phi_3(2) & \phi_4(2) & \phi_5(2) & \phi_6(2) & \phi_7(2) & \phi_8(2) \\ \phi_0(3) & \phi_1(3) & \phi_2(3) & \phi_3(3) & \phi_4(3) & \phi_5(3) & \phi_6(3) & \phi_7(3) & \phi_8(3) \\ \phi_0(4) & \phi_1(4) & \phi_2(4) & \phi_3(4) & \phi_4(4) & \phi_5(4) & \phi_6(4) & \phi_7(4) & \phi_8(4) \\ \phi_0(5) & \phi_1(5) & \phi_2(5) & \phi_3(5) & \phi_4(5) & \phi_5(5) & \phi_6(5) & \phi_7(5) & \phi_8(5) \\ \phi_0(6) & \phi_1(6) & \phi_2(6) & \phi_3(6) & \phi_4(6) & \phi_5(6) & \phi_6(6) & \phi_7(6) & \phi_8(6) \\ \phi_0(7) & \phi_1(7) & \phi_2(7) & \phi_3(7) & \phi_4(7) & \phi_5(7) & \phi_6(7) & \phi_7(7) & \phi_8(7) \\ \phi_0(8) & \phi_1(8) & \phi_2(8) & \phi_3(8) & \phi_4(8) & \phi_5(8) & \phi_6(8) & \phi_7(8) & \phi_8(8) \end{pmatrix}$$

La parte real de Φ^* es

$$\Phi_r^* = \begin{pmatrix} 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\ 1.000 & 0.766 & 0.174 & -0.500 & -0.940 & -0.940 & -0.500 & 0.174 & 0.766 \\ 1.000 & 0.174 & -0.940 & -0.500 & 0.766 & 0.766 & -0.500 & -0.940 & 0.174 \\ 1.000 & -0.500 & -0.500 & 1.000 & -0.500 & -0.500 & 1.000 & -0.500 & -0.500 \\ 1.000 & -0.940 & 0.766 & -0.500 & 0.174 & 0.174 & -0.500 & 0.766 & -0.940 \\ 1.000 & -0.940 & 0.766 & -0.500 & 0.174 & 0.174 & -0.500 & 0.766 & -0.940 \\ 1.000 & -0.500 & -0.500 & 1.000 & -0.500 & -0.500 & 1.000 & -0.500 & -0.500 \\ 1.000 & 0.174 & -0.940 & -0.500 & 0.766 & 0.766 & -0.500 & -0.940 & 0.174 \\ 1.000 & 0.766 & 0.174 & -0.500 & -0.940 & -0.940 & -0.500 & 0.174 & 0.766 \end{pmatrix}$$

y la parte imaginaria de Φ^* es

$$\Phi_i^* = \begin{pmatrix} 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & -0.643 & -0.985 & -0.866 & -0.342 & 0.342 & 0.866 & 0.985 & 0.643 \\ 0.000 & -0.985 & -0.342 & 0.866 & 0.643 & -0.643 & -0.866 & 0.342 & 0.985 \\ 0.000 & -0.866 & 0.866 & 0.000 & -0.866 & 0.866 & 0.000 & -0.866 & 0.866 \\ 0.000 & -0.342 & 0.643 & -0.866 & 0.985 & -0.985 & 0.866 & -0.643 & 0.342 \\ 0.000 & 0.342 & -0.643 & 0.866 & -0.985 & 0.985 & -0.866 & 0.643 & -0.342 \\ 0.000 & 0.866 & -0.866 & 0.000 & 0.866 & -0.866 & 0.000 & 0.866 & -0.866 \\ 0.000 & 0.985 & 0.342 & -0.866 & -0.643 & 0.643 & 0.866 & -0.342 & -0.985 \\ 0.000 & 0.643 & 0.985 & 0.866 & 0.342 & -0.342 & -0.866 & -0.985 & -0.643 \end{pmatrix}$$

La solución del producto $(\Phi_r^* + j\Phi_i^*)x$ es:

$$X = \begin{pmatrix} 5.0000 \\ 2.87939 \\ -0.5321 \\ -1.0000 \\ 0.6527 \\ 0.6527 \\ -1.0000 \\ -0.5321 \\ 2.8794 \end{pmatrix}$$

Note que X no contiene elementos imaginario.

A continuación se va a reconstruir la señal original usando los coeficientes de Fourier obtenidos calculando,

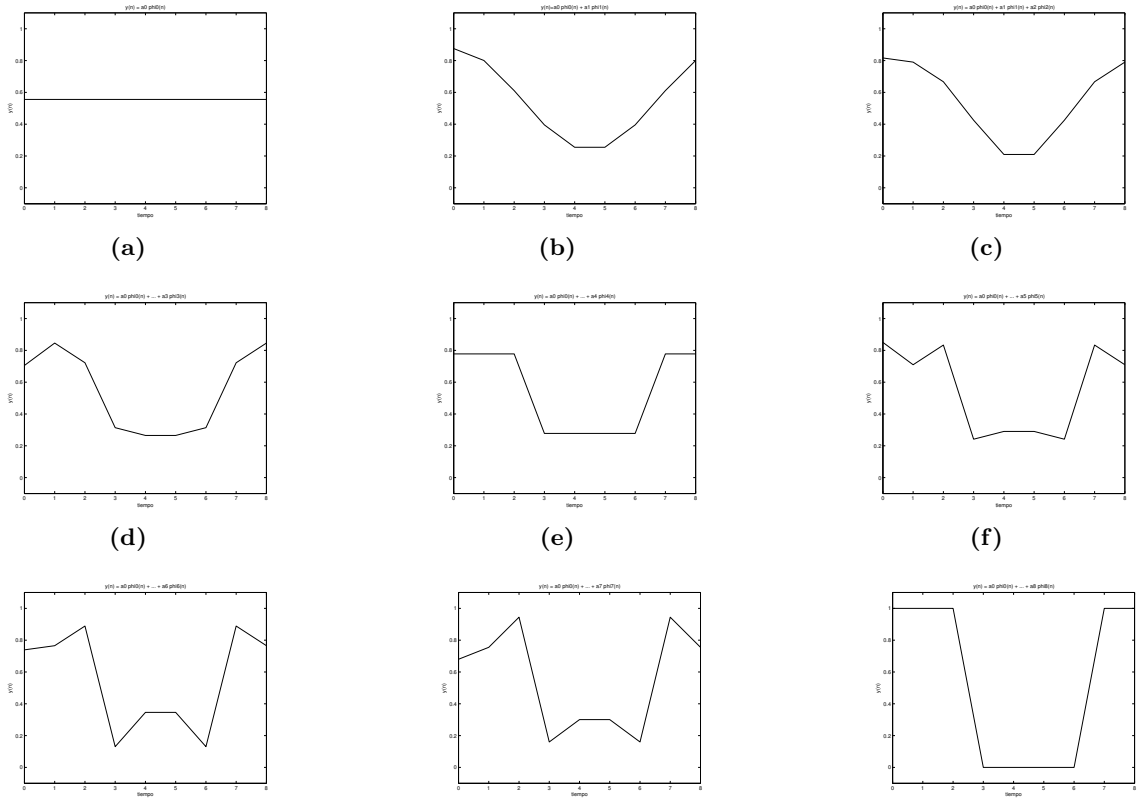
$$y = \Phi X$$

Para poder observar la importancia de cada uno de los coeficientes encontrados, primero se va a reconstruir la señal usando un solo coeficiente de manera que $X = [X[0], 0, 0, 0, 0, 0, 0, 0, 0]^T$, después se van a usar dos coeficientes, de modo que $X = [X[0], X[1], 0, 0, 0, 0, 0, 0, 0]^T$, y así sucesivamente hasta utilizar los 9 coeficientes. Los resultados de este ejercicio se muestran en las Figs. 5.2(a)-5.2(i).

5.3. Transformada Rapida de Fourier

La trasformada Discreta de Fourier TDF es $O(N^2)$ y existe la Transformada Rápida de Fourier FFT la cual es $O(N \log N)$, para una señal en una sola dimensión. Para el caso de señales en dos dimensiones, la TDF es $O((NM)^2)$ y la FFT es $O(NM \log NM)$, razón por la cual se utiliza la FFT. Sin embargo, la FFT requiere que el tamaño de la señales sea potencia de 2, es decir $N = 2^r$ y $M = 2^s$. Todas la propiedades de la transformada de Fourier son validas para el caso multidimensional y para hacer la demostración solo hay que tener en cuenta que se trata de arreglos bidimensionales.

Danielson y Lanczos mostraron que la transformada Discreta de Fourier para una señal x de longitud N, puede ser descrita como la suma de dos transformadas Discretas de Fourier, cada uno de tamaño $N/2$. Esto lo podemos mostrar a partir de la definición de la TDF (5.17), podemos reagrupar la suma en los componentes $N/2$ pares e impares como



(a) **(b)** **(c)**

(d) **(e)** **(f)**

(g) **(h)** **(i)**

Figura 5.2: Reconstrucción de la señal usando los coeficientes de Fourier

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j(\frac{2\pi k}{N})n} = \sum_{n=0}^{N/2-1} x[2n]e^{-j(\frac{2\pi k}{N})2n} + \sum_{n=0}^{N/2-1} x[2n+1]e^{-j(\frac{2\pi k}{N})(2n+1)}$$

Agrupando términos tenemos

$$X[k] = \sum_{n=0}^{N/2-1} x[2n]e^{-j(\frac{2\pi k}{N})2n} + e^{-j(\frac{2\pi k}{N})} \sum_{n=0}^{N/2-1} x[2n+1]e^{-j(\frac{2\pi k}{N})(2n)}$$

$$X[k] = \sum_{n=0}^{N/2-1} x[2n]e^{-j(\frac{2\pi k}{N/2})n} + W[N] \sum_{n=0}^{N/2-1} x[2n+1]e^{-j(\frac{2\pi k}{N/2})n}$$

El paso recursivo de la Transformada Rapida de Fourier es

$$TDF(x) = TDF(x_p) + W(N) \times TDF(x_i)$$

y el caso Base será cuando la señal sea de tamaño $N = 1$

$$X[0] = x[0]$$

donde x_p y x_i son vectores de tamaño $N/2$ con los coeficientes pares e impares de x y $W(N) = e^{-j(\frac{2\pi k}{N})}$.

5.3.1. Ejemplo

Calcular la Transformada Rapida de Fourier de la señal $x = [3, 2, 1, 5, 9, 4, 7, 6]^T$

$$\begin{aligned}
TDF(3,2,1,5,9,4,7,6) &= TDF(3,1,9,7) + W(8)TDF(2,5,4,6) \\
&= TDF(3,9) + W(4)TDF(1,7) \\
&\quad +W(8)(TDF(2,4) + W(4)TDF(5,6))) \\
&= TDF(3) + W(2)TDF(9) \\
&\quad +W(4)(TDF(1) + W(2)TDF(7)) \\
&\quad +W(8)(TDF(2) + W(2)TDF(4)) \\
&\quad +W(8)W(4)(TDF(5) + W(2)TDF(6))
\end{aligned}$$

Sustituyendo valores tenemos

$$\begin{aligned}
X[k] = 3 + 9 \times e^{-j(2\pi k/2)} + 1 \times e^{-j(2\pi k/4)} + 7 \times e^{-j(3\pi k/2)} \\
+2 \times e^{-j(2\pi k/8)} + 4 \times e^{-j(10\pi k/8)} + 5 \times e^{-j(6\pi k/8)} + 6 \times e^{-j(14\pi k/8)} \\
X[k] = [37.0000, -6.70711 + j8.1213, 4.0000 + j5, -5.2929 - j3.8787, \\
3.0000, -5.29289 + j3.8787, 4.0000 - j5.0000, -6.7071 - j8.1213]^T
\end{aligned}$$

5.3.2. Complejidad

Consideremos que el número de elementos es potencia de 2, así por ejemplo $N = 32 = 2^5$. Definimos C_1 el tiempo para calcular la TDF de una señal con un elemento y C_2 el tiempo para sumar la TDF de los elementos pares e impares. Así el tiempo para calcular la TDF de una señal con N elementos lo resolvemos con la recurrencia

$$T(N) = 2T(N/2) + NC_2$$

lo cual significa que el tiempo calcular la TDF de N datos será equivalente al tiempo de la TDF de dos señales la mitad del tamaño más el tiempo de sumar los N elementos

Por ejemplo en el caso de tener $N = 32$ elementos los tiempo los calculamos de la manera siguiente:

Llamado	*Regresa*
$T(32) = 2T(16) + 32C_2$	$T(32) = 32C_1 + 160C_2$
$T(16) = 2T(8) + 16C_2$	$T(16) = 16C_1 + 64C_2$
$T(8) = 2T(4) + 8C_2$	$T(8) = 8C_1 + 24C_2$
$T(4) = 2T(2) + 4C_2$	$T(4) = 4C_1 + 8C_2$
$T(2) = 2T(1) + 2C_2$	$T(2) = 2C_1 + 2C_2$
$T(1) = C_1$	C_1

De la solución podemos generalizar

$$T(N) = NC_1 + kNC_2$$

donde $k = log_2(N)$

De acuerdo con esto la complejidad es

$$T(N) = NC_1 + Nlog_2(N)C_2 = O(Nlog(N))$$

5.3.3. Implementación

Para su implementación se tomo el código del libro Numerical Recipes in C correspondiente a la función four1. Para facilitar su uso se escribió la función fft1 la cual se encarga de ordenar los vectores p y q correspondientes a las partes real e imaginaria.

```
/**
 * Funcion Auxiliar para calcular la Transformada Rapida de Fourier
 * @param p Señal de entrada
 * @param q Señal de salida
 * @param npts Numero de elementos
 * @param Direccion -1 Transformada de Fourier
 *                   1 Transformada Inversa de Fourier
 */

void funciones::fft1(float *p, float *q, int npts, int Direccion) {
    float *fh = Vector(2 * (npts + 1));
    float fact = 1.0;
    int j;
    if (Direccion == 1)
        fact = 1.0 / (float) (npts);

    for (j = 0; j < 2 * npts; j += 2) {
        fh[j + 1] = p[j / 2];
        fh[j + 2] = q[j / 2];
    }

    four1(fh, npts, Direccion);

    for (j = 0; j < 2 * npts; j += 2) {
        p[j / 2] = fact * fh[j + 1];
        q[j / 2] = fact * fh[j + 2];
    }
}

/**
 * Transformada Rapida de Fourier en 1 dimension
 * @param data datos de entrada
 * @param nn tamaño de la muestra
 * @param isign -1 Transformada Rapida de Fourier
 *               1 Transformada Inversa de Fourier
 */

void funciones::four1(float *data, int nn, int isign) {
    int n, mmax, m, j, istep, i;
    float wtemp, wr, wpr, wpi, wi, theta;
    float tempr, tempi, tt;
```

```
n = nn << 1;
j = 1;
for (i = 1; i < n; i += 2) {
    if (j > i) {
        tt = data[j];
        data[j] = data[i];
        data[i] = tt;
        tt = data[j + 1];
        data[j + 1] = data[i + 1];
        data[i + 1] = tt;
    }
    m = n >> 1;
    while (m >= 2 && j > m) {
        j -= m;
        m >>= 1;
    }
    j += m;
}
mmax = 2;
while (n > mmax) {
    istep = 2 * mmax;
    theta = 6.28318530717959 / (isign * mmax);
    wtemp = sin(0.5 * theta);
    wpr = -2.0 * wtemp * wtemp;
    wpi = sin(theta);
    wr = 1.0;
    wi = 0.0;
    for (m = 1; m < mmax; m += 2) {
        for (i = m; i <= n; i += istep) {
            j = i + mmax;
            tempr = wr * data[j] - wi * data[j + 1];
            tempi = wr * data[j + 1] + wi * data[j];
            data[j] = data[i] - tempr;
            data[j + 1] = data[i + 1] - tempi;
            data[i] += tempr;
            data[i + 1] += tempi;
        }
        wr = (wtemp = wr) * wpr - wi * wpi + wr;
        wi = wi * wpr + wtemp * wpi + wi;
    }
```

```
        mmax = istep;
    }
}
```

5.4. Transformada de Fourier de algunas funciones interesantes

5.4.1. Exponencial Compleja

Dado $x[n] = e^{j\left(\frac{2\pi}{N}\right)k_0 n}$ la transformada de Fourier esta dada como

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\left(\frac{2\pi}{N}\right)kn}$$

$$X[k] = \sum_{n=0}^{N-1} e^{j\left(\frac{2\pi}{N}\right)k_0 n} e^{-j\left(\frac{2\pi}{N}\right)kn}$$

$$X[k] = \sum_{n=0}^{N-1} e^{j\left(\frac{2\pi}{N}\right)(k_0-k)n}$$

La sumatoria tendrá solución N en el caso de que $k = k_0$ dado que $e^0 = 1$ y en caso contrario aplicamos la formula (5.16)

$$s[n] = \frac{1-a^N}{1-a} = \frac{1-e^{j\left(\frac{2\pi}{N}\right)(k_0-k)N}}{1-e^{j\left(\frac{2\pi}{N}\right)(k_0-k)}} = 0$$

Por lo tanto la transformada de Fourier es

$$e^{j\left(\frac{2\pi}{N}\right)k_0 n} \overset{\mathcal{F}}{\leftrightarrow} N\delta\left(k-k_0\right)$$

5.4.2. Función seno

Dada la función seno, podemos hacer la representación de esta función como

$$sen\left(\frac{2\pi}{N}k_0 n\right) = \frac{1}{2j}\left[e^{j\left(\frac{2\pi}{N}\right)k_0 n} - e^{-j\left(\frac{2\pi}{N}\right)k_0 n}\right]$$

dado que tenemos la suma de dos exponenciales complejas, la transformada de Fourier es

$$sen\left(\frac{2\pi}{N}k_0n\right) \overset{\mathcal{F}}{\leftrightarrow} -j\frac{N}{2}\delta\left(k-k_0\right)+j\frac{N}{2}\delta\left(k+k_0\right)$$

5.4.3. Función coseno

La representación de la función coseno la podemos dar como

$$\cos\left(\frac{2\pi}{N}k_0n\right)=\frac{1}{2}\left[e^{j\left(\frac{2\pi}{N}\right)k_0n}+e^{-j\left(\frac{2\pi}{N}\right)k_0n}\right]$$

de manera similar que en la función seno tenemos

$$\cos\left(\frac{2\pi}{N}k_0n\right) \overset{\mathcal{F}}{\leftrightarrow} \frac{N}{2}\delta\left(k-k_0\right)+\frac{N}{2}\delta\left(k+k_0\right)$$

5.4.4. Función impulso unitario

Recordemos que la función impulso unitario es

$$\delta\left(n\right)=\begin{cases} 1 \text{ si } n=0 \\ 0 \text{ en caso contrario} \end{cases}$$

la transformada de Fourier para esta la calculamos

$$X[k]=\sum_{n=0}^{N-1}\delta\left(n-n_0\right)e^{-j\left(\frac{2\pi}{N}\right)kn}$$

$$X[k]=e^{-j\left(\frac{2\pi}{N}\right)kn_0}$$

podemos comprobar que $\left|e^{-j\left(\frac{2\pi}{N}\right)kn_0}\right|=1$ para cualquier valor que tome k por lo que finalmente la magnitud de la transformada de Fourier es

$$\delta\left(n-n_0\right) \overset{\mathcal{F}}{\leftrightarrow} 1$$

5.4.5. Constante

Consideremos una función $x(n) = a$ la transformada de Fourier es

$$X[k] = \sum_{n=0}^{N-1} ae^{-j\left(\frac{2\pi}{N}\right)kn}$$
$$X[k] = a\sum_{n=0}^{N-1} e^{-j\left(\frac{2\pi}{N}\right)kn}$$

la sumatoria tendrá la solución N para $k = 0$ y 0 para $k \neq 0$ por lo cual

$$a \overset{\mathcal{F}}{\leftrightarrow} Na\delta(k)$$

5.4.6. Escalón Unitario

La función escalón unitario esta definida como

$$u(n_0) = \begin{cases} 1 \text{ si } n >= n_0 \\ 0 \text{ en caso contrario} \end{cases}$$

la transformada de Fourier estará dada como

$$U[k] = \sum_{n=0}^{N-1} u(n_0)e^{-j\left(\frac{2\pi}{N}\right)kn}$$
$$U[k] = \sum_{n=n_0}^{N-1} e^{-j\left(\frac{2\pi}{N}\right)kn}$$
$$U[k] = \sum_{n=0}^{N-1} e^{-j\left(\frac{2\pi}{N}\right)kn} - \sum_{n=0}^{n_0-1} e^{-j\left(\frac{2\pi}{N}\right)kn}$$

Lo cual da como resultado

$$U[k] = \begin{cases} N - n_0 \text{ si } k = 0 \\ \widehat{U}[k] \text{ en caso contrario} \end{cases}$$

con

$$\begin{aligned}
\widehat{U}[k] &= -\frac{1-e^{-j\left(\frac{2\pi}{N}\right)kn_0}}{1-e^{-j\left(\frac{2\pi}{N}\right)k}} \\
\widehat{U}[k] &= -\frac{e^{-j\left(\frac{\pi}{N}\right)kn_0}\left(e^{j\left(\frac{\pi}{N}\right)kn_0}-e^{-j\left(\frac{\pi}{N}\right)kn_0}\right)}{e^{-j\left(\frac{\pi}{N}\right)k}\left(e^{j\left(\frac{\pi}{N}\right)k}-e^{-j\left(\frac{\pi}{N}\right)k}\right)} \\
\widehat{U}[k] &= -\frac{\sin(\frac{\pi n_0 k}{N})}{\sin(\frac{\pi k}{N})}e^{-j\left(\frac{\pi}{N}\right)k(n_0-1)}
\end{aligned}$$

5.4.7. Caja

Consideremos una función $x\,(n)$ dada por la siguiente expresión

$$c[n] = \begin{cases} 1 \text{ si } -d < n < d \\ 0 \text{ si no} \end{cases}$$

Aplicando la formula podemos ver

$$\begin{aligned}
C[k] &= \sum_{n=-N/2}^{N/2} c[n]e^{-j\left(\frac{2\pi}{N}\right)kn} \\
C[k] &= \sum_{n=-d}^{d} e^{-j\left(\frac{2\pi}{N}\right)kn}
\end{aligned}$$

Podemos organizar los términos de la siguiente forma

$$\begin{aligned}
C[k] &= \sum_{n=-d}^{-1} e^{-j\left(\frac{2\pi}{N}\right)kn} + \sum_{n=0}^{d} e^{-j\left(\frac{2\pi}{N}\right)kn} \\
C[k] &= \sum_{n=1}^{d} e^{-j\left(\frac{2\pi}{N}\right)k(-n)} + \sum_{n=0}^{d} e^{-j\left(\frac{2\pi}{N}\right)kn} \\
C[k] &= \sum_{n=0}^{d} e^{j\left(\frac{2\pi}{N}\right)kn} - 1 + \sum_{n=0}^{d} e^{-j\left(\frac{2\pi}{N}\right)kn}
\end{aligned}$$

Lo cual da como resultado

$$C[k] = \begin{cases} \text{si } k = 0 \\ \ldots\ldots\ 2d+1 \\ \text{si no} \\ \ldots\ldots\ \left(\frac{1-e^{j\frac{2\pi}{N}k(d+1)}}{1-e^{j\frac{2\pi}{N}k}} + \frac{1-e^{-j\frac{2\pi}{N}k(d+1)}}{1-e^{-j\frac{2\pi}{N}k}} - 1 \right) \end{cases}$$

En la figura 5.3 se presenta la parte real de esta función ya que la imaginaria es cero.

La transformada de Fourier continua de esta función, esta dada por la siguiente expresión

$$C[k] = \int_{-\infty}^{\infty} c(t) e^{-j(\frac{2\pi}{N})kt} dt$$

al sustituir nuestra función obtenemos

$$\begin{aligned} C[k] &= \int_{-d}^{d} e^{-j(\frac{2\pi}{N})kt} dt \\ C[k] &= \frac{1}{\left[-j(\frac{2\pi}{N})k\right]} \, e^{-j(\frac{2\pi}{N})kt} \Big|_{-d}^{d} \\ C[k] &= \frac{1}{\left[-j(\frac{2\pi}{N})k\right]} \, (-2j) \, sen \left[\frac{2\pi}{N} kd \right] \\ C[k] &= N \frac{sen \left[\frac{2\pi}{N} kd\right]}{\pi k} \end{aligned}$$

la cual luce como 5.4

5.4.8. Ejemplos

En un programa en Java que utilice las clases funciones y gráfica, obtener y explicar brevemente la Transformada de Fourier Discreta (TFD) de las siguientes señales:

1. $x_1[n] = \cos(\frac{2\pi}{N}n) + \sin(\frac{10\pi}{N}n)$
2. $x_2[n] = \delta[n] + \delta(n-3)$
3. $x_3[n] = 1 + \delta(n+5) + cos(\frac{6\pi}{N}n) + \exp^{-j(\frac{2\pi}{N})n}$

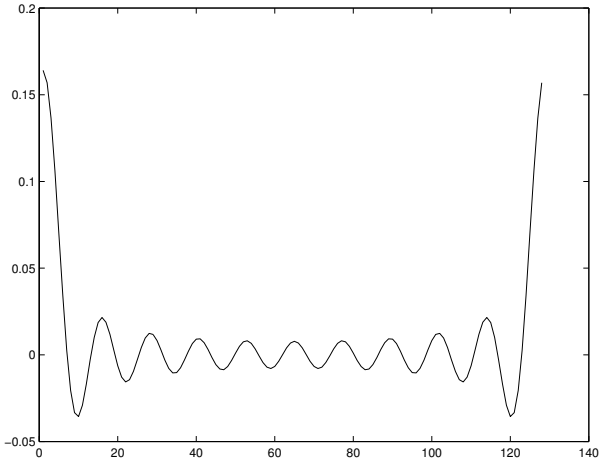


Figura 5.3: Función Sinc con d = 10 y N = 128.

