UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Aplicación de captura de datasets y demostración de algoritmos de detección de ritmo cardíaco mediante análisis de secuencias de vídeo

Autor: Julia Simón Chico
Tutor: José María Martínez Sánchez

abril 2019

Julia Simón Chico
Aplicación de captura de datasets y demostración de algoritmos de detección de ritmo cardíaco mediante análisis de secuencias de vídeo

Julia Simón Chico
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

En primer lugar, quería agradecérselo a mi tutor, Chema, que es el que me ha dado la oportunidad de realizar este trabajo con el que he podido aprender tanto y es el que me ha estado apoyando y ayudando durante el tiempo que he estado trabajando en él.

Quería agradecérselo a toda mi familia que me ha apoyado en todas las decisiones que he ido durante estos años, pero en especial a mis padres y mis hermanos que han sido los que han estado ahí cuando más los necesitaba.

A todos mis compañeros que han estado conmigo a lo largo de estos años de carrera, y que han hecho que los días en los que teníamos que pasar muchas horas en la universidad se hicieran mucho más llevaderos. En especial, agradecérselo a Álvar mi compañero de laboratorio durante toda la carrera con el que he pasado horas, tanto dentro de la universidad como fuera, ya fuera haciendo prácticas, conociendo nuevos países o en fiestas brasilerias.

No quería dejar sin mencionar a Álvaro el que me ha sufrido durante este último año y me ha ayudado haciéndome magia para que fuera un poco más informática y aprendiera por fin a usar emacs y latex.

Por último, quería agradecérselo a mis amigos de toda la vida que, aunque en algunas épocas he estado un poco desaparecida siempre han estado apoyándome.

Resumen

El objetivo de este trabajo es desarrollar una aplicación que permita la generación de *datasets* de vídeos y probar diferentes algoritmos de detección del ritmo cardíaco mediante el análisis de secuencias de vídeo. Para la creación de los *datasets* la aplicación integra un dispositivo de bluetooth con el cual poder registrar y guardar el ritmo cardíaco real del usuario al que se le está grabando (*groud-truth*).

La principal razón por la cual se ha realizado este trabajo ha sido debido a la importancia que tiene el bombeo de la sangre por parte del corazón a las diferentes partes del cuerpo, y a que gracias a estos métodos de cálculo del ritmo cardíaco mediante el análisis de secuencias de vídeo son mucho menos intrusivos. Con esta aplicación vamos a conseguir probar de manera mucho más sencilla los diferentes algoritmos que se vayan creando en este ámbito.

Basándonos en las necesidades se ha desarrollado una aplicación íntegramente creada en C#, lenguaje elegido principalmente para poder hacer uso de las funcionalidades aportadas por la API de Windows para la Kinect v2, cámara con la que se van a grabar los diferentes vídeos. Con este lenguaje además hemos podido incorporar de manera sencilla una interfaz de usuario para Windows, XAML, que nos facilita la conexión entre las diferentes vistas y la parte lógica de la aplicación.

Para medir el ritmo cardíaco real se va a hacer uso de un dispositivo de Bluetooth de baja energía (BLE) con la ayuda de los protocolos existentes de Bluetooth que nos permiten conectarlo de manera programática y en concreto de las últimas librerías añadidas por Microsoft en la API de UWP (*Universal Windows Platform*) para dispositivos Bluetooth.

Como el objetivo principal de la aplicación es que permita que se puedan probar algoritmos nuevos se ha decidido integrar un algoritmo que hace uso de secuencias de vídeo para calcular el ritmo cardíaco para poder probar esta funcionalidad. El algoritmo que se ha integrado es un algoritmo sencillo que calcula el ritmo cardíaco mediante los cambios en la luz producidos por el bombeo de la sangre.

Por último, se ha creado un pequeño *dataset* de vídeos en color y de imágenes en profundidad que nos permite probar algoritmos nuevos sin necesidad de tener que generar nuevos vídeos.

Palabras clave

Ritmo cardíaco, *dataset*, bluetooth, imágenes en profundidad, Visual Studio, Kinect, C# , frames, UWP, algoritmo, vista, *frame rate*

Abstract

The objective of this Final Degree Thesis is to create an application that allows to create a dataset of videos y try different algorithms that calculates the heart rate using video sequences. For the creation of the datasets, the application integrates a Bluetooth device that allows to register and save the real heart rate of the subject that is being recorded (ground-truth).

The main reason that leads us to elaborate this thesis is the importance that has the heart beat in the blood circulation to all the parts of the body, and because these methods calculate the heart rate using video which less intrusive than other methods that we use nowadays. With this application we are going to try the different algorithms in a simpler way.

Based on these needs we have developed an application in C#, language selected because its haves the application program interface that we need to use the Kinect v2, camera that we are going to use to record the videos. This programming language also let us add a user interface easily using Windows XAML which also facilitates the connexion between the views and the logic part of the application.

To measure the real heart rate, we use a Bluetooth low energy device (BLE) with the help of the existing Bluetooth protocols that we use to connect it in a programmatic way, in particular with the latest libraries added by Microsoft in the UWP (Universal Windows Platform) API for Bluetooth devices.

The main objective of the application is that we can use it to try new algorithms, that is the reason why we have implemented one algorithm that uses videos to calculate the heart rate. This algorithm is simple and uses the light changes in the skin of the subject produced by the heartbeat.

Lastly, we have created a small dataset of videos using color and depth images to try new algorithms in the future without the need of generate new videos.

Keywords

Heart rate, dataset, Bluetooth, depth images, color images Visual Studio, Kinect, C#, frames, UWP, algorithm, views, frame rate

ÍNDICE

Listas

Lista de figuras

1

Introducción

1.1. Motivación

El corazón es uno de los órganos vitales, es el encargado de bombear la sangre para que llegue a todas las partes de nuestro cuerpo, y de esta manera conseguir también transportar el oxígeno necesario para realizar funciones vitales en diferentes órganos y músculos. Por este motivo conocer el ritmo cardíaco, que es la velocidad con la que el corazón va a bombear la sangre, va a tener una gran importancia porque sus variaciones pueden ser significativas para conocer estados fuera de los rangos normales, y de esta forma poder detectar posibles enfermedades.

Hoy en día, los principales métodos que existen para medir el ritmo cardíaco requieren de dispositivos que estén en contacto directo con la persona. Esto implica una cierta invasión personal lo que para ciertas personas puede ser incómodo o incluso un problema, como podría ser el caso de ancianos o bebés los cuales al tener la piel más sensible podrían sufrir efectos negativos con el uso de aparatos que requieran contacto físico. De aquí surge la necesidad de desarrollar métodos que no impliquen un contacto físico, y por ello se van a estudiar las diferentes formas de calcular el ritmo cardíaco a partir de secuencias de vídeo.

Existen dos métodos principales para el cálculo del ritmo cardíaco mediante secuencias de vídeo, uno basado en los cambios de color que se producen por la circulación de la sangre, y otro basado en los micro-movimientos producidos en la cabeza del individuo. En ambos casos son imperceptibles para el ojo humano.

Desde que aparecieron los diferentes estudios que demostraban que era posible el cálculo del ritmo cardíaco se han desarrollado diferentes algoritmos que permiten estimar el ritmo cardíaco analizando las diferencias de color en las secuencias de vídeo o los movimientos en las zonas de interés a lo largo del vídeo.

La finalidad de este trabajo es crear una aplicación que permita probar los diferentes algoritmos que se vayan creando para poder hacer una comparativa entre unos y otros y ver con cuál se obtiene una mejor estimación del ritmo cardíaco. Además, se quiere que la aplicación también sea capaz de

grabar nuevos vídeos y guardarlos en un *dataset*, así como el ritmo cardíaco que tenía la persona en ese momento (*ground-truth*) para conseguir con esta aplicación poder hacer todas las pruebas de manera automática. La parte de la grabación se hará con la cámara Kinect y un dispositivo bluetooth que mida el ritmo cardíaco real.

1.2. Objetivos

El objetivo principal de este trabajo es crear una aplicación que automatice la generación de vídeos tanto en color como en profundidad para que puedan ser usados en esta misma aplicación para probar los diferentes algoritmos desarrollados para el cálculo del ritmo cardíaco.

Para llevar este objetivo a cabo se van a cumplir los siguientes objetivos:

- Estudio del estado del arte.
- Estudio del funcionamiento de C# y el uso de la Kinect v2 en este lenguaje.
- Desarrollo e integración del módulo de grabación con la Kinect v2.
- Estudio del funcionamiento de dispositivos BLE (Bluetooth Low Energy) en C# y en concreto la integración de la Polar H7.
- Desarrollo e integración del módulo de bluetooth.
- Estudio de un algoritmo para el cálculo del ritmo cardíaco.
- Desarrollo del módulo para integrar diferentes algoritmos e inclusión del algoritmo estudiado.
- Generación de un pequeño *dataset* con algunos vídeos en profundidad y color grabados con la aplicación desarrollada.

1.3. Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1: Motivación, objetivos y organización de la memoria.
- Capítulo 2: Estado del arte en el análisis del ritmo cardíaco y su obtención mediante análisis de secuencias de vídeo.
- Capítulo 3: Diseño de la aplicación y sus componentes principales.
- Capítulo 4: Desarrollo de la aplicación.
- Capítulo 5: Conclusiones y trabajo futuro.
- Bibliografía
- Apéndice A: Manual de instalación.

- Apéndice B: Manual de usuario.
- Apéndice C: Manual para inclusión de nuevos algoritmos.

2

Estado del Arte

2.1. Introducción

Para realizar este trabajo se va a proceder a entender cómo funciona el corazón y vamos a proceder a entender los diferentes métodos de cálculo del ritmo cardíaco. Por último, se va a investigar sobre los dispositivos que vamos a utilizar para las grabaciones que son la Kinect y los dispositivos BLE para obtener el ritmo cardíaco real.

El estudio de cómo funciona el corazón nos va a ayudar a entender como se calcula el ritmo cardíaco y la importancia de conocerlo.

Se ha investigado sobre las diferentes técnicas existentes para el cálculo del ritmo cardíaco mediante análisis de secuencias de vídeo para ver cómo enfocar la aplicación que queríamos realizar. Los primeros estudios conocidos son de detección del ritmo cardíaco mediante los cambios de color que se producían debido al bombeo de la sangre. Esto es debido a que la sangre según el momento en que nos encontremos entre latido y latido hay diferentes tonos imperceptibles para el ojo humano.

A partir de la primera publicación, que demostraba que mediante técnicas de amplificación del vídeo podían percibirse cambios de color y también de movimiento [1] se han desarrollado métodos para el cálculo del ritmo cardíaco. Aunque las publicaciones que siguieron a esta primera fueron centradas en los cambios de color, también se han publicado algunas centradas en los micro-movimientos producidos en la cabeza por la circulación de la sangre [2]. Para el estudio de los micro-movimientos, se han hecho también estudios en imágenes de profundidad con el fin de mantener la privacidad del individuo [3].

Debido a esto vamos a enfocar la aplicación de forma que se puedan hacer grabaciones en color para los métodos de color o movimiento y en profundidad para los métodos de movimiento, por ello vamos a estudiar y a hacer uso de la Kinect v2 que permite grabar en ambos tipos. Posteriormente, en la sección 2.4 de este trabajo se explica de manera detallada por que se eligió esta cámara en concreto. Por último, se estudia el uso de dispositivos BLE para medir el ritmo cardíaco real para obtener datos reales de éste con el fin de poder hacer comparaciones con los algoritmos desarrollados.

2.2. Funcionamiento del corazón

El corazón es uno de los órganos vitales, esta formado por una serie de vasos sanguíneos que entran y salen de él, y de varios músculos que se van a encargar de bombear la sangre para que llegue a todas las partes del cuerpo.

En la figura 2.1 podemos ver como es el corazón lo que nos ayudará a entender cómo se produce el ciclo cardíaco.

