ANEXOS

Anexo 1: Nomenclatura para la identificación de anuncios

Describiremos en este apartado la nomenclatura empleada para nombrar los archivos de audio procedentes de diferentes emisoras de radio.

La nomenclatura adoptada es la siguiente:

xx_yy_zz_fs_nb_ppp

donde:

xx	Identificador de anuncio
yy	Identificador de emisora
zz	Identificador de versión
fs	Frecuencia de muestreo
nb	Número de bits
ppp	Tipo de procesado aplicado al anuncio

donde el valor del identificador de anuncio hace referencia a uno de los siguientes anuncios:

01	Cruzcampo Sevilla F.C.
02	Sangeplas
03	Pescados y mariscos Angelito
04	Centercit
05	Bacardi Limón
06	Fama
07	Good Energy
08	Showen Largo
09	Cuña Cadena 100
10	Arena Largo
11	00.com
12	Diario de Sevilla
13	ONCE
14	Instituto MAPFRE
15	Viajes Marsans
16	ONCE 2
17	Viajes Marsans 2
18	Cuña Cadena Dial
19	Señales horarias
20	Carrefour
21	Vanakinporta
22	Iceland
23	Hipercor
24	Natuka
25	La huerta de Burguillos
26	Hipoteca Naranja

27	11819 – Información
28	Carnet por puntos – Utilizar el móvil
29	ONCE – Senado
30	First Outlet
31	Desenfreno SMS Movistar
32	Permiso por puntos – Concurso
33	MAS Tomate rojo de Conil
34	Amena Globe Icon
35	Carnet por puntos – Velocidad
36	Localia
37	Hipoteca iBanesto
38	CCC – Ana Auxiliar de Enfermeria
39	CarGlass
40	Ifama Cocinas
41	Renault Servicios
42	Cuenta Ahora de Mediatix
43	Permiso por puntos – Conducir por ciudad
44	ABC
45	Grabación de duración: 64 minutos y 54 segundos
46	Grabación de duración: 30 minutos y 20 segundos

Por su parte, el valor del identificador de emisora hace referencia a una de las siguientes emisoras:

01	Radio Morón (SER)
02	Radio Giralda
03	Onda Melodía
04	R5 Todo Noticias (RNE)
05	Andalucía Información
06	Radio 1 (RNE)
07	Sevilla F.C. Radio
08	Radio Amanecer
09	Punto Radio
10	Canal .3
11	Radio Clásica (RNE)
12	Puerto Gelves
13	M80 Radio
14	Onda Cero Sevilla
15	Máxima FM
16	Radio Sevilla
17	Canal 7
18	Radio 3 (RNE)
19	Cadena 100
20	Kiss FM
21	Elite Radio
22	Radio Olé Sevilla
23	Cadena Dial Sevilla
24	Radio Sevilla 2
25	Canal Fiesta Radio
26	Canal Sur Radio
27	Rock & Gol

28 Onda Sevilla Radio
29 Onda San Pablo
30 Europa FM

Y el tipo de procesado aplicado al anuncio hace referencia a la distorsión que aplicamos sobre ciertos anuncios para realizar las pruebas de robustez del apartado 4.4. Si no aparece indicativo ninguno, significa que ese anuncio está en su formato original, es decir no se ha realizado sobre él ningún tipo de procesado:

mp1 codificación/decodificación mp3 con VBR nivel 10
mp2 codificación/decodificación mp3 con VBR nivel 50
mp3 codificación/decodificación mp3 con VBR nivel 100

wm1 codificación/decodificación Windows Media Audio 9.1 con CBR 5 Kbps, 8 KHz., 16 bits, mono
wm2 codificación/decodificación Windows Media Audio 9.1 con CBR 6 Kbps, 8 KHz., 16 bits, mono

ftp filtro pasa-todo: $H(z) = \frac{0.81z^2 - 1.64z + 1}{z^2 - 1.64z + 0.81}$

amp compresión de amplitud con la siguiente tasa de compresión: 8.94:1 para $|A| \geq$ -28.6 dB; 1.73:1 para -46.4 dB<|A|<-28.6 dB; 1:1.61 para $|A| \leq$ -46.4 dB.

ecu ecualizador de 10 bandas con los siguientes parámetros:

Freq. (Hz.)	31	62	125	250	500	1 K	2 K	4 K	8 K	16 K
Ganancia (dB)	-3	+3	-3	+3	-3	+3	-3	+3	-3	+3

fpb filtro paso-banda tipo Butterworth de segundo orden con frecuencias de corte de 100 Hz y 6000 Hz

te1 modificación de la escala temporal en un -2%
te2 modificación de la escala temporal en un +2%
te3 modificación de la escala temporal en un -4%
te4 modificación de la escala temporal en un +4%

ve1 cambio lineal de la velocidad en un -1%
ve2 cambio lineal de la velocidad en un +1%
ve3 cambio lineal de la velocidad en un -4%
ve4 cambio lineal de la velocidad en un +4%

Anexo 2: Códigos de Matlab

2.1. fingerprint.m

```
% Funcion principal para el calculo de un fingerprint

% Fecha de creacion: 10 de Julio de 2006

function F=fingerprint(audio)

Fs=8000;         % frecuencia de muestreo de la señal de audio
N=2960;          % numero de muestras por frame
overlap=31/32;   % factor de overlap
desplaz=floor(N-N*overlap); % desplazamiento de la ventana para
                            % cumplir con el factor de overlap
num_band=33;     % numero de bandas en que dividimos cada frame
f_inferior=300; % frecuencia inferior para la division en bandas
f_superior=2000; % frecuencia superior para la division en bandas

ventana=hanning(N);

% Dos opciones para solventar el problema de no ser el numero de
% muestras en la señal multiplo de N:

% 1º opcion: Rellenar con ceros el audio para que sea multiplo del
% numero de muestras por frame

%%sup=ceil(length(audio)/N);
%%relleno=sup*N-length(audio);
%%audio=[audio;zeros(relleno,1)];


% 2º opcion: Quedarnos con el numero entero de muestras de la señal
% mas alto posible que sea multiplo de 'desplaz'

numero_fingers=floor((length(audio)-N)/desplaz); % numero de huellas
                                                 % sin contar la primera
audio=audio(1:(N+numero_fingers*desplaz));

Eperanterior=zeros(num_band,1);     % energia de las bandas del frame
                                    % anterior
m=1;             % indices para guardar los bits de los fingerprints


% Calculo del vector para la division en 33 bandas separadas
% logaritmicamente entre 300 Hz y 2000 Hz

indice=logspace(log10(f_inferior),log10(f_superior),num_band+1);
indice=round(indice*N/Fs);

for k=0:desplaz:length(audio)-N
    frame=audio(k+1:k+N,1);
    frame=ventana.*frame;
    FRAME=abs(fft(frame));

    bandaSup=FRAME(indice(num_band):indice(num_band+1),1);
    for k=num_band:-1:2
        banda=FRAME(indice(k-1):indice(k),1);
```

```
        Eactual=calc_energia(banda);
        Esuperior=calc_energia(bandaSup);

        F(m,k-1)=bit_derivation(Eactual,Esuperior,Eperanterior(k-
                                                1,1),Eperanterior(k,1));


        % actualizamos variables
        bandaSup=banda;
        Eperanterior(k,1)=Esuperior;
        if k==2                 % obligatorio para evitar coger un
                                % Valor equivocado al llamar a
                                % "bit_derivation"
            Eperanterior(k-1,1)=Eactual;
        end
    end
    m=m+1;
end
```

2.2. calc_energ.m

```
% Calcula la energia de una banda
% Sub-funcion empleada para el calculo de un fingerprint

% Fecha de creacion: 10 de Julio de 2006

function E=calc_energ(x)

[M,N]=size(x);

if M>=N
    E=x'*x;
else
    E=x*x';
end
```

2.3. bit_derivation.m

```
% Calculo de un elemento de un fingerprint

% Parámetros de entrada:
%   e=energia de la banda
%   e_banterior=energia de la banda anterior
%   e_tanterior=energia de la banda en el periodo anterior
%   e_tb=energia de la banda posterior en el periordo anterior

% Fecha de creacion: 10 de Julio de 2006

function F=bit_derivation(e,e_banterior,e_tanterior,e_tb)

ED=e-e_banterior-(e_tanterior-e_tb);

if ED>0
    F=1;
else
    F=0;
end
```

2.4. busqueda.m

```
% Funcion que devuelve la distancia de Hamming entre los dos
% fingerprints que recibe como parámetros de entrada

% En las matrices de entrada van los fingerprints correspondientes al
% tiempo de deteccion de la señal de audio de entrada y el fingerprint
% de uno de los anuncios contenidos en la base de datos

% Fecha de creacion: 12 de Julio de 2006

function d=busqueda(detectado,anuncio)
[m1,n1]=size(detectado);
[m2,n2]=size(anuncio);

for k=1:m2-m1
    d(k)=sum(sum(abs(anuncio(k:k+m1-1,:)-detectado)));
end
```