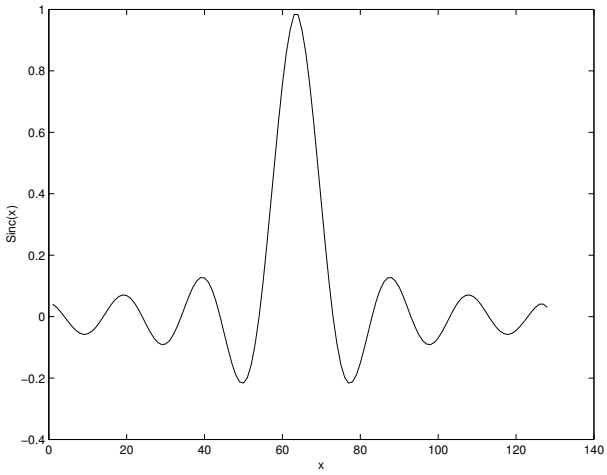
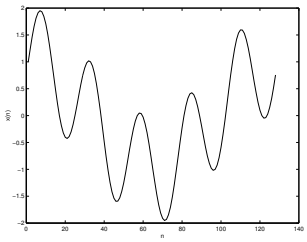


Figura 5.4: Función Sinc con k = 10 y N = 128.

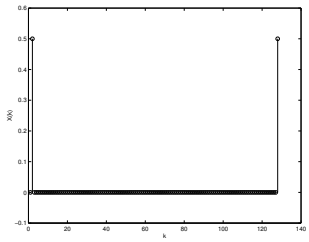
Ejercicio 1

La figura 5.4.8 muestra las cuatro gráficas que se obtienen al aplicar la TFD en la señal 1. Se puede observar en la figura 5.5(a) la señal que contiene parte real pero no contiene parte imaginaria, además la parte real se forma por un seno y un coseno. El resultado de aplicar la TFD a esta señal equivale a aplicar la TFD a las dos señales que la componen por separado y luego sumarlas. En las figuras 5.5(b) y 5.5(c) se puede observar el resultado de la aplicación de la TFD a la señal 1, cuya solución analítica es:

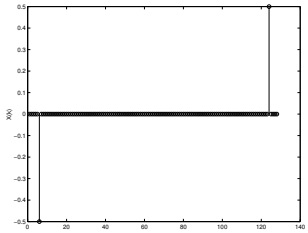
$$X_1[k] = \frac{N}{2}\delta(k-1) + \frac{N}{2}\delta(k+1) - j\frac{N}{2}\delta(k-5) + j\frac{N}{2}\delta(k+5)$$



(a) ***x 1***



(b) ***Parte real de la FFT(x)***



(c) ***Parte imaginaria de la FFT(x)***

Figura 5.5: Señal 1 y su transformada de Fourier

Con las figuras podemos constatar que el coseno efectivamente contribuye con dos impulsos positivos en $k = 1$ y $k = -1$. Asimismo, el seno contribuye con dos pulsos con diferente signo en la parte imaginaria de la señal, el positivo en $k = 5$ y el negativo en $k = -5$.

Ejercicio 2

La solución analítica de la señal 2 es:

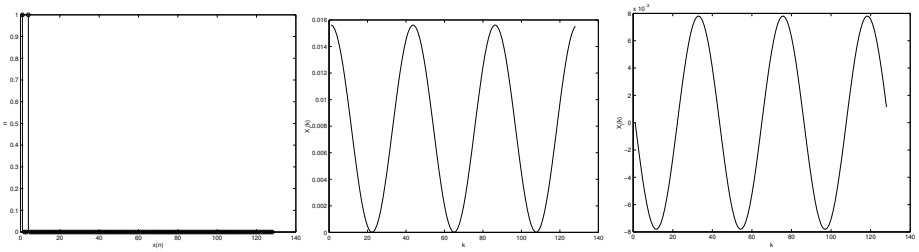
$$X_2[k] = 1 + \cos\left(\frac{6\pi}{N}k\right) - j\sin\left(\frac{6\pi}{N}k\right)$$

Las figura 5.4.8 muestra las gráficas de la señal original su TFD. Observando las gráficas 5.6(b) y 5.6(c) se puede observar que existe un coseno con frecuencia 3 en la parte real y un seno con igual frecuencia en la parte imaginaria. La contribución del impulso en el origen casi no se percibe en las gráficas, ya que es es una constante que se suma al resto de la señal.

Ejercicio 3

Las figuras 5.7(a) y 5.7(b) muestra la parte real e imaginaria respectivamente de la señal 3. Al aplicar la TFD de manera analítica a la señal se obtiene:

$$X_3[k] = N\delta[k] + \cos\left(\frac{6\pi}{N}k\right) - j\sin\left(\frac{6}{N}\right) + N\delta(k+1)$$



(a) ***Señal real*** **(b)** ***Parte real de la FFT(x)*** **(c)** ***Parte imaginaria de la FFT(x)***

Figura 5.6: Señal 2 y su transformada de Fourier

La contribución de la constante se puede observar en la gráfica de la figura 5.7(c) como un impulso en 1 de la parte real. La contribución del impulso es coseno con frecuencia 5 en la parte real y un seno negativo con frecuencia 5 en la parte imaginaria (figura 5.7(d)). El coseno contribuye con un par de impulsos en la parte real, cuando $k = 3$ y $k = -3$, y finalmente el exponencial complejo contribuye con un pulso en $k = -1$.

5.5. Propiedades de la Transformada Discreta de Fourier

La transformada de Fourier discreta, presenta algunas propiedades un cuanto diferentes que la transformada continua. A continuación se mencionan algunas y se da la demostración de tales

5.5.1. Periodicidad

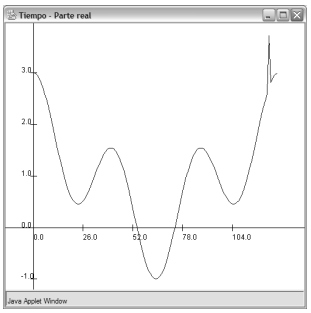
Sea una señal $x[n]$, mostrar que la transformada de Fourier, $X[k]$ tiene periodo N.

Si la función es periódica entonces

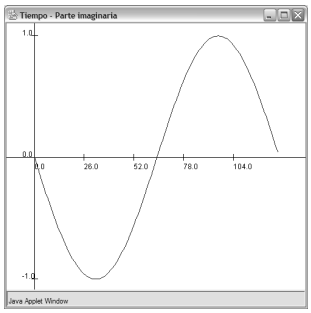
$$X[k] = X[k + N]$$

La TDF (5.17) de x es

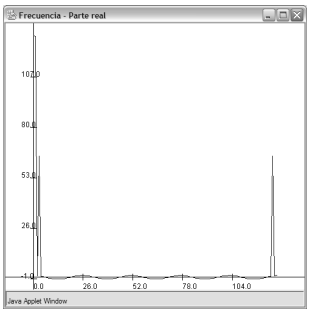
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\left(\frac{2\pi}{N}\right)nk}$$



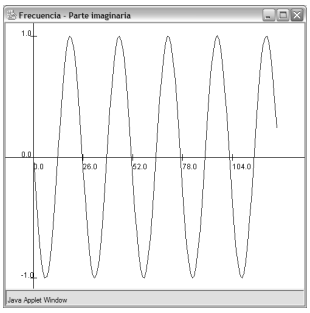
(a) *Señal real en el tiempo*



(b) *Señal imaginaria en el tiempo*



(c) *Señal real en la frecuencia*



(d) *Señal imaginaria en la frecuencia*

Figura 5.7: Señal 3 y su transformada de Fourier

y la TDF para $x(n+N)$ se calcula a partir de (5.17) como

$$\begin{aligned} X[k+N] &= \sum_{n=0}^{N-1} x[n]e^{-j\left(\frac{2\pi}{N}\right)n(k+N)} \\ X[k+N] &= \sum_{n=0}^{N-1} x[n]e^{-j\left(\frac{2\pi}{N}\right)nk}e^{-j\left(\frac{2\pi}{N}\right)nN} \\ X[k+N] &= \sum_{n=0}^{N-1} x[n]e^{-j\left(\frac{2\pi}{N}\right)nk} = X[k] \end{aligned}$$

Note que

$$X[k+N] = X[k]$$

5.5.2. Linealidad

Si

$$x_1[n] \overset{\mathcal{F}}{\leftrightarrow} X_1[k]$$
$$x_2[n] \overset{\mathcal{F}}{\leftrightarrow} X_2[k]$$

entonces

$$ax_1[n] + bx_2[n] \overset{\mathcal{F}}{\leftrightarrow} aX_1[k] + bX_2[k]$$

Prueba: Comenzamos por la TDF (5.17)

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\left(\frac{2\pi}{N}\right)nk}$$

para la suma de $ax_1[n] + bx_2[n]$ queda

$$\hat{X}[k] = \sum_{n=0}^{N-1} \left[ax_1[n] + bx_2[n]\right] e^{-j\left(\frac{2\pi}{N}\right)nk}$$
$$\hat{X}[k] = a\sum_{n=0}^{N-1} x_1[n]e^{-j\left(\frac{2\pi}{N}\right)nk} + b\sum_{n=0}^{N-1} x_2[n]e^{-j\left(\frac{2\pi}{N}\right)nk}$$
$$\hat{X}[k] = aX_1[k] + bX_2[k]$$

5.5.3. Desplazamiento en tiempo

Si

$$x[n] \overset{\mathcal{F}}{\leftrightarrow} X[k]$$
$$x[n - n_0] \overset{\mathcal{F}}{\leftrightarrow} e^{-j\left(\frac{2\pi}{N}\right)n_0 k}X[k]$$

Demostración:

La TDF (5.17) de la señal desplazada $x[n - n_0]$ es:

$$X[k] = \sum_{n=0}^{N} x[n - n_0]e^{j\left(\frac{2\pi}{N}\right)nk}$$

haciendo en cambio de variable $\widehat{n} = n - n_0$ tenemos

$$\begin{aligned} X_2[k] &= \sum_{\widehat{n}=-n_0}^{N-1-n_0} x[\widehat{n}] e^{-j\left(\frac{2\pi}{N}\right)(\widehat{n}+n_0)k} \\ X_2[k] &= \sum_{\widehat{n}=0}^{N-1} x[\widehat{n}] e^{-j\left(\frac{2\pi}{N}\right)\widehat{n}k} e^{-j\left(\frac{2\pi}{N}\right)n_0 k} \\ X_2[k] &= e^{-j\left(\frac{2\pi}{N}\right)n_0 k} \sum_{\widehat{n}=0}^{N-1} x[\widehat{n}] e^{-j\left(\frac{2\pi}{N}\right)\widehat{n}k} \\ X_2[k] &= e^{-j\left(\frac{2\pi}{N}\right)n_0 k} X[k] \end{aligned}$$

5.5.4. Desplazamiento en frecuencia

Ahora si hacemos un desplazamiento en frecuencia tenemos que

$$e^{j\left(\frac{2\pi}{N}\right)nk_0} x[n] \overset{\mathcal{F}}{\leftrightarrow} X[k - k_0]$$

Demostración: Escribimos la formula de la TDIF (5.14)

$$x_2[k] = \frac{1}{N} \sum_{k=0}^{N-1} X[k - k_0] e^{j\left(\frac{2\pi}{N}\right)nk}$$

haciendo el cambio de variable $\widehat{k} = k - k_0$ tenemos

$$\begin{aligned} x_2[k] &= \frac{1}{N} \sum_{\widehat{k}=-k_0}^{N-1-k_0} X[\widehat{k}] e^{j\left(\frac{2\pi}{N}\right)n\left(\widehat{k}+k_0\right)} \\ x_2[k] &= \frac{1}{N} \sum_{\widehat{k}=0}^{N-1} X[\widehat{k}] e^{j\left(\frac{2\pi}{N}\right)n\widehat{k}} e^{j\left(\frac{2\pi}{N}\right)nk_0} \\ x_2[k] &= \frac{1}{N} e^{j\left(\frac{2\pi}{N}\right)nk_0} \sum_{\widehat{k}=0}^{N-1} X[\widehat{k}] e^{j\left(\frac{2\pi}{N}\right)n\widehat{k}} \\ x_2[k] &= e^{j\left(\frac{2\pi}{N}\right)nk_0} x[n] \end{aligned}$$

5.5.5. Conjugación

La definición del conjugado de un numero complejo es

$$\begin{aligned} x &= R + jI \\ x^* &= R - jI \end{aligned}$$

Mostrar que:

$$x^*[n] \overset{\mathcal{F}}{\leftrightarrow} X^*[-k]$$

comenzaremos por

$$\begin{aligned} X_2[k] &= \sum_{n=0}^{N-1} x^*[n] e^{-j\left(\frac{2\pi}{N}\right)nk} \\ X_2[k] &= \left(\sum_{n=0}^{N-1} x[n] e^{-j\left(\frac{2\pi}{N}\right)n(-k)} \right)^* \\ X_2[k] &= X^*[-k] \end{aligned}$$

5.5.6. Inversión en Tiempo

Mostrar que

$$x[-n] \overset{\mathcal{F}}{\leftrightarrow} X[-k]$$

Comenzaremos por calcular la transformada de Fourier de $x(-n)$

$$X_2[k] = \sum_{n=0}^{N-1} x[-n] e^{-j\left(\frac{2\pi}{N}\right)nk}$$

hacemos el cambio de variable $m = -n$

$$\begin{aligned} X_2[k] &= \sum_{m=0}^{-N+1} x[m] e^{-j\left(\frac{2\pi}{N}\right)(-m)k} \\ X_2[k] &= \sum_{m=0}^{N-1} x[m] e^{-j\left(\frac{2\pi}{N}\right)m(-k)} \\ X_2[k] &= X[-k] \end{aligned}$$

5.5.7. Escalamiento en tiempo

Vamos a calcular la TDF de una señal discreta escalada x_e un factor a a partir de una señal x con N muestras de la siguiente manera:

$$x_e[n] = \begin{cases} x[n] \text{ Si } n \text{ es múltiplo de } a \\ 0 \text{ de lo contrario} \end{cases}$$

$$\forall n \in \{0, 1, 2, \ldots, M-1\}$$

De acuerdo con lo anterior la señal escalada x_e será de tamaño $M = aN$ pero algunos de estos valores son cero. La transformada de Fourier para este caso es:

$$X_e[k] = \sum_{n=0}^{M-1} x_e[n] e^{-j\left(\frac{2\pi}{M}\right)nk}$$

Para la señal x_e solo tenemos N elementos diferentes de cero, con lo cual podemos escribir:

$$X_e[k] = \sum_{n=0}^{N-1} x[an] e^{-j\left(\frac{2\pi}{aN}\right)(na)k} = \sum_{n=0}^{N-1} x[an] e^{-j\left(\frac{2\pi}{N}\right)nk}$$

haciendo $m = an$ tenemos

$$\begin{aligned} X_e[k] &= \sum_{n=0}^{N-1} x[m] e^{-j\left(\frac{2\pi}{N}\right)\left(\frac{m}{a}\right)k} \\ X_e[k] &= \sum_{m=0}^{a(N-1)} x[m] e^{-j\left(\frac{2\pi}{N}\right)m\left(\frac{k}{a}\right)} \\ &= X\left[\frac{k}{a}\right] \end{aligned}$$

Note que el rango de la TDF, también se escala.

5.5.8. Convolución

El teorema de la convolución afirma que si $\mathcal{F}[x(n)] = X[k]$ y $\mathcal{F}[y(n)] = Y[k]$ entonces

$$x[n] * y[n] \overset{\mathcal{F}}{\leftrightarrow} X[k]Y[k]$$

demostración:

Consideremos que la convolución está $g[n] = x[n] * y[n]$ dada por

$$g[n] = \sum_{m=0}^{N-1} x[m]\, y[n-m]$$

y la transformada de Fourier de $g[n]$ es

$$\begin{aligned} G[k] &= \sum_{n=0}^{N-1} g[n] e^{-j\left(\frac{2\pi}{N}\right)nk} \\ G[k] &= \sum_{n=0}^{N-1} \left(\sum_{m=0}^{N-1} x[m] y[n-m] \right) e^{-j\left(\frac{2\pi}{N}\right)nk} \end{aligned}$$

cambiando el orden de la sumatoria

$$G[k] = \sum_{m=0}^{N-1} x[m] \left(\sum_{n=0}^{N-1} y[n-m] e^{-j\left(\frac{2\pi}{N}\right)nk} \right)$$

haciendo $\widehat{n} = n - m$

$$\begin{aligned} G[k] &= \sum_{m=0}^{N-1} x[m] \left(\sum_{n=0}^{N-1} y[\widehat{n}] e^{-j\left(\frac{2\pi}{N}\right)(\widehat{n}+m)k} \right) \\ G[k] &= \left(\sum_{m=0}^{N-1} x[m] e^{-j\left(\frac{2\pi}{N}\right)mk} \right) \left(\sum_{\widehat{n}=0}^{N-1} y[\widehat{n}] e^{-j\left(\frac{2\pi}{N}\right)\widehat{n}k} \right) \\ G[k] &= X[k]Y[k] \end{aligned}$$

5.5.9. Multiplicación

De manera inversa podemos ver que si $\mathcal{F}[x(n)] = X[k]$ y $\mathcal{F}[y(n)] = Y[k]$ entonces

$$Nx[n]y[n] \overset{\mathcal{F}}{\leftrightarrow} X[k] * Y[k]$$

demostración:

Consideremos que la convolución $G[k] = X[k] * Y[k]$ está dada por

$$G[k] = \sum_{l=0}^{N-1} X[l]Y[k-l]$$

y su transformada Inversa de Fourier (5.14) la podemos calcular como:

$$\begin{aligned} g[n] &= \frac{1}{N}\sum_{k=0}^{N-1}\left(G[k]\right)e^{j\left(\frac{2\pi}{N}\right)nk} \\ g[n] &= \frac{1}{N}\sum_{k=0}^{N-1}\left(\sum_{l=0}^{N-1}X[l]Y[k-l]\right)e^{j\left(\frac{2\pi}{N}\right)nk} \end{aligned}$$

cambiando el orden de la sumatoria

$$g[n] = \frac{1}{N}\sum_{l=0}^{N-1}X[l]\left(\sum_{k=0}^{N-1}Y[k-l]e^{j\left(\frac{2\pi}{N}\right)nk}\right)$$

haciendo $\widehat{k} = k - l$

$$\begin{aligned} g[n] &= \frac{1}{N}\sum_{l=0}^{N-1}X[l]\left(\sum_{\widehat{k}=-l}^{N-1-l}Y[\widehat{k}]e^{j\left(\frac{2\pi}{N}\right)n\left(\widehat{k}+l\right)}\right) \\ g[n] &= N\left(\frac{1}{N}\sum_{l=0}^{N-1}X[l]e^{j\left(\frac{2\pi}{N}\right)nl}\right)\left(\frac{1}{N}\sum_{\widehat{k}=0}^{N-1}Y[\widehat{k}]e^{j\left(\frac{2\pi}{N}\right)n\widehat{k}}\right) \\ g[n] &= Nx[n]y[n] \end{aligned}$$

5.5.10. Diferenciación en Tiempo

Demostrar que

$$x[n] - x[n-1] \overset{\mathcal{F}}{\leftrightarrow} \left(1 - e^{-j\left(\frac{2\pi}{N}\right)k}\right)X[k]$$

tenemos que

$$x[n] \overset{\mathcal{F}}{\leftrightarrow} X[k]$$

y

$$x[n-1] \overset{\mathcal{F}}{\leftrightarrow} e^{-j\left(\frac{2\pi}{N}\right)k}X[k]$$

por superposición demostramos la primera.

5.5.11. Diferenciación en Frecuencia

Demostrar que

$$X[k] - X[k-1] \overset{\mathcal{F}}{\leftrightarrow} \left(1 - e^{j(\frac{2\pi}{N})k}\right) x[n]$$

tenemos que

$$X[k] \overset{\mathcal{F}}{\leftrightarrow} x[n]$$

y

$$X[k-1] \overset{\mathcal{F}}{\leftrightarrow} e^{j(\frac{2\pi}{N})k} x[n]$$

por superposición demostramos la primera.

5.5.12. Propiedades de Simetría de la transformada de Fourier.

Para cualquier secuencia discreta x la parte real la podemos calcular haciendo

$$\mathcal{R}\left[x[n]\right] = \frac{1}{2}\left[x[n] + x^*[n]\right]$$

y la parte imaginaria como

$$\mathcal{I}\left[x[n]\right] = \frac{1}{2}\left[x[n] - x^*[n]\right]$$

Si $x[n]$ es una secuencia real entonces:

$$\frac{1}{2}\left[x[n] - x^*[n]\right] = 0$$

lo que es equivalente a

$$x[n] = x^*[n]$$

Al aplicar la transformada de Fourier tendremos que

$$X[k] = X^*[-k]$$

A partir de esto se observa que la parte real de $X[k]$ es una función par de k y la parte imaginaria de $X[k]$ es una función impar de k. De manera similar, la magnitud de $X[k]$ es una función par y el ángulo de de fase es una función impar. Además

$$\mathcal{E}\left(x[n]\right) \overset{\mathcal{F}}{\leftrightarrow} \mathcal{R}\left(X[k]\right)$$

y

$$\mathcal{O}\left(x[n]\right) \overset{\mathcal{F}}{\leftrightarrow} \mathcal{I}\left(X[k]\right)$$

demostración:

En el caso de una señal real y par se cumple que $x[n] = x[-n]$ y que $x[n] = x^*[n]$ por lo tanto en el dominio de Fourier

$$X[k] = X[-k] = \mathcal{R}\left(X[-k]\right) + j\mathcal{I}\left(X[-k]\right)$$
$$X[k] = X^*[-k] = \mathcal{R}\left(X[-k]\right) - j\mathcal{I}\left(X[-k]\right)$$

la única posibilidad de que esto sea cierto es que la parte imaginaria sea cero. Así la transformada de Fourier se transforma en

$$X[k] = \sum_{n=0}^{N} x[n] \cos\left(\frac{2\pi}{N} nk\right)$$

a esta ecuación es común que se le conozca como la transformada coseno.

En el caso de una señal real e impar, se cumple que $x[n] = -x[-n]$ $x[n] = x^*[n]$, entonces en el dominio de Fourier

$$X[k] = -X[-k] = -\mathcal{R}\left(X[-k]\right) - j\mathcal{I}\left(X[-k]\right)$$
$$X[k] = X^*[-k] = \mathcal{R}\left(X[-k]\right) - j\mathcal{I}\left(X[-k]\right)$$

la única posibilidad de que esto ocurra es que la parte real sea cero. La transformada de Fourier en este caso queda como

$$X[k] = j \sum_{n=0}^{N} x[n] sen\left(\frac{2\pi}{N} nk\right)$$

5.5.13. Relación de Parseval.

Si x y X son la señal y su transformada discreta de Fourier respectivamente, entonces tenemos que:

$$N \sum_{n=0}^{N-1} |x[n]|^2 \overset{\mathcal{F}}{\leftrightarrow} \sum_{k=0}^{N-1} |X[k]|^2$$

podemos representar como

$$N\sum_{n=0}^{N-1} x[n]x^*[n] = \sum_{k=0}^{N-1} X[k]X[k]^*$$

Tomando únicamente el segundo termino de la ecuación anterior

$$E = \sum_{k=0}^{N-1} X[k]X[k]^*$$

$$E = \sum_{k=0}^{N-1} X[k]\left[\sum_{n=0}^{N-1} x[n]e^{-j\left(\frac{2\pi}{N}\right)nk}\right]^*$$
$$E = \sum_{k=0}^{N-1} X[k]\sum_{n=0}^{N-1} x^*[n]e^{j\left(\frac{2\pi}{N}\right)nk}$$

cambiando el orden en que se hacen las sumatorias

$$= N\sum_{n=0}^{N-1} x^*[n]\left[\frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\left(\frac{2\pi}{N}\right)nk}\right]$$
$$= N\sum_{n=0}^{N-1} x^*[n]x[n]$$
$$= N\sum_{n=0}^{N-1} |x[n]|^2$$

5.5.14. Resumen de Propiedades

Dado lo anterior podemos resumir en la siguiente tabla las propiedades de la transformada discreta de Fourier.

Propiedad	Señal en el tiempo	Señal en la frecuencia	Referencia
Periodicidad	$x[n]$	$X[k]$ periodo N	5.5.1
Linealidad	$ax_1[n] + bx_2[n]$	$aX_1[k] + bX_2[k]$	5.5.2
Desp. en tiempo	$x[n - n_0]$	$e^{-j\left(\frac{2\pi}{N}\right)n_0 k}X[k]$	5.5.3
Desp. en frecuencia	$e^{j\left(\frac{2\pi}{N}\right)nk_0}x[n]$	$X[k - k_0]$	5.5.4
Conjugación	$x^*[n]$	$X^*[-k]$	5.5.5
Inversión en tiempo	$x[-n]$	$X[-k]$	5.5.6
Escalamiento en el Tiempo	$x[an]$	$X[k/a]$	5.5.7
Convolución	$x[n] * y[n]$	$X[k]Y[k]$	5.5.8
Multiplicación	$Nx[n]y[n]$	$X[k] * Y[k]$	5.5.9
Dif. en tiempo	$x[n] - x[n-1]$	$\left(1 - e^{-j\left(\frac{2\pi}{N}\right)k}\right)X[k]$	5.5.10
Dif. en Frecuencia	$\left(1 - e^{j\left(\frac{2\pi}{N}\right)n}\right)x[n]$	$X[k] - X[k-1]$	5.5.11
Simetría conjugada	$x[n]$ real	$X[k] = X^*[k]$	5.5.12
Simetría	$x[n]$ real y par	$X[k]$ real y par	5.5.12
Simetría	$x[n]$ real e impar	$X[k]$ imaginaria e impar	5.5.12
Parseval	$N\sum_{n=0}^{N-1}\lvert x[n]\rvert^2$	$\sum_{k=0}^{N-1}\lvert X[k]\rvert^2$	5.5.13

5.6. Transformada de Discreta de Fourier en Dos dimensiones.

En dos dimensiones las exponenciales complejas (5.13) las expresaremos como

$$\begin{aligned} \phi_{k,l}(n,m) &= e^{j((2\pi nk)/N + (2\pi ml)/M)} \\ \phi_{k,l}(n,m) &= \phi_k(n)\phi_l(m) \end{aligned} \tag{5.18}$$

Definidas las exponenciales complejas (5.18) y de acuerdo con (5.17) podemos escribir la Transformada Discreta de Fourier en 2D (TDF2), como

$$X[k,l] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x[n,m]e^{-j\left(\frac{2k\pi}{N}n + \frac{2l\pi}{M}m\right)} \tag{5.19}$$

reorganizando términos tenemos:

$$X[k,l] = \sum_{m=0}^{M-1}\left[\sum_{n=0}^{N-1} x[n,m]e^{-j\left(\frac{2\pi}{N}\right)nk}\right]e^{-j\left(\frac{2\pi}{M}\right)ml}$$

La formulación de la Transformada Discreta Inversa de Fourier en dos dimensiones (TDIF2) esta dada por (5.20):

$$x[n,m] = \frac{1}{NM}\sum_{k=0}^{N-1}\sum_{l=0}^{M-1} X[k,l]e^{j\left(\frac{2k\pi}{N}n+\frac{2l\pi}{M}m\right)} \tag{5.20}$$

reorganizando términos

$$x[n,m] = \frac{1}{N}\sum_{k=0}^{N-1}\left[\frac{1}{M}\sum_{l=0}^{M-1} X[k,l]e^{j\left(\frac{2l\pi}{M}m\right)}\right]e^{j\left(\frac{2k\pi}{N}n\right)}$$

En la siguiente sección se muestra la implementación de la TDF y TDIF en dos dimensiones.