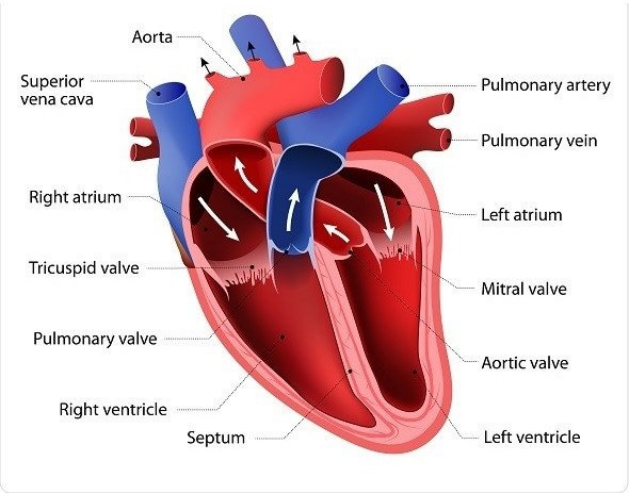

Figura 2.1: Representación del corazón y circulación de la sangre en éste. (Referencia [4])

El ciclo cardíaco dispone de dos fases, una de contracción y otra de relajación. En la fase de relajación, llamada diástole, es en la que se introduce la sangre en el corazón, llenándose los ventrículos. En la de contracción, llamada sístole, es en la cual el corazón expulsa la sangre hacia fuera. Tenemos dos ventrículos que funcionan igual, la diferencia es que uno envía la sangre a la aorta (el izquierdo) y otro a la arteria pulmonar (el derecho). Es la sangre que circula por la aorta la que va a hacer que se produzcan los pequeños movimientos en la cabeza, así como los cambios de color.

La duración de un ciclo cardíaco podemos definirlo como el tiempo que transcurre desde que comienza un ciclo hasta que comienza el siguiente. Es decir, desde el momento de diástole de un ciclo hasta la siguiente diástole. En la figura 2.2 podemos ver una representación del ciclo cardíaco.

El ritmo cardíaco se define como el número de ciclos cardíacos, latidos, que se producen en un determinado intervalo de tiempo. Normalmente se van a medir el número de latidos por minuto. El ritmo cardíaco medio en condiciones de reposo es de entre 50 y 100 latidos por minuto, o pulsaciones por minuto (ppm). Pueden encontrarse por debajo en el caso de deportistas. El ritmo cardíaco varía en función de las necesidades del individuo en cada momento, por ello en momentos en los que se esté haciendo ejercicio físico este aumentará para que se pueda enviar oxígeno a todas las partes del cuerpo de manera rápida.

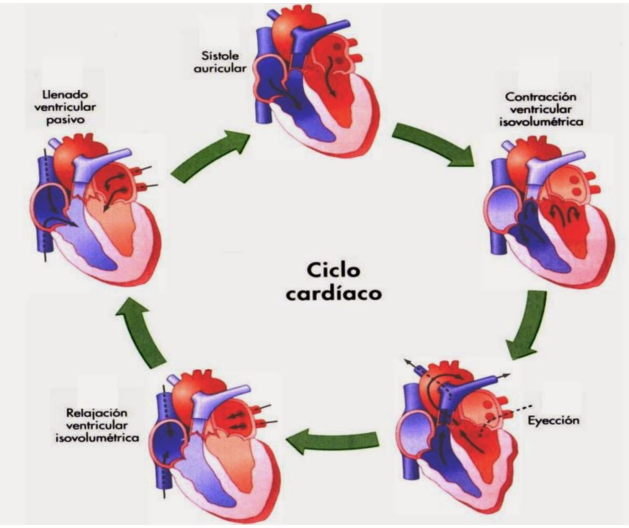

Figura 2.2: Ciclo cardíaco. (Referencia [5])

El ritmo cardíaco es un indicador muy importante, ya que ritmos cardíacos muy altos o bajos en reposo pueden ser significativos de enfermedades.

2.3. Métodos de obtención del ritmo cardíaco mediante análisis de vídeos

Las primeras técnicas de obtención del ritmo cardíaco mediante el análisis de vídeos se remontan al año 2002 cuando se hicieron las primeras aproximaciones utilizando cámaras térmicas [6]. En 2007 se hizo la primera publicación de detección basada en color, en la que se grababa con cámaras convencionales [7]. Pero es desde 2012, con la publicación hecha por el MIT [1] cuando de verdad se empiezan a popularizar estas técnicas y a surgir nuevas investigaciones, apareciendo más estudios en color [6–12] y los primeros basados en movimiento [2].

2.3.1. Métodos basados en cambios de color

El método basado en cambios de color es el más utilizado actualmente. Se basa en los cambios de color, imperceptibles al ojo humano, que se producen por la circulación de la sangre. Estos cambios de color se producen en parte por la presencia de le hemoglobina en cada momento. En la sístole los tonos de piel serán más rojizos que en la fase de diástole.

Existen algunas aplicaciones desarrolladas basándose en este método: en los cambios de color producidos en la parte de la cara o en el dedo por la circulación sanguínea [13]. En concreto a continuación se muestra la aplicación *Cardiio: pulso* que posee dos métodos para medir el ritmo cardíaco,

uno con la cámara frontal para medir el ritmo cardíaco con las imágenes del rostro y otro con la cámara trasera en la cual se pone el dedo sobre ésta y lo mide con los cambios de color en el dedo. En la figura 2.3 podemos ver capturas realizadas en esta aplicación para ver su funcionamiento.

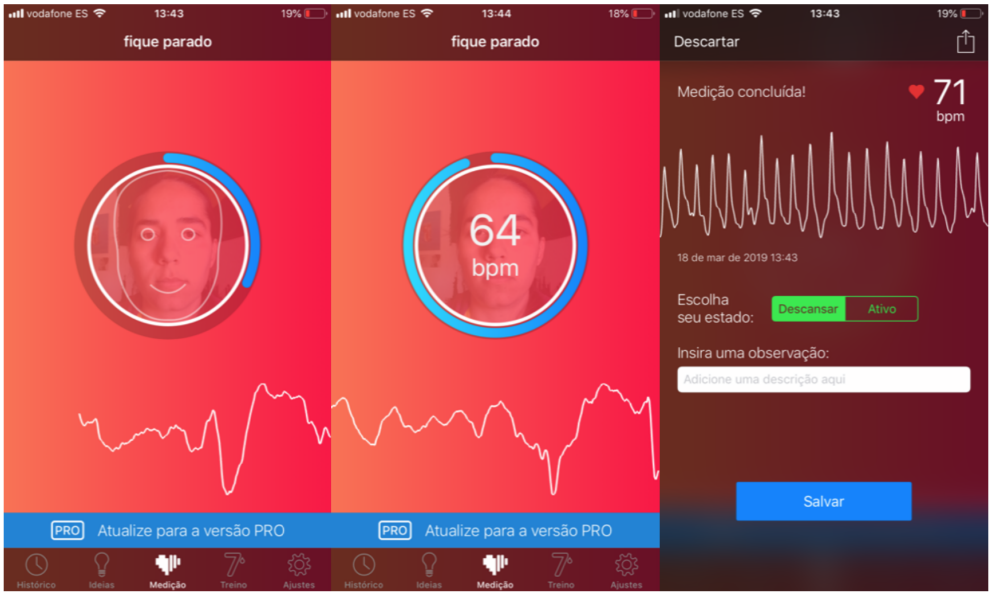

Figura 2.3: Demostración de la aplicación para la medición del ritmo cardíaco con el rostro. En la izquierda vemos como se centra la cara, la segunda durante la medición y la tercera el resultado final que muestra la información obtenida.

En las diferentes investigaciones que hay para calcular el ritmo cardíaco utilizando las variaciones en los cambios de color del vídeo se ha visto que hay los siguientes métodos:

- Comparar *frames* en el espacio de HSI (*Hue, Saturation, Intensity*) [14].
- Separación de las componentes RGB (*Red Green Blue*) para que después de ser normalizadas y procesadas se realice un análisis de componentes independientes (ICA - *Independent Component Analysis*) o un análisis de componentes principales (PCA - *Principal Component Analysis*) [12, 15].
- Separación de las componentes RGB y después realizar un procesado diferente a ICA o PCA [8, 10].
- Uso de la banda G (*Green*) del rostro para aplicar análisis espectral regresivo [16].
- Procesado de las imágenes basándonos en la diferencia de la intensidad del brillo [7].
- Separación de las componentes RGB o de las componentes YCbCr para su descomposición piramidal [1, 17].
- Realizar una operación AND sobre la imagen binarizada en base a la clasificación del tono de piel según la escala cromática de Fitzpatrick y la imagen en el espacio YUV [9].

Para todos estos métodos se pueden aplicar diferentes procesados para calcular la frecuencia predominante, por ejemplo, podríamos hacer uso de la FFT o DCT. También, en estos métodos hay que tener en cuenta la iluminación ya que esta va a influir significativamente en el funcionamiento correcto de estos algoritmos.

Una de las últimas publicaciones que se han realizado sobre el método de procesamiento de color fue el trabajo de fin de grado de Ana Martín Doncel el pasado año 2018 [17], en el cuál se implementaba el método explicado por el MIT [1]. Lo primero que se hace es seleccionar unas regiones de interés sobre las que aplicar el algoritmo, para ello con el algoritmo de Viola-Jones [18] se detecta la cara y se seleccionan la parte de la frente, la nariz y la boca. Después se va a amplificar el vídeo como se muestra en la figura 2.4 Consiste en separar la imagen en distintas bandas espacio-frecuencias para después filtrarlas en el rango de las frecuencias del corazón (40 – 90 en reposo). Estas bandas se amplifican con un valor alpha que se fue variando para ver cuál era el mejor, y después se junta con la imagen original. Sobre esta imagen se calcula el ritmo cardíaco en su espacio transformado, usando la transformada discreta del coseno (DCT) o la transformada discreta de Fourier (DFT), y la frecuencia cardíaca se corresponde con la frecuencia de mayor energía en ese espacio.

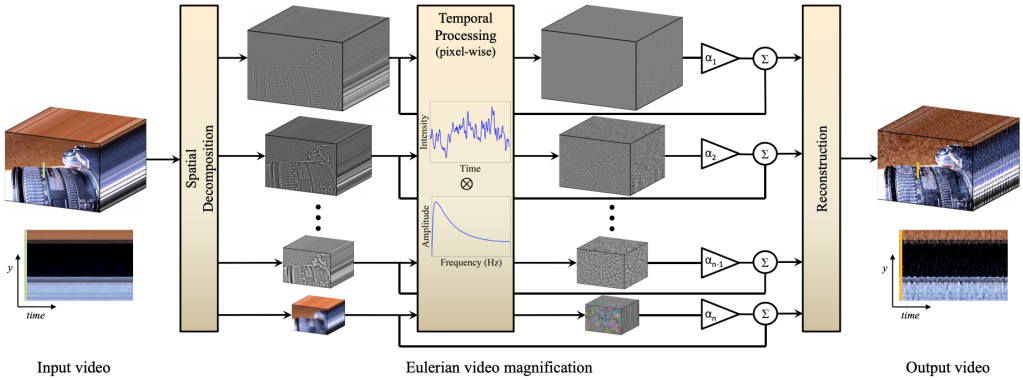

Figura 2.4: Esquema de amplificación del vídeo. (Referencia [1])

2.3.2. Métodos basados en movimiento

Este método se basa en los pequeños movimientos periódicos que se producen en la cabeza del individuo al circular la sangre por las arterias carótidas y la aorta. Este movimiento se debe a la aceleración que se produce en la sangre al producirse la sístole. Estos movimientos son imperceptibles al ojo humano, pero como ocurría con los cambios de color, si amplificamos el vídeo vamos a poder ver estos movimientos.

El primer trabajo que medía el ritmo cardíaco utilizando un método basado en los movimientos producidos por la circulación de la sangre fue el publicado por Balakrishnan en 2013 [2]. En este trabajo, lo que se hace es primero una detección de la cara mediante el algoritmo de Viola-Jones [18].

Después se buscaban una serie de puntos de interés (POI) en la zona de la cara, esto puntos de interés se extraían utilizando el algoritmo de *Good Features to Track* [19]. Una vez obtenidos los puntos de interés, se va a realizar un seguimiento de estos en cada uno de los *frames* del vídeo mediante el algoritmo de KLT (*Kanade Lucas Tracking*) [20], y de esta forma vamos a obtener las trayectorias de los puntos. En la figura 2.5 podemos ver una imagen con la trayectoria de los puntos de interés. Una vez que se tienen las trayectorias se realiza un filtrado de estas para quedarse sólo con las que pueden estar en el rango de la frecuencia cardíaca de un adulto. Se filtra para que se mantengan las frecuencias que se van a encontrar en el rango que interés para el cálculo del ritmo cardíaco. Por último, se hace un análisis de componentes principales (PCA) sobre estas trayectorias filtradas y así se obtienen las pulsaciones por minuto escogiendo la trayectoria que más se ajuste, que es la que tiene una frecuencia principal.

