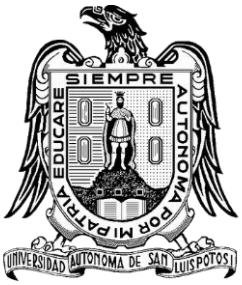
Universidad Autónoma de San Luis Potosí

Facultad de Ciencias

Ingeniería Biomédica



Implementación de algoritmo Pan-Tompkins para la detección de complejos QRS en MATLAB® y variación de parámetros.

Proyecto Final: Procesamiento Digital de Señales.

Profesor: PhD. Bersaín Reyes

Anguiano Piña Uriel 0253434

Castillo González Rodrigo 0252114

Chávez Duarte Daniel A. 0229526

Figueroa Govea Francisco S. 0250448

Mayo del 2018

Resumen

El electrocardiograma es el registro de la actividad eléctrica del corazón, su señal está conformada por ondas identificadas y denominadas complejo cardiaco (P, Q, R, S y T). Cada onda proporciona diferente información sobre el corazón. Los complejos QRS (picos R) representan la onda de mayor amplitud en la señal de electrocardiograma y el intervalo entre cada uno es empleado para el cálculo de la frecuencia cardiaca.

Para la detección de dichos picos, con base en el algoritmo Pan- Tompkins, se llevó a cabo un proceso de remuestreo, para posteriormente implementar un sistema de filtrado (filtro pasa-banda, filtro derivador, rectificación, filtro pasa-bajas) cuyo objetivo fue la atenuación de las ondas que no corresponden a los picos R, finalmente se ejecutó un proceso de umbralización para identificar los ya mencionados complejos QRS. El objetivo de este trabajo es evaluar el algoritmo implementado en términos de exactitud, a partir de la variación de parámetros en los distintos procesamientos aplicados a la señal de electrocardiograma (remuestreo, filtrado y umbralización).

Palabras clave: electrocardiograma, complejo QRS, re-muestreo, filtro pasa-bajas, filtro pasa-bandas, umbralización, exactitud.

Introducción

El electrocardiograma (ECG) es el registro de la actividad eléctrica del corazón, representa la suma algebraica de los potenciales desarrollados en las fibras miocárdicas. Se registra de manera indirecta o *derivada*, empleando una colocación de electrodos superficiales en puntos convencionales. Estos métodos convencionales en que se registran los la actividad eléctrica del corazón reciben el nombre de derivaciones [1].

El ECG suministra bastante información sobre el estado y funcionamiento del corazón. La señal obtenida del ECG posee una morfología particular para un corazón sano y las ondas que la componen han sido estudiadas y relacionadas a cada proceso fisiológico del corazón (Figura 1). Además, ciertas modificaciones a la forma característica han sido distinguidas y resultan útiles para el diagnóstico de enfermedades.

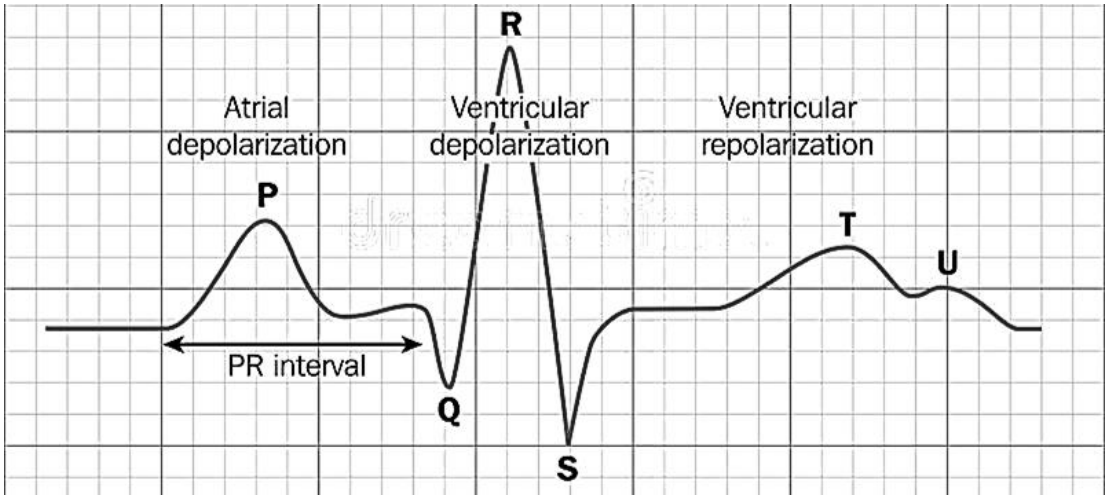


Figura 1. Representación ideal la señal ECG. Cada ciclo cardiaco compuesto de la onda P, un complejo QRS, la onda T y la onda U (casi nunca apreciable). En cada etapa se muestra su relación con los sucesos fisiológicos del corazón.

Uno de los aspectos de mayor importancia y aplicación en la práctica clínica es la identificación de complejos QRS (picos R) en tiempo real. Para ello en 1985 Jiapus Pan y Willis J. Tompkins propusieron el que en la actualidad es el más famoso y utilizado algoritmo para la detección de complejos QRS (Algoritmo Pan-Tompkins).

Sin embargo la detección del QRS es complicada, no solo por la variabilidad propia de la señal de ECG, sino también por el ruido presente en su adquisición. Este ruido está compuesto por ruido muscular, artefactos causados por el movimiento de electrodos, el ruido en la línea de alimentación, entre otros [2].

Una descripción general sobre el algoritmo Pan-Tompkins es la siguiente [2]:

- Como primer paso y con el objetivo reducir el ruido muscular y aquel derivado de la línea de alimentación se emplea un filtro pasa-bandas, compuesto por un filtro pasa-altas y un filtro pasa-bajas puestos en cascada. Las frecuencias de corte empleadas en el diseño implementado por Pan-Tompkins son 5 y 12 Hz.