5.6.1. Ejemplo

Calcular la TDF2 para la señal bidimensional

$$x = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

De acuerdo con las formulaciones, podemos hacer la TDF en una dimensión para cada uno de los renglones para luego proceder por columna. La matriz de exponenciales complejas por columnas es

$$\Phi_r^* = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

y por renglones es

$$\Phi_c^* = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Comenzamos haciendo la Transformada de Fourier por renglones.

Así para el primer renglón calculamos

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 10+j0 \\ -2+j2 \\ -2+j0 \\ -2-j2 \end{bmatrix}$$

para el segundo renglón calculamos

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 10+j0 \\ 2-j2 \\ 2+j0 \\ 2+j2 \end{bmatrix}$$

Los dos renglones quedan

$$\begin{bmatrix} 10+j0 & -2+j2 & -2+j0 & -2-j2 \\ 10+j0 & 2-j2 & 2+j0 & 2+j2 \end{bmatrix}$$

Por columnas hacemos

$$\begin{aligned} X &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 10+j0 & -2+j2 & -2+j0 & -2-j2 \\ 10+j0 & 2-j2 & 2+j0 & 2+j2 \end{bmatrix} \\ X &= \begin{bmatrix} 20+j0 & 0+j0 & 0+j0 & 0+j0 \\ 0+j0 & -4+j4 & -4+j0 & -4-j4 \end{bmatrix} \end{aligned}$$

5.7. Implementación de la Transformada Discreta de Fourier en 2D

Analizando las expresiones reorganizadas de la TDF2 y TDIF2 podemos ver que son equivalentes a realizar la Transformada en una dirección del arreglo bidimensional y después realizarlo en la otra dirección. Adicionalmente en las formulaciones dadas por (5.19) y (5.20), podemos ver que la diferencia entre ambas es un signo y una división, por lo tanto la implementación genérica de ambas es:

```
private static void DFT_2D(double entrada_r[][], double entrada_i[][],
        double salida_r[][], double salida_i[][], int signo){

    int N = entrada_r.length, M = entrada_r[0].length, n, k;
    double Cr[][] = new double [M][M];
    double Ci[][] = new double [M][M];
```

```
double Rr[][] = new double [N][N];
double Ri[][] = new double [N][N];
double w , suma_r, suma_i, factor =1.0;
int i, j;
int max = N >M ? N : M;

double b_r[] = new double[max];
double b_i[] = new double[max];


w = 2.0*Math.PI/(double) N;

for(n=0; n<N; n++){
    for(k=n; k<N; k++) {
        Rr[n][k] = Math.cos(w*n*k);
        Rr[k][n] = Rr[n][k];
        Ri[n][k] = signo*Math.sin(w*n*k);
        Ri[k][n] = Ri[n][k];
    }
}

w = 2.0*Math.PI/(double) M;

for(n=0; n<M; n++){
    for(k=n; k<M; k++) {
        Cr[n][k] = Math.cos(w*n*k);
        Cr[k][n] = Cr[n][k];
        Ci[n][k] = signo*Math.sin(w*n*k);
        Ci[k][n] = Ci[n][k];
    }
}

factor = signo == 1 ? 1.0/(double) N : 1.0 ;

for(i=0; i<N; i++) {
    for(k=0; k<M; k++) {
        suma_r = 0;
        suma_i = 0;

        for(n=0; n<M; n++){
            suma_r += (Cr[k][n]*entrada_r[i][n] - Ci[k][n]*entrada_i[i][n]);
```

```
                suma_i += (Cr[k][n]*entrada_i[i][n] + Ci[k][n]*entrada_r[i][n]);
            }
            salida_r[i][k] = suma_r*factor;
            salida_i[i][k] = suma_i*factor;
        }
    }

    factor = signo == 1 ? 1.0/(double) M : 1.0 ;

    for(j=0; j<M; j++) {

        for(k=0; k<N; k++) {
            suma_r = 0;
            suma_i = 0;

            for(n=0; n<N; n++){
                suma_r += (Rr[k][n]*salida_r[n][j] - Ri[k][n]*salida_i[n][j]);
                suma_i += (Rr[k][n]*salida_i[n][j] + Ri[k][n]*salida_r[n][j]);
            }
            b_r[k] = suma_r;
            b_i[k] = suma_i;
        }

        for(i=0; i<N; i++){
            salida_r[i][j] = b_r[i]*factor;
            salida_i[i][j] = b_i[i]*factor;
        }
    }
}
```

Para el calculo de la TDF2 y de ls TDIF haremos uso de las funciones

```
public static void Fourier(double entrada_r[][], double entrada_i[][],
        double salida_r[][], double salida_i[][]) {
    DFT_2D(entrada_r, entrada_i, salida_r, salida_i, -1);
}

public static void Inversa_Fourier(double entrada_r[][], double entrada_i[][],
        double salida_r[][], double salida_i[][]) {
    DFT_2D(entrada_r, entrada_i, salida_r, salida_i, 1);
}
```

5.7.1. Ejemplo

Transformada de Fourier de una señal bidimensional

Dada la señal $x\left[n,m\right]=\cos\left(\frac{2*\pi}{N}\left(k_0n+l_0m\right)\right)$ determinar su transformada de Fourier.

Esta señal puede ser representada como la suma de dos exponenciales complejas

$$x[n,m]=\cos\left(\frac{2*\pi}{N}\left(k_0n+l_0m\right)\right)$$
$$=\frac{1}{2}e^{j\left(\frac{2\pi}{N}\right)(k_0n+l_0m)}+\frac{1}{2}e^{-j\left(\frac{2\pi}{N}\right)(k_0n+l_0m)}$$

por lo tanto

$$X[k,l]=\frac{N}{2}\delta\left(k-k_0,l-l_0\right)+\frac{N}{2}\delta\left(k+k_0,l+l_0\right)$$

En la figura 5.8 podemos ver la señal bidimensional correspondiente $x[n,m]=\cos\left[\frac{2*\pi}{N}\left(10n+20m\right)\right]$ y su transformada de Fourier. Note que en la imagen de la transformada de Fourier aparecen dos picos en las coordenadas $[10,20]$ y $[118,108]$, lo cual, corresponde con la deducción anterior.

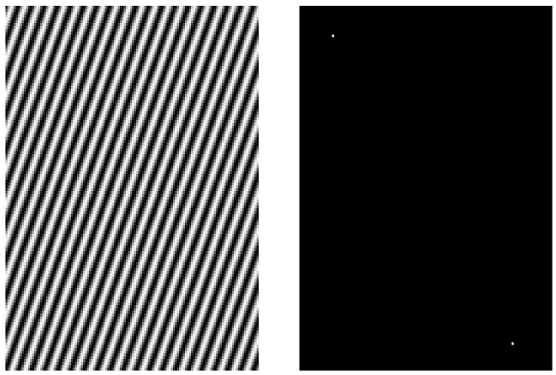


Figura 5.8: Señal coseno en dos dimensiones

5.7.2. Ejemplo

El primer renglón de una imagen es creado utilizando una distribución normal $N(0,1)$ (ver 5.10) y los renglones subsiguientes se calculan de acuerdo con la siguiente sucesión

$f[n,m] = f\,[n-1, m-1]$ tal que se produce un patrón como el mostrado en la figura 5.9 (a la izquierda). Explique a qué se debe que la magnitud de la transformada de Fourier esta dominado por una linea recta a 45 grados. La transformada de Fourier del ruido gaussiano la podemos ver a la derecha de la figura 5.9 .

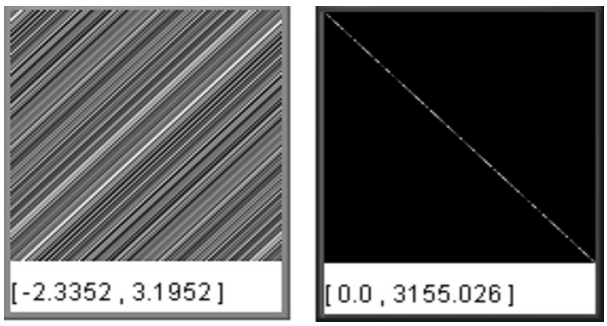


Figura 5.9: A la izquierda Patron regular creado con una señal uni–dimensional trasladada (ruido Gaussiano) y a la derecha Magnitud de su transformada de Fourier

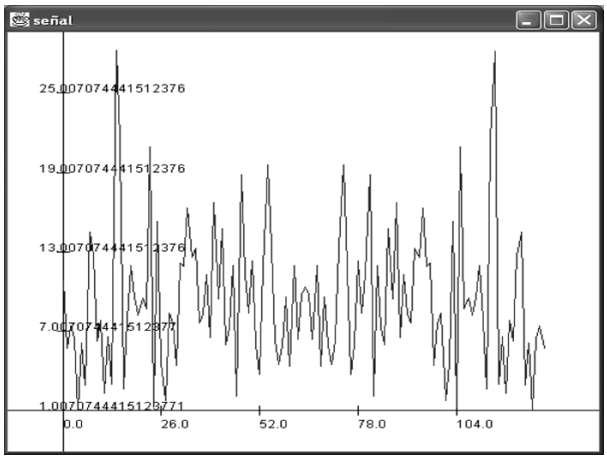


Figura 5.10: Valores aleatorios calculado con una distribución Normal con media cero y varianza unitaria

Demostración:

El primer renglón de imagen fue generado utilizando una distribución normal, y los renglones subsecuentes utilizando la recursión

$$f[n,m] = f\,[n-1, m-1]$$

la transformada de Fourier para $f[n,m]$ es $F[k,l]$ si aplicamos la propiedad de translación

de la TF, podemos ver:

$$f[n,m] \overset{\mathcal{F}}{\leftrightarrow} F[k,l]$$
$$f[n-n_0,m-m_0] \overset{\mathcal{F}}{\leftrightarrow} e^{-j\left(\frac{2\pi}{NM}\right)(n_0k+m_0l)}F[k,l]$$

lo cual nos da como resultado la transformada de Fourier $F(x)$, esta multiplicada por una exponencial compleja $e^{-j\left(\frac{2\pi}{NM}\right)(n_0k+m_0l)}$ cuyo máximo se localiza sobre la recta $n_0k+m_0l = 0$, dado que $n_0 = m_0 = 1$, tendremos que la pendiente de

esta linea recta es 45 grados.

5.8. Convolución utilizando TF.

En una imagen tenemos que la referencia se encuentra en la esquina superior izquierda. Cuando estamos utilizando la transformada de Fourier debemos recordar que está, considera que las señales tienen periodos N en la dirección de x y M en la dirección de y. Así, una imagen estará representada virtualmente, por un conjunto de copias infinitas es un espacio infinito.

Para realizar una translación de la imagen, una manera es convolucionar la imagen con la función delta de Dirac $\delta(n-n_0,m-m_0)$ donde n_0 y m_0 es el desplazamiento, hay que recordar que la convolución es Lineal e invariante a la translación, por lo que, la convolución de dos señales no se vera afectada cuando hacemos la translación del kernel.

Cuando aplicamos el teorema de la convolución

$$x[n,m] * y[n,m] \overset{\mathcal{F}}{\leftrightarrow} X[k,l]Y[K,l]$$

debemos tomar en cuenta que si el kernel se desplaza, la transformada de Fourier del kernel estará multiplicada por una exponencial compleja

$$x[n-n_0,y-m_0] \overset{\mathcal{F}}{\leftrightarrow} X[k,l]e^{-j\frac{2\pi}{NM}(n_0k+m_0l)}$$
$$x[n,m] * \delta(n-n_0,m-m_0) \overset{\mathcal{F}}{\leftrightarrow} X[k,l]e^{-j\frac{2\pi}{NM}(n_0k+m_0l)}$$

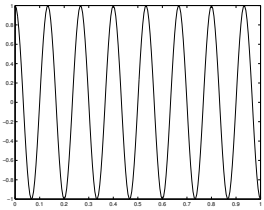
Cuando aplicamos el teorema de la convolución tendremos

$$x[n,m] * (y[n,m] * \delta(n-n_0,m-m_0)) \overset{\mathcal{F}}{\leftrightarrow} X[k,l]\left[Y[k,l]e^{-j\frac{2\pi}{NM}(n_0k+m_0l)}\right]$$
$$(x[n,m] * \delta(n-n_0,m-m_0)) * y[n,m] \overset{\mathcal{F}}{\leftrightarrow} \left(X[k,l]e^{-j\frac{2\pi}{NM}(n_0k+m_0l)}\right)Y[k,l]$$
$$x[n-n_0,m-m_0] * y[n,m] \overset{\mathcal{F}}{\leftrightarrow} \left(X[k,l]e^{-j\frac{2\pi}{NM}[n_0k+m_0l]}\right)Y[k,l]$$

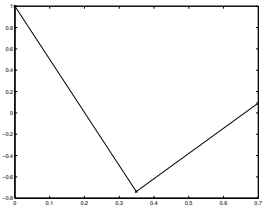
lo cual significa que $x[n-n_0,m-m_0]$, estará desplazada a las coordenadas $[n_0,m_0]$

5.9. Teorema del Muestreo para señales discretas

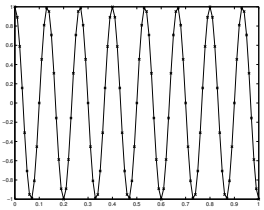
Consideramos el caso de una señal continua en el tiempo dada por $x(t) = cos(15\pi t)$. Es claro que la señal tiene una frecuencia angular $w = 15\pi$, una frecuencia $f = 7.5$ y un periodo $T = \frac{1}{7.5}$. Esto significa que si graficamos la señal tendremos siete ciclos y medio por segundo y la gráfica para esta lucirá como la figura 5.11(a). Si esta misma función la submuestreamos con incrementos de $\frac{\pi}{9}$ lucirá como la fig 5.11(b) y esta misma a intervalos de 0.01 lucirá como 5.11(c). ¿Cual es el valor de incremento al que debo discretizar mi señal sin perder información?.



(a) ***Señal original***



(b) ***Señal muestreada a*** $\frac{\pi}{9}$



(c) ***Señal muestreada a 0.01***

Figura 5.11: Señal $cos(15\pi t)$ con diferentes muestreos

Podemos definir una señal muestreada $x_m[n]$ con un periodo T_m como:

$$x_m[n] = x(nT_m) \tag{5.21}$$

Aplicando (5.21) a la señal $x(t) = cos(15\pi t)$ tenemos una señal muestreada

$$x_m[n] = cos((15\pi T_m)n)$$

note que la frecuencia de la señal original se ve afectada por el periodo de muestreo. La transformada de Discreta de Fourier de una función coseno con periodo de muestreo T_m es:

$$cos\left(\frac{2\pi k_0 T_m}{N} n\right) \overset{\mathcal{F}}{\leftrightarrow} \frac{N}{2}[\delta(k - T_m k_0) + \delta(k + T_m k_0)]$$

En la figura 5.12, se muestra la TDF de la señal con diferentes periodos de muestreo $T_m = 2/150$, $T_m = 4/150$, $T_m = 6/150$, $T_m = 8/150$, $T_m = 10/150$ y $T_m = 12/150$. Note en esta figura que la los impulso se encuentran en diferentes posiciones de k_0, debido a que alteramos la frecuencia original de la señal al multiplicarla por el periodo de muestreo T_m. ¿Ahora que relación tienen?.

Recordemos que la TDF solamente tiene valores de frecuencia que son múltiplos de $2\pi/N$, a acuerdo con esto, todas las frecuencias ω_0 tendrán una representación dada por:

$$\omega_0 T_m = \frac{2\pi}{N} k_0$$

y el múltiplo k_0 de la frecuencia $2\pi/N$ lo podemos encontrar como (5.9)

$$k_0 = \frac{N\omega T_m}{2\pi} \tag{5.22}$$

De acuerdo con y considerando que la señal tiene $N = 100$ muestras, en la siguiente tabla, se presentan los valores de k_0 para diferentes valores de muestreo T_m

Cuadro 5.4: Multiplos de la frecuencia $2\pi/100$ para la señal $cos(15\pi T_m n)$

T_m	k_0
2/150	10
4/150	20
6/150	30
8/150	40
10/150	50
12/150	60

Note que cuando tenemos un periodo de muestreo $T_m = 12/150$ el múltiplo de frecuencia $k_0 = 60$. Si solamente tenemos múltiplos que van de $-N/2$ hasta $N/2$, tenemos el problema que no podemos representar la señal sin que se traslapen sus transformada.

5.9.1. Teorema del muestreo de Nyquist

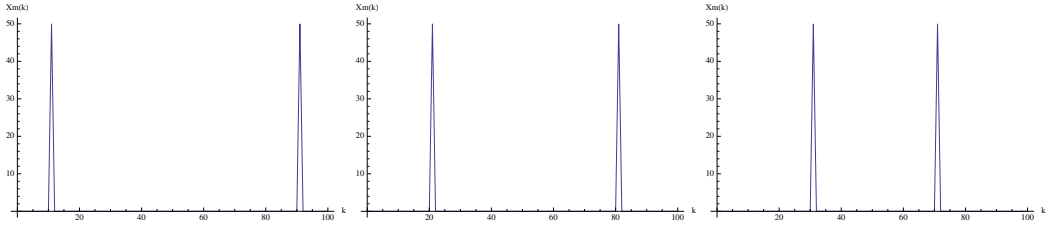
De acuerdo con la antes expuesto tenemos una cuota máxima:

$$k_0 < \frac{N}{2}$$

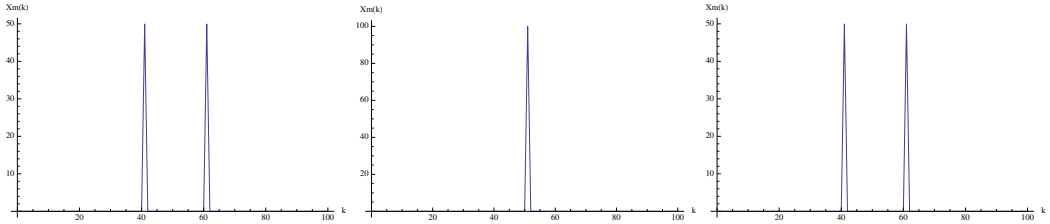
donde k_0 es el múltiplo de nuestra frecuencia que queremos representar y $N/2$ es el múltiplo de frecuencia máxima que podemos representar.

En virtud de esto podemos calcular para una frecuencia ω_0, muestreada con periodo T_m el valor de k_0 utilizando (5.9) y sustituyendo en la ecuación anterior

$$k_0 = \frac{N\omega_0 T_m}{2\pi} < \frac{N}{2}$$



(a) $T_m = 2/150$ **(b)** $T_m = 4/150$ **(c)** $T_m = 6/150$



(d) $T_m = 8/150$ **(e)** $T_m = 10/150$ **(f)** $T_m = 12/150$

Figura 5.12: TDF de la señal $cos(15\pi t)$ con diferentes valores de T_m

$$\omega_0 < \frac{\pi}{T_m}$$
$$2\omega_0 < 2\frac{\pi}{T_m}$$

de donde se desprende la formulación del Teorema de Nyquist

$$2\omega_0 < \omega_m \tag{5.23}$$

donde ω_m es la frecuencia de muestreo y esta definida por $\frac{2\pi}{T_m}$. Esta ecuación la podemos poner también como

$$2\frac{2\pi}{T_0} < \frac{2\pi}{T_m}$$
$$2T_m < T_0$$

Con esto podemos concluir que para una señal con ancho de banda ω_0 tenemos que muestrear, si queremos recuperar la señal original, al menos a una frecuencia de dos veces el ancho de banda o que el periodo de muestreo tiene que ser al meno dos veces menor que el periodo de la señal.

En la figura 5.13 se muestra una señal coseno, con un periodo $T_0 = 1333.33$, la cual fue muestreada con un periodo mayor que $T_m = 28$. Para este caso podemos notar que la

señal muestreada para nada luce como una señal de la frecuencia original y el Teorema de Nyquist no se cumple ya que $2T_0 > Tm$

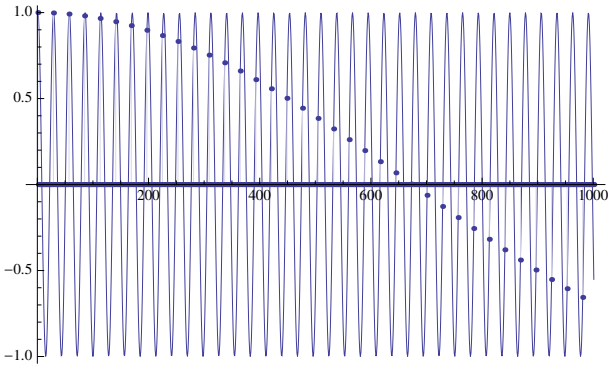


Figura 5.13: **Ejemplo de una señal con bajo muestro**

5.9.2. Transformada Discreta de Fourier de un tren de Pulso

La transformada de Fourier de una función $\delta(n - n_0)$ es una exponencial compleja dada por

$$X[k] = e^{-\left(\frac{2\pi}{N} n_0\right)k}$$

La transformada de Furier de un tren de pulsos $p[n]$

$$p[n] = \sum_{m=0}^{\frac{N}{T}-1} \delta(n - mT)$$

por linealidad la podemos calcular utilizando

$$P[k] = \sum_{m=0}^{\frac{N}{T}-1} e^{-\left(\frac{2\pi}{N} mT\right)k}$$

Utilizado la (5.16) podemos ver que

$$X[k] = \frac{1 - e^{-\left(\frac{2\pi}{N}\frac{N}{T}T\right)k}}{1 - e^{-\left(\frac{2\pi}{N}T\right)k}} = 0$$

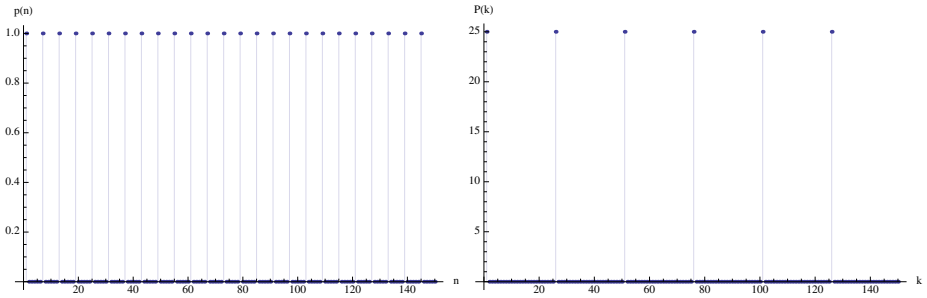
Sin embargo cuando m es múltiplo de N/T tenemos que $e^{-\left(\frac{2\pi}{N}\frac{N}{T}lT\right)k} = 1$ para $l = 0, 1, 2, \ldots$

$$P[k] = \begin{cases} 1 & m = 0 \\ 1 & m = \frac{N}{T} \\ 1 & m = \frac{2N}{T} \\ 1 & m = \frac{3N}{T} \\ \vdots & \vdots \end{cases}$$

de manera compacta lo podemos escribir como

$$P[k] = \sum_{l=0}^{\infty} \delta\left(k - \frac{N}{T}l\right)$$

Podemos concluir que la TDF de un tren de pulso con periodo T es un tren de pulso con un periodo N/T. En la figura 5.14(a) se muestra un tren de pulsos con periodo $T = 6$ y su TDF en 5.14(b). Note que el periodo de la TDF es $N/T = 25$, para una señal con $N = 150$



(a) ***Tren de pulsos con periodo 6*** **(b)** ***Transformada de Fourier***

Figura 5.14: Tren de Pulsos y su transformada

Señal muestrada con un tren de pulsos

Una forma de muestrear una señal es multiplicando la señal $x[n]$ por un tren de pulso $p[n]$ cuya TDF esta dada por

$$Nx[n]p[n] \overset{\mathcal{F}}{\leftrightarrow} X[k] * P[k]$$

En la figura 5.15(a) se muestra una Gaussiana $g(n;\mu,\sigma)$ con media $\mu = 0$ y varianza $\sigma = 10$, la cual se muestrea con un tren de pulsos $p(n;T)$, con periodo $T = 6$, como el que se muestra en la figura 5.15(c). La transformada de furier $G[k]$ de la Gaussiana se muestra en la figura 5.15(b) y en la figura 5.15(d) la TDF del tren de pulsos. Finalmente en la figura 5.15(e) se muestra la TDF de la función $g(n;0,10)$ multiplicada por el tren de pulsos $p(n;6)$

Para el ejemplo de la figura 5.15(e) podemos notar que tenemos una cuota superior que no debemos superar. Si definimos el ancho de banda k_b como la mitad del tamaño de la Gaussiana, tenemos que para que la Gussiana no se mezcle con los datos de otra Gaussiana, se debe cumplir:

$$2k_b < \frac{N}{T_m}$$

si multiplicamos ambos lados por $2\pi/N$

$$2\left(\frac{2\pi}{N}\right)k_b < \left(\frac{2\pi}{N}\right)\frac{N}{T_m}$$

tenemos

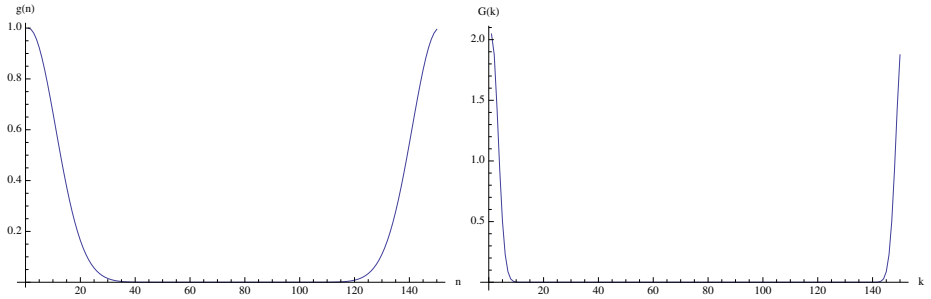
$$2\left(\frac{2\pi}{N}\right)k_b < \frac{2\pi}{T_m}$$

$$2\omega_b < \omega_m$$

que coincide con el resultado de la ecuación (5.23) del Teorema de Nyquist.