2.5. probabilidad_eleccion_umbral.m

```
% Prueba para estudiar el umbral de decisión. El objetivo de esta
% prueba  es estudiar la probabilidad de falsa alarma en funcion del
% umbral de decision que tomemos.

% Fecha de creacion: 14 de Enero de 2006

audio=wavread('Anuncios\Procesado\45_11_01_8000_16');
audio=audio(1:length(audio)*3/4);

% Sobre el trozo de audio, escogemos fingerprints que no se solapen
% entre si, es decir, fingerprints independientes unos de otros. Para
% ello, escogemos trozos de 3 segundos al azar y eliminamos 6 segundos
% alrededor del trozo escogido para mantener la independencia entre
% fingerprints

Fs=8000;
trozo=24000;    % 24000 muestras correspondientes a 3 segundos
elimina=6*Fs/2;

for k=1:450

    h=round((length(audio)-trozo)*rand(1));% valor escogido al azar
                                           % para tomar un fingerprint
                                           % al azar
    fing=fingerprint(audio(h:h+trozo-1,1));
    finger(k,:)=reshape(fing',1,229*32);   % colocamos en forma de
                                % fila los bits correspondientes a los
                                % fingerprints de 3 segundos de audio
    if h>=trozo
        audio=[audio(1:h-
elimina,1);audio(h+trozo+elimina+1:length(audio),1)];
    else
        audio=[audio(1:h,1);audio(h+trozo+elimina+1:length(audio),1)];
    end
end

% Comparamos cada fingerprints de cada 3 segundos de audio con el
% resto y contabilizamos el numero de bits iguales

n=zeros(1,229*32);

for t=1:min(size(finger))-1
    for q=t+1:min(size(finger))
        dist=sum(abs(finger(t,:)-finger(q,:)));
        n(1,dist)=n(1,dist)+1;
    end
end

plot((1:229*32)/229/32,n/93961,'yx');

% Pintamos encima la PDF de una distribucion normal de media 0.5 y
% desviacion tipica 0.0148

hold on;
plot_dist_normal(0.5,0.0157);
```

2.6. estad.m

```
% Calcula la media y la desviacion tipica del vector de entrada

% Fecha de creacion: 19 de Enero de 2007

function [media, desv] = estad(x)


long=length(x);

media=sum([1:long]/long.*x)/sum(x);

desv=sqrt(sum((([1:long]/long-media).^2).*x)/sum(x));
```

2.7. prueba_eleccion_umbral_pdf.m

```
% Prueba de los algoritmos de extracion y busqueda de fingerprints.
% El objetivo es calcular la funcion densidad de probabilidad para
% detecciones correctas y para falsas alarmas

% Fecha de creacion del archivo: 17-01-2007

load BaseDatosFingerprints;
finger_audio=f_22_23_01_8000_8;

n=zeros(1,229*32);

for m=1:12

    switch m
        case 1, audio=wavread('Anuncios\Procesado\22_23_01_8000_8');
        case 2, audio=wavread('Anuncios\Procesado\22_23_02_8000_8');
        case 3, audio=wavread('Anuncios\Procesado\22_23_03_8000_8');
        case 4, audio=wavread('Anuncios\Procesado\22_23_04_8000_8');
        case 5, audio=wavread('Anuncios\Procesado\01_07_01_8000_8');
        case 6, audio=wavread('Anuncios\Procesado\08_19_01_8000_8');
        case 7, audio=wavread('Anuncios\Procesado\05_19_01_8000_8');
        case 8, audio=wavread('Anuncios\Procesado\05_23_01_8000_8');
        case 9, audio=wavread('Anuncios\Procesado\15_19_01_8000_8');
        case 10, audio=wavread('Anuncios\Procesado\16_19_01_8000_8');
        case 11, audio=wavread('Anuncios\Procesado\17_19_01_8000_8');
        case 12, audio=wavread('Anuncios\Procesado\18_23_01_8000_8');

    end

    for l=3:10003
        detectado=audio(l:l+24000-1,1);
        finger_detectado=fingerprint(detectado);
        distancia=busqueda(finger_detectado,finger_audio);
        resultado=min(distancia);
        n(1,resultado)=n(1,resultado)+1;
    end
end

plot((1:229*32)/229/32,n/sum(n),'x')
```

2.8. plot_distribuciones.m

```
% Dibuja todas las posibles distribuciones que mas se aproximan a la
% PDF de alarmas correctas

% Variables de entrada:
%   x1: vector que contiene las distancias entre fingerprints
%   x2: vector que contiene la cantidad de veces que se produce la
%       misma distancia entre fingerprints
%   m: tamaño de los fingerprints

% Fecha de creacion: 29 de Enero de 2006

function plot_distribuciones(x1,x2,m)

hold on; plot((1:229*32)/229/32,x2/sum(x2),'x')

hold on;

[a,b]=betafit(x1);
plot_dist_beta(a(1,1),a(1,2),m);

a=raylfit(x1);
plot_dist_rayleigh(a,m);

[a,b]=lognfit(x1);
plot_dist_lognormal(a(1,1),a(1,2),m);

[a,b]=wblfit(x1);
plot_dist_weibull(a(1,1),a(1,2),m);

xlabel('Tasa de error entre dos fingerprints');
ylabel('PDF');

hold off;
```

2.9. plot_dist_normal.m

```
% Dibuja la densidad de probabilidad de una distribucion normal con
% media 'med' y desviacion tipica 'sigma'

% Fecha de creacion: 21 de Enero de 2007

% El valor de entrada 'm' es el numero de bits por fingerprints

function plot_dist_normal(med,sigma,m)

aleat=round(m*(med+sigma*randn(1,10000000)));

aux=zeros(1,m);

for k=1:length(aleat)
    aux(1,aleat(1,k))=aux(1,aleat(1,k))+1;
end

plot((1:m)/m,aux/10000000,'r')
```

2.10. valor_umbral.m

```
% Calculo del umbral de decision T que iguala las dos areas por debajo
% de las funciones densidad de probabilidad de la distribucion normal
% y de la distribucion de Weibull

% Fecha de creación: 25 de Febrero de 2007

n=load('Resultados\prueba_eleccion_umbral_pdf_12anuncios.mat');
%variable de partida que contiene el resultado de la prueba
%prueba_eleccion_umbral_pdf

long=length(n.n);
corte=700;           % indice a partir del cual se discrimina entre
                     % valores para el calculo de Pf y Pn

% Seleccionamos la parte correspondiente a la probabilidad Pn
n1=n.n;
n1(corte:long)=zeros(1,long-corte+1);

% Seleccionamos la parte correspondiente a la probabilidad Pf
n2=n.n;
n2(1:corte)=zeros(1,corte);

% Transformamos n1 y n2 en vectores que me permitan el calculo de los
% parametros de las dos distribuciones consideradas (normal y weibull)

vect1=transforma_formato(n1);
vect2=transforma_formato(n2);

% Calculo de los parametros que mejor aproximan las distribuciones
[a,b]=wblfit(vect1);
[c,d]=normfit(vect2);

i=1;

% Calculo de las probabilidades Pf y Pn en funcion de alpha
for alpha=0.03:0.01:0.45
    prob1(i)=1-wblcdf(alpha,a(1,1),a(1,2));
    prob2(i)=normcdf(alpha,c,d);
    i=i+1;
end

plot([0.03:0.01:0.45],prob1);
hold on;
plot([0.03:0.01:0.45],prob2);
hold off;

figure
plot_dist_weibull(a(1,1),a(1,2),long);
hold on;
plot_dist_normal(c,d,long);
hold off;
```

2.11. transforma_formato.m

```
% Funcion que da como salida un vector a partir del cual se puede
% calcular la distribucion de probabilidad que me se asemeja.
% La entrada a esta funcion es un vector que contiene la funcion
% densidad de probabilidad resultado de alguna prueba

% Fecha de creacion: 24 de Enero de 2007

function dist=transforma_formato(x)

m=length(x);
aux=(1:m)/m;
incr=1;
for k=1:length(x)
    if x(k)~=0
        for l=1:x(k)
            dist(1,incr)=aux(k);
            incr=incr+1;
        end
    end
end
```

2.12. plot_dist_beta.m

```
% Dibuja densidad de probabilidad de una distribucion beta

% Fecha de creacion: 21 de Enero de 2007

% El valor de entrada 'm' es el numero de bits por fingerprints

function plot_dist_beta(a,b,m)

aleat=round(m*betarnd(a,b,1,1000000));

aux=zeros(1,m);

for k=1:length(aleat)
    if aleat(1,k)~=0
        aux(1,aleat(1,k))=aux(1,aleat(1,k))+1;
    end
end

plot((1:m)/m,aux/1000000,'r')
```

2.13. plot_dist_rayleigh.m

```
% Dibuja densidad de probabilidad de una distribucion rayleigh

% Fecha de creacion: 21 de Enero de 2007

% El valor de entrada 'm' es el numero de bits por fingerprints

function plot_dist_rayleigh(a,m)

aleat=round(m*raylrnd(a,1,1000000));

aux=zeros(1,m);

for k=1:length(aleat)
    if aleat(1,k)~=0
        aux(1,aleat(1,k))=aux(1,aleat(1,k))+1;
    end
end

plot((1:m)/m,aux/1000000,'g')
```