- Después se aplica un filtro derivador a la señal para acentuar las pendientes más pronunciadas, ubicadas en el complejo QRS.
- Luego se rectifica la señal, ayudando a reducir la detección de falsos positivos. Esto se realiza obteniendo el cuadrado de la señal punto a punto, siendo además una amplificación no lineal.
- Un filtro pasa-bajas dado por un integrador por ventaneo genera una señal con información de la pendiente de la onda R. El algoritmo utiliza un ventaneo de 30 muestras, definido a partir del ancho del complejo QRS (150 ms).
- Por último se establecen dos umbrales adaptables, los cuales al ser sobrepasados indican la localización de los picos R. El umbral más alto se emplea para un primer análisis, el más bajo es usado si después de cierto intervalo de tiempo no se detecta un QRS.

En conclusión, el algoritmo Pan-Tompkins detecta la ubicación de los complejos QRS de una señal de ECG empleando información de su pendiente, su amplitud y su ancho. Al ser probado con bases de datos y en la práctica clínica se obtuvieron buenos resultados, de ahí su relevancia.

Objetivo

Como objetivo de este proyecto se plantea la implementación del algoritmo para detección de complejos QRS planteado por Pan-Tompkins en software Matlab®. También la variación de los parámetros de frecuencias y umbral de discriminación con el fin de hallar aquellos que brinden la mayor exactitud.

Metodología

Algoritmo

La implementación del algoritmo de detección de complejos QRS en Matlab® se realizó siguiendo las características del algoritmo original ya descritas. Sin embargo al no ser una detección en tiempo real se creó una etapa de remuestreo previo. Además modificaciones a los filtros facilitaron su elaboración por funciones propias del software. La figura 2 muestra el procesamiento general aplicado en el algoritmo implementado.

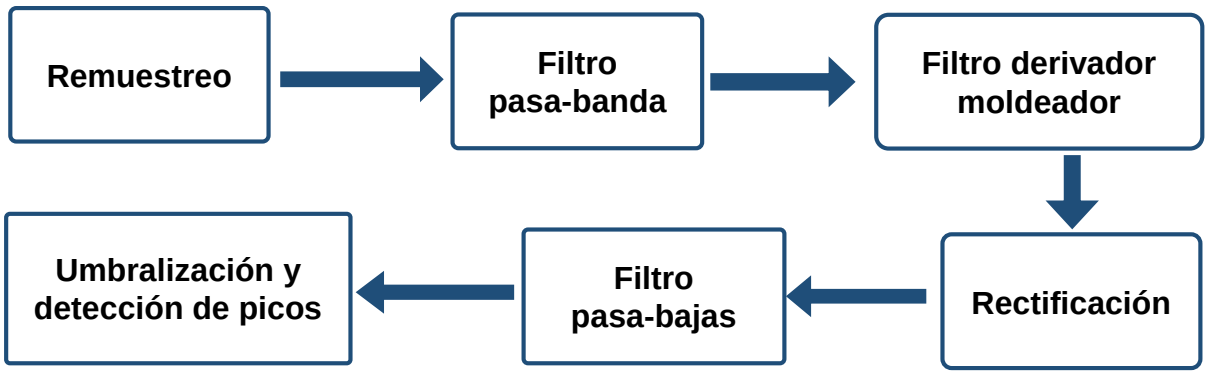


Figura 2. Diagrama de flujo del algoritmo de detección de complejos QRS implementado.

El **remuestreo** permite analizar el funcionamiento del algoritmo con señales ECG obtenidas a una frecuencia de muestreo muy alta, generando la misma señal de entrada con una menor cantidad de datos. Al analizar una menor cantidad de datos se reduce el costo computacional del procesamiento y se acelera el procedimiento. Sin embargo se debe contemplar la posibilidad de perder información de la señal con dicho remuestreo.

El **filtro pasa-banda** es implementado mediante un filtro FIR de orden 80 empleando la funciones '*fir1'* y '*filtfilt'* incluidas en Matlab®. Evidentemente un filtro de orden bajo reduciría la exactitud en la detección, sin embargo uno de mayor orden resultaría contraproducente en cuanto al costo computacional, siendo que arrojaría resultados casi iguales.

El **filtro derivador** corresponde simplemente a la aproximación de la derivada de la señal de entrada en cada punto dada por la siguiente ecuación:

$$\frac{d\,x[n]}{dt}\Big|_{t=f\cdot n} \approx x[n+1] - x[n]$$

Donde $x[n]$ es la señal discreta de entrada (resultado del primer filtro aplicado), t es el tiempo, f la frecuencia a la que fue remuestreada la señal.

La **rectificación** se realiza elevando la señal al cuadrado, como se muestra en la siguiente ecuación:

$$y[n] = x^2[n]$$

Donde $y[n]$ es la señal de salida ya rectificada y $x[n]$ es la señal de entrada (resultado de la previa derivación).

Un **filtro pasa-bajas** es implementado sustituyendo al integrador (utilizado por Pan y Tompkins), cumpliendo la misma función. Al igual que el pasa-bandas, es implementado como un filtro FIR de orden 80 utilizando las funciones '*fir1'* y '*filtfilt'* incluidas en el software.

Por último se tiene la **umbralización** y **detección de complejos QRS**. El umbral no se definió en términos de la amplitud de la señal, sino en un porcentaje de la máxima amplitud que esta posee. Esto con la finalidad de que no deba ser ajustado para señales que por diferentes causas poseen diferentes amplitudes. El umbral implementado da como resultado ventanas en las que se deben identificar los máximos en la amplitud de la señal. Existen dos alternativas: encontrar dichos máximos en la señal ECG, o encontrar en la señal resultante todo el procesamiento previo.