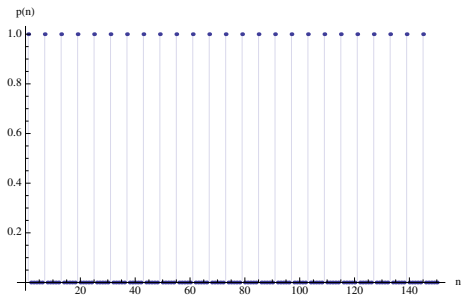
5.10. Teorema del Muestreo señales continuas

En esta sección se presenta todo el análisis con señales continuas a diferencia de la sección anterior donde se presento con señales discretas.

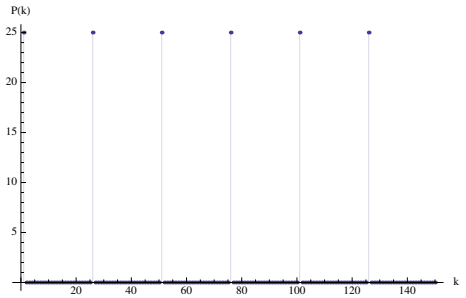


(a) $g(n;0,10)$

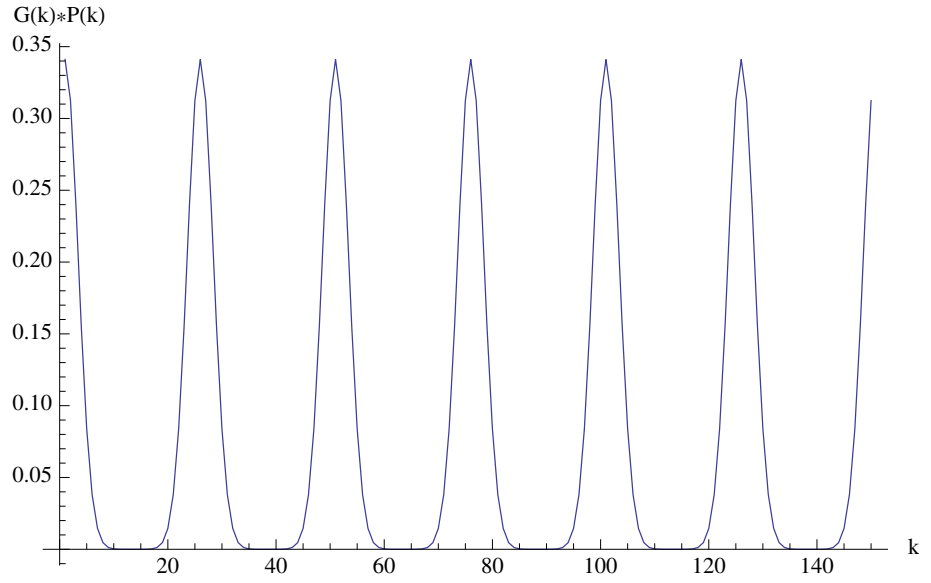
(b) ***TDF***$(g(n;0,10))$



(c) $p(n;6)$ ***con periodo 6***



(d) ***TDF***(p)



(e) ***TDF***$(g(n;0,10)p(n;6))$

5.10.1. Transformada de Fourier de una señal periódica

Supongamos que una señal con periodo T_0 (con $\omega_0 = \frac{2\pi}{T_0}$), pudiera representarse como la serie

$$x(t) = \sum_{k=-\infty}^{k=\infty} X[k]e^{j\omega_0 kt} \tag{5.24}$$

multiplicando ambos miembros de la ecuación por $e^{-jn\omega_0 t}$ e integrando en el intervalo

$$\int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} x(t)e^{-jn\omega_0 t}dt = \int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} \sum_{k=-\infty}^{k=\infty} X[k]e^{j\omega_0 kt}e^{-jn\omega_0 t}dt$$

reagrupando los términos tenemos

$$\int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} x(t)e^{-jn\omega_0 t}dt = \sum_{k=-\infty}^{k=\infty} X[k]\left[\int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} e^{j\omega_0(k-n)t}\right]dt$$

La evaluación de la integral en los corchetes es, considerando que $k \neq n$:

$$\begin{aligned}
\int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} e^{j\omega_0(k-n)t}dt &= \frac{1}{j(k-n)\omega_0}e^{j\omega_0(k-n)t}\Big|_{-\frac{T_0}{2}}^{\frac{T_0}{2}} \\
&= \frac{1}{j(k-n)\omega_0}\left[e^{j(k-n)\pi} - e^{-j(k-n)\pi}\right] \\
&= \frac{1}{j(k-n)\omega_0}\left[-1+1\right] \\
&= 0
\end{aligned}$$

en el caso de que $k = n$, la integral se transforma en

$$\int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} dt = T_0$$

Por lo tanto la transformada de Fourier de una señal periódica ecuación (5.24) es

$$X[n] = \frac{1}{T_0}\int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} x(t)e^{-jn\omega_0 t}dt \tag{5.25}$$

5.10.2. Transformada de Furier de un Tren de Pulsos

Un tren de pulsos lo podemos definir por

$$\begin{aligned}
p(t) = \cdots + \delta(t+2T) + \delta(t+T) + \delta(t) + \\
\delta(t-T) + \delta(t+2T) + \cdots
\end{aligned}$$

la cual podemos representar en forma compacta como

$$p(t) = \sum_{k=-\infty}^{\infty} \delta\left(t - kT\right)$$

donde $T = \frac{2\pi}{\omega_s}$. Note que esta ecuación es una función periódica y su transformada puede ser calculada utilizando el resultado de la ecuación (5.25)

En este caso hablamos de una transformada de Fourier continua por lo cual haremos la transformada de Fourier utilizando la definición continua

$$P(\omega) = \int_{-\infty}^{+\infty} p(t) e^{-j\omega t} dt$$

sustituyendo tenemos

$$P(\omega) = \sum_{k=-\infty}^{\infty} \int_{-\infty}^{+\infty} \delta\left(t - kT\right) e^{-j\omega t} dt \tag{5.26}$$

Aplicando el resultado de la ecuación (5.25) tenemos

$$P(\omega) = \sum_{k=-\infty}^{\infty} \frac{1}{T_0} \int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} \delta\left(t - kT\right) e^{-j\omega t} dt$$

de donde se desprende que si $k = 0$

$$\frac{1}{T_0} \int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} \delta\left(t - kT\right) e^{-j\omega t} dt = \frac{1}{T} = \frac{1}{T} \delta(\omega - 0\omega_s)$$

entonces da como resultado un tren de pulsos equiespaciados en la frecuencia con $\omega_s = \frac{2\pi}{T}$.

$$P(\omega) = \sum_{k=-\infty}^{\infty} \delta\left(\omega - k\omega_s\right)$$

5.10.3. Señal Muestreada en el Tiempo

Una señal muestreada se calcula como

$$x_p(t) = x(t)p(t)$$

Esto significa que tenemos que hacer la convolución de la TDF de la señal $X[k]$ con un tren de pulsos $P[k]$. En la figura se muestra una función gaussiana

donde $p(t)$ es un tren de pulsos. Aplicando el teorema de la convolución tenemos que la transformada de Fourier de esta señal es

$$\begin{aligned} X_p(\omega) &= X(\omega) * P(\omega) \\ X_p(\omega) &= X(\omega) * \sum_{k=-\infty}^{\infty} \delta\left(\omega - k\omega_m\right) \\ &= \sum_{k=-\infty}^{\infty} X(\omega - k\omega_m) \end{aligned}$$

lo cual significa que tendremos copias de nuestra señal $X(\omega)$ repetidas cada $\frac{2\pi}{T}$. ¿Pero que ocurre si el ancho de banda de la señal $X(\omega)$ sobre pasa el valor $\frac{2\pi}{T}$?.

Llamemos ω_m la frecuencia de muestreo la cual esta dada como

$$\omega_m = \frac{2\pi}{T_m}$$

y ω_b el ancho de banda de la señal $x(t)$, podemos ver que si

$$\omega_m > 2\omega_b$$

podemos reconstruir exactamente la señal continua $x\left(t\right)$ a partir de la señal muestreada $x_p\left(t\right)$. La frecuencia de muestreo ω_m también se conoce como la frecuencia de Nyquist.

Podemos notar que

$$x_p[n] = x(nT) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X\left(\omega\right) e^{j\omega nT} d\omega$$

dado que la señal en el dominio de la frecuencia es periódica podemos considerar que

$$x_p[n] = \frac{1}{2\pi} \sum_{r=-\infty}^{\infty} \int_{(2r-1)\pi/T}^{(2r+1)\pi/T} X(\omega) e^{j\omega nT} d\omega$$

y cada término de la suma puede ser reducido a una integral sobre el rango $\frac{\pi}{T}$ a $\frac{\pi}{T}$ haciendo el cambio de variable $\omega = \Omega + \frac{2\pi r}{T}$

$$x_p[n] = \frac{1}{2\pi} \sum_{r=-\infty}^{\infty} \int_{-\pi/T}^{\pi/T} X\left(\Omega + \frac{2\pi r}{T}\right) e^{j\Omega nT} e^{j2\pi rn} d\Omega$$

Si intercambiamos el orden de integración y considerando que $e^{j2\pi rn} = 1$ para cualquier valor de n y r, tenemos que:

$$x_p[n] = \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} \left[\sum_{r=-\infty}^{\infty} X\left(\Omega + \frac{2\pi r}{T}\right) e^{j\Omega nT} \right] d\Omega$$

haciendo un cambio de variable $\Omega = \frac{\omega}{T}$

$$x_p[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[\frac{1}{T} \sum_{r=-\infty}^{\infty} X\left(\frac{\omega}{T} + \frac{2\pi r}{T}\right) e^{j\omega n} \right] d\omega$$

de esta ecuación tenemos que

$$X_p(\omega) = \frac{1}{T} \sum_{r=-\infty}^{\infty} X\left(\frac{\omega}{T} + \frac{2\pi r}{T}\right)$$

o si hacemos $\widehat{\omega} = \frac{\omega}{T}$ tenemos

$$X_p(\widehat{\omega}T) = \frac{1}{T} \sum_{r=-\infty}^{\infty} X\left(\widehat{\omega} + \frac{2\pi r}{T}\right)$$

5.10.4. Integración de la señal continua.

Ahora que hacemos para recuperar la señal continua a partir de la señal muestreada. Puesto que la señal $X_p(\omega)$ solo esta definida en el rango $\left[-\frac{\pi}{T}, \frac{\pi}{T}\right]$, calcularemos la antitrasformada en estos límites, además $X_p(\omega T) = \frac{1}{T} X(\omega)$

$$x(t) = \frac{1}{2\pi}\int_{-\frac{\pi}{T}}^{+\frac{\pi}{T}} TX_p(\omega T)\, e^{j\omega t} d\omega$$

$$= \frac{T}{2\pi}\int_{-\frac{\pi}{T}}^{+\frac{\pi}{T}} \left[\sum_{n=-\infty}^{\infty} x(nT)\, e^{-j\omega nT}\right] e^{j\omega t} d\omega$$

$$= \sum_{n=-\infty}^{\infty} x(nT) \left[\frac{T}{2\pi}\int_{-\frac{\pi}{T}}^{+\frac{\pi}{T}} e^{j\omega(t-nT)} d\omega\right]$$

$$= \sum_{n=-\infty}^{\infty} x(nT)\, \frac{sen\left(\frac{\pi}{T}(t-nT)\right)}{\left(\frac{\pi}{T}\right)(t-nT)}$$

El código para realizar esta integración es:

```
public static double Interpola(double x[], double t, double T)
{
  int n, N = x.length, i;
  double val =0;

  for(i=-3*N; i<3*N; i++)
  {
    n = (i % N);
    if(n<0) n = N + n;
    val += x[n] * Sinc( (t - (double) i * T) * Math.PI / T);
  }
  return val;
}
```

5.10.5. Ejemplos

Ejemplo 1

Aquella frecuencia que de acuerdo con el teorema de muestreo, debe ser excedida por la frecuencia de muestreo se llama razón de Nyquist. Determine la razón de Nyquist para las siguientes señales

a) $x(t) = 1 + \cos(2000\pi t) + sen(4000\pi t)$.

La transformada de Fourier para esta señal es

$$x[k] = \delta(0) + \frac{1}{2}\left[\delta(w - 2000\pi) + \delta(w + 2000\pi)\right] + \frac{j}{2}\left[\delta(w - 4000\pi) - \delta(w + 4000\pi)\right]$$

lo cual indica que el espectro de frecuencia tendrá valores en el intervalo $[-4000\pi, 4000\pi]$, o bien que $\omega_b = 4000\pi$. De acuerdo con esto la razón de Nyquist es $\omega_s = 8000\pi$.

b) $x(t) = \frac{sen(200\pi t)}{\pi t}$. Podemos ver que la antitransformada de Fourier de una caja es

$$\begin{aligned} x(t) &= A\int_{-d}^{d} e^{j\omega t} d\omega \\ x(t) &= \frac{A}{jt}\, e^{j\omega t}\Big|_{-d}^{d} \\ x(t) &= \frac{1}{jt}\,(2j)\, sen\,[dt] \\ x(t) &= \frac{2A sen\,[dt]}{t} \end{aligned}$$

de donde podemos ver que la transformada de Fourier de esta señal es una caja de ancho $d = 200\pi$ y altura $A = \frac{1}{2\pi}$, por lo tanto su ancho de banda será de $\omega_b = 200\pi$ y su razón de Nyquist será $\omega_s = 400\pi$.

Ejemplo 2

Dada la señal continua $x(t) = sen^2(10\pi t)$, calcular:

a) El ancho de banda correspondiente a esta señal

b) La transformada discreta de Fourier si la señal es muestreada en con periodo $T_s = \frac{5}{2N}$ segundos, con un número de muestras $N = 128$.

c) Cual es la transformada discreta de Fourier si la señal es muestreada con un periodo del doble del anterior.

Para esta señal aplicando identidades trigonométricas tenemos que:

$$x(t) = sen^2(10\pi t) = \frac{1}{2}\,(1 - cos(20\pi t))$$

Respuestas

a) La frecuencia del primer termino es $\omega_0 = 0$ y del segundo termino es $\omega_1 = 20\pi$, con lo cual tenemos que el ancho de banda es $\omega_b = 20\pi$

b)En este caso la frecuencia de muestreo es $\omega_m = \frac{2\pi}{\frac{5}{2N}} = \frac{4N\pi}{5}$ considerando 128 muestras la frecuencia de muestreo es $\omega_m = 321.6991$, note que

$$\omega_m >= 2 * \omega_b$$
$$321.6921 >= 40\pi$$

Nuestra señal cumple con el teorema del muestreo, ahora es necesario calcula k de la frecuencia $\frac{2\pi}{N}$ para ello

$$x_p[n] = x(nT)$$
$$x_p[n] = \frac{1}{2}\left(1 - cos(20\pi nT)\right)$$
$$x_p[n] = \frac{1}{2}\left(1 - cos(\frac{2\pi}{N}kn)\right)$$

Por comparación podemos ver que $20\pi T = \frac{2\pi}{N}k$ con lo cual tenemos

$$\begin{aligned} \frac{2\pi}{N}k &= 20\pi T \\ k &= \frac{20\pi TN}{2\pi} \\ k &= \frac{20\pi\frac{5}{2N}N}{2\pi} \\ k &= 25 \end{aligned}$$

Con esto tenemos que la transformada Discreta de Fourier es

$$X[k] = 64\delta[k] - 32\left(\delta(k-25) + \delta(k+25)\right))$$

c) Si duplicamos el periodo de Muestreo $T_s = \frac{5}{N}$ aplicando los pasos anteriores tenemos

$$\begin{aligned} \frac{2\pi}{N}k &= 20\pi T \\ k &= \frac{20\pi TN}{2\pi} \\ k &= \frac{20\pi \frac{5}{N}N}{2\pi} \\ k &= 50 \end{aligned}$$

y la TDF es

$$X[k] = 64\delta[k] - 32\left(\delta(k-50) + \delta(k+50)\right))$$

Note como la frecuencia de muestreo afecta la transformada discreta de Fourier.

Ejemplo 3

Para el ejemplo anterior considerando un periodo de muestreo $T_s = 0.0390625$, calcular utilizando el código dado y los valores muestreados $x_p[n]$, los valores de la función en $t = T$, $t = 1.5T$ y $t = 2.0T$.

Solución

Para llenar el arreglo dada la función $x(t) = sen^2(10\pi t)$ se hizo

$$x_p[n] = sen^2(10\pi nT)$$

Al aplicar

	t	calculados	valores reales
T	0.0390625	0.886505	0.886505
1.5 T	0.05859375	0.928864	0.928864
2.0 T	0.078125	0.402455	0.402455

Transformadas Coseno y Wavelet

6.1. Transformada Coseno a partir de la Transformada de Fourier

Partiremos de una señal x de tamaño $2N-1$ dada por (6.27) la cual es real y par. La transformada de Fourier de esta señal también será real y par de acuerdo con la propiedades de la Transformada de Fourier.

$$x(n) = x(2N-1-n) \tag{6.27}$$

de otra forma

$$\begin{aligned} x(0) &= X(2N-1) \\ x(1) &= X(2N-2) \\ x(2) &= X(2N-3) \\ &\vdots \end{aligned}$$

Un ejemplo de esta señal podría ser $x = [1,2,3,4,4,3,2]$. La transformada de Fourier de cualquier señal x la con la forma dada por (6.27) la podemos calcular como:

$$X(k) = \sum_{n=0}^{2N-2} x(n) e^{-j\frac{2\pi nk}{2N-1}}$$

La sumatoria la podemos dividir en dos partes

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi nk}{2N-1}} + \sum_{n=N}^{2N-2} x(n) e^{-j\frac{2\pi nk}{2N-1}}$$

Si en la segunda sumatoria hacemos el cambio de variable $n = 2N - 1 - \hat{n}$ tenemos que $\hat{n} = 2N - 1 - n$ y podemos escribir

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi nk}{2N-1}} + \sum_{\hat{n}=N-1}^{1} x(2N-1-\hat{n}) e^{-j\frac{2\pi(2N-1-\hat{n})k}{2N-1}}$$

Reorganizando términos tenemos

$$X(k) = x(0) + \sum_{n=1}^{N-1} x(n) e^{-j\frac{2\pi nk}{2N-1}} + \sum_{n=1}^{N-1} x(2N-1-n) e^{-j\frac{2\pi(-n)k}{2N-1}} \cancel{e^{-j\frac{2\pi(2N-1)k}{2N-1}}}^{1}$$

Dado que $x(n) = x(2N-1-n)$ podemos simplificar

$$X(k) = x(0) + \sum_{n=1}^{N-1} x(n) \left(e^{-j\frac{2\pi nk}{2N-1}} + e^{+j\frac{2\pi nk}{2N-1}} \right)$$

$$X(k) = x(0) + 2\sum_{n=1}^{N-1} x(n) Cos\left(\frac{2\pi nk}{2N-1}\right) \tag{6.28}$$

La función que resulta ya es una sucesión de cosenos por lo cual podemos decir que es una Transformada Coseno implementada a partir de la transformada de Fourier.

El Código en Java para hacer la implementación de la Transformada Coseno es:

```
static public void MiTCos(float x[], float X[]) {
    int N = x.length;
    int n, k;
    double w;

    w = (2.0*Math.PI/(2.0*N-1.0));

    for(k=0; k<N; k++){
        X[k] = x[0];
        for(n=1; n<N; n++)
            X[k] += x[n]*2.0*Math.cos(w*k*n);
    }
}
```

6.1.1. Transformada Inversa

Para llevar a cabo la transformada inversa consideraremos los mismos principios que la transformada Coseno. Por definición la transformada Inversa es:

$$x(n) = \frac{1}{2N-1} \sum_{k=0}^{2N-2} X(k) e^{j\frac{2\pi nk}{2N-1}}$$

La sumatoria la podemos dividir en dos partes

$$x(n) = \frac{1}{2N-1} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi nk}{2N-1}} + \frac{1}{2N-1} \sum_{k=N}^{2N-2} X(k) e^{j\frac{2\pi nk}{2N-1}}$$

Si en la segunda sumatoria hacemos el cambio de variable $k = 2N-1-\hat{k}$ tenemos que $\hat{k} = 2N-1-k$ y podemos escribir

$$x(n) = \frac{1}{2N-1} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi nk}{2N-1}} + \frac{1}{2N-1} \sum_{\hat{k}=N-1}^{1} X(2N-1-\hat{k}) e^{j\frac{2\pi(2N-1-\hat{k})n}{2N-1}}$$

Reorganizando términos tenemos

$$x(n) = \frac{1}{2N-1} \left(X(k) + \sum_{k=1}^{N-1} X(k) e^{j\frac{2\pi nk}{2N-1}} + \sum_{k=1}^{N-1} X(2N-1-k) e^{j\frac{2\pi n(-k)}{2N-1}} \cancel{e^{-j\frac{2\pi(2N-1)n}{2N-1}}} \right)^{1}$$

Dado que $X(k) = X(2N-1-k)$ podemos simplificar

$$x(n) = \frac{1}{2N-1} \left(X(0) + \sum_{k=1}^{N-1} X(k) \left(e^{j\frac{2\pi nk}{2N-1}} + e^{-j\frac{2\pi nk}{2N-1}} \right) \right)$$

Finalmente

$$x(n) = \frac{1}{2N-1} \left(X(0) + 2 \sum_{k=1}^{N-1} X(k) Cos \left(\frac{2\pi nk}{2N-1} \right) \right) \tag{6.29}$$

6.1.2. Ejemplo

Consideremos una señal $x_1 = [1, 2, 3, 4, 4, 3, 2]$ y otra señal $x_2 = [1, 2, 3, 4]$ calcular:

1. Calcular la transformada de Fourier x_1,
2. Calcular la transformada Coseno dada por (6.28) para x_2 ,
3. Calcular la transformada Coseno Inversa dada por (6.29) para $X_2(k)$ y
4. Concluir

La Transformada de Fourier de x_1 es

$$X_1(k) = [19, -5.04892, -0.307979, -0.643104, -0.643104, -0.307979, -5.04892]$$

La Transformada Coseno de x_2 es

$$X_2(k) = [19.0, -5.04892, -0.307979, -0.643104]$$

La transformada Coseno inversa para $X_2(k)$ es

$$\hat{x}_2(n) = [1, 2, 3, 4]$$

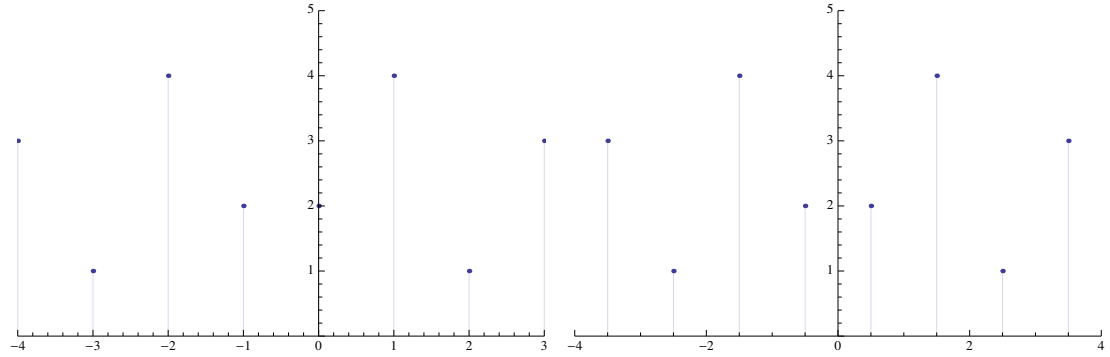
Note que los términos de $X_2(k)$ corresponden exactamente a los primero 4 términos de $X1(k)$.

6.2. Definición de la DCT

Para derivar la DCT de una señal real de con N muestras $x' = [x'(0), x'(1), \cdots, x'(N-1)]$, primeramente construiremos una señal $x'(n)$ con $2N$ puntos

$$x(n) = \begin{cases} x'(n) & (0 \leq n \leq N-1) \\ x'(-n-1) & (-N \leq n \leq -1) \end{cases} \tag{6.30}$$

En la Figura 6.15(f) se muestra la imagen correspondiente a la señal generada con (6.30). Pare esta señal $x(n)$ asumiremos que tiene periodo $2N$ y es par con respecto al punto $n = -1/2$. Si aplicamos una translación de $1/2$ hacia la derecha de la señal $x(n)$ haciendo el cambio de variable $n' = n + 1/2$, entonces tenemos que $x(n) = x(n' - 1/2)$ y la señal resultante es par con respecto a $n' = 0$ como se muestra en la Figura 6.15(g).