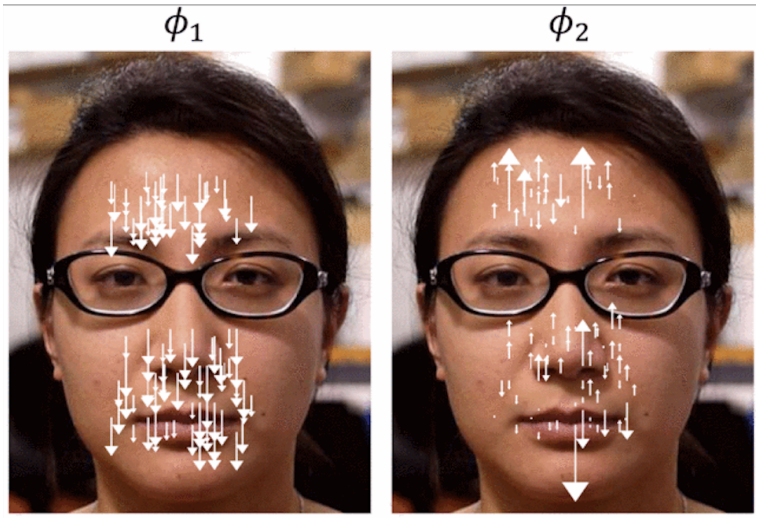

Figura 2.5: Imagen de un usuario con los vectores de la trayectoria de los puntos de interés. (Imagen extraída del trabajo de Balakrishnam [2])

Tras la aparición de este sistema se han ido desarrollando una serie de variaciones respecto a éste para intentar mejorarlo, y también para intentar utilizar imágenes en profundidad que mantengan más la privacidad del individuo que las de color.

En 2014, Irani et al. [21] desde la Universidad de Aalborg proponen una serie de modificaciones en el algoritmo propuesto por el MIT para mejorar el algoritmo. Entre éstas está hacer un suavizado para evitar que afecten mucho los movimientos voluntarios de la cabeza, y se cambió el uso de la DFT (Transformada discreta de Fourier) por la DCT (Transformada discreta del coseno).

En 2015, Erik Velasco Salido utilizó en su Trabajo de Fin de Grado [22] el algoritmo del MIT añadiéndole algunas extensiones con el fin de conseguir que el algoritmo propuesto por el MIT pueda ser utilizado para secuencias vídeo en profundidad grabadas con la Kinect. Es la primera aproximación a secuencias de vídeo que no tengan color.

Por último, en el año 2018, Claudia Fernández Refoyo, partiendo del trabajo realizado por Erik Velasco hizo el primer algoritmo en su Trabajo de Fin de Grado [3] que de manera automática calculaba el ritmo cardíaco con imágenes en profundidad grabadas con la Kinect v2.

2.4. Kinect

En este trabajo buscábamos una aplicación que sirviera para poder probar el mayor número de algoritmos posibles por eso se decidió hacer uso de la Kinect para grabar secuencias de vídeo.

La primera Kinect, la v1 (figura 2.6), fue creada por Microsoft en 2010 con el fin de ser usada para la videoconsola de la XBOX. Esta cámara obtenía imágenes en color (RGB), en profundidad (depth) y en infrarrojos (IR) para permitir jugar a ciertos juegos.

El dispositivo esta formado por una cámara RGB, un sensor de profundidad, un micrófono multidireccional y un procesador específico. Para las imágenes en color se usará la cámara RGB, y para las imágenes en profundidad se utiliza el sensor de profundidad que es un proyector de infrarrojos con un sensor CMOS que permite saber la distancia. Además, la ventaja de estas imágenes de profundidad es que no dependen de la iluminación.

Figura 2.6: Kinect v1 (izquierda) y Kinect v2 (derecha).

En el año 2013 Microsoft sacó la Kinect v2 (figura 2.6), que lo que hacía era mejorar algunos aspectos de la anterior. Algunas de las mejoras fueron que aumentó mucho la calidad de las imágenes en color, se podían rastrear hasta 6 cuerpos ya que, aunque antes detectaban 6 cuerpos sólo se podían seguir 2, la conexión USB pasó de 2.0 a 3.0, y el sistema operativo mínimo para poder procesarla pasó de Windows 7 a Windows 8.1. Por último, un aspecto importante de mejoría de la Kinect v2 frente a la v1 es que para calcular las imágenes en profundidad ésta va a usar ToF (*Time of Flight*). Este método consiste en que la Kinect v2 envía unos rayos en infrarrojo y con el tiempo que tardan en volver calcula la distancia a la que se encuentran los objetos. La versión anterior utilizaba *light patterns*, que consiste en que se envía una forma concreta de luz infrarroja y por la forma en la que esta se deforma al encontrarse con los objetos la cámara es capaz de saber donde están los objetos. El método que utiliza la versión 2 es más estable, y no se van a producir tantas interferencias, además, un problema

de la versión 1 es que no se podrían utilizar varias Kinects a la vez ya que podrían interferir entre ellas mientras que esto no ocurre con la versión dos.

2.5. Dispositivos BLE (*Bluetooth Low Energy*) para medir el ritmo cardíaco

Los dispositivos BLE (*Bluetooth Low Energy*), como su nombre indica son dispositivos de bluetooth de baja energía y coste bastante reducido, que mantienen el rango de alcance similar al de los dispositivos bluetooth originales. Se desarrolló y se empezó a comercializar por *Bluetooth Special Interest Group* [23] para ser utilizado en aplicaciones para el cuidado de la saludo o aplicaciones *fitness* (para registro de actividades deportivas).

Tras hacer un estudio, que se explica con más detalle en la sección 3.2, de las diferentes pulseras que había en el mercado y de los inconvenientes de cada una de ellas, como problemas con los tipos de datos o que estos se encontraban encriptados, se decidió utilizar la banda Polar H7 (figura reffig:polar). Ésta era la más sencilla a la hora de implementarla en el lenguaje que íbamos a utilizar para la parte de bluetooth que eran las extensiones de bluetooth incluida en la última versión de *Visual Studio*, en concreto la parte de UWP (*Universal Windows Platform*). Además, al situarse en el corazón las mediciones son más exactas que con los dispositivos que se sitúan en la muñeca del usuario.

Figura 2.7: Polar H7.

2.6. Conclusiones

Basándonos en todo esto se ha querido desarrollar una aplicación en la que se puedan comprender algoritmos de ambos tipos, es decir que puedan estar basados en movimiento o en color. Debido a esto se escogió la Kinect v2 como cámara para grabar los vídeos ya que nos permite grabar vídeos en color que podrán ser utilizados para métodos tanto basados en movimiento como basados en color y, además, nos permite grabar imágenes en profundidad que nos permite probar nuevos algoritmos de

movimiento con este tipo de imágenes que mantienen más la privacidad del individuo. Esta cámara también nos permite grabar imágenes en infrarrojos que también se podrían utilizar para probar algunos de los algoritmos de movimiento o de los de color que se centran en un único canal. Al haber escogido esta cámara, el software de esta adaptado para Windows, y por ello se decidió utilizar C# por su comodidad frente a C++. Además, este lenguaje nos permitía hacer uso de las librerías de Windows de bluetooth en C# [24–26] para poder integrar el pulsímetro de manera sencilla haciendo uso del mismo lenguaje de programación.

3

Diseño

3.1. Introducción

En este capítulo se van a explicar las decisiones de diseño que se han tomado para el desarrollo de la aplicación. En primer lugar, en la sección 3.2 vamos a explicar los motivos por los que se ha elegido un hardware concreto y los entornos de programación que se van a utilizar para la aplicación. Después, en la sección 3.3 se va a explicar el diseño general de la aplicación el cuál va a estar dividido en diferentes módulos, el módulo principal (subsección 3.3.1), el módulo de grabación, que en nuestra implementación se ha centrado en la grabación para la Kinect v2 (subsección 3.3.2), el módulo del pulsímetro (subsección 3.3.3) y el módulo de los algoritmos (subsección 3.3.4).

3.2. Selección de hardware y entorno de programación

Antes de comenzar a trabajar en el diseño y en el posterior desarrollo de la aplicación se hicieron una serie de elecciones respecto al hardware y el software que se iba a utilizar.

El objetivo principal de la aplicación es que se puedan grabar vídeos los cuales se usarán después para probar los diferentes algoritmos que añadamos. Como hemos visto algunos de los algoritmos para movimiento que existen utilizan imágenes en profundidad por ello se decidió utilizar la Kinect v2 que nos permitía tanto grabar en profundidad como en color. Podríamos haber utilizado la Kinect v1 pero se decidió la v2 ya que como se explicaba en la sección 2.4 se mejoraba la calidad de las imágenes en color, y además al añadir la tecnología de ToF para las imágenes en profundidad se obtenían mejores imágenes de este tipo ya que este método es más estable y tiene menos interferencias.

Para elegir el pulsímetro que íbamos a utilizar se probaron diferentes dispositivos hasta encontrar el más adecuado. Primero se probó con la Polar H7 que es la que se utilizó en el trabajo de fin de grado desarrollado por Sergio Ruiz para una versión diferente de esta aplicación [27], pero, aunque funcionaba se intentó buscar otra que no fuera tan intrusiva. Se probó con la Fitbit Versa, pero se descartó ya que los datos llegan encriptados y no se detectaba cuál era el que se correspondía con el ritmo cardíaco. Por último, se probó con una Xiaomi Band 2, pero los datos que enviaba esta pulsera

estaban en otro formato diferente por lo que para procesarlos había que conocer información detallada de estos la cual no teníamos disponible. Por estos motivos, aunque la Polar H7 era un poco más intrusiva fue la elegida para el desarrollo.

Una vez elegidos los componentes de hardware procedimos a elegir el entorno de programación. La Kinect v2 es perteneciente a Windows por lo que la API estaba únicamente para C#, C++ y Python. Entre estos tres lenguajes al final se decidió utilizar C# ya que la creación de proyectos de este tipo en Visual Studio es bastante sencilla, además, nos permitía crear una interfaz gráfica de manera sencilla utilizando XAML (*eXtensible Application Markup Language*), y nos permitía integrar el módulo del bluetooth con las últimas extensiones de UWP (*Universal Windows Platform*) que se habían incluido en *Visual Studio 2017* cuando se trabaja en ordenadores con Windows 10.

3.3. Diseño general

Para el diseño se ha decidido separar la aplicación en diferentes módulos para facilitar su compresión. Se ha decidido así puesto que en la aplicación tenemos módulos bastante complejos en la parte de la lógica de la aplicación y de esta manera conseguimos separar los posibles errores de estos módulos del resto de la aplicación. Además, conseguimos que, si queremos realizar cambios en algún módulo, no se vayan a ver afectados los demás. Los módulos se han separado según las funciones que realizan:

- Módulo principal de la aplicación: en este módulo tenemos las diferentes vistas, es decir la parte de la interfaz gráfica del usuario, y todas las conexiones con los otros módulos para que, según las opciones elegidas en la vista, se representen de manera correcta los datos seleccionados.
- Módulo de grabación, Kinect v2: en este módulo vamos a tener toda la lógica relacionada con la grabación en nuestra aplicación, en concreto nos vamos a centrar en la grabación con la Kinect v2 que es la que se ha implementado. Se van a explicar las conexiones con la Kinect v2 y obtención de las imágenes de ésta.
- Módulo del pulsímetro: va a ser el módulo que se encarga de la conexión con el pulsímetro, y el intercambio de datos.
- Módulo de los algoritmos: en este módulo vamos a ir añadiendo los diferentes algoritmos que se vayan integrando en la aplicación. Se encarga de con las diferentes imágenes ir calculando el ritmo cardíaco con el algoritmo que elijamos.

En la figura 3.1 podemos observar un diagrama de cómo quedaría la aplicación separada según sus diferentes módulos, y las conexiones entre ellos.