Para más detalles sobre la implementación y funcionamiento del algoritmo propuesto, revisar el Apéndice B.

Prueba y variación de parámetros

Las pruebas del algoritmo, en las que se evaluó la exactitud en la detección al variar parámetros de frecuencias y umbralización, se realizaron utilizando la concatenación de 30 segundos de señal ECG extraídos de registros de 23 pacientes contenidos en la base de datos *MIT-BIH Arrhythmia Database*. Los registros de ECG habían sido remuestreados a 1000 Hz y normalizados.

Los parámetros que no variados fueron:

- Frecuencias de corte de filtro pasa-bandas, fijadas en 5 y 12 Hz, así establecidas por Pan-Tompkins.
- El grado de los filtros FIR empleados, 80 como ya fue descrito.

Los parámetros modificados fueron:

- La frecuencia de remuestreo, con el objetivo de encontrar aquella que reduzca significativamente el número de datos sin comprometer la información de la señal.
- La frecuencia de corte del filtro pasa-bajas, buscando la que de la mayor exactitud en la detección.
- El umbral de discriminación, para identificar cuál reduce los falsos-positivos y los falsos-negativos aumentando la exactitud.

El primer caso analizado fue la variación de la frecuencia de remuestreo y el umbral de discriminación simultáneamente, manteniendo la frecuencia de corte del filtro pasa-bajas fija arbitrariamente en 5 Hz. La frecuencia fue variada en el intervalo de 150 a 450 Hz y el umbral entre 1% y 20%. Para cada combinación de estos dos parámetros fue calculada la exactitud en la detección.

El otro caso analizado fue la variación de la frecuencia de corte del filtro pasa-bajas y el umbral simultáneamente. La frecuencia fue variada en el intervalo de 1 a 100 Hz y nuevamente el umbral entre 1% y 20%. En base a los resultados obtenidos en el primer análisis se eligió un valor de frecuencia de remuestreo fija y de igual forma para cada combinación de parámetros se calculó la exactitud en la detección.

Resultados y Discusión

En el análisis de variación de frecuencia de remuestreo y umbral de discriminación se obtuvo que para frecuencias de remuestreo menores a 250 la exactitud de la detección se reduce notablemente (un máximo cercano a 0.8). Entre mayor sea esta frecuencia aumenta la exactitud, sin embargo a partir de la frecuencia de 300 Hz el aumento es mínimo, manteniéndose prácticamente constante (un máximo cercano a 0.97). En la forma en la que se planteó esta prueba, un resultado así era esperado. Se buscaba aquella frecuencia de remuestreo mínima que permita reducir el número de datos y no perdiera información de la señal.

En cuanto al umbral, valores menores al 2.5% generaron exactitudes muy bajas, incluso menores a 0.5, esto por la detección de muchos falsos-positivos. El valor óptimo se encontró entre 4% y 8%, pues para valores mayores a este la exactitud comienza a decrecer por falsos-negativos.

La figura 3 muestra los resultados obtenidos de esta prueba, donde los valores elegidos para maximizar la exactitud en la detección fueron: frecuencia de remuestreo a 350 Hz y umbral de discriminación al 4%.

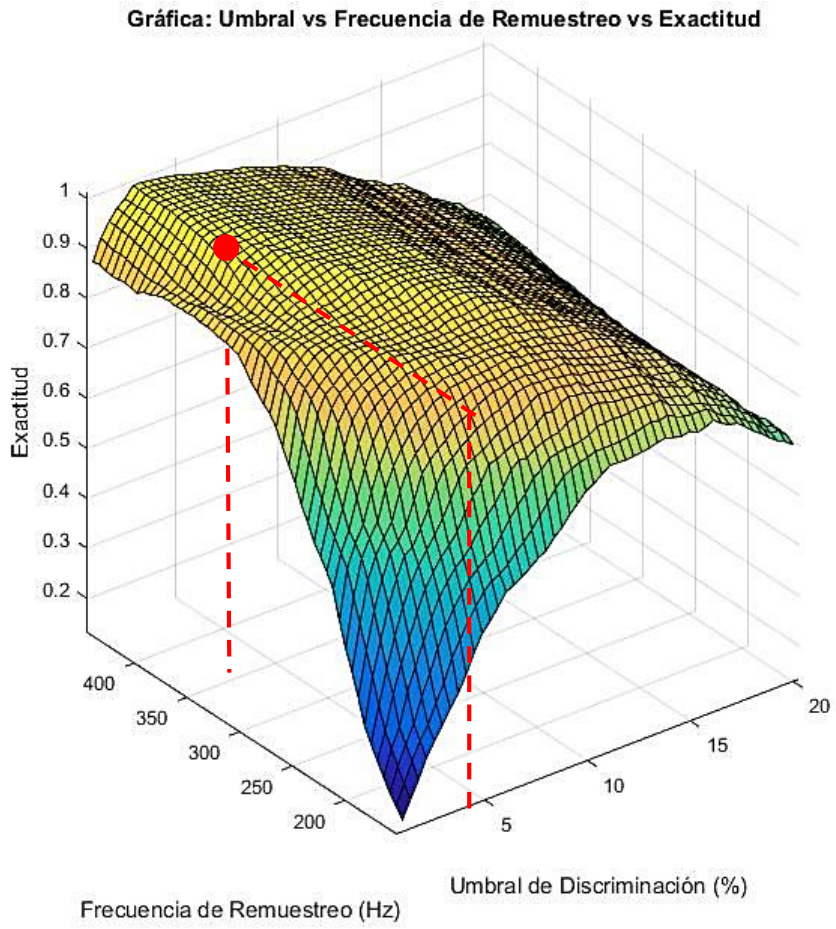


Figura 3. Gráfica: Umbral de discriminación vs. Frecuencia de remuestreo vs. Exactitud. En color rojo se señalan los valores seleccionados para maximizar la exactitud.

