INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA EN ELECTRÓNICA

IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA MEDICIÓN DEL PULSO CARDÍACO MEDIANTE PROCESAMIENTO DE VÍDEO EN LA TARJETA DE DESARROLLO JETSON TX1

Informe de Proyecto del Graduación para optar por el título de Ingeniería en Electrónica con el grado académico de Licenciatura

Jafet Andrés Chaves Barrantes
Carné: 2013037524

Supervisado por:
Prof. Ing. Carlos Meza
Prof. Ing. Miguel Hernández
Prof. Ing. Mauricio Segura

CARTAGO, I SEMESTRE 2018

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Electrónica

Proyecto de graduación

Acta de aprobación

Defensa de Proyecto de Graduación
Requisito para optar por el título de Ingeniero en Electrónica Grado Académico de Licenciatura
Instituto Tecnológico de Costa Rica

El Tribunal Evaluador aprueba la defensa del proyecto de graduación denominado **Implementación de un prototipo para la medición del pulso cardíaco mediante procesamiento de video en la tarjeta de desarrollo Jetson TX1**, realizado por **Jafet Andrés Chaves Barrantes** y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

Ing. Miguel Hernández Rivera
Profesor lector

TEC | Tecnológico de Costa Rica
Ingeniería Electrónica

Ing. Mauricio Segura Quirós
Profesor lector

Ing. Carlos Meza Benavides
Profesor asesor

Cartago, 12 de Junio de 2018

Declaración de Autenticidad

Declaro, en honor a la verdad, que el presente documento para optar por el título de Licenciatura en Ingeniería Electrónica, aquí escrito, es de mi absoluta autoría. Para la realización del mismo he empleado conocimientos propios, también he consultado las referencias bibliográficas pertinentes y además he contado con el asesoramiento de los ingenieros encargados de la empresa RidgeRun, lugar donde se realizó el proyecto.

Por lo anterior asumo total responsabilidad sobre el contenido de este trabajo ante el Instituto Tecnológico de Costa Rica y/o terceros, en caso de cualquier falta o irregularidad que pudiera ocasionar el incumplimiento de lo declarado.

Cartago, 18 de Junio del 2018.

Jafet Andrés Chaves Barrantes
Cédula: 402270061

Resumen

El desarrollo de técnicas no invasivas para el monitoreo médico ha tenido un gran auge en años recientes gracias a displicinas científicas como la visión por computador y la inteligencia artificial. En la búsqueda de aplicaciones para el sistema embebido Jetson TX1 se plantea en la empresa *RidgeRun Engineering*, entonces, la necesidad de implementar un sistema de procesamiento de imágenes basado en el algoritmo de Magnificación Euleriana de vídeo (MEV) para la detección del pulso cardíaco. El presente documento describe el diseño y la implementación de este algoritmo para estimar el pulso cardíaco de una persona a través de imágenes de vídeo.

La aplicación se puede dividir en dos etapas principales: la implementación del algoritmo de MEV en C++, incluyendo la incorporación del GPU, y el posprocesamiento realizado sobre el vídeo con la señal de interés amplificada. En la primera etapa se plantea la arquitectura para la implementación del algoritmo y para la segunda se describe el acercamiento seguido para la estimación del pulso.

Se han realizado distintas pruebas para verificar el sistema desarrollado, que incluyen: el cálculo del índice de similitud estructural de imágenes (SSIM), la comparación cuantitativa de la estimación del pulso cardíaco con mediciones simultáneas de un oxímetro y pruebas de rendimiento. Los resultados permiten demostrar que se puede estimar, utilizando el GPU, el pulso cardíaco con porcentajes de exactitud cercanos al 95 %.

Palabras clave: GPU, Jetson TX1, magnificación euleriana de vídeo, pulso cardíaco, procesamiento digital de señales, procesamiento digital de imágenes, OpenCV, sistema en chip.

Abstract

Non-invasive techniques for vital signs monitoring have been developing in the last years due to interdisciplinary fields like the computer vision and artificial intelligence. So, in RidgeRun Engineering, there is a need for implementing an image processing system based on the Eulerian video Magnification (EVM) algorithm to detect the heart pulse in the embedded system Jetson TX1, as a way to explore its capabilities. This document describes the design and implementation of this system to detect the heart pulse of a person from video images.

The application can be split into two main stages: the implementation of the EVM algorithm in C++ and the post-processing necessary in the video with the magnified signal of interest. In the first stage, the architecture of the algorithm implementation is detailed and for the second stage, the approach to extract the heart pulse is described.

A wide variety of tests have been done to verify the implementation, this includes: the calculation of the structural similarity (SSIM), the quantitative comparison of the heart pulse with simultaneous measurements from an oximeter and performance tests. The results demonstrate that the solution, using the GPU, is capable to estimate the heart pulse with an accuracy near the 95 %.

Keywords: Digital image processing, Digital signal processing, Eulerian Video Magnification, GPU, heart pulse, Jetson TX1, OpenCV, system on chip.

A mi familia, el incondicional apoyo en mi vida

Agradecimientos

Quiero agradecer en primer lugar a mi madre, mis hermanos, mis abuelos, ellos siempre me han apoyado y mostrado el camino hacia mi superación personal y profesional en todo momento. A mi difunto padre, gracias a su esfuerzo siempre tuve las mejores oportunidades de estudio, lo cual considero como el mayor legado que me pudo haber dejado.

También quiero agradecer a los profesores del Instituto Tecnológico de Costa Rica que estuvieron presentes a lo largo de mi carrera, especialmente, por su paciencia, vocación y enseñanzas durante mi formación profesional. Asimismo a mis compañeros y amigos del TEC que estuvieron presentes en esta etapa de mi vida.

Agradezco de igual manera a la empresa RidgeRun por la gran oportunidad de desarrollar el proyecto de graduación, a todos los compañeros de trabajo que de una u otra manera mostraron su apoyo en la realización del mismo. Asimismo a los ingenieros y colegas Jose Jiménez, Michael Grüner, Carlos Agüero por toda la ayuda técnica y acompañamiento brindados.

Jafet Andrés Chaves Barrantes
Cartago, 18 de Junio del 2018

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

Introducción

1.1. Aplicaciones multimedia y sistemas embebidos

Las aplicaciones multimedia se caracterizan por requerir una gran capacidad de procesamiento debido a la complejidad y al número de operaciones de los algoritmos que se necesitan implementar. La tendencia actual en el mercado para este tipo de aplicaciones es utilizar sistemas embebidos basados en el paradigma de la computación paralela y heterogénea. A partir de estas tecnologías se ha vislumbrado la posibilidad de abrir nuevas aplicaciones en campos como la robótica, drones, redes neuronales, realidad virtual, visión artificial y en general cualquier solución que se beneficie del aprovechamiento de la paralelización de cálculos.

La computación heterogénea se le denomina a un paradigma en las ciencias de la computación donde se busca que las operaciones que se requieren durante la ejecución de una aplicación se distribuyan entre el uso de distintos procesadores especializados, entre ellos la Unidad Central de Procesamiento (CPU), la Unidad de Procesamiento Gráfico (GPU), el Procesador Digital de Señales (DSP), el Procesador de Imágenes (ISP), la Unidad de Procesamiento de Imágenes (IPU), la Matriz de Compuertas Programables (FPGA), módulos de conectividad, módulos de captura multimedia, módulos de codificación de audio/vídeo, etc [23]. De acuerdo con la tarea específica así se adecua la carga de trabajo para el procesador más apropiado.

Esta forma de cálculo computacional trae grandes beneficios como la disminución del consumo energético, eficiencia térmica y una mayor velocidad de procesamiento respecto a la utilización de un único procesador central de propósito general (CPU) ya sea multinúcleo o simple.

En este sentido, se han ampliado las posibilidades de los dispositivos móviles y los sistemas empotrados al tener en un mismo sistema en chip (*System on Chip*, SoC por sus siglas en inglés) todos estos tipos de procesadores y alcanzar la potencia de cálculo de las supercomputadoras de hace una década, pero en una plataforma de muchísima menor escala y costo [24].

Similarmente, la computación paralela se entiende como la utilización de múltiples recursos de hardware de forma concurrente. Desde la perspectiva de hardware, una computadora es un sistema paralelo sin embargo la forma más típica de desarrollar código de programación (en las aplicaciones más simples y regulares) es serializar la tarea haciendo que solo se pueda ejecutar una instrucción por pulso de reloj, empleando un único procesador. Aún en la actualidad para muchas aplicaciones de propósito general lo más conveniente es utilizar este esquema, pero, cuando se requiere un alto rendimiento y la máxima eficiencia al menor costo de potencia posible (especialmente en sistemas embebidos), lo que se necesita es dividir los procesos en bloques concretos de trabajo para ser ejecutados simultáneamente y aprovechar al máximo el paralelismo natural en la arquitectura de los procesadores modernos.

En particular, el proyecto busca aprovechar las ventajas que brindan los GPU frente a los CPU. En un inicio, los GPU se crearon para el renderizado de gráficos en 3D, lo cual es una tarea altamente paralelizable y que utiliza una gran cantidad de información. En el fondo, un GPU está compuesto de muchos procesadores simples, que individualmente se quedan atrás con un CPU en términos de capacidad de cálculo computacional, sin embargo, el potencial del GPU se encuentra en integrar un gran número de estos procesadores simples, que en conjunto son capaces de acelerar en gran medida los programas que tengan la posibilidad de ejecutarse en paralelo. En los recientes años, se ha visto el potencial de esta nueva forma de llevar a cabo los procesos de cómputo, de esta manera ha surgido el concepto de los GPGPU (*General Purpose Graphics Processing Unit*), que abarca la utilización de los GPU para muchas otras aplicaciones aparte del renderizado de gráficos, como lo pueden ser las simulaciones o el procesamiento digital de señales.

Es de gran interés para RidgeRun, por su ubicación en el mercado, dar soporte y desarrollar aplicaciones para estas tecnologías vanguardistas, puesto que, para lograr la ejecución de tareas en estos dispositivos existe detrás un serio tiempo de trabajo, habilidad y dominio del SoC para facilitar la implementación de soluciones de acuerdo con las necesidades de los consumidores y clientes. A causa de ello, la compañía es capaz de posicionarse como un referente importante en el desarrollo de sistemas embebidos tanto a nivel nacional como internacional.

La empresa RidgeRun, LLC es una organización que se enfoca en el desarrollo de productos y servicios basados en sistemas embebidos para ambientes GNU/Linux. Cabe destacar también que ha desarrollado una amplia experiencia en la adaptación de soluciones utilizando el framework de código abierto para aplicaciones multimedia llamado GStreamer. Por lo anterior, este trabajo le ha permitido formar contacto con otras compañías importantes como NXP Semiconductors, Nvidia Corporation, Xilinx, Inc., Texas Instruments, entre muchas más [1]. En sus instalaciones ubicadas en Tres Ríos, Cartago, Costa Rica, cuentan con alrededor de 30 desarrolladores, entre los cuales se encuentran ingenieros en electrónica altamente capacitados en el área de los sistemas embebidos, así como ingenieros de computación, ingenieros eléctricos y en computadores.

La gama de servicios que ofrece actualmente la empresa depende fundamentalmente de explorar nuevas aplicaciones y demostrar las capacidades para procesamiento multimedia de los sistemas en chip de última generación desarrollados por los distintos líderes de fabricación de este tipo de circuitos integrados. Otro aspecto importante, es que los desarrolladores en RidgeRun tienen una participación notable dentro de la comunidad del código abierto (*open-source* como se conoce en inglés), por lo que cada avance, tutorial, solución u aplicación que se logre implementar llega a beneficiar a otros ingenieros dentro de la misma compañía y alrededor del mundo.

Dentro de los proyectos de código abierto con los que trabaja RidgeRun, se encuentra GStreamer, el cual consiste en una plataforma o marco de trabajo multimedia (*framework*, para su significado en inglés). Este se compone tanto de bibliotecas y una API (por las siglas en inglés para *Aplication Programming Interface*) para escribir aplicaciones y complementos. Está escrito en lenguaje de programación C y entre muchas de sus funcionalidades permite reproducción de audio o vídeo, codificación y decodificación de archivos, edición no lineal, streaming en la web, entre otros. Se basa en el manejo de archivos multimedia por el principio de tubería (*pipeline*). Se ha construido para soportar múltiples sistemas operativos como: Linux, Microsoft Windows, Mac OS X, Solaris y en las más importantes arquitecturas de hardware (x86, PowerPC, ARM, MIPS) tanto de 32 como 64 bits. Esta plataforma brinda las herramientas para construir programas que permitan ecualización de audio, filtrado de vídeo, reproductores de multimedia, servidores para streaming en línea, manejo de multihilos, entre otros [2].

Particularmente, una de las plataformas de *hardware* referentes en el mercado es el kit de desarrollo Nvidia® Jetson, donde se encuentra uno de los sistemas en chip más avanzados de la industria actualmente, denominado como Tegra TX1. El mismo se compone de 256 núcleos de GPU Nvidia Maxwell, CPU ARM Cortex-A57 de 64 bits y 4 núcleos, soporte de

vídeo 4K a 60 fps en formatos de compresión H.265, VP9 y H.264 (con sus propias unidades de hardware para codificación y decodificación) y doble ISP (procesador de señales de vídeo). Asimismo, la Jetson TX1 posee 4 GB LPDDR4 (memoria RAM), 16 GB eMMC (memoria no volátil), múltiples puertos y conexiones para periféricos como HDMI, USB 2.0, USB 3.0, CSI, PCIe, 802.11ac WiFi, Bluetooth 4.0, entre muchos más [3] - [5].

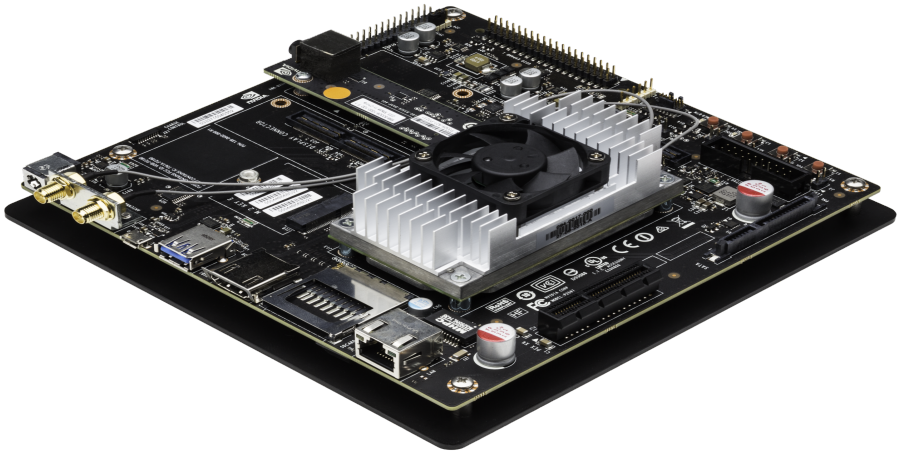

Figura 1.1: Tarjeta de desarrollo Nvidia Jetson TX1

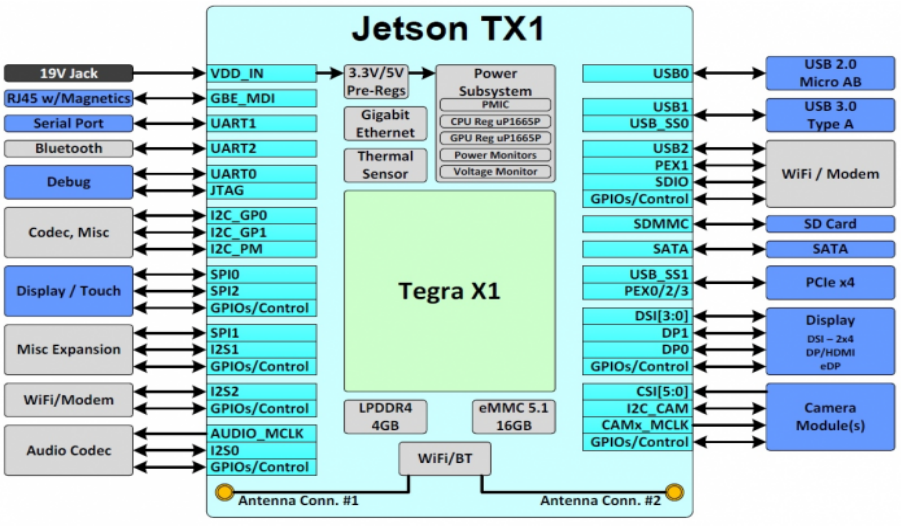

Figura 1.2: Diagrama de módulos de la Nvidia Jetson TX1

Para aprovechar al máximo todas las bondades de hardware que ofrece la Jetson TX1 (y de otros procesadores gráficos), Nvidia introdujo la plataforma para computación paralela llamada CUDA. La misma se inventó en 2006 y es una solución pionera en la computación mediante el uso de procesadores gráficos. Adicionalmente soporta los lenguajes de programación comunes como C/C++, Matlab o Python por medio de paquetes de desarrollo de software y API's. Su principal objetivo es facilitarles a los desarrolladores la posibilidad acelerar sus aplicaciones a través de las GPUs [6], [7].

El éxito que ha logrado RidgeRun mediante el trabajo conjunto con otros grandes fabricantes de semiconductores le ha permitido representar confianza y prestigio en el mercado. Es común en este ámbito laboral que, si el trabajo satisface las expectativas de una compañía como Texas Instruments, Xilinx o Nvidia, por ejemplo, implica que ellas mismas recomienden a sus diferentes clientes el uso de los paquetes de software o soluciones desarrolladas por los ingenieros en RidgeRun. Por lo anterior, mediante la constante innovación, se amplía la visibilidad de la empresa en múltiples nichos de mercado como dispositivos móviles, cámaras digitales, productos médicos, seguridad y vigilancia, entre otros. En última instancia, también se impulsa su crecimiento económico en un área tan competitiva y de gran demanda como el de los sistemas embebidos.

1.2. Contexto e importancia del proyecto

Dentro de la labor que se lleva a cabo en RidgeRun está la exploración y desarrollo de las capacidades multimedia de los productos (placas de desarrollo) de distintos fabricantes. Actualmente se trabaja con las tarjetas de la compañía de Nvidia Corporation y más específicamente con uno de sus sistemas en chip denominado Tegra X1, particularmente se ha tenido éxito con este, por lo que interesa mucho examinar su potencial como plataforma de cálculo computacional, particularmente en aplicaciones que impliquen el manejo de multimedia.

Por estos motivos en la empresa RidgeRun se contempla la oportunidad de adquirir un mayor dominio en estas tecnologías pioneras, aprovechando principalmente la experiencia en aplicaciones multimedia y sistemas empotrados. Cabe resaltar también, que el proyecto se inspira en la posibilidad de mejorar el acceso a servicios médicos a la población en general.

En el contexto actual, el uso de sistemas en chip avanzados está revolucionando muchas áreas de la industria como lo son el cuidado de la salud, robótica, automóviles o los mercados de finanzas [8]. Recientemente la utilización de procesadores gráficos ha proyectado una nueva etapa en el campo de la inteligencia artificial, llamado aprendizaje profundo. El aprendizaje profundo permite la abstracción de ideas e información a partir de algoritmos

aplicados a una multitud de datos, de esta manera se amplía en gran medida la capacidad de las empresas de predecir comportamientos en el mercado, aplicar mejoras en sus productos, ofrecer nuevos servicios, etc [9]. Concretamente, los datos pueden ser, por ejemplo, píxeles en una imagen o vectores recopilados por una red de sensores. En este sentido, el aprendizaje profundo ha mejorado disciplinas como la visión por computador y reconocimiento de señales.

En el área de la salud, el desarrollo de soluciones basadas en GPUs está transformando el modo en que se le da tratamiento médico a las personas. Por ejemplo, los algoritmos de reconocimiento de patrones en imágenes han demostrado ser más rápidos y exactos en el diagnóstico del cáncer a través de resonancias magnéticas. No se trata solo de detectar posibles riesgos a la salud, sino que también mejora la toma de decisiones de los médicos en el momento de aplicar una intervención clínica. Para cierto nivel de diagnóstico se ha visto la posibilidad de descartar la necesidad de profesionales en la salud, lo que brindaría servicios médicos de calidad a muy bajo costo. Otro gran avance que permite el aprendizaje profundo es la migración de los tratamientos médicos tradicionales a formas no invasivas para el paciente, procedimientos como las biopsias están quedando en el pasado gracias al escaneo de imágenes médicas. El aprendizaje profundo ha abierto mayores posibilidades en la investigación médica asistida por inteligencia artificial, esto ha permitido acelerar el desarrollo de nuevos medicamentos, el diagnóstico de imágenes médicas, planes de tratamiento mejor enfocados, detección temprana de enfermedades, entre mucho más [10].

1.3. Enfoque de la solución

El proyecto puede ubicarse dentro de la disciplina científica de la visión por computador o visión artificial. Asimismo, específicamente para este trabajo, se incluyen otras áreas afines a esta disciplina como las matemáticas, procesamiento de señales/imágenes, ciencias de la computación, física, análisis de vídeo y biomedicina. En la visión por computador se busca la aplicación de métodos y tecnologías para la adquisición, procesamiento, análisis y extracción de información cuantitativa o cualitativa a partir de imágenes (o vídeo) obtenidas de la realidad [11].

La base principal del proyecto yace en el algoritmo denominado como Magnificación Euleriana de Vídeo (EVM por sus siglas en inglés para *Eulerian Video Magnification*) [12]. Consiste en un método, dentro del procesamiento digital de imágenes, desarrollado por investigadores y estudiantes del Instituto Tecnológico de Massachussets (MIT, por sus siglas en inglés) para realzar o amplificar variaciones temporales en grabaciones de vídeo, especialmente aquellas que son prácticamente imperceptibles o imposibles de observar a

simple vista [13]. El ojo humano tiene una limitada capacidad espacio - temporal y en el contenido de un vídeo existe información muy valiosa que se puede extraer y analizar, aún en secuencias que parecen virtualmente estáticas.

El ojo humano se ve afectado por limitaciones biológicas que restringen la percepción visual tanto en el dominio espacial como en el temporal. Existe una gran cantidad de información en artefactos visuales que resultan imposibles de detectar a simple vista. Por ejemplo, la circulación de la sangre a través del cuerpo provoca variaciones de color en la piel, lo que puede utilizarse para revelar la periodicidad de la señal del pulso cardíaco. Desde esta perspectiva, el pulso cardíaco humano puede estimarse aplicando técnicas de análisis en frecuencia a los valores de los píxeles en las imágenes (*frames* contenidos en una señal de vídeo). Un método para permitirle al ojo ser capaz de ver tales variaciones imperceptibles es a través de la Magnificación Euleriana de Vídeo (MEV por su siglas).

Específicamente, se emplea el principio de detectar el pulso cardíaco con secuencias de vídeo a partir de variaciones en el color de la piel cada vez que la sangre es bombeada por el corazón y esta viaja alrededor del cuerpo, para ello se va a utilizar el algoritmo anteriormente descrito. La frecuencia o pulso cardíaco es uno de los signos vitales y poseer una buena salud implica que este se encuentre en un rango adecuado [14]. Adicionalmente, el ritmo cardíaco en condiciones de reposo varía de acuerdo con la edad de la persona, en general los niños con un mes de edad poseen el ritmo más acelerado (70 a 190 latidos por minutos), mientras que a partir de los diez años las personas estabilizan y bajan su ritmo cardiaco entre 60 a 100 latidos por minuto. Dentro las condiciones que pueden acelerar los latidos por minuto están: la temperatura corporal, realizar ejercicio, la posición del cuerpo o las emociones [14]. Durante un diagnóstico médico la medición del pulso cardíaco permite evaluar la presencia de enfermedades cardiovasculares, partiendo de este hecho se observa el gran potencial para desarrollar sistemas no invasivos y sin contacto con el paciente para aplicaciones clínicas.