(f) I_1 (g) I_1

Figura 6.15: Reconstrucción de la señal usando los coeficientes de Fourier

La transformada discreta de Fourier para la señal $x(n'-1/2)$ la podemos calcular con

$$X(k) = \sum_{n'=-N+1/2}^{N-1/2} x\left(n' - \frac{1}{2}\right) e^{-j2\pi n'k/2N}$$

Podemos hacer la extensión de la exponencial en su forma rectangular y dado que x es real expresar

$$X(k) = \sum_{n'=-N+1/2}^{N-1/2} x\left(n' - \frac{1}{2}\right) cos(2\pi n'k/2N) - j \sum_{n'=-N+1/2}^{N-1/2} x\left(n' - \frac{1}{2}\right) sin(2\pi n'k/2N)$$

Dado que la señal x es par y que el $cos(.)$ y el $sin(.)$ son par e impar respectivamente con respecto a $n' = 0$ entonces la segunda sumatoria será cero.

$$X(k) = \sum_{n'=-N+1/2}^{N-1/2} x\left(n' - \frac{1}{2}\right) cos(2\pi n'k/2N)$$

Separando la sumatoria en dos términos

$$X(k) = \sum_{n'=-N+1/2}^{-1/2} x\left(n' - \frac{1}{2}\right) cos(2\pi n'k/2N) + \sum_{n'=1/2}^{N-1/2} x\left(n' - \frac{1}{2}\right) cos(2\pi n'k/2N)$$

$$X(k) = \sum_{n'=1/2}^{N-1/2} x\left(-n' + \frac{1}{2}\right) cos(-2\pi n' k/2N) + \sum_{n'=1/2}^{N-1/2} x\left(n' - \frac{1}{2}\right) cos(2\pi n' k/2N)$$

Dado que la señal es par entonces $x(n' - 1/2) = x(-n' + 1/2)$ podemos simplificar la expresión y obtener

$$X(k) = 2 \sum_{n'=1/2}^{N-1/2} x\left(n' - \frac{1}{2}\right) cos(2\pi n' k/2N)$$

Finalmente si hacemos el cambio de variable $n = n' - 1/2$ obtenemos

$$X(k) = 2 \sum_{n=0}^{N-1} x(n) cos\left(\frac{\pi(n + 1/2)k}{N}\right) \tag{6.31}$$

6.2.1. Transformada Coseno Inversa

Para derivar la DCT inversa de una señal real X' con N muestras $X' = [X'(0), X'(1), \cdots, X'(N-1)]$, primeramente construiremos una nueva señal X con $2N$ puntos dada por (6.32)

$$X(k) = \begin{cases} X'(k) & (0 \leq k \leq N-1) \\ X'(-k-1) & (-N \leq k \leq -1) \end{cases} \tag{6.32}$$

Pare esta señal $X(k)$ asumiremos que tiene periodo $2N$ y es par con respecto al punto $k = -1/2$. Si aplicamos una translación de $1/2$ hacia la derecha de la señal $X(k)$ haciendo el cambio de variable $k' = k + 1/2$, entonces tenemos que $X(k) = X(k' - 1/2)$ y la señal resultante es par con respecto a $k' = 0$.

La transformada discreta de Fourier inversa para la señal $X(k' - 1/2)$ la podemos calcular con

$$x(n) = \frac{1}{2N} \sum_{k'=-N+1/2}^{N-1/2} X\left(k' - \frac{1}{2}\right) e^{j2\pi n k'/2N}$$

Podemos hacer la extensión de la exponencial en su forma rectangular y dado que X es real podemos expresar

$$\begin{aligned} x(n) &= \frac{1}{2N} \sum_{k'=-N+1/2}^{N-1/2} X\left(k' - \frac{1}{2}\right) cos(2\pi n k'/2N) \\ &+ \frac{j}{2N} \sum_{k'=-N+1/2}^{N-1/2} X\left(k' - \frac{1}{2}\right) sin(2\pi n k'/2N) \end{aligned}$$

Dado que la señal X es par y que el $cos(.)$ y el $sin(.)$ son par e impar respectivamente con respecto a $n' = 0$ entonces la segunda sumatoria será cero.

$$x(n) = \frac{1}{2N} \sum_{k'=-N+1/2}^{N-1/2} X\left(k' - \frac{1}{2}\right) cos(2\pi n k'/2N)$$

Separando la sumatoria en dos términos

$$x(n) = \frac{1}{2N} \left(\sum_{k'=-N+1/2}^{-1/2} X\left(k' - \frac{1}{2}\right) cos(2\pi n k'/2N) + \sum_{k'=1/2}^{N-1/2} X\left(k' - \frac{1}{2}\right) cos(2\pi n k'/2N) \right)$$

$$x(n) = \frac{1}{2N} \left(\sum_{k'=1/2}^{N-1/2} X\left(-k' + \frac{1}{2}\right) cos(-2\pi n k'/2N) + \sum_{k'=1/2}^{N-1/2} X\left(k' - \frac{1}{2}\right) cos(2\pi n k'/2N) \right)$$

Dado que la señal es par entonces, es decir $X(k'-1/2) = X(-k'+1/2)$, podemos simplificar la expresión y obtener

$$x(n) = \frac{2}{2N} \sum_{k'=1/2}^{N-1/2} X\left(k' - \frac{1}{2}\right) cos(2\pi n k'/2N)$$

Finalmente si hacemos el cambio de variable $k = k' - 1/2$ obtenemos

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) cos\left(\frac{\pi n(k+1/2)}{N}\right) \tag{6.33}$$

6.2.2. Transformadas Coseno Utilizadas

Se pude tener un sin número de definiciones de la transformada coseno, sin embargo, centraremos en cuatro principalmente:

tipo	Definición
1	$X_1(k) = \sqrt{\frac{2}{N-1}}\left(\frac{x(0)}{2} + (-1)^s \frac{x(n-1)}{2} \sum_{n=1}^{N-2} x(n) cos\left(\frac{\pi n k}{N-1}\right)\right)$
2	$X_2(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) cos\left(\frac{\pi(n+1/2)k}{N}\right)$
3	$X_3(k) = \frac{1}{\sqrt{N}}\left(x(0) + 2\sum_{n=1}^{N-1} x(n) cos\left(\frac{\pi n(k+1/2)}{N}\right)\right)$
4	$X_4(k) = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) cos\left(\frac{\pi(n+1/2)k}{N}\right)$

Las transformadas discreta de coseno inversa para los tipos 1,2,3 y 4 es la tipo 1, 3, 2 y 4 respectivamente.

6.2.3. Ejemplo de aplicación

Dada una señal $x[n] = sin(0.5n) + N(0, 0.3)$ queremos obtener una señal filtrada eliminando el 75 % de los coeficientes correspondientes a las altas frecuencias. Considere que se tienen $N = 64$ muestras.

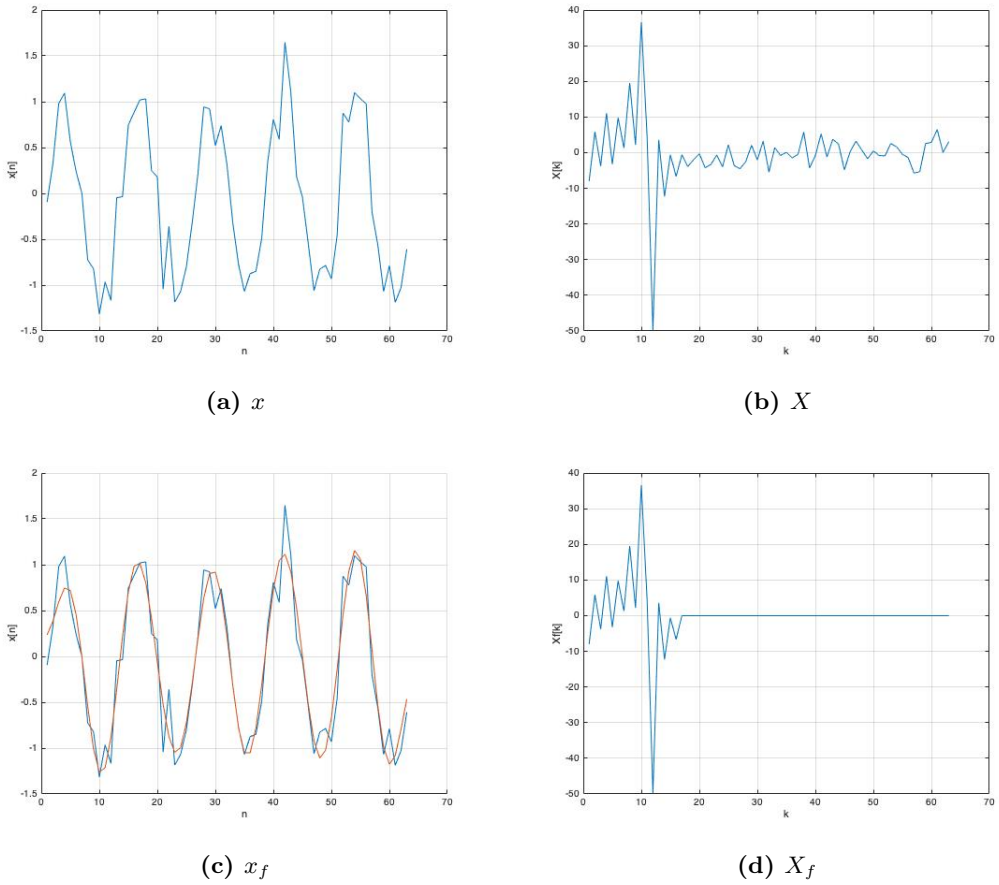
En la Figura 6.16 se muestran los resultados en cuatro sub figuras. En la Figura 6.16(a) se muestra la señal original la cual tiene una frecuencia $w = 0.5$. La Figura 6.16(b) muestra la transformada de Coseno de la señal x, note como cerca de la frecuencia fundamental se muestra el máximo de la señal. La Figura 6.16(c) muestra la Transformada Coseno una vez que se eliminaron el 75 % de las altas frecuencias y finalmente en la Figura 6.16(d) sobre impuestas la señal original y la señal muestreada.

La frecuencia fundamental la podemos calcular haciendo

$$0.5n = \frac{\pi(k + 1/2)n}{64}$$

despejando tenemos

$$k = \frac{64 \times 0.5}{\pi} - \frac{1}{2} = 9.6858$$



(a) x **(b)** X

(c) x_f **(d)** X_f

Figura 6.16: Ejemplo de filtrado y compactación de una señal utilizando Transformada Coseno

6.2.4. Transformada Discreta Coseno en dos Dimensiones

En forma similar al cálculo de la TDC en una dimensión donde se definió una señal par (6.30), en dos dimensiones partimos de la transformada discreta de Fourier para la señal bidimensional par desplazada $x(n' - 1/2, m' - 1/2)$.

$$X(k,l) = \sum_{n'=-N+1/2}^{N-1/2} \sum_{m'=-M+1/2}^{M-1/2} x\left(n' - \frac{1}{2}, m' - \frac{1}{2}\right) e^{-j2\pi n'k/2N - j2\pi m'k/2M}$$

Aplicando la propiedad asociativa de la suma y la propiedad de los exponentes podemos escribir como (6.34)

$$X(k,l) = \sum_{n'=-N+1/2}^{N-1/2} \left(\sum_{m'=-M+1/2}^{M-1/2} x\left(n' - \frac{1}{2}, m' - \frac{1}{2}\right) e^{-j2\pi m'k/2M} \right) e^{-j2\pi n'k/2N} \quad (6.34)$$

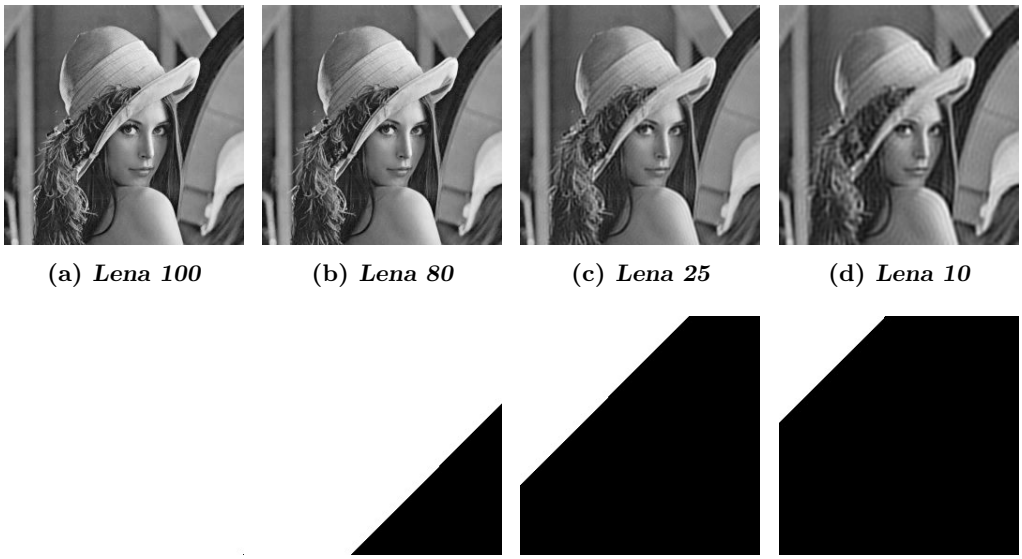
Si definimos X_c como (6.35) por podemos escribir (6.34). Esto significa que podemos calcular la transformada coseno primero por columnas X_c (6.34) y posteriormente por renglones (6.36).

$$X_c(k,l) = \sum_{m'=-M+1/2}^{M-1/2} x\left(n' - \frac{1}{2}, m' - \frac{1}{2}\right) e^{-j2\pi m'k/2M} \quad (6.35)$$

$$X(k,l) = \sum_{n'=-N+1/2}^{N-1/2} X_c\left(n' - \frac{1}{2}, m' - \frac{1}{2}\right) e^{-j2\pi n'k/2N} \quad (6.36)$$

6.2.5. Compresión de Imágenes utilizando TDC

Una aplicación de la TDC es la compresión de imágenes. Para mostrar comó se lleva cabo la compresión vamos a considerar una imagen a la cual le aplicaremos la TDC y nos quedaremos con el 100, 80, 25 y 10 % de las bajas frecuencias, a estas imágenes recortadas se les aplica la TDC inversa y el resultado se muestra en la Figura 6.17. En el primer renglón Figuras 6.17(a), 6.17(b), 6.17(c) y 6.17(d), se muestra las imágenes, resultado de dejar pasar 100, 80, 25 y 10 % de las bajas frecuencias. En el segundo renglón Figuras 6.17(e), 6.17(f), 6.17(g) y 6.17(h) se muestra en blanco los valores de la TDC que se conservan y en negro los que se eliminan. Note que se puede tener un recorte significativo de altas frecuencias y la señal que se obtiene mantiene mucha información de la imagen original, condición que permite hacer la compresión de la imagen.



(a) *Lena 100* (b) *Lena 80* (c) *Lena 25* (d) *Lena 10*

(e) *Mascara 100* (f) *Mascara 80* (g) *Mascara 25* (h) *Mascara 10*

Figura 6.17: Ejemplo de filtrado y compactación de una imagen utilizando Transformada Coseno

6.3. Transformada Wavelet

La Transformada Discreta Wavelet TDW de una señal x se calcula pasándolo a través de una serie de filtros. Primero, la señal x se pasan a través de un filtro de paso bajas con respuesta al impulso g para posteriormente muestrearla a la mitad. Esta operación se puede representar como la convolución dada por (6.37).

$$y_{low}[n] = \sum_{k=\infty}^{\infty} g[k]x[2n-k] \tag{6.37}$$

Posteriormente la señal x es convolucionada con un filtro pasa altas y mestreada a la mitad de acuerdo con (6.38). El proceso completo se muestra en la Figura 6.18

$$y_{high}[n] = \sum_{k=\infty}^{\infty} h[k]x[2n-k] \tag{6.38}$$

El código Java para realizar la TDW se muestra a continuación:

```
static void pwt(double a[], double cc[], double cr[], int n, int isign) {
```

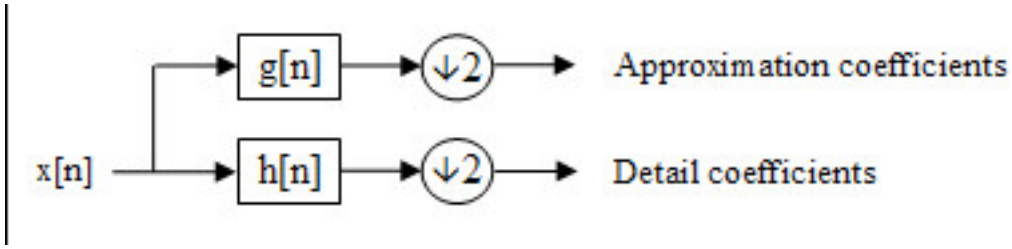


Figura 6.18: Ejemplo de aplicación de la transformada Wavelet

```
double ai,ai1,wksp[];
int i, ii, j, k, nh, ncof;

if (n < 4) return;

ncof = cc.length; // Numero de coeficientes de los kerneles
//System.out.println("Numero de coeficientes " + ncof);

nh=n >> 1;        // Calcula la mitad del arreglo
n = nh << 1;      // forza a que n sea par
wksp = new double[n];

for (j=0;j<n;j++) wksp[j]=0.0;

if (isign >= 0) {
    for (ii=0,i=0;i<n;i+=2,ii++) {
        for (k=0;k<ncof;k++) {
            j = (i+k)< n ? i+k : i+k-n;
            wksp[ii] += cc[k]*a[j];
            wksp[ii+nh] += cr[k]*a[j];
        }
    }
}
else {
    for (ii=0,i=0;i<n;i+=2,ii++) {
        ai=a[ii];
        ai1=a[ii+nh];
        for (k=0;k<ncof;k++) {
            j= (i+k) < n ? i+k : i+k-n;
            wksp[j] += cc[k]*ai;
            wksp[j] += cr[k]*ai1;
```

Cuadro 6.5: Transformada Wavelet para una señal discreta

x	56	40	8	24	48	48	40	16
$X^{(k)}$	d_{low}				d_{high}			
$X^{(1)}$	48	16	48	28	8	-8	0	12
$X^{(2)}$	32	38	16	10	8	-8	0	12
$X^{(3)}$	35	-3	16	10	8	-8	0	12

```
            }
        }
    }

    for (j=0;j<n;j++) a[j]=wksp[j];
}
```

6.3.1. Ejemplo

Dada la señal $x = [56, 40, 8, 24, 48, 48, 40, 10]$ y los filtros $g = [1/2, 1/2]$ y $h = [1/2, -1/2]$ determinar la transformada discreta wavelet de x.

Los resultados de la TDW se muestran en la tabla 6.5. En el primer renglón se muestra la señal original y en los siguiente renglones la TDW para diferentes niveles $k = 1, 2, 3, \cdots$. Note que el primer elemento del último renglón corresponde al promedio de la señal.

6.3.2. Antitransformada Wavelet

Consideremos el caso en que se utilizan dos kerneles $g = [c, c]$ y $h = [c, -c]$. Con esta condición podemos ver que:

$$d_{low}[i] = x[i] \times c + x[i+1] \times c$$
$$d_{high}[i] = x[i] \times c - x[i+1] \times c$$

Si sumamos y restamos $d_{low}[i]$ y $d_{high}[i]$ tenemos

$$d_{low}[i] + d_{high}[i] = 2x[i] \times c$$
$$d_{low}[i] - d_{high}[i] = 2x[i+1] \times c$$

Cuadro 6.6: Transformada Discreta Wavelet Inversa para una señal

X	35.0	-3.0	16.0	10.0	8.0	-8.0	0.0	12.0
$X^{(k)}$	d_{low}				d_{high}			
$X^{(3)}$	35.0	-3.0						
$X^{(2)}$	32.0	38.0	16.0	10.0				
$X^{(1)}$	48.0	16.0	48.0	28.0	8.0	-8.0	0.0	12.0
x	56.0	40.0	8.0	24.0	48.0	48.0	40.0	16.0

Finalmente obtenemos

$$x[i] = \frac{d_{low}[i] + d_{high}[i]}{2c}$$
$$x[i+1] = \frac{d_{low}[i] - d_{high}[i]}{2c}$$

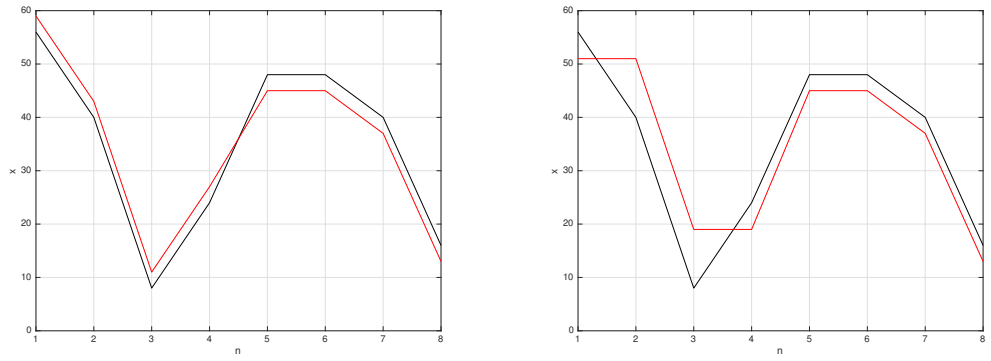
Esto nos permite ver qué la transformada wavelet es invertible y se puede mostrar que la invertibilidad no es exclusiva de kerneles con dos valores solamente. En la tabla 6.6 se muestra el procedimiento inverso para calcular la inversa de la TDW.

6.3.3. Transformada Wavelet para compresión de señales

Una aplicación de la TDW es la compresión de señales. Para ello se toma como criterio poner en cero los coeficientes con los valores menos significativos. Así por ejemplo dada la señal $x = [56.0, 40.0, 8.0, 24.0, 48.0, 48.0, 40.0, 16.0]$ y su transformada Wavelet es $X = [35, -3, 16, 10, 8, -8, 0, 12]$ podemos construir dos señales. La primera una señal donde se hacen cero todos los valores con valor absoluto menor o igual a 4 $X_1 = [35, 0, 16, 10, 8, -8, 0, 12]$ y una segunda señal donde se hacen cero los valores cuyo valor absoluto es menor igual a 9 $X_1 = [35, 0, 16, 10, 0, 0, 0, 12]$. Las antitransformadas de estás señales son $x_1 = [59.0, 43.0, 11.0, 27.0, 45.0, 45.0, 37.0, 13.0]$ y $x_2 = [51.0, 51.0, 19.0, 19.0, 45.0, 45.0, 37.0, 13.0]$ respectivamente. En la Figura 6.19(a) se muestra la señal x en negro y la señal x_1 en rojo y de manera similar para la Figura 6.19(b).

6.3.4. Transformada Wavelet en dos dimensiones

La transformada Discreta Wavelet pude realizarse en mas de una dimensión al igual que la convolución. Sin embargo si consideramos kerneles separables en cada dimensión la TDW



(a) ***Señal original y filtrada con*** $|X(k)| \geq 4$ **(b)** ***Señal original y filtrada con*** $|X(k)| \geq 9$

Figura 6.19: Ejemplo de filtrado y compactación de una señal utilizando Transformada Wavelet

puede realizarse de manera optima. Así dado un filtro pasa bajas g y un pasa altas h la transformada Wavelet puede llevarse a cabo mediante (6.39) y (6.40)

$$y_{low}[n,m] = \sum_{k=\infty}^{\infty} \left[\sum_{l=\infty}^{\infty} x[2n-k, 2m-l] g[l] \right] g[k] \tag{6.39}$$

$$y_{high}[n,m] = \sum_{k=\infty}^{\infty} \left[\sum_{l=\infty}^{\infty} x[2n-k, 2m-l] h[l] \right] h[k] \tag{6.40}$$

El código en Java correspondiente se muestra a continuación. La función pwd_T recibe una imagen I, el nivel al cual se va aplicar la TDW p y una variable $ising = 1$ para transformada directa y $ising = -1$ para transformada inversa. La función $pwd2d$ realiza la transformada Wavelet utilizando los kerneles $g = [1/\sqrt{2}, 1/\sqrt{2}]$ y $h = [1/\sqrt{2}, -1/\sqrt{2}]$