En base a los resultados obtenidos, para el análisis de variación de frecuencia de corte del filtro pasa-bajas y umbral de discriminación se fijó la frecuencia de remuestreo en 350 Hz.

Para frecuencia de corte bajas se obtuvo la mayor exactitud, a partir de los 35 Hz esta decrece con una pendiente muy pronunciada. En cuanto al umbral los resultados fueron los mismos que en la prueba anterior, entre 4% y 8% se obtiene valores de exactitud buenos.

La figura 4 muestra los resultados de la prueba, donde los valores elegidos para maximizar la exactitud fueron: frecuencia de corte 10 Hz y umbral de discriminación 4%. Una frecuencia de corte tan baja es resultado de la misma frecuencia cardiaca (60 a 100 latidos por minuto en reposo), recordando que la frecuencia de detección de complejos corresponde a esta misma.

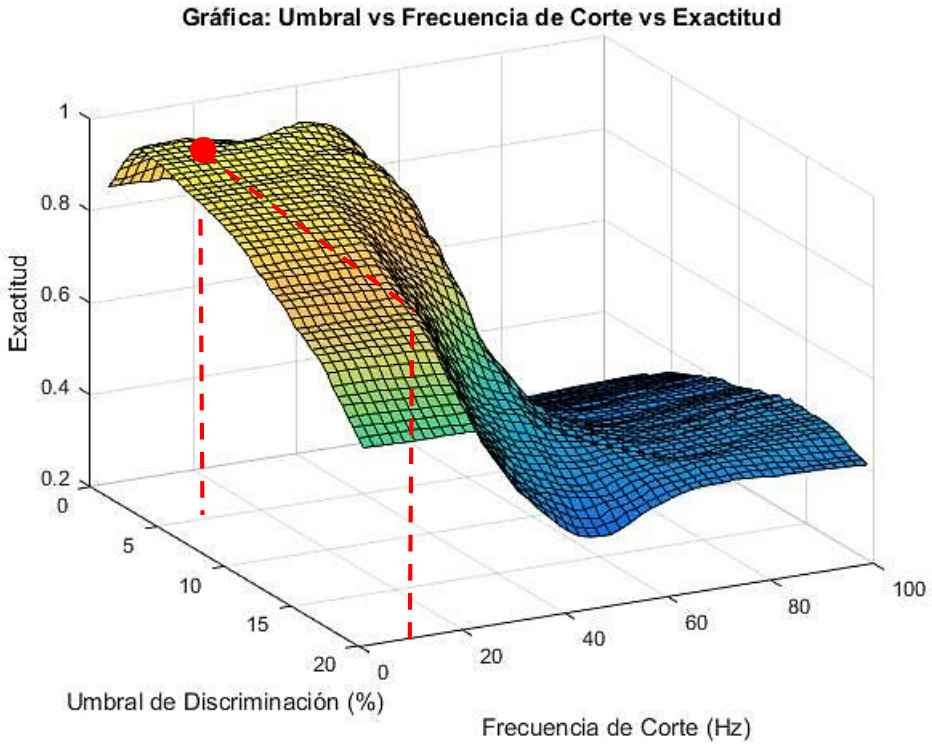


Figura 4. Gráfica: Frecuencia de corte del filtro pasa-bajas vs. Umbral de discriminación vs. Exactitud. En color rojo se señalan los valores seleccionados para maximizar la exactitud.

Ambas pruebas fueron aplicando el umbral y la detección de complejos (búsqueda del máximo por ventaneo) a señal resultante del remuestreo. Para más detalles sobre la realización de estas pruebas, revisar el Apéndice C.

Finalmente, a continuación se exponen los resultados obtenidos en la detección de complejos QRS mediante el algoritmo implementado empleando los parámetros encontrados como óptimos. La figura 5 muestra un segmento de la señal resultante a cada paso del procesamiento aplicado, entregando al final (luego del filtro pasa-bajas) una señal de picos a la que se le aplica el umbral.

La figura 6 muestra propiamente los resultados, mostrando un segmento de la señal original (remuestreada) y la ubicación de los picos R encontrados. Para ilustrar limitaciones del algoritmo y los errores sucedidos en la identificación, la figura 7 (a) deja ver un falso-positivo detectado en una elevación generada por ruido. La figura 7 (b) muestra un falso-negativo al no detectar lo que claramente es un complejo QRS.