Por otra parte, para ejecutar el algoritmo de magnificación euleriana con señales vídeo de buena resolución y en tiempos de ejecución viables para aplicaciones comerciales se necesita de una potencia de cómputo que cumpla con altas demandas, adicionalmente en [13] se afirma que el método puede ser acelerado a partir de la utilización de GPUs. Adicionalmente, dentro del trabajo actual en RidgeRun se cuenta con la plataforma computacional embebida Nvidia® Jetson TX1. Esta placa de desarrollo se puede describir como un supercomputador en un módulo de hardware, ideal para la implementación de sistemas empotrados de alto rendimiento.

La medición del pulso cardíaco puede ser realizada en forma de un autoexamen o por otras personas si se conocen los lugares del cuerpo de donde puede obtenerse y la forma de medirse [14]. Sin importar el hecho de que pulso cardíaco se puede obtener de forma muy sencilla, hasta para la técnica más simple se requiere de la intervención de una persona, esto en la realidad no es viable en el monitoreo continuo de pacientes en un hospital dado la que técnica requeriría la presencia de esa persona en todo momento, por ejemplo. Para solventar esta necesidad de obtener el pulso de una forma más automática y remota, se han desarrollado a lo largo de los años distintos dispositivos e instrumentos, entre los cuales se encuentran: pulsómetros, estetoscopios, oxímetros, marcapasos, caudalímetros magnéticos, electrocardiogramas y monitores de signos vitales.

A pesar de que existen múltiples opciones en el mercado, el proyecto busca implementar una solución alternativa con los mínimos requerimientos de hardware posibles. Además, las técnicas convencionales requieren de equipo médico que es cableado y difícil de movilizar, mientras que prototipo busca implementarse en una placa de 20x20 cm. En la figura 1.3 se muestra un esquema muy general de la solución planteada

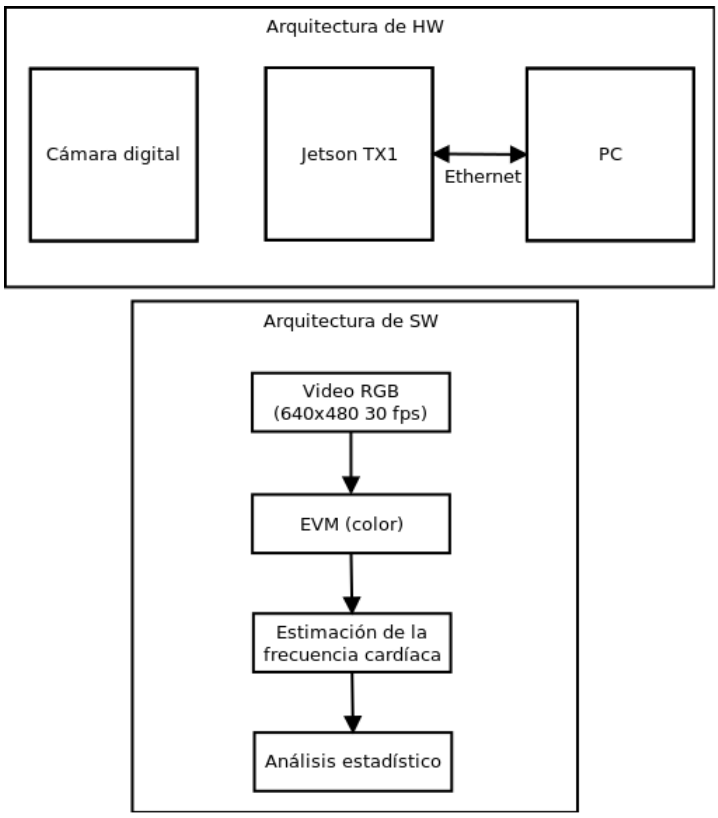

Figura 1.3: Diagrama general de la arquitectura de hardware y software.

Como se mencionó anteriormente, el proyecto se basa en gran parte en la implementación del algoritmo Magnificación Euleriana de Vídeo (a lo largo del documento se seguirá refiriendo a este como MEV) [13], explicado con mayor detalle en la sección 2.5. Este algoritmo puede utilizarse para revelar los más sutiles movimientos en estructuras como grúas o edificios, así como vibraciones en objetos. Otra gran posibilidad, es emplear este método para revelar movimientos periódicos propios de la fisiología del cuerpo humano como el movimiento de los ojos, la expansión de la caja toráxica durante la respiración, oscilaciones en la temperatura corporal, entre otros. En vista de estos avances, en los últimos años se ha desarrollado un campo extenso de investigación alrededor del estudio de los "movimientos o cambios visuales imperceptibles" expandiendo las aplicaciones en áreas como vibrometría [15], análisis de materiales, análisis estructural [16], dinámica de fluidos [17], captura remota de audio [18], así como la propuesta continua de nuevos algoritmos [19], [20].

Es importante destacar que es posible extraer el pulso cardíaco a partir de vídeo mediante otras técnicas de procesamiento. Uno de estos acercamientos se explica en [30], en esta metodología se aplica al vídeo un algoritmo de detección facial y separación ciega de fuentes a los canales de color (BSS, siglás en para *Blind Source Separation*) por el método de ICA (*Independent Component Analysis*). En este enfoque se separan fuentes de ruido en el vídeo, por ejemplo, cambios en la iluminación o el movimiento del sujeto. Se sabe además que la imagen de vídeo se compone de una mezcla de múltiples señales en los 3 canales de color RGB (rojo, verde y azul) [21].

El método ICA permite recuperar las señales fuente (ciclo cardíaco) a partir de su separación de otras señales no deseadas. Para generar la señal en los canales RGB se extrae una región de interés de las imágenes en el vídeo mediante un algoritmo de detección facial y se promedia la amplitud de color de los píxeles en la región para cada canal, estas señales se normalizan mediante el promedio y la desviación estándar.

Estas señales RGB son procesadas con el método ICA para encontrar 3 señales fuente, a nivel computacional ICA se puede implementar con el algoritmo JADE (*joint approximate diagonalization of eigenmatrices*). De las 3 señales obtenidas, normalmente la segunda componente (verde) es la que contiene la información del pulso cardíaco (señal de fotopletismografía), esta segunda componente se elige para aplicarle FFT y la componente de mayor amplitud en el espectro de frecuencia corresponde al valor del pulso cardíaco (procesamiento similar al método por magnificación euleriana) [21]. En [22] se explica un método similar a BSS, denominado PCA (Principal Component Analysis). En la figura 1.4 se ilustra el proceso del algoritmo ICA, tomado de [21].

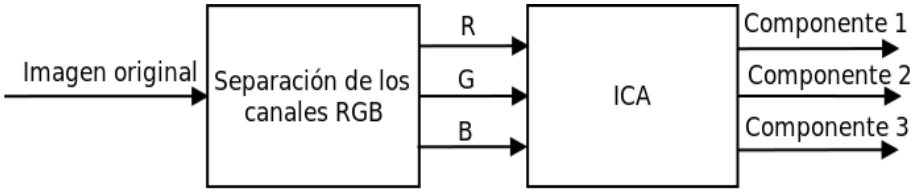

Figura 1.4: Vista general del algoritmo ICA

[21]

El acercamiento tecnológico elegido permite que la información biomédica obtenida pueda ser integrada a una plataforma portátil dentro del paradigma actual del IoT ("Internet of Things") mediante comunicación inalámbrica. Asimismo, el sistema embebido puede ser fácilmente adaptable para monitorear simultáneamente otros signos vitales como la frecuencia respiratoria o la temperatura corporal a partir de un vídeo cualquiera, utilizando en principio los mismos componentes de hardware y realizando ajustes en la arquitectura de software. Por otro lado, el uso del vídeo como fuente de información biomédica permite visualizar la posibilidad de resolver el problema de desarrollar un sistema para el monitoreo continuo de pulso cardíaco; además de ser no invasivo, de bajo costo, libre de cables, electrodos, más cómodo y libre de molestias para el paciente respecto a otras técnicas y dispositivos convencionales.

1.3.1. Objetivos y alcances del proyecto

El objetivo general del presente proyecto es el diseño e implementación de un sistema capaz de medir el pulso cardíaco del ser humano a través de una señal vídeo, incorporándolo en un sistema embebido con suficientes recursos de hardware para el tipo de procesamiento requerido y bajo consumo energético, como lo es la tarjeta Jetson TX1. El enfoque de la solución para este proyecto en particular es la utilización del algoritmo de magnificación euleriana de vídeo presentado en [13]. Primeramente se necesita la implementación del algoritmo en un lenguaje de programación, en este caso C/C++, posteriormente se adapta esta implementación para aprovechar el procesador gráfico (GPU) disponible en la tarjeta TX1. En este sentido posteriormente se busca comparar y seleccionar una de las dos implementaciones (una que hace uso del GPU y la otra que no). Seguidamente se aplica la extracción la señal de interés en el vídeo procesado a través de la definición de una región de interés, en este caso se busca la señal del pulso cardíaco. Finalmente es necesario hacer el análisis de los datos de la señal en el vídeo a través de técnicas convencionales del DSP para finalmente obtener el dato de la frecuencia cardíaca.

El sistema prototipo que implementa el algoritmo de magnificación euleriana de vídeo se planteó con las siguientes limitaciones:

- El sistema debe desarrollarse en la tarjeta de desarrollo Jetson TX1.
- El sistema procesa vídeo de resolución 640x480 a 30 fps.
- Los vídeos a procesar emplean codificación H.264.
- El procesamiento para la estimación del pulso se realiza fuera de línea, en otras palabras, la adquisición del valor del pulso cardíaco no ocurre en tiempo real.
- La implementación debe estar escrita en lenguaje de programación C++.
- El dispositivo de hardware empleado para la aceleración de la implementación es el GPU de la tarjeta TX1.
- La codificación del procesador gráfico debe realizarse con herramientas de programación basadas en CUDA.
- La comparación entre la implementación que usa exclusivamente el CPU y en la que se usa del GPU se realizará a través de los parámetros de uso de memoria, tiempo de ejecución y porcentaje de utilización.
- La configuración del procesamiento del algoritmo está definida por 4 parámetros: el nivel de las pirámides gaussianas, la frecuencia inferior y superior filtrado temporal y el factor de amplificación.

Por su parte también se requiere que el sistema cumpla con los siguientes requisitos:

- La estimación del pulso cardíaco se encuentra dentro un porcentaje de error relativo de alrededor del 5 % respecto a una medición de referencia.
- La latencia provocada por el procesamiento no es mayor a 250 ms para cada imagen del vídeo.

1.4. Estructura del documento

En el capítulo 2 se recopilan brevemente los conceptos teóricos fundamentales para entender el desarrollo y funcionamiento de la implementación del sistema en cada una de sus distintas partes. En el capítulo 3 se detalla el diseño del prototipo del sistema, así como el proceso de optimización y las limitaciones encontradas dentro de la solución seleccionada. En el capítulo 4 se presentan los resultados obtenidos a partir de las distintas pruebas realizadas a la implementación del sistema. En el capítulo 5 se muestra la discusión y análisis de los resultados. Finalmente en el capítulo 6 se rescatan las conclusiones más importantes del trabajo realizado, así como recomendaciones para los trabajos futuros.

CAPÍTULO 2

Fundamentos Teóricos

2.1. Fundamentos de procesamiento de imágenes digitales

En su concepto más general una imagen no solo se genera a partir de valores de intensidad de la luz, también se pueden generar imágenes a partir de la temperatura o las vibraciones, sin embargo, para poder ser presentadas a la vista humana estas son transformadas a alguna representación visible [25]. Desde el punto de vista matemático las imágenes digitales consisten en una matriz de dos dimensiones de valores de intensidad, los cuales poseen parámetros estadísticos localmente y que combinados definen objetos, bordes y regiones [26].

Las unidades más pequeñas de información en una imagen digital se les llama píxeles (proveniente de contracción en inglés de *picture element*), estos son los valores individuales que conforman una imagen digital. En otras palabras, una imagen digital no es más que un conjunto de muestras acomodadas en un arreglo bidimensional de datos, estas muestras son los píxeles. En este sentido un píxel es un valor limitado entre 0 y 255 donde 0 corresponde al color negro y 255 al color blanco, se le suele llamar a esta representación como la escala de grises y es la forma más común de representar una imagen pues como se señala en [25], posee las siguientes ventajas:

- Valores entre 0 y 255 se pueden representar mediante un byte, lo cual es una unidad de información muy conveniente de almacenar en una computadora.

- La gran cantidad de muestras (píxeles) compensa por el poco rango de valores que hay entre 0 y 255 (niveles de cuantización). Esto hace que valores adyacentes y cercanos en magnitud sean percibidos por el ojo como un valor intermedio.

- El ojo humano no es capaz de percibir cambios de brillo más pequeños que 1/256 (0.39 %), así que en este caso no se mejora la imagen presentada a un observador si esta tiene un mayor rango de valores para los píxeles.

La información en las imágenes se encuentra codificada en el dominio espacial, el análogo para este tipo de señales del dominio temporal. Lo anterior implica que las características en una imagen están definidas por los bordes en vez de sinusoides, esto significa que el número de muestras de un determinado espacio está definido por qué tan pequeño se necesitan las características de ser vistas, más que por las restricciones que establece el teorema del muestreo.

Lo anterior no conlleva que las imágenes no puedan presentar *aliasing*, pero este efecto es visto como una inconveniencia más que un problema [25]. A grandes rasgos mediante la figura 2.1 se puede observar como una de las más grandes consideraciones que se deben hacer a la hora de tratar con imágenes digitales, es la inmensa cantidad de información que estas contienen. La figura 2.1 ilustra esta característica de las imágenes digitales.

2.1.1. Espacios de color

Los espacios de color se refieren a cómo combinar los distintos componentes de color en un orden dado para obtener un color en particular. El espacio de color más simple de todos es la escala de grises, en donde que los únicos componentes disponibles son el blanco y el negro. La forma usual de representar estos valores de los píxeles es por medio de un rango entre 0 y 255, sin embargo, también es posible generar imágenes con colores distintos al negro, blanco o grises. Para ello se utilizan 8 bits para representar cada uno de los valores de intensidad del rojo, verde y azul, estos tres componentes establecen el espacio de color RGB. La escogencia de estos tres colores como primarios se debe a la fisiología del ojo humano. Con tres bytes para representar color se pueden generar alrededor 16.8 millones de colores diferentes [25], aún así es posible también es utilizar más bits para representar los valores de color, a este característica se le conoce como profundidad de color o bpp (*bits per pixel*). En la figura 2.2 se muestra una representación del espacio de color RGB.

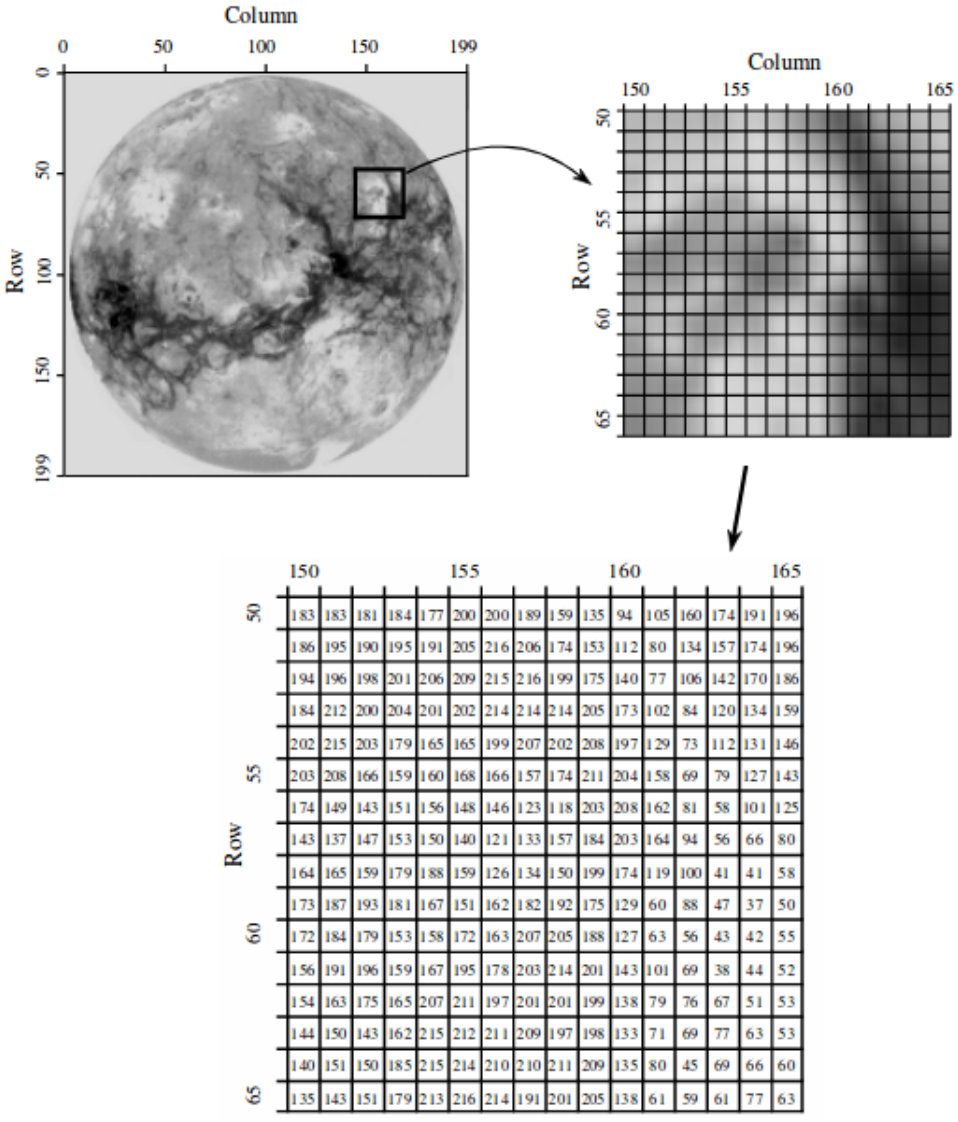

Figura 2.1: Ejemplo de la estructura de una imagen digital. Fuente [25]

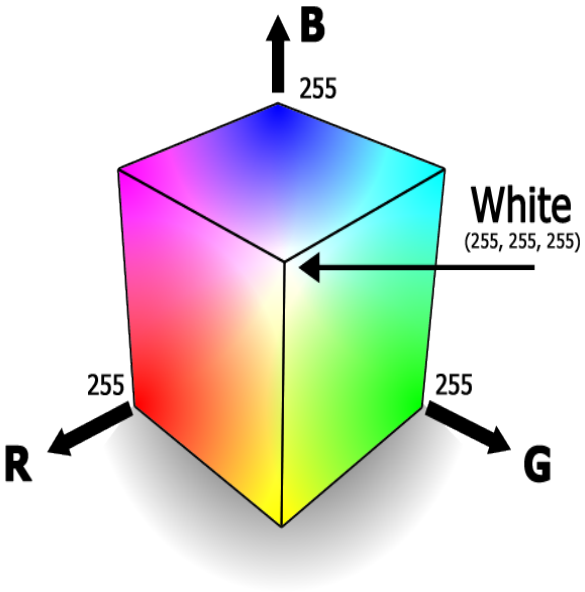

Figura 2.2: Representación del espacio de color RGB.

2.2. Filtrado de imágenes en el dominio espacial

El procesamiento espacial de imágenes ocurre en el llamado dominio espacial, el cual no es más que el plano mismo de la imagen [26]. A diferencia de otro tipo de señales como el audio, donde su información se encuentra distribuida en el dominio de la frecuencia, en imágenes, naturalmente, la información está codificada en el dominio espacial. El procesamiento espacial de imágenes puede englobarse dentro del procesamiento lineal de imágenes que abarca otras técnicas conocidas del DSP clásico, como lo son la convolución y la transformada de Fourier [25].

Precisamente, dentro el procesamiento espacial existen dos grandes ramas, que son el filtrado espacial y las transformaciones de intensidad [26]. Para este caso interesa particularmente el primer tipo, donde se trabajan las imágenes a través de sus distintos píxeles y su respectiva vecindad. A partir de este tratamiento se derivan muchas maneras de mejorar, filtrar y aplicar efectos en las imágenes como lo pueden ser las técnicas de difuminado, nitidez, detección de bordes, detección de objetos, restauración, compresión, mejoramiento, eliminación de ruido, etc.

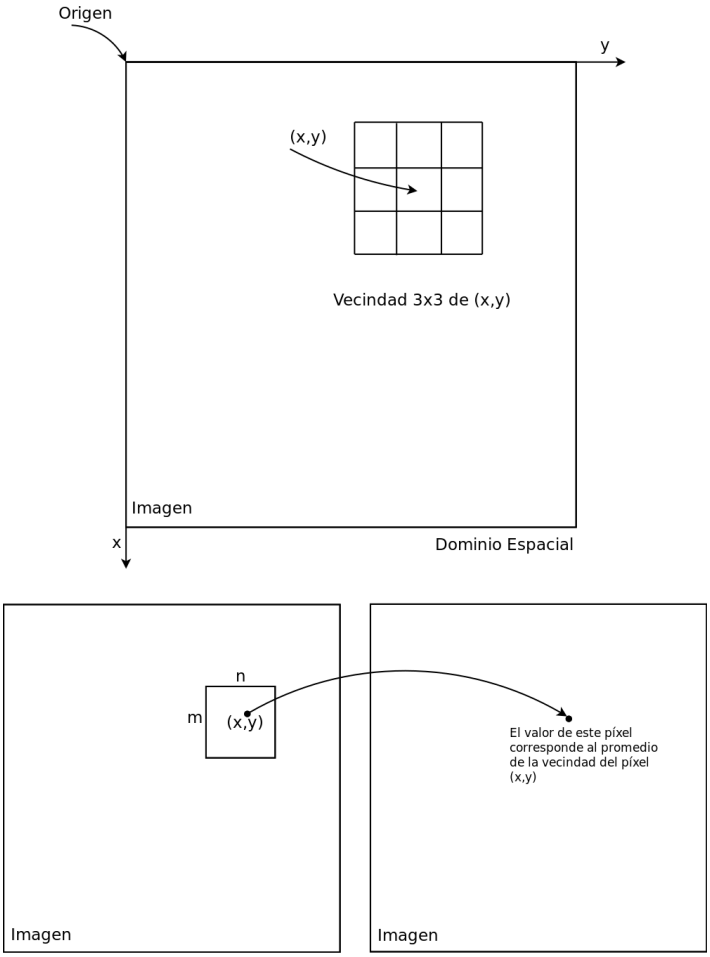

Figura 2.3: Representación del dominio espacial (arriba) y ejemplo de una operación de vecindad de píxeles (abajo). Fuente [2]

Cuando se habla del procesamiento de imágenes a través de filtrado espacial se está considerando necesariamente la manipulación directa de los píxeles en una imagen [26] por lo que un concepto básico dentro del filtrado espacial de imágenes, es de la vecindad de un píxel. En este tipo de procesamiento las operaciones dependen tanto del valor de un píxel en particular como de su vecindad o los píxeles a su alrededor. La figura 2.3 ilustra los conceptos mencionados anteriormente. De forma general una operación en el dominio espacial puede definirse como sigue [26]:

$$g(x, y) = T[f(x, y)] \tag{2.1}$$

El procesamiento que utiliza la vecindad de píxeles da como resultado un valor para el píxel en la mismas coordenadas del centro de la región denotada como S_{xy}, la cual define que esta varía de acuerdo a la posición de cada uno de los píxeles en las coordenadas xy, centrada sobre el píxel en esas coordenadas. En otras palabras, el resultado de la operación

sobre un píxel en las coordenadas xy se debe a la contribución de la vecindad del píxel, el origen de esta vecindad se mueve a la siguiente posición y así sucesivamente, el tamaño de esta vecindad o región (en imágenes digitales) es generalmente un rectángulo de 3x3 o 5x5 [26]. De forma general el filtrado está descrito por la Ec. 2.2, como se puede observar esta ecuación es similar a la definición de la correlación discreta de una señal, en el dominio de las imágenes esta operación suele llamarse correlación espacial [26]:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{2.2}$$

Esta ecuación describe que para cualquier punto de la imagen la operación consiste en una serie de sumas de productos de los coeficientes del filtros con los valores de los píxeles delimitados en la vecindad, el coeficiente $w(0,0)$ siempre está alineado con el centro de la vecindad. De forma análoga la convolución espacial se puede definir a partir de la correlación espacial (y viceversa), el procedimiento es el mismo excepto que el filtro es rotado 180 grados, de igual forma a las mecánicas de convolución y correlación de señales en 1 dimensión. En la figura 2.4 se ilustra este proceso.