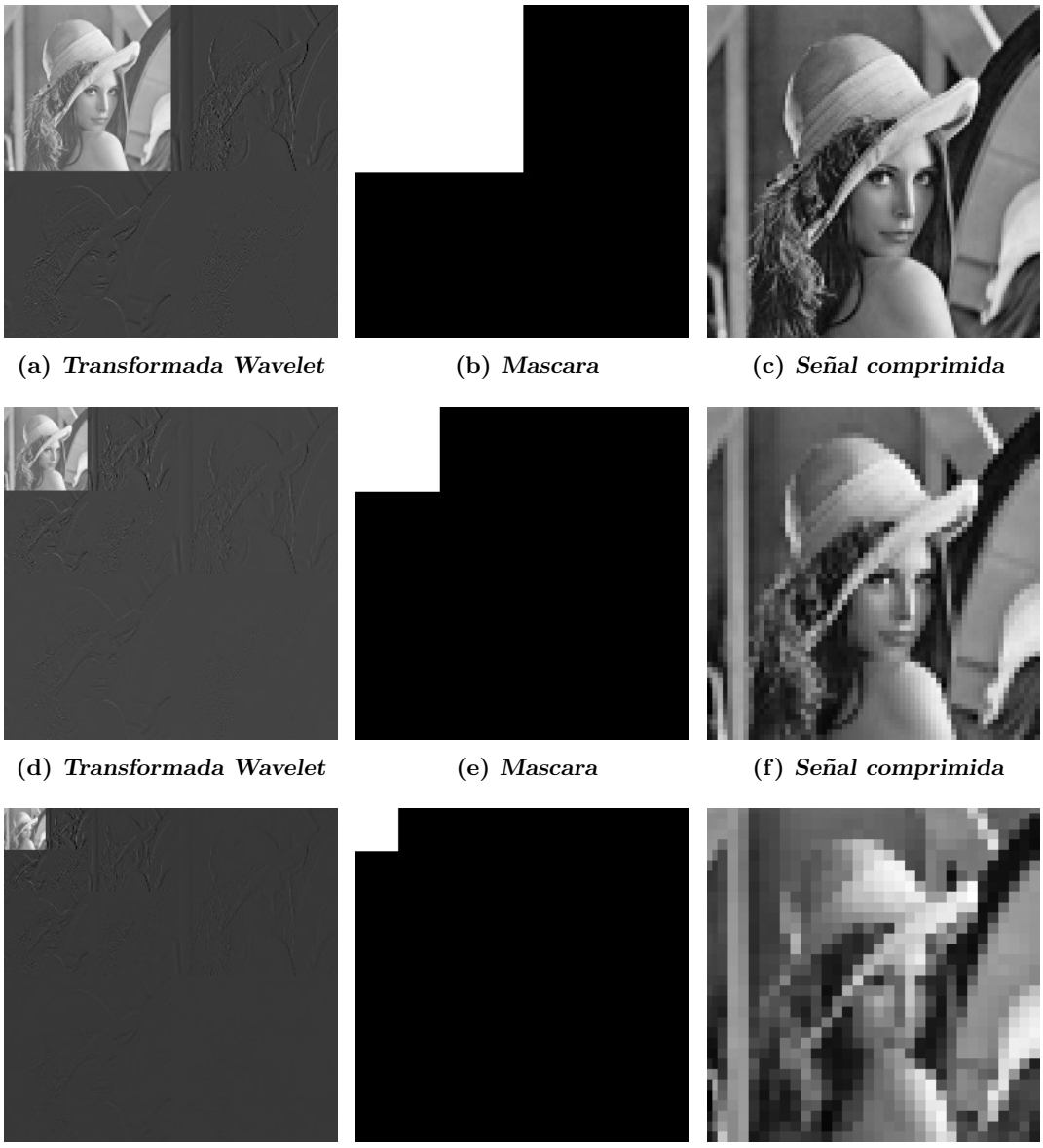
```
static public void pwd_T(double I[][], int p, int ising) {
     int i, j, nr = I.length, nc = I[0].length, fact;

     Imagen a;
     a = new Imagen(I, "datos");
     System.out.println("Datos " + p + " " + ising);

     if(ising > 0 ) fact = 1;
     else fact = 1 << (p-1);
```

```
        for(i=0; i < p; i++) {
            //System.out.println(fact + " " + p);
            pwd2d(I, nr/fact, nc/fact, ising);
            if(ising > 0) fact *= 2;
            else fact /=2;
        }
    }

    static public void pwd2d(double I[][], int nr, int nc, int ising){
        int i, j, N;

        N = nr > nc ? nr : nc;
        // Haar Wavelet Transform

        double c = 1.0/Math.sqrt(2);
        double cc[] = {c, c};
        double cr[] = new double[2];
        cr[0] = -cc[1]; cr[1] = cc[0];


        //Daubechies D4 Wavelet Transform

        double aux[] = new double[N];

        for(i=0; i<nr; i++) {
            for(j=0; j<nc; j++)
                aux [j] = I[i][j];
            pwt(aux, cc, cr, nc, ising);
            for(j=0; j<nc; j++)
                I[i][j] = aux [j];
        }

        for(j=0; j<nc; j++) {
            for(i=0; i<nr; i++)
                aux [i] = I[i][j];
            pwt(aux, cc, cr, nr, ising);

            for(i=0; i<nr; i++)
                I[i][j] = aux [i] ;
        }
```

```
}


static void pwt(double a[], double cc[], double cr[], int n, int isign) {
    double ai,ai1,wksp[];
    int i, ii, j, k, nh, ncof;

    //if (n < 4) return;

    ncof = cc.length; // Numero de coeficientes de los kerneles
    //System.out.println("Numero de coeficientes " + ncof);

    nh=n >> 1;        // Calcula la mitad del arreglo
    n = nh << 1;      // forza a que n sea par
    wksp = new double[n];

    for (j=0;j<n;j++) wksp[j]=0.0;

    if (isign >= 0) {
        for (ii=0,i=0;i<n;i+=2,ii++) {
            for (k=0;k<ncof;k++) {
                j = (i+k)< n ? i+k : i+k-n;
                wksp[ii] += cc[k]*a[j];
                wksp[ii+nh] += cr[k]*a[j];
            }
        }
    }
    else {
        for (ii=0,i=0;i<n;i+=2,ii++) {
            ai=a[ii];
            ai1=a[ii+nh];
            for (k=0;k<ncof;k++) {
                j= (i+k) < n ? i+k : i+k-n;
                wksp[j] += cc[k]*ai;
                wksp[j] += cr[k]*ai1;
            }
        }
    }

    for (j=0;j<n;j++) a[j]=wksp[j];
}
```

6.3.5. Ejemplo de compresión de Imágenes

Una forma de llevar a cabo la compresión de imágenes utilizando la Transformada Wavelet es hacer cero los valores de frecuencia alta. El la Figura 6.20 se muestra un ejemplo de la imagen de Lena. La primer columna corresponde a la Transformada Wavelest de la imagen, la segunda columna a la mascara utilizada para eliminar frecuencias y la tercer columna la anti transformada Wavelet. Por renglones tenemos diferentes niveles de la transformada Wavelet.



(a) ***Transformada Wavelet*** (b) ***Mascara*** (c) ***Señal comprimida***

(d) ***Transformada Wavelet*** (e) ***Mascara*** (f) ***Señal comprimida***

(g) ***Transformada Wavelet*** (h) ***Mascara*** (i) ***Señal comprimida***

Figura 6.20: Ejemplo de filtrado y compactación de una imagen utilizando Transformada Wavelet

Filtros.

7.1. Filtros Pasa bajas.

La idea de un filtro es permitir el paso de solamente un cierto ancho de banda de un espectro de frecuencias. En el caso de un filtro pasa bajas la frecuencias que este dejará pasar son aquellas que se encuentran cerca de la frecuencia cero o componente de C.D.

El filtro pasa bajas más simple es $h = [\frac{1}{2}, \frac{1}{2}]$ consideremos una señal dada por $x = [1,1,1,2,2,2,2,1,1,1]$ al realizar la convolución de esta señal con el kernel tenemos

$$\begin{aligned} z_1 &= x * h \\ &= [1,1,1,2,2,2,2,1,1,1] * [\frac{1}{2}, \frac{1}{2}] \\ &= [1,1,1,1.5,2,2,1.5,1,1,1] \end{aligned}$$

si volvemos a convolucionar con el mismos kernel tenemos

$$z_2 = [1,1,1,1.25,1.75,1.75,1.25,1,1,1]$$

el aplicar sucesivamente el filtrado pasa–bajas dará lugar a una señal plana. En lugar de convolucionar varias veces la señal x con el kernel h resulta más practico hacer la convolución del kernel consigo mismo dando lugar a

1,1

1,2,1

1,3,3,1

1,4,6,4,1

Pero veamos la transformada de Fourier del kernel propuesto para ser filtro pasa–bajas.

$$\begin{aligned} H[k] &= \sum_{k=0}^{N} h(n) e^{-j\left(\frac{2\pi}{N}\right)nk} \\ &= \left[1 \times e^{-j\left(\frac{2\pi}{N}\right)0k} + 1 \times e^{-j\left(\frac{2\pi}{N}\right)1k}\right] \\ &= \left[1 + e^{-j\left(\frac{2\pi}{N}\right)k}\right] \end{aligned}$$

la magnitud de este filtro es

$$\begin{aligned} |H(k)| &= \sqrt{\left(1 + \cos\frac{2\pi k}{N}\right)^2 + \left(sen\frac{2\pi k}{N}\right)^2} \\ &= \sqrt{1 + 2\cos\frac{2\pi k}{N} + \cos^2\frac{2\pi k}{N} + sen^2\frac{2\pi k}{N}} \\ &= \sqrt{2 + 2\cos\frac{2\pi k}{N}} \\ &= 2\cos\frac{\pi k}{N} \end{aligned}$$

note que esta función tiene un máximo en cero y un mínimo en N, razón por la cual, deja pasar las bajas frecuencias y atenúa las altas frecuencias. En la figura podemos ver el comportamiento del filtro pasa bajas en el dominio de la frecuencia (ver figura 7.21).

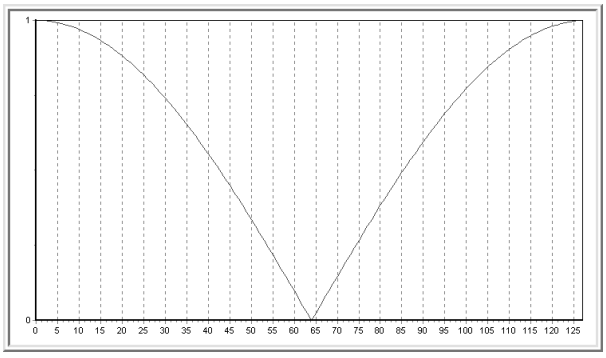


Figura 7.21: Respuesta a la frecuencia del filtro de pasabajas

7.2. Filtros Pasa Altas.

El filtro pasa bajas más simple es $h = \left[-\frac{1}{2}, \frac{1}{2}\right]$ consideremos una señal dada por $x = [1,1,1,2,2,2,2,1,1,1]$ al realizar la convolución de esta señal con el kernel tenemos

$$\begin{aligned} z_1 &= x * h \\ &= [1,1,1,2,2,2,2,1,1,1] * \left[-\frac{1}{2}, \frac{1}{2}\right] \\ &= [1,0,0,-0.5,0,0,0.5,0,0,0] \end{aligned}$$

El aplicar sucesivamente el filtrado pasabajas dará lugar a una señal plana. En lugar de convolucionar varias veces la señal $x(n)$ con el kernel $h(n)$ resulta más práctico hacer la convolución del kernel consigo mismo dando lugar a

+1,-1

+1,-2,+1

+1,-3,+3,-1

+1,-4,+6,-4,+1

Pero veamos la transformada de Fourier del kernel propuesto para ser filtro pasa altas.

$$\begin{aligned} H(k) &= \sum_{k=0}^{N} h(n) e^{-j\left(\frac{2\pi}{N}\right)nk} \\ &= \left[1 \times e^{-j\left(\frac{2\pi}{N}\right)0k} - 1 \times e^{-j\left(\frac{2\pi}{N}\right)1k}\right] \\ &= \left[1 - e^{-j\left(\frac{2\pi}{N}\right)k}\right] \end{aligned}$$

la magnitud de este filtro es

$$
\begin{aligned}
|H(k)| &= \sqrt{\left(1-\cos\frac{2\pi k}{N}\right)^2+\left(sen\frac{2\pi k}{N}\right)^2} \\
&= \sqrt{1-2\cos\frac{2\pi k}{N}+\cos^2\frac{2\pi k}{N}+sen^2\frac{2\pi k}{N}} \\
&= \sqrt{2-2\cos\frac{2\pi k}{N}} \\
&= 2sen\frac{\pi k}{N}
\end{aligned}
$$

note que esta función tiene un máximo en $\frac{N}{2}$ y un mínimo en 0, razón por la cual, deja pasar las altas frecuencias y atenúa las bajas frecuencias. En la figura podemos ver el comportamiento del filtro pasa bajas en el dominio de la frecuencia (ver figura 7.22).

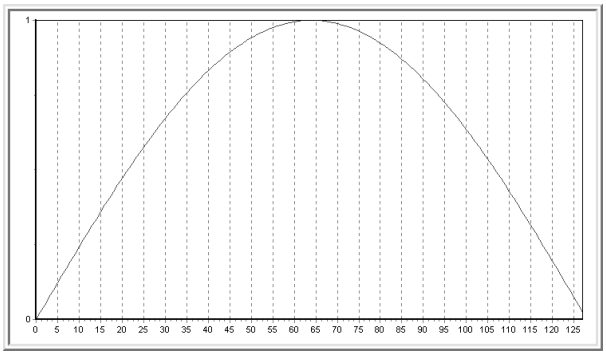


Figura 7.22: Respuesta a la frecuencia del filtro de pasa altas

7.3. Filtro pasa bajas Butterworth.

La magnitud al cuadrado de la respuesta de un filtro Butterworth esta dada por la expresión

$$
|H(k)|^2 = \frac{1}{1+\left(\frac{2\pi k}{N}\right)^{2n}}
$$

Note que $|H(0)|^2 = 1$ y que $\left|H\left(\frac{N}{2\pi}\right)\right|^2 = 0.5$, lo cual le da su característica de ser un filtro pasa bajas, pero adicionalmente este filtro tiene otro parámetro de control que es el

exponente n al cual esta elevado la frecuencia. Si cambiamos este valor tendremos que la ventana del filtro se modifica tal como se observa en la figura 7.23.

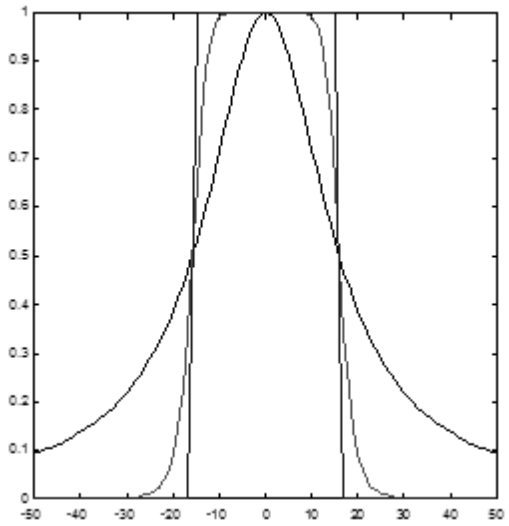


Figura 7.23: Respuesta a la frecuencia del filtro de Butherword

Podemos notar que para valores bajos de n la ventana es suave y para valores grandes tiende a una ventana cuadrada.

7.4. Filtros de pasa banda.

La ecuación de una función Gaussiana g la podemos representar por:

$$g(\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

y podemos demostrar que la transformada de Fourier es otra Gaussiana G dada como

$$g(\mu, \sigma_t) \overset{\mathcal{F}}{\leftrightarrow} G(\mu, \sigma_\omega)$$

y que $\sigma_t \sigma_\omega \geq K$. Esta expresión es la formulación del principio de incertidumbre y el caso de una Gaussiana se tiene la igualdad. El principio de incertidumbre establece que no podemos tener definición simultáneamente en el dominio de la frecuencia y del tiempo.

Esta función puede ser utilizada como filtro pasa bajas cuando $\mu = 0$, pero con valores diferentes cambiaremos la banda de la señal que queremos filtrar. Así por ejemplo para filtrar una señal consideraremos que la Gaussiana tiene una media cero y que aplicaremos una translación en el tiempo y/o en la frecuencia de la señal dada por μ

$$G(k) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{k^2}{2\sigma^2}}$$

si queremos hacer el filtrado en una frecuencia $w_k = \frac{2\pi}{N}k$ simplemente situamos una Gaussiana en $G(k-\mu)$ y otra en $G(k+\mu)$ la transformada de Fourier de estas será

$$\begin{aligned} G(k) &\overset{\mathcal{F}}{\leftrightarrow} g(n) \\ \frac{1}{2}G(k-\mu) &\overset{\mathcal{F}}{\leftrightarrow} \frac{1}{2}g(n)e^{j\left(\frac{2\pi}{N}\mu n\right)} \\ \frac{1}{2}G(k+\mu) &\overset{\mathcal{F}}{\leftrightarrow} \frac{1}{2}g(n)e^{-j\left(\frac{2\pi}{N}\mu n\right)} \end{aligned}$$

el filtro en la frecuencia es la suma de $G(k-\mu)$ y $G(k+\mu)$ por lo que el filtro de convolución es la suma de estos dos

$$\begin{aligned} h_1[n] &= \frac{1}{2}g(n)\left[\cos\left(\frac{2\pi}{N}\mu n\right) + jsen\left(\frac{2\pi}{N}\mu n\right)\right] \\ h_2[n] &= \frac{1}{2}g(n)\left[\cos\left(\frac{2\pi}{N}\mu n\right) - jsen\left(\frac{2\pi}{N}\mu n\right)\right] \\ h[n] &= h_1[n] + h_2[n] = g(n)\cos\left(\frac{2\pi}{N}\mu n\right) \end{aligned}$$

7.4.1. Ejemplo

Una señal $x(t) = seno(10t) + N(0, 0.3)$, esta corrompida con ruido de Naturaleza Gaussiana con media cero y varianza 0.3. Entonar calcular la media y la varianza de un filtro Gaussiano pasa banda dada una frecuencia de muestreo $T_s = 0.05$ seg y un número total de muestras de $N = 64$

Podemos calcular el múltiplo de la frecuencia $\frac{2\pi}{N}$ recordando que la frecuencia de la señal muestreada es $\omega_s = \omega t$. Con esto tenemos que

$$\begin{aligned} k &= \frac{\omega T N}{2\pi} \\ k &= \frac{10 \times 0.05 \times 64}{2\pi} \\ k &= 5.0929 \end{aligned}$$

Haciendo esto tenemos que nuestra Gaussiana la colocaremos en $\mu = 5.0929$ y dado que no tenemos una frecuencia entera utilizaremos una varianza $\sigma = 0.5$. El la figura 7.24(a), se muestra la señal original con ruido y en la figuras 7.24(b) y 7.24(c) la parte real e imaginaria de su transformada Discreta de Fourier. El la figura 7.24(d) se muestra las Gaussiana entonadas para filtrar la señal y en la figura 7.24(e) el resultado de haber aplicado el filtro.

7.5. Filtro de Gabor

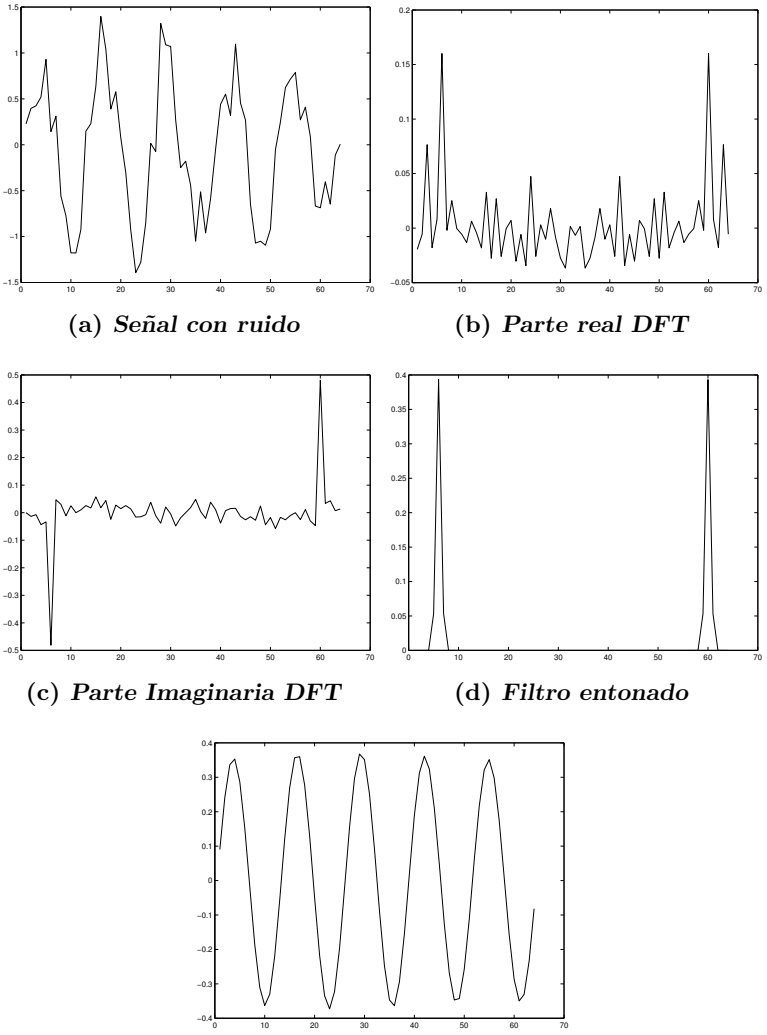
Pero que ocurre si en lugar de colocar un filtro en $\frac{1}{2}G\left(k-\mu\right)$ y otro en $\frac{1}{2}G(k+\mu)$, pusiéramos uno solo en $G\left(k-\mu\right)$

$$h[n] = g(n)\left[\cos\left(\frac{2\pi}{N}\mu n\right) + jsen\left(\frac{2\pi}{N}\mu n\right)\right]$$

note que la parte real es equivalente a la encontrada anteriormente, pero además tenemos una parte compleja. El hecho de tener solamente un filtro G en el dominio de la frecuencia, nos da más información acerca de la banda que andamos buscando

$$\begin{aligned} h_R[n] &= g(n)\cos\left(\frac{2\pi}{N}\mu n\right) \\ h_I[n] &= g(n)sen\left(\frac{2\pi}{N}\mu n\right) \end{aligned}$$

a este tipo de filtro se le conoce como filtro de Gabor. La magnitud del filtro nos dará los puntos de la señal donde la frecuencia para la cual entonamos el filtro produce esa señal, siendo mas alta en los puntos de dicha frecuencia.



(a) ***Señal con ruido*** **(b)** ***Parte real DFT***

(c) ***Parte Imaginaria DFT*** **(d)** ***Filtro entonado***

(e) ***Señal filtrada***

Figura 7.24: Ejemplo de filtrado de una señal con ruido

7.5.1. Filtro de Gabor en dos dimensiones

Para el caso de dos dimensiones el Kernel de Gabor que utilizaremos es

$$h[n,m] = g(n,m)e^{j\left(\frac{2\pi}{N}nk_0+\frac{2\pi}{M}ml_0\right)}$$

La parte real del filtro es

$$h_R[n,m] = g(n,m)\cos\left(\frac{2\pi}{N}nk_0 + \frac{2\pi}{M}ml_0\right)$$

y la parte imaginaria es

$$h_I[n,m] = g(n,m)\sin\left(\frac{2\pi}{N}nk_0 + \frac{2\pi}{M}ml_0\right)$$

Dado que la convolución requiere de $(NM)^2$ operaciones, resulta mas rápido implementar el kernel de Gabor de manera separable, así el número de operaciones será $N^2M + NM^2$. De manera separable la parte real queda y aplicando identidades trigonométricas podemos escribir:

$$\begin{aligned} h_R[n,m] &= g(n) * g(m)\left[\cos\left(\frac{2\pi}{N}nk_0\right)\cos\left(\frac{2\pi}{M}ml_0\right) - \sin\left(\frac{2\pi}{N}nk_0\right)\sin\left(\frac{2\pi}{M}ml_0\right)\right] \\ h_R[n,m] &= \left[g(n)\cos\left(\frac{2\pi}{N}nk_0\right)\right] * \left[g(m)\cos\left(\frac{2\pi}{M}ml_0\right)\right] - \left[g(n)\sin\left(\frac{2\pi}{N}nk_0\right)\right] * \left[g(m)\sin\left(\frac{2\pi}{M}ml_0\right)\right] \end{aligned}$$

Si definimos

$$\begin{aligned} g_1(n) &= g(n)\cos\left(\frac{2\pi}{N}nk_0\right) \\ g_2(n) &= g(n)\sin\left(\frac{2\pi}{N}nk_0\right) \\ g_3(m) &= g(m)\cos\left(\frac{2\pi}{M}ml_0\right) \\ g_4(m) &= g(m)\sin\left(\frac{2\pi}{M}ml_0\right) \end{aligned}$$

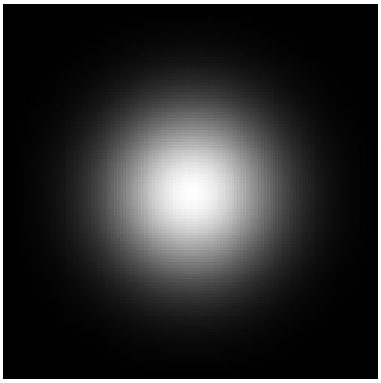
podemos escribir la parte real como

$$h_R[n,m] = g_1(n) * g_3(m) - g_2(n) * g_4(m)$$

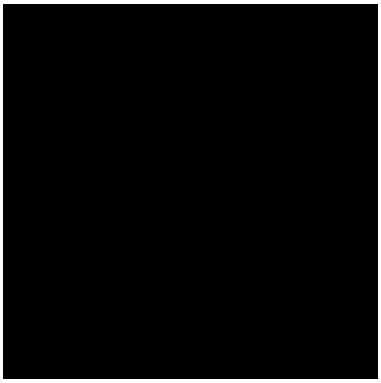
De forma similar podemos calcular la parte imaginaria y llegar al siguiente kernel separable.

$$h_I[n,m] = g_1(n) * g_4(m) + g_2(n) * g_3(m)$$

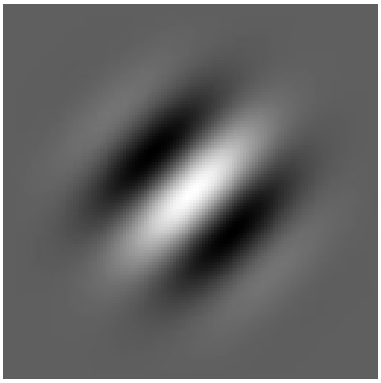
En la figura 7.5.1, podemos ver unos ejemplos del filtro de Gabor para diferentes valores de frecuencia espacial.



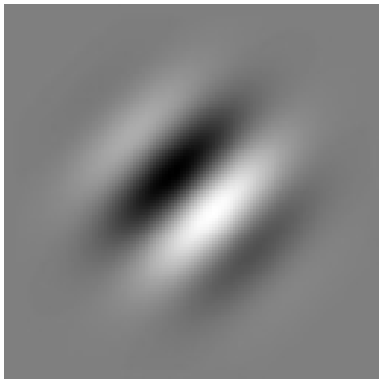
(a) ***Parte real, entonado en la frecuencia espacial (0,0)***



(b) ***Parte imaginaria, entonado en la frecuencia espacial (0,0)***



(c) ***Parte real, entonado en la frecuencia espacial (2,2)***



(d) ***Parte imaginaria, entonado en la frecuencia espacial (2,2)***

Figura 7.25: Filtros de Gabor

Resultado: f_r, f_i
Dadas k_0, l_0, σ_n, σ_m y I;
para $n = -3\sigma_n$ **a** $3\sigma_n$ **hacer**
  $g[n+3\sigma_n] = \frac{1}{\sqrt{2\pi}\sigma} exp^{-n^2/(2\sigma_n^2)}$;
  $g_1[n+3\sigma_n] = g[n+3\sigma_n] \times cos(2\pi n k_0/N)$;
  $g_2[n+3\sigma_n] = g[n+3\sigma_n] \times sin(2\pi n k_0/N)$;
fin
para $m = -3\sigma_m$ **a** $3\sigma_m$ **hacer**
  $g[m+3\sigma_m] = \frac{1}{\sqrt{2\pi}\sigma} exp^{-m^2/(2\sigma_m^2)}$;
  $g_3[m+3\sigma_m] = g[m+3\sigma_m] \times cos(2\pi m l_0/M)$;
  $g_4[m+3\sigma_m] = g[m+3\sigma_m] \times sin(2\pi m l_0/M)$;
fin
Calcular $f_r^{(1)}[n,m] = I[n,m] * g_1[n] * g_3[m]$;
Calcular $f_r^{(2)}[n,m] = I[n,m] * g_2[n] * g_4[m]$;
Calcular $f_i^{(1)}[n,m] = I[n,m] * g_1[n] * g_4[m]$;
Calcular $f_i^{(2)}[n,m] = I[n,m] * g_2[n] * g_3[m]$;
para $n = 0$ **a** $N-1$ **hacer**
  para $m = 0$ **a** $M-1$ **hacer**
    Calcular $f_r[n,m] = f_r^{(1)}[n,m] - f_r^{(2)}[n,m]$;
    Calcular $f_i[n,m] = f_i^{(1)}[n,m] + f_i^{(2)}[n,m]$;
  fin
fin

Algoritmo 4: Filtro de Gabor

7.6. Filtro de Wiener

Dadas dos señales x y y y un kernel h podemos establecer la siguiente relación entre las señales y el kernel dada por la convolución.

$$y[n] = x * h[n] = \sum_{m=0}^{N-1} x[m]h[n-m]$$

lo cual indica que si realizamos convolución con un kernel h de x, tendremos una señal y. Aplicando transformada de Fourier podemos intentar calcular el valor de la señal $x[n]$ haciendo:

$$y[n] = x * h[n]$$
$$Y[k] = X[k]H[k]$$

de esta expresión podemos despejar el valor de $X[k] = Y[k]/H[k]$, pero si un valor de $H[k] = 0$, no tenemos manera de estimar el valor del mismo.