2.14. plot_dist_lognormal.m

```
% Dibuja densidad de probabilidad de una distribucion lognormal

% Fecha de creacion: 21 de Enero de 2007

% El valor de entrada 'm' es el numero de bits por fingerprints

function plot_dist_lognormal(media,sigma,m)

aleat=round(m*lognrnd(media,sigma,1,1000000));

aux=zeros(1,m);

for k=1:length(aleat)
    aux(1,aleat(1,k))=aux(1,aleat(1,k))+1;
end

plot((1:m)/m,aux/1000000,'m')
```

2.15. plot_dist_weibull.m

```
% Dibuja densidad de probabilidad de una distribucion weibull

% Fecha de creacion: 21 de Enero de 2007

% El valor de entrada 'm' es el numero de bits por fingerprints

function plot_dist_weibull(a,b,m)

aleat=round(m*wblrnd(a,b,1,1000000));

aux=zeros(1,m);

for k=1:length(aleat)
    if aleat(1,k)~=0
        aux(1,aleat(1,k))=aux(1,aleat(1,k))+1;
    end
end

plot((1:m)/m,aux/1000000,'k')
```

2.16. fingerprint_actualizable.m

```
% Funcion principal para el calculo de los fingerprints donde los
% parametros de entrada de la funcion son,precisamente, las
% caracteristicas con que queremos calcular los fingerprints

% PARAMETROS DE ENTRADA:
%  N           :numero de muestras por frame
%  overlap     :factor de overlap
%  num_band    :numero de bandas en que dividimos cada frame
%  f_inferior  :frecuencia inferior para la division en bandas
%  f_superior  :frecuencia superior para la division en bandas

% Fecha de creacion: 8 de Enero de 2007

function F=fingerprint(audio,N,overlap,num_band,f_inferior,f_superior)

Fs=8000;        % frecuencia de muestreo de la señal de audio
desplaz=floor(N-N*overlap); % desplazamiento de la ventana para
                            % cumplir  con el factor de overlap

ventana=hanning(N);

% Nos quedamos con el numero entero de muestras de la señal mas alto
% posible que sea multiplo de 'desplaz'
numero_fingers=floor((length(audio)-N)/desplaz); % numero de huellas
                                                 % sin contar la primera
audio=audio(1:(N+numero_fingers*desplaz));

Eperanterior=zeros(num_band,1);     % energia de las bandas del frame
                                    % anterior
m=1;        % indices para guardar los bits de los fingerprints
```

```
% Calculo del vector para la division en 'num_band' bandas separadas
% logaritmicamente entre la frecuencia inferior y la frecuencia
% superior escogidas para la division en bandas

indice=logspace(log10(f_inferior),log10(f_superior),num_band+1);
indice=round(indice*N/Fs);

for k=0:desplaz:length(audio)-N
    frame=audio(k+1:k+N,1);
    frame=ventana.*frame;
    FRAME=abs(fft(frame));

    bandaSup=FRAME(indice(num_band):indice(num_band+1),1);
    for k=num_band:-1:2
        banda=FRAME(indice(k-1):indice(k),1);

        Eactual=calc_energia(banda);
        Esuperior=calc_energia(bandaSup);

        F(m,k-1)=bit_derivation(Eactual,Esuperior,Eperanterior(k-
1,1),Eperanterior(k,1));


        % actualizamos variables
        bandaSup=banda;
        Eperanterior(k,1)=Esuperior;
        if k==2       % obligatorio para evitar coger un valor
                      % equivocado al llamar a "bit_derivation"
            Eperanterior(k-1,1)=Eactual;
        end
    end
    m=m+1;
end
```

2.17. prueba_N.m

```
% Prueba del algoritmo de fingerprints.
% En esta prueba vamos variando el numero de muestras por frame (N),
% es decir, el tamano de la ventana de Hamming.

% OBJETIVO: ver hasta que valor del tamaño de la ventana de Hamming
% podemos aumentar obteniendo un funcionamiento correcto del
% algoritmo. Al tener un mayor tamaño de la ventana, el numero de
% sub- fingerprints por cada 3 segundos de anuncio detectado sera
% menor, y, por tanto, el tiempo de computacion disminuira

% Fecha de creacion: 8 de Enero de 2007

% Caracteristicas de la prueba

% N=parametro variable

over_lap=31/32;
num_band=33;
finf=300;
fsup=2000;
alpha=0.18;              % umbral para considerar deteccion correcta.
                          % Calculado en la fase de analisis

time_detect=24000;       % numero de muestras correspondientes al
             % tiempo del trozo de audio que se coge. 24000 se
             % corresponden a 3segundos*8000muestras/segundo

fi=fopen('p_N_2960_1000_23960','w');    % abrimos un fichero para
                                          % almacenar los resultados

% Indices necesarios:
p=1;

for N=2960:1000:23960

 % Calculo de los fingerprints de los anuncios empleados en la prueba

 f_22_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_02_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_02_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_03_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_03_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_04_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_04_8000_8'),N,over_lap,num_band,finf,fsup);

 f_01_07_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \01_07_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_08_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \08_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_09_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \09_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
```

```
f_10_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \10_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_11_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \11_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_12_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \12_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_13_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \13_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_14_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \14_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_17_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \17_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_18_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \18_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_20_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \20_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

  % simulamos como si se cogiera un intervalo de tiempo de una
  % emision de radio y se analizara si se corresponde con algun
  % anuncio de la base de datos

   anuncio_detectado=wavread('Anuncios\Procesado\22_23_01_8000_8');

   t=1;
   s=1;
   key=1;
   for repeticion=1:1000     repeticion para cada valor de N

       l=round((length(anuncio_detectado)-time_detect)*rand(1));
% valor escogido al azar para tomar un trozo al azar del anuncio leido

       t1=cputime;

 finger_detectado=fingerprint_actualizable(anuncio_detectado(l:l+time_
                          detect-1,1),N,over_lap,num_band,finf,fsup);
       aux_times(repeticion)=cputime-t1;

 % La primera vez que entramos en el bucle, preparamos una variable,
 % 'probab', para almacenar las probabilidades de cada anuncio de ser
 % considerado como detectado

       if key==1
           key=0;
           [m,n]=size(finger_detectado);
           num_bits=m*n;    % Nº de bits que se comparan en la
                            % busqueda
           probab_alarm=zeros(1,m*n);
           probab_no_alarm=zeros(1,m*n);
       end

       j=1;
```

```
distancia(j)=min(busqueda(finger_detectado,f_22_23_01_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j); % almacena distancia en
                                      % deteccion correcta para calculo
                                      % de probabilidad de alarma
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_02_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_03_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_04_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_01_07_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j); % almacena distancia en
                                % deteccion correcta para calculo de
                                % probabilidad de falsa alarma
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_08_19_01_8000_8));

      probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+
      1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_09_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
```

```
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_10_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_11_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_12_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_13_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_14_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_17_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_18_23_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_20_23_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;

        [h,q]=min(distancia);
```

```
        if (h/num_bits<alpha)   % si el porcentaje de bits incorrectos
                                % es superior al alpha se considera que
                                % el trozo de audio que se esta
                                % estudiando no se corresponde con ningun
                                % anuncio de nuestra base de  anuncios
            fprintf(fi,'Anuncio detectado: ANUNCIO NUMERO %1.0f con
una probabilidad de deteccion correcta del %2.2f\n',q,(1-
h/num_bits)*100);
        else
            fprintf(fi,'NO SE HA DETECTADO NINGUN ANUNCIO. La
probabilidad de deteccion mas alta ha sido de %2.2f\n',(1-
h/num_bits)*100');
        end
    end

times(p)=mean(aux_times);     % Guardamos el tiempo empleado en
                        % calcular el finprint para cada valor de N
figure;
plot((1:m*n)/m/n,probab_alarm/sum(probab_alarm),'rx')
hold on;
plot((1:m*n)/m/n,probab_no_alarm/sum(probab_no_alarm),'yx')
plot((1:m*n)/m/n,zeros(1,m*n),'kx')
title(['N = ',num2str(N)]);
xlabel('Tasa de error de bit entre fingerprints');
ylabel('pdf')

% Añadimos una linea vertical que simula el umbral
plot([alpha alpha],[0 max(probab_no_alarm/sum(probab_no_alarm))])

hold off;

% Calculo de los parametros de la distribucion que mas se aproxima a
% cada uno de los dos casos:

% Para evitar problemas con 'wblfit'
if(min(dist_alarm(p,:))==0)
    dist_alarm(p,:)=dist_alarm(p,:)+10^(-6);
end

 [a,b]=wblfit(dist_alarm(p,:));
[c,d]=normfit(dist_no_alarm(p,:));

% Probabilidad de producirse una deteccion incorrecta

prob1(p)=1-wblcdf(alpha,a(1,1)/(m*n),a(1,2));
prob2(p)=normcdf(alpha,c/(m*n),d/(m*n));

fprintf(fi,'Tamaño de la ventana de Hamming: %1.0f muestras\n',N);
fprintf(fi,'Numero de bits por fingerprint detectado:
4.0fx%4.0f\n',m,n);
fprintf(fi,'Tiempo en calcular el fingerprint: %2.4f\n',times(p));
fprintf(fi,'Pn: %2.4e   Pf: %2.4e\n\n\n',prob1(p),prob2(p));

p=p+1;

end

% Dibujamos la relacion del tiempo de calculo del fingerprint con
% respecto al parametro analizado
```

```
plot(2960:1000:24000,times);

fi=fopen('p_N_2960_1000_23960','r');
type p_N_2960_1000_23960            % Abre el fichero creado
```