Se considera que una imagen puede ser tratada como entrada a un sistema caracterizado por su respectiva respuesta al impulso, por lo tanto la imagen resultante viene dada por la convolución de la imagen entrante con la respuesta al impulso de este sistema. En la jerga del procesamiento de imágenes, suele conocerse a esta respuesta impulsional del sistema como la función de dispersión del punto o PSF (*Point Spread Function*, por sus siglas en inglés) [3]. Toda la información sobre el efecto que recibirá la imagen viene contenido en la PSF, matemáticamente esta operación se puede entender por medio de la integral de convolución en dos variables según la Ec. 2.3:

$$g(x,y) = \int_{-\infty}^{\infty} f(u,v) h(x-u, y-v) du dv \tag{2.3}$$

Del análisis de sistemas lineales se puede extender que la función $h(x,y)$ corresponde con la función de transferencia en el dominio de la frecuencia denotada como $H(u,v)$, a esta función también se le suele llamar filtro, máscara, ventana, plantilla o *kernel* (esta es la operación aplicada sobre la vecindad). Es por esta razón que se dice que los puntos que constituyen una imagen se "dispersan" a través del espacio, descrito convencionalmente por medio de las variables xy. En la figura 2.5 se muestra la operación de un filtro espacial.

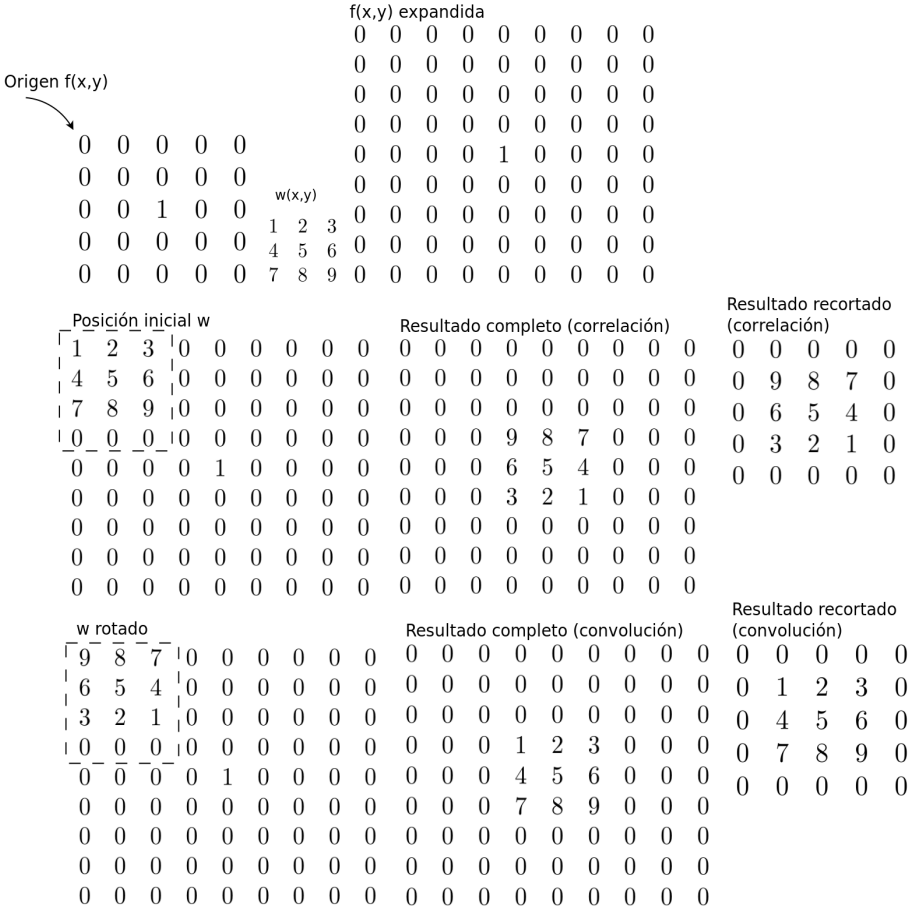

Figura 2.4: Correlación y convolución espacial discreta. Fuente [26]

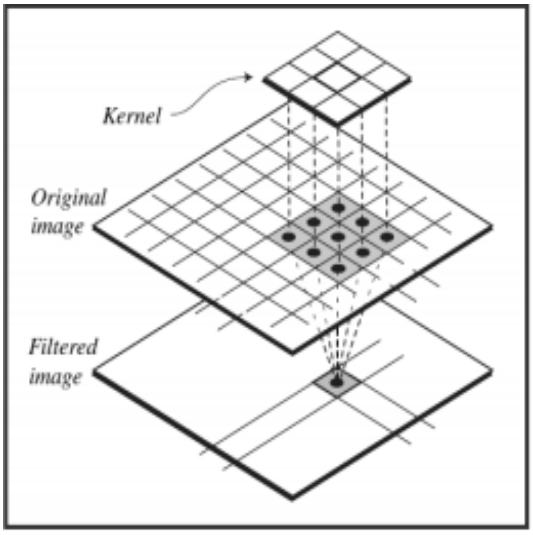

Figura 2.5: Esquema de un filtro espacial

En procesamiento de imágenes se utilizan distintas PSFs o filtros, a continuación se describen algunos de ellos, en la figura 2.6 se ilustra cada uno de ellos.

- "Pillbox": tiene una forma casi circular, con bordes rectos. Es la PSF de una lente desenfocada [25].

- Cuadrada: Todos los coeficientes poseen el mismo valor. Funciona como filtro suavizador (permite difuminar los bordes de los objetos, las transiciones agudas de intensidad se vuelven menos notables), tiene la propiedad de que no es simétrico, por lo que el difuminado en los puntos de las esquinas es diferente a las zonas verticales y horizontales.

- Gaussiano: Al igual que con el filtro “pillbox” se utiliza como un promediador (en este caso un ponderador porque sus coeficientes no son todos iguales), análogo al filtro de media móvil de señales unidimensionales. Imágenes que son procesadas por sistemas con este tipo de PSF presentan un efecto de difuminado y los bordes entre los objetos de la escena serán más difíciles de distinguir (la imagen se vuelve borrosa), los efectos de esto se ven en una mejor razón de señal - ruido o SNR [25]. Por este efecto que provocan suelen conocerse como filtros suavizadores o filtros pasabajos. Su forma viene definida por la ecuación siguiente:

$$h(x, y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{2.4}$$

- Realzador de bordes: Este tipo de filtros realizan la operación dual de los filtros suavizadores, por el contrario realzan los bordes de los objetos en una escena, suelen conocerse como filtros paso altos. Esta es la PSF que describe el procesamiento de imágenes que ocurre en la retina del ojo, viene descrito por la función delta [25].

Una muy importante aplicación de los filtros suavizadores (promediar la imagen) es lograr una representación más tosca de los objetos de interés en una escena, es decir, que los objetos pequeños se mezclan con la escena y las regiones más extensas se vuelven más homogéneas haciéndolas más sencillas de detectar [26]. Los efectos de un filtro suavizador se muestran en la figura 2.7.

Como se puede observar de la figura 2.6 la versión discreta de las PSFs viene dada por una matriz de coeficientes de distintos tamaños (3x3, 5x5, 11x11, etc), pero como es de esperar en imágenes, dada la inmensa cantidad de datos, las operaciones requeridas pueden ser muy numerosas, lo que se traduce en requerir potencias de cálculo elevadas para poder llevar

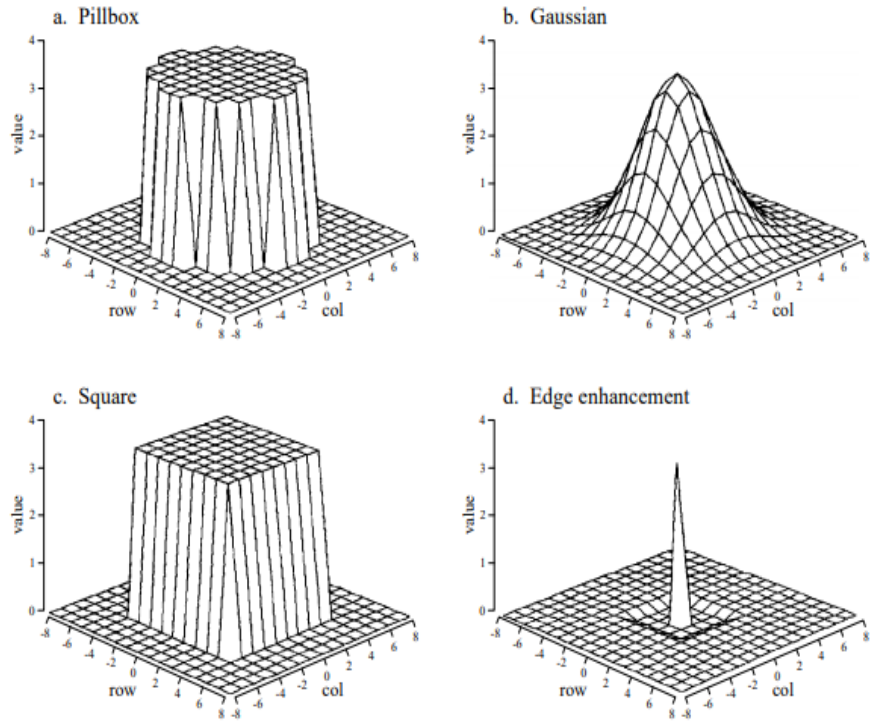

Figura 2.6: Ejemplos de funciones de dispersión del punto. Fuente [25]

a cabo este tipo de procesamiento en tiempos razonables. La capacidad de cálculo requerida es proporcional al tamaño del *kernel*, a simple vista a partir de la Ec.2.3 se ve que para cada punto de la imagen se requiere una multiplicación y una suma, lo que se podría traducir en una inmensa cantidad de operaciones, irremediablemente esto puede volver inviable este tipo de procesamiento para un *kernel* de gran tamaño, esto además es una de las razones por la que suelen preferirse filtros más pequeños de 3x3 o 5x5.

En resumen, se ha visto hasta el momento que el filtrado espacial viene definido por la vecindad del píxel y la operación que se aplica a estos píxeles en la región. La imagen resultante se va originando como resultado de la operación "moviéndose" a lo largo de la imagen original un píxel a la vez. Cabe mencionar que estos filtros pueden ser lineales o no lineales, este último es algo que no es posible realizar si se transforma la imagen al dominio de la frecuencia mediante la transformada de Fourier [26]. A continuación se explica cómo esta técnica se puede combinar para conformar las llamadas pirámides de imágenes.

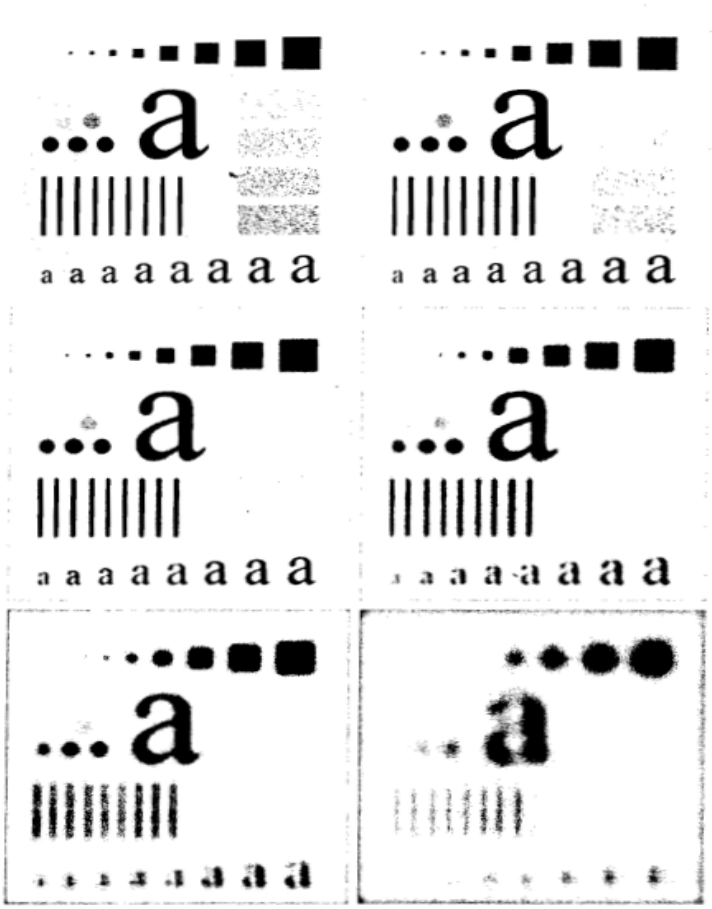

Figura 2.7: Ejemplo del efecto de un filtro suavizador. Fuente [26]

2.3. Pirámides de Imágenes

Las pirámides de imágenes son un concepto de gran utilidad en el análisis multiresolución de imágenes [26]. En el inicio de su introducción [27] fue empleado con fines de compresión y visión artificial. La construcción de pirámides es una técnica ampliamente utilizada en el procesamiento digital de imágenes, en su forma más básica se basa en alterar la resolución de una imagen (tanto aumentarla como disminuirla). Su definición se puede establecer como una colección de imágenes a distintas resoluciones. Existen distintos tipos de pirámides, entre algunas conocidas están:

- Gaussianas: Se utilizan para disminuir el tamaño de una imagen (*downsampling*).
- Laplacianas: Se utilizan para producir una imagen del tamaño original que surge de la reconstrucción de niveles superiores de la pirámide (*upsampling*), son muy conocidas por sus aplicaciones en compresión de imágenes, combinación de imágenes.
- Wavelet/QMF
- Pirámides orientables

De forma general, un sistema que produce pirámides de imágenes se ilustra en la figura 2.8. La construcción de las pirámides ocurre de manera iterativa, cuando el procesamiento no requiere de la imagen de predicción residual el filtro de interpolación y procedimiento para devolver la imagen a una resolución mayor, pueden ser omitidos [26].

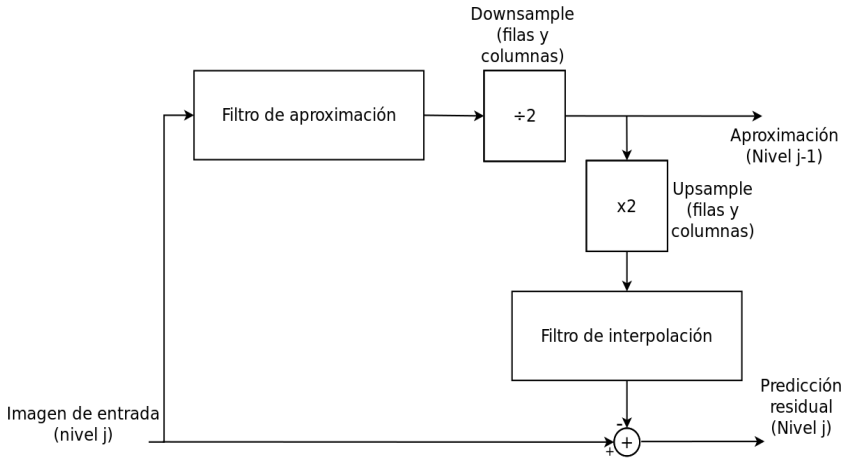

Figura 2.8: Esquema simple de un sistema para generar pirámides de imágenes. Fuente [26]

2.3.1. Pirámides Gaussianas

Las pirámides gaussianas son un caso particular donde se utilizan filtros espaciales gaussianos (suavizadores) y no se genera la imagen de predicción residual. Se trata de una descomposición espacial de la imagen de n niveles producto de la combinación de dos técnicas muy comunes del procesamiento de imágenes, el cambio de resolución y la aplicación de filtros de un filtro espacial, véase la figura 2.9.

Como resultado de operar sobre la imagen original (nivel 0) se obtiene una imagen con la mitad de la resolución (*downsampling*) y a la que se le ha aplicado un filtro de difuminado (*blurring*). Sucesivamente se opera sobre los niveles siguientes de la pirámide hasta alcanzar el nivel deseado. La promediación que ocurre con los píxeles provoca que la imagen pierda los detalles, lo que implica una pérdida irremediable de la información. Si los niveles inferiores son devueltos al tamaño original (*upsampling*) esta pérdida de información se vuelve evidente, a esta imagen suele conocerse como una aproximación o interpolación, como lo muestra la figura 2.10. Aunque esta pérdida de información en niveles de menor resolución puede considerarse inconveniente, en general, se considera una optimización de cálculo cuando se requiere el análisis de grandes regiones de la imagen y no detalles en particular. Cuando se necesita el análisis de objetos individuales se hace uso de los niveles de alta resolución.

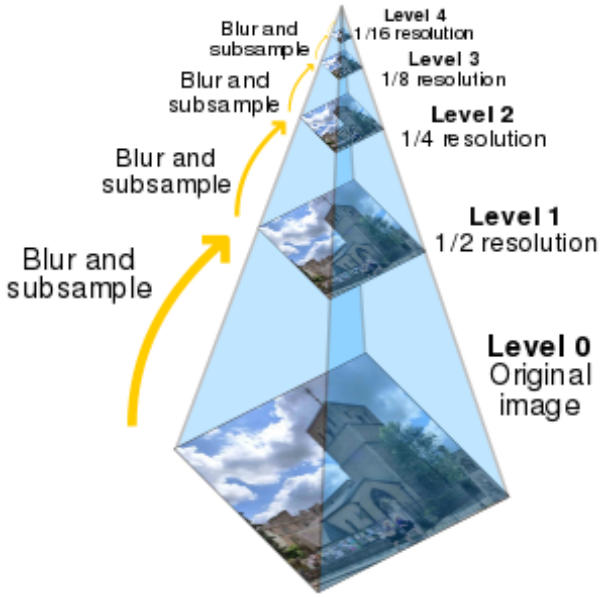

Figura 2.9: Esquema de una pirámide gaussiana

Figura 2.10: Primeros seis niveles de una pirámide gaussiana e imagen de aproximación a partir del nivel 2

El fundamento de esta técnica se remonta a [27], donde fue introducida como una manera de compactar la información en las imágenes. Esta técnica se basa en el principio de que los píxeles en una vecindad están altamente correlacionados, por lo tanto almacenar la información de una imagen a partir de los valores directos de los píxeles es ineficiente puesto que mucha de la información es redundante [27]. Primeramente se introduce el concepto de la pirámide gaussiana antes de plantear la técnica completa que nombran como pirámides laplacianas, dados sus resultados similares a operar una imagen con operadores laplacianos en mejoramiento de imágenes para una aplicación específica. La pirámide gaussiana puede verse como un paso dentro del algoritmo que conlleva la construcción de una pirámide laplaciana.

De la figura 2.11 se destaca que para generar el nivel g_{i+1} de la pirámide se aplica un filtro paso bajo (promediador), este procesamiento se realiza mediante la convolución de los píxeles en una vecindad con los coeficientes de un filtro que se generan de una familia de funciones de ponderación, una de estas funciones se asemeja a la distribución gaussiana o distribución normal, de aquí el nombre de la técnica [27]. Cada píxel de un nivel de la pirámide gaussiana se genera como el promedio ponderado de la vecindad del nivel anterior, asimismo se reduce el número de píxeles a la mitad, suponiendo que una imagen se puede acomodar como un arreglo unidimensional, en la figura 2.11 se ilustra este procedimiento.

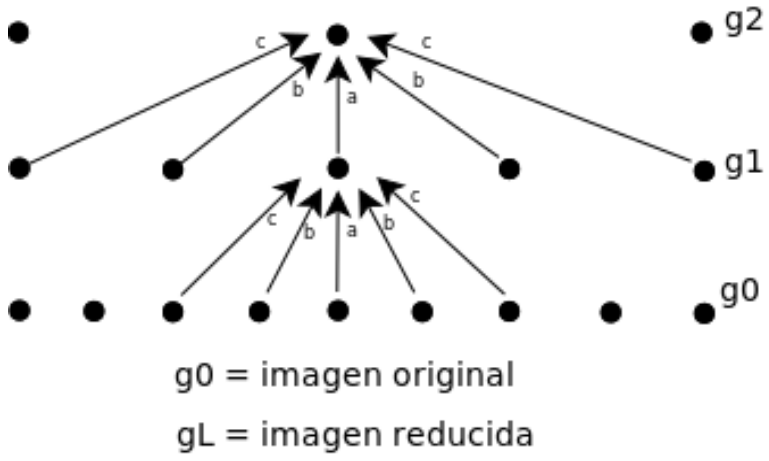

Figura 2.11: Versión unidimensional (ventana de tamaño 5) de la operación sobre los píxeles en la generación de una pirámide gaussiana. Fuente [27]

La función de ponderación que conforma al filtro ($w(n, m)$) se restringe a las siguientes propiedades [27]:

- Es separable, es decir: $w(m, n) = \hat{w}(n)\hat{w}(m)$
- Es simétrico, por lo tanto $w(i) = w(-i)$ para $i = 0, 1, 2$

- Los coeficientes poseen una contribución igualitaria, es decir que para todos los puntos en un nivel dado de la pirámide, estos contribuyen con la misma ponderación (1/4) al nivel siguiente, para el caso de un filtro de tamaño 5x5 se asume que $w(0) = a, w(1) = w(-1) = b, w(2) = w(-2) = c$, además $\sum_{m=-2}^{2} \hat{w}(m) = 1$, así las propiedades se cumplen cuando:

$$a + 2c = 2b$$
$$\hat{w}(0) = a$$
$$\hat{w}(1) = 0{,}25$$
$$\hat{w}(2) = 0{,}25 - a/2$$

La figura a 2.12 muestra las distintas formas que puede adoptar la función del filtro de acuerdo a la escogencia del parámetro a, se observa que para un valor de a = 0.4 la función adopta una forma muy similar a la distribución gaussiana [27].

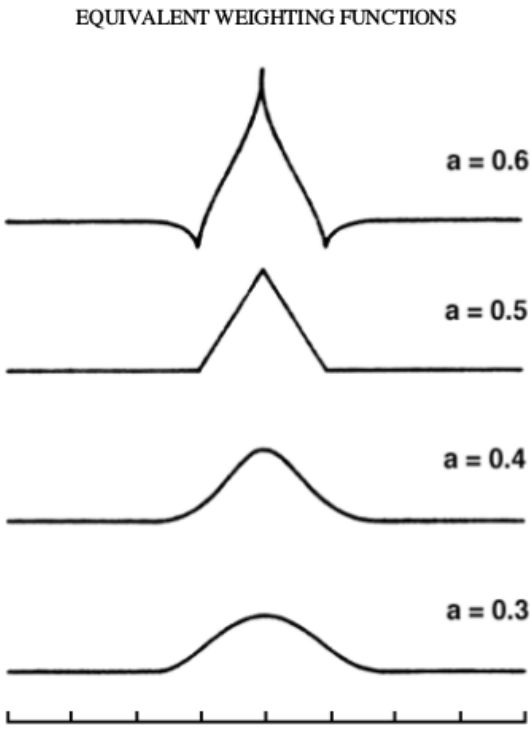

Figura 2.12: Forma de las funciones de ponderación de las pirámides. Fuente [27]

La operación inversa (expansión) de la imagen a su tamaño original requiere de la aplicación de un filtro interpolador que se puede implementar bajo distintos métodos como lo pueden ser el método del vecino más cercano, interpolación bilineal o interpolación bicúbica.