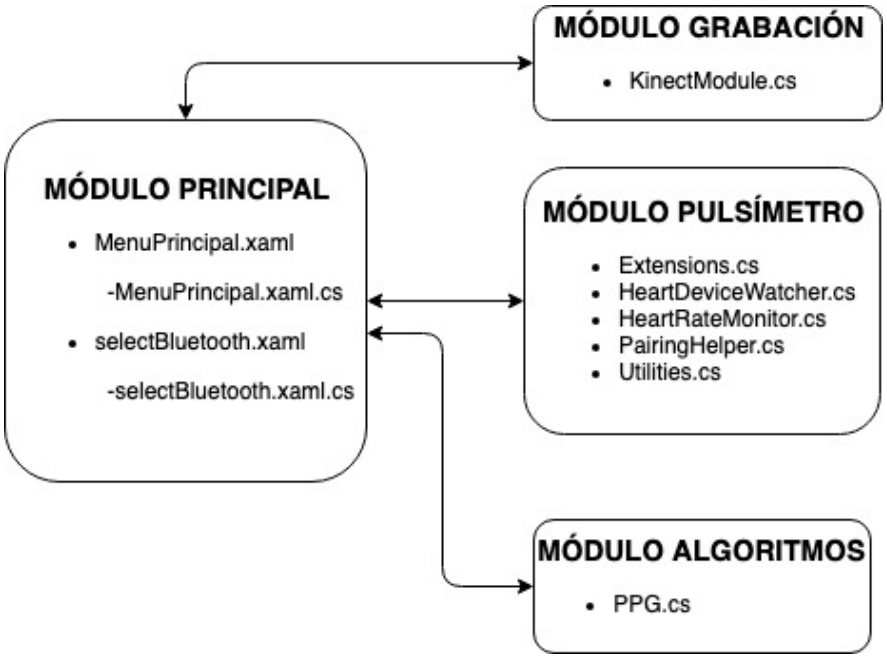

Figura 3.1: Diagrama general de la aplicación.

3.3.1. Módulo principal

El módulo principal está desarrollado todo junto dentro de un mismo proyecto, y es el que se va a encargar de representar la parte visual y de realizar las conexiones con los diferentes módulos auxiliares.

Para la parte de las vistas se estuvieron mirando las diferentes tecnologías a la hora de crear diferentes vistas que se conectaran con C#. Primero se planteó la opción de Windows Forms, pero no era muy efectivo a la hora de representar imágenes por lo que se descartó esta opción. Se plantearon también algunos lenguajes web, pero eso suponía que el ordenador requería estar conectado a la red a la hora de ejecutar el programa. Por ello, finalmente se decidió utilizar XAML (*eXtensible Application Markup Language*).

XAML es un lenguaje descriptivo meta-marcado desarrollado por Microsoft [28]. Este lenguaje pertenece a la parte de .NET del *framework*, es decir la parte relacionada con la presentación visual de Windows, en concreto pertenece a la parte de Microsoft de WPF (*Windows Presentation Fundation*). Y fue creado con la idea de facilitar la creación de aplicaciones C# por lo que para nuestro caso va a ser muy conveniente.

Al formar parte del conjunto de .NET, XAML permite mostrar las imágenes de la Kinect v2 de manera muy sencilla, ya que es capaz de representar las imágenes de color o de profundidad convertidas a *Bitmap*.

Una vez elegido el lenguaje de programación que se iba a utilizar se decidieron las vistas que se

iban a necesitar. Tendremos `MainWindow.xaml` y `selectBluetooth.xaml`.

- `MainWindow.xaml`, va a ser la vista en la que tendremos dos ventanas, una para la grabación de nuevos vídeos y otra para probar los algoritmos.
- `selectBluetooth.xaml`, es la vista en la cual vamos a seleccionar el dispositivo de bluetooth a usar.

Es desde la parte de C# asociada a la vista principal (`MainWindow.xaml.cs`) desde la cuál se van a realizar las conexiones con los distintos módulos auxiliares. Va a actuar como un controlador que intercambia los datos entre las vistas y el modelo.

En la vista vamos a coger los datos necesarios para grabar, para seleccionar el elemento de bluetooth que vamos a utilizar y para probar diferentes algoritmos.

- En la vista del bluetooth simplemente se va a seleccionar uno de los dispositivos emparejados.
- En la vista para grabar se van a obtener los datos del tiempo de grabación, modo de grabación, y opcionalmente el nombre que queremos darle al vídeo grabado.
- En la vista de la prueba de algoritmos se va a seleccionar el modo de los vídeos que se van a usar, el vídeo en concreto a utilizar y el algoritmo a utilizar. En este momento sólo hay un algoritmo que funciona con color por lo que tendremos que seleccionar siempre este modo para poder probarlo.

Además, también desde `MainWindow.xaml.cs` se van a leer los ficheros de datos que tenemos en los cuales está la información de los diferentes vídeos que hay, y los datos de estos vídeos: dirección donde están, ppm reales en ese vídeo y modo de grabación. Estos ficheros están en formato txt por su comodidad, y por como está implementada la aplicación ésta se encarga de escribir en ellos y leerlos siguiendo el formato. En los ficheros, cada línea corresponde con un vídeo, y dentro de estas, separados por tabulaciones tenemos los datos de nombre del directorio, modo de grabación, tiempo de duración, frame rate y nombre del vídeo en caso de que lo tenga. Leyendo estos ficheros nos va a permitir generar las vistas de la aplicación en las cuales tengamos la información de los diferentes vídeos que tenemos disponibles, y los modos en los que estos han sido grabados para poder seleccionar alguno de los algoritmos disponibles para ese tipo de grabaciones.

3.3.2. Módulo de grabación: Kinect v2

El módulo de grabación es en el que se encuentran las comunicaciones con la cámara que va a realizar el vídeo, en este trabajo se ha decidido utilizar la Kinect v2 por lo que en esta sección nos centraremos en explicar como se ha realizado la conexión con esta cámara. Para las comunicaciones se ha seguido la API v2 de Windows para la Kinect [29]. Toda la información que se maneja de la Kinect v2 se

encuentra en la clase `KinectModule` del fichero `KinectModule.cs`. En esta clase encontramos todos los métodos necesarios para establecer la conexión con la Kinect v2 y recibir los datos enviados por ésta: para establecer la conexión, los que abren los diferentes sensores para detectar los distintos tipos de imágenes, y, por último, tenemos los métodos para la transferencia de las imágenes captadas. En estos últimos, tendremos métodos que reciben las *frames* y otros que se encargan de transformar las *frames* recibidas que son del tipo `MultiSourceFrame` en bitmaps que se transformen en imágenes .bmp para que puedan ser representadas en nuestra aplicación mientras se van recibiendo, y para guardarlas para poder usarlas en los algoritmos en casos posteriores. Como las imágenes llegan a mayor velocidad que la de procesamiento por parte del programa se han añadido una serie de colas para que se vayan recibiendo todos los *frames* y encolando, y se vayan desencolando para exportarse como imágenes sin que se pierdan *frames* por el tiempo de procesamiento.

En este módulo se buscaba la mayor eficiencia y que no hubiera pérdida de imágenes, por ello se planteó hacerlo en C++ ya que procesaba las imágenes más deprisa, y aunque se desarrolló este módulo en ese lenguaje, al final se decidió descartar por la dificultad de integrarlo con el resto de la aplicación.

Este módulo va a permitir grabar en color, profundidad (*depth*) o infrarrojos, también permite grabar en todos los métodos a la vez, pero no es lo más recomendable puesto que se pierden algunas de las imágenes. El *frame rate* que se va a utilizar va a ser el automático de la cámara, que es de 30 frames/s en *depth* e infrarrojos y en color es de 15 frames/s. Aunque en teoría la cámara es capaz de grabar a 30 frames/s en color, sólo se van a recibir a la mitad de *frames* por el gran tamaño de estas y la dificultad para procesarlas. Aunque para reproducir un vídeo normal este *frame rate* podría considerarse bajo, para nuestro caso, teniendo en cuenta que las frecuencias en las que se encuentra el ritmo cardíaco que es entre [0.75, 2] Hz (equivalente a [45, 120] pulsaciones/minuto), por el teorema de Nyquist que nos dice que necesitamos una frecuencia de muestreo mayor al doble de la frecuencia que estamos buscando, entonces en este caso, aunque tengamos un *frame rate* de 15 frames/s no vamos a tener problemas al calcular el ritmo cardíaco.

Como acabamos de ver según el teorema de Nyquist podríamos grabar con un *frame rate* bastante menor al utilizado ya que incluso si aumentáramos el rango de frecuencias que analizar, teniendo en cuenta por ejemplo que se quisieran medir hasta ritmos cardíacos de 240 pulsaciones/minuto, su equivalente en hercios sería de 4Hz, el *frame rate* que necesitaríamos seguiría siendo muy bajo, de 8 frames/s. Se plantea que en un futuro se pudiera realizar un subsampling del *dataset* en el cual solo se guardaran algunas de las imágenes para algoritmos que quieran trabajar al límite. También se considera interesante ver si de verdad los algoritmos obtienen los mismos resultados con un mayor o un menor *frame rate*. Si en estos casos funcionaran bien cabría plantearse la opción de guardar menos imágenes de mayor calidad en casos en los que no se disponga de un gran procesador ya que la calidad de la imagen puede afectar más en el cálculo del pulso.

3.3.3. Módulo del pulsímetro

Este módulo se divide en dos partes: una en la cuál se establece la conexión con el dispositivo y otra para la transferencia de los datos. Para que se pueda conectar el dispositivo, este tiene que estar emparejado con el ordenador. Esta parte se puede hacer tanto fuera como dentro de la aplicación. Y para la transferencia de datos, únicamente se van a transferir los datos relacionados con el ritmo cardíaco.

Para este módulo se han utilizado las tecnologías de UWP (*Universal Windows Platform*), en concreto las relacionadas con la parte de conexión con dispositivos (*Windows.devices*), para hacer uso de las extensiones que se añadieron para la conexión con dispositivos de bluetooth BLE. La documentación de estas partes las encontramos en la página de Microsoft [24–26]. En esta documentación se explica como para obtener el ritmo cardíaco tenemos que mirar las diferentes características del dispositivo bluetooth que tenemos conectado hasta que encontramos la que se corresponde con este ritmo cardíaco que será la que tenga el nombre de *HeartRate*. Una vez que hemos seleccionado la característica vamos a tener que añadir un método que sea el *listener*, que se va a encargar de recibir los eventos que se generan por el dispositivo bluetooth del tipo `RateChangedEventArgs`. De ahí ya podríamos obtener las pulsaciones ya que en ese tipo de elemento tenemos el atributo `BeatsPerMinute` que nos va a dar exactamente el valor del ritmo cardíaco en ese momento en el que ha cambiado.

3.3.4. Módulo para los algoritmos

Este módulo es el que se van a ir incluyendo los diferentes algoritmos que se van a probar. Se van a poder añadir tantos algoritmos como se considere, y después desde la vista principal se seleccionará cuál de ellos se quiere utilizar.

Dentro de este módulo, cada clase se corresponde con un algoritmo. En la aplicación actual se ha incluido un único algoritmo básico basado en las variaciones en imágenes en color producidas por los cambios de luz transmitida o reflejada debidas al pulso cardíaco (*PPG Algorithm*). Este algoritmo se encontraba ya implementado en C++ [30] y para la aplicación se a adaptado a C#. En este algoritmo se van a ir leyendo las imágenes que se encuentran en el directorio que se corresponda con el vídeo seleccionado por el usuario, y sobre estas se va a aplicar el algoritmo hasta obtener un valor medio de las pulsaciones.

Se planteó incluir algún otro de los algoritmos disponibles de TFG anteriores desarrollados en Matlab para probarlos, pero se descartó puesto que ese no era el objetivo principal de este trabajo de fin de grado.

En el apéndice C encontramos de manera detallada una guía sobre como se pueden añadir nuevos

algoritmos en la aplicación de manera sencilla.

Cabe destacar que la aplicación se ha decidido desarrollar de tal forma que tenga compatibilidad con *datasets* anteriores, y no únicamente con los nuevos vídeos grabados por lo que cuando vayamos a probar los algoritmos podremos elegir también que *dataset* queremos utilizar. Esto se decidió puesto que teníamos muchos vídeos de diferentes personas, a diferentes distancias y grabados tanto en profundidad como en color que fueron grabados para probar los algoritmos desarrollados por Ana Martín Doncel [17] y Claudia Fernández Refoyo [3]. La diferencia de estos vídeos con respecto a los nuevos es que en estos únicamente tenemos guardado el ritmo cardíaco medio por lo que no tendremos valores instantáneos.

4

Desarrollo

4.1. Módulo principal

En el módulo principal vamos a tener integrada toda la parte de interfaz gráfica, es decir la UI (interfaz del usuario) y toda la parte que actúa como controlador que se encarga de conectar la UI con la lógica de la aplicación para que se realicen las diferentes acciones. En el apéndice B tenemos el manual de usuario de la aplicación.

4.1.1. Interfaz gráfica

La interfaz de usuario va a estar formada por dos vistas principales, una para grabar los vídeos (figura 4.1) y otra para probar los diferentes algoritmos (figura 4.2).