2.18. prueba_Overlap.m

```
% Prueba del algoritmo de fingerprints.
% En esta prueba vamos variando el overlap.

% OBJETIVO: ver hasta que valor del overlap podemos disminuir
% obteniendo un funcionamiento correcto del algoritmo.

% Fecha de creacion: 1 de Febrero de 2007

% Caracteristicas de la prueba

% Overlap=parametro variable

N=2960;
num_band=33;
finf=300;
fsup=2000;
alpha=0.18;          % umbral para considerar deteccion correcta

time_detect=24000;    % numero de muestras correspondientes al tiempo
                      % del trozo de audio que se coge. 24000 se
                       % corresponden a 3segundos*8000muestras/segundo

fi=fopen('p_Overlap','w');      % abrimos un fichero para almacenar los
                                % resultados

% Indices necesarios:
p=1;

for over_lap=[31/32 24/32 16/32 12/32 8/32]

 % Calculo de los fingerprints de los anuncios empleados en la prueba

 f_22_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_02_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_02_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_03_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_03_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_04_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_04_8000_8'),N,over_lap,num_band,finf,fsup);

 f_01_07_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \01_07_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_08_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \08_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_09_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \09_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_10_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \10_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
```

```
f_11_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \11_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_12_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \12_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_13_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \13_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_14_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \14_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_17_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \17_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_18_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \18_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_20_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \20_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

   % simulamos como si se cogiera un intervalo de tiempo de una
   % emision de radio y se analizara si se corresponde con algun
   % anuncio de la base de datos

    anuncio_detectado=wavread('Anuncios\Procesado\22_23_01_8000_8');

    t=1;
    s=1;
    key=1;
    for repeticion=1:1000     repeticion para cada valor de N

        l=round((length(anuncio_detectado)-time_detect)*rand(1));
% valor escogido al azar para tomar un trozo al azar del anuncio leido

        t1=cputime;


  finger_detectado=fingerprint_actualizable(anuncio_detectado(l:l+time_
                           detect-1,1),N,over_lap,num_band,finf,fsup);

        aux_times(repeticion)=cputime-t1;

   % La primera vez que entramos en el bucle, preparamos una variable,
   % 'probab', para almacenar las probabilidades de cada anuncio de
   % ser considerado como detectado

        if key==1
            key=0;
            [m,n]=size(finger_detectado);
            num_bits=m*n;      % Nº de bits que se comparan en la
                               % busqueda
            probab_alarm=zeros(1,m*n);
            probab_no_alarm=zeros(1,m*n);
        end

        j=1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_01_8000_8));
```

```
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);          % almacena distancia en
            % deteccion correcta para calculo de probabilidad de alarma
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_02_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_03_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_04_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_01_07_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j); % almacena distancia en
                              % deteccion correcta para calculo de
                              % probabilidad de falsa alarma
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_08_19_01_8000_8));

      probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+
      1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_09_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;
```

```
distancia(j)=min(busqueda(finger_detectado,f_10_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_11_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_12_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_13_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_14_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_17_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_18_23_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_20_23_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;

        [h,q]=min(distancia);

        if (h/num_bits<alpha)   % si el porcentaje de bits incorrectos
% es superior al alpha se considera que el trozo de audio que se esta
```

```
% estudiando no se corresponde con ningun anuncio de nuestra base de
% anuncios
            fprintf(fi,'Anuncio detectado: ANUNCIO NUMERO %1.0f con
una probabilidad de deteccion correcta del %2.2f\n',q,(1-
h/num_bits)*100);
        else
            fprintf(fi,'NO SE HA DETECTADO NINGUN ANUNCIO. La
probabilidad de deteccion mas alta ha sido de %2.2f\n',(1-
h/num_bits)*100');
        end

    end

times(p)=mean(aux_times);     % Guardamos el tiempo empleado en
            % calcular el finprint para cada valor del overlap


    end

figure;
plot((1:m*n)/m/n,probab_alarm/sum(probab_alarm),'rx')
hold on;
plot((1:m*n)/m/n,probab_no_alarm/sum(probab_no_alarm),'yx')
plot((1:m*n)/m/n,zeros(1,m*n),'kx')
title(['Overlap = ',num2str(over_lap)]);
xlabel('Tasa de error de bit entre fingerprints');
ylabel('pdf');

%Añade una linea vertical que simula el umbral
plot([alpha alpha],[0 max(probab_no_alarm/sum(probab_no_alarm))])


hold off;

% Para evitar problemas con 'wblfit'
if(min(dist_alarm(p,:))==0)
    dist_alarm(p,:)=dist_alarm(p,:)+10^(-6);
end

% Calculo de los parametros de la distribucion que mas se aproxima a
% cada uno de los dos casos:
[a,b]=wblfit(dist_alarm(p,:));
[c,d]=normfit(dist_no_alarm(p,:));

% Probabilidad de producirse una deteccion incorrecta

prob1(p)=1-wblcdf(alpha,a(1,1)/(m*n),a(1,2));
prob2(p)=normcdf(alpha,c/(m*n),d/(m*n));


fprintf(fi,'Factor de overlap: %1.4f \n',over_lap);
fprintf(fi,'Numero de bits por fingerprint detectado:
%4.0fx%4.0f\n',m,n);
fprintf(fi,'Tiempo en calcular el fingerprint: %2.4f\n',times(p));
fprintf(fi,'Pn: %2.4e   Pf: %2.4e\n\n\n',prob1(p),prob2(p));

p=p+1;

end
```

```
% Dibujamos la relacion del tiempo de calculo del fingerprint con
% respecto al parametro analizado

plot(times);

fi=fopen('p_Overlap','r');
type p_Overlap
```

2.19. prueba_NumBand.m

```
% Prueba del algoritmo de fingerprints.
% En esta prueba vamos variando el numero de bandas

% Fecha de creacion: 3 de Febrero de 2007

% Caracteristicas de la prueba

% num_band=parametro variable

N=2960;
over_lap=31/32;
finf=300;
fsup=2000;
alpha=0.18;                  % umbral para considerar deteccion
correcta

time_detect=24000;           % numero de muestras correspondientes al
                  % tiempo del trozo de audio que se coge. 24000 se
                  % corresponden a 3segundos*8000muestras/segundo

fi=fopen('p_NumBand','w');      %abrimos un fichero para almacenar los
                                % resultados

% Indices necesarios:
p=1;

for num_band=[33 17 9]

  % Calculo de los fingerprints de los anuncios empleados en la prueba

 f_22_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_02_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_02_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_03_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_03_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_04_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_04_8000_8'),N,over_lap,num_band,finf,fsup);

 f_01_07_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \01_07_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_08_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \08_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_09_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \09_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_10_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \10_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_11_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \11_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
```

```
f_12_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \12_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_13_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \13_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_14_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \14_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_17_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \17_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_18_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \18_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_20_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \20_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

   % simulamos como si se cogiera un intervalo de tiempo de una
   % emision de radio y se analizara si se corresponde con algun
   % anuncio de la base de datos

    anuncio_detectado=wavread('Anuncios\Procesado\22_23_01_8000_8');

    t=1;
    s=1;
    key=1;
    for repeticion=1:1000     repeticion para cada valor de N

        l=round((length(anuncio_detectado)-time_detect)*rand(1));
% valor escogido al azar para tomar un trozo al azar del anuncio leido

        t1=cputime;


  finger_detectado=fingerprint_actualizable(anuncio_detectado(l:l+time_
                           detect-1,1),N,over_lap,num_band,finf,fsup);

        aux_times(repeticion)=cputime-t1;

   % La primera vez que entramos en el bucle, preparamos una variable,
   % 'probab', para almacenar las probabilidades de cada anuncio de
   % ser considerado como detectado

        if key==1
            key=0;
            [m,n]=size(finger_detectado);
            num_bits=m*n;   % Nº de bits que se comparan en la
                            % busqueda
            probab_alarm=zeros(1,m*n);
            probab_no_alarm=zeros(1,m*n);
        end

        j=1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_01_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
```

```
        end
        dist_alarm(p,t)=distancia(j);          % almacena distancia en
            % deteccion correcta para calculo de probabilidad de alarma
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_02_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_03_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_04_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_01_07_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j); % almacena distancia en
                              % deteccion correcta para calculo de
                              % probabilidad de falsa alarma
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_08_19_01_8000_8));

      probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+
      1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_09_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_10_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
```

```
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_11_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_12_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_13_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_14_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_17_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_18_23_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_20_23_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;

        [h,q]=min(distancia);

        if (h/num_bits<alpha)   % si el porcentaje de bits incorrectos
% es superior al alpha se considera que el trozo de audio que se esta
% estudiando no se corresponde con ningun anuncio de nuestra base de
% anuncios
```

```
            fprintf(fi,'Anuncio detectado: ANUNCIO NUMERO %1.0f con
una probabilidad de deteccion correcta del %2.2f\n',q,(1-
h/num_bits)*100);
        else
            fprintf(fi,'NO SE HA DETECTADO NINGUN ANUNCIO. La
probabilidad de deteccion mas alta ha sido de %2.2f\n',(1-
h/num_bits)*100');
        end

    end

times(p)=mean(aux_times);     % Guardamos el tiempo empleado en
           % calcular el finprint para cada valor del nº bandas

figure;
plot((1:m*n)/m/n,probab_alarm/sum(probab_alarm),'rx')
hold on;
plot((1:m*n)/m/n,probab_no_alarm/sum(probab_no_alarm),'yx')
plot((1:m*n)/m/n,zeros(1,m*n),'kx')
title(['Numero de bandas = ',num2str(num_band-1)]);
xlabel('Tasa de error de bit entre fingerprints');
ylabel('pdf');

%Añade una linea vertical que simula el umbral
plot([alpha alpha],[0 max(probab_no_alarm/sum(probab_no_alarm))])

hold off;

% Para evitar problemas con 'wblfit'
if(min(dist_alarm(p,:))==0)
    dist_alarm(p,:)=dist_alarm(p,:)+10^(-6);
end

% Calculo de los parametros de la distribucion que mas se aproxima a
% cada uno de los dos casos:
[a,b]=wblfit(dist_alarm(p,:));
[c,d]=normfit(dist_no_alarm(p,:));

% Probabilidad de producirse una deteccion incorrecta

prob1(p)=1-wblcdf(alpha,a(1,1)/(m*n),a(1,2));
prob2(p)=normcdf(alpha,c/(m*n),d/(m*n));

fprintf(fi,'Numero de bandas: %1.0f \n',num_band);
fprintf(fi,'Numero de bits por fingerprint detectado:
%4.0fx%4.0f\n',m,n);
fprintf(fi,'Tiempo en calcular el fingerprint: %2.4f\n',times(p));
fprintf(fi,'Pn: %2.4e   Pf: %2.4e\n\n\n',prob1(p),prob2(p));

p=p+1;

end

% Dibujamos la relacion del tiempo de calculo del fingerprint con
% respecto al parametro analizado

plot(times);

fi=fopen('p_NumBand','r');
type p_NumBand      % Abre el fichero creado
```