En su lugar utilizaremos técnicas de Regularización, para ello, minimizamos la función

$$|y - x * h|^2$$
$$\text{sugeto a } |l * x|^2$$

donde $y[n]$ es la señal filtrada, $x[n]$ es la señal origen, $l[n]$ es un operador lineal que escogeremos de la mejor manera para facilitar la solución y $h[n]$ es el kernel. Para nuestro calculo procedemos

$$U(x) = \sum_{n=0}^{N-1} [y[n] - x * h[n]]^2 + \beta \sum_{n=0}^{N-1} |l * x[n]|^2$$
$$U(x) = \sum_{n=0}^{N-1} \left[y[n] - \sum_{m=0}^{N-1} x[m]h[n-m] \right]^2 + \beta \sum_{n=0}^{N-1} \left[\sum_{m=0}^{N-1} x[m]l[n-m] \right]^2$$

Para encontrar el valor de x que minimiza esta expresión hacemos

$$\begin{aligned} \frac{\partial U(x)}{\partial x[m]} = -2 \sum_{n=0}^{N-1} \left[y[n] - \sum_{m=0}^{N-1} x[m]h[n-m] \right] h[n-m] \\ +2\beta \sum_{n=0}^{N-1} \left[\sum_{m=0}^{N-1} x[m]l[n-m] \right] l[n-m] \\ = 0 \end{aligned}$$

reorganizado términos tenemos

$$\sum_{n=0}^{N-1} \left[\sum_{m=0}^{N-1} x[m]h[n-m] \right] h[n-m] + \beta \sum_{n=0}^{N-1} \left[\sum_{m=0}^{N-1} x[m]l[n-m] \right] l[n-m] = \sum_{n=0}^{N-1} y[n]h[n-m]$$

Definiendo $h^r[n] = h[-n]$ y $l^r[n] = l[-n]$:

$$\sum_{n=0}^{N-1}\left[\sum_{m=0}^{N-1} x[m]h[n-m]\right]h^r[m-n]+\beta\sum_{n=0}^{N-1}\left[\sum_{m=0}^{N-1} x[m]l[n-m]\right]l^r[m-n] = \sum_{n=0}^{N-1} y[n]h^r[m-n]$$

$$\sum_{n=0}^{N-1}\left[x*h[n]\right]h^r[m-n] + \beta\sum_{n=0}^{N-1}\left[x*l[n]\right]l^r[m-n] = y*h^r[m]$$

$$x*h*h^r[m] + \beta x*l*l^r[m] = y*h^r[m]$$

Aplicando el teorema de la convolución y las transformadas Discretas de Fourier para $DFT([x[-n]) = DFT([x^r[n]) = X[-k] \equiv X^*[k]$ y $DFT([h[-n]) = DFT([h^r[-n]) = H[-k] \equiv H^*[k]$ podemos reescribir:

$$X[k]H[k]H^*[k] + \beta X[k]L[k]L^*[k] = Y[k]H^*[k]$$

$$X[k]|H[k]|^2 + \beta X[k]|L[k]|^2 = Y[k]H^*[k]$$

Despejando para $X[k]$

$$X[k] = \frac{Y[k]H^*[k]}{|H[k]|^2 + \beta\,|L[k]|^2}$$

al sacar la antitrasformada de discreta de Fourier de X calculamos la señal x aproximada.

En dos dimensiones el filtro queda

$$X[k,l] = \frac{Y[k,l]H^*[k,l]}{|H[k,l]|^2 + \beta\,|L[k,l]|^2} \tag{7.41}$$

En general podemos considerar que $L[k,l] = cte$ y simplificar la expresión haciendo $\gamma = \beta|L[k,l]|^2$

$$X[k,l] = \frac{H^*[k,l]Y[k,l]}{|H[k,l]|^2 + \gamma} \tag{7.42}$$

Resultado: x
Dadas h, y y γ ;
Calcular $H = TDF[h]$ y $Y = TDF[y]$;
para $k \leftarrow 0$ **a** $N-1$ **hacer**
  para $l \leftarrow 0$ **a** $M-1$ **hacer**
    Calcular $X[k,l] = \frac{H[k,l]^* Y[k,l]}{|H[k,l]|^2+\gamma}$;
  fin
fin
Calcular $x = TDF^{-1}[X]$;

Algoritmo 5: Algoritmo de Winer

7.7. Como entonar un filtro

Resulta más natural al momento de entonar un filtro, hablar de frecuencias cuyas unidades son ciclos por segundo o Hertz. Dado el proceso de discretización, perdemos la noción del tiempo y nuestra frecuencia en el dominio discreto ahora esta dada en ciclos/muestra. La pregunta es como convertir Hertz a ciclos/muestra.

Consideremos una función muestreada con tamaño N, en el dominio de Fourier en la posición $k = N$ tenemos que N representa la frecuencia angular 2π, por lo tanto, podemos representar todas las frecuencia como múltiplos de $\frac{2\pi}{N}$ como $\omega = \frac{2\pi}{N}k$ [rad/muestra] y la frecuencia es $f_k = \frac{k}{N}$ [ciclos/muestra]. Dada una frecuencia de muestreo f_m en [muestras/seg] podemos calcular el tiempo necesario para una muestra, así : $1muestra = (1/f_m)\, seg$, sustituyendo esto tenemos:

$$f_t = \frac{k}{N}\ [ciclos/\,(1/f_m)\, seg]$$
$$f_t = \frac{f_m k}{N}\ [ciclos/seg]$$
$$f_t = f_m f_k\ [ciclos/seg]$$

finalmente $k = \frac{f_t}{f_m} N$.

7.7.1. Ejemplo

Consideremos una señal $x(t) = sen(200\pi t) + ruido$, la cual es muestreada con una $f_s = 800$ Hz, calcular el valor al cual debe ser entonado el filtro pasa bandas para un tamaño de la muestra $N = 128$.

La frecuencia de la señal es $f_t = 200\pi/(2\pi) = 100$ hz, por lo tanto el valor al que debe entonarse es $k = \frac{100}{800}128 = 16$. De otra forma

$$x_m[n] = sen(200\pi Tn) = sen\left(200\pi\frac{n}{800}\right) = sen\left(\frac{2\pi}{128}16n\right)$$

7.8. Filtro Elimina Banda

Dada una señal $x(n)$ el filtro pasa banda nos permitirá para la señal dada, solo mostrar la señal para la cual fue entonado el filtro. Pero como hacer si queremos mostrar la señal eliminando una banda dada.

En el domino de la frecuencia multiplicamos la transformada de Fourier de la señal por una Gaussiana posicionada en k_0, esto es

$$\widehat{X}_{k_0}(k) = X(k)\frac{1}{\sqrt{2\pi}\sigma_k}e^{\frac{-(k-k_0)^2}{2\sigma_k^2}}$$

Para eliminar la señal hacemos

$$\widehat{X}_{-k_0}(k) = X(k)\frac{1}{\sqrt{2\pi}\sigma_k}\left[1 - e^{\frac{-(k-k_0)^2}{2\sigma_k^2}}\right]$$

y aplicamos la transformada de Fourier

$$\begin{aligned}\widehat{x}_{-k_0}(n) &= x(n) * cte\left[\delta(n) - g(n)\,e^{\frac{2\pi k_0}{N}n}\right] \\ &= cte\left[x(n) - x(n) * g(n)\,e^{\frac{2\pi k_0}{N}n}\right] \\ &= x(n) - \widehat{x}_{k_0}(n)\end{aligned}$$

De la expresión anterior podemos deducir que el filtro elimina banda consiste en restar a la señal original, la señal después de aplicar el filtro pasa banda.

7.9. Filtro de Mediana

El filtro de mediana se utiliza para eliminar un tipo especial de ruido conocido como ruido sal y pimienta. Una manera de generar el ruido sal y pimienta en un porcentaje α a una imagen dada I es generado dos números aleatorios $0 \leq p \leq 1$ y $0 \leq q \leq 1$ con los cuales se puede alterar el valor de la imagen de la siguiente manera:

$$\text{Si } p \leq \alpha \quad J[r,c] = \begin{cases} 0 & \text{si } q < 0.5 \\ 255 & \text{si no} \end{cases}$$

$$\text{si no} \qquad J[r,c] = I[r,c]$$

Un filtro de mediana consiste en buscar en una vecindad de tamaño $(2w+1) \times (2w+1)$ la mediana de un conjunto de píxeles de la imagen con ruido J y reemplazar el valor del pixel centrada por la mediana. Así por ejemplo para un pixel en las coordenadas $[r,c]$ tenemos un conjunto de valores

$$W_{r,c} = \begin{bmatrix} J[r-w,c-w] & \cdots & J[r-w,c+w] \\ \vdots & J[r,c] & \vdots \\ J[r+w,c-w] & \cdots & J[r+w,c+w] \end{bmatrix}$$

El siguiente paso es hacer el ordenamiento de mayor a menor de los valores en $W_{r,c}$ utilizando el algoritmo Quick Sort y tomar el valor que se encuentra a la mitad.

Para calcular la mediana de cada pixel, será necesario hacer un barrido sobre todos los píxeles de la imagen J. El filtro de mediana tiene la característica de preservar los bordes. En seguida se muestra el código correspondiente a este filtro.

La Figura 7.26 a) muestra la imagen de Lena al aplicar un 1 % de ruido sal y pimienta y la Figura 7.26 b) la imagen filtrada después de aplicar un filtro de mediana en ventanas de tamaño 3×3.

```
void funciones::FiltroMediana(float **J, float **Img_fil,
   int Nr, int Nc, int w) {
    int i, j, k, l, n, Nw;
    float *window;

    Nw = (2*w+1)*(2*w+1);

    window = new float[Nw];
```

```
    for(i=0; i<Nr; i++)
        for(j=0; j<Nc; j++) {
            n = 0;
            media = 0;
            for(k=i-w; k<=i+w; k++)
                for(l=j-w; l<=j+w; l++){
                    if(k >=0 && k < Nr && l >=0 && l < Nc ) {
                        window[n] = J[k][l];
                        n++;
                    }
                }
            funciones::QuickSort(window, 0, n-1);

            Img_fil[i][j] = window[n/2];
        }
}
```

a) *b*)

Figura 7.26: A la izquierda imagen de Lena con 1 % de ruido y a la derecha la imagen al aplicar el filtro de mediada con ventanas de tamaño 3×3

7.10. Filtro Binario

En el caso de tener una imagen binaria aplicar un filtro de convolución dará como resultado una imagen con números reales. En este caso la convolución la reemplazaremos por

$$y[n,m] = \bigvee_{k=-\infty}^{\infty} \bigvee_{l=-\infty}^{\infty} x[k,l]\, h[n-k, m-l] = x \vee h$$

donde $\bigvee$ representa la operación or y esta operación es conocida como la or convolución o dilatación. La implementación en OpenCV se muestra a continuación.

```
void Dilacion(CvMat *ent, CvMat *sal, CvMat *h) {
    int i, j, k, l;
    int nr, nc, w, s;

    nr = ent->rows;
    nc = sal->cols;
    w = h->rows;

    for(i=0; i<nr-w; i++)
        for(j=0; j<nc-w; j++){
            s = 0;
            for(k=0; k<w; k++)
                for(l=0; l<w; l++) {
                    s += punto(ent, i+k, j+l)*punto(h, k, l);
                }
            punto(sal, i, j) = s >=1 ? 1 : 0;
        }
}
```

Por otro lado podemos calcular la and convolución como

$$y[n,m] = \bigwedge_{k=-\infty}^{\infty} \bigwedge_{l=-\infty}^{\infty} x[k,l]\, h[n-k, m-l] = x \wedge h$$

La or convolución también es conocida como operación de dilación y la and convolución como contracción o erosión.

```
void Erosion(CvMat *ent, CvMat *sal, CvMat *h) {
    int i, j, k, l;
    int nr, nc, w, s;
```

```
    nr = ent->rows;
    nc = sal->cols;
    w = h->rows;

    for(i=0; i<nr-w; i++)
        for(j=0; j<nc-w; j++){
            s = 1;
            for(k=0; k<w; k++)
                for(l=0; l<w; l++) {
                        s *= punto(ent, i+k, j+l)*punto(h, k, l);
                    }
                punto(sal, i, j) = s;
            }
}
```

Si hacemos la aplicación de ambos $(x \vee h) \wedge h$, tenemos primero una dilación y posteriormente una contracción, lo cual permite eliminar ruido de la imagen binaria. Se definen cerradura a la operación $(x \vee h) \wedge h$ y la apertura a la operación $(x \wedge h) \vee h$

Para obtener los bordes de la imagen podemos hacer la diferencia de la imagen original con la imagen dilatada haciendo $x - (x \vee h)$.

7.11. Filtro de Membrana

En esta sección describiremos la aplicación de la regularización al cálculo de un filtro de membrana. La regularización es una técnica de adaptable de filtrado de señales que permite estimar componentes de baja frecuencia. Abordaremos el problema desde un punto de vista estadístico, para ello, calcularemos primero el estimador de máxima verosimilitud, agregaremos información a priori y finalmente con regla de Bayes calcularemos el filtro.

7.11.1. Probabilidad de un evento

Si un experimento puede dar como resultado cualquiera N resultado diferentes igualmente probables, y si exactamente n de estos resultado corresponden al evento A, entonces la probabilidad del evento A es

$$P(A) = \frac{n}{N}$$

7.11.2. Probabilidad condicional

La probabilidad de que ocurra un evento B cuando se sabe que ha ocurrido algún otro evento A se denomina probabilidad condicional y se denota como $P(B|A)$.

Consideremos el espacio muestral $S = \{1, 2, 3, 4, 5, 6, 7\}$, donde cada uno de los eventos son igualmente probables y estos eventos se agrupan en dos conjuntos $A = \{1, 2, 3, 4\}$ y $B = \{3, 4, 5, 6, 7\}$, tal que $S = A \cup B$. De acuerdo a la definición de probabilidad de un evento, la probabilidad de A es $P(A) = 4/7$ y la probabilidad del conjunto B es $P(B) = 5/7$. En la figura podemos ver, de manera mas clara, como se agrupan estos conjuntos (ver fig 7.27).

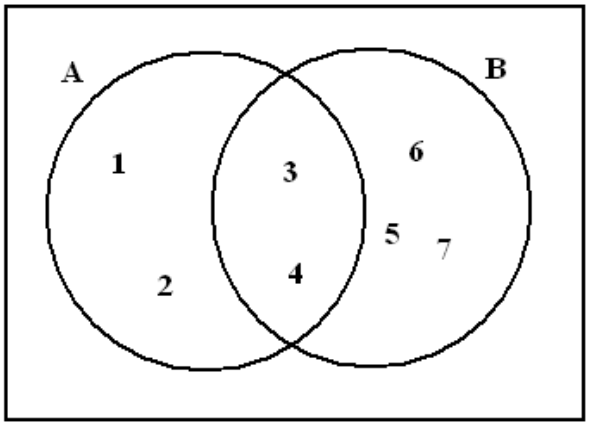


Figura 7.27: Ejemplo de dos conjuntos

Note que existe una intersección entre los conjuntos A y B, en notación de conjuntos este conjunto es $A \cap B$ y la probabilidad de estos elementos es $2/7$.

Ahora queremos determinar la probabilidad de que ocurra un elemento del conjunto A dado que solo tengo solamente elementos del conjunto B, la cual denotaremos por $P(A|B)$. Aplicando nuestra definición podemos ver que

$$P(A|B) = \frac{n\{A \cap B\}}{n\{B\}}$$

si dividimos esta ecuación entre el total de elementos del espacio muestral tenemos

$$P(A|B) = \frac{n\{A \cap B\}/N}{n\{B\}/N} = \frac{P(A \cap B)}{P(B)}$$

Para nuestros datos tenemos

$$P(A|B) = \frac{2}{5}$$

Por otro lado la probabilidad de $P(B|A) = 1/2$. Note que estos valores siguen cumpliendo nuestra definición de probabilidad muestral.

Dado lo anterior, podemos definir la probabilidad condicional de B dado A, que se denota $P(B|A)$ com

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

7.11.3. Independencia

Dos eventos A y B son independientes si y solo si $P(B|A) = P(B)$ y $P(A|B) = P(A)$, también se cumple que $P(A \cap B) = P(A)P(B)$.

7.11.4. Regla de Bayes

Si los eventos $B_1, B_2, ..., B_k$ constituyen una partición del espacio muestral S, donde $P(B_i) \neq 0$, entonces, para cualquier evento A de S

$$P(B_r|A) = \frac{P(B_r)P(A|B_r)}{\sum_{i=1}^{k} P(B_i)P(A|B_i)}$$

ver [**?**]

7.11.5. Estimador de máxima verosimilitud.

Para encontrar el estimador de máxima verosimilitud f, dado un conjunto de observaciones g, suponemos que estos tienen una distribución de probabilidad normal con varianza $\sigma^2 = 1/2$, la cual se expresa como:

$$p(g_i|f_i) = \frac{1}{C_1} e^{-(f_i - g_i)^2}$$

considerando que los datos son generados de manera independiente tenemos que la distribución de probabilidades conjunta es

$$p(g|f) = \prod_{i=0}^{N-1} \frac{1}{C_1} e^{-(f_i - g_i)^2}$$

Note que la probabilidad conjunta es máxima cuando $g_i = f_i$. Queremos calcular una señal suave, por lo que, nuestra información a priori será

$$p(f) = \prod_{i=0}^{N-1} \frac{1}{C_2} e^{-\lambda \nabla^2 f_i} = \prod_{i=0}^{N-1} \frac{1}{C_2} e^{-\lambda (f_i - f_{i-1})^2}$$

aplicando la regla de Bayes, encontramos la expresión para la probabilidad a posteriori dada por:

$$p(f|g) = \frac{p(g|f) * p(f)}{p(g)} \tag{7.43}$$

$$p(f|g) = \prod_{i=0}^{N-1} \frac{1}{C} e^{-\left[(f_i - g_i)^2 + \lambda (f_i - f_{i-1})^2\right]}$$

$$p(f|g) = \prod_{i=0}^{N-1} \frac{1}{C} e^{-U(f)}$$

La ecuación 7.43 es la distribución de probabilidad a posteriori, el mínimo de esta función, lo encontramos cuando la función de energía $U(f)$ es mínima. Nuestro problema lo traducimos en calcular el mínimo de la siguiente función:

$$U(f) = \sum_{i=0}^{N-1} \left[(f[i] - g[i])^2 + \lambda \left(f[i] - f[i-1] \right)^2 \right]$$

la cual representa la energía potencial almacenada en un sistema de resortes acoplados, tal como se muestra en la figura 7.28.

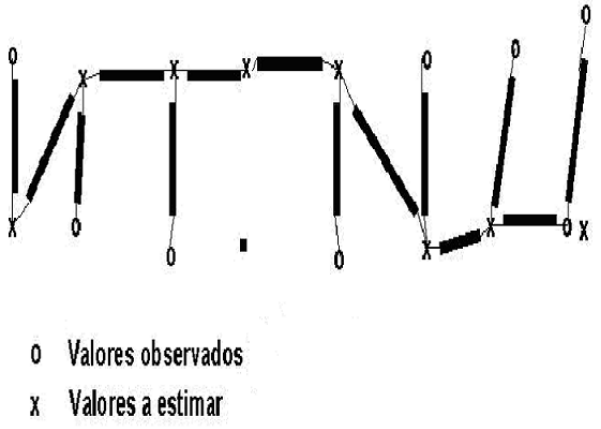


Figura 7.28: Sistema de resortes equivalente al filtro de membrana

Para calcular el valor de f que minimiza la función $U(f)$, hacemos $\frac{\partial U(f)}{\partial f[i]} = 0$

$$\frac{\partial U(f)}{\partial f[i]} = 2\left(f[i] - g[i]\right) + 2\lambda\left(f[i] - f[i-1]\right) - 2\lambda\left(f[i] - f[i]\right) = 0 \qquad (7.44)$$

lo cual nos da el siguiente sistema lineal de ecuaciones :

$$\begin{bmatrix} 1+\lambda & -\lambda & & & & & & \\ -\lambda & 1+2\lambda & -\lambda & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & -\lambda & 1+2\lambda & -\lambda & & & \\ & & & -\lambda & 1+2\lambda & -\lambda & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & -\lambda & 1+2\lambda & -\lambda \\ & & & & & & -\lambda & 1+\lambda \end{bmatrix} \begin{bmatrix} f[0] \\ f[1] \\ \vdots \\ f[i] \\ f[i+1] \\ \vdots \\ f[N-2] \\ f[N-1] \end{bmatrix} = \begin{bmatrix} g[0] \\ g[1] \\ \vdots \\ g[i] \\ g[i+1] \\ \vdots \\ g[N-2] \\ g[N-1] \end{bmatrix}$$

7.11.6. Interpretación del filtro de Membrana en el dominio de la Frecuencia.

Tomando la transformada discreta de Fourier a ambos miembros de la ecuación 7.45 y agrupando los términos obtenemos:

$$F[k] + \lambda\left(F[k] - F[k]e^{-\left(\frac{2\pi}{N}\right)k}\right) - \lambda\left(F[k]e^{\left(\frac{2\pi}{N}\right)k} - F[k]\right) = G[k]$$

agrupando términos obtenemos

$$\left[1 + 2\lambda - \lambda e^{-\left(\frac{2\pi}{N}\right)k} - \lambda e^{\left(\frac{2\pi}{N}\right)k}\right] F[k] = G[k]$$
$$\left[1 + 2\lambda - 2\lambda\cos\left(\frac{2\pi}{N}k\right)\right] F[k] = G[k]$$

finalmente tenemos que

$$F[k] = \frac{1}{\left[1 + 2\lambda - 2\lambda\cos\left(\frac{2\pi}{N}k\right)\right]} G[k] = H[k]G[k]$$
$$H[k] = \frac{1}{\left[1 + 2\lambda - 2\lambda\cos\left(\frac{2\pi}{N}k\right)\right]}$$

donde la función $H[k]$ es un filtro pasa bajas equivalente al filtro de membrana. Note que cuando el valor de λ aumenta tenemos que el ancho de banda se disminuye, en las figuras 7.29 y 7.30 podemos ver la respuesta a la frecuencia de este filtro para valores de λ 1 y 1000 respectivamente.

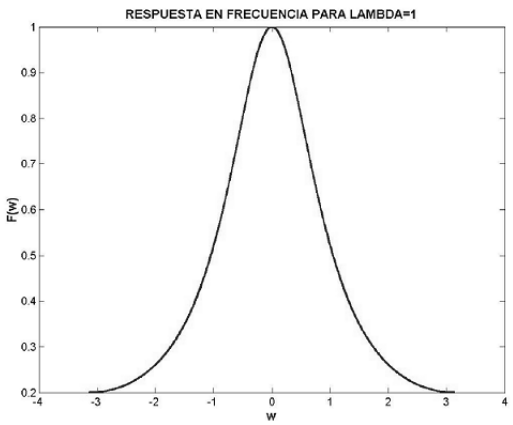


Figura 7.29: Respuesta a la frecuencia para el filtro de membrana con $\lambda = 1$

En dos dimensiones tenemos

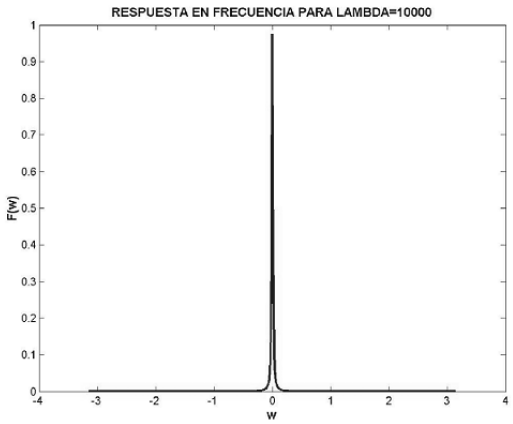


Figura 7.30: Respuesta a la frecuencia para el filtro de membrana con $\lambda = 1000$

$$U(f) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \left[(f[n,m] - g[n,m])^2 + \lambda \left(f[n,m] - f[n-1,m] \right)^2 + \lambda \left(f[n,m] - f[n,m-1] \right)^2 \right]$$

La derivada es

$$\frac{\partial U(f)}{\partial f[n,m]} = 2 \left(f[n,m] - g[n,m] \right) +$$
$$2\lambda \left(f[n,m] - f[n-1,m] \right) - 2\lambda \left(f[n+1,m] - f[n,m] \right)$$
$$2\lambda \left(f[n,m] - f[n,m-1] \right) - 2\lambda \left(f[n,m] - f[n,m+1] \right) = 0$$

La Transformada Discreta de Fourier queda

$$\left[1 + 4\lambda - \lambda e^{\left(\frac{2\pi}{N}\right)k} - \lambda e^{-\left(\frac{2\pi}{M}\right)k} - \lambda e^{\left(\frac{2\pi}{M}\right)l} - \lambda e^{\left(-\frac{2\pi}{M}\right)l} \right] F[k,l] = G[k,l]$$
$$\left[1 + 4\lambda - 2\lambda \cos\left(\frac{2\pi}{N}k\right) - 2\lambda \cos\left(\frac{2\pi}{M}l\right) \right] F[k,l] = G[k,l]$$

El Filtro en el dominio de Fourier queda

$$H[k,l] = \frac{1}{\left[1 + 4\lambda - 2\lambda \cos\left(\frac{2\pi}{N}k\right) - 2\lambda \cos\left(\frac{2\pi}{M}l\right) \right]}$$

Resultado: f
Dadas g, y y λ ;
Calcular $G = TDF(g)$;
para $k \leftarrow 0$ **a** $N-1$ **hacer**
  para $l \leftarrow 0$ **a** $M-1$ **hacer**
    Calcular $F[k,l] = \frac{G[k,l]}{1+4\lambda-2\lambda cos(2\pi k/N)-2\lambda cos(2\pi l/M)}$;
  fin
fin
Calcular $f = TDF^{-1}(F)$;

Algoritmo 6: Filtro de Membrana

Aplicaciones

8.1. Registro

Previo al problema de registro tenemos el problema de calcular una imagen destino I_2 a partir de una imagen origen I_1 y una transformación P.