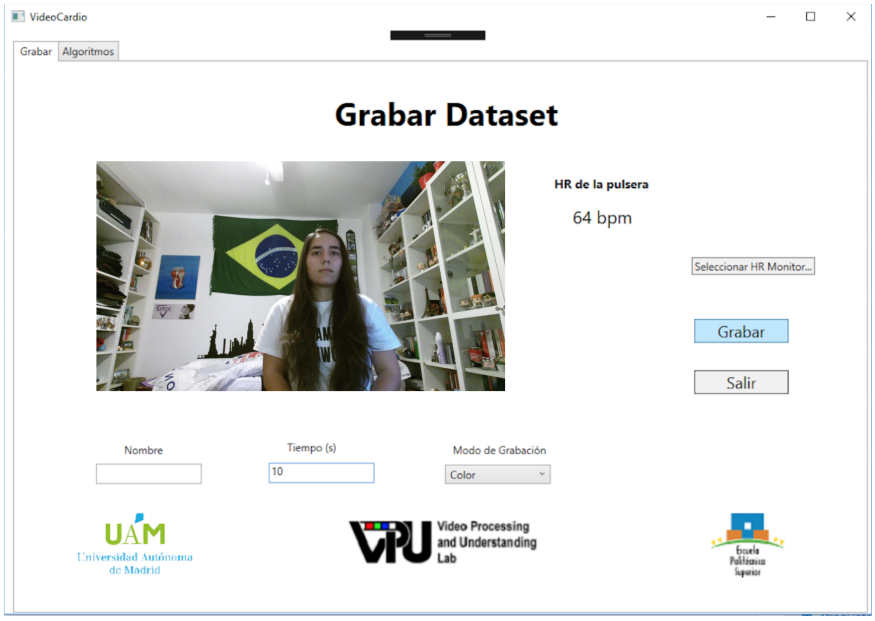

Figura 4.1: Vista de la aplicación en la pestaña de grabar vídeos.

Como podemos ver, en la vista de grabar vamos a seleccionar el tipo de modo en el que quere-

mos grabar, el tiempo y el nombre del vídeo, así como el dispositivo bluetooth que queremos utilizar. Si grabamos sin tener ningún dispositivo de bluetooth seleccionado se grabará el vídeo, pero no se guardarán datos del ritmo cardíaco. Antes de comenzar cualquier tipo de grabación se va a comprobar que se haya introducido un valor de tiempo válido, y un nombre válido (sin espacios). Además, en esta vista, si se selecciona el dispositivo bluetooth se va a ir actualizando el valor de este en la vista de manera periódica.

Para que se puedan actualizar las diferentes partes de la vista se han hecho los métodos asíncronos que se actualizan cada cierto tiempo. En el caso de las pulsaciones, se va a actualizar cada segundo (que serán también los datos de ritmo cardíaco que se guardarán), y en el caso de las imágenes se va a actualizar según el *frame rate* con el que se este grabando o se haya grabado el vídeo concreto, para esta versión de la aplicación el *frame rate* es constante, de 15 frames/s en color y 30 frames/s en *depth* o *infrared*. Estas actualizaciones se producen por métodos que son llamados mediante *timers* que hemos incluido en la aplicación. Con la implementación que tenemos de los *timers* va a funcionar para cualquier tipo de *frame rate* ya que estos se basan en actualizarse cada cierto tiempo según le indiquemos. Se ha añadido el botón de pausar porque por la forma en la que funcionan los *timers*, cuando llegan al final comienza a reproducirse de nuevo.

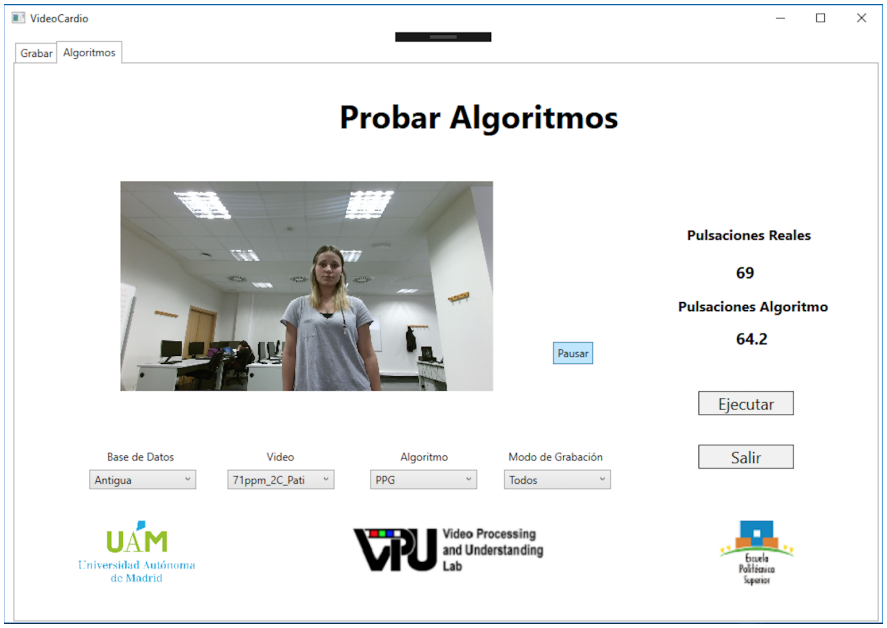

Figura 4.2: Vista de la aplicación en la pestaña de probar algoritmos.

En el caso de la vista para probar algoritmos según se ve en la figura 4.2, se ve que los elementos a seleccionar son:

- La base de datos: Tenemos integradas dos bases de datos, una antigua en la cual sólo está guardado el ritmo cardíaco medio, y una nueva que tiene guardado tanto el valor medio como las pulsaciones tomadas cada segundo de grabación. En la siguiente subsección 4.1.2 se

muestra cómo están guardados los diferentes datos y en la sección 4.5 se explica con detalle como están organizadas las carpetas y archivos txt en los que está toda la información.

- El modo de grabación: color, *depth* o *infrared*.
- El vídeo del cuál se quiere calcular el ritmo cardíaco.
- El algoritmo que vamos a probar.

La selección del vídeo, algoritmo y modo de grabación están relacionadas. Si seleccionamos un modo de grabación en concreto, se van a filtrar los vídeos para que sólo aparezcan los que son de este modo, y lo mismo ocurre con los algoritmos que sólo aparecerán los algoritmos que se pueden probar con ese modo de grabación. Al principio aparece seleccionado el modo de grabación "Todos" que implica que se puedan seleccionar todos los vídeos de los diferentes modos y todos los algoritmos existentes. También, cuando pulsemos un algoritmo, si este esta limitado a un único modo, sólo nos aparecerán los vídeos de ese modo.

4.1.2. Lógica de la aplicación: conexión con los módulos auxiliares

Las llamadas a los módulos auxiliares se producen al pulsar en los diferentes botones.

En la vista de grabación tenemos dos botones qué nos conectan con los módulos auxiliares. Por un lado, tenemos el botón de seleccionar el dispositivo bluetooth, el cual nos lleva a una vista auxiliar que nos permite emparejar nuevos dispositivos bluetooth que no estén emparejados con el ordenador o seleccionar alguno de los que ya está emparejados con el ordenador para medir el ritmo cardiaco. Aunque podemos seleccionar cualquiera de los dispositivos de bluetooth que hay, si no se selecciona el correcto nos aparecerá un mensaje de que no se detecta ninguna señal de ritmo cardíaco, y en casos de que haya problemas de conexión nos mostrara un mensaje que indica que hay un problema al establecer la conexión con el dispositivo, en la figura 4.3 podemos ver el flujo entre las vistas y métodos invocados para emparejar un dispositivo y establecer la conexión. Una vez seleccionado el dispositivo y establecida la conexión, el método `HrParserOnValueChanged` se encargará de recibir de manera asíncrona los datos de ritmo cardiaco recibidos por la aplicación. En la subsección 4.3 se explica cómo funciona el módulo del dispositivo BLE para recibir los datos.

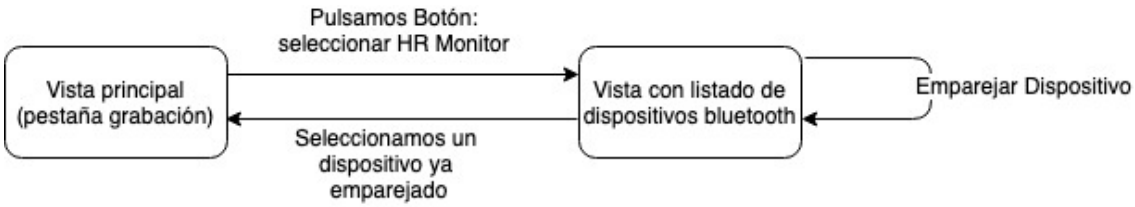

Figura 4.3: Diagrama conexión con el dispositivo BLE.

En esta vista también se encuentra el botón de grabar. En el momento de pulsar ese botón, primero

se van a extraer los valores introducidos del nombre, tiempo y modo para detectar posibles errores. Si todo es correcto se llama al método que comienza la grabación que lo que hace es crear un hilo auxiliar que establece una conexión en la Kinect v2 creando una sesión. Se abren los sensores y se comienzan a recibir los *frames* de la Kinect v2. La sesión se mantiene activa durante el tiempo establecido en el que se están recibiendo imágenes. Una vez pasado este tiempo se van a cerrar los sensores de la Kinect v2 y se cierra la sesión. Por otro lado, durante el tiempo que se va a estar grabando, se va a ir actualizando la imagen que aparece en pantalla según llegan las imágenes, además, el método que controla el tiempo que está activa la Kinect v2 (`recordFramesEnable()`) se va a encargar también de ir guardando los valores del ritmo cardíaco tomados cada segundo. Una vez terminada la grabación, vamos a guardar todos los datos en dos ficheros que después utilizaremos para probar los algoritmos y poder comparar los valores de pulso calculados con el real. En un primer fichero se guardará el nombre del directorio, el tiempo de grabación, método de grabación, el valor del ritmo cardíaco medio y el nombre del vídeo en caso en que se especifique, en la figura 4.4 podemos ver el aspecto de este fichero, donde cada dato esta separado por tabulaciones. Por otro lado, el segundo fichero se utilizará para guardar los datos del ritmo cardíaco instantáneo, se guardará el nombre del directorio del vídeo, el tiempo de duración de este (para saber cuantos datos de ritmo cardíaco tenemos de ese vídeo) y cada una de las mediciones de ritmo cardíaco tomadas cada segundo durante la grabación, un ejemplo de este fichero se encuentra en la figura 4.5.

```
PATH    MODE    DURACION        BPM     NAME
52_JuliaSentada DEPTH   10      52      Julia1
59_Julia_Depth_D4       DEPTH   12      59      Julia3
61_julia        DEPTH   8       61
PatiColor_69ppm COLOR   10      69
```

Figura 4.4: Fichero con la información del vídeo, *path*, modo, duración, ritmo cardíaco y nombre (opcional).

```
PATH                    DURACION        BPM
17-01-2019_18-18-29     10      69      63      67      67      69      71      71      72      70      70
08-02-2019_11-58-11     10      72      76      76      74      73      71      70      71      70      70
```

Figura 4.5: Fichero de datos del ritmo cardíaco, *path*, duración y pulsaciones (*ground-truth*).

En la vista de probar los algoritmos tenemos el botón de ejecutar. Este botón lo que hace es, primero comprobar los datos seleccionados en los combobox, y mandar la información al algoritmo que se quiera probar que se selecciona en uno de estos combobox. En esta información se mandan el directorio donde está el vídeo seleccionado, el modo de grabación si hiciera falta, el tiempo de duración del vídeo y el *frame rate* con el que se grabó el vídeo. Para añadir nuevos algoritmos, se ha explicado de forma detallada en el apéndice C. En esta versión de la grabación no se le pasa al módulo del algoritmo los datos de las pulsaciones (*ground-truth*) puesto que no son necesarios para el cálculo del ritmo cardíaco en este, y el módulo principal que es el que se encarga de mostrar los resultados pos pantalla va a tener estos datos y los devueltos por el algoritmo para poder representarlos. Aquí se

plantea como trabajo futuro añadir un módulo que se encargue de guardar los resultados obtenidos por los algoritmos a vídeo, y hacer las comparaciones entre los resultados obtenidos por los diferentes algoritmos y los datos reales. Se plantea hacer un posible estudio de datos de calidad del algoritmo, calculando la desviación máxima y mínima o el tiempo con más de un 10 % de varianza. Además, también sería interesante añadir en esta comparativa una comparativa entre los resultados obtenidos por los diferentes algoritmos.