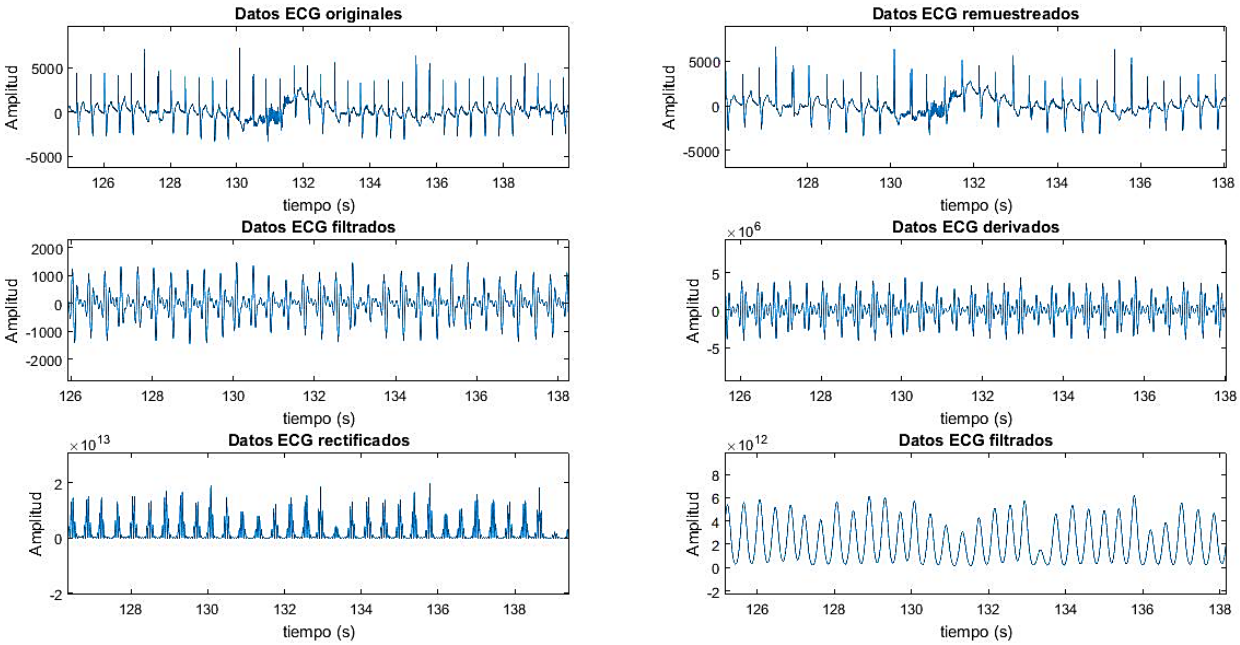


Figura 5. Procesamiento Aplicado a la señal ECG. Se muestra paso a paso, el procesamiento aplicado a la señal ECG, las gráficas corresponden a las señales resultantes de cada paso. (1) Señal original, (2) Señal remuestreada, (3) Señal filtrada con pasa-bandas, (4) Señal derivada, (5) Señal rectificada y (6) Señal filtrada con pasa-bajas.

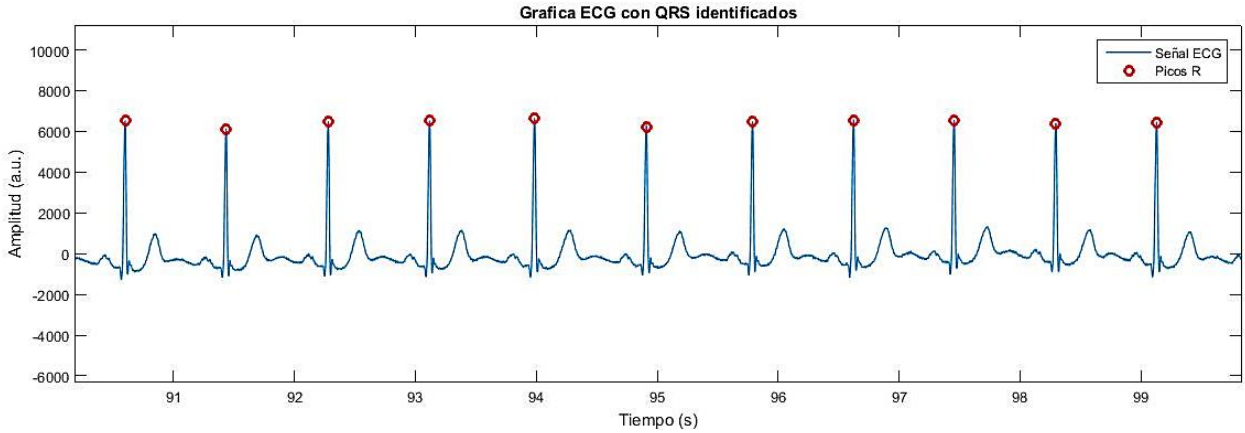
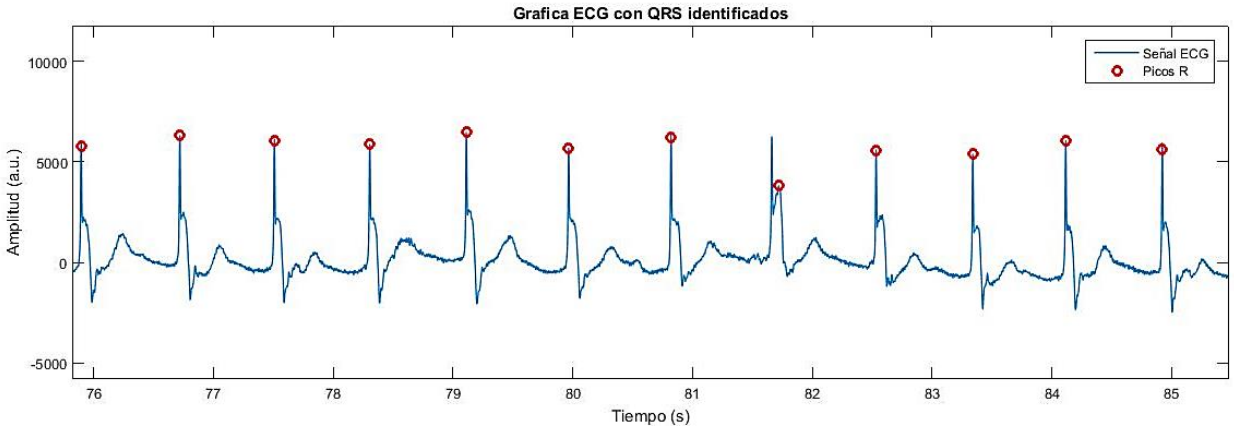
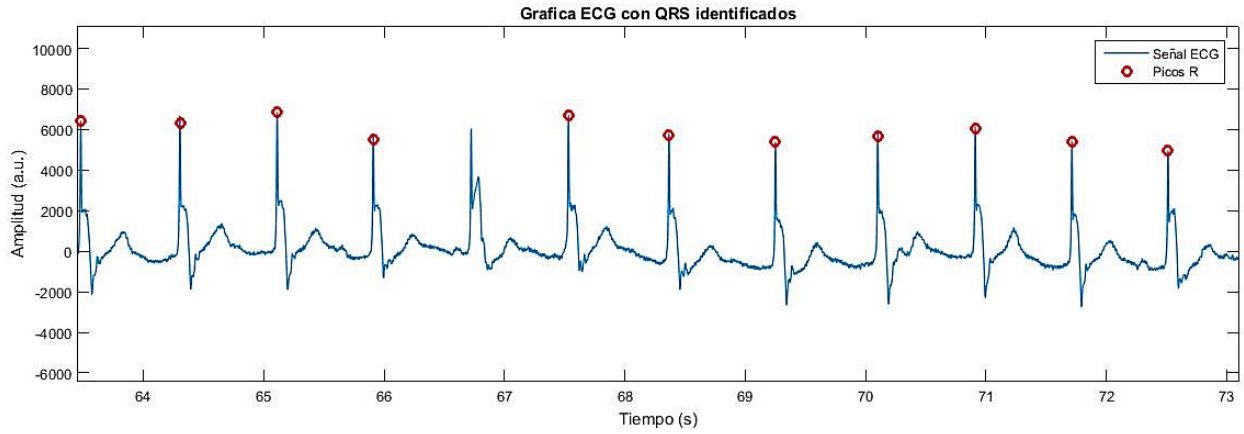