2.20. prueba_TimeDetect.m

```
% Prueba del algoritmo de fingerprints.
% En esta prueba vamos variando el tiempo de captura

% Fecha de creacion: 3 de Febrero de 2007

% Caracteristicas de la prueba

% time_detect=parametro variable

N=2960;
num_band=33;
over_lap=31/32;
finf=300;
fsup=2000;
alpha=0.18;                % umbral para considerar deteccion correcta

fi=fopen('p_TimeDetect','w');      %abrimos un fichero para almacenar
                                    % los resultados

% Indices necesarios:
p=1;

for time_detect=[24000 20000 16000 12000 8000 4000]  % 3, 2.5, 2, 1.5,
                                                % 1 y 0.5 segundos de captura

  % Calculo de los fingerprints de los anuncios empleados en la prueba

 f_22_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \22_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_02_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \22_23_02_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_03_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \22_23_03_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_04_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \22_23_04_8000_8'),N,over_lap,num_band,finf,fsup);

 f_01_07_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \01_07_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_08_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \08_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_09_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \09_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_10_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \10_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_11_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \11_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_12_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \12_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
```

```
f_13_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \13_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_14_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \14_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_17_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \17_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_18_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \18_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_20_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \20_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

   % simulamos como si se cogiera un intervalo de tiempo de una
   % emision de radio y se analizara si se corresponde con algun
   % anuncio de la base de datos

    anuncio_detectado=wavread('Anuncios\Procesado\22_23_01_8000_8');

    t=1;
    s=1;
    key=1;
    for repeticion=1:1000     % repeticion para cada valor de N

        l=round((length(anuncio_detectado)-time_detect)*rand(1));
% valor escogido al azar para tomar un trozo al azar del anuncio leido

        t1=cputime;

  finger_detectado=fingerprint_actualizable(anuncio_detectado(l:l+time_
                           detect-1,1),N,over_lap,num_band,finf,fsup);

        aux_times(repeticion)=cputime-t1;

    % La primera vez que entramos en el bucle, preparamos una
    % variable, 'probab', para almacenar las probabilidades de cada
    % anuncio de ser considerado como detectado

        if key==1
            key=0;
            [m,n]=size(finger_detectado);
            num_bits=m*n;   % Nº de bits que se comparan en la
                           % busqueda
            probab_alarm=zeros(1,m*n);
            probab_no_alarm=zeros(1,m*n);
        end

        j=1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_01_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);          % almacena distancia en
            % deteccion correcta para calculo de probabilidad de alarma
        t=t+1;
```

```
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_02_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_03_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_04_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_01_07_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j); % almacena distancia en
                                % deteccion correcta para calculo de
                                % probabilidad de falsa alarma
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_08_19_01_8000_8));

      probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+
      1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_09_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_10_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;
```

```
distancia(j)=min(busqueda(finger_detectado,f_11_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_12_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_13_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_14_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_17_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_18_23_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_20_23_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;

        [h,q]=min(distancia);

        if (h/num_bits<alpha)   % si el porcentaje de bits incorrectos
% es superior al alpha se considera que el trozo de audio que se esta
% estudiando no se corresponde con ningun anuncio de nuestra base de
% anuncios
            fprintf(fi,'Anuncio detectado: ANUNCIO NUMERO %1.0f con
una probabilidad de deteccion correcta del %2.2f\n',q,(1-
h/num_bits)*100);
        else
```

```
            fprintf(fi,'NO SE HA DETECTADO NINGUN ANUNCIO. La
probabilidad de deteccion mas alta ha sido de %2.2f\n',(1-
h/num_bits)*100');
        end

    end

times(p)=mean(aux_times);           % Guardamos el tiempo empleado en
                  % calcular el finprint para cada valor de t.deteccion
figure;
plot((1:m*n)/m/n,probab_alarm/sum(probab_alarm),'rx')
hold on;
plot((1:m*n)/m/n,probab_no_alarm/sum(probab_no_alarm),'yx')
plot((1:m*n)/m/n,zeros(1,m*n),'kx')
title(['Tiempo de deteccion = ',num2str(time_detect/8000),'
segundos']);
xlabel('Tasa de error de bit entre fingerprints');
ylabel('pdf');

% Añade una linea vertical que simula el umbral
plot([alpha alpha],[0 max(probab_no_alarm/sum(probab_no_alarm))])
hold off;

% Para evitar problemas con 'wblfit'
if(min(dist_alarm(p,:))==0)
    dist_alarm(p,:)=dist_alarm(p,:)+10^(-6);
end

% Calculo de los parametros de la distribucion que mas se aproxima a
% cada uno de los dos casos:
[a,b]=wblfit(dist_alarm(p,:));
[c,d]=normfit(dist_no_alarm(p,:));

% Probabilidad de producirse una deteccion incorrecta

prob1(p)=1-wblcdf(alpha,a(1,1)/(m*n),a(1,2));
prob2(p)=normcdf(alpha,c/(m*n),d/(m*n));


fprintf(fi,'Tiempo de captura: %1.0f segundos\n',time_detect/8000);
fprintf(fi,'Numero de bits por fingerprint detectado:
%4.0fx%4.0f\n',m,n);
fprintf(fi,'Tiempo en calcular el fingerprint: %2.4f\n',times(p));
fprintf(fi,'Pn: %2.4e   Pf: %2.4e\n\n\n',prob1(p),prob2(p));

p=p+1;

end

% Dibujamos la relacion del tiempo de calculo del fingerprint con
% respecto al parametro analizado

plot(times);

fi=fopen('p_TimeDetect','r');
type p_TimeDetect      % Abre el fichero creado
```