2.4. Filtrado de imágenes en el dominio de la frecuencia

2.4.1. Señales discretas en el dominio de la frecuencia

Como se mencionó anteriormente, la información de las imágenes se encuentra contenida en el dominio espacial, sin embargo, es posible realizar operaciones de transformación de dominios en las imágenes. Al realizar estas transformaciones es posible perder un gran nivel de interpretación para cierto tipo de relaciones en las imágenes pero también puede revelar otras relaciones que tal vez no eran tan directas en el dominio original (igualmente la mayoría de ellas tienen la característica de ser reversible), un ejemplo común de ello es la transformación al dominio de la frecuencia, al igual que ocurre con las señales unidimensionales.

Usualmente cuando se realiza la transformación de la imagen al dominio de la frecuencia esta resulta en una conjunto aleatorio de píxeles (las relaciones espaciales se pierden), por lo que desde el punto de vista de interpretación de la información de las imágenes, la transformación al dominio de la frecuencia no resulta útil. No obstante, la gran ventaja que presenta esta transformación es en términos de eficiencia de cálculo, dada la propiedad que la convolución corresponde a la multiplicación en el dominio de la frecuencia. La forma de implementar esta operación es a través de la FFT (*Fast Fourier Transform*), el cual es un algoritmo basado en la DFT (*Discrete Fourier Transform*) y su operación inversa, descritas en la Ec.2.5 y Ec.2.6. [28]

$$F(w) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi(nw/N)} \tag{2.5}$$

$$f(n) = \frac{1}{N} \sum_{w=0}^{N-1} F(w) e^{j2\pi(nw/N)} \tag{2.6}$$

La transformada discreta de Fourier (DFT) se puede extender en múltiples dimensiones. Específicamente en el caso de imágenes la DFT y su transformación inversa vienen definidas por la Ec. 2.7 y 2.8 [28]. Al igual que con señales unidimensionales, dado que el dominio espacial está contenida una señal discreta real se produce que el espectro de Fourier de una imagen sea periódico, lo cual a su vez implica que se mantiene una simetría tanto en la magnitud (función par) y la fase (función impar).

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N)} \tag{2.7}$$

$$f(x,y) = \frac{1}{NM} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M+vy/N)} \tag{2.8}$$

Una interpretación que se puede obtener de la ecuaciones anteriores es que el espectro F(u,v) contiene todos los valores de f(x,y) pero con la adición de los componentes exponenciales complejos. Esto implica que resulte muy complicado relacionar directamente la información de la imagen en el dominio espacial con sus respectivos valores en el dominio de la frecuencia. Sin embargo, la frecuencia está directamente relacionada con las razones de cambio por lo que los cambios en las regiones y patrones (variaciones de intensidad) de una imagen pueden asociar con las componentes espectrales. En este sentido, los valores de la imagen que describen variaciones lentas están asociadas a las frecuencias bajas, mientras que los cambios abruptos de intensidad (bordes de un objetos, límites de la imagen, ruido) se relacionan con las altas frecuencias [26]. En la figura 2.13 se ilustra un ejemplo de ello.

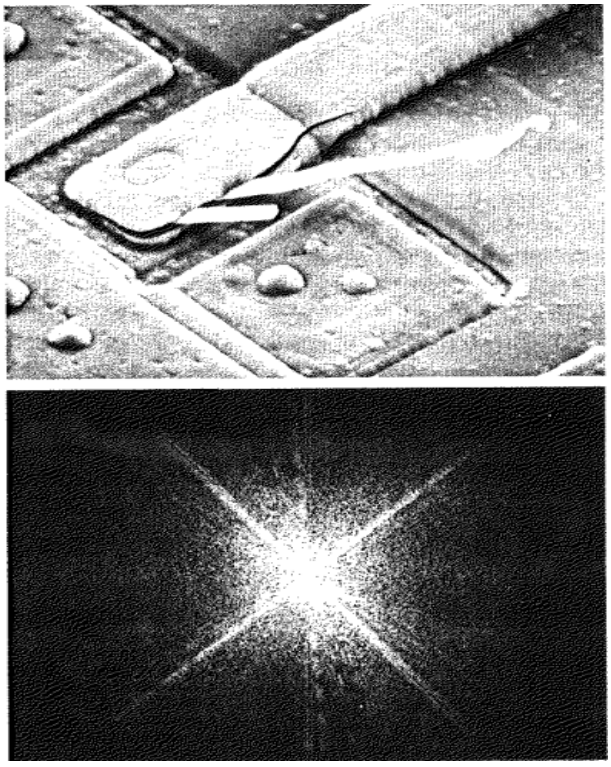

Figura 2.13: Espectro en frecuencia de una imagen en escala de grises. Fuente [26]

2.4.2. Consideraciones sobre la frecuencia de muestreo y la FFT

El algoritmo de la transformada rápida de Fourier (FFT) toma una señal discreta en el dominio del tiempo y aplica la DFT para convertir esta al dominio de la frecuencia. Mediante

la representación de esta señal transformada, se obtiene el espectro magnitud, fase y potencia. Usualmente suele definirse el gráfico de tal forma que la frecuencia corresponde al eje horizontal y el valor de magnitud, fase o potencia en el eje vertical. En la figura 2.15 se ilustra un ejemplo de ello.

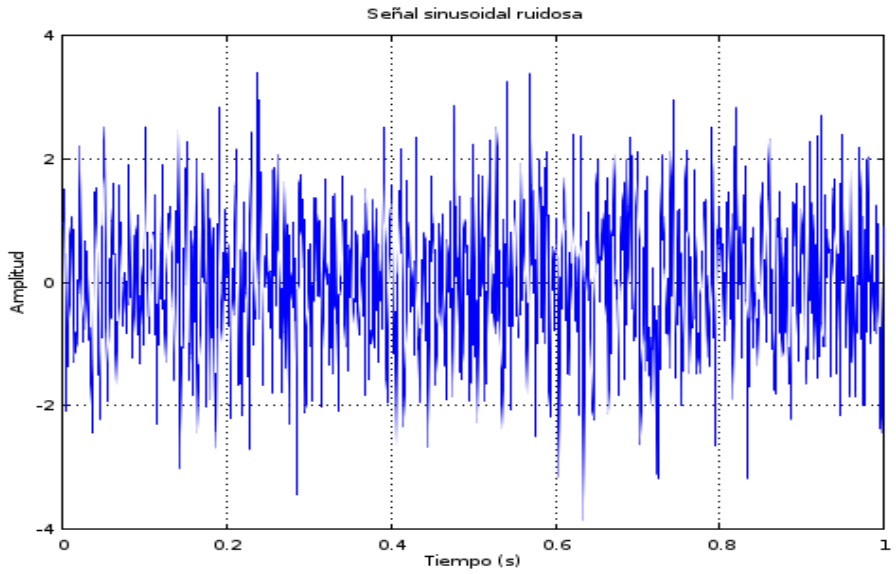

Figura 2.14: Señal sinusoidal ruidosa con frecuencia principal de 100 Hz en el dominio temporal.

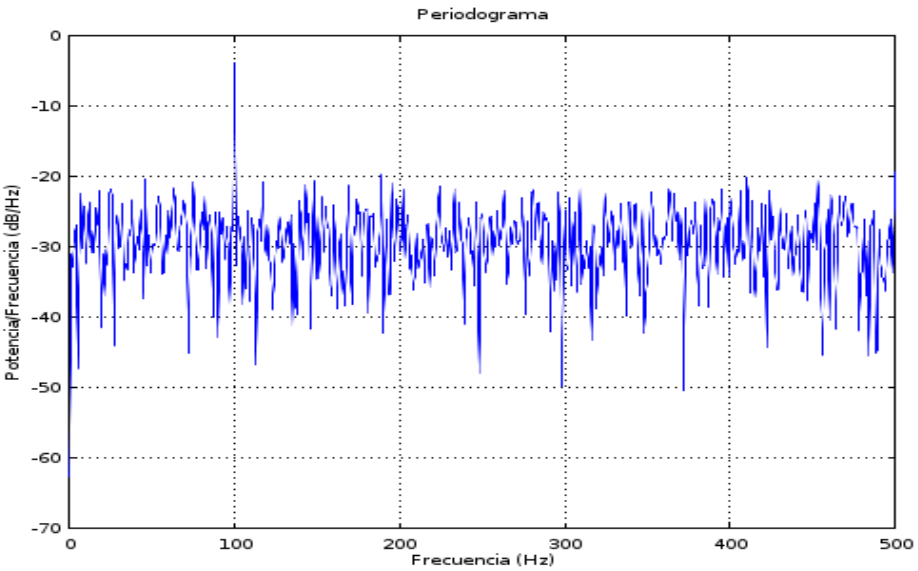

Figura 2.15: Densidad espectral de potencia de una señal sinusoidal ruidosa con frecuencia principal de 100 Hz.

Como se menciona en [31], el rango de frecuencia y la resolución del resultado de la densidad espectral de la señal está dada por la frecuencia de muestreo (F_s) y el número de muestras (N). El último punto de la gráfica de densidad está ubicado a la frecuencia dada por la Ec.2.9 y la resolución por la Ec.2.10

$$F_L = \frac{F_s}{2}\frac{F_s}{N} \tag{2.9}$$

$$\Delta f = \frac{F_s}{N} \tag{2.10}$$

2.5. Magnificación Euleriana de Vídeo

La magnificación euleriana de vídeo se basa en la descomposición espacial y el filtrado temporal para resaltar cambios sutiles en una secuencia de imágenes [13]. En el proceso se requiere la computación de pirámides gaussianas (filtrado espacial), así también posteriormente se aplica un filtrado temporal a cada banda de frecuencia espacial al considerar la intensidad de cada píxel como una serie en el tiempo, de esta manera es posible por ejemplo, se puede emplear una banda de filtrado específica para extraer solo las variaciones entre 24 a 240 pulsaciones por minuto del corazón. El filtrado temporal es homogéneo en todas las bandas espaciales, así como en todos los niveles de las pirámides de la imagen [13].

Según se señala en [13], MEV es un algoritmo que utiliza tanto filtrado espacial como temporal de vídeo. El fundamento principal del algoritmo está en amplificar variaciones temporales en una secuencia de imágenes. Todo el método asume que la imagen entera está cambiando, excepto que la frecuencia, amplitud y fase de los componentes de la escena son diferentes y que la señal de interés se encuentra allí. En este sentido, la amplificación del cambio se convierte en el realce de una banda particular, así es como se rescata la variación particular que se quiere encontrar. A continuación se describe brevemente el fundamento matemático que sienta las bases para el algoritmo de MEV. En la figura 2.16 se muestra un esquema general de este algoritmo.

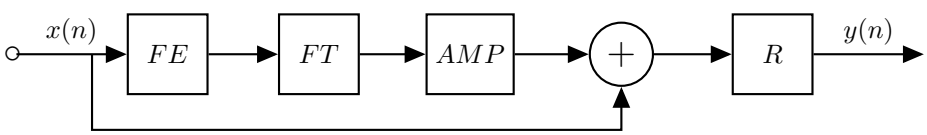

Figura 2.16: Diagrama general del algoritmo del magnificación euleriana de vídeo. [13]

2.5.1. Movimiento de primer orden

Como se indica en [13], el análisis a partir del cual se construye el concepto fundamental de MEV es través de una aproximación con la expansión en series de Taylor de primer orden.

Primero se asume el caso de una señal en una dimensión representado por la función $I(x,t)$, la cual denota una señal de la intensidad de un punto particular en la imagen en tiempo t y una posición x. Este punto describe un movimiento traslacional, por lo que puede describirse también a partir de una función de desplazamiento en el tiempo $\delta(t)$, tal que la intensidad puede escribirse según la Ec.2.11, a partir de esta ecuación se define el movimiento magnificado de la Ec.2.12, donde α representa el factor de amplificación [13].

$$I(x,t) = f(x + \delta(t)) \tag{2.11}$$

$$I(x,t) = f(x + (1+\alpha)\delta(t)) \tag{2.12}$$

Expandiendo la Ec. 2.12 a través de una serie de Taylor de primer orden se obtiene que,

$$I(x,t) = f(x) + \delta(t)\frac{d}{dx}f(x) \tag{2.13}$$

De la Ec. 2.13 se puede observar que el segundo término es el cual representa al cambio. Ahora bien, en el mejor de los casos si la señal se filtra con un sistema con una banda de paso ancha dejando sin atenuar solo el segundo término, según la Ec.2.13, posteriormente al multiplicar esta señal por α (amplificación) y añadirla luego a la señal original, se obtiene la señal de la Ec.2.14.

$$I(x,t) = f(x) + (1+\alpha)\delta(t)\frac{d}{dx}f(x) \approx f(x + (1+\alpha)\delta(t)) \tag{2.14}$$

Esta última ecuación es muy importante porque indica que es posible relacionar la amplificación de una señal filtrada en pasabanda temporalmente con la magnificación del movimiento por un factor $1 + \alpha$.

En el caso no ideal solo una parte de la señal del cambio se encuentra dentro de la banda de paso del filtro temporal, esto provoca que se den diferentes factores de atenuación (dependientes de la frecuencia) para las componentes de frecuencia de la función $\delta(t)$. Por lo anterior, se puede se puede afirmar que el filtrado pasa banda permite una aproximación de la magnificación del cambio de la señal de interés, esto conlleva a la afirmación que el algoritmo no funciona correctamente para imágenes con frecuencias espaciales altas. En otras palabras, la expansión en series de Taylor de primer orden se vuelva inexacta para tanto grandes cantidades de amplificación α y de movimiento $\delta(t)$ [13].

2.6. Similitud estructural de imágenes

La similitud de imágenes (SSIM) es un concepto que se refiere a cómo medir las diferencias entre imágenes, es específicamente útil cuando se requiere comprobar cuánta información se pierde tras la aplicación de un algoritmo de compresión o algún otro tipo de filtrado. Un procedimiento relativamente barato, en términos de cálculo computacional, es el cálculo de la razón señal a ruido de pico (PSNR), este parámetro de medición se define a partir del error cuadrático medio (MSE, Ec.2.15) entre dos imágenes. Valores típicos se ubican entre 30 dB a 50 dB, donde entre más alto el valor significa que la diferencia entre las imágenes es poca. La Ec.2.15 muestra la definición del error cuadrático medio y la Ec.2.16 el PSNR.

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |I_1(i,j) - I_2(i,j)|^2 \tag{2.15}$$

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) \tag{2.16}$$

Un método más robusto y consistente con la percepción humana es el algoritmo de similitud estructural (SSIM). El algoritmo retorna un valor entre 0 y 1 (índice de similitud) para cada canal de color, donde 1 implica que las imágenes son las mismas. Requiere más potencia de cálculo computacional dado el hecho de que requiere de un filtro suavizador gaussiano. Una diferencia notable respecto a métodos clásicos (MSE, PSNR) es que este método se basa en un modelo de la percepción que toma en cuenta la degradación de la imagen como un cambio medible en la información estructural. La información estructural es la idea de que los píxeles albergan una fuerte interdependencia con otros píxeles vecinos.

Estas dependencias son fundamentales para la detección de la estructura de los objetos en una escena visual en particular [32]. La figura 2.17 ilustra el funcionamiento de este algoritmo.

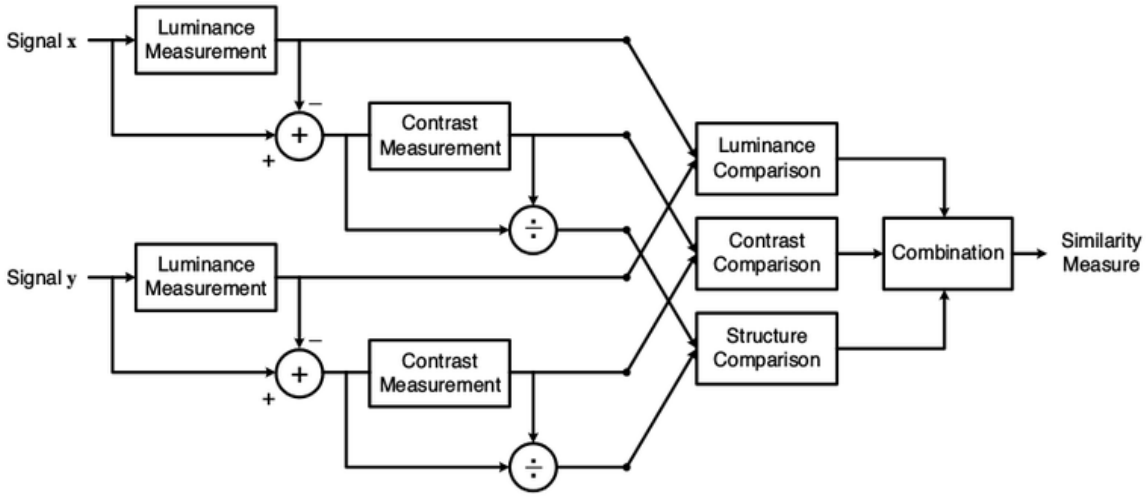

Fig. 3. Diagram of the structural similarity (SSIM) measurement system.

Figura 2.17: Esquema general del algoritmo de SSIM. Fuente [32]

La fórmula del algoritmo de SSIM se basa en la comparación de tres mediciones: la luminancia, el contraste y la estructura. De la figura 2.17 se pueden extraer las siguientes etapas del algoritmo:

1. La luminancia se obtiene como el promedio de la intensidad de los píxeles (promedio de una señal discreta, denotado como μ).

2. Se extrae la luminancia de la señal original, el resultado forma parte del cálculo de la señal de contraste.

3. La señal del contraste se calcula como la desviación estándar (raíz cuadrada de la varianza de la señal, denotado como σ), el valor normalizado a partir del contraste y la señal resultante del paso 2, se le llama la estructura.

4. Se realizan los pasos 1 al 3 para ambas señales. La similitud se obtiene como la comparación entre los valores de luminancia, contraste y estructura, según la Ec.2.17.

$$MSE = \frac{(2\mu_x\mu_y + C_1)(2\sigma_x y + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2) + C_2} \tag{2.17}$$

2.7. Open Computer Vision

Open Computer Vision (OpenCV) consiste en una biblioteca *open source* lanzada bajo la licencia BSD. Es una plataforma de software que se utiliza ampliamente para aplicaciones en el campo de la visión por computador. Nace a finales de la década de 1990 como parte de un proyecto de investigación de la compañía Intel, está escrita en lenguaje de programación C (en sus inicios) y C++ y tiene soporte para los siguientes sistemas operativos: Linux, Windows, Mac OS X. Además tiene un continuo desarrollo de interfaces para ser utilizada a través de otros lenguajes de programación como C#, Ruby, Python, entre otros. [29]

La biblioteca está ampliamente optimizada por lo que está enfocado hacia aplicaciones en tiempo real. Uno de sus objetivos principales es ofrecer una plataforma común para los desarrolladores de aplicaciones en el campo de la visión por computador, que se base en un código fácil de leer, con optimizaciones de rendimiento, portable y rápido de crear. [29]

2.7.1. Fundamentos de OpenCV

La estructura de OpenCV se basa en dividir sus diferentes funciones y tipos de objetos en módulos, a continuación algunos de los más importantes:

core: Contiene todos las definiciones de los tipos de objetos y operaciones más básicas.

imgproc: Sección de la biblioteca donde están contenidas las distintas clases para las operaciones básicas de procesamiento de imágenes, como lo son los filtros u operadores para convolución.

highgui: Un paquete de funciones muy ligero para implementar interfaces gráficas de usuario, por ejemplo mostrar imágenes o tomar entradas de usuario.

vídeo: Sección de la biblioteca que permite la lectura y escritura de vídeo.

gpu: La parte de biblioteca para incorporar el uso de GPUs soportadas por la plataforma de software CUDA que contiene las funciones optimizadas para este tipo de procesadores.

La biblioteca OpenCV contiene muchos tipos de datos que permite manejar la implementación de una aplicación en particular de la forma más conveniente, estos tipos de datos están orientados a manejar de forma intuitiva y fácil los distintos conceptos del procesamiento de imágenes, así también existen primitivas y plantillas para extender estas estructuras para las necesidades en particular. [30]

OpenCV, como una biblioteca orientada al procesamiento de imágenes se esfuerza por realizar un tratamiento de la información de la forma más intituiva posible. Para ello uno de los tipos de datos esenciales dentro de la biblioteca corresponde a la clase cv::Mat, la cual está conformada por excelencia como el pilar fundamental bajo el que se establece la biblioteca (en su más reciente implementación en C++) [30], muchas funciones en esta biblioteca reciben y retornan este tipo de dato. La clase cv::Mat se conforma como un arreglo de un número arbitrario de dimensiones y tipos de elementos. Las imágenes pueden almacenarse directamente en estos objetos lo que posibilita su tratamiento a través de la biblioteca como un arreglo bidimensional de valores, más específicamente, se enfoca en tratar las imágenes a nivel digital, como una matriz que contiene los valores de intensidad de todos los píxeles.

Un objeto de la clase cv::Mat se compone de dos partes: el encabezado (*header*) y un puntero a la matriz que contiene todos los píxeles en memoria. En el encabezado se encuentra toda la información respectiva a las dimensiones de la matriz, el tamaño, la dirección en la que se encuentran los datos, entre otros. De lo anterior se deriva la más importante característica del objeto Mat, cuando se copia una matriz (es decir al utilizar el operador = o el constructor de la clase) solo se copia el encabezado y el puntero a los datos, no los datos en sí mismos, lo que permite evitar copias innecesarias de todos los datos a través del programa que podrían ralentizar el programa o incluso detener su ejecución completamente por orden del sistema operativo, esto se logra a través de un sistema de referencia por conteo. Aún así, la biblioteca ofrece funciones específicamente para copiar los datos de los objetos Mat. Aunque sencillo, este es un concepto que resulta esencial para entender el funcionamiento de la biblioteca y su posible integración en una aplicación específica. Tómese como ejemplo el siguiente código.

```
Mat A, C;                  // Crea solo el encabezado
A = imread(argv[1],
CV_LOAD_IMAGE_COLOR);      // Lee una imagen

Mat B(A)                   // Hace una copia por medio del constructor

C = A;                     // Asigna una matriz por medio de otra
```

De acuerdo a lo que se mencionó anteriormente todos los objetos apuntan al mismo conjunto de datos. Los encabezados se tratan por separado pero cualquier modificación a los datos que apuntan es vista por los 3 objetos Mat. A primera vista esto no parece tal útil pero por el contrario sí adquiere gran importancia cuando se necesita obtener una subregión de la imagen a partir de los mismos datos pero cambiando el encabezado del objeto Mat (precisamente editando solo los límites de la matriz), la subregión tambien suele conocerse como región de interés (ROI).

CAPÍTULO 3

Diseño e implementación

3.1. Preparación del ambiente de desarrollo

El paquete de desarrollo de software Nvidia JetPack (versión 3.1) se utilizó para habilitar un entorno de desarrollo en la tarjeta TX1. El mismo puede ser descargado de la página oficial de Nvidia por medio de la suscripción al programa de desarrolladores. Este paquete ofrece todo el conjunto de herramientas para desarrollar aplicaciones con fines a la inteligencia artificial y la visión por computador en la tarjeta de evaluación Jetson TX1. En la figura 3.1 se muestra la estructura de este paquete de soporte para la tarjeta TX1.

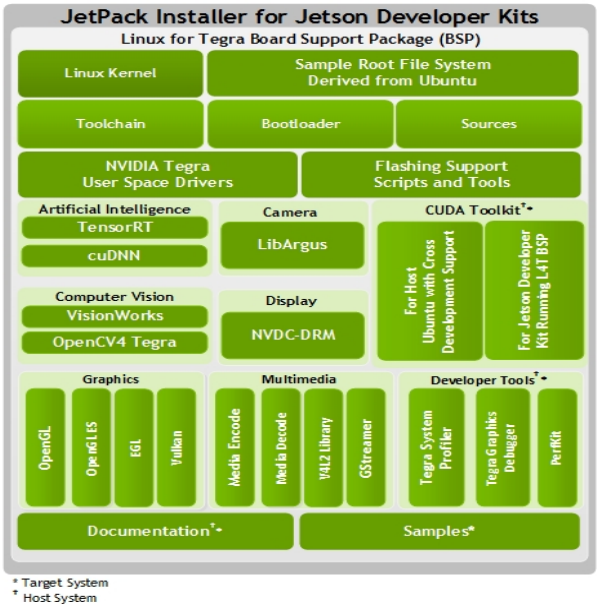

Figura 3.1: Arquitectura del paquete de soporte de la tarjeta Jetson TX1.