Figura 6. Identificación de Complejos QRS. En azul un segmento de la señal ECG y en rojo la ubicación de los complejos identificados. En este intervalo no existe error.



a)



b)

Figura 7. Limitaciones y errores del algoritmo. Para ambos casos en azul un segmento de la señal ECG y en rojo la ubicación de los complejos QRS identificados. (a) Poco antes del segundo 82 se distingue un falso-positivo en la identificación desfasada del QRS. (b) Aproximadamente en el segundo 67 se distingue un falso-negativo al no identificar el QRS.

Para más detalles sobre la obtención de los resultados obtenidos a partir de los parámetros ya mencionados consultar el apéndice D.

Conclusiones

La exactitud obtenida en la identificación de picos R al alternar los parámetros antes mencionados (umbral, frecuencia de remuestro y frecuencia de corte en pasa-bajas) cambia de manera significativa, por lo que resulta de vital importancia elegir aquellos cuyos valores no comprometan la exactitud y además optimicen el gasto computacional.

Realizar el análisis propuesto para este estudio, permitió identificar esos parámetros obteniendo muy buenos resultados: La señal remuestreada a 350 Hz reduce en gran medida el tamaño de la señal optimizando el procesamiento y la exactitud obtenida fue de 0.98.

Este proyecto solo contempló la implementación de un algoritmo ampliamente utilizado, modificando ciertos parámetros en pro de mejorar su exactitud en la detección. Sin embargo, como trabajo futuro resultaría valioso además realizar un estudio de la variación de las etapas de procesamiento en la búsqueda de desarrollar un propio algoritmo.

Apéndice A. Funciones disponibles

Las siguientes funciones ya existentes fueron empleadas en el algoritmo:

Función ***tread_wfdb.m*:** Hace posible leer el archivo WFDB “mit_sample.dat”, que contiene los valores de la señal empleada.

Función **load_eaf.m:** Hace posible leer el archivo de anotaciones de referencia “mit_sample.eaf generando una estructura. El campo *time* lista los tiempos de ocurrencia de las ondas R.

Función **save_eaf.m:** Permite almacenar las anotaciones generadas por el propio algoritmo de detección.

Función **eaf_compare.m:** Realiza una comparación de dos archivos de anotaciones. Para ello, se emplea la estructura de anotaciones verdaderas y la estructura de anotaciones de prueba.

Apéndice B. Funciones implementadas para el algoritmo

Función *Remuestrear*

```
function r = Remuestrear(vector, fv, fn)
%Función que toma una señal de entrada con una frecuencia fv y la remuestrea
a una nueva frecuencia fn
%Se emplea la función resample para realizar esto

%Argumentos: vector-señal entrada/fv-frecuencia de la señal entrada/fn-
frecuencia con la que se remuestreará.
%            r-senal salida remuestreada
[a,b]=rat(fn/fv); %Fracción simplificada de fn/fv
r=resample(vector, a, b); %Crear en memoria el nuevo vector
end
```

Función *Pasabandas*

```
function Spb = Pasabandas(sig, fs, N, fpa, fpb)
%Función que aplica un filtro pasa-bandas a una señal
%Entradas: sig - señal entrada que se desea filtrar
%          fs - frecuencia de muestreo de la señal entrada
%          N - orden del filtro
%          fpa - frecuencia de corte pasa altas
%          fpb - frecuencia de corte pasa bajas
     fc=[fpa,fpb]; % Vector de frecuencias de corte
    wn = fc*2/fs; % Normalización de las frecuencias de corte
    B=fir1(N, wn, 'bandpass'); % Creación de filtro
    Spb=filtfilt(B,1,sig); % Aplicación del filtro
end
```