2.21. prueba_global_N.m

```
% Analisis de la probabilidad de error en deteccion escogiendo los
% valores mas optimos de cada uno de los parametros que intervienen en
% el calculo de los fingerprints.

% En esta prueba tomamos con punto de partida el valor optimo de N y
% vamos variando, primeramente, número de bandas y el tiempo de
% detección

%Fecha de creacion: 1 de Abril de 2007


% num_band variable
% time_detect variable

N=12960;     % valor optimo
over_lap=31/32;
finf=300;
fsup=2000;
alpha=0.18;                % umbral para considerar deteccion correcta


fi=fopen('p_global_N_numband_timedetect','w');     %abrimos un fichero
                                        % para almacenar los resultados

% Calculo de los fingerprints de los anuncios empleados en la prueba

p=1;

for num_band=[9 17 33]
    for time_detect=[16000 20000 24000]

 f_22_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \22_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_02_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \22_23_02_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_03_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \22_23_03_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_04_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \22_23_04_8000_8'),N,over_lap,num_band,finf,fsup);

 f_01_07_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \01_07_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_08_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \08_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_09_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \09_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_10_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \10_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_11_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \11_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
```

```
f_12_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \12_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_13_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \13_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_14_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \14_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_17_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \17_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_18_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \18_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_20_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \20_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

          % simulamos como si se cogiera un intervalo de tiempo de una
          % emision de radio y se analizara si se corresponde con
          % algun anuncio de la base de datos
      anuncio_detectado=wavread('Anuncios\Procesado\22_23_01_8000_8');

         t=1;
         s=1;
         key=1;

         for repeticion=1:1000      % repeticion para cada valor de N
                l=round((length(anuncio_detectado)-
time_detect)*rand(1));
% valor escogido al azar para tomar un trozo al azar del anuncio leido

                t1=cputime;


                finger_detectado=fingerprint_actualizable(anuncio_dete
                ctado(l:l+time_detect-
                1,1),N,over_lap,num_band,finf,fsup);

                aux_times(repeticion)=cputime-t1;

       % La primera vez que entramos en el bucle, preparamos una
       % variable, 'probab', para almacenar las probabilidades
       % de cada anuncio de ser considerado como detectado

                 if key==1
                    key=0;
                    [m,n]=size(finger_detectado);
                    num_bits=m*n;    % numero de bits que se comparan
                                     % en la busqueda
                    probab_alarm=zeros(1,m*n);
                    probab_no_alarm=zeros(1,m*n);
                 end

        j=1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_01_8000_8));
        if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
```

```
        end
        dist_alarm(p,t)=distancia(j);          % almacena distancia en
              deteccion correcta para calculo de probabilidad de alarma
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_02_8000_8));
        if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_03_8000_8));
        if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_04_8000_8));
        if distancia(j)~=0

probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_01_07_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j); % almacena distancia en
                              % deteccion correcta para calculo de
                              % probabilidad de falsa alarma
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_08_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_09_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_10_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_11_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
```

```
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_12_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_13_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_14_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_17_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_18_23_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_20_23_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;

        [h,q]=min(distancia);

        if (h/num_bits<alpha)    % si el porcentaje de bits incorrectos
% es superior al alpha se considera que el trozo de audio que se esta
% estudiando no se corresponde con ningun anuncio de nuestra base de
% anuncios
            fprintf(fi,'Anuncio detectado: ANUNCIO NUMERO %1.0f con
una probabilidad de deteccion correcta del %2.2f\n',q,(1-
h/num_bits)*100);
        else
            fprintf(fi,'NO SE HA DETECTADO NINGUN ANUNCIO. La
probabilidad de deteccion mas alta ha sido de %2.2f\n',(1-
h/num_bits)*100');
        end
         end

        times(p)=mean(aux_times);

        figure;
        plot((1:m*n)/m/n,probab_alarm/sum(probab_alarm),'rx')
        hold on;
```

```
        plot((1:m*n)/m/n,probab_no_alarm/sum(probab_no_alarm),'yx')
        plot((1:m*n)/m/n,zeros(1,m*n),'kx')
        title(['N = ',num2str(N),'  NumBand = ',num2str(num_band-1),'
                 DetectionTime = ',num2str(time_detect/8000),' seg.']);
        xlabel('Tasa de error de bit entre fingerprints');
        ylabel('pdf');

        %Añadimos una linea vertical que simula el umbral
        plot([alpha alpha],[0
max(probab_no_alarm/sum(probab_no_alarm))])
        hold off;

        % Para evitar problemas con 'wblfit'
        if(min(dist_alarm(p,:))==0)
            dist_alarm(p,:)=dist_alarm(p,:)+10^(-6);
        end

        % Calculo de los parametros de la distribucion que mas se
        % aproxima a cada uno de los dos casos:
        [a,b]=wblfit(dist_alarm(p,:));
        [c,d]=normfit(dist_no_alarm(p,:));

        % Probabilidad de producirse una deteccion incorrecta

        prob1(p)=1-wblcdf(alpha,a(1,1)/(m*n),a(1,2));
        prob2(p)=normcdf(alpha,c/(m*n),d/(m*n));

        fprintf(fi,'Tamaño de la ventana de Hanning: %1.0f
                                                    muestras\n',N);
        fprintf(fi,'Numero de bandas: %2.0f muestras\n',num_band-1);
        fprintf(fi,'Overlap: %1.4f muestras\n',over_lap);
        fprintf(fi,'Tiempo de captura: %1.0f
                                        segundos\n',time_detect/8000);
        fprintf(fi,'Tiempo en calcular el fingerprint: %2.4f   Numero
                                    de anuncios: %1.0f\n',times(p),j);
        fprintf(fi,'Numero de bits por fingerprint detectado: %3.0f
                                                    x%3.0f\n',m,n);
        fprintf(fi,'Pn: %2.4e   Pf: %2.4e\n\n\n',prob1(p),prob2(p));

        p=p+1;

    end
end

fi=fopen('p_global_N_numband_timedetect','r');
type p_global_N_numband_timedetect % Abre el fichero creado
```

2.22. prueba_global_NumBand.m

```
% Analisis de la probabilidad de error en deteccion escogiendo los
% valores mas optimos de cada uno de los parametros que intervienen en
% el calculo de los fingerprints.

% En esta prueba tomamos con punto de partida el valor optimo del
% numero de bandas espectrales y vamos variando, primeramente, el
% numero de muestras por frame y luego el tiempo de captura

%Fecha de creacion: 1 de Abril de 2007


% N variable
% time_detect variable

num_band=9;      % valor optimo
over_lap=31/32;
finf=300;
fsup=2000;
alpha=0.18;               % umbral para considerar deteccion correcta


fi=fopen('p_global_numband_N_timedetect','w');      %abrimos un fichero
                                        % para almacenar los resultados

% Calculo de los fingerprints de los anuncios empleados en la prueba

p=1;

for N=[11960:-1000:2960]
    for time_detect=[8000 12000 16000 20000 24000]

        if (N<time_detect)


 f_22_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_02_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_02_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_03_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_03_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_04_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_04_8000_8'),N,over_lap,num_band,finf,fsup);

 f_01_07_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \01_07_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_08_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \08_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_09_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \09_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_10_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \10_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
```

```
f_11_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \11_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_12_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \12_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_13_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \13_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_14_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \14_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_17_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \17_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_18_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \18_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

f_20_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                    \20_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

          % simulamos como si se cogiera un intervalo de tiempo de una
          % emision de radio y se analizara si se corresponde con
          algun
          % anuncio de la base de datos


    anuncio_detectado=wavread('Anuncios\Procesado\22_23_01_8000_8');

         t=1;
         s=1;
         key=1;

          for repeticion=1:1000      % repeticion para cada valor de N
               l=round((length(anuncio_detectado)-
time_detect)*rand(1));
% valor escogido al azar para tomar un trozo al azar del anuncio leido

               t1=cputime;

               finger_detectado=fingerprint_actualizable(anuncio_dete
               ctado(l:l+time_detect-
               1,1),N,over_lap,num_band,finf,fsup);

               aux_times(repeticion)=cputime-t1;

         % La primera vez que entramos en el bucle, preparamos una
         % variable, 'probab', para almacenar las probabilidades
         % de cada anuncio de ser considerado como detectado

               if key==1
                  key=0;
                  [m,n]=size(finger_detectado);
                  num_bits=m*n;     % numero de bits que se comparan
                                    % en la busqueda
                  probab_alarm=zeros(1,m*n);
                  probab_no_alarm=zeros(1,m*n);
               end
```

```
        j=1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_01_8000_8));
        if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);          % almacena distancia en
            % deteccion correcta para calculo de probabilidad de alarma
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_02_8000_8));
        if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_03_8000_8));
        if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_04_8000_8));
        if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_01_07_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j); % almacena distancia en
                            % deteccion correcta para calculo de
                            % probabilidad de falsa alarma
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_08_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_09_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_10_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;
```

```
distancia(j)=min(busqueda(finger_detectado,f_11_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_12_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_13_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;
distancia(j)=min(busqueda(finger_detectado,f_14_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_17_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_18_23_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_20_23_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;

        [h,q]=min(distancia);

        if (h/num_bits<alpha)   % si el porcentaje de bits incorrectos
% es superior al alpha se considera que el trozo de audio que se esta
% estudiando no se corresponde con ningun anuncio de nuestra base de
% anuncios
            fprintf(fi,'Anuncio detectado: ANUNCIO NUMERO %1.0f con
una probabilidad de deteccion correcta del %2.2f\n',q,(1-
h/num_bits)*100);
        else
            fprintf(fi,'NO SE HA DETECTADO NINGUN ANUNCIO. La
probabilidad de deteccion mas alta ha sido de %2.2f\n',(1-
h/num_bits)*100');
        end
          end

        times(p)=mean(aux_times);

        figure;
```

```
        plot((1:m*n)/m/n,probab_alarm/sum(probab_alarm),'rx')
        hold on;
        plot((1:m*n)/m/n,probab_no_alarm/sum(probab_no_alarm),'yx')
        plot((1:m*n)/m/n,zeros(1,m*n),'kx')
        title(['N = ',num2str(N),'  NumBand = ',num2str(num_band-1),'
                 DetectionTime = ',num2str(time_detect/8000),' seg.']);
        xlabel('Tasa de error de bit entre fingerprints');
        ylabel('pdf');

        %Añadimos una linea vertical que simula el umbral
        plot([alpha alpha],[0
max(probab_no_alarm/sum(probab_no_alarm))])
        hold off;

        % Para evitar problemas con 'wblfit'
        if(min(dist_alarm(p,:))==0)
            dist_alarm(p,:)=dist_alarm(p,:)+10^(-6);
        end

        % Calculo de los parametros de la distribucion que mas se
        % aproxima a cada uno de los dos casos:
        [a,b]=wblfit(dist_alarm(p,:));
        [c,d]=normfit(dist_no_alarm(p,:));

        % Probabilidad de producirse una deteccion incorrecta

        prob1(p)=1-wblcdf(alpha,a(1,1)/(m*n),a(1,2));
        prob2(p)=normcdf(alpha,c/(m*n),d/(m*n));

        fprintf(fi,'Tamaño de la ventana de Hanning: %1.0f
                                                    muestras\n',N);
        fprintf(fi,'Numero de bandas: %2.0f muestras\n',num_band-1);
        fprintf(fi,'Overlap: %1.4f muestras\n',over_lap);
        fprintf(fi,'Tiempo de captura: %1.1f
                                          segundos\n',time_detect/8000);
        fprintf(fi,'Tiempo en calcular el fingerprint: %2.4f   Numero
                                    de anuncios: %1.0f\n',times(p),j);
        fprintf(fi,'Numero de bits por fingerprint detectado: %3.0f
                                                    x%3.0f\n',m,n);
        fprintf(fi,'Pn: %2.4e   Pf: %2.4e\n\n\n',prob1(p),prob2(p));

        p=p+1;
        end
    end
end


fi=fopen('p_global_numband_N_timedetect','r');
type p_global_numband_N_timedetect
```