La transformación afín P la definimos como

$$P = \begin{bmatrix} p_0 & p_1 & p_2 \\ p_3 & p_4 & p_5 \\ 0 & 0 & 1 \end{bmatrix}$$

y podemos calcular las coordenadas de Transformación afín haciendo el producto

$$\begin{bmatrix} \hat{c}(P) \\ \hat{r}(P) \\ 1 \end{bmatrix} = \begin{bmatrix} p_0 & p_1 & p_2 \\ p_3 & p_4 & p_5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c \\ r \\ 1 \end{bmatrix}$$

En forma compacta podemos escribir

$$\begin{aligned} \hat{c}(P) &= cp_0 + rp_1 + p_2 \\ \hat{r}(P) &= cp_3 + rp_4 + p_5 \end{aligned} \tag{8.45}$$

donde el vector de parámetros de la transformación esta definido por $P = [p_0, p_1, p_2, p_3, p_4, p_5]$ y la imagen de destino estará dada como $I_2(\hat{r}, \hat{c}) = I(r, c)$ para todos los píxeles de la imagen.

El Problema de registro

El problema de registro es inverso al problema de calcular la imagen destino I_2, utilizando un transformación afín. Ahora nos dan un par de imágenes la origen I_1 y la destino I_2 y

hay calcular el vector de parámetros de la transformación afín P. Por lo tanto la incógnita es P.

Podemos plantear el problema como un problema de mínimos cuadrados donde tenemos una función de error $E(P)$ que depende del vector P. Entonces debe existir un valor de P tal que haga que la función de error sea cero o un valor suficientemente pequeño. La función de error que proponemos es:

$$E(P) = \sum_{r=0}^{N_r} \sum_{c=0}^{N_c} (I_1(r,c) - I_2(\hat{r}(P), \hat{c}(P)))^2 \tag{8.46}$$

donde los valores de $\hat{r}$ y $\hat{c}$ se calculan mediante (8.45).

La manera de minimizar (8.46) es utilizando los métodos de Newton. Dos de los métodos que vamos a analizar son el método Gauss-Newton y el método de Levembert Marquart. Para ambos es necesario calcular el Gradiente de la función de error $\nabla E(P)$ y hacer una estimación del Hessiano $\nabla^2 E(P)$.

En el mínimo de la función $P^\star$ el Gradiente de la función debe valer cero para que la función $E(p)$ este en un mínimo.

$$\nabla E(P) = \begin{bmatrix} \frac{\partial E(P)}{\partial p_0} \\ \frac{\partial E(P)}{\partial p_1} \\ \frac{\partial E(P)}{\partial p_2} \\ \frac{\partial E(P)}{\partial p_3} \\ \frac{\partial E(P)}{\partial p_4} \\ \frac{\partial E(P)}{\partial p_5} \end{bmatrix}$$

Para determinar los valores del gradiente comenzaremos por calcular el k-ésimo termino del Gradiente $\frac{\partial E(P)}{\partial p_k}$:

$$\frac{\partial E(P)}{\partial p_k} = \frac{\partial \sum_{r=0}^{N_r} \sum_{c=0}^{N_c} (I_2(\hat{r}(P), \hat{c}(P)) - I(r,c))^2}{\partial p_k}$$

$$\frac{\partial E(P)}{\partial p_k} = 2 \sum_{r=0}^{N_r} \sum_{c=0}^{N_c} (I_2(\hat{r}(P), \hat{c}(P)) - I(r,c)) \frac{\partial I_2(\hat{r}(P), \hat{c}(P))}{\partial p_k}$$

Note que todos los términos del gradiente no cambiaran, excepto en el cálculo de la derivada de la imagen destino con respecto a cada parámetro. Para calcular la derivada de la imagen aplicamos la regla de la cadena

$$\frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_k} = \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial \hat{r}(P)}\frac{\partial \hat{r}(P)}{\partial p_k} + \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial \hat{c}(P)}\frac{\partial \hat{c}(P)}{\partial p_k}$$

$$\frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_k} = \left[\frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial \hat{r}(P)}, \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial \hat{c}(P)}\right] \begin{bmatrix} \frac{\partial \hat{r}(P)}{\partial p_k} \\ \frac{\partial \hat{c}(P)}{\partial p_k} \end{bmatrix}$$

$$\frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_k} = \left[\frac{\partial \hat{r}(P)}{\partial p_k}, \frac{\partial \hat{c}(P)}{\partial p_k}\right] \nabla I_2((\hat{r}(P),\hat{c}(P))$$

Para cada uno de los parámetros de la transformación afín la derivada de la imagen destino queda como

$$\frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_0} = [r,0]\,\nabla I_2((\hat{r}(P),\hat{c}(P))$$

$$\frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_1} = [c,0]\,\nabla I_2((\hat{r}(P),\hat{c}(P))$$

$$\frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_2} = [1,0]\,\nabla I_2((\hat{r}(P),\hat{c}(P))$$

$$\frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_3} = [0,r]\,\nabla I_2((\hat{r}(P),\hat{c}(P))$$

$$\frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_4} = [0,c]\,\nabla I_2((\hat{r}(P),\hat{c}(P))$$

$$\frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_5} = [0,1]\,\nabla I_2((\hat{r}(P),\hat{c}(P))$$

Podemos escribir en forma abreviada

$$J^T(P,r,c) = \begin{bmatrix} \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_0} \\ \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_1} \\ \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_2} \\ \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_3} \\ \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_4} \\ \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial p_5} \end{bmatrix} = \begin{bmatrix} c & 0 \\ r & 0 \\ 1 & 0 \\ 0 & c \\ 0 & r \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial \hat{r}(P)} \\ \frac{\partial I_2(\hat{r}(P),\hat{c}(P))}{\partial \hat{c}(P)} \end{bmatrix}$$

en forma compacta podemos escribir

$$J^T(P,r,c) = M(r,c)\nabla I_2(\hat{r}(P),\hat{c}(P))$$

donde $J(P,r,c)$ es el Jacobiano de la función de error, $M(r,c)$ es la matriz de coordenadas y $\nabla I_2(\hat{r}(P),\hat{c}(P))$ es el gradiente de la imagen destino para una transformación dada.

El vector gradiente lo podemos expresar como

$$\nabla E(P) = \begin{bmatrix} \frac{\partial E(P)}{\partial p_0} \\ \frac{\partial E(P)}{\partial p_1} \\ \frac{\partial E(P)}{\partial p_2} \\ \frac{\partial E(P)}{\partial p_3} \\ \frac{\partial E(P)}{\partial p_4} \\ \frac{\partial E(P)}{\partial p_5} \end{bmatrix} = 2\sum_{r=0}^{N_r}\sum_{c=0}^{N_c}(I2(\hat{r}(P),\hat{c}(P)) - I1(r,c))J^T(P,r,c) \tag{8.47}$$

Método de Gauss-Newton

Para este método utilizaremos el gradiente dado por (8.47) y la aproximación del Hessiano con

$$H(P) = 2\sum_{r=0}^{N_r}\sum_{c=0}^{N_c} J^T(P,r,c)J(P,r,c)$$

Para la iteración del método resolvemos el sistema de ecuaciones para $\delta^{(k)}$ dado como

$$H(P^{(k)})\delta^{(k)} = -\nabla E(P^{(k)})$$

y las actualizaciones del vector de parámetros se hace mediante

$$P^{(k+1)} = P^{(k)} + \delta^{(k)}$$

Método de Levenberg-Marquardt

Para el método de Levenberg-Marquardt se resuelve el sistema de ecuaciones

$$[H(P^{(k)}) + \mu]\delta^{(k)} = \nabla E(P^{(k)})$$

donde la μ se elige de tal suerte que la matriz $[H(P^{(k)}) + \mu]$ sea una matriz definida positiva en cada iteración.

Entonces dada una transformación inicial $P^{(0)}$ los métodos calculan una sucesión $P^{(1)}, P^{(2)}, ...P^{*}$ resolviendo un sistema de ecuaciones.

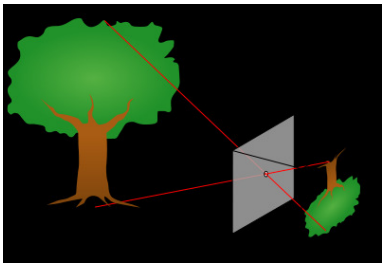
Como valor inicial es conveniente dar la identidad así $P^{(0)} = [p_0^{(0)}, p_1^{(0)}, p_2^{(0)}, p_3^{(0)}, p_4^{(0)}, p_5^{(0)}] = [1, 0, 0, 0, 1, 0]$

Calibración de Camáras

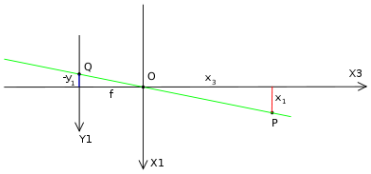
En esta sección daremos el modelo de una cámara y la manera de calcular el módelo de la misma. El modelo esta diseñando principalmente para sensores como CCD pero también es aplicable a otras cámaras.

9.1. El modelo de cámara pinhole

Consideremos el punto central de proyección del espacio en un plano. Centraremos la proyección del espacio Euclidiano de coordenadas y consideraremos que el plano es $Z = f$, el cual es llamado el plano focal de la cámara. Bajo el modelo de camara pinhole, un punto de coordenadas en el espacio $X = [X, Y, Z]^T$ es mapeado en el punto del plano de la imagen donde las lineas se unen en el centro de proyeción. Esto puede mostrarse por triángulos similares en las figuras 9.1(a) y 9.1(b).



(a) ***Tres dimensiones***



(b) ***dos dimensiones***

Figura 9.1: Modelo de la cámara de pinhole

En la figura, por triangulos similares podemos ver que el punto $[X, Y, Z]^T$ es mapeado a un punto en el plano de la cámara como $[fX/Z, fY/Z, f]^T$. Esto lo podems escribir de forma matricial como

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

esta matriz puede ser escrita como $\text{diag}(f, f, 1)[I|0]$ así la matriz de proyección para una cámara con modelo pinhole es $P = \text{diag}(f, f, 1)[I|0]$. La expresión anterior asume que el origen de coordenadas en el plano de la imagen es el punto principal. En la práctica no es así por lo cual debemos aplicar un mapeo

$$(X, Y, Z)^T| \rightarrow (fX/Z + p_x, fX/Z + p_y)^T$$

esta ecuación puede expresarse como

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{pmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

donde

$$K = \begin{pmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{pmatrix}$$

y $x = K[I|0]X_{\text{cam}}$ donde la matriz K es llamada la matriz de calibración de la cámara.

En general puntos en el espacio son expresados en términos de coordenadas euclidianas diferentes, conocidas como marco de coordenadas del mundo. Estos dos marco son relacionados por una rotación y una translación. Si $\tilde{X}$son las coordenadas homogeneas del mundo estas pueden ser representadas en el marco de la camara como como $\tilde{X}_{\text{cam}} = R\left(\tilde{X} - \tilde{C}\right)$, donde $\tilde{C}$ representa las coordenadas de la cámara centradas en las coordenadas del mundo. El modelo puede ser escrito en coordenadas homogenas como

$$x_{\text{cam}} = \begin{pmatrix} R & -R\tilde{C} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Poniendo todo junto tenemos $x = KR\left[I \middle| -\tilde{C}\right] X$ donde la matriz de la cámara es $P = K[R|t]$ donde $t = -R\tilde{C}$

9.2. Cámaras CCD

El modelo de cámara pinhole asume que las coordenadas de la imagen tienen igual escala en ambos ejes. En el caso de la cámaras CCD existe una posibilidad adicional de no tener pixeles cuadrados. Por lo tanto manejaremos en lugar de usar un factor de escala f en la diagonal consideraremos que existe un número de pixeles en la direción x llamado m_x y uno en la direción y llamado m_y por lo tanto nuestra matriz de parámetros queda como

$$K = \begin{pmatrix} fm_x & & x_0 \\ & fm_y & y_0 \\ & & 1 \end{pmatrix} = \begin{pmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{pmatrix}$$

Podemos consideran un modelo mas general si agregamos un parametro s al que denominaremos sesgo. En la mayoría de la cámaras normales este valor es cero, sin embargo lo consideraremos en el sentido de tener un modelo más general.

$$K = \begin{pmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{pmatrix}$$

El modelo de la cámara proyectiva es una matriz no cuadrada con tres renglones y cuatro columnas, por lo tanto el modelo tendrá 11 grados de libertad. Este modelo lo podemos escribir como $P = M\left[I\,|M^{-1}p_4\right] = KR\left[I\,\middle|-\tilde{C}\right]$donde p_4 es la última columna de la matriz P. El modelo completo es conocido como el módelo de la cámara proyetiva.

9.3. Propiedades de la cámara Proyectiva

Algunas propiedades de la cámara proyectiva son

- Centro de la camara. El centro de la cámara es el vector C el cual puede ser calculado para una cámara finita como el vector en el espacio nulo es decir PC=0. Esto se resuelve utilizando descomposición en Valores singulares y

- Puntos columna. Para i =1,2,3 los vectores columna $p_i$son los puntos de fuga en la imagen correpondientes a los ejes X, Y y Z. Columna p_4 es la imagen de las coordenadas del origen.

- Plano principal. El plano principal de la cámara is p^3, el último renglón de P.

- Ejes del plano. Los planos p^1y p^2 representan los planos a través del centro de la cámara.

- Punto principal. Punto de imagen $x_0 = Mm^3$es el punto principal de la cámar, donde m^{3T} es el tercer renglón de la matriz M.

- Rayo principal. El rayo principal de la camara es el rayo que pasa a través de la cámara con centro C con dirección m^{3T}. El eje principal es el vector $v = \det(M)m^3$es la dirección hacia el frente de la cámara.

9.4. Calibración de Cámaras a partir de puntos correspondientes

La calibración de cámaras la definiremos como el proceso de calcular los parámetros internos a partir de la observación de una imagen y sus puntos correspondiente en el mundo real. Comenzaremos por dada la matriz de proyección, calcular la matriz de parametros internos K, el centro de la cámara C y la matriz de rotación de la cámara $P = KR[I|C]$.

9.4.1. Centro de la Cámara

Podemos ver de lo expuesto en la sección anterior que el centro de la cámara dado la matriz de proyección se puede calcular como $PC = 0$ cuya solución es el vector con el menor eigenvalor una vez calculado la descomposición en valores singulares (espacio nulo).

9.4.2. Descomposición RQ

Definamos la matriz $M = KR$ la cual es una matrix de 3 renglones por 3 columnas. Para ello tendremos que hacer una factorización matricial. Recordemos que la matriz R puede calcularse como la multiplicación de tres matrices de rotación en cada eje como $R = R_z^T R_y^T R_x^T$ dado que todas estas matrices son ortogonales, entonces podemos escibir

$M = KR = KR_z^T R_y^T R_x^T$
$MR_x R_y R_z = K$

Las matrices de rotación estan definidas como

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_x & -s_x \\ 0 & s_x & c_x \end{pmatrix}, R_y = \begin{pmatrix} c_y & 0 & s_y \\ 0 & 1 & 0 \\ -s_y & 0 & c_y \end{pmatrix} \; y \; R_x = \begin{pmatrix} c_y & -s_z & 0 \\ s_z & 1 & 0 \\ 0 & 0 & c_y \end{pmatrix}$$

donde $c_i = \cos(\theta_i)$ y $s_i = \text{seno}(\theta_i)$

Dada la matriz M Los pasos son

1. Multiplicar $M' = MR_x$ y calcular c_x y $s_x$para que el elemento $M'_{3,2} = 0$ y $c_x^2 + s_x^2 = 1$. La solución es

$c_x = \frac{M_{3,3}}{\sqrt{M_{3,2}^2+M_{3,3}^2}} y s_x = -\frac{M_{3,2}}{\sqrt{M_{3,2}^2+M_{3,3}^2}}$
2. Multiplicar $M" = M'R_y$ y calcular c_y y $s_y$para que el elemento $M"_{3,1} = 0$ y $c_y^2 + s_y^2 = 1$. La solución es

$c_y = \frac{M_{3,3}}{\sqrt{M_{3,1}^2+M_{3,3}^2}} y s_y = -\frac{M_{3,1}}{\sqrt{M_{3,1}^2+M_{3,3}^2}}$
3. Finalmente multiplicar $K = M"R_z$ y calcular c_z y $s_z$para que el elemento $M"_{2,1} = 0$ y $c_z^2 + s_z^2 = 1$. La solución es

$c_z = -\frac{M_{2,2}}{\sqrt{M_{2,1}^2+M_{2,2}^2}} y s_z = \frac{M_{2,1}}{\sqrt{M_{2,1}^2+M_{2,2}^2}}$

ver calibra_camaras.java

9.4.3. Ejemplo

Dada la matriz de proyección de la cámara

$$P = \begin{pmatrix} 353.553 & 339.645 & 277.744 & -1.44946 \times 10^6 \\ -103.528 & 23.3212 & 459.607 & -632525. \\ 0.707107 & -0.353553 & 0.612372 & -918.559 \end{pmatrix}$$

determinar el centro de la cámara y la matriz de parámetros internos.

La matriz M de acuerdo con la definición es:

$$M = \begin{pmatrix} 353.553 & -103.528 & 0.707107 \\ 339.645 & 23.3212 & -0.353553 \\ 277.744 & 459.607 & 0.612372 \end{pmatrix}$$

Para encontrar el centro de la cámara hacemos la descomposición en Valores singulares de la matriz $[u, s, v] = svd(P)$ donde

$$u = \begin{pmatrix} -0.916531 & -0.399962 & -0.000424686 \\ -0.399962 & 0.916532 & -0.000479022 \\ -0.000580829 & 0.00026918 & 1. \end{pmatrix}$$

$$s = \begin{pmatrix} 1.58146 \times 10^6 & 0. & 0. & 0. \\ 0. & 406.369 & 0. & 0. \\ 0. & 0. & 0.837954 & 0. \end{pmatrix}$$

$$v = \begin{pmatrix} -0.000178718 & -0.581477 & 0.723847 & 0.371391 \\ -0.000202738 & -0.281691 & -0.607393 & 0.742782 \\ -0.000277203 & 0.76324 & 0.327293 & 0.557086 \\ 1. & 0.0000505429 & 0.0000969494 & 0.00037139 \end{pmatrix}$$

El vector v_4 es el que tiene el menor eigenvalue así que dividiendo este entre $v_{4,4}$ tenemos que el centro de la camára es $C = \{1000., 2000., 1500., 1.\}$

De la descomposición QR tenemos que la matriz M puede representarse como:

$$K \times R = \begin{pmatrix} 468.1648 & 91.2251 & 300.0001 \\ 0.0 & 427.201 & 199.9999 \\ 0.0 & 0.0 & 1.0 \end{pmatrix} \times \begin{pmatrix} 0.4138 & 0.9091 & 0.0471 \\ -0.5733 & 0.2201 & 0.7892 \\ 0.7071 & -0.3535 & 0.6124 \end{pmatrix}$$

9.4.4. Calculo del modelo utilizando puntos 3D - 2D

Así dado un conjunto de puntos en 3D $Q = \{Q_1, Q_2, ..., Q_N\}$ con $Q_i = (X_i, Y_i, Z_i)^T$ y otro conjunto de puntos correspondientes en el plano de la imagen $q = \{q_1, q_2, ..., q_N\}$ con $q_i = (x_i, y_i, w_i)^T$ podemos determinar el modelo de la cámara con al menos 6 puntos correspondientes $X_i \longleftrightarrow x_i$resolviendo

$$\begin{pmatrix} x_i \\ y_i \\ w_i \end{pmatrix} = \begin{pmatrix} P_{1,1} & P_{1,2} & P_{1,3} \\ P_{2,1} & P_{2,2} & P_{2,3} \\ P_{3,1} & P_{3,2} & P_{3,3} \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} Q_i^T P_1 \\ Q_i^T P_2 \\ Q_i^T P_3 \end{pmatrix}$$

Para resolver calculamos un vector paralelo, el cual tiene un producto vectorial igual a cero $q_i \times PQ_i = 0$. El sistema resultante lo podemos calcular encontrando el vector en el espacio nulo con descomposición en valores singulares. El sistema resultante a resolver es

$$q_i \times PQ_i = \begin{pmatrix} \hat{i} & \hat{j} & \hat{k} \\ x_i & y_i & w_i \\ Q_i^T P_1 & Q_i^T P_2 & Q_i^T P_3 \end{pmatrix} = \begin{pmatrix} y_i Q_i^T P_3 - w_i Q_i^T P_2 \\ x_i Q_i^T P_3 - w_i Q_i^T P_1 \\ x_i Q_i^T P_3 - w_i Q_i^T P_1 \end{pmatrix} = 0$$

de lo cual podemos desprender

$$\begin{pmatrix} 0 & -w_iQ_i^T & y_iQ_i^T \\ -w_iQ_i^T & 0 & x_iQ_i^T \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = AP = 0$$

9.4.5. Ejemplo

Dados los los puntos correpondientes Q = { [0.0, 0.0, 0.0], [100.0, 0.0, 0.0], [100.0, 100.0, 0.0], [0.0, 100.0 , 0.0], [0.0, 0.0, 100.0], [100.0, 0.0 100.0], [100.0, 100.0, 100.0], [0.0, 100.0, 100.0]} y q = {[1577.9715837523775, 688.6057400776651], [1667.8746657863205, 758.2462570249891], [1562.6523714350803, 725.2525691697816], [1483.8812040033367, 660.6388854847863], [1658.2870049496003, 684.1821822330893], [1762.4087684498734, 758.8465253032915], [1645.2811209801278, 723.3689600937458], [1554.5611061379307, 654.4720145727945]}, calcular

a) La matriz de proyección de la cámara,
b) La posición del centro de la cámara,
c) La matriz de Rotación y
d) la matriz K de parámetros internos.

$A_{1..6}$

$$\begin{pmatrix} 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & -99.9999 & 0. \\ 100. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & -99.9999 & -99.9999 \\ 100. & 100. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & -99.9999. \\ 0. & 100. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 100. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & -99.9999 & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. \\ 100. & 100. & 100. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. \end{pmatrix}$$

$A_{7..12}$

$$\begin{pmatrix} 0. & -0.9999 & 0. & 0. & 0. & 688.606 \\ 0. & 0. & 0. & 0. & 0. & -1577.97 \\ 0. & -0.9999 & 75824.6 & 0. & 0. & 758.246 \\ 0. & 0. & -166787. & 0. & 0. & -1667.87 \\ 0. & -0.9999 & 72525.3 & 72525.3 & 0. & 725.253 \\ 0. & 0. & -156265. & -156265. & 0. & -1562.65 \\ 0. & -0.9999 & 0. & 66063.9 & 0. & 660.639 \\ 0. & 0. & 0. & -148388. & 0. & -1483.88 \\ -99.9999 & -0.9999 & 0. & 0. & 68418.2 & 684.182 \\ 0. & 0. & 0. & 0. & -165829. & -1658.29 \\ -99.9999 & -0.9999 & 75884.7 & 0. & 75884.7 & 758.846 \\ 0. & 0. & -1.27487 \times 10^{10} & 0. & -1.27487 \times 10^{10} & -1.27487 \times 10^{6} \\ 0. & 0. & -164528. & -164528. & -164528. & -1645.28 \\ 0. & 0. & -1.01742 \times 10^{10} & -1.01742 \times 10^{10} & -1.01742 \times 10^{6} & \end{pmatrix}$$

a) Comenzamos por hacer la SVD. El último vector de la matriz v después de la descomposición es
$V_{12} = [2.2356E-4, 2.1476E-4, 1.7562E-4, -0.9165, -6.5463E-5, 1.4746E-5, 2.9062E-4, -0.3999, 4.4712E-7, -2.2356E-7, 3.8721E-7, -5.8082E-4]$
La matriz de proyección

$$P = \begin{bmatrix} 2.2356E-4 & 2.1476E-4 & 1.7562E-4 & -0.9165 \\ -6.5463E-5 & 1.4746E-5 & 2.9062E-4 & -0.3999 \\ 4.4712E-7 & -2.2356E-7 & 3.8721E-7 & -5.8082E-4 \end{bmatrix}$$

b) El centro de la cámara es
Para calcular el centro de la cámara calculamos svd de P. La matriz v es

$$P = \begin{bmatrix} -1.0E-4 & 0.5815 & -0.7237 & 0.3714 \\ -1.0E-4 & 0.2817 & 0.6074 & 0.7428 \\ -2.0E-4 & -0.7631 & -0.3272 & 0.5571 \\ 1.0 & 0.0 & 0.0 & 4.0E-4 \end{bmatrix}$$

El vector con el eigenvalor menor es $v_4 = [0.3714, 0.7128, 0.5571, 4.0E-4]$ por lo tanto el centro de la cámara es $C = [1000.0007, 2000.002, 1500.0003, 1.0]$

De la descomposición QR tenemos que la matriz M puede representarse como:

$$K \times R = \begin{pmatrix} 468.1648 & 91.2251 & 300.0001 \\ 0.0 & 427.201 & 199.9999 \\ 0.0 & 0.0 & 1.0 \end{pmatrix} \times \begin{pmatrix} 0.4138 & 0.9091 & 0.0471 \\ -0.5733 & 0.2201 & 0.7892 \\ 0.7071 & -0.3535 & 0.6124 \end{pmatrix}$$