Un sistema de comunicaciones

Julián Gutiérrez (51141) — Pablo Pauli (51185) — Ivan Itzcovich (53891) — Gustavo Del Giudice (51289)

Resumen

Este trabajo practico consiste en la estimación de un canal mediante cuadrados mínimos

1. Introducción

Un modelo muy simple de un sistema de comunicaciones consiste de un transmisor que envia un dato cada cierto tiempo.Los datos son modificados por el canal, es decir, por el medio donde son transmitidos.Esta modificacion se genera por la respuesta al impulso del canal. Además de ser modificados por el canal, los datos son afectados por ruido blanco Gaussiano aditivo. Teniendo en cuenta estas modificaciones, el receptor obtiene una respuesta. Lo que nos interesa es, usando la respuesta que obtiene el receptor, conseguir la señal que se envio al inicio de la comunicacion. El problema es que las modificaciones al canal y la longitud de la respuesta al impulso no suelen ser conocidas, por lo tanto deberemos estimarlos. Si la longitud es conocida, es posible estimar el canal utilizando cuadrados minimos. Para esto, hay que enviar una señal conocida por el receptor. A esto se lo denomida secuencia de entrenamiento.Luego, se estima el canal planteando otro problema de cuadrados minimos.

2. Desarrollo

2.1. Transformada de Fourier en dos dimensiones

La Transformada de Fourier puede ser generalizada a varias dimensiones. En nuestro caso, una imagen puede interpretarse como una señal de dos dimensiones, por tanto utilizaremos una generaliación bidimensional de la Transformada de Fourier sobre variables discretas (Ecuación **??**). Análogamente, para el proceso de anti-transformación utilizamos una generalización bidimensional de la anti-transformada sobre variables discretas (Ecuación **??**). Las funciones **??** y **??** del anexo muestran una simple implementación de las ecuaciones anteriores respectivamente. Sin embargo utilizando diferentes propiedades es posible realizar optimizaciones a las mismas para lograr una mayor eficiencia de cómputo. Es por esto que a fines prácticos utilizamos las funciones `fft2` y `ifft2` de *Octave* que calculan la Transformada Rápida de Fourier y la Antitransformada Rápida de Fourier respectivamente.

$$X_{l,k} = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x_{n,m} e^{-i\frac{2\pi}{N}(nl+mk)} \qquad (1)$$

$$x_{n,m} = \frac{1}{N^2}\sum_{l=0}^{N-1}\sum_{k=0}^{N-1} X_{l,k} e^{+i\frac{2\pi}{N}(nl+mk)} \qquad (2)$$

2.2. Imágenes en amplitud y fase

A modo de ejemplo tomamos una imagen arbitraria, en una escala de 256 grises, llamada `saturno` (**??**). A la misma le aplicamos la Transformada de Fourier en dos dimensiones y remapeando el resultado al intervalo entero $[0, 255]$ obtuvimos su representación en fase (**??**), al mismo tiempo aplicando módulo al resultado resultó su representación en amplitud (**??**). Para comprobar el proceso podríamos aplicar la Antitransformada de Fourier para obtener nuevamente la imagen original.

2.3. Filtros de imágenes utilizados

Un filtro es una función que opera contra la representación en fase de una señal, existiendo así diferentes tipos de filtros según su comportamiento. Los filtros pasa bajos son aquellos que eliminan las altas frecuencias y, en el campo de las imágenes, permiten suavizar una imagen, eliminar ruido y detalles pequeños de poco interés ya que sólo afecta a zonas con muchos cambios. Por otro lado, los filtros pasa alto, quienes eliminan las bajas frecuencias, intensifican detalles, bordes y cambios de alta frecuencia y atenúan las zonas de tonos uniformes.

Para aplicar un filtro a una imagen es necesario representarlo inicialmente como una matriz, como así también la imagen con la sobre la que se operará. Luego debemos aplicar la Transformada de Fourier a esta imagen, multiplicar elemento a elemento el resultado contra el filtro y, finalmente aplicar la Antitransformada de Fourier al resultado para recuperar la imagen filtrada. Es decir, los filtros se aplican en la frecuencia. Si x es la matriz que contiene a la imagen, X será su transformada y $xfil$ será la imagen con el filtro aplicado. Se denomina H al filtro y F a la Transformada Discreta de Fourier Bidimensional.

$$\begin{aligned} X &= F(x) \\ xfil &= F^{-1}(H * X) \end{aligned}$$

Para ejemplificar la utilización de los filtros implementamos tres diferentes y su definición se presenta a continaución:

-

$$H_{k,l} = \begin{cases} 0 & \text{si } 0 \leq k \leq 400, 190 \leq l \leq 210 \\ 0 & \text{si } 190 \leq k \leq 210, 0 \leq l \leq 400 \\ 1 & \text{en otro caso} \end{cases} \quad (3)$$

- Filtro Gaussiano

$$H_{k,l} = e^{-0,01(k^2+l^2)} \quad (4)$$

- El damero

$$H_{k,l} = \begin{cases} 0 & \text{si } l+k \text{ es par} \\ 1 & \text{si } l+k \text{ es par} \end{cases} \quad (5)$$

3. Resultados y conclusiones

Por lo observado pordemos concluir que realizar el filtrado digial de imágenes mediante la Transformada Discreta de Fourier bidimensional es de baja complejidad en su implementación. Al mismo tiempo proporciona flexibilidad en el diseño de soluciones de filtrado y rapidéz si se utiliza una primitiva eficiente como la que ofrece *Octave*.

4. Bibliografía

- Fierens, Pablo. *Guía 1*, Métodos Numéricos Avanzados. ITBA, 1er Cuatrimestre 2014

5. Anexo

Aquí se pueden ver las funciones de *GNU Octave* utilizadas para este análisis.

El *script* `initializeGlobals.m` inicializa las variables globadas utilizadas.

initializeGlobals.m

```
function  initializeGlobals()
global h;
global L;
L = 3;
ganancia = 1/10;
h = ganancia * (1+randn(L,1));%%this h is unknown, we want to estimate it.
endfunction
```

El *script* `TPE1.m` implementa la simulacion del sistema de comunicacion .

TPE1.m

```
function solution = TPE1()%the h used here is not global, we want to find it.
load initializeGlobals.m;
load train.m;
load transmit.m;
initializeGlobals();
global L;

img = imread('../img/lena512.bmp');
s = double(img(1,L));
M = columns(s);
S = toeplitz([zeros(1,M-L) s'],zeros(1,M));%%case M=L
r = train(S);
h = r/S;%TODO: find h with chol and backwards substitution.
%%%%%%%%%%%% Second Part %%%%%%%%%%%%%%%%%
a = double(img);
M = columns(a);
H = toeplitz([zeros(1,M-L) h'],zeros(1,M));
P = rows(a);
Rfinal = zeros()
Sfinal = [];
for k=1:rows(a)
r = transmit(a(k,:));
Rfinal = [Rfinal r];
s= r/H;
Sfinal = [Sfinal; s;];
endfor
Sfinal = uint8(Sfinal');
imshow(Sfinal);


endfunction
```

El *script* `train.m` implementa la secuencia de entrenamiento.

train.m

```
function r = train(S)
global h;
r = S * h;
endfunction
```

El *script* `transmit.m` simula la transmicion por el canal.

transmit.m

```
function r = transmit(s)
global h;
global L;
M = columns(s);
printf('transmit M = %d L = %d\n',M, L);
H = toeplitz([zeros(1,M-L) h'],zeros(1,M));%this is H original, using global h.
r = H*s;%+ ruido
endfunction
```