2.23. prueba_global_TimeDetect.m

```
% Analisis de la probabilidad de error en deteccion escogiendo los
% valores mas optimos de cada uno de los parametros que intervienen en
% el calculo de los fingerprints.

% En esta prueba tomamos con punto de partida el valor optimo del
% tiempo de captura y vamos variando, primeramente, el numero de
% muestras por frame y el numero de bantas

%Fecha de creacion: 1 de Abril de 2007


% N variable
% num_band variable

time_detect=8000;      % valor optimo: 1 segundo
over_lap=31/32;
finf=300;
fsup=2000;
alpha=0.18;            % umbral para considerar deteccion correcta


fi=fopen('p_global_timedetect_N_numband_v2','w');        %abrimos un
                              % fichero para almacenar los resultados

% Calculo de los fingerprints de los anuncios empleados en la prueba

p=1;

for N=7960:-1000:2960
    for num_band=[9 17 33]

 f_22_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_02_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_02_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_03_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_03_8000_8'),N,over_lap,num_band,finf,fsup);

 f_22_23_04_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \22_23_04_8000_8'),N,over_lap,num_band,finf,fsup);

 f_01_07_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \01_07_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_08_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \08_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_09_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \09_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_10_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \10_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_11_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \11_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
```

```
 f_12_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \12_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_13_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \13_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_14_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \14_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_17_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \17_19_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_18_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \18_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

 f_20_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \20_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

          % simulamos como si se cogiera un intervalo de tiempo de una
          % emision de radio y se analizara si se corresponde con
          algun
          % anuncio de la base de datos

      anuncio_detectado=wavread('Anuncios\Procesado\22_23_01_8000_8');

         t=1;
         s=1;
         key=1;

          for repeticion=1:1000      % repeticion para cada valor de N
                l=round((length(anuncio_detectado)-
time_detect)*rand(1));
% valor escogido al azar para tomar un trozo al azar del anuncio leido

                t1=cputime;

                finger_detectado=fingerprint_actualizable(anuncio_dete
                ctado(l:l+time_detect-
                1,1),N,over_lap,num_band,finf,fsup);

                aux_times(repeticion)=cputime-t1;

            % La primera vez que entramos en el bucle, preparamos una
            % variable, 'probab', para almacenar las probabilidades
            % de cada anuncio de ser considerado como detectado

                 if key==1
                    key=0;
                    [m,n]=size(finger_detectado);
                    num_bits=m*n;      %numero de bits que se comparan
                                                  % en la busqueda
                    probab_alarm=zeros(1,m*n);
                    probab_no_alarm=zeros(1,m*n);
                 end

        j=1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_01_8000_8));
        if distancia(j)~=0
```

```
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);          % almacena distancia en
            % deteccion correcta para calculo de probabilidad de alarma
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_02_8000_8));
        if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_03_8000_8));
        if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_04_8000_8));
        if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
        end
        dist_alarm(p,t)=distancia(j);
        t=t+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_01_07_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j); % almacena distancia en
                              % deteccion correcta para calculo de
                              % probabilidad de falsa alarma
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_08_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_09_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_10_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_11_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
```

```
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_12_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_13_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_14_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_17_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_18_23_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;
        j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_20_23_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
        dist_no_alarm(p,s)=distancia(j);
        s=s+1;

        [h,q]=min(distancia);

        if (h/num_bits<alpha)    % si el porcentaje de bits incorrectos
% es superior al alpha se considera que el trozo de audio que se esta
% estudiando no se corresponde con ningun anuncio de nuestra base de
% anuncios
            fprintf(fi,'Anuncio detectado: ANUNCIO NUMERO %1.0f con
una probabilidad de deteccion correcta del %2.2f\n',q,(1-
h/num_bits)*100);
        else
            fprintf(fi,'NO SE HA DETECTADO NINGUN ANUNCIO. La
probabilidad de deteccion mas alta ha sido de %2.2f\n',(1-
h/num_bits)*100');
        end

          end

        times(p)=mean(aux_times);


        figure;
        plot((1:m*n)/m/n,probab_alarm/sum(probab_alarm),'rx')
```

```
        hold on;
        plot((1:m*n)/m/n,probab_no_alarm/sum(probab_no_alarm),'yx')
        plot((1:m*n)/m/n,zeros(1,m*n),'kx')
        title(['N = ',num2str(N),'  NumBand = ',num2str(num_band-1),'
                 DetectionTime = ',num2str(time_detect/8000),' seg.']);
        xlabel('Tasa de error de bit entre fingerprints');
        ylabel('pdf');

        %Añadimos una linea vertical que simula el umbral
        plot([alpha alpha],[0
max(probab_no_alarm/sum(probab_no_alarm))])
        hold off;


        % Para evitar problemas con 'wblfit'
        if(min(dist_alarm(p,:))==0)
            dist_alarm(p,:)=dist_alarm(p,:)+10^(-6);
        end

        % Calculo de los parametros de la distribucion que mas se
        % aproxima a cada uno de los dos casos:
        [a,b]=wblfit(dist_alarm(p,:));
        [c,d]=normfit(dist_no_alarm(p,:));


        % Probabilidad de producirse una deteccion incorrecta

        prob1(p)=1-wblcdf(alpha,a(1,1)/(m*n),a(1,2));
        prob2(p)=normcdf(alpha,c/(m*n),d/(m*n));


        fprintf(fi,'Tamaño de la ventana de Hanning: %1.0f
                                                    muestras\n',N);
        fprintf(fi,'Numero de bandas: %2.0f muestras\n',num_band-1);
        fprintf(fi,'Overlap: %1.4f muestras\n',over_lap);
        fprintf(fi,'Tiempo de captura: %1.1f
                                        segundos\n',time_detect/8000);
        fprintf(fi,'Tiempo en calcular el fingerprint: %2.4f   Numero
                                    de anuncios: %1.0f\n',times(p),j);
        fprintf(fi,'Numero de bits por fingerprint detectado: %3.0f
                                                    x%3.0f\n',m,n);
        fprintf(fi,'Pn: %2.4e   Pf: %2.4e\n\n\n',prob1(p),prob2(p));

        p=p+1;

    end
end


fi=fopen('p_global_timedetect_N_numband_v2','r');
type p_global_timedetect_N_numband_v2
```