3.2. Análisis del funcionamiento del algoritmo de MEV

El fin del primer diseño realizado para la implementación del algoritmo fue permitir la obtención de un vídeo procesado en el que se pudiera visualizar la amplificación de la variación del color en las imágenes. El algoritmo se puede emplear con dos fines principales: la amplificación del movimiento y la amplificación de la variación color. Como se explica en [13], para el segundo caso se necesitan los siguientes pasos, según se ilustra en la figura 3.2.

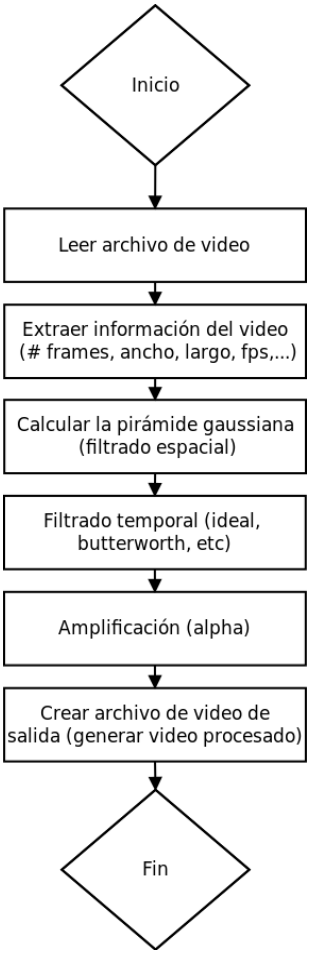

Figura 3.2: Etapas generales de la implementación del algoritmo de MEV.

Asimismo como se detalla [13], se pueden definir 4 parámetros para la configuración del algoritmo: el nivel de las pirámides gaussianas, las frecuencias de corte inferior y superior del filtrado temporal, y el factor de amplificación. Sin embargo la implementación requiere además de 3 parámetros adicionales: el nombre de los archivos de entrada y salida y el tamaño del bloque de imágenes a procesar, esta es una característica incorporada en el diseño que no forma parte del algoritmo de MEV, el cual se explica en la sección 3.2.1.

El desarrollo de la implementación del algoritmo de MEV se desarrolló enteramente en lenguaje de programación C++ y por medio de la biblioteca de código abierto OpenCV. En la figura 3.3. Este diseño corresponde a la primera implementación realizada del algoritmo. Se puede observar que este primer diseño se basa en tratar al algoritmo como un único bloque o módulo basado en una única clase que operaba sobre una entrada de vídeo y se obtenía un vídeo de salida procesado. Una desventaja de esta implementación era la gran dificultad de optimizar y comprobar individualmente cada una de las etapas del algoritmo según la figura 3.2.

Otra desventaja es la imposibilidad de cambiar la librerías sin tener que cambiar toda la implementación, dada la alta dependencia con la plataforma de OpenCV, en otras palabras, esta arquitectura no permite una implementación reutilizable a nivel de código. Sin embargo en esta etapa del diseño, con esta implementación si es posible una verificación funcional de la implementación según el método explicado en la sección 3.3.

Según el diseño de la figura 3.3 la única clase que contiene la implementación (*color_evm*) se encarga del procesado de las pirámides gaussianas, el filtrado temporal en el dominio de la frecuencia y la aplicación del factor amplificación. Para la captura y escritura de los archivos de vídeo se emplearon objetos de las clases cv::vídeoCapture y cv::vídeoWriter, respectivamente.

3.2.1. Optimización del uso de memoria

Una de las consideraciones más importantes dentro de la implementación del algoritmo de MEV es que el dispositivo de hardware empleado es la tarjeta de evaluación del sistema en chip Tegra X1, llamada Jetson TX1. Como tal, el mismo es un sistema embebido y por tanto los recursos de hardware son limitados. De las limitantes encontradas para la puesta en funcionamiento del programa es la cantidad de memoria volátil (RAM) en la Jetson TX1, específicamente se cuenta con 4 GB como se mencionó anteriormente. En una primera impresión pareciera que esta cantidad es suficiente, sin embargo, hay varios aspectos a considerar. Uno de ellos es que no toda la memoria está siempre disponible, el sistema operativo reserva parte de esta para procesos propios a través del *kernel* o puede ser que otras aplicaciones estén utilizando recursos de memoria. La forma más sencilla de revisar el uso de memoria RAM en Linux es a través de los comandos *free -m* o *watch vmstat -s*.

La implementación del algoritmo hace uso de representación numérica de los píxeles en punto flotante de 32 bits para el cálculo de las pirámides gaussianas y la aplicación del filtrado temporal por medio de la DFT, por lo que si se procesan vídeos de resolución de

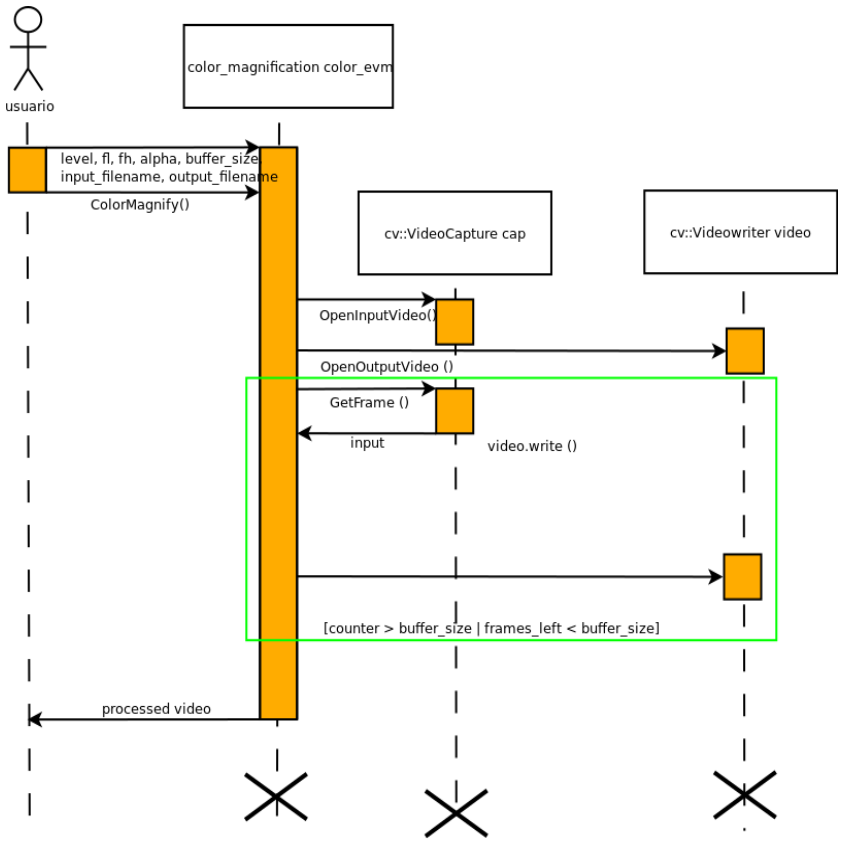

Figura 3.3: Arquitectura de la primera implementación del algoritmo de MEV.

640x480 en bloques de 100 imágenes, se emplea aproximadamente 1 GB de memoria de RAM (asumiendo que no realizan más copias en el programa).

Otro aspecto es que, dado el hecho de que se están empleando grabaciones de vídeo para aplicar el procesamiento, no se puede aplicar el algoritmo sobre archivos de este tipo de duración muy extensa, en otras palabras, la definición del algoritmo en [13] no toma en consideración la necesidad mantener en memoria las imágenes. Aquí es donde entra la aplicación de técnicas conocidas en el campo del DSP para sobrellevar esta factor limitante.

Una primera solución es tratar a las imágenes dentro del vídeo como un vector de datos cuya cantidad viene dada por la duración del vídeo y los cuadros por segundo de la grabación. Para ello se divide el vector de imágenes en bloques de tamaño definido, es decir se truncan las muestras de la señal. El tamaño que debe tener este bloque está definido por la frecuencia principal de la señal de interés y la frecuencia de muestreo del vídeo, a través del teorema de muestreo de Nyquist-Shannon. Los efectos de este truncamiento o enventanado (*windowing*) de las muestras depende del tipo de función de ventana utilizada. Para el diseño presentado, se empleó una ventana rectangular.

3.3. Verificación de la implementación del algoritmo de MEV

Tras obtener el vídeo procesado, los resultados de la implementación se verificaron a través del cálculo de un parámetro cuantitativo, un método que permite comprobar los resultados más allá de una comparación visual entre los vídeos. El mecanismo utilizado fue el índice de similitud estructural de imágenes explicado en la sección 2.6. Este algoritmo se implementó a través de la biblioteca OpenCV, toma dos vídeos, los compara cuadro por cuadro y permite calcular el índice de similitud (SSIM) para cada plano de color. Por facilidad de lectura de los resultados, el índice se convirtió a porcentaje, así por ejemplo, entre más cercano a 1 (100 %) menor diferencia existe entre el vídeo de referencia y el vídeo procesado. Como vídeo de referencia se tomó el vídeo procesado a través del programa escrito en Matlab, que está disponible en la página oficial del laboratorio (CSAIL) que publicó el artículo sobre el algoritmo de magnificación euleriana de vídeo [12].

3.4. Modularización de la implementación del algoritmo de MEV

El rediseño de la arquitectura de la sección 3.2 se enfocó en la aplicación de los principios SOLID de la programación orientada a objetos para lograr una arquitectura más estructurada. De acuerdo con la figura 2.16, se puede modularizar el algoritmo en etapas o bloques independientes. Cabe destacar que para la segunda implementación se empleó la misma solución de código obtenida en la implementación del primer diseño, sin embargo este código se distribuyó en clases independientes, de tal forma que la implementación del algoritmo como se logró primeramente requiere de la instanciación y comunicación de estos módulos de una manera lógica, de acuerdo con las figuras 3.4 y 3.5.

El hecho de basar la implementación del algoritmo en el concepto de la arquitectura *clean* permite hacer que el código orientado a ejecutarse en el procesador gráfico (GPU) pueda ser tratado como otros módulos. Así, por ejemplo, se puede desacoplar de la implementación la sección del código para la construcción de las pirámides en el CPU y acoplar la ejecución de este proceso en el GPU. En este sentido, la aceleración del algoritmo a través de la utilización de un coprocesador al CPU, se puede lograr a través de la división de las secciones de código que se deben ejecutar específicamente en el GPU. Todo lo anterior se puede lograr aún cuando la implementación de los módulos utilizan bibliotecas distintas para el procesamiento de imágenes. Esto se logró a través de la aplicación de técnicas como el polimorfismo y herencia. En las secciones siguientes se detallan los aspectos del diseño de esta arquitectura 3.4.7.

Cada uno de los módulos está dividido de acuerdo con cada una de las etapas del algoritmo estas partes del rediseño se describirán a continuación de forma individual.

3.4.1. Adquisición de las imágenes a partir del vídeo

Para la captura de los cuadros contenidos se utilizan los objetos de la clase *vídeoCapture* de la biblioteca OpenCV. A través de este objeto se realiza la decodificación a partir del vídeo en un archivo de formato mp4 y posteriormente extracción de los datos de los píxeles. Los datos de los píxeles en el espacio de color RGB para cada imagen son colocados en una matriz con los tres canales de imagen. Como se menciona en la sección 2.7, la colocación del valor de los píxeles en una matriz se puede lograr a través de la utilización de los objetos *cv::Mat*. Este tipo de objeto es la piedra angular de la implementación del algoritmo a través de la biblioteca OpenCV puesto que todos los módulos emplean esta estructura de datos en su procesamiento.

3.4.2. Filtrado espacial: Construcción de las pirámides

El primer paso del procesamiento es aplicar la descomposición en una pirámide de nivel n para cada cuadro contenido en el vídeo de entrada. Como se señala en [13], para la amplificación de color es necesario calcular la pirámide gaussianas de las imágenes, aunque en el caso más general del algoritmo (amplificación de movimiento) se necesitan las pirámides laplacianas que se construyen a partir de las mencionadas primeramente, como se explica en la sección 2.3. En OpenCV el proceso de cálculo de las pirámides gaussianas se puede hacer a través de la convolución espacial (ver sección 2.2) de las imágenes con el filtro de la Ec.3.1, el cual corresponde a un filtro binomial de tamaño 5. Por último se reduce cada dimensión de la imagen a la mitad del tamaño original.

El objetivo de este procesamiento es la reducción de ruido de la señal y además facilita la extracción de la señal puesto que ayuda a la eliminación de frecuencias altas en la imagen, este efecto se aprecia como un suavizado en los bordes de la imagen. El último nivel de la pirámide de la imagen es acumulado en un vector (de acuerdo al tamaño configurado de bloque) antes de proseguir con el cálculo en las siguientes etapas. La figura 3.6 muestra el procesamiento requerido para procesar la pirámide gaussianas, el mismo debe repetirse para todas las imágenes de vídeo entrantes como se ve en la figura 3.4.

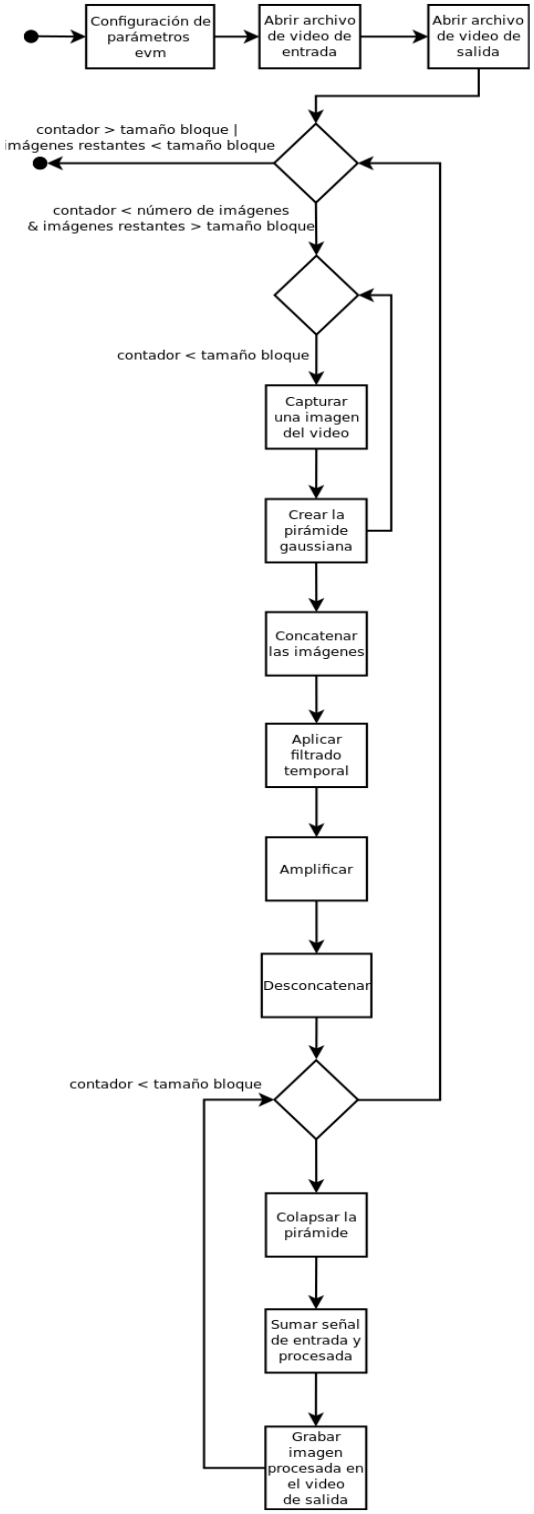

Figura 3.4: Diagramas de estados de la implementación del algoritmo de magnificación euleriana de vídeo.

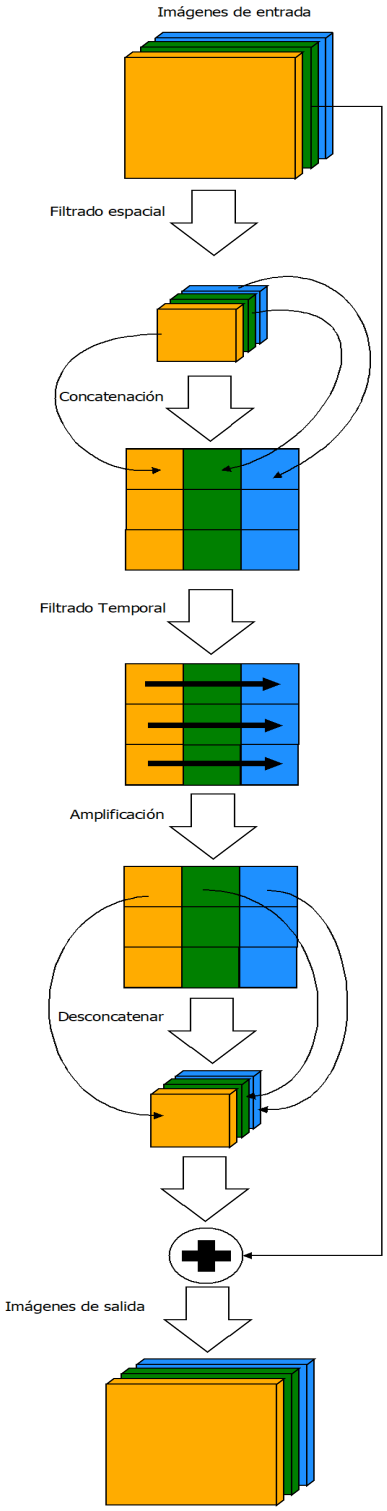

Figura 3.5: Diagrama del procesamiento en las imágenes.

$$H = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{3.1}$$

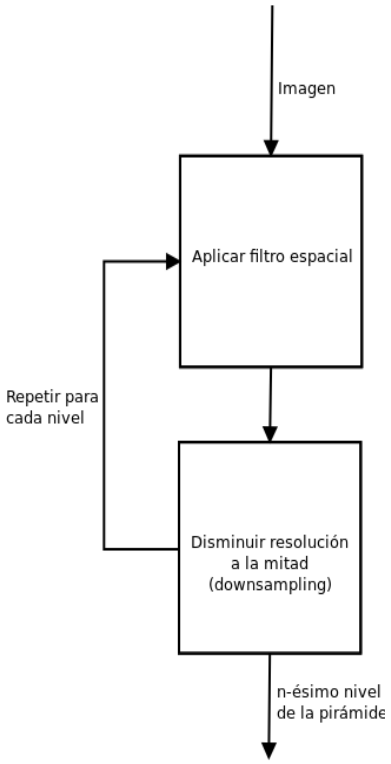

Figura 3.6: Procesamiento para la construcción de la pirámide.

3.4.3. Filtrado temporal: Extracción de la señal de interés

Previo a la aplicación del filtrado temporal, cada imagen del vector que compone al bloque se reacomoda, como se observa en la figura 3.5, en una columna de matriz única M. Esto se realiza por motivos de optimización de operaciones en el procesador y conveniencia de procesamiento durante la aplicación de la transformada discreta de Fourier en las imágenes (ver sección 2.4). Este reacomodo de las imágenes también se da debido a que en esta etapa del algoritmo interesa interpretar al píxel en la posición (x, y) de la imagen 0 como parte de un vector unidimensional de datos conformado por los otros píxeles en la posición (x, y) de la imagen 1 y así con todo el resto de imágenes sucesivas. Esta única matriz tiene tamaño $n \times m$, donde m es el tamaño del bloque configurado y n es el número de píxeles del último nivel de la pirámide gaussiana.

Después de la separación de la imagen en distintas bandas de frecuencia espacial (niveles de la pirámide), se utiliza el último nivel de la pirámide o la imagen de menor resolución calculada y a la misma se le aplica un filtrado temporal paso banda. En [13] se menciona que para el caso de la amplificación de la variación de color requiere un filtro de banda angosta, ya sea un filtro ideal o filtros IIR de orden bajo con transiciones agudas en las frecuencias de corte. Dada que la señal de interés es la frecuencia del pulso cardíaco, se puede utilizar una banda de filtrado ubicada dentro del rango de 0.4 Hz a 4 Hz (24 - 240 bpm), en el caso de la implementación, esta banda es ajustable a través de las frecuencia de corte inferior y superior.

En este sentido, el realce del cambio resulta en la amplificación de una banda de frecuencia de particular interés. En general se puede afirmar que la selección del filtro en esta etapa está sujeto a la aplicación, por lo que el algoritmo y la implementación tienen esa flexibilidad.

A nivel de implementación el primer paso es separar la imagen en sus tres canales del espacio de color RGB, dado que la función que aplica la DFT en la imagen para un canal individual. Para resolver esta limitación en la biblioteca el mismo procesamiento en el dominio de la frecuencia es iterado 3 veces sobre canal de color, después cada uno de estos canales son recombinados después de aplicar la IDFT.

En esta etapa antes de aplicar la DFT, la matriz de imágenes reacomodadas es expandida hasta un tamaño óptimo por cuestiones de eficiencia de cálculo. Este tamaño se obtiene mediante el cálculo de la potencia de 2 más cercana para cada unas de las dimensiones de la imagen (largo y ancho). La DFT es aplicada en cada fila de forma independiente, al igual que el filtrado y la IDFT. El filtrado ocurre en el dominio de la frecuencia por medio de una máscara que se construye a partir de las frecuencias de corte inferior y superior definidas al inicio de la ejecución, por lo tanto la convolución en el dominio temporal sucede a una multiplicación en el dominio de la frecuencia. El filtro aplicado fue la ventana rectangular, cuyos límites están definidos por la Ec.3.2, L corresponde al número de muestras tomadas, en este caso, el número total de imágenes procesadas y F_s corresponde con los cuadros por segundo del flujo de vídeo. En la figura 3.7 se muestra este procesamiento, esta etapa se ejecuta una sola vez por bloque de imágenes.

$$f' = \frac{2 * f * L}{F_s} \tag{3.2}$$

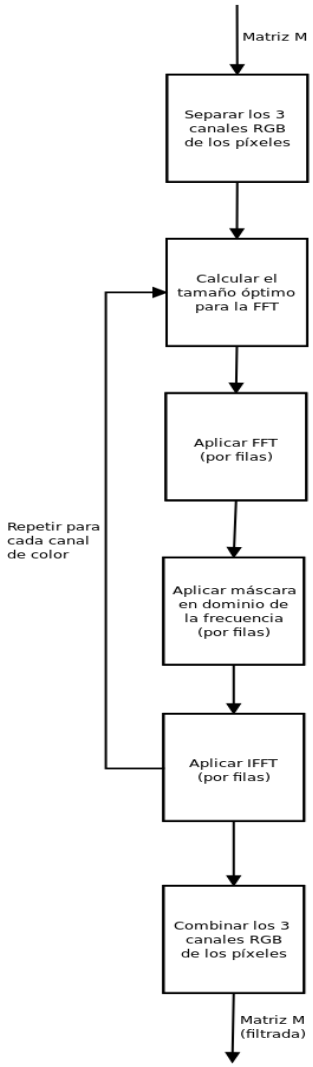

Figura 3.7: Procesamiento para el filtrado de las imágenes en el dominio de la frecuencia.

3.4.4. Amplificación

A partir del proceso de la figura 3.5, se tiene que en esta etapa se aplica el factor de amplificación la matriz M. El factor de amplificación debe seleccionarse de acuerdo al vídeo de entrada o la aplicación al igual a como sucede con las frecuencias de corte. Este factor de amplificación responde a la definición de la Ec.2.12. A nivel de implementación, en el caso de amplificación de color, el procesamiento de esta etapa es simple puesto que solo implica la multiplicación de todos los valores filtrados de matriz M con un valor constante, este parámetro se ajusta al inicio de la ejecución del algoritmo.