Función *Derivar*

```
function der = Derivar(sig, fs)
%Función que aproxima la derivada de una señal
%Entradas: sig- señal de entrada
%           fs- frecuencia de muestreo de la señal entrada
%Salidas: der - derivada de la señal
B=[fs -fs]; %Creación de coeficientes de un filtro derivador
der=filtfilt(B, 1, sig); % Aplicación del filtro
end
```

Función *Rectificación*

```
function rect=Rectificacion(sig)
%Función que rectifica una señal de entrada dada elevando los valores al
cuadrado
%Entradas: sig- Señal de entrada a rectificar
%Salidas: rect - Señal de salida rectificada
     rect=sig.^2;   %Elevación al cuadrado, punto a punto
end
```

Función *Pasabajas*

```
function Spb=Pasabajas(sig, fs, N, fpb)
%Función que aplica un filtro pasabajas a una señal de entrada
%Entradas: sig - señal entrada que se desea filtrar
%          fs - frecuencia de muestreo de la señal entrada
%          N - orden del filtro
%          fpb - frecuencia de corte pasa bajas
%Salidas: Spb - Señal de salida filtrada
    wn = fpb*2/fs; % Normalización de las frecuencias de corte
    B=fir1(N, wn, 'low'); % Creación de filtro (Coeficientes)
    Spb=filtfilt(B, 1, sig); % Aplicación del filtro
end
```

Función *Umbral*

```
function y=Umbral(s, umbral)
%Función que aplica umbralización a una señal, arrojando 1´s en los índices
%que sobrepasen el umbral y 0's aquellos que no lo hagan.
%Entradas: s - Señal de entrada a la que se le aplica umbralización
%   umbral - umbral aplicado dado en un porcentaje del máximo de la señal
entrada.
%Salidas: y - señal salida
     U=(umbral/100)*max(s); % Calculo del umbral en valor de amplitud
    y=zeros(1, length(s));  % Se guarda el memoria el vector de salida con
ceros
    for i=1:length(s)
       if(s(i)>U)
        y(i)=1;     % Guardar 1's en índices de la señal que sobrepasen
umbral
       end
    end
end
```

Función *DetectorPicos*

```
function dpr=DetectorPicos(s, vu)
%Función que identifica máximos a una señal de entrada en intervalos dados
%por una señal de impulsos rectangulares
%Entradas: s - Señal de entrada a la que se identifican los máximos
%          vu - Señal de impulsos que contiene intervalos de búsqueda

derivada=zeros(1,length(vu)); % Vector auxiliar para generar intervalos dada
una señal
% de impulsos, contendrá 1 en índices de inicio de búsqueda y -1 en índices
de fin de búsqueda.
for i=2:length(vu)
derivada(i)=vu(i)-vu(i-1);
end

m1=zeros(1, length(find(derivada))/2); % Vector de índices de inicio de
búsqueda
m2=m1; % Vector de indicen de fin de búsqueda

n1=1; %variable auxiliar que permite guardar índices
n2=1; %variable auxiliar que permite guardar índices    ...
```

```
for i=1:length(derivada)

    if(derivada(i)>0)
        m1(n1) = i;             % Guardar índices de intervalos de búsqueda
        n1=n1+1;              % de acuerdo a la derivada
    end
    if (derivada(i)<0)
        m2(n2) = i;
        n2=n2+1;
    end
end

dpr=zeros(1, length(m1)); %Vector que contiene la ubicación de los máximos

for i=1:length(m1)
    vt=s(m1(i):m2(i));
    dpr(i)=m1(i)+find(max(vt)==vt)-1;    % Se busca el máximo en los
intervalos antes definidos
end
end
```

Función *DeteccionQRS*

```
function [AnnTest,e] = DeteccionQRS (data,AnnTruth,fs,fn,w1,w2,wc,U)
%Función que ubica los complejos QRS, dada una señal de ECG. Aplica
remuestreo, filtro pasa-bandas,
%  derivación, rectificación, filtro pasa-bajas y umbralización. Además
%  genera gráficas a cada procesamiento aplicado.
% Entradas: data - señal entrada de ECG
%         AnnTruth - Estructura con las anotaciones verdades acerca de la
señal
%         fs - Frecuencia a la que esta muestreada la señal entrada
%         fn - frecuencia de remuestreo
%         w1,w2 - frecuencias de corte filtro pasa-bandas
%         wc - frecuencia de corte filtro pasa-bajas
%         U - Umbral que se aplicará a la señal

% Salidas: Anntest - Estructura con anotaciones de detección
%                     time: tiempos de detección / unit: ones
%          e - exactitud en la detección al comparar estructuras

time=(1/fs)*(0:length(data)-1);

%Gráfica de datos originales
figure(1)
plot(time, data)
xlabel('tiempo (s)')
ylabel('Amplitud');
title('Datos ECG originales')

%Remuestreo
dataR=Remuestrear(data, fs, fn);
timen=(1/fn)*(0:length(dataR)-1);
%Grafica de señal remuestreada
figure(2)
```

```
plot(timen, dataR)
xlabel('tiempo (s)')
ylabel('Amplitud');
title('Datos ECG remuestreados')

%Filtrado pasa bandas
N=80;
dataF=Pasabandas(dataR,fn,N,w1,w2);
%Grafica de señal filtrada
figure(3)
plot(timen, dataF)
xlabel('tiempo (s)')
ylabel('Amplitud');
title('Datos ECG filtrados')

%Filtro derivador
dataD=Derivar(dataF, fn);
%Grafica de la señal derivada
figure(4)
plot(timen, dataD)
xlabel('tiempo (s)')
ylabel('Amplitud');
title('Datos ECG derivados')
%Rectificación
dataRect=Rectificacion(dataD);
%Grafica de la señal rectificada
figure(5)
plot(timen, dataRect)
xlabel('tiempo (s)')
ylabel('Amplitud');
title('Datos ECG rectificados')

%Filtro pasa bajas
N=80;
dataF2=Pasabajas(dataRect, fs, N, wc);
%Grafica de la señal filtrada
figure(6)
plot(timen,dataF2)
xlabel('tiempo (s)')
ylabel('Amplitud');
title('Datos ECG filtrados')

%Aplicación del umbral
dataU=Umbral(dataF2,U);

%Detección de pR
picosR=DetectorPicos(dataR, dataU);

%Guardar en estructura, los tiempos obtenidos
tiempos=(1/fn)*(picosR-1);
AnnTest.time=tiempos;
AnnTest.unit=ones(1,length(tiempos));

Sp=eaf_compare(AnnTruth,AnnTest,'Window',0.15,'Print','off');
e =Sp.Confuse(1,1)/sum(Sp.Confuse(:));

end
```