2.24. p_robustez_SNR.m

```
% Analisis de la dependencia de las probabilidades de falsa alarma y
% de no alarma en funcion de la relacion señal a ruido

% Fecha de creacion: 10 de Abril de 2007

N=4960;
num_band=33;
over_lap=31/32;
finf=300;
fsup=2000;
alpha=0.18;                % umbral para considerar deteccion correcta

time_detect=8000;  % numero de muestras correspondientes al tiempo del
                   % trozo de audio que se coge. 24000 se corresponden
                   % a 3segundos*8000muestras/segundo

% Calculo de los fingerprints de los anuncios para la base de datos

 f_22_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \22_23_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_22_23_02_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \22_23_02_8000_8'),N,over_lap,num_band,finf,fsup);
 f_22_23_03_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \22_23_03_8000_8'),N,over_lap,num_band,finf,fsup);
 f_22_23_04_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \22_23_04_8000_8'),N,over_lap,num_band,finf,fsup);
 f_01_07_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \01_07_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_08_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \08_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_09_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \09_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_10_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \10_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_11_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \11_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_12_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \12_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_13_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \13_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_14_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \14_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_17_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \17_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_18_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \18_23_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_20_23_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                      \20_23_01_8000_8'),N,over_lap,num_band,finf,fsup);

% Simulamos como si se cogiera un intervalo de tiempo de una emision
% de radio a la cual le sumamos un ruido blanco gaussiano.
% Luego se analiza si se corresponde con algun anuncio de la base de
% datos y las probabilidades de error (Pn y Pf)

anuncio_detectado=wavread('Anuncios\Procesado\22_23_01_8000_8');
energia=calc_energia(anuncio_detectado);
```

```
long=length(anuncio_detectado);

p=1;

for snr=5:60

    energia_ruido=energia*10^(-snr/10);
    amp_noise=sqrt(energia_ruido/long);
    ruido=amp_noise*randn(long,1);
    senal_captada=anuncio_detectado+ruido;

    t=1;
    s=1;
    key=1;
    for repeticion=1:1000
         l=round((length(senal_captada)-time_detect)*rand(1));

finger_detectado=fingerprint_actualizable(senal_captada(l:l+time_detec
                                  t-1,1),N,over_lap,num_band,finf,fsup);

         if key==1
            key=0;
            [m,n]=size(finger_detectado);
            num_bits=m*n;     %numero de bits que se comparan en la
                             % busqueda
            probab_alarm=zeros(1,m*n);
            probab_no_alarm=zeros(1,m*n);
         end

         j=1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_01_8000_8));
         if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
         end
         dist_alarm(p,t)=distancia(j);
         t=t+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_02_8000_8));
         if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
         end
         dist_alarm(p,t)=distancia(j);
         t=t+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_03_8000_8));
         if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
         end
         dist_alarm(p,t)=distancia(j);
         t=t+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_22_23_04_8000_8));
         if distancia(j)~=0
probab_alarm(1,distancia(j))=probab_alarm(1,distancia(j))+1;
         end
```

```
         dist_alarm(p,t)=distancia(j);
         t=t+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_01_07_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
         dist_no_alarm(p,s)=distancia(j);     % almacena distancia en
                              % deteccion correcta para calculo de
                              % probabilidad de falsa alarma
         s=s+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_08_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
         dist_no_alarm(p,s)=distancia(j);
         s=s+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_09_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
         dist_no_alarm(p,s)=distancia(j);
         s=s+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_10_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
         dist_no_alarm(p,s)=distancia(j);
         s=s+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_11_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
         dist_no_alarm(p,s)=distancia(j);
         s=s+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_12_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
         dist_no_alarm(p,s)=distancia(j);
         s=s+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_13_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
         dist_no_alarm(p,s)=distancia(j);
         s=s+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_14_19_01_8000_8));

probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
         dist_no_alarm(p,s)=distancia(j);
         s=s+1;
         j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_17_19_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
         dist_no_alarm(p,s)=distancia(j);
         s=s+1;
```

```
            j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_18_23_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
            dist_no_alarm(p,s)=distancia(j);
            s=s+1;
            j=j+1;

distancia(j)=min(busqueda(finger_detectado,f_20_23_01_8000_8));
probab_no_alarm(1,distancia(j))=probab_no_alarm(1,distancia(j))+1;
            dist_no_alarm(p,s)=distancia(j);
            s=s+1;

    end

    SNR(p)=snr;

    % Calculo de la BER entre fingerprints y su desviacion tipica
    ber_media(p)=(mean(dist_alarm(p,:)))/(m*n);
    ber_desv(p)=(std(dist_alarm(p,:)))/(m*n);

    % Dibujamos algunas PDFs
    if (snr==5 || snr==10 || snr==15|| snr==20 || snr==25 || snr==30)
        figure;
        plot((1:m*n)/m/n,probab_alarm/sum(probab_alarm),'rx')
        hold on;
        plot((1:m*n)/m/n,probab_no_alarm/sum(probab_no_alarm),'yx')
        plot((1:m*n)/m/n,zeros(1,m*n),'kx')
        %plot([alpha alpha],[0
max(probab_no_alarm/sum(probab_no_alarm))])
        title(['SNR (dB) =',num2str(snr)]);
        xlabel('Tasa de error de bit entre fingerprints');
        ylabel('pdf');
        hold off;
    end

    % Para evitar problemas con 'wblfit'
    if(min(dist_alarm(p,:))==0)
        dist_alarm(p,:)=dist_alarm(p,:)+10^(-6);
    end

    % Calculo de los parametros de la distribucion que mas se aproxima
    % a cada uno de los dos casos:
    [a,b]=wblfit(dist_alarm(p,:));
    [c,d]=normfit(dist_no_alarm(p,:));

    % Probabilidad de producirse una deteccion incorrecta

    prob1(p)=1-wblcdf(alpha,a(1,1)/(m*n),a(1,2));
    prob2(p)=normcdf(alpha,c/(m*n),d/(m*n));

    p=p+1;
end

% Dibujamos la relacion Pn con SNR
figure;
hold on;
plot(SNR,prob1);
```

```
xlabel('SNR (db)')

% Dibujamos la relacion Pf con SNR
plot(SNR,prob2,'r');

legend('Pn','Pf');

% Dibujamos el limite por encima del cual consideramos probabilidad de
% error permitida


hold off

% Dibujamos la relacion BER con SNR

figure;
plot(SNR,ber_media);
hold on
plot(SNR,ber_media+(ber_desv/2),'r');
plot(SNR,ber_media-(ber_desv/2),'r');
hold off
```

2.25. p_robustez_PROCESADO.m

```
% Analisis de la BER entre fingerprints de un anuncio con algun tipo
% de procesado y los fingerprints correspondientes a los anuncios
% originales

% Fecha de creacion: 16 de Abril de 2007

N=4960;
num_band=33;
over_lap=31/32;
finf=300;
fsup=2000;
% alpha=0.18;       % umbral para considerar deteccion correcta

time_detect=8000;

m=20;      % mxn es el tamaño de los fingerprints
n=32;

fi=fopen('p_robustez','a'); % fichero para guardar los resultados de
                            % todas las pruebas de robustez

% Calculo de los fingerprints correspondientes a los anuncios
% originales

 f_08_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \05_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_10_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \17_19_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_12_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \20_23_01_8000_8'),N,over_lap,num_band,finf,fsup);
 f_13_19_01_8000_8=fingerprint_actualizable(wavread('Anuncios\Procesado
                     \29_23_01_8000_8'),N,over_lap,num_band,finf,fsup);


% Calculo de los fingerprints correspondientes a la señal de audio que
% sufre algun tipo de procesado

anuncio_detectado1=wavread('Anuncios\Procesado\05_19_01_8000_8_wm2');
anuncio_detectado2=wavread('Anuncios\Procesado\17_19_01_8000_8_wm2');
anuncio_detectado3=wavread('Anuncios\Procesado\20_23_01_8000_8_wm2');
anuncio_detectado4=wavread('Anuncios\Procesado\29_23_01_8000_8_wm2');


% Para calcular la media de la BER y su desviacion típica,
% repetimos el proceso 1000 veces para cada anuncio
for j=1:1000

    l=round((length(anuncio_detectado1)-time_detect)*rand(1));
    finger_detectado1=fingerprint_actualizable(anuncio_detectado1(l:l+
                        time_detect-1,1),N,over_lap,num_band,finf,fsup);
    distancia1(j)=min(busqueda(finger_detectado1,f_08_19_01_8000_8));

    l=round((length(anuncio_detectado2)-time_detect)*rand(1));
    finger_detectado2=fingerprint_actualizable(anuncio_detectado2(l:l+
```

```
                    time_detect-1,1),N,over_lap,num_band,finf,fsup);
    distancia2(j)=min(busqueda(finger_detectado2,f_10_19_01_8000_8));

    l=round((length(anuncio_detectado3)-time_detect)*rand(1));
    finger_detectado3=fingerprint_actualizable(anuncio_detectado3(l:l+
                    time_detect-1,1),N,over_lap,num_band,finf,fsup);
    distancia3(j)=min(busqueda(finger_detectado3,f_12_19_01_8000_8));

    l=round((length(anuncio_detectado4)-time_detect)*rand(1));
    finger_detectado4=fingerprint_actualizable(anuncio_detectado4(l:l+
                    time_detect-1,1),N,over_lap,num_band,finf,fsup);
    distancia4(j)=min(busqueda(finger_detectado4,f_13_19_01_8000_8));


end

% Calculo de la BER entre fingerprints y su desviacion tipica
p=1;

ber_media(p)=mean(distancia1)/(m*n);
ber_desv(p)=std(distancia1)/(m*n);
ber_max(p)=max(distancia1)/(m*n);
p=p+1;
ber_media(p)=mean(distancia2)/(m*n);
ber_desv(p)=std(distancia2)/(m*n);
ber_max(p)=max(distancia2)/(m*n);
p=p+1;
ber_media(p)=mean(distancia3)/(m*n);
ber_desv(p)=std(distancia3)/(m*n);
ber_max(p)=max(distancia3)/(m*n);
p=p+1;
ber_media(p)=mean(distancia4)/(m*n);
ber_desv(p)=std(distancia4)/(m*n);
ber_max(p)=max(distancia4)/(m*n);

% Guardamos los resultados
fprintf(fi,'\n\nCodificacion/decodificacion WMA 9.1 con CBR 6
Kbps\n');
fprintf(fi,'Número de iteraciones: 1000\n');
fprintf(fi,'                                BER     Desv. Estandar  Ber
max.\n');
for i=1:4
    fprintf(fi,'Resultados del anuncio %1.0f:    %1.4f     %1.4f
%1.4f\n',i,ber_media(i),ber_desv(i),ber_max(i));
end

fi=fopen('p_robustez','r');
type p_robustez      % Abre el fichero creado
```