3.4.5. Reconstrucción

Como se observa de la figura 3.5, se necesita realizar la operación inversa realizada antes del filtrado filtrado temporal. En el proceso de esta etapa se organizan nuevamente las

imágenes a partir de los píxeles contenidos en cada columna de la matriz M. A estas imágenes se les aplica la reconstrucción de la pirámide gaussiana, para ello primero se duplica el tamaño de la imagen colocando ceros en las filas y las columnas, después se aplica la convolución espacial de la imagen con el filtro de la Ec.3.3. Al final de este etapa se tiene cada imagen del vídeo filtrada y del tamaño original. Finalmente para generar la imagen que se grabará en el vídeo de salida, la imagen filtrada es sumada a la imagen original. Esta etapa se ilustra en la figura 3.8.

$$H = \frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{3.3}$$

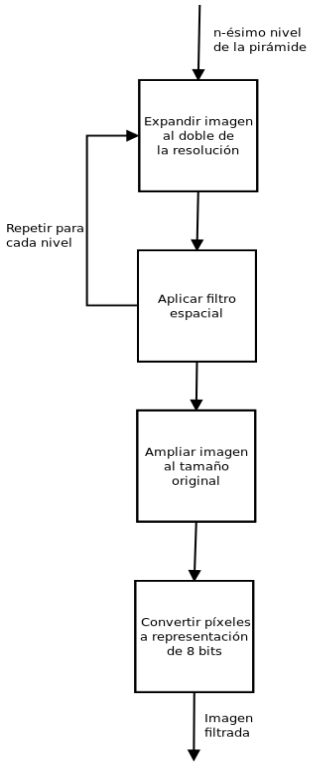

Figura 3.8: Procesamiento para la reconstrucción de las imágenes.

3.4.6. Escritura de las imágenes en un archivo de vídeo

Esta etapa no es necesariamente parte del algoritmo de MEV, pero sí implementó con el fin de visualizar las imágenes procesadas. Para la escritura de las imágenes procesadas se genera un archivo de vídeo a través de la clase vídeoCapture de la biblioteca OpenCV. Esta

clase permite la codificación del archivo de vídeo en diferentes formatos, para este caso el programa utiliza la extensión .avi.

3.4.7. Arquitectura de la implementación modularizada

El programa que implementa el algoritmo de magnificación euleriana de vídeo está inspirado en el concepto de la arquitectura *clean* y los principios SOLID para la programación orientada a objetos. El uso de los principios de programación SOLID es un requisito de los proyecto de código abierto de la empresa RidgeRun. Entre muchas de las ventajas de la aplicación de estos principios, principalmente está la facilidad de implementar pruebas unitaria de código, además el código se vuelve automatizable, fácil de documentar, reutilizable y en general adquiere un grado de más profesionalidad. En la figura siguiente se ilustra la arquitectura diseñada.

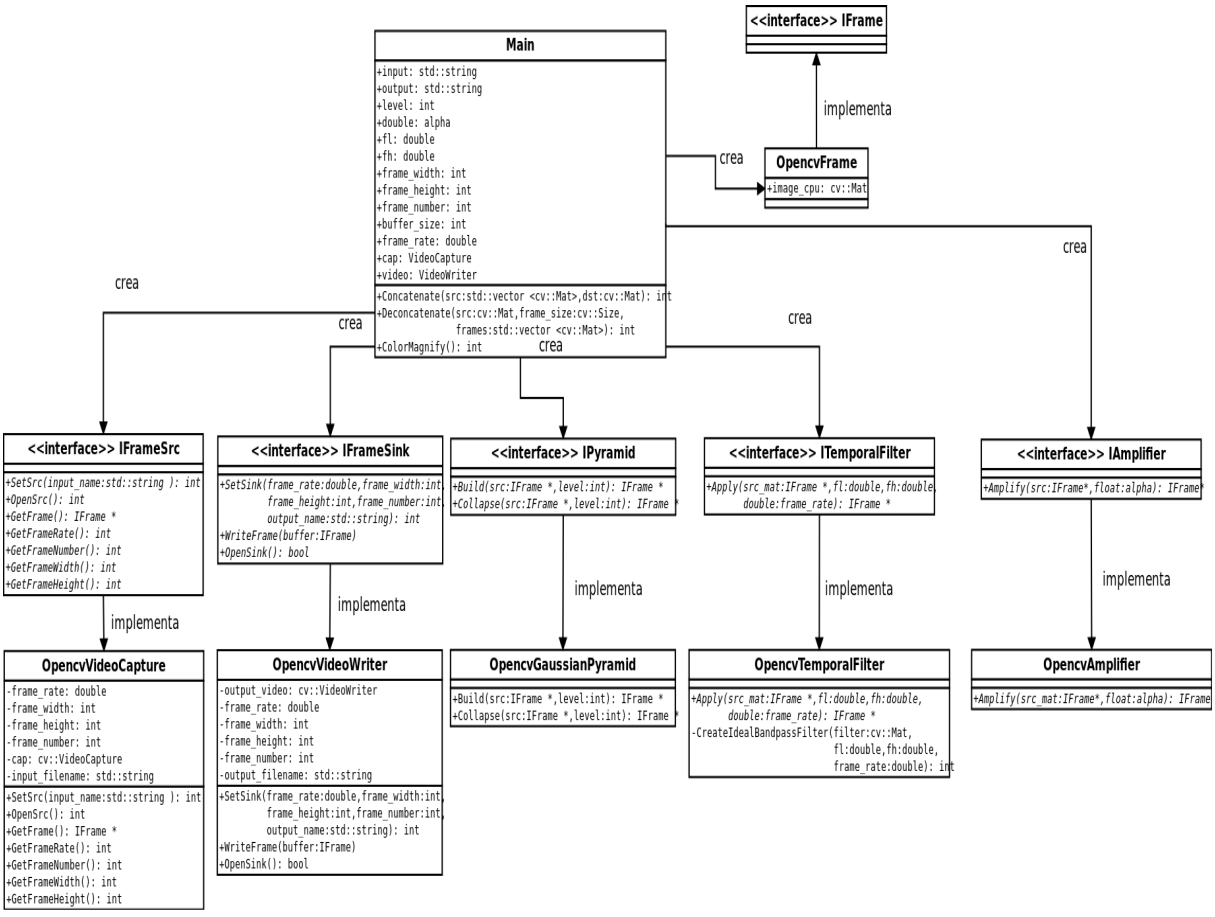

Figura 3.9: Diagramas de clases de la implementación del algoritmo de magnificación euleriana de vídeo (CPU).

Según la figura 3.9 la arquitectura está dividida en clases o módulos, cada uno de ellos se divide de tal forma que siguen cada uno de los pasos necesarios para aplicar el algoritmo de magnificación euleriana de vídeo. La construcción del programa se da a partir de la utilización de interfaces que permiten acoplar una implementación particular de cada uno de estos módulos (independiente de los modos de captura/escritura del vídeo o de bibliotecas). En el presente caso se empleó en la implementación la biblioteca OpenCV. A continuación se describen cada uno de los módulos:

- OpencvvídeoCapture: Permite la extracción de las imágenes a partir de un archivo de vídeo. Se encarga de verificar la existencia del archivo de vídeo, obtener la información sobre las características del vídeo (resolución, cuadros por segundo), la apertura y decodificación del archivo y finalmente la incorporación de las imágenes, más precisamente los valores de sus píxeles, en una estructura de datos (matriz) en el programa.
- OpencvvídeoWrite: Permite la creación de un archivo de vídeo a partir del procesamiento sobre el archivo de vídeo original.
- OpencvGaussianPyramid: Construye las pirámides gaussianas de cada imagen en el vídeo.
- OpencvTemporalFilter: Esta clase aplica el filtrado temporal sobre la colección de imágenes contenidas en las pirámides. Se encarga de calcular la máscara de filtrado y aplicarla en el dominio de la frecuencia por medio de una multiplicación con la señal de entrada y la respuesta en frecuencia del filtro.
- OpencvAmplifier: Aplica la multiplicación de una constante (α de la Ec.2.12) sobre las imágenes filtradas temporalmente.
- OpencvAdder: Se encarga de realizar la operación de adición sobre la señal original y la señal procesada.

3.5. Implementación en el GPU

Una forma de extender el funcionamiento del programa para que pueda hacer uso de coprocesadores externos al CPU es mediante la plataforma de desarrollo de software llamada CUDA. A través de las extensiones en los lenguajes C y C++ se pueden ejecutar funciones concretas en los GPU que estén soportados por esta plataforma. La tarjeta de evaluación Jetson TX1 cuenta con uno de estos GPUs dentro del *System on Chip* (SoC).

Las funciones que se ejecutan en el GPU se procesan a través de una multitud de hilos en paralelo dentro de los numerosos procesadores (núcleos) del GPU. En otras palabras CUDA se puede interpretar como una capa de acceso a un conjunto de instrucciones y elementos de computación paralela orientados a ser ejecutados en el GPU. En el contexto de CUDA se le suele llamar *kernel* a estas funciones específicas para ser ejecutadas en el GPU.

Un aspecto importante es que en la arquitectura del SoC TX1 aplica del principio de interacción heterogénea en el CPU y el GPU. La figura 3.10 ilustra este concepto. Como se puede observar de esta figura, la interacción entre el CPU y el GPU, requiere de la transferencia de datos entre cada uno de los procesadores, usualmente la colocación de los datos en cada uno de los espacios de memoria de los procesadores requiere de dos operaciones conocidas como subir y descargar, subir se refiere al proceso de reservar y colocar el espacio de memoria de los datos del CPU al GPU, mientras que descargar se refiere al proceso inverso. Es ampliamente conocido que estas transferencias de datos en memoria tienen un alto costo en el rendimiento del programa en términos del tiempo de ejecución y la latencia que implica el procesamiento del algoritmo, por lo que deben limitarse estas transferencias durante la ejecución del programa.

Las secciones de la implementación del algoritmo orientadas a ejecutarse en el GPU fueron implementados a través de la biblioteca OpenCV. Esta biblioteca posee la extensión gpu (en la versión 2.4), la cual permite la utilización de las funciones de CUDA por medio de la capa de alto nivel que conforma OpenCV. La incorporación del GPU en las etapas del algoritmo implica incorporar el módulo para el GPU dentro del flujo de procesamiento mostrado en la figura 3.4. Previo a la transferencia de datos entre el anfitrión (CPU) y el dispositivo (GPU) y ejecución es necesaria la reservación del espacio de memoria, la mejor forma de lograr esto a través de la biblioteca OpenCV es por medio del objeto *cv::gpu::GpuMat*. Este objeto permite el almacenamiento de memoria del GPU, utiliza la misma interfaz que su análogo el objeto *cv::Mat*, con algunas limitaciones. Para la realización de la sección del programa a ejecutarse en el procesador gráfico se adaptaron las principales etapas de procesamiento (filtrado espacial, temporal y amplificación) a través de sus funciones equivalentes dentro del módulo gpu de la biblioteca OpenCV.

Como se mencionó anteriormente, la arquitectura basada en módulos permite incorporar una nueva implementación de cada una de las etapas del algoritmo, sea que esta utilice otra biblioteca o implique la incorporación de otros dispositivos de hardware. En este sentido la implementación que emplea al GPU se implementó a partir de las mismas interfaces de la figura 3.9. Los únicos módulos que no emplean el GPU son la etapa de lectura y escritura de las imágenes.

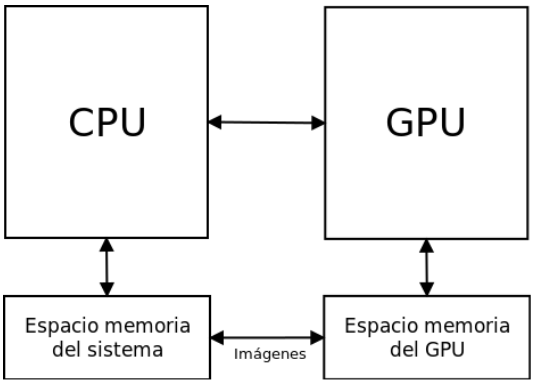

Figura 3.10: Esquema básico de la interacción entre el CPU y el GPU.

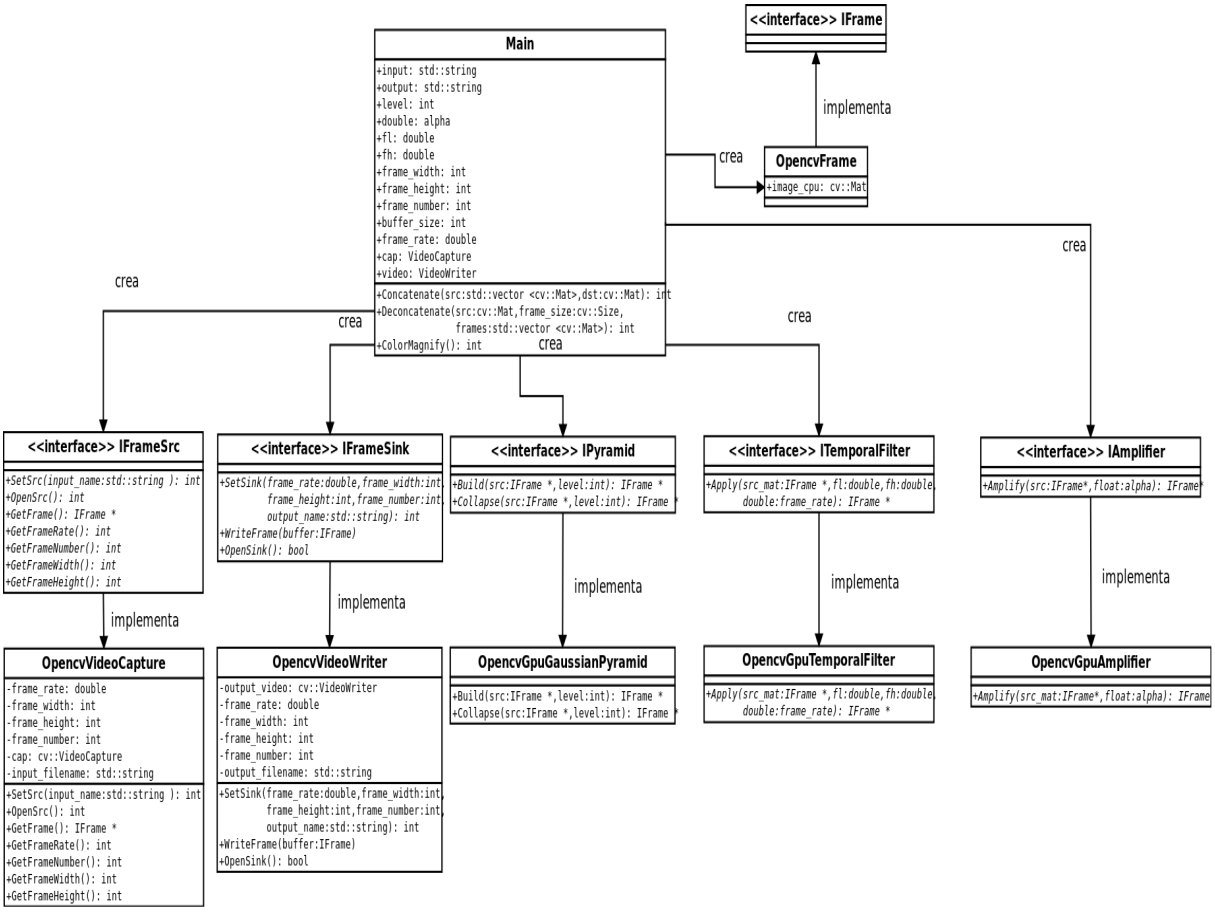

Figura 3.11: Diagramas de clases de la implementación del algoritmo de magnificación euleriana de vídeo (GPU).

3.6. Extracción de la señal del pulso cardíaco

La técnica empleada busca aplicar el principio de detectar el pulso cardíaco con secuencias de vídeo a partir de variaciones en el color de la piel provocadas por el flujo de la sangre a través del cuerpo, para ello se utilizó el algoritmo anteriormente descrito en la sección 2.5.

La recuperación de la señal del pulso cardíaco a partir del vídeo procesado se realizó a través de las herramientas de Octave, por facilidad en la transferencia de los datos procesados entre la tarjeta Jetson TX1 y el PC, se empleó el protocolo SSH. El posterior procesamiento adicional (al algoritmo de MEV) donde se extrae la información del pulso cardíaco a partir de la variación de color en un único píxel o en un área específica de la imagen (por ejemplo la frente del sujeto de prueba) se realiza por medio de la biblioteca OpenCV, es decir, por sí solo el algoritmo de magnificación euleriana no permite calcular directamente el pulso cardíaco, en vez de esto, se trata de un procesamiento general a un vídeo donde se rescatan las minúsculas variaciones de color o movimiento de acuerdo al filtrado espacio-temporal que requiera una aplicación específica.

El proceso empleado para la extracción de la componente principal de la señal del pulso cardíaco se muestra en la figura 3.13. Para obtener este dato primero se calcula o reconstruye una señal (datos de intensidad de color de los píxeles tomados en el tiempo), posteriormente se realiza un análisis en el dominio de la frecuencia de la señal recuperada, a partir de la variación del color en una región específica de las imágenes en el vídeo (generalmente la cara), como se muestra en la figura 3.12.

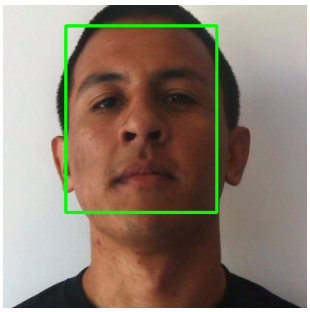

Figura 3.12: Detección de la región de interés (cara).

Para la extracción de la señal se realiza la detección del rostro de la persona en el vídeo procesado por medio del algoritmo de MEV. La recuperación de la señal se realiza mediante el uso de la biblioteca OpenCV, para ello se obtienen las coordenadas de la región de la cara del sujeto en las imágenes a través del algoritmo de detección de objetos Viola-Jones. OpenCV utiliza este algoritmo y clasificadores en cascada previamente entrenados para detectar muchos

tipos de objetos, uno de estos clasificadores permite la detección de rostros. Por cada rostro detectado en la imagen el algoritmo retorna las coordenadas en (x, y) así como la altura y el ancho de la región de interés que contiene el rostro. Las dimensiones y ubicación de la región de interés varían dinámicamente a través del análisis del vídeo. De la región de interés se promedian espacialmente con igual peso todos los píxeles para reconstruir una muestra de la señal por cada componente RGB de la imagen analizada, las señales extraídas tendrán el mismo número de muestras que la cantidad total de imágenes en el vídeo. Cada una de estos valores de la señal se guardaron en archivos de texto para su posterior visualización y análisis por medio de Octave.

El posterior análisis y visualización se realizó por medio de un *script* en Octave que se basa en tomar los archivos de texto que contienen las muestras de la señal del pulso cardíaco. Los datos fueron procesados a través de un filtro FIR de media móvil, según la Ec.3.4, para suavizar la señal y combatir la degradación por motivos de ruido, se encontró que la aplicación de este filtro disminuyó el error en la técnica de selección de la máxima amplitud. Posteriormente se aplica la FFT (*Fast Fourier Transform*) a las señales obtenidas (1 por cada canal) para obtener su representación en el dominio de la frecuencia y finalmente se encarga de graficar la magnitud del espectro de frecuencia (representando solo el lado positivo). La frecuencia del pulso cardíaco obtenida corresponde a la frecuencia a la máxima del espectro (componente frecuencial de mayor amplitud) dentro del ancho de banda de interés de 0.4 Hz a 4 Hz aproximadamente [13]. En la figura 3.13

$$y(n) = \frac{1}{M+1} \sum_{k=0}^{M} x(n-k) \tag{3.4}$$

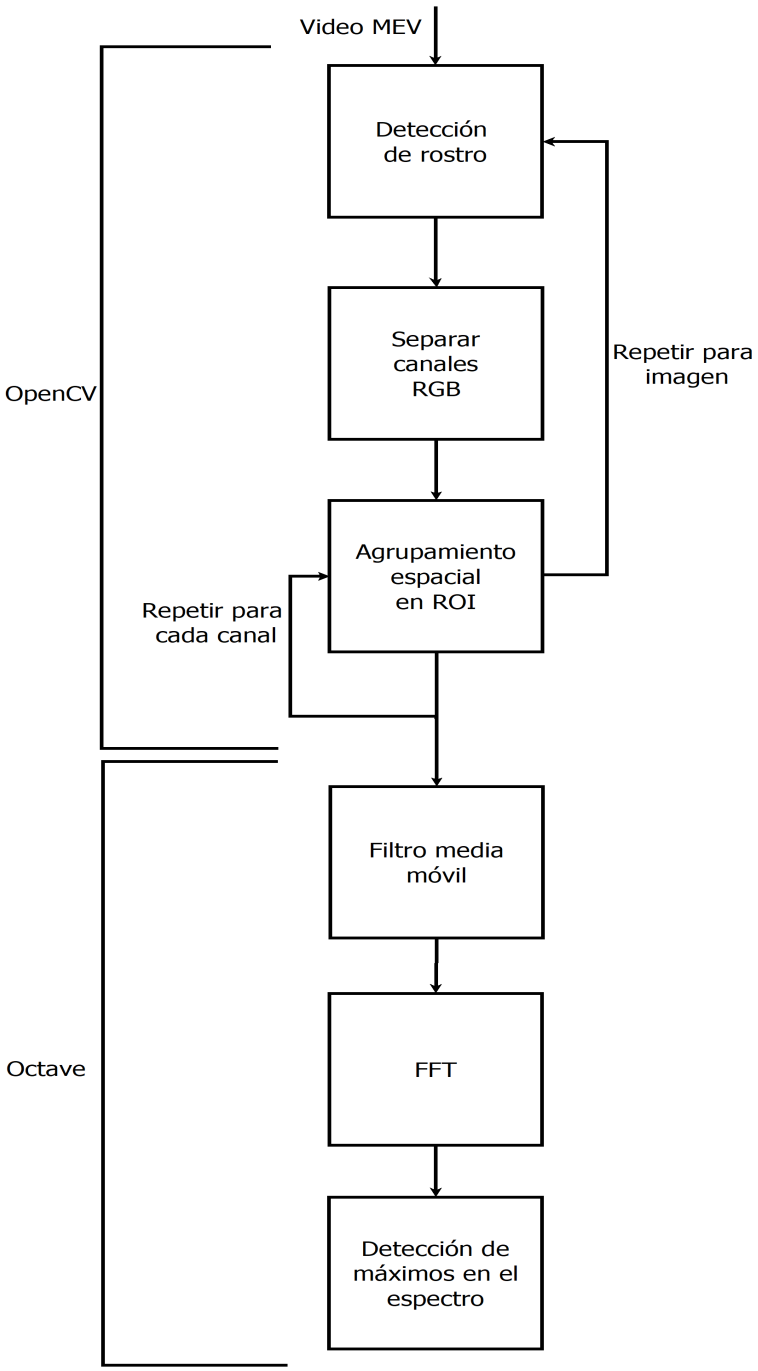

Figura 3.13: Esquema general del procesamiento para la extracción del pulso cardíaco.