Apéndice C. Códigos con iteración para determinar eficiencia al variar parámetros

```
%Variación de parámetros: Frecuencia de re muestreo y umbral de detección
clear all, close all, clc
U=linspace(1,20,50)';
F=151:6:450;
E=zeros(length(F), length(U));

% Cargar datos
data=tread_wfdb('mit_sample.dat');
fs =1000;
time=(1/fs)*(0:length(data)-1);
AnnTruth=load_eaf('mit_sample.eaf');

%Iteración
for i=1:length(F)
    for j=1:length(U)

        dataR=Remuestrear(data, fs, F(i));
        timen=(1/F(i))*(0:length(dataR)-1);

        w1=4;
        w2=11;
        N=80;
        dataF=Pasabandas(dataR, F(i),N,w1,w2);

        dataD=Derivar(dataF, F(i));

        dataRect=Rectificacion(dataD);

        wc=5;
        N=80;
        dataF2=Pasabajas(dataRect, fs, N, wc);

        dataU=Umbral(dataF2, U(j));

        picosR=DetectorPicos(dataR, dataU);

        tiempos=(1/F(i))*(picosR-1);

        AnnTest.time=tiempos;
        AnnTest.unit=ones(1,length(tiempos));

        Sp=eaf_compare(AnnTruth,AnnTest,'Window',0.15,'Print','off');

        E(i, j)=Sp.Confuse(1,1)/sum(Sp.Confuse(:));

    end
end

surf(U, F, E)
xlabel('Umbral de Discriminación (%)')
ylabel('Frecuencia de Remuestreo (Hz)')
zlabel('Exactitud')
title('Gráfica: Umbral vs Frecuencia de Remuestreo vs Exactitud')
```

```
%Variación de parámetros: Frecuencia de corte (pasabajas) y umbral de
detección
clear all, close all, clc
U=linspace(1,20,50)';
F=linspace(1,99,50)';
E=zeros(length(F),length(U));

% Cargar datos
data=tread_wfdb('mit_sample.dat');
fs =1000;
time=(1/fs)*(0:length(data)-1);
AnnTruth=load_eaf('mit_sample.eaf');

%Iteración
for i=1:length(F)
    for j=1:length(U)

        dataR=Remuestrear(data,fs,350);
        timen=(1/350)*(0:length(dataR)-1);

        w1=4;
        w2=11;
        N=80;
        dataF=Pasabandas(dataR,350,N,w1,w2);

        dataD=Derivar(dataF,350);

        dataRect=Rectificacion(dataD);

        N=80;
        dataF2=Pasabajas(dataRect, fs, N, F(i));

        dataU=Umbral(dataF2,U(j));

        picosR=DetectorPicos(dataF2,dataU);

        tiempos=(1/350)*(picosR-1);

        AnnTest.time=tiempos;
        AnnTest.unit=ones(1,length(tiempos));

        Sp=eaf_compare(AnnTruth,AnnTest,'Window',0.15,'Print','off');

        E(i, j)=Sp.Confuse(1,1)/sum(Sp.Confuse(:));

    end

end

surf(U, F, E)
xlabel('Umbral de Discriminación (%)')
ylabel('Frecuencia de Corte (Hz)')
zlabel('Exactitud')
title('Gráfica: Umbral vs Frecuencia de Corte vs Exactitud')
```

Apéndice D. Código de detección de picos R utilizando los parámetros más eficientes

```
clear all, close all, clc

% Cargar datos
data=tread_wfdb('mit_sample.dat');
fs =1000;
time=(1/fs)*(0:length(data)-1);
AnnTruth=load_eaf('mit_sample.eaf');
p=zeros(1,length(AnnTruth.time));

%Gráfica de datos originales
figure(1)
plot(time,data)
xlabel('tiempo (s)')
ylabel('Amplitud');
title('Datos ECG originales')

%Detección de complejos QRS

[AnnTest,E]=DeteccionQRS(data,AnnTruth,fs,350,5,12,10,4)

%Grafica de pR encontrados
p=zeros(1,length(AnnTest.time));

for n=1:length(p)

    for m=1:length(data)

       if(time(m)>=AnnTest.time(n))
         p(n)=data(m);
         break
       end

    end
end

figure
plot(time,data)
hold on
plot(AnnTest.time,p,'x')
xlabel('Tiempo (s)'); ylabel('Amplitud (a.u.)')
title('Grafica ECG con QRS identificados')
legend('Señal ECG', 'Picos R')
```

Referencias

[1] Federación Argentina de Cardiología (s.f). Electrocardiografía. [online] Recuperado de: http://fac.org.ar/faces/comites/tecnenf/ecg01.PDF [9 Feb. 2018].

[2] Pan and W. Tompkins, "A Real-Time QRS Detection Algorithm", IEEE Transactions on Biomedical Engineering, vol. -32, no. 3, pp. 230-236, 1985.