Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado

ECGMiner: un software para digitalizar electrocardiogramas

GRADO EN INGENIERÍA INFORMÁTICA

MENCIÓN EN COMPUTACIÓN

Autor:
Adolfo Fernández Santamónica

Tutoras:
Rocío Carratalá Sáez
Cristina Rueda Sabater

Agradecimientos

A todas aquellas personas que habéis estado a mi lado: no puedo expresar con palabras lo agradecido que me siento por todo vuestro apoyo a lo largo de este proyecto, ni tampoco podía entregar este trabajo sin dejar por escrito mi gratitud hacia vosotros.

A mis padres José Luis y María José, por vuestro apoyo y cariño incondicional y por celebrar con orgullo todos mis logros.

A mis hermanos David y Fabio, por vuestro constante apoyo y admiración por mi trabajo, que aún no entendiéndolo en profundidad, siempre me habéis preguntado por él con ilusión. Ojalá os sirva de referencia para vuestro futuro académico.

A mi tutora Rocío, por haber valorado mi trabajo y haber creído en mí desde el primer día.

A mi tutora Cristina, por haberme ofrecido esta oportunidad de trabajo única y por haber sido una inagotable fuente de energía y motivación en todo momento.

Al cardiólogo Alberto Pérez Castellanos, por su colaboración en el proceso de validación de la herramienta desarrollada.

Resumen

El electrocardiograma (ECG) es el método no invasivo más importante para dilucidar información sobre el corazón y para el diagnóstico de enfermedades cardiovasculares. Normalmente, las empresas fabricantes de sistemas de ECG proporcionan imágenes de ECG, pero almacenan los datos numéricos en un formato propietario y cifrado, de modo que no es interpretable y, por tanto, no se puede utilizar para el diagnóstico automático. En la literatura existen trabajos previos centrados en digitalizar imágenes de ECG para obtener los valores numéricos correspondientes. Las principales limitaciones de dichos trabajos son que requieren la selección manual de las regiones de interés, que sólo proporcionan información parcial de las señales y que ofrecen una precisión limitada.

Este trabajo presenta ECGMiner, un software para digitalizar imágenes de ECG. Este software realiza la digitalización en tres pasos: 1) preprocesado, que incluye un reconocimiento de la imagen y una posterior eliminación de la cuadrícula; 2) extracción de las señales, en el que se detectan las regiones de interés y las señales a extraer; 3) posprocesado, en el que se detectan los pulsos de referencia y se normalizan las señales. Adicionalmente, se ofrece la posibilidad de extraer los metadatos con la información clínica del paciente anotada en el ECG.

Las capacidades de digitalización de ECGMiner han sido evaluadas utilizando el coeficiente de correlación de Pearson (PCC) y la raíz del error cuadrático medio (RMSE) sobre los ECG de dos grandes bases de datos públicas y ampliamente utilizadas en el campo: LUDB y PTB-XL. Se han comparado los valores digitalizados por ECGMiner para las señales de ambas bases de datos con los valores originales. También se ha validado la capacidad del software para digitalizar correctamente las ondas P, R y T, típicamente usadas en el análisis de ECG. En concreto, los valores de PCC se sitúan entre 0,971 y 0,995, y los de RMSE entre 11,4 y 30,9.

El software ECGMiner es de acceso abierto, fácil de instalar, fácil de usar y capaz de digitalizar con precisión los datos de la señal ECG. ECGMiner supera a los algoritmos de digitalización existentes en términos de PCC y RMSE.

Palabras clave: ECG, imágenes, digitalización, software

Abstract

The electrocardiogram (ECG) is the most important noninvasive method for elucidating information about the heart and for the diagnosis of cardiovascular diseases. Typically, ECG system manufacturing companies provide ECG images, but store the numerical data in a proprietary and encrypted format, so that it is not interpretable and therefore cannot be used for automatic diagnosis. Previous work in the literature has focused on digitizing ECG images to obtain the corresponding numerical values. The main limitations of such work are that it requires manual selection of the regions of interest, provides only partial signal information, and offers limited accuracy.

This work presents ECGMiner, a software for digitizing ECG images. This software performs digitization in three steps: 1) preprocessing, which includes image recognition and subsequent grid removal; 2) signal extraction, in which regions of interest and signals to be extracted are detected; 3) post-processing, in which reference pulses are detected and signals are normalized. Additionally, it offers the possibility of extracting metadata with the patient's clinical information annotated on the ECG.

The digitization capabilities of ECGMiner have been evaluated using Pearson's correlation coefficient (PCC) and root mean square error (RMSE) on ECGs from two large public databases widely used in the field: LUDB and PTB-XL. The values digitized by ECGMiner for signals from both databases have been compared with the original values. The ability of the software to correctly digitize P, R and T waves, typically used in ECG analysis, has also been validated. Specifically, the PCC values are between 0.971 and 0.995, and the RMSE values are between 11.4 and 30.9.

ECGMiner software is open access, easy to install, easy to use and capable of accurately digitizing ECG signal data. ECGMiner outperforms existing digitization algorithms in terms of PCC and RMSE.

Keywords: ECG, images, digitization, software

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se describe el contexto del proyecto. A continuación, se motiva el trabajo haciendo referencia al grupo de investigación que lo ha planteado y al modelo matemático con el que dicho grupo aspira a continuar su investigación gracias a la herramienta desarrollada en este proyecto. Tras esto, se explicita el objetivo del proyecto y, finalmente, se detalla la estructura del presente documento.

1.1. Contexto

En los últimos años, la tecnología ha llevado al poder computacional a nuevos horizontes, impulsando una revolución sin precedentes en el procesamiento de información. Con cada nueva generación, los límites se empujan aún más lejos, permitiendo a las máquinas realizar tareas más complejas y ayudar significativamente en la toma de decisiones de problemas críticos, tales como el diagnóstico médico. Todo ello, sumado a la popularización del estudio e investigación de algoritmos para analizar señales bioeléctricas, ha alentado a la comunidad científica a desarrollar modelos matemáticos y software que sirvan como soporte a los profesionales sanitarios. La señales bioeléctricas son señales de baja amplitud y frecuencia provenientes del cuerpo humano, generadas por reacciones electroquímicas de cierto tipo de células al ser excitadas. Entre algunas de estas señales se encuentran el electrocardiograma (ECG) o el electroencefalograma (EEG). Concretamente, el ECG juega un papel fundamental en el caso del estudio del sistema cardíaco, ya que es probablemente el método no invasivo más importante a la hora de detectar y diagnosticar enfermedades del corazón.

1.2. Motivación

Recientemente, el Grupo de Inferencia con Restricciones (GIR) [1, 2] de la Universidad de Valladolid ha desarrollado un procedimiento para el análisis de señales enormemente prometedor, el modelo 3DFMM$_{ecg}$ [3] (*3D Frequency Modulated Möbius*). El enfoque trata de ser universal, de forma que compite con la clásica descomposición de Fourier o la descomposición en ondículas, combinando una formulación con muy buenas propiedades computacionales y estadísticas, junto con la enorme ventaja que posee gracias a la interpretabilidad de sus parámetros y su robustez frente al ruido. Pese a que ya se ha probado su potencial aplicación en múltiples campos de estudio, se quiere dirigir su desarrollo hacia un diagnóstico clínico en tiempo real. Sin embargo, la mayoría de hospitales solo disponen de información gráfica de las señales generadas por el electrocardiógrafo, ya sea en papel o imágenes escaneadas. Normalmente, las empresas fabricantes

no proveen estas señales numéricamente, sino que las almacenan en un formato propietario, cifrado de tal forma que no pueda ser interpretado ni, por tanto, útil.

1.3. Objetivo

En este proyecto se persigue desarrollar una herramienta que digitalice imágenes de ECG, de forma que consiga obtener las señales digitalizadas numéricamente a partir de las imágenes generadas por un electrocardiógrafo. Este software será de código abierto, de manera que no solo sea de utilidad para el grupo de investigación, sino también para futuros usuarios que necesiten obtener datos de ECG.

Este proyecto está estrechamente relacionado con el Trabajo de Fin de Grado en Estadística «Clasificación de enfermedades cardíacas a partir de los parámetros del modelo 3DFMMecg» [4], en el cual se ha llevado a cabo un análisis para la clasificación de enfermedades cardíacas a partir de datos de ECG. Aunque los dos trabajos tienen objetivos claramente diferenciados, ambos tienen como denominador común el ECG y el modelo 3DFMM$_{ecg}$, por lo que su correspondiente explicación teórica será común en ambas memorias.

1.4. Estructura del documento

En cuanto a la estructura de este documento, se organiza de la siguiente manera:

Capítulo 1. Introducción. En este capítulo se describe el contexto del proyecto, así como la motivación y el objetivo del mismo.

Capítulo 2. Marco teórico. En este capítulo se detallan todos los conceptos teóricos necesarios para comprender el electrocardiograma y el modelo 3DFMM$_{ecg}$. Además, se presenta una revisión del estado del arte en lo que a la digitalización de ECG se refiere.

Capítulo 3. Planificación. En este capítulo se describe el modelo de desarrollo del proyecto, el cual incluye su planificación inicial, el camino crítico y el seguimiento de la misma. Asimismo, se presenta un análisis de riesgos y un presupuesto detallando el coste estimado que supone la realización del proyecto.

Capítulo 4. Análisis y diseño. En este capítulo se describe la elicitación de requisitos, las tecnologías empleadas, la arquitectura y el diseño detallado del sistema, el diseño del algoritmo de digitalización y el diseño de la interfaz de usuario.

Capítulo 5. Implementación. En este capítulo se detallan las desviaciones respecto al diseño inicial, la organización del código y la disponibilidad y licencia de la herramienta.

Capítulo 6. Pruebas. En este capítulo se presentan los distintos casos de prueba planteados para comprobar la corrección de la aplicación, así como los resultados de las mismas. Además, se explica cómo se ha evaluado la precisión de la herramienta. Finalmente, se ilustra el potencial de la herramienta al ser utilizada junto con el modelo 3DFMM$_{ecg}$.

Capítulo 7. Conclusiones. En este capítulo se exponen las conclusiones obtenidas tras la finalización del proyecto, así como las propuestas de trabajo futuro y la valoración personal.

Apéndice A. Dependencias software. En este apéndice se incluyen las dependencias software de la herramienta.

Apéndice B. Manual de instalación. En este apéndice se incluye el manual de instalación de la herramienta.

Apéndice C. Manual de usuario. En este apéndice se incluye el manual de usuario de la herramienta.

Capítulo 2

Marco teórico

En este capítulo se detallan todos los conceptos teóricos necesarios para comprender el ECG y el modelo matemático 3DFMMecg. Además, se presenta una revisión del estado del arte en lo que a la digitalización de ECG se refiere.

2.1. El electrocardiograma

La señal del ECG [5] es un registro de la actividad eléctrica del corazón. Es probablemente el método de diagnóstico no invasivo más importante de la medicina y se utiliza desde hace décadas para la detección de alteraciones y patologías cardíacas con un riesgo mínimo para el paciente. Para hacer una interpretación del ECG, conviene familiarizarse con la fisiología del corazón. El corazón es un órgano muscular que se divide en cuatro cámaras: dos aurículas (derecha e izquierda) y dos ventrículos (derecho e izquierdo). Las aurículas están situadas en la parte superior y los ventrículos en la parte inferior. Durante el ciclo cardíaco, la sangre fluye de las aurículas a los ventrículos y, luego, es bombeada hacia las arterias durante la contracción ventricular. El corazón genera una señal eléctrica que es producida por la despolarización y repolarización de las células del músculo cardíaco. La despolarización ocurre cuando los iones de sodio (Na+) ingresan en las células musculares cardíacas, lo que provoca una carga eléctrica positiva en las células. Esto desencadena la contracción del músculo cardíaco y produce el primer sonido del corazón. La repolarización ocurre cuando los iones de potasio (K+) salen de las células musculares cardíacas, lo que restaura la carga eléctrica negativa en las células y permite que el corazón se relaje y se prepare para el siguiente latido. Esto produce el segundo sonido del corazón.

Para registrar el comportamiento del campo eléctrico producido por las células cardíacas se colocan una serie de electrodos sobre la piel del paciente, tanto en las extremidades como en localizaciones específicas del tórax. Al igual que para realizar una fotografía no se suele tomar una única instantánea, ya que se quiere ver el resultado desde distintos ángulos, con el ECG sucede lo mismo. Las 12 derivaciones del ECG son las direcciones en las que se registra la actividad eléctrica del corazón, y dan lugar a distintas series de tiempo con las diferencias de potenciales (voltajes) medidas entre dos puntos. Estas series temporales se etiquetan con el nombre de su respectiva derivación y, según el plano eléctrico en el que se registren, se pueden clasificar en dos grupos: las derivaciones de las extremidades del plano frontal y las derivaciones precordiales del plano horizontal (ver Figura 2.1).

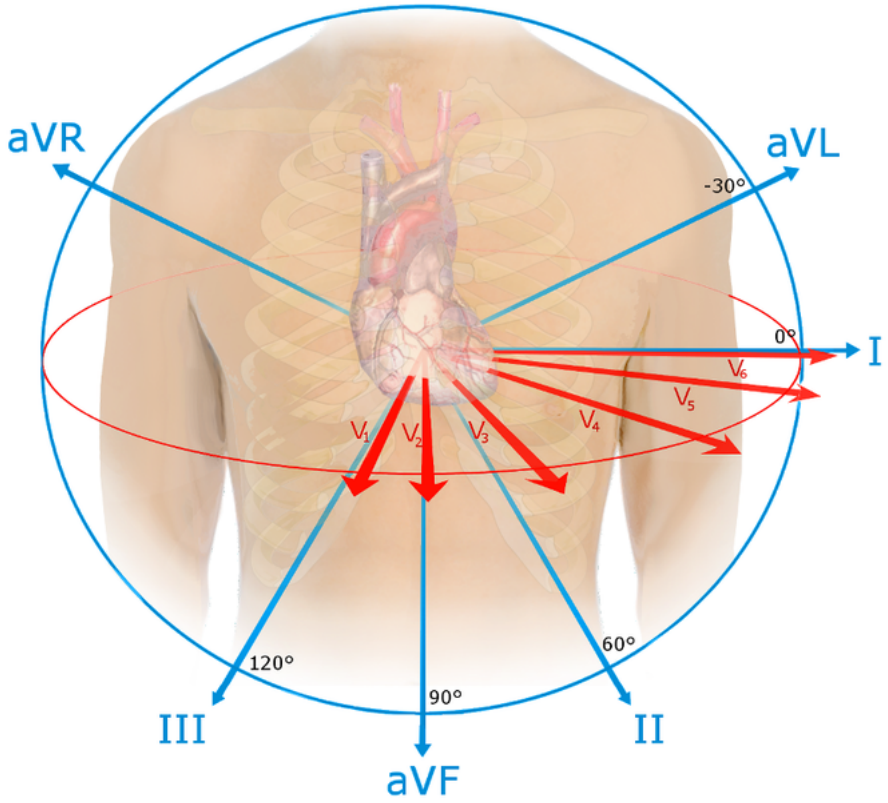


Figura 2.1: Orientación espacial de las derivaciones del ECG [6].

Las derivaciones del plano frontal registran las diferencias de potencial a través de los electrodos situados en las extremidades. Estas se pueden clasificar a su vez en otros dos grupos: derivaciones bipolares y derivaciones monopolares aumentadas. Las primeras son tres (I, II y III). Estas derivaciones forman lo que se conoce como el triángulo de Einthoven, ya que verifican una ecuación matemática conocida como la ley de Einthoven:

$$II = I + III \tag{2.1}$$

Por otro lado, las derivaciones monopolares aumentadas son también tres (aVR, aVL y AVF), y, matemáticamente, se pueden obtener mediante combinaciones lineales de las anteriores:

$$aVL = \frac{I - II}{2} \tag{2.2}$$

$$aVR = -\frac{I + II}{2} \tag{2.3}$$

$$aVF = \frac{II + III}{2} \tag{2.4}$$

Por último, las derivaciones del plano horizontal son monopolares, se nombran con una V mayúscula y el correspondiente número del 1 al 6 (V1, V2, V3, V4, V5 y V6).

El ECG posee información de todas estas derivaciones en latidos consecutivos a lo largo del tiempo. A la hora de representar el ECG, el formato más común es el que se describe en la Figura 2.2, el cual recoge información de las 12 derivaciones a lo largo de 10s (segundos). Está formado por una rejilla de colores rojizos o anaranjados en la que se colocan las derivaciones en color negro

o azul para que destaquen. En el eje de abscisas se mide el tiempo y en el eje de ordenadas se mide el voltaje. Por norma general, cada cuadrado pequeño de la rejilla equivale 0,04s y 0,1mV (milivoltios).

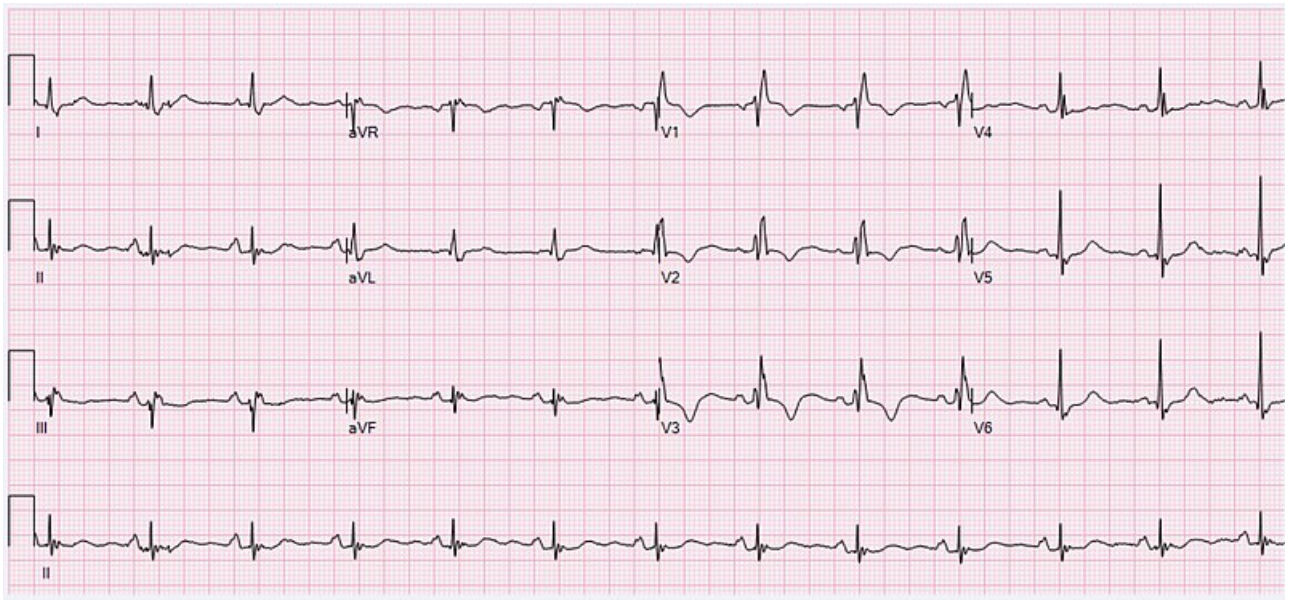


Figura 2.2: ECG de una mujer de 78 años con ritmo sinusal normal y bloqueo de rama derecha [7].

Por las limitaciones de espacio que había antiguamente cuando solo se trabajaba en papel, no resultaba útil medir los 10s para cada una de las 12 derivaciones, ya que se podían cruzar las señales y apenas distinguirse correctamente los trazos. Es por ello que, de forma habitual, se trabaja con la disposición 3x4 (3 filas y 4 columnas). Esto quiere decir que, para los 10 segundos totales de duración que tiene típicamente un ECG, se imprimen los 2,5 primeros segundos correspondientes a las derivaciones bipolares I, II y III. A continuación, desde el segundo 2,5 al segundo 5, las señales de las derivaciones aVR, aVL, y aVF. Después, del segundo 5 hasta el 7,5, las derivaciones precordiales V1, V2 y V3. Finalmente, durante los últimos 2,5 segundos se colocan las derivaciones V4, V5 y V6. Una variable importante para el diagnóstico del ECG es la frecuencia cardíaca; para medirla correctamente sobre única derivación, se suele añadir en la parte inferior una tira de ritmo, la cual es una derivación extendida durante la duración total del ECG. Para ello, se suele emplear la derivación II por ser considerada como la más informativa.

Otro detalle a tener en cuenta es la presencia de pequeños pulsos rectangulares situados al principio o al final de cada señal: los pulsos de referencia. Al encontrarse apiladas las señales unas encima de otras, estos pulsos son de utilidad para marcar los valores de 0mV y 1mV para cada una de las filas de señales. Cabe añadir que la rejilla con las señales suele estar rodeada de un marco blanco que contiene algunas variables clínicas extraídas del ECG o incluso datos relativos al paciente (nombre, edad, etc).

Existen variaciones de la estructura citada anteriormente. Por ejemplo, se pueden emplear hasta tres tiras de ritmo en vez de una, disposiciones 6x2 y 12x1, o incluso otro formato que reordena las derivaciones de una manera considerada como más "anatómica", denominado formato Cabrera (ver Figura 2.3).

$$\begin{pmatrix} I & aVR & V1 & V4 \\ II & aVL & V2 & V5 \\ III & aVF & V3 & V6 \end{pmatrix} \qquad \begin{pmatrix} aVL & II & V1 & V4 \\ I & aVF & V2 & V5 \\ -aVR & III & V3 & V6 \end{pmatrix}$$

(a) Formato estándar. (b) Formato Cabrera.

Figura 2.3: Formatos más habituales de un ECG de 12 derivaciones.

Si nos acercamos a visualizar un segmento en el que se recoja un único latido, por ejemplo en la derivación II, se puede observar que el latido del corazón está compuesto por diferentes ondas, las cuales representan los distintos eventos que se producen durante el ciclo cardíaco. En la Figura 2.4 se puede observar la morfología de un latido de un ECG normal (paciente sano), aunque las individualidades y patologías de cada paciente pueden provocar modificaciones sustanciales. La onda P es la primera del ciclo cardíaco y representa la despolarización auricular. A continuación, aparece el complejo QRS conformado por las ondas Q, R y S generalmente. Este complejo indica la despolarización de los ventrículos y la repolarización de las aurículas, que ocurren simultáneamente, aunque muchas veces la repolarización auricular queda oculta en el QRS y no llega a apreciarse en el ECG. Finalmente, la onda T cierra el ciclo a través de la repolarización ventricular.

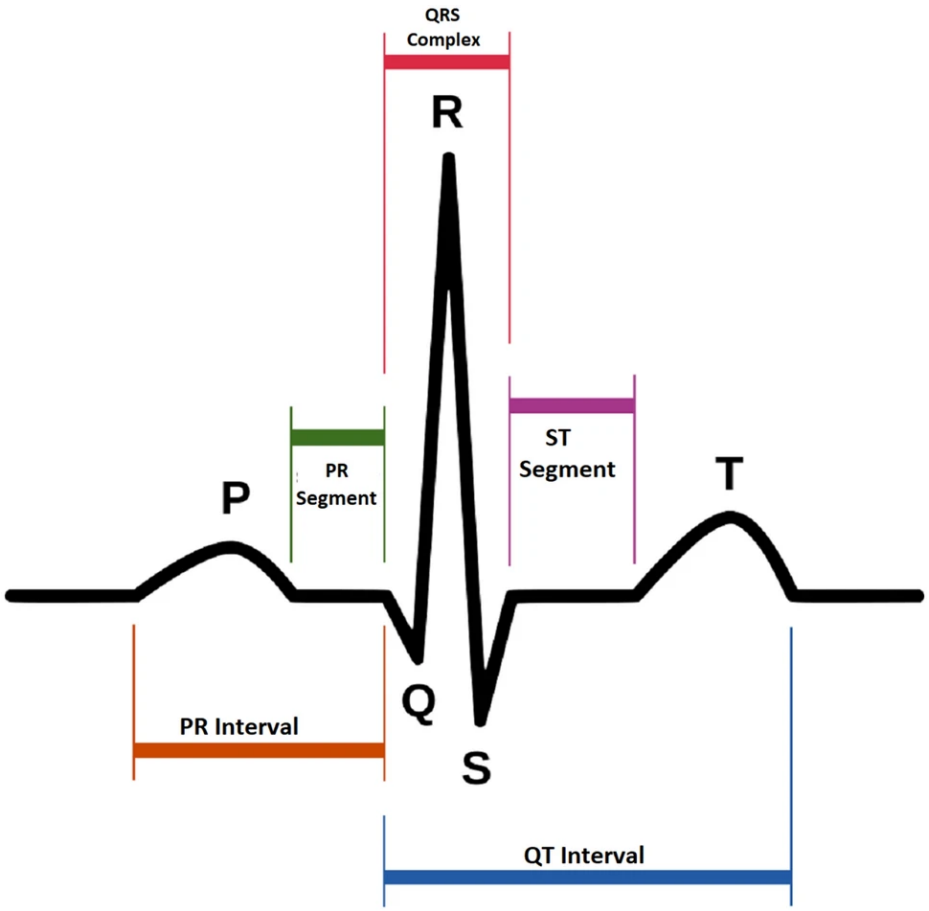


Figura 2.4: Morfología de un ECG normal (paciente sano) típico [8].

También existen otra serie de elementos de interés clínico. Los segmentos PR y ST, miden la distancia entre los comienzos o finales de dos ondas. Los intervalos PR y QT, miden el tiempo entre dos eventos de actividad eléctrica. El intervalo RR mide el tiempo entre las ondas R de dos latidos distintos.

2.2. El enfoque FMM

El enfoque FMM (*Frequency Modulated Möbius*) es un procedimiento para el análisis de señales que compite con la clásica descomposición de Fourier o la descomposición en ondículas, combinando una formulación con muy buenas propiedades computacionales y estadísticas junto con la enorme ventaja que posee gracias a la interpretabilidad de sus parámetros y su robustez frente al ruido. En esta sección se presentará el modelo 3DFMM$_{ecg}$ y se motivará con distintos argumentos matemáticos y biológicos.

2.2.1. Señales oscilatorias

Una señal oscilatoria [9] es una señal que se repite cíclicamente a intervalos regulares a lo largo del tiempo. Durante este periodo, la señal puede ser más simple y llegar a oscilar una única vez (ver Figura 2.5), o ser más complicada y tener más de una oscilación, como es el caso del ECG.

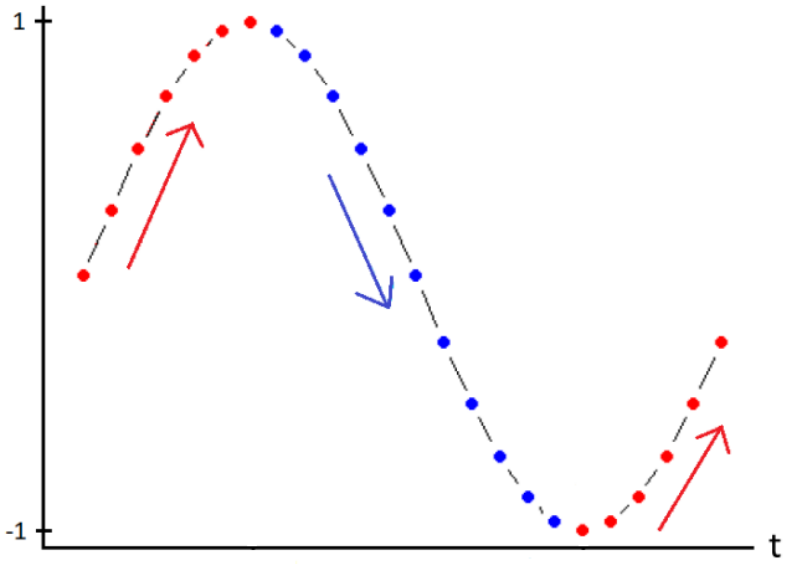


Figura 2.5: Esquema de una señal circular con una única oscilación. Adaptado de [9].

Las señales se pueden registrar en cualquier intervalo de tiempo, pero es habitual realizar una transformación del espacio euclídeo al círculo unidad, definiéndose así la señal como:

$$\mu(t) = \cos\left(\phi(t)\right), \quad 0 < \phi(t) \leq 2\pi, \quad 0 < t \leq 2\pi \tag{2.5}$$

Un ejemplo de señal oscilatoria simple con una única oscilación es la señal:

$$\phi(t) = \beta + 2\arctan\left(\omega \tan\left(\frac{t-\alpha}{2}\right)\right); \quad t \in (0, 2\pi] \tag{2.6}$$

A partir de ella, se define la señal compleja:

$$S(t) = \tau(c) = e^{i\phi(t)} \tag{2.7}$$

donde $\tau(c)$ se conoce como la transformación de Möbius, definida en el círculo unidad como:

$$\tau(c) = b\frac{c-a}{1-\bar{a}c}\ ; \quad a,b,c \in \mathbb{C}\ ; \quad |a|<1\ ; \quad |b|=1\ ; \quad |c|=1 \tag{2.8}$$

De esta manera, una onda de Möbius es la parte real de una señal compleja definida por la transformación de Möbius:

$$W(t,\alpha,\beta,\omega) = \cos\left(\phi(t)\right) \tag{2.9}$$

Los parámetros que caracterizan la onda son el parámetro de localización α, el parámetro de anchura ω y el parámetro de dirección de onda β. A partir de este objeto matemático surge la onda FMM, la cual añade un cuarto parámetro de escala A: $AW(t,\alpha,\beta,\omega)$. En la Figura 2.6 se pueden observar distintos ejemplos ilustrativos de como cambian las ondas a medida que se varían los distintos parámetros de las mismas.

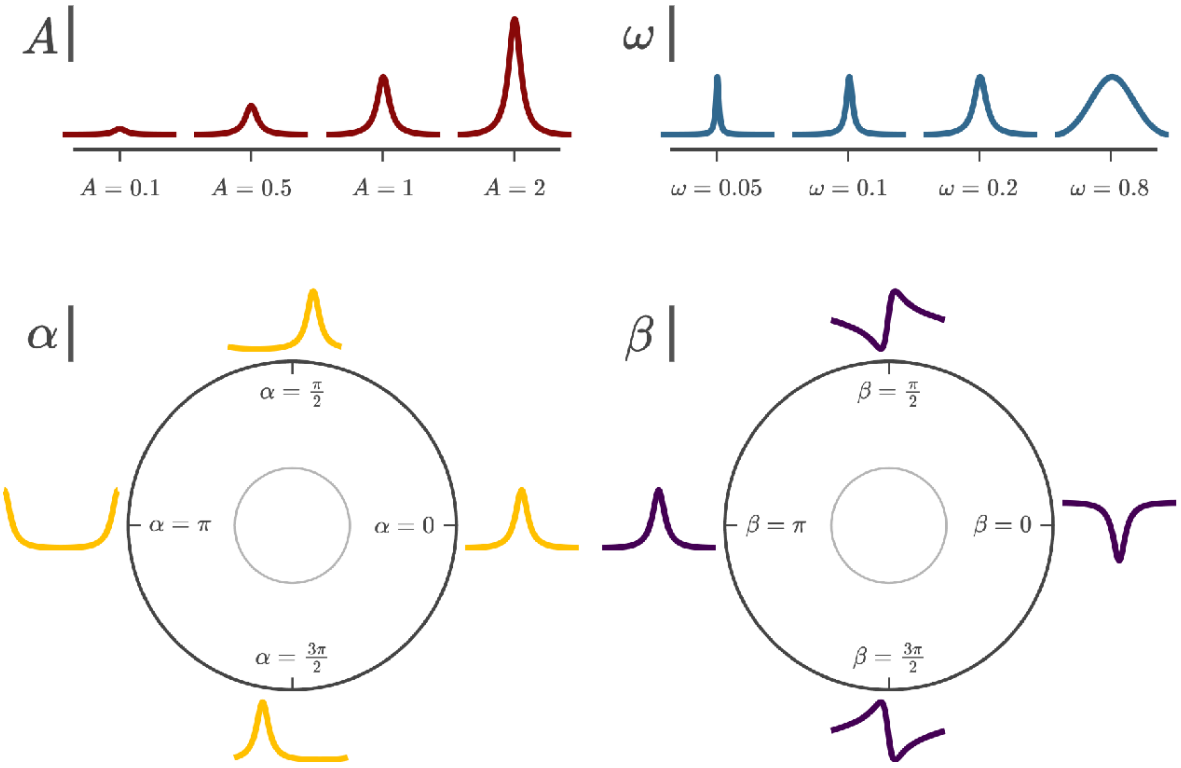


Figura 2.6: Ondas FMM de la forma $AW(t,\alpha,\beta,\omega)$ para varios valores de los parámetros (A,α,β,ω). Salvo que se indique lo contrario $A=1$, $\alpha=0$, $\beta=\pi$ y $\omega=0{,}2$ [10].

2.2.2. El modelo FMM$_{ecg}$

En la práctica, muchas de las señales oscilatorias asociadas a procesos biológicos, como son las señales del ECG, cuentan con más de una oscilación. De ello, surge la necesidad de formular modelos con varias componentes que tengan en cuenta la presencia de múltiples oscilaciones.

El modelo FMM$_{ecg}$ [11] es un modelo paramétrico multicomponente, formulado como la suma del conjunto de ondas de una derivación del ECG, a lo que se le añade además un término de ruido. Sea $X(t_i)$ la observación i-ésima del voltaje medido sobre una derivación $(t_1 < ... < t_n)$, se define el modelo:

$$X(t_i) = M + \sum_{J\in\{P,Q,R,S,T\}} A_J W(t_i,\alpha_J,\beta_J,\omega_J) + \epsilon(t_i) \tag{2.10}$$

donde

1. $M \in \mathbb{R},\ \beta_J \in (0, 2\pi],\ \omega_J \in [0, 1],\ A_J \in \mathbb{R}^+$
2. $\alpha_P \leq \alpha_Q \leq \alpha_R \leq \alpha_S \leq \alpha_T$
3. $(\epsilon(t_1), ..., \epsilon(t_n))' \sim N_n(0, \sigma I)$

El parámetro M es el *intercept* o término independiente del modelo, el cual describe el nivel de voltaje base de la señal. La restricción entre los parámetros α indica que las ondas se encuentran localizadas de acuerdo al orden biológico ($P \to Q \to R \to S \to T$). Se asume que los errores $\epsilon(t_i)$ son independientes y están normalmente distribuidos con media 0 y desviación típica σ.

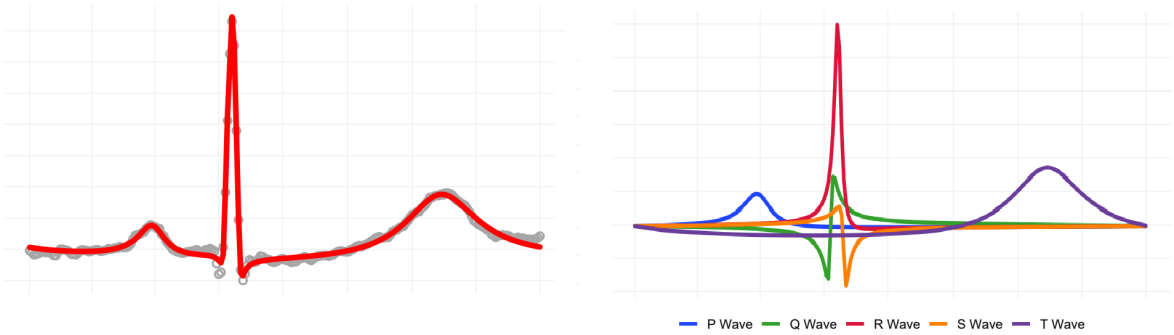


Figura 2.7: Ejemplo ilustrativo del ajuste del modelo FMM$_{ecg}$ sobre un latido típico de un ECG. Adaptado de [12].

2.2.3. El modelo 3DFMM$_{ecg}$

Los modelos cardíacos parten de la premisa general de que el campo eléctrico en el corazón es un proceso tridimensional representado por un vector $\vec{D}(t)$ que varía su magnitud y su dirección con el tiempo [13, 14]. Habitualmente $\vec{D}(t)$ describe tres bucles. Los bucles P y T son elípticos, mientras que el bucle QRS tiene una forma más irregular (ver Figura 2.8). Por otro lado, las señales del ECG son proyecciones de $\vec{D}(t)$ en 12 direcciones de dicho vector, las conocidas como derivaciones ($Lset \in \{I, II, III, aVR, aVL, aVF, V1, V2, V3, V4, V5, V6\}$). Adicionalmente, dichas señales son observadas con un cierto error.

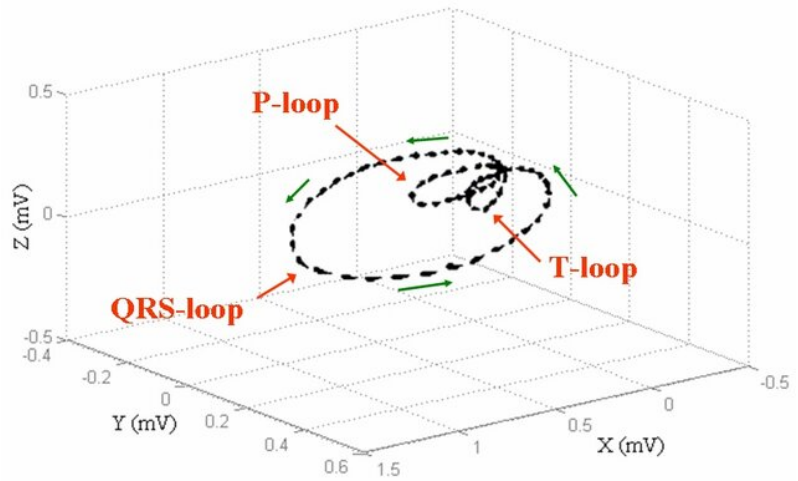


Figura 2.8: Trayectoria tridimensional que describe el vector $\vec{D}(t)$ para un ciclo cardíaco normal. Adaptado de [14].

La novedad del modelo 3DFMM$_{ecg}$ [3] parte de la asunción de que $\vec{D}(t)$ combina las señales eléctricas de exactamente cinco fuentes diferenciadas, correspondiéndose con las cinco señales primordiales del ECG:

$$\vec{D}(t) = \vec{d}_P(t) + \vec{d}_Q(t) + \vec{d}_R(t) + \vec{d}_S(t) + \vec{d}_T(t) \tag{2.11}$$

Por tanto, el vector $\vec{D}(t)$ se puede expresar como una suma de ondas, cada una de las cuales se corresponde con una onda FMM. Estas ondas se encuentran sincronizadas, lo que quiere decir que los parámetros α y ω son compartidos por todas las derivaciones. Sea $X^L(t_i)$ la observación i-ésima del voltaje medido sobre la derivación L, $(t_1 < ... < t_n)$, $(t_i \in (0, 2\pi])$, se define el modelo 3DFMM$_{ecg}$ como:

$$X^L(t_i) = M^L + \sum_{J \in \{P,Q,R,S,T\}} A_J^L W(t_i, \alpha_J, \beta_J^L, \omega_J) + \epsilon^L(t_i) \tag{2.12}$$

donde, para $L \in Lset$ y $J \in \{P, Q, R, S, T\}$:

1. $M^L \in \mathbb{R},\ \beta_J^L \in (0, 2\pi],\ \omega_J \in [0, 1],\ A_J^L \in \mathbb{R}^+$
2. $\alpha_P \leq \alpha_Q \leq \alpha_R \leq \alpha_S \leq \alpha_T$
3. $(\epsilon^L(t_1), ..., \epsilon^L(t_n))' \sim N_n(0, \sigma^L I)$

Las restricciones descritas anteriormente garantizan la identificabilidad de los parámetros. Para ello, se realiza una primera etapa de preprocesado [3], seguida del propio algoritmo de identificación con el objetivo de obtener el estimador máximo verosímil, los cuales se describirán a continuación.

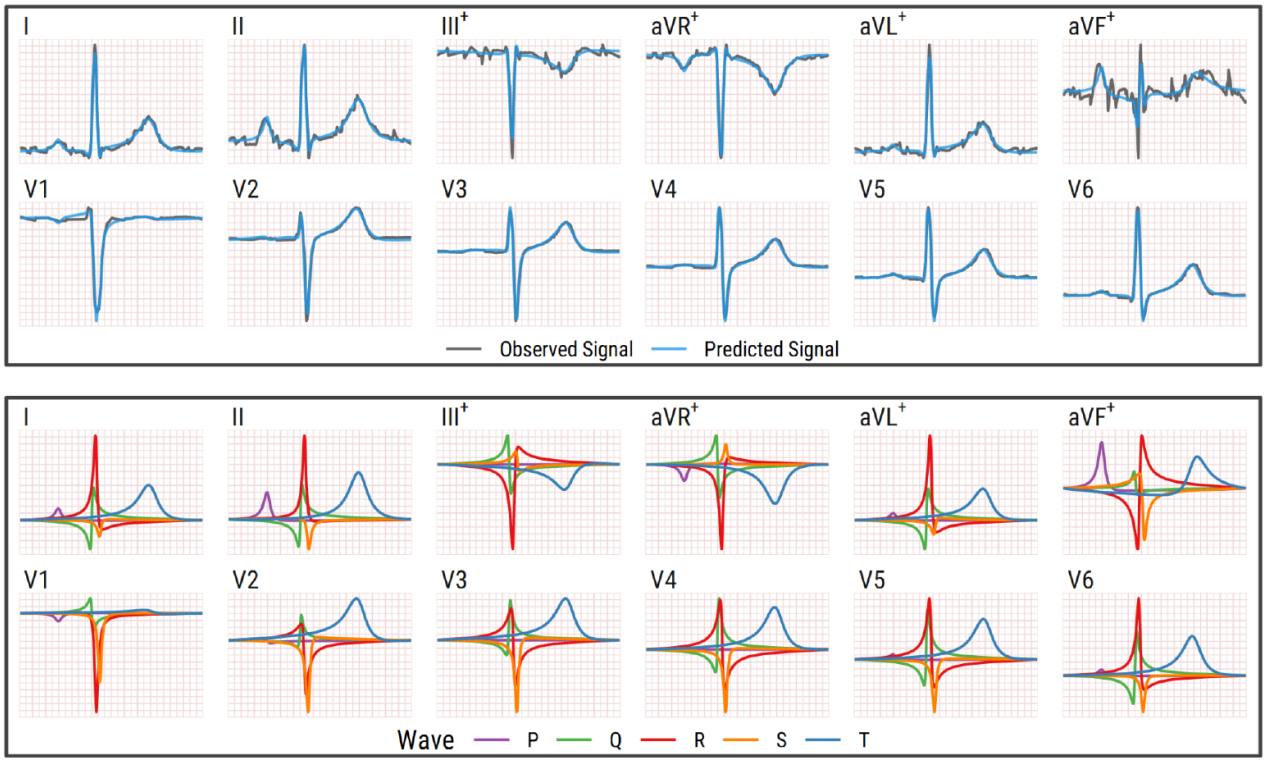


Figura 2.9: Ejemplo ilustrativo del ajuste del modelo 3DFMM$_{ecg}$ sobre las 12 derivaciones de un ECG. Adaptado de [3].

Preprocesado

Las señales del ECG suelen contener gran cantidad de ruido, desviaciones y artefactos que pueden afectar gravemente a la calidad de su análisis. Es por ello que se necesita una etapa previa de preprocesado para obtener fragmentos del ECG fiables y dividirlos latido a latido. Los distintos pasos de este proceso son:

1. **Eliminación de la línea base**

   Para cada uno de los fragmentos del ECG, se aplica un filtro de baja frecuencia basado en el ajuste de un modelo de regresión polinómica. Restando estas estimaciones de cada una de las señales se consigue corregir las posibles desviaciones.

2. **Detección mono-derivación de complejos QRS**

   Se realiza un análisis independiente mediante el algoritmo de Pan Tompkins [15] para cada una de las señales, y así detectar los complejos QRS de cada una de las derivaciones individualmente. Adicionalmente, se emplea la mediana de los intervalos RR (tiempo entre cada par de complejos QRS) para eliminar posibles falsos positivos, es decir, los complejos QRS demasiado largos o demasiado cortos.

3. **Detección multi-derivación de complejos QRS**

   Las detecciones obtenidas del paso anterior son combinadas para obtener un conjunto de anotaciones de referencia común. Las detecciones individuales se ordenan incrementalmente para construir grupos de al menos cuatro detecciones adyacentes de manera que los grupos estén lo suficientemente lejos los unos de los otros. A continuación, se calcula la mediana de cada uno de los grupos, llamada t^{QRS}. Con esto se consigue reducir el número de falsos positivos y de complejos no detectados en alguna de las derivaciones. A partir de este punto, solo se trabaja con las derivaciones del conjunto $Lred = \{I, II, V1, V2, V3, V4, V5, V6\}$.

4. **Segmentación del ECG y revisión**

   Cada uno de los latidos segmentados se obtienen a través de $[t^{QRS} - 40\,\%RR,\ t^{QRS} + 60\,\%RR]$. Por otro lado, se lleva a cabo un último filtrado para eliminar los latidos insignificantes:

   - Los primeros y últimos latidos que sean notablemente inferiores que la mediana de los intervalos RR o con una proporción de observaciones constante en cualquiera de los extremos.
   - Los latidos para los cuales t^{QRS} se sitúa fuera de un 35 %-45 % de la longitud del latido.
   - Los latidos con una gran diferencia entre los puntos finales de la señal con respecto a la amplitud del dichos latidos.
   - Los latidos con problemas de conductividad cuya amplitud se exceda de los límites estándar o presenten anomalías de amplitud respecto a la señal completa.

   Se considera que el ECG podrá analizarse correctamente y no se descartará por tener valores inaceptables de ruido o distorsión si hay al menos:

   - Tres latidos en el conjunto de anotaciones de referencia.

- Tres derivaciones con más de tres latidos.
- Un 20 % de los latidos esperados de acuerdo con la frecuencia de muestreo.

La salida del preprocesado está compuesta por las anotaciones t^{QRS} y por la segmentación de latidos del ECG.

Algoritmo de identificación y estimación

El problema de identificación de ondas se reduce a resolver el siguiente problema de optimización:

$$\arg\min_{\theta\in\Theta} \sum_{L\in Lset} \frac{1}{\sigma^L} \sum_{i=1}^{n} \{X^L(t_i) - [M^L + \sum_{J\in\{P,Q,R,S,T\}} A_J^L W(t_i, \alpha_J, \beta_J^L, \omega_J)]\}^2 \tag{2.13}$$

donde θ es el vector de parámetros del modelo y Θ es el espacio paramétrico. Para un ECG atípico o con mucho ruido, Θ puede ser más reducido para conseguir una correcta identificación de las ondas. Tanto σ^L como $L \in Lset$ son identificados durante el proceso de identificación. Ocasionalmente se puede haber descartado alguna derivación en la etapa de preprocesado debido a artefactos ruidosos. Se requiere información sobre al menos una de las derivaciones I, II, V2 o V5. A continuación, se inicia un proceso iterativo MI con dos etapas diferenciadas: el paso M en el que se realiza la estimación de las ondas y el paso I en el que se asignan las ondas. El algoritmo finaliza cuando no se obtienen diferencias significativas en la función objetivo del problema de optimización. La Figura 2.10 muestra esquemáticamente mediante un diagrama de flujo el funcionamiento del algoritmo.

Paso M: El paso M obtiene al menos 5 ondas FMM mediante un algoritmo de *backfitting* ajustado simultáneamente sobre todas las derivaciones de *Lred*. Típicamente son 5 ondas, pero en presencia de ruido o morfologías patológicas, las ondas de interés pueden estar ocultas y se pueden necesitar más ondas. Los valores σ^L se inicializan a 1 y son actualizados en cada iteración del algoritmo, promediando las diferencias cuadráticas entre los valores esperados y los valores predichos por el modelo. Los estimadores finales de M^L, A_J^L y B_J^L se obtienen resolviendo un problema de regresión lineal múltiple.

Paso I: La onda R se asigna en primer lugar, siendo aquella de mayor variabilidad explicada entre las más cercanas al t^{QRS} y teniendo además una deflexión positiva en las derivaciones I o II y un pico negativo en la derivación V2. A continuación, se preasignan las ondas P, Q, S y T usando $\alpha_P \leq \alpha_Q \leq \alpha_R \leq \alpha_S \leq \alpha_T$. En la mayoría de los casos esta preasignación se corresponde con la asignación final, aunque pueden requerirse reasignaciones estableciendo umbrales sobre los parámetros principales del modelo. En concreto, los componentes ruidosos son detectados por poseer valores muy bajos o muy altos de ω.

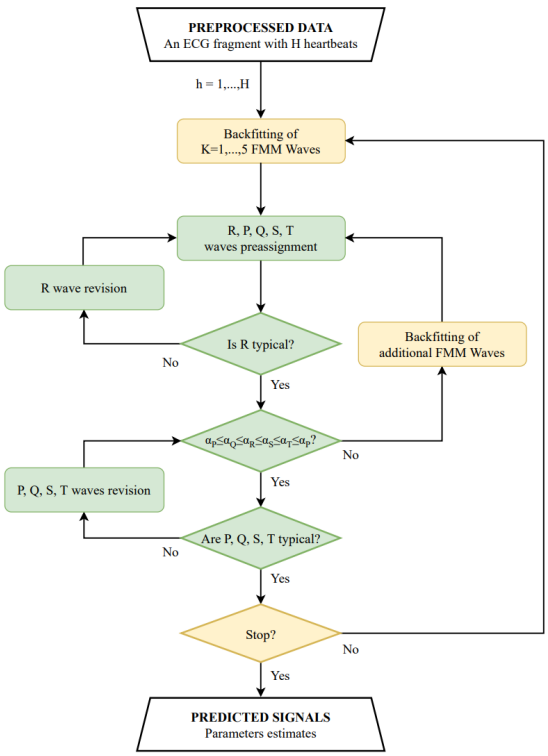


Figura 2.10: Diagrama de flujo del algoritmo de identificación del modelo 3DFMM$_{ecg}$. Los bloques amarillos se corresponden con el paso M y los verdes con el paso I. Adaptado de [3].

Medidas de evaluación

Para estudiar el ajuste de este modelo, hay que confeccionar una métrica que tenga en cuenta todos los latidos de las 12 derivaciones del ECG. En el artículo original se propone una medida de calidad global $\bar{R} \in [0, 1]$. Esta medida se puede interpretar como la proporción de variabilidad de la señal que es explicada a partir de los valores predichos. Cuanto más próxima sea a 1, mejor será el modelo. Se formula como:

$$\bar{R} = \frac{1}{12} \sum_{L \in Lset} median(R^2_{L_b}) \tag{2.14}$$

donde $R^2_{L_b}$ es el coeficiente de determinación (R cuadrado) para un latido b de una derivación L con valor medio $\overline{X}^L_b$:

$$R^2_{L_b} = 1 - \frac{\sum_{i=1}^{n} (X^L_b(t_i) - \widehat{X^L_b}(t_i))^2}{\sum_{i=1}^{n} (X^L_b(t_i) - \overline{X}^L_b)^2} \tag{2.15}$$

Por otro lado, PRD es otra métrica ampliamente usada en la literatura relativa a la compresión de datos:

$$PRD_{L_b} = \sqrt{\frac{\sum_{i=1}^{n} (X^L_b(t_i) - \widehat{X^L_b}(t_i))^2}{\sum_{i=1}^{n} (X^L_b(t_i) - \overline{X}^L_b)^2}} = \sqrt{1 - R^2_{L_b}} \tag{2.16}$$

2.3. Estado del arte

Desde hace años, se han aplicado diferentes enfoques para abordar el problema de la digitalización de ECG. Muchos de los procedimientos de digitalización de señales ECG disponibles tienen por objetivo el cálculo del ritmo cardíaco (o intervalo entre latidos). Con este objetivo, encontramos trabajos como [16] y [17], que miden la distancia entre los diferentes picos cardíacos y su número a lo largo del intervalo de tiempo que se está midiendo. Otros trabajos digitalizan señales de ECG con fines específicos, como [18], que presenta un conjunto de herramientas basadas en Matlab para extraer ciertos parámetros ampliamente utilizados en cardiología [19], incluyendo la duración del intervalo PR, el intervalo QT y el complejo QRS. Alternativamente, algunos autores ofrecen herramientas que presentan un enfoque más amplio, tratando de proporcionar información general de la señal ECG digitalizada que puede ser posteriormente analizada, como [20]. Concretamente, en este trabajo se propone un software basado en Matlab que sí que extrae las señales completas, pero no realiza una segmentación de las señales en las respectivas derivaciones, ni distingue los pulsos de referencia. Alineado con este artículo, en [21], los autores presentan también un software de Matlab, en este caso capaz de digitalizar señales de ECG en papel, obteniendo señales de ECG digitales. Sin embargo, cuenta con la limitación de que el usuario tiene que recortar manualmente las derivaciones del ECG. Por otro lado, también se ha explorado el uso de redes neuronales e inteligencia artificial en la digitalización de ECG. Trabajos como [22] se centran en la digitalización del ECG aprovechando algoritmos de binarización y diagnóstico basados en técnicas de *deep learning*.

Cabe destacar que se ha observado de manera generalizada que los conjuntos de ECG usados como validación en todos los trabajos anteriormente descritos son de un tamaño relativamente reducido (de apenas cientos en el mejor caso). Además, dichos datos suelen ser privados o se han seleccionado frente a otros sin un criterio justificado, lo cual dificulta enormemente una comparación objetiva de los resultados.

Capítulo 3

Planificación

En este capítulo se describe el modelo de desarrollo del proyecto, la planificación inicial, el camino crítico y el seguimiento realizado sobre la planificación inicial. Además, se presenta un análisis de riesgos y un presupuesto detallado del coste estimado que supone la realización del proyecto.

3.1. Modelo de desarrollo

Los modelos de desarrollo de software recogen una serie de técnicas y marcos de trabajo a la hora de desarrollar un sistema informático. El llamado ciclo de vida de un proyecto (SDLC) describe todos los procesos de un proyecto de software desde la propia planificación del mismo hasta, incluso, su mantenimiento.

En este caso, se ha optado por el enfoque clásico del modelo en cascada [23, 24], el cual consiste en un procedimiento lineal caracterizado por realizar una división en etapas secuenciales. A diferencia de los modelos iterativos, cada una de las fases se ejecuta una única vez. Los resultados de cada una de las etapas sirven como punto de partida para la siguiente.

Se ha decidido optar por este enfoque y no por otro tipo de metodologías, como las ágiles (las cuales se centran en adaptarse de forma flexible a cambios constantes) por la clara definición inicial del objetivo del proyecto y las tareas en que lo componen. La complejidad de este trabajo reside en el correcto desarrollo algorítmico y la calidad de los resultados, más que en cubrir un amplio espectro de funcionalidades y casos de uso. Además, este modelo destaca por su sencillez y facilidad de aplicación, lo cual simplifica enormemente la planificación.

3.1.1. Planificación inicial

Se ha planeado inicialmente una duración de 75 días laborales a media jornada (4 horas por día trabajado) desde el inicio del proyecto el 17 de febrero de 2023 hasta su finalización el 1 de junio de 2023, para así cumplir con los objetivos y ajustarse a la carga de trabajo establecida por el plan de estudios de 12 créditos ECTS, equivaliendo un crédito ECTS a 25 horas de trabajo. En la Figura 3.1 se muestra el listado con la planificación inicial de las tareas y en la Figura 3.2 se muestra el diagrama de Gantt asociado a las mismas.

Nombre de tarea	Duración	Comienzo	Fin
1 TFG	**75 días**	**17/02/2023**	**01/06/2023**
1.1 Inicialización	**10 días**	**17/02/2023**	**02/03/2023**
1.1.1 Fijado de objetivos	2 días	17/02/2023	19/02/2023
1.1.2 Planificación	6 días	20/02/2023	26/02/2023
1.1.3 Revisión del estado del arte	4 días	27/02/2023	02/03/2023
1.2 Análisis y Diseño	**14 días**	**03/03/2023**	**22/03/2023**
1.2.1 Elicitación de requisitos	2 días	03/03/2023	06/03/2023
1.2.2 Modelado	6 días	07/03/2023	14/03/2023
1.2.3 Diseño de la interfaz	3 días	20/03/2023	22/03/2023
1.3 Implementación	**25 días**	**23/03/2023**	**26/04/2023**
1.3.1 Codificación	25 días	23/03/2023	26/04/2023
1.3.2 Documentación	25 días	23/03/2023	26/04/2023
1.4 Pruebas	**10 días**	**27/04/2023**	**10/05/2023**
1.4.1 Búsqueda de bases de datos de ECG	4 días	27/04/2023	02/05/2023
1.4.2 Batería de pruebas	4 días	27/04/2023	02/05/2023
1.4.3 Validación	6 días	03/05/2023	10/05/2023
1.5 Elaboración de la memoria	**65 días**	**03/03/2023**	**01/06/2023**
1.5.1 Redacción	53 días	03/03/2023	16/05/2023
1.5.2 Revisión final	12 días	17/05/2023	01/06/2023

Figura 3.1: Listado con la planificación inicial de las tareas del TFG.

En primer lugar, se llevará a cabo la tarea de inicialización, constituida a su vez por tres subtareas: fijar los objetivos del proyecto, realizar la propia planificación y revisar el estado del arte investigando acerca de los trabajos previos realizados sobre digitalización de electrocardiogramas. A partir de este punto, se comenzará a elaborar la memoria del trabajo, en la que se irán redactando todos los avances que se vayan haciendo. Paralelamente a esto, se comenzará con la etapa de análisis y diseño, en la que se hará una recopilación de los requisitos del sistema, se crearán los distintos modelos y se hará un diseño esquemático de la interfaz. Hecho esto, se procederá con la implementación, que consiste tanto en la codificación propiamente dicha del programa como en la correcta documentación de la misma. A continuación, se llevará a cabo la etapa de pruebas. Se preparará una batería de pruebas sencilla para comprobar la corrección del código. Mientras tanto, se procederá paralelamente a buscar en la literatura distintas bases de datos públicas de ECG para, consiguientemente, validar la precisión de la digitalización. Acabadas estas tareas, se dejará un margen para terminar de redactar todos los resultados obtenidos durante la experimentación y se concluirá con una revisión final de todo el trabajo.

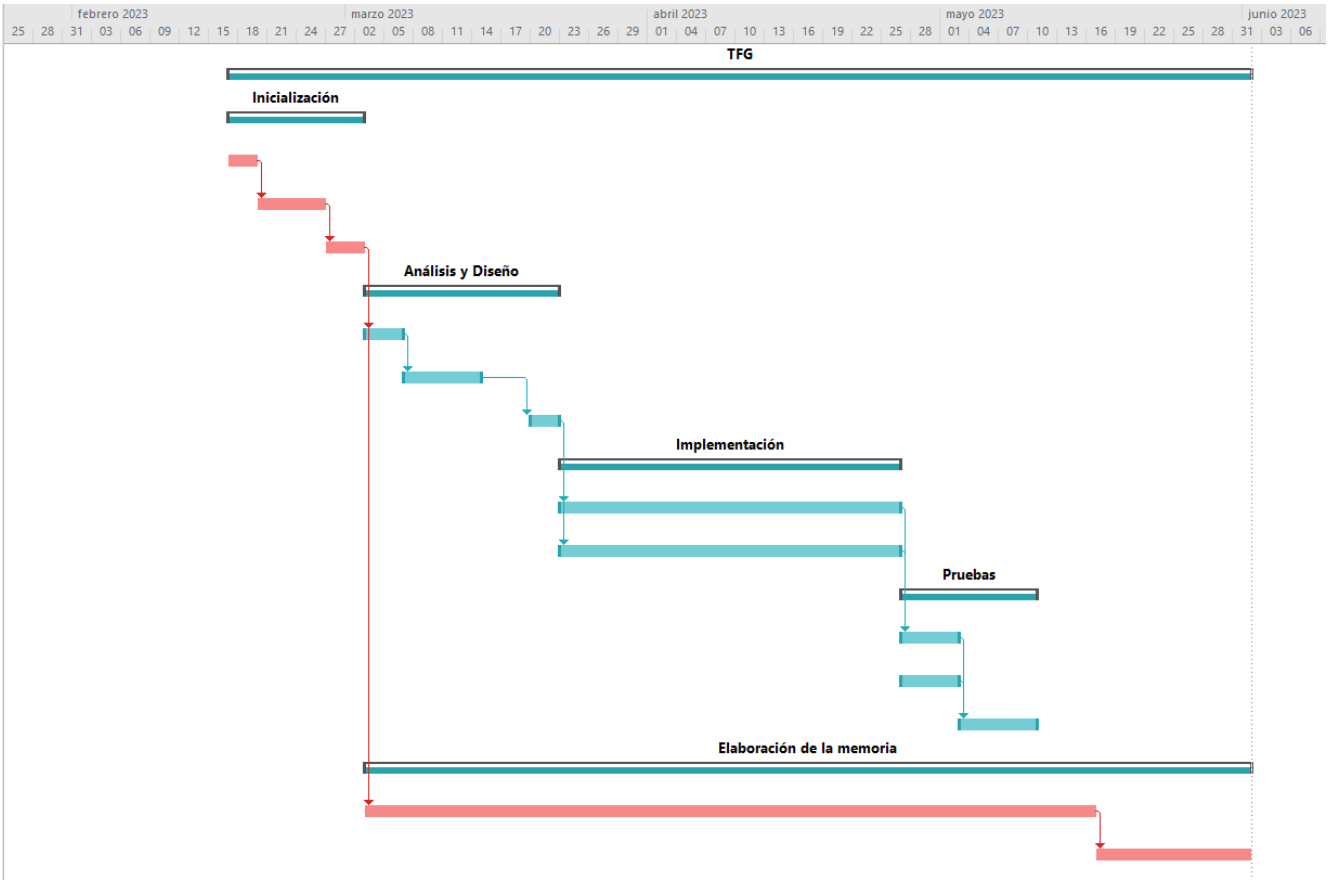


Figura 3.2: Diagrama de Gantt de la planificación inicial de las tareas del TFG.

3.1.2. Camino crítico

El camino crítico [25] es el camino de actividades más largo posible de la red de actividades, y marca por tanto la fecha de finalización del proyecto. Una demora en cualquiera de las actividades que lo constituyen genera automáticamente un retraso en la entrega. En el diagrama de Gantt mencionado anteriormente (ver Figura 3.2) se han marcado en rojo las tareas de este camino crítico. Está formado, en primer lugar, por las subtareas relativas a la inicialización. A continuación, se enlaza con la elaboración de la memoria, la cual es con diferencia la tarea que más se extiende en el tiempo, debido a que se pretende ir trabajando en ella mientras también se va construyendo la herramienta. Finalmente, se enlaza con la revisión final, ya que es la tarea que marca la fecha de finalización del proyecto.

3.1.3. Seguimiento

En términos generales, la duración real de las etapas del proyecto se ha correspondido casi en su totalidad con la planificación inicial estimada. Esto es fue, en parte, gracias a que se intentó dejar cierto margen de tiempo a cada tarea respecto a la duración que se preveía a priori que iba a tener. Incluso, la propia fecha de finalización del TFG se planeó hasta casi un mes antes del plazo de defensa de la convocatoria ordinaria. En la Figura 3.3 se muestra el listado con la temporalización real de las tareas y en la Figura 3.4 se muestra el diagrama de Gantt asociado las mismas. Tanto de las tareas de inicialización como de las tareas de análisis y diseño se respetaron las fechas de comienzo y fin. La implementación se terminó tres días antes de lo que se tenía previsto, el 23 de abril, por lo que se decidió aprovechar el tiempo para empezar a trabajar en las

pruebas anticipadamente. Por el contrario, sí que se produjo un retraso significativo en la tarea de validación, debido a ciertas complicaciones producidas por la complejidad de la muestra de ECG escogida y por la disponibilidad del cardiólogo que supervisó el proceso. Estos detalles se explicarán más en profundidad en el Capítulo 6. Se extendió otros siete días laborales, hasta el 28 de abril. La redacción de la memoria se alargó por tanto hasta el día 22 de abril y la revisión final hasta el 16 de junio, debido a que las tutoras sugirieron una gran cantidad de cambios para mejorar la calidad de la memoria.

Nombre de tarea	Duración	Comienzo	Fin
1 TFG	**86 días**	**17/02/2023**	**16/06/2023**
1.1 Inicialización	**10 días**	**17/02/2023**	**02/03/2023**
1.1.1 Fijado de objetivos	2 días	17/02/2023	19/02/2023
1.1.2 Planificación	6 días	20/02/2023	26/02/2023
1.1.3 Revisión del estado del arte	4 días	27/02/2023	02/03/2023
1.2 Análisis y Diseño	**14 días**	**03/03/2023**	**22/03/2023**
1.2.1 Elicitación de requisitos	2 días	03/03/2023	06/03/2023
1.2.2 Modelado	6 días	07/03/2023	14/03/2023
1.2.3 Diseño de la interfaz	3 días	20/03/2023	22/03/2023
1.3 Implementación	**23 días**	**23/03/2023**	**23/04/2023**
1.3.1 Codificación	23 días	23/03/2023	23/04/2023
1.3.2 Documentación	23 días	23/03/2023	23/04/2023
1.4 Pruebas	**17 días**	**24/04/2023**	**16/05/2023**
1.4.1 Búsqueda de bases de datos de ECG	4 días	24/04/2023	27/04/2023
1.4.2 Batería de pruebas	4 días	24/04/2023	27/04/2023
1.4.3 Validación	13 días	28/04/2023	16/05/2023
1.5 Elaboración de la memoria	**76 días**	**03/03/2023**	**16/06/2023**
1.5.1 Redacción	57 días	03/03/2023	22/05/2023
1.5.2 Revisión final	19 días	23/05/2023	16/06/2023

Figura 3.3: Listado con la temporalización real de las tareas del TFG.

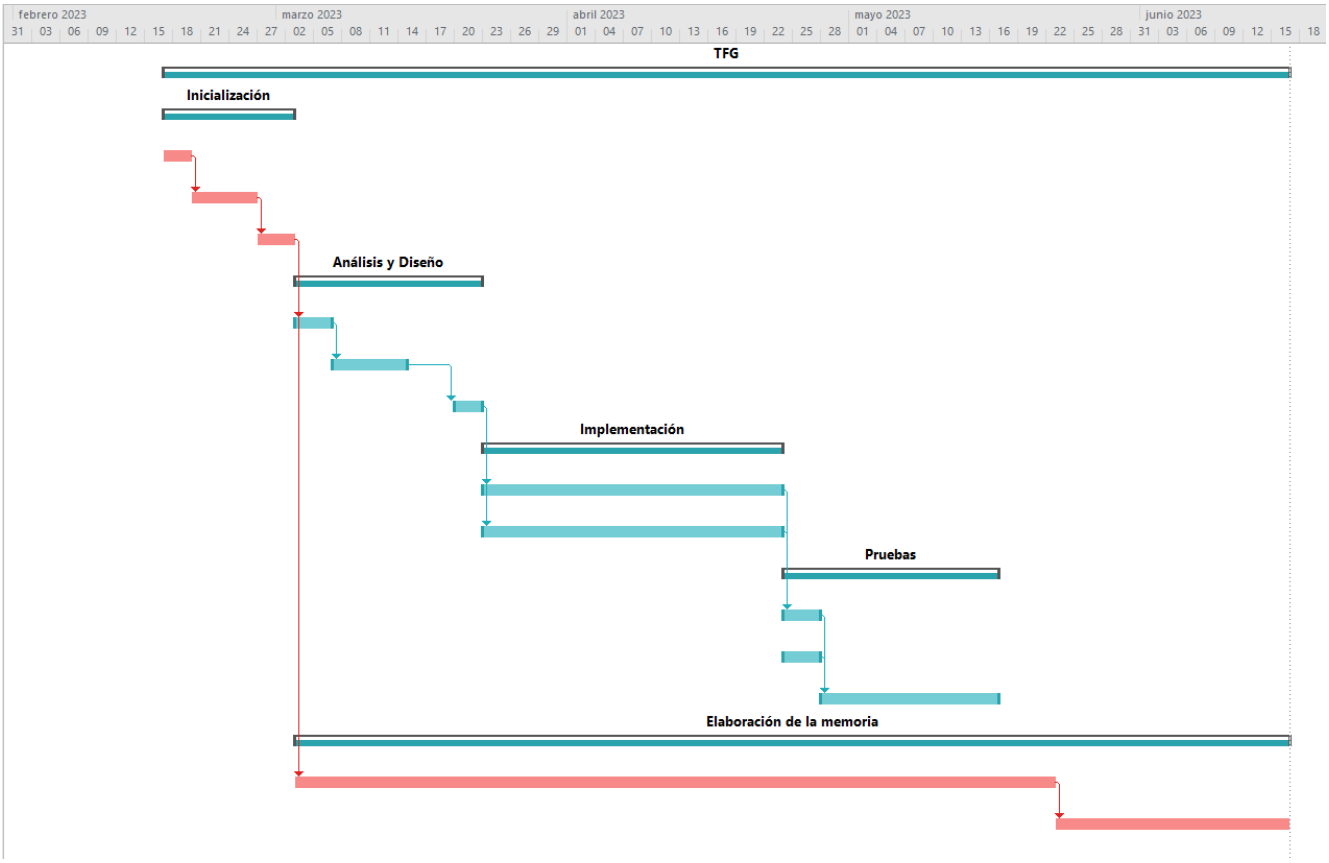


Figura 3.4: Diagrama de Gantt con el seguimiento de las tareas del TFG.

3.2. Análisis de riesgos

Durante el desarrollo de cualquier proyecto de software, es crucial identificar los riesgos potenciales que pueden surgir. Estos riesgos pueden afectar negativamente el éxito del proyecto, impactando en el coste y la calidad del software. Están caracterizados por su incertidumbre ya que, aunque bien es cierto que muchos de ellos pueden ser detectados precozmente, otros tantos pueden salir a la luz en plena fase de desarrollo si no se ha realizado un correcto análisis durante la etapa de planificación. Se pueden clasificar en dos grupos: los riesgos de negocio y los riesgos de proyecto.

Los riesgos de negocio son aquellos que pueden impactar el modelo de negocio y la rentabilidad del producto entregado. Los riesgos de proyecto son aquellos que ponen en peligro la planificación y ejecución del proyecto, lo que puede afectar a su calidad y fecha de finalización. Por el carácter del trabajo que se está presentando, se analizarán a continuación únicamente los riesgos de proyecto. Estos pueden a su vez categorizarse en cuatro subgrupos según su naturaleza (ver Figura 3.5).

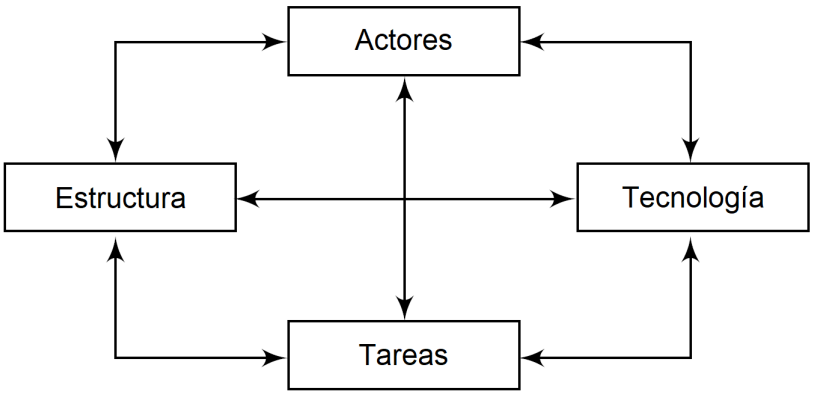


Figura 3.5: Marco de trabajo de riesgos de Lyynen–Mathiassen–Ropponen. Adaptado de [25].

- **Actores**: referidos a toda la gente involucrada en el desarrollo del software en cuestión.
- **Tecnología**: engloba tanto a la tecnología usada para implementar la aplicación como de la propia embebida dentro del producto final. Típicamente derivan de no adaptarse a las necesidades tecnológicas que más favorecen al producto que se desea desarrollar.
- **Estructura**: hace referencia a todos los sistemas y estructuras administrativas, incluyendo aquellas que afectan directamente a la planificación y al control.
- **Tareas**: relativo a todo el trabajo planeado y a las fechas de entrega.

Para cada riesgo, se estima la probabilidad de que se materialice y el impacto que puede provocar en la duración del proyecto en caso de que ocurra. Aunque para proyectos de mayor grado de complejidad se puedan emplear directamente los valores numéricos de estas dos magnitudes, se tomó la decisión de construir descriptores cualitativos para ganar un mayor grado de comprensión y legibilidad (ver Tabla 3.1).

Nivel de probabilidad	Rango
Alto	>50 % de ocurrencia
Significativo	30-50 % de ocurrencia
Moderado	10-29 % de ocurrencia
Bajo	<10 % de ocurrencia

(a) Niveles de probabilidad de riesgos.

Nivel de impacto	Rango
Alto	>30 % del tiempo
Significativo	20-29 % del tiempo
Moderado	10-19 % del tiempo
Bajo	<10 % del tiempo

(b) Niveles de impacto de riesgos.

Tabla 3.1: Descriptores categóricos de los distintos niveles de probabilidad e impacto de riesgos.

A continuación, se detallan los riesgos identificados durante la etapa de planificación, incluyendo una breve descripción de los mismos. Para reducir la probabilidad de ocurrencia y su impacto, se establecen los correspondientes planes de mitigación y contingencia:

ID	R01
Título	Baja médica por enfermedad
Categoría	Actores
Descripción	El desarrollador se encuentra indispuesto por algún tipo de patología que le impide trabajar.
Nivel de probabilidad	Bajo
Nivel de impacto	Significativo
Plan de mitigación	▪ Planificar cada tarea con cierto margen temporal respecto a la fecha máxima de entrega.
Plan de contingencia	▪ Replanificar las tareas, añadiéndoles mayor duración. ▪ Incrementar el esfuerzo en las tareas no acabadas tras la recuperación de la enfermedad.

Tabla 3.2: Riesgo R01: Baja médica por enfermedad.

ID	R02
Título	Mala comunicación con el equipo de tutorización
Categoría	Actores
Descripción	Una mala comunicación con el equipo de tutorización del proyecto que afecta al correcto desarrollo de la aplicación.
Nivel de probabilidad	Bajo
Nivel de impacto	Significativo
Plan de mitigación	▪ Planificar reuniones con frecuencia semanal y de forma presencial, cuando sea posible, para mejorar la comunicación y aclarar las dudas que vayan surgiendo. ▪ Utilizar medios de mensajería instantánea para agilizar el intercambio de mensajes.
Plan de contingencia	▪ Realizar más reuniones de las planificadas para tratar de solucionar los problemas que hayan ocurrido.

Tabla 3.3: Riesgo R02: Mala comunicación con el equipo de tutorización.

ID	R03
Título	Falta de formación del desarrollador
Categoría	Actor
Descripción	El desarrollador no tiene la suficiente formación para manejar con soltura todas las tecnologías de las que se hace uso en el proyecto.
Nivel de probabilidad	Moderado
Nivel de impacto	Moderado
Plan de mitigación	▪ Elegir tecnologías con las que el desarrollador esté familizarizado. ▪ Realizar un periodo de formación previo de aquellas herramientas con las que se haya trabajado menos.
Plan de contingencia	▪ Solicitar ayuda al equipo de tutoras o incluso a contactos externos que tengan experiencia con las herramientas. ▪ Buscar fuentes alternativas en caso de que la formación previa haya sido insuficiente.

Tabla 3.4: Riesgo R03: Falta de formación del desarrollador.

ID	R04
Título	Planificación poco realista
Categoría	Estructura
Descripción	Se ha planificado el desarrollo del proyecto de manera poco realista, asignando menor tiempo del necesario a las tareas.
Nivel de probabilidad	Moderado
Nivel de impacto	Alto
Plan de mitigación	▪ Revisar y comentar con las tutoras la planificación antes de comenzar con el desarrollo. ▪ Revisar trabajos previos de años previos para hacer una estimación de lo que puede durar este.
Plan de contingencia	▪ Aumentar la dedicación semanal. ▪ Retrasar la fecha de finalización del proyecto.

Tabla 3.5: Riesgo R04: Planificación poco realista.

ID	R05
Título	*Gold plating*
Categoría	Tareas
Descripción	Se han añadido funcionalidades que no son necesarias y/o no se han solicitado.
Nivel de probabilidad	Bajo
Nivel de impacto	Moderado
Plan de mitigación	▪ Establacer correctamente desde el inicio toda la funcionalidad que se desea añadir. ▪ Comentar con las tutoras los cambios que se van añadiendo y obtener retroalimentación de los mismos.
Plan de contingencia	▪ Preguntar a individuos ajenos al proyecto sobre su opinión acerca de si consideran que la funcionalidad extra puede tener alguna utilidad o no resulta de interés. ▪ Desechar la funcionalidad innecesaria.

Tabla 3.6: Riesgo R05: *Gold plating*.

ID	R06
Título	Cambios tardíos en los requisitos
Categoría	Estructura
Descripción	Se han modificado significativamente los requisitos del sistema en una fase avanzada del proyecto.
Nivel de probabilidad	Bajo
Nivel de impacto	Alto
Plan de mitigación	▪ Realizar varias reuniones previas con las tutoras. ▪ Hacer una extensa revisión del estado del arte para confirmar que no haya ninguna funcionalidad o requisito extra que se quiera añadir.
Plan de contingencia	▪ Aumentar la dedicación semanal. ▪ Retrasar la fecha de finalización del proyecto.

Tabla 3.7: Riesgo R06: Cambios tardíos en los requisitos.

ID	R07
Título	Resultados no acordes con lo esperado
Categoría	Tareas
Descripción	Los resultados de la experimentación no han sido satisfactorios, obteniéndose menor precisión de la esperada.
Nivel de probabilidad	Moderado
Nivel de impacto	Significativo
Plan de mitigación	▪ Revisar el estado del arte y hacer una recopilación de los distintos algoritmos que se han empleado y se ha demostrado que ofrecen buenos resultados. ▪ Pedir consejo a las tutoras acerca de los posibles problemas que pueda acarrear la metodología que se desea emplear.
Plan de contingencia	▪ Refinar los algoritmos utilizados e ir iterativamente comparando los resultados hasta obtener alguna mejora. ▪ Retrasar la fecha de finalización del proyecto.

Tabla 3.8: Riesgo R07: Resultados no acordes con lo esperado.

Una vez descritos todos los riesgos detectados, se elaboró una matriz de riesgos, en la cual se colocan cada uno de ellos atendiendo a su probabilidad e impacto (ver Figura 3.6). Las celdas de

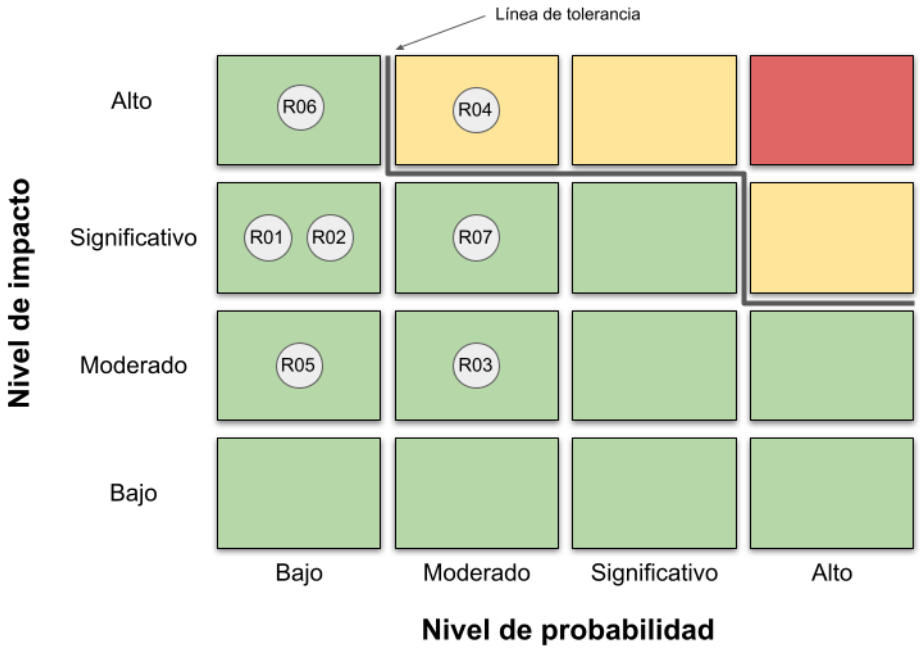


Figura 3.6: Matriz de riesgos del proyecto.

arriba a la derecha están delimitadas con una línea de riesgo. Los riesgos que se encuentren dentro de esta zona serán aquellos que pueden causar daños más graves en el proyecto y, por tanto, son a los que se les tiene que prestar mayor atención. Se observa que el único riesgo que sobrepasa esta línea es R04 ("Planificación poco realista"), debido a que es aquel que puede provocar un mayor retraso en la fecha de entrega del trabajo, y por tanto, se le deberá dar especial importancia.

3.3. Presupuesto

En esta sección se presenta una estimación del coste que tendría la realización del proyecto en el ámbito del mercado laboral. En la Tabla 3.9 se puede ver un resumen del presupuesto desglosado por concepto, así como el presupuesto total del proyecto, el cual asciende a 5646,64€.

En cuanto al coste de recursos humanos, hay que contabilizar un único sueldo de desarrollador de software. Según la página web de empleo *Indeed* [26], el sueldo promedio para un desarrollador de software junior en España es de 12,78€ la hora. Además de la nómina que recibe el empleado mes a mes, la cual constituye en torno a un 70 % del gasto de la empresa, hay que añadir el 30 % restante por la Seguridad Social, indemnizaciones, gasto en formación, etc. De esta forma se obtiene un coste añadido de 5,48€/hora, lo que hace un total de 18,26€ por cada hora trabajada. Así, para las 300 horas de elaboración del trabajo, el presupuesto para contratar al desarrollador sería de 5.478€.

Para el software, se utilizarán dos licencias profesionales de pago. Para la creación del diseño de la interfaz se empleará *Balsamiq Cloud* [27], cuyo precio mínimo es de 8,41€/mes, suponiendo un total de 33,64€ para los 4 meses de trabajo en los que se desarrollará el proyecto. Por otro lado, para el modelado del sistema se ha elegido *Astah Professional* [28], siendo su licencia de 7,50€/mes, lo cual supone un total de 30€. Para el resto de programas empleados, o bien se usará la versión gratuita, o bien son de uso gratuito directamente.

Finalmente, en cuanto al hardware, como ordenador de trabajo se utilizará un ordenador de sobremesa personalizado valorado en 1.575€, con una vida útil de 5 años. Cuenta con 16 GB de RAM DDR4 3200MHz, una CPU AMD Ryzen™ 5 3600 @ 3.60 GHz y una tarjeta gráfica RTX 2060 SUPER. El coste amortizado se calcula dividiendo el precio total de la máquina entre el número de meses de vida útil (60 meses), lo que da un precio de 26,25€/mes, haciendo un total de 105€ para el proyecto completo durante los 4 meses.

Concepto	**Precio**	**Tiempo**	**Total**
Desarrollador	18,26€/hora	300 horas	5.478€
Licencia Balsamiq Cloud	8,41€/mes	4 meses	33,64€
Licencia Astah Professional	7,50€/mes	4 meses	30€
PC sobremesa	26,25€/mes	4 meses	105€
			5.646,64€

Tabla 3.9: Presupuesto estimado para la realización del proyecto en el ámbito del mercado laboral.

Capítulo 4

Análisis y diseño

En este capítulo se describe el análisis, en particular, la elicitación de requisitos (funcionales y no funcionales). Por otro lado, se describe el diseño de la herramienta presentando las tecnologías empleadas, la arquitectura y el diseño detallado del sistema, el diseño del algoritmo de digitalización y el diseño de la interfaz gráfica de usuario.

4.1. Análisis

La etapa de elicitación de requisitos es la primera etapa a la hora de abstraer un problema de software para su correcta resolución. Se considera una parte crucial del desarrollo de software, ya que establece las expectativas y las especificaciones del mismo. En esencia, los requisitos son un conjunto de características, capacidades y comportamientos que deben estar presentes para cumplir con las necesidades del usuario final. Los requisitos descritos a lo largo de esta sección se han basado en la retroalimentación recibida por el grupo de investigación. A continuación se presentan los requisitos funcionales y no funcionales que debe satisfacer la herramienta desarrollada.

4.1.1. Requisitos funcionales

Los requisitos funcionales son aquellos que describen qué es lo que el software debe hacer para que funcione correctamente, así como las características específicas que deben satisfacerse. El software ECGMiner debe satisfacer los requisitos funcionales que se describen a continuación.

ID	RF01
Título	Digitalizar las 12 derivaciones del ECG
Descripción	El sistema deberá digitalizar las 12 derivaciones de una imagen de ECG, tanto en formato estándar como en formato Cabrera, y almacenar los valores numéricos de cada una de las señales. También soportará disposiciones 3x4, 6x2 y 12x1 y deberá poder digitalizar hasta tres tiras de ritmo.

Tabla 4.1: Requisito funcional RF01: Digitalizar las 12 derivaciones del ECG.

ID	RF02
Título	Extraer metadatos
Descripción	El sistema deberá ofrecer la opción de extraer los posibles metadatos escritos en el marco que rodea a la rejilla del ECG.

Tabla 4.2: Requisito funcional RF02: Extraer metadatos.

ID	RF03
Título	Calcular la traza de la digitalización
Descripción	El sistema deberá calcular la traza de la digitalización, resaltando con distintos colores cada una de las señales digitalizadas sobre una copia de la imagen original del ECG.

Tabla 4.3: Requisito funcional RF03: Calcular la traza de la digitalización.

ID	RF04
Título	Visualizar traza de la digitalización
Descripción	El sistema deberá ser capaz de mostrar la traza de los ECG cuya digitalización haya finalizado.

Tabla 4.4: Requisito funcional RF04: Visualizar traza de la digitalización.

ID	RF05
Título	Cancelar digitalización
Descripción	El sistema deberá ser capaz de cancelar la digitalización, de manera que solo se terminen de digitalizar los ECG cuyo proceso se inició previamente al momento de la cancelación.

Tabla 4.5: Requisito funcional RF05: Cancelar digitalización.

ID	RF06
Título	Cambiar ruta del directorio de salida
Descripción	El sistema deberá ofrecer la opción de cambiar la ruta del directorio de salida de los ficheros resultantes de la digitalización.

Tabla 4.6: Requisito funcional RF06: Cambiar ruta del directorio de salida.

ID	RF07
Título	Interpolar señales
Descripción	El sistema deberá ofrecer la opción de interpolar las señales a un número de observaciones concretas.

Tabla 4.7: Requisito funcional RF07: Interpolar señales.

ID	RF08
Título	Notificar al usuario si hay un error
Descripción	El sistema debe verificar que la imagen a digitalizar es un ECG con un formato válido; en caso contrario, deberá notificar al usuario.

Tabla 4.8: Requisito funcional RF08: Notificar al usuario si hay un error.

4.1.2. Requisitos no funcionales

Los requisitos no funcionales describen cómo debe funcionar el sistema. Estos requisitos se centran en los aspectos de rendimiento, seguridad, confiabilidad y usabilidad, entre otros. Los requisitos no funcionales son críticos para garantizar que el software sea eficaz y cumpla con las expectativas del usuario final. La herramienta ECGMiner debe satisfacer los requisitos no funcionales descritos a continuación.

ID	RNF01
Título	Código abierto
Descripción	El sistema deberá ser de código abierto, permitiendo a otros usuarios usarlo bajo licencia MIT.

Tabla 4.9: Requisito no funcional RNF01: Código abierto.

ID	RNF02
Título	Formatear las señales en CSV
Descripción	El formato del fichero de salida que contenga las señales del ECG será CSV.

Tabla 4.10: Requisito no funcional RNF02: Formatear las señales en CSV.

ID	RNF03
Título	Formatear la traza de digitalización en PNG
Descripción	El formato del fichero de salida que contenga la imagen de la traza de digitalización del ECG será PNG.

Tabla 4.11: Requisito no funcional RNF03: Formatear la traza de digitalización en PNG.

ID	RNF04
Título	Formatear los metadatos en TXT
Descripción	El formato del fichero de salida que contenga los metadatos del ECG será TXT.

Tabla 4.12: Requisito no funcional RNF04: Formatear los metadatos en TXT.

ID	RNF05
Título	Digitalizar múltiples ECG
Descripción	El sistema deberá permitir que el usuario seleccione más de una imagen ECG a digitalizar, en lugar de realizar una solicitud de digitalización por cada una de dichas imágenes.

Tabla 4.13: Requisito no funcional RNF05: Digitalizar múltiples ECG.

ID	RNF06
Título	Digitalizar eficientemente
Descripción	El sistema deberá digitalizar de manera eficiente, con tiempos de menos de 10 segundos por ECG, disponiendo de un ordenador personal con un procesador AMD Ryzen 5 3600 o superior y al menos 16 GB de RAM.

Tabla 4.14: Requisito no funcional RNF06: Digitalizar eficientemente.

ID	RNF7
Título	Utilizar Python como lenguaje de desarrollo
Descripción	El sistema deberá ser desarrollado con Python.

Tabla 4.15: Requisito no funcional RNF7: Desarrollar con Python.

4.2. Diseño

En la fase de diseño se ha decido qué tecnologías utilizar, la arquitectura, el algoritmo y la interfaz gráfica de usuario. A continuación se detallan las decisiones de diseño relativas a cada uno de estos aspectos.

4.2.1. Tecnologías empleadas

Para la correcta elaboración de este proyecto se han utilizado las distintas herramientas y tecnologías que se describen a continuación, clasificadas en tres grupos, según su utilidad: documentación/diseño, control/gestión y desarrollo.

- **Herramientas de documentación y diseño**
  - **Astah**: es una herramienta de modelado de software mediante la cual se pueden elaborar diagramas UML y otro tipo de diagramas de software. Una de las ventajas de Astah es que es fácil de usar y ofrece una amplia variedad de plantillas de diagramas. Se ha empleado para elaborar todos los diagramas y modelos de diseño presentados en este documento.
  - **Balsamiq**: es una herramienta de diseño de interfaz de usuario (UI) que permite crear bocetos y prototipos de interfaces gráficas de usuario de forma rápida y sencilla. Se ha utilizado para diseñar el boceto de la interfaz gráfica de usuario de la herramienta.

- **Herramientas de control y gestión**

  - **Git**: es una herramienta gratuita y de código abierto de control de versiones. Permite añadir y modificar cambios sobre un proyecto de manera rápida y eficaz. Está basado en un sistema de ramas, lo cual permite añadir cambios completamente independientes entre sí para probar ideas, corregir errores, etc. Se ha empleado para mantener distintas versiones de ECGMiner, hacer un seguimiento de las mismas e ir integrando y combinando correctamente los distintos cambios realizados sobre el código.
  - **GitHub**: es un servicio en la nube basado en Git que aloja un sistema de control de versiones para el desarrollo colaborativo de software. Esta herramienta se ha usado a modo de repositorio remoto del software, debido a que ECGMiner es de código abierto y libre acceso.

- **Herramientas de desarrollo**

  - **Python**: es un lenguaje de programación interpretado de alto nivel que es muy popular en el mundo de la informática, especialmente en el ámbito de la inteligencia artificial. Es conocido por su facilidad de uso y su amplia variedad de bibliotecas. Para el desarrollo de este proyecto, se ha usado concretamente Python 3.10, debido a que es una versión reciente y es compatible con las distintas bibliotecas utilizadas.
  - **Qt**: es un marco de trabajo para el desarrollo de interfaces gráficas de usuario multiplataforma. Permite a los desarrolladores crear aplicaciones que funcionan en diferentes sistemas operativos, como Windows, macOS y Linux. Para crear la interfaz gráfica de usuario de ECGMiner se ha usado la adaptación de esta librería para Python denominada PyQt.
  - **OpenCV**: es una biblioteca de código abierto que facilita a los desarrolladores el procesamiento de imágenes y vídeo para crear aplicaciones de visión por computadora. Se ha utilizado para el tratamiento de las imágenes de los ECG.
  - **Tesseract OCR**: es un motor de reconocimiento óptico de caracteres de código abierto. Permite a los desarrolladores crear aplicaciones que pueden reconocer texto en imágenes y escanear documentos. Una de las ventajas de Tesseract es que es muy preciso en la detección de texto y es fácil de usar como una biblioteca de terceros en otros proyectos de programación. Se ha utilizado para ofrecer la funcionalidad de extraer los metadatos de los ECG.

4.2.2. Diseño de la arquitectura

MVC (Modelo-Vista-Controlador) [29] es un patrón arquitectónico comúnmente utilizado en ingeniería de software para modelar interfaces gráficas de usuario. Este patrón enfatiza la separación entre la lógica de negocio y su visualización, lo cual produce un desacoplamiento que mejora la división del trabajo y la mantenibilidad del sistema. Está formado por los siguientes elementos:

- **Modelo**: representa los datos del dominio y su respectiva lógica de negocio asociada. También se encarga de gestionar el almacenamiento y la recuperación de los datos persistentes.

- **Vista**: se encarga de generar la interfaz de la aplicación, en la cual se representa la información que se le muestra al usuario.

- **Controlador**: actúa como intermediario entre la vista y el modelo, regulando el flujo de información entre ambos. Se encarga de gestionar los eventos generados en la vista por las acciones del usuario.

Se ha utilizado la variante pasiva de este patrón, en el cual el controlador manipula el modelo de forma exclusiva, produciéndose así un mayor desacoplamiento entre los componentes. La variante activa sin embargo sí que permite que el modelo notifique los cambios a la vista o al grupo de vistas.

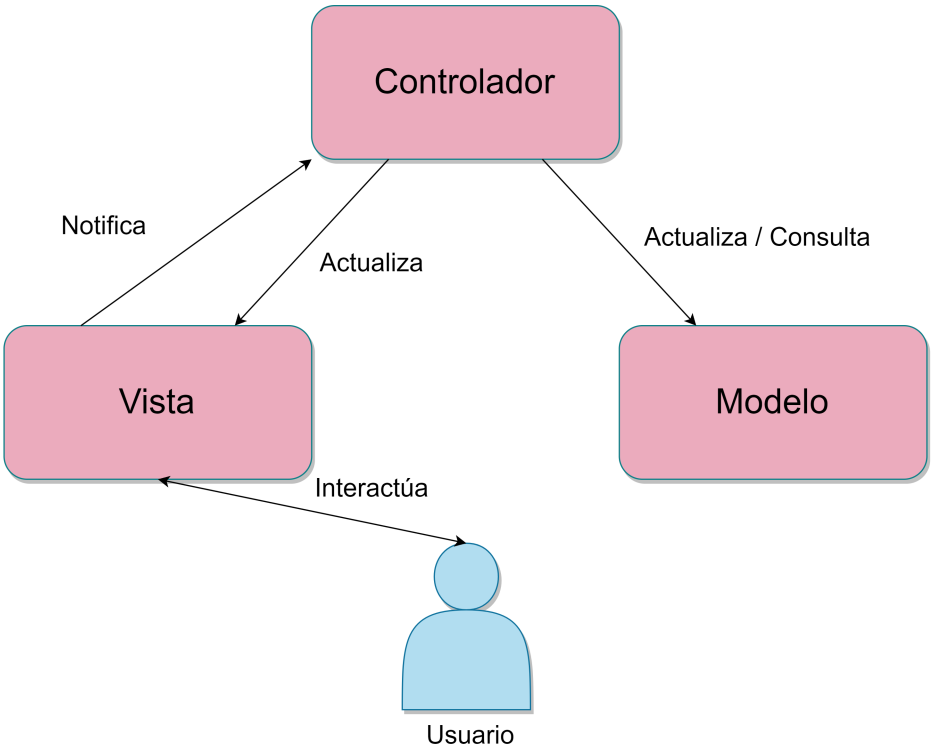


Figura 4.1: Esquema del patrón arquitectónico MVC pasivo.

En cuanto a la arquitectura de la aplicación, ésta se divide en tres paquetes principales:

- **app**: en este paquete se encuentran todas las clases relativas a la aplicación y a los respectivos componentes que conforman el patrón MVC.

- **digitization**: en este paquete se encuentran las clases relativas al algoritmo de digitalización, el cual se explicará detalladamente en la Sección 4.2.4.

- **utils**: en este paquete se encuentran las clases de utilidad, empleadas recurrentemente por varias clases de los otros paquetes.

La Figura 4.2 muestra la arquitectura general descrita y la Figura 4.3 las distintas dependencias (*uses style*) que existen entre los paquetes.

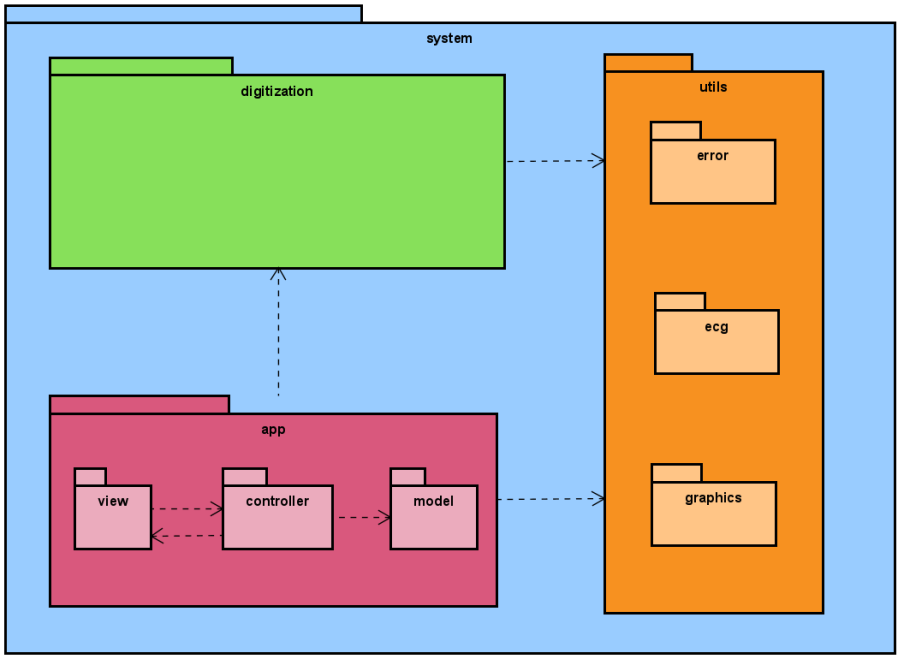


Figura 4.2: Arquitectura general del sistema.

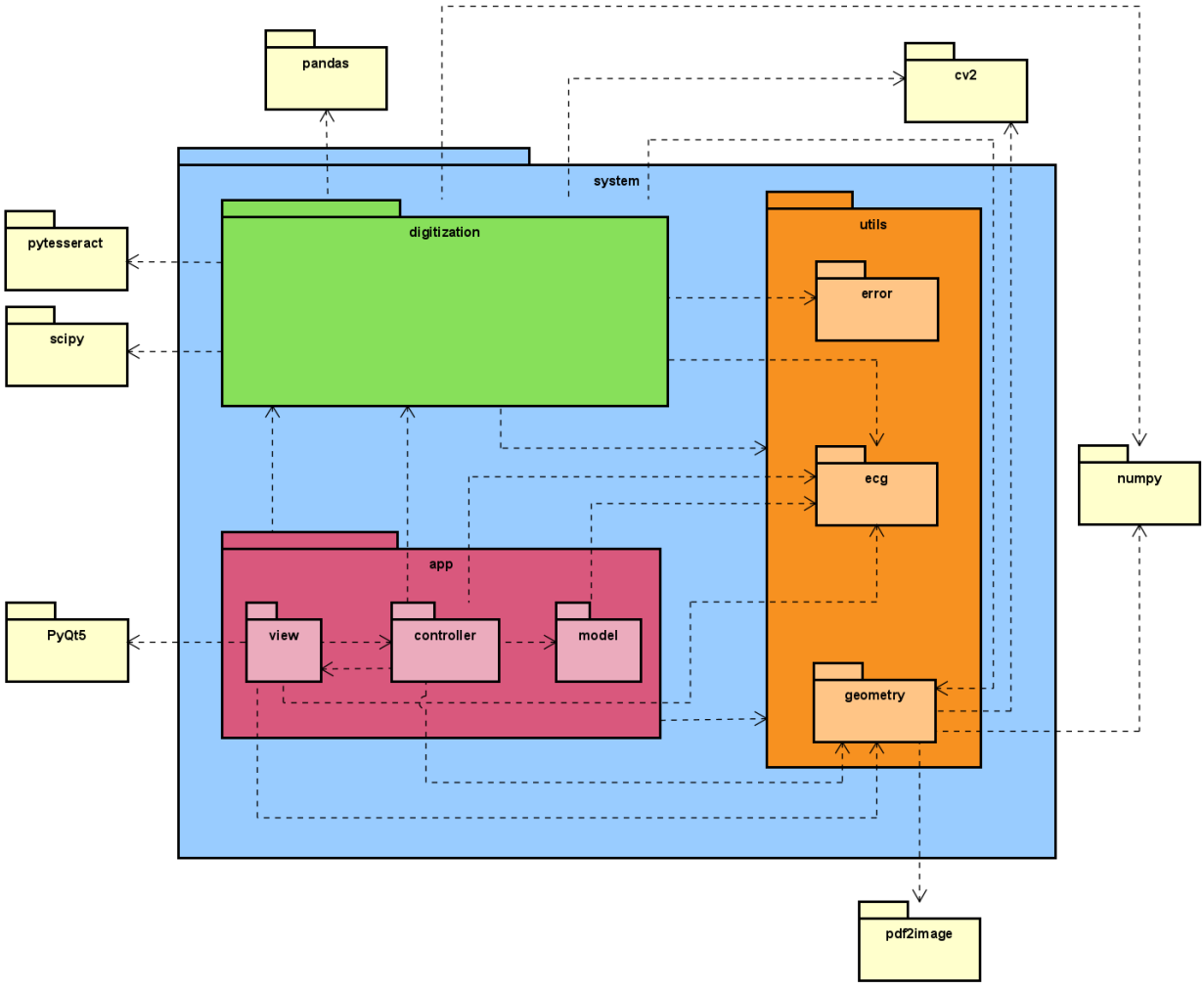


Figura 4.3: *Uses style* general del sistema.

4.2.3. Diseño detallado del sistema

El paquete `app` se divide, a su vez, en tres subpaquetes (ilustrado en la Figura 4.4):

- `model`: contiene la clase `Model`, la cual almacena toda la información de configuración y del estado de la digitalización.
- `view`: contiene la clase `View`, la cual contiene todos los elementos gráficos con los que interactúa el usuario y los eventos asociados a los mismos.
- `controller`: contiene la clase `Controller`, la cual se encarga de gestionar los eventos de la clase `View`.

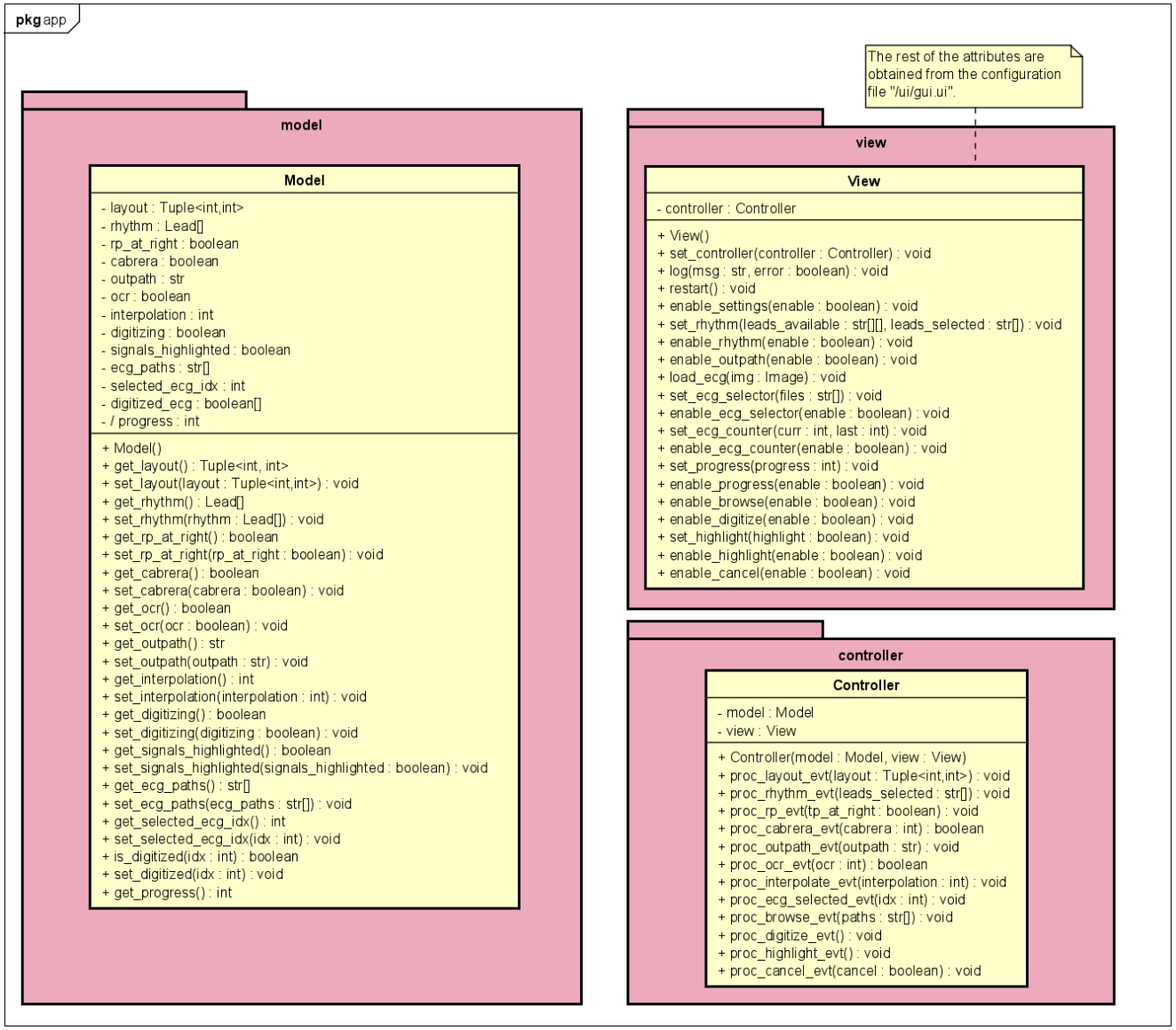


Figura 4.4: Diseño detallado del paquete `app`.

El paquete `digitization` tiene todas las clases referentes al algoritmo de digitalización per se: `Digitizer`, `Preprocessor`, `SignalExtractor`, `Postprocessor` y `MetadataExtractor`. La clase `Digitizer` actúa como Fachada (*Facade*) [29], es decir, proporciona una interfaz sencilla para usar el sistema, de manera que el usuario no se tenga que preocupar de inicializar individualmente los objetos y de ejecutar los métodos en el orden correcto, lo cual facilita enormente su comprensión y mantenibilidad. La Figura 4.5 ilustra el diseño de este paquete.

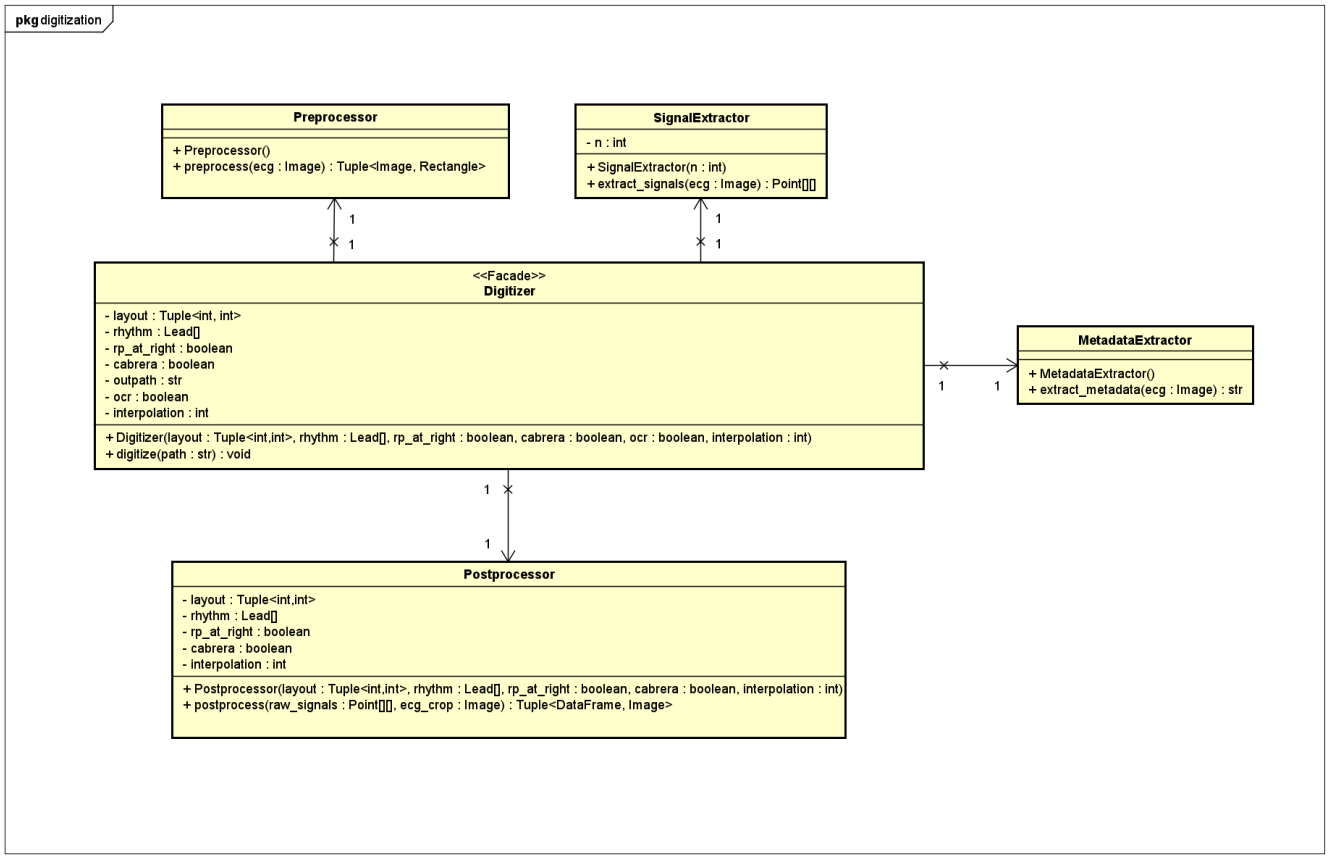


Figura 4.5: Diseño detallado del paquete `digitization`.

Finalmente, el paquete `utils` está dividido en tres subpaquetes (ilustrado en la Figura 4.6):

- `ecg`: contiene la enumeración `Lead` (con los nombres de las 12 derivaciones del ECG) y la clase `Format` (con dos métodos estáticos que permiten obtener la lista de derivaciones tanto en formato estándar como en formato Cabrera).

- `graphics`: contiene las distintas clases relativas a la información gráfica. En primer lugar, la clase `Image` es una representación abstracta de una imagen, la cual puede transformarse entre los distintos espacios de color definidos por la enumeración `ColorSpace`. Por otro lado, la clase `Point` almacena una tupla de coordenadas enteras (x,y). La clase `Rectangle` está conformada por dos puntos.

- `error`: contiene la clase `DigitizationError`, la cual es una excepción utilizada para gestionar los distintos errores que puedan ocurrir relativos a la digitalización.

Figura 4.6: Diseño detallado del paquete `utils`.

4.2.4. Diseño del algoritmo de digitalización

En esta sección se describen las distintas etapas que constituyen el proceso completo de digitalización, el cual se compone de:

1. **Preprocesado**: se reconoce la composición de la imagen. El algoritmo identifica qué parte de la imagen contiene la cuadrícula con las señales y elimina la rejilla, dejando solo las señales con un fondo limpio.

2. **Extracción de las señales**: se detectan las regiones de interés y se identifican las coordenadas de los píxeles que constituyen las señales.

3. **Posprocesado**: se detectan los pulsos de referencia y se segmentan las señales en cada una de las distintas derivaciones. Finalmente, se normaliza cada una de las señales y se almacenan en una estructura matricial.

4. **Extracción de metadatos (opcional)**: se extraen los metadatos situados en el marco exterior de la imagen.

En la Figura 4.7 se presenta un diagrama de flujo con los distintos pasos del algoritmo.

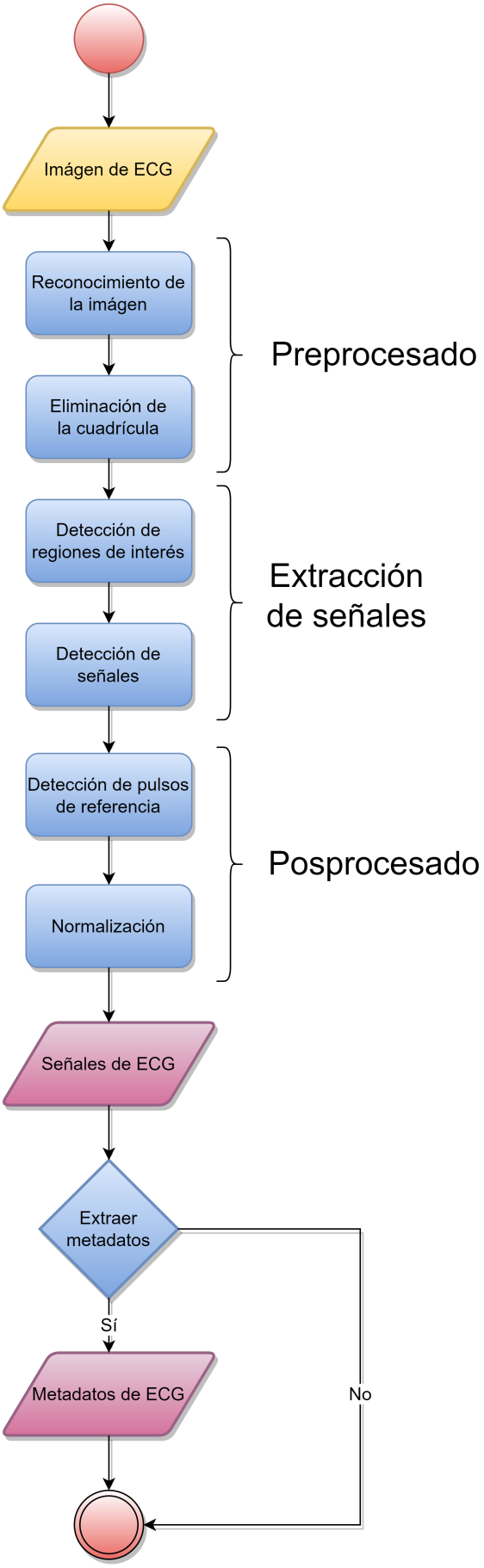


Figura 4.7: Diagrama de flujo del algoritmo de digitalización.

Preprocesado

Reconocimiento de la imagen

Esta primera fase del algoritmo tiene por objetivo identificar la ubicación de la información de metadatos y las señales trazadas en la imagen del ECG. Intuitivamente, se puede suponer que las señales se situarán dentro del área rectangular más grande de la imagen, el cual se detectará mediante los siguientes pasos:

1. Se simplifica la imagen extrayendo los bordes de la imagen para reducir la cantidad de datos a procesar. Esto se hace utilizando el operador Canny Edge [30], que utiliza un algoritmo de varias etapas para detectar los bordes de una imagen.

2. Se obtienen las regiones delimitadas para construir polígonos y descartar los bordes sueltos restantes. Esto se consigue aislando cada contorno aplicando el algoritmo de Suzuki [31], el cual define relaciones jerárquicas entre los bordes.

3. Se transforman los polígonos en rectángulos utilizando el algoritmo de Ramer-Douglas-Peucker [32, 33], el cual permite reducir el número de puntos utilizados en la aproximación de una curva.

4. Se toma el rectángulo con el área más grande como la región que contiene la rejilla con las señales.

Eliminación de la cuadrícula

Las líneas de cuadrícula entorpecen el proceso de digitalización y pueden inducir errores. Además, la cuadrícula contiene muchos puntos cuyo tratamiento podría ralentizar considerablemente el cálculo en las etapas posteriores. Así pues, una vez identificada la región rectangular, el objetivo es eliminar la cuadrícula, así como el ruido y los pequeños artefactos que pueda contener.

Basándose en el modelo de color HSV (*Hue*, *Saturation*, *Value*), una imagen puede verse como una matriz multidimensional; en particular, como una matriz $M \times N \times 3$, con M filas, N columnas y 3 canales, respectivamente asociados a *Hue*, *Saturation* y *Value*. El canal *Value* podría verse como la "fuerza del color"; los valores bajos se asocian a las señales (porque tienen colores oscuros) y el resto de elementos de la rejilla y otros artefactos toman valores más altos. Considerando gráficos en color de 8 bits, cada elemento contiene un valor entre 0 y 255. Para detectar la señal y eliminar el resto, el software realiza una transformación a una escala de grises. A continuación, se aplica una máscara a la imagen para eliminar o conservar cada uno de sus píxeles. Esta máscara se determina mediante una técnica de umbralización automática basada en el método de Otsu [34]. Este método determina un valor umbral que sirve para discriminar los píxeles que hay que eliminar de los que hay que conservar. En resumen, el algoritmo devuelve un único umbral de intensidad que separa los píxeles en dos clases, primer plano (*foreground*) y fondo (*background*). Este umbral se calcula maximizando la varianza entre clases en el histograma de intensidad (véase la Figura 4.8). Una vez establecido el umbral, cualquier píxel cuyo valor sea inferior o igual a él, se establece en 0, y cualquier valor superior a él se establece en 1.

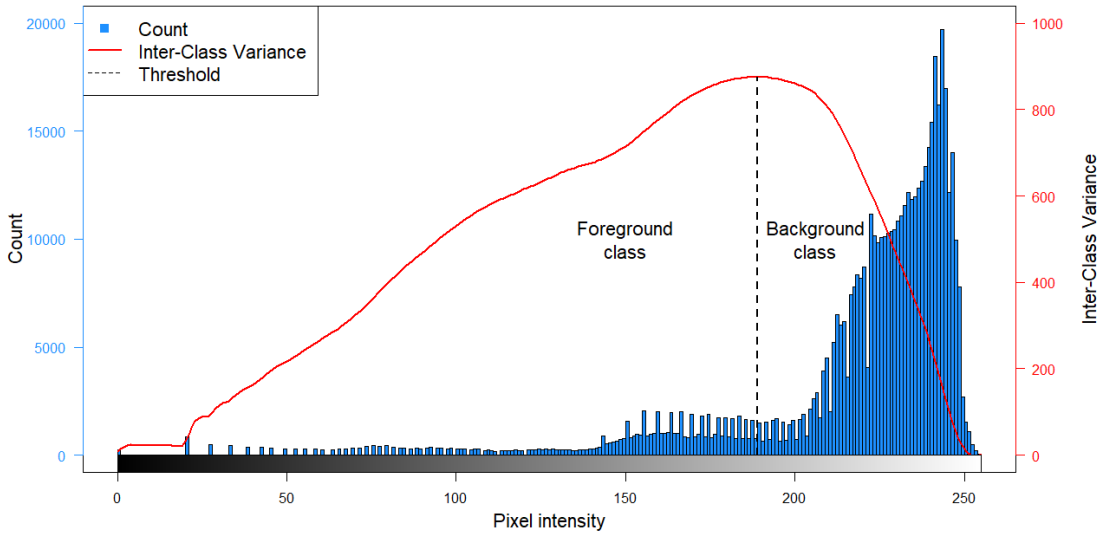


Figura 4.8: Gráfico que muestra los resultados de aplicar el método de Otsu a una imagen de ECG.

Extracción de señales

Detección de regiones de interés

Antes de extraer las señales, lo primero que hay que hacer es localizar las regiones de interés (ROI) en la imagen, que se tomarán como los "centros" de cada señal aproximadamente (véanse las líneas magenta en la Figura 4.9).

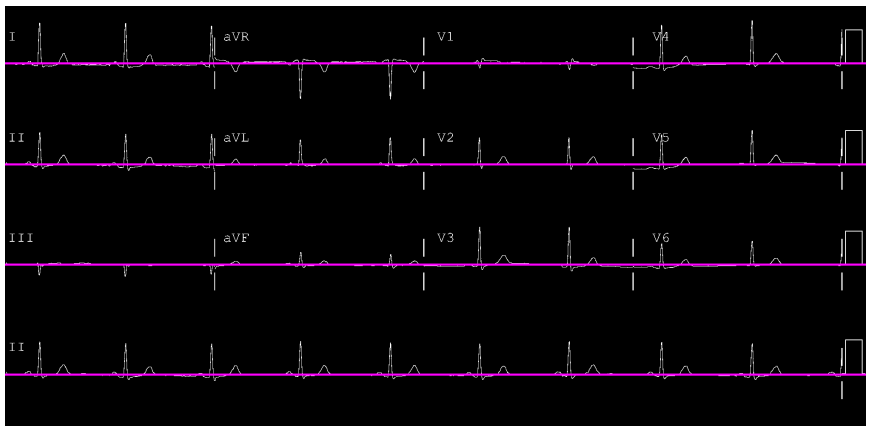


Figura 4.9: Ejemplo ilustrativo de un ECG con sus ROI resaltadas. Las líneas magenta identifican los centros.

Otros trabajos de la literatura utilizan estas ROI para recortar cada derivación, ya sea manualmente [21, 35, 36] o utilizando un mecanismo automático de recorte de derivaciones [37, 38]. El recorte manual provoca un enorme cuello de botella en el proceso de digitalización, ya que hay que dedicar mucho tiempo a realizar los recortes. El recorte automático tiene el riesgo de fallar cuando se digitalizan ECG con ciertas patologías, como el bloqueo de rama (ver Figura 4.10), lo que hace casi imposible separar correctamente las señales en rectángulos. Para evitar este problema, en lugar de recortar zonas, ECGMiner trabaja con la imagen completa durante todo el proceso.

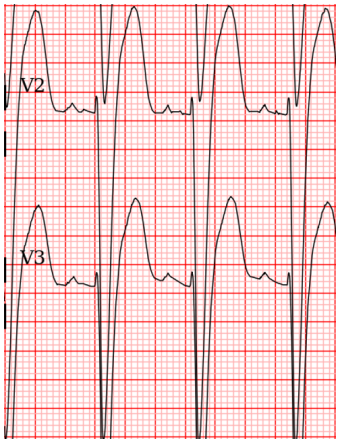


Figura 4.10: Recorte de las derivaciones V2 y V3 de un ECG patológico con bloqueo de rama.

El software detecta las ROI aplicando una ventana deslizante de 10 píxeles sobre toda la imagen a lo largo del eje vertical y calculando la desviación estándar de cada ventana de píxeles. Los centros de las ventanas con mayor desviación se identifican como las ROI y se asociarán a los centros de cada una de las señales a extraer (ver las cruces magenta Figura 4.11).

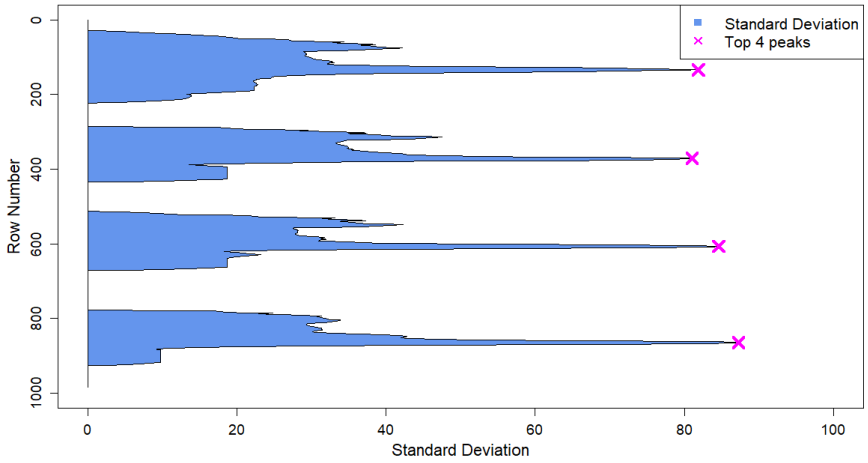


Figura 4.11: Ejemplo ilustrativo de la desviación estándar (azul) basada en filas obtenida para una imagen de ECG. Los cuatro puntos con mayor desviación se encuentran marcados con una cruz magenta.

Detección de señales

El procedimiento de detección de señales se realiza una vez que se han detectado las regiones de interés (ver Algoritmo 1). Consiste en almacenar las conexiones más probables entre las distintas regiones de píxeles negros consecutivos (denominadas clústers) en base a una función de coste personalizada, con el fin de identificar los píxeles pertenecientes a cada una de las señales. Para ello, se itera sobre cada columna de píxeles de la imagen y sobre cada uno de los clústers que contienen. Para cada clúster, se crea un enlace con el clúster de la columna inmediatamente anterior que minimice la función de coste. Esta función considera la región de interés que se está tomando como referencia, por lo que todo el proceso de enlace se realiza para cada una de las ROI marcadas. Para una ROI (r), un clúster (x_1) y otro clúster de la columna anterior (x_2), la función de coste viene definida por la Ecuación (4.1).

$$C(x_1, x_2, r) = S(x_2, r) + D(x_2, r) + \frac{M}{5} \times G(x_1, x_2) \tag{4.1}$$

Algoritmo 1 Extracción de señales

Input: *ecg* (matriz binaria) de dimensiones $M \times N$

$LEN \leftarrow 2$
$SCORE \leftarrow 3$
$rois \leftarrow get\_rois(ecg)$
$roi\_n \leftarrow len(rois)$
$cache \leftarrow HashMap()$
for $col \in \{ini\_col \ ... \ (N-1)\}$ **do**
    $prev\_clusters \leftarrow get\_clusters(ecg, col-1)$
    if $len(prev\_clus) = 0$ **then**
        continue
    end if
    $clusters \leftarrow get\_clusters(ecg, col)$
    for $c \in clusters$ **do**
        $cache[(col, c)] \leftarrow rep(\emptyset, roi\_n)$
        for $roi\_i \in \{0 \ ... \ (roi\_n-1)\}$ **do**
            $costs \leftarrow HashMap()$
            for $pc \in prev\_clusters$ **do**
                $node \leftarrow (col-1, pc)$
                $ctr \leftarrow ceil(mean(pc))$
                if $node \notin cache.keys()$ **then**
                    $val = (ctr, \emptyset, 1, 0)$
                    $cache[node] = rep(val, roi\_n)$
                end if
                $s \leftarrow cache[node][roi\_i][SCORE]$
                $d \leftarrow abs(ctr - rois[roi\_i])$
                $g \leftarrow gap(pc, c)$
                $costs[pc] \leftarrow s + d + M/5 \times g$
            end for
            $best \leftarrow costs.min()$
            $y \leftarrow ceil(mean(best))$
            $p \leftarrow (col-1, best)$
            $l \leftarrow cache[p][roi\_i][LEN] + 1$
            $s \leftarrow costs[best]$
            $cache[(col, c)][roi\_i] \leftarrow (y, p, l, s)$
        end for
    end for
end for
$raw\_signals \leftarrow backtracking(cache, rois)$
return raw_signals

El primer sumando de la función de coste es el *coste acumulativo* (S) de x_2 para la región de interés r. D es la *distancia Manhattan* desde el centro de x_2 a r, y G es el *gap* o espacio vertical en blanco entre x_1 y x_2. Este espacio será cero si están en contacto directo entre sí. Nótese que este término está ponderado por un quinto del número de filas (M). Esto significa que, aunque estén permitidas, las desconexiones a lo largo de la señal están severamente penalizadas.

Para almacenar eficientemente todas estas operaciones, se utiliza una estructura de tipo *hash-map* (con pares clave-valor) que actúa como caché. La clave es el clúster correspondiente y su valor asociado es una lista de arrays de longitud 4 (un array por cada ROI) que contiene: 1) la coordenada de la fila del punto medio del clúster; 2) el clúster de la columna anterior con el que se ha conectado; 3) la longitud total de la señal hasta ese momento; 4) el coste acumulado del clúster anterior.

Una vez evaluadas todas las columnas de píxeles, el algoritmo realiza un poroceso de *backtracking* sobre la caché. Esto se hace para obtener los caminos más largos con puntuación mínima para cada ROI, y almacenar las coordenadas de cada uno de los píxeles que las constituyen. Durante el recorrido hacia atrás, se aplicará sobre cada señal un algoritmo de detección de *peaks* para tratar de detectar los complejos QRS y perfilar sus ondas. Este refinamiento ayuda a digitalizar las ondas del ECG con un poco más de precisión, lo que tiene un considerable interés y valor clínico.

Posprocesado

Detección de pulsos de referencia

El primer paso del posprocesado consiste en detectar los pulsos de referencia de las señales. Para ello, se recorren las señales desde los extremos y se busca el patrón simétrico de mayor longitud. Hecho esto, se memorizan las coordenadas de 0mV y 1mV de cada pulso y, a continuación, se eliminan estas subsecuencias de las señales.

Normalización

Durante el segundo paso del posprocesado, las señales se dividen en partes iguales, cada una de las cuales se corresponde con una de las derivaciones de acuerdo con el formato de entrada que el usuario haya seleccionado. Antes de guardar las señales, se aplica un escalado previo a cada derivación l, como se muestra en la Ecuación (4.2), donde $y^{[l]}$ es el vector de las coordenadas fila de la derivación l, $v_0^{[l]}$ y $v_1^{[l]}$ son, respectivamente, las coordenadas de fila de los valores de referencia de 0mV y 1mV para esa derivación y $v^{[l]}$ es un vector con la misma longitud que $y^{[l]}$ con todos los elementos iguales a $v_0^{[l]}$.

$$s^{[l]} = \frac{1}{v_0^{[l]} - v_1^{[l]}} \times (v^{[l]} - y^{[l]}). \tag{4.2}$$

Adicionalmente, se aplica una interpolación lineal para fijar el mismo número de observaciones en todas las derivaciones. El usuario también puede fijar previamente el número de observaciones totales si así lo desea. Esta operación no afecta a la forma de las señales si se utiliza como técnica de *upsampling* para obtener un mayor número de observaciones que las representadas originalmente. Finalmente, ECGMiner almacena las señales de las derivaciones en una estructura de tipo matricial y las guarda en un archivo CSV, respetando el orden temporal y añadiendo valores nulos cuando no se dispone de observaciones para una derivación en un momento determinado.

La Figura 4.12 representa todas las etapas que se cumplen secuencialmente una vez identificado el rectángulo con la pista, hasta que se digitaliza la pista.

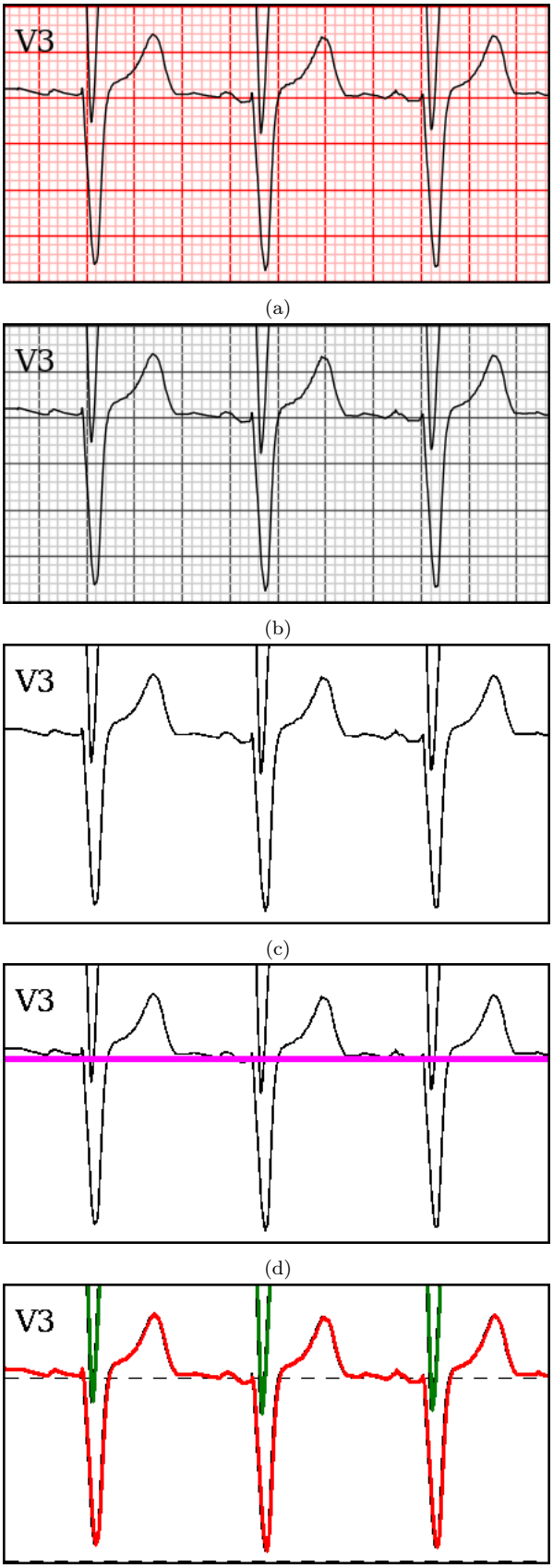


Figura 4.12: Secuencia de transformación de un recorte de una derivación a lo largo de la digitalización: (a) derivación de ECG recortada, (b) conversión a escala de grises, (c) binarización (blanco/negro), (d) ROI identificada y resaltada en magenta, (e) señal de la derivación digitalizada resaltada en rojo (parte de otra derivación detectada arriba resaltada en verde).

Extracción de metadatos (opcional)

En caso de que el usuario desee extraer la información demográfica y los metadatos del electrocardiograma en un archivo de texto, existe una etapa adicional que utiliza el motor de reconocimiento óptico Tesseract OCR [39]. Tesseract permite emitir una cadena única de caracteres con las palabras estructuradas en un párrafo, tal y como haría un humano al transcribirlo. En la Figura 4.13 se puede ver un ejemplo de un ECG sobre el que se han extraído los metadatos.

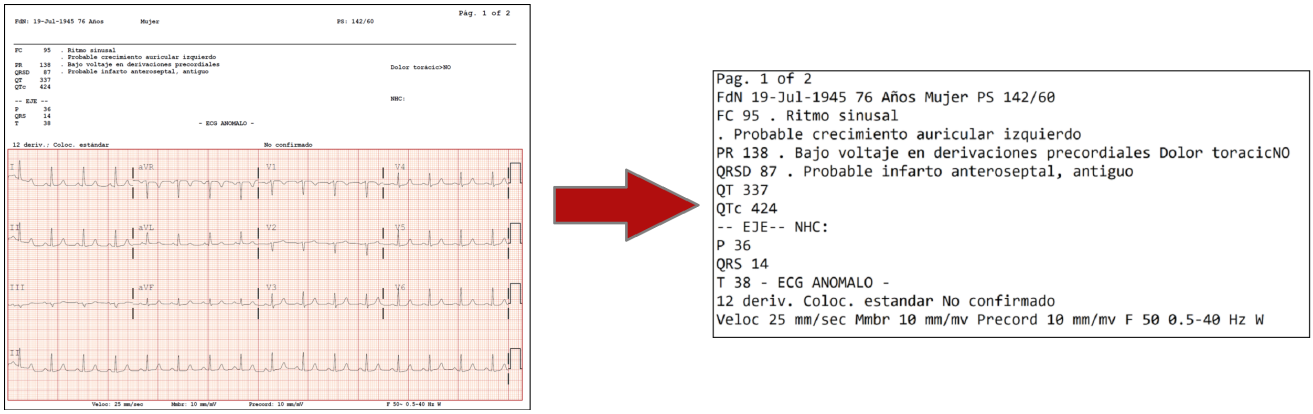


Figura 4.13: Ejemplo ilustrativo de la opción de extracción de metadatos sobre un ECG.

4.2.5. Diseño de la interfaz de usuario

Se prevé que ECGMiner sea utilizado por usuarios no necesariamente familiarizados con el manejo de la terminal del sistema para interactuar con el software. Por este motivo, se ha decidido dotar a la herramienta de una interfaz gráfica de usuario (GUI) que permita interactuar directamente con ella. En la Figura 4.14 se muestra un boceto de la GUI.

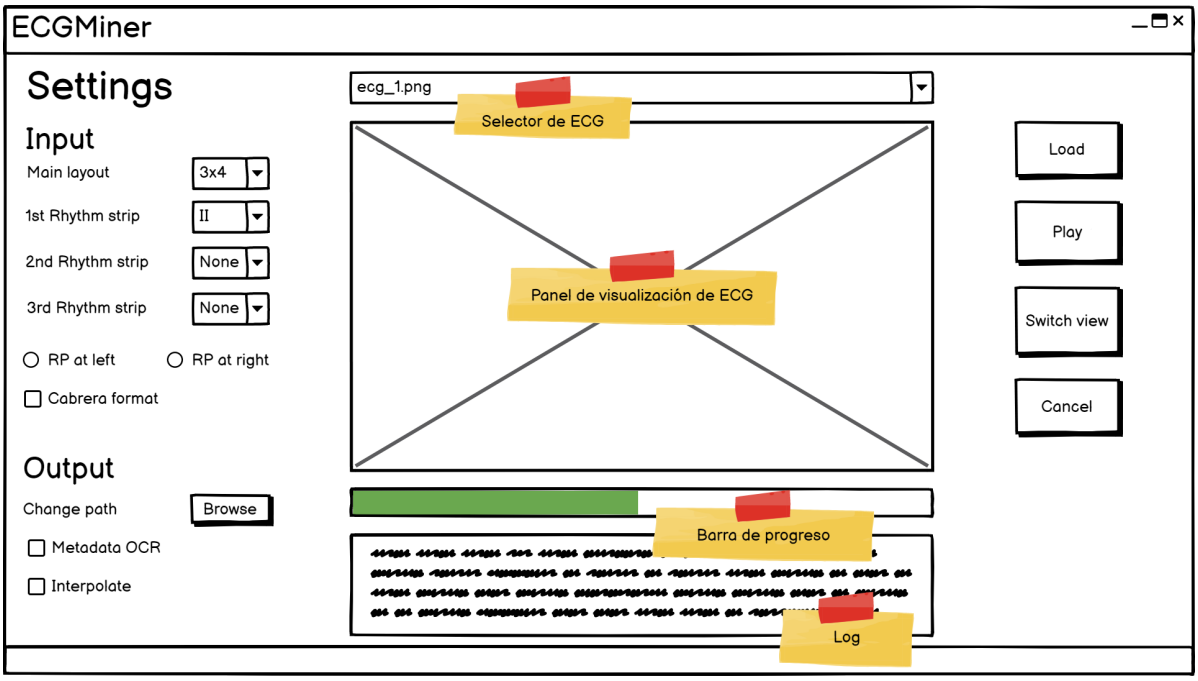


Figura 4.14: Boceto de la interfaz gráfica de usuario de ECGMiner.

En la parte izquierda de la GUI se encuentra la sección de `Settings`, en al cual se puede cambiar la configuración tanto del formato de entrada de los ECG, como de las distintas configuraciones de salida. Está compuesta por:

- `Input`: permite al usuario seleccionar el formato de entrada de los ECG a digitalizar. Esto incluye:
  - `Main layout`: la distribución principal de las señales. En particular, el software permite formatos 3x4, 6x2 o 12x1.
  - `1st Rhythm strip`: nombre de la derivación correspondiente a la primera tira de ritmo. Puede ser None.
  - `2nd Rhythm strip`: nombre de la derivación correspondiente a la segunda tira de ritmo. Puede ser None.
  - `3rd Rhythm strip`: nombre de la derivación correspondiente a la tercera tira de ritmo. Puede ser None.
  - `RP at Left/RP at right`: seleccionar uno u otro dependiendo de si los pulsos de referencia del ECG están situados a la izquierda o a la derecha de la cuadrícula.
  - `Cabrera format`: marcar si el formato de colocación de las derivaciones es el formato Cabrera en lugar del formato estándar.

- `Output`: permite al usuario indicar los detalles relativos a los resultados generados. Incluye:
  - `Change path`: establece la ruta donde se almacenarán los archivos de resultados, tanto de las señales del ECG, como de las imágenes con las trazas de la digitalización, como de los ficheros de texto con los metadatos.
  - `Metadata OCR`: si se selecciona, el software procesa los metadatos de los ECG y los almacena en un archivo de texto al final de la ejecución.
  - `Interpolate`: si se selecciona, el software interpolará los valores digitalizados a un número de observaciones concreto. Aparecerá un *combo box* para que el usuario escoja el número de observaciones que desee.

En el centro de la interfaz GUI hay cuatro elementos útiles para obtener información del proceso de digitalización. De arriba a abajo:

- `Selector y panel de visualización de ECG`: permite navegar entre los distintos ECG cargados para visualizarlos.
- `Barra de progreso`: mientras se digitalizan los ECG, esta barra muestra el porcentaje de digitalizaciones ya completadas.
- `Log`: imprime la hora y el nombre de los ECG que ya han sido digitalizados. En caso de que el software falle en la digitalización de un ECG en particular, informará del error por consola y continuará con la digitalización del resto de ECG pendientes.

En la parte derecha de la GUI hay cuatro botones para interactuar con la ejecución del software:

- `Load`: al hacer clic aquí, el usuario podrá cargar de su dispositivo los ECG que desea digitalizar. Se pueden seleccionar uno o varios ECG.
- `Play`: sirve para iniciar el proceso de digitalización.
- `Switch view`: cambia el modo de visualización del ECG actualmente seleccionado en el cuadro de visualización. Para un ECG que ya ha sido digitalizado, el software puede mostrar cada derivación detectada por el algoritmo de extracción trazando líneas de color en el cuadro de visualización, superpuestas a las derivaciones de la imagen original. Además, cada pulso de referencia detectado se representará mediante dos líneas discontinuas que corresponden a los valores de 0mV y 1mV.
- `Cancel`: permite cancelar el proceso de digitalización.

Capítulo 5

Implementación

En este capítulo se detallan las desviaciones respecto al diseño inicial, la organización del código y la disponibilidad y licencia de la herramienta.

5.1. Desviaciones respecto al diseño inicial

En general, la implementación fue bastante fiel al diseño realizado originalmente. Sin embargo, se observó que la GUI se congelaba durante el proceso, no pudiendo así visualizar los resultados de los ECG ya digitalizados hasta que terminase. Por este motivo, se añadió una paralelización al sistema. Tal y como se ilustra en la Figura 5.1, el conjunto de imágenes de ECG se divide en distintos lotes. A continuación, se lanzan varios subprocesos de CPU, cada uno de los cuales se encarga de digitalizar uno de los lotes de imágenes. Esto además reduce el tiempo de cómputo aunque, debido a las limitaciones de la librería y del propio sistema de multienhebrado de Python, no se pudo conseguir una implementación del todo eficiente. Por ello, se plantea como trabajo futuro optimizarlo más en profundidad.

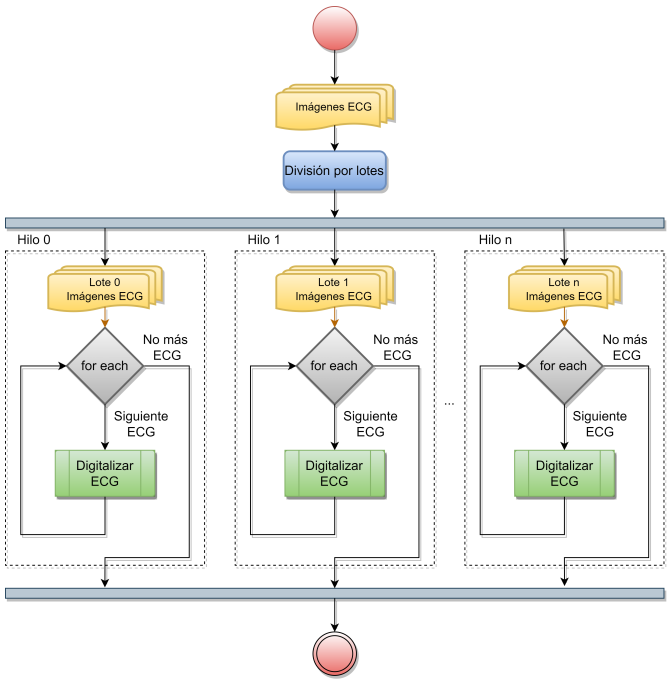


Figura 5.1: Diagrama de flujo del tratamiento por lotes en paralelo.

Para ello, se creó la clase `Thread`, la cual hereda de la clase `QRunnable` del paquete `PyQt5` y se encarga de digitalizar un lote de ECG.

```python
# Standard library imports
from typing import Callable, Iterable
from os.path import realpath, sep

# Third-party imports
from PyQt5.QtCore import QRunnable, pyqtSlot

# Application-specific imports
from app.controller.SignalContainer import SignalContainer
from utils.error.DigitizationError import DigitizationError
from digitization.Digitizer import Digitizer


class Thread(QRunnable):
    """
    Thread in charge of digitize a batch of ECG.
    """

    def __init__(
        self, digitizer: Digitizer, paths: Iterable[str], is_digitizing: Callable
    ) -> None:
        """
        Initialization of the thread.

        Args:
            digitizer (Digitizer): Object in charge of the digitization of a single ECG.
            paths (Iterable[str]): Input paths of the ECG image files.
            is_digitizing (Callable): Function to check if the system is digitizing or not.
        """
        super().__init__()
        self.__digitizer = digitizer
        self.__paths = paths
        self.__is_digitizing = is_digitizing
        self.__signals = SignalContainer()

    @pyqtSlot()
    def run(self):
        """
        Digitize the batch of ECG with the settings specified in the model.
        It will stop if the model state says digitization has stopped.
        """
        for i, path in self.__paths:
            filename = realpath(path).split(sep)[-1]
            if not self.__is_digitizing():
                break
            try:
                self.__digitizer.digitize(path)
            except DigitizationError as e:
                self.__signals.error.emit(i, str(e) + f" ({filename})")
            else:
                msg = filename + " digitized"
                self.__signals.progress.emit(i, msg)
```

```
54          self.__signals.finished.emit()
55
56      def progress_connect(self, func: Callable) -> None:
57          """
58          Connect the progress signal with a certain function.
59
60          Args:
61              func (Callable): Function to be connected.
62          """
63          self.__signals.progress.connect(func)
64
65      def finished_connect(self, func: Callable) -> None:
66          """
67          Connect the finished signal with a certain function.
68
69          Args:
70              func (Callable): Function to be connected.
71          """
72          self.__signals.finished.connect(func)
73
74      def error_connect(self, func: Callable) -> None:
75          """
76          Connect the error signal with a certain function.
77
78          Args:
79              func (Callable): Function to be connected.
80          """
81          self.__signals.error.connect(func)
```

Además, se añadió la clase `SignalContainer` para añadir las distintas señales con las que el controlador pudiera comunicarse con los hilos, tanto para indicar el progreso, como para finalizar, como para informar de un error.

```
1 # Third-party imports
2 from PyQt5.QtCore import QObject, pyqtSignal
3
4 class SignalContainer(QObject):
5     """
6     Container of the possible signals of a Miner Thread (progress, finished and
      error).
7     """
8
9     progress = pyqtSignal(int, str)
10    finished = pyqtSignal()
11    error = pyqtSignal(int, str)
```

En la clase `Controller` se añadieron por tanto los distintos *callbacks* que se ejecutan en función de las señales recibidas. Cabe destacar que se añadió un atributo al controlador para llevar la cuenta del número de hilos activos. También, se añadió otro atributo con el tiempo total en el que se inició la digitalización, para así informar al usuario de lo que ha tardado la digitalización al finalizar.

```
1 # Standard library imports
2 import time
3 from os.path import basename, dirname, realpath, sep, splitext
4 from typing import Iterable, Optional, Tuple
```

```
5
6  # Third-party imports
7  import numpy as np
8  from PyQt5.QtCore import QThreadPool
9
10 # Application-specific imports
11 from app.controller.Thread import Thread
12 from app.model.Model import Model
13 from app.view.View import View
14 from utils.ecg.Lead import Lead
15 from utils.ecg.Format import Format
16 from utils.graphics.Image import Image
17 from digitization.Digitizer import Digitizer
18
19
20 class Controller:
21     """
22     Main controller of ECG Miner app. It contains a all the logic to process the
23     inputs from the view and communicate with the model.
24     """
25
26     def __init__(self, model: Model, view: View) -> None:
27         """
28         Initialization of the controller.
29
30         Args:
31             model (Model): Model of the app.
32             view (View): View of the app.
33         """
34         self.__model = model
35         self.__view = view
36         self.__active_threads = 0
37         self.__ini_time = None
38
39     ...
40
41     def proc_digitize_evt(self) -> None:
42         """
43         Process the digitize event with the selected configuration.
44         """
45         total_paths = self.__model.ecg_paths
46         self.__model.digitizing = True
47         self.__view.enable_progress(True)
48         # Disable settings, browse and cancel
49         self.__view.enable_digitize(False)
50         self.__view.enable_settings(False)
51         self.__view.enable_browse(False)
52         # ThreadPool
53         pool = QThreadPool.globalInstance()
54         self.__active_threads = max(1, pool.maxThreadCount() - 1)
55         # Split pending ECG in a partition
56         pending_paths = np.array(
57             [
58                 (i, path)
59                 for i, path in enumerate(total_paths)
60                 if not self.__model.is_digitized(i)
61             ],
62             dtype=object,
```

```
        )
        split = np.array_split(pending_paths, self.__active_threads)
        self.__view.log(f"STARTING DIGITIZATION of {len(total_paths)} files")
        self.__ini_time = time.time()
        for i in range(self.__active_threads):
            miner = Digitizer(
                layout=self.__model.layout,
                rhythm=self.__model.rhythm,
                rp_at_right = self.__model.rp_at_right,
                cabrera=self.__model.cabrera,
                outpath=self.__model.outpath,
                ocr=self.__model.ocr,
                interpolation=self.__model.interpolation,
            )
            is_digitizing = lambda: self.__model.digitizing
            worker = Thread(miner, split[i], is_digitizing)
            worker.finished_connect(self.finished_callback)
            worker.progress_connect(self.progress_callback)
            worker.error_connect(self.error_callback)
            pool.start(worker)

    def finished_callback(self) -> None:
        """
        Callback for a thread when it has finished.
        """
        self.__active_threads -= 1
        if self.__model.progress == 100:
            self.__view.log(
                f"FINISHED DIGITIZATION of {len(self.__model.ecg_paths)} files "
                + f"({round(time.time() - self.__ini_time,2)} s)"
            )
        if self.__active_threads == 0:
            self.__view.enable_browse(True)
            self.__view.enable_cancel(False)

    def progress_callback(self, idx: int, msg: str) -> None:
        """
        Callback for a thread to report its progress.
        """
        self.__model.set_digitized(idx)
        self.__view.log(msg)
        self.__view.set_progress(self.__model.progress)

    def error_callback(self, idx: int, msg: str) -> None:
        """
        Callback for a thread when an error has ocurred.
        """
        self.__model.set_digitized(idx)
        self.__view.log(msg, error=True)
        self.__view.set_progress(self.__model.progress)
```

Finalmente, cabe destacar que para la implementación de las clases `Rectangle`, `Point` y `Format` se utilizó el decorador `dataclass`, el cual es una función que agrega internamente distintos métodos automáticamente tales como el constructor, los *getters* y *setters*, etc. Además, implementa automáticamente la función *hash*, la cual es necesaria para utilizar un objeto como clave de un diccionario de Python.

5.2. Organización del código fuente

En esta sección se explica la organización del código y la estructura de directorios del mismo. En la Figura 5.2 se muestra el contenido del directorio raíz. Además de las carpetas principales, contiene los siguientes ficheros:

- El fichero `.gitignore` contiene el archivo con los nombres de los ficheros y carpetas auxiliares que no se han subido al repositorio.
- El fichero `ECGMiner.spec` contiene el fichero de configuración para instalar la herramienta.
- El fichero `LICENSE` contiene la licencia bajo la que está sujeto el software (ver Sección 5.3).
- El fichero `README.md` contiene información descriptiva acerca de la herramienta, incluyendo la propia guía de instalación (ver Apéndice B).
- El fichero `requirements.txt` contiene las distintas dependencias de Python que se necesitan instalar (ver Apéndice A).

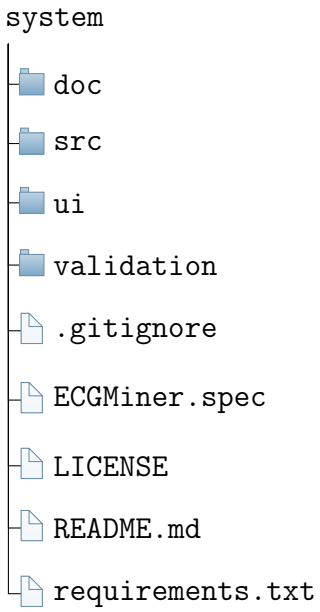


Figura 5.2: Estructura general del repositorio.

En la Figura 5.3 se muestra el contenido del directorio `src`, el cual contiene todos los paquetes de Python con sus respectivas clases, correspondiéndose con la arquitectura explicada en el Capítulo 4. Adicionalmente, se incluye el fichero `__main__.py` que se encarga de poner en funcionamiento la aplicación.

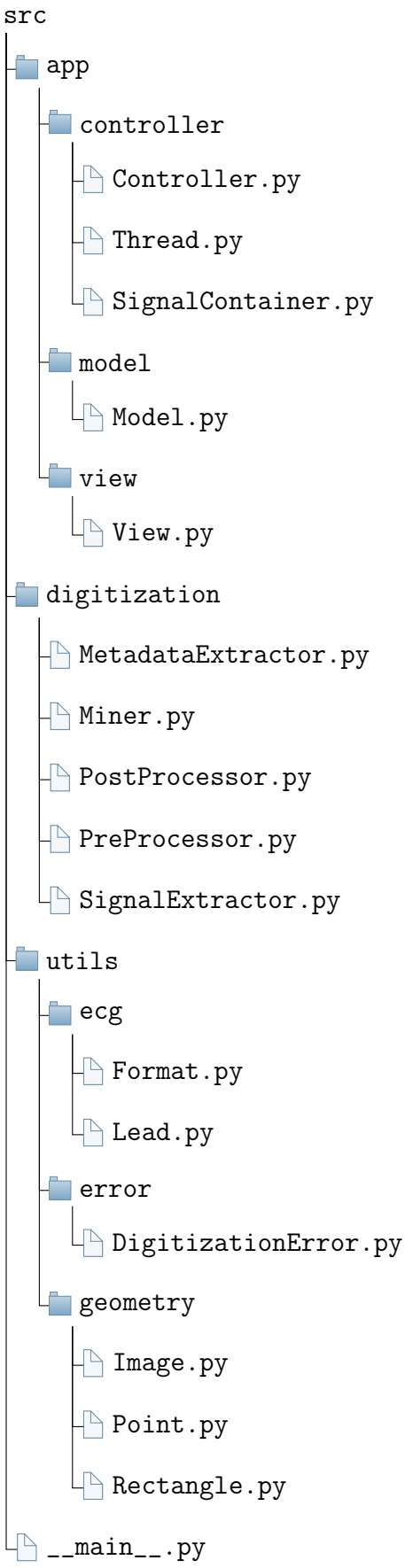


Figura 5.3: Estructura del directorio `src`.

En la Figura 5.4 se muestra el contenido del directorio `ui`. Contiene las distintas imágenes e iconos utilizados en la interfaz gráfica. Además, el fichero `gui.ui` contiene el fichero de configuración con el diseño de la vista realizado con Qt Designer, el cual es un *framework* usado para diseñar interfaces gráficas de usuario basadas en Qt.

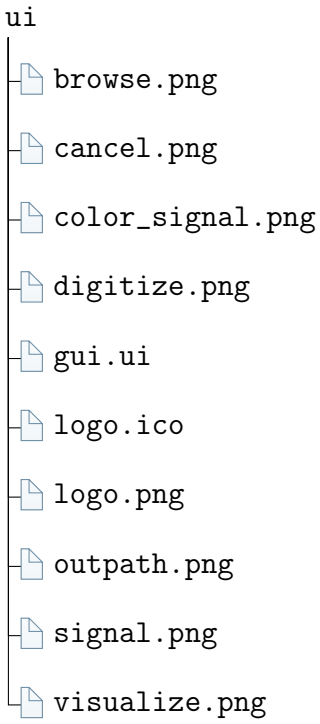


Figura 5.4: Estructura del directorio `ui`.

Finalmente, en la Figura 5.4 se muestra el contenido del directorio `validation`, en el cual se incluyen todos los datos y scripts utilizados en el proceso de validación del software (ver Sección 6.3). En la carpeta `LUDB` se encuentran las imágenes y señales de los ECG de la base de datos con su mismo nombre, tanto en su versión original como en su versión digitalizada. El fichero `LUDB_validation.ipynb` contiene el cuaderno de pruebas sobre dicha base de datos. Análogamente, la carpeta `PTB-XL` tiene una estructura muy similar a la de `LUDB`, con la diferencia de que no se incluyen las imágenes de los ECG debido a que ocupan demasiado espacio en el repositorio, lo cual se explica en el fichero `README.md`. Además, se adjunta el fichero `ptbxl_database` con todos los metadatos de PTB-XL. Finalmente, se incluye un fichero `render.py` con el que se han renderizado todas las imágenes a partir de las señales.

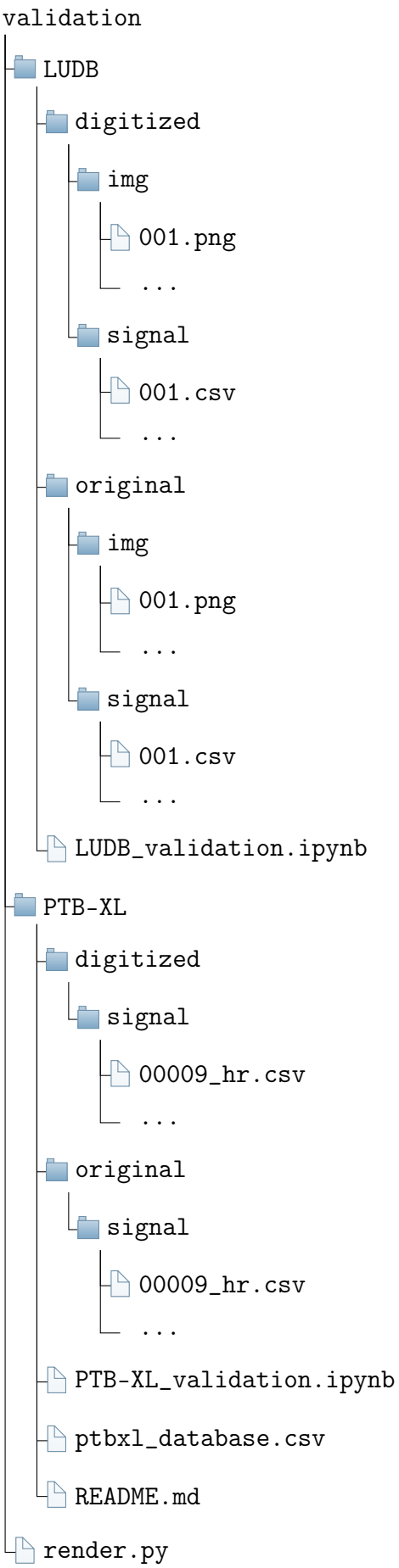


Figura 5.5: Estructura del directorio `validation`.

5.3. Disponibilidad del código y licencia

El software ECGMiner es totalmente de código abierto. Su código fuente, así como los scripts utilizados para la validación, los experimentos y la guía de instalación, se proporcionan en el siguiente repositorio de GitHub: `https://github.com/adofersan/ecg-miner`. Este repositorio se ha dotado de la licencia de software MIT, la cual permite a cualquier persona obtener una copia del software y utilizarlo, modificarlo o distribuirlo, tanto para fines comerciales como no comerciales, siempre y cuando se incluya una copia de la licencia y el aviso de derechos de autor en todas las copias del software. También establece que el titular de los derechos de autor del software no será responsable de ningún daño directo, indirecto, incidental, especial, ejemplar o consecuente que surja del uso del software.

Capítulo 6

Pruebas

En este capítulo se presentan los distintos casos de prueba planteados para comprobar el funcionamiento de la aplicación, así como los resultados de las mismas. Además, se explica cómo se ha evaluado la precisión de la herramienta. Finalmente, se ilustra el potencial de la herramienta al ser utilizada junto con el modelo 3DFMMecg.

En lo referente a la evaluación de la precisión ofrecida por ECGMiner, cabe destacar que no solamente se han utilizando bases de datos con imágenes ECG públicas para las cuales se conocen los valores numéricos asociados (lo cual permite evaluar cuantitativamente la precisión de la digitalización), sino que también se ha contado con la ayuda de un cardiólogo especializado en el análisis de ECG, el cual ha constatado que la herramienta digitaliza adecuadamente las ondas P, R y T del ECG, las cuales son claves para un posterior diagnóstico.

6.1. Casos de prueba

En esta sección se detallan los distintos casos de prueba que se han usado para comprobar el correcto funcionamiento de la herramienta. Por un lado, se ha probado que la herramienta funcione con distintas configuraciones iniciales y formatos. Por otro lado, se ha probado que la herramienta responda correctamente frente a los distintos fallos que pueda encontrar durante la digitalización.

ID	CP01
Título	Digitalizar ECG con formato estándar.
Descripción	Digitalizar un ECG con formato estándar. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Fichero CSV con las 12 señales de las derivaciones del ECG e imagen PNG con la traza de la digitalización en el mismo directorio que la imagen original.

Tabla 6.1: Caso de prueba CP01. Digitalizar con ECG formato estándar.

ID	CP02
Título	Digitalizar ECG con formato Cabrera.
Descripción	Digitalizar un ECG en formato Cabrera 3x4. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se clica en el *radio button* de formato Cabrera. 4. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Fichero CSV con las 12 señales de las derivaciones del ECG e imagen PNG con la traza de la digitalización en el mismo directorio que la imagen original.

Tabla 6.2: Caso de prueba CP02. Digitalizar con ECG formato Cabrera.

ID	CP03
Título	Digitalizar ECG con dos tiras de ritmo.
Descripción	Digitalizar un ECG en formato estándar 3x4 con dos tiras de ritmo (la II y la V1).
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se selecciona en el *combo box* de la primera tira de ritmo la derivación II. 4. Se selecciona en el *combo box* de la segunda tira de ritmo la derivación V1. 5. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Fichero CSV con las 12 señales de las derivaciones del ECG e imagen PNG con la traza de la digitalización en el mismo directorio que la imagen original.

Tabla 6.3: Caso de prueba CP03. Digitalizar ECG con dos tiras de ritmo.

ID	CP04
Título	Digitalizar ECG con tres tiras de ritmo.
Descripción	Digitalizar un ECG en formato estándar 3x4 con dos tiras de ritmo (la II, la V1 y la V5).
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se selecciona en el *combo box* de la primera tira de ritmo la derivación II. 4. Se selecciona en el *combo box* de la segunda tira de ritmo la derivación V1. 5. Se selecciona en el *combo box* de la tercera tira de ritmo la derivación V5. 6. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Fichero CSV con las 12 señales de las derivaciones del ECG e imagen PNG con la traza de la digitalización en el mismo directorio que la imagen original.

Tabla 6.4: Caso de prueba CP04. Digitalizar ECG con tres tiras de ritmo.

ID	CP05
Título	Digitalizar ECG 6x2.
Descripción	Digitalizar un ECG en formato estándar 6x2. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se selecciona en el *combo box* de *main layout* la disposición 6x2. 4. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Fichero CSV con las 12 señales de las derivaciones del ECG e imagen PNG con la traza de la digitalización en el mismo directorio que la imagen original.

Tabla 6.5: Caso de prueba CP05. Digitalizar ECG 6x2.

ID	CP06
Título	Digitalizar ECG 12x1.
Descripción	Digitalizar un ECG en formato estándar 12x1. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se selecciona en el *combo box* de *main layout* la disposición 12x1. 4. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Fichero CSV con las 12 señales de las derivaciones del ECG e imagen PNG con la traza de la digitalización en el mismo directorio que la imagen original.

Tabla 6.6: Caso de prueba CP06. Digitalizar ECG 12x1.

ID	CP07
Título	Digitalizar lote de ECG.
Descripción	Digitalizar un lote de 100 ECG en formato estándar 3x4. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se seleccionan los ECG del lote a digitalizar. 3. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Ficheros CSV con las 12 señales de las derivaciones del ECG e imágenes PNG con las trazas de las digitalizaciones en el mismo directorio que las imágenes originales.

Tabla 6.7: Caso de prueba CP07. Digitalizar lote de ECG.

ID	CP08
Título	Extraer metadatos con Tesseract OCR.
Descripción	Digitalizar un ECG en formato estándar 3x4 y extraer sus metadatos. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se marca el *check box* de *Metadata OCR*. 4. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Fichero CSV con las 12 señales de las derivaciones del ECG, imagen PNG con la traza de la digitalización en el mismo directorio que la imagen original y fichero TXT con los metadatos.

Tabla 6.8: Caso de prueba CP08. Extraer metadatos con Tesseract OCR.

ID	CP9
Título	Interpolar señales.
Descripción	Digitalizar un ECG en formato estándar 3x4 e interpolar sus señales a 2000 observaciones. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se marca el *check box* de interpolar. 4. Se selecciona en el *spin box* 2000 observaciones. 5. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Fichero CSV con las 12 señales de las derivaciones del ECG (2000 observaciones) e imagen PNG con la traza de la digitalización en el mismo directorio que la imagen original.

Tabla 6.9: Caso de prueba CP9. Interpolar señales.

ID	CP10
Título	Cambiar ruta de salida.
Descripción	Digitalizar un ECG en formato estándar 3x4 y almacenarlo en un directorio distinto. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se clica en el botón de `Browse` para cambiar la ruta del directorio de salida. 4. Se selecciona el directorio "D:\Users". 5. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Fichero CSV con las 12 señales de las derivaciones del ECG e imagen PNG con la traza de la digitalización en el directorio "D:\Users".

Tabla 6.10: Caso de prueba CP10. Cambiar ruta de salida.

ID	CP11
Título	Ver traza de un ECG digitalizado.
Descripción	Digitalizar un ECG en formato estándar 3x4 y visualizar la traza de la digitalización desde la propia herramienta. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se clica en el botón de `Play` y se espera a que termine la digitalización. 4. Se clica en el botón de `switch view` para cambiar la vista del ECG por el de su traza de digitalización.
Resultado esperado	La herramienta muestra la traza de digitalización del ECG.

Tabla 6.11: Caso de prueba CP11. Ver traza de digitalización.

ID	CP12
Título	Cancelar digitalización.
Descripción	Digitalizar un lote de 100 ECG en formato estándar 3x4 y cancelar la digitalización durante el proceso. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se seleccionan los ECG del lote a digitalizar. 3. Se clica en el botón de `Play`. 4. Se clica en el botón de cancelar cuando la barra de progreso haya alcanzado el 50 %.
Resultado esperado	El sistema informa al usuario de que se esperará a que terminen los procesos actuales y se cancelará el resto de digitalizaciones. Se obtienen los ficheros CSV con las 12 señales de las derivaciones del ECG e imágenes PNG con las trazas de las digitalizaciones en el mismo directorio que las imágenes originales. No se obtendrán los ficheros de los ECG no digitalizados debido a la cancelación.

Tabla 6.12: Caso de prueba CP12. Cancelar digitalización.

ID	CP13
Título	Digitalizar un no ECG.
Descripción	Intentar digitalizar una imagen que no se corresponda con un ECG.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el no ECG. 3. Se clica en el botón de `Play`.
Resultado esperado	El sistema informa al usuario de que no se ha podido llevar a cabo la digitalización, o bien debido a que no se han detectado correctamente las regiones de interés, o bien por que no se han detectado correctamente los pulsos de referencia.

Tabla 6.13: Caso de prueba CP13. Digitalizar un no ECG.

ID	CP14
Título	Digitalizar ECG sin pulsos de referencia.
Descripción	Intentar digitalizar un ECG editado del que se han eliminado los pulsos de referencia.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG sin los pulsos de referencia. 3. Se clica en el botón de `Play`.
Resultado esperado	El sistema informa al usuario de que no se ha podido llevar a cabo la digitalización debido a que no se han detectado correctamente los pulsos de referencia.

Tabla 6.14: Caso de prueba CP14. Digitalizar ECG sin pulsos de referencia.

ID	CP15
Título	Digitalizar ECG con configuración errónea.
Descripción	Intentar digitalizar un ECG cuya estructura no es consistente con la configuración introducida por el usuario.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar con formato estándar 6x2. 3. Se clica en el botón de `Play`.
Resultado esperado	El sistema informa al usuario de que no se ha podido llevar a cabo la digitalización debido a que no se han detectado correctamente las regiones de interés.

Tabla 6.15: Caso de prueba CP15. Digitalizar ECG con configuración errónea.

ID	CP16
Título	Ver traza de un ECG no digitalizado.
Descripción	Intentar visualizar la traza de un ECG en formato estándar 3x4 sin haberlo digitalizado previamente. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se clica en el botón de `switch view` para cambiar la vista del ECG por el de su traza de digitalización.
Resultado esperado	El sistema informa al usuario de que no se ha encontrado la imagen con la traza de la digitalización del ECG.

Tabla 6.16: Caso de prueba CP16. Ver traza de un ECG no digitalizado.

ID	CP17
Título	Extraer metadatos sin Tesseract OCR.
Descripción	Intentar digitalizar un ECG en formato estándar 3x4 y extraer sus metadatos sin tener Tesseract OCR instalado en el sistema correctamente. El resto de la configuración se mantiene por defecto.
Entrada	1. Se clica en el botón de `Load`. 2. Se selecciona el ECG a digitalizar. 3. Se marca el *check box* de *Metadata OCR*. 4. Se clica en el botón de `Play` y se espera a que termine la digitalización.
Resultado esperado	Fichero CSV con las 12 señales de las derivaciones del ECG, imagen PNG con la traza de la digitalización en el mismo directorio que la imagen original. El sistema informa al usuario de que no se ha encontrado la instalación de Tesseact OCR-Engine.

Tabla 6.17: Caso de prueba CP17. Extraer metadatos sin Tesseract OCR.

6.2. Resultados de las pruebas

A continuación, se muestra una tabla con los resultados de los casos de prueba. Un "OK" indica que se ha obtenido el resultado esperado, en caso contrario, se detalla el resultado obtenido y la solución aplicada.

Caso de prueba	**Resultado obtenido**	**Solución**
CP01	OK	-
CP02	OK	-
CP03	OK	-
CP04	OK	-
CP05	OK	-
CP06	OK	-
CP07	OK	-
CP08	Se lanzaba una excepción por no encontrar la ruta de instalación de Tesseract.	Se fijó la ruta de instalación por defecto a la misma que se indica en el manual de instalación (ver Apéndice B).
CP09	Aparecía la misma señal interpolada para todas las derivaciones.	Se movió el código relativo a la interpolación dentro del bucle en el que se itera sobre las distintas derivaciones.
CP10	OK	.
CP11	OK	-
CP12	OK	-
CP13	OK	-
CP14	Se lanzaba una excepción por división por cero, debido a que las referencias de voltaje de 0mV y 1mV se detectaban en el mismo píxel.	Se añadió una comprobación para confirmar que los voltajes de referencia no coincidieran en el mismo píxel.
CP15	OK	-
CP16	Se lanzaba una excepción por no encontrar la imagen con la traza de la digitalización.	Se añadió una comprobación para ver si existía la imagen con la traza de la digitalización, en caso contrario, se informa al usuario.
CP17	OK	-

Tabla 6.18: Resultados de los casos de prueba junto con sus correspondientes soluciones.

6.3. Validación

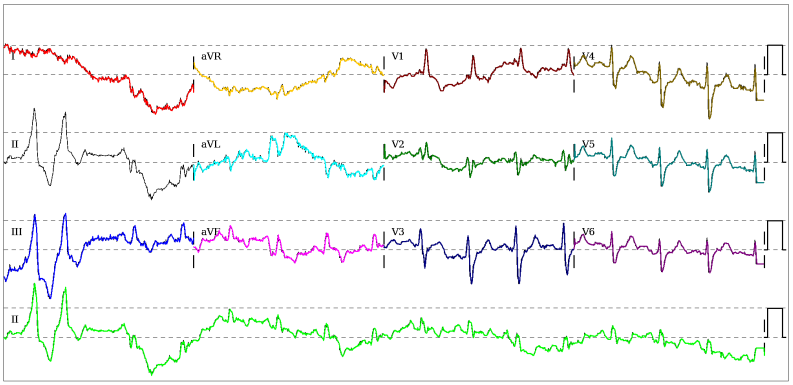
El objetivo de esta sección es mostrar el proceso que se ha llevado a cabo para evaluar la precisión de ECGMiner. Para ello, se usaron LUDB [40, 41] y PTB-XL [42, 43], dos bases de datos de ECG humanos disponibles públicamente en Physionet [44]. LUDB es una base de datos reciente que contiene 200 ECG con múltiples etiquetas de diagnóstico, mientras que PTB-XL es una base de datos mucho más extensa y ampliamente utilizada en la literatura científica. Hasta donde se ha investigado, ésta es la mayor base de datos utilizada en la validación de la digitalización de ECG. Se ha considerado un subconjunto de 2.203 del total de 21.837 ECG en PTB-XL, estratificados en el *fold 10* que se recomienda para la validación, ya que contiene ECG revisados y etiquetados por

expertos [42]. Las características demográficas y clínicas de los sujetos incluidos en el estudio son similares en ambas bases (ver Tabla 6.19), con aproximadamente un 40 % de participantes sanos con ECG normales. Cabe señalar que la proporción de mujeres y la edad media de PTB-XL es algo superior a la de LUDB.

Atributo	**LUDB**	**PTB-XL**
N	200	2.203
Edad	51,97 ± 19,25	60,92 ± 17,45
Mujer (%)	42,5	48,39
ECG normal (%)	37,5	41,44

Tabla 6.19: Caracterización de las bases de datos LUDB y PTB-XL por tamaño de la muestra (N), edad (media ± desviación estándar), sexo y porcentaje de diagnóstico ECG normal.

(a) LUDB



(b) PTB-XL

Figura 6.1: Ejemplos ilustrativos de digitalización de ECGMiner de 12 derivaciones en LUDB y PTB-XL. La señal de ECG original se muestra en negro y las líneas de diferentes colores corresponden a las derivaciones digitalizadas a través de las señales.(a): salida para el paciente con ID 185 en LUDB. (b: salida para el paciente con ID 1157 en PTB-XL.

En lo que respecta a la forma de las señales, existen diferencias notables entre estas bases de datos; la Figura 6.1 ilustra estas diferencias. La adquisición de datos del LUDB procede de un único dispositivo, mientras que los dispositivos de origen del PTB-XL son diversos. Además, PTB-XL es mucho más amplia y cuenta con una mayor variedad de patologías cardíacas. Por lo tanto, la heterogeneidad y complejidad de las señales de ECG se reflejan mejor en PTB-XL que en LUDB.

Los ECG de ambas bases de datos, cuyas señales tienen una duración total de 10s, se renderizaron con una resolución de 200dpi (*dots per inch*) en modo de visualización 3x4, formato estándar y la derivación II como tira de ritmo, siguiendo las normas de la American Heart Association [45].

Se realizaron tres pruebas de validación:

- Test 1: similitud de las 12 derivaciones (LUDB) (Sección 6.3.4). Se realizó para tener comparación objetiva entre las señales originales y digitalizadas de la base de datos LUDB.
- Test 2: similitud de las 12 derivaciones (PTB-XL) (Sección 6.3.5). Se realizó para evaluar la precisión de la herramienta sobre un conjunto de datos mucho más amplio y heterogéneo como es PTB-XL, y comprobar así si los resultados diferían o no respecto a los de LUDB.
- Test 3: similitud de las ondas P, R y T (LUDB) (Sección 6.3.6). Para este test se contó con el apoyo del médico especialista en cardiología Alberto Pérez Castellanos, perteneciente al Servicio de Cardiología del Hospital Universitario Son Espases (Palma, Islas Baleares). La colaboración del cardiólogo permitió estudiar la capacidad de digitalización sobre las ondas del ECG. No se llevó a cabo sobre PTB-XL porque resultaba desorbitado el esfuerzo de etiquetar manualmente casi 2000 ECG. Además, se esperaba que los test 1 y 2 mostraran ya resultados similares en ambas bases de datos.

6.3.1. Selección de ECG

Se realizó un proceso de selección de ECG para cada una de las bases de datos, eliminando aquellos ECG que presentaran artefactos que interfirieran significativamente en el proceso de validación. En primer lugar, se eliminaron los ECG de sujetos que llevaran marcapasos, debido a que producen una serie de segmentos verticales en cada una de las derivaciones, mezclándose así con las ondas (ver Figura 6.2). Estos sujetos conllevaron eliminar el 4,5 % y el 1,32 % de LUDB y PTB-XL, respectivamente.

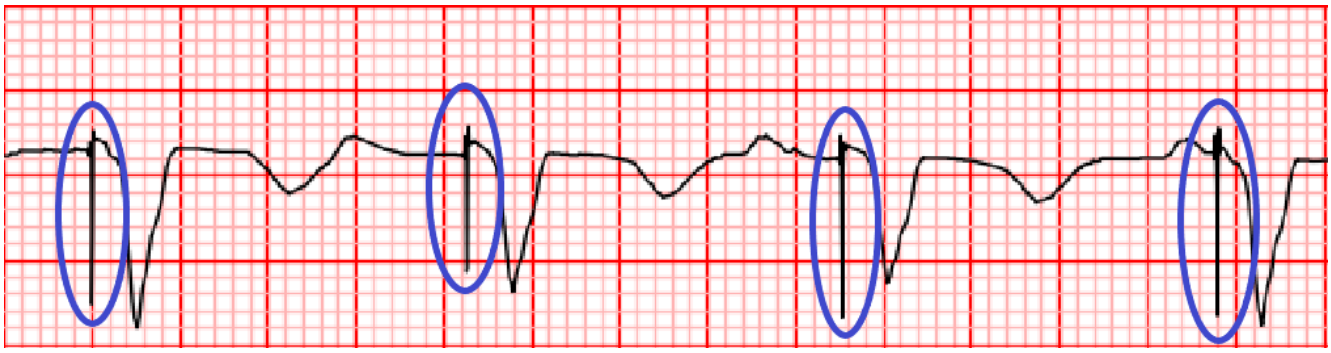
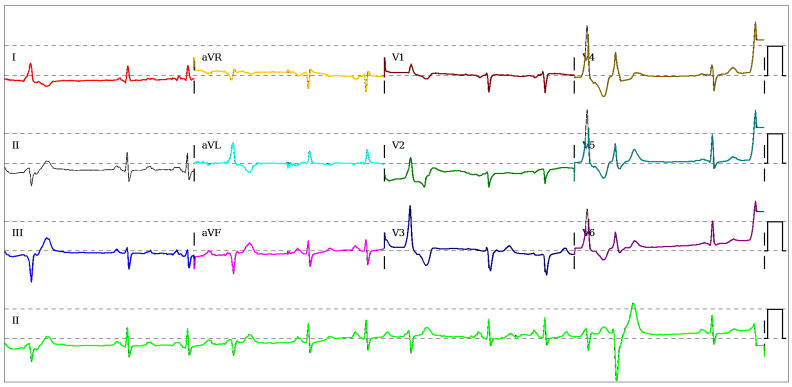


Figura 6.2: Recorte de la imagen de un ECG de un paciente con marcapasos. Para cada latido, se genera un artefacto en forma de segmento vertical (redondeado en azul).

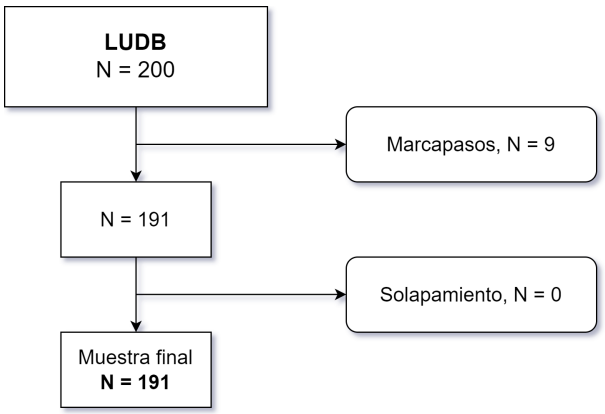
Se identificaron otros dos inconvenientes principales que dificultan la digitalización, relativos a las limitaciones de formato y espacio de las imágenes de ECG. Por un lado, el solapamiento del voltaje con los bordes del papel del ECG o entre derivaciones, especialmente marcado en las derivaciones precordiales. Por otro lado, la mezcla de la identificación alfanumérica de la derivación con las señales del ECG (véanse dichos ejemplos en la Figura 6.3). Ambos problemas se detectaron manualmente bajo la supervisión del cardiólogo y se descartaron del análisis. En concreto, se excluyó el 14.39 % de los ECG de PTB-XL.

(a) Solapamiento

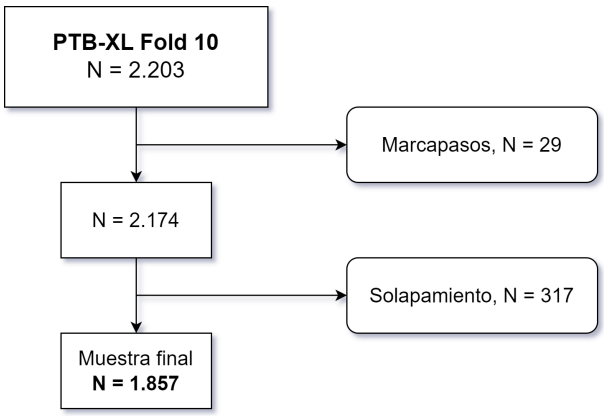


(b) Letras

Figura 6.3: Ejemplos ilustrativos de fallos de ECGMiner en PTB-XL. La señal de ECG original se muestra en negro y las diferentes líneas de color corresponden a las señales de las derivaciones digitalizadas. (a) Salida para el paciente ID 3275 con el cruce de derivaciones en derivaciones precordiales, (b) salida para el paciente ID 3805 con la identificación V1-V3 interpuesta con la señal ECG.



(a) LUDB



(b) PTB-XL

Figura 6.4: Diagramas de flujo del proceso de selección de ECG para cada una de las bases de datos. (a) Selección de LUDB, (b) selección de PTB-XL.

Obsérvese que el ajuste manual de las imágenes de ECG fallidas puede corregir la mayoría de estos problemas. En la Figura 6.4 se puede ver esquemáticamente el flujo de descarte seguido para cada una de las bases de datos.

6.3.2. Métricas de validación

A continuación, se definen las métricas empleadas para la validación. Se utilizaron el coeficiente de correlación de Pearson (PCC) y el error cuadrático medio (RMSE), debido a que son las dos medidas que se utilizan habitualmente en la validación de señales y, más concretamente, en el ámbito de la digitalización de ECG [20, 36, 37].

El PCC mide la fuerza y la dirección de la relación lineal entre los resultados originales y digitalizados. Oscila entre $[-1, 1]$, siendo 1 el valor indica una correlación positiva perfecta, lo que significa que cada vez que una variable aumenta, la otra también lo hace en proporción constante. Cuando el valor de la correlación es -1, indica una correlación negativa perfecta, lo cual quiere decir que cuando una variable aumenta, la otra disminuye en proporción constante. Si el valor de la correlación es 0, indica que no hay una relación lineal clara entre las dos variables analizadas. Se define como:

$$PCC = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\tilde{y}_i - \bar{\tilde{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}\sqrt{\sum_{i=1}^{n}(\tilde{y}_i - \bar{\tilde{y}})^2}} \tag{6.1}$$

donde

- n es el número total de observaciones de cada señal,
- y_i es el valor de la i-ésima observación de la señal original,
- $\bar{y}$ es la media de los valores de la señal original,
- $\tilde{y}_i$ es el valor de la i-ésima observación de la señal digitalizada,
- $\bar{\tilde{y}}$ es la media de los valores de la señal digitalizada.

El RMSE es un valor positivo que cuantifica las diferencias entre el resultado original y el digitalizado en μV (microvoltios). Cuanto más cercano a cero sea el RMSE, mejor será la precisión de la digitalización. Siguiendo la notación descrita para el PCC, el RMSE se define como:

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\tilde{y}_i - y_i)^2}{n}} \tag{6.2}$$

6.3.3. Rendimiento

Para realizar las distintas pruebas, ECGMiner se ejecutó en un ordenador equipado con una CPU AMD Ryzen 5 3600 @ 3,60 GHz (compuesta por 6 núcleos y un total de 12 hilos) bajo el sistema operativo Microsoft Windows 10 (versión 22H2 - compilación OS 19045.2486) utilizando Python 3.10.7. Como referencia de rendimiento, cabe mencionar que el proceso de digitalización de 100 ECG tarda 103 segundos en completarse si ECGMiner se ejecuta de forma secuencial con un solo hilo. Aprovechando la ejecución paralela de ECGMiner con cuatro hilos en el sistema mencionado, este proceso tarda 82 segundos.

6.3.4. Test 1: similitud de las 12 derivaciones (LUDB)

En el panel superior de la Figura 6.1 se muestra un ejemplo del rendimiento de ECGMiner en LUDB. La Tabla 6.20 muestra los valores de PCC y RMSE. El PCC es superior a 0,97 para todas las derivaciones, lo que indica una concordancia sustancial entre las señales original y digitalizada, con desviaciones medias de 10 píxeles (es decir, con la media de la desviación estándar igual a 0,010). En particular, la concordancia es casi perfecta en las derivaciones precordiales (V1, V2, V3, V4, V5 y V6). Estos resultados se ven confirmados por el RMSE. En comparación con los resultados de [36], estos RMSE disminuyen en más de un 44 % de media.

Lead	**PCC**	**RMSE**
I	0,979 ± 0,013	23,736 ± 5,154
II	0,988 ± 0,009	16,595 ± 4,731
III	0,971 ± 0,022	21,060 ± 6,559
aVR	0,987 ± 0,008	18,930 ± 4,087
aVL	0,977 ± 0,024	19,121 ± 6,158
aVF	0,981 ± 0,016	17,152 ± 5,469
V1	0,994 ± 0,003	13,814 ± 2,774
V2	0,993 ± 0,007	12,830 ± 3,139
V3	0,993 ± 0,005	12,649 ± 3,638
V4	0,992 ± 0,004	15,307 ± 3,178
V5	0,993 ± 0,004	13,559 ± 3,280
V6	0,994 ± 0,004	11,384 ± 3,462

Tabla 6.20: Media ± desviación estándar de PCC y RMSE en las 12 derivaciones de los ECG de LUDB.

6.3.5. Test 2: similitud de las 12 derivaciones (PTB-XL)

En el panel inferior de la Figura 6.1 se muestra un ejemplo del rendimiento de ECGMiner en PTB-XL. La tarea de digitalización es mucho más difícil en este caso porque varias señales están corrompidas por artefactos.

El PCC y el RMSE se muestran en la Tabla 6.21. A pesar de lo intrincado de las señales de ECG PTB-XL, el PCC es superior a 0,97 para todas las derivaciones, con desviaciones medias de 10 píxeles. Los valores de RMSE son ligeramente mayores que en LUDB, pero son completamente comparables. Pese a todo ello, aún así se reducen en más de un 24 % (de media) con respecto a los comunicados en [36], en la que se utiliza un conjunto de ECG mucho más sencillo y con un tamaño muestral de 80 ECG, de los cuales además se descartó el 50 % por tener errores demasiado grandes de digitalización.

LEAD	PCC	RMSE
I	0,978 ± 0,016	26,754 ± 9,719
II	0,988 ± 0,012	20,108 ± 8,368
III	0,975 ± 0,021	22,233 ± 10,207
aVR	0,987 ± 0,010	19,921 ± 6,144
aVL	0,980 ± 0,018	18,028 ± 9,133
aVF	0,983 ± 0,014	16,491 ± 7,885
V1	0,995 ± 0,003	17,062 ± 5,530
V2	0,994 ± 0,006	25,382 ± 11,039
V3	0,993 ± 0,006	27,619 ± 12,731
V4	0,991 ± 0,007	30,891 ± 13,677
V5	0,992 ± 0,007	25,978 ± 12,140
V6	0,994 ± 0,006	17,760 ± 9,339

Tabla 6.21: Media ± desviación estándar de PCC y RMSE en las 12 derivaciones de los ECG de PTB-XL.

6.3.6. Test 3: similitud de las ondas P, R y T (LUDB)

En esta etapa de la validación, se compararon distintos elementos de interés clínico extraídos del ECG. Debido a que los algoritmos que realizan la extracción de estas características no son del todo precisos y cuentan con distintos sesgos, se decidió contar con la ayuda de un cardiólogo que realizara las anotaciones personalmente. En primer lugar, las versiones original y digitalizada de los ECG se renombraron e intercambiaron aleatoriamente. Esto se hizo para que el cardiólogo no estuviera sugestionado al tener que etiquetar las dos versiones de cada ECG una seguida de otra. A continuación, el profesional Alberto Pérez Castellanos anotó las ubicaciones en milisegundos, de las ondas P, R y T del ECG con líneas verticales de distintos colores. Por último, se emparejaron las imágenes originales y digitalizadas y se calcularon el PCC y el RMSE para tener en cuenta las diferencias entre ambos tiempos de anotación. Los resultados mostraron un alto rendimiento en términos de PCC (con valores medios de 0,999) para P, R y T. Además, se observó un RMSE medio de 5,5ms a lo largo de las tres ondas. En otras palabras, los errores entre las ubicaciones de las anotaciones fueron inferiores a un píxel.

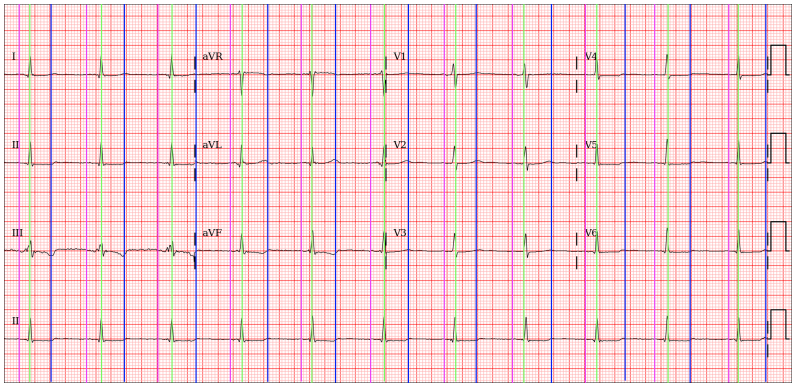
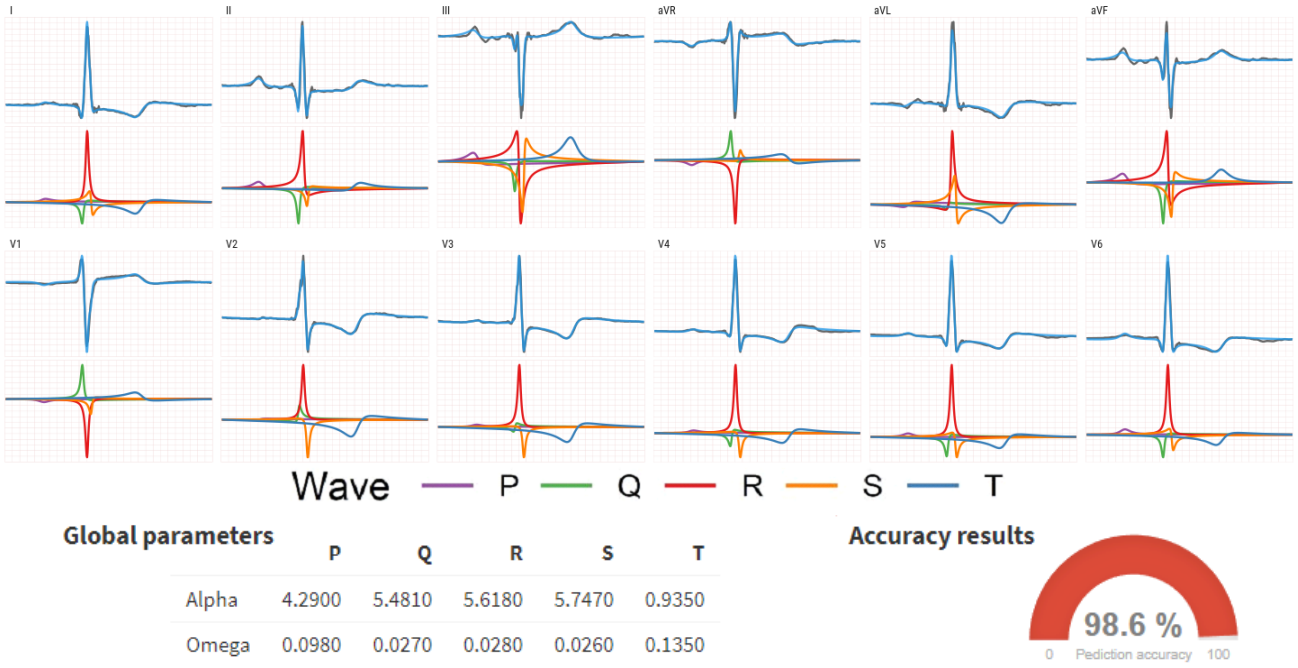


Figura 6.5: Ejemplo de un ECG con las anotaciones realizadas por el cardiólogo. Para cada latido están marcadas en magenta las ondas P, en verde las ondas R y en azul las ondas T.

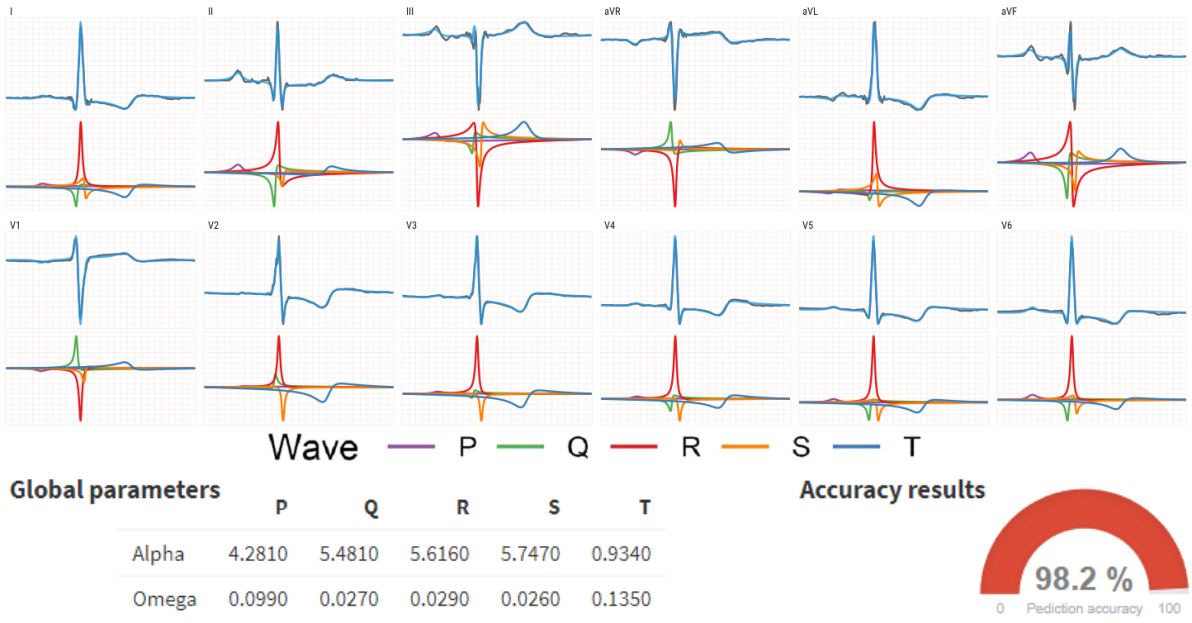
6.4. Aplicación del modelo 3DFMM$_{ecg}$

Tal como se ha explicado en el Capítulo 1, este proyecto se planteó con la intención de aprovechar la digitalización ofrecida por ECGMiner para nutrir el modelo 3DFMM$_{ecg}$ y, así, en un futuro, realizar análisis automáticos de ECG. Por este motivo, se decidió realizar una última prueba con la que comparar los resultados del ajuste del modelo 3DFMM$_{ecg}$ sobre los ECG de LUDB tanto en su versión original como en su versión digitalizada, para ver si realmente difería mucho la estimación entre una versión y otra. En la Figura 6.6 se puede ver un ejemplo de los resultados de ajuste del ECG 15 de LUDB y en la Figura 6.7 los resultados de ajuste de dicho ECG digitalizado.



	P	Q	R	S	T
Alpha	4.2900	5.4810	5.6180	5.7470	0.9350
Omega	0.0980	0.0270	0.0280	0.0260	0.1350

Figura 6.6: Resultados del ajuste del modelo 3DFMM$_{ecg}$ sobre el ECG 15 de LUDB en su versión original.



	P	Q	R	S	T
Alpha	4.2810	5.4810	5.6160	5.7470	0.9340
Omega	0.0990	0.0270	0.0290	0.0260	0.1350

Figura 6.7: Resultados del ajuste del modelo 3DFMM$_{ecg}$ sobre el ECG 15 de LUDB en su versión digitalizada.

En dicho ejemplo se observa que el ajuste es prácticamente idéntico en ambas versiones, manteniéndose algunos de los parámetros iguales (α_Q, α_S, ω_Q, ω_T, etc.) y otros ligeramente distintos pero de un orden muy similar (α_P, α_T, ω_P, ω_R, etc.). En cuanto a la precisión (*accuracy*) del modelo, definida por la medida de ajuste global $\bar{R}$ (descrita en la Sección 2.2.3), solo fue un 0,4 % inferior, pasando de 0,986 a 0,982. El ajuste del resto de ECG de LUDB no llegó a diferir más de un 1,5 % en el peor caso.

A la vista de los resultados, tanto el cardiólogo como el grupo de investigación comentaron el gran potencial de aplicación de la herramienta para obtener datos de señales de ECG, para ser así posteriormente analizados con el modelo 3DFMM$_{ecg}$. De esta forma, se pretende dirigir su desarrollo hacia un procedimiento completo que permita hacer un diagnóstico automático interpretable y eficaz en tiempo real, de manera que pueda ser útil como apoyo en la toma de decisiones de los profesionales sanitarios.

Capítulo 7

Conclusiones

En este Trabajo Fin de Grado se ha desarrollado ECGMiner: un software para digitalizar electrocardiogramas. El software cuenta con las siguientes características:

- Permite trabajar con los formatos de presentación de ECG más habituales (3x4, 6x2, 12x1, Cabrera, etc.).
- Digitaliza de forma precisa las señales del ECG. El proceso de validación ha demostrado una correlación casi perfecta entre las señales de las 12 derivaciones del ECG. Para ello, se han empleado las bases de datos LUDB y PTB-XL. Ambas son abiertas y de libre acceso, lo que facilita la comparación de resultados con respecto a futuros trabajos de digitalización. Además, hasta donde se ha investigado, este es el conjunto de datos más grande que se ha usado para validar la digitalización de ECG.
- Está dotado de una interfaz gráfica de usuario minimalista, con una sola ventana principal y operaciones sencillas.
- Se puede instalar tanto en Windows como en Linux. La guía de instalación consta de menos de diez simples pasos.

Además, ECGMiner se presenta como un software de código abierto, lo cual contribuye a la ciencia abierta y alinea este trabajo con los principios de la Declaración De San Francisco Sobre La Evaluación De La Investigación (DORA)[1]. Tal como se ha descrito anteriormente en este documento, la herramienta se ha alojado en un repositorio de GitHub para que cualquiera pueda disponer de ella bajo licencia MIT.

Finalmente, cabe destacar que este proyecto ha dado lugar a un artículo científico, en el cual figuro como primer autor. Se ha enviado a la revista "Computer Methods and Programs in Biomedicine" (JCR Q1) y se encuentra actualmente en proceso de revisión.

7.1. Limitaciones y trabajo futuro

Las principales limitaciones encontradas de ECGMiner fueron la presencia de cruces y solapamientos, tanto entre las propias señales, como con las pequeñas etiquetas de los nombres de cada derivación. Para tratar de eliminar correctamente los nombres de las derivaciones, se investigará

[1] https://sfdora.org/read/read-the-declaration-espanol/

más a fondo acerca de los motores ópticos de reconocimiento de caracteres que puedan detectar letras del alfabeto romano y que funcionen bien en presencia de ruido. Podría incluso llevarse a cabo la modelización de una red neuronal que ayude a solventar esta tarea. Por otro lado, la solución ante los casos de cruces extremos entre las señales no resulta tan evidente. Aprovechándose de que la herramienta trabaja con la imagen completa en todo momento y que no hace recortes de cada derivación, se puede modificar el algoritmo de extracción para memorizar la morfología de la señal hasta un cierto instante de tiempo. De esta forma, se podría realizar un suavizado en caso de que se encuentre un patrón excesivamente inusual.

Finalmente, se plantea realizar un estudio más en profundidad de la escalabilidad de ECGMiner. Aunque esto no fue un objetivo que se planteó desde un inicio, sí que se decidió añadir el procesamiento por lotes de los ECG para aumentar la velocidad de la herramienta. Las limitaciones propias del intérprete de Python y de la librería Qt no consiguen aprovechar todo el potencial de computación del procesador, por lo que se podría migrar parte o incluso la totalidad de los cálculos internos a lenguajes compilados que ofrezcan mayores prestaciones, tales como C o C++, y utilizar herramientas propias de computación de altas prestaciones, como OpenMP, para lograr ejecuciones multihilo que presenten una alta eficiencia paralela.

7.2. Valoración personal

Este trabajo ha sido una excelente toma de contacto con el mundo de la investigación científica. Desde el primer momento, me he sentido como uno más dentro del grupo con el que he trabajado. He podido desarrollar libremente mis ideas, pero con la enorme ventaja de tener un excelente consejo y evaluación por parte de mis tutoras.

Resulta increíblemente satisfactorio haber logrado todos los objetivos propuestos durante las primeras semanas, pero lo que es verdaderamente gratificante es sentir que mi trabajo será de ayuda para el grupo con el que he estado trabajando e incluso para otros usuarios que quieran utilizarlo. Siento que he aportado mi pequeño grano de arena a la comunidad.

Agradezco enormemente la formación que he adquirido a lo largo de estos años. Los conocimientos de algoritmos y estructuras de datos han sido fundamentales para la construcción la herramienta. Por otro lado, todas las asignaturas relacionadas con el diseño y la ingeniería de software han ayudado a una correcta planificación y estructuración para obtener un software de calidad. Finalmente, he podido reforzar mis conocimientos y mi destreza con el lenguaje de programación Python, además de haber trabajado en el campo de la visión artificial, en el cual no tenía mucha experiencia.

Bibliografía

[1] Cristina Rueda Sabater. *Grupo de Investigación de Inferencia Estadística con Restricciones*. URL: http://www.eio.uva.es/gir-ier/ (Último acceso 16-02-2023).

[2] Cristina Rueda Sabater. *The FMM Project*. URL: http://www.eio.uva.es/the-fmm-project/ (Último acceso 16-02-2023).

[3] Cristina Rueda et al. «A unique cardiac electrocardiographic 3D model. Toward interpretable AI diagnosis». En: *iScience* 25.12 (2022), pág. 105617. ISSN: 2589-0042. DOI: https://doi.org/10.1016/j.isci.2022.105617. URL: https://www.sciencedirect.com/science/article/pii/S2589004222018892.

[4] Adolfo Fernández Santamónica, Cristina Rueda Sabater y Yolanda Larriba González. «Clasificación de enfermedades cardíacas a partir de los parámetros del modelo 3DFMM». Trabajo de Fin de Grado. Departamento de Estadística e Investigación Operativa de la Universidad de Valladolid, 2023.

[5] Antoni Bayés de Luna. *Manual de Electrocardiografía Básica*. 13º ed. CADUCEO MULTIMEDIA, S. L., 2014. Cap. 1.

[6] Npatchett. *Spatial orientation of EKG leads*. 27 de mar. de 2015. URL: https://commons.wikimedia.org/wiki/File:EKG_leads.png (Último acceso 18-03-2023).

[7] Ewingdo. *ECG NSR with RBBB 74 bpm*. 16 de nov. de 2020. URL: https://commons.wikimedia.org/wiki/File:ECG_NSR_with_RBBB_74_bpm.jpg (Último acceso 18-03-2023).

[8] Saira Aziz, Sajid Ahmed y Mohamed-Slim Alouini. «ECG-based machine-learning algorithms for heartbeat classification». En: *Scientific Reports* 11.1 (sep. de 2021). DOI: 10.1038/s41598-021-97118-5. URL: https://doi.org/10.1038/s41598-021-97118-5.

[9] Yolanda Larriba et al. «Order restricted inference in chronobiology». En: *Statistics in Medicine* 39 (3 nov. de 2019), págs. 265-278. DOI: 10.1002/sim.8397. URL: https://onlinelibrary.wiley.com/doi/10.1002/sim.8397.

[10] Alejandro Rodríguez-Collado y Cristina Rueda. «Functional Clustering of Neuronal Signals with FMM Mixture Models». En: 2022.

[11] Cristina Rueda, Yolanda Larriba y Adrian Lamela. «The hidden waves in the ECG uncovered revealing a sound automated interpretation method». En: *Scientific Reports* 11.1 (feb. de 2021), pág. 3724. ISSN: 2045-2322. DOI: 10.1038/s41598-021-82520-w. URL: https://doi.org/10.1038/s41598-021-82520-w.

[12] Cristina Rueda et al. «The FMM Approach to Analyze Biomedical Signals: Theory, Software, Applications and Future». En: *Mathematics* 9.10 (2021). ISSN: 2227-7390. DOI: 10.3390/math9101145. URL: https://www.mdpi.com/2227-7390/9/10/1145.

[13] J H Holt Jr et al. «A study of the human heart as a multiple dipole electrical source. I. Normal adult male subjects». en. En: *Circulation* 40.5 (nov. de 1969), págs. 687-696.

[14] Mario Versaci, Giovanni Angiulli y Fabio La Foresta. «A Modified Heart Dipole Model for the Generation of Pathological ECG Signals». En: *Computation* 8.4 (2020). ISSN: 2079-3197. DOI: 10.3390/computation8040092. URL: https://www.mdpi.com/2079-3197/8/4/92.

[15] Jiapu Pan y Willis J. Tompkins. «A Real-Time QRS Detection Algorithm». En: *IEEE Transactions on Biomedical Engineering* BME-32.3 (1985), págs. 230-236. DOI: 10.1109/TBME.1985.325532.

[16] P. Reddy Gurunatha Swamy, Srinivasan Jayaraman y M. Girish Chandra. «An improved method for digital time series signal generation from scanned ECG records». En: *2010 International Conference on Bioinformatics and Biomedical Technology* (2010), págs. 400-403.

[17] T.P. Exarchos et al. «A platform for wide scale integration and visual representation of medical intelligence in cardiology: the decision support framework». En: *Computers in Cardiology, 2005.* 2005, págs. 167-170. DOI: 10.1109/CIC.2005.1588062.

[18] Sudaraka Mallawaarachchi, M. Prabhavi N. Perera y Nuwan D. Nanayakkara. «Toolkit for extracting electrocardiogram signals from scanned trace reports». En: *2014 IEEE Conference on Biomedical Engineering and Sciences (IECBES).* 2014, págs. 868-873. DOI: 10.1109/IECBES.2014.7047635.

[19] Kim E. Barrett et al. «Origin of the Heartbeat & the Electrical Activity of the Heart». En: *Ganong's Review of Medical Physiology, 25e.* New York, NY: McGraw-Hill Education, 2018. URL: accessmedicine.mhmedical.com/content.aspx?aid=1115831435.

[20] Lakshminarayan Ravichandran et al. «Novel Tool for Complete Digitization of Paper Electrocardiography Data». En: *IEEE Journal of Translational Engineering in Health and Medicine* 1 (2013), págs. 1800107-1800107. DOI: 10.1109/JTEHM.2013.2262024.

[21] Vincenzo Randazzo et al. «Development and Validation of an Algorithm for the Digitization of ECG Paper Images». En: *Sensors* 22.19 (2022). ISSN: 1424-8220. DOI: 10.3390/s22197138. URL: https://www.mdpi.com/1424-8220/22/19/7138.

[22] Siddharth Mishra et al. «ECG Paper Record Digitization and Diagnosis Using Deep Learning». En: *Journal of Medical and Biological Engineering* 41.4 (2021). ISSN: 1424-8220. DOI: 10.1007/s40846-021-00632-0.

[23] B. Hughes, M. Cotterell y R. Mall. *Software Project Managment.* 5ª ed. New Delhi: McGraw-Hill Companies, Inc., 2009. Cap. 4, págs. 76-78.

[24] A. Alshamrani y A. Bahattab. «A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model». En: *International Journal of Computer Science Issues (IJCSI)* 12 (1 2015), págs. 106-111. URL: https://www.ijcsi.org/papers/IJCSI-12-1-1-106-111.pdf.

[25] B. Hughes, M. Cotterell y R. Mall. *Software Project Managment.* 5ª ed. New Delhi: McGraw-Hill Companies, Inc., 2009. Cap. 7.

[26] Indeed. *¿Cuánto se gana como uno Desarrollador/a junior en España?* URL: https://es.indeed.com/career/desarrollador-junior/salaries (Último acceso 25-02-2023).

[27] Balsamiq. *Balsamiq Cloud Plans & Pricing.* URL: https://balsamiq.com/buy/#cloud (Último acceso 25-02-2023).

[28] Astah. *Individual Licensing Options.* URL: https://astah.net/pricing/individual/ (Último acceso 25-02-2023).

[29] Yania Crespo González Carvajal. *Apuntes de la asignatura "Planificación y Diseño de Sistemas Computacionales"*. Departamento de Informática, Escuela de Ingeniería Informática. Universidad de Valladolid. 2023.

[30] John Canny. «A Computational Approach to Edge Detection». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8 (6 1986), págs. 679-698. DOI: 10.1109/TPAMI.1986.4767851. URL: https://ieeexplore.ieee.org/abstract/document/4767851/authors#authors.

[31] S Suzuki y K. Abe. «Topological structural analysis of digitized binary images by border following». En: *Computer Vision, Graphics, and Image Processing* 30 (1 1985), págs. 32-46. DOI: 10.1016/0734-189X(85)90016-7. URL: https://www.sciencedirect.com/science/article/pii/0734189X85900167.

[32] U Ramer. «An iterative procedure for the polygonal approximation of plane curves». En: *Computer Graphics and Image Processing* 1 (3 1972), págs. 244-256. DOI: 10.1016/S0146-664X(72)80017-0. URL: https://www.sciencedirect.com/science/article/pii/S0146664X72800170.

[33] D. H. Douglas y T. K. Peucker. «ALGORITHMS FOR THE REDUCTION OF THE NUMBER OF POINTS REQUIRED TO REPRESENT A DIGITIZED LINE OR ITS CARICATURE». En: *Cartographica: The International Journal for Geographic Information and Geovisualization* 10.2 (1973), págs. 112-122. DOI: 10.3138/fm57-6770-u75u-7727. URL: https://utpjournals.press/doi/abs/10.3138/FM57-6770-U75U-7727.

[34] Nobuyuki Otsu. «A Threshold Selection Method from Gray-Level Histograms». En: *IEEE Transactions on Systems, Man, and Cybernetics* 9 (1 1979), págs. 62-66. DOI: 10.1109/TSMC.1979.4310076. URL: https://ieeexplore.ieee.org/document/4310076/authors#authors.

[35] Fabio Badilini et al. «ECGScan: a method for conversion of paper electrocardiographic printouts to digital electrocardiographic files». En: *Journal of Electrocardiology* 38.4 (2005), págs. 310-318. ISSN: 0022-0736. DOI: 10.1016/j.jelectrocard.2005.04.003. URL: https://www.sciencedirect.com/science/article/pii/S0022073605000774.

[36] JD Fortune et al. «Digitizing ECG image: A new method and open-source software code». En: *Computer Methods and Programs in Biomedicine* 221 (2022), pág. 106890. ISSN: 0169-2607. DOI: 10.1016/j.cmpb.2022.106890. URL: https://www.sciencedirect.com/science/article/pii/S0169260722002723.

[37] Mohammed Baydoun et al. «High Precision Digitization of Paper-Based ECG Records: A Step Toward Machine Learning». En: *IEEE Journal of Translational Engineering in Health and Medicine* 7 (2019), págs. 1-8. ISSN: 2168-2372. DOI: 10.1109/JTEHM.2019.2949784. URL: https://ieeexplore.ieee.org/document/8894038.

[38] Xiaohan Sun et al. «A Novel Method for ECG Paper Records Digitization». En: *Computers in Cardiology, 2019*. Vol. 46. 2019. DOI: 10.22489/CinC.2019.264.

[39] R. Smith. «An Overview of the Tesseract OCR Engine». En: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)* (2007), págs. 629-633. DOI: 10.1109/ICDAR.2007.4376991. URL: https://ieeexplore.ieee.org/abstract/document/4376991/citations?tabFilter=papers#citations.

[40] A. Kalyakulina et al. «LUDB: A New Open-Access Validation Tool for Electrocardiogram Delineation Algorithms». En: *IEEE Access* 8 (2020), págs. 186181-186190. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3029211. URL: https://doi.org/10.1109/ACCESS.2020.3029211.

[41] A. Kalyakulina et al. *Lobachevsky University Electrocardiography Database (version 1.0.1)*. 2021. URL: https://doi.org/10.13026/eegm-h675.

[42] P. Wagner et al. «PTB-XL, a large publicly available electrocardiography dataset». En: *Scientific Data* 7 (2020). DOI: 10.1038/s41597-020-0495-6. URL: https://doi.org/10.1038/s41597-020-0495-6.

[43] P. Wagner et al. *PTB-XL, a large publicly available electrocardiography dataset (version 1.0.1)*. 2020. URL: https://doi.org/10.13026/x4td-x982.

[44] A. L. Goldberger et al. «PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals». En: *Circulation* 101.23 (2000), págs. 215-220. DOI: 10.1161/01.CIR.101.23.e215. URL: https://www.ahajournals.org/doi/10.1161/01.cir.101.23.e215.

[45] Paul Kligfield et al. «Recommendations for the standardization and interpretation of the electrocardiogram: part I: the electrocardiogram and its technology: a scientific statement from the American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology; the American College of Cardiology Foundation; and the Heart Rhythm Society endorsed by the International Society for Computerized Electrocardiology». En: *Circulation* 115.10 (2007), págs. 1306-1324.

Apéndice A

Dependencias software

Para el correcto funcionamiento de ECGMiner es necesario disponer en el sistema de Python 3.10.7 (o superior). Además, también son necesarios los siguientes módulos de Python:

```
altgraph==0.17.3
future==0.18.2
numpy==1.23.5
opencv-python-headless==4.6.0.66
packaging==21.3
pandas==1.5.2
pdf2image==1.16.0
pefile==2022.5.30
Pillow==9.3.0
pyinstaller==5.6.2
pyinstaller-hooks-contrib==2022.13
pyparsing==3.0.9
PyQt5==5.15.7
PyQt5-Qt5==5.15.2
PyQt5-sip==12.11.0
pytesseract==0.3.9
python-dateutil==2.8.2
pytz==2022.6
pywin32-ctypes==0.2.0
scipy==1.9.3
six==1.16.0
```

Apéndice B

Manual de instalación

Los pasos a seguir para instalar ECGMiner son:

1. Clonar el repositorio de GitHub:
```
git clone https://github.com/adofersan/ecg-miner.git
```
2. Instalar Python 3.10.7 (o superior) y añadirlo al PATH.
3. Instalar el paquete "virtualenv" de Python:
```
pip install virtualenv
```
4. Crear un entorno virtual llamado "venv" en la carpeta raíz del proyecto:
```
virtualenv venv
```
5. Activar el entorno virtual.
   - En Linux:
   ```
   source venv/bin/activate
   ```
   - En Windows:
   ```
   ./venv/Scripts/activate
   ```
6. Instalar las dependencias de paquetes de Python.
```
pip install -r requirements.txt
```
7. Construir el proyecto.
```
pyinstaller ECGMiner.spec
```
8. (Opcional) Instalar el motor Tesseract OCR para utilizar la función de extracción de metadatos.
   - En Linux instalarlo y añadirlo al PATH.
   - En Windows instalarlo en `C:\ProgramFiles\Tesseract-OCR\tesseract.exe`.

Apéndice C

Manual de usuario

PASO 1: iniciar ECGMiner

En la Figura C.1 se puede ver el estado inicial de la herramienta. En el lado izquierdo de la GUI se encuentra el panel de configuración con las opciones relativas al formato de entrada y a las opciones de salida. En el centro se sitúa el panel de visualización de ECG, la barra de progreso y el *log*. En el lado derecho (de arriba a abajo) aparecen los botones para cargar ECG, iniciar la digitalización, cambiar el modo de la visualización y cancelar la digitalización. Inicialmente todos los elementos de la interfaz aparecen deshabilitados (color gris) a excepción del botón de carga de ECG (icono de carpeta).

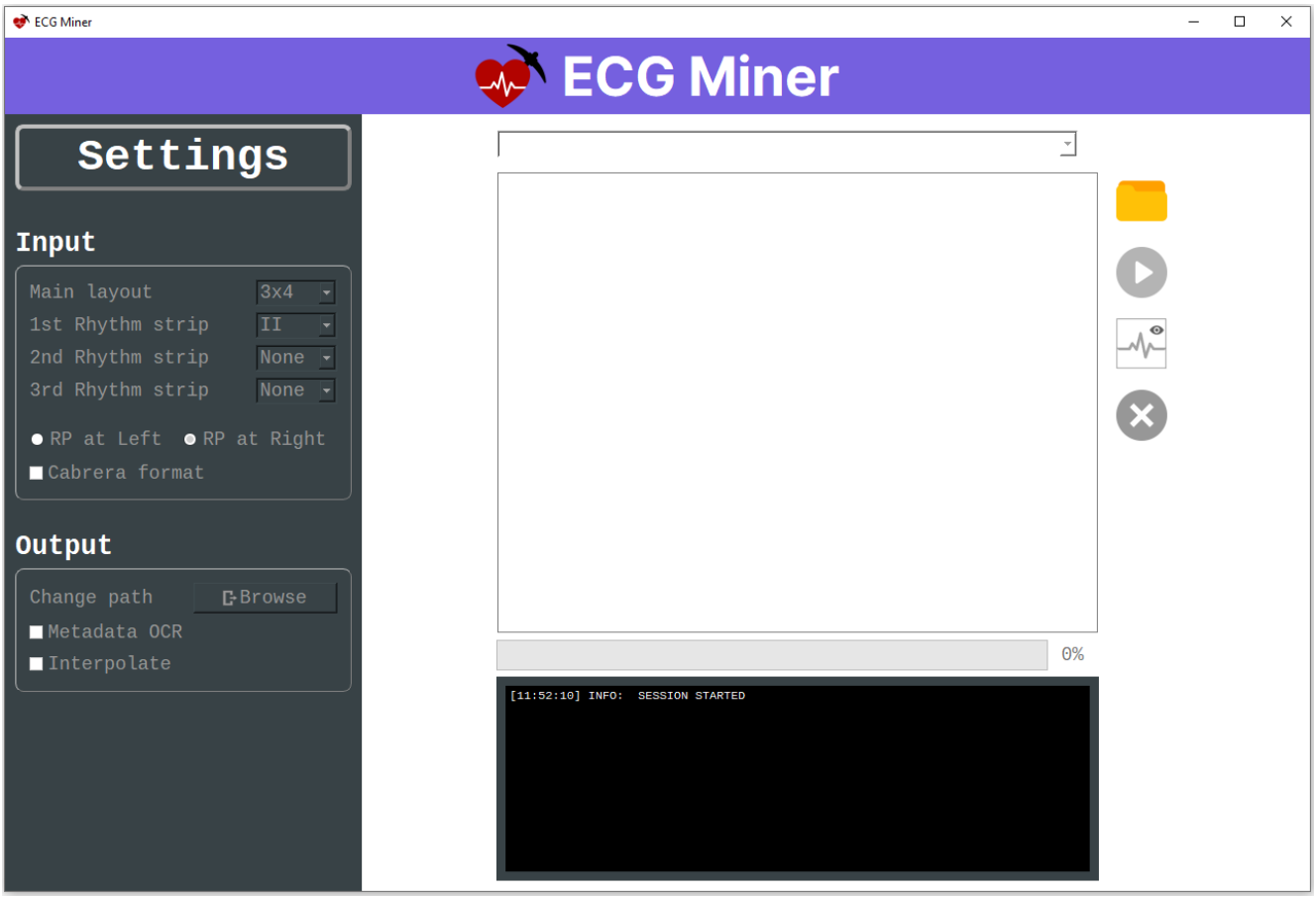


Figura C.1: Captura del estado inicial de ECGMiner.

PASO 2: seleccionar las imágenes ECG a digitalizar

Antes de iniciar el proceso de digitalización hay que indicar a ECGMiner qué imagen(es) hay que digitalizar. Para ello, hay que hacer clic sobre el botón de carga de ECG. Al clicar en él, se abre un cuadro de diálogo que permite seleccionar el/los ECG que se desean digitalizar (ver Figura C.2). La herramienta admite imágenes almacenadas en formato *.jpeg, *.jpg, *.pdf y *.png.

Una vez se hayan seleccionado los ECG, el paso se completa clicando en el botón de `Abrir`.

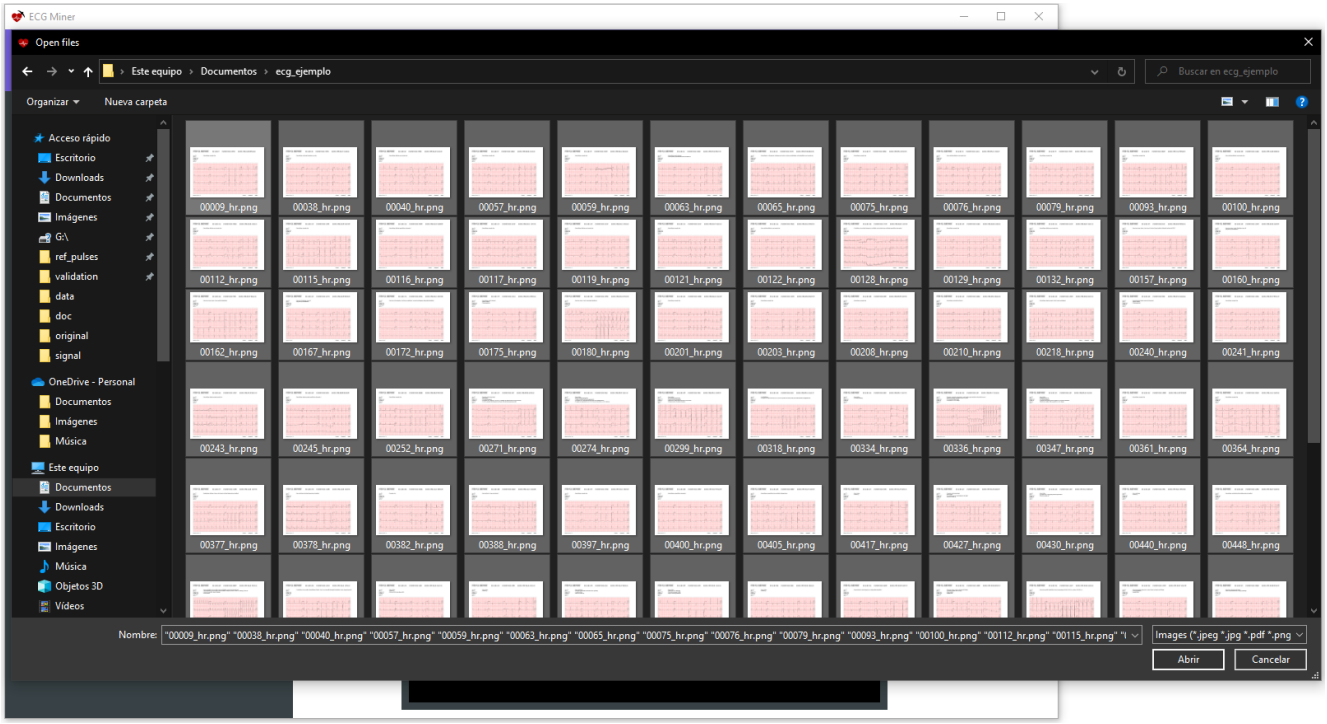


Figura C.2: Captura del proceso de carga de imágenes de ECGMiner.

PASO 3: indicar la configuración de las imágenes cargadas

Tras el PASO 2, la carga de imágenes se habrá hecho efectiva y se habilitarán todos los elementos de la interfaz (aquellos que inicialmente estaban en color gris), para indicar así al usuario que puede proceder a iniciar la digitalización (ver Figura C.3) cuando así lo desee.

En el ejemplo descrito en este manual se ha mantenido la configuración de entrada (`Input`) por defecto y se ha marcado la opción de extraer metadatos y de interpolar las señales a 5.000 observaciones concretamente, pero el usuario puede modificar esta selección para adaptarla a su caso particular.

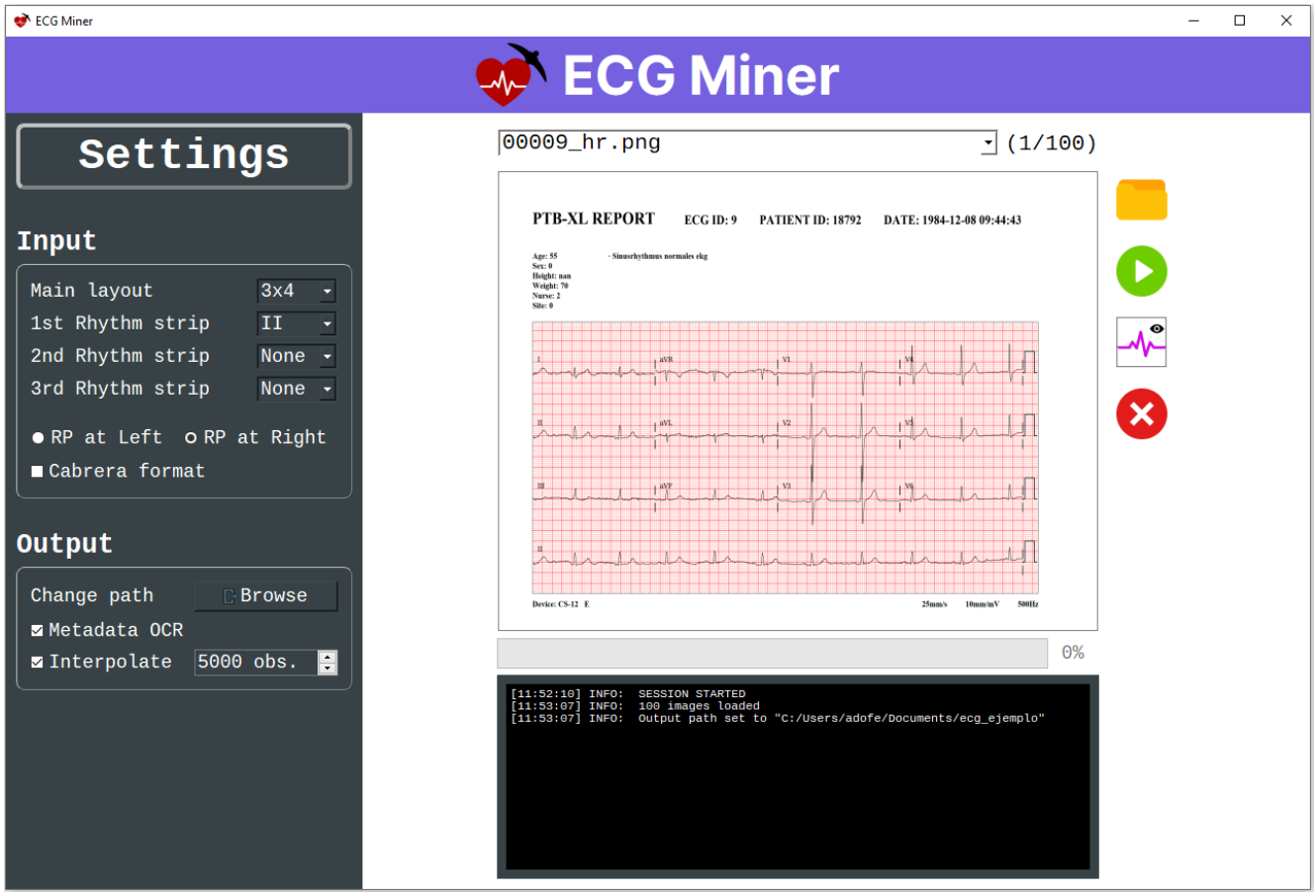


Figura C.3: Captura del momento previo a la digitalización de ECGMiner.

PASO 4: digitalizar

Para iniciar la digitalización se debe clicar en el botón verde (`Play`).

Al iniciarse la digitalización se deshabilitan el cuadro de configuración, el botón de carga y el botón de inicio (ver Figura C.4). Durante el proceso, se puede navegar sobre los distintos ECG mediante el *combo box* situado sobre el panel de visualización. Con el botón de cambio de vista (latido sobre una rejilla cuadrada) se puede cambiar el modo de visualización para ver la traza de un ECG ya digitalizado. De esta manera se puede comprobar si ha habido errores durante el proceso.

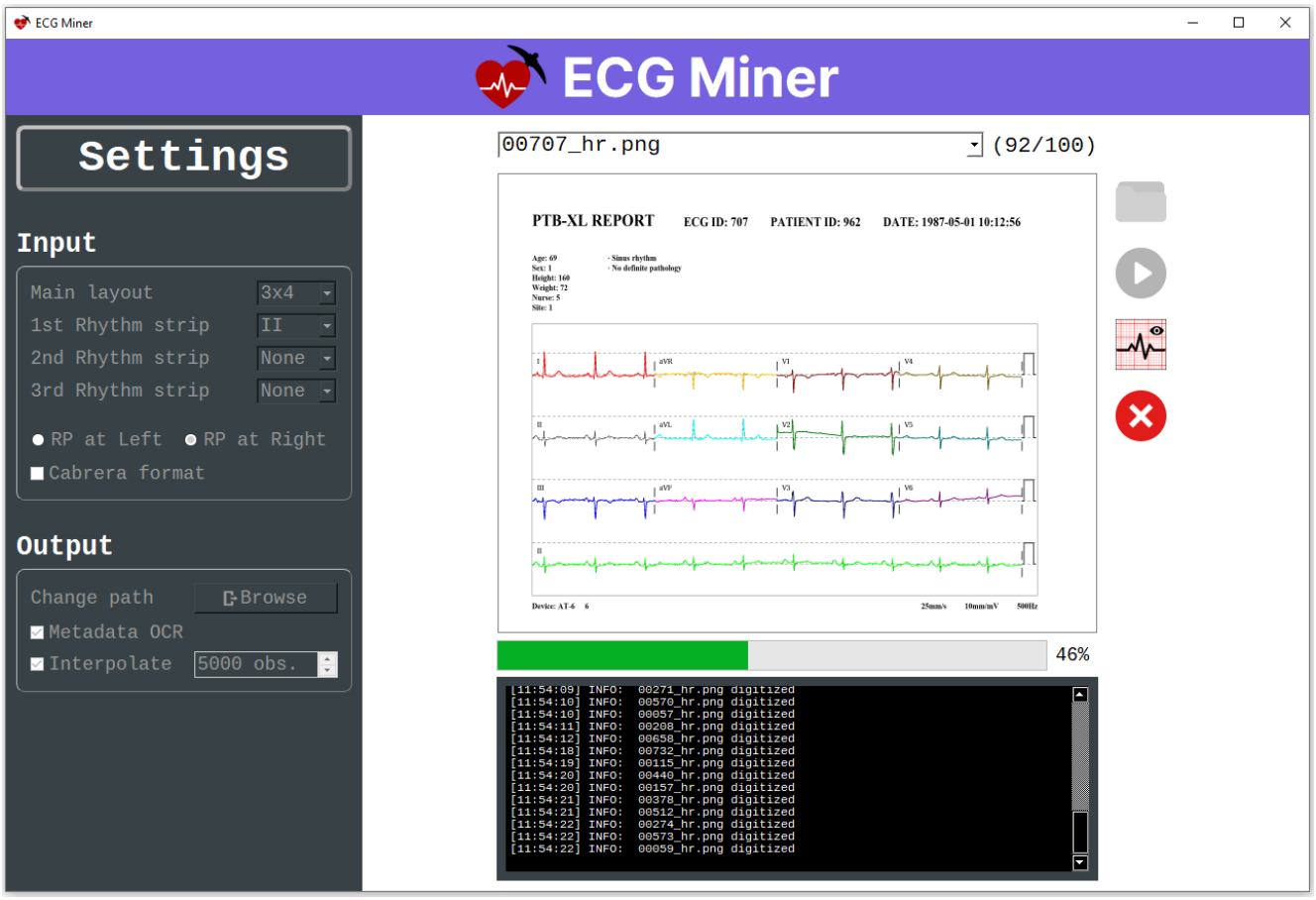


Figura C.4: Captura durante el momento de la digitalización de ECGMiner.

PASO 5: proceso completado

Después de haberse rellenado la barra de progreso al 100 %, todos los ficheros con los resultados de la digitalización de cada ECG aparecen almacenados en el dispositivo (ver Figura C.5). Continúa habilitado el panel de visualización y el botón para ver la traza de digitalización de los ECG. También se encuentra habilitado el botón de carga por si se quiere iniciar una nueva digitalización.

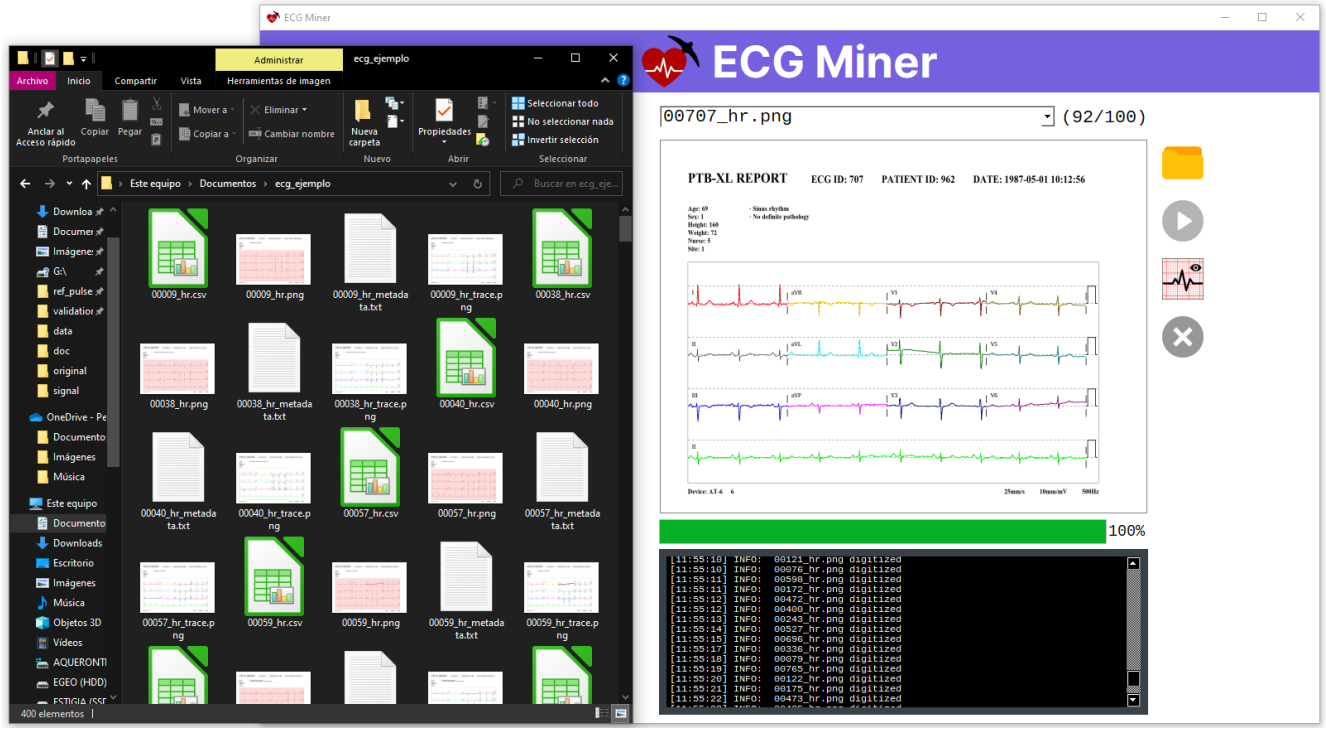


Figura C.5: Captura después de la digitalización de ECGMiner.