CAPÍTULO 4

Pruebas y resultados

4.1. Comprobación de la implementación del algoritmo MEV

Para demostrar la veracidad de los resultados de la implementación del algoritmo MEV (ver sección 2.5) se empleó el índice de similitud estructural de imágenes explicado con más detalle en la sección 2.6. El cálculo de este índice fue hecho para cada cuadro individual de los vídeos procesados tanto en la implementación que se ejecuta en el CPU como la que hace uso del GPU. Dado que el índice del SSIM se calcula para cada imagen, los porcentajes mostrados en la tabla 4.2 corresponden a un promedio del índice de cada imagen, esto resulta en una manera de puntaje general por componente de color en el espacio RGB. El vídeo utilizado como referencia para esta prueba se puede encontrar en la página del CSAIL en la sección de resultados en el apartado correspondiente al algoritmo MEV [12], específicamente, se emplearon los mismos parámetros del algoritmo tanto para la ejecución en el CPU como la del GPU, esta configuración se enumera en la tabla 4.1:

Cuadro 4.1: Parámetros de configuración para la comprobación de la implementación del algoritmo MEV

Parámetro	Valor
Nivel de la pirámide	4
Frecuencia de corte inferior	0.83
Frecuencia de corte superior	1
Factor de amplificación	50

Cuadro 4.2: Comparación de la implementación del CPU y GPU a través del SSIM

Procesador	PSNR	SSIM canal rojo (%)	SSIM canal verde (%)	SSIM canal azul (%)
CPU	31,861	95,1	94,9	92,9
GPU	31,405	95	94,7	92,7

4.2. Comparación del rendimiento en el CPU y en el GPU

Para la comparación del rendimiento entre la implementación del algoritmo que no hace del GPU y la que sí lo utiliza se recolectaron los datos presentados en las tablas 4.3 y 4.4. Los datos se obtuvieron a partir de la utilización de la herramienta llamada *tegrastats* que forma parte de la plataforma de soporte de la tarjeta Jetson TX1 según la sección 3.1.

La herramienta *tegrastats* permite la recolección de información sobre el uso del recursos de hardware directamente de los archivos manejados por el *kernel* de Linux. En este sentido la información proveniente de archivos como /proc/stat o /sys/devices/platform/host1x/gpu.0/load, es interpretada y desplegada en la salida estándar.

La prueba realizada implicó la ejecución del algoritmo con dos vídeos de distintas resoluciones una de 640x480 píxeles y otro de 1920x1080 píxeles. La herramienta *tegrastats* brinda la información, cada cierto intervalo definido por el usuario, de cada uno de los núcleos del CPU (4 para el caso de la TX1), el GPU, el uso de memoria, decodificadores y codificadores de audio y vídeo, entre otros. Para el caso del porcentaje de utilización del CPU *tegrastats* brinda un porcentaje individual por cada núcleo relativo a la frecuencia de reloj de acuerdo a la carga de trabajo del procesador, en este sentido el porcentaje de utilización mostrado en las 4.3 y 4.4 corresponde al promedio de las sumas de los porcentajes individuales de cada núcleo.

Cuadro 4.3: Estadísticas de rendimiento para un vídeo de 640x480 @30 fps

Implementación		MEV (CPU)	MEV (GPU)
% Utilización del CPU	Máximo	133	120
	Promedio	109	84
% Utilización del GPU	Máximo	0	76
	Promedio	0	35
Uso de memoria (GB)	Máximo	2,3	2,8
Tiempo ejecución	Promedio por imagen (ms)	6,91 (∼145 fps)	34,2 (∼29 fps)
	Total (s)	2.08	10.48

Cuadro 4.4: Estadísticas de rendimiento para un vídeo de 1920x1080 @30 fps

Implementación		MEV (CPU)	MEV (GPU)
% Utilización del CPU	Máximo	136	136
	Promedio	107	94
% Utilización del GPU	Máximo	0	96
	Promedio	0	27
Uso de memoria (GB)	Máximo	3,1	3,5
Tiempo ejecución	Promedio por imagen (ms)	13,71 (∼73 fps)	44,33 (∼23 fps)
	Total (s)	2.16	6,84

4.3. Extracción del pulso cardíaco

4.3.1. Variación de los parámetros del algoritmo

Otra de las de pruebas realizadas fue someter el procesamiento del algoritmo MEV con distintas configuraciones de parámetros y posteriormente analizar la afectación de ello en la recuperación de la señal del pulso cardíaco. En las figuras siguientes se muestran algunos de los resultados obtenidos. Para estas pruebas se utilizó el mismo vídeo de los resultados de la sección 4.1, con los mismos parámetros de configuración de la tabla 4.1.

En conjunto también se muestra el comportamiento de los resultados del procesamiento aplicado a un vídeo grabado a través de una cámara digital (Canon EOS 60D). En este vídeo el sujeto se fue grabado en condiciones de mucha agitación física, para ello los parámetros del algoritmo deben ser ajustados a estas nuevas condiciones. La ejecución del algoritmo se ajustó en este caso con un nivel de las pirámides gaussianas de 4, una frecuencia de corte inferior de 2.4 Hz, una frecuencia de corte superior de 2.7 Hz, un factor de amplificación de 50 y un tamaño de bloque de imágenes de 100.

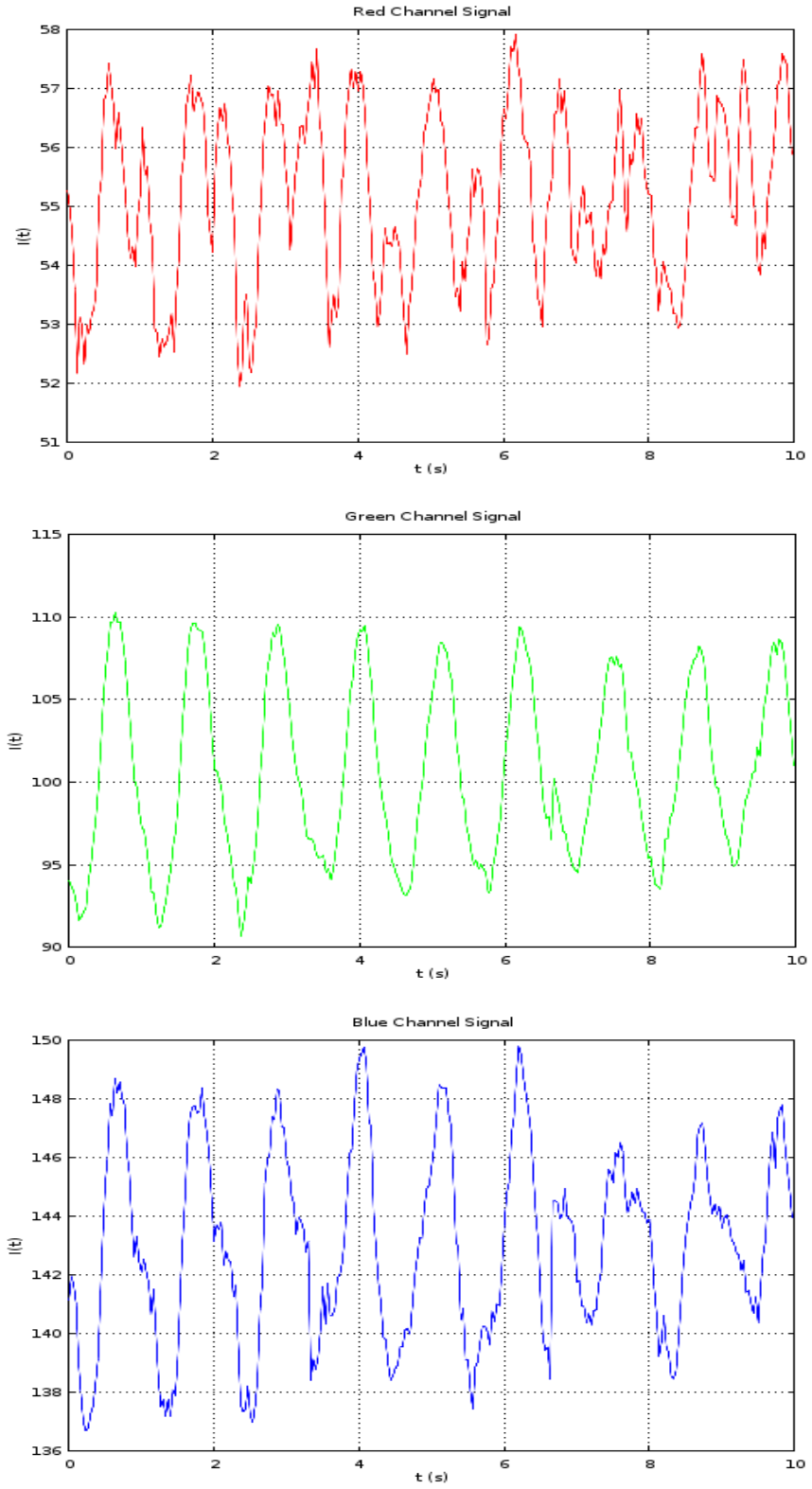

Figura 4.1: Variación de la intensidad de los píxeles de los canales RGB del vídeo de referencia.

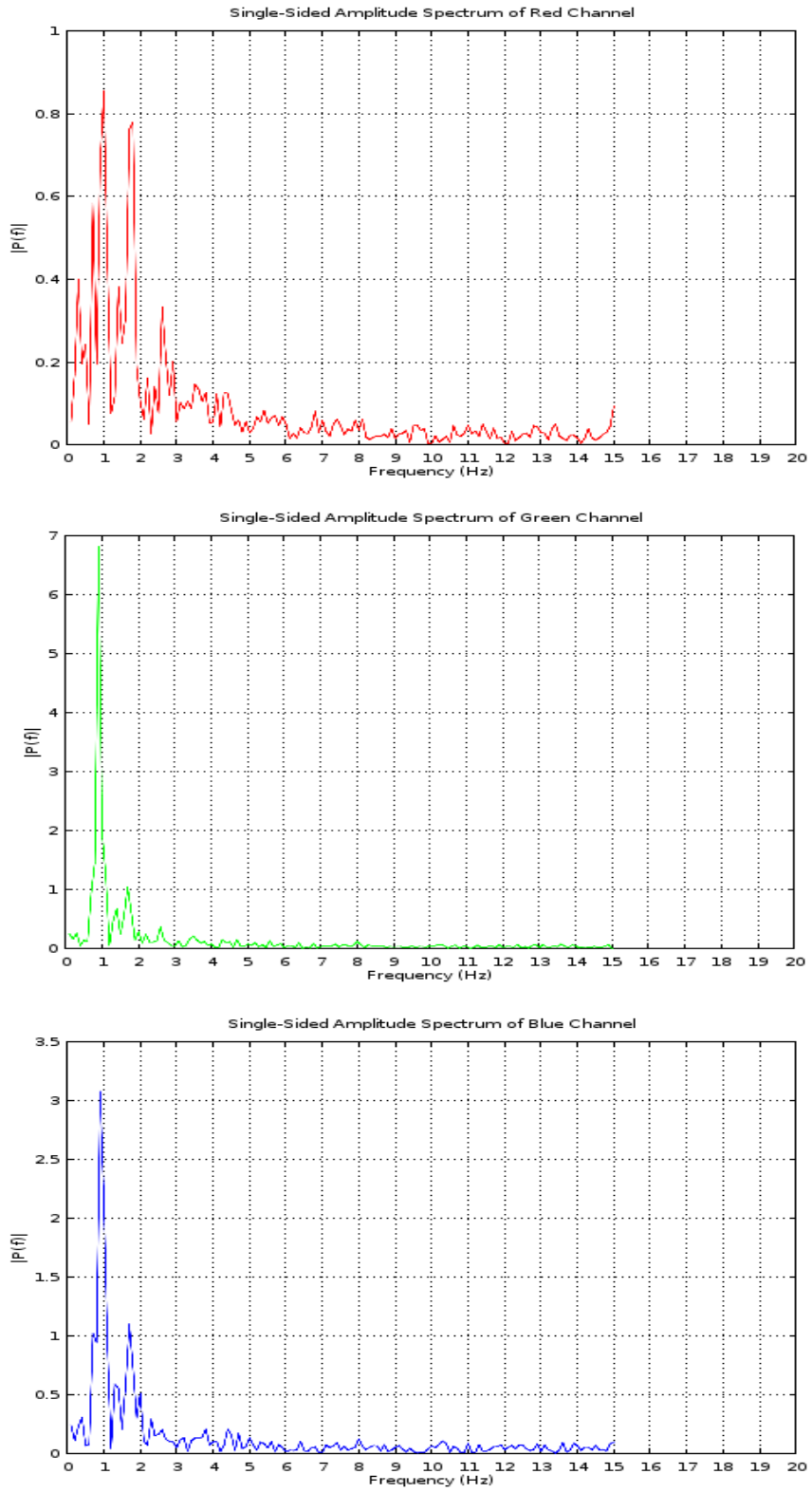

Figura 4.2: Espectro de frecuencias de las señales de intensidad de los píxeles de los canales RGB del vídeo de referencia.

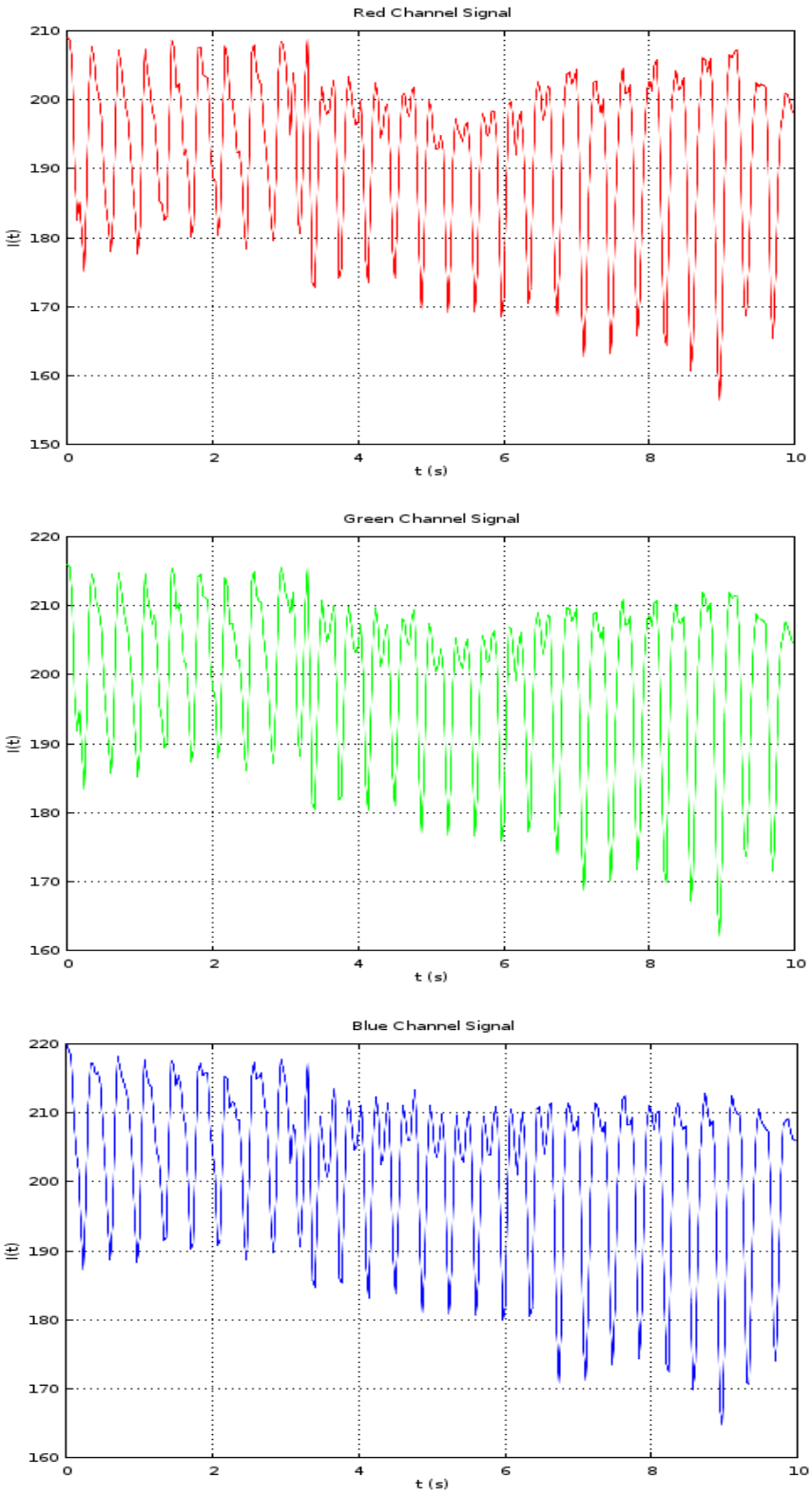

Figura 4.3: Variación de la intensidad de los píxeles de los canales RGB del vídeo grabado.

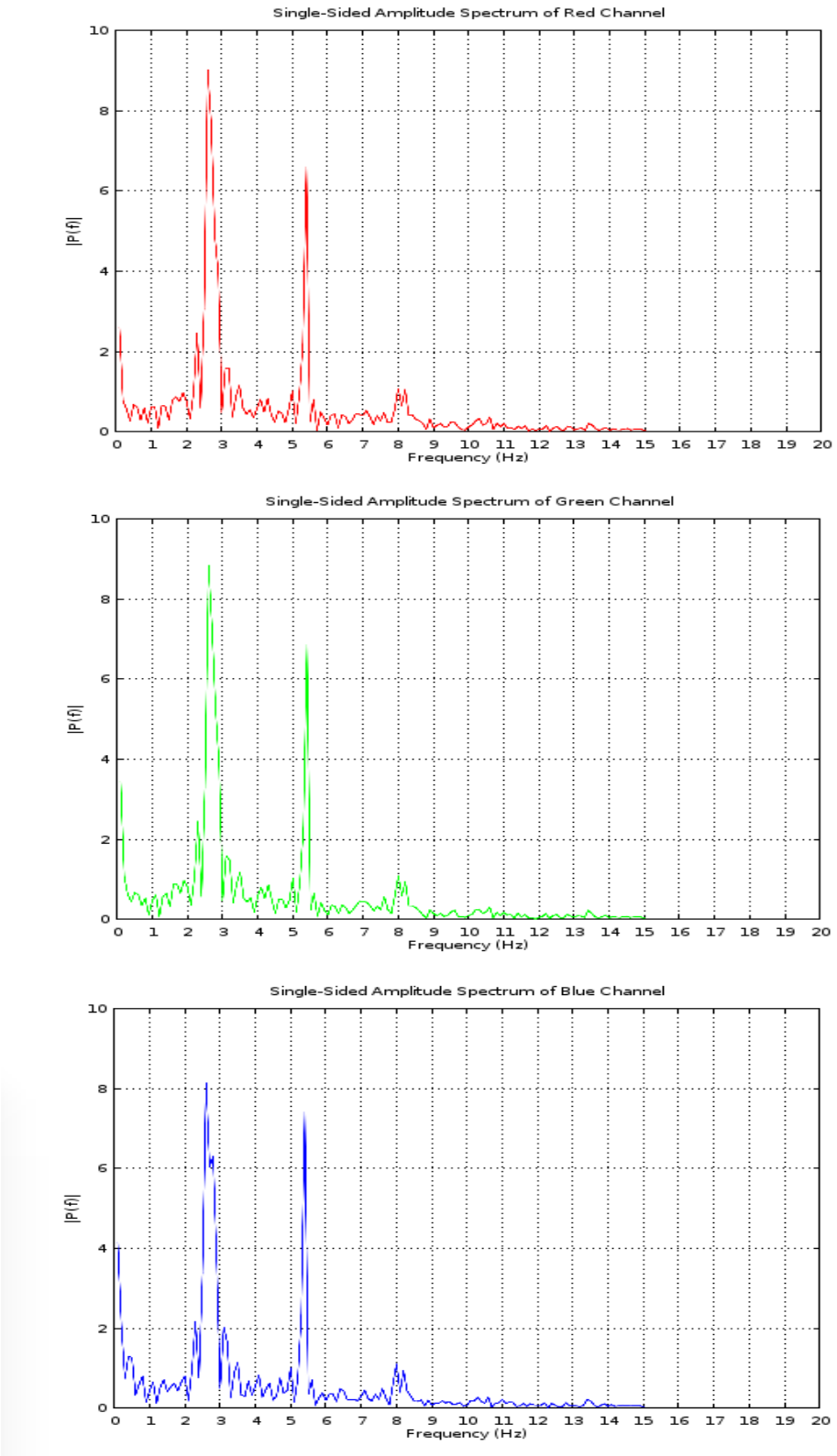

Figura 4.4: Espectro de frecuencias de las señales de intensidad de los píxeles de los canales RGB del vídeo grabado.

4.3.2. Variación de los sujetos de prueba

La adquisición de los datos para las pruebas de la extracción del pulso cardíaco se realizó a partir de vídeos de 10 s aproximadamente, con una resolución de 640x480 píxeles y 60 cuadros por segundo. Los datos de vídeo fueron contrastados simultáneamente con mediciones de un oxímetro de pulso de la marca Medline (modelo HCSM70P). Las mediciones fueron tomadas a partir de un grupo de 5 personas de entre 23-28, con distintas tonalidades de piel, dentro de una habitación con iluminación natural proveniente de una ventana, de estas personas voluntarias se grabaron vídeos tanto en condiciones de reposo y agitación. La cámara digital empleada para la grabación de los vídeos fue el modelo Canon EOS 60D. En las tablas 4.5, 4.6, 4.7, 4.8, 4.9 se resumen los datos obtenidos, tanto las estimaciones a través del algoritmo de MEV, las mediciones de referencia y el porcentaje de exactitud definido a través de la Ec.4.1.

$$\%Exactitud = 100\,\% - \left(\frac{|PC_r - PC_e|}{PC_r} * 100\,\%\right) \tag{4.1}$$

Cuadro 4.5: Resultados del monitoreo del pulso cardíaco para el sujeto 1

Medición			1	2	3	4	5	6
Estado	Agitado	Oxímetro (bpm)	97	95	86	85	82	80
		Estimado (bpm)	102	102	84	92	90	90
		Exactitud (%)	94,9	92,6	97,7	91,8	90,2	87.5
	Reposo	Oxímetro (bpm)	55	54	55	57	56	57
		Estimado (bpm)	54	51	48	60	56	60
		% Exactitud	98,2	94,4	87,3	94,7	100	94,7

Cuadro 4.6: Resultados del monitoreo del pulso cardíaco para el sujeto 2

Medición			1	2	3	4	5	6
Estado	Agitado	Oxímetro (bpm)	156	153	145	140	137	135
		Estimado (bpm)	156	156	138	138	138	126
		Exactitud (%)	100	98	95,2	98,57	99,3	93,3
	Reposo	Oxímetro (bpm)	74	74	74	72	73	75
		Estimado (bpm)	74	72	66	78	75	72
		% Exactitud	100	97,3	89,2	91,7	97,3	96

Cuadro 4.7: Resultados del monitoreo del pulso cardíaco para el sujeto 3

Medición			1	2	3	4	5	6
Estado	Agitado	Oxímetro (bpm)	176	168	163	161	160	156
		Estimado (bpm)	176	176	156	156	156	156
		Exactitud (%)	100	95,2	95,7	96,9	97,5	100
	Reposo	Oxímetro (bpm)	97	99	96	96	96	95
		Estimado (bpm)	109	106	108	102	102	105
		% Exactitud	87,5	92,9	87,5	93,8	93,8	89,5

Cuadro 4.8: Resultados del monitoreo del pulso cardíaco para el sujeto 4

Medición			1	2	3	4	5	6
Estado	Agitado	Oxímetro (bpm)	103	92	91	90	90	87
		Estimado (bpm)	102	90	84	84	84	84
		Exactitud (%)	99	97,8	92,3	93,3	93,3	96,6
	Reposo	Oxímetro (bpm)	86	82	82	82	88	87
		Estimado (bpm)	96	86	84	84	96	84
		% Exactitud	88,4	95,1	97,6	97,6	90,1	96,6

Cuadro 4.9: Resultados del monitoreo del pulso cardíaco para el sujeto 5

Medición			1	2	3	4	5	6
Estado	Reposo	Oxímetro (bpm)	100	94	93	87	91	95
		Estimado (bpm)	110	90	96	92	90	98
		% Exactitud	90	95,7	96,8	94,3	98,9	96,8

A partir de los datos obtenidos se procedió a sintetizar los mismos a través de dos herramientas estadísticas: el gráfico de dispersión y el gráfico Bland-Altman. Esto se realizó con el fin de facilitar la interpretación cuantitativa de los datos a través de cálculos de la estadística descriptiva. Por su parte el gráfico de dispersión de la figura 4.5 señala la tendencia de las muestras tomadas y la capacidad de las estimaciones a través de la implementación del algoritmo de MEV frente a las mediciones de referencia. En el caso ideal se tendría que cada valor estimado correspondrá al mismo que el medido lo que correspondería con un coeficiente de correlación de Pearson (R) de 1. En el caso de las mediciones obtenidas, R corresponde a 0,985, cercano a 1 por lo que se puede interpretar que existe una alta dependencia lineal entre los datos, es decir si la medición de referencia aumenta la estimación a través del algoritmo de MEV aumenta también proporcionalmente y forma constante.

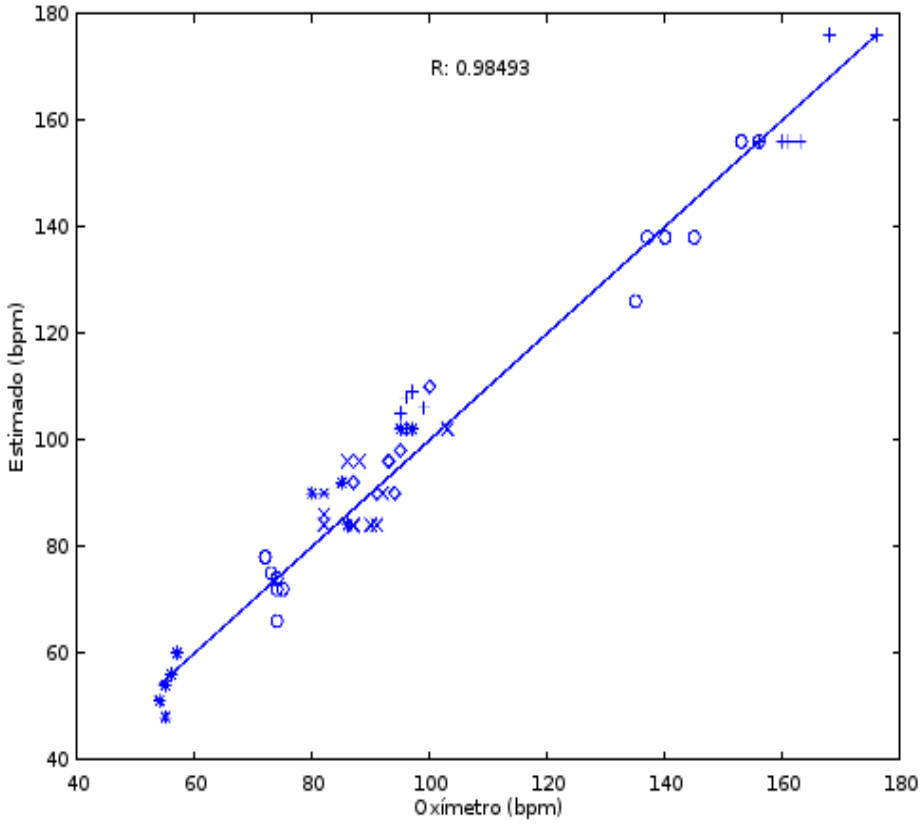

Figura 4.5: Gráfico de dispersión de la medición del oxímetro vs la estimación por medio del algoritmo MEV.

Dado que a partir de la figura 4.5 se muestra que las mediciones entre ambas métodos tienen una alta con correlación es posible ampliar el análisis estadístico de los datos a través del gráfico de Bland-Altman. Se sabe que ambas mediciones contienen un grado de error, de esta manera el cálculo de la media y la desviación estándar de las diferencias entre los datos permite definir un intervalo de confianza. En este caso las líneas horizontales en la figura 4.6 muestra la media y límites del intervalos de confianza del 95 %. Como se puede puede observar, la gran parte de las mediciones obtenidas se encuentra dentro de este intervalo de confianza fijado.

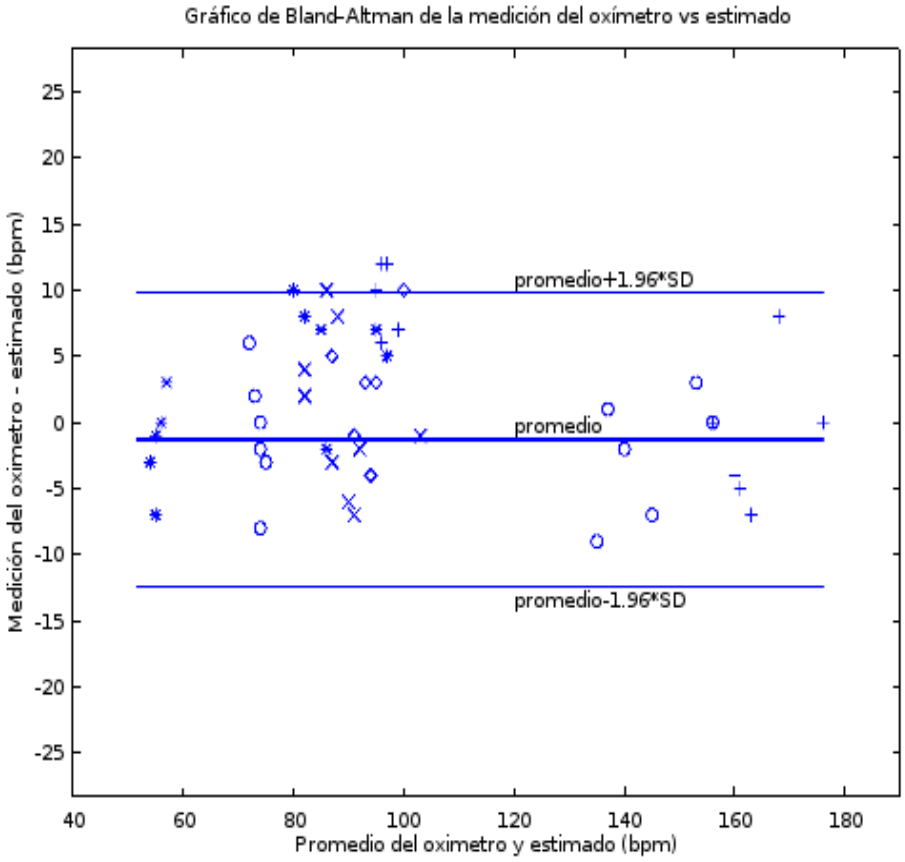

Figura 4.6: Gráfico de Bland-Altman para las mediciones del oxímetro vs la estimación por medio del algoritmo MEV.

CAPÍTULO 5

Análisis de resultados

A partir de los datos de la sección 4.1 se puede afirmar que la incorporación de los módulos que se ejecutan en el GPU no afectan los resultados de la implementación original que emplea el solo el CPU de la Jetson TX1. Si existiese alguna discrepancia por debajo del 90 % en el SSIM esta sería despreciable puesto que en general se puede asegurar que los resultados del procesamiento digital de señales aplicados sobre los píxeles son los mismos. Asimismo se puede observar de la tabla 4.2 que la implementación del algoritmo se asemeja como mínimo en 92,9 % que corresponde al índice de similaridad para el canal del color azul. Se puede definir también un porcentaje global de similaridad al obtener el promedio de los tres canales, el cual ronda el 94,1 %.

Se tienen, a partir de la sección 4.2, los resultados de la prueba de rendimiento realizada los procesador central y el procesador gráfico. Una de las principales observaciones a partir de las mediciones las tablas 4.3 y 4.4 es que la implementación de las interfaces de cada uno de los módulos del algoritmo a través del módulo del gpu de OpenCV no implica un menor tiempo de ejecución para cada imagen. Aún así, para ambas implementaciones se logra obtener un tiempo de ejecución para el procesamiento del algoritmo mucho menor a los 250 ms como se planteó en un inició.

En aplicaciones para sistemas embebidos, en términos generales, es crítica la reducción de la carga de trabajo sobre el CPU. El sistema en chip TX1 posee muchos recursos de hardware adyacentes al CPU. de acuerdo con los principios de la computación heterogénea es preferible repartir las tareas entre los distintos procesadores. En este sentido, un aspecto importante a considerar es que el diseño de la arquitectura para la implementación del algoritmo de MEV a través de módulos no estaba orientado principalmente a la optimización de los tiempos de ejecución, sino en realizar una mejor distribución de la carga de trabajo y uso de los recursos. Esta consideración es visible en la disminución de la carga de trabajo

sobre el procesador central para los dos tipos de vídeos probados. Las mediciones muestran que se reduce la carga sobre el CPU en alrededor del 23 % y en un 12 %, respectivamente. Estas mediciones fueron realizadas a lo largo de toda la de ejecución del programa, lo cual implica una parte de construcción del vídeo de salida que ocurre enteramente en el CPU y no forma parte necesariamente del algoritmo de MEV, por lo que el porcentaje de utilización CPU se puede reducir aún más delegando esta etapa a otros procesadores, por ejemplo, los decodificadores de vídeo.

Asimismo se tiene que el procesador gráfico es más robusto en cuanto a la carga de trabajo que puede soportar, es decir, se observa como el cambio de procesamiento a un vídeo mayor resolución afecta en gran medida el tiempo que le toma implementación que no utiliza este coprocesador para procesar cada imagen, en este caso el tiempo de ejecución para cada imagen aumenta en casi un 100 % mientras que para la implementación que se apoya en el GPU este tiempo aumenta en aproximadamente un 30 %.

Es evidente que la utilización de la librería OpenCV limita los tiempos de ejecución de la implementación en el GPU. Para mejorar los tiempos de ejecución del algoritmo, es necesario explorar otras técnicas y herramientas para la implementación de las interfaces de la arquitectura mostrada en la sección 3.5. Una opción es la realización de *kernels* (funciones específicas para C CUDA) optimizadas para la lógica que implica el algoritmo de MEV y posteriormente lograr incorporar estas librerías específicas en una ejecución en tiempo real.

En cuanto a los resultados de la extracción del pulso cardíaco. De acuerdo con la sección 4.3.1 se debe rescatar un aspecto importante a partir de la figura 4.3 el cual es precisamente la variación de la amplitud de la señal de intensidad de los píxeles en los tres canales. Se puede observar que esta afectación de la señal se introduce aún después del agrupamiento espacial aplicado en los píxeles de la región de interés. Esto hace indicar que el algoritmo de MEV es sensible a las variaciones de iluminación de las grabaciones de vídeo y el movimiento en la escena que se esté grabando. La ubicación de la fuente de la luz en la imagen debe ser entonces un factor a considerar antes de la adquisición de las imágenes y la posterior aplicación del algoritmo de magnificación euleriana de vídeo. Por su parte en la figura 4.1 se nota como esta distorsión de las señales no está presente y en el caso de la señal del canal verde se tiene casi una sinusoidal pura a la frecuencia que indica el pulso cardíaco, lo que muestra que sí es posible rescatar la señal fotopletismográfica del flujo sanguíneo.

En este sentido también se puede observar como la variación en la amplitud se evidencia en el análisis espectral realizado sobre las señales de los tres canales. En la figura 4.4 se tiene cómo la variación de amplitud se traslada a una componente espectral en las frecuencia muy

bajas, por debajo de 1 Hz. Así también se ve la presencia clara de la frecuencia cardiovascular denotada como la componente de mayor amplitud en todo el espectro. Aunque se presenta una segunda componente de presencia considerable en los espectros de la figura 4.4, la misma puede ser descartada del análisis puesto que el rango de interés en general ronda la banda de frecuencia de 0.4 Hz a 4 aproximadamente.

El pulso cardíaco es una señal que no mantiene constante su frecuencia (no aparece a un intervalo constante), como se puede observar a partir de los datos de las tablas de la sección 4.3.1 aún para mediciones realizadas sobre la misma persona este valor no se mantiene constante y posee una variabilidad natural, según los datos de referencia. Sin embargo el procesamiento aplicado en los vídeos permite revelar que esta técnica aproxima el valor del pulso cardíaco en su mayoría con un porcentaje de exactitud de alrededor del 95 %, incluso se pueden observar estimaciones con 100 % de exactitud.

Una limitación encontrada en las pruebas realizadas es la resolución con la que se cuenta en el espectro de frecuencia estimado de las señales de intensidad de los píxeles. A partir de la Ec.2.10 se tiene que la resolución de puntos en el espectro depende del número de muestras tomadas y la frecuencia de muestreo. El segundo es un parámetro que solo se puede ajustar de acuerdo al flujo de vídeo que se haya procesado, en este caso, este se encuentra fijado en 30 fps. El número de muestras también viene establecido por los cuadros por segundo y duración del vídeo, al haber optado por procesar vídeos de 10 s, se cuenta con una resolución de frecuencia en la estimación del pulso de 0,1 Hz.

Posteriormente se optó por hacer un análisis estadístico a partir de los 5 sujetos de prueba como se detalla en la sección 4.3.2. Primero a través del gráfico de dispersión se puede observar la fuerte dependencia lineal entre las mediciones, lo que se traduce en un coeficiente de correlación alto, cercano a 0,99. El gráfico de dispersión permite visualizar la capacidad del procesamiento de seguir las mediciones de referencia, sin embargo esto no dice mucho sobre la variabilidad de los valores estimados del pulso cardíaco. Para resolver esto se procedió con la obtención del gráfico de Bland-Altman de la figura 4.6. A partir de este análisis se obtuvo que los datos se distribuyen con una media entre las diferencias de las mediciones de -1.3 bpm y con los límites del intervalo de confianza del 95 % ubicado a $\pm 11,1$ bpm, aproximadamente. Esta muestra de datos permite entonces afirmar que la posibilidad de fallar en el valor del pulso cardíaco en más 11 bpm como mucho es igual al 5 %. Este tipo de análisis permite visualizar el grado de error en las mediciones más allá de un simple cálculo de porcentaje de error relativo.

CAPÍTULO 6

Conclusiones y recomendaciones

6.1. Conclusiones

En síntesis, se ha descrito el proceso de diseño e implementación de un sistema que permite la estimación del pulso cardíaco a partir de vídeo a partir del algoritmo de magnificación euleriana que se ejecuta en el sistema embebido Jetson TX1. Para este propósito se utilizaron dos dispositivos de hardware disponibles en el sistema en chip de la tarjeta de desarrollo así como la biblioteca OpenCV y otras herramientas como Octave, todo en un ambiente de desarrollo basado GNU/Linux.

Se ha verificado asimismo, a través del índice estructural de imágenes, que las implementaciones del algoritmo de MEV mediante el lenguaje de programación C++ y OpenCV, tanto la que hace uso del GPU como la que no, presentan prácticamente los mismos resultados entre sí y respecto a los resultados de referencia tomados de [12].

De las varias implementaciones realizadas para el algoritmo de MEV se ha seleccionado la arquitectura modularizada que hace uso del GPU, puesto que, aunque presenta mayores tiempos de ejecución respecto a la ejecución exclusiva en el CPU, es la que muestra los mejores beneficios en términos de la reducción de la carga de trabajo sobre el procesador central, un aspecto crítico en el uso de sistemas embebidos, y la oportunidad de optimización (sobre la misma arquitectura) en el futuro para que esta puesta funcionar como un sistema en tiempo real. Se ha demostrado que es posible dividir el procesamiento de las imágenes a través del algoritmo MEV en bloques individuales para hacer frente a las limitaciones de memoria en el sistema embebido empleado.

Los resultados de la prueba de rendimiento indican que la utilización de la biblioteca OpenCV, es una limitante en cuanto al tiempo de ejecución. Por lo tanto, la búsqueda de alternativas en la implementación del código para el GPU es necesaria. La arquitectura propuesta permite cambiar las bibliotecas para el procesamiento de imágenes sin la necesidad de reescribir la implementación del algoritmo de magnificación euleriana de vídeo desde cero.

Se han presentado los pasos en el procesamiento de señales aplicado posteriormente para la estimación del pulso cardíaco. Se han realizado pruebas sobre distintas personas voluntarias con la técnica propuesta y la misma permitió resultados en la mayoría de los casos con un porcentaje de exactitud cercano al 95 %, respecto a una medición simultánea con otro dispositivo, como le fue un oxímetro. Se ha visto cómo la técnica utilizada para la estimación del pulso no está sujeta a la implementación del algoritmo de MEV y por lo tanto el esquema de posprocesamiento puede ser variado de acuerdo a necesidades particulares a partir de las condiciones en que son grabados los vídeos.

6.2. Recomendaciones

Para un sistema que requiera ser ejecutado en tiempo real se recomienda que la implementación del algoritmo de MEV sea incorporado dentro de la interfaz de programación GStreamer. Asimismo es posible incorporar *kernels* o funciones altamente optimizadas y personalizadas en CUDA C para la ejecución paralela del algoritmo. Con esto será posible reducir los tiempos ejecución para cada imagen (cuadros por segundo). También para la toma de búferes de video, en tiempo real, será necesaria la incorporación dentro bibliotecas especializadas como GStreamer.

El posterior procesamiento para la estimación del pulso puede variar la estrategia del procesamiento de señales para ser más robusta en casos donde la señal del pulso cardíaco se encuentra altamente contaminada por artefactos indeseados en el vídeo procesado. Es decir, la solución planteada y las pruebas realizadas asumen condiciones establecidas y controladas de iluminación y estabilidad de la imagen al momento de tomar los vídeos, pero un sistema no invasivo, en tiempo real debe ser capaz de sobrellevar una gran gama de condiciones en el ambiente donde se toman las imágenes.

Asimismo se puede reducir la carga de procesamiento del algoritmo de MEV al reducir el número de píxeles a procesar. Para ello se puede aplicar la segmentación de la cara de la persona y posteriormente aplicar MEV únicamente sobre los píxeles contenidos en esta región de interés. De igual manera se ha descrito como el algoritmo de MEV es sensible a la iluminación y al movimiento de la imagen, para el primer caso es posible aplicar un procesamiento previo para la corrección automática de la iluminación o aplicar la conversión de las imágenes a otros espacios de color, para el segundo se puede aplicar una estabilización de la imagen.

Bibliografía

[1] RidgeRun. Internet: http://www.ridgerun.com/, 2018 [31 Mayo, 2018].

[2] GStreamer. Internet: https://gstreamer.freedesktop.org/, 2018 [31 Mayo, 2018].

[3] "Jetson TX1", Embedded Linux Wiki. Internet: http://elinux.org/Jetson_TX1#Jetson_TX1_Module, 2017 [31 Mayo, 2018].

[4] "Tegra". Internet : http://www.nvidia.es/object/tegra-x1-processor-es.html, 2018 [31 Mayo, 2018].

[5] "Embedded Systems". Internet: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html, 2017 [31 Mayo, 2018].

[6] "Arquitectura de GPU Maxwell". Internet: http://www.nvidia.es/object/maxwell-gpu-architecture-es.html, 2017 [31 Mayo, 2018].

[7] "Cuda Zone". Internet: https://developer.nvidia.com/cuda-zone, 2018 [31 Mayo, 2018].

[8] "Deep Learning and AI Solutions for Industries". Internet: https://www.nvidia.com/en-us/deep-learning-ai/industries/, 2017 [31 Mayo, 2018].

[9] "From Winning Go to Making Dough: What Can Deep Learning Do for Your Business?". Internet: https://blogs.nvidia.com/blog/2016/10/17/deep-learning-help-business/, 2017 [31 Mayo, 2018].

[10] "How will artificial intelligence change healthcare?". Internet: https://www.forbes.com/sites/quora/2017/06/09/how-will-artificial-intelligence-change-healthcare/#6100531b6d67, 2017 [31 Mayo, 2018].

[11] L. Fei-Fei. "Machine Learning and Computer Vision". Internet: https://www.cs.princeton.edu/courses/archive/spring07/cos424/lectures/li-guest-lecture.pdf [31 Mayo, 2018].

[12] “Video Magnification”. Internet: http://people.csail.mit.edu/mrub/vidmag/, 2015 [31 Mayo, 2018].

[13] H. Wu et al. “Eulerian video magnification for revealing subtle changes in the world”. *ACM Trans. Graph. (Proceedings SIGGRAPH 2012)*, 31(4), 2012.

[14] M. MacGill. “¿Qué es la frecuencia cardiaca? ¿Cuál es la normal?” . Internet: https://www.medicalnewstoday.com/articles/291182.php, 2016 [31 Mayo, 2018].

[15] A. Davis, et al. "Visual vibrometry: Estimating material properties from small motion in vídeo."*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[16] J.G. Chen, et al. "Modal identification of simple structures with high-speed vídeo using motion magnification."*Journal of Sound and Vibration*, 345 (2015): 58-71.

[17] T. Xue, et al. Refraction wiggles for measuring fluid depth and velocity from vídeo."*European Conference on Computer Vision*, Springer, Cham, 2014.

[18] A. Davis, et al. "The visual microphone: Passive recovery of sound from video". Internet: http://hdl.handle.net/1721.1/100023, 2014. [31 Mayo, 2018].

[19] C. Liu, et al. "Motion magnification."*ACM transactions on graphics (TOG)*, 24.3 (2005): 519-526.

[20] N. Wadhwa, M. Rubinstein, F. Durand, W. T. Freeman. "Phase-based vídeo motion processing". *ACM Transactions on Graphics (TOG)*, 32(4), 80, 2013.

[21] M.Z Poh, D.J. McDuff, & R.W. Picard. "Non-contact, automated cardiac pulse measurements using vídeo imaging and blind source separation."*Optics express* 18.10 (2010): 10762-10774.

[22] G. Balakrishnan, F. Durand, J. Guttag. “Detecting pulse from head motions in vídeo”. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3430-3437, 2013.

[23] E. Fustero. “¿Qué es la computación heterogénea?". Internet: https://blogthinkbig.com/computacion-heterogenea-2, 2013 [31 Mayo, 2018].

[24] P. Moorhead, Forbes, 2013. “Qualcomm: Don’t Confuse Heterogeneous Computing (HC) With Heterogeneous Multiprocessing (HMP)”. Internet: https://www.forbes.com/forbes/welcome/?toURL=https://www.forbes.com/sites /patrickmoorhead/2013/11/18/qualcomm-dont-confuse-heterogeneous-computing-hc-with-heterogeneous-multiprocessing-hmp/&refURL=&referrer=#41b21a7847a5, 2013 [31 Mayo, 2018].

[25] S. Smith. "The scientist and engineer's guide to digital signal processing". San Diego, CA, USA: California Technical Publishing, 2011

[26] R C. Gonzalez, R E. Woods. Digital image processing. Upper Saddle River, NJ, USA: Prentice-Hall, Inc, 2006

[27] P J. Burt, and E H. Adelson. "The Laplacian pyramid as a compact image code". *IEEE Transactions on Communications*, vol 31, issue 4. 1983.

[28] P. Alvarado. "Procesamiento y Análisis Digital de Imágenes". Internet: http://www.ietec.org/palvarado/PAID/paid.pdf, 2017 [31 Mayo, 2018].

[29] A. Kaehler, G. Bradski. "Learning OpenCV..º'Reilly Media, Inc.", 2008.

[30] A. Kaehler, G. Bradski. Learning OpenCV 3: computer vision in C++ with the OpenCV library". O'Reilly Media, Inc.", 2016.

[31] "Spectral Measurements (Part 2)". Internet: http://www.ni.com/white-paper/3995/en/, 2015 [31 Mayo, 2018].

[32] W. Zhou, et al.Ïmage quality assessment: from error visibility to structural similarity". *IEEE transactions on image processing* 13.4 (2004): 600-612.

APÉNDICE **A**

Abreviaturas

API	Interfaz de programación de aplicaciones
EVM	*Eulerian vídeo Magnification*
FFT	*Fast Fourier Transform*
MEV	Magnificación Euleriana de vídeo
DSP	Procesamiento Digital de Señales
ROI	*Region of Interest*
SSIM	*Structural Similarity Index*

APÉNDICE **B**

Aclaración legal sobre el uso de datos personales

De acuerdo con la Ley n.º 8968, publicada en La Gaceta n.º 170 de 05 de setiembre de 2011, los datos biomédicos reportados en este documento fueron tomados con el consentimiento de las personas involucradas. Además, a parte de este documento, todo registro, en cualquier formato escrito o digital, fue destruido después de su utilización en este trabajo. Asimismo el manejo de los datos reportados queda a disposición únicamente de un especialista en ciencias de la salud.