En la aplicación, podemos ver que no aparece la opción de grabar y ejecutar un algoritmo a la vez, esto es debido a que sólo tenemos un algoritmo integrado, y no lo calcula a tiempo real por lo que no se ha añadido en la aplicación actual. Pero, como era interesante que esto se pudiera hacer, en el código realizado se ha dejado comentado un botón que permite ejecutar y grabar a la vez, con un semicódigo de cómo se implementaría. En el semicódigo se explica lo que haría el botón, llamaría al método que comienza la grabación de manera asíncrona y a su vez llamaría al algoritmo con los datos del directorio en el que se van a ir grabando los vídeos. Para que funcione bien, el algoritmo debería empezar con un poco de retraso respecto de la grabación ya que la Kinect v2 necesita un poco de tiempo para empezar a grabar ya que se tienen que abrir los sensores, y además tarda un poco en guardar las imágenes.

4.2. Módulo de grabación: Kinect v2

Este módulo implementa la clase `KinectModule` y se encuentra en el fichero correspondiente `KinectModule.cs`, se encarga de realizar la conexión con la Kinect v2 para obtener las imágenes, esto se hace con la ayuda de la API de *Kinect for Windows* [29], para el código base también se utilizó parte del código desarrollado en el trabajo de fin de grado de Sergio García en 2017 [27].

Al instanciar la clase `KinectModule` se crea una sesión con un id en concreto y se genera un fichero *.lock* que almacena este id. La Kinect no puede tener varias sesiones activas, por lo que cada cierto tiempo tenemos que comprobar si nuestra sesión sigue activa o se ha establecido una nueva en cuyo caso habría que liberar los recursos y cerrar la sesión, esto se hace con la ayuda del *lock*.

Posteriormente se abren los sensores, que pueden ser de varios tipos, en concreto nosotros vamos a abrir únicamente los que nos interesan según el modo de grabación. Serán color, *Depth* (profundidad), *infrared* (infrarrojos) o los tres a la vez en el caso de que queramos grabar todas. La Kinect v2 envía las *frames* con el tipo `MultiSourceFrame` por lo que independientemente de los sensores que tengamos activos vamos a recibir los datos de la misma forma. Con el método `KinectFrameArrivedEvent` recibimos la nueva *frame*, y desde ahí llamamos al método `KinectArrived` que se encargar de procesarla en función del tipo de *frame* que sea y de encolarla para que después se guarde en el directorio correspondiente. Se van a encolar debido a que las imágenes llegan más rápido de lo que el ordenador es capaz de procesarlas para guardarlas.

Cuando se van a guardar las imágenes las tenemos como pixeles y las vamos a convertirlas a bitmap (mapa de bits), y cuando las tenemos en este tipo vamos a hacer los ajustes necesarios del tamaño, color y formato de RGB para guardarla como una imagen en formato bmp (mapa de bits).

En la figura 4.6 podemos ver en el diagrama el recorrido de un `MultiSourceFrame` cuando lo recibimos.

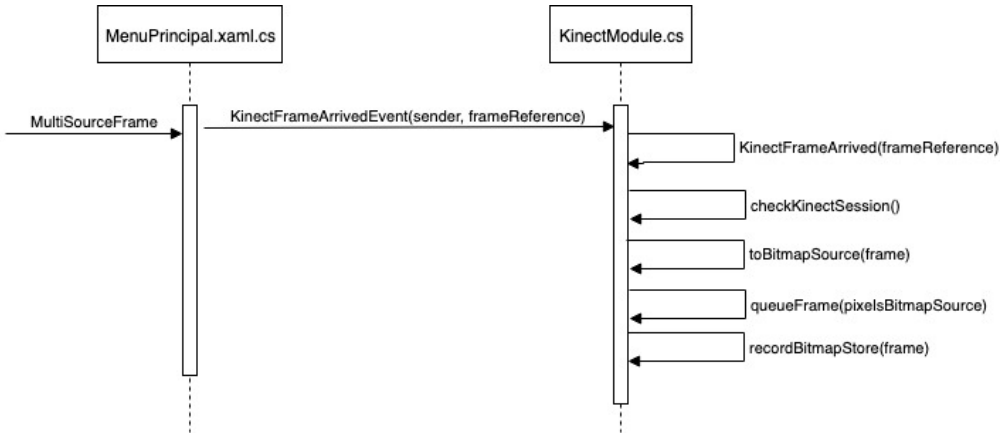

Figura 4.6: Ciclo de vida de una frame (MultiSourceFrame) que llega de la Kinect v2.

Durante el desarrollo de este módulo nos dimos cuenta de una serie de limitaciones que tenía la máquina de la Kinect v2 al conectarse con el ordenador.

La principal limitación es que por mucho que la cámara sea capaz de registrar *frames* de profundidad y de color al mismo tiempo, el ordenador no es capaz de procesar ambas a la vez, y sólo procesa una de ellas en cada momento determinado de tiempo. Al principio se pensó que podía ser debido a que el procesador del ordenador utilizado no era muy alto, pero después se probó en un ordenador más potente y seguía apareciendo el mismo problema. Debido a esto, para las grabaciones se deberá seleccionar un modo en concreto ya que sino es posible que no se registren todos los *frames*.

Otra limitación es que las imágenes en color sólo se recibían con un *frame rate* de 15frames/s, el motivo por el que esto ocurre no se ha llegado a encontrar, pero una posible solución a este problema que se barajó fue transformar el módulo de grabación de la Kinect v2 a C++. De esta forma se conseguía que se guardaran todas las imágenes en color con un *frame rate* de 30frames/s. Esta implementación se encuentra disponible en un repositorio de github publico que se ha creado [31]. Esta opción se decidió desechar puesto que la integración del módulo en C++ en nuestra aplicación era demasiado complejo, y con el *frame rate* que conseguíamos se podían seguir aplicando los algoritmos basados en movimiento y color sin que influyera en los resultados.

Por estos motivos nos hemos planteado si hubiera sido mejor seleccionar otra cámara para realizar las grabaciones, en concreto las de color. Esto se sale del tiempo de TFG asignado, pero se propone como trabajo futuro. Se propone ampliar este módulo para que podamos seleccionar el tipo de cámara que queremos utilizar, la Kinect, una webcam u otros sensores de *depth*. Para añadir esta parte a la

aplicación únicamente tendríamos que modificar la vista de grabación para que permita seleccionar la cámara que vamos a utilizar, y añadir en nuestro proyecto las nuevas formas de captura de vídeo. También se plantea en este ámbito dar la opción de realizar simultáneamente una grabación con dos cámaras de maneras simultánea para poder tener el mismo vídeo desde dos modos diferentes y sobre la misma persona. Para esto quizás sería interesante utilizar la Kinect v1 para las imágenes en profundidad puesto que esta necesita menos procesador que la v2 y así podríamos procesar al mismo tiempo las imágenes recibidas por otra cámara como podría ser una webcam que estuviera grabando en color.

4.3. Módulo pulsímetro

Este módulo va a ser el que se encargue de recibir los datos del ritmo cardíaco que envía el dispositivo BLE. Este módulo no se ha creado desde el principio, sino que se ha utilizado el código de una aplicación ya existente [32] y se han hecho las modificaciones necesarias para que funcionara en nuestra aplicación. Para esto se ha estudiado todo el código y se han modificado algunos aspectos con la ayuda de la documentación de Microsoft [24–26]

Este módulo esta dividido en varias partes, cada una de las cuales se encuentra en un fichero, los ficheros son: `HeartRateDeviceWatcher.cs`, `HeartRateMonitor.cs`, `PairingHelper.cs` y `Utilities.cs`

En el fichero `HeartRateDeviceWatcher.cs` tenemos la parte de la conexión con los dispositivos, se encarga de buscar los dispositivos BLE que hay al alcance del ordenador y mostrarnos una lista con los que hay disponibles, tanto los que tenemos emparejados como los que están sin emparejar. Desde ahí, también se van a poder emparejar nuevos o desemparejar los ya existentes, esto se va a hacer con los métodos `PairDeviceAsync` y `UnpairDeviceAsync` que se encuentran en el archivo `PairingHelper.cs`.

Como se ha explicado antes en el módulo principal, un dispositivo emparejado se puede seleccionar para que sea del cual estamos recibiendo datos del ritmo cardíaco. Una vez que se ha seleccionado el dispositivo, se va a crear la clase `HeartRateMonitor` con el dispositivo correspondiente, que es la que se va a encargar de monitorizarlo. Lo que se hace es una vez creada esta clase se va a establecer con ella una conexión asíncrona que nos permite seguir interactuando con la aplicación mientras se reciben los datos del ritmo cardíaco.

Cuando se crea esta clase se hacen una serie de comprobaciones para ver si es un dispositivo válido al que nos podemos conectar y si el dispositivo envía datos de ritmo cardíaco, si alguna de estas comprobaciones falla se indicará y se desconectará tras informarlo.

A partir de este momento, si la conexión se ha establecido correctamente, se van a ir recibien-

do los datos del ritmo cardíaco con los dos métodos asíncronos que tenemos implementados en el archivo del módulo principal (`MainWindow.xaml.cs`), `HrParserOnValueChange` y `HrDeviceConnectionStatusChanged`. El primero se encarga de recibir los eventos de cambio del valor del ritmo cardíaco, no es periódico, sino que si cambia el valor se envía un evento que lo notifica, y de esta manera vamos a tener el todo momento el valor actual del ritmo cardíaco. El segundo método es el que se encarga de recibir los cambios en el estado de la conexión. Si hay un cambio en la conexión pero el dispositivo sigue conectado lo que vamos a hacer es recoger la información del dispositivo pero se va a seguir escuchando y recibiendo los eventos de este con respecto al ritmo cardíaco, en cambio, si al producirse un cambio en la conexión el dispositivo ya no esta conectado, se va a actualizar la vista principal para indicar que no hay datos del dispositivo.

4.4. Módulo algoritmos

En este módulo de la aplicación es en el que se van a ir introduciendo los diferentes algoritmos utilizados y que queremos probar. En concreto es un proyecto en el que vamos a tener las diferentes clases en C#, las cuales cada una representará a un algoritmo. Al crear una de las clases del algoritmo, lo único que se le suministra es el directorio en el que se encuentran las imágenes sobre las que se va a aplicar el algoritmo. Actualmente, los vídeos están guardados en *frames* separadas, en formato bmp, por lo que los algoritmos que se añadan en un futuro tienen que tener esto en cuenta a la hora de leerlas de los ficheros. En el apéndice C se encuentra explicado con detalle el procedimiento que hay que seguir para integrar un algoritmo nuevo en la aplicación. Para probar el correcto funcionamiento de este módulo se ha adaptado un algoritmo ya existente que calcula el ritmo cardíaco de manera sencilla, este algoritmo se encuentra en el fichero `PPG.cs`. En la siguiente subsección 4.4.1 se explica de manera detallada el funcionamiento de este algoritmo.

4.4.1. Algoritmo PPG

El algoritmo que se ha implementado para probar este módulo esta basado en la fotopletismografía (photoplethysmography, PPG), que mide los cambios del volumen de la sangre mediante las variaciones de luz lo que nos permite calcular el ritmo cardíaco. Este algoritmo fue originalmente implementado por Melisa Mozifian en C++ quien, basándose en un algoritmo ya existente en Python [33], lo adaptó para C++ [30]. En la figura 4.7 podemos ver una imagen en la que se muestra la aplicación original desarrollada en Python en funcionamiento.

Este algoritmo hacía uso de las librerías de OpenCV, Boost y GNU Scientific Library (GSL), pero en este trabajo de fin de grado se ha adaptado para hacer su implementación en C#, y para hacer uso de las librerías que se utilizan para este lenguaje. En concreto, para nuestra adaptación se va a hacer uso de la librería EmguCV que es la correspondiente a OpenCV pero adaptada a C#, y de la librería

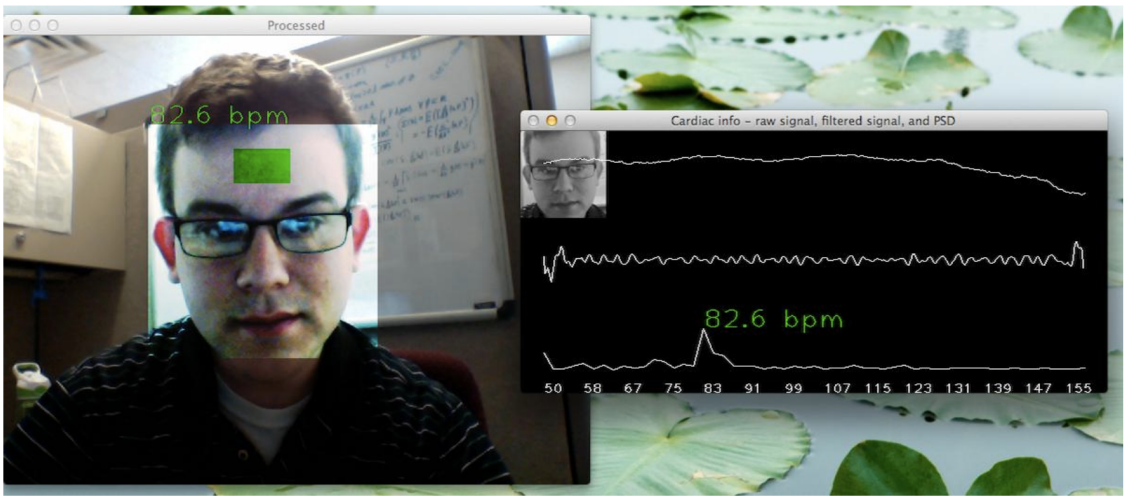

Figura 4.7: Prueba del algoritmo de PPG en su versión en Python [33]

AForge para los diferentes cálculos matemáticos que vamos a necesitar. Adicionalmente también se ha introducido una función para el cálculo de la ventana Hamming necesario para el algoritmo.

En este algoritmo, lo primero que se hace es introducir un detector de cara, *face classifier*, en concreto se ha utilizado el *face classifier* `haarcascade_frontalface_alt.xml`. Después se va a ir leyendo cada una de las imágenes y procesándola. Para cada imagen, buscamos la parte de la frente que es la zona sobre la que se va a aplicar el algoritmo, y se va a calcular la media de esa sección de la cara. Para que se pueda aplicar el algoritmo necesitamos un mínimo de 10 muestras, que se corresponden con la parte de la imagen correspondiente a la frente, y un máximo de 250, en caso de tener más se quitarían las más antiguas y se aplicaría sobre las últimas y en caso de tener menos no se realizaría el cálculo. Una vez que tenemos las muestras necesarias, se les aplica una la ventana de hamming, después se le aplica la transformada de Fourier (FFT), con los datos ahí extraídos, lo filtramos con un filtro de frecuencias para coger los valores que se encuentran entre el máximo y mínimo establecido de pulsaciones (50 y 180), seleccionando los índices del filtro que se corresponden con los valores de la FFT, y ahí seleccionamos la frecuencia que tenga un valor más alto que será la frecuencia predominante y la que se corresponderá con el valor de las pulsaciones. Además, aunque el filtro se tenía en 180, se ha añadido un filtro para que no se añadan los casos de pulsaciones mayores que 145 ya que como los vídeos están hechos en reposo, valores tan altos probablemente sean debidos a cambios de luz altos que no serán causados por el volumen de sangre.

4.5. Generación del dataset

Uno de los objetivos de este trabajo de fin de grado era generar un pequeño *dataset* para probar los diferentes algoritmos que se pudieran implementar en un futuro. Para ello se ha pedido la ayuda a algunos compañeros y familiares para tener vídeos de diferentes personas, tanto en color como

en profundidad. Los vídeos generados se encuentran guardados en una carpeta dentro de la cuál tenemos dos carpetas, una llamada vídeos antigua en la cual tenemos una serie de vídeos que se realizaron antes de implementar el módulo del bluetooth, y en los cuales se apuntaba posteriormente manualmente el ritmo cardíaco medio del vídeo mirando los datos obtenidos en la aplicación de la fitbit ya que para estos se utilizó la pulsera fitbit Versa para mirar las pulsaciones, y otra que se llama vídeos nueva en la cual se encuentran algunos vídeos generados tras tener la aplicación finalizada con todos los módulos integrados. En ambos casos, tenemos registrados en un txt todos los vídeos que hay, el nombre de estos, el modo en el que fueron grabados y las pulsaciones del usuario al grabarlos como se mostraba anteriormente en la figura 4.4. Y para el caso de los vídeos grabados después de integrar el módulo del pulsímetro, hay otro txt en el que se encuentra el nombre del vídeo, la duración y las pulsaciones guardadas cada segundo de grabación como veíamos en la figura 4.5.

No todos los vídeos generados han sido guardados finalmente debido al gran tamaño de estos y a que con la aplicación se pueden generar nuevos vídeos de manera sencilla.

5 Conclusiones y trabajo futuro

5.1. Conclusiones

El objetivo de este trabajo de fin de grado era implementar una aplicación que permitiese generar *datasets* con secuencias de vídeos a la vez que se registraban las pulsaciones reales como *ground-truth*, y que permitiese probar diferentes algoritmos.

Hemos conseguido desarrollar toda la aplicación satisfactoriamente cumpliendo los principales objetivos. Por un lado, tenemos la parte de la aplicación que permite grabar nuevos vídeos con la ayuda de la Kinect v2 y con el pulsímetro registrar las pulsaciones reales. Hemos conseguido conectar la cámara para extraer los datos que esta nos envía y hemos accedido a los datos enviados por un dispositivo BLE, en concreto por la Polar H7. Con estos módulos hemos realizado un pequeño *dataset* con vídeos, tanto en *depth* como en color, grabados a diferentes personas para poder probar los algoritmos. Además, se ha creado un manual de usuario en el que se explica de manera sencilla el uso de la aplicación para que se pueda aumentar el *dataset* en un futuro (apéndice B).

Por otro lado, tenemos la parte de la aplicación en la que hemos integrado el algoritmo PPG que nos ha permitido probar que funciona la parte de algoritmos para vídeos en color concretamente. En esta parte de la aplicación se podrán integrar futuros algoritmos de manera sencilla.

5.2. Trabajo futuro

A la vista del desarrollo que hemos realizado y de las diferentes opciones que se vieron durante el desarrollo de este, se plantea trabajar en el futuro en:

- Implementar nuevos algoritmos en C# que se puedan integrar fácilmente en la aplicación para que se puedan probar ahí directamente y comparar los resultados entre unos y otros.
- Intentar integrar el módulo de la Kinect v2 desarrollado en C++ en la aplicación que tenemos ya que este módulo funciona mucho más rápido que en C#.
- Integrar un nuevo módulo que permita grabar con otras cámaras que no sean la Kinect v2

y que nos permitan grabar en color con un *frame rate* mayor al actualmente conseguido, se podría añadir una webcam, la Kinect v1 y otros sensores multimodalidad (e.g. Asus Xtion2 3D-Sensor).

- Investigar de manera más profunda las diferentes opciones para conectar dispositivos de bluetooth para a la hora de realizar una grabación no necesitar una pulsera Polar H7 ya que se considera que esta es un poco intrusiva, y sería mejor hacerlo con pulseras que midan el ritmo cardíaco.
- Adaptar la aplicación para que permita probar algoritmos a tiempo real, es decir que mientras graba se puedan probar los algoritmos sobre esas imágenes recibidas.
- Añadir conexiones con una base de datos local o externa para guardar los datos de los vídeos y estos y así poder extraerlos de manera mucho más efectiva haciendo uso de consultas SQL para filtrar por modo de grabación o tiempo de duración sin necesidad de tener que recorrernos todo el fichero que tiene esta información. Además, con esta forma nos evitaríamos posibles datos incorrectos ya que al introducirlo en tablas de la base de datos se comprueba el tipo de dato introducido.
- Bajar el *frame rate* hasta el mínimo con el cuál según el teorema de Nyquist los algoritmos deberían funcionar y probar que de verdad se cumplen y si los resultados siguen siendo iguales o parecidos los obtenidos en vídeos con un mayor *frame rate*
- Guardar los resultados obtenidos por diferentes algoritmos para los vídeos en archivos, y poder realizar comparativas entre ellos. Además, hacer cálculos de calidad de los algoritmos, como el cálculo de desviaciones máximas y mínimas, con respecto a las pulsaciones reales.

Bibliografía

[1] H.-Y. Wu, M. Rubinstein, E. Shih, J. Guttag, F. Durand, and W. Freeman, "Eulerian video magnification for revealing subtle changes in the world," *ACM Trans. Graph.*, vol. 31, pp. 65:1–65:8, July 2012.

[2] G. Balakrishnan, F. Durand, and J. Guttag, "Detecting pulse from head motions in video," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3430–3437, June 2013.

[3] C. Fernández Refoyo, *Detección de ritmo cardiaco mediante análisis de secuencias de video en modalidad sin color*.

[4] https://psicologiaymente.com/salud/partes-del-corazon.

[5] https://tucuerpohumano.com/c-sistema-circulatorio/ciclo-cardiaco/.

[6] I. Pavlidis and J. Levine, "Thermal image analysis for polygraph testing," *IEEE Engineering in Medicine and Biology Magazine*, vol. 21, pp. 56–64, Nov 2002.

[7] C. Takano and Y. Ohta, "Heart rate measurement based on a time-lapse image," *Medical Engineering and Physics*, vol. 29, no. 8, pp. 853 – 857, 2007.

[8] W. Verkruysse, L. O. Svaasand, and J. S. Nelson, "Remote plethysmographic imaging using ambient light.," *Opt. Express*, vol. 16, pp. 21434–21445, Dec 2008.

[9] F. Bousefsaf, C. Maaoui, and A. Pruski, "Continuous wavelet filtering on webcam photoplethysmographic signals to remotely assess the instantaneous heart rate," *Biomedical Signal Processing and Control*, vol. 8, no. 6, pp. 568 – 574, 2013.

[10] L. Wei, Y. Tian, Y. Wang, T. Ebrahimi, and T. Huang, "Automatic webcam-based human heart rate measurements using laplacian eigenmap," in *Computer Vision – ACCV 2012* (K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, eds.), (Berlin, Heidelberg), pp. 281–292, Springer Berlin Heidelberg, 2013.

[11] L. Carvalho, H. Virani, and M. Kutty, "Analysis of heart rate monitoring using a webcam," *Analysis*, vol. 3, pp. 6593–6595, 2014.

[12] S. Kwon, H. Kim, and K. S. Park, "Validation of heart rate extraction using video imaging on a built-in camera system of a smartphone," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 2174–2177, Aug 2012.

[13] https://www.cardiio.com/.

[14] P. Sahindrakar, G. de Haan, and I. Kirenko, "Improving motion robustness of contact-less monitoring of heart rate using video analysis," *Technische Universiteit Eindhoven, Department of Mathematics and Computer Science*, 2011.

[15] T. Pursche, J. Krajewski, and R. Moeller, "Video-based heart rate measurement from human faces," in *2012 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 544–545, Jan 2012.

[16] J. B. Bolkhovsky, C. G. Scully, and K. H. Chon, “Statistical analysis of heart rate and heart rate variability monitoring through the use of smart phone cameras,” *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1610–1613, 2012.

[17] A. Martín Doncel, *Detección de ritmo cardíaco mediante análisis de secuencias de vídeo en color*.

[18] P. Viola, M. Jones, *et al.*, “Rapid object detection using a boosted cascade of simple features,” *CVPR (1)*, vol. 1, pp. 511–518, 2001.

[19] and, “Good features to track,” in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, June 1994.

[20] B. D. Lucas, T. Kanade, *et al.*, “An iterative image registration technique with an application to stereo vision,” 1981.

[21] R. Irani, K. Nasrollahi, and T. B. Moeslund, “Improved pulse detection from head motions using dct,” in *2014 international conference on computer vision theory and applications (VISAPP)*, vol. 3, pp. 118–124, IEEE, 2014.

[22] E. Velasco Salido, “Detección de ritmo cardiaco mediante vídeo,” 2015.

[23] https://www.bluetooth.com/.

[24] https://docs.microsoft.com/en-us/uwp/api/windows.devices.

[25] https://docs.microsoft.com/en-us/uwp/api/windows.devices.bluetooth.

[26] https://docs.microsoft.com/en-us/windows/uwp/devices-sensors/bluetooth.

[27] S. García Ruiz, “Aplicación de detección de ritmo cardíaco mediante análisis de secuencias de vídeo,” 2017.

[28] https://docs.microsoft.com/en-us/uwp/api/Windows.UI.Xaml.

[29] https://docs.microsoft.com/en-us/previous-versions/windows/kinect/.

[30] https://github.com/melfm/pulse-detector.

[31] https://github.com/juliasimn/kinectCplus.

[32] https://github.com/kirk-quinbar/HeartRateLE.

[33] https://github.com/thearn/webcam-pulse-detector.

Apéndices

A

Manual de instalación

Para el correcto funcionamiento de la aplicación se necesitan una serie de elementos de hardware y software que se indican a continuación.

A nivel de hardware, además del ordenador, se necesita:

- La Kinect v2.
- El adaptador de Kinect v2 para el ordenador.
- El sensor de frecuencia cardíaca Polar H7.
- Adaptador de bluetooth para el ordenador en caso de que no sea portátil.

Para el funcionamiento general de la aplicación, a nivel de software vamos a necesitar:

- Un ordenador con Windows 10.
- Visual Studio 2017. De visual estudio hay que añadir los workloads:
  - Universal Windows Platform development
  - .NET desktop development
  - ASP.NET and web development
- Kinect for Windows SDK 2.0

Si se hace en alguna versión anterior de Windows o de Visual Studio, la parte de la Kinect puede funcionar, pero la parte del bluetooth no funcionará ya que esta desarrollado con una extensión que se añadió en el WPF de Visual Studio 2017 que sólo funciona sobre Windows 10.

Adicionalmente, al haberse añadido un algoritmo de prueba, para que este funcione correctamente tenemos que añadir en el Visual Studio la extensión de emgu CV, en la página: `http://www.emgu.com/wiki/index.php/Download_And_Installation` encontramos una guía con la instalación necesaria.

Para este algoritmo también necesitamos la librería AForge que ya tenemos incluida en el proyecto, pero si hubiera algún problema con las referencias habría que volver a incluirla, para ello, esta

librería se encuentra disponible en la página oficial: `http://www.aforgenet.com/framework/downloads.html`.

Una vez tenemos todas las instalaciones, abrimos con Visual Studio el .sln del proyecto y lanzamos la aplicación desde ahí.

B

Manual de usuario

La aplicación completa esta dividida en dos partes principales. Una primera que se encarga de realizar los vídeos y guardarlos y otra que se encarga de probar los diferentes algoritmos. En la aplicación tenemos dos pestañas y en cada una de estas se encuentra la funcionalidad de cada una de estas partes. En ambas pestañas tenemos un botón de salir con el cual se cerrará la aplicación.

Primero vamos a explicar como funciona la pestaña encargada de la grabación de los vídeos (ver figura B.1).

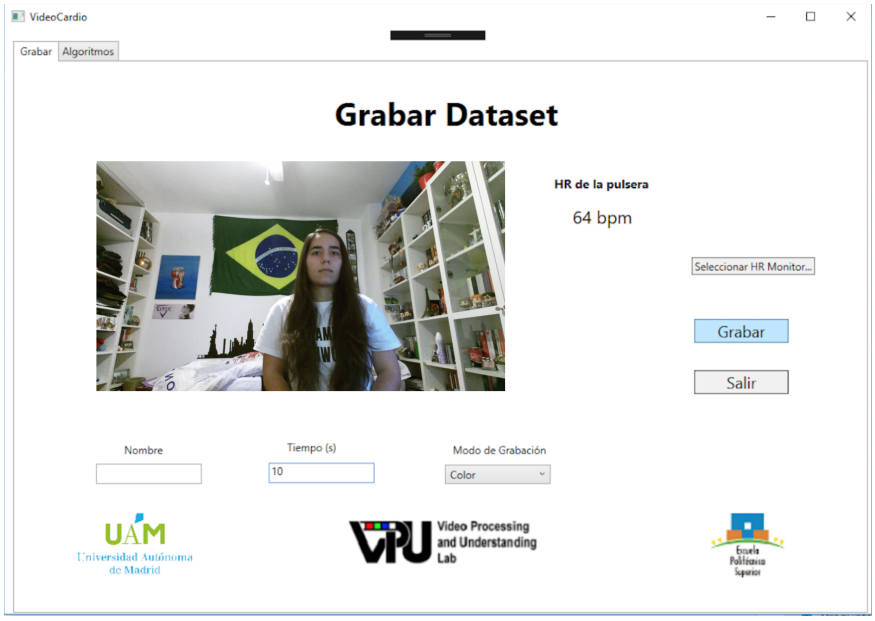

Figura B.1: Vista de la aplicación en la pestaña de grabar vídeos.

En la pestaña encargada de la grabación, el usuario, antes de grabar el vídeo tendrá que pulsar el botón de seleccionar HR Monitor y en la nueva pestaña que aparece (figura B.2) seleccionar Polar H7. Puede ser que ya estuviera emparejada o no. Si aparece en la lista de las no emparejadas habrá que pulsarla y después darle al botón de emparejar, y después en la lista de las emparejadas, habría que seleccionarla y pulsar al botón de seleccionar que aparece abajo. Si ya estuviera emparejada sólo haría falta hacer la segunda parte.

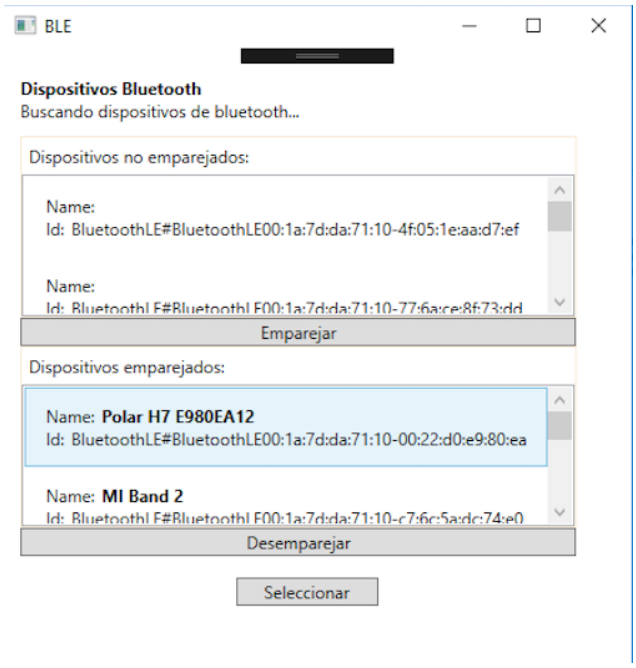

Figura B.2: Vista de la aplicación para conectar el pulsímetro.

Una vez que tenemos el monitor del pulso cardíaco, el usuario tiene que introducir el tiempo, que indica el tiempo de duración del vídeo a realizar (en el campo que tenemos debajo de “Tiempo (s)”, seleccionar un modo de grabación (color, *depth* o *infrared*) en el combobox, y opcionalmente poner un nombre al vídeo (en el campo que tenemos debajo de “Nombre”. Si no se introduce el nombre, se va a guardar con nombre de la fecha y hora actuales. Tras esto se pulsará el botón de grabar y comenzará la grabación. Hay que tener en cuenta que para que el vídeo sea uno válido, el usuario debería estar frente a la cámara, sin más personas en la zona de la cámara para que no hubiera problemas con la detección de la cara, y debería llevar puesto el pulsímetro que usamos.

A continuación se explica en detalle cómo funciona la pestaña que tenemos para probar los algoritmos (ver figura B.3).

En esta pestaña, tenemos que seleccionar:

- La base de datos a utilizar, tenemos una antigua en la que hay vídeos hechos antes de integrar lo del bluetooth por lo que sólo se anotaba manualmente el ritmo cardíaco, o la nueva que tiene los vídeos ya con la parte del pulsímetro funcionando.
- Después tenemos que seleccionar el modo de grabación que queremos, de base están todos, así en el listado de vídeos aparecen todos. Si seleccionamos un modo en concreto, al elegir vídeo sólo aparecen los de ese modo.
- Seleccionamos el vídeo que queremos utilizar de los disponibles.
- Seleccionamos el algoritmo a probar, actualmente sólo está disponible el PPG que funciona únicamente con imágenes en color.

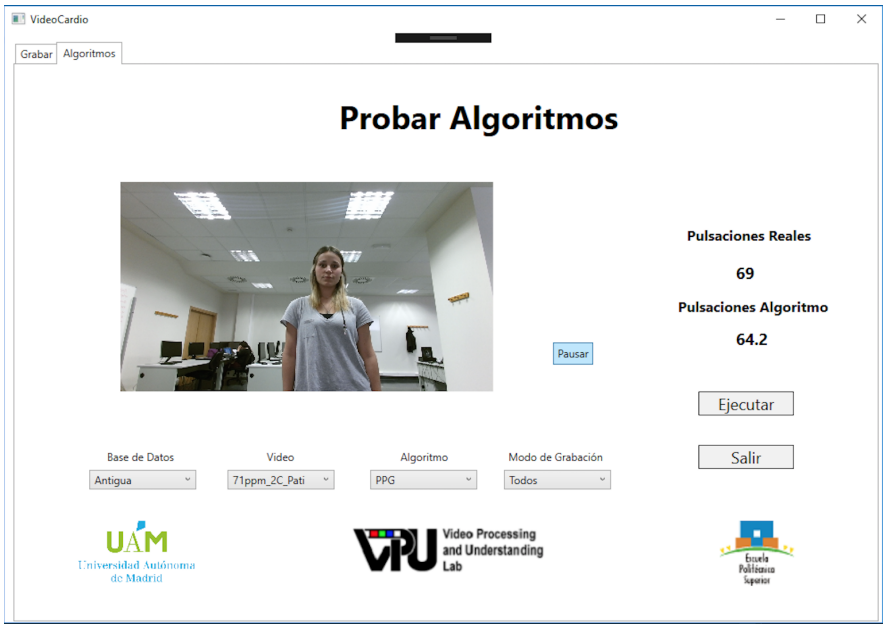

Figura B.3: Vista de la aplicación en la pestaña de probar algoritmos.

Por último, una vez que hemos seleccionado todos estos elementos vamos a pulsar el botón de ejecutar y se comenzará la ejecución del algoritmo seleccionado sobre el vídeo seleccionado para obtener el ritmo cardíaco calculado.

Adicionalmente, vemos que en esta pestaña también hay un botón de pausar, esto se debe a que por la implementación que hay para representar los vídeos, estos una vez que se han terminado de representar, vuelven a empezar automáticamente, pero si pulsamos el botón de pausar dejarán de reproducirse.

C

Añadir un nuevo algoritmo

Antes de añadir un nuevo algoritmo hay que tener en cuenta que el programa esta diseñado para que el principal suministre la ruta de donde se encuentra el vídeo, y que el propio algoritmo se encargue de leer el vídeo. Los vídeos no están guardados en formato vídeo, sino en imágenes, una imagen por cada *frame*. Por este motivo, tenemos dos opciones para enviar los datos al algoritmo:

- Mandar la ruta y que el algoritmo sabiendo como es la ruta y que todos los *frames* están guardados con el nombre de la carpeta, el tipo de imagen y acaban en .bmp se encargue de leerlos.
- Transformas las imágenes a vídeo de forma externa al programa, y en vez de pasar la ruta donde están las imágenes, le pasemos directamente la ruta completa con el nombre del vídeo que vamos a leer.

Para obtener los datos del *ground-truth* se encuentran en un fichero txt en el cual con el nombre del video, que sería el mismo que se corresponde a la última parte de la ruta del fichero, se pueden encontrar las pulsaciones guardadas cada segundo mientras se grababa. Esto, como se comentó anteriormente, sólo ocurre en los vídeos de la nueva base de datos. Los de la antigua base de datos únicamente tienen el valor medio que se podrá pasar por parámetro al algoritmo desde el módulo principal.

Pasos para añadir un nuevo algoritmo:

- Añadir al proyecto que se llama “Algoritmos” las clases necesarias para el algoritmo.
- En el proyecto principal (App_Kinect_HeartRate):
  - En la parte visual, acceder a la ventana de algoritmos, y en el combobox “combobox_algoritmo” y añadir un ítem con el algoritmo que vamos a añadir.
  - En el fichero, “MainWindow.xaml.cs”, en el método “ejecutar_btn_Click”, en el ultimo if, añadir la condición necesaria para que se ejecute el algoritmo. Por ejemplo, en algoritmo ya introducido, sólo funciona con imágenes de color, y se llama PPG, su condición sería: “combobox_algoritmo.Text == ’PPG’ && combobox_tipo.Text ==

'Color' ". Y dentro de ese if la información necesaria para que se ejecute el algoritmo.

- En el proyecto hay añadidas dos librerías externas que se necesitan para el algoritmo introducido, estas librerías son emguCV y AForge, además sólo están añadidas las referencias que se necesitan para ese caso. Si se necesitara alguna librería externa más, o alguna extensión más de las que ya tenemos, habría que añadirlas como referencias en el proyecto de "Algoritmos". Para añadir estas nuevas librerías tendríamos que añadir en las referencias del proyecto los dll correspondientes a las librerías necesarias. Es recomendable que si se añade alguna librería nueva se guarden los dll dentro del proyecto y no en otros lugares del ordenador para facilitar la movilidad de la aplicación de unos ordenadores